Linear Models for Classification Tasks

Ryan Miller

Grinnell College
Statistics

Shortcomings of Linear Regression

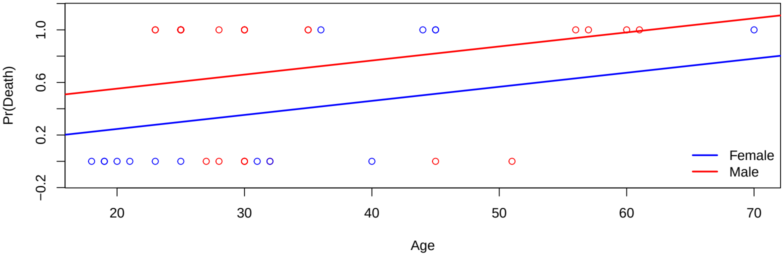
We previously introduced linear regression as a way to model the relationship between a set of predictors:

$$Y = w_o + w_1X_1 + w_2X_2 + \ldots + w_pX_p + \epsilon$$

- When Y is a binary variable, this model is problematic because its predicted values can fall outside of $[0, 1]$

Example (Donner Party Survival)

This model contains two predictors, “Age”, and “Sex” (expressed using one-hot encoding):



The model predicts males aged 60+ have more than a 100% probability of death, for males aged 70+ its 120%

Grinnell College
Statistics

Generalized Linear Models

- **Generalized Linear Models** provide the theoretical framework for adapting the basic structure of linear regression to classification tasks
  - To begin, linear regression can be viewed as the model:

$$Y \sim N(z, \sigma), \text{ where: } z = w_o + w_1 X_1 + w_2 X_2 + \ldots$$

- In this model, two components are clearly displayed:
  - A *linear predictor*, z (called a prediction score by data scientists)
  - A probability model that explains some of the variability in the outcome

Logistic Regression

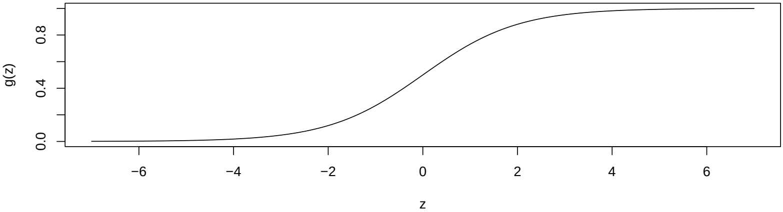
- The Normal distribution isn't suitable for a binary outcome, but the *Bernoulli distribution* is:

$$Y_i \sim Ber(g(z_i))$$

- The mean parameter of a Bernoulli distribution is $Pr(Y = 1)$
  - So, we must transform our linear predictors using a function, $g()$, such that z_i, which could be any real number is mapped to values between 0 and 1

Logistic Regression

Logistic regression is a generalized linear model that uses the *Bernoulli distribution* and the **sigmoid function**: $g(z_i) = \frac{1}{1+e^{-z_i}}$



The observed outcome (ie: $y_i = 0$ or $y_i = 1$) is a realized value from a Bernoulli distribution with a mean of $g(z_i)$

Parameters and Cost Function

The cost function used to estimate the weights of a logistic regression model is **cross-entropy loss**:

$$\text{Cost} = -\frac{1}{n}\sum_{i=1}^{n}\big(y_i log(g(z_i)) + (1-y_i) log(1-g(z_i))\big)$$

Or, more intuitively:

$$\begin{aligned} \text{Cost}_i &= -\frac{1}{n} log(g(z_i)) & \text{if } y_i = 1 \\ \text{Cost}_i &= -\frac{1}{n} log(1-g(z_i)) & \text{if } y_i = 0 \end{aligned}$$

- For observations in the positive class, higher z values correspond with lower costs
  - As z_i increases, $g(z_i) \rightarrow 1$ and $log(1) = 0$

Grinnell College
Statistics

Optimization

The cross-entropy cost function doesn't have a closed form solution. For simplicity, we'll consider *only one stochastic gradient descent*:

$$\frac{\partial \text{Cost}}{\partial \mathbf{w}} = -y_i \mathbf{x}_i \left(\frac{g(\mathbf{x}_i)(1-g(\mathbf{x}_i))}{g(\mathbf{x}_i)} \right) + (1-y_i)\mathbf{x}_i \left(\frac{g(\mathbf{x}_i)(1-g(\mathbf{x}_i))}{1-g(\mathbf{x}_i)} \right)$$

Note that by chain rule: $\nabla log(g(z_i)) = \frac{1}{g(z_i)} * \frac{\partial g}{\partial z_i} * \frac{\partial z_i}{\partial \mathbf{w}}$

- $\frac{1}{g(z_i)}$ is the derivative of $log(g(z_i))$ with respect to $g(z_i)$
- $\frac{\partial g}{\partial z_i} = g(\mathbf{x}_i)(1-g(\mathbf{x}_i))$
- $\frac{\partial z_i}{\partial \mathbf{w}} = \mathbf{x}_i$

These components of the second term are found similarly.

