Support Vector Machines (SVM)

History of SVM

- SVM is related to statistical learning theory [3]
- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details
- SVM is now regarded as an important example of “kernel methods”, one of the key area in machine learning
  - Note: the meaning of “kernel” is different from the “kernel” function for Parzen windows

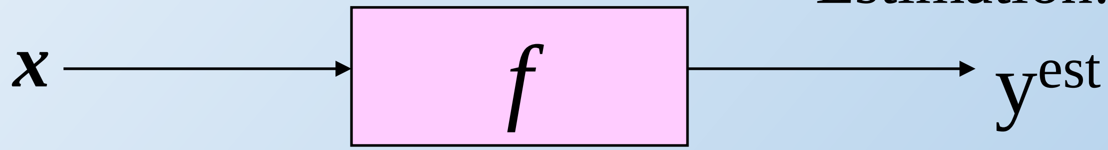
[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.

[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.

[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

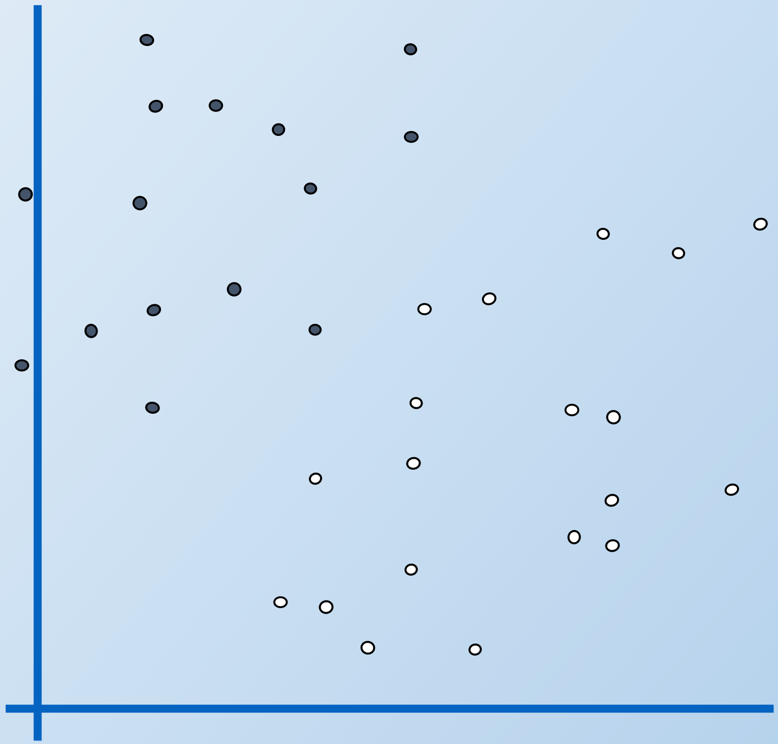
Linear Classifiers

Estimation:



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

- denotes +1
- denotes -1

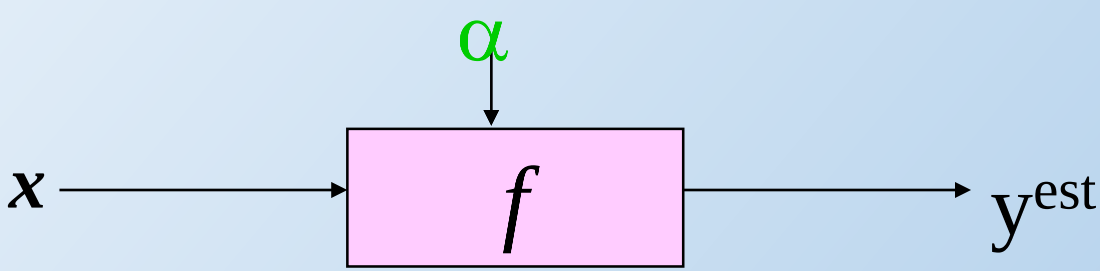


w: weight vector

x: data vector

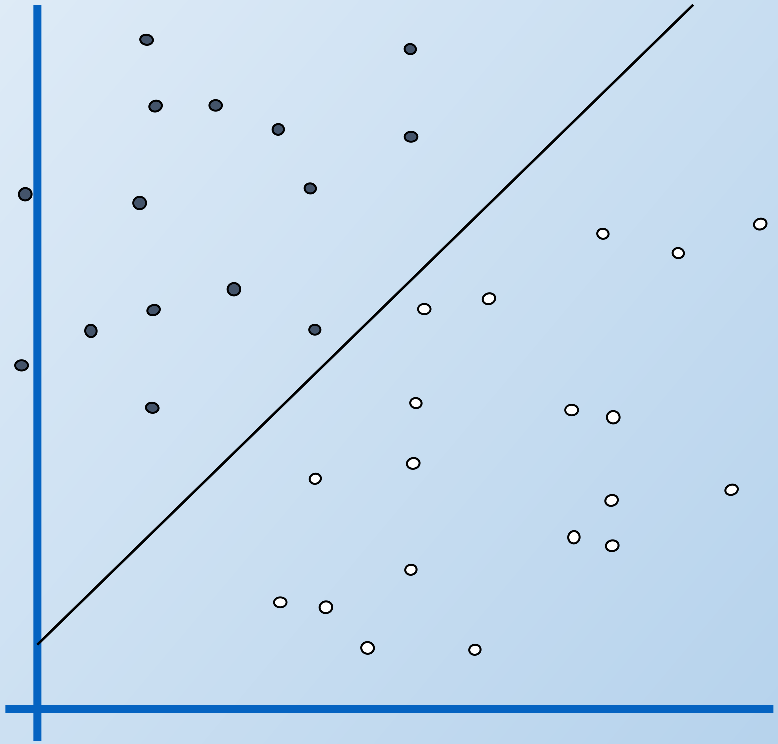
How would you classify this data?

