# Homework Assignment 2

SDS 385 Statistical Models for Big Data

Answer

Please upload the HW on canvas by 10pm Nov 14th. 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).

$$f(\beta) = \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda||\boldsymbol{\beta}||_2^2$$
$$\frac{\partial f}{\partial \boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{X\beta} - \boldsymbol{y}) + 2\lambda\boldsymbol{\beta} = 0$$
$$\boldsymbol{X}^T\boldsymbol{X}\beta - \boldsymbol{X}^T y + 2\lambda\boldsymbol{\beta} = 0$$
$$(1 + 2\lambda)\boldsymbol{\beta} = \boldsymbol{X}^T y$$
$$\boldsymbol{\beta} = \frac{1}{1 + 2\lambda}\boldsymbol{X}^T y$$

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

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

$$\begin{aligned}
f(\beta) &= \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda||\boldsymbol{\beta}||_1 \\
&= \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda\sum_{i=1}^{p}|\beta_i| \\
\frac{\partial f}{\partial \boldsymbol{\beta}} &= \boldsymbol{X}^T(\boldsymbol{X\beta} - \boldsymbol{y}) + \lambda\delta\sum_{i=1}^{p}|\beta_i| \\
&= \boldsymbol{\beta} - \boldsymbol{X}^T y + \lambda\delta\sum_{i=1}^{p}|\beta_i| \\
\frac{\partial f}{\partial \beta_i} &= \beta_i - \boldsymbol{v_i}^T y + \lambda\delta_i\sum_{j=1}^{p}|\beta_j|
\end{aligned}$$

$\boldsymbol{v_i}$ is the $i^{th}$ column of $\boldsymbol{X}$

$$\delta_i\sum_{j=1}^{p}|\beta_j| = \begin{cases} 1, & \text{if } \beta_i > 0 \\ -1, & \text{if } \beta_i < 0 \\ [-1,1], & \text{if } \beta_i = 0 \end{cases}$$

$$\frac{\partial f}{\partial \beta_i} = \begin{cases} \beta_i - v_i^T y + \lambda, & \text{if } \beta_i > 0 \\ \beta_i - v_i^T y - \lambda, & \text{if } \beta_i < 0 \\ \in [\beta_i - v_i^T y - \lambda, \beta_i - v_i^T y + \lambda], & \text{if } \beta_i = 0 \end{cases}$$

$$\begin{gathered}
(1)\ \beta_i - v_i^T y + \lambda = 0, \beta_i > 0 \\
\beta_i = v_i^T y - \lambda, \text{if } v_i^T y > \lambda \\
(2)\ \beta_i - v_i^T y - \lambda = 0, \beta_i < 0 \\
\beta_i = v_i^T y + \lambda, \text{if } v_i^T y < -\lambda \\
(3)\ \beta_i = 0, v_i^T y \in [-\lambda, \lambda], -v_i^T y \in [-\lambda, \lambda] \\
-v_i^T y - \lambda \in [-2\lambda, 0], -v_i^T y + \lambda \in [0, 2\lambda] \\
0 \in [-v_i^T y - \lambda, -v_i^T y + \lambda]
\end{gathered}$$

$$\beta_i = \begin{cases} v_i^T y - \lambda, & \text{if } v_i^T y > \lambda \\ v_i^T y + \lambda, & \text{if } v_i^T y < -\lambda \\ 0, & \text{if } v_i^T y \in [-\lambda, \lambda] \end{cases}$$

(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?
Assume $\boldsymbol{x} = (0,1), \boldsymbol{y} = (1,0)$

$$\begin{gathered}
||\boldsymbol{x}||_0 = ||\boldsymbol{y}||_0 = 1 \\
||t\boldsymbol{x} + (1-t)\boldsymbol{y}||_0 = ||(1-t,t)||_0 = 2 \text{ for } 0 < t < 1 \\
||t\boldsymbol{x} + (1-t)\boldsymbol{y}||_0 > t||\boldsymbol{x}||_0 + (1-t)||\boldsymbol{y}||_0 = 1
\end{gathered}$$

This contradicts the definition of convex function, so $||\boldsymbol{\beta}||_0$ is not convex. Therefore, it is not a convex optimization problem.

