KERNELS, SVM

CS5691- Pattern Recognition & MACHINE LEARNING

Murphy 14.1, 14.2.1-14.2.6, 14.3, 14.4, 14.5

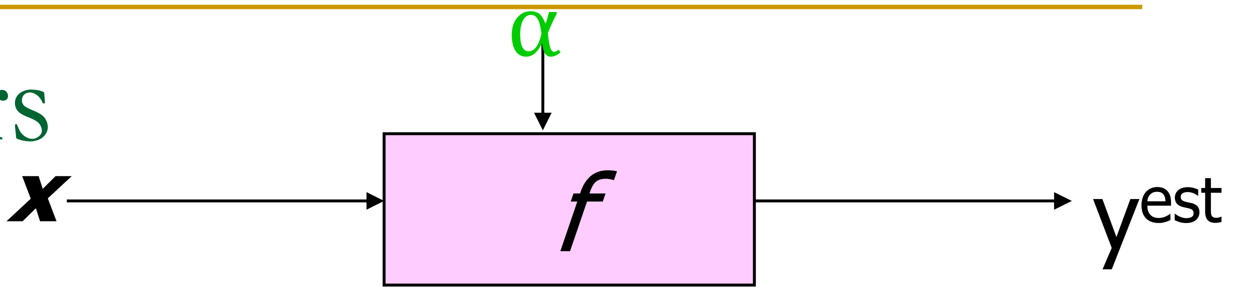
+ Wiki + Online Tut notes;

Eg - http://www.luigifreda.com/wp-content/uploads/2018/01/lec9.pdf

Introduction

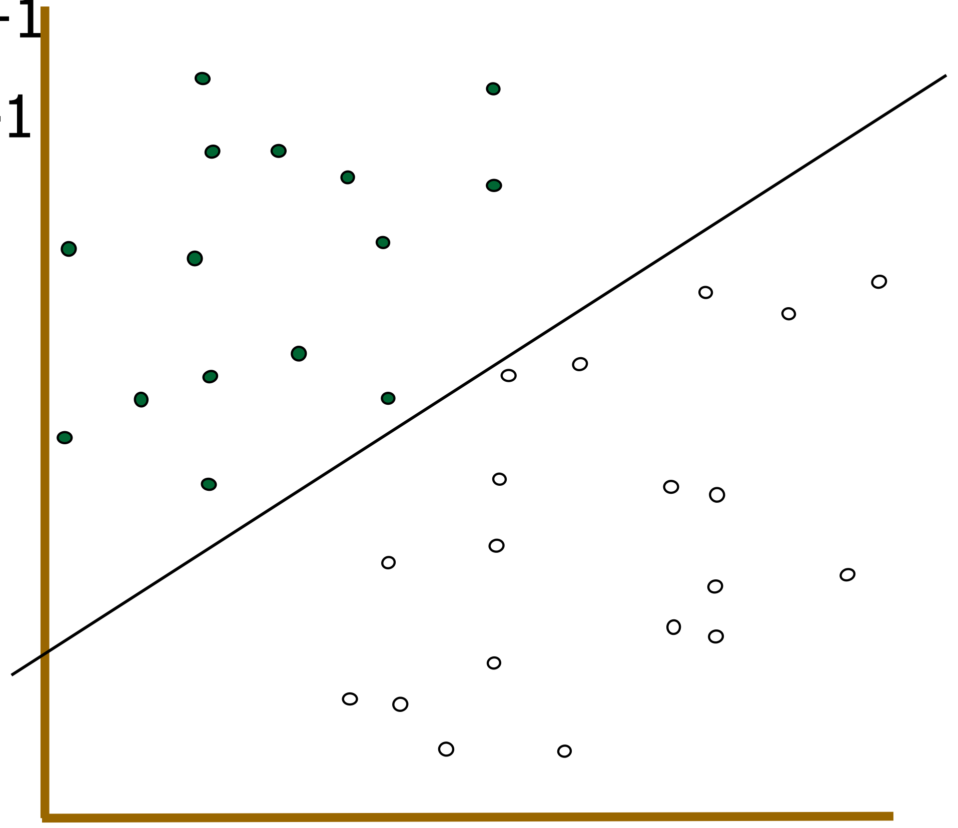
- How do we represent a text document or protein sequence, which can be of variable length?
- One approach is to define a generative model for the data, and use the inferred latent representation and/or the parameters of the model as features, and then to plug these features in to standard methods
- Another approach is to assume that we have a way of measuring the similarity between objects, that doesn't require preprocessing them into feature vector format
- For example, when comparing strings, we can compute the edit distance between them

Linear Classifiers



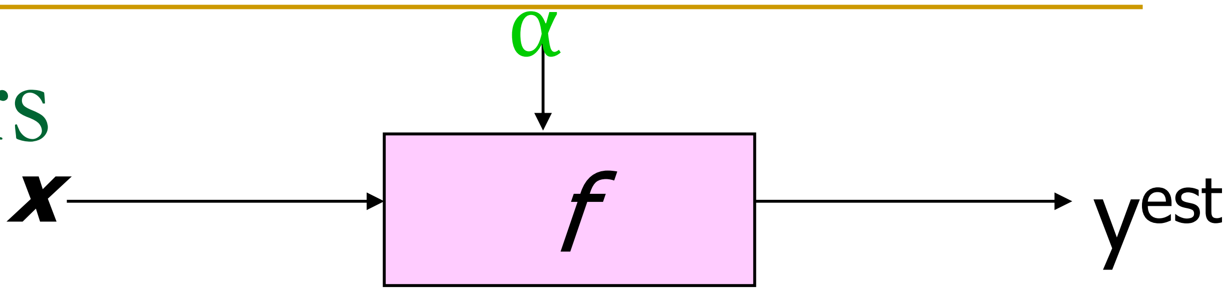
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- ● denotes +1
- ○ denotes -1

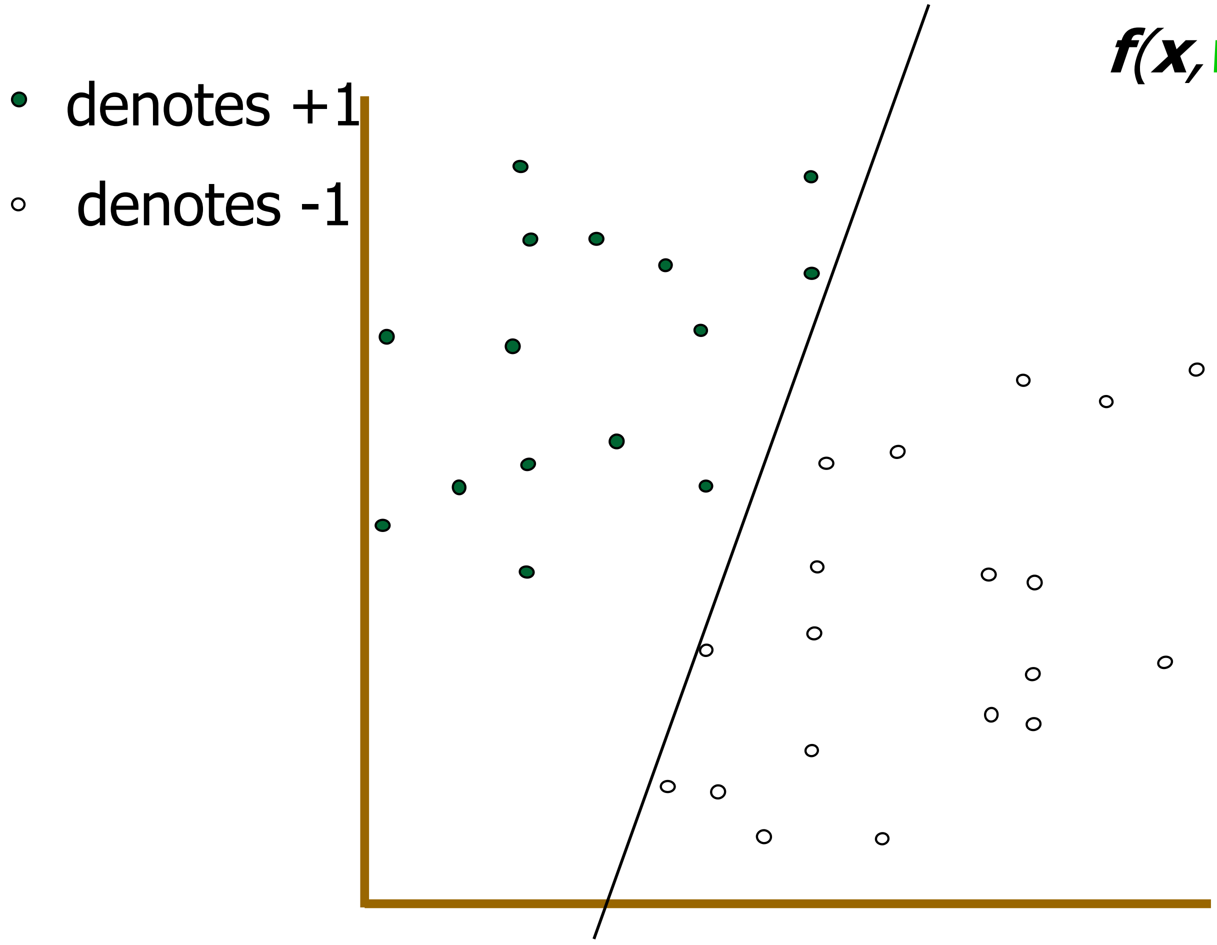


How would you classify this data?

