COMP9417 - Machine Learning
Homework 3: MLEs and Kernels

Introduction In this homework we first continue our exploration of bias, variance and MSE of estimators. We will show that MLE estimators are not unnecessarily unbiased, which might affect their performance in small samples. We then delve into kernel methods: first by kernelizing a popular algorithm used in unsupervised learning, known as K-means. We then look at Kernel SVMs and compare them to fitting linear SVMs with feature transforms.

Points Allocation There are a total of 28 marks.

- Question 1 a): 2 marks
- Question 1 b): 2 marks
- Question 1 c): 4 marks
- Question 2 a): 1 mark
- Question 2 b): 1 mark
- Question 2 c): 2 marks
- Question 2 d): 2 marks
- Question 2 e): 2 marks
- Question 2 f): 3 marks
- Question 2 g): 2 marks
- Question 3 a): 1 mark
- Question 3 b): 1 mark
- Question 3 c): 1 mark
- Question 3 d): 1 mark
- Question 3 e): 3 marks

What to Submit

- A **single PDF** file which contains solutions to each question. For each question, provide your solution in the form of text and requested plots. For some questions you will be requested to provide screen shots of code used to generate your answer — only include these when they are explicitly asked for.

- **.py file(s) containing all code you used for the project, which should be provided in a separate .zip file.** This code must match the code provided in the report.
- You may be deducted points for not following these instructions.
- You may be deducted points for poorly presented/formatted work. Please be neat and make your solutions clear. Start each question on a new page if necessary.
- You **cannot** submit a Jupyter notebook; this will receive a mark of zero. This does not stop you from developing your code in a notebook and then copying it into a .py file though, or using a tool such as **nbconvert** or similar.
- We will set up a Moodle forum for questions about this homework. Please read the existing questions before posting new questions. Please do some basic research online before posting questions. Please only post clarification questions. Any questions deemed to be *fishing* for answers will be ignored and/or deleted.
- Please check Moodle announcements for updates to this spec. It is your responsibility to check for announcements about the spec.
- Please complete your homework on your own, do not discuss your solution with other people in the course. General discussion of the problems is fine, but you must write out your own solution and acknowledge if you discussed any of the problems in your submission (including their name(s) and zID).
- As usual, we monitor all online forums such as Chegg, StackExchange, etc. Posting homework questions on these site is equivalent to plagiarism and will result in a case of academic misconduct.
- You may **not** use SymPy or any other symbolic programming toolkits to answer the derivation questions. This will result in an automatic grade of zero for the relevant question. You must do the derivations manually.

When and Where to Submit

- Due date: Week 8, Monday **July 15th**, 2024 by **5pm**. Please note that the forum will not be actively monitored on weekends.
- Late submissions will incur a penalty of 5% per day **from the maximum achievable grade**. For example, if you achieve a grade of 80/100 but you submitted 3 days late, then your final grade will be $80 - 3 \times 5 = 65$. Submissions that are more than 5 days late will receive a mark of zero.
- Submission must be made on Moodle, **no exceptions**.

Question 1. Maximum Likielihood Estimators and their Bias

Let $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} N(\mu, \sigma^2)$. Recall that in Tutorial 2 we showed that the MLE estimators of μ, σ^2 were $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$ where

$$\hat{\mu}_{\text{MLE}} = \overline{X}, \qquad \text{and} \qquad \hat{\sigma}^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

In this question, we will explore these estimators in more depth.

(a) Find the bias and variance of both $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$. **Hint:** You may use without proof the fact that

$$\text{var}\left(\frac{1}{\sigma^2}\sum_{i=1}^{n}(X_i - \overline{X})^2\right) = 2(n-1)$$

What to submit: the bias and variance of the estimators, along with your working.

(b) Your friend tells you that they have a much better estimator for σ^2, namely:

$$\tilde{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

Discuss whether this estimator is better or worse than the MLE estimator:

$$\hat{\sigma}^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

Be sure to include a detailed analysis of the bias and variance of both estimators, and describe what happens to each of these quantities (for each of the estimators) as the sample size n increases (use plots). For your plots, you can assume that $\sigma = 1$.

What to submit: the bias and variance of the new estimator. A plot comparing the bias of both estimators as a function of the sample size n, a plot comparing the variance of both estimators as a function of the sample size n, use labels/legends in your plots. A copy of the code used here in solutions.py

(c) Compute and then plot the MSE of the two estimators considered in the previous part. For your plots, you can assume that $\sigma = 1$. Provide some discussion as to which estimator is better (according to their MSE), and what happens as the sample size n gets bigger. *What to submit: the MSEs of the two variance estimators. A plot comparing the MSEs of the estimators as a function of the sample size n, and some commentary. Use labels/legends in your plots. A copy of the code used here in solutions.py*

Question 2. A look at clustering algorithms

Note: Using an existing/online implementation of the algorithms described in this question will result in a grade of zero. You may use code from the course with reference.

