Homework #2

CSE 446/546: Machine Learning
Profs. Jamie Morgenstern and Ludwig Schmidt
Due: **Monday** October 31, 2022 11:59pm
A: 84 points, **B**: 18 points

Conceptual Questions

A1. The answers to these questions should be answerable without referring to external materials. Briefly justify your answers with a few words.

a. *[2 points]* Compared to L2 norm penalty, explain why a L1 norm penalty is more likely to result in sparsity (a larger number of 0s) in the weight vector.

b. *[2 points]* In at most one sentence each, state one possible upside and one possible downside of using the following regularizer: $\left(\sum_i |w_i|^{0.5}\right)$.

c. *[2 points]* True or False: If the step-size for gradient descent is too large, it may not converge.

d. *[2 points]* In at most one sentence each, state one possible advantage of SGD over GD (gradient descent), and one possible disadvantage of SGD relative to GD.

Solution:

a. Suppose features 1 and 2 are very correlated. Then the L2 loss favors $w_1 = w_2 = 1$ over $w_1 = 2, w_2 = 0$ because the loss in the former case is $1^2 + 1^2 = 2$, and in the latter case it is $2^2 + 0 = 4$. On the other hand, the L1 loss does not favor either of these ways of distributing weight 2 among the two features. Thus, it is more likely that the latter, sparse option ($w_1 = 2, w_2 = 0$) will be chosen.
b. Upside: heavily favors sparsity. Downside: non-convex.
c. True.
d. Advantage of SGD: each iteration is O(1) instead of O(d), where d is the dimensionality of the data. Disadvantage: the descent trajectory is more noisy, requiring more steps to converge.

Convexity and Norms

A2. A *norm* $\|\cdot\|$ over $\mathbb{R}^n$ is defined by the properties: (i) non-negativity: $\|x\| \geq 0$ for all $x \in \mathbb{R}^n$ with equality if and only if $x = 0$, (ii) absolute scalability: $\|a\,x\| = |a|\,\|x\|$ for all $a \in \mathbb{R}$ and $x \in \mathbb{R}^n$, (iii) triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

a. *[3 points]* Show that $f(x) = (\sum_{i=1}^n |x_i|)$ is a norm. (Hint: for (iii), begin by showing that $|a+b| \leq |a|+|b|$ for all $a, b \in \mathbb{R}$.)

b. *[2 points]* Show that $g(x) = \left(\sum_{i=1}^n |x_i|^{1/2}\right)^2$ is not a norm. (Hint: it suffices to find two points in $n = 2$ dimensions such that the triangle inequality does not hold.)

Context: norms are often used in regularization to encourage specific behaviors of solutions. If we define $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$ then one can show that $\|x\|_p$ is a norm for all $p \geq 1$. The important cases of $p = 2$ and $p = 1$ correspond to the penalty for ridge regression and the lasso, respectively.

Solution:

a. Non-negativity is clear: sum of non-negative numbers is non-negative. For $a, b \in \mathbb{R}$, $|ab| = |a||b|$ and $|a + b| \leq |a| + |b|$ by separately considering the four cases of signs of a and b. The former property implies absolutely scalability via

$$\left(\sum_{i=1}^n |a x_i|\right) = \left(\sum_{i=1}^n |a||x_i|\right) = |a| \left(\sum_{i=1}^n |x_i|\right).$$

The latter property implies the traingle inequality via

$$\left(\sum_{i=1}^n |x_i + y_i|\right) \leq \left(\sum_{i=1}^n |x_i| + |y_i|\right) = \left(\sum_{i=1}^n |x_i|\right) + \left(\sum_{i=1}^n |y_i|\right).$$

b. Let $x = (4, 0)$ and $y = (0, 4)$. Then $g(x - y) = (2 + 2)^2 = 16$. But $g(x) + g(y) = 4 + 4 = 8$, contradicting the triangle inequality.

B1. *[6 points]* For any $x \in \mathbb{R}^n$, define the following norms: $\|x\|_1 = \sum_{i=1}^n |x_i|$, $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$, $\|x\|_\infty := \lim_{p\to\infty} \|x\|_p = \max_{i=1,\ldots,n} |x_i|$. Show that $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1$.