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 $\boldsymbol{X}$.

$$\begin{aligned} f(\beta) &= \frac{1}{2}(\boldsymbol{X\beta - y})^T(\boldsymbol{X\beta - y}) + \lambda\sum_{i=1}^{p} 1(\beta_i \neq 0) \\ &= \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X\beta} - \boldsymbol{y}^T\boldsymbol{X\beta} + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} + \lambda\sum_{i=1}^{p} 1(\beta_i \neq 0) \\ &= \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} - \boldsymbol{y}^T\boldsymbol{X\beta} + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} + \lambda\sum_{i=1}^{p} 1(\beta_i \neq 0) \\ &= \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} + \frac{1}{2}\sum_{i=1}^{p} \beta_i^2 - 2v_i^T\boldsymbol{y}\beta_i + 2\lambda I_{(\beta_i \neq 0)} \end{aligned}$$

$$\text{Let } g(\beta_i) = \beta_i^2 - 2v_i^T\boldsymbol{y}\beta_i + 2\lambda I_{(\beta_i \neq 0)}$$

$$\text{if } \beta_i = 0, g(\beta_i) = 0$$

$$\text{if } \beta_i \neq 0, g(\beta_i) = \beta_i^2 - 2v_i^T\boldsymbol{y}\beta_i + 2\lambda = (\beta_i - v_i^T\boldsymbol{y})^2 + 2\lambda - (v_i^T\boldsymbol{y})^2$$

$$\beta_i = v_i^T\boldsymbol{y}, \min\ g(\beta_i) = 2\lambda - (v_i^T\boldsymbol{y})^2$$

$$\text{if } |\boldsymbol{v_i^T y}| > \sqrt{2\lambda}, 2\lambda - (v_i^T\boldsymbol{y})^2 < 0, \text{so } \tilde{\beta}_i = \boldsymbol{v_i^T y}$$

$$\text{if } |\boldsymbol{v_i^T y}| \leq \sqrt{2\lambda}, 2\lambda - (v_i^T\boldsymbol{y})^2 \geq 0, \text{so } \tilde{\beta}_i = 0$$

The hard thresholding estimator is proved.

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 $\lambda$ 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 $D$?

*Solution.* The matrix $D$ is $D_{i,i} = -1, D_{i,i+1} = 1, \forall i = 1, 2, \ldots, n-1$; $D_{i,j} = 0$ otherwise, i.e.

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

■

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

*Solution.* By the calculus of subdifferential, if $g(x) = f(Ax+b)$, then $\partial g(x) = A^T \partial f(Ax+b)$. Hence, the subgradients are $\partial f(x) = x - z + \lambda D^T p$,

where $p_i = \begin{cases} 1 & (Dx)_i > 0 \\ -1 & (Dx)_i < 0 \\ [-1,1] & (D_x)_i = 0 \end{cases}, \forall i = 1, 2 \dots n.$ ■

(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.

*Solution.* See "p1_fused_lasso.html" for codes and results. Figure 1 shows the convergence rates with different learning rates. For both plots (either $\lambda = 0.01$ or 1), when $\eta$ is chosen appropriately, either $\eta = 0.01, 0.1$ or with decreasing schedules (linear decay or square-root decay), the loss converges quickly. If $\eta$ is too large ($\eta = 1$ in the right plot), it diverges; If $\eta$ is too small ($\eta = 0.001$), it converges much slower. Notice that subgradient descent does not always con-verges to the same point (for example, global minimum or critical points) and sometimes even does not converge.

