1.3.05
5TM

20 : In t. interest of "diversity" we may weight wts some power (<1) of their pc's for t. entire QA corpus ("Diversity" means access to subtrees of a great variety of cands.)

I could try ~ 4T325.31-.32 : i.e. wt. $\propto$ no. of QA's worked w. "acceptable pc". T. "acceptable threshold" for each problem is part of t. problem defn. — in which case S-induction is very much like d-induction.

Hvr. while Zivmc is a very general universal p.f., I'd like methods of S-ind. soln. for various other common forms of S. functs: like Gauss from induction (using "arby context sets" for induction ... & other compl. S. funct forms, → 3.00

.06

10: INEPT

SN On Inept Monte Carlo Lsrch & t. St. Petersburg Paradox.

Suppose t. trainer knows t soln (to a INV problem) has $pc = p_0^{\circ}$ & $cc = c_0^{\circ}$.

Then if we do m.c. carlo trials w. t. p(prob) of a trial $\propto$ its pc & we are able to estimate its pc, so we can spend up to $cc_i = k\,pc_i$ on trial i. We slowly ↑ k as t. srch progresses.

⟨My impressn is that one spent an inordinate amt. of time on hy pc cands that didn't converge.⟩ But still, what is expected time to soln?

If k is some function of t "trial no." i, ($\equiv t$)

To get to soln. $k\,p_{c_0} \geq c_{c_0}$ so $k \geq \frac{cc_0}{pc_0}$. also, proby of soln at such times is pc_0.

So we expect $\frac{1}{pc_0}$ trials after $k \geq \frac{cc_0}{pc_0}$.

20 For a given value of k, a trial takes expected $cc = \sum (pc_i \cdot k) = k \sum pc_i \cdot pc_i$

$\sum pc_i^2$ is usually $<<1$ but can be "~ 1." So after $k \geq \frac{cc_0}{pc_0}$ (i.e. its large enuf)

we need $\sim \frac{1}{pc_0}$ trial costing $\frac{cc_0}{pc_0}$ each $= \frac{1}{pc_0}\left(\frac{cc_0}{pc_0}\right)$ which is $\frac{1}{pc_0}$ time t CJS, $\frac{cc_0}{pc_0}$

$\frac{cc_0}{pc_0}$ would be ~cc needed for regular Lsrch.

Now that's t. time needed after we get k large enuf. How fast ($\frac{c_{2n}}{}$) should k increase w. t? Tho t. problem may not be easy to solve, for whatever strategy is used,

This kind of Lsrch is $\sim \frac{1}{pc_0}$ slower that regular Lsrch.

If there are few hy pc cands $\sum pc_i^2$ can be $\ll 1$. 1

30 Actually $\frac{1}{pc_0}$ trials costing $\frac{cc_0}{pc_0} \cdot \sum (pc_i^2)$ each or $\frac{cc_0}{pc_0} \cdot \frac{\sum pc_i^2}{pc_0}$.

$\sum pc_i^2$ can be quite small: Say $\tilde{pc}$ is t. largest pc, & it is quite small (but $>> pc_0$).

$\sum pc_i^2$ maybe $\approx \tilde{pc}^2$, and $\frac{\tilde{pc}^2}{pc_0}$ could be <1! If there are $\frac{1}{\tilde{pc}}$ cands of $pc = \tilde{pc}$ then $\sum \tilde{pc}^2 = \frac{\tilde{pc}^2}{\tilde{pc}} = \tilde{pc}$

& are $\frac{\sum \tilde{pc}^2}{pc_0} = \frac{\tilde{pc}}{pc_0}$ which is prob $>> 1$.

But even if t. time spent after $k \geq \frac{cc_0}{pc_0}$ is not very large, t problem of how k is a funct of t, would have to be solved.

1.4.05

BOOSTING.00

STM

2

"BOOSTING":

Say one has a large source of not v.g. problem solvers:

T. problems have unmanned or partially solns. The may have problematic solns over a set of possible/finite string replys. If X_i are the problems & Y_i the solns, then a PSM will map the X_i into V_i we want $(Y_i - V_i)$ = difference (error) to be as small as possible for a (linear combin. of the psm's.

T. Boosting idea is a successive approach method to get good wts. for a large no. of PSM's that are individually not v.g., but very good in a properly wtd. combn.

T. method used, gives more wt. to those problems for which the past solns haven't been too good. So at all times we have a wtd sum of PSM's: & we introduce a new PSM that is supposed to be interested precisely in the problems that the old wtd Σ didn't work so well on.

[It reminds me of my "Problem Pool" method in which cands were tested on the hardest problems first — but I'm not sure there is any real relationship here]

→ Google do Boosting Algorithms to get set of papers.

The papers in Boosting are in D:/PS: Dated 1.4.05.

One long 34pp (.ps) paper is on "Text categorization" dated 2000: (My Name is "Boosting Long Text Categ. Good Bibl. .ps")

Text categ. work is of int. to me: I want to see what kinds of (abstractions & concepts) they use to build "learners" (= simple PSMs) — so I can use their Bibliography.

As for the "Boosting" method itself — it's a way to solve a certain kind of optimize problem — that I may find useful. I think it is not very economical w.r.t. SSZ, because it doesn't re-optimize entire set of PSM's each time.

(Boosting Density Estimation 2002) (PS(1.4.05): I am mainly interested in density estimation, so this maybe a useful paper.

What is unclear to me: They have this New PSM and its is supposed to try to solve a certain set of (the most difficult previous) problems. How do they modify the PSM to solve a certain subset of problems? It must be modifiable! — I guess its a "trainable system" that has several adjustable params so it can "fit" a correspondence a certain (perhaps small) set of (cases examples). So each PSM is a little "learning system" It is a "micro learner". — thus, one would end up w. "little learners" that could extrapolate for a cases each. The total bc of the system would be very large! So "poor generalization capability" — but in some of these systems the total bound seems to be bounded! How does that come about?? I will need to read those papers more! . . . .

Trouble is they tend to not discuss the "micro learners" in much detail.

An imp point is that "Boosting" is not just a method of getting optimum wts for a set of predn methods. It seems to involve choice of "weak learners": So I really don't yet understand it.

Scan: PS Density Estimation By Boosting 2002.pdf for perhaps clearer discussion of points I'm confused about. (scan 20)

1.4.05

1.04.05

STM

3

~~2 days ago~~ Arthur; Prof Barnes,

20:1:08 : Perhaps the jumping from a set of ~~OJ's~~ that work ~~many~~ problems, to OJ's that work only (many) problems is ideas "Escape" in PPM! (I discussed this sometime ago! [320.31-321.03])

Anyway! T. main diffy. probs for "Strong A.I." :

1) writing good TSQ: author for Lsrch or "ProblemPool" & its variants,

2) ~~Having~~ In Lsch phase of TM say "phase 1": Having found several OJ's for various corpus sizes: How to efficiently Srch for OJ for larger corpus. How does the new srch have access to recogs that were found in earlier srchs? How do we avoid testing entire past corpus, when we make (for our new Cand (what appears to be)) a small(ish) change/part successful OJ (of smaller corpus). (How can we know what part of past corpus is effected by a modification of OJ?

3) What are some good ways to represent S-functs? TM should be able to decide which to use in any particular problem. Also how is info on effectiveness of various S-funct forms transmitted to srchs for different S-funct forms? → (6.20)

4) What are some efficient ways to update the Pd that Guides Lsrch (or any Par srch)? How can we find good concepts (≈ subtrees or sets of subtrees) for induction in PPM.

Gen discussn: Suppose we have an "Adequate TSQ" for Lsrch on the QA problems. By definition, this means that "Lsrch would work". Hvr, the CJS can be nominally very large, if it is actually $P_C \cdot CC_0$, where CC_0 is CC needed to test entire corpus. By using the Recog. functs, we can find sized corpus needing testing. The analysis, (by the trainer) of the TSQ must include discovery of those R functs. (Remember that T. P_{C^*} is an upper bnd on the time spent on Cands.... we may reject Cands in much less than that time. Hvr, if there are some Cands of hy PC that do not converge, all of TPCs will be used on such Cands.

Still, the R functs could considerably reduce $P_C \cdot CC_0$ for the Cand that the trainer has in Mind, and it is this CC that is the "CJS". (This is by no means "Solves the problem" — let's go on to another point; for each QA, the trainer has a threshold of PC to be assigned to the "right answer". Hvr, there is the possy that the "true" generator, M, of the observed Ai's will occasionally generate an Ai of very low PC .... much less than the trainer anticipates!

On the other hand, remember that reduction is an OE prob... we just try to get as good a OJ as we can in the available time. This would seem to mean "no acceptance threshold" for any particular Ai (so it's not really much like a FNV problem!). We want to maximize the PC of the corpus within the available time.

A Reminder! That the idea of a TSQ is that, after solving the early problems, TM has within it various Cands that have vastly reduced the search time for finding solns to subsequent problems. These Cands can be in the form of definitions also in addition to collecting of data on

STM

10:3.40 [redacted] t.p.b. of [redacted] "Tokens" that follow various kinds of "contexts"
Seems to me that the TSQ approach to TM has to work! Even my model of how TM (expresses) stores
solns to problems (in terms of useful concs) may be significantly distant from the way humans
do it. This may give a huge (but I hope manageable) "constant factor" between human & TM's, in solving problems.

To start off, we can let Trainer derive TSQ's that don't need "R" funcs to be computed by TM
These will be separate TSQs each w. a distinct R" --- a distinct soln by each.
They will each have a distinct (set of) O^j functions, & will be trained on
distinct corpi, but the scoring data used for searching for solns will, to a large extent,
be shared ... Also in searching, TM will know which O^j's w. corpus of
solns were used for which R's (i.e. w. O^j will be "coded"/"indexed".

So: to start, we may ignore the R problem, & the problem of testing entire conds on
the entire QA corpus ... because that corpus will be small.

Then: We will be generating O^j's and testing them on QA corpus; How to do it testing
economically: They will depend much on forms of c-functs.
One way to think about it: we want best evals of $O^j(A_i|Q_i)$ (= a pc.).
Expressing O^j's as a 3-tume would seem to be poor in terms of cc, but nice in
terms of Universality. If we make O^j an arbitrary (perhaps 3a) real
funct of A_i's Q_j's, then we have to normalize, otherwise it is easy to maximize w. product.
To normalize, will have to consider many possible values of A_i ... which sounds expensive)
and [redacted] very inexact. ...

23; 385.34 -AO
A pseudo 3-tume way would be to have several O^j's: each O^j corresponds to
a distinct [redacted] R (ruling in part of c-3tume. Say we have 4 or 8 or 16 or .... 2^k for some k.
They are trained on all QA problems ... (which is a true approxn of the 3tume).
We then use "Ti closest" "Edit distance" between one of the outputs of the "target" A_i's.

26

I discard .23-.26 before, but I seem to forget the "details" - here there are no
complex "details"! .23-.26 is a correct approach. The main deviation of (.23-.26) from
the previous 3tume w. T. R in and R is that in this last case, R can take values that have distinct
wts. --- i.e. not all R values are of = wt (as in .23-.26). In (.23-.26) the only way to get more
wt. is to have duplicates or approximate duplicates [redacted] as many of the 2^k outputs
Actually, it is possible to make an exact "approx" of 3tume via (.23-.26) by suitable exact
duplication: Say k = 4: Then if an R in 3tume had a wt. of W (w ≤ 1), we would
have $\frac{w}{2^4} = \frac{w}{16}$ duplicates of that A output. (in particular)

Here, .23-.26 slows down the testing procedure by a factor of $\approx 2^k$. TM could learn
to find the min functions of the 2^k to set ± each A_i [redacted] — which might significantly
reduce cc's by Quick Abort(;).

.00 :4090 : In line w. .23–.26: Perhaps have ~~very~~ many O^J's ~~that~~ each of which are fairly good on as many QA's as possl. For a new QA – to do this prodn, simply wt. them all ~~...~~ acc t. no. of QA's they do well ~~...~~ on. ~~Perhaps only count # exact~~ hits on ti's.

— Maybe more wt. for O^J's that had hy scores on "closely related" Q's.

This needs much Theoretical analysis!

□

Suppose as .00 ff we have ~~a~~ soft Recog. functions and many O^J funcs & we have the R functions give a wt. to each O^J for each possl QA. We would then get products for all QA's & we could get t. products which we want to Maxz. — But what is t. pc of t. dcon? ~~...~~ Many O^J's are diff so we can't simply add their pc's.

0 In t. part of TDS12 report on Rec funcs, there were a bunch of Recog. funcs, together were able to decide which O^J sets should be used on a given Q. These Elementary R functs were d funcs & they combined to form a d. funct on t. input Q's.

Note that its much better to have d-R funcs (acqu.) because we cut t. corpus up (determinately) (sharply) & are able to save cc in testing w. certainty.

In using S-R funcs, (stochastic acquisition) we may be able to use ~~...~~ probilistic testing on randly

17 Chosen [illegible] Q's of t. corpus. (~~Also~~ ~~Better~~ → Chose R. ~~...~~ trials in strict "d.ffy" order).

.18 .00 ff is closely related to a recent "problem pool" method, in which I would try to find O^J's that worked as many of t. probs as possl. ... picking various subsets of t. ~~...~~ Q's, to "specialize" on. There, t. idea was to use t. partial solns (t. ~~...~~ ...) as a source of "parts" to (i.e. "working corpus") eventually find an O^J that solved all of t. Q's satisfactorily.

In ~~...~~ .00–.17, we would be satisfied w. a set of O^J's – (no individual one that was able the feasible to work all problems) and an R funct to assign Q's to these O^J's (or O^J's to these Q's).

Say we only had 2 O^J's & an assoc R funct to assign pc's to Q's — as to which of t. 2 O^J's to use.

.27 ~~So~~ We would then code t. corpus of n Q's in 2^n difrnt ways, corresponding to difrnt. Choices of O^J's for each problem. If, for a particular Q, R assigns P_1 to O^1 & P_2 to O^2,

.29 & (O^2 / O^1) gets (P_2 / P_1') for t. ribe answer, then t. system assigns $P_1P_1' + P_2P_2'$ to t. correct answer.

.30 — For t. ~~...~~ entire corpus, we multiply all of these ~~...~~ sums for t. individual Q's to get final pc for t. corpus, } These very conservative estimates.

31 — But we also have to mult by pc of t. R funct & t. pc's of ~~...~~ O^1 & of O^2.

32 · It would seem that if O^1 usually gets small P_1 for various Q's, that we should not completely } Seems to be impt. — Diffl Q

33 "charge" it its entire pc of dcon? ← 6.02

34 A (Falacious) argument against ~~...~~ (.27–.31) is that as t. wts. of O^1 (i.e. its P_1's) get smaller & smaller,

35 we have a discontinuity at P_1 (for all Q's) → 0, since t. pc of O^1's dcon. becomes suddenly unnecy.

A counter agt. to (.34–.35) is that we always have ll codes in which O^1 is not used at all, so they don't have t. penalty of t. pc of O^1's dcon cost.

.38 ~~For every~~ Say we have many O^J's: for every subset of them, we have pc of t. corpus w.r.t. them mult by their pc's of creation! Mult by pc of creation of assoc. R function that assigns O^J's to Q's!

1.6.05

5TM6

.00: : In general, this last ~~[illegible]~~ R funct may differ for each subset of O^j's — as will its pc of creation.

.01 If there are n O^j's there will be 2^n possl codes (including the identity code).

.02 (5.32–.33) Because of ll coding and adding of pc's (as 5.29), an additional O^j may not be such a big expense (pc of creation-wise). Also, say O^1 was used for all but one Q, in which case O^2 was used. T. pc of O^2 creation plus (the pc of creation of the) opt. Recogn funct. would, indeed, be needed to justify using O^2 on that Q.

T. moral is: It is much better if we can do it w. only one O^j, but if not, then use > one O^j.

.04 Also, t. additional O^j's need not be very expensive, if they share ways to handle contexts.

So perhaps 5.18–6.09 may be a viable way to use many different O^j functs: There remain Q's:
1) What is form of S-functs? Would be good to be a "Turing Complete Univl. set" (w.r.t.) t.
2) What is good way to do srch for such functions? Any good way to avoid ~~[illegible]~~ verifying the entire corpus, whenever a new O^j is found?

Re ② Srch: A not bad way is in "Prob. Pool" Mode: to try to find O^j's that work ~~[illegible]~~ many Q's well. I wrote a fair amt. on that approach (4+325.30 — 5T1.04 is relevant, ~~[illegible]~~; 320-31; 311.30–.40 ← Base Idea; 314.30 ← Impt; 315.15; 316.03; 5TM5.18ff). One diffy was how to wt. Cands in corpus that solved varying nos of problems w. varying pc's, so one could properly (rn) from them — say viz ≈ PPM.

v:3.16 1.7.05 ~~[illegible]~~ On representation of S-functions: Some problems/goals: ① We want it to be "easy" (low cc) to get pc of an object: This is for testing Cand. functions ② we want it to be easy to generate objects of hy pc ··· say in ≈ pc order: This is for prediction: it can be much slower than ①, hvr, since we do ① much more frequently (in (de)TM) ③ T. method of generating t. S-functs: It should be universal & it should be easy to compute pc of each S-funct. ④ T. representation system should be match w. t. "Regy detection Scheme" (e.g. by PPM) ··· so t. scheme can easily find regys in functs of that language"

3.2: ~~[illegible]~~ SN On representation of functions (Both S's): Instead of sum(A3, α) where α is some expressn: keep α as a token; so all operations are in t. simple form of linking tokens: how they are to be combined. So all pgms are of t. form A, B, C, where A tells how B & C are to be combined. This is supposed to make it easy for PPM to recognize regys. So a pgm would be a series of definitions of "Non-Terminals". The would normally be written in an order so it could be easily compiled into a tree — (but maybe not for recursive functs).

Hvr, I'm afraid t. regy. would not work ~~[illegible]~~ as is: Its really ~~[illegible]~~ that t. sub trees be recognized — & t. regy system doesn't do this in t. "standard way"!

There is, hvr, t. possy that if we use a system that would not recognize regys of a certain

STM ✓ S-Funcs .14

00: Sort, that the system would learn to "get along" w.o. using these recogs — by perhaps creating functions that don't "use" them (?? — unclear on this pt... whether it's useful or likely!)

SN Generalized of "Context": A context is a set of suffixes that reduces entropy of the corpus following it. Previously we had considered only the immediate "suffix" of a context. Here we consider any pgm that compresses the subsequent corpus. If a certain context makes it likely that the next symbol to follow would be "t", then this is a (potentially) useful "context." E.g. it is a sub-generalization of my Gruval generalization of context... as the entire previous corpus does now knowledge of it, influenced our prediction (≡ compression ≡ entropy) of the future. (As opposed to "MacroContext")

ON INV problems: Use of L search for soln. of INV probs seems to work the probs by "direct algm" — that humans usually convert them to OZ problems & solve them as OZ probs. But a good regular INV search would be able to find the best way to solve the problem... If it would convert this to OZ, it would do that. Furthermore, we can design TSG's that encourage/enable that kind of problem soln. — So L search can solve INV probs "optimally".

A trouble is: In Phase I the OZ solns are not very good. The OZ methods & datums are not V.G. In Phase II we do much better.

7 S-First Representation: For D.P.'s on reals, a very common way is to have 1 output be the μ & σ of a Gaussian or other D.P. (say Gamma D.P.). Since these kinds of funcs are the main kinds needed for phase 2, this means we only need to do this kind of induction in phase 1 (& we need phase 1 at all!). Actually, we only need μ's as output — the σ can be optimally fitted to the data — as in usual "time series regression"

1·22·05 For S. funcs w. string output: S. CFG's are not bad. We can modify/generalize them in various ways (e.g. transformational grammars). Actually, many different types of grammars have been proposed for English. We can ideally devise a grammar that can generate all the known grammars; plus any others. We can generate a Universal "grammar, grammar" — but the main idea is to find a grammar grammar that will easily generate grammars that are known to be useful.

S-context sensitive grammars are perhaps universal & quite powerful. "Universal" in the sense of being able to simulate a Universal Turing Machine. They are also nice because they are an "add on" to S-CFG's so we can have FTM learn CFG's then probably learning for CSG would not be very different (?!).

The only way I know to discover S-grammars is BU — i.e. hill climbing — starting from a (very expensive) grammar that fits corpus & successively improving it.... usually greedily, but may be not always ☺. → 8.00

10:7.40: Anyhow, perhaps very likely way to generate s-strings is by a relatively short seq. of operations — logically defined, but w. some uncertainty in some of the operations. This would seem to be the way we answer questions that require string replys ....

Q's themselves can have strings a/o numbers in them.

Distributions of this kind can arise from "case" statements becoming not 100% accurate — a having PDS for each case, rather than deterministic results.

We'd like s-trees to have non-zero pc for all string outputs ... but we may avoid this need in a lazy way:

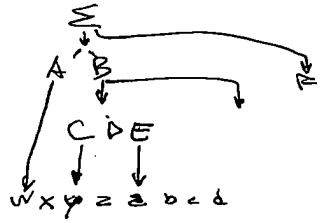
Given a corpus, we fit a s-function to it, so all pc's are chosen, & then we find params to give the whole model max pc. The model can then predict. It can give pc=0 for certain future events, however. If it does, actually do so, we will have to completely discard it & find a new s-model.

Simplest examples been seq. w. h symbols: we have a simple k component vector that determines choices.

Next = PPM & various Algms to decide on what context to use to get a k-vector for choosing next symbol. The Algms can do 11 probs, so we added them (in a wtd. way).

Next, we do operations on the past to get pd. vectors for the next symbol. The operations on the past give something like contexts that are nested.

SN | In CFG's when we generate a string: Let us look at the "derivation" I think each terminal symbol in the final output has a "linear" history of choices — so perhaps we can usefully use ~ PPM to predict the symbols — using that kind of "Regularity". Lets look at the derivation tree of a sentence:



The latter 'z' has context (∈ BΣ) or ΣBE.

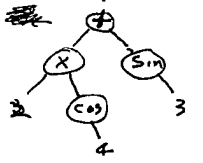
w has context ΣA

so z & w have common context, Σ.

Y & z have common context ΣB.

We can do some kinds of contexts in constructions of functions (which have a CFG) (Unless they are recursive(?))

Some simple function trees:



4 has context +, X, cos.

2 " " +, X

3 " " +, sin.

These are not, tho, the contexts one gets in a Polish or RPN expression of the functs.

+(X(2, cos(4)), sin(3))

4667 SMC
Stalpon line >5Min/hr.

DO : Is there any good (cheap) way(s) to implement this kind of conduct?

I want to include my method of evaluating the utility of a particular improvement of the search algm. I had a nice simple equation for the efficacy of a method, but I have not been able to find it! The eqn appears around line .33 on left side of page.

It may be possible to recreate it: Maybe this: say I have several such methods of various speeds & strengths. They have distinct ΔE profiles w.r.t. CC spent. Each time we use a particular method, we get a drop, which if our corpus size & quality, so each method helps itself ... But the analysis ended up w. some spurious factor that was very reasonable! You do find that analysis! I think I made refs to it several times in TM notes.

Was it in the form $\frac{\sigma}{E_{MX} - \bar{E}}$ was discovered?

SN on K. Szlotowicz & J. Schmidt. Probabilistic Incremental Program Evolution (PIPE) 1997 [ps. 7/26/04]

4TPM (81.17 more common.
185.00-.03 more serious commute.

To generate pgm, they use fixed, standard trees: k types of functions, l kinds of literals so each branch pt has k+l possys. h is the max arity of all functions.

There is a "common complete, infinite tree" w. n branchings at each node. This tree has simple coords of each node. When a function is created, each of its nodes maps onto a certain one of the nodes of the "common" tree. — This enables us to compare/"correspondence" different functions — so we may learn.

Main kind of learning: we probabilistically construct a population of candidates using probys at the nodes of the common tree. We take the best cand of each generation, and move probys at nodes to make the best of Generation more likely. } for Generation Based Learning.

Also we occasionally do the forgoing using the best cand of all generations so far" — "Elitist" learning
Also they use a certain mutation rate "of probys in "common Tree" to get learning. Also; but the mutation algm is very strange! 6.2.1 of Document

I certainly haven't gone into this paper very deeply: But my impression is that the method used to learn, is not very good. Their associating learned probabilities w. each node of the "common tree" is amusing — a mite actually lead to good functions!
That they only learn from the best cand. in each generation, seems wasteful, hmm.
In General, it might be worthwhile to go over this paper more carefully.

The probys at the nodes of the common tree correspond to the probys in fixed length Economies of PBIL (Population based Incremental Learning) (Baluja & Caruana, 1995) — they also use copy the learning methods of PBIL.

Grammar Model-based Prgm Evoln. Doc 2003 · Guyon Australia - mainly Canberra (!).
This seems about the same as what I had in mind! The use of a SCFG for summarizing a population of function trees, is very good. Sub-trees are recognized in the grammar discovery process. [Note: useful common]

Hyvr p4 col II, ¶27: They're doing this close to ordinary GP. They keep selection, population to keep, etc., is not v.g.

My $\frac{G_c}{G_{max} - M_G}$ would probably be much better. They just use 'top ½' of the population to create the S-CFG; 50% is unlikely to be optimum.

Q. big Q is whether to use SCFG or some kind of enhanced PPM. This last makes it possible to update the Grammar very frequently, which does the "$\frac{d\,G}{dt}$". Also, depends on how good SCFG's they can get in "available time". At any rate, it would seem that SCFG's would be able to discover any common subtrees in the set of trees.

So my impression of the paper are:

1) [.07]

2) PPM may be better than SCFG -- but I would stick to one if. If they have a usable SCFG discovery prgm -- try to get it from them.

3) They don't work the nice problems. They just use problems that others have worked on to show their methods superior. They don't do TSQS.

4) They don't give system ability to "watch the fitness evoln".

5) Phase II in which the constructive search of S-models is regarded as a problem for the system to solve: It has a universal model accessible to it.

Think about simultaneous LSTM & BU search, using S-CFG for PL parsing. Doing PC order is difficult, however.

HA! We can do simult LSTM & BU search on S-CFG's! (or S-CSG's);
It is "easy" to list trees in PC order: Use ordinary LSTM search with list of waiting all paths until pc gets below Threshold = Th"; Then do Th := Th/2.
My memory of it was that it was somewhat like "Arithcoding".
Main idea: That given a pc threshold, it is always possible to list all trees w. pc > Th.
It may be like this: one explores a tree of decisions until either a tree is completed or until pc < Th — at which point one backtracks, (perhaps increasing pc), then continues.
Alternatives: backtracking (which is time-efficient, but memory intensive?), start over as "root" whenever you complete a tree or get pc < Th.

What the algm does, is simply an exhaustive depth-first search of the trees, in which the termination of a path is either a leaf or pc > Th. It keeps track of the pc level of each node so it can update Th properly after a "backtrack". I think a stack can be used to keep track of where one is in the net, & what the pc's of the nodes are. (A possible serious inefficiency of this algm is that it can spend much time going out on branches that the pc, soon becomes unlikely that it will have enough pc to complete the tree.) — Here, if one is doing LSTM, one of the three terminations criteria is not pc < Th, but is cc > T · pc. This last results in

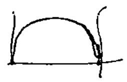
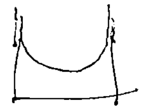
: In discussing Advantages of ALP (or Stoch Comp) over MDL — I said that the diversity of Models was useful in dealing w. new dfft. problems — so we tended to use ALP rather than MDL in A.I.

Another area of use is in GP: If we use a single grammar to Model Population, this is MDL: Hvr, if we use several Grammars, we have much less likelyhood of getting stuck at a local optimum — which is called "Premature Convergence" in "GA talk".

If we are using S.C.F Grammars to Model our population, we can get diversity in our models, if we obtain them by "early splitting". — What this means: In finding a good Grammar, we start out with a grammar that fits the data, but it gives low probability to the corpus. To obtain better models, we try modifications of the grammar ... Chunking and Merging are two common types of modification. Often, at a point in the calculation there will be several possible modifications that will all give better fits to the data. If we follow these branches and continue improving them, we will tend to end up with several grammars that are quite different — giving the desired diversity of trials.

Poss. Inadequacies, Bottlenecks in IAP Model(s) of TM.

1) A MTM — QATM could be v.g. using PPM, (substring context augmented PPM), S-CFG, S-CSG
Poss Bottleneck (a) Dfflt to write TSQ (b) It can't do self improvement of its induction search algm, since this is an S-induction problem. The system s. can't ever be very smart (unless the search algs I've listed are "good enough") → Hvr. see 13.00 !!
Also TM would probly have to learn Markov (agic's how to search (?)).
The This MTM could be a "small TM" project. Could give useful experience using PPM and PPM as CFG insrch.
2) I'm not sure cfg's can recognize subtrees (subfunctions) occurring w. high unusually frequency. ← BUT see 37. I think each NT's always represent nguts only. It may be that a scfg isn't any (or much) better than PPM! Perhaps they can work w. smaller ssz! Also regys detected by CFG could be essentially distinct from those recognized by PPM. → 12.00 → 12.00
29
3) If the lang for expressing Functions (≡ cands) has definitional facilities, then PPM can be v.g. for noticing unusual freqs & unusual freqs w. certain contexts. Still, even after a function has been "defined", it will not recognize $F(\text{don't care}, B)$, where "don't care" is of arby size. — See .37
4) Perhaps best search method would be sub-tree context PPM; but implementing it would be non-trivial & possibly slow. Normally speed & cleverness can be traded off — but certain kinds of regys may not succumb to that sort of "Boosting". → 12.20 Space
37
35
32. DGF ~
$F(\sim, \alpha)$: "$\sim$" means "Don't care". , so $f(\sim, \alpha)$ is an ngust. So since cfg's can recognize some ngusts, maybe they can "recognize" $f(\sim, \alpha)$.
Note that "$\sim$" can be a legit N.T. i.e. the set of all finite strings: cheap to define! May be included as one of the "free" definitions.

$p^{\frac{1}{2}}(1-p)^{\frac{1}{2}}$  $p^{-\frac{1}{2}}(1-p)^{-\frac{1}{2}}$ 

precorpos $(1\frac{1}{2}, 1\frac{1}{2})$ — precorpos $(=\frac{1}{2}, \frac{1}{2})$

20 (SPEC 11.29) . We could compare PPM & cfg in compressing a set of unordered strings. Note that PPM is normally not used this way, so we have to be careful in the comparison: PPM can be used for unordered strings: I did devise a vg. very fast way to do it. We just use a special "ending" symbol, (Δ, say) betw. strings, & treat the whole thing like a sequential corpus. We can get pc's for the first symbol in any little string. It has context Δ only: ~~somewhat~~ In fact any context string is allowed to go past "Δ". (I may have already written much on this!) I think this is easy to arrange when we put all pcs of the corpus in lexical order. For the initial symbol in a string, we may have no context — just frequency.

0 For the final symbol, we usually have much context, and for the Δ following, (which we do need a pc for) has much context.

For these unordered strings, we can also use PPM backwards & seem to get same compression.

Hvr. Note [11.37]! CFG's may be able to recognize $F(\alpha, \alpha)$ (don't care) the "AND" of the sets $F(\alpha, \alpha)$ and $F(\beta, \alpha)$ is $F(\beta\alpha)$.

Hvr. in CFG's we do OR (union) of sets, but not AND (intersection) of sets. There is some ① above proof of the "α" symbol, hvr.

20 : 11.35 : 5) For QA induction, I considered using $\prod_{i=1}^{n} P(Q_i A_i)$ as a (partial) score, because it was easier to devise schemes for ass'ns of form $(Q_i A_i)$ rather than devise Pd's for $P(A|Q)$. We can do this for all forms of Q,A ... (string a/o numeric). This makes it possl. to start w. poor grammar (y. A.M. grammar) & hill climb by "chunking & merging & pretty other operations". T. trouble is, this is for a CFG & is not universal. T. CSG is universal, but I'm not sure as to whether it's an easy way to write things in.

Ordinary UMCs that stop after they print an output. all define s. grammars, but they are not practical ways to define a P.D.

30 **One Possl. way:** Consider the pgm with a "case" decision on get of 1 integer 1 to n ; its output is $[\alpha_i]_{i=1}^{n}$ α_i are strings, say.

This pgm has a certain length to decod the unordered set, $[\alpha_i]$ (including the pc's of choices)

How can we modify the pgm to ↑ the total pc? Are there good trial steps to ↑ pc?

The only probable part is the 1 to n integer; we can lower the current pc & try to ↑ the pc of the set $[\alpha_i]_{i=1}^{n}$: Ordinarily, this would not easily extrapolate to a new element.

Another reasonable possy is 13.18: perhaps close to how humans do it?

→13.00

STM:

00:12:40 6) Would it be possl. to get Phase 1 [as a s-function IND solver] to work on improving induction such for d-induction?

01 The corpus pairs would be [(the sequence of accepted O_i^j's up to the i^{th} problem), (the O^j for the $i+1^{th}$ problem)]$_{i=1}^{n-1}$

02 It then uses the (" n ") to be extrapolated,

It would seem that in theory, This is a legit QA s-induction problem. To work it, TM would have to have an adequate TSQ! So Phase 1 is capable of SI.

Hvr, in Phase II TM has lots of other OZ problems, so the training situation is, presumably, much better. [ALSO (N.B.) Phase 2 UNDERSTANDS optimization Much better than Phase 1 does.]

0 Also Note Phase 1 can improve its s-induction algms in way used that for d-induction

To restate .01 - .02 The QA's of corpus are

$Q_i = (O_1^j, O_2^j \cdots O_i^j)$; $A_i = O_{i+1}^j$; The problem is: Given Q_{n+1} to get a pd for A_{n+1}.

TM is given $[Q_i, A_i]_{i=1}^n$

So, to start, we will use (perhaps augmented) PPM or S-CFG to do this.

But in solving this s-induction problem, TM is able to use universal instruction set, so this is still the diffy of 12.26 ("5)").

8 7) A possl. soln. to a pd on unordered set of strings (& to the QA (both strings) problem).

0 Early "AI" problems (Simon & Newell) were of the form: Given initial state of system Q_0, set of sequential xforms $[F_i]$, xfer Q_0 into A_0 using the set $[F_i]$, and there is some metric on how good the seq of xforms is: The problem of $A_0 = \min$ function of Q_0, using a set of (universal) computer insts, is of this sort. Their "GPS" was a set of heurs for this problem.

My take on this was the OP-OB algebra: Here the trainee learns to "Chunk" various Ops & various Obs. to make the problems easier. A combination of OP followed by an OB followed by an OP is "feedback". (Action - observation - Action = Experiment),

At any time, there will be a set of problems that the trainee is "Good at" & other problems will take longer, or get low scores, or be insolvable.

10 The technique of learning may be simply "Chunking" & possibly merging. Newell & associates had a "SAIL" system that they purported to be a general system for A.I. — it was rather complex.

The Big Break thru in GPS was the idea of "Difference obs" & assoc. w. them Ops to reduce that difference. Instead of optimizing a single Goal, they had a Vector Goal & had difrnt ways to deal w. each Goal component.

→ See 16.10

∴ $\sigma^2_{obs} \cdot \frac{n+1}{n-1}$ for ~~to~~ Homo. induction: 1 multiplicative coef.

≋ $y_i = a x_i$ + random gaussian noise. " N_i "

$a \to S \equiv \sum_{i=1}^{n} (a x_i - y_i)^2 = \min \quad \frac{dS}{da} = \sum 2\, x_i (a x_i - y_i) \quad \frac{d S^2}{\delta a^2} = 2 \sum x_i^2$.

$= 2 \sum x_i^2 - 2 \sum x_i y_i$ so $\tilde{a} = \frac{\sum x_i^2}{\sum x_i y_i}$

Q is $\sum (\tilde{a} x_i - y_i) = 0$?

$\sum_j \left(\frac{\sum x_i^2}{\sum x_j y_i} x_j - y_j \right)$ so is $\frac{\sum x_i^2}{\sum x_i y_i} \cdot \sum y_j = \sum y_j$? if so then $\tilde{a} = \frac{\sum x_i^2}{\sum x_i y_i} = \frac{\sum y_i}{\sum x_i}$.

$\tilde{S} = \sum \tilde{a}^2 x_i^2 - 2 \tilde{a} x_i y_i + y_i^2$

$= \tilde{a}^2 \cdot \sum x_i^2 - 2 \tilde{a} \sum x_i y_i + \sum y_i^2$

$\tilde{a}^2 \sum x_i^2 - 2 \sum x_i^2 + y_i^2$

$\tilde{S} = (\tilde{a}^2 - 2) \sum x_i^2 + \sum y_i^2$

Best try it empirically for various $a \to \overline{N^2}$ values.
A ≤ Σ present: n = 3.

A = : n =   generate seq; keep running track of $\sum_{j=} x_{i-j}^2$ ... "60" $i=0$

α $Axx = \sum_{j=0}^{n} x_{i-j}^2$ | $AL = \frac{Axx}{Axy}$ → $x_{i+2} = A \cdot x_{i+1} + G$ | $x_{i-j}\, x_{i-j+1}$

$Axy = \sum_{j=0}^{n} x_{i-j}\, x_{i-j+1}$ | Sig ~~~~ $(AL \times x_{i+1} - x_{i+2})^2$

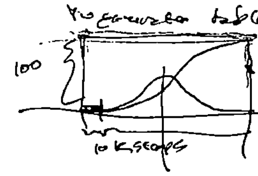
Gaussian noise:

z = 0 : ~~Sqsum~~ Sum = 0
z = z + 1 ; ~~Sqsum~~ Sum = Sum + Sig 1
i = i + 1 : ~~~~ . If i < 1000 then go to α
Sig = Sum / n · (n+1)/(n-1)
Sig =

Initialize: for $i = 2$ to 1000 ($x_1 = 0$)

$x_i = x_{i-1} \cdot A + G$ ← Gauss Noise.

To generate table for Gauss noise.



Test Gaussian gen to see that it has right μ, σ.

For k > 1, the truth of $\sigma^2 \frac{\cdot}{\cdot}$ could depend on t · coefs : whether or roots of c. char polynomial (are all inside unit ○ could be very impt! I could just choose coefs at random, then reject + seq. if it gets too far from Φ — if it looks like its diverging.

Best Drop this for time Being! For All Notes: ~~Peso~~ $\sigma^2 \frac{n+m}{n-m}$ Say that I'm not sure of formulas at present time.

5 TM

MEJ Paradox: 00

"MEJ"

5.00: (8) **A BIG Q:** on [illegible] SIAP 48.02.720 the formula $\frac{\sigma}{G_{max} - \bar{G}}$ is Questioned!

It is indep of threshold if the G are simply the order of merit of cand. So the formula depends much on the G funct. w.r.t. a monotonic function of G — like using "fitness proportional selection" in GA.

It may be possible to use a function of G and a suitable threshold so as to maximize $\frac{\sigma}{G_{max} - \bar{G}}$. Simultaneous opt of threshold and functional form.

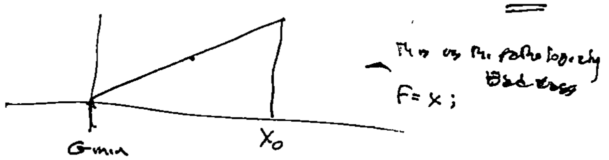
It would seem that the MEJ formula would have to work if G was a measured utility.

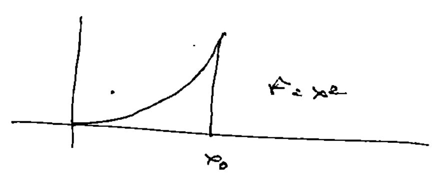
Hvr, the only useful output of MEJ is X_0, the threshold, (or any other way to define it). language corpus used to define the search language. But the output (i.e. corpus or language) depends (perhaps much) on the functional form of G — i.e. any monotonic xform will change the output!

The PSG (or other prodn device) used to get a PD on the chosen lang: would like the μ & σ of the PD to be same as that of the population being modeled, if we use the MEJ eq. for lang selection.

---

look at pictures at SIAP 48.206: Mu suggests MEJ ought to work for that Density profile: Consider $\overline{M_{oj}} = \frac{\sigma}{\bar{G} - G_{min}}$





$\frac{1}{X_0}\int_0^{X_0} x = \frac{X_0^2}{2} \div X_0 = \frac{X_0}{2}$.

This is first moment of $x = 1$ from 0 to X_0

$\frac{1}{X_0}\int^{X} x^2 = \frac{X_0^3}{3}\cdot\frac{1}{X_0} = \frac{X_0^2}{3} = \mu$

First moment of x (to X_0).

mean = $\frac{1}{X_0}$.

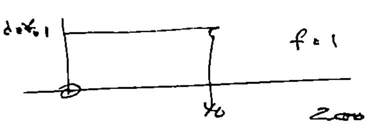
second moment = $\frac{1}{X_0}\int^{X} x^3 = \frac{X_0^3}{4}$

σ^2 = sec moment − (first mom)^2

mean of square

---

uniform distr



Zero moment $\int^{X_0} 1 = X_0$

~~mean value = $\frac{1}{X_0}\int^{X_0}$ = $\frac{X_0}{X_0}$ = μ~~ ~~First moment~~

first moment = $\int_0^{X_0}\frac{1}{\mu}\cdot X = \int_0^{X_0} x = \frac{X_0^2}{2}$

n^{th} moment = $\int_0^{X_0} x^n = \frac{X_0^{n+1}}{n+1}$

mean = $\frac{M_1}{M_0}$; $\sigma^2 = \frac{M_2}{M_0} - \left(\frac{M_1}{M_0}\right)^2$

$\frac{X_0^3}{3X_0} - \left(\frac{X_0^2}{2X_0}\right)^2$

$\sigma^2 = \frac{X_0^2}{3} - \frac{X_0^2}{4} = \frac{X_0^2}{12}$ | $\mu = \frac{X_0^2}{2X_0} = \frac{X_0}{2}$

$\sigma = \frac{X_0}{\sqrt{12}}$ $\mu = \frac{X}{2}$

so $\frac{\sigma}{\mu} = \frac{2}{\sqrt{12}}$ = indep of X_0

$F = x$; Zeroth moment $\int_0^{X_0} x = \frac{X_0^2}{2}$.

n^{th} moment $\frac{X_0^{n+2}}{n+2}$

∴ mean = $\frac{M_1}{M_0} = \frac{X_0^3}{3}\cdot\frac{2}{X_0^2} = \frac{2}{3}X_0$

$\frac{M_2}{M_0} = \frac{X_0^4}{4}\frac{2}{X_0^2} = \frac{1}{2}X_0^2$

$\sigma^2 = X_0^2\left(\frac{1}{2} - \left(\frac{2}{3}\right)^2\right)$ $\frac{1}{2} - \frac{4}{9} = \frac{9-8}{18} = \frac{1}{18}$

so $\sigma = X_0\cdot\frac{1}{\sqrt{18}}$

$\mu = X_0\cdot\frac{2}{3}$ | so $\frac{\sigma}{\mu}$ is constant.

$F = x^2$ $M_0 = \frac{X_0^3}{3}$ $M_1 = \frac{X_0^4}{4}$ $M_2 = \frac{X_0^5}{5}$

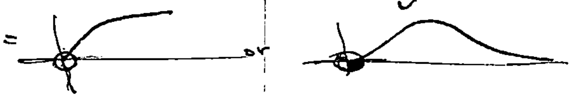
$\frac{M_2}{M_0} = \frac{3}{5}X_0^2$ $\frac{M_1}{M_0} = \frac{X_0^4}{4}\frac{3}{X_0^3} = X_0\frac{3}{4}$

So $\frac{\sigma}{\mu}$ is again constant.

Stop 6.00

STM "Linearity of G" .06

00:15.40 : The moral is, if f is a pure power of x, then $\frac{\sigma}{\mu}$ is constant.
But any other (non-power) relation of f to x would probly give a non-constant ratio..
So perhaps the constancy of $\frac{\sigma}{\mu}$ is a characteristic of x^n only
Hvr. $\frac{\sigma}{\mu}$ would not be constant if f = or



Perhap what characterizes (LINEARITY) : $f(x) = x^n$ is that $\frac{\ddot{f}}{f} - \left(\frac{\dot{f}}{f}\right)^2$ is constant. f-1 = Sf.

06 Re: fitness funct: I have 2 defns: ① G_0 w. prob p is equiv to $G = p_0 G$
② G is linear if the utility of G to G+d is a function of d only & not a funct of G. → This ≈ "utility is ∝ G" } w. proby
I think these 2 defns are the same, but no proof) Perhaps all utilities are differences betw. 2 states

:1340 : 8) On "Phase 2": I had considered starting out on phase 2 (instead of doing Phase 1 first)
This does seem like an alternative path to TM & could be usefully persued in ||.

(margin note: deciding on how to manage coeffs is a funct of SSZ, & a very interesting problem!)

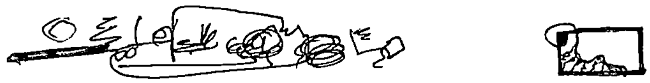
SN King & Cover (Entropy of Printed English w 1978 (IEEE LT)). Using 12 indiv. "experts" to predict next symbol in printed English; 1 expert got much better than others, but mixing the 12 predns w. = wts. got better than best expert!
So Q is, what wts to used (K & C show that wts ∝ pc obtained thus far are ≈ best expert but say pc of entire seq is divided by n k, no. of experts.
A good way to find wts. Chose wts ∋ if they were used to predict each symbol of entire corpus, the product of probs would be max. Getting coefs is a N.L. optzn problem, solvable by Lev, Marq method. — possibly use small sample of corpus to get good approx. wts, then use entire corpus to "fine tune"

29 This is essentially a problem in linear prediction ✓! — we have to optimize k coefs. Hvr, if k is ∞ or just very large optimizing coefs becomes a very [illegible] "Judge". Perhaps the no of coeffs (or equiv) could vary w. SSZ. → 18.00

0 SN In devising a usable S-funct for string output: P(A|Q) : A is a string:
Somehow Q is made to create a grammar from which A is to be created. In general, testing such a grammar is not diff't, because in the TSQ, A's will be of hy pc, so they will be easy to find by Lsrch (trying generative strings in ≈ pc order)
The elements of the Grammar can be a set of functions to be combined.
So O' is a function on Q's that produces a set of strings that define a grammar plus a set of continuous params for that Grammar. The problem was: how to get those params? they have to be a function of Q. — I wrote about this diffy recently... at ≈ Spec 17.00

1.31.05

5 TM



0 : 16.40 : objection to t. ideas : A possl. implementatn. TM categorizes Q_{10} (say) into a set of Q's of t. past. (Recognition functions of early part of ID 51 Aropcs). That set of Q's all had t. same set of primitives & params — woops!, so they would all be essentially t. same Q w.i. D.F. over A's. Not very General) (actually not so bad! — see (.10))

Alternatively : All Q's use t. some set of functions to be combined, but each Q generates a diffnt set of pc's for using these functs in t. grammar.

