Generalized Linear Discriminant Functions

- Decision boundaries which separate between classes may not always be linear
- The complexity of the boundaries may sometimes require the use of highly non-linear surfaces

Generalized Linear Discriminant Functions

A popular approach to generalize the concept of linear decision functions is to consider a generalized decision function as:

$$g(\mathbf{x}) = w_0 + w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_N f_N(\mathbf{x})$$

where $f_i(\mathbf{x})$, $i = 1,2,\ldots N$ are scalar functions of the pattern $\mathbf{x}$

N doesn't need to be equal to d

$f_i(\underline{x}) : \mathbb{R}^d \longrightarrow \mathbb{R}$

- Introducing $f_0(\mathbf{x}) = 1$ we get:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{N} w_i f_i(\mathbf{x}) = \sum_{i=0}^{N} w_i f_i(\mathbf{x}) = \mathbf{w}'^T \mathbf{x}'$$

where $\mathbf{w}' = [w_0, w_1, \ldots, w_N]^T$ and $\mathbf{x}' = [f_0(\mathbf{x}), f_1(\mathbf{x}), \ldots, f_N(\mathbf{x})]$

- This latter representation of $g(\mathbf{x})$ implies that any decision function defined by

$$g(\mathbf{x}) = w_0 + w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_N f_N(\mathbf{x})$$

  - can be treated as linear in the $(N + 1)$ dimensional space $(N + 1 > n)$
  - $g(\mathbf{x})$ maintains its non-linearity characteristics in the pattern space

- A commonly used generalized decision function is $g(\mathbf{x})$ for which $f_i(\mathbf{x})$ are polynomials

$$g(\mathbf{x})=\mathbf{w'}^T\mathbf{x'}$$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- A special case is a quadratic decision functions for a 2-dimensional feature space

$$g(\mathbf{x}) = \mathbf{w}'^T\mathbf{x}' = w_0 + w_1x_1^2 + w_2x_1x_2 + w_3x_2^2 + w_4x_1 + w_5x_2$$

where

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & w_4 & w_5 \end{bmatrix}^T$$

$$\mathbf{x}' = \begin{bmatrix} 1 & x_1^2 & x_1x_2 & x_2^2 & x_1 & x_2 \end{bmatrix}^T$$

- For patterns **x** in d-dimensional space, the most general quadratic decision function is given by:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j .$$

since $x_i x_j = x_j x_i$, one can assume that $w_{ij} = w_{ji}$ without loss of generality

- Linear discriminants have $d+1$ parameters,
- Quadratic discriminant has $d+1+d(d+1)/2$ parameters
  It produce more complicated separating surfaces
- This is the total number of weights which are the free parameters of the problem

  - If for example d = 3, the weight vector is 10-dimensional
  - If for example d = 10, the weight vector is 56-dimensional

$$10 + 1 + \frac{10 \times 11}{2} = 66$$

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j.$$

Quadratic Discriminant Functions

The commonly used quadratic decision function can be represented as the general *d*- dimensional quadratic surface:

$$g(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

where the matrix $\mathbf{A} = [a_{ij}]$, the vector $\mathbf{b} = [b_1, b_2, \ldots, b_d]^T$ and c, dependon the weights w_{ii}, w_{ij}, w_i of equation

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j.$$

- Introducing $f_0(\mathbf{x}) = 1$ we get:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{N} w_i f_i(\mathbf{x}) = \sum_{i=0}^{N} w_i f_i(\mathbf{x}) = \mathbf{w}^T \mathbf{x}'$$

where $\mathbf{w}' = [w_0, w_1, \ldots, w_N]^T$ and $\mathbf{x}' = [f_0(\mathbf{x}), f_1(\mathbf{x}), \ldots, f_N(\mathbf{x})]$

- This latter representation of $g(\mathbf{x})$ implies that any decision function defined by

$$g(\mathbf{x}) = w_0 + w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_N f_N(\mathbf{x})$$

  - can be treated as linear in the $(N + 1)$ dimensional space $(N + 1 > n)$
  - $g(\mathbf{x})$ maintains its non-linearity characteristics in the pattern space

- A commonly used generalized decision function is $g(\mathbf{x})$ for which $f_i(\mathbf{x})$ are polynomials

$$g(\mathbf{x})=\mathbf{w'}^T\mathbf{x'}$$

- A special case is a quadratic decision functions for a 2-dimensional feature space

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}' = w_0 + w_1x_1^2 + w_2x_1x_2 + w_3x_2^2 + w_4x_1 + w_5x_2$$

where

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & w_4 & w_5 \end{bmatrix}^T$$

$$\mathbf{x}' = \begin{bmatrix} 1 & x_1^2 & x_1x_2 & x_2^2 & x_1 & x_2 \end{bmatrix}^T$$

- For patterns **x** in d-dimensional space, the most general quadratic decision function is given by:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j .$$

since $x_i x_j = x_j x_i$, one can assume that $w_{ij} = w_{ji}$ without loss of generality

- Linear discriminants haved+1 parameters,
- Quadratic discriminant has d+1+d(d+1)/2 parameters
  It produce more complicated separating surfaces
- This is the total number of weights which are the free parameters of the problem

  - If for example d = 3, the weight vector is 10-dimensional
  - If for example d = 10, the weight vector is 56-dimensional

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j.$$

Quadratic Discriminant Functions

The commonly used quadratic decision function can be represented as the general *d*- dimensional quadratic surface:

$$g(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

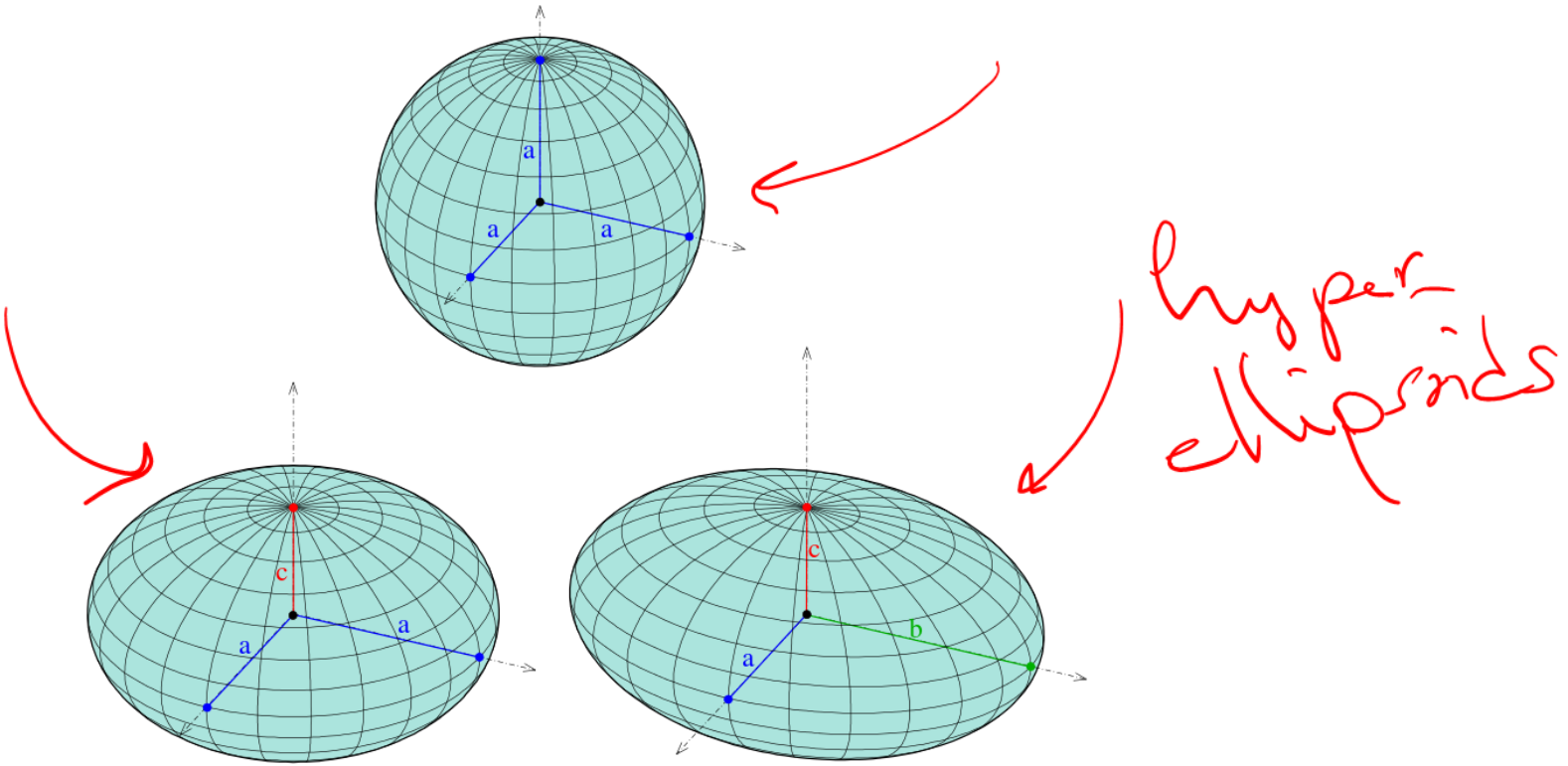
where the matrix $\mathbf{A} = [a_{ij}]$, the vector $\mathbf{b} = [b_1, b_2, \ldots, b_d]^T$ and c, dependon the weights w_{ii}, w_{ij}, w_i of equation

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j.$$

Quadratic Discriminant Functions

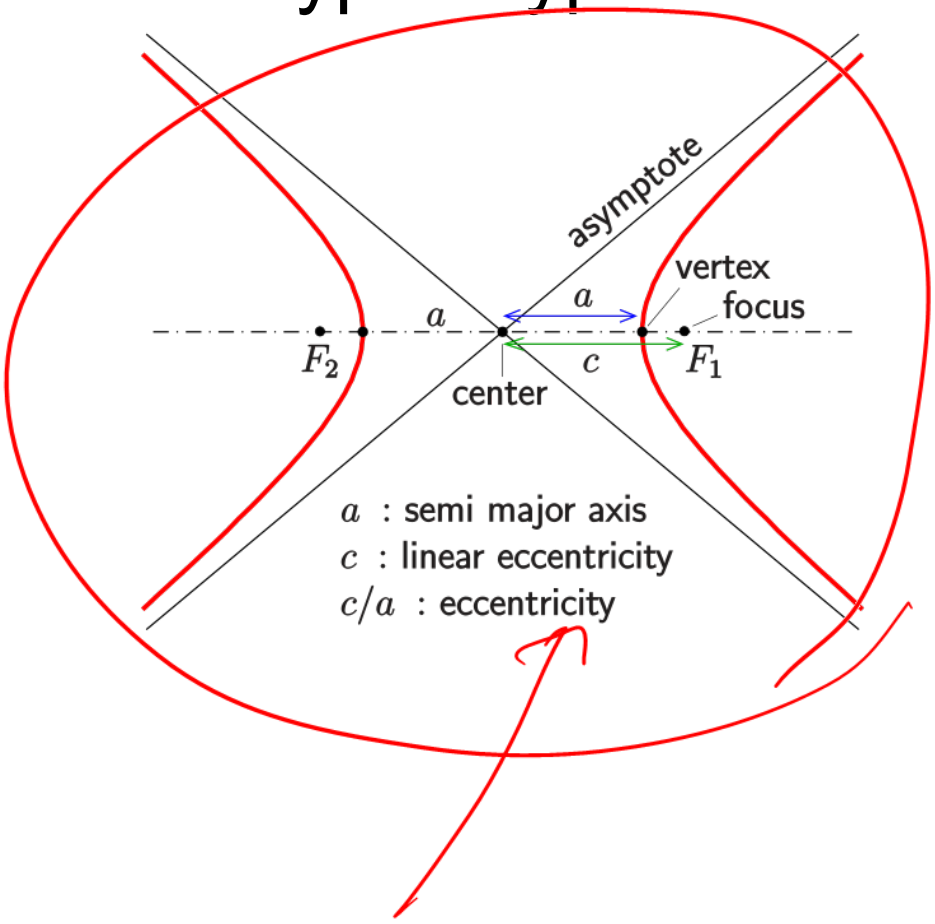
If A is positive definite then the decision function is a hyper-ellipsoid with axes in the directions of the eigenvectors of **A**

