Practical Machine Learning for Software Engineering

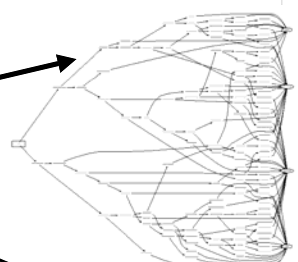
Tim Menzies,
NASA/WVU IV&V, USA
Tim@menzies.com
http://tim.menzies.com

The Eighteenth National Conference on Artificial Intelligence
July 28 - August 1, 2002
Edmonton, Alberta, Canada

Sound bites

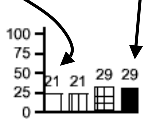
- Knowledge famines
  - SE= data starved
- Controllers, not just classifiers
  - Don't tell me what is, tell me what to change

Stop staring at the scenery and tell me where to steer or what to dodge



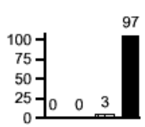
BASELINE

500 examples of bad--, bad, ok, good



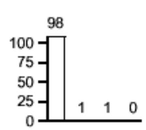
BEST ACTION

6.7 <= RM < 9.8 And 12.6 <= Ptratio < 15.9



WORST ACTION

0.6 <= NOX < 1.9 and 17.16 <= LSTAT < 39



A spectrum of machine learning methods

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2,Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Note: this is an empirical categorization, not a theoretical one.

Preliminaries

Hello, nice to see you,
are you in the right place?

About the audience

- Industrial practitioner-oriented.
- Material is suitable for:
  - AI-novice or
  - the technical manager of software engineering projects.
- (Also, for ML researchers:
  - A head's up on today's industrial realities)

About the author

- Background:
  - Commercial consultant: ES, OO
  - Academic: KA, ML, RE
    - Ph.D. in KA: General Principles for testing KBS
    - Currently:
      - SE research chair, NASA/WVU IV&V facility, USA
  - Applications work:
    - ML for decision making early in the software life cycle

About the author's biases

- Want to augment today's industrial software practices
- Industry needs K.I.S.S. techniques,
  - knowledge farming, not data mining
  - Mature tools, well documented
    - E.g. decision tree learners
    - Not (yet) e.g. inductive logic programming

Further reading

- **Practical Machine Learning for Software Engineering and Knowledge Engineering**,
  - T. Menzies,
  - *Handbook of Software Engineering and Knowledge Engineering*, (volume 1, 2001)
  - Available from http://tim.menzies.com/pdf/00ml.pdf
  - All references [X] in this presentation come from this paper.
  - Extra, newer, references marked as {X} shown at end of paper

Symbols in curly brackets = new refs

Numbers in square brackets =Old refs

Further reading (other kinds of ML)

- {Mendonca99}: great review article on ML
  - Large list of available tools
- Michalski's excellent survey of ML types [25]
- Neural nets [11]
- Data mining [22]
- Special issue SEKE journal, knowledge discovery [26]
- Worth watching: inductive logic programming [2,7]
  - Come by IJCAI 2011 and I'll tell you all about it's applications
- Genetic algorithms: {Goldberg89}.
- Bayesian learning {Cheeseman88}

More further reading

- International workshop on model-based requirements engineering, San Diego, 2001
- Many excellent papers including ….
  - Neural networks
    - learn predictors for software development effort
  - Model checking and machine learning
    - to learn a restriction that reduces the search space within a program
  - Treatment learners
    - to find project management actions.
    - to learn key features of a model
  - Statistical methods
    - For data mining and risk prediction

Alphabet soup

- AI= artificial intelligence
- COTS= commercial off-the-shelf packages
- ES= expert systems (a.k.a. KBS)
- KA= knowledge acquisition
- KBS= knowledge-based systems
- KDD = knowledge discovery in databases
- K.I.S.S.= keep it simple, silly
- ML= machine learning
- RE= requirements engineering
- SE= software engineering

Expectation management (FAQ1)

- "But you only talked about old-fashioned learners that used e.g. decision trees…"
  - Yes. K.I.S.S.
- "But you didn't talk much about data mining"
  - Known SE case studies don't use large data sets
  - I "farm", not mine.
  - But some SE data mining examples presented
- "You went on and on about your treatment learner"
  - Yup: there is a reason I wrote this tutorial.
  - #include salesResistance.h

Expectation management (FAQ2)

- "You skipped some slides."
  - Perhaps I did. Life is short.
- "It took a while before it got technical."
  - Before getting geek-ish, we spend 40 (ish) slides on executive education.
- "Some material was rushed" or "I wanted more details on X"
  - Pique : to excite to action by causing resentment or jealousy; to stimulate; to prick; as, to pique ambition, or curiosity.

Data = medium, Model= none

S.E. examples of model = learn(data)

Road map

ML SE case studies from here

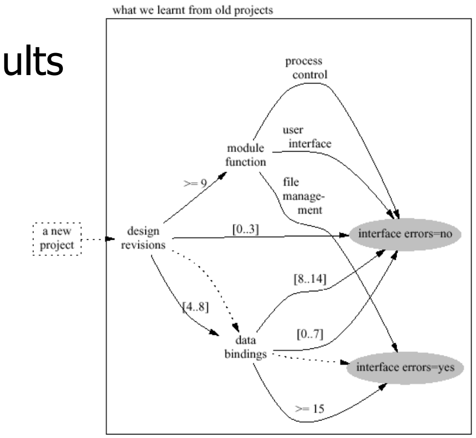
		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Some case studies in ML for SE

- Case studies use off-the-shelf tools
- Case studies are "what", not "how"
  - We'll do "how" later

Predicting software faults

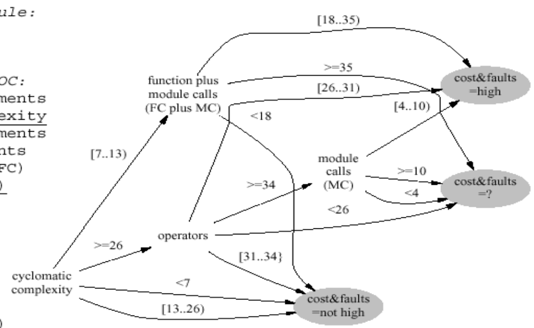
- data bindings: domain-specific metric for assessing module interrelationship.
- interface errors: errors arising out of interfacing software modules.
- [30]



Predicting software faults [37]

Across whole module:
total operators
total operators

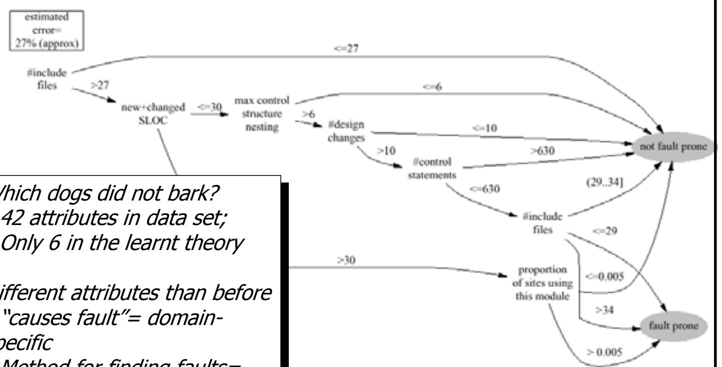
Averages per KSLOC:
assignment statements
<u>cyclomatic complexity</u>
executable statements
decision statements
function calls (FC)
<u>module calls (MC)</u>
<u>FC plus MC</u>
IO statements
IO parameters
origin
operands
<u>operators</u>
comments (C)
source lines (SL)
SL minus C
format statements



Note what isn't there

- The missing bit:
  - "Was there any particular aspect of the crime calling for additional study?"
  - "Yes" replied Holmes, and pointed to the curious incident of the dog in the nighttime.
  - Inspector Gregory replied, "The dog did nothing in the night-time."
  - Holmes said, "That was the curious incident."

