CIS5300 - Speech and Language Processing - Chapter 5 Notes

Jonathon Delemos - Chris Callison Burch

July 4, 2025

0.1 Abstract - Logistic Regression

In this chapter we introduce an algorithm that is admirably suited for discovering the link between features or clues and some particular outcome: **logistic regression**. Indeed, logistic regression is one of the most important analytic tools in the social and natural sciences. In natural language processing, logistic regression is the base- line supervised machine learning algorithm for classification, and also has a very close relationship with neural networks. As we will see in Chapter 7, a neural network can be viewed as a series of logistic regression classifiers stacked on top of each other. Thus the classification and machine learning techniques introduced here will play an important role throughout the book.

0.2 *Components of Probabilistic Machine Learning Classifier*

- **Feature Representation** - for each input, there will be vector of features.
- **Sigmoid and Softmax** - tools for classification. We will show the formulas needed later.
- **Cross-Entropy Loss Function** - minimizing the loss corresponding to error in training.
- **Stochastic Gradient Descent** - This means updating the weighted variables each time a vector is processed, as opposed to waiting for a batch transaction then updating the weights.

0.3 The Sigmoid Classifier/Function

For this step of our probabalistic machine learning classifier, we will focus on making a simple value that is a sum of the weights multiplied by the input parameter added with the bias term. This is the **sigmoid classifier (z)**. The bias (b) can be thought of as a base *y-intercept* for the model.

$$z = (\sum_{i=1}^{n} w_i x_i) + b$$

This sigmoid value can range anywhere between - ∞ and ∞. It is not bound by the legal probabilties of 0-1, which we will see later.
To create a probability, we'll pass the z through the **sigmoid function** $\sigma(z)$. This is also called the logistic function.

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+exp(-z)}$$

We are almost there. If we apply the sigmoid to the sum of the weighted features, we get a number 0 and 1. To make it a probability, we just need to make sure that the two cases p(y=1) and p(y=0) sum to 1.

$$P(y=1) = \sigma(w \cdot x + b)$$

Recall:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+exp(-z)}$$

Substitution for the z:

$$= \frac{1}{1 + exp(-(w \cdot x + b))}$$

Next, the probability that y = 0.

$$P(y = 0) = \frac{exp(-(w \cdot x + b))}{1 + exp(-(w \cdot x + b))}$$

0.4 5.2 - Classification with Logistic Regression

To make decisions that will snap the result to a *0* or *1*, we create a **decision boundary**.

$$\text{decision}(x) = \begin{cases} 1 & \text{if } P(y = 1 \mid x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

For example, we will work a problem from start to finish.

$$p(+|x) = P(1|x) = \sigma(w \cdot x + b)$$

These vectors are 6x1 * 6x1. This results in a 1x1 answer which can later be added to the bias.

$$= \sigma[2.5, -5.0, -1.2, .5, 2.0, .7] \cdot [3, 2, 1, 3, 0, 4.19] + .1$$

$$= \sigma(.833)$$

$$= .70$$

$$p(-|x) = P(0|x) = 1 - \sigma(w \cdot x + b)$$

$$= .30$$

If we want to scale the results, we can calculate the μ, σ, and normalize for best interpretation. This will create z-scores.

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} x_i^j$$

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{j=1}^{m} (x_i^j - \mu_i)^2}$$

Now, we substitute:

$$x_i = \frac{x_i - \mu_i}{\sigma_i}$$

Normalized:

$$x_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$$

0.5 5.3 - Multinomial Logistic Regression

Sometimes we need more than two classes. Perhaps we might want to do 3-way sentiment classification (positive, negative, or neutral). Or we could be assigning some of the labels we will introduce in Chapter 17, like the part of speech of a word (choosing from 10, 30, or even 50 different parts of speech), or the named entity type of a phrase (choosing from tags like person, location, organization). In such cases, we use **Multinomial Logistic Regression**, also called **softmax regression**.
When we have multiple potential classes that could be correct, we call the correct class **one-hot vetor**. The job of the classifier is to produce an estimate vector $\hat{y}$. The multinomial logistic classifer uses a generalization of the sigmoid, called the **softmax** function.

$$softmax(z_i) = \frac{expz_i}{\sum_{j=1}^{K} exp(z_j)} 1 \leq i \leq K$$

Recall:

$$z = (\sum_{i=1}^{n} w_i x_i) + b$$

$$exp(z) = e^z$$

More succintly, we can call the softmax a method to calculate the probability of each class within a vector. The next step is now to determine correctness and update our weights. This is what allows for machine learning.

