CS 189 Introduction to Machine Learning
Spring 2018

HW5

Your self-grade URL is http://eecs189.org/self_grade?question_ids=1_1,1_2,2_1,2_2,2_3,2_4,2_5,3_1,3_2,3_3,4_1,5,6.

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 $\mathbf{x}$.**

Solution: Recall that a function is convex if for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ we have

$$f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \le \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2)$$

for all $0 \le \lambda \le 1$. For $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2$, the triangle inequality gives us:

$$\begin{aligned}\|\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 - \mathbf{b}\|_2 &= \|\lambda(\mathbf{x}_1 - \mathbf{b}) + (1-\lambda)(\mathbf{x}_2 - \mathbf{b})\|_2 \\ &\le \|\lambda(\mathbf{x}_1 - \mathbf{b})\|_2 + \|(1-\lambda)(\mathbf{x}_2 - \mathbf{b})\|_2 \\ &= \lambda\|\mathbf{x}_1 - \mathbf{b}\|_2 + (1-\lambda)\|\mathbf{x}_2 - \mathbf{b}\|_2,\end{aligned}$$

which shows that f is convex.

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

Solution: Using the tool provided (or computing gradients by hand), we get

$$\mathbf{x}_6 = \mathbf{x}_8 = [4.2, 5.6]$$

and

$$\mathbf{x}_7 = \mathbf{x}_9 = [4.8, 6.4].$$

Examine the formula

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

and notice that $\mathbf{x}_{i+1}$ only depends on $\mathbf{x}_i$, regardless of what step we are on (this is only the case because we are using a constant step size). It means that if $\mathbf{x}_i = \mathbf{x}_j$, then $\mathbf{x}_{i+1} = \mathbf{x}_{j+1}$. Thus, this pattern will repeat indefinitely and we do not reach the optimal solution. It is also helpful to notice that

$$\|\mathbf{x}_6\|, \|\mathbf{x}_7\| \in \mathbb{Z}.$$

In general, notice that

$$-\nabla f(\mathbf{x}_i) = \frac{\mathbf{b} - \mathbf{x}_i}{\|\mathbf{x}_i - \mathbf{b}\|}$$

will always have unit length. In fact, we can prove by induction that $\mathbf{x}_k$ is always an integer multiple of the unit vector $\frac{\mathbf{b}}{\|\mathbf{b}\|}$. Thus,

$$\|\mathbf{x}_k\| \in \mathbb{Z}.$$

If we are lucky and $\|\mathbf{b}\|$ happens to be an integer (or very close to integer), then we will reach or get near the optimal solution. Otherwise, our step size is too coarse to hit it.

In fact, for the function $\|\mathbf{x} - \mathbf{b}\|$, a constant step size will rarely work. We are too prone to "jumping over" our solution. A decreasing step size is necessary and we will investigate decreasing stepsizes next.

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

Solution: Notice that we can express $\mathbf{x}_{i+1}$ as the sum of all the steps taken, so we can express its norm as

$$\|\mathbf{x}_{i+1}\| = \|\mathbf{x}_0 - \sum_{j=0}^{i} (\frac{5}{6})^j \nabla f(\mathbf{x}_i)\| \leq \|\mathbf{x}_0\| + \sum_{j=0}^{i} \|(\frac{5}{6})^j \nabla f(\mathbf{x}_i)\|$$

But all of $\|\nabla f(\mathbf{x}_i)\|$ are exactly 1, so we can write this as

$$\|\mathbf{x}_{i+1}\| \leq \|\mathbf{x}_0\| + \sum_{j=0}^{i} (\frac{5}{6})^j \|\nabla f(\mathbf{x}_i)\| = 0 + \sum_{j=0}^{i} (\frac{5}{6})^j \leq 6.$$

In words, we can never "travel" a distance of more than 6 from $\mathbf{0}$, so we will never get to our optimal solution $\mathbf{b}$ if $\|\mathbf{b}\| > 6$. So, while we need a decreasing step size, we also have to be sure it does not decrease too quickly.

