The Bayes classifier

Consider (X, Y) where

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

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

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

The ***Bayes classifier*** (denoted h^*) 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)

Characterizations of the joint distribution

Let $f_{X,Y}(\mathbf{x}, k)$ denote the joint distribution of (X, Y)

- We can factor this as $f_{X,Y}(\mathbf{x}, k) = \pi_k f_{X|Y}(\mathbf{x}|k)$
  - $\pi_k = \mathbb{P}[Y = k]$ is the ***a priori probability*** of class k
  - $f_{X|Y}(\mathbf{x}|k)$ is the class conditional pdf (the pdf of $X|Y = k$)

- Alternatively (or via Bayes rule), we can factor this as $f_{X,Y}(\mathbf{x}, k) = \eta_k(\mathbf{x}) f_X(\mathbf{x})$
  - $\eta_k(\mathbf{x}) = \mathbb{P}[Y = k|X = \mathbf{x}]$ is the ***a posteriori probability*** of class k
  - $f_X(\mathbf{x})$ is simply the marginal pdf of X

The Bayes classifier

Theorem

The classifier $h^*(\mathbf{x}) := \arg\max_k \eta_k(\mathbf{x})$ satisfies

$$R(h^*) = \min R(h)$$

where the min is over all possible classifiers.

To calculate the Bayes classifier/Bayes risk, we need to know $\eta_k(\mathbf{x})$

Alternatively, since $\pi_k f_{X|Y}(\mathbf{x}|k) = \eta_k(\mathbf{x}) f_X(\mathbf{x})$, to find the maximum $\eta_k(\mathbf{x})$ it is sufficient to know $\pi_k f_{X|Y}(\mathbf{x}|k)$

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***

A first plugin method: Naïve Bayes

The Naïve Bayes classifier is one common approach based on estimating the distribution of the data and then plugging this into the Bayes classifier

Makes a (probably ***naïve***) assumption:

Let $X = [X(1), \ldots, X(d)]^T \in \mathbb{R}^d$ denote the random feature vector in a classification problem and Y the corresponding label

The naïve Bayes classifier assumes that, given Y, $X(1), \ldots, X(d)$ are ***independent***

Sometimes known as “Idiot Bayes”

What does the NB assumption buy us?

The major advantage of NB is that we only need to estimate ***scalar/univariate*** densities

Let $f_{X|Y}(\mathbf{x}|k)$ be the probability law (density or mass function) of $X|Y = k, k = 1, \ldots, K$

By the NB assumption

$$f_{X|Y}(\mathbf{x}|k) = \prod_{j=1}^{d} f_{X(j)|Y}(x(j)|k)$$

where $f_{X(j)|Y}(x(j)|k)$ denotes the probability law of $X(j)|Y = k$

Naïve Bayes classifier

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be training data and set

- $\widehat{\pi}_k = \dfrac{|\{i : y_i = k\}|}{n}$
- $\widehat{f}_{X(j)|Y}(x(j)|k) =$ any estimate of $f_{X(j)|Y}(x(j)|k)$ based on $\{x_i(j) : y_i = k\}$

The NB classifier is then given by

$$\widehat{h}(\mathbf{x}) = \arg\max_k \ \widehat{\pi}_k \prod_{j=1}^d \widehat{f}_{X(j)|Y}(x(j)|k)$$

So, how can we determine $\widehat{f}_{X(j)|Y}(x(j)|k)$?

Continuous features

Suppose that $X(j)|Y = k$ is a continuous random variable

Options for estimating $\widehat{f}_{X(j)|Y}(x(j)|k)$ include

- parametric estimate (e.g., Gaussian MLE)
- kernel density estimate
- quantize to a discrete random variable

