Text Processing on the Web

Week 9

Machine Learning and Text Classification

Recap

- Evaluation of summaries
  - N-gram overlap is well correlated
- Summarization
  - PageRank: nodes are sentences; edges are similarities
  - Extractive summarization with smaller units
    - Alignment (Jing) and Bayesian Noisy Channel model
  - Supervised and unsupervised approaches to choosing sentences

- An Introduction to Machine Learning
  - Supervised version: train and test
  - Learn patterns from vector of features and class
  - Prefer simpler hypotheses

Outline

Overview of machine learning variants

More on text classification as machine learning

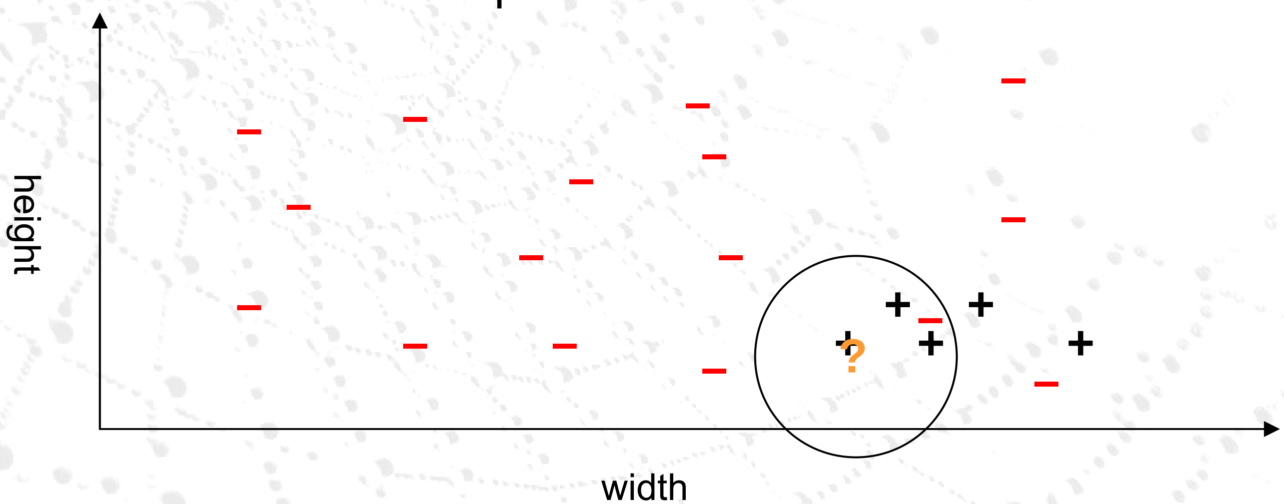
- Feature selection
- Dataset skew
- Classification methods on your own (chapter readings)

Learners

- Nearest Neighbors
- Regression
- Neural Networks
- Naïve Bayes
- Decision Trees
- Support Vector Machines
- Maximum Entropy

Nearest Neighbor

- A type of instance based learning – no model
- Remembers all of the past instances
- Uses the nearest old data point as answer



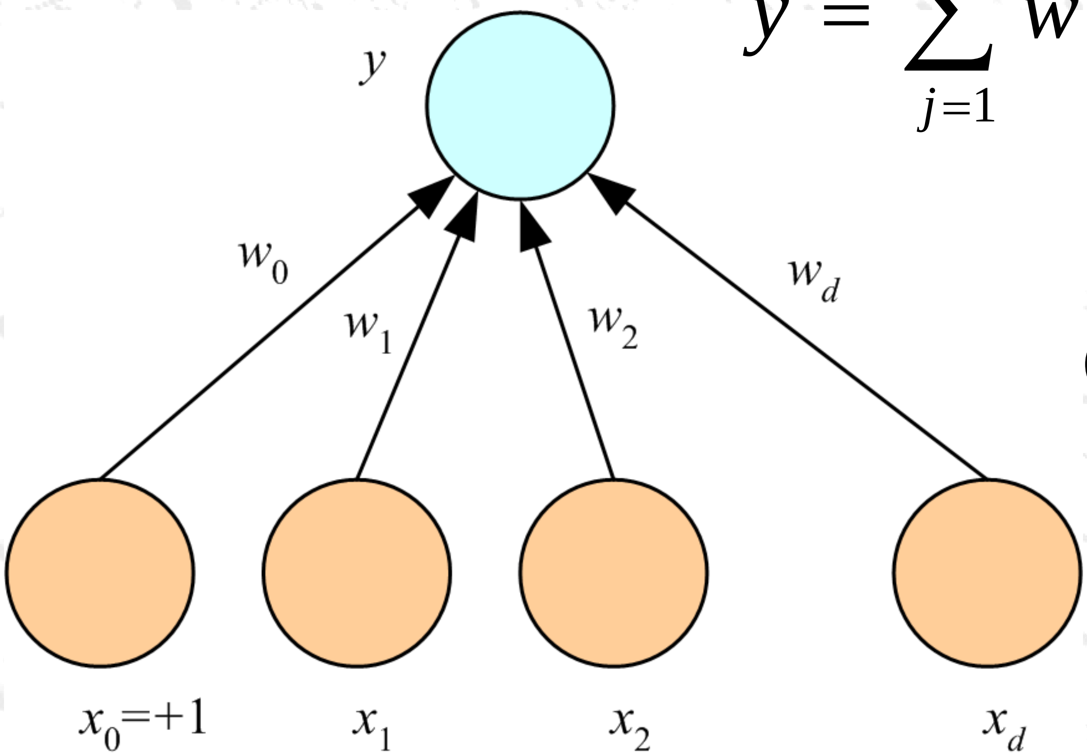
- Above, a problem with |**x**| = 2 and f(**x**) = {+,-}
- Generalize to kNN, that is, take the average class of the closest k neighbors.

