Chapter 5

Linear Discriminant Functions

Discriminant Function

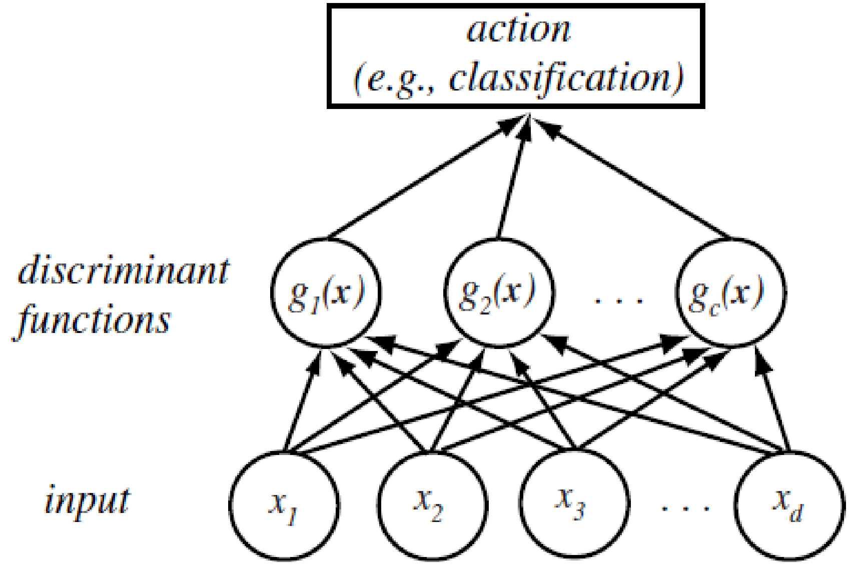
Discriminant functions

$g_i : \mathbf{R}^d \to \mathbf{R} \qquad (1 \le i \le c)$

- *Useful way to represent classifiers*
- *One function per category*

Decide ω_i

if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$



Minimum risk: $g_i(\mathbf{x}) = -R(\alpha_i \mid \mathbf{x}) \quad (1 \le i \le c)$

Minimum-error-rate: $g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) \quad (1 \le i \le c)$

Discriminant Function (Cont.)

Decision region

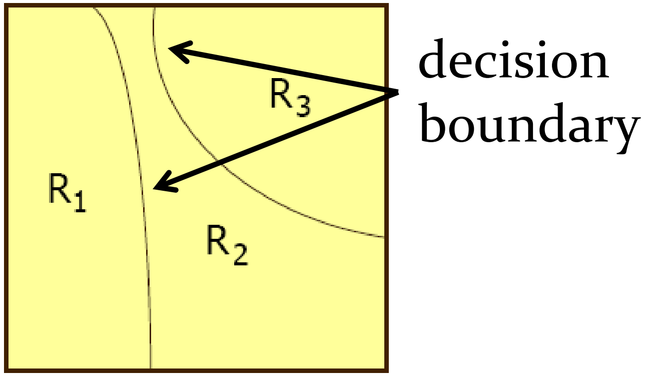
c discriminant functions $g_i(\cdot)\ (1 \leq i \leq c)$ → c decision regions $\mathcal{R}_i \subset \mathbf{R}^d\ (1 \leq i \leq c)$

$$\mathcal{R}_i = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{R}^d : g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq i\}$$

where $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset\ (i \neq j)$ and $\bigcup_{i=1}^{c} \mathcal{R}_i = \mathbf{R}^d$

Decision boundary

surface in feature space where ties occur among several largest discriminant functions



Linear Discriminant Functions

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0} \qquad (i = 1, 2, \ldots, c)$$

$\mathbf{w}_i$: weight vector (权值向量, *d*-dimensional)

w_{i0} : bias/threshold (偏置/阈值, scalar)

$\mathbf{x} = (x_1, x_2, x_3)^t$ $\qquad$ $d = 3,\ c = 3$

$g_1(\mathbf{x}) = x_1 - 2x_2 + 4x_3$ $\qquad$ $\mathbf{w}_1 = (1, -2, 4)^t,\ w_{10} = 0$

$g_2(\mathbf{x}) = x_1 + 3x_3 + 4$ $\qquad$ $\mathbf{w}_2 = (1, 0, 3)^t,\ w_{20} = 4$

$g_3(\mathbf{x}) = -2$ $\qquad$ $\mathbf{w}_3 = (0, 0, 0)^t,\ w_{30} = -2$

Generalized Linear Discriminant Functions(广义线性判别方程)

Quadratic discriminant function (二次判别函数)

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

Polynomial discriminant function (多项式判别函数)

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

(e.g. *3rd-order polynomial*)

……

Generalized Linear Discriminant Functions (Cont.)

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x}) \quad \Longleftrightarrow \quad g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

$\mathbf{a}$: *the* $\hat{d}$*-dimensional weight vector* $(a_1, a_2, \ldots, a_{\hat{d}})^t$

$\mathbf{y}$: *the* $\hat{d}$*-dimensional transformed feature vector*

$(y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{\hat{d}}(\mathbf{x}))^t$ *, where* $y_i(\mathbf{x})$ *can be viewed as a feature detecting subsystem*

The resulting discriminant function *g*(**x**) may not be linear in **x**, but it is linear in **y**.

Generalized Linear Discriminant Functions (Cont.)

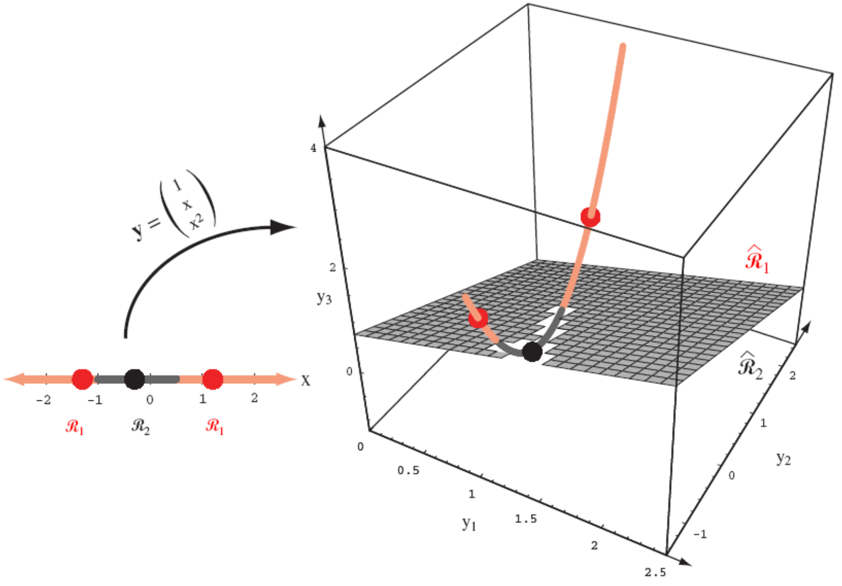
An example (quadratic in $\mathbf{x}$ ➔ linear in $\mathbf{y}$)

$$g(\mathbf{x}) = a_1 + a_2 x + a_3 x^2$$

$$\mathbf{a} = (a_1, a_2, a_3)^t$$

$$\mathbf{y} = \begin{pmatrix} y_1(\mathbf{x}) \\ y_2(\mathbf{x}) \\ y_3(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

$$g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$



$$\mathbf{a} = (-1, 1, 2)^t$$

Augmentation Representation

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = w_0 + \mathbf{w}^t \mathbf{x}$$

$$\underbrace{\mathbf{a} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}}_{\textbf{augmented weight vector}} \qquad \underbrace{\mathbf{y} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}}_{\textbf{augmented feature vector}}$$

$$g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

Transform the task of finding weight vector **w** and bias w_0 into the task of finding **a** (d+1 parameters)

The Two-Category Case

Training set

$$\mathcal{D} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \quad \left(\mathbf{y}_i \in \mathbf{R}^{d+1},\ \Omega = \{\omega_1, \omega_2\}\right)$$

The task

Determine the (augmented) weight vector $\mathbf{a} \in \mathbf{R}^{d+1}$ where $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$ can classify all training samples in $\mathcal{D}$ correctly

$\mathbf{a}^t\mathbf{y}_i > 0$ if $\mathbf{y}_i$ is labeled ω_1

$\mathbf{a}^t\mathbf{y}_i < 0$ if $\mathbf{y}_i$ is labeled ω_2

*Simplified treatment: "**normalization**", i.e. replace* $\mathbf{y}_i$ *with* $-\mathbf{y}_i$ *if it is labeled* ω_2

$\mathbf{a}^t\mathbf{y}_i > 0$ for all (normalized) $\mathbf{y}_i$

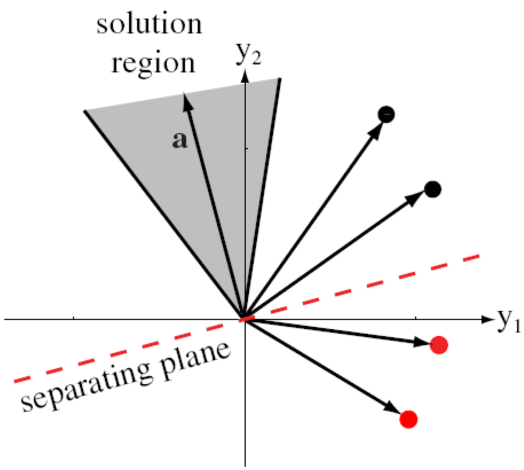
The Two-Category Case (Cont.)

The separable case

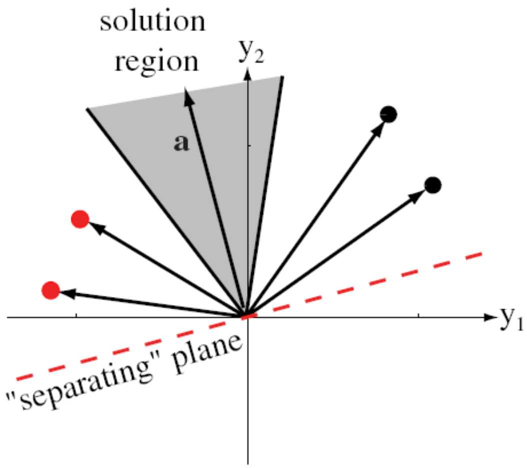
$\mathbf{a}^t\mathbf{y}_i > 0$ ⇒ **a** should be on the *positive* side of the hyperplane defined by $\mathbf{a}^t\mathbf{y}_i=0$ with $\mathbf{y}_i$ being the norm vector

Solution region

the **intersection of *n* half-spaces** yielded by the *n* training samples



original training samples



normalized training samples

The Two-Category Case (Cont.)

Solution to a, *i.e.* (augmented) weight vector $(g(\mathbf{x}) = \mathbf{a}^t\mathbf{y})$

Minimize a **criterion/objective function (准则函数) *J*(a)**

based on the normalized training samples $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$

$$J(\mathbf{a}) = -\sum_{i=1}^{n} \texttt{sign}\big[\,\mathbf{a}^t\mathbf{y}_i > 0\,\big]$$

$$J(\mathbf{a}) = -\sum_{i=1}^{n} \mathbf{a}^t\mathbf{y}_i$$

$$J(\mathbf{a}) = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b_i)^2$$

……

How to minimize the criterion function *J*(a)?

Gradient Descent (梯度下降)

Gradient Descent

Taylor Expansion (泰勒展式)

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^t \cdot \Delta\mathbf{x} + O(\Delta\mathbf{x}^t \cdot \Delta\mathbf{x})$$

$f : \mathbf{R}^d \to \mathbf{R}$: a real-valued *d*-variate **function**

$\mathbf{x} \in \mathbf{R}^d$: **a point** in the *d*-dimensional Euclidean space

$\Delta\mathbf{x} \in \mathbf{R}^d$: a **small shift** in the *d*-dimensional Euclidean space

$\nabla f(\mathbf{x})$: **gradient** of $f(\cdot)$ at $\mathbf{x}$

$O(\Delta\mathbf{x}^t \cdot \Delta\mathbf{x})$: the **big oh order** of $\Delta\mathbf{x}^t \cdot \Delta\mathbf{x}$ **[appendix A.8]**

Gradient Descent (Cont.)

Taylor Expansion (泰勒展式)

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^t \cdot \Delta\mathbf{x} + O(\Delta\mathbf{x}^t \cdot \Delta\mathbf{x})$$

What happens if we set Δx to be *negatively proportional* to the gradient at x, i.e.:

$$\Delta\mathbf{x} = -\eta \cdot \nabla f(\mathbf{x})$$ (η being a *small* positive scalar)

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) - \eta \cdot \nabla f(\mathbf{x})^t \cdot \nabla f(\mathbf{x}) + O(\Delta\mathbf{x}^t \cdot \Delta\mathbf{x})$$

being *non-negative*

ignored when $O(\Delta\mathrm{x}^t \cdot \Delta\mathrm{x})$ is small

Therefore, we have $f(\mathbf{x}+\Delta\mathbf{x}) \leq f(\mathbf{x})$!