Discrete features

Now suppose that $X(j)|Y = k$ takes only the values $z_1, \ldots, z_L$

Define $n_k = |\{i : y_i = k\}|$

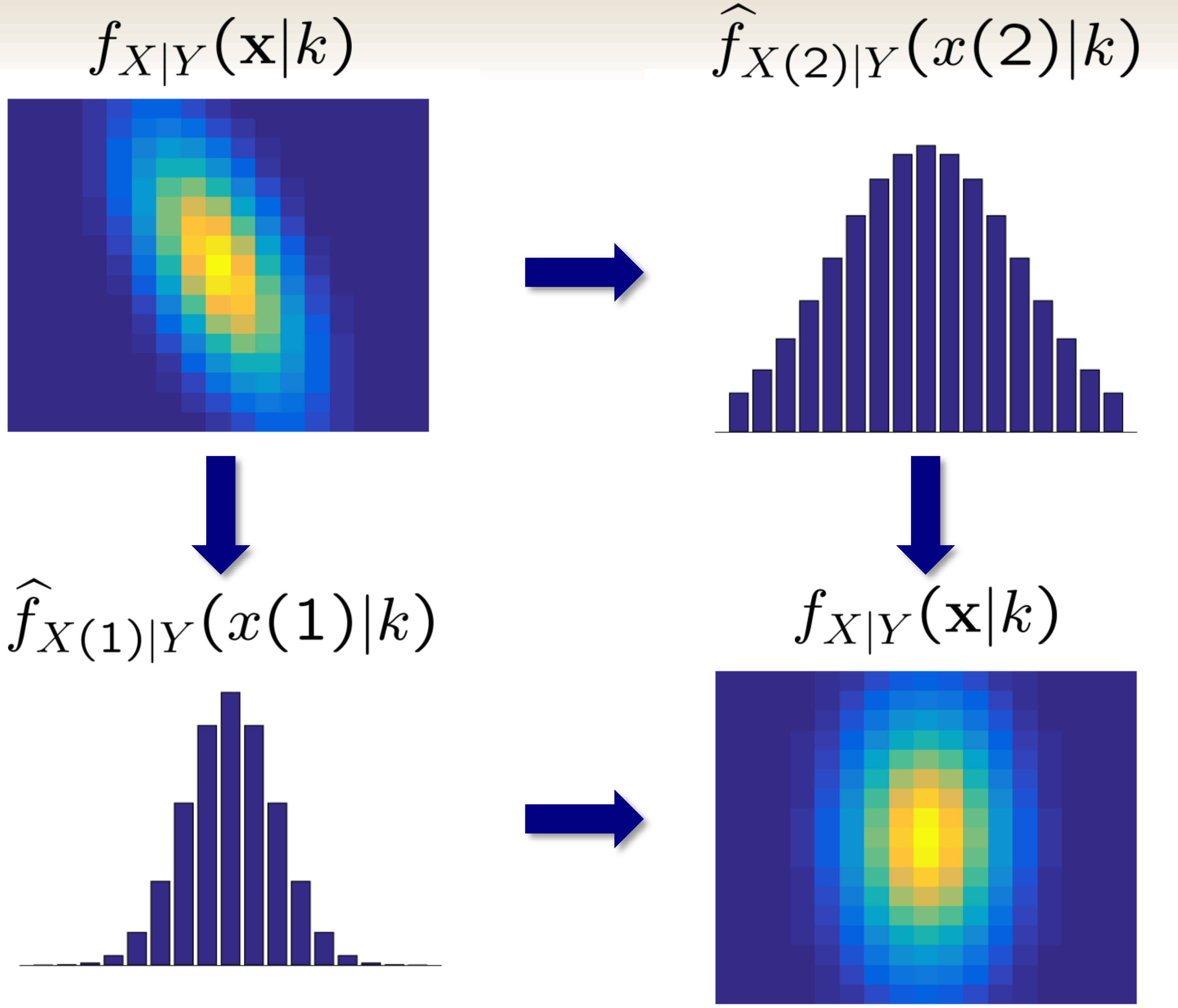
$$n_k(j, z_\ell) = |\{i : y_i = k \text{ and } x_i(j) = z_\ell\}|$$

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

Then a natural (maximum likelihood) estimate of $\mathbb{P}[X(j) = z_\ell | Y = k]$ is

$$\widehat{f}_{X(j)|Y}(z_\ell|k) = \frac{n_k(j, z_\ell)}{n_k}$$

Example


$f_{X|Y}(\mathbf{x}|k)$
$\widehat{f}_{X(2)|Y}(x(2)|k)$
$\widehat{f}_{X(1)|Y}(x(1)|k)$
$f_{X|Y}(\mathbf{x}|k)$

Example: Document classification

Suppose we wish to classify documents into categories

- 0 = politics
- 1 = sports
- 2 = finance
- …

One simple (but useful) way to represent a document is as $\mathbf{x} = [x(1), \ldots, x(d)]^T$ with d representing the number of words in our "vocabulary" and $x(j)$ representing the number of times word j occurs in the document

"bag of words" model

Multinomial Naïve Bayes

We can think of each document as consisting of n words which are distributed among the d choices in the vocabulary independently at random

- *multinomial distribution*

All that is required in this model is to estimate the probability of each word under the different categories:

- For each class k, compute the number of times each word appears (across all documents); denote this $n_k(\text{word})$
- Compute the estimate

$$\mathbb{P}(\text{word}|k) = \frac{n_k(\text{word})}{\sum_{j=1}^{d} n_k(\text{word}_j)}$$

Multinomial Naïve Bayes

With this estimate, for a new document summarized by the bag-of-words model $\mathbf{x} = [x(1), \ldots, x(d)]^T$, we can compute

$$\begin{aligned} \mathbb{P}(\mathbf{x}|k) &= \prod_{j=1}^{d} \mathbb{P}(x(j)|k) \\ &= \prod_{j=1}^{d} \mathbb{P}(\mathsf{word}_j|k)^{x(j)} \end{aligned}$$

This gives us the classifier

$$\widehat{h}(\mathbf{x}) = \arg\max_k \ \widehat{\pi}_k \prod_{j=1}^{d} \mathbb{P}(\mathsf{word}_j|k)^{x(j)}$$

Undesirable property

It is possible that after training our classifier, we may be given an input $\mathbf{x}$ where some $x(j) \neq 0$ for some word that was not observed in the training data for some class k

For such a class, our previous estimate would yield $\mathbb{P}(\text{word}_j|k) = 0$, and hence

$$\mathbb{P}(\mathbf{x}|k) = \prod_{j=1}^{d} \mathbb{P}(\text{word}_j|k)^{x(j)} = 0$$

In this case, class k will never be predicted

This is rather unfortunate…

Laplace smoothing

It could happen that every training document in "sports" contains the word "ball"

What happens when we get an article about hockey?

