Introduction to Machine Learning (67577)

Exercise 1
Linear Algebra, Infi, Probability

March 2019

Contents

Submission guidelines **2**

Warm-up - Algebra Recap **2**

SVD **3**

Power Iteration **3**

Projection matrices **4**

Multivariate Calculus **4**

Multivariate Gaussian- practical question **5**

Concentration inequalities - practical question **5**

Concentration inequalities - Biased Coins **6**

Submission guidelines

- The assignment is to be submitted via Moodle only.
- Files should be zipped to ex_1_First_Last.zip (First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- All answers to both theoretical part and practical part, should be submitted as a single pdf file. We encourage you to typeset your answers (either Lyx or Word), but it is not mandatory. Students who choose to submit handwritten scans are warned that incomprehensible answers due to handwriting or bad scan, will result in point reduction just like wrong answers are. Moreover, the scans should be integrated with the plots to a single coherent pdf file.
- Each question is enumerated. Use those numbers when answering your questions.
- FIGURES: Your results should be self explanatory and readable: each figure should have a **unique informative title** and each axis should have a **meaningful label**. How to add title and axis labels? For instance, click here. Ambiguous graphs will result in point reduction for the whole answer.
- CODE: All of your Python code files used in the practical question should be attached in the zip file. Other then that there are no special requirements or guidelines for your code. Note that if your code doesn't run on the CS environment you will not get points for this question.
- Note: Although **some of the questions are optional (i.e. worth 0 points)**, it is highly recommended to answer all questions- they related to the following lectures and will help you to get better understanding.
- To conclude: Submit one zip file containing
  - 1 pdf, with answers to both theoretical and practical tasks.
  - .py files for the two practical tasks.

Warm-up - Algebra Recap

1. (optional) For the three subsets defined in class, prove that each is a vector space (i.e., fulfilling all required axioms):
   - $im(A)$
   - $im(A^\top)$
   - $ker(A)$
2. (optional) If X and Y are affine spaces, then every affine transformation $f\colon X \to Y$ is of the form $x \mapsto Mx + b$, where $M\colon X \to Y$ is a linear transformation, x is a vector in X, and b is a vector in Y. Prove or give a counterexample:
   (a) Every affine transformation is a linear transformation.

(b) Every linear transformation is a affine transformation.

3. (2pt) Calculate the projection of $v = (1, 2, 3, 4)$ on the vector $w = (0, -1, 1, 2)$.

4. (2pt) Calculate the projection of $v = (1, 2, 3, 4)$ on the vector $w = (1, 0, 1, -1)$.

5. (optional) Prove that the angle between two non-zero vectors $v, w \in \mathbb{R}^m$ is ± 90 iff $\langle v, w \rangle = 0$.

6. (optional)Prove that Orthonormal matrices are isometric transformations.

SVD

7. (6pt) Assume A is invertible. Write a formula for the inverse of A using only the matrices U, D, V where UDV^T is the SVD decomposition of A. Many applications use the inverse of given matrices. Explain why knowing the SVD decomposition of matrix is usefull in this context.

8. (10pt) Find the SVD of
$$C = UDV^T = \begin{pmatrix} 5 & 5 \\ -1 & 7 \end{pmatrix}$$
I.e., find the matrices U, D, V^T where U, V are orthogonal matrices and D is diagonal.

   Do the following steps:

   - Calculate $C^T C$.
   - Deduce V and D (hint: use the *Eigenvalues Decomposition*; You can either use the function *numpy.linalg.eig* in python or refresh your memory and do it manually).
   - Find U using the equality $CV = UD$.

Power Iteration

In this section we will implement an algorithm for SVD decomposition, we will use the relation between SVD of A to EVD of $A^T A$ that we saw in the tirgul:

9. (7pt) For some $A \in M_{m \times n}(\mathbb{R})$, define $C_0 = A^T A$.

   Let $\lambda_1, \lambda_2, ...\lambda_n$ be the eigenvalues of C_0, with the corresponding eigenvectors $v_1, v_2, ...v_n$, ordered such that $\lambda_1 \geqslant \lambda_2 \geqslant ... \geqslant \lambda_n$.

   Assume $\lambda_1 > \lambda_2$, where λ_1 is the largest eigenvalue and λ_2 is the second-largest one.

   Define $b_{k+1} = \frac{C_0 b_k}{\|C_0 b_k\|}$, and initialize b_0 randomly.