Solution:

Without loss of generality, $|x_1| = \max_{i=1,\ldots,n} |x_i|$. Then

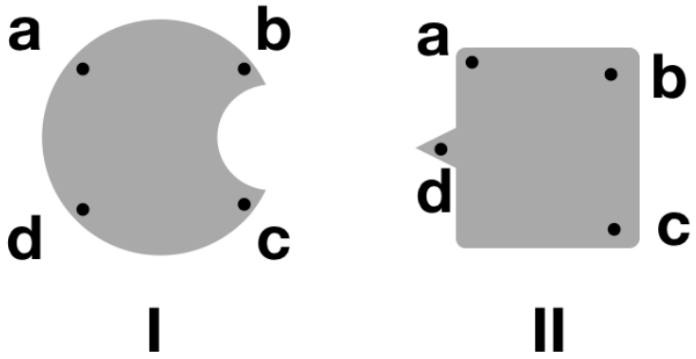
$$||x||_\infty = |x|_1 = (|x|_1^2)^{1/2} \leq (|x|_1^2 + |x_2|^2 + \cdots + |x_n|^2)^{1/2} = ||x||_2.$$

This proves the first inequality. The second one follows from

$$||x||_1^2 = \left(\sum_{i=1}^n |x_i|\right)^2 = \sum_{i,j} |x_i x_j| = \sum_i |x_i|^2 + \sum_{i\neq j} |x_i x_j| \geq \sum_i |x_i|^2 = ||x||_2^2$$

by taking squares of both sides.

A3. *[2 points]* A set $A \subseteq \mathbb{R}^n$ is *convex* if $\lambda x + (1-\lambda)y \in A$ for all $x, y \in A$ and $\lambda \in [0,1]$. For each of the grey-shaded sets below (I-II), state whether each one is convex, or state why it is not convex using any of the points a, b, c, d in your answer.



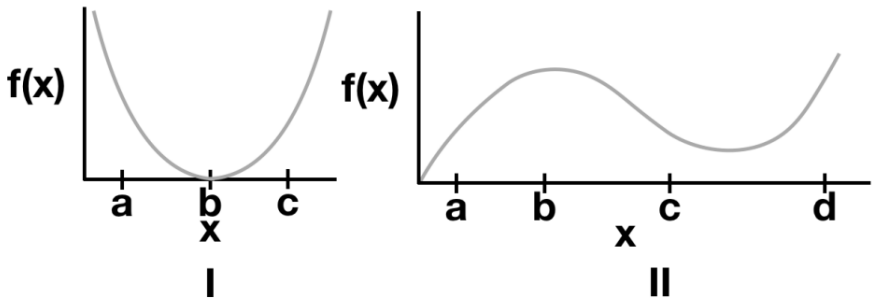
Solution:

I. Non-convex because the line between b and c is not contained in the set.
II. Non-convex because the line between a and d is not contained in the set.

A4. *[2 points]* We say a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex on a set A if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all $x, y \in A$ and $\lambda \in [0,1]$. For each of the functions shown below (I-II), state whether each is convex on the specified interval, or state why not with a counterexample using any of the points a, b, c, d in your answer.

a. Function in panel I on $[a, c]$

b. Function in panel II on $[a, d]$



Solution:

I. Convex, because the line segment between $f(x)$ and $f(y)$ is above the graph of f for all x, y in $[a, c]$.
II. Non-convex, because the line segment between $f(a)$ and $f(c)$ is below the graph.

B2. For $i = 1, \ldots, n$ let $\ell_i(w)$ be convex functions over $w \in \mathbb{R}^d$ (e.g., $\ell_i(w) = (y_i - w^\top x_i)^2$), $\|\cdot\|$ is any norm, and $\lambda > 0$.

a. *[3 points]* Show that

$$\sum_{i=1}^{n} \ell_i(w) + \lambda \|w\|$$

is convex over $w \in \mathbb{R}^d$ (Hint: Show that if f, g are convex functions, then $f(x) + g(x)$ is also convex.)

b. *[1 point]* Explain in one sentence why we prefer to use loss functions and regularized loss functions that are convex.

