# Week 5 - Classification

CM of 2h followed by a TD of 1h

We come back today to a **supervised** setting but we change of task.

Instead of doing regression, we will be talking about **classification**.

Mathematically, instead of observing real numbers, we observe categories.

And we will start with a bit of math.

-- **Bayes classifier**

Suppose our data is such that $X \in \mathbb{R}^p$ and $Y \in \{0, 1\}$ with $(X, Y) \sim p(x, y)$

> - The $X$ can be different health markers (blood pressure, weight, etc.) and $Y$ is whether a person has diabetes or not
> - The $X$ can be a time series recording from the brain and $Y$ is whether the persons is thinking about his car or his wife

**Goal**: determine a function $f$ that takes as input $X$ and outputs an estimation for $Y$ with **minimum** error

- In **regression** this error was a squared difference between real values, here we do something different

Our loss function is defined as

$$\mathcal{L}(f) = \mathbb{E}_{(X,Y)\sim p(x,y)}[\mathbf{1}_{\{f(X)\neq Y\}}] = \iint \mathbf{1}_{\{f(x)\neq y\}}\, p(x, y)\mathrm{d}x\mathrm{d}y = \mathrm{Prob}\left(f(X) \neq Y\right)$$

> We want to minimize our probability of being wrong. Sounds reasonable, right?

We want to determine a function $f$ that **minimizes** this loss function.

- **Remember** that we can always write a joint expectation as follows:

$$\begin{aligned}
\mathbb{E}_{(X,Y)\sim p(x,y)}[g(X, Y)] &= \iint g(x, y)\, p(x, y)\mathrm{d}x\mathrm{d}y \\
&= \iint g(x, y)\, p(y|x)p(x)\mathrm{d}y\mathrm{d}x \\
&= \int \left( \int g(x, y)p(y|x)\mathrm{d}y \right) p(x)\mathrm{d}x \\
&= \int \mathbb{E}_{Y\sim p(y|x)}\left[g(X, Y)|X = x\right]\, p(x)\mathrm{d}x \\
&= \mathbb{E}_{X\sim p(x)}\left[\mathbb{E}_{Y\sim p(y|x)}\left[g(X, Y)|X = x\right]\right] \\
&= \mathbb{E}_{X}\left[\mathbb{E}_{Y|X=x}\left[g(X, Y)\right]\right]
\end{aligned}$$

So we can rewrite the loss function as (remember that $Y \in \{0, 1\}$)

$$
\begin{aligned}
\mathcal{L}(f) &= \mathbb{E}_X\Big[\mathbb{E}_{Y|X=x}[\mathbf{1}_{\{f(X)\neq Y\}}]\Big] \\
&= \mathbb{E}_X\Big[\mathbb{P}(Y=1|X=x)\ \mathbf{1}_{\{f(x)\neq 1\}} + \mathbb{P}(Y=0|X=x)\ \mathbf{1}_{\{f(x)\neq 0\}}\Big] \\
&= \mathbb{E}_X\Big[\mathbb{P}(Y=1|X=x)\ \mathbf{1}_{\{f(x)=0\}} + \mathbb{P}(Y=0|X=x)\ \mathbf{1}_{\{f(x)=1\}}\Big] \\
&= \mathbb{E}_X\Big[\Big(1-\mathbb{P}(Y=0|X=x)\Big)\ \mathbf{1}_{\{f(x)=0\}} + \Big(1-\mathbb{P}(Y=1|X=x)\Big)\ \mathbf{1}_{\{f(x)=1\}}\Big]
\end{aligned}
$$

- **Important**: to minimize this loss function it suffices to see what is going on for each fixed $x$. We will construct the function $f$ for each $x$
- Note that for each fixed $x$ we can have either $f(x)=0$ or $f(x)=1$
  - Choosing $f(x)=0$, we contribute with $1-\mathbb{P}(Y=0|X=x)$ to the loss function
  - Choosing $f(x)=1$, we contribute $1-\mathbb{P}(Y=1|X=x)$ to the loss function
- We would like that for **each** $x$ the $f(x)$ **contributes the least** to the loss function:

$$
f(x) = \underset{k}{\operatorname{argmin}}\ \Big(1-\mathbb{P}(Y=k|X=x)\Big) = \underset{k}{\operatorname{argmax}}\ \mathbb{P}(Y=k|X=x)
$$

In words, we can **conclude** that the best classifier $f(x)$ is one that will **assign** to $x$ the class with the maximum conditional probability.

