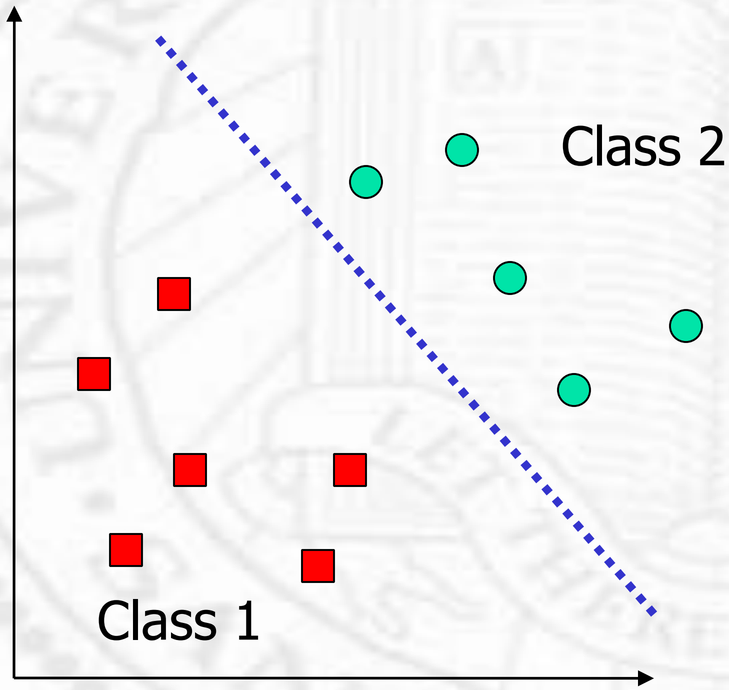
Support Vector Machines

More Generally Kernel Methods

2 Important Concepts

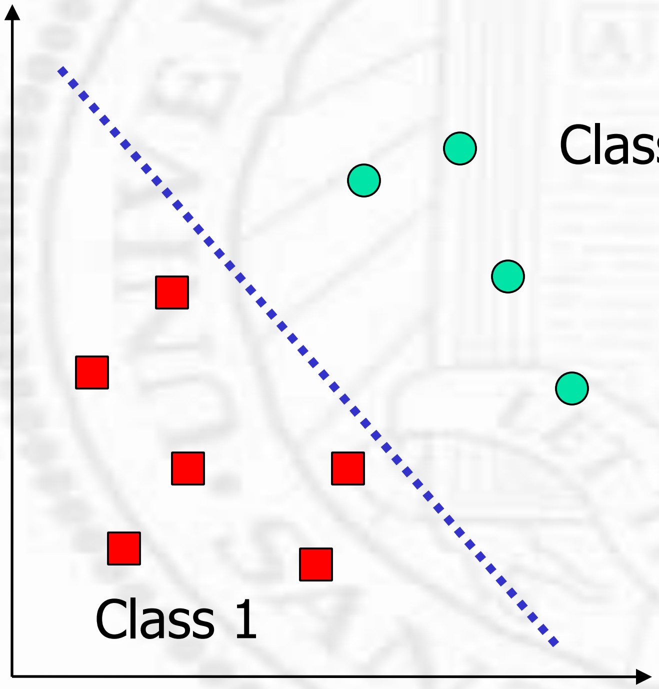
- Seek linear decision boundary with two new concepts
  - Optimization based on maximizing margins (not GD)
  - Kernel mapping for better linear separation ("massaging data")

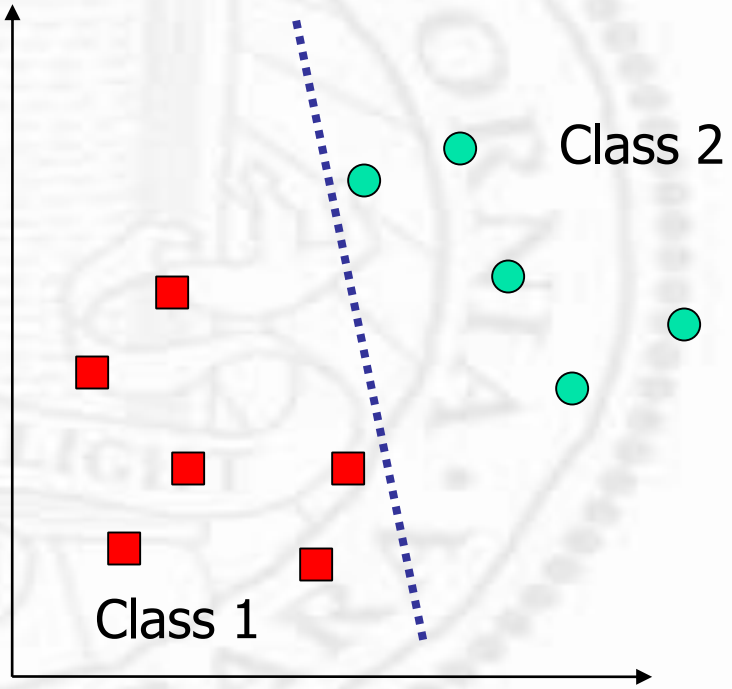
Two Class Problem: Linear Separable Case



- Many decision boundaries can separate these two classes
- Which one should we choose?

Example of Bad Decision Boundaries

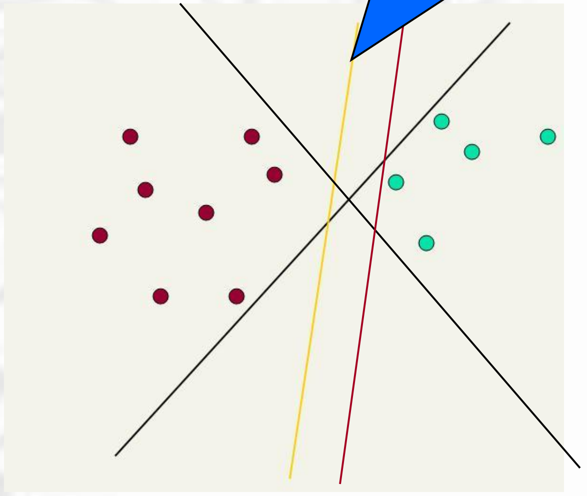




Linear classifiers: Which Hyperplane?

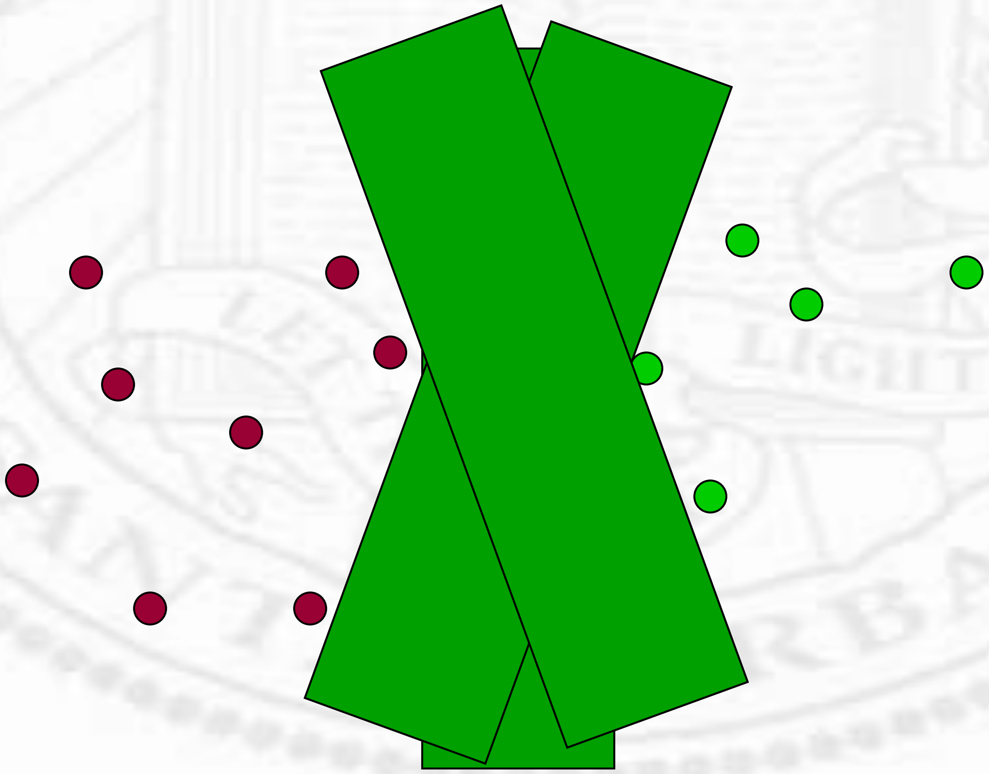
- Lots of possible solutions for *a,b,c*.
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron, GD
- Support Vector Machine (SVM) finds an optimal solution.
  - Maximizes the distance between the hyperplane and the "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