Linear Classifiers

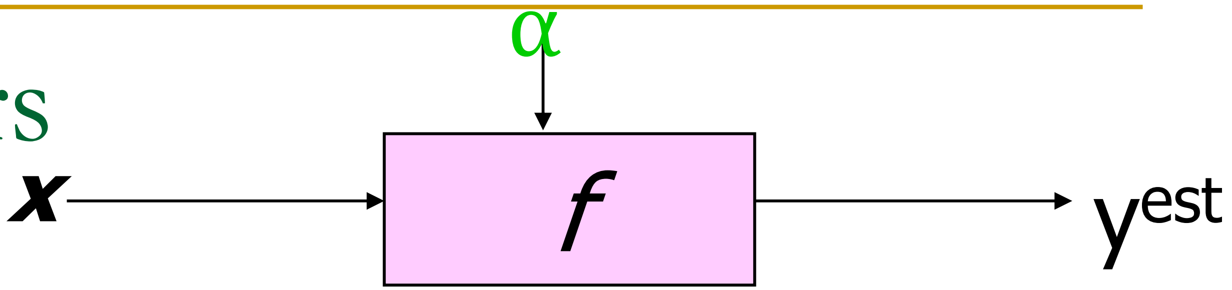


***f*(*x*,*w*,*b*) = *sign*(*w* *x* + *b*)**



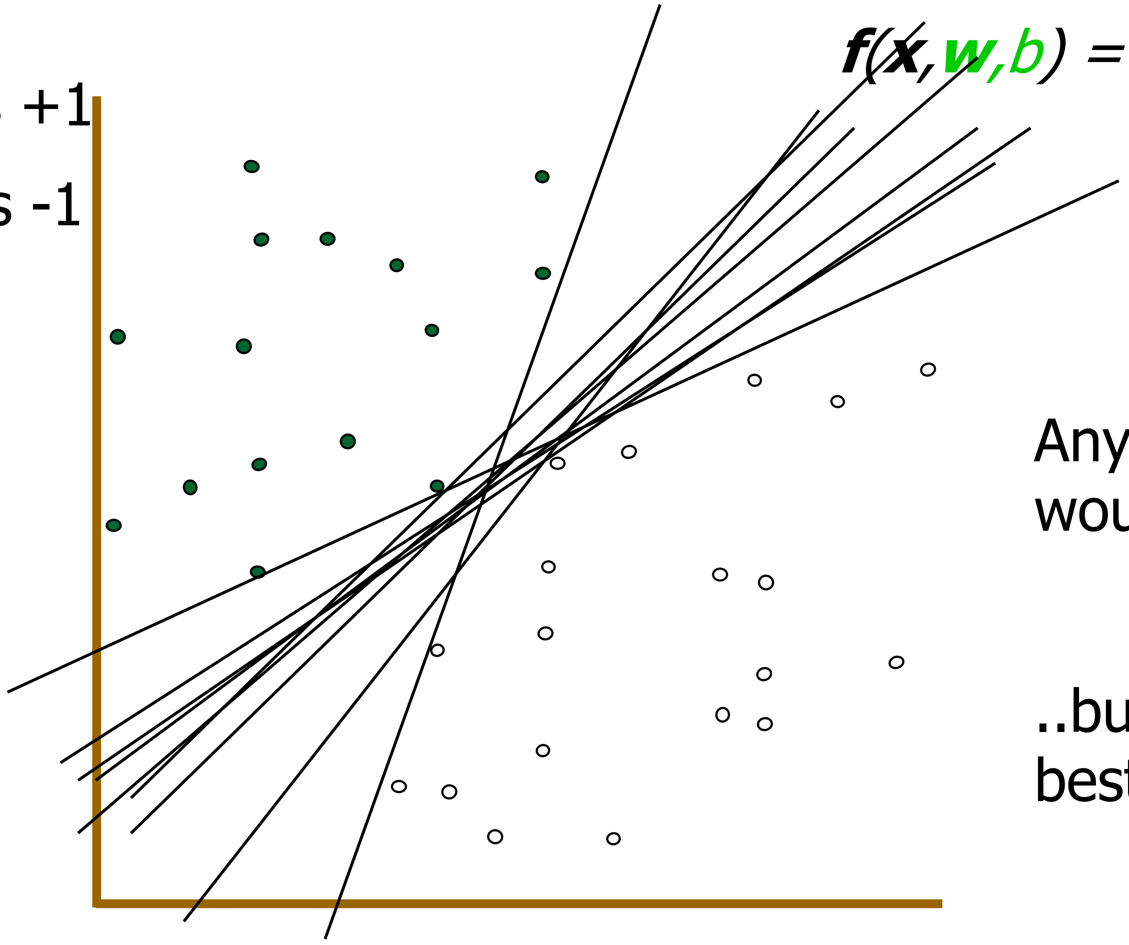
How would you classify this data?

Linear Classifiers



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

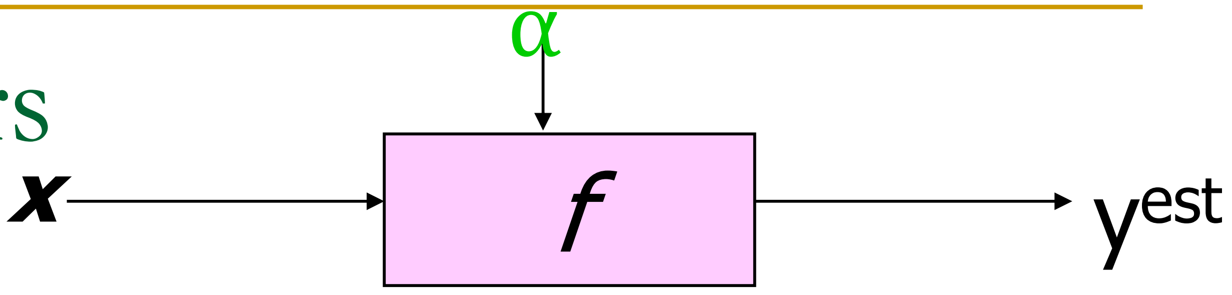
- ● denotes +1
- ○ denotes -1



Any of these would be fine..

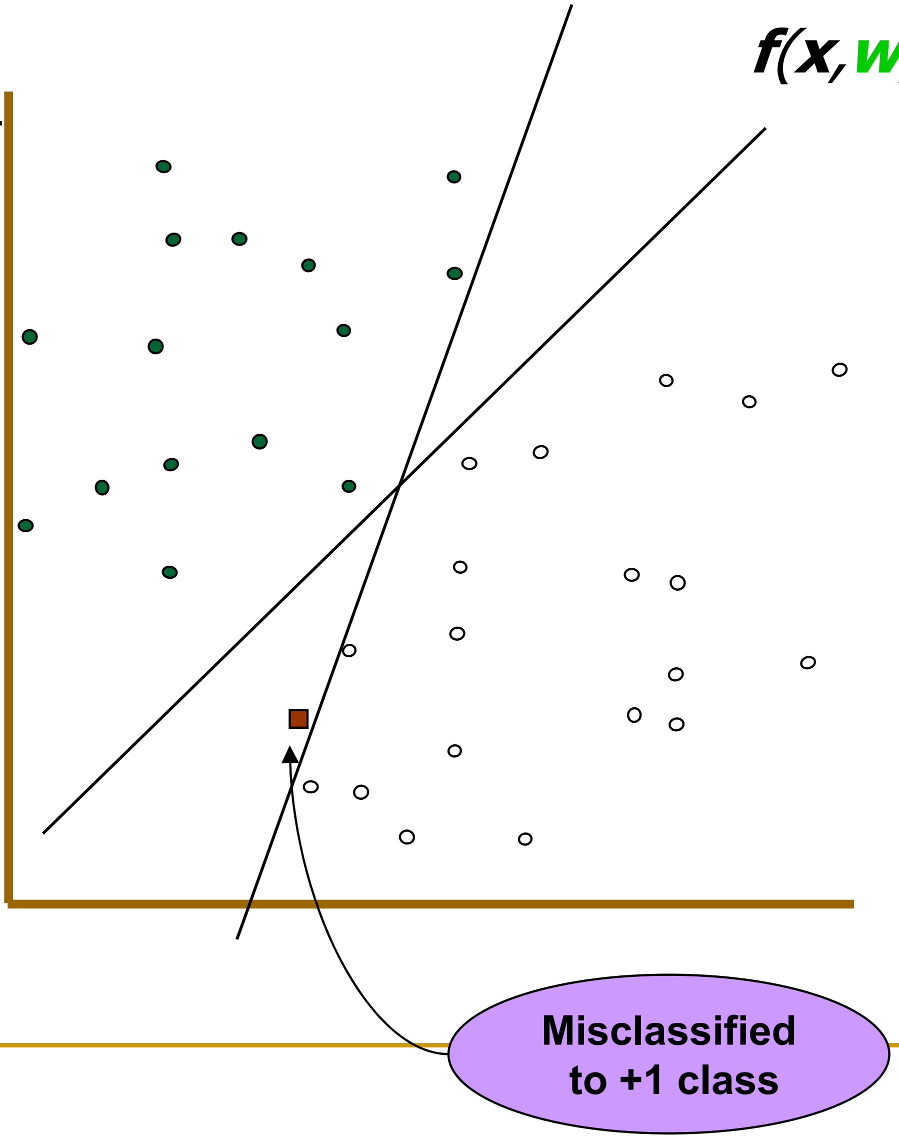
..but which is best?

Linear Classifiers



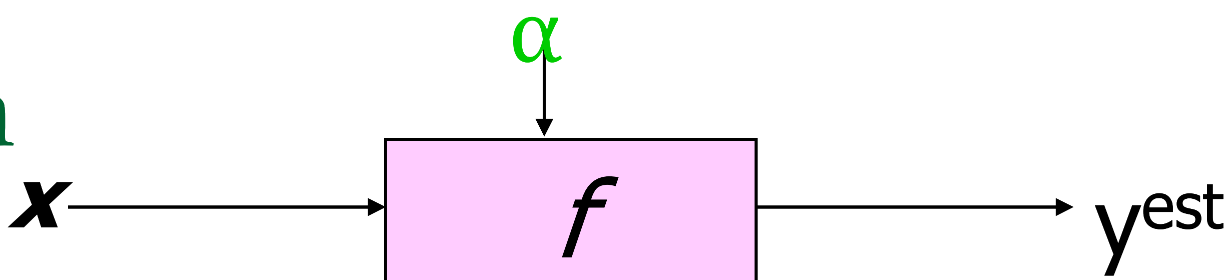
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- ● denotes +1
- ○ denotes -1



How would you classify this data?

