Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

02 - Logistic Regression

François Pitié

Assistant Professor in Media Signal Processing
Department of Electronic & Electrical Engineering, Trinity College Dublin

[4C16/5C16] Deep Learning and its Applications — 2025/2026

Motivation

With **Linear Regression**, we looked at linear models, where the output of the problem was a **continuous** variable (eg. height, car price, temperature, ...).

Very often you need to design a **classifier** that can answer questions such as: what car type is it? is the person smiling? is a solar flare going to happen? In such problems the model depends on categorical variables.

Logistic Regression (David Cox, 1958), considers the case of a binary variable, where the outcome is 0/1 or true/false.

Despite its name, logistic regression is a model for classification, not regression.

There is a whole zoo of classifiers out there. Why are we covering logistic regression in particular?

Because logistic regression is the building block of Neural Nets.

Introductory Example

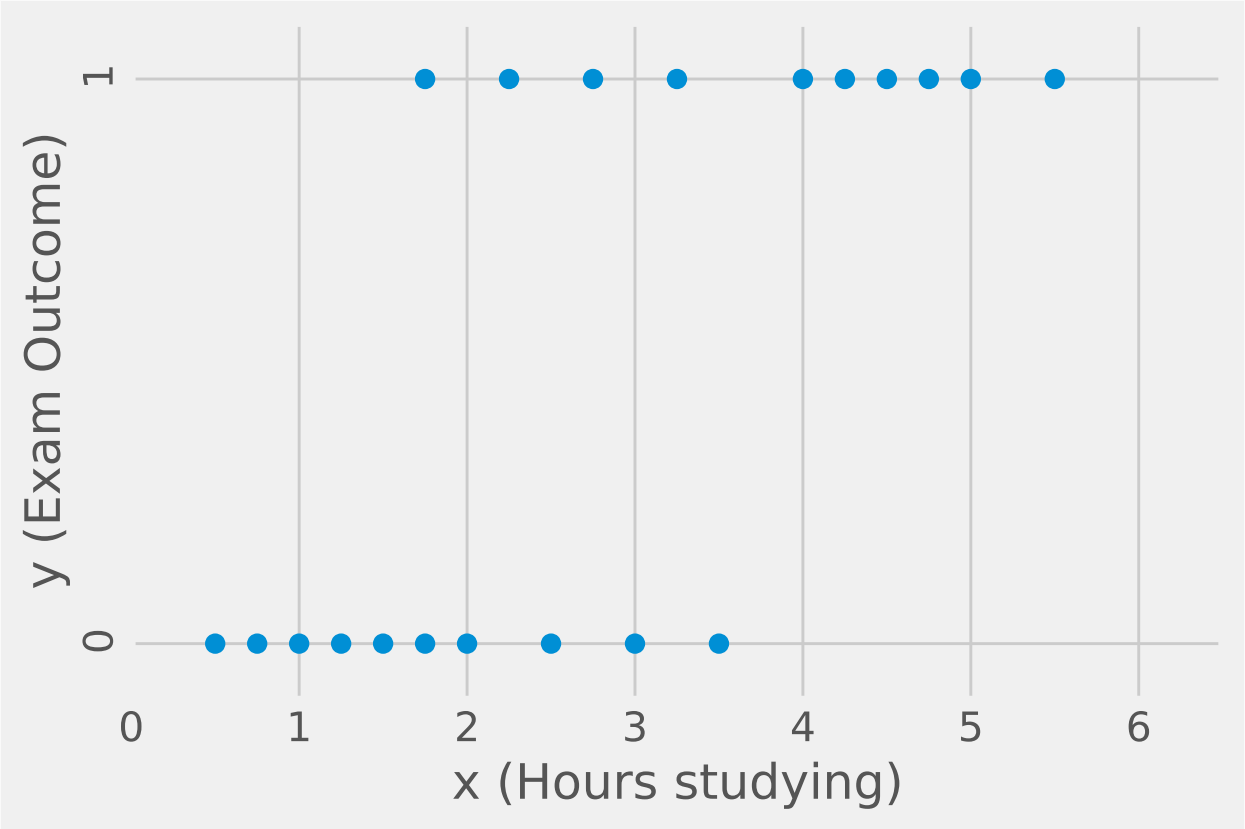
We'll start with an example taken from Wikipedia:

A group of 20 students spend between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability that the student will pass the exam?

Introductory Example

The collected data looks like so:

```
Studying Hours          : 0.75 1.00 2.75 3.50 ...
result (1=pass,0=fail) : 0    0    1    0    ...
```



Regression?