Gradient Descent (Cont.)

Basic strategy

Minimize $J(\mathbf{a})$ by iteratively updating **a** with gradient descent:

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k))$$

1. **begin** **initialize** **a**, threshold $\theta,\ \eta(\cdot),\ k \leftarrow 0$
2. **do** $k \leftarrow k+1$
3. $\mathbf{a} \leftarrow \mathbf{a} - \eta(k)\nabla J(\mathbf{a})$
4. **until** $\|\eta(k)\nabla J(\mathbf{a})\| < \theta$
5. **return** **a**
6. **end**

Basic Gradient Descent

Gradient Descent (Cont.)

Choice of learning rate

$\eta(k)$ is too small → convergence is slow

$\eta(k)$ is too large → overshoot (过冲) or even diverge

second-order Taylor expansion

$$J(\mathbf{a}) \simeq J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a} - \mathbf{a}(k)) + \frac{1}{2}(\mathbf{a} - \mathbf{a}(k))^t\,\mathbf{H}\,(\mathbf{a} - \mathbf{a}(k))$$

$\mathbf{H}$: *Hessian matrix* with elements $\partial^2 J/\partial a_i \partial a_j$ evaluated at **a**(k)

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k))$$

$$J(\mathbf{a}(k+1)) \simeq J(\mathbf{a}(k)) - \eta(k)\|\nabla J\|^2 + \frac{1}{2}\eta(k)^2 \nabla J^t\,\mathbf{H}\,\nabla J$$

Optimal choice

$$\eta(k) = \frac{\|\nabla J\|^2}{\nabla J^t\,\mathbf{H}\,\nabla J}$$

Newton's Algorithm

$$J(\mathbf{a}) \simeq J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a} - \mathbf{a}(k)) + \frac{1}{2}(\mathbf{a} - \mathbf{a}(k))^t \mathbf{H}\,(\mathbf{a} - \mathbf{a}(k))$$

set $\partial J / \partial \mathbf{a} = \mathbf{0}$

$$\nabla J + \mathbf{H}\,\mathbf{a} - \mathbf{H}\,\mathbf{a}(k) = \mathbf{0} \Rightarrow \mathbf{a}(k+1) = \mathbf{a}(k) - \mathbf{H}^{-1}\nabla J$$

1. **begin initialize a**, threshold θ
2. **do**
3. $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{H}^{-1}\nabla J(\mathbf{a})$
4. **until** $\|\mathbf{H}^{-1}\nabla J(\mathbf{a})\| < \theta$
5. **return a**
6. **end**

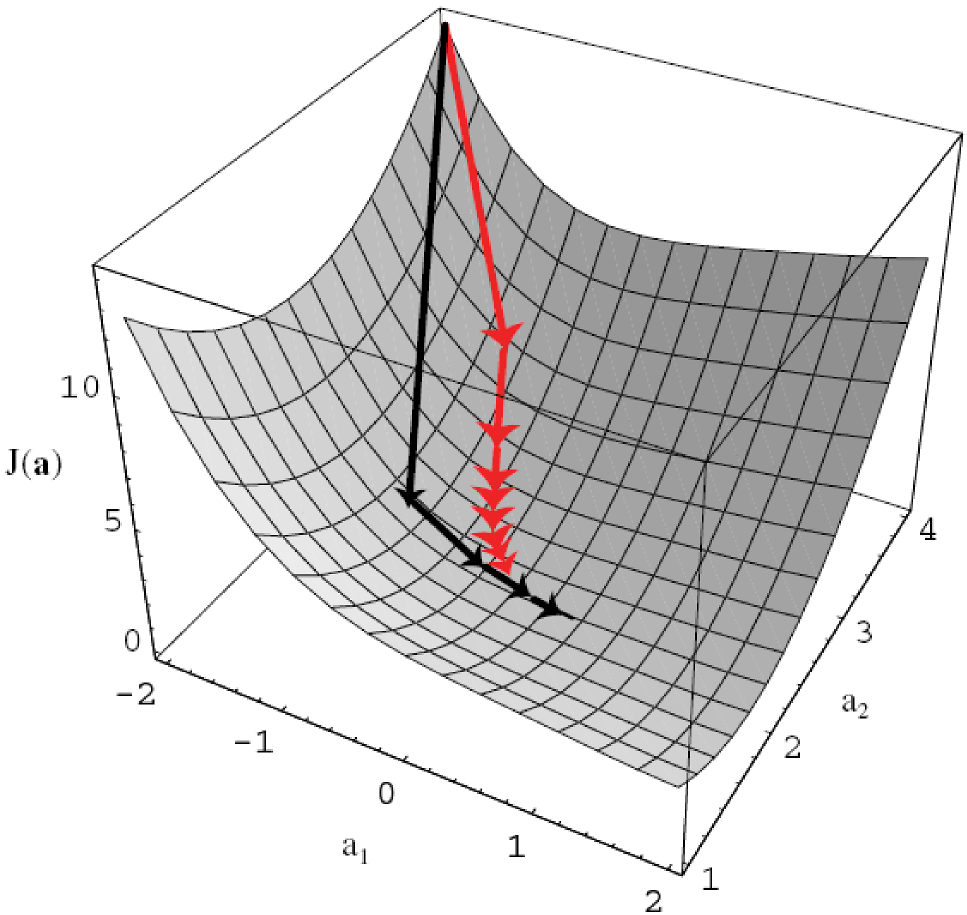
Newton Descent

Advantage: *better step size than simple gradient descent*

Disadvantage: $O(d^3)$ *complexity for matrix inversion; even not applicable if* $\boldsymbol{H}$ *is singular*

In practice, fix $\eta(k)$ to constant η

Gradient Descent vs. Newton's Algorithm



Red sequence
steps of gradient descent

Black sequence
steps of Newton's algorithm

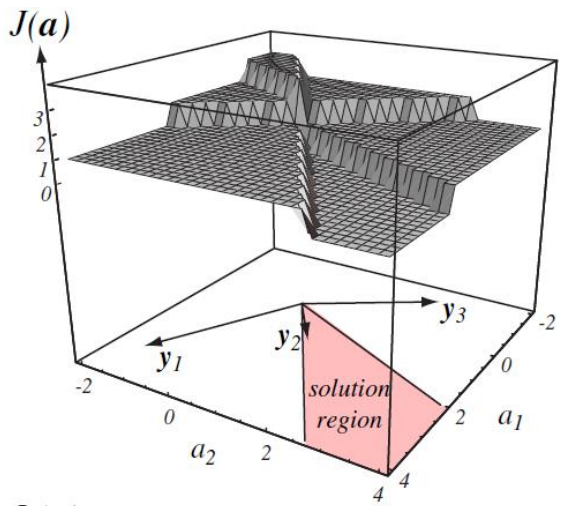
Newton's algorithm usually leads to greater improvement than gradient descent per step, but with added computational burden of inverting the Hessian matrix

Perceptron Criterion Function

Given the **normalized** training samples $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$, set the criterion function $J(\mathbf{a})$ as the **number of examples misclassified by a**, i.e.:

$$J(\mathbf{a}) = \sum_{i=1}^{n} 1_{\mathbf{a}^t\mathbf{y}_i \leq 0}$$

Piecewise constant function
(分段常数函数)



Not compatible with the gradient descent procedure for function minimization

The gradient is almost all 0
(except on the boundary of piecewise region)

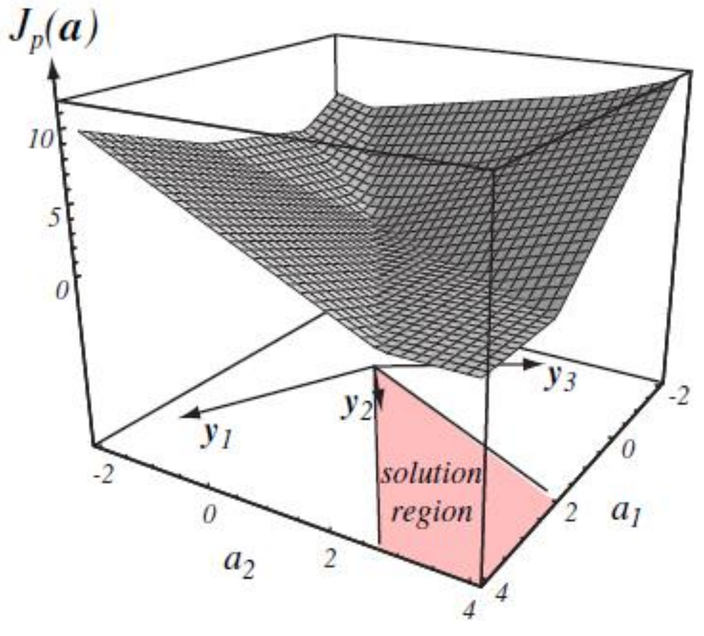
Perceptron Criterion Function (Cont.)

A better choice

$$J_p(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{a}^t\mathbf{y})$$

$$\mathcal{Y} = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \leq 0, 1 \leq i \leq n\}$$

the set of samples misclassified by **a**



The gradient: $\nabla J_p = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{y})$

The iterative update rule:

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}$$

$\mathcal{Y}_k$: *the set of samples misclassified by* $\mathbf{a}(k)$

Batch Perceptron Algorithm

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}$$

1. **begin** **initialize** **a**, threshold $\theta,\ \eta(\cdot),\ k \leftarrow 0$
2. **do** $k \leftarrow k+1$
3. $\mathcal{Y}_k = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \le 0, 1 \le i \le n\}$
4. $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}$
5. **until** $\|\eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}\| < \theta$
6. **return a**
7. **end**

Batch Perceptron

Batch mode

The next weight vector is obtained by adding some multiple of the sum of the misclassified samples to the present weight vector

Single-Sample Correction

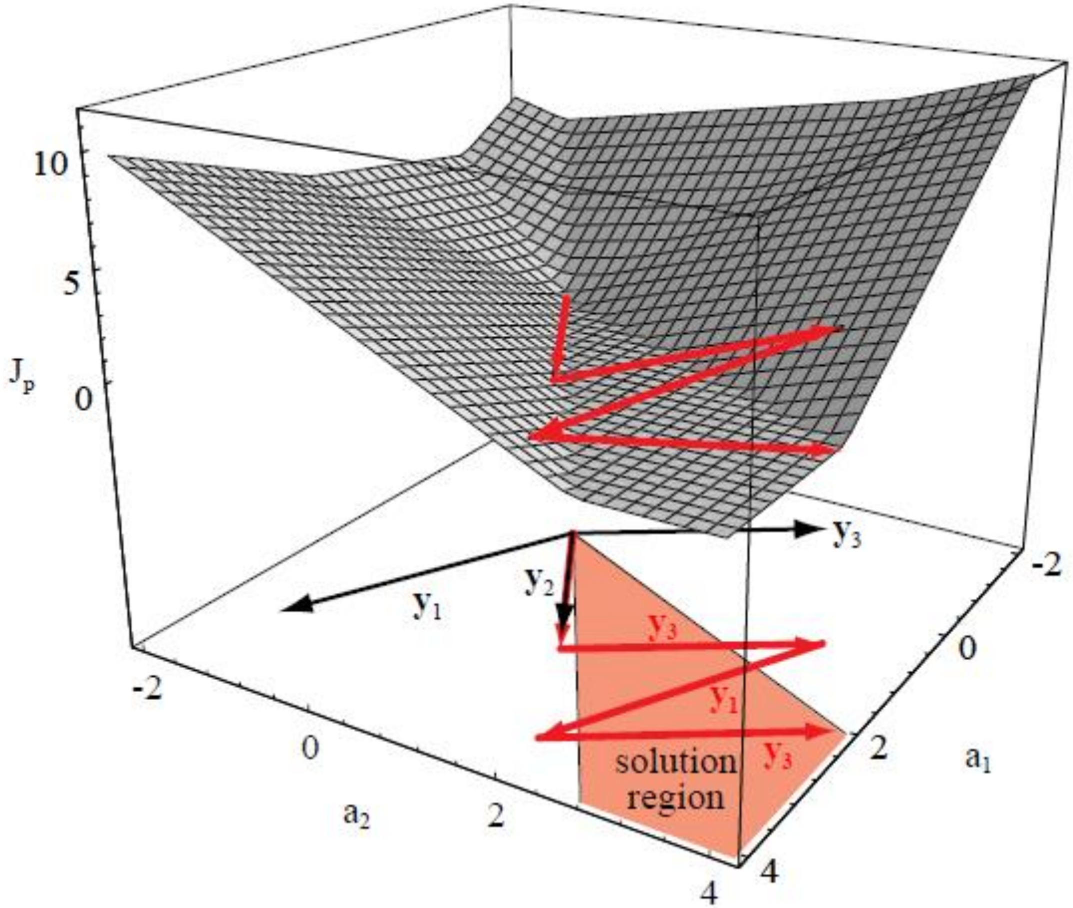
Batch m

Update (c
on a set o
samples

Notatio

$\mathbf{y}_1, \mathbf{y}_2,$ y

$\mathbf{y}^1$ y



de

uentially
es a

ing
ly
sified
nce

Fixed-increment: $\eta(k) = 1$

Single-Sample Correction (Cont.)

