Introduction to

CS60092: Information Retrieval

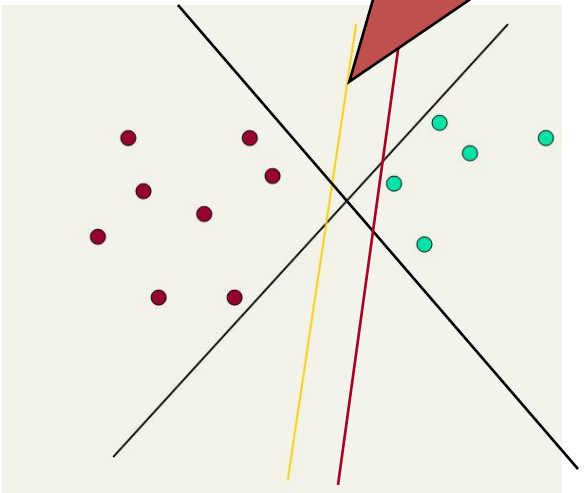
SVM and Ranking

Sourangshu Bhattacharya

Linear classifiers: Which Hyperplane?

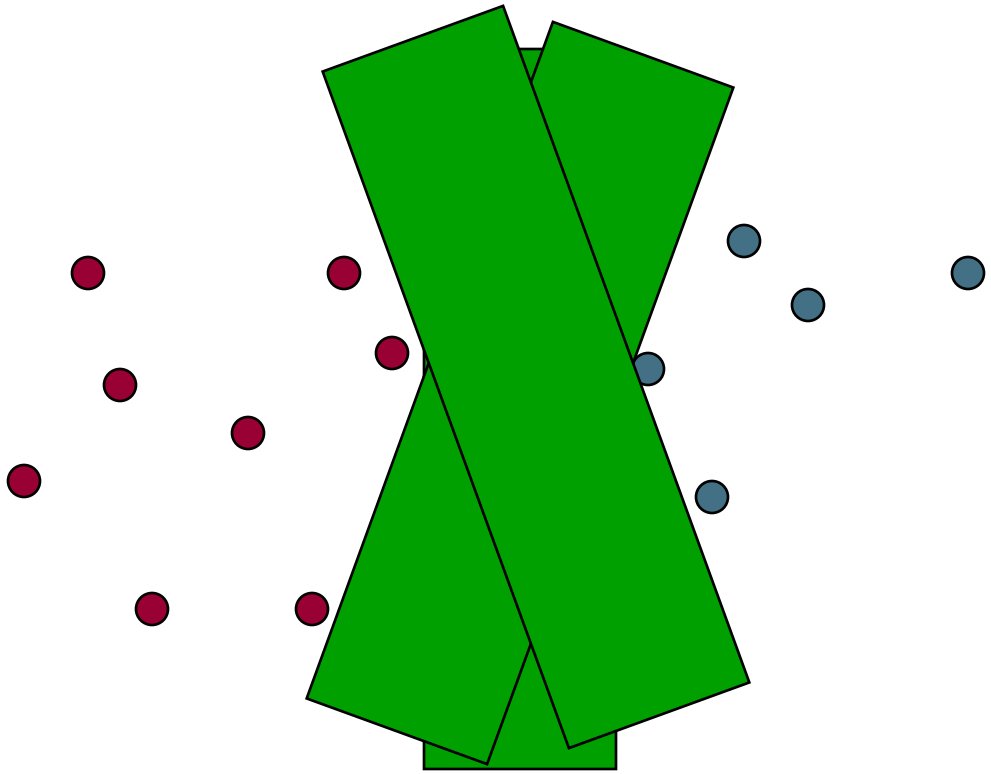
- Lots of possible choices for *a, b, c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- A Support Vector Machine (SVM) finds an optimal* solution.
  - Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:
$ax + by - c = 0$



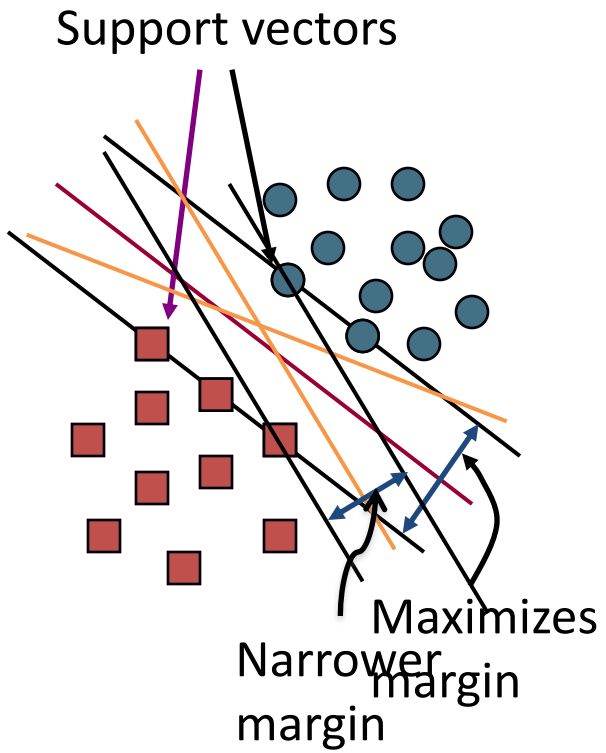
Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*



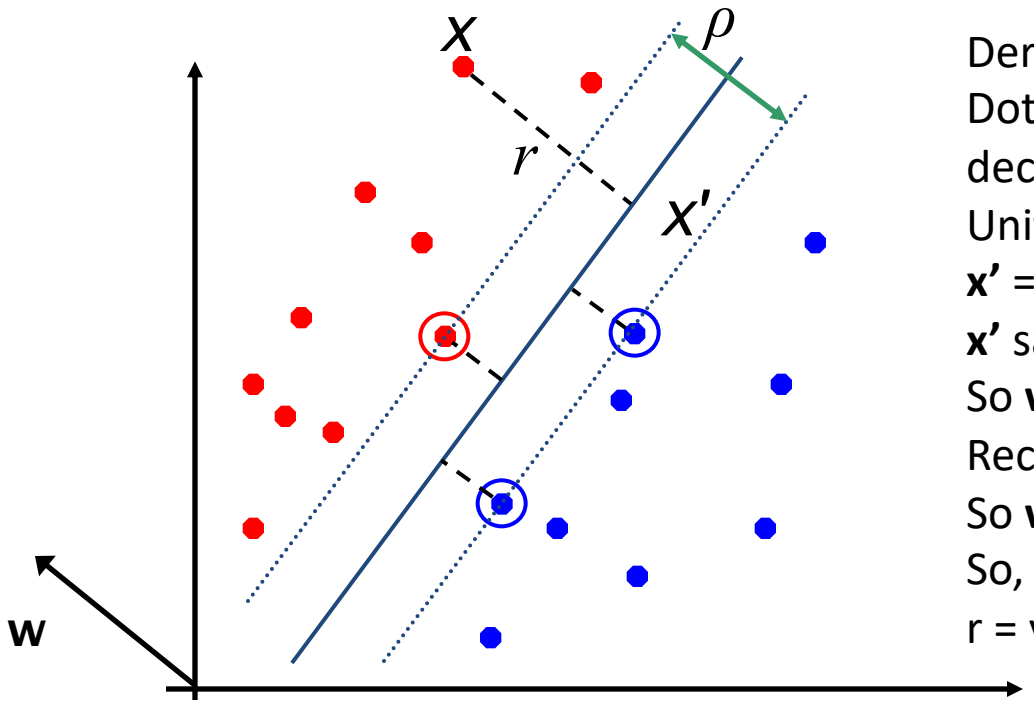
*but other discriminative methods often perform very similarly

Maximum Margin: Formalization

- **w**: decision hyperplane normal vector
- $\mathbf{x}_i$: data point *i*
- y_i: class of data point *i* (+1 or -1) NB: Not 1/0
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- Functional margin of $\mathbf{x}_i$ is: $y_i (\mathbf{w}^T\mathbf{x}_i + b)$
- The functional margin of a dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin
- **Problem:** we can increase this margin simply by scaling **w**, **b**….

Geometric Margin

- Distance from example to the separator is $r = y\dfrac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the width of separation between support vectors of classes.



Derivation of finding *r:*
Dotted line **x'** − **x** is perpendicular to decision boundary so parallel to **w**.
Unit vector is **w**/|**w**|, so line is r**w**/|**w**|.
x' = **x** – yr**w**/|**w**|.
x' satisfies $\mathbf{w}^T\mathbf{x}'$ + b = 0.
So $\mathbf{w}^T$(**x** –yr**w**/|**w**|) + b = 0
Recall that |**w**| = sqrt($\mathbf{w}^T\mathbf{w}$).
So $\mathbf{w}^T\mathbf{x}$ –yr|**w**| + b = 0
So, solving for r gives:
r = y($\mathbf{w}^T\mathbf{x}$ + b)/|**w**|

Linear SVM Mathematically

The linearly separable case

- Assume that the functional margin of each data item is at least 1, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

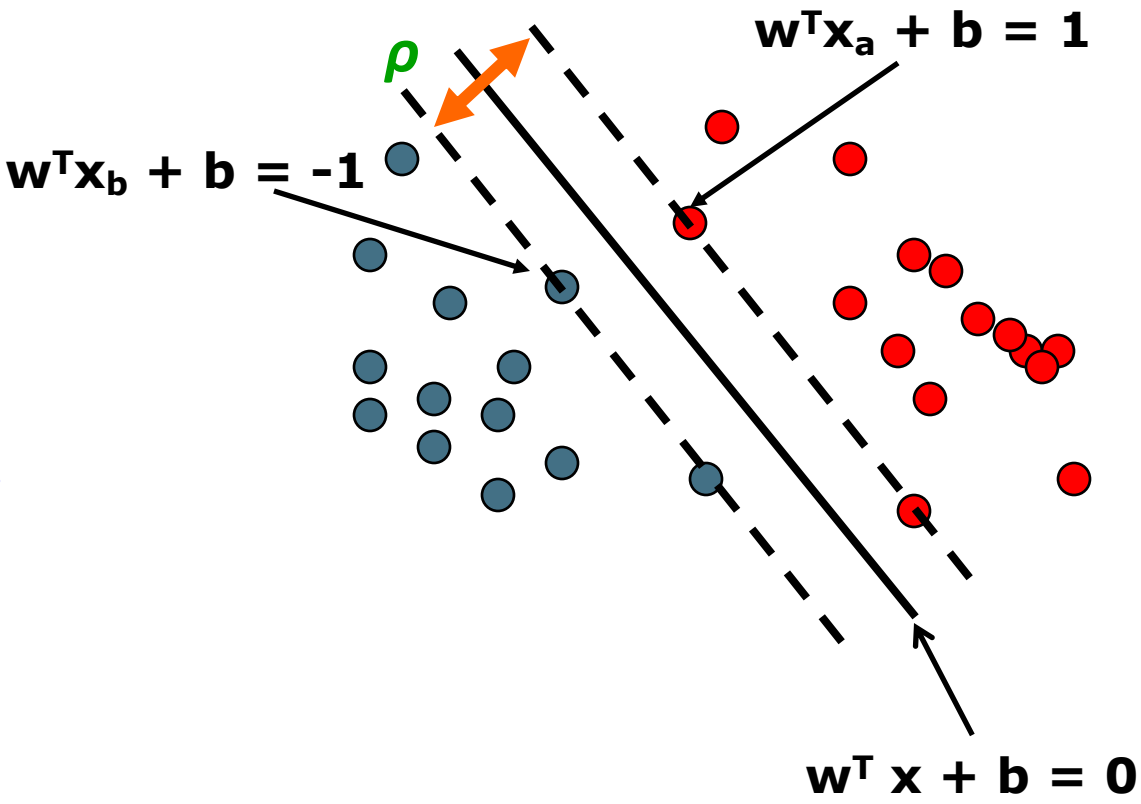
$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

