02 - Logistic Regression

François Pitié

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

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. That is, the outcome is 0/1 or true/false.

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

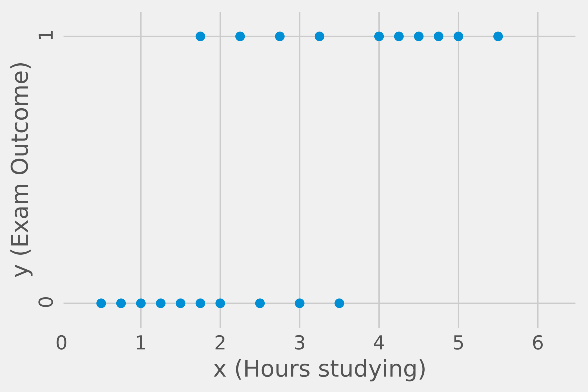
We’ll start with an example from Wikipedia:

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    ...
```



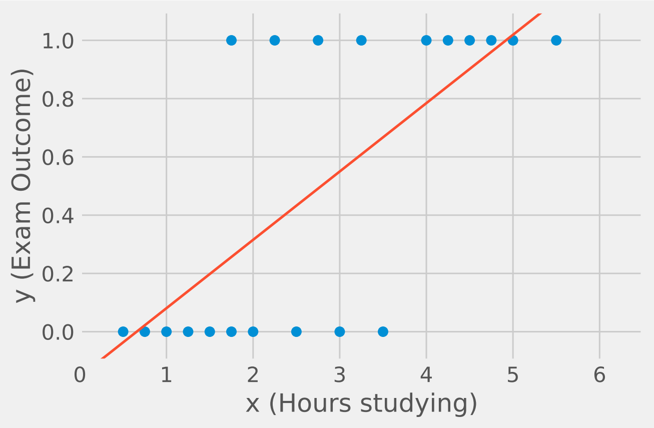
Linear Approximation

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

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = w_0 + w_1 x_1 + \cdots$$

Linear Approximation

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



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.

Obviously this is not optimal as we have optimised $\mathbf{w}$ so that $\mathbf{x}^\top \mathbf{w}$ matches y and not so that $[\mathbf{w}^\top \mathbf{x} > 0.5]$ matches y.

Let's see how this can be done.

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 $\mathbf{x}^\intercal\mathbf{w}$, such that the sign of $\mathbf{x}^\intercal\mathbf{w}$ tells us about the outcome y:

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

The quantity $\mathbf{x}^\intercal\mathbf{w}$ is sometimes called the **risk score**. It is a scalar value. The larger the value of $\mathbf{x}^\intercal\mathbf{w}$ is, the more certain we are that $y = 1$.

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** and the risk score $\mathbf{x}^\top \mathbf{y}$ is also called the logit.

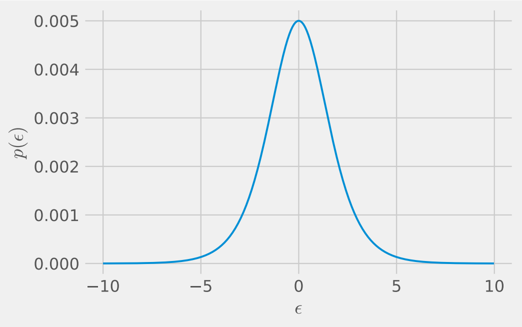


Figure: pdf of the logistic distribution

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

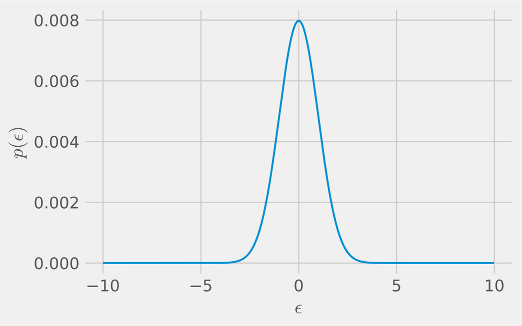


Figure: pdf of the normal distribution

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.

Logistic Model

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

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

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

since ϵ is symmetrically distributed around 0, it follows that

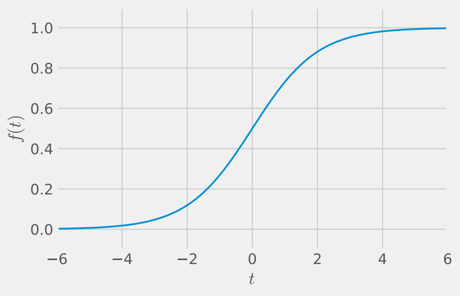
$$p(y = 1|\mathbf{x}) = p(\epsilon < \mathbf{x}^\top \mathbf{w})$$

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**:

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



Thus we have a simple model for the likelihood of success $h_{\mathbf{w}}(\mathbf{x}) = p(y = 1|\mathbf{x})$:

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

The likelihood of failure is simply given by:

$$p(y = 0|\mathbf{x}) = 1 - h_{\mathbf{w}}(\mathbf{x})$$

Exercise:

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

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

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

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

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

Thus whereas in linear regression we try to answer the question:

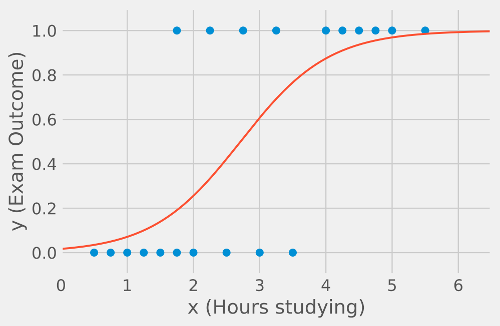
What is the value of y given $\mathbf{x}$?

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

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

Logistic Regression

Below is the plot of an estimated $h_{\mathbf{w}}(\mathbf{x}) \approx p(y = 1|\mathbf{x})$ for our problem:



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

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 $\mathbf{x}_i$, the likelihood is given by:

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