Although the output y is binary, we could still attempt to fit a linear model via least squares:

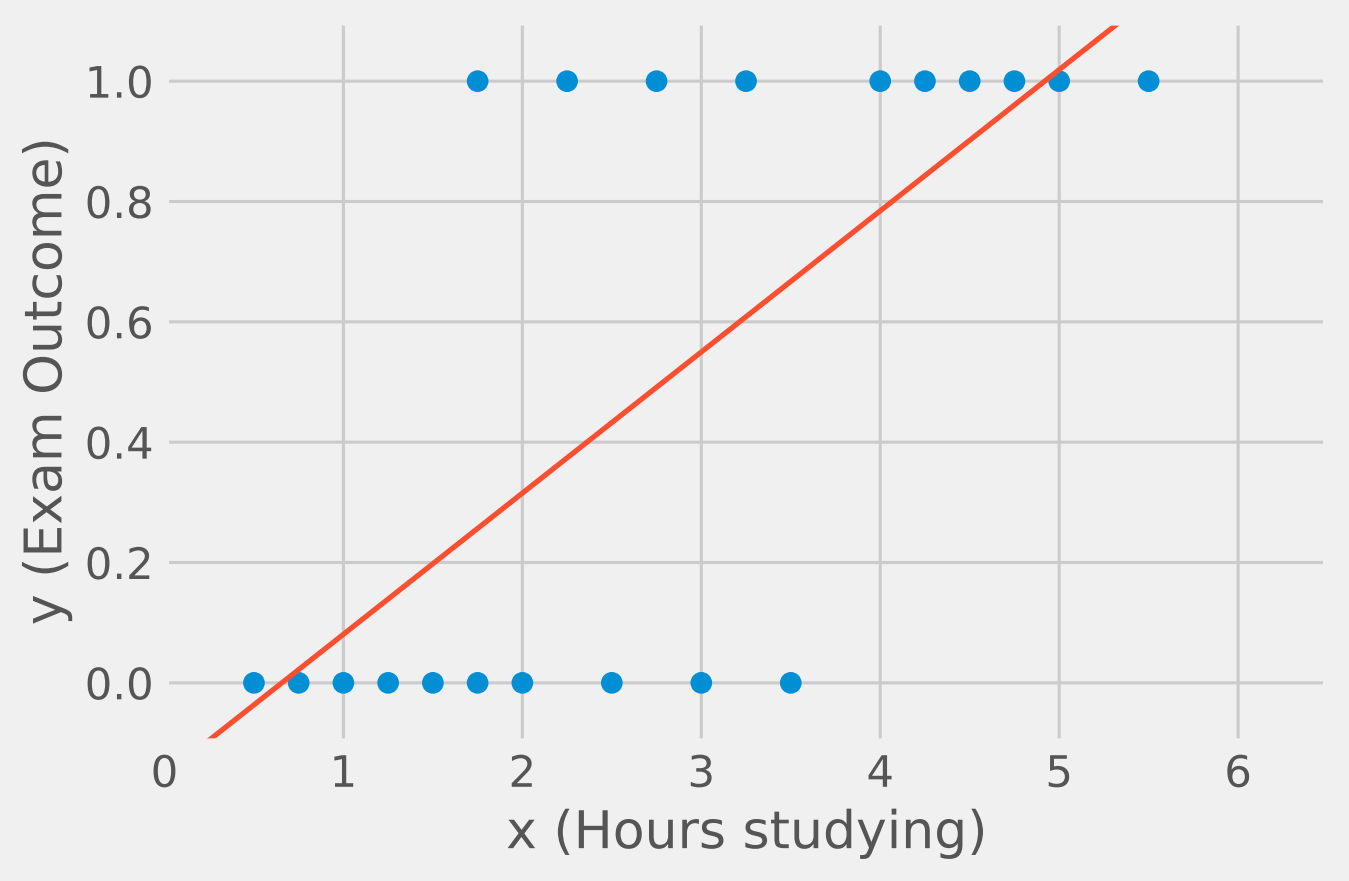
$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = w_1 x_1 + \cdots + w_p x_p$$

where $h_{\mathbf{w}}(\mathbf{x})$ is the *prediction* given model parameters $\mathbf{w}$ and input features $\mathbf{x}$.

Regression?

This is what the least squares estimate $h_{\mathbf{w}}(\mathbf{x})$ looks like:

$$h_w(x) \approx 0.18 \times x + 0.08$$



Regression?

The model prediction $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ is continuous, but we could apply a threshold to obtain the binary classifier as follows:

$$y = [\mathbf{x}^\top \mathbf{w} > 0.5] = \begin{cases} 0 & \text{if } \mathbf{x}^\top \mathbf{w} \leq 0.5 \\ 1 & \text{if } \mathbf{x}^\top \mathbf{w} > 0.5 \end{cases}$$

and the output would be 0 or 1.

On our example we would have:

$$y = \begin{cases} 0 & \text{if } 0.18 \times x + 0.08 \leq 0.5 \\ 1 & \text{if } 0.18 \times x + 0.08 > 0.5 \end{cases}$$

Obviously, we have some issues with that approach...

Regression?

Example: a student studied 100 hours and is successful:

$$h_w(x) = 100 \times 0.18 + 0.08 = 18.08 > 0.5$$

But in terms of LS, the error $\varepsilon^2 = (1 - h_w(x))^2 = 17.1^2$ is large, when everything is actually perfectly fine.

