Lisa Yan
CS109

Lecture Notes #25
June 3, 2020

Logistic Regression

Based on a chapter by Chris Piech

Before we get started, I want to familiarize you with some notation:

$$\theta^T \mathbf{X} = \sum_{i=1}^{n} \theta_i X_i = \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_n X_n \qquad \text{weighted sum}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \text{sigmoid function}$$

Logistic Regression Overview

Classification is the task of choosing a value of y that maximizes $P(Y|\mathbf{X})$. Naïve Bayes worked by approximating that probability using the naïve assumption that each feature was independent given the class label.

For all classification algorithms you are given n I.I.D. training datapoints $(\mathbf{x}^{(1)}, y^{(1)})$, $(\mathbf{x}^{(2)}, y^{(2)})$, ... $(\mathbf{x}^{(n)}, y^{(n)})$ where each "feature" vector $\mathbf{x}^{(i)}$ has $m = |\mathbf{x}^{(i)}|$ features.

Logistic Regression Assumption

Logistic Regression is a classification algorithm (I know, terrible name) that works by trying to learn a function that approximates $P(Y|X)$. It makes the central assumption that $P(Y|X)$ can be approximated as a sigmoid function applied to a linear combination of input features. Mathematically, for a single training datapoint $(\mathbf{x}, y)$ Logistic Regression assumes:

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma(z) \text{ where } z = \theta_0 + \sum_{i=1}^{m} \theta_i x_i$$

This assumption is often written in the equivalent forms:

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma(\theta^T \mathbf{x}) \qquad \text{where we always set } x_0 \text{ to be 1}$$

$$P(Y = 0|\mathbf{X} = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x}) \qquad \text{by total law of probability}$$

Using these equations for probability of $Y|X$, we can create an algorithm that select values of θ that maximize that probability for all data. I am first going to state the log probability function and partial derivatives with respect to θ. Then later we will (a) show an algorithm that can chose optimal values of θ and (b) show how the equations were derived.

An important realization is that given the best values for the parameters (θ), logistic regression often can do a great job of estimating the probability of different class labels. However, given bad—or even random—values of θ, it does a poor job. The amount of "intelligence" that your logistic machine laerning algorithm has is dependent on having good values of θ.

Log Likelihood

In order to choose values for the parameters of logistic regression, we use Maximum Likelihood Estimation (MLE). As a result, we will have two steps: (1) Write the log-likelihood function, and (2) find the values of θ that maximize the log-likelihood function.

The labels that we are predicting are binary, and the output of our logistic regression function is $P(Y = 1|\mathbf{X} = \mathbf{x})$. This means that we can (and should) interpret the Logistic Regression model as a Bernoulli random variable: $Y|\mathbf{X} = \mathbf{x} \sim \text{Ber}(p)$, where $p = \sigma(\theta^T\mathbf{x})$.

To start, here is a super slick way of writing the probability of one datapoint (recall that this is the non-piecewise form of writing the probability mass function of a Bernoulli random variable):

$$P(Y = y|X = \mathbf{x}) = \sigma(\theta^T\mathbf{x})^y \cdot \left[1 - \sigma(\theta^T\mathbf{x})\right]^{(1-y)}$$

Now that we know the probability mass function of a single datapoint, we can write the *conditional* likelihood of all the data, where each datapoint is independent:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} P(Y = y^{(i)}|X = \mathbf{x}^{(i)}) \\ &= \prod_{i=1}^{n} \sigma(\theta^T\mathbf{x}^{(i)})^{y^{(i)}} \cdot \left[1 - \sigma(\theta^T\mathbf{x}^{(i)})\right]^{(1-y^{(i)})} \end{aligned}$$

And if you take the log of this function, you get the log-conditional likelihood of the training dataset for Logistic Regression.