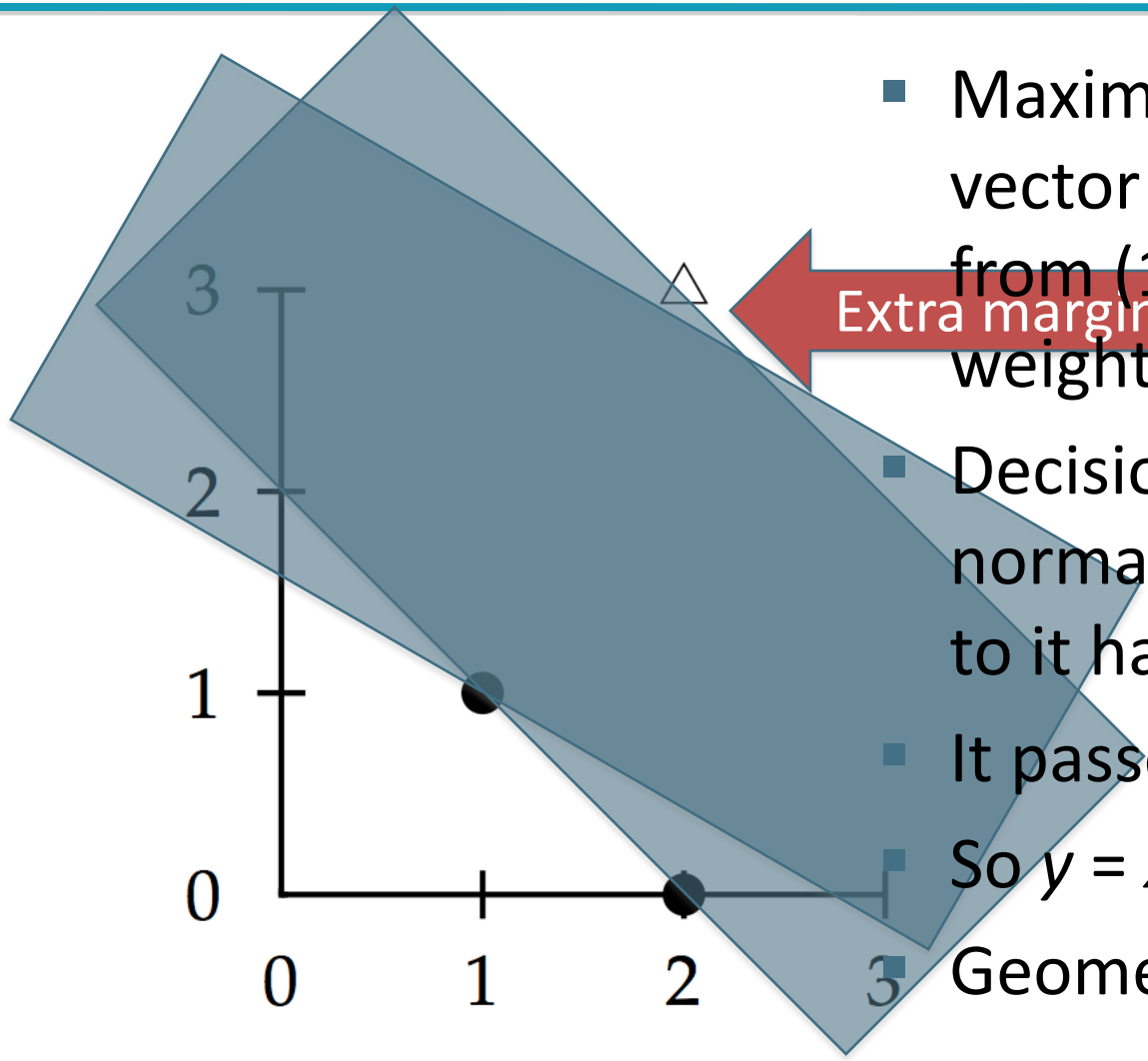
$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

- **Hyperplane**
  $\mathbf{w}^T \mathbf{x} + b = 0$
- **Extra scale constraint**:
  $\min_{i=1,\ldots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$
- This implies:
  $w^T(x_a - x_b) = 2$
  $\rho = \|x_a - x_b\|_2 = 2/\|\mathbf{w}\|_2$

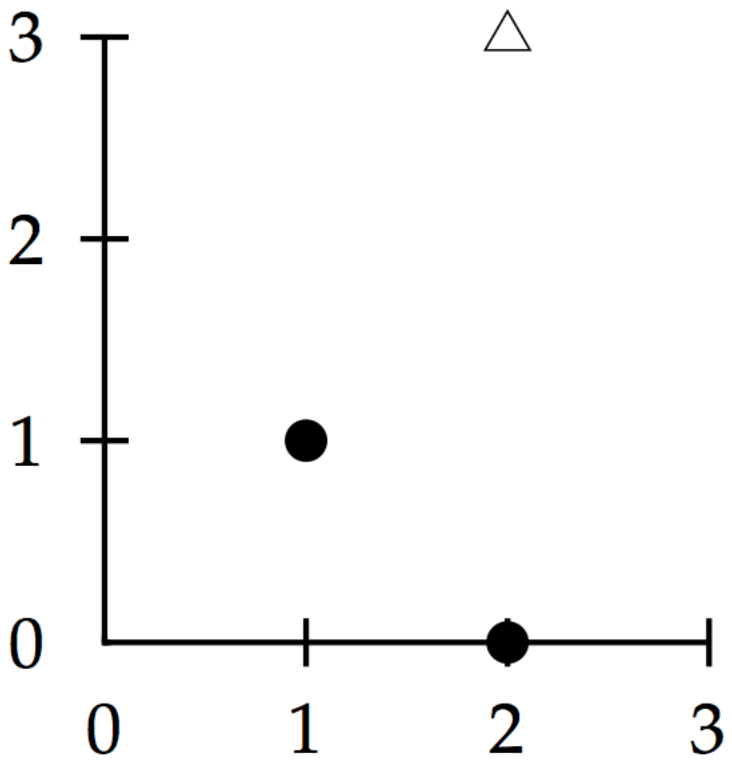


Worked example: Geometric margin



- Maximum margin weight vector is parallel to line from (1, 1) to (2, 3). So weight vector is (1, 2).
- Decision boundary is normal ("perpendicular") to it halfway between.
- It passes through (1.5, 2)
- So $y = x_1 + 2x_2 - 5.5$
- Geometric margin is $\sqrt{5}$

Worked example: Functional margin



- Let's minimize w given that $y_i(w^T x_i + b) \geq 1$
- Constraint has = at SVs; $w = (a, 2a)$ for some a
- $a+2a+b = -1$ $\quad 2a+6a+b = 1$
- So, $a = 2/5$ and $b = -11/5$
  Optimal hyperplane is:
  $w = (2/5, 4/5)$ and $b = -11/5$
- Margin ρ is $2/|w|$
  $= 2/\sqrt{(4/25+16/25)}$
  $= 2/(2\sqrt{5}/5) = \sqrt{5}$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i=1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min $\|\mathbf{w}\|$= max $1/\|\mathbf{w}\|$):

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \sum\alpha_i - \frac{1}{2}\sum\sum\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\sum\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

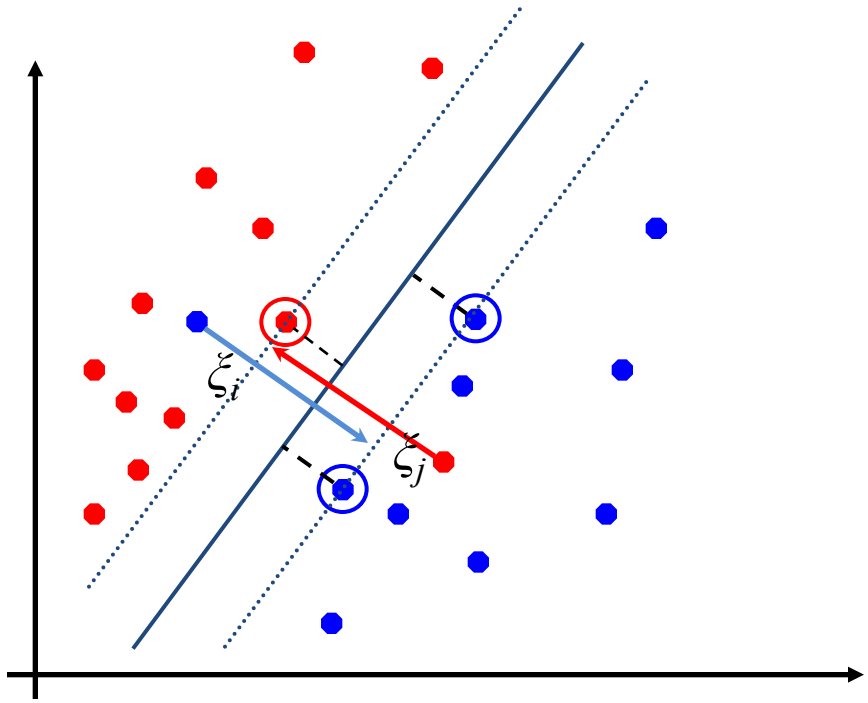
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting
  - A regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x_i}$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$
$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x}_k$ where $k = \underset{k'}{\text{argmax}}\ \alpha_{k'}$

w is not needed explicitly for classification!

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$

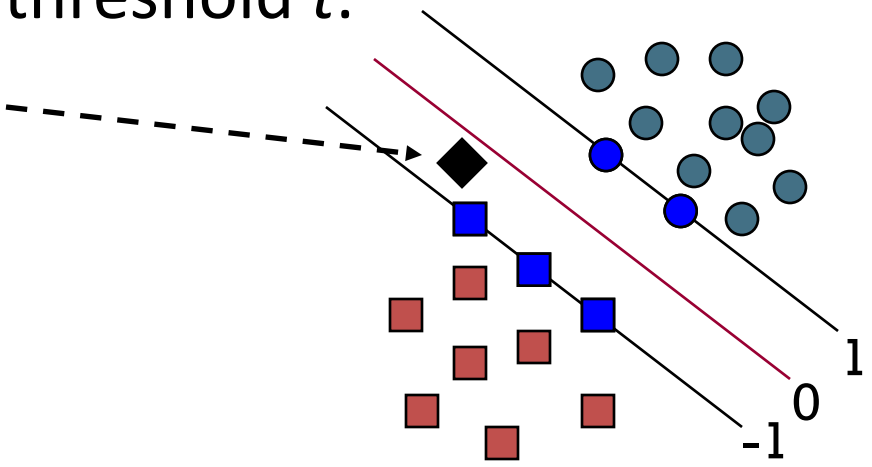
Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $\mathbf{w}^T\mathbf{x} + b = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$
    - Decide class based on whether < or > 0
  - Can set confidence threshold *t*.

Score > *t*: yes

Score < -*t*: no

Else: don't know



Linear SVMs: Summary

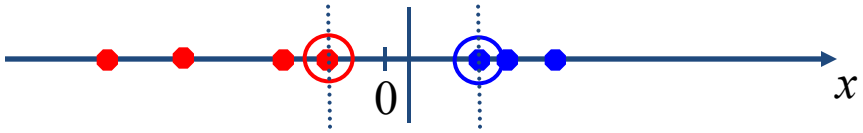
- The classifier is a *separating hyperplane.*
- The most "important" training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

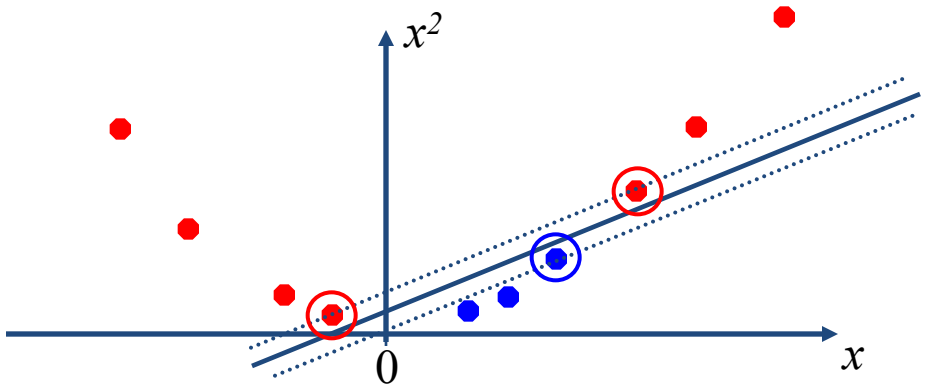
Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:



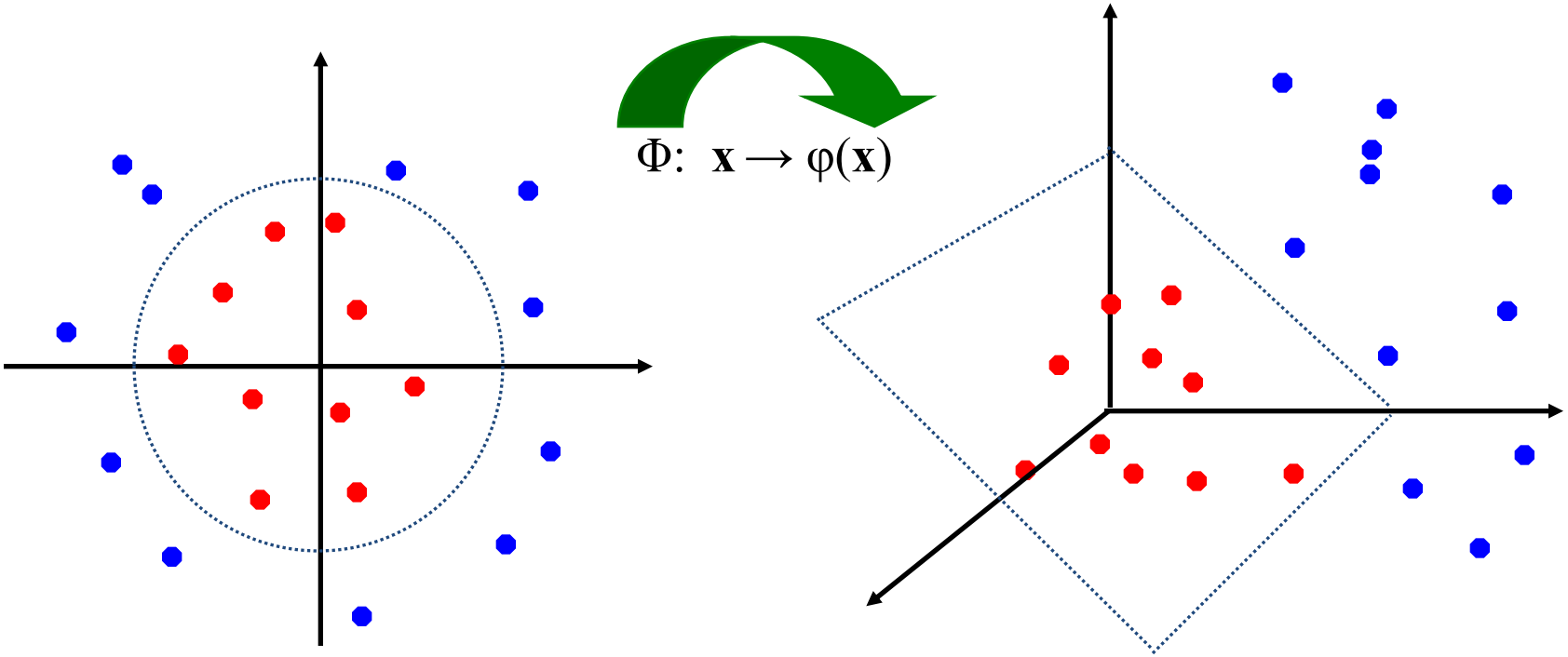
- But what are we going to do if the dataset is just too hard?