To avoid this problem, it is common to instead use

$$\mathbb{P}(\text{word}|k) = \frac{n_k(\text{word}) + 1}{\sum_{j=1}^{d} n_k(\text{word}_j) + d}$$

We can think of this as the result of adding d "pseudo-samples" to our data to ensure that each word occurs at least once

Related to Laplace's rule of succession: *Lapalce smoothing* (Bayesian estimate with a Dirichlet prior)

Other variants

- Bernoulli Naïve Bayes
  - used when the features are binary-valued
  - example: word occurrence vectors (vs word count vectors)
  - simply need to estimate probability of 1 vs 0 for each feature
- Gaussian Naïve Bayes
  - models continuous features as univariate Gaussian densities
  - estimates mean and variance of data to fit a Gaussian to each feature
- Many other extensions
  - Gaussian mixture models
  - kernel density estimates
  - …

Linear discriminant analysis (LDA)

In linear discriminant analysis (LDA), we make a different assumption than Naïve Bayes

Now, we do not require the features to be independent, but we make the (strong) assumption that

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

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

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_k, the mean vectors $\boldsymbol{\mu}_k$, and the covariance matrix $\boldsymbol{\Sigma}$ are all unknown

To estimate these from the data, we use

$$\widehat{\pi}_k = \frac{|\{i : y_i = k\}|}{n}$$

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

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

"pooled covariance estimate"

Resulting classifier

The LDA classifier is then

$$\begin{aligned}
\widehat{h}(\mathbf{x}) &= \arg\max_k \widehat{\pi}_k \cdot \phi(\mathbf{x}; \widehat{\boldsymbol{\mu}}_k, \widehat{\boldsymbol{\Sigma}}) \\
&= \arg\max_k \log \widehat{\pi}_k + \log \phi(\mathbf{x}; \widehat{\boldsymbol{\mu}}_k, \widehat{\boldsymbol{\Sigma}}) \\
&= \arg\max_k \log \widehat{\pi}_k - \frac{1}{2}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_k)^T \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \widehat{\boldsymbol{\mu}}_k) \\
&= \arg\min_k \underbrace{(\mathbf{x} - \widehat{\boldsymbol{\mu}}_k)^T \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \widehat{\boldsymbol{\mu}}_k)} - 2 \log \widehat{\pi}_k
\end{aligned}$$

