


Classification with logistic regression

Carl McBride Ellis

 `carl.mcbride@u-tad.com`

In regression we calculate a real value (\hat{y}) for each point.

In classification we assign a label to each point. Each label is a value from a set of classes, for example, in binary classification the classes are usually 0 or 1 *i.e.* $y \in \{0, 1\}$

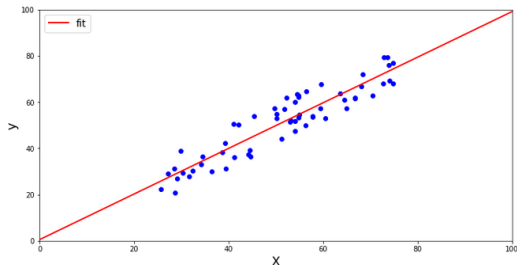
We do not always want classes, but rather probabilities *i.e.* although both belong to the class “1”, $0.51 \neq 0.99$

(scikit methods: `predict` y [predict_proba](#))

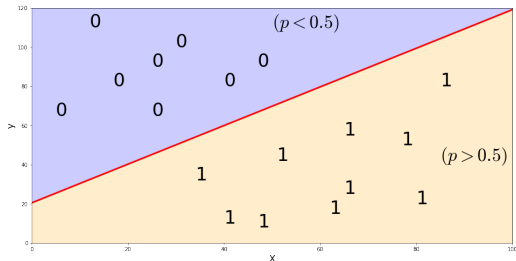
(Note: By convention the majority class is 0, and is sometimes called the '*negative class*')

(Advanced material: The probabilities are not necessarily well calibrated. See: [“Probability calibration”](#))

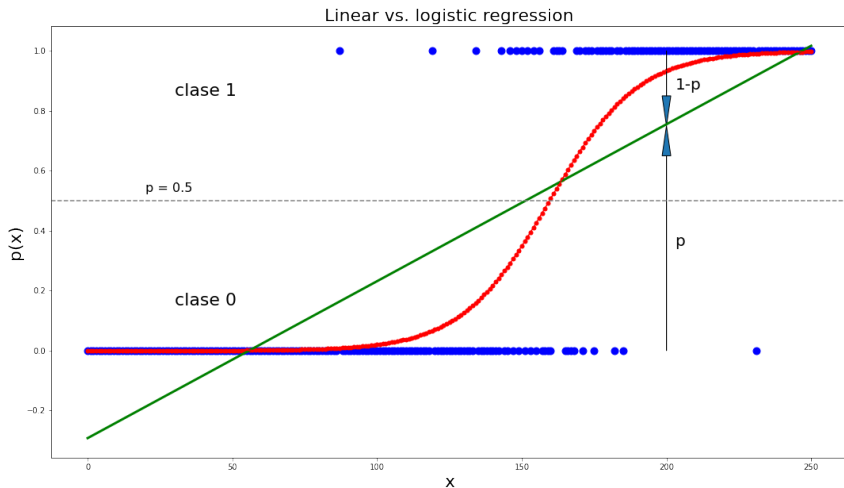
In regression we wish to find the line closest to all of the points



On the other hand in classification we wish to find the line that best divides the classes, called the “*decision boundary*”



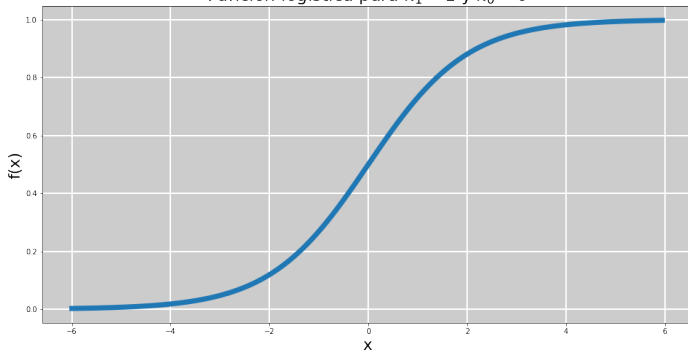
Logistic regression univariate example:



Logistic function: $\mathbb{R} \rightarrow (0, 1)$

$$\text{probabilities}(p) \leftarrow \text{logistic}(x) := \frac{1}{1 + e^{-(\kappa_1 x + \kappa_0)}}$$

Función logística para $\kappa_1 = 1$ y $\kappa_0 = 0$



The \hat{y} that we are now calculating are probabilities (p)

