Course: Comtemporary Algorithms T.II/2019-20

Lecture 17: Gradient Descent & SGD

4 March 2020

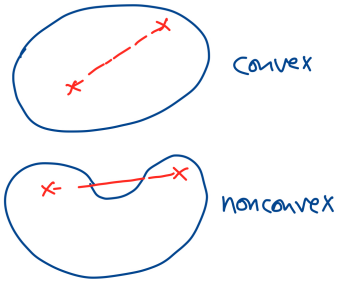
Lecturer: Dr. Kanat Tangwongsan *Scribe: Kriangsak T. & Apivich H.*

1 Convex Functions

First, we have to define the notion of convexity. The formal definition of convex sets are as follows.

Definition 1.1. A set K is *convex* if for all $x, y \in K$, $\lambda x + (1-\lambda)y \in K$ for all $\lambda \in [0,1]$.

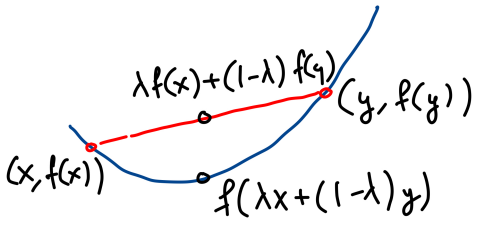
Geometrically, a set is convex if any line between any two points in the set does not leave the set. Note that in the definition above, $\lambda x + (1-\lambda)y$ can be thought of as a line connecting two points x and y together.



From here, we can define convex functions as follows.

Definition 1.2. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for any $x, y \in \mathbb{R}^n$, $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all $\lambda \in [0,1]$.

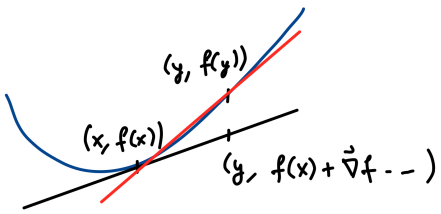
We can imagine this in 2D space. Geometrically, if you draw a line from $(x, f(x))$ to $(y, f(y))$, then the line will always be "above" the function. We can also say that the region above the function (or the set $\{(x,y) \in \mathbb{R}^2 : f(x) \leq y\}$ if in 2D) is a convex set.



A useful result about differentiable convex functions is the following.

Fact 1.3. *For differentiable convex function f, $f(y) \geq f(x) + \nabla f(x)^T(y - x)$.*

This is simply due to the fact that for convex function, the graph always will be curving away, so approximating the function using a straight line will always give a lower value than in reality. Note that here we have used the symbol ∇ (or the gradient operator), which is the way to generalise the derivative to higher dimensions.



We can also define something called the Hessian matrix as the following.

Definition 1.4. For twice-differentiable function f, we can define the *Hessian matrix* as Hf, where

$$(Hf)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

We can also determine the convexity of a function by its Hessian matrix.

Fact 1.5. *A twice-differentiable function f is convex if and only if its Hessian matrix Hf is positive semidefinite for all x (i.e. the eigenvalue of Hf is always positive).*

Finally, we will define the notion of a function being λ-Lipschitz.

Definition 1.6. A function $f : \mathbb{R}^n \to \mathbb{R}$ is λ-Lipschitz if for all x, in $\mathbb{R}^n$, $\|f(x) - f(y)\| \leq \lambda\|x - y\|_2$.

Intuitively, λ-Lipschitz functions are functions which doesn't change too rapidly. If you travel some distance d away from a point, then you would expect the function to only change by λd amound at most (we won't use this definition in this lecture, but we'll define it anyway).

2 Gradient Descent

2.1 The Problem

After much definitions, we will now define the problem that we want to solve. In this lecture, we are given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, and we would like to find some $x^* \in \mathbb{R}^n$ which will give the lowest possible value of $f(x^*)$.

From early calculus classes, this is a simple task as we know the minimum point must be such that $\nabla f(x^*) = 0$. However, this is not always easy to find. An example of such is if we consider the function

$$f(x) = \frac{1}{2}x^T Ax - b^T x,$$

we can see that $\nabla f(x) = Ax - b$, and to find where $\nabla f(x) = 0$ means finding the solution to $Ax = b$, which is not necessarily easy. We will therefore come up with a different method to do this.

