Supervised Learning Methods

- k-nearest-neighbors
- Decision trees
- Neural networks
- Naïve Bayes
- **Support vector machines (SVM)**

12

Support Vector Machines

Chapter 18.9 and the optional paper "Support vector machines" by M. Hearst, ed., 1998

Acknowledgments: These slides combine and modify ones provided by Andrew Moore (CMU), Carla Gomes (Cornell), Mingyue Tan (UBC), Jerry Zhu (Wisconsin), Glenn Fung (Wisconsin), and Olvi Mangasarian (Wisconsin)

13

What are Support Vector Machines (SVMs) Used For?

- Classification
- Regression and data-fitting
- Supervised and unsupervised learning

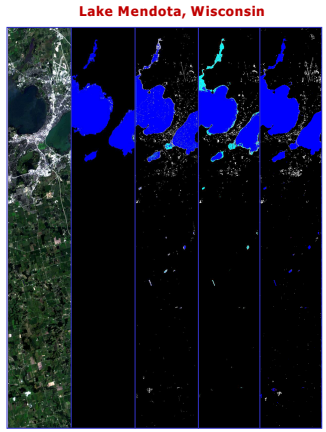
14

Lake Mendota, Madison, WI

- Identify areas of land cover (land, ice, water, snow) in a scene
- Two methods:
  - Scientist manually-derived
  - Support Vector Machine (SVM)

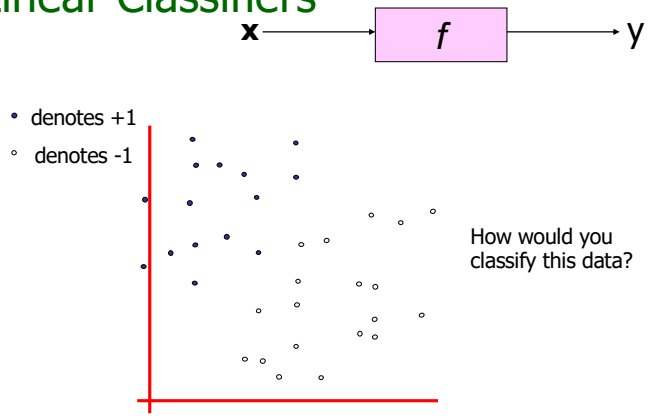
Classifier	Expert Derived	SVM
cloud	45.7%	58.5%
ice	60.1%	80.4%
land	93.6%	94.0%
snow	63.5%	71.6%
water	84.2%	89.1%
unclassified	45.7%	

Courtesy of Steve Chien of NASA/JPL



15

Linear Classifiers



16

Linear Classifiers (aka Linear Discriminant Functions)

- Definition:

  A function that is a linear combination of the components of the input (column vector) **x**:

$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j + b = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$

  where **w** is the weight (column vector) and *b* is the bias

- A **2-class classifier** then uses the rule:

  Decide class c_1 if $f(\mathbf{x}) \geq 0$ and class c_2 if $f(\mathbf{x}) < 0$

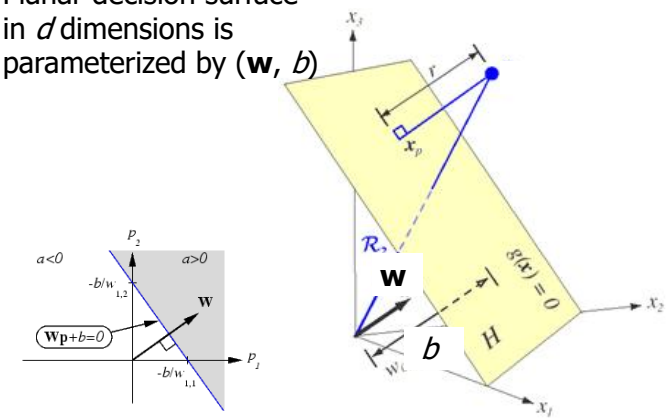
  or, equivalently, decide c_1 if $\mathbf{w}^{\mathrm{T}}\mathbf{x} \geq -b$ and c_2 otherwise

17

Planar decision surface in *d* dimensions is parameterized by (**w**, *b*)

