# Logistic Regression

## Binary Classification Problem

**Dataset.** Consider a collection of features $\mathbf{X} = \{X_1, \ldots, X_d\}$, such that $dom(X_i) \subseteq \mathbb{R}$ for all $i = 1 \ldots d$. These are our *independent variables*.

Consider also an additional variable $Y$, such that $dom(Y) = \{0, 1\}$ or $dom(Y) = \{-1, +1\}$. This is our *binary dependent variable*.

Let $X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ be a collection of *data points*, such that $(\forall j \in 1 \ldots n)(\mathbf{x_j} \in \mathbb{R}^d)$. Let $\mathbf{y} = \{y_1, \ldots, y_n\}$ such that $(\forall j \in 1 \ldots n)(y_j \in dom(Y))$. We write $X$ as

$$\mathbf{X} = \left( \begin{array}{cccc} X_1 & X_2 & \ldots & X_d \\ \hline x_{11} & x_{12} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nd} \end{array} \right)$$

We also write $\mathbf{x_i} = (x_{i1}, \ldots, x_{id})$.

The *binary classification problem* can be specified as follows:

> Build a function $f : \mathbb{R}^d \longrightarrow dom(Y)$ that predicts the binary label of a data point $\curvearrowleft \in \mathbb{R}^d$.

**Dependent Variable.** In classification scenarios, the dependent variable $Y$ is typically considered to be *categorical*. Many classification methods, in order to allow for the use of mathematical functions to represent classification decisions, assume that $Y$ takes *numeric values*. For *binary classification problems*, some methods take advantage of treating values of $Y$ as 0 and 1, while other methods (primarily those structured around *separating planes*) take advantage of treating values of $Y$ as $-1$ and $+1$. In what follows, we will treat levels of the dependent variable $Y$ (i.e., the class labels) as whatever values that suit the best the method we are studying.

If $dom(Y) = \{v_1, v_2\}$, we sometimes use abbreviations $\mathbf{X}_{v_1}$ and $X_{v_2}$ to represent all data points belonging to classes $v_1$ and $v_2$ respectively.

## Logistic Regression

Let us assume that $dom(Y) = \{0, 1\}$.

In this case:

- Our independent variables are numeric
- Out dependent variable is represented in terms of a numeric value, so we can treat it as numeric.

Therefore, *we could attempt linear regression* as the method of predicting $y = f(\mathbf{x})$ given the training set $\langle X, Y \rangle$. However... linear regression predictors of the form

$$L(\beta) = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$$

**have unbounded values** (see Figure 1), and therefore for a fixed set of parameters $\beta$, as values of individual $x_i$s grow, the value $L_\beta(\mathbf{x})$ will grow.

We could create a predictor as follows (see Figure 2:

$$F(\mathbf{x}) = \left\{ \begin{array}{ll} 0, & \text{if} L_\beta(\mathbf{x}) \leq 0.5 \\ 1, & \text{otherwise} \end{array} \right.$$

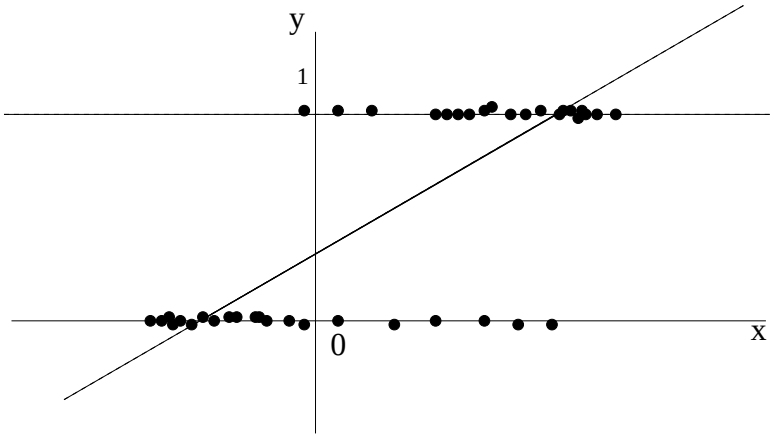


Figure 1: Linear Regression fails to properly predict the dependent variable values as depednent value grows in absolute value.

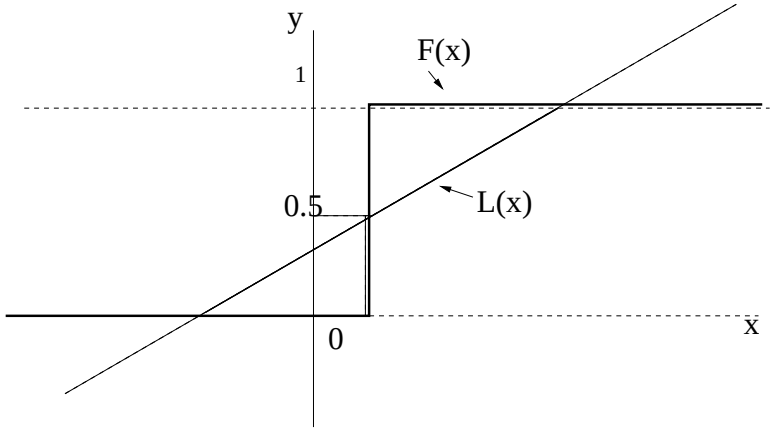


Figure 2: Building a 0-1 predictor out of linear regression predictor.