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k$$

1. **begin initialize a**, $j \leftarrow 0, \; k \leftarrow 0$
2. **do** $j \leftarrow j+1$
3. $i = ((j-1) \bmod n) + 1$
4. **if** $\mathbf{y}_i$ is misclassified by **a**
5. **then** $k \leftarrow k+1; \; \mathbf{y}^k = \mathbf{y}_i; \; \mathbf{a} \leftarrow \mathbf{a} + \mathbf{y}^k$
6. **until** k is kept unchanged for n consecutive rounds
7. **return a**
8. **end**

Fixed-Increment Single-Sample Perceptron

If all training samples are ***linearly separable***

Theorem 5.1

[pp.230]

The single-sample perceptron ***converges to a solution vector***

Single-Sample Correction (Cont.)

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k$$

1. **begin initialize a**, margin *b*, $\eta(\cdot)$, $j \leftarrow 0$, $k \leftarrow 0$
2. **do** $j \leftarrow j + 1$
3. $i = ((j-1) \bmod n) + 1$
4. **if** $\mathbf{a}^t\mathbf{y}_i \leq b$
5. **then** $k \leftarrow k+1$; $\mathbf{y}^k = \mathbf{y}_i$; $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\mathbf{y}^k$
6. **until** *k* is kept unchanged for *n* consecutive rounds
7. **return a**
8. **end**

Variable-Increment Perceptron with Margin
（带裕量的变增量感知器）

Convergence conditions for ***linearly separable*** *training samples*

$$\eta(k) \geq 0$$

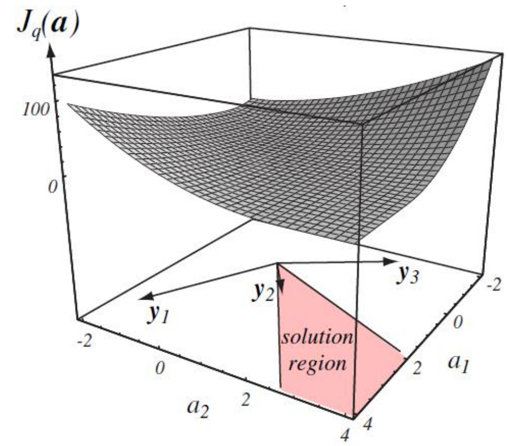
$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty \qquad \lim_{m\to\infty} \frac{\sum_{k=1}^{m} \eta^2(k)}{\left(\sum_{k=1}^{m} \eta(k)\right)^2} = 0$$

Other Criterion Functions

$$J_q(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (\mathbf{a}^t\mathbf{y})^2$$

$$\mathcal{Y} = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \le 0, 1 \le i \le n\}$$

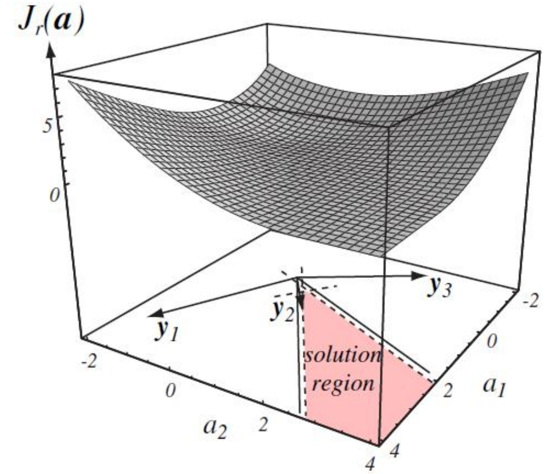
$$\nabla J_q = 2\sum_{y\in\mathcal{Y}} (\mathbf{a}^t\mathbf{y})\mathbf{y}$$



$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{\mathbf{y}\in\mathcal{Y}} \frac{(\mathbf{a}^t\mathbf{y} - b)^2}{\|\mathbf{y}\|^2}$$

$$\mathcal{Y} = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \le 0, 1 \le i \le n\}$$

$$\nabla J_r = \sum_{\mathbf{y}\in\mathcal{Y}} \frac{\mathbf{a}^t\mathbf{y} - b}{\|\mathbf{y}\|^2}\mathbf{y}$$



Minimum Squared Error (MSE; 最小平方误差)

inequalities → equalities

$$\mathbf{a}^t\mathbf{y}_i > 0 \quad\Rightarrow\quad \mathbf{a}^t\mathbf{y}_i = b_i$$

- b_i: *some arbitrarily chosen positive constant*
- ***update**: misclassified samples ➔ all samples*

$$\begin{pmatrix} y_{10} & y_{11} & \cdots & y_{1d} \\ y_{20} & y_{21} & \cdots & y_{2d} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ y_{n0} & y_{n1} & \cdots & y_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\mathbf{Ya} = \mathbf{b}$$

Y: $n \times (d+1)$ matrix

a: $(d+1)$-dimensional weight vector

b: n-dimensional column vector

Minimum Squared Error (Cont.)

$$\mathbf{Ya} = \mathbf{b}$$

$\mathbf{Y}$: $n \times (d+1)$ matrix

Usually, $n \geq d+1$ ➔ **Y is overdetermined** (超定) ➔ **no exact solution for a**

Sum-of-squared-error criterion function

$$J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (\mathbf{a}^t \mathbf{y}_i - b_i)^2$$

$$\nabla J_s = \sum_{i=1}^{n} 2(\mathbf{a}^t \mathbf{y}_i - b_i)\mathbf{y}_i = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$$

$$\nabla J_s = \mathbf{0} \Rightarrow \mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b} \Rightarrow \mathbf{a} = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b} \Rightarrow \mathbf{a} = \mathbf{Y}^\dagger\mathbf{b}$$

$$\mathbf{Y}^\dagger = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t$$

pseudo-inverse (伪逆) of **Y**

$\mathbf{Y}^\dagger\mathbf{Y} = \mathbf{I}$, but $\mathbf{Y}\mathbf{Y}^\dagger \neq \mathbf{I}$ in general

Minimum Squared Error (Cont.)

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\mathbf{Y}^t(\mathbf{Y}\mathbf{a}(k) - \mathbf{b})$$

Batch mode

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)(b_k - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k$$

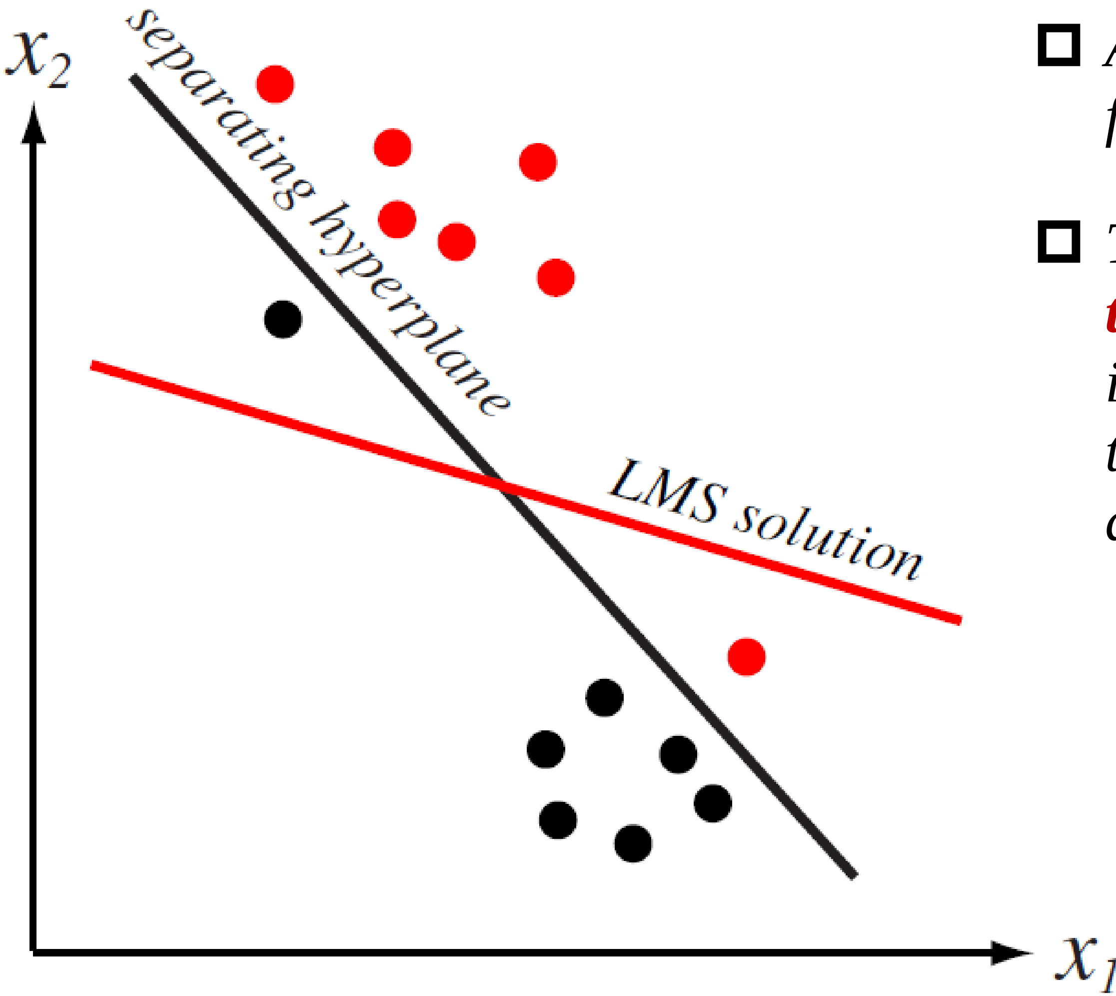
Single-sample mode

*the **Widrow-Hoff** rule or **LMS** (least-mean-squared) rule*

1. **begin initialize a, b**, threshold θ, $\eta(\cdot)$, $k \leftarrow 0$
2. **do** $k \leftarrow k+1$
3. $i = ((k-1) \bmod n) + 1$
4. $\mathbf{y}^k \leftarrow \mathbf{y}_i;\ b_k \leftarrow b_i; \mathbf{a}_* \leftarrow \mathbf{a};\ \mathbf{a} \leftarrow \mathbf{a} + \eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k$
5. **until** $\|\eta(k)(b_k - \mathbf{a}_*^t\mathbf{y}^k)\mathbf{y}^k\| < \theta$
6. **return a**
7. **end**

LMS (least-mean-squared)

Minimum Squared Error (Cont.)



- *A common choice of $\eta(k)$ for LMS: $\eta(k)=1/k$*
- *The LMS solution **minimizes the sum-of-squared errors**, i.e. the (squared) distance of the training samples to the decision hyperplane*

LMS solution may not be a separating hyperplane

Multi-Category Generalization

Training set

$$\mathcal{D} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \ \left(\mathbf{y}_k \in \mathbf{R}^{d+1}, \ \Omega = \{\omega_1, \ldots, \omega_c\}\right)$$

The task

Determine the set of (augmented) weight vectors $\{\mathbf{a}_i \in \mathbf{R}^{d+1} \mid 1 \leq i \leq c\}$ where $g_i(\mathbf{x}) = \mathbf{a}_i^t \mathbf{y} \ (1 \leq i \leq c)$ can classify all training samples in $\mathcal{D}$ correctly

if $\mathbf{y}_k$ $(1 \leq k \leq n)$ is labeled ω_i

$$\forall j \neq i : \mathbf{a}_i^t \mathbf{y}_k > \mathbf{a}_j^t \mathbf{y}_k$$

Multi-Category Generalization (Cont.)