Regression?

The core issue is that with Least Squares, we are optimising the model to make the output $\mathbf{x}^\top \mathbf{w}$ match the target $y \in \{0, 1\}$.

However, what we should be optimising is our classification rule, for instance $[\mathbf{x}^\top \mathbf{w} > 0.5]$, to match the target y.

The MSE loss function heavily penalises correctly classified but distant points (e.g. a student who studies 10 hours and passes), pulling the decision boundary away from a good position.

We need a model designed for probabilities, not for direct value prediction.

General Linear Model

The general problem of general linear models can be presented as follows. We are trying to find a linear combination of the data $\mathbf{x}^\top \mathbf{w}$, such that the sign of $\mathbf{x}^\top \mathbf{w}$ tells us about the outcome y:

$$y = [\mathbf{x}^\top \mathbf{w} + \epsilon > 0]$$

General Linear Model

The general problem of general linear models can be presented as follows. We are trying to find a linear combination of the data $\mathbf{x}^\top \mathbf{w}$, such that the sign of $\mathbf{x}^\top \mathbf{w}$ tells us about the outcome y:

$$y = [\mathbf{x}^\top \mathbf{w} + \epsilon > 0]$$

The quantity $\mathbf{x}^\top \mathbf{w}$ is sometimes called the **risk score**. It is a scalar value that grades the certainty of belonging to one class of the other:

$$\begin{aligned} \mathbf{x}^\top \mathbf{w} \gg 0 \Rightarrow & \quad y = 1 \\ \mathbf{x}^\top \mathbf{w} \ll 0 \Rightarrow & \quad y = 0 \\ \mathbf{x}^\top \mathbf{w} \approx 0 \Rightarrow & \quad \text{undecided} \end{aligned}$$

The risk score does a dimensional reduction: it combines multiple input features into a single number.

General Linear Model

The general problem of general linear models can be presented as follows. We are trying to find a linear combination of the data $\mathbf{x}^\top \mathbf{w}$, such that the sign of $\mathbf{x}^\top \mathbf{w}$ tells us about the outcome y:

$$y = [\mathbf{x}^\top \mathbf{w} + \epsilon > 0]$$

The error term is represented by the random variable ϵ. Multiple choices are possible for the distribution of ϵ.

In logistic regression,
the error ϵ is assumed to follow a **logistic distribution**,
the risk score $\mathbf{x}^\mathsf{T}\mathbf{w}$ is called the logit.

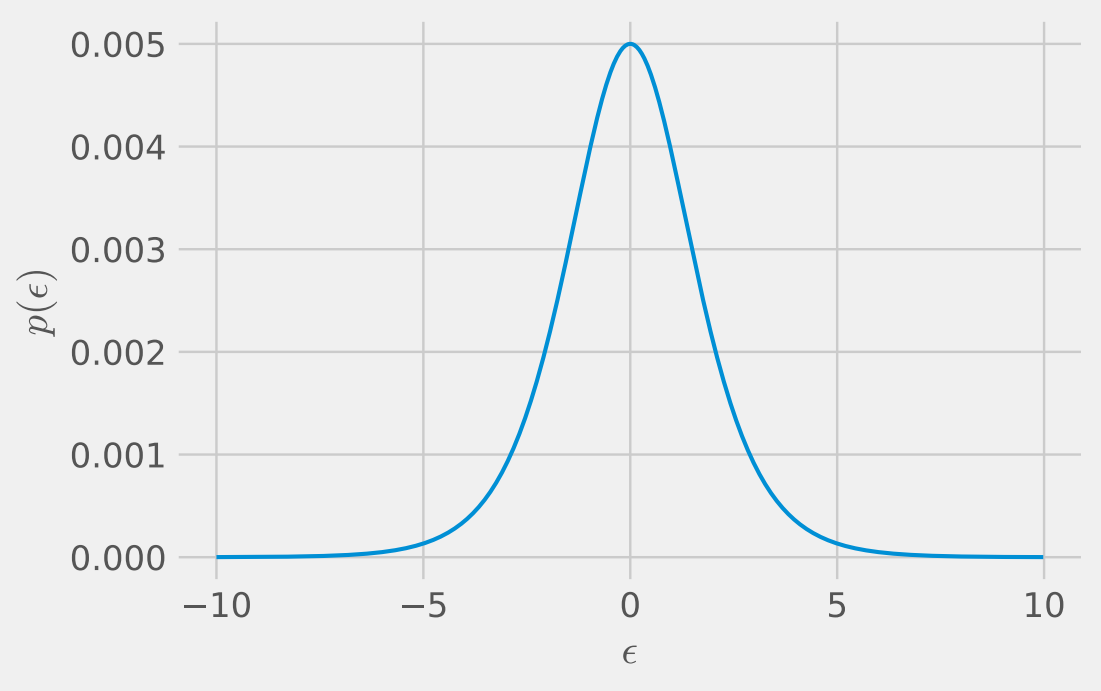


Figure: pdf of the logistic distribution

In probit regression,
the error ϵ is assumed to follow a **normal distribution**,
the risk score $\mathbf{x}^\mathsf{T}\mathbf{w}$ is called the probit.

