EE-556: MATHEMATICS OF DATA: FROM THEORY TO COMPUTATION
LABORATORY FOR INFORMATION AND INFERENCE SYSTEMS
FALL 2015

INSTRUCTOR:
PROF. VOLKAN CEVHER

LAB EXERCISE-1 (FOR LECTURES 1,2 AND 3)

This lab is intended as a review of linear algebra, probability, convexity and complexity. It covers lectures 1-3 in the course.

1 Guidelines

- This lab should be completed in three lab sessions.
- In case you can not complete it, there will be some make-up possibilities but with a 50% penalty on the grade (please refer to the syllabus for details).
- This lab consist of four parts, each one covering a topic. The estimated time each section should take is indenticated.
- **Please read through the exercises prior to attending the lab session**.
- For this lab, you will need extra space to write down your answers, so please bring along some paper with you.
- After completing each problem, call one of the teaching assistants to verify your answers.

2 A Matlab Tour of Linear Algebra *(3 hours)*

PROBLEM 1: MATRIX NORMS

Matrix norms are numerous (e.g., nuclear, spectral, Frobenius, etc.), and in this course we will make extensive use of them, so it is important to be at ease with them from the very beginning. You can start by typing `help norm` in MATLAB.

In this problem, we will need some random matrices of different dimensions. For this purpose, we will use the following four built-in functions of MATLAB:

`rand`	to generate uniformly distributed pseudorandom numbers;
`randi`	to generate pseudorandom integers from a uniform discrete distribution;
`randn`	to generate normally distributed pseudorandom numbers;
`randperm`	to generate a random permutation.

For more information on these functions and to learn how to use them, use the `help` command of MATLAB (for example, type `help randn` in the command window). We use `tic toc` commands to measure the elapsed time during the experiments. If you are not familiar with the use of `tic toc`, type `help tic` in the command window.

Some of the matrix norms are computationally cheap to evaluate. For instance, the Frobenius norm of an $n \times p$ matrix A is defined as the square root of the sum of the absolute squares of its entries:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{p} |a_{i,j}|^2},$$

and we can compute it with a single pass over the entries of the matrix.

On the other hand, not all matrix norms are easy to compute. In fact, evaluating norms may easily become a computational burden of your implementations, especially when the dataset is large. The nuclear norm and the spectral norm are the most famous examples of this. As you might remember from the class, the nuclear norm is defined as the sum of the singular values, and the spectral norm is the maximum singular value of the matrix.

The first part of this exercise may seem a bit strange to you at first glance. We do not ask you to implement anything, because our teaching assistants have already done it for you. Yet, since they are lazy, they did not write down the comments in the code. You will read the code, use it, and you write the comments that explain the code where we left it blank for you to complete. You need to show that you understand the story behind the experiments, and to convince us that you recognise the take-home message of this problem.

You are going to plot some figures in parts a, b and f. **Either save these figures or leave them open**, since you are supposed to show them to the teaching assistant once you finish the problem. **Do not skip the warm-up problems**, as you will not get any points from the whole problem unless you show your results for these parts. These parts are labelled as *warm-up* so that you can freely ask for help from the teaching assistants, in case you have troubles.

(a) (0 point, warm-up) Read the code in `pb1a.m` (copied below) and complete the comments where question marks are shown. Explain the experiment in one sentence and run the code to observe the plot.

```
dim     = [];
avtime  = [];
for scale = 1:20
    n               = scale*50;
    p               = n;
    dim(end+1,1)    = n*p;

    tottime = 0;
    for trial = 1:20
        A = randn(n,p);                     % ???
        tic;
        norm(A, 'fro');                     % ???
        tottime = tottime + toc;
    end

    avtime(end+1,1) = tottime / trial;      % ???
end

figure
plot(dim, avtime)
title('computational time vs problem dimensions - Frobenius Norm')
xlabel('problem dimensions')
ylabel('average time(s)')
```

Do you have a Netflix account? Have you ever wondered how Netflix recommends you movies for your personal taste? We will discuss the matrix completion problem and recommender systems in the following weeks of this course, but now we would like to use it to motivate how large the datasets can be that we need to work with.

Netflix held an open competition for the best recommender system algorithm to predict user ratings for films between 2006 and 2009, and they provided a training dataset of size 480189 users and 17770 movies [1].

Make an educated guess: How long would it take to compute the Frobenius norm of the Netflix dataset?

(b) (0 point, warm-up) Now let us try a similar experiment with the spectral norm. For this, you need to complete the blanks in the code of `pb1b.m`. If you need a hint, type `help norm` in the command window.

You should observe a different behaviour from the Frobenius norm. Clearly, the computation of the spectral norm for the Netflix dataset would be impractical. As a take-home message, you should observe and remember the following:

$$As\ the\ data\ size\ increases,\ the\ amount\ of\ time\ needed\ to\ compute\ the\ spectral\ norm\ of\ the\ matrix\ increases\ drastically!$$

However, this is not the end of the story.

(c) (0 point, warm-up) The computation of the spectral norm is unavoidable in many real world problems. This brings us to the discussion of the **exact computation vs. approximation**.

Recall the definition of the spectral norm:

$$\|\mathbf{A}\| = \|\mathbf{A}\|_{2\to 2} = \sup_{\|\mathbf{x}\|_2 \le 1} \|\mathbf{A}\mathbf{x}\|_2.$$

A simple idea to approximate the spectral norm is the following: Generate many different random unit vectors $\mathbf{x}$. Apply the matrix $\mathbf{A}$ to these random vectors and find the maximum of the 2-norm of the resulting vectors.

The code in the file `pb1c.m` simulates this method. Complete the code and run it for different sizes of datasets: $n = 2, 3, 4, 5$ and write the number of trials needed to find a good approximation for each case. Do you think that this method can be useful in practice?

(d) (5 points) Now, let us assume that the matrix $\mathbf{A}$ is symmetric. Let $\{\lambda_j, \mathbf{v}_j\}_{j=1}^n$ denote the eigenpairs of $\mathbf{A}$. Since $\mathbf{A}$ is symmetric, the singular vectors and the eigenvectors of $\mathbf{A}$ are the same, and we may assume that the eigenvectors v_j form an orthonormal basis of $\mathbb{R}^n$. Assume that λ_1 is the eigenvalue with largest magnitude, i.e., $|\lambda_1| > |\lambda_j|$ for $2 \leq j \leq n$. Then, by definition, λ_1 is the spectral norm of the matrix $\mathbf{A}$ and $\mathbf{v}_1$ is the vector $\mathbf{x}$ that maximizes $\|\mathbf{A}\mathbf{x}\|_2$.

