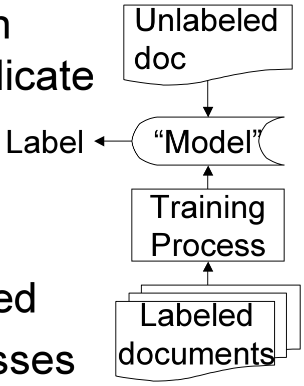
Document Classification (Supervised Learning)

Soumen Chakrabarti
IIT Bombay
www.cse.iitb.ac.in/~soumen

Definition and motivating scenarios

- Entities = documents
  - Document models at different levels of detail
- Each document has a label taken from a finite set; a label could indicate
  - "News article is about cricket"
  - "Email is spam"
  - "Doc pair is (nearly) duplicate"
- Training set of with labels provided
- Test doc w/o labels: system guesses
- Many applications



Evaluating classifiers: recall, precision

- Document can have only one label
  - Confusion matrix $M[i,j]$ = number of docs with "true" label i assigned label j by classifier
  - Accuracy = sum of diagonals / total over matrix
- Document can have multiple labels (classes)
  - For each label c set up a 2×2 matrix $M_c[i,j]$
  - True label-set includes c (i=1,0)
  - Classifier's guessed label set includes c (j=1,0)
  - Recall for label c = $M_c[1,1]/(M_c[1,1]+M_c[1,0])$
  - Precision for label c = $M_c[1,1]/(M_c[1,1]+M_c[0,1])$

0,0	0,1
1,0	1,1

Averaging over labels, break-even

- Macro-averaging over labels
  - Overall recall (precision) is average over labels
  - Less populated labels get undue representation
- Micro-averaging over labels
  - Add up all the M_c matrices into one matrix M
  - Compute recall and precision of M
  - Labels appearing on many docs dominate score
- $F_1 = 2 \times$ precision $\times$ recall / (precision + recall)
- Recall and precision usually inversely related
  - Vary system parameters to get trade-off
  - Find intersection of PR-plot with P=R (breakeven)

Vector space model

- Document *d* is a point in Euclidean space
  - Each dimension corresponds to a term *t*
- Component along axis *t* = product of…
  - $$\mathrm{TF}(d,t)=\begin{cases}0 & \text{if } n(d,t)=0\\ 1+\log\left(1+\log n(d,t)\right) & \text{otherwise}\end{cases}$$
  - $$\mathrm{IDF}(t)=\log\frac{1+|D|}{|D_t|}$$
  - Here *n*(*d*,*t*) = #times *t* occurs in *d*, *D* = entire collection, D_t = documents containing *t*
- Ad-hoc choices, but validated by decades of Information Retrieval research

Components for rare terms scaled up

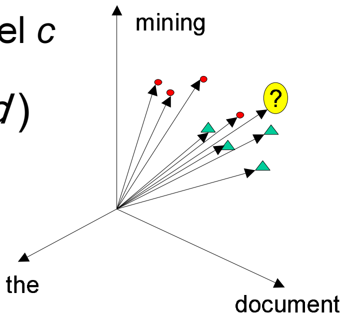
Large term frequencies dampened

Nearest-neighbor classifiers

- At training time, record each doc *d* as a labeled point in vector space
- Test doc *q* also mapped to vector space
- Similarity between *q* and *d* is cos(*q*,*d*)
- Pick *k* training documents most similar to *q*
  - *k*NN$_c$(*q*) = subset which has label *c*

$$\mathrm{score}(c,q)=b_c+\sum_{d\in k\mathrm{NN}_c(q)}\cos(q,d)$$

- b_c is a tuned constant for each class



Multivariate binary model

- Faithful vs. practical models
  - Attribute = term, phrase, sentence, para, …?
  - Enormous number of dimensions (30k—500k)
  - Difficult to model joint distribution in any detail
- “Set of words” (multivariate binary)
  - Doc = bit vector with #elems = size of vocabulary
  - Bit at position *t* = [term *t* appears in doc]
  - Term counts and ordering ignored
- Naïve independence assumption

$$Ᵽ\left(\vec{d}\right) = \prod_{t\in d}\phi_t \prod_{t\notin d}\left(1-\phi_t\right) \qquad Ᵽ\left(\vec{d}\mid c\right) = \prod_{t\in d}\phi_{c,t} \prod_{t\notin d}\left(1-\phi_{c,t}\right)$$

Multinomial (bag-of-words) model

- Author samples length ℓ (total term count) from a suitable length distribution
- Each of ℓ terms chosen by sampling independently from a multinomial distribution of terms
- Simplifying (crude!) assumptions
  - Terms independent of each other, unordered
  - Equally surprised by 1st and 101st occurrence!

$$\Pr\left(\vec{d}\right) = \Pr(\ell)\binom{\ell}{\{n(d,t)\}}\prod_{t\in d}\theta_t^{n(d,t)} \qquad \Pr\left(\vec{d}\mid c\right) = \Pr(\ell\mid c)\binom{\ell}{\{n(d,t)\}}\prod_{t\in d}\theta_{c,t}^{n(d,t)}$$

Naïve Bayes classifiers

- For simplicity assume two classes {–1,1}
- *t*=term, *d*=document, *c*=class, ℓ_d=length of document *d*, *n*(*d*,*t*)=#times *t* occurs in *d*
- Model parameters
  - Priors Pr(*c*=–1) and Pr(*c*=1)
  - $\theta_{c,t}$=fractional rate at which *t* occurs in documents labeled with class *c*
- Probability of a given *d* generated from *c* is

$$\Pr(d \mid c, \ell_d) = \cancel{\binom{\ell_d}{\{n(d,t)\}}} \prod_{t \in d} \theta_{c,t}^{n(d,t)}$$

