Introduction to

Information Retrieval

CS276: Information Retrieval and Web Search

Christopher Manning and Pandu Nayak

Lecture 13: Decision trees and machine learning on documents

Some slides borrowed from John Canny

Text classification

- Last lecture: Basic algorithms for text classification
  - Naive Bayes classifier
    - Simple, cheap, high bias, linear
  - K Nearest Neighbor classification
    - Simple, expensive at test time, high variance, non-linear
  - Vector space classification: Rocchio
    - Simple linear discriminant classifier; perhaps too simple*
- Today
  - Decision trees
  - Some empirical evaluation and comparison
  - Decision tree ensembles
    - Will lead into using tree-based methods (GBRT) for ranking
  - Text-specific issues in classification

Text Classification Evaluation: Classic Reuters-21578 Data Set

- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Reuters Text Categorization data set (**Reuters-21578)** document

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

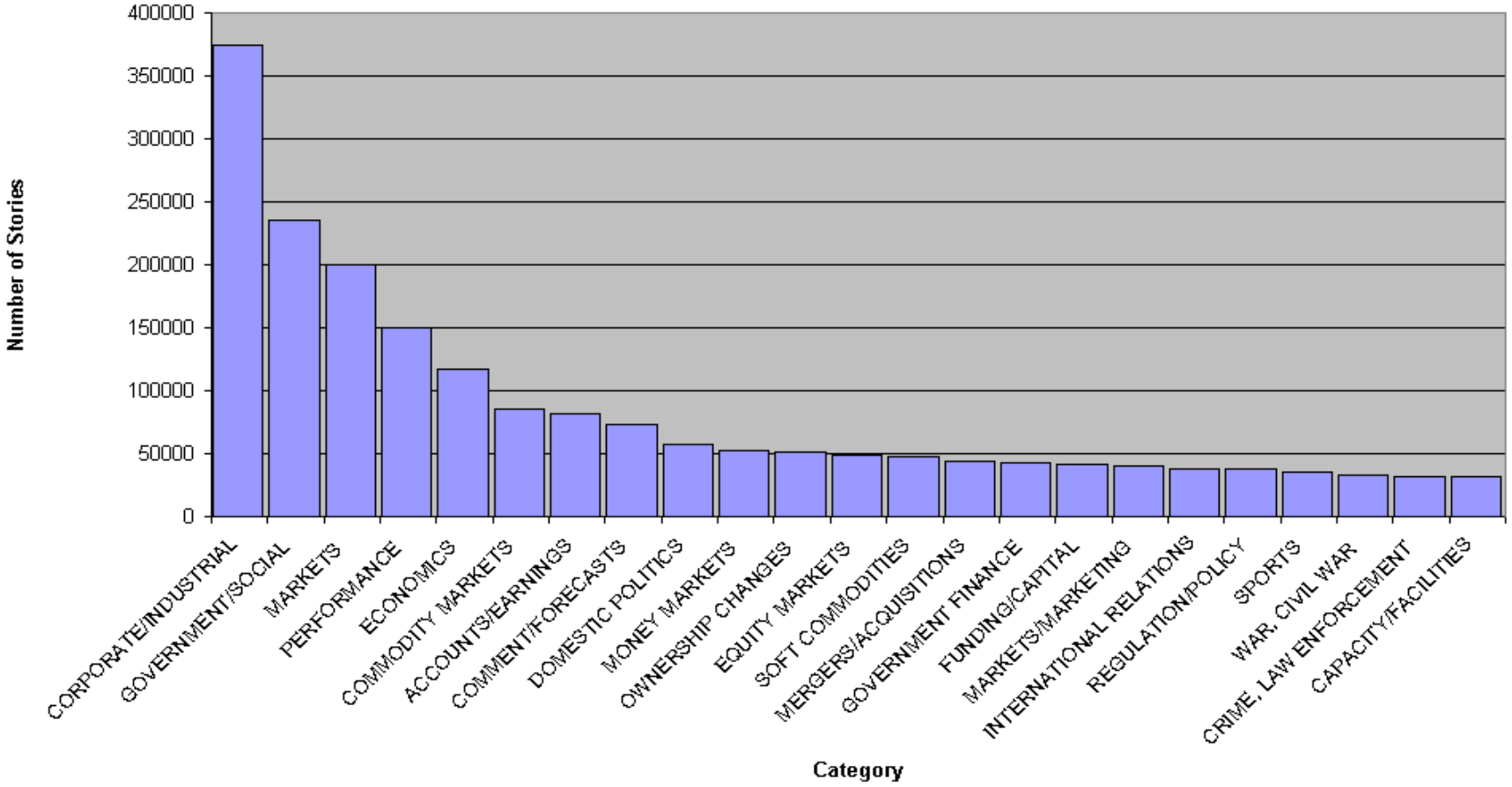
Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