- It is easy to show that this same expression extends to when we have **more than two classes**

It seems that the **problem of classification is solved**, right ?

> Show figure with the Bayes decision boundary on our simulated dataset
>
> The boundary is defined as the points $x$ where $\mathbb{P}(Y=1|X=x) = \mathbb{P}(Y=0|X=x)$

Well, the thing is that we **never** have access to the **true distribution** $P(X,Y)$ generating the data and, therefore, to the conditional probability $P(Y|X)$.

In fact, the Bayes classifier is what we call an **unattainable gold standard** to which we would like to be as close as possible

> Note that the it is also the **lowest classification error** that we can obtain.
>
> It is not necessarily zero, but it is the **minimum value** that we could hope to get.
>
> As such, it serves as a **the ideal baseline** for showcasing new classifier ideas with simulated data.

There are many ways of attempting to **estimate** this conditional probability and then classify data points based on that. In what follows, we will consider 4 different types of classifiers:

- K-nearest neighbors (K-NN)
- Logistic Regression
- Linear Discriminant Analysis
- Naive Bayes classifier -- **I will talk about this one next week**

These are **not necessarily** the most performant types of classifier nowadays, but they form the **basis for** many other powerful ones that have been developed in recent years, such as support vector machines and neural networks.

-- **K-NN**

This is probably the **simplest type of classifier** that one could envision.

We approximate the conditional probability for a given point $x_0$ based on the labels of other data points from a training dataset.

- Define $\mathcal{N}_0$ as the set of $K$ points from the training dataset that are the **closest** to $x_0$
- Estimate the probability for $x_0$ being in class $k$ as

$$\mathbb{P}(Y = k|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} \mathbf{1}_{\{y_i = k\}}$$

- Classify $x_0$ into the class with the largest estimated probability.

> Show some figures for KNN
>
> Show figure with training-testing error when $1/K$ increases

- K-NN is a **nonparametric** classifier,
  - It makes absolutely **no hypothesis** about the data
- It can be **very flexible** (which is good) but may **not be optimal** when we know something about the data generating distribution (which is bad).
- It works OK for low dimensions -- problems with the **curse of dimensionality**

The next few classifier schemes make **some hypothesis** about the data.

-- **Logistic regression**

For convenience, we will still consider that $Y \in \{0, 1\}$

We would like to approximate the conditional probability $\mathbb{P}(Y = 1|X)$ using a linear model, like

$$f(X) = \beta_0 + \beta^T X$$

where $\beta = [\beta_1, \ldots, \beta_p] \in \mathbb{R}^p$

However, $f(X)$ can attain any real value, so it is not a good model for probabilities.

We use a **logistic function** instead:

$$\mathbb{P}(Y = 1|X = x) = p(x) \approx \hat{p}(x) = \frac{\exp\left(f(x)\right)}{1 + \exp\left(f(x)\right)} = \frac{\exp\left(\beta_0 + \beta^T x\right)}{1 + \exp\left(\beta_0 + \beta^T x\right)}$$

which is a number between 0 and 1 and, therefore, a possible model for probabilities.

> Draw a figure with how a logistic function looks like:
>
> - Always positive
> - Between zero and one

The next natural question to ask is **how can we fit the parameters** $\beta$ from this model?