Naïve Bayes = linear discriminant

- When choosing between the two labels
  - Terms involving document length cancel out
  - Taking logs, we compare

$$\log \Pr(c=1) + \sum_{t \in d} n(d,t) \log \theta_{1,t} : \log \Pr(c=-1) + \sum_{t \in d} n(d,t) \log \theta_{-1,t}, \text{ or}$$

$$\sum_{t \in d} \left(\log \theta_{1,t} - \log \theta_{-1,t}\right) n(d,t) + \left(\log \Pr(c=1) - \log \Pr(c=-1)\right) : 0$$

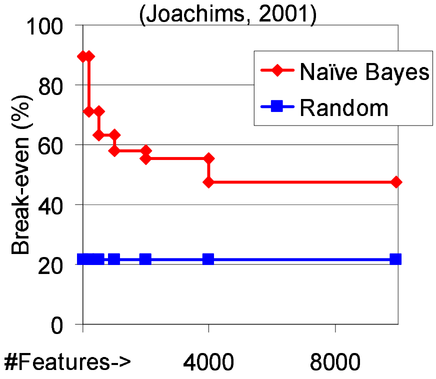
- The first part is a dot-product, the second part is a fixed offset, so we compare

$$\alpha_{\mathrm{NB}} \cdot d + b : 0$$

- Simple join-aggregate, very fast

Many features, most very noisy

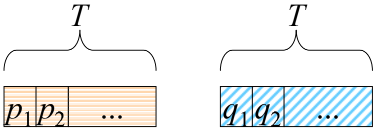
- Sort features in order of decreasing correlation with class labels
- Build separate classifiers
  - 1—100, 101—200, etc.
- Very few features suffice to give highest possible accuracy



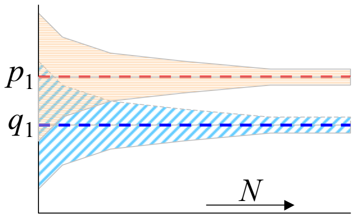
- Want to select that subset of features leading to highest accuracy
  - Reduced space and time requirements
  - May even improve accuracy by reducing "over-fitting"

Feature selection in the binary model

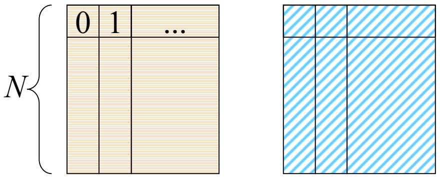
Model with unknown parameters



Confidence intervals



Observed data



Pick $F \subseteq T$ such that models built over F have high separation confidence

Feature selection by accumulation

- Add “best” features to an empty set
- Several measures of association between labels and features
  - Standard chi-square test of dependence

$$\chi^2 = \sum_{\ell,m} \frac{n(k_{11}k_{00} - k_{10}k_{01})^2}{(k_{11}+k_{10})(k_{01}+k_{00})(k_{11}+k_{01})(k_{10}+k_{00})}$$

  - Mutual information between term and label

$$\mathrm{MI}(I_t, C) = \sum_{\ell,m} \frac{k_{\ell m}}{n} \log \frac{k_{\ell m} / n}{(k_{\ell 0}+k_{\ell 1})(k_{0m}+k_{1m}) / n^2}$$

  - Fisher’s index

$$\mathrm{FI}(t) = \frac{\sum_{c_1,c_2} \left(\mu_{c_1,t} - \mu_{c_2,t}\right)^2}{\sum_{c} \frac{1}{|D_c|} \sum_{d \in D_c} \left(x_{d,t} - \mu_{c,t}\right)^2}$$

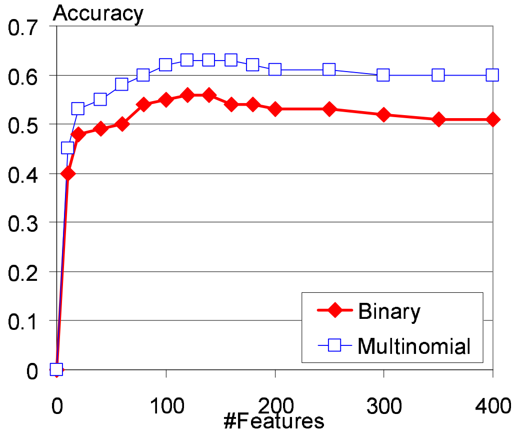
- May include good but redundant features

Feature selection by truncation

- Starting with all terms, drop worst features
- *P* and *Q* are conditionally independent given *R* if Pr(*p*|*q*,*r*) = Pr(*p*|*r*) for any *p*,*q*,*r*
  - *Q* gives no extra info about *P* over and above *R*
- *T*=full feature set, *M*=a subset of features, *X*=“event” for a given term ($X \notin M$)
- *M* is a “Markov blanket” for *X* if *X* is conditionally independent of $T \cup C - M - X$
  given *M*
- Search for *X* and drop *X* from feature set *F* while Pr(*C*|*F*) remains close to Pr(*C*|*T*)
- Computationally expensive

Effect of feature selection

- Sharp knee in accuracy achieved with a very small number of features
- Saves class model space
  - Easier to hold in memory
  - Faster classification
- Mild *decrease* in accuracy beyond a maximum
  - Worse for binary model

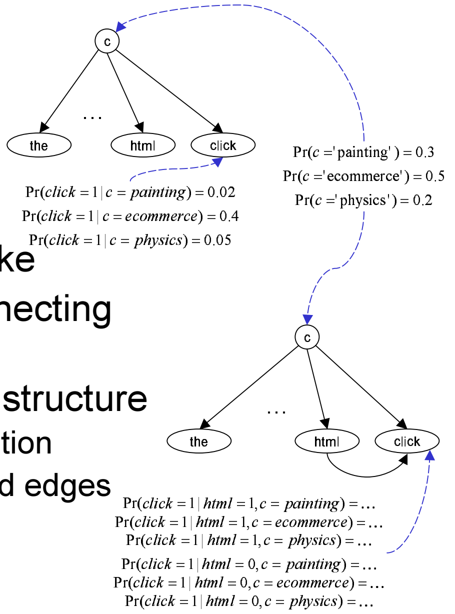


Limitations and better techniques

- Problems with naïve Bayes classifiers
  - Seek to model Pr(*d*|*c*): difficult because *d* has very large number of dimensions
  - Independence assumption gives terrible estimates (although decision boundaries may be ok)
- Remedies
  - Drop (some) independence assumptions: from naïve Bayes to low-degree Bayesian networks
  - Estimate Pr(*c*|*d*) directly instead of going via Pr(*d*|*c*): maximum entropy and regression
  - Discriminative (vs. probabilistic) classification

