ECE C143A/C243A - Neural signal processing and machine learning
Prof. Jonathan Kao
(with notes adapted from Prof. Byron Yu, CMU)

Discrete Classification

1 Some preliminaries

There are a few useful matrix properties we will use in this lecture. These include the following:

$$\begin{aligned}
\frac{d}{d\mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x} &= (\mathbf{A}+\mathbf{A}^T)\mathbf{x} \\
\frac{d}{d\mathbf{X}}\mathrm{Tr}(\mathbf{X}^{-1}\mathbf{A}) &= -\mathbf{X}^{-T}\mathbf{A}^T\mathbf{X}^{-T} \\
\frac{d}{d\mathbf{X}}\log|\mathbf{X}| &= \mathbf{X}^{-T}
\end{aligned}$$

We will also use the fact that for matrices A, B, C, D, we have that

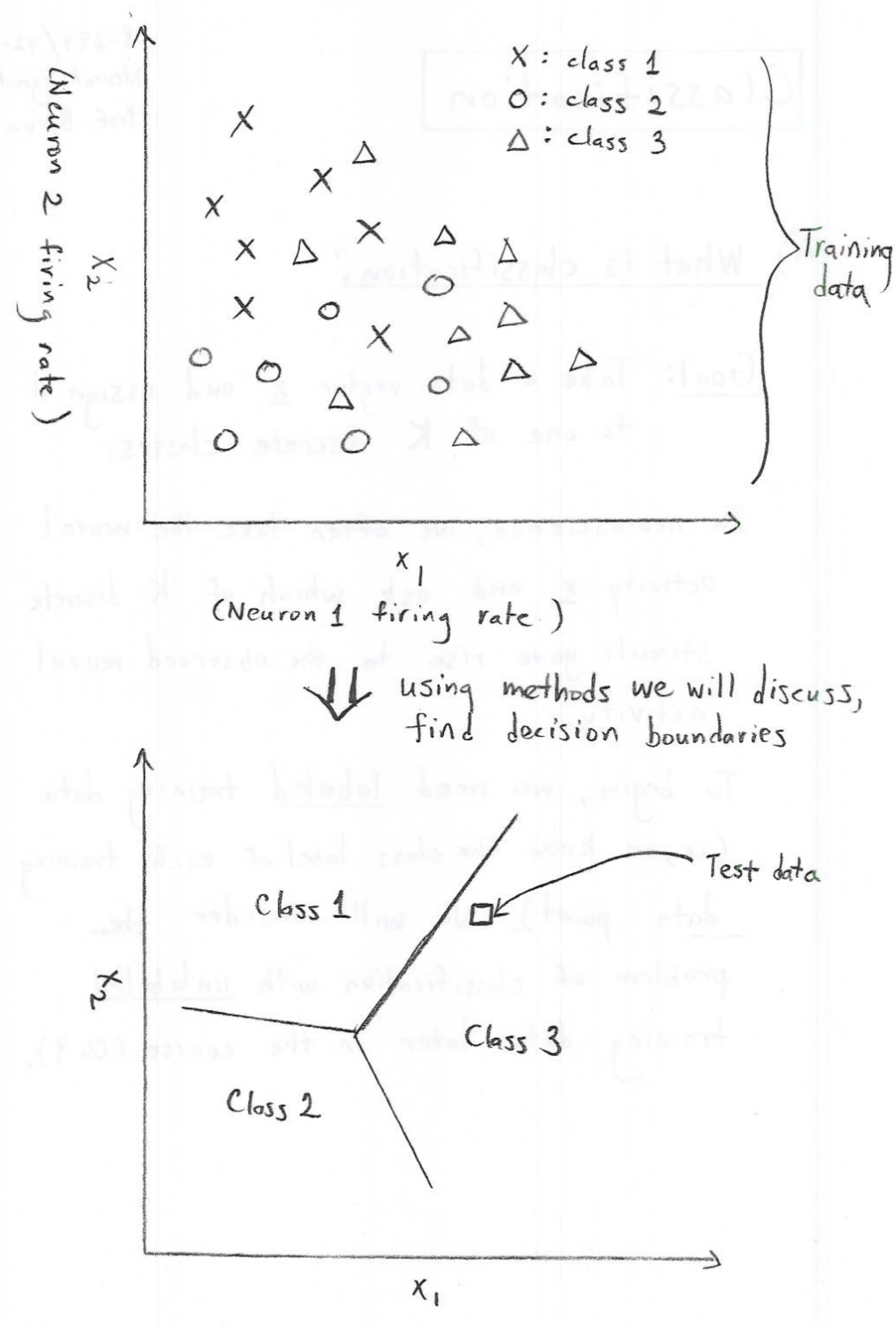
$$\mathrm{Tr}(ABCD\ldots) = \mathrm{Tr}(BCD\ldots A) = \mathrm{Tr}(CD\ldots AB)$$

