Support Vector Machines

- Optimally defined decision surface
- Typically nonlinear in the input space
- Linear in a higher dimensional space
- Implicitly defined by a kernel function

Acknowledgments: These slides combine and modify ones provided by Andrew Moore (CMU), Jerry Zhu (Wisconsin), Glenn Fung (Wisconsin), and Olvi Mangasarian (Wisconsin)

What are Support Vector Machines Used For?

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

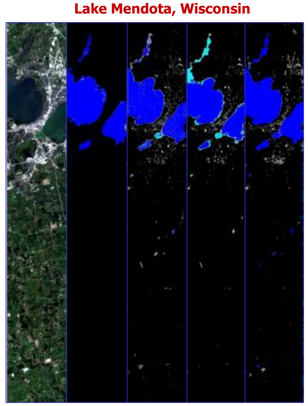
Lake Mendota, Madison, WI

- 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%	

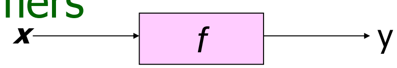
Courtesy of Steve Chien of NASA/JPL

Lake Mendota, Wisconsin



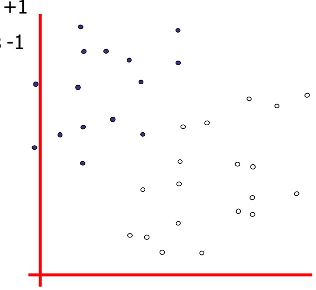
Visible Image | Expert Labeled | Expert Derived | Automated Ratio | SVM

Linear Classifiers



$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

- • denotes +1
- ◦ denotes -1



How would you classify this data?

Linear Classifiers
(aka Linear Discriminant Functions)

- Definition

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

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

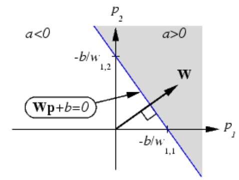
  where **w** is the weight (column vector) and *b* is the bias
- A **2-class classifier** then uses the rule:
  Decide class c_1 if $f(\boldsymbol{x}) \geq 0$ and class c_2 if $f(\boldsymbol{x}) < 0$
  $\Leftrightarrow$ Decide c_1 if $\mathbf{w}^{\mathrm{T}}\boldsymbol{x} \geq -b$ and c_2 otherwise

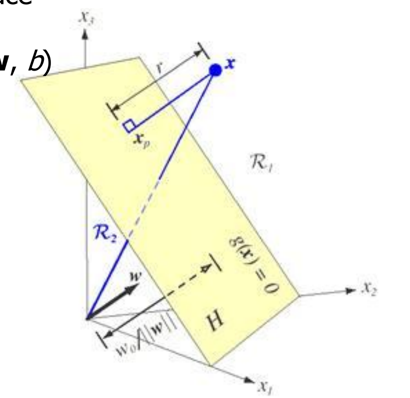
---

w is the plane's normal vector

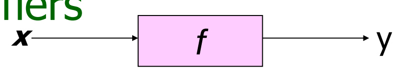
b is the distance from the origin

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



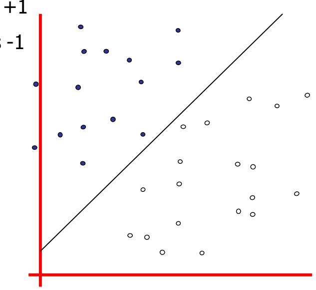


Linear Classifiers



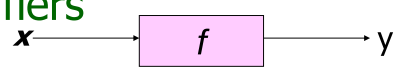
$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

- denotes +1
- denotes -1



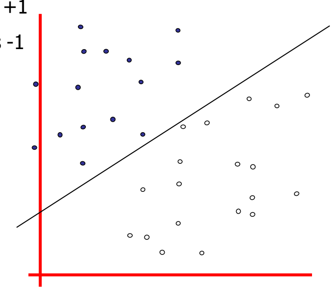
How would you classify this data?