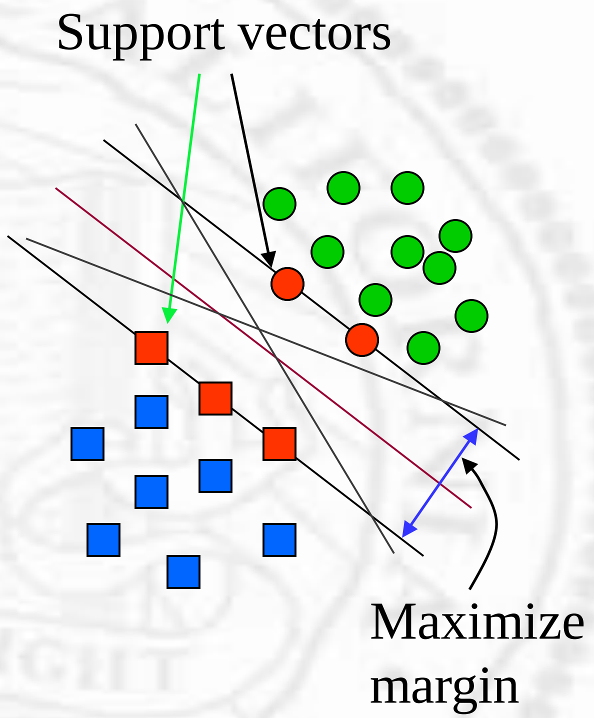
- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



6

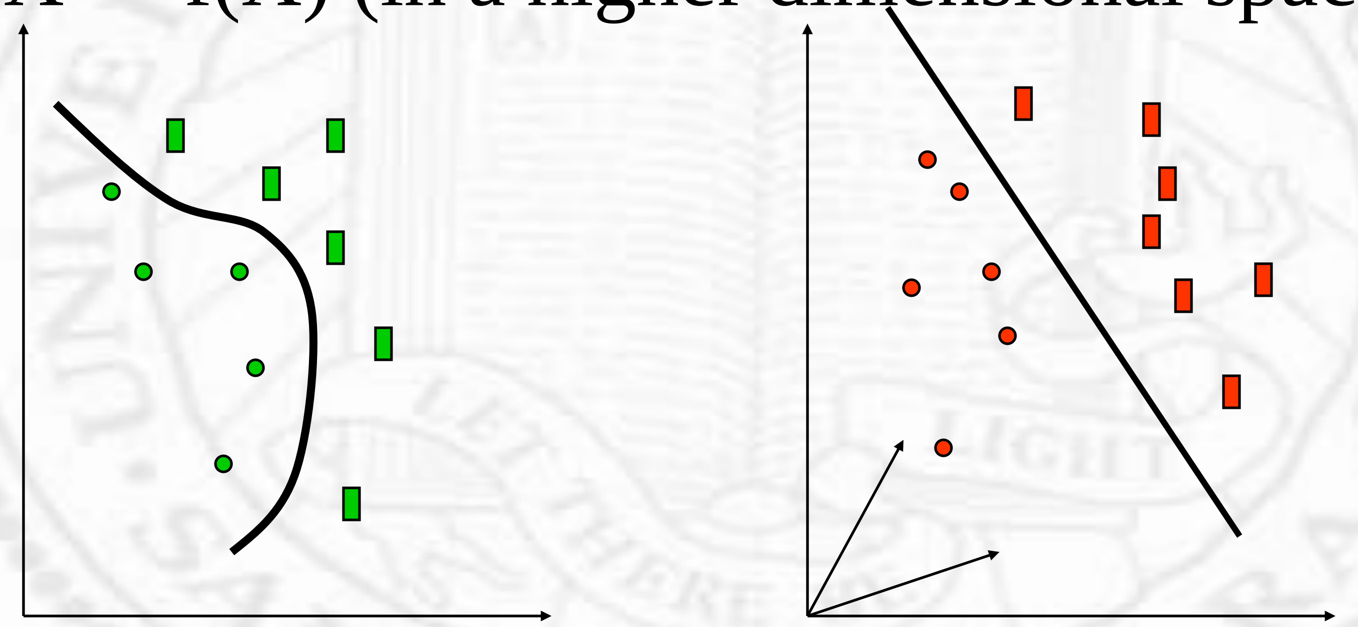
Support Vector Machine (SVM)

- 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*.
- *Quadratic programming* problem
- Seen by many as most successful current text classification method



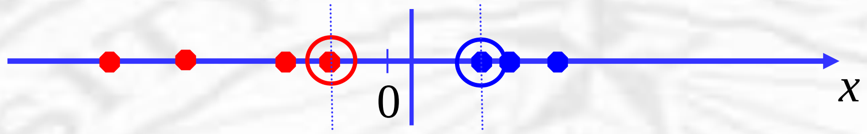
SVM with Kernel Mapping

- Original feature space might not be well conditioned
- X -> f(X) (in a higher dimensional space)

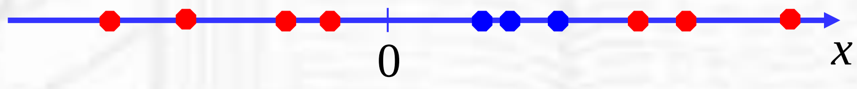


Non-linear SVMs

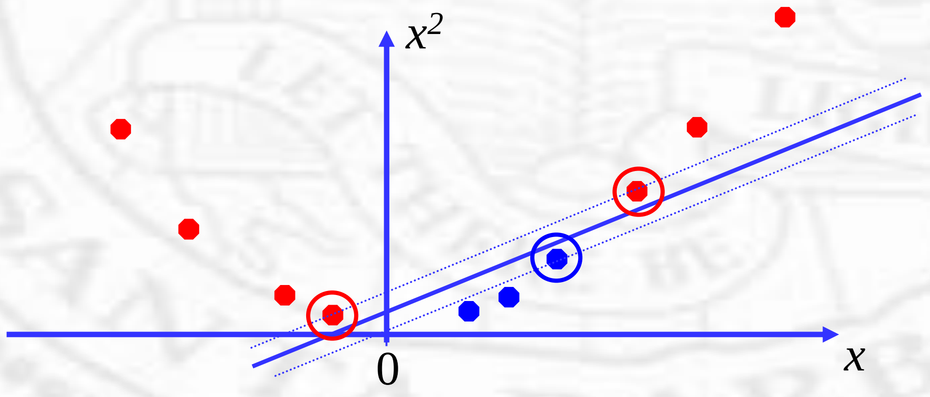
- Datasets that are linearly separable (with some noise) work out great:



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

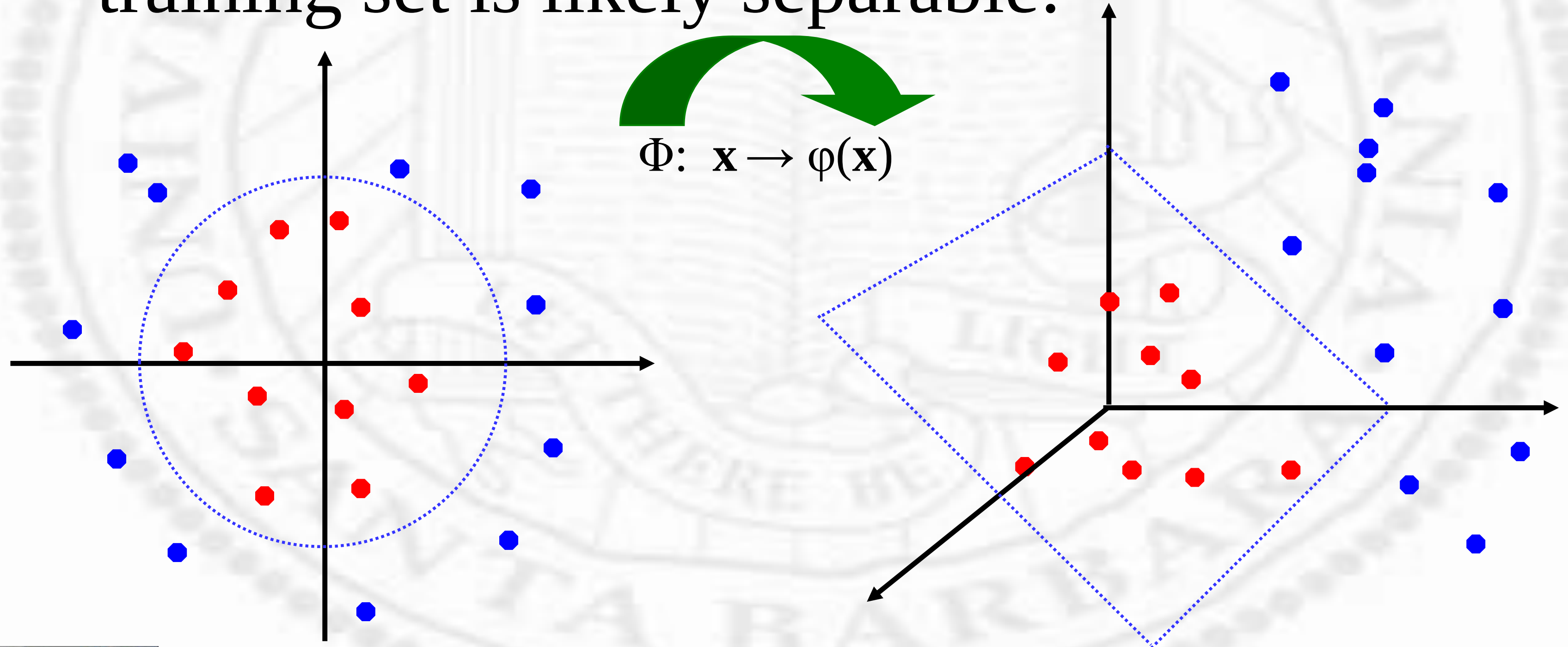


- How about … mapping data to a higher-dimensional space:



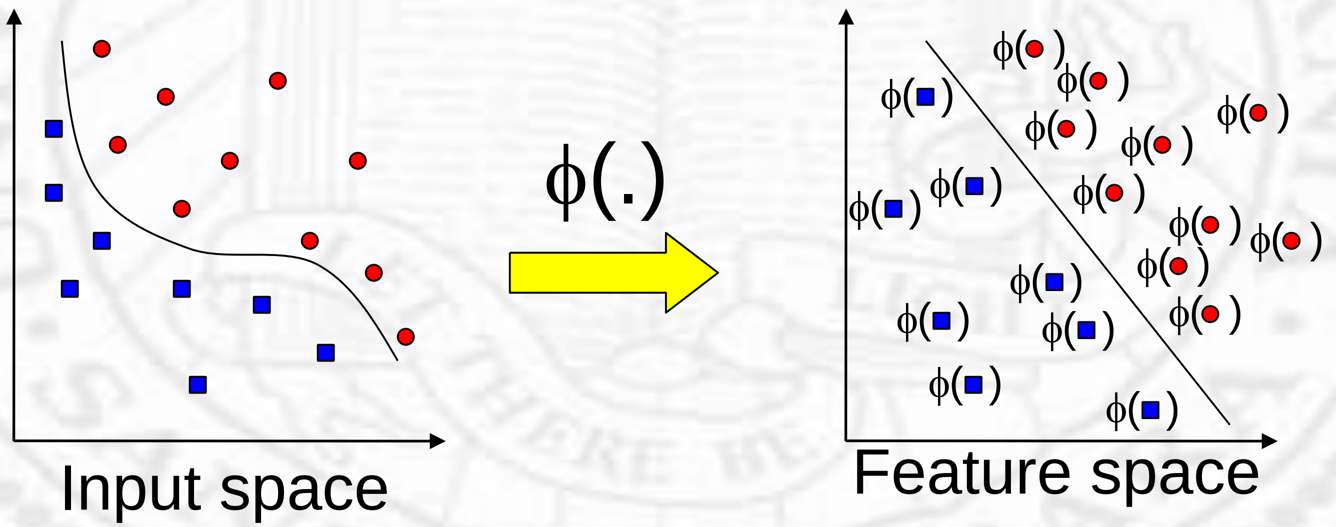
Non-linear SVMs: Feature spaces

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



Transformation to Feature Space

- High computation burden due to high-dimensionality and hard to get a good estimate
- "Kernel tricks" for efficient computation: Only the inner products of feature vectors are used, mapping is not explicitly computed (efficiency in time and space)



Kernel-Induced Feature Spaces

- Recall from Perceptron learning rule, weight is a combination of feature vectors (those that are difficult to handle, or classified wrongly)

$$\mathbf{w}^{(k+1)} = \begin{cases} \mathbf{w}^{(k)} + cy\mathbf{x} & y(\mathbf{w}^{(k)} \cdot \mathbf{x}) < 0, \mathbf{x} \in +or- \\ \mathbf{w}^{(k)} & otherwise \end{cases}$$

Kernel-Induced Feature Spaces

- Or, the decision rule involves only inner product of features, not features themselves

$$h(\mathbf{x}) = \sum_i \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x}) \qquad w = \sum_i \alpha_i y_i \mathbf{x}_i$$

- The same holds true for mapped space
  - Dimension is not important (may not know the mapping)
  - Inner products can be evaluated at the low-dimensional space

$$h(\mathbf{x}) = \sum_i \alpha_i y_i (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})) \qquad w = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)$$

Kernels

- A function that returns the value of dot product of mapping of two arguments

$$k(\mathbf{x}_1,\mathbf{x}_2)=\phi(\mathbf{x}_1)\cdot\phi(\mathbf{x}_2)=\left\langle\phi(\mathbf{x}_1),\phi(\mathbf{x}_2)\right\rangle$$

  - The same perceptron style learning can be applied (just replace all dot products with kernels; however, SVM has more sophisticated learning rules)

Example

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, 2x_1x_2, x_2^2) \in F = R^3$$

$$k(\mathbf{x}, \mathbf{y}) = (x_1y_1 + x_2y_2)^2$$

$$= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$= \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

More Example

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_2, x_2^2) \in F = R^4$$

$$k(\mathbf{x}, \mathbf{y}) = (x_1y_1 + x_2y_2)^2$$

$$= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2$$

$$= (x_1^2, x_1x_2, x_1x_2, x_2^2) \cdot (y_1^2, y_1y_2, y_1y_2, y_2^2)$$

$$= \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

Even More Example

$$\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = \frac{1}{\sqrt{2}}(x_1^2 - x_2^2, 2x_1x_2, x_1^2 + x_2^2) \in F = R^3$$

$$k(\mathbf{x},\mathbf{y}) = (x_1y_1 + x_2y_2)^2$$

$$= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2)\cdot(y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$= \frac{1}{\sqrt{2}}(x_1^2 - x_2^2, 2x_1x_2, x_1^2 + x_2^2)\cdot\frac{1}{\sqrt{2}}(y_1^2 - y_2^2, 2y_1y_2, y_1^2 + y_2^2)$$

$$= \phi(\mathbf{x})\cdot\phi(\mathbf{y})$$

- The interpretation of mapping ϕ is not unique even with a single κ function

Gaussian Kernel

- Identical to the Radial Basis Function

$$\kappa(\mathbf{x},\mathbf{z}) = <\phi(\mathbf{x}),\phi(\mathbf{z})> = \exp(\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2})$$

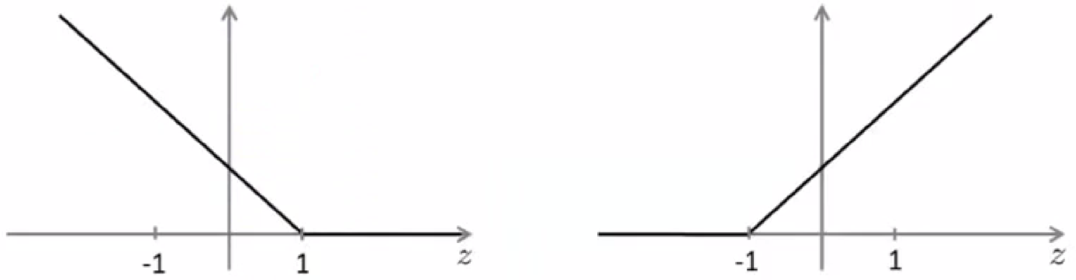
- Recall that $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$

- The feature dimension is infinitely high in this case
- Embedding is not explicitly computed (impossible), only inner product is needed in SVM

SVM Cost Function

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

- Similar to logistic function, but piecewise linear, no penalty for z>1 y=1, and z<-1, y=0

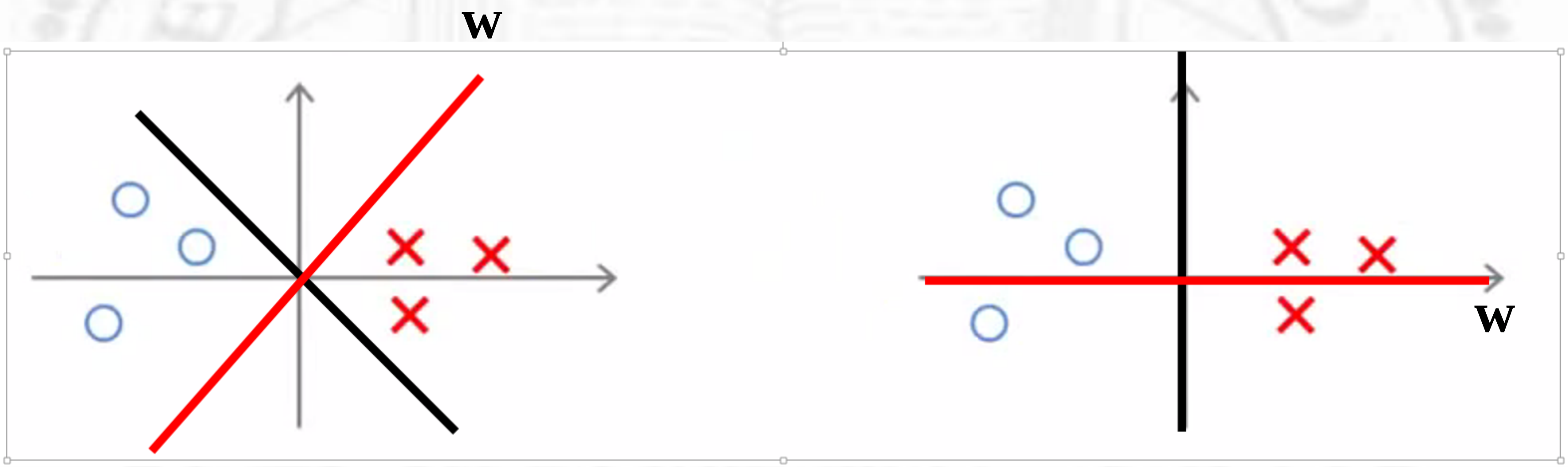


If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

Margin

- for $w^T x$ to be >1 (for y=1) and <-1 (for y=-1), w must be larger in the left figure than the right. So large margin imply small $|w|^2$