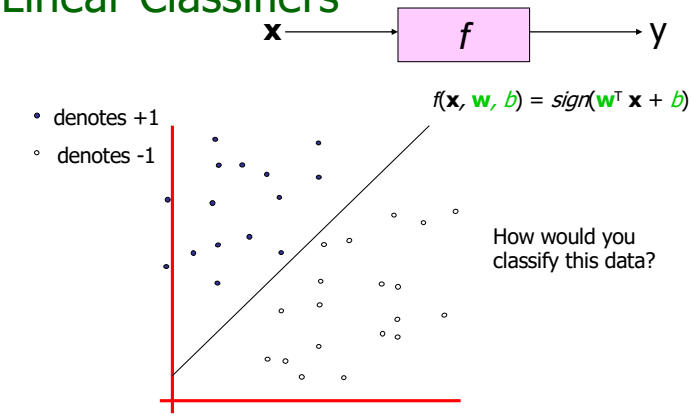
w is the plane's normal vector

b is the distance from the origin



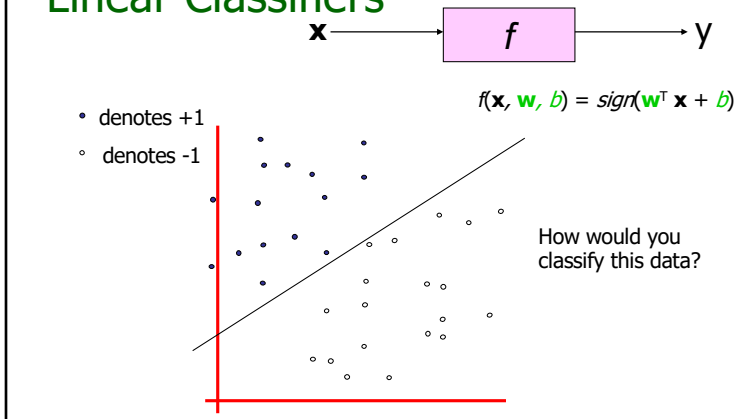
18

Linear Classifiers



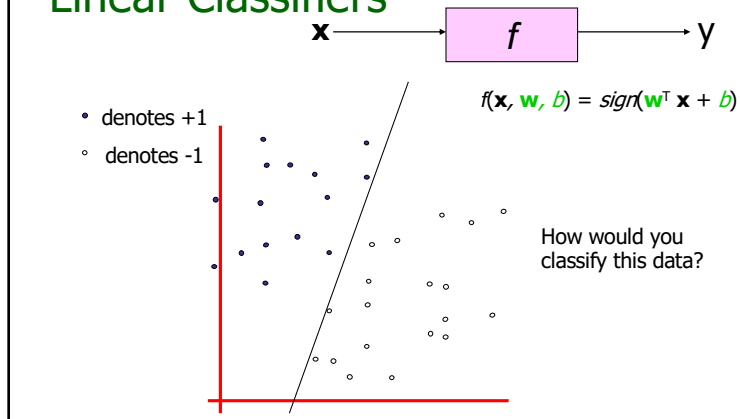
21

Linear Classifiers



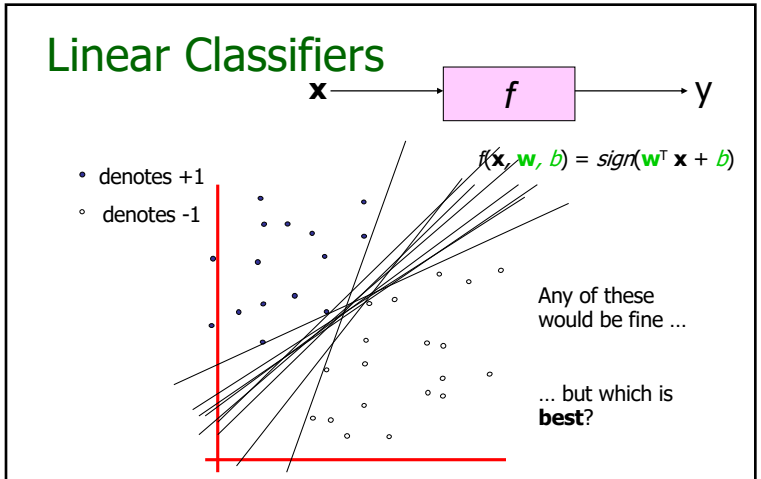
22

Linear Classifiers



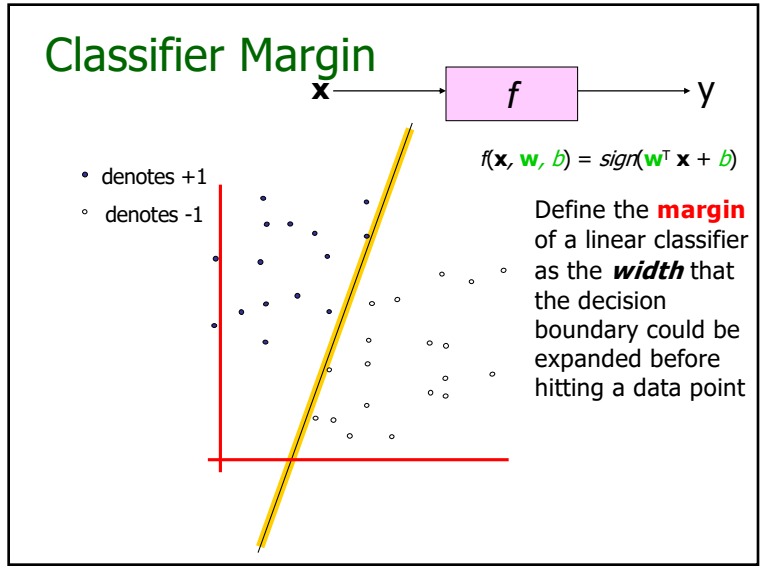
23

Linear Classifiers



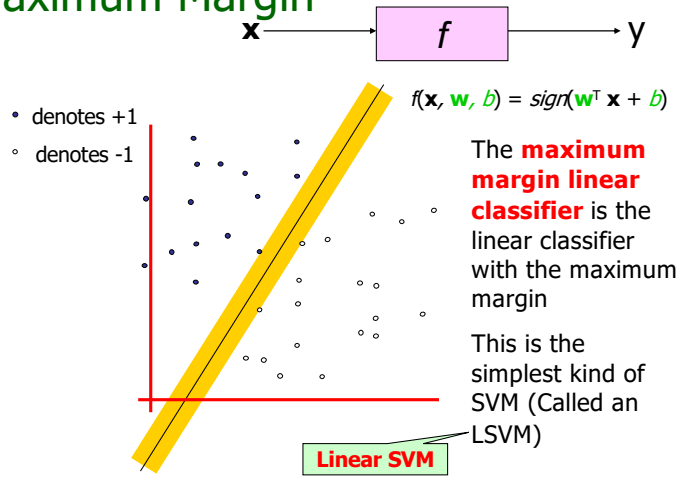
24

Classifier Margin



25

Maximum Margin

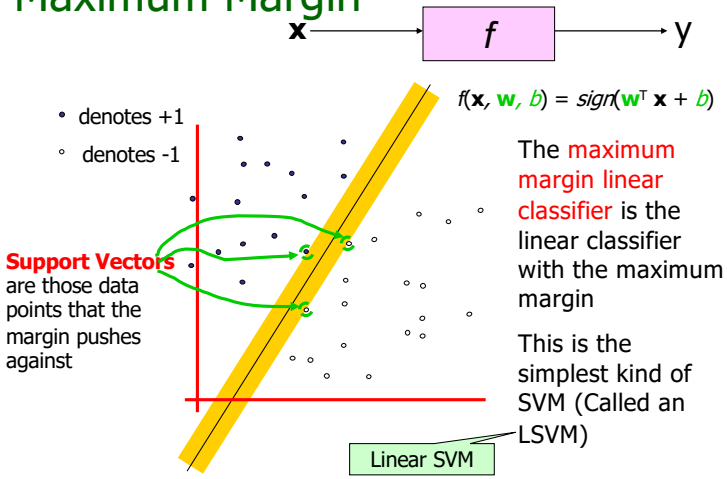


The **maximum margin linear classifier** is the linear classifier with the maximum margin