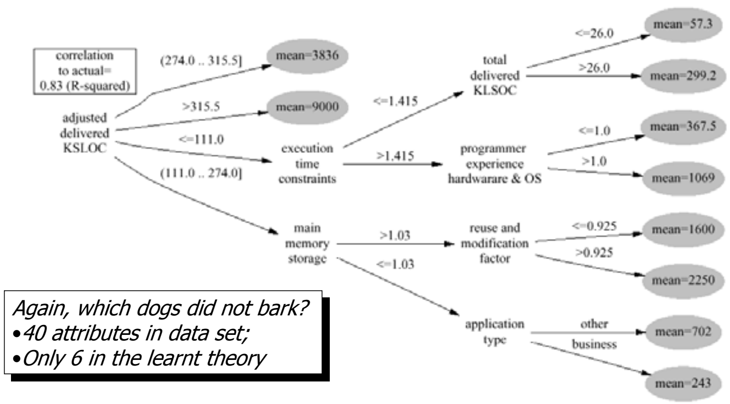
Predicting software faults (again) [16]



Which dogs did not bark?
- *42 attributes in data set;*
- *Only 6 in the learnt theory*

Different attributes than before
- *"causes fault"= domain-specific*
- *Method for finding faults= general*

Predicting development times (in months) [36]



Again, which dogs did not bark?
- *40 attributes in data set;*
- *Only 6 in the learnt theory*

What's missing is important

- Missing from the previous trees:
  - The majority of the 18 attributes
  - Only 4 in the tree
  - And one of them is cyclomatic complexity of ill-repute [10,p295]
- First control statement:
  - Don't bother trying to adjust 18-4=12 of the variables

So ML for SE is easy, right? WRONG!

Non-monotonic reasoning for conflict resolution [Mylopoulos92]

Auto-gen tests from specification [Zlatareva93]

Logic programming [Kowalski88]

Model-based diagnosis [Poole90]

Design: you need one!!

? ***Folks are too darn lazy. More process, more tools!***

! ***Premise: SE is model/data starved***

A puzzle:

Why not more ML in SE?

Road map

Thesis:
SE is here

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Why not more ML in SE?

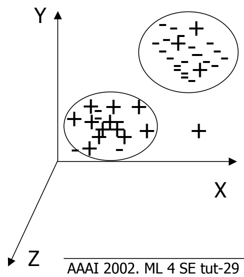
- Amazingly short literature about ML for SE
- Why #1: maybe-
  - Doesn't work? Wrong! (see below)
  - Works too well? Industry won't disclose it's competitive edge? Perhaps

Why not more ML in SE? (2)

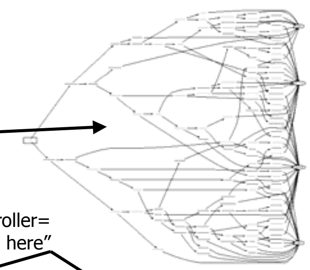
- Why #2: my theory:
  - SE managers want controllers, not predictors
    - "Don't tell me we are heading for a cliff, tell me what to do about it."
      OR
    - "Don't tell me we are going ok, tell what to do so we are likely to do OK in the future."
  - ML needs data and SE is a data-starved domain.

Standard ML: classifiers, not controllers

- INPUT: data+classes=instances:
  - E.g.<x=1,y=2,class=+>
- Learn descriptors of the clusters
  - If X < 4 then if Y < 4 then class = +
  - If X > 4 then if Y > 4 then class = -



What does a manager want?



Option 1: classifier=
a map of "you are here"

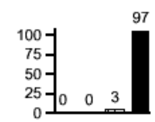
Option 2: controller=
a map of "where to go from here"

BASELINE

500 examples of bad--, bad, ok, good

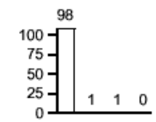
BEST ACTION

6.7 <= RM < 9.8 And 12.6 <= Ptratio < 15.9



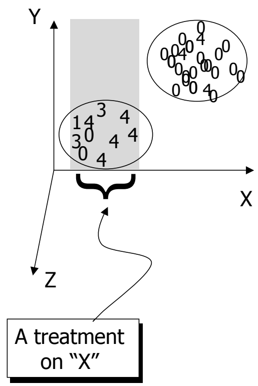
WORST ACTION

0.6 <= NOX < 1.9 and 17.16 <= LSTAT < 39



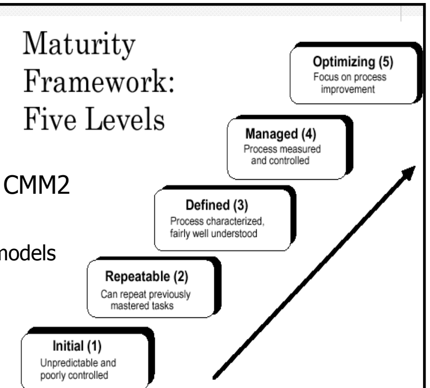
Learning controllers

- Given a partially ordering between classes
  - E.g. lowPayShortTermWork is worse than highPayLongTermWork
- Learn policies that "nudge" us towards "better" and away from "good"
  - Smaller policies >> larger policies
  - Assumption: some attributes can control the domain



SE= data starved

- ◈ Most organizations < CMM2
- ◈ The axiom famine
  - If CMM < 3, then no models
- ◈ The data famine
  - If CMM < 4 then no meaningful data
  - COTS= data castles (you can't get it)
  - dot-coms= no "past experience"



		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Data mining

- From repositories of data, learn theories
- (Data mining) can only be as good as the data one collects. Having good data is the first requirement for good data exploration. There can be no knowledge discovery on bad data. [22]

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Knowledge farming

- When data is absent:
  - Plant a seed:
    - Some quickly built theory of a domain
  - Grow the data:
    - Execute the theory, collect the logs
  - Harvest:
    - Summarize the logs

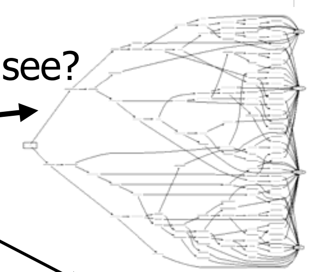
		available data			
		none	small	medium	huge
pre-existing models	none	#1.Plant=quickly build models		#3. harvest= summarize logs	
	small	#2. Grow= execute to build a log of behavior			
	medium				
	huge				

Q: The new models only summarize your old models. So what is the value added?

A: Accessibility. Summarizations can be customized to the interests of the audience. Interactions tacit in #1 are obvious in #3.

What do you want to see?

Classifiers?



Monitors?

Controllers?

BASELINE

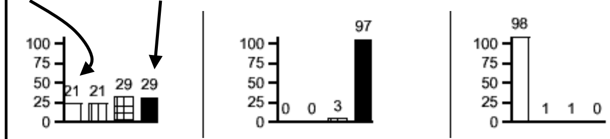
500 examples of bad--, bad, ok, good

BEST ACTION

6.7 <= RM < 9.8 And 12.6 <= Ptratio < 15.9

WORST ACTION

0.6 <= NOX < 1.9 and 17.16 <= LSTAT < 39



Q: When is ML practical for SE?

- A: (Timm) SE= data-starved domains,
- Before learning from data
  - Need a modeling process to generate a theory
  - To generate data sets.
- ML practical for SE when the modeling and learning stages are
  - simple
  - inexpensive.
- See below

Just a minute: data mining is never practical for SE?

- Average CMM < 2 (usually),
  - mining unlikely in average SE.
- Exceptions:
  - Predicting faulty software modules [16, 30,37]
  - Predicting development time [36]
- Discussed above

So why all the buzz on data mining?

- Welcome to the wired world-wide-web world
  - Data sets galore
    - E.g. millions of examples of someone browsing your web site
    - E.g. Understanding gigabytes of data from satellite remote sensors, telescopes scanning skies, human genome data, scientific simulations
  - Practical off-the-shelf association rule learners
    - When they buy THIS, what ELSE do they buy?
    - 1GB of data: 10,000,000 examples
  - Solves a NEW problem
    - What does it offer for the ye olde SE problem?

