Generative Gaussian Models

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Generative Classifiers

We consider a (closed set) classification problem

We have a pattern $\boldsymbol{x}_t$ that we want to classify as belonging to one of k classes

Probabilistic model: we assume that $\boldsymbol{x}_t$ is a realization of R.V. $\boldsymbol{X}_t$

We also assume that its (unknown) class label can be described by R.V. $C_t \in \{1 \dots k\}$

$1 \dots k$ are the class labels[1]

[1] The actual values used to represent the classes are irrelevant, without loss of generality we assume classes are labeled using progressive integers. For binary problems, we will, in some cases, encode classes with $\{1, 0\}$.

Generative Classifiers

Optimal Bayes decision[2]: assign the class with highest *posterior* probability $c_t^* = \arg\max_c P(C_t = c | \boldsymbol{X}_t = \boldsymbol{x}_t)$

For example, we can consider an object classification task

$\boldsymbol{x}_t$ is the representation of an image

Labels represent what object is depicted (e.g. cat $= 1$, dog $= 2$, rabbit $= 3$, ...)

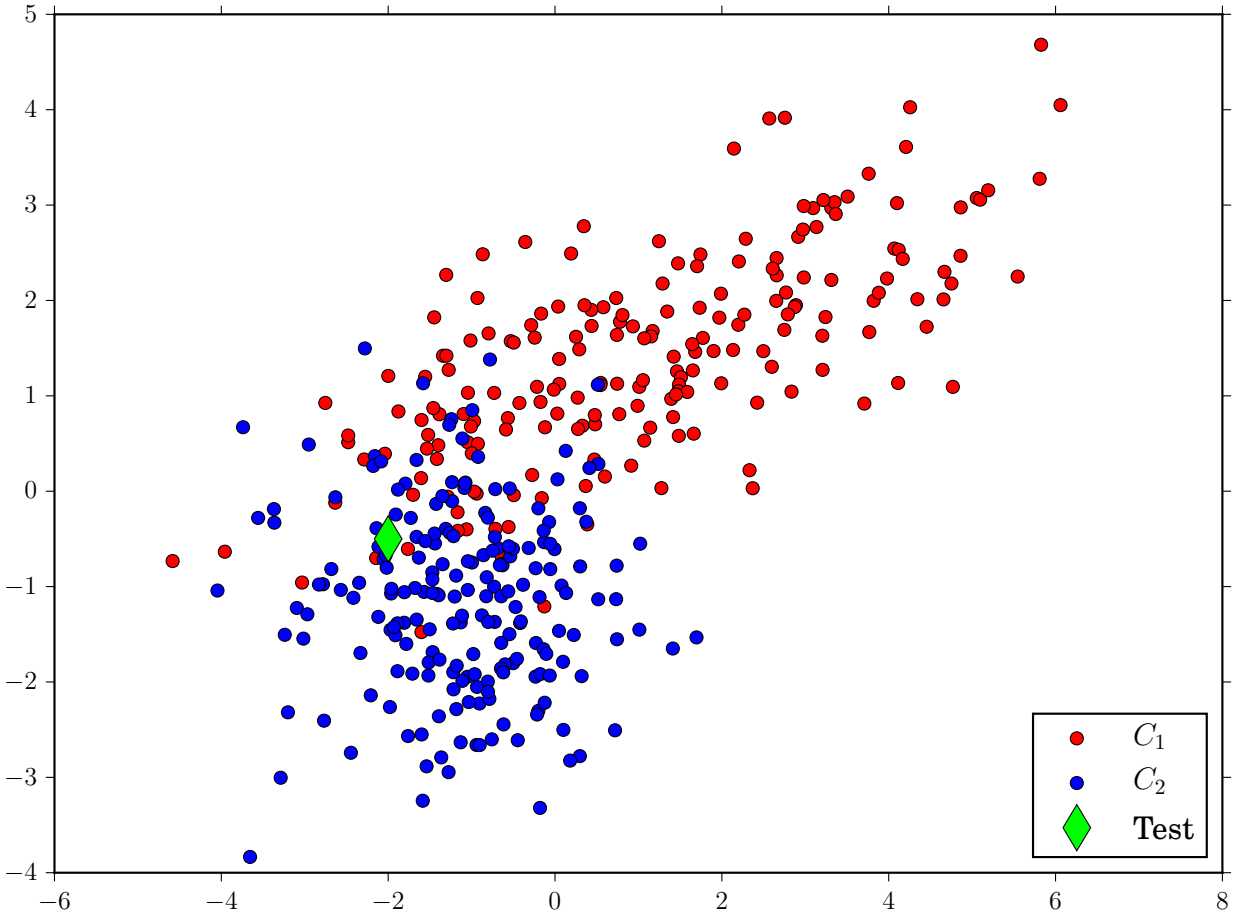
We want to find which label c_t is more likely for $\boldsymbol{x}_t$

For all labels $c \in \{1 \dots K\}$, we compute $P(C_t = c | \boldsymbol{X}_t = \boldsymbol{x}_t)$, i.e. the probability that the class C_t for the test sample t is c, conditioned on the observed value $\boldsymbol{X}_t = \boldsymbol{x}_t$

[2] Assuming uniform cost of errors

Generative Classifiers

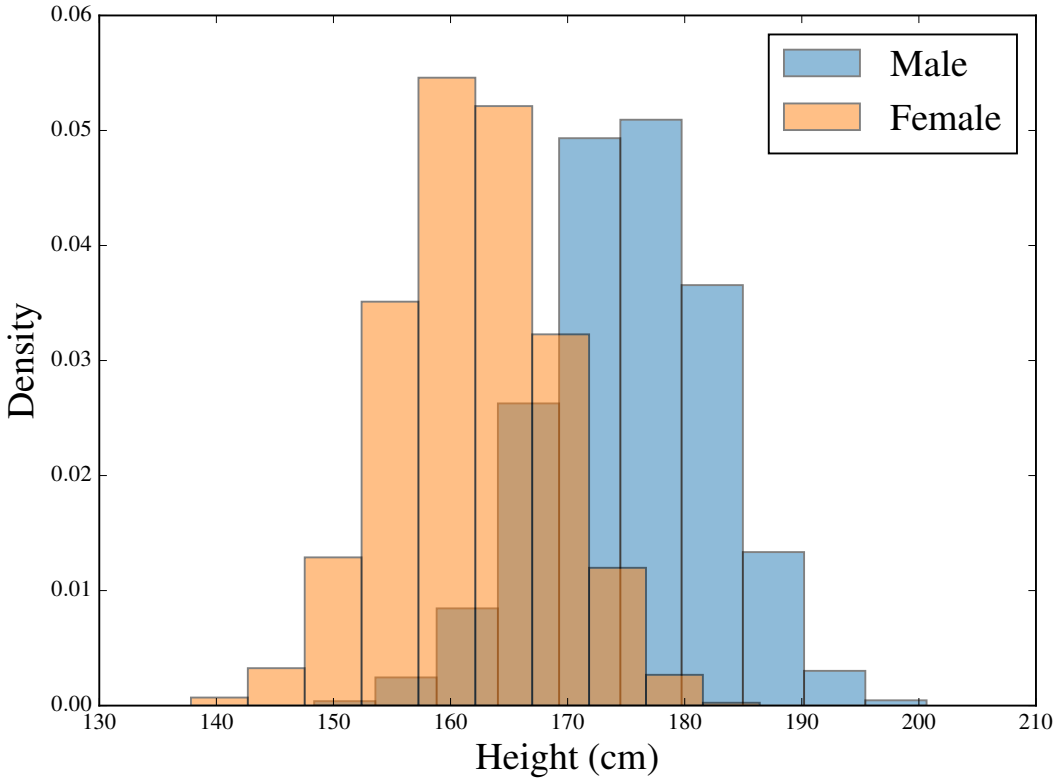
A binary example



What class is the green sample from?

Generative Classifiers

A univariate, binary example: forensics — infer the gender from the height



What's the gender of a 174 cm tall suspect?

