

Lec 10: Gradient Descent, AdaGrad, Newton's Method (CS3780/5780, Sp26)

1 Logistic Regression Recap and Iterative Optimization Procedure

Logistic regression assumption:

$$P_{\mathbf{w}}(Y = +1|x) = \frac{1}{1 + \exp(-\mathbf{w}^\top x)}$$

The MLE estimate is given by

$$\hat{\mathbf{w}}_{\text{MLE}} = \arg \max_{\mathbf{w} \in \mathbb{R}^d} \prod_{i=1}^n P(x_i, y_i) = \arg \min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^n \log \left(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i) \right)$$

If we consider MAP estimate, we have

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg \min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^n \log \left(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i) \right) - \log(P(\mathbf{w}))$$

Unfortunately for us, the above optimization problems don't have a closed form. In general, for many ML models, we need to perform optimization of the form

$$\arg \min_{\mathbf{w} \in \mathbb{R}^d} \ell(\mathbf{w}) = \arg \min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell_i(\mathbf{w})$$

To this end, in this lecture we will consider general optimization algorithms to minimize cost function of form $\ell(\mathbf{w})$ where ℓ is a continuous differentiable function. Specifically, we will consider the so called hill climbing algorithms that start at some initial parameter \mathbf{w}_0 and iteratively update them to hopefully converge to the minimizer of ℓ . Specifically the general scheme we consider is of form:

```
Initialize  $\mathbf{w}_0 \in \mathbb{R}^d$ ,  $t = 0$ , Converged = False
While (Converged == False):
    Pick direction  $\mathbf{s}_t \in \mathbb{R}^d$ 
     $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{s}_t$ 
     $t = t + 1$ 
    If  $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| < \delta$  Then Converged = True
End While
```

In this lecture we will figure out what choices of \mathbf{s}_t are good choices for optimization.

2 Gradient Descent Algorithm

A good mental picture to have to figure out a good direction or step \mathbf{s}_t to take at a given location w_t to move to the next location is to perhaps imagine skiing down a slope or walking down a slope when blindfolded (or under zero visibility). Picture this scenario, would you be able to walk down to the valley of a hill on a foggy day without directly being able to see in front of you (say you were a superhero and aren't worried about bumping into things or tumbling down since you are careful). If you picture this, I bet you will be able to walk down the mountain to the valley. How did you do this?

I bet you walked down in the direction that slopes down at your current location. This is gradient descent. Gradient descent algorithm takes a small step in the direction where the slope points in the steepest direction downwards. This direction can be calculated using the gradient or derivative of the function at the current location. To get a clear picture, let us first start with the one dimensional case first. From the definition of derivative we have for any function f

$$f'(x) = \lim_{s \rightarrow 0} \frac{f(x+s) - f(x)}{s}$$

When this function is smooth enough, from the above definition one can conclude that for small s ,

$$f(x+s) \approx f(x) + f'(x) \cdot s$$

Using this co-ordinate wise of ℓ we have that for small \mathbf{s} , for any coordinate i ,

$$\ell(\mathbf{w}[i] + s[i]) \approx \ell(\mathbf{w}[i]) + s[i] \cdot \frac{\partial}{\partial w[i]} \ell(\mathbf{w})$$

Recall the definition of gradient of a function given by

$$\nabla \ell(\mathbf{w}) = \begin{bmatrix} \frac{\partial}{\partial \mathbf{w}[1]} \ell(\mathbf{w}) \\ \frac{\partial}{\partial \mathbf{w}[2]} \ell(\mathbf{w}) \\ \vdots \\ \frac{\partial}{\partial \mathbf{w}[d]} \ell(\mathbf{w}) \end{bmatrix}$$

Using the above we see that

$$\ell(\mathbf{w} + \mathbf{s}) \approx \ell(\mathbf{w}) + \nabla \ell(\mathbf{w})^\top \mathbf{s}$$

More formally using first order Taylor approximation one can conclude that for small \mathbf{s} ,

$$\ell(\mathbf{w} + \mathbf{s}) = \ell(\mathbf{w}) + \nabla \ell(\mathbf{w})^\top \mathbf{s} + O(\|\mathbf{s}\|^2)$$

From this perspective lets think about what is the direction that we should step in to ensure $\ell(\mathbf{w} + \mathbf{s})$ is as small as we can make it when \mathbf{s} is small enough. Since for small \mathbf{s} , we have that $O(\|\mathbf{s}\|^2)$ is quadratically smaller, we might want to choose \mathbf{s} to be a small step in the direction of $-\nabla \ell(\mathbf{w})$. That is, let us pick $\mathbf{s} = -\eta \nabla \ell(\mathbf{w})$. This yields that gradient descent algorithm as follows:

Initialize $\mathbf{w}_0 \in \mathbb{R}^d$, $t = 0$, Converged = False

While (Converged == False):

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \ell(\mathbf{w}_t)$$

$$t = t + 1$$

If $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| < \delta$ Then Converged = True

End While

The scalar $\eta > 0$ used above is referred to as step size.