Data = small ➔ medium,
Model= none

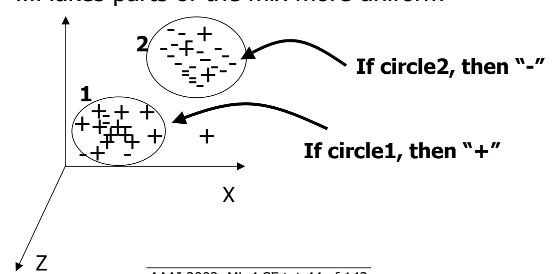
Model=learn(data)

Road map

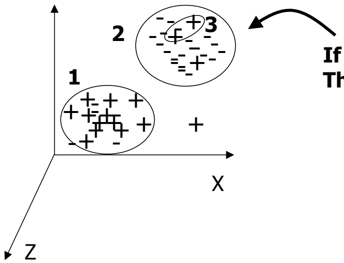
		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Learning = simple

- Given examples with a mix of classes
- Find a "Description"
  - Which, if applied…
  - …Makes parts of the mix more uniform



Repeat, recursively, to find more complex descriptions



If circle2,
Then If circle3 then "+"
Then "-"

1R = no recursive descent

C4.5 = repeat till remaining space includes less than "minobs" examples

Method

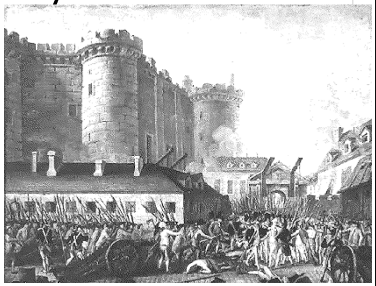
- Define a "diversity" metric
- For attribute ranges seen examples
  - Divide examples on that range
  - Measure diversity before and after division
- Best attribute range=
  - One that reduces the diversity the most
- Repeat recursively

Measures of diversity

- Simpson diversity index: biologists
- 1- repeatRate: cryptographers
- Gini index: econometricians
  - As used in CART {Breiman84}
- Entropy: information theorists
  - As used in C4.5 [33]

Low vs high "diversity"

- Diversity=0
  - All examples belong to one class
- Diversity = maximum
  - When all classes equally represented
- Best "splitter" decreases diversity the most.

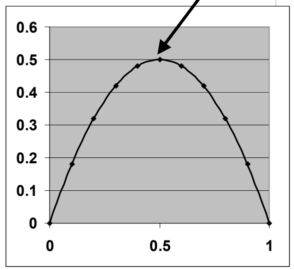


- The French revolution: Liberté, Égalité, Fraternité,
- The knowledge revolution: liberté, latte, et diversité

GINA index

- Assuming 2 outcomes
  - % occurrence of two classes in an example set = P1, P2
  - P2=1-P1
- Diversity = measure of likelihood of pulling the same class twice from an example set.
  - $P1^2*P2^2$
- Compliment if sum of all diversity measures
  - $D(P1,P2) = 1 - (P1^2*P2^2) = 1-(P1^2*(1-P1)^2)$
  - $= 1- (P1^2+(1-P1)*(1-P1))$
  - $= 2*P1-2* P1^2 = 2*P1*(1-P1)$

Diversity=max (P1=P2)



C4.5's Entropy measure

- GINA, C4.5 generates trees
- C4.5
  - Different trees can be assessed via their "information content"
    - A.ka. Entropy
  - To make a tree:
    - Split the dataset on the most informative attribute range
    - Repeat for the subsets

Class

```
TSH <= 6 : negative
TSH > 6 :
|   FTI <= 64 :
|   |   TSHmeasured = f: negative
|   |   TSHmeasured = t:
|   |   |   T4Umeasured = f: compensated hypothyroid
|   |   |   T4Umeasured = t:
|   |   |   |   thyroidsurgery = f: primary hypothyroid
|   |   |   |   thyroidsurgery = t: negative
|   FTI > 64 :
|   |   onthyroxine = t: negative
|   |   onthyroxine = f:
|   |   |   TSHmeasured = f: negative
|   |   |   TSHmeasured = t:
|   |   |   |   thyroidsurgery = t: negative
|   |   |   |   thyroidsurgery = f:
|   |   |   |   |   TT4 > 150 : negative
|   |   |   |   |   TT4 <= 150 :
|   |   |   |   |   |   TT4measured = f: primary hypothyroid
|   |   |   |   |   |   TT4measured = t: compensated hypothyroid
```


C4.5's Tree = "message" (more)

- For two class datasets discrete datasets
  - P= #positive examples
  - N= # negative examples
  - $A_1, A_2, \ldots A_v$= the different values of A
  - P_i N_i, examples with attribute A_i
- Information required for that tree is:

$$I(p,n) = -\left(\frac{p}{p+n}\right) log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right) log_2\left(\frac{n}{p+n}\right)$$

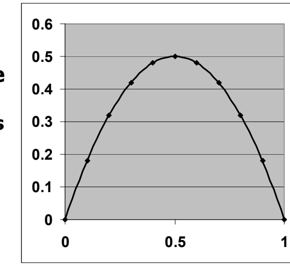
C4.5's Tree = "message" (yet more)

- For two class datasets discrete datasets
  - P= #positive examples
  - N= # negative examples
  - A_1,A_2,... A_v= the different values of A
  - P_i N_i, examples with attribute A_i
- Split on A_i
  - Best split has highest "gain" in expected weighted average value of the information in that split

$$E(A) = \sum_{i=1}^{v} \left(\frac{p_i + n_i}{p + n} \right) I(p_i, n_i) \qquad gain(A) = I(p, n) - E(A)$$

Learners differ on their diversity function

- **Recall the GINA index function for two-class systems ... ... ➔**
- **Different diversity functions have different shapes**
  - **Therefore propose different splits**
- **GINA:**
  - **Favor splits that isolate largest target classes in one branch**
- **C4.5:**
  - **Favors balanced splits**
- **Some data mining packages allow customizations of splitting function**
  - **Since there is no best splitter**
- **Me? Off-the-shelf C4.5**



An example using C4.5

- C4.5 [33]
  - International standard in ML
    - Not the "gold" standard but the "old" standard
    - New learners benchmarked against C4.5
- Need a data file:
  - X.data
- Need a data dictionary:
  - X.names
- (btw, author=Quinlan=Australian)

C4.5's Golf.names (the data dictionary)

- cat golf.names

```
Play, Don't Play.

outlook:      sunny, overcast, rain.
temperature:  continuous.
humidity:     continuous.
windy:        true, false.
```

Classes

Attribute1-discrete

Attribute2-continuous

Tip: much faster with discrete than continuous

C4.5's
Golf.data (the examples)

outlook	temp	humidity	wind	
sunny,	85,	85,	false,	Don't Play
sunny,	80,	90,	true,	Don't Play
overcast,	83,	88,	false,	Play
rain,	70,	96,	false,	Play
rain,	68,	80,	false,	Play
rain,	65,	70,	true,	Don't Play
overcast,	64,	65,	true,	Play
sunny,	72,	95,	false,	Don't Play
sunny,	69,	70,	false,	Play
rain,	75,	80,	false,	Play
sunny,	75,	70,	true,	Play
overcast,	72,	90,	true,	Play
overcast,	81,	75,	false,	Play
rain,	71,	96,	true,	Don't Play

If don't know, write '?"

Tip: the less '?' the better

Class: last entry On each line

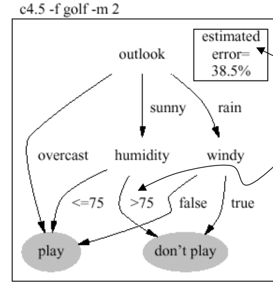
Running C4.5

- c4.5 -f stem –m minobs
- c4.5 –f golf –m 2

Minimum # of e.g.s needed to justify making a new sub-tree

Defaults to "2"

c4.5 -f golf -m 2



Error rate too high! (I aim for<=20%)

Humidity only Interesting Above and Below 75%

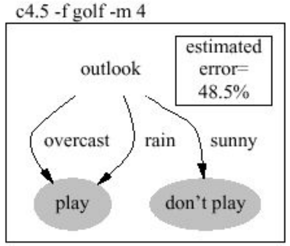
Hey! Where did temperature go?

C4.5 decided that temperature was not "informative"

c4.5 –f golf –m 4

- Larger "minobs"
- Smaller tree
  - Easier to read
  - Less accurate

c4.5 -f golf -m 4



Often, trees MUCH bigger

```
C4.5 [release 8] decision tree generator    Sun Jun 25 17:33:55 2000
----------------------------------------

    Options:
    File stem <circ>

Read 378 cases (6 attributes) from circ.data

Decision Tree:

light3 = light: good (8.0)
light3 = dark:
|   light1 = light: good (13.0/4.0)
|   light1 = dark: bad (357.0/45.0)

Simplified Decision Tree:

light1 = light: good (21.0/5.9)
light1 = dark: bad (357.0/50.0)

Tree saved

Evaluation on training data (378 items):

     Before Pruning           After Pruning
    ----------------   ---------------------------
    Size      Errors   Size      Errors   Estimate

       5   49(13.0%)      3   49(13.0%)    (14.8%)   <<
```

Bad practice to test on examples seen in training.