Small-degree Bayesian networks

- Directed acyclic graph
  - Nodes = random variables (1 for class label, 1 for each term)
  - Edges connect coupled variables
- Naïve Bayes: hub-and-spoke
- General model: edges connecting dependent terms
- Problem: induce the graph structure
  - Precompute pairwise correlation
  - Greedy addition of nodes and edges
- Computationally expensive



$\Pr(c = \text{'painting'}) = 0.3$

$\Pr(c = \text{'ecommerce'}) = 0.5$

$\Pr(c = \text{'physics'}) = 0.2$

$\Pr(\mathit{click} = 1 \mid c = \mathit{painting}) = 0.02$

$\Pr(\mathit{click} = 1 \mid c = \mathit{ecommerce}) = 0.4$

$\Pr(\mathit{click} = 1 \mid c = \mathit{physics}) = 0.05$

$\Pr(\mathit{click} = 1 \mid \mathit{html} = 1, c = \mathit{painting}) = \ldots$

$\Pr(\mathit{click} = 1 \mid \mathit{html} = 1, c = \mathit{ecommerce}) = \ldots$

$\Pr(\mathit{click} = 1 \mid \mathit{html} = 1, c = \mathit{physics}) = \ldots$

$\Pr(\mathit{click} = 1 \mid \mathit{html} = 0, c = \mathit{painting}) = \ldots$

$\Pr(\mathit{click} = 1 \mid \mathit{html} = 0, c = \mathit{ecommerce}) = \ldots$

$\Pr(\mathit{click} = 1 \mid \mathit{html} = 0, c = \mathit{physics}) = \ldots$

Maximum entropy classifiers

- Training documents (d_i, c_i), $i = 1 \ldots N$
- Want model Pr(c |d) using parameters μ_j as

$$\Pr(c \mid d) \propto \frac{1}{Z(d)} \prod_{t \in d} \mu_{c,t}^{n(d,t) / \sum_{\tau} n(d,\tau)}$$

- Constraints given by observed data

For each (c,t): $\sum_d \underline{\underline{\Pr}}(d) \Pr(c \mid d) \frac{n(d,t)}{\sum_{\tau \in d} n(d,\tau)} = \sum_d \underline{\underline{\Pr}}(d,c) \frac{n(d,t)}{\sum_{\tau \in d} n(d,\tau)}$

- Objective is to maximize entropy of p

$$H(p) = -\sum_{d,c} \underline{\underline{\Pr}}(d) \Pr(c \mid d) \log \Pr(c \mid d)$$

- Features
  - Numerical non-linear optimization
  - No naïve independence assumptions

Maxent classifier = linear discriminant

- Comparing two classes

$$\text{Ᵽ}\ \ c=1|d)\propto\prod_{t\in d}\mu_{1,t}^{n(d,t)/\sum_\tau n(d,\tau)} \quad : \quad \text{Ᵽ}\ \ c=-1|d)\propto\prod_{t\in d}\mu_{-1,t}^{n(d,t)/\sum_\tau n(d,\tau)}$$

$$\sum_{t\in d}\frac{n(d,t)}{\sum_\tau n(d,\tau)}\log\mu_{1,t} \quad : \quad \sum_{t\in d}\frac{n(d,t)}{\sum_\tau n(d,\tau)}\log\mu_{-1,t}$$

- Nonlinear perceptron: $c = \text{sign}(\alpha\cdot d+b)$
- Linear regression: Fit α to predict c (=1 or –1, say) directly as $c = \alpha\cdot d+b$
  - Widrow-Hoff update rule:

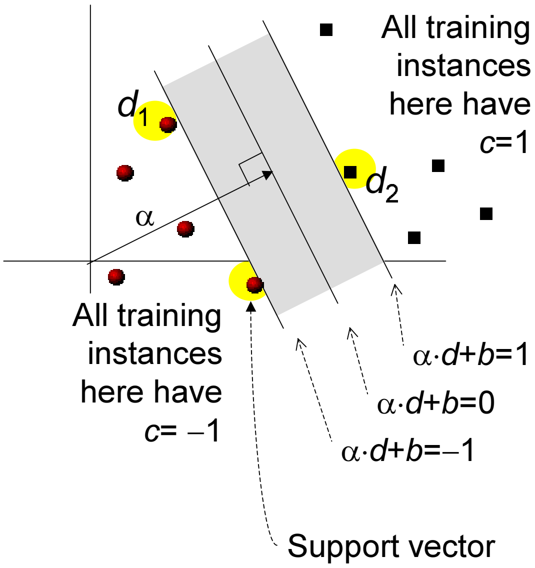
$$\alpha^{(i)}\leftarrow\alpha^{(i-1)}+2\eta(\alpha^{(i-1)}\cdot d_i+b-c_i)d_i$$

Linear support vector machine (LSVM)