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

Solution: The key realization here is that all our gradient steps have $\|\nabla f(\mathbf{x}_i)\| = 1$ and move along the same vector towards $\mathbf{b}$ (to be precise, we could prove by induction that our $\mathbf{x}_i$ and $\nabla f(\mathbf{x}_i)$ will always be in the span of $\mathbf{b}$). We can write

$$\|\mathbf{x}_{i+1}\| = \|\mathbf{x}_0 - \sum_{j=0}^{i} (\frac{1}{j+1}) \nabla f(\mathbf{x}_i)\| = \sum_{j=0}^{i} \|(\frac{1}{j+1}) \nabla f(\mathbf{x}_i)\| = \sum_{j=1}^{i} \frac{1}{j+1} \approx \ln i.$$

This sum diverges, so we will reach the optimal solution $\mathbf{b}$ eventually, but it could take quite a while. For $\mathbf{b} = [4.5, 6]$ it takes about 1000 steps. In generally, the number of steps it takes will be exponential in $\|\mathbf{b}\|$, since we have

$$\ln i \approx \|\mathbf{b}\| \implies i \approx e^{\|\mathbf{b}\|}.$$

We may not hit $\mathbf{b}$ exactly. Once we exceed $\mathbf{b}$, we start oscillating around it. Let i_0 be the first step where we "cross over" $\mathbf{b}$. After that, we know we are always moving towards $\mathbf{b}$ (whether we are stepping backwards or forwards), so after step i (assuming $i > i_0$), we can be at most $\frac{1}{i+1}$ from the optimal solution (if we are more than $\frac{1}{i+1}$ away, that means our previous step went in the wrong direction).

(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.

Solution: The first two step sizes did not work for $\mathbf{A} = \mathbf{I}$, so they will not work in general. The last step sizes *does* always work, though it may take a very long time to converge.

A perspective to notice is that the $\mathbf{x}_i$ are still ultimately moving towards $\mathbf{x}^*$, but not on a straight line but instead on a curved path. The path has finite length, so we will eventually reach $\mathbf{x}^*$. Once we are close to $\mathbf{x}^*$, successive steps will oscillate around $\mathbf{x}^*$.

Formally: This was not asked in the question, but here is the proof: We have

$$\nabla f(\mathbf{x}) = \frac{\mathbf{A}^\top(\mathbf{A}\mathbf{x}-\mathbf{b})}{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|}$$

and therefore $\|\nabla f(\mathbf{x})\|_2^2 \leq \sigma_{max}^2(\mathbf{A})$. From convexity, we furthermore have

$$\nabla f(\mathbf{x}_i)^\top(\mathbf{x}^* - \mathbf{x}_i) \leq f(\mathbf{x}^*) - f(\mathbf{x}_i)$$

which yields

$$\begin{aligned}
\|\mathbf{x}_{i+1} - \mathbf{x}^*\|_2^2 &= \|\mathbf{x}_i - t_i\nabla f(\mathbf{x}_i) - \mathbf{x}^*\|_2^2 \\
&= \|\mathbf{x}_i - \mathbf{x}^*\|_2^2 - 2t_i\nabla f(\mathbf{x}_i)^\top(\mathbf{x}_i - \mathbf{x}^*) + t_i^2\|\nabla f(\mathbf{x}_i)\|_2^2 \\
&\leq \|\mathbf{x}_i - \mathbf{x}^*\|_2^2 - 2t_i(f(\mathbf{x}_i) - f(\mathbf{x}^*)) + t_i^2\sigma_{max}^2(\mathbf{A}) \\
&\leq \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 - 2\sum_{k=0}^{i} t_k(f(\mathbf{x}_k) - f(\mathbf{x}^*)) + \sigma_{max}^2(\mathbf{A})\sum_{k=0}^{i} t_k^2 \\
&\leq \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 - 2(f(\mathbf{x}_{best}) - f(\mathbf{x}^*))\sum_{k=0}^{i} t_k + \sigma_{max}^2(\mathbf{A})\sum_{k=0}^{i} t_k^2
\end{aligned}$$

where $\mathbf{x}_{best} = \arg\min_{0\leq k\leq i} f(\mathbf{x}_k)$. We denote $\|\mathbf{x}_0 - \mathbf{x}^*\|_2 = d_0$. This yields

$$0 \leq d_0^2 - 2(f(\mathbf{x}_{best}) - f(\mathbf{x}^*))\sum_{k=0}^{i} t_k + \sigma_{max}^2(\mathbf{A})\sum_{k=0}^{i} t_k^2$$

Hence,

$$f(\mathbf{x}_{best}) - f(\mathbf{x}^*) \leq \frac{d_0^2 + \sigma_{max}^2(\mathbf{A})\sum_k t_k^2}{\sum_k t_k} = \frac{d_0^2 + \sigma_{max}^2(\mathbf{A})\sum_k \frac{1}{(1+k)^2}}{\sum_k \frac{1}{1+k}} \xrightarrow[i\to\infty]{} 0$$

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.