Doesn't sound much good ! ← 42

A better way to deal w. t. problem of s-functs w. string output : Try some actual cases in which I have a stochastic soln. I want to implement.

0 · One Common Case is t. Baum Seq. Same Q, but a finite no. of A's, More advanced is a continuum of Real A's.

For t. Discrete case, I can postulate t. fixed Grammar model & post hoc, find params that give max pc to corpus.

Say Q is a Context (like in PPM): we have partial matchings for various "partial sizes" of t. Q : How to Mix t. preds of t. various context lengths ?

Perhaps find a way to do t. wts. of (16.20 – .29) "Optimum" wt. assignment.

What about a more interesting (complex) stoch string output ?

Consider text prediction: PPM is one way, but reasoning about what's going on, is another

What about predicting t. course of a story?

19 Would Boolean Belief Nets (BBNs) be useful.? They are a network of probilistic I/O transducers. → 18.07

0 SN on the "Gram Mod Based Prog. Evol'n" Shan McKay, Rohan Baxter et al.

P7 eq (10) : Expectation $\frac{\sqrt{F(\hat\theta)}}{P(\hat\theta)P(D_n|\hat\theta)} \cdot$ [illegible] $e^{\frac{c}{2}} \cdot \left(\frac{1}{12}\right)^{\frac{c}{2}}$

n ≡ no. of data items

$F(\theta) = \frac{n^{c-1}}{\prod_{i=1}^{c} \theta_i}$ $P(\hat\theta) = \frac{1}{B_c(\alpha_1 \cdots \alpha_c)} \prod_{i=1}^{c} \hat\theta_i^{\alpha_i - 1}$

$\exp(\text{Eq}(12)) = \left(\prod \hat\theta_i^{\alpha_i + n_i - \frac{1}{2}}\right)^{-1} \cdot B_c(\alpha_1 \cdots \alpha_c) \cdot (n)^{\frac{1}{2}(c-1)} \cdot e^{\frac{c}{2}} \left(\frac{1}{12}\right)^{\frac{c}{2}}$

A B

Factors A & B look like Stirling Approxn to Product of factorials divided by factorial of sum.

0 Set α_i = [illegible].

$\frac{(\sum n_i)^{\frac{1}{2}(c-1)}}{\prod_{i=1}^{c} \hat\theta_i^{n_i + \frac{1}{2}}}$ ⊗ $\frac{(n^{\frac{1}{2}})^{c-1}}{\prod\left(\frac{n_i}{n}\right)^{n_i} \cdot \prod\left(\frac{n_i}{n}\right)^{\frac{1}{2}}} = \frac{(n^{\frac{1}{2}})^{c-1}}{\left(\frac{\prod n_i^{n_i}\sqrt{n_i}}{n^n(\sqrt{n})^c}\right)} = \frac{\prod n_i!}{n!} \sqrt{\frac{1}{(\pi)^{c-1}}}$

$\hat\theta_i = \frac{n_i}{n}$ — no $\hat\theta_i = \frac{n_i + \alpha_i}{n + \sum\alpha_i}$

M = $\sum\alpha_i$

So should we put "π" instead of "n" here ?

So I get something they do except for constant factor of $(\sqrt{\pi})^{c-1}$

exp of MML is reciprocal of pc

~~So they have a factor~~ [illegible] $(n^{\frac{1}{2}})^{c-1}$ [illegible]

Work work this out !

for t. formula; if α_i all = 1 then formula is $\frac{k-1!\ \prod n_i!}{(k-1+\sum n_i)!}$ if all $\alpha_i = 1$

If some $\alpha_i \neq 1$ then ~ $\frac{\prod(\alpha_i + n_i - 1)!}{\prod(\alpha_i - 1)!}$ [illegible] $(\sum(n_i + \alpha_i) - 1)!$

2.2.05

STM BBN.03 SM.12 Z.141,20 SVM.30

30: 16.29 (Spec): One rough and dirty approach: pick k best codes thus far. Specifying the k dim vector of weights, will have a certain pc. We want to select k ⇒ the pc of the vector ("width" in space) times the multiplier of pc obtained, is max. So we try various k values & see how much the pc weight...

.03: 17.19 **BBN**: One way to do BBN's: A particular Net of this sort will have a certain set of adjustable params. Finding good values to code a particular corpus can be very difficult. A possible way: Say k params in net; pick random pts in k space: for each, compute pc of corpus (ΣP_i); try many random k vectors, get P_i: keep 100 best (of 1000) k vectors in memory. After many trials, make predictions by using simple mean of prods of 100 best k vectors.

0 Will this work? How many random trials is needed before it begins to get good?

Sounds like a "Wisdom of Crowds" problem!

12. **SM** 2.03 I might want a set of securities. Could be used to realize Cover's scheme: Or perhaps Cover's scheme w. finite (or exponential) many. We might find Tom's system by selecting the "Best k" (k ≠ 1000 k is just a name of a number — an integer) of 1000 different stocks on the style of (.00)!

0: Well, I wrote IAP55 (Bayes Learning / Power Bayes Library) which does the $\int\cdots\int \prod_{i=1}^{c} p(i)^{n_i}\, dp_1\cdots p_c$ ($p_i \geq 0$, $\Sigma p_i = 1$)

I got $\frac{(c-1)!\,\prod(n_i!)}{(c-1+\Sigma n_i)!}$ — Not far from what I expected.

It is similar to formula used in Feb 2000 TM in Letter to Wolff.

I also used an additional factor of $\frac{1}{c+1}$: But they had to do w. some extra coding costs... I don't remember what, hwr. ($n_z = 1 \mid \Sigma n_i = 1$)

I'm uneasy about this: Consider the "polynomial theorem": expand $(p_1+p_2\cdots p_c)^{\Sigma n_i}$: for terms involving $\frac{(\Sigma n_i)!}{\prod n_i!}$; if we integrate over the $\Sigma p_i = 1$ hyperplane, we get the area of the hyperplane, which is a function of c, but not of Σn_i: If we look at the integrals we divided terms by, hwr. $\prod n_i!$ cancels, $(c-1+\Sigma n_i)!$ only partly cancels: $(c-1)!$ doesn't cancel. — it may be area of hyperplane; but c factors remaining $n, n-1 \cdots n-c+1$ remain: they shouldn't!

30 **SVM** They first try classification separation using hyperplanes; If this doesn't work well, (Actually they use a particular type of N.L. xfm that gives "UNIVERSAL" results — unclear as to what kind of "Universality" it is.) They use a "Random-ish" non-linear xfm into a dimension > that of original hyperplane test. Then they try hyperplane separation in hyper dim space.

If the xform to hyper dim space is random, it costs nothing. If we do 10 random trials & select best one, it may be o.k., but we have SOY distortion. Taking average of several random n.l. xforms would give no SOY!

2.5.05
STM

Rissanen's: $\log_2^* n$ .00

0: SN on using $2^{-N(\log_2^* n)}$ as a prior for integers:
Actually using $\frac{1}{n}$ (which diverges) may be ok, and much simpler. We are only interested in ratios of pc's, and this is way be good enough!

.3 T. Main Problems T. 3rd IAP Lecture describes Alpha: Phases) & 2.
Also a TGP system that supposed to → grow → Strong A.I.
What are differences in these systems?

1) For Alpha: Preliminary exp on MTM type problems (Math). Use of Lsp-Fns lang. & Lsrch.
Using PPM or subst to context augmented PPM, or S-CFG discovery for search.
How good this could be, is unclear. We may be able to "soupup" the search algm so it would
0 learn some interesting Math.
11 → Perhaps teach it Logic & train it to use logic in problem solving. (.20)

2) Getting to S-induction in Phase 1 — is not so easy! I can do
QA induction from strings w/o nos. to nos. — a not bad, but not
universal way. The output is a no. & we try to get low m.s. error. Ideally, the output would
15 be an arby p.d., but I haven't figured out a practical way to do it yet. → (.33)
→ The 3 input UMC is universal, but I have no good way to fit it to a corpus. → (.33)

→ T. problems HOW TO Express/define/formalize/implement S-Functions

9
0: (.11) —— For D induction: Using Lsrch seems v.g., but perhaps use "partial fit" from t. No. of Problems
a Cand. solves. It is certainly wasteful not to use this info.) We have been considering
something like it for "problem Pool" training Corps, in which we got partial P.B.
(By Wt. in t. corpus) on t. basis of how many problems, & how hard, was obtained.
In both d-induction & S-induction, a Big Ques. how to allocate "Wts in corpus" on
t. basis of how many probs solved & which ones solved (& in t. case of S-induction,
how bad was failure: pc = 0 is a possy for certain of t. induction problems.
One way to deal w/ this is by finding ways of doing it that were (a) of hi pc (b) effective
2 in t. past. E.g. in S-induction, if a Cand assigns ϕ to a particular
correct A_i, we can give it a default pc, or a pc that depends somehow on
t. Q_i or some other "Correlate".

33: (.15) → It would probly not be diff to define & implement a P.D. of all possl. P.D.'s on any range
e.g. (0,1), $(-\infty, \phi)$, $(\phi, +\infty)$. Hvr, normally, if its a monomodal d.f., there are only a few possys
& they are good enuf (Gauss, & Gamma). Modns could be ① skinnier/fatter tails ② skewness
③ Multimodality. So we could get "all possl. functional forms" by allowing arby (normed)
functions — but actually, only a few forms will get much use & ∴ have very
hi aprobs. So don't worry about universality, for now — this looks good enuf for now → (Spec. 20.00)

NB: Implementing the distance could be done by the "R" input of the 3 IUM model of induction. This also applies to approach of

0: (Spec (19.40)): Can I genz. this idea to strings (discrete)? One way: we have several distance measures below. Strings that can measure the "error" of a particular An production.

12. (1) Information distance: Universal, Correct, very expensive (.025) (2) Same as (1) but using a non-universal (set of x mns (or a universal set w. heavy wt. on an important subset of mcts)) in the style of Newell-Simon GPS: TM learns how to do it & then having least bound in "available time" (3) Edit distance (I don't know how expensive this is: ext. average) (4) Hamming distance

06 (cheap but not r.g.).

---

(2) (.025) could include any "invented" distance measures appropriate to certain Question Classes.

3) Continued from 19.19. We have what seem to be 2 kinds of Gores for srch: a) Lsrch b) BU srch. In Lsrch, d-induction is simple (normally); we just make trials in PC order. The Guiding pd is modified by past success. Hwr, in (mem. 19.20, we may want to use a amt of wt. given to previous "successful" conds — depending on how many probs they successfully solved. The most serious problem is updating of the Guiding PD.

I expect to use either S.CFG's a/o PPM. Modified to detect subtrees as context. Since I'm not really using Lsrch (I'm not doing inv. probs & I'm not doing the Lsrch version for OZ probs), I don't expect a big good induction (until I get to phase 2).

A possi. way to use BU srch for d-induction: Use "problem pool". BU srch tries to find

.19 a fun that will solve as many probs in pool as possi., using 19.20 as "feedback" info. → 30

0 → SN Mite it not be possi. (practical) to use Lsrch for OZ probs (in particular for searching for good induction Model(s))? The search is over optzn techniques, & TM starts out w. a grammar that can readily construct good optzn. techqs. (Productand set of optzn tqs.) This would (perhaps) be very Similar to Starting TM work on "Phase 2 first"

OZT's = set of optzn techniques

SN I usually think of OZT's as being independently tested — but if OZT's work by making trials, Then they are not at all independent, since they can use each other's trials info. Hwr, a v.g. induction System could look at a bunch of trials & suggest a trial w. very hy expected G. & So in this way, a v.g. induction system could do much v.g. "Self. Improvement"

0: .19 On Phase 1 S-induction: We can use 2 part codes, both for numeric outputs & for string output (.00–.06). In "Lsrch" we do trials in order of pc of the discrete "first part" of the code. For numeric outputs, for each "first part" trial, we have to optimize parameters & get width of optimum. If there are several good optima, we have to get value & width of each & "sum" them. It mite be worth while to do random params ... (but I vaguely remember this not working!) under certain conds... it mite work fine using steps. Another trick is to use a small SSZ & find broad peak, then ↑ SSZ & narrow down peak. There is prob. an optimum rate to ↑ SSZ.

P(A|Q) v.s. P(A,Q).

For S-induction w. string output, → See 21.00 → 21.00

20:20-40: S-induction w. string output: For "Lsrch": 2 part prog: do first part in pc order, which will have a string output: pc if Pcs is not the desired fn, use 20_00 .06 to get to correct fn. This can be time consuming & we might try using analogs of: "nonsense" output at 20.30-40.

T. forg. were born for "Lsrch": For BU srch, we start w 1 or more cands that give pc>ϕ. to corpus, but are usually rather poor. We then use a kind of "hill climbing" by induction on corpus of cands & their Gave values (= pc's). There are many many other techniques of this sort in the Math/Engrg community, & a sizable subset in the Mach. Lrng. community. I am most familiar w. Feed-forward ANN & GP/GA. — But there are other impl. approaches: SVM, BBN, Decision trees, ... (See Duda, Hart, Stork for more ways). PCA(ϕ) v.s. PCA,Q).

While recurrent ANN's are universal in a Turing sense, feed forward (& type I am familiar w.) are not — So I'm doing GP: I think BBN is also not "Turing Universal". Decision trees prob. isn't. SVM's I don't know, but they would seem to be universal only in the sense that feed forward ANN's are "universal".

"Inductive Logic Prog" may be Turing Universal. } Are there any other T. universal systems?
GP/GA
Recurrent ANN

SN: 19.03ff Here I have a Bunch of Parts & I'm trying to put them together to make a TM, & I'm seeing if the parts don't work write, or if they don't fit together, or if I can fix them or get different parts. So far, seems to be going along OK:

No Really Bad (problems troubles) found!

19.03 ff is the beginning of a parts list! Perhaps a quick run over what I think are important components & "parts" that may need to be.

1) Problems for QA T≤Q: Language(s) for expression of QA probs & solns.
Use of "Loop": forms P(Q,A) or P(A|Q). both possl.
for MTM A_i = F_j(Q_k) is available.

2) For d-Induction on QA probs (MTMs): Use Lsrch: Guiding PD: PPM, PPM + substring discovery, discovery, Operations CSG discovery.

2-7-05
STM

3.2k uni ~ 5pp

A possl. Intro to talks! We have this program that can learn all of its past experience & make predictions/judgements based on this experience in the best possl. way. It could bring all of this experience to bear on any problem given to it.

We have what appears to be a way to actually write such a program. We have a program that could, indeed, learn optimally from its past in the best possible way — but there is a catch — the program takes too much time and memory.

We will describe a sequence of demo programs that enable us to overcome this difficulty.

~~We start out with a program designed to be practical only.~~

First a description of the prediction system that is very accurate, very economical of data — but is much too expensive to run.

Next explain how ALP works & what its problems are.

At a certain point in the above talk, discuss universality: Feed forward ANNs (and R-B's + BBN's) are "universal" in a certain sense: Describe just how they are universal.

Then explain how they are not "fully universal" that there is a very narrow category of things they can learn & a very broad category of things they cannot learn.

Introduce to had examples of things that a Universal feed forward ANN cannot learn.

Well, we can regard it as a comb. search problem. To fix an ANN to do a certain need a certain amt. of info to describe a functional form. Using a Universal machine, we need much fewer bits to describe functional forms that have occurred often or have "recently" occurred before.

A sing wave can be described by 2 #'s by a UTM — no matter how many cycles are given. To fix ANN needs a no. of bits or # no. of cycles.

2.8.05
5 TM

David Lindsay [Room 131] (near 23)
david@cs.rhul.ac.uk
or D.G.Lindsay@ " " " "

: CFG dizzy! In analysing Trav Grammars, I got in to a serious problem when I considered that there were very many diffrn ways to parse a string a set of strings in the simple grammar:

$\Sigma \to$ .1 a
.1 b
.15 c
.2 d
.3 e
.05 → F
.1 → N
N → abc

i.e. If N occured n times in the corpus, there were 2^N diffrnt ways one could parse that corpus! — assigning either N or directly abc — Thus the particular string had a much higher pc than any one parsing would give to it.

11:52 P Δ3
2:56 R
6:31 a Δ3½
12:18 A
4:12 4-to
6:14 2

2.10.05 — 22.00 ff is infinite directory for Intro to first talk: But I may want to First Define "Strong A.I.", then How could we achieve this? (Desirability of ALI) What are impediments to such an approach?

1) Defn. of S.AI: Ability to synth. S.I. on hardware. This plus "Moore's Law" gives extremely capable A.I. in finite ("short") time. In

R.K. on work: Toward History-based Grammars: = HBG
E. Black
F. Jelinek
J. Lafferty
et al.
= "HGB"

One test of Turing Universality is Existance of an "Undecidability" ≃ "Halting Problem".
A better positive Criterion: That any Universal Modeling System can Simulate any other in a finite no of insts — or only a finite amount "x to" longer.

On convergence of $\Sigma p(i)$!
Say Normalization constant = $A > 1$. $A = \frac{1}{\prod(1-u_i)}$

$\ln A = -\Sigma \ln(1-u_i)$

$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 \dots$

$\ln' = \frac{1}{(1+x)}$ $\quad \ln'' = -\frac{1}{(1+x)^2}$ $\quad \frac{2}{(1+x)^3}$

$\ln(1-x) = -x + x^2 - 2x^3 + 6x^4 \dots$

$\Sigma \ln(1+x_i) = -\Sigma x_i + \Sigma x^2 \dots$

$\Sigma \ln(1-x_i) = -\ln A = -\Sigma x_i + \Sigma x^2 - \Sigma x^3$

$\Sigma x_i = \ln A + \Sigma x^2$ $\quad \Sigma x^2 < \text{say } \ln A$

So $\Sigma x_i < \sim 2 \ln A$

((: I need to see just what I want in a scheme to detect proper regys in functions (Lisp a/c Forth a/o machine lang). CFG's augmented PPMs are 2 systems I need to see how Good they are. This seems like a light problem needing to be understood! → 24.00 Spec.

2.9.05
5TM

2.9.05 — ①

11:20
1:38 $2\frac{1}{3}$
4:59 $3\frac{1}{3}$

10 : On comparing predn models that have different nos. of continuous params: n_0, n_1 ; $n_0 < n_1$

One way: Increase the no. of params in the smaller param. model ~~~~ so it has n_1 params

Then, to compare 2 models integrate each over the n_1 dim space.

This usually means that $\int 1\, dim = 1$ for each dimension. We end up w. a coding cost that is the integral over all of the n_i dim space — for $i = 0$ or 1.

So this is nothing new! We have to get an apriori density function on param. space.

~~We can't assume that~~ How to do this is still unclear: the different params could be

0 → of diffrnt "dimension" & ∴ not comparable. In ordinary linear regression they are sort of comparable, and "~±1.0". When this is not true, we could do it!

Unclear as to what to do.

IMPT! ((If the params are really different, the only way to get a reasonable apriori for them is to look at "similar" predn probs "in the past"

"~ to situations in past" may be via complicated functions, etc — or recursively. !

This has to be analysed in greater detail: It's very impt!

20 (IN) · It may be easy to store the equivt. of several (|| CFG's in a W.B. xfmr (PPM).

very compt dgtr for storing || Models of corpus. { A given context could occur in a low pc grammar, but have very hy pc within that grammar. The product of these 2 pc's could be > than the corresponding pc's in a much more likely grammar!

---

0 . On Normzn of Stoc. Comp — like ALP model. We could use unnormzd form. ~~~~ Pc ratios would all approach correct values, but I do no know how much slower, hvr.

MAJOR HEURISTIC .06 – .09

Quick Abort : New View? : .11 – .17

α1 – α14 follows 5TM25

Then 5TM26 follows.

So α1 → (5TM) 25α1
→ 2 → 25α2 ..

0: 23.90 : Out of 2332 : Ideally, we want TM to be able to look at a bunch of function trees, & find a set of common sub-trees and then decide which definitions would be best : (considering overlaps of the trees). Criteria (actual code sizes) will have to be defined/calculated/approx'd.

16 2.17.05 Hmm, A MAJOR (DIFFCLT) HEURISTIC! After solving a problem (w. Diffy):
To try to find Heuristic Rule(s) ⇒ that Soln. would be faster. Occasionally,
>1 heur can be gleaned from a Single Soln. How to put Rel Heur in More Usable
.09 form --- UNCLEAR. 2 KINDS of Heur: (a) ↑ of PC of Soln. (use a "standard"
{ ma Prods for finding Reps's in corpus) (b) "Quick Abort" techniques. — I Recall I
" had parsed this to cover a (large range of srch. heurs. One (not so General, but perhaps
Very Useful) — as a Cand. is being Generated, develop "obs" to predict
.15 ↑ pc of failure (Unclear that this is not covered by normal assignment
.16 of (or app to ends to be "Aborted"). — Seem 26.06 for one kind of "Quick abort" Rest doesn't
Seem to be covered.
.17 → ?? Could it be that the pc method of .15 – .16 covers most (if not all) QuickAbort methods? >
— in which case, finding good predictive assigning good pc's to conds (& parts of conds)
is all that one has to do in Phase 1.

SN Trouble w. Bernoulli Seq. formula! Say 2 symbols only: If n_0 φ's & n_1 1's have occurred.
Prem (pc of 1)/(pc of 0) (for next Symbol) is $\frac{n_1+1+1}{n_0+1+0}$ ← This is ratio of approps for n_1+1, n_0+0.
Thus $n_1 \to n_1+1$; $n_0 \to n_0+0$. These approps were obtained by direct Integration!

$$\int_0^1 p^{n_0}(1-p)^{n_1}dp = \frac{n_0!\,n_1!}{(n_0+n_1+1)!}$$ Ohh! it's Ok.! ☺

[redacted] [redacted]

2.11.05 STM 25 α 1

(5 IAP STM α) This Page follows STM 25

Re: Paul Vitanyi: I've known him for many years: Since he and Ming Li ~~began to think about writing "Kolmogorov Complexity and its applications"~~ [An Introduction to]. first came to talk to me, ~~[illegible]~~ as ~~[illegible]~~ a founder of the field of ~~[illegible]~~ program-length complexity — preparatory to their writing "An Introduction to Kolmogorov Complexity and its Applications".

~~[illegible]~~

His main influence on me, has been to force me to express in more exact form, ideas that I had been a bit vague about.

His influence on this field has ~~[illegible]~~ been enormous. At the beginning it was thru his application of ~~[illegible]~~ ideas about ~~[illegible]~~ randomness to a variety of proofs in Mathematics. Later, his influence was largely thru the effect of his book on Kolmogorov complexity — it reviewed much of the work in the field and ~~[illegible]~~ continues ~~the only ever~~ to be the only available text book in this area. In addition the book has been an important ~~[illegible]~~ stimulus for continued research by others.

As a person, I have been much impressed by his ~~[illegible]~~ curiosity and erudition in a great variety of fields — in ~~[illegible]~~ Mathematics, Science and other Arts. He is a prodigious world traveler. Likes to meet new and interesting people ... taste new and interesting food.

[SN] Very Impt idea! Try to see what kinds of functional Regys — see those cheap very fast SONY graphic CPU's — used in PS2 or PS3 — occur in Machine Code e/o for it or perhaps a specially constructed super-fast Machine. I did have this idea (long ago) of a Machine w. many addresses being sent around. A few (often duplicated) function parameters. Addresses would stand in line to get processed. Perhaps work out General problem of Regys in Languages / Machines arbitrary instructs. — See (2.12.05 5 IAP 5 TM) — Ea 2.00 .00 for descr. of T. Address Machine

(SN) while Sci. patents would show innovation, it would prob. not show approach to singularity (much), — We would have Lisp, say, plus a complete set of Risc insts. [insts] So Gradually TM would learn to use the Risc insts!

One trick: when one wants to do such a recursive fn, a special symbol follows the "data" symbol, that enables special insts that make recursive ideas possl.

— α 1 to .00

2·12·05 STM
SAIL
STM } mix

2542

T. ADDRESS Machine (TAM)

: T. address machine (TAM): Each address holds name of the function sub-funct to be evaluated. — It keeps track of where we are in the [illegible] and knows when we are done. The addresses change true for each sub-funct evaln. When (too many) addresses need a certain funct evaln., duplicate evaluators are constructed (deconstructed) by a higher order system.

11 -2-1/4
2446
5555 3/6

In learning ANL: The notation: (If-Then-Push) +, Add. would be used.
If means that "+, "ob" is juxtaposed w. Add" + "Op" : In general, we may want to modify the language syntax so say "If Then" is one word, so Push [illegible] (If Then) Push is a prefix of +, Add. ~~Or maybe If +~~ If Then, automatically pops next symbol. It maybe that (If Then Push) should be a primitive. So If Then push +, Add means If + then pop & push Add.
So we may be able to modify syntax so as to make ~~PPM a better~~ rules in PPM more suitable to PPM. → a 4.00 spec.

m.glickstein@ucl.ac.uk

.17R

Actually, there are at least 2 ways to do the Grammar: Corpus interacting w. Grammar. ⇒ α3.00

SN Discuss "Grammar Grammar" in Deriving how to assign aprip. to the Grammar: The discussn. I have may be O.G. : But look at it! — The Grammar Grammar can end up w. better (layers) for the entire corpus.
— better for starting out — for small "modifns of corp. corpus.

(Inadequate Apriori 1980)

SN Intro Lecture: What is my Goal in S. AI? (Terry Jeffers Psych. info about him)

Requisitionary — (Must be Plans in Barrel)

Too little work [illegible].

Why are impediments in most A.I. work that prevent S. AI?
1) Early stuff: Poor understanding of Learning probability, more than [illegible]
2) 3) Use of incomplete subjective induction. Not a full stochastic model. — Lack of ~~No [illegible]~~ Only in the prob lang. Little Inter probability.
3) Poor TSQs
4)

1985 role of [illegible] done.
② NIPS comparison
③ AI Subgroup on Uncertainty in A.I.

Expected Value.

Most of pc $\frac{1}{n}$ of $P = A\alpha^{-n}$ then pc of extra NT is always α & usually ≈1.

10:α2.17 : In defining the set of NT's, we need to give prob of an integer; n; $\frac{1}{n}$ is actually not bad, because we (probably) don't have to normalize. It does, hvr, give lots of wt. to large n values. $E\,n = \infty$: very fast divergence. $2^{-\log^* n}$ also has $E\,p(n)$ diverge rapidly. I impt thing is $\frac{P(n+1)}{P(n)} = \frac{n}{n+1}$ close to 1. In the case of $P(n) = A\alpha^{n}$ (A = normaliz const) $\frac{P(n+1)}{P(n)} = \alpha^{-1}$ which is usually close to 1
(I have computation for E(n) in terms of α within last wk. or so of STM. notes)
So usually (particularly if n > 4, say, $\frac{P(n+1)}{P(n)}$ being close to 1, will give essentially no "actual" boost for "another NT"! Hvr an additional boost is that of defining the NT →13

10 [SN] Could the large discussion be relevant to linear regression (HMC) problem? Perhaps not. Just saying we need another coef, isn't where the pc loss — its in the narrowness of the coefficient DF — which .00 →.08 doesn't help with.

3:.08 → Another novel series pc is in the # of the no. of Terminals & non terminals (= r) so referring to say one pc of it more. — for k references to symbols the r is amount to [redacted] (#28) Cite goto
$\left(\frac{r}{r+1}\right)^k \approx e^{-\frac{k}{r}}$.

This "small ↓ in pc" again occurs when we add a "production", or when we ↓ no. of productions (Merge). This is because telling the no. of productions for each NT requires statement of an integer., So in both cases ↑ or ↓ by 1 of no. of NT's or ↑↓ by 1 of no. of productions for a pc, will be multi of p by a no. "close to 1".

The third kind of "punctuation symbol" is the one that marks the end of a production string". I think that again "Δpc ≈ 1.
The fourth kind is "end of list of production strings", & this also has a "Δpc" ratio of close to 1.

I think the moral is, we can use $\frac{1}{n}$ or 1 or α (a constant close to 1) for any of these quantity. Saves a lot of bookkeeping & calcn.!

It looks like the most complicated part (by far) of calculating pc of a grammar model is involved w. ambiguity ... multiple parses.

30 Probably it would be best to (code) treat the sequence of production strings as a single string w. punct. symbols betw. production strings. — It would seem that if a symbol (T. or N.T.) appeared w. hy freq. in one production, it would tend to have hy freqs in all productions. I could look at this Q empirically — I really haven't ever tried any of this on real data

2·14·05

25 ~~24~~

~~α 23~~
STM

SAARBrocken
SARB

20 : α2.16 — In a related context: At SAARB in ANL using stack: The trees were generated by a cFL! So if "If . . . ." was chosen, we got
If α Then β else γ : (β) was a certain Type, which would restrict them considerably. Also, if I had a pd of α, β γ based on past successful functions, this would be very useful. So ⟨If, Then, Else⟩ would have its own "ppm" BW xfm. We'd like to context to go further, if possl. — e.g. The context of "If Then else" could be considered.

0 To what extent can I represent all that I want by a (½) CFG?
I.e. If a certain Config occurs w. enuf freq., then I define a ~~[illegible]~~ then its frequency.

3. One trouble w. "defining" is that one tends to parse only 1 way, so one loses other (possibly better) defns. By returning to "Ambiguity" of a CFG, one could get around that.

15 [.13 - .15 gets around an objection that I had to "defns" i.e. they "froze" the parsing. In fact, whenever a bunch of new defns have been obtained, one should periodically "reparse the past" (like Wolff)]

20 Actually, humans don't reparse (in view of new concs) very often: — Mainly because it's time consuming, difft, & often unpleasant. Tho it can be very pleasant, indeed, when a much better parse (≡ formulation ≡ code) for the past is found.

Some Effects of reparsing: Reassignment of (perhaps very) different pc's to various concs, & the total IR + pc of the entire corpus is much ↑,

30 — We can expect much better predn in future!

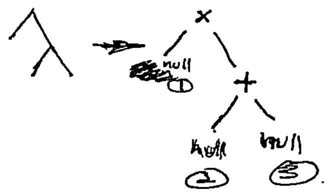
-31 ⟶ On devising simple (non-looping) Grammar, using sub-trees only: | search by frequency. It is easy to define a sub-tree! We just give its top node, then for each branch, we have usually a N.T. or T. Hvr, we add as a terminal, + null — which makes that point an argument of the resultant subtree: (See α5.01) ⟶

SN1 on use of unnamed pc for QA lang: This will bias towards functs that are Total recursive.

2.14.05

∝. STM

>0 : Size A.



This gives the subtree $\times$ +, w. 3 arguments:

$\times(\text{argt}_1, +(\text{argt}_2, \text{argt}_3))$

Once we define a subtree, it becomes a N.T. & can occur as an argt to any function. Hvr, we may want to keep track of freqs of various N.T.'s & T's at each "context" of 1 or more nodes.

This "context" idea is certainly impt. — but unclear as to how to implement it!

Anyway, This "finding of common sub-strings" is just "chunking" — no magic. The grammar itself is loop free. Because of its non-recursivity (= no loops) it gets very large. Mindful of PPM in that respect.

I did write about how (in theory, but not really practically) how to use sub-trees as contexts. If I could, indeed, implement that w. the foregoing sub-trees as defined objects, I mite get a fairly powerful function predictor.

Perhaps if I did define sub-trees, using defined sub-trees as contexts in ordinary PPM, mite work O.K.!

$$\boxed{5N} \quad \int_t^\infty z^n e^{-\alpha z} dz = \frac{n!}{\alpha^{n+1}} e^{-\alpha t} \cdot \sum_{j=0}^{n} \frac{(\alpha t)^j}{j!}$$

$= \frac{(\alpha t)^j}{j!}$

I thot this finite sum mite be useful for summing effects of ll parses for data of α in defining α as a seq of primitive symbols, but the formula isn't quite the same. I think, hvr, that there is a corresponding approxn. that does give a well-known function (or maybe close to it).

$$\approx \sum_{r=1}^{r} \frac{\prod (n_i - r)!}{\left(\left(\sum (1 + n_i - r)\right) - 1\right)!} \qquad \frac{\left(\left(\sum (1 + n_i - r)\right) - 1\right)!}{\prod (n_i - r)!}$$

: But this is unlikely! — the 2 factors probably don't exactly cancel.

Also this "r" assumes that all of the primitive symbols are used in α (the newly defined N.T.) once. ←

The exact no. of symbols of each type & the no. of each symbol in the original corpus determine the range of summation of "r".

Perhaps it's easiest to do this as an integral ..... Some kind of modification of Gamma Beta function

Perhaps the simplest form of the summation!

[SN] Re: Discussion of Apri's "Subjectivity" in Lecture 1. I may want to expand this: I did write how the apri of a person changes during his life, but I didn't make it clear how this was relevant to anything!

Say this reverse: info at birth plus ...

I may express more exactly: The Apriori at any particular point in a person's life, is based on the sum of all of his experiences plus the info built into him at birth. The Apri itself, is a kind of statistical summary of all of that info. Needless to say, these apri a fortiori, the data on which it is based, will differ much from person to person.

For those that feel science must, of necessity, be purely objective — this would seem to be bad news — The there is, indeed, a lot of this apriori that just about all humans have in common, there is much that is highly individualized.

How can we deal with this difficulty? — I will divert to a closely associated problem!

How statisticians deal with apriori information?

A common way is to select a set of hypotheses to consider, then find which one best fits the new (non-apriori) data. For example, in linear regression, we decide, apriori, to use 6 linear coefficients to predict the next element of a time series. The coefficients can be assumed to have a uniform a priori distribution. (1) (Fn. we will gloss over certain non-catastrophic difficulties with this approach).

The statistician says "If we make these assumptions about the models for the data, then we will obtain such and such predictions with such and such variance. These are rather objective statements.

i.e. apris do not converge $\sum_{-\infty}^{+\infty} f(x)\,dx = \infty$. — hvr. They may not have to converge: we may be interested in relative sizes of params so we can get rel. wts of various hypoths.

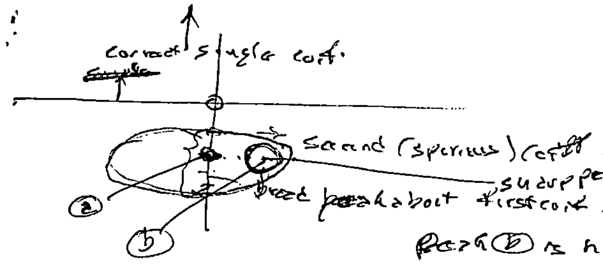
: [SN] Ah! Consider 6 coefs for linear Regression! We get a sharp distribution of 6 coefs. then in the subspace of 5 coefs (w. 1 coef = 0), we get a broader fit, w. much more total wt. but its pred'ns aren't as good.

It may be that if we averaged over the entire 6 dim space, & the true generator of the seq. was only 5 coefs, then we would still get terrible wts for the sets of coefs. It may be that the peak for 6 coefs will be narrower but if the 6th coef is zero, the peak is very broad. Easier to visualize if there were actually 1 coef but we consider a space of 2 coefs.

2.15.65

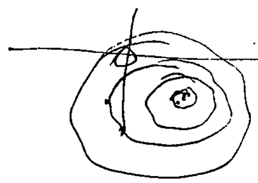
∝ STM

00



Peak (b) is higher than peak (a), but much narrower

What would the ms error contours look like?



It would be worth while to do an empirical study of (~~[illegible]~~ coef and generator) and consideration of the 2 dim space of 2 coefs. What do the contours look like?

0

Actually, the diff't coefs could be = "equivalent" in pract in each direction, if we are doing linear regression on a time series. But if each ~~[illegible]~~ linear coef is entirely diff't (say of diff't dimension), this is not true, and it's unclear as to how to compare wts. ~~[illegible]~~ for the diff't coefs — in terms of info content. One way would be to look at the past and see what relative wts were of what relative importance.

20 In the case of two series (linear or non-linear) we can look at the past of the same time series to see what relative wts should be used.

In specifying coefs of the most general kind: One kind of info to specify the "center of the distribution" — another kind to specify distance from the center. The first kind is needed for naïve prediction. The second kind holds most of the info for large size, hy precision predns. It's usually this second kind (w. large size) that statisticians are most concerned w.

A unifying ~~method~~ metric for coefs of diff't dimension is ~~[illegible]~~ the partial derivative of the ~~[illegible]~~ (ms error) wrt each coef. In general we have a Hessian Matrix. If off diag terms ~~=~~ ≠ 0, then the coefs are not indep and ~~[illegible]~~, to ask for the "amt. of info" in a particular coef, is meaningless!

30

31 So how to compare Hessians of diff't nos. of dimensions? Since Hessian is symmetric, it is diagonalizable by rotation: so ∃ linear combs of coefs that lead to indep "coefs" w. diff't $\frac{\partial^2 \text{ms}}{\partial x^2}$ ~~[illegible]~~ for each "coef". So one inserts info in each coeff by amount necy to ~~[illegible]~~ ↓ ms error by same amount. After the coefs have each gotten to pt. of = contribn to error, then each coef ~~[illegible]~~ will henceforth have same no. of bits of info. So, initially (as perhaps part of the "discrete" form of the model) one gets the coefs down to precisions, so that each error in coeff contributes equally to error. Then henceforth — same precision in each direction (in terms of bits of precision)

2.15.05
5TM

25 α8

SPEC
α7.90 : but not in terms of simple error in each "coiff" : This gets us t. coif of ln n in t. "Bcost" form of info — I guess Rissanen Got "½", which seems right. T. Big problem is t. main Bcost not ∝ ln n : Th. discrete part.

01 **So α 7.31–.40 may be great break thru !** → 5TM36.01

Hvr, to start off, t. determinant of t. Hessian matrix gives t. initial (pc)$^{-1}$ of t. "peak". For Rio info is ∝ t. no. of dimensions in t. space.

In general, t. amt. of precision needed to specify t. peak will depend on ssz.

For large ssz, t. info needed is ∝ ln ssz : — in cost ∝ t. no of dims. Hvr, for small ssz info needed is more complex, & depends much on apri info.

To compare k coifs w. k+1 coifs: Compare determinants of Hessians of t. 2 situations. This will give relative pc's & to start off. As more precision, equal amts of info will be invested in each dimension. So it may well be that pc at peak divided by Determinant of Hessian (or maybe $\sqrt{|Hess|^{-1}}$?) is measure of pc at that peak.

(CAM ~3½ 4:30AM 7:47 3:17)

N.B. In adding up pc of various pts in coif space, I was using hypercubes w. same size in each direction. This is wrong. Use as orthobasis, the Eigenvectors of t. Hessian & make t. small hyper parallelepipeds w. edges ∝ (the basis vectors eigenvectors) (or perhaps $\frac{1}{\text{eigenvectors}}$). This automatically makes all directions in space equivalent.

I still don't have a good intuitive grasp of what's going on, hvr. !
I'll have to keep going over this in various ways until it becomes intuitively clear. (or false ☺)

Still, t. specification of t. Hessian as part of t. "discrete term of t. model" ... ? ☺?

Say one changes dimensions (inches to ft in one of t. coifs) — Hessian is mult by 12. (or most likely 12^2?)

Maybe we look at each dim as being derived (operationally, t. dimension is just a "small", additive (discrete part of derm) part. — yet this discrete additive part is very impt for small ssz. Hessian for linear regression is easy to compute!

$S = \Sigma$ sq. err $= \sum_{j=1}^{n}\left(x_{j+1} - \sum_{i=0}^{k-1} x_{j-i}\, z_i\right)^2 : \frac{\partial S}{\partial z_r}$ ~~$\sum_{j=1}^{n} x_{j+1} \sum_{i=0}^{k-1} \ldots x_{j-i} z_i \; x_{j-r}$~~ ok.

Apriori

0 (Sols) ① use of ½ APL for fixing. (Good Apriori functions; but be sure recursion is possl.
② for $\frac{3}{\phi}$ use ± very large no. (± is random)
③ → $\frac{\partial S}{\partial z_r} = 2\sum_{j=1}^{n} x_{j-r}\left(x_{j+1} - \sum_{i=0}^{k-1} x_{j-i} z_i\right)$ (+ A mentha 2) : Surprisingly: Hessian is already diag (no cross products)

Diag elements "not =", but "≈". I got this result some time ago when working on "$\sigma^2 \frac{n+m}{n-m}$". They differ much when k was close to n or $k \gtrsim \frac{1}{2} n$.

HA! It maybe that "Fisher Info" is relevant: It has to do w. derivative of ln of something ... which seems in direction of what I want.

2.16.05 25 α9

α STM

4:32 run:
back 5:00p

$\sqrt{x^2 \cdot y^2 \cdot z^2} = xyz$

0 Perhaps a Good Way to think about α 7,31 → 40ff: Think of SSZ=1; N=1: This

31 Gives the Hessian & t. Discrete part of t. dem. T. rest of t. dem is just "narrowing down & distribn." (.20)

SN As an example to test various/linear predn methods: Consider t. fune series w. an

∞ of coefs: T. char. eq. is $\prod_{i=0}^{\infty} (x - a_i)$ maybe $i = 1$ to ∞, $\sum_{i=0} |a_i| < 1$ → So t. a_i's are random; once chosen, they are "fixed".

4 $a_i = e^{-\frac{i}{\lambda}} R$ — R is a random no. chosen uniformly betw. −1 & +1. λ is like to "expected no. of coefs".

T. roots of char eq. are all in "unit ○".

Say λ = 3: for small n (≡ SSZ) we only are able to "resolve" a few coefs., because t. coefs for i ≫ 3 coef will be very small if i ≫ 3. As n ↑ we are able to resolve

0 smaller & smaller coefs.

→ Another possy is $a_i = \frac{1}{2} e^{-\frac{i^2}{\lambda}} \cdot (-1)^i$ ... $a_i = \frac{1}{2} e^{-\frac{i^2}{\lambda}}$

Hvr, if i = 0, $a_i = 1$ which is not so good — So .04 L will be o.k. for i=0 (usually).

So I just put $(\frac{1}{2})$ in as a factor.

Anyway! Re: t. "Hessian" form: for linear Regressn., it looks like I may end up w. something close to $\frac{n+k}{n-k} \sigma^2_{obs}$. Just use anyth for k=1 (finding mean & predns using mean) ———

Then do k dimensional analog, via t. diagonalized Hessian.

.01 So, to compare amt. of info in 6 v.s. 5 coef. models: for 5 coefs: we have constant: containing Dem of model + shape of Hessian (SSZ) than $\frac{5}{2} \log_2 n$ bits to deceb pt. of interest + $n \log \sigma$ (.25)

α β perhaps

for 6 coef Dem of model — shape of Hessian + $\frac{6}{2} \log n$ . . . . . . + (.25) $n \log \sigma'$

α' β'

α & α' maybe ≈ | I don't know about β v.s β'! They may differ by < 1 bit(?)

25 Also, both models have to be mult by pc. of corpus wrt that model. Since there are just n predns,

It will be ∝ σ^{-n} v.s. $(\sigma')^{-n}$ in t. 2 cases.

For large n t. 2 impt terms are $\frac{5}{2} \log_2 n + n \log \sigma$ v.s. $\frac{6}{2} \log_2 n + n \log \sigma'$.

for k coefs ~~$\frac{k}{2} \log n + n \log \sigma$~~ $\frac{k}{2} \ln n + n \ln \sigma_k$ v.s. $\frac{k'}{2} \ln n + n \ln \sigma_{k'}$

or $k \ln n + n \ln \sigma^2_k$ v.s. $k' \ln n + \ln \sigma^2_{k'}$. T. comparison of t. 2 Hessians should also be made: Unclear how to do this. or $n^k \cdot (\sigma^2)^n$ or $n^{\frac{k}{2}} \cdot \sigma^n$

. Look at $\left(\frac{p(\vec{w}_0)}{|H(\vec{w}_0)|}\right)^{\frac{1}{2}}$ as width of distribn; $p(\vec{w}_0)$ is pc of corpus wrt. that point in $\vec{w}_0$ space.

← See last p. of Lecture 2 for this eq.

|Hessian| = 2K · σ^2 $p(\vec{w}_0) \propto \sigma^n$ | $p^{\frac{1}{2}} \cdot H^{-\frac{1}{2}}$ | = $\frac{\sigma^{\frac{1}{2}n}}{2\sqrt{K} \cdot \sigma}$ = $\frac{\sigma^{\frac{1}{2}n - 1}}{2\sqrt{K}}$

$|H| \approx (\sigma^2)^k$ $p \propto \sigma^{-n}$ → $\frac{\sigma^{3n-2}}{4K}$?

maybe not. $\sqrt{\frac{p^k}{|H|}} = \sqrt{\frac{\sigma^{-kn}}{\sigma^{2k}}} = \sqrt{\sigma^{-k(n+2)}}$

$= \sigma^{-\frac{kn}{2} - 1}$!?

T. stuff about t. Hessian in last p of Lecture 5 is wrong.

2.1905

~~9~~ 5TM

25 α10

(in case of linear regression.)

oo : $|H| \neq \sigma^{2k}$. ← This would be Hessian of the total sq. error (perhaps)

— but what I'm interested in is the Hessian of $\propto \sigma^{-n}$ — the p.c. of the corpus wrt the model (?)

I may know $\frac{\partial^2 n\sigma^2}{\partial(\text{any coeff})^2} \approx n\sigma^2$ so $\frac{\partial^2(\sigma^2)}{\partial(\text{coeff})^2} \approx \sigma^2$.

If $\frac{\partial(\sigma)^2}{\partial x^2} = \alpha$ then what's $\frac{\partial^2 \sigma^n}{\partial x^2}$? $\frac{d\sigma^2}{dx} = 2\sigma\frac{d\sigma}{dx}$

$\frac{d^2\sigma^2}{dx^2} = \frac{d}{dx}\left(\frac{d\sigma^2}{dx}\right) = \frac{d}{dx}\left(2\sigma\frac{d\sigma}{dx}\right) = 2\left(\frac{d\sigma}{dx}\right)^2 + 2\sigma\frac{d^2\sigma}{dx^2}$.