- Want a vector α and a constant b such that for each document d_i
  - If c_i=1 then $\alpha\cdot d_i+b \geq 1$
  - If c_i=–1 then $\alpha\cdot d_i+b \leq -1$
- I.e., $c_i(\alpha\cdot d_i+b) \geq 1$
- If points d_1 and d_2 touch the slab, the projected distance between them is

$$2\big/\sqrt{\|\alpha\|}$$

- Find α to maximize this



■ All training instances here have c=1

All training instances here have c= –1

$\alpha\cdot d+b$=1

$\alpha\cdot d+b$=0

$\alpha\cdot d+b$=–1

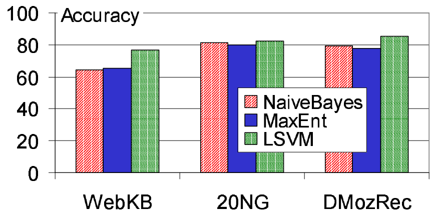
Support vector

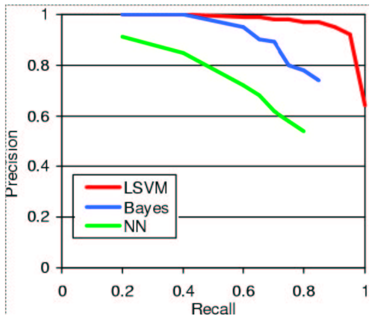
SVM implementations

- α_{SVM} is a linear sum of support vectors
- Complex, non-linear optimization
  - 6000 lines of C code (SVM-light)
- Approx $n^{1.7—1.9}$ time with n training vectors
- Footprint can be large
  - Usually hold all training vectors in memory
  - Also a cache of dot-products of vector pairs
- No I/O-optimized implementation known
  - We measured 40% time in disk seek+transfer

Comparison of accuracy

- Naïve Bayes has mediocre accuracy
- Nearest neighbor has varied reports, depending on tuning parameters
- Support vector machines most consistently superior
- Benchmarks don’t say the whole story
  - Multi-topic docs, hierarchy
  - Dynamic collections
  - Confidence scores





Summary

- Many classification algorithms known
- Tradeoff between simplicity/speed and accuracy
  - Support vector machines (SVM)—most accurate but complex and slow
  - Maximum entropy classifiers
  - Naïve Bayes classifiers—fastest and simplest but not very accurate
- Mostly linear discriminant techniques
  - Can we achieve the speed of naïve Bayes and the accuracy of SVMs?

Fisher’s linear discriminant (FLD)

- Used in pattern recognition for ages
- Two point sets *X* (*c*=1) and *Y* (*c*=–1)
  - $x \in X$, $y \in Y$ are points in *m* dimensions
  - Projection on unit vector α is $x \cdot \alpha$, $y \cdot \alpha$
- Goal is to find a direction α so as to maximize

Square of distance between projected means

$$J(\alpha) = \frac{\left(\frac{1}{|X|}\sum_{x \in X} x \cdot \alpha - \frac{1}{|Y|}\sum_{y \in Y} y \cdot \alpha\right)^2}{\frac{1}{|X|}\sum_{x \in X}(x \cdot \alpha)^2 - \left(\frac{1}{|X|}\sum_{x \in X} x \cdot \alpha\right)^2 + \frac{1}{|Y|}\sum_{y \in Y}(y \cdot \alpha)^2 - \left(\frac{1}{|Y|}\sum_{y \in Y} y \cdot \alpha\right)^2}$$

Variance of projected *X*-points

Variance of projected *Y*-points

Some observations

- Hyperplanes can often completely separate training labels for text; more complex separators do not help (Joachims)
- NB is *biased*: α_t depends only on term *t*—SVM/Fisher do not make this assumption
- If you find Fisher's discriminant over only the support vectors, you get the SVM separator (Shashua)
- Even *random* projections preserve inter-point distances whp (Frankl+Maehara 1988, Kleinberg 1997)

Hill-climbing

- Iteratively update $\alpha_{\text{new}} \leftarrow \alpha_{\text{old}} + \eta \nabla J(\alpha)$ where η is a "learning rate"
- $\nabla J(\alpha) = (\partial J/\partial\alpha_1, \ldots, \partial J/\partial\alpha_m)^T$ where $\alpha = (\alpha_1, \ldots, \alpha_m)^T$
- Need only $5m + O(1)$ accumulators for simple, one-pass update
- Can also write as sort-merge-accumulate

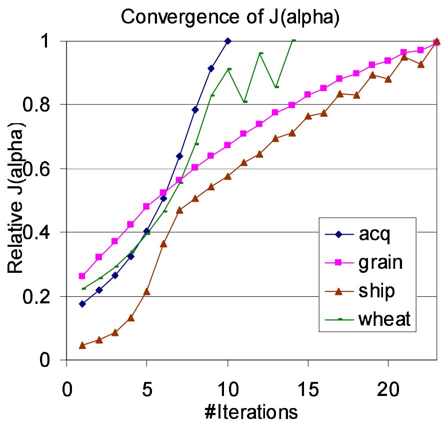
$$\sum_{x\in X} x\cdot\alpha \quad \text{(sa)} \qquad \sum_{y\in Y} y\cdot\alpha \quad \text{(sa)}$$

$$\forall i: \sum_{x\in X} x_i \quad (m\text{ numbers}) \qquad \forall i: \sum_{y\in Y} y_i \quad (m\text{ numbers})$$

$$\forall i: \sum_{x\in X} x_i(x\cdot\alpha) \quad (m\text{ numbers}) \qquad \forall i: \sum_{y\in Y} y_i(y\cdot\alpha) \quad (m\text{ numbers})$$

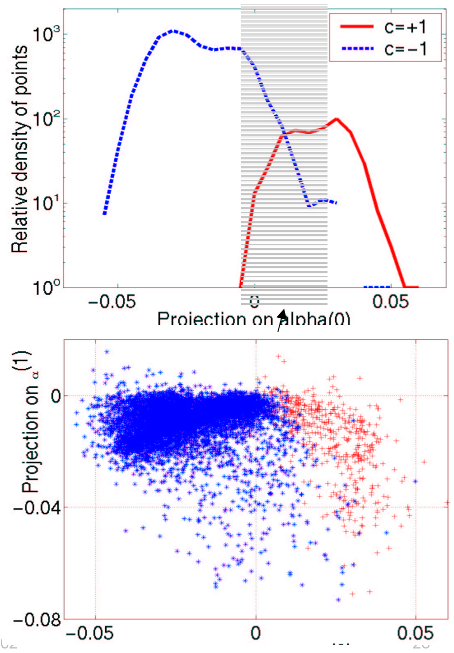
Convergence

- Initialize α to vector joining positive and negative centroids
- Stop if $J(\alpha)$ cannot be increased in three successive iterations
- $J(\alpha)$ converges in 10—20 iterations
  - Not sensitive to problem size
- 120000 documents from http://dmoz.org
  - LSVM takes 20000 seconds
  - Hill-climbing converges in 200 seconds