Kesler's Construction

Transform the multi-category classification problem into the binary classification problem

$\{\mathbf{a}_i \in \mathbf{R}^{d+1} \mid 1 \leq i \leq c\}$

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_c \end{bmatrix}$$

binary classification model: **$c(d+1)$-dimensional weight vector**

if $\mathbf{y}_k$ ($1 \leq k \leq n$) is labeled ω_i

$$\boldsymbol{\eta}_{i1}^k = \begin{bmatrix} -\mathbf{y}_k \\ \vdots \\ \mathbf{y}_k \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \boldsymbol{\eta}_{ij}^k = \begin{array}{c} \begin{bmatrix} 0 \\ \vdots \\ \mathbf{y}_k \\ \vdots \\ -\mathbf{y}_k \\ \vdots \\ 0 \end{bmatrix} \end{array} \begin{array}{l} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{array} \qquad \ldots, \quad \boldsymbol{\eta}_{ic}^k = \begin{bmatrix} 0 \\ \vdots \\ \mathbf{y}_k \\ \vdots \\ 0 \\ \vdots \\ -\mathbf{y}_k \end{bmatrix}$$

binary (normalized) training set: **$n(c-1)$ training samples each being $c(d+1)$-dimensional**

Related Topic I

Principal Component Analysis (PCA)

Curse of Dimensionality (维数灾难)

*The **curse of dimensionality** refers to the phenomena that occur when classifying, organizing, and analyzing high dimensional data that does not occur in low dimensional spaces*

e.g.: *Maximum-Likelihood (ML) estimation for Gaussian pdf*

Computational Complexity

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum\nolimits_{k=1}^{n} \mathbf{x}_k \Rightarrow O(nd) \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum\nolimits_{k=1}^{n} (\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t \Rightarrow O(nd^2)$$

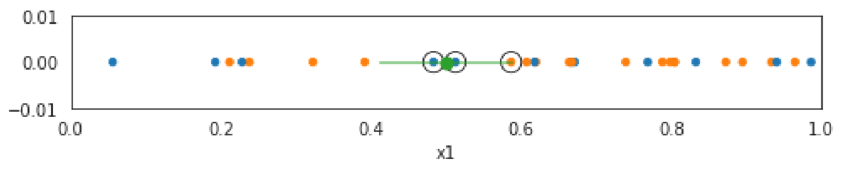
$$g(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}})^t \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}) - \frac{1}{2}\ln|\hat{\boldsymbol{\Sigma}}| + \ln P(\omega) \Rightarrow O(d^3)$$

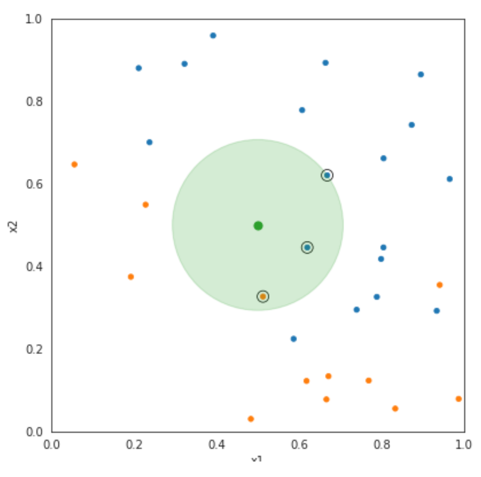
Overfitting

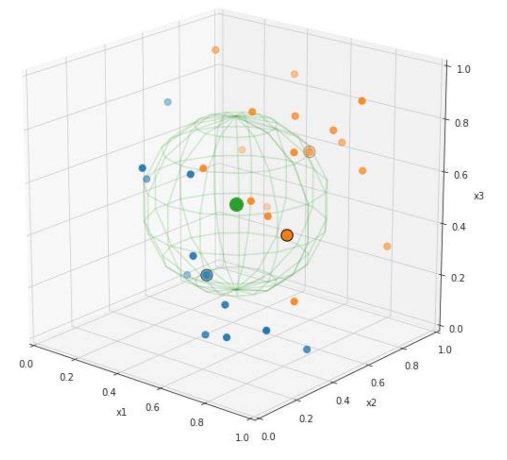
$\#$ **parameters** $\gg$ $\#$ **examples** $\Rightarrow$ unreliable parameter estimation

Curse of Dimensionality – k-NN

- In high dimensional spaces, sample that are drawn from a probability distribution are unlikely to be close.



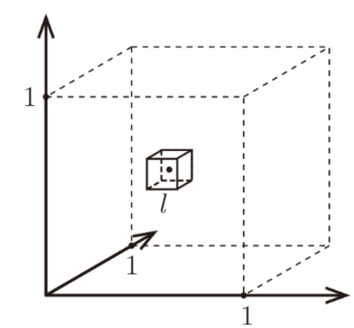




Curse of Dimensionality – k-NN

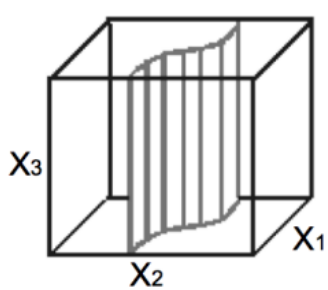
- Formally, imagine unit cube $[0,1]^d$, samples are uniformly distributed within the cube
  - Let l be the edge length of the smallest hyper-cube that contains all k-nearest neighbors of a point
  - $l^d \approx \frac{k}{n}$
  - $l \approx \left(\frac{k}{n}\right)^{\frac{1}{d}}$
  - Suppose $n = 1000, k = 10$

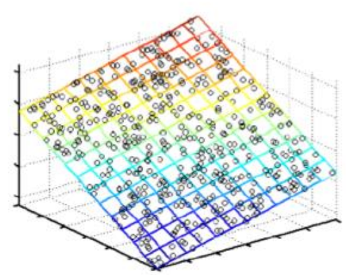
d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

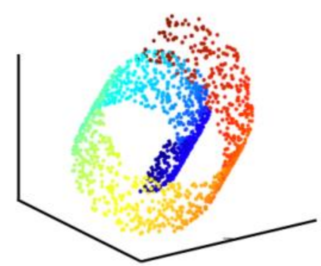


Two types of dimensionality reductions

- Feature selection
  - only a few features are relevant to the task
- Latent features
  - a (linear) combination of features provides a more efficient representation than the observed features (e.g. PCA-Principle Component Analysis)





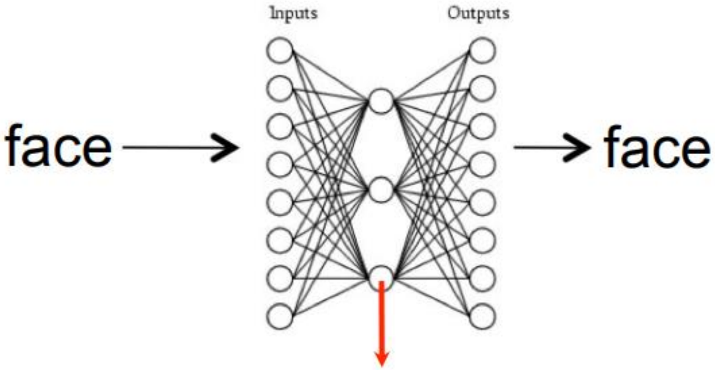


A Facial Recognition Example

- Option 1:
  - enumerate all 7 billion faces, update as necessary
- Option 2:
  - learn a low dimensional representation that can be used to describe any face
  - Principle Component Analysis (PCA) is a solution
  - Neural Networks also offer (lots of) solutions

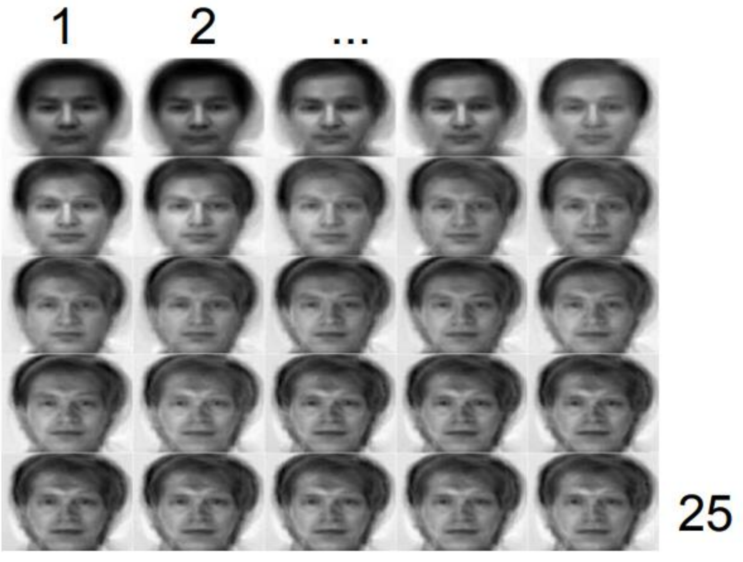
Principal Components Analysis (PCA)

$$face_i = \sum_k c_{ik} eigenface_k$$

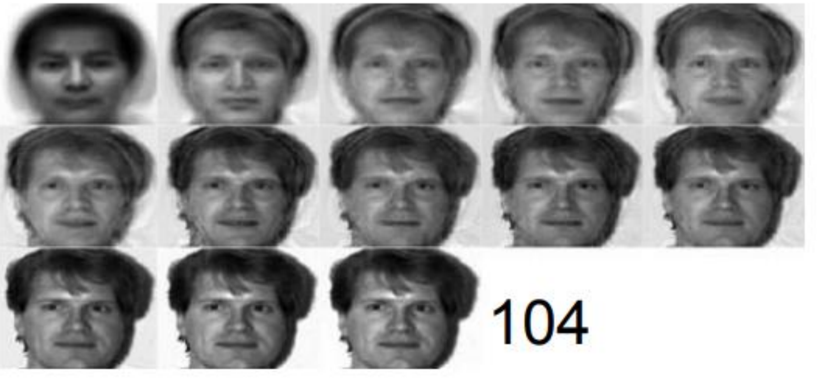


Face reconstruction using PCA

Reconstruction using the first 25 components (eigenfaces), one at a time



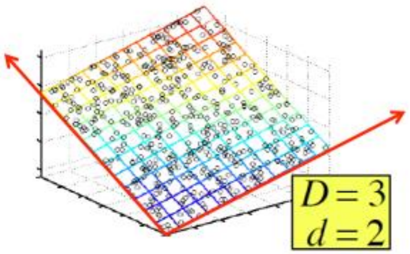
Same, but adding 8 PCA components at each step

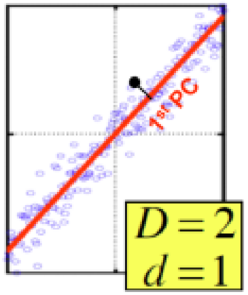


In general: top k dimensions are the k-dimensional representation that minimizes reconstruction (sum of squared) error.

Principal Component Analysis (PCA)

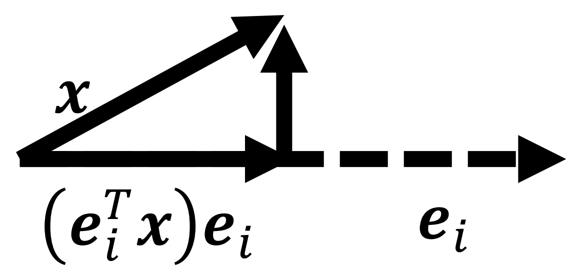
- Given data points in D-dimensional space, project them onto a lower dimensional space while preserving as much information as possible.
  - Principal components are orthogonal directions that capture variance in the data
  - 1st PC: direction of greatest variability in the data
  - 2nd PC: next orthogonal (uncorrelated) direction
    - remove variability in the first direction
    - then find the next direction of greatest variability
  - Etc.





Principal Component Analysis (PCA)

- Orthogonal projection
  - If $\boldsymbol{e}_i \in \mathbb{R}^d$, $||\boldsymbol{e}_i|| = \boldsymbol{e}_i^T \boldsymbol{e}_i = 1$
  - Then$(\boldsymbol{x}^T \boldsymbol{e}_i)\boldsymbol{e}_i$ is the orthogonal projection of $\boldsymbol{x}$ on $\boldsymbol{e}_i$
- Consider that if we want to find some number d' of $\boldsymbol{e}_1, \dots, \boldsymbol{e}_{d'}$
  - The **reconstruction error** is $(d' = 1)$
  - $\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \left|| \boldsymbol{x}_i - (\boldsymbol{e}^T \boldsymbol{x}_i)\boldsymbol{e}^T |\right|^2$
- How to find directions with the best reconstruction of the training samples?



Principal Component Analysis (Cont.)

