Homework Assignment 2

SDS 385 Statistical Models for Big Data

Please upload the HW on canvas by 10pm Nov 12th. Please type up your homework using latex. We will not accept handwritten homeworks[1].

1. Consider a design matrix $\boldsymbol{X}$ such that $\boldsymbol{X}$ has orthonormal columns, i.e. $\boldsymbol{X^T X} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the $p \times p$ identity matrix. Consider the following regularization:

$$\min_{\beta} \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda||\boldsymbol{\beta}||_m \tag{1}$$

   (a) Derive the solution to equation (1) for $m = 2$ (ridge regression).

   (b) Derive the solution to equation (1) for $m = 1$ (lasso).

   (c) When $m = 0$, the penalty involves $||\boldsymbol{\beta}||_0 = \sum_{i=1}^{p} \mathbb{1}(\beta_i \neq 0)$.

      i. Is this a convex optimization problem? Why or why not?

      ii. Show that the solution to equation (1) with $m = 0$ is given by $\tilde{\boldsymbol{\beta}}$, where

$$\tilde{\beta}_i = \begin{cases} \boldsymbol{v_i^T y}, & \text{if } |\boldsymbol{v_i^T y}| > \sqrt{2\lambda} \\ 0 & \text{if } |\boldsymbol{v_i^T y}| \leq \sqrt{2\lambda} \end{cases}$$

      This is also called the hard thresholding estimator. $\boldsymbol{v_i}$ is the i^{th} column of $\boldsymbol{X}$.

Solution:

1.a) For the ridge regression where m = 2, the problem becomes to solve the minimization to find the β:

$$\min_{\beta} \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda||\boldsymbol{\beta}||_2$$

which is also find the solution of

$$\min_{\beta}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

Now note that,

$$\frac{\partial(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y})}{\partial\boldsymbol{\beta}} = 2\boldsymbol{X}^T(\boldsymbol{X\beta} - \boldsymbol{y})$$

and

$$\frac{\partial\lambda\boldsymbol{\beta}^T\boldsymbol{\beta}}{\partial\boldsymbol{\beta}} = 2\lambda\boldsymbol{\beta}$$

[1]Two of these homeworks were adapted from A. Dimakis and C. Caramanis's class

Together we get to the first order condition that:

$$2\boldsymbol{X}^T(\boldsymbol{X\beta} - \boldsymbol{y}) + 2\lambda\boldsymbol{\beta} = 0$$

$$\boldsymbol{X^T y} = \boldsymbol{X}^T\boldsymbol{X\beta} + \lambda\boldsymbol{\beta}$$

Isolate the β yields the solution:

$$\hat{\boldsymbol{\beta}}_{ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X^T y}$$

1.b) For the situation of lasso where m = 1, the loss function becomes like:

$$\min_{\beta}\frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda||\boldsymbol{\beta}||_1 \tag{2}$$

In this case, we define $\hat{\boldsymbol{\beta}}_{OLS}$ as the least squares estimator that

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{OLS} &= arg\min_{\beta}\frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X^T y} \\ &= \boldsymbol{X^T y}\end{aligned}$$

then the first term of equation (2) can be expanded as:

$$\begin{aligned}\min_{\beta}\frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) &= \min_{\beta}\frac{1}{2}(\boldsymbol{X^T X} + \boldsymbol{\beta^T\beta} - 2\boldsymbol{y^T X\beta} + \boldsymbol{y^T y}) \\ &= \min_{\beta}(\frac{1}{2}\boldsymbol{\beta^T\beta} - \boldsymbol{y^T X\beta} + \frac{1}{2}\boldsymbol{y^T y}) \\ &= \min_{\beta}(\frac{1}{2}\boldsymbol{\beta^T\beta} - \boldsymbol{y^T X\beta})\end{aligned}$$

We note that $\hat{\boldsymbol{\beta}}_{OLS} = \boldsymbol{X^T y}$ and we need point out that $\boldsymbol{X} \in \Re^{n\times p}$, $\boldsymbol{y} \in \Re^{n\times 1}$. The previous problem of equation(2) can be rewritten as:

$$\min_{\beta}\sum_{i=1}^{p} -\hat{\beta}_{i(OLS)}\beta_i + \frac{1}{2}\beta_i^2 + \lambda|\beta_i|$$