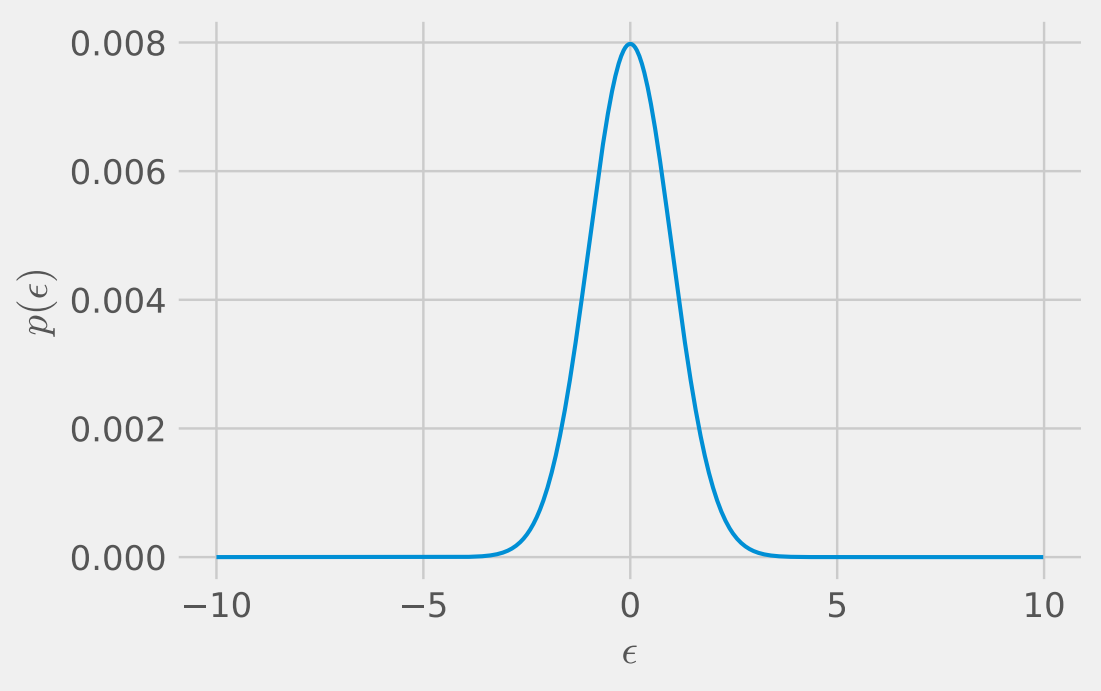


Figure: pdf of the normal distribution

For our purposes, there is not much difference between *logistic* and *logit* regression. The main difference is that logistic regression is numerically easier to solve.

From now on, we'll only look at the logistic model but note that similar derivations could be made for any other model.

Logistic Regression Model

Consider $p(y = 1|\mathbf{x}, \mathbf{w})$, the **likelihood** that the output is a success:

$$\begin{aligned} p(y = 1|\mathbf{x}, \mathbf{w}) &= p(\mathbf{x}^\top \mathbf{w} + \epsilon > 0) \\ &= p(\epsilon > -\mathbf{x}^\top \mathbf{w}) \end{aligned}$$

since ϵ is symmetrically distributed around 0, it follows that

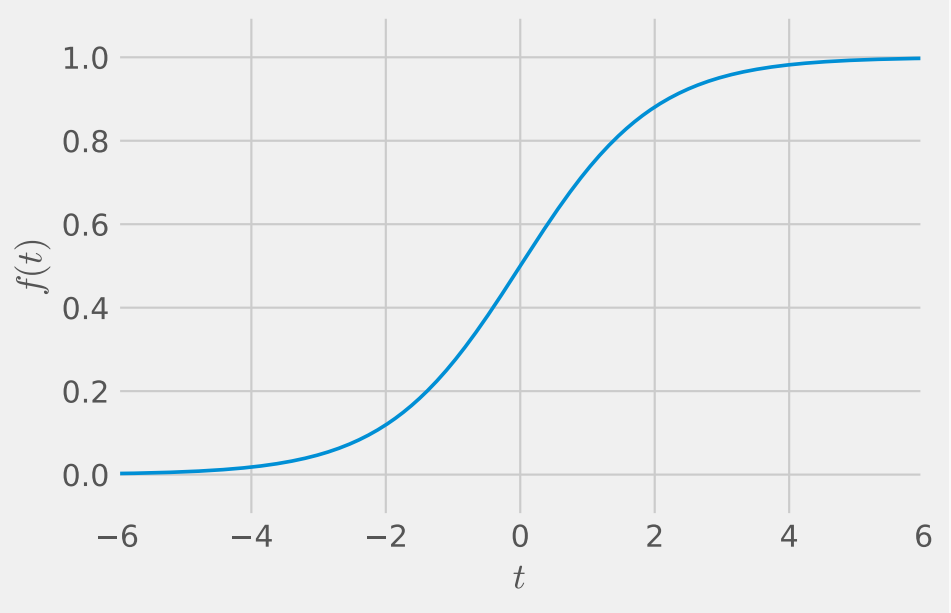
$$p(y = 1|\mathbf{x}, \mathbf{w}) = p(\epsilon < \mathbf{x}^\top \mathbf{w})$$

Because we have made some assumptions about the distribution of ϵ, we are able to derive a closed-form expression for the likelihood.

