Homework 2 [45 points] / Updates in magenta

Questions related to the homework can be posted on the Piazza forum site, but do not post answers to homework problems. We will try to respond promptly to posted questions. However, please do not send post questions after 9pm the day before the assignment is due.

You may discuss the homework problems and computing issues with other students in the class. However, you must write up your homework solution on your own. In particular, do not share your code or homework files with other students.

This homework will build upon the ideas presented in the lectures. `R` will be used extensively throughout the homework. We suggest you start early for this homework, especially if you don't have any previous experience with a programming language. If there are any difficulties starting the homework, please contact the TA immediately.

The homework is split into two compulsory parts, and one optional part covering the following:

Part 1: Numerical stability (warm up) (one evening)

Part 2: Investigative simulations (the meat) (one to two evenings)

Part 3: Homework survey - anonymized on CMS (cool down) (one evening)

As always, text in blue denotes what needs to be done. There will be some suggestions in normal font which might require an extra step or two. However, these are optional, and meant for those who want a challenge. We suggest reading through the entire homework at least once before starting on it. Several hints are given in the footnotes as well.

Files to submit on CMS

Do check to see that you have submitted all these files. Each part will be graded as a whole. Instructions on what to submit will be in blue throughout the homework. The writeups should either be TeXed up or done using R Markdown.

1. Part One [15 points]
   - `polyfunction.r`
   - `writeup_one.html` or `writeup_one.pdf`
2. Part Two [30 points]
   - `gen_SB.r`
   - `writeup_two.html` or `writeup_two.pdf`

Part One: Numerical Stability

We have discussed the idea of finite precision in lectures and lab, as well as the limitations of finite precision. For example, in Lab 1 we looked at the line:

```
sqrt(2) * sqrt(2) == 2
```

and in Lab 2, we considered the plot of $f(x) = \frac{1-\cos x}{x^2}$ in the interval $[-\epsilon, \epsilon]$, $\epsilon = 4 \times 10^{-8}$.

In this part of the homework, we will investigate how finite precision plays a part in numerical stability for the evaluation of polynomials. We will consider three different ways to evaluate polynomials (in `R`).

We express a polynomial as:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \tag{0.1}$$

The Direct Method

The direct method evaluates the polynomial explicitly. In other words, we compute the sum $\sum_{i=0}^{n} a_i x^i$.

Horner's Method

This method represents the polynomial in (Equation 0.1) as:

$$f(x) = (\ldots((a_n x + a_{n-1})x + a_{n-2})\ldots a_1)\, x + a_0$$

To give an example, we would then compute:

$$a_3 x^3 + a_2 x^2 + a_1 x + a_0 \qquad \text{as} \qquad ((a_3 x + a_2)x + a_1)x + a_0$$

The Factorization Method

If we knew the roots of the polynomial $f(x)$, then we could compute:

$$f(x) = r_0(x - r_1)(x - r_2)\ldots(x - r_n)$$

The \$641.42[1] question: *Do you think these three methods all give the same numerical result in* `R`*?* We will find out in this part of the homework.

Our first task is to write a function called `polyfunction` which takes in three inputs:

- `x`
- `vec`, a vector either corresponding to the $n+1$ dimensional vector $(a_0, \ldots, a_n)^T$ or $(r_0, \ldots, r_n)^T$, which depends on the
- `type`, a string input which can only take the values `direct`, `horner`, and `factor`.

and computes the value of $f(x)$. Use the following template to create the function:

```
polyfunction<-function(x, vec, type){
  # Put code here
}
```

Submit the file `polyfunction.r` on CMS. Please put a copy of the code in the writeup as well. This function should not assume any fixed degree n of the polynomial. The function can also be written such that it takes in an input of vectors `x`, and has the output be a vector of corresponding $f(x)$, but this is not required (though it will make it easier to do the future parts).