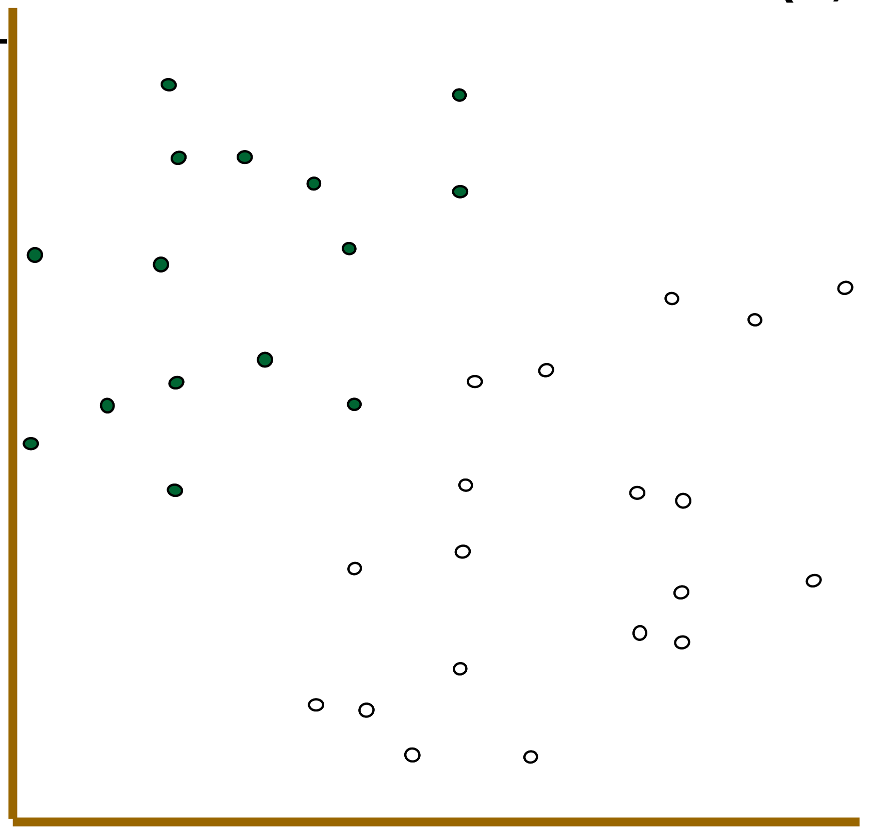
Classifier Margin



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

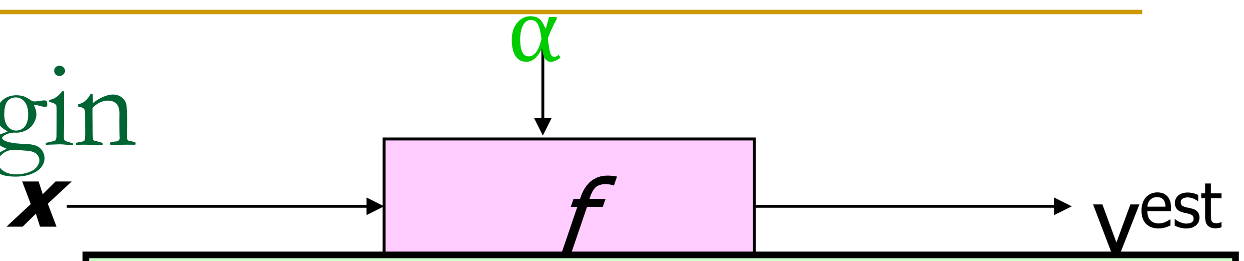
● denotes +1

○ denotes -1

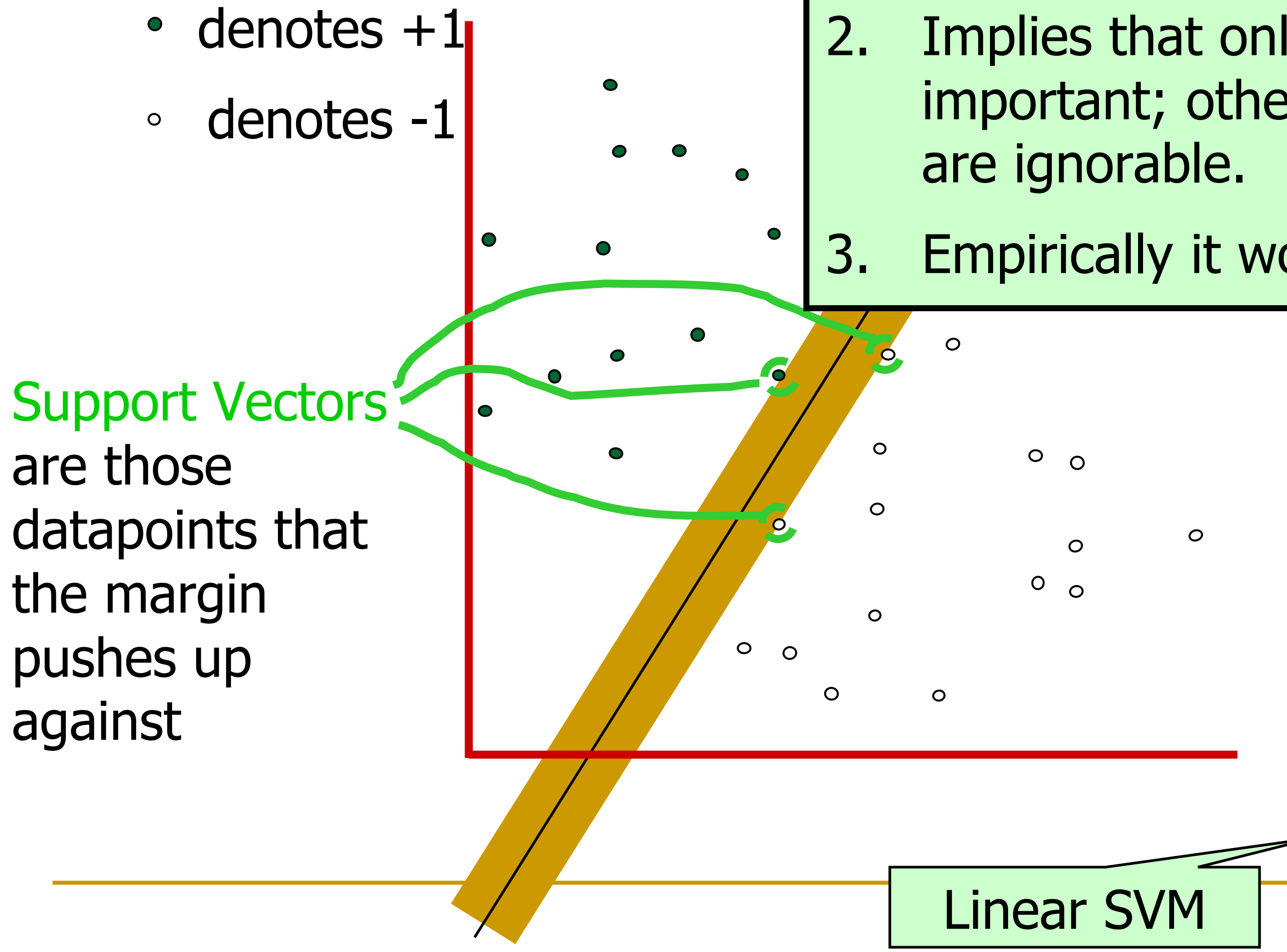


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



1. Maximizing the margin is good according to intuition
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very well.



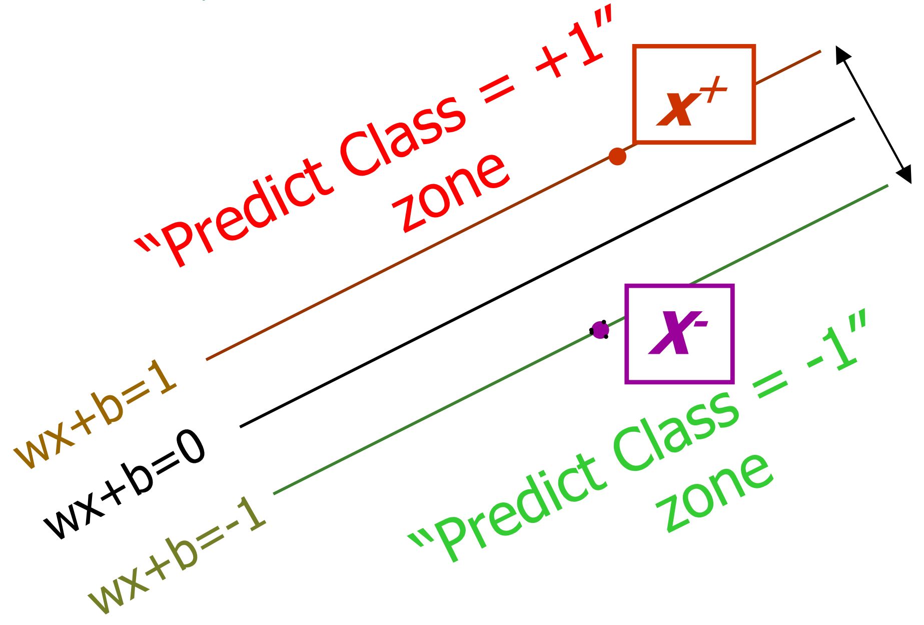
Support Vectors are those datapoints that the margin pushes up against

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

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

Linear SVM

Linear SVM Mathematically



M=Margin Width

What we know:

- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$
- $\mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Linear SVM Mathematically

- Goal: **1) Correctly classify all training data**

$$\left.\begin{array}{ll} wx_i + b \geq 1 & \textit{if } y_i = +1 \\ wx_i + b \leq 1 & \textit{if } y_i = -1 \end{array}\right\}$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$

2) Maximize the Margin $M = \dfrac{2}{|w|}$

same as minimize $\dfrac{1}{2}w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize
$$\boxed{\begin{array}{ll} \Phi(w) = \dfrac{1}{2}w^t w & \\ y_i(wx_i + b) \geq 1 & \forall i \end{array}}$$

subject to

$$\text{minimize } f(\mathbf{x}) \tag{10.107a}$$

$$\text{subject to } \quad a_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i \quad \text{for } 1 \le i \le p \tag{10.107b}$$

$$c_j(\mathbf{x}) \ge 0 \quad \text{for } 1 \le j \le q \tag{10.107c}$$

where $f(\mathbf{x})$ and $-c_j(\mathbf{x})$ for $1 \le j \le q$ are convex functions. The main results, which are analogous to those in Sec. 2.8, are described by the next two theorems.

Theorem 10.7 *Globalness and convexity of minimizers in CP problems*

(a) If $\mathbf{x}^*$ is a local minimizer of a CP problem, then $\mathbf{x}^*$ is also a global minimizer.

(b) The set of minimizers of a CP problem, denoted as S, is convex.

(c) If the objective function $f(\mathbf{x})$ is strictly convex on the feasible region $\mathcal{R}$, then the global minimizer is unique.

CP – Convex Programming

Theorem 10.9 *Duality in convex programming* *Let* $\mathbf{x}^*$ *be a minimizer, and* λ^*, μ^* *be the associated Lagrange multipliers of the problem in Eq. (10.107). If* $\mathbf{x}^*$ *is a regular point of the constraints, then* $\mathbf{x}^*$, λ^*, *and* μ^* *solve the dual problem*

$$\underset{\mathbf{x},\lambda,\mu}{\text{maximize}}\ L(\mathbf{x},\lambda,\mu) \tag{10.109a}$$

$$\text{subject to :}\quad \nabla_x L(\mathbf{x},\lambda,\mu) = 0 \tag{10.109b}$$

$$\mu \geq 0 \tag{10.109c}$$

In addition, $f(\mathbf{x}^*) = L(\mathbf{x}^*, \lambda^*, \mu^*)$.

Example 10.16 Find the Wolfe dual of the standard-form LP problem

$$\text{minimize } \mathbf{c}^T\mathbf{x} \qquad (10.110a)$$

$$\text{subject to: } \mathbf{A}\mathbf{x} = \mathbf{b} \qquad \mathbf{A} \in R^{p\times n} \qquad (10.110b)$$

$$\mathbf{x} \geq \mathbf{0} \qquad (10.110c)$$

Solution The Lagrangian is given by

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{c}^T\mathbf{x} - (\mathbf{A}\mathbf{x} - \mathbf{b})^T\boldsymbol{\lambda} - \mathbf{x}^T\boldsymbol{\mu}$$

and the dual problem can be stated as

$$\underset{\boldsymbol{\lambda},\, \boldsymbol{\mu}}{\text{maximize}} \ \mathbf{b}^T\boldsymbol{\lambda} \qquad (10.1$$

$$\text{subject to: } \mathbf{c} - \mathbf{A}^T\boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0} \qquad (10.1$$

$$\boldsymbol{\mu} \geq \mathbf{0} \qquad (10.1$$

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

$$f(w,b) = \frac{1}{2}|w|^2$$

$$g(w,b) = y_i(\vec{x} \cdot \vec{w} + b) - 1 = 0$$

$$L_{\min(w,b)}(w,b) = \frac{1}{2}|w|^2 - \sum_i \alpha_i [y_i(\vec{x_i} \cdot \vec{w} + b) - 1]$$

By considering: $\frac{|w|^2}{\partial w} = \frac{\vec{w} \cdot \vec{w}}{\partial w}$

$$\frac{L(w,b)}{\partial w} = \vec{w} - \sum_i \alpha_i y_i \vec{x}$$

$$\frac{L(w,b)}{\partial b} = - \sum_i \alpha_i y_i$$

Solving the Optimization Problem

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **Need to optimize a *quadratic* function subject to *linear* constraints.**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized, and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

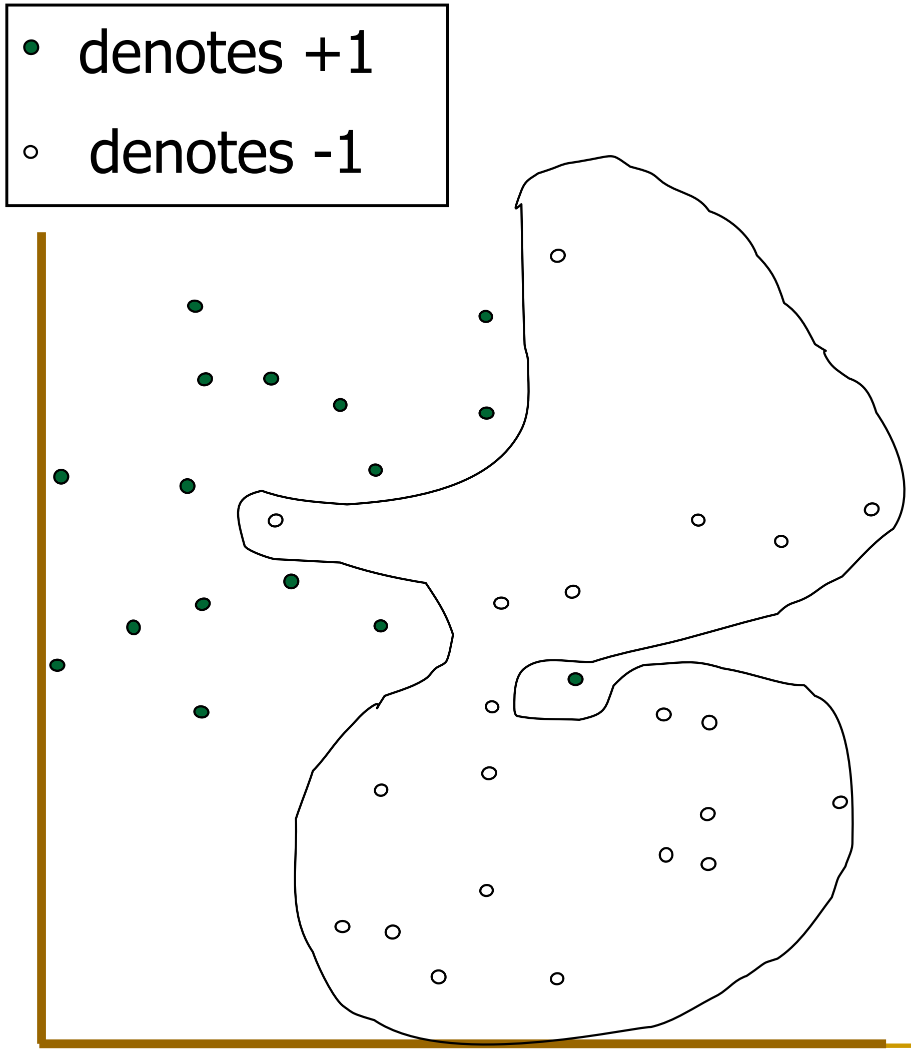
$$\boxed{\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w^T x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0}$$

- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$\boxed{f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i}^{\mathbf{T}} \mathbf{x} + b}$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$ between all pairs of training points.