Solution

a. Let $\alpha \in [0, 1]$, and $w, v \in \mathbb{R}^d$. Then for any $i = 1, \ldots, n$,

$$\begin{aligned}
&\ell_i(\alpha w + (1-\alpha)v) + \lambda \|\alpha w + (1-\alpha)v\| \\
&\leq \alpha \ell_i(w) + (1-\alpha)\ell_i(v) + \lambda\alpha\|w\| + \lambda(1-\alpha)\|v\| \text{ by convexity of } \ell \text{ and triangle inequality} \\
&= \alpha\left(\ell_i(w) + \lambda\|w\|\right) + (1-\alpha)\left(\ell_i(v) + \lambda\|v\|\right)
\end{aligned}$$

Taking the sum over i, we get

$$\begin{aligned}
&\sum_i \ell_i(\alpha w + (1-\alpha)v) + \lambda\|\alpha w + (1-\alpha)v\| \\
&\leq \alpha\left(\sum_i \ell_i(w) + \lambda\|w\|\right) + (1-\alpha)\left(\sum_i \ell_i(v) + \lambda\|v\|\right).
\end{aligned}$$

b. Convex loss function has unique global minimum that can be found with gradient descent. Regularization allows us to trade (some) bias for (some) variance, increasing the overall error.

Lasso on a Real Dataset

Given $\lambda > 0$ and data $\left(x_i, y_i\right)_{i=1}^n$, the Lasso is the problem of solving

$$\arg\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \sum_{i=1}^n (x_i^T w + b - y_i)^2 + \lambda \sum_{j=1}^d |w_j|$$

where λ is a regularization parameter. For the programming part of this homework, you will implement the coordinate descent method shown in Algorithm 1 to solve the Lasso problem in `coordinate_descent_algo.py`. You may use common computing packages (such as `numpy` or `scipy`), but do not use an existing Lasso solver (e.g., of `scikit-learn`).

Algorithm 1: Coordinate Descent Algorithm for Lasso

while *not converged* **do**
  $b \leftarrow \frac{1}{n}\sum_{i=1}^n \left(y_i - \sum_{j=1}^d w_j x_{i,j}\right)$
  for $k \in \{1, 2, \cdots d\}$ **do**
    $a_k \leftarrow 2\sum_{i=1}^n x_{i,k}^2$
    $c_k \leftarrow 2\sum_{i=1}^n x_{i,k}\left(y_i - (b + \sum_{j\neq k} w_j x_{i,j})\right)$
    $w_k \leftarrow \begin{cases} (c_k + \lambda)/a_k & c_k < -\lambda \\ 0 & c_k \in [-\lambda, \lambda] \\ (c_k - \lambda)/a_k & c_k > \lambda \end{cases}$
  end
end

Before you get started, the following hints may be useful:

- Wherever possible, use matrix libraries for matrix operations (not `for` loops).
- There are opportunities to considerably speed up parts of the algorithm by precomputing quantities like a_k before the for loop; you are permitted to add these improvements (and it may save you some time).
- As a sanity check, ensure the objective value is nonincreasing with each step.
- It is up to you to decide on a suitable stopping condition. A common criteria is to stop when no element of w changes by more than some small δ during an iteration. If you need your algorithm to run faster, an easy place to start is to loosen this condition.
- You will need to solve the Lasso on the same dataset for many values of λ. This is called a regularization path. One way to do this efficiently is to start at a large λ, and then for each consecutive solution, initialize the algorithm with the previous solution, decreasing λ by a constant ratio (e.g., by a factor of 2).
- The smallest value of λ for which the solution $\widehat{w}$ is entirely zero is given by

$$\lambda_{max} = \max_{k=1,\ldots,d} 2\left|\sum_{i=1}^n x_{i,k}\left(y_i - \left(\frac{1}{n}\sum_{j=1}^n y_j\right)\right)\right| \tag{1}$$

  This is helpful for choosing the first λ in a regularization path.

A5. We will first try out your solver with some synthetic data. A benefit of the Lasso is that if we believe many features are irrelevant for predicting y, the Lasso can be used to enforce a sparse solution, effectively differentiating between the relevant and irrelevant features. Suppose that $x \in \mathbb{R}^d, y \in \mathbb{R}, k < d$, and data are generated independently according to the model $y_i = w^T x_i + \epsilon_i$ where

$$w_j = \begin{cases} j/k & \text{if } j \in \{1, \ldots, k\} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is noise (note that in the model above $b = 0$). We can see from Equation (2) that since $k < d$ and $w_j = 0$ for $j > k$, the features $k + 1$ through d are irrelevant for predicting y.
Generate a dataset using this model with $n = 500, d = 1000, k = 100$, and $\sigma = 1$. You should generate the dataset such that each $\epsilon_i \sim \mathcal{N}(0, 1)$, and y_i is generated as specified above. You are free to choose a distribution from which the x's are drawn, but make sure standardize the x's before running your experiments.

