Assignment 6: Neural Network & Image Classification & Audio Clustering

UVA CS 6316/4501 :
Machine Learning (Fall 2016)

Out: Nov. 17th, 2016
Due: Dec. 12 (Mon) midnight 11:55pm, 2016 @ Collab

a *The assignment should be submitted as source codes plus the report in PDF format through Collab. If you prefer hand-writing the writing part of answers, please convert them (e.g., by scanning) into PDF form. Malformed submissions will be penalized in grading.*

b *For questions and clarifications, please post on piazza. TA Weilin (wx4ed@virginia.edu) or Kamran (kk7nc@virginia.edu) will try to answer there.*

c *Policy on collaboration:*

Homework should be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, with the honor code, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework: Homework is worth full credit at the midnight on the due date. Each student has three extension days to be used at his or her own discretion throughout the entire course. Your grades would be discounted by 15% per day when you use these 3 late days. You could use the 3 days in whatever combination you like. For example, all 3 days on 1 assignment (for a maximum grade of 55%) or 1 each day over 3 assignments (for a maximum grade of 85% on each). After you've used all 3 days, you cannot get credit for anything turned in late.*

1 Question 1. Neural Network Playground (TA: Weilin)

Tensorflow Playground is an interactive tool for learning neural networks (more specifically, multi-layer-perceptron[1] networks). A customized version of Tensorflow Playground for HW6 is available at `http://www.cs.virginia.edu/yanjun/teach/2016f/HW6/`.

First, we will get familiar with the interface of Tensorflow Playground. Let's get to it!

You can choose from four different datasets on the left side of the page in the DATA panel: Circle, Exclusive Or, Gaussian, and Spiral.The actual data point locations are available on the right side under the OUTPUT label. Do not change the ratio of training to test data or click the REGENERATE button throughout this question. Batch size indicates how many samples are used in your mini-batch gradient descent. You may change this parameter if you want.

The neural network model is in the middle of DATA and OUTPUT. This model is a standard "feed-forward" neural network, where you can vary: (1) the input features (2) the number of hidden layers (3) the number of neurons at each layer. By default, it uses only the raw inputs X_1 and X_2 as features, and no hidden layers. You will need to change theses attributes later.

[1]A perceptron model typically uses a heaviside step function as its activation. Since we use a sigmoid (or other differentiable nonlinearity), our representation not exactly a perceptron model, but that is what we call it

Several hyper-parameters are tunable at the top of the page, such as the learning rate, the activation (non-linearity) function of each neuron, the regularization norm as well as the regularization rate.

There are three buttons at the top left for you to control the training of a neural network: Reset, Run/Pause and Step (which steps through each mini-batch of samples at a time).

1.1 Hand-crafted Feature Engineering

Now that we are familiar with the Playground, we will start to build models to classify samples.

First you are going to earn some experience in hand-crafted feature engineering with a simple perceptron model with no hidden layers, sigmoid activations, no regularization. In other words, don't change the model you're given by default when loading the page.

A perceptron (single-layer artificial neural network) with a sigmoid activation function is equivalent to logistic regression. As a linear model, it cannot fit some datasets like the Circle, Spiral, and Exclusive Or. To extend linear models to represent nonlinear functions of x, we can apply the linear model to a transformed input $\phi(x)$.

One option is to manually engineer ϕ. This can easily be done in the Playground since you are given 7 different features to choose from in the FEATURES panel.

Task: You are required to find the best perceptron models for the four datasets, **Circle, Exclusive Or, Gaussian and Spiral** by choosing different features. Try to select as few features as possible. For the best model of each dataset, you should report the selected features, iterations and test loss in a table. If you also change other hyper-parameters, *e.g.* the learning rate, you should include them in your report.

For each dataset (Circle, Exclusive Or, Gaussian and Spiral), please include a **web page screenshot** of the result in your report and explain why this configuration works.

(Note: Don't worry if you cannot find a good perceptron for the Spiral dataset. For the other three datasets, the test loss of a good model should be lower than 0.001.)

1.2 Regularization

Forget the best features you have found in last question. You should now select all the possible (*i.e.* all 7) features to test the regularization effect here.

You are required to work on the three datasets: **Circle, Exclusive Or and Gaussian**.