Linear Classifiers



$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

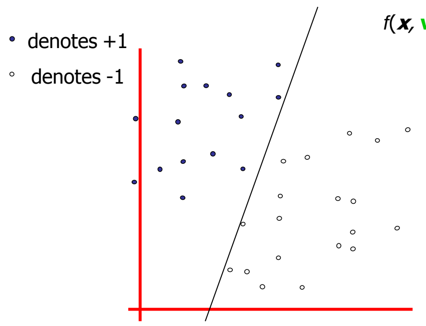
- denotes +1
- denotes -1



How would you classify this data?

Linear Classifiers

$\boldsymbol{x} \rightarrow \boxed{f} \rightarrow y$

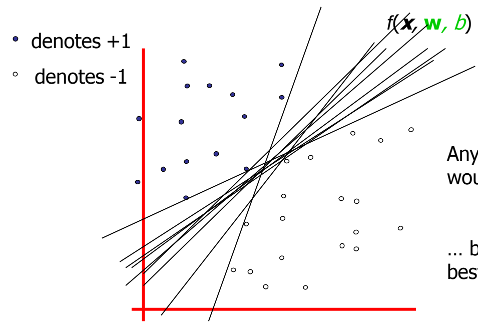


$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

How would you classify this data?

Linear Classifiers

$\boldsymbol{x} \rightarrow \boxed{f} \rightarrow y$



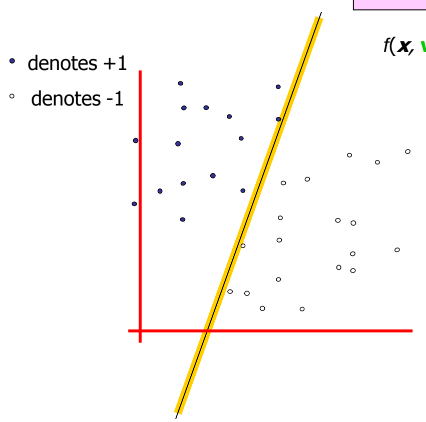
$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

Any of these would be fine ...

... but which is best?

Classifier Margin

$\boldsymbol{x} \rightarrow \boxed{f} \rightarrow y$

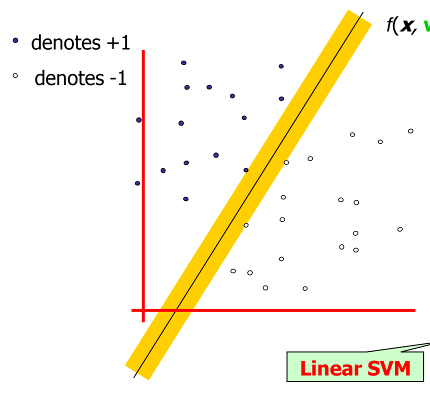


$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a data point

Maximum Margin

$\boldsymbol{x} \rightarrow \boxed{f} \rightarrow y$

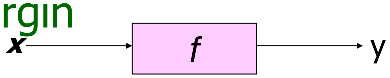


$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$

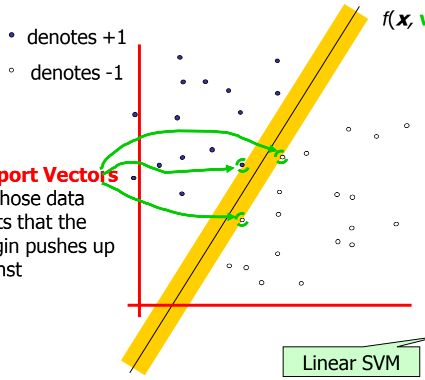
The **maximum margin linear classifier** is the linear classifier with the maximum margin!