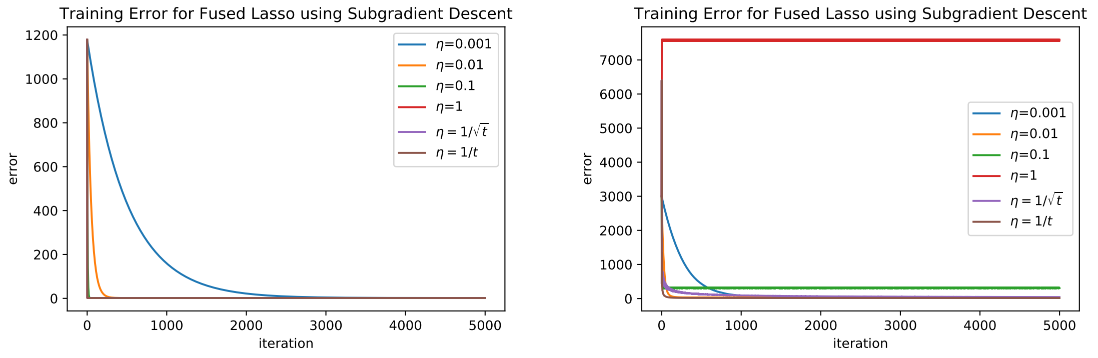


Figure 1: Plot for (a): Convergence Rates Using Subgradient Descent for Different Stepsizes. Left: $\lambda = 0.01$; Right: $\lambda = 1$.

■

(d) The problem can be written as

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

Show that the dual of the above problem is:

$$\min_u \frac{1}{2} u^T D D^T u - u^T D z. \quad \text{s.t. } \|u\|_\infty \le \lambda \tag{3}$$

*Proof.* The Lagrangian is

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

The dual function $g(u) = \inf_{x,y} \mathcal{L}(x, y, u)$.
If $\|u\|_\infty > \lambda$ and $i = \arg\max_k |u_k|$, then let $y'_i = sign(u_i) \cdot \infty$ and $y'_j = 0, \forall j \neq i$, we have $\lambda\|y'\|_1 - u^T y' = (\lambda - \|u\|_\infty) \cdot \infty = -\infty$, since $\lambda - \|u\|_\infty < 0$. Hence, $\inf_y \lambda\|y\|_1 - u^T y \leq \lambda\|y'\|_1 - u^T y' = -\infty$.
Otherwise, consider $\|u\|_\infty \leq \lambda$, we have $\inf_y \lambda\|y\|_1 - u^T y = 0$ since $\lambda\|y\|_1 - u^T y \geq (\lambda - \|u\|_\infty)\|y\|_1 \geq 0$ (by Cauchy-Schwarz inequality: $u^T y \leq \|u\|_\infty\|y\|_1$) and the equality holds when $y = 0$. Then, $g(u) = \inf_x \frac{1}{2}\|x - z\|_2^2 + u^T Dx$, and take the derivative over $x$, we have

$$x - z + D^T u = 0 \Rightarrow x = z - D^T u$$

Hence, plug in $x = z - D^T u$, the dual problem $\max_u g(u)$ is

$$\begin{aligned}&\max_u \frac{1}{2}\|z - D^T u - z\|_2^2 + u^T D(z - D^T u) \quad s.t. \quad \|u\|_\infty \leq \lambda \\ \Leftrightarrow &\min_u \frac{1}{2} u^T D D^T u - u^T Dz \quad s.t. \quad \|u\|_\infty \leq \lambda\end{aligned}$$

□

(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 $\infty$ 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?

*Solution.* See "p1_fused_lasso.html" for codes and results. In theory, convergence rate for subgradient descent is $\epsilon \sim \frac{1}{\sqrt{T}}$ and for proximal gradient descent is $\epsilon \sim \frac{1}{T}$. Figure 2 also shows proximal gradient descent converges much faster than subgradient descent. Note that the initial and final losses are different since the loss objectives are different for primal and dual problems.

