Outline

Mathematical Background
- Lecture 4: Intro to Optimization
- Lecture 5: Gradient Descent

Lecture 5: Gradient Descent
- Gradient Descent
  - Descent Direction
  - Step Size
  - Convergence
- Stochastic Gradient Descent
  - Difference between GD and SGD
  - Why does SGD work?
Gradient Descent

The algorithm:

\[ x^{(t+1)} = x^{(t)} - \alpha^{(t)} \nabla f(x^{(t)}), \quad t = 0, 1, 2, \ldots, \]

where \( \alpha^{(t)} \) is called the **step size**.
Why is the direction $-\nabla f(x)$?

- Recall (Lecture 4): If $x^*$ is optimal, then
  \[ \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[ f(x^* + \epsilon d) - f(x^*) \right] = \nabla f(x^*)^T d \geq 0, \forall d \]
  \[ \implies \nabla f(x^*)^T d \geq 0, \forall d \]

- But if $x(t)$ is not optimal, then we want
  \[ f(x(t) + \epsilon d) \leq f(x(t)) \]

- So,
  \[ \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[ f(x(t) + \epsilon d) - f(x(t)) \right] = \nabla f(x(t))^T d \leq 0, \text{ for some } d \]
  \[ \implies \nabla f(x(t))^T d \leq 0 \]
Descent Direction

Pictorial illustration:
- $\nabla f(x)$ is perpendicular to the contour.
- A search direction $d$ can either be on the positive side $\nabla f(x)^T d \geq 0$ or negative side $\nabla f(x)^T d < 0$.
- Only those on the negative side can reduce the cost.
- All such $d$’s are called the descent directions.
The Steepest $d$

Previous slide: If $x^{(t)}$ is not optimal yet, then some $d$ will give

$$\nabla f(x^{(t)})^T d \leq 0.$$ 

So, let us make $\nabla f(x^{(t)})^T$ as negative as possible.

$$d^{(t)} = \arg\min_{\|d\|_2=\delta} \nabla f(x^{(t)})^T d,$$

We need $\delta$ to control the magnitude; Otherwise $d$ is unbounded.

The solution is

$$d^{(t)} = -\nabla f(x^{(t)})$$

Why? By Cauchy Schwarz,

$$\nabla f(x^{(t)})^T d \geq -\|\nabla f(x^{(t)})\|_2 \|d\|_2.$$ 

Minimum attained when $d = -\nabla f(x^{(t)})$.

Set $\delta = \|\nabla f(x^{(t)})\|_2$. 
Steepest Descent Direction

**Pictorial illustration:**
- Put a ball surrounding the current point.
- All $d$’s inside the ball are feasible.
- Pick the one that minimizes $\nabla f(x)^T d$.
- This direction must be parallel (but opposite sign) to $\nabla f(x)$.
Step Size

The algorithm:

$$x^{(t+1)} = x^{(t)} - \alpha^{(t)} \nabla f(x^{(t)}), \quad t = 0, 1, 2, \ldots,$$

where $\alpha^{(t)}$ is called the step size.

1. **Fixed step size**
   
   $$\alpha^{(t)} = \alpha.$$

2. **Exact line search**
   
   $$\alpha^{(t)} = \arg\min_{\alpha} f \left( x^{(t)} + \alpha d^{(t)} \right),$$

   E.g., if $f(x) = \frac{1}{2} x^T H x + c^T x$, then
   
   $$\alpha^{(t)} = -\frac{\nabla f(x^{(t)})^T d^{(t)}}{d^{(t)T} H d^{(t)}}.$$

3. **Inexact line search**:

   Amijo / Wolfe conditions. See Nocedal-Wright Chapter 3.1.
**Convergence**

Let $x^*$ be the global minimizer. Assume the followings:

- Assume $f$ is twice differentiable so that $\nabla^2 f$ exist.
- Assume $0 \preceq \lambda_{\text{min}} I \preceq \nabla^2 f(x) \preceq \lambda_{\text{max}} I$ for all $x \in \mathbb{R}^n$
- Run gradient descent with exact line search.