In general, a good reference is available at: `http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html` or simply google "matrix reference manual." Another good reference is the matrix cookbook, available at: `http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf` or simply google "matrix cookbook."

2 What is classification?

The goal of classification is to take a data vector $\mathbf{x}$ and assign it to one of K discrete classes.

In neuroscience, we often take the neural activity $\mathbf{x}$ and ask which of k discrete stimuli gave rise to the observed neural activity.



To begin, we need *labeled* training data (i.e., we know the class label of each training data point). We will consider the problem of classification with *unlabeled* training data later on in the course.

3 Classifying using generative models

There are two phases during classification: training and testing. (In machine learning formally, there are usually training, validation, and testing sets. For this class, we'll simplify the problem by just considering training and testing sets.)

In the **training phase**, we fit class-conditional densities $P(\mathbf{x}|C_k)$ and class priors $P(C_k)$ to training data (for $k = 1, \ldots, K$).

In the **testing phase**, we:

- Compute $P(C_k|\mathbf{x})$, where $\mathbf{x}$ is the test data, using Bayes rule, i.e.,

$$\begin{aligned} P(C_k|\mathbf{x}) &= \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})} \\ &= \frac{P(\mathbf{x}|C_k)P(C_k)}{\sum_{j=1}^{K} P(\mathbf{x}|C_j)P(C_j)} \end{aligned}$$

- Decode the class $\hat{k} = \arg\max_k P(C_k|\mathbf{x})$ to test data $\mathbf{x}$.

4 Generative mdoels

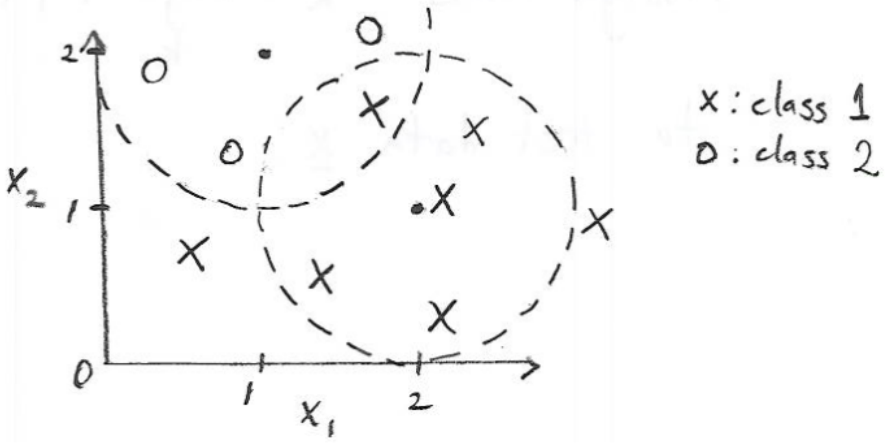
The probabilities $P(\mathbf{x}|C_k)$ and $P(C_k)$ define a "probabilistic generative model." This means that we can generate synthetic data from the model.

For example, say there are two classes and $\mathbf{x} \in \mathbb{R}^2$, with

$$\begin{aligned} P(C_1) &= 0.7 \\ P(C_2) &= 0.3 \\ P(\mathbf{x}|C_1) &= \mathcal{N}\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ P(\mathbf{x}|C_2) &= \mathcal{N}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \end{aligned}$$

To genereate one synthetic data vector $\mathbf{x}$, first flip a biased coin with probability 0.7 of coming up heads, and,

- If heads, draw from the Gaussian $P(\mathbf{x}|C_1)$.