Numerical Methods

- Maximize margin or minimize $|w|^2$ cannot be done by gradient descent but by quadratic programming (DON'T try it yourself, find a readily available implementation SVM-light, SVMlib)
- $h(x) = \sum_i \alpha_i y_i (x_i \cdot x) + b \qquad h(x) = \sum_i \alpha_i y_i K(x_i \cdot x) + b$
- i is the number of support vectors, x_i is the i-th support vector, a_i is the weight, and b is a bias

The Optimization Problem

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$ hard magin
- A constrained optimization problem

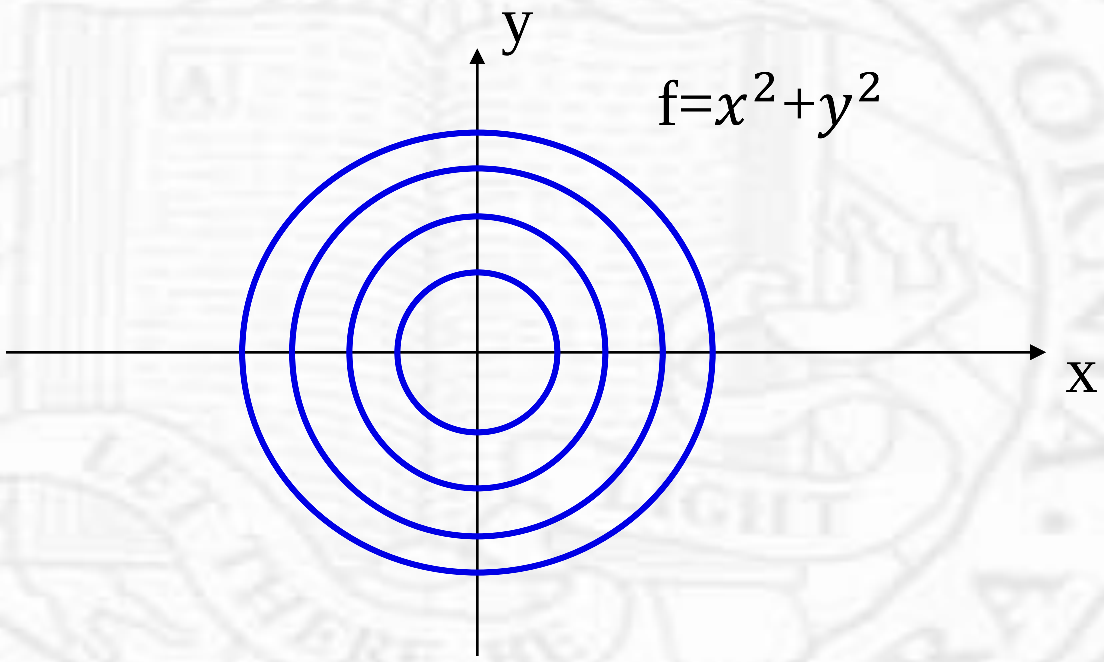
$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

- $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

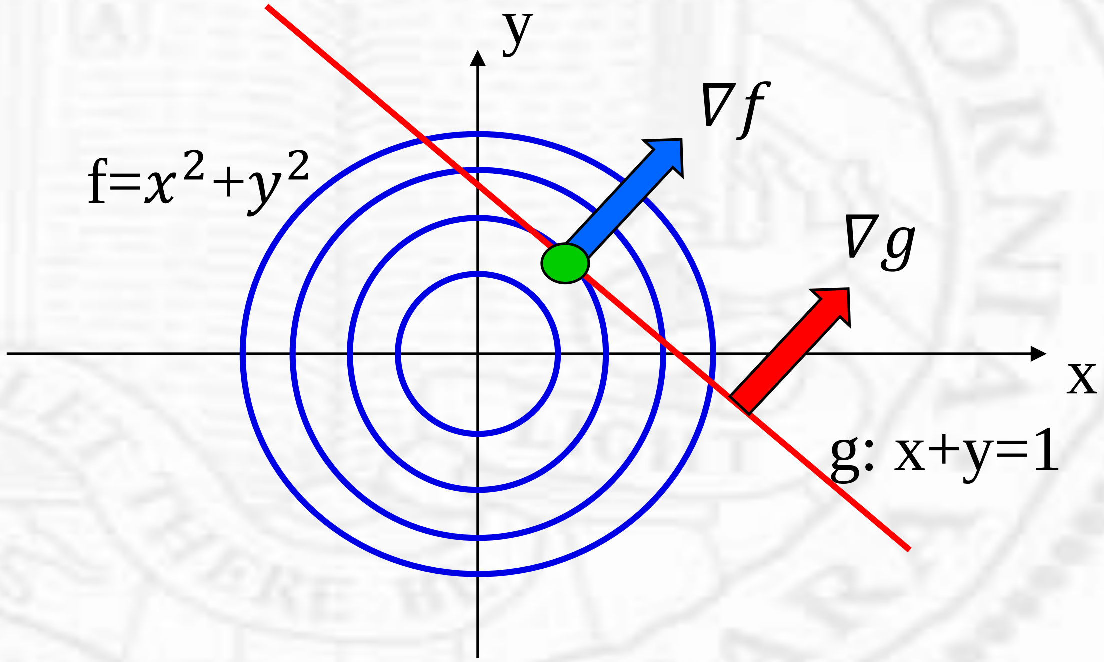
*Optimization f(**X**), no constraints*

- $\nabla f = 0$



*Optimization f(**X**), equality constraints*

- g(**X**) =0:
  - Min f +λg or $\nabla f + \lambda \nabla g = 0$



*Optimization f(**X**), inequality constraints*

- h(**X**)<=0: Min f +λh or $\nabla f + \lambda \nabla h = 0$
  - Necessary condition for optimality: ∇f=0
    - Outside feasible region (h(**X**)>0), not a solution
    - Inside feasible region, a solution (h(**X**)<0), h does not matter, $\lambda = 0$
    - On the boundary of feasible region h(**X**)=0, h behaves just like g (an equality constraint)
    - For minimization problem
      - ∇f must point inside
      - ∇h must point outside
      - $\nabla f = -\lambda \nabla h, \lambda > 0$

KKT condition

$$h(x) \leq 0$$
$$\lambda > 0$$
$$\nabla f + \lambda \nabla h = 0$$

Lagrangian of Original Problem

Minimize $\frac{1}{2}||\mathbf{w}||^2$ subject to $1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \le 0$ for $i = 1,$

- The Lagrangian is

Lagrangian multipliers

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

$\alpha_i \ge 0$

  - Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

- Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero,

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i} \rightarrow \frac{\partial L}{\partial w} = 0$$

$$\boxed{\sum_{i=1}^{n} \alpha_i y_i = 0} \longrightarrow \frac{\partial L}{\partial b} = 0$$

$$L = \frac{1}{2} w^T w + \sum \alpha_i (1 - y_i (w^T x_i + b))$$

$$= \frac{1}{2} w^T w + \sum \alpha_i \;- w^T \sum \alpha_i y_i x_i + b \sum \alpha_i y_i$$

$$= \frac{1}{2} w^T w + \sum \alpha_i \;- w^T w$$

$$= -\frac{1}{2} w^T w + \sum \alpha_i$$

$$= -\frac{1}{2} (\alpha_i y_i x_i)^T (\alpha_j y_j x_j) \;\; + \sum \alpha_i$$

$$= \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i \, y_j x_i \cdot x_j$$

The Dual Optimization Problem

- We can transform the problem to its dual

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Dot product of X

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

α's → New variables (Lagrangian multipliers)

- This is a convex quadratic programming (QP) problem
  - Global maximum of α_i can always be found

→well established tools for solving this optimization problem (e.g. cplex)