a. *[10 points]* With your synthetic data, solve multiple Lasso problems on a regularization path, starting at λ_{max} where no features are selected (see Equation (1)) and decreasing λ by a constant ratio (e.g., 2) until nearly all the features are chosen. In plot 1, plot the number of non-zeros as a function of λ on the x-axis (Tip: use `plt.xscale('log')`).

b. *[10 points]* For each value of λ tried, record values for false discovery rate (FDR) (number of incorrect nonzeros in $\widehat{w}$/total number of nonzeros in $\widehat{w}$) and true positive rate (TPR) (number of correct nonzeros in $\widehat{w}$/k). Note: for each j, $\widehat{w}_j$ is an incorrect nonzero if and only if $\widehat{w}_j \neq 0$ while $w_j = 0$. In plot 2, plot these values with the x-axis as FDR, and the y-axis as TPR.

   Note that in an ideal situation we would have an (FDR,TPR) pair in the upper left corner. We can always trivially achieve $(0, 0)$ and $(\frac{d-k}{d}, 1)$.

c. *[5 points]* Comment on the effect of λ in these two plots in 1-2 sentences.

Solution:

- **Part a:** Plot 1.
- **Part b:** Plot 2.
- **Part c:** 1-2 sentence explanation.
- **Code** on Gradescope through coding submission
- **All code you wrote** in the write-up, with correct page mapping.

A6. We'll now put the Lasso to work on some real data in `crime_data_lasso.py`. . Download the training data set "crime-train.txt" and the test data set "crime-test.txt" from the website. Store your data in your working directory, ensure you have the `pandas` library for Python installed, and read in the files with:

```
import pandas as pd
df_train = pd.read_table("crime-train.txt")
df_test = pd.read_table("crime-test.txt")
```

This stores the data as Pandas `DataFrame` objects. `DataFrame`s are similar to Numpy `array`s but more flexible; unlike `array`s, `DataFrame`s store row and column indices along with the values of the data. Each column of a `DataFrame` can also store data of a different type (here, all data are floats). Here are a few commands that will get you working with Pandas for this assignment:

```
df.head()                    # Print the first few lines of DataFrame df.
df.index                     # Get the row indices for df.
df.columns                   # Get the column indices.
df[''foo'']                 # Return the column named ''foo''.
df.drop(''foo'', axis = 1)   # Return all columns except ''foo''.
df.values                    # Return the values as a Numpy array.
df[''foo''].values          # Grab column foo and convert to Numpy array.
df.iloc[:3,:3]               # Use numerical indices (like Numpy) to get 3 rows and cols.
```

The data consist of local crime statistics for 1,994 US communities. The response y is the rate of violent crimes reported per capita in a community. The name of the response variable is `ViolentCrimesPerPop`, and it is held in the first column of `df_train` and `df_test`. There are 95 features. These features include many variables. Some features are the consequence of complex political processes, such as the size of the police force and other systemic and historical factors. Others are demographic characteristics of the community, including self-reported statistics about race, age, education, and employment drawn from Census reports.

The goals of this problem are threefold: (*i*) to encourage you to think about how data collection processes affect the resulting model trained from that data; (*ii*) to encourage you to think deeply about models you might train and how they might be misused; and (*iii*) to see how Lasso encourages sparsity of linear models in settings where d is large relative to n. **We emphasize that training a model on this dataset can suggest a degree of correlation between a community's demographics and the rate at which a community experiences and reports violent crime. We strongly encourage students to consider why these correlations may or may not hold more generally, whether correlations might result from a common cause, and what issues can result in misinterpreting what a model can explain.**

The dataset is split into a training and test set with 1,595 and 399 entries, respectively[1]. We will use this training set to fit a model to predict the crime rate in new communities and evaluate model performance on the test set. As there are a considerable number of input variables and fairly few training observations, overfitting is a serious issue. In order to avoid this, use the coordinate descent Lasso algorithm implemented in the previous problem.

a. *[4 points]* Read the documentation for the originalcversion of this dataset: `http://archive.ics.uci.edu/ml/datasets/communities+and+crime`. Report 3 features included in this dataset for which historical *policy* choices in the US would lead to variability in these features. As an example, the *number of police* in a community is often the consequence of decisions made by governing bodies, elections, and amount of tax revenue available to decision makers.

b. *[4 points]* Before you train a model, describe 3 features in the dataset which might, if found to have nonzero weight in model, be interpreted as *reasons* for higher levels of violent crime, but which might actually be a *result* rather than (or in addition to being) the cause of this violence.

