Returning to classification

Consider (X, Y) where

- X is a random vector in $\mathbb{R}^d$
- $Y \in \{0, \ldots, K-1\}$ is a random variable (depending on X)

Let $h : \mathbb{R}^d \to \{0, \ldots, K-1\}$ be a ***classifier***
with ***probability of error/risk*** given by

$$R(h) := \mathbb{P}[h(X) \neq Y]$$

The ***Bayes classifier*** (denoted $h^\star$) is the optimal classifier, i.e., the classifier with smallest possible risk

We can calculate this explicitly if we know the joint distribution of (X, Y)

The Bayes classifier

Theorem

The classifier $h^\star(\mathbf{x}) := \arg\max_y \eta_y(\mathbf{x})$ satisfies

$$R^\star = R(h^\star) \leq R(h)$$

for any possible classifier h

Recall: $\eta_y(\mathbf{x}) := p_{Y|X}(y|\mathbf{x}) = \mathbb{P}[Y = y|X = \mathbf{x}]$

We can equivalently write $h^\star(\mathbf{x}) = \arg\max_y \pi_y f_{X|Y}(\mathbf{x}|y)$
where $\pi_y = \mathbb{P}[Y = y]$

Generative models and plug-in methods

The Bayes classifier requires knowledge of the joint distribution of (X, Y)

In learning, all we have is the training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$

A generative model is an assumption about the unknown distribution

- usually very simplistic
- often ***parametric***
- build classifier by estimating the parameters via training data
- plug the result into formula for Bayes classifier
  - ***"plug-in" methods***

Linear discriminant analysis (LDA)

In linear discriminant analysis (LDA), we make a (strong) assumption that

$$X|Y = y \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma})$$

for $y = 0, \ldots, K-1$

Here $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate Gaussian/normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

$$\phi(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right\}$$

Note: Each class has the same covariance matrix $\boldsymbol{\Sigma}$

Parameter estimation

In LDA, we assume that the prior probabilities π_y, the mean vectors $\boldsymbol{\mu}_y$, and the covariance matrix Σ are all unknown

To estimate these from the data, we use

$$\widehat{\pi}_y = \frac{|\{i : y_i = y\}|}{n}$$

$$\widehat{\boldsymbol{\mu}}_y = \frac{1}{|\{i : y_i = y\}|} \sum_{i:y_i=y} \mathbf{x}_i$$

$$\widehat{\Sigma} = \frac{1}{n} \sum_{y=0}^{K-1} \sum_{i:y_i=y} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_y)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_y)^T$$

"pooled covariance estimate"

Resulting classifier

The LDA classifier is then

$$\widehat{h}(\mathbf{x}) = \arg\max_y \widehat{\pi}_y \cdot \phi(\mathbf{x}; \widehat{\boldsymbol{\mu}}_y, \widehat{\boldsymbol{\Sigma}})$$

$$= \arg\max_y \log \widehat{\pi}_y + \log \phi(\mathbf{x}; \widehat{\boldsymbol{\mu}}_y, \widehat{\boldsymbol{\Sigma}})$$

$$= \arg\max_y \log \widehat{\pi}_y - \frac{1}{2}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_y)^T \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \widehat{\boldsymbol{\mu}}_y)$$

$$= \arg\min_y \underbrace{(\mathbf{x} - \widehat{\boldsymbol{\mu}}_y)^T \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \widehat{\boldsymbol{\mu}}_y)} - 2 \log \widehat{\pi}_y$$

squared ***Mahalanobis distance***
between $\mathbf{x}$ and $\boldsymbol{\mu}$

$$d_M(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

Example

Suppose that $K = 2$

$$d_M^2(\mathbf{x}; \widehat{\boldsymbol{\mu}}_0, \widehat{\boldsymbol{\Sigma}}) - 2\log\widehat{\pi}_0 \underset{1}{\overset{0}{\lessgtr}} d_M^2(\mathbf{x}; \widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\Sigma}}) - 2\log\widehat{\pi}_1$$

