CS 189 Introduction to Machine Learning
Spring 2018

HW5

1 Getting Started

Read through this page carefully. You may typeset your homework in latex or submit neatly handwritten/scanned solutions. Please start each question on a new page. Deliverables:

1. Submit a PDF of your writeup, **with an appendix for your code**, to assignment on Gradescope, “HW5 Write-Up”. If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.
2. If there is code, submit all code needed to reproduce your results, “HW5 Code”.
3. If there is a test set, submit your test set evaluation results, “HW5 Test Set”.

After you’ve submitted your homework, watch out for the self-grade form.

(a) Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it. We just want to make it *extra* clear so that no one inadverdently cheats.

I certify that all solutions are entirely in my words and that I have not looked at another student’s solutions. I have credited all external sources in this write up.

This homework is due **on Wednesday, July 25th at 11:59pm.**

2 Step Size in Gradient Descent

By this point in the class, we know that gradient descent is a powerful tool for moving towards local minima of general functions. We also know that local minima of convex functions are global minima. In this problem, we will look at the convex function $f(\mathbf{x}) = ||\mathbf{x} - \mathbf{b}||_2$. Note that we are using "just" the regular Euclidean ℓ_2 norm, *not* the norm squared! This problem illustrates the importance of understanding how gradient descent works and choosing step sizes strategically. In fact, there is a lot of active research in variations on gradient descent. Throughout the question we will look at different kinds of step-sizes. Constant step size vs. decreasing step size. We will also look at the the rate at which the different step sizes decrease and draw some conclusions about the rate of convergence. Notice that we want to make sure the we get to some local minimum and we want to do it as quickly as possible.

You have been provided with a tool in `step_size.py` which will help you visualize the problems below.

(a) Let $\mathbf{x}, \mathbf{b} \in \mathbb{R}^d$. **Prove that $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$ is a convex function of $\mathbf{x}$.**

(b) We are minimizing $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{b} = [4.5, 6] \in \mathbb{R}^2$, with gradient descent. We use a constant step size of $t_i = 1$. That is,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - t_i \nabla f(\mathbf{x}_i) = \mathbf{x}_i - \nabla f(\mathbf{x}_i).$$

We start at $\mathbf{x}_0 = [0, 0]$. **Will gradient descent find the optimal solution? If so, how many steps will it take to get within 0.01 of the optimal solution? If not, why not?** Prove your answer. (Hint: use the tool to compute the first ten steps.) **What about general $\mathbf{b} \neq \mathbf{0}$?**

(c) We are minimizing $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{b} = [4.5, 6] \in \mathbb{R}^2$, now with a decreasing step size of $t_i = (\frac{5}{6})^i$ at step i. That is,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - t_i \nabla f(\mathbf{x}_i) = \mathbf{x}_i - (\frac{5}{6})^i \nabla f(\mathbf{x}_i).$$

We start at $\mathbf{x}_0 = [0, 0]$. **Will gradient descent find the optimal solution? If so, how many steps will it take to get within 0.01 of the optimal solution? If not, why not?** Prove your answer. (Hint: examine $\|\mathbf{x}_i\|_2$.) **What about general $\mathbf{b} \neq \mathbf{0}$?**

(d) We are minimizing $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{b} = [4.5, 6] \in \mathbb{R}^2$, now with a decreasing step size of $t_i = \frac{1}{i+1}$ at step i. That is,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - t_i \nabla f(\mathbf{x}_i) = \mathbf{x}_i - \frac{1}{i+1} \nabla f(\mathbf{x}_i).$$

We start at $\mathbf{x}_0 = [0, 0]$. **Will gradient descent find the optimal solution? If so, how many steps will it take to get within 0.01 of the optimal solution? If not, why not?** Prove your answer. (Hint: examine $\|\mathbf{x}_i\|_2$, and use $\sum_{i=1}^{n} \frac{1}{i}$ is of the order $\log n$.) **What about general $\mathbf{b} \neq \mathbf{0}$?**

(e) Now, say we are minimizing $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$. Use the code provided to test several values of $\mathbf{A}$ with the step sizes suggested above. Make plots to visualize what is happening. We suggest trying $\mathbf{A} = [[10, 0], [0, 1]]$ and $\mathbf{A} = [[15, 8], [6, 5]]$. **Will any of the step sizes above work for all choices of $\mathbf{A}$ and $\mathbf{b}$?** You do not need to prove your answer, but you should briefly explain your reasoning.

3 Convergence Rate of Gradient Descent

In the previous problem, you examined $||\mathbf{A}\mathbf{x} - \mathbf{b}||_2$ (without the square). You showed that even though it is convex, getting gradient descent to converge requires some care. In this problem, you will examine $\frac{1}{2}||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2$ (with the square). You will show that now gradient descent converges quickly.

