Fast and accurate text classification via multiple linear discriminant projections

Soumen Chakrabarti
Shourya Roy
Mahesh Soundalgekar
IIT Bombay
www.cse.iitb.ac.in/~soumen

Introduction

- Supervised learning of labels from high-dimensional data has many applications
  - Text topic and genre classification
- Many classification algorithms known
  - Support vector machines (SVM)—most accurate
  - Maximum entropy classifiers
  - Naïve Bayes classifiers—fastest and simplest
- Problem: SVMs
  - Are difficult to understand and implement
  - Take time almost quadratic in #instances

Our contributions

- Simple Iterated Multiple Projection on Lines (SIMPL)
  - Trivial to understand and code (600 lines)
  - *O*(#dimensions) or less memory
  - Only sequential scans of training data
  - Almost as fast as naïve Bayes (NB)
  - As accurate as SVMs, sometimes better
- Insights into the best choice of linear discriminants for text classification
  - How do the discriminants chosen by NB, SIMPL and SVM differ?

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 is a linear discriminant

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

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

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

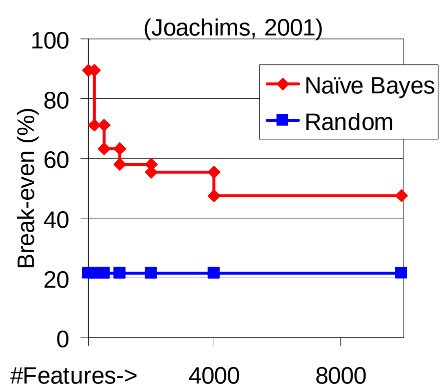
- The first part is a dot-product, the second part is a fixed offset, so we compare

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

- Simple join-aggregate, very fast

Many features, each fairly noisy

- Sort features in order of decreasing correlation with class labels
- Build separate classifiers
  - 1—100, 101—200, etc.
- Even features ranked 5000 to 10000 provide lift beyond picking a random class



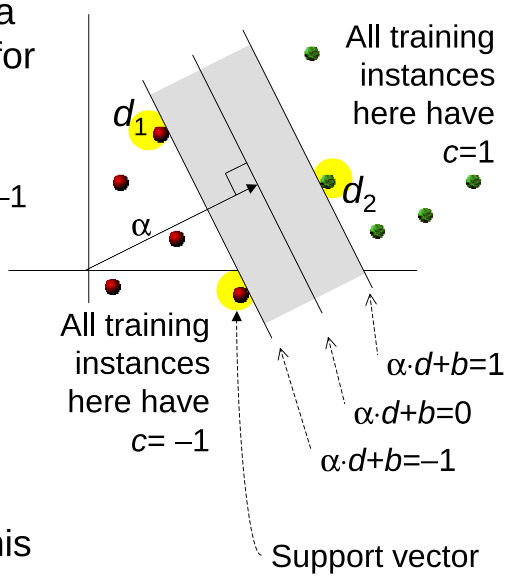
- Most features have tiny amounts of useful, noisy and possibly redundant info—how to combine?
- Naïve Bayes, LSVM, maximum entropy—all take linear combinations of term frequencies

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



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 seek+transfer

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)

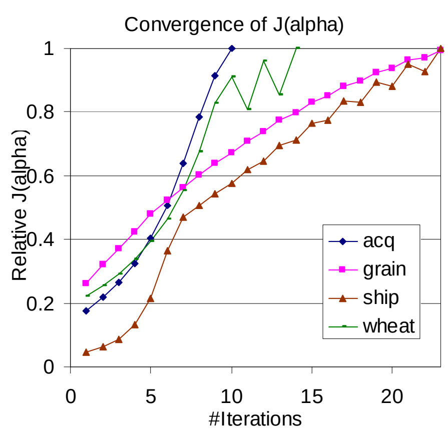
Hill-climbing

- Iteratively update $\alpha_{new} \leftarrow \alpha_{old} + \eta \nabla J(\alpha)$ 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

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

Convergence

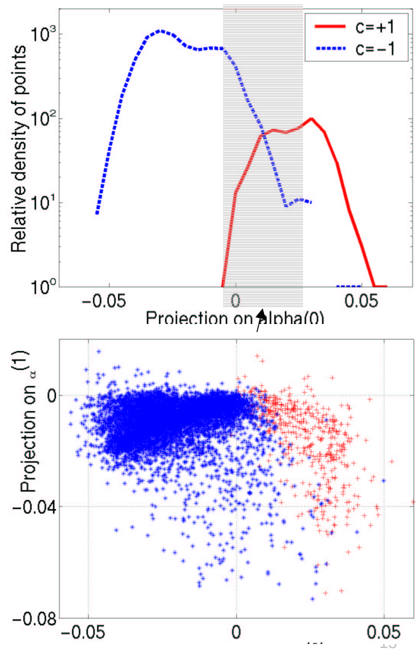
- Initialize α to vector joining positive and negative centroids
- Stop if $J(\alpha)$ cannot be increased in three successive iterations
- $J(\alpha)$ converges in 10—20 iterations
  - Not sensitive to problem size