Generative Classifiers

Simple model: assume that the samples are independent and distributed according to $\boldsymbol{X}_t, C_t \sim \boldsymbol{X}, C$, for any test sample t

Let the joint density of $\boldsymbol{X}, C$ be $f_{\boldsymbol{X},C}$

We can compute the joint likelihood for the hypothesized class c for the observed test sample $\boldsymbol{x}_t$:

$$f_{\boldsymbol{X}_t,C_t}(x_t, c) = f_{\boldsymbol{X},C}(x_t, c)$$

Since we are considering a closed-set classification problem, from Bayes rule we can compute the class *posterior* probability

$$P(C_t = c|\boldsymbol{X}_t = \boldsymbol{x}_t) = \frac{f_{\boldsymbol{X},C}(x_t, c)}{\sum_{c' \in \mathcal{C}} f_{\boldsymbol{X},C}(\boldsymbol{x}_t, c')}$$

Generative Classifiers

The joint density for $(\boldsymbol{X}_t, C_t)$ can be expressed as

$$f_{\boldsymbol{X}_t, C_t}(\boldsymbol{x}_t, c) = f_{\boldsymbol{X}, C}(\boldsymbol{x}_t, c) = f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|c)P_C(c)$$

We build a (parametric) model for the class-conditional density $f_{\boldsymbol{X}|C}(\boldsymbol{x}|c) = f_{\boldsymbol{X}|C}(\boldsymbol{x}|c, \boldsymbol{\theta})$

The class-conditional density describes the distribution of the samples of each class

The model parameters affect the class conditional density

Generative Classifiers

The term $P_C(c)$ is an *application-dependent class prior*, and (typically) does not depend on the model parameters

The class prior probability represents the probability of a sample belonging to a given class, before we actually see the sample

For example, if we want to identify pictures of a cat among a set of pictures, $P(cat)$ represent the probability that taking a random picture to classify it will actually show a cat

Within the frequentist framework, we can think of the prior as the frequency of cat pictures among the data that we will have to classify

Generative Classifiers

The application prior represents task-specific knowledge, which may not be directly known to the classifier

We will consider it as a task specification (in some cases we may have to estimate this prior as well — in this case we can resort to the frequentist interpretation and employ as application prior the frequency of the samples of each class *in the application scenario*)

Class prior probability

Pay attention that the application prior *is not* necessarily the frequency of the classes in the *training set* (or in validation / evaluation sets, as we will see shortly)

The frequency of samples of each class in a dataset is the empirical prior *for that dataset*

Why do we need to make the distinction?

- The training set should mimic as close as possible the evaluation $\longrightarrow$ shouldn't it mimic the application prior as well?

In practical cases it's simply *not viable* that the training set empirical prior reflects the application prior

Class prior probability

In practical cases it's simply *not viable* that the training set empirical prior reflects the application prior

- We may not know the application prior in advance — for example we may want to build a classifier that can adapt to different scenarios, without the need to re-train the model each time
- The application prior may not balanced — e.g., a class has very low prior probability
  - 2-class example: application prior $P(dog) = 0.001$ and $P(cat) = 0.999$ — we expect almost all pictures to represent cats
  - Training set: if we want 1000 dog pictures to build a robust dog model, we need to add 999000 cat pictures
  - We waste time collecting probably useless cat samples
  - We waste time processing probably useless cat samples

Gaussian Classifier

We factorized the joint density in class-conditional and prior distribution

Our goal now consists in modeling class-conditional densities $f_{X|C}(\boldsymbol{x}_t|c)$

We consider problems where the observations are continuous $\boldsymbol{x} \in \mathbb{R}$

How can we model $f_{X|C}(\boldsymbol{x}_t|c)$?

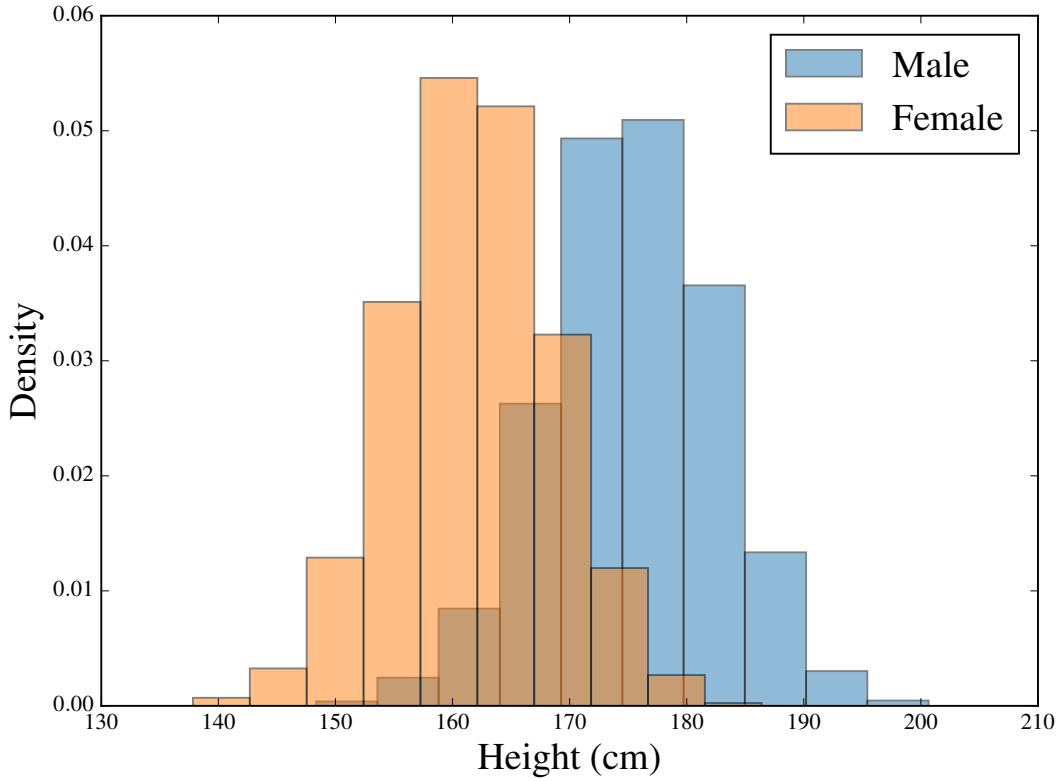
Of course, the answer depends on the data

In the following, we assume that the data of each class can be effectively modeled by a (Multivariate) Gaussian Distribution

We will need to verify the performance of the model

Generative Classifiers

We start with a univariate example — gender inference

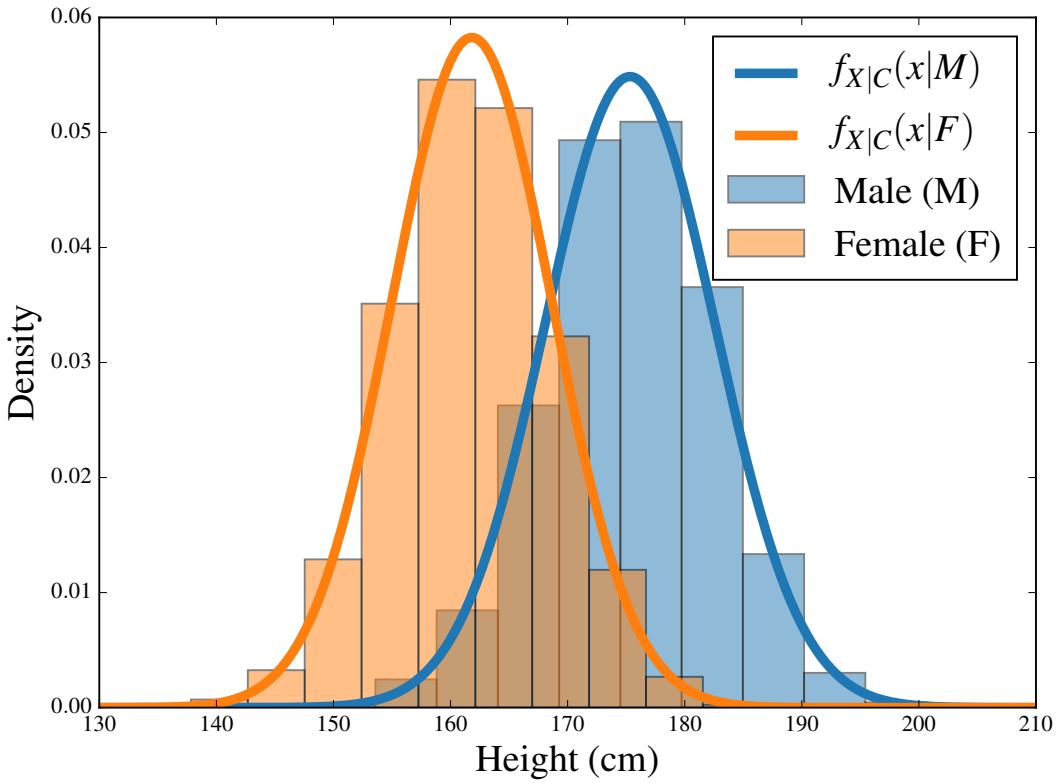


We assume we have a large set of height measurements for the population under consideration

Generative Classifiers

Intuitively, we can fit a Gaussian density over the samples of each class

We can use ML estimates to fit a Male ($C = M$) and a Female ($C = F$) Gaussian



Generative Classifiers

The ML parameters are the mean and variance of the samples of the Male and Female classes, respectively[3]

$$\mu_M = \frac{1}{N_M} \sum_{i|C_i=M} x_i \approx 175.33 \text{ cm} , \quad \sigma_M^2 = \frac{1}{N_M} \sum_{i|C_i=M} (x_i - \mu_M)^2 \approx 52.89 \text{ cm}^2$$

$$\mu_F = \frac{1}{N_F} \sum_{i|C_i=F} x_i \approx 161.82 \text{ cm} , \quad \sigma_F^2 = \frac{1}{N_F} \sum_{i|C_i=F} (x_i - \mu_F)^2 \approx 46.89 \text{ cm}^2$$

