

# Introduction to Machine Learning (67577)

## Recitation 6 Classification - Probabilistic Models

Second Semester, 2023

### Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Logistic Regression</b>                       | <b>2</b> |
| 1.1      | A Probabilistic Model For Noisy Labels . . . . . | 2        |
| 1.1.1    | The Hypothesis Class . . . . .                   | 3        |
| 1.1.2    | Learning via Maximum Likelihood . . . . .        | 3        |
| <b>2</b> | <b>Bayes Classifiers</b>                         | <b>4</b> |
| 2.1      | Linear Discriminant Analysis . . . . .           | 6        |
| 2.2      | Quadratic Discriminant Analysis . . . . .        | 9        |

## 1 Logistic Regression

Last week we discussed about classification algorithms. We divided the world to classes, and for a given sample we wanted to determine which class it belongs to. But what happens if we are not 100% sure which class it belongs to?

This week we will continue to talk about classification, but instead of asking "which class this sample belongs to?" we will ask "what is the **probability** that this sample belongs to a certain class?"

### 1.1 A Probabilistic Model For Noisy Labels

Recall that in linear regression, when assuming Gaussian errors we modeled the relation  $\mathcal{X} \rightarrow \mathcal{Y}$  as  $\mathbf{y} = \mathbf{X}\mathbf{w} + \varepsilon$  and  $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_m)$ . Notice that as  $\varepsilon$  is a random variable,  $\mathbf{y}$  too is a random variable distributing as a multi-variate Gaussian:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 I_m) \quad (1)$$

Focusing on a pair  $(\mathbf{x}_i, y_i)$ , we can think of the above as the **conditional probability** of  $y_i$  given  $\mathbf{x}_i$ :

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \mathcal{N}(y_i|\phi_{\mathbf{w}}(\mathbf{x}_i), \sigma^2) \quad \text{where} \quad \phi_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} \quad (2)$$

where the notation of  $\mathcal{N}(y_i|\mathbf{x}_i, \mathbf{w})$  means the probability of observing the response  $y_i$  for the feature vector of  $\mathbf{x}_i$  and coefficients vector  $\mathbf{w}$ . We also condition on  $\mathbf{w}$  (though is not a random variable) to explicitly state the dependence on the model parameters. In other words, we assumed that each sample  $(\mathbf{x}, y)$  is such that the **expected value** of the label  $y$  is linear in  $\mathbf{x}$ . As we are dealing with a regression model and  $y_i \in \mathbb{R}$ , the support of the random variable  $y_i|\mathbf{x}_i, \mathbf{w}$  is  $\mathbb{R}$ .

Let us adapt the model above to fit for classification problems. We would like to assume that  $y_i$  distributes **Bernoulli** with a probability  $p(\mathbf{x}_i)$  that somehow relates with  $\mathbf{x}_i$ , and captures how likely is  $y_i$  of being 1:

$$p(y_i|\mathbf{x}_i) = \text{Ber}(y_i|p(\mathbf{x}_i)) \quad (3)$$

How shall  $p(\mathbf{x}_i)$  relate with  $\mathbf{x}_i$ ? Unlike the linear regression model, we cannot assume a linear function  $\phi_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$  as  $\phi_{\mathbf{w}} \in \mathbb{R}$  while  $p(\mathbf{x}_i)$  is restricted to  $[0, 1]$ . Instead, we would like to choose some **link** function  $\phi_{\mathbf{w}} : \mathbb{R} \rightarrow [0, 1]$  that is monotone increasing and maps  $(-\infty, \infty)$  bijectively to  $(0, 1)$ . Define the relation to be:

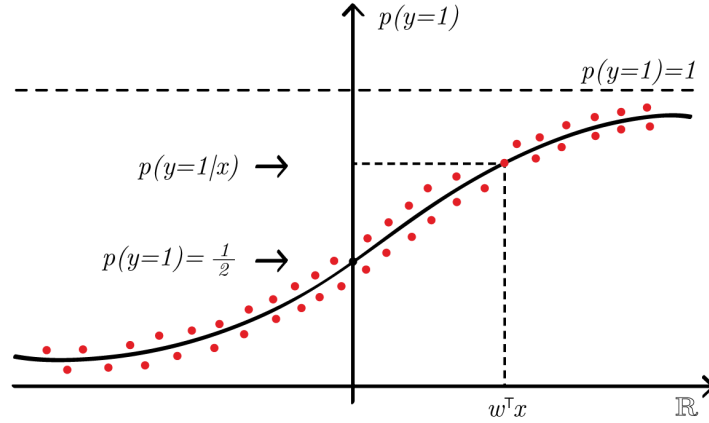
$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \text{Ber}(y_i|\phi_{\mathbf{w}}(\mathbf{x}_i)), \quad \phi_{\mathbf{w}} := \sigma(\mathbf{x}^\top \mathbf{w}) \quad (4)$$