Then, (Nocedal-Wright Chapter 3, Theorem 3.3)

\[
f(x^{(t+1)}) - f(x^*) \leq \left( 1 - \frac{\lambda_{\text{min}}}{\lambda_{\text{max}}} \right)^2 \left( f(x^{(t)}) - f(x^*) \right)
\]

\[
\leq \left( 1 - \frac{\lambda_{\text{min}}}{\lambda_{\text{max}}} \right)^4 \left( f(x^{(t-1)}) - f(x^*) \right)
\]

\[
\leq \vdots
\]

\[
\leq \left( 1 - \frac{\lambda_{\text{min}}}{\lambda_{\text{max}}} \right)^{2t} \left( f(x^{(1)}) - f(x^*) \right).
\]

Thus, $f(x^{(t)}) \to f(x^*)$ as $t \to \infty$. 
Understanding Convergence

- Gradient descent can be viewed as **successive approximation**.
- Approximate the function as

  \[ f(x^t + d) \approx f(x^t) + \nabla f(x^t)^T d + \frac{1}{2\alpha} \|d\|^2. \]

- We can show that the \(d\) that minimizes \( f(x^t + d) \) is \( d = -\alpha \nabla f(x^t) \).
- This suggests: Use a **quadratic function** to locally approximate \( f \).
- Converge when curvature \( \alpha \) of the approximation is not too big.
Advice on Gradient Descent

- Gradient descent is useful because
  - Simple to implement (compared to ADMM, FISTA, etc)
  - Low computational cost per iteration (no matrix inversion)
  - Requires only first order derivative (no Hessian)
  - Gradient is available in deep networks (via back propagation)
- Most machine learning has built-in (stochastic) gradient descents
- Welcome to implement your own, but you need to be careful
  - Convex non-differentiable problems, e.g., $\ell_1$-norm
  - Non-convex problem, e.g., ReLU in deep network
  - Trap by local minima
  - Inappropriate step size, a.k.a. learning rate
- Consider more “transparent” algorithms such as CVX when
  - Formulating problems. No need to worry about algorithm.
  - Trying to obtain insights.
Outline

Mathematical Background

- Lecture 4: Intro to Optimization
- Lecture 5: Gradient Descent

Lecture 5: Gradient Descent

- Gradient Descent
  - Descent Direction
  - Step Size
  - Convergence
- Stochastic Gradient Descent
  - Difference between GD and SGD
  - Why does SGD work?
Most loss functions in machine learning problems are separable:

\[
J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(g_{\theta}(x^n), y^n) = \frac{1}{N} \sum_{n=1}^{N} J_n(\theta).
\]  

(1)

For example,

- **Square-loss:**

\[
J(\theta) = \sum_{n=1}^{N} (g_{\theta}(x^n) - y^n)^2
\]

- **Cross-entropy loss:**

\[
J(\theta) = - \sum_{n=1}^{N} \left\{ y^n \log g_{\theta}(x^n) + (1 - y^n) \log (1 - g_{\theta}(x^n)) \right\}
\]

- **Logistic loss:**

\[
J(\theta) = \sum_{n=1}^{N} \log(1 + e^{-y^n \theta^T x^n})
\]
**Full Gradient VS Partial Gradient**

Vanilla **gradient descent**:

\[
\theta^{t+1} = \theta^t - \eta^t \nabla J(\theta^t) .
\]  

(2)

The full gradient of the loss is

\[
\nabla J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla J_n(\theta)
\]  

(3)

**Stochastic gradient descent**:

\[
\nabla J(\theta) \approx \frac{1}{|B|} \sum_{n \in B} \nabla J_n(\theta)
\]  

(4)

where \( B \subseteq \{1, \ldots, N\} \) is a random subset. \(|B| = \text{batch size.} \)
SGD Algorithm

Algorithm (Stochastic Gradient Descent)