Here are (some) examples of what the function should return, given the following inputs and knowing that:

$$(x-1)(x-2)(x-3) = x^3 - 6x^2 + 11x - 6$$

```
avec = c(-6, 11, -6, 1)  #
# Note r_0 should be 1, i.e. 1*(x-1)*(x-2)*(x-3)
rvec = c(1,1,2,3)

polyfunction(1,avec,"direct")
# [1] 0

polyfunction(2,avec,"horner")
# [1] 0

polyfunction(3,rvec,"factor")
```

[1] adjusted for inflation since 1950

```
# [1] 0

polyfunction(1.211,rvec,"factor")
# [1] 0.2978309

polyfunction(1.211,rvec,"I don't know what to do for this question")
# Error in polyfunction(1.211,rvec, "I don't know what to do for this question"):
# See the TA at once

# Though you can personalize the error message - type ?stop

# If the function is written such that it could take in a vector
# of inputs, then

xvec = c(1,2,3,1.211)
polyfunction(xvec,rvec,"factor")
# [1] 0.0000000 0.0000000 0.0000000 0.2978309
```

With this function, we will consider the following polynomial:

$$(x-1)^9 = x^9 - 9x^8 + 36x^7 - 84x^6 + 126x^5 - 126x^4 + 84x^3 - 36x^2 + 9x - 1$$

and evaluate this polynomial using these three methods over several intervals[2]. It might be a good idea to write a function that does the next section all at once, similar to what we went through in Lab 2.

A Comparison Of These Methods

We will use a *grid* of 101 values for our intervals. Thus, if we consider an interval $[a, b]$, then we would be looking at points:

$$a := i_1 < i_2 < \ldots < i_{100} < i_{101} =: b$$

where each point is equally spaced apart. For *each* of the three intervals:

- $[0.999, 1.001]$
- $[0.99, 1.01]$

[2]Since the majority of the points will be based on the analysis of this polynomial and the function, it would be essential that `polyfunction` is written correctly, and that the representation of `avec` and `rvec` is correct. One way to do so is to directly check the output of `polyfunction(seq(1,9),rvec,"factor")` and `polyfunction(seq(1,9),avec,"horner")`. In this homework, we will give a reminder to check the required functions in the footnotes, and suggest ways to do so. In subsequent homeworks, it'll be up to you.

- $[0.9, 1.1]$

plot the results of these three evaluations on the respective grids. There should be three different plots, each corresponding to a different interval. In each of these plots, the results of the above three evaluation methods should be shown. These plots should be clearly labelled with a legend, and the three evaluation methods clearly distinguishable from each other.

Put these plots in the writeup together with the code needed to generate them.

How do each of these three methods perform on the corresponding intervals, and why do you think this is the case?

Root Finding

One consequence of these differences is in finding the roots of a continuous[3] function, i.e. where $f(x) = 0$ (of course we know the truth in this case). We can do this by looking for successive points in x where $f(x) < 0$ on one side of x and $f(x) > 0$ on the other. For a grid of evaluation points, we can just report the mid-point between two values of x where $f(x)$ must cross zero.

Using the evaluations computed for the interval $(0.99, 1.01)$, write code to find all the roots for $f(x) = (x-1)^9$ evaluated by all three methods. Show both code and the results in the writeup, and give a brief commentary on them.

Here we assume that if we have values $i_a < 0 < i_b$, then $\frac{i_a + i_b}{2} \approx 0$.

Takeaway message: *Be careful in coding root finding functions*[4] *- they may return no roots*[5] *or more than the actual number of roots.*

Check that the writeup has answered everything in blue and that it flows smoothly. Then submit the writeup as `writeup_one.html` or `writeup_one.pdf`. You do not need to copy each question in blue and put it in the writeup before writing your answers, eg you can have appropriately titled sections in the writeup where the questions in blue are answered. If it helps, pretend you are making an investigative report and presenting your findings with relevant code / diagrams.

[3] Look at the Intermediate Value Theorem to see why this holds.
[4] We might do this in labs or future homeworks when we cover optimization techniques.
[5] Give an example where a root finding function might return no roots in an interval, but a root exists.

Part Two: Investigative Simulations

Bad things come in threes - A Student

This part of the homework will reinforce the concepts of *matrix algebra*, understanding the difference between *computational statistics* and *theoretical statistics*, *dense* and *sparse* matrices, as well as introduce several new `R` commands. In particular, we hope that the skills learnt in this homework will be useful.

Whether you end up doing research, or get employed in a vocation which is statistics / CS related, knowing how to run simulations for verification purposes is an important skill to have. Here are two common scenarios where simulations are run:

- Verifying that empirical results matches theory
- Comparing algorithms against each other (and picking the better one)

and usually, our benchmarks are *speed* and *accuracy*. In this part of the homework, we will look at *random projections* (in terms of both speed and accuracy).