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

Maximum Margin



$f(\boldsymbol{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \boldsymbol{x} + b)$



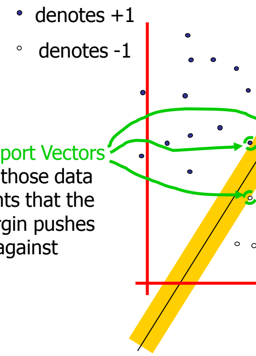
Support Vectors are those data points that the margin pushes up against

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

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

Linear SVM

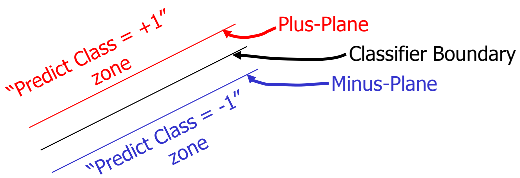
Why Maximum Margin?



Support Vectors are those data points that the margin pushes up against

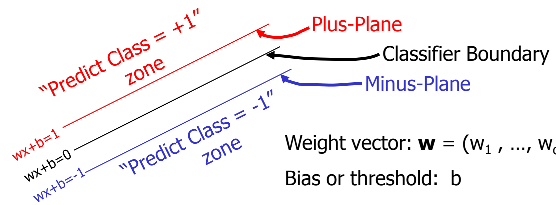
1. Intuitively this feels safest
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification
3. Robust to outliers since the model is immune to change/removal of any non-support-vector data 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

Specifying a Line and Margin



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

Specifying a Line and Margin



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

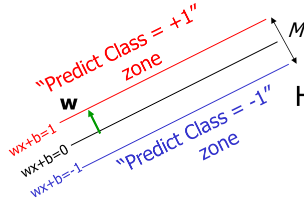
Bias or threshold: b

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

The dot product $\mathbf{w}^T \cdot \boldsymbol{x} = \sum w_i x_i$ is a scalar: $\boldsymbol{x}$'s projection onto $\mathbf{w}$

Classify as	+1	if	$\mathbf{w}^T \cdot \boldsymbol{x} + b \geq 1$
	-1	if	$\mathbf{w}^T \cdot \boldsymbol{x} + b \leq -1$
	?	if	$-1 < \mathbf{w}^T \cdot \boldsymbol{x} + b < 1$

Computing the Margin



M = Margin (width)

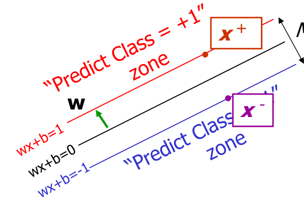
How do we compute M in terms of **w** and b?

- Plus-plane = $\mathbf{w}\,\boldsymbol{x} + b = +1$
- Minus-plane = $\mathbf{w}\,\boldsymbol{x} + b = -1$

Note: From now on, transpose symbol on **w** implied

Claim: The vector **w** is perpendicular to the Plus-Plane

Computing the Margin



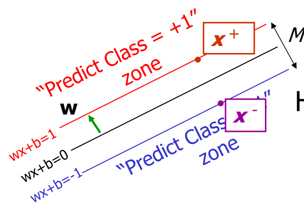
M = Margin

How do we compute M in terms of **w** and b?

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

Any location in R^m: not necessarily a data point

Computing the Margin

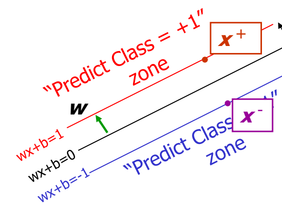


M = Margin

How do we compute M in terms of **w** and b?