It turns out that by setting

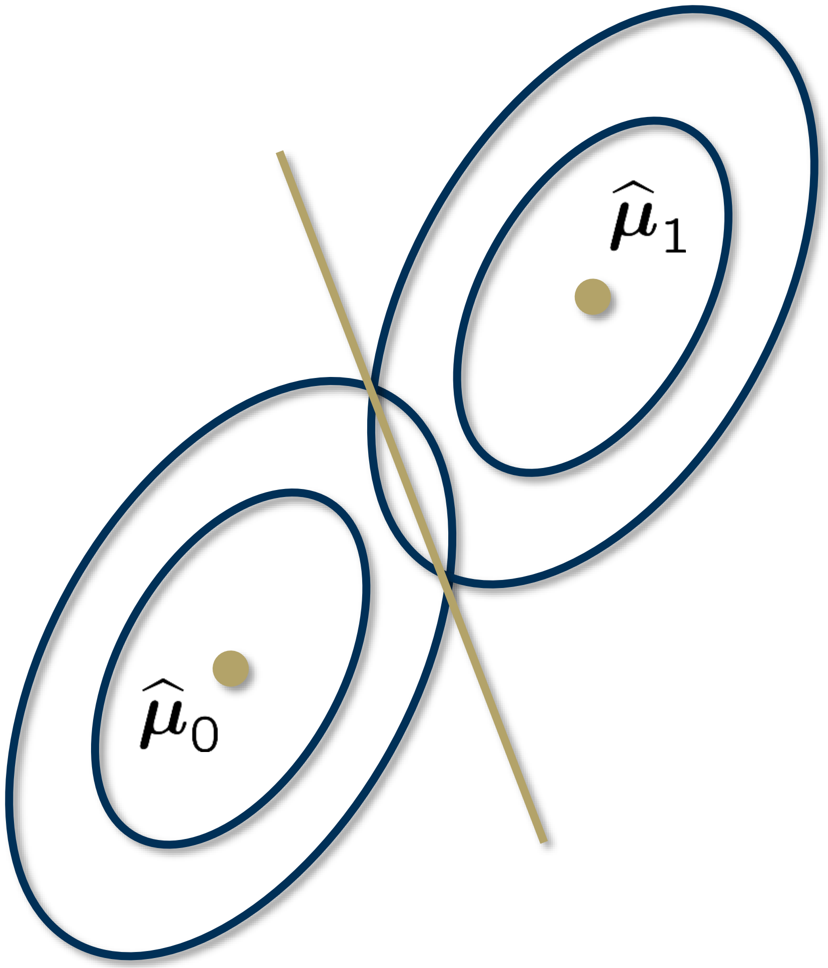
$$\mathbf{w} = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_0)$$

$$b = \tfrac{1}{2}\widehat{\boldsymbol{\mu}}_0^T\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_0 - \tfrac{1}{2}\widehat{\boldsymbol{\mu}}_1^T\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_1 + \log\widehat{\frac{\pi_1}{\pi_0}}$$

we can re-write this as $\mathbf{w}^T\mathbf{x} + b \underset{1}{\overset{0}{\lessgtr}} 0$ *linear classifier*

Example

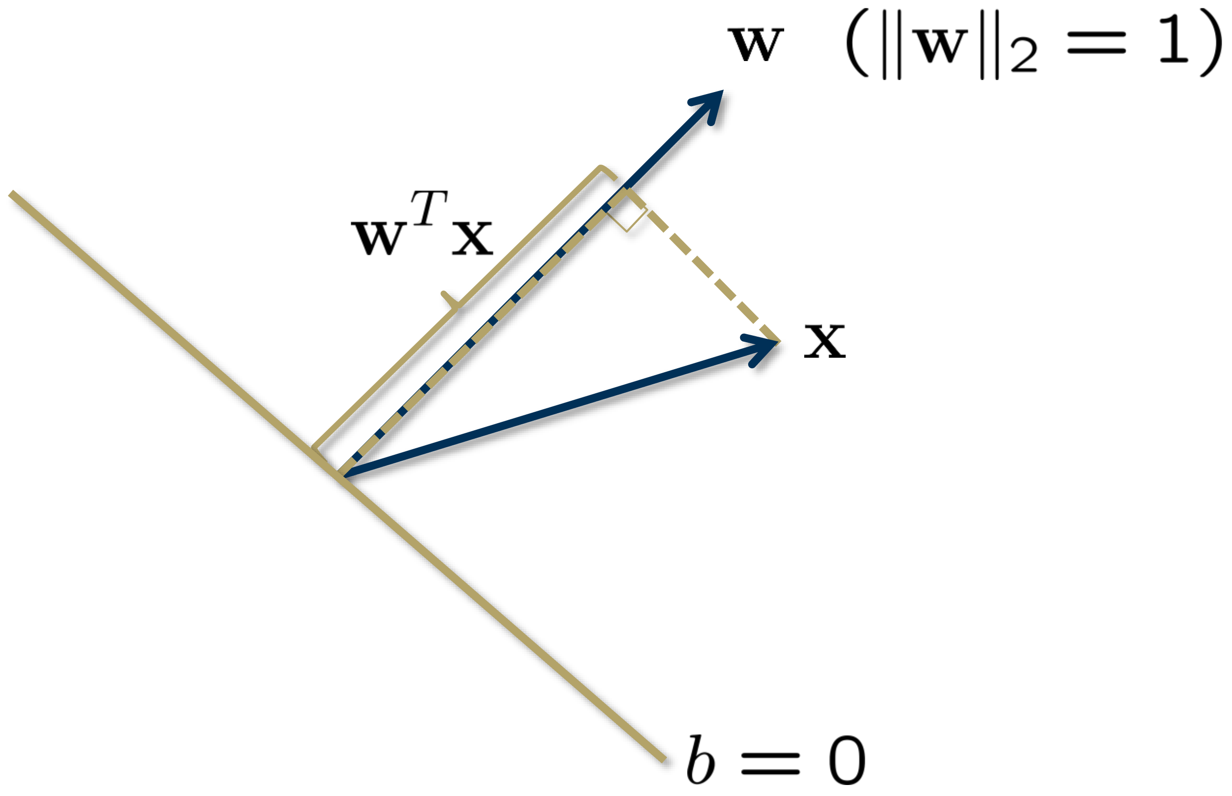
Recall that the contour $\{\mathbf{x} : d_M(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = c\}$ is an ***ellipse***