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

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

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

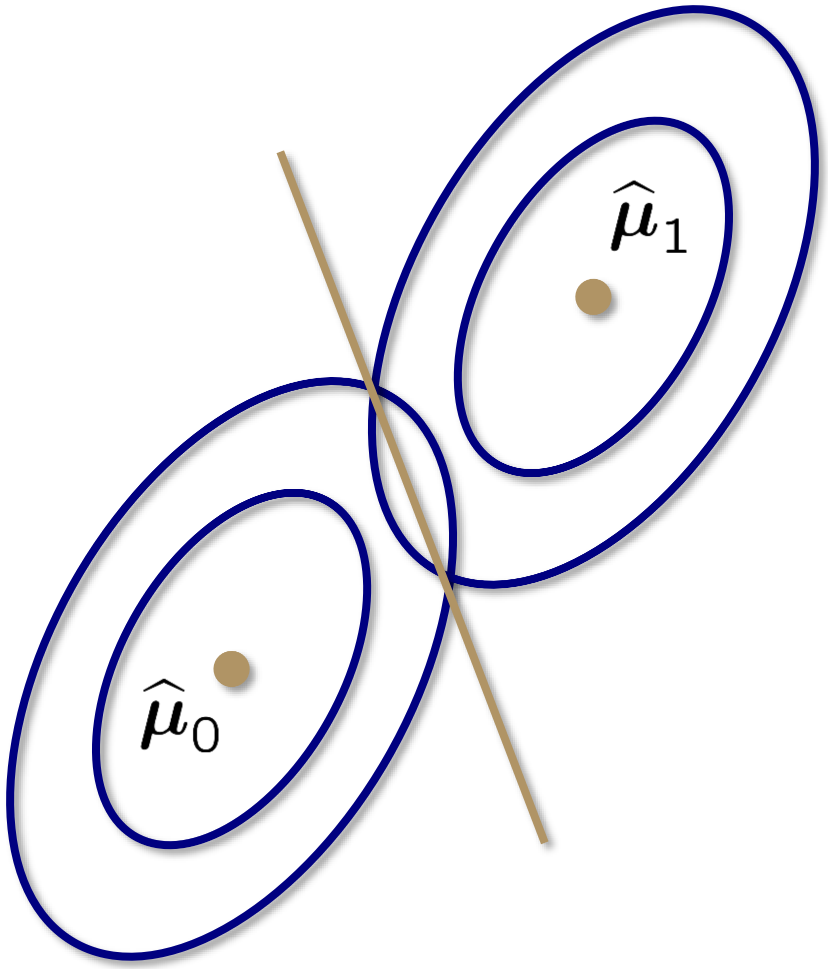
we can re-write this as

$$\mathbf{w}^T\mathbf{x} + b \underset{1}{\overset{0}{\lessgtr}} 0$$

linear classifier

Example

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

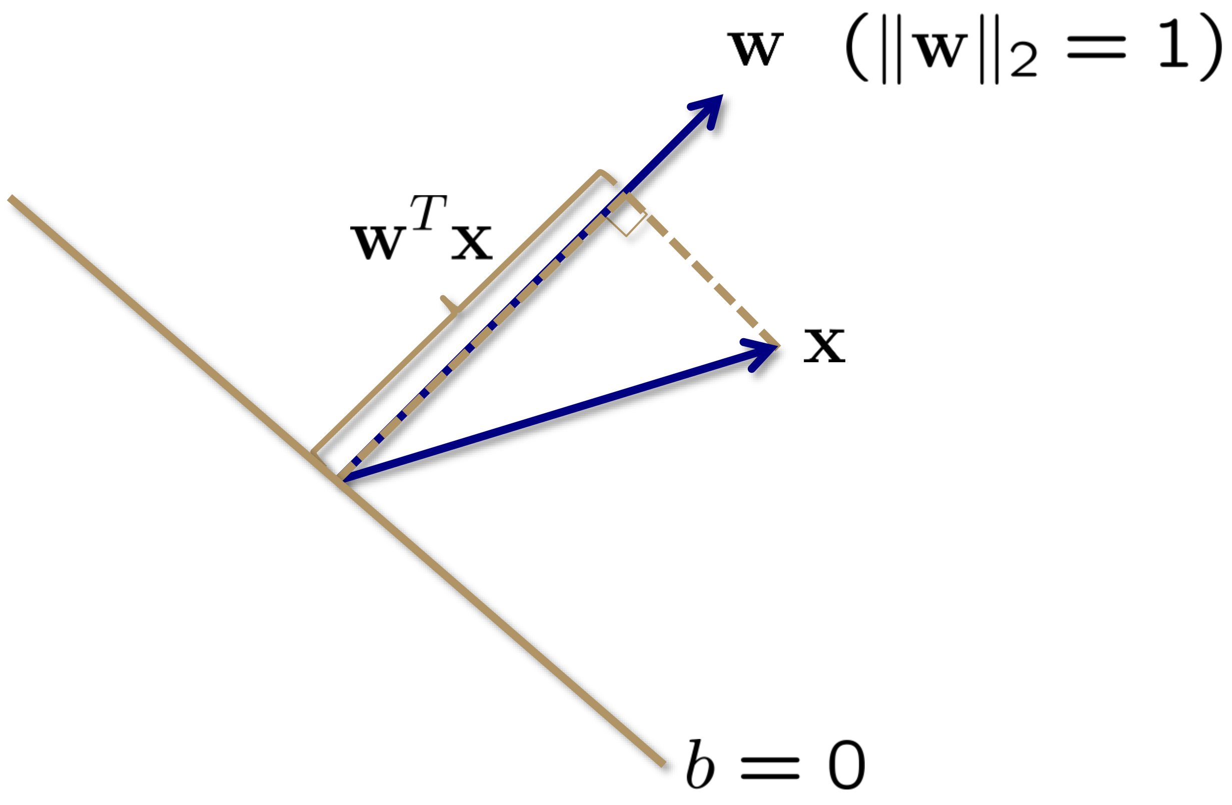