This is the simplest kind of SVM (Called an LSVM)

26

Maximum Margin

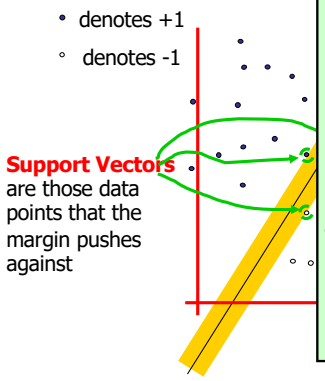


The maximum margin linear classifier is the linear classifier with the maximum margin

This is the simplest kind of SVM (Called an LSVM)

27

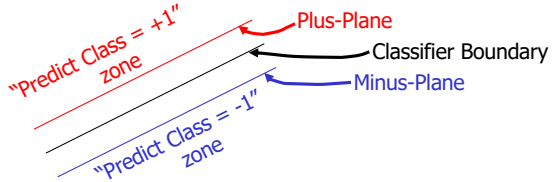
Why the Maximum Margin?



1. Intuitively this feels *safest*
2. If we've made a small *error* in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification
3. *Robust* to outliers since the model is immune to change/removal of any non-support-vector data points
4. There's some *theory* (using "VC dimension") that is related to (but not the same as) the proposition that this is a good thing
5. *Empirically it works very well*

28

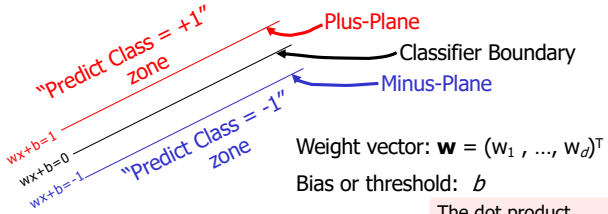
Specifying a Line and Margin



- How do we represent this mathematically?
- … in *d* input dimensions?
- An example $\mathbf{x} = (x_1, \ldots, x_d)^T$

29

Specifying a Line and Margin



Weight vector: $\mathbf{w} = (w_1, \ldots, w_d)^T$

Bias or threshold: b

- Plus-plane = $\mathbf{w}^T \mathbf{x} + b = +1$
- Minus-plane = $\mathbf{w}^T \mathbf{x} + b = -1$

The dot product $\mathbf{w}^T \cdot \mathbf{x} = \sum_{j=1}^{d} w_j x_j$ is a scalar: $\mathbf{x}$'s projection onto $\mathbf{w}$

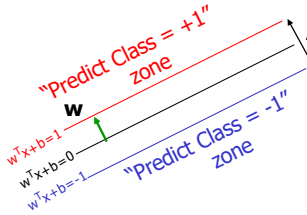
Classify as +1 if $\mathbf{w}^T \mathbf{x} + b \geq 1$

-1 if $\mathbf{w}^T \mathbf{x} + b \leq -1$

? if $-1 < \mathbf{w}^T \mathbf{x} + b < 1$

30

Computing the Margin



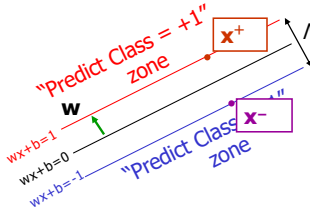
How do we compute M in terms of $\mathbf{w}$ and b?

- Plus-plane = $\mathbf{w}^T \mathbf{x} + b = +1$
- Minus-plane = $\mathbf{w}^T \mathbf{x} + b = -1$

Claim: The vector $\mathbf{w}$ is perpendicular to the Plus-Plane and the Minus-Plane

31

Computing the Margin



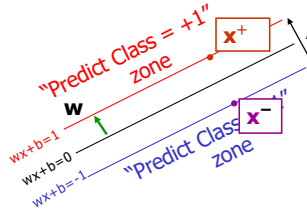
How do we compute M in terms of $\mathbf{w}$ and b?

- Plus-plane = $\mathbf{w}^T \mathbf{x} + b = +1$
- Minus-plane = $\mathbf{w}^T \mathbf{x} + b = -1$
- The vector $\mathbf{w}$ is perpendicular to the Plus-Plane
- Let $\mathbf{x}^-$ be any point on the Minus-Plane
- Let $\mathbf{x}^+$ be the closest Plus-Plane-point to $\mathbf{x}^-$

Any location in $\mathbb{R}^d$; not necessarily a data point

33

Computing the Margin

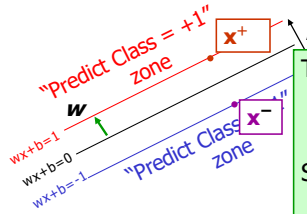


How do we compute M in terms of $\mathbf{w}$ and b?

- Plus-plane = $\mathbf{w}^T \mathbf{x} + b = +1$
- Minus-plane = $\mathbf{w}^T \mathbf{x} + b = -1$
- The vector $\mathbf{w}$ is perpendicular to the Plus-Plane
- Let $\mathbf{x}^-$ be any point on the Minus-Plane
- Let $\mathbf{x}^+$ be the closest Plus-Plane-point to $\mathbf{x}^-$
- Claim: $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$ for some value of λ

34

Computing the Margin



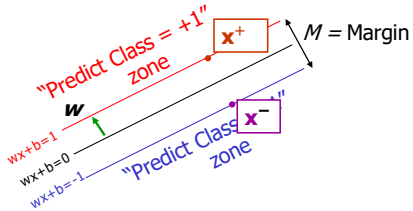
The line from $\mathbf{x}^-$ to $\mathbf{x}^+$ is perpendicular to the planes

So to get from $\mathbf{x}^-$ to $\mathbf{x}^+$ travel some distance in direction $\mathbf{w}$

- Plus-plane = $\mathbf{w}^T \mathbf{x} + b = +1$
- Minus-plane = $\mathbf{w}^T \mathbf{x} + b = -1$
- The vector $\mathbf{w}$ is perpendicular to the Plus-Plane
- Let $\mathbf{x}^-$ be any point on the Minus-Plane
- Let $\mathbf{x}^+$ be the closest Plus-Plane-point to $\mathbf{x}^-$
- Claim: $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$ for some value of λ Why?

