Mini-project 2
COMP 5/630 Spring 2021

Guidelines

Submission. Submit a *single pdf document* via Canvas that includes your solutions, figures, and printouts of code. For readability, you may attach the code printouts at the end of the solutions. Submissions may be 48 hours late with a 50% deduction. Submissions more than 48 hours after the deadline will be given zero.

Plagiarism. We might reuse problem set questions from previous years, covered by papers and webpages. We expect the students not to copy, refer to, or look at the solutions in preparing their answers. Since this is a graduate class, we expect students to want to learn and not google for answers.

Collaboration. The homework must be done individually, except where otherwise noted in the assignments. 'Individually' means each student must hand in their own answers, and each student must write their own code in the programming part of the assignment. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that you will be taking the responsibility to make sure you personally understand the solution to any work arising from such a collaboration.

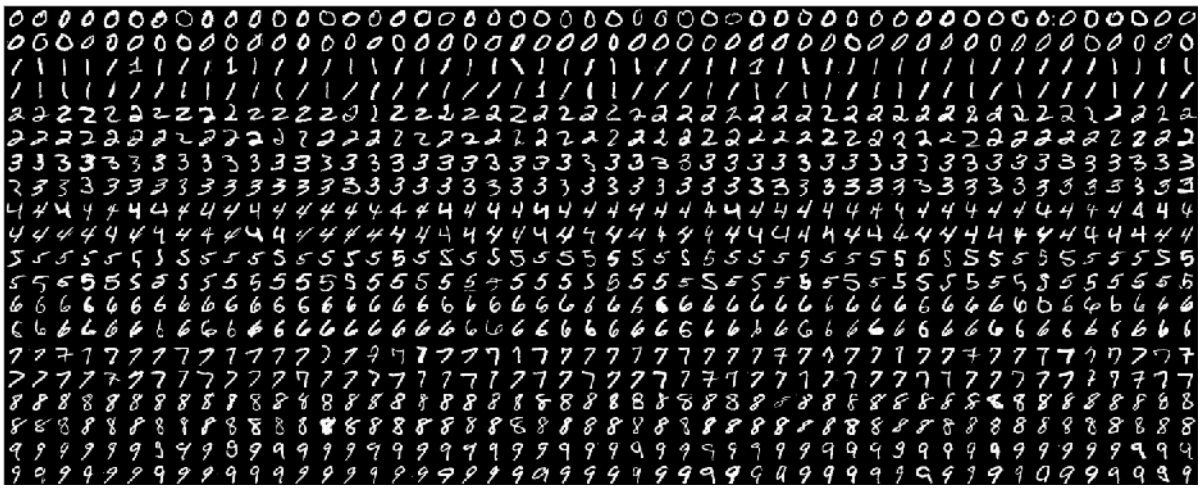
Using other programming languages. Python is recommended. You are free to use other languages such as Matlab or Java. We won't be able to answer or debug programming questions, as this is not a programming language course.

1 Data

The dataset for this project is the famous MNIST dataset, which contains 10 classes and 780 features (pixels).
https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#mnist
Download the mnist.scale.bz2 dataset. After download it, please **split the data to a 70:30 ratio for training/testing.**

mnist

- Source: [YL98a]
- Preprocessing: Feature values are stored by rows of each image
- # of classes: 10
- # of data: 60,000 / 10,000 (testing)
- # of features: 780 / 778 (testing)
- Files:
  - mnist.bz2
  - mnist.t.bz2 (testing)
  - mnist.scale.bz2 (scaled to [0,1] by dividing each feature by 255)
  - mnist.scale.t.bz2 (testing) (scaled to [0,1] by dividing each feature by 255)
  - mnist.mat (dense matlab format)
  - mnist.t.mat (testing, dense matlab format)



2 Basics

2.a Binary-class two-layer neural network

Now implement a two-layer neural network for binary classification with a hidden layer of H=100 units and K=2 output units, one for each class. Assume that each hidden unit is connected to all the inputs and a bias feature. You can use the ReLU and Sigmoid functions as the link/activation function.

2.b Multiclass logistic regression

This method is optional, NOT required

We can easily extend the binary logistic regression model (aka, linear model plus the sigmoid function) to handle multiclass classification. Let's assume we have K different classes, and posterior probability for class k is given by:

$$P(Y = k|X = x) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{i=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})}.$$

2.c Linear SVM

You need to implement the linear SVM algorithm using subgradient (Fig. 1 in the Pegasos paper, you don't need to read the whole paper! We also talked in class, but did not officially formulate the algorithm). This algorithm is originally designed for binary classification, but you can use the method we talked about in class to extend it to the multi-class classification tasks.

2.d Kernel Perceptron

In this part, you will extend the perceptron training and prediction to use polynomial kernels. A polynomial kernel of degree d is defined as:

$$K_{(poly)}^{d}(\mathbf{x}, \mathbf{z}) = (a + b\mathbf{x}^T \mathbf{z})^d$$

Here, $a \geq 0$, $b > 0$ and $d \in \{1, 2, \ldots, \infty\}$ are hyperparameters of the kernel.

3 Method

You need to implement all the methods mentioned below.

3.a Indirect Multiclass to binary reductions

Implement the kernel perceptron method yourself and use the meta-methods we talk about in class (for your information, the method is the Error-correcting output code (ECOC) discussed in MLaPP 16.6.2) to reduce the multi-class classification problem to binary classification problems.

3.b Direct Multiclass classification

Now tweak the two-layer neural network to work for multi-class problems. Try to combine the two basics (2.a &2.b) together to implement your own multi-class two-layer neural networks.

Next you can also use the decision tree algorithm you implement in your mini-project 1 for direction multiclass classification.

So now you have four methods to compare: (1) kernelized perceptron with meta-method ECOC, (2) linear SVM with meta-method ECOC, (3) multi-class two-layer neural networks, and (4) vanilla decision-tree approach.

4 Evaluation

For various parts of the homework you will be asked to report the accuracy and confusion matrix. The accuracy is simply the fraction of labels you got right. The confusion matrix C is a 10×10 Array where

$$C_{ij} = \sum_{k=1}^{N} [y_k = i] \cdot [ypred_k = j].$$

More information about the confusion matrix can be found in
https://en.wikipedia.org/wiki/Confusion_matrix.