$\frac{d\sigma^n}{dx} = n\sigma^{n-1}\frac{d\sigma}{dx}$

$\frac{d^2\sigma^2}{dx} = n(n-1)\sigma^{n-2}\left(\frac{d\sigma}{dx}\right)^2 + n\sigma^{n-1}\frac{d^2\sigma}{dx^2}$

This may work out o.k. — but drop it for the while: I have to work on the Lectures (i.e. 3).

---

o I think the correct formula for "width" times P is

$\left(\frac{P^k}{|H|}\right)^{\frac{1}{2}} \cdot P = |H|^{-\frac{1}{2}} \cdot P^{\frac{k}{2}+1}$: The value of $|H|$, hvr, must be computed

viz. coff $\frac{\partial \sigma^{-n}}{\partial x} = -n\sigma^{-n-1}\frac{\partial\sigma}{\partial x}$

$\frac{\partial^2\sigma^{-n}}{\partial x^2} = -n\sigma^{-n-1}\frac{\partial^2\sigma}{\partial x^2} + n(n+1)\left(\frac{d\sigma}{dx}\right)^2$ (← = 0)

$= \frac{-n}{\sigma^{n+1}} \cdot \frac{\partial^2\sigma}{\partial x^2}$. say $\frac{\partial^2\sigma}{\partial x^2} \approx 2\sigma^2_{\text{observed}} \approx 2\sigma_0^2$.

so $|H| = $ ~~...~~ $P = \sigma_0^{-n}$

o $P = \sigma_0^{-n}$

$\left(\frac{-n}{\sigma_0^{n+1}} \cdot 2\sigma_0^2\right)^k = \left(\frac{-2n}{\sigma_0^{n-1}}\right)^k$

$\frac{P^k}{|H|} = \frac{\sigma_0^{-nk} \cdot \sigma_0^{(n-1)k}}{(-2n)^k} = \frac{\sigma_0^{-nk} \cdot \sigma_0^{nk-k}}{(-2)^k \cdot n^k} = \frac{\sigma_0^{-k}}{(-2)^k \cdot n^k}$

$\sqrt{\frac{P^k}{|H|}} \cdot P = \frac{\sigma_0^{-k}}{(-2)^k \cdot n^k} \cdot \frac{1}{\sigma_0^n} = \frac{\sigma_0^{-(n+k)}}{(-1)^k\, 2^k\, n^k}$.

± from $(-)^k$ irrelevant. Also mult by 2^k for actual full width.

so $\frac{\sigma_0^{-(n+k)}}{n^k}$ ln → $-(n+k) \cdot \ln\sigma_0^2 + k\ln n$.

30

2.19.05
5TM

The gamma distribution as a summary of data.

$$\prod_i \left(\frac{1}{\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \right) = \max. \qquad \text{if } \sum x_i = \mu_0$$

$$\frac{1}{\sigma} e^{-\frac{\alpha}{2\sigma^2}} = \max \qquad \sigma e^{\frac{\alpha}{2\sigma^2}} = \min$$

$$e^{\frac{\alpha}{2\sigma^2}} + \sigma \cdot \left(-\frac{\alpha}{\sigma^3}\right) e^{\frac{\alpha}{2\sigma^2}} = 0$$

$$1 = \frac{\alpha}{\sigma^2} \quad \text{so } \alpha = \sigma^2.$$

$$\prod_i \left(x_i^{\alpha} e^{-\frac{x_i}{\beta}} \right) = \underbrace{(\prod x_i)^{\alpha}}_{A} e^{-\frac{\sum x_i}{\beta}} = \max \qquad \overset{B}{}$$

$$A^{\alpha} e^{-\frac{B}{\beta}} = \max$$

We need normalizing constants, hm.

$$\int_0^{\infty} x^{\alpha} e^{-\frac{x}{\beta}} dx = \frac{?}{N}.$$

$$\frac{x}{\beta} = y \qquad \frac{dx}{\beta} = dy \qquad x = \beta y \qquad dx = \beta\, dy$$

$$\int \beta^{\alpha} y^{\alpha} e^{-y} dy \cdot \beta = \beta^{\alpha+1} \int y^{\alpha} e^{-y} dy = \Gamma(\alpha+1)\, \beta^{\alpha+1}$$

No! $\alpha!$

$$\int_0^{\infty} x^{\alpha} e^{-\frac{x}{\beta}} dx = \Gamma(\alpha+1)\, \beta^{\alpha+1}$$

$\Gamma(\alpha) \to \alpha!$

$$\frac{A^{\alpha} \cdot \beta^{\alpha+1}}{\Gamma(\alpha)} \cdot e^{-\frac{B}{\beta}}$$

A & B are known. The trick α & β so this is max.

$\Gamma(x+1) = x!$

$$\int_0^{\infty} x^{\alpha} e^{-x} = \Gamma(\alpha) \overset{?}{=} \alpha! \text{ ?}$$

$$\int_0^{\infty} e^{-x} = 1 = 0! = 1 \text{ also}$$

$\Gamma(1) = 0! = 1$ also

$$\int x e^{x} dx =$$

$$d\left(x e^{-x}\right) = (-x^2 + 1) e^{-x}$$

Normalizing for each is $\dfrac{1}{\frac{\Gamma(\alpha)}{\alpha!}\beta^{\alpha+1}}$

So we want max $\left(\Gamma(\alpha)\, \beta^{\alpha+1}\right)^{-n} \cdot A^{\alpha} \cdot e^{-\frac{B}{\beta}}$

A & B are known. We want α & β so as to max this.

The values of α & β are functions of A & B & n, so one can make

Tables. Maybe $\beta \propto B = \sum x_i$?

Geometric mean: n^{th} root $\Gamma(\alpha)\, \beta^{\alpha+1} \left(A^{\frac{1}{n}}\right)^{\alpha} e^{-\frac{B}{n}/\beta}$

so $A \to A^{\frac{1}{n}}$; $B \to \frac{B}{n}$ would fix it so we only have to max wrt A & B.

replace A & B by arithmetic Geom means.

$$\max \left(\Gamma(\alpha)\, \beta^{\alpha+1}\right)^{-1} A^{\alpha} e^{-\frac{B}{\beta}} \qquad \max \frac{A^{\alpha} e^{-\frac{B}{\beta}}}{\alpha!\, \beta^{\alpha+1}}$$

$$= \frac{A^{\alpha}}{\alpha!\, e^{\alpha+1} \left(\frac{B}{\alpha+1}\right)^{\alpha+1}}$$

$$\max_{(\alpha,\beta)} \; -\ln \Gamma(\alpha) + -(\alpha+1)\ln\beta + \alpha \ln A - \frac{B}{\beta}$$

$$\frac{\partial}{\partial \alpha} = \left(\frac{d}{d\alpha} \ln \Gamma(\alpha)\right) - \ln\beta + \ln A = 0$$

$$\frac{\partial}{\partial \beta} \quad -\frac{(\alpha+1)}{\beta} + \frac{B}{\beta^2} = 0 \qquad \alpha+1 = \frac{B}{\beta} \qquad \text{so } \boxed{\beta = \frac{B}{\alpha+1}} \text{ — correct!}$$

or $B = \beta(\alpha+1)$ or $\alpha+1 = \frac{B}{\beta}$ $\qquad \beta = \frac{B}{\alpha+1}$

$$\text{so max} \left(\Gamma(\alpha) \left(\frac{B}{\alpha+1}\right)^{\alpha+1}\right)^{-1} A^{\alpha} e^{-(\alpha+1)} = \Gamma(\alpha)^{-1} \cdot \frac{(\alpha+1)^{\alpha+1}}{e^{\alpha+1} B^{\alpha+1}} A^{\alpha} = \frac{1}{\Gamma(\alpha)} \frac{(\alpha+1)!\, A^{\alpha}}{\sqrt{(\alpha+1)2\pi}}$$

$$\text{Max} \simeq \frac{(\alpha+1)^{\alpha} \cdot A^{\alpha}}{e^{\alpha+1}} = \boxed{\text{Max} \sqrt{(\alpha+1)} \cdot \alpha \cdot A^{\alpha}}$$

$$\text{max} \quad \frac{A^{\alpha+1}}{e^{\alpha+1}} \cdot \left(\frac{\alpha+1}{B}\right)^{\alpha+1} \cdot \frac{1}{\Gamma(\alpha)^{\alpha+1}}$$

$$C = \left(\frac{A}{e\,B}\right) \qquad \left(\frac{e \cdot (\alpha+1)}{\Gamma(\alpha)}\right)^{\alpha+1}$$

So given C find $\alpha \to$ $\left(\dfrac{C(\alpha+1)}{\Gamma(\alpha)}\right)^{\alpha+1}$ is max

$\dfrac{(\alpha+1)^{\alpha+1}}{e^{\alpha+1}} \sqrt{\alpha+1 \cdot 2\pi} \simeq (\alpha+1)! = \Gamma(\alpha+2)$

$\Gamma(n) = (n-1)!$

$\Gamma(n+1) = n\,\Gamma(n-1) = n\,\Gamma(n)$

$\therefore \Gamma = n!$

$$d\left(x^n e^{-x}\right) = (n x^{n-1} - x^n) e^{-x}$$

2.19.05

α · STAT

25 α 12

$\max_\alpha \quad \sqrt{(\alpha+1)} \cdot \alpha \cdot A^\alpha \rightarrow \frac{1}{2}\ln(\alpha+1) + \ln\alpha + \alpha \ln A$

$\frac{1}{2}\frac{1}{\alpha+1} + \frac{1}{\alpha} + \ln A = 0 \qquad \frac{1}{\alpha+1} + \frac{2}{\alpha} + 2\ln A = 0$

$\frac{\alpha + 2\alpha + 2}{(\alpha+1)\alpha} = -2\ln A.$

No! $\int_0^\infty x^n e^{-x} dx = X!$ not $\Gamma(x)$.

$\frac{3\alpha+2}{(\alpha+1)\alpha} = -2\ln A.$ ⇐ quadratic in α.

$\alpha' = \alpha + 1$

$\frac{1}{\alpha'} + \frac{2}{\alpha'-1} = -2\ln A \qquad \frac{\alpha'-1+2\alpha'}{\alpha'(\alpha'-1)} = -2\ln A.$

$\alpha'' = \alpha + \frac{1}{2}$

$\frac{1}{\alpha''+\frac{1}{2}} + \frac{2}{\alpha''-\frac{1}{2}} = -2\ln A \qquad \frac{3\alpha''-1}{\alpha''(\alpha''-1)} = -2\ln A$

$\frac{\alpha''-\frac{1}{2}+\alpha''+1}{\alpha''^2-\frac{1}{4}} = \frac{2\alpha''+\frac{1}{2}}{\alpha''^2-\frac{1}{4}}$

12.10
2.26
4:45
9:12

So, it's the $\max \sqrt{(\alpha+1)} \cdot A^\alpha \Rightarrow \frac{1}{2}\ln(\alpha+1) + \alpha \ln A$

$\frac{1}{2}\frac{1}{\alpha+1} + \ln A = 0 \qquad \frac{1}{\alpha+1} = -2\ln A \quad \therefore \alpha+1 = -\frac{1}{2\ln A}$

A = Geometric mean. So $\frac{1}{\alpha+1} < 0$! ?

I must have forgotten some important property!

$\alpha + 1 < 0 \quad ; \quad \alpha < -1$

unlikely!

$\max \left(\alpha!\,\beta^{\alpha+1}\right)^{-1} A^\alpha e^{-\frac{B}{\beta}} \quad \therefore \beta = \frac{B}{\alpha+1} \quad \therefore \frac{B}{\beta} = \alpha+1$

$\max \frac{B^{(\alpha+1)}}{\alpha!\,(\alpha+1)^{\alpha+1}} e^{-\alpha-1} A^\alpha = \frac{(\alpha+1)!}{(\alpha+1)^{\alpha+1}} \cdot \frac{e^{-\alpha+1}}{\alpha+1} \cdot B^{(\alpha+1)} A^\alpha$

$\frac{2}{2} \cdot \sqrt{(\alpha+1)\cdot 2\pi} \cdot \frac{1}{\alpha+1} \cdot B^{(\alpha+1)} \cdot A^\alpha$

$\frac{1}{\Delta} \sqrt{\frac{2\pi}{\alpha+1}} \cdot \frac{A^{\alpha+1}}{B^{\alpha+1}} = \max (\alpha+1)^{-\frac{1}{2}} \left(\frac{AB}{B}\right)^{\alpha+1}$

$(\alpha+1)^2 = X \qquad \frac{1}{X} \cdot (AB)^{X^2} = \max$

$-\frac{1}{2}\ln(\alpha+1) + (\ln(AB))(\alpha+1)$

$\ln X = 2X \ln(AB) \quad / \quad = X \ln((AB)^2)$

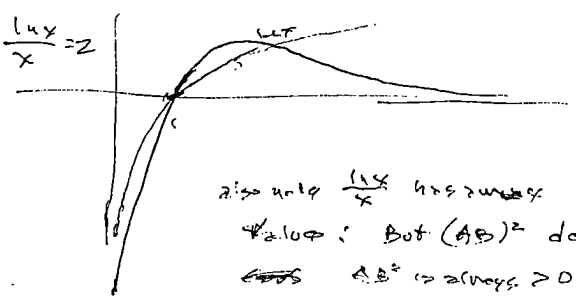
$X = (AB)^{2X}$

$X = (AB^2)^X$

$\frac{\ln X}{X} = 2$

$X = e^{X\cdot 2}$

See if I made a mistake in ∝ 11.22 ff. Maybe $\Gamma(\alpha)$ should be $\alpha!$



for $\lambda > 0$, (known sign) solve.

One problem for max, another for min!

also only $\frac{\ln x}{x}$ has a max.

Value: But $(AB)^2$ does not have a max value > 0.

AB^2 is always > 0 : (and I have given wrong)

2.21.05

5TM

25 α 13

$\int_0^\infty x^\alpha e^{-\frac{x}{\beta}} dx = \alpha! \cdot \beta^{\alpha+1}$?

$\frac{x}{\beta} = y \quad \frac{dx}{\beta} = dy \quad dx = \beta\, dy$

$x = \beta y$

$\int \beta^\alpha y^\alpha e^{-y} \cdot \beta\, dy = \beta^{\alpha+1} \int_0 y^\alpha e^{-y} dy = \beta^{\alpha+1} \alpha!$

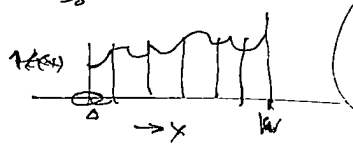
so d.f. $= \frac{1}{\beta^{\alpha+1}\alpha!} x^\alpha e^{-\frac{x}{\beta}}$: first moment =

$\frac{1}{\beta^{\alpha+1}\alpha!} \int x^{\alpha+1} e^{-\frac{x}{\beta}} dx = \frac{\beta^{\alpha+2}(\alpha+1)!}{\beta^{\alpha+1}\alpha!} = (\alpha+1)\beta$ mean $=$

B = actual mean.
A = Geom mean.

Geom mean of d.f. $f(x)$

$\int_0^\infty f(x) = 1$



$\left(\prod_{i=1}^{\frac{W}{\Delta}} F(i\Delta)\right)^{\frac{\Delta}{W}} = \prod \left(f(i\Delta)\right)^{\frac{\Delta}{W}}$ $\quad (\alpha+1)\beta = B$

$= \exp \frac{\Delta}{W} \Sigma \left(\ln f(i\Delta)\right)$

$= \exp \int_0^W \ln f(x)\, dx$

$\exp\left(\int_0^\infty \alpha \ln x \cdot \frac{x}{\beta}\right)$: T. Geom mean of any $x^\alpha e^{-\frac{x}{\beta}}$ d.f would be zero!

Max $\frac{A^\alpha}{\beta^{\alpha+1}} \frac{\alpha+1}{e^{\alpha+1}} (\alpha+1)! \rightarrow \left(\frac{A}{B}\right)^{\alpha+1} \frac{1}{\frac{(\alpha+1)^{\alpha+1}}{e^{\alpha+1}} \sqrt{(\alpha+1)2\pi}} \frac{e^{\alpha+1}}{\alpha+1}$

Max $= \left(\frac{A}{B}\right)^{\alpha+1} \cdot \sqrt{\alpha+1}$

$= \left(\left(\frac{A}{B}\right)^2\right)^{\alpha+1} \cdot (\alpha+1) =$ max $\qquad x = \alpha+1$ $\qquad \left[\left(2\ln\frac{A}{B}\right) \cdot x\right] + \ln x =$ max

$2\ln\frac{A}{B} + \frac{1}{x} = 0$

A = Geom, B = arith mean. $\qquad 1, \frac{1}{2} \rightarrow .7$ geom $\leq$ arith.

$\frac{A}{B} < 1 \qquad \alpha+1 = \frac{1}{2\ln\frac{B}{A}}$

$\alpha+1 = \frac{1}{-2\ln\frac{A}{B}}$

$\alpha = \frac{1}{2\ln\frac{B}{A}} - 1$

B = Arith mean
A = Geom mean.

$\alpha+1 \approx 2\ln\frac{B}{A}$.

This assumes Stirling Approx is exact.

So the value that we end up maximizing is $\equiv \alpha+1 = \frac{1}{2\ln\frac{B}{A}}$. Still to compute the goodness of fit,

In a PC sense it is $\frac{1}{2\ln\left(\frac{B}{A}\right)}$ maybe (divided or multiplied) by A or to that nth power.

So the f. pc of the Gamma D.f. (the best fit of data) is not used to rank.

Based used the D.f. Data sets is all $>$ (or $<$) a certain real,

---

For this calc. we need Arith & Geom means of data.

& Gaussian d.f. " " Av. R mean & mean of squares of data.

~~2.23.05~~
2.21.05
STM

25 of 14

This is the end of Prop 25 of 1 to 25 of 14 secy.
STM 26 comes next.

: on pretty data to $\frac{p(1-p)^{(n_0+n_1+1)}}{n_0!\,n_1!}$ peaks at $p = \frac{n_0}{n_0+n_1}$.

T. set of pts on $p \in [0,1]$ for n data pts, ~~[illegible]~~.

Say x_i $\left[\begin{matrix}n\\1\end{matrix}\right.$ are data: $\bar{x} = \frac{1}{n}\sum x_i$. probly $n_0 = \bar{x}\cdot\alpha$, $n_1 = (1-\bar{x})\alpha$

α should be $\approx n$. usually

so, what is α?

we want max $\left(\bar{x}^{\bar{x}}\cdot(1-\bar{x})^{(1-\bar{x})}\right)^{\alpha}\cdot\frac{(n+1)!}{(\alpha\bar{x})!\,((1-\bar{x})\alpha)!}$ ← constant

No!

$$\frac{\cdots}{\alpha\bar{x}!\,(1-\bar{x})\alpha!} = \frac{e^{\alpha\bar{x}}\;e^{\alpha(1-\bar{x})}}{(\alpha\bar{x})^{\alpha\bar{x}}\;(\alpha(1-\bar{x}))^{\alpha(1-\bar{x})}}\cdot\frac{1}{\sqrt{2\pi\,\alpha\bar{x}}\cdot\sqrt{2\pi\,(1-\bar{x})\alpha}}$$

No!

$$= \frac{e^{\alpha}}{\alpha^{\alpha\bar{x}}\cdot\alpha^{\alpha(1-\bar{x})}\left(\bar{x}^{\bar{x}}\cdot(1-\bar{x})^{1-\bar{x}}\right)^{\alpha}\;2\pi\alpha\sqrt{\bar{x}(1-\bar{x})}}$$

$$= \frac{e^{\alpha}}{\alpha^{\alpha}\;\alpha\cdot\text{const}} = \frac{e^{\alpha}}{\alpha^{\alpha}}\cdot\sqrt{\alpha}\;\sqrt{\alpha}\,\text{const} = \frac{1}{\alpha!\sqrt{\alpha}}$$

Nb! I want $\frac{\prod x_i^{\bar{x}\alpha}(1-x_i)^{(1-\bar{x})\alpha}}{\bar{x}!\,((1-\bar{x})\alpha)!} = \max$

$$= \left(\prod x_i\right)^{\bar{x}\alpha}\cdot\prod(1-x_i)^{(1-\bar{x})\alpha} = c^{\alpha}$$

$$= \frac{e^{\alpha}\cdot c^{\alpha}}{\alpha^{\alpha}\cdot\alpha\sqrt{\bar{x}(1-\bar{x})}}$$

so $\left(\max_{\alpha}\right)$ $\frac{e^{\alpha}\,\alpha^{\alpha}}{\alpha^{\alpha}\,\alpha} \sim \frac{c^{\alpha}}{\alpha!\cdot\sqrt{\alpha}}$

take ln: $\alpha + (1+\ln c) - \alpha\ln\alpha - \ln\alpha$

$\frac{\partial}{\partial\alpha} = 1+\ln c - \ln\alpha - 1 - \frac{1}{\alpha}$: $\ln c - \ln\alpha - \frac{1}{\alpha} = 0$

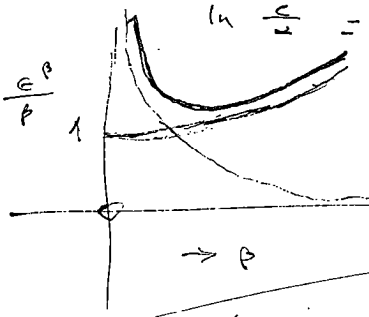
$\ln\frac{c}{\alpha} = \frac{1}{\alpha}$

$\ln c = \frac{1}{\alpha} + \ln\alpha$. $c = \alpha e^{\frac{1}{\alpha}}$

$\frac{1}{\alpha} = \beta$; $c = \frac{1}{\beta}e^{\beta}$

$\frac{e^{\beta}}{\beta}$ β has a min at $-\ln\beta + \beta$: $-\frac{1}{\beta} + 1 = 0$

$\beta = 1$



So c must be ≥ 1.

Also since diff = 0 at 2 pts, we assume 1 is max, the other Min.

$c = \left(\prod x_i\right)^{\bar{x}} \Big/ \prod(1-x_i)^{1-\bar{x}}$

so c ← [illegible] $\bar{x}$ completely characterizes the data. or $\bar{x}_i$, [illegible] x_i

& Geom mean of $(1-x_i$: 3 params.

2.21.05
5TM

26
26

o : Re: "Lecture 3": What are the major apparent obstacles to final TM?

In "Phase 1"

1) In d-induction phase: Use Lisp-like language:

a) method of implementing recursion is unclear in TSQ

b) Method of efficiently searching over corpus is unclear: I'd like a quick way to test new cases: putting problems in order of empirical difficulty is not bad. — it's a "Quick sort" technique.

c) related to (b) The "recognition function" idea in the IDSIA report, seems good, but needs much more work! Go back over it carefully! → see 28.22 for note.

2) In S-induction phase:

→ a) The 3 input UMC (3IO) is universal, I don't yet see a practical way to implement it. (The "2 input" part is "under control")

b) I have a way to get 2 kinds of probabilistic output of reals: ① Betw. 2 reals say from 0 to 1 use Beta Distribu. ② from 0 to ∞ use Gamma D.F. ③ from −∞ to ∞ use Gaussian D.F. These are all monomodal d.f.'s. — But these methods are not universal.

It may be that by using "a uniform D.F. from 0 to 1 as one of the problem data" as another input to a universal lang (Lisp, say) I can get any output D.F. This idea may be able to solve/clarify problems of spread in, say, "How Many bits" to the linear & non-linear.)

c) I don't have a good way to get probabilistic D.f.s on string outputs. Some ways ① Output is a single string. We use some kind of distance function to other strings, to get a D.f. on other (non-central) strings. Initial output string can be center of a cluster but not part of the cluster (?). The distance function can be: Edit dist; Hamming dist; An "info distance" using instruction codes. This last can be approximate (not use full inst. set as in "Edit dist") or fuller set, & "heuristic psing" (Newell-Sim. GPS) to create desired corpus strings (expensive). "T. Cover's distance" may be essentially identical to the "2 input UMC".

3) In Both D & S Induction of Phase 1:

a) I have 3 approaches to using rings of past:

① PPM; May work a bit; but would be much better if sub-trees could be recognized as "contexts": I have a way that would probably work, but it seems very slow.

② S-CFG discovery! I don't know how good this is — to what extent does it recognize sub-trees as contexts or as "chunks". Also, I don't know whether it works very well, & I suspect it is slow. I could use S-Context Sensitive Grammar w. context being in form(s) of sub-trees.

2.21.05
5 TM

→ A big possible trouble: that selected that ratio may not be monotonic in x_0 — (i.e. Threshold or expunts). a/o be a bumpy, irratic function of "x_0" (or expunts).

: (3) "G.P." for "ranges of past" — use of S-cfg for summarizing corpus seems good. (see 26.35 for some diffys w. this):
Use of $\frac{\sigma_G}{G_{max} - \mu_G}$ as criterion for threshold selection — seems good, at first glance, but is very dependent on G or c used. Here G or c is PC, so its unambiguous.
T. Q is: will this formula work w. that G or c? — It may, because [illegible] this G or c may be "linearized". → 28.10 discusses Phase I O. Induction TSQ's.

Phase II

— 1) The induction for Phase II involves string to real distributions. They are usually monomodal, so I can use either Gauss or Gamma D.F. for output: This means t. output need be but a say to real's (usually = arith mean of D.F.), I can then find params of t. D.F. to give best fit.
[Output is P.D. of G for application of PSM$_i$ to problem$_j$ for a certain fixed time.]
IDSIA has an analysis of this. T. system assigns a PC to a given PSM, depending on how good it has been in predicting G values in past. (Ideally t. algm looks at (PSM$_i$, Prob$_j$ & t) (t = time allowed) & gives P.D. as output.

I haven't much look'd into practical aspects of realizing it.

— 2) I want to have a grammar to generate PSM's. Seems like a very interesting "problem in itself". It could be a cfg — am not sure it need be a "Turing universal" grammar in order to generate t. universally describable PSM's) In fact, a Loopless Grammar may be adequate. ??

3) It may be possible to do phase II first w.o. doing phase 1 ever! → A counter Arg't, that Phase II's induction algm. would be req'd to have experience in many kinds of problems — e.g. those in Phase 1.
I haven't look'd into this much ... I have no idea what t. diffys are. I originally had t. idea that a fairly good induction system would be needed, say a well developed "Phase 1 S-induction" system ... but this may not be necessary — But .20 does have to be worked on eventually. If I did it now (first) & .23 turned out to be feasible, I'd save a lot of time!
Also it would be solving Phase 1-type probs so it would eventually get that info.

4) I want Phase II to be able to do arbitrarily good self-improvement.
— Which is t. main idea of Phase II! Whether t. system of .23 could do this, is unclear; Well, it is able to improve search algm for estimates of Expected value of G for (PSM$_i$, Prob$_j$, T) → 33.20

Logical Reasoning: Much used in "Heuristic PSing", Math, etc: I haven't gone into it at all, hvr. — & it's prob'ly an impt. part of "intelligence". T. plan was to have TM lrn logic as a Math'cal study, then learn to correlate it w. Real Situations that would arise in other kinds of problems.
It seems likely that Logical reasoning would be impt. in solving Phase II's problem of allocating PSMs to probs.
Also creating new PSMs.

20 : : 26.00 – 27.40 is a kind of "outline of State of TM": I would do well to "flesh out" the outline by bibl. refs. In particular, it would make a good way to organize my notes! Make special indexing for ⟨notes not in the outline⟩.

[**SN** | **NB** Look at Abstract of IDSIA report. This is an Earlier Version of the present "Phase I, Phase II" system. Using the tricks of PPM, PSG Dizzy & Enhanced GP: would it also be workable?]

0 **Re: Phase 1 diffies!** Writing & learning of TSQ!

Possl. initial TSQs: ① from Elementary Algebra Book: Notation: Function Evalu: (Soln. of linear, (open diffy) Then N.linear equs) of greater complexity, Differentiation, (open diffy) integration.) definit, indefinit, numerical, ② Learning of Notation of Maple, Mathematica etc. → soln equs, integration.

Will PPM' (augmented?), PSG (augmented?), G.P. (augmented) be adequate?

Will ~~[illegible]~~ contemplated TSQ's be adequate — are there many more "intermediate concepts" needed in learning than we anticipate?

It may, indeed, be that we cannot get far enuf in Phase I to get to a useful Phase II! → Note that Phase II needs $-induction w. numeric output. This, but suggests no TSQ's: SM & /o HR would seem to be reasonable TSQ's. 26.12–20 discusses how to implement

-2 ~~Not mentioned~~ (in 26.00 – 27.40) (26.07 does mention it) is the "Recognition function" (= "Cases" function) of Early part of IDSIA report. These Recognize problem types & allocate an appropriate set of O^J functions to them. This markedly reduces the amt. of time needed to check each O^J modifn. on past cases: Only cases of appropriate R category need be tested. This Accords w. much "Human Scientifying".

30 | SN | **ON "A·LIFE"**: A-life is interesting & diverse because we almost immediately get into a great diversity of problem & solution types, brot about by the diversity of life forms itself! It is competitive and then competition is strong, diverse, & usually of same level of diffy as the competing forms. The life forms solving problems are themselves available (via parasites) as **Parts of** solns to new problems.

[redacted] STM

Karl Newell : [redacted] coauthor } T. Sci Community as an Intelligent System } ≃ MIT Report

! Sumac revisited: keep 100 best models; but for search, use Grammar or PPM to extrapolate search so one doesn't lose diversity as TSP continues. Try to keep diversity at a certain minimum level: at least Parkman. at all times.

Could .00—ff be "brief soln" to TM? — See .05 ff for objections & suggestions to overcoming objections.

05 There may be several distinct ways to "↑ Diversity"! If one doesn't ↑ diversity, one saves best 100 solns. to Induction problem. As TSP continues, fewer & fewer of the models fit the data "acceptably" — until none do.

07 One way to ↑ diversity (at least fast enuf to compensate by loss of diversity via .05—.07) is to make stoch Models of good cands more far (viz S-CFG or PPM). This gives a large Soft (probabilistic) language of cands that can be very "diverse" — but not very "clever". Could I formulate the creation of a set of "clever" diverse cands as an OZ problem?

The Amount of CC spent on S-CFG discovery a/o Augmenting PPM could be regulated by the amount of diversity needed.

NB Increasing amt. of time spent "updating" PPM: S-PSG discovery, will ↑ accuracy of predictions — is this necessarily tied in w. "Diversity ↑"? ⇒ .50 THIS MUST BE ANALYSED.

Another trick: I was a bit uncertain about how to model a Gen (≡ "Littlest Gund") in ways that would (1) ↑ speed of comp. of Gen, of test approx (2) Enable inversion of the Gen to find Cands of by Gr. In both of these applics., it would be well if TM could "watch the Gen being Computed" — or be given the Alg. that Computes the Gen, — which are a "No-No" in Genetic/evolutionary prob. Solving, because it's much not like Org. Evoln.

It would be well to attempt modeling a Gen on "reversible = invertible" funcs, so that we could more easily implement search for Cands of by Gen.

30: (20.) One method to ↑ diversity was to store "best 100 or best 1000" cands. at all times. In S-PSG discovery the Cands that branched away from each other, early in Hill climbing usually will tend to be diverse. This is also true of cands obtained by branching early in TSP. However, if one discovers one hasn't enuf diversity, it is probly not practical to backtrack back to the beginning! One could Backtrack back a few critical paths, however, and one could retain more early branches in S-PSG discovery (say, retain all early branches until the no. of stored nodes was 100 or 1000 or whatever. One could store 10 000 early branches & then slowly winnow them down to only 1000 or 100 as we work toward an acceptable set of cands.

This is domain for Sci Community

20:28.40 ! ALIFE: cont: One way to get it to solve more practical problems: Make Environment have features that one can learn to take advantage of. E.g. T. rules of reproduction are always, slowly changing. Any of the rules about getting food, consuming other organisms, etc, are subject to slow change — (perhaps "randomly"... unclear as to how 'random', & "random in what way")

Part of this is analogous to peoples in varying climates developing more rapidly than those in constant (benign) climates.

While I forget: is there a fund! still, how do we get the population to solve what are to the designer, "useful problems"?

Perhaps by experiments, the designer could try various "rules (inventions changes)", to see what the community is able to deal w. effectively.

Mite starts by giving equations to solve. For a soln., individuals gets more food/reproduction rights or whatever. We would then want individuals to be able to help one another: possibly complete as well. This "equalizing" reward — would want it to fit in w. other "normal" kinds of competition within the ALIFE community; Perhaps have a "patent system" so a person gets credit every time someone uses his idea(s). — But have reasonable limit on duration of patent ... so it soon becomes "Public Domain".

could .13–.16 be adequate by itself? The fraction of "reproductive rewards" obtainable by "solving problems" is an important param of the system.

I have to work out mechanics of presentation of problems: realization by population that they get rewarded for soln, etc. Mechanics of TSQ's, etc.

30

Criticism of [adequacy for Strong A.I. of "A proposal for Strong A.I." .00 ##]

In Phase 1, "an adequate TSQ" ; adequate "ragy sych alsm" means only that th. subfunctions) needed for a soln. to th. new problem have been defined (or equiv.) — so th. soln. has a not to small a prob. This can, for many large problems, be still very small.

A cleverer machine will somehow be able to consider th. exact nature of th. problem & assign very large pc's to certain functions; also by "logical analysis", come upon a soln. w. pc = 1 (or very hy).

It is .00 - .03 that characterizes a "successful" Phase 1. Its about as far as Phase 1 can get. It assumes all cognit. "subtasks" (≡ subfuncts) can be found by the "ragy sych" [Ragy sych is wrt. corpus of past successful cands]

There are, then, 2 sych types in Phase I: (1) Ragy Sych over past successful cands — to find useful sub-funcs. (2) Search over combinations of declrd funcs to find solns to problems. ||| T. Q was: In which combo(s) of these was Phase II to be applied?

(2) can use contexts of various kinds. (1) could be augmented by "context" — th. contexts of th. subfuncts that have been found.

Both .19 & .20 could speed up TM considerably. Since "context" in its most general form, involves th. entire previous corpus & any ragys on that corpus — that "context" can be quite powerful). (tho of course, contexts of progressively greater power/complexity become progressively more d.if. to find).

It would seem that th. 2 sychs at (.16 - .19 / .00 - .03) would be rather well declrd. & not hard to do in a near optimum way — they are rather "simple" sychs. (Tho it may be poss. to speed up detection of sub-funcs). S.CRG d.scy a/o. augmented PPM might be adequate. I guess I was thinking that phase II could detect more ragy types using focused contexts (.19 - .20 ; focused context. .23 - .24)

Any "context" detected by "ragy sych" should be translatable into a way of searching (≡ sych) cands more efficiently — so th. 2 sychs at (.16 - .19 / .00 - .03) must of necessity be closely related. Essentially th. "ragy sych" is creating a stochastic Grammar

Context Sensitive Grammars (speccal) : 32

00:31.40 : to generate subsequent cands. A "perfect" Phase I would use full ALP to extrapolate past set of cands (= Bag induction) to propose future cands.

Tho this would be "not bad", Phase I still doesn't understand what "optimization means! — So clearly Phase II has much to offer!

A pluperfect Phase I would use ALP to directly extrapolate past $PS_1, PS_2, \ldots, PS_n$ re (Q_{n+1}, A_{n+1}).

I guess I'm still reasoning on Ph II's all over operation! I did want to construct a "complete" grammar over all the PSM's I know of. — But the method of deciding which PSM was appropriate for each problem, is (perhaps closely linked) problem of discovering a new PSM appropriate to a new problem .... Phase I haven't much worked on.

One kind of "perfection" for Phase II would be if it gave v.g. P.P.'s for $(PSM_i, Prob_j, T)$. However, even so, we have normally, not such a good set of PSM's i.e. there is the problem of finding PSM_i's that have good "fig. of merit" for their P.P. So we need ways to generate Good PSM's for a given prob$_i$, a way to find P.P. v.g. PSM's for that prob$_j$.

3.1.04 SN T. problem of finding common sub-trees in "successful cand corpus", would seem to be identical (or almost identical) to the problem of discovering s-CFG's: except in s-CFG discov, we also look for "Merges", which can give loops. So maybe s-CFG d. is uniformly better than simply looking for common sub-trees.

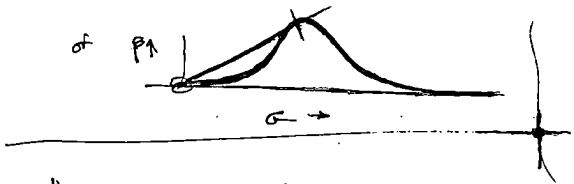
Common sub tree stuff is also done as a (serious) enhancement of PPM. Is there any way to merge CFG & PPM to get a system uniformly better than each? I.e. CFG doesn't seem to use context (much or at all) in choosing substitutions. Context sensitive Grammars would help much; But I have to make sure the proper kinds of contexts (i.e. subtrees) are used.

Cure Cancer! .II — PHASE I What needs to be Done .00

: What needs to be done for Phase I!

1) Write pgms for PPM, S-cfG disc.
2) Write TSQ's { First for D. Induction / Next " S. Induction
3) Augment PPM w. subtree Contexts:
Augment S-cfG to S-cSG dirty w being subtrees or Contexts.
← { Both imply kinds of Augmentations / PPM one has been started already: T: S-cfG → S-cSG has to be figured out on More
4) Try improving GP à Getting it to work w. interpolation rng,
Then TSQ's, Then various improvements of GP →
I'm having second thots on σ_G GMX−μM as very useful // It needs better theo. understanding! : If G = "linear" it ought to work à G=pc (or lupc) ought to be linear.
(4.5) As is Phase I can only have numeric output for S- ind (But could have string output for D-induction) 34.2 (in 3 mc) may make string output possl.
5) Work on Phase II.
6) T. "Cure Cancer" problem is Very Impt. Any long-term research that TM does — internally also in R. Mod. involves solving that problem.
It may be easy to devise a TSQ of "cure Cancer" - type problems of ↑ diffy: E.g. slowly ↑ "horizon" (Horizon = no of steps doable before finish). Or no of possl. steps; diversity of results can be "Adjusted" for the TSQ.
My "Soln" of "Cure Cancer" may be a bit complex: I may be able to significantly simplify it by expressing it in Recursive form. : Lots of work on "Approx Dynamic Pgms" started by ..... Look up "Approx. Dyn. Prog" on Google.
7) Testing each O'ni cand. on the entire previous Corpus is very inefficient. Several approaches:
a) The Recognizer/actors of Early parts of DPs still report b) Putting past probs in order of diffy. ... c) ≈ Problem Pool approach.

: 27.32. Phase II (cont): 5) It should be possl. for TM to look at a problem, à accurately (or somehow get a gd on all possl. PSMs) one or more PSMs that would have hy FOM (Fig. of Merit) for that problem j. i.e. hyθ ↑G ?



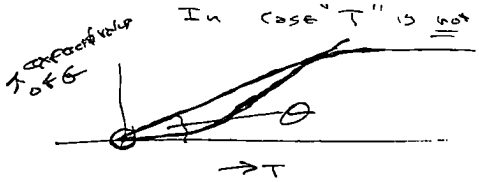
Actually, it may be O.K. to select PSM w. max μ_G for $(PSM_i, Prob_j, T)$. — T. details of t. D.P. are usually not of interest!

Anr. to get t. μ's $_{i,j,T}$ I may need to get t. entire G v.s. pc curve for each (i,j,T).

— This is so I can best fit t curves using negative as well as positive data.

It may be that I can get a simply approx formula for μ, given t. set of G values for past successes & set of G values for past failures.

In case "T" is not given: I may want to work first on PSM w. Max θ!



3.3.05
STM

2.5. — 33 TM 19
3-1 — 14
11
25 B
24 d.

Phase II: .10 → .31 ← impt. Idea of Ph II! Its General Dean.
3 Input UMC .21 : Breakthru? ?? ← looks Good!
Phase I .00

.10: SN : After a successful cand search; — yielding 1 or more acceptable cands — Th. following "regy search" examines those new/successful cands — expecting to find novel (cones/sub-functions). I.e. How do t. new cands "differ" from t. older "non-successful ~~cands~~ on current problem" cands?
.23

"world tour."
N. Zealand
Aust: Maybe 2 or 3 places Monash, Sydney, capital of Aust
RHUL London
Lugano?
MIT
Calif. — Some how — Ask McGill Stanford, or Lorez.
China
Japan
Singapore (hot!).
India.
Santa Fe
IBM Almeda?

.10: **ON PHASE II: Does Ph II really Understand "what Optim means"?** [very well] How much Better is it than Phase I in its "Understanding"? A good Optimizer will look at an OZ problem, consider different ways to solve it, decide on one on t. basis of max expected G in t. available time. This is what Phase II does.
I guess what I felt was "missing" was: we gave TM "$G = 1 - x^2$": to find $x \ni$ G is as large as possl. (in available time). In t. present case, a person would usually do $\frac{dG}{dx} = 0$ at $x = 0$ as a cand. Then try it, then try $x = 0 \pm .01$ to a final max.
TM could have a general second order n-linear optim routine "built in".
— Hvr. I'd ("also") like TM to be able to discover routines like that "on its own" → .31

.24 **On 3IUMC** (for String Output): We can simply use a general Lisp function w. 3 inputs
For paolung, "easy" problems, searching over t. "R" input shouldn't take much time, since solns will be of type. A Bern seq could be easily done this way. (Hvr, do consider
.24 t. soln to t. Bern seq. obtained by integrating over all possl. p vectors: T. General Beta distribution. $(c-1)!\prod n_i! / (c-1+\sum n_i)!$) → 35,03 → 35.00

On Reparsing in QATM: [solns] Very impt thing to do! T. set of reparses is t. Cart. product of reparses for each QA problem soln. — (Well maybe not such a great idea! — T. "individual solns" are each for a subset of t corpus, & the subsets "Nest"..... A different kind of soln. i.e. QA problem has many solns. to various subsets of t. QA corpus → 35.10
Maybe a better way to solve QA problem — close to "Problem Pool" method.

.20 **On PhaseII**: After TM has collected {a lot of "parts" of PSMS (PSMS & partially assembled PSMS, ~~This is in a good position~~ & data on FOM of various PSMS for various probs} W. this data, TM should be able to construct PSM's that are likely to have hy FOM for a particular problem. ← This Expresses PhII's Goal as an OZ problem, not [illegible] as a PhI problem!
34

Could PhII's main problem be worked on by P&I's "regy. srch" algm(s)? Well, I think G.P. can work on it. G.P. being a type of BU srch. — This isn't a "real" "OZ" srch, hvr. : Its a "Better than" srch.
I could use a search over a finite no. of PSM's [as in IDSIA Report] to find t. one w. best FOM for a given problem. This is a step toward .31 - .34: It could be used for t. OZ problem of .34 — so this would be a "soln" to PhII — tho a very "Round About" soln!

3.5.05
5TM

○ (Spec 34.24 .23) : **Sizumes** (cont.): 34.21-23 seems like it ought to work! (so Universality for String Outputs).

For Zims; numeric output: T. output could be the mean & var. & t. R input would tell how many s.d.'s away t. desired "A" is. ← (The Prior is not clear in my mind.)

○ 34.24 I still don't see how to unify It's treatment of t. Bern seq. w. t. Beta function formula Approach.

6✓

2.I 4,6,7 6

2.II 1,9,5 9

3-I 10

3-II 6 6

4-II 6

1) APL recursive?
2) Tail. rec. v.s. Prim. Rec.
3) Efficient defns of Post. rec. funct.

○ : 34.30 : T. "Repairing" idea is related to t. tradeoff betw. making "definitions" & using large constants (like PPM) for Preds. I prob. want to do Both ... but this somehow seems "inefficient"!

○ : I. Witten & E. Frank : Data Mining — Practical Machine Lrng tools — 2000
Discusses t. WEKA data mining System: I may have this c/o its on t. web. David Lindsey used these pgms to implement his "Improving Reliability ..." paper. ← see WEKA (Google).
Also: <"r" data mining"> Google: first hit. (Download a 120 PDF of "R" handbook. "Data Mining w. R")
Also "MySql"

○ : T. Langs I've Considered for PhI! Lisp, Forth, assembly, APL:
1) Is APL recursive? If not, can it be easily "recursivized"?
2) There are forms of recursion that don't use a stack, that are much faster than usual "stack"-type langs
Are they capable of full recursion? — can they express Ak. funct?

One way:

```
Rec.funct(z):
Do   x=f(x) , y=g(y)
until  y = z
Rec. funct = x
```

Modulo: we can count t. recursions n (i = count)
& x = f(x, i) , y = g(y, i). — Or various
mods, as: y = g(y) ; x = f(x, y).

3.5.05

5TM

25 $\alpha(\frac{1}{3})$ is found [illegible] bet. TM 25 & TM 26.

.25 α 14.40" → : On t. "Hessian" : T. Hessian is specified in the "decrate" part of t. corpus D ... This includes any "dimensionality" into that may be needed for each coeff/param of t. Hessian.

(25 α 8.01 Spec)

12 Midnight 3½
325
555 2½

SS 66.2 kg
3.6.05
Say − 1 lb for Pants, sok, Shirt.
65.6 no clothes
144.32 lbs
.6 kg for clothes = 1.32 lbs!

: Partial recursive. v.s. Prim. recursive. In fact, we never use partial rec functs: We use part. rec functs w. a "Computation Bound". This makes them "Total rec" — but are they really prim. rec (i.e. necessarily via "Do" loops?)?

Look up Chris Moores website! He has a very clear paper explaining Prim. rec. v.s. part. rec. v.s. total rec. functs. My idea of prim. rec. involving Do loops is a bit uncertain: I think this paper clarifys it. If I can't find it; look up "Chris Moore" papers in "PS" directory.

: On "Phase II"! Instead of t way I "now have it planned" (— Not such a good plan! — see 34.34!) Could we teach TM what OZ is, by example? At first I thot that TM would simply try to "imitate" t. "example" & miss t. essential point. (I'm not now clear on why this was so). I later found that I could modify cc's of various insts. so as to make "copying" not an option: — In general, I think I have to review this..:

Say I show T.M. various OZ prob. defns ($M_i(\cdot)$, T_i), & either simple solns — i.e. t. value of "x", or t. actual prog. that found t. "optimum x". UNFORTLY. From t. either of Phase I set 2, TM would not be able to genz. t. "optimum" concept optimiza..