Our initial estimate, $\mathbf{x}_1$ can be expressed in terms of this basis, i.e.,

$$\mathbf{x}_1 = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j$$

for some coefficients α_j, where we assume that $\alpha_1 \neq 0$.

Assume that $\lambda_1 = 1$, without loss of generality, and show that $\mathbf{A}^k\mathbf{x}_1$ converges to $\alpha_1\mathbf{v}_1$ as k goes to infinity. In other words, in the limit of $k \to \infty$, $\mathbf{A}^k\mathbf{x}_1$ differs from $\mathbf{v}_1$ only by a constant factor.
Hint: Recall the definition of eigenvalue and eigenvector: $\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_j$. Note that, $|\lambda_j| < |\lambda_1| = 1$ for $2 \leq j \leq n$.

(e) (10 points) This observation leads to the following result: We can use the unit vector $\mathbf{x}_k$ in the direction of $\mathbf{A}^k\mathbf{x}_1$ to approximate λ_1 since:

$$\lim_{k\to\infty} \|\mathbf{A}\mathbf{x}_k\|_2 = \lim_{k\to\infty} \frac{\|\mathbf{A}\mathbf{A}^k\mathbf{x}_1\|_2}{\|\mathbf{A}^k\mathbf{x}_1\|_2} = \|\mathbf{A}\mathbf{v}_1\|_2 = |\lambda_1|\|\mathbf{v}_1\|_2 = |\lambda_1|.$$

The code in `pb1e.m` simulates the following scheme: In the first iteration, we generate a random unit vector $\mathbf{x}_1$. If this estimate is not accurate, we take the unit vector $\mathbf{x}_2$ in the direction of $\mathbf{A}\mathbf{x}_1$ as the new estimate. In general, if the kth estimate $\mathbf{x}_k$ is not accurate, we take the unit vector in the direction of $\mathbf{A}\mathbf{x}_k$ as the new estimate $\mathbf{x}_{k+1}$. However, the code has some missing lines. Complete the code and run it for different sizes of datasets: $n = 2, 3, 5, 100$ and 1000. Write down the number of iterations needed to find a good approximation for each case.

This method is called as the **power method** [2] and we will analyze it rigorously in the laboratory session of the 4th week.

(f) (5 points) The power method can be applied for non-symmetric matrices, and even for rectangular matrices. Note that $\mathbf{A}^T\mathbf{A}$ is a symmetric matrix, even if $\mathbf{A}$ is not. Furthermore, the largest singular value of $\mathbf{A}$ is the square root of the largest eigenvalue of $\mathbf{A}^T\mathbf{A}$.

Show that $\mathbf{A}^T\mathbf{A}$ is a symmetric matrix. Then, complete the code in `pd1f.m`, which simulates the same scheme as in the previous part, except that we take the unit vector in the direction of $\mathbf{A}^T\mathbf{A}\mathbf{x}_k$ instead of $\mathbf{A}\mathbf{x}_k$ as the new estimate $\mathbf{x}_{k+1}$.

Run the code for different sizes of datasets: $n = 2, 3, 5, 100$ and 1000. Write down the number of iterations needed to find a good approximation for each case. Note that the random matrix $\mathbf{A}$ that we generate in the code is rectangular for this problem.

(g) (10 points) The power method is a useful tool to approximate the spectral norm of matrices (soon, you will learn that it is not limited to that), and in fact, MATLAB has a built-in function that uses the power method to estimate spectral norm. Type `help normest` in the command window to learn about it.

Complete the missing lines in `pb1g.m`, run it, and observe the plot. Compare the behaviour of the exact computation of the spectral norm and the approximation by power method for different data sizes.

Problem 2: Some Basics of Linear Algebra

In this exercise we will create a matrix and do a general review of Linear Algebra over that matrix. To begin with, recall that a diagonalizable matrix $\boldsymbol{A}$ is a matrix that admits an eigenvalue decomposition,

$$\boldsymbol{A} = \boldsymbol{V}\Sigma\boldsymbol{V}^{-1}$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix that contains the eigenvalues on its diagonal, and $\boldsymbol{V}$ contains the eigenvectors in its columns. Remember also that for this decomposition $\boldsymbol{A}$ must be a square matrix, which is not a requirement, for example, for the singular value decomposition.

One of the basic classifications of square matrices is based on their invertibility. Nonsingular matrices are those that can be inverted. They have full rank, i.e. the dimension of the space spanned by their columns is equal to the dimension of the space they are in. Furthermore their determinant is nonzero. On the other hand, singular matrices do not have inverses; they are rank-deficient (not full rank), and their determinants are zero.

The eigenvectors of a diagonalizable matrix $\boldsymbol{A}$ are linearly independent, since otherwise $\boldsymbol{V}$ would be a singular matrix and could not be inverted. As a result, $\boldsymbol{A}$ would no have an eigenvalue decomposition. In the light of this short review, please answer the following questions.

(a) (5 points) Provide a diagonalizable but non-diagonal matrix with eigenvalues $-2, -1, 0, 0, 1, 2, 3, 3, 3$. Hint: You can use `diag` to put the eigenvalues on the diagonal of $\mathbf{\Sigma}$. As for $\boldsymbol{V}$, produce a random matrix and check that it is not singular (the functions `det` or `rank` can be useful).

(b) (5 points) Can the rows of your matrix be linearly independent? Why or why not? Hint: A fundamental theorem in linear algebra says that the row rank of a matrix is equal to its column rank.

(c) (10 points) Provide a set of vectors that is a basis for the null space of your matrix, and seven vectors that spans its range space. Remember that null space of a matrix $\boldsymbol{A}$, $\mathcal{N}(\boldsymbol{A})$, is defined as $\mathcal{N}(\boldsymbol{A}) = \{\mathbf{x} : \boldsymbol{A}\mathbf{x} = \mathbf{0}\}$ and the range space $\mathcal{R}(\boldsymbol{A})$, is defined as $\mathcal{R}(\boldsymbol{A}) = \{\boldsymbol{A}\mathbf{x} : \mathbf{x} \in \mathbb{R}^p\}$. Hint: Begin by replacing your matrix with its eigenvalue decomposition in these definitions.