Remarks on kNN

- Inductive bias
  - Similar classification of nearby instances…
- Curse of dimensionality
  - Similarity metric mislead by irrelevant attributes
  - Solutions:
    - Weight each attribute differently:
      - Use cross-validation to automatically choose weights
    - Stretch each axis by a variable value.
- Efficient memory indexing is necessary
  - Databases: kd-tree

Perceptrons – A basis for regression and neural networks

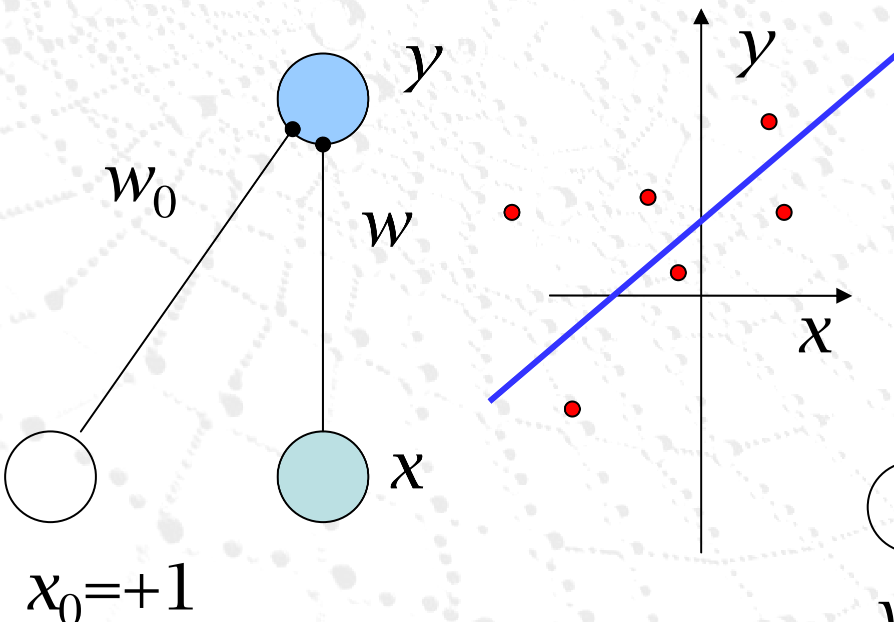
$$y = \sum_{j=1}^{d} w_j x_j + w_0 = \mathbf{w}^T \mathbf{x}$$

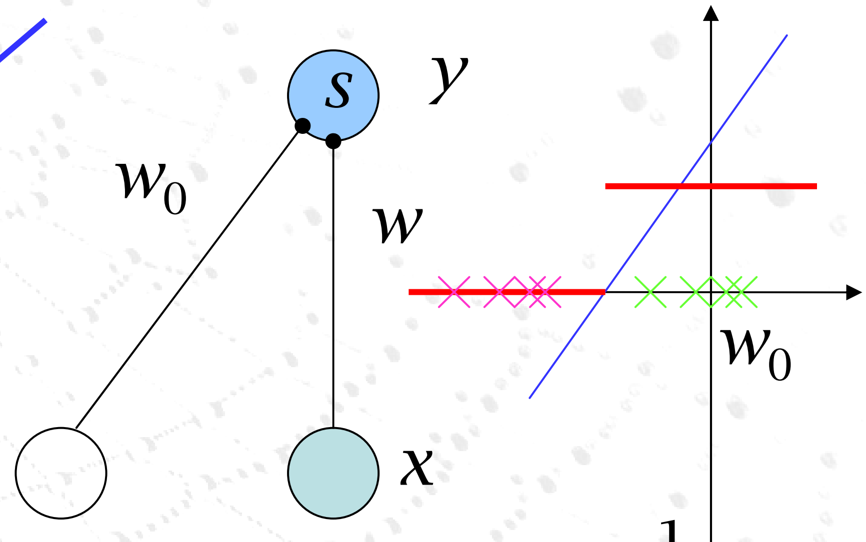


(Rosenblatt, 1962)

What a Perceptron Does

- Regression: y=wx+w0
- Classification: y=1($wx+w_0>0$)





$$y = \text{sigmoid}(o) = \frac{1}{1 + \exp\left[-\boldsymbol{w}^T \boldsymbol{x}\right]}$$