Multiple discriminants

- Separable data points
  - SVM succeeds
  - FLD fails to separate completely
- Idea
  - Remove training points (outside the gray zone)
  - Find another FLD for surviving points only
- 2—3 FLDs suffice for almost complete separation!
  - 7074→230→2



SIMPL (only 600 lines of C++)

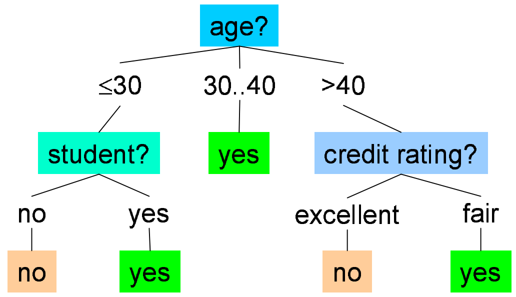
- Repeat for k = 0, 1, …
  - Find $\alpha^{(k)}$, the Fisher discriminant for the current set of training instances
  - Project training instances to $\alpha^{(k)}$
  - Remove points well-separated by $\alpha^{(k)}$

  while $\geq$1 point from each class survive
- Orthogonalize the vectors $\alpha^{(0)}$, $\alpha^{(1)}$, $\alpha^{(2)}$,…
- Project all training points on the space spanned by the orthogonal α's
- Induce decision tree on projected points

Decision tree classifier

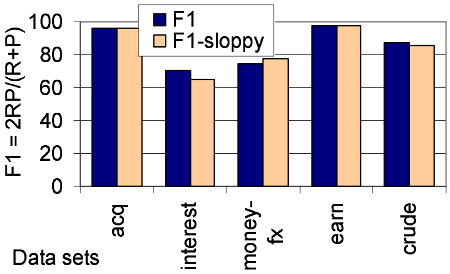
- Given a table with attributes and label
- Induce a tree of successive partitions on the attribute space
- Path in tree = sequence of tests on attrib values
- Extensive research on construction of trees

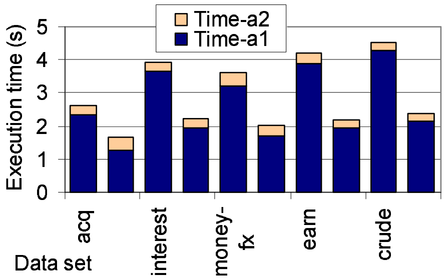
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes



Robustness of stopping decision

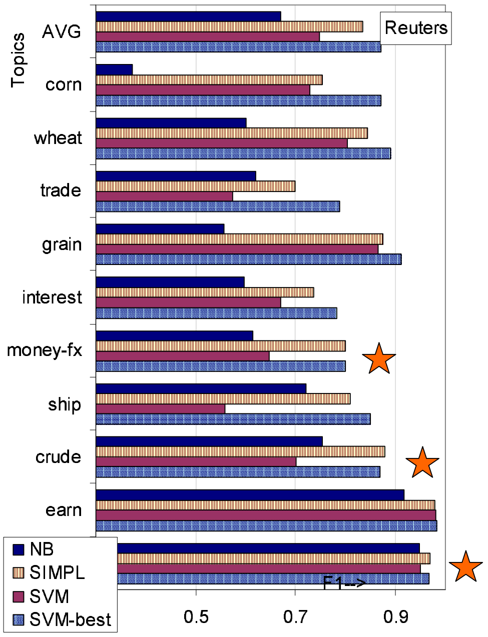
- Compute $\alpha^{(0)}$ to convergence
- Vs., run only half the iterations required for convergence
- Find $\alpha^{(1)}, \ldots$ as usual
- Later αs can cover for slop in earlier αs
- While saving time in costly early-α updates
  - Later αs take negligible time





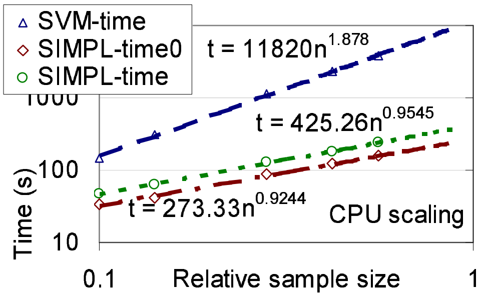
Accuracy

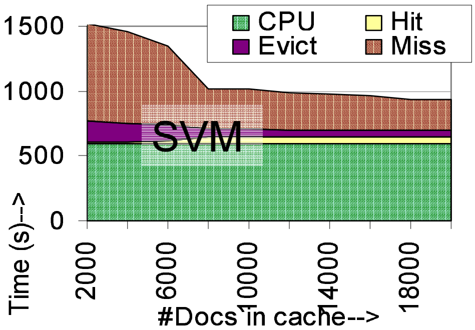
- Large improvement beyond naïve Bayes
- We tuned parameters in SVM to give “SVM-best”
- Often beats SVM with default params
- Almost always within 5% of SVM-best
- Even beats SVM-best in some cases
  - Especially when problem is not linearly separable

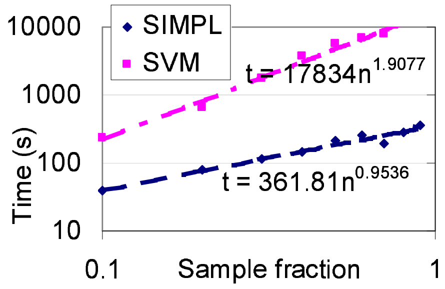


Performance

- SIMPL is linear-time and CPU-bound
- LSVM spends 35—60% time in I/O+cache mgmt
- LSVM takes 2 orders of magnitude more time for 120000 documents







