ELEC 548 Byron's Review of Classification

A What is Classification

Classification is a classic instance of **supervised learning**. The goal is to define a **classifier** which properly assigns a new data point – let's call it **x** – to the appropriate discrete class $C_k \in \{C_1, \dots, C_K\}$. While there are solutions to classification problems (i.e., nonparametric) which can be posed when the number of classes, K, is not known *a priori* or is otherwise unbounded, below we will assume that it not just countable, but also finite and typically specified as part of the problem statement. Notice that the discrete number of classes is what makes this a *classification* problem — the machine learning equivalent for continuous data would generally be termed *regression*.

The process of solving a a classification problem involves two stages: **Training** – optimizing the classifier using labeled training data, and **Testing** – evaluating the performance of the optimized classifier on labeled testing data. Thus, we always divide our labeled data into two: a test data set to evaluate performance and a separate training data set to ensure that the classifier we have learned generalizes. (Aside - if we didn't do this, what would be the best performing classifier? A lookup table!) In cases when there are hyperparameters to be optimized – for example in the model or statistical distribution of the data – we can split the labeled data into three groups: a training set, a **validation** data set to pick the best model, and a test set to evaluate final performance.

Classification is a machine learning problem with a wide variety of formulations and solutions. Below, we review classification using **Probabilistic Generative Models**. For more information as well as a description of other classification approaches, a good resource is *Pattern Recognition and Machine Learning* by Christopher Bishop.

Training Data **Decision Boundaries**

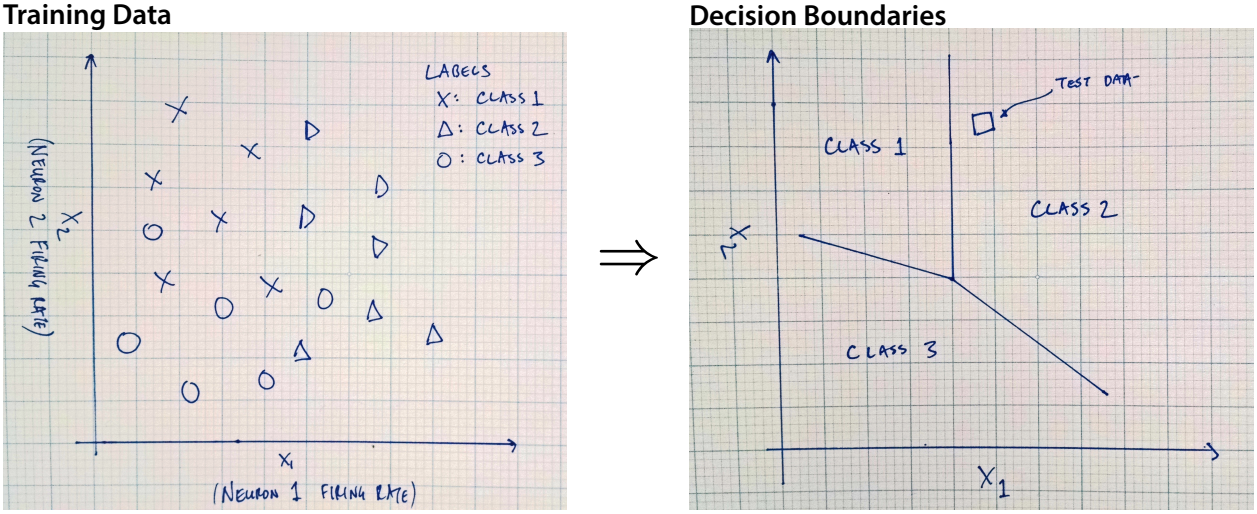

Figure 1: The classification problem involves taking this training data and finding decision boundaries for new data points.

B Classification Using Probabilistic Generative Models

In a classifier built using a probabilisitic generative model, there are two densities for each class $k \in \{1, \dots, K\}$:

— the class-conditional density, Pr $(\mathbf{x} \mid C_k)$ and

— the class priors Pr (C_k)

To **train** the classifier, we can use *maximum likelihood parameter estimation*. To use the classifier (i.e., on **test** data), we chose the class which maximizes the *a posteriori* probability. In other words, we

