Chris Piech
CS109

Lecture #22
Nov 12th, 2018

Gradient Ascent

Maximum Likelihood Refresher

Our first algorithm for estimating parameters is called Maximum Likelihood Estimation (MLE). The central idea behind MLE is to select that parameters (θ) that make the observed data the most likely. The data that we are going to use to estimate the parameters are going to be n independent and identically distributed (IID) datapoints: $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$. A small aside on notation: $x^{(i)}$ represents the ith datum (it is not x raised to the power of i)! This is different from $X^{(i)}$, which random variable for the ith datum.

Likelihood

First we define the likelihood of our data given parameters θ:

$$L(\theta) = \prod_{i=1}^{n} f(X^{(i)} = x^{(i)} | \theta)$$

This is the probability of all of our data. It evaluates to a product because all $X^{(i)}$ are independent. Now we chose the value of θ that maximizes the likelihood function. Formally $\hat{\theta} = \underset{\theta}{\text{argmax}}\, L(\theta)$.

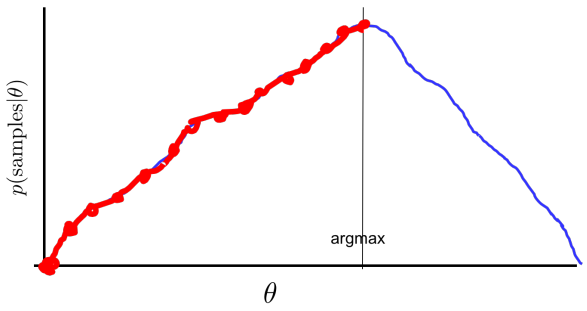
A cool property of argmax is that since log is a monotone function, the argmax of a function is the same as the argmax of the log of the function! That's nice because logs make the math simpler. Instead of using likelihood, you should instead use log likelihood: $LL(\theta)$.

$$LL(\theta) = \log \prod_{i=1}^{n} f(X^{(i)} = x^{(i)} | \theta) = \sum_{i=1}^{n} \log f(X^{(i)} = x^{(i)} | \theta)$$

To use a maximum likelihood estimator, first write the log likelihood of the data given your parameters. Then chose the value of parameters that maximize the log likelihood function. Argmax can be computed in many ways. Most require computing the first derivative of the function.

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:

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

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:

```
Initialize: θ_j = 0 for all 0 ≤ j ≤ m

Repeat many times:
    gradient[j] = 0 for all 0 ≤ j ≤ m

    Calculate all gradient[j]'s based on data
    and current setting of theta

    θ_j += η * gradient[j] for all 0 ≤ j ≤ m
```

Linear Regression Theory

MLE is an algorithm that can be used for any probability model with a derivable likelihood function. As an example lets estimate the parameter θ in a model where there is a random variable Y such that $Y = \theta X + Z$, $Z \sim N(0, \sigma^2)$ and X is an unknown distribution. As a note, linear regression often includes an intercept term. None of the theory changes and MLE for Linear Regression becomes slightly harder to learn.

For each sample, you are told the value of X. Thus θX is a number and $\theta X + Z$ is the sum of a Gaussian and a number. This implies that $Y|X \sim N(\theta X, \sigma^2)$. Our goal is to chose a value of θ that maximizes the probability of IID data: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})$.

We approach this problem by first finding a function for the log likelihood of the data given θ. Then we find the value of θ that maximizes the log likelihood function. To start, use the PDF of a Normal to express the probability of $Y|X, \theta$:

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

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

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

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

Remove positive constant multipliers and terms that don't include θ. We are left with trying to find a value of θ that maximizes:

$$
\hat{\theta} = \underset{\theta}{\operatorname{argmax}} - \sum_{i=1}^{n} (y^{(i)} - \theta x^{(i)})^2
$$

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 θ.

$$
\begin{aligned}
\frac{\partial}{\partial \theta} - \sum_{i=1}^{n} (y^{(i)} - \theta x^{(i)})^2 &= - \sum_{i=1}^{n} \frac{\partial}{\partial \theta} (y^{(i)} - \theta x^{(i)})^2 \\
&= - \sum_{i=1}^{n} 2(y^{(i)} - \theta x^{(i)})(-x^{(i)}) \\
&= \sum_{i=1}^{n} 2(y^{(i)} - \theta x^{(i)})(x^{(i)})
\end{aligned}
$$

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

```
Initialize: θ = 0

Repeat many times:
    gradient = 0

    For each training example (x, y):

        gradient += 2(y - θ x)(x)

    θ += η * gradient
```