A few words of caution about the choice of step size η even when optimizing simple functions:

- If η is too small we will take for ever to converge
- If η is too large we can even diverge.

To see the above, take the function $\ell(w) = (w - 1)^2$ and take $w_0 = 2$. Can you come up with a step size threshold so that η larger than this threshold, will cause gradient descent to diverge? Next think of how having very small step size can also cause trouble in this example.

Some words of wisdom:

1. We can also use step size η_t that varies with time/step t , typically people pick η_t that decreases with time
2. When loss ℓ is convex (Eg. Logistic regression case) then in theory $\eta_t = 1/\sqrt{t}$ works with provable guarantees on convergence.
3. In general, when gradients are on average small, one might want to use larger step size and vice versa.

2.1 Why Does Gradient Descent Work?

Let us get some better intuition of why/when does gradient descent work. To see this, note that for a one dimensional function ℓ , when the second derivative of the loss is bounded, using Taylor theorem one has that for any s ,

$$\ell(w + s) \leq \ell(w) + s \cdot \ell'(w) + \frac{H}{2} s^2$$

for an appropriate smoothness constant H . The analogous vector result tells us that for any vector $\mathbf{s} \in \mathbb{R}^d$,

$$\ell(\mathbf{w} + \mathbf{s}) \leq \ell(\mathbf{w}) + \nabla \ell(\mathbf{w})^\top \mathbf{s} + \frac{H}{2} \|\mathbf{s}\|^2$$

(all we need for this to hold is that for any \mathbf{w} , the eigen values of $\nabla^2 \ell(\mathbf{w})$ are bounded in magnitude by H). Now for gradient descent we have $\mathbf{s}_t = -\eta \nabla \ell(\mathbf{w}_t)$. Hence,

$$\ell(\mathbf{w}_{t+1}) \leq \ell(\mathbf{w}_t) - \eta \|\nabla \ell(\mathbf{w}_t)\|^2 + \frac{H\eta^2}{2} \|\nabla \ell(\mathbf{w}_t)\|^2$$

Hence from this one can conclude that for any step-size $\eta < 1/2H$,

$$\ell(\mathbf{w}_{t+1}) \leq \ell(\mathbf{w}_t) - \frac{\eta}{2} \|\nabla \ell(\mathbf{w}_t)\|^2$$

The above already tells us a few things. First, since $-\frac{\eta}{2}\|\nabla\ell(\mathbf{w}_t)\|^2$ is always negative, gradient descent with steps size smaller than $1/2H$ will ensure that loss function one each step will only decrease. Next, note that how much the loss function falls in each step depends on how large $\|\nabla\ell(\mathbf{w}_t)\|^2$ is. Larger the gradient the more progress we make in that step. This tells us something interesting for the cases when loss function ℓ has only one minima or stationary point, for instance in the convex case. It tells us that either we make good progress or, gradient is small and hence we are already close to convergence.

3 Adagrad

The default gradient descent algorithm uses a step size η or η_t at each step. Let us consider a scenario where gradient descent can have very slow convergence. For this we will consider a 2-dimensional simple quadratic convex problem.

$$\ell(\mathbf{w}) = 0.01 * \mathbf{w}[1]^2 + 2 * \mathbf{w}[2]^2$$

The minimizer of the above function is clearly $\mathbf{w} = \mathbf{0}$. But recall our 1-D example from the previous section. If step-size was small, we would get slow convergence and if step-size is large the method would diverge. Now with that example in mind, think of the above cost function. For any step-size η , we see that if η is not too small, then on first coordinate it can diverge. But in view of this if we make η small, then on the second coordinate, this step-size would take a long time to converge. Thus we see that we are forced into slow convergence for gradient descent with small step-size to avoid diverging on the first coordinate.