Goal: Find linear projections with good representation ability

Input	A set of *n* *d*-dimensional samples $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ $(\mathbf{x}_k \in \mathbf{R}^d)$
Output	**Orthonormal (标准正交)** projection bases $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{d'}\}$ $(d' \leq d)$ which can best represent the (centered) samples

$$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \; J_{d'}$$

$$\mathbf{s.t.}: \mathbf{e}_i^t \mathbf{e}_i = 1 \;\; (1 \leq i \leq d')$$

$$\mathbf{e}_i^t \mathbf{e}_j = 0 \;\; (i \neq j)$$

$$\mathbf{m} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k$$

sample mean

$$a_{ki} = \mathbf{e}_i^t(\mathbf{x}_k - \mathbf{m})$$

projection of (centered) $\mathbf{x}_k$ *on* $\mathbf{e}_i$

$$J_{d'} = \sum_{k=1}^{n} \left\| \left(\sum_{i=1}^{d'} a_{ki}\mathbf{e}_i \right) - (\mathbf{x}_k - \mathbf{m}) \right\|^2$$

PCA criterion function

Principal Component Analysis (Cont.)

$$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \quad J_{d'}$$

$$\text{s.t.}: \mathbf{e}_i^t\mathbf{e}_i = 1 \ (1 \le i \le d')$$

$$\mathbf{e}_i^t\mathbf{e}_j = 0 \ (i \neq j)$$

$$\mathbf{m} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$

sample mean

$$a_{ki} = \mathbf{e}_i^t(\mathbf{x}_k - \mathbf{m})$$

projection of (centered) $\mathbf{x}_k$ *on* $\mathbf{e}_i$

$$J_{d'} = \sum_{k=1}^{n}\left\|\left(\sum_{i=1}^{d'}a_{ki}\mathbf{e}_i\right) - (\mathbf{x}_k - \mathbf{m})\right\|^2$$

$$= \sum_{k=1}^{n}\left[\left(\sum_{i=1}^{d'}a_{ki}\mathbf{e}_i\right)^t\left(\sum_{i=1}^{d'}a_{ki}\mathbf{e}_i\right) - 2\left(\sum_{i=1}^{d'}a_{ki}\mathbf{e}_i\right)^t(\mathbf{x}_k - \mathbf{m}) + \|\mathbf{x}_k - \mathbf{m}\|^2\right]$$

$$= \sum_{k=1}^{n}\left[\sum_{i=1}^{d'}a_{ki}^2\|\mathbf{e}_i\|^2 - 2\sum_{i=1}^{d'}a_{ki}\mathbf{e}_i^t(\mathbf{x}_k - \mathbf{m}) + \|\mathbf{x}_k - \mathbf{m}\|^2\right]$$

$$= \sum_{k=1}^{n}\left[-\sum_{i=1}^{d'}a_{ki}^2 + \|\mathbf{x}_k - \mathbf{m}\|^2\right]$$

(to next slide…)

Principal Component Analysis (Cont.)

$$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \quad J_{d'}$$
$$\text{s.t.}: \mathbf{e}_i^t\mathbf{e}_i = 1 \ (1 \le i \le d')$$
$$\mathbf{e}_i^t\mathbf{e}_j = 0 \ (i \ne j)$$

$$\mathbf{m} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$

sample mean

$$a_{ki} = \mathbf{e}_i^t(\mathbf{x}_k - \mathbf{m})$$

projection of (centered) $\mathbf{x}_k$ *on* $\mathbf{e}_i$

$$J_{d'} = \sum_{k=1}^{n}\left[-\sum_{i=1}^{d'} a_{ki}^2 + \|\mathbf{x}_k - \mathbf{m}\|^2\right]$$
$$= -\sum_{i=1}^{d'}\sum_{k=1}^{n}\mathbf{e}_i^t(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^t\mathbf{e}_i + \sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2$$
$$= -\sum_{i=1}^{d'}\mathbf{e}_i^t\mathbf{S}\mathbf{e}_i + \sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2$$

can be ignored

$$\left(\mathbf{S} = \sum_{k=1}^{n}(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^t\right)$$

- *symmetric*
- *positive semi-definite*

$$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \quad J_{d'}$$
$$\text{s.t.}: \mathbf{e}_i^t\mathbf{e}_i = 1 \ (1 \le i \le d')$$
$$\mathbf{e}_i^t\mathbf{e}_j = 0 \ (i \ne j)$$

relaxation

$$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \quad J_{d'}$$
$$\text{s.t.}: \mathbf{e}_i^t\mathbf{e}_i = 1 \ (1 \le i \le d')$$

Principal Component Analysis (Cont.)

$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \quad J_{d'}$

$\text{s.t.}: \mathbf{e}_i^t\mathbf{e}_i = 1 \ (1 \leq i \leq d')$

$$\mathbf{m} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$

$$J_{d'} = -\sum_{i=1}^{d'}\mathbf{e}_i^t\mathbf{S}\mathbf{e}_i \quad \left(\mathbf{S} = \sum_{k=1}^{n}(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^t\right)$$

Lagrangian function

$$L(\{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}\},\{\lambda_1,\ldots,\lambda_{d'}\}) = -\sum_{i=1}^{d'}\mathbf{e}_i^t\mathbf{S}\mathbf{e}_i + \sum_{i=1}^{d'}\lambda_i(\mathbf{e}_i^t\mathbf{e}_i - 1)$$

$$\frac{\partial L}{\partial \mathbf{e}_i} = \mathbf{0} \Rightarrow -2\mathbf{S}\mathbf{e}_i + 2\lambda_i\mathbf{e}_i = \mathbf{0} \Rightarrow \mathbf{S}\mathbf{e}_i = \lambda_i\mathbf{e}_i$$

- $\lambda_i \geq 0$: *eigenvalue of* **S**
- $\mathbf{e}_i$: *unit-norm eigenvector of* **S** *w.r.t.* λ_i

$$J_{d'} = -\sum_{i=1}^{d'}\mathbf{e}_i^t\mathbf{S}\mathbf{e}_i = -\sum_{i=1}^{d'}\lambda_i\|\mathbf{e}_i\|^2 = -\sum_{i=1}^{d'}\lambda_i \Rightarrow$$

choose top d' eigenvalues of **S** $\Rightarrow$ identify unit-norm $\mathbf{e}_i$ w.r.t. λ_i $\Rightarrow$ $\mathbf{e}_i^t\mathbf{e}_j=\mathbf{0}$ $(i\neq j)$ naturally follows

Principal Component Analysis (Cont.)

Goal: Find linear projections with good representation ability

Principal component

PCA

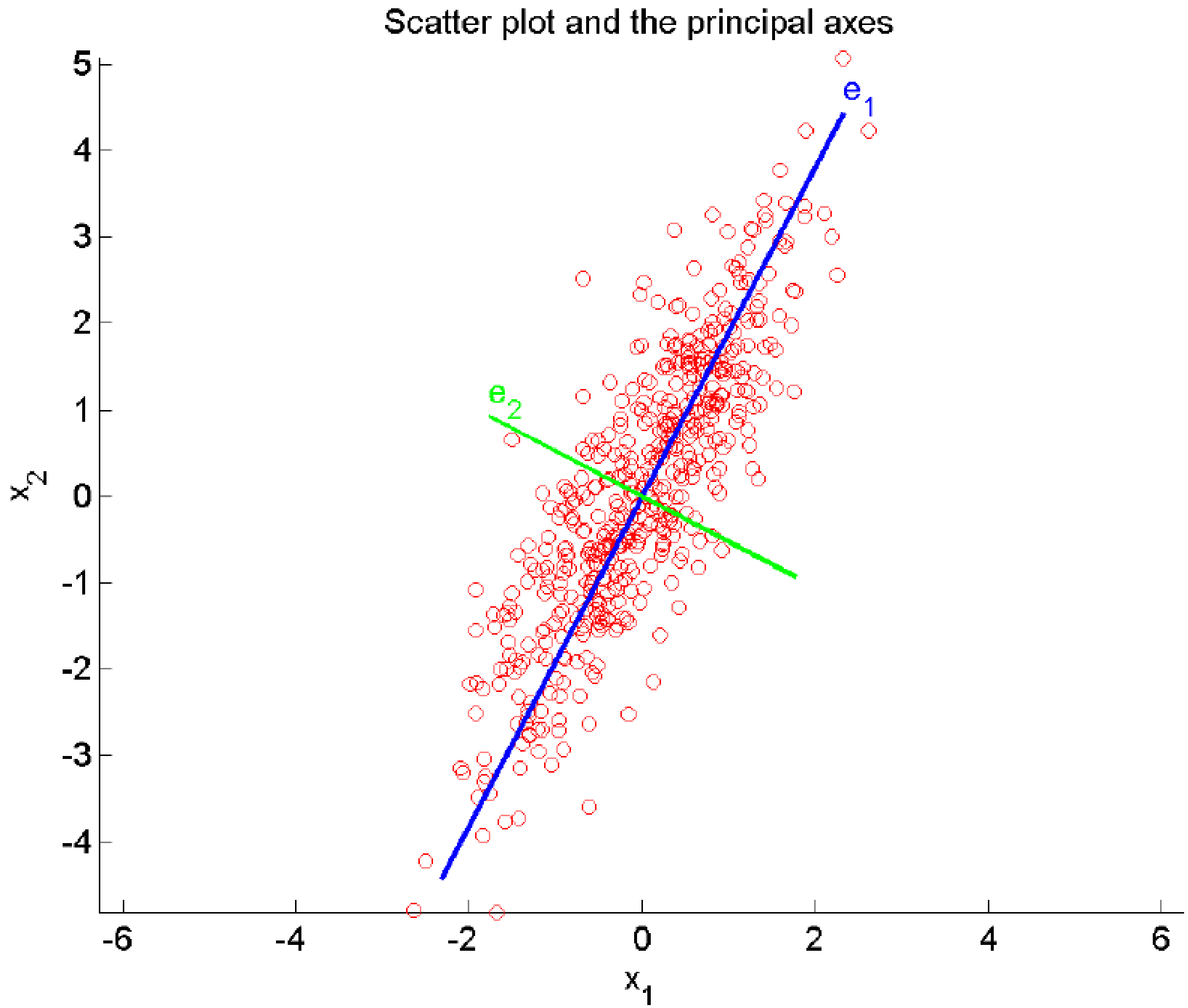
$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \ (\mathbf{x}_k \in \mathbf{R}^d)$ $\Rightarrow$ $\tilde{\mathcal{D}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_n\} \ (\boxed{\tilde{\mathbf{x}}_k} \in \mathbf{R}^{d'})$

1. Set $\mathbf{m} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k$ and $\mathbf{S} = \sum_{k=1}^{n}(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^t$
2. Identify top d' eigenvalues $\{\lambda_1, \ldots, \lambda_{d'}\}$ of **S** and their unit-norm eigenvectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_{d'}\}$
3. Form the $d \times d'$ linear projection matrix **W** by aligning the unit-norm eigenvectors in column, i.e. $\mathbf{W} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{d'}]$
4. Set $\tilde{\mathbf{x}}_k = \mathbf{W}^t(\mathbf{x}_k - \mathbf{m}) \ (1 \leq k \leq n)$

Principal Component Analysis (PCA)

Principal Component Analysis (Cont.)

An illustrative example of PCA
(in two-dimensional case)



Summary: PCA

- PCA only considers linear projections
- Covariance matrix is of size $d \times d$
  - Solution: Singular Value Decomposition (SVD)
- PCA restricts to orthogonal vectors that minimizes reconstruction error
  - Independent Component Analysis (ICA) finds statistical independent directions using information theory

PCA vs. Neural Networks

PCA	Neural Networks
Unsupervised dimensionality reduction	Supervised dimensionality reduction
Linear representation that gives best squared error fit	Non-linear representation that gives best squared error fit
No local minima (exact)	Possible local minima (gradient descent)
Non-iterative	Iterative
Orthogonal vectors (“eigenfaces”)	Auto-encoding NN with linear units may not yield orthogonal vectors

Questions

- Is this really how humans characterize and identify faces?