Task A: Try both L1 and L2 regularization with different (non-zero) regularization rates. In the report, you are required to compare the decision boundary and the test loss over the three models trained with similar number of iterations: no regularization, L1 regularized, L2 regularized.

For each dataset (Circle, Exclusive Or and Gaussian), please include a **web page screenshot** of the result in your report and explain why this configuration works.

Task B: We have learned that L1 regularization is good for feature selection. Take a look at the features with significantly higher weights. Are they the same as the ones you select in last question? Write down the results you observe in your report. (You can get the feature weights by moving the mouse pointer over the dash lines.)

1.3 Automated Feature Engineering with Neural Network

While we were able to find different parameters which were able to make good predictions, the previous two sections required a lot of hand-engineering and regularization tweaking :(We will now explore the power of a neural network's ability to *automatically* learn good features :) Let's try it out on two datasets: the **Circle** and **Exclusive Or**.

Here we should only select X_1 and X_2 as features (since we are trying to automatically learn all other features from the network). As we have previously seen, a simple perceptron is not going to learn the correct boundaries since both datasets are not linearly separable. However, a more complex neural network should be able to learn an approximation of the complex features that we have selected in the previous experiments.

You can click the + button in the middle to add some hidden layers for the model. There is a pair of + and - buttons at the top of each hidden layer for you to change the number of the hidden units. Note that each hidden unit, or neuron, is the same neuron from Lecture 17 (possibly varying the sigmoid activation function).

Task: Find a set of neural network model parameters which allow the model to find a boundary which correctly separates the testing samples. Report the test loss and iterations of the best model for each dataset. If you modify the other parameters (e.g. activation function), please report them too. Can it beat or approach the result of your hand-crafted feature engineering?

For each dataset (Circle, Exclusive Or), please include a **web page screenshot** of the result in your report and explain why the configuration would work.

1.4 Spiral Challenge (Extra credit)

Congratulations on your level up!

Task: Now try to find a model that achieves a test loss lower than 0.01 on the **Spiral** dataset. You‘re free to use other features in the input layer besides X_1 and X_2, but a simpler model architecture is preferred. Report the **input features, network architecture, hyper parameters, iterations and test loss** in a table. For simplicity, please represent your network architecture by the hidden layers **a-b-c-...**, where a, b, c are number of hidden units of each layer respectively.

Please include a **web page screenshot** of the result in your report and explain why this configuration works.

You are now a neural network expert (on the Playground)!

2 Question 2. Image Classification (TA: Weilin)

An online tutorial might be helpful to you for finishing this assignment,
`http://blog.yhathq.com/posts/image-classification-in-Python.html`
For this assingment, you will be comparing different ML techniques for recognizing hand-written digits. The data set is available as zip.train.gz and zip.test.gz. Each example in the train and test data is a the number in question followed by 256 grayscale values representing a 16×16 image. Values have already been normalized. You can read a full description of the data set in the zip.info.txt file available in collab.

For each question, you will apply a different model to the dataset using scikit-learn. The purpose of this assignment is to further develop your intuition in regards to machine learning methods and to give you experience with dimensionality reduction. Visualization of some samples are provided in the subdir "image_digits" in the attached data ZIP file.

2.1 Decision Tree

Using sklearn.tree.DecisionTreeClassifier, apply the decision tree classifier to the dataset and include the following within the written report. Make sure to fill in the method stub for number_recognition.decision_tree(train, test).

- Choose three parameters to customize for the decision tree, and run your model using those parameters.

- What is the testing error using those parameters?
- What should each of those parameters do for the model?

2.2 K Nearest Neighbors

Use scikit-learn's sklearn.neighbors.KNeighborsClassifier for these questions and fill in the respective function in the provided file.

- Select five different values for k and report the test and training error.
- Based on this information, what value of k would you choose for this data set and why?
- What explanation do you have for KNN's performance on this data set? How do you think distance weighting would impact performance, why?

2.3 Support Vector Machine

You should be well acquainted with the SVM classifier after HW3. We will use the same library (sklearn.svm.SVC) for this homework problem.

- How do the rbf, linar, and polynomial kernels compare on this data set? Please include the parameters you use for each one.
- Theoretically, do you think a SVM is well suited to this problem, why or why not?

2.4 PCA

Principal component analysis allows us to reduce the dimensions of our data while retaining as much variance as possible, thus retaining information to separate our data points. Scikit learn provides multiple ways to perform dimension reduciton with PCA. For this problem, sklearn.decomposition.RandomizedPCA is recommended.