Compute w_i from training data to minimize error

Error often measured by squared error

Multi-class classification

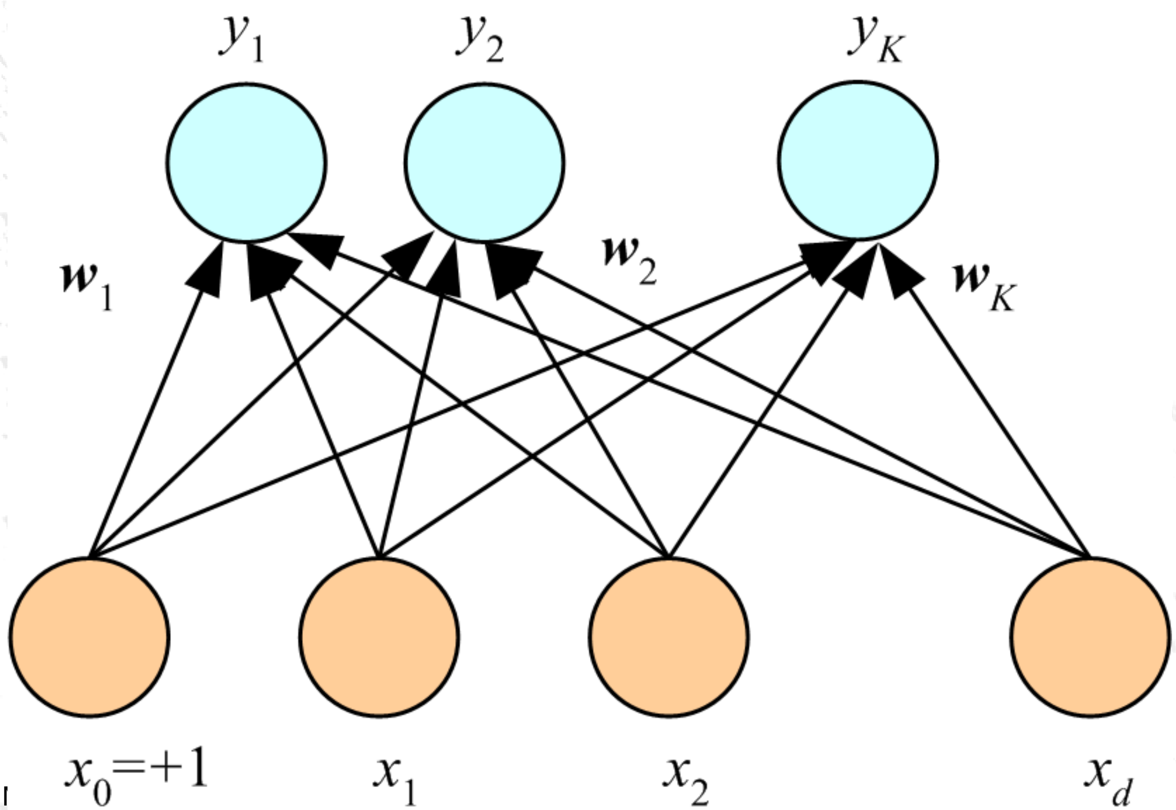
Classification:

$$o_i = \boldsymbol{w}_i^T \boldsymbol{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

choose C_i

if $y_i = \max_k y_k$



Learning weights

- Iterative learning is applied in such algorithms

1. Set weights uniformly or randomly
2. Calculate errors
   - Either on full batch of training data on on single instances
3. Update weights to minimize errors and repeat

$$\Delta w_{ij}^{t} = \eta\left(r_i^t - y_i^t\right)x_j^t$$

$$\text{Update} = \text{LearningFactor} \cdot \left(\text{DesiredOutput} - \text{ActualOutput} \right) \cdot \text{Input}$$

- Many names for different ways of doing this:
  - Gradient descent (delta rule, LMS)
  - Backpropagation (for networks)

Remarks on Regression/ANN

- Perceptron units can be layered together to form networks
- Pros (Networks):
  - Robust to noise
  - Good for high dimensional data
- Pros (Regression):
  - Can predict continuous values
- Cons:
  - Network versions of this can be very slow to train
  - People generally can’t interpret the resulting model

Naïve Bayes

Create a model from the training data:

NaïveBayesLearn(*examples)*
  For each target value v_j
    $P'(v_j) \leftarrow$ estimate $P(v_j)$
    For each attribute value a_i of each attribute a
      $P'(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$

Predict:

ClassfyingNewInstance(x)
  v_{nb}= argmax $P'(v_j) \Pi P'(a_i|v_j)$

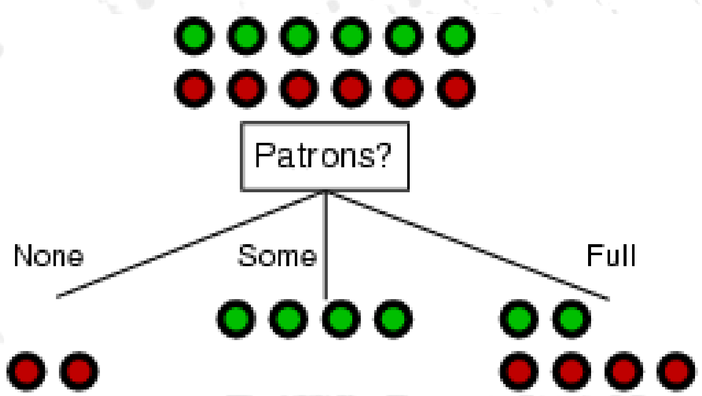
Remarks on Naïve Bayes

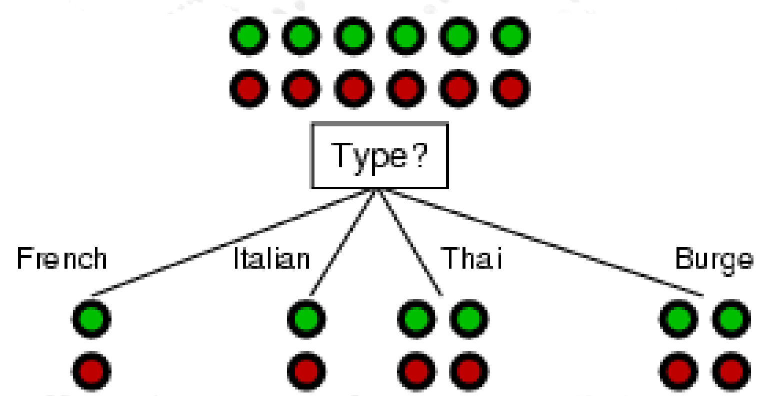
- Very fast to learn and apply
  - Decomposes model to
    - a prior distribution of the classes, and
    - posterior distributions of features given a class
  - A good baseline algorithm to test with