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

Computing the Margin



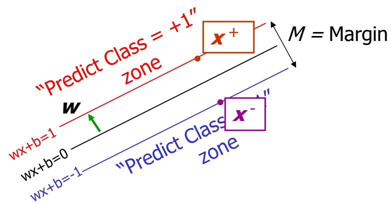
M = Margin

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

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

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

Computing the Margin

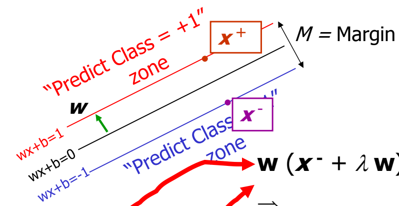


What we know:

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

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

Computing the Margin



What we know:

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

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

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

$\Rightarrow$

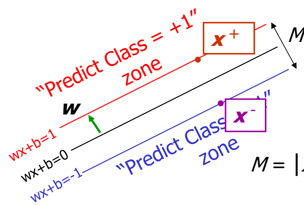
$\mathbf{w}\ \boldsymbol{x}^- + b + \lambda\ \mathbf{ww} = 1$

$\Rightarrow$

$-1 + \lambda\ \mathbf{ww} = 1$

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

Computing the Margin



M = Margin = $\frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$

$$\lambda = \frac{2}{\mathbf{w}\cdot\mathbf{w}}$$

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

$$= \lambda\,|\mathbf{w}| = \lambda\sqrt{\mathbf{w}\cdot\mathbf{w}}$$

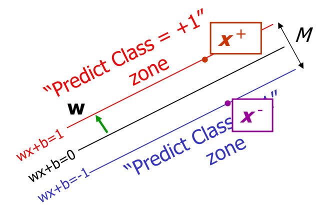
$$= \frac{2\sqrt{\mathbf{w}\cdot\mathbf{w}}}{\mathbf{w}\cdot\mathbf{w}} = \frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$$

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

What we know:

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

Learning the Maximum Margin Classifier



M = Margin = $\frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$

Given a guess of $\mathbf{w}$ and b we can

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

So now we just need to write a program to search the space of $\mathbf{w}$'s and b's to find the widest margin that matches all the data points. *How?*

SVM as Constrained Optimization

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

- Assume N training points (x_k, y_k), $y_k = 1$ or -1
- Subject to each training point on the correct side (the constraint), i.e.,

  subject to $y_k(\mathbf{w}^T x_k + b) \geq 1$ for all k

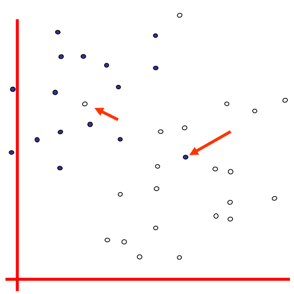
This is a **Quadratic optimization problem**, which can be solved efficiently

SVMs: More than Two Classes

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

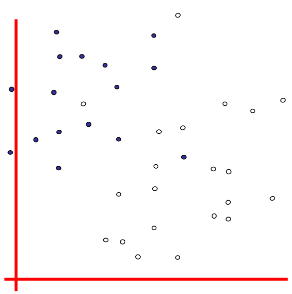
SVM: Non Linearly-Separable Data

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



SVM: Non Linearly-Separable Data

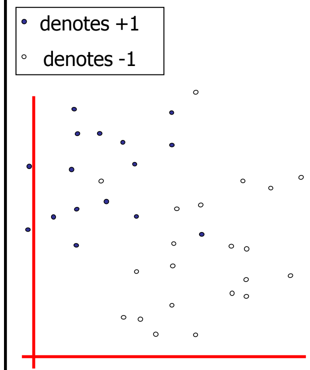
- Two solutions:
  - Allow a few points on the wrong side (**slack variables**)
  - Map data to a higher dimensional space, and do linear classification there (**kernel trick**)



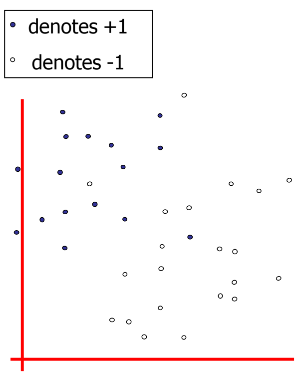
Non Linearly-Separable Data

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

What should we do?



What should we do?