■

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

*Solution.* See "p1_fused_lasso.html" for codes and results. Figure 3 shows the scale of the regularization parameter $\lambda$ determines the denoising effect. Large $\lambda$ ($\lambda = 1, 10$) has a better denoising effect than small $\lambda$ ($\lambda = 0.01, 0.1$).

■

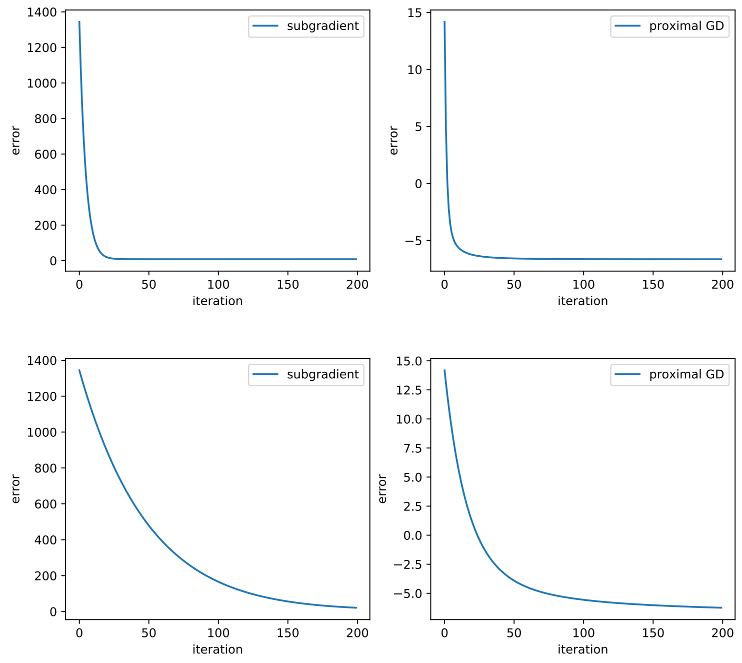


Figure 2: Plot for (e): Convergence rates using subgradient descent for primal problem and proximal gradient for dual problem using different stepsizes with $\lambda = 0.1$. For both Upper and Lower, left is subgradient descent and right is proximal gradient descent. Upper: $\eta = 0.1$; Lower: $\eta = 0.01$.