N_M and N_F are the number of male and female samples, and the sums extend over the male (first row) or female (second row) samples of the dataset

We are now able to compute the likelihood for the two classes for the 174 cm tall suspect

$$f_{X|C}(174|M) = \mathcal{N}(174|\mu_M, \sigma_M^2) \approx 0,05395$$
$$f_{X|C}(174|F) = \mathcal{N}(174|\mu_F, \sigma_F^2) \approx 0.01198$$

[3] In the following slides we will drop the unit of measurement

Generative Classifiers

It's approximately 4.5 times more likely to observe a height of 174 cm in the male population

However, this is not sufficient to answer whether the sample is from a male or a female

We need to compute the class *posterior* probability, which depends also on the class *prior* probability

$$P(C = M|X = 174) = \frac{f_{X|C}(174|M)P(C = M)}{f_X(174)}$$

$$P(C = F|X = 174) = \frac{f_{X|C}(174|F)P(C = F)}{f_X(174)}$$

If we want to just compare the two probabilities, we don't need to compute the normalization term $f_X(174)$

Gaussian Classifier

The *class posterior ratio* for the two hypotheses is

$$\frac{P(C=M|X=174)}{P(C=F|X=174)} = \frac{f_{X|C}(174|M)}{f_{X|C}(174|F)} \frac{P(C=M)}{P(C=F)}$$

The prior probabilities represent the probability that, a priori, we expect to observe a male or female sample

In this example, we may not have any knowledge of the suspect gender, so we may assume $P(C=M) = P(C=F) = \frac{1}{2}$

In this case

$$\frac{P(C=M|X=174)}{P(C=F|X=174)} = \frac{f_{X|C}(174|M)}{f_{X|C}(174|F)} \approx 4.5$$

i.e., the probability that the sample is from a male is 4.5 times higher than the probability that it is from a female

Gaussian Classifier

In other cases, we may have other information sources that lead us believe that the suspect is more likely to be from a specific gender.

As an example, we may believe for other reasons that the probability that the suspect is female is 90%: $P(C = F) = 0.9$ and $P(C = M) = 0.1$

In this case, the posterior ratio becomes

$$\frac{P(C = M|X = 174)}{P(C = F|X = 174)} = \frac{f_{X|C}(174|M)}{f_{X|C}(174|F)} \frac{P(C = M)}{P(C = F)} \approx \frac{4.5}{9} = \frac{1}{2}$$

i.e., the probability that the suspect is female is still twice the probability that the suspect is make, *even though* the evidence suggests otherwise

In this case, the evidence is not strong enough to change our prior belief.

Gaussian Classifier

We will now formalize the method we just employed in the example

We assume that our data, given the class, can be described by a Gaussian distribution

$$(\boldsymbol{X}_t|C_t = c) \sim (\boldsymbol{X}|C = c) \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

We have one mean and one covariance matrix per class

If we knew $\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c$, then we could compute $f_{\boldsymbol{X}_t|C_t=c}$ as

$$f_{\boldsymbol{X}_t|C_t}(\boldsymbol{x}_t|c) = f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|c) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

We do not, however, know the values for the model parameters $\boldsymbol{\theta} = [(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \ldots (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$

Gaussian Classifier

On the other hand, we have at our disposal a labeled *training* dataset

$$\mathcal{D} = \{(\boldsymbol{x}_1, c_1) \dots (\boldsymbol{x}_n, c_n)\}$$

$\mathcal{X} = \{\boldsymbol{x}_1 \dots \boldsymbol{x}_n\}$ are the observed samples

$\mathcal{C} = \{c_1 \dots c_n\}$ are the corresponding class labels $c_i \in \{1 \dots k\}$

We want to learn the model parameters from the data

Gaussian Classifier

We assume that, given the model parameters $\boldsymbol{\theta}$, observations are *independent and identically distributed* (i.i.d.)

$$[(\boldsymbol{X}_i, C_i) \perp\!\!\!\perp (\boldsymbol{X}_j, C_j)] \,|\boldsymbol{\theta}$$

and

$$\forall i, \quad (X_i, C_i)|\boldsymbol{\theta} \sim (X, C)|\boldsymbol{\theta}$$

i.e., we assume that both the *training* set and *evaluation* samples are independent (given the model parameters) and they are distributed in the same way

Gaussian Classifier

Since we assume Gaussian distribution for $\boldsymbol{X}|C$, we have

$$(\boldsymbol{X}_i|C_i = c, \boldsymbol{\theta}) \sim (\boldsymbol{X}_t|C_t = c, \boldsymbol{\theta}) \sim (\boldsymbol{X}|C = c, \boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

i.e., the class-conditional distribution for all observations is a Gaussian with class-dependent mean $\boldsymbol{\mu}_c$ and class-dependent covariance matrix $\boldsymbol{\Sigma}_c$

Again, the model parameters are $\boldsymbol{\theta} = [(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \dots (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$

We follow a frequentist approach, and we thus want to compute an *estimator* (or point estimate) $\boldsymbol{\theta}^*$ of the model parameters

We will then use the estimated parameters to compute

$$f_{\boldsymbol{X}_t|C_t}(\boldsymbol{x}_t|c) \approx \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_c^*, \boldsymbol{\Sigma}_c^*)$$

Gaussian Classifier

We have seen that a possible way to estimate the model parameters is to maximize the data (log-)likelihood

The *data likelihood* for $\boldsymbol{\theta}$ consists of the joint density of the observed training set variables, given the parameter vector $\boldsymbol{\theta}$:

$$\mathcal{L}(\boldsymbol{\theta}) = f_{X_1 \ldots X_n, C_1 \ldots C_n | \boldsymbol{\theta}}(\boldsymbol{x}_1 \ldots \boldsymbol{x}_n, c_1 \ldots c_n | \boldsymbol{\theta})$$

Since we assume i.i.d. observations, we can factorize the likelihood as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= f_{X_1 \ldots X_n, C_1 \ldots C_n | \boldsymbol{\theta}}(\boldsymbol{x}_1 \ldots \boldsymbol{x}_n, c_1 \ldots c_n | \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} f_{X,C|\boldsymbol{\theta}}(\boldsymbol{x}_i, c_i | \boldsymbol{\theta}) \end{aligned}$$

Gaussian Classifier

We now introduce our *model* for the *joint density*.

Our generative model assumes that the joint density consists of the product of a parametetric class-conditional density

$$f_{X|C,\boldsymbol{\theta}}(\boldsymbol{x}|c,\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$$

and an (application) dependent prior probability $P_C(c)$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{n} f_{X,C|\boldsymbol{\theta}}(\boldsymbol{x}_i, c_i|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} f_{X|C,\boldsymbol{\theta}}(\boldsymbol{x}_i|c_i,\boldsymbol{\theta}) P(c_i) \\ &= \prod_{i=1}^{n} \mathcal{N}\left(\boldsymbol{x}_i|\boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i}\right) P(c_i)\end{aligned}$$

Gaussian Classifier

We again consider the log-likelihood

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \log \mathcal{L}(\boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} \log \mathcal{N}\left(\boldsymbol{x}_i | \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i}\right) + \sum_{i} \log P(c_i) \\ &= \sum_{c=1}^{k} \sum_{i|c_i=c} \log \mathcal{N}\left(\boldsymbol{x}_i | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right) + \xi\end{aligned}$$

where ξ collects the *class prior probability* terms that *do not depend* on $\boldsymbol{\theta}$, and thus are *irrelevant for the maximization* with respect to $\boldsymbol{\theta}$

The log-likelihood corresponds to a sum over all classes of the conditional log-likelihood of the samples belonging to each class

$$\ell(\boldsymbol{\theta}) = \sum_{c=1}^{k} \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) + \xi \ , \quad \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \sum_{i|c_i=c} \log \mathcal{N}\left(\boldsymbol{x}_i | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$$

Gaussian Classifier

We observe that we can maximize ℓ by separately maximizing the terms $\ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$

$\ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ is simply the log-likelihood of a Gaussian model for the data of class c

We are independently estimating the Gaussian densities that best describe the data of each class c

For univariate R.V.s, we have already shown that the ML solution corresponds to the class mean and variance

We now consider the general case for multivariate samples

Gaussian Classifier

The log-density for a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is

$$\log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{D}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

or, in terms of precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$

$$\log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{D}{2}\log 2\pi + \frac{1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Lambda}(\boldsymbol{x}-\boldsymbol{\mu})$$

Gaussian Classifier

The log-likelihood $\ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ can thus be expressed as

$$\ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = k + \frac{N_c}{2} \log |\boldsymbol{\Lambda}_c| - \frac{1}{2} \sum_{i|c_i=c} (\boldsymbol{x}_i - \boldsymbol{\mu}_c)^T \boldsymbol{\Lambda}_c (\boldsymbol{x}_i - \boldsymbol{\mu}_c)$$