Our objective function is now a sum of objectives, each corresponding to a separate variable $_i$, so they may each be solved individually. And then, we solve the parts, which the sum of parts is equal to the whole.
Fix a certain i, we want to minimize:

$$-\hat{\beta}_{i(OLS)}\beta_i + \frac{1}{2}\beta_i^2 + \lambda|\beta_i| \tag{3}$$

if $\hat{\beta}_{i(OLS)} > 0$, then we must have $\beta_i \geq 0$ since otherwise we could flip its sign and get a lower value for the objective function. Likewie if $\hat{\beta}_{i(OLS)} < 0$, then we must choose $\beta_i \leq 0$.
Case 1: $\hat{\beta}_{i(OLS)} > 0$. Since $\beta_i \geq 0$, equation(3) becomes like

$$L = -\hat{\beta}_{i(OLS)}\beta_i + \frac{1}{2}\beta_i^2 + \lambda\beta_i$$

we need to make

$$\frac{\partial L}{\partial \beta_i} = -\hat{\beta}_{i(OLS)} + 2\beta_i + \lambda = 0$$

and then we get:

$$\beta_i = \hat{\beta}_{i(OLS)} - \lambda$$

And this is only feasible if the right-hand side is non-negative, so in this case the acutal solution is

$$\hat{\beta}_{i(lasso)} = (\hat{\beta}_{i(OLS)} - \lambda)^+ = sgn(\hat{\beta}_{i(OLS)})(|\hat{\beta}_{i(OLS)}| - \lambda)^+$$

Case 2: $\hat{\beta}_{i(OLS)} \leq 0$. This implies that $\beta_i \leq 0$ and thus

$$L = -\hat{\beta}_{i(OLS)}\beta_i + \frac{1}{2}\beta_i^2 - \lambda\beta_i$$

Similar as before in case 1, we can get a result that:

$$\beta_i = \hat{\beta}_{i(OLS)} + \lambda = sgn(\hat{\beta}_{i(OLS)})(|\hat{\beta}_{i(OLS)}| - \lambda)$$

To summarize, in both cases, to ensure the feasibility, we can get a desired form as below:

$$\hat{\beta}_{i(lasso)} = (\hat{\beta}_{i(OLS)} - \lambda)^+ = sgn(\hat{\beta}_{i(OLS)})(|\hat{\beta}_{i(OLS)}| - \lambda)^+$$

1.c) When m = 0, our loss function becomes like

$$\min_{\beta} \frac{1}{2}(\boldsymbol{X\beta - y})^T(\boldsymbol{X\beta - y}) + \lambda\|\beta\|_0$$

1.c.i) To prove if it is a convex problem, we can do it by proving if the function f follows the Jensen inequality rule that for any $0 \leq t \leq 1$,

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y})$$

In our case, I define that:
1. f($\boldsymbol{z}$) = $\|\boldsymbol{z}\|_0$, $\boldsymbol{z} \in R^n$
2. $\|\boldsymbol{y}\|_0$ = number of non-zero elements in vector x = a.
3. $\|\boldsymbol{y}\|_0$ = number of non-zero elements in vector y = b.
Also let us assume $a \geq b$ to keep this simple. Consider 2 cases:
1. Non-zero elements in vector x and y are at totally different places i.e. not a single element overlap at any position i.e. at max a+b=n.
2. All elements overlap in terms of positions i.e. indices where we have non-zero elements in $\boldsymbol{y}_i s$ subset of, indices where we have non-zero element in $\boldsymbol{x}$.
All other cases would be in between these two extreme cases. I will disprove Jensen inequality for these 2 cases.

$$LHS = f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) = \|t\boldsymbol{x} + (1-t)\boldsymbol{y}\|_0$$

Note that since we assume in both the vectors are positive real numbers and so multiplying a positive number(t) to a vector will not affect its count of non-zero elements, this implies:

$$LHS = \|\boldsymbol{x} + \boldsymbol{y}\|_0$$

1).LHS = a+b (since we assume $\boldsymbol{x}$ and $\boldsymbol{y}$ have non-zero elements at completely different positions)
2).LHS = a (since we assume overlap of all element positions are $a \geq b$)

$$\begin{aligned} RHS = tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}) &= t\|\boldsymbol{x}\|_0 + (1-t)\|\boldsymbol{y}\|_0 \\ &= ta + (1-t)b \\ &= ta + b - tb \\ &= b + t(a-b) \end{aligned}$$

Now consider things case wise:
1. LHS - RHS = a + b - b - t(a-b) = (1-t)a + b > thus we can prove that LHS > RHS.
2. LHS - RHS = a - b - t(a-b)) = (1-t)(a-b) > 0 thus we can prove that LHS > RHS, where we have disproved the Jensen inequality rule.
Overall, we can see that, when m=0, it is a non-convex problem.