Now, we will run the Lasso solver. Begin with $\lambda = \lambda_{\max}$ defined in Equation (1). Initialize all weights to 0. Then, reduce λ by a factor of 2 and run again, but this time initialize $\hat{w}$ from your $\lambda = \lambda_{\max}$ solution as your initial weights, as described above. Continue the process of reducing λ by a factor of 2 until $\lambda < 0.01$. For all plots use a log-scale for the λ dimension (Tip: use `plt.xscale('log')`).

c. *[4 points]* Plot the number of nonzero weights of each solution as a function of λ.

d. *[4 points]* Plot the regularization paths (in one plot) for the coefficients for input variables `agePct12t29`, `pctWSocSec`, `pctUrban`, `agePct65up`, and `householdsize`.

e. *[4 points]* On one plot, plot the squared error on the training and test data as a function of λ.

f. *[4 points]* Sometimes a larger value of λ performs nearly as well as a smaller value, but a larger value will select fewer variables and perhaps be more interpretable. Inspect the weights $\hat{w}$ for $\lambda = 30$. Which feature had the largest (most positive) Lasso coefficient? What about the most negative? Discuss briefly.

g. *[4 points]* Suppose there was a large negative weight on `agePct65up` and upon seeing this result, a politician suggests policies that encourage people over the age of 65 to move to high crime areas in an effort to reduce crime. What is the (statistical) flaw in this line of reasoning? (Hint: fire trucks are often seen around burning buildings, do fire trucks cause fire?)

[1]The features have been standardized to have mean 0 and variance 1.

Solution:

- **Parts a, b:** 1-2 sentence explanation.
- **Part c:** Plot 1.
- **Part d:** Plot 2.
- **Part e:** Plot 3.
- **Parts f, g:** Answers and 1-2 sentence explanation.
- **Code** on Gradescope through coding submission.
- **All code you wrote** in the write-up, with correct page mapping.

Gradient Descent

A7. Consider an experiment in which you have n observations and corresponding labels $\{x_i, y_i\}_{i=1}^n$ such that $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$. Let $X \in \mathbb{R}^{n \times d}$ be the data matrix containing x_i^T as rows. For this question, assume X has full rank, such that $X^T X$ is invertible.

As we saw before, in linear regression we would like to come up with a model $w \in \mathbb{R}^d$ that minimizes the loss

$$L(w) = ||y - Xw||_2^2 = (y - Xw)^T (y - Xw)$$

We know from lecture that the optimal value for w is $w^* = (X^T X)^{-1} X^T y$. However, we can also use gradient descent instead of calculating this directly. In this question, we will start to prove that for a good choice of the learning rate, the gradient descent algorithm arrives at the same optimal result.

a. *[5 points]* Show (step-by-step) that the gradient $\nabla_w L(w) \in \mathbb{R}^d$ is equal to

$$\nabla_w L(w) = 2(X^T X)w - 2X^T y$$

b. *[3 points]* In gradient descent, we start with a vector w_0 and iteratively update it with the rule

$$w_{k+1} = w_k - \alpha \nabla_{w_k} L(w_k)$$

Using this rule, show that

$$||w_{k+1} - w^*|| = ||(I - 2\alpha X^T X)(w_k - w^*)||$$

(Hint: We have that $\nabla_{w^*} L(w^*) = 0$, thus you can replace $\nabla_{w_k} L(w_k)$ with $\nabla_{w_k} L(w_k) - \nabla_{w^*} L(w^*)$ in your solution, and then expand the definitions.)

c. *[4 points]* Since $X^T X$ is positive semi-definite, all its eigenvalues are nonnegative. Let h be the smallest, and H be the largest eigenvalue of $X^T X$. Using the previous part, show that

$$||w_{k+1} - w^*|| \leq \rho ||w_k - w^*||$$

where $\rho = \max(|1 - 2\alpha h|, |1 - 2\alpha H|)$.

Hints:

- Let λ be the eigenvalue of a matrix $A \in \mathbb{R}^{d \times d}$ that has the largest magnitude. Then for any vector $x \in \mathbb{R}^d$ we have $||Ax|| \leq |\lambda| \cdot ||x||$.)
- If $\gamma_1, \ldots, \gamma_d$ are the eigenvalues of a matrix A, then $\gamma_1 + 1, \ldots, \gamma_d + 1$ are the eigenvalues of $A + I$).