The Logistic Function

The function $f : t \mapsto f(t) = p(\epsilon < t)$ is the c.d.f. of the logistic distribution and is also called the **logistic function** or **sigmoid**:

$$f(t) = \frac{1}{1 + e^{-t}}$$



Logistic Regression Model

Thus we have a simple model for the likelihood of success:

$$p(y = 1|\mathbf{x}, \mathbf{w}) = p(\epsilon < \mathbf{x}^\top \mathbf{w}) = f(\mathbf{x}^\top \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{x}^\top \mathbf{w}}}$$

The likelihood of failure is simply given by:

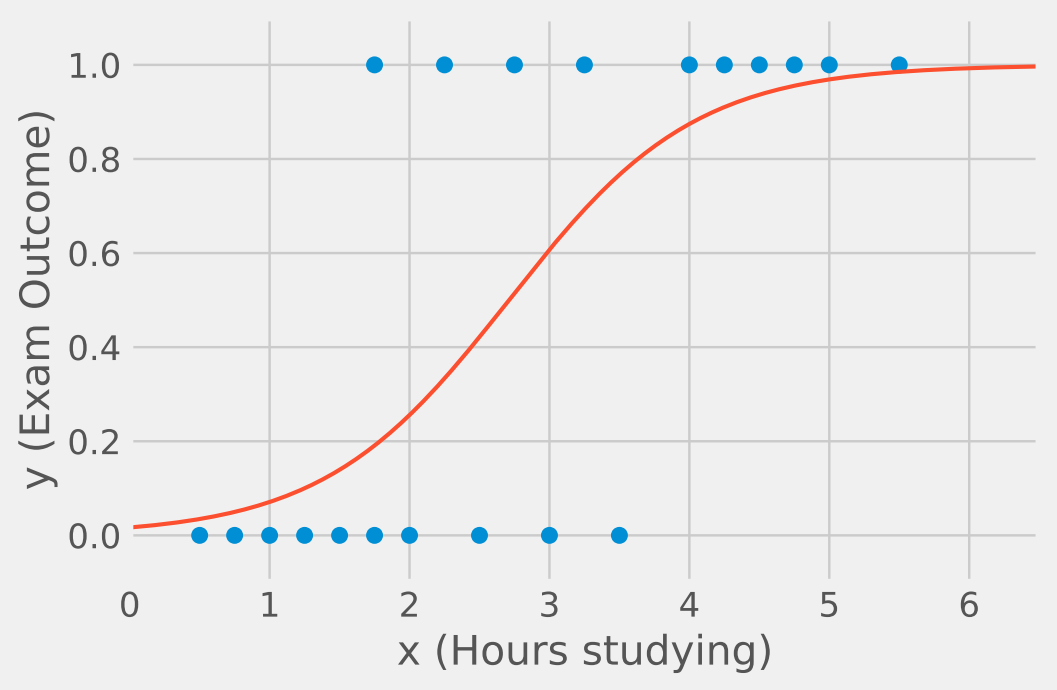
$$p(y = 0|\mathbf{x}, \mathbf{w}) = 1 - p(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{+\mathbf{x}^\top \mathbf{w}}}$$

Exercise:

show that $p(y = 0|\mathbf{x}, \mathbf{w}) = h_{\mathbf{w}}(-\mathbf{x})$

Logistic Regression Model

Below is the plot of $p(y = 1|\mathbf{x}, \mathbf{w}) = 1/(1 + \exp(-(w_0 + w_1 x)))$ for our problem (using optimal values of w_0 and w_1):



The results are easy to interpret: there is about 60% chance to pass the exam if you study for 3 hours.

Logistic Regression vs. Least Squares

In **linear regression**, the model prediction $h_{\mathbf{w}}(\mathbf{x})$ was a direct prediction of the outcome:

$$h_{\mathbf{w}}(\mathbf{x}) = \hat{y}$$

In **logistic regression**, the model prediction $h_{\mathbf{w}}(\mathbf{x})$ is an estimate of the **likelihood** of the outcome:

$$h_{\mathbf{w}}(\mathbf{x}) = p(y = 1|\mathbf{x}, \mathbf{w})$$

Whereas in linear regression, we try to answer the question:

What is the expected value of y given $\mathbf{x}$?

In logistic regression (and any other general linear model), we try instead to answer the question:

What is the probability that $y = 1$ given $\mathbf{x}$?