- Using your best K value for KNN, select three different values of PCs you choose for PCA and present your test error results. Why would you want to use PCA with KNN?
- Do the same for your implementation of the SVM. What do you notice about its impact on SVM's performance and how do you explain it?

2.5 Submission

Please submit your source code and PDF report containing your written answers via collab.

2.6 Grading

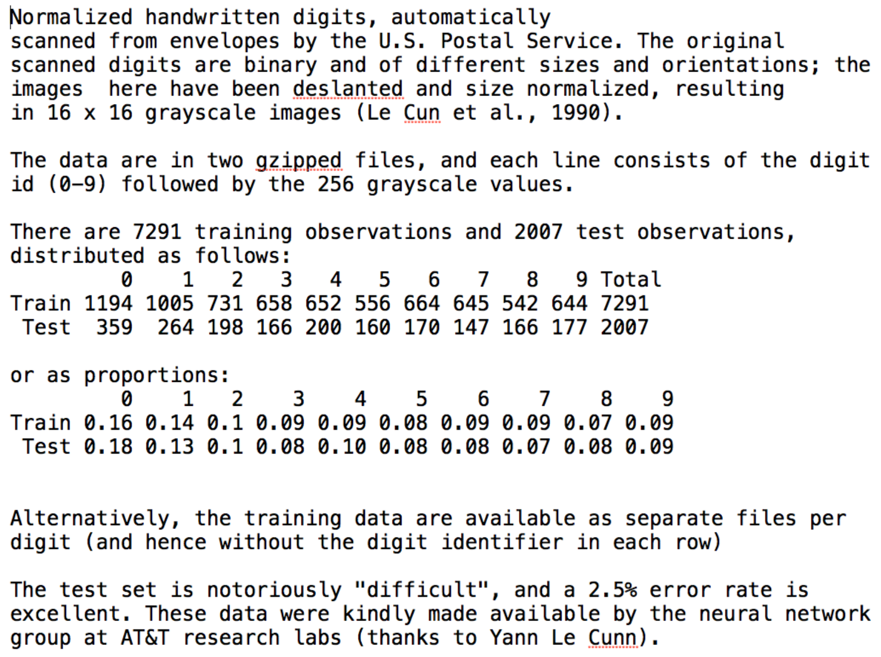
In collab, you will find starter code in number_recognition.py. This file will be run from the command line. For example:

```
python dtree path/to/training_data path/to/testing_data
```

Feel free to add any methods or classes to your implementation, so long as your preserve the given command line behavior for your submission.

Congratulations ! You have implemented a state-of-the-art machine-learning toolset for an important OCR image labeling task !

Figure 1: Description of this data set (zip.info.txt file)



```
Normalized handwritten digits, automatically
scanned from envelopes by the U.S. Postal Service. The original
scanned digits are binary and of different sizes and orientations; the
images  here have been deslanted and size normalized, resulting
in 16 x 16 grayscale images (Le Cun et al., 1990).

The data are in two gzipped files, and each line consists of the digit
id (0-9) followed by the 256 grayscale values.

There are 7291 training observations and 2007 test observations,
distributed as follows:
         0    1   2   3   4   5   6   7   8   9 Total
Train 1194 1005 731 658 652 556 664 645 542 644 7291
 Test  359  264 198 166 200 160 170 147 166 177 2007

or as proportions:
         0    1   2    3    4    5    6    7    8    9
Train 0.16 0.14 0.1 0.09 0.09 0.08 0.09 0.09 0.07 0.09
 Test 0.18 0.13 0.1 0.08 0.10 0.08 0.08 0.07 0.08 0.09


Alternatively, the training data are available as separate files per
digit (and hence without the digit identifier in each row)

The test set is notoriously "difficult", and a 2.5% error rate is
excellent. These data were kindly made available by the neural network
group at AT&T research labs (thanks to Yann Le Cunn).
```

You can do a lot of cool things by extending the above toolset, for instance, build an app recognizing handwritten digits using "node.js", e.g.
`http://blog.yhathq.com/posts/digit-recognition-with-node-and-python.html`

3 Unsupervised Learning with Clustering (TA: Kamran)

In this programming assignment, you are required to implement clustering algorithm: K-means Clustering and Gaussian Mixture Model. A ZIP file has been provided ("data_sets_clustering.zip") that includes two different datasets. Please follow all instructions for submitting source code.