35

Computing the Margin

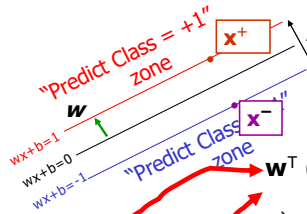


What we know:

- $\mathbf{w}^T \mathbf{x}^+ + b = +1$
- $\mathbf{w}^T \mathbf{x}^- + b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $||\mathbf{x}^+ - \mathbf{x}^-|| = M$

36

Computing the Margin



What we know:

- $\mathbf{w}^T \mathbf{x}^+ + b = +1$
- $\mathbf{w}^T \mathbf{x}^- + b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $||\mathbf{x}^+ - \mathbf{x}^-|| = M$

It's now easy to get M in terms of $\mathbf{w}$ and b

$\mathbf{w}^T (\mathbf{x}^- + \lambda \mathbf{w}) + b = 1$

$\Rightarrow$

$\mathbf{w}^T \mathbf{x}^- + b + \lambda \mathbf{w}^T \mathbf{w} = 1$

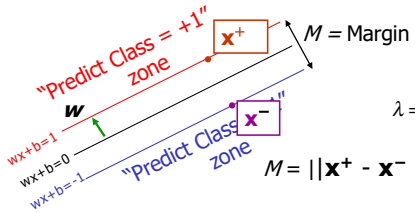
$\Rightarrow$

$-1 + \lambda \mathbf{w}^T \mathbf{w} = 1$

$\Rightarrow \quad \lambda = \dfrac{2}{\mathbf{w}^T \mathbf{w}}$

37

Computing the Margin



What we know:

- $\mathbf{w}^T \mathbf{x}^+ + b = +1$
- $\mathbf{w}^T \mathbf{x}^- + b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $||\mathbf{x}^+ - \mathbf{x}^-|| = M$

$M = \text{Margin} = \dfrac{2}{\sqrt{\mathbf{w}^T \mathbf{w}}}$

$\lambda = \dfrac{2}{\mathbf{w}^T \mathbf{w}}$

$M = ||\mathbf{x}^+ - \mathbf{x}^-|| = ||\lambda \mathbf{w}||$

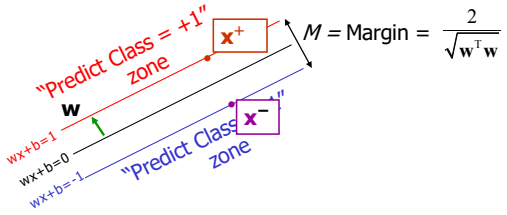
$= \lambda \|\mathbf{w}\| = \lambda \sqrt{\mathbf{w}^T \mathbf{w}}$

$= \dfrac{2\sqrt{\mathbf{w}^T \mathbf{w}}}{\mathbf{w}^T \mathbf{w}} = \dfrac{2}{\sqrt{\mathbf{w}^T \mathbf{w}}}$

$= \dfrac{2}{\|\mathbf{w}\|}$ = M, margin size

38

Learning the Maximum Margin Classifier



Given a guess of **w** and *b*, we can

1. Compute whether all data points in the correct half-planes
2. Compute the margin

So now we just need to write a program to search the space of **w**'s and *b*'s to find the widest margin that correctly classifies all the data points *How?*

39

SVM as Constrained Optimization

- Unknowns: **w**, *b*
- Objective function: maximize the margin:
  $M = 2 / ||\mathbf{w}||$
- Equivalent to **minimizing** $||\mathbf{w}||$ or $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

- *N* training points: $(\mathbf{x}_k, y_k)$, $y_k = +1$ or -1
- Subject to each training point correctly classified, i.e.,

  subject to $y_k(\mathbf{w}^T\mathbf{x}_k + b) \geq 1$ for all k — *N* constraints

This is a **quadratic optimization problem (QP)**, which can be solved efficiently

42

Classification with SVMs

Given a new point **x**, we can classify it by

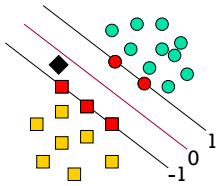
- Computing score: $\mathbf{w}^T\mathbf{x} + b$
- Deciding class based on whether < 0 or > 0

- If desired, can set confidence threshold *t*

Score > *t*: yes

Score < -*t*: no

Else: don't know



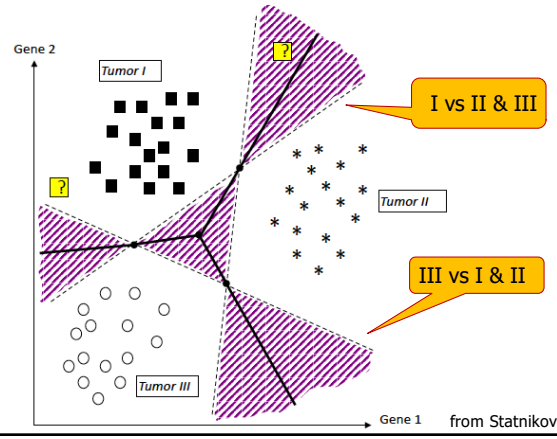
43

SVMs: More than Two Classes

- **SVMs can only handle two-class problems**
- *k*-class problem: Split the task into *k* **binary** tasks and learn *k* SVMs:
  - Class 1 vs. the rest (classes 2 — *k*)
  - Class 2 vs. the rest (classes 1, 3 — *k*)
  - …
  - Class *k* vs. the rest
- Pick the class that puts the point *farthest into its positive region*

44

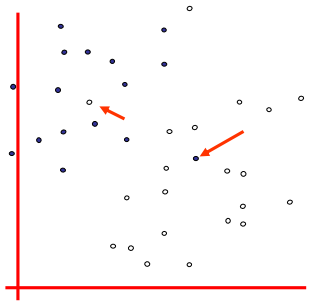
One-versus-rest multicategory SVM method



45

SVMs: Non Linearly-Separable Data

What if the data are ***not*** linearly separable?