We can rewrite the log-likelihood in different ways:

$$\ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = k + \frac{N_c}{2} \log |\boldsymbol{\Lambda}_c| - \frac{1}{2} \operatorname{Tr} \left(\boldsymbol{\Lambda}_c \sum_{i|c_i=c} (\boldsymbol{x}_i - \boldsymbol{\mu}_c)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)^T \right) \tag{1}$$

$$= k + \frac{N_c}{2} \log |\boldsymbol{\Lambda}_c| - \frac{1}{2} \sum_{i|c_i=c} \boldsymbol{x}_i^T \boldsymbol{\Lambda}_c \boldsymbol{x}_i + \boldsymbol{\mu}_c^T \boldsymbol{\Lambda}_c \sum_{i|c_i=c} \boldsymbol{x}_i - \frac{N_c}{2} \boldsymbol{\mu}_c^T \boldsymbol{\Lambda}_c \boldsymbol{\mu}_c$$

$$= k + \frac{N_c}{2} \log |\boldsymbol{\Lambda}_c| - \frac{1}{2} \operatorname{Tr} \left(\boldsymbol{\Lambda}_c \sum_{i|c_i=c} \boldsymbol{x}_i \boldsymbol{x}_i^T \right) + \boldsymbol{\mu}_c^T \boldsymbol{\Lambda}_c \sum_{i|c_i=c} \boldsymbol{x}_i - \frac{N_c}{2} \boldsymbol{\mu}_c^T \boldsymbol{\Lambda}_c \boldsymbol{\mu}_c \tag{2}$$

Gaussian Classifier

From (2) we observe that the log-likelihood depends on the data only through the *statistics*

$$Z_c = N_c$$
$$\boldsymbol{F}_c = \sum_{i|c_i=c} \boldsymbol{x}_i$$
$$\boldsymbol{S}_c = \sum_{i|c_i=c} \boldsymbol{x}_i \boldsymbol{x}_i^T$$

These are also called *sufficient statistics*: they collect all the information contained in the dataset that is relevant for the estimation of $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$

Gaussian Classifier

We can find the maximum of ℓ_c by taking the derivatives of ℓ_c and setting them equal to 0:

$$\begin{cases} \nabla_{\boldsymbol{\Lambda}_c} \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Lambda}_c) = \mathbf{0} \\ \nabla_{\boldsymbol{\mu}_c} \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Lambda}_c) = \mathbf{0} \end{cases}$$

From (2), the derivative with respect to $\boldsymbol{\mu}_c$ is

$$\nabla_{\boldsymbol{\mu}_c} \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Lambda}_c) = \boldsymbol{\Lambda}_c \sum_{i|c_i=c} \boldsymbol{x}_i - N_c \boldsymbol{\Lambda}_c \boldsymbol{\mu}_c$$

Solving for $\nabla_{\boldsymbol{\mu}_c} \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Lambda}_c) = \mathbf{0}$ gives

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i|c_i=c} \boldsymbol{x}_i$$

i.e., the mean of samples belonging to class c

Gaussian Classifier

From (1), we can compute the derivative[4] w.r.t. $\boldsymbol{\Lambda}_c$:

$$\nabla_{\boldsymbol{\Lambda}_c} \ell_c(\boldsymbol{\mu}_c, \boldsymbol{\Lambda}_c) = \frac{N_c}{2}\boldsymbol{\Lambda}^{-T} - \frac{1}{2}\sum_{i|c_i=c}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$$

Assuming that $\sum_{i|c_i=c}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$ is positive-definite (we can check that it's also symmetric), solving for $\boldsymbol{\Lambda}_c^{-1}$ gives

$$\boldsymbol{\Sigma}_c = \boldsymbol{\Lambda}_c^{-1} = \frac{1}{N_c}\sum_{i|c_i=c}(\boldsymbol{x}_i - \boldsymbol{\mu}_c)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)^T$$

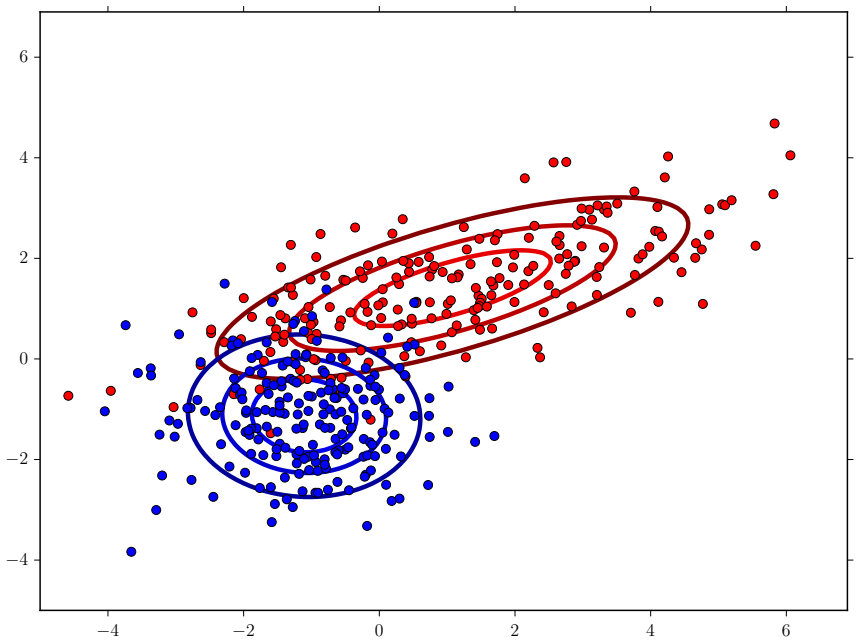
i.e., the covariance matrix of samples belonging to class c, computed using the data mean $\boldsymbol{\mu}_c$

[4] Since $\boldsymbol{\Lambda}_c$ is a precision matrix, it should be symmetric positive definite. We therefore should maximize ℓ_c under such constraint. In practice, we solve the problem for unconstrained $\boldsymbol{\Lambda}_c$, and then we check whether the unconstrained solution satisfies the constraints. Since it does, it's also the optimal solution for the constrained problem

Gaussian Classifier

Summarizing, the ML solution is given by

$$\boldsymbol{\mu}_c^* = \frac{1}{N_c} \sum_{i|c_i=c} \boldsymbol{x}_i \ , \quad \boldsymbol{\Sigma}_c^* = \frac{1}{N_c} \sum_{i|c_i=c} (\boldsymbol{x}_i - \boldsymbol{\mu}_c^*)(\boldsymbol{x}_i - \boldsymbol{\mu}_c^*)^T$$



We can then compute the likelihood of class c for test point $\boldsymbol{x}_t$ as

$$f_{\boldsymbol{X}_t|C_t}(\boldsymbol{x}_t|c) = f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|c) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_c^*, \boldsymbol{\Sigma}_c^*)$$

Gaussian Classifier

Let's now consider a binary task, with two classes[5] $C \in \{h_1, h_0\}$

We assign the label to a test sample $\boldsymbol{x}_t$ according to the highest posterior probability, comparing $P(C = h_1|\boldsymbol{x}_t)$ to $P(C = h_0|\boldsymbol{x}_t)$

We can express the comparison in terms of class posterior ratio

$$r(\boldsymbol{x}_t) = \frac{P(C = h_1|\boldsymbol{x}_t)}{P(C = h_0|\boldsymbol{x}_t)}$$

or, alternatively, in terms of its logarithm

$$\log r(\boldsymbol{x}_t) = \log \frac{P(C = h_1|\boldsymbol{x}_t)}{P(C = h_0|\boldsymbol{x}_t)}$$

[5] For binary problems it's common to label classes as 1 (target hypothesis, true hypothesis, ...) and 0 (non-target hypothesis, false hypothesis, null hypothesis, ...). As we have already said, the chosen labeling scheme is irrelevant for our discussion — we denote the class labels as h_1 and h_0

Gaussian Classifier

If the log-ratio is greater than 0, then the point will be assigned to class h_1, otherwise it will be assigned to class h_0

The class posterior ratio can be rewritten to make explicit its dependency on the likelihoods $f_{X|C}(\boldsymbol{x}_t|c)$ and prior class probabilities:

$$\begin{aligned}
\log r(\boldsymbol{x}_t) &= \log \frac{P(C=h_1|\boldsymbol{x}_t)}{P(C=h_0|\boldsymbol{x}_t)} \\
&= \log \frac{f_{X,C}(\boldsymbol{x}_t, h_1)}{f_X(\boldsymbol{x}_t)} \cdot \frac{f_X(\boldsymbol{x}_t)}{f_{X,C}(\boldsymbol{x}_t, h_0)} \\
&= \log \frac{f_{X|C}(\boldsymbol{x}_t|h_1)P(C=h_1)}{f_{X|C}(\boldsymbol{x}_t|h_0)P(C=h_0)} \\
&= \log \frac{f_{X|C}(\boldsymbol{x}_t|h_1)}{f_{X|C}(\boldsymbol{x}_t|h_0)} + \log \frac{P(C=h_1)}{P(C=h_0)}
\end{aligned}$$

Gaussian Classifier

The first element of the sum is the *log-likelihood ratio*

$$llr(\boldsymbol{x}_t) = \log \frac{f_{X|C}(\boldsymbol{x}_t|h_1)}{f_{X|C}(\boldsymbol{x}_t|h_0)}$$