&#3;</BODY></TEXT></REUTERS>

Newer Reuters data: RCV1: 810,000 docs

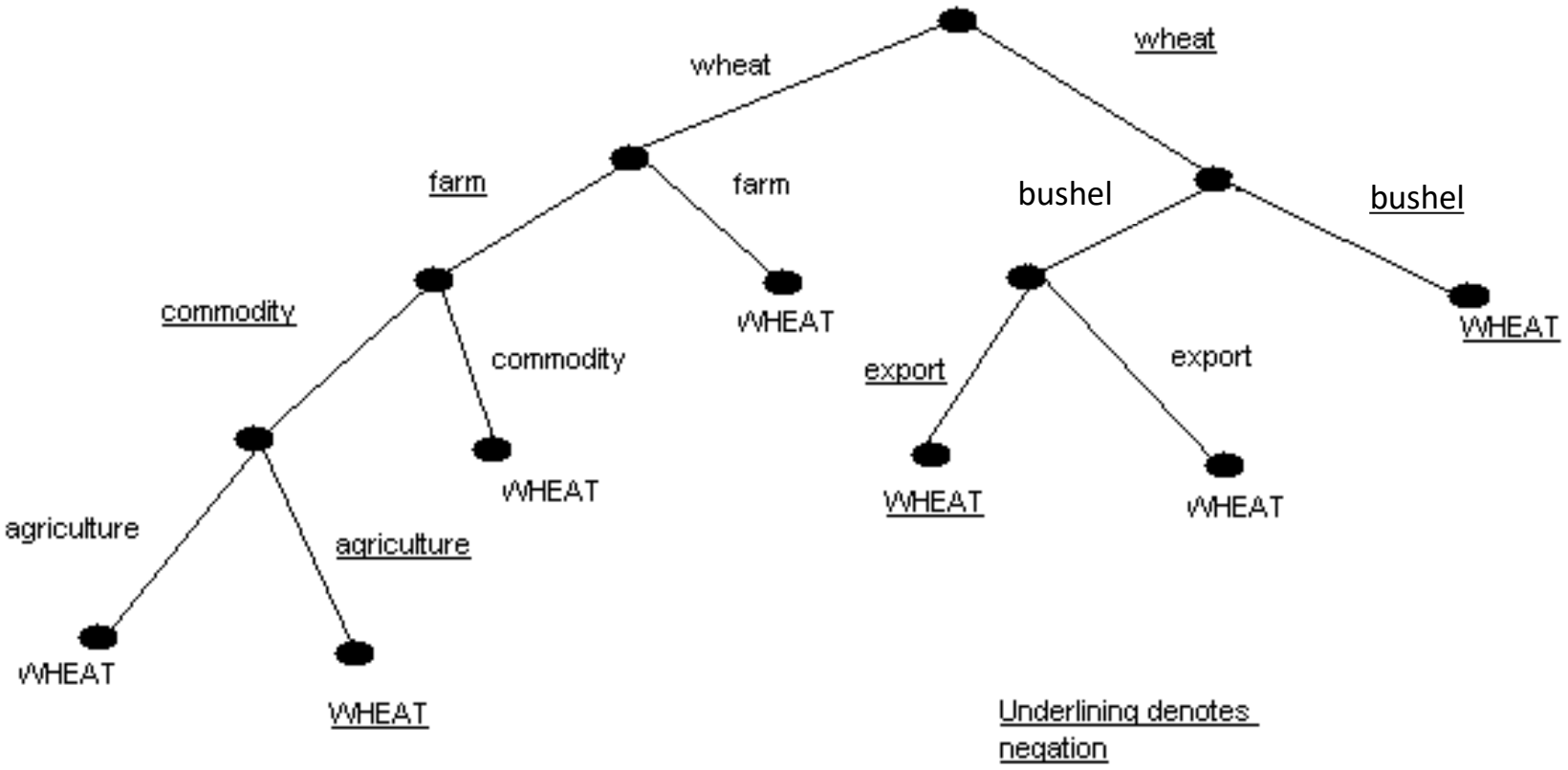
- Top topics in Reuters RCV1



Decision Trees for text classification

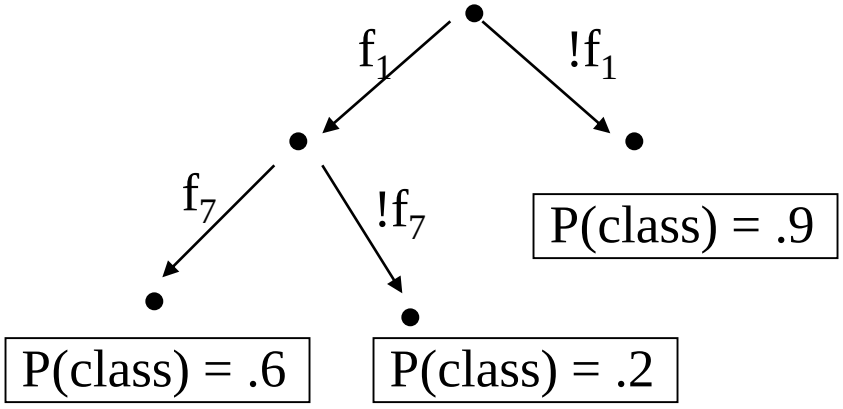
- A tree with internal nodes labeled by terms
- Branches are labeled by tests on the weight that the term has (or just presence/absence)
- Leaves are labeled by categories
- Classifier categorizes document by descending tree following tests to leaf
- The label of the leaf node is then assigned to the document
- Most decision trees are binary trees (never disadvantageous; may require extra internal nodes)

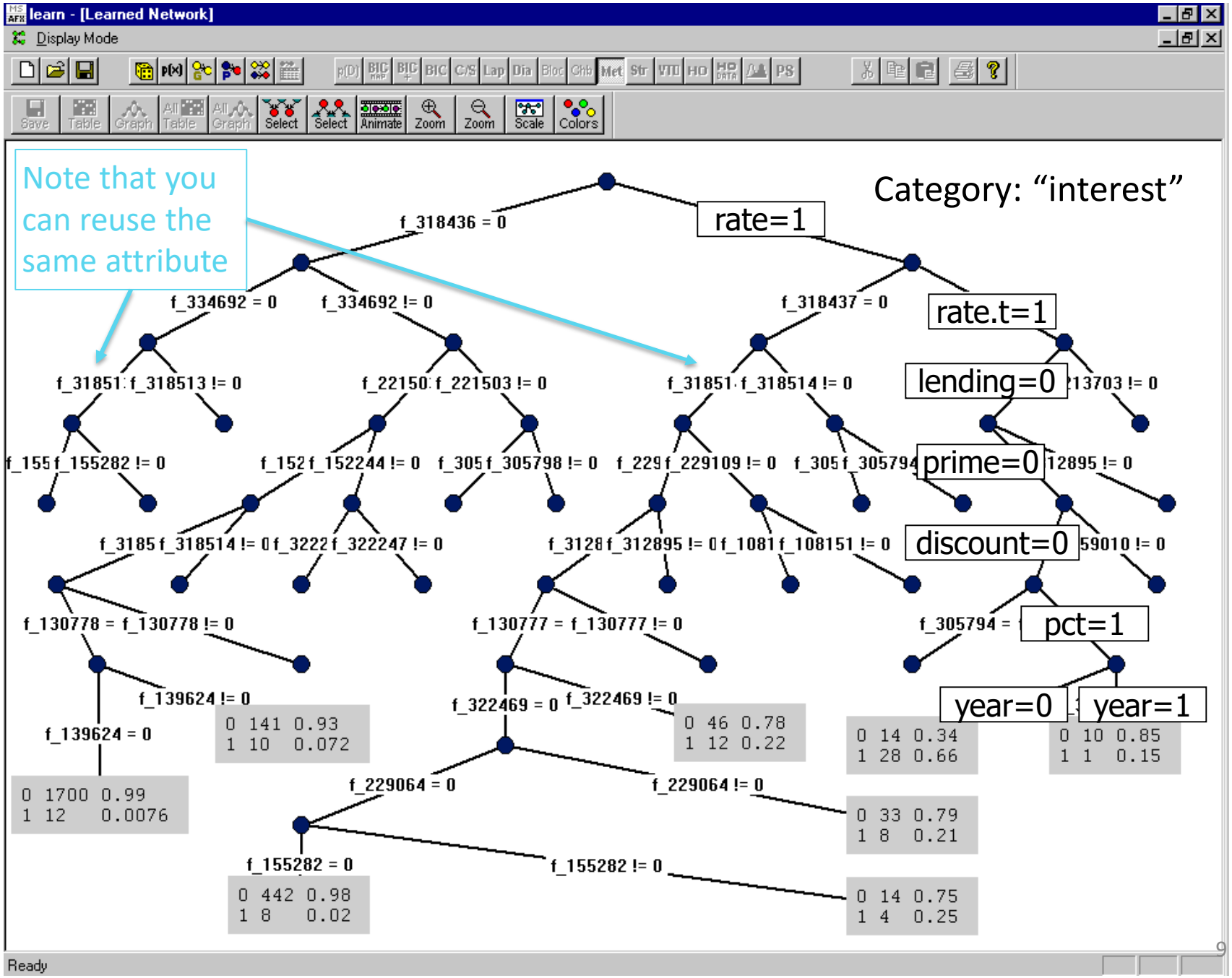
Decision Tree Example



Decision Tree Learning

- Learn a sequence of tests on features, typically using top-down, greedy search
  - At each stage choose unused feature with highest Information Gain
- At leaves, either categorical (yes/no) or continuous decisions



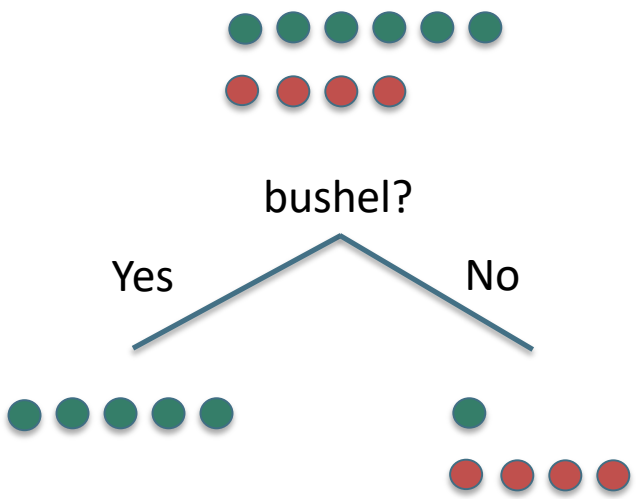

learn - [Learned Network]
Display Mode
Save
Table
Graph
All Table
All Graph
Select
Select
Animate
Zoom
Zoom
Scale
Colors
Note that you can reuse the same attribute
Category: "interest"
rate=1
f_318436 = 0
f_318437 = 0
rate.t=1
f_334692 = 0
f_334692 != 0
lending=0
prime=0
discount=0
pct=1
year=0
year=1
f_130778 = f_130778 != 0
f_130777 = f_130777 != 0
f_139624 != 0
f_139624 = 0
f_322469 = 0
f_322469 != 0
f_229064 = 0
f_229064 != 0
f_155282 = 0
f_155282 != 0
0 141 0.93
1 10 0.072
0 1700 0.99
1 12 0.0076
0 46 0.78
1 12 0.22
0 14 0.34
1 28 0.66
0 10 0.85
1 1 0.15
0 33 0.79
1 8 0.21
0 442 0.98
1 8 0.02
0 14 0.75
1 4 0.25
Ready

Decision tree learning

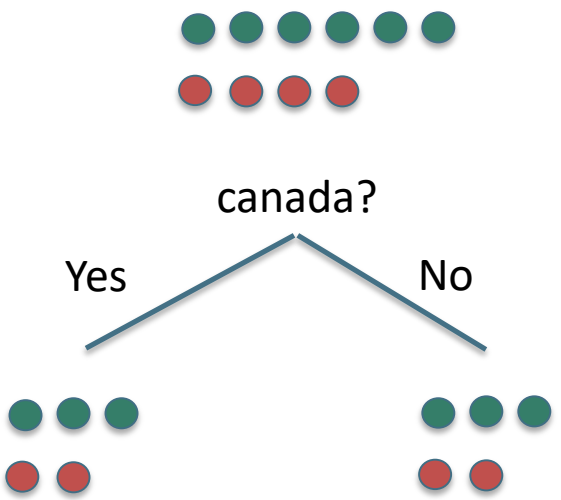
- If there are k features, a decision tree might have up to 2^k nodes. This is almost always much too big!
- We want to find "efficient" (small but effective) trees.
- We can do this in a **greedy manner by recursively choosing a best split feature at each node**.

Choosing an attribute

- Idea: a good features splits the examples into subsets that are (ideally) "all positive" or "all negative"
  - This is binary case. Same idea and method works for n-ary case except for a bias towards many valued features.