$\widehat{\boldsymbol{\mu}}_1$
$\widehat{\boldsymbol{\mu}}_0$

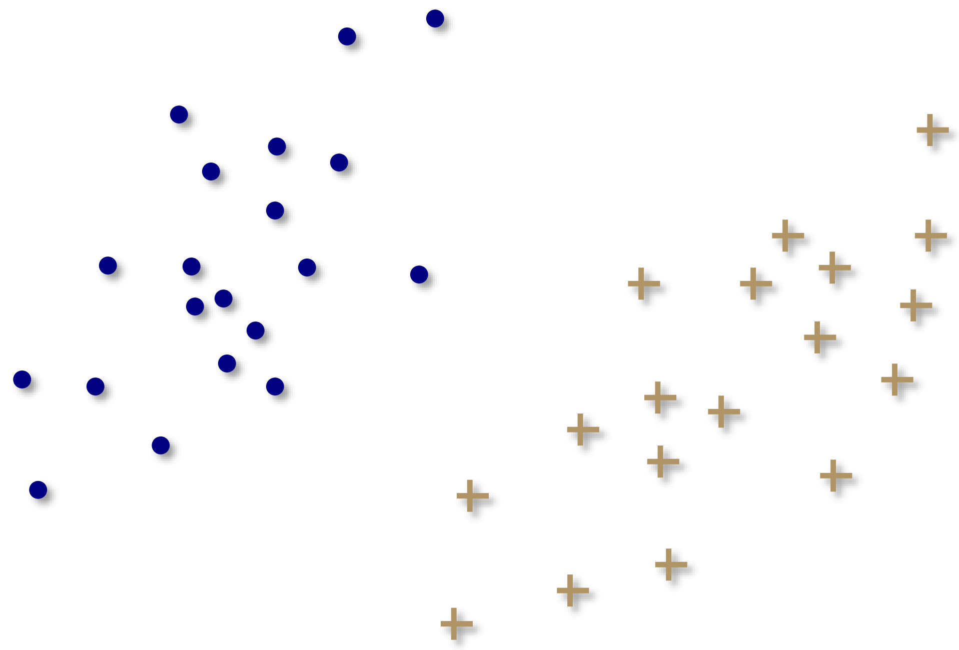
picture assumes
$\pi_0 = \pi_1$

Linear classifiers

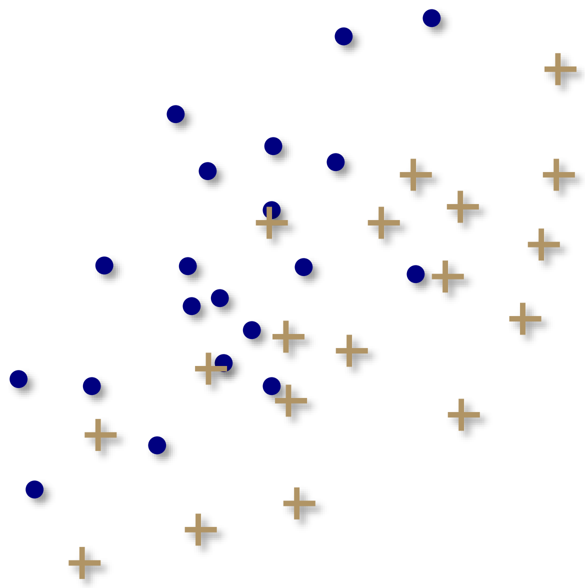
In general, why does $\mathbf{w}^T\mathbf{x} + b \underset{1}{\overset{0}{\lessgtr}} 0$ describe a linear classifier?



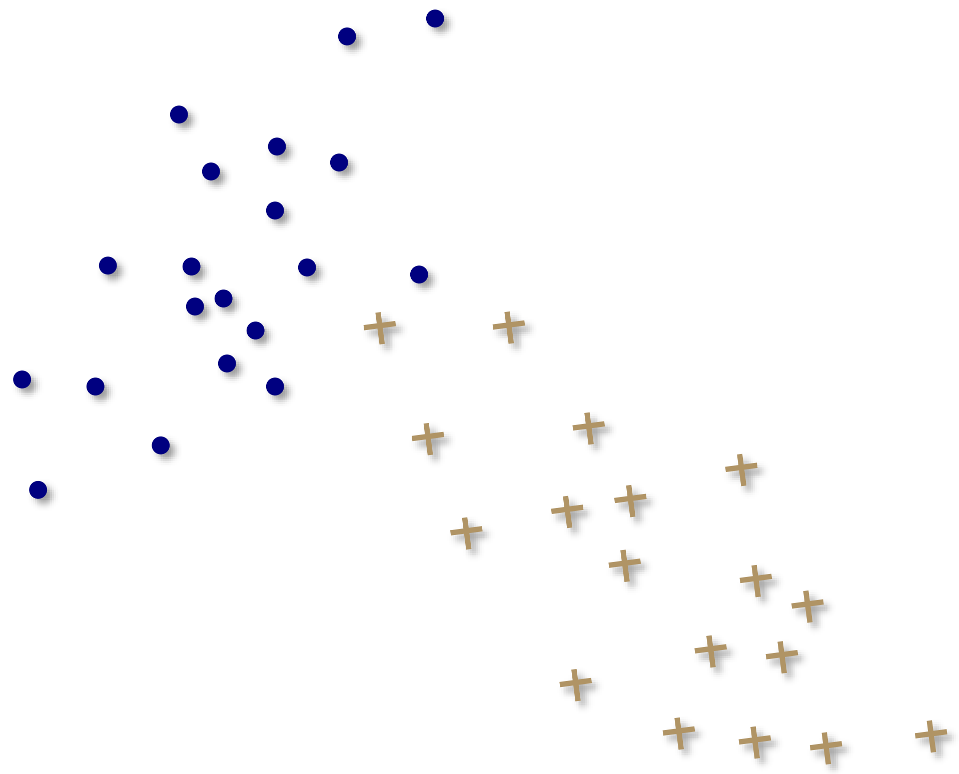
When is LDA appropriate?