You are required to submit a source-code file "clustering.py" containing the necessary functions for training and evaluations. The maximum number of iterations to be performed for both algorithms is 1000.

DO NOT use scikit-learn package in this problem and please implement from scratch.

3.1 Data description

We have provided two different datasets for clustering tasks.

- **Dataset 1** : The first dataset consists of height and weight data for average people and baseball players. First column contains human height (inches) and second column has human weight (lbs), while third column has true labels of samples that will be used only for evaluations.

- **Dataset 2** : The second dataset is for a speech versus music classification task. This dataset has been preprocessed and first 13 columns contain 13 features extracted from audio files. Last column has true labels of samples that will be used only for evaluations.

3.2 load data

- (Q1) You are required to code the following function for loading datasets:
  X = loadData(fileDj)

3.3 K-means Clustering

- (Q2) Next, code the following function to implement k-means clustering:
  labels = kmeans(X, k, maxIter)
  Here X is the input data matrix, k is the number of clusters and maxIter is the maximum number of the iterations selected by you (max value =1000).
- (Q3) Implement k-means clustering for **Dataset 1**(use first two columns in the file as input) and use scatter() function in the matplotlib package to visualize the result. The two clusters must be in different colors.
- (Q4) Implement k knee-finding method for **Dataset 1** and k = {1,2,...,6} to select value of k (number of clusters) and plot graph for k versus objective function value (e.g. Slide 99, Lecture 20).
- (Q5) Now, code the following function to calculate the purity metric for the evaluation of results:
  purityMetric = purity(labels, trueLabels)
  Use this function to evaluate the results of (Q3)

3.4 Gaussian Mixture Model

- In this section, assume k = 2.
- (Q6) You are required to code the following function in order to implement Gaussian Mixture Model:
  labels = gmmCluster(X, k, covType, maxIter)
  Here X is the input data matrix, k is the number of clusters and maxIter is the maximum number of the iterations selected by you (max value =1000). covType is a parameter for the types of covariance matrices. It can be set to two values – "diag" and "full". "diag" means that the covariance matrices are diagonal matrices and "full" means that the covariance matrices are full matrices. In this problem, we assume that the covariance matrices are same.
- Note: (1) We assume that the two covariance matrices of two clusters are same. In another word, when the type covariance matrices are full matrices, we can estimate the same covariance matrix by

  $$\widehat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^T(x_i - \bar{x})$$

  .
  Here n is the number of samples and x_i is the i-th sample. If you need more information, please refer to : `https://en.wikipedia.org/wiki/Sample_mean_and_covariance`. For calculation, you can simply use the function: **numpy.cov()**. When the type of covariance matrices are set as diagonal matrices, (i.e., we only consider the variance of variables), you can simply use the diagonal of full covariance matrices and set other entries as 0. After you have estimated these covariance matrix, you just consider them as the known covariance matrices Σ (since we assume different clusters share are same known covariance). Then our formulation is the same as the pages 43-46 of L21.
- Note: (2) Furthermore, the homework problems are about multivariate Gaussian distribution, which is slightly different from the Gaussian equations used in E-step of L21 slide-45.

  $$p(x = x_i|\mu = \mu_j) = \frac{1}{\sqrt{(2\pi)^p det(\Sigma)}} exp(-\frac{1}{2}(x_i - \mu_j)^T\Sigma^{-1}(x_i - \mu_j))$$

  . Here p is the number of features. In this assignment, therefore, the equation for the E-step should be:

  $$[z_{ij}] = \frac{\frac{1}{\sqrt{(2\pi)^p det(\Sigma)}} exp(-\frac{1}{2}(x_i - \mu_j)^T\Sigma^{-1}(x_i - \mu_j))p(\mu = \mu_j)}{\sum_{s=1}^{k}\frac{1}{\sqrt{(2\pi)^p det(\Sigma)}} exp(-\frac{1}{2}(x_i - \mu_s)^T\Sigma^{-1}(x_i - \mu_s))p(\mu = \mu_s)} \quad (1)$$

  The equation in the M-step is the same as slide 46 in L21.

- (Q7) Implement the Gaussian Mixture Model for **Dataset 1**(use first two columns in the file as input) and use