A good feature



A bad feature: Wheat is still 60%

Using Information Theory

Entropy is defined at each node based on the class breakdown:

- Let p_i be the fraction of examples in class *i*.
- Let p_i^f be the fraction of elements with feature *f* that lie in class *i*.
- Let $p_i^{\neg f}$ be the fraction of elements without feature *f* that lie in class *i*

Finally let p^f and $p^{\neg f}$ be the fraction of nodes with (respectively without) feature *f*

Information Gain

Before the split by *f*, entropy is

$$E = -\sum_{i=1}^{m} p_i \log p_i$$

After split by f, the entropy is

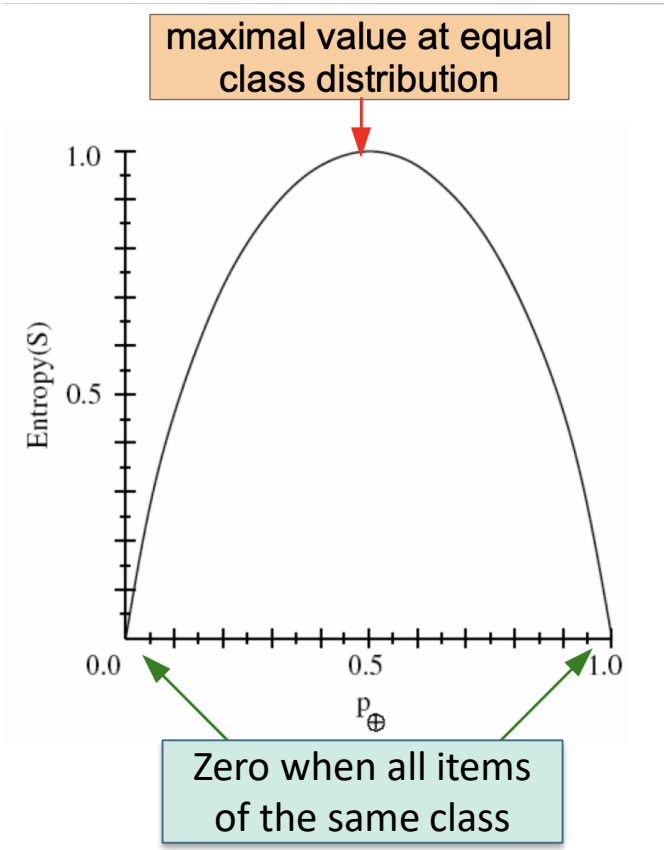
$$E_f = -p^f \sum_{i=1}^{m} p_i^f \log p_i^f - p^{\neg f} \sum_{i=1}^{m} p_i^{\neg f} \log p_i^{\neg f}$$

The information gain = $E - E_f$ (information = – entropy)

Using Information Theory

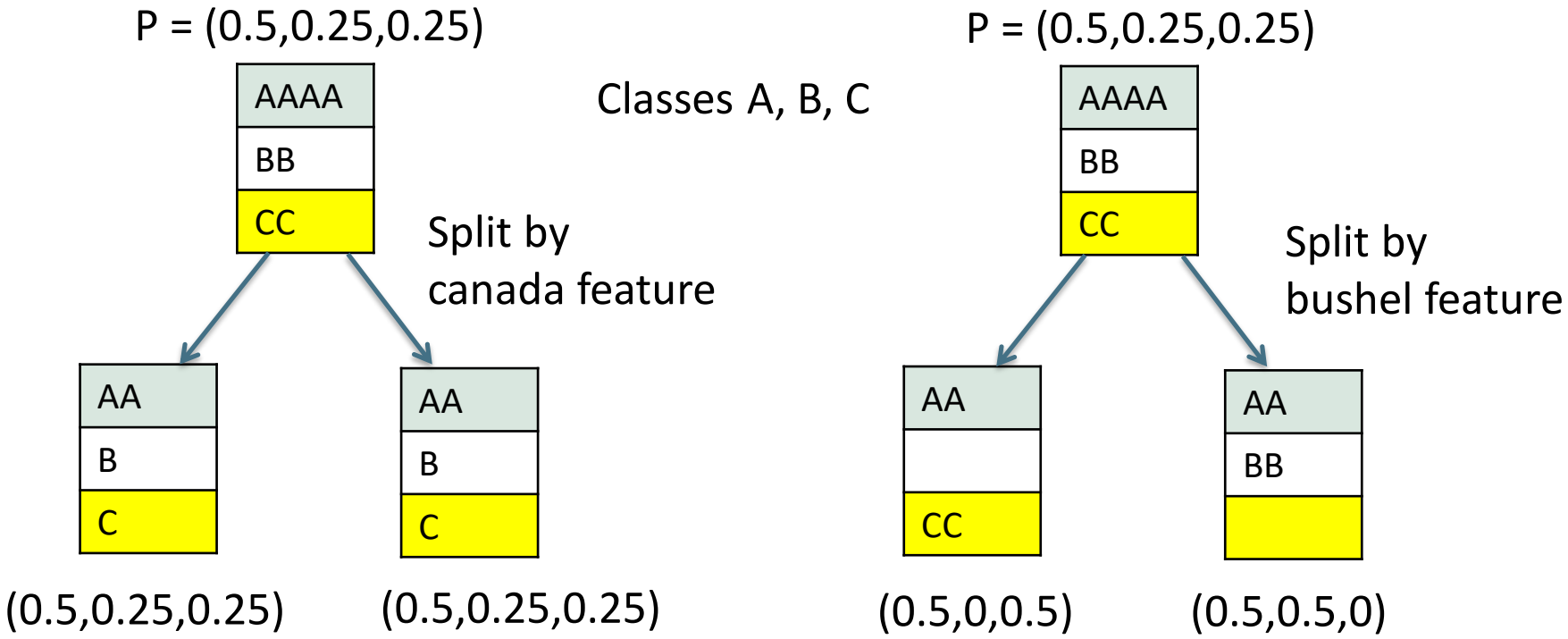
Entropy: amount of uncertainty in class distribution

Two class case:



Example

$$E = -\sum_{i=1}^{m} p_i \log p_i$$



$E = -\sum p_i \log p_i$ =

0.5*1+0.25*2+0.25*2 = 1.5 bits

After:

E_f = (0.5+0.5)*1.5= 1.5 bits

No gain!

Before: E = 1.5 bits

After:

E_f = (0.5+0.5)*1 bits = 1 bits

Gain = $E - E_f$ = 0.5 bits

Choosing best features

At each node, we choose the feature *f* which **maximizes the information gain**.

This tends to be produce mixtures of classes at each node that **are more and more "pure"** as you go down the tree.

If a node has examples all of one class c, we make it a leaf and output "c". Otherwise, we potentially continue to build

If a leaf still has a mixed distribution, we output **the most popular class** at that node.

Numeric features (e.g., tf-idf, etc.)

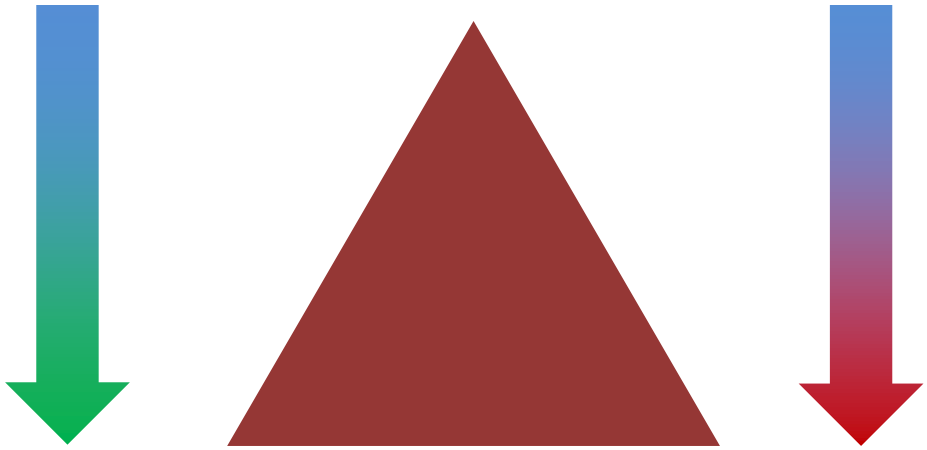
- Commonly make a binary split (f < t), but where?
- Exhaustively: evaluate each split point between observed values for information gain.
  - Slow.
  - Can be made a bit more efficient by optimizing counting
- Discretize into bins
  - Divide all numeric values into *k* bins.
  - Feature is treated as if categorical
  - Binning can be based on statistics of the entire dataset
    - For instance one might use *k*-means clustering on values of feature

When to stop?