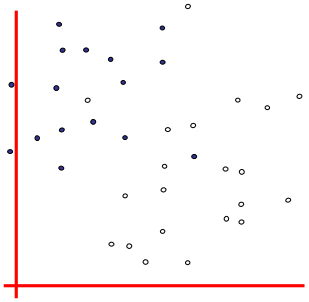


46

SVMs: Non Linearly-Separable Data

Two approaches:

- Allow a few points on the wrong side (**slack variables**)
- Map data to a higher dimensional space, and do linear classification there (**kernel trick**)



47

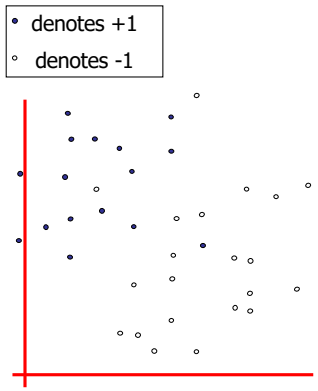
Non Linearly-Separable Data

Approach 1: Allow a few points on the wrong side (**slack variables**)

"Soft Margin Classification"

48

What Should We Do?



49

What Should We Do?

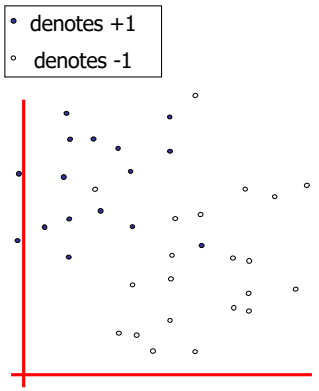
Idea:

Minimize

$||\mathbf{w}||^2 + C$(# train errors)

Tradeoff "penalty" parameter

There's a serious practical problem with this approach



51

What Should We Do?

Idea:

Minimize

$||\mathbf{w}||^2 + C$(# train errors)

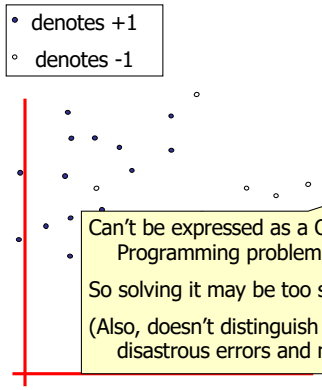
Tradeoff "penalty" parameter

erious practical
th this approach

Can't be expressed as a Quadratic Programming problem.

So solving it may be too slow.

(Also, doesn't distinguish between disastrous errors and near misses)

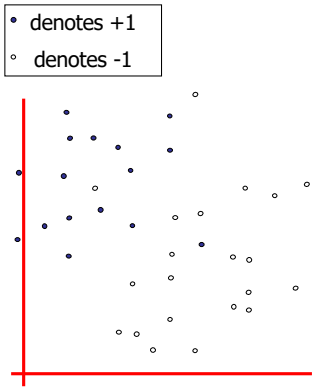


52

What Should We Do?

Minimize

$||\mathbf{w}||^2 + C$(distance of all "misclassified points" to their correct place)



53

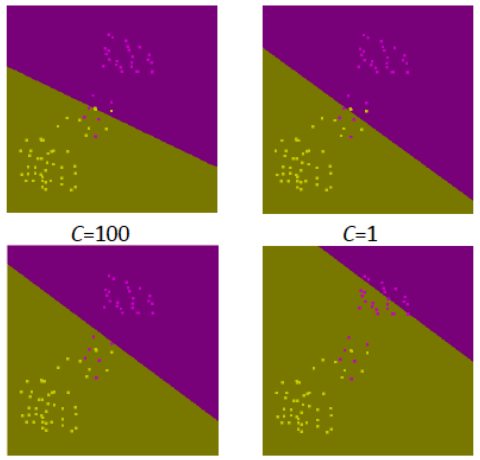
Choosing *C*, the Penalty Factor

Critical to choose a good value for the constant penalty parameter, *C*, because

- *C* too big means very similar to LSVM because we have a high penalty for non-separable points, and we may use many support vectors and ***overfit***
- *C* too small means we allow many misclassifications in the training data and we may ***underfit***

54

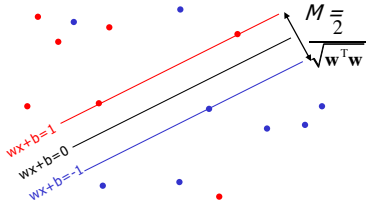
Choosing *C*



from Statnikov et al.

55

Learning Maximum Margin with Noise



Given guess of **w**, *b*, we can

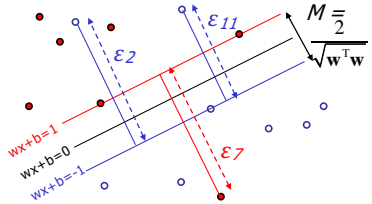
1. Compute sum of distances of points to their correct zones
2. Compute the margin width

Assume *N* examples, each ($\mathbf{x}_k$, y_k) where y_k = +1 / -1

What should our optimization criterion be?

56

Learning Maximum Margin with Noise



"slack variables"

Given guess of **w**, *b*, we can

1. Compute sum of distances of points to their correct zones
2. Compute the margin width

Assume *N* examples, each ($\mathbf{x}_k$, y_k) where y_k = +1 / -1

What should our optimization criterion be?

Minimize $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{k=1}^{N}\varepsilon_k$

How many constraints will we have? ***N***

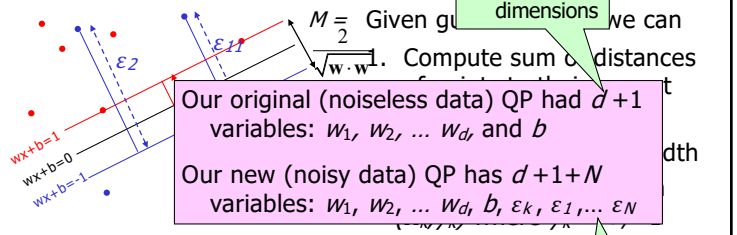
What should they be?