Dataset with noise



- **Hard Margin: So far we require all data points be classified correctly**

  - No training error

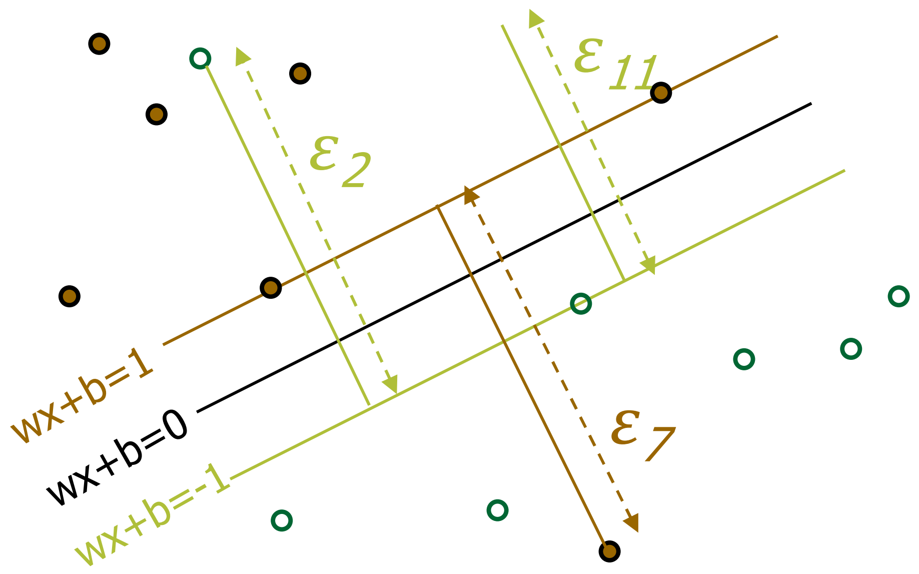
- **What if the training set is noisy?**

  - Solution 1: use very powerful kernels

OVERFITTING!

Soft Margin Classification

***Slack variables ξi* can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

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

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^\mathbf{T}\mathbf{x_i} + \mathrm{b}) \geq 1$

- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter *C* can be viewed as a way to control overfitting.**

Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i - b)\right)\right] + \lambda\|\mathbf{w}\|^2. \qquad (2)$$

Primal

Minimizing **(2)** can be rewritten as a constrained optimization problem with a differentiable objective function in the following way.

For each $i \in \{1, \ldots, n\}$ we introduce a variable $\zeta_i = \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i - b)\right)$. Note that ζ_i is the smallest nonnegative number satisfying $y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \zeta_i$.

Thus we can rewrite the optimization problem as follows

$$\begin{aligned}&\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|\mathbf{w}\|^2\\ &\text{subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i - b\right) \geq 1 - \zeta_i \ \text{ and } \ \zeta_i \geq 0,\ \text{for all } i.\end{aligned}$$

This is called the *primal* problem.

Dual

By solving for the Lagrangian dual of the above problem, one obtains the simplified problem

$$\begin{aligned}&\text{maximize}\,\, f(c_1 \ldots c_n) = \sum_{i=1}^n c_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n y_i c_i (\mathbf{x}_i^\mathsf{T}\mathbf{x}_j) y_j c_j,\\ &\text{subject to } \sum_{i=1}^n c_i y_i = 0,\,\text{and } 0 \leq c_i \leq \frac{1}{2n\lambda}\;\text{for all }i.\end{aligned}$$

This is called the *dual* problem. Since the dual maximization problem is a quadratic function of the c_i subject to linear constraints, it is efficiently solvable by quadratic programming algorithms.

Here, the variables c_i are defined such that

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

Moreover, $c_i = 0$ exactly when $\mathbf{x}_i$ lies on the correct side of the margin, and $0 < c_i < (2n\lambda)^{-1}$ when $\mathbf{x}_i$ lies on the margin's boundary. It follows that $\mathbf{w}$ can be written as a linear combination of the support vectors.

The offset, b, can be recovered by finding an $\mathbf{x}_i$ on the margin's boundary and solving

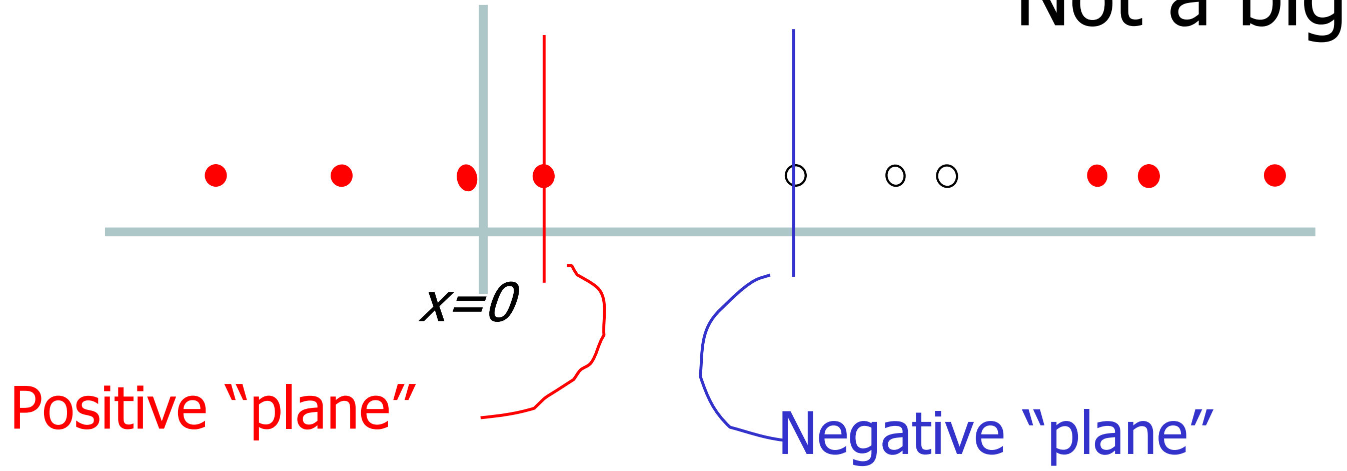
$$y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) = 1 \iff b = \mathbf{w}^\mathsf{T}\mathbf{x}_i - y_i.$$

(Note that $y_i^{-1} = y_i$ since $y_i = \pm 1$.)

Hard 1-dimensional Dataset

What would SVMs do with this data?

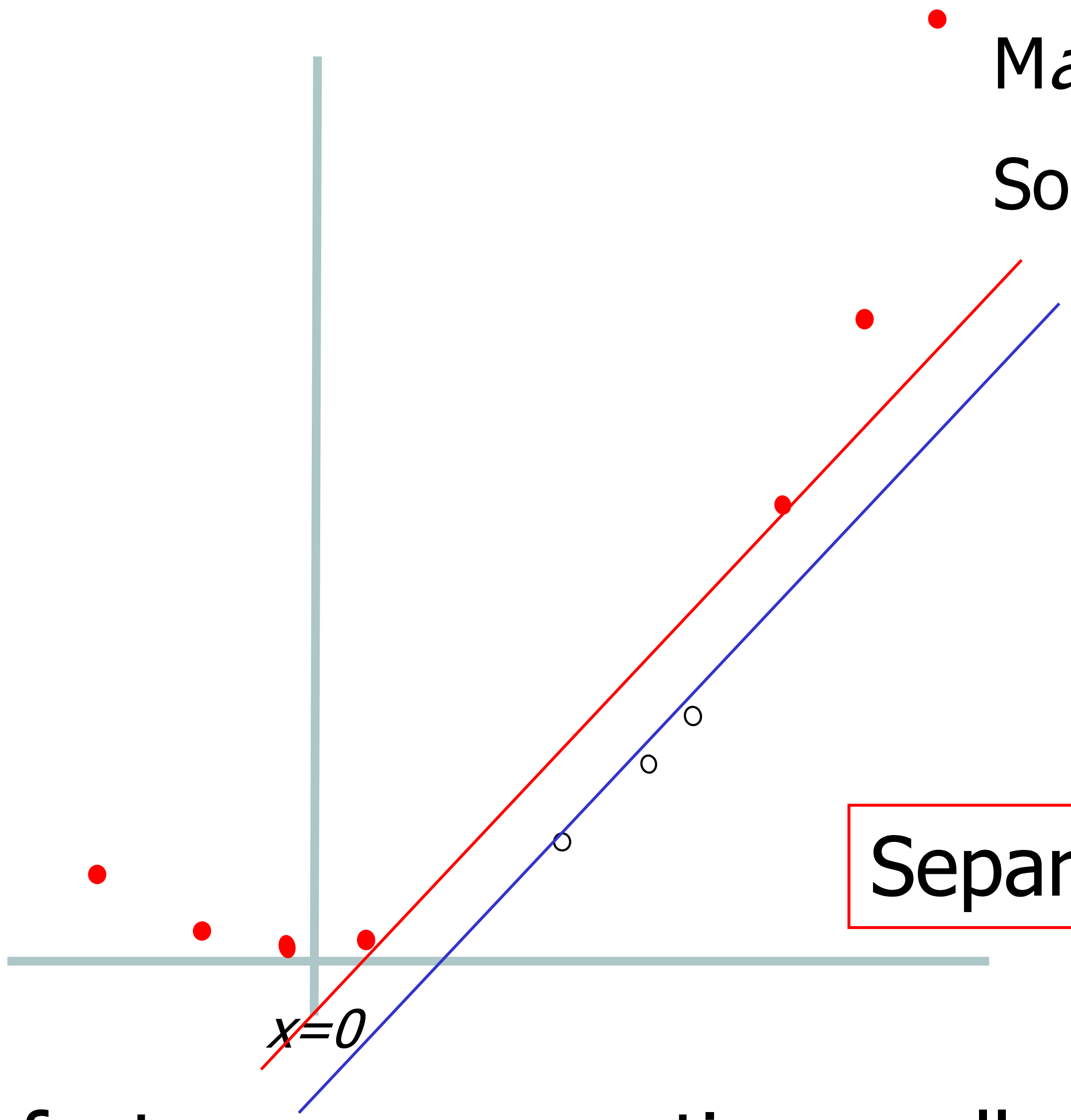
Not a big surprise



Doesn't look like slack variables will save us this time…

taken from Andrew W. Moore 23

Hard 1-dimensional Dataset



M*ake up a new feature!*

Sort of…

… computed from original feature(s)

$$\boxed{\mathbf{z}_k = (x_k, x_k^2)}$$

Separable! MAGIC!

New features are sometimes called *basis functions.*

Now drop this "augmented" data into our linear SVM.

taken from Andrew W. Moore

Kernels and Linear Classifiers

Let $\vec{x} = [\vec{x}_1, \vec{x}_2] \in \mathbb{R}^2$ be a vectorial represenation
of object $x \in \mathcal{X}$

Let $\phi : \mathcal{X} \to \mathcal{K} \subset \mathbb{R}^3$ feature map be given by

$$\phi(\vec{x}) \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T \in \mathcal{K} \subset \mathbb{R}^3$$

Def. Feature space: $\mathcal{K}$

We will use linear classifiers in this feature space.

