Gradient Descent, Feature Engineering, and Regularization

Somani, Neel - Bao, Jason

February 6, 2018

1 Introduction

Recall the univariate OLS solution:

$$\sum_{i=1}^{n}(x_i * w - y_i)^2 \tag{1}$$

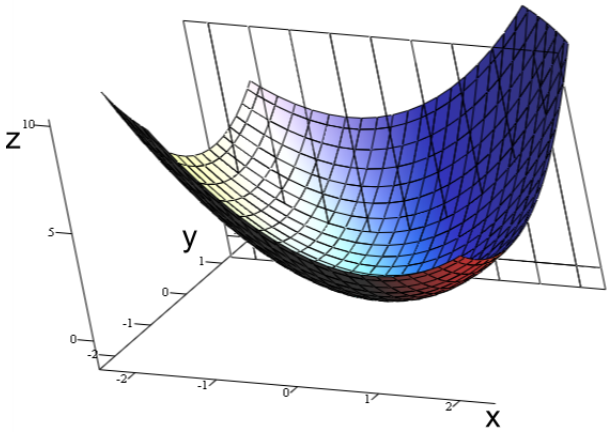
What kind of equation is this, if you factor it out? Answer: It's a quadratic.

2 Convex Functions

Generally, quadratics are of the form $f(\vec{x}) = \vec{x^T} * A * \vec{x} + b^T * \vec{(x)} + c$. If A is *positive semidefinite*, then f is *convex*. That is, if:

$$\forall x : \vec{x^T} * A * \vec{x} \geq 0 \tag{2}$$

then f is particularly easy to optimize (i.e., convex), by a method called *gradient descent*. By convex, we mean that the function looks something like this (a "bowl" shape):



3 Gradient Descent

Sometimes there's no easy solution like OLS to an optimization problem. What would you do if you were at a random point in that bowl, and you were trying to find the bottom? (Application: You have a graph of various inputs into a product that you're producing, and you've graphed the price. You're trying to find the minimum price to produce your product.)

You would probably look around, find the direction of the steepest descent, and move in that direction. That's exactly what gradient descent does:

$$w_n + 1 = w_n - \epsilon * \nabla(loss) \tag{3}$$

where w_0 is a random vector, and $\nabla(loss)$ is the direction that the multivariate function decreases most rapidly. ϵ represents the "step size," or how quickly we jump around the function. In other words, we guess a point is the minimum, figure out what direction the function decreases most rapidly, and move in that direction by some amount.

Why not just set ϵ to be as large as possible? Answer: You might overstep the minimum.

4 Revisiting: Feature Engineering

Suppose you see samples from the real world and you know there is some good approximation of $y(x_1, x_2)$ that has the form $\hat{y}(x_1, x_2) = w_1 sin(x_1 x_2) + w_2 e^{x_1} + w_3 x_2^2 + w_4 x_1$. $\hat{y}$ is a function that takes in an input vector, and outputs a "featurized" vector.

We want to solve for w that optimizes:

$$\min_w \sum_{i=1}^{m} (\hat{y}(x_{ii}, x_{zi}) * w - y_i)^2$$

The idea is that we can set up an equation that's linear with respect to w – even though the input vector $\hat{y}$ is not linear – and we can find the optimal value of w using OLS.

5 Feature Engineering: Encoding Categories

One hot encoding is a way to quantify non-quantitative (categorical) data. For example, if we want one of the features to be "color" we can one-hot encode the different types of colors with "1" if the color is present and "0" if the color is not, with a separate dimension for each different color.

The reason for doing a one-hot encode like this is because it doesn't imply a relationship between categorical data. For example, if we encoded the colors instead as 0 for black, 1 for red, 2 for blue, and 3 for orange, our model will assume blue to be closer to orange then black to orange. For example, in the OLS model, this is easily seen as $(2-3)^2$ is a larger error than $(0-3)^2$. Another problem is that if our model identifies a sample point as being halfway between black (0) and blue (2), then it might average the two and instead output red (1).

6 Business Application: Overfitting

We notice that if there are enough features, we can perfectly fit any dataset by setting each data point to a separate feature, ensuring 100% training accuracy.

How do we quantify how "good" our model is? One way is to get the mean squared error of some data that we didn't use for training, which we'll call the *test set*. We use our model to predict $\hat{y}$ and calculate

$$\sum_{y \in TestSet} (\hat{y} - y)^2 \tag{4}$$

In lecture, we saw how overfitting affected the model that we trained on in practice.

7 Dimensionality Reduction: A Solution to Overfitting

While we didn't dive too deeply into this in lecture, we talked about what variance was intuitively – the spread of a set of data, or how much a particular set of data varied.

We figured that the direction with the most variance would be the direction that would be most useful for regression/classification. (Using the directions with maximal variance is called PCA, or principal components analysis.) But in the case that there was a significant amount of noise in one direction, that direction was identified as the "most significant," even though the noise isn't actually helpful for prediction.

8 Regularization: Another Solution to Overfitting

How else can we combat overfitting? For one, we can penalize "complexity," or in other words, penalize using more features/encourage using a smaller weight

vector.

In the univariate case, instead of minimizing the loss function:

$$\sum_{i=1}^{n}(x_i^T w - y_i)^2 \tag{5}$$

we might minimize:

$$\sum_{i=1}^{n}(x_i^T w - y_i)^2 + \lambda \sum_{i=1}^{n}(w_i^2) \tag{6}$$

where λ is some constant that we pick (depending on how much we want to penalize complexity), called the "regularization constant." (We'll talk about how to pick a good regularization constant later in the course.) In other words, we're including the magnitude of the weight vector in our optimization function, in order to discourage weighting arbitrary features that are likely fitting to noise.

The specific optimization function in (6) is called "ridge regression." There's another optimization function called LASSO regression, which has its own set of advantages:

$$\sum_{i=1}^{n}(x_i^T w - y_i)^2 + \lambda \sum_{i=1}^{n}(w_i) \tag{7}$$

Again, the purpose of the regularization $\lambda \sum_{i=1}^{n}(w_i)$ is to penalize complexity, which is an attempt at reducing overfitting.