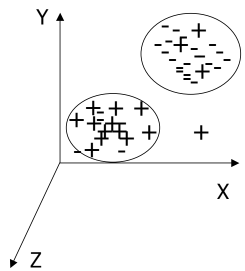
Mis-classification rate on training set

Estimate on future data

More readable, ?? Less Accurate

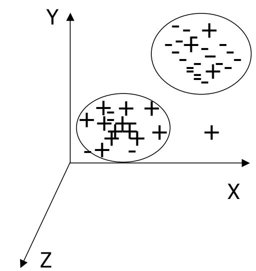
Digression: on errors in learning

- Usually an error in the descriptors
- Any inductive generalizations lose data
  - By definition
- Theory may not be contained in the data (e.g. Z)



Digression: on errors in learning (2)

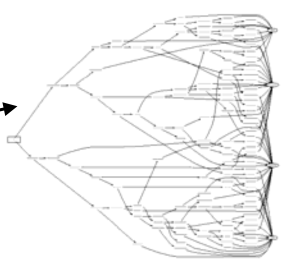
- Language of the learnt theory may be incomplete.
- If perfect theory, lose of future generalization
  - Need to throw away some details



Digression: on errors in learning (3)

- Real world theories can be too large to view.

They get MUCH bigger than this



- So, after learning, comes pruning
- Pruning = throwing away some of the theory

C4.5's Generalizations

- Continuous values
- Missing values
- N classes
- Extensions:
  - X-val
  - Pruning (cull bushy trees)
  - Rule generation (?? Easier to read)
  - Boosting and bagging

10-way cross validation (xval)

- Don't test on the training set
- For a dataset with class frequency distribution F
  - Divide into (e.g.) 10 buckets, preserving F
  - For I = 1 to 10,
    - Remove bucket I
    - Train on all other nine buckets
    - Test on bucket I
- Final error = average of xval errors
- All automated in standard C4.5 distribution
  - xval.bash c00 10 -m2048

Bagging and Boosting (1 of 3)

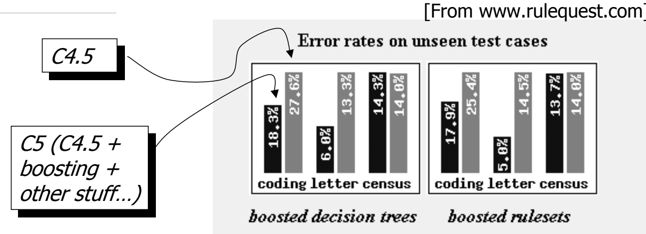
- A man with one watch knows the time.
  - A man with two is never sure
  - A man with 10 watches, 8 of which say "bedtime" is confident that it is time to sleep
- Ensembles of classifiers can be more accurate than any of it's members:
  - Strangely, only if some of them disagree

Bagging and boosting (2 of 3)

- Bagging:
  - Learn from data divided into N overlapping sets
- Boosting:
  - Learn from examples misclassified last time
  - Boosting focuses on harder and harder problems
- Combination rules can be very simple
  - Unweighted voted can suffice
  - Votes, weighted by probability of single conclusion

Bagging and boosting (3 of 3)

[From www.rulequest.com]



- {Dietterich97}:
  - Boosting > Bagging > raw-C4.5

Is Occam's razor blunt?

- "Entia non sunt multiplicanda praeter de necessitatum." -- William of Occam (c 1350)
- See, we tried it, and the reverse worked better
- Lesson: seek Swiss army knives
  - Lots of blades
  - That all cut slightly differently

How many examples are enough?

- Depends on the noise in the data
- Best case:
  - Platonic examples:
    - Each one extracted from a domain expert that represents exactly a distinct different case
  - Only two classes
  - # examples = dozens
- Typically:
  - Play with 100s, learn with 1000s
  - Warning- this is a gross generalization

How many examples = too much?

- With tricks, C4.5 runtimes grow linearly on the size of the dataset
- For "off-the-shelf" C4.5,
  - Windows NT, 128MB ram,
  - gcc compiler, cygwin environment
  - 12 continuous attributes per row
  - Limit=300,000 examples
  - Could have gotten more under (e.g.) Linux

Some runtimes

- C4.5:
  - E.g.1
    - 50 numeric attributes
    - 150641 examples
    - 2 hours
  - E.g.2.
    - 12 attributes, discrete
    - 1000 examples
    - A few seconds
- C5.0 (evaluation copies: http://www.rulequest.com/download.html)
  - E.g.1
  - 15 minutes

Never enough data

- Learn decision trees for 11 problems
  - using half or all the available data (thousands of examples)
- In all but 1 case:
  - More data= less error
  - More data = larger theories
  - Implications for the reuse enterprise?

The Catlett Results

domain	tree size change	Error rate change
demon	*0.97*	*0.51*
wave	1.91	0.95
diff	1.46	0.69
othello	1.68	0.8
heart	1.61	0.65
sleep	1.73	0.91
hyper	1.74	0.83
hypo	1.45	0.85
binding	1.51	0.82
replace	1.38	0.8
euthy	1.33	0.61
mean	1.52	0.77

J. Catlett, Inductive learning from subsets or Disposal of excess training data considered harmful, Australian Workshop on Knowledge Acquisition for Knowledge-Based Systems, Pokolbin, 1991, pages53-67

Data = huge
Model = none

KDD:
Knowledge Discovery in
(very very large) Databases

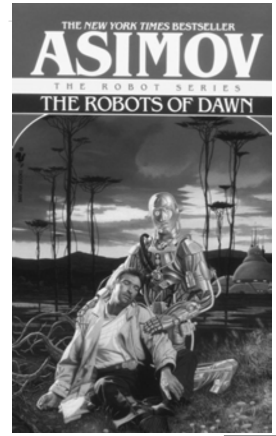
Road map

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

What is KDD?

- Non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data
- Can be done by (e.g.) C4.5, CART, et.al.
- BUT, if data sets large, gets more complicated.

www.amazon.com



- **Customers who bought this book also bought:**
- *The Naked Sun* by Isaac Asimov
- *The Caves of Steel* by Isaac Asimov
- *I, Robot* by Isaac Asimov
- *Robots and Empire* by Isaac Asimov

The Data Mining Desiderata (1 of 2) {Bradley98}

1. Require one scan (or less) of the database if possible.
   - A single data scan is considered costly, early termination if appropriate is highly desirable.
2. On-line "anytime" behavior:
   - a "best" answer is always available, with status information on progress, expected remaining time, etc. provided
3. Suspendable, stoppable, resumable;
   - incremental progress saved to resume a stopped job.
4. Ability to incrementally incorporate additional data with existing models efficiently.

The Data Mining Desiderata (2 of 2)

5. Work within confines of a given limited RAM buffer.
   - Ooops, good-bye C4.5
   - Argued against by some.
     - "Memory is cheap": {Webb00}, TAR2
6. Utilize variety of possible scan modes: sequential, index, and sampling scans if available.
7. Ability to operate on forward-only cursor over a view of the database.
   - This is necessary since the database view may be a result of an expensive join query, over a potentially distributed data warehouse, with much processing required to construct each row (case).

From classifiers to association rules

- Classifiers
  - Ranges ::== (Attribute$_X$ Op Value$_Y$)+
  - Op ::== >=, >, =, <, <=
  - Ranges ➔ class=X
- Association rule learners
  - Ranges1 ➔ Ranges2
  - Ranges1 $\cap$ Ranges2 = $\varnothing$
- AR learning= classifiers if..
  - |Ranges2|=1
  - "Attribute" is just a classification
  - "Op" is just "="

Classifiers vs Association rules

- Target:
  - Classifiers seek a small set of pre-defined targets
    - The classes.
  - For association rule learners, the target is less constrained.
    - Any combination of ranges.

Support and confidence

- Examples = D , containing items I
  - 1: Bread, Milk
    2: <u>Beer</u>, <u>Diaper</u>, Bread, Eggs
    3: <u>Beer</u>, Coke, <u>Diaper</u>, <u>Milk</u>
    4: <u>Beer</u>, Bread, <u>Diaper</u>, <u>Milk</u>
    5: Coke, Bread, <u>Diaper</u>, <u>Milk</u>

- LHS ➔ RHS = {Diaper,Milk} ➔ Beer

- Support = | LHS U RHS| / | D | = 2/5 = 0.4
- Confidence = | LHS U RHS | / | LHS | = 2/3 = 0.66

- Support-based pruning- reject rules with s < mins
- Check support before checking confidence

Example of support-based pruning

1Item	Count
Bread	4
Coke	**2**
Milk	4
Beer	3
Diaper	4
Eggs	**1**

2Item	Count
{Bread,Milk}	3
{Bread,Beer}	**2**
{Bread, Diaper}	3
{Milk,Beer}	**2**
{Milk, Diaper}	3
{Beer,Diaper}	3

3Item	Count
{Bread,Milk, Diaper}	3
{Milk, Diaper ,Beer}	**2**

Support-based pruning
• Min support =3

Ignore subsets of items of size N,
• only if N-1 support > min-support

Without pruning: $^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With pruning: 6 + 6 + 2 = 14

Classifiers vs Association rules (again)

- Classifiers:
  - Assume entire example set can fit into RAM.
- Association rule learners
  - Very big data sets.
- {Agrawal93}: the APRIORI algorithm:
  - very large data sets
  - 10,000,000 examples
  - 843MB

BTW, does KDD solve the SE problem?