Say we give TM several OZ techniques for a given problem. All but one are negative cases (I think we can get t. Phase I to use neg data)

. 34.31 is about t. best (treatment/understanding) of PhII thus far, but it's still "incomplete" (t. diffy of 34.34).

33 [Mainly We can devise various OZ methods: (Avery (large no. in "Literature"). Perhaps use one of these (not really optimum) OZ methods on 34.34!

One kind of OZ is a general "Hill Climbing" problem: We try several PSM_i's on a P_{GMO}, & we get their assoc FOM_i's. From this info we are able to construct new trial PSM's, & their FOM's, and so on to "peak" in available time.

. 40 T. problem of: Given a bunch of PSM_i, P_{GMO} → FOM_i pairs, to guess at a new PSM of hy Fo(M) → 37.00

12:90
2:10 1:30
2:07 1:03
433 1:16
623 2:10
8:22 2:40

Phase II (runt)
0:36.40 is a standard H.C. / G.A. sub problem. .005 We can do it our tricky way: to approximate t. FOM_i => PSM_j, Probs relationship by a fast function a/o an invertable funct, so we can get PSM_j's off FOM_i's. This trick can be used in G.A. or any other H.C. Scheme.

Another Approach: T. max of a finite set of nos. is a clearly defined function; Could we ask TM to extrapolate Phase II to an ∞ of nos?

T. problem seems to be "Recursive": if we had a OZ algm, we could use it (1.234-31-34)
better
og to get a better OZ alg. So perhaps we write regard 34.31-34 as a OZ algm. "In power" —
o that could be applied many times to some initial OZ algm, to get better & better OZ algms,
og -.10 may be adequate! Go Over 34.31-.34 & 5(.09-.10) to see if it really can/Work!

Eventually, we would discover, for long-time use, a v.g. OZ algm for 34.34 & this would solve t. problem. So (at present), I think .09-.10 (& 34.31-.34) may be adequate for Phase II.

We note that t. OZ algm. for 34.34 does have to be significantly improved during TM's life... since at first that algm. will be able to optz over only a finite set: Then, eventually, it will have to optimize over an infinite set of PSM's (that it, itself generates). It is at this point - t. transition from finite set to ∞ .... that t. "hard inverse" occurs.

We would need a good TS ⓐ to get TM (to learn) to do this particular finite set → ∞ set induction.

20
21 3.7.05 36.40 — 37.005 is a view of PhII that is a standard H.C. problem. Grammatical induction (Augmented w. context sensitivity ...), PPM (augmented w. sub-tree contexts (& prunings)) could be useful tricks for this, as well as G.P. — or any other methods I'll be using for PhI regy srch.

So, the PhI regy srch is/it so hot, we are using it at a META level, to choose good PSM algms, so it is effectively Much Better

Note that both PhI regy srch & this PhII (secondorder PhI) srch are capable of creating new objects w. hy expected Gore (or "FOM") — but, presumably, t. 2nd order PhII srch is potentially Better. It should be possl. to go up one (or more) level(s) higher ... to get even better PSM's: but it may be too time consuming for t. amt. of "Goodness" obtained.

b Note that all 3 methods contemplated for PhI regy srch (PPM, CFG, G.P.) would work as well for PhII PSM (creation/srch) — So any improvement on PhI regy srch is usually applicable to PhII PSM srch.

So: .09-.10 may be fine, in general, but using a PhI regy srch for PhII PSM srch/creation is a simple, probly good soln

→ A Big Problem is (estimate) eval'n of t. FOM for various PSM's that have been created! (most all PSM's we create will have zero ssz! So how to get FOM? I guess by an inducting process. We have lots of (PSM_j, Robj, T_k) → FOM_{j,k} examples, we just have to find (inductive) algms that extrapolate to new tuplets

3.7.05
STM

38

38

Sand stuff on Consciousness.

Aug 777

20 : So **37.21** ff is prbly a more or less adequate Model for PhII · 34.31–34, Rom Perhaps 37.21ff. I suspect that it may be close to what I wrote in IDSIA Report:— except for recent ideas of using (PPM, PSG diag, GP,) for Phase I regy srch & also for final 34.34 srch of Ph II.

I do want to see how far I can go w. those 3 "wash inducers" for Ph I regy srch.

35 Look at Dern of Ph. II in IDSIA: See what has to be added. (o the Ram) → .30

The points of 33.00ff suggest that Ph I may be not far from theoretical realization!

Perhaps TSQ's are most critical, diff't. problem rite now. Perhaps try t. teach maintained on Sol of taking a diff't problem & working out a "concept net" to solve it: This could be done for D-induction ("Math" problems).

RE: TSQ's: At one pt. I wrote a bunch of things that I expected TM would need/want to lrn preparatory to "lrng Algebra".

An alternative approach would have TM lrn. t. meanings of various commands in Maple/Mathematica and & be able to execute them.

0 : Relooking at IDSIA Report: Abstract: What was derbd was Phase II only:

PhI was not derbd in "Abstract":

Re: Abstract: Some updating can follow each problem Soln but w. each new problem, we may want to augment t. PSM set. in t. direction of that new problem.

Was there anything in the Report that told how PSM's were added/deleted?

0 (.05) → I think that a real Understanding of Ph II would occur if I could write a TSQ for it that would really be able to discover "good PSM's": (These Good PSM's could be normal part of TSQ — designed in by "Trainer")

"Logical Analysis" must be impt. part of this, hvr, & I don't know how it will work

→ Is it possbl. to do "logic" as a kind of probabilistic reasoning? → Perhaps!!

3.13.05
5 TM

0 : [SN] In Math, there is a thing called "Non-Standard Analysis". calculus & continuous concepts using only discrete concepts: Edward Nelson devised a PC scheme for it. 1) Non-standard Analysis — Nelson; PC.
Look into N.S.An. It may bring a nicer continuum book. Study output & Num output of # S tracts. T. models I'm using now are not so hot! 2) How Ph II is ~ to Ph I.

.04 [SN] Ph II can be regarded as a slight generalization of Ph I in following way!
Ph I initially solves only probs that are Maxzn. of PC. It does this by looking at functions that have been good in past, making S-gramm. for those functs, & using the S-gramm to generate trials; Either in Lsrch or BU mode.

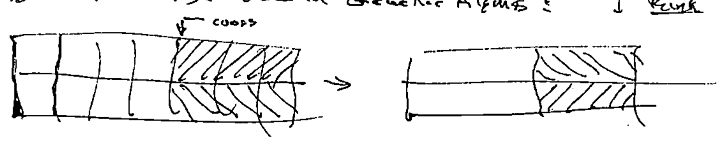
0 We can do t. same thing in Ph II, but instead of PC's to maxz, we have to maximize t. FOM (fig of merit) for $\{PSM_i, Prob_0, T_0\}$. $Prob_0$, T_0 are fixed, so its a d.f. over PSM_i. ~~[struck]~~ PSM_i should be t. set of all possl. PSM's, just as in Ph I,
.3 t. arg t. is A, which is t. set of all possl ~~[struck]~~ answer strings. → (.30)

Some possl. differences: Th. Gorc for Ph I may be "linear"; t. Gorc for Ph II ~~[struck]~~ is not nec'ly linear: it is, for OZ problems, t expected value of an arby Gorc — & that Gorc need not be linear. It may be actually Mandatory that we linearize t. Gorc before trying to Maximize it. !

.10: (.75) → Anyway, for this Phase II induction we do have a suitable corpus of $(G, PSM_i, Prob_j, T_j)$'s (A_i, Q_j)
(.30R) Then gives a pure Ph I problem, since pc's are assigned to these A_i Q_j pairs.
We still need something like expected value of G — which does seem we need it linearized if we are to try to find optimum strat. to get it. (This is discussed/mentioned in footnote in IDSIA Report.

.27 For Phase II INV problems (usually solved as OZ probs), t. Gorc is automatically linear! we want to minimize time to soln. → (40.20)

.3 : BFI (Boys from Ireland) ~ 3 Irish authors: O'Neill, Ryan, Collins. Use of Bach Naur form to generate strings (usually pgms or functions): Use in Genetic Algms: I think t. troubles, their "Crossover"
D:\PS\Grammar Discovery\Grammatic Evolution



As I see it, usually everything to r.t. of crossover pt. is essentially random, because t. choices it makes after "crossover" are usually entirely different from t. choices it made before crossover.
Occasionally a crossover will "preserve state", in which case we will get "real" "crossover".
Their use of "wrapping" is equivt. to having t. finite chromosome repeated ∞ times. So each chrom is ∞ long.
It would seem that Kozas "GP" would be uniformly better. → spec 40.00

3-13-05
5 PM

and lives?
Petersen N 97
Evolving Caching Algms in C by GP

00: (Spec 39.40) : They pride themselves on being "closer to Natural Evoln" — this is good, only if you understand the feature you copying. It may or may not be appropriate.

In the set of papers in "Forum Evoln" folder:

[2.ps] — contains most info — is 13pp long! other papers were 6pp long.
discusses "Prune & Duplicate" 2 operators that make it different from normal GA.
In this Paper they also discuss their "crossover" method.
"Introns" are bytes of chromo that occur after the function has been completely defined.
By Pruning, they remove these "introns" — so "crossover" is more likely to do something.
(A crossover at a "intron" produces no effect at all on individual).

0 By "gene" they mean "Byte" in chromosome.
Their discussion of "Gene duplication" sounds insane — if my interpretation of what they are doing, is correct. Any dup of a gene usually will produce a completely different effect than it did originally.
They say (7.ps) that their Caching algm is clearly better than that evolved by GP. (!)
Seems unlikely! — Unless I seriously misunderstand what they're doing!
5.ps "Under the hood" analyzes how it is (supposed) to work.
I downloaded 6.PS & a winzip file — both ≡ 5.PS.
4.PS is very short nothing new I think.
6.PS is big file but only 7pp. 2.ps is 11 pp — Most info.

20 : 39.27 : Actually 39.50 is a bit deceptive: while the corpus for the needed induction is, o.k., what the PHI output is ··· a Pd. on $\{PSM_i, G\}$. The actual form of this output may not be useful. i.e. we may have to put in various trial PSM_i's and various G's & get Pc's out — or many other not so useful forms.
What we'd like is ≈ either we put in a try PSM_i in & get EG for output, or ideally a list of PSM_i's in EG order — largest first.
This "Form of output" is certainly a major problem in all "s function" calculations.
29 Getting what we want could involve a lot of (expensive) srch! → (41.00)

30 [SN] On ngms (or whatever) of (unusually low empirical freq.) I never really found a good way to use data of such ngms: An Alternative View. If a given ngm has "unusually low" freq. in a given Context, then this is a "hint" to look for other ngms at that context having unusually hi freq. — This will ↓ length of code

3.13.05 — 41

5TOY

Rev: .20 — .40

8:26

0: (Spec 40.29) : Actually, that Big Srch cost May not be so bad!
It may well be that we end w. only a small no. of PSM's, but we do constantly spend time looking for (new/better) ones.

[SN] Back to an old Q: Could we teach TM what Optzn is by a suitable TSQ: (by a suitable set of examples)?

One kind of TSQ: An OZ prob. descr. is given $G(\cdot), T_0$. Paired w. it is a pgm to solve it. (Q_2 … A_i)
We give PhI many of these (Q,A) pairs. Would it be able to induce what was needed in A_i?

A trouble is: that the examples given are not necly the best possl. ways to solve the particular problem. → Perhaps Main Q: Can T. transfer think of a good recursive defn. of Optzn? } CRITICAL IDEA!
If not, it's unlikely that Trainer can devise TSQ for "Optzn".

[If I have a clear idea as to what I mean by "Optzn", it must be possl. to ⓐ Get a recursive funct. to Express it. ⓑ Get a TSQ ∋ TM can learn it.]

→ Hint! That PhI doesn't understand Optzn! It's soln. methods are indep of "Horizon", T_0.

0: How has advanced in TM, since Sol 89?

1) ~ ALL PGMS from "one name of technique of PhII: a unified treatment of Soln. of INV, OZ probs. ("Mixed Corpus Theorem"?) (MCT?)
2) Understanding of how Logic fits in: (42.13)
3) Understanding and proof of convergence of "Big Corpus", "QA Corpus"
4) Use of QA form for PhI simplifies TSQ design
5) Understanding of what kinds of things mite be useable for probs w. String vs. Numeric output. This still needs much work. 3 forms of Numeric output (Gall Hono Model).
6) The use of PMM's PSG dscy are but slight improvements in my understanding of how to search for solns. in PhI. G.P. may be of interest (particlly for BU srch) if $\frac{\sigma}{G_{Max} - M_G}$ works. — But certainly I was aware in '89 that srch for sub-trees was impt.
7) Understanding of 2 modes of Srch: L srch v.s. BU srch! when each is appropriate.
8) Listing of Characteristics of STRONG A.I. that only ALP seems to have (Frustr):
1) Accuracy for small ssz. (smaller for ssz) 2) Diversity shall codes 3) Turing Complete ∴ partial rec. funcs
9) Understanding as to why partial rec functs necy — They have higher PC than corresponding prim. rec. functs.

3.16.05

STM

Logical Reasoning via ALP : 13

0 [SN] On Lsrch v.s. Bu srch. Originally I had idea that if I knew tsQ was designed by trainer for Lsrch, I'd use Lsrch only. If I was uncertain about whether a Lsrch soln existed or that it likely non-existed, I'd use both Lsrch & Bu srch simultly.

Q: Could I get any extra diversity by using Bu srch && Lsrch all the time?

—— Ans: No! If Lsrch soln is available, it has >> pc than Bu soln. & would give cones much much better than those of Bu srch.

If TM can't find a Lsrch soln. to a problem, this means it didn't ... the conc(s) that the trainer intended — & Trainer should try modifying of TSQ so TM can "get" required cones. ("Hints" may or may not be what's needed!).

10 [SN] on starting TM on PhII (No PhI): PhII really has to solve some hard problems if we want it to really improve the set of PSM's we initialize it w. It will (very prob'ly) have to understand "Logical Reasoning".

13 [SN] On logical Reasoning [BREAKTHRU?] In past thot about logical reasoning, I thot it must be an essentially different thing than probilistic reasoning "because": I. difference between. pc = .99 & pc = 1 is that pc = 1 can be used ∞ times w. pc = 1, so no attenuation to infinite no. of times use. pc = .99 (or pc = 1-ε) does not directly extrapolate to ∞ times: $(1-\epsilon)^{\infty} = 0$ for all values of ε > 0 & ε > 0 for all probilistic reasoning (otherwise all possys are not covered & a system is not universal)

Breakthru

→ How., we can get extrapolation to ∞ uses from pc = (1-ε), by a trial recursive definition. pc's of (1-ε) (small ε) will be a favor for extrapoln. to ∞ uses of (1-ε).

20 We are motivated to try the (recursive (∞ use)) rule empirically. If it works well, it will be pc of corpus & we retain it.

If .13 ff is valid, we may not have to regard "logical reasoning" as a special kind of reasoning. It can be a (perhaps) normal branch of probilistic reasoning. Even in non "logical" reasoning in which pc's are close to 0 & 1. Even in non "logical" reasoning we would want to try the recursive rule(s), setting, if pc's are close to 0 or 1, we would ...

HA! The foregoing may be able to give a good way of looking at Math as an "Empirical Science" — which it would seem to be anyway, because of Gödel's thms.

0 Some impt. pts to emphasize in talks:

1) "Comp Cmplx theory" is barking up wrong tree; we are interested in small SSz; P = NP or P ≠ NP is usually (but not always) irrelevant. (& "Only slot long").

2) ALP does well w. small SSz partly because it uses lots of apri info. — It has a good, general way to use apri info.

5 3) We need diversity (|| models) for good induction.

36 4) We need partly using models to give high pc's to simple (ideas/cones)

5) We need the completeness of a "bulldozer" — Turing Complete: Other forms of "universality" are inadequate.

6) Training Seqs are needed to get info into system.

3-19-05

A MAJOR PROBLEM : (.17)

5TM

.10 : Re: Sub-function Trees: The CFG mite be able to express some of them; then just a search for sub-function trees i.e. re-parse parsing Corpus wrt. them would be very useful. CFG does give recursion of a kind, in addition; But we could get this from sub-funct trees also, if we used a suitable language. The CFG recursion could be entirely different from t. direct (Lisp-say) recursion!

So what remains to be done for PhI? : Perhaps hardest part is TSQ:

A prob'y v.g. way to deal with this: If TM can't find soln. w acceptable cc., then "As a Hint", add a Lisp Macro that should make t. such easier. The "Macro" would be ~~[illegible]~~ a "low cost" defn. inserted into TM's vocab. But it would have to turn to user

10 There is a simple BU method of doing QA problems: we use a simple grammar: $Q_1 \to A_1$; $Q_2 \to A_2$ ... $Q_n \to A_n$. If any Q's are identical, then we have seq. over their A's — Initial, "Bad but acceptable" grammar to Hill-climb on.

So start out w. simple PhI TSQ: compare Lsrch using PPM, CFG deriv, then augmentations of them, then do BU srch using GP. — Do them all for same initial corpus, to compare them : Compare PPM & CFG deriv wrt. [pc that they give t. corpus]

.17 [SN] In PhI, I had been using fast Meta TSQ, t. successful seq. of O^j's thus far! i.e. $[O^j]_{j=1}^{n}$. Is this a (v.g./Best) way?

Alternatively, Use t. set of O^j's that solved t. problems thru $[QA]_n$. (Even if there is only 1!) : Then get a ~~pc~~ 's grammar thru that set for expr pdn.

Is this equiv. to t. s-lang that is obtained by $P(S \mid O_n^j)$?

Where O_n^j is t. best O^j that does all of t. problems. — This is t. pc equiv. of t. no. of bits needed to specify S, given O_n^j : S is a cand pgm for t. problem corpus $[Q_j A_j]_{j=1}^{n+1}$.

This may be a better way to look at ~~[illegible]~~ this problem than before.

It is a Critical Q in PhI

1) Q of Meta PhI's TSQ
2) (Essential (needs) Adequate features) for strong A.I.

.20 [One direction: for Single O^j corpus: Just use "Best" last O^j. Then use PPM (or 'sub-tree feature Augmented PPM) to get PD for cands. Or use CFGs also perhaps augmented for CSG perhaps w. "sub-trees" → (44.00)]

.20 [SN] : What are a set of (necy essential features / sufficient adequate features) of a STRONG A.I. system?

(3 (.35, .6) 42.35-.40 (4 ss are about t. same)

Tony Complete lang | diversity thru (1) models | Adequate TSQ. | Another is Better (adequate, meta) Good learning System.

From these Essential features, it may be possl. to use a great diversity of alternative ways to ~~get~~ Strong A.I.

3.19.05

BIAS Assoc w. Srch ALGM is (O.K./nacy) : (.27)

.0 (see 43.28): This will begin to work only when t O^j is quite large.

.01 Another possy is to use several "good" O^j's & construct a common Grammar or use them w. Augmented PPM, as elements of an "unordered corpus" but give each O^j its proper wt. (e.g. ≡ t. pc it gives its corpus). This will mean that t. individual O^j's don't have to be so large, but their total length should be large.

I guess .01 seems more attractive than using all previous "successful" O^j's as corpus — somehow t. set of O^j's that solve thru "n+1" seems more "natural" a corpus.

An apparent problem w. .01 is that, it may (esp w. many ‖ O^j's) not give enough ssg of useful sub trees.

.1 A related idea is ≈ OOPS: Problem sets given in sequence. TM solves each set: t. info of probs in each set are tightly coupled, but data betw. sets much less tightly coupled.

Also note: in INV probs, we are looking for a single algm that can solve all inv. probs ← this is partly ok. within a const factor of optimum ‖ we do want direct coupling betw different problems, but in OOPS, I suspect t coupling betw. diffrnt prob-sets is too weak — & is only of one kind!

.13 A other (perhaps adequate) view: T. corpus is t. set of pairs → 46.20

$\left[[Q_i A_i]_{i=1}^{K} O^j_K \right]_{K=1}^{n}$ This really is t. history. Express that corpus in more compact form; there is much redundancy; $[Q_i, A_i, O^j_i]_{i=1}^{n}$ does express all of t. info. — T. Q is: does $[Q_i, A_i]_{i=1}^{n}\ O^j_n$ really have better info? — i.e. are all O^j_r $(r<n)$ uniformly worse than O^j_n?

.2 In discussing incompatibility, diversity, etc: refer to total wt. eq. (3) factors!

.21 $\sum_i 2^{-|M_i|} s_i$ is as BIG as poss. (Not exactly right: we should also include

.22 prediction ... but this actually gets us into trouble: it enables us to bias t. srch! → .27 (o.k.: No diffy!) (34)

So using .21 as a criterion would seem to get around t. search bias diffy, but it means no "One shot try" (2, 3 shot try is o.k., hvr).

Maybe try ≈ .21, but sum over all "next symbol" possys.

.27 (.22) AH HA! Beside thru! Any Bias int. search is Legit.! The bias is int. counter of to Apripd & t. srch algm: T. combination of t. 2 yields t. final "effective apripd" The usr can bias "effective apripd" by manipulating t./ theoretical apripd a/o t. srch algm.

.3 These 2 sources of bias are unavoidable & really desirable, since they give usr. ways to insert his apri info [Hvr, use of a universal UMC for apripd, does constrain apripd a lot. The modification of t. aprid due to search methods, is not so clear! — As example of extreme effect of search algm. — Consider an srch algm that rejects all codes that give ≤ .9 pc for the "1" continuation! 1 Shot Try.

.33

.34 (.22) If we include Preds, we want to include codes for t. next symbols (M t. wt. & all poss.). In t. case of OSL, this would → 47.30 mean that we might add one or more M_i's that had definitions for t. ngms involved in possl. continuns.

3·22·95 STM

Micro Cosmic God — Ted Sturgeon

Astounding SF Ap 41: pp 46 – 68

46 Jimmy Kidder.

Banker: Conant

48 factory made food
AL alloy
Life pump?
Synth Chlorophy:
Propeller at Kidd
U238 power

49 President Conant at Bank.

50₁ formalize pc & reduce it to such tour terms that he know which expts to not try

50₂ Oxalic Acid → Clotting Agent for Blood!
Elixir Made Sleep Unnec.

51₁ speeding up life/metabolism x20
Generation in 8 days: life span 15 days
6 hrs gestation, eggs hatch in 3 hrs.
Sexual maturity in 4 days.
Female → 4 eggs.
Males died 2, 3 hrs after Mating.
3" long. Much NH₃ in atm. for Them
So Couldn't live in Earth Atmosphere

51₂ 4 races. insure diversity?

Neoterics

[52₁ How Kidder Got them to solve initial problems.
Wars betw. sections of Tank.

52₂ Making Strong Alloy: —
"Rule of Fear".

53: Kidder (looks like picture of Calvin Moores)

54, T. Teletype & t. "sort of social roles" chosen by Neoterics

56₁,₂ Power source, an xmtr

57 How it works! ☹

Another approach: Have Pool of problems accessable by population: Each problem has its own stated Prize for soln. Harder probs, big prizes. Prizes are "Money". Money can be used to buy "reproduction services". Individuals can cooperate to solve problems & share prizes.

Money is "Carrot": T. "Stick" is death of individuals & genetic kin if indiv. does "Bad" things.
"Bad" ≈ Not following orders.

Also "Good" works rewarded by well being in life.

The Pgs system has Bog of all computer pgms: It does what you tell it to do, not nec'ly what you want it to do! → 46.00

¶52₁,₂: Subject N's to bad weather of various extremes, to get them to invent adaptive devices, to survive.
So Basically, Each Threat is to entire race & t. individuals can cooperate to solve it.
The "Crusher" & t. Al alloy Soln. is a similar problem.

52₂: T. "rule of fear"!

→ 53: The teletype w. commands: They had to develop what was asked for — full time: no other projects by anyone — Or They would be penalized much (½ population killed).

3 stages of Education of N's! (Neoterics)

1) Organic Evoln: Diversification by Genetics, followed by Selection by Catastrophe.
or: Slow catastrophe w. Genetic "teaching".

2) Impending Catastrophe: Culture solves problem.

3) Culture can Read instructions (has "language")
If instructions are not followed — Catastrophe follows.

20 1) has to get Culture to state where 2) is possl.
2 1) or 2) has to furnish N's w. language.
Lang. can either grow up in culture or (better) be externally imposed — Contrary much into — Easier for "God" to understand & give orders in it.

Getting 1) Org Evoln to get set of individuals to cooperate enuf to form a culture & (w. perhaps a lit. help) develop a complex adequately language.

Getting some cooperation (like symbiotic species; interdependant symbiotic species) (symbiosis) may not be so hard; but getting a group of individuals of same species to cooperate on common projects, seems diff!

A first step toward cooperation: "Mating":
by having "Mating" done by >> 2 individuals, a stronger step toward "community" (& "culture") is taken.

Family raises children: Family is extended when kids "mate".
1) I want individuals to (work on/solve) problems
2) " " 2 or more individuals to be able to cooperate in solving problems.

S(U) Note that this is an "ALife" problem, not "GA" — I'm not so sure! They did have viruses, but it looked like GA rather than ALife — e.g. There were not a lot of animals/plants interacting w. t. Alsos.

STM "Micro Cosmic God" (cont) – Sturgeon

(45.40R): As is, the concl of "indivs" as pgms. How do they "interact"? How do they reproduce? It may be that after the system has set of suitably communicative, bright individuals, it may be that little advancement is thru "org evoln" most advancement thru development of technology in the system of individuals.

Tho, we could have org evoln continues — it's easy to continue to "improve" the pgms that represent "individuals".

Indivs. can communicate w. eachother by "Messages" in some common lang. This is same as lang. that "God" uses to give them (orders/assignments).

Note that we may want individuals to be similar enuf so that ① communication can occur ② that observations by 1 individual can be useful to another indvl.

— Could individuals be like Object oriented pgms. "Objects"? — They would be a subclass of "objects".

(45.31) Re: Money: Can also be used to buy things that might generate more money. — unclear how to do this, hvr. T. advantage of Money is that individuals can have many different interests (→ "diversity") & be rewarded for them.

44.13 SN ≈ Problem Pool idea: In OOPS! Consider Tow of Hanoi; we are given a seq. of problems w. 1, 2, 3, 4 ... discs on the pikes to be inverted. J. considers the sequence of final pgms that solved the problem {program$_i$}$_{i=1}^{n}$, pgm$_i$ solves i disc problem.

So his corpus is {pgm$_i$}$_{i=1}^{n}$. The only ways he notices is freq. of use of each inst.

Two badnesses ① This is too weak a way to help much (2d Lsrch use some "context") ② Lsrch doesn't work that way anyhow. The corpus that guides adaptivity of Lsrch is not {pgm$_i$}$_{i=1}^{n}$, but the pgms that found the pgm$_i$'s.

J used the same pgm to find all of the pgm$_i$'s — i.e. ≈ "Lsrch"

5 TM

10: "Modeling Statistical Data Compression" Steven de Rooij Sep 2003
≡ D\PS\ PPM Augmented Thesis.PS 3/24/05

Seems Readable! He seems to get maybe 10% more compression than PPM. (Much more compression for Lisp code).

1.934 = his / 2.160 = PPMD* = .895 so ~10% better) books Calgary corpus

1.571 his / 1.972 PPMD* = .7966 ~ 20% better) Lisp (prog L)

He did much worse for some EXCEL files hvr: — That was only case he did worse on.

His Blurring Prior (pp 60-61): I could do this much better w. probly simpler results, by using a β distribn. & smoothing it w. a kernel that is a β distrbn of constant Var, but w. a mean that corresponds to point being smoothed.

I suspect result is almost trivial! We end up w. a β distrbn w/ same mean as before but Var increases (just as if we smoothed Gaussns. w. Gaussns i.e. $\sigma^2 = \sigma_1^2 + \sigma_2^2$.)

Also this "Blurring" stuff can probly be treated exactly in ALP.

Anyway, his discussn & Criticism of PPM is probly well worth reading: Put my mind in good state to improve PPM — Runs ~ 20 pp, but I don't have to read all of it!

He also discusses M. Li's DNA compress!

He used a binary alphabet — which would seem to eliminate "escape" problem — (No, it doesn't at all!).

Before reading this J very carefully, familiarize myself w. my "solns" to PPM problems.

D/PS/ MDL-Search.PS Ian Davidson → 3(25/24).05 → 3.25(24?)/05 ! Describes "Gibbs sampling" & other srch methods. ← This guy wrote me Q's about ALP. I'm not sure we understood his Q or my reply! He seems to be using ~ ALP ! with statistical SVMs over Models. (Stock Complexity?),

Perhaps send him an augmented Version of Lecture I!

0:44.33 Re: Search alg influence on ALP & a Flock of Specimens : we should at least use srch algm that → ALP as cc→∞. Hvr, we can still have an algm. that is Bad for T < Ackerman(10) seconds & → consi. method for larger T. cc, so it approaching → ALP as cc→∞. Hvr. srch Alg. is chosen by Usr, so he will avoid such methods that seem to change as cc→∞.

3.25.05
5TM

48

(17)
(97)

20: : [redacted] More "Remarks on Ph II". We can view Ph II in a narrow way: i.e. T. general improvement of Ph I's search — or, more generally, t. improvement of all/any PSM's. Usually (?) t. more General problem is easier: also Corpus has more "diversity" —

13 so cross-fertilizn is possi. btwn PSM improvements → (.07)

[SN] Another thing I hadn't considered much in open. problems: That TM could watch G(·) operate — so it knows how G (t. optzn criterion) works. Optzn thy is (possi/impossi.) in GA; it is assumed to be impossi.

07 (.03) → Having a large Corpus of PSM's certainly makes it easier to construct a useful S-Grammar for them.)) impt!

0 Tho t. method of doing Ph II in IDSIA report is "Not Bad", it's certainly only t single way to define "Optzn for TM. There should be a more general way: - but it eludes me!
So it would be good to have Ph II solving a great diversity of problem types

13 (for a while) it works on improving Ph I's srch. → (.30)

→ G, Z, Schmidhu: 2004 "Adaptive online allocation to srch algms."
3 authors
D/PS/-3/25/05: "My title "Adaptive CC allocation-Juergen-2004.
I have 2 versions: Power Pt presentation in PDF 38 slides
PS. in 12 pp - actual paper.
This ↑ is very close to my Ph II problem. Their method could be much improved, [redacted]
(i.e. t. set of PSM's is finite & fixed for each run) but prob'ly ok. a useful ideas, perhaps.
20 They don't use exactly like Gorc, etc, but --- O.K.
It can be used "to start off" on a serious Phase II.
(Pt 3 of Juergen's "DOTA" 29 Sept 2004 rsch proposal to FNSNF has / suggestions for improvement.
↑ In Juergen's folder, ideas for

30: (.13) → Ph II: ← 3.28.05 On [redacted] improvement of PhI's Metasrch: If t. lang. we use to generate trial PSM's for PI Metasrch is universal, Then its most Meta aspects don't have to be v.g. If they are poor, it will just take a longer time to get a good srch algm for PhI, but we will get it eventually. So we will get a "Bootstrap" effect.

T. usual way Ph II works: We have a universal S.G. ran using to corpus, a set of good PSMS. One impt Goal is to get a good PSM for Ph I's srch problem.
We have (One) (or 2) default/initial PSM for this, & we want a BU srch to improve it. (Already,
↑ say AvgPPM or Avg PSGdiry
Avg PPM & Avg PSGdiry are quite close to each other)
So we use t/ PSM generating Grammar (universal) to create cands for PhI srch. We use — (50.00)

3.26.05
STM

.00 : On ALP & uncomputability, etc.

.01 1) T. uses definition of "primitive recursive functions" is that it only uses composition & "primitive recursion" — i.e. $h(\vec{x},0) = f(\vec{x})$ [known], $h(\vec{x},y+1) = g(\vec{x},y,h(\vec{x},y))$ [known]

.02

.03 It does not use "μ recursion": i.e. $h(\vec{x}) = \mu_y f(\vec{x},y)$ [known] = smallest value of $y \ni f(\vec{x},y)=0$

"legal" in prim rec. is also projection! $T_i^n(x_1 \cdots x_n) = x_i$ $(1 \le i \le n)$

So it would seem that the Ackermann funct would not use μ recursion & be prim. rec.
Since Ack funct. is a triple recursion, it's not immediately clear how we got it using .02.
Perhaps .03 is used!

10 It would seem that any sequence of functions trying to approx ALP, would each be computable & very probably prim. recursive. As soon as we found a funct that had no value (or seemed to have no value) for any args, we would drop that function.

14 So if we tried to prim. rec. functs in their recursively enumerable order, we should probably approach ALP.
No! — Because there are total recursive functs that are not prim. recursive. (rare, however — e.g. Ackermann (.01)) . So probably .14 is "not bad".
If we tried all the prim r functs to approximate ALP! eventually the functions would get so long that we wouldn't have to consider them; so in theory ALP using prim. r functs is well defined & "computable" .... but in practice, no more computable than if

20 we used Partial rec. functs !

On thing that distinguishes ALP from usual prim. rec. srch: ALP will try args & functs that will occasionally be beyond CB., so it will not use those functs on that round of Lsrch.
Hvr. in using prim. rec. functs, people usually consider only functs that converge & they devote enuf time to them to enable them to converge. — They don't delete functs because they take too long.

If one lists (recursively) all prim rec functs, I don't think the ones w. hyp ever take much time? .... Maybe not: a prim rec funct of n can take ~ n operations to compute (e.g. x!) actually > x operations as x! may be ~ x ln x. x^{x^x} can take a lot of ops for large x.

30 Basically ALP Does μ recursion & prim from Rec. doesn't.
Since ALP only approximates μ recursion — Could approximate prim rec. simulate it? —
Also would it simulate it w. much of pc? (Ro Koorobeck / Tho info ms. approximate μ recursion, should be some way to do in prim rec. simulating μ recursion.
I really don't understand this!

48 to space

: to evaluate these cands. Evaln. can be done in 2 possible ways:

1) Direct: we try it out on PhI's corpus - (or a Example of that corpus)

2) We use a faster algm for approx. evaln. (This is from IDSia Report): .

This has developed an algm that can look at any problem, PSM pair & get a Good/Bad pair.

(This is faster than (1) (.08) operating on the full PhI corpus, but much less accurate ☹.

We have to Balance this inaccuracy w. (2) operating on samples of various sizes.

→ But 1) is perhaps easiest, & we can hill climb on it.

A big Plus for 2) is that to evaln A algm. works for a great Diversity of PSM's & problem types.

This Diversity is imptt in suggesting novel solns. to problems,

It means that PSM source by Grammar gives a good diverse sort of Cands for all kinds of probs (not just PhI's/ more probs).

Also by Mixing in probs from various problem types, it has a larger ssz than if we used only from only 1 type of problem to optimize PSM's for that problem. (e.g. the PhI problem).

CONTEXT in agnostic PSM: Consider "External context" (like is problem in Math, Physics, Chem, SM, Sociology ...) we can assign a multivalued symbol to this info. How it fits in w. other contexts in PSM is unclear. I did work out a way to work in sub-tree contexts. Perhaps it can be applied to it as well.

"Regular Expressions" are easy to recognize (I think). Are Reg expressions simply a Finite state Grammar? Can they help discover Common sub-trees?

In expressions like [+, A, B.] we can recognize that A is a sym. because the string A starts w (and ends w) & the parens are always "Balanced". Balancing is easy to detect. We open w. "(" and x→x+1. every (increases x by 1, every) ↓ x by 1. If X ever gets < 0, we have a syntax error. If X = 0 before end of string A, then A is not a proper variable. So (goes inc X,) goes dec X: when X = 0, if we are at end of string A then A is a proper variable, otherwise, it isn't.

In .26 ff, if we use Polish or (Rpolish) we should be able to do same thing.

In my "One, Two tries" pgm, I have large nodes and info about nodes. Could I use that info to detect common sub-trees? (or a particular sub-tree "context"?).

00: : [SN] On S.CFG discy: Stolcke uses 2 operations Merge, Chunk to hillclimb to peak.

It should also be possl. to do inverse operations: ① invent new NT's — which is chunking!

So Merge & Chunk may be inverses! Maybe no need for new operations?

Any Merge can be undone by a suitable Chunking.

Any Chunking, hvr, need not be inverted by a merge!

Say a certain chunk, find 2 rhs's of diff NT's & chunk them.

.08 e.g.

$N_1 \to N_3$
$\to abc$

$N2 \to abc$
$\to N_4$

10 .11 Chunk: $N_5 \to abc$; $N_1 \to N_3$, N_5; $N_2 \to N_4$, N_5

Merge N_5, abc } Seems difficult!

.13 Hvr $S \to aabb$
$\to ab$

$N_5 \to abc$

Chunk: $S \to aN_1b$
$\to N_1$
$N_1 \to ab.$

15 Inverts by Merging N_1, S. No!

No: have reason to do this, hvr!

So .08 – .11 does invert by merging N_5.

Chunk $N_5 \to abc$
$N_1 \to N_3$
$\to N_5$
20 $N_2 \to N_4$
N_5

Inverting Chunking means replacing N_1, say, by its definition in all cases: This means that N_1 is not yet in recursive form.

Inverse of Chunking is elimination of a Non-rec NT by substitution in all cases.
((look into subst not all cases!))

Inverse Merge, would be breaking up a NT into 2 NT's; Can be done if 1 NT is in recursive role in which the "divided NT" occurred. This may be complicated! — since we then have to double up every

But it must sometimes, because some Merges ↓ pc!
Have to see why this would ever ↑ pc,

→ If a NT has many (n) RHS's, there are many ways to divide it. $(2^n - 2) \div 2 = 2^{n-1} - 1$

30 So "Divide" is inverse of "Merge"
"Expand" is inverse of "Chunk" (= "Define")

Chunking can sometimes be a kind of divide. — But creates new n.t's.

But all chunking can't be done by dividing! e.g.

$S \to AabcB$
$\to RS$
$R \to abcA$
$\to abcR$

} so $T \to abc$ would be a good Chunk cand. — But no way to do this by Dividing.

Merge & Expand ↓ no. of NTS
Chunk & Divide ↑ no. of NT's

Chunk sometimes = Divide but not always
Merge " " Expand " " " "

58.06 → Space

4.2.05
5TOY

∞ : [SN] Gen Notes: Paper D/PS/ VCDim – Recent – Compr Compl of Ling : 4/2/05 ← Submitted to arXiv · 17 Fobos hat at Royal Holloway! ©
Says VC Dim ≈ K comp. for · "Concepts": Also interesting (perhaps) results on EC of finding ~~models~~ induction Models, He seems confused about Hutters showing ~~[illegible]~~ Robustness of ALP conv. Thm — being related to Ryabko's results on C.C as funct of problem complexity. He may be too closely ~~tie~~ tied to poly v.s. Non poly ~~[illegible]~~ categorizn.

[SN] URL: [Patterns] in Mathematica" URL http:// documents.wolfram.com/mathematica/book/section-2.3.1
~~[illegible]~~ 14 pp : This notation for "Patterns" seems very useful: Probly I'd want [2.3.1 thru 2.3.14] TM to learn these (definitions/notations)

16 I'm not sure that t. notation is "consistant" — (or that it means what one would expect it to mean)
— But that these kinds of "Patterns" are important, is certain!
Could we devise a tsq so TM would discover patterns because they were Useful!, rather than because they are a soln to a simple QA induction problem on "ting datas". ?
Anyway, for TM to invent "structural" idns/equalities : would be a good way for it to learn how to do algebra, & much of Math.
Can all of this stuff be done easily in Lisp?

$x = \frac{1}{1+x}$

[[Do try to Genz. Mathematica "Patterns" Devise lang. to fit them so that they ~~could~~ can easily be devd Viz. ~~these~~ "Primatives" in t. lang.]] ← Primatives

$x^2+x-1=0$

10 [SN] D\PS\ Grammar discovery 4/3/05 & SCFGs may be improper: Pg 300
Improper means Σ all probs < 1, due to many infinite strings that never terminate.

$\frac{-1\pm\sqrt{1+4}}{2}$

22 They get eq. $S_{n+1} = q + pS_n^2$ which leads to Σ probs being < 1 if $p > \frac{1}{2}$
How I could get eq ↗

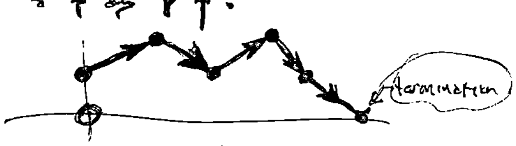
$x \approx .618$

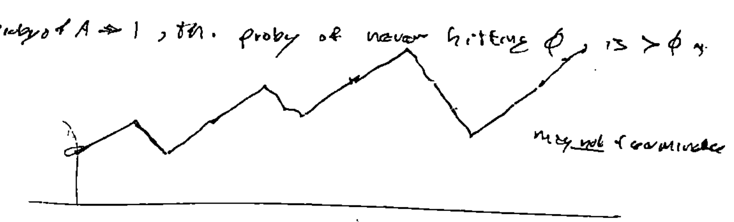
T Grammar is $A \xrightarrow{p} AA$, $\xrightarrow{q = (1-p)} a$. If $p > q$ then we will tend to have intermediate strings of terms & N.terms w. ~~[illegible]~~ a fraction $p-q$ A's in them. i.e. non terminating. so if $p > q$ we will have a certain frac'n that don't ~~terminate~~ terminate & this fraction ↑ as p ↑.

$x = 1-x^2$
$x = \sqrt{1-x}$
Both wrong

An exact Random walk model: Up 1 step = $A \to AA$; Down 1 step = $A \to 1$.
we start at +1. If proby of $A \to AA$ > proby of $A \to 1$, th. proby of never hitting φ, is > φ & ↑ as P ↑.

0





→ Still, I'd like to know how they Got. t. eq. (22) $S_{n+1} = q + pS_n^2$.
It is clear that t. basic thing is true: s. CFG's can be "incons." — But they say that Grammars that try to model real data will be consis. (These non terminating strings are not in t. corpus!)
Q: Say we have a corpus Generated by a incons. Grammar: Might it be possl. that

4.3.05
5 TM

SM.30

γ Gamma | λ lambda.

.0 The semimeasure induced by the inconsi' grammar (for that corpus) would ⓐ have a shorter desc length than the consi grammar assigned to the corpus? The idea is that the inconsi grammar should give e.g. relative probys of the class in the corpus.

.03 So what we want to max is ⟨(pc of inconsi' grammar) × (pc of corpus wrt grammar) × (normzn constant of grammar wrt that corpus.)⟩.

.00 Would .03 not give a result different than maxing ⟨(pc of consi grammar) × (pc of corpus wrt that Grammar)⟩ — i.e. would these 2 options extrapolate differently?

Maybe small difference — only detectable w. very large corpora!??

2 1

0 **SN** Admin Note ⓒ: When in College I had to move rapidly thru a lot of science — even when I didn't understand part of it. (I would remember what I didn't understand, however) Outside of college I would often stop at a pt. & work thru it until I understood it: This could take a wk. (or more!).

What I need to do now is: Move Thru TM construction. Parts I don't understand: write out clearly what the problem is, & move on (unless it is a critical problem ~~so~~ so can't "Move on".

— So keep updated list of unsolved or "not completely satisfactory" solved problems: & notes to work on them.

Summarize state of present development

Tom •
Murray •
Doe ?
Judy ?
Alex •
Ray
Grace

5 **SN** **SM.** In predicting a stock by its correlations w. individual stocks in a pool of 1000 other stocks, a correlation has to give an ↑ in predn by a factor of 1000 before one can justify specifying it as a predictor! — But using all cross predns in // has a cost of zero (its all a pri).

This last isn't so bad. Most drivers will have correln of ≈ 0 anyway

Using = wt. for all 1000 drivers would give predn. of zero most of time? (wts would be hy for drivers that gave good (low $σ^2$) predns. — so maybe not so close to zero predn for pool?) This is kinda — correl. w. = wts for all 12 predictors

Also, my idea of certain stocks getting "News" earlier than others. The stocks slow to respond are those held by Big funds! (That can't buy/sell rapidly in a moving mkt.) They respond late and/or over a longish period of time (slowly)

STM

00 : HA! By chance I ran across 98TM 110 ;--- 132

ibcb 110.00 is on t. Mixed Corpus Prm which I seem to have just discovered!

" 132.01ff is on SGA (Super GA). The Idea was to model a GA Score by a Stoch Grammar So t. pc of t. Grammar was a funct of t. Score (or a funct of t. pc of Grammar was = t. Score), = Fitness Funct.

In present work on QA induction, t. Score is a PS!

I once tried to fit a SCFG to a score, but I really didn't know what I was doing!

One of t. (apparently) neat tricks: If t. Score was not a pc, I would make t. case count assoc w. each cand equal to be = t. fitness funct. Of course t. fitness funct had to be modified so that they looked like pc's — but in a common GA, making reproduction pc = "fitness" is common — tho I suspect that t. people that use it don't know how "flakey" it is! (i.e. that any funct of a fitness funct is an "equiv" fitness funct) ≡ Legit

for the $\frac{\sigma}{Max - M}$ criterion, it is well to have a "linearized" Score (≡ fitness funct).

4 In ibcb (98TM 129.33) an interesting Note: ← There may be a copy of this page! look at SOM! Given a very large Corpus: One doesn't want to code all of it: We have a certain cc available & we want to use it best to beat on problems of most value.

I did get t. "Encyc problem soln" — (Tho I don't remember just what was t. problem & soln.!). I think the idea was that TM has this Encyclopedia which is "free". To answer Q's it has to point to parts of it: these pointers have pc's (& bc's)

0 TM indexes t. Encyc. (a Meta task) so that less cc is needed to do each task.

21 Another Remark on Older notes: 3TM 425.00 — .06 Idea of using Maple/fact, for d-induction problems for Ph. I. — Objection was that these inductions would not readily extrapolate to s-induction — well maybe they would! T. way it works for s-induction we get functions that output t. "most likely" A. This is a "cluster center". The Random R inputs to 3 inputs modify t. pgm to give more distant members of t. cluster. It would seem that t. codes used for D-induction, would be good for getting "cluster centers". Testing trial "R" values shouldn't take very long because we are mainly interested in hy pc outputs!