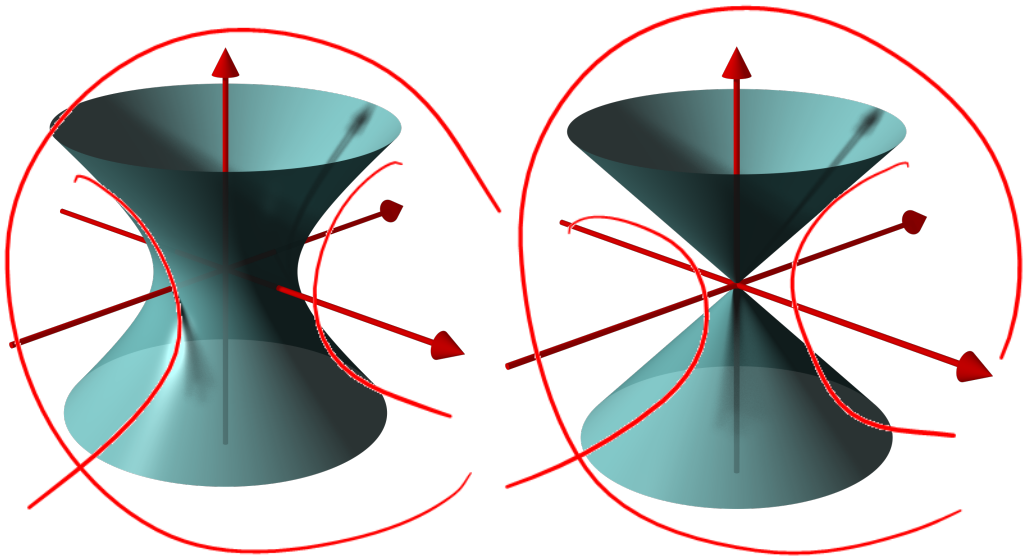
In particular: if $\mathbf{A} = \mathbf{I}_d$ (Identity), the decision function is simply the *d*-dimensional hypersphere

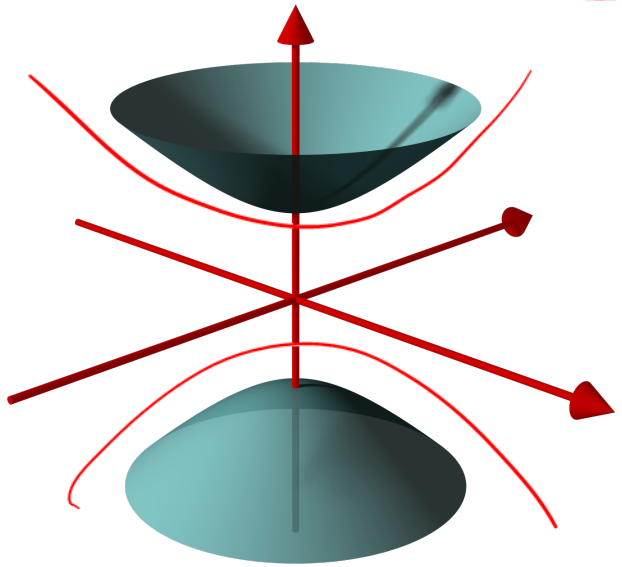


Quadratic Discriminant Functions

If **A** is indefinite, the decision function describes a hyper-hyperboloid







Quadratic Discriminant Functions

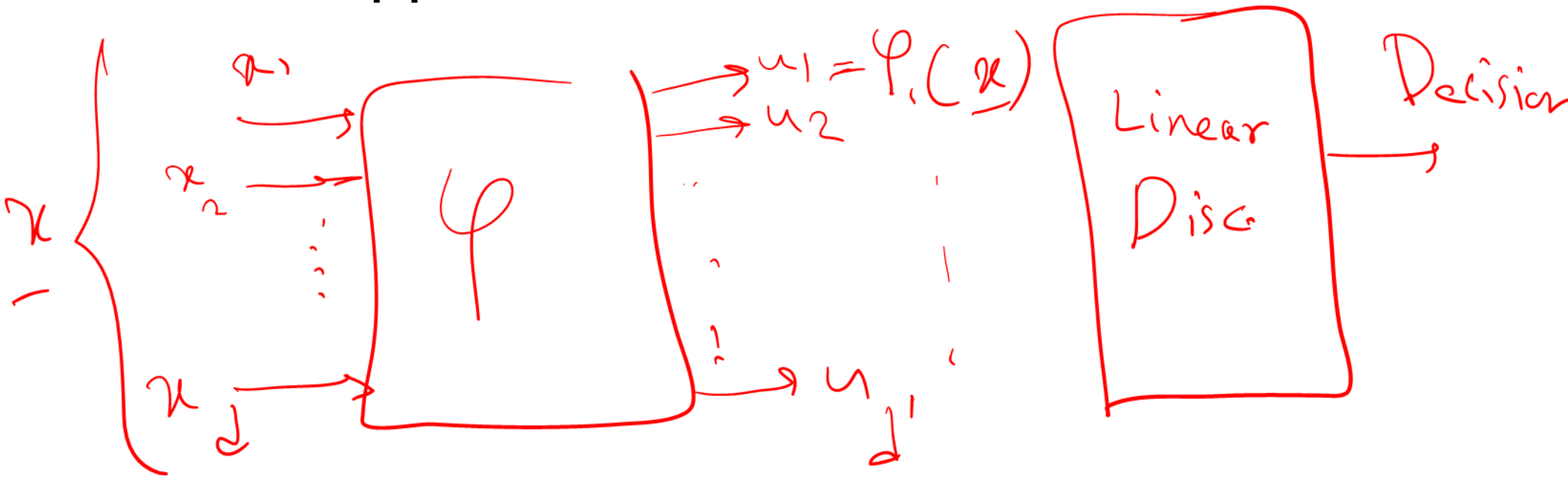
In conclusion: it is only the matrix **A** which determines the shape and characteristics of the decision function

Using Nonlinear Discriminants for Classification

- Simply use **x'** space as the new feature space
- Hopefully patterns are linearly separable in this space
- Use any training method to find a linear classifier in the new space
- Linear decision boundaries in this space are equivalent to nonlinear decision boundaries in the original space.

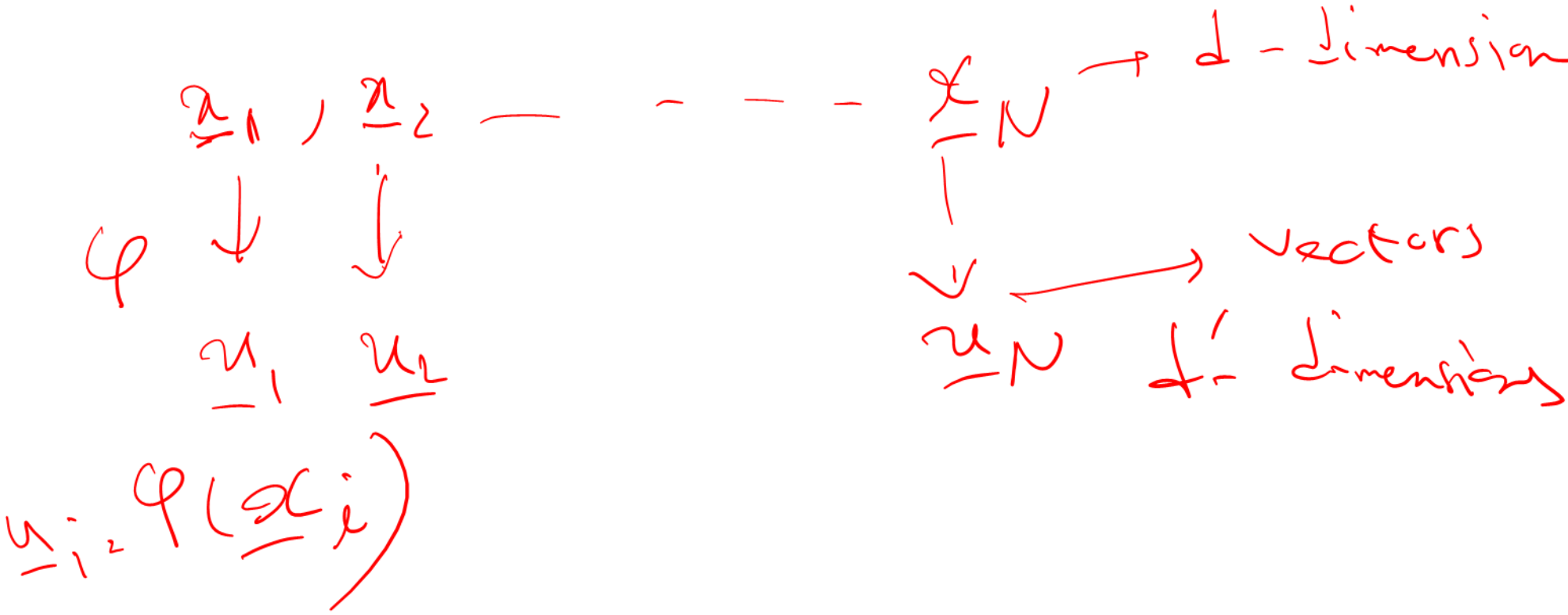
Using Nonlinear Discriminants for Classification

- This approach is called the Φ-Machine:



Using Nonlinear Discriminants for Classification

- This approach is called the Φ-Machine:



More xamples of nonlinear mappings

- Gaussian or Radial Basis Functions (RBFs)

$$x'_j = e^{\frac{-\|\mathbf{x}-\mathbf{x}_j\|^2}{2\sigma^2}}$$

- The j^{th} new feature x'_j of a vector **x** is calculated as a measure of its similarity with training vector x_j

What are the disadvantage of this approach?

- The number of parameters to be calibrated may go out of hand
- We do not know what an optimal feature set is: should we use quadratic, cubic, Gaussian, or all of those functions to have nonlinear features?

What are the disadvantage of this approach?