Ongoing and future work

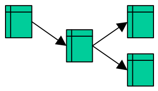
- SIMPL vs. SVM
  - Can we analyze SIMPL?
    - LSVM is theoretically sound, more general
    - Under what conditions will SIMPL match LSVM/SVM?
  - Comparison of SIMPL with non-linear SVMs
- More realistic models
  - Document talks about multiple topics
  - Labels form a “is-a” hierarchy like Yahoo!
  - Labeling is expensive: minimize labeling effort (active learning)
  - Exploit hypertext features for better accuracy

Hypertext Mining

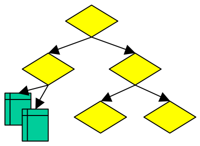
Soumen Chakrabarti
IIT Bombay
www.cse.iitb.ac.in/~soumen

Learning hypertext models

- Entities are pages, sites, paragraphs, links, people, bookmarks, clickstreams…
- Transformed into simple models and relations
  - Vector space/bag-of-words
  - Hyperlink graph
  - Topic directories
  - Discrete time series



```
occurs(term, page, freq)
cites(page, page)
```



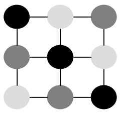
```
is-a(topic, topic)
example(topic, page)
```

Challenges

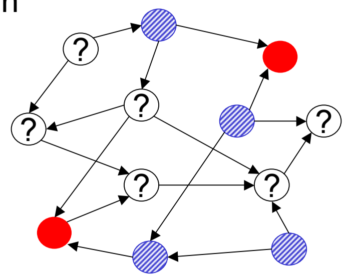
- Complex, interrelated objects
  - Not a structured tuple-like entity
  - Explicit and implicit connections
    - Document markup sub-structure
    - Site boundaries and hyperlinks
    - Placement in popular directories like Yahoo!
- Traditional distance measures are noisy
  - How to combine diverse features? (Or, a link is worth a _?_ words)
  - Unreliable clustering results

Classifying interconnected entities

- Early examples:
  - Some diseases have complex lineage dependency
  - Robust edge detection in images
- How are topics interconnected in hypertext?
- Maximum likelihood graph labeling with many classes



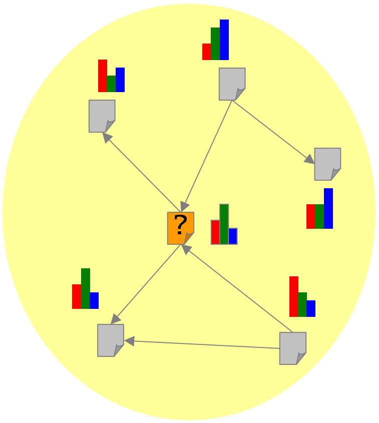
Finding edge pixels in a differentiated image



	🟥	🟦
🟥	0.6	0.4
🟦	0.3	0.7

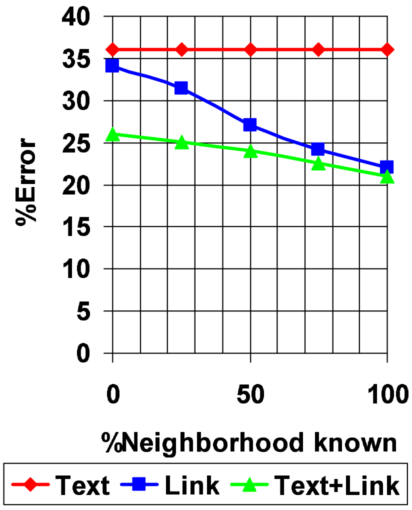
Enhanced models for hypertext

- *c*=class, *d*=text, *N*=neighbors
- Text-only model: Pr(*d*|*c*)
- Using neighbors' text to judge my topic: Pr(*d*, *d*(*N*) | *c*)
- Better recursive model: Pr(*d*, *c*(*N*) | *c*)
- Relaxation labeling until order of class probabilities stabilizes



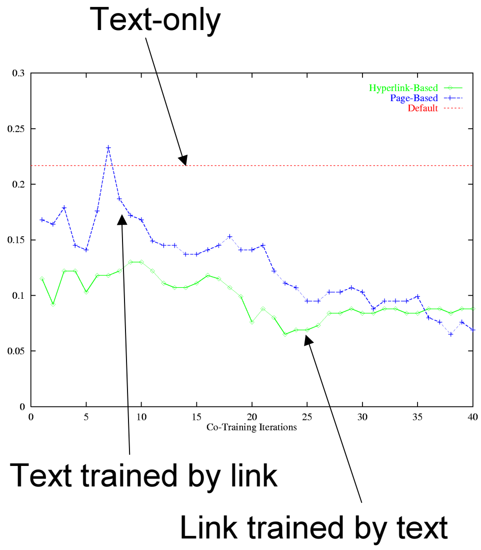
Unified model boosts accuracy

- 9600 patents from 12 classes marked by USPTO; text+links
- 'Forget' and re-estimate fraction of neighbors' classes (semi-supervised)
- 40% less error; even better for Yahoo
- Improvement even with 0% of neighborhood known



Co-training

- Divide features into two (nearly) class-conditionally independent sets, e.g. text and links
- Use labeled data to train two classifiers
- Repeat for each classifier
  - Find unlabeled instance which it can label most confidently
  - Add to the training set of the other classifier
- Accuracy improves barring local instabilities



Text-only

Text trained by link

Link trained by text

Modeling social networks

- The Web is a evolving social network
  - Other networks: phone calls, coauthored papers, citations, spreading infection,…
- Erdös-Renyi random graph model: each of *n*(*n* -1) edges created i.i.d. w.p. *p*
  - Threshold properties for number of connected components, connectivity
- Does not model social networks well:
  - The Web keeps growing (and changing)
  - Edge attachment is preferential (“winners take all”)
  - “Winning” nodes have high “prestige”

Preferential attachment

Goal: a simple, few/zero parameter evolution model for the Web

- Start with m_0 nodes
- Add one new node *u* every time step
- Connect new node to *m* old nodes
- Probability of picking old node v is $d_v \,/\, \Sigma_w\, d_w$, where *w* ranges over all old nodes
- Interesting questions:
  - How does the degree of a node evolve?
  - What is the degree distribution?