- How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T\, [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

  $= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ where $\phi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial **$K(x,z) = (1+x^Tz)^d$**
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2}$$

- Haven't been very useful in text classification

Multi-class SVM

Intuitive formulation: without regularization / for the separable case

$$\max_W \left[\sum_i {w_{y^i}}^\top x^i - \max_j (\mathbf{1}\{j \neq y^i\} + w_j^\top x^i) \right]$$

Primal problem: QP

$$\min_{w_1,\ldots,w_K} \quad \frac{1}{2}\|(w_1,\ldots,w_K)\|^2 + C\sum_{ik}\xi_{ik}$$

$$\text{s.t.} \quad \forall (i,k), \quad {w_{y^i}}^\top x^i - w_k^\top x^i \geq \mathbf{1}\{k \neq y^i\} - \xi_{ik}$$

Solved in the dual formulation, also Quadratic Program

Main advantage: Sparsity (but not systematic)

- Speed with SMO (heuristic use of sparsity)
- Sparse solutions

Drawbacks:

- Need to recalculate or store $x_i^T x_j$
- Outputs not probabilities

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>

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

(a)

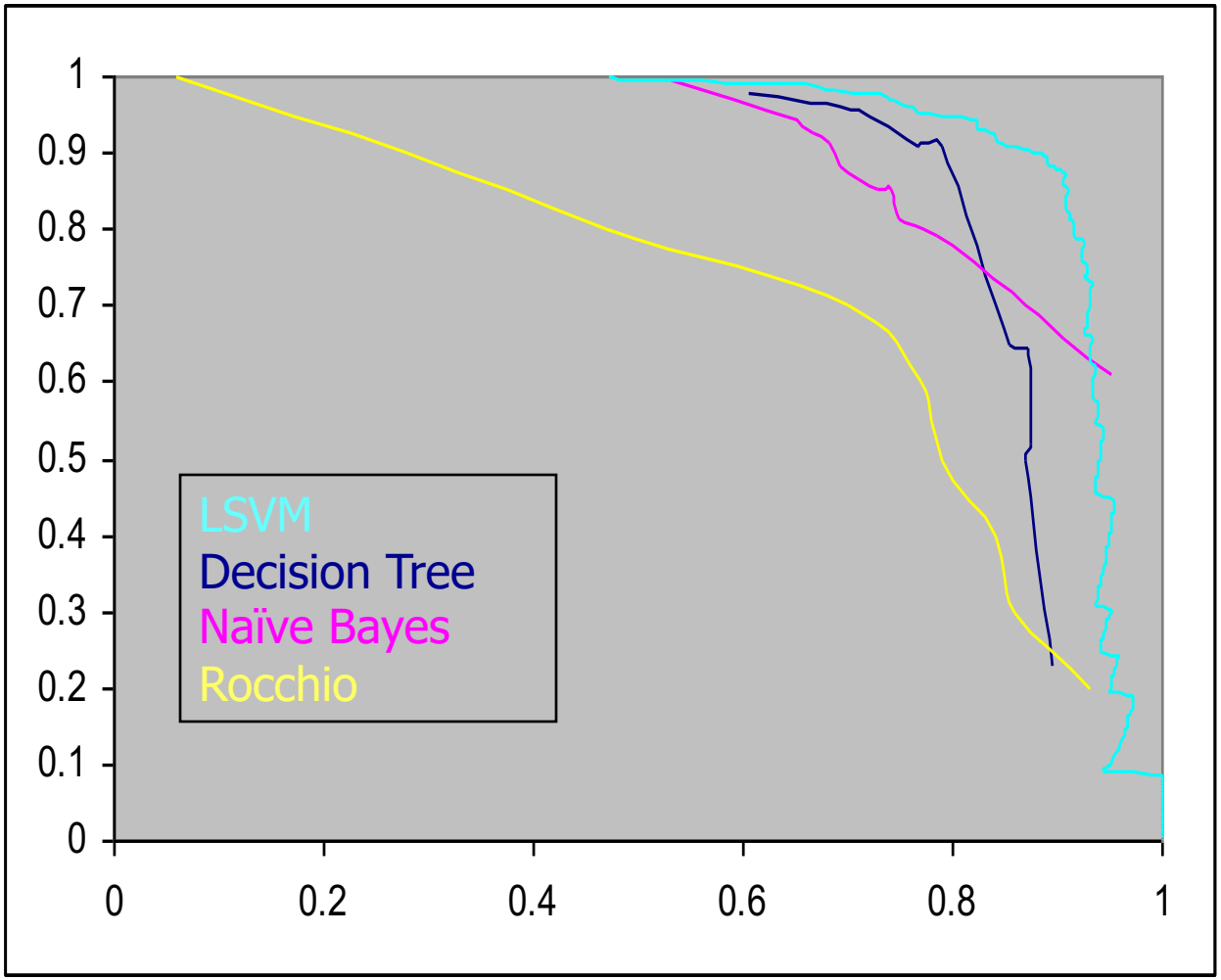
	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(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

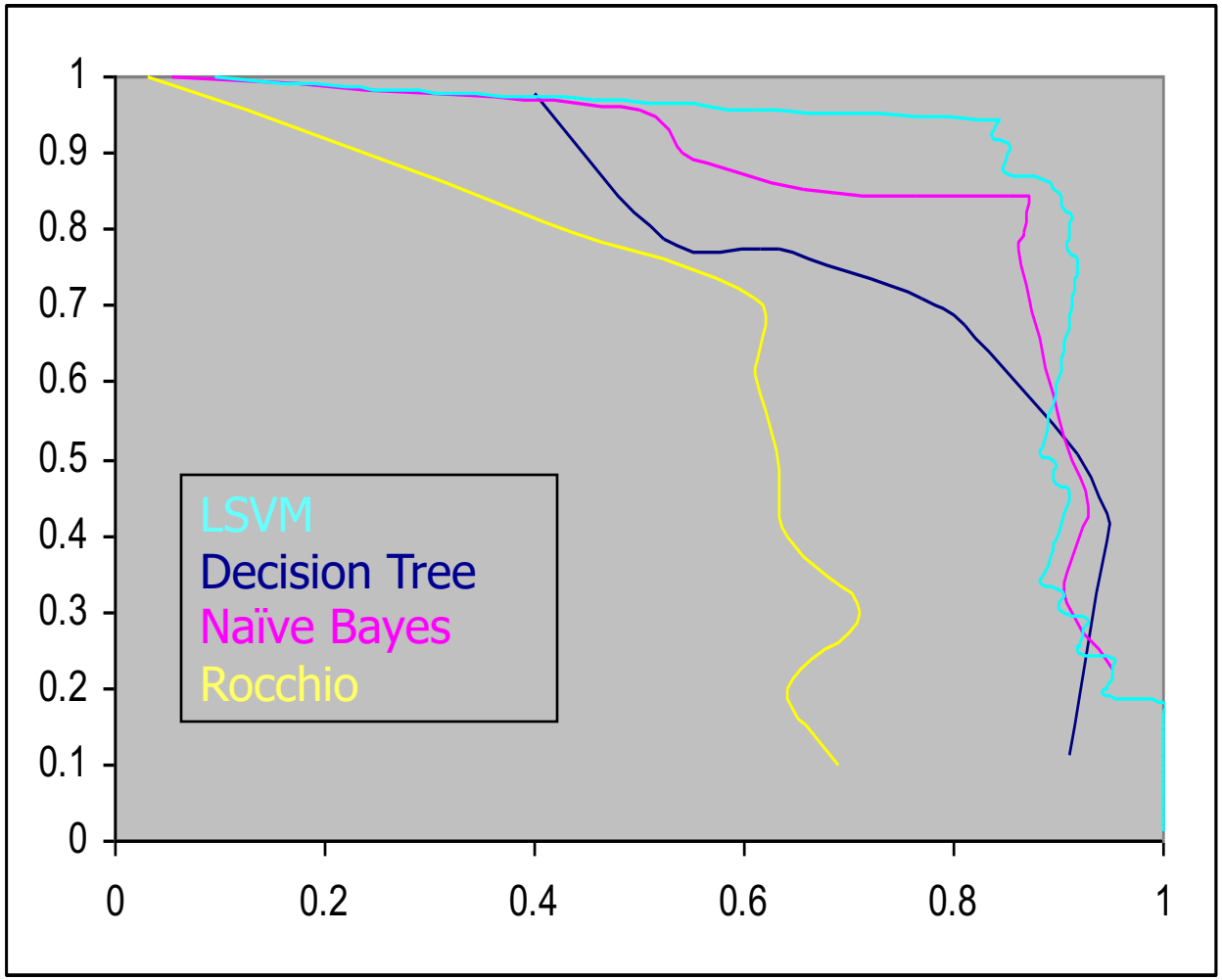
Evaluation measure: F_1

Precision-recall for category: Crude



Dumais (1998)

Precision-recall for category: Ship



Dumais (1998)

Yang&Liu: SVM vs. Other Methods

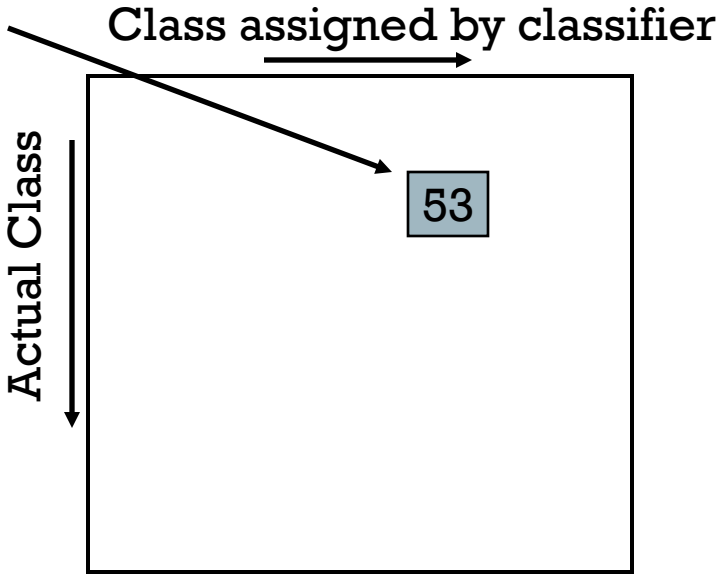
Table 1: Performance summary of classifiers

method	miR	miP	miF1	maF1	error
SVM	.8120	.9137	.8599	.5251	.00365
KNN	.8339	.8807	.8567	.5242	.00385
LSF	.8507	.8489	.8498	.5008	.00414
NNet	.7842	.8785	.8287	.3765	.00447
NB	.7688	.8245	.7956	.3886	.00544

miR = micro-avg recall; miP = micro-avg prec.;
miF1 = micro-avg F1; maF1 = macro-avg F1.

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

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 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 the SVM!

- 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

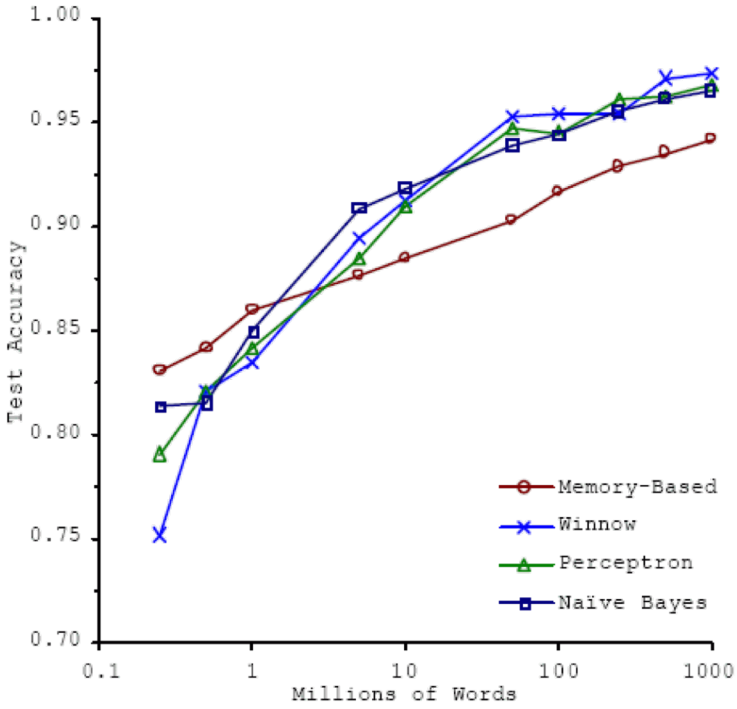
A huge amount of data?