(d) (0 points , self-study) What is the dimension of the null space of your matrix? And that of its range space, i.e. the rank? What do you expect the sum of these dimensions to be? Please justify your answer with a result from the lecture.

(e) (10 points) Compute the singular values of this matrix. If you prefer to do this using MATLAB, you are allowed to use the function `eig` but not `svd`. Hint: A relation between eigen- and singular values covered in the lecture can help you. What basic difference do you notice between these singular and eigenvalues?

(f) (0 points, self-study) We would like to obtain a matrix with orthogonal eigenvectors. Applying an orthogonalization algorithm to the eigenvectors of your existing matrix does not help us, because the resulting orthogonal vectors will not be the eigenvectors anymore, although they span the same space. Instead create a new matrix $\boldsymbol{A}$ with the same eigenvalues using an orthogonal matrix $\boldsymbol{V}$. Hint: The MATLAB functions `orth` or `qr` can help you.

Problem 3: Linear Operators and Matrices

We know from Lecture 1 that any linear operator in a finite dimensional space can be represented by a matrix. In this exercise we will implement a linear operator as matrix multiplication and see that this is sometimes beneficial and sometimes not.

As an example, we take the Discrete Cosine Transform (DCT), which is a linear operator. The DCT transforms a vector $\mathbf{x} \in \mathbb{R}^p$ to $\tilde{\mathbf{x}} \in \mathbb{R}^p$ according to

$$\tilde{\mathbf{x}}_k = \mathcal{D}(\mathbf{x}) := w_k \sum_{l=1}^{p} \mathbf{x}_l \cos\left(\frac{\pi}{2p}(k-1)(2l-1)\right)$$

where

$$w_k = \begin{cases} \frac{1}{\sqrt{p}} & \text{if } k = 1 \\ \frac{2}{\sqrt{p}} & \text{if } 2 \le k \le p \end{cases}$$

As a self study, you can show the linearity of this operator in $\mathbb{R}^p$ by simply verifying

- $\mathcal{D}(\mathbf{x} + \mathbf{y}) = \mathcal{D}(\mathbf{x}) + \mathcal{D}(\mathbf{y})$;
- $\mathcal{D}(\alpha\mathbf{x}) = \alpha\mathcal{D}(\mathbf{x})$, where α is any scalar.

The DCT matrix $\boldsymbol{D} \in \mathbb{R}^{p\times p}$ that represents this operator can be obtained using the `dctmtx` function of MATLAB, which takes the dimension of the space p as input. Then one can compute the DCT with the following matrix-vector multiplication

$$\mathcal{D}(\mathbf{x}) = \boldsymbol{D}\mathbf{x}$$

You can observe that the computational complexity of this method of implementation is of the order of p^2 when computing all the DCT coefficients using the full DCT matrix. (Otherwise we could only multiply with rows of the DCT matrix that correspond to desired coefficients). However, there is a fast transform algorithm that allows one to do this transformation in $O(plog(p))$ time, and the `dct` function of MATLAB uses this algorithm. Another benefit of this algorithm is that it does not require storing a huge DCT matrix to multiply with. As a warm-up, do the following

- Take a random vector and DCT matrix of dimension $p = 2^{12}$ and compare the outputs of matrix-vector multiplication and `dct`. Do you obtain the same DCT coefficients?
- Compare the time it takes to do this transformation on a high-dimensional vector.

Now we move to the 2D-DCT. In this case the linear operator consists of first applying the 1D transform to the rows, and then to the columns, of a 2D signal $\mathbf{X}$, i.e. $\tilde{\mathbf{X}} = \boldsymbol{D}\mathbf{X}\boldsymbol{D}^T$.

(a) (5 points) What is the approximate number of flops for explicit matrix multiplication? What is the memory required to store this matrix (as a function of p)? Time this transformation for $p = 2^7$, and report the memory use of this matrix in MATLAB's Workspace by typing `whos D`?

(b) (5 points) Compute the DCT using the `dct2` function of MATLAB. Measure the time on MATLAB for the same p value.

(c) (10 points) Give an analytical expression for the matrix $\boldsymbol{K}$ that satisfies $\boldsymbol{K}vec(\mathbf{X}) = vec(\texttt{dct2}(\mathbf{X}))$, where the function vec vectorizes a matrix $\mathbf{X}$ by stacking all its columns into a column vector (it can be carried out by typing $\mathbf{X}(:)$ in MATLAB). Hint: Slide 26 from Lecture 1 can help you.

(d) (10 points) From part (c), one can naively do the transformation by vectorizing the 2D signal, multiplying with $\boldsymbol{K}$ and reshaping into a 2D signal. What is the approximate number of flops in this case? What is the memory required to store the matrix $\boldsymbol{K}$ (as a function of p)? Observe the time and memory use on MATLAB's Workspace and compare with the previous two methods.

Now we go back to the 1D case for simplicity to investigate the scenarios where implementing the linear operator via matrix multiplication can be faster than a fast DCT transformation. Here we are interested in computing only first n elements of the DCT transformation. Take $p = 1024$ and sweep $n = 1 : 10 : p$. For each n value, take the first n rows of the pre-computed DCT matrix $\boldsymbol{D}$ and time the matrix-vector multiplication (time in total 500 of these multiplications if you want to have a stable observation).

(e) (10 points) Plot the time of the multiplication versus n and compare this with the empirical fast DCT transform times, which should be more-or-less constant as a function of n. Under approximately what value of n is matrix multiplication faster?

(f) (0 points, self-study) How would you expect this value to grow as a function of p? Hint: Consider the theoretical computational complexities of both methods.

