Linear Models for Classification Tasks

Ryan Miller

Outline

1. Logistic regression for binary classification
2. Parameters and optimization
3. Softmax regression

Shortcomings of Linear Regression

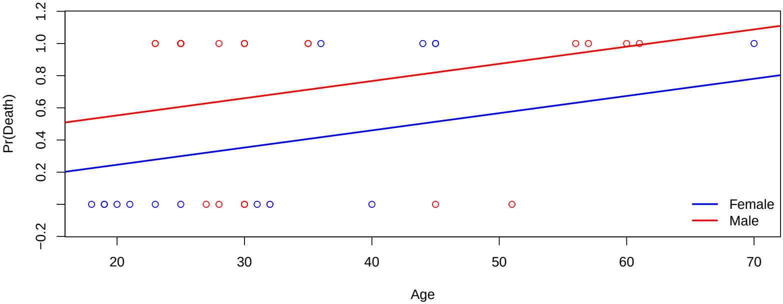
Linear regression is a supervised learning approach that models the dependence of a numeric outcome on a set of predictors as linear:

$$Y = w_o + w_1 X_1 + w_2 X_2 + \ldots + w_p X_p + \epsilon$$

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

Example (Donner Party Survival)

This model contains two predictors, “Age”, and “Sex” (which is incorporated into the predictor matrix using one-hot encoding):



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

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:

$$Y \sim N(\eta, \sigma \mathbf{I}), \text{ where: } \eta = w_o + w_1 X_1 + w_2 X_2 + \dots$$

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

Generalized Linear Models

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

$$Y \sim Ber(g(\eta))$$

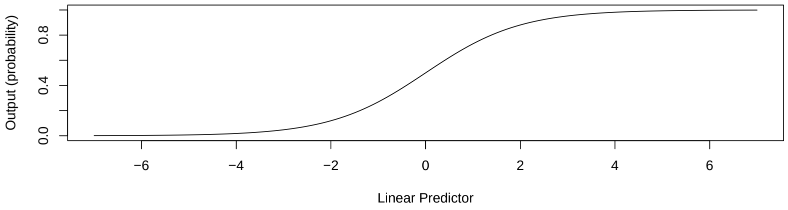
- The mean of the Bernoulli distribution is defined as $Pr(Y = 1)$
  - So, we must transform our linear predictors using a function, $g()$, such that only inputs between 0 and 1 are possible

Logistic Regression

Logistic regression is a generalized linear model that uses the *Bernoulli distribution* and the **sigmoid function**:

$$g(\eta) = \frac{1}{1+exp(-\eta)}$$

This function maps prediction scores to probabilities in the follow manner:



Observed outcomes (ie: $y_i = 0$ or $y_i = 1$) are considered samples from a Bernoulli distribution with a mean of $g(\eta)$

Parameters and Cost Function

- Just like linear regression, logistic regression involves weights that must be estimated from the data
  - However, a different cost function should be used, the most popular being **cross-entropy loss**:

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

More intuitively:

$$\text{Cost}_i = -\frac{1}{n}log(g(\eta_i)) \qquad \text{if } y_i = 1$$

$$\text{Cost}_i = -\frac{1}{n}log(1-g(\eta_i)) \qquad \text{if } y_i = 0$$

Parameters and Cost Function

- Just like linear regression, logistic regression involves weights that must be estimated from the data
  - However, a different cost function should be used, the most popular being **cross-entropy loss**:

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

More intuitively:

$$\text{Cost}_i = -\frac{1}{n}log(g(\eta_i)) \qquad \text{if } y_i = 1$$
$$\text{Cost}_i = -\frac{1}{n}log(1-g(\eta_i)) \qquad \text{if } y_i = 0$$

- Notice observations with large prediction scores make minimal contributions to the cost function if they belong to the positive class
  - As an observation's prediction score, η_i increases, $g(\eta_i) \to 1$ and $log(1) = 0$

Optimization

The cross-entropy cost function doesn't have a closed form solution, so it needs to be optimized by gradient descent. 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: $\nabla log(g(\eta_i)) = \frac{1}{g(\eta_i)} * \frac{\partial g}{\partial \eta_i} * \frac{\partial \eta_i}{\partial \mathbf{w}}$

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

Similar arguments apply to the other term

Stochastic Gradient Descent Algorithm

Skipping some algebra, the gradient function for only one stochastic gradient descent reduces to:

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

Leading to the following update scheme:

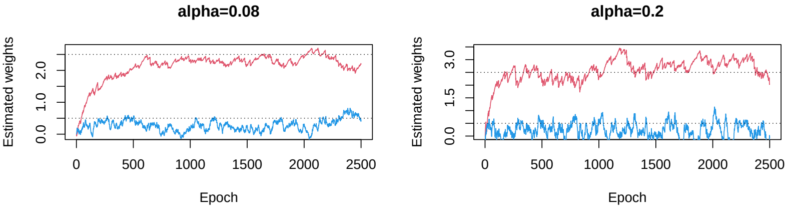
$$\mathbf{w}^{(j)} = \mathbf{w}^{(j-1)} + \alpha(g(\eta_i^{(j-1)}) - y)\mathbf{x}_i$$

Where the prediction score, η_i, is computed using weights from previous iteration.

Optimization

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

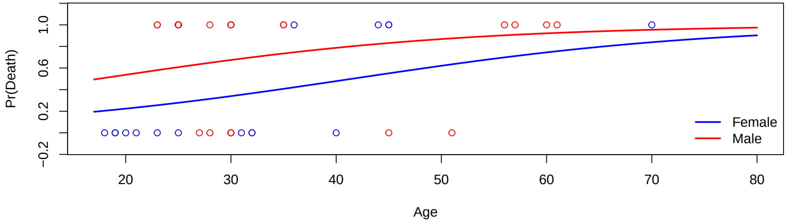
$$Y \sim Ber\left(\frac{1}{1+exp(-(0.5+2.5x_1))}\right)$$



Compared to linear regression (which had a closed form solution), logistic regression is much more difficult to optimize

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 it 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 settings
  - **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/