1.c.ii) Still in our case, we need to solve the problem that

$$arg\min_{\beta} \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda||\boldsymbol{\beta}||_0$$

This question can be re-written as:

$$arg\min_{\beta} \|\boldsymbol{X\beta} - \boldsymbol{y}\|_2^2 + 2\lambda\|\boldsymbol{\beta}\|_0$$

where

$$\boldsymbol{X\beta} = \begin{bmatrix} x_{11}\beta_1 + x_{12}\beta_2 + ... + x_{1p}\beta p \\ x_{21}\beta_1 + x_{22}\beta_2 + ... + x_{2p}\beta p \\ \cdot \\ \cdot \\ \cdot \\ x_{n1}\beta_1 + x_{n2}\beta_2 + ... + x_{np}\beta p \end{bmatrix} \boldsymbol{y} = [y_1, y2, y3, ..., y_n]^T$$

To make it easy, we use β_i to represent each elements in the vector $\boldsymbol{\beta}$, Then we can make our loss function expanded like:

$$f(\beta_i) = \|(\boldsymbol{v_i}^T)^{-1}\beta_i - \boldsymbol{y}\|_2^2 + 2\lambda\|\beta_i\|_0$$

We should know that

$$\|\beta_i\|_0 = \begin{cases} 1 & \text{if } \beta_i \neq 0 \\ 0 & \text{if } \beta_i = 0 \end{cases}$$

So our problem becomes:

$$f(\beta_i) = \begin{cases} \|(\boldsymbol{v_i}^T)^{-1}\beta_i - \boldsymbol{y}\|_2^2 + \lambda & \text{if } \beta_i \neq 0 \\ \|\boldsymbol{v_i}^T\boldsymbol{y}\|_2^2 & \text{if } \beta_i = 0 \end{cases}$$

Thus, for the part that $\beta_i \neq 0$, the minimum can be got when $(\boldsymbol{v_i}^T)^{-1}\beta_i = \boldsymbol{y}$, and also

$$|\boldsymbol{v_i}^T\boldsymbol{y}| > \sqrt{2\lambda}$$

When β_i =0, $\|\boldsymbol{v_i}^T\boldsymbol{y}\|_2^2 < 2\lambda$, hereby,

$$|\boldsymbol{v_i}^T\boldsymbol{y}| \leq \sqrt{2\lambda}$$

Thus, we get the hard thresholding estimator that:

$$\hat{\beta}_i = arg\min_{\beta} f(\beta_i) = \begin{cases} \boldsymbol{v_i}^T\boldsymbol{y} & \text{if } |\boldsymbol{v_i}^T\boldsymbol{y}| > \sqrt{2\lambda} \\ 0 & \text{if } |\boldsymbol{v_i}^T\boldsymbol{y}| \leq \sqrt{2\lambda} \end{cases}$$

2. Consider the following problem of fused Lasso, or total variation de-noising.

$$\min_x \frac{1}{2}\|x - z\|_2^2 + \lambda \sum_{i=1}^{n-1} |x_{i+1} - x_i|.$$

Here, z is a noisy signal and λ is non-negative.

(a) Write this problem as follows:

$$\min_x \frac{1}{2}\|x - z\|_2^2 + \lambda \|Dx\|_1.$$

where D is a $n-1 \times n$ matrix. What is D?

(b) Write down the subgradient of the objective function.

(c) Implement the subgradient descent algorithm. On the noisy.txt dataset, apply the algorithm and show the convergence plot against number of iterations. Play with different stepsizes and discuss how that affects the convergence.

(d) The problem can be written as

$$\begin{aligned} &\min_x \frac{1}{2}\|x - z\|_2^2 + \lambda \|y\|_1 \\ &\text{s.t. } y = Dx \end{aligned} \tag{4}$$

Show that the dual of the above problem is:

$$\begin{aligned} &\min_u \frac{1}{2} u^T D D^T u - u^T D z. \\ &\text{s.t. } \|u\|_\infty \leq \lambda \end{aligned} \tag{5}$$

(e) Now implement the proximal gradient algorithm for the dual problem. In order to do this, you may need to look at the proximal operator given below:

$$\text{prox}_t(x) = \arg\min_z \frac{\|x - z\|^2}{2t} + I_\lambda(z)$$

where $I_\lambda(z) = 0$ if $|z| \leq \lambda$ and ∞ otherwise. In this case, the proximal operator gets reduced to a projection operator. On the same datset, apply this algorithm and show the convergence plot against the number of iterations. How does this compare with the subgradient method you implemented before?

(f) Finally, for the dual proximal gradient method, show the effect of different λ values by plotting the denoised curve superimposed on the original noisy curve.