- Timm definition:
  - SE = helping a community evolve a common and executable understanding of a domain in a cost-effective manner
  - Large manual part
  - Typically a data starved activity
- So, IMHO, KDD solves a new problem
  - A new and exciting problem
    - Understanding gigabytes of data from satellite remote sensors, telescopes scanning skies, human genome data, scientific simulations, web demons watch users
  - But not the olde SE problem

Data = small
Model = some

Belief networks

model2 = learn(data,model1)

Road map

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

tampering → alarm → exodus → report
0.02 (run away!) (hello, operator? I want to report a fire)

fire → alarm
fire → smoke
0.01

```
Pr of tampering = 0.02
     Pr of fire = 0.01

Pr of smoke  given [fire=yes] = 0.90
Pr of smoke  given [fire=no]  = 0.01

Pr of report given [exodus=yes] = 0.75
Pr of report given [exodus=no ] = 0.01
Pr of exodus given [alarm=yes ] = 0.88
Pr of exodus given [alarm=no  ] = 0.001

Pr of alarm  given [tampering=yes,fire=yes] = 0.50
Pr of alarm  given [tampering=yes,fire=no ] = 0.85
Pr of alarm  given [tampering=no, fire=yes] = 0.99
Pr of alarm  given [tampering=no, fire=no ] = 0.0001
```

Use Bayesian analysis to update probabilities, given new information.

Q1: What if there is a report, but no smoke?

tampering → alarm → exodus → report = YES!!
0.50 (was 0.02)

fire → alarm
fire → smoke = NO!!
0.03 (was 0.01)

Q2: What if there is a report, and smoke?

tampering → alarm → exodus → report = YES!!
0.03 (was 0.02)

fire → alarm
fire → smoke = YES!!
0.97 (was 0.01)

Example from : {Poole98}, p371

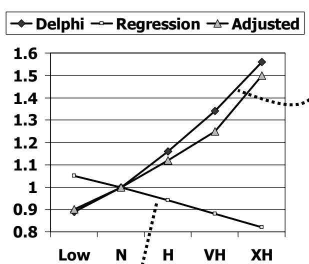
Source = http://www.swi.psy.uva.nl/projects/SWI-Prolog/download.html
= http://www.cs.ubc.ca/spider/poole/ci/code.tar.gz
Files = code/acp/bnet.pl;
code/acp/bnet_t1.pl

Case study #1: {Chulani99}

- The COCOMO-II project
  - Open-source software cost estimation
  - Reuse vs effort:

    XH : multiple product lines
    VH : across product lines
    H : across program
    N : across project
    L : none
  - Regression over data from 83 software projects
  - Regression conflicted with "Delphi values"
- Tune regression values using Delphi expectations

Expected: reuse increase effort



Data: reuse lowers effort

(data + delphi + tuning) > data

	COCOMO-II (1997)		COCOMO-II (1998)	
Pred(X)	**83 projects**	**161 projects**	**161 projects-based on Delphi**	**161 projects-based on Bayesian**
Pred(20)	46	54	48	63
Pred(25)	49	59	55	68
Pred(30)	52	63	61	75

Percentage of estimated effort within X% of actual

Case study #2 {Fenton00}

Naïve (common) model:
pre-release effort
➔ post-release faults

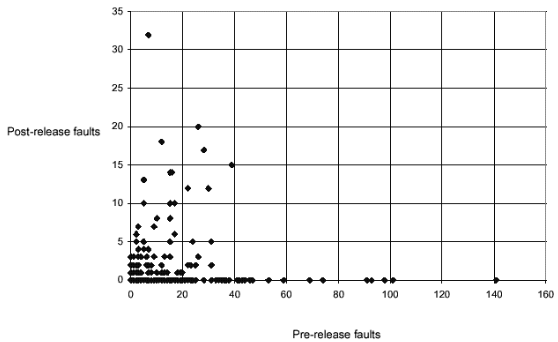
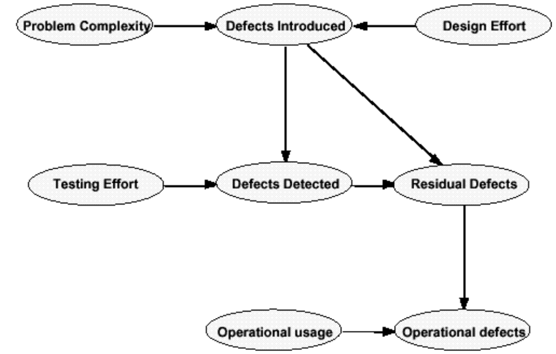
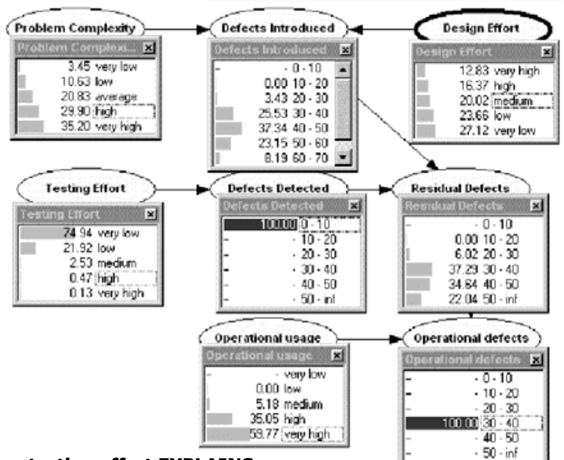


Figure 5 Scatter plot of pre-release faults against post-release faults for a major system (each dot represents a module)

Non-naïve model (simple version)





Low testing effort EXPLAINS
1) some observed operational defects and
2) low pre-release defects

Data = none
Model = a lot

Knowledge farming

data = execute(model1)
model2 = learn(data)

Road map

Thesis:
SE is here

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

To repeat: When is ML practical for SE?

- SE= data-starved domains
  - Usually (but see counter-examples above)
- Before learning from data
  - Need a modeling process to generate a theory
  - To generate data sets.
- Timm: ML practical for SE when the modeling and learning stages are
  - simple
  - inexpensive.

Knowledge Farming

- Plant a seed= lightweight modeling
- Grow the datasets= random simulations
- Harvest= summarize

Plant the seed (example from [4])

- Seeds must be fast to build
  - Not require data we don’t have right now
  - E.g. not the precise numerics we can’t get without further study.
- Use a qualitative model.
  - Numeric X ➔ qualitative X’
  - X’ = + if X > 0
  - X'= 0 if X = 0
  - X'= - if X < 0

Qualitative circuits

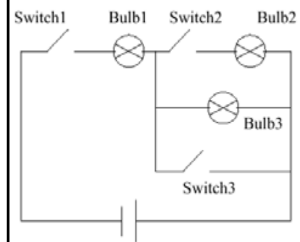
```
%blub(Mode,Light,Volts,Amps)
bulb(blown,dark,  Any,     0).
bulb(ok,    light, +,      +).
bulb(ok,    light, -,      -).
bulb(ok,    dark,  0,      0).
```

```
%classification(B1, B2, B3,Class)
% needs 2 our of three bulbs working
classification( ok, ok, B3,   good):- !.
classification( ok, B2, ok,   good):- !.
classification( B1, ok, ok,   good):- !.
classification( B1, B2, B3,   bad).
```

```
%switch(State,Volts,Amps)
switch(on,     0,    Any).
switch(off,    Any,  0).
```

```
%sum(X,Y,     Z).
sum(+, +,     +).
sum(+, 0,     +).
sum(+, -,     Any).
sum(0,  +,    +).
sum(0,  0,    0).
sum(0,  -,    -).
sum(-,  +,    Any).
sum(-,  0,    -).
sum(-,  -,    -).
```