where  $\sigma : \mathbb{R} \mapsto [0, 1]$  is the **sigmoid** function, also known as the **logistic** function:

$$\sigma(x) := \frac{e^x}{e^x + 1} \quad (5)$$

This function is indeed monotone increasing and maps  $(-\infty, \infty)$  bijectively to  $(0, 1)$ :

- As  $\mathbf{x}^\top \mathbf{w} \rightarrow -\infty$  then  $\sigma(\mathbf{x}^\top \mathbf{w}) \rightarrow 0$ . This means that it is "very unlikely" that the label is 1:  $p(y_i = 1|\mathbf{x}_i, \mathbf{w}) \rightarrow 0$ .
- As  $\mathbf{x}^\top \mathbf{w} \rightarrow \infty$  then  $\sigma(\mathbf{x}^\top \mathbf{w}) \rightarrow 1$ . This means that it is "very likely" that the label is 1:  $p(y_i = 1|\mathbf{x}_i, \mathbf{w}) \rightarrow 1$ .



**Figure 1:** Illustration of fitted logistic function for values corresponding to  $\mathbf{x}^\top \mathbf{w}$ .

**R** In (4) we modeled the logistic regression model for binary classification problems. Notice that the Bernoulli distribution can be seen as a private case of the Multinomial distribution  $Multinomial(p_1, \dots, p_K)$ ,  $\sum_i p_i = 1$ ,  $0 \leq p_i \leq 1$ . We can expand the above logistic regression model to fit multi-classification problems by extending the sigmoidal function to what is known as the **softmax** function, also denoted by  $\sigma : \mathbb{R}^K \mapsto [0, 1]^K$  such that its  $i^{th}$  entry is:  $\sigma(\mathbf{x})_i = e^{x_i} / \sum_{j=1}^K e^{x_j}$

### 1.1.1 The Hypothesis Class

So we would like to define the hypothesis class of logistic regression as:

$$\mathcal{H}_{logistic} := \left\{ h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{x}^\top \mathbf{w}) \mid \mathbf{w} \in \mathbb{R}^{d+1} \right\} \quad (6)$$

where  $\mathbf{w} \in \mathbb{R}^{d+1}$  since we incorporate the intercept variable into  $\mathbf{w}$  (and a zeroth coordinate of 1 to  $\mathbf{x}$ ) similar to the way we did in the linear regression hypothesis class. Notice that the hypotheses are defined  $h_{\mathbf{w}} : \mathbb{R}^{d+1} \rightarrow [0, 1]$  and not  $h_{\mathbf{w}} : \mathbb{R}^{d+1} \rightarrow \{0, 1\}$  as required for classification problems. Since  $\{0, 1\} \subset [0, 1]$ , we can use the training sample to select a function in  $\mathcal{H}_{logistic}$ . This means we will be able to train a model, but how will we predict over new samples? Suppose our learner chose some  $h_{\mathbf{w}} \in \mathcal{H}_{logistic}$ . As we think of  $h_{\mathbf{w}}(\mathbf{x})$  as an estimate of the probability that the label corresponding to  $\mathbf{x}$  is 1, we can use it for classification. If  $h_{\mathbf{w}}(\mathbf{x})$  is low, the label is likely to be 0, and if  $h_{\mathbf{w}}(\mathbf{x})$  is high, the label is likely to be 1. Choosing some **cutoff** value  $\alpha \in [0, 1]$ , our class prediction will be:  $\hat{y} := \mathbb{1}_{[h_{\mathbf{w}}(\mathbf{x}) > \alpha]}$ . To choose a fitting value for  $\alpha$  we can calculate the Type-I and Type-II errors of the classifier and plot its ROC curve.

