Support Vector Machine

Chenghua Lin
Chenghua.Lin@abdn.ac.uk

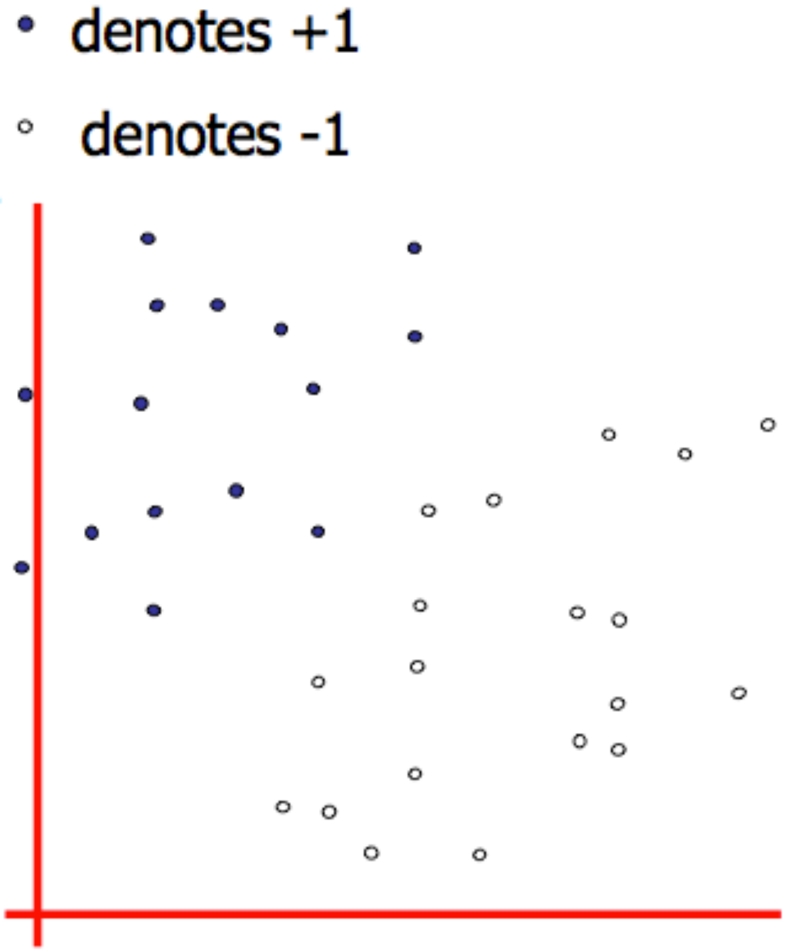
Outline

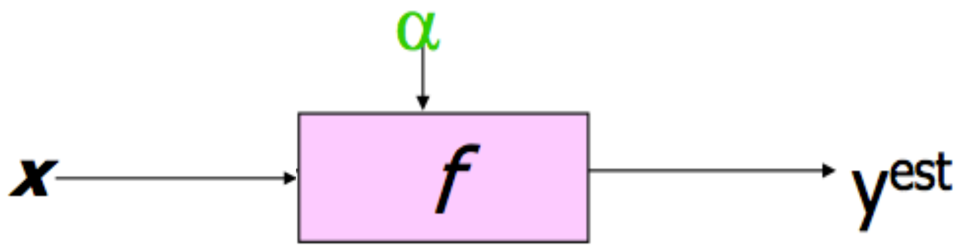
- Probabilistic and Bayesian Analytics
- Classification
  - Naïve Bayes Classifier
  - Support vector machines (SVM)
  - Decision Trees
- Association Rule Mining
- Feature Selection
- Visualization I and II
- Case study
- Data Mining Issues

What You Should Know

- Linear SVMs
- The definition of a maximum margin classifier
- What QP (Quadratic Programming) can do for you
  - for this class, you don't need to know how it does it
- How we deal with noisy data, i.e. misclassified data
- How we permit non-linear boundaries
- How SVM Kernel functions permit us to pretend we're working with ultra-high-dimensional basis- function terms

Linear Classifiers

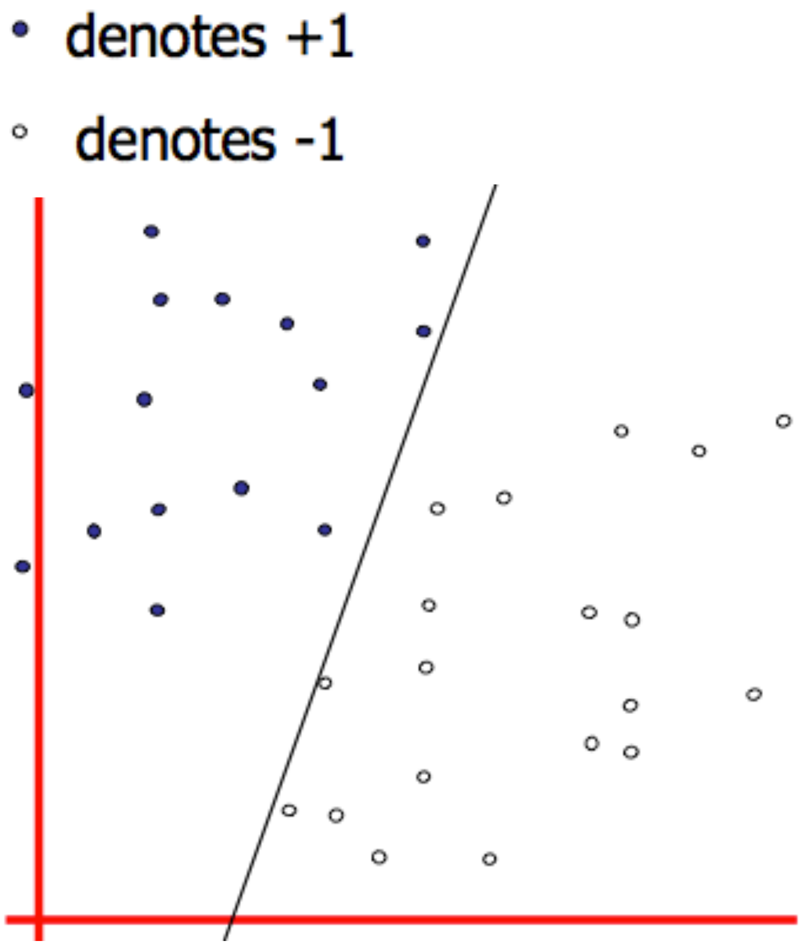


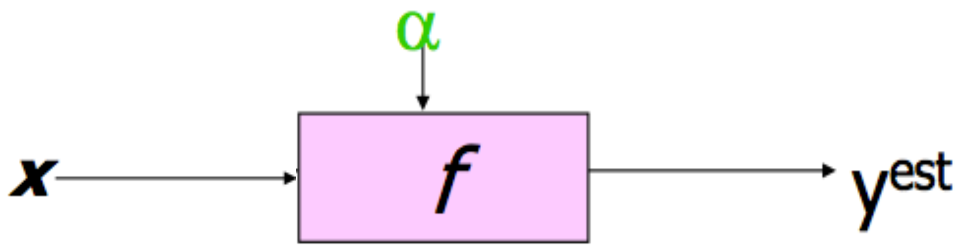


$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

How would you classify this data?