Linear Classifiers



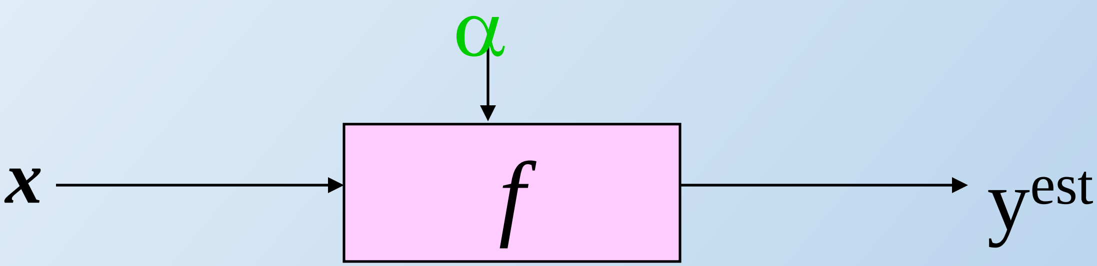
$f(x,w,b) = sign(w.\ x - b)$

- denotes +1
- denotes -1

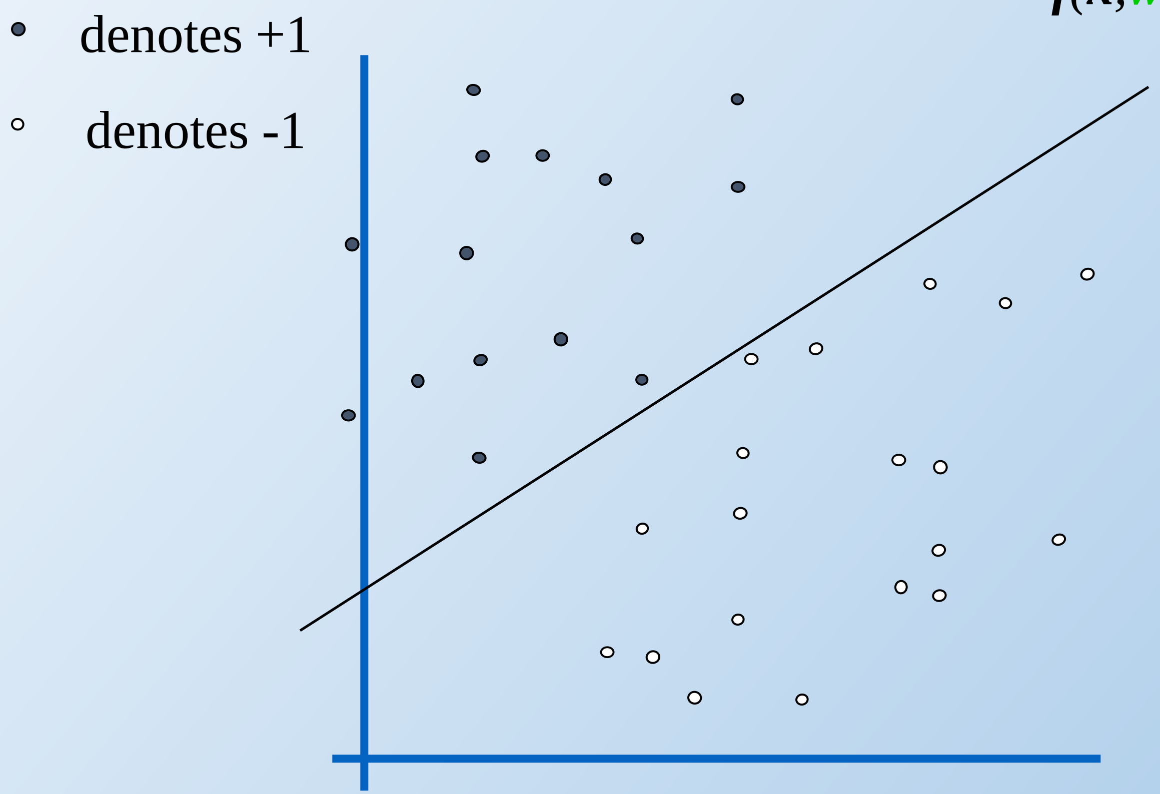


How would you classify this data?

Linear Classifiers

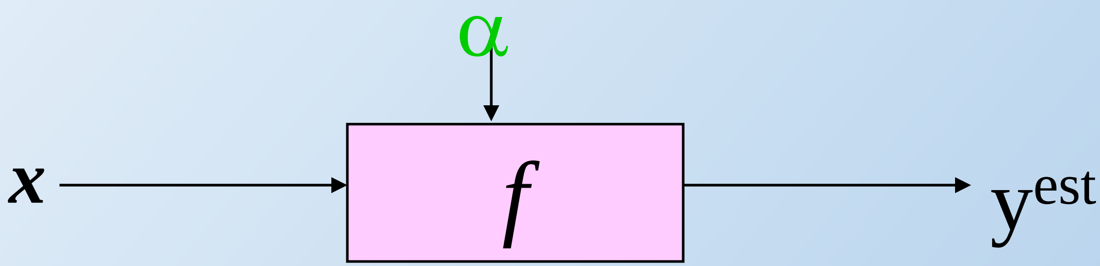


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



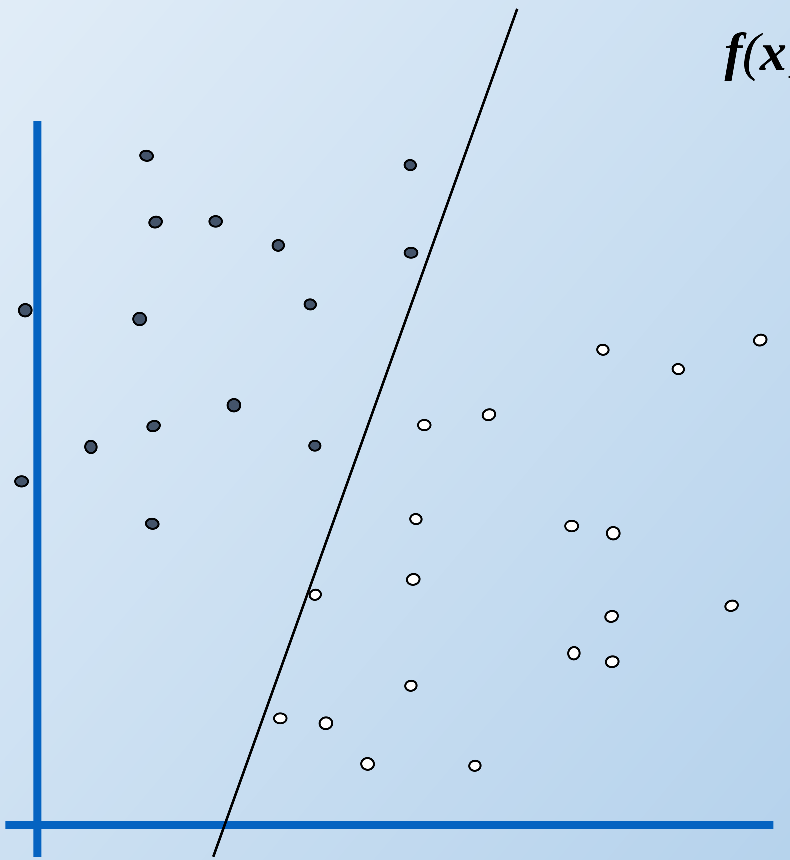
How would you classify this data?

Linear Classifiers



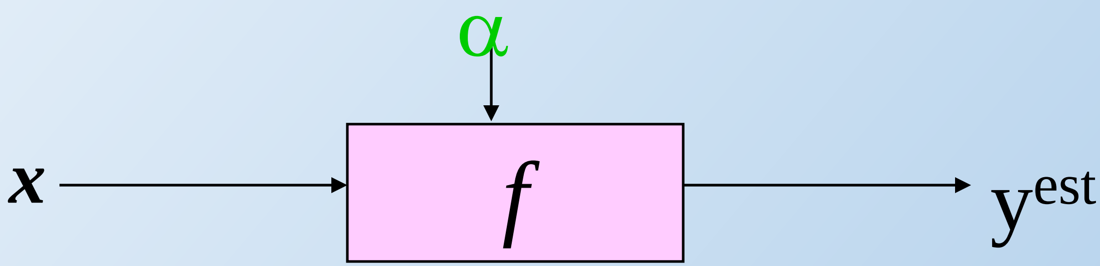
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- ● denotes +1
- ○ denotes -1



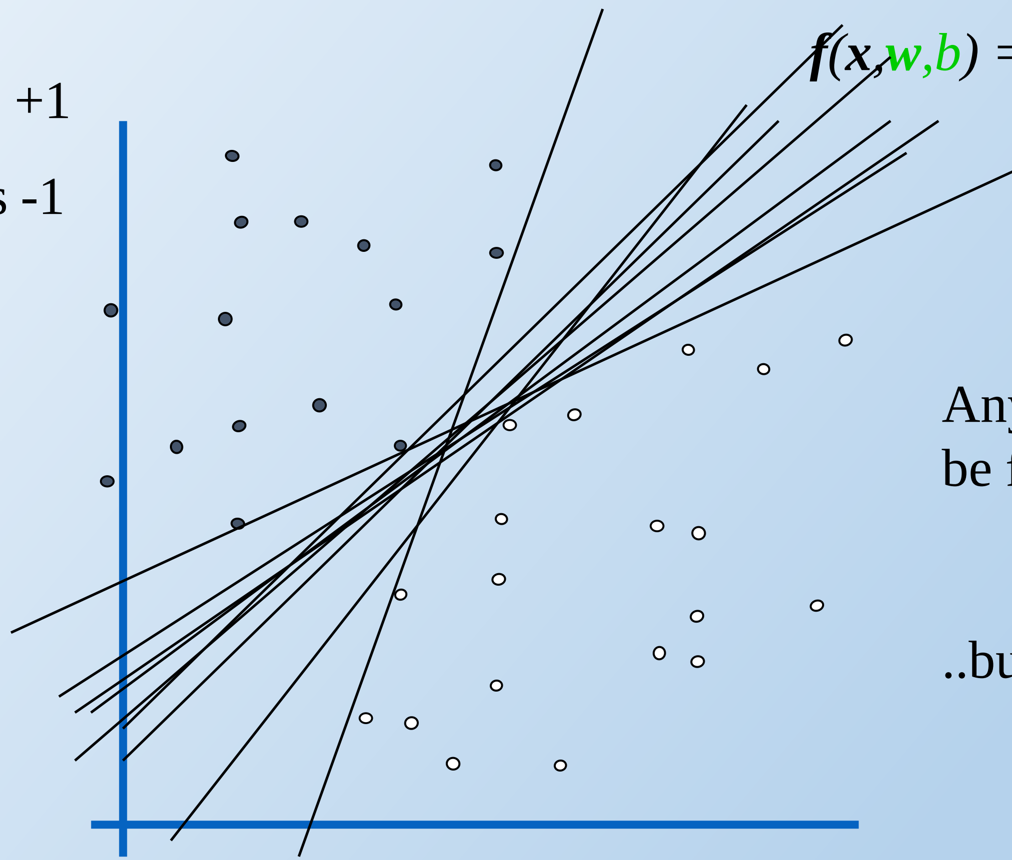
How would you classify this data?