   Show that: $\lim_{k \to \infty} b_k = v_1$

   Hint: use EVD decomposition of C_0 and represent b_0 accordingly. You can assume that $b_0 = \sum_{i=0}^{n} a_i v_i$, where $a_1 \neq 0$. As b_0 is initialized randomly, the probability of $a_1 = 0$ is zero.

   Optional:

   Why is the assumption $\lambda_1 > \lambda_2$ necessary? Give an example of a matrix C_0 with $\lambda_1 = \lambda_2$ and b_0 where $\lim_{k \to \infty} b_k$ does not exist. C_0 in your example can be any matrix, not necessarily equal to $A^T A$ for any A.

10. (optional) Write python function "get_largest_eigenvector(C, num_iters)" which gets a symmetric numpy 2d array C and a number of iterations. The function should return b_{num_iters}, in numpy 1d array as approximation to the eigenvector of the largest eigenvalue of C.

11. (optional) Define: $B_1 = \lambda_1 v_1 v_1^T$, and $C_1 = C_0 - B_1$. Show that if we apply the procedure above to C_1 we will get $\lim_{k\to\infty} b_k = v_2$ (v_2 is the eigenvector corresponding to the second largest eigenvalue of C_0)

12. (optional) Write python function "get_eigenvectors(C, num_iters)" which gets numpy 2d array C and returns numpy 2d array coresponding to the eigenvectors of C.

13. (optional) Write python function "SVD(A)" which gets numpy 2d array A and returns:

    - numpy 2d array coresponding to the left singular vectors of A
    - numpy 1d array coresponding to the singular values of A
    - numpy 2d array coresponding to the right singular vectors of A

    (find U, D using the equality $CV = UD$)

Projection matrices

Let V be a k-dimensional subspace of $\mathbb{R}^d$, and let $v_1, v_2, ..., v_k \in \mathbb{R}^d$ be an orthonormal basis of V . Define $P = \sum_{i=1}^{k} v_i v_i^\top$. The matrix P is called a "projection matrix" onto the subspace V. Prove the following useful properties of projection matrices:

14. (3pt) P is symmetric.

15. (3pt) The vectors $v_1, v_2, ..., v_k$ are eigenvectors of P, all with eigenvalue equal to 1.

16. (3pt) Write a possible EVD of P.

17. (3pt) $P^2 = P^\top P = PP^\top = P$

18. (optional) $(I - P)\, P$ is the 0 matrix.

19. (3pt) $x \in V \Rightarrow Px = x$.

Multivariate Calculus

20. (optional) Let $x \in \mathbb{R}^n$ be a fixed vector and $U \in \mathbb{R}^{n\times n}$ a fixed orthogonal matrix. Calculate the Jacobian of the function $f : \mathbb{R}^n \to \mathbb{R}^n$:

$$f(\sigma) = U diag(\sigma) U^T x$$

    Here $diag(\sigma)$ is an $n \times n$ matrix where $diag(\sigma)_{ij} = \begin{cases} \sigma_i & i = j \\ 0 & i \neq j \end{cases}$

21. (7pt) Use the chain rule to calculate the gradient of h:

$$h(\sigma) = \frac{1}{2} \|f(\sigma) - y\|^2$$

22. (7pt) Finish the calculation of the Jacobian of the softmax function (we started in the tirgul):

$$g(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Multivariate Gaussian- practical question

In this question we will examine the three-dimensional case. We want to show how linear transformations affect the data set and in result the covariance matrix. First we will generate random points with mean values at the origin and unit variance $\sigma(x) = \sigma(y) = \sigma(z) = 1$.

23. (4pt) Download the file 3d_gaussian.py. Use the identity matrix as the covariance matrix to generate random points and than plot them (with the given function).

24. (4pt) Transform the data with the following scaling matrix:

$$S = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

    Plot the new points. What does the covariance matrix look like now (both analytically and numerically)?

25. (4pt) Multiply the scaled data by random orthogonal matrix. Plot the new points. What does the covariance matrix look like now?

26. (4pt) In the Tirgul we claimed that the marginal distribution of a gaussian is still gaussian. Plot the projection of the data to the x, y axes. What does it look like? Add the plot to the submission.

27. (4pt) In the Tirgul we claimed that the conditional distribution of a gaussian is still gaussian. Only for points where $0.1 > z > -0.4$: Plot the projection of the points to the x, y axes. What does it look like? Add the plot to the submission.

Concentration inequalities - practical question

In this practical question we will visualize the concentration inequalities seen in Tirgul 2. Reminder: we saw 2 results regarding the sample complexity (the number of samples needed to ensure the approximation (ϵ) and confidence (δ) levels desired). The first result is corollary 2 (that uses Chebyshev's inequality) and the second result is corollary 4 (that uses Hoeffding's inequality). We would like to compare between these 2 bounds with empirical data.