Linear Classifiers

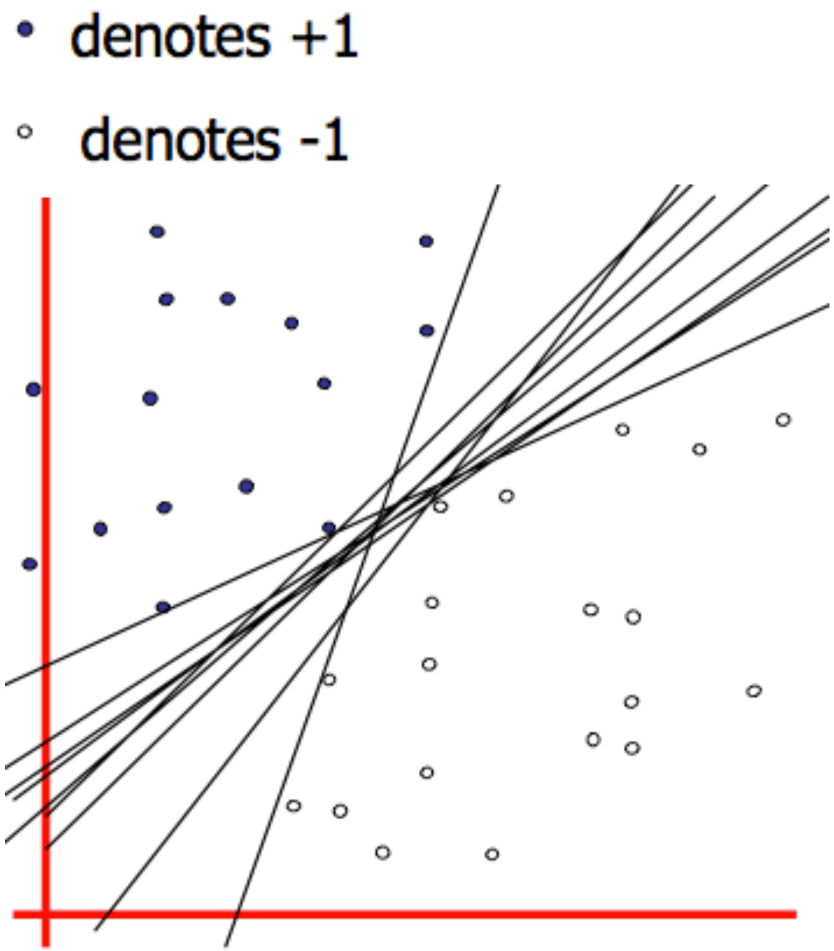


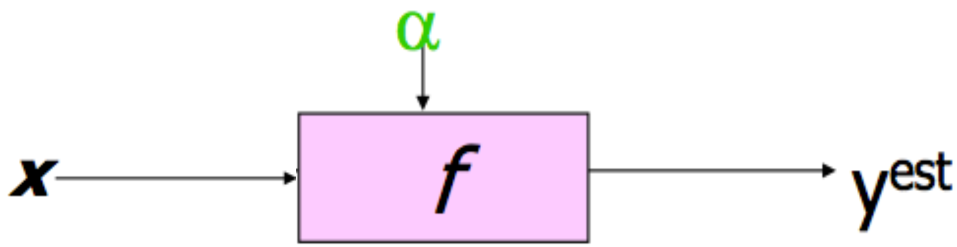


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

How would you classify this data?

Linear Classifiers



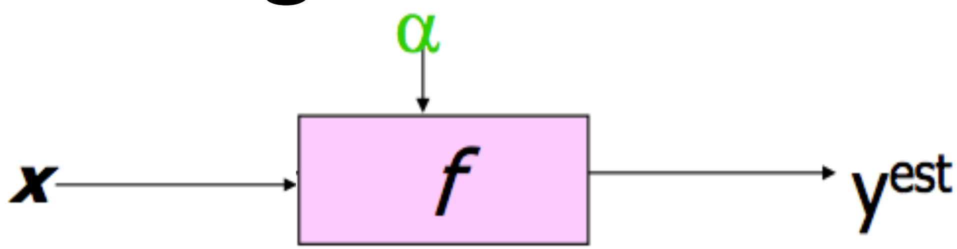


$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

Any of these would be fine …

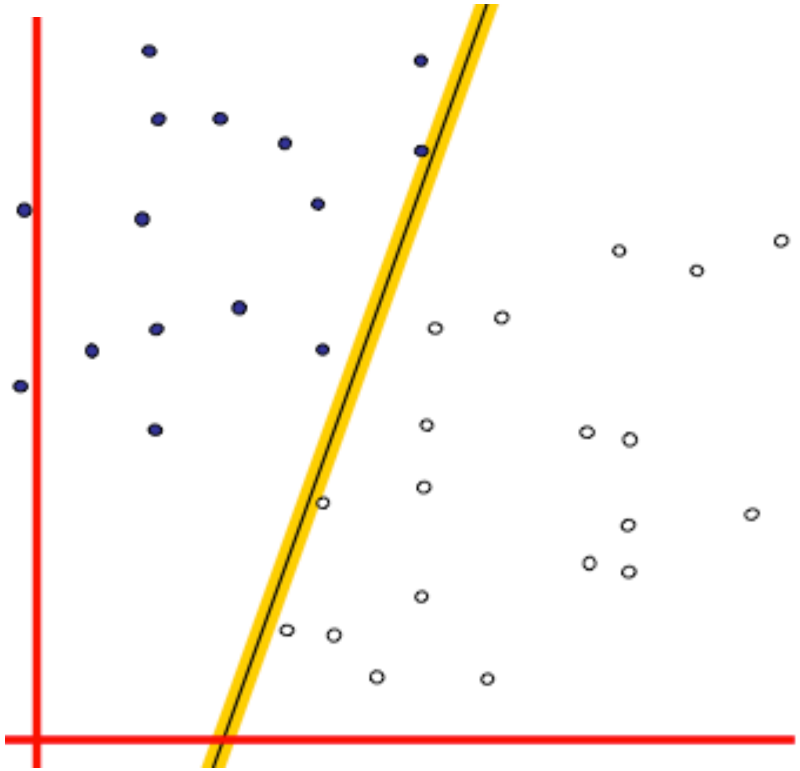
… but which is best?