Linear Classifiers



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

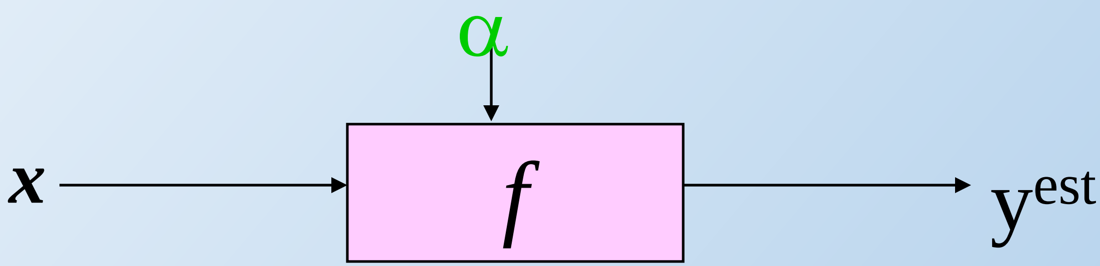
- ● denotes +1
- ○ denotes -1



Any of these would be fine..

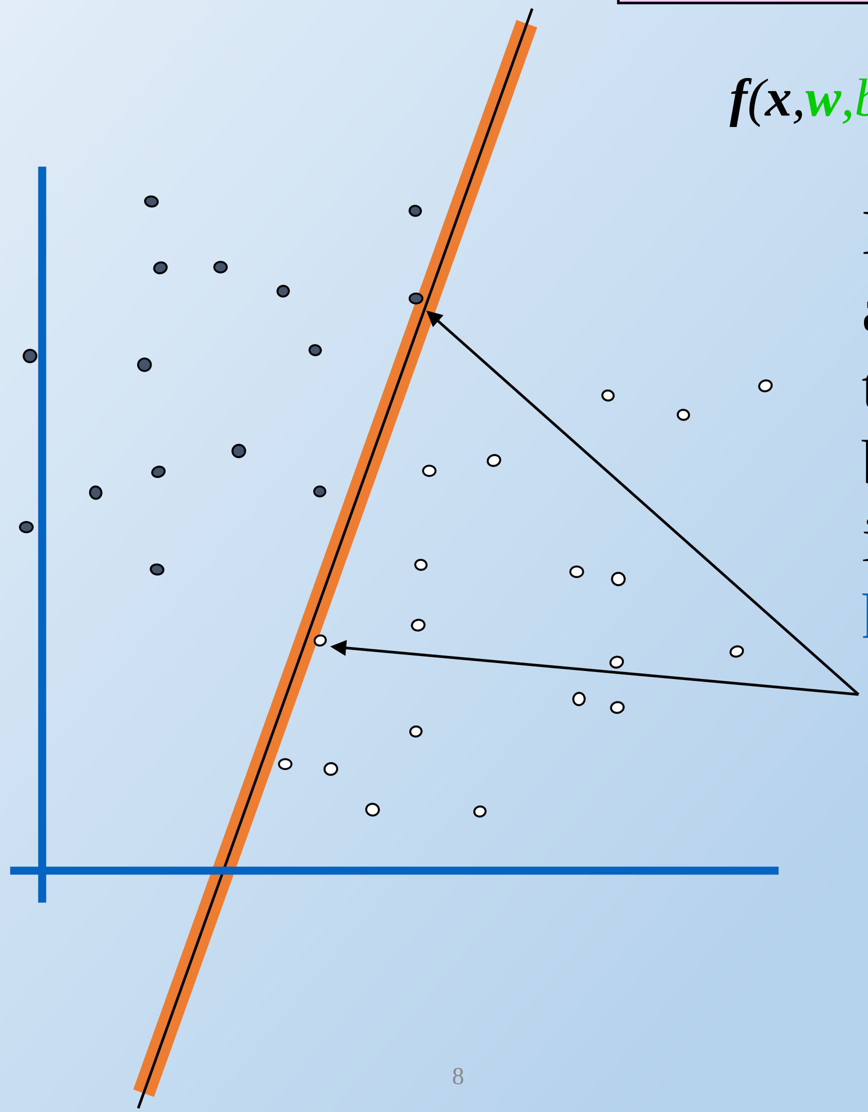
..but which is best?

Classifier Margin



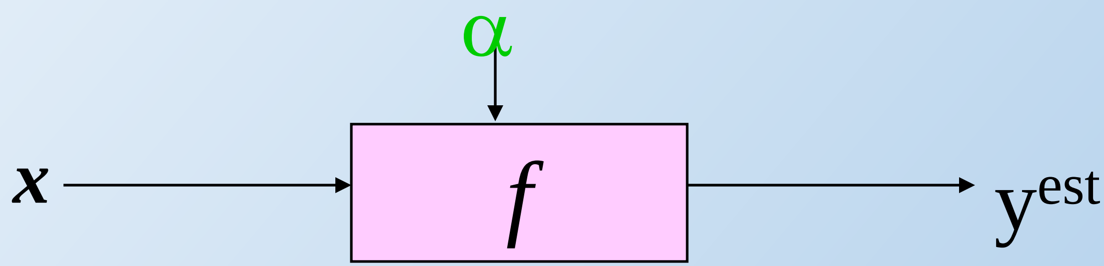
$$f(\boldsymbol{x}, \boldsymbol{w}, b) = sign(\boldsymbol{w}. \boldsymbol{x} - b)$$