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

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

This works because $z^0 = 1$.

The likelihood over all observations is then:

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

Maximum Likelihood

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

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

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

In the Machine Learning community, the error function is also frequently called a **loss function**. Thus here we would say: the loss function is the cross-entropy.

We could have considered optimising the parameters $\mathbf{w}$ using other error 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 problem is that $h_{\mathbf{w}}$ is non-convex and makes the minimisation of $E_{LS}(\mathbf{w})$ much harder than when using cross-entropy.

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

Optimisation: gradient descent

To minimise the error function, we need to resort to **gradient descent**, which is a general method for nonlinear 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 set the right size for a fixed learning rate η.

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 repeat 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 error function to minimise is:

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

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

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:

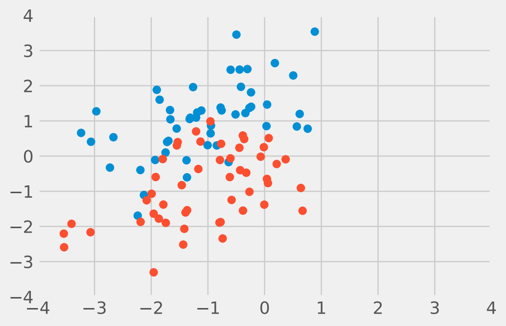
1. set an initial weight vector $\mathbf{w}^{(0)}$ and
2. **for** $t = 0, 1, 2, \cdots$ **do until convergence**
3. compute the gradient

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

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.

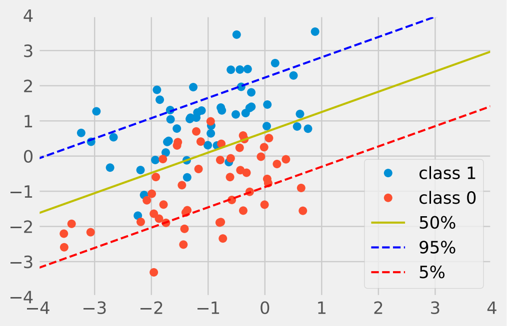


Example

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

It is very often that you have to deal with more than 2 classes.

one-vs.-all

The simplest way to consider a problem that has more than 2 classes is to adopt the one-vs.-all (or one-against-all) strategy:

For each class k, you can train a single binary classifier ($y = 0$ for all other class, and $y = 1$ for class k). The classifiers return a real-valued likelihood for their decision.

The one-vs.-all prediction returns the label for which the corresponding classifier reports the highest likelihood.

The **one-vs.-all** approach is a very simple one. However it is an heuristic that has many problems.

One problem is that for each binary classifier, the negative samples (from all the classes but k) are more numerous and more heterogeneous than the positive samples (from class k).

A better approach would be to have a unified model for all classifiers and jointly train them. The extension of Logistic regression that just does this is called multinomial logistic regression.

Multinomial Logistic Regression

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

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

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.

Softmax Optimisation

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

The total likelihood is:

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

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

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

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**.