Solution: *Quick note:* First let us clarify why gradient descent on the norm squared converges faster than gradient descent on the norm. The norm is convex but not strictly convex. Therefore gradient descent will only obtain the slow error rate $O(\frac{1}{\sqrt{k}})$ where k is the number of iterations (we did not prove this, but our step size selection in the last part of the last problem led to a worse bound $O(\frac{1}{\log k})$ and with a more refined analysis, we can get $O(\frac{1}{\sqrt{k}})$). On the other hand, the squared norm is strictly convex and therefore gradient descent can obtain the fast linear error rate $O(\beta^k)$ for $\beta < 1$ as we show in the remainder of this problem.

The general formula for the gradient update is

$$\mathbf{x}_n = \mathbf{x}_{n-1} - \gamma \nabla f(\mathbf{x}_{n-1})$$

which in this case is

$$\mathbf{x}_n = \mathbf{x}_{n-1} - \gamma(\mathbf{A}^\top \mathbf{A}\mathbf{x}_{n-1} - \mathbf{A}^\top \mathbf{b}) = (\mathbf{I} - \gamma \mathbf{A}^\top \mathbf{A})\mathbf{x}_{n-1} - \gamma \mathbf{A}^\top \mathbf{b}.$$

Since we assumed $\mathbf{b} = \mathbf{0}$, this works out to

$$\mathbf{x}_n = (\mathbf{I} - \gamma \mathbf{A}^\top \mathbf{A})\mathbf{x}_{n-1},$$

so by induction we can show

$$\mathbf{x}_n = (\mathbf{I} - \gamma \mathbf{A}^\top \mathbf{A})^n \mathbf{x}_0.$$

(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?

Solution: An important fact in linear algebra is that if we take a matrix $\mathbf{B}$ with eigenvalues λ_i to the power of k, then the resulting matrix $\mathbf{B}^k$ has eigenvalues λ_i^k with the same eigenvectors. The proof of this fact (inductively) is

$$\mathbf{B}^k\mathbf{v} = \mathbf{B}^{k-1}\mathbf{B}\mathbf{v} = \mathbf{B}^{k-1}\lambda\mathbf{v} = \lambda\mathbf{B}^{k-1}\mathbf{v} = \lambda\lambda^{k-1}\mathbf{v} = \lambda^k\mathbf{v}.$$

Let $\mathbf{B} = (\mathbf{I} - \gamma \mathbf{A}^\top \mathbf{A})$. In order for the state evolution to be stable, we need *all* the eigenvalues of $\mathbf{B}$ to be less than or equal to 1 in absolute value. More formally, this connects to an identity we have previously reviewed, which is

$$\|\mathbf{B}^k\mathbf{v}\|_2 \leq |(\lambda_{\max}(\mathbf{B}))^k| \|\mathbf{v}\|_2 = (|\lambda_{\max}(\mathbf{B})|)^k \|\mathbf{v}\|_2.$$

So if the largest eigenvalue of $(\mathbf{I}-\gamma\mathbf{A}^\top\mathbf{A})$ (in absolute value) is bounded by 1, then the system is stable.

(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**

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

Solution: We would like to solve

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

Recall that $0 < \lambda_{\min}(\mathbf{A}^\top\mathbf{A}) \leq \lambda_{\max}(\mathbf{A}^\top\mathbf{A})$ and that $\gamma > 0$. Then, the optimal solution is when the two values are equal in absolute value but opposite in value.

$$-(1-\gamma\lambda_{\max}(\mathbf{A}^\top\mathbf{A})) = 1-\gamma\lambda_{\min}(\mathbf{A}^\top\mathbf{A})$$