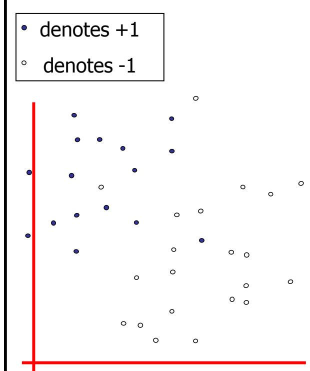


Idea 1:

Find minimum $||\boldsymbol{w}||^2$ while minimizing number of training set errors

Problem: Two things to minimize makes for an ill-defined optimization

What should we do?



Idea 1.1:

Minimize

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

Tradeoff parameter

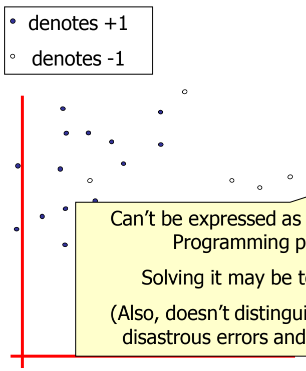
There's a serious practical problem with this approach

What should we do?

Idea 1.1:

Minimize

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



Tradeoff parameter

Can't be expressed as a Quadratic Programming problem.

Solving it may be too slow.

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

erious practical

th this

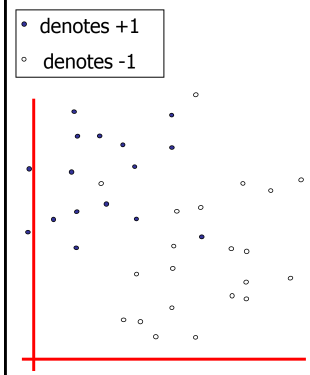
So… any other ideas?

What should we do?

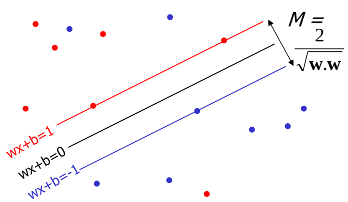
Idea 2.0:

Minimize

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



Learning Maximum Margin with Noise



Given guess of $\boldsymbol{w}$, b, we can

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

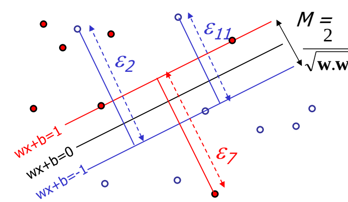
Assume N examples, each $(\boldsymbol{x}_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning Maximum Margin with Noise



"slack variables"

Given guess of $\boldsymbol{w}$, b we can

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

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

What should our quadratic optimization criterion be?

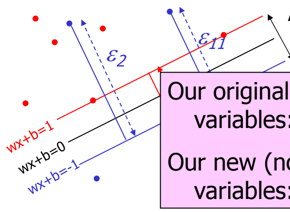
Minimize $\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{k=1}^{N}\varepsilon_k$

How many constraints will we have? N

What should they be?

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

Learning Maximum Margin with Noise



$M = \frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$

d = # input dimensions

Given gu... we can

1. Compute sum o... distances ... t

... dth

Our original (noiseless data) QP had d+1 variables: w_1, w_2, … w_d and b

Our new (noisy data) QP has d+1+N variables: w_1, w_2, … w_d, b, ε_k, ε_1,… ε_N

What should our quadratic optimization criterion be?

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

How many constraints will we have? N

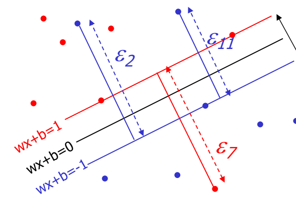
N = # examples

What should they be?

$\boldsymbol{w}\cdot\boldsymbol{x}_k + b \geq 1 - \varepsilon_k$ if $y_k = +1$

$\boldsymbol{w}\cdot\boldsymbol{x}_k + b \leq -1 + \varepsilon_k$ if $y_k = -1$

Learning Maximum Margin with Noise



$M = \frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$

Given guess of $\boldsymbol{w}$, b we can

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

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

What should our quadratic optimization criterion be?