- denotes +1
- denotes -1

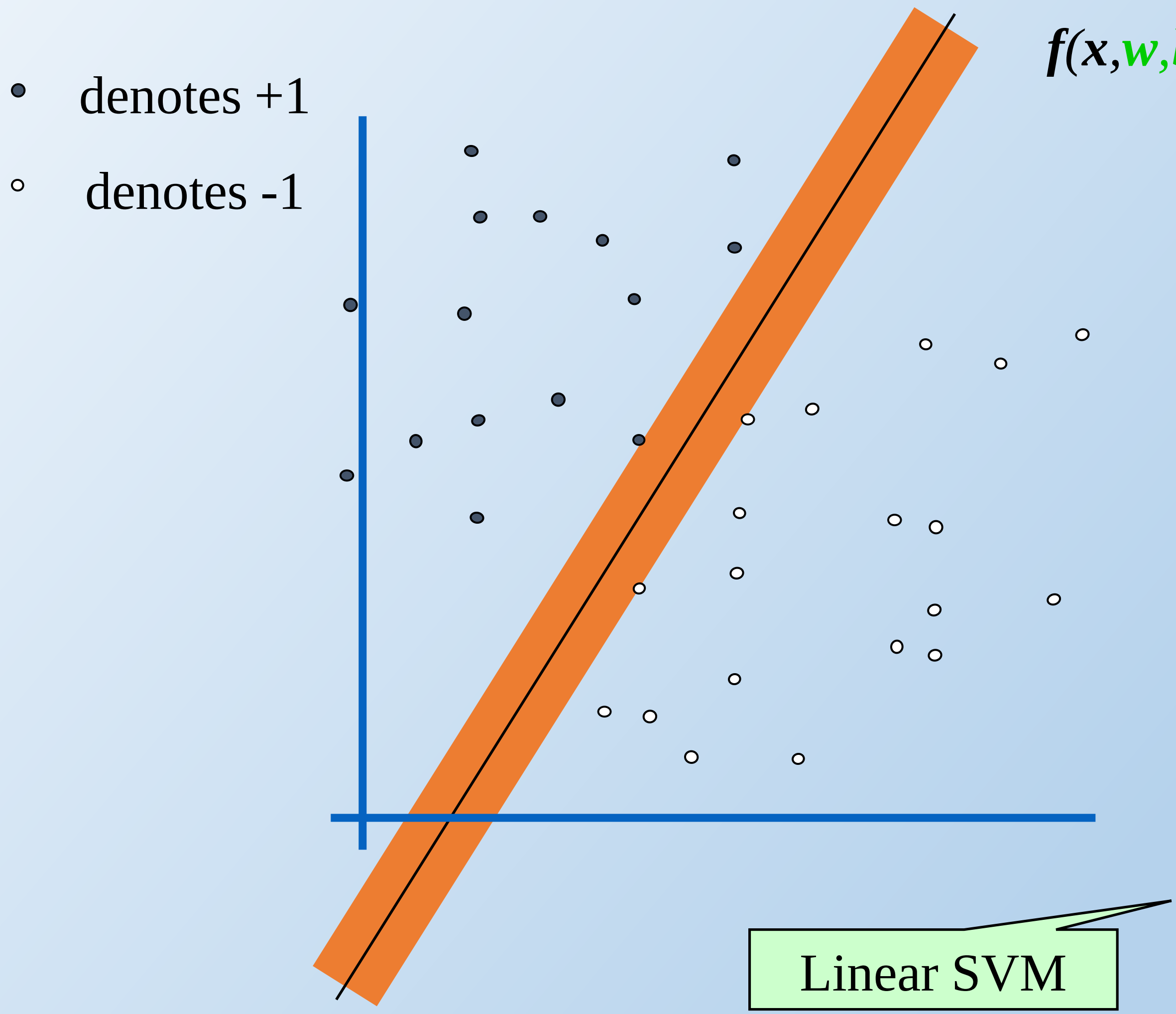


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



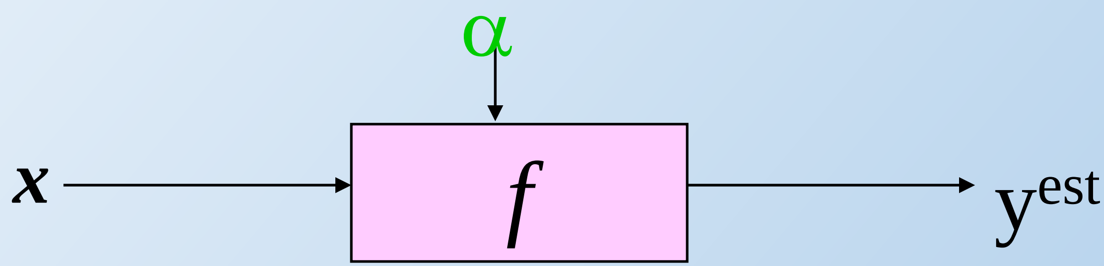
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$



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

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

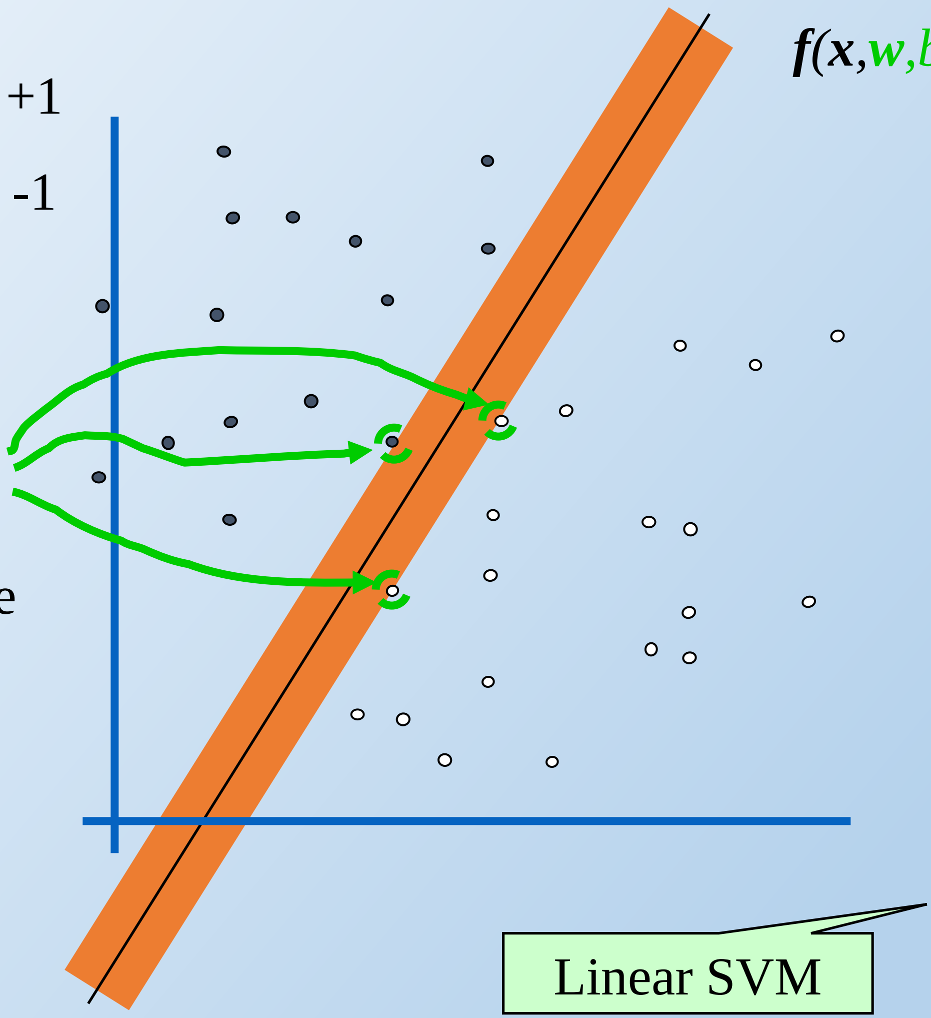
Maximum Margin



$f(x,w,b) = sign(w.\ x + b)$

- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against



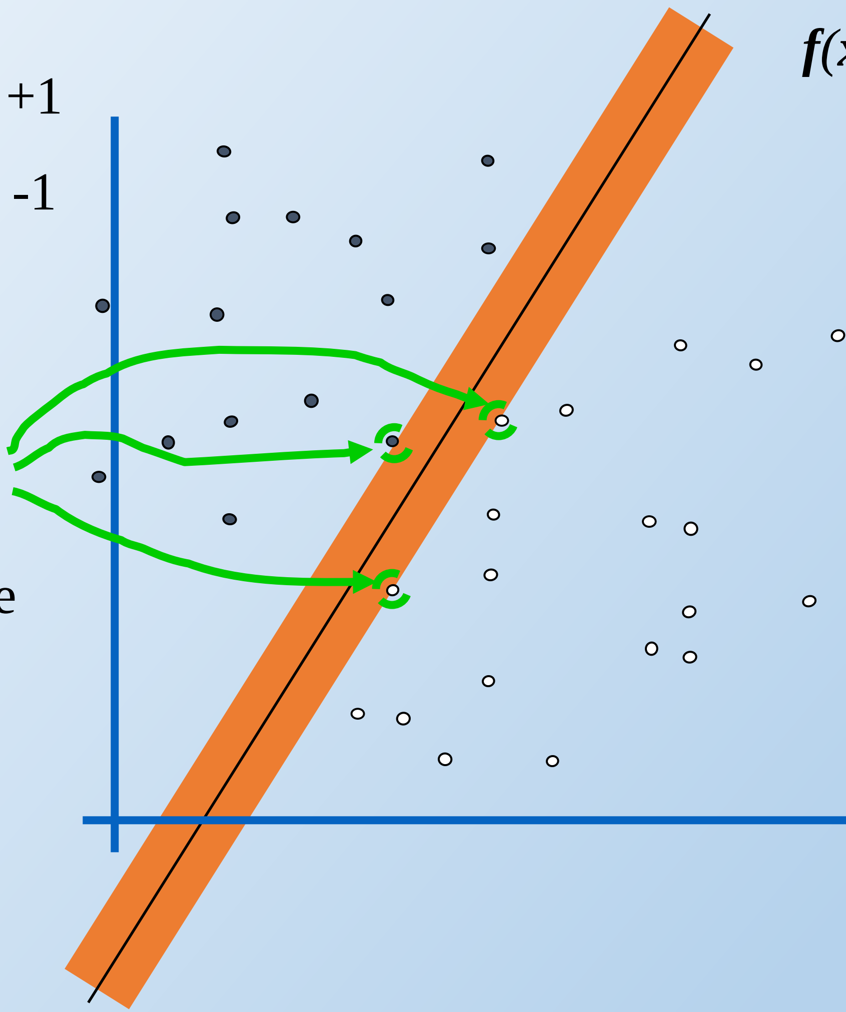
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