- 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 quite impractical

- Naïve Bayes can come back into its own again!
  - Or other advanced 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



Summary

- Support vector machines (SVM)
  - Choose hyperplane based on support vectors
    - Support vector = "critical" point close to decision boundary
  - (Degree-1) SVMs are linear classifiers.
  - Kernels: powerful and elegant way to define similarity metric
  - Perhaps best performing text classifier
    - But there are other methods that perform about as well as SVM, such as regularized logistic regression (Zhang & Oles 2001)
  - Partly popular due to availability of good software
    - SVMlight is accurate and fast – and free (for research)
    - Now lots of good software: libsvm, TinySVM, ….
- Comparative evaluation of methods
- Real world: exploit domain specific structure!

Resources for today's lecture

- Christopher J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition
- S. T. Dumais. 1998. Using SVMs for text categorization, IEEE Intelligent Systems, 13(4)
- S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *CIKM '98*, pp. 148-155.
- 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/

Machine learning for IR ranking

- There's some truth to the fact that the IR community wasn't very connected to the ML community
- But there were a whole bunch of precursors:
  - Wong, S.K. et al. 1988. Linear structure in information retrieval. *SIGIR 1988.*
  - Fuhr, N. 1992. Probabilistic methods in information retrieval. *Computer Journal.*
  - Gey, F. C. 1994. Inferring probability of relevance using the method of logistic regression. *SIGIR 1994.*
  - Herbrich, R. et al. 2000. Large Margin Rank Boundaries for Ordinal Regression. *Advances in Large Margin Classifiers.*

Why weren't early attempts very successful/influential?

- Sometimes an idea just takes time to be appreciated…
- **Limited training data**
  - Especially for real world use (as opposed to writing academic papers), it was very hard to gather test collection queries and relevance judgments that are representative of real user needs and judgments on documents returned
    - This has changed, both in academia and industry
- Poor machine learning techniques
- Insufficient customization to IR problem
- Not enough features for ML to show value

Why wasn't ML much needed?

- Traditional ranking functions in IR used a very small number of features, e.g.,
  - Term frequency
  - Inverse document frequency
  - Document length
- It was easy to tune weighting coefficients by hand
  - And people did
  - You guys did in PA3
    - Some of you even grid searched a bit

Why is ML needed now?

- Modern systems – especially on the Web – use a great number of features:
  - Arbitrary useful features – not a single unified model
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains "~"?
  - Page edit recency?
  - Page length?
- The *New York Times* (2008-06-03) quoted Amit Singhal as saying Google was using over 200 such features.

Simple example: Using classification for ad hoc IR

- Collect a training corpus of (*q, d, r*) triples
  - Relevance *r* is here binary (but may be multiclass, with 3–7 values)
  - Document is represented by a feature vector
    - **x** = (α, ω) α is cosine similarity, ω is minimum query window size
      - ω is the the shortest text span that includes all query words
      - Query term proximity is a **very important** new weighting factor
  - Train a machine learning model to predict the class *r* of a document-query pair

example	docID	query	cosine score	ω	judgment
Φ_1	37	linux operating system	0.032	3	*relevant*
Φ_2	37	penguin logo	0.02	4	*nonrelevant*
Φ_3	238	operating system	0.043	2	*relevant*
Φ_4	238	runtime environment	0.004	2	*nonrelevant*
Φ_5	1741	kernel layer	0.022	3	*relevant*
Φ_6	2094	device driver	0.03	2	*relevant*
Φ_7	3191	device driver	0.027	5	*nonrelevant*

Simple example: Using classification for ad hoc IR

- A linear score function is then

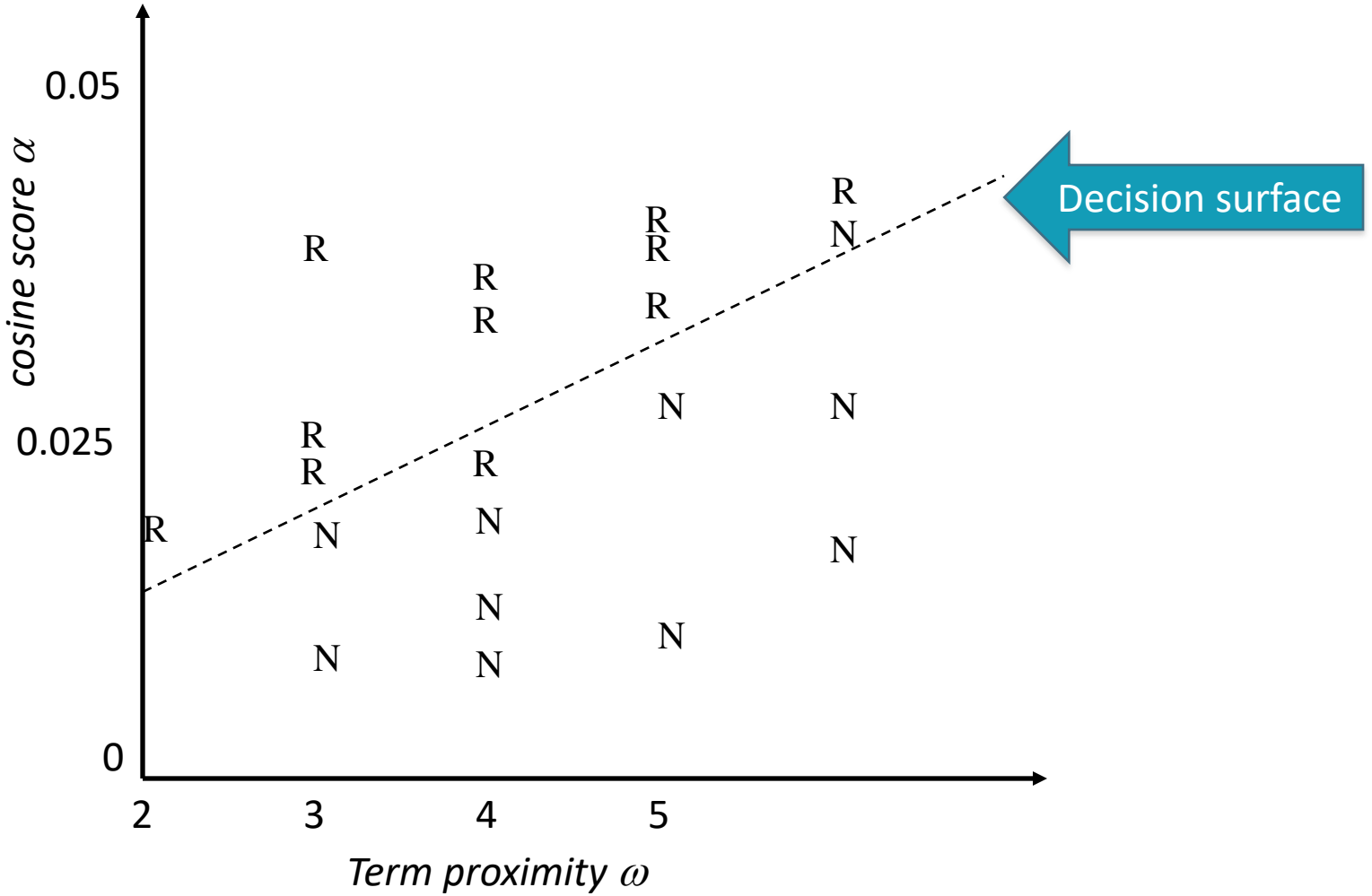
$$Score(d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c$$

- And the linear classifier is

Decide relevant if $Score(d, q) > \theta$

- … just like when we were doing text classification

Simple example: Using classification for ad hoc IR



More complex example of using classification for search ranking [Nallapati 2004]

- We can generalize this to classifier functions over more features
- We can use methods we have seen previously for learning the linear classifier weights

An SVM classifier for information retrieval

[Nallapati 2004]

- Let $g(r|d,q) = \mathbf{w} \bullet f(d,q) + b$
- SVM training: want $g(r|d,q) \leq -1$ for nonrelevant documents and $g(r|d,q) \geq 1$ for relevant documents
- SVM testing: decide relevant iff $g(r|d,q) \geq 0$

- Features are *not* word presence features (how would you deal with query words not in your training data?) but scores like the summed (log) tf of all query terms
- Unbalanced data (which can result in trivial always-say-nonrelevant classifiers) is dealt with by undersampling nonrelevant documents during training (just take some at random) [there are other ways of doing this – cf. Cao et al. later]

An SVM classifier for information retrieval

[Nallapati 2004]

- Experiments:
  - 4 TREC data sets
  - Comparisons with Lemur, a state-of-the-art open source IR engine (Language Model (LM)-based – see *IIR* ch. 12)
  - Linear kernel normally best or almost as good as quadratic kernel, and so used in reported results
  - 6 features, all variants of tf, idf, and tf.idf scores

An SVM classifier for information retrieval [Nallapati 2004]

Train \ Test		Disk 3	Disk 4-5	WT10G (web)
Disk 3	LM	**0.1785**	**0.2503**	0.2666
	SVM	0.1728	0.2432	**0.2750**
Disk 4-5	LM	**0.1773**	**0.2516**	0.2656
	SVM	0.1646	0.2355	**0.2675**

- At best the results are about equal to LM
  - Actually a little bit below
- Paper's advertisement: Easy to add more features
  - This is illustrated on a homepage finding task on WT10G:
  - Baseline LM 52% success@10, baseline SVM 58%
  - SVM with URL-depth, and in-link features: 78% S@10

"Learning to rank"

- Classification probably isn't the right way to think about approaching ad hoc IR:
  - Classification problems: Map to a unordered set of classes
  - Regression problems: Map to a real value [Start of PA4]
  - Ordinal regression problems: Map to an *ordered* set of classes
    - A fairly obscure sub-branch of statistics, but what we want here
- This formulation gives extra power:
  - Relations between relevance levels are modeled
  - Documents are good versus other documents for query given collection; not an absolute scale of goodness

"Learning to rank"

- Assume a number of categories ***C*** of relevance exist
  - These are totally ordered: $c_1 < c_2 < \ldots < c_J$
  - This is the ordinal regression setup
- Assume training data is available consisting of document-query pairs represented as feature vectors ψ_i and relevance ranking c_i

- We could do ***point-wise* learning**, where we try to map items of a certain relevance rank to a subinterval (e.g, Crammer et al. 2002 Prank, 2005 Chu and Keerthi)
- But most work does ***pair-wise* learning**, where the input is a pair of results for a query, and the class is the relevance ordering relationship between them

Pointwise learning

[Chu and Keerthi 2005]

- Given *r* ranks and and n_j examples for the j^{th} rank.
- Let the training examples be x_i, $i=1,\ldots,n_j$
- Parameters *w* and b_j can be learned by solving:

$$\min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\xi},\boldsymbol{\xi}^*} \ \tfrac{1}{2}\langle \boldsymbol{w}\cdot\boldsymbol{w}\rangle + C\sum_{j=1}^{r}\sum_{i=1}^{n^j}\left(\xi_i^j + \xi_i^{*j}\right)$$

subject to

$$\begin{aligned}
&\langle \boldsymbol{w}\cdot\phi(x_i^j)\rangle - b_j \le -1 + \xi_i^j, && \xi_i^j \ge 0,\ \forall i,j;\\
&\langle \boldsymbol{w}\cdot\phi(x_i^j)\rangle - b_{j-1} \ge +1 - \xi_i^{*j}, && \xi_i^{*j} \ge 0,\ \forall i,j;\\
&b_{j-1} \le b_j,\ \forall j.
\end{aligned}$$

