Intro to Machine Learning (CS771A, Autumn 2023)
Homework 1
Due Date: Sept 11, 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 `hw1sol.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)

(Another Simple Classifier) Consider a classification model where we are given training data $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ from K classes. Each input $\boldsymbol{x}_n \in \mathbb{R}^D$ and each class c is defined by two parameters, $\boldsymbol{w}_c \in \mathbb{R}^D$ and a $D \times D$ positive definite (PD) matrix $\mathbf{M}_c$, $c = 1, 2, \ldots, K$. Assume N_c denotes the number of training examples from class c. Suppose we estimate $\boldsymbol{w}_c$ and $\mathbf{M}_c$ by solving the following optimization problem

$$(\hat{\boldsymbol{w}}_c, \hat{\mathbf{M}}_c) = \arg\min_{\boldsymbol{w}_c, \mathbf{M}_c} \sum_{\boldsymbol{x}_n : y_n = c} \frac{1}{N_c} (\boldsymbol{x}_n - \boldsymbol{w}_c)^\top \mathbf{M}_c (\boldsymbol{x}_n - \mu_c) - \log |\mathbf{M}_c|$$

(note that, in the above objective, the $\log |\mathbf{M}_c|$ term ensures positive definiteness of $\mathbf{M}_c$ because the determinant of a PD matrix is always non-negative)

For the given objective/loss function, find the optimal values of $\boldsymbol{w}_c$ and $\mathbf{M}_c$. Also, what will this model reduce to as a special case when $\mathbf{M}_c$ is an identity matrix?

Problem 2 (10 marks)

(Consistent or Not?) An important notion for a classifier is that of *consistency*. A classification algorithm is said to be *consistent* if, whenever it has access to **infinite** amounts of training data, its error rate approaches the optimal error rate (a.k.a. *Bayes optimal*). Consider the noise-free setting (i.e., every training input is labeled correctly). Here, the Bayes optimal error rate is zero. Is the one-nearest-neighbor algorithm consistent in this setting? Briefly justify your answer in 100 words or less.

Problem 3 (10 marks)

(Decision Trees for Regression) When constructing Decision Trees for classification, we prefer to split on features that divide a node such that the set of labels at the children nodes is as "pure" as possible, i.e., we want each child node to consist of examples such that one label dominates the other label(s). Entropy/information-gain can quantify the purity in the classification setting. Suggest a good criteria to choose a feature to split on if we were doing *regression* instead of classification (so the labels are real-valued instead of discrete labels in classification)? Your criteria should somehow quantify the homogeneity/diversity of the set of *real-valued* labels of the examples at each node. You may define it using equations or state it in words (but be precise).

Problem 4 (20 marks)

(Linear Regression viewed as Nearest Neighbors) Show that, for the unregularized linear regression model, where the solution $\hat{\boldsymbol{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y}$, the prediction at a test input $\boldsymbol{x}_*$ can be written as a weighted sum of all the training responses, i.e.,

$$f(\boldsymbol{x}_*) = \sum_{n=1}^N w_n y_n$$

Give the expression for the weights w_n's in this case and briefly discuss (<50 words) in what way these weights are different from the weights in a *weighted version* of K nearest neighbors where each w_n typically is the inverse distance of $\boldsymbol{x}_*$ from the training input $\boldsymbol{x}_n$. **Note:** You do not need to give a very detailed expression for w_n (if it makes algebra messy) but you must give a precise meaning as to what w_n depends on and how it is different from the weights in the weighted K nearest neighbors.

Problem 5 (20 marks)

(Feature Masking as Regularization) Consider linear regression model by minimizing the squared loss function $\sum_{n=1}^N (y_n - \boldsymbol{w}^\top\boldsymbol{x}_n)^2$. Suppose we decide to mask out or "drop" each feature x_{nd} of each input $\boldsymbol{x}_n \in \mathbb{R}^D$, independently, with probability $1-p$ (equivalently, retaining the feature with probability p). Masking or dropping out basically means that we will set the feature x_{nd} to 0 with probability $1-p$. Essentially, it would be equivalent to replacing each input $\boldsymbol{x}_n$ by $\tilde{\boldsymbol{x}}_n = \boldsymbol{x}_n \circ \boldsymbol{m}_n$, where $\circ$ denotes elementwise product and $\boldsymbol{m}_n$ denotes the $D \times 1$ binary mask vector with $m_{nd} \sim \text{Bernoulli}(p)$ ($m_{nd} = 1$ means the feature x_{nd} was retained; $m_{nd} = 0$ means the feature x_{nd} was masked/zeroed).

Let us now define a new loss function using these masked inputs as follows: $\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n)^2$. Show that minimizing the *expected* value of this new loss function (where the expectation is used since the mask vectors $\boldsymbol{m}_n$ are random) is equivalent to minimizing a **regularized** loss function. Clearly write down the expression of this regularized loss function.

Problem 6 (40 marks)

(Programming Problem) Your task is to implement and test prototype based classification using the provided dataset ("Animals with Attributes" version 1 - AwA_v1). You may download the dataset using this link: `https://tinyurl.com/cs771-a23-hw1dat` (caution: the data is in a couple of hundred MBs; there is also a README file that contains the description of the data). In this dataset, each input represents an image of an animal and output is the class (what this animal is). The dataset has a total of 50 classes and the number of features for each input is 4096 (note: these features were extracted by a deep learning model and you don't need to perform any other preprocessing of features for this problem).