- If tails, draw from the Gaussian $P(\mathbf{x}|C_2)$.

The philosophy of generative models is as follows. If we generate synthetic data from the model, and it looks a lot like the real data we're trying to model, then this constitutes a good model for the real data. We can then use the generative model to make optimal inferences, decisions, etc.

5 Training phase: maximum-likelihood and parameter estimation

One common technique to train models using machine learning is to maximize the likelihood of the data with respect to the model parameters. The idea is that, given a model, the best parameters are those under which the observed data was most probable. We'll go through a concrete example of this.

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

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

- $t_n = 1$ denotes class C_1.
- $t_n = 0$ denotes class C_2.

We also set,

$$\begin{aligned}
\textbf{Pr}\ (t_n = 1) &= P(C_1) \\
&= \pi \\
\textbf{Pr}\ (t_n = 0) &= P(C_2) \\
&= 1 - \pi
\end{aligned}$$

Let's consider a data point, $\mathbf{x}_n \in \mathbb{R}^D$. The probability of having observed this data point and it coming from C_1 is

$$\begin{aligned}
P(\mathbf{x}_n, C_1) &= P(\mathbf{x}_n|C_1)P(C_1) \\
&= \mathcal{N}(x_n|\mu_1, \Sigma)\pi
\end{aligned}$$

Similarly, the probability of having observed this data point and it it coming from C_2 is

$$\begin{aligned}
P(\mathbf{x}_n, C_2) &= P(\mathbf{x}_n|C_2)P(C_2) \\
&= \mathcal{N}(\mathbf{x}_n|\mu_2, \Sigma)(1 - \pi)
\end{aligned}$$

In the training set, we observe N data points together, as well as their classes. Therefore, the *likelihood* of this data under our model is given by:

$$\begin{aligned}
\mathcal{L} &= P(\{\mathbf{x}_n, t_n\}|\pi, \mu_1, \mu_2, \Sigma) \\
&= \prod_{i=1}^{N} \left(\mathcal{N}(\mathbf{x}_n|\mu_1, \Sigma)\pi)\right)^{t_n} \left(\mathcal{N}(\mathbf{x}_n|\mu_2, \Sigma)(1 - \pi)\right)^{1-t_n}
\end{aligned}$$

Products and exponentials are often inconvenient to work with, and so we deal with this by taking the log of the likelihood. Because $\log(\cdot)$ is a monotonically increasing function of its argument, maximizing $\log(\mathcal{L})$ is equivalent to maximizing $\mathcal{L}$.

Concretely,

$$\begin{aligned}
\log \mathcal{L} &= \sum_{n=1}^{N} (t_n \log \mathcal{N}(\mathbf{x}_n|\mu_1, \Sigma) + t_n \log \pi + ... \\
&\qquad +(1 - t_n) \log \mathcal{N}(\mathbf{x}_n|\mu_2, \Sigma) + (1 - t_n) \log(1 - \pi))
\end{aligned}$$

where

$$\log \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma) = -\frac{1}{2}(\mathbf{x}_n - \mu_k)^T \Sigma^{-1}(\mathbf{x}_n - \mu_k) - \frac{1}{2}\log|\Sigma| - \frac{D}{2}\log(2 \cdot 3.14159...)$$

Next, we want to find want the optimal parameters $\pi, \mu_1, \mu_2, \Sigma$ are.

Now, we find each parameter via optimization.

1. Finding π.

To optimize, we set:

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

To help simplify this expression, we define N_1 to be the number of data points from C_1 and N_2 to be the number of data points from C_2. That is,

$$\begin{aligned}N_1 &= \sum_{n=1}^{N} t_n \\ N_2 &= \sum_{n=1}^{N}(1-t_n)\end{aligned}$$

With this, we solve for π:

$$\begin{aligned}(1-\pi)\sum_{n=1}^{N} t_n - \pi\sum_{n=1}^{N}(1-t_n) &= 0 \\ (1-\pi)N_1 - \pi(N - N_1) &= 0\end{aligned}$$

and therefore,

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

2. Finding μ_1.

$$\begin{aligned}\frac{\partial \log \mathcal{L}}{\partial \mu_1} &= \sum_{n=1}^{N}\left[t_n \, \frac{1}{2} \, 2 \, \Sigma^{-1}(\mathbf{x}_n - \mu_1)\right]\\ &= \mathbf{0}\end{aligned}$$