0 3TM 423.32 — 424.29 on "Soma" (Simmons Machine): Using BU srch! We retain best 100 solns thus far. Say we use Rea (the augmented PPM) to construct a pd. on poss l. trials. When a new problem comes in: we try all 100 "Best" on that problem. If solns aren't adequate, we use t. grammar to generate trials until we get a satisfactory soln. We then pick 100 best cands thus far as t. basis for t. next problem. I think this tends to keep hy "diversity".

I've forgotten what t. "2 kinds of PDS" were: Maybe one was a pd. that one starts with." The other was "a pd." modified by experience, that is used to guide Lsrch (or even Bsrch).

As I see it now, there are several (>2) pd's of interest:

4.9.05

5TM

AZ-141 : 002

5.5

10 : 5440 : 1) T. true aprip w. zero corpus, obtained from reference UMC, using CBOOA. True ALP

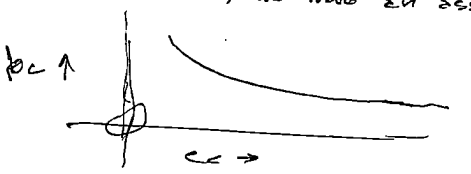
2) T. initial assignment of cpt's to t. symbols of t. lang. used for (1)

3) After TM has solved n problems, we have a corpus, $\left[Q_i A_i\right]_{i=1}^{m}, O_j^i]_{j=r}^{n}$. ?

This corpus plus t. "true ALP" of (1) induces a new pd. — which is t. "true apriori" for t. next problem.

4) Since we do not have time for good approx of 3), we use PPM, augmented PPM, S-CFG & augmented S-CFG.

Some time ago I did work on: If we have a "not so good" method of induction w. a certain cc, we can always repeatedly (w. Boosting) to get better PC values, but for higher cc. [Assume for t. whole Post Pgm is true; If means that for each prob evaluator, we have an assoc set of expected pd's for corpus & each w. its own cc.



For a distinct prob evaluator we'd have a distinct curve: So, unless on curve covered t. other completely there would be a crossover pt, cc_0, where for $cc > cc_0$ we want to switch prob evaluators.

For a simple L-searcher for prob codes ... certainly larger cc/almost always gets better codes.

But consider PPM: w. larger cc we could use longer contexts, but this doesn't help much. (Using larger corpus helps, hvr.). By using more complex contexts, (subtrees) — we can use larger amts of time to search over larger subtrees.

Similarly, in S-CFG's we can do longer searches for better grammars, we can retain more ll parses. — Use contexts consistently. Contexts can be of various kinds — per haps subtrees?

SN At many pts. in past I felt I had (as I do now) a v.g. picture of t. path of TM development. One such pt. is 99TM503 "PLAN"

STM

:0: • How to use (Aup) ZHH for predn/compressn.

To compute the pc of the next symbol of a seq:

Compute the pc. of active seq. containing continn. S_i [$i=1,\dots$ No. of symbols in alphabet] using simple (Dir's rule, Laws rule, Bayes).

Call this p_i^1 ("1" means 1gram dictn. of length 1)

Compute $p_i^{k_i}$: This is pc of seq. w. S_i for next symbol, declining a k symbol suffix.

$\sum_{k_i} p_i^{k_i}$ sums pc's of all possl parses of the corpus, using that definition.

k_i will have a max value — This is when the suffix assoc w/ (k_i+1) has not occurd before in the corpus

∴ pc of S_i at that point is $\sum_{k_i=1}^{k_i^{max}} p_i^{k_i}$.

This pc should be already normalized. If not then I have missed some rede. ←?

The actual pc of S_i at that pt. will be $\sum_{k_i=1}^{k_i^{max}} p_i^{k_i} \Big/ \sum_i \left(\sum_{k_i=1}^{k_i^{max}} p_i^{k_i}\right)$

Actually the normalizing denom. factor in .0g is not 1.0 but if we have included all codes,

→ we can use, instead of that denominator, the pc assigned to the corpus w. S_i deleted ←(?)

.11 may be false. The coding for the seq. containing the aux symbol S_i may be quite different from the codes for the seq. not containing just S_i.

At any rate, .0g is probly correct. The need to compute $p_i^{k_i^{max}}$ for all values of i seems like a drag! There may be a way to avoid it! Would using .11 be a good enough approxn? We could try to compress, using it.

We could also study relations of .0g, to .11 : i.e. how does the denom of .0g compare w. the pc of the corpus w the last S_i omitted

Any way the denom of .0g increases cc of the system by a factor of "r" (= Size of alphabet. = the radix of the seq.)

One way to decide on this is to use Binary alphabet — hvr. more symbols to compute: It's apparent ↓ in cc is $\frac{2r}{2\log_2 r}$ i.e. use cc is multiplied by factor $\frac{2}{r}$ for each symbol, but [prediction] uses $\log_2 n$ as many symbols. For an alphabet of 128 symbols (ascii), it's still a factor of $7 \times 2 = 14$.

There may be tricks to reduce this. Perhaps get rid of factor of 2?

In TM, I may be dealing w. smaller alphabets, but still $\frac{2r}{\log_2 r} > 1$ so I should use Binary.

Use other log basis $b \log_b r$ min wrt b.

$\log_b r = \frac{\ln r}{\ln b}$: $\frac{b}{\ln b} \cdot \ln r$ is min when b is min. $\frac{\ln b}{b}$ is min

$\frac{1}{b} \cdot \frac{1}{b} - \frac{\ln b}{b^2} = 0$

$\ln b = 1$

$2/\ln 2 = 2.88$

$1.5/\ln 1.5 = 3.699$

$3/\ln 3 = 2.7307$

$4/\ln 4 = 2.88$

So its best at $e/\ln e = 2.71828$

So $b = 2$ is best. I can't see how to do $b < 2$.

Clearly b is best for $b = e$, but only by small factor $\frac{2.718}{2.88}$; Using radix 3 gives $\frac{2.7307}{2.88}$

So using radix 3 is slightly better than 2. Using radix 4 is same as radix 2

Hmm. There is more : using binary may give cheaper operations in general!

N.B. In considering various constant sizes: Only consider multiples of 8 bits (i.e. Byte Boundaries)

It's unlikely that working by breaking up bytes would be useful here ... The one could try it out!

57.00 spec

5 RTM

10: (spec 56.40): The foregoing discussions are directly applicable to Sub-tree Contexts.
→ We do have to find way(s) to describe sub-trees for defining them.
The Mathematical Expression for getting pc's & for summing over all parses are identical to those for ordinary String Contexts.

0 [SN] The Horizon Problem for RTM: My argt v.s. having Horizon (≡ H) = ∞ was that If one had a problem to work on & some (S.I. self-improvement) to do, TM should always do S.I. first. If one had time T to do a bunch of probs & S.I., One would do all S.I. first, Then the problems. If T → ∞, one never works on problems! So H = ∞ is not good!
Another approach, hvr. TM has many problems, but (Prob i) has to be finished by T_i. If there are ∞ problems, TM would spend almost all time before probs, in S.I.
This would not solve Probs very well, but in the "long run" it would give good solns for probs i (for $i \ggg 0$ — i.e. $i \to \infty$). This would continue indefinitely, TM would always spend as little time on problems as possl. (so as to not get Gore = ∞ for any of the problems). We assume that mean gore is the final Goal.

40 [SN] Mark Johnson: S CFG models of Trees: DAPS\ 4.13.04
21 This seems close to what I want. He does use "Context" for prodn, but I think his
.22 "contexts" are too small. — But I must read this paper (20 pp!).
Also, I take it that they assume only one parse: {The corpus consists of pre-parsed SS} which is my corpus, else
The remarks on Context (.21–.22) may be the main point. They look at the corpus of trees. The freqs assoc. w. each choice/Type in the tree are used as the pc values for a S-CFG.
Hvr, I don't know if they use POS labld trees, & get freq of the POS transitions, — or just what they do!

30 I really don't remember just what I had in mind when I saw S-CFG disc. as a great breakthru! The Policy drvn of the functions was the corpus. Just how (why) CFGs would be good at devng, regys in that corpus, is unclear! The ss in the corpus are, indeed, ss of a CFG..... but I know what the CFG is! — its Lisp or AZ or whatever I'm using to desc. functions.
So CFGD would seem to be not relevant at all!

HA! A possl way to retrieve subtrees ~ to present one! The problem is similar to finding a certain pattern in a 2 dim picture. What we do is take the pattern of interest and convolute it w. the corpus. Hy peaks occur at pts ~ to the pattern. This need not → 59.00 (Spec 59.00)

4.12.05

STM

Kieffer | IT (IEEE) Trans July 2000 (1 paper)
May 2000 (2 papers)
<Jia, Yang> → Dec 2003 (2 dimensional compression).

0 : 51.40 : On Heuristics for ~~compress~~ rules for compression ≥ corpus. ← Heurs.

NB [Stolcke has 2 rules Merge & Chunk: Since neither work 100% of time, ~~I guess~~] Major Previous link in STM
the inverses of these rules must occasionally compress. Try to discover Conds under which rules & invs. work. ←

Similarly Kieffer (?) et al do "chunking only" rules. How they get 4 rules is not clear — it would seem that only 2 kinds would be possl.

They also have methods of extending f. system to 2 dim. ~~=~~ compresn. Look into this.

They ~~compress~~ do ~~the~~ their rules until corpus is "irreducible" — I don't know what they mean by that... (see?)... Actually, what they ~~say~~ should mean is ~~Irreduc~~ Incompressible by any of the listed ~~heuris~~ heuristics.

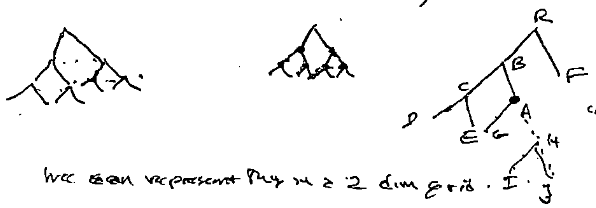
[So Go over Kieffer (?) & Stolcke papers (& perhaps others on ~~=~~ CFG discovery) & see what heurs they use for compressn. rules. This is perhaps one of most impt. ideas in "Hill climbing Compressn" (≡ Coding & Recoding).

On organization of "The Lecture Notes". As of present LN has:

1) Introdu. What is strong AI?
2) Listing of Needed tools. (Lsrch, PPM, PSG discy's, GP

} this discussn can certainly be improved!

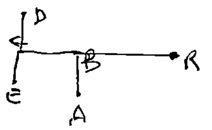
.00 (Space 57.40) : be exact, but it would find good cands. for matches! . Hvr. a 2 Dimensional convolution would be expensive, — but we can do it cheaply w. Fast Four. xfm. There may be some
02 analog of this for Trees .... say trees w. only 1 or 2 branches/node. → (96.00 Space) ⟶
.03



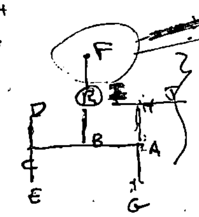
consider context of point A It is B R C D E

one can go past the Root.)

We can represent this in a 2 dim grid. I



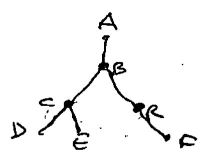
or



Here, the convention is that the children of each node (N) are on segments ⊥ the segment going to parent of the node (N). So for a large tree, the nodes would end up on top of one another occasionally. I don't know if it matters that R and I overlap.

0 any distance! — we are normally going up the tree only. We could make segments smaller as we went down the tree.

For each node in a (Function Cand.), we can make a binary tree going back from that node. e.g. from A, we have



So the problem is simply matching Trees of known Root characters, [redacted] Seems impossl. to do a 1 dim. ordering!! Could 2 dims do it?

What about Context Sensitive Grammars? Do they help?

A CFG for .03 might be:

R → BF
B → CA
C → DE

} No, That's not it!

CFG ↓
R → B(α, β)
α → C(D,E)
β → A

CSG
R → B(α, β)
β → A (so R → B(α, A))
β α A → B(C(D,E) A) (context.
So α → C in the context β ···· A has a certain ...

ideally we should be able to put "don't care" here = " * ".

This context is only of depth 1 or 2, hvr.

We can perhaps deepen it! R B α A → RB(C(D,E); A). Context of any depth can be represented this way!

Perhaps of interest, but the problem of finding / Matching contexts in the past corpus remains!

Q: to what extent can we use a "Regular Expression" (= Finite State lang)
to Express desired Contexts? We prob'ly need an individual Reg. Exp. for each type of branch; so 2^n expressions for branches of length n.

If one looks at the expressions for the functions (in Polish, say), we can ask for certain characters in certain positions. (Not exactly: A given "position" can have expressions in it of any length. Could we modify the notation to fix this?

AH! Given the tree structure  : Each node is given a fixed address relative to the root. ∴ This makes it easy to ask for all trees w. R at root, A at position 5, B at position 3, etc. We might use a Zato coding for this! Or perhaps Hash codes could do equiv. of Zato coding, ⟶ 60.30

39 ⟶ Could we use a "Content Addressable Memory"?

STM

Space

>0 : 59.40 : In CAM! How do we arrange to get one content at a time? Actually, we are only interested int. population of one of t. subaddressed of t. addressed locations.

It may be possl. to get CAM, using a register Memory in an "inverted way"! — Hmm, only 32 (or perhaps 64) address lines . . 64 may be cut off for Zobo today (?).

5

0 : BNF On context: Contexts whose previous corpus (+ external" memory).

Essentially, we want to predict next-symbol of O^T : If we use Phase I to do this : we are in $TM_4 = TM_2$ (≃ Phase II)!

This is not, hmm, optimal predn. : TM is just trying to do trials similar to previously successful trials. He doesn't yet understand what "Opter" means to any great extent.

Still, it's potentially Much better than simply using augmented PPM. I do have to design sub-tree a TSQ. so. it could usefully work on this problem. Presumably, there would be "Index symbols" telling TM Metrits working on a distant part of t. sub-corpus from t.

external

usual (Main line) TSQ. : i.e. t. Meta predn. problem is an "internal" problem.

ON "Lincos" : (Hans Freudenthal) This is a book on Language & Logic; It attempts to devise a language (w.o. pictures) that a could be sent to distant extraterrestrials --- that they could learn to understand. He expects t. extraterrs would get models of Humans & our cultures from t. TSQ he has devised. I would like to try his TSQ on my Lang. Algm. He starts out by teaching Arithmetic, algebra, Logic! It uses unary notation for nos., then binary — expects E.T's could quickly learn to translate betw. t. two!

0 ; 59.39 : While t. fuzzy. is true, I don't know if it's possl. to arrange the addressing of various Nodes, so That One can pick a node & it's children & parents are always spaced t. same distance from it.

I.e. if node address is N, then children are always at N+a, N+b; parent is at N+c } sounds impossl.

a, b, c are same for all nodes.

General Conclusion: That's SG discy. Doesn't seem very relevant to induction on structures of known structure. Extending PPM by substring Contexts is impt., hmm, it is of my cc — unless I can find better ways. Try to find my original work on "How to best do PPM" & how to extend Phase one. Then methods to substring Contexts.

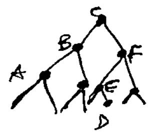
(10) Is impt. : It means that I can start improving th I's meta-induction w.o. all of Phase II.

But I do Need a Good TSQ!

4.15.05
STM

Rev: .09

: For Initial "Context" in PhI I could start by using ordinary Polish notation.
Ordinary PPM — adjusted for a corpus of separate shortish strings (functions)
Contrast compression of above by using contexts of direct ancestors only ;



So A has context BC ; D has context EFC.
We may add info on whether ancestor was "rt." or "left".
(or more exactly, whether child was rt. or left)
(Some ancestors only have 1 child).

.09 - Most Impt Ideas Since IDSIA! ① use of PPM ② 2 kinds of Srches : LSRCH & BU srch,
③ Possl. use of S-CFG discy for OZ problems — ④ Possl. use of Max-Mann to those Subset of Corps.
⑤ Better Understanding of "Incomputability" of ALP & why ALP is best for smart TMs.
⑥ Reason why I'm not doing Recent machines — that its harder problem than the one I'm working on, & it can be probly worked on by the system I'm designing.
— which will have to solve problems like "Cure Cancer" (≡ designing rsch Road Map path)
⑦ Calcns of how to Get proper PC's of symbols for regular PPM & for PPM w. sub-tree contexts (4TM 223.32 - 227.40)
⑧ List of Needed improvements in GA. — including (access to (Understanding of)) the fitness function Core) {

STM

0 : Matt Mahoney : PAQ6 V2 — file compressor: — the .CPP files readable by clicking on it or by "Fonfind"
See several files in D\PS\ 4.16.05: Also See Bookmarks PAQ6
This includes the .htm file which lists & codes itself:
There is a long .txt file in which I've written comments, but the format is changed so the diagrams are off: The diagrams in HTML are O.K., tho.

Anyway he uses many pred'n PEMs in ll & uses wtd Σ. The wts are adaptively adjusted to min the cc of output. He uses a "smoothing λ" of 250 & says this balances "noise vs. response time" — but I'm not sure it's referring to "250 bits or symbols" … I think Mahoney is using Bits as symbols … ("λ = 250" means the relevant "sub-corpus" is ≥ 250 bytes long — i.e. it represents the most recent part of the total corpus)

0 One very effective method he uses: Uses dictionary of words: Dictionary gives pc to each word (I guess) This method alone gets [illegible] 1.66 bits/symbol for entire Calgary corpus. [illegible]
Using mix of Dictionary & several other pred'n methods, they get 1.522 for Calg corpus.
The "Mixing" of pred'n methods is certainly good for the Calg. corpus: Also if TM has no idea what external context is (unusual situation, tho)

He has several, apparently A.H. "tweaking" things that he does in computing pc's. — Much of this funny A.H. adjusting may be motivated by his paper on "ANN for Compression" q.v.
His first PhD thesis proposal (which he didn't execute) was in 1998. See D\PS\(4.16.05) for the proposal. ("Cost of Natural Lang Modeling") : by "cost" he means both time & cc.
He expected to do it in 4 stages: — I think he bit off more than he could chew for a PhD thesis) The part of what he proposed would probly be adequate for PhD!
20 {Character (letters) / Lexical (words) / Semantic (word assocns) / Syntactic (sentence struc)} — See Rating (136) for detailed theory analysis.
Hvr., indep. of Mahoney, progress in each of these 4 directions, continues & to some extent, certain of them are being integrated.

26 q.V. "Compression — Thesis Proposal — Mahoney" D/PS/4.16.05 : 44 pp — He mentions "Trigram topic" 11 MB corps → 1.211 bits/symbol !!
I will want to familiarize myself w. the ideas he mentions — They are oriented toward General (Compressing) pred'n
He mentions various areas of real interest. He didn't — Sch.j : Heil's : 1996 (I have this in D\PS\4/16/05) Sequenced Neural Net Compression
0 seem aware of work on PPM augmentation by the Girl (PhD thesis) — use of english words & also Grammar Markers.

SN Mahoney's method of adaptively assigning wts. to ll Pems, seems in retrospect, to ≅ what I had in mind, in response to the king-Koo result on "= wts" being better than "total pc of corpus thus far assigned by PEM".
Actually, there has been much recent hi level work on this problem Dawid, (1986?), and Russian group RHUC.
Mah does simple reinforcement "update — with a certain "decay λ" = 250 cases. The exact details are probly not very imp. — & results almost indep of the exact params) he reinforces. (I may want diff't λ's for each PEM) → But not it λ is our estimate of how long the recent sub-corpus is
At another point: "SSE" he modified prob'y slopes, like curves used for

4.17.05
STM

.00 : "CODEC's" in Telephony (= non-linear Modulat of (dc) Volts → Volts corresp so as to ↑ (lossy) $\frac{S}{N}$)

.03 Anyway: To what extent could I use Mahoney's ppm almost directly for Phase I Mota long? He does give the ppm itself .... Tho I'm not sure he gives all of the

.05 parallel predn pgms .... like the Dictionary pgm.

.03 – .05 is fairly by cc (depending, hvr, on how big corpus is & how many & which all predictors we use)

Anyway, we may be able to get as good results (for a corpus of ≈ Lisp functions) using ~~small only~~ PPM w. sub-tree contexts

It may be necy to make some of the PPM's in Mah's system "Sequential" — but ~~program~~ This would probly be easy for most Pgms.

10

0 : [SN] ~~...~~ 4.28.05 While I am interested in "General OZ" methods as a final Goal for TM, The QA predn (of Phase I) is impt & does demand really impt TM tasks. ~~In General, Predn is a special kind of OZ problem. There are several techniques that are particularly good for it (like EM). But in general, the "srch for reggs" may cover all that predn is!~~ Hvr. a "regy" is a rather general "type of "object"". "Finding sub-functions" is an impt part of it, but I dont think, the entire set of aspects..

{ Hvr. this is tautology since "regy" is defined to be compression = hyperfct.

Probably the only way to get "all reggs" is exhaustive Lsrch. — Tho probably not best way for any specific CB. — i.e. for each specific CB there will be a srch method and each corpus that is better than Lsrch for that CB. — e.g. for a long corpus, & short CB. Lsrch is not best, BV srch certainly is Better (unless the P.S. guiding Lsrch is V.G.!)

0

One common way is "Coding & Recoding": One can get to a local Max, hvr., so Backtracking is necy. Hvr, this is Diffrnt from usual Hill climbing Backtracking, in that one goes back to the "Best code" of an earlier corpus (= earlier pt. in the TSQ).

4.19.05
STM

p 1606

001 : Re: PPM predn! Would it ↑ compressn. if we were able to (define) predict "words".?
There would be "Multiple parsings" since t. words would not form a "prefix set".
Lemple - Ziv uses very long "words"!
We could use words as predictors only a/o use them as contexts, (predictors).

20.9.09

0 [SN] A possl. way to organize Lecture Series: Describe Strong AI in terms of "subgoals"
1) ≡ Top subgoal: Q ≡ i Eq. problems
2) Next : Sequential predn. / QA predn. Phase I (Only seq. string → pc predn needed for Phase II)
3) Next : Sequential string predn. for Phase I meta problem.
For each sub-goal, perhaps (for my own use at least) give nec. & suff. conds for each sub-goal.

For 3) I need to give p.d. over {PSM_i (string), $Prob_j$ (string), cc_k (real) (fixed?), Gore (real) 0 to 60 or ±∞. or 0 to 1 ≡ for pc probs ... impt.}

for 2) ⟨ String → pc of next symbol or next word or words phrase or "rest of entire corpus"⟩.
The strings will represent p.ms: If in ≈ Lisp they will be function words.
In other computer langs, t. reqs will be of different forms. I should study other lang's like for t. (assembly) Mach lang.)

0 On t. "2 kinds of PD's" Question! One p.d. is t. original PD that is used to evaluate models — to weight them.
The second is t. PD used for search. It is t. updated pd. of t. "Investigator" or t. "Scientist".
Hvr, it would seem that for induction one should always use t. updated a priori!
Usefl, If we start at time = 0 w. a priori P_0, & we given corpus C, then $P_{0,C}$ will
.35 induce t. updated a priori. [≡ $P_{0,C}$] Hvr, in inducing t. Contmn. of C, we will use a prior P_0 to evaluate models. If t. corpus is augmented by D, we can use $P_{0,C}$ to evaluate those models.
Hvr. for corpus C,D we still would use P_0 to evaluate Models. → (See 65.00 65.10 ← in particular! spec!)

STM

Soln. of "2 PD's" Question/problem! (.10–.21)

00 : Re: 64:30–.40 (on 2 kinds of Pd's): In numeric seq. predn. ("regressn") a good way to do it would be to regard t. time series as a TSQ. So after getting a very

02 rough fit for t. first 10 items, one uses Pd as t. a priori pd for t. next 10 items and so on.

This is the "Sumac": The "summary" for continuous params is a continuous pd — we have a mean & varc. (we can use any pd we like… But Gauss, Gamma, Beta

05 d.f.'s are easy to use. — Much known about them, easy to compute moments.

The discussn. of .10–.23 clarifies how [illegible] (.00–.02) works.

→ Also, maybe we can use it to deal w. equivalent ll codes getting results? → see 66.10 for v.g. partial soln

Well this is a diff[illegible]. problem!! Say we use a MtCar approach: for corpus up to (t_1) (which is small)

.09 We obtain (i remember) a set of continuous params that did well "up to (t_1)". → for augmentation of corpus, C, (.24)

10 : Re: 64.30–40 While using $P_{o,c}$ (64.35) as an actually equivalent P.D. (as well as a search guidance PD) may seem "Dishonest", it is not, we are still (?) using P_0 for t. a priori

.11

.12 of t. entire corpus.

If we feel jittery about .10–.11, we can actually apply P_0 directly to (C + augmentation) — so actually, when one does "use" $P_{o,c}$ to search for code for entire corpus (but w. t. part for C already coded) but one uses P_0 for a [illegible] Gaussian w.r.t. t. aug. corpus.

If we used t. "exact" (cc = ∞) $P_{o,c}$ then there would be no difference in results, between

(a) using $P_{o,c}$ to [illegible] as guide for augmentation of corpus, alone.

.20 (b) using P_0 as guide for (C + augmentation)

Hur, since we are not using cc = ∞ (Θ), we don't do that! i.e. we use our approximate

.21 $P_{0,c}$ for "Guidance" to search for code for augmentation, but then use P_0 as guide for entire (C + aug).

.22 Now: T. [illegible] can be used to find solns. for predns w. models having continuous params. —

.23 Which is what .00–.05 is about.

24 (.09) → Then for continuation of corpus from t_1 we use same set of trials as before but w. [illegible] p.d. obtained by a pr p (t=0) times (pc of corpus to t) for that set of params. We test t. next corpus segment $t+\Delta t$ on t. same set of params, using wts = pc of corpus up to t times a pri p of set of params. So we are using same initial set of

30 params, but w. progressively changing wts. — & these wts should influence Mt. Carlo selection of trials.

.32 Perhaps another — "Practical" approach: We start w. a very small subcorpus, & we obtain $\vec{\mu}$ and $\vec{\sigma}^2$ whatever t. pod value of t. params that derive from it, &

.34 & t. Variance matrix. (34.2) Then when a new data piece is added (which may be one or may be 10 new data pieces), we use $\vec{\mu}$ on that data & we do trials near $\vec{\mu}$ — and so that we control how much $\vec{\mu}$ & $\vec{\sigma}^2$ change due to new data. When a new chunk of data arrives, loop to ∝ (34.2)

T. main bad part about this technique is t. computation req. we would need $\sim \frac{(n+1)^2}{2}$ (n = no. of params) evaluations to get t. components of $\vec{\sigma}^2$. It may be possl in some cases to do O.K. w. diag elements only

STM

Getting rid of (degeneracy / duplication) on Models! 10.

Great Breakthru!

SM .34 .31

00 : ! So only n evals. We may need 2 extra pts for each eval, to get second deriv

so $\approx \frac{n^2}{2} \times 2 = \approx n^2$.

We could use this method to start out on many different rand. discrete models

Each model has different set of continuous params. At first we can filter out

bad models by using very small corpus & using a bunch of 'random valued

params for each. The promising discrete models then are evaluated more exactly

by usual n.l. optim methods. We then do 65.32 to continue evals of each

of the discrete models. We retain the best (say) few discrete models, — unless some

"" way back the "best" model.

0 Aft! Naturally, several of the models will be really identical. (e.g. $xy = yx$

& other xforms would change one model into the other. In general, to start out

If n models have same no. of params, we will suspect they may be identical - - -

& any rate for each no. of dims we will have a set of models — we will use same

set of param values for these models (to start out). If several models have same

outputs, we assume they are identical and we drop all but one & multiply &

apply of that model by the no. of same models.

z 17 This (cheaply) gets rid of much redundancy.

For more confidence in the identity of 2 funcs: either use 64 bit random nos or

2 x 32 bit or 4 x 16 bit: 4 x 16 bits is better because it covers a larger range of

20 input space. 8 x 8 bit random trials would be even better .... but would probly take longer.

(Rq in "properly" designed system, neither 8 x 8 bit nor 4 x 16 bit trials should fail about

same rate as one 64 bit trial.

This is for real m, real output funcs.

for any other combinations of strings & reals, we can sometimes get some distinguishing

of identical funcs by trying random inputs — but I suspect that we'll

have to do say 8, 8 bit random strings or 4, 16 bit random strings, ect:

One trouble is that if range of outputs may have only a few values

(In extreme case: T/F) — In this case, random inputs to 2 functions

30 would & same outputs, would not be so certain an indication of identity.

---

31 To what extent does 65.24 — 66.17 solve the SM problem? It is a real→real function.

The problem of retrieving the stocks can also be done by "tracing step" — so as to data

string errors, we are able to access more & more and stocks to use for "drivers".

---

34 → A possl. "soln" to the "cost of calling a driver" problem: Say we have 5K already scored stocks

& other indicators/possibly usable for drivers. Given a stock (to) be predicted: list all

5K indicators in order of abs size of cross corrln w. So. Then cost of accessing

the one w. kth highest corrln will be $\frac{1}{k}$ ("normed") $\frac{1}{k}$. (We may not have to normalize since

STM SN.01

spec
0 : 66.40 ; all possible use drivers — so we only need to know relative pc's of drivers: so it may be OK.
(Pc of it is an "improper prior")

Another possible pc of reference might be α^k, where α is adjusted so that k (normalized) expected value of k. choice is some "½ powerpoint" of the distribution of drivers.

On a simplest level, the "correlation" of S_0 w. indicator Z_i would be mean value of ⟨sign of ΔZ_i yesterday * values of ΔS_0 today⟩. Other "correlns" could be on basis of how well Z_i predicts S_0, given 1, or 2 or 3 ... ext. previous days of Z_i data.

One useful cross correln: say Z_i is Bob's value on day t. S^{t+1} is S's value on day t+1:

Then a useful cross correln is $\frac{(Z_t \cdot S^{t+1})}{(Z_t^2 \cdot (S^{t+1})^2)^{\frac{1}{2}}}$ ← averaged over T values.

If this value is large, it means that a large Z_t value is likely to be followed by a large S^{t+1} value.

T. system could be spending enormous amts of cc /chips for good driver/drivens & associated predictors. T. Q is, what is T. "Hard Problem" it would be working on? — May be finding good driver/driven pairs? → .16
Unclear as to how 65.24–66.17 (which is on how to find good predictive functions) would help much — I usually think of cross prodn as using very simple models. Perhaps we could use more complex models.

16 Hrr. finding obscure driver/driven pairs seems like not such a good idea, because there low pc means they are expensive to use. The 10 best pairs & Pc of 10 penalty in "earnings"!

0 SN 2 "Expo": One way to explain STRONG A.I.! Phase I is a Turing universal "Booster" for any sequential predictor that is used for its TM_2. However, this is not really Phase II. It's a system that eventually gets a v.p. TM_2 (= TM_1) for Phase I. It still doesn't understand Optan very well — but it is a v.p. "Predictor" for Phase II. ⟨Predictor means/Probability (G | PSM_i, $Problem_j$, CC_k)⟩ solving.

0 SN 4/23/05 In my early study of cfg's I assumed a "Preschool Grammar" for a group of Grammars to be found. If I actually used a Grammar Grammar to generate examples, a TM learning a GG^2, could it discover the "correct" Grammars? (Note: a T. GG^2 used is actually very simple & "Promiscuous" Grammar.

SN2 What would happen if I used Rudy Rucker's "Bopper" PSM but randomly changed some bits in the Original PSM? Not very interesting unless I could get a PSM to actually solve problems!

SN3 Perhaps Prediction of R. "Bopper's" next move would be a useful TM training problem? A TSR could be derived from progressively more complex "Boopers"

4.23.05
5 PM

0 : On Ph I [redacted] TM's problem is to get good pd for cands. ~~Tr~~ Its corpus could be ~~TM's~~ TM's [sx corpus, soln$_z$] pairs: possibly along w. t. best/ problem ~~guessed~~ for best soln$_z$. As is t. solns are simply ~~p~~ bc's < a threshold given by Trainer.

SN ☺ we can regard ~~b~~ t TSQ given by "trainer" as a "Training set of data" / as used in "Old Faithful" (?)

0 : (SN) On searching ~~for~~ subtrees: ~ worst case analysis.
A / related problem that has ~~been~~ analysed much: Given 2 graphs to find common subtrees

.12 In ~~our~~ our problem we are given a known tree w. known root, (this is the partial cand that we are probabilistically constructing). We want partial matches w. that (known) tree, in ~~t~~ all previous "successful" trials.

Say ~~[struck]~~ the previous "successful" trials have a total of n symbols in them. ~~The~~ The root of ~~tree~~ tree to be matched has a symbol w. / unconditional pc = P_0. So we have $P_0 \cdot n$ possl. ~~m~~ matches.
It will take ~ $\ln_k n$ to find each of them k = size of alphabet.

How many comparisons ~~for~~ each of those $P_0 n$ cases? Each of those searches is a small tree

19 search. T. no. of comparisons = t. ~~no~~ "largest matching subtree"

0 The number of subtrees ~~of~~ of t. known tree (.12) that have 1 or more matches in t. previous ~~s~~

21 successful trials can be very large. Can this no. be > n? → (28)

Of the $P_0 n$ subtrees of interest, most will be very small. Each of them will have a "size number" telling how many nodes they have. If ~~a~~ all trees only have branches of arity ~~[struck]~~ ≤ 2 then we will have ≤ 2 · "size no$_i$" ~~a~~ comparisons for tree$_i$. (tree$_i$ is common to "known tree" & at least one "successful trial". So "size no$_i$" itself × max arity will give no. of comparisons.

26. → So total no. of comparisons will be << n & > $P_0 n$.

28 : (~~21~~ 21) If the known tree has L leaves (= terminal / most distant from root nodes), then we have at least 2^L possl. sub nodes that ~~[struck]~~ are at depth of known tree or depth − 1.
~~For the whole~~ If t. whole tree is of depth, D we will have < $D \cdot 2^L$ subtrees. Since each

30 "depth level" has at least 1 less "terminal", I think we end up w. < $2 \cdot 2^L = 2^{L+1}$ subtrees.
Hvr, I'm not sure of t. 2^{L+1} result, it may be much larger. Even $D \cdot 2^L$ may be (very low / completely wrong)
2^L is a lower bound on no. of subtrees.

Def KT If there are N nodes in the "KT" (≡ known tree) then certainly 2^N is an upper bound to no of subtrees
But between 2^n & 2^N is a large gap!!

Instead of working on .12 ff, consider a simpler problem! How many trees are there w. just n nodes? I may have solved a more general problem in Sbarb (~1989–91) i.e. how many functions are there w. n nodes? — I'm not sure just what I solved. I got a functional (recursion) eq. and then obtained an / exact closed form soln. by guessing. ~~This~~ That soln. may be ~~[struck]~~ closer to what I want / need that .12 ff!

Spec. 69.20 —

5TM

.10 : Dropping the Q of the no. of possl. contexts (for 1-time being): The practical Q is to use these subtrees for predn. A fairly General Context, consists of short ~~[illegible]~~ that define sets of subtrees. Associated with each such context is its aprp ($\approx 2^{-\text{length of desc}}$) mult by the pd it assigns to the next symbol. It codes not all of the corpus because its contexts do not occur in every case. So consider the total extra compression it gives for the symbols for which it's relevant!

I ~~[illegible]~~ had a big discussn on this when I figured out how to use ordinary context subtrees for predn. — Try to find it! **4TM 223.31 – 227.40** is v.g. discussn. of this & related Q's.

10: : SN For regular PPM & PPM w. subtree contexts, it's likely that ~~[illegible]~~ we will not get v.g. predns unless the corpi are very large. This is essentially a Table lookup lrng. (Gross Memorization) For much faster "~~[illegible]~~ lrng. (i.e. much less ssz) we have ~~[illegible]~~ to use "mgmsk": These are ~~cheap~~ descriptions of defined sets of ~~[illegible]~~ ~~linear~~ subtree contexts.

Hvr, even tho. it would be "expensive" in training & in needed RAM (= CC) we may want to do it for PhI "Meta lrng" (= PhI's TM2) — This enables us to "pool" Corpus Data from many difnt lrng Domains. In my early work on sub-tree context (perhaps ~~[illegible]~~ near by preceding 4TM 223.31) I thot of pools of trees of same size — or the size (= Cardinality) would be key to aprp (= pc). Maybe something like $\frac{1}{k}$ for size k ⟨Note this is an "improper prior" & commonly used for "relative pc's". But pooling Trees of same size would seem to remove the

.19 whole point of sub-trees for presenting specific functions → .30

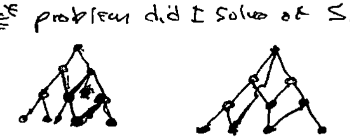
20 : 68.40 : SN ~~[illegible]~~ I think the idea was ≈ like this: Given l leaves and a set of functions of various arities, how many trees ~~[illegible]~~ with d functions, are there? ~~[illegible]~~ ⟨NB this notation will change when I find Searb 89-91 soln.⟩ NB Order of functions & the identity function (of arity = 1).

For d = 0 ~~[illegible]~~, there are l possys. $f(0) = l$. For ~~[illegible]~~ d = 1, there are $F(1) = \sum_{i=1}^{n_f} l^{a_i}$ ~~[illegible]~~ n_f = number of diff functions, a_i = arity of i'th function type.

~~d=1 has l possys v.t. identity function~~. This assumes ~~[illegible]~~ no symmetries (like commutativity) in function inputs.

.27 So maybe $F(d+1) = \sum_{i=1}^{n_f} (F(d))^{a_i}$ ← NO! | So just what problem did I solve at SAARB?

.28 I think the Functions were not trees but just directed graphs: A "concept net".



(N.B.!) Actually, I prob'ly want (in TM) something like this, rather than a "tree". (The Note I'm working on Contexts, which are not of — Functions)

30: (.19) → So it's likely that I will not "pool" in this way: but subtrees of same ~~[illegible]~~ size may get same aprp. — So .20 ff deals w. Cardinality of ~~[illegible]~~ trees as a funct of ~~[illegible]~~ ≈ no. of nodes. & may give ≈ value of pc (= cost of coding). — Hvr, Note 4TM 227.33

The analysis of .27 – .28 will be needed for pc comptn. — also I have devised a "context" data str. so I can get pc's of them.

While the no. of ~~[illegible]~~ functions & functional contexts is quite large, I can assign ~~an ordinal~~ linear (assign) order no. to each of them and store their case counts in a hash code.

5TM

.30 → Hmm, see (.14) for exact, simple Soln!

I There are lots of curious/dirties w. fuzzy stuff: Hmm, I think the analysis of 4TM 223.32 - 227.40 (i.e. the $\alpha, \bar{\alpha}$ coding) is correct: but the pc of derby & context is rather diff. i) The contexts I'm using do not have the generality of the functions I analysed in SAARB. Cf. 69.28 R.

.04 Second, the contexts run backwards! They are trees but different from the forward functions

.05 See 59.03 ff [Note 2Sep 59.39 on "Content Addressable Memory"]

.06 A method of analysis that I don't remember considering: Analyze the Lisp expressions themselves. — "All strings of Lisp symbols are legal functions" (≈ trees).

.06 might be encoded in country no. of functions.

If I do get into some difficult Estimating issues of determinations of contexts: Possibly Use a Monte Carlo runs & fit curve to data --- as I did on SAAR.

.09 → Re: the pc of .04 - .05! Since contexts are trees! perhaps I understood them by the Polish notation!

— In which case it may be easy to list/count them. Looking at 59.03 ff: It looks like the "backwards" form of "contexts" is same as the word notation. Backward Nodes have same "arity" as forward nodes — so does one describe them in Polish.

.14 → An exact way to get down pc of any tree (forward trees or backward (contexts) trees) is to multiply the pc's of each character used in generating the tree. The results at each pt will be integers — so: products of integers. There are some errors in picking Once Single trees. — since many trees are identical, thru symmetries, commutativity, assoc... ect.

.20 : Re: Implementation & eq. of (4TM 225.14)! Various approxns are possl. For summations the $\frac{N^N}{e^N}\sqrt{2\pi N+1}$ (Note Leaving symmetry M!) Better than Stirling for small N works for N=0, N=1. N=½

Usually only largest, or a few large terms are needed.

Also note (4TM 225.18) || the Stirling (simple) approxn can be used. Even for n=1 it is not too bad!

$1! \approx \frac{1^1}{e^1}\sqrt{2\pi\cdot 1} = \frac{2.5066}{2.71828} = .922$ rather than 1 ≈ 7.8% low but gives much closer to 1.0

$2! \approx \frac{2^2}{e^2}\sqrt{4\pi} = .91$ 4% low.

Usually, n will be >>1 & in general, errors in wts. of ≈ 8% would usually be Acceptable.

$3! \quad \frac{3^3}{e^3}\sqrt{6\pi} = 5.836 \quad : \quad \frac{5.836}{6} = .972 \quad : \quad 2.7\%$ low.

(4 26 05)

.30 → My Guess is that the exact analysis of 4TM 223.32 - 227.40:

(a) would be better if it were used to determine norms rather than contexts. This gives OSL, but also simplicity... but is it as accurate?

I think Better proof/understand.. | If you don't define norms, you don't get many multiple parsings — much

.37 (b) The mechanics of the analysis is equiv to a more exact formulation of the "escape costs" i.e. the pc changes in going to a distant "context length" or "function Tree size".

Maybe Wrong! Soap 80.00

It suggests that the complex problem of weighting "forest trees" might be done empirically by an adaptive weighting model of the escape cost "functions"

5 TM

[redacted]

10 The analysis of the derv cost of any tree (≡ context subtree) as in 70.14, is very neat!

[4·27·05] While it would seem that the eq. of 4TM 225.14 would be the "exact" way to do predn., it superficially seems at odds w. the usual PPM "escape" pc's. My impression: They use the longest matches w. past "reference" them to get prediction pc, they shorten down context — to first one w. valid predn. Each ↓ of context length ↓ wt. by say factor of α. α ≈ 1/n where n is corpus length, up to that time.

((AZ 141 ↓ pc for longer contexts, ↑ pc by freq. of occurrence of Match. It does not seem to be getting a monotonic ↓ in wt as context length ↓. (!! — This seems like a Big difference 10 — betw. PPM systems & AZ 141!))

11 **SN** How to get hyper precision pc's from Compression Systems w. 1 Byte resoln:
To get pc of next symbol; consider all (k) symbol continuations of each possl. "next symbol". I don't know how much precision one would get for (k=1)
As is to get pc for 1 symbol, one has to do 256 different symbol freqs: — to only be the first most likely 10 would be enough — so maybe 100 or (much less) 2 symbol contns.
Looks up freqs 1 grams & 2 grams.
Hvr, it prob. would be a lot easier to look at the C++ code & get pc's directly.

Could I hash code trees?

20 : Re: Sub-trees: All I really need is good way to store/index sub-trees.
I can use the PPM directly (as in PPM) or as "definitions"
Definitions have advantage of speed, compactness . . . ;
But disadvantage of Definite "commitment" — that can only be "uncommitted" by "reparsing" — which is usually done for only a few recent "levels" of definition.

30 **SN** [4·28·05] Hash coding of trees? Each node in the tree has a unique path to it. When a new tree occurs, we want all past trees w. same paths to any of its nodes.
The paths can be hash coded. Hvr, this means that every node has its own hash code.
33 Well, put these paths in Lex. order! — No need for Hashing. We are interested in the "content" parts of each path. Each node will have an address of its entire tree.
I'll have to work out the details of this: It seems to be mean any lol, but it may not be even marginally practical from an ec. pt. of view.

T. trouble is, there are an enormous no. of "paths". Consider a corpus of O(1) functions! This is a sort of predn algorithm (cond'l. pds) that needs been successful int. past — in t. sense of "accepted by Tracer".

Each symbol in t. corpus has an associated tree, w. leaves as "sets of productions" (see discussn of "AZ" pd in one of appendices of IDSIA report.)

Some of t. trees are short (of depth rarely); others are full depth of t. function def'n's. If M is t. mean no of nodes in a function, and one has N functions in corpus so NM nodes in all. Each node will have on avg M parent nodes on its paths.

Q.E. — Say you have a binary tree: depth d $2^{d+1}-1$ nodes:

			0→1
	$2^{d+1}-1$	$2^{d+1}-1$	1→3
	$+2\times(2^{d}-1)$	$2^{d+1}-2$	2→7
	$+4\times 2^{d-1}-1)$	$2^{d+1}-4$	3→15
		$2^{d+1}-2^{n}$	

So sum is $\sum(d\ 2^{d+1})$ so ∴ No of Nodes on next level must be deeper than that.

$d2^{n+1}-(2^{n+1}-1) = (d-1)2^{n+1}+1$ nodes in all of t. paths.

14 So ≈ (d-1) no. of nodes on next level. So ⇒ t. storage needed only d × no. of nodes in Corpus?

Say .14 is true: We could do prediction using "all path contexts" for each parent symbol to be predicted. For a sub tree of depth d' — we would have to $2^{d'}$ associated possl "paths". Each path would have an associated dfr over t. alphabet of associated SSZ. From this info, using t. ideas of

18 ATM 223.31–227.40 we can get a P.D. for t. next sy bol. — all of t. "path contexts" working in ll to get a final p.d.

... for predn? Are these methods about t. same? If not, just how do they differ? Is one much better than the other?

It would seem that "context path" method would be easy to implement by storing path contexts in Lex order.

HVR: Using these "paths" for predn is not t. same as using subtrees for predn. To get subtrees, I need t. AND of a bunch of paths. What I've been considering is a ll "OR" of paths, —>1 often. Say I had a list of all of t. "maximal paths" for a node, that had cases not past. Certain subsets of these paths will have a non-empty AND i.e. common subtrees as cases. What I'd like is common subtrees & big case count. Next have larger SSZ's in narrow predns & much wt. (≡ short descrs).

4.29.05 SN for ME-Carlo Lsrch! Generate trials randomly. Then use a hashcode of completed d'crn. of trial — to see if there is "previous work" on that trial. If yes, continue that work w. a counterpart Δ cc; if not, make a new hash slot for Δ cc on that trial. Hvr. one will have many duplicate functions w. distinct d'crns! One can suspect this is Gates over idealized problems?

SN T. trials in AZ are not "your standard trees" / New Functions can be generated of arb arity, & used in subsequent parts of t. cand. d'crn.

Can Price "Common path" trick of 71.33ff be used to do ≈ PPMon AZ?