- When all the examples at a node are of the same class
- When a fixed tree depth *d* is reached
- When there isn't an attribute you can split on where the split differentiates classes with statistical significance (e.g., with chi-square or Fisher's Exact)
- Commonest/best: Use separate validation data
  - Grow a big tree (perhaps with depth threshold)
  - Prune nodes bottom up that fail to (significantly) improve classification performance on the validation data

Decision Tree Models

- As tree depth increases, bias decreases and variance generally increases. Why? (Hint: think about k-NN)



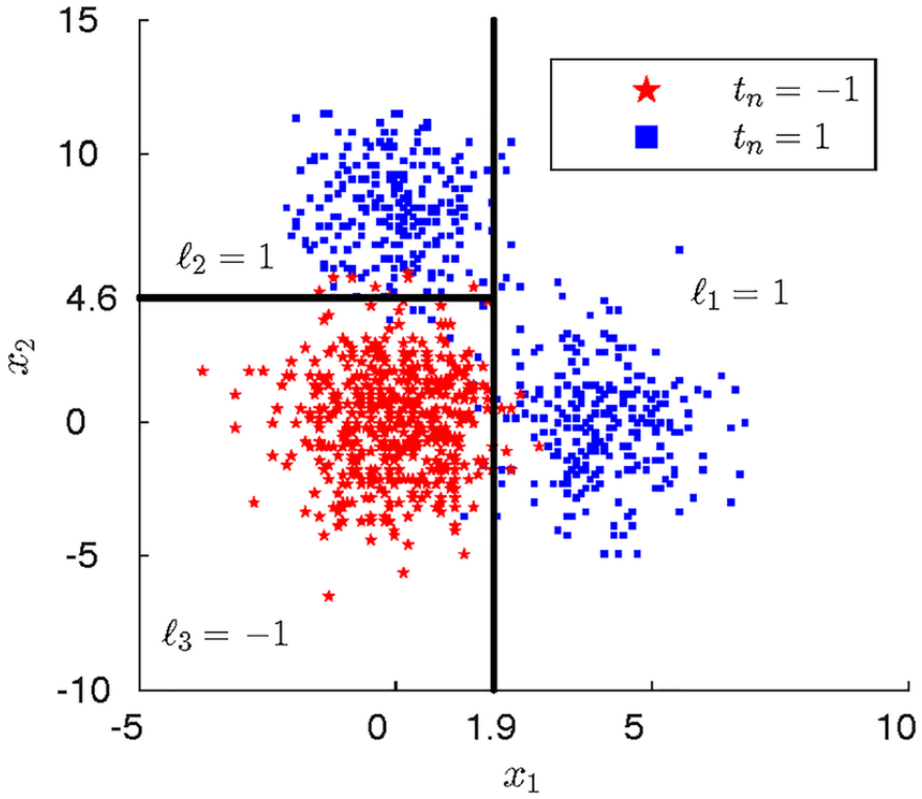
Bias decreases with tree depth

Variance increases with tree depth

- Some of the stuff fit deep in a tree is fairly random facts and associations in the training data

Decision trees classify via rectangular regions

The (rectangular) regions are aligned with feature axes
The model cannot learn arbitrary linear decision boundaries
Overall, the result is a non-linear classifier



Decision Tree Learning for Text

- Most people's intuitions are that text has many words and you have a lot of weak-evidence features
  - Hence, use of a small number of feature tests is potentially bad for text classification
  - But in fact the method can sometime do pretty well – such as for the Reuters dataset. Topics can have marker words.
- Decision trees are easily interpreted by humans – much more easily than methods like Naive Bayes
  - You can extract rules from decision trees, in fact.

Text classification
Per class evaluation measures

- Recall: Fraction of docs in class *i* classified correctly:

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

- Precision: Fraction of docs assigned class *i* that are actually about class *i*:

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

- Accuracy: (1 - error rate) Fraction of docs classified correctly:

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average.
- Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

Micro- vs. Macro-Averaging: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83

- Microaveraged score is dominated by score on common classes

Dumais et al. 1998: Reuters – Break-even F1

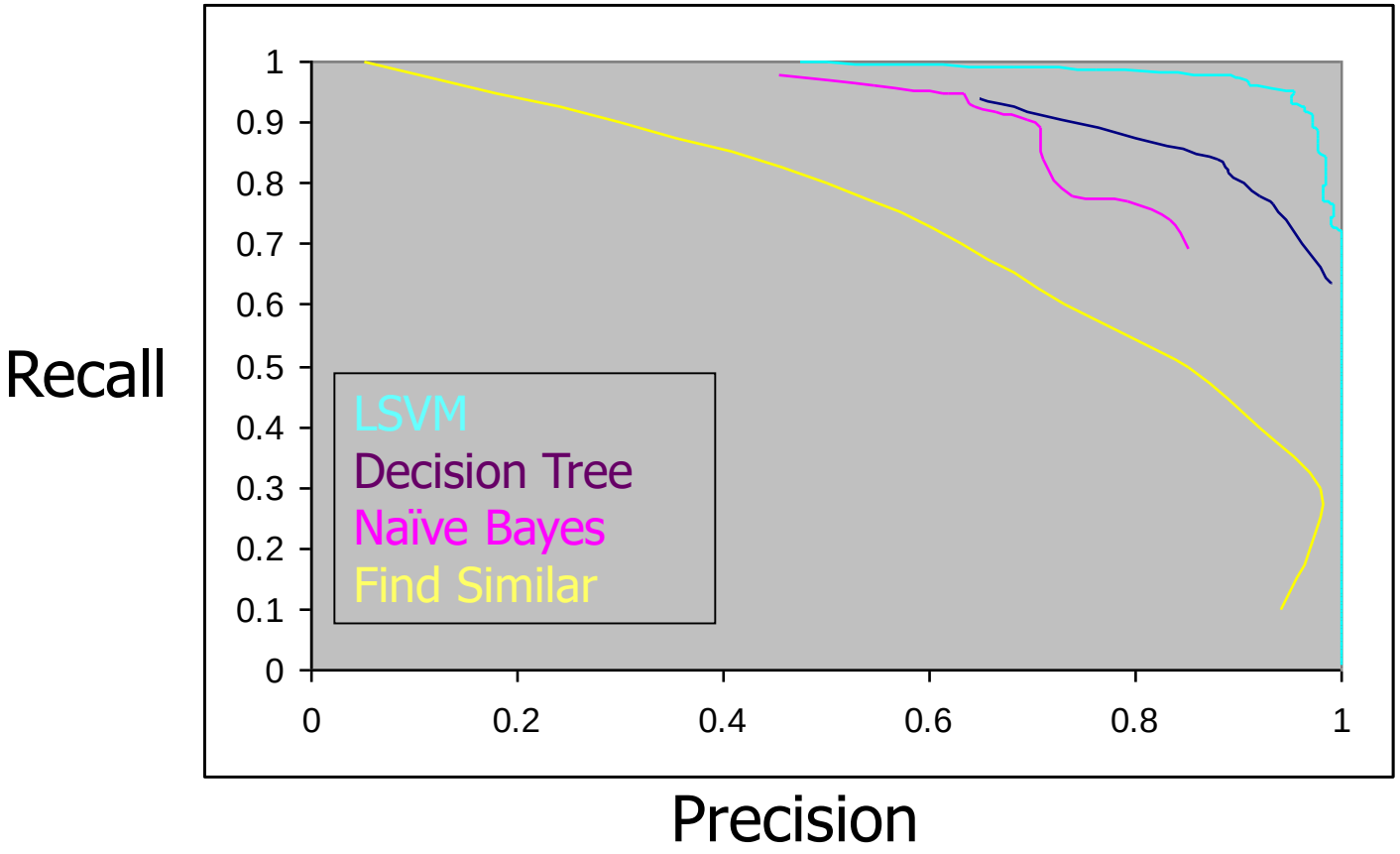
	Findsim	NBayes	BayesNets	Trees	LinearSVM
earn	92.9%	95.9%	95.8%	97.8%	98.2%
acq	64.7%	87.8%	88.3%	89.7%	92.8%
money-fx	46.7%	56.6%	58.8%	66.2%	74.0%
grain	67.5%	78.8%	81.4%	85.0%	92.4%
crude	70.1%	79.5%	79.6%	85.0%	88.3%
trade	65.1%	63.9%	69.0%	72.5%	73.5%
interest	63.4%	64.9%	71.3%	67.1%	76.3%
ship	49.2%	85.4%	84.4%	74.2%	78.0%
wheat	68.9%	69.7%	82.7%	92.5%	89.7%
corn	48.2%	65.3%	76.4%	91.8%	91.1%
Micro					
Avg Top 10	64.6%	81.5%	85.0%	88.4%	91.4%
Avg All Cat	61.7%	75.2%	80.0%	na	86.4%