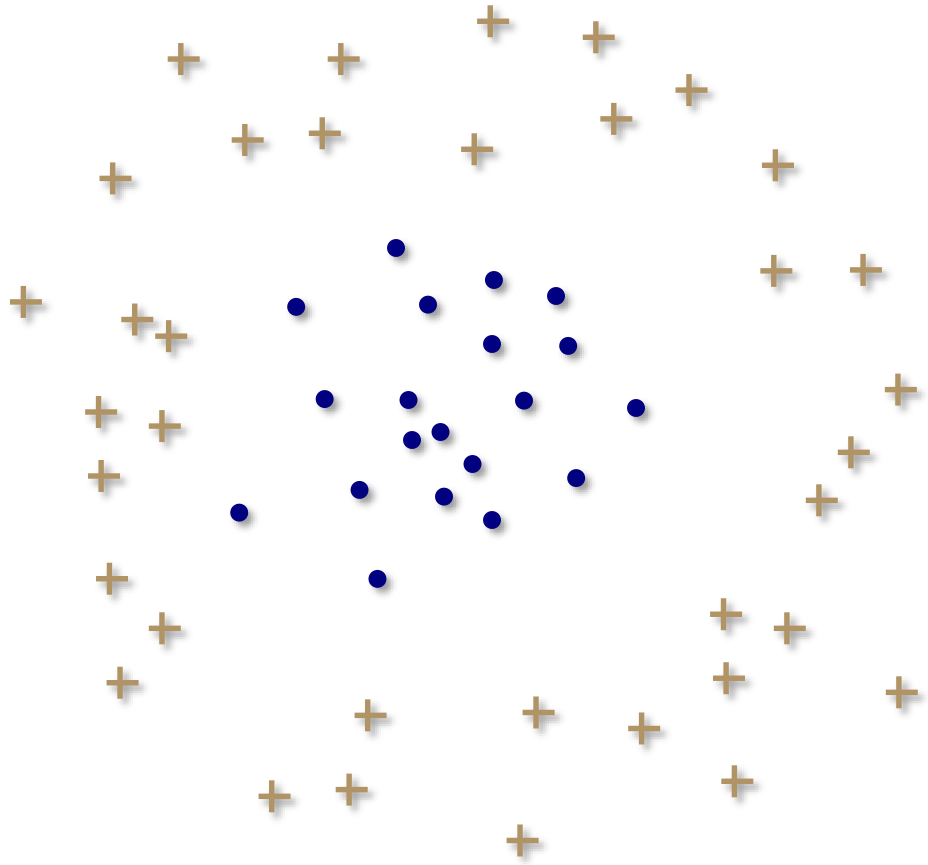
When is LDA appropriate?



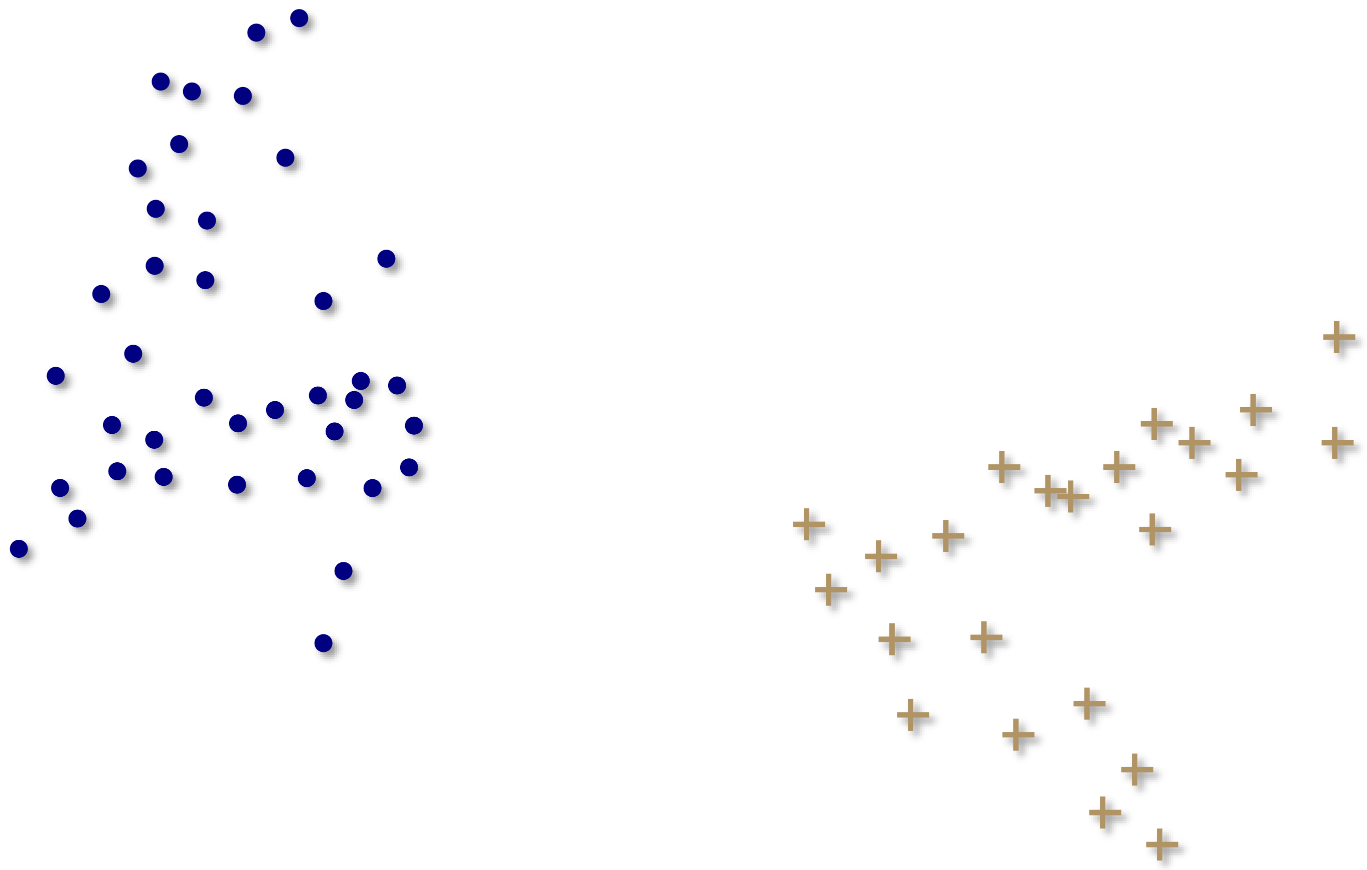
When is LDA appropriate?



