Basics on convex optimization and gradient descent

1 Convex set

$C \subset \mathbb{R}^n$ such that $x, y \in C \implies tx + (1-t)y \in C$ for all $0 \leq t \leq 1$.

Convex combination of $x_1, \ldots, x_k \in \mathbb{R}^n$: any linear combination $\theta_1 x_1 + \cdots + \theta_k x_k$ with $\theta_i \geq 0, i = 1, \ldots, k$, and $\sum_{i=1}^k \theta_i = 1$ Convex hull of a set C, conv(C), is all convex combinations of elements.

Show that following sets are convex sets:

- Trivial ones: empty set, point, line
- Norm ball: $\{\mathbf{x} : \|\mathbf{x}\| \leq r\}$, for given norm $\|\cdot\|$, radius r
- Hyperplane: $\{\mathbf{x} : \mathbf{a}^T\mathbf{x} = \mathbf{b}\}$, for given a, b
- Halfspace: $\mathbf{x} : \mathbf{a}^T\mathbf{x} \leq b$
- Affine space: $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$, for given A, b

What is a convex function?

Why convexity? as, we can broadly understand and solve convex optimization problems. It is in general difficult to get a general solution scheme for a nonconvex problem.

1.1 Important properties of convex set

Separating hyperplane theorem: two disjoint convex sets have a separating between hyperplane them

Mathematically: if C, D are nonempty convex sets with $C \cap D = \phi$, then there exists a, b such that

$C \subset \{\mathbf{x} : \mathbf{a}^T\mathbf{x} \leq \mathbf{b}\} \quad D \subset \{\mathbf{x} : \mathbf{a}^T\mathbf{x} \geq \mathbf{b}\}$

Supporting hyperplane theorem: a boundary point of a convex set has a supporting hyperplane passing through it

Mathematically: if C is a nonempty convex set, and $\mathbf{x}_0 \in bd(C)$, then there exists $\mathbf{a}$ such that

$C \subset \{\mathbf{x} : \mathbf{a}^T\mathbf{x} \leq \mathbf{a}^T\mathbf{x}_0\}$

1.2 Operations preserving convexity

- Intersection: the intersection of convex sets is convex
- Scaling and translation: if C is convex, then $aC + b = \{ax + b : x \in C\}$ is convex for any a, b

- Affine images and preimages: if $f(x) = Ax+b$ and C is convex then $f(C) = \{f(x) : x \in C\}$ is convex, and if D is convex then $f^{-1}(D) = \{x : f(x) \in D\}$ is convex

Many other operations also preserve convexity. It is good to check whether first property of convexity holds or not.

2 Convex functions

$f : \mathbb{R}^n \to R$ such that $\text{dom}(f) \subseteq \mathbb{R}^n$ convex (The domain (dom) of a function is the set of all possible inputs for the function.), and $f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$ for $0 \leq t \leq 1$ and all $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$

In words, function lies below the line segment joining $(\mathbf{x}, f(\mathbf{x})$ and $(\mathbf{y}, f(\mathbf{y}))$

Concave function: opposite inequality above, so that f concave $\iff -f$ convex

Important terminologies:

- Strictly convex: $f(t\mathbf{x}+(1-t)\mathbf{y}) < tf(\mathbf{x})+(1-t)f(\mathbf{y})$ for $0 < t < 1$. In words, f is convex and has greater curvature than a linear function
- Strongly convex with parameter $m > 0 : f - m\|\mathbf{x}\|^2/2$ is convex. In words, f is at least as convex as a quadratic function

Note: strongly convex $\implies$ strictly convex $\implies$ convex

2.1 Examples of convex functions

- Univariate functions:
  - Exponential function: e^{ax} is convex for any a over R
  - Power function: x^a is convex for $a \geq 1$ or $a \leq 0$ over R^+ (nonnegative reals)
  - Power function: x^a is concave for $0 \leq a \leq 1$ over R^+
  - Logarithmic function: log x is concave over R++
- Affine function: $\mathbf{a}^T\mathbf{x} + b$ is both convex and concave
- Quadratic function: $\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}+\mathbf{b}^T\mathbf{x}+c$ is convex provided that $\mathbf{Q} \succeq 0$ (positive semidefinite)
- Least squares loss: $\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2$ is always convex (since $\mathbf{A}^T\mathbf{A}$ is always positive semidefinite)
- Max function: $f(\mathbf{x}) = \max\{x_1, ..., x_n\}$ is convex

2.2 Key properties of convex functions

First-order characterization: if f is differentiable, then f is convex if and only if $\text{dom}(f)$ is convex, and $f(\mathbf{y}) \geq f(\mathbf{x}) + \bigtriangledown f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$. Therefore, for a differentiable convex function $\bigtriangledown f(\mathbf{x}) = 0 \iff \mathbf{x}$ minimizes f.