2.2 Gradient Descent Algorithm

The idea is we can try to "walk" to the solution. We can start with some initial guess of the minimum, and pick the next value we should walk to. We see that given some value, we should walk to the point which is lower down. This corresponds to walking in the direction of $-\nabla f$. We can keep walking until we reach the minimal point.

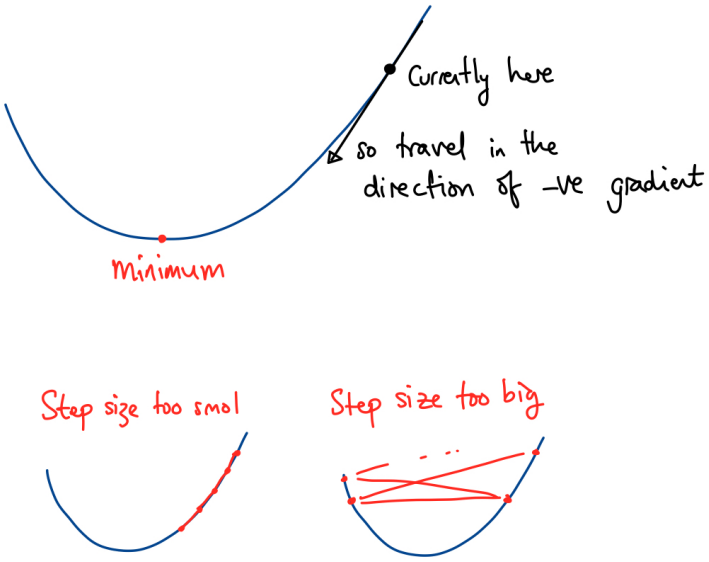
We can express the algorithm as the following.

Algorithm 1 Gradient Descent Algorithm

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

The algorithm requires the function f which we want to optimise. We also need some initial starting point x_0 which starts off the iterations. In the end, we take the mean of all the steps that the algorithm walks to.

Note that we also have the parameter η_t which is known as the step size. This determines how big each of the steps will be. There is no good rule for what it should be, but a step size too slow and the solution converges slowly, but a step size too big and you will keep "stepping over" the actual solution.



2.3 Analysis

We can now try to quantitatively describe the performance of the Gradient Descent algorithm. We make the following claim about the algorithm.

Theorem 2.1. *Let x^* be the minimiser of $f : \mathbb{R}^n \to \mathbb{R}$. If f is convex and differentiable and satisfies $\|\nabla f(x)\|_2 \leq G$ for all $x \in \mathbb{R}^n$, 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 $f(\hat{x}) \leq f(x^*) + \epsilon$.*

Proof. To prove the theorem, we will use the fact (which will be proven later in Lemma 2.2) that

$$\sum_t f(x_t) \leq T f(x^*) + \frac{1}{2}G^2 T\eta + \frac{1}{2\eta}\|x_0 - x^*\|_2^2.$$

We can see that

$$\begin{aligned}
f(\hat{x}) &= f\left(\frac{1}{T}\sum_t x_t\right) \\
&\leq \frac{1}{T}\sum_t f(x_t) && \text{by convexity of } f \\
&\leq f(x^*) + \frac{1}{2}G^2\eta + \frac{1}{2T\eta}\|x_0 - x^*\|_2^2 && \text{by above} \\
&= f(x^*) + \frac{G}{\sqrt{T}}\|x_0 - x^*\|_2 && \text{by our choice of } \eta \\
&= f(x^*) + G\|x_0 - x^*\|_2\sqrt{\frac{\epsilon^2\|x_0 - x^*\|_2^2}{G^2}} && \text{by our choice of } G \\
&= f(x^*) + \epsilon
\end{aligned}$$

which proves our theorem. □

An interesting observation from the proof above is that we can see that from Line 4 of the maths above, $f(\hat{x}) \leq f(x^*) + \frac{G}{\sqrt{T}}\|x_0 - x^*\|_2$, i.e. $f(\hat{x}) - f(x^*) \leq O(\frac{1}{\sqrt{T}})$. This roughly shows that the error in our solution goes down proportional to $\frac{1}{\sqrt{T}}$.

We now have one more thing that we need to prove from above.