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 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
```

   *Solution.* Total number of unique singles is 131266 and average is 168.694 per book. (See ”p4_hash.html” for codes and results.)

   ■

[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.

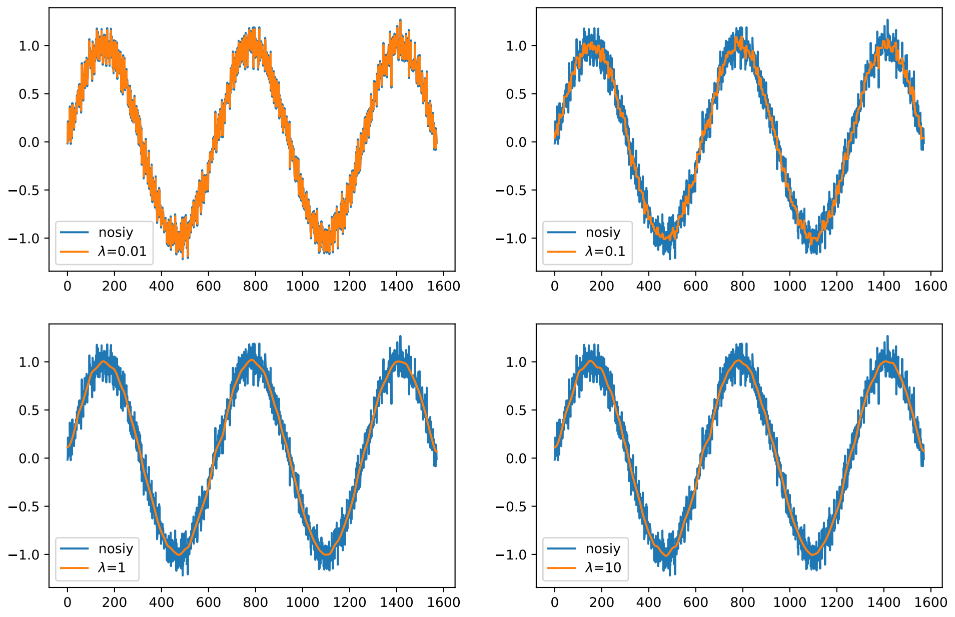


Figure 3: Plot for (f). Denoised curve superimposed on the original noisy curve using dual proximal gradient descent with different$\lambda$ values: $\lambda = 0.01, 0.1, 1, 10$.

(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.*

*Solution.* The triplet (bookID = t7998, estimatedJaccard = 1.0, trueJaccard = 1.0). (See "p4_hash.html" for codes and results.) ■

(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.

*Solution.* See ”p4_hash.html” for codes and results. Figure 4 shows the plots of average false positive rate as a function of $b$ with fixed $r$ and as a function of $r$ with fixed $b$, respectively.
For fixed $r = 2$, as $b$ increases, the false positive rate increases. Since there are more chances (i.e. more bands) that the signatures agree in at least one band, the match rule becomes ”loose”. Then, there are more candidates, hence, the number of candidate pairs do not match each other considering the whole article (false candidates) grow. Note that since from Figure 5, there is only one document with which the Jaccard is larger than 0.8, the rate depends on how many candidate it finds.
For fixed $b = 10$, as $r$ increases, the false positive rate decreases. Since if $r$ is large, two articles must match more shingles in a band, then, the match rule becomes ”loose”. Hence, the number of candidates decreases, the false positive candidates decrease correspondingly. ■