Random Projections

Random projections are commonly used when dealing with the notion of *distance* in large datasets, as they preserve distance under expectation. We will illustrate what *preserving distance under expectation* means in this homework. As an aside, here are some applications of random projections (not exhaustive):

- Calculating similarity / dissimilarity between pairs of documents
- Clustering (distance-based) algorithms [Das00]
- Matrix factorizations [HMT11]
- Compressing principal components for extremely large matrices [AH14], [CJ15]
- in SVMs (inner product kernel) [PBMD12]
- Hypothesis testing in cases where $n \approx p$ or $p > n$ [SLR14]

and there is ongoing research on random projections, primarily on the structure of random projections, and algorithms using random projections[6].

[6]Do speak to the TA about research next fall if the homework makes you feel excited about random projections.

Three Notions Of Distance

Suppose we are given two p dimensional vectors, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, i.e. $\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$, $\mathbf{y} = (y_1, y_2, \ldots, y_p)^T$ and we are interested in the notion of distance. We will consider three simple notions of distance. First, we could compute the *length* (norm) of each vector, which is given by:

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{p} x_i^2 \right)^{\frac{1}{2}} \tag{0.2}$$

Secondly, the *Euclidean distance* between them:

$$\|\mathbf{x} - \mathbf{y}\|_2 = \left(\sum_{i=1}^{p} (x_i - y_i)^2 \right)^{\frac{1}{2}} \tag{0.3}$$

Thirdly, their *inner product* (also called the dot product):

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{p} x_i y_i \tag{0.4}$$

Where does the expectation come in? We will introduce a random vector $\mathbf{r} \in \mathbb{R}^p$, i.e. $\mathbf{r} = (r_1, r_2, \ldots, r_p)^T$, where each r_i are i.i.d. from some distribution with mean μ and variance σ^2. Let v_1 be the inner product $\langle \mathbf{x}, \mathbf{r} \rangle$, and let v_2 be the inner product $\langle \mathbf{y}, \mathbf{r} \rangle$.

We will do some warm up calculations here, which will be included in the writeup.

First, write down an expression for v_1^2 and v_2^2.

Secondly, suppose that for r_i, we have $\mu = 0$, and $\sigma^2 = 1$. Thus, this implies that $\mathbb{E}[r_i] = 0$ and $\mathbb{E}[r_i^2] = 1$. Therefore, find an expression for the expectations $\mathbb{E}[v_1^2]$ and $\mathbb{E}[v_2^2]$.

Similarly, write down an expression for $\mathbb{E}[(v_1 - v_2)^2]$ and $\mathbb{E}[v_1 v_2]$.

Compare these expressions with the three notions of distance mentioned above. What do you notice?

This suggests a way to get an estimate of any of the three above notions of distance. We could repeatedly generate random vectors $\mathbf{r} \in \mathbb{R}^p$, and continue evaluating the respective expressions and taking the overall mean. For example (be careful of the difference in notation here - we are now looking at vectors instead of individual terms in these vectors), if we wanted an estimate of the norm of $\mathbf{x}$, we could generate k i.i.d. random vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k$, and

evaluate $v_1 = \langle \mathbf{x}, \mathbf{r}_1 \rangle, v_2 = \langle \mathbf{x}, \mathbf{r}_2 \rangle, \ldots, v_k = \langle \mathbf{x}, \mathbf{r}_k \rangle$. By squaring these v_is and taking their mean, we have an estimate of $\|\mathbf{x}\|_2^2$ (by the law of large numbers).

In general, what usually happens is that this matrix equation is used:

$$V = \frac{1}{\sqrt{k}} XR \tag{0.5}$$

and V is computed. Here, X is a $n \times p$ matrix, and R is a $p \times k$ matrix. We can think of each row of X corresponding to our initial observations (eg in the previous part, we had $\mathbf{x} \in \mathbb{R}^p$, and $\mathbf{y} \in \mathbb{R}^p$), and each column of R corresponding to the generation of $\mathbf{r}_1, \ldots, \mathbf{r}_k$.