It represents the ratio between the likelihood of observing the sample given that it belongs to h_1 or to h_0

The second term represents the prior (log)-odds. For a binary problem, we have

$$P(C = h_1) = \pi \ , \quad P(C = h_0) = 1 - P(C = h_1) = 1 - \pi$$

thus

$$\log r(\boldsymbol{x}_t) = \log \frac{f_{X|C}(\boldsymbol{x}_t|h_1)}{f_{X|C}(\boldsymbol{x}_t|h_0)} + \log \frac{\pi}{1-\pi}$$

Gaussian Classifier

As we have mentioned, π reflects the prior probability for class h_1 given a specific application

The first term, the log-likelihood ratio (LLR), is what our system should focus on providing — we want our system to be application-independent as much as possible

At deployment time the LLR should then be combined with task-specific prior probabilities to compute posterior class log-probability ratios

Gaussian Classifier

The optimal decision is based on the comparison

$$\log r(\boldsymbol{x}_t) \gtrless 0$$

which means that we compare

$$\log r(\boldsymbol{x}_t) = \log \frac{f_{X|C}(\boldsymbol{x}_t|h_1)}{f_{X|C}(\boldsymbol{x}_t|h_0)} + \log \frac{\pi}{1-\pi} \gtrless 0$$

i.e., we assign classes based on

$$llr(\boldsymbol{x}_t) = \log \frac{f_{X|C}(\boldsymbol{x}_t|h_1)}{f_{X|C}(\boldsymbol{x}_t|h_0)} \gtrless -\log \frac{\pi}{1-\pi}$$

Gaussian Classifier

The log-likelihood ratio acts as a *score*, with a probabilistic interpretation

Greater scores values imply our system favors class h_1, lower values mean it favors class h_0

The decision requires comparing the llr to a threshold t that depends on the application, through the class prior probability π

Later we shall see that we can (and should) also account for different costs for different kind of errors — we will show that this corresponds to using different *effective* priors

Gaussian Classifier

Let's see what kind of decision surfaces correspond to the llr of the Gaussian classifier with parameters $[(\boldsymbol{\mu}_1, \boldsymbol{\Lambda}_1^{-1}), (\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1})]$

We can compute the log–likelihood ratio

$$llr(\boldsymbol{x}) = \log \frac{\mathcal{N}(\boldsymbol{x}|h_1)}{\mathcal{N}(\boldsymbol{x}|h_0)} = \log \frac{\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Lambda}_1^{-1})}{\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1})}$$

The decision function is *quadratic* in $\boldsymbol{x}$:

$$llr(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{b} + c$$

with

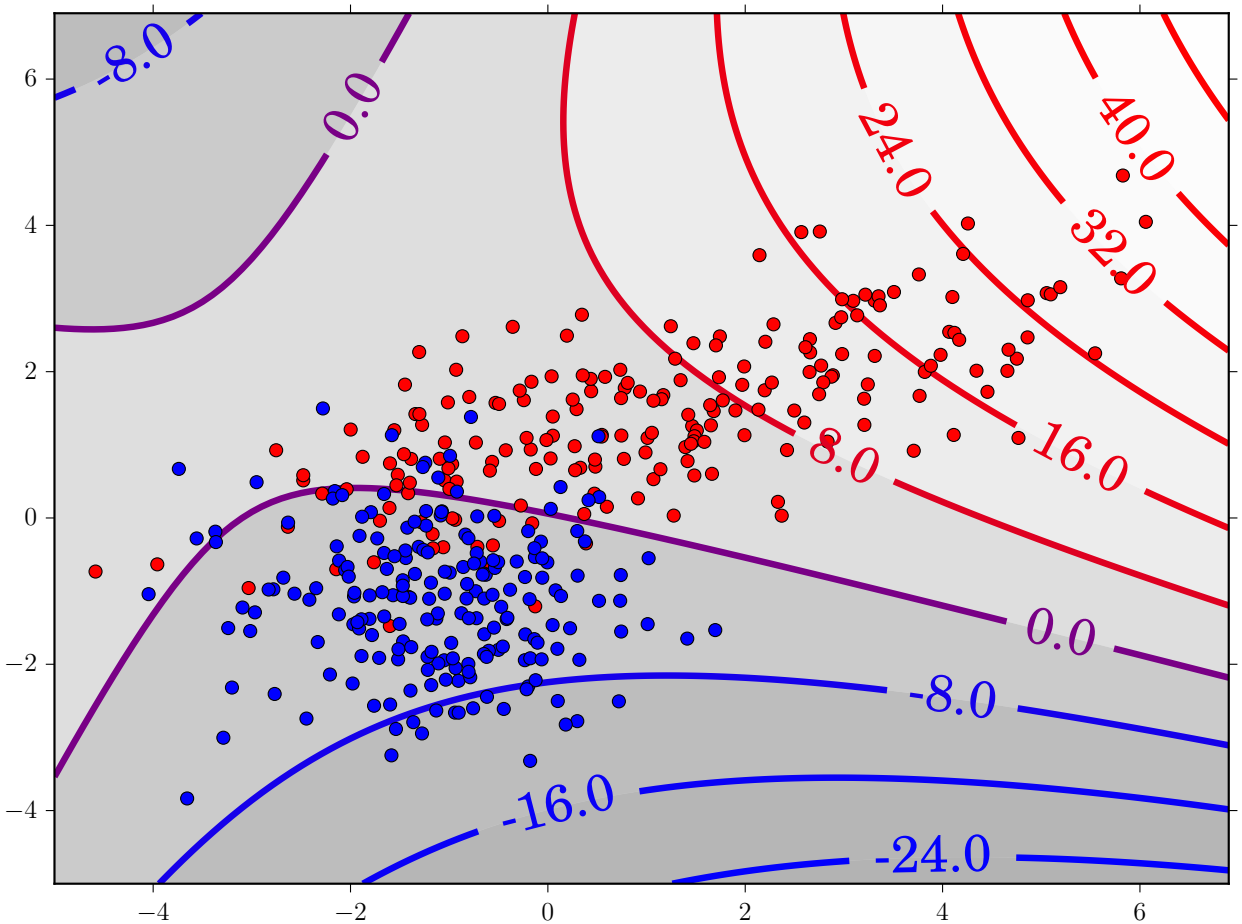
$$\boldsymbol{A} = -\frac{1}{2} (\boldsymbol{\Lambda}_1 - \boldsymbol{\Lambda}_0)$$

$$\boldsymbol{b} = (\boldsymbol{\Lambda}_1 \boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0)$$

$$c = -\frac{1}{2} \left(\boldsymbol{\mu}_1^T \boldsymbol{\Lambda}_1 \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^T \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0\right) + \frac{1}{2} \left(\log |\boldsymbol{\Lambda}_1| - \log |\boldsymbol{\Lambda}_0|\right)$$

Gaussian Classifier

Binary problem — decision boundaries



Gaussian Classifier

For multiclass problems $C \in \{h_1, h_2 \dots h_k\}$ we can compute closed-set posterior probabilities as

$$P(C = h_i|\boldsymbol{x}_t) = \frac{f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|h_i)P(h_i)}{\sum_{h' \in \{h_1, h_2 \dots h_k\}} f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|h')P(h')}$$

Optimal decisions require choosing the class with highest posterior probability $c_t^* = \arg\max_h P(C = h|\boldsymbol{x}_t)$. Posterior probabilities are proportional to

$$P(C = h_i|\boldsymbol{x}_t) \propto f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|h_i)P(h_i)$$