In the original space $\mathbb{R}^2$ for a given $\mathbf{w} \in \mathbb{R}^3$ the decision surface is:

$$\boxed{\tilde{X}_0(\mathbf{w}) = \{\vec{x} \in \mathbb{R}^2 \mid w_1\vec{x}_1 + w_2\vec{x}_2^2 + w_3\vec{x}_1\vec{x}_2 = 0\}}$$

- This is nonlinear in $\vec{x} \in \mathbb{R}^2$
- This is linear in the feature space $\phi(\vec{x}) \in \mathcal{K} \subset \mathbb{R}^3$

Kernels and Linear Classifiers

$$\phi(\vec{x}) \doteq [\phi_1(\vec{x}), \phi_2(\vec{x}), \phi_3(\vec{x})] \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T$$

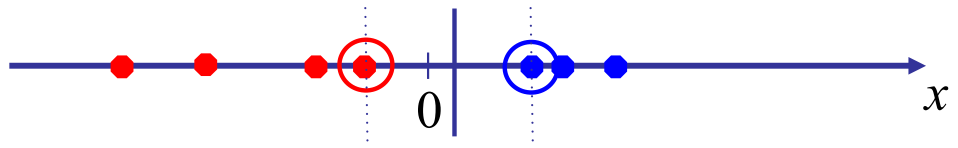
Feature functions

- We seek for a small set of basis vectors $\{\phi_i\}$ which allows perfect discrimination between the classes in $\mathcal{X}$ (**Feature selection**)

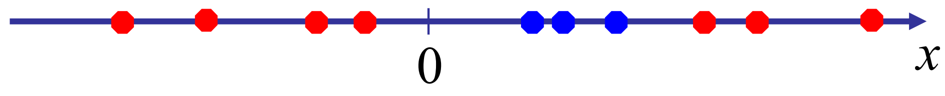
- If we have too many features $\Rightarrow$ overfitting can happen.

Non-linear SVMs

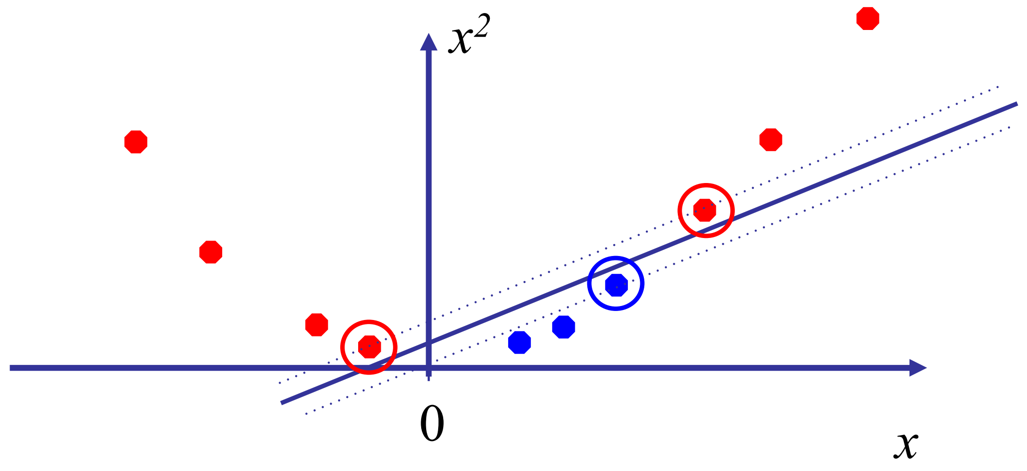
- Datasets that are linearly separable with some noise work out great:



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

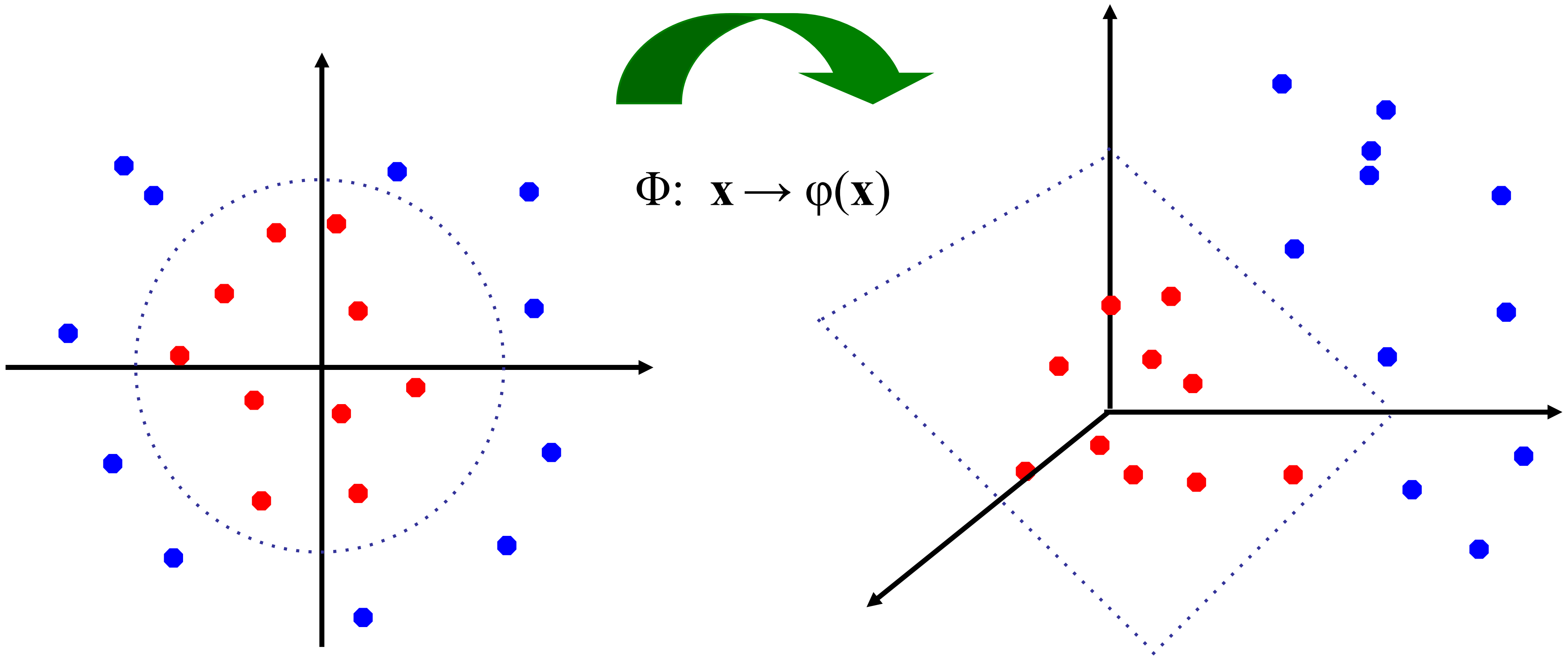


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

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



Kernel functions

- We define a kernel function to be a real-valued function of two arguments, $\kappa(\mathbf{x}, \mathbf{x}') \in R$, for $\mathbf{x}, \mathbf{x}' \in \mathrm{X}$.
- *X* is some abstract space
- Typically the function has the following properties:
    - Symmetric
    - Non-negative
    - Can be interpreted as a measure of similarity

- We will discuss several examples of kernel functions

RBF kernels

- **Squared exponential kernel** (SE kernel) or **Gaussian kernel** $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{x}')\right)$

- If **Σ** is diagonal, this can be written as

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_{j=1}^{D} \frac{1}{\sigma_j^2}(x_j - x_j')^2\right)$$

We can interpret the σ_j as defining the **characteristic length scale** of dimension *j*

- If **Σ** is spherical, we get the isotropic kernel

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

An example of RBF (Radial basis function) kernel (since it is a function of ||x – x'||) where σ^2 is known as the **bandwidth**

Kernels for comparing documents

- If we use a bag of words representation, where $\mathbf{x}_{ij}$ is the number of times words j occurs in document i, we can use the **cosine similarity**

$$\kappa(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{\mathbf{x}_i^T \mathbf{x}_{i'}}{||\mathbf{x}_i||_2 ||\mathbf{x}_{i'}||_2}$$

- Unfortunately, this simple method does not work very well
  - Stop words (such as “the” or “and”) are not discriminative
  - Similarity is artificially boosted when a discriminative word occurs multiple times
- Replace the word count vector with Term frequency inverse document frequency (**TF-IDF**)

Kernels for comparing documents

- Define the term frequency as:

$$\mathrm{tf}(x_{ij}) \triangleq \log(1 + x_{ij})$$

- This reduces the impact of words that occur many times with a document
- Define inverse document frequency where N is the total number of documents

$$\mathrm{idf}(j) \triangleq \log \frac{N}{1 + \sum_{i=1}^{N} \mathbb{I}(x_{ij} > 0)}$$

- Our new kernel has the form

$$\kappa(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_{i'})}{||\phi(\mathbf{x}_i)||_2 ||\phi(\mathbf{x}_{i'})||_2}$$

$$\phi(\mathbf{x}) = \text{tf-idf}(\mathbf{x})$$

$$\text{tf-idf}(\mathbf{x}_i) \triangleq [\mathrm{tf}(x_{ij}) \times \mathrm{idf}(j)]_{j=1}^{V}$$

Mercer (positive definite) kernels

- Gram matrix is defined as

$$\mathbf{K} = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ & \vdots & \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

- If the Gram matrix is positive definite for any set of inputs, the Kernel is a Mercer kernel
- Mercer's theorem: If the Gram matrix is positive definite, we can compute an eigenvector decomposition of it as follows: $\mathbf{K} = \mathbf{U}^T \mathbf{\Lambda} \mathbf{U}$

  where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues $\lambda_i > 0$
- Now consider an element of $\mathbf{K}$

$$k_{ij} = (\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{U}_{:,i})^T (\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{U}_{:j})$$

$$k_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \qquad \phi(\mathbf{x}_i) = \mathbf{\Lambda}^{\frac{1}{2}} \mathbf{U}_{:i}$$

Using kernels inside GLMs

- We define a kernel machine to be a GLM (generalized linear model) where the input feature vector has the form

$$\phi(\mathbf{x}) = [\kappa(\mathbf{x}, \boldsymbol{\mu}_1), \ldots, \kappa(\mathbf{x}, \boldsymbol{\mu}_K)]$$

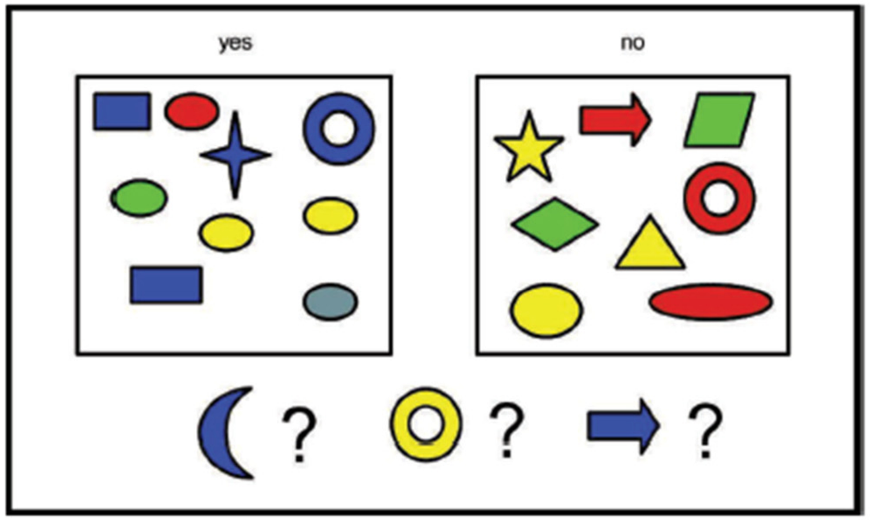
  where $\mu_k \in X$ are a set of K centroids

- If κ is an RBF kernel, this is called an RBF network
- We will discuss ways to choose the μ_k parameters
- Note that in this approach, the kernel need not be a Mercer kernel.
- We can use the kernelized feature vector for logistic regression by defining (using Bernoulli Dist.)