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

Why Maximum Margin?

- denotes +1
- denotes -1

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

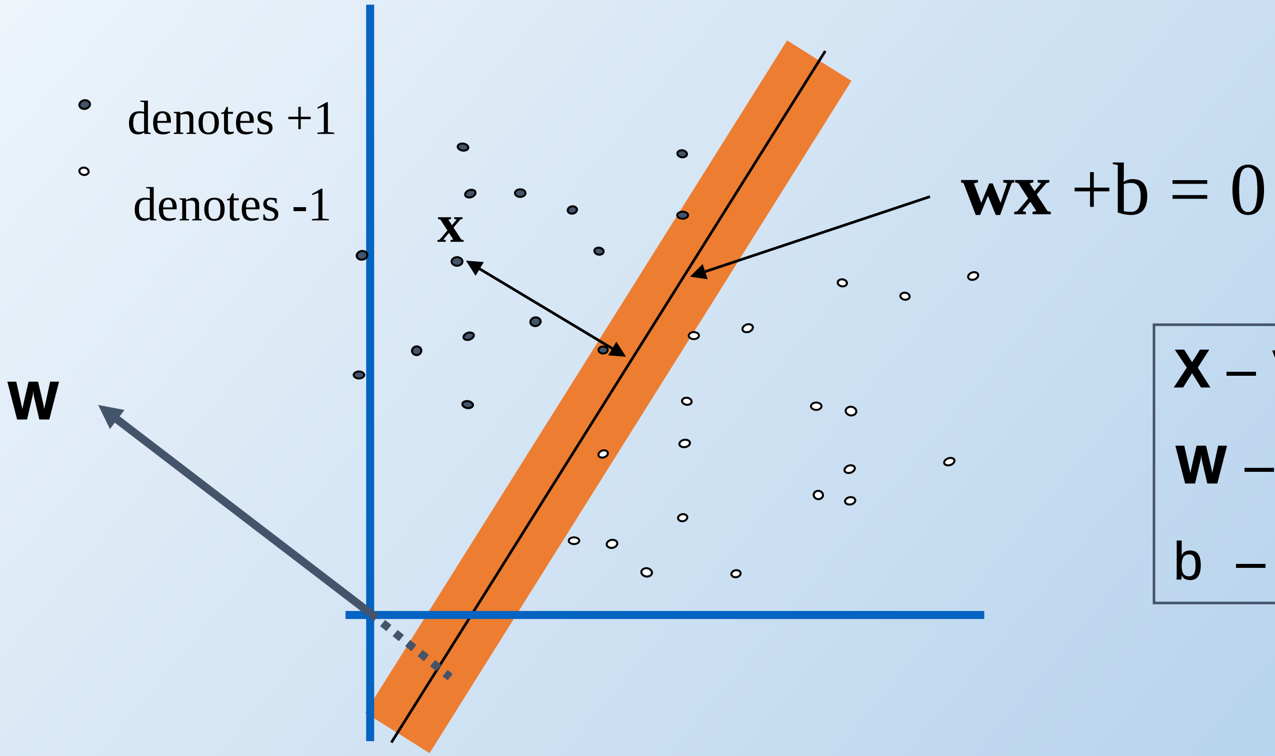


Support Vectors are those datapoints that the margin pushes up against

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

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

How to calculate the distance from a point to a line?



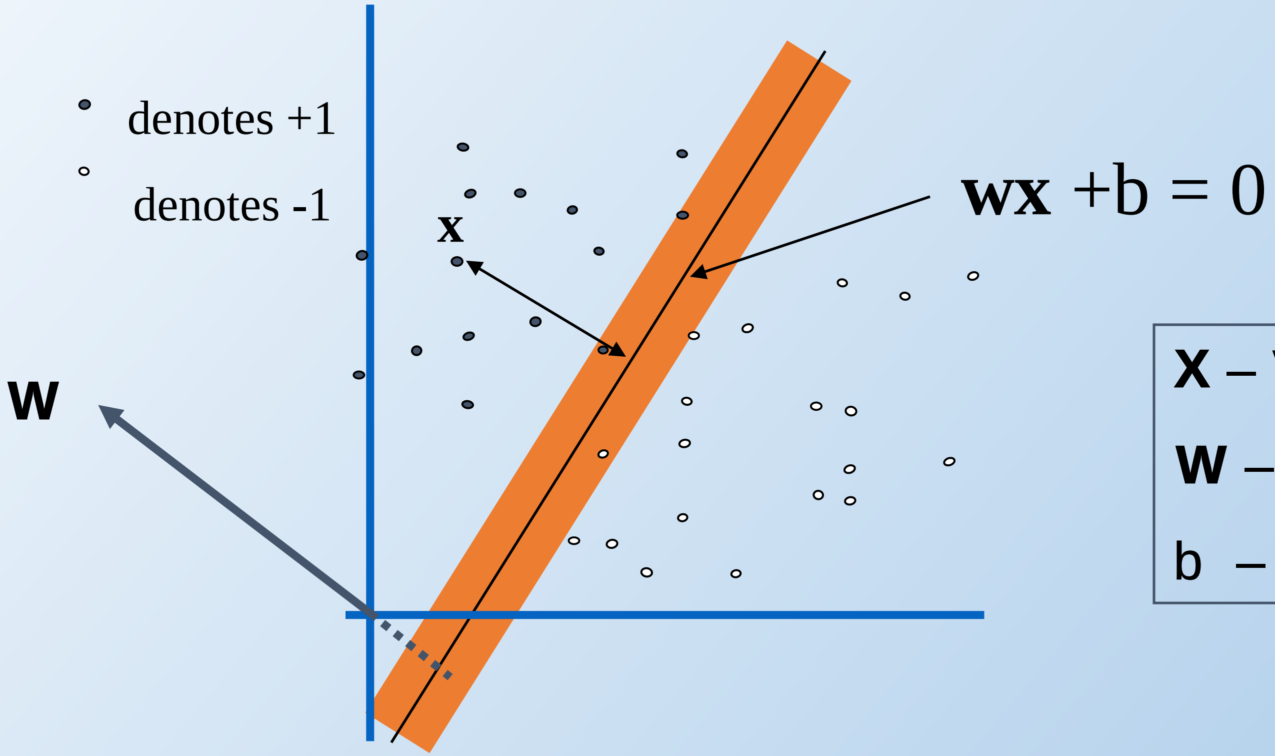
X – Vector

W – Normal Vector

b – Scale Value

- In our case, $w_1*x_1+w_2*x_2+b=0$,
- thus, $\mathbf{w}=(w_1,w_2)$, $\mathbf{x}=(x_1,x_2)$