- There should be a more automatic way of nonlinear transformation of vectors
- Two examples: Support Vector Machines and Neural Networks

Optimization Using Lagrange Multipliers

- Problem: find an optimum of $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x})=0$
- Solution: set up a Lagrangian function $L(\mathbf{x}, \lambda)$:

$$L(\underline{x}, \lambda) = f(\underline{x}) + \lambda g(\underline{x})$$

- When the constraint $g(\mathbf{x}^*)=0$ is satisfied, $L(\mathbf{x}^*, \lambda) = f(\underline{x}^*)$

Optimization Using Lagrange Multipliers

- The Lagrange method:

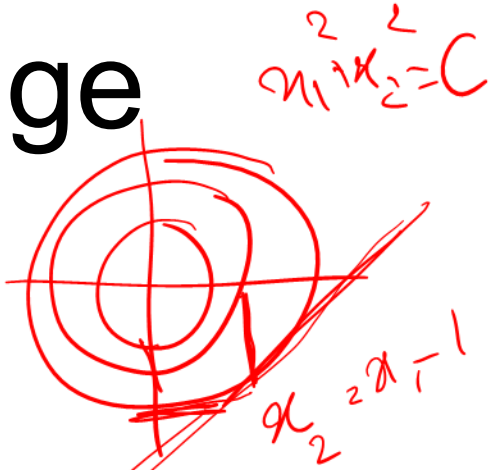
$$\nabla_{\mathbf{x},\lambda} L(\mathbf{x},\lambda) = \mathbf{0} \Rightarrow$$

$$\nabla_{\underline{x},\lambda}\left(f(\underline{x}) + \lambda\, g(\underline{x})\right) = \underline{0}$$

$$\begin{cases} \nabla_{\underline{x}} f(\underline{x}) + \lambda \nabla_{\underline{x}} g(\underline{x}) = 0 \\ \qquad \text{Generally not equal to zero} \\ \dfrac{\partial L(\underline{x},\lambda)}{\partial \lambda} = 0 \Rightarrow g(\underline{x}) = 0 \end{cases}$$

Optimization Using Lagrange Multipliers

- Example: Let $f(\mathbf{x})=||\mathbf{x}||^2$ $\mathbf{x}=[x_1\ x_2]^T$ subject to: $x_2-x_1=-1$



$x_2-x_1+1=0$

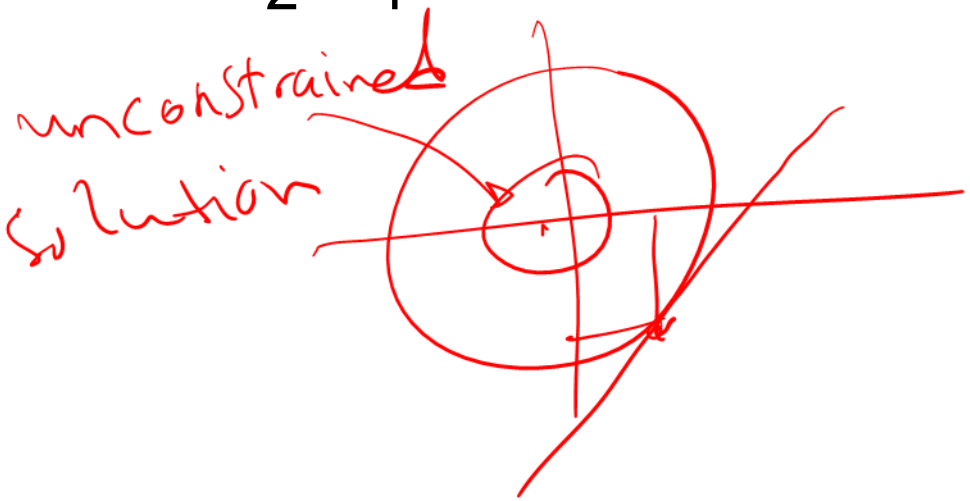
$f(\underline{x}) = x_1^2+x_2^2$ $\quad g(\underline{x}) = x_2-x_1+1=0$

$L(\underline{x},\lambda) = x_1^2+x_2^2+\lambda(x_2-x_1+1)$

$$\begin{cases} 2x_1-\lambda=0 \\ 2x_2+\lambda=0 \\ x_2-x_1+1=0 \end{cases} \qquad \begin{cases} x_1+x_2=0 \\ x_2-x_1+1=0 \end{cases} \quad \begin{matrix} x_2=-1/2 \\ x_1=1/2 \end{matrix}$$

Optimization Using Lagrange Multipliers

- Example: Let $f(\mathbf{x})=||\mathbf{x}||^2$ $\mathbf{x}=[x_1\ x_2]^T$ subject to: $x_2-x_1=-1$



Lagrange Optimization with Multiple Constraints

- Problem: find an optimum of $f(\mathbf{x})$ subject to the constraints $g_i(\mathbf{x})=0$, $i = 1,2,\ldots, R$
- $\mathbf{x}$ has d dimensions and $R < d$
- Solution: set up a Lagrangian function $L(\mathbf{x}, \lambda)$:

$$L(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \sum_{i=1}^{R} \lambda_i g_i(\underline{x})$$

- When for all i the constraints $g_i(\mathbf{x})=0$ are satisfied, $L(\mathbf{x}, \lambda) = f(\underline{x})$

Lagrange Optimization with Multiple Constraints

- Then $\nabla_{\mathbf{x},\underline{\lambda}} L(\mathbf{x},\underline{\lambda}) = \mathbf{0} \Rightarrow$ $\nabla_{\underline{x},\underline{\lambda}} \left(f(\underline{x}) + \lambda g(\underline{x}) \right) = 0$

$\nabla_{\underline{x}} f(\underline{x}) + \lambda \nabla_{\underline{x}} g(\underline{x}) = 0$

generalized not equal to zero

$\nabla_{\underline{\lambda}} L(\underline{x}, \underline{\lambda}) = 0 \Rightarrow g_i(\underline{x}) = 0$

Lagrange Optimization with Inequality Constraints

- Problem: find the minimum of $f(\mathbf{x})$ subject to the constraint $h(\mathbf{x}) \geq 0$
- $\mathbf{x}$ has d dimensions
- Solution: set up a Lagrangian function $L(\mathbf{x}, \lambda)$:

$$L(\underline{x}, \lambda) = f(\underline{x}) - \lambda h(\underline{x})$$

Lagrange Optimization with Inequality Constraints

- Case 1: when the constraint $h(\mathbf{x}) \geq 0$ is satisfied when $\mathbf{x}=\mathbf{x}^*$, the point that minimizes $f(\mathbf{x})$, i.e. when the solution to constrained and unconstrained problems are the same, one can solve the problem with $\lambda=0$ and $L(\mathbf{x}, \lambda)=f(\mathbf{x})$

Lagrange Optimization with Inequality Constraints

- Case 1: when the constraint $h(\mathbf{x}) > 0$ is satisfied for $\mathbf{x}=\mathbf{x}^*$, the point that minimizes $f(\mathbf{x})$, i.e. when the solution to constrained and unconstrained problems are the same, one can solve the problem with $\lambda=0$ and $L(\mathbf{x}, \lambda)=f(\mathbf{x})$

Lagrange Optimization with Inequality Constraints

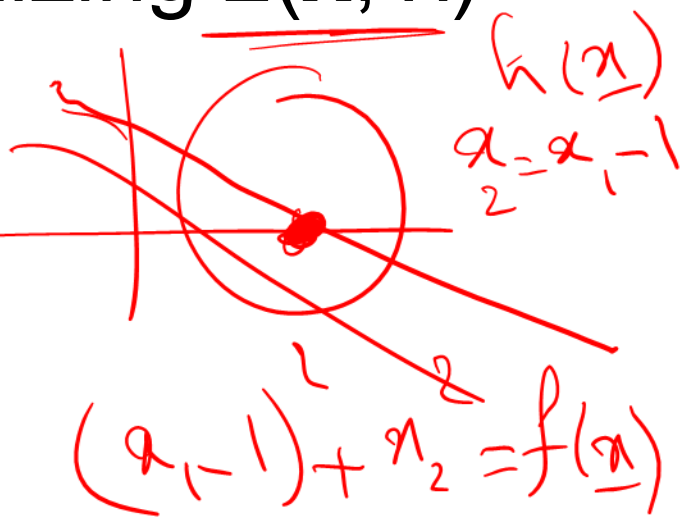
- Case 2: constrained minimum is on $h(\mathbf{x}) = 0$ one has to consider minimizing $L(\mathbf{x}, \lambda)$ with $\lambda \neq 0$, i.e.

minimize with respect to $\underline{x}$ and maximize with respect to λ

$$\nabla_{\mathbf{x}} L(\mathbf{x},\lambda) = \mathbf{0} \Rightarrow$$

$$\nabla_{\underline{x}} f(\underline{x}) - \lambda \nabla_{\underline{x}} h(\underline{x}) = 0$$

$$\frac{\partial L(\underline{x},\lambda)}{\partial \lambda} = 0 \rightarrow \text{non-zero } \lambda \quad \lambda \neq 0$$



Lagrange Optimization with Inequality Constraints

- Summary
- $L(\mathbf{x}, \lambda)=f(\mathbf{x})- \lambda h(\mathbf{x})$
- In Case 1: $\lambda=0$, $h(\mathbf{x}^*)>0$
- In Case 2: $\lambda>0$, $h(\mathbf{x}^*)=0$
- Therefore, in both cases, $\lambda h(\mathbf{x}^*)=0$

Solution

Karush-Kuhn-Tucker Conditions

- $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda h(\mathbf{x})$
- Find min of L with respect to $\mathbf{x}$
- Find max of L with respect to λ
  - $h(\mathbf{x}) \geq 0$
  - $\lambda \geq 0$
  - $\lambda h(\mathbf{x}) = 0$

KKT

$$\nabla_{\underline{x}} f(\underline{x}) - \lambda \nabla_{\underline{x}} h(\underline{x}) = 0$$

$$\frac{\partial L(\underline{x}, \lambda)}{\partial \lambda} = 0$$

Multiple Constraints

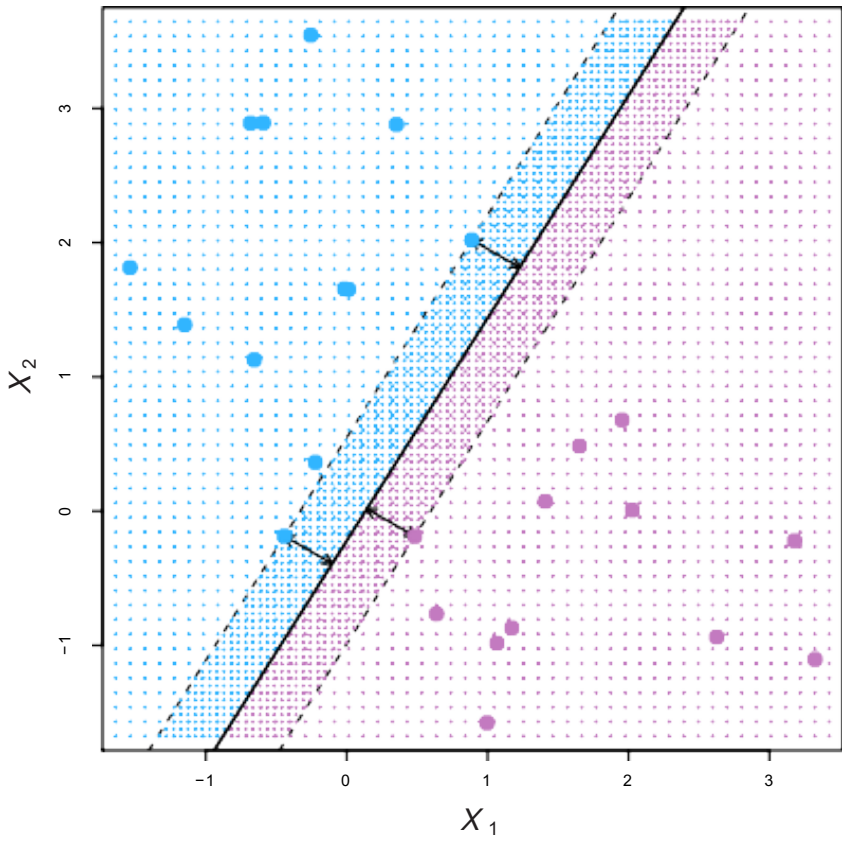
- Assume that there are multiple equality and inequality constraints
- Min $f(\mathbf{x})$
  subject to $g_i(\mathbf{x})=0$
  and $h_j(\mathbf{x}) \geq 0$

$$L(\mathbf{x},\underline{\lambda},\underline{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{R} \lambda_i g_i(\mathbf{x}) - \sum_{j=1}^{R'} \mu_j h_j(\mathbf{x})$$

Similar. Require: $\mu_j \geq 0$ and $\mu_j h_j(\mathbf{x}) = 0$.