picture assumes
$\pi_0 = \pi_1$

Linear classifiers

In general, why does $\mathbf{w}^T\mathbf{x} + b \underset{1}{\overset{0}{\lessgtr}} 0$ describe a linear classifier?

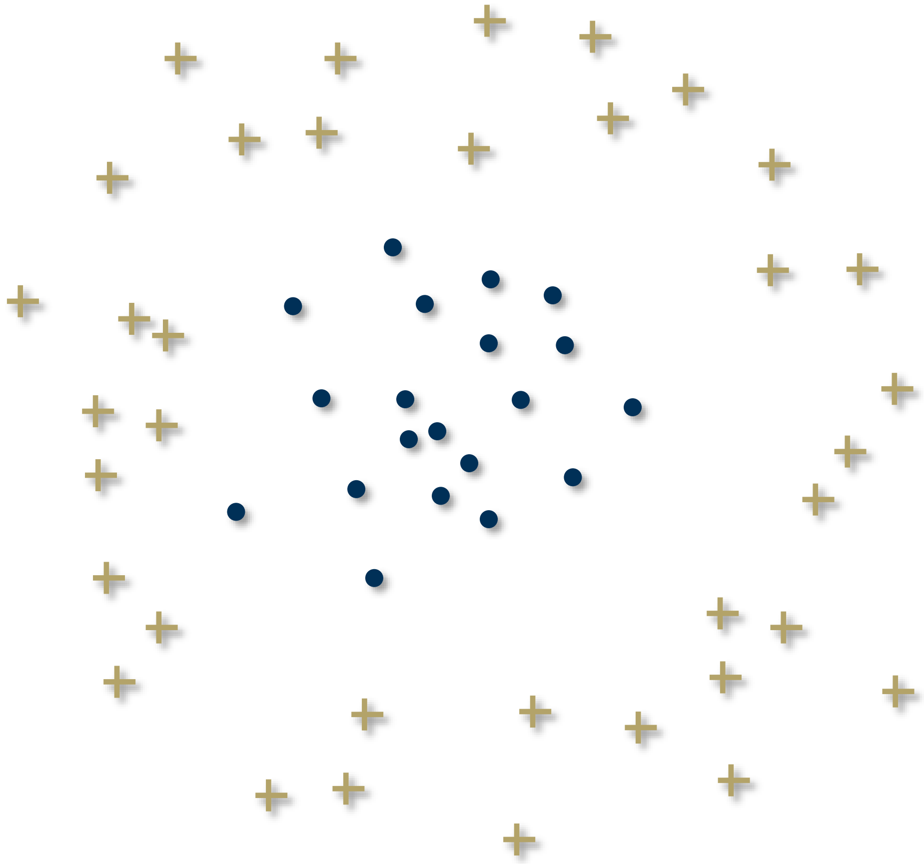


When is LDA appropriate?