Classifier Margin



• denotes +1

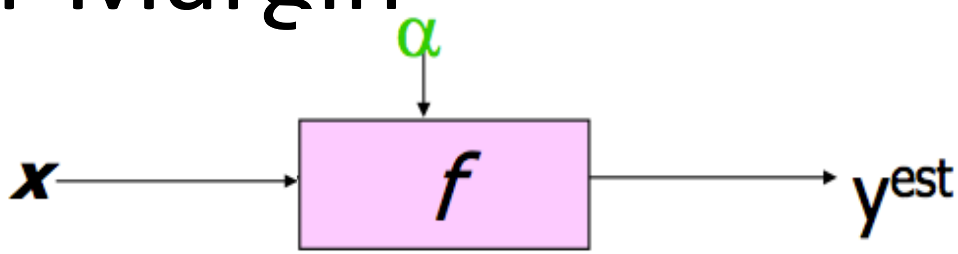
◦ denotes -1



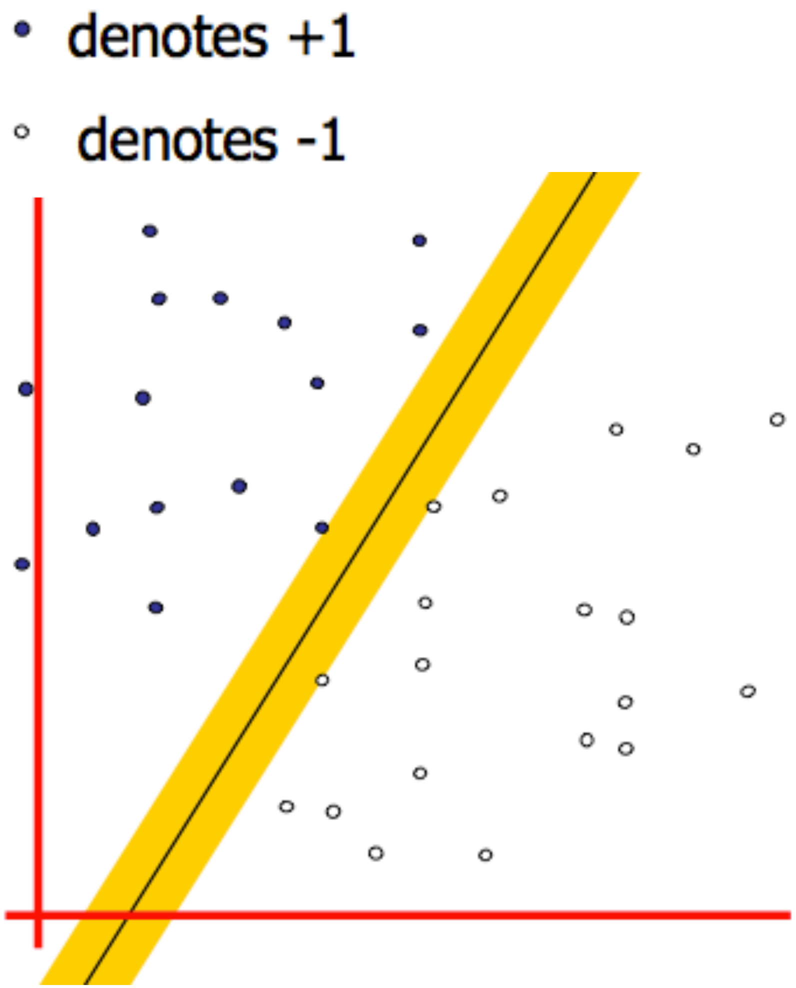
$$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w}\cdot\mathbf{x} - b)$$

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Classifier Margin



$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

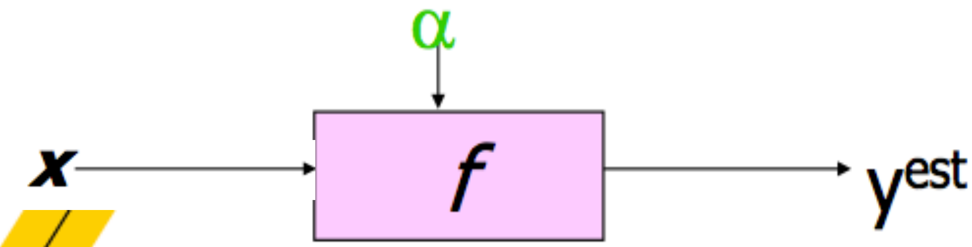


The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM, i.e. Linear SVM)