- Has problems with correlated features
  - Assumes independence between features
    - Each feature's probability is simply multiplied through
  - In practice, this doesn't seem to be too much of a problem

Decision Trees

- Divide and conquer strategy
- Sequentially choose a dimension of **x** to split on that makes the subproblems as easy as possible
- "easy" = information gain





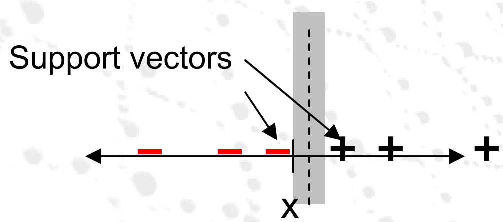
Remarks on Decision Trees

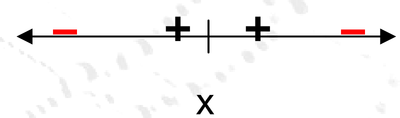
- Normal training speed; fast testing
  - Complexity proportional to |**x**| and # of instances
  - Need to compute best feature after every new rule
  - But just need to apply tree rules in testing
- Pros
  - Easy to analyze: people easily understand hypotheses, easier for post-analysis
- Cons:
  - Can overfit data easily
  - Large inductive bias: considers only on feature at a time
  - Most methods adopt a version of pruning to give some assurance of the generalizability of its rule

Support Vector Machines

A complex topic, let's just go over the very basic

- Basic SVMs use a line (hyperplane) to separate the classes
  - Draws a line to maximize the margin between the classes
  - Only care about data instances (support vectors) near the boundary; other instances are not used



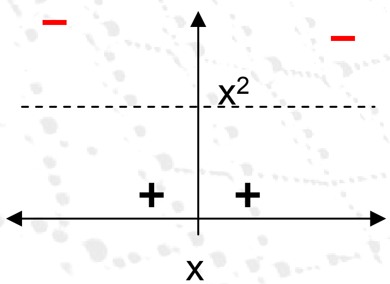


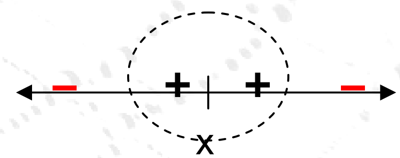
- Left is linearly separable with one line but the right is not

Support Vector Machines

Solution: Map the data into a higher dimensional space

- This is called the **kernel trick**
- This guarantees that it will be separable, allowing non linear classification
- Relies on *k*(x,y), a *kernel* function that takes two points in the original input space and calculates their distance





- The same data set is now separable

Remarks on SVMs

- A learner that seems to have good performance for many different scenarios
- Sensitive to choice of kernel function
  - That is, how to calculate how close two data points are
  - Variety of kernel functions to try
  - Sequence data and tree data structures can be compared using different kernels
- Running time depends heavily on kernel function

The Maximum Entropy Principle

A type of constraint satisfaction: find a model that fits all of the training data

- Use an exponential model

Weight (to be learned)

Features (usually binary-valued)

$$p_s = \frac{\exp\left(-\sum_i \lambda_i f_i(s)\right)}{Z}$$

Normalization (to make it a probability)

- Given some set of constraints which must hold, what is the best model among those available?
  - Answer: the one with maximum entropy
  - Meaning that it doesn't assume more than what is necessary

- Why? ...philosophical answer:
  - Occam's razor, don't pretend you know something you don't

Example

- Throwing the “unknown” die
  - do not know anything – we should assume a fair die

    (uniform distribution ~ max. entropy distribution)

- Throwing unfair die
  - we know: p(4) = 0.4, p(6) = 0.2, nothing else
  - best distribution?
  - do not assume anything about the rest:

1	2	3	4	5	6
0.1	0.1	0.1	0.4	0.1	0.2

Remarks: Max Ent

- Similar in spirit to SVM's max margins
  - Make hypothesis as general as possible
- Features
  - Are usually binary valued
  - Used a lot in sequence labeling tasks
  - Often encode previous decisions in sequence learning
    - E.g., last word was labeled as an adjective
- Is the basis for a number of more complex sequence labeling models (more on this later)
  - Max. Entropy Markov Models (MEMM)
  - Conditional Random Fields (CRF)

Other issues in Text Classification

- Feature selection
- Weighting schemes
- Choice of classifier

Recap on Text Classification

- Use a machine learning technique to assign a document d to a category c

Some characteristics:

  – |D| >> |C|, where there are numerous examples for each C
  – Represent each d as a set of features $f_1 \ldots f_n$, typically each w in vocabulary is a feature, weighted by tf.idf
  – Results in thousands of features

Curse of Dimensionality

Two problems:

- Some learning methods don't work well with thousands of features.
- Many datasets don't have enough examples to generate **sufficient statistics** for features

When the statistics can sub for the distribution in inference decisions

Solution?