PROBLEM 4: MATRIX FACTORIZATIONS

In this problem we will be concerned with efficiently computing the determinant of a $p \times p$ matrix $\boldsymbol{A}$, which can be defined according to the Laplace (cofactor) expansion as

$$det(\boldsymbol{A}) = \sum_{j=1}^{p} (-1)^{i+j} a_{i,j} M_{i,j}$$

where $a_{i,j}$ are the elements of matrix $\boldsymbol{A}$, and the minor $M_{i,j}$ is the determinant of the $(p-1) \times (p-1)$ matrix obtained by removing the i^{th} row and j^{th} column. As the formula suggests implicitly, you can use any index i between 1 and p.

We start with an elegant matrix called the Pascal matrix of dimension 8×8. If you do not need to know what it is, it suffices to type `pascal(8)` in MATLAB to obtain it. We are now interested in computing the determinant of this matrix.

(a) (0 points, self study) The recursive function below computes the determinant using Laplace's expansion. Compare it to the built-in `det` function of MATLAB to verify that it produces the correct determinant of the Pascal matrix.

```
function det=my_det(X)
if numel(X)==1
    % The determinant equals to the matrix itself when it is 1x1
   det=X;
else
    det=0;
    for i=1:size(X,1)
    % Minor:
    M= [ X(2:end, 1:i-1) X(2:end, i+1:end) ];
    % Laplace formula:
    det=det+(-1)^(1+i)  * X(1,i)   * my_det(M);
    end
end
```

(b) (0 points, self study) Compare the total computational time of your function with `det` on an 8×8 and then on a 9×9 matrix. Which function is faster?

(c) (0 point, self study) Prove that the computational complexity of determinant computation with Laplace expansion, i.e., number of flops, is proportional to $n!$.

By now you should have noticed that Pascal matrices are *unimodular*, i.e. they have determinant 1. You should have also noticed that computing the determinant using its Laplace expansion is not feasible. The correct way to do this is to use matrix factorizations, which require a number of flop operations that is on the order of n^3, a significant improvement over $n!$. We will see through the exercises below which matrix factorizations are good choices for this specific task of determinant computation.

(d) (0 points, self study) Let $\boldsymbol{A} = \boldsymbol{BC}$ be a factorization of the matrix $\boldsymbol{A}$, where $\boldsymbol{B}$ and $\boldsymbol{C}$ are triangular matrices. We know from Lecture 1 that $det(\boldsymbol{BC}) = det(\boldsymbol{B})det(\boldsymbol{C})$. Therefore, to show that such factorization of $\boldsymbol{A}$ can provide us its determinant efficiently, it remains to show that the determinants of triangular matrices are easy to compute. Indeed, they are equal to the product of their diagonal elements. Prove that this it true.

(e) (0 points, self study) Following the recipe in the previous part, compute the determinant of a 10×10 Pascal matrix by the eigenvalue, QR and LU decompositions. Functions such as `prod`, `diag`, `eig`, `qr` and `lu` will be useful. Exclude from the computation the determinant of Q in the QR decomposition and that of P in the LU, which are either 1 or -1. So for this problem it is enough to determine the determinant up to a sign ambiguity.

(f) (0 points, self study) Now compare the time it takes for each method. Which factorization provides the fastest computation? Hint: You can increase the simulation times with a for loop of e.g. 1000 iterations to have more stable observations about factorization times by averaging out deviations in time values. Also it is better to work with a larger dimensional random matrix, i.e., 50×50.

(g) (0 points, self study) Now assume that your matrix is a positive-definite symmetric matrix. Which decomposition would provide an even faster computation? Justify your answer analytically by comparing the computational cost of two fastest factorizations and demonstrate this advantage by timing the computation times as above. Hint: Check Lecture-1 to find a decomposition that applies to this specific type of matrices.

3 Basic Probability and Statistics *(1.5 hours)*

This part of the laboratory will prepare you for the forthcoming lectures, in which we deal with optimization problems from machine learning and statistics.

The more difficult excercises are marked with *.

Each problem begins with a warm-up question, for which we also show the solution. If you do not understand those solutions, you are free to discuss them with the TAs.

PROBLEM 1: THE MOST USEFUL BOUND IN PROBABILITY THEORY (30 points)

Recall the definition of probability measure in Page 6 of Lecture 2. In this problem, we will rigorously prove **the** most useful bound in probability theory, The Union Bound, stating that given any n events $E_1, E_2, ..., E_n$, we have

$$P(\cup_{i=1}^n E_i) \leq \sum_{i=1}^n P(E_i).$$

Prove the following statements.

(a) (0 point, warm-up) For any two events A and B such that $A \subseteq B$, let us prove that $P(A) \leq P(B)$.

We use a decomposition rule of sets: Let A, B be any sets. Then $B = (A \cap B) \cup (A^c \cap B)$, where A^c is the complement of the set A. Notice that this is a union of two *mutually exclusive (or disjoint)* sets, and hence we can apply the third axiom on slide 6 of Lecture 2 to conclude $P(B) = P(A \cap B) + P(A^c \cap B)$. Since $A \subset B$, we have $A \cap B = A$. The last step is to apply the axiom that $P(C) \geq 0$ for any event C, and therefore $P(B) = P(A) + P(A^c \cap B) \geq P(A)$.

(b) (10 points) Prove the union bound $P(\cup_{i=1}^n E_i) \leq \sum_{i=1}^n P(E_i)$.

(Hint: For any two events E_1 and E_2, prove that $P(E_1 \cup E_2) \leq P(E_1) + P(E_2)$. For this, you can use another useful decomposition rule of sets: $E_1 \cup E_2 = (E_1 \cap E_2^c) \cup E_2$. Applying this bound recursively will finish the proof.

To use the recursive arguments, you need to write the union of n events as a union of 2 events.)

(c) (10 points) When does the equality hold? Give a sufficient condition. (This means that you need to give a condition such that whenever it is satisfied, the equality holds.)

Problem 2: Deriving Estimation Error through the Central Limit Theorem (30 points)

Recall the CLT (Central Limit Theorem, slide 16 of Lecture 2), stating that for **any** random variable X and its iid (independent and identically distributed) copies $\{X_i\}_{i=1}^n$, we have $\lim_{n\to\infty} \frac{\sum_{i=1}^n X_i - n\mu}{\sigma\sqrt{n}} \to \mathcal{N}(0,1)$ in distribution. In this exercise we will see how the CLT is useful in establishing estimation error bounds for a wide class of estimators.