Maximum Margin Classifiers

- Linear Discriminants formulated based on optimization

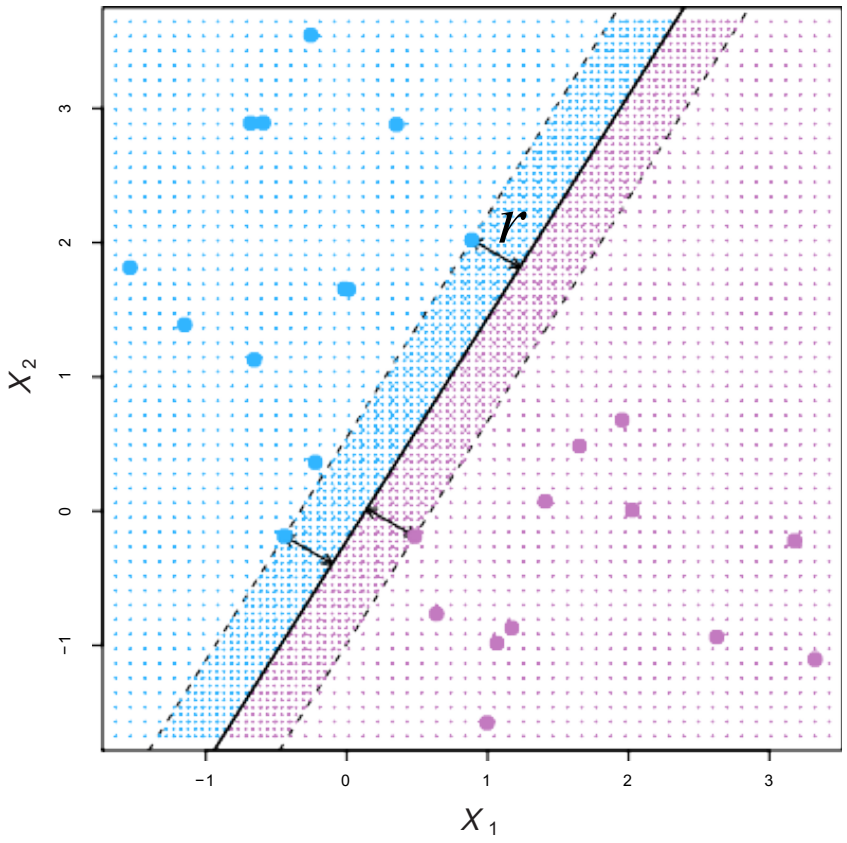


$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

- Remember
  - $z(i)$=1 if x(i) in ω_1
  - $z(i)$=-1 if x(i) in ω_2
- The decision rule becomes:
  - $z(i)\mathbf{w'}^T\mathbf{x'}(i) > 0$ implies
  - $z(i)\mathbf{w'}^T\mathbf{x'}(i) < 0$ implies
  - In short, $g(z(i)\mathbf{x}(i))>0$ implies

Maximum Margin Classifiers

- Linear Discriminants formulated based on optimization



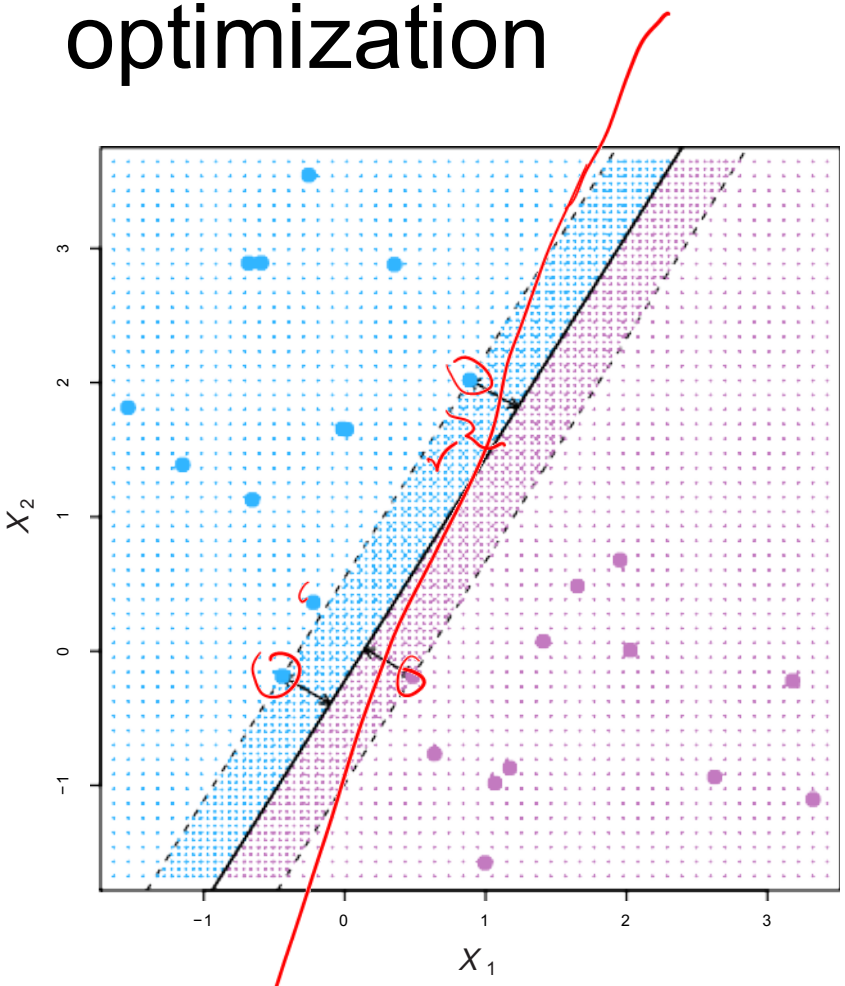
$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

- Absolute distance:

$$r_i = \left|\frac{g(\mathbf{x}_i)}{\|\mathbf{w}\|}\right| = \frac{z_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{\|\mathbf{w}\|} > 0$$

Maximum Margin Classifiers

- Linear Discriminants formulated based on optimization



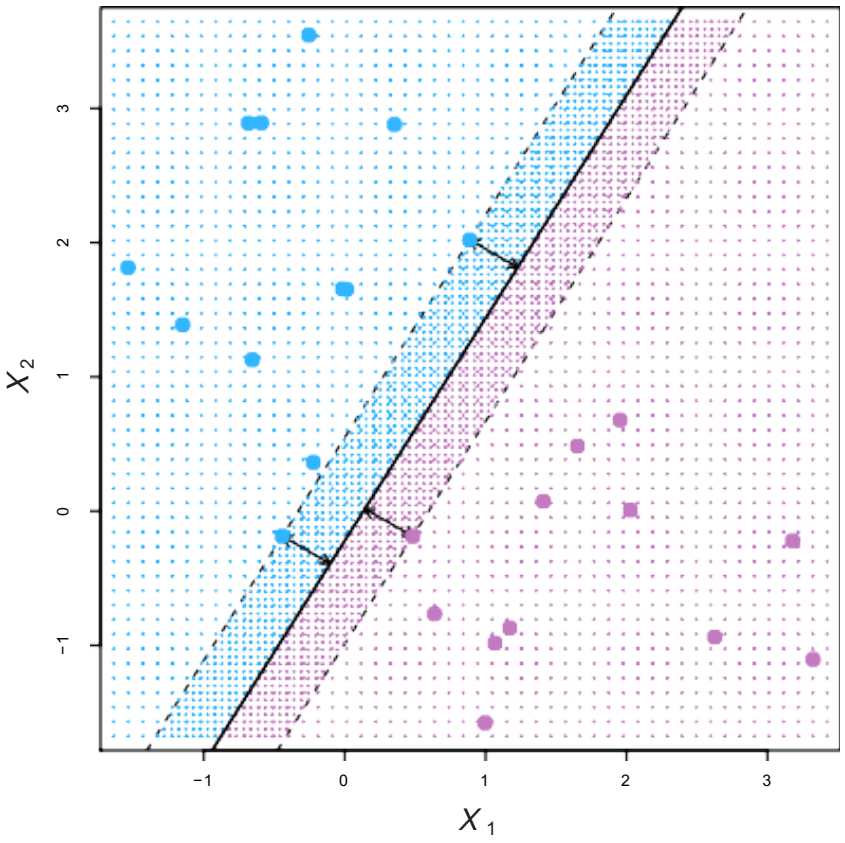
- We want

$$r_i \geq M > 0, \forall i$$

- Where M is the *maximum possible margin*.

Maximum Margin Classifiers

- Linear Discriminants formulated based on optimization



- We want the maximum possible margin *M* for which:

$$r_i \geq M > 0, \forall i$$

- i.e.

$$\frac{z_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{\|\mathbf{w}\|} \geq M > 0, \forall i$$

Maximum Margin Classifiers

- If $\mathbf{w}^*$ is a solution, $\alpha\,\mathbf{w}^*$ is also a solution
- To find a unique $\mathbf{w}$, we impose the following constraint

$$\|\mathbf{w}\|.M=1$$

- Therefore,

$$\frac{z_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{\|\mathbf{w}\|} \geq M > 0, \forall i$$

$$\Rightarrow z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq$$

Maximum Margin Classifiers

- This means maximize *M such that*

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, \forall i$$

constraints

$$M = \frac{1}{\|\mathbf{w}\|}$$

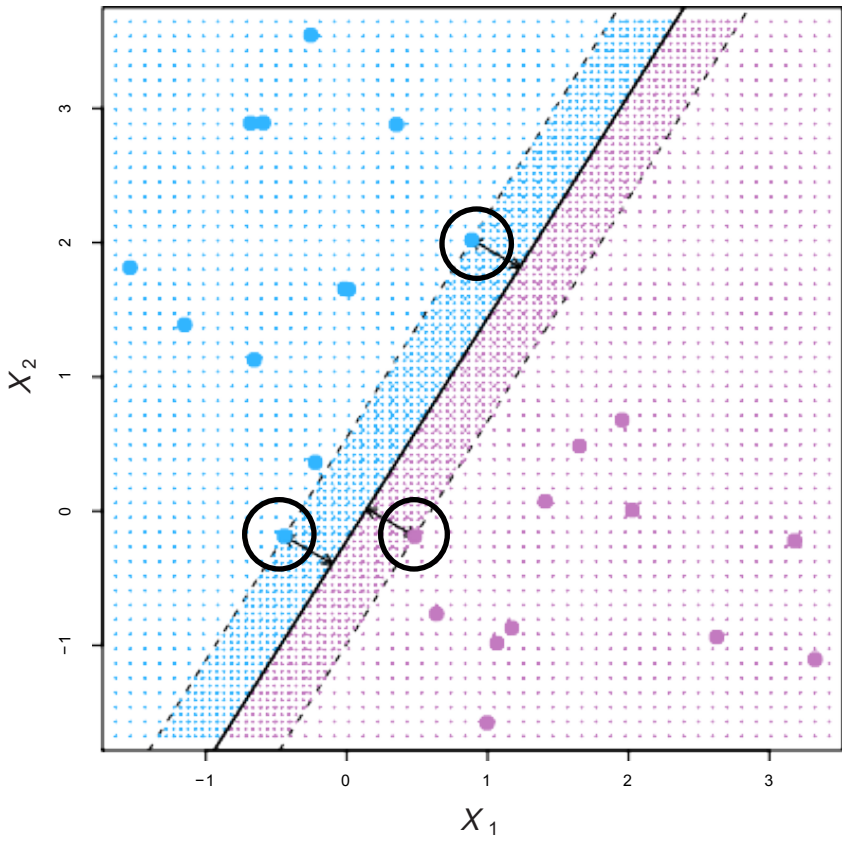
- This can be formulated as a constrained minimization problem

$$\min \|\mathbf{w}\|$$

$$\text{s.t. } z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, \forall i \quad i = 1, \ldots, N$$

