Beyond Linear Regression and Gradient Descent

Authors: Shruti Kapur, Bhavna Kaparaju, Ram Kapistalam. (PDF)

To start the class, we touched on "Almost" Linear models (such as Piecewise Linear) and nonlinear models. We then discussed the nature of Universal Approximators and some specific examples of this concept such as Neural Nets. Finally, we looked at three methods to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible.

Universal Approximators

The class first explained the idea of linear regression forms as a subset of universal approximators. Universal approximators are functions or theorems that can produce an output which is an accurate approximation to any desired degree of accuracy of functions in certain familiar spaces of functions. MLPs, polynomial functions and certain neural networks are good examples of such approximators.

Minimizing Cost

The class then further built on the concept of minimizing the cost E(a) = SSE of MSE. We can think of different ways of finding the weights. The first involves linear regression, which gives us a direct solutions with optimal weights (w*).

The second method is the very interesting Newton Raphson method of finding the roots. This method essentially uses the idea that a continuous and differentiable function can be approximated by a straight line tangent to it. This method gives a one step guess to the optimum weights, when we consider that E(w) is quadratic in w and w* is the optimum solution.

Though the class did not delve much into this method, we studied a bit and found two interesting facts associated to it. The first is that the function may not always converge (counter-intuitively). Its convergence theory is for "local" convergence which means we should start close to the root, where "close" is relative to the function we are dealing with. Far away from the root we can have highly nontrivial dynamics. The second interesting fact is that Newton's method will fail in cases where the derivative is zero. When the derivative is close to zero, the tangent line is nearly horizontal and hence may overshoot the desired root (numerical difficulties). Solution: Try another initial point.

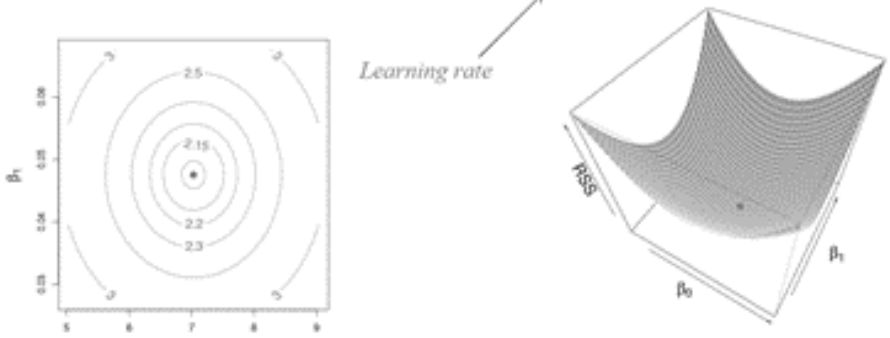
Gradient Descent

The third method and the core focus of the class was the idea of a Gradient Descent. The concept is that we initially guess the optimum weight, call it w0 and iteratively update w by going 'down' the gradient till we reach the optimum weight, w*. This method works even for nonlinear models.

For instance, if the cost function E(w) is quadratic in w, weights can be obtained by incrementally moving down the cost surface in weight space.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E$$

Learning rate



The learning rate is important – too low would cause slow convergence and too high would cause instability. There are variations in the learning rates.

- For instance, an adaptive learning rate involves Choosing a decrease constant d that shrinks the learning rate over time: $\eta(t+1) := \eta(t)/(1+t \times d)$
- Another method is momentum learning by adding a factor of the previous gradient to the weight update for faster updates: $\triangle w_{t+1} := \eta \bigtriangledown J(w_{t+1}) + \alpha \triangle w_t$

There are a variety of gradient descent optimization algorithms in the market today, such as Momentum, Nesterov, Adagrad, Adadelta, RMSprop, Adam and Nadam. Adam is the most popular one – ADAM stands for adaptive Moment Estimation. ADAM computes adaptive learning rates for each parameter. It also keeps an exponentially decaying average of past gradients mt, similar to Momentum.

The class also briefly mentioned the contrast between True Gradient Descent and Stochastic Gradient Descent. One difference mentioned is that True Gradient Descent uses a Batch Algorithm, meaning that it involves all of the training data. The Medium article in "Additional Resources" explains why that is in-depth, but here are a couple reasons that a batch algorithm is used:

- There are less oscillations and noisy steps taken towards the global minima of the loss function because you are updating the parameters by taking the average of all training samples rather than just one sample.

- It allows for a more stable gradient descent convergence and stable error gradient than Stochastic Gradient Descent

Still, there are some disadvantages mentioned related to True Gradient Descent. The most common issues relate to the processing time needed for this method since you are processing the entire training sample. Another common problem with True Gradient Descent is that sometimes a stable error gradient can lead to a local minima rather than the optimal global minima.

Additional Resources

Newton-Raphson Method: https://towardsdatascience.com/newtons-method-the-visual-intuition-12a346f4d89

Andrew Ng on Gradient Descent: https://www.youtube.com/watch?v=rIVLE3condE

Article on Gradient Descent: https://towardsdatascience.com/gradient-descent-algorithms-and-adaptive-learning-rate-adjustment-methods-79c701b086be

Variations of Gradient Descent: https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1