ELE2742 / ENG1467 - PUC-Rio, 2019.1

Lecture 13: Gradient Descent and Stochastic Gradient Descent

DATE

Lecturer: Kanat Tangwongsan *Scribe: Kriangsak Thuiprakhon*

1 Gradient Descent

1.1 Remarks

(i) What the heck is $\eta \nabla f(x)$? The second-order Taylor's expansion :

$$f(y) \approx f(x) + f'(x) \cdot (\vec{y} - \vec{x}) + \frac{1}{2} f''(x)(\vec{y} - \vec{x})^2$$

In $d \geq 2$,

$$f(\vec{y}) \approx f(\vec{x} + \nabla f(\bar{x})^T (\vec{y} - \vec{x}) + \frac{1}{2} (\vec{y} - \vec{x})^T \nabla^2 f(x)(\vec{y} - \vec{x})$$

where $\nabla^2 f(x)$ is Hessian $Hf(k)$. This potentially complex Hf can be approximated by an extremely simple term $\frac{1}{\eta} I$, where I is the identity matrix. That is,

$$\tilde{f}(\vec{y}) = f(\vec{x}) + \nabla f(\bar{x})^T (\vec{y} - \vec{x}) + \frac{1}{2\eta} ||\vec{y} - \vec{x}||_2^2$$

This turns out to be convex for some complex reasons. The question now is how to minimize $\tilde{f}(\vec{y})$?. To do this, by convexity, set$\nabla y \tilde{f}(\vec{y}) = 0$. Then,

$$\nabla y \tilde{f}(\vec{y}) = \nabla f(\vec{x}) + \frac{1}{\eta} (\vec{y} - \vec{x}) = 0 \Longleftrightarrow \vec{y} - \vec{x} = -\frac{1}{\eta} \nabla f(\vec{x})$$

This technique is called ***minimization of local approximations*** where $f + \nabla$ is local approximation and $\frac{1}{2\eta} ||\vec{y} - \vec{x}||$ is proximity.

(ii) In practice, how to choose η_t? Fixing η_t throughout an approximation is not always a good idea. Therefore, a popular heuristic is to have a decaying η_t

(iii) For further refinement, for instance, the function f needs to be *strongly* convex (i.e., the function must "bend").

1.2 Special Case Popular in Machine Learning/ Deep Learning

- minimize a loss function $f(\vec{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\vec{\theta})$
- Example: In linear regression with l_2 loss. Data points $(\vec{x_{i,j}}, y_i)$ where y_i is a label. The loss function is given by,

$$\text{loss} = \sum_{i=1}^{n}(y_i - \vec{x_i}^T\vec{\theta})^2$$

Minimizing this loss function is equivalent to minimizing

$$f(\vec{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \vec{x_i}^T\vec{\theta})^2$$

- Notice that running GD takes $O(nd)$ per iteration. This is too costly, there shall be a way to reduce this.
- Fun fact: $\mathbb{E}_i[f_i(\vec{\theta})] = f_i(\vec{\theta})$ and $\nabla_\theta f_i(\vec{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\vec{\theta})$ This fact gives rise to Stochastic Gradient Descent (SGD) whereby space requirement and time complexity per iteration is reduced.

2 Stochastic Gradient Descent

We can express the algorithm as the following.

Algorithm 1 Stochastic Gradient Descent Algorithm

function GRADIENTDESCENT(f, x_0)
  for $t = 1, 2, \ldots, T-1$ **do**
    $x_t \leftarrow x_{t-1} - \eta_t \nabla \mathbb{E} f_i(x_{t-1})$
  return $\hat{x} = \frac{1}{T}\sum_t x_t$

SGD modifies this so that.

Algorithm 2 Stochastic Gradient Descent Algorithm

function GRADIENTDESCENT(f, x_0)
  for $t = 1, 2, \ldots, T-1$ **do**
    choose an i at random or round robin.
    $x_t \leftarrow x_{t-1} - \eta_t \nabla f_i(x_{t-1})$
  return $\hat{x} = \frac{1}{T}\sum_t x_t$

Theorem 2.1. *Let x^* be the minimiser of $f :^n \to$. If f is convex and differentiable and satisfies $\|\nabla f_i(x)\|_2 \le G$ for all $x \in^n$, then setting $T = \frac{G^2}{\epsilon^2}\|x_0 - x^*\|_2^2$ and $\eta_t = \eta = \frac{\|x_0 - x^*\|}{G\sqrt{T}}$ gives $\mathbb{E} f(\hat{x}) \le f(x^*) + \epsilon$.*

This is the same old proof. The interesting bits are:

Claim 2.2. *Let* $\Phi_t = \frac{1}{2\eta}\|x_t - x^*\|_2^2$. *Then,* $\mathbb{E}\left[f(\vec{x_t} + (\Phi_{t+1} - \Phi_t)\right] \leq f(x^*) + \frac{1}{2}\eta G^2$

Condition $\mathbb{E}$ on the history until iteration that produced $\vec{x_t}$

$$\begin{aligned}
\mathbb{E}[f(\vec{x_t}) + \Phi_{t+1} - \Phi_t] &\leq \mathbb{E}\left[f(\vec{x_t}) + \frac{1}{2}\eta\left(||\underbrace{\vec{x_{t+1}} - \vec{x_t}}_{\Delta x}||_2^2 + 2\Delta x^T(\vec{x_t} - x^*)\right)\right] \\
&\leq \mathbb{E}[f(\vec{x_t}] + \frac{1}{2}\eta G^2 - \nabla f(\vec{x})^T(\vec{x_t} - x^*) \\
&= \frac{1}{2}\eta G^2 + \underbrace{f(\vec{x_t}) + \nabla f(\vec{x})^T(x^* + \vec{x_t}}_{\leq f(x^*))}
\end{aligned}$$

Note:

$$\begin{aligned}
&\Delta x = -\eta f_i(\vec{x_t}) \\
&\quad \Rightarrow ||\Delta x||_2^2 \leq \eta^2 G^2 \\
&\quad \Rightarrow \mathbb{E}_i[\Delta x] = -\eta\mathbb{E}[\Delta_{f_i}(\vec{x_t}] = -\eta\Delta f(\vec{x_t})
\end{aligned}$$

Other tricks used to implement SGD:

- $\mathbb{E}_i[f_i(\vec{x} = \boldsymbol{f}_i(x)$ but potentially <u>not</u> concentrated. Therefore, This potentially does not converge as fast as performing the full gradient.
- To improve further, Pick a "batch" of indices. Say, $B \leq [n]$ and use an update rule such that

$$\vec{x_t} = \vec{x_{t-1}} - \frac{\eta}{|B|}\sum_{i\in B}\nabla f_i(\vec{x_{t-1}}).$$

  Note: $\mathbb{E}\left[\frac{1}{|B|}\cdot\sum_{i\in B}\nabla f_i(\vec{x})\right] = \frac{1}{|B|}\sum_{i\in B}\nabla f_i(\vec{x}) = \nabla f(\vec{x})$

By doing this, minibatch sampling, the variance is reduced by about $\frac{1}{|B|}$ and the cost per iteration shrinks down to $O(|B|\cdot d)$. This is considered a good compromise in practice since the convergence rate, (i.e.,$f(\hat{x}) - f(x^*)$), is $O\left(\sqrt{\frac{|B|}{T} + \frac{|B|}{T}}\right)$ as opposed to $O\left(\frac{1}{T}\right)$ in a normal GD algorithm.