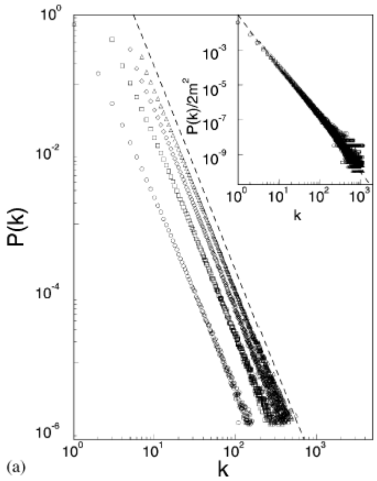
Model predictions and reality

- t_i is the time step when node *i* is added
- $k_i(t)$ = expected degree of node *i* at time step *t*

$$k_i(t) = m\sqrt{\frac{t}{t_i}}$$

$$P(k_i(t) = k) \approx \frac{2m^2 t}{(m_0 + t)}\frac{1}{k^3}$$

- Can we develop a notion of "prestige" to enhance Web search?



Google and PageRank

- Random surfer roaming for ever on the Web
- At page *u*, make one of two decisions
  - With probability *d*, jump to a Web page u.a.r
  - With probability 1-*d*, walk to a outlinked page *v*
- Irreducible, aperiodic Markov process
- Prestige of a node = steady state probability

$$p(v) = \frac{d}{|V|} + (1-d) \sum_{(u,v)\in E} \frac{p(u)}{\mathrm{g} \qquad (u)}$$

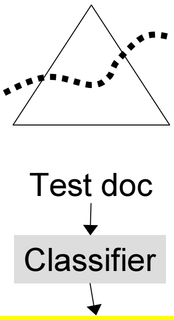
- Eigen problem involving the vertex adjacency matrix of the Web, solved by power iterations

Hubs and authorities

- Many Web pages are “link collections” (hubs)
  - Cites other authoritative pages but have no intrinsic information content
  - Similar to survey papers in academia
- Enhance the prestige model to **two** scores
  - Hub score *h*(*u*) for each node *u*
  - Authority score *a*(*v*) for each node *v*
- Coupled system: $\boldsymbol{a} = \boldsymbol{E}^T\boldsymbol{h}$ and $\boldsymbol{h} = \boldsymbol{E}\boldsymbol{a}$
  - In other words, $\boldsymbol{h} = \boldsymbol{E}\boldsymbol{E}^T\boldsymbol{h}$ and $\boldsymbol{a} = \boldsymbol{E}^T\boldsymbol{E}\boldsymbol{a}$
- Eigen problems of $\boldsymbol{E}\boldsymbol{E}^T$ and $\boldsymbol{E}^T\boldsymbol{E}$
  - Solved by power iterations

How to characterize “topics”

- Web directories—a natural choice
- Start with http://dmoz.org
- Keep pruning until all leaf topics have enough (>300) samples
- Approx 120k sample URLs
- Flatten to approx 482 topics
- Train text classifier (Rainbow)
- Characterize new document *d* as a vector of probabilities $\mathbf{p}_d = (\Pr(c|d)\ \forall c)$

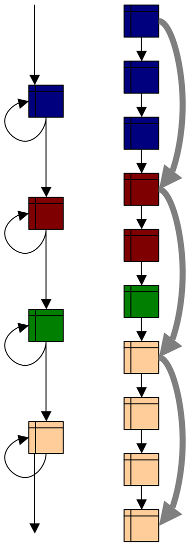

Test doc

Classifier

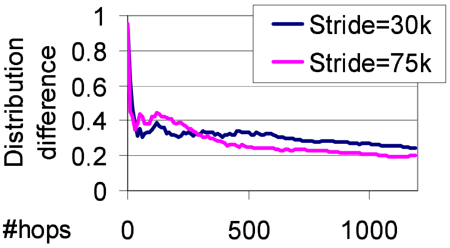
Topic	Prob
Arts	0.1
Computers	0.3
Science	0.6

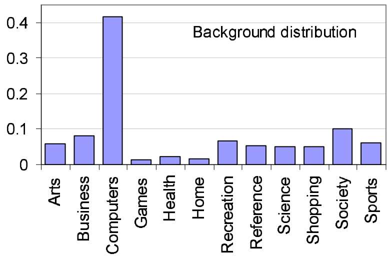
Background topic distribution

- What fraction of Web pages are about Health?
- Sampling via random walk
  - PageRank walk (Henzinger et al.)
  - Undirected regular walk (Bar-Yossef et al.)
- Make graph undirected (link:…)
- Add self-loops so that all nodes have the same degree
- Sample with large stride
- Collect topic histograms



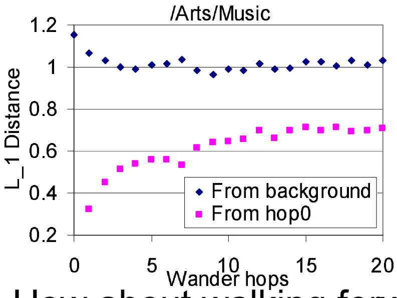
Convergence

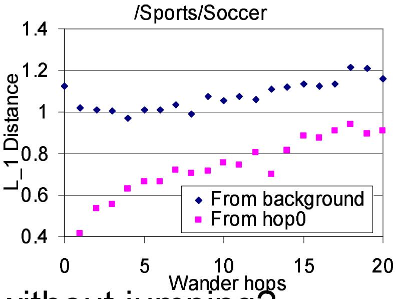




- Start from pairs of diverse topics
- Two random walks, sample from each walk
- Measure distance between topic distributions
  - L_1 distance $|\mathbf{p}_1 - \mathbf{p}_2| = \Sigma_c|p_1(c) - p_2(c)|$ in [0,2]
  - Below .05 —.2 within 300—400 physical pages