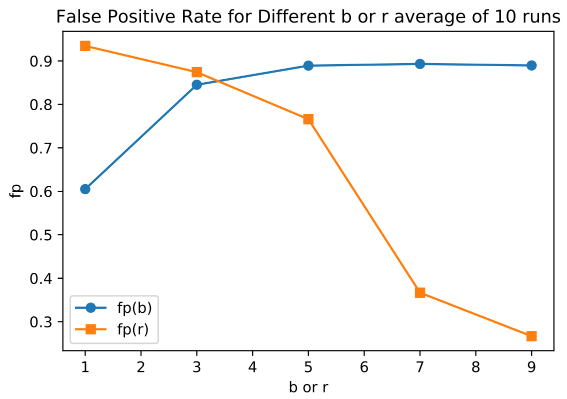


Figure 4: Plot for (c). False positive rate for different b or r over average of 10 runs. Blue: $b = 1,3,5,7,9$ with $r = 2$; Orange: $r = 1,3,5,7,9$ with $b = 10$.

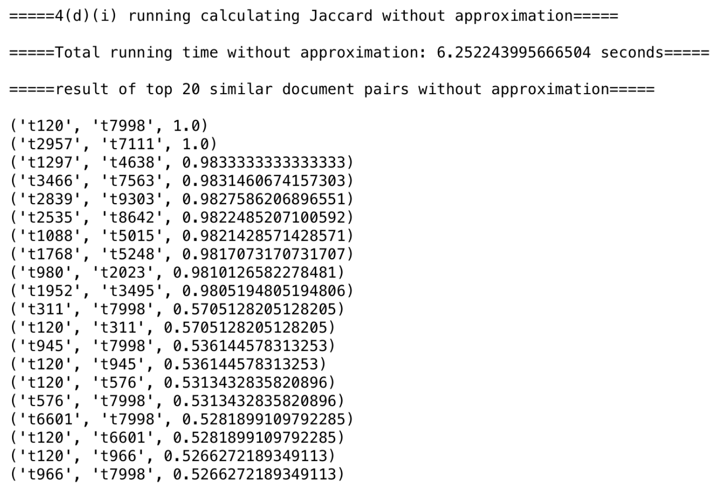
(d) Find the twenty most similar article pairs without any approximation. How long did it take? Now find the twenty most similar article pairs using LSH (with b,r as hyperparameters). Compare your results for different $(b,r)$ pairs.

*Solution.* The results without approximation are in Figure 5. The total running time is 6.25 seconds, which is quite long since it needs to compare $\frac{1000\times(1000-1)}{2} =$

499, 500 pairs. If $n$ grows, the running time will be much longer. Figure 6 shows that the lsh approximation with pair $b = 7$ and $r = 1$, which has the same top20 results as true Jaccard, takes only 0.55 seconds, which is almost 1/11 of original comparison without any approximation.

Other selected results using lsh are in Figure 7. When $r$ is big, the matching rule is more "strict", less than 20 pairs are selected as "similar articles". The corresponding Jaccard score are high (which means they are true similar documents). When $r$ is small, the rule is "loose" now, so the corresponding true Jaccard is not that high. When $b$ is big, for example $b = 10$, it takes more time to run since it needs to hash more bands and compare more candidates.

More results of different $(b, r)$ pairs, where $b \in \{1, 3, 5, 7, 9, 10\}$ and $r \in \{1, 2, 3, 5, 7, 9\}$, are in "p4_hash.html". ■



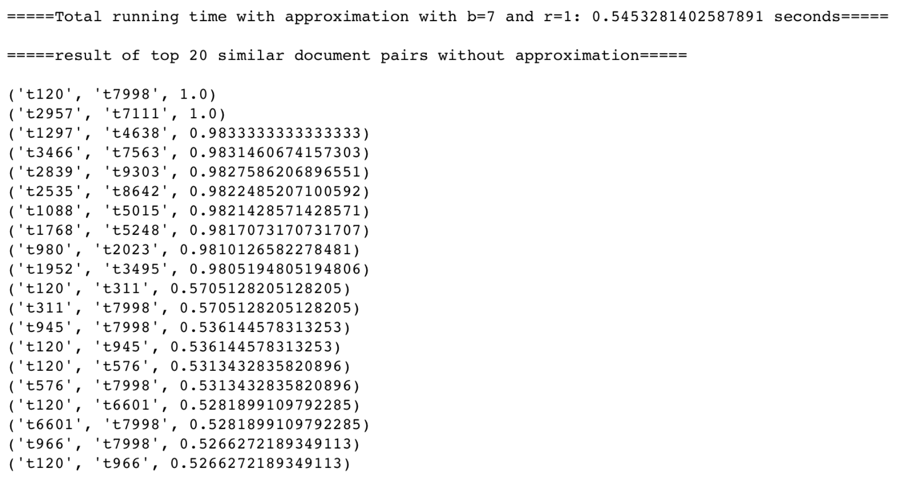
```
=====4(d)(i) running calculating Jaccard without approximation=====