Related Topic II

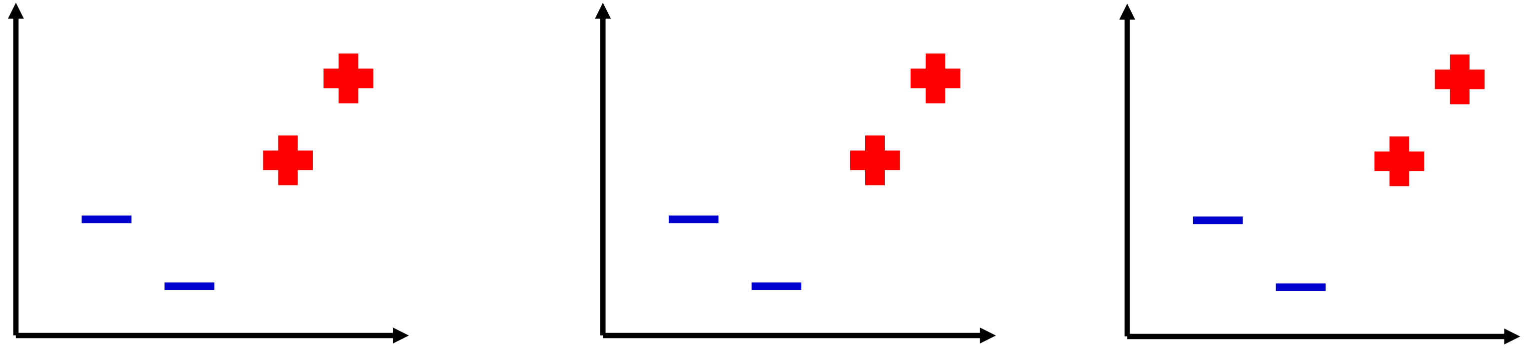
Support Vector Machine (SVM)

Support Vector Machine

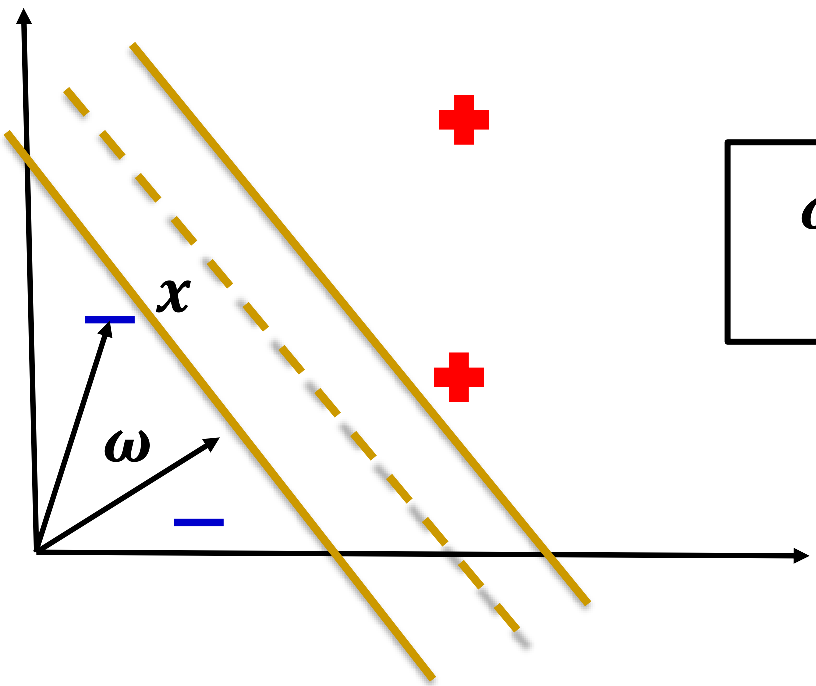
- Vladimir Vapnik
  - Born in the Soviet Union
    - PhD in statistics, 1964
    - Co-invented the VC dimension
      - Vapnik-Chervonenkis Theory, 1974
  - Moved to the U.S. in 1990
    - Jointed AT&T
    - Developed SVM algorithm in the 90's

Vladimir N. Vapnik
1936-Present

Decision Boundaries



Decision Boundary - SVM

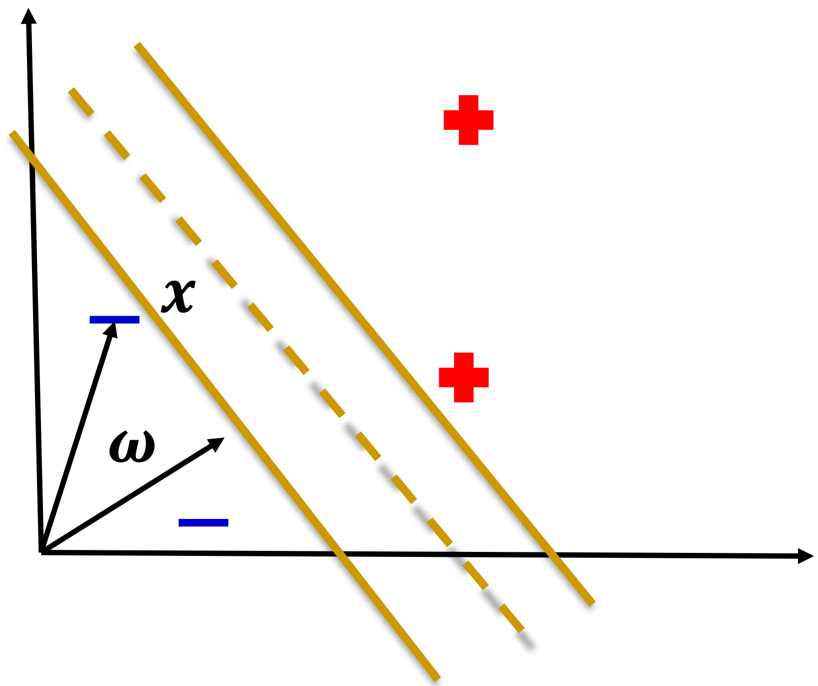


$\boldsymbol{\omega} \cdot \boldsymbol{x} \geq \boldsymbol{c} \qquad \boldsymbol{c} = -\boldsymbol{b}$

$\boldsymbol{\omega} \cdot \boldsymbol{x} + \boldsymbol{b} \geq 0,\ then\ class\ +$

Decision Rule

Decision Boundary - SVM



$$\omega \cdot x \geq c \qquad c = -b$$

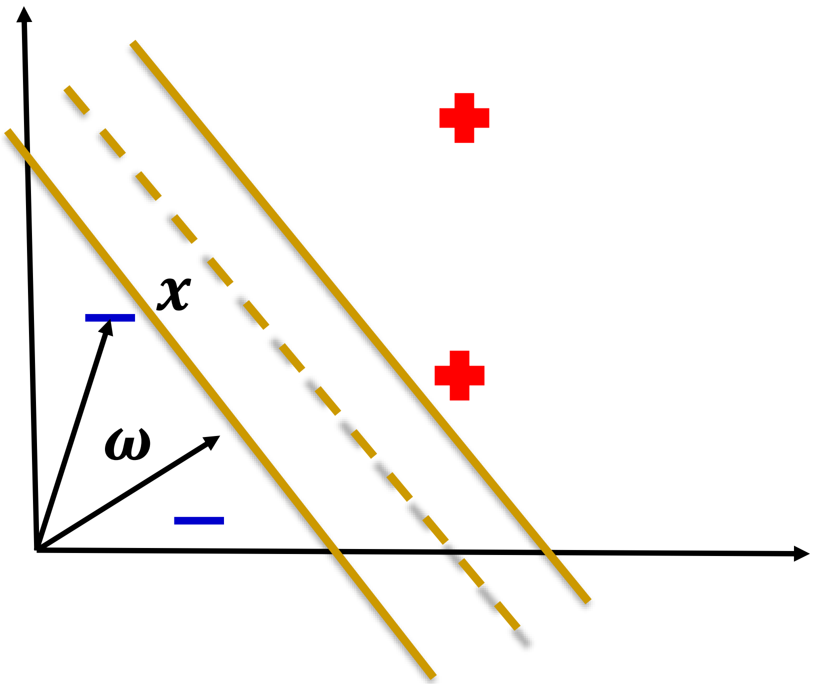
$$\omega \cdot x + b \geq 0, \textit{ then class } +$$

$$\omega \cdot x_+ + b \geq 1, \quad \textit{then class } +$$
$$\omega \cdot x_- + b \leq -1, \textit{ then class } -$$

y_i such that: $y_i = +1$ for class +

$y_i = -1$ for class -

Decision Boundary - SVM



$$\boldsymbol{\omega} \cdot \boldsymbol{x}_+ + \boldsymbol{b} \geq 1, \quad \textit{then class} +$$

$$\boldsymbol{\omega} \cdot \boldsymbol{x}_- + \boldsymbol{b} \leq -1, \textit{then class} -$$

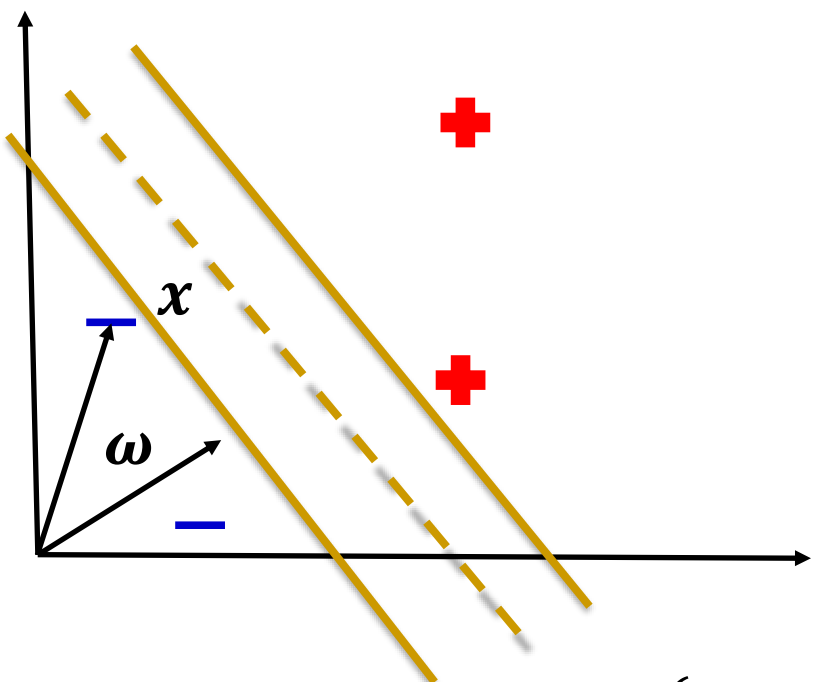
y_i such that: $y_i = +1$ for class +

$y_i = -1$ for class -

$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) \geq 1, \quad \textit{then class} +$$

$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) \geq 1, \quad \textit{then class} -$$

Decision Boundary - SVM



$$\boldsymbol{\omega} \cdot \boldsymbol{x}_+ + \boldsymbol{b} \geq 1, \quad \textit{then class} +$$
$$\boldsymbol{\omega} \cdot \boldsymbol{x}_- + \boldsymbol{b} \leq -1, \textit{then class} -$$

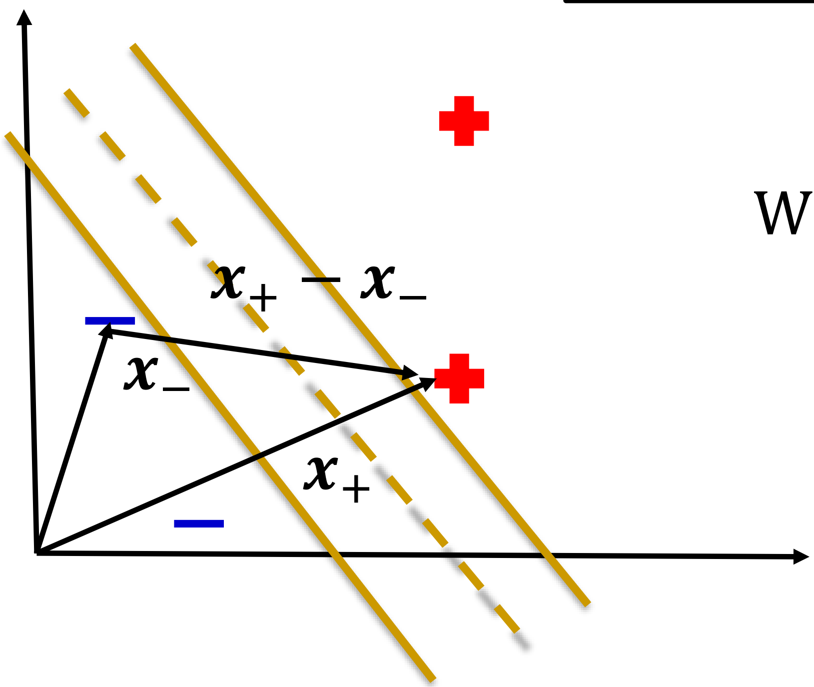
y_i such that: $y_i = +1$ for class +
$y_i = -1$ for class -

$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) \geq 1, \quad \textit{for class} +$$
$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) \geq 1, \quad \textit{for class} -$$

$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) - 1 \geq 0,$$
$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) - 1 = 0, \textit{for boundary cases}$$

Decision Boundary - SVM

$$\boxed{y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) - 1 = 0, \textit{ for support vectors}}$$