Recall: % labeled in category among those stories that are really in category

Precision: % really in category among those stories labeled in category

Break Even: When recall equals precision

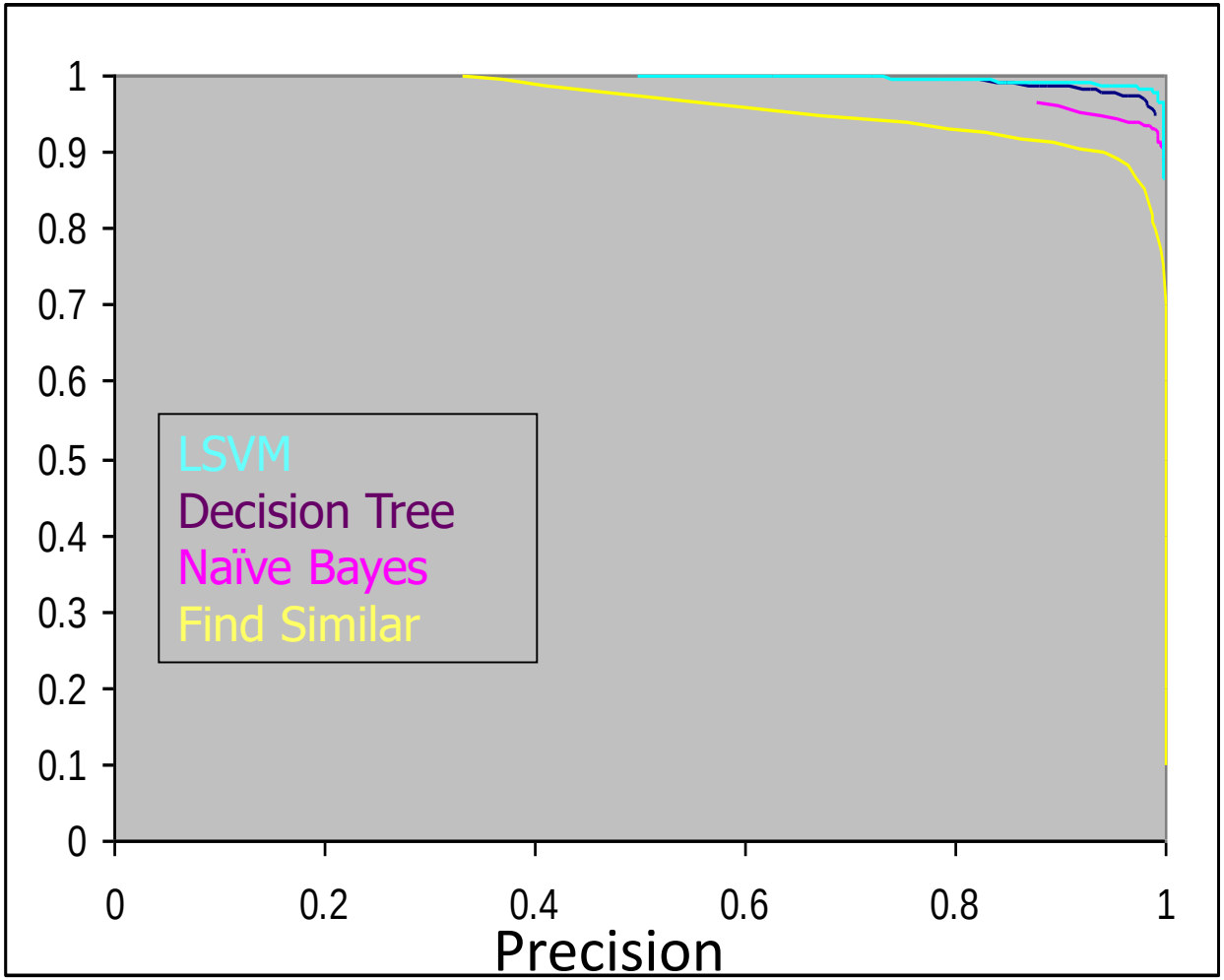
Reuters ROC - Category Grain



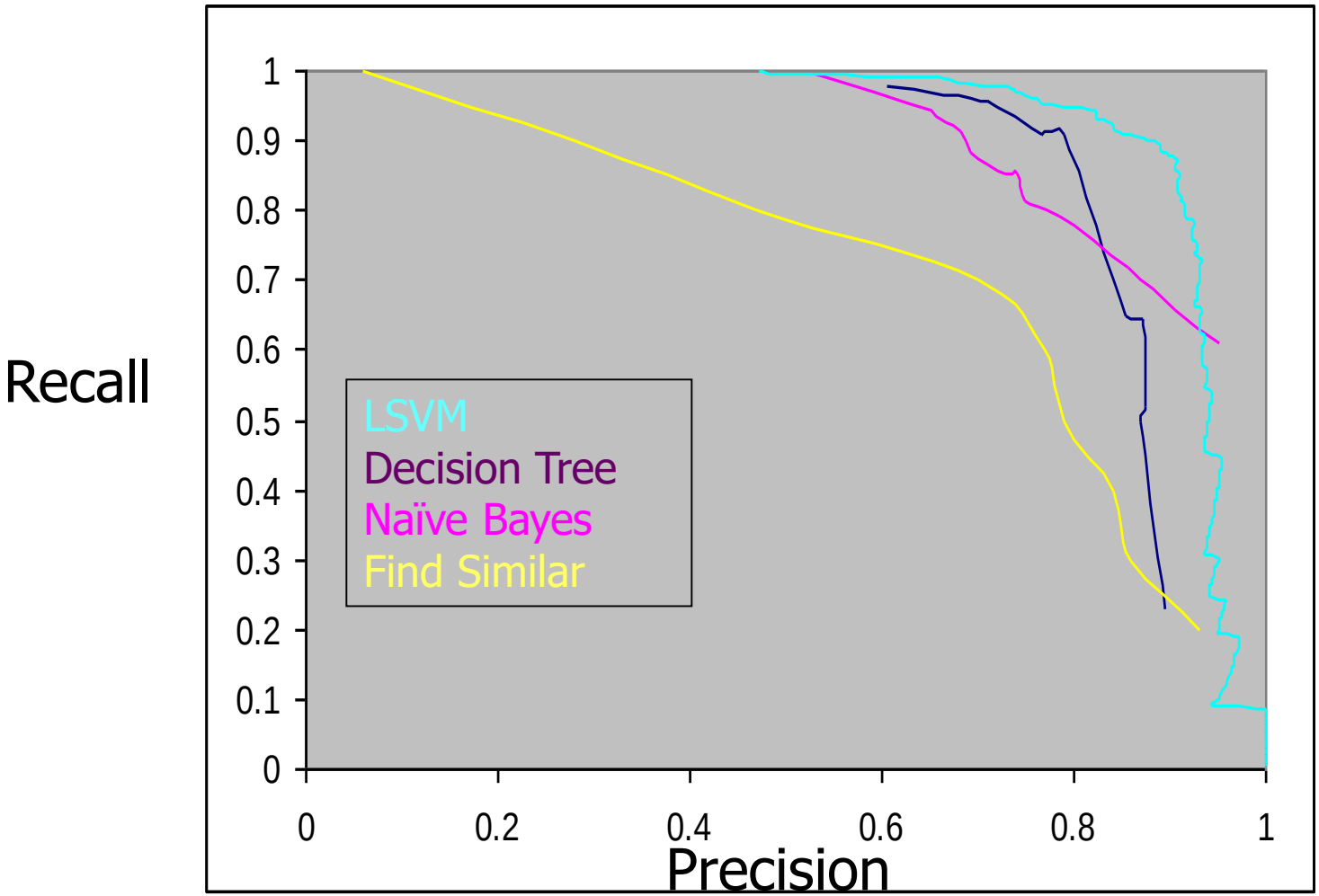
Recall: % labeled in category among those stories that are really in category