Maximum Margin Classifiers

- Linear Discriminants formulated based on optimization



- After optimization, some vectors satisfy

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) = 1, \forall i \in S$$

or

$$\frac{z_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{\|\mathbf{w}\|} = M, \forall i \in S$$

- The set of all of those vectors is denoted S and they are called support vectors that are at the edge of the margin

Lagrange Optimization for Maximum Margin Classifiers

- Minimize

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to constraint:

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 \geq 0, \forall i$$

Lagrange Optimization for Maximum Margin Classifiers

- The Lagrangian becomes:

$$L(\mathbf{w}, w_0, \underline{\lambda}) = \frac{1}{2}\|\underline{w}\|^2 - \sum_{i=1}^{N} \lambda_i [z_i(\underline{w}^T x_i + w_0) - 1]$$

- which must be minimized with respect to **w** and maximized with respect to λ (Primal Problem)

Lagrange Optimization for Maximum Margin Classifiers

$\frac{1}{2}(w_1^2 + w_2^2 + \cdots + w_d^2)$

- The optimization problem is easier to solve in the Dual Form. Calculate the Gradients:

$$\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \underline{\lambda}) = \nabla_{\underline{w}} \left(\frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \lambda_i [z_i(\underline{w}^T \underline{x}_i + w_0) - 1] \right)$$

$$\Rightarrow \quad \underline{w} - \sum_{i=1}^{N} \lambda_i z_i \underline{x}_i = 0$$

$$\Rightarrow \quad \underline{w}^* = \sum_{i=1}^{N} \lambda_i z_i \underline{x}_i$$

Lagrange Optimization for Maximum Margin Classifiers

- The optimization problem is easier to solve in the Dual Form. Calculate the Gradients:

$$\frac{\partial L(\mathbf{w}, w_0, \underline{\lambda})}{\partial w_0} = -\sum_{i=1}^{N} \lambda_i z_i = 0$$

$$\sum_{i=1}^{N} \lambda_i z_i = 0$$

Lagrange Optimization for Maximum Margin Classifiers

- In summary, from calculating the Gradients we have:

$$\mathbf{w}^* = \sum_{i=1}^{N} \lambda_i z_i \mathbf{x}_i \qquad \sum_{i=1}^{N} \lambda_i z_i = 0$$

- Plugging them in the Primal Lagrangian

$$L(\mathbf{w}, w_0, \underline{\lambda}) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 - \sum_{i=1}^{N} \lambda_i \left[z_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right]$$

$$\lambda_i \geq 0, \forall i$$

yields the Dual Lagrangian

Lagrange Optimization for Maximum Margin Classifiers

$$\mathbf{w}^* = \sum_{i=1}^{N} \lambda_i z_i \mathbf{x}_i \qquad \sum_{i=1}^{N} \lambda_i z_i = 0$$

$$L(\mathbf{w}, w_0, \underline{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left[z_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right]$$

$$\lambda_i \geq 0, \forall i$$

$$L_D(\underline{\lambda}) = \frac{1}{2} \left(\sum_{i=1}^{N} \lambda_i z_i x_i^T \right) \left(\sum_{j=1}^{N} \lambda_i z_i x_j \right) - \frac{1}{2} w^T w$$

Lagrange Optimization for Maximum Margin Classifiers

$$\mathbf{w}^* = \sum_{i=1}^{N} \lambda_i z_i \mathbf{x}_i \qquad \sum_{i=1}^{N} \lambda_i z_i = 0$$

$$L(\mathbf{w}, w_0, \underline{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left[z_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right]$$

$$\lambda_i \geq 0, \forall i$$

$$L_D(\underline{\lambda}) = -\sum_{i=1}^{N} \lambda_i z_i \mathbf{w}^T \mathbf{x}_i - \sum_{i=1}^{N} \lambda_i z_i w_0 + \sum_{i=1}^{N} \lambda_i$$

$$= -\sum_{i=1}^{N} \lambda_i z_i \left(\sum_{j=1}^{N} \lambda_j z_j \mathbf{x}_j^T \right) \mathbf{x}_i$$

Lagrange Optimization for Maximum Margin Classifiers

$$\mathbf{w}^* = \sum_{i=1}^{N} \lambda_i z_i \mathbf{x}_i \quad \sum_{i=1}^{N} \lambda_i z_i = 0$$

$$L(\mathbf{w}, w_0, \underline{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left[z_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right]$$

$$\lambda_i \geq 0, \forall i$$

$$L_D(\underline{\lambda}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j z_i z_j \mathbf{x}_i^T \mathbf{x}_j + \sum \lambda_i$$

maximize with respect to $\underline{\lambda}$

Lagrange Optimization for Maximum Margin Classifiers

- The Dual Problem has to be solved only with respect to $\underline{\lambda}$. Minimize:

$$L_D(\underline{\lambda}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}\lambda_i$$

with:

$$\sum_{i=1}^{N}\lambda_i z_i = 0$$

$$\lambda_i \geq 0, \forall i$$

$$\mathbf{w}^* = \sum_{i=1}^{N}\lambda_i z_i \mathbf{x}_i$$

Optimization Techniques

- For simple problems, the optimization can be solved algebraically.
- For complex problemss, a variety of techniques are available, including *Quadratic Programming (QP).*

Non-Separable Patterns: Support Vector Classifier (SVC)

- What if patterns are non-separable by a linear discriminant? The optimization has no solution.
- Can we reformulate the optimization problem so that it has a solution for the non-separable case as well?
- The answer is yes, by introducing some extra cost for misclassified patterns.

Soft Margin

Non-Separable Patterns: Support Vector Classifier (SVC)

- We introduce *slack variables* ξ_i
- The slack variable ξ_i tells us where the i^{th} observation is located, relative to the hyperplane and relative to the margin.
  - If $\xi_i = 0$ then the i^{th} observation is on the correct side of the margin

Non-Separable Patterns: Support Vector Classifier (SVC)

- The slack variable ξ_i tells us where the i^{th} observation is located, relative to the hyperplane and relative to the margin.
  - If $\xi_i > 0$ then the i^{th} observation is on the wrong side of the margin, and we say that the i^{th} observation has *violated* the margin.

Non-Separable Patterns: Support Vector Classifier (SVC)

- The slack variable ξ_i tells us where the i^{th} observation is located, relative to the hyperplane and relative to the margin.
  - If $\xi_i > 1$ then it is on the wrong side of the hyperplane.
- This means

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

Support Vector Classifier

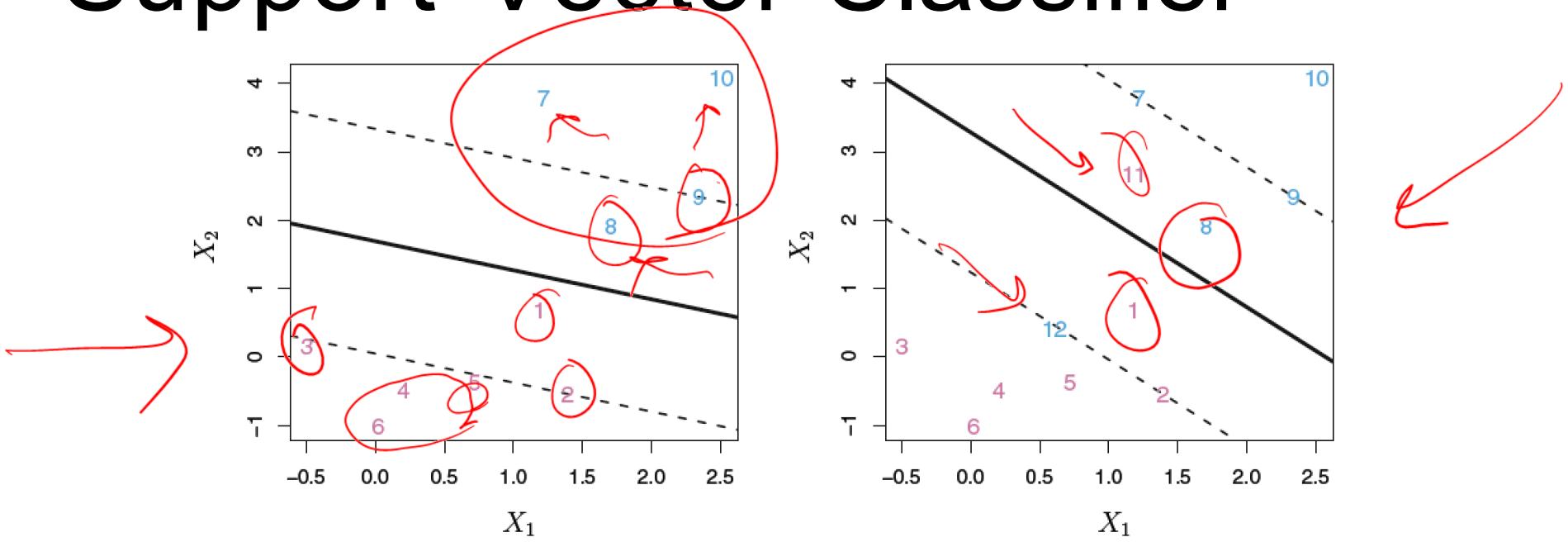


FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* $3, 4, 5,$ *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation 1 is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12*. These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

Non-Separable Patterns: Support Vector Classifier (SVC)

- We formulate the optimization problem by considering a *budget B* for the sum of slack variables:

$$\sum_{i=1}^{N} \xi_i \leq B$$

Non-Separable Patterns: Support Vector Classifier (SVC)

- We formulate the optimization problem by considering a *budget B* for the sum of slack variables.
- Minimize

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to

$$B - \sum_{i=1}^{N} \xi_i \geq 0$$

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

Soft margin constraint

Non-Separable Patterns: Support Vector Classifier (SVC)

- One can rewrite the optimization problem using a multiplier C to remove the constraint on slack variables:
- Minimize

$$\frac{1}{2}\|w\|^2 - C\left(B - \sum_{i=1}^{N} \xi_i\right)$$

$$= \frac{1}{2}\|w\|^2 - CB + C\sum_{i=1}^{N} \xi_i$$

subject to

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

Non-Separable Patterns: Support Vector Classifier (SVC)

- Because *CB* is a constant, the problem can be reformulated as:
- Minimize

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

subject to

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

Non-Separable Patterns: Support Vector Classifier (SVC)

- Using Lagrange Multipliers, the problem becomes optimizing

$$L(\mathbf{w}, w_0, \underline{\lambda}, \underline{\mu}) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\left[z_i(\underline{w}^T x_i + w_0) - 1 + \xi_i\right] - \sum_{i=1}^{N}\mu_i\xi_i$$

Non-Separable Patterns: Support Vector Classifier (SVC)

- It can be expressed in the dual form:

$$L_D(\underline{\lambda}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}\lambda_i$$

- Except that the constraint $\lambda_i \geq 0, \forall i$ becomes

$$0 \leq \lambda_i \leq C, \forall i$$

Try to prove it

Details of Optimization Problem

- B bounds the sum of the ξ_i's and C penalizes them, and so B determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.