=====Total running time without approximation: 6.252243995666504 seconds=====

=====result of top 20 similar document pairs without approximation=====

('t120', 't7998', 1.0)
('t2957', 't7111', 1.0)
('t1297', 't4638', 0.9833333333333333)
('t3466', 't7563', 0.9831460674157303)
('t2839', 't9303', 0.9827586206896551)
('t2535', 't8642', 0.9822485207100592)
('t1088', 't5015', 0.9821428571428571)
('t1768', 't5248', 0.9817073170731707)
('t980', 't2023', 0.9810126582278481)
('t1952', 't3495', 0.9805194805194806)
('t311', 't7998', 0.5705128205128205)
('t120', 't311', 0.5705128205128205)
('t945', 't7998', 0.536144578313253)
('t120', 't945', 0.536144578313253)
('t120', 't576', 0.5313432835820896)
('t576', 't7998', 0.5313432835820896)
('t6601', 't7998', 0.5281899109792285)
('t120', 't6601', 0.5281899109792285)
('t120', 't966', 0.5266272189349113)
('t966', 't7998', 0.5266272189349113)
```

Figure 5: Output of top20 pairs using actual Jaccard scores between all pairs. The tuple of each pair is (bookID, bookID, actual Jaccard score).

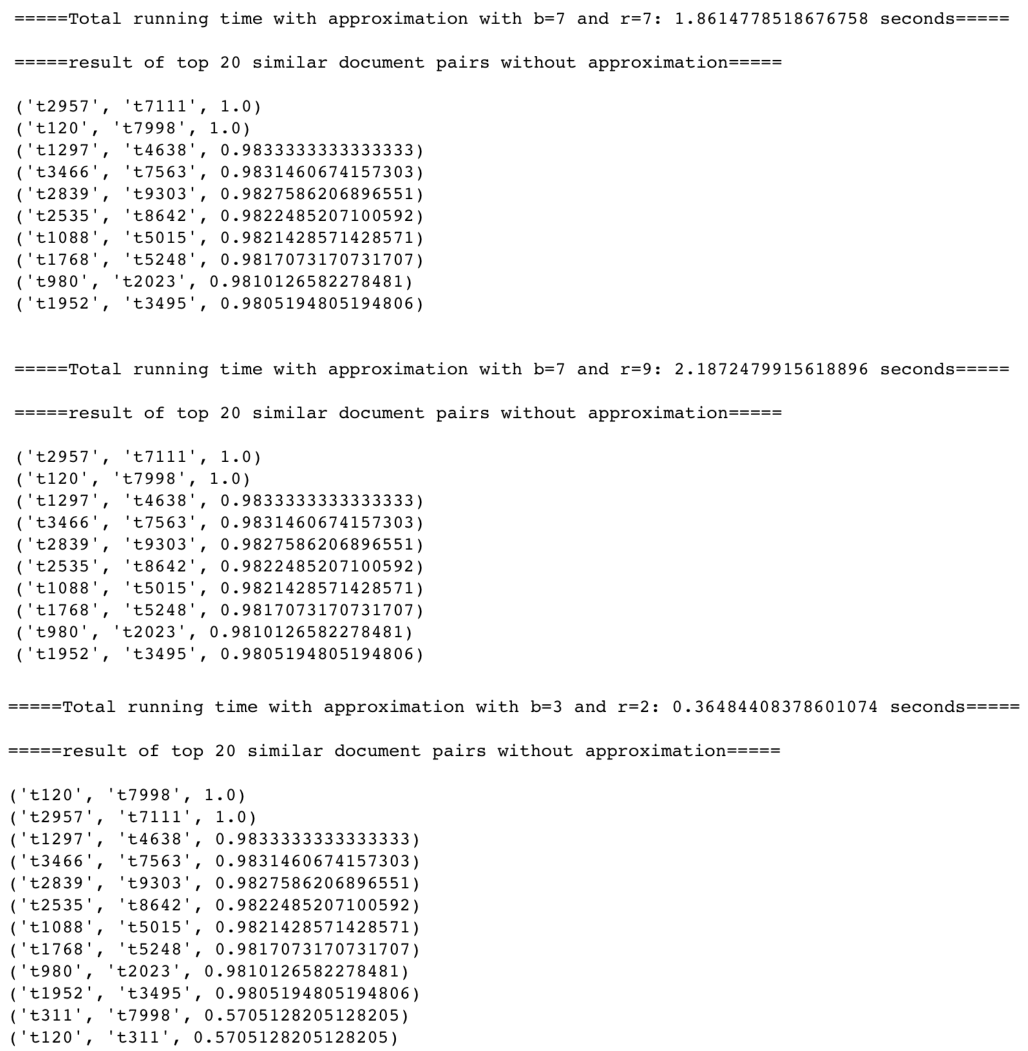


