Intro to Machine Learning (CS771A, Autumn 2023)
Homework 2
Due Date: Nov 15, 2023 (11:59pm)

Instructions:

- Only electronic submissions will be accepted. Your main PDF writeup must be typeset in LaTeX (please also refer to the "Additional Instructions" below).
- The PDF writeup containing your solution has to be submitted via Gradescope `https://www.gradescope.com/`.
- We have created your Gradescope account (you should have received the notification). Please use your IITK CC ID (not any other email ID) to login. Use the "Forgot Password" option to set your password.

Additional Instructions

- We have provided a LaTeX template file `hw2sol.tex` to help typeset your PDF writeup. There is also a style file `ml.sty` that contain shortcuts to many of the useful LaTeX commends for doing things such as boldfaced/calligraphic fonts for letters, various mathematical/greek symbols, etc., and others. Use of these shortcuts is recommended (but not necessary).
- Your answer to every question should begin on a new page. The provided template is designed to do this automatically. However, if it fails to do so, use the `\clearpage` option in LaTeX before starting the answer to a new question, to *enforce* this.
- While submitting your assignment on the Gradescope website, you will have to specify on which page(s) is question 1 answered, on which page(s) is question 2 answered etc. To do this properly, first ensure that the answer to each question starts on a different page.
- Be careful to flush all your floats (figures, tables) corresponding to question n before starting the answer to question $n+1$ otherwise, while grading, we might miss your important parts of your answers.
- Your solutions must appear in proper order in the PDF file i.e. solution to question n must be complete in the PDF file (including all plots, tables, proofs etc) before you present a solution to question $n+1$.

Problem 1 (20 marks)

(SGD for K-means Objective) Recall the K-means objective function: $\mathcal{L} = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}||x_n - \mu_k||^2$. As we have seen, the K- means algorithm minimizes this objective by taking a greedy iterative approach of assigning each point to its closest center (finding the z_{nk}'s) and updating the cluster means $\{\mu_k\}_{k=1}^{K}$. The standard K-means algorithm is a batch algorithm and uses all the data in every iteration. It however can be made online by taking a random example x_n at a time, and then (1) assigning x_n "greedily" to the "best" cluster, and (2) updating the cluster means using SGD on the objective $\mathcal{L}$. Assuming you have initialized $\{\mu_k\}_{k=1}^{K}$ randomly and are reading one data point x_n at a time,

- How would you solve step 1?
- What will be the SGD-based cluster mean update equations for step 2? Intuitively, why does the update equation make sense?
- Note that the SGD update requires a step size. For your derived SGD update, suggest a good choice of the step size (and mention why you think it is a good choice).

Problem 2 (10 marks)

(An Ideal Projection) Suppose we are given some labeled training data $\{\boldsymbol{x}_n, y_n\}$ with inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and labels $y_n \in \{-1, +1\}$. We want to project the inputs into *one* dimension using a projection direction given by a vector $\boldsymbol{w} \in \mathbb{R}^D$ such that, after the projection, the distance between the means of the inputs from the two classes becomes as large possible, and the inputs within each class become as close to each other as possible. Write down the objective/loss function that will achieve this and briefly justify the reason. You don't need to solve it.

Problem 3 (10 marks)

(Eigenchangers!) Suppose we wish to do PCA for an $N \times D$ matrix $\mathbf{X}$ and assume $D > N$. The traditional way to do PCA is to compute the eigenvectors of the covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$ (assuming centered data). Show that, if someone instead gives you an eigenvector $\boldsymbol{v} \in \mathbb{R}^N$ of the matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^\top$, you can use it to get an eigenvector $\boldsymbol{u} \in \mathbb{R}^D$ of $\mathbf{S}$. What is the advantage of this way of obtaining the eigenvectors of $\mathbf{S}$?

Problem 4 (30 marks)

(Latent Variable Models for Supervised Learning) Consider learning a regression model given training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$. Let us give a small twist to the standard probabilistic linear model for regression that we have seen in the class. In particular, we will be introducing a latent variable $\boldsymbol{z}_n$ with each training example $(\boldsymbol{x}_n, y_n)$, where $\boldsymbol{z}_n \in \{1, 2, \ldots, K\}$ denotes the cluster which $\boldsymbol{x}_n$ belongs to, and assuming a multinoulli prior on $\boldsymbol{z}_n$, i.e., $p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{z}_n|\pi_1, \ldots, \pi_K)$.

The model also has K weight vectors $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$. Given the cluster id $\boldsymbol{z}_n$, we will assume that $p(y_n|\boldsymbol{z}_n, \boldsymbol{x}_n, \mathbf{W}) = \mathcal{N}(y_n|\boldsymbol{w}_{\boldsymbol{z}_n}^\top\boldsymbol{x}_n, \beta^{-1})$. Note that although there are K weight vectors in the overall model, only one of them is being used to model y_n depending on the value of $\boldsymbol{z}_n$. Also note that the model for the responses y_n is still discriminative, since the inputs are not being modeled.

The latent variables are $\mathbf{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)$ and the global parameters are $\Theta = \{(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K), (\pi_1, \ldots, \pi_K)\}$.

(1) Give a brief explanation (max. 5 sentences) of what the above model is doing and why you might want to use it instead of the standard probabilistic linear model which uses a single weight vector $\boldsymbol{w}$ as models each response as $p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(y_n|\boldsymbol{w}^\top\boldsymbol{x}_n, \beta^{-1})$.