Second-order characterization: if f is twice differentiable, then f is convex if and only if $\text{dom}(f)$ is convex, and $\bigtriangledown^2 f(\mathbf{x}) \succeq 0$ (positive definite) for all $\mathbf{x} \in \text{dom}(f)$.

Jensen's inequality: if f is convex, and X is a random variable supported on $\text{dom}(f)$, then $f(E[X]) \leq E[f(X)]$.

Let us first consider the unconstrained, smooth convex optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$. Here f is convex, differentiable and defined over the full $\mathbb{R}^n$.

If f is strictly convex, then the solution is unique.

Taylor series

In calculus, a Taylor series is an infinite sum of terms that represents a function as a power series around a specific point. It is one of the most powerful tools in analysis and approximations. Given a function $f(x)$ that is infinitely differentiable around a point a, its Taylor series expansion around a is given by:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

where:

- $f^{(n)}(a)$ is the n-th derivative of $f(x)$ evaluated at a,
- $n!$ is the factorial of n, and
- $(x-a)^n$ represents powers of $(x-a)$.

If a is set at 0, it is also called the Maclaurin series.

Taylor Series for Common Functions

1. Exponential Function

The Taylor series expansion of e^x around $a = 0$ (also known as the Maclaurin series) is:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

2. Sine Function

The Taylor series for $\sin(x)$ around $a = 0$ is:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

3. Cosine Function

The Taylor series for $\cos(x)$ around $a = 0$ is:

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

Remainder Term

In practice, we truncate the sum after a finite number of terms. The Taylor series only approximates the function up to a certain order. The error introduced by truncating the series is given by the remainder term, which is:

$$R_n(x) = \sum_{j=n+1}^{\infty} \frac{f^{(j)}(a)}{j!}(x-a)^j = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{n+1}$$

for some c between a and x.

The Taylor series is a fundamental tool in calculus for approximating functions near a given point. In statistics, it is widely applied for methodological developments, numerical implementations, and theoretical explorations.

If $|x - a| < \epsilon$ (small) i.e. we are interested in learning the function locally around a, then a Taylor series-based approximation for f with first n terms only would have a maximum error of $\frac{f^{(n+1)}(c)}{(n+1)!}\epsilon^{n+1}$. Thus, with more terms, the error decreases exponentially $[\epsilon^{n+1} = \exp\left(-n\frac{1}{\log\epsilon}\right)]$. Oftentimes, setting $n = 2$ is sufficient to get a good approximation.

Gradient Descent

3 Introduction

We further assume that 'finite' solution exists (there are convex problems that get minimized out in infinity, but we assume we aren't in that case).

Under these assumptions, we denote the optimal criterion value by $f^* = \min_{\mathbf{x}} f(\mathbf{x}) = f(\mathbf{x}^*)$, thus the solution is attained at $x = x^*$.

The Gradient Descent algorithm works as follows:

1. Choose an initial point $\mathbf{x}^{(0)} \in \mathbb{R}^n$
2. Repeat: $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - t_k \bigtriangledown f(\mathbf{x}^{(k-1)})$

3. Stop at some point (will be discussed soon)

Essentially, after choosing some initial point $\mathbf{x}^{(0)}$, we move it in the direction of the negative gradient (this points us in a direction where the function is decreasing) by some positive amount t_1, calling this $\mathbf{x}_1$. And the same process is repeated.

4 Interpretation

The second-order Taylor expansion of f gives us $f(\mathbf{y}) \approx f(\mathbf{x}) + \bigtriangledown f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{1}{2}(\mathbf{y}-\mathbf{x})^T \bigtriangledown^2 f(\mathbf{x})(\mathbf{y}-\mathbf{x})$.

Consider the Quadratic approximation of f, replacing $\bigtriangledown^2 f(\mathbf{x})$ by $\frac{1}{t}\mathbf{I}$ (which alters the curvature given by the Hessian with a much simpler notion of curvature - something spherical), we have $f(\mathbf{y}) \approx f(\mathbf{x}) + \bigtriangledown f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{1}{2t}\|\mathbf{y}-\mathbf{x}\|_2^2$

This leads to a convex quadratic form. We know we can minimize it just by setting its gradient to 0.

Minimizing the RHS w.r.t. $\mathbf{y}$, we get $\frac{\partial f(y)}{\partial y} \approx \bigtriangledown f(\mathbf{x}) + \frac{1}{2}(\mathbf{y}-\mathbf{x}) = 0 \implies \mathbf{y} = \mathbf{x} - t \bigtriangledown f(\mathbf{x})$. This gives us the gradient descent update rule. In other words, gradients descent actually chooses the next point to minimize the approximated $f(\mathbf{y})$.