The K-means algorithm is the simplest and most intuitive clustering algorithm available. The algorithm takes two inputs: the (unlabeled) data $X_1, \ldots, X_n$ and a desired number of clusters K. The goal is to identify K groupings (which we refer to as clusters), with each group containing a subset of the original data points. Points that are deemed similar/close to each other will be assigned to the same grouping. Algorithmically, given a set number of iterations T, we do the following:

1. Initialization: start with initial set of K-means (cluster centers): $\mu_1^{(0)}, \mu_2^{(0)}, \ldots, \mu_K^{(0)}$.
2. For $t = 1, 2, 3, \ldots, T$:

- For $i = 1, 2, \ldots, n$: Find nearest mean to X_i by solving

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|X_i - \mu_k^{(t-1)}\|_2^2.$$

- For $k = 1, \ldots, K$: set[1] [2]

$$C_k^{(t)} = \{X_i \text{ such that } k_i = k\}$$
$$\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{X_i \in C_k^{(t)}} X_i.$$

(a) Consider the following data-set of $n = 5$ points in $\mathbb{R}^2$:

$$\left\{ \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \end{pmatrix}, \begin{pmatrix} 6 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix} \right\}.$$

You run K-means on this dataset with $K = 2$ and initial cluster centers $\mu_1^{(0)} = (5, 2)^T, \mu_2^{(0)} = (4, 5)^T$. Compute the cluster centers at the next two iterations: $\mu_1^{(1)}, \mu_2^{(1)}$ and $\mu_1^{(2)}, \mu_2^{(2)}$ by hand. Be sure to show your working.

What to submit: your cluster centers and any working, either typed or handwritten.

(b) Your friend tells you that they are working on a clustering problem at work. You ask for more details and they tell you they have an unlabelled dataset with $p = 10000$ features and they ran K-means clustering using Euclidean distance. They identified 52 clusters and managed to define labellings for these clusters based on their *expert* domain knowledge. What do you think about the usage of K-means here? Do you have any criticisms?

What to submit: some commentary.

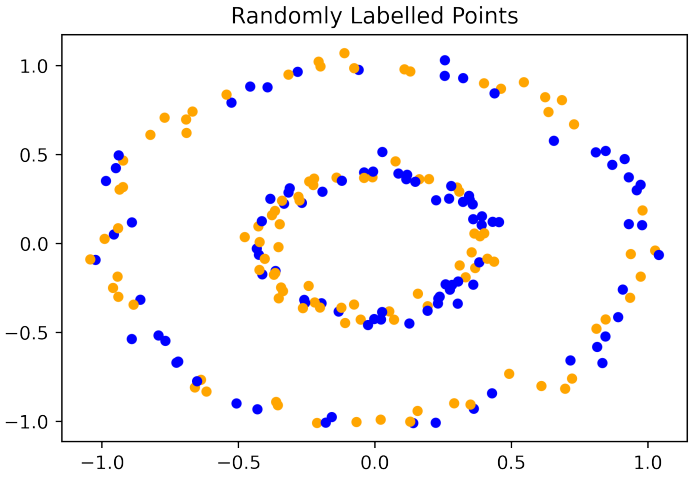
(c) Consider the data and random clustering generated using the following code snippet:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets

X, y = datasets.make_circles(n_samples=200, factor=0.4, noise=0.04, random_state=13)
colors = np.array(['orange', 'blue'])