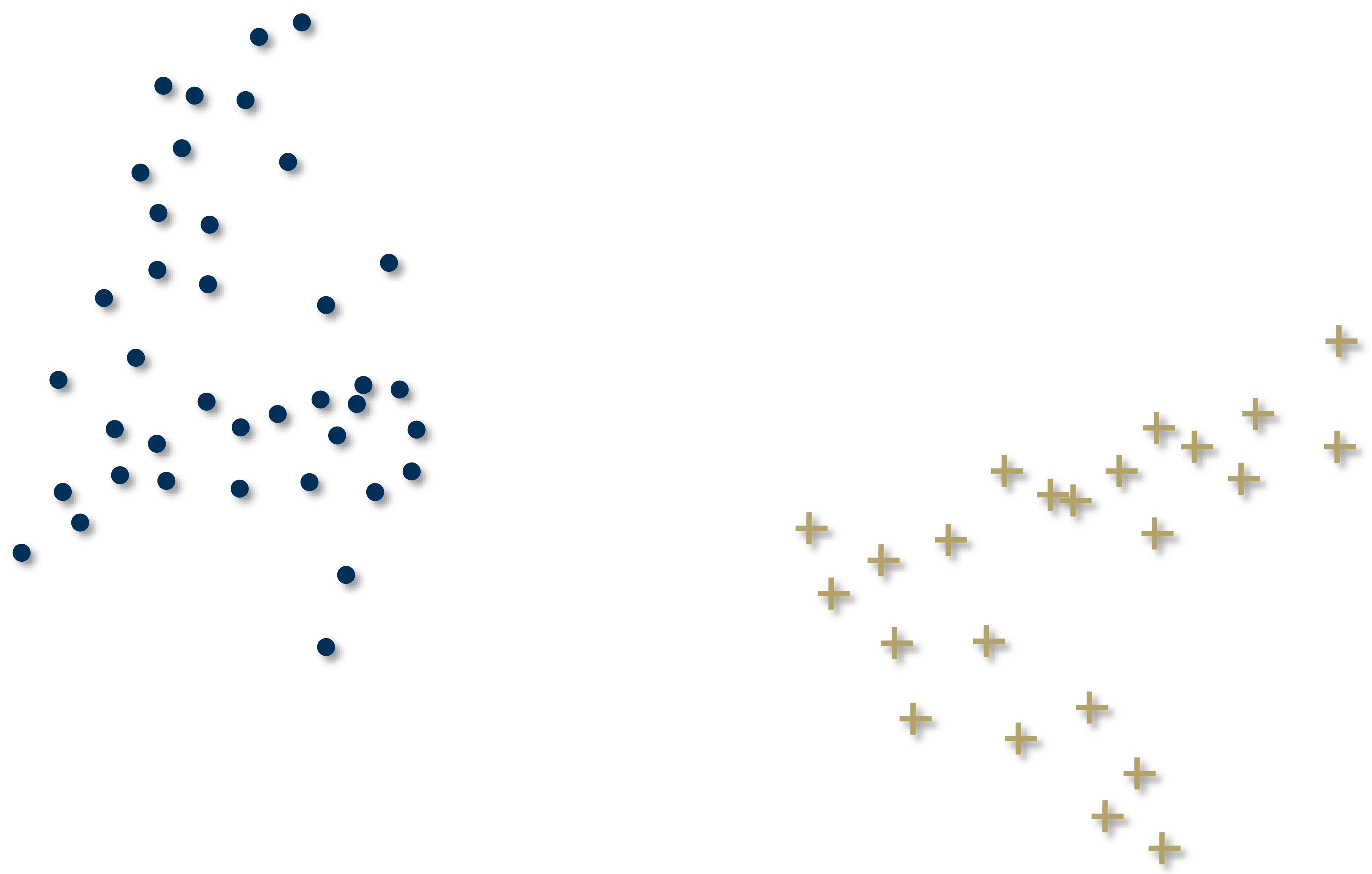
When is LDA appropriate?

When is LDA appropriate?

When is LDA appropriate?