A qualitative circuit




```
circuit(Sw1,Sw2,Sw3,B1,B2,B3,L1,L2,L3):-
    VSw3 = VB3,                  % 2
    sum(VSw1, VB1, V1),          % 3
    sum(V1,VB3,+),               % 4
    sum(VSw2,VB2,VB3),           % 5
    switch(Sw1,VSw1,C1),         % 6
    bulb(B1,L1,VB1,C1),          % 7
    switch(Sw2,VSw2,C2),         % 8
    bulb(B2,L2,VB2,C2),          % 9
    switch(Sw3,VSw3,CSw3),       % 10
    bulb(B3,L3,VB3,CB3),         % 11
    sum(CSw3,CB3,C3),            % 12
    sum(C2,C3,C1).               % 13
```

```
go  :- tell('circ.data'), go1, told.
go1 :- functor(X,circuit,9), forall(X, example(X)). % > 700 solutions


example(circuit(Sw1,Sw2,Sw3,B1,B2,B3,L1,L2,L3)) :-
   classification(B1,B2,B3,Class),
   format('~a,~a,~a,~a,~a,~a,~a~n',[Sw1,Sw2,Sw3,L1,L2,L3,Class]).
```

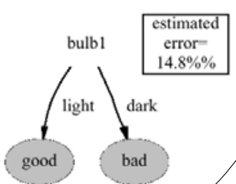

Results from > 700 examples

circ.names:

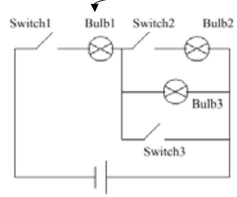
good,bad.
switch1: on, off.
switch2: on, off.
switch3: on, off.
bulb1: light, dark.
bulb2: light, dark.
bulb3: light, dark.

Command line:

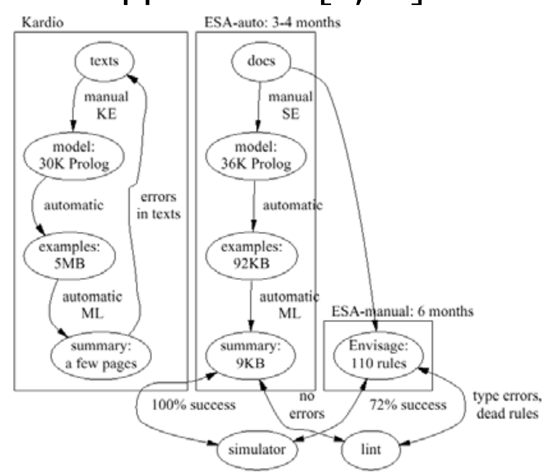
c4.5 -f circ -m 2



Watching bulb1 tells us the rest. Insightful? Or dull?



Real applications [4,29]



Data = none
Model = a lot (part 2)

Knowledge farming with TAR1

data = execute(model1)
model2 = learn(data)
model3 = key_parts_of(model2)

Road map

Case studies

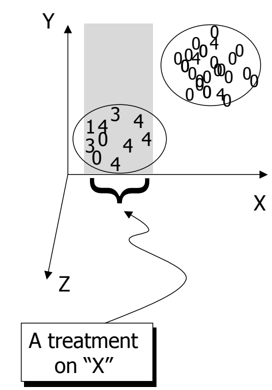
		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

So, it's a solved problem, right?

- Just build it quick,
- Run it at lot (random inputs)
- Summarize as controllers, not just classifiers

Reminder: learning controllers

- Swing through the trees
- Looking for attributes with
  - different ranges leading to different classifications
- Score classes:
  - "good",
  - "bad"
- Return attribute ranges that increase frequency of "good"
- Not a classifier, but a controller



So tell me how to control my software projects [23]

And it gets worse that this!

			NASA software projects						
			KC-1 (very new project)		FB-3 (moderately new project)			BJ-1 (very mature project)	
	ranges		now1	changes1	now2	$changes2_a$	$changes2_b$	now3	
Scale drives	prec = 0..5	precedentness	0, 1		2,3			4,5	
	flex = 0..5	development flexibility	1, 2, 3, 4	1	?	3,4	0-5	0,1	
	resl = 0..5	architectural analysis or risk resolution	0, 1, 2	2	?		0-5	4,5	
	team = 0..5	team cohesion	1, 2	2	4			3,4	
	pmat = 0..5	process maturity	0, 1, 2, 3	3	?		0-5	4,5	
Product attributes	rely = 0..4	required reliability	4		4			4	
	data = 1..4	database size	2		?		1-4	1,2	
	cplx = 0..5	product complexity	4, 5		3,4,5			3,4,5	
	ruse = 1..5	level of reuse	1, 2, 3	3	?		1-5	4,5	
	docu = 0..4	documentation requirements	1, 2, 3	3	1			3,4	
Platform attributes	time = 2..5	execution time constraints	?		5	4		2,3	
	stor = 2..5	main memory storage	2, 3, 4	2	?		2-5	3,4	
	pvol = 1..4	platform volatility	1		?		2-4	1,2	
Personnel attributes	acap = 0..4	analyst capability	1, 2	2	2			2,3,4	
	pcap = 0..4	programmer capability	2		2			2,3	
	pcon = 0..4	programmer continuity	1, 2	2	?		0-4	2,3	
	aexp = 0..4	analyst experience	1, 2		?		0-4	3,4	
	pexp = 0..4	platform experience	2		?		0-4	3,4	
Project attributes	ltex = 0..4	experience with language and tools	1, 2, 3	3	2			3,4	
	tool = 0..4	use of software tools	1, 2		1	2,3		3,4	
	site = 0..5	multi-site development	2		?		0-5	?	
	sced = 0..4	time before delivery	0, 1, 2	2	?		0-4	2	
	# of what-ifs (combinations of $nowX \cup changesX$) =		$6 * 10^6$		$3 * 10^9$		10^9	10^7	

Software risk estimation model

- COCOMO- open-source software estimation tool [1]
  - Needs SLOC
  - Needs tuning of internal parameters
  - Pred(25)<=75
- The Madachy model of software risk [20]
  - "Risk"= risk of running over the planned development time
  - Tables to "tweak" the COCOMO tables

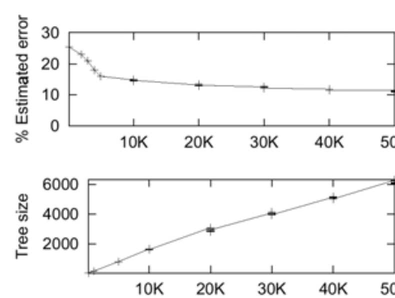
		rely=				
		very low	low	nominal	high	very high
sced=	very low	0	0	0	1	2
	low	0	0	0	0	1
	nominal	0	0	0	0	0
	high	0	0	0	0	0
	very high	0	0	0	0	0

Ensemble learning

- Learn 45 trees:
  - For 3 SLOC guesses
  - For 3 tunings
  - For 5 increasing sample sizes
- Query the trees for attributes that in most trees (>67%) improve class ratios

5 sample sizes

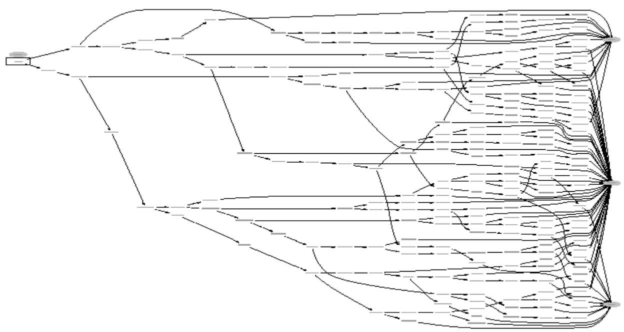
- Monte carlo sampling of inputs
  - Stop when error rates stabilized



Error not zero

HUGE trees: condense with a tree query language

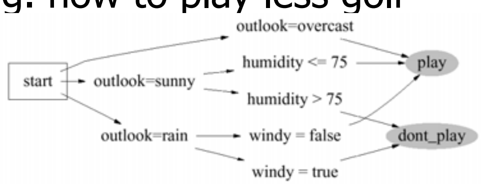
250 nodes
(<< 6000 nodes)



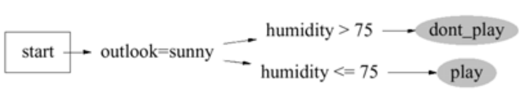
TAR1

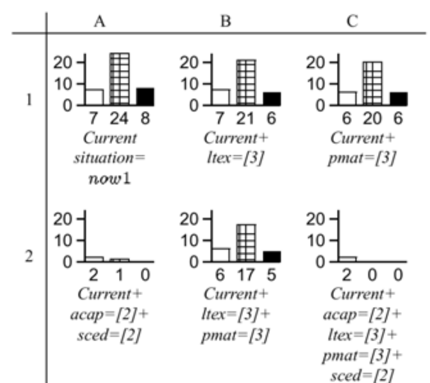
- Don't show the 45 trees
- Show the control strategy learnt from the trees
- Find attributes that appear on branches to different conclusions
  - But using different ranges
- R1= attribute.range(s) ➔ bad
- R2= attribute.range(s) ➔ good
- Control = R2-R2

E.g. how to play less golf



- Prune all branches that contradict outlook=sunny
- Decrease relative frequency of "play"





Which dogs did not bark?