When is LDA appropriate?

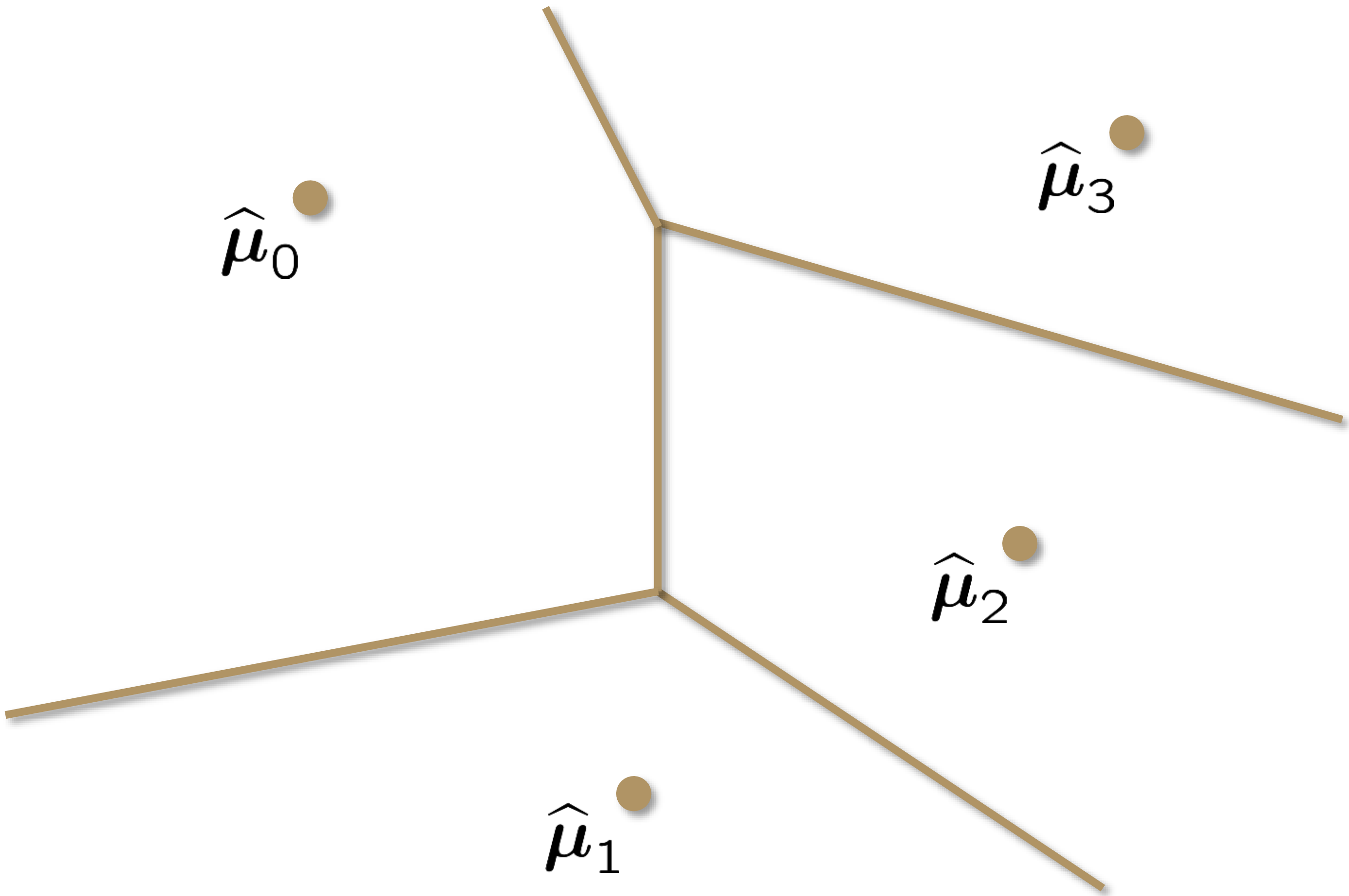


When is LDA appropriate?