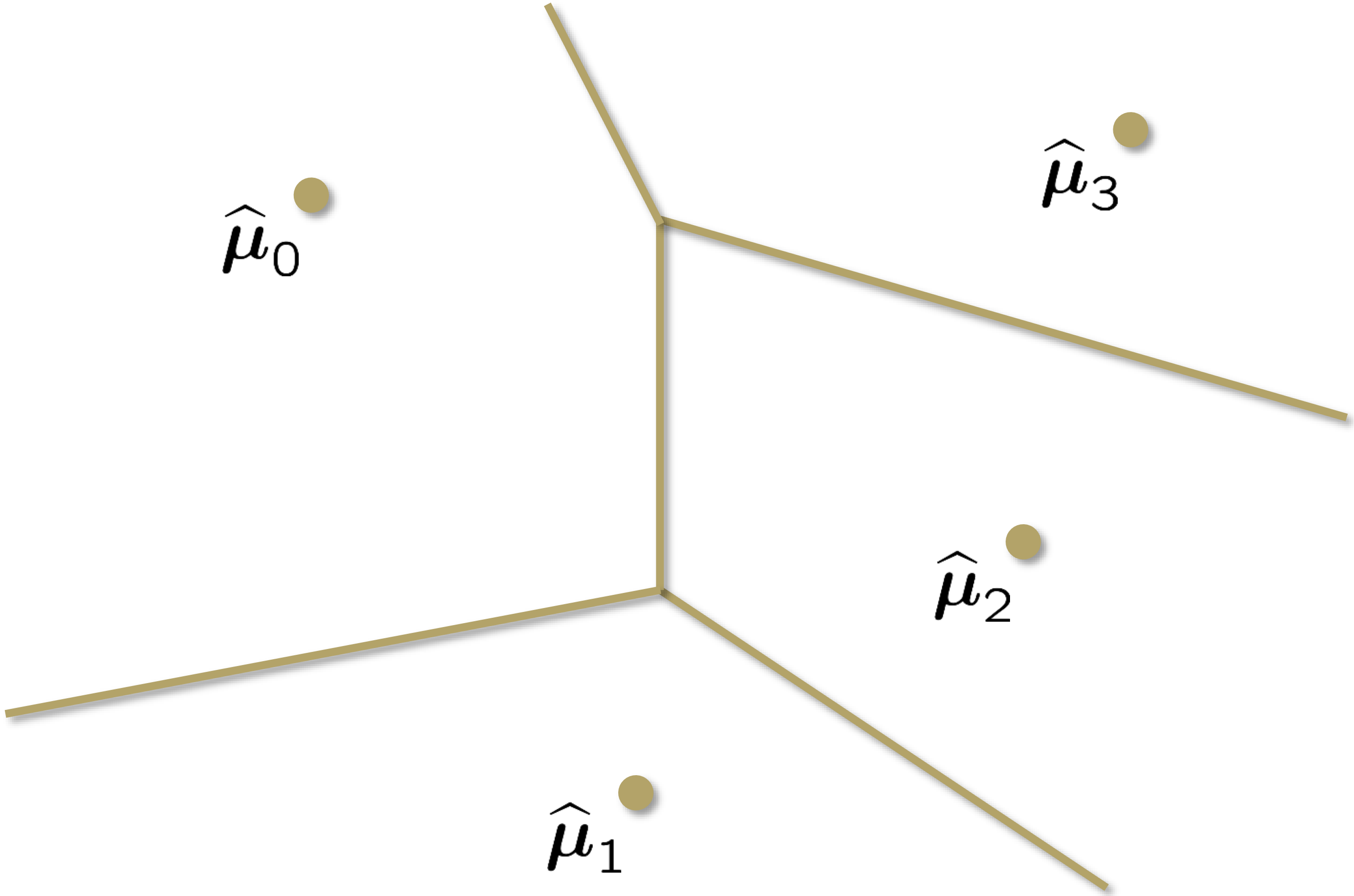


When is LDA appropriate?



More than two classes

The ***decision regions*** are convex polytopes
(intersections of linear half-spaces)



Quadratic discriminant analysis (QDA)

What happens if we expand the generative model to

$$X|Y = y \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$$

for $y = 0, \ldots, K-1$?

Set $\widehat{\boldsymbol{\Sigma}}_y = \frac{1}{|\{i : y_i = y\}|} \sum_{i:y_i=y} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_y)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_y)^T$

Proceed as before, only this case the decision boundaries will be ***quadratic***

Example

Challenges for LDA

The generative model is rarely valid

Moreover, the number of parameters to be estimated is

- class prior probabilities: $K - 1$
- means: Kd
- covariance matrix: $\frac{1}{2}d(d+1)$

If d is small and n is large, then we can accurately estimate these parameters (provably, using Hoeffding and similar)

If n is small and d is large, then we have more parameters than observations, and will likely obtain very poor estimates

- first apply a dimensionality reduction technique to reduce d
- assume a more structured covariance matrix

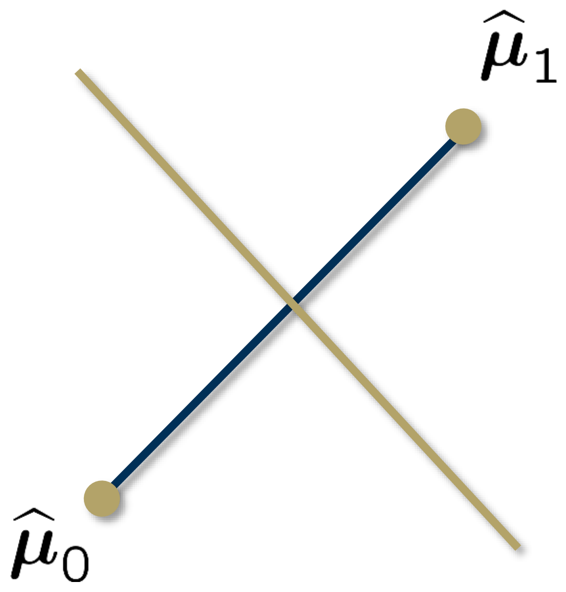
Example

Structured covariance matrix:

Assume $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$ and estimate $\widehat{\sigma}^2 = \frac{1}{d}\text{tr}(\widehat{\boldsymbol{\Sigma}})$

If $K = 2$ and $\widehat{\pi}_0 = \widehat{\pi}_1$, then LDA becomes

$$\frac{1}{\widehat{\sigma}^2}\|\mathbf{x} - \widehat{\boldsymbol{\mu}}_0\|_2^2 \underset{1}{\overset{0}{\lessgtr}} \frac{1}{\widehat{\sigma}^2}\|\mathbf{x} - \widehat{\boldsymbol{\mu}}_1\|_2^2 \iff \|\mathbf{x} - \widehat{\boldsymbol{\mu}}_0\|_2^2 \underset{1}{\overset{0}{\lessgtr}} \|\mathbf{x} - \widehat{\boldsymbol{\mu}}_1\|_2^2$$



nearest centroid classifier

Another possible escape

Recall from the very beginning of the lecture that the Bayes classifier can be stated either in terms of maximizing $\pi_y f_{X|Y}(\mathbf{x}|y)$ or $\eta_y(\mathbf{x})$