**Intuition.** The $F()$ predictor looks reasonable. It is a step function that jumps from the value of 0 to the value 1 (remember, these are class labels) at some boundary point $\mathbf{x}$, such that our linear regressor $L_\beta(\mathbf{x}) = 0.5$ (in multidimensional space, the set of $\mathbf{x}$, such that $L_\beta(\mathbf{x}) = 0.5$ forms a subspace that serves as a separating boundary).

The boundary points are determined by the training data $X, Y$ (which in turn determine the coefficients $\beta$ for the linear regressor).

In such a setting the linear regression function $L()$ is **no longer a predictor** of the class $Y$. This role is played by $F()$. What **does** $L()$ predict, though?

It is *tempting* to think that $L()$ is predicting *some sort of probability*, given that we break at $L()$ taking the value 0.5. However, because $L()$ is unbounded, while probabilities range from 0 to 1, this is clearly not the case.

**Odds and Log-odds.** While probability, the value $P(Y = 1)$ ranges from 0 to 1, another quantity related to probabilities, namely, the odds ratio:

$$Odds = \frac{P(Y = 1)}{1 - P(Y = 1)}$$

ranges from 0 to $+\infty$.

So... (drum roll) ...

we could choose for our linear predictor $L(\beta) = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$ to *attempt to predict the odds*. This is still not *quite what we want* as $L()$ ranges from $-\infty$ to $+\infty$.

However... (drum roll again)...

The quantity known as **log odds** (or log-odds ratio):

$$LogOdds = \ln \frac{P(Y = 1)}{1 - P(Y = 1)}$$

**does have the range**

$$[-\infty, +\infty]$$

As such, we could, as the basis for our prediction procedure, take the following designation:

$$\ln \frac{P(Y = 1}{1 - P(Y = 1)} = L(\beta)(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$$

(Parenthetically, note that $P(Y_i = 1) \geq 0.5$ is our boundary condition for assigning a data point $\mathbf{x_i}$ to class 1.)

**Finding the solution.** Let's remove the logarithm from the left hand side.

$$\frac{P(Y = 1)}{1 - P(Y = 1)} = e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d}$$

Simplifying:

$$P(Y = 1) = e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d} - P(Y = 1) e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d}$$

i.e.,

$$P(Y = 1)(1 + e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d}) = e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d}$$

or, in other words:

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d}}{(1 + e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d})} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d)}}$$

The function $f(x) = \frac{e^x}{1+e^x}$ is known as the logistic function, and thus, the probability that a specific data point belongs to class 1 is predicted by the logistic function applied to the linear predictor.

**Training the model.** To find the coefficients $\beta$ we need to build a cost function that we will optimize.

Recall, in our training data, given a data point $\mathbf{x_i} \in \mathbf{X}$ it's class label $Y_i$ takes one of the values: 0 or 1.

The value $\hat{p}_i = \frac{1}{1+e^{-(\beta_0+\beta_1 x_i 1+\ldots+\beta_d x_i d)}}$ ranges from 0 to 1.

Our "error" on a single data point can be expressed as follows:

$$Error(\beta, \mathbf{x_i}) = -Y_i \ln(\hat{p}_i) - (1 - Y_i) \ln(1 - \hat{p}_i)$$

This expression is known as Log loss.

The full cost function can be then expressed as

$$LL(\beta) = -\sum_{\mathbf{x_i} \in \mathbf{X}} Error(\beta, \mathbf{x_i}) = \sum_{\mathbf{x_i} \in \mathbf{X}} (Y_i \ln(\hat{p}_i) + (1 - Y_i) \ln(1 - \hat{p}_i)).$$

This value needs to be minimized.

**Gradient Descent.** Let us look at the gradient of $LL()$.

$$\frac{\partial Error(\beta)}{\partial \beta_j} = \frac{\partial Error}{\partial \hat{p}_i} \frac{\partial \hat{p}_i}{\partial \beta_j} = \left( -\frac{Y_i}{\hat{p}_i} + \frac{1 - Y_i}{1 - \hat{p}_i} \right) (\hat{p}_i (1 - \hat{p}_i \cdot x_{ij}) = (\hat{p}_i - Y_i) x_{ij}$$

Therefore,

$$\frac{\partial LL(\beta)}{\partial \beta_j} = \sum_{\mathbf{x_i} \in \mathbf{X}} (\hat{p}_i - Y_i) x_{ij}$$

Our gradient descent procedure therefore is:

**Step 1.** Pick learning rate $\eta > 0$.

**Step 2.** Pick $\beta^0 = (\beta_0^0, \ldots \beta_d^0)$

**Step 3.** Update repeatedly $\beta_j^{t+1} = \beta_j^t + \eta \sum_{\mathbf{x_i} \in \mathbf{X}} (Y_i - \hat{p}_i) x_{ij}$

**Step 4.** until convergence.