and the proportionality factor is the same for all classes

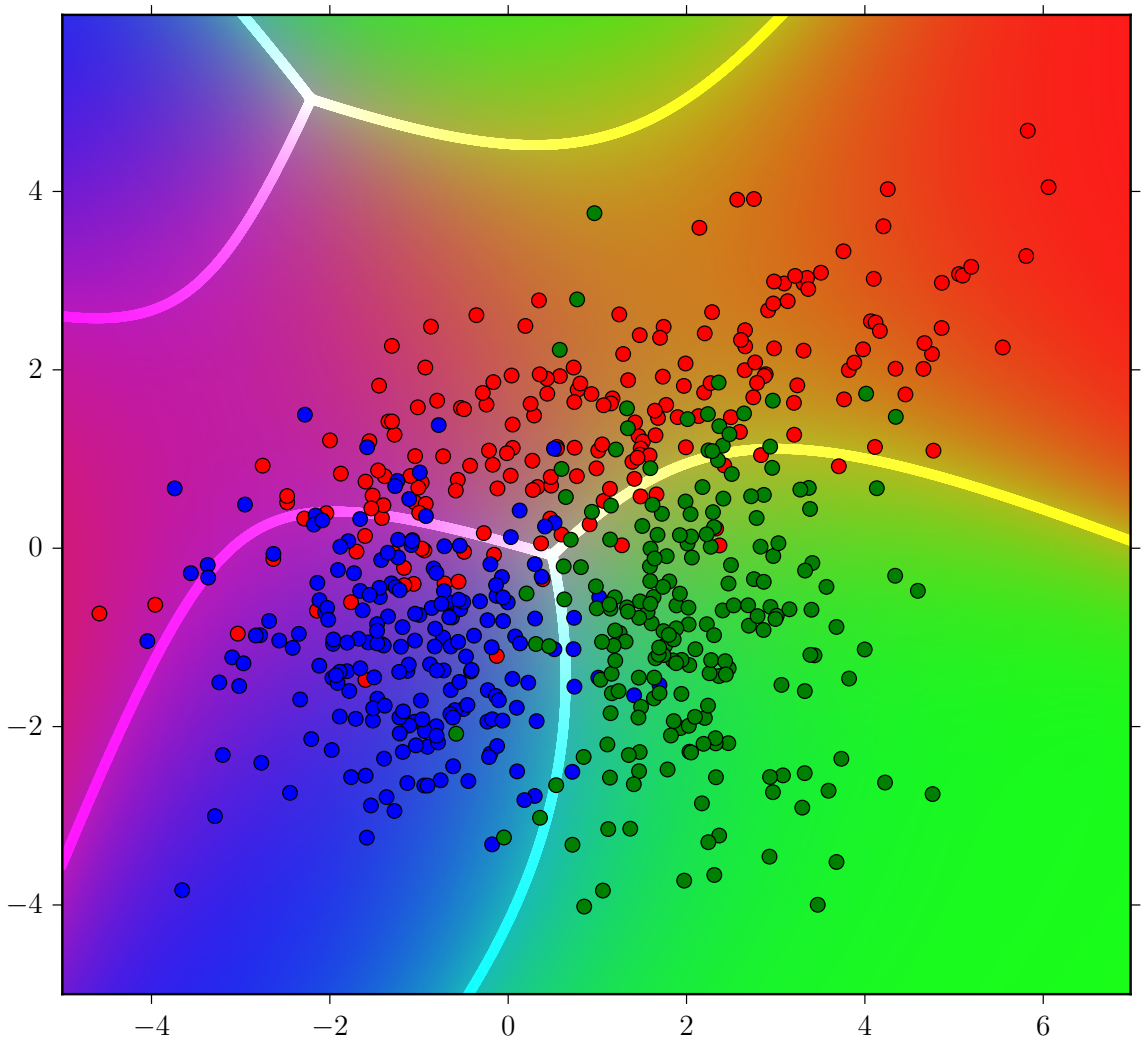
Optimal decisions can thus be computed as

$$c_t^* = \arg\max_h f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|h)P(h) = \arg\max_h \log f_{\boldsymbol{X}|C}(\boldsymbol{x}_t|h) + \log P(h)$$

Again, the first term should be the output of the classifier, whereas the second term $\log P(h)$ depends on the application

Gaussian Classifier

3 class problem — class posteriors and pair-wise boundaries



Gaussian Classifier

The Gaussian model requires computing a mean and a covariance matrix for each class

If the samples are few compared to their dimensionality, then the estimates can be inaccurate

The issue is more evident for covariance matrices, since they have $\frac{D\times(D+1)}{2}$ independent elements

The off-diagonal terms of $\mathbf{\Sigma}_c$ represent the covariances of the different components of our feature vectors

Gaussian Classifier

If we know that, for each class, the different components are approximately independent, we can simplify the estimate assuming that the density of $\boldsymbol{X}|C$ can be factorized over its components

$$f_{\boldsymbol{X}|C}(\boldsymbol{x}|c) \approx \prod_{j=1}^{D} f_{X_{[j]}|C}(x_{[j]}|c)$$

where $x_{[j]}$ is the j-th component of $\boldsymbol{x}$ (not to be confused with $\boldsymbol{x}_j$, the j-th dataset sample)

This model is called *Naive Bayes*

The assumption is not tied to any specific distribution — we can even employ a different distribution family for each component

Gaussian Classifier

The naive Bayes assumption, combined with Gaussian assuptions, models the distribution densities $f_{X_{[j]}|C}(x_{[j]}|c)$ as univariate Gaussians

$$f_{X_{[j]}|C}(x_{[j]}|c) = \mathcal{N}(x_{[j]}|\mu_{c,[j]}, \sigma^2_{c,[j]})$$

We can again compute the ML estimates. The log-likelihood factorizes over sample components:

$$\mathcal{L}(\boldsymbol{\theta}) \propto \prod_{i=1}^{n}\prod_{j=1}^{D} \mathcal{N}\left(\boldsymbol{x}_{i,[j]}|\mu_{c_i,[j]}, \sigma^2_{c_i,[j]}\right)$$

$$\ell(\boldsymbol{\theta}) = \xi + \sum_{c=1}^{k}\sum_{i|c_i=c}\sum_{j=1}^{D} \log \mathcal{N}\left(\boldsymbol{x}_{i,[j]}|\mu_{c,[j]}, \sigma^2_{c,[j]}\right)$$

$$= \xi + \sum_{j=1}^{D}\sum_{c=1}^{k}\sum_{i|c_i=c} \log \mathcal{N}\left(\boldsymbol{x}_{i,[j]}|\mu_{c,[j]}, \sigma^2_{c,[j]}\right)$$

Gaussian Classifier

We can optimize the log-likelihood independently for each component

For each component, we have the log-likelihood of a Gaussian model

The ML solution is

$$\mu^*_{c,[j]} = \frac{1}{N_c} \sum_{i|c_i=c} x_{i,[j]} \ , \quad \sigma^2_{c,[j]} = \frac{1}{N_c} \sum_{i|c_i=c} (x_{i,[j]} - \mu_{c,[j]})^2$$

Gaussian Classifier

We can observe that the density for a sample $\boldsymbol{x}$ can be expressed as

$$f_{\boldsymbol{X}|C}(\boldsymbol{x}|c) = \prod_{j=1}^{D} \mathcal{N}(x_{[j]}|\mu^*_{c,[j]}\sigma^2_{c,[j]}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

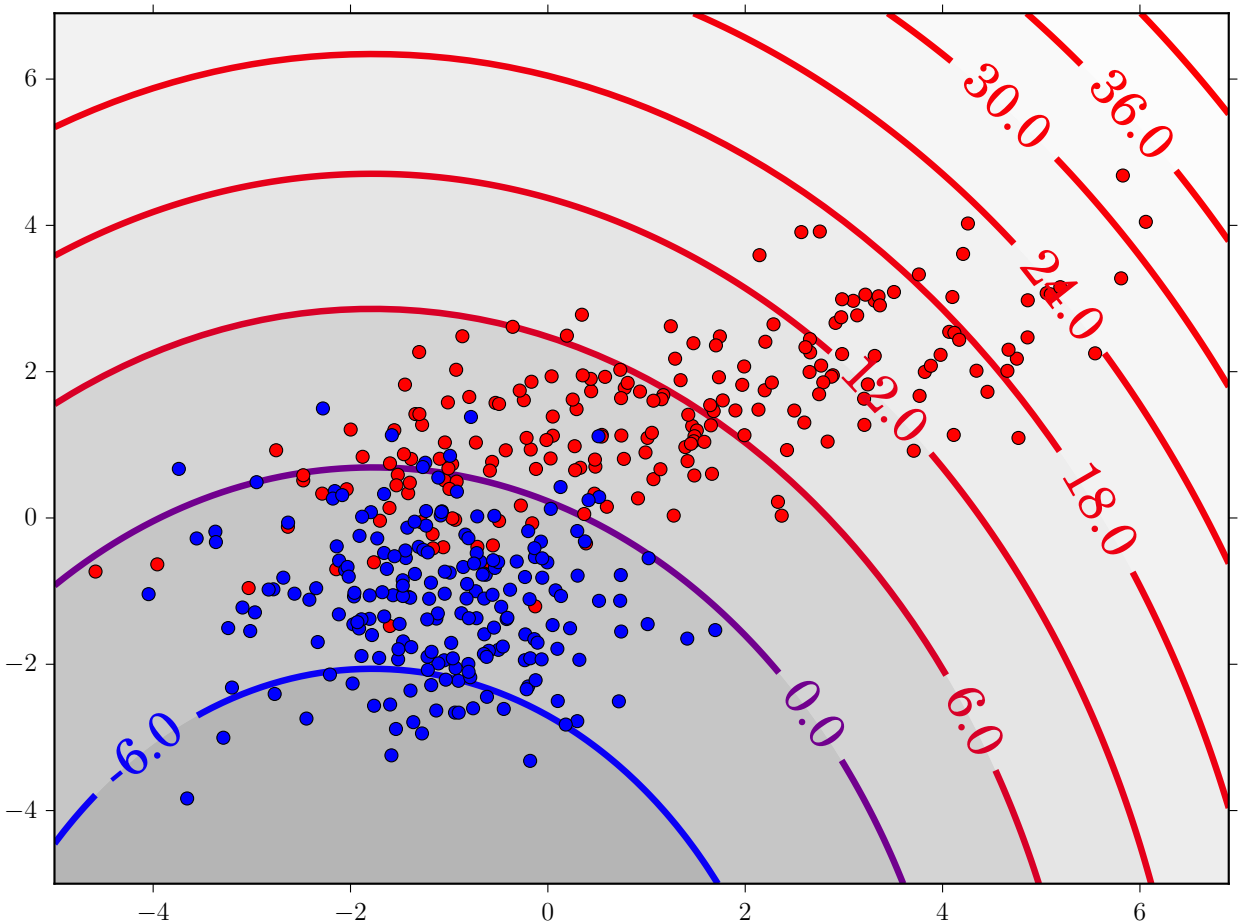
where

$$\boldsymbol{\mu}_c = \begin{bmatrix} \mu_{c,[1]} \\ \mu_{c,[2]} \\ \vdots \\ \mu_{c,[D]} \end{bmatrix}, \quad \boldsymbol{\Sigma}_c = \begin{bmatrix} \sigma^2_{c,[1]} & 0 & \dots & 0 \\ 0 & \sigma^2_{c,[2]} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma^2_{c,[D]} \end{bmatrix}$$