### 1.1.2 Learning via Maximum Likelihood

Once we have defined the logistic regression model (4) and hypothesis class (6), we would like to come up with a learner. To do so we will use the *maximum likelihood principle*. Recall, that by the maximum likelihood principle, we estimate the parameters (the desired hypothesis) as those that give the highest probability to the data.

Let  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$  be our sample of independent observations, assuming that  $\mathbf{y}_i \sim Ber(\phi_{\mathbf{w}}(\mathbf{x}_i))$

where  $\phi$  is the logistic function. Therefore, the likelihood of  $\mathbf{w} \in \mathbb{R}^{d+1}$  is:

$$\begin{aligned}
 \mathcal{L}(\mathbf{w}|\mathbf{X}, \mathbf{y}) &= \mathbb{P}(y_1, \dots, y_m | \mathbf{X}, \mathbf{w}) \\
 &= \prod_i \mathbb{P}(y_i | \mathbf{x}_i, \mathbf{w}) \\
 &= \prod_{i:y_i=1} \mathbb{P}(y_i | \mathbf{x}_i, \mathbf{w}) \cdot \prod_{i:y_i=0} \mathbb{P}(y_i | \mathbf{x}_i, \mathbf{w}) \\
 &= \prod_{i:y_i=1} p_i(\mathbf{w}) \cdot \prod_{i:y_i=0} (1 - p_i(\mathbf{w})) \\
 &= \prod p_i(\mathbf{w})^{y_i} (1 - p_i(\mathbf{w}))^{1-y_i}
 \end{aligned} \tag{7}$$

where  $p_i(\mathbf{w}) = \sigma(\mathbf{x}_i^\top \mathbf{w})$ . Since the log function is monotone increasing we can maximize the log-likelihood  $\ell(\mathbf{w}) := \log \mathcal{L}(\mathbf{w})$  instead:

$$\begin{aligned}
 \ell(\mathbf{w}|\mathbf{X}, \mathbf{y}) &= \sum_{i=1}^m [y_i \log(\sigma(\mathbf{x}_i^\top \mathbf{w})) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i^\top \mathbf{w}))] \\
 &= \sum_{i=1}^m \left[ y_i \log\left(\frac{e^{\mathbf{x}_i^\top \mathbf{w}}}{1 + e^{\mathbf{x}_i^\top \mathbf{w}}}\right) + (1 - y_i) \log\left(\frac{1}{1 + e^{\mathbf{x}_i^\top \mathbf{w}}}\right) \right] \\
 &= \sum_{i=1}^m \left[ y_i (\log(e^{\mathbf{x}_i^\top \mathbf{w}}) - \log(1 + e^{\mathbf{x}_i^\top \mathbf{w}})) - (1 - y_i) \log(1 + e^{\mathbf{x}_i^\top \mathbf{w}}) \right] \\
 &= \sum_{i=1}^m [y_i \mathbf{x}_i^\top \mathbf{w} - \log(1 + e^{\mathbf{x}_i^\top \mathbf{w}})]
 \end{aligned} \tag{8}$$

And therefore, choosing the function  $h_{\mathbf{w}} \in \mathcal{H}_{\text{logistic}}$  by applying the maximum likelihood principle means that:

$$\hat{\mathbf{w}} := \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^{d+1}} \sum_{i=1}^m [y_i \mathbf{x}_i^\top \mathbf{w} - \log(1 + e^{\mathbf{x}_i^\top \mathbf{w}})] \tag{9}$$

In a following recitation, we will prove that the logistic regression objective function is concave, which means that it can be solved efficiently using gradient based methods, such as Gradient Descent or the Newton–Raphson method.

**R** Instead of deriving the learner using the maximum likelihood principle, we could derive it using the ERM principle with the following loss function:  $\ell(h_{\mathbf{w}}) := \log(1 + \exp(-y \langle \mathbf{w}, \mathbf{x} \rangle))$ . We would have reached the same optimization expression.