Precision: % really in category among those stories labeled in category

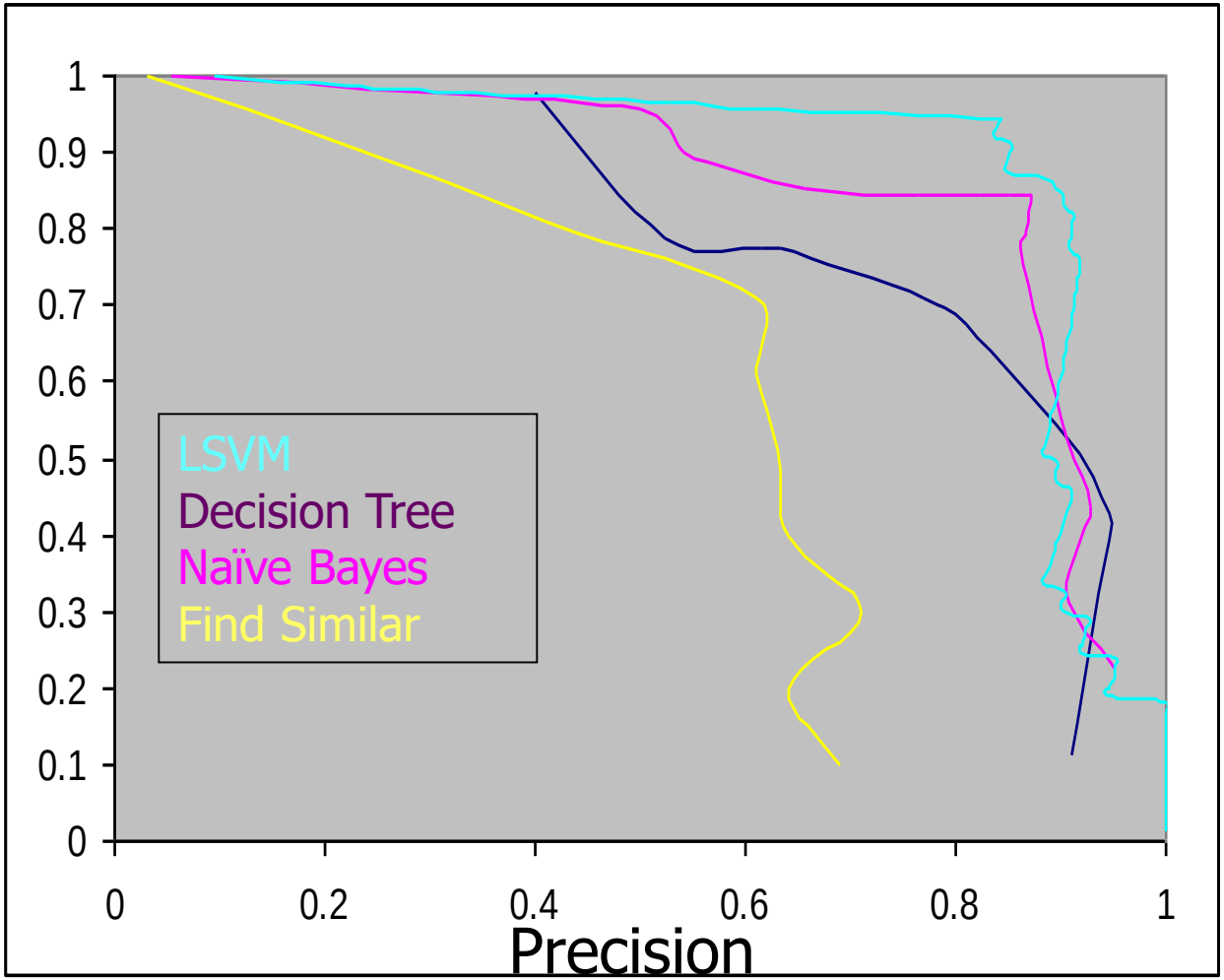
ROC for Category - Earn



ROC for Category - Crude



ROC for Category - Ship



(a)

	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(b)

	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

Evaluation measure: F_1

The discriminative alternative: Logistic Regression and Support vector machines

- Directly predict class conditional on words:
- (Binary) Logistic Regression:

$$\log \frac{P(C \mid d)}{P(\overline{C} \mid d)} = \alpha + \sum_{w \in d} \beta_w \times w$$

- Tune parameters β_w to optimize conditional likelihood or "margin" (SVM) in predicting classes
- What a statistician would probably tell you to use if you said you had a categorical decision problem (like text categorization)

LogR/SVM Performance

- Early results with LogR were disappointing, because people didn't understand the means to *regularize* (smooth) LogR to cope with **sparse textual features**
- Done right, LogR clearly outperforms NB in text categorization and batch filtering studies
- SVMs were seen as the best general text classification method in the period c. 1997– 2005
- LogR seems as good as SVMs (Tong & Oles 2001)
- But now challenged by:
  - Neural net methods (improve word similarity models)
  - Ensemble methods, e.g. **random forests, boosting**

Ensemble Methods

Are like **Crowdsourced machine learning algorithms**:

- Take a collection of simple or *weak* learners
- Combine their results to make a single, better learner

Types:

- **Bagging:** train learners in parallel on different samples of the data, then combine by voting (discrete output) or by averaging (continuous output).
- **Stacking:** feed output of first-level model(s) as features into a second-stage learner like logistic regression.
- **Boosting:** train subsequent learners on the filtered/weighted output of earlier learners so they fix the stuff that the earlier learners got wrong

Random Forests

Grow K trees on datasets sampled from the original dataset with replacement (bootstrap samples), p = number of features.

- Draw K bootstrap samples of size N (size of original dataset)
- Grow each Decision Tree, by selecting a random set of m out of p features at each node, and choosing the best feature to split on.
  - Typically m might be e.g. sqrt(p)
- **Runtime:** Aggregate the predictions of the trees (most popular vote) to produce the final class.

Random Forests

Principles: we want to take a vote between different learners so we don't want the models to be too similar. These two criteria ensure diversity in the individual trees:

- Data bagging: Draw K bootstrap samples of size N:
  - Each tree is trained on different data.
- Feature bagging: Grow a Decision Tree, by selecting a random set of m out of p features at each node, and choosing the best feature to split on.
  - Corresponding nodes in different trees (usually) can't use the same feature to split.

Random Forests

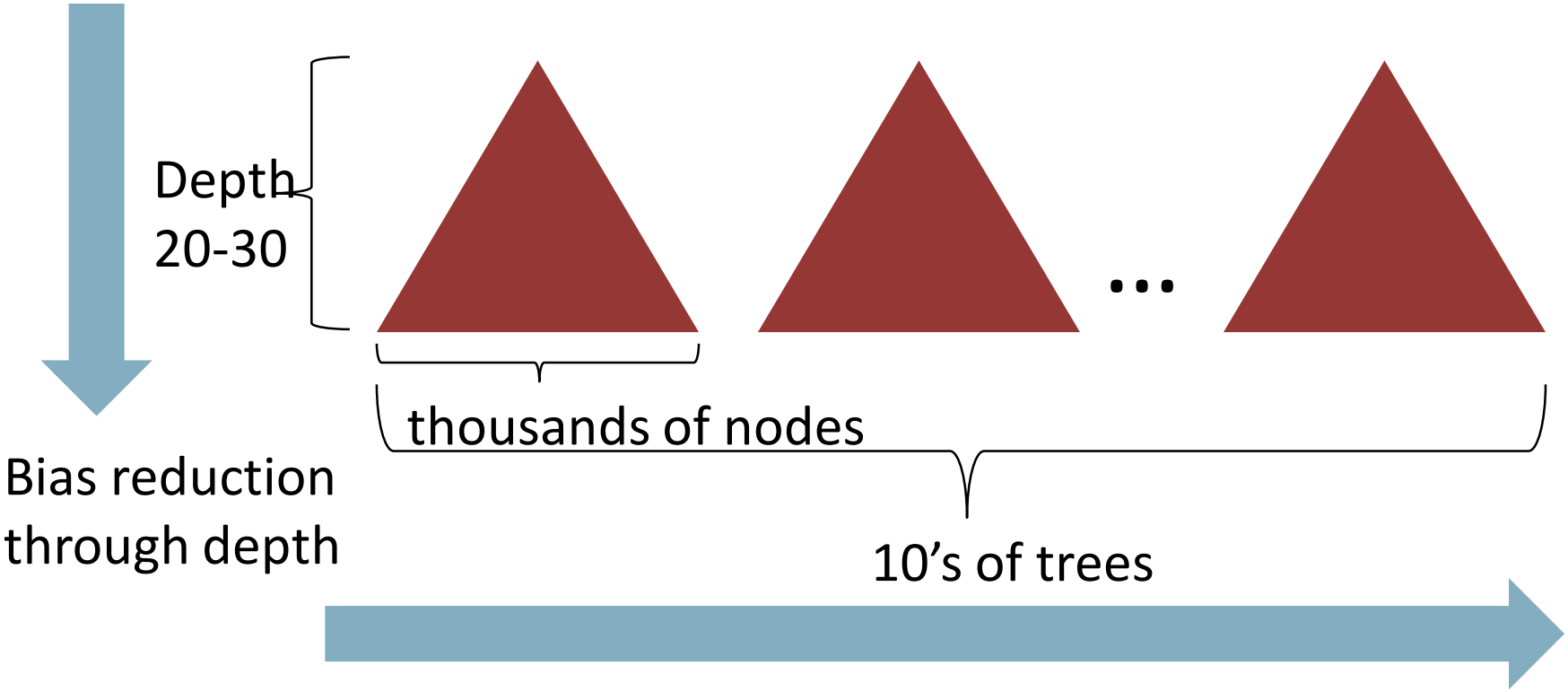
- **Very popular in practice,** at one point the most popular classifier for dense data (<= a few thousand features)
- **Easy to implement** (train a lot of trees).
- **Parallelizes easily** (but not necessarily efficiently). Good match for MapReduce.
- **Now not quite state-of-the-art accuracy** – Gradient-boosted trees (less features) and Deep NNs (vision, speech, language, …) generally do better
- **Needs many passes over the data** – at least the max depth of the trees. (<< boosted trees though)
- **Easy to overfit** – need to balance accuracy/fit tradeoff.