In LDA, we are estimating $\pi_y f_{X|Y}(\mathbf{x}|y)$, which is equivalent to the full joint distribution of (X, Y)

All we ***really*** need is to be able to estimate $\eta_y(\mathbf{x})$

- we don't need to know $f_X(\mathbf{x})$

LDA commits one of the cardinal sins of machine learning:

Never solve a more difficult problem as an intermediate step

Can we do better?

Another look at plugin methods

Suppose $K = 2$

Define $\eta(\mathbf{x}) = \eta_1(\mathbf{x})$
$$= 1 - \eta_0(\mathbf{x})$$

In this case, another way to express the Bayes classifier is as

$$h^\star(\mathbf{x}) = \begin{cases} 1 & \text{if } \eta(\mathbf{x}) \geq 1/2 \\ 0 & \text{if } \eta(\mathbf{x}) < 1/2 \end{cases}$$

Note that we do not actually need to know the full distribution of (X, Y) to express the Bayes classifier

All we really need is to decide if $\eta(\mathbf{x}) \geq 1/2$

Gaussian case

Suppose that $K = 2$ and that $X|Y = y \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma})$

$$\eta(\mathbf{x}) = \frac{\pi_1 \phi(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma})}{\pi_1 \phi(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + \pi_0 \phi(\mathbf{x}; \boldsymbol{\mu}_0, \boldsymbol{\Sigma})}$$

$$= \frac{\pi_1 e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)}}{\pi_1 e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)} + \pi_0 e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)}}$$

$$= \frac{1}{1 + \frac{\pi_0}{\pi_1} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)}}$$

$$= \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}$$

Logistic regression

This observation gives rise to another class of plugin methods, the most important of which is logistic regression, which implements the following strategy

1. Assume $\eta(\mathbf{x}) = \dfrac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$ $(\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R})$

2. Directly estimate $\mathbf{w}, b$ (somehow) from the data

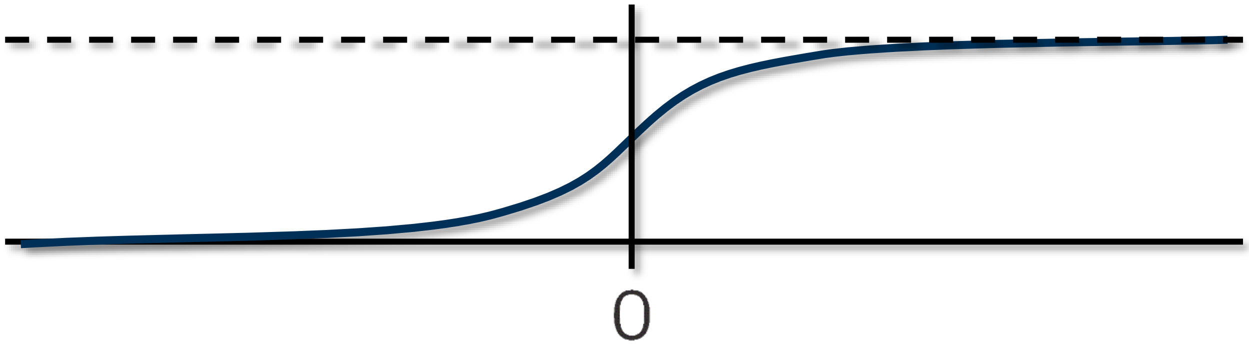
3. Plug the estimate

$$\widehat{\eta}(\mathbf{x}) = \frac{1}{1 + e^{-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})}}$$

   into the formula for the Bayes classifier

The logistic function

The function $\frac{1}{1+e^{-t}}$ is called a ***logistic*** function (or a ***sigmoid*** function in other contexts)



The logistic regression classifier

Denote the logistic regression classifier by

$$\widehat{h}(\mathbf{x}) = 1_{\left\{\widehat{\eta}(\mathbf{x}) \geq 1/2\right\}}(\mathbf{x})$$

Note that $\widehat{h}(\mathbf{x}) = 1 \iff \widehat{\eta}(\mathbf{x}) \geq \frac{1}{2}$