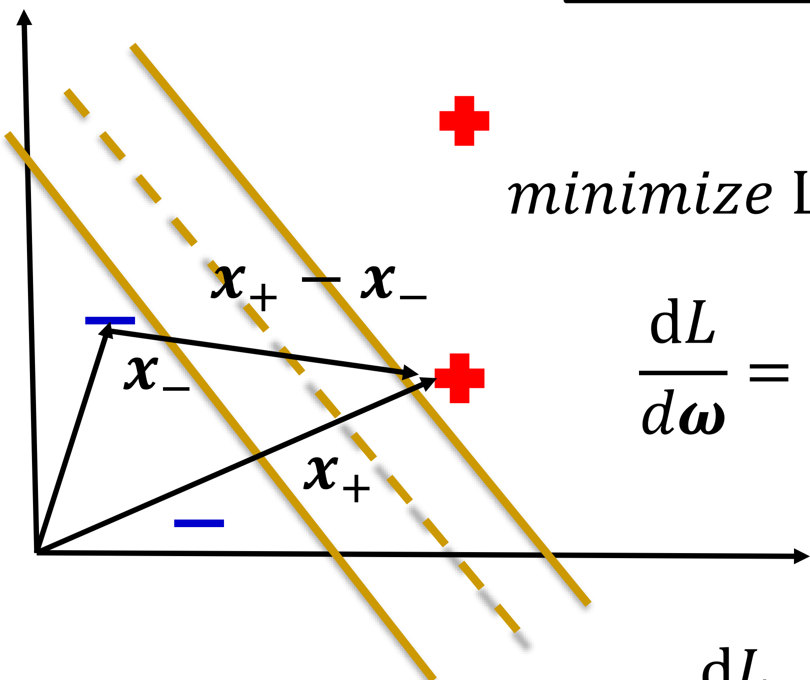
$$Width = (\boldsymbol{x}_+ - \boldsymbol{x}_-) \cdot \frac{\boldsymbol{\omega}}{||\boldsymbol{\omega}||} = \frac{2}{||\boldsymbol{\omega}||}$$

$1 - \boldsymbol{b}$ $\quad$ $1 + \boldsymbol{b}$

$$\max \frac{2}{||\boldsymbol{\omega}||} \Leftrightarrow \max \frac{1}{||\boldsymbol{\omega}||} \Leftrightarrow \min ||\boldsymbol{\omega}|| \Leftrightarrow \min \frac{1}{2} ||\boldsymbol{\omega}||^2$$

Decision Boundary - SVM

$$\boxed{y_i(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + \boldsymbol{b}) - 1 = 0, \textit{for support vectors}}$$



$$minimize\ \mathrm{L} = \frac{1}{2}||\boldsymbol{\omega}||^2 - \sum \alpha_i [y_i (\boldsymbol{\omega} \cdot \boldsymbol{x_i} + \boldsymbol{b}) - 1]$$

$$\frac{\mathrm{d}L}{d\boldsymbol{\omega}} = \boldsymbol{\omega} - \sum \alpha_i y_i \boldsymbol{x}_i = 0 \qquad \boxed{\boldsymbol{\omega} = \sum \alpha_i y_i \boldsymbol{x}_i}$$

$$\frac{\mathrm{d}L}{d\boldsymbol{b}} = -\sum \alpha_i y_i = 0 \qquad \boxed{-\sum \alpha_i y_i = 0}$$

Weight Vector

$$\boxed{y_i(\boldsymbol{\omega}\cdot\boldsymbol{x}_i+b)-1=0,\ \textit{for support vectors}}$$

$$\mathrm{L}=\frac{1}{2}||\boldsymbol{\omega}||^2-\sum\alpha_i[y_i\,(\boldsymbol{\omega}\cdot\boldsymbol{x_i}+\boldsymbol{b})-1]$$

$$\mathrm{L}=\frac{1}{2}(\sum\alpha_i y_i\boldsymbol{x}_i)\left(\sum\alpha_j y_j\boldsymbol{x}_j\right)-\sum\alpha_i y_i\boldsymbol{x}_i\left(\sum\alpha_j y_j\boldsymbol{x}_j\right)-\boxed{\sum\alpha_i y_i}b+\sum\alpha_i$$

$$\mathrm{L}=\sum\alpha_i-\frac{1}{2}\sum_i\sum_j\alpha_i\alpha_j y_i y_j\boxed{\boldsymbol{x}_i\cdot\boldsymbol{x}_j}$$

$$\boxed{\boldsymbol{\omega}=\sum\alpha_i y_i\boldsymbol{x}_i}$$

$$\boxed{-\sum\alpha_i y_i=0}$$

Decision Rule for Prediction

$$\boldsymbol{\omega} \cdot \boldsymbol{x} + \boldsymbol{b} \geq 0, \textit{ then class} +$$

$$\boxed{\boldsymbol{\omega} = \sum \alpha_i y_i \boldsymbol{x}_i}$$

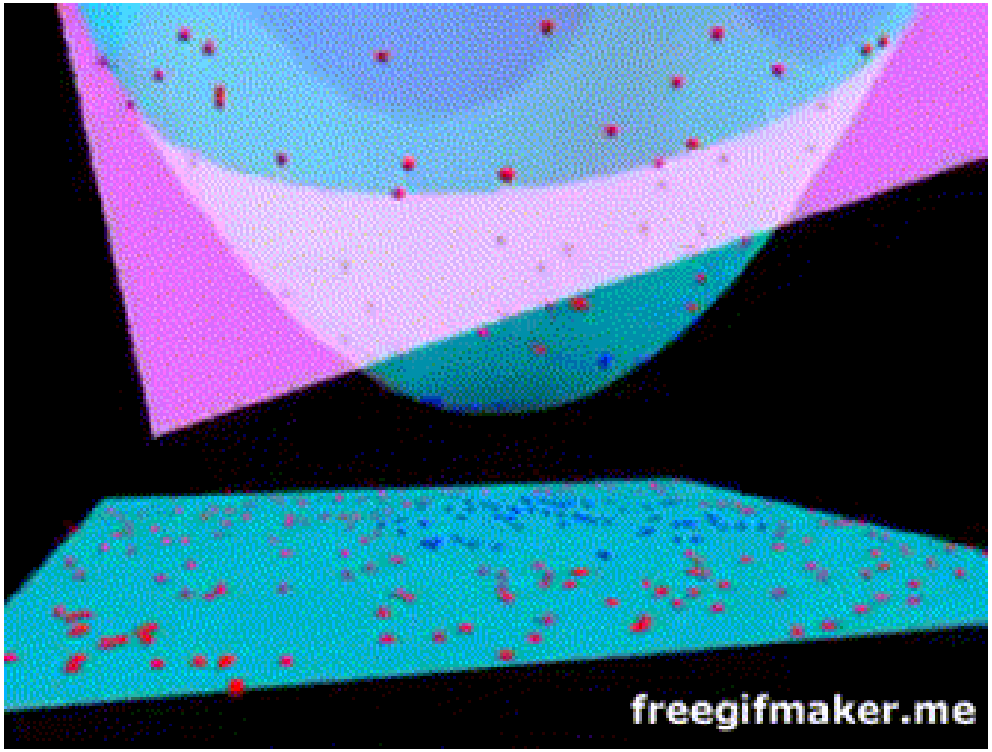
$$\sum \alpha_i \, y_i \boldsymbol{x}_i \cdot \boldsymbol{u} + b \geq 0$$

Class is +

$$\mathrm{L} = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

Non-separable cases

- Let's utilize the fact that both the loss function and the decision boundary only depend on dot products among samples
  - Suppose we have a transformation $\phi(\boldsymbol{x})$
  - We only really need $\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)$
  - Then we only really really need $K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)$
  - K is called a kernel function.


freegifmaker.me

Some Kernel Functions

- No change (linear kernel)
  - $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$
- Polynomial
  - $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1)^{\mathrm{n}}$
- Radial basis function
  - $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}}$

Summary

- Unusual choice of separation strategy:
  - Maximize "street" between groups
- Attack maximization problem:
  - Lagrange multipliers + hairy mathematics
- New problem is a quadratic minimization
  - Susceptible to fancy numerical methods
- Result depends on dot products only
  - Enables use of kernel methods

Credits

- The flow of this SVM lecture goes to
  - Patrick Winston, Professor of Artificial Intelligence
  - Director of MIT Artificial Intelligence Lab (1992-1997)
  - Taught 6.034: Artificial Intelligence

1943-2019

https://ocw.mit.edu/courses/6-034-artificial-intelligence-fall-2010/

Related Topic III

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA; 线性判别分析)

Goal: Find linear projections with good discriminant ability

Input

- The set of c class labels $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$
- The set of n d-dimensional training examples $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_c$ where $\mathcal{D}_i$ consists of n_i training examples $(n = \sum_{i=1}^{c} n_i)$ with label ω_i .

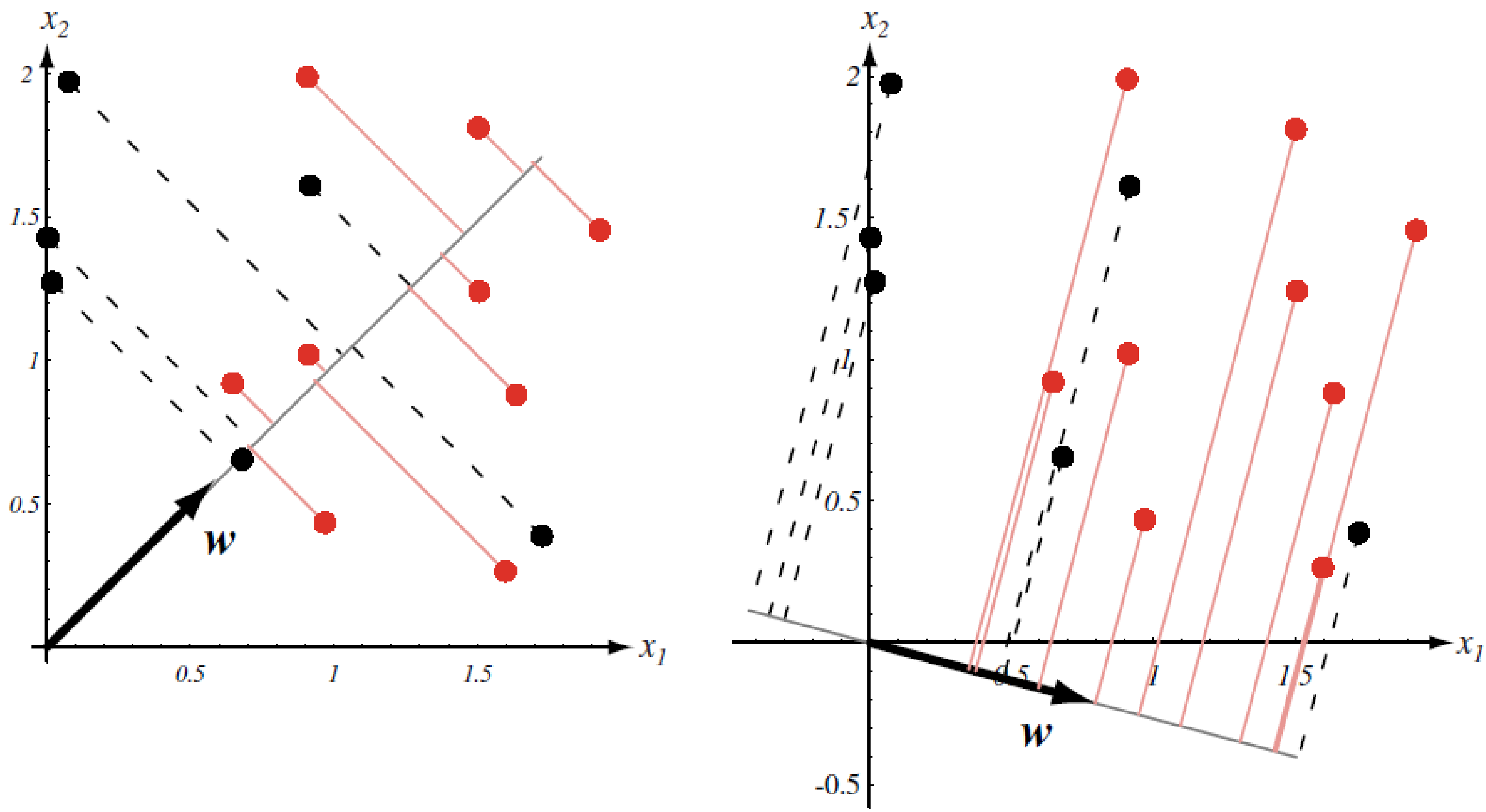
Output

A linear projection direction $\mathbf{w} \in \mathbf{R}^d$ where the projected training examples with different labels are well separated

a.k.a. ***Fisher Discriminant Analysis (FDA)***

Ronald Fisher
Fellow of the Royal Society (FRS)
(1890-1962)

Linear Discriminant Analysis (Cont.)