Solution:

2.a) It is not hard to generate D from $\sum_{i=1}^{n-1} |x_{i+1} - x_i|$ that:

$$DX = D_{(n-1)\times(n)} \begin{bmatrix} x_1 \\ x_2 \\ : \\ : \\ x_n \end{bmatrix}_{n\times 1} = x_2 - x_1 + x_3 - x_2 + ... + x_n - x_{n-1}$$

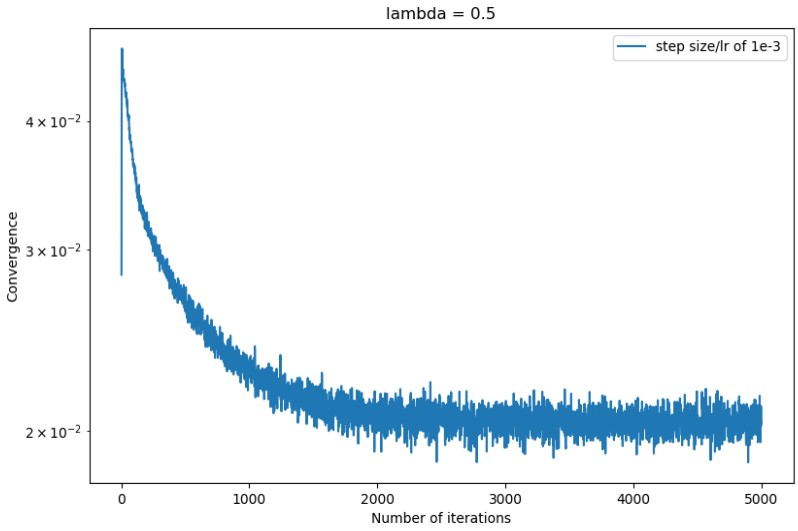
Thus,

$$D = \begin{bmatrix} -1 & 1 & 0 & ... & 0 & 0 \\ 0 & -1 & 1 & ... & 0 & 0 \\ : & & & & & \\ : & & & & & \\ 0 & 0 & 0 & ... & -1 & 1 \end{bmatrix}_{(n-1)\times n}$$

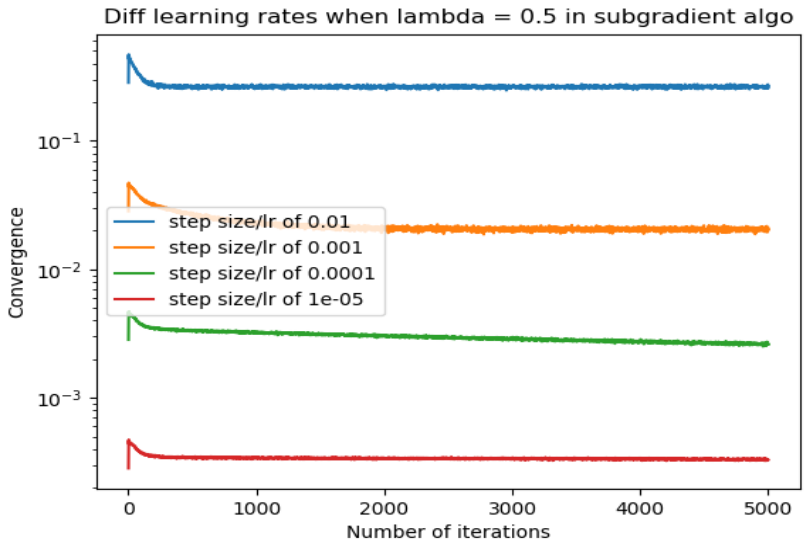
2.b) The subgradient of the objective function is:

$$\frac{\partial F(x)}{\partial x} = \frac{\partial(\frac{1}{2}\|x - z\|_2^2 + \lambda\|Dx\|_1)}{\partial x} = x - z + \lambda D^T sgn(Dx)$$

2.c) First, I tried the learning rate at 0.001. It does not converge for the iterations in 5000 when I set the tolerance of 0.00001.(Python code is attached with the assignment). And the result is showed as below:



Then different step sizes are plotted together to check the effect of step size. The figure is showed as below:



We can see from the figure above that when step size is smaller, the convergence is smaller proportionally. But they tend to be in the same shape.

2.d) We can re-write the fused lasso optimization problem as:

$$arg\min_x \frac{1}{2}\|x-z\|_2^2 + \lambda\|y\|_1$$
$$\text{s.t. } y = Dx$$

Then the Lagrangian becomes:

$$\begin{aligned}\Lambda(\beta, z, u) &= \frac{1}{2}\|z-x\|_2^2 + \lambda\|y\|_1 + u^T(Dx-y) \\ &= \frac{1}{2}\|z-x\|_2^2 + u^TDx + \lambda\|y\|_1 - u^Ty\end{aligned}$$

The dual function $g(u)$, is the infimum of Λ over β and z. Notice that if $\|u\|_\infty > \lambda$ then $g(u) = -\infty$. Otherwise, the infimum over of $\lambda\|y\|_1 - u^Ty$ is equal to zero.
In the case when $\|z\|_\infty \leq \lambda$, the value of $g(u)$ depends only on the x infimum, given by:

$$arg\min_x \frac{1}{2}\|x-z\|_2^2 + u^TDx = -\frac{1}{2}\|z-D^Tu\|_2^2$$

Finally, the dual problem is to find the minimum of $g(u)$ over all u. We can change the formula by stating it as:

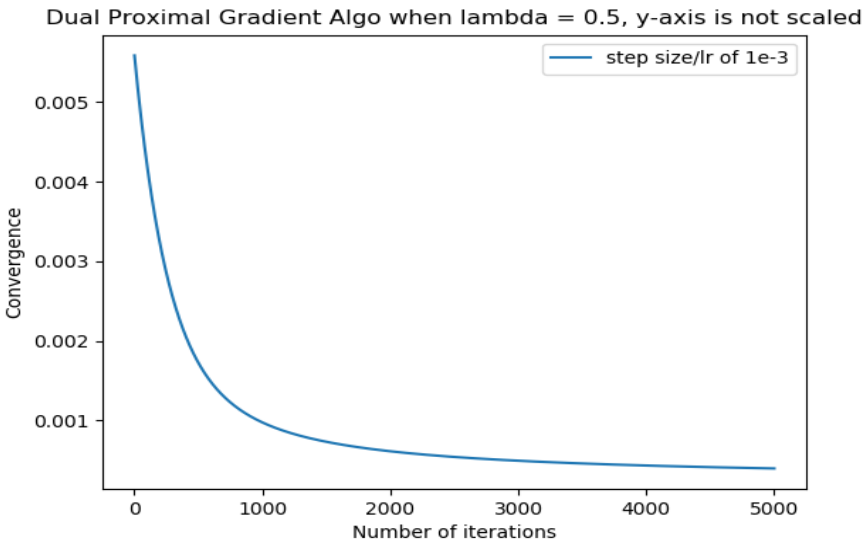
$$\min_u \frac{1}{2}\|z-D^Tu\|_2^2 = \min_u \frac{1}{2}(z^Tz - 2u^TDz + u^TDD^Tu)$$
$$\text{s.t. } \|u\|_\infty \leq \lambda$$

Since it is about the u optimization, we can get rid of the term without u, which is equal to the equation we need to prove that the dual problem of fused lasso is also:

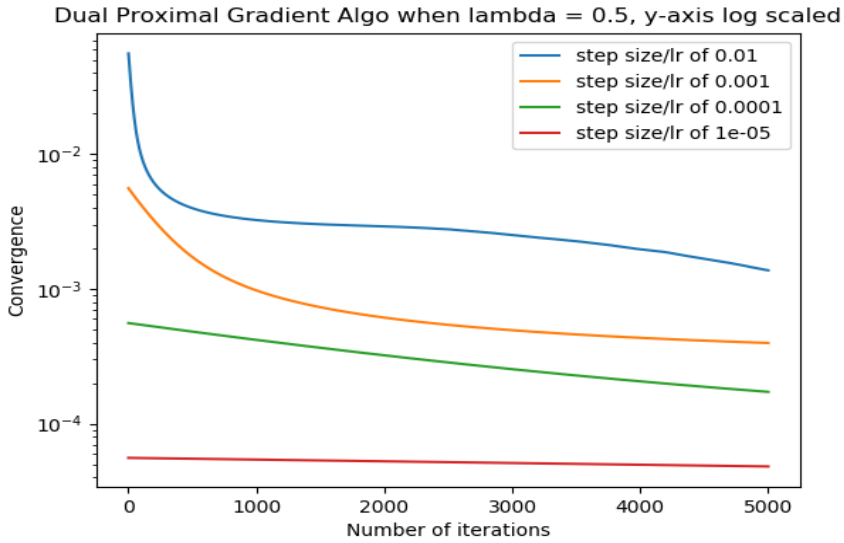
$$\min_u(-u^TDz + \frac{1}{2}u^TDD^Tu)$$

$$\text{s.t. } \|u\|_\infty \leq \lambda$$

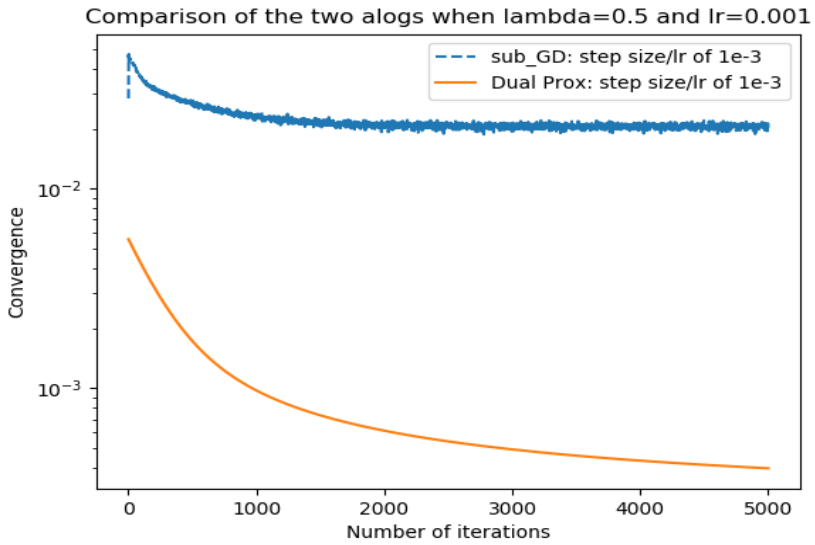
2.e) Since t here is a constant, so the code does not include it here. The plot below is also set when $\lambda = 0.5$, learning rate at 0.001 and tolerance at 0.00001. It does not converge at this parameters setting.