[AZ = Alt Zrkl as in t. appendix of IDSIA Report seems to Not do recursion — but it's C for do Lisp (& Prmk), so it can do "If x=1, F(x)=3 else F(x)=G(F(x-1))"]

TM

SN | Differential Eqns are a way to define some recursive functions (≈ Prim. rec. functs).
Since they are for continuous functs, this would seem to go from a different
domain of "defines" than usual discrete defns of recursive functs.
e.g. is there a differential eq. that generates the Γ funct? — look in
Bur. Stands. Book. (Hypergeometric D.E.?); Is this continuous func. idea related to Shannon/Moore/Cuss work on defining Universality in continuous machines?

SN | In my earlier PPM to function trees (≈ AZ), I've always been
using "adjacent" nodes and "nodes accessible by continuous path" as "context".
In general, if we allow Nonsets as contexts, this will not be "effectively" so.
E.g. In raster representation of 2 (or more) dim. images. —
Our "very context" is on Nodes that are "exactly just k nodes behind" — where k is
the length of a raster line.

SN To prove that any CPM (Computable Prob Measure) can be realized by a suitable Turing Mc. output
w. random input! For each output bit, use the Method of Arith. coding. Tom Cover 1978
Peter Elias → David Willis & J. Rissanen.

: Main Big Problems for TM:

1) ≈ write a sort of popular explan of the present expected path! Partly as introdu. for discussn w. Marvin. Also Monty! 2 kinds of search/TSQ: Lsrch v.s. BU.

2) Work out exact/approximate PPM/AZ methods for Sequential Corpus. Compare w. standard PPM methods — see if big diff dience.

3) After 2) work out how to deal w. subtree contexts; (possibly subtree sets).
Both exact math & deriving Approxs are impt.

4) Designing Language (≈ Lisp) for Ing. in Ph I.

5) Investigate further, Machine lang. w.r.t. learnability of progs.

6) Devise TSQ's for Phase I (≈ Maple, Mathica ... etc, to start) — Eventually try to get it to Learn Theory of various Early A.I. prgms.

7) From good set of PSM's: Factor them down into ≈ Loopless CFG ...
later, perhaps looped CFG.

8) Design of TSQ for Phase II.

9) Lsrch on contin. param. functions: I may have this under control - but I'm not sure.

10) Lsrch for stuncts on string output. (Idea of "Counter plus insts in derivations" is good, but needs work. also need Good functional forms.)

5TM

11) For BU srch: Use of S-CFG: Then S-CSG & other enhancements.

BU srch is ≈ improved GP. $\frac{\sigma}{Mx-\mu}$ may be good — for "linearized case" — & G. Faves I normally use (for my several industr Prob solving) are & ≈ linearized ... e.g. Maley, PC, time, cc.

12) Long "English".

13) T. Problem of when to Make Definitions: How to represent/using them; How to "undefine" in Backtracking: When & how much to Backtrack (≡ recodify).

Rite now: ■ 73.21 & .23 are most impt: .21 is also part of "Lecture Series"

So outline of 73.21: "Road Map to Strong A.I."

Early work on Heuristics and Expert systems: Inspired by N.S. Logical Precursors & Humans abilities: GPS, ect.

In 1956 we all had ideas for A.I. but no actual pgms (Mc Carthy came closest by working on LISP ≈ a Math project for A.I.).

However, my main interest has been ML (Machine Learning)! I wrote first pgm in 1956, then 1960, 1968, 1975: Tbliss; discovery of cond. Prob. 86, 89, 2002.

5.2.05

SN In case of sym. args of a function! (for 2 sym args., say (commutativity), consider & treat as to have 1 arg: ↑ unordered pair x,y. Can this help in detecting, utilizing Symmetry?

SN In applying CFGdizy to ~~any~~ functions in any known lang (like APL, Lisp, Az) has already known ↑ large, & it's a unique parsing (by! (Contexts" for usual S-CFG's are normally trivial & involve a depth of 1. However, we can consider deeper stochastic contexts, which brings us to our present Generalization of PPM. So it looks like CFGdizy would ~~add~~ ~~explode~~ adding to ↑ PPM approach. (or vice versa ☺).

SN ≥ "Classic PPM" approach to targ. We find & (largest adj's cont sub trees) to ↑ present Predictors that have occurred in past at least once. We use (in 11) all of them that give > 0 pc to next (known) symbol (for compression). If none of them works, try all sub trees that have one fewer symbol, & use "escape prob" to approximate ↑ pc wts. If this yields no correctly predicting sub trees, form smaller contexts by deleting one symbol (in all ways) from present "deleted" sub trees giving another escape penalty ... & ... Continue until a context is found that predicts ↑ next (known) symbol.

The targ. can be used to get pc's for all of ↑ possl. "next symbols". — May/be not perfect" but it is "implicit" classical PPM. ~~Probably~~ it probably has a much lower cc than ↑ correct ~~full~~ LP way, but still much less cc ~~than~~ than full ~~LP~~ LP (≡ Az 19(c).

5TM

writer (theory)
(2) visual proc
reviewer.

0 : If 2 subtrees are in ‖ & have same wt., one merely adds their case counts.
This may not be legit if many of their cases are identical (i.e. "multiple double counting").
The only way I know to resolve this sort of thing is by compression Analysis.
Parallel codes will, in general often use case counts on identical events. — A Big Q is whether
Their wts. are really (very close) (identical).

.07 I Guess One of t. big outstanding Q's is: In my way of coding, I seem to usually
.08 get less wt. as size of context object ↑. (.085) PPM seems to get More wt. as size of context object ↑.
.09 Admittedly wt. is "fonce" ~ size of context object mult by pc of past corpus as coded by that object.
D Also we have to ½ multiply by no. of different parsing ways. ← (This isn't exactly "Multiplying" since t. various parsings have different wts.
Note that "escape" pc ≅ $\tilde{\propto} \frac{1}{n}$ ("n" symbols thus far)

In .07 – .085 this ↓ in wt. is ∝ pc of symbol. Hvr., in .085 – .09, t. ↑ in wt. is ∝ n, which is much larger. Maybe not! $\frac{1}{n}$ is about t. pc of t. symbol to be coded, for t. context that didn't predict that symbol.

It may be a Thm that t. largest context does get a wt. (viz AZ141) $\tilde{\propto}$ n times that of t. next largest context.

0 : (SN) On "improper Priors" In linear Regression, within a given no. of coefs, improper priors can be used to give relative wts to preds. Between d.f.s obtained using these improper priors, for different nos. of coefs, we can't use an improper prior; hvr., ! we are summing over d.f.s that are all of t. same size, so we have to use a properly converging (Σ=1) prior.

.26 ——— Best use a prior that was "useful in t. past"! Perhaps have an (optimized)
.27 function that looks at a problem & outputs an "expected no. of coefs" for that problem.
Then we use $\propto \alpha^k$ for prior of k coefs w. α adjusted so that "expected
.29 no. of coefs" is as given by t. function of .26 – .27.

0 ——— AH! In .26 – .29 "in t. past", we don't know what α to use, so use all values: Each value α w. precision Δ costs $\frac{1}{\Delta}$ pc. So we get total pc.

31 $F(c) = \int_0^\infty C(\alpha)\,d\alpha$ where $C(\alpha)$ is pc of corpus, using α=α. Say C_a & C_b are
32 2 possible continuations of corpus; then $\frac{F(C_a)}{F(C_b)}$ will be t. predicted pc of these contins!

An objection: T. pc of specifying any α between 0 & ∞ is not all t. same. Very large nos. have larger codes. Hvr, The point is, we assume uniform d.f. over all reals — as in an "improper" d.f.

To what extent is t. result of .31 – .32 influenced by what may be t. "true" a priori on α! Say k cost of learn??

5 4 05 76

5TM .09 – .23 & .29 – .31 Somewhat of a "Breakthru" in predn w. string output — Applies to both Ph I & Ph II. It really "Bootstraps" Ph I !

00 (SN) Rémi Ba[redacted]rrère : Computer Algebra!
PS: LISP – Mathematica 4.8.05 !) I made hard copy that I put in Wolfram's "Mathematica" book.
This Guy is excited about Maxima as a fusion of function, symbol, algebraic languages — & able to express approxns. in a nice, useful way. 8 page paper.
It discusses 2 very General Approxn methods that I don't know about: — "Perturbation Expressions" & "Variational formulations".
It is clear that there are several parts of Math that I should learn — so [redacted] to be able to do better approxns !

09 (SN) An early developement in Ph II: Int. "Grammar" of PSM's; To decide on t. next symbol
10 [struck through] in a "derivation", TM considers t. "recent state" of t. system as a Q & t. next symbol as
.11 an A & uses this as a corpus for a Ph I prediction.
Unfortly, t. QA problem of .09 – .11 is string → string predn. — t. kind that I'm least able to deal w. (tho there has been recent progress]. A suggested soln. to .09 – .11 that's "Not Bad": Try various [struck through] & O^i (deterministic) : Each will have a prediction score for t. past — [struck through] fraction of times its been correct. We then take some kind of wtd mean of t. predns of these O^i's. One needs a lot of [struck through] t. O^i's to cover all [struck through] possl A's (say t. A alphabet is [struck through] 256 [struck through] symbols ... (or probly we could do w. many fewer !)
20 Another way to do predn is by Nguts. Each [struck through] function [struck through] F^i would recognize a certain set of contexts. Each set has a prediction vector for next symbol ("A")
– We do wtd mean of (pc of F^i mult by pc of its corpus) (i.e. t. cases it "recognizes") — We can use the <, ≥ method used in AZ 141 to get [struck through] relative pc of corpus wts for various F^i's.
23 These F^i's are like "definitions".
.24 Of course .09 – .23 doesn't solve Ph II's problem — even at a prelimy level — It does provide for V.G. [struck through] s-langs for PSM's ... an impt part of Ph II's work.
Hvr, t. rest of Ph II's work involves doing (Prob$_j$, PSM$_j$, cc$_k$ → G) predictions; Now a string → Num predn, which we know how to do, to a large extent.
29 → HVR! .09 – .23 can be used in Ph I's main problem ! We normally use a augmented PPM to get cands for Ph I. We can use .09 – .23 to get much better,
31 much more general kinds of predns for Ph. I.

32 A BIG Problem w. t. forgg: We accumulate this large set of F^i's. Trying them all on t. past corpus would be (eventually when we have Very Many of them), very cc expensive.
We can either arrange a "hyper order problem" to decide which ones to try, or find some Standard way of ordering [struck through] [struck through] F^i trials.
One Good thing about t. F^i's : they can be trained !! → See 27.04 for one possl. way to use t. F^i's.

5/5/05
STM

:0 : IN Ph II, I'd like TM to devise a D.f. over PSM's for a particular problem, So there would be an emphasis on PSM's that would "do well" on that problem. As is, I got a D.f. on PSMs, & a s-funct that relates (PSMs, prob pairs) to a df of Gsc v.s. cc. — which seems very time consuming

04 EN For BU such we may want to do strict Mt Carlo trials: In this case, in 76.32–40, we can choose the F^i functions in pc order until we get one that says "yes" — then generate t. trial according to t. pc's assoc. w. the F^i funct that was chosen.

07 Actually 76.29–.40 isn't so easy from another pt. of view. Say F^i_z is a possibly useful parameter of Ngrams: To evaluate $F^{z^i}_z$ for predn, we have to try to find cases in past where it was relevant ... Not so easy (cf. cc!). Testing all possys in past is too expensive. We may try Sampling - Mt Carlo-wise.

13 One way to save cc.: say t. following symbol is α. We examine all cases in which α occurred & we look at (set of previous contexts). Can we find a common characteristic w.in that set? We can try various F^i's on that set. Using Lex Ordering (as in PPM) we can easily find sets like. We can also easily find contexts of seqs 2 or 3 or more symbols using t. Lex ordering trick.

Hm. Re .13 ff: It would also be nice to have negative cases (In addition to t. corpus of positive cases).

:40 Say we simply try to Compress t. corpus of positive instances — i.e. find regys in it — This is induction on (ordered?) set of strings! One way to do this is to consider it as a "language" needing a "Grammar". We can try CFG, CSG etc. This seems to be drifting towards recognition that t. corpus is a lang that we know t. d-Grammar of. ← But what about t. idea that this new approach would give us t. power of "Ngrams"?

:30 → Actually, we can apply Ph I zoo level to Ph I level 1 directly! Say Level Zoo is t. Seq (QA)₁, (QA)₂ ‥ (QA)ₙ! Level 1 is $O^j_1, O^j_2, O^j_3 - O^j_n$.

We give t. system various kinds of QA problems — including those of extrapolating seqs of strings or regression of single strings. i.e. either Q = string₁,...,stringₖ : A = stringₖ₊₁

:34 or Q = string of symbols, $a_1, a_2 \cdots a_n$, $A = a_{n+1}$ from such capability, we can then predict the $O^j_1 \cdots O^j_n$ seq mas a Q as ; or we can regress

78.00 Spec

5/6/05
5TM

00 : **77.34** $z_1 \ldots z_n$ as t. beginning of O^j_{n+1}, & we want z_{n+1}, t. D.f. on t. next symbol.

Of course, TM must have had ~~ade~~ an adequate TSQ leading to such a problem — or it will do poorly!

Also, even if 77.30 ff were done perfectly, we would still have only a v.g. PHI, that didn't "understand" what "optimization" meant. It would simply give a very good "a priori" for O^j_{n+1}, based on past O^j's: T. D.f. wouldn't understand/know what t. goal of it all was! — i.e. good prodn of t. QATSQ.

0 (SN) Re: ↑ I'd like TM to be able to divide up its corpus into sub-corpi (by indexing t. sub corpi, say — which trainer could teach it). Say it did this (or had this done for it). A q. is just how to "share" info (and how much) betw sub. corp? This seems to be an Impt. problem. One way: for each subcorpus, we have a vector, telling how much wt. we give to data from ~~othe~~ other sub corpi: so we have this $k \times k$ matrix for k sub-corpi. (I don't know if matrix is sym. or has other simplifns.)

Anyway, T. system is able to look at/new problems, assign ~~sub corp~~ them to sub corpi & to modify wts in sub corpi.

Also its able to merge old sub corpi & create new sub corpi by ~~Select~~ gathering together a subcorpus based on former members of other subcorpi.

20 **Re: 77.30ff**: It's not clear just when in development to start this kind of self improvement. One could just start out by periodicly testing to see if TM can compress ~~of~~ t. O^j sequences. If it can compress by some Δ Then we can figure out what fraction of time to spend on self improvement. See 4TM12.33 for how to do this. Stuff leading to — Go over this now: it's imp. idea

10 ! (SN) One "not bad" way to do SI for PHI! Do search for predictive subtree contexts. T. search for single context sub-trees is perhaps simple. T. search for "n-gons(?)" may involve "recogn. functions" of F^2 76.20, is more diff (>> cc).

This doesn't seem to be "Boosting" (Regenerative) hvr. — its just a way to improve "open loop" t. zero level PhI

Certain kinds of subtrees & perhaps subtree sets can be represented as vectors like Zato Codes. — And we can use a kind of dot product w. present situation, to get wts of relevance.

5/7/05

STM

~ Rem .00 , wednesday .20

This is on s functs on string predictions

A summary of various kinds of predn of strings (actually individual characters).

1) Simple contexts as in PPM.

2) " " but paths thru a tree

3) Contexts that are single substrings

4) " " " " sets of substrings.

5) F^z ("Recognition Patterns" type contexts). They are a variety of 4) : A way to describe a set of contexts.

for 1)...4) we use Bern predn.: Exact context describes a subcorpus : I guess we should use $\xi \cdot \alpha, \alpha$ method of coding.

.03R — But what about the multiplicity & modality of D.f. and wts of contexts? — associated

The F^z are unions (; they/and) sets of {sets of characters} — but the characters need not be sets of adjacent --- they can be placed anywhere in the tree. Also they can include sets of substrings of each to chars that have never occurred before, see 11,.13

Hmm, for just about all of 1)...5) .03R poses a problem.

Also, we can view each of 1)...5) in 2 ways: (a) as pure context w. an assoc P.P. on the "symbols to follow"

(b) As pure defined objects, that are to include the char to be predicted. As defined objects, the framework of .03R must hold, since we have all these parsing methods. So does .03R) hold for contexts (that are not "defined objects to be used in encoding a corpus") ??

6) An older strong O^3 type of predn, looks at the entire previous part of the function being generated & uses (as a function of that "previous part") a d.f. on the next character.

7) See 92.00 - 93.00 also 93.24-.25 for a column of 79.20 -.40

SN One way to get a good (?) universal (?) d.f. on strings as a function of the past ξ . We have a bunch of ξ-functions, each with its own wt. These functions are any ξ plus one or more initial symbols of $A\xi$. So A is generated like strings in PPM the corpus in using PPM — possibly one symbol at a time.

Each of the ξ functs has a d.f. on one or more of the chars in the alphabet. Like in PPM, the ξ-functs that have a d.f. on very many chars, tend to have less wt.

If I could work .20 ff out, it would seem to make it not so ex pensive to find pc's of any conceivable "A" — just as PPM can assign a pc to any particular continuation of the corpus.

In thinking about "string output" predn : I tend to think of a function often outputting a single string w. hy pc --- Prediction of these strings, of course, may take more cc. .20 deals w. The latter aspect, not so much the former.

A similar Related to .20 ff : A function looks at the tail end of a corpus (augmented by a tentative symbol) & assigns a pc to "next string", (also maybe a wt.) — "that's trmb" — .32R — It could assign wts to partial strings completed w. "don't care"

Any funct that assigns pc's to any strings $\equiv$ a "ξ-language",

For strings w. condition .32R, we can conceive of a new kind of "Grammar" — for L→R on line parsing. If we had a (stochastic) grammar for complete ss only, we could define a grammar to deal w. .32R i.e. essentially L→R parsing. → See 92.10-93.10 + 93.24-.25 for continuation of this

5.7.05

STM

How to do Sub-tree Contexts in the PPM manner: .10–.14 w. trim of .19–.21

0:70.33 : [70.33] (SN) I have been trying to look at PPM as an approxn of AZ141! But it looks like it isn't!

It's an independent method of coding. 4TM 223–227 Discusses AZ141.

PPM certainly seems much easier (etc.) than AZ141 — Tho earlier than 4TM223 I think

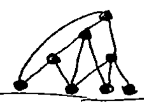
I worked out a way to do ≈ AZ141 — but I'm not sure it was rite — it may have been just an approxn of PPM!

72.00?? — NO!!

.06 Anyway, recently, I worked out a way to do PPM-like coding for sub-tree contexts, Using escape pencil tree, and 2° other coding costs (.10) — Very close to 4TM 223–227

(SN) I have been considering simple function trees; but the kind AZ generates is!

I think they are easy to count.



← [(SN) I'm not sure I included this type in my analysis! (at SAARB)]

0: .06 Anyway, Thought I haven't been able to find it! The way it works, hvr., we find the largest contexts in past consis w. present context. Say they are all of size k. If the case counts of all of this set include the correct pred'n, of α we use total case counts for the pd. of (next α symbol) (A possl. problem w. multiple counts. — Maybe not, if all contexts of size k are mut. exclusive.

.14 [α] If the size k contexts give zero pc for (next actual α symbol) do escape "to" Contexts of size k−1 & make pd from their case counts: if pc = 0 for correct predn, loop to [α] (.14)

A more exact (perhaps) method starts by looking at all contexts of (Max size (k)) that fit. It then uses pc to specify which of them give pred | w. pc > 0. of the correct symbol α. If there are r contexts of size k, — The coder does know both k and r — then there are 2^r differt possl. subsets of context to specify! (rather Expensive!!) — one of the 2^r is None of them → "Escape" to size k−1.

9 → Another approach is to use $pc = \frac{1}{r}$ for each of the ll codes that do give pc > 0 for rite predn.

20 → so if r_1 of the r contexts do predict acceptably: we have r_1 ll codes: each of pc = $\frac{1}{r}$.

.21 So: we just add the case counts of the r_1 contexts (They are mutually exclusive) and give the sum, wt. $\frac{1}{r}$.

22 { → Hvr. the $r - r_1$ contexts that predict α ("not α") also each give legit codes of pc = $\frac{1}{r}$. Which would suggest that we have to use all k sized contexts in ll — & simply add their case counts.! }

Is .22 Legit? Do we have to use all ll codes? — My impression is "No": that the $n - n_1$ codes would "Not code the corpus" — That .19–.20 is correct.

26 So .10 – .14 is ok. if trim of .19 – .21 is added.

The major problem w. using sub trees as contexts is the (apparent) diffy of keeping track of them, storing them, getting access to (the relevant subset of) them when needed.

0 One poss. way to do this is to have, say 16 (= 2^4) parallel stores of corpus: one for each of 16 paths backward (for binary nodes). Each one is in Lex order, like PPM — The store & index paths backward to distance 4

Another way is to try to find TMPT tree contexts. How they could be stored & accessed when needed, remains to be discovered!

Another way is to use definitions of F^2 funcs (≡ sets of subtrees). They have advantage of needing less SSZ since they include subtrees that have never occurred. Ideally, we could devise a function that would look at the suffix of a corpus & get a P.d. over F^2 = nymsts. that are most likely to be relevant!

5/8/05
STM

00 : A way to get good prediction "P" functs: Collect a large set of context (sub trees) that precede each possl symbol, & look for regys within each of these sets — try to compress each of them.

Maybe that kind of compressn is not so good, I'd like ways to use the sets for categorizn.

— i.e. given an element (≡ tree) to find the d.f. of sets it is likely to be in: A categorizn problem. Perhaps SVM could be used — tho for elements that are strings, I don't know if SVM's are usable. If I could find good "features" of the trees, I might use "decision trees" lists

06 Actually in the same spirit of the above is regarding a "pd on the next character" to be a QA problem for PHI: "Q" is the context (postfix), "A" is the next character: The corpus is the set of all (postfix, following symbol) pairs that we have in the corpus (If the continuous corpus has n symbols, we have just n-1 pairs. .06 ff is a v.g. way to regard PHI's self improvement! (≡ SI) —

0 That we want α (the next symbol) to be a pd on the postfix of the corpus thus far — so we try various d functs to get this, each D funct will be into a fraction of the times, P (→ PC).

Or, we may be looking for Boolean functions on the past (→ T/F) (≡ Obs) in which the T's (or F's) give a narrow d.f. on the next symbol, α.

In general, I want to try to integrate the QA problem into form of 70.20, & the problem of generalizing what PPM does — prediction of next symbol/token.

: A really good way of predn is making Daughters & periodically Reparsing all of the corpus. — Think more about that!

(SN) The example of Clark's not working may have been a Decision Tree (MDL) followed by Pruning (which the Dev langth? — but gave better predns).

My Impressn of Pruning Dec Trees (pp 662-3 in A.I. a Modern approach 2003 (2nd Edn) is that normally (un pruned) Dec. Trees, uses the "Best" split available — it does not, hwr, consider ssz & pc of the Model.dern. In the pruning discussn, ssz is considered, but Model pc is given a default value (i.e. same value as "Null hypoth.") — They actually use "Hypoth testing" formalism to decide if a "split" is legit.

The "95% confidence level" (thats "standard") is equivalent (I think) to a certain apriori of Model. — (not exactly; .30ff is better explanan.)

30 → The way hypoth testing works: it decides the relative pc of the Hypoth vs. Null Hypoth. I would (in ALP) express this as relative wts. of H vs. H_0. The 95% conf level means 5 to 95 or 1 to 19 ratio of wts of H_0 vs H for acceptance of H. This is actually not v.g. for predn. (tho maybe o.k. for "Hypoth testing" which is a bad idea anyway!) NB: .34

.34 I guess they assume null hypoth (H_0) & H have same apriori: which simplifies things.

I would probly do all splits down to "50% confidence" i.e. when H & H_0 have = wt.

36 Another term on normal Decision trees would be "less Greed" i.e. Consider other orderings of splits.

5.10.05
STM

10 : On "splits" say before split we have n_0^0 0's & n_1^0 1's.
After a split we get n_0^1, n_1^1 || n_0^2, n_1^2

The original pc of the data was $\frac{n_0!\, n_1!}{(n_0+n_1+1)!} = P_0$

After the split, pc of data is $\frac{n_0^1!\, n_1^1!}{(n_0^1+n_1^1+1)!} \cdot \frac{n_0^2!\, n_1^2!}{(n_0^2+n_1^2+1)!} = P_1$

If $P_1 > P_0$ the split does something useful.

My impression: if $\frac{n_0^1}{n_1^1} = \frac{n_0^2}{n_1^2} = \frac{n_0^0}{n_1^0}$ then $P_1 < P_0$.

{ $\frac{n_0^1}{n_1^1} = \frac{n_0^2}{n_1^2}$ always implies [both] $= \frac{n_0^0}{n_1^0}$: in fact, the equality of any 2 of the 3 ratios implies the third equals the other 2. }

Otherwise, if there is any inequality in the ratios, $\exists N$ > for all $n_0^0 > N$,

.11 $P_1 > P_0$:

12 .10–.11 is probably provable but I'd like to be sure! also know how large N must be for various deviations.

On second thought, I'm not sure .10–.11 is true! — it may be necessary that the ratios in the children differ in a "useful" way from that of parents.

Try some experiments w. nos before trying a proof!

start w. $n_0^0 = 5$, $n_1^0 = 5$: then do various n_0^1, n_1^1 pairs (which imply n_0^2, n_1^2 values)

0 (FN) the foregoing analysis is for precorpus = 0,0. which uses apri wts of 1 for 0,1.
Using apri wts of .5 would give slightly different results. (probably smaller SSZ needed to get equal wts of H_0, H. = Dirichlet Dist)

On .00 ff: If the corpus really was formed from a deterministic Decision tree, then — — —
This gives a way to try to find that model. Hvr, no such model may exist ... i.e. it's a stochastic system. In this case, we can take a note from PPM!

We will say, end up w. a specific (symbol, result, classifn) for each (set / string) of "features" / "properties"

If we are doing predn — trying to compress a corpus, this is not any good. We have to be able to get pc's of other poss. symbols. (Whenever we make a split, we are not 100% sure of that split.) [Get back to this later!]

0 Consider only binary classifn. Each final classifn will be 0 or 1 (w. pc =1 only).
Say it is 1 & we have to code a zero! we can always do this by backing up only one level. The (If we have > 2 classifn symbols we may have to go up > 1 level), previous level will give a pd over 0 & 1 that is not deterministic. Alternatively we can stay at the final level & make proby ≠ 0 or 1, because of finite SSZ.

Hvr, when we have > 2 ct. symbols, SSZ or not we have to back up at least one level to get all symbol pc's > 0. To backup one level we may use "escape" pc; But

5.10.05
5TM

o. I'm really not 100% sure this is the best way.

But any way, the merger of Decision trees & PPM seems like a v.g. idea. [It may be somehow also related to BBN's.]

The merger enables us to get a "pc for every legal symbol" from dec. trees — so its one ~~[illegible]~~ kind of useful Model for string → string predns because very possl. cntprs is assigned a pc > 0.

Normally, the way ~~DT~~ DT Dec. tree works: we have a set of (strings/objects) that we want to classify. We have a seq. of Recognition functions that do (partial) classifn. Each R funct. will classify an object Ob_i into a (or more) internally indexed categories. Each category/class is then acted on by another R funct that further categorizes Ob_i, etc until we come to last R funct that categorizes Ob_i into the set of (initially desired) categories. The order "next" type of R to use depends on the ~~[illegible]~~ Output "category" of the previous R.

o. W.r.t. the Training Corpus: ~~In each of the~~ we apply the above set of R funcs to each member of the set of examples. At each point, a R funct will have ~~[illegible]~~ a sub set of the examples ~~[illegible]~~ divided up among its own internal categories: each internal cat. will have a population distribution of final classn cats. These distributions can be used whenever an "escape" chance for an "unknown" problem into that R funct.

[SN] Remember, when estimating pc of "escape", the value of a assoc n is the ssz of the context escaped from: Often n will be quite small: < 10. Even if corpus is very large, n will still be very small, because the context will be a long string. If there is only 1 symbol type following the largest context, its pc is $1 - n\frac{1}{n}$ = (1 - (escape pc)) (not 1.0).

o. Each F_i function has 2 or more output categories: unless it is a terminal F_i — each category will feed into another "F_{ij}". If we use them for compression of serial text, the ~~[illegible]~~ cardinalities of the sets of terminal cats in each F_{ij} will increment w. each trial, as soon as the final (cat. ~~[illegible]~~ of ~~[illegible]~~) is known. So we can use it for text compression & get compression comparisons w. other methods.

---

o. Useful kinds of "F_i" functions: 1) The "OS" funcs on "ob/op" a (gabor 2) Is it about Math/Chem/Phys ...? 3) If its about Math: subcategories Algebra, Geom.: Is it eq. solving?

(These cats. can be learned (by user) as a part of the TSQ. — Then put into situations in which these cats would be useful.

One kind of cat. that has been studied: "continuous Param x is > k". We have a continuum of tests — one for each value of k. We can do an n way split by putting in n-1 k values; each category will be betw. 2 k's. This could be regarded as a "Quantization" ~~"Pozuring"~~ of data — to put a set of continuous variables into useful "Bins" — what are good set of bin boundaries?

One can do ~~[illegible]~~ continuous ~~[illegible]~~ of "Bins" using continuous functions. An example: Age: All s's (subjects) are either of ~~[illegible]~~ age < x or ≥ x. After such a "split" → (See 84.12)

SM: Genzn of Cover's ~~SM Schema~~: 34

0 [SN] A Big Criticism of PPM & (Some Augmented PPM) & of DecTrees! That the concs. used depend on contexts that have explicitly occurred in the past. Elements w. elements that have never occurred, are never defined. This means (I think) much larger ssz needed for lrnig (i.e. slow lrng).

For PHI slow lrng (large ssz) is not so important! — what is important is that the Universe of Concs available be very large, relevant to probs to be solvd. — i.e. PhI can need excessive ssz, but if it gets good concs for prob in PH II — that's o.k.! ← But check on this!

/

1 : Decision Trees: Any decision (classifiers) loses info. The m̄ of a good decision (tree) is that it loses as little as possbl.

12 ; (D3, 40 spac): How do we do new continuous or discrete splits? Well age & height (& wts, ect) could map to a pt. in a dimensional space, rather than a pt. on a line. Then later, (discrete?) partitioning of that space would give "splits". (.195) SVM's is one way of partitioning an Nrspace (by → hy dimensions)
14 A partition can be unconnected ("undrafteble" is <18 & >40 yrs.)

(.195) brings us into the ideas of linear & (SVM) non-linear partition functions. — A very large area of "Rsch". Hvr, unlike SVM, we have a continuous Gore for partitions (?) (Maybe not continuous, but certainly not a funct of only a few "support vectors".

(ANN) (radial basis functs, see .20 ff) etc. can (partizes) be usefully regarded as decision trees on a continuous space: The idea of (.10 - .11) (to lose as little info as possl. by making partitions) is a universal Gore.

20 Lots of ways to do partitions in n dim space: One is (maybe Radial Basis funct?) to catz. a pt. Corpus is pts in space of value +1 or −1. Do wtd Σ distance from all pts (Distance can be any ↓ funct: Gaussn, $\frac{1}{r}$, ect! It need not be normalizable since only finite sums avrgd. If wt. of $\binom{<0}{>0}$ do cat $\binom{0}{1}$

.23 One can map to space of arby hy dim. before doing .20 − .23 (as in SVM)

One can do all discrete partitioning first, then all continuous space partitions.

Re: ANN : This may give a better (no overfitting) way to do ANN — Hvr, one will have to minimize no. of Nodes & wts

② A way to get good predns out of large ANN: Divide Corpus into two & (C₀)(C₁ fastest.) :

29 Using random initial conds, to use/formula feasbl not to hill climb & stop when we get to max Gore on C₁ (or so) do/random many initial trials like .29: The ... of that gore on C₁ will be a good estimate for future.
I'm not so sure of my reasoning in .30! The idea is that each initial cond. gives a code for C₀. Hvr, it does not — it gives a sequence of codes. Picking the best one, still gives us SOP. If the random init. conds each corresponded to a code for C₁, then it would be logical to average them w.o. fear of SOP.

34. [SM] Say we have a SM strat. w. k continuous params: for each small hypercube in k space we invest some amount. We can then compute the amount of yield at the optimum pt at the end of the run. It is equivalent to frequent "re balancing" amongst the k dim. param. components

5.12.05

5TM4

:Q how does SVM get > 2 classifications? One poss. way: Say we need 10 cats.: $c_1, c_2 \cdots c_{10}$ In the hyd. space, we find a hyperplane that separates the c_i cases from the rest of the cases; so we need 10 hyperplanes.

A Big Q about SVM is "How likely is it that a random N.L. xfm will yield a space in which the cats. in question are linearly separable? →30

ANN (feed forward) can be rather directly trained to do multi category classifn: Say we have 10 output nodes. The node w. max output can be regarded as the best way to classify a given input. Also, the distribution of output levels gives a measure of how much confidence one has in that classifn. One could in fact, use FF ANN to get probys for each cat.

Well, anyway, I want to summarize my conclusions about Decision trees and their genzu. Essentially, a Dec Tree is a [classifn. rule tester?] kind of cascade of a bunch of classifn. rules. These rules can be any d. or s. mappings from (num / strings) to (num / strings). They can be continuous a/o discrete. All of the techniques in the book "Pattern Classifn" are relevant.

The similarity of all of these problems makes it clear that they all have similar Gorcs. Also, that they can all be generalized to Q.A. problems (they are all Q.A. problems.) — Here, in QA, I ask for P.D. on Q. Classifn usually asks for a "Best" (≡ "Decision") on what the Q should be.

This is very important if I can show similarities between the classifn methods — so I can make grammar for them all, to extrapolate them.

:(04)→ I. problems very similar to n.l. regression: we want a funct of K variables [.31R] [funcs of k-dim. vectors ≡ corpus] w. l params. Closer than to l.i. regression: find a function of K vars that separates the sample into 2 sets — so hy values in one set, low values in another. — This maps corpus into a 1 dim space.

A we have to find the l params to max some .. of them. In SVM, we have vectors, a corpus of k dim. vectors, we then select a l >> k dimensional (n.l. functions) xfm to a k dim. space, and we want a l dim. hyperplane w. a certain Gorc. Presumably the ssz is >> l, the no. of params in the hyperplane. → 86.30

In the case of regression, the initial choice of functions w. l params is in a discrete space of functional forms — so it is cheap, information-wise.

For SVM, say we have a corpus of S, K dim vectors. We [x]form into a l >> k dim space via the

5.13.05

STM

0: : xfm Σ_n (i = 0 | ℓ) ⇒ [I found a very neat way to express all functions of degree d of r variables — including all lower degree terms.

$\Sigma_n = \sum_{i=0}^{k}\sum_{j=0}^{k} x_i x_j \cdot a_{ijn}$ gives all quad, linear, const forms. $x_0 \equiv 1$.

05 n = 1 | ℓ

$\Sigma_n = \sum_{i=0}^{k}\sum_{j=0}^{k}\sum_{m=0}^{k} x_i x_j x_m a_{ijmn}$ gives all cubic, quadratic, linear, constant. $x_0 \equiv 1$

Because of commutivity of mult., the "a" arrays are very symmetric or we just sum over $i \leq j \leq m$ → 88.01

For each value of d = known degree of Σ, we can compute no. of indep. terms need.

10 We mult by ℓ to get the no. of coefs needed of a ℓ dim space.

11 Say we chose the "a" at random in the unit hypercube. (we may want a larger space). → 88.01

N.B. If there are ℓ datapts in a ℓ dim space, then if each data vector is a basis vector, we can probably use a hyperplane to separate the set into any

.14 desirable 2^ℓ distinct dicotomies. If we have ℓ dim space and SSZ = ℓ we can (.24)

(SN) in 00.11 We get ℓ times as many a coefs as used for one coord.

Hvr, we could just use one set of "a" coefs & chose sub-sets of them for each dimension of Σ. An easy way would be to have $a_{ijm\ldots} \equiv 1$ for all indices, then chose subsets of the a... at random to get the Σ components. Easy to do because

.20 we don't have to mult by a..... We could also make it less likely that two diff. Σ components would be identical (tho no great tragedy if they were!) by allowing the coef "−1" half of the time — so even if two components chose same set of coefs, they would be unlikely to all have the same signs.

.24 (.14) → linearly separate them. Hvr, what's the chance of extrapoln to SSZ > ℓ? It would seem that the h.plane would be ad-hoc for ℓ data pts in ℓ dim space. It would seem that larger $\frac{SSZ}{\ell}$ (for data set) would give better extrapoln. — The Vapnic would say that the "separation margin" is the thing that insures good extrapoln. — This way, hvr, be functionally related to? So for fixed SSZ, chose ℓ to maximize separation distance.

30 (85.31R) → This is actually doable! : Maybe easier than SVMs?!!

We have S, k dim vector in our sample. We want ℓ param function that will separate them w. max distance betw. the 2 sets (In the sample we know which set each case is in)

To start we want function that minimizes / ℓ dim. overlap of the 2 types of cases:

hy — we have 6 overlaps here. We want to ↓ to zero, then try to get further distance betw 2 groups of cases.

d — overlap

low

So the work consists of 2 non-linear optimizations.

The first is discrete, the second continuous.

.39 We can actually do one continuous optzn — we want the distance "d" to be as small as possl. — eventually as negative as possl.

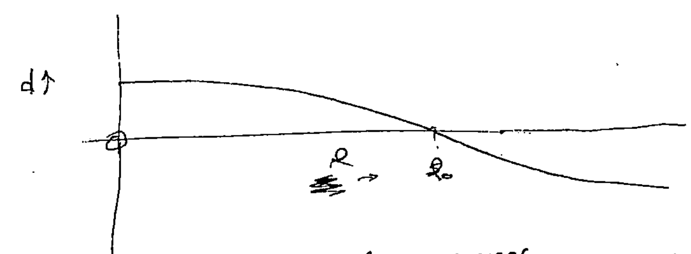
5.13.05
STM

(SM, 32)

.00 : while f. optzn of 86.39–40 is continuous, it probly has discontinuous derivatives every
.01 once in a while. Because of discon deriv., it's not so easy to optz these functions, → (.13) → 88.30 for better way to deal w. discontinuous Derivatives

I guess we want small "d" & small no. of params in f. function.

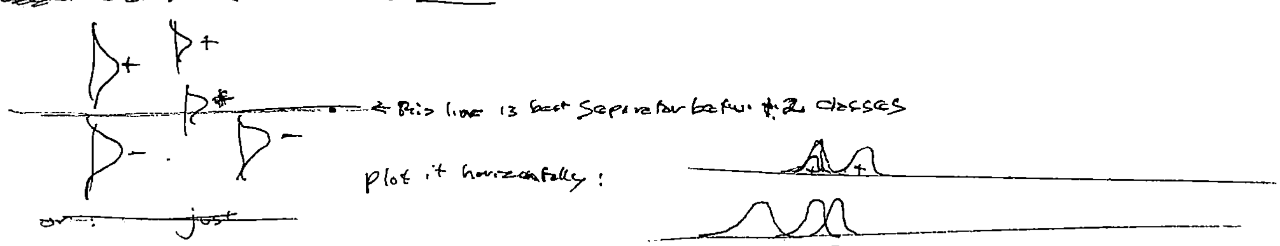
Apossl plot of d v.s. ~~[illegible]~~ & ~~[illegible]~~



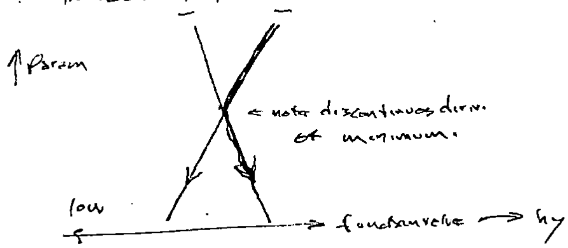
• For certain data sets, separation is impossl., since we have same vector labels +1, −1.

• We may want to remove such cases from the corpus.

3: (.01) → One way to ~ deal w. this is to use relatively large Δx_i when computing derivatives. —
Would like to be able to get second derivs but probly the discontinuity makes it impractical!
A way to smooth out the derivatives: Instead of having points for each case, we have ~~[illegible]~~ Gaussian (or whatever) "smears". Unclear as to how one "scores" "overlap".



say no "smearing". Vertical coord is variation of ~~[illegible]~~ a param of f. classifn. function.
Horizontal is motion of + & − cases.



Other than smoothing out discontinuous derivatives (which I haven't yet shown how to do! —
Another way is probly what Vapnik does:
Move along edges toward optimization.
→ Hvr. ~~[illegible]~~ in his problem, one always moves in st. lines — but not in f. present problem.

But I'm not really sure of this either! — He does solve a Quadratic Optzn. problem, hvr.

.32 For (SM) (1) Driven has 2 levels (up/down) drivers (including Driven) — when used as driver — have 3 levels!
Try to get 2 days (previous day only) but if not² available, one, or none — using "escape" facility.

(2) TM is constantly looking for "compressible" spacio-temporal configs of securities & any other available time series. Main thing sought is ~~temporal~~ diffusion of info in temporal direction. When one has a "Home Computer" one should have TM always in Background, looking for unused cpu cycles that it could put to work in f. SM srch!

5.14.05
5TM

Eypo ... (40?)

88

.00 .005 : Q: Is it true that problems in which ∃ a classification Rates deterministic — are usually "Toy Problems"? So all of MATH has to do w. "Toy Problems"? ☺.

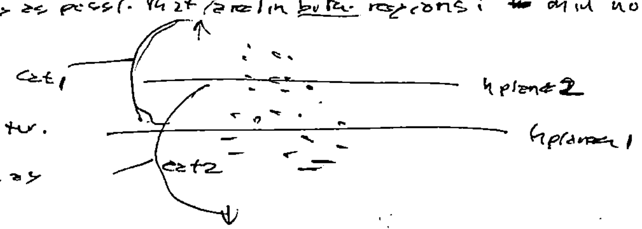
.01 : 86.41 : To do the summation of 86.05: The condition $i \leq j \leq m \ldots$ is easy to get in nested "DO" loops.

The values of a_{ijm} can be stored serially (linearly) in a 1 dim. array: To get the next value of a, (after incrementing i j or m) we just increment a's 1 dim index.

If a is 0 or 1, we can fill the 1 dim a array by generating 64 bit random nos. & shifting the bits into the array — (or other means).

.07 For SVMs w. linear separation: The problem is to find 2 hyperplanes

→ ① All cases in Cat 1 are on same side of h plane 1 and classified as ①
② " " " Cat 2 " " " " " " 2 " " " " " ②
③ The 2 H planes are ∥;
④ There are as few cases as possl. that are in both regions: (classified) min no. of cases betw the h. planes:
⑤ If there are no cases betw. the H planes, we want them as far apart as possl.



.17

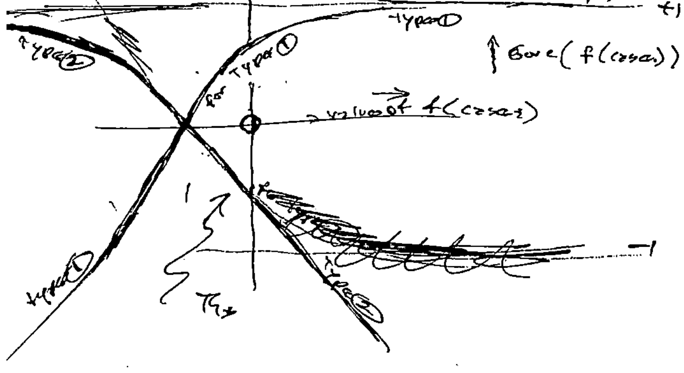
.0 For non-linear separation; perhaps include $i, j, m \cdots = -1$ i.e. one neg exponent.
Note: If there are no neg exponents, some of coeffs a_{ijm} have to be < 0, if we want the points φ to separate the 2 classes.

In general, the Separation Criteria in SVM are Not the Best Crit. not probabilistic).
Using a more realistic/probabilistic Gorc may give nice, continuous Gorc!
Perhaps if one sticks to quadratic functions, the no. of local maxima in the optzn, will be small. .262 (one?) also — or perhaps a random initial pt. & optzn → local peak.

.26 So one repeats .263 many times to get a good one.

0 : 87.01 : A Good Gorc (to MAXIMIZE) w. Continuous Derivatives! Instead of the Gorc of ≈ SVM of .07 - .17;

A Nice Gorc w Continuous Derivatives



The idea is that for type ① cases, much motivation to modify to to make f(case) , if f(case) is < 0: Hvr it saturates at $G_1(f(\ )) = +1$
for type ② cases $G_2(f(\ )) = -G_1(f(\ ))$ so we have saturation at $G = -1$.

$G(x) =$ There should be a cheap way to compute $G(x)$!

$G(x) \cong \frac{x\,e^{-x}}{1+e^{-x}} + 1$ for large x it is ≈ 1
for large neg x it is ≈ x + 1

$= \frac{(x+1)e^{-x} + 1}{e^{-x} + 1} = \frac{(x+1) + e^{x}}{\ldots} = \ldots$

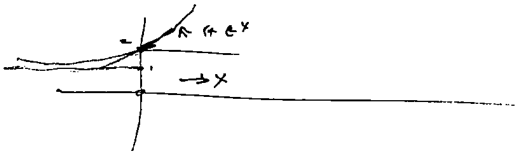
5.1995
STM

50k 3K/mm

2hrs.

$\frac{x}{2+x}$ $\frac{1}{1+2.7}$ ~.3
.3
$\frac{.5}{1+e^{.5}}$ → .188

30 : $G(x) \approx \frac{1+e^x+x}{1+e^x} = \boxed{1+\frac{x}{1+e^x}}$ look at $\frac{x}{1+e^x}$



$\frac{x}{1+e^x}$ has a peak at $x>1$?!

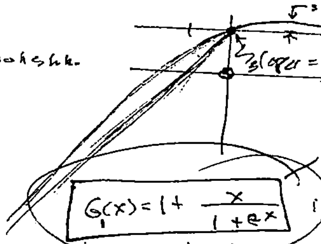
$\ln x - \ln(1+e^x)$

$\frac{1}{x} - \frac{e^x}{1+e^x} = 0$ $xe^x = 1+e^x$ $x = 1+e^{-x}$

x	0	.1	.2	∞
$\frac{x}{1+e^x}$	0	$\frac{.1}{1+1+.1}$	$\frac{.2}{2+.2}$	0

1.37
1.266
1.289
1.277
1.278

So it looks like:



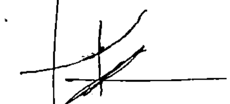
max, slope = ½ ; ~.278

$G_2(x) = -G_1(x)$.