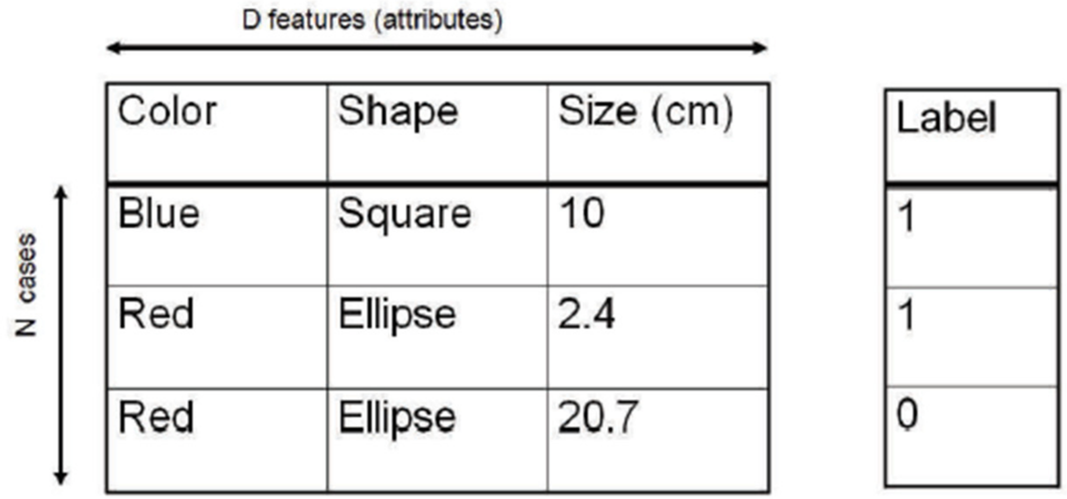
$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \text{Ber}(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})).$$

Design Matrix

Consider a simple toy example of classification



(a)


D features (attributes)
N cases

Color	Shape	Size (cm)
Blue	Square	10
Red	Ellipse	2.4
Red	Ellipse	20.7

Label
1
1
0

(b)

- Two classes of object which correspond to labels 0 and 1 The inputs are colored shapes as shown in (a). These have been described by a set of D features or attributes, which are stored in an $N \times D$ design matrix $\boldsymbol{X}$, shown in (b).

Using kernels inside GLMs

- Use kernelized feature vector inside a linear regression

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \sigma^2).$$

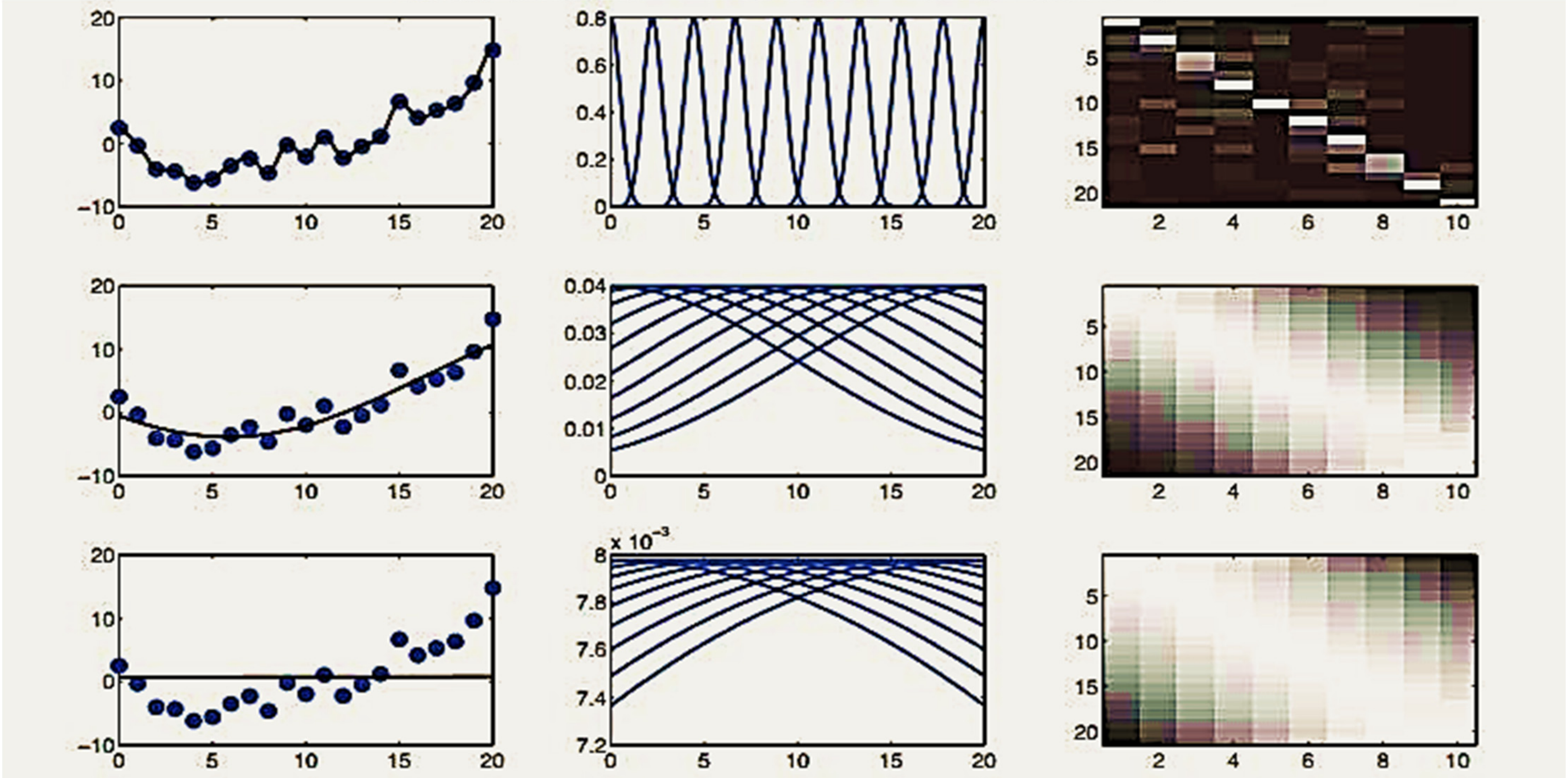
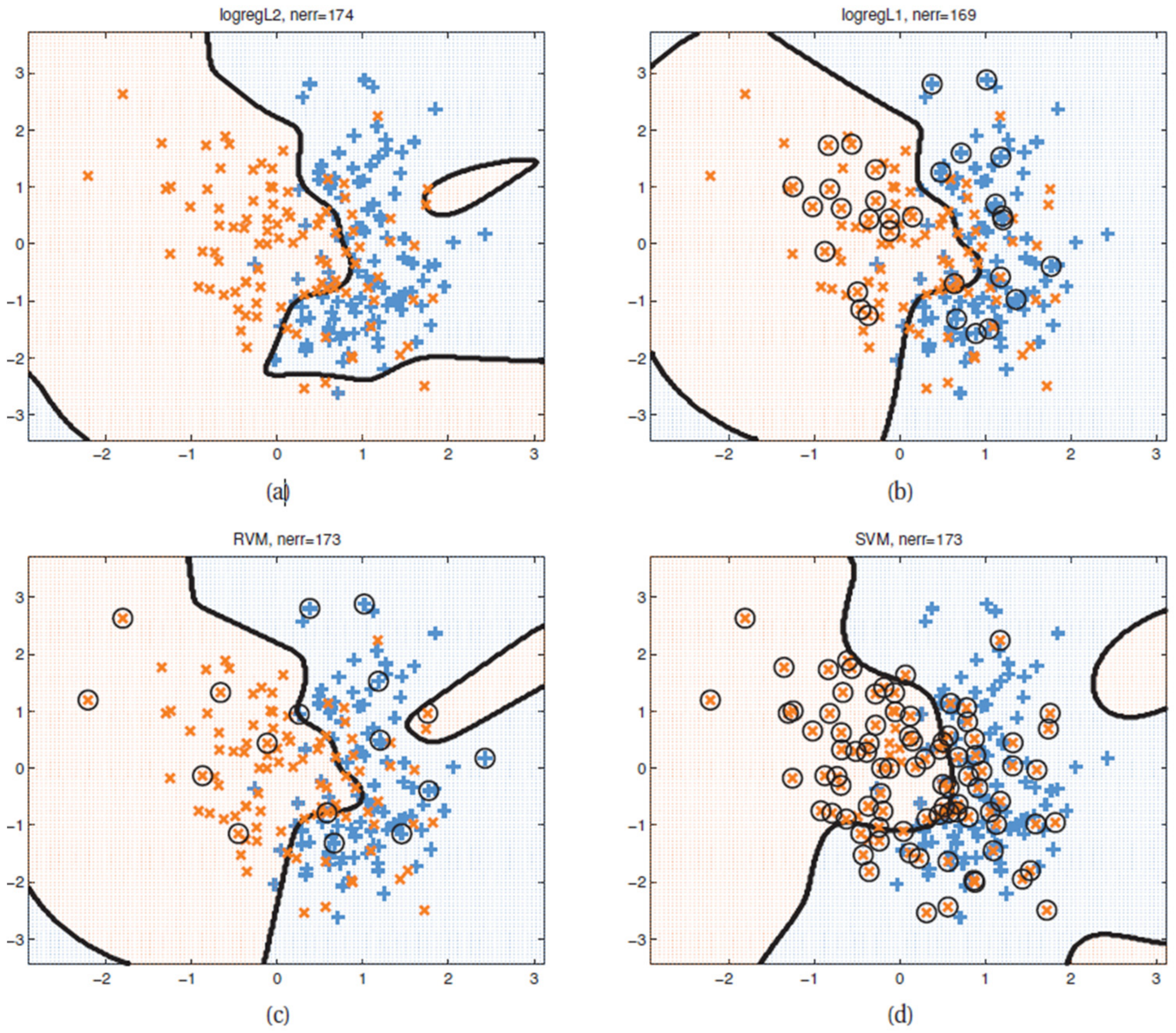
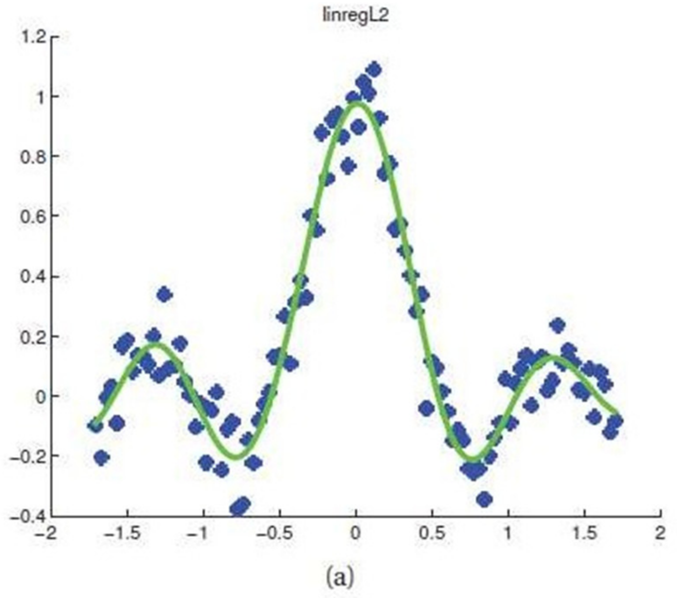
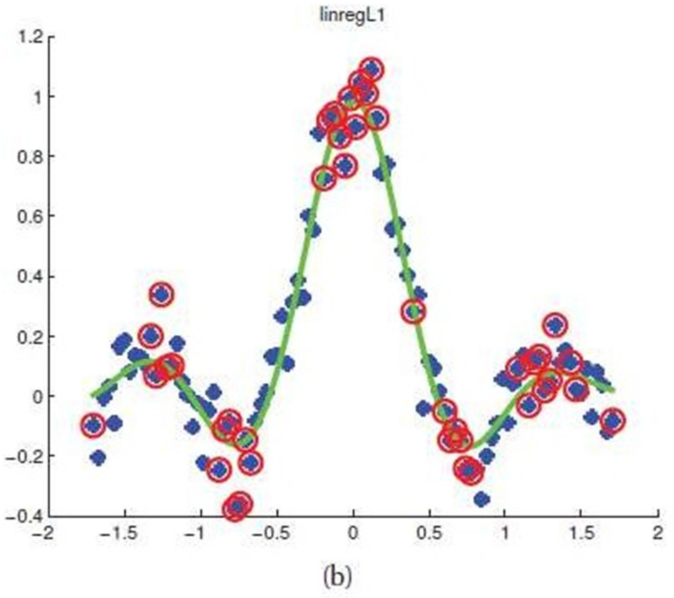


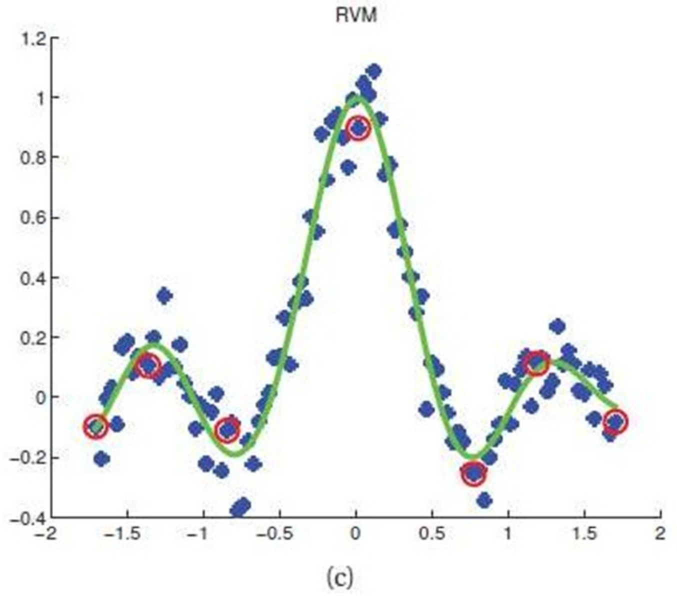
Figure 14.3 RBF basis in 1d. Left column: fitted function. Middle column: basis functions evaluated on a grid. Right column: design matrix. Top to bottom we show different bandwidths: $\tau = 0.1$, $\tau = 0.5$, $\tau = 50$. Figure generated by `linregRbfDemo`.