However, we are going to give a small twist to the basic prototype based classification problem. The training set provided to you has only examples from 40 of the classes. We will refer to these 40 classes as "seen classes" (have training data for these classes) and the remaining 10 classes as "unseen classes" (do not have training data for these classes). The test inputs will be only from these 10 unseen clases.

Recall that prototype based classification requires computing the mean of each class. While computing the means of the 40 seen classes is easy (since we have training data from these classes), what we actually need is the mean of the remaining 10 classes (since these are our test classes). How do we get these means?

Well, we clearly need some *additional information* about the classes in order to solve this problem (without that there is no hope of solving this problem). To this end, you are provided an 85-dimensional **class attribute vector** $\boldsymbol{a}_c \in \mathbb{R}^{85}$, for each class c (both seen as well as unseen classes). Each class attribute vector contains information about that class and consists of 85 binary-valued attributes representing the class (e.g., whether this animal has stripes). You may think of each $\boldsymbol{a}_c$ as a *class feature* vector which tells us what the class looks like.

Now consider two ways how these class attribute vectors can be used to obtain the means of unseen classes:

- **Method 1:** Model the mean $\mu_c \in \mathbb{R}^{4096}$ of each unseen class c (where $c = 41, \ldots, 40$) as a *convex* combination of the means $\mu_1, \ldots, \mu_{40}$ of the 40 seen classes (again note that $\mu_1, \ldots, \mu_{40}$ can be computed easily since we do have training data from these 40 classes)

$$\boldsymbol{\mu}_c = \sum_{k=1}^{40} s_{c,k}\boldsymbol{\mu}_k, \quad c = 41, \ldots, 50$$

  where $\boldsymbol{s}_c = [s_{c,1}, s_{c,2}, \ldots, s_{c,40}]$ is a vector of similarities of the unseen class c with each of the 40 seen classes. Here each similarity is defined as the inner product of the class attribute vectors of two classes, e.g., $s_{c,k} = \boldsymbol{a}_c^\top \boldsymbol{a}_k$ is the similarity between two clases c and k. **Note:** We will also *normalize* the vector $\boldsymbol{s}_c$ as $s_{c,k} = s_{c,k}/\sum_{\ell=1}^{40} s_{c,\ell}$ so that it sums to 1 (and thus can be used as weights in the convex combination defined above). This procedure will give us the means of 10 unseen classes and then you can apply the prototype based classifer to predict the labels of each test input.

- **Method 2:** Train a linear model that can predict the mean $\mu_c \in \mathbb{R}^{4096}$ of any class using its class attribute vector $\boldsymbol{a}_c \in \mathbb{R}^{85}$. We can train this linear model using $\{(\boldsymbol{a}_c, \mu_c)\}_{c=1}^{40}$ as our training data and then apply it to predict μ_c for each unseen class using its class attribute vector $\boldsymbol{a}_c$. Note that this can be posed as a multi-output regression problem $\mu_c = \mathbf{W}\boldsymbol{a}_c$ where $\boldsymbol{a}_c \in \mathbb{R}^{85}$ is the input, $\mu_c \in \mathbb{R}^{4096}$ is the vector-valued regression output, and $\mathbf{W}$ is 85×4096 matrix of weights that need to be learned. The solution to this multi-output regression problem is $\mathbf{W} = (\mathbf{A}_s^\top \mathbf{A}_s + \lambda \mathbf{I})^{-1}\mathbf{A}_s^\top \mathbf{M}_s$ where $\mathbf{A}_s$ is the 40×85 matrix containing the class attribute vectors of 40 seen classes, and $\mathbf{M}_s$ is the 40×4096 matrix containing the means of the 40 seen classes. Once you have $\mathbf{W}$, you can compute the mean μ_c of any unseen class as $\mathbf{W}\boldsymbol{a}_c$ where $\boldsymbol{a}_c$ is its class attribute vector. This procedure will give us the means of 10 unseen classes and then you can apply the prototype based classifer to predict the labels of each test input.

You task is to implement both methods. In your main write-up, report the test-set classification accuracy for each by comparing the respective model's predictions with the provided ground truth labels (accuracy is the percentage of test inputs with correctly predicted classes). For method 2, try $\lambda = 0.01, 0.1, 1, 10, 20, 50, 100$ and report the test accuracy for each value of λ. Which value of λ gives the best test set accuracy?

Note: In the dataset folder, I've also provided a brief pseudo-code (or rather a summary) of the procedure to follow for the implementation (since this is the very first assignment, I felt it would be helpful for you all).

Important: To predict the class of a test input, you should only compute its distances from the means of 10 unseen classes, not from the means of the seen classes (since the test data only consists of inputs from the unseen classes). The other setting where the test data can consist of both seen and unseen class inputs is also possible but we are not considering it here.

Deliverables: The code for each method, submitted as separate files. For method 1, name the file `convex.py`; likewise, for method 2, name the file `regress.py`. Your codes should be easily readable. Please use comments wherever necessary. Also include a README file to briefly describe how to run the code. All the code and README should be zipped together and submitted as a single file named `yourrollnumber.zip`. Please DO NOT submit the data provided.