## 2 Bayes Classifiers

When deriving the logistic regression model, we assumed a probability distribution over the response set  $\mathcal{Y}$  and treated the observations as deterministic values influencing the distribution of the response. We could however *assume* that both the response set and the domain set follow some *joint probability distribution*  $\mathcal{F}$  over  $\mathcal{X} \times \mathcal{Y}$ . Under such assumption we are now able to look at the data from two different perspectives. Given the sample  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ , we could first consider  $\mathbf{x}_1, \dots, \mathbf{x}_m$  as fixed and learn  $y_i | \mathbf{x}_i$ . That is, the distribution of the response given the observation. This is the perspective used in the logistic regression, as well as in the linear regression model. For the second perspective, we consider  $y_1, \dots, y_m$  as fixed and ask how do the observations depend on the responses  $\mathbf{x}_i | y_i$ .

$$\underbrace{f_{Y|X=\mathbf{x}}(y) \cdot f_X(\mathbf{x})}_{\text{Perspective I}} = \underbrace{f_{X,Y}(\mathbf{x}, y)}_{\text{Joint Probability Distribution}} = \underbrace{f_{X|Y=y}(\mathbf{x}) \cdot f_Y(y)}_{\text{Perspective II}} \tag{10}$$

■ **Example 2.1** Consider the classification task of separating images of cats and dogs. Let the domain space be RGB images of 1024-by-1024 pixels and the response set be  $\mathcal{Y} := \{\text{cat}, \text{dog}\}$ . Further assume there exists some joint probability distribution  $\mathcal{F}$  over  $\mathcal{X} \times \mathcal{Y}$ . Considering the first perspective where we wish to learn the conditional distribution  $y_i | \mathbf{x}_i$ . By doing so we try to discriminate a given picture being either of cat or of dog. That is, we simply try to understand how to differentiate between these two possibilities. If however we consider the second perspective, we wish to learn the conditional distribution of  $\mathbf{x}_i | y_i$ . This probability distribution describes what sort of observations might be seen for a given response. That is, what do pictures of cats or of dogs look like. ■

Besides providing insights into the manner in which different samples behave - how do cat pictures look? how do dog pictures look? - what benefit do we get from considering this second perspective? How can it be used for predicting the response of a given sample? To answer this question we use the Bayes' Law of conditional probability. Given a joint probability distribution function  $f_{X,Y}$  over the observation  $\mathbf{x}$  and response  $y$  we can express the conditional distribution  $y | \mathbf{x}$  using the conditional distribution  $\mathbf{x} | y$ :

$$\underbrace{f_{Y|X=\mathbf{x}}(y)}_{\text{posterior}} = \frac{\underbrace{f_{X|Y=y}(\mathbf{x})}_{\text{likelihood}} \cdot \underbrace{f_Y(y)}_{\text{prior}}}{\underbrace{f_X(\mathbf{x})}_{\text{evidence}}} \quad (11)$$

where  $f_{X|Y=y}(\mathbf{x})$  is the likelihood function,  $f_Y(y)$  is the marginal distribution of  $Y$  and  $f_X(\mathbf{x})$  the marginal distribution of  $X$  functioning as a normalization factor. The marginal  $f_Y$  reflects our *belief* regarding the true value of  $y$  *before* (i.e. *prior*) observing the data. Therefore, the *A-Posteriori* distribution  $f_{Y|X=\mathbf{x}}$ , i.e. the conditional of  $y$  *after* observing the data, is the weighting of the likelihood function by our prior belief and the evidence of the data.

From this relation we are able to derive the Bayes Optimal classifier:

**Definition 2.1** Let  $f_{\mathcal{D}}$  be a joint probability distribution function over  $\mathcal{D} := \mathcal{X} \times \mathcal{Y}$ . The *Bayes Optimal Classifier* is defined as

$$h^{\text{Bayes}}(\mathbf{x}) := \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f_{Y|X=\mathbf{x}}(y)$$

That is, we predict the response that (given the observation) achieves the highest probability. Since we are searching for a maximizer, we can use Bayes' Law to express the Bayes Optimal classifier as

$$h^{\text{Bayes}}(\mathbf{x}) := \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f_{Y|X=\mathbf{x}}(y) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \frac{f_{X|Y=y}(\mathbf{x}) f_Y(y)}{f_X(\mathbf{x})} \stackrel{(*)}{=} \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f_{X|Y=y}(\mathbf{x}) f_Y(y)$$

where  $(*)$  is because given  $\mathbf{x}$ ,  $f_X(\mathbf{x})$  is constant over the different values of  $y$ .