Random forward walk without jumps

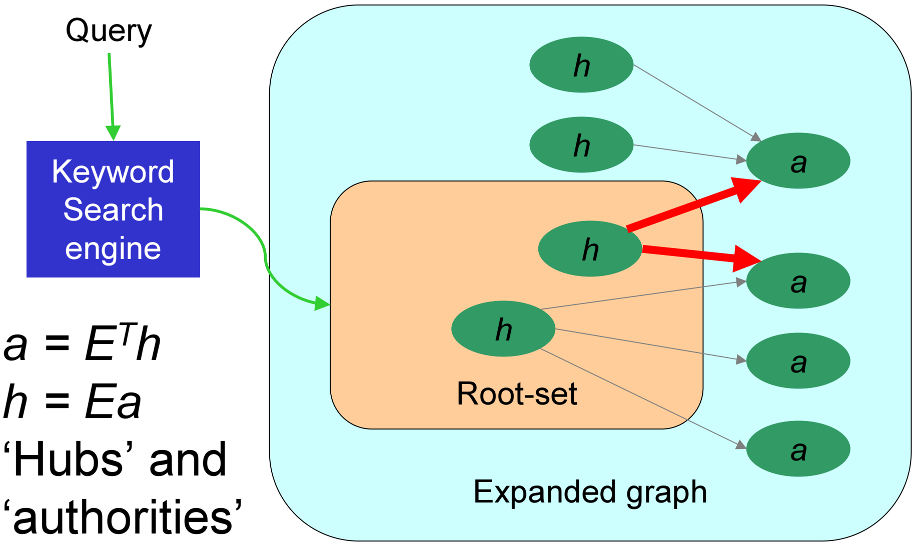




- How about walking forward without jumping?
  - Start from a page u_0 on a specific topic
  - Sample many forward random walks $(u_0, u_1, \ldots, u_i, \ldots)$
  - Compare $(\Pr(c|u_i)\ \forall c)$ with $(\Pr(c|u_0)\ \forall c)$ and with the background distribution
- Short-range topical locality on the Web

Hyperlink Induced Topic Search (HITS)

Also called “topic distillation”



Query

Keyword Search engine

Root-set

Expanded graph

$a = E^T h$

$h = Ea$

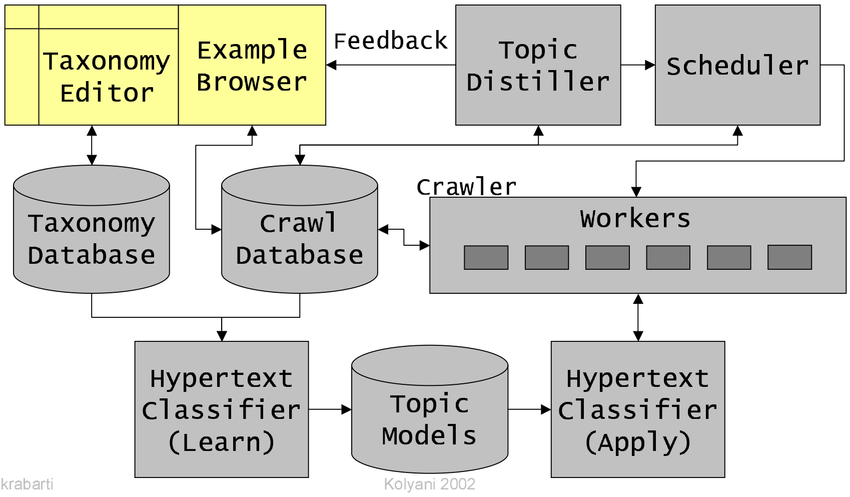
‘Hubs’ and ‘authorities’

Focused crawling

- HITS/PageRank on whole Web not very meaningful
- HITS expands root set by only one link
  - Two or more links introduce too much noise
- Can we filter out the noise?
  - Yes, using document classification
  - Can expand the graph indefinitely
- Formulation
  - Set of topics with examples, a chosen topic
  - Start from chosen examples, run for fixed time
  - Maximize total relevance of crawled pages w.r.t. chosen topic

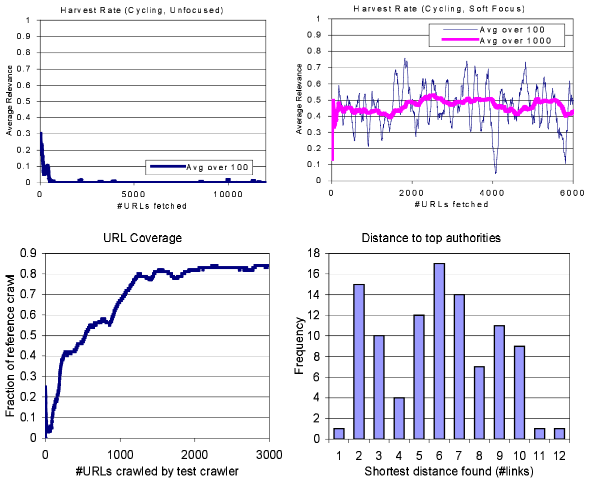
Focused crawling system overview

- If u is relevant and u→v then v is likely to be relevant
- If u→v1 and u→v2 and v1 is relevant then v2 is likely to be relevant



Focused crawling results

- High rate of "harvesting" relevant pages
- Robust to perturbations of starting URLs
- Great resources found 12 links from start set



Conclusion

- Application of statistics and mining to a new form of data: hypertext and the Web
- New challenges
  - Tackling a large number of dimensions
  - Modeling irregular, interrelated objects
  - Extracting signal from evolving, noisy data
  - Scaling language processing to the Web
- www.cse.iitb.ac.in/laiir/
- Mining the Web: Discovering Knowledge from Hypertext Data
  www.cse.iitb.ac.in/~soumen/mining-the-web/