Consider a simple 1-dimensional estimation problem: $b_i = x^\natural + W_i$, where the noise W_i's are iid and follow an **unknown** distribution W with mean 0 and variance σ^2. Note that we do not make any assumption on the noise distribution W.

We would like to estimate $x^\natural$. Noticing that the mean of each sample is simply $x^\natural$, we can use the sample mean $\hat{x} = \frac{1}{n}\sum_{i=1}^n b_i$ to estimate $x^\natural$. Here n is the number of samples.

What is the performance of this estimator in terms of $\mathbb{E}|x^\natural - \hat{x}|^2$? If you are an experienced statistician, you can immediately guess the answer: $\mathbb{E}|x^\natural - \hat{x}|^2 \simeq \frac{\sigma^2}{n}$. Let us demonstrate this through the following elementary arguments:

(a) (0 points, warm-up) First, let us rewrite $\hat{x} - x^\natural$ as follows:

$$\begin{aligned}\hat{x} - x^\natural &= \frac{1}{n}\sum_{i=1}^n b_i - x^\natural \\ &= \left(\frac{\sum_{i=1}^n b_i - nx^\natural}{\sqrt{n}\sigma}\right)\cdot\frac{\sigma}{\sqrt{n}} \\ &\triangleq T\frac{\sigma}{\sqrt{n}}.\end{aligned}$$

By the CLT, T converges to $\mathcal{N}(0,1)$ as $n\to\infty$. Hence, $\mathbb{E}\|\hat{x} - x^\natural\|_2^2$ is approximately equal to $\mathbb{E}\|\frac{\sigma}{\sqrt{n}}N\|_2^2$, where N is standard normal $\mathcal{N}(0,1)$. To sum up,

$$\mathbb{E}|\hat{x} - x^\natural|^2 \simeq \mathbb{E}|\frac{\sigma}{\sqrt{n}}N|^2 = \frac{\sigma^2}{n}\mathbb{E}|N|^2 = \frac{\sigma^2}{n}$$

as $n\to\infty$.

(b) (10 points) Let us consider the special case where $W \sim \mathcal{N}(0,\sigma^2)$. In this case, we can directly derive $E|\hat{x} - x^\natural|^2$. Indeed, each b_i follows $\mathcal{N}(x^\natural,\sigma^2)$ and is independent of the other samples. Recall some simple rules on random variables:

(i) $\mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$ for any random variables $X_1,...,X_n$ (not necessarily independent).

(ii) $\mathbb{E}[aX] = a\mathbb{E}[X]$ for any real number a.

(iii) Let $\mathbb{V}(X)$ be the variance of X. Then for independent $X_1,...,X_n$, $\mathbb{V}(\sum_{i=1}^n X_i) = \sum_{i=1}^n \mathbb{V}(X_i)$.

(iv) $\mathbb{V}(aX) = a^2\mathbb{V}(X)$ for any real number a.

(v) If $X\sim\mathcal{N}(\mu_X,\sigma_X^2)$, $Y\sim\mathcal{N}(\mu_Y,\sigma_Y^2)$, and X and Y are independent, then $X+Y\sim\mathcal{N}(\mu_X+\mu_Y,\sigma_X^2+\sigma_Y^2)$.

Derive the the distribution of $\hat{x} - x^\natural$, and calculate $E|\hat{x} - x^\natural|^2$ in this case. Does it coincide with the result in (a)?

(c) (20 points) Following (b), for any $t>0$, provide a "meaningful" bound on $P\left(|\hat{x} - x^\natural| > t\right)$ in terms of n, σ, and t. We will not accept meaningless bounds like $P\left(|\hat{x} - x^\natural| > t\right) \le \frac{n\sigma t}{n\sigma t}$. (Hint: You can use the Chebyshev's inequality: For any random variable X with mean μ and variance σ^2, we have $P(|X-\mu|>t) \le \frac{\sigma^2}{t^2}$ for any $t>0$.)

The arguments in this exercise can be easily made rigorous. They also readily extend to estimation problems of any dimension by using multidimensional versions of the CLT, where in this case the performance measure becomes $\mathbb{E}\|\hat{x} - x^\natural\|_2^2$.

Problem 3: Performance Measures (40 points)

Recall that, for a given estimator, there exist various ways of assessing its performance (see slide 28 of Lecture 2). It is often the case that, in different literature, the performance is characterized using different metrics. Therefore, it is important to understand the relation betweens these metrics.

Recall the norm ordering: $\|x\|_\infty \le \|x\|_2 \le \|x\|_1$. In general $\|x\|_p \le \|x\|_q$ if $p \ge q$.

Let $x^\natural$ be the true parameter, and $\hat{x}$ be the estimator.

(a) (0 point, warm-up) Let us see that $\mathbb{E}\|x^\natural - \hat{x}\|_1 \le 1$ implies $\mathbb{E}\|x^\natural - \hat{x}\|_2 \le 1$, but not vice versa. Indeed, since $\|x^\natural - \hat{x}\|_2 \le \|x^\natural - \hat{x}\|_1$, in the case of $\mathbb{E}\|x^\natural - \hat{x}\|_1 \le 1$ we have $\mathbb{E}\|x^\natural - \hat{x}\|_2 \le \mathbb{E}\|x^\natural - \hat{x}\|_1 \le 1$. On the other hand, let us provide a deterministic counterexample to the reverse inequality: $\hat{x} - x^\natural = [\frac{1}{\sqrt{2}};\frac{1}{\sqrt{2}}]$. In this case, $\|\hat{x} - x^\natural\|_2 = 1$, while $\|\hat{x} - x^\natural\|_1 = \sqrt{2} > 1$.

(b) (10 points) Show that $P(\|x^\natural - \hat{x}\|_1 \leq 0.5) \leq P(\|x^\natural - \hat{x}\|_2 \leq 0.5)$.

(Hint: To be rigorous, let us define two events: $A = \{\|x^\natural - \hat{x}\|_1 \leq 0.5\}$ and $B = \{\|x^\natural - \hat{x}\|_2 \leq 0.5\}$. If you can show that $A \subseteq B$, then we can apply Problem 1.(a) to finish the proof. To show that the an event is contained in another event, you can use the definition: "*A* is contained in *B* if every element in *A* is also an element in *B*.")