28. (7pt) We have a coin with unknown bias p that we wish to estimate with accuracy ϵ and confidence δ. What are the m we found in tirgul 2 such that if we have more than m i.i.d samples of our coin we can estimate p with the accuracy and confidence needed ?

29. (17pt) You are given 100000 sequences of 1000 coin tosses (arranged in a matrix, "data", of 100000 rows and 1000 columns. To generate the data use the commands:

    - import numpy
    - data = numpy.random.binomial(1, 0.25, (100000,1000))

Define a variable "epsilon" which gets the values $[0.5, 0.25, 0.1, 0.01, 0.001]$.

(a) For the first 5 sequences of 1000 tosses (the first 5 rows in "data"), plot the estimate $\overline{X}_m = \frac{1}{m}\sum_{i=1}^{m} x_i$ as a function of m (i.e the mean of all tosses up to m). 1 figure with 5 plots (each row in a different color). What do you expect to see in this plot as m grows?

(b) For each bound (Chebyshev and Hoeffding) and for each ϵ, plot the upper bound on $P_{X_1,\ldots,X_m}\left(\left|\overline{X}_m - \mathbb{E}[X]\right| \geqslant \epsilon\right)$ (derived in class) as a function of m (where m ranges from 1 to 1000). 5 figures with 2 plots each (mention in the title of each plot what is ϵ and use a different color for each bound)[1].

(c) You are now told that $p = 0.25$. On top of the figures from the previous question, plot the percentage of sequences that satisfy $\left|\overline{X}_m - \mathbb{E}[X]\right| \geqslant \epsilon$ as a function of m (now you know $\mathbb{E}[X] = p = 0.25$). What are you expecting to see in these plots? Explain.

Concentration inequalities - Biased Coins

Consider N coins $Z_1, \ldots, Z_N$, where the bias of the i-th coin is denoted by p_i, and the corresponding distribution is denoted by $\mathcal{D}_i$. A learning algorithm gets as an input N independent sequences of size m, where the i-th sequence is drawn i.i.d. according to $\mathcal{D}_i$. The algorithm's goal is to find the index of a coin with minimal bias. As in the coin prediction problem, we introduce two parameters $\epsilon, \delta \in (0,1)$, and require that with probability of at least $1-\delta$, the algorithm returns an index $\hat{i}$ such that

$$p_{\hat{i}} \leqslant \min_{i \in N} p_i + \epsilon \ .$$

30. (optional) Let $\epsilon \in (0,1)$. For each coin Z_i, let E_i be the event consisting of the inputs for which $|\hat{p}_i - p_i| > \epsilon/2$. Let $E = \bigcup_{i=1}^{N} E_i$. Show that if E doesn't occur, then for every $j \in [N]$, we have

$$p_{\hat{i}} \leqslant p_j + \epsilon \ .$$

31. (optional) Let $\delta \in (0,1)$. Use Hoeffding's inequality to bound the probability of E_i by δ/N (i.e. find m which yield this bound over the probability)

32. (optional) Deduce a learning algorithm for this task whose sample complexity (the size m of every one of the N sequences above that is needed by the algorithm in order to satisfy the requirements, as a function of ϵ and δ) is bounded by $m(\epsilon,\delta) \leqslant \left\lceil \frac{2}{\epsilon^2} \cdot \log(\frac{2N}{\delta}) \right\rceil$.

Assume now that there exists at least one index $i \in [N]$ such that $p_i = 0$. Of course, the algorithm does not know the identity of this index.

33. (optional) Let $\epsilon \in (0,1)$. As before, we define a "bad" event E_i for each coin. If $p_i > \epsilon$, let E_i be the event consisting of the inputs for which $\hat{p}_i = 0$. Otherwise, let $E_i = \varnothing$. Let $E = \bigcup_{i=1}^{N} E_i$. Show that for every $j \in [N]$, we have $p_j > \epsilon \Longrightarrow \mathbb{P}(E_j) \leqslant \exp(-\epsilon m)$

34. (optional) Conclude a learning algorithm for this task whose sample complexity of the algorithm is bounded by $m(\epsilon,\delta) \leqslant \left\lceil \frac{\log(N/\delta)}{\epsilon} \right\rceil$.

[1] if the bound calculated is greater than 1, you should substitute it with 1. This is because we are bounding a probability which is always smaller than 1.