- Use dimensionality reduction
- Use feature selection

Classification Method

- Choice of methods (Global vs. local classifier)
  - Global: one multi-class classifier
  - Local: Many binary classifiers, making Y/N decisions

AI	**OS**	**Media**	
Classifier 1	Classifier 2	Classifier	Local Classifier

One Global Classifier
Output: 1 of N categories

Global Classifier

Feature Selection

- Selecting / eliminating features based on criteria on a feature's (term's) distribution (or weight)
- Decision of local vs. global features
  - Global: one set of features for one or more classifiers
  - Local: each classifier uses own (local) features

AI	**OS**	**Media**	
AI Machine Reasoning	Operating Embedded Software	Multimedia Video Speech	Local Dictionary
AI, Operating, Multimedia			Global Dictionary

Choice of features and feature selection method have largest influence on categorization performance.

IR and TC

To think about … carefully

- IR favors rare features
  - Retains all non-trivial terms
  - Use IDF to select rare features
- TC needs common features in each category
  - DF is more important than IDF

What are the differing characteristics of these two problems?

Feature Selection Methods

- DF: Document Frequency
- IG: Information Gain
- MI: Mutual Information
- CHI: X^2 statistic

Term/Class Contingency Table

	T_k=1 (Occurs)	T_k=0 (Absent)
C_i=1 (Relevant)	A	C
C_i=0 (Non-relevant)	B	D

Selection Methods, 1

- DF: throw away all terms that occur in less than n documents
  - Equate noise with rare terms
  - But IR assumes such rare terms can indicate content, so we typically don't set this too aggressively

- IG: measure number of bits of information that can be used for category prediction

$$G(t) = -\sum_{i=1}^{m} P(c_i)\log P(c_i) + P(t)\sum_{i=1}^{m} P(c_i \mid t)\log P(c_i \mid t) + P(\bar{t})\sum_{i=1}^{m} P(c_i \mid \bar{t})\log P(c_i \mid \bar{t})$$

Constant across all features

Bias when present

Bias when absent

Selection Methods, 2

- MI - consider how often t and c co-occur (corrected by chance)

Estimated by

- Combine I(t,·) scores for all classes by avg() or max()
  - Which strategy makes sense for global features? For local?

- Sensitive to term frequency. For terms with equal frequency, rare terms are favored. MI scores only comparable when frequency is similar.

Smaller penalty for rarer terms

Selection Methods, 3

- CHI: X^2 statistic measures lack of independence between t and c.
  - Uses one degree of freedom to judge extremeness

$$X^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C)\times(B+D)\times(A+B)\times(C+D)}$$

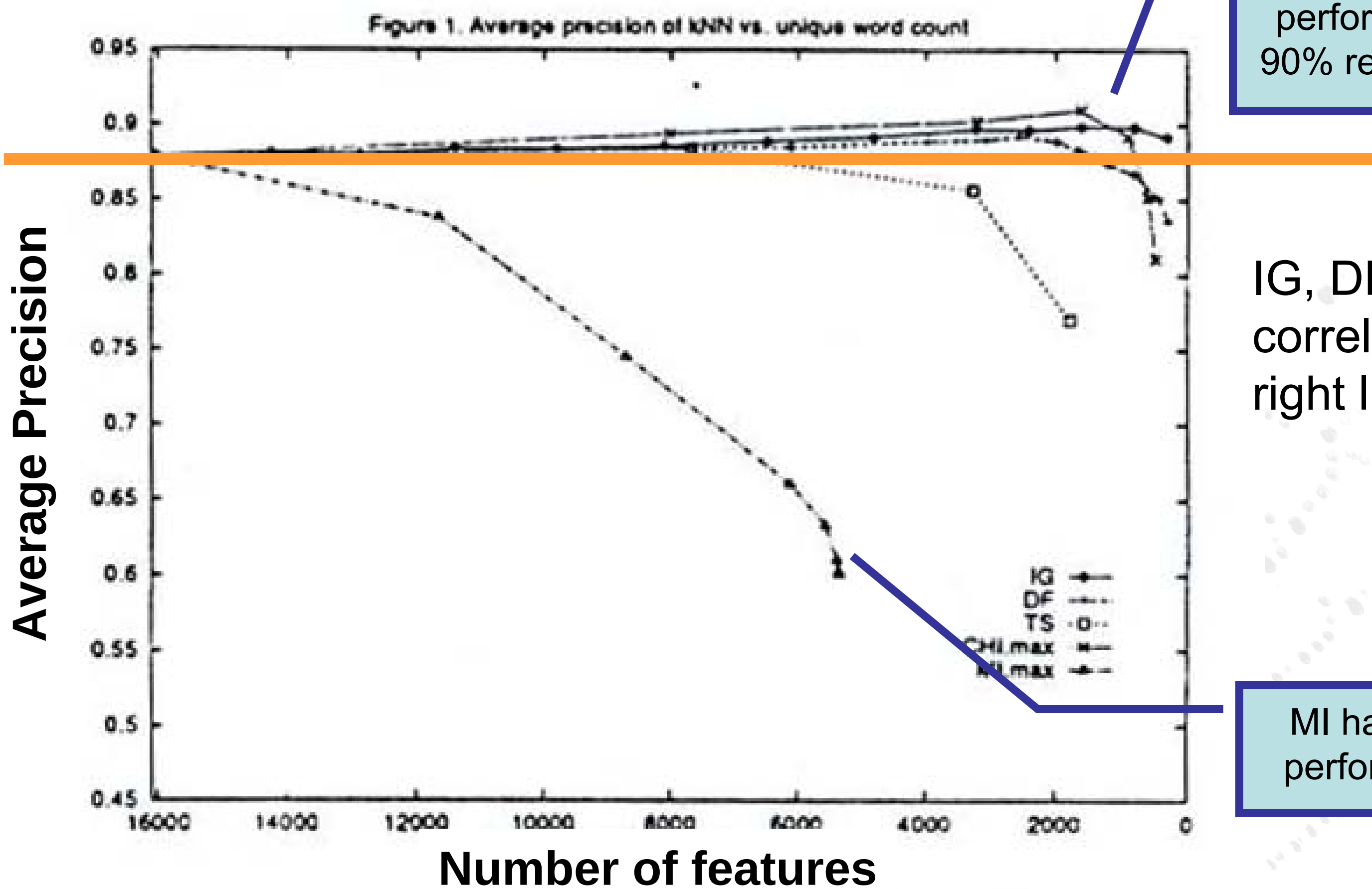
	$T_k=1$	$T_k=0$
$C_i=1$	A	C
$C_i=0$	B	D