(c) (10 points) Let $\hat{x}$ and $x^\natural$ both be p-dimensional. Prove that $\mathbb{E}\|x^\natural - \hat{x}\|_2 \leq \sqrt{p}\mathbb{E}\|x^\natural - \hat{x}\|_\infty$. (Hint: Can you compare $\|x\|_2$ with $\|x\|_\infty$?)

(d) (10 points) *With appropriate definitions of parameters s, p, and n, the following two results appear frequently in modern literature on high-dimensional statistics:

- $\mathbb{E}\|x^\natural - \hat{x}\|_2 \leq \sqrt{\frac{slog\frac{p}{s}}{n}}$
- $\mathbb{E}\|x^\natural - \hat{x}\|_2^2 \leq \frac{slog\frac{p}{s}}{n}$.

Are these two results comparable, in the sense that one implies the other? Are they equivalent?

(Hint: Search "Jensen's inequality".)

(e) *Recall that the estimation error can be presented in either the expected estimation error or high probability statements (slide 29 of Lecture 2). Neither of these two imply the other, but usually the high probability results are preferred. Although this can be made precise, the required techniques are beyond the scope of our course. Instead, we provide the following example, which captures the main reason why the high probability results are preferred.

- (5 points) Suppose that $x^\natural = 0.5$, and our estimator $\hat{x}$ lies uniformly in $[0, 1]$. Prove that $\mathbb{E}|x^\natural - \hat{x}| = 0.25$. What is the probability that $|x^\natural - \hat{x}| \geq 0.255$? (Hint: You need to use the definition of expectation. You may also need the pdf of the uniform distribution in $[0, 1]$, which is $f(x) = 1$ if $x \in [0, 1]$, and $f(x) = 0$ otherwise.)
- (15 points) Suppose that $x^\natural = 0.5$, our estimator $\hat{x}$ is an arbitrary random variable on the interval $[0, 1]$, and $\hat{x}$ achieves $P(|x^\natural - \hat{x}| \leq 0.25) \geq 0.99$. Prove that $\mathbb{E}|x^\natural - \hat{x}| \leq 0.255$.

This example shows that high probability results can often imply the expected estimation error results, but not vice versa.

*PROBLEM 4: CONCENTRATION OF SUB-GAUSSIAN RANDOM VARIABLES (Self-Study, 0 points)

Note: You are warmly encouraged, but not required, to finish this exercise. The bound we prove in this problem is essential to virtually all modern statistical problems. If you are interested in doing research in statistics or statistical learning theory, this will be the first thing to learn. The techniques required to finish this problem are not covered by our lecture notes. However, they should al bel covered in any basic probability course.

A zero-mean random variable X is said to be *sub-Gaussian* if $\psi_X(\lambda) \leq \frac{\lambda^2 \nu}{2}$, where $\nu > 0$ is a constant, and ψ_X is the *log moment generating function* of X:

$$\psi_X(\lambda) = \log \mathbb{E}[e^{\lambda X}],\ \lambda > 0.$$

An important property of sub-Gaussian random variable is that it always *concentrates*:

$$P(|X| > t) \leq 2e^{-\frac{t^2}{2\nu}}$$

for any $t > 0$. By concentrate we mean that, with exponentially high probability, the random variable is always near its mean (in our case the mean of X is zero). The implication of this property is that such random variables are "almost constant" when we are dealing with high-probability results.

You may use the following properties in this problem:

1. Markov Inequality: For any positive random variable Z, we have $P(Z > t) \leq \frac{\mathbb{E}[Z]}{t}$ for any $t > 0$.
2. Sum of Independent Random Variables: Let $Z = \sum_{i=1}^n X_i$, where X_i's are independent. Then $\psi_Z(\lambda) = \sum_{i=1}^n \psi_{X_i}(\lambda)$.
3. Expectation of Independent Random Variables: Let X and Y be independent random variables. Then $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.

In the following, let X be a zero-mean, sub-Gaussian random variable, and let Y, Z be any random variables. Let $\lambda, t > 0$ be any positive numbers.

(a) (0 points) Prove the generalized Markov inequality: For any positive monotonically increasing function $\phi(x) > 0$, we have

$$P(Y \geq t) \leq P(\phi(Y) \geq \phi(t)) \leq \frac{\mathbb{E}\phi(Y)}{\phi(t)}.$$

Rigorously, let us consider two events $A = \{Y \geq t\}$ and $B = \{\phi(Y) \geq \phi(t)\}$. If we can show that $A \subset B$, then we can apply Problem 1.(a) to conclude the first half of the inequality.

Your task is to show that indeed $A \subset B$, and to derive the second half of the inequality.

(b) (0 points) Show that

$$P(Y \geq t) \leq \exp(-\psi_Y^*(t))$$

where

$$\psi_Y^*(t) = \sup_{\lambda \geq 0}(\lambda t - \psi_Y(\lambda)).$$

(Hint: Apply (a) with $\phi(t) = e^{\lambda t}$ for $\lambda > 0$.)

(c) (0 points) Prove the Sub-Gaussian concentration:

$$P(|X| > t) \leq 2e^{-\frac{t^2}{2\nu}}.$$

(Hint: Recall that $\psi_X(\lambda) \leq \frac{\lambda^2 \nu}{2}$. Can you provide a bound for $\psi_X^*(\lambda)$?)

(d) (0 points) Let $Z = \sum_{i=1}^n X_i$, where the X_i's are iid copies of X. Prove that

$$\psi_Z(\lambda) = n\psi_X(\lambda).$$

(e) (0 points) Following (d), prove that

$$P\left(\left|\frac{Z}{n}\right| > t\right) \leq 2e^{-\frac{nt^2}{2\nu}}.$$

The last result shows that the sample average of Sub-Gaussian random variables concentrates tightly around its mean, in the sense that the exponential rate of decay grows linearly with the number of samples n.

The meaning of the name "sub-Gaussian" stems from the fact that the "tail" ($P(|X| > t)$) of a sub-Gaussian random variable is "lighter" than the Gaussian random variable, as shown in (c). In probability theory, "light tail" means "well-behaved" or "less randomness". For example, in the case that $\nu = 0$, X is deterministic, and there is no randomness. Such an understanding plays a key role in many applications of modern probability theory, including statistics, machine learning, and stochastic optimization.

