Programming Assignment #2

Question 5: Multiclass Classification

In this question, you will again be tasked with classification of two-dimensional points with four different classes. You will be provided training_samples.txt, training_labels.txt, and testing_samples.txt. These will be in the same format as a comma separated text file.

(a) Load the training data into your program. In two dimensions, plot the training data coloring each point by its class label. In your answer include a graphic showing this plot and provide a brief description of the patterns you see in the dataset.

(b) Use the formula for the slides to update a multidimensional linear classifier using the softmax function and cross entropy loss. You may initialize your weights vector as the all zeroes matrix. Choose a step-size parameter η and number of epochs T which seems to work well for the problem. Record the loss after each update of the w parameters. Plot the loss function from time 0 to time T. Provide a brief description of the loss curve, is it going down consistently? If not, is it possible there is an issue with your hyperparameters η and T?

(c) Finally, using your final weights matrix to compute the softmax scores of each test sample, make a prediction for each test sample, then plot the predictions of the testing set in two-dimensions (colored by predicted label.) Does it line up with the decision rules you expected from part (a)? If not, why?

Question 6: Neural Network Classification

In this question, you will be tasked with classifying the nonlinear XOR dataset using a neural network in the PyTorch package. We have provided a jupyter-notebook going through this implementation which we demoed during the discussion section. There are plenty of online resources explaining how to download and use jupyter-notebook but feel free to ask on Piazza if there are any lingering questions.

(a) Make sure you can replicate the results from the demo section:
- plotting the training and testing datasets
- plotting the predictions of the network
- plotting the loss curve over time (epochs)

(b) Make adjustments to the hyperparameters to see how stable/ unstable the neural network is, how does this affect the performance? Make sure you at least do the following:
- try at least three different learning rates
- try at least two architectures (change the hidden_size or change the number of layers)

Write a brief description of what you think happened to the network when you made these changes. Did the same learning rate work for the two different architectures or did you need to change it?

(bonus) consider changing the training size "N_trn", the batch size, the optimizer, and the total number of epochs.

Question 7: Kernel Classification

In this question, you will be tasked with classifying the nonlinear XOR dataset using a kernel machine/ support vector machine.

(a) Using the same data generated from the neural network XOR question above, compute the kernel matrix of the training dataset using the RBF/ Gaussian kernel from the slides using hyperparameter $\sigma^2 = 0.1$.

Gram/kernel matrix becomes:

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

Gaussian kernel or Radial basis function (RBF) kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}}$$

for some $\sigma > 0$.

(b) Now compute the inverse of the kernel matrix K^{-1} and compute $\alpha = K^{-1}y$

(c) Now compute the prediction on the test set by first computing the kernel matrix between the training set and the testing set. This matrix should be of size (N_trn x N_tst) and multiply it with the vector α (which is of size N_trn) to get the final test predictions of size N_tst.

$$\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

Plot the test predictions and compare them to the true XOR function. Does the SVM properly learn the true nonlinear distribution?