Maximum Likelihood

To estimate the weights $\mathbf{w}$, we will again use the concept of **Maximum Likelihood**.

Maximum Likelihood

As we've just seen, for a particular observation $\mathbf{x}_i$ and model $\mathbf{w}$, the likelihood is given by:

$$p(y = y_i|\mathbf{x}_i, \mathbf{w}) = \begin{cases} p(y = 1|\mathbf{x}_i, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}_i) & \text{if } y_i = 1 \\ p(y = 0|\mathbf{x}_i, \mathbf{w}) = 1 - h_{\mathbf{w}}(\mathbf{x}_i) & \text{if } y_i = 0 \end{cases}$$

As $y_i \in \{0, 1\}$, this can be written in a slightly more compact form:

$$p(y = y_i|\mathbf{x}_i, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}_i)^{y_i}(1 - h_{\mathbf{w}}(\mathbf{x}_i))^{1-y_i}$$

This works because $t^0 = 1$ for $t \in \mathbb{R}^*$.

Assuming independent observations, the likelihood over all observations is:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n} h_{\mathbf{w}}\left(\mathbf{x}_i\right)^{y_i}(1 - h_{\mathbf{w}}(\mathbf{x}_i))^{1-y_i}$$

Maximum Likelihood

We want to find $\mathbf{w}$ that maximises the likelihood $p(\mathbf{y}|\mathbf{X})$. It is again equivalent, but more convenient, to instead minimise the negative log likelihood:

$$\begin{aligned} E(\mathbf{w}) &= -\ln(p(\mathbf{y}|\mathbf{X}, \mathbf{w})) \\ &= \sum_{i=1}^{n} -y_i \ln\left(h_{\mathbf{w}}(\mathbf{x}_i)\right) - (1 - y_i) \ln\left(1 - h_{\mathbf{w}}(\mathbf{x}_i)\right) \end{aligned}$$

This loss function that we need to minimise is called the cross-entropy.

Note that we can also consider the *average* cross-entropy:

$$E(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} -y_i \ln\left(h_{\mathbf{w}}(\mathbf{x}_i)\right) - (1 - y_i) \ln\left(1 - h_{\mathbf{w}}(\mathbf{x}_i)\right)$$

We could have considered optimising the parameters $\mathbf{w}$ using other loss functions. For instance we could have tried to minimise the least square error as we did in linear regression:

$$E_{LS}(\mathbf{w}) = \sum_{i=1}^{n} \left(h_{\mathbf{w}}(\mathbf{x}_i) - y_i\right)^2$$

The solution would not maximise the likelihood, as would the cross-entropy loss, but maybe that would still be a reasonable thing to do? The problem is that $h_{\mathbf{w}}$ is a non-convex sigmoid, which makes the minimisation of $E_{LS}(\mathbf{w})$ much harder than with the cross-entropy.

This is in fact a mistake that the Neural Net community did for a number of years before switching to the cross entropy loss function.

Optimisation: gradient descent

The minimisation of the loss function can be done through the use of the **gradient descent** algorithm, which is a general method for non-linear optimisation and which will be at the core of neural networks optimisation.

We start at $\mathbf{w}^{(0)}$ and take steps along the steepest direction $\mathbf{v}$ using a fixed size step as follows:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \eta \mathbf{v}^{(n)}$$

η is called the learning rate and controls the speed of the descent.

What is the steepest slope $\mathbf{v}$?

Optimisation: gradient descent

Without loss of generality we set $\mathbf{v}$ to be a unit vector (ie. $\|\mathbf{v}\| = 1$). Then, moving $\mathbf{w}$ to $\mathbf{w} + \eta\mathbf{v}$ yields a new error as follows:

$$E(\mathbf{w} + \eta\mathbf{v}) = E(\mathbf{w}) + \eta \left(\frac{\partial E}{\partial \mathbf{w}}\right)^{\mathsf{T}} \mathbf{v} + O(\eta^2)$$

which reaches a minimum when

$$\mathbf{v} = -\frac{\frac{\partial E}{\partial \mathbf{w}}}{\|\frac{\partial E}{\partial \mathbf{w}}\|}$$

Optimisation: gradient descent

now, it is hard to find a good value for the learning rate η and we usually adopt an adaptive step instead. Thus instead of using

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} - \eta \frac{\frac{\partial E}{\partial \mathbf{w}}}{\| \frac{\partial E}{\partial \mathbf{w}} \|}$$

we usually use the following update step:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} - \eta \frac{\partial E}{\partial \mathbf{w}}$$

Optimisation: gradient descent

Recall that the cross-entropy loss function is:

$$E(\mathbf{w}) = \sum_{i=1}^{n} -y_i \ln(h_{\mathbf{w}}(\mathbf{x}_i)) - (1 - y_i) \ln(1 - h_{\mathbf{w}}(\mathbf{x}_i))$$

and that $h_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{x}^\top \mathbf{w}) = \dfrac{1}{1 + e^{-\mathbf{x}^\top \mathbf{w}}}$

Exercise:

Given that the derivative of the sigmoid f is $f'(t) = (1 - f(t))f(t)$, show that

$$\frac{\partial E}{\partial \mathbf{w}} = \sum_{i=1}^{n} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\, \mathbf{x}_i$$

Optimisation: gradient descent

The overall gradient descent method looks like so:

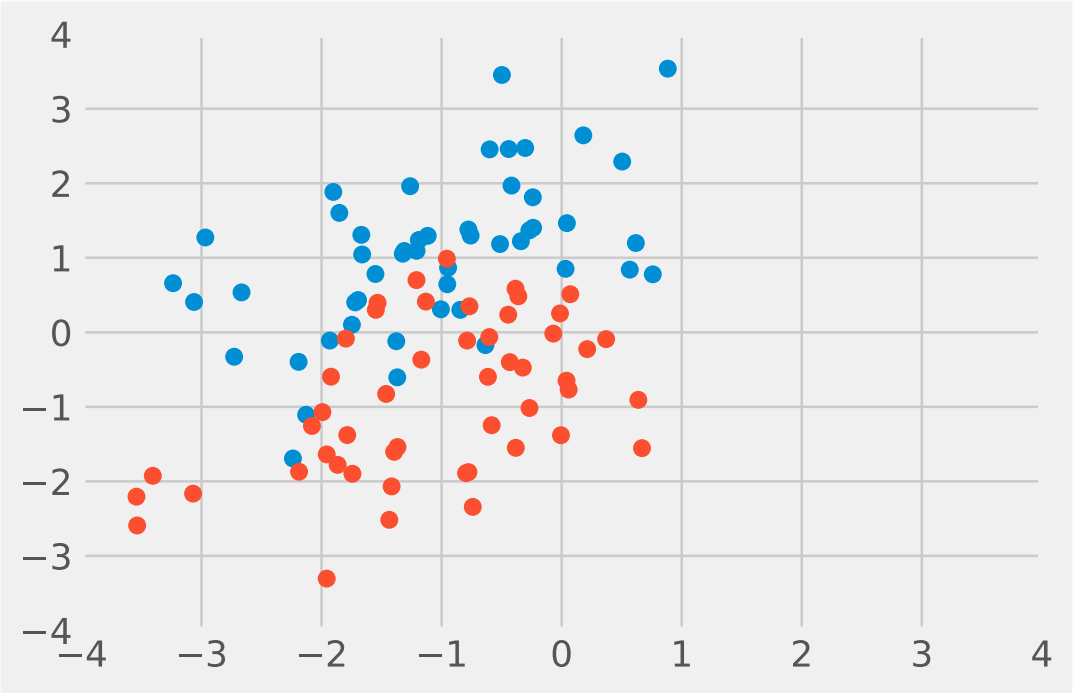
1. set an initial weight vector $\mathbf{w}^{(0)}$ and
2. **for** $t = 0, 1, 2, \cdots$ **do until convergence**
3. compute the gradient

$$\frac{\partial E}{\partial \mathbf{w}} = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-\mathbf{x}_i^\top \mathbf{w}}} - y_i \right) \mathbf{x}_i$$

4. update the weights: $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E}{\partial \mathbf{w}}$

Example

Below is an example with 2 features.