```
=====Total running time with approximation with b=7 and r=1: 0.5453281402587891 seconds=====

=====result of top 20 similar document pairs without approximation=====

('t120', 't7998', 1.0)
('t2957', 't7111', 1.0)
('t1297', 't4638', 0.9833333333333333)
('t3466', 't7563', 0.9831460674157303)
('t2839', 't9303', 0.9827586206896551)
('t2535', 't8642', 0.9822485207100592)
('t1088', 't5015', 0.9821428571428571)
('t1768', 't5248', 0.9817073170731707)
('t980', 't2023', 0.9810126582278481)
('t1952', 't3495', 0.9805194805194806)
('t120', 't311', 0.5705128205128205)
('t311', 't7998', 0.5705128205128205)
('t945', 't7998', 0.536144578313253)
('t120', 't945', 0.536144578313253)
('t120', 't576', 0.5313432835820896)
('t576', 't7998', 0.5313432835820896)
('t120', 't6601', 0.5281899109792285)
('t6601', 't7998', 0.5281899109792285)
('t966', 't7998', 0.5266272189349113)
('t120', 't966', 0.5266272189349113)
```

Figure 6: Same top20 pairs using lsh approximation with $b = 7$ and $r = 1$ as pairs without approximation. The tuple of each pair is (bookID, bookID, actual Jaccard score).



```
=====Total running time with approximation with b=7 and r=7: 1.8614778518676758 seconds=====

=====result of top 20 similar document pairs without approximation=====

('t2957', 't7111', 1.0)
('t120', 't7998', 1.0)
('t1297', 't4638', 0.9833333333333333)
('t3466', 't7563', 0.9831460674157303)
('t2839', 't9303', 0.9827586206896551)
('t2535', 't8642', 0.9822485207100592)
('t1088', 't5015', 0.9821428571428571)
('t1768', 't5248', 0.9817073170731707)
('t980', 't2023', 0.9810126582278481)
('t1952', 't3495', 0.9805194805194806)


=====Total running time with approximation with b=7 and r=9: 2.1872479915618896 seconds=====

=====result of top 20 similar document pairs without approximation=====

('t2957', 't7111', 1.0)
('t120', 't7998', 1.0)
('t1297', 't4638', 0.9833333333333333)
('t3466', 't7563', 0.9831460674157303)
('t2839', 't9303', 0.9827586206896551)
('t2535', 't8642', 0.9822485207100592)
('t1088', 't5015', 0.9821428571428571)
('t1768', 't5248', 0.9817073170731707)
('t980', 't2023', 0.9810126582278481)
('t1952', 't3495', 0.9805194805194806)

=====Total running time with approximation with b=3 and r=2: 0.36484408378601074 seconds=====

=====result of top 20 similar document pairs without approximation=====

('t120', 't7998', 1.0)
('t2957', 't7111', 1.0)
('t1297', 't4638', 0.9833333333333333)
('t3466', 't7563', 0.9831460674157303)
('t2839', 't9303', 0.9827586206896551)
('t2535', 't8642', 0.9822485207100592)
('t1088', 't5015', 0.9821428571428571)
('t1768', 't5248', 0.9817073170731707)
('t980', 't2023', 0.9810126582278481)
('t1952', 't3495', 0.9805194805194806)
('t311', 't7998', 0.5705128205128205)
('t120', 't311', 0.5705128205128205)
('t120', 't576', 0.5313432835820896)
('t576', 't7998', 0.5313432835820896)
('t966', 't7998', 0.5266272189349113)
('t120', 't966', 0.5266272189349113)
('t4490', 't7998', 0.5159420289855072)
('t120', 't4490', 0.5159420289855072)
('t120', 't7058', 0.4972067039106145)
('t7058', 't7998', 0.4972067039106145)
```

Figure 7: Other selected outputs of top20 pairs using lsh approximation. The tuple of each pair is (bookID, bookID, actual Jaccard score).