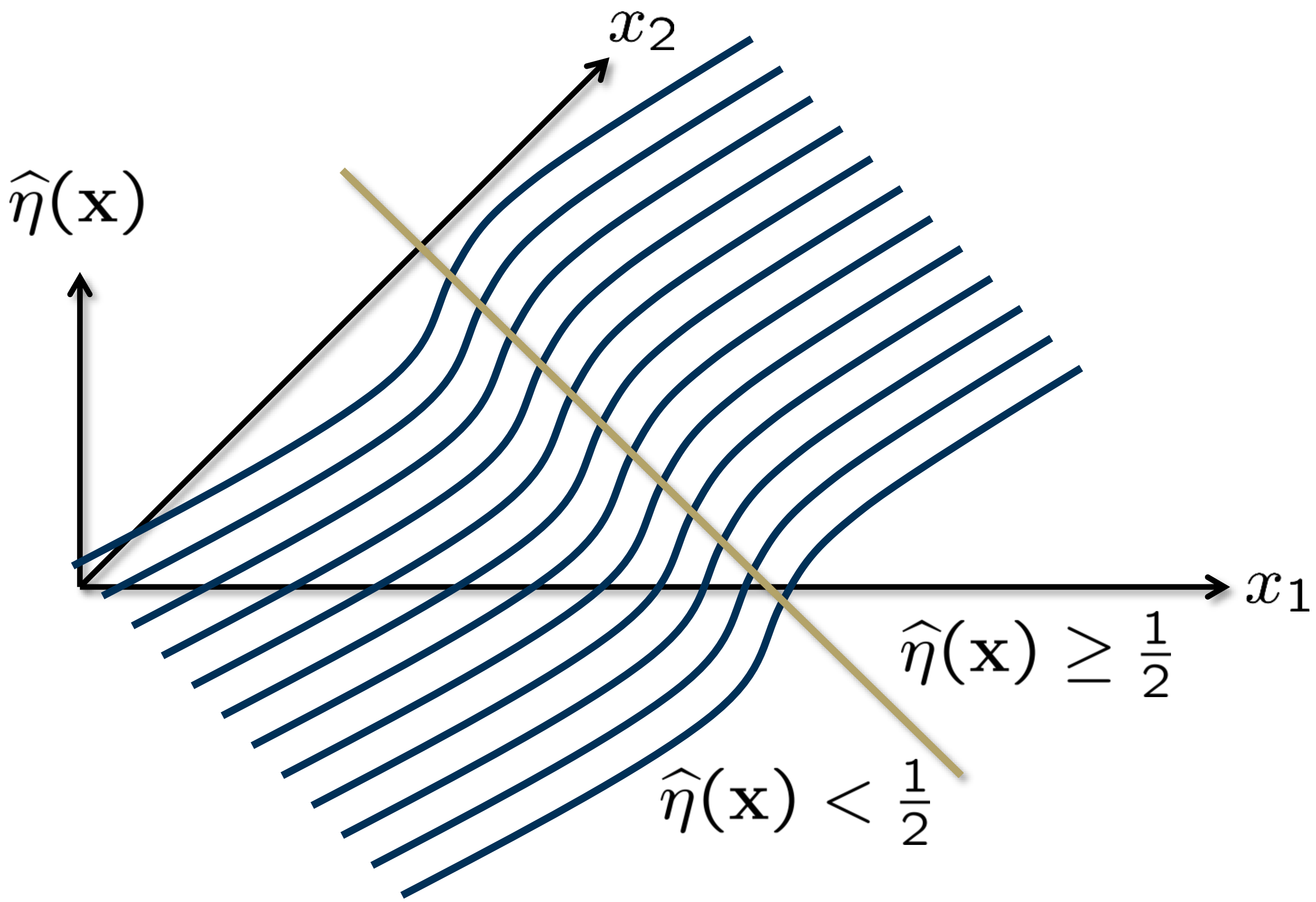
$$\iff \frac{1}{1+\exp\left(-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})\right)} \geq \frac{1}{2}$$

$$\iff \exp\left(-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})\right) \leq 1$$

$$\iff (\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b}) \geq 0$$

So $\widehat{h}(\mathbf{x}) = \begin{cases} 1 & \text{if } \widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b} \geq 0 \\ 0 & \text{otherwise} \end{cases}$ ***linear classifier***

Example



x_2
$\widehat{\eta}(\mathbf{x})$
x_1
$\widehat{\eta}(\mathbf{x}) \geq \frac{1}{2}$
$\widehat{\eta}(\mathbf{x}) < \frac{1}{2}$

Estimating the parameters

Challenge: How to estimate the parameters for

$$\eta(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}$$

One possibility: $\mathbf{w} = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_0)$

$$b = \frac{1}{2}\widehat{\boldsymbol{\mu}}_0^T \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_0 - \frac{1}{2}\widehat{\boldsymbol{\mu}}_1^T \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_1 + \log \frac{\widehat{\pi_1}}{\widehat{\pi_0}}$$