In linear regression our task was to find the parameters β_1 and β_0 of the straight line that best fit the data.

$$\hat{y}(x) = \beta_1 x + \beta_0$$

In logistic regression our task is to find the parameters κ_1 and κ_0 of the logistic function that best fits the data.

$$\ln \left(\frac{\hat{y}}{1-\hat{y}} \right) = \kappa_1 x + \kappa_0$$

(the *log-odds* is known as the *logit link function* (g))

In classification we calculate the probability of belonging to a class
Notes:

- probability is multiplicative
- $p(d_1 \text{ and } d_2) = p(d_1) \cdot p(d_2)$
- for binary classification
- $p(class = 1) = 1 - p(class = 0)$
- properties of logarithms
- $\ln(p_1 \cdot p_2) = \ln(p_1) + \ln(p_2) = \sum_i \ln(p_i)$
- logarithms in the domain (0,1) have a negative range

classification loss function = **log loss** or “*cross-entropy*” loss

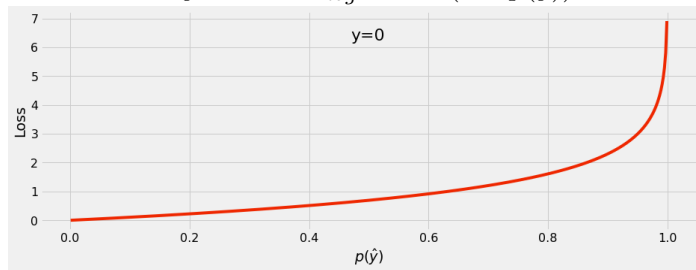
- $L_{log} = -y \ln(p(\hat{y})) - (1 - y) \ln(1 - p(\hat{y}))$
- $J_{log} = -\frac{1}{N} \sum (y \ln(p(\hat{y})) + (1 - y) \ln(1 - p(\hat{y})))$

where $p(\hat{y})$ is the probability of belonging to class 1,
thus $(1 - p(\hat{y}))$ is the probability of belonging to class 0.

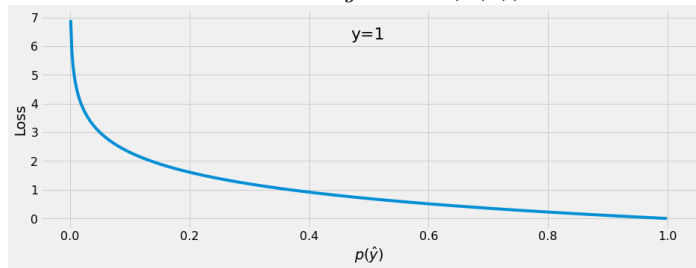
This will give us a convex cost function, so we should always be able to find the global minima.

What does the loss \mathcal{L} look like?

If the class of y is 0 the $\mathcal{L}_{log} = -\ln(1 - p(\hat{y}))$



If the class of y is 1 the $\mathcal{L}_{log} = -\ln(p(\hat{y}))$



Explainability

Logistic regression, just like its homologue linear regression, has the advantage that their predictions, although not necessarily the best, are relatively easy to understand, for an example see the notebook [Titanic explainability: Why me? asks Miss Doyle](#)

```
from sklearn.linear_model import LogisticRegression
```

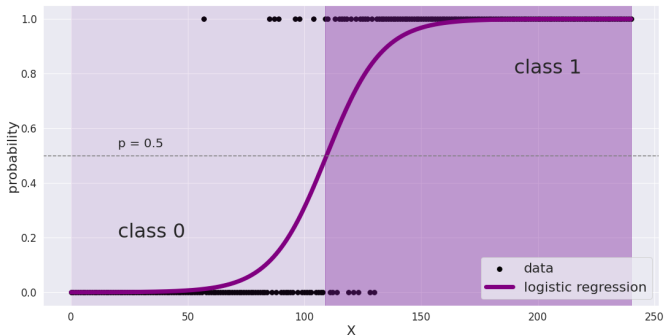
Practical session

Use scikit-learn to perform logistic regression

`notebook_ML_classification_logistic.ipynb`

Dichotomized logistic regression

Our logistic regression provides probabilities. To create classes we use a *decision threshold* (the default choice is $p = 0.5$)



WARNING: Dichotomization destroys extremely valuable information:
 $p = 0.51 \neq p = 0.99$ but both end up being class=1

(notebook "False positives, false negatives and the discrimination threshold")