For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and a vector $\mathbf{b} \in \mathbb{R}^n$, consider the quadratic function $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ such that $\mathbf{A}^\top\mathbf{A}$ is positive definite.

Throughout this question the *Cauchy-Schwarz inequality* might be useful: Given two vectors $\mathbf{u}, \mathbf{v}$:

$$|\mathbf{u}^\top \mathbf{v}| \leq \|\mathbf{u}\|_2 \|\mathbf{v}\|_2,$$

with equality only when $\mathbf{v}$ is a scaled version of $\mathbf{u}$.

(a) First, consider the case $\mathbf{b} = \mathbf{0}$, and think of each $\mathbf{x} \in \mathbb{R}^d$ as a "state". Performing gradient descent moves us sequentially through the states, which is called a "state evolution". **Write out the state evolution for n iterations of gradient descent using step-size $\gamma > 0$. i.e. express $\mathbf{x}_n$ as a function of $\mathbf{x}_0$.** Use $\mathbf{x}_0$ to denote the initial condition of where you start gradient descent from.

(b) A state evolution is said to be stable if it does not blow up arbitrarily over time. Specifically, if state n is

$$\mathbf{x}_n = \mathbf{B}^n \mathbf{x}_0$$

then we need *all* the eigenvalues of $\mathbf{B}$ to be less than or equal to 1 in absolute value, otherwise $\mathbf{B}^n$ might blow up $\mathbf{x}_0$ for large enough n.

When is the state evolution of the iterations you calculated above stable when viewed as a dynamical system?

(c) Define

$$\varphi(\mathbf{x}) = \mathbf{x} - \gamma \nabla f(\mathbf{x}),$$

for some constant step size $\gamma > 0$, and define

$$\beta = \max\left\{ |1 - \gamma \lambda_{\max}(\mathbf{A}^\top \mathbf{A})|, |1 - \gamma \lambda_{\min}(\mathbf{A}^\top \mathbf{A})| \right\}.$$

Let $\lambda_{\min}(\mathbf{A}^\top\mathbf{A})$ denote the smallest eigenvalue of the matrix $\mathbf{A}^\top\mathbf{A}$; similarly, let $\lambda_{\max}(\mathbf{A}^\top\mathbf{A})$ denote the largest eigenvalue of the matrix $\mathbf{A}^\top\mathbf{A}$. Assume that

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{\alpha}{2} \beta^{2k} \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2.$$

The convergence rate is a function of β, so it's desirable for β to be as small as possible. Recall that β is a function of γ, so we want to pick γ such that β is as small as possible, as a function of $\lambda_{\min}(\mathbf{A}^\top\mathbf{A}), \lambda_{\max}(\mathbf{A}^\top\mathbf{A})$. **Write the resulting convergence rate as a function of $\kappa = \frac{\lambda_{\max}(\mathbf{A}^\top\mathbf{A})}{\lambda_{\min}(\mathbf{A}^\top\mathbf{A})}$, That is, show that**

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{\alpha}{2}\left(\frac{\kappa-1}{\kappa+1}\right)^{2k}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2$$

4 SGD on OLS

Recall that the ordinary least squares problem can be written as:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{n}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = \min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top\mathbf{w} - \mathbf{y}_i)^2 = \min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w})$$

where $f_i(\mathbf{w}) := (\mathbf{x}_i^\top\mathbf{w} - \mathbf{y}_i)^2$. Here $\mathbf{x}_i^\top$ is the ith row of matrix $\mathbf{X}$ and f_i is the loss of the training example i. We implement SGD as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha_t \nabla f_{i_t}(\mathbf{w}^t).$$

where i_t is uniformly sampled from all samples $\{1, 2, \cdots, n\}$ (and is independently drawn for each iteration t).

(a) We will now use simulations to compare various first order methods for solving OLS. In particular, we want to plot the estimation error $||\mathbf{w}^t - \mathbf{w}^*||_2$ against the gradient descent step in the following 3 scenarios:

- When there exists an exact solution $\mathbf{X}\mathbf{w}^* = \mathbf{y}$, plot the errors when you are using SGD and a constant learning rate.
- When there does not exist an exact solution $\mathbf{X}\mathbf{w}^* = \mathbf{y}$, plot the errors when you are using SGD and a constant learning rate. Also plot the errors for GD and the same constant learning rate.
- When there does not exist an exact solution $\mathbf{X}\mathbf{w}^* = \mathbf{y}$, plot the errors when you are using SGD and a decaying learning rate.

Using the attached starter code, **implement the gradient updates** in the function `sgd()` , while all the plotting functions are already there. **Show the 3 plots you obtain using the starter code. Report the average squared error computed in the starter code. What's your conclusion?**

5 Midterm Re-do

Make corrections to your solutions to the midterm exam problems.

This question is intended to give you a second chance to get things right. We will release a copy of the problem statements after the midterm.

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn the material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.