Given any two rows $\mathbf{x}_i$, $\mathbf{x}_j$ in X, explain how V could be used in Equation (0.5) to estimate the norm of $\mathbf{x}_i, \mathbf{x}_j$, the Euclidean distance between $\mathbf{x}_i, \mathbf{x}_j$, and the inner product between $\mathbf{x}_i, \mathbf{x}_j$. The explanation should just be 1-2 sentences, with reference to the calculations made in the earlier part.

Hint: If this level of matrix algebra is uncomfortable, try setting $n = 2$, and $k = 4$, and do some matrix multiplication, before extending this to the general case.

We will now implement *random projections* in `R`.

Three Probability Distributions

We first introduce three probability distributions, each with mean 0, and second moment 1. They are:

- Our favourite distribution, the *Normal* distribution
- The *Rademacher* distribution, which takes on two values ± 1 with equal probability
- The *Sparse Bernoulli* distribution with parameter s, $r \sim SB(s)$ is such that:

$$r = \begin{cases} \sqrt{s} & \text{with probability } \frac{1}{2s} \\ -\sqrt{s} & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \end{cases}$$

  Note that the *Sparse Bernoulli* distribution with parameter 1 is just the *Rademacher* distribution.

Show that the Sparse Bernoulli distribution has mean 0 and second moment 1 for all $s \geq 1$.

The first (major) task of this part: Write a function `gen_SB.r`, which takes in 3 inputs: p, k, s, and gives a $p \times k$ matrix R as an output, where each r_{ij} entry in R is generated from $SB(s)$.

```
gen_SB<-function(p,k,s){
  ## Put code here
  ## You are not allowed to use packages to generate this automatically
  ## but can use standard r--- commands (eg rnorm, runif, rbinom etc..)
  return(R)
}
```

Submit the file `gen_SB.r` on CMS. Put a copy of this code in the writeup as well.

As we will be using this function (to generate R) and run simulations, do check that the function works[7].

The rest of this question will be (mostly) devoted to running simulations. The general loop structure to run these simulations has been given in Lab 3. Feel free to modify this loop structure for the homework.

Simulation Process

We will set $n = 1000$, and consider *three* different Xs, with $p \in \{100, 1000, 10000\}$. Each X has values generated from the Normal distribution. Run the following code to generate these Xs.

```
n = 1000
set.seed(1)
X1 = matrix(rnorm(n*100), nrow = n)
X2 = matrix(rnorm(n*1000), nrow = n)
X3 = matrix(rnorm(n*10000), nrow = n)
```

The output of X_1, X_2 and X_3 **should not be displayed** at all. One point will be deducted if you submit pages and pages of numbers. This may seem like common sense, but in HW1, I had people displaying the matrix of the dataset Olympic[8]. In general, only submit output if the question asks for it, and always submit plots (unless otherwise stated).

We consider these matrices X_1, X_2, X_3, and *three* types of random projection matrices,

$$R_{\text{normal}} \qquad R_{\text{rademacher}} \qquad R_{\text{SB(20)}}$$

where in these matrices:

[7] Here's two easy ways to check this. One, check that the mean and standard deviation of these entries tend to what is expected. Two, generate this with parameter $s = 1$. Does the output seem reasonable?

[8] and in the grad level version of this course, I had a few grad students giving me 10 pages...of a large identity matrix. Don't..just..don't.

- the entries in R_{normal} are drawn from $N(0, 1)$
- the entries in $R_{\text{rademacher}}$ are drawn from $\{\pm 1\}$ with equal probability
- the entries in $R_{\text{SB(20)}}$ are drawn from $SB(20)$

and we will consider $V = \frac{1}{\sqrt{k}} XR$, where we use different Xs and Rs for varying k. Note that there will be **nine** possible combinations - in short:

$$\begin{array}{lll} V_1 = \frac{1}{\sqrt{k}} X_1 R_{\text{normal}} & V_2 = \frac{1}{\sqrt{k}} X_1 R_{\text{rademacher}} & V_3 = \frac{1}{\sqrt{k}} X_1 R_{\text{SB(20)}} \\ V_4 = \frac{1}{\sqrt{k}} X_2 R_{\text{normal}} & V_5 = \frac{1}{\sqrt{k}} X_2 R_{\text{rademacher}} & V_6 = \frac{1}{\sqrt{k}} X_2 R_{\text{SB(20)}} \\ V_7 = \frac{1}{\sqrt{k}} X_3 R_{\text{normal}} & V_8 = \frac{1}{\sqrt{k}} X_3 R_{\text{rademacher}} & V_9 = \frac{1}{\sqrt{k}} X_3 R_{\text{SB(20)}} \end{array}$$

For each possible pair of X, R, we will be investigating both the time taken and accuracies of the estimates for the Euclidean distance and inner product[9].