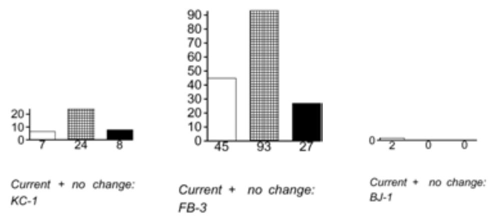
- *11 proposed changes*
- *2^11=2048*
- *7 don't always change class frequencies*
- *2 change, a little*
- *2 change a lot*

Enough here to make a reasoned management decision about merits of (e.g.) process improvement

©Tim Menzies, 2002

Does it "work"?

- Did we clone N projects, and their programmers, and run these with control and treated groups?
  - Well…
- But the learnt profiles reflect intuitions "on the floor":
  - KC1.risk >> FB3.risk >> BJ1.risk



©Tim Menzies, 2002

Data = none
Model = a lot (part 3)

More knowledge farming

©Tim Menzies, 2002

Road map

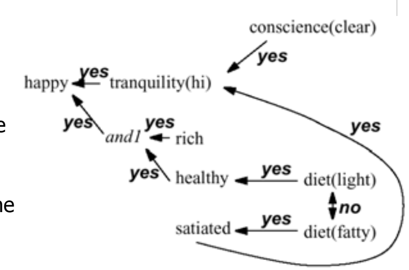
TAR2= a better TAR1

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

©Tim Menzies, 2002

Data sets from "what-ifs"

- Given at theory with contradictory possibilities,
- Then can only go over here OR over there
- So any single simulation accesses some subset of the variables



$Path_1$ $\texttt{happy} \leftarrow \texttt{tranquility(hi)} \leftarrow \texttt{conscience(clear)}$

$Path_2 : \texttt{happy} \leftarrow \texttt{tranquility(hi)} \leftarrow \texttt{satiated} \leftarrow \texttt{diet(fatty)}$

$$Path_3 : \texttt{happy} \leftarrow \texttt{and1} \begin{cases} \leftarrow \texttt{rich} \\ \leftarrow \texttt{healthy} \leftarrow \texttt{diet(light)} \end{cases}$$

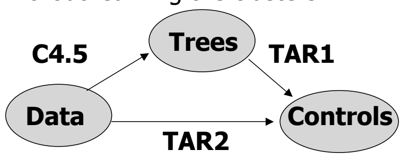
Path1 and Path2 OR

Path1 and Path3

Datasets with lots of "?"

Current research

- TAR2:
  - Learning control strategies from what-if logs
  - ?? Simple frequency counts will do
- Curiously:
  - we can learn differences between clusters…
  - … without learning the clusters.

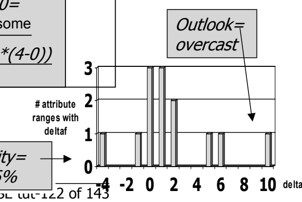


outlook	temp	humidity	wind	hours on course
sunny,	85,	86,	false,	none (2^1=$2)
sunny,	80,	90,	true,	none
sunny,	72,	95,	false,	none
rain,	65,	70,	true,	none
rain,	71,	96,	true,	none
rain,	70,	96,	false,	some (2^2=$4)
rain,	68,	80,	false,	some
rain,	75,	80,	false,	some
sunny,	69,	70,	false,	lots (2^3=$8)
sunny,	75,	70,	true,	lots
overcast,	83,	88,	false,	lots
overcast,	64,	65,	true,	lots
overcast,	72,	90,	true,	lots
overcast,	81,	75,	false,	lots

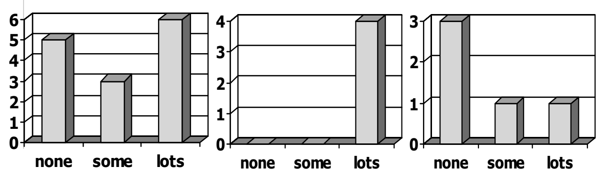
Harvest: improve the scores

1. ***Different*** **frequency counts within different class**
2. ***Weighted*** **by the score difference between the classes**
3. ***Normalized*** **by total frequency**

Delta(outlook.overcast)=10=
Lots → none   Lots → some
((8-2)(4-0))+((8-4)*(4-0))*
(4+0+0)



Harvesting from data: golf



If no change, Then lots of golf 6/(6+3+5)= 43%

If outlook=overcast, Then lots of golf 4/(4)= 100%

If humidity=90..97 Then lots of golf 1/(4)= 25%

assumes causality, controllability

Least change: bribe DJs to lie about the weather

Least monitor: watch the humidity- alert if rising over 90%

Harvesting from data: housing

BASELINE

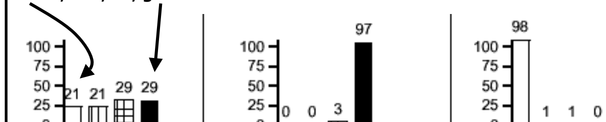
500 examples of bad--, bad, ok, good

BEST ACTION

6.7 <= RM < 9.8 And 12.6 <= Ptratio < 15.9

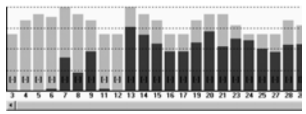
WORST ACTION

0.6 <= NOX < 1.9 and 17.16 <= LSTAT < 39



Harvesting from ARRT

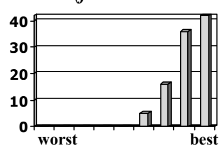
- ARRT: manual risk balancing



- TAR2+ARRT:
  - 88 possible actions ($2^{88} \approx$ billion*billion*billion combinations)
  - Random combinations → <benefit,cost>
  - Seek actions→ high benefit, low cost

Untreated:

Treated (found automatically in 88 secs):



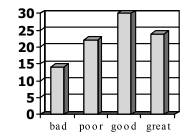
P697=Y
P919=Y
P753=Y *do!*
P692=N *don't!*

Harvesting from seed2: CMM-2

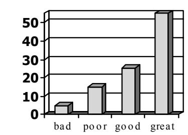
(A) Using ultra-lightweight formal methods such as proposed by Leveson et.al.

(B) sharing requirements documents around the development team in some searchable hypertext format

(C) Build test stubs



stableRequirements if
effectiveReviews @ 0.3
and requirementsUsed @ 0.3
....
and (workProductsIdentified @ 0.3
or ...
or softwareTracking @ 0.3).



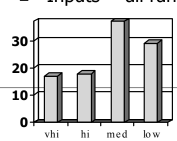
1. **Lower cost of formal reviews at milestones (via A?)**
2. **Do periodic software reviews**
3. **Lower cost of unit testing (via C?)**

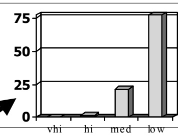
1. **Same**
2. **Not 2**
3. **Lower cost of requirements used (via B?)**

Warning: general conclusions may not apply to specific projects (as we shall see)

Model= COMOCO Risk model

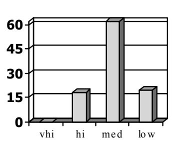
- Inputs = all ranges

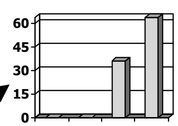




•Sced=4: Time =160% of schedule
•Pcap=4: programmer capability > 90th percentile
•Pmat=2: CMM level 2

- Inputs = KC1

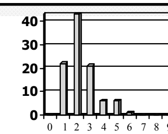




•Sced=2: Time = 100% of schedule
•Acap=2: analyst capability ≈ 55th percentile

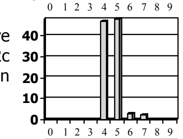
Using TAR2 for acquiring knowledge

- Qualitative constraint model of a circuit :
  - 9 switches, 9 bulbs, 3 batteries
- Many unknowns:
  - E.g. bulbs may be blown/ok
- Score each run by # shining light bulbs (max=9)
- Run I: Find top treatment T learnt from run I. T must be:
  - Acceptable to users
  - And Possible
  - Constrain run I+1 with T
- 1 question culls 90% of options