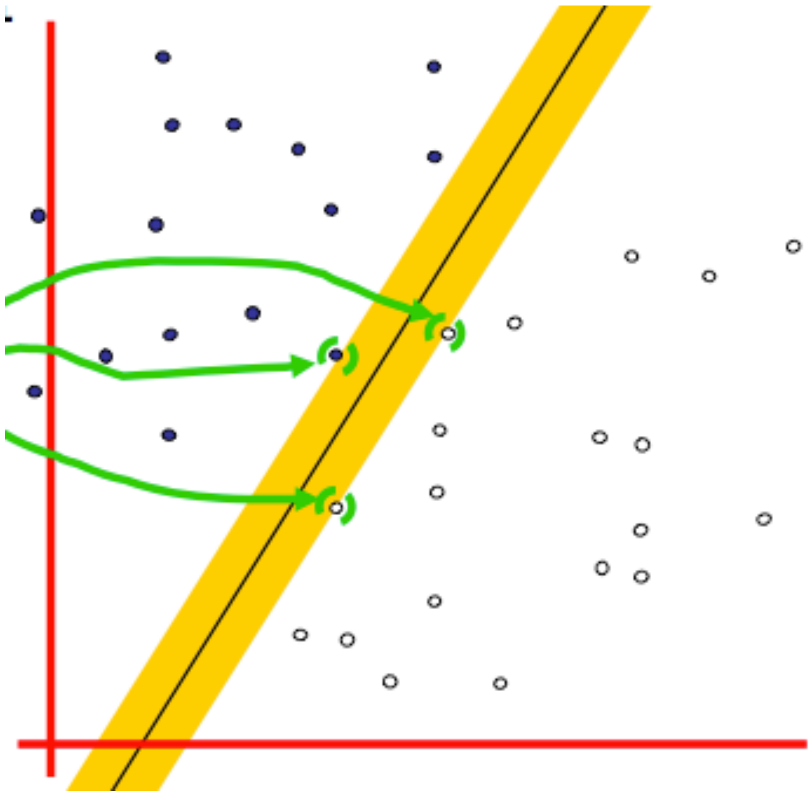
Maximum Margin

- denotes +1
- ∘ denotes -1



$$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w}\cdot\mathbf{x} - b)$$

Support Vectors are those datapoints that the margin pushes up against



The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

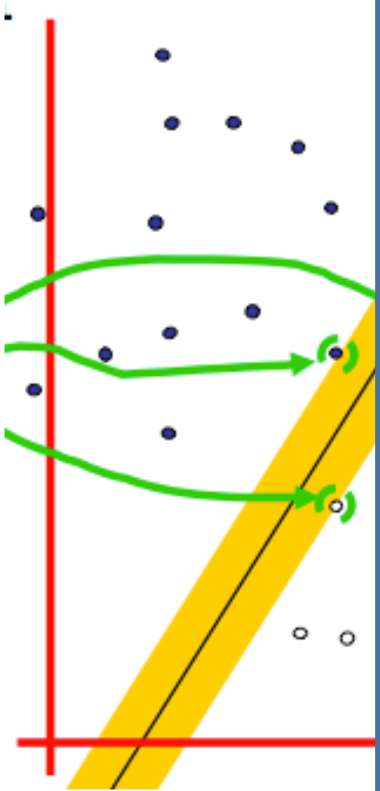
This is the simplest kind of SVM (Called an LSVM, i.e. Linear SVM)

Why Maximum Margin

• denotes +1

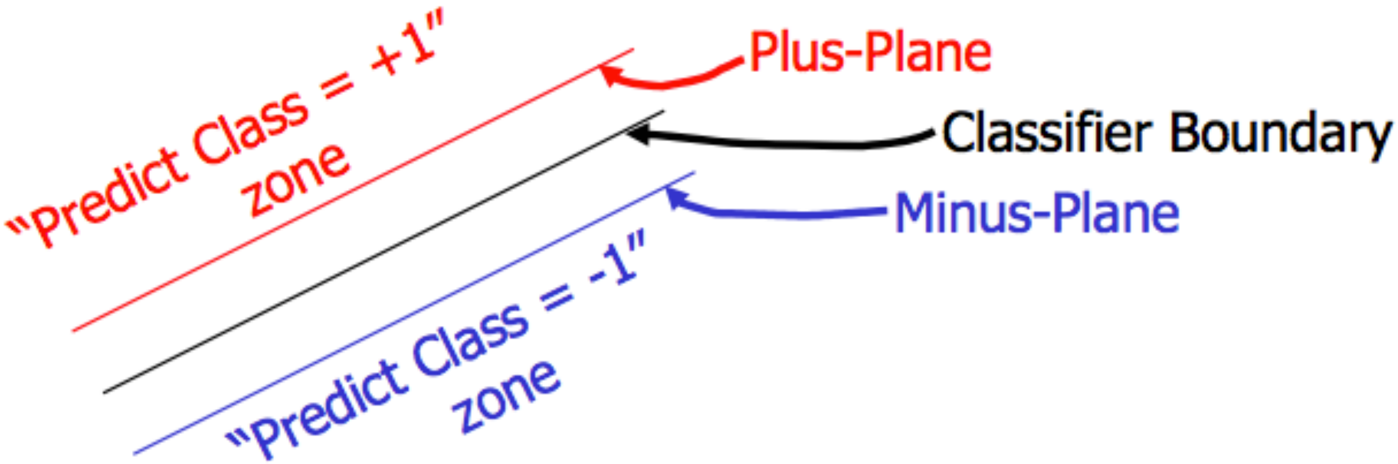
◦ denotes -1

Support Vectors are those datapoints 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. LOOCV is easy since the model is immune to removal of any non-support vector datapoints.
4. Empirically it works very very well.

Specifying a line and margin



- Plus-plane = {**x**:**w**.**x**+b=+1}
- Minus-plane= {**x**:**w**.**x**+b=-1}

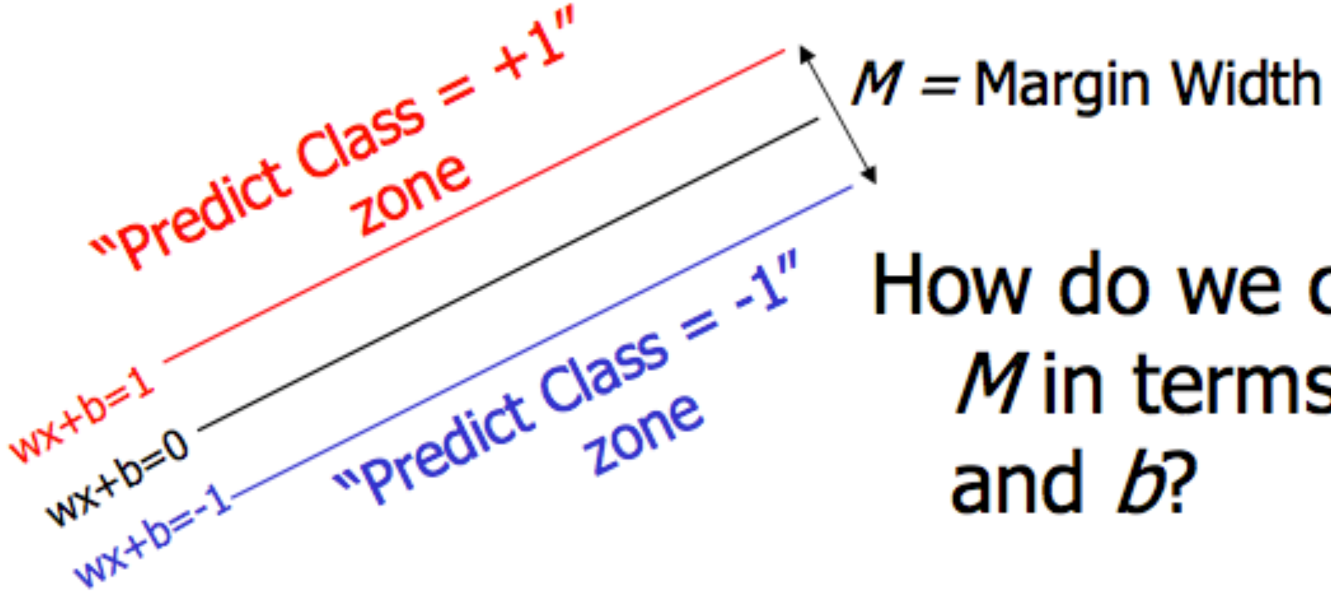
Classify as..