This leaves us with the following equation,

$$\Sigma^{-1}\left(\sum_{n=1}^{N} t_n \mathbf{x}_n\right) = \Sigma^{-1}\left(\mu_1 \sum_{n=1}^{N} t_n\right)$$

which means that

$$\boxed{\mu_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n}$$

Analogously,

$$\boxed{\mu_2 = \frac{1}{N_1}\sum_{n=1}^{N}(1 - t_n)\mathbf{x}_n}$$

3. Finding Σ.

Focusing on only the terms that involve Σ,

$$\begin{aligned}\log \mathcal{L} &= \sum_{n=1}^{N}\left[t_n \left(-\frac{1}{2}\mathrm{Tr}(\Sigma^{-1}(\mathbf{x}_n - \mu_1)(\mathbf{x}_n - \mu_1)^T - \frac{1}{2}\log|\Sigma|\right)\right.\\ &\qquad \left. +(1 - t_n)\left(-\frac{1}{2}\mathrm{Tr}(\Sigma^{-1}(\mathbf{x}_n - \mu_1)(\mathbf{x}_n - \mu_1)^T - \frac{1}{2}\log|\Sigma|\right)\right]\end{aligned}$$

Hence,

$$\begin{aligned}\frac{\partial \log \mathcal{L}}{\partial \Sigma} &= \sum_{n=1}^{N}\left[t_n\left(-\frac{1}{2}\cdot -\Sigma^{-1}(\mathbf{x}_n-\mu_1)(\mathbf{x}_n-\mu_1)^T\Sigma^{-1}-\frac{1}{2}\Sigma^{-1}\right)\right.\\ &\quad \left.+(1-t_n)\left(-\frac{1}{2}\cdot -\Sigma^{-1}(\mathbf{x}_n-\mu_2)(\mathbf{x}_n-\mu_2)^T\Sigma^{-1}-\frac{1}{2}\Sigma^{-1}\right)\right]\\ &= \mathbf{0}\end{aligned}$$

Rearranging yields

$$\frac{1}{2}\sum_{n\in C_1}(\mathbf{x}_n-\mu_1)(\mathbf{x}_n-\mu_1)^T-\frac{1}{2}N_1\Sigma+\frac{1}{2}\sum_{n\in C_2}(\mathbf{x}_n-\mu_2)(\mathbf{x}_n-\mu_2)^T-\frac{1}{2}N_2\Sigma=\mathbf{0}$$

Therefore,

$$\boxed{\Sigma=\frac{N_1}{N}S_1+\frac{N_2}{N}S_2}$$

where

$$\begin{aligned}S_1 &= \frac{1}{N_1}\sum_{n\in C_1}(\mathbf{x}_n-\mu_1)(\mathbf{x}_n-\mu_1)^T\\ S_2 &= \frac{1}{N_2}\sum_{n\in C_2}(\mathbf{x}_n-\mu_2)(\mathbf{x}_n-\mu_2)^T\end{aligned}$$

6 Test phase: assigning a new data point to a class

Recall what the point of the test phase is. We want to assign a class to a given data point according to our fitted model. Here,

$$\begin{aligned}
\hat{k} &= \arg\max_k P(C_k|\mathbf{x}) \\
&= \arg\max_k \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})} \\
&= \arg\max_k P(\mathbf{x}|C_k)P(C_k) \\
&= \arg\max_k (\log P(\mathbf{x}|C_k) + \log P(C_k)) \\
&= \arg\max_k \left(\mu_k^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log P(C_k)\right) \\
&= \arg\max_k a_k(\mathbf{x})
\end{aligned}$$

What do the decision boundaries look like in $\mathbf{x}$ space?

7 Hyperplanes

A hyper plane is the D-dimensional generalization of a line in 2-dim space and a plane in 3-dim space.

A hyperplane is defined as the set of all $\mathbf{x}$ such that

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0 \tag{1}$$

where $\mathbf{w}$ determines the direction of the hyperplane and w_0 determines its offset from the origin.

A few facts.

1. $\mathbf{w}$ is orthogonal to the hyperplane.