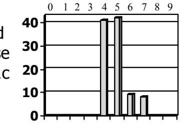
Runs = 35228

Leave Sw2c open



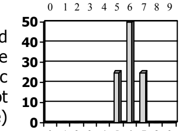
Runs= 3264

And close Sw1c



Runs= 648

And close Sw3c (not done)



Runs = 32

Generality (I): runtimes

		Attributes				
domain	# examples	# continuous	# discrete	# classes	$\lvert lhs \rvert$	run-times (secs)
wine*	178	13	0	3	2	0
housing*	506	13	0	4	2	1
car*	1,728		6	4	2	0
page blocks*	5,473	10	0	5	2	2
circuit+	35,228	0	18	10	4	4
cocomo+	30,000	0	23	4	1	2
satellite+	30,000	0	99	9	5	86
reachness	25,000 250,000	4 4	9 9	4 4	2 1	3 23

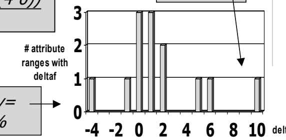
Generality (II)

Delta(outlook.overcast)=10=

$$\frac{\underbrace{(8-2)*(4-0)}_{\text{Lots} \rightarrow \text{none}}+\underbrace{(8-4)*(4-0)}_{\text{Lots} \rightarrow \text{some}}}{(4+0+0)}$$

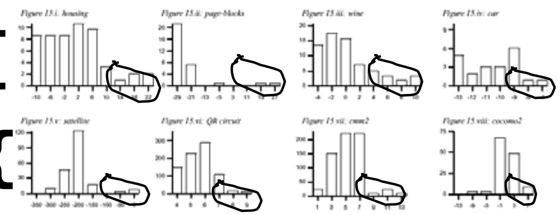
Outlook= overcast

Humidity= 90.. 96%



UC Irvine examples

SE examples



Where do you get it?

Cost = $0

- TAR2:
  - http://www.ece.ubc.ca/twiki/bin/view/Softeng/TreatmentLearner
- APRIORI:
  - http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori/apriori.html#download
- And many other sites with numerous algorithms
  - E.g. http://www.cs.ualberta.ca/~tszhu/softwares/PublicDomain/
  - E.g. http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html
  - E.g. ML++:
    - A public domain "C" library of common algorithms:
    - Naive Bayes, ID3, MC4 , Decision Tables , Holte's OneR , CN2,…
    - http://www.sgi.com/tech/mlc/utils.html
  - E.g. …

Cost > $0

- C4.5:
  - Comes with the book [33]
- C5.0:
  - http://www.rulequest.com/download.html
- Microsoft SQL SERVER 2000™
  - Comes with numerous machine learning tools
  - Proprietary algorithms
- Etc.
  - "data mining consultancy" in Google
  - 850 links.

Summary

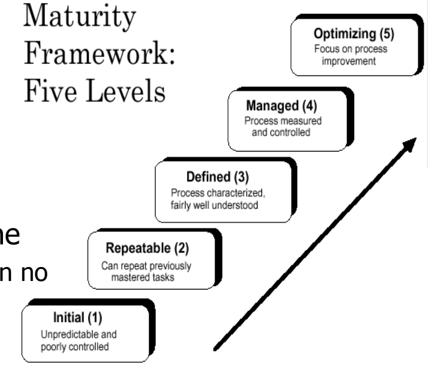
Road map

		available data			
		none	small	medium	huge
pre-existing models	none		C4.5, C5, CART, TAR2		KDD (APRIORI)
	small	Knowledge farming: TAR1, TAR2, Kardio, ESA-auto	Belief networks		
	medium				
	huge				

Knowledge Famines

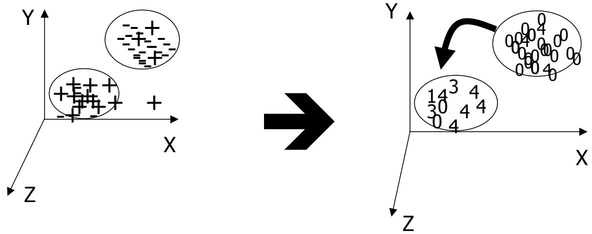
- Most software organizations < CMM2
- The axiom famine
  - If CMM < 3, then no models
- The data famine
  - If CMM < 4 then no meaningful data

Maturity Framework: Five Levels

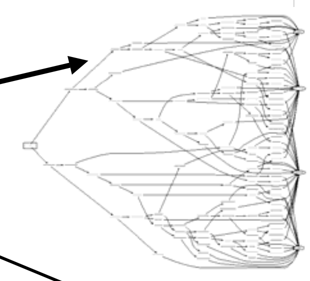


Controllers, not just classifiers

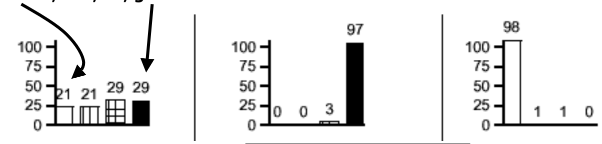
- Software managers want to know "what to change", not just "what is"



Stop staring at the scenery and tell me where to steer or what to dodge



BASELINE	BEST ACTION	WORST ACTION
500 examples of bad--, bad, ok, good	**6.7 <= RM < 9.8 And 12.6 <= Ptratio < 15.9**	**0.6 <= NOX < 1.9 and 17.16 <= LSTAT < 39**



New references

<being the papers I've seen or written since writing "Practical Machine Learning for Software Engineering and Knowledge Engineering">

New (1 of 4)

- {Agrawal93}
  - R.Agrawal and T.Imeilinski and A.Swami **Mining Association Rules between Sets of Items in Large Databases**, Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, USA
- {Bradley98}
  - P. Bradley, U. Fayyad, and C. Reina. **Scaling clustering algorithms to large databases**. In KDD'98.
- {Breiman84}
  - L. Breiman, J. Friedman, R. Olshen, C. Stone, **Classification and Regression Trees**.Wadsworth Int. Group, 1984
- {Cheeseman88}
  - P. Cheeseman, D. Freeman, J. Kelly, M. Self, J. Stutz, and W. Taylor. **Autoclass: a bayesian classification system**. In Proceedings of the Fifth International Conference on Machine Learning. Morgan Kaufmann, 1988
- {Chulani99}
  - S. Chulani and B. Boehm and B. Steece, **Bayesian Analysis of Empirical Software Engineering Cost Models**, IEEE Transaction on Software Engineerining, 25, 4, July/August, 1999

New (2 of 4)

- {Dietterich97}
  - Dietterich, T. G., (1997). **Machine Learning Research: Four Current Directions** *AI Magazine*. 18 (4), 97-136. ftp://ftp.cs.orst.edu/pub/tgd/papers/aimag-survey.ps.gz
- {Fenton00}
  - N. Fenton, M. Neil **Software Metrics: A Roadmap**, ICSE 2000. Available from http://www.dcs.qmul.ac.uk/~norman/papers/metrics_roadmap.pdf
- {Goldberg89}
  - David E. Goldberg. ***Genetic Algorithms*** *in Search*, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts, 1989.

New (3 of 4)

- {Mendonca99}
  - M. Mendonca and N.L. Sunderhaft, **Mining Software Engineering Data: A Survey**, A DACS State-of-the-Art Report. Available from http://www.dacs.dtic.mil/techs/datamining/, September, 1999
- {Menzies01a}
  - T. Menzies and Y. Hu, **Reusing models for requirements engineering**, First International Workshop on Model-based Requirements Engineering, 2001,Available from http://tim.menzies.com/pdf/01reusere.pdf
- {Menzies01b}
  - T. Menzies and Y. Hu, **Constraining discussions in requirements engineering**, First International Workshop on Model-based Requirements Engineering, 2001,Available from http://tim.menzies.com/pdf/01lesstalk.pdf
- {Menzies01c}
  - T. Menzies and J. Kiper, **Better reasoning about software engineering activities,** Automated Software Engineering, 2001,Available from http://tim.menzies.com/pdf/01ml4re.pdf

New (4 of 4)

- {Poole98}
  - D. L. Poole, A. K. Mackworth, and R. G. Goebel. **Computational Intelligence: A Logical Approach**. Oxford University Press, New York, 1998
- {Webb00}
  - **Efficient search for association rules**, G. Webb, Proceeding of KDD-2000 Boston, MA, 2000,