1. Given \( \{(x^n, y^n) \mid n = 1, \ldots, N\} \).
2. Initialize \( \theta \) (zero or random)
3. For \( t = 1, 2, 3, \ldots \)
   - Draw a random subset \( B \subseteq \{1, \ldots, N\} \).
   - Update
     \[
     \theta^{t+1} = \theta^t - \eta^t \frac{1}{|B|} \sum_{n \in B} \nabla J_n(\theta) \tag{5}
     \]
   - If \( |B| = 1 \), then use only one sample at a time.
   - The approximate gradient is **unbiased**: (See Appendix for Proof)
     \[
     \mathbb{E} \left[ \frac{1}{|B|} \sum_{n \in B} \nabla J_n(\theta) \right] = \nabla J(\theta).
     \]
Interpreting SGD

- Just showed that the SGD step is unbiased:

$$\mathbb{E} \left[ \frac{1}{|B|} \sum_{n \in B} \nabla J_n(\theta) \right] = \nabla J(\theta).$$

- Unbiased gradient implies that each update is

  $$\text{gradient} + \text{zero-mean noise}$$

- Step size: SGD with constant step size does not converge.
- If $\theta^*$ is a minimizer, then $J(\theta^*) = \frac{1}{N} \sum_{n=1}^{N} J_n(\theta^*) = 0$. But

$$\frac{1}{|B|} \sum_{n \in B} J_n(\theta^*) \neq 0,$$

  since $B$ is a subset.

- Typical strategy: Start with large step size and gradually decrease: $\eta^t \to 0$, e.g., $\eta^t = t^{-a}$ for some constant $a$. 
Perspectives of SGD

Classical optimization literature have the following observations.

- Compared to GD in **convex** problems:
  - SGD offers a **trade-off** between accuracy and efficiency
  - More iterations
  - Less gradient evaluation per iteration
  - Noise is a by-product

Recent studies of SGD for **non-convex** problems found that

- SGD for training deep neural networks works
- SGD finds solution faster
- SGD find a better local minima
- Noise matters
GD compared to SGD

Convex

Non-convex
Smoothing the Landscape

Analyzing SGD is an active research topic. Here is one by Kleinberg et al. (https://arxiv.org/pdf/1802.06175.pdf ICML 2018)

- The SGD step can be written as GD + noise:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta (\nabla f(\mathbf{x}^t) + \mathbf{w}^t)$$

$$= \mathbf{x}^t - \eta \nabla f(\mathbf{x}^t) - \eta \mathbf{w}^t.$$

$$\overset{\text{def}}{=} \mathbf{y}^t$$

- $\mathbf{y}^t$ is the “ideal” location returned by GD.
- Let us analyze $\mathbf{y}^{t+1}$:

$$\mathbf{y}^{t+1} \overset{\text{def}}{=} \mathbf{x}^{t+1} - \eta \nabla f(\mathbf{x}^{t+1})$$

$$= (\mathbf{y}^t - \eta \mathbf{w}^t) - \eta \nabla f(\mathbf{y}^t - \eta \mathbf{w}^t)$$

- Assume $\mathbb{E}[\mathbf{w}] = 0$, then

$$\mathbb{E}[\mathbf{y}^{t+1}] = \mathbf{y}^t - \eta \nabla \mathbb{E}[f(\mathbf{y}^t - \eta \mathbf{w}^t)]$$
Smoothing the Landscape

Let us look at \( \mathbb{E}[f(y^t - \eta w^t)] \):

\[
\mathbb{E}[f(y - \eta w)] = \int f(y - \eta w)p(w) \, dw,
\]

where \( p(w) \) is the distribution of \( w \).

\( \int f(y - \eta w)p(w) \, dw \) is the convolution between \( f \) and \( p \).

\( p(w) \geq 0 \) for all \( w \), so the convolution always smooths the function.

Learning rate controls the smoothness

Too small: Under-smooth. You have not yet escaped from bad local minimum.