More than two classes

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


$\widehat{\boldsymbol{\mu}}_0$
$\widehat{\boldsymbol{\mu}}_3$
$\widehat{\boldsymbol{\mu}}_2$
$\widehat{\boldsymbol{\mu}}_1$

Quadratic discriminant analysis (QDA)

What happens if we expand the generative model to

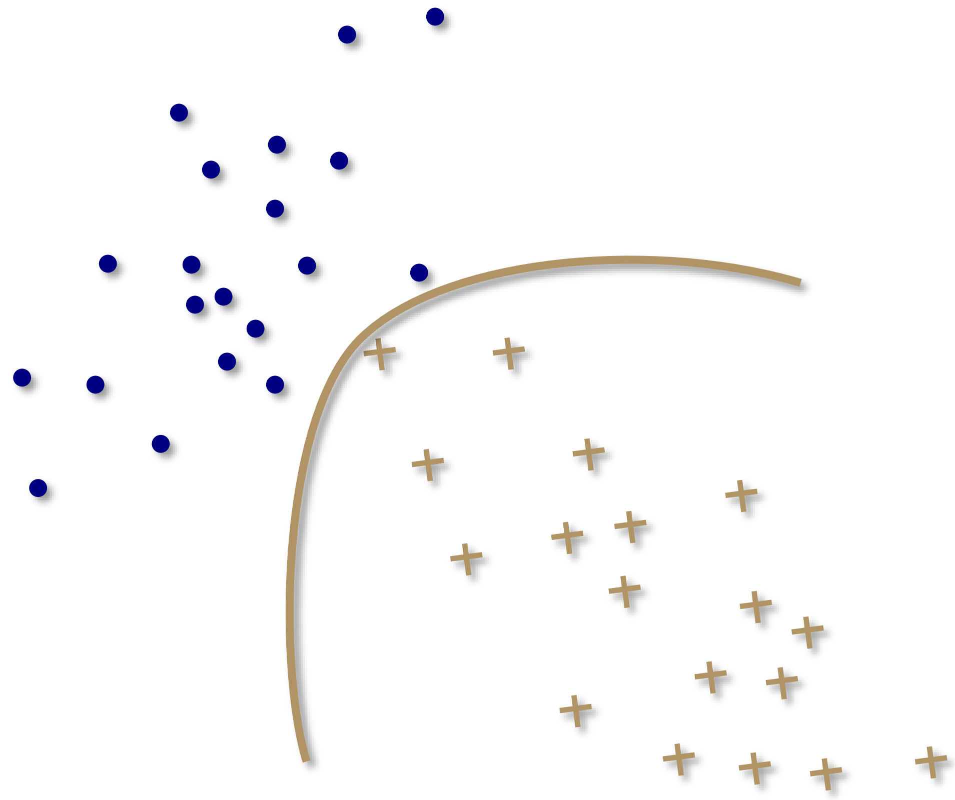
$$X|Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

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

Set $\widehat{\boldsymbol{\Sigma}}_k = \dfrac{1}{|\{i : y_i = k\}|} \displaystyle\sum_{i:y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^T$

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)

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 (more on this later)
- assume a more structured covariance matrix

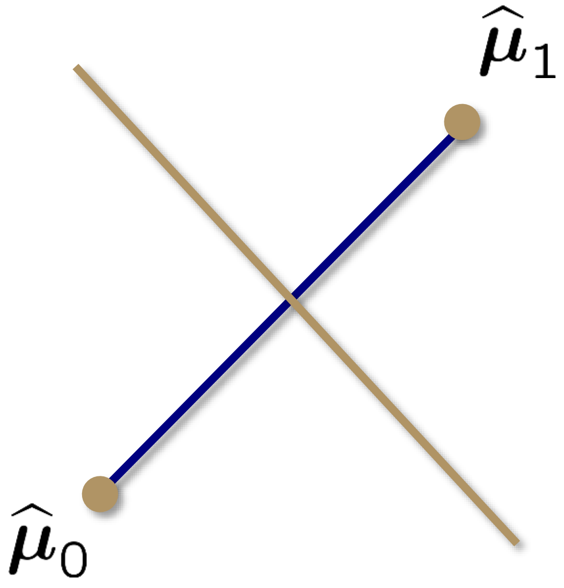
Example

Structured covariance matrix:

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

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

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


$\widehat{\boldsymbol{\mu}}_1$
$\widehat{\boldsymbol{\mu}}_0$

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 $\pi_k f_{X|Y}(\mathbf{x}|k)$ or $\eta_k(\mathbf{x})$

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

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

- 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

We will see a better approach to this problem next time