Boosted Decision Trees

- A more recent alternative to random Forests
- In contrast to RFs whose trees are trained **independently**, BDT trees are trained **sequentially** by **boosting**:
  - Each successive tree is trained on weighted data which emphasizes instances incorrectly labeled by the previous trees.
- Both methods can produce very high-quality models
- But boosted decision trees are now normally the method of choice for datasets with a medium number of features

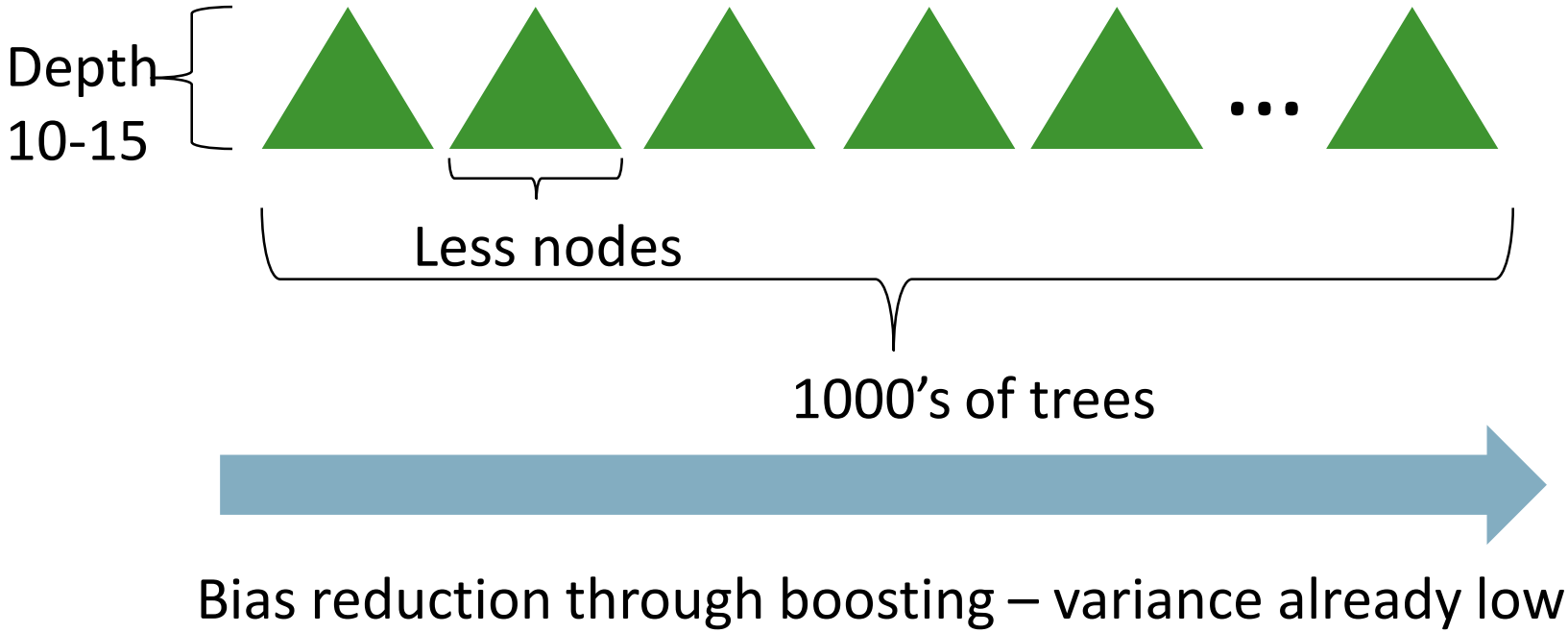
Random Forests vs Boosted Trees

- The "geometry" of the methods is very different:
- Random forest use 10's of deep, large trees:



Random Forests vs Boosted Trees

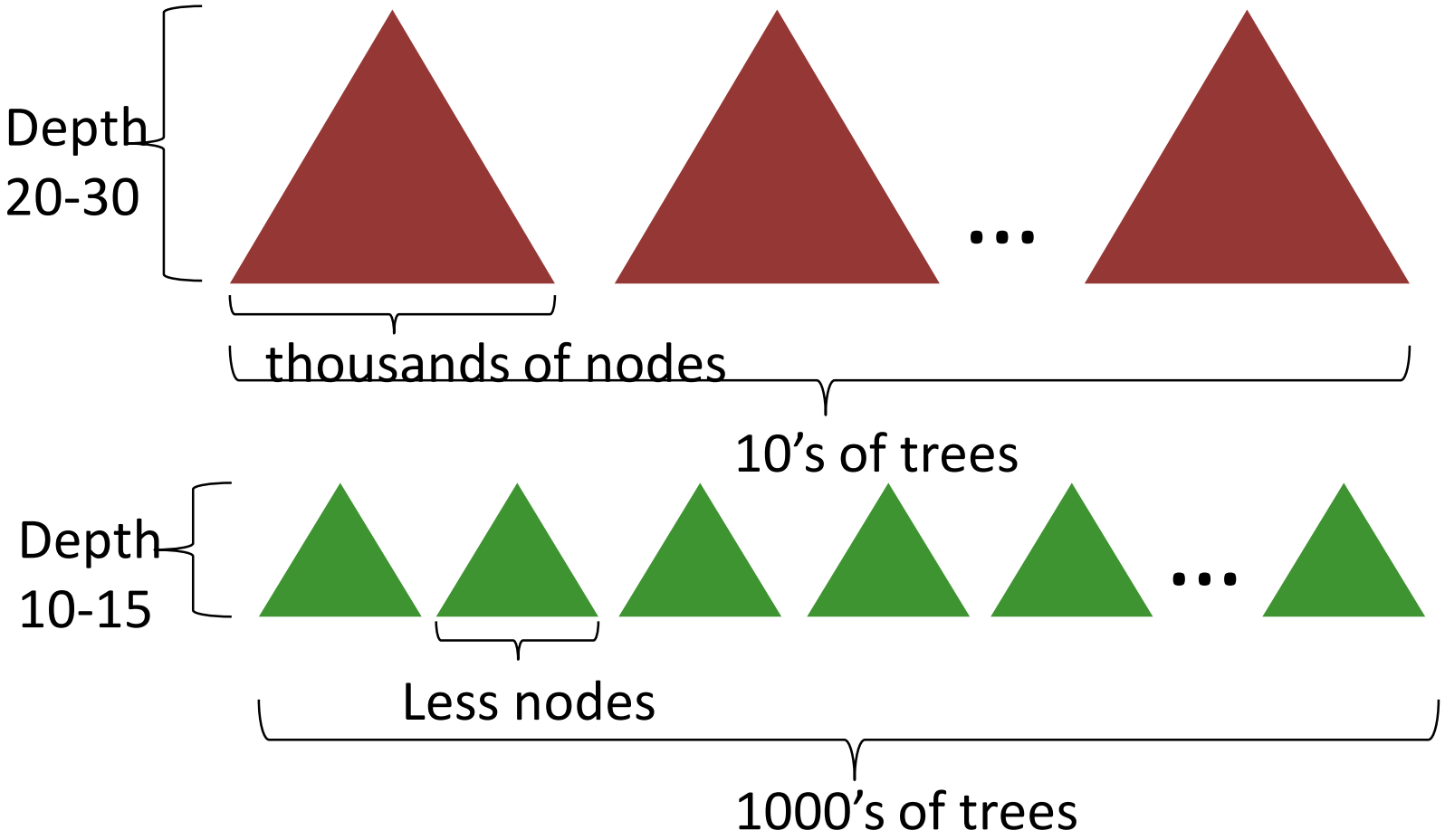
- The "geometry" of the methods is very different:
- Boosted decision trees use 1000's of shallow, small trees:



Bias reduction through boosting – variance already low

Random Forests vs Boosted Trees

- RF training embarrassingly parallel, can be very fast
- Evaluation of trees (runtime) also much faster for RFs



The Real World

P. Jackson and I. Moulinier. 2002. *Natural Language Processing for Online Applications*

- "There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate intranets, government departments, and Internet publishers"

- "Understanding the data is one of the keys to successful categorization, yet this is an area in which most categorization tool vendors are extremely weak. Many of the 'one size fits all' tools on the market have not been tested on a wide range of content types."

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Manually written rules

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - Construe: 94% recall, 84% precision over 675 categories (Hayes and Weinstein IAAI 1990)
- Amount of work required is huge
  - Estimate 2 days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

- Perfect!
- We can use all our clever classifiers
- Roll out logistic regression/SVMs/random forests!