To fit a logistic regression model to data one uses the concept of **maximum likelihood** estimation, which we will discuss in the TD later today. The main thing to know is that there is **no closed form** solution to fitting the parameters of the logistic regression, as opposed to multiple linear regression from before.

- The loss function here is also known as **cross-entropy** loss and is very well known in deep learning community, although people often don't know what it means.

Note that **although** the relation between $X$ and $p(X)$ is non-linear, the **boundary** between the regions in which the data is considered from one class or the other is **linear**. A **boundary** is defined as

$$\mathbb{P}(Y=1|X) = \mathbb{P}(Y=0|X) \iff p(X) = 1 - p(X) \iff \log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta^T X = 0$$

and, therefore, the line separating the two regions is given by $\beta_0 + \beta^T X = 0$

> Confirm this result with a Figure
>
> Show the command for logistic regression in R
>
> - Mention that we can also do **statistical tests** on the coefficients, but we won't be doing it here

We can **extend** the logistic regression model to cases with **more than just two** classes. A common approach is to follow what people call a *softmax* encoding, which gives for each class $k = 1, \ldots, K$

$$\mathbb{P}(Y=k|X=x) = \frac{\exp\left(\beta_{k0} + \beta_k^T x\right)}{\sum_{\ell=1}^{L} \exp\left(\beta_{\ell 0} + \beta_\ell^T x\right)}$$

This is also sometimes called **multinomial regression**

Note that softmax can be seen as a way of assigning values between 0 and 1 to the results of different non-linear functions $f_k(x)$ and ensure that they sum to one (i.e. a probability function). This is often used in neural networks and elsewhere.

-- **Generative classification**

In logistic regression we were modeling directly the conditional probability $\mathbb{P}(Y=k|X=x)$ and this is what we call a **discriminative approach** to classification:

- We are only interested in being good in separating the classes.

A different approach would be to model the statistics of the data generating distribution $P(X,Y)$ and, from that, obtain an approximation to conditional probability of the Bayes classifier. This is what we call a **generative approach** to classification:

- Being able to separate classes is a byproduct from knowing the full distribution of the data.

But you may ask: **why bother with generative classification?**

There are a few reasons:

- If the distribution of the data is approximately **Gaussian** and there are not that many training samples, using generative approaches gives better results
- If we start with two classes and then **add a new class**, a discriminative approach has to be retrained from scratch, while generative models can easily be extended

- It is **easier to fit** the parameters of generative models as compared to discriminative ones

> I have included a PDF file to the website with more comparisons between the two approaches

Using Bayes' theorem we can write

$$\mathbb{P}(Y=k|X=x)=\frac{\mathbb{P}(X=x|Y=k)\mathbb{P}(Y=k)}{\mathbb{P}(X=x)}=\frac{\mathbb{P}(X=x|Y=k)\mathbb{P}(Y=k)}{\sum_{\ell}\mathbb{P}(X=x|Y=\ell)}$$

In generative model, we assume knowing:

- the probability density function $f_k(X)$ of the data $X$ for each class $k$
- the prior distribution $\pi_k$ of how many observations belong to each class

We have then:

$$\mathbb{P}(Y=k|X=x)=\frac{\pi_k f_k(x)}{\sum_{\ell}\pi_\ell f_\ell(x)}$$

-- **Linear discriminant analysis**

In LDA, we assume that the data from each class follows a Gaussian distribution

$$f_k(x)=\mathcal{N}(\mu_k,C_k)$$

and that the **covariance** matrices of all classes **are the same**, i.e. $\forall k, C_k=C$