**Theorem 2.1** Let  $f_{X,Y}$  be a joint probability distribution function over  $\mathcal{X} \times \mathcal{Y}$  for  $\mathcal{Y} = [k]$ ,  $k \in \mathbb{N}$ . The Bayes optimal classifier  $h^{\text{Bayes}}(\mathbf{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f_{Y|X=\mathbf{x}}(y)$  is the optimal classifier with respect to the misclassification error. Namely that for any hypothesis  $h : \mathcal{X} \rightarrow \mathcal{Y}$  it holds that

$$L_{\mathcal{D}}(h^{Bayes}) \leq L_{\mathcal{D}}(h).$$

*Proof.* Let  $h : \mathcal{X} \rightarrow \mathcal{Y}$  be a hypothesis. We'll find an expression for the misclassification error:

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [h(\mathbf{x}) \neq y] = \int_{\mathbf{x}, y} f_{X,Y}(\mathbf{x}, y) \mathbb{1}_{[h(\mathbf{x}) \neq y]} d\mathbf{x} dy = \int_{\mathbf{x}} f_X(\mathbf{x}) \int_y f_{Y|X=\mathbf{x}}(y) \mathbb{1}_{[h(\mathbf{x}) \neq y]} dy d\mathbf{x}$$

Lets now focus on the inner integral over  $y$ :

$$\int_y f_{Y|X=\mathbf{x}}(y) \mathbb{1}_{[h(\mathbf{x}) \neq y]} dy = \int_y f_{Y|X=\mathbf{x}}(y) (1 - \mathbb{1}_{[h(\mathbf{x}) = y]}) dy = 1 - f_{Y|X=\mathbf{x}}(h(\mathbf{x}))$$

From the definition of a Bayes optimal classifier we get:

$$\begin{aligned} f_{Y|X=\mathbf{x}}(h^{Bayes}(\mathbf{x})) &\geq f_{Y|X=\mathbf{x}}(h(\mathbf{x})) \\ 1 - f_{Y|X=\mathbf{x}}(h(\mathbf{x})) &\geq 1 - f_{Y|X=\mathbf{x}}(h^{Bayes}(\mathbf{x})) \end{aligned}$$

Putting it all together we have that:

$$L_{\mathcal{D}}(h) = \int_{\mathbf{x}} f_X(\mathbf{x}) (1 - f_{Y|X=\mathbf{x}}(h(\mathbf{x}))) d\mathbf{x} \geq \int_{\mathbf{x}} f_X(\mathbf{x}) (1 - f_{Y|X=\mathbf{x}}(h^{Bayes}(\mathbf{x}))) d\mathbf{x} = L_{\mathcal{D}}(h^{Bayes})$$

■

It is important to note that in reality we do not know the underlying distribution  $\mathcal{D}$ , to whom all we have is a mere window in the form of the dataset. Therefore, we cannot program an algorithm that would find the maximizer of the posterior distribution. As such, we must think of the Bayes optimal classifier as an Oracle - a “black box” entity capable of solving the problem of finding the maximizer of the posterior.

## 2.1 Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) algorithm is a realization of the Bayes Optimal classifier. In this model we assume that samples of different labels have different Gaussian distributions.

**Definition 2.2** Let  $\Omega = \{1, \dots, K\}$  for  $K \in \mathbb{N}$  be a sample space. A random variable  $X : \Omega \rightarrow [0, 1]$  follows a *Multinomial* distribution with parameter  $\boldsymbol{\pi} \in [0, 1]^K, \sum \pi_i = 1$  if  $\mathbb{P}(X = j) = \pi_j, j \in [K]$ . We denote  $X \sim \text{Mult}(\boldsymbol{\pi})$ .

Explicitly, the LDA model assumes the following generative model:

1. Each sample “selects” a label  $y_i$  according to a multinomial distribution with  $K$  classes.
2. Then, the sample itself is drawn from the conditional probability of  $X|Y$  where  $X$  denotes the random variable of sampling some samples  $X = \mathbf{x}$  and  $Y$  denotes the random variable of  $Y = y, y \in [K]$ . The distribution used to model  $X|Y$  is a Gaussian distribution where each label is characterized by a different mean vector  $\{\boldsymbol{\mu}_k \in \mathbb{R}^d\}_{k=1}^K$  but the same covariance matrix  $\Sigma \in \mathbb{R}^{d \times d}$ .