  - Again, use max() or avg() to combine X^2 scores
  - Diff between MI: X^2 is normalized, can compare across terms with different frequencies
  - But not reliable for low frequency terms

Evaluation

- Used older Reuters-22173 dataset
  - 92 categories, 9610 and 3662 train and test docs, respectively, 16K terms after standard preprocessing
  - Distribution is **skewed:** some classes have less than 5 training docs, one class has 30% of all training docs
- Evaluating using average 11 point precision
  - Compute precision at 11 recall points of 0, 10, 20…100%, then average
  - Use a global kNN classifier

Results



Selection improves performance with up to 90% reduction in featues

IG, DF, CHI all well correlated (upper right lines

MI has poor performance

Feature Selection

- DF is simplest, still shown to be competitive
- CHI (in subsequent tests) works better for local classification methods
  - CHI,MI,IG all take time linear to size of training set to do selection; all favor common terms

- Manual selection of good features works best

- Dimensionality Reduction (PCA/LSA) can be used as well

Outline

Feature selection

>>> Weighting schemes

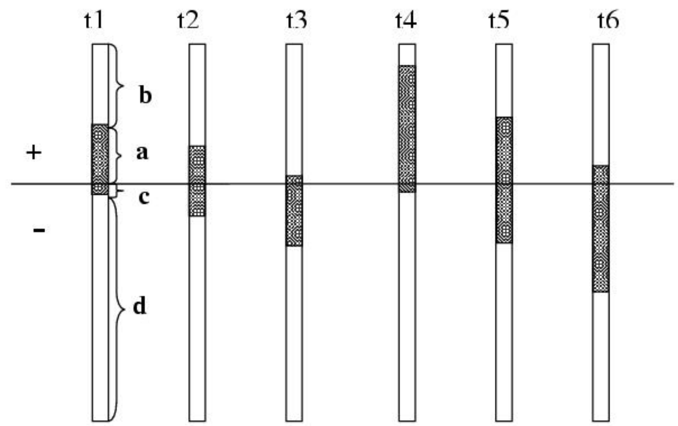
Choice of Classifier

Weighting of features

- Feature weighting plays a role in certain types of classifiers: SVM, kNN.
  - What about NB?
- Support Vector Machines shown to be competitive in accuracy in classification
  - Shown to be attributable more to text representation than kernel function (Leopold 02)

Weighting schemes

- TF
- Log TF
- ITF



- IDF
- TF.IDF
- Log TF.IDF

	$T_k=1$	$T_k=0$
$C_i=1$	A	C
$C_i=0$	B	D

- TF.CHI
- TF.RF

Sensitive to Classification

RF = log (2+a/c)

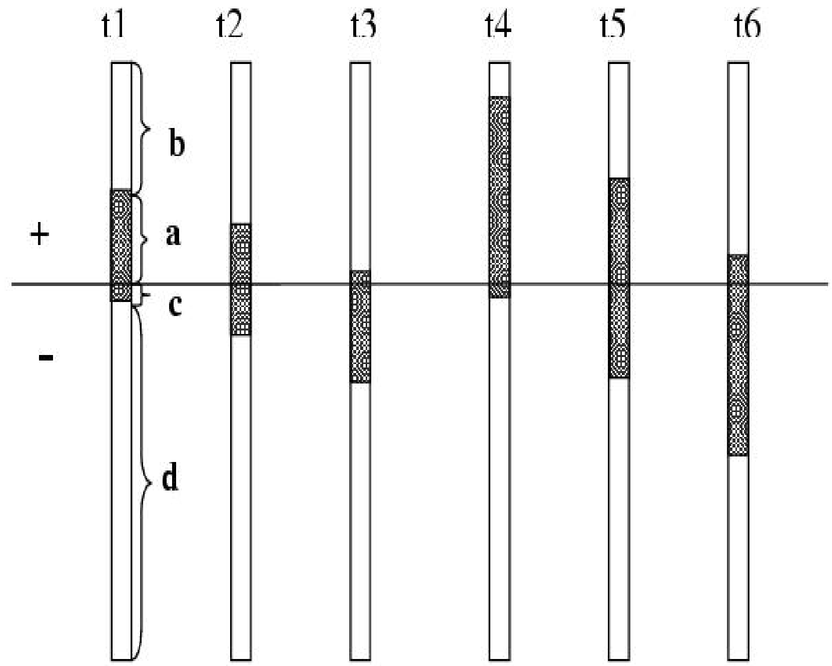
IDF = N/(a+c)