(2) Derive an ALT-OPT algorithm to estimate $\mathbf{Z}$ and (MLE of) Θ, and clearly write down each step's update equations. For $\mathbf{Z}$, you must give the update equation for each individual latent variable $z_n, n = 1, \ldots, N$.

Likewise, for Θ, you must give the update equation for each $\boldsymbol{w}_k, k = 1, \ldots, K$, and $\pi_k, k = 1, \ldots, K$. Also, what will be the update of each z_n if $\pi_k = 1/K, \forall k$. Give a brief intuitive explanation (max 1-2 sentences) as to what these updates are doing.

Problem 5 (30+30+10 = 70 marks)

The submitted code must be your own. You may discuss with your classmates but, ultimately, it must be written by you. The submitted code will be checked for plagiarism using software tools before grading begins.

(Programming Problem: Part 1) For this problem, your task will be to implement kernel ridge regression (see the solution to mid-sem exam question 6) and ridge regression with landmark based features (let's call the latter simply "landmark-ridge"), and train and test these two models on the provided dataset (already split into training and test). For both models, you have to use the RBF kernel $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\gamma||\boldsymbol{x}_n - \boldsymbol{x}_m||^2)$ with bandwidth parameter $\gamma = 0.1$. The training and test datasets are given in files `ridgetrain.txt` and `ridgetest.txt`. In each file, each line is an example, with first number being the input (a single feature) and the second number being the output.

You need to do the following

1. For kernel ridge regression, train the model with the regularization hyperparameter $\lambda = 0.1$ and use the learned model to predict the outputs for the test data. Compare the model's predictions with the true test outputs (given to you) by plotting on a graph. In particular, plot all the test inputs and the corresponding true outputs (x, y) on a 2D plot in blue color, and likewise all the test inputs and corresponding predicted outputs in red color. Repeat this exercise for $\lambda = 1, 10, 100$. What do you observe from the plots? For each case, also report the root-mean-squared-error (RMSE) on the test data, which is defined as square root of the mean of squared differences of true outputs and the predicted outputs.

2. For landmark-ridge, you first need to extract landmark based features using the RBF kernel and you will then use data with these features to train a linear ridge regression model. Again, keep the regularization hyperparameter fixed at $\lambda = 0.1$. Try $L = 2, 5, 20, 50, 100$ uniformly randomly chosen landmark points from the training set, train the model for each case and compute the predictions on the test data. Plot the results for each case just like you did in part 1 (but only for $\lambda = 0.1$). What do you observe from the plots? What's the RMSE in each case? What value of L seems good enough?

(Programming Problem: Part 2) For this problem, your task will be to implement the K-means clustering algorithm and try it on a provided toy (but interesting) dataset (`kmeans_data.txt`) consisting of points in two dimensions. The provided dataset also has 2 clusters (so you would use $K = 2$). However, the data is such that the standard K-means will NOT work well since the clusters are not spherical and not separable linearly (you can check this by plotting the data in 2D using a scatter plot). You will consider two ways to handle this issue.

1. **Using Hand-crafted Features:** Propose a feature transformation to the original data that will transform it such that K-means will be able to learn the correct clusters! Before proposing the transformation, plot the original data to see what transformation(s) might probably work. Apply your K-means implementation on this transformed version of the data to verify if your transformation works. Plot your obtained clustering results by showing points (in the original 2D space) in cluster 1 in red and points in cluster 2 in green.

2. **Using Kernels:** Although you can kernelize K-mean via the kernel K-means algorithm, you will try something else. You will use the landmark based approach to extract good features and your implementation of standard K-means on these features. The kernel that you will use for the landmark based approach is the RBF kernel $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\gamma||\boldsymbol{x}_n - \boldsymbol{x}_m||^2)$ with bandwidth parameter $\gamma = 0.1$. We are going

to try something (that may seem) crazy: We will pick $L = 1$ (yes, just ONE) landmark point randomly from the dataset and see whether $L = 1$ (which basically means just a single landmark based feature) is good enough to learn a correct clustering (at least for this data). It turns out that some landmark choices will be work and some won't work. Try 10 runs of the algorithm, each time with a different randomly chosen (single) landmark and check the obtained clustering. For each run, produce a plot as you did in part 1 and, on the same plot, also show the chosen landmark point in blue color. Justify why you get a correct clustering in some cases and a not-so-correct looking clustering in other cases.

Important Note: To avoid randomness in cluster mean initialization, we will choose the first two points in the dataset (the first two lines in the provided dataset) as the initial cluster centers.

(Programming Problem: Part 3) You are provided a subset of MNIST digits data (as a pickle file) with 10,000 examples from digits 0-9 (10 classes). Loading the pickle file will show `X` and `Y` fields contain the digit input features (784 dimensional) and labels (0-9), respectively, of the 10,000 examples.

Use PCA and tSNE to project the inputs to two dimensions (for PCA and tSNE, you may use existing library functions such as those from scikit-learn). For each method (PCA and tSNE), visualize the projected data in two dimensions and color each input using a color based on its class (so all inputs from the same class should use the same color). What difference do you observe between PCA and tSNE plots?

Deliverables: For each of the three parts, you should prepare separate Python notebooks, zip them together in a single zip file (named as yourrollnumber-problem-5.zip) and upload at the following Dropbox link: `https://www.dropbox.com/request/YKHU6d5RX4nSXDdNV6gd`. In the main PDF writeup, write about your observations for each of the three parts and include the plots in the PDF.