— use Bayes' rule to compute $\Pr(C_k \mid \mathbf{x})$

$$\begin{aligned}\Pr(C_k \mid \mathbf{x}) &= \frac{\Pr(\mathbf{x} \mid C_k)\Pr(C_k)}{\Pr(\mathbf{x})} \\ &= \frac{\Pr(\mathbf{x} \mid C_k)\Pr(C_k)}{\sum_{i=1}^{K}\Pr(\mathbf{x} \mid C_i)\Pr(C_i)}\end{aligned} \tag{1}$$

— assign $\mathbf{x}$ to class C_m where

$$m = \underset{k}{\text{argmax}}\,\Pr(C_k \mid \mathbf{x}) \tag{2}$$

B.1 Philosophy of Probabilistic Generative Models

$\Pr(\mathbf{x} \mid C_k)$ and $\Pr(C_k)$ define a "probabilistic generative model" (or just "generative model"). Unlike other classification schemes, this means that we can generate synthetic data using our model.
For example, imagine there are two classes and $\mathbf{x} \in \mathbb{R}^2$.

$$\begin{aligned}\Pr(C_1) &= 0.7 \\ \Pr(C_2) &= 0.3 \\ \Pr(\mathbf{x} \mid C_1) &= \mathcal{N}\Big(\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\Big) \\ \Pr(\mathbf{x} \mid C_2) &= \mathcal{N}\Big(\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\Big)\end{aligned}$$

To generate synthetic data from this model begin by sampling from your handy Uniform$[0, 1)$ random number generator.

— If the uniform r. v. sample is < 0.7, draw a data sample from the Gaussian $\Pr(\mathbf{x} \mid C_1)$.

— If the uniform r. v. sample is ≥ 0.7, draw a data sample from the Gaussian $\Pr(\mathbf{x} \mid C_2)$.

(Note that this is the same as flipping a biased coin with probability of heads being 0.7.)

Philosophical point: If we generate synthetic data from our model and it resembles the real data we are trying to classify, then we can have some confidence that our model of the data is good. This will mean that our classifier will be functional and we can make inferences, decisions, etc.

B.2 Maximum Likelihood Parameter Estimation

Once we write down the *training data likelihood*, maximum likelihood parameter estimation is not complicated. What is the data likelihood? Let us assume we are given training data: $\{\mathbf{x}_n, t_n\}, \quad n = 1, \ldots, N$, where for each of the N training data, t_n is the label for data point x_n. Then, the *data likelihood* is just:

Data Likelihood

$$\begin{aligned}\mathcal{L} &\equiv \Pr\left(\{\mathbf{x}_n, t_n\} \mid \boldsymbol{\theta}\right) \\ &= \prod_{i=1}^{N}\sum_{k=1}^{K} \Pr\left(\mathbf{x}_n \mid C_k, \boldsymbol{\theta}\right) \Pr\left(t_n = k \mid C_k, \boldsymbol{\theta}\right) \delta(t_n = k)\end{aligned} \tag{3}$$

where $\boldsymbol{\theta}$ are the parameters of the densities and $\delta(t_n = k) = 1$ if $t_n = k$ and 0 otherwise.

With a probabilistic generative model approach, in general, we will choose the parameters $\boldsymbol{\theta}$ that maximize the data likelihood (maximum likelihood parameter estimation).

Example: Two classes with Gaussian class-conditional density and shared covariance

Training data: $\{\mathbf{x}_n, t_n\}$, $n = 1, \ldots, N$ where

$t_n = 1$ denotes membership in C_1 and
$t_n = 0$ denotes membership in C_2

Let $\Pr(t_n = 1) = \Pr(C_1) = \pi$
$\Pr(t_n = 0) = \Pr(C_2) = (1 - \pi)$

For a data point $\mathbf{x}_n \in \mathbb{R}^D$,

$$\begin{aligned}\Pr(\mathbf{x}_n, C_1) &= \Pr(\mathbf{x}_n \mid C_1) \Pr(C_1) = \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \cdot \pi \\ \Pr(\mathbf{x}_n, C_2) &= \Pr(\mathbf{x}_n \mid C_2) \Pr(C_2) = \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \cdot (1 - \pi)\end{aligned}$$