np.random.seed(123)
random_labeling = np.random.choice([0,1], size=X.shape[0], )
plt.scatter(X[:, 0], X[:, 1], s=20, color=colors[random_labeling])
plt.title("Randomly Labelled Points")
plt.savefig("Randomly_Labeled.png")
plt.show()

```

The random clustering plot is displayed here:

[1] Recall that for a set S, $|S|$ denotes its cardinality. For example, if $S = \{4, 9, 1\}$ then $|S| = 3$.

[2] The notation in the summation here means we are summing over all points belonging to the k-th cluster at iteration t, i.e. $C_k^{(t)}$.



Implement K-means clustering from scratch on this dataset. Initialize the following two cluster centers:

$$\mu_1^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mu_2^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

and run for 10 iterations. In your answer, provide a plot of your final clustering (after 10 iterations) similar to the randomly labeled plot, except with your computed labels in place of `random_labelling`. Do you think K-means does a good job on this data? Provide some discussion on what you observe. *What to submit: some commentary, a single plot, a screen shot of your code and a copy of your code in your .py file.*

(d) You decide to extend your implementation by considering a feature transformation which maps 2-dimensional points (x_1, x_2) into 3-dimensional points of the form $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. Run your K-means algorithm (for 10 iterations) on the transformed data with cluster centers:

$$\mu_1^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \mu_2^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}.$$

Note for reference that the nearest mean step of the algorithm is now:

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\| \phi(X_i) - \frac{1}{|C_k^{(t-1)}|} \sum_{X_j \in C_k^{(t-1)}} \phi(X_j) \right\|_2^2, \tag{1}$$

where $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$. In your answer, provide a plot of your final clustering using the code provided in (c) as a template. Provide some discussion on what you observe. *What to submit: a single plot, a screen shot of your code and a copy of your code in your .py file, some commentary.*

(e) You recall (from lectures perhaps) that directly applying a feature transformation to the data can be computationally intractable, and can be avoided if we instead write the algorithm in terms of

a function h that satisfies: $h(x, x') = \langle \phi(x), \phi(x') \rangle$. Show that the nearest mean step in (1) can be re-written as:

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \big(h(X_i, X_i) + T_1 + T_2 \big),$$

where T_1 and T_2 are two separate terms that may depend on $C_k^{(t-1)}$, $h(X_i, X_j)$ and $h(X_j, X_\ell)$ for $X_j, X_\ell \in C_k^{(t-1)}$. The expressions should **not** depend on ϕ. *What to submit: your full working.*

(f) With your answer to the previous part, you design a new algorithm: Given data $X_1, \ldots, X_n$, the number of clusters K, and the number of iterations T:

1. Initialization: start with initial set of K clusters: $C_1^{(0)}, C_2^{(0)}, \ldots, C_K^{(0)}$.
2. For $t = 1, 2, 3, \ldots, T$:
   - For $i = 1, 2, \ldots, n$: Solve

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \big(h(X_i, X_i) + T_1 + T_2 \big).$$

   - For $k = 1, \ldots, K$, set

$$C_k^{(t)} = \{X_i \text{ such that } k_i = k\}.$$

The goal of this question is to implement this new algorithm from scratch using the same data generated in part (c). In your implementation, you will run the algorithm two times: first with the function:

$$h_1(x, x') = (1 + \langle x, x' \rangle),$$

and then with the function

$$h_2(x, x') = (1 + \langle x, x' \rangle)^2.$$

For your initialization (both times), use the provided `initial_clusters`, which can be loaded in by running `inital_clusters = np.load('init_clusters.npy')`. Run your code for at most 10 iterations, and provide two plots, one for h_1 and another for h_2. Discuss your results for the two functions. *What to submit: two plots, your discussion, a screen shot of your code and a copy of your code in your .py file.*

(g) The initializations of the algorithms above were chosen very specifically, both in part (d) and (f). Investigate different choices of intializations for your implemented algorithms. Do your results look similar, better or worse? Comment on the pros/cons of your algorithm relative to K-means, and more generally as a clustering algorithm. For full credit, you need to provide justification in the form of a rigorous mathematical argument and/or empirical demonstration. *What to submit: your commentary.*

Question 3. Kernel Power

Consider the following 2-dimensional data-set, where y denotes the class of each point.

index	x_1	x_2	y
1	1	0	-1
2	0	1	-1
3	0	-1	-1
4	-1	0	+1
5	0	2	+1
6	0	-2	+1
7	-2	0	+1

Throughout this question, you may use any desired packages to answer the questions.

(a) Use the transformation $x = (x_1, x_2) \mapsto (\phi_1(x), \phi_2(x))$ where $\phi_1(x) = 2x_2^2 - 4x_1 + 1$ and $\phi_2(x) = x_1^2 - 2x_2 - 3$. What is the equation of the best separating hyper-plane in the new feature space? Provide a plot with the data set and hyperplane clearly shown.

What to submit: a single plot, the equation of the separating hyperplane, a screen shot of your code, a copy of your code in your .py file for this question.

(b) You wish to fit a hard margin SVM using the `SVC` class in `sklearn`. However, the SVC class only fits soft margin SVMs. Explain how one may still effectively fit a hard margin SVM using the `SVC` class. *What to submit: some commentary.*

(c) Fit a hard margin linear SVM to the **transformed** data-set in part (a). What are the estimated values of $(\alpha_1, \ldots, \alpha_7)$. Based on this, which points are the support vectors? What error does your computed SVM achieve?

What to submit: the indices of your identified support vectors, the train error of your SVM, the computed α's (rounded to 3 d.p.), a screen shot of your code, a copy of your code in your .py file for this question.

(d) Consider now the kernel $k(x, z) = (2 + x^\top z)^2$. Run a hard-margin kernel SVM on the **original** (untransformed) data given in the table at the start of the question. What are the estimated values of $(\alpha_1, \ldots, \alpha_7)$. Based on this, which points are the support vectors? What error does your computed SVM achieve?

What to submit: the indices of your identified support vectors, the train error of your SVM, the computed α's (rounded to 3 d.p.), a screen shot of your code, a copy of your code in your .py file for this question.

(e) Provide a **detailed** argument explaining your results in parts (i), (ii) and (iii). Your argument should explain the similarities and differences in the answers found. In particular, is your answer in (iii) worse than in (ii)? Why? To get full marks, be as detailed as possible, and use mathematical arguments or extra plots if necessary.

What to submit: some commentary and/or plots. If you use any code here, provide a screen shot of your code, and a copy of your code in your .py file for this question.