An illustrative example of LDA *(in binary & 2-dimensional case)*

Linear Discriminant Analysis (Cont.)

Two heuristic principles for good separation

- **Principle 1:** within-class variance should be small

$$\mathbf{m} = \frac{1}{n}\sum_{\mathbf{x}\in\mathcal{D}}\mathbf{x}$$

global sample mean

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in\mathcal{D}_i}\mathbf{x}$$

sample mean for ω_i

$$J_W(\mathbf{w}) = \sum_{i=1}^{c}\left(\sum_{\mathbf{x}\in\mathcal{D}_i}(\mathbf{w}^t\mathbf{x} - \mathbf{w}^t\mathbf{m}_i)^2\right)$$

within-class variance after projection

$$J_W(\mathbf{w}) = \sum\nolimits_{i=1}^{c}\left(\sum\nolimits_{\mathbf{x}\in\mathcal{D}_i}\left(\mathbf{w}^t(\mathbf{x}-\mathbf{m}_i)\right)^2\right)$$

$$= \sum\nolimits_{i=1}^{c}\left(\sum\nolimits_{\mathbf{x}\in\mathcal{D}_i}\mathbf{w}^t(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t\mathbf{w}\right)$$

$$= \sum\nolimits_{i=1}^{c}\left(\mathbf{w}^t\left(\sum\nolimits_{\mathbf{x}\in\mathcal{D}_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t\right)\mathbf{w}\right)$$

$$= \sum\nolimits_{i=1}^{c}\mathbf{w}^t\mathbf{S}_i\mathbf{w} = \mathbf{w}^t\mathbf{S}_W\mathbf{w}$$

$$\mathbf{S}_W = \sum\nolimits_{i=1}^{c}\mathbf{S}_i$$

$$= \sum_{i=1}^{c}\left(\sum_{\mathbf{x}\in\mathcal{D}_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t\right)$$

within-class scatter matrix

(类内散度矩阵)

Linear Discriminant Analysis (Cont.)

Two heuristic principles for good separation

- **Principle 2:** between-class variance should be large

$$\mathbf{m} = \frac{1}{n}\sum_{\mathbf{x}\in\mathcal{D}} \mathbf{x}$$

global sample mean

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in\mathcal{D}_i} \mathbf{x}$$

sample mean for ω_i

$$J_B(\mathbf{w}) = \sum_{i=1}^{c}\left(n_i\left(\mathbf{w}^t\mathbf{m}_i - \mathbf{w}^t\mathbf{m}\right)^2\right)$$

between-class variance after projection

$$\begin{aligned}
J_B(\mathbf{w}) &= \sum\nolimits_{i=1}^{c}\left(n_i\,\left(\mathbf{w}^t(\mathbf{m}_i-\mathbf{m})\right)^2\right)\\
&= \sum\nolimits_{i=1}^{c}\left(n_i\,\mathbf{w}^t(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^t\mathbf{w}\right)\\
&= \mathbf{w}^t\left(\sum\nolimits_{i=1}^{c} n_i(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^t\right)\mathbf{w}\\
&= \mathbf{w}^t\mathbf{S}_B\mathbf{w}
\end{aligned}$$

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^t$$

between-class scatter matrix

(类间散度矩阵)

Linear Discriminant Analysis (Cont.)

Two heuristic principles for good separation

- **Principle 1:** within-class variance should be small
- **Principle 2:** between-class variance should be large

LDA criterion function

$$J(\mathbf{w}) = \frac{J_B(\mathbf{w})}{J_W(\mathbf{w})} = \frac{\mathbf{w}^t\mathbf{S}_B\mathbf{w}}{\mathbf{w}^t\mathbf{S}_W\mathbf{w}}$$

$$\mathbf{S}_W = \sum_{i=1}^{c}\left(\sum_{\mathbf{x}\in\mathcal{D}_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t\right)$$

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^t$$

$$\max_{\mathbf{w}} \frac{\mathbf{w}^t\mathbf{S}_B\mathbf{w}}{\mathbf{w}^t\mathbf{S}_W\mathbf{w}} \quad \Longleftrightarrow \quad \max_{\mathbf{w}} \mathbf{w}^t\mathbf{S}_B\mathbf{w} \quad \mathbf{s.t.}: \mathbf{w}^t\mathbf{S}_W\mathbf{w} = 1$$

Linear Discriminant Analysis (Cont.)

$$\max_{\mathbf{w}} \ \mathbf{w}^t \mathbf{S}_B \mathbf{w}$$

$$\text{s.t.} : \mathbf{w}^t \mathbf{S}_W \mathbf{w} = 1$$

Lagrangian function

$$\mathbf{S}_W = \sum_{i=1}^{c} \left(\sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t \right) \qquad \mathbf{m} = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{D}} \mathbf{x}$$

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t \qquad \mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$$

$$L(\mathbf{w}, \lambda) = \mathbf{w}^t \mathbf{S}_B \mathbf{w} + \lambda (1 - \mathbf{w}^t \mathbf{S}_W \mathbf{w})$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow 2\mathbf{S}_B \mathbf{w} - 2\lambda \mathbf{S}_W \mathbf{w} = \mathbf{0} \Rightarrow \boxed{\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}} \Rightarrow \boxed{\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w} = \lambda \mathbf{w}}$$

generalized eigenvalue problem

if $\mathbf{S}_W$ *is nonsingular*

eigenvalue problem

$$\mathbf{w}^t \mathbf{S}_B \mathbf{w} = \lambda \mathbf{w}^t \mathbf{S}_W \mathbf{w} = \lambda$$

when n is large, $\mathbf{S}_W$ *is generally nonsingular*

- choose largest eigenvalue λ of $\mathbf{S}_W^{-1} \mathbf{S}_B$
- identify eigenvector $\mathbf{w}$ w.r.t. λ , and re-scale $\mathbf{w}$ such that $\mathbf{w}^t \mathbf{S}_W \mathbf{w} = 1$

Linear Discriminant Analysis (Cont.)

A few notes

$$\mathbf{S}_W + \mathbf{S}_B = \sum_{\mathbf{x}\in\mathcal{D}} (\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^t$$

Eq.114 [pp.121]

$$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B = \sum_{\mathbf{x}\in\mathcal{D}} (\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^t$$

total scatter matrix (总体散度矩阵)

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t = \mathbf{AB}$$

$$\mathbf{A} = [n_1(\mathbf{m}_1 - \mathbf{m}), \ldots, n_c(\mathbf{m}_c - \mathbf{m})] \in \mathbf{R}^{d\times c}$$

$$\mathbf{B} = [(\mathbf{m}_1 - \mathbf{m}), \ldots, (\mathbf{m}_c - \mathbf{m})]^t \in \mathbf{R}^{c\times d}$$

Block matrix multiplication
(分块矩阵乘法)

$$\sum\nolimits_{i=1}^{c} n_i(\mathbf{m}_i - \mathbf{m}) = \mathbf{0}$$

the c columns of **A** *are linearly dependent*

$$\mathrm{rank}(\mathbf{A}) \leq c-1$$

$$\mathrm{rank}(\mathbf{S}_B) \leq c-1$$

$$\mathrm{rank}\left(\mathbf{S}_W^{-1}\mathbf{S}_B\right) \leq c-1$$

Linear Discriminant Analysis (Cont.)

$\max_{\mathbf{w}} \ \mathbf{w}^t\mathbf{S}_B\mathbf{w}$

s.t. : $\mathbf{w}^t\mathbf{S}_W\mathbf{w} = 1$

$\Rightarrow \mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w} \xrightarrow{\text{rank}\left(\mathbf{S}_W^{-1}\mathbf{S}_B\right) \leq c-1}$

c-1 **non-zero** eigenvalues λ_i for $\mathbf{S}_W^{-1}\mathbf{S}_B$ along with their orthogonal eigenvectors $\mathbf{w}_i$

LDA

$$\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_c \ (\mathbf{x} \in \mathcal{D} \subset \mathbf{R}^d) \xrightarrow{\text{LDA}} \tilde{\mathcal{D}} = \tilde{\mathcal{D}}_1 \cup \cdots \cup \tilde{\mathcal{D}}_c \ (\tilde{\mathbf{x}} \in \tilde{\mathcal{D}} \subset \mathbf{R}^{c-1})$$

1. Set $\mathbf{m} = \frac{1}{n}\sum_{\mathbf{x}\in\mathcal{D}} \mathbf{x}$ and $\mathbf{m}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in\mathcal{D}_i} \mathbf{x} \ \ (1 \leq i \leq c)$
2. Set $\mathbf{S}_B = \sum_{i=1}^{c} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$ and $\mathbf{S}_W = \sum_{i=1}^{c}\left(\sum_{\mathbf{x}\in\mathcal{D}_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t\right)$
3. Choose the c-1 **non-zero** eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_{c\text{-}1}\}$ for $\mathbf{S}_W^{-1}\mathbf{S}_B$ and identify their orthogonal eigenvectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{c\text{-}1}\}$ with $\mathbf{w}_i^t\mathbf{S}_W\mathbf{w}_i = 1 \ \ (1 \leq i \leq c\text{-}1)$
4. Form the $d \times (c-1)$ linear projection matrix $\mathbf{W}$ by aligning the orthogonal eigenvectors in column, i.e. $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{c-1}]$
5. Set $\tilde{\mathbf{x}} = \mathbf{W}^t\mathbf{x} \ \ (\forall \mathbf{x} \in \mathcal{D})$

Linear Discriminant Analysis (LDA)

Summary

- Linear discriminant functions
  - Model: weight vector & bias (one per class)
- Generalized linear discriminant functions
  - Linear w.r.t. the transformed features
  - Augmented representation
- The two-category case
  - Notation: “normalization” of negative examples
  - Linear separable ➔ Solution region
  - Gradient descent & Newton’s algorithm

Summary (Cont.)

- Criterion function
  - Perceptron criterion function
    - Batch mode, single-sample mode
  - Minimum squared error (MSE) criterion function
    - Inequalities → equalities
    - Pseudo-inverse of matrix **Y**
    - Batch mode, single-sample mode (Widrow-Hoff/LMS rule)
- Multi-category generalization
  - Kesler's construction: multi-category classification → binary classification

Summary (Cont.)

- Principal component analysis (PCA)
  - Linear projections with good representation ability

$$\min_{\mathbf{e}_1,\ldots,\mathbf{e}_{d'}} \quad J_{d'}$$

$$\mathbf{s.t.}:\ \mathbf{e}_i^t\mathbf{e}_i = 1\ (1 \leq i \leq d')$$

$$\mathbf{e}_i^t\mathbf{e}_j = 0\ (i \neq j)$$

$$\mathbf{m} = \frac{1}{n}\sum\nolimits_{k=1}^{n}\mathbf{x}_k$$

sample mean

$$a_{ki} = \mathbf{e}_i^t(\mathbf{x}_k - \mathbf{m})$$

projection of (centered) $\mathbf{x}_k$ *on* $\mathbf{e}_i$

$$J_{d'} = \sum_{k=1}^{n}\left\|\left(\sum_{i=1}^{d'}a_{ki}\mathbf{e}_i\right) - (\mathbf{x}_k - \mathbf{m})\right\|^2$$

PCA criterion function

$$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}\ (\mathbf{x}_k \in \mathbf{R}^d) \xrightarrow{\textbf{PCA}} \tilde{\mathcal{D}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_n\}\ (\boxed{\tilde{\mathbf{x}}_k} \in \mathbf{R}^{d'})$$

Principal component

Summary (Cont.)

- Linear discriminant analysis (LDA)
  - Linear projections with good discrimination ability
    - **Principle 1:** within-class variance should be small
    - **Principle 2:** between-class variance should be large

$$\max_{\mathbf{w}} \ \mathbf{w}^t \mathbf{S}_B \mathbf{w}$$

$$\text{s.t.}: \ \mathbf{w}^t \mathbf{S}_W \mathbf{w} = 1$$

$$\mathbf{S}_W = \sum_{i=1}^{c} \left(\sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t \right) \qquad \mathbf{m} = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{D}} \mathbf{x}$$

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t \qquad \mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$$

$$\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_c \ (\mathbf{x} \in \mathcal{D} \subset \mathbf{R}^d) \xrightarrow{\textbf{LDA}} \tilde{\mathcal{D}} = \tilde{\mathcal{D}}_1 \cup \cdots \cup \tilde{\mathcal{D}}_c \ (\tilde{\mathbf{x}} \in \tilde{\mathcal{D}} \subset \mathbf{R}^{c-1})$$