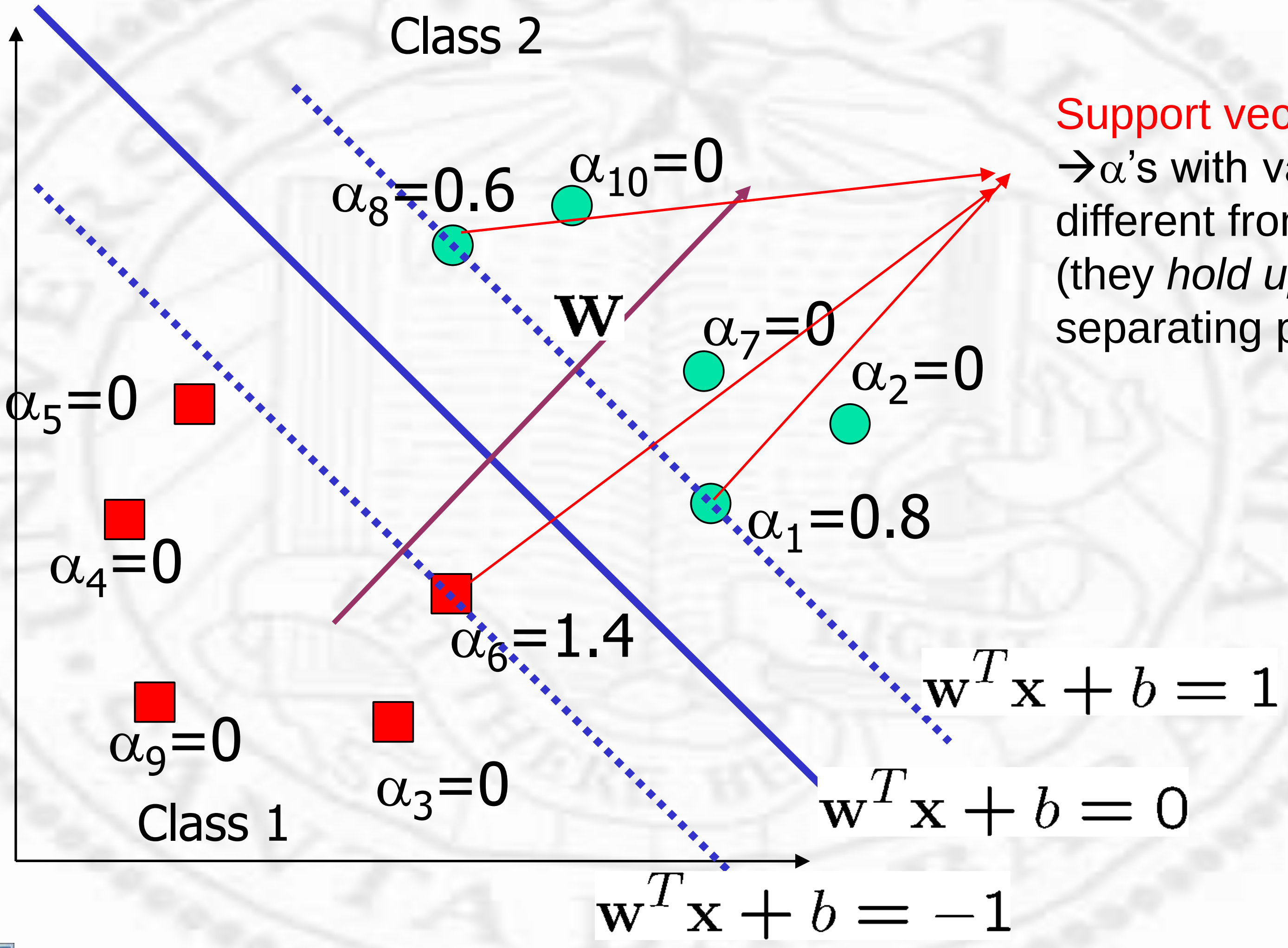
$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Quadratic Programming

- The cost is quadratic
  - Without constraint, there is a unique minimum
  - Any descent search technique (e.g., gradient descent) should *eventually* find the minimum
- Added twist
  - The search is with constraint

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

A Geometrical Interpretation



Support vectors
→α's with values different from zero (they *hold up* the separating plane)!

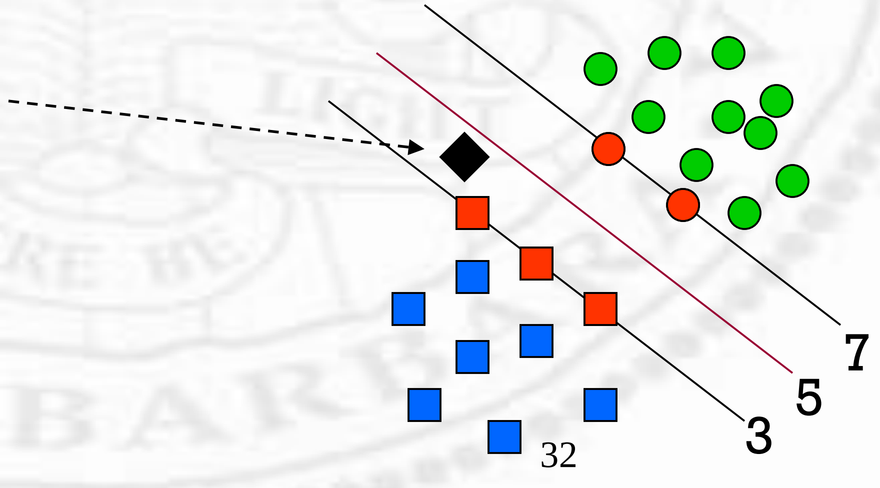
Classification with SVMs

- Given a new point (x_1,x_2), we can score its projection onto the hyperplane normal:
  - In 2 dims: score = $w_1x_1+w_2x_2+b$.
  - I.e., compute score: $wx + b = \Sigma\alpha_i y_i \mathbf{x_i}^{\mathbf{T}}\mathbf{x} + b$
  - Set confidence threshold t.

Score > t: yes

Score < -t: no

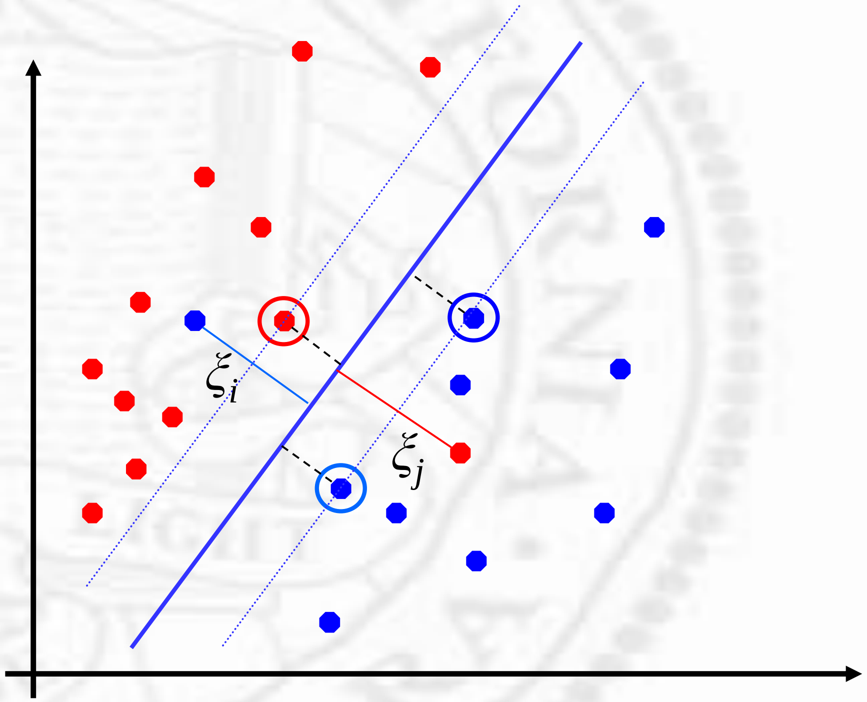
Else: don't know



32

Soft Margin Classification

- If the training set 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)