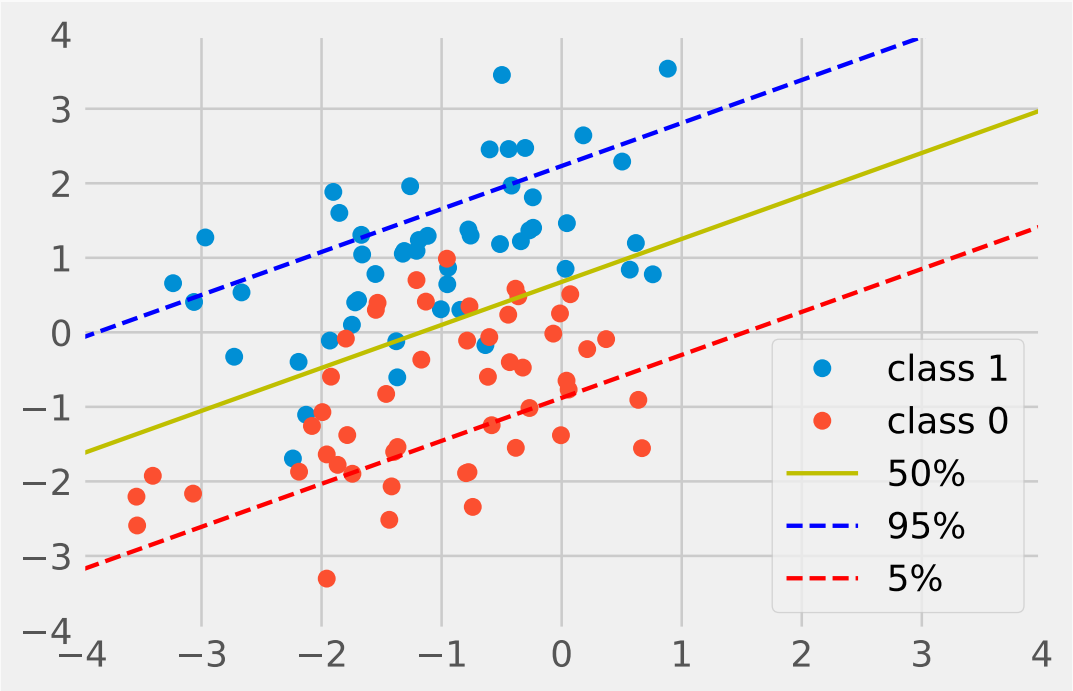


Example

The estimate for the probability of success is

$$h_{\mathbf{w}}(\mathbf{x}) = 1/(1 + e^{-(-1.28-1.09x_1+1.89x_2)})$$

Below are drawn the lines that correspond to $h_{\mathbf{w}}(\mathbf{x}) = 0.05$, $h_{\mathbf{w}}(\mathbf{x}) = 0.5$ and $h_{\mathbf{w}}(\mathbf{x}) = 0.95$.



Multiclass Classification

In many applications you have to deal with more than 2 classes.

In these cases, we need to use multinomial logistic regression, which is an extension of logistic regression to more than 2 classes.

Multinomial Logistic Regression

In Multinomial Logistic Regression, each of the binary classifier is based on the following likelihood model:

$$p(y = C_k|\mathbf{x}, \mathbf{w}) = \mathrm{softmax}(\mathbf{x}^\top \mathbf{w})_k = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

C_k is the class k and $\mathrm{softmax} : \mathbb{R}^K \to \mathbb{R}^K$ is the function defined as

$$\mathrm{softmax}(\mathbf{t})_k = \frac{\exp(t_k)}{\sum_{j=1}^{K} \exp(t_j)}$$

In other words, $\mathrm{softmax}$ takes as an input the vector of logits for all classes and returns the vector of corresponding likelihoods.

Multinomial Logistic Regression

For instance, say we have 3 classes A, B, C. For each class we compute a specific score $\mathbf{x}^\top \mathbf{w}_A$, $\mathbf{x}^\top \mathbf{w}_B$, $\mathbf{x}^\top \mathbf{w}_C$. These 3 scores are then turned into 3 probabilities that sum up to 1 using softmax:

$$\mathbf{x} \rightarrow \begin{bmatrix} \mathbf{x}^\top \mathbf{w}_A = -1.2 \\ \mathbf{x}^\top \mathbf{w}_B = +3.1 \\ \mathbf{x}^\top \mathbf{w}_C = -0.9 \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} p(A|\mathbf{x}) = 0.0131 \\ p(B|\mathbf{x}) = 0.9691 \\ p(C|\mathbf{x}) = 0.0177 \end{bmatrix}$$

where $p(A|\mathbf{x}) = \exp(-1.2)/(\exp(-1.2) + \exp(3.1) + \exp(-0.9)) = 0.0131$

Multinomial Cross Entropy

To optimise for the parameters. We can take again the **maximum likelihood** approach.

Combining the likelihood for all possible classes gives us:

$$p(y|\mathbf{x}) = p(y = C_1|\mathbf{x})^{[y=C_1]} \times \cdots \times p(y = C_K|\mathbf{x})^{[y=C_K]}$$

where $[y = C_1]$ is 1 if $y = C_1$ and 0 otherwise.

The total likelihood is:

$$p(y|\mathbf{X}) = \prod_{i=1}^{n} p(y_i = C_1|\mathbf{x}_i)^{[y=C_1]} \times \cdots \times p(y_i = C_K|\mathbf{x}_i)^{[y=C_K]}$$

Multinomial Cross Entropy

Taking the negative log likelihood yields the cross entropy error function for the multiclass problem:

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\ln(p(y|\mathbf{X})) = -\sum_{i=1}^{n}\sum_{k=1}^{K}[y_i = C_k]\ln(p(y_i = C_k|\mathbf{x}_i))$$

Note that this is still the same cross entropy as presented in the binary classification. It may be convenient to remind ourselves that we are working with 2 or more classes, therefore this form is sometimes referred to as the **multinomial cross entropy** and, the one we've seen for the two-class problem as the **binary cross entropy**.

Similarly to logistic regression, we can use a gradient descent approach to find the K weight vectors $\mathbf{w}_1, \cdots, \mathbf{w}_K$ that minimise this cross entropy expression.

Take Away

With **Logistic Regression**, we look at linear models, where the output of the problem is a **binary categorical** response.

Instead of directly predicting the actual outcome as in least squares, the model proposed in logistic regression makes a prediction about the **likelihood of belonging to a particular class**.

Finding the maximum likelihood parameters is equivalent to minimising the **cross entropy** loss function. The minimisation can be done using the **gradient descent** technique.

The extension of Logistic Regression to more than 2 classes is called the **Multinomial Logistic Regression**.