Namely, for any  $i \in 1, \dots, m$  we assume that:

$$\begin{aligned} y_i &\sim \text{Mult}(\boldsymbol{\pi}) \\ \mathbf{x}_i | y_i &\sim \mathcal{N}(\boldsymbol{\mu}_{y_i}, \Sigma) \end{aligned} \tag{12}$$

Under these assumptions, predicting the class of a new sample is done by simply using the Bayes Law over the Bayes Optimal classifier to get:

$$\hat{y}(\mathbf{x}) := \operatorname{argmax}_y \mathbb{P}(y|\mathbf{x}) = \operatorname{argmax}_y \frac{\mathbb{P}(\mathbf{x}|y)\mathbb{P}(y)}{\mathbb{P}(x)}$$

**Claim 2.2** Let  $f_{X,Y}$  be the pdf of  $\mathcal{D}$  a joint distribution over  $\mathcal{X} \times \mathcal{Y}$  and suppose

$$y \sim \text{Mult}(\pi), \quad \mathbf{x}|y \sim \mathcal{N}(\mu_k, \Sigma)$$

for  $\pi \in [0, 1]^K, \sum \pi_k = 1$ . Then the Bayes Optimal classifier is given by:

$$\hat{y}(\mathbf{x}) := \operatorname{argmax}_k a_k^\top \mathbf{x} + b_k, \quad a_k := \Sigma^{-1} \mu_k, \quad b_k := \log(\pi_k) - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k$$

*Proof.* To prove the claim we begin with expressing  $f_{Y|X}(k)$  using Bayes Law:

$$f_{Y|X=\mathbf{x}}(k) = \frac{f_{X|Y=k}(\mathbf{x}) \cdot f_Y(k)}{f_X(\mathbf{x})} = \frac{f_{X|Y=k}(\mathbf{x}) \cdot f_Y(k)}{\sum_{k'} f_{X|Y=k'}(\mathbf{x}) \cdot f_Y(k')} = \frac{\pi_k \cdot \mathcal{N}(\mathbf{x}|\mu_k, \Sigma)}{\sum_{k'} \pi_{k'} \cdot \mathcal{N}(\mathbf{x}|\mu_{k'}, \Sigma)}$$

Notice, that since we wish to maximize the posterior distribution  $f_{Y|X}$  for a *given* sample  $\mathbf{x}$ , we can ignore the evidence  $f_X(\mathbf{x})$ . Then, by the pdf of the multivariate Gaussian then:

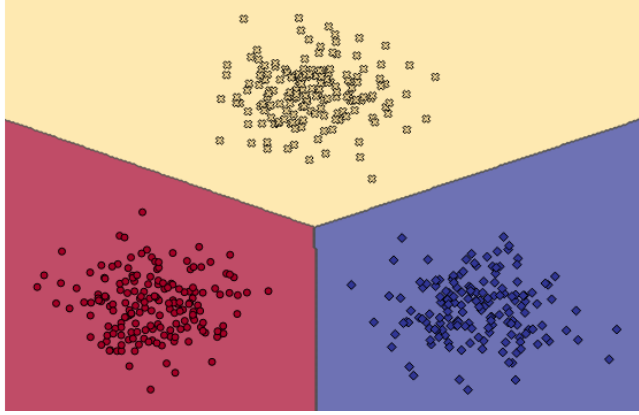
$$\begin{aligned} \operatorname{argmax}_k f_{Y|X=\mathbf{x}}(k) &= \operatorname{argmax}_k \frac{f_{X|Y=k}(\mathbf{x}) \cdot f_Y(k)}{f_X(\mathbf{x})} \\ &= \operatorname{argmax}_k \pi_k \cdot \mathcal{N}(\mathbf{x}|\mu_k, \Sigma) \\ &= \operatorname{argmax}_k \pi_k \cdot \frac{1}{Z} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^\top \Sigma^{-1}(\mathbf{x} - \mu_k)\right) \\ &= \operatorname{argmax}_k \log(\pi_k) - \frac{1}{2} \mathbf{x}^\top \Sigma^{-1} \mathbf{x} + \mathbf{x}^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k \\ &= \operatorname{argmax}_k \log(\pi_k) + \mathbf{x}^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k \end{aligned}$$

for  $Z := \sqrt{(2\pi)^d |\Sigma|}$  the Gaussians' normalization factor. Denote  $a_k := \Sigma^{-1} \mu_k, b_k := \log(\pi_k) - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k$  and we conclude that  $\hat{y}(\mathbf{x}) = \operatorname{argmax}_k a_k^\top \mathbf{x} + b_k$ . ■

Notice, that we were able to remove  $\mathbf{x}^\top \Sigma^{-1} \mathbf{x}$  since we assumed that the classes are generated from Gaussians with the *same* covariance matrix, and therefore the expression did not depend on any specific class  $k$ . The claim above, besides showing that under the LDA assumptions we are dealing with a Bayes classifier, also tells us something about the classifier learned. Looking at the expression derived from the assumptions, we see that the classification is in fact done by some *linear separator/discriminant*.