4 Convexity of functions and sets *(1 hour)*

PROBLEM 1: CONVEXITY DEFINITIONS (15 points)

In this problem, we investigate the equivalence of two alternative definitions of a convex function. For simplicity, we consider only univariate functions, with $\mathrm{dom}(f) = \mathbb{R}$, in this exercise. However, the result also holds for multivariate functions on any convex domain.
Recall that a function $f : \mathbb{R} \to \mathbb{R}$ is convex if and only if $\forall x_1, x_2 \in \mathbb{R}, \forall \alpha \in [0, 1]$, we have:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

(a) (10 points) Show that any convex differentiable function f satisfies, $\forall x, y \in \mathbb{R}$,

$$f(y) \geq f(x) + f'(x)(y - x) \tag{1}$$

where $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$ is the derivative of f at x.
HINT: Start by showing that $\frac{f(x+\alpha(y-x))-f(x)}{\alpha} \leq f(y) - f(x)$ holds $\forall x, y \in \mathbb{R}, \alpha \in (0, 1]$.

(b) (5 points) Show that any differentiable function f that satisfies $f(y) \geq f(x) + f'(x)(y-x), \forall x, y \in \mathbb{R}$ is convex.
HINT: The result follows by applying the inequality (1) twice.

PROBLEM 2: STATIONARY POINTS (16 points)

In this problem, we observe some of the challenges we encounter when trying to find the optimal point of a function.

(a) (3 points) Plot the function $f(x) = (x^3 - x)^3$ in MATLAB (or your favorite plotter). You can use the following code:

```
x=-1.1:.01:1.1;
plot(x, (x.^3-x).^3)
shg
```

By observing the graph, can you identify, approximately, the stationary points of f?

(b) (2 points) What is the minimum value achieved by f ?

(c) (8 points) Derive the gradient of f, and compute the stationary points of f.

(d) (3 points) Is f a convex function? Explain briefly.

PROBLEM 3: LOG-DETERMINANT (25 points)

The inputs of functions are not limited to scalars and vectors; they can also be matrices. In this problem, we study one such interesting function: $f(X) = -\log\det\mathbf{X}$, defined for $\mathbf{X} \succ 0, \mathbf{X} \in \mathbb{R}^{n\times n}$. The goal of this problem is to prove the convexity of f.

(a) (7 points) Show that the function $h(t) = -\log(1+\lambda t)$, defined for some $\lambda > 0$ is convex, on its domain $(\frac{-1}{\lambda}, +\infty)$. Is it strictly convex? HINT: You can use a property of the second derivative of convex functions.

(b) (5 points) Type the following code in MATLAB:

```
n = 5;
I = eye(n); % Generate an nxn Identity matrix
t= rand;
M=randn(n); % Generate an nxn Random matrix
eig(I + t*M)
1+t*eig(M)
```

Repeat the experiment a few times with different values of n, t, M. You should observe that the eigenvalues of $\boldsymbol{I} + t\boldsymbol{M}$ for any $t \in \mathbb{R}, M \in \mathbb{R}^{n\times n}$ are $1 + t\lambda_i$ where λ_i are the eigenvalues of $\boldsymbol{M}$. Using the fact that the eigenvalues of any square matrix $\boldsymbol{M}$ are the roots of the characteristic polynomial $det(\boldsymbol{M} - \lambda\boldsymbol{I}) = 0$, show that the above observation is indeed always true.

(c) (10 points) The following proposition gives another equivalent definition of convexity. For the purpose of this question, you can assume it is true (you are encouraged to prove it to yourself).

Proposition 1. *A function $f : \mathbb{R}^{n\times m} \to \mathbb{R}$ is convex if and only if it is convex along each line, i.e., the function $g : \mathbb{R} \to \mathbb{R}$, $g(t) = f(\mathbf{Z} + t\mathbf{V})$, with $\mathrm{dom}(g) = \{\mathbf{Z} + t\mathbf{V} \in \mathrm{dom}(f)\}$, is convex in t, for any $\mathbf{Z} \in \mathrm{dom}(f), \mathbf{V} \in \mathbb{R}^{n\times n}$.*

By Proposition 1, it is sufficient to prove that f is convex along each line to prove its convexity. Given any two symmetric matrices $\mathbf{Z}, \mathbf{V}$ with $\mathbf{Z} \succ 0$, we define $g(t) = f(\mathbf{Z} + t\mathbf{V})$ for any t such that $\mathbf{Z} + t\mathbf{V} \succ 0$. Show that

$$g(t) = -\sum_{i=1}^{n} \log(1 + t\lambda_i) - \log\det\mathbf{Z}$$

where λ_i are the eigenvalues of $\mathbf{Z}^{-1/2}\mathbf{V}\mathbf{Z}^{-1/2}$. Here $\mathbf{Z}^{-1/2}$ denotes the square root of $\mathbf{Z}^{-1}$; i.e., $\mathbf{Z}^{-1/2}\mathbf{Z}^{-1/2} = \mathbf{Z}^{-1}$ (the square root of a matrix is unique for any positive semi-definite matrix).

(d) (3 points) What property of convex functions allows us to conclude that g is convex?

PROBLEM 4: CONVEX SETS (14 points)

Which of the following sets are convex? Explain.

1. The set $\{\mathbf{x} \mid \alpha \le \mathbf{a}^T\mathbf{x} \le \beta\}$
2. The set of symmetric matrices $\{\mathbf{X} \mid \mathbf{X} = \mathbf{X}^T\}$
3. The set of s-sparse vectors $\{\mathbf{x} \mid \|\mathbf{x}\|_0 \le s\}$
4. The affine hyperplane $\{\mathbf{x} \mid \boldsymbol{A}\mathbf{x} = \boldsymbol{b}\}$
5. The intersection of r convex sets $\mathcal{S}_1, \cdots, \mathcal{S}_r$, i.e. $\{\mathbf{x} \mid \mathbf{x} \in \bigcap_{i=1}^r \mathcal{S}_i\}$
6. The union of r convex sets $\mathcal{S}_1, \cdots, \mathcal{S}_r$, i.e. $\{\mathbf{x} \mid \mathbf{x} \in \bigcup_{i=1}^r \mathcal{S}_i\}$
7. The epigraph of a convex function $f : Q \to \mathbb{R}$, where $Q \subseteq \mathbb{R}$, $\{(x, y) \mid x \in Q, y \in \mathbb{R}, f(x) \le y\}$

5 Asymptotic notation *(30 min)*

This part of the laboratory is meant to help you revise and get more comfortable with the asymptotic notation we saw in Lecture 3.

