CS 189 Introduction to Machine Learning
Fall 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 to assignment on Gradescope, "HW5 Write-Up". If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.

(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 **Wednesday, October 10th at 10pm.**

2 Step Size in Gradient Descent

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. We will also look at the the rate at which the different step sizes decrease and draw some conclusions about the rate of convergence. You have been provided with a tool in `step_size_helper.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) **For $\nabla_{\mathbf{x}} f(\mathbf{x})$, determine where it is well-defined and compute its value.**

(c) We are minimizing $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{b} = [6, 4.5] \in \mathbb{R}^2$, 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 (for this and subsequent problems, this is in terms of $\|\mathbf{x_k} - \mathbf{x_*}\|_2$)? If not, why not?** Prove your answer. (Hint: use the tool to compute the first ten steps.) **What about general $\mathbf{b} \neq \mathbf{0}$?**

(d) We are minimizing $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{b} = [6, 4.5] \in \mathbb{R}^2$, now with a decreasing step size of $t_i = (\frac{6}{7})^i$ at step i. That is,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - t_i \nabla f(\mathbf{x}_i) = \mathbf{x}_i - (\frac{6}{7})^i \nabla f(\mathbf{x}_i).$$

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 $\mathbf{b} \neq \mathbf{0}$?**

(e) We are minimizing $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{b} = [6, 4.5] \in \mathbb{R}^2$, 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 $\mathbf{b} \neq \mathbf{0}$? State (roughly—up to the correct order) the amount of steps required to reach the optimum in the general case, for whichever cases are possible.**

(f) 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 $\mathbf{A} = [[1, 0], [0, 10]]$ and $\mathbf{A} = [[6, 5], [15, 8]]$. **Will any of the step sizes above work (i.e. eventually be within any presribed distance ϵ) 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.

(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 $\gamma > 0$. That is, explicitly express $\mathbf{x}_n$ as a function of $\mathbf{x}_0$ (and not as a function of any other iterates).**

(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. To see this, consider the case when $\mathbf{x}_0$ is an eigenvector of $\mathbf{B}$ corresponding to an eigenvalue with magnitude greater than 1.

When is the state evolution of the iterations you calculated above stable? Express this condition in terms of eigenvalues of $\mathbf{A}^\top\mathbf{A}$.

(c) We want to bound the progress of gradient descent in the general case, when $\mathbf{b}$ is arbitrary. To do this, we first show a slightly more general bound, which relates how much the spacing between two points changes if they *both* take a gradient step. If this spacing shrinks, this is called a contraction. Define map taking an iterate to its next step as $\varphi(\mathbf{x}) = \mathbf{x} - \gamma\nabla f(\mathbf{x})$, for some constant step size $\gamma > 0$. **Show that for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$,**

$$\|\varphi(\mathbf{x}) - \varphi(\mathbf{x}')\|_2 \leq \rho\|\mathbf{x} - \mathbf{x}'\|_2$$

where $\rho = \max\left\{|1 - \gamma\lambda_{\max}(\mathbf{A}^\top\mathbf{A})|, |1 - \gamma\lambda_{\min}(\mathbf{A}^\top\mathbf{A})|\right\}$. Note that $\lambda_{\min}(\mathbf{A}^\top\mathbf{A})$ denotes the smallest eigenvalue of the matrix $\mathbf{A}^\top\mathbf{A}$; similarly, $\lambda_{\max}(\mathbf{A}^\top\mathbf{A})$ denotes the largest eigenvalue of the matrix $\mathbf{A}^\top\mathbf{A}$. (Hint: You may use the fact regarding the spectral norm (the "induced 2-norm"): $\|Qz\|_2 \leq \|Q\|_2\|z\|_2$, where $\|Q\|_2 := \sigma_1(Q)$.)

(d) Now we give a bound for progress after k steps of gradient descent. Define

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}).$$

Show that

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2 = \|\varphi(\mathbf{x}_k) - \varphi(\mathbf{x}^*)\|_2$$

and conclude that

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2 \leq \rho^{k+1}\|\mathbf{x}_0 - \mathbf{x}^*\|_2.$$

(e) Now, denote $D = \|\mathbf{x}_0 - \mathbf{x}^*\|_2$, and assume $\rho < 1$. Assume we stop gradient descent after k iterations. **Give a sufficient condition on k to ensure we stopped within ϵ of the optimal $\mathbf{x}_*$. Write your answer in terms of positive quantites.**

(f) However, what we often care about is progress in the objective value $f(\mathbf{x})$. That is, we want to show how quickly $f(\mathbf{x}_k)$ is converging to $f(\mathbf{x}^*)$. We can do this by relating $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ to $\|\mathbf{x}_k - \mathbf{x}^*\|_2$, or even better, relating $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ to $\|\mathbf{x}_0 - \mathbf{x}^*\|_2$, for some starting point $\mathbf{x}_0$. First, **show that**

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \frac{1}{2}\|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|_2^2.$$

(g) **Now, show that**

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{\lambda_1}{2}\|\mathbf{x}_k - \mathbf{x}^*\|_2^2,$$

for $\lambda_1 = \lambda_{\max}(\mathbf{A}^\top\mathbf{A})$, and show that

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{\lambda_1}{2}\rho^{2k}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2.$$

(Hint: Compare with parts (c) and (d)).