Details of Optimization Problem

- We can think of B as a budget for the amount that the margin can be violated by the N observations and C as the cost of violations.
- If $B = 0$ (i.e. $C = \infty$) then there is no budget for violations to the margin, i.e. the cost of violation is very high, so all ξ_i's = 0, which leads to the maximal margin hyperplane optimization problem

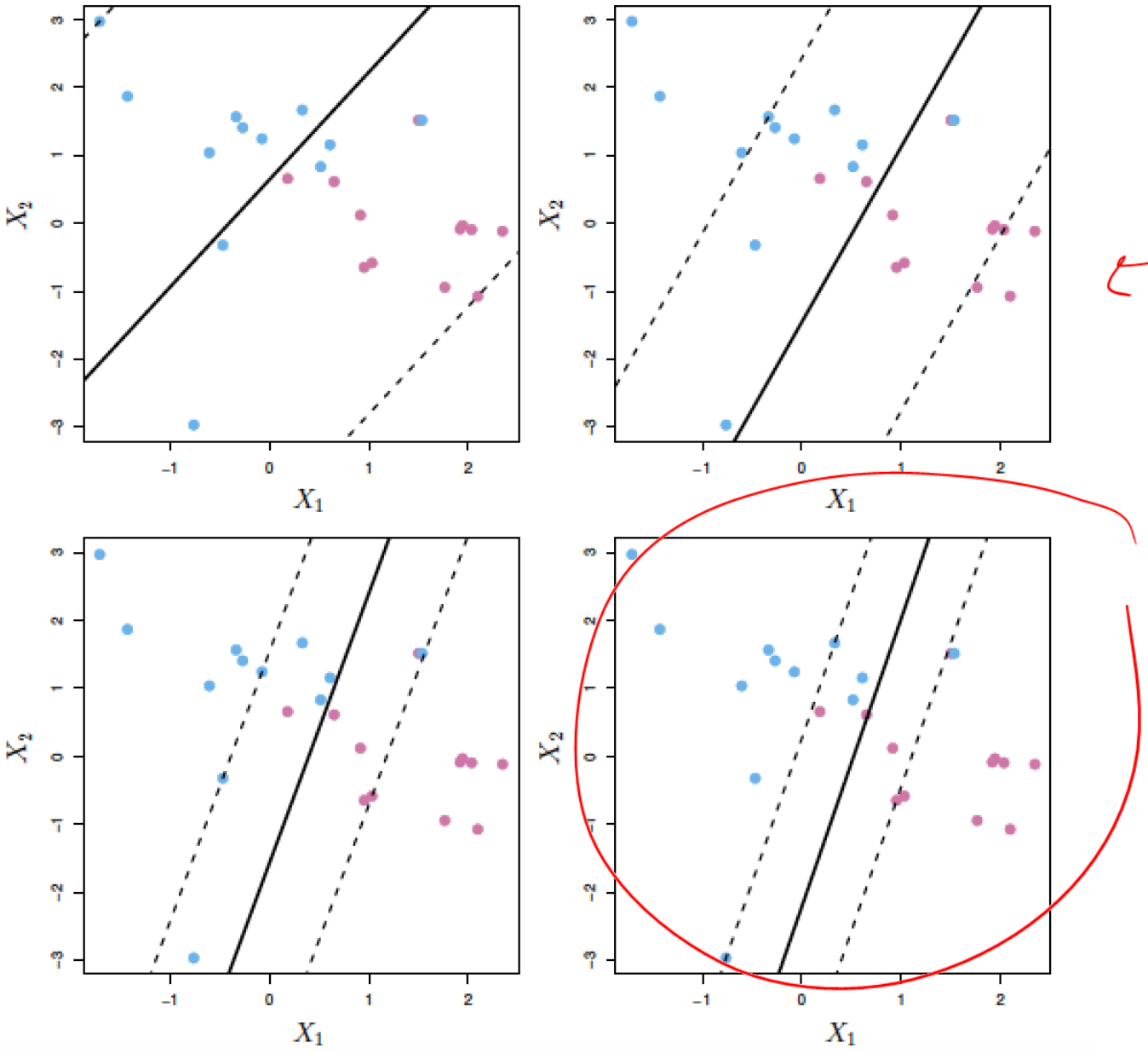
Details of Optimization Problem

For $B > 0$ no more than B observations can be on the wrong side of the hyperplane, because if an observation is on the wrong side of the hyperplane then $\xi_i > 1$, and the optimization problem requires that the sum of ξ_i's be less than B.

Details of Optimization Problem

- As the budget *B* increases (or when the cost *C* decreases), we become more tolerant of violations to the margin, and so the margin will widen.
- Conversely, as *C* increases (i.e. *B* decreases), we become less tolerant of violations to the margin and so the margin narrows.
    - An example is shown in the next slide.

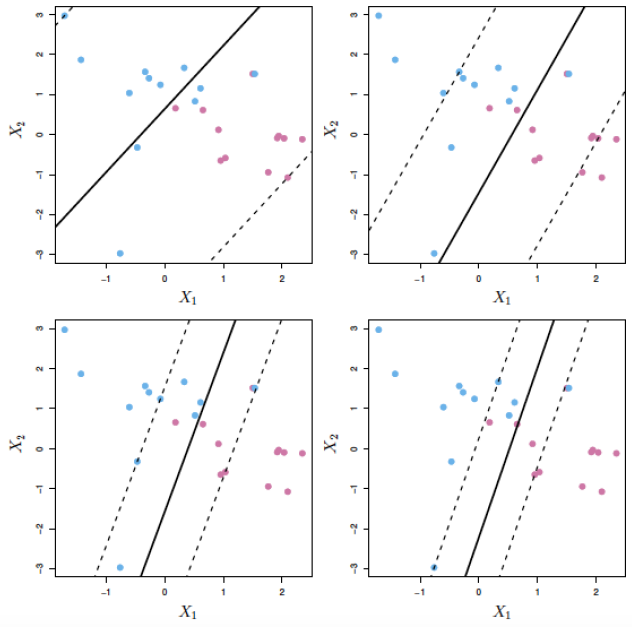
C is a regularization parameter



C is a regularization parameter

A support vector classifier was fit using four different values of the tuning parameter *C*.

- The smallest value of *C* was used in the top left panel, and larger values were used in the top right, bottom left, and bottom right panels.



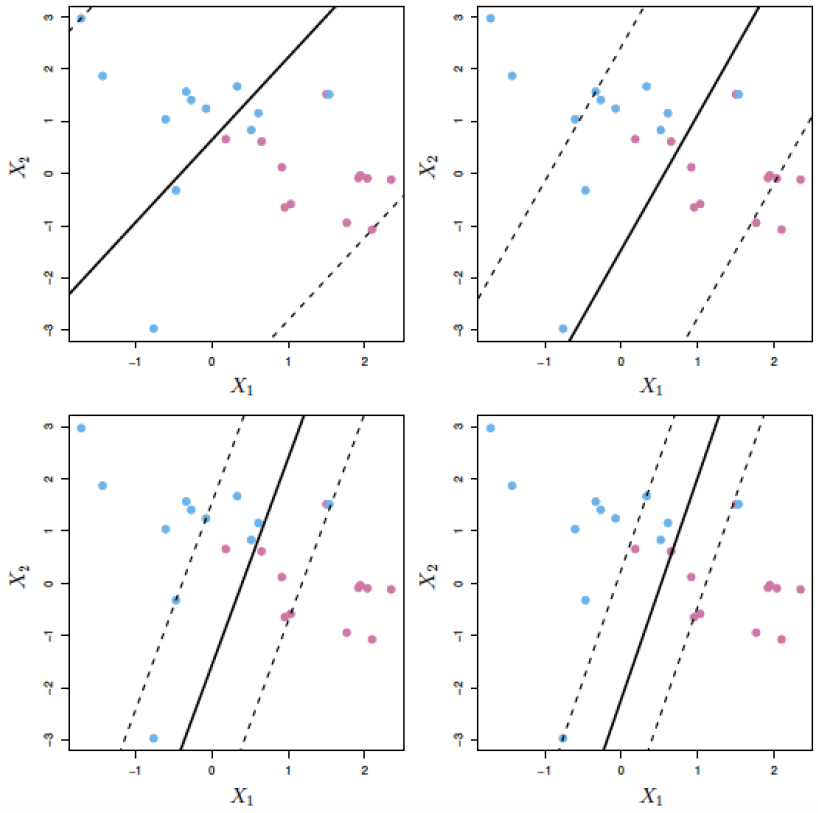
C is a regularization parameter

A support vector classifier was fit using four different values of the tuning parameter *C*.

- When *C* is small, there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large.
- As *C* increases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

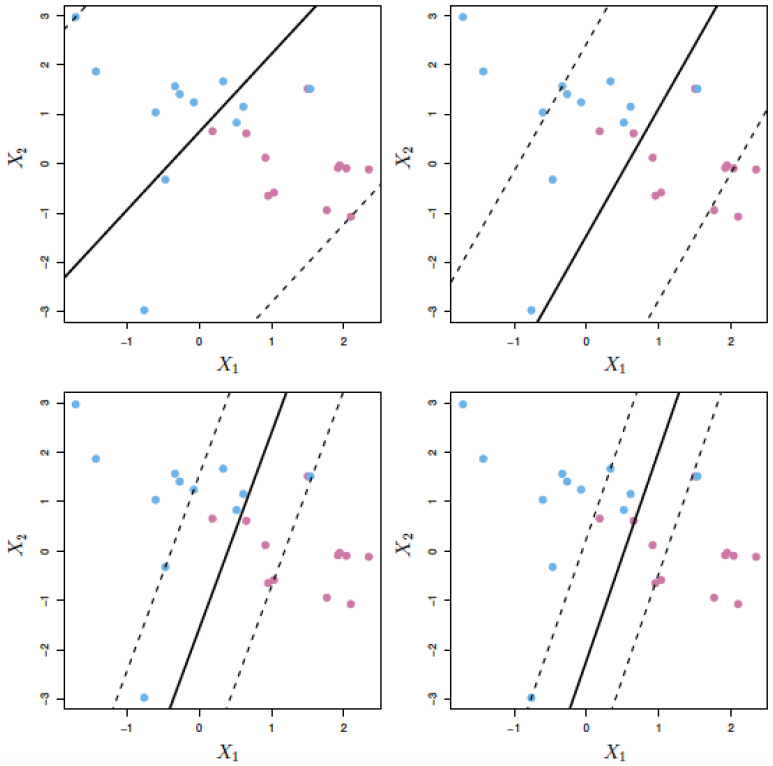
C is a regularization parameter

- *C* is treated as a tuning parameter generally chosen via cross-validation.
- *C* controls the bias-variance trade-off of the statistical learning technique.



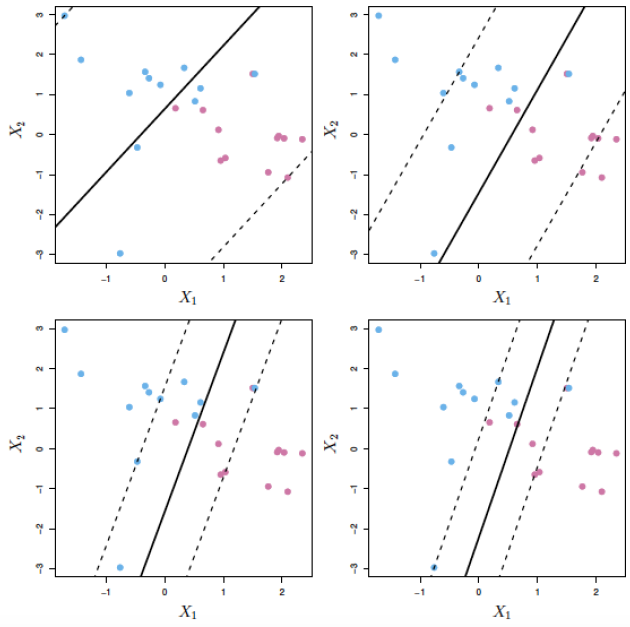
C is a regularization parameter

- When *C* is large, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.