Estimate the Margin



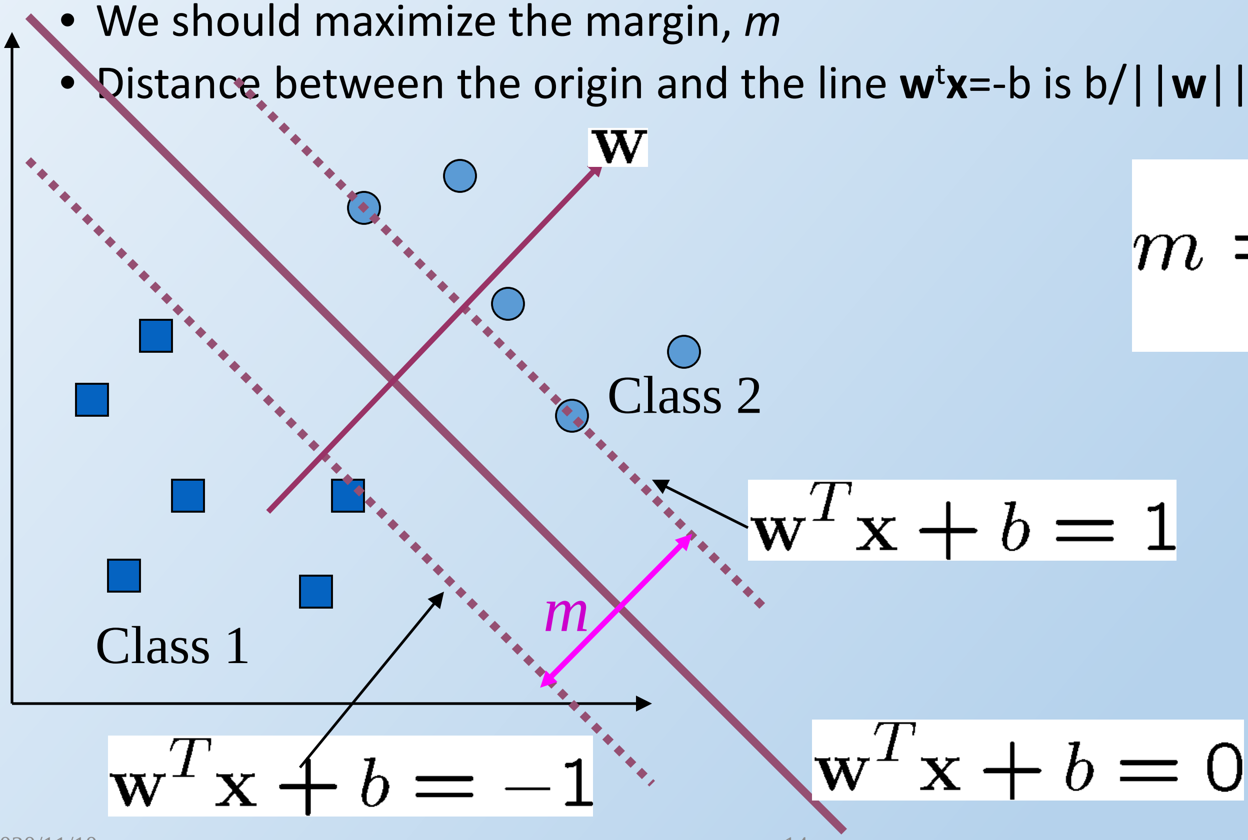
X – Vector
W – Normal Vector
b – Scale Value

- What is the distance expression for a point **x** to a line **wx**+b= 0?

$$d(\mathbf{x}) = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*
  - Distance between the origin and the line $\mathbf{w}^t\mathbf{x}$=-b is b/||**w**||



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

Finding the Decision Boundary

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i

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

- The decision boundary should classify all points correctly $\Rightarrow$
- To see this: when y=-1, we wish (wx+b)<1, when y=1, we wish (wx+b)>1. For support vectors, we wish y(wx+b)=1.
- The decision boundary can be found by solving the following constrained optimization problem

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

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

The Dual Problem (we ignore the derivation)

- The new objective function is in terms of α_i only
- It is known as the dual problem: if we know **w**, we know all α_i; if we know all α_i, we know **w**
- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized!
- The dual problem is therefore:

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

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

Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

The Dual Problem

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

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

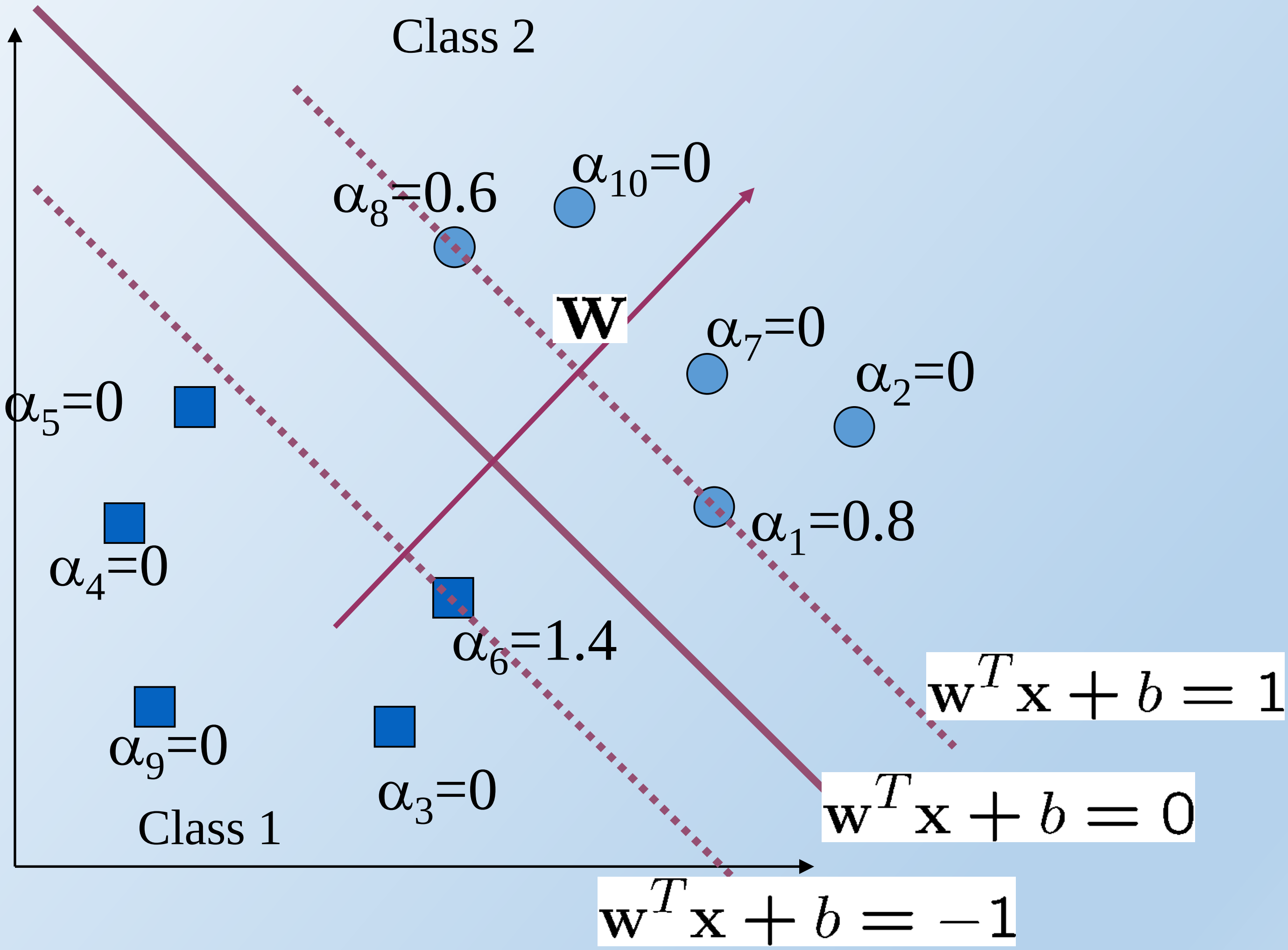
- This is a quadratic programming (QP) problem
  - A global maximum of α_i can always be found
- **w** can be recovered by $$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Characteristics of the Solution

