Lisa Yan
CS109

Lecture Notes #24
June 1, 2020

Linear Regression and Gradient Ascent

Based on a chapter by Chris Piech and Lisa Yan

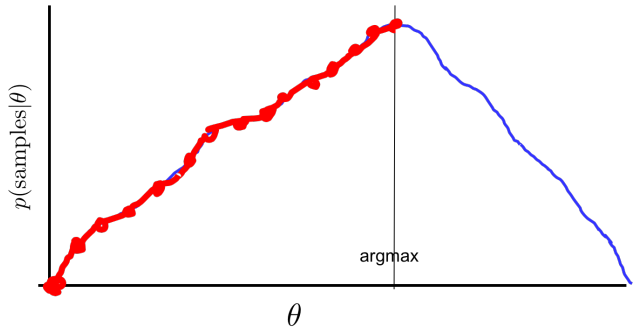
Regression

Regression is a second category of Machine Learning prediction algorithms. You have a prediction function $\hat{Y} = g(\mathbf{X})$ as before, but you would like to predict a Y that takes on a **continuous** number.

We won't elaborate on the regression task too much, because classification (with discrete Y) already has a plethora of modern computer science applications—image recognition, sentiment analysis of text, and text authorship, to name a few. However, we will explore *linear regression* (where we model g as a linear function) and learn a truly valuable iterative optimization algorithm (the "butter" to machine learning's "bread," if you will) called **gradient ascent.**

Gradient Ascent Optimization

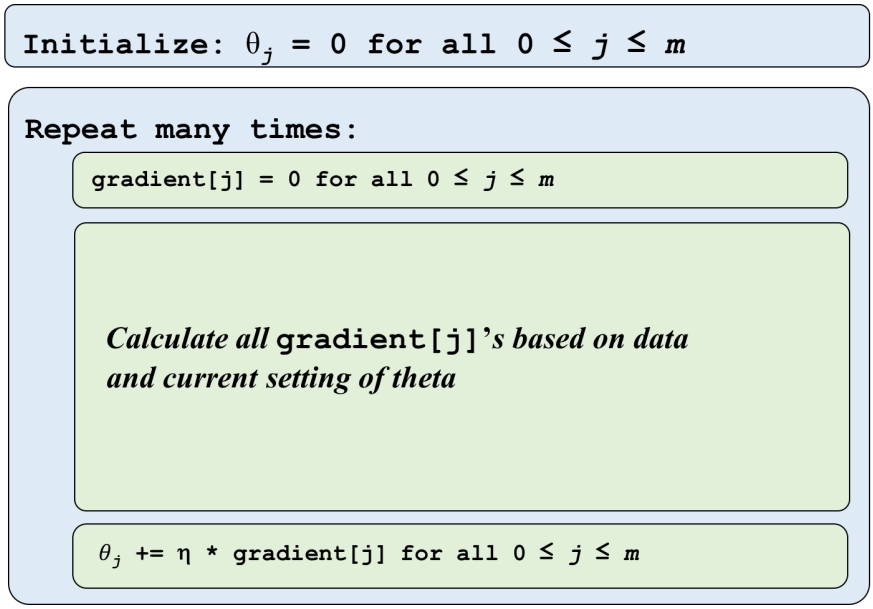
In many cases we can't solve for argmax mathematically. Instead we use a computer. To do so we employ an algorithm called gradient ascent (a classic in optimization theory). The idea behind gradient ascent is that if you continuously take small steps in the direction of your gradient, you will eventually make it to a local maxima.



Start with theta as any initial value (often 0). Then take many small steps towards a local maxima. The new theta after each small step can be calculated as:

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j}$$

Where “eta" (η) is the magnitude of the step size that we take. If you keep updating θ using the equation above you will (often) converge on good values of θ. As a general rule of thumb, use a small value of η to start. If ever you find that the function value (for the function you are trying to argmax) is decreasing, your choice of η was too large. Here is the gradient ascent algorithm in pseudo-code:



Linear Regression

Suppose we are working with 1-dimensional observations, i.e., $\mathbf{X} =< X_1 >= X$. Linear Regression assumes the following linear model for prediction, which has two parameters: a and b:

$$\hat{Y} = g(\mathbf{X}) = aX + b$$

Using this model, we would like to determine the optimal parameters according to some optimization objective. We discuss two approaches: an analytical approach that minimizes mean squared error, and a computational approach that maximizes training data likelihood. With one important assumption (which we’ll get to later), the two approaches are equivalent.

Analytical Solution with Mean Squared Error

For regression tasks, we usually decide a prediction $hatY = g(X)$ that minimizes the *mean squared error* (MSE) "loss" function:

$$\theta_{MSE} = \underset{\theta}{\text{argmin}}\, E[(Y - \hat{Y})^2] = \underset{\theta}{\text{argmin}}\, E[(Y - g(\mathbf{X}))^2] = \underset{\theta}{\text{argmin}}\, E[(Y - aX - b)^2]$$

With our linear prediction model, we determine $\theta_{MSE} = (a_{MSE}, b_{MSE})$ by differentiating the mean squared error with respect to a and b:

$$\frac{\partial}{\partial a} E[(Y - aX - b)^2] = E[-2(Y - aX - b)X] = -2E[XY] + 2aE[X^2] + 2bE[X]$$
$$\frac{\partial}{\partial b} E[(Y - aX - b)^2] = E[-2(Y - aX - b)] = -2E[Y] + 2aE[X] + 2b$$

Setting derivatives to 0 and solving for simultaneous equations:

$$a_{MSE} = \frac{E[XY] - E[X]E[Y]}{E[X^2] - (E[X])^2} = \frac{\text{Cov}(X, Y)}{\text{Var}(X)} = \rho(X, Y)\frac{\sigma_X}{\sigma_Y}$$
$$b_{MSE} = E[Y] - aE[X] = \mu_Y - a\mu_X$$
$$Y = \rho(X, Y)\frac{\sigma_Y}{\sigma_X}(X - \mu_X) + \mu_Y$$