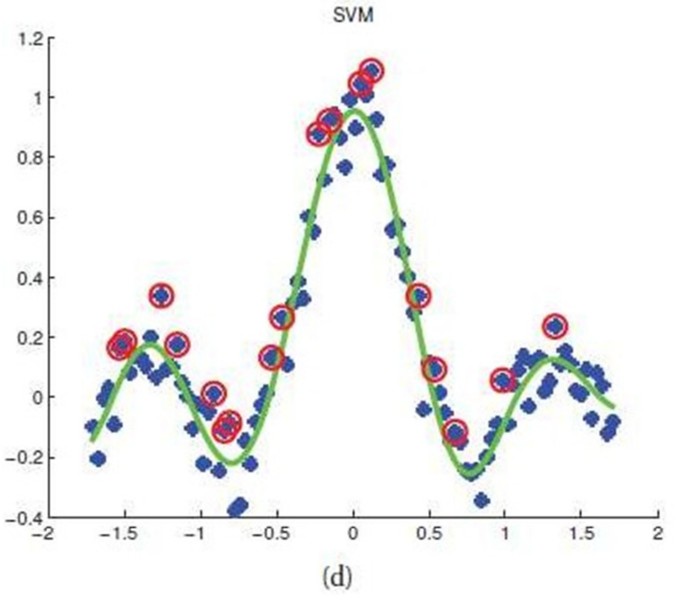


Example of non-linear binary classification using an RBF kernel with bandwidth $\sigma = 0.3$. (a) L2VM with $\lambda = 5$. (b) L1VM with $\lambda = 1$. (c) RVM. (d) SVM with $C = 1/\lambda$ chosen by cross validation. Black circles denote the support vectors









Example of kernel based regression on the noisy sinc function using an RBF kernel with bandwidth σ = 0.3. (a) L2VM with λ = 0.5. (b) L1VM with λ = 0.5. (c) RVM. (d) SVM regression with $C = 1/\lambda$ chosen by cross validation, and ϵ = 0.1. Red circles denote the retained training exemplars.

Kernelized ridge regression

- Applying the kernel trick to distance-based methods was straightforward
- It is not so obvious how to apply it to parametric models such as ridge regression
- **The primal problem**
  - Let $\mathbf{x} \in R^D$ be some feature vector, and $\mathbf{X}$ be the corresponding $N \times D$ design matrix
  - Minimize

$$J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||^2$$

  - The optimal solution is given by

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^T\mathbf{y} = (\sum_i \mathbf{x}_i\mathbf{x}_i^T + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^T\mathbf{y}$$

Kernelized ridge regression

- We can partially kernelize this, by replacing $\mathbf{X}\mathbf{X}^T$ with the Gram matrix $\mathbf{K}$
- But what about the leading $\mathbf{X}^T$ term?
- Let us define the following **dual variables**:

$$\alpha \triangleq (\mathrm{K} + \lambda \mathrm{I}_N)^{-1} \mathrm{y}$$

- Then we can rewrite the **primal variables** as follows

$$\mathrm{w} = \mathrm{X}^T \alpha = \sum_{i=1}^{N} \alpha_i \mathrm{x}_i$$

- This tells us that the solution vector is just a linear sum of the N training vectors. When we plug this in at test time to compute the predictive mean, we get

$$\hat{f}(\mathrm{x}) = \mathrm{w}^T \mathrm{x} = \sum_{i=1}^{N} \alpha_i \mathrm{x}_i^T \mathrm{x} = \sum_{i=1}^{N} \alpha_i \kappa(\mathrm{x}, \mathrm{x}_i)$$

Kernelized ridge regression

- So we have successfully kernelized ridge regression by changing from primal to dual variables
- This technique can be applied to many other linear models, such as logistic regression
- The cost of computing the dual variables $\boldsymbol{\alpha}$ is $O(N^3)$, whereas the cost of computing the primal variables $\boldsymbol{w}$ is $O(D^3)$
- However, prediction using the dual variables takes $O(ND)$ time, while prediction using the primal variables only takes $O(D)$ time

Support vector machines (SVMs)

- Consider the ℓ_2 regularized empirical risk function

$$J(\mathbf{w}, \lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda ||\mathbf{w}||^2 \qquad \hat{y}_i = \mathbf{w}^T \mathbf{x}_i + w_0$$

- If L is quadratic loss, this is equivalent to ridge regression
- We can rewrite these equations in a way that only involves inner products of the form $\mathbf{x}^T\mathbf{x}$, which we can replace by calls to a kernel function, $\kappa(\mathbf{x}, \mathbf{x})$
- This is kernelized, but not sparse
- If we replace the quadratic loss with some other loss function, we can ensure that the solution is sparse, so that predictions only depend on a subset of the training data, known as **support vectors**
- This combination of the kernel trick plus a modified loss function is known as a **support vector machine** or **SVM**

SVMs for regression

- This is a standard quadratic program in $2N + D + 1$ variables.
- The optimal solution has the form

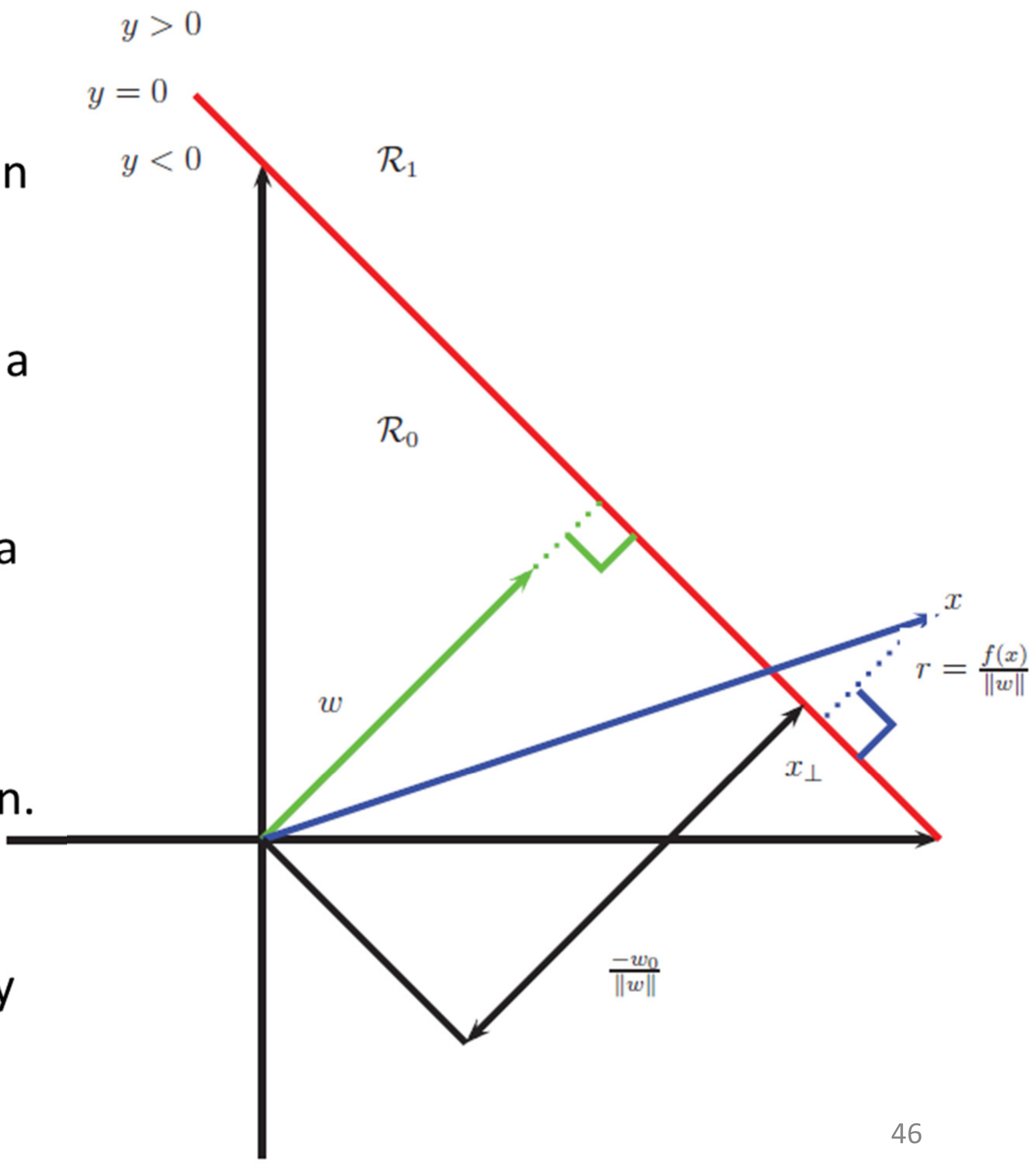
$$\hat{\mathbf{w}} = \sum_i \alpha_i \mathbf{x}_i$$

  where $\alpha_i \geq 0$

- Furthermore, it turns out that the $\boldsymbol{\alpha}$ vector is sparse, because we don't care about errors which are smaller than ϵ. The $\mathbf{x}_i$ for which $\alpha_i > 0$ are called the **support vectors**. These are points for which the errors lie on or outside the ϵ-tube

The large margin principle

Illustration of the geometry of a linear decision boundary in 2d. A point $\mathbf{x}$ is classified as belonging in decision region R_1 if $f(\mathbf{x}) > 0$, otherwise it belongs in decision region R_0; here $f(\mathbf{x})$ is known as a **discriminant function**. The decision boundary is the set of points such that $f(\mathbf{x}) = 0$. $\boldsymbol{w}$ is a vector which is perpendicular to the decision boundary. The term w_0 controls the distance of the decision boundary from the origin. The signed distance of $\mathbf{x}$ from its orthogonal projection onto the decision boundary, $\boldsymbol{x}_\perp$, is given by $f(\mathbf{x})/||\boldsymbol{w}||$.



The large margin principle

- Here, we derive the Equation form a completely different perspective.