- Many of the α_i are zero (see next page for example)
  - **w** is a linear combination of a small number of data points
  - This "sparse" representation can be viewed as data compression as in the construction of knn classifier
- $\mathbf{x}_i$ with non-zero α_i are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let t_j (j=1, ..., s) be the indices of the s support vectors. We can write
- For testing with a new data **z**
  - Compute $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$ and classify **z** as class 1 if the sum is positive, and class 2 otherwise
  - Note: **w** need not be formed explicitly

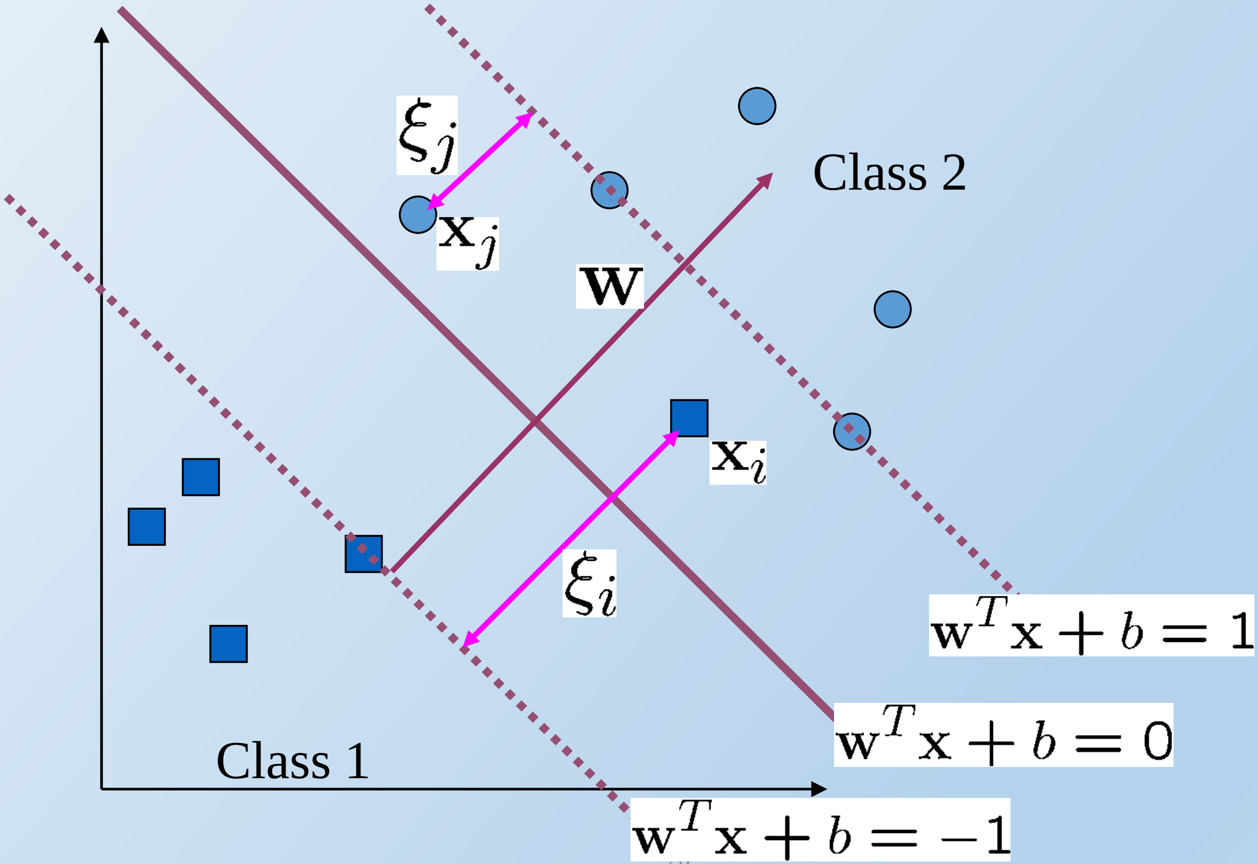
$$\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$$

A Geometrical Interpretation



Allowing errors in our solutions

- 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



Soft Margin Hyperplane

- If we minimize $$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$
  - ξ_i are "slack variables" in optimization
  - Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors
- We want to minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$
  - *C* : tradeoff parameter between error and margin
- The optimization problem becomes

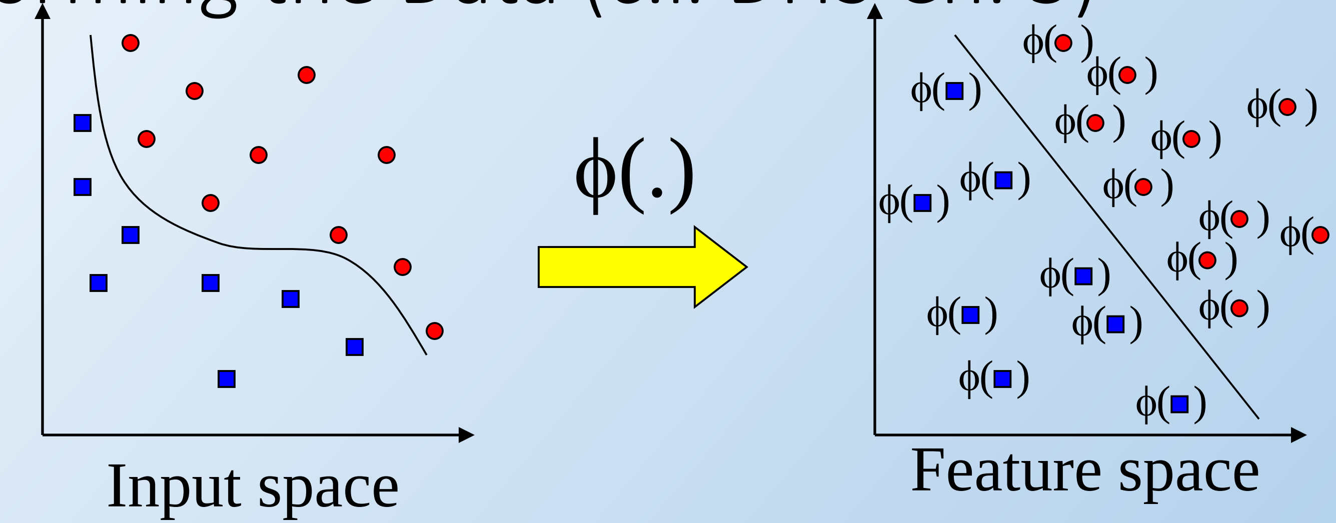
$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation

Transforming the Data (c.f. DHS Ch. 5)



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

The Kernel Trick

- Recall th

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

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

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by

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

An Example for ϕ(.) and K(.,.)

- Suppo $\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- An in $\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$
- So, if we define the kernel function as follows, there is no need to carry out ϕ(.) explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out ϕ(.) explicitly is known as the kernel trick

Examples of Kernel Functions

Not all similarity measures can be used as kernel function, however – there are theoretical requirements given by Mercer's theorem (we'll skip this)

Some comonly-used kernel functions are:

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / (2\sigma^2))$$

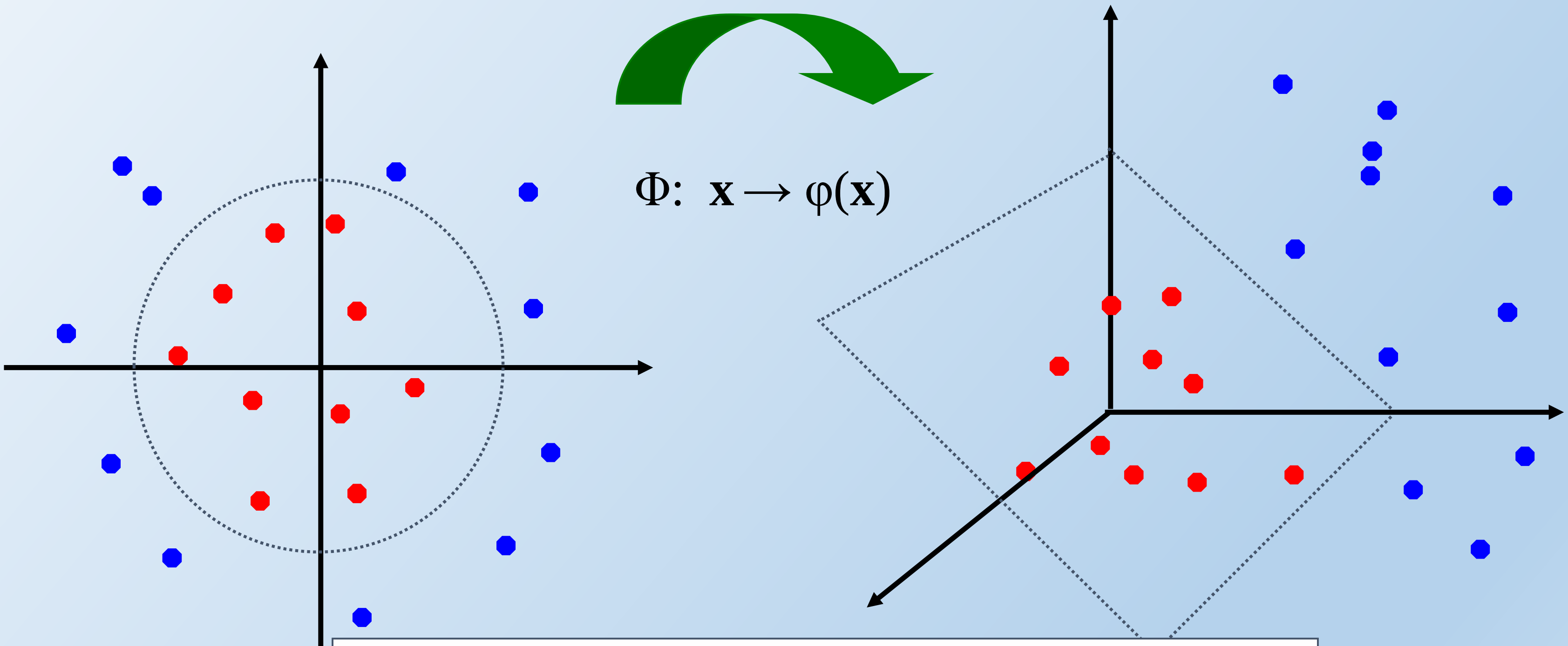
  - Closely related to radial basis function neural networks
  - The feature space is infinite-dimensional
- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

  - It does not satisfy the Mercer condition on all κ and θ

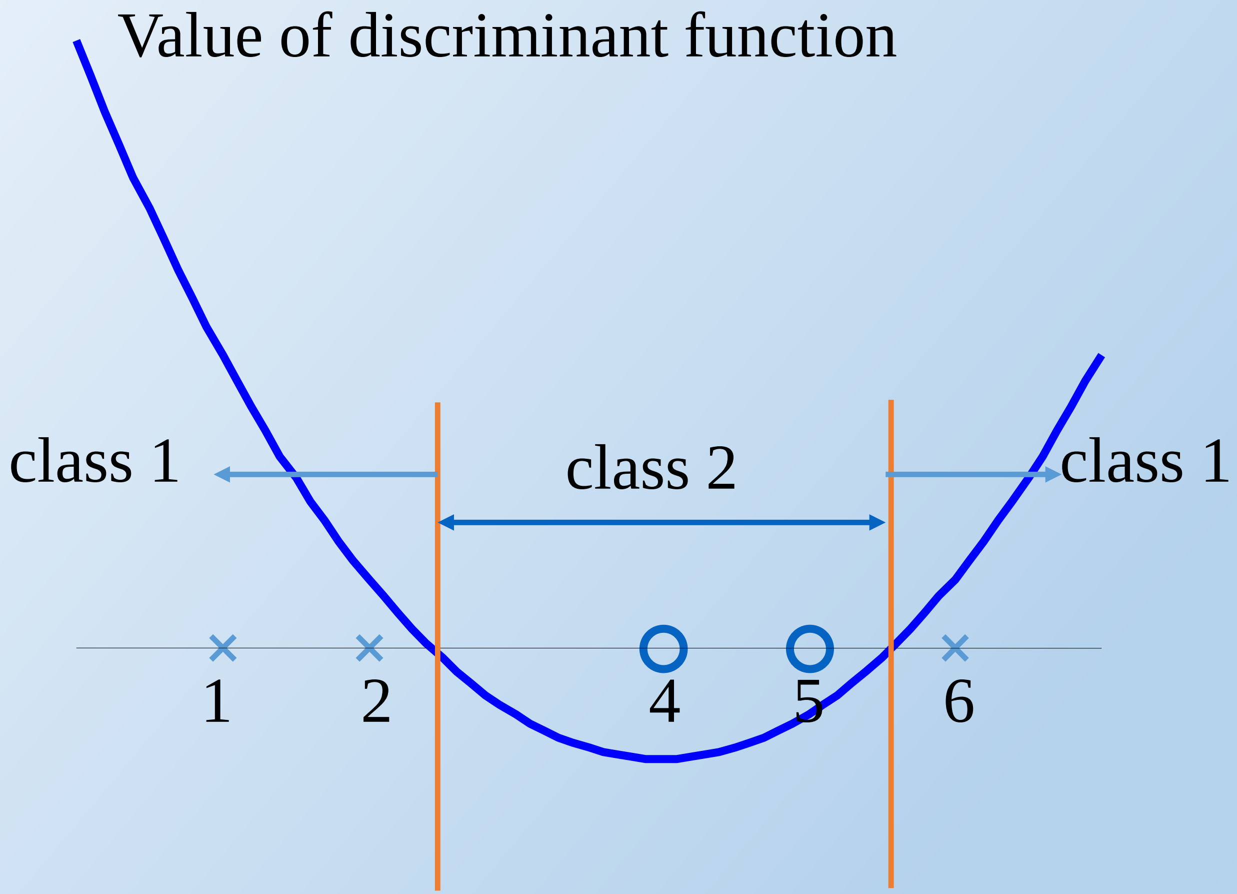
Non-linear SVMs: Feature spaces

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



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

Example



Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

Example

- Suppose we have 5 one-dimensional 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, …, 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
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
  - Note that the constraints are indeed satisfied
  - The support vectors are {x_2=2, x_4=5, x_5=6}
- The discriminant function is

$$
\begin{aligned}
& f(z) \\
&= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + \overset{\alpha_5}{4.833}\overset{y_5}{(1)}\overset{K(z,x_5)}{(6z+1)^2} + b \\
&= 0.6667z^2 - 5.333z + b
\end{aligned}
$$

- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1,

  as x_2 and x_5 lie on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = 1$

  and x_4 lies on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = -1$

  All three give b=9

$$\Rightarrow \quad f(z) = 0.6667z^2 - 5.333z + 9$$

Summary: Steps for Classification

- Prepare the data matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Summary

Two key concepts of SVM: maximize the margin and the kernel trick

Strengths

- Training is relatively easy
  - We don't have to deal with local minima like in ANN
  - SVM solution is always global and unique
- Unlike ANN, doesn't suffer from "curse of dimensionality".
  - How? Why? We have infinite dimensions?!
  - Maximum Margin Constraint: DOT-PRODUCTS!
- Less prone to overfitting
- Simple, easy to understand geometric interpretation.

Weaknesses

- Training is slow compared to ANN
  - Because of Constrained Quadratic Programming
- Essentially a binary classifier
  - However, there are some tricks to evade this.
- Very sensitive to noise
  - A few outliers can completely throw off the algorithm
- Drawback: The choice of Kernel function.
  - There is no "set-in-stone" theory for choosing a kernel function

Review Questions

- What is the maximum margin hyperplane?
- What is a support vector?
- How are support vectors found?
- How do SVMs deal with non-separable classes?
- What is the kernel trick in support vector machines?