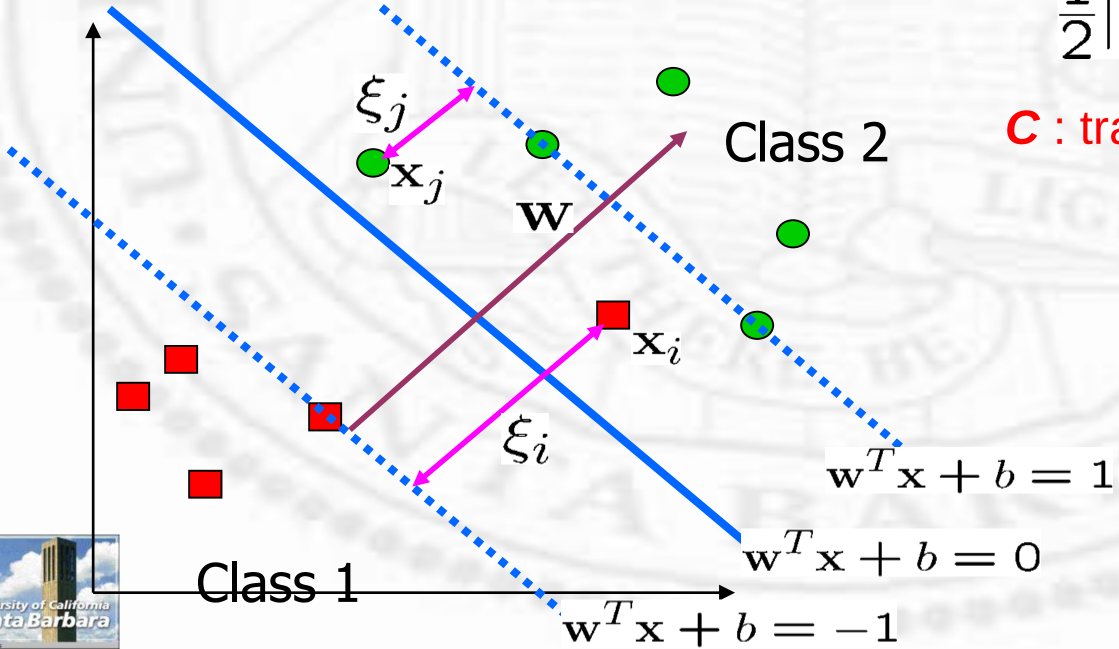
Soft margin

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}$+b
- ξ_i approximates the number of misclassified samples

New objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors



ξ_j
$\mathbf{x}_j$
$\mathbf{w}$
Class 2
$\mathbf{x}_i$
ξ_i
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
Class 1
$\mathbf{w}^T\mathbf{x} + b = -1$

Linear Non-Separable Case

- When classes are not separable
  - Introduce slack variables and relax constraints

$$y_i(\mathbf{w}\cdot\mathbf{x}_i) \ge 1-\xi_i \qquad\qquad \xi_i \ge 0$$

  - Minimize the objective functions
    - Allow some samples into the buffer zone, but minimize the number and the amount of protrusion

$$L = \frac{1}{2}|\mathbf{w}|^2 + C\sum_i \xi_i \qquad\qquad C :> 0\ (weight)$$

Wolfe Dual

$$L = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_i \xi_i \qquad 0 \le \xi_i \le 1, C > 0$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C$$

General Case

$$L = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_i \xi_i \qquad 0 \le \xi_i \le 1, C > 0$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C$$

Example

$$k(\mathbf{x}_1,\mathbf{x}_2) = (1+\mathbf{x}_1\cdot\mathbf{x}_2)^2 = (1+\mathbf{x}_{11}\mathbf{x}_{21}+\mathbf{x}_{12}\mathbf{x}_{22})^2$$

$$= (1+\mathbf{x}_{11}^2\mathbf{x}_{21}^2 + 2\mathbf{x}_{11}\mathbf{x}_{12}\mathbf{x}_{21}\mathbf{x}_{22} + \mathbf{x}_{12}^2\mathbf{x}_{22}^2 + 2\mathbf{x}_{11}\mathbf{x}_{21} + 2\mathbf{x}_{12}\mathbf{x}_{22})$$

$$\varphi(\mathbf{x}_i) = [1, \mathbf{x}_{i1}^2, \sqrt{2}\mathbf{x}_{i1}\mathbf{x}_{i2}, \mathbf{x}_{i2}^2, \sqrt{2}\mathbf{x}_{i1}, \sqrt{2}\mathbf{x}_{i2}]^T$$

– *Xor*

$\mathbf{X}$	y
$(-1,-1)$	-1
$(-1,+1)$	$+1$
$(+1,-1)$	$+1$
$(+1,+1)$	-1

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

$$L = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$
$$-\frac{1}{2}(9\alpha_1^2 + 9\alpha_2^2 + 9\alpha_3^2 + 9\alpha_4^2$$
$$-\alpha_1\alpha_2 - \alpha_1\alpha_3 - \alpha_1\alpha_4$$
$$-\alpha_2\alpha_3 - \alpha_2\alpha_4$$
$$-\alpha_3\alpha_4)$$

$$L = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)$$

Example (cont.)

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$$
$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$$
$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$$
$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$$
$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8} \quad >=0$$

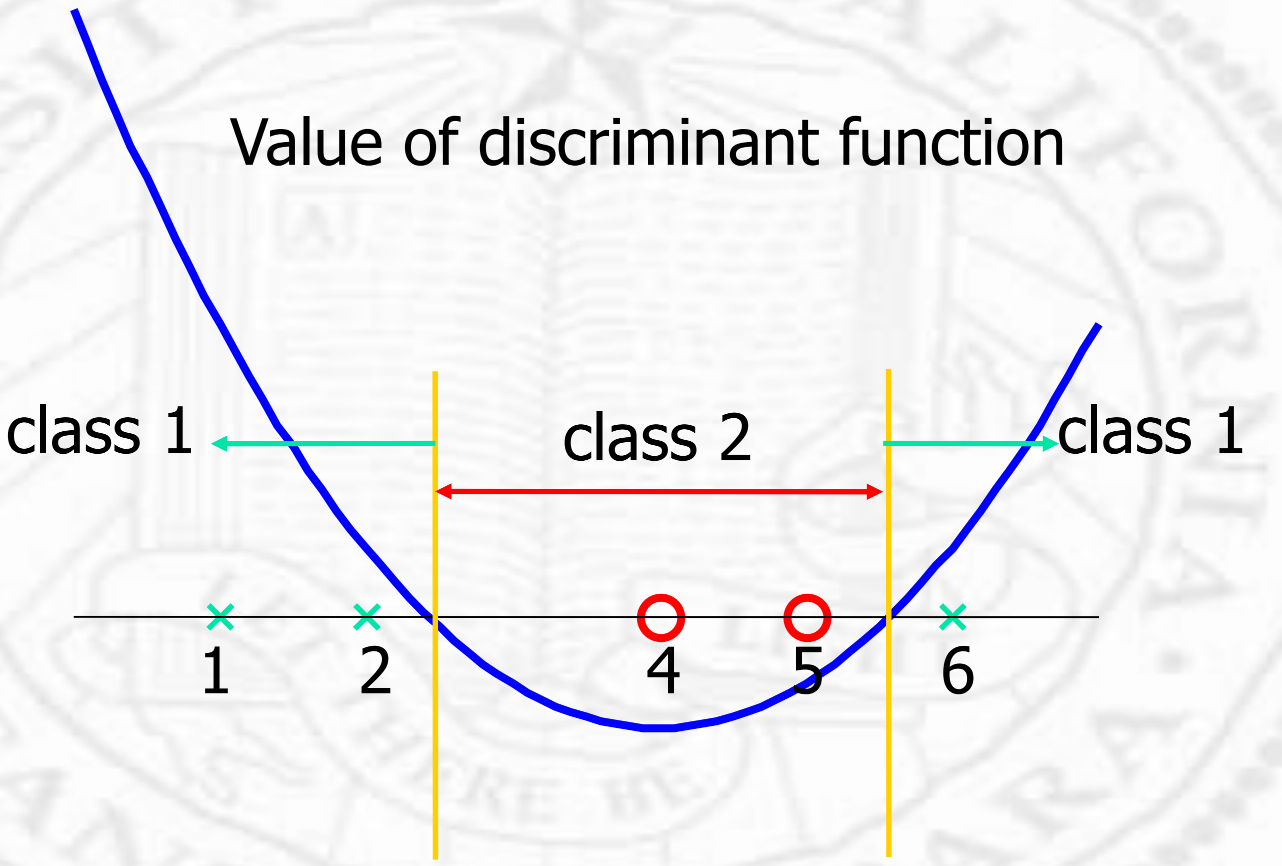
$$\boxed{y = < \mathbf{w}, \phi(\mathbf{x}) > = -x_1 x_2}$$

SVM can solve Xor!

$$\mathbf{w} = \sum_i \alpha_i y_i \phi(\mathbf{X}_i) = \frac{1}{8}[-\phi(\mathbf{X}_1) + \phi(\mathbf{X}_2) + \phi(\mathbf{X}_3) - \phi(\mathbf{X}_4)]$$

$$= \frac{1}{8}\left[\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix}\right] = \begin{bmatrix} 0 \\ 0 \\ -\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example: 5 1D data points



Example

- 5 1D data points
  - $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i ($i=1, \ldots, 5$) by

$$\text{max.} \quad \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

- By using a QP solver, we get

  $\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$

  - Verify (at home) that the constraints are indeed satisfied
  - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$

- The discriminant function is

$$\begin{aligned} f(y) &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

$$y_i(\mathbf{w}^T\phi(z)+b)=1$$

- *b* is recovered by solving $f(2)=1$ or by $f(5)=-1$ or by $f(6)=1$, as x_2, x_4, x_5 lie on f(y) and all give $b=9$ with

$$f(y) = 0.6667x^2 - 5.333x + 9$$

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Evaluation: Classic Reuters Data Set

- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles
- 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.