$y_k(\mathbf{w}^{\mathrm{T}}x_k + b) \geq 1-\varepsilon_k$ for all k

Note: ε_k = 0 for points in correct zone

57

Learning Maximum Margi…ise



$M = \frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$ Given gu…we can

1. Compute sum o…distances …t …dth …

d = # input dimensions

Our original (noiseless data) QP had *d* +1 variables: $w_1, w_2, \ldots w_d$ and *b*

Our new (noisy data) QP has *d* +1+*N* variables: $w_1, w_2, \ldots w_d, b, \varepsilon_k, \varepsilon_1, \ldots \varepsilon_N$

N = # examples

What should our optimization criterion be?

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N}\varepsilon_k$

How many constrai…ll we have? ***N***

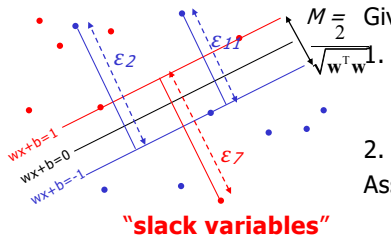
What should they be?

$\mathbf{w}^T\mathbf{x}_k + b \geq 1 - \varepsilon_k$ if $y_k = +1$

$\mathbf{w}^T\mathbf{x}_k + b \leq -1 + \varepsilon_k$ if $y_k = -1$

58

Learning Maximum Margin with Noise



"slack variables"

$M = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$ Given guess of **w** , *b* we can

1. Compute sum of distances of points to their correct zones
2. Compute the margin width

Assume *N* examples, each $(\mathbf{x}_k, y_k)$ where y_k = +1 / -1

What should our optimization criterion be?

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N}\varepsilon_k$

How many constraints will we have? ***2N***

What should they be?

$\mathbf{w}^T\mathbf{x}_k + b \geq +1 - \varepsilon_k$ if $y_k = +1$

$\mathbf{w}^T\mathbf{x}_k + b \leq -1 + \varepsilon_k$ if $y_k = -1$

$\varepsilon_k \geq 0$ for all *k*

60

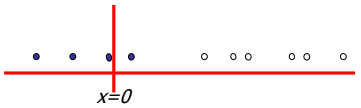
Non Linearly-Separable Data

Approach 2: Map data to a higher dimensional space, and then do linear classification there (called the **kernel trick**)

64

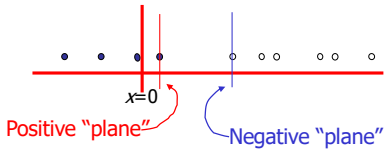
Suppose we're in 1 Dimension

What would SVMs do with this data?



65

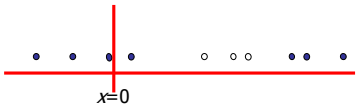
Suppose we're in 1 Dimension



66

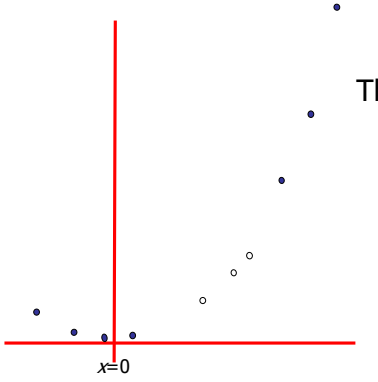
Harder 1D Dataset: *Not* Linearly-Separable

What can be done about this?



67

Harder 1D Dataset



The **Kernel Trick**: Preprocess the data, mapping **x** into a higher dimensional space, **z** = Φ(**x**)

For example:

$$\Phi(x) = (x\ , x^2)$$

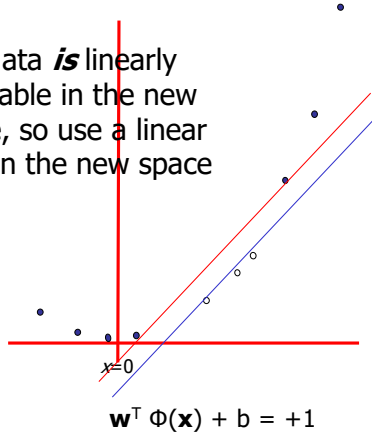
Here, Φ maps data from 1D to 2D

In general, map from *d*-dimensional input space to *k*-dimensional **z** space

68

Harder 1D Dataset

The data ***is*** linearly separable in the new space, so use a linear SVM in the new space



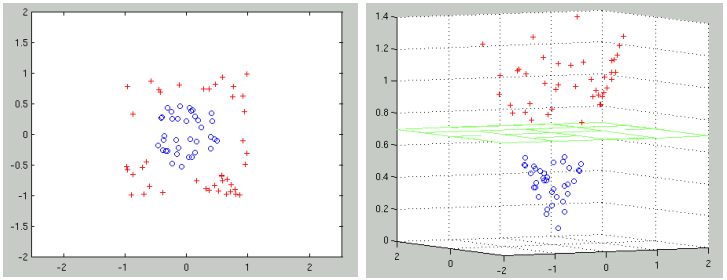
The **Kernel Trick**: Preprocess the data, mapping **x** into a higher dimensional space, **z** = Φ(**x**)

$$\Phi(x) = (x\ , x^2)$$

69

Another Example

$$\Phi(x_1, x_2) = (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$

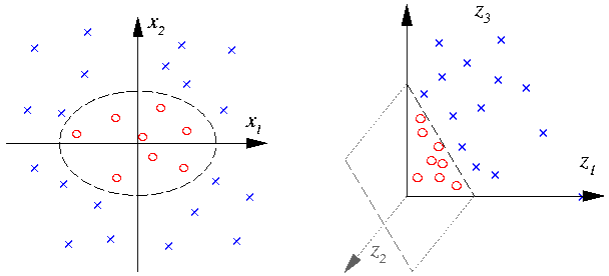


Project examples into some higher dimensional space where the data ***is*** linearly separable, defined by **z** = Φ(**x**)

