CS 419M Introduction to Machine Learning **Spring 2021-22**

Lecture 14-1: Gradiend Descent and Stochastic Gradient Descent Algo

Lecturer: Abir De *Scribe: Lyric Khare*

14.1 Notifications

The following topics are important for machine learning:

1. 25_{th}March - Quiz
2. 16_{th} Mar $+23_{th}$ Mar $+1_{st}$ Apr (maybe) guest lecture from Dr. Ashish Tendulkar @Google
3. From 30_{th} March onwards- In person classes only, **No hybrid mode**
4. Next two weeks (3_{rd} and 4_{th}) week of March- 2 in person tutorial session

14.2 Gradient Descent Algorithm

14.2.1 Statement

In the last lecture; we saw the gradient descent algorithm.

$$\theta_{t+1} = \theta_t - \gamma \left[\frac{dl_\theta}{\theta}\right]_{\theta=\theta_t} \tag{14.1}$$

$$t = 1, 2, 3, \ldots, T \tag{14.2}$$

$$\bar{\theta} = \frac{\sum_{i=1}^{T} \theta_i}{T} \tag{14.3}$$

Now we assume, suppose the true optimum of the loss: if

$$\theta^* = arg(\min_\theta l_\theta)$$

then as $T \to \infty$,

$$l_{\bar{\theta}} - l_{\theta^*} \to 0 \Rightarrow \bar{\theta} \to \theta^*$$

14.2.2 Proof

To prove this let:

$$l_{\bar{\theta}} \equiv l(\bar{\theta})$$

using Jenson inequality,

$$l(\bar{\theta}) = l\left(\frac{\sum_{i=1}^{T} \theta_i}{T}\right) \leq \frac{\sum_{i=1}^{T} l(\theta_i)}{T}$$

Now

$$\begin{aligned}
& l(\bar{\theta}) - l(\theta^*) \\
\leq & \frac{\sum_{i=1}^{T} l(\theta_i)}{T} - l(\theta^*) \\
= & \frac{1}{T}\sum_{i=1}^{T} \left(l(\theta_i) - l(\theta^*)\right) \\
\leq & \frac{1}{T}\sum_{i=1}^{T} (\theta_i - \theta^*)^T \left(\frac{dl(\theta)}{d\theta}\right)_{\theta=\theta_i} \\
\leq & \frac{1}{T}\left(\frac{1}{2\gamma}\|\theta^*\|^2 + \frac{\gamma}{2}\sum_{i=1}^{T} u_i^2\right) \qquad \text{where } u_i = \left(\frac{dl(\theta)}{d\theta}\right)_{\theta=\theta_i} \\
= & \frac{1}{T}\left(\frac{1}{2\gamma}\beta^2 + \frac{\gamma}{2}Tp^2\right) \qquad \|\theta^*\| \leq \beta, u_i \leq p
\end{aligned} \tag{14.4}$$

Now if $\gamma = T^{\alpha}$

$$= \frac{\beta^2}{2T^{1-\alpha}} + \frac{p^2}{2T^{\alpha}}$$

and lowest value of cofficient of $\beta^2 + p^2$ will be at $\alpha = 0.5$
i.e.

$$l(\bar{\theta}) - l(\theta^*) \sim \mathcal{O}\left(max\left(\frac{1}{T^{\alpha}} - \frac{1}{T^{1-\alpha}}\right)\right)$$

14.2.3 Time and Space trade-off

14.2.3.1 Load, run and repeat

In this method an element (instance) of data is loaded in the RAM from HD, then loss is computed for that instance and added to $loss_{prev}$
The loading from HD to RAM increases the computational time.

Pseudo code
% D is dataset, i is an instance

loss=0
for **i** in **D**:
load(i)
loss=loss+ComputeLoss(i)
return loss

14.2.3.2 Load all and then run

In this method all data is first loaded and then loss is computed which decrases the computational time but increases space requirement significacntly.

Pseudo code
% D is dataset, i is an instance

loss=0
load(D)
for **i** in **D**:
loss=loss+ComputeLoss(i)
return loss

or
loss=0
load(D)
loss=ComputeLoss(D)

14.2.3.3 Minibatch GD

This is an optimised method which uses a batch size B where $B \subset D$ to load in the RAM initially and then computes loss. B is chosen as per space availability and then optimized for computational time too.

14.2.4 Summary

- Gradient descent is a helpful algorithm for convex functions
- Gradient descent has to converge at a rate or speed of $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$
- GD is prefered for 100 instances but not for 10^7 instances because computation of the loss on the entire dataset consume a lot of memory (CPU/GPU/RAM) which is a con
- Stochastic GD's pro is it uses one instace to compute gradient at each iteration

- It is faster in most circumstances (for convexfunction) for computation of loss on whole dataset
- In stochastic GD if u_i is small then rate of convergence rate will go down

14.3 Group Details and Individual Contribution

name	roll no.	sections
Lyric Khare	20D170022	14.2.1,14.2.2, 14.2.4
Kulkarni Sneha Santosh	200100090	14.2.3
Nishkarsh Bansal	200010052	14.1