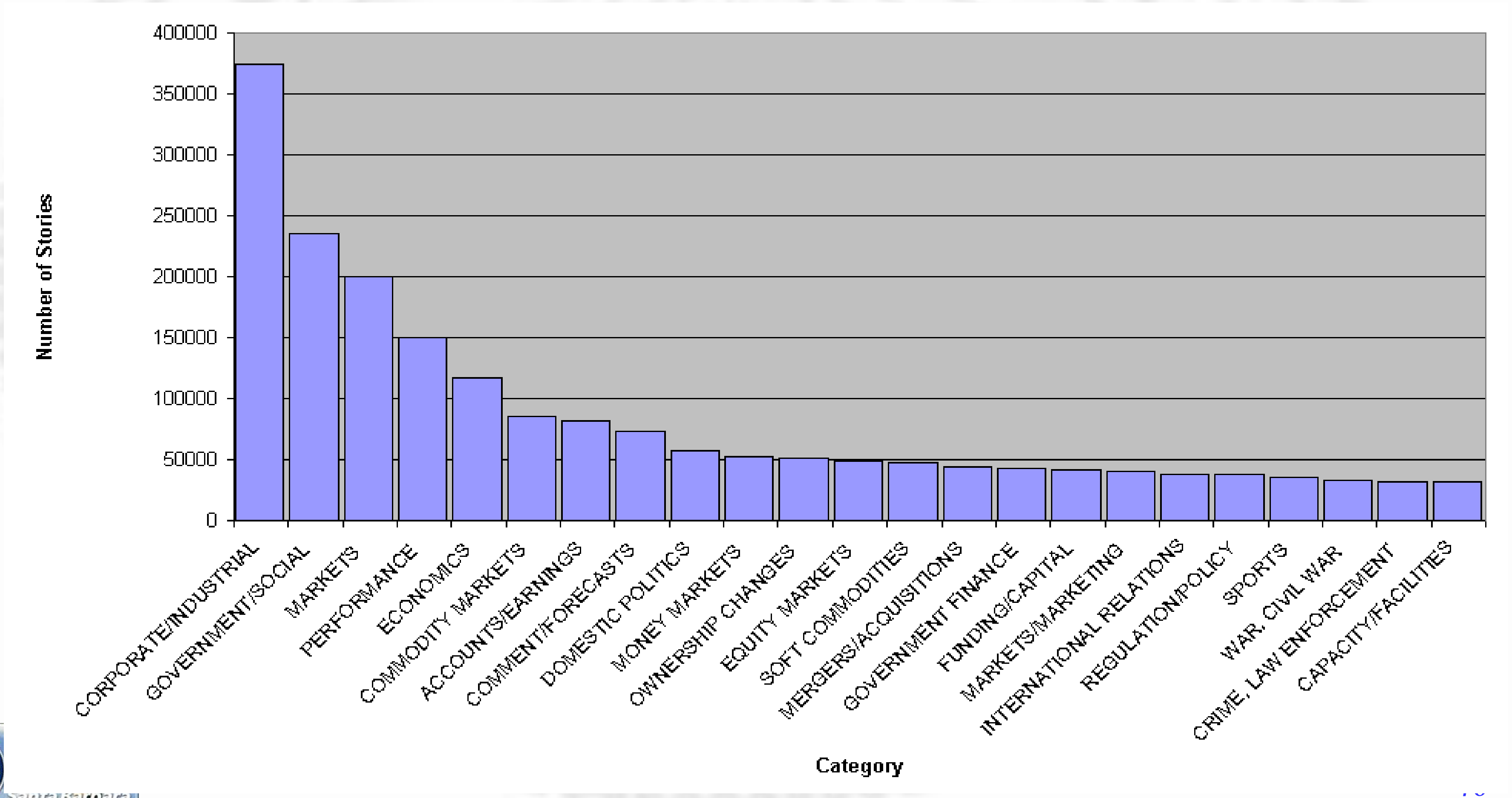
Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control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

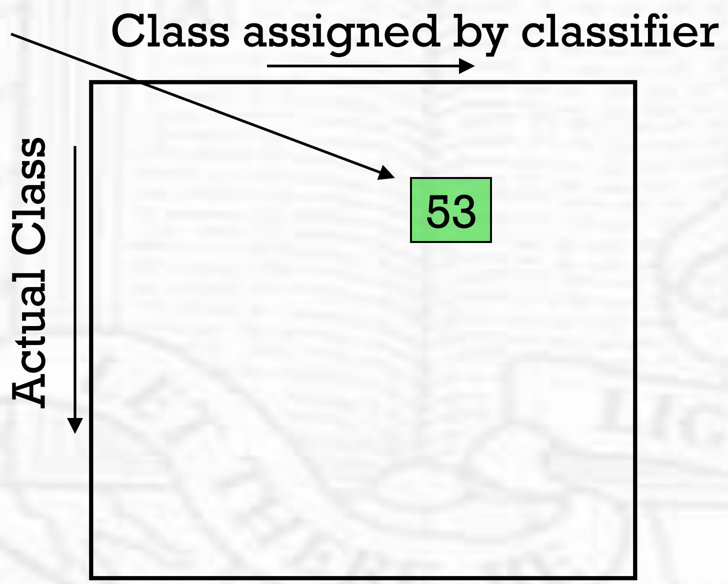
New Reuters: RCV1: 810,000 docs

- Top topics in Reuters RCV1



Good practice department: 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

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*: $\frac{c_{ii}}{\sum_j c_{ji}}$
- "Correct rate": (1- error rate) Fraction of docs classified correctly: $\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$

Dumais et al. 1998: Reuters - Accuracy

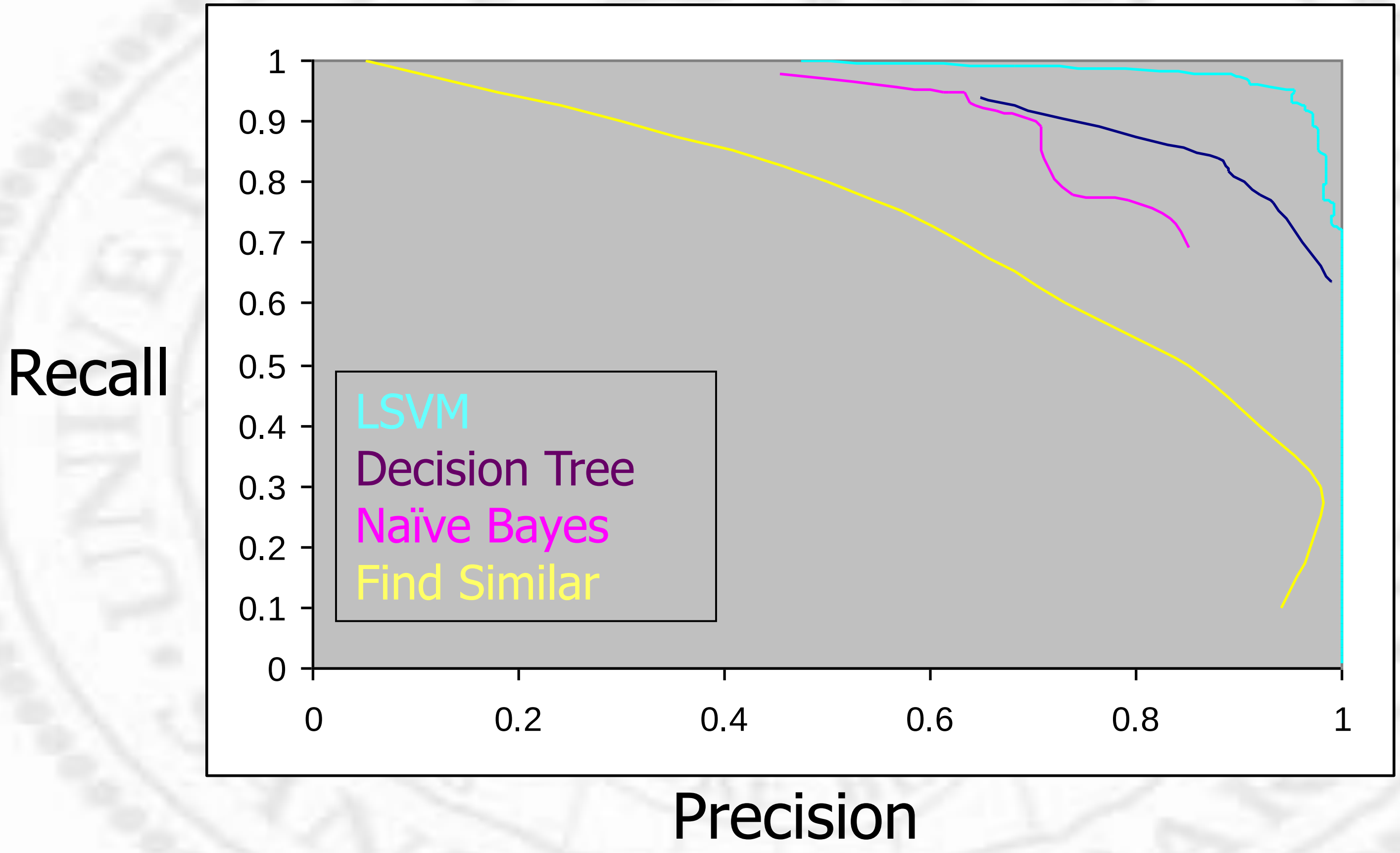
	Rocchio	NBayes	Trees	LinearSVM
earn	92.9%	95.9%	97.8%	98.2%
acq	64.7%	87.8%	89.7%	92.8%
money-fx	46.7%	56.6%	66.2%	74.0%
grain	67.5%	78.8%	85.0%	92.4%
crude	70.1%	79.5%	85.0%	88.3%
trade	65.1%	63.9%	72.5%	73.5%
interest	63.4%	64.9%	67.1%	76.3%
ship	49.2%	85.4%	74.2%	78.0%
wheat	68.9%	69.7%	92.5%	89.7%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