The data likelihood for N data points is then

$$\begin{aligned}\mathcal{L} &\equiv \Pr\left(\{\mathbf{x}_n, t_n\} \mid \boldsymbol{\theta}\right) \\ &= \prod_{i=1}^{N} \Big(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \cdot \pi\Big)^{t_n} \Big(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \cdot \pi\Big)^{1-t_n}\end{aligned} \tag{4}$$

$$\begin{aligned}\log \mathcal{L} = \sum_{i=1}^{N} \Big[& t_n \log\left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right) + t_n \log(\pi) + \\ & (1 - t_n) \log\left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right) + (1 - t_n) \log(1 - \pi)\Big]\end{aligned} \tag{5}$$

where

$$\begin{aligned}&\log\left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma})\right) = \\ &\quad -\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) - \frac{1}{2} \log|\boldsymbol{\Sigma}| - \frac{D}{2} \log(2\pi)\end{aligned} \tag{6}$$

(i) Find π

Our goal is to find the value of π which maximizes $\mathcal{L}$. Because of the way we've formulated the problem this is simple calculus - take the gradient and set equal to zero.

$$\frac{\partial \log \mathcal{L}}{\partial \pi} = \sum_{n=1}^{N} \left[t_n \cdot \frac{1}{\pi} - (1 - t_n) \cdot \frac{1}{1-\pi} \right] = 0$$
$$\frac{1}{\pi} \sum_{n=1}^{N} t_n - \frac{1}{1-\pi} \sum_{n=1}^{N} (1 - t_n) = 0$$

Let N_1 = the number of points from C_1 in the training data set: $N_1 = \sum_{n=1}^{N} t_n$, similarly $N_2 = \sum_{n=1}^{N} (1 - t_n) = N - N_1$.

$$\frac{1}{\pi} N_1 - \frac{1}{1-\pi} (N - N_1) = 0$$
$$(1 - \pi) N_1 = \pi (N - N_1)$$

$$\boxed{\pi = \frac{N_1}{N}} \tag{7}$$

What this means is that the maximum likelihood estimate of the class prior probabilities are just the fraction of the training data assigned to each class.

(i) Find $\boldsymbol{\mu}_1$

Because of the way we've formulated the problem this is still simple calculus - take the gradient and set equal to zero. If you're not familiar with quadratic forms, take a look at the table of useful matrix calculus identities at the end.

$$\frac{\partial \log \mathcal{L}}{\partial \boldsymbol{\mu}} = \frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{N} \Big[t_n \log \left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right) + t_n \log(\pi) +$$
$$(1 - t_n) \log \left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right) + (1 - t_n) \log(1 - \pi) \Big]$$

Keeping only the terms that depend on $\boldsymbol{\mu}_1$:

$$
\begin{aligned}
&= \frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{N} \Big[t_n \log \left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right) \Big] \\
&= \frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{N} \Big[-t_n \frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \Big] \\
&= -\sum_{n=1}^{N} \Big(t_n \frac{1}{2} 2 \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \Big) = 0 \\
&\implies \boldsymbol{\Sigma}^{-1} \sum_{n=1}^{N} \Big(t_n \mathbf{x}_n \Big) = \boldsymbol{\Sigma}^{-1} \Big(\boldsymbol{\mu}_1 \sum_{n=1}^{N} t_n \Big)
\end{aligned}
$$

$$
\boxed{\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \mathbf{x}_n} \tag{8}
$$

Similarly, for $\boldsymbol{\mu}_2$:

$$
\boxed{\boldsymbol{\mu}_2 = \frac{1}{N - N_1} \sum_{n=1}^{N} (1 - t_n) \mathbf{x}_n} \tag{9}
$$

So what we've shown is that the maximum likelihood estimate of the class means are the sample means of the training data assigned to each class.

(i) Find $\boldsymbol{\Sigma}$

Keeping only the terms that depend on $\boldsymbol{\Sigma}$:

$$
\begin{aligned}
\log \mathcal{L} &= \sum_{i=1}^{N} \Big[t_n \log \left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right) + (1 - t_n) \log \left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right) \Big] \\
&= \sum_{i=1}^{N} \Big[-t_n \Big(\frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) - \frac{1}{2} \log |\boldsymbol{\Sigma}| \Big) \\
&\qquad - (1 - t_n) \Big(\frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2) - \frac{1}{2} \log |\boldsymbol{\Sigma}| \Big) \Big] \\
&= \sum_{i=1}^{N} \Big[-\frac{t_n}{2} \Big(\mathsf{Tr} \left(\boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} \right) - \log |\boldsymbol{\Sigma}| \Big) \\
&\qquad - \frac{1 - t_n}{2} \Big(\mathsf{Tr} \left(\boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} \right) - \log |\boldsymbol{\Sigma}| \Big) \Big] \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} &= \sum_{i=1}^{N} \Big[-\frac{t_n}{2} \Big(-\boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} \boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1} \Big) \\
&\qquad - \frac{1 - t_n}{2} \Big(-\boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} \boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1} \Big) \Big] \\
&= 0
\end{aligned}
$$