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

How many constraints will we have? N

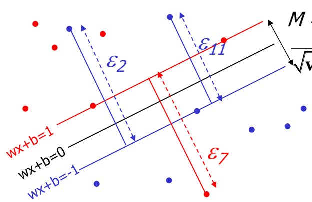
What should they be?

$\boldsymbol{w}\cdot\boldsymbol{x}_k + b \geq 1 - \varepsilon_k$ if $y_k = +1$

$\boldsymbol{w}\cdot\boldsymbol{x}_k + b \leq -1 + \varepsilon_k$ if $y_k = -1$

There's a bug in this QP. Can you spot it?

Learning Maximum Margin with Noise



$M = \frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$

Given guess of $\boldsymbol{w}$, b we can

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

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

What should our quadratic optimization criterion be?

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

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

What should they be?

$\boldsymbol{w}\cdot\boldsymbol{x}_k + b \geq +1 - \varepsilon_k$ if $y_k = +1$

$\boldsymbol{w}\cdot\boldsymbol{x}_k + b \leq -1 + \varepsilon_k$ if $y_k = -1$

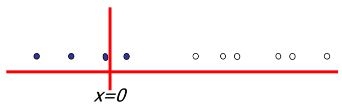
$\varepsilon_k \geq 0$ for all k

Non Linearly-Separable Data

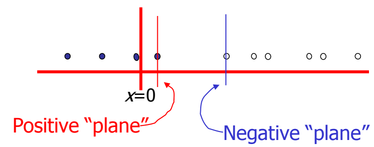
- Approach 2: Map data to a higher dimensional space, and do linear classification there (**kernel trick**)

Suppose we're in 1 Dimension

What would SVMs do with this data?

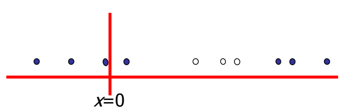


Suppose we're in 1 Dimension

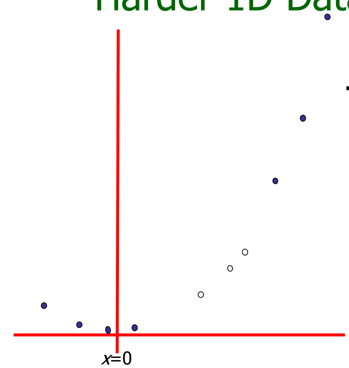


Harder 1D Dataset: Not Linearly-Separable

What can be done about this?



Harder 1D Dataset



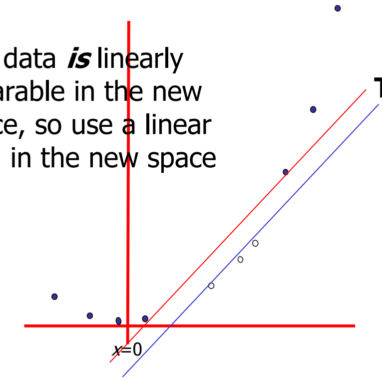
The Kernel Trick: Preprocess the data, mapping ***x*** into a higher dimensional space, $\Phi(\boldsymbol{x})$

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

Here, Φ maps data from 1D to 2D

Harder 1D Dataset

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



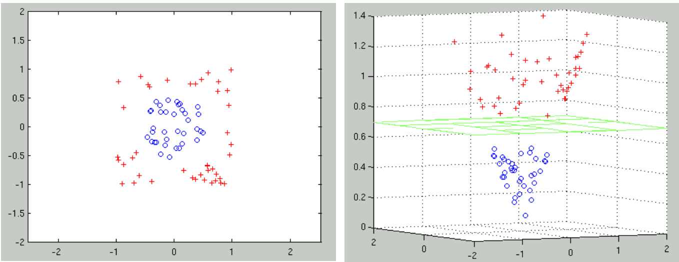
$\mathbf{w}^{\mathrm{T}}\,\Phi(\boldsymbol{x}) + b = +1$