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

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

Break Even: (Recall + Precision) / 2

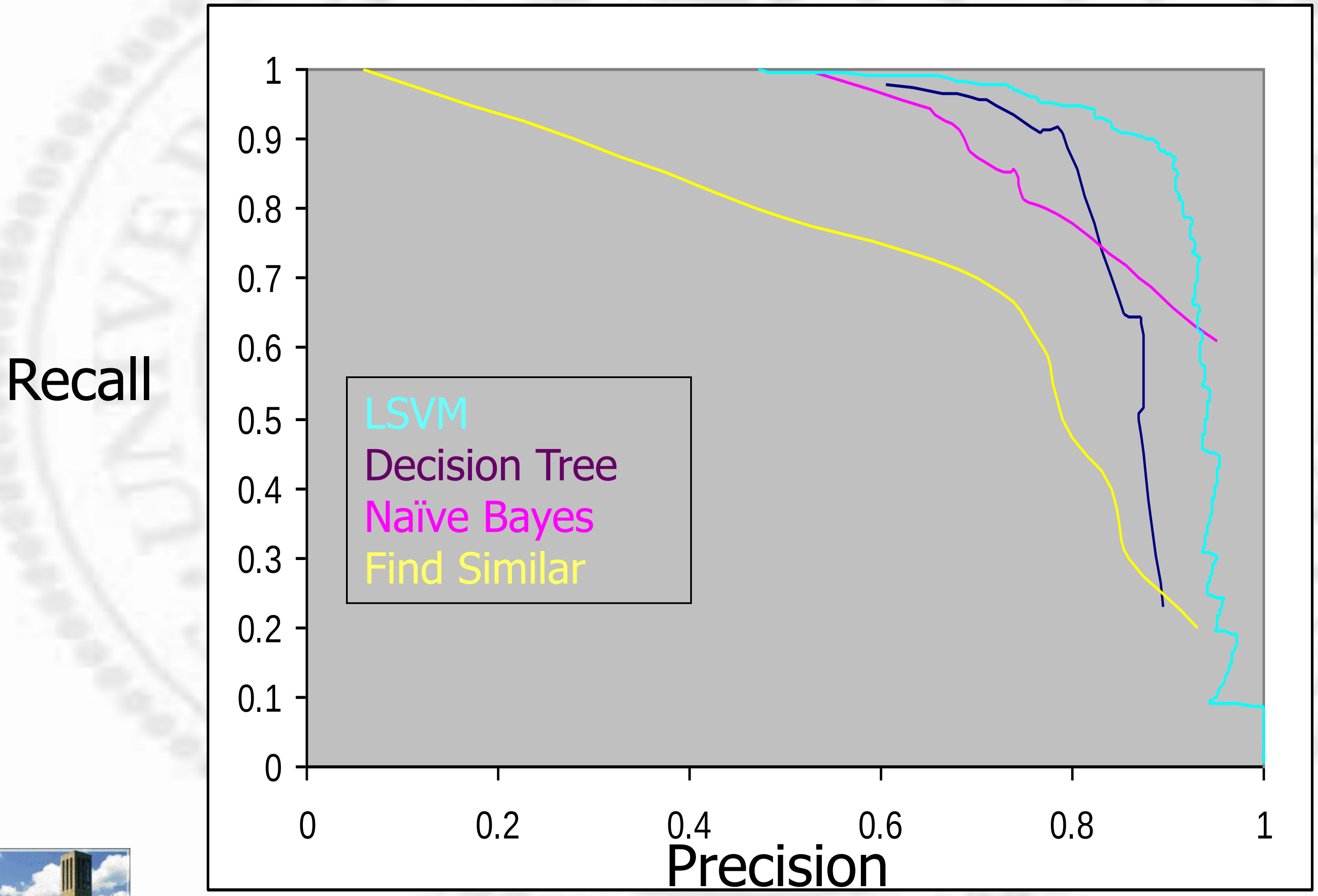
Reuters ROC - Category Grain



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

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

ROC for Category - Crude



Linear Programming

- Standard max
- Standard min

maximize $\boldsymbol{c}^T\boldsymbol{x}$
subject to the constraints $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ and $\boldsymbol{x} \geq 0$

minimize $\boldsymbol{y}^T\boldsymbol{b}$
subject to the constraints $\boldsymbol{y}^T\boldsymbol{A} \geq \boldsymbol{c}^T$ and $\boldsymbol{y} \geq 0$

Standard Max Example

- (x1,x2): (#pants, #shirts)
- (b1,b2): (#buttons, #zippers)
- (c1,c2): (profit/pant, profit/shirt)
- (y1, y2): ($/button, $/zipper) <- shadow price
- Aij: i:(buttons, zippers), j:(pants, shirts) use

$$\max \quad [c1, c2]\begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

$$\text{Subject to} \begin{bmatrix} bn1 & bn2 \\ zp1 & zp2 \end{bmatrix}\begin{bmatrix} x1 \\ x2 \end{bmatrix} \leq \begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

x1 ≥0, x2 ≥0

$$\min \quad [y1, y2]\begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

$$\text{Subject to } [y1, y2]\begin{bmatrix} bn1 & bn2 \\ zp1 & zp2 \end{bmatrix} \geq [c1 \quad c2]$$

y1 ≥0, y2 ≥0

Standard Max

- Max is primal
- Make pants/shirts by yourself
- Max profits for you
- Stay within available buttons and zippers

- Min is dual
- Sell your material to a buyer
- Min costs for buyer
- Buyer must offer more than what you can earn by making things yourself

Standard Min Example

- (y1,y2): (pound of meat, pound of veggi)
- (b1,b2): ($/pound meat, $/pound veggi)
- (c1,c2): (required calorie, required protein)
- Aij: i:(meat, veggi), j:(calorie, protein) provided
- (x1, x2): ($/1 calorie supplement, $/1 protein supplement) <- shadow price

$$\min \quad [y1, y2]\begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

Subject to $[y1, y2]\begin{bmatrix} c/pmeat & p/pmeat \\ c/pveggi & p/pveggi \end{bmatrix} \geq [c1 \quad c2]$

y1 $\geq$0, y2 $\geq$0

$$\max \quad [c1, c2]\begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

Subject to $\begin{bmatrix} c/meat & p/meat \\ c/veggi & p/veggi \end{bmatrix}\begin{bmatrix} x1 \\ x2 \end{bmatrix} \leq \begin{bmatrix} b1 \\ b2 \end{bmatrix}$

x1 $\geq$0, x2 $\geq$0

Standard Min

- Min is primal
- Buy meat and veggi to provide calorie and protein
- Min cost for you
- Supply enough nutrient

- Max is dual
- Aliens sell you magic calorie and protein pills
- Max profits for seller
- seller must offer less than what you pay to buy for yourself

Some Facts about LP

- All linear programming problems can be transformed into either standard max or standard min
- If max if feasible then min if feasible and vice versa
- Optimal max is also the optimal min (no gap)

Conversion from Primal to Dual

$$\min \quad [y1, y2]\begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

Subject to $[y1, y2]\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \geq [c1 \quad c2] \Rightarrow$ c1-a11*y1-a21*y2≤0

c2-a12*y1-a22*y2≤0

y1 ≥0, y2 ≥0

$max_{x1,x2} min_{y1,y2}$ b1*y1+b2*y2

+x1*(c1-a11*y1-a21*y2)

+x2*(c2-a12*y1-a22*y2)

$max_{x1,x2} min_{y1,y2} c1 * x1 + c2 * x2$

+y1*(b1-a11*x1-a12*x2)

+y2*(b2-a21*x1-a22*x2)

$\Downarrow$

$$\max \quad [c1, c2]\begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

Subject to $\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix}\begin{bmatrix} x1 \\ x2 \end{bmatrix} \leq \begin{bmatrix} b1 \\ b2 \end{bmatrix}$

x1 ≥0, x2 ≥0