The *naive Bayes Gaussian* classifier corresponds to a Multivariate Gaussian classifier with *diagonal* covariance matrices (this *does not hold in general*)

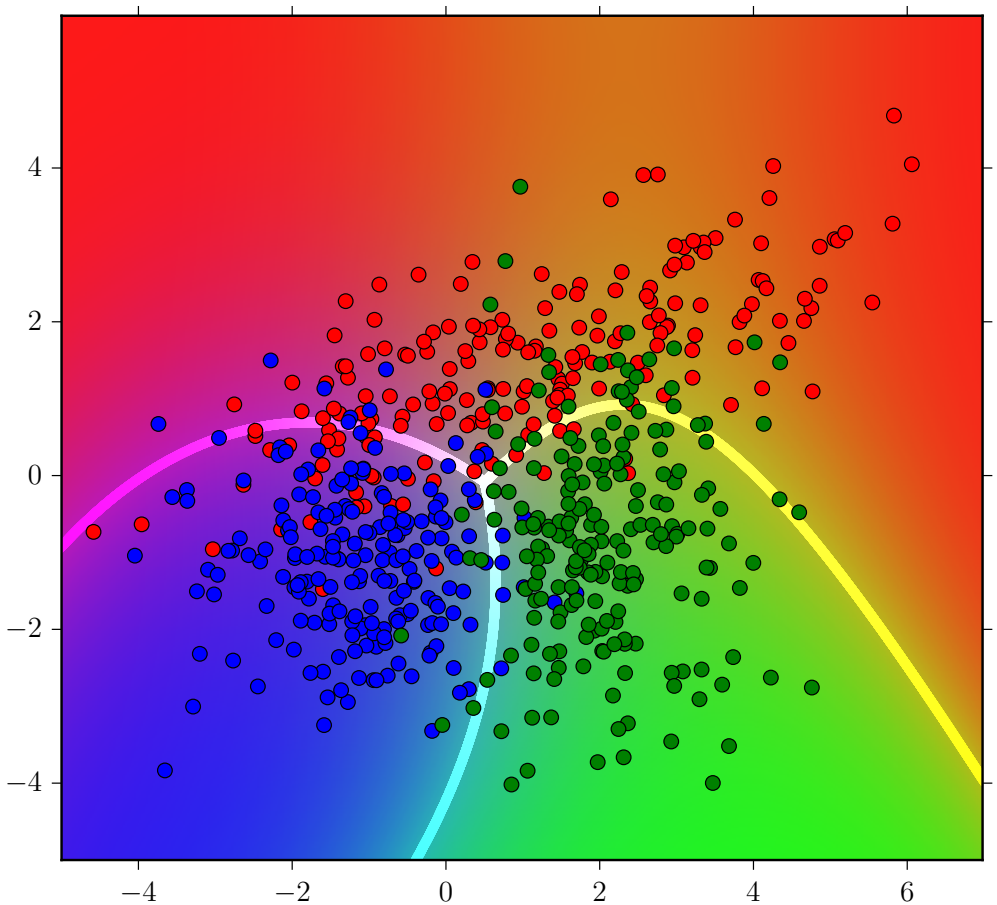
Gaussian Classifier

Binary problem — naive Bayes Gaussian classifier



Gaussian Classifier

3 class problem — naive Bayes Gaussian classifier



Gaussian Classifier

Another common Gaussian model assumes that the covariance matrices of the different classes are *tied*

- Class-independent noise: $\boldsymbol{x}_{c,i} = \boldsymbol{\mu}_c + \boldsymbol{\varepsilon}_i$, $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1})$
- Badly-conditioned problems (large dimensional data, small number of samples) — A single shared covariance matrix can be more easily estimated

The tied covariance model assumes that

$$f_{\boldsymbol{X}|C}(\boldsymbol{x}|c) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma})$$

i.e., each class has its own mean $\boldsymbol{\mu}_c$, but the covariance matrix is the same for all classes

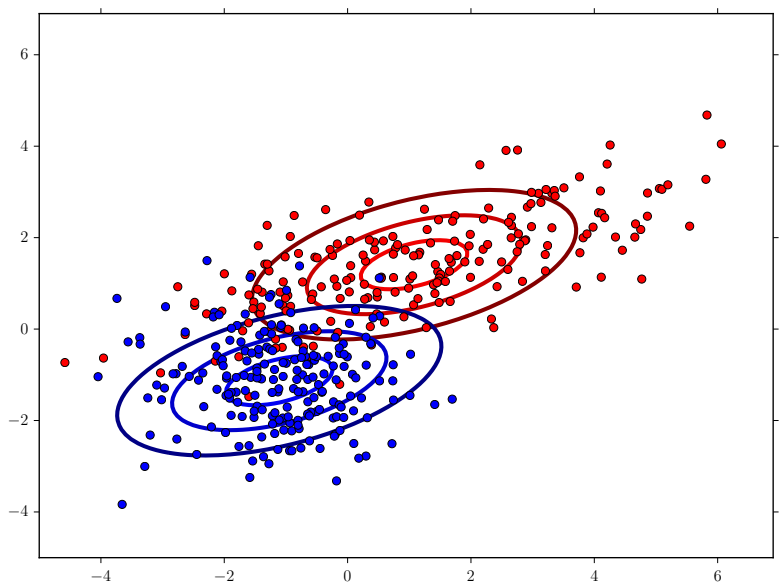
Gaussian Classifier

Again, we can estimate the parameters using the ML framework. In this case the log-likelihood does not factorize over classes

The ML solution is

$$\boldsymbol{\mu}_c^* = \frac{1}{N_c} \sum_{i|c_i=c} \boldsymbol{x}_i \ , \quad \boldsymbol{\Sigma}^* = \frac{1}{N} \sum_c \sum_{i|c_i=c} (\boldsymbol{x}_i - \boldsymbol{\mu}_c)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)^T$$

where N is the number of samples $N = \sum_{c=1}^{k} N_c$



Gaussian Classifier

Let's compute the binary log–likelihood ratios for the tied model:

$$\begin{aligned} llr(\boldsymbol{x}) &= \log \frac{f_{X|C}(\boldsymbol{x}|h_1)}{f_{X|C}(\boldsymbol{x}|h_0)} \\ &= \log \frac{\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Lambda}^{-1})}{\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_0, \boldsymbol{\Lambda}^{-1})} \\ &= \boldsymbol{x}^T \boldsymbol{b} + c \end{aligned}$$

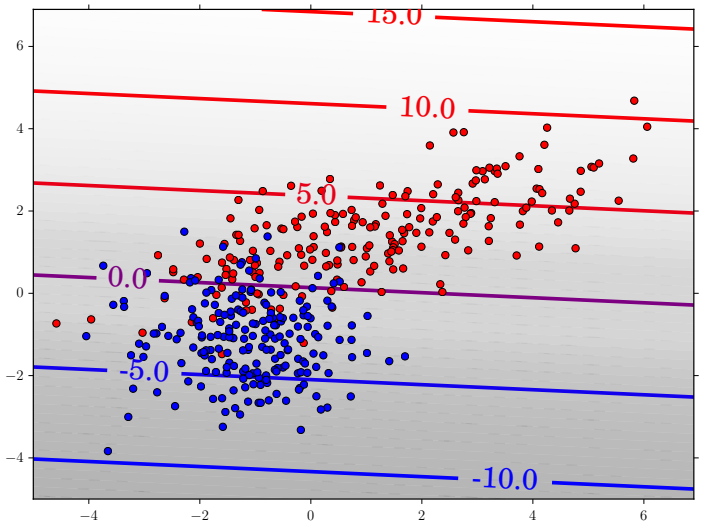
with

$$\begin{aligned} \boldsymbol{b} &= \boldsymbol{\Lambda} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \\ c &= -\frac{1}{2} (\boldsymbol{\mu}_1^T \boldsymbol{\Lambda} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^T \boldsymbol{\Lambda} \boldsymbol{\mu}_0) \end{aligned}$$