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{||\mathbf{w}||}$$

- where r is the distance of $\mathbf{x}$ from the decision boundary whose normal vector is $\mathbf{w}$, and $\boldsymbol{x}_\perp$ is the orthogonal projection of $\mathbf{x}$ onto this boundary

$$f(\mathbf{x}) \quad = \quad \mathbf{w}^T\mathbf{x} + w_0 = (\mathbf{w}^T\mathbf{x}_\perp + w_0) + r\frac{\mathbf{w}^T\mathbf{w}}{||\mathbf{w}||}$$

- Now $f(\mathbf{x}_\perp) = 0$ so $0 = \mathbf{w}^T\mathbf{x}_\perp + w_0$
- Hence

$$f(\mathbf{x}) = r\frac{\mathbf{w}^T\mathbf{w}}{\sqrt{\mathbf{w}^T\mathbf{w}}} \qquad r = \frac{f(\mathbf{x})}{||\mathbf{w}||}$$

The large margin principle

- We would like to make this distance $r = f(\mathbf{x})/||\boldsymbol{w}||$ as large as possible
- Intuitively, the best one to pick is the one that maximizes the margin, i.e., the perpendicular distance to the closest point
- In addition, we want to ensure each point is on the correct side of the boundary, hence we want $f(\mathbf{x}_i)\, y_i > 0$.
- So our objective becomes

$$\max_{\mathbf{w}, w_0} \min_{i=1}^{N} \frac{y_i(\mathbf{w}^T \mathbf{x}_i + w_0)}{||\mathbf{w}||}$$

The large margin principle

- Our objective:

$$\max_{\mathbf{w},w_0} \min_{i=1}^{N} \frac{y_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{||\mathbf{w}||}$$

- Note that by rescaling the parameters using $\boldsymbol{w} \rightarrow k\boldsymbol{w}$ and $w_0 \rightarrow kw_0$, we do not change the distance of any point to the boundary, since the k factor cancels out when we divide by $||\boldsymbol{w}||$.
- Therefore let us define the scale factor such that $y_i f_i = 1$ for the point that is closest to the decision boundary
- We therefore want to optimize

$$\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, i = 1:N$$

- The constraint says that we want all points to be on the correct side of the decision boundary with a margin of at least 1

Soft margin constraints

- If the data is not linearly separable (even after using the kernel trick), there will be no feasible solution in which $y_i f_i \geq 1$ for all i.
- We replace the hard constraints with the **soft margin constraints** that $y_i f_i \geq 1 - \xi_i$.
- Our objective was:

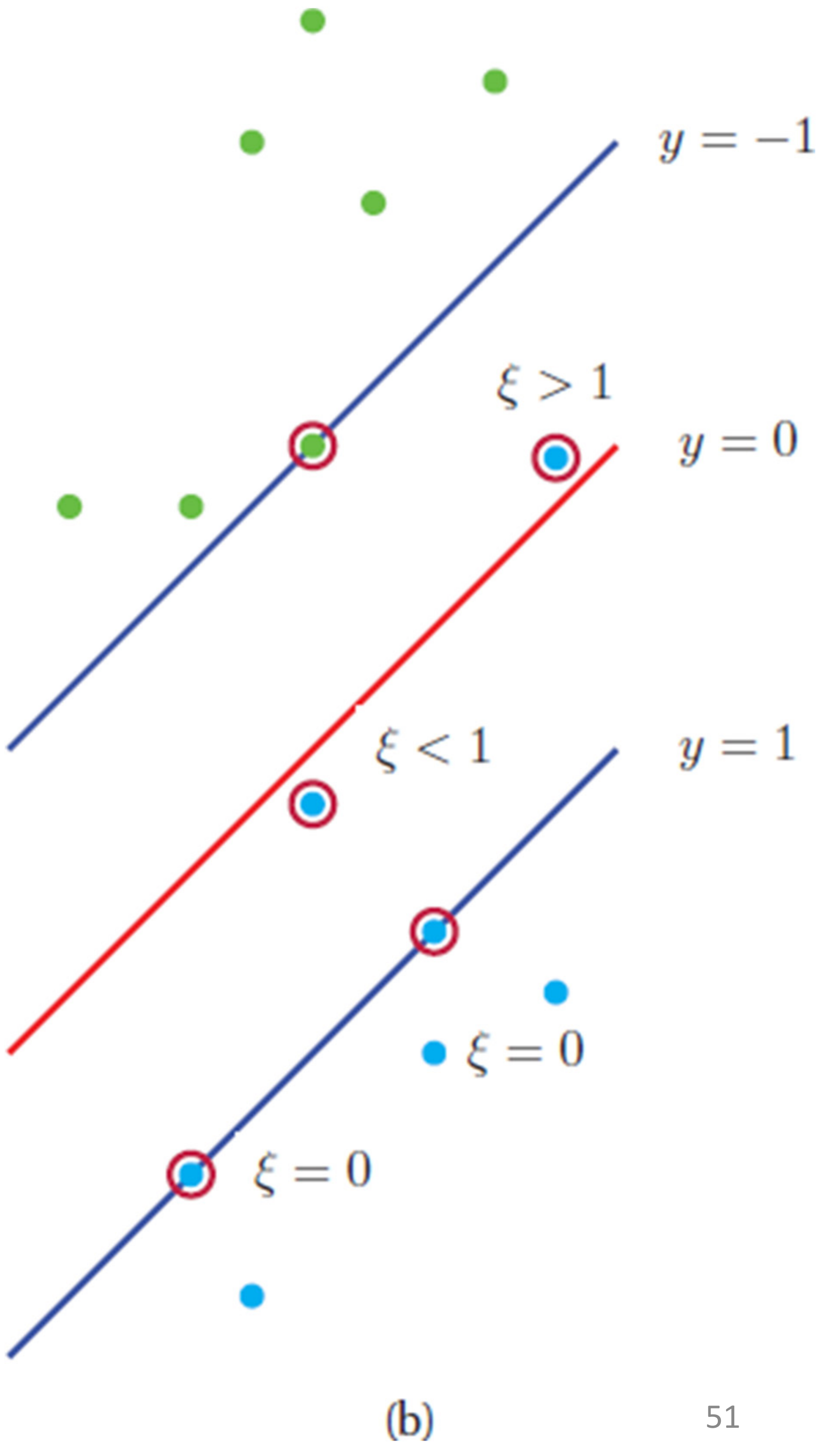
$$\min_{\mathbf{w}, w_0} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1, i = 1 : N$$

- The new objective becomes

$$\min_{\mathbf{w}, w_0, \boldsymbol{\xi}} \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N} \xi_i \quad \text{s.t.} \quad \xi_i \geq 0, \;\; y_i(\mathbf{x}_i^T \mathbf{w} + w_0) \geq 1 - \xi_i$$

Soft margin constraints

- We therefore have introduced slack variables $\xi_i \geq 0$ such that $\xi_i = 0$ if the point is on or inside the correct margin boundary, and $\xi_i = |y_i - f_i|$ otherwise
- $0 < \xi_i \leq 1$ the point lies inside the margin, but on the correct side of the decision boundary
- If $\xi_i > 1$, the point lies on the wrong side of the decision boundary
- Points with circles around them are support vectors.

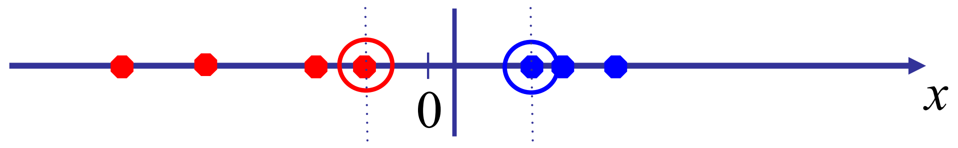


Choosing *C*

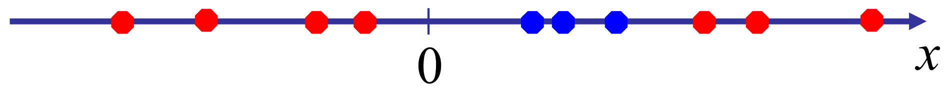
- Typically *C* is chosen by cross-validation.
- *C* interacts quite strongly with the kernel parameters.
- To choose *C* efficiently, one can develop a path following algorithm
- The basic idea is to start with λ large, so that the margin $1/||\boldsymbol{w}(\lambda)||$ is wide, and hence all points are inside of it and have $\alpha_i = 1$
- By slowly decreasing λ, a small set of points will move from inside the margin to outside, and their α_i values will change from 1 to 0, as they cease to be support vectors

Non-linear SVMs

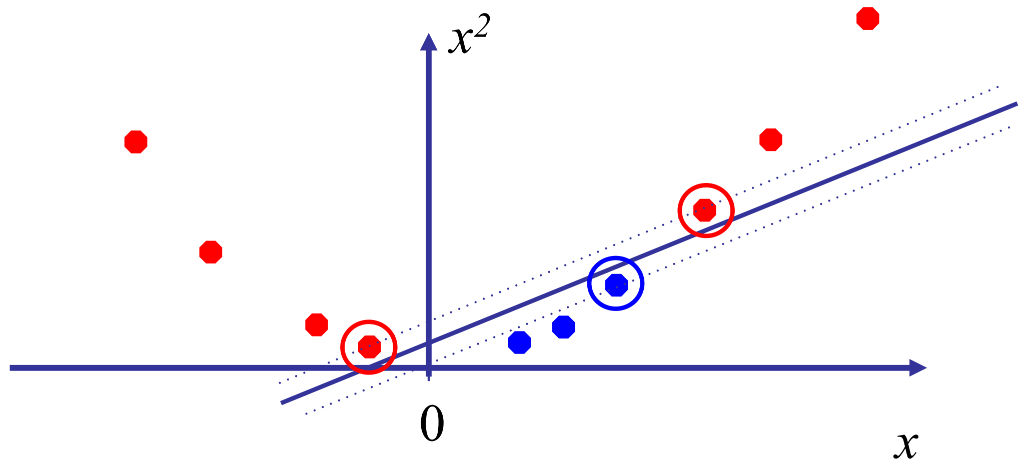
- Datasets that are linearly separable with some noise work out great:



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



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



Kernels

Definition: **(Gram matrix, kernel matrix)**

Gram matrix $G \in \mathbb{R}^{m \times m}$ of kernel k at $\{x_1, \ldots, x_m\}$:

$$\left.\begin{array}{l}\text{Given a kernel } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ \text{and a training set } \{x_1, \ldots, x_m\}\end{array}\right\} \Rightarrow G_{ij} \doteq k(x_i, x_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

Definition: **(Feature space, kernel space)**

$$\mathcal{K} \doteq span\{\phi(x) \mid x \in \mathcal{X}\} \subset \mathbb{R}^n$$

Kernel technique

Definition:

Matrix $G \in \mathbb{R}^{m \times m}$ is positive semidefinite (PSD)

$\Leftrightarrow G$ is symmetric, and $0 \leq \boldsymbol{\beta}^T G \boldsymbol{\beta} \ \forall \boldsymbol{\beta} \in \mathbb{R}^{m \times m}$

$$\left.\begin{array}{l}\text{Given a kernel } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ \text{and a training set } \{x_1, \ldots, x_m\}\end{array}\right\} \Rightarrow G_{ij} \doteq k(x_i, x_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{K}}$$

Lemma:

The Gram matrix is symmetric, PSD matrix.

Proof:

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{n \times m} \Rightarrow G = \mathbf{X}^T \mathbf{X} \in \mathbb{R}^{m \times m}$$

$$0 \leq \langle \mathbf{X}\boldsymbol{\beta}, \mathbf{X}\boldsymbol{\beta} \rangle_{\mathcal{K}} = \boldsymbol{\beta}^T G \boldsymbol{\beta}$$

The "Kernel Trick"

- **To produce linear separability in Higher Dimension, the linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:**

$$K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$$

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example:**

2-dimensional vectors $x = [x_1\ \ x_2]$; let $K(x_i,x_j) = (1 + x_i^T x_j)^2$,

Need to show that $K(x_i,x_j) = \varphi(x_i)^T\varphi(x_j)$:

$$K(x_i,x_j) = (1 + x_i^T x_j)^2,$$

$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T\, [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$$

$$= \ \varphi(x_i)^T\varphi(x_j), \quad \text{where } \varphi(x) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$

- Polynomial of power *p*: $K(\mathbf{x_i},\mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^p$

- Gaussian (radial-basis function network):

$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x_i},\mathbf{x_j}) = \tanh(\beta_0 \mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x}_i, \mathrm{x}_j) + b$$

- **Optimization techniques for finding α_i's remain the same!**

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**

  - only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**

  - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

SVM Applications

- **SVM has been used successfully in many real-world problems**

  - text (and hypertext) categorization

  - image/object classification

  - bioinformatics (Protein classification, Cancer classification)

  - hand-written character recognition

Summary:

SVM classifiers involve three key ingredients:

The kernel trick : prevent underfitting

Sparsity, large margin principle : prevent overfitting