Note that if we want to know describe the **boundary** region between classes $k$ and $\ell$ we can write

$$\log\left(\frac{\mathbb{P}(Y=k|X=x)}{\mathbb{P}(Y=\ell|X=x)}\right)=\log\frac{f_k(x)}{f_\ell(x)}+\log\frac{\pi_k}{\pi_\ell}=0$$

and in the specific case of Gaussian distributions for each class, we can show that

$$\log\left(\frac{\mathbb{P}(Y=k|X=x)}{\mathbb{P}(Y=\ell|X=x)}\right)=(\mu_k-\mu_\ell)^T C^{-1}x+\log\frac{\pi_k}{\pi_\ell}-\frac{1}{2}(\mu_k+\mu_\ell)^T C^{-1}(\mu_k-\mu_\ell)=0$$

which shows that we have a **linear** boundary between the classes

In fact, we can define for each class a **discriminant function** (which is linear in $x$)

$$\delta_k(x)=\mu_k^T C^{-1}x+\log\pi_k-\frac{1}{2}\mu_k^T C^{-1}\mu_k$$

and the classifier will assign the class for which the discriminant function is the highest, i.e.

$$f(x)=\underset{k}{\operatorname{argmax}}\,\delta_k(x)$$

Note that for obtaining such functions, all we need is to be able to **estimate** $\hat{\pi}_k, \hat{\mu}_k$ and $\hat{C}$ from **data**.

> Show Figure with results for LDA (show that we get ellipses)

-- **Similar but different**

We have seen that for both logistic regression and LDA, we have a **linear** boundary between the classes.

Does that mean they are the same ? No, because these boundaries are estimated differently.

Remember that we have

$$\mathbb{P}(X, Y) = \mathbb{P}(Y|X) \times \mathbb{P}(X)$$

- In **logistic regression**, we estimate an approximation only to $\mathbb{P}(Y|X)$ leaving $\mathbb{P}(X)$ completely untouched. In other words, we make no assumptions whatsoever about $\mathbb{P}(X)$ and this information is not used in the estimation of the parameters.
- In **LDA**, we make an assumption about the form of $\mathbb{P}(X)$ which is directly related to $f_k$ and $\pi_k$ and this has an impact over the estimation of the parameters of the classifier. One can see this information as a form of **regularization**.
- It should be noted that in LDA we **can't use qualitative predictors**, since the base assumption is that the data $X$ on each class $k$ follows some Gaussian distribution

---

-- **Notes on cross-entropy for logistic regression**:

We want to **minimize** the KL divergence (which is a measure of discrepancy between pdf's) between

$$f_{Y|X}(y|X = x) = p(x)\delta(y - 1) + (1 - p(x))\delta(y)$$

and

$$\hat{f}_{Y|X}(y|X = x) = \hat{p}(x)\delta(y - 1) + (1 - \hat{p}(x))\delta(y - 1)$$

in average for **different choices of** $x$ so we write (you will learn this in information theory and other more advanced courses on the theory of statistics).

$$\mathcal{L} = \mathbb{E}_x\Big[\mathrm{KL}\big(f\|\hat{f}\big)\Big] = \mathbb{E}_x\left[\int \log \frac{f_{Y|X}(y \mid X = x)}{\hat{f}_{Y|X}(y \mid X = x)} f_{Y|X}(y \mid X = x)\, \mathrm{d}y\right]$$

which can be simplified to

$$\begin{aligned} \mathcal{L} &= -\mathbb{E}_x\Big[p(x) \log \hat{p}(x) + (1 - p(x)) \log\big(1 - \hat{p}(x)\big)\Big] \\ &\approx -\frac{1}{N}\sum_{i=1}^{N}\Big(p(x_i) \log \hat{p}(x_i) + (1 - p(x_i)) \log\big(1 - \hat{p}(x_i)\big)\Big) \end{aligned}$$

where the $x_i$ are coming from a dataset with $N$ samples.

Usually, we don't know the actual true values of $p(x_i)$ but just the label of a given $x_i$ so we approximate $p(x_i) \approx y_i$ and have the loss function as:

$$\mathcal{L} \approx -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \log \hat{p}(x_i) + (1 - y_i) \log\big(1 - \hat{p}(x_i)\big)\Big)$$

This is what people call a **cross-entropy loss function**.