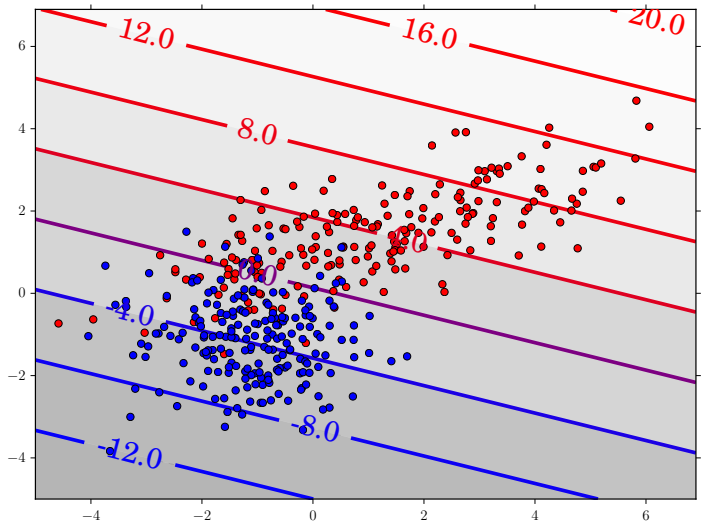
i.e., a *linear* function of $\boldsymbol{x}$

Gaussian Classifier

Binary classifier — tied covariances



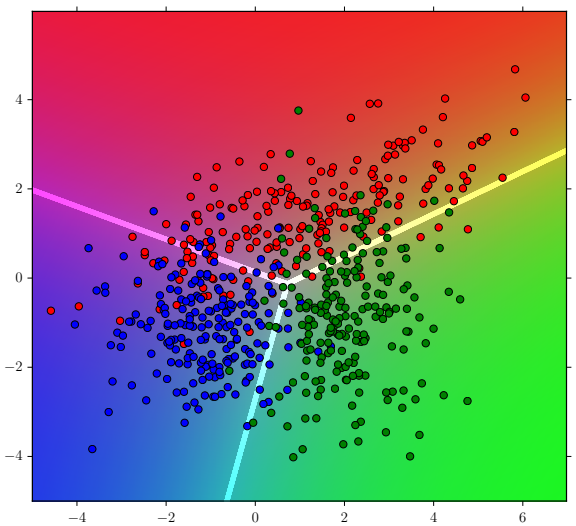
Multivariate



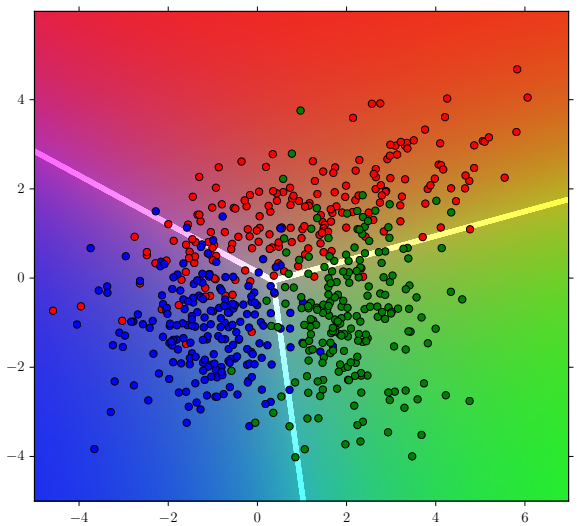
Naive Bayes

Gaussian Classifier

Multiclass classifier — tied covariances



Multivariate



Naive Bayes

Gaussian Classifier

The model is also closely related to LDA

Remember that two–class LDA looks for the direction which maximizes the generalized Rayleigh quotient

$$\frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$$

with

$$\begin{aligned} \boldsymbol{S}_W &= \boldsymbol{\Lambda}^{-1} \\ \boldsymbol{S}_B &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \end{aligned}$$

We have seen that we can solve the problem by applying the following transformations

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{x} \\ \boldsymbol{S}'_W &= \boldsymbol{I} \\ \boldsymbol{S}'_B &= \boldsymbol{\Lambda}^{\frac{1}{2}} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Lambda}^{\frac{1}{2}} \end{aligned}$$

Gaussian Classifier

Since $\boldsymbol{v} = \boldsymbol{\Lambda}^{\frac{1}{2}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ is a vector, the leading eigenvector of $\boldsymbol{S}'_B$ is

$$\boldsymbol{\nu} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$$

Projection over the LDA subspace is, up to a scaling factor, given by

$$\boldsymbol{w}^T\boldsymbol{x} = k \cdot \boldsymbol{x}^T\boldsymbol{\Lambda}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

This corresponds to the classification rule of the Gaussian model with tied covariances!

Indeed, LDA assumes that all classes have the same within–class covariance

The Gaussian model with one covariance per class is also known as Quadratic Discriminant Analysis

Gaussian Classifier

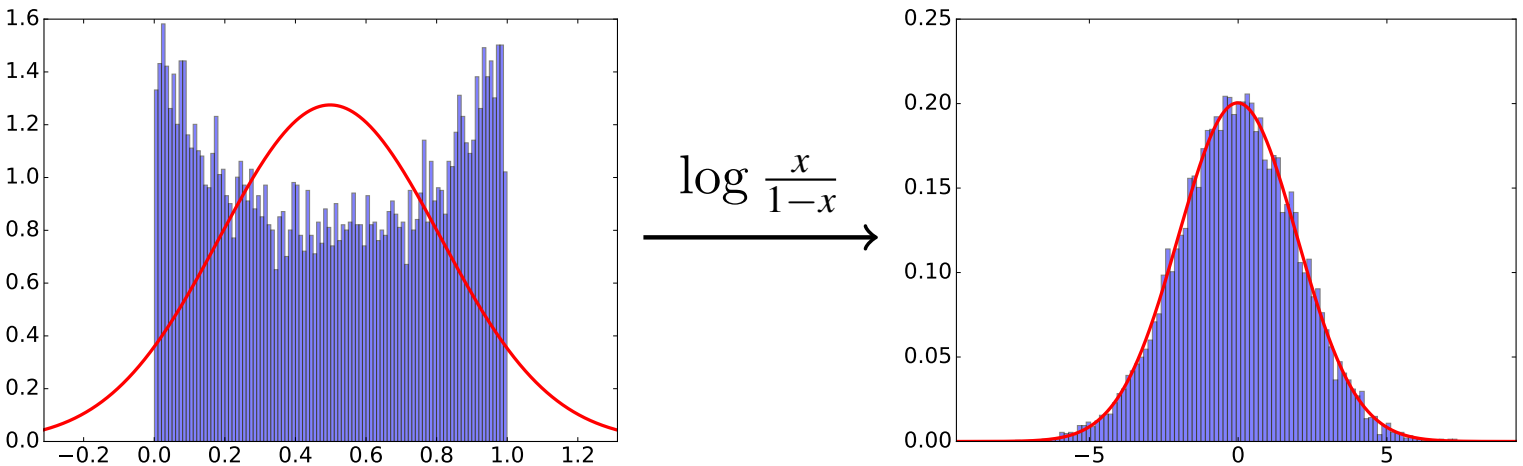
Practical considerations:

- If data is high-dimensional, PCA can simplify the estimation
- PCA also allows removing dimensions with very small variance (e.g. in the MNIST dataset, pixels that are white for all images regardless of the digit)
- Multivariate models perform better if we have enough data to reliably estimate the covariance matrices
- Naive Bayes can simplify the estimation, but may perform poorly if data are highly correlated
- Tied covariance models can capture correlations, but may perform poorly when classes have very different distribution — on the other hand, if we have reason to believe that covariances should be very similar, then the model will provide a more reliable estimate

Gaussian Classifier

Practical considerations:

- If a Gaussian model is not adequate for our data we can use a different distribution that is more appropriate
- Alternatively, the Gaussian model may still be effective for *transformed* data



Multiclass Gaussian Classifier

MNIST — Error rates for Gaussian classifier

Classifier	PCA (100)	PCA (50)	PCA (9)	PCA + LDA (100 $\rightarrow$ 9)
Naive Tied Gaussian	13.7%	14.4%	25.0%	12.3%
Tied Gaussian	12.3%	12.6%	23.7%	12.3%
Naive Gaussian	12.2%	12.3%	23.4%	11.4%
Gaussian	4.3%	3.6%	12.2%	10.2%

Multiclass Gaussian Classifier

Comments:

- The best model is the unconstrained MVG — classes have significantly different within class covariance matrices
- The Naive Bayes assumption is not good in this case (strong within-class correlations)
- PCA helps reducing the complexity — from 100 to 50 dimensions we have a significant gain. If we reduce too much we have bad results again

Multiclass Gaussian Classifier

Comments:

- Tied model + LDA achieve the same performance as Tied model without LDA — The LDA subspace contains all the information used by the tied model
- The Naive tied model, without LDA, performs slightly worse — it ignores within-class correlations.
- Our implementation of LDA whitens the within-class covariance matrix — Tied naive model and Tied model are equivalent, since $\boldsymbol{\Sigma} = \boldsymbol{I}$