Next, we will apply softmax in logistic regression and combine it with probability.

$$p(y_k = 1 \mid x) = \frac{exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} exp(w_j \cdot x + b_j)}$$

The purpose of this equation is to allow for the vectors to be calculated with different classes. The result could look like this:

$$z = [.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

One helpful interpretation is to look at W (weight) as the prototype correct answer.
Features in multinomial logistic regression act like features in binary logistic regres sion, with the difference mentioned above that well need separate weight vectors and biases for each of the K classes.

0.6 Cross-Entropy Loss

The cross-entropy loss function is a function that calculates for how close the classifer output is to the correct output (0 or 1).

$$L(\hat{y}, y) = \hat{y}\Delta y$$

This represents the change in the predicted value versus the actual value. We choose the parameters w,b that maximize the log probability of the true y labels in the training data.
Here is the base formula.

$$p(y \mid x) = y \log \hat{y}(1 - y) \log(1 - \hat{y})$$

The basic premise of this cross-entropy loss filter is we want the loss to be smaller if the model's estimate is close to correct, and bigger if the model is confused.

0.7 5.6 - Gradient Descent

Our goal with gradient descent is to find the optimal weights: minimize the loss function we've defined for the model. In machine learning, we generally refer to the pair of learned variables as θ. So, θ = (w,b). Conviently, this is a convex equation. That means it has a minimum point. Using a partial derivative, we can easily calculate the minimum value. By setting this value to zero, that will determine the input value. This is the input value that will minimize the cross-entropy loss. The magnitude of the amount to move in gradient descent is the value of the slope. A higher learning rate η indicates a higher step rate. In logistic regression, and due to multinomial regression, we always have multiple w being expressed as a vector. In each dimension, we are trying to find the value that will minimuze the loss. Starting the derivations:

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta), y)$$

The gradient for Logistic Regression:

$$\frac{\partial L_{CE}(\hat{y}, y)}{\partial w_j} = (\delta(w \cdot x + b) - y)x_j$$

0.8 5.6 - Stochastic Gradient Descent

The basic concept behind stochastic application is to update the weighted variables. With larger batches of training sets, the w and b should be redefined after each pass through.

There is an example here on page 19 that shows all parts of the logistic regression model being used to generate a result.

0.9 5.7- Regularization

There is a problem with learning weights that make the model perfectly match the training data. If a feature is perfectly predictive of the outcome because it happens to only occur in one class, it will be assigned a very high weight. The weights for features will attempt to perfectly fit details of the training set, in fact too perfectly, modeling noisy factors that just accidentally correlate with the class. This problem is called overfitting. A good model should be able to generalize well from the training.
Here we also here the terms euclidean distance and manhattan distance. These are very interesting ways of planning routes or measuring the distance from the correct location.

We update the variables here:

$$\hat{\theta} = argmax_\theta \sum_{i=1}^{m} \log P(y^i \mid x^i) - \alpha R(\theta)$$

0.10 Summary

This weeks reading focuses on logistic regression and classification.

- Logistic regression is a supervised learning technique tht extracts features from the input, passes them through a sigmoid, and generates a probability.
- Logistic regression can be done with multiple classes, this is called multinomial logistic regression.
- Softmax is how we compute multinomial regression.
- Weights are learned using *cross-entropy loss*.
- Minimizing loss is done through iterative algorithms like gradient descent.
- Regularization is used to avoid fitting.

0.11 Questions

It would be nice to speak with an expert. Not sure if my understanding of the material is entirely accurate.