Computation versus Theory

Suppose we generated a random R from the distribution $SB(s)$, for a fixed S. Then note that we can *factor* out the $\sqrt{s}$ term. In particular, we can write

$$R = \sqrt{s}\widetilde{R}$$

where $\widetilde{R}$ is a matrix of 1s, -1s, and 0s.

Furthermore, with our random projection equation Equation 0.5, we would calculate $V = \frac{1}{\sqrt{k}} XR$, and then compute $A := VV^T$ where each a_{ij} entry is an estimate of the inner product between any two rows $\mathbf{x}_i, \mathbf{x}_j$ in X.

Note that:

$$\begin{aligned} VV^T &= (\frac{1}{\sqrt{k}} XR)(\frac{1}{\sqrt{k}} XR)^T \\ &= \frac{1}{k} XRR^T X^T \\ &= \frac{1}{k} X(\sqrt{s}\widetilde{R})(\sqrt{s}\widetilde{R}^T) X^T \\ &= \frac{s}{k} X\widetilde{R}\widetilde{R}^T X^T \end{aligned}$$

[9] Writing a helper function to generate these random matrices may be helpful - think of how we constructed `polyfunction` in Part One.

and hence:

$$(\frac{1}{\sqrt{k}}XR)(\frac{1}{\sqrt{k}}XR)^T = \frac{s}{k}X\widetilde{R}\widetilde{R}^TX^T$$

Therefore, supposing we had the value of R, $\widetilde{R}$, and X stored in `R`, we could either run:

```
## First line
VtV_one = ((1/sqrt(k)) * X %*% R) %*% t(((1/sqrt(k)) * X %*% R))
## Second line
VtV_two = s/k (X %*% Rtilde) %*% t(X %*% Rtilde)
```

Is there a reason why we might prefer to run one line of code compared to the other, even though $(VV^T)_1 = (VV^T)_2$? Justify your answer.

The function `gen_SB` can be modified to generate $\widetilde{R}$. If this is done, the updated version can be submitted on CMS.

Speed of random projections

Suppose we are interested in the similarity of all possible pairs of vectors in our dataset X. One possible way to get a notion of similarity is to compute the inner product (dot product) of every possible pair of vectors.

This can be done by computing $B := XX^T$. Then each b_{ij} is the inner product of $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$.

Find the time taken to compute XX^T for each X_1, X_2, X_3.

Let's use the `R` command `proc.time()` to do so. Here is example output:

```
proc.time()
#    user   system  elapsed
# 883.814   19.604 2394.123   ## so I started using R about 40 mins ago
```

In particular, we will focus on the third value (`elapsed`), which is the time from when `R` was first opened until `proc.time()` was typed. It is common to do something like this when evaluating the time taken to run a function:

```
tic = proc.time() # familiar if you've used Matlab
```

```
## Some function / code (where the time needs) to be measured

toc = proc.time() - tic

toc[3]  # 3rd element

# elapsed
#  0.008
# units should be in seconds
```

Additionally, since we might expect matrix multiplication to be faster if a matrix is mostly filled with zeroes, then we should store $R_{\text{SB}(20)}$ as a sparse matrix. To convert matrices to a sparse format in `R`, we need to load the package `Matrix`, and then type:

```
R = Matrix(R)  ## Note the caps
```

where R is a matrix.

How fast are random projections in computing an estimate of all pairs of inner products? We will find out in this section. Thus, the current task is to do the following.