Alternative: ***Maximum likelihood estimation***

For convenience, set $\boldsymbol{\theta} = (b, \mathbf{w})$

Note that $\eta(\mathbf{x})$ is really a function of both $\mathbf{x}$ and $\boldsymbol{\theta}$, so we will use the notation $\eta(\mathbf{x}; \boldsymbol{\theta})$ to highlight this dependence

The *a posteriori* probability of our data

Suppose that we knew $\boldsymbol{\theta}$. Then we could compute

$$
\begin{aligned}
\mathbb{P}[y_i|\mathbf{x}_i;\boldsymbol{\theta}] &= \mathbb{P}[Y_i = y_i|X_i = \mathbf{x}_i;\boldsymbol{\theta}] \\
&= \begin{cases} \eta(\mathbf{x}_i;\boldsymbol{\theta}) & \text{if } y_i = 1 \\ 1 - \eta(\mathbf{x}_i;\boldsymbol{\theta}) & \text{if } y_i = 0 \end{cases} \\
&= \eta(\mathbf{x}_i;\boldsymbol{\theta})^{y_i}(1-\eta(\mathbf{x}_i;\boldsymbol{\theta}))^{1-y_i}
\end{aligned}
$$

Because of independence, we also have that

$$
\begin{aligned}
\mathbb{P}[y_1,\ldots,y_n|\mathbf{x}_1,\ldots\mathbf{x}_n;\boldsymbol{\theta}] &= \prod_{i=1}^{n} \mathbb{P}[y_i|\mathbf{x}_i;\boldsymbol{\theta}] \\
&= \prod_{i=1}^{n} \eta(\mathbf{x}_i;\boldsymbol{\theta})^{y_i}(1-\eta(\mathbf{x}_i;\boldsymbol{\theta}))^{1-y_i}
\end{aligned}
$$

Maximum likelihood estimation

We don't actually know $\boldsymbol{\theta}$, but we do know $y_1, \ldots, y_n$

Suppose we view $y_1, \ldots, y_n$ to be fixed, and view $\mathbb{P}[y_1, \ldots, y_n | \mathbf{x}_1, \ldots \mathbf{x}_n; \boldsymbol{\theta}]$ as just a function of $\boldsymbol{\theta}$

When we do this, $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{P}[y_1, \ldots, y_n | \mathbf{x}_1, \ldots \mathbf{x}_n; \boldsymbol{\theta}]$ is called the ***likelihood*** (or likelihood function)

The method of ***maximum likelihood*** aims to estimate $\boldsymbol{\theta}$ by finding the $\boldsymbol{\theta}$ that ***maximizes*** the ***likelihood*** $\mathcal{L}(\boldsymbol{\theta})$

In practice, it is often more convenient to focus on maximizing the ***log-likelihood***, i.e., $\log \mathcal{L}(\boldsymbol{\theta})$

The log-likelihood

To see why, note that the likelihood in our case is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{n} \eta(\mathbf{x}_i; \boldsymbol{\theta})^{y_i} (1 - \eta(\mathbf{x}_i; \boldsymbol{\theta}))^{1-y_i}$$

Thus, the log-likelihood is given by

$$\begin{aligned}
\ell(\boldsymbol{\theta}) &= \log \mathcal{L}(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{n} y_i \log \eta(\mathbf{x}_i; \boldsymbol{\theta}) + (1 - y_i) \log(1 - \eta(\mathbf{x}_i; \boldsymbol{\theta})) \\
&= \sum_{i=1}^{n} y_i \boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i - \log(1 + e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})
\end{aligned}$$

$$\widetilde{\mathbf{x}} = [1, x(1), \ldots, x(d)]^T \quad \boldsymbol{\theta} = [b, w(1), \ldots, w(d)]^T$$

Maximizing the log-likelihood

How can we maximize

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i - \log(1 + e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})$$

with respect to $\boldsymbol{\theta}$?

Find a $\boldsymbol{\theta}$ such that $\nabla \ell(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_{d+1}} \end{bmatrix} = 0$

(i.e., compute the partial derivatives and set them to zero)

Computing the gradient

It is not too hard to show that

$$\nabla \ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \nabla \left(y_i \boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i - \log(1 + e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i}) \right)$$

$$= \sum_{i=1}^{n} \widetilde{\mathbf{x}}_i \left(y_i - \frac{1}{1 + e^{-\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i}} \right) = 0$$

This gives us $d + 1$ equations, but they are ***nonlinear*** and have no closed-form solution

How can we solve this problem?

Optimization

Throughout signal processing and machine learning, we will very often encounter problems of the form

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \, f(\mathbf{x})$$

(or $\underset{\boldsymbol{\theta} \in \mathbb{R}^{d+1}}{\text{minimize}} -\ell(\boldsymbol{\theta})$ for today)

In many (most?) cases, we cannot compute the solution simply by setting $\nabla f(\mathbf{x}) = 0$ and solving for $\mathbf{x}$

However, there are many powerful ***algorithms*** for finding $\mathbf{x}$ using a computer