+1 if **w**.**x**+b>=1

-1 if **w**.**x**+b<=-1

Universe Explodes if -1 < **w** . **x** + b < 1

Compute the margin width

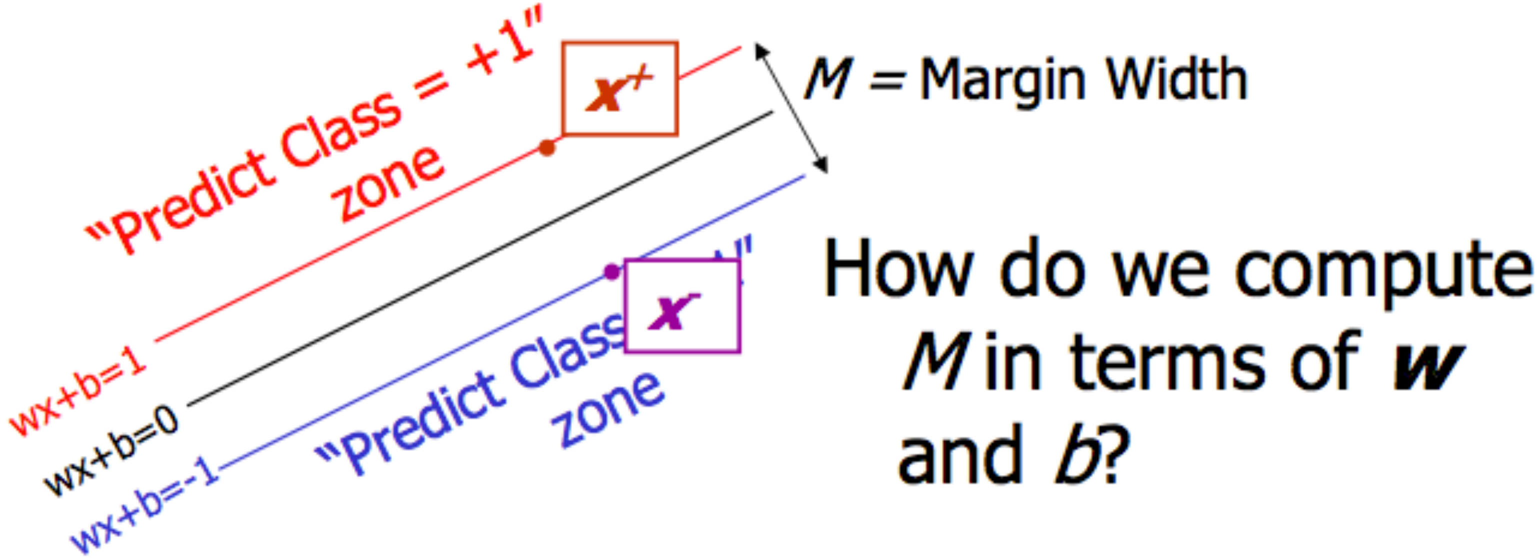


How do we compute *M* in terms of ***w*** and *b*?

- Plus-plane = {**x**:**w**.**x**+b=+1}
- Minus-plane= {**x**:**w**.**x**+b=-1}

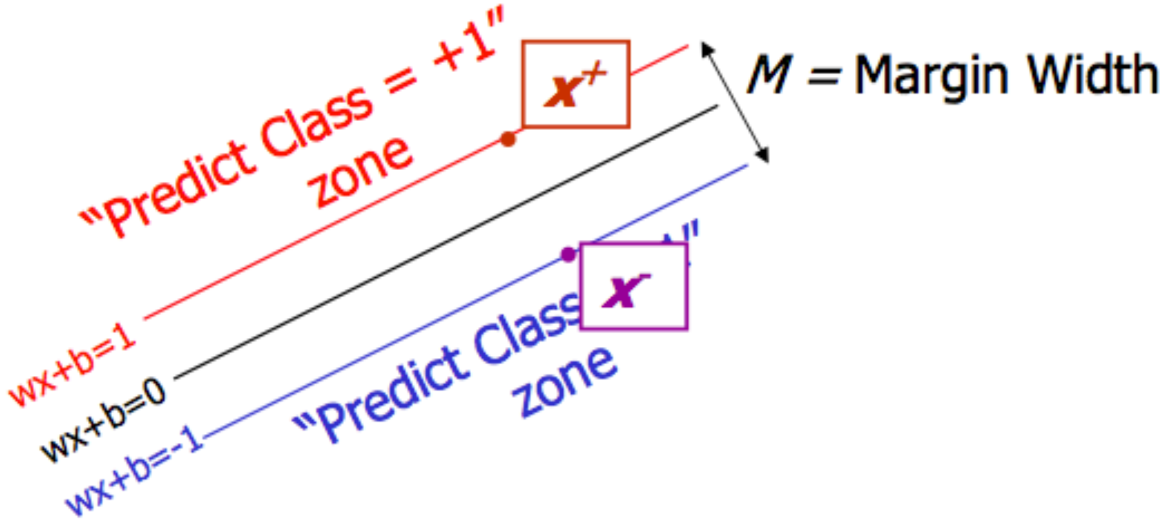
Claim: The vector **w** is perpendicular to the Plus Plane. Why?