- For k ranging from 10, 20, 30, 40, 50, 100, 250, 500, 1000, find the total time taken to compute $VV^T = \left(\frac{1}{\sqrt{k}}XR\right)\left(\frac{1}{\sqrt{k}}XR\right)^T$ with each of the nine possible combinations of $V = \frac{1}{\sqrt{k}}XR$. Ensure that $R_{\text{SB}(20)}$ is in sparse format when matrix multiplication is done.
- Repeat this 10 times, and take the average time[10].
- For each X_i, plot the time taken against the number of columns k for each R. Thus, the plot should have three lines, each denoting the time taken using a different R. Add a fourth line denoting the time taken to compute $X_iX_i^T$. Ensure that the plots are well labelled, together with a legend. Alternatively , plot log(time) versus k if it makes the plots look more informative.
- Display the plots in the writeup (with the code to generate them) and give a brief commentary on what the plots show.

Suggestion: Write a function to accomplish the above all at once. However, try it out with a few values of k to ensure the function works. The above simulations should take at most 10-15 minutes to complete (if code is not optimized), otherwise faster.

[10]In practice, we would usually repeat this for 500-1000 iterations instead of 10, to be sure of our results. If the homework is started early, feel free to run this for more iterations to get a smoother plot.

Note: Here we are assuming that finding the Euclidean distance for every pair of vectors would take the same amount of time as calculating the inner product[11].

Accuracy of random projections

How accurate are random projections? We'll find out, by looking at the errors of the estimated Euclidean distance of just the first two rows of X. Thus, we will only consider the first two rows of X_1, X_2, and X_3 now.

Find the actual value of the Euclidean distance between the first two rows of X_1, X_2 and X_3.

Recall that the MSE of an estimator $\overline{X}$ is given by:

$$MSE(\overline{X}) = \mathbb{E}[(\overline{X} - \mu)^2]$$

Then, the current task is to:

- For k ranging from 10, 20, 30, 40, 50, 100, 250, 500, 600, 700, 800, 900, 1000, we will compute an estimate of the Euclidean distance between X_1 and X_2, using $V = \frac{1}{\sqrt{k}} X[1:2,]R$ with each of the nine possible combinations of $V = \frac{1}{\sqrt{k}} X[1:2,]R$. We could also compute this with k in intervals of 2, i.e. {2,4,6,..., 1000} to get a more accurate sense of the accuracy. While it does seem it would take a long time (with a naïve loop, there's a much faster way to do this by generating the columns k needed all at once, and taking a subset of them each time. Feel free to implement this instead if you are comfortable with this technique.
- Repeat this 100 times, and compute the MSE for the Euclidean distance for each combination. For example, if we denoted $ED_1, \ldots, ED_{100}$ to be the estimated Euclidean distances for the 100 repetitions, then our MSE would be:

$$\frac{\sum_{i=1}^{100} (ED_i - \text{Actual ED})^2}{100}$$

- Compute the relative MSE given by $\frac{\text{MSE}}{\text{Actual ED}}$.
- For each X_i, make a plot of the relative MSE of the Euclidean distance with the 3 different types of matrices R. Each of these plots should have 3 lines, each denoting the relative MSE using a different R. Alternatively, plot log(relative MSE) versus k if it makes the plots look more informative. Ensure that the plots are well labelled, together with a legend.

[11] but feel free to calculate the time taken to compute Euclidean distance for each vector as well and have 6 plots instead of 3 (or 3 plots with clearly labelled lines denoting time taken for Euclidean distance and inner products).

- Display the plots in the writeup (with the code to generate them) and give a brief commentary on what the plots show.
- For these plots, which distribution (of r_{ij} in R) has the lowest MSE for Euclidean distance? Do you think this depends on the size of the matrix? Give a brief explanation.
- Instead of a brief explanation, give an expression for the mean square error, and / or show a bound for this. Hint: Consider fourth moments and the (Markov inequality + Taylor expansion). This is for the mathematicians and the probability theorists who prefer a challenge. For example, if we were just looking at norms, then we want to find f_1, f_2 for:

  $$\mathbb{P}\left[\|\mathbf{v}\|^2 < (1-\epsilon)\|\mathbf{x}\|^2\right] \leq f_1$$

  and

  $$\mathbb{P}\left[\|\mathbf{v}\|^2 > (1+\epsilon)\|\mathbf{x}\|^2\right] \leq f_2$$

  in order to write:

  $$\mathbb{P}[(1-\epsilon)\|\mathbf{x}\|^2 \leq \|\mathbf{v}\|^2 \leq (1+\epsilon)\|\mathbf{x}\|^2] \leq f_1 + f_2$$

There is ongoing research about the "best structure" / "best distribution" for random matrices R. However, this structure also depends on the data matrix X. In our homework, we simulated X from a Normal distribution. Could some probability distributions work better on dense matrices (or sparse matrices, or if X came from another distribution)? Furthermore, what if the entries of R are not i.i.d.? These are some things to think about. Feel free to play about with generating different Rs, and submit a paper to a conference if you discover something new (optional).