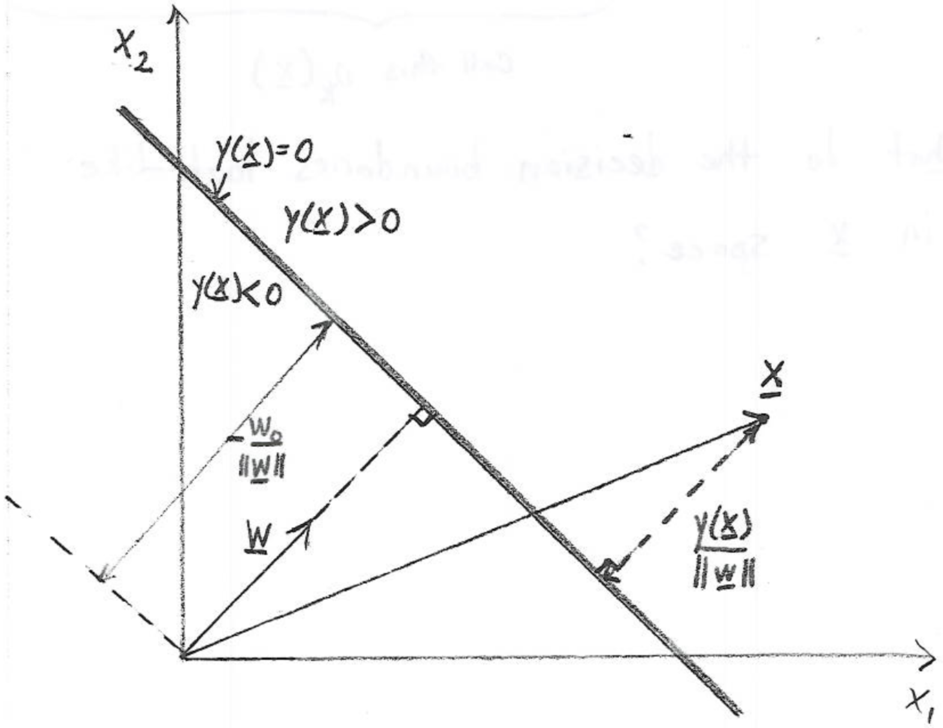
   To show this, consider two points $\mathbf{x}_A$ and $\mathbf{x}_B$ lying on the hyperplane. By construction, we have that $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$. Thus,

   $$\mathbf{w}^T\mathbf{x}_A + w_0 = \mathbf{w}^T\mathbf{x}_B + w_0$$

   so that

   $$\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$$

   Now, $\mathbf{x}_A - \mathbf{x}_B$ is a vector lying in the hyperplane. Thus, $\mathbf{w}$ is orthogonal to any vector lying in the hyperplane.



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

   Let $\mathbf{x}$ be a point on the hyperplane, and thus $\mathbf{w}^T\mathbf{x} + w_0 = 0$. The normal distance is the projection of $\mathbf{x}$ onto $\mathbf{w}$. Hence, the distance from the origin is

$$\left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^T \mathbf{x} = -\frac{w_0}{||\mathbf{w}||}$$

3. Normal distance from any point $\mathbf{x}$ to hyperplane is $y(\mathbf{x})/||\mathbf{w}||$.

   To show this, we project $\mathbf{x}$ onto $\mathbf{w}$, then subtract $-w_0/||\mathbf{w}||$. Thus,

$$\left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^T \mathbf{x} + \frac{w_0}{||\mathbf{w}||} = \frac{y(\mathbf{x})}{||\mathbf{w}||}$$

8 Linear decision boundaries

From earlier, a point $\mathbf{x}$ is assigned to class C_k if $a_k(x) > a_j(x)$ for all $j \neq k$. Thus, the decision boundary between class C_k and class C_j is given by $a_k(\mathbf{x}) = a_j(\mathbf{x})$. Let $a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$, where

$$\begin{aligned}
\mathbf{w}_k &= \Sigma^{-1}\mu_k \\
w_{k0} &= -\frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k + \log P(C_k)
\end{aligned}$$

The decision boundary is thus:

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

This takes the same form as the hyperplane from earlier, so the decision boundary is a $(D-1)$ dimensional hyperplane in $\mathbb{R}^D$.