Side Note: While we calculate conditional likelihood here, it is worthwhile noting that maximizing log-likelihood and log-conditional likelihood are equivalent:

$$\begin{aligned} LL(\theta) &= \sum_{i=1}^{n} \log\big(f(\mathbf{x}^{(i)}, y^{(i)}|\theta)\big) = \sum_{i=1}^{n} \log\big(f(\mathbf{x}^{(i)}|\theta)P(y^{(i)}|\mathbf{x}^{(i)}, \theta)\big) && \text{Chain rule} \\ &= \sum_{i=1}^{n} \log\big(f(\mathbf{x}^{(i)})f(y^{(i)}|\mathbf{x}^{(i)}, \theta)\big) && \mathbf{X}, \theta \text{ independent} \\ \theta_{MLE} = \underset{\theta}{\text{argmax}} LL(\theta) &= \underset{\theta}{\text{argmax}} \left(\sum_{i=1}^{n} \log f(\mathbf{x}^{(i)}) + \log f(y^{(i)}|\mathbf{x}^{(i)}, \theta)\right) && \text{Log of products} \\ &= \underset{\theta}{\text{argmax}} \left(\sum_{i=1}^{n} \log f(y^{(i)}|\mathbf{x}^{(i)}, \theta)\right) && \text{Constants w.r.t. } \theta \end{aligned}$$

Gradient of Log Likelihood

In MLE, now that we have a function for log-likelihood, we simply need to chose the values of θ that maximize it. We can find the best values of θ by using an optimization algorithm. However, in order to use an optimization algorithm, we first need to know the partial derivative of log likelihood with respect to each parameter. First I am going to give you the partial derivative (so you can see how it is used). Then I am going to show you how to derive it.

Here is the partial derivative of log-likelihood with respect to each parameter θ_j:

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Parameter estimation using gradient ascent optimization

Once we have an equation for Log Likelihood, we chose the values for our parameters (θ) that maximize said function. In the case of logistic regression we can't solve for θ mathematically. Instead we use a computer to chose θ.

To do so we employ an algorithm called gradient ascent. That algorithms claims that if you continuously take small steps in the direction of your gradient, you will eventually make it to a local maxima. In the case of Logistic Regression you can prove that the result will always be a global maxima.

The small step that we continually take given the training dataset can be calculated as follows:

$$\begin{aligned} \theta_j^{\text{new}} &= \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}} \\ &= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}, \end{aligned}$$

where η is the magnitude of the step size that we take. If you keep updating θ using the equation above you will converge on the best values of θ!

Pro-tip: Don't forget that in order to learn the value of θ_0, you can simply define $\mathbf{x}_0 = 1$ for all datapoints.

Derivations

In this section we provide the mathematical derivations for the log-likelihood function and the gradient. The derivations are worth knowing because these ideas are heavily used in Artificial Neural Networks.

Our goal is to calculate the derivative of the log likelihood with respect to each theta. To start, here is the definition for the derivative of sigma with respect to its inputs:

$$\frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1 - \sigma(z)] \qquad \text{to get the derivative with respect to } \theta\text{, use the chain rule}$$

Derivative of gradient for one datapoint ($\mathbf{x}$, y):

$$
\begin{aligned}
\frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j}(1-y)\log[1-\sigma(\theta^T \mathbf{x}] && \text{derivative of sum of terms} \\
&= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1-y}{1-\sigma(\theta^T x)}\right] \frac{\partial}{\partial \theta_j}\sigma(\theta^T x) && \text{derivative of } \log f(x) \\
&= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1-y}{1-\sigma(\theta^T x)}\right] \sigma(\theta^T x)[1-\sigma(\theta^T x)]x_j && \text{chain rule + derivative of sigma} \\
&= \left[\frac{y-\sigma(\theta^T x)}{\sigma(\theta^T x)[1-\sigma(\theta^T x)]}\right] \sigma(\theta^T x)[1-\sigma(\theta^T x)]x_j && \text{algebraic manipulation} \\
&= \left[y-\sigma(\theta^T x)\right] x_j && \text{cancelling terms}
\end{aligned}
$$

Because the derivative of sums is the sum of derivatives, the gradient of theta is simply the sum of this term for each training datapoint.