Too large: Over-smooth. You may miss a local minimum.
Smoothing the Landscape
Reading List

Gradient Descent


Stochastic Gradient Descent

Appendix
Proof of Unbiasedness of SGD gradient

Lemma

If \( n \) is a random variable with uniform distribution over \( \{1, \ldots, N\} \), then

\[
\mathbb{E} \left[ \frac{1}{|B|} \sum_{n \in B} \nabla J_n(\theta) \right] = \nabla J(\theta).
\]

Denote the density function of \( n \) as \( p(n) = 1/N \). Then,

\[
\mathbb{E} \left[ \frac{1}{|B|} \sum_{n \in B} \nabla J_n(\theta) \right] = \frac{1}{|B|} \sum_{n \in B} \mathbb{E} [\nabla J_n(\theta)] = \frac{1}{|B|} \sum_{n \in B} \left\{ \sum_{n=1}^{N} J_n(\theta) p(n) \right\}
\]

\[
= \frac{1}{|B|} \sum_{n \in B} \left\{ \frac{1}{N} \sum_{n=1}^{N} J_n(\theta) \right\} = \frac{1}{|B|} \sum_{n \in B} \nabla J(\theta) = \nabla J(\theta).
\]
Q&A 1: What is momentum method?

- The momentum method was originally proposed by Polyak (1964).
- Momentum method says:

\[ x^{t+1} = x^t - \alpha \left[ \beta g^{t-1} + (1 - \beta) g^t \right], \]

where \( g^t = \nabla f(x^t) \), and \( 0 < \beta < 1 \) is the damping constant.
- Momentum method can be applied to both gradient descent and stochastic gradient descent.
- A variant is the Nesterov accelerated gradient (NAG) method (1983).
- Importance of NAG is elaborated by Sutskever et al. (2013).
- The key idea of NAG is to write \( x^{t+1} \) as a linear combination of \( x^t \) and the span of the past gradients.
- Yurii Nesterov proved that such combination is the best one can do with first order methods.
Q&A 1: What is momentum method?

Here are some references on momentum method.

- Sutskever et al. (2013), “On the importance of initialization and momentum in deep learning”,
- UIC Lecture Note
  https://www2.cs.uic.edu/~zhangx/teaching/agm.pdf
- Yurii Nesterov, “Introductory Lectures on Convex Optimization”, 2003. (See Assumption 2.1.4 and discussions thereafter)
  https://distill.pub/2017/momentum/
Q&A 2: With exact line search, will we get to a minimum in one step?

- No. Exact line search only allows you to converge faster. It does not guarantee convergence in one step.
- Here is an example. The function is called the rosenbrock function.
Q&A 3: Any example of gradient descent?

- Consider the loss function
  \[ J(\theta) = \|A\theta - y\|^2 \]

- Then the gradient is
  \[ \nabla J(\theta) = 2A^T(A\theta - y) \]

- So the gradient descent step is
  \[ \theta^{t+1} = \theta^t + \eta 2A^T(A\theta^t - y). \]

- Since this is a quadratic equation, you can find the exact line search step size (assume \( d = -\nabla f(\theta) \)):
  \[ \eta = -\|d\|^2/(d^T A^T Ad). \]
Q&A 4: In finding the steepest direction, why is $\delta$ unimportant?

- The constraint $\|d\| = \delta$ is necessary for minimizing $\nabla f(x)^T d$. Without the constraint, this problem is unbounded below and the solution is $-\infty$ times whatever direction $d$ that lives on the negative half plane of $\nabla f$.

- For any $\delta$, the solution (according to Cauchy Schwarz inequality), is

$$d = -\delta \frac{\nabla f(x)}{\|\nabla f(x)\|}.$$ 

You can show that this $d$ minimizes $\nabla f(x)^T d$ and satisfies $\|d\| = \delta$.

- Now, if we use this $d$ in the gradient descent step, the step size $\alpha$ will compensate for the $\delta$.

- So we can just choose $\delta = 1$ and the above derivation will still work.
Q&A 5: What is a good batch size for SGD?

There is no definite answer. Generally you need to look at the validation curve to determine if you need to increase/decrease the mini-batch size. Here are some suggestions in the literature.

  
  [batch size] is typically chosen between 1 and a few hundreds, e.g. [batch size] = 32 is a good default value, with values above 10 taking advantage of the speedup of matrix-matrix products over matrix-vector products.

  
  The presented results confirm that using small batch sizes achieves the best training stability and generalization performance, for a given computational cost, across a wide range of experiments. In all cases the best results have been obtained with batch sizes $m = 32$ or smaller, often as small as $m = 2$ or $m = 4$. 