
Instructions:

- The assignment is not difficult, but you will need to start early to finish it in time. I would urge you to read *all* the questions within the next 3-4 days. Start early!
- Queries about the assignment on or after Oct. 19th may not be answered. This will entirely depend on the availability and convenience of the TAs. Please contact them sooner if you have doubts.
- Instructions regarding the report and code are strictly enforced. In the event that your code does not work out of the box, you will be requested for a demo. However, during the demo, you are expected to be able to answer any questions about the theory and programming components.
- You can use any library you want (unless explicitly stated otherwise), but do not copy code. We will be running MOSS on your code. We already have statistics of potential defaulters from previous assignments. Plagiarism will be dealt with strictly.

1. [*Theory*: 2 points] Consider the logistic regression model with a target variable $y_i \in \{-1, 1\}$ for a given data $(\boldsymbol{x}_i, y_i)$ for $i = 1...N$ with $\boldsymbol{x}_i \in \mathbb{R}^d$. If we define $p(y_i = 1|f(\boldsymbol{x}_i)) = 1/(1 + e^{-f(\boldsymbol{x}_i)})$ where $f(\boldsymbol{x}_i)$ is given by $f(\boldsymbol{x}_i) = \boldsymbol{w}^\top \boldsymbol{x}_i + b$, show that the negative log likelihood, with the addition of a quadratic regularization term, takes the form given by (1) .

$$\sum_{i=1}^{N} E_{LR}(y_i, f(\boldsymbol{x}_i)) + \lambda||\boldsymbol{w}||^2 \tag{1}$$

where

$$E_{LR}(yf(\boldsymbol{x})) = \ln(1 + \exp(-yf(\boldsymbol{x})))$$

Read Section 7.1.2 from Bishop and comment on the logistic error function in contrast to the hinge error function typically used in SVMs.

2. [*Programming*: 8 points] This programming assignment requires you to apply SVMs (linear and kernelized) to the MNIST dataset and plot the Receiver Operator Characteristic (ROC) curves in each case.

Data: Download the MNIST Dataset for evaluation. Create a smaller subset by selecting 2000 samples per class from the training set. Similarly, create a test set of 500 samples per class from the testing set. **NEVER** use the samples in the testing subset for training or validation.

1. Linear SVM for binary classification
   - Choose samples from the training and testing subsets corresponding to the digits 3 and 8 for the binary classification problem. Use the samples only from the training set for training and validation.
   - Use the soft-margin linear SVM formulation and perform five-fold cross-validation with grid search for finding an appropriate regularization parameter, C . How would you find a reasonable choice for defining the grid?
   - With the chosen value of C, learn the soft-margin SVM with the training data.
   - Test the performance of your learned SVM on the test set for the binary problem. Plot the ROC curve for this SVM (See below for details about ROC curves).
2. Linear SVM for multi-class classification

- Use the one-versus-all multi-class linear SVM model.
- Use the complete training subset of 20000 samples and perform a five-fold cross-validation with grid search to find an appropriate regularization parameter, C.
- Learn a soft-margin SVM using C from the previous step.
- Test the performance of your learned SVM on the test subset of 5000 samples. Plot the ROC curve for this SVM (See below for details about ROC curves).

3. Radial Basis Function (RBF) Kernel + SVM for multi-class classification

   - Use the one-versus-all multi-class SVM model with RBF kernels.
   - Use the complete training subset of 20000 samples and perform five-fold cross validation with grid search for finding C and scale parameter γ of the RBF kernel. How would you find a reasonable choice for defining the grid for γ? One option is to vary γ between the tenth and 90th percentile of all pairwise distances between training data.
   - Using the parameters from the previous step, train the kernelized SVM for all the training data.
   - Test the performance of your learned SVM on the test subset of 5000 samples. Plot the ROC curve for this SVM (See below for details about ROC curves).

You are encouraged to use an implementation of SVMs from libSVM, as opposed to extensions in Matlab or scikit-learn. However, using the latter does not entitle a penalty. In either case, please take a look at libSVM's practical guide for suggestions to make SVMs work better in practice.

Quantitative Evaluation:
We will use the following metrics for quantitative comparison of classification outputs with the ground truth labels:

1. Classification Accuracy vs False Positive Rate with ROC curve. For more information on ROC based performance evaluation look at the Matlab's documentation and/or Scikit-learn's documentation. You may use the corresponding functions to generate the ROC and include the plot in your report. However, you should know what Classification Accuracy, True Positive Rate, False Positive Rate, False Negative Rates and other such classification error metrics are.

Report
Important: You need to submit a **single pdf** including all your results. Legible snapshots of your theory solutions should be added as pages in your report. Any other format (.jpg,.png,.doc/.docx) will not be accepted and a zero will be awarded to every question not there in the pdf report. If multiple pdf files are submitted, *only one* arbitrarily chosen pdf will be graded.

- For each part above, you need to briefly explain the steps performed.
- Plot the average cross-validation error plotted against the different values of C. In the kernelized SVM case, also show the cross-validation error *w.r.t.* C and γ both.
- Show the ROC curves individually in each of the three cases.
- Finally, show the ROC plot from parts 2 and 3 above on the same graph. Comment on which approach seems to be better.
- If you are doing the extra credit question below, in a separate figure plot the ROC curves of the three approaches (linear SVMs, kernelized SVMs, KPCA+kNN) together and comment on the performance. Can you compare these performances?