- 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



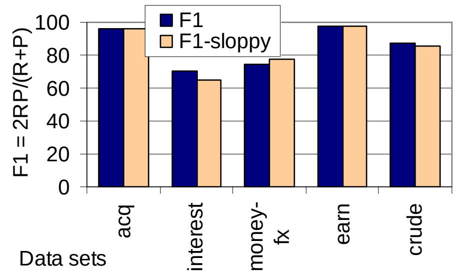
SIMPL (only 600 lines of C++)

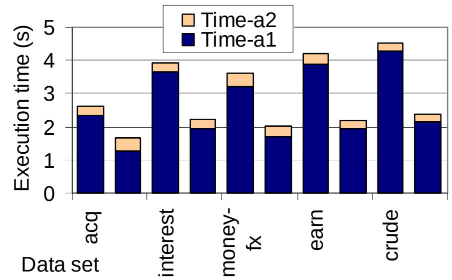
- Repeat for k = 0, 1, …
  - Find $\alpha^{(k)}$, the Fisher discriminant for the current set of training instances
  - Project training instances to $\alpha^{(k)}$
  - Remove points well-separated by $\alpha^{(k)}$

  while ≥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

Robustness of stopping decision

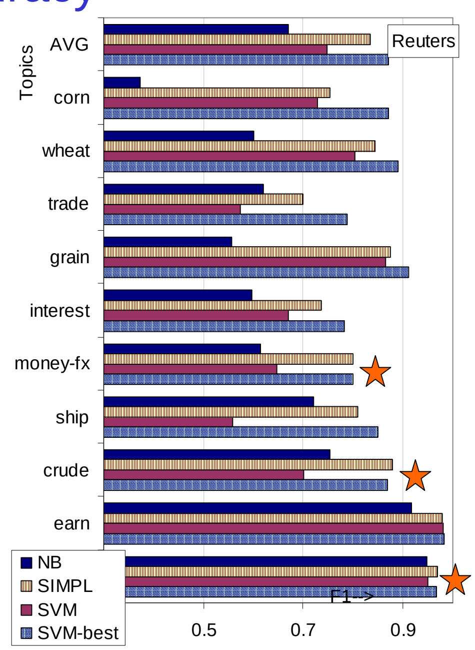
- Compute $\alpha^{(0)}$ to convergence
- Vs., run only half the iterations required for convergence
- Find $\alpha^{(1)}$,… as usual
- Later αs can cover for slop in earlier αs
- While saving time in costly early-α updates
  - Later αs take negligible time





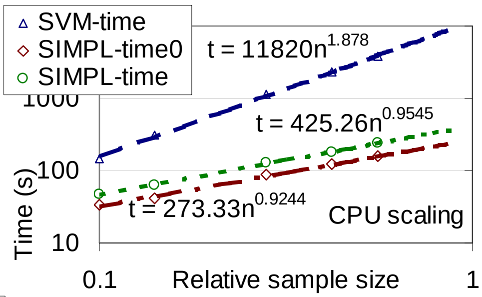
Accuracy

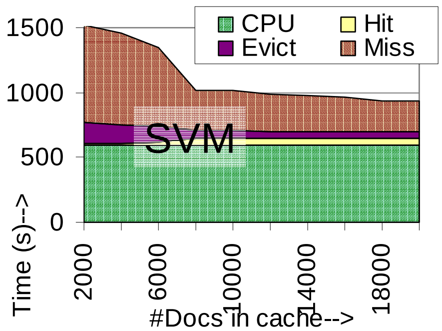
- Large improvement beyond naïve Bayes
- We tuned parameters in SVM to give "SVM-best"
- Often beats SVM with default params
- Almost always within 5% of SVM-best
- Even beats SVM-best in some cases
  - Especially when problem is not linearly separable

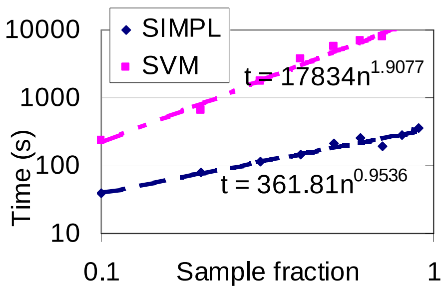


Performance

- 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







Summary and future work

- SIMPL: a new classifier for high-dimensional data
  - Low memory footprint, sequential scan
  - Orders of magnitude faster than LSVM
  - Often as accurate as LSVM, sometimes better
- An efficient “feature space transformer”
- How will SIMPL behave for non-textual, high-dim data?
- Can we analyze SIMPL?
  - LSVM is theoretically sound, more general
  - When will SIMPL match LSVM/SVM?