Using Random Projections For Very Large Datasets

By now, we should have three plots of the time taken, and three plots of the relative MSE of Euclidean distances. We have also simulated X_1, X_2, X_3, but with $p = 100, 1000, 10000$.

Either answer (for the statisticians / theorists):

By only making reference to these six plots, and perhaps the bounds if they have been proven, describe a heuristic to follow if we were given $X_{n\times p}$, $p \gg 10000$, and had to find the pairwise Euclidean distances between all pairs of observations (observation: a row $\mathbf{x}_{i\cdot}$) in X, keeping in mind a balance between speed and accuracy. Feel free to run some extra simulations to check if the heuristic makes sense.

OR answer (for the machine learning oriented students):

In a CS / machine learning context, suppose we have stored the text of **all** the webpages in the world as vectors $\mathbf{x} = (x_1, x_2, \ldots, x_p)^T \in \mathbb{R}^p$ where each x_k, $1 \leq k \leq p$ takes on the values $\{0, 1\}$. In particular, if word i appears at least once in the webpage, then $x_k = 1$, else $x_k = 0$.

For example, suppose there are only three words in the world. { `cat`, `dog`, `boy` } . If the webpage had the text "cat cat dog dog dog", then the vector $\mathbf{x}$ representing the webpage would be (1 1 0), assuming that word 1 was `cat`, word 2 was `dog`, and word 3 was `boy` because only `cat` and `dog` appeared in this webpage.

One could imagine that n, the number of total webpages would be large. One could also imagine that p, the total number of unique words would be large as well (eg, the English language has over a million words) - but what about other European languages using the Roman alphabet? Or script like languages such as Tamil or Mandarin? That's not even assuming tpyos.

For each pair of webpages i, j, we want to find how similar how each of these webpages are, and one measure of distance we'd like to use is the *resemblance*, given by the fraction $\frac{\text{Number of common words in both webpages}}{\text{Number of unique words in both webpages}}$ (it helps to think of this as "intersection over union"). Being a smart Cornell student, you make the following observations:

1. The number of common words in both webpages is just the inner product of the vectors $\mathbf{x}_i, \mathbf{x}_j$.
2. The number of unique words in both webpages is basically $\|\mathbf{x}_i\|_2^2 + \|\mathbf{x}_j\|_2^2 - \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.
3. I've learnt about this technique called random projections which *might* help me find the *resemblance*, because I can estimate norms and inner products using random projections. I do not need the exact value since there's really not much difference between "{Website A and website B are 70% similar}, and {Website A and website B are 72.12452% similar} - at least we know they have a high degree of similarity and can do something like "<search engine name> has found other websites extremely similar to this, and omitted them from this search"[12].

By only making reference to these six plots, and perhaps the bounds if they have been proven, describe a heuristic to follow to find each pairwise *resemblance* between every pair of

[12]Describe how plagiarism detectors such as `Turnitin` might work. This is a semi-serious question, but consider the fact that we can't just check for presence of words, because it is possible to have two very dissimilar documents repeating most of the same words, but in different order and frequency. Or what if a student created "random" garbagey words, but white-texted them and submitted to `Turnitin` so the TA / prof won't notice them? What else could you do (but within this Euclidean distance / inner product / norm) context?

webpages, keeping in mind a balance between speed and accuracy[13]. Feel free to run some extra simulations to check if the heuristic makes sense. Hint: If you run extra simulations, remember that the vectors $\mathbf{x}_i$ will not be drawn from a normal distribution. In fact, they're going to be sparse with mostly 0s rather than 1s. If enough people are interested in this (make a Piazza post), I can post real data on CMS for the simulations.

Check that the writeup has answered everything in blue and that it flows smoothly. Then submit the writeup as `writeup_two.html` or `writeup_two.pdf`. You do not need to copy each question in blue and put it in the writeup before writing your answers, eg you can have appropriately titled sections in the writeup where the questions in blue are answered. If it helps, pretend you are making an investigative report and presenting your findings with relevant code / diagrams.