Computing the margin width



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

Computing the margin width

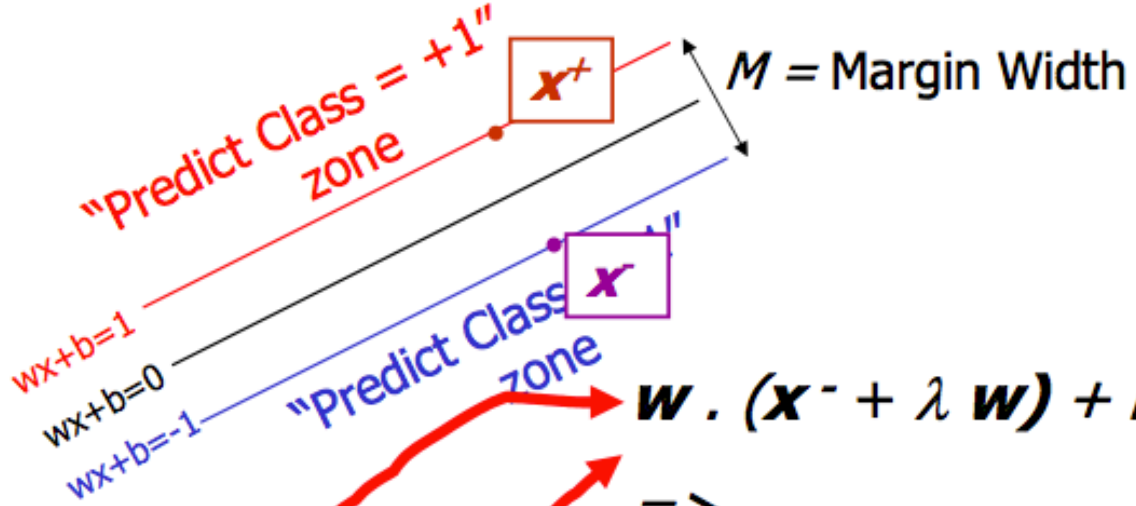


What we know:

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

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

Computing the margin width



What we know:

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

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

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

=>

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

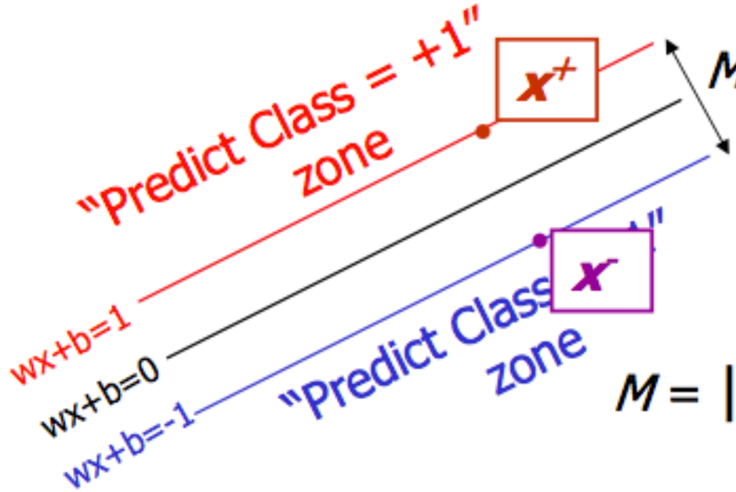
=>

$$-1 + \lambda \, \mathbf{w} \cdot \mathbf{w} = 1$$

=>

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

Computing the margin width



$$M = \text{Margin Width} = \frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$$

What we know:

- $\boldsymbol{w} \,.\, \boldsymbol{x}^{+} + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^{-} + b = -1$
- $\boldsymbol{x}^{+} = \boldsymbol{x}^{-} + \lambda\, \boldsymbol{w}$
- $|\boldsymbol{x}^{+} - \boldsymbol{x}^{-}| = M$
- $$\lambda = \frac{2}{\mathbf{w}.\mathbf{w}}$$

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

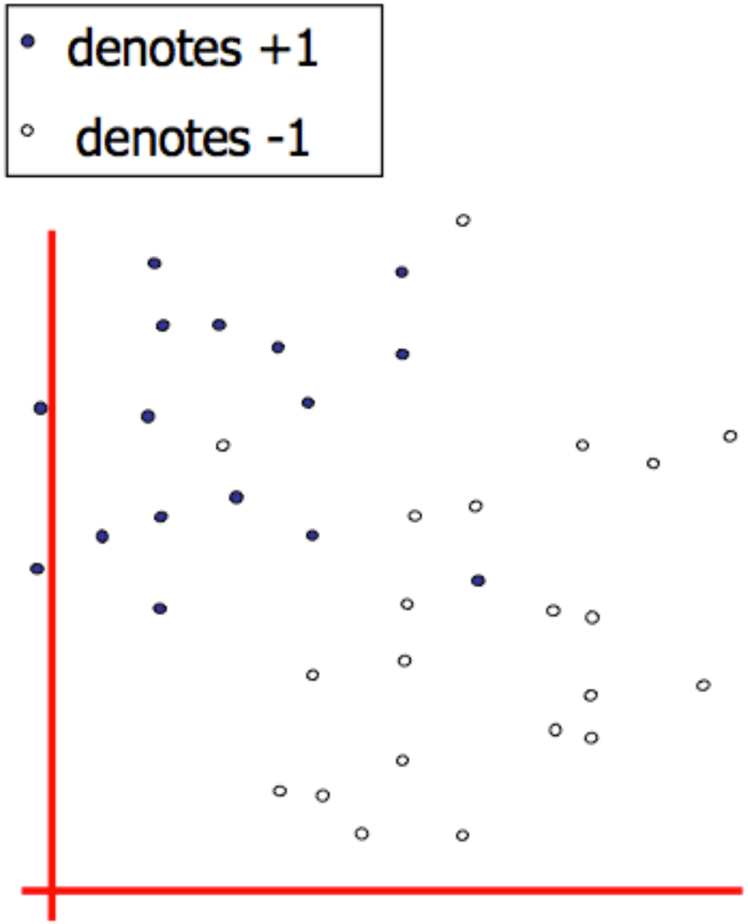
$$= \lambda\, |\mathbf{w}| = \lambda \sqrt{\mathbf{w}.\mathbf{w}}$$

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

Learning the Maximum Margin Classifier

- Given a guess of **w** and b we can
  - Compute whether all data points in the correct half-planes
  - Compute the width of 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 matches all the datapoints.
- How? Learning via Quadratic Programming
  - Out of the scope of the course