Grinnell College
Statistics

Stochastic Gradient Descent Algorithm

Skipping a fair bit of algebra, the gradient function for only one stochastic gradient descent simplifies to:

$$(g(z_i) - y)\mathbf{x}_i$$

Leading to the following update scheme:

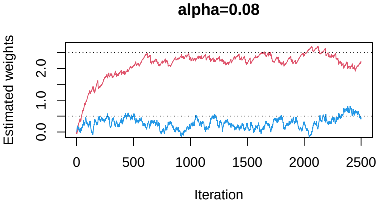
$$\mathbf{w}^{(j)} = \mathbf{w}^{(j-1)} - \alpha * \left(g(z_i^{(j-1)}) - y)\mathbf{x}_i\right)$$

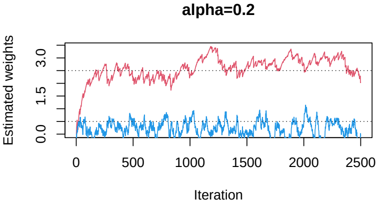
Noting that the prediction score, $z_i^{(j-1)}$, involves the weights from the previous iteration, $\mathbf{w}^{(j-1)}$

Grinnell College
Statistics

The examples below demonstrate only one stochastic gradient descent on 100 data-points generated such that:

$$Y \sim Ber\left(\frac{1}{1+exp(-(0.5+2.5x_1))}\right)$$

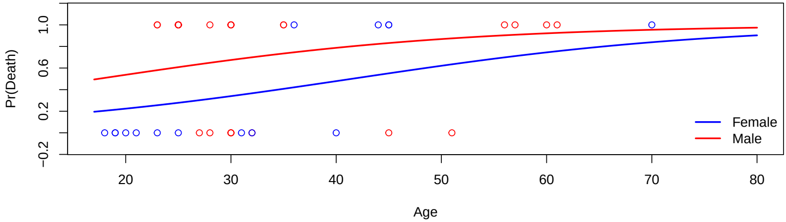




Grinnell College
Statistics

Visualizing Logistic Regression

Below is what the fitted logistic regression model looks like for the Donner Party example:



The most important take-away is that our model follows a defined parametric structure, and always yields predicted probabilities between 0 and 1.

Softmax Regression

- Logistic regression is designed for binary outcomes; however, the method can be generalized to multi-label classification
  - **Softmax regression**, also known as multinomial logistic regression, models the probability of class membership for each class via:

$$Pr(y_i = K) = \frac{exp(\mathbf{w}_K^T \mathbf{x}_i)}{\sum_{l=1}^{N_k} exp(\mathbf{w}_l^T \mathbf{x}_i)}$$

- The cost function for softmax regression is:

$$\text{Cost} = -\sum_{i=1}^{n} \sum_{l=1}^{k} \mathbb{1}(y_i = l) * log\left(\frac{exp(\mathbf{w}_l^T \mathbf{x}_i)}{\sum_{l=1}^{k} exp(\mathbf{w}_l^T \mathbf{x}_i)}\right)$$

- For $k = 2$, this simplifies to the cross-entropy cost function of logistic regression

Softmax Regression

- Softmax regression is unusual in the sense that uses a redundant set of parameters
  - That is, there are a set of weights for each of the k classes, but the same predictions could be obtained with weights for $k-1$ classes
  - This is evident when comparing the method with logistic regression, where $k=2$ but only 1 set of weights is estimated

Softmax Regression

- Softmax regression is unusual in the sense that uses a redundant set of parameters
  - That is, there are a set of weights for each of the k classes, but the same predictions could be obtained with weights for $k-1$ classes
  - This is evident when comparing the method with logistic regression, where $k=2$ but only 1 set of weights is estimated
- For this reason, there's little value in studying the weights of a softmax regression model, as there are multiple sets of weights that also will optimize the cost function
  - This is in contrast to logistic regression, where the exponentiation of a weight reflects the multiplicative impact of a 1-unit change in that variable on the odds of outcome belonging to the positive class

For more on Softmax Regression: http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/