(h) As before, denote $D = \|\mathbf{x}_0 - \mathbf{x}^*\|_2$, and assume $\rho < 1$. Assume we stop gradient descent after k iterations. **Give a sufficient condition on k to ensure our function value when we stop, $f(\mathbf{x_k})$, is within η of the optimal function value $f(\mathbf{x}_*)$. Write your answer in terms of positive quantites (assume D is large). How does the scaling in η in this bound differ from the scaling in ϵ in the bound derived in part (e)?**

(i) Finally, the convergence rate is a function of ρ, so it's desirable for ρ to be as small as possible. Recall that ρ is a function of γ, so we want to pick γ such that ρ 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 ρ as a function of $\kappa = \frac{\lambda_{\max}(\mathbf{A}^\top\mathbf{A})}{\lambda_{\min}(\mathbf{A}^\top\mathbf{A})}$**. This quantity is known as the *condition number* of $\mathbf{A}^\top\mathbf{A}$.

4 A Simple Classification Approach

Make sure to submit the code you write in this problem to "HW5 Code" on Gradescope.

Classification is an important problem in applied machine learning and is used in many different applications like image classification, object detection, speech recognition, machine translation and others.

In *classification*, we assign each datapoint a class from a finite set (for example the image of a digit could be assigned the value $0, 1, \ldots, 9$ of that digit). This is different from *regression*, where each datapoint is assigned a value from a continuous domain like $\mathbb{R}$ (for example features of a house like location, number of bedrooms, age of the house, etc. could be assigned the price of the house).

In this problem we consider the simplified setting of classification where we want to classify data points from $\mathbb{R}^d$ into *two* classes. For a linear classifier, the space $\mathbb{R}^d$ is split into two parts by a hyperplane: All points on one side of the hyperplane are classified as one class and all points on the other side of the hyperplane are classified as the other class.

The goal of this problem is to show that even a regression technique like linear regression can be used to solve a classification problem. This can be achieved by regressing the data points in the training set against -1 or 1 depending on their class and then using the level set of 0 of the regression function as the classification hyperplane (i.e. we use 0 as a threshold on the output to decide between the classes).

Later in lecture we will learn why linear regression is not the optimal approach for classification and we will study better approaches like logistic regression, SVMs and neural networks.

(a) The dataset used in this exercise is a subset of the MNIST dataset. The MNIST dataset assigns each image of a handwritten digit their value from 0 to 9 as a class. For this problem we only keep digits that are assigned a 0 or 1, so we simplify the problem to a two-class classification problem.

Download and visualize the dataset (example code included). Include three images that are labeled as 0 and three images that are labeled as 1 in your submission.

(b) We now want to use linear regression for the problem, treating class labels as real values $y = -1$ for class "zero" and $y = 1$ for class "one". In the dataset we provide, the images have already been flattened into one dimensional vectors (by concatenating all pixel values of the two dimensional image into a vector) and stacked as rows into a feature matrix $\mathbf{X}$. We want to set up the regression problem $\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ where the entry y_i is the value of the class (-1 or 1) corresponding to the image in row $\mathbf{x}_i^\top$ of the feature matrix. **Solve this regression problem for the training set and report the value of $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ as well as the weights $\mathbf{w}$.** For this problem you may only use pure Python and numpy (no machine learning libraries!).

(c) Given a new flattened image $\mathbf{x}$, one natural rule to classify it is the following one: It is a zero if $\mathbf{x}^\top\mathbf{w} \leq 0$ and a one if $\mathbf{x}^\top\mathbf{w} > 0$. **Report what percentage of the digits in the training set are correctly classified by this rule. Report what percentage of the digits in the test set are correctly classified by this rule.**

(d) **Why is the performance typically evaluated on a separate test set (instead of the training set) and why is the performance on the training and test set similar in our case?** We will cover these questions in more detail later in the class.

(e) Unsatisfied with the current performance of your classifier, you call your mother for advice, and she suggests to use random features instead of raw pixel features. Specifically, she suggests to use the feature map

$$\phi(\mathbf{x}) = \cos\Big(\mathbf{G}^\top \mathbf{x} + \mathbf{b}\Big) ,$$

where each entry of $G \in \mathbb{R}^{d \times p}$ is drawn i.i.d. as $\mathcal{N}(0, 0.01)$ and each entry in the vector $b \in \mathbb{R}^p$ is drawn i.i.d from the uniform distribution on $[0, 2\pi]$. Note that $\cos$ should be taken point wise, i.e. $\phi(x)$ should output a vector in $\mathbb{R}^p$. **With** $p = 5000$**, report what percentage of digits are correctly classified using this approach on the training set and test set. Make sure to adapt the classification rule, i.e. the threshold set for the outputs.**