It can be shown, by taking the log *–likelihood* ratio between the likelihood for being classified for a class divided the likelihood for being classified for the other class, that the decision boundary between the classes is linear, similar to [Figure 2](#).

Multi-class LDA Decision Boundary



**Figure 2: LDA Decision Boundaries for a multiclass setup of three Gaussians.**

### Learning A LDA Classifier

So, in order to predict using an LDA classifier we need to know the class probabilities  $\pi$ , the Gaussian means  $\{\mu_k\}$  and the covariance matrix  $\Sigma$ . To do so we derive the maximum likelihood estimators. Let us generalize the binary LDA model (i.e. of classification) to a multiclassification LDA model.

Then, the LDA model assumptions are:

$$\begin{aligned} y &\sim \text{Mult}(\pi) \\ \mathbf{x}|y = k &\sim \mathcal{N}(\mu_k, \Sigma) \end{aligned} \quad (13)$$

Given a training set  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$  then the likelihood is given by:

$$\begin{aligned} \mathcal{L}(\Theta|\mathbf{X}, \mathbf{y}) &= f_{X,Y|\Theta}(\{(\mathbf{x}_i, y_i)\}_{i=1}^m) \\ &\stackrel{iid}{=} \prod_{i=1}^m f_{X,Y|\Theta}(\mathbf{x}_i, y_i) \\ &= \prod_{i=1}^m f_{X|Y=y_i}(\mathbf{x}_i) \cdot f_{Y|\Theta}(y_i) \\ &= \prod_{i=1}^m \mathcal{N}(\mathbf{x}_i|\mu_{y_i}, \Sigma) \cdot \text{Mult}(y_i|\pi) \end{aligned}$$

Since the log transformation is monotonous increasing finding the maximizer of the likelihood is equivalent to finding the maximizer of the log-likelihood.

$$\begin{aligned} \ell(\Theta|\mathbf{X}, \mathbf{y}) &= \log(\prod_i \mathcal{N}(\mathbf{x}_i|\mu_{y_i}, \Sigma) \cdot \text{Mult}(y_i|\pi)) \\ &= \sum_i \log(\mathcal{N}(\mathbf{x}_i|\mu_{y_i}, \Sigma)) + \log(\text{Mult}(y_i|\pi)) \\ &= \sum_i \log\left(\frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu_{y_i})^\top \Sigma^{-1}(\mathbf{x}_i - \mu_{y_i})\right)\right) + \log(\pi_{y_i}) \\ &= \sum_i \log(\pi_{y_i}) - \frac{d}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma| - \frac{1}{2}(\mathbf{x}_i - \mu_{y_i})^\top \Sigma^{-1}(\mathbf{x}_i - \mu_{y_i}) \\ &= \sum_k \left[ n_k \log(\pi_k) - \frac{1}{2} \sum_{i:y_i=k} (\mathbf{x}_i - \mu_k)^\top \Sigma^{-1}(\mathbf{x}_i - \mu_k) \right] - \frac{md}{2} \log(2\pi) - \frac{m}{2} \log |\Sigma| \end{aligned}$$

for  $n_k = \sum_i \mathbb{1}_{[y_i=k]}$ . To find the maximizers we derive with respect to the different parameters  $\{\pi_k\}, \{\mu_k\}, \Sigma$  and equate to zero. However, before doing so, recall the constraint on  $\pi$ :  $\pi \in$



$[0, 1]^K$ ,  $\sum_k \pi_k = 1$ . To solve the optimization problem with the constraint we use the Lagrange Multipliers method. Since the constraint is  $\sum_k \pi_k = 1 \iff \sum_k \pi_k - 1 = 0$ , we define the function  $g(\pi) = \sum_k \pi_k - 1$  and the Lagrangian

$$\mathcal{L} = \ell(\Theta|\mathbf{X}, \mathbf{y}) - \lambda g(\pi)$$

Now, we derive with respect to each of the parameters including  $\lambda$ . Beginning with the class probabilities then:

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \frac{\partial}{\partial \pi_k} \ell(\Theta|\mathbf{X}, \mathbf{y}) - \lambda \frac{\partial}{\partial \pi_k} g(\pi) = \frac{n_k}{\pi_k} - \lambda = 0 \implies \pi_k = \frac{n_k}{\lambda} \quad (14)$$


To find the value of  $\lambda$  we replace  $\pi$  in the constraint with the expression found in (14).

$$1 = \sum_k \pi_k = \sum_k \frac{n_k}{\lambda} \iff \lambda = m$$

and therefore, the MLE of the class probabilities are  $\hat{\pi}_k^{MLE} = \frac{n_k}{m}$ . To find the MLE of the Gaussian parameters notice that the log-likelihood above is identical to the one derived of a single Gaussian while considering only samples sharing the same label. As such

$$\hat{\mu}_k^{MLE} = \frac{1}{n_k} \sum_i \mathbb{1}_{[y_i=k]} \mathbf{x}_i, \quad \hat{\Sigma}^{MLE} = \frac{1}{m} \sum_i (\mathbf{x}_i - \hat{\mu}_{y_i}^{MLE}) (\mathbf{x}_i - \hat{\mu}_{y_i}^{MLE})^\top$$

The covariance estimator above is called the *pooled covariance* where we account for the fact that each sample originates from a different Gaussian with a different expectation. As written above, this is the *biased* estimator. The *unbiased* estimator is given by replacing the  $\frac{1}{m}$  factor with  $\frac{1}{m-K}$ .

 Looking closely at the expressions derived we in fact realize that the MLE predicts the values proportional to what is found in the training set.

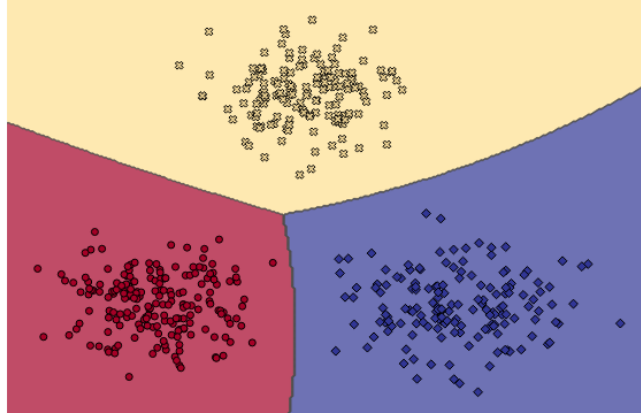
## 2.2 Quadratic Discriminant Analysis

In the LDA algorithm we assumed the data is generated from a set of Gaussians, differing in their mean but sharing the same covariance matrix (12). The Quadratic Discriminant Analysis algorithm allows different covariance matrices. That is, for any  $i \in 1, \dots, m$  we assume that:

$$\begin{aligned} y_i &\sim \text{Mult}(\pi) \\ \mathbf{x}_i | y_i = k &\sim \mathcal{N}(\mu_k, \Sigma_k) \end{aligned} \quad (15)$$

By enabling different covariance matrices the quadratic expression (in  $\mathbf{x}$ ) of  $\mathbb{P}(y = k|\mathbf{x})$  does not cancel out. This in turn causes the decision boundaries between classes to be quadratic rather than linear. In both Figure 2 and Figure 3 the same data was used to fit either the LDA or the QDA models. We can see that while the decision boundaries of the LDA fit (Figure 2) are linear, in the case of QDA (Figure 3) we get curved (quadratic) boundaries.

Multi-class QDA Decision Boundary



*Figure 3: QDA Decision Boundaries for a multiclass setup of three Gaussians.*

### Learning A QDA Classifier

Fitting a QDA classifier is very similar to the process of fitting a LDA classifier. The difference is in the estimation of the covariance matrices. In this case we fit a different covariance matrix for each class based on the samples of the class:

$$\hat{\Sigma}_k^{MLE} := \frac{1}{m_k} \sum_{i: y_i=k} (\mathbf{x}_i - \hat{\mu}_k^{MLE}) (\mathbf{x}_i - \hat{\mu}_k^{MLE})^\top \quad (16)$$