Pointwise learning

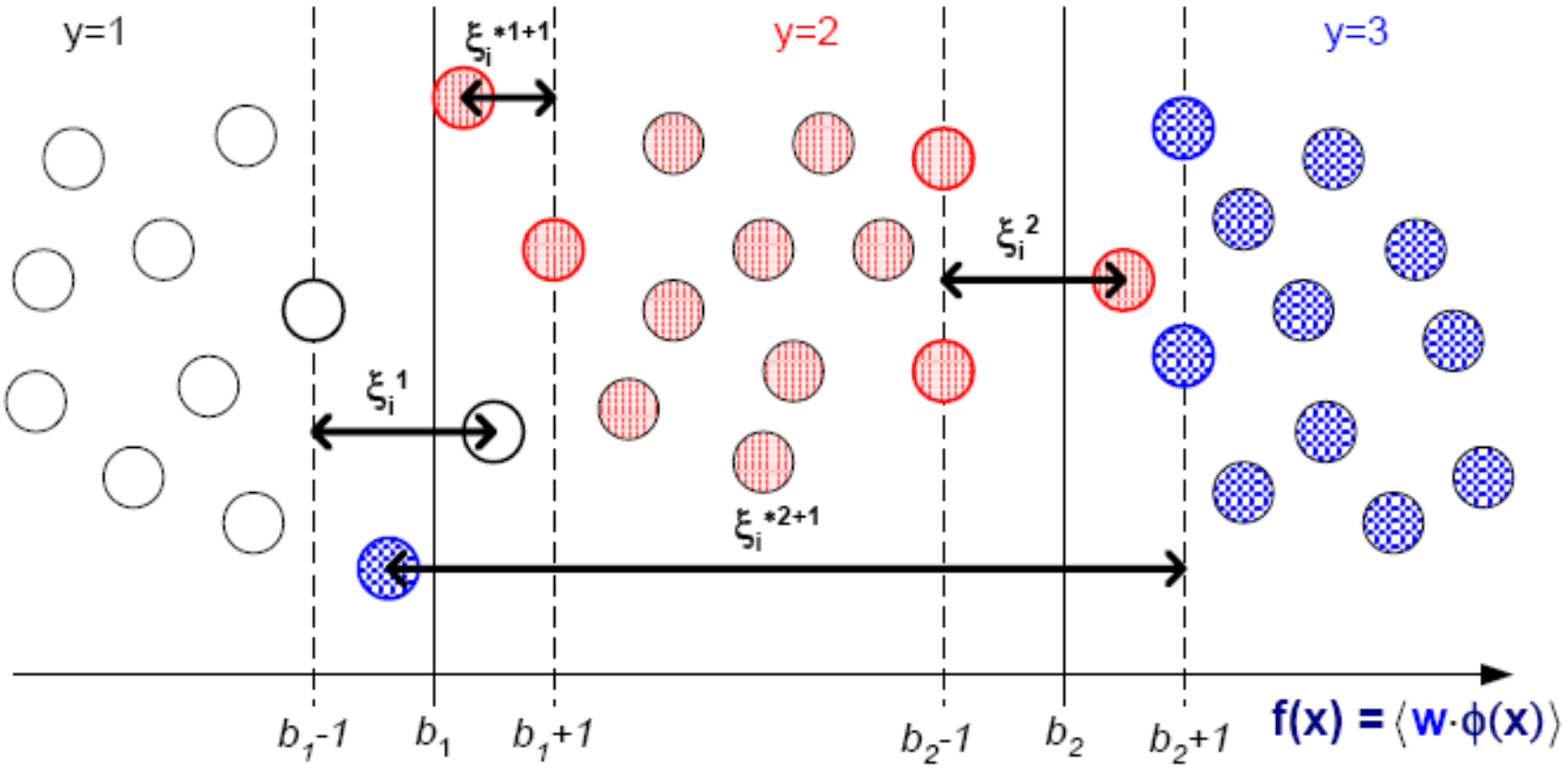
[Chu and Keerthi 2005]

- Given a new example x, its rank can be predicted as:
  $rank = argmin_j \, w^T x < b_j$

Pointwise learning

[Chu and Keerthi 2005]

- Goal is to learn a threshold to separate each rank



Pairwise learning: The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Aim is to classify instance pairs as correctly ranked or incorrectly ranked
  - This turns an ordinal regression problem back into a binary classification problem
- We want a ranking function f such that

$$c_i > c_k \text{ iff } f(\psi_i) > f(\psi_k)$$

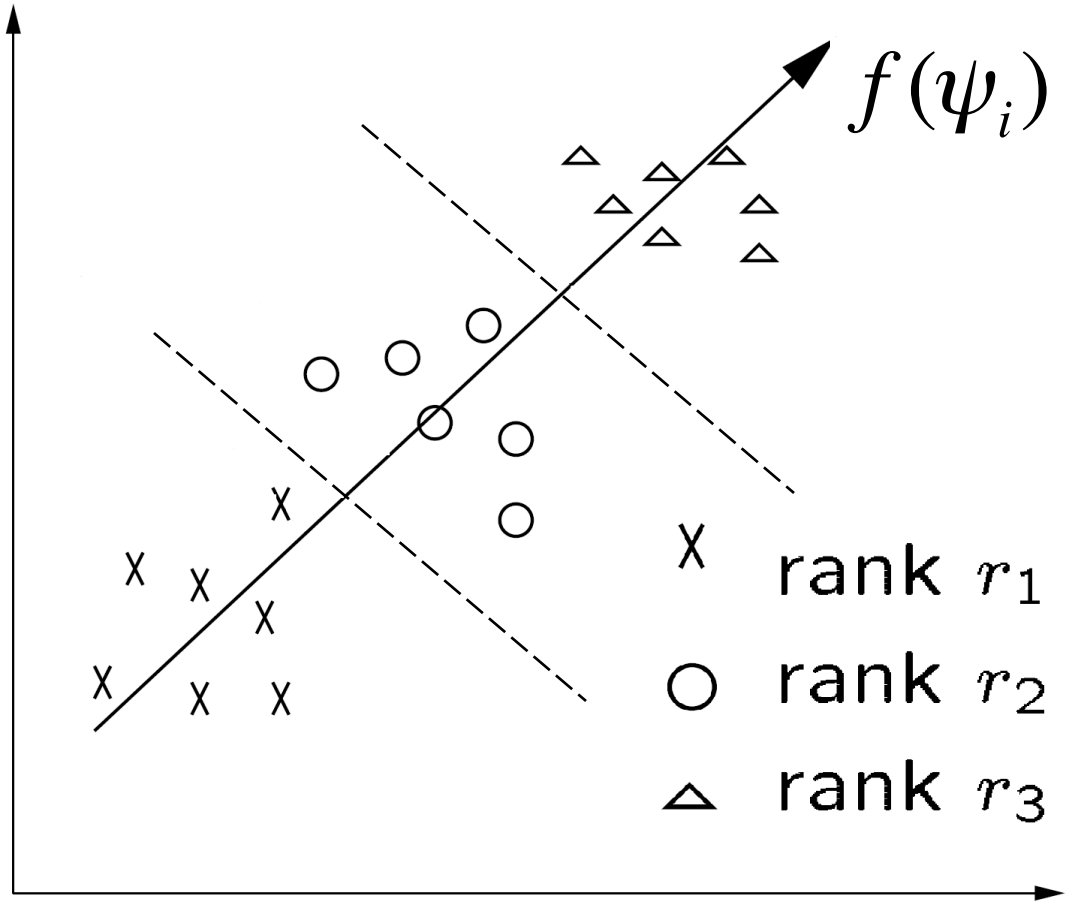
- … or at least one that tries to do this with minimal error
- Suppose that f is a linear function

$$f(\psi_i) = \mathbf{w} \bullet \psi_i$$

The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Ranking Model: $f(\psi_i)$



The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Then (combining the two equations on the last slide):

$$c_i > c_k \text{ iff } \mathbf{w}\bullet(\psi_i - \psi_k) > 0$$

- Let us then create a new instance space from such pairs:

$$\Phi_u = \Phi(d_i, d_j, q) = \psi_i - \psi_k$$

$$z_u = +1, 0, -1 \text{ as } c_i >,=,< c_k$$

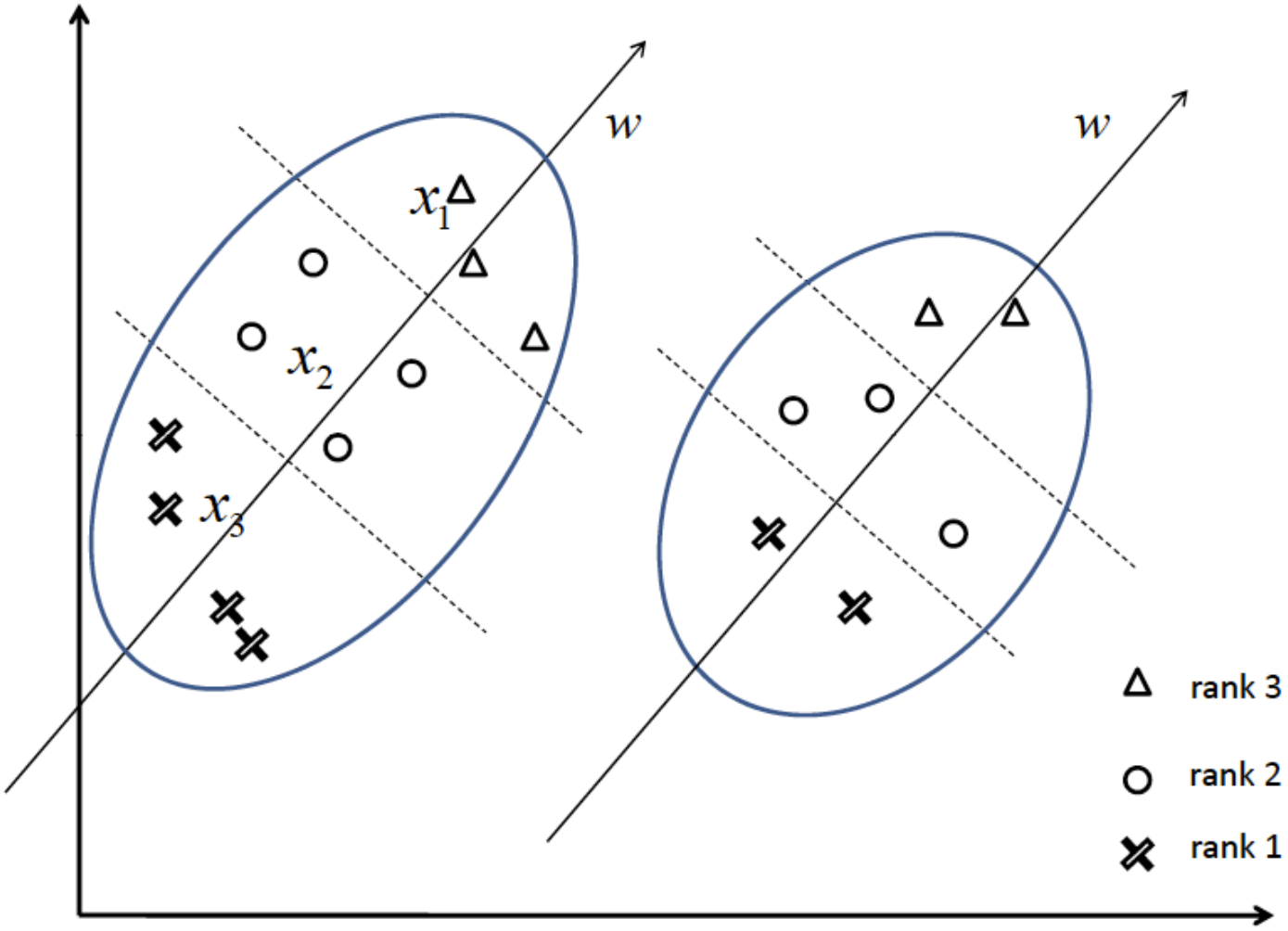
- We can build model over just cases for which $z_u = -1$
- From training data $S = \{\Phi_u\}$, we train an SVM

The Ranking SVM

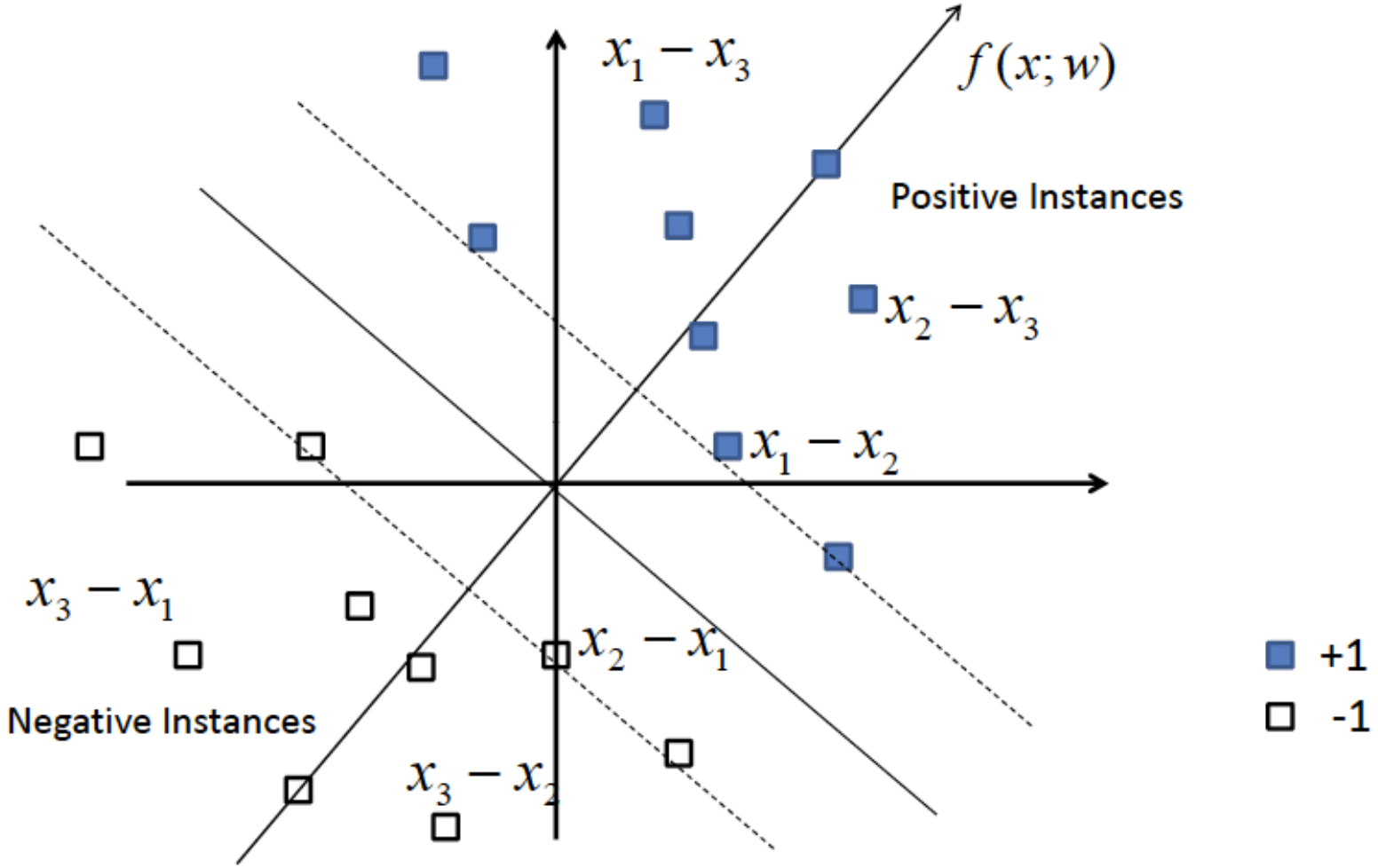
[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- The SVM learning task is then like other examples that we saw before
- Find $\mathbf{w}$ and $\xi_u \geq 0$ such that
  - $\frac{1}{2}\mathbf{w}^T\mathbf{w} + C \Sigma \xi_u$ is minimized, and
  - for all Φ_u such that $z_u < 0$, $\mathbf{w} \bullet \Phi_u \geq 1 - \xi_u$
- We can just do the negative z_u, as ordering is antisymmetric
- You can again use SVMlight (or other good SVM libraries) to train your model (SVMrank specialization)