Lemma 2.2. *Let f and x^* be defined as we have done in Theorem 3.1. Then,*

$$\sum_t f(x_t) \leq Tf(x^*) + \frac{1}{2}G^2T\eta + \frac{1}{2\eta}\|x_0 - x^*\|_2^2.$$

Proof. Let $\Phi_t = \frac{1}{2\eta}\|x_t - x^*\|_2^2$. Then, we can see that

$$\begin{aligned}
f(x_t) + \Phi_{t+1} - \Phi_t &= f(x_t) + \frac{1}{2\eta}\left(\|x_{t+1} - x^*\|_2^2 - \|x_t - x^*\|_2^2\right) \\
&= f(x_t) + \frac{1}{2\eta}\left((x_{t+1} - x^*)^T(x_{t+1} - x^*) - (x_t - x^*)^T(x_t - x^*)\right).
\end{aligned}$$

Let $\Delta x_t = x_{t+1} - x_t$ and $y = x_t - x^*$. Then,

$$\begin{aligned}
f(x_t) + \Phi_{t+1} - \Phi_t &= f(x_t) + \frac{1}{2\eta}\left((\Delta x_t + y)^T(\Delta x_t + y) - y^Ty\right) \\
&= f(x_t) + \frac{1}{2\eta}\left(\|\Delta x_t\|^2 + 2\Delta x^Ty\right).
\end{aligned}$$

However, we know that $\Delta x_t = x_{t+1} - x_t = -\eta \nabla f(x_t)$, and so

$$\begin{aligned} f(x_t) + \Phi_{t+1} - \Phi_t &= f(x_t) + \frac{1}{2\eta}\left(\eta^2 \|\nabla f(x_t)\|^2 + 2\nabla f(x_t)^T y\right) \\ &\leq \frac{1}{2}G^2\eta + \underbrace{f(x_t) - 2f(x_t)^T(x_t - x^*)}_{\leq f(x^*) \text{ by the property of convex functions}} \\ &\leq \frac{1}{2}G^2 + f(x^*) \end{aligned}$$

We have shown that $f(x_t) + \Phi_{t+1} - \Phi_t \leq \frac{1}{2}G^2 + f(x^*)$. From this, we can therefore see that

$$\begin{aligned} \sum_t f(x_t) &= \sum_{t=1}^{T}[f(x_t) + \Phi_t - \Phi_{t-1}] - \Phi_T + \Phi_0 \\ &\leq \sum_{t=1}^{T}[f(x_t) + \Phi_t - \Phi_{t-1}] + \frac{1}{2\eta}\|x_0 - x^*\|_2^2 \quad \text{definition of } \Phi_t, \text{ and since } \Phi_T \geq 0 \\ &\leq Tf(x^*) + \frac{1}{2}TG^2 + \frac{1}{2\eta}\|x_0 - x^*\|_2^2 \quad \text{from above} \end{aligned}$$

which proves our lemma. □

2.4 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,

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

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").

3 Stochastic Gradient Descent

3.1 Special Case Popular in Machine Learning/Deep Learning

- In machine learning, we will often want to minimise a loss function $f(\vec{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\vec{\theta})$ with $\vec{\theta}$ being some model parameter.
- 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.

3.2 SGD Algorithm

We can express the algorithm as the following.

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 $\hat{x} = \frac{1}{T}\sum_t x_t$

We can also state a similar bound proof as before.

Theorem 3.1. *Let x^* be the minimiser of $f : \mathbb{R}^n \to \mathbb{R}$. If f is convex and differentiable and satisfies $\|\nabla f_i(x)\|_2 \leq G$ for all $x \in \mathbb{R}^n$, 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 $\mathbb{E}f(\hat{x}) \leq f(x^*) + \epsilon$.*

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

Claim 3.2. *Let $\Phi_t = \frac{1}{2\eta}\|x_t - x^*\|_2^2$. Then, $\mathbb{E}\Big[f(\vec{x_t} + (\Phi_{t+1} - \Phi_t)\Big] \leq f(x^*) + \frac{1}{2}\eta G^2$*

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

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

Note:

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

Other tricks used to implement SGD:

- $\mathbb{E}_i[f_i(\vec{x})] = f_i(x)$ but potentially <u>not</u> concentrated (i.e. high variance). 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: $\mathbb{E}\Big[\frac{1}{|B|} \cdot \sum_{i \in B}\nabla f_i(\vec{x})\Big] = \frac{1}{|B|}\sum_{i \in B}\nabla f_i(\vec{x}) = \nabla f(\vec{x})$

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 $O\Bigg(\sqrt{\frac{|B|}{T} + \frac{|B|}{T}}\Bigg)$as opposed to $O\big(\frac{1}{T}\big)$ in a normal GD algorithm.