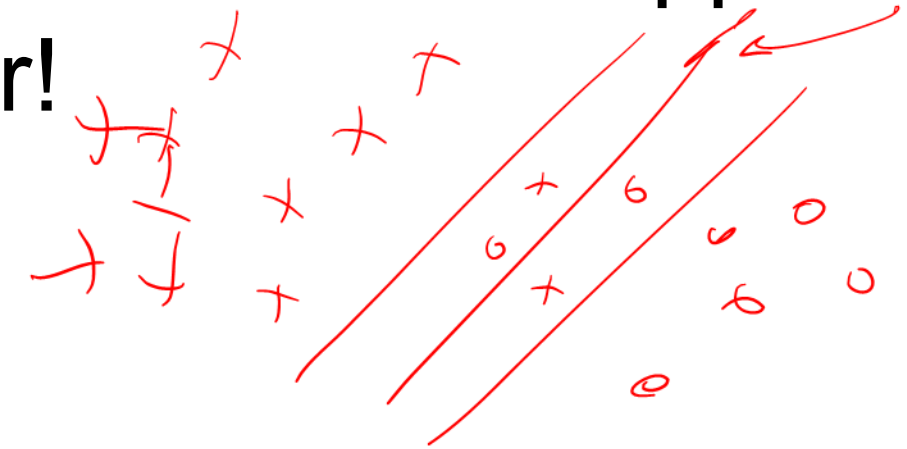
C is a regularization parameter

- On the other hand, when *C* is smaller, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.



Support Vectors

- Only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained.
- In other words, an observation that lies strictly on the correct side of the margin does not affect the support vector classifier!



Support Vectors

- Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin.

Support Vectors

- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors . These observations do affect the support vector classifier.

Support Vectors

The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane.

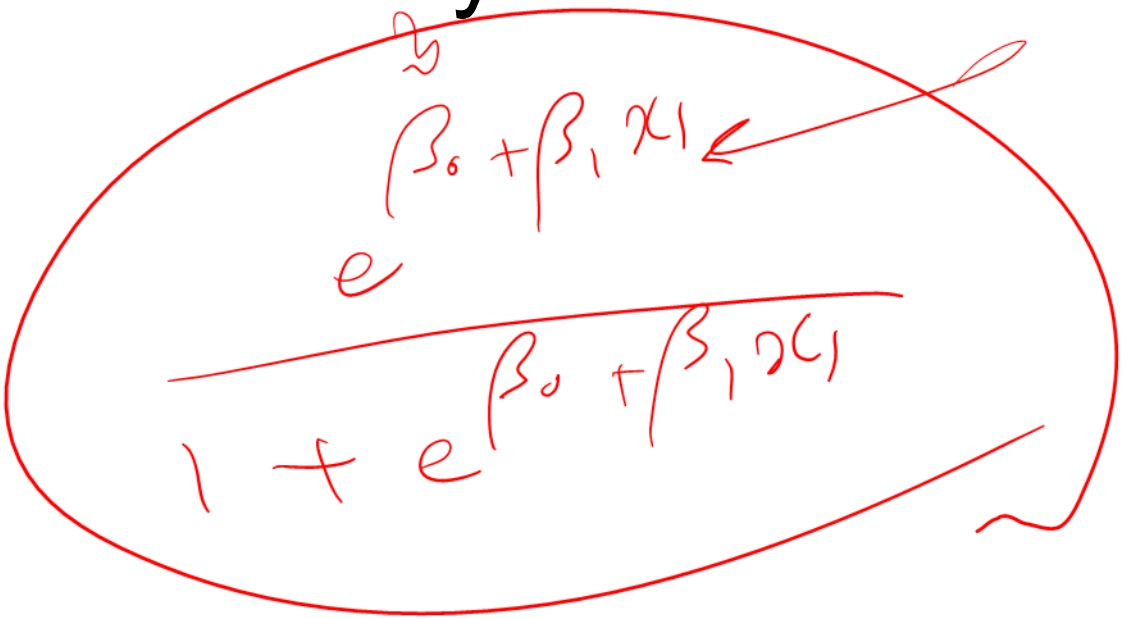
(Robustness to outliers)

Comparison with Bayesian LDA

- This property is distinct from some of the other classification methods that we have seen, such as linear discriminant analysis.
- The LDA classification rule depends on the mean of all of the observations within each class, as well as the within-class covariance matrix computed using all of the observations.

Comparison with LR

In contrast, logistic regression, unlike LDA, has very low sensitivity to observations far from the decision boundary. In fact we will see that the support vector classifier and logistic regression are closely related.



Regularization Formulation of SVC

- Let us consider the SVC optimization again

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

subject to

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

Regularization Formulation of SVC

- The constraint

$$z_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

- Can be rewritten as

$$\xi_i \geq 1 - z_i(\mathbf{w}^T\mathbf{x}_i + w_0), \forall i$$

$$\xi_i \geq 0, \forall i$$

- Together,

$$\xi_i = \max\left[0, 1 - z_i(\mathbf{w}^T\mathbf{x}_i + w_0)\right], \forall i$$

Regularization Formulation of SVC

- The SVC optimization problem becomes minimizing

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\max\left[0, 1 - z_i(\mathbf{w}^T\mathbf{x}_i + w_0)\right]$$

or

$$J(\mathbf{w}) = \sum_{i=1}^{N}\max\left[0, 1 - z_i(\mathbf{w}^T\mathbf{x}_i + w_0)\right] + \alpha\|\mathbf{w}\|^2$$

- Here, $\sum_{i=1}^{N}\max\left[0, 1 - z_i(\mathbf{w}^T\mathbf{x}_i + w_0)\right]$ is the loss function and $\|\mathbf{w}\|^2$ is L2 (Ridge) penalty.

loss + penalty form

hinge loss

L1 Penalized Support Vector Machines

- One can use L1 (Lasso Penalty) instead of Ridge Penalty in the optimization

$$J(\mathbf{w}) = \sum_{i=1}^{N} \max\left[0, 1 - z_i(\mathbf{w}^T \mathbf{x}_i + w_0)\right] + \alpha \left\|\mathbf{w}\right\|_1$$

- L1 Penalized SVMs can be used to perform classification and variable selection simultaneously
  - It provides a *sparse classification model*

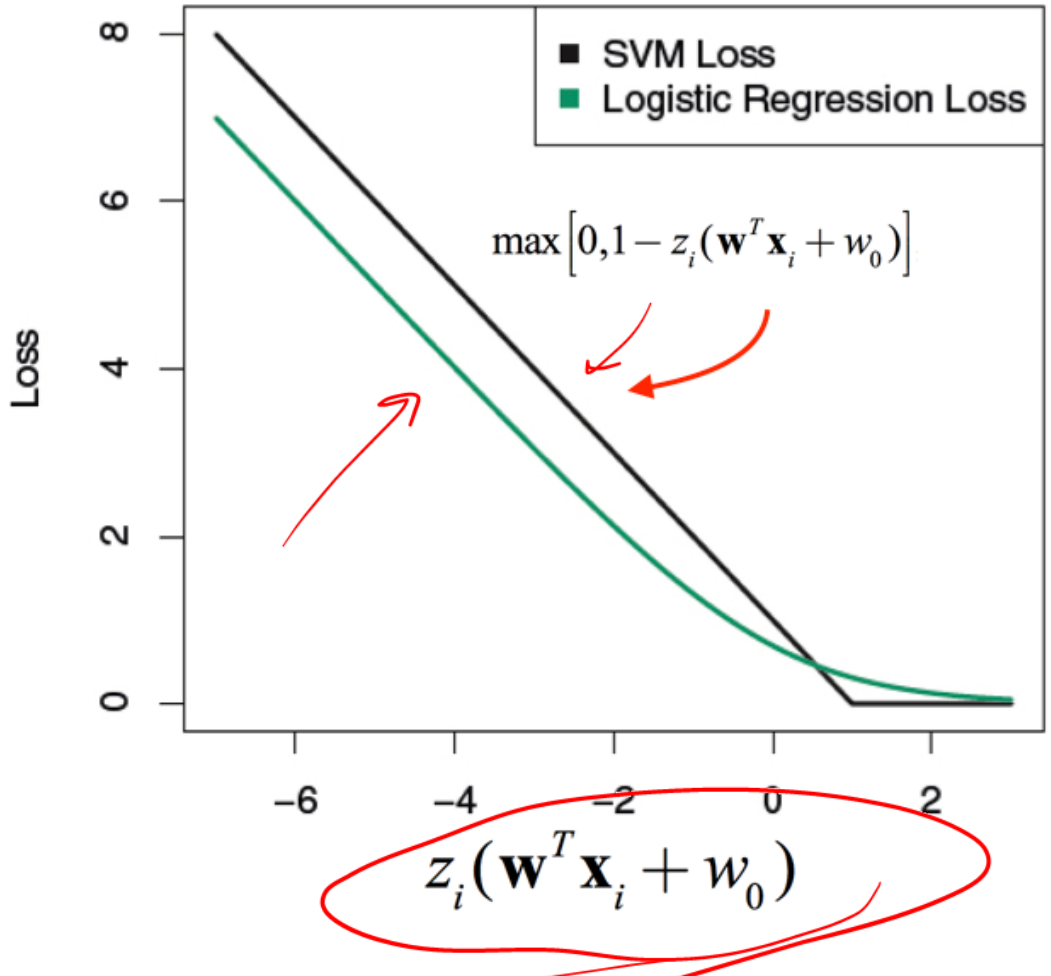
cross validation

elastic-net penalty

$\alpha \|w\|_1 + (1-\alpha)\|w\|_2$

Hinge Loss vs. Negative Log Likelihood

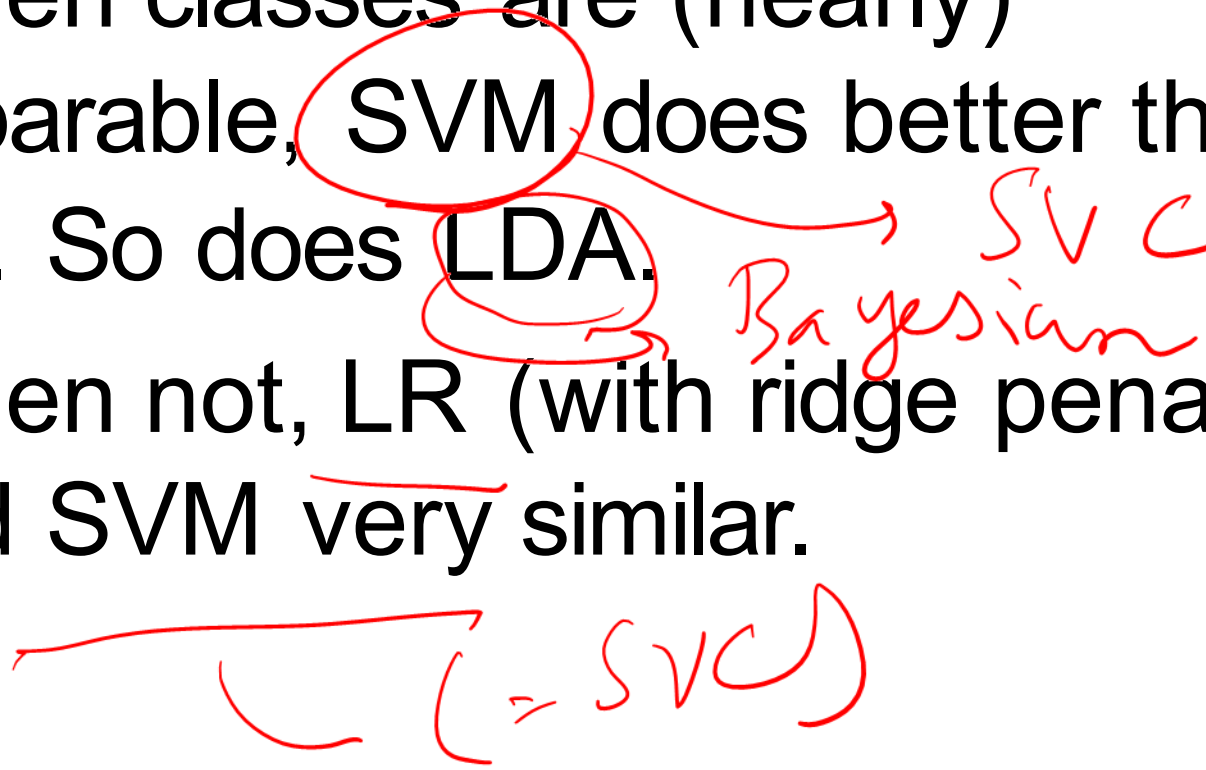
They look very similar, that is why LR and SVC usually yield similar results