Next figure shows the algorithm convergence in different step sizes.

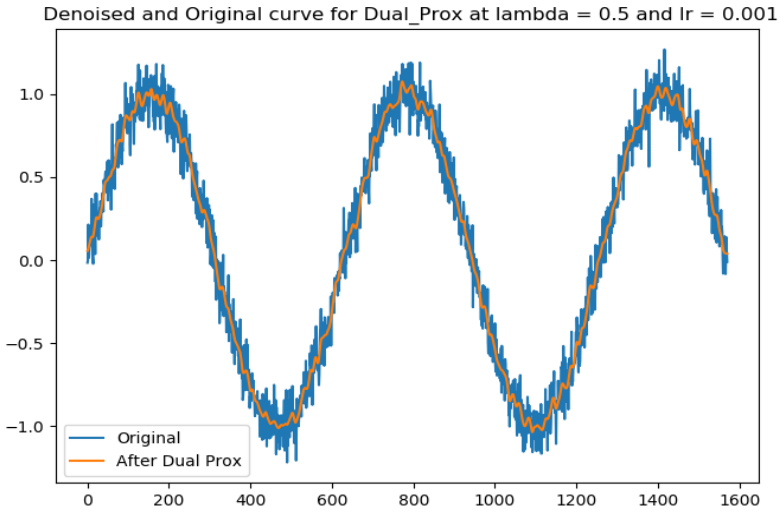


To compare with the subgradient method, I plot the two algorithms together with specific parameter settings that lr=0.001.

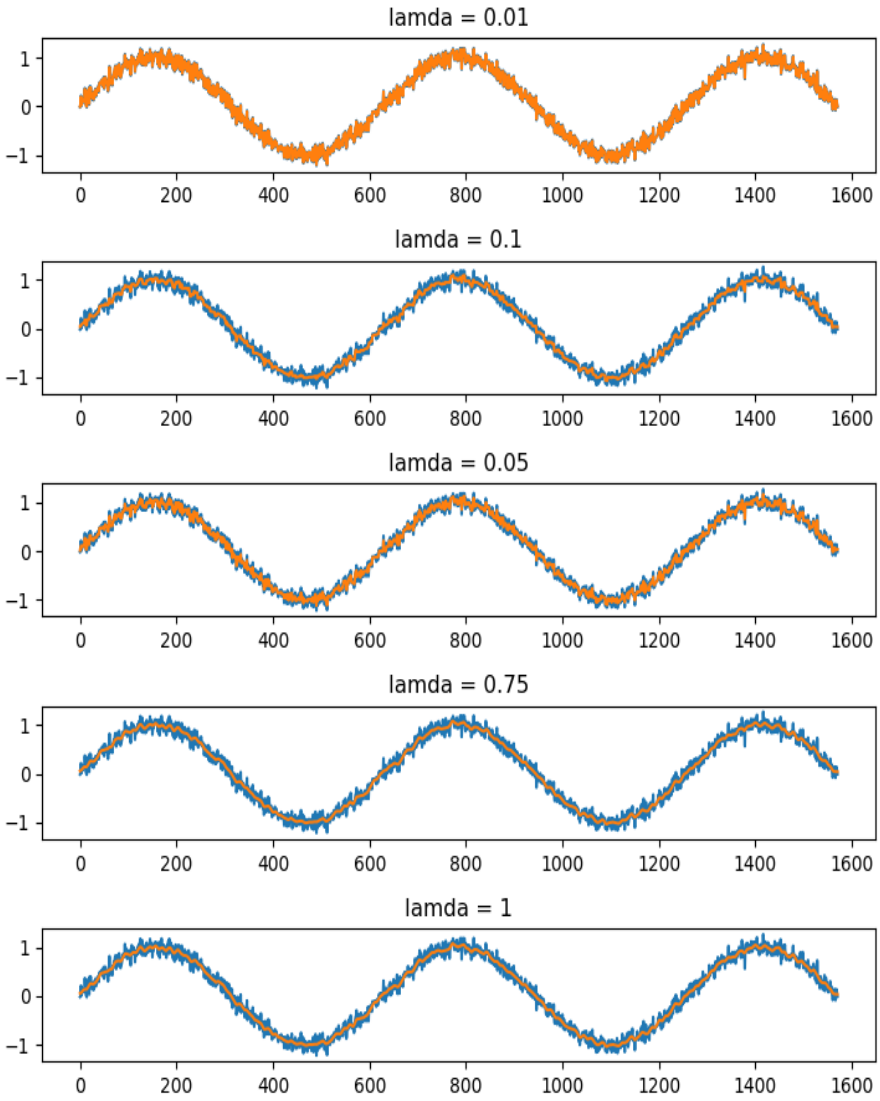


We can see from the figure below that, at this point, the dual proximal gradient method works better than the subgradient method.