Wait, those are our best parameters? But we don't know the distributions of X and Y, and therefore we don't know true statistics on X and Y. We estimate these statistics based on our observed training data, Our model is therefore as follows (where $\bar{X}$ and $\bar{Y}$ are the sample means computed from the training data:

$$\hat{Y} = g(X = x) = \hat{\rho}(X, Y)\frac{\hat{\sigma_Y}}{\hat{\sigma_X}}(x - \bar{X}) + \bar{Y}$$
$$\hat{a}_{MSE} = \frac{\sum_{i=1}^{n}(x^{(i)} - \bar{X})(y^{(i)} - \bar{Y})}{\sum_{i=1}^{n}(x^{(i)} - \bar{X})^2}$$
$$\hat{b}_{MSE} = \bar{Y} - \hat{a}_{MSE}\bar{X}$$

Computational Solution with Maximum Likelihood

That seemed somewhat anticlimactic: we had this optimal prediction function, but we had to estimate the parameters of the prediction function from the training data. Let's borrow an idea from our parameter estimation unit by maximizing the likelihood of our training data!

Recall that our training data has n datapoints: $((x^{(1)}, y^{(1)}), ((x^{(2)}, y^{(2)}), \ldots, ((x^{(n)}, y^{(n)})$, generated i.i.d. according to the joint distribution of X and Y, $f(X, Y|\theta)$. We can model this joint distribution by incorporating our regression model: $Y = \hat{Y} + Z = aX + b + Z$, where $\hat{Y} = g(X) = aX + b$ is our prediction and Z is our error (i.e., noise) between our prediction $\hat{Y}$ and the actual Y.

We approach the problem of finding a and b that maximize the likelihood of our train data by first finding a distribution involving Y, X, and $\theta = (a, b)$. We then find the value of θ that maximizes the log-likelihood function.

If we assume $Z \sim \mathcal{N}(0, \sigma^2)$ and X follows some unknkown distribution, then we can calculate the conditional distribution of Y given X is some number x and we have some parameter values $\theta = (a, b)$ as simply $Y = aX + b + Z$. This is just the sum of a Gaussian and a number, thereby implying that $Y|X, \theta \sim \mathcal{N}(aX + b, \sigma^2)$, which has PDF

$$f(Y = y|X = x, \theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-ax-b)^2}{2\sigma^2}}.$$

Now we are ready to write the likelihood function, then take its log to get the log likelihood function:

$$\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} f(y^{(i)}, x^{(i)}|\theta) && \text{Let's break up this joint} \\
&= \prod_{i=1}^{n} f(y^{(i)}|x^{(i)}, \theta) f(x^{(i)}) && \text{Chain rule, } f_X(x^{(i)}) \text{ is independent of } \theta \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-(y^{(i)}-ax^{(i)}+b)^2/(2\sigma^2)} \cdot f(x^{(i)}) && \text{Substitute in the conditional distribution of } Y|X, \theta
\end{aligned}$$

$$\begin{aligned}
LL(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-(y^{(i)}-ax^{(i)}-b)^2/(2\sigma^2)} f(x^{(i)}) && \text{Substitute in } L(\theta) \\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-(y^{(i)}-ax^{(i)}-b)^2/(2\sigma^2)} + \sum_{i=1}^{n} \log f(x^{(i)}) && \text{Log of a product is the sum of logs} \\
&= n \log \frac{1}{\sqrt{2\pi}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^{(i)} - ax^{(i)} - b)^2 + \sum_{i=1}^{n} \log f(x^{(i)})
\end{aligned}$$

Our goal is to find parameters a, b that maximize likelihood. Remember that argmax is invariant of logarithmic transformations and positive scalar constants, and additive constants? Let's remove positive constant multipliers and terms that don't include θ. We are left with trying to find a value of θ that maximizes:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[-\sum_{i=1}^{n} (y^{(i)} - ax^{(i)} - b)^2 \right]$$

To solve this argmax we are going to use Gradient Ascent. In order to do so we first need to find the derivative of the function we want to argmax with respect to both parameters in θ:

$$\begin{aligned}
\frac{\partial}{\partial a}\left[-\sum_{i=1}^{n}(y^{(i)}-ax^{(i)}-b)^2\right] &= -\sum_{i=1}^{n}\frac{\partial}{\partial a}(y^{(i)}-ax^{(i)}-b)^2 \\
&= -\sum_{i=1}^{n}2(y^{(i)}-ax^{(i)}-b)(-x^{(i)}) \\
&= 2\sum_{i=1}^{n}(y^{(i)}-ax^{(i)}-b)(x^{(i)}) \\
\frac{\partial}{\partial b}\left[-\sum_{i=1}^{n}(y^{(i)}-ax^{(i)}-b)^2\right] &= 2\sum_{i=1}^{n}(y^{(i)}-ax^{(i)}-b)
\end{aligned}$$

This first derivative can be plugged into gradient ascent to give our final algorithm:

```
a, b = 0, 0                  # initialize θ
repeat many times:
  gradient_a, gradient_b = 0, 0
  for each training example (x, y):
    diff = y - (a * x + b)
    gradient_a += 2 * diff * x
    gradient_b += 2 * diff
  a += η * gradient_a        # θ += η * gradient
  b += η * gradient_b
```

If you run gradient ascent for enough training (i.e., update) steps, you will find that for linear regression, the maximum likelihood estimators (assuming zero-mean, normally distributed noise between predicted $\hat{Y}$ and actual Y) is equivalent to the mean squared error estimators. Cool!!