09 $\boxed{G_1(x) = 1+\frac{x}{1+e^x}}$ is not bad but it has a slight bump. Probably not of much significance. It might be cheaper to use $\frac{x}{1+2^x}$ because there is a very rapid approx for 2^x & (Rarely for $\log_2$ ... but is there for 2^x?)

Q's about "degenerate solns" use $G_{1,2}$: The function f can easily expand, contract & more more.

(x+1)



40 ! $G(x) = 1+\frac{1+x}{1+e^x}$ looks better! — No! $\frac{1+x}{1+e^x}$ has no max value

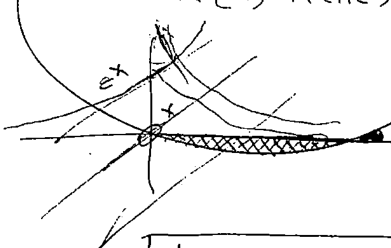
$\ln(1+x) - \ln(1+e^x)$ → $\frac{1}{1+x} - \frac{e^x}{1+e^x} = 0$ $\frac{1}{1+x} = \frac{1}{1+e^{-x}}$ ~~so x = e^{-x} which has no real soln.~~

$x - 1 = e^{-x}$

$x = e^{-x}$ does have a soln., so there is a peak!

$x \approx .5671$ 4 / 367 / 47

$\frac{1+x}{1+e^x}$ → 0 as $x \to +\infty$ whoops it = $\frac{1}{2}$ at $x=0$!
→ $-x$ as $x \to -\infty$ yet $x = 0$ at ...



So a peak at $x \approx$ .56714 (is this Euler's const? ... unlikely)

$\frac{1+x}{1+e^x} = x$ at the peak which is .567 > f(∞) so it's worse than .09

However, I should check both of them again.

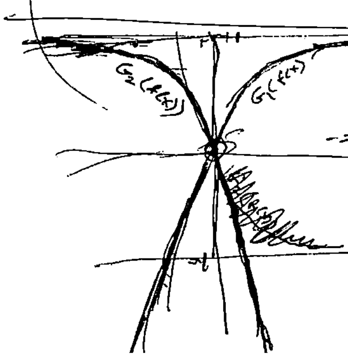
$.278 \times 2 = .556$

which of peak jump at only .278 just about ½ of .56714

$\boxed{\text{Latest try is } G_1(x) = 1-e^{-f(x)} \;;\; G_2(x) = 1-e^{f(x)}}$

As is this binary classifn scheme would seem to be feasible, programmable;
30 I'd have to ① find a good problem to compare w. SVM.
② Get a good nonlinear optimizer; The one in "Num Rec." may be adequate
I'm not sure.

33



There is ~~extreme punishment~~ exponential punishment for very bad cands! This seems to be main complaint about this Gen'l.

$G_1 = 1-e^{-f(x)}$: $G_2 = 1-e^{+f(x)}$

5-14-05
SJM

$\boxed{1 - \ln(1+e^{-x})}$ ≢ is such a function — But on 92.10 — 93.10 I end up not needing it! (desired) — but

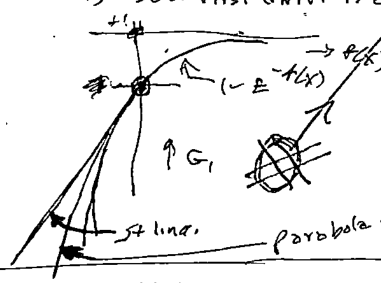
10 ; For cases in which the same vector(s) (are) both class 1 & class 2, the best place TM could put them is at $f=0$

$1 - e^{-f} + 1 - e^{+f}$ is min at $f=0$ $e^{x}+e^{-x} = 2\cosh x$ = min at $x=0$

I still haven't related the forgy classify scheme to PC!

Say we had a perfect classify: How much "info work" would be needed? .150

AH! A way to fix the G curve of $G_1 = 1 - e^{-f(x)}$: patch on a st. line tangent to the curve, so the first deriv. is continuous (But the second deriv. would not be!) ← This may be okay for certain N.L. optzn algm.



AH! Match curve to second deriv. This will give a parabola — which ↓ much slower than $-e^{-f(x)}$ (!).

Will this make it much more time consuming to calculate the zeroth, first, & second Derivatives?

For N.L. optzn., use full matrix method because all second derivs are available

→ Since we have the (full) Hessian & (matrices) get them because we have to invert the matrices — so getting their determinants (is) but not much extra work

So we can get Hessians for "flat min such" — but maybe be sure I have a probabilistic gore vector using the Hessian. → See 92.10 ff for a better way to understand this stuff in a v.g. probabilistic way →

.17

Partial derivatives: $G = G(f(\vec{x}))$. $\frac{\partial G}{\partial x_i} = \frac{dG}{df}\Big|_{f(\vec{x})} \cdot \frac{\partial f}{\partial x_i}\Big|_{\vec{x}}$

20 $\frac{\partial^2 G}{\partial x_j \partial x_i} = \frac{\partial}{\partial x_j}\left(\frac{dG}{df}\Big|_{f(\vec{x})}\right) \cdot \frac{\partial f}{\partial x_i}\Big|_{\vec{x}} + \frac{dG}{df}\Big|_{f(\vec{x})} \cdot \frac{\partial^2 f}{\partial x_j \partial x_i}\Big|_{\vec{x}}$.

$= \frac{\partial f}{\partial x_j}\Big|_{\vec{x}} \cdot \frac{d^2 G}{df^2}\Big|_{f(\vec{x})} \cdot \frac{\partial f}{\partial x_i}\Big|_{\vec{x}} + \text{" "}$

.25 $\boxed{\frac{\partial^2 G}{\partial x_i \partial x_j} = \frac{\partial f}{\partial x_i}\Big|_{\vec{x}} \cdot \frac{\partial f}{\partial x_j}\Big|_{\vec{x}} \cdot \frac{d^2 G}{df^2}\Big|_{f(\vec{x})} + \frac{dG}{df}\Big|_{f(\vec{x})} \cdot \frac{\partial^2 f}{\partial x_i \partial x_j}\Big|_{\vec{x}}}$ ← I'm not at all sure of this formula!

If I use functs that have no derivs > 2 I could check this numerically, exactly.

$\boxed{G(f) = f + f^2}$, $\boxed{f(x,y) = 1 + x + y + x^2 + y^2 + xy}$ at $x=y=0$; $f=1$. $G = 1+1 = 2$

$\frac{dG}{df} = 1 + 2f$; $\frac{d^2G}{df^2} = 2$

50 $x \uparrow 1$; $y = 0$ $\frac{\Delta G}{\Delta f} = 1+6 = 7$ $\Delta f = 1+1+1 = 3$

$G(x,y) = f + f^2 = 1 + G...$ $\Delta G = 3 \times 2 = 6$; Also $\Delta G = \Delta f + \Delta f^2 = 2 + 4 = 6$ so it checks.

Say $x = 1$, $y = 1$; $\frac{\partial f}{\partial x} = 1 + 2 = 3 = \frac{\partial f}{\partial y} = 3$ $\frac{\partial^2 f}{\partial y \partial y} = 1$

Because $\Delta x = 1 \to x = 1$, $y = 0$; $f \to 1+1+1 = 3$; $G = 3 + 3^2 = 12$

1+1 $\frac{\Delta G}{\Delta x} = \frac{12-2}{1} = 10$ $\frac{dG}{df}\Big|_{f(\vec{x})=1, f=1} = 1 + 2f = 1 + 2 \cdot 1 = 3$ $\frac{\partial f}{\partial x} = 1 + 2x = 1$ so $\frac{\partial G}{\partial x_i} = \frac{dG}{df}\Big|_{f(\vec{x})} \cdot \frac{\partial f}{\partial x_i}\Big|_{\vec{x}}$

$\Delta f = 3 - 1 = 2$ $3 = 3 \cdot 1$

Try making the x, y very small = ε, so (for) $x = \epsilon$, $y = 0$: $f = 1 + \epsilon + \epsilon^2$. $G_{\epsilon=0} = 1+1 = 2$

$G = 1 + \epsilon + \epsilon^2 + (1 + \epsilon + \epsilon^2)^2$

$= 1 + \epsilon + 1 + 2\epsilon = 2 + 3\epsilon$

$\frac{\partial G}{\partial x} = 3$ $\therefore \Delta G = 3\epsilon$ $(1+\epsilon+\epsilon^2)^2 = 1 + \epsilon^2 + ...$

5.16.05
STM

GOOD!!

91

→ A New Genl. soln. of [S. Predn for strings], .33ff Also note .30 may be "developable"

1 for $x=y=\epsilon$: $f = 1+\epsilon+\epsilon+\epsilon^2+\epsilon^2+\epsilon^2 = 1+2\epsilon+3\epsilon^2$.

$G = f+f^2 = 1+2\epsilon+3\epsilon^2 + (1+2\epsilon+3\epsilon^2)^2 \to (1+4\epsilon^2 + 2\cdot3\epsilon^2 + 4\epsilon +)$

$2+6\epsilon+13\epsilon^2$ $\qquad$ $1+10\epsilon^2+4\epsilon$

I have [redacted] distinguish betw x & y, — just call them x & y for remainder of present?! → See good analysis (92.10)

: **Expo:** Start Lecture 1 by Explaining how Just about all Lang pgms thus far have been Weak: Thus t. Lang. used has not been "Universal":

Discuss various kinds of ~~...~~ claims to Universality:

1) LZ compressn (Automata bugs)

3) Feed forward Neural nets: Can aproximate any ~~function~~ almost continuous function. Arbs tangle of data can be very big.

[redacted]

2) Fourier ~~...~~ expansion, Taylor expension : needs many data pts : Each method recognizes (orth set of funcs) certain forms easily.

Fourier recognizes sine waves; Taylor recognizes Integral powers.
orth (sets) funcs recognize funcs ~ to orth set : ~~...~~

4) Any Formalism for predictive functions that has no "halting problem" ~~...~~ cannot ~~have a complete set of~~ be complete — it cannot include partial recur. functs.
One way to detect absence of Pr funcs. P.r. funcs ~~would~~ will occasionally run out of ~~compt~~ time and/or memory. If your system does not have this problem — it is not using p.r. funcs & it's incomplete.
On t. other hand, if your pgm occasionally runs out of time or Memory, This is not nearly a ~~...~~ proof that it's using p.r. funcs...

: [SN] On string output S funcs (Big Problem)! One way is to have $\vec{O}$ be a general set of pgms/strings to create from Q's. That for each Q we want to find shortish functions of Q that give t. A def. $\vec{O}$ operates on Q to "simplify" or "condition" it. — Then R (extra bits) gives A.

33 * ≈ SUMS (Ezard) could do t. job of ~~p~~ string prediction in fol. way(s)

*① Using Binary string corpus, t. pc. of next bit is given by funcs. that look at past & classify past as predicting 0 or 1. Each function gets score, depending on how many right & wrong categories it got, wtd by its apr pc (≈ pgm. length). Since each fct has pc>0, all predns of sequences have pc's > 0.

② We can ~~be~~ use radix > 1 (say 40 chars: 26 alphabets, 10 nos, space, period maybe comma, hyphen.)

5. 17. 05

5 TM

00 : We can do ?? by functions on past that have 40 character output & score them same way as binary is scored. For short corpi & large potential alphabet: we may want to use alphabet of only symbols that have occurd upto now, & also an escape pc, for new chars that have never occurd befo.

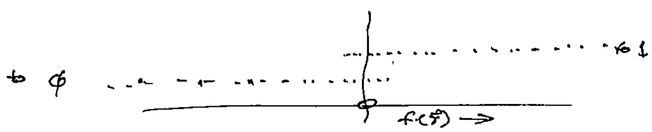
One nice thing about the formulation is that this s-predn. is an easy genzn of d-prediction!

The formulation is described in terms of sequential predn, it would work as well for unordered ?? & the usual QA. pred.

0 (79.40, 90.17, 91.03) >: Going back to the binary classifn problem for real vector input: I really want a more probablistic approach.

If categories are sharp edged, the params of predictive functions give discontinuous changes in classifn errors. For very large corpi: apparently not such a problem — but still, second derivatives can be very noisy!

Perhaps a better approach; We have this corpus of k dim vectors, we classify the data for each — to genz. to new k vectors. $[\vec{x}_i]$ is the (corpus data set). If we have functions that map $\vec{x}_i$ onto $-\infty \cdots +\infty$ ($\equiv f(\vec{x})$) & we want to devise a (symmetric) functions that assign a pc to a $f(\vec{x})$ value, w.r.t. its being mapped to ϕ vs. to 1. Say we got an f( ) like so: — that is almost separates 2 classes — but some overlap.



I'd like a function of f that gives the pc of an $\vec{x}$ having that $f(\vec{x})$ value being 1. [$\frac{1}{1+2^f}$ is one case: see (93.00R) its ... ± (tanh f)]

Presumably the 2 pc's would sum to 1. We could use a Gaussian d.f. — decide on its μ & σ^2, we could use $\mu = 0$, & $\sigma^2 = 1$ then try to find an $f(\vec{x})$ that gives best apsif. → See 93, about $\mu=0$, $\sigma=1$. Well, if it did get a function that separated the 2 classes, then we would want to modify the funct so the gaussian in the "middle" was arbitrarily small w.r.t. the distance betw the 2 classes' "support points", so if complete separation occurs, we get no pc's — or pc's = 1 only — but function itself may be costly.

If there is no complete separation (i.e. overlap exists) then we have to "scale" f( ) or chng μ & σ to optimize the predns.

Hvr, often, we can reduce the no. of params of f, until separation is "no longer possl" (or we have not been able to separate w. the given CB).

On, there may be cases in which a given vector has a classifn of 0 sometimes & 1 other times (for the same vector) — so separation must be impossl.

If complete separation occurs, we can measure its pc. Other functions w. fewer parameters may give a greater pc to the data, hvr. So we take a wtd mean of the predictions given by all possl. f( ) functions — so we always end up w. pc's < 1. ... Good(!)

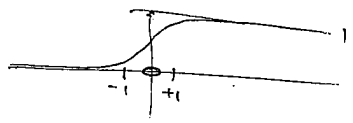
~~BBN's can be [illegible]~~ → 94.40

STM

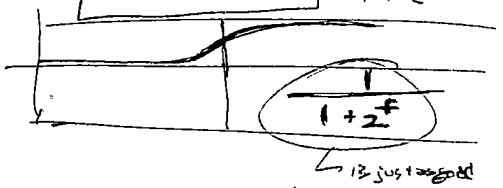
maybe easier to compute — e^x is easy to approximate fast.

: say $G_0(f) \equiv \int_{-\infty}^{f} e^{-\frac{x^2}{2}} dx$.

$G_1(f) \equiv 1 - G_1(f)$.



One use $\frac{1}{2}(\tanh f + 1) = \frac{1}{1+e^{2f}}$



is just as good in present case.

we want $i \to f(\vec{x}) \to \prod_{k_i \text{ class}=0} G_0(f(\vec{x}_i)) \cdot \prod_{k_2 \text{ class}=1} G_1(f(\vec{x}_i)) = \max$.

So, with discussion at 92.10 ff, this would seem to work fine! — also note that G is still in present case. its derivatives are continuous.

In some cases, we have large SSE's complete separation; we use n params of f. If we used $n-1$ params we could get separation. Here, $n-1$ params gives a significantly lower pc for f. corpus, so while its pc's for ~~many~~ cases are not all 1, they get much less wt. (tho not ϕ wt.) than f that f. funct. that actually separated the corpus.

92.22 : About the fixed dim, $\sigma = 1$ model. (NOT BAD!) What we are trying to do is get an $f(\vec{x})$ that combines the Gaussians, gives good pc's for the corpus. Also, in the case in which complete sep. does not occur, we get a Hessian in the params that define f, & f. determinant gives us a measure of the wt. of the soln, (G) — so we know how much wt. to use in combining it w. other solns. For n params, we can try for several solns. & use their predns in ||, using associated ϕ wts. good!

: Actually, getting P.D. for 0 or 1 as the next bit in a binary seq. looks like a bounded T.S. predn. problem for ALP. We can use many d predictors in || (≡ XLP) or use fewer sophisticated ones in ||. Tho I'm still very uncertain about this last! One way is simply to try to get a numerical output betw. 0 & 1 (as a prod of expc of Σ), & we could use a no more output from $-\infty$ to $+\infty$, then squash it down betw. 0 & 1 — T. idea being, that any function then becomes logist. I guess Lisp can express an arby string to real function.

Is there a close relation betw .24 - .25 and 92.10 - 93.10? T. "squashing function" that brings $-\infty$ to $+\infty$ down to (0,1) is .00 - .03 ! $G_0(f)$ & $G_1(f) = 1 - G_0(f)$. Perhaps .24 - .25 is a small (if not all) Gemzu of 92.10 - 93.10 ! i.e. T. squashing funct. can be somewhat arby.

(SM) A Big difference betw. .24 - .25 & PPM (Prog to do binary predn), is that PPM requires a very large corpus & enormous storage facilities for T. regys found.

HMM: We could use SVM to do string → string predn! Use string to read functions in ~~Algol~~ ≈ Lisp. Use support vectors to decide if 0 or 1 is next symbol — or use 92.10 - 93.10 for classifn. of next symbol — or T. gauzen. of 93.24 - .25

In using SVM, as we add data for T. time series, say, we keep the same hyperplane space & assoc. functions, but we slightly move the separating hyperplanes. To get probabilities, we will need a way to deal w. "imperfect in complete separation" — Its likely that there is much SUM Literature on ways to do this; (92.10 - 93.10) could work also — Maybe better, maybe worse

5.18.05 Expo .03, 04, 05 CURE CANCER: .04; .09ff 94

5TM

0: : IN SVM, large distance betw Support Vectors means we need less info to specify t. Separating Hplanes.

[SN] CIAP → **UCP** = Unintended Consequences problem.

.03 Expo ① "Why I am not now working on Rental. Machines". Short paper.

.04 → ② How to express "Cure Cancer" as a "well defined problem". (related to ①)

.05 ③ The Science in t. "Science in Star Trek" (actually t. epistemology "Star Trek science") Explains that that book assumed our present (narrow) theories of physics ... Unlikely to be true "forever".

.09 Actually ② (.04) is not yet in t. shape for "expo"! I'm not so sure my "soln" is — of t. "Cure Cancer" problem — adequate — How do you explain to TM what an "adequate approxn" is? Much of t. reason why is Monte scientific probabilities.

One way would be to ask TM to devise a decision rule (for what to do at t. outcome of each experiment) so that t. proby of get a what's defined to be "cure" (w.in 10 yrs, say) is Maximized. I.e. TM can view this as a "ordinary" (But very difft) OZ problem.

A way to "fill in" t. various sub-problems — Have a human do t. planning of t. rsch & see just what is being done & how it could be optimized.

A big problem is that t. results of experiments can be many many difnt possys, So we have to find good ways to categorize them & use t. categories to decide on "what to do next".

0

Back to ~~string → num~~ string → String functions; ⑤ Fourhouse Strings the recent tricks (like SVM's) have been like using ANN for digital logic — like t. walking dog — surprising that it works at all — never mind that it doesn't work well!

→ Try using methods humans used (Bernoulli seq. Counting pc's Logs "Dixies rule") & try to genz. them.

But do write Review of recent ⑤ funcs w. string output & rel. relations to ANN (& SVM is also close to ANN). — As linear separability. Perceptrons — at least.

.09 [SN] Reading Stuff on kernels, Rad basis funcs, SVMs in "may be feed forward ANN, & fuzzy Logic" "Classify" "Pattern" (Duda, Hart, Stork) book; Try to unify this all w. my own version of SVM's: Can I write a grammar for All of these? — Showing just how they differ & suggesting new additions to "T. language".

0 Do BBN's fit into this?

Ordinary ff. ANNs (feedforward) w. 1 output can (perhaps) be regarded as a realization of 93.40 – 93.10! Also, as a "linear hyperplane separation" system, but if we use tanh x as t. "squashing funct.", we can translate it to 0, 1 (by $\frac{(\tanh x + 1)}{2}$) and get pc's! If we don't confine ourselves to feed forward configs, we can get loops & oscillation (≡ "non-convergence") may to approach Turing Universality!

BBN's can perhaps be regarded as a type of FFANN!

Jan 22.05

95

STM

: First I want a clear understanding of "kernel" as used in "Patt. classifn"
It is used in a general way to describe many "Patt disc" methods

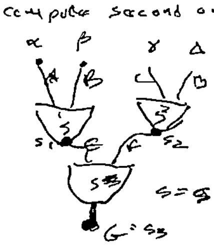
[SN] CMOS Logic is a kind of squashing function : (Also TTL logic!) — But in both cases the "Analog" (i.e. ≠ 0, 1) region is too small.

T. Sorge (94.28ff) suggests that I might use RANN or Cris Moores (Shannons) "Universal Analog Computer" (these last can be simulated in Digital to experiment w. them.) For RANN : If we have a fixed set of "Nodes". Each node can have several inputs, & its single output can go to many other inputs. So say we decide on a 100 wt. machine & we decide on which half the nodes is the output, & which nodes are input. (We could have special input, output nodes that do no summation or squashing. Anyway, we ① have to connect the 100 wts betw. the nodes node pts (at input & at output) & we have to assign wts to the nodes.

This sounds like too big a problem to do ab initio for each new QA to lrn. — or for each little improvement of "TM₂" — or just for "Metainduction" in Phase I.

In usual RANN, the connexions are kept fixed & wts are "updated" wrt. performance on (at least) I/O pair. Simple first order partial derivs are used: $\frac{\partial \Delta G}{\partial w_{ij}}$ ← w.r.t. j . Second partials not used. (too many wts : the 100 × 100 matrix is solvable).

How to compute second order derives?



S = squashing funct.

$\frac{\partial G}{\partial E} = \frac{dS}{dx}\Big|_{x=E+F} = S'\big|_{E+F}$ $\qquad \frac{\partial^2 S(E+F)}{\partial^2 E} = S''\big|_{E+F}$

$\frac{\partial^2 G}{\partial E^2} = \frac{d^2S}{dx^2}\Big|_{E+F}$.

N.B. [I'm not at all certain of this formula!]

$\frac{\partial^2 G}{\partial E \partial F}$ = (See 90.25) = $S_1 S_2 \cdot S''\big|_{E+F} + S' \cdot \phi$ $\qquad \frac{\partial ES_1}{\partial E} = S_1$

at E+F, at C+D

$= S_1 S_2 \cdot S''$

Actually I'm not really about it at
rate: The input to third node is $S_1 \cdot E + S_2 F$. so viz.

$\frac{\partial G}{\partial E} = S_1$, $\frac{\partial G}{\partial F} = S_2$.

$\frac{\partial^2 G}{\partial E^2} = S_1^2 S''$; $\frac{\partial^2 G}{\partial E \partial F} = S_1 S_2 S''$. | $\frac{\partial G}{\partial A} = \alpha \cdot S_1' \cdot E \cdot S_3'$

D − C − (B − A)
D − C − B + A
D − B − (C − A)
D − B − C + A
so ∂x∂y = ∂y∂x.

at any rate, the

00 (space) (59.03) ? That 2 dim convolution would not be exactly what I want, but may be close enuf to be give fairly good contexts. A problem w. it is that it doesn't give what are necessarily "properly connected", subgraphs. It may be that a 1 dim convolution would do as well. The kernal would not nec. have adjacent nodes on it ... it'd could have large gaps.

03

Note that a tree can be generated in (at least) 2 ways: Bottom up or top down. Bottom-up was used in T. Dent of AZ lang. in Appendix of IDSIA report.

R girl ... Ferrariz (psl) described a method of doing GA in which the notation

10 describing the graph of (0 3 L) was 1, 2, 3, 4, 5, 6 ... 15 !! She used normal ("GA" rather than "GP") crossover. She said she got unusually good results !!

The string 2,3 would have for "context" 4567" on one side & "1" on the other. ← useful

1, 2 would have context 3 ← useful.

2 would have context 1 on one side (useful), 3 on the other somewhat useful

3, 4, 5 ... on that other side ... not so useful.

5 would have 4 as (useful) context 6, 7 as not so useful contexts.

T. notation Ferrariz uses is not clear in her paper, but it is (other papers that are) developments, extensions, of her work.

20

SN

A Maximum Entropy Approach to Sampling in EDA — The Single Connected Case

Ocho, Hons, Soto, Mühlenbein

Ps to old ✗ (Ciarp 2003) PS ← why this title? → 9/12/03

This seems to discuss problem of finding proper context trees, ect. — But I'm not sure: 8 pp. — try to read it. They discuss (binary) Bayesian nets, & the relevant trees. Chng title to "Tree context discy 2003" — on HP467

Physical Realizations of strong A.I.: Look for various "technologies" that achieve Turing universality. Find one in which Macros are easy to detect — i.e. easy to extract commonly occurring Contexts. Matches need not be exactly correct. It should be easy to verify that a particular "Technology" is universal w. this property. Lots of technologies are universal, but even it's hard to prove they are universal. Ones w. the desired property → should be a lot easier to show they are universal.

30

One approx Method for "Context matching" is "discrete convolution" — using analog of FFT. Cor. can be derived). Finding FFT of string plus new (Concat) string s' should be possible to do in an incremental way, since FFT is linear! — Not meaning of "convolution" is diff.

.0 : On ANN: If it is a kind of trial, one can randomly move connections from one node to a "near by" node: Would it not eventually configure itself so that this would only be a good way to do trial (ray?)

In t. function country method & t. method of Generating trials in t: AZ lang. of IDSIA report Appendix: Each new function was allowed to choose any previous nodes or primitives "below" it. The notation is polish n (RPN) But they things referred to are different. Devise a notation for it and look at kinds of useful regys. They may be much different from t. sub. trees I've been worried about. Hvr., it does have "de functions" in it, which makes t. meaning of "context" a little diffrnt. !

.0 BTW, t PPM system (or modifns to allow subtree contexts) might be used to find useful definitions: These would be either proper contexts or "n-tuples" (That include t. thing to be predicted). Defining of an n tup. is desirable if t. parsing of t. data using t. defines is almost always better than any other parse.

.0 ≈ **Expo**: T. methods for PhI improvement is essentially [Super Boosting]: It is better than ordinary Boosting in that we are oriented toward a Turing Universal predictor. (Note 94.28)

Ordinary Boosting I picks t. best of several PEMs or does a linear comb of them.
— (Or tries to get system to work harder on QA's that don't work well) ← ??

It is of interest that very many different present day, already developed systems for induction
26 can be used for TM_2 ! Feed Forward ANN for single bit prediction, PPM w. or w.o. various enhancements (like subtree contexts)., SVM (for predicting next bit.... but how to get pc ?.... see (93.32 or 97.30
RANN or Universal Analog Machines ← (94.04) ;|| (93.24-25 is a sort of General way to get (S) string predn
Decision tree can be used perhaps; BBNS; —— Very many old methods in "Pattern classifn" Duda Hart, Stork → 98.00

29 [SN] I think the way to get pc's out of SVM's was to ↓ dim. of space until we couldn't get complete separation to N dims. Then using N-1 dims we don't get separation but a d.f. over 0,1 as predns. N-1 Dims. Adding 1 dim to get separation has pc = "escape cost" and we add this pc to t. pc of t. correct predn. via t. N dim. separator. Also t. escape cost is hyer if t. distance betw. separating planes is small. SMALL

[SN] If t. Normalized Pc's of [0 à 1] are close to "correct", then t. ratio of them (unnormed) must be close to correct.
— Which is why I was interested in t. normed values — whether or not I used them directly.

20: (EPGC 97.29): I can refer to the direct QA induction as "primary induction": it uses Turing machines. The "secondary induction" (=TM2) need not be Turing-universal (but can be). It can use the variety of induction techniques mentioned in 97.26-.29; Eventually we can use primary induction to do easy induction — Any improvement in this will have to be via Phase 2.

For Phase 2: The Grammar of Inductive Systems" can be obtained by studying the systems in "Patt Classfn" & "A.I. a Mod Approach". In "Pattern Classifn" the systems are (AI Perils) for induction only (o.h. to "boost" easy induction of Phase 1), "AI a Mod Approch" discusses other kinds of problems as well. — OZ probs: "Patt Classfn" also discusses OZ probs soln methods of various kinds.

What I'd like to do is get a (large) corpus for pg. I. — possibly use some "non-program" kinds of problems. Then see which induction systems can compress most, "per unit time" (& exactly how have not been found related to eg. of ~4TM 1,2,4 1,2,4 ...).

: A major problem will, of course, be Initial TSQ. It will be easier on the trainer if the machine is very "fast" (by Epc), since the examples don't have to be as close together (perhaps) don't have to be sent out so carefully. For the early TSQ, TM2 criteria to be very "smart", because it has such a small corpus.

30 → At this point, I really need to write a good Review of just how the Devices in "Patt. Classfn" & "AI a Mod Appr" are to be applied to Easy Induction. → See 97.26-.29 for a list.

5.26.05

STM

Understanding Optimization +.30

:00 : D Duda, H, Stork : p 319 — 324 : § 6.9.2 & 6.9.3 : & 6.9.4

6.9.2 gives Newton's method, but it involves inverting the Hessian matrix & is ~ N^3 problem — too long.

6.9.4 is "Conj. Grad descent" which gets same soln as Newton's method, but only involves N jumps.

[presumably, each jump takes time $< N^2$!] & gets same result as Newton's method.

As I search, this must be equiv to Inverting the Hessian in $< N^3$ — so it is a way to invert symmetric matrices (perhaps positive definite) in $< N^3$ time!

Also, If the results are the same; ① why does one value of "β" give different results than Another? — "One deals w. higher order non-linearity's Better"!

:09 If Conj Grad is an approx soln to inverting a matrix; It can be used to get a fast, successive approximation to the Inv. of the Matrix.

When the matrix has no inverse (singularity, this means at least one eval = 0)

Does the "Conj. Grad Desc" get pseudo inverse?

(Why is the pseudo inverse of a symmetric matrix w. one or more zero evals = ∞ evals.)

Well; the Pinverse inverses all evals but the zeros — which it leaves invariant. — easy to see this in the coord system in which the original (symm) matrix is diagonalized (is always possible to get to this coord system by Rotating a symm matrix).

:20 : SN on "BOOSTING": This is an induction method that uses several (or many) PCMs for prediction. When given a corpus of many cases, it tries to find a way to combine the PCM's to work better on the cases in which the system had worked most poorly in the past.

This sort of technique would seem to be very useful for a QA corpus! So "check it out" more carefully.... See if it can be adapted to a more Universal Induction System than are normally used in "Boosting".

"AdaBoost" makes very strong claim for "Boosting" on p 665 || Also see "Pattern Classification" pp 476-480 on Boosting.

:30 : Re: "Understanding Optimization": Knowing about local Minima & how to deal w. them is one possible direction of "Understanding". Various optimization techniques in Simulated Annealing & GA-GP have ways to deal w. this. In ANN there are "Accelerators" like "Momentum" Newton's Method (in N dim) is also a way, but it is that Con. Grad "method" of jugate (.09)

Also, fitting good Hessian (Good statistical fit), than using Newton's Method.

→ It might be possible to get V.G. induction in Ph.I w.o. going to Ph.II!

5TM

{On Ordering A_i's in pc order — .30.
{On ~~...~~ Langs for generating BAGS — .20 } ← Classic problem: Engr.: Many Solns

.10 : 99.20 & 99.30 are both very encouraging! Suggesting that I can get a very good Phase 1 ~~w.o.~~ Phase 2, using Existing Trad techniques!

0 SN on convergence of Newton's Method! Use statistical (N.M.) method, but have size of "Box of data acceptance" be k. Large k ≈ hi temp. in Sim. Annealing. —
Start on hi k
Start on hi k & slowly " " it down to smaller k values.
Start w. hi k

0 : On s-langs: we can have a fixed or growing no. of symbols (includes defined symbols

.21 ≡ "tokens"). ~~The termination~~ One of them is the termination symbol, (Δ, say).
T. pc of Δ will be constant in simple langs, but more generally it can be any function of t. previous symbols. This gives a nice way to deal w. Big corp:
If t. prob of Δ = 0 for all contexts, we have t. "continuous" (univ. perhaps) d.f.
A lang can, using this convention, have some finite strings in its corpus & some infinite strings.

.10 On putting outputs of a stoch generator of symbols (including Δ (.21)), in pc order.
In generating strings (sequencely), keep them in bins of width ε — ε can be m ... This ε is ... in mto so ... logs (tm length),
We first generate all strings of length 1 symbol, & put them in Bins according to
.34 length. (α) We expand all strings in t. shortest bin, by 1 symbol & ~~...~~ transfer them to new bins (usually not t same as previous lowest), loop to (α)(.34). As soon as one string terminates, we have an upper bound on length of shortest string. We may want to no longer keep bins longer than that. We will periodically find t. shortest terminating string thus far. When t. shortest bin = length of shortest known terminating string we stop. → (spec 101.00)

T. "Bins" can be "linked lists" — we take from t. whole list when we "develop" its strings.

5.28.05

100 pts

STM partial rec. V.S. prim recursive funct?.

Normally a For loop could never give an "undefined" output.

Hvr is "$(3 \div 0)$" is one of the functions which have an "undefined" value.

A "for" loop w. large index can approximate an "Until", in folg way. Say the Until condition is $f(x) > 0$. Each loop gets new value for $f_1(x)$ and $f(x)$: when $f(x) > 0$, we Jump out of loop w. $F_1(x)$. To simulate this jumping out of loop

The recursion eqns. have modified $f_1(x)$ & $f(x)$. for each recursion

Hvr, If $f(x) > 0$ sets a flag $= 1$

If flag $= 0$, recursion proceeds normally for each cycle of loop.

If flag $\neq 0$, each cycle of loop does not change $f_1(x)$ or $f(x)$ — so when $f(x) = 0$ we keep $f_1(x)$ constant.

Just before end of loop, we look at the loop index $= i$; if $i = n$, and flag is still zero, then output is "undefined"

Or $f_1(x)$ is initially set at "undefined"

only when the flag is set to 1 does it change to an actual value.

Alternatively: If at the loop simulation, we end up at end of loop w. no acceptable value for $F_1(x)$, we set $F(x) = \frac{1}{0}$ → undefined.

GTM **SUMAC.30**

.00 (Spec) (100.40): Its not obvious that 100.30 ff will always capture t. shortest A.
A possi. way to prove it: Given that abcde is t. shortest A:
"a" must have been in t. shortest bin.
"ab" " " " " " " " "
"abc" " " " " " " " " " etc. } I think these are all true.

So I think 100.30 ff can be used to generate all A in pc order . . . .

.06 Its not a particularly cheap method! If l is length of shortest A, we will have to generate all non-terminating strings of length < l. ← because
Actually .06 would probly have to be true, no matter how we did it! Every non-term string of length < l has t. possy of terminating in length < l. T. no of strings of pc < l will be "like" c^l, where c is related to (say ½ of) t. alphabet size. — But see (.19)

100.30 ff is impt. not only in Generating A's in pc order, but in generating Cands in Lsrch! (→ which probly takes much larger fraction of time.) So it's a very impt. problem — I do want to try t. find some approxt., faster solns. One way would be to complete branching at t. beginning of t. A (A/cand) Generation, then follow only t. more likely characters, as we go on. Say, develop only 10% or 20% most probable (A's / Cands)

.19 (.12) Actually, t. problem may not be as (.12 / .11) make it look! It may well be that as t. strings get longer, their continuations become very narrowly (pc) limited — So there really aren't very many solns. w. pc > say $\frac{1}{1000}$.
Actually, the routine of 100.30 to get strings in pc order, can be done perhaps simpler by Conventional Lsrch.

.24 Hvr, Note that if 100.30 ff is used for Lsrch, and we did T ← 3T (t. fastest way), I think we would ↑ speed by factor of $\sqrt{3}$, since we would never waste cc by "over jumping" the soln. in the last T ← 3T "round". Hvr, to offset this, t. bookkeeping is more complex.) NO! Note (.27)!

.27 Woops! .24 ff would not work for Lsrch — since it is necy to search all w. CB = PT, i have T ↑. Hvr, it mite be useful for ‖ srch (?) ← That it doesn't (ch check!)

.30 SUMAC's! T. essential qualities! After each prodn, t. diversity of t. ll codes does not ↓ (much — or ↑ — its ~~ t same). T. Sumac always retains its Universality.
So w. each new problem, we may chose a subset of pgms that are "up to" previous problem. Also, we have to do search to ↑ diversity lost by deleting pgms. We can do this by augmenting old pgms that didn't work w. new problem, a/o by "backtracking" to augment solns of an Earlier Problem.

Started by saying I would have liked to title the lecture "Path to Strong AI" or "How to find them" — But this gives portmanty
Pathology to Strong A.I.: A repair Manual) on what I will do in the near future and just why I will be doing these things.
Alternatively, it may be viewed as a proposal for Strong A.I.

0: **Expo:** This is ideas for Modification of Lecture on

Diff (2) Absence of TSQ or inadequate TSQ's.
Desc. child being brought up and educated; then he solves problems. How he then discovered.

First. Anew " " " . . . . . problems " " " "

This is how most high performance problem solvers work.
[redacted] have how bro.
3 systems. Multiple areas of work in this direction.

1) Wrappers. (vito Einstein not ...)

2) GA-GP-etc: explain how each new population has a new problem. (Rite but not applied to GP as a whole) — Troll parts got lost

3) Incr. Lrng. .... Relatively little work done on this (Butcheck on it)
TSq's are a kind of "Incr. Lrng." ... but .... how they differ.

[redacted]

(1) [redacted] Inadequate Induction [redacted] System.

a) Inadequate language: Discuss: polynomial (Best ... — Akaike Criterion, MDL,
[redacted]
Fourier Series
ff. Neural nets.
SVM's w. hy dim. Non-linear Map.

Universality: These Langs for expressing funct will discover approximate any desirable function to any ° of accuracy, but to do so requires a very large SSZ.

Using an appropriate Universal lang. (ie. Turing universal) we do approximation w. very small error w. min SSZ. To do this requires ∞ cc.

~~To get close, more~~ If we use < ∞ cc, we can usually get close. This inability is what makes this rule "incomputable". How. we can never know exactly how close! While this may seem like a disadvantage of using strongly universal set of functions for induction — it is not. No other induction system can do this either — furthermore, no other kind of induction system can do as well. ~~I will go into the limitations of induction system~~... Furthermore, in practical applications, we usually don't have to know how close our induction is to the best possible — ~~it is only necessary~~ we usually have to know how well it will usually do in the future — This problem is solvable by cross validation techniques that have nothing to do with the models used — they are essentially independent of the kinds of models used.

Perhaps Discuss (Sample size error vs. model error)

b) Inadequate Srch strat. I did once spot what that was an inadequate srch strat in some early A.I. work (Ex. description of the search technique was clear enough so I could see some) — How. there's been an enormous amount of very good work done on Opt./search/Ing programming. If you have an adequate induction language, it should be possible to find a function that fits your data rather well in a reasonable time using available techniques.

Universal? variety?

1) Def^n of Strong A.I. (eg. what my goal is).

β → 00124

Inadequacies of Current A.I. systems.

1) Induction is impt. part of capabilities of A.I. (Tho it's not All Parts needed — it's major problem)

Inadequate (esp. for induction.

How ANNs are inadequate: ~~ff~~ANN, Fourier expansion, poly expansion, ; GA, GP (usually).

They are "Universal" in the sense that given enough parameters, they can express any function with any desired precision. However they need not be particularly efficient in this — One may need an enormous no. of params & very fine precision to get desired approxn.

There is another kind of universality that's much better at this: i.e. The universality of an ordinary computing machine, or any Turing complete instruction set.

If you try to approx a sine wave with a ~~ff~~ANN you will need a very large no. of nodes and wts to get good approx. ~~At no point in the approximation will the net restrict~~

However, using T. universal approxns: ~~there~~ after a certain point it's seem to be part of same curve, the system is able to recognize this ~~fact~~ and make a very accurate approximation with relatively few parameters.

~~ff~~ANNs are unable to do this. — Anything that is ~~not Turing Universal~~, is not able to do this in all desired cases.

The inadequacy of most "universal systems" is in their need for an enormous no. of data points before they can approximate a function.

The advantage of Turing universality is the "small size"

I want a kind of outline of the talk:

First: Define what my Goals in Strong A.I. are: That if they can be met, the system could probably be trained to ~~do the things the passing the Turing test~~ understand spoken speech and gestures and even do things like pass the Turing test. — However those are aspects of A.I. ~~that I'm not particularly~~ in which I have little interest.

I will list several important deficiencies in current A.I. research. While many (existing) systems are free of many of these criticisms, there is ~~nothing~~ no system that I know of that's free of ~~all~~ of the deficiencies. [In the final lecture I will ~~not~~ describe a system that I believe deals with all of the difficulties in an adequate manner

---

~~One of major problems in A.I. is Machine learning~~ I'm

I will identify the strong A.I. problem with that of Machine learning: Given a corpus of ~~data~~ relevant historical info. — How best to solve a particular ~~problem~~ well-defined problem. How can the machine best bring to bear the info in the corpus to help in the solution of the problem.)

We will begin with pure induction problems & the later show how an understanding of induction can be used to solve many general problems ~~more~~ very effectively

STM

first, the induction problem. In a simple form, we are given a sequence of symbols and we want [struck] the pd. of the next symbol.

Discuss Unobtainability of universality: L.R., ff. ANN, polynomial approx.
List topics I want to make ... in proper order — (or note Missordering!)

Some possl. Attacks:

The bad news is that you really have to [struck] consider a universal D.F. — which is strictly incomputable!

T. good news is that you don't have to actually compute the Univl. D.F. — you can get good predictions and good estimates on how accurate they are w.o. computing the Universal D.F.

More bad news: Most A.I. Researchers don't know how to do this.

More Good news: I will tell you how to do it.

[struck] It is [struck] possible to get very nice exact results by narrowly restricting the class of concepts considered. In a benign Academic Environment, the Instructor will tell the student that he need only consider linear models (or ff neural nets) ANNs with 1 hidden layer or whatever. The bad news is that A real easy more in the unclassified unclassified unclustered "outside world" is not giving any such information. — It's pretty much an [struck] "anything Goes" situation. — He doesn't know how to deal w. this [struck], so he imposes the kind of restrictions his instructor imposed. Not good!

The Good news is that I will tell you how to deal with this problem.

"Well defined class of Academic Concepts."

In predicting t. Stock market: NYSE does have rules about trading stock — but you will not find any rule that says only linear models for price variation are allowed. [struck] It's really "Anything Goes"!

To realize that it is not only the stock market that has this (unruly) (rule-less) quality.... just about anything else in the real world has the same features: There is no real limit on the kinds of laws, regularities, and spontaneities that might occur.

SN A possl. understanding of Kolmog's Theorem about representing N dim. continuous functs by [struck] N or (N+1) continuous functs of 1 variable.

— If a funct is continuous in a region, then assume it has a max + a — derivative over t. region. This gives a simple Lipshitz condition satisfied by t. function. It profoundly limits t. no. of functs possl. on a 1ⁿ cube w. precision Δ. Given t. peak derivative.

We can say how many functs there are of this sort ↑ compared w. peak deriv. = ∞

Say $f(\Delta, d)$ is no. of functions on unit cube of grid size Δ & peak derivative = d.
$f \to \infty$ as $d \to \infty$. Hvr. n-dim functs can have lots of info in them if they are allowed very large d. values!

→ 108.00

6.4.05

STM

.10 : On ANN for digital [Binary (or N-ary conversion Not-pack?)] time Series probn.. If P_i is its outputs, we want

.01 $\prod_i P_i^{x_i}(1-P_i)^{\bar{x}_i} = \max$; $x_i = 0$ or 1 ; $\bar{x}_i = 1 - x_i$ $x_0, x_1 \cdots x_n$ = the time series.

P_i represents pc of the output given by the ANN.

.03 If output is $-\ln P_i$ or $-\ln(1-P_i)$ we want to maximize [$\sum_i$ of outputs] — which is conventional error function — So we may be able to use standard ANN prgms, w. non-standard "output squashing function". So we use Newton's method or Conj. Grad. Method to find wts.

.06 Advantage of This ANN over PPM is that it allows more "complicated" functions;

.07 Disadvantage is its [small memory capy] (≈ ∝ no. of wts in net), long Time to converge ↑ w. no. of wts. in Net. PPM can remember very big, complex contexts.

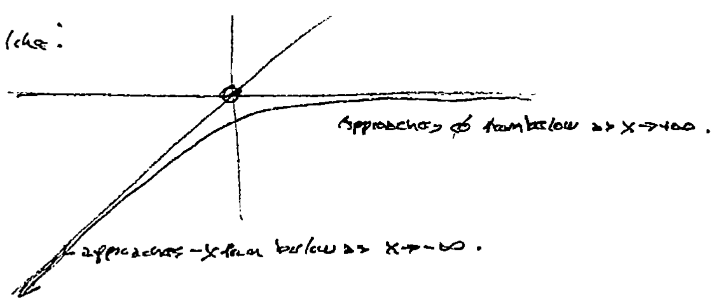
.09 We can use data over & over again — (not necc. sequentially.)

(Unclear in my mind as to Details of This!)

.0 Also, we can use Recurrent ANN. For small no of wts, + config of wts, we have large srch spaces, long srch time. { As Net begins to converge, we can ↑ [in] "This is probably in a loop because its been running for ↑ seconds" }

How do we handle "undefined" decisions in these trials? — its different from a UTM machine, in which all subsequent outputs become "Undefined".

.20 : 90:10 : [SN] a way back [90.10], I wanted a function like:



$-\ln(1+e^{x})$ is such a function:

It is the ln of the squashing function, (.03)

: $\frac{1}{1+e^{x}}$

.30 : .06 – .09 seems impt! ANN can use only a few params (wts). — (The usual way is to use Very many wts & cross validation) But in general, contrast, say linear/n.l. regressn (few params) w. PPM (which stores whole corpus). (In n.l. regressn can store covariance matrix & update it via Xwindo (little Memory Needed) or Rwindo (Much more Mmry needed).

PPM w. Xwindo or Rwindo would ~~[illegible]~~ but has it if it had a finite ("small") corpus, & would have not so good Compressn.