Problem 1: Growth of the factorial (20 points)

(a) We want to show that $\log n! \in \Theta(n \log n)$, for $n \in \mathbb{N}$. Recall that $f(n) \in \Theta(g(n))$ iff $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$:

  (*i*) (5 points) Recall the definition $f(n) \in O(g(n))$ iff $\exists c > 0, \exists n_0,$ such that $|f(n)| \leq c|g(n)|, \forall n \geq n_0$. Show that $\log n! \in O(n \log n)$.

  (*ii*) (5 points) Recall the definition $f(n) \in \Omega(g(n))$ iff $\exists c > 0, \exists n_0,$ such that $|f(n)| \geq c|g(n)|, \forall n \geq n_0$. Show that $\log n! \in \Omega(n \log n)$.
  HINT: Start by showing that $n! \geq (\frac{n}{2})^{\frac{n}{2}}, \forall n \geq 1$. For notational simplicity, you can assume that n is even.

(b) (5 points) Recall the definition $f(n) \in o(g(n))$ iff $\lim_{n\to+\infty} \frac{|f(n)|}{|g(n)|} = 0$. Show that $n! \in o(n^n)$.
HINT: Start by showing that $\frac{n!}{n^n} \leq 2^{-n/2}, \forall n \geq 1$. For notational simplicity, you can assume that n is even.

(c) (5 points) Recall the definition $f(n) \in \omega(g(n))$ iff $\lim_{n\to+\infty} \frac{|f(n)|}{|g(n)|} = +\infty$. Show that $n! = \omega(2^n)$.
HINT: Start by showing that $\frac{n!}{2^n} \geq \frac{n}{4}, \forall n \geq 1$.

Problem 2: Ordering functions by asymptotic growth (10 points)

Rank the following functions in increasing order of growth; that is, find an arrangement $g_1, g_2, ..., g_{10}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), ..., g_9 = \Omega(g_{10})$. Also identify functions of equivalent growth, i.e., $g_i(n) = \Theta(g_j(n))$.

$$g_1(n) = \log\log n,\quad g_2(n) = n^{0.1},\quad g_3(n) = 2^n,\quad g_4(n) = 1,\quad g_5(n) = n,\quad g_6(n) = n!,\quad g_7(n) = n^{\frac{1}{\log n}},\quad g_8(n) = \log n,\quad g_9(n) = (\sqrt{2})^{\log_2 n},\quad g_{10}(n) = \log^{51} n.$$

HINT: Recall that $a^b = e^{b \log a} = 2^{b \log_2 a}$.

References

[1] Martin Jaggi and Marek Sulovsky *A Simple Algorithm for Nuclear Norm Regularized Problems.* ICML 2010 - 27th International Conference on Machine Learning, June 21-24, 2010.

[2] Richard von Mises and H. Pollaczek-Geiringer *Praktische Verfahren der Gleichungsauflösung,* ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik 9, 152-164, 1929.

Lab #1
EE-556 Fall 2015
Instructor Verification Sheet

Name: ____________________________ Date of Lab: ___________

1 A Matlab Tour of Linear Algebra

Problem 1: Matrix Norms

(d) (5 points) Verified: ________________ Date: ___________

(e) (5 points) Verified: ________________ Date: ___________

(f) (5 points) Verified: ________________ Date: ___________

(g) (5 points) Verified: ________________ Date: ___________

Problem 2 : Some Basics of Linear Algebra

(a) (5 points) Verified: ________________ Date: ___________

(b) (5 points) Verified: ________________ Date: ___________

(c) (5 points) Verified: ________________ Date: ___________

(e) (5 points) Verified: ________________ Date: ___________

Problem 3: Linear Operators and Matrices

(a) (5 points) Verified: ________________ Date: ___________

(b) (5 points) Verified: ________________ Date: ___________

(c) (5 points) Verified: ________________ Date: ___________

(d) (5 points) Verified: ________________ Date: ___________

(e) (10 points) Verified: ________________ Date: ___________

2 Basic Probability and Statistics

Problem 1: THE MOST USEFUL BOUND IN PROBABILITY THEORY

(b) (10 points) Verified: ________________ Date: ___________

(c) (10 point) Verified: ________________ Date: ___________

Problem 2: Deriving estimation error through the central limit theorem

(b) (10 points) Verified: ________________ Date: ___________

(c) (20 points) Verified: ________________ Date: ___________

Problem 3: PERFORMANCE MEASURES

(b) (10 point) Verified: ________________ Date: ___________

(c) (10 points) Verified: ________________ Date: ___________

(d) (10 points) Verified: ________________ Date: ___________

(e)
- (5 points) Verified: ________________ Date: ___________
- (15 points) Verified: ________________ Date: ___________

3 Convexity of functions and sets

Problem 1: Convexity definitions

(a) (10 points) Verified: ________________ Date: ___________

(b) (5 points) Verified: ________________ Date: ___________

Problem 2: Stationary points

(a) (3 points) Verified: ________________ Date: ___________

(b) (2 points) Verified: ________________ Date: ___________

(c) (8 points) Verified: ________________ Date: ___________

(d) (3 points) Verified: ________________ Date: ___________

Problem 3: Log-Determinant

(a) (7 points) Verified: ________________ Date: ___________

(b) (5 points) Verified: ________________ Date: ___________

(c) (10 points) Verified: ________________ Date: ___________

(d) (3 points) Verified: ________________ Date: ___________

Problem 4: Convex sets

(14 points) Verified: ________________ Date: ___________

4 Asymptotic Notation

Problem 1: Growth of the factorial

(a) (*i*) (5 points) Verified: ________________ Date: ___________

(*ii*) (5 points) Verified: ________________ Date: ___________

(b) (5 points) Verified: ________________ Date: ___________

(c) (5 points) Verified: ________________ Date: ___________

Problem 2: Ordering functions by asymptotic growth

(10 points) Verified: ________________ Date: ___________