The Kernel Trick: Preprocess the data, mapping $\boldsymbol{x}$ into a higher dimensional space, $\Phi(\boldsymbol{x})$

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

Another Example

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



- Project examples into some higher dimensional space where the data ***is*** linearly separable, defined by $\boldsymbol{z} = \Phi(\boldsymbol{x})$
- Can formulate optimization problem so that objective function depends *only* on dot products of the form

  $\Phi(\boldsymbol{x}_i)^{\mathrm{T}} \cdot \Phi(\boldsymbol{x}_j)$ where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are two data points
- Example:

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

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

- Claim: Can compute kernel function K *without* explicitly computing $\Phi(\boldsymbol{x})$ or $\mathbf{w}$
- Dimensionality of $\boldsymbol{z}$ space is generally *much larger* than the dimensionality of input space $\boldsymbol{x}$

What's Special about a Kernel?

- Say data is 2D: $\boldsymbol{s} = (s_1, s_2)$
- We decide to use a particular mapping into 6D space:

  $\Phi(\boldsymbol{s}) = (s_1^2, s_2^2, \sqrt{2}\, s_1 s_2, s_1, s_2, 1)$
- Let another point be $\boldsymbol{t} = (t_1, t_2)$
- Then,

  $\Phi(\boldsymbol{s})^{\mathrm{T}} \cdot \Phi(\boldsymbol{t}) = s_1^2\, t_1^2 + s_2^2\, t_2^2 + 2 s_1 s_2 t_1 t_2 + s_1 t_1 + s_2 t_2 + 1$
- Let the **kernel** be $K(\boldsymbol{s}, \boldsymbol{t}) = (\boldsymbol{s}^{\mathrm{T}} \cdot \boldsymbol{t} + 1)^2 = (s_1 t_1 + s_2 t_2 + 1)^2$
- $K(\boldsymbol{s}, \boldsymbol{t}) = \Phi(\boldsymbol{s})^{\mathrm{T}} \cdot \Phi(\boldsymbol{t})$
- We save computation by using K

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

- Many other kernels
- Hacking with SVMs: create various kernels, hope their space Φ is meaningful, plug them into SVM, pick one with good classification accuracy
- Kernel can be combined with slack variables

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

Description of the Data

- For every paper:
  - Machine readable text was created using a scanner
  - 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
- The dataset consists of 118 papers:
  - 50 Madison papers
  - 56 Hamilton papers
  - 12 disputed papers

Function Words Based on Relative Frequencies

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*

Feature Selection for Classifying the Disputed Federalist Papers

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

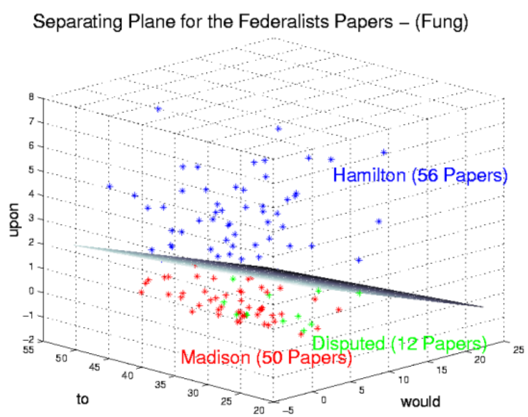
Hyperplane Classifier Using 3 Words

- A hyperplane depending on three words was found:

$$0.537\,to + 24.663\,upon + 2.953\,would = 66.616$$

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

Results: 3D Plot of Hyperplane



SVM Applet

http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

Summary

- Learning linear functions
  - Pick separating plane that maximizes margin
  - Separating plane defined in terms of support vectors only
- Learning non-linear functions
  - Project examples into 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