[13]This is in essence the same question, but with a bit more context behind it. There are also better techniques (also statistical) which can find the *resemblance* instead of using random projections (they're good for Euclidean distances, but don't perform so well for inner products), but it does serve as some motivation.

Part Three: Feedback On Homework Assignments

You can answer any subset of the following questions under Homework Survey 1 on CMS. This will help us design / create future homework problems. The survey will be anonymized. The questions are on CMS, but here is a copy of these questions. Please only submit the survey after homework 2 is completed, because the survey questions will reference homework 2.

1. **Time Taken / Length**
   (a) How long did you spend on this assignment?
   (b) The length of this assignment was `too short / just right / too long` for a 4 credit course.
   (c) The homework is reasonable to do within a span of two weeks `Yes / No`
2. **Learning**
   (a) The first two assignments `did not reinforce / reinforce` the skills / concepts taught in lecture
   (b) The first two assignments `did not reinforce / reinforce` the skills / concepts covered in lab
   (c) Do you feel you have learnt anything from the first two assignments? `Yes / No / Unsure`
   (d) Do you feel you can apply the concepts from the first two assignments out of this class? `Yes / No / Unsure`
3. **Relevance and Difficulty Level**
   (a) What I have learnt in lecture was `not relevant / relevant` to the first two assignments.
   (b) What I covered in labs was `not relevant / relevant` to the first two assignments.
   (c) What I have learnt in lecture `prepared me / did not prepare me` for the first two assignments.
   (d) What I covered in labs `prepared me / did not prepare me` for the first two assignments.
   (e) This assignment was `too easy / just nice / too hard`.
4. **Style** *The style of the first two assignments are different from previous semesters (more write up based rather than short answer questions). We have also tried to write as much detail as possible to prevent any ambiguity and to provide some motivation, leading to slightly (?) longer than average assignments.*

(a) The current assignments are `not detailed / just nice / too verbose`

(b) There is `no ambiguity / some ambiguity / lots of ambiguity` in the homework problems.

(c) On a scale of having assignments to be writeup / investigative based to short answer question based, where on the scale would you prefer?

(d) Are there any other types of homework problems you would prefer?

5. **Academic Support** *We provide office hours and help on Piazza for assignments. However, we try to walk a fine line on Piazza in providing help. In general, we try not to reply to questions which can be easily solved after a minute or two of thought / easily googled, but we may reply if the assignment due date draws near.*

(a) Academic help / support in labs is `non-existent` / `so-so` / `good`

(b) Academic help / support during office hours is `non-existent` / `so-so` / `good`

(c) Academic help / support on Piazza is `non-existent` / `so-so` / `good`

(d) I prefer to get help `during office hours / on Piazza / both`

(e) I only use Piazza to ask questions about the homework `Yes / No`

(f) I reply on Piazza to classmates' queries about the homework `Yes / No`

(g) I rather wait for someone to post a question I want answered on Piazza instead of posting for myself `Yes / No`

(h) I would use Piazza to discuss interesting concepts covered in the homework `Yes / No`

6. **Misc**

(a) What do you feel was the best part about this homework / last homework?

(b) What do you feel was the worst part about this homework / last homework?

(c) Any other feedback?

References

[AH14] Farhad P. Anaraki and Shannon Hughes. Memory and Computation Efficient PCA via Very Sparse Random Projections. In Tony Jebara and Eric P. Xing, editors, Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 1341–1349. JMLR Workshop and Conference Proceedings, 2014.

[CJ15] L.-H. Chen and C.-R. Jiang. Multi-dimensional Functional Principal Component Analysis. ArXiv e-prints, October 2015.

[Das00] Sanjoy Dasgupta. Experiments with Random Projection. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI '00, pages 143–151, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev., 53(2):217–288, May 2011.

[PBMD12] Saurabh Paul, Christos Boutsidis, Malik Magdon-Ismail, and Petros Drineas. Random Projections for Support Vector Machines. CoRR, abs/1211.6085, 2012.

[SLR14] Radhendushka Srivastava, Ping Li, and David Ruppert. Raptt: An exact two-sample test in high dimensions using random projections. arXiv preprint arXiv:1405.1792, 2014.