2.f) The figure below shows a set-up learning rate and $\lambda = 0.5$.



For different λ, the result is shown below.



For the figure above, blue lines shows the original data while orange ones are denoised curve. All the learning rates are at 0.001 for these different λs. We can see from it that it is getting smoother when λ is getting bigger.

3. Download the dataset articles-1000.txt (1.6MB), which contains 1000 articles. Each row corresponds to an article, represented by an ID (e.g., t120) and its content (e.g., The Supreme Court in Johnnesberg on Friday...).

   (a) Convert each article into a set of 2-shingles (bigrams). The goal is to map each article ID to a set[2] of IDs of shingles that article contains. You can do this by going over the data once as follows: for each article, first split[3] it into a list of words, remove the stopwords[4] , generate a list of 2-shingles, and then hash each shingle to a 32-bit integer (i.e., the shingle ID) using CRC32 hash. The resulting

[2]You can use Python sets to do that. A set should contain unique elements.
[3]Use Python String split() method to do that– no need to remove the punctuations.
[4]you can remove the stop words specified by stopwords.words("english") from nltk.corpus.

shingle IDs range from 0 to $2^{32}-1$. What is the total number of unique shingles across all the books? What is the average number of shingles present per book?

Here is an example of a 2-shingle and its hashed value (i.e., the shingle ID):

```
import binascii
shingle = "machine learning"
shingleID = binascii.crc32(shingle) & 0xffffffff
```

(b) Generate MinHash signatures using 10 hash functions, where each is as follows: $h(x) = c_1 x + c_2 \bmod p$.

Set $p = 4294967311$, which is a prime number larger than $2^{32}-1$. Uniformly sample 10 values of $c_1, c_2 \in \{1, 2, ..., p-1\}$ and compute the corresponding MinHash signatures.

For the 1st article, compare its signature vector with that of the rest of the articles and estimate their Jaccard similarity (i.e., compute the percentage of signatures that are equal). Find the book that has the largest (estimated) similarity with the 1st book then compute the actual Jaccard similarity between the two books (based on the computed shingle sets). Your result should be a triplet (bookID, estimatedJaccard, trueJaccard).

Hint: remember, you do not need to generate 10 permutations. Use the trick we learned in class, you can also find it in Section 3.3.5 of the "Mining of Massive Datasets" book linked from the class website.

(c) **Amplification** Use the LSH technique in Section 3.4 of the same book, construct $n = br$ MinHash signatures, where b is the number of bands, and r is the number of hash values in each band. Find all the articles that are "similar" to the 1st article (i.e., articles that agree in at least one band of signatures), and put it into a set called S (excluding the 1st article itself). Let $Sim(i, j)$ be the actual Jaccard similarity between articles i and j. Given a threshold t, define the percentage of false positives (fp) in set S as

$$\text{False positives} = \frac{|\{i \in S : Sim(i, 1) < t\}|}{|S|}.$$

Set t = 0.8, plot fp as a function of b (for $b = 1, 3, 5, 7, 9$ with $r = 2$). Similarly, plot fp as a function of r (for $r = 1, 3, 5, 7, 9$ with $b = 10$). For each (b, r) pair, plot the average value of fp over 10 realizations. Give a short comparison of the two.

(d) Find the 5 most similar article pairs without any approximation. How long did it take? Now find the 5 most similar article pairs using LSH (with b,r as hyperparameters). Compare your results for different (b, r) pairs.

Solution:

3.a)

```
problem a:
total unique shingle  131266
average single per article  168.694
average unique single per article  131.266
```

If I change the word to lower case then remove the stopwords, the result is different

from not lowering cases.
If not lower the case in the articles: the result is showed. While if I lower it, the unique shingles becomes to 129121. I choose not lowering when do next questions.