James, (ISLR, Hastie - Tibshirani)

Which to use: SVM or Logistic Regression

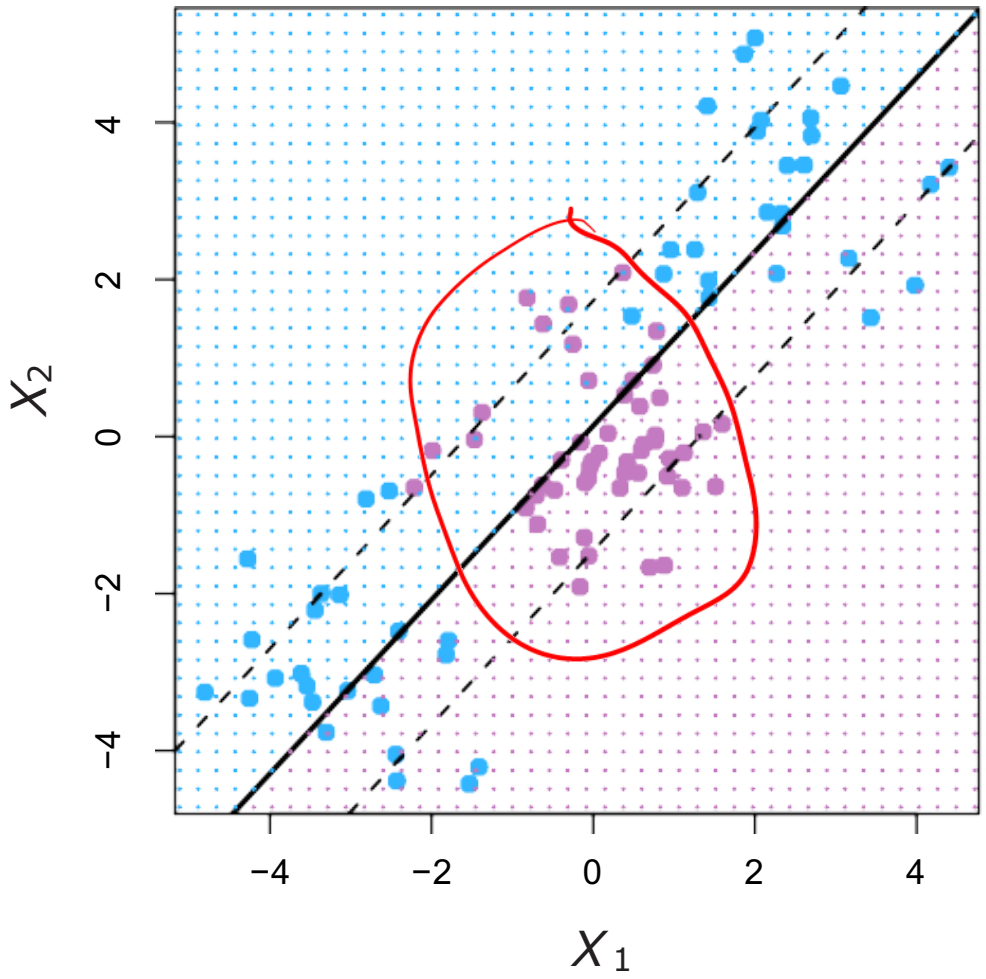
- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not, LR (with ridge penalty) and SVM very similar.



Which to use: SVM or Logistic Regression

- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and Bayesian LDA as well, but computations are more expensive.

Linear boundary can fail



Sometime a linear boundary simply won't work, no matter what value of C.

The example on the left is such a case.

What to do?

Nonlinear Features: Support Vector Machines (SVMs)

- One can use a new set of features

$$\mathbf{u}_i = \varphi(\mathbf{x}_i)$$

- $\mathbf{u}_i$ can have higher dimensions than $\mathbf{x}_i$
- The issue is that we do not really know how to choose φ
- When the dimensions are very large, computational burdens become very high

The Kernel Trick: Support Vector Machines (SVMs)

- There is a more principled way of introducing nonlinearities to the model
- Let us get back to the dual formulation of the SVC classifier, but using the extended feature space:

$$L_D(\underline{\lambda}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \mathbf{u}_i^T\mathbf{u}_j + \sum_{i=1}^{N}\lambda_i$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \left[\varphi(\mathbf{x}_i)\right]^T\left[\varphi(\mathbf{x}_j)\right] + \sum_{i=1}^{N}\lambda_i$$

expanded

Positive Definite

a Cost

$K(x_i, x_j)$

The Kernel Trick: Support Vector Machines (SVMs)

$$L_D(\underline{\lambda}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \mathbf{u}_i^T\mathbf{u}_j + \sum_{i=1}^{N}\lambda_i$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \left[\varphi(\mathbf{x}_i)\right]^T\left[\varphi(\mathbf{x}_j)\right] + \sum_{i=1}^{N}\lambda_i$$

- Roughly speaking, any function that can be written in the following form is called a Kernel:

$$\kappa(\mathbf{x}_i,\mathbf{x}_j) = \left[\varphi(\mathbf{x}_i)\right]^T\left[\varphi(\mathbf{x}_j)\right]$$

  - It is an extension of inner product, and measures the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$

The Kernel Trick: Support Vector Machines (SVMs)

- Rewriting the dual problem using the Kernel function:

$$L_D(\underline{\lambda}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z_i z_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{N}\lambda_i$$

- It turns out that one does not even need to know φ explicitly to use the Kernel
- In fact, for some specific Kernels φ is infinite dimensional!

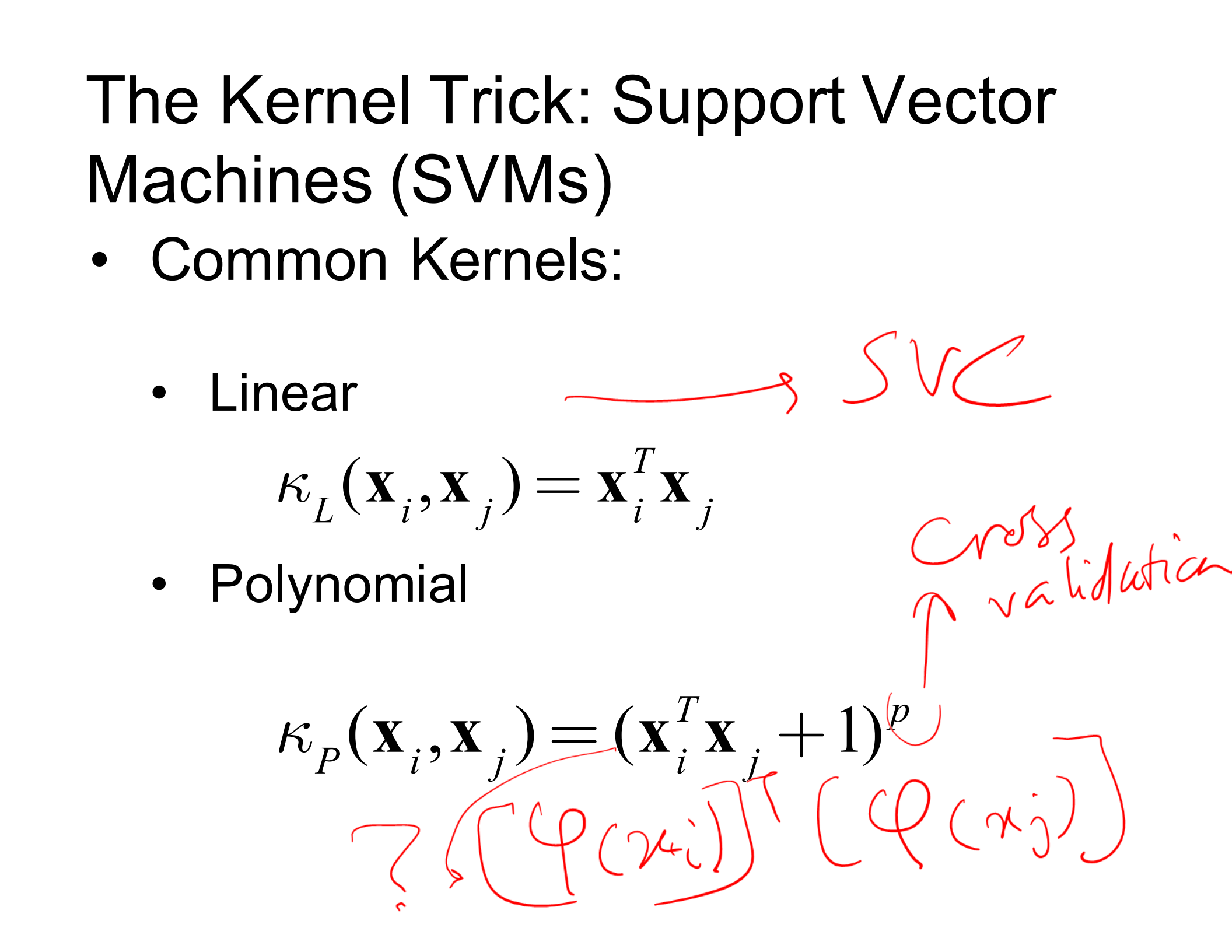
The Kernel Trick: Support Vector Machines (SVMs)

- Common Kernels:
  - Linear

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

  - Polynomial

$$\kappa_P(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p$$



The Kernel Trick: Support Vector Machines (SVMs)

- Common Kernels:
  - Radial Basis Function (RBF) or Gaussian

$$\kappa_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$$

  - or

$$\kappa_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

The Kernel Trick: Support Vector Machines (SVMs)

- Common Kernels:
  - Laplace Kernel

$$\kappa_{Lap}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\| \mathbf{x}_i - \mathbf{x}_j \right\|\right)$$

  - Sigmoid Kernel

$$\kappa_{Sig}(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i^T \mathbf{x}_j + c)$$

The Kernel Trick: Support Vector Machines (SVMs)

- All Common Kernels have hyperparameters that have to be determined using cross-validation.
- According to Mercer's theorem, there is little justification for preference of one type of Kernel over others