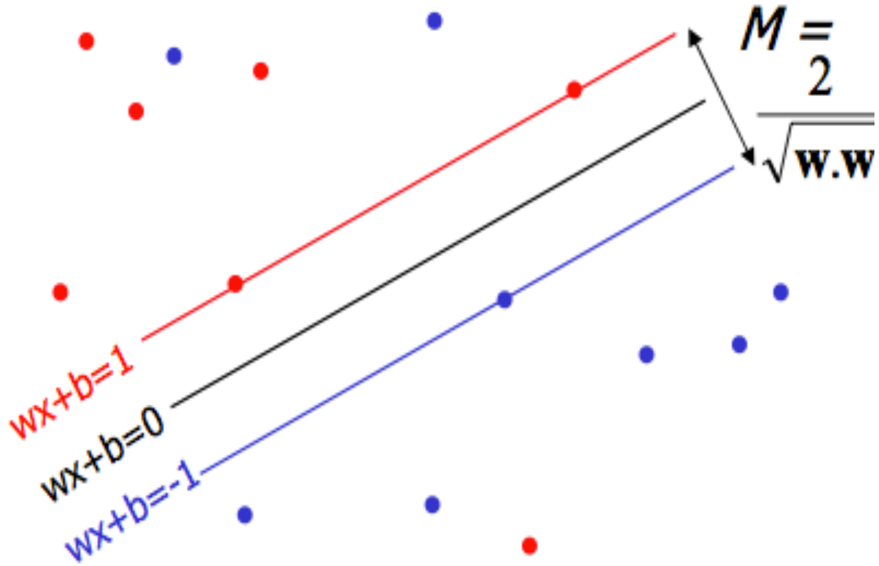
Noise, Uh-oh!



This is going to be a problem!
What should we do?

Minimize: **w.w** + D
where D is the distance of error points to their correct place

Learning Maximum Margin with Noise

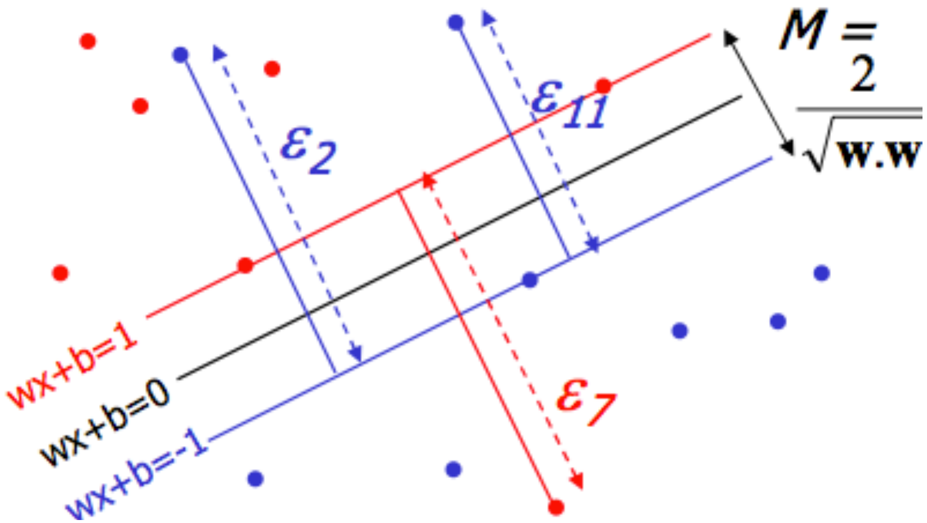


Given guess of W, b, we can

- Compute sum of distances of points to their correct zones
- Compute the margin width. Assume R datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1

- How many constraints will we have?

Learning Maximum Margin with Noise



What should our quadratic optimization criterion be?

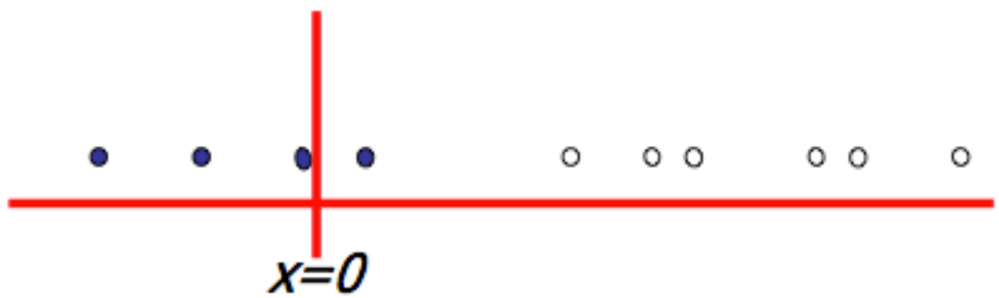
Minimize $\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$

Given guess of W, b can

- Compute sum of distances of points to their correct zones
- Compute the margin width. Assume R datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1
- ε: the slack variable
- C: control the trade-off between the slack variable and the margin.

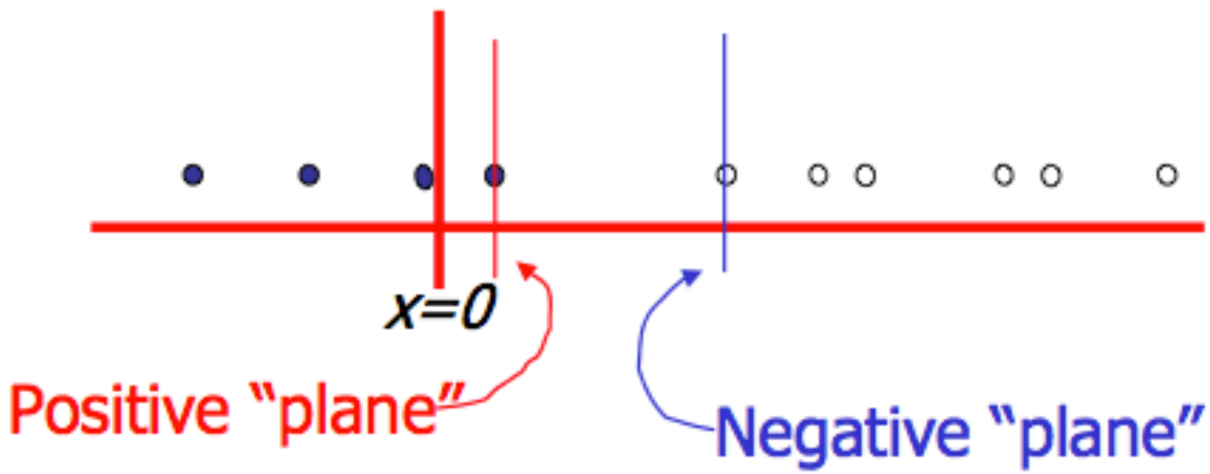
Suppose we’re in 1-dimension

What would SVMs do with this data?