Solution

a.

$$\begin{aligned}\nabla_w L(w) &= \nabla_w (y - Xw)^T (y - Xw) = \left(\nabla_w (y - Xw)^T\right)(y - Xw) + (y - Xw)^T \nabla_w (y - Xw) \\ &= -X^T (y - Xw) + (y - Xw)^T (-X) = X^T Xw - X^T y - y^T X + w^T X^T X = 2X^T Xw - 2X^T y.\end{aligned}$$

b.

$$\begin{aligned}||w_{k+1} - w^*|| &= ||w_k - \alpha \nabla_{w_k} L(w_k) - w^*|| \\ &= ||w_k - \alpha \nabla_{w_k} L(w_k) - (w^* - \alpha \nabla_{w^*} L(w^*))\,|| \\ &= ||w_k - 2\alpha(X^T X)w_k + 2\alpha X^T y - \left(w^* - 2\alpha(X^T X)w^* + 2\alpha X^T y\right)|| \\ &= ||\left(I - 2\alpha X^T X\right) w_k - \left(I - 2\alpha X^T X\right) w^*)\,|| \\ &= ||(I - 2\alpha X^T X)(w_k - w^*)||\end{aligned}$$

c.

$$||w_{k+1} - w^*|| = ||(I - 2\alpha X^T X)(w_k - w^*)|| \leq |\lambda| \cdot ||(w_k - w^*)||,$$

where λ is the eigenvalue of $I - 2\alpha X^T X$ with the largest magnitude. Since the eigenvalues of $X^T X$ are between h and H, the eigenvalues of $I - 2\alpha X^T X$ are between $1 - 2\alpha H$ and $1 - 2\alpha h$. In particular, their absolute values are bounded by $\rho = \max(|1 - 2\alpha h|, |1 - 2\alpha H|)$.

B3. Here, we continue the previous A problem and finish the proof.

a. *[3 points]* By induction on k, show that $||w_k - w^*|| \leq \rho^k ||w_0 - w^*||$ for any positive integer k.

b. *[5 points]* Show that if $\alpha = \frac{1}{h+H}$, then $0 \leq \rho < 1$. Then show that for any error term $\varepsilon > 0$, if

$$k > \log_{1/\rho}\left(\frac{||w_0 - w^*||}{\varepsilon}\right)$$

then $||w_k - w^*|| < \varepsilon$. This shows that w_k gets arbitrarily close to w^* as the number of iterations increases, meaning that the gradient descent method converges to the optimal solution.

Solution

a. The base step of $k = 0$ is clear: $||w_0 - w^*|| \leq ||w_0 - w^*||$. The inductive step is

$$\begin{aligned}||w_k - w^*|| &\leq \rho ||w_{k-1} - w^*|| \text{ from the problem above} \\ &\leq \rho \rho^{k-1} ||w_0 - w^*|| = \rho^k ||w_0 - w^*|| \text{ by the inductive hypothesis}\end{aligned}$$

b. If $\alpha = \frac{1}{h+H}$, then

$$\begin{aligned}\rho &= \max(|1 - 2\frac{h}{h+H}|, |1 - 2\frac{H}{h+H}|) \\ &= \max(|1 - \frac{h}{h+H} - \left(1 - \frac{H}{h+H}\right)|, |1 - \frac{H}{h+H} - \left(1 - \frac{h}{h+H}\right)|) \\ &= \max(|-\frac{h}{h+H} + \frac{H}{h+H}|, |-\frac{H}{h+H} + \frac{h}{h+H}|) \\ &= |\frac{H-h}{h+H}| = \frac{H-h}{h+H} \in [0, 1) \text{ since } H > h \geq 0.\end{aligned}$$

Note that $\rho = 1$ is actually possible and is achieved iff $h = 0$. If $X^T X$ is full rank, then its eigen-values are non-zero, and $\rho < 1$ because $h > 0$. Now, if

$$k > \log_{1/\rho}\left(\frac{||w_0 - w^*||}{\varepsilon}\right) = \log_{\rho}\left(\frac{\varepsilon}{||w_0 - w^*||}\right),$$

then

$$||w_k - w^*|| \leq \frac{\varepsilon}{||w_0 - w^*||}||w_0 - w^*|| = \varepsilon.$$

Administrative

A8.

a. *[2 points]* About how many hours did you spend on this homework? There is no right or wrong answer :)