70

Another Example

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



Project examples into some higher dimensional space where the data ***is*** linearly separable, defined by **z** = Φ(**x**)

71

Algorithm

1. Pick a Φ function
2. Map each training example, **x**, into the new higher-dimensional space using **z** = Φ(**x**)
3. Solve the optimization problem using the nonlinearly transformed training examples, **z**, to obtain a Linear SVM (with or without using the slack variable formulation) defined by **w** and *b*
4. Classify a test example, **e**, by computing: *sign*(**w**^{T} · Φ(**e**) + *b*)

72

Improving Efficiency

- Time complexity of the original optimization formulation depends on the dimensionality, *k*, of **z** (*k* >> *d*)
- We can convert the optimization problem into an equivalent form, called the *Dual Form*, with time complexity $O(N^3)$ that depends on *N*, the number of training examples, *not k*
- Dual Form will also allow us to rewrite the mapping functions in Φ in terms of "kernel functions" instead

73

Dual Optimization Problem

Maximize $\sum_{k=1}^{N} \alpha_k + \sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k^{\mathrm{T}} . \mathbf{x}_l)$

subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\quad \sum_{k=1}^{N} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

Then classify with:

f(**x**,**w**,*b*) = *sign*(**w**$^{\mathrm{T}}$**x** - *b*)

N examples: ($\mathbf{x}_k$, y_k)
where y_k = +1 / -1

74

Dual Optimizatio

Dot product of two examples

Maximize $\sum_{k=1}^{N} \alpha_k + \sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k^{\mathrm{T}} . \mathbf{x}_l)$

subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\quad \sum_{k=1}^{N} \alpha_k y_k = 0$

New variables; Examples with $\alpha_k > 0$ will be the support vectors

Then define:

$$\mathbf{w} = \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

Then classify with:

f(**x**,**w**,*b*) = *sign*(**w**$^{\mathrm{T}}$**x** - *b*)

N examples: ($\mathbf{x}_k$, y_k)
where y_k = +1 / -1

75

Algorithm

- Compute *N* x *N* matrix *Q* by computing $y_i y_j (\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j)$ between *all* pairs of training examples
- Solve the optimization problem to compute α_i
  for *i* = 1, …, N
  - Each non-zero α_i indicates that example $\mathbf{x}_i$ is a support vector
- Compute **w** and b
- Then classify test example **x** with:

  f(**x**) = *sign*(**w**$^{\mathrm{T}}$ **x** – *b*)

76

Dual Optimization Problem (After Mapping)

Maximize $\sum_{k=1}^{N} \alpha_k - \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k)^{\mathrm{T}} \Phi(\mathbf{x}_l))$

Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\quad \sum_{k=1}^{N} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

Then classify with:

f(**x**,**w**,*b*) = *sign*(**w**$^{\mathrm{T}}$ Φ(***x***) - *b*)

N examples: ($\mathbf{x}_k$, y_k)
where y_k = +1 / -1

80

Dual Optimizzation Problem (After Mapping)

Maximize $\sum_{k=1}^{N}\alpha_k - \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_k\alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l(\Phi(\mathbf{x}_k)^{\mathrm{T}}\Phi(\mathbf{x}_l))$

Subject to these constraints: $0 \le \alpha_k \le$ *N* (N dot products to compute this matrix) $\sum_{k=1}^{N} y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \boldsymbol{\Phi}(\mathbf{x}_k)$$

$$b = y_K(1-\varepsilon_K) - \mathbf{x}_K.\mathbf{w}_K \quad \text{where } K = \arg\max_k \alpha_k$$

Then classify with:

f(**x**,**w**,*b*) = *sign*(**w**$^{\mathrm{T}}$ Φ(**x**) - *b*)

N examples: ($\mathbf{x}_k$, y_k)
where y_k = +1 / -1

81

- Dual formulation of the optimization problem depends on the input data only in dot products of the form:
  $\Phi(\mathbf{x}_i)^{\mathrm{T}} \cdot \Phi(\mathbf{x}_j)$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two examples
- We can compute these dot products efficiently for certain types of Φ's where $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^{\mathrm{T}} \cdot \Phi(\mathbf{x}_j)$
- Example:

$$\Phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\Phi(\mathbf{x}_i)^{\mathrm{T}} \cdot \Phi(\mathbf{x}_j) = (\mathbf{x}_i^{\mathrm{T}} \cdot \mathbf{x}_j)^2 = K(\mathbf{x}_i, \mathbf{x}_j)$$

- Since the data *only* appears as dot products, we do *not* need to map the data to higher dimensional space (using Φ(**x**)) because we can use the kernel function *K* instead

82

$$\Phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\begin{aligned}
&\Phi(x_i)^{\mathrm{T}} \cdot \Phi(x_j) \\
&= x_{i_1}^2 x_{j_1}^2 + \sqrt{2}x_{i_1}x_{i_2}\sqrt{2}x_{j_1}x_{j_2} + x_{i_2}^2 x_{j_2}^2 \\
&= x_{i_1}^2 x_{j_1}^2 + 2x_{i_1}x_{i_2}x_{j_1}x_{j_2} + x_{i_2}^2 x_{j_2}^2 \\
&= (x_{i_1}x_{j_1} + x_{i_2}x_{j_2})^2 \\
&= (x_i^{\mathrm{T}} \cdot x_j)^2
\end{aligned}$$

83

Kernel Functions

- A kernel, $K(\mathbf{x}_i, \mathbf{x}_j)$, is a dot product in *some* feature space
- A kernel function is a function that can be applied to pairs of input examples to evaluate dot products in some corresponding (possibly infinite dimensional) feature space
- We do *not* need to compute Φ explicitly

84

What's Special about a Kernel?