Two queries in the original space



Two queries in the pairwise space



Aside: The SVM loss function

- The minimization

$$\min_{\mathbf{w}} \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C \Sigma \xi_u$$

and for all Φ_u such that $z_u < 0$, $\mathbf{w} \bullet \Phi_u \geq 1 - \xi_u$

- can be rewritten as

$$\min_{\mathbf{w}} (1/2C)\mathbf{w}^T\mathbf{w} + \Sigma \xi_u$$

and for all Φ_u such that $z_u < 0$, $\xi_u \geq 1 - (\mathbf{w} \bullet \Phi_u)$

- Now, taking $\lambda = 1/2C$, we can reformulate this as

$$\min_{\mathbf{w}} \Sigma [1 - (\mathbf{w} \bullet \Phi_u)]_+ + \lambda \mathbf{w}^T\mathbf{w}$$

- Where $[]_+$ is the positive part (0 if a term is negative)

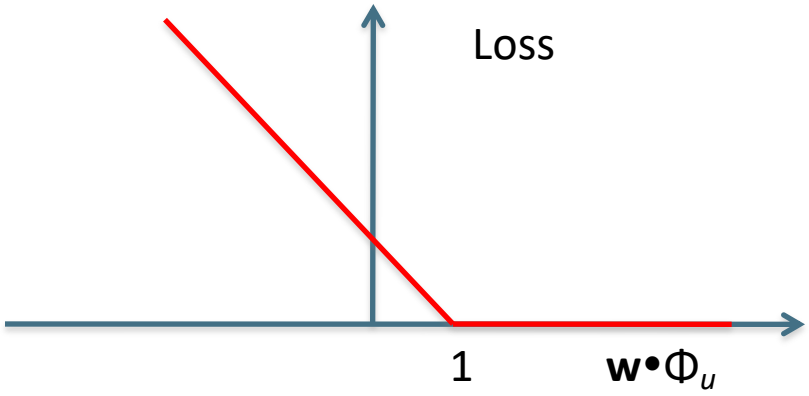
Aside: The SVM loss function

- The reformulation

Hinge loss | Regularizer of $\|\mathbf{w}\|$

$$\min_{\mathbf{w}} \Sigma\, [1 - (\mathbf{w} \bullet \Phi_u)]_+ + \lambda \mathbf{w}^T \mathbf{w}$$

- shows that an SVM can be thought of as having an empirical **"hinge" loss** combined with a **weight regularizer**



Two Problems with Direct Application of the Ranking SVM

- Cost sensitiveness: negative effects of making errors on top ranked documents

  d: *definitely relevant*, p: *partially relevant*, n: *not relevant*
  ranking 1: p d p n n n n
  ranking 2: d p n p n n n

- Query normalization: number of instance pairs varies according to query

  q1: d p p n n n n
  q2: d d p p p n n n n n
  q1 pairs: 2*(d, p) + 4*(d, n) + 8*(p, n) = 14
  q2 pairs: 6*(d, p) + 10*(d, n) + 15*(p, n) = 31

Adapting the Ranking SVM for (successful) Information Retrieval

[Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, Hsiao-Wuen Hon SIGIR 2006]

- A Ranking SVM model already works well
  - Using things like vector space model scores as features
  - As we shall see, it outperforms them in evaluations
- But it does not model important aspects of practical IR well
- This paper addresses two customizations of the Ranking SVM to fit an IR utility model

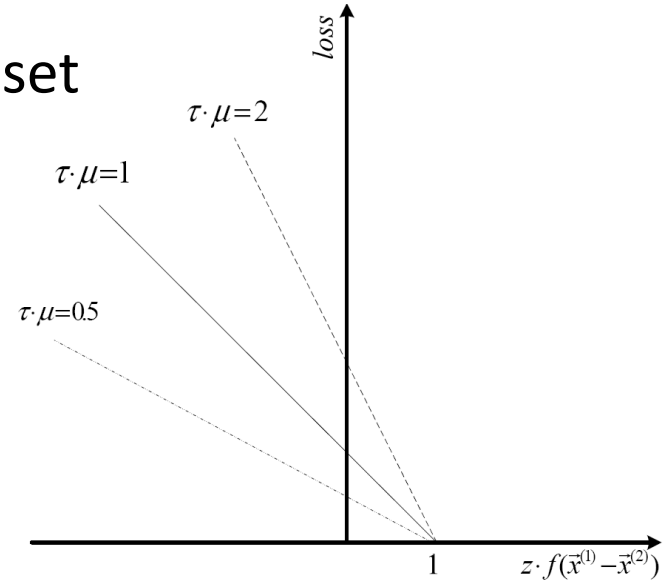
The ranking SVM fails to model the IR problem well …

1. Correctly ordering the most relevant documents is crucial to the success of an IR system, while misordering less relevant results matters little
   - The ranking SVM considers all ordering violations as the same
2. Some queries have many (somewhat) relevant documents, and other queries few. If we treat all pairs of results for a query equally, queries with many results will dominate the learning
   - But actually queries with few relevant results are at least as important to do well on

These problems are solved with a new Loss function

$$\min_{\vec{w}} L(\vec{w}) = \sum_{i=1}^{l} \tau_{k(i)} \mu_{q(i)} \left[1 - z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle \right]_+ + \lambda \left\| \vec{w} \right\|^2$$

- τ weights for type of rank difference
  - Estimated empirically from effect on NDCG
- μ weights for size of ranked result set
  - Linearly scaled versus biggest result set



Alternative: Optimizing Rank-Based Measures

[Yue et al. SIGIR 2007]

- If we think that MAP is a good approximation of the user's utility function from a result ranking
- Then, let's directly optimize this measure
  - As opposed to some proxy (weighted pairwise prefs)
- But, there are problems …
  - Objective function no longer decomposes
    - Pairwise prefs decomposed into each pair
  - Objective function is flat or discontinuous

MAP vs ROC score

- Given two ranked lists p and p-hat:

$$\mathrm{MAP}(p, \hat{p}) = \frac{1}{rel} \sum_{j:p_j=1} Prec@j,$$

$$\mathrm{ROC}(p, \hat{p}) = \frac{1}{rel \cdot (|C| - rel)} \sum_{i:p_i=1} \sum_{j:p_j=0} \mathbf{1}_{[\hat{p}_i > \hat{p}_j]},$$

MAP vs ROC score

- Different:

Doc ID	1	2	3	4	5	6	7	8
p	1	0	0	0	0	1	1	0
$rank(h_1(\mathbf{x}))$	8	7	6	5	4	3	2	1
$rank(h_2(\mathbf{x}))$	1	2	3	4	5	6	7	8

Hypothesis	MAP	ROCArea
$h_1(\mathbf{x})$	0.59	0.47
$h_2(\mathbf{x})$	0.51	0.53

MAP vs Accuracy

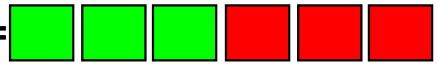
- Example:

Doc ID	1	2	3	4	5	6	7	8	9	10	11
p	1	0	0	0	0	1	1	1	1	0	0
$rank(h_1(\mathbf{x}))$	11	10	9	8	7	6	5	4	3	2	1
$rank(h_2(\mathbf{x}))$	1	2	3	4	5	6	7	8	9	10	11

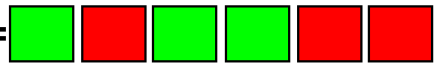
Hypothesis	MAP	Best Acc.
$h_1(q)$	0.56	0.64
$h_2(q)$	0.51	0.73

Structural SVMs [Tsochantaridis et al., 2005]

- Structural SVMs are a generalization of SVMs where the output classification space is not binary or one of a set of classes, but some complex object (such as a sequence or a parse tree)
- Here, it is a complete (weak) ranking of documents for a query
- The Structural SVM attempts to predict the complete ranking for the input query and document set
- The **true labeling** is a ranking where the relevant documents are all ranked in the front, e.g.,

y= 

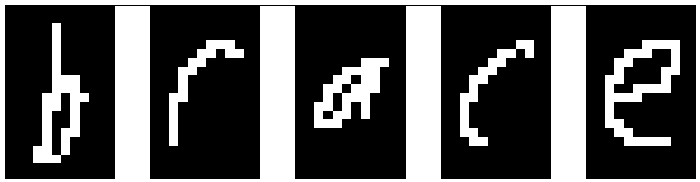
- An **incorrect labeling** would be any other ranking, e.g.,

y′= 

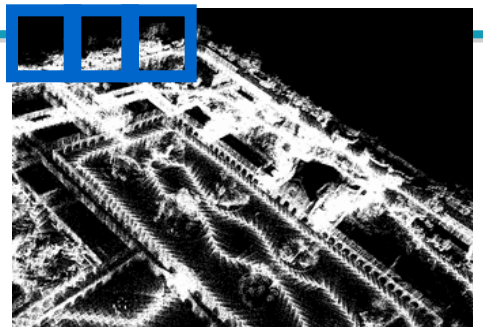
- **There are an intractable number of rankings, thus an intractable number of constraints**!

Structured classification

Local Classification

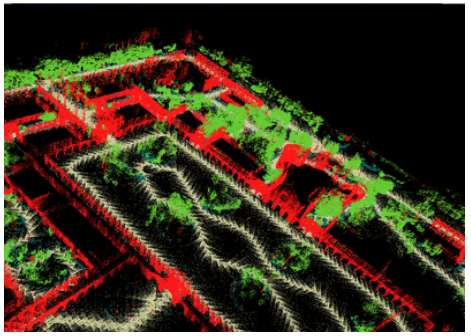


x



b r a r e

y

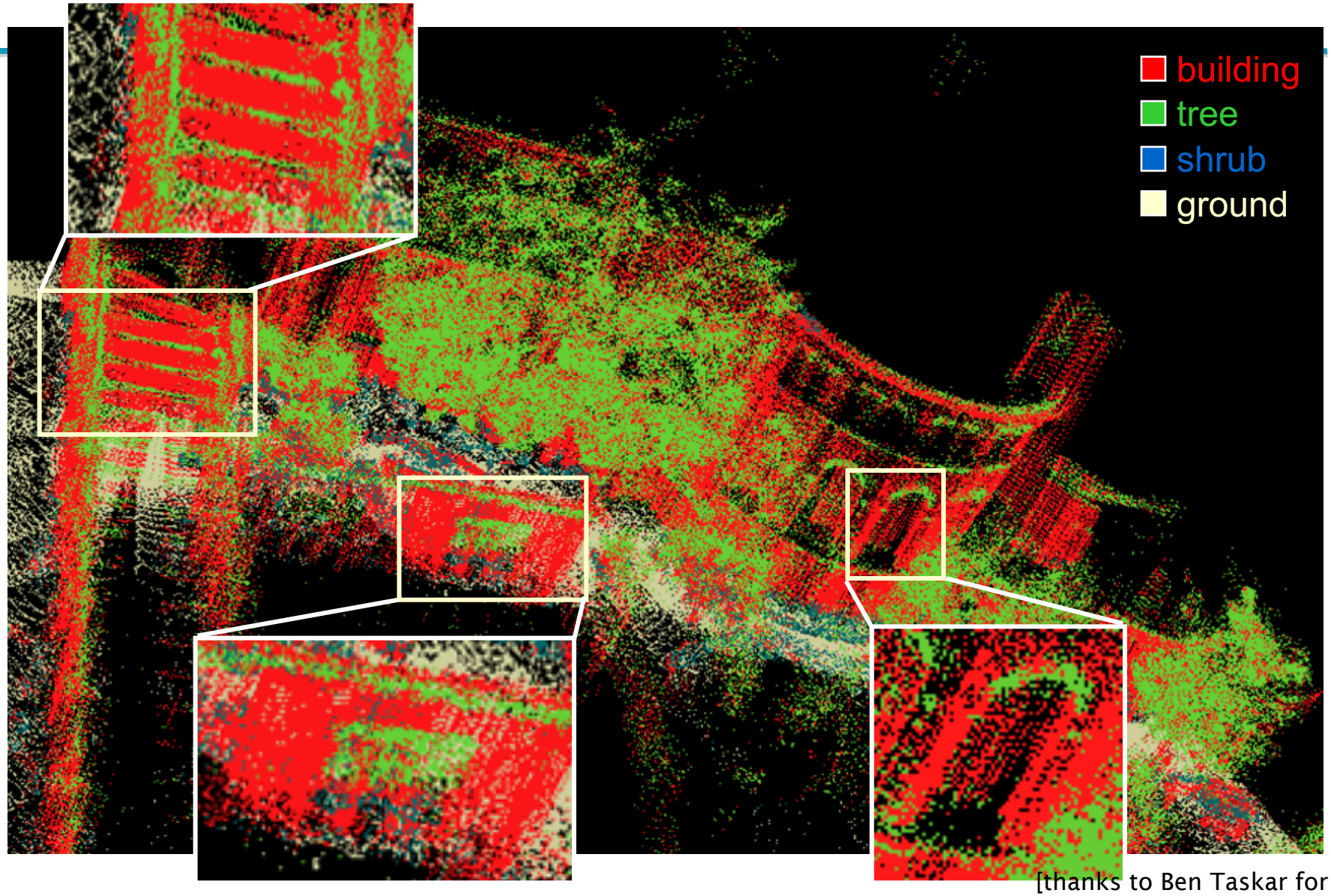


Classify using local information

⇒ Ignores correlations!