- But if you are using an SVM/NB etc., you should probably be prepared with the "hybrid" solution where there is a Boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately

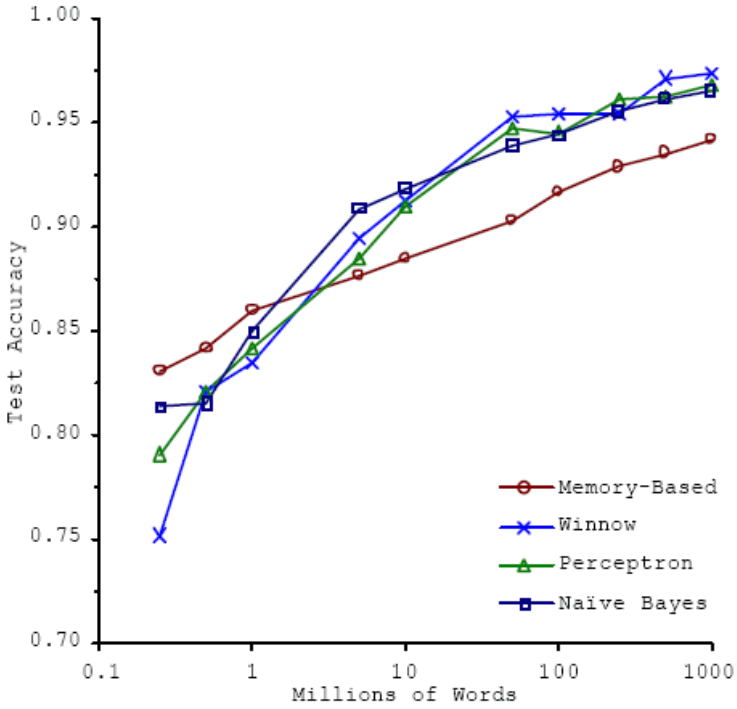
A huge amount of data?

- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are less practical

- Naïve Bayes can come back into its own again!
  - Or other methods with linear training/test complexity like **(regularized) logistic regression** (though much more expensive to train)

Accuracy as a function of data size

- With enough data the choice of classifier may not matter much, and the best choice may be unclear
  - Data: Brill and Banko on context-sensitive spelling correction
- But the fact that you have to keep doubling your data to improve performance is a little unpleasant

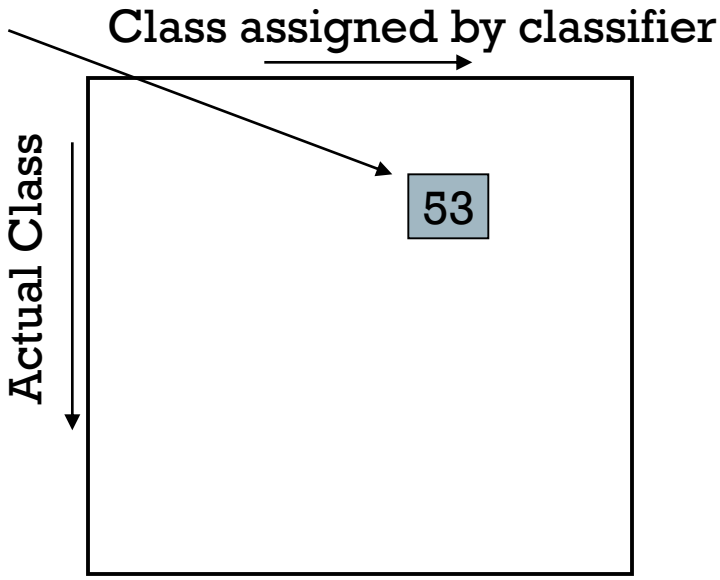


How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Classifier combination is always a useful technique
      - Voting, bagging, or boosting multiple classifiers
    - Much literature on hierarchical classification
      - Mileage fairly unclear, but helps a bit (Tie-Yan Liu et al. 2005)
      - **Definitely helps for scalability**, even if not in accuracy
    - May need a hybrid automatic/manual solution

Good practice department: Make a confusion matrix

This (*i*, *j*) entry means 53 of the docs actually in class *i* were put in class *j* by the classifier.



- In a perfect classification, only the diagonal has non-zero entries
- Look at common confusions and how they might be addressed

Good practice department: N-Fold Cross-Validation

- Results can vary based on sampling error due to different training and test sets.
- Average results over multiple training and test sets (splits of the overall data) for the best results.
- Ideally, test and training sets are independent on each trial.
  - But this would require too much labeled data.
- Partition data into N equal-sized disjoint segments.
- Run N trials, each time using a different segment of the data for testing, and training on the remaining N–1 segments.
- This way, at least test-sets are independent.
- Report average classification accuracy over the N trials.
- Typically, N = 10.

Good practice department: Learning Curves

- In practice, labeled data is usually rare and expensive.
- Would like to know how performance varies with the number of training instances.
- *Learning curves* plot classification accuracy on independent test data (*Y* axis) versus number of training examples (*X* axis).
- One can do both the above and produce learning curves averaged over multiple trials from cross-validation

How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - E.g., an author byline or mail headers
- Aim to collapse things that would be treated as different but shouldn't be.
  - E.g., part numbers, chemical formulas

- Does putting in "hacks" help?
  - You bet! Easiest way to improve practical systems
    - Feature design and non-linear weighting is *very* important in the performance of real-world systems

Upweighting

- You can get a lot of value by differentially weighting contributions from different document zones:
- That is, you count as two instances of a word when you see the word in, say, the abstract
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
    - Like what we talked about for BM25F
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al,* 2002)

Two techniques for zones

1. Have a completely separate set of features/parameters for different zones like the title
2. Use the same features (pooling/tying their parameters) across zones, but upweight the contribution of different zones

- Commonly the second method is more successful: it costs you nothing in terms of sparsifying the data, but can give a very useful performance boost
  - Which is best is a contingent fact about the data

Text Summarization techniques in text classification

- Text Summarization: Process of extracting key pieces from text, normally by features on sentences reflecting position and content
- Much of this work can be used to suggest weightings for terms in text categorization
  - See: Kolcz, Prabakarmurthi, and Kalita, CIKM 2001: Summarization as feature selection for text categorization
  - Categorizing with title,
  - Categorizing with first paragraph only
  - Categorizing with paragraph with most keywords
  - Categorizing with first and last paragraphs, etc.

Does stemming/lowercasing/… help?

- As always, it's hard to tell, and empirical evaluation is normally the gold standard
- But note that the role of tools like stemming is rather different for TextCat vs. IR:
  - For IR, you often want to collapse forms of the verb *oxygenate* and *oxygenation*, since all of those documents will be relevant to a query for *oxygenation*
  - For TextCat, with sufficient training data, stemming *does no good.* It only helps in compensating for data sparseness (which can be severe in TextCat applications). *Overly aggressive stemming can easily degrade performance.*

Measuring Classification Figures of Merit

- Accuracy of classification
  - Main evaluation criterion in academia
- Speed of training statistical classifier
  - Some methods are very cheap; some very costly
- Speed of classification (docs/hour)
  - No big differences for most algorithms
  - Exceptions: kNN, complex preprocessing requirements
- Effort in creating training set/hand-built classifier
  - human hours/topic

Measuring Classification Figures of Merit

- Not just accuracy; in the real world, there are economic measures:
  - Your choices are:
    - Do no classification
      - That has a cost (hard to compute)
    - Do it all manually
      - Has an easy-to-compute cost if you're doing it like that now
    - Do it all with an automatic classifier
      - Mistakes have a cost
    - Do it with a combination of automatic classification and manual review of uncertain/difficult/"new" cases
  - Commonly the last method is cost efficient and is adopted
    - With more theory and Turkers: Werling, Chaganty, Liang, and Manning (2015). On-the-Job Learning with Bayesian Decision Theory. http://arxiv.org/abs/1506.03140

A common problem: Concept Drift

- Categories change over time
- Example: “president of the united states”
  - 1998: clinton is great feature
  - 2018: clinton is bad feature
- One measure of a text classification system is how well it protects against concept drift.
  - Favors simpler models like Naïve Bayes
- Feature selection: can be bad in lessening protection against concept drift

Summary

- Decision trees
  - Simple non-linear, discriminative classifier
  - Easy to interpret
  - Moderately effective for text classification
- Logistic regression and Support vector machines (SVM)
  - Linear discriminative classifiers
  - Close to state of art (except perhaps NNs) for a single classifier
  - We're not covering them in this year's class
- Classifier ensembles
  - Random forests (bagging)
  - Boosting
- Comparative evaluation of methods
- Real world: exploit domain specific structure!

Resources for today's lecture

- S. T. Dumais. 1998. Using SVMs for text categorization, IEEE Intelligent Systems, 13(4)
- Yiming Yang, Xin Liu. 1999. A re-examination of text categorization methods. 22nd Annual International SIGIR
- Tong Zhang, Frank J. Oles. 2001. Text Categorization Based on Regularized Linear Classification Methods. *Information Retrieval* 4(1): 5-31
- Trevor Hastie, Robert Tibshirani and Jerome Friedman. *Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer-Verlag, New York.
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- Fan Li, Yiming Yang. 2003. A Loss Function Analysis for Classification Methods in Text Categorization. ICML 2003: 472-479.
- Tie-Yan Liu, Yiming Yang, Hao Wan, et al. 2005. Support Vector Machines Classification with Very Large Scale Taxonomy, SIGKDD Explorations, 7(1): 36-43.
- 'Classic' Reuters-21578 data set: http://www.daviddlewis.com/resources/testcollections/reuters21578/