Important: Make sure *all* plots have properly labeled axes with clearly labeled legends, i.e., appropriate markers to tell which curve belongs to which experiment/data. Plots will only be accepted if they are clearly labeled.

Code

Submit a zipped folder named 'RollNo.zip' (not .rar, not .tar.gz). In the zip file you must have two directories, with the following structure

1. Folder 1: Models
   In this directory you will put your learned models. We will run your models on our test dataset.
2. Folder 2: Source
   In the directory you will put your source code.
3. file: Report
   Include your report as a single .pdf file in the same zipped folder.

Please adhere to the following naming convention We will assume that your model is the object returned by `svmtrain()` function in libSVM.
In the Models directory.

1. **Matlab**:
   You can use "`save LinearMod model_linear`" command to save the model where the file name is "`LinearMod`" and name of variable containing the model is "`model_linear`".
   First Part: Your model name should be "`model_linear`" and File name: "`LinearMod.mat`".
   Second Part: For multi-class classification you will be using one-against-all approach. So you will be generating 10 SVMs corresponding each digit.
   File Name: "multi1.mat" Name of variable containing model: "multi1"
   File Name: "multi2.mat" Name of variable containing model: "multi2"
   Similarly for the rest of the models.
   Third Part: RBF kernel with multi-class classification File Name: "Rbf1.mat" Model Name: "rbf1"
   File Name: "Rbf2.mat" Model Name: "rbf2"
   Similarly for the rest of the models.
2. **Python**:
   In python you can use utility functions:
   To store the model:
   `svm_save_model('heart_scale.model', m)`
   To read the model: `m = svm_load_model('heart_scale.model')`

   First Part: Your model name should be 'model_linear.model'
   Second Part: For multi-class classification you will be using one-against-all approach. So you will be generating 10 SVMs corresponding each digit.
   Model Name: "multi1.model"
   Model Name: "multi2.model"
   Similarly for the rest of the models.

   Third Part: RBF kernel with multi-class classification *Model Name*: "rbf1.model"
   Model Name: "rbf2.model"
   Similarly for the rest of the models.

In the Sources directory:
In addition to all your source code and dependencies (except libSVM, matlab toolboxes, or scikit-learn) please include a readme.txt that explains how to run your code.

3. [*Programming Extra Credit*: 3 points] You are required to implement kernelized PCA (KPCA) using an RBF Kernel with k-nearest neighbor (kNN) classification. Owing to the simplicity of the technique, **you are required to implement KPCA and k-nearest neighbor classification yourself**. You would need to follow the steps below:

1. Use 150 randomly selected samples per class from the training set of the MNIST dataset.
2. Create a test set of 100 samples per class from the test set of MNIST.
3. Construct the kernel matrix, which in this case will be 1200×1200, center it, and perform KPCA.
4. Project the validation data on to the PCs in the kernelized feature space, followed by kNN classification with $k = 3$.
5. Use a five-fold cross-validation scheme with grid search to estimate the γ parameter for an RBF kernel. Use the classification accuracy to select the best value of γ.
6. Perform KPCA + kNN classification on the 1000 test samples and report the classification accuracy.

As discussed in class, the kernel representation is only a change of basis of the feature space (canonical bases to bases derived from data points themselves). Therefore, we can perform Principal Component Analysis (PCA) in the kernel space. Please see the detailed derivations of kernel PCA in Max Welling's notes.

4. [*Theory Extra Credit*: 2 points] Consider the Lagrangian given by (2) for the regression support vector machine (See CB-Ch-7.1.4). By setting the derivatives of the Lagrangian with respect to $\boldsymbol{w}, b, \xi_i, \widehat{\xi}_i$ to zero and then back substituting to eliminate the corresponding variables, show that the dual Lagrangian is given by (3).

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b, \xi_i, ..\xi_N, \widehat{\xi}_i, ...\widehat{\xi}_N) =& C\sum_{i=1}^{N}(\xi_i + \widehat{\xi}_i) + \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{N}(\mu_i\xi_i + \widehat{\mu}_i\widehat{\xi}_i) - \\ & \sum_{i=1}^{N} a_i(\epsilon + \xi_i + f(x_i) - y_i) - \sum_{i=1}^{N}\widehat{a}_i(\epsilon + \widehat{\xi}_i - f(x_i) + y_i) \end{aligned} \tag{2}$$

Here, a_i, $\widehat{a}_i$, μ_i and $\widehat{\mu}_i$ are Lagrangian variables, $\xi \geq 0$ and $\widehat{\xi} \geq 0$ are slack variables, where $\xi_i \geq 0$ corresponds to a point for which $y_i > f(\boldsymbol{x}_i) + \epsilon$ and $\widehat{\xi}_i > 0$ corresponds to a point for which $y_i < f(\boldsymbol{x}_i) - \epsilon$

$$\widehat{\mathcal{L}}(\boldsymbol{a}, \widehat{\boldsymbol{a}}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{N}(a_i - \widehat{a}_i)(a_j - \widehat{a}_j)k(x_i, x_j) - \epsilon\sum_{i=1}^{N}(a_i + \widehat{a}_i) + \sum_{i=1}^{N}(a_i - \widehat{a}_i)y_i \tag{3}$$

Here, $f(x) = \boldsymbol{w}^\top\phi(\boldsymbol{x}) + b$ and kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}')$, where $\phi(\boldsymbol{x})$ denotes a fixed feature space transformation.