3.b)

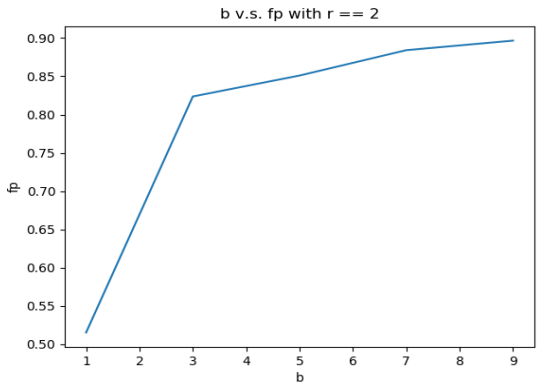
```
problem b:
hw2_prob3.py:73: RuntimeWarning: overflow encountered in long_scalars
  h = (c1 * x + c2) % p
802 1.0 1.0
```

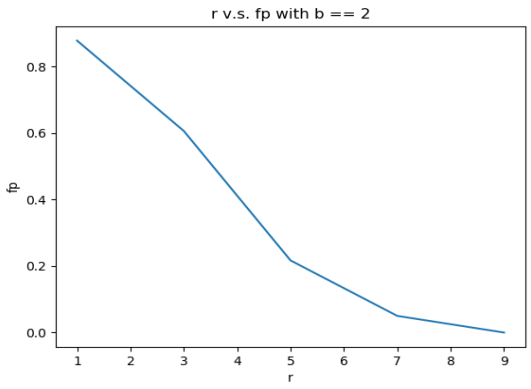
The index from the output is 802, of which the article id is "t7998". The estimated similarity is 1, and its Jaccard similarity is also 1.

3.c)

```
problem c:
b: 1, r: 2
b: 3, r: 2
b: 5, r: 2
b: 7, r: 2
b: 9, r: 2
b: 2, r: 1
b: 2, r: 3
b: 2, r: 5
b: 2, r: 7
b: 2, r: 9
```

Each (b,r) pair is shown above.





The left figure shows the plot when b = 1,3,5,7,8 with r = 2 and t = 0.8. The right figure above shows the fp-r function that when r = 1,3,5,7,9 with b = 10.

3.d)

```
problem d:
0 1000
100 1000
200 1000
300 1000
400 1000
500 1000
600 1000
700 1000
800 1000
900 1000
time elapsed for finding top 5 similar pairs without any approximation: 13.212219953536987
time elapsed for finding top 5 similar pairs using LSH with b: 1, r: 1: 0.36277198791503906
time elapsed for finding top 5 similar pairs using LSH with b: 1, r: 3: 0.6701269149780273
time elapsed for finding top 5 similar pairs using LSH with b: 1, r: 5: 1.0447969436645508
time elapsed for finding top 5 similar pairs using LSH with b: 1, r: 7: 1.3780429363250732
time elapsed for finding top 5 similar pairs using LSH with b: 1, r: 9: 1.888383388519287
time elapsed for finding top 5 similar pairs using LSH with b: 3, r: 1: 1.305314064025879
time elapsed for finding top 5 similar pairs using LSH with b: 3, r: 3: 2.1264357566833496
time elapsed for finding top 5 similar pairs using LSH with b: 3, r: 5: 3.5825958251953125
time elapsed for finding top 5 similar pairs using LSH with b: 3, r: 7: 4.674108982086182
time elapsed for finding top 5 similar pairs using LSH with b: 3, r: 9: 5.888751029968262
time elapsed for finding top 5 similar pairs using LSH with b: 5, r: 1: 2.094040870666504
time elapsed for finding top 5 similar pairs using LSH with b: 5, r: 3: 3.5541739463806152
time elapsed for finding top 5 similar pairs using LSH with b: 5, r: 5: 5.15767502784729
time elapsed for finding top 5 similar pairs using LSH with b: 5, r: 7: 7.461442947387695
time elapsed for finding top 5 similar pairs using LSH with b: 5, r: 9: 9.477487087249756
time elapsed for finding top 5 similar pairs using LSH with b: 7, r: 1: 2.66694974899292
time elapsed for finding top 5 similar pairs using LSH with b: 7, r: 3: 4.657615900039673
time elapsed for finding top 5 similar pairs using LSH with b: 7, r: 5: 7.5261430740356445
time elapsed for finding top 5 similar pairs using LSH with b: 7, r: 7: 10.17553186416626
time elapsed for finding top 5 similar pairs using LSH with b: 7, r: 9: 13.817530870437622
time elapsed for finding top 5 similar pairs using LSH with b: 9, r: 1: 3.550344944000244
time elapsed for finding top 5 similar pairs using LSH with b: 9, r: 3: 6.550416946411133
time elapsed for finding top 5 similar pairs using LSH with b: 9, r: 5: 10.03914999961853
time elapsed for finding top 5 similar pairs using LSH with b: 9, r: 7: 13.803581953048706
time elapsed for finding top 5 similar pairs using LSH with b: 9, r: 9: 17.35904288291931
(base) Ambers-MacBook-Pro:HW2 ippjune$
```

It cost 13.21221995s to find the top 5 similar pairs without any approximation. The comparison is also shown above with different combinations of b and r.

Notation at very last)

All the codes are attached in the assignment file with two separate python files in the file names of "hw2_prob2.py" and "hw2_prob3.py"