Suppose we're in 1-dimension

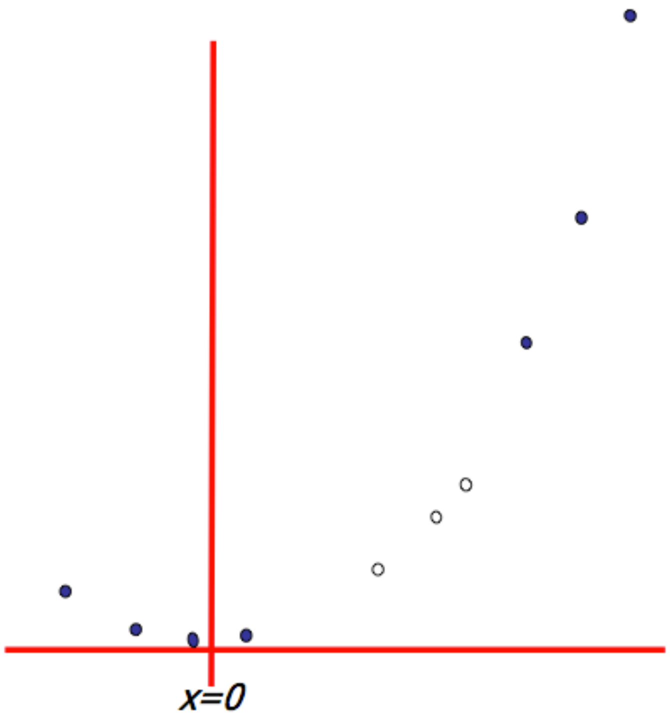
Not a big surprise



Harder 1-dimensional dataset

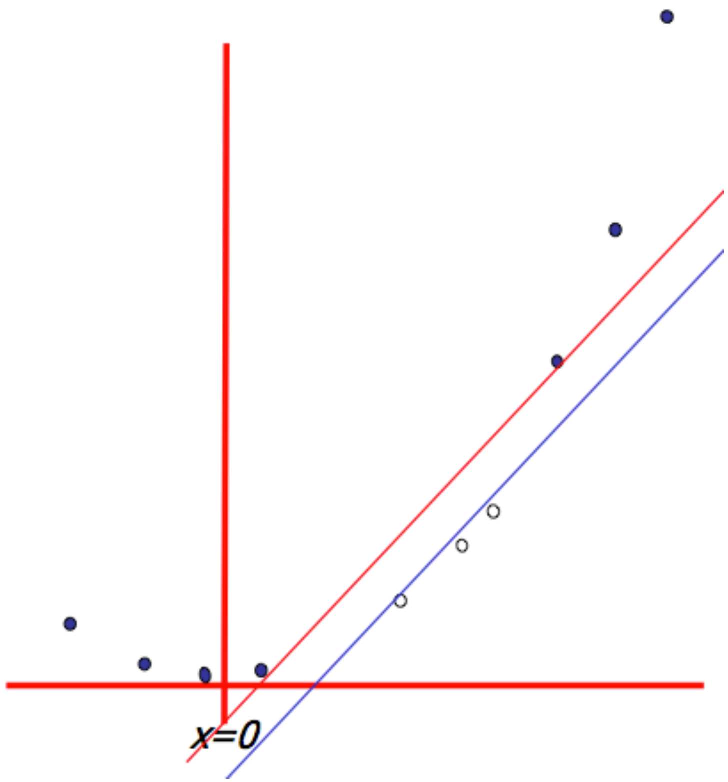
What can be done about this? i.e. linear not separable

Harder 1-dimensional dataset



- Non-linear basis functions to rescue!
- To project original datapoints to higher dimensions within which datapoints are separable
- $\mathbf{z}_k = (x_k, x_k^2)$

Harder 1-dimensional dataset



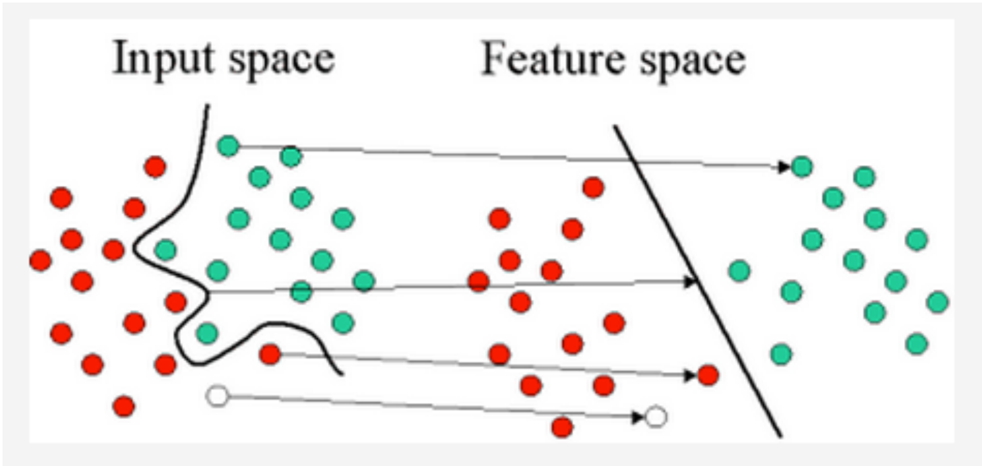
- Non-linear basis functions to rescue!
- To project original datapoints to higher dimensions within which datapoints are separable
- $\mathbf{z}_k = (x_k, x_k^2)$

Common SVM Kernel functions

Kernel trick: SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Kernel functions:

- polynomial functions
- radial basis functions
- sigmoid functions



Summary

- The definition of a maximum margin classifier
- How Maximum Margin can be turned into a QP problem
- How we deal with noisy data, i.e. misclassified data
  - slack variable
- How we permit non-linear boundaries
  - SVM Kernel functions permit us to pretend we're working with ultra-high-dimensional basis-function terms
  - And in the new feature space, datapoints are linearly separable

Readings

- An excellent tutorial on VC-dimension and Support Vector Machines:
  – C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998. http://citeseer .nj.nec.com/burges98tutorial.html
- The VC/SRM/SVM Bible:
  – Statistical Learning Theory by Vladimir Vapnik, Wiley- Interscience; 1998