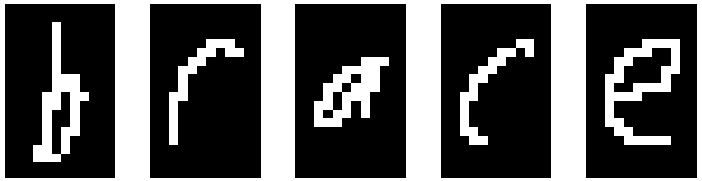
[thanks to Ben Taskar for slide

Local Classification

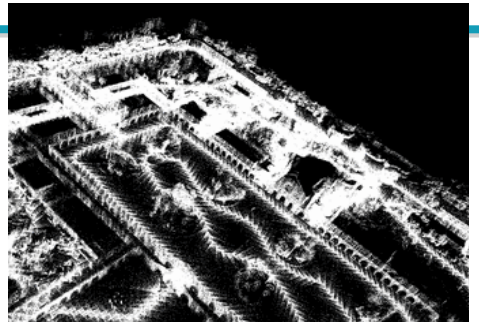


[thanks to Ben Taskar for sli

Structured Classification

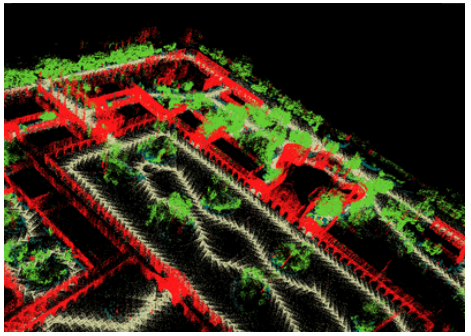


x



b r a c e

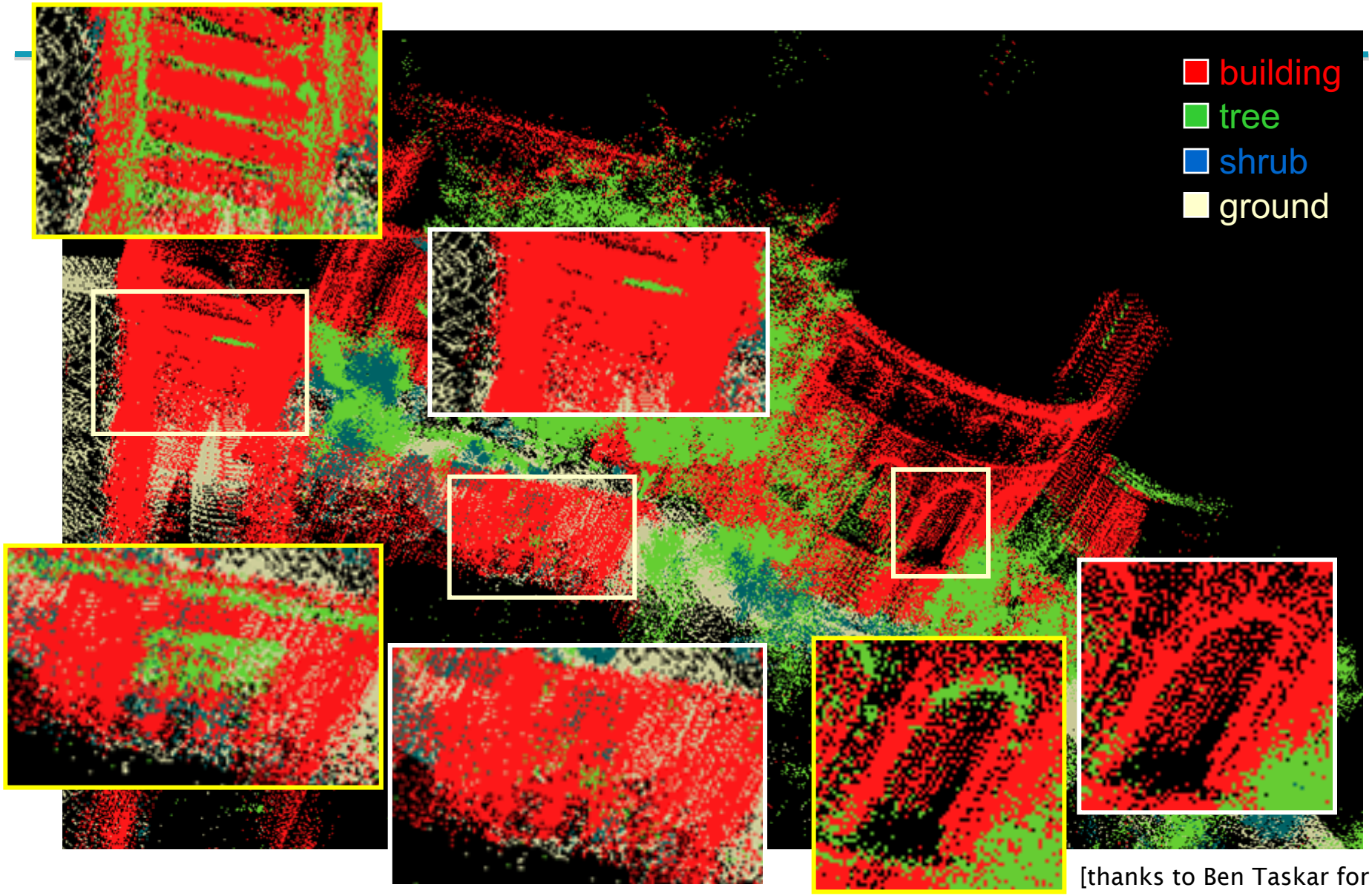
y



- Use local information
- Exploit correlations

[thanks to Ben Taskar for slide!]

Structured Classification



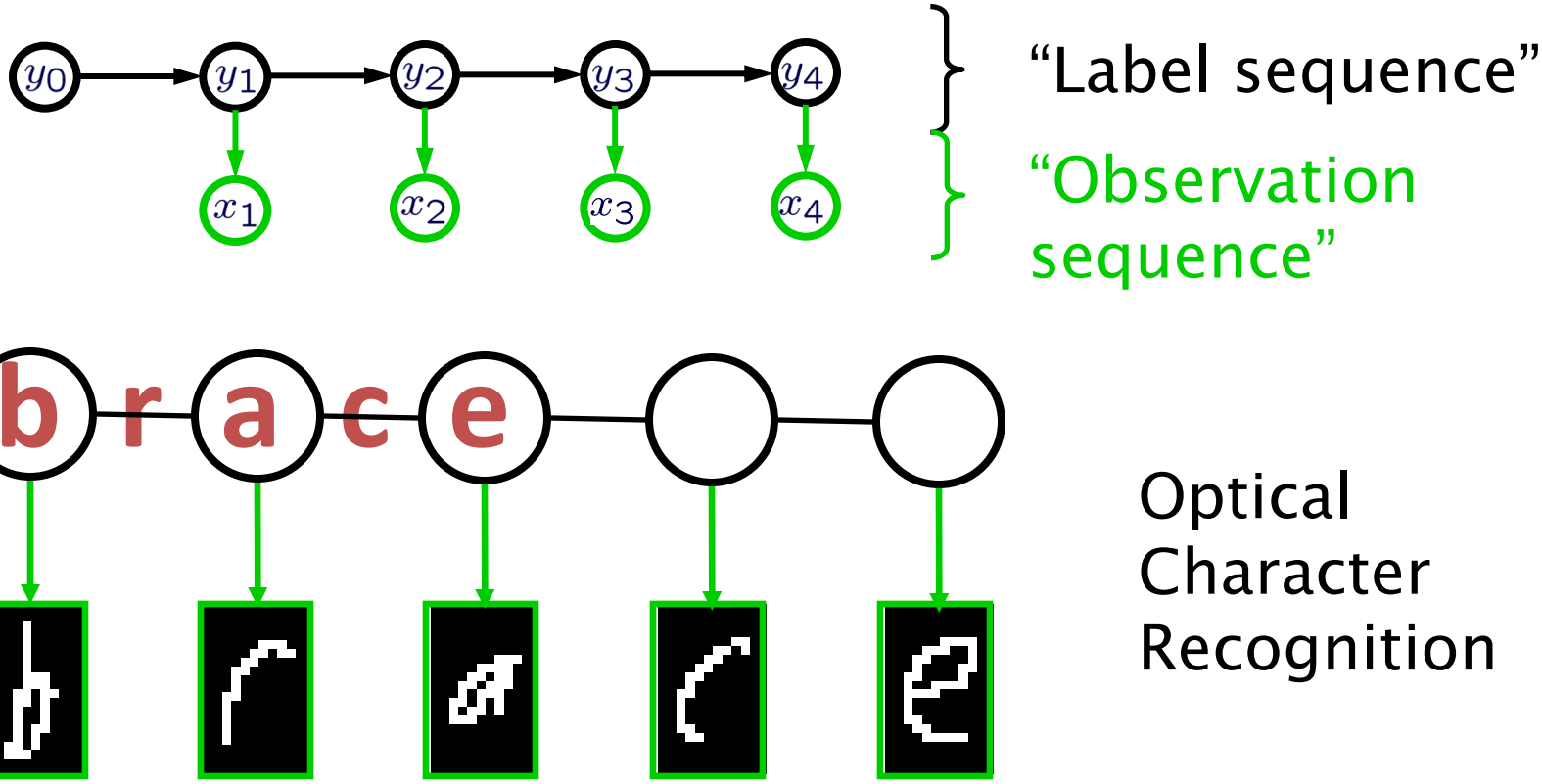
[thanks to Ben Taskar for sli

Structured Classification

- Structured classification : direct approaches
  - Generative approach: Markov Random Fields (Bayesian modeling with graphical models)
  - Linear classification:
    - Structured Perceptron
    - Conditional Random Fields (counterpart of logistic regression)
    - Large-margin structured classification

Structured classification

Simple example HMM:



Structured Model

- Main idea: define scoring function which **decomposes** as sum of features scores k on "**parts**" p:

$$score(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}) = \sum_{k,p} w_k^\top \phi_k(\mathbf{x}_p, \mathbf{y}_p)$$

- Label examples by looking for max score:

$$prediction(\mathbf{x}, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} score(\mathbf{x}, \mathbf{y}, \mathbf{w})$$

space of feasible outputs

- Parts = nodes, edges, etc.

Decoding and Learning

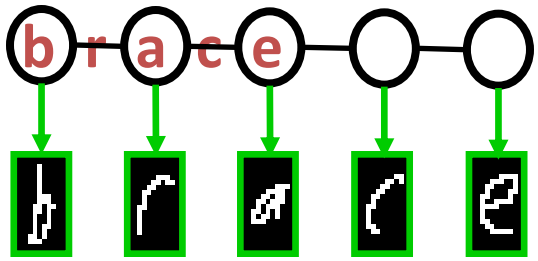
Three important operations on a general structured (e.g. graphical) model:

$$\underset{y_1,\ldots,y_n}{\text{argmax}}\, p(y_1, \ldots, y_n|x)$$

$$\forall j, \quad p(y_j|x)$$

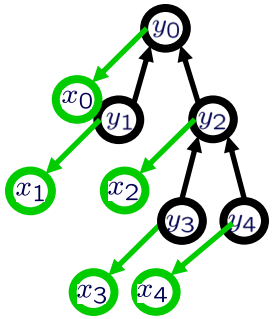
- **Decoding:** find the right label sequence
- **Inference:** compute probabilities of labels
- **Learning:** find model + parameters *w* so that decoding works

HMM example:



- **Decoding:** Viterbi algorithm
- **Inference:** forward-backward algorithm
- **Learning:** e.g. transition and emission counts (case of learning a generative model from fully labeled training data)

Decoding and Learning



- **Decoding:** algorithm on the graph (eg. max-product)
- **Inference:** algorithm on the graph (eg. sum-product, belief propagation, junction tree, sampling)
- **Learning:** inference + optimization

Use **dynamic programming** to take advantage of the structure

1. Focus of graphical model class
2. Need 2 essential concepts:
    1. cliques: variables that directly depend on one another
    2. features (of the cliques): some functions of the cliques

Our favorite (discriminative) algorithms

Perceptron: $\max_{\mathbf{w}} \sum_i \left[\mathbf{w}^\top \Phi(x^i, y^i) - \max_y \mathbf{w}^\top \Phi(x^i, y) \right]$

[mistake driven]

CRF: $\max_{\mathbf{w}} \sum_i \left[\mathbf{w}^\top \Phi(x^i, y^i) - \text{softmax}_y \mathbf{w}^\top \Phi(x^i, y) \right]$

(Conditional Random Field)

[max conditional likelihood]

M^{3}net: $\max_{\mathbf{w}} \sum_i \left[\mathbf{w}^\top \Phi(x^i, y^i) - \max_y (\ell(y, y^i) + \mathbf{w}^\top \Phi(x^i, y)) \right]$

[max. margin]

The devil is the details.....