This gives $\gamma = \frac{2}{\lambda_{\max}(\mathbf{A}^\top\mathbf{A})+\lambda_{\min}(\mathbf{A}^\top\mathbf{A})}$.
Then, the convergence result from the previous part can be written as

$$\begin{aligned} f(\mathbf{x}_k) - f(\mathbf{x}^*) &= \frac{\alpha}{2}\left(\frac{\lambda_{\max}(\mathbf{A}^\top\mathbf{A})-\lambda_{\min}(\mathbf{A}^\top\mathbf{A})}{\lambda_{\max}(\mathbf{A}^\top\mathbf{A})+\lambda_{\min}(\mathbf{A}^\top\mathbf{A})}\right)^{2k}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 \\ &= \frac{\alpha}{2}\left(\frac{\kappa-1}{\kappa+1}\right)^{2k}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 \end{aligned}$$

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:

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

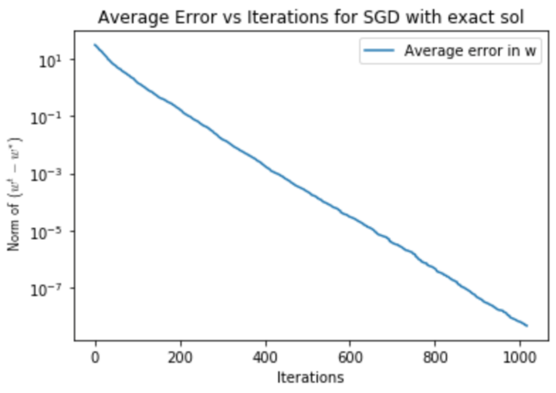
where i_t is uniformly sampled from all samples $\{1, 2, \cdots, n\}$ (and is independently drawn for each iteration t).

(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:

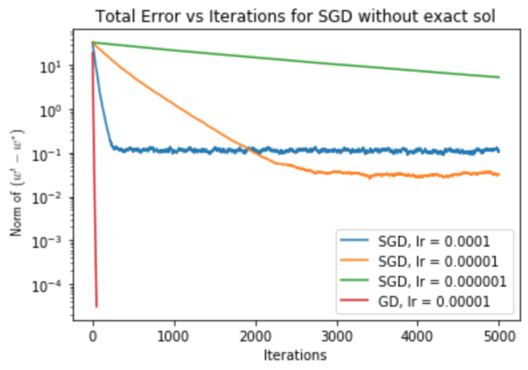
- When there exists an exact solution $\mathbf{X}\mathbf{w}^* = \mathbf{y}$, plot the errors when you are using SGD and a constant learning rate.
- When there does not exist an exact solution $\mathbf{X}\mathbf{w}^* = \mathbf{y}$, plot the errors when you are using SGD and a constant learning rate. Also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

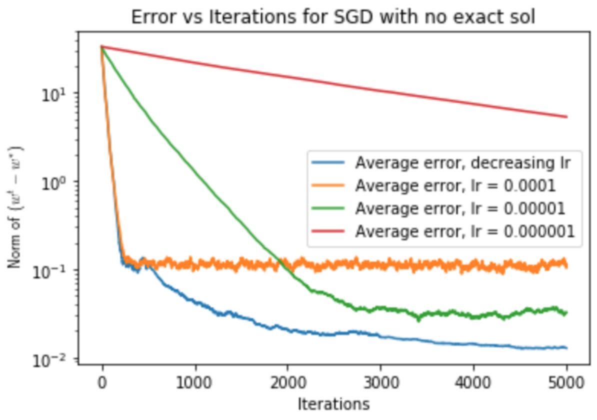
Solution:



The MSE is 4.6e-09.



For lr = 0.0001, the final average error is 0.10. For lr = 0.00001, the final average error is 0.033. For lr = 0.000001, the final average error is 5.32, because it has not converged after 5000 iterations. If it had converged, it would reach a lower error. The GD final error is 3.04e-5. The iteration in the plot is how many gradient steps taken. Note that each GD step is n times more computationally expensive than a SGD step, where n is the number of data points involved.



The SGD with decaying learning rate has an error of 0.013 at iteration 5000 and it would continue to decrease if we run more steps.

In conclusion, when there is an exact solution, SGD with constant learning rate can find the optimal solution. When there is no exact solution, SGD with constant learning rate will approach a place close to the optimal, and the closeness is related to the learning rate. However, when the learning rate is too small, it will take a long time to converge. GD with a constant learning rate can still converge to the optimal. Lastly, SGD with decaying learning rate can converge to the optimal and faster than SGD with a small enough learning rate.

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.