We can also think of the above approximation as a sum of 2 terms:

- A linear approximation to f given by $f(\mathbf{x}) + \bigtriangledown f(\mathbf{x})^T(\mathbf{y}-\mathbf{x})$.
- A proximity term to $\mathbf{x}$ given by $\|\mathbf{y}-\mathbf{x}\|_2^2$ with weight $\frac{1}{2t}$.

The above two steps present a tradeoff between the linear approximation step and closeness to the current value. The proximity term keeps us close to $\mathbf{x}$. Our aim would thus be to keep $\frac{1}{2t}$ small as the linear approximation is executed continuously, it will take $\mathbf{y}$ to infinity.

5 How to choose step sizes

An important aspect of gradient descent is pickings the step size. Actually, this is the most important step. Now there are several softwares to compute gradient automatically (they are called Automatic Differentiation like CppAD, Pytorch, tensorflow, etc.). But the step-size tuning still may require some human adjustments. Below are some methods which are used for this purpose.

5.1 Fixed step size

The simplest strategy is to take the step sizes to be fixed. So we choose $t_k = t$ for all $k = 1, 2, 3, \ldots$. There are some problems with doing this

- If t is too large, gradient descent can diverge.
- If t is too small, gradient descent can be (very) slow to converge.

We need a t which is 'correct' for the given problem.

5.2 Backtracking Line Search

An alternative which is very popular in practice is to use adaptive step sizes not just something fixed, but instead trying to guess the right step size at every iteration via some heuristic. The most popular such heuristic is called Backtracking Line Search. This works as follows:

- First, fix parameters $0 < \beta < 1$ and $0 < \alpha \leq \frac{1}{2}$.
- At each iteration (of gradient descent), start with $t = t_{init}$ (something relatively large), and while $f(\mathbf{x} - t \bigtriangledown f(\mathbf{x})) > f(\mathbf{x}) - \alpha t \| \bigtriangledown f(\mathbf{x}) \|_2^2$ shrink $t := \beta t$. Else, perform the gradient update $\mathbf{x}_{new} = \mathbf{x} - t \bigtriangledown f(\mathbf{x})$.

At each iteration (of gradient descent), start with $t = t_{init}$ (something relatively large), and while the update criterion essentially tells that if the progress we make by going from $\mathbf{x}$ to $\mathbf{x} - t \bigtriangledown f(\mathbf{x})$ is bigger than the progress we had $f(\mathbf{x}) - \alpha t \| \bigtriangledown f(\mathbf{x}) \|_2^2$, then we make t smaller (βt). So by shrinking t, we move from right to left on the graph, as long as the line is beneath the function, and then stop when it exceeds it.

5.3 Exact Line Search

We could also choose a step to take the best we can along the direction of the negative gradient, called Exact Line Search:

$$t = \text{argmin}_{\mathbf{s} \geq 0} f(\mathbf{x} - \mathbf{s} \bigtriangledown f(\mathbf{x}))$$

where $\mathbf{x}$ is fixed above (its the point we're currently at during gradient descent), and we find the value of $\mathbf{s}$ that allows us to do the absolute best we can along that line segment. Trying to make this as small as possible over all $\mathbf{s}$ is a 1-dimensional optimization problem. In principle, we might think it's a good idea to optimize this. But this is usually not possible to do directly; and approximations to the same are not as efficient as general backtracking. So is not worth the effort (except for fairly simple problems maybe).

I would recommend doing this 1-D optimization adaptively. Most of the theoretical results suggest that the step size parameter should be reduced over iteration. So, we can run above optimization after each 20-th iteration and also optimize in the range $(0, t_{current})$.

For a convex f, we can show that in order to achieve an error difference of ϵ with the optimal value, we need to run $O(1/\epsilon)$ iterations of gradient descent. That means, the number of required iterations increase inverse-linearly with respect to the desired error bound.

6 Application to linear regression

6.1 Simple linear regression

In case of linear regression, we have $f(\boldsymbol{\beta}) = \frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = \frac{1}{n} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \frac{1}{n} \sum_i (y_i - \mathbf{x}_i \boldsymbol{\beta})^2$. The fraction $\frac{1}{n}$ helps to keep the gradient value at a manageable range. It has some additional role in 'stochastic' gradient descent.

$f'(\boldsymbol{\beta}) = -\frac{2}{n} \mathbf{X}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$

6.2 Generalized linear regression

Here, we focus on logistic regression for binary data. Let $(y_i, \mathbf{x}_i)_{1 \leq i \leq n}$ be the set of observations where $y_i \in \{0, 1\}$. The logistic regression model is,

$$\begin{aligned} y_i \sim & \text{Bernoulli}(p_i) \\ p_i = & \frac{1}{1 + \exp(\mathbf{x}_i^T \boldsymbol{\beta})} \end{aligned}$$

For this problem, $f(\boldsymbol{\beta})$ will be average negative log-likelihood. What is $f'(\boldsymbol{\beta})$?