The idea behind fixing this is to have a different step size for the various coordinates. Specifically we would like to set step size for each co-ordinate based on square-root of the sum of gradients on each coordinates. This is exactly the Adagrad algorithm. The algorithm is as follows:

Initialize $\mathbf{w}_0 \in \mathbb{R}^d$, $\mathbf{z}_0 = 0$, $t = 0$, Converged = False

While (Converged == False):

$$\mathbf{g}_t = \nabla\ell(\mathbf{w}_t)$$

$$\forall i, \mathbf{z}_{t+1}[i] = \mathbf{z}_t[i] + \mathbf{g}_t[i]^2$$

$$\forall i, \mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta \frac{\mathbf{g}_t[i]}{\sqrt{\mathbf{z}_t[i] + \epsilon}}$$

$$t = t + 1$$

$$\text{If } \|\mathbf{w}_t - \mathbf{w}_{t-1}\| < \delta \text{ Then Converged} = \text{True}$$

End While

Where in the above $\epsilon > 0$ is used to avoid divide by 0. The above algorithm uses the idea that on coordinates where sum of past gradient squares are large we use smaller step-size and when sum of gradient squares are small then stepsize is large thus using different stepsizes on different coordinates.

Now think about what happens when one uses Adagrad on the example given by

$$\ell(\mathbf{w}) = 0.01 * \mathbf{w}[1]^2 + 2 * \mathbf{w}[2]^2$$

4 Newton's Method: Second Order Method

Consider a 45 degree rotated version of the cost function

$$\ell(\mathbf{w}) = 0.01 * \mathbf{w}[1]^2 + 2 * \mathbf{w}[2]^2$$

In this rotated example, the Adagrad trick does not help. each coordinate is identical since its rotated at exactly 45 degree angle making it symmetric on each coordinate.

A general idea one can use is to consider instead of first order Taylor expansion instead use second order one. Again as before lets start with the one dimensional case. In this case, for small s ,

$$\ell(w + s) \approx \ell(w) + \ell'(w) \cdot s + \frac{1}{2} \ell''(w) s^2$$

This can be extended to the vector version by replacing $\ell'(w)$ by gradient and $\ell''(w)$ by a matrix called the Hessian matrix $\nabla^2 \ell(\mathbf{w})$ where

$$\nabla^2 \ell(\mathbf{w})[i, j] = \frac{\partial^2}{\partial \mathbf{w}[i] \partial \mathbf{w}[j]} \ell(\mathbf{w})$$

and $\nabla^2 \ell(\mathbf{w})[i, i] = \frac{\partial^2}{\partial \mathbf{w}[i]^2} \ell(\mathbf{w})$ thus giving the general second order Taylor approximation:

$$\ell(\mathbf{w} + \mathbf{s}) = \ell(\mathbf{w}) + \nabla \ell(\mathbf{w})^\top \mathbf{s} + \frac{1}{2} \mathbf{s}^\top \nabla^2 \ell(\mathbf{w}) \mathbf{s} + O(\|\mathbf{s}\|^3)$$

Hence same as the gradient descent idea, when we are not too far from the minima (Eg. after a gradient descent warm start), one can find \mathbf{s} that minimizes the second order approximation of the function $\ell(\mathbf{w} + \mathbf{s})$ around $\ell(\mathbf{w})$. That is find \mathbf{s} that minimizes

$$\ell(\mathbf{w}) + \nabla \ell(\mathbf{w})^\top \mathbf{s} + \frac{1}{2} \mathbf{s}^\top \nabla^2 \ell(\mathbf{w}) \mathbf{s}$$

To this end, let us take derivative w.r.t. \mathbf{s} and solve for the case when this derivative is 0. This yields:

$$\mathbf{s} = - \left(\nabla^2 \ell(\mathbf{w}) \right)^{-1} \nabla \ell(\mathbf{w})$$

For a quadratic loss, since second order Taylor expansion is exact, think about convergence of Newton's method for such quadratic problems.

Thus Newton's method has the iterates

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \left(\nabla^2 \ell(\mathbf{w}_t) \right)^{-1} \nabla \ell(\mathbf{w}_t)$$

Words of Wisdom:

1. What is the time complexity of each step of Newton's method? Just the matrix if $O(d^2)$ further inverting it is $O(d^{2.8})$
2. For Newton's method to work well we need to be not too far from the minima. Hence typically one starts with few steps of gradient descent as warm start. Newton's method can have stability issues especially when the Hessian matrix is ill conditioned (barely invertible)