If we left and right multiply by $\boldsymbol{\Sigma}$,

$$0 = \sum_{i=1}^{N} \left[\frac{t_n}{2} \Big((\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} - \boldsymbol{\Sigma} \Big) + \frac{1 - t_n}{2} \Big((\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} - \boldsymbol{\Sigma} \Big) \right]$$

Note that the labels, $\{t_n\}$ select which of the two terms are actually calculated for each data point. Also getting rid of the $\frac{1}{2}$ terms, we can write this as

$$\begin{aligned} 0 &= \sum_{n \in C_1} \Big((\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} - \boldsymbol{\Sigma} \Big) + \sum_{n \in C_2} \Big((\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} - \boldsymbol{\Sigma} \Big) \\ 0 &= \sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} - N_1 \boldsymbol{\Sigma} + \sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} - N_2 \boldsymbol{\Sigma} \end{aligned}$$

Thus,

$$\begin{aligned} \boldsymbol{\Sigma} &= \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2, \text{ where} \\ \mathbf{S}_1 &= \frac{1}{N_1} \sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} \\ \mathbf{S}_2 &= \frac{1}{N_1} \sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\mathsf{T} \end{aligned} \tag{10}$$

So what we've shown is that the maximum likelihood estimate of the shared covariance matrix is a weighted sum of the sample covariances of each of the classes ($\mathbf{S}_1$ and $\mathbf{S}_2$) where the weighting by the fraction of points in each class. These three results make sense intuitively - absent any other information, the sample estimates of the training data will be the best estimates of the parameters.

B.3 Test Phase: Assigning a new data point to a class

Once we have trained our model, we will want to assign data to the best class. As discussed in lecture, aspects of the problem might imply cost/loss functions that would result in biased assignments. Absent these considerations, the best assignment is the *maximum a posteriori* class.

MAP Assignment

$$\begin{aligned} \hat{k} &= \operatorname*{argmax}_m \Pr(C_m \mid \mathbf{x}) \\ &= \operatorname*{argmax}_m \frac{\Pr(\mathbf{x} \mid C_m) \Pr(C_m)}{\Pr(\mathbf{x})} \\ &= \operatorname*{argmax}_m \Pr(\mathbf{x} \mid C_m) \Pr(C_m) \\ &= \operatorname*{argmax}_m \log \Pr(\mathbf{x} \mid C_m) + \log \Pr(C_m) \end{aligned} \tag{11}$$

where we've simplified by dropping the $\Pr(\mathbf{x})$ term, which is common to all classes.

Example cont'd: Assignment for 2 Gaussians with shared covariance

For the example of two classes with class-conditional Gaussian densities with a shared covariance matrix, we can further simplify by dropping all the terms which are common to both classes

$$\begin{aligned}\hat{k} &= \underset{m}{\text{argmax}}\ \Pr(C_m \mid \mathbf{x}) \\ &= \underset{m}{\text{argmax}}\ \log \Pr(\mathbf{x} \mid C_m) + \log \Pr(C_m) \\ &= \underset{m}{\text{argmax}}\ \underbrace{(\boldsymbol{\mu}_m^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_m^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_m + \log \Pr(C_m))}_{\text{call this } a_m(\mathbf{x})} \end{aligned} \tag{12}$$

This function creates a decision boundary in **x** space. What does it look like?

C Hyperplanes

A hyperplane is the D-dimensional generalization of a line in 2-dimensional space and a plane in 3-dimensional space. A hyperplane is defined as the set of all **x** such that

$$y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0. \tag{13}$$

Here, **w** determines the **direction** of the hyperplane and w_0 determines the offset from the origin. An example in 2-dimensions is to the right in Figure 2.