6.6.05
STM

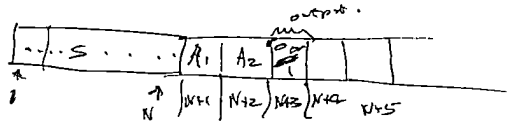
106

T: HALTING Problem

6-6-05
STM

: Proof of unsolvability of Halting problem:

S is a program of N bits. When S is ~~terminated~~, at address 1, and addresses N+1 & N+2 contain start and end of pgm X, then S if started at address 1, will decide if X will eventually halt or never halt, & print 1 or 0 at N+3, then ~~HALT~~



It will go to N+4 for its next inst. (which can be "HALT" — But need not be.)

N+4	N+5
If N+3 = 1 then	HALT
go to N+4 (loop)	
else, go to N+5	
i.e. if N+3 = 0	

Consider a pgm that goes from 1 to N+5

Set $A_1 = 1$, $A_2 = N+5$

Start pgm at 1. Eventually it must print 0 or 1 at N+3, since we asked it about pgm 1 thru N+5

If it prints 0 it has decided the pgm will never halt. . . . but in this case, S pgm continues to N+4 then N+5 & HALTS

" " 1 " " " " will HALT " " " " N+4 and loops to N+4 infinitely

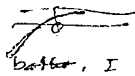
So it looks like the pgm. didn't work right.

6.2.05

STM

Rev:

106: Halting problem: Quick, easy, complete proof

105:.20 Easy way to get  funct., for SS.10 ! But unclear as to whether I want it!

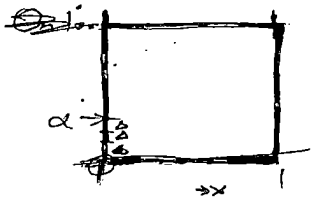
105.01 may be better, I will have to decide on just what shape of fcts for ANN is best.

.00 On use of ~~[illegible]~~ ff ANN (or possibly RANN) for discrete pdn — i.e. p.d. on strings ; s.d.f. on STRINGS

.30 Difference betw. ANN & PPM in kinds of info in Models. Bears more Scrutiny)

104.30: A poss. way to understand Kol's Thm on approx of N dim. functs by 2N, 1 dim. functs, in terms of "MB".

10:104.40: On Hall's thm on approxn. of functions. If there exist finite values of derivatives on approximation region, then we have Lipschitz condition: Consider 2 dim region $0 \le x \le 1$, $0 \le y \le 1$: Grain Size = Δ.

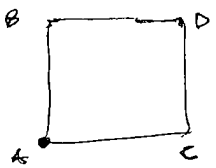


consider edge, α; $x=0$, $0<y<1$: In distance Δ, function can move Δz ("Z" from Lipschitz). So, in vicinity Δz, α can have $\sim 3^{\frac{1}{\Delta}}$ values.

In each Δ distance the function f goes up, down, or stays same. (3 way question)

Therefore row to α has less info, since its change is limited wrt. α, but also wrt. near by Δ's. The min restriction is if

2.05: In Bay area,



B & C are the same: Then D can be $C+0$, or $\pm \Delta z$.

If B & C differ by $2\Delta z$, then D can only be the mean of B & C.

Figure out how many configs on 1×1 region seems difficult!

Say $\delta = \Delta z$.

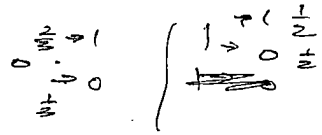
To simplify, consider lattice w. integral spacing and $\delta = 1$.

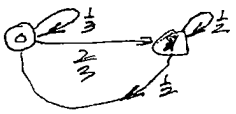
				0, 1	$\pm\frac{1}{2}$ $\pm\frac{1}{2}$ 2
	± 1	1, 1 → $(\pm 1, 0)$	3	-1, 1	0 1
		0, 0 → $0 \pm 1, 0$	3		
	± 1	< -1 $-1 \pm 1, -1$	3		

To state the problem: We have an $N \times N$ integer-valued δ & z integer functions on the $N \times N$ pts: Restrictions on the functions: No pts can be > 1 above or below adjacent pts. How many functions are possible? May depend on edge conds: Say all edges $= 0$.

$0 \to 0, \pm 2$ "1" $\to 3$; $1 \to 0, 1$ $\qquad 0 \to X_{3 \times 3}^{2 \times 2}$

So a messy exam! 2 states: $0, \pm 1$:





So we can get "entropy" of row next to all 0's.

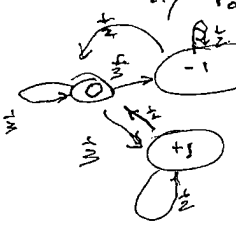
$P_0 = \frac{1}{2} P_1 + \frac{1}{3} P_0$ $\qquad P_1 + P_0 = 1$ $\qquad \frac{2}{3} P_0 = \frac{1}{2} P_1$

$P_1 = \frac{1}{2} P_1 + \frac{2}{3} P_0$ $\qquad \frac{2}{3} P_0 = \frac{1}{2} P_1$

$4 P_0 = 3 P_1$

$P_0 = \frac{4}{7}$, $P_1 = \frac{3}{7}$

Locally $P_0 = \frac{4}{7}$, $P_{-1} = \frac{3}{14}$, $P_{+1} = \frac{3}{14}$. Here a +1 can't follow a -1. !



$P_0 = \frac{1}{3} P_0 + \frac{1}{2} P_{+1} + \frac{1}{2} P_{-1}$

$P_{-1} = \frac{1}{3} P_0 + \frac{1}{2} P_{-1}$

$P_{+1} = \frac{1}{3} P_0 + \frac{1}{2} P_{+1}$

$P_{-1} + P_{+1} + P_0 = 1$ $\qquad \frac{2}{3} P_0 + \frac{2}{3} P_0 + P_0 = 1$

$\frac{1}{2} P_{-1} = \frac{1}{3} P_0$

$\frac{1}{2} P_{+1} = \frac{1}{3} P_0$

$\frac{7}{3} P_0 = 1$ $\quad P_0 = \frac{3}{7}$

$P_{+1} = \frac{2}{3} P_0 = \frac{2}{7}$

$P_{-1} = P_{+1} = \frac{2}{7}$

1st entropy of "1st nearest row" given by transitions $P_0 = \frac{3}{7}$, $P_{\pm 1} = \frac{2}{7}$? Row states are not "indep".

Proxy is more like $S = \sum_{i,j} P_i \ln(P_{ij} | P_i)$

I think this will work!

for f. entropy of a [struck out] [struck out] by current row:

0:10840 To get Entropy of next row. OH, that's it! Consider all possl. local configus that determine t. value of f at a pt on next row. They will have certain pt's: We have a mapping of a set of pt's of configs

03 into itself. For this config. to be "stationary", gives a (solvable) set of linear eqns.

A • → X y
↑ ↖
• •
B C

determine X

$A = +1$
$B = -1$
$C = -2$ is possl.

since it gives [struck out]
$x = 0, y = -1$

So given $A = +1$, $B = -1$, C is constrained by B only.

[struck out] So knowing A, B, we can get D.f. for X.

I still don't have a clear idea as to how t. (discrete) mappings go.

Maybe diagonal strips for states!

A • • X
B • • y
• C

$A, B \to X$; $B, C \to y$ This may be IT!

So if AB has a certain freq.
& BC " " " indept of A
then we get D.f. on X, y.

possl. pairs
$-1 \to -1$
$0 \to 0$
$+1 \to +1$
$-1 \to 0$
$0 \to +1$
$+1 \to 0$
$0 \to -1$

No: $A \to B$ can be $A = -1$ & $B = +1$ or $A = 0$ & $B = \pm 2$

+2
+1
0
-1
-2
so 5 possys.

. . . .
.   .
.   .
. . . .

so 5 jump probs $\begin{matrix} \pm 2 \\ \pm 1 \\ 0 \end{matrix}$ - $p_{\pm 2}$, $p_{\pm 1}$, p_0

one way to get $\Delta = \pm 2$ [struck out] $p_{+2} \cdot p_{+2}$; [struck out]

Another way: $p_{+2} \cdot p_{+1}$; $p_{+2} \cdot p_0$

$$\begin{pmatrix} p_{+2} \\ + p_{+1} \\ + p_0 \end{pmatrix} \begin{pmatrix} p_{+2} \\ p_{+1} \\ p_0 \end{pmatrix} p_2 = (p_{+2} + p_{+1} + p_0)^2 = p_{+2}$$

[struck out] [struck out]

$p_1 = 2\left[(p_0 + p_1 + p_2) \cdot p_2\right]$; ways:

$p_0 =$

$2 = 2 + 2$
$1 = 1 + 0$, $0 + 1$ also $+1 + 2$, $+2 - 1$
$0 = 2 - 2$; $1 - 1$, $0 + 0$

6.8.05
5 PM

.00 : On sets of functions that may be "partial recursive": Fractals & the Mandelbrot set. Suggests ways to get (perhaps) cheap sets of functions that may be occasionally "undecidable".

The Mand. set does $f(z) = z^2 + c$: $f^{(n)}(z)$. For each value of (complex) c, we start w. z=0 and find n ∋ $|z|^2 > 2$ (outside ⊙).

n ≤ 255; if n ≥ 255 then assign n = 255. n = the color assoc w. c plotted.

So this is a way of assigning a no. integer n to each c value. Of course it takes time n to get "breakout" value n.

.10 Another param: "2", the radius of the "breakout circle" ≡ R

So C, R → n (complex, real → integer). We startout w. z=0; if we start out w. a different z value we get a diff. outputs so z_0 could be a param.

C, z_0, R → n (complex, complex, real → integer).

.15 To get complex no as funct of

.16 c, z_0, n (cx, cx, int); use complex. $(c + z^2)^n$ = complex no. (start at z_0) (We get real radius; real θ angle.)

For real values maybe use $R \to \frac{2}{1+e^{tR}}$ (big R means near 0, Red means 1)

.20 We don't have equal diffy approaching 1 or 0 — so not so hot.

A better way: take the real (or im) part of .15 – .16. It goes for −∞ to ∞, so squash it to 0,1 by $\frac{1}{2}(1 + \tanh R)$.

The escape radius R of 2 cannot be usefully increased. As soon as the iterate gets outside $2 |c|^2$, it must → ∞.

So essentially c, z_0 (2 complex nos.) are mapped by f(z) into positive integers, n that are used to represent colors. They could be ASCII.

.30 — So $\mathbb{C}^2$ into pos integers. So how can I get induction out of this? → (111.02)

This is of course kind of full of Wolfram's CA's. → Rudy Rucker's expansion into real outputs, of (PDE; approx. solns).

.34 An interesting thing about fractal pictures, is that in a real or complex region, we get more and more "continuous" info as the coords of "z" go to higher order bits!

So all of the bits (even the most precise) of z, give about equally impt info. — So it's like a sequential code!

see → 111.08

6.8.05
5TM

10 : 110.40: so [110.34 – .40 is an impt. remark]

Q: How many bits precision do they normally use on say fractint?

12 : 110.30 : To get prediction: Say the entire 2 dim complex plane or the entire 4 Dim. 2 complex Mand/Julia space is developed into 256 ("colors" = Ascii symbols).
Consider a certain fixed resolution, so the space is divided into fixed size pixels.
To do predictions of the sequence $x_1 x_2 \cdots x_n$ (x_i are ascii or radix 256 integers)
we look for as many cases of the horizontal (say) sequence $x_1 \cdots x_n$ as we can find.
The P.D. over the next symbol is our predn d.f.

8 To find the seq $x_1 \cdots x_n$: we just watch while the fractal is being created, pixel by

11 pixel. This is actually practical for small n. → See 112.20 for argument v.s. practicality of ↑ (.02 – .10)

12 Say we have "coded" $x_1 \cdots x_n$ this way : we have 1000 codes that we've found : we now know continuations of $x_1 \cdots x_n$ & we want to extrapolate them: Well, we just look at the points that code thru x_n, & we

15 increase resolution, to get their many possl. continuations.
Is this Legit? .02 – .11 seems legit. But .12 – .15 very questionable! } The legitimacy of .12 – .15 is suggestable (4.10 ff! Also, it's suggests .02 – .11 is not legit!

Actually, it's an interesting Q! not nearly ridiculous!

20 Say we apriori fix n at 2 corresponding to 2^{16} = 64k possl. sequences or : just 1 byte = 256 direct possys. So we run the PTM, remember all pixels of "color" x_1. Then we ↑ resoln. by 8 bits, & we continue all of the color x_1 cases to find ones w. continn. x_2 ; we remember them, then ↑ resoln further by 8 bits, to get matchg to x_3, ect.

This may be a legit. method of coding the corpus! — but if its a 8 byte sequence, this is 64 bit precision, a 24 byte → 192 bit precision.

30 Since we only want colors that correspond to the x_i's, we can save some time if we code the x_i so that the low no. pixels are "most likely",

32 HMM: This is mindful of a simple minded version of SUMAC!
i.e. the trick of .12 – .15 is where the simple minded SUMAC normally fails — by decreasing diversity. Hvr, if I use the Full (i.e. incomputable)) Sumac, its just regular

36 ALP — which I can approximate by (.02 – .11), & use it for small n, for "secondary predn" (i.e. TM_2). It will not be as good as PPM because it has so little memory of past : But use it in ll w. PPM for finding really crazy regys! (see 112.00)

38

STM

(Spec 111.40): So: go over 111.02–.11 again: will it work (w. suitable f(z))? T. fractal function maps complex nos. into 1...n (integers). T. time to output integer i is ∝ i.
So it is actually 2 maps: (one) real → 1...n; (two) ∈ real input → 1...n.
T. big Q is: what can we do to make t. fractal pgm more like Universal UMC? —
It seems clear that if it's calculating ≡ partial rec funct, it t. commonest form (rather than just a prim. rec. funct).

Wolfram had some conjectures on what t. output of a device would look like if it were universal. It might be well to try various fractal functions to see which ones seem most to act like what W. thot a UMC output should look like.

One way to (probably) get a ≈ UMC is to have t. first few bits of an input, describe t. function f(z) that's used to generate t. fractal. This first function desc facility should be good enuf to generate all prim. rec. functs. We might insert as part of t. primitive vocabulary: +, −, ×, ÷; e^z, ln z (in which z is a complex arg; so sin, cos, etc. are included — I don't know about inverses of trig & h trig.

Another possy is to have it be able to generate "Cordic" functions. → Hur, NOTE 113.24–.26!

Alternative approach: do 1 dim fractals: we put in a real no. & get out t. no. of times it recursed before getting > 4, say.

Or, use 2 dim xfm on a 2 dim vector $\vec{x}$ rather than complex nos. Use fixed pt integer arith, only.

REMEMBER! T. Goal here is to get some rather sophisticated predn for relatively short "contexts" to be used as 111.36–.38 for TM_2 in ‖ w. ≈ PPM.
(t relative wts of this u.s. ≈ PPM will be obtained "adaptively".

(Spec 111.11) Rethinking → 111.02–.11: This may not be so hot! The code for a particular horizontal sequence of pixel "colors" will be t. ≡ coord of t. first (or last) pixel of t. sequence plus t. length of t. sequence. So, in this sense, since (n, t. seq. length we are considering) is known & fixed for all trials, we can forget about it! So 111.02–.11 may be: "OK" After all! — BUT I'm not quite sure!!

On t. other hand, this method of predn. does not use t. entire corpus of cases that I want TM_2 to use! — in fact it uses no cases at all — it's a purely a priori predictor — but it is a kind of prediction because it induces a p.d. on all sequences of length k — but k can't be very large since computation time is exponential in k. — But k has to be large enuf to include t. corpus ... so looks bad.

6.8.05 Wed
5 PM

J.G.

0: The trick of 111.12–.15 could (might) give us much larger corpus possi., but suffers (a) from diffy 111.32 (simple minded SUMAC). Also Q of whether superhy resoln needed could be achieved w. reasonable cc. The 111.12–.15 trick can be used every once in a while to ↑ "diversity": i.e. to keep no. of codes for corpus > 1000, say — normally one just continues w.o. ↑ in resoln. This would deal w. diversity problem (perhaps), There is Q about whether it would give good predn!

T. problem of how to do ≥1000 bit resoln. economically — This may or may not be a problem. The details would have to be looked into. Fractint does seem able to ↑ resoln. arbly, w.o. diffy, but check this! ← I really don't know how this would work! (.22)

SN In QBasic 4.5 There is a Manual on Creating & Maintaining Quick Basic Libraries. These are for Pgms & Srtns that are used in writing new QB pgms. See if I have this in QB 4.0 (or 4.5?) & see if it is in PB.

Power Basic Consol Compiler (Version 4.0) Phone 1.800.780.770 Order no. 372117. Give Email addr. & they w Ship it. Cust. no. 31126

In our copy version of Visual Basic 6.0 (2 CDs): Product ID is
82892 – 884 – 7646991 – 92385 (≡ CD key) CD set: X04–12942 (4th e.) 1991–1998

PP385
In WinXP: to remove a password! Control Panel → user Accts → pick a task → change an Acct. → select an acct to change — click Remove The (My) Password or click on one of the Accounts that appears in the "Pick an acct. to change" section of the Users Accts window — enter your current password to confirm your identity. Then click "Remove The (my) password" or "remove the password".

Book Windows XP Home Edition ($20) Complete (U.G.) (Sybex) Win XP Home & Pro 6 books (Sybex) Denny & BAD Tyler Not nearly as good as?

"Removing a Pswd disables protection for that acct., so it is again freely accessible to anyone." "a diffnt xp book". Removing password is not same as setting password to "blank"

.22: (.09) → It is conceivable that one could keep on ↑ resoln, but the diversity would not ↑ — in which case, one would probably be stuck, & the predn. could not continue!

.22a Int. idea: I had that of early bits in code defining the funct to be used (f(z)) & thus setting "stuck" w. a particular f(z) hence forth — but a truly

.22b universal funct should have subsequent bits that are able to mod fy f(z) in an arby way!

.10 At present: Several apparent diffys in 111.12–.15: (a) that f(z) is "Universal enuf" (b) perhaps that subseq. bits can effectively Modify f(z) (.24–.26) (c) that the idea of ↑ resoln. of 111.12–.15 will work at all! (c) that there is a way to do this w.o. excess cc (possibly there is a way to used in perhaps fractint), (d) Will the system often get "stuck" w. not enuf diversity & be unable to predict or continue coding the corpus (deal w. this by "Back tracking"). (e) How to adapt this system to an unordered QA corpus (maybe unnecy: use this system for TM_2 only). (F) Any way to try it on a corpus w. simple regyst — (G) Say we have pixels a, b, c adjacent: all same "color!" ≡ G. If we code then a, then ↑ in resol n. to any amount will give same color, G. T. way I → (11.400 spec.)

30: (Spec 113.40) Expected to deal w. This: That I would originally have many codes for t. corpus ending in G. Some would predict a new color at some reason, & some would predict G: Another way to deal w. this difty is by "BACKTRACKING".

Stuff on Coverage by Pass = Misc N.G. (Recent) "Discover" Sometime June, or possibly May, July

0 : T. Sequence of Real (say) pts. would have cords like 1010^{∞}, 101 (various 3 bits seqs.) 101 011 6^{∞}, 101011 (& 3 bit seqs.) (instead of 3 bit chunks, use at least 8 or 10 or 12
11 over of gaseqs to get a large enough number so that colors or small pc's could get reasonable estimates. So R. code of .10 - .11 look like a sequence
14 of 3 bit "Bytes"! — so looks legit! — In fact, it looks more legit
15 than using 111.02 — 111.11. In fact, 111.12 - .15 is SUMAC. — & can utilize Backtracking (Frank!) [One way to expand diversity is to use defnt. rows (cols) of pixels. (≡ Defnt values for complex input.)] → .23
19

0 Another possibly Relevant idea is Chris Moore's demo that a simple dynamics System can represent an arby Time. Successive orbits representing (↔ correspondence) to Time Tape Motion. (1992 or 1994 or 95 or 1996 ?)

:3 : (.19) → : T. way I'm thinking about it now, there could be no "BACKtrack", because all possl. codes are investigated! We arrange t. "Radix" (= 8 = 3 bits in .10 - .14) so that a reasonable no. of each output symbol that is produced is adequate to get good precision and small likelyhood of "dead end" (≡ no cases of t. needed symbol occurring).

0. A way to introduce "Backtracking": say k is t. no. of symbols in t. alphabet of t. corpus. R is t. radix being used in t. coding (R will be an integral power of 2 usually, but R can be t. radix of t. no. system used to represent t. complex no., Z.

Say $\bar{k}$ is $\frac{1}{\bar{p}}$ where $\bar{p}$ is t. mean pc of a symbol.

R will be (about) t. no. of codes per character of t. radix R code dera. $\frac{R}{K}$ is (about) t. no. of times a output symbol occurs for each input symbol.

Rave any R input symbols ≈ "$\bar{k}$ output symbols".

If we develop all input symbols, one of t. output symbols will not occur a fraction of t. time ≈ $\frac{\bar{k}}{R}$. — Thus clobbering us — Since we "develop" all possl. code express input strings, if a output corpus symbol does not occur, we have a "dead end" — we cannot backtrack because all possl. codes have been tried. — A way to deal w. this:

STM

•

.10: Actually, if $\frac{R}{k} > 1$ we will have an exponentially # no of cases of most corpus outsymbols! So take $\frac{R}{k} \sim 10$ or 100; But develope only a randomly chosen subset of all codes for t. corpus. T. fraction developed can be so that t. No of coded cases of each corpus symbol is ~100 or maybe 1000.
So $\frac{R}{K}$ is comfortably large, i.e. we develope only a small fraction of legit codes. Then if we run into ↓ of diversity (or "Dead end" which is an extreme case of low diversity (≡ ∅ diversity), we can backtrack by developing more of t. codes starting 1 or more symbols back into t. past.
.0 So, As is, it would seem that I have an ideal SUMAC system, w. sort of adjustable "Backtracking" when needed!

T. prob. suggests that I misread an ordinary Sumac System w. occasional backtrack, but usually t. diversity was kept by enough so that there was no need for backtrack!
I always thought of Simple Sumac as automatically ↓ of diversity — Hvr., not so!
.18 ⟶ Each new corpus symbol can have several new codes extending many of the old codes.
I really have to look into .18 more Carefully: It may well be that ordinary
.20 Somac is normaly quite good!

Fractal prediction contrast w. normal Sumac in that fractal compression always has exactly one corpus char for each code char.

Re: t. cc of t. Fractal codes: to compute F(z) for progressive small increments of z, used derivatives! T. increments are so small that derivatives are t. G. (we can get upper bnd on error by knowledge of second derivatives.

.30! Sumac as stated above is only for ~ true Series prediction. Could I adapt it to
.31 do TMs (on QA-TMs)? Or could I somehow use a Sumac for TM_s (QA)?
.315
.32 Other realizations of Universality: .315 ⟶ CORDIC ↔ (17.28
Turing Matching — Emil Post (Wolfram's Cellular Automata, LIFE game, Dynamic Systems, Chris Moore (Az, Lisp), FORTH
Turing, String matching systems ... That technique method of Generating Trig, log functs) may make it easier to apply Sumac to Problems | Mechanic lang (RISC)
of QA predn, BAG predn, time Series Predn, a Gram 1, 2, Adm. Compression ....
Can we Genz. Cordic to do more Universal Prugs? It does Generate functions normally defined by integration (see Chris Moore paper) — so can we genz it to include all partial recursive functs?
It may be able to do all prim. rec. functs.
⟶ Consider inst Set of SONY Playstation 3 (PS3). Low cost per inst execution, hy speed, ...

6.10.05
JTM

F3	320 x 200 :	256
F4	640 x 480 :	16
F10	320 x 400	256
SF1	360 x 480	256

SF2 no
3 "
SF5 Synch monor.
6 "
7 "
8 —

SF9 320x200x16
SF10 no

0:115.40 : Away to think about latest SUMAC idea: Say k is radix of corpus, R is radix of Z representation (of "f(z)"). f( ) then maps Z^R space to ^{k} corpus space.

If R = k then for one f(·), the spaces are identical. Otherwise f( ) maps Z space to a part of itself (≡ corpus space). This means that certain corpus strings will have no codes (deadend). If R > k then we can have "deadends" occasionally, but as $\frac{R}{k} >> 1$, it becomes unlikely . . . .

Particularly after the corpus has many symbols in it. The redundance of representation is $\left(\frac{R}{k}\right)^l$ . . . where l is the length of the code there for.

[SN] fractint for Dos V.19.3 allows zooming to depth 10^{1600} — $1600 \times \frac{\ln 10}{\ln 2} = 5315.98$ Bits long. (≈3.32192)

So $2^{5315.98} = 10^{1600}$: A questions, how they do it, does it slow things down much, Could be easily extended?

Actually, this could amount to adequate predn. of radix 8 corpus. $\frac{10}{8}$ is only 1.25, but for long corpus, this can give adequate diversity! $1.25^{10} \cong 9.3$ So for corpus of length. "octs" (radix 8) 183 we have diversity factor of 100.

On Dosfractint 19.2 (I think): I kept on zooming in on a st. line edge betw. 2 colors but it kept on being a st. line edge: So this can be an impt property of fractals.

CF1 no
2 no
3 "
CF4 640x200 16
5 320x480x256 ≅ SF1
CF8 — hang.
AF4
AF6 + 376x564x256 — on V19 but not V20!
400X600 x256 — Best
ATI ultrames 800x600 no
video7 no
NoTandy 25 color Wid 512x482 256

I downloaded fraint 200.zip. This is a dos (version 20.): In general, The windows version has fewer impt features.

4.57 kby/sec = 36.56 k bits/sec) download by phone: Not Bad!

— is now on 512 many stick in (Ray) directory

[NB] On fract. SUMAC: It uses not the "Continuous" but the "discrete" until DP. — i.e. each code tells machine when to stop. So its necy to use Cover's "Extensive Complexity" — which too long to compute. I'm not so sure: I reference machines not only have one way out to get ALP.

[SN] Criticism of Fractal prediction: The chars in Corpus are not all treated Symmetricly at all! Apriori, any Permutation of corpus symbol names should leave the apri d.f. invariant — but In fractal predn, The chars assoc. w. small "Breakout numbers" are quite different from those w. large "Breakout nos.". Hvr, if one assigns the corpus symbols to the breakout nos. on the basis of observed past frequency — this objection is less strong.

.36 The objection has not been eliminated! — 116.00

.37 One way to get symmetric corpus symbol assignments. The first orbit to "breakout" will do so at a certain angle: We can assign corpus symbols to angle intervals out of 2π. Certain f( ) functions could give some bias on these angles, hvr. Stop 117.10

5 TM

10 (see 116.36): My former objection to Source was (partly) that ("errors" or omissions) early in t. corpus codes, could never be corrected — except by a fortuitous backtrack: Prob not so. — a good Source should be able to compensate for early errors — but a suitable tsp. perhaps must search as it's needed. This is (partly) t. idea of being effectively able to modify early codes (or, t. effects of early codes on t. present problem which is all that's necy)) a

(see 116.20) SN Some random Notes on fractal predn: Using a Circular topology of output in 116.37 ff: we can easily divide up output in complex plane by comparing its real and im. parts: re>0 gives that im>0 gives t.f. (re|>|im| gives t.f. etc.

1) C. Moores set of Analog f(z)'s
2) D.F.m ⊙.

SN2 A Continuous D.F. on finite circle is a 4pt d.f. to add to ±cos(0,+90|0,+1|i.e. 0,2π. The reason is a single "phase" pt. for any broad d.d's t. function 2(cos x+θ)+1 normalized will do it. Its peaks at θ, low pt. at θ+π. Hvr, its variance is large. If can't have only small vars.

Putting $e^{\frac{-x^2}{2\sigma^2}}$ on t. unit ⊙ would do it. Whether t. result would be easily manipulated Math'ly ... I don't know. Just expand $e^{\frac{-x^2}{2\sigma^2}}$ in a Fseries.

Very possibly useful integrals are known: $\int_{-\infty}^{\infty} e^{\frac{-x^2}{2\sigma^2}} \cos^n x\,dx = ?$ — is a fnct of $\frac{n}{\sigma}$ probably maybe mult by n^2 or σ^2 (or n or σ ⊙)

23 SN3 We can Use C. Moores (Shannon's) set of universal analog. functions for f(z) in t. fractal Source. Perhaps a code would start w. a short binary string that decides t. discrete aspects of t. function, followed by 1 (or more) Continuous parameters. Look at C. Moores Paper in "Ps" file. I think I have a first draft P.P. in "Hard Copy (But)" The "Integrate" function is Expensive if

27 done digitally. → 119.20

28: 115.315 Perhaps try to get more General functions via Cordic.

0 Consider t. same-types of 115.32 & others: One reason I liked w. Lisp was that progs were easy to spot (common subtrees). So TM2's job was easier. Hvr, in t. fastest ideas about SUMAC, I don't Consider using a TM2 at all! So, say I used Machine lang. & no TM2. How good could t. system be? In General, I've been Thinking about SUMAC's w.o. considering that. Hvr, it would perhaps be a different Corpus (kind of for TM2) from t. one I've been Considering for QATM (?). Maybe not so distant! In both we have codes & langs of codes, & compressions achieved for each of t. codes.

If TM2 is not used/considered, then RISC Machine code would probably be best. perhaps I could start out very slowly w. no TM2, then use TM, for TM2 when t. TSQ

Was adequate. This may take a long time! Also if the primary primitive lang. does not have easily recognized regexps in its codes, this $TM_2 = TM_1$ will still have a hard time!

Actually, perhaps the best thing to do: for each lang, try to see what kinds of regexps occur in it. Existence of a detection of subpgms would seem to be a major hdwr.

Perhaps just write a bunch of pgms in each lang. & think of a common thing in each. See how systems are used/detectable.

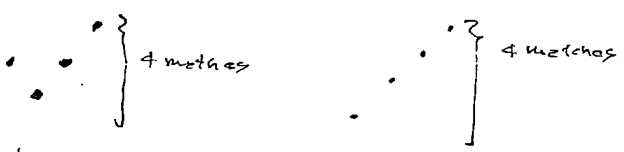
Hdwr., for Sumac as I've been thinking about it, we have a single equivalence corpus, & a set of trees that are codes for it. For a corpus that is many strings of symbols, the mechanics of Sumac has to be revised ... it's applicable at all!

On Parallel realization of TM. Its easy to parallelize secondary TM_1 — very little cross communication needed. But for TM_2 (secondary induction), all of the secondary corpi of the TM_1's are needed. These can be updated slowly, hdwr. the secy corpus is quite large — many parallel codes for primary corpus. Hdwr., when TM_2 finds new regexps they can be given to TM_1 rather slowly — as those regexps are discovered slowly.

: Context Tree Srch (for TM2): Say we only have binary branches in the context tree. For depth 4 discovery, we have $16 \in 2^4$ separate PPM corpi (≡ BTREE corpi) — one for each path for distance 4. So for a particular context to be matched, we look in all 16 (depth 4) corpi. There will be much overlap, of course. Could we simply pool all case counts of depth 4 matches? For each depth (1, 2, 3 or 4) we will know how many matches occured, & the d.f. of predicted next symbol". From this data, we can make a p.d. over predicted symbols. How good would it be?

Note that .20 - .24 seems quite different from what I originally had in mind.

Formerly a "nth degree match" was a subtree match to the "query tree" in which there were n matching symbols that were reachable in a connected path of matches. So: A and B both have 4 old style matches; New style: A has 1 depth 3 match, 2 " 2 " 5, 1 " 1 "



B has 1 depth 4, 1 " 3, 1 " 2, 1 " 1

6.12.05 119

≈ TAB

4; 42 and 43

STM

10 : The "16 corpi" could find all matches of depth $[i]_0^4$ [illegible] individually. The optimum method of pooling the data, is unclear now.

My Test Bias is that (on 118.37), A is "Better than" B. — That more compact context, small depth, but many contig. symbols is important.

0 : There were 2 possible "objections" to Sumac: ① That I'd not be able to maintain diversity; the backtracking could do this. ② How to do TM_2 was always a major problem — what regys in the TM_1 code & how to recognize them.

To some extent, ① is an empirical Q to be found by Expt.

② Is somewhat Empirical, but needs much theoretic analysis to decide on what (2 particular Primitive Lang.) is "good"/"adequate"/"fast"/"cheap" → see 150.10 ff / 151.06 for progression ≈ "Sumac"

0 : 117.27 : Dynamic Systems as Tmcs. See Cris Moore Paper ≈ 1996: Actually, I think "by dynamic system" any kind of continuous mapping from t=n to t=n+1 would work for his analysis. The impt. Q is just how he did the mapping from continuous funcs to Tmc. states & tape storage/the data. Try similar maps using easily calculable Continuous funcs. Find out where "output tape" is. Can I devise a uio machine or equivt.? The Theory arrays to do induction on regular (non-uio) Tmcs. — But where is the "output" is an impt. Q.

36 To what extent is the fractal pixel color a possibly useful "output" for a universal Mach.? → 121.20

0 : On Sumac (w.o. TM_2) & "Phase I" with TM_2: Why does TM_2 seem to help so much? Sumac, as is, is universal, tho it may need backtrack (also ways to effectively modify early code.) (or whatever else is wrong.)

32 In TM_2, we put a pd on codes for TM_1. This is effectively putting the output of a stoch

33 grammar (in general a Umc output for "random" input) as input to the "final" umc.

So why is this TM_1, TM_2 combin better than the "simple" Sumac? TM_1 (to start) is good at finding a standard set of common regys in the corpus. It effectively compresses them, so TM_1 can concentrate on finding new kinds of regularities. Once a new kind of regularity is discovered, it should be incorporated into TM_2's "Stoch Grammar". → See 120.09 for good UNDERstanding of this!

6.13.05
5 TM

∞ : It would be nice, if, indeed, we could incorporate all or most newly discovered regy types into TM_2's "Stretch Grammar". T. regys in TM_2's S. Gramm would have to be detected by "TM_3" (= TM_1 eventually/usually/always). "TM_3" does come into play until (1) t. corpus of codes for t. primary corpus is large enuf (2) TM_2 has had a proper/adequate TSQ ~~for t. requi~~ to make likely, feasible t. kind of regy. recognition we need.

Still, I want to understand clearly, advantages of a $TM_{1,2}$ .... device v.s. a simpler "1 level" Sumac.

09 [AHH]! T. $TM_{1,2,3}$ ... Is well understood as a variety of "Coding & Recoding ..." —
0 an ordinary, Legit method of coding, equiv. to a "well coded" Sumac! In fact, this View, gives us a much better understanding of "TM_2's corpus" in QATM! It enables us to more clearly define just what are "Legit-Corpi" for TM_2 in QATM!! ☺.

See if I can Unify t. View of (.09) w. ~~[illegible]~~ t. View of 119.32–.33, as TM_2 providing a "Preprocessing" of t. "Random" inputs to TM_1's "Reference Unit".
17 [6.16.05] → Woops! Not so fast!: .09 may be OK for BU srch, but it's not clearly relevant to Lsrch: In Lsrch we do ~~[illegible]~~ trial Codes in pc order, so "recoding" couldn't ordinarily ↓ code length. T. only possy is that somehow, t. recoding by TM_2 uses a → (126.10)

9

0 : Note on Image Compression: One simple compression scheme: Code t. diffrnce betw. successive scan lines (non-lossy). Each pixel has a context of 1. Actually, this may not be any better than just coding ~~each~~ pixle in each line as difrnce betw. that pixel & t. previous pixle in t. line. Each pixl. has ~~[illegible]~~ 2 closest pixle contexts a.
-3

scan line n → C B D
scan line n+1 → A [X]
line n+2 →

[X] has A & B as same distance & C, D not much further.
(X) has ~~[illegible]~~ closest context of ≈ (A C B D). But how do we use this for coding? [X is closer t. $\frac{A+B}{2}$ than to either A or B. ∓

Maybe try $\frac{A+\frac{1}{2}C+B+\frac{1}{2}D}{3}$ as a guess for X. It would be nice if we could store contexts in Lex order, like in PPM. My impressn. is that t. problem is w to storing, retrieving subtree contexts. Look at more successful image coding schemes: Can they be modified to help code, retrieve sub-string contexts

In t. case of images, we will be able to store t. context ACBD's & corresponding p.D. of X.

We could do a Lempl. Ziv. analog & look at t. previous case w. t. "largest common context in t. past" (whatever that means). My impression is that t. single dimension quality of text did make L.Z. coding feasible. — A 2 dim analog ... ? How?

6.13.05 Ultimate Dynamic Systems as Computers: Noise problem .20

5 T/M

: What about 1 & 2 dim Lossy coding? (I think lossy is easily modified to not lose, by adding code for amt. of error.)

One V.G. lossy code is MP3 for audio: V.g. selection of what info to discard, for 2 dim. images, DFT's are used & (or Wavelets plans?). Info discarded is in hyper frequency components: So loss of resolution

I think there are V.G. lossy codes for Movies &/or 2 d. images. Also latest JPEG may be significantly better than normal JPeg.

Look at Dondori's 2000, 2003 papers comparing JPEG compression of [redacted] images w. LZ of images. Note this may be old JPeg — Not v.g. (?) — More Recent JPeg may be Much Better!

: 119.36: Dynamic Systems as UMCs: [New idea]: I was concerned that "Real World" dynamic Systems would always have noise "from External World" in hyper order bits of Memory — So that Computer would have an effective upper limit on precision &/or on RAM capacity per System param (= degrees of freedom).

For predn viz a UMC w. random input, We can have 2 kinds of input for a UMC. Input by User: Random input by Outside world. To do predn, the user wants to find many also short codes. (The noise from outside world is not purely random — it does have "regs", but these regs simply add info to that input, so we need more of it to qualify for "truely" random input.

How it works: User finds (User) inputs that give corpus up to nth bit. He then repeats these inputs to get p.d. on next bit. Each input may have to be repeated several times in order to get 1% of the "nth bit"'s because of "outside world noise". Certain input codes will be less noisy than others. (I suspect short codes will be less noisy). Certain long codes will be so noisy that they will be effectively useless? Also certain long codes will not be able to code. Known corpus w. much repeatability. I imagine text & repeatable code for corpus would could be have repeatable regions.

In general, I think that prediction w. this system will not be difficult, But use

5. TM

may be limited in the lengths of codes that are usable — so corpus size will be limited. We probably can deal w. this by using many Dynamic Systems machines. → .08

To do prediction, we can just run in many codes and if they reduce the corpus, continue. Most codes. For codes that successfully reduce known corpus, we will repeat them many several times to get d.f.s on their countings. We start w. a search in (length) PC order of codes. When we find a code for known corpus, we continue it for prediction, & repeat it for predictions to get p.d. for predictions.

-08 : .05 Just how we use many machines to deal w. limited capacity of each Dynamic System — is unclear. How do we predict a Very large corpus? How do we do larger TSQS?

Note that the individual (micro) computers need not approach universality. They can be small parts of a larger machine that does approach Universality. (w. limitations in addition to small memory)

One (sort of trivial) way to combine M machines: Each one is essentially a highly variable sub-machine that uses only a few bits of each dynamic variable (~ degrees of freedom) — like a sub-memory or portion Arith unit. They are then connected in "usual" computer ways. So the memories can add up to very large size. However, communication betw M machines will be relatively expensive so we want to reduce it's need as much as poss.

Soln of QA probs; also seq. of induction probs by GP! GA/GP → Note objection: 123.11

One of Dittys was need to verify each new cand. on the entire corpus of problems. This was partly solved by ordering probs by difficulty & trying hardest first. An important idea that I may have missed in past work: Before GP is used to solve BUSeach problems in induction. As such, we always start out w. a default code for each problem that the prgm tries to improve by a Hill climbing. So every Cand will have at least a PC of code for the entire corpus.

CONTRAST w. LSeach, in which PC = 0 until the first code is found — which is quite interesting Grammar!

A cand will not be tried for entire corpus, unless it does well for the hardest probs thus far. Also, for perhaps almost all of the "very easy" part of the corpus, we don't have to test cand on all cases. — Maybe just some representative cases.

So, using a good TSQ & GP and CFGD for search, one might be able to do fairly well. This is close to what I had in mind for QA TM, (Phrases) but there, I used LSeach TSQ's, & TM_2 to develop a grammar good for TM_1's each. However, since Lisp progs are already in a cfg — it's not clear as to what Grammar discovery would do. Obtaining good pc's is one problem to be

10 world on! also Context sensitivity à definitions of [redacted].sub-trees.

Just how sub-trees defn. would occur is unclear.

- Also, perhaps $\frac{\sigma}{M_{2K}-M}$ criterion to search sub corpus.

As each case comes in, I sort its G_New into a file w. G ordering. As I do so, I keep track of $G_i \in G^2$: I can then get changes in $M_i \in Z$ for any subset of Cands w. $G_i < G_{new}$ so I can tell if $\frac{\sigma}{M_{2K}-M}$ has changed. I would do this $\frac{\sigma}{M_{2K}-M}$ calcn only in these cases.

I. $\frac{\sigma}{M_{2K}-M}$ criterion was originally designed for a PPR corpus. Using it on a CFG or CSG corpus may not work so well.

15 I. discussion of 122.20ff is a bit deceptive. The problems are not like each problem is an individual to be solved. It's not like finding a short code for each subcorpus.

13 The 2 different problems must be put in this form: We have an overall O^2 (for QATM). The O^2 is a pd on symbols & macros (=tokens) & segments of symbols & sub trees.

From each O^2 we have to find 2 (or set of) short functions from $(Q_i \text{ to } A_j]_{j=1}^{N}$

So O^2 is a def over/2 function spaces, & the jth problem is to find functions in this space that map from Q_i to A_j. To do a prediction from Q into A, we just act pd on functions (which is O^2) & go to A's. — Actually this is the "natural" Q & soln: O^2 is a pd on the set of functs from Q to A.

"derivation grammar"?

I forget just what the form is:

each string in the derivation is an acceptable sentence.

NB T. pd of .21 is not a developable lang in which every string of symbols is a function. It could well be that certain short strings of functs have no pc, but/local shorter strings do not exist, & CFG can be of this form.

The P.D. of interest will be a finite set of functions: but such a soln would not be universal: i.e. it would not assign pc > ϕ to all possl. Any — So we do want a grammar of an infinite lang — so if it's a CFG, it must have at least 1 loop → see 124.06

An input idea of 122.22ff & 123.11ff is that Boosting — for BU search — may be relevant to these problems. — But normal Boosting doesn't seem relevant to Lsrch.

32 A modification of the Boosting idea related to Lsrch in QATM! — (Perhaps).

In order to apply Boosting to an Lsrch: the way we apply it is to modify the d.p. Bn & "guiding P.D." that is used for Lsrch. Perhaps certain functions are used in solns to certain QA's that don't work for pc's for these solns. We then modify pc's of the primitive set of functs, so that the pc of the active corpus is raised. — What this is not closely related to "Boosting".

36 → 32–36 sounds very much like Lapp's idea for a Bin Seq!

Lo temp Gleo to fix wires on "Universality": (for Lucy)

in 122.20ff One way to ~~complete~~ test a D.f. on functions, wrt a set of QA problems. List the functions in pc order: try each function on all QAs in the corpus, & list which ones it works for. After enuf functions have been tried so that all QAs have at least 1 found that works, we can sum the pc's of each relevant function for each QA, to get ~~the~~ pc for that QA. Then get product of all pc's for all QAs to get pc of corpus wrt. that D.f. on functions.

:124.29 I wrote about this particular model of QA interaction before (123.20ff-.29). Much criticism: I don't remember if it worked correctly or not — i.e. was it universal? If the p.d. on functs gave >0 pc to every recursive function, then the system is universal. Actually, it corresponds to the 3 input case! A Big Diff for QATM is that it is not normed & perhaps very expensive to normalize?

? → What was not universal was a "search technique" to find the D.f. on functions (?!?) (.12)

— I wanted a universal d.f. over enumerations {p.d's = programs on funct. on functions spaces.}

(.09) Halts may be normalizable not so diffic! for 3 input case: the "R" input need not be interpreted as $pc = 2^{-|R|}$. I can use any interpretation I like, & get them to sum ≤ to ∞. I can get them to be semi measures, because certain R values will give no output for certain Q inputs. (.151) ↳ I could just use the semi measure product to get a Π "pcf", & I select functions to max this value (for a given CB ≤ const). This would bias toward functions that summed to near 1. ! → (.25)

So: Anyway: to normalize, we just empirically find a normalizing constant for each "3 input case" — Or say $R_i]_{i=0}^{\infty}$ are the legal (integer) values of R_i. (Ri has an automatic "end of string" marker) we use some $F(R_i) \ni \sum_{i=0}^{\infty} F(i) = A (\in \infty)$ then A^{-1} is the normalizing constant. This assumes all (R_i, Q_j) converge (.151). (f(i) = α^i (for some α) was our guess: α is "typical" ? $\alpha = 1+\epsilon$. normalizing $\Rightarrow \epsilon \alpha^i$ is expected value of i^{th}.)

Q: Is there any pathological case that would give very bad results (.151 → .19)?

Q: In my earlier work cited in .06: Did I really have any serious objection to (123.20ff-.29 (∃ 3 input case)?

(.17) → Also, remember the results on the "continuous" Univ. d.f. — Here the semi measure ~~was~~ appeared to measure rapidly as $\epsilon \to 2^{-\infty}$ the probability of "0" → 0 rapidly for large corpi: So maybe Normalizing is not such a big problem.

So I now have at least ∃2 not very distant solns to the p.d. on

— String A in QATM: ① T. "obvious" soln of 123.20-.29; 124.06ff

② p.d. over {0, 1, End of string Symbol} as a function of {Qj and the first k bits of Aj}

On second look: This d.f. over 3 symbols .... I don't think I solved it! I guess I could modify the 1 dim d.f. on [0, 1] to be a 2 dim d.f. on (0, 1, stop) — but it's not so neat — particularly since the stop symbol has different properties from 0, 1.

Over in .18 (.24R) I discuss normalization of the "A" d.f. and it does give a way to assign pc to the "stop" symbol — but this is the probability of "stop" in R, not in A ...

So the 2 "solns" (for ① vs ②) (.30) (.31) are not the "same".

Wait Norman. See note on (.30R). V.G. Looks like it will not be a problem when we get down to any step in the TSQ (e.g. to BC step)

Hmm I'm not sure that proof applies in general case! — If it does apply but the univ. proof of convergence of QATM ~~is~~ not proof of Sol'78 Norm 3