$$\text{CHI} = \frac{N(ab-bc)^2}{(a+c)(a+b)(b+d)(c+d)}$$

Relevant Frequency

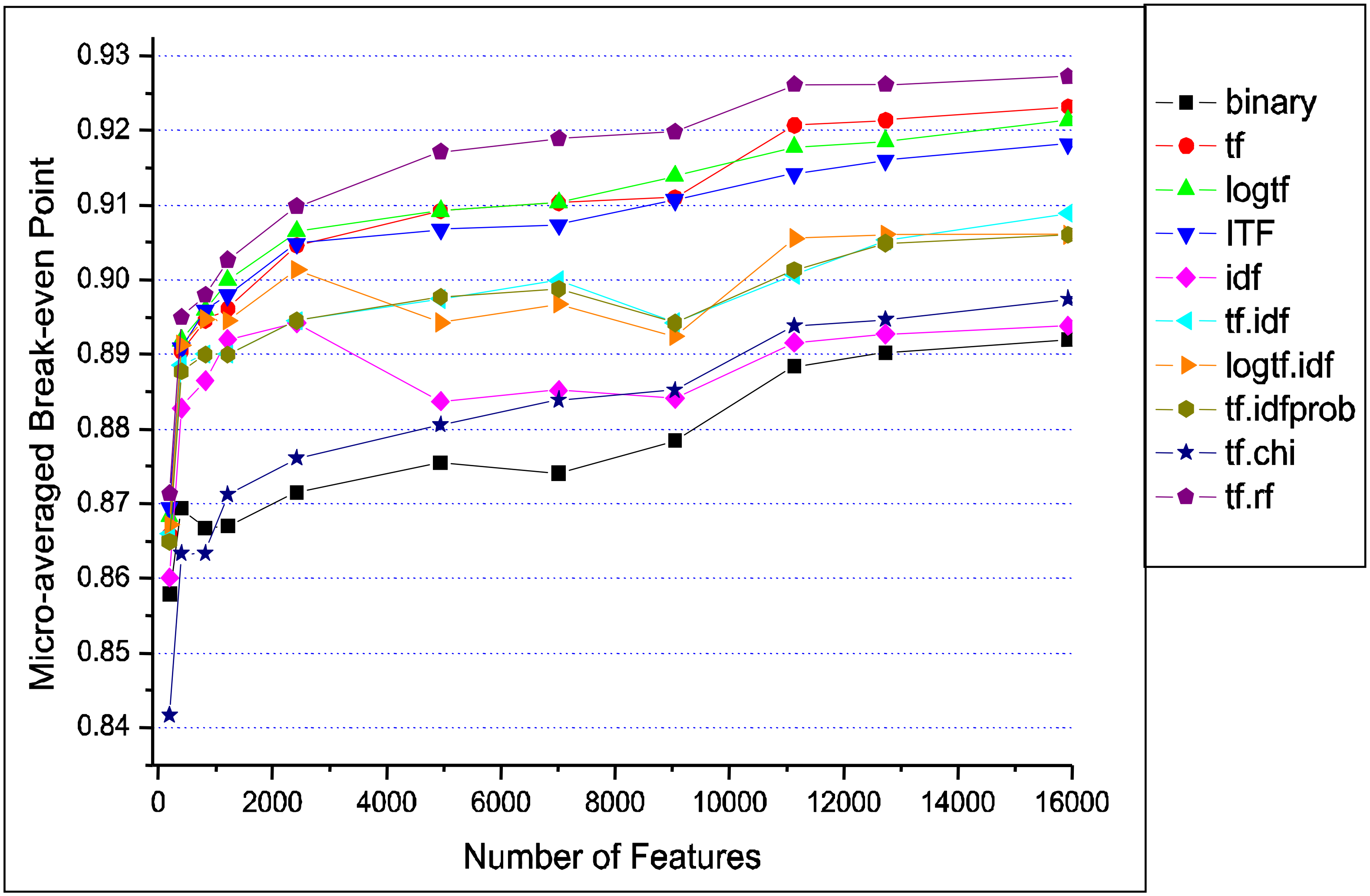
- First three = idf_1
  last three = idf_2

- RF = ratio of **a** to **c** as important,
  while taking into account relative rarity of term



To think about: how is this different from CHI? From IG? From MI?