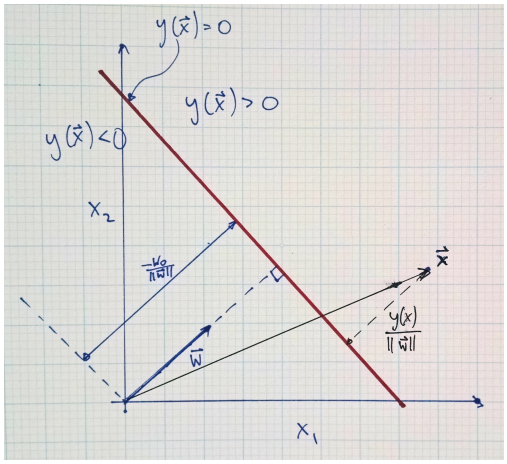

Figure 2: Hyperplane in 2D

Hyperplane Facts:

1. **w** is *orthogonal* to the hyperplane it defines.

   Consider two points, $\mathbf{x}_A$ and $\mathbf{x}_B$ which lie on a hyperplane.

$$\begin{aligned} y(\mathbf{x}_A) &= y(\mathbf{x}_B) = 0 \\ \mathbf{w}^\mathsf{T}\mathbf{x}_A + w_0 &= \mathbf{w}^\mathsf{T}\mathbf{x}_B + w_0 \\ \mathbf{w}^\mathsf{T}\underbrace{(\mathbf{x}_A - \mathbf{x}_B)}_{\text{a vector lying in the hyperplane}} &= 0 \end{aligned}$$

   $\implies$ **w** is orthogonal to any vector lying in the hyperplane.

2. The normal distance from the origin to the hyperplane is $-\dfrac{w_0}{\|\mathbf{w}\|}$.

Let **x** be a point on the hyperplane

$$\implies \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0$$

The normal distance to the plane is the projection of this (arbitrary vector) **x** onto **w**

$$(\frac{\mathbf{w}}{\|\mathbf{w}\|})^\mathsf{T}\mathbf{x} = -\frac{w_0}{\|\mathbf{w}\|}_{//}$$

3. The normal distance from any point **x** to the hyperplane is $\frac{y(\mathbf{x})}{\|\mathbf{w}\|}$.

   Project **x** onto **w**, then subtract $\frac{\mathbf{w}}{\|\mathbf{w}\|}$.

$$(\frac{\mathbf{w}}{\|\mathbf{w}\|})^\mathsf{T}\mathbf{x} + \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}_{//}$$

D Linear Decision Boundaries

In our example problem, in equation (12), a point **x** is assigned to class C_k if $a_k(\mathbf{x}) > a_j(\mathbf{x})$ for all $j \neq k$. Thus, the **decision boundary** between class C_k and class C_j is given by $a_k(\mathbf{x}) = a_j(\mathbf{x})$.

As above, we can define $a_k(\mathbf{x}) = \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k0}$, where

$$\mathbf{w}_k = \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k$$
$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^\mathsf{T}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k + \log \Pr(C_k)$$

The decision boundary is thus

$$(\mathbf{w}_k - \mathbf{w}_j)^\mathsf{T}\mathbf{x} + (w_{k0} - w_{j0}) = 0. \tag{14}$$

Note that this is the same form as (13), and thus the decision boundary *in the case of Gaussian classes with a shared covariance matrix* is a $(D-1)$-dimensional hyperplane in $\mathbb{R}^D$.

Appendix: Useful Matrix Properties and Calculus Identities

Derivative of a quadratic form for vector **x**

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} &= (\mathbf{A}+\mathbf{A}^{\mathsf{T}})\,\mathbf{x} \\ &= 2\mathbf{A}\mathbf{x} \text{ when } \mathbf{A} \text{ is symmetric}\end{aligned} \tag{15}$$

Derivative of a matrix product for matrix **X**

$$\frac{\partial}{\partial \mathbf{X}}\,\mathrm{Tr}(\mathbf{X}^{-1}\mathbf{A}) = -\mathbf{X}^{-\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{X}^{-\mathsf{T}} \tag{16}$$

Derivative of the determinant of **X**

$$\frac{\partial}{\partial \mathbf{X}}\log|\mathbf{X}| = \mathbf{X}^{-\mathsf{T}} \tag{17}$$

Invariance of matrix trace to rotation

$$\mathrm{Tr}(\mathbf{ABCD}\ldots) = \mathrm{Tr}(\mathbf{BCD}\ldots\mathbf{A}) = \mathrm{Tr}(\mathbf{CD}\ldots\mathbf{AB}) = \ldots \tag{18}$$

You will find more on the wikipedia "Matrix calculus" page or by searching for the "matrix reference manual".