- Say 1 example (in 2D) is: $\mathbf{s} = (s_1, s_2)$
- We decide to use a particular mapping into 6D space:
  $\Phi(\mathbf{s})^T = (s_1^2, s_2^2, \sqrt{2}s_1s_2, s_1, s_2, 1)$
- Let another example be $\mathbf{t} = (t_1, t_2)$
- Then,

$$\Phi(\mathbf{s})^T \cdot \Phi(\mathbf{t}) = s_1^2 t_1^2 + s_2^2 t_2^2 + 2s_1s_2t_1t_2 + s_1t_1 + s_2t_2 + 1$$
$$= (s_1t_1 + s_2t_2 + 1)^2$$
$$= (\mathbf{s}^T\cdot\mathbf{t} + 1)^2$$

- So, define the **kernel** function to be $K(\mathbf{s}, \mathbf{t}) = (\mathbf{s}^T\cdot\mathbf{t} + 1)^2 = \Phi(\mathbf{s})^T \cdot \Phi(\mathbf{t})$
- We save computation by using kernel K

85

Some Commonly Used Kernels

- Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Quadratic kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^2$
- Polynomial of degree d kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^d$
- Radial-Basis Function (Gaussian) kernel:
  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2 / \sigma^2)$

- Many possible kernels; picking a good one is tricky
- Hacking with SVMs: create various kernels, hope their space Φ is meaningful, plug them into SVM, pick one with good classification accuracy
- Kernel usually combined with slack variables because no guarantee of linear separability in new space

86

Algorithm

- Compute N x N matrix Q by computing
  $y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ between all pairs of training points
- Solve optimization problem to compute α_i
  for $i = 1, \ldots, N$
  - Each non-zero α_i indicates that example $\mathbf{x}_i$ is a support vector
- Compute $\mathbf{w}$ and b
- Classify test example $\mathbf{x}$ using:
  $f(\mathbf{x}) = sign(\mathbf{w}^T \mathbf{x} - b)$

89

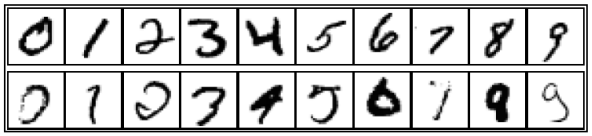
Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

→ Lots of very successful applications!

92

Handwritten Digit Recognition



3-nearest-neighbor = 2.4% error
400–300–10 unit MLP = 1.6% error
LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms) ≈ 0.6% error

93

Example Application: The Federalist Papers Dispute

- Written in 1787-1788 by Alexander Hamilton, John Jay, and James Madison to persuade the citizens of New York to ratify the U.S. Constitution
- Papers consisted of short essays, 900 to 3500 words in length
- Authorship of 12 of those papers have been in dispute (Madison or Hamilton); these papers are referred to as the disputed Federalist papers

94

Description of the Data

- For every paper:
  - Computed relative frequencies of 70 words that Mosteller-Wallace identified as good candidates for author-attribution studies
  - Each document is represented as a vector containing the 70 real numbers corresponding to the 70 word frequencies

"Bag of words"

- The dataset consists of 118 papers:
  - 50 Madison papers
  - 56 Hamilton papers
  - 12 disputed papers

95

70-Word Dictionary

1	*a*	15	*do*	29	*is*	43	*or*	57	*this*
2	*all*	16	*down*	30	*it*	44	*our*	58	*to*
3	*also*	17	*even*	31	*its*	45	*shall*	59	*up*
4	*an*	18	*every*	32	*may*	46	*should*	60	*upon*
5	*and*	19	*for*	33	*more*	47	*so*	61	*was*
6	*any*	20	*from*	34	*must*	48	*some*	62	*were*
7	*are*	21	*had*	35	*my*	49	*such*	63	*what*
8	*as*	22	*has*	36	*no*	50	*than*	64	*when*
9	*at*	23	*have*	37	*not*	51	*that*	65	*which*
10	*be*	24	*her*	38	*now*	52	*the*	66	*who*
11	*been*	25	*his*	39	*of*	53	*their*	67	*will*
12	*but*	26	*if*	40	*on*	54	*then*	68	*with*
13	*by*	27	*in*	41	*one*	55	*there*	69	*would*
14	*can*	28	*into*	42	*only*	56	*things*	70	*your*

96

Feature Selection for Classifying the Disputed Federalist Papers

- Apply the SVM algorithm for feature selection to:
  - Train on the 106 Federalist papers with known authors
  - Find a classification hyperplane (LSVM) that uses as few words as possible
- Use the hyperplane to classify the 12 disputed papers

97

Hyperplane Classifier Using 3 Words

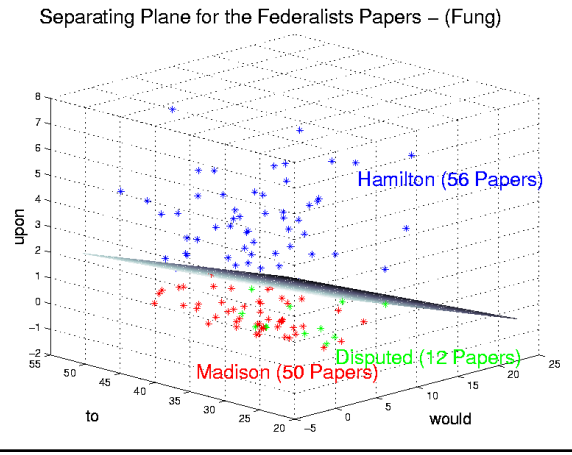
- A hyperplane depending on three words was found:

$$0.537\textit{to} + 24.663\textit{upon} + 2.953\textit{would} = 66.616$$

- All disputed papers ended up on the Madison side of the plane

98

Results: 3D Plot of Hyperplane



99

Summary

- Learning linear functions
  - Pick ***separating hyperplane*** that maximizes margin
  - Separating plane defined in terms of support vectors (small number of training examples) *only*
- Learning non-linear functions
  - Project examples into a higher dimensional space
  - Use kernel functions for efficiency
- Generally avoids overfitting problem
- Global optimization method; no local optima
- Can be expensive to apply, especially for multi-class problems
- Biggest Drawback: The choice of kernel function
  - There is no "set-in-stone" theory for choosing a kernel function for any given problem
  - Once a kernel function is chosen, there is only ONE modifiable parameter, the error penalty *C*

101

Software

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

102