Results on Reuters 21578



Significance Tests Results on Reuters

#-features	McNemar’s Test
200	{tf.chi} << all the others
400-1500	{binary, tf.chi}<<{all the others}
2500	{binary, tf.chi}<<{idf, tf.idf, tf.idf-prob}<{all the others}
5000+	{binary, idf, tf.chi}<<{tf.idf, logtf.idf, tf.idf-prob}<<{tf, logtf, ITF}<{tf.rf}

‘<’ and ‘<<’ denote better than at significance level 0.01 and 0.001 respectively; ‘{}’ denote no significant difference

Outline

Feature selection

Weighting schemes

>>> Choice of Classifier

Choice of Classifiers

- Used X^2 or IG for feature selection
- Then used feature sizes that resulted in best F_1 score, shown below in parentheses

Methods tested

- SVM (10k)
- kNN (2.4k, with k=45)
- NNet (1k)
- NB (2k)
- Rocchio* - from other paper – not directly comparable

Evaluations

- Reuters 21578 dataset (ModApte aka ApteMod)
  - More modern version of the Reuters 22173 set
  - 7769 Train, 3019 Test docs, |V| = 24240 after preprocessing
  - Also **heavily skewed**: most freq class 2K+ docs, over 70 of 90 total class have less than 100 instances

- Used F_1 scores to evaluate
  - Macro average = each class has equal weight
  - Micro average = each instance has equal weight

Pop quiz: When can you have very high macro average but low micro average? What about vice versa?

Results

	MicR	MicP	$MicF_1$	$MacF_1$
NB	0.7688	0.8245	0.7956	0.3886
NNet	0.7842	0.8785	0.8287	0.3763
KNN	0.8339	0.8807	0.8567	0.5242
SVM	0.8120	0.9137	0.8599	0.5251

- Micro-averaged F1 shows:

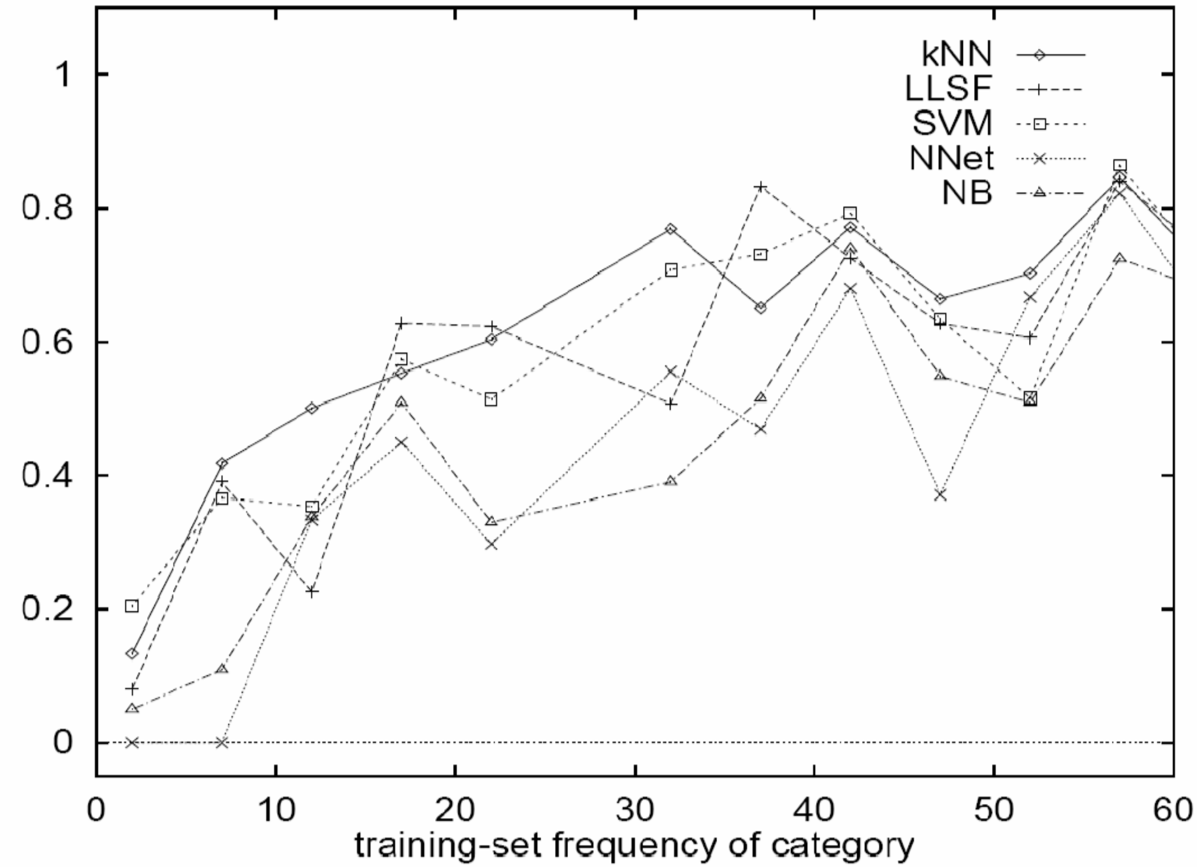
  SVM > kNN >> NNet >> {NB,Rocchio*}

- Macro-averaged F1 shows:

  {SVM,kNN} >> {NNet,NB}

Sensitivity to training frequency

- NNet clearly worse
- But others not conclusive
- Rest of graph (60+ freq) is more smooth
- Here, # docs a surrogate for less data



Summary

- This week: other TC issues
  - Feature selection / weighting
  - Dataset skew / # of examples
- To think about: how is TC different from IR
  - Relevance Info
  - TC acting as a filter for more detailed IR?

References

- Lan Man, Chew-Lim Tan, Hwee-Boon Low, Sam-Yuan Sung (05) A comprehensive comparative study on term weighting schemes for text categorization with support vector machines, Poster Paper in WWW '05.
- Debole and Sebastiani (04) An analysis of the Relative Difficulty of the Reuters-21578 Subsets. In LREC '04. http://nmis.isti.cnr.it/sebastiani/Publications/LREC04.pdf