(Averaged) Perceptron

For each datapoint $\mathbf{x}^i$

Predict: $\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \Phi(\mathbf{x}^i, \mathbf{y})$

Update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \underbrace{\left(\Phi(\mathbf{x}, \mathbf{y}^i) - \Phi(\mathbf{x}^i, \hat{\mathbf{y}}_i)\right)}_{\text{update if } \hat{\mathbf{y}}_i \neq \mathbf{y}^i}$

Averaged perceptron: $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$

Example: multiclass setting

Predict: $\hat{y}_i = \arg\max_y \; w_y^\top x^i$

Update: if $\hat{y}_i \neq y^i$ then

$$w_{y^i,t+1} = w_{y^i,t} + \alpha x^i$$
$$w_{\hat{y}_i,t+1} = w_{\hat{y}_i,t} - \alpha x^i$$

Feature encoding:

$$\begin{aligned}
\Phi(\mathbf{x}^i, y=1)^\top &= [\mathbf{x}^{i\top}\ 0\ \ldots\ 0] \\
\Phi(\mathbf{x}^i, y=2)^\top &= [0\ \mathbf{x}^{i\top}\ \ldots\ 0] \\
&\vdots \\
\Phi(\mathbf{x}^i, y=K)^\top &= [0\ 0\ \ldots\ \mathbf{x}^{i\top}] \\
\mathbf{w}^\top &= [w_1^\top\ w_2^\top\ \ldots\ w_K^\top]
\end{aligned}$$

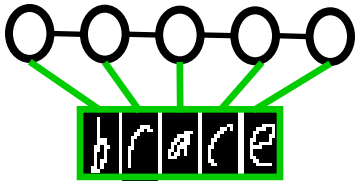
Predict: $\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; \mathbf{w}_t^\top \Phi(\mathbf{x}^i, \mathbf{y})$

Update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \underbrace{\left(\Phi(\mathbf{x}, \mathbf{y}^i) - \Phi(\mathbf{x}^i, \hat{\mathbf{y}}_i)\right)}_{\text{update if } \hat{\mathbf{y}}_i \neq \mathbf{y}^i}$

CRF

Z difficult to compute with complicated graphs

$$\frac{\exp \mathbf{w}^\top \Phi(y^i|x^i)}{\sum_y \exp \mathbf{w}^\top \Phi(y|x^i)}$$



Conditioned on all the observations

Introduction by Hannah M.Wallach

http://www.inference.phy.cam.ac.uk/hmw26/crf/

MEMM & CRF, Mayssam Sayyadian, Rob McCann

anhai.cs.uiuc.edu/courses/498ad-fall04/local/my-slides/crf-students.pdf

M^3net

No Z …

The margin penalty $\ell(y, y^i)$ can "factorize" according to the problem structure

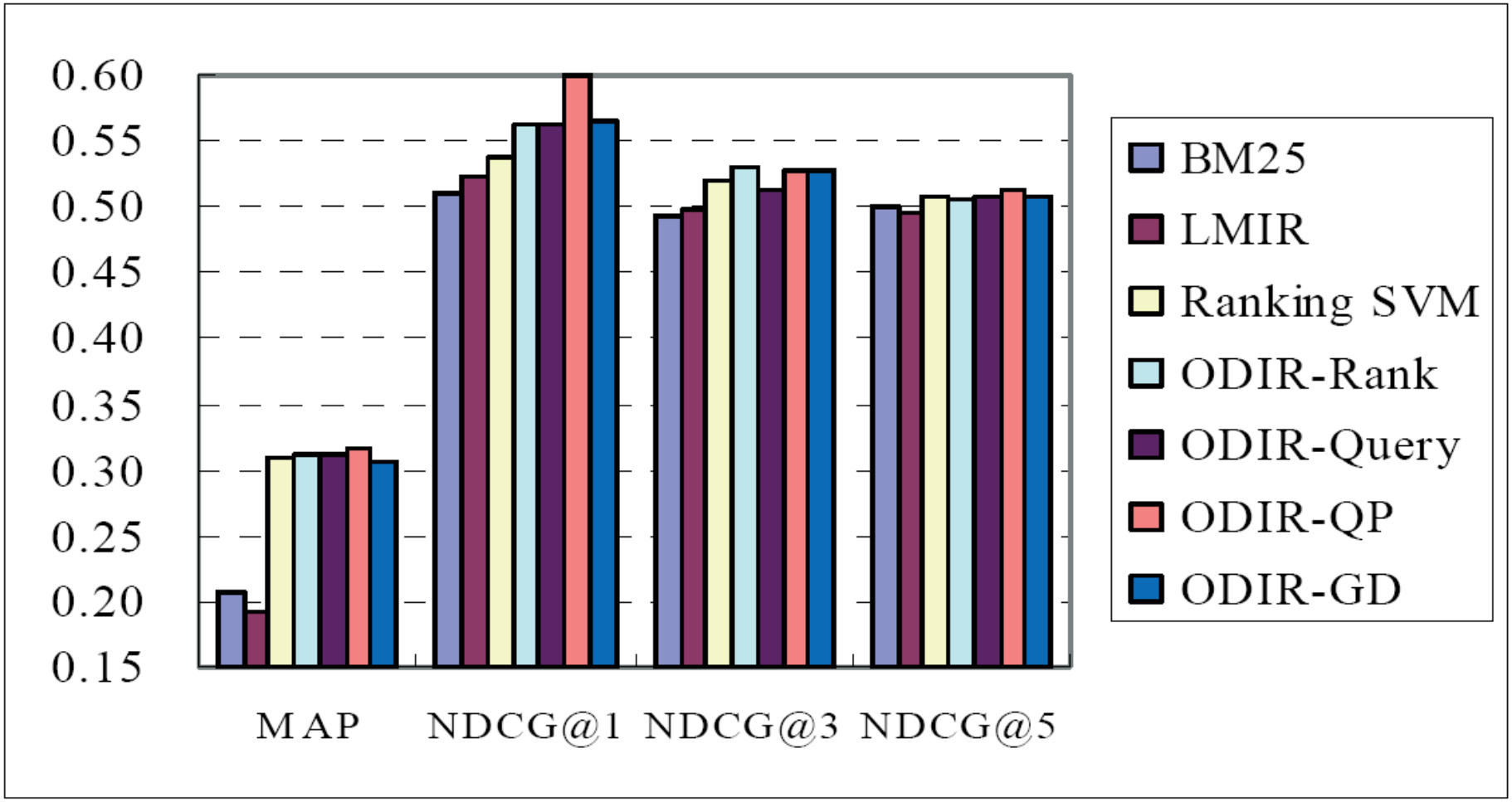
Introduction by Simon Lacoste-Julien

http://www.cs.berkeley.edu/~slacoste/school/cs281a/project_report.html

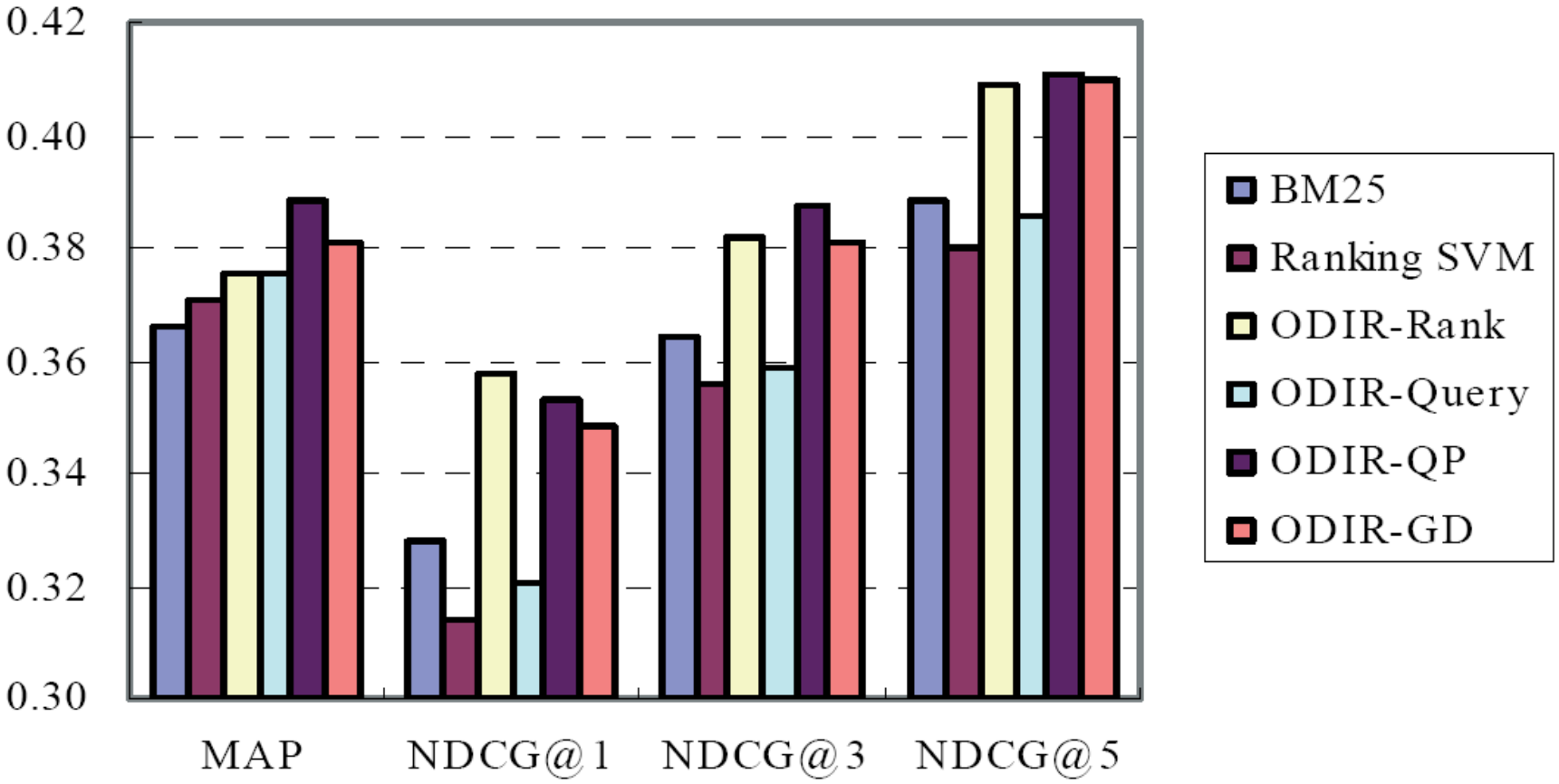
Experiments

- OHSUMED (from LETOR)
- Features:
  - 6 that represent versions of tf, idf, and tf.idf factors
  - BM25 score (*IIR* sec. 11.4.3)
    - A scoring function derived from a probabilistic approach to IR, which has traditionally done well in TREC evaluations, etc.

Experimental Results (OHSUMED)



Experimental Results (MSN search)



Other machine learning methods for learning to rank

- Of course!
- I've only presented the use of SVMs for machine learned relevance, but other machine learning methods have also been used successfully
  - Boosting: RankBoost
  - Ordinal Regression loglinear models
  - Neural Nets: RankNet
  - (Gradient-boosted) Decisision Trees

The Limitations of Machine Learning

- Everything that we have looked at (and most work in this area) produces *linear* models of features by weighting different base features
- This contrasts with most of the clever ideas of traditional IR, which are *nonlinear* scalings and combinations of basic measurements
  - log term frequency, idf, pivoted length normalization
- At present, ML is good at weighting features, but not at coming up with nonlinear scalings
  - Designing the basic features that give good signals for ranking remains the domain of human creativity

Resources

- *IIR* secs 6.1.2–3 and 15.4
- LETOR benchmark datasets
  - Website with data, links to papers, benchmarks, etc.
  - http://research.microsoft.com/users/LETOR/
  - Everything you need to start research in this area!
- Nallapati, R. Discriminative models for information retrieval. *SIGIR 2004.*
- Cao, Y., Xu, J. Liu, T.-Y., Li, H., Huang, Y. and Hon, H.-W. Adapting Ranking SVM to Document Retrieval, *SIGIR 2006.*
- Y. Yue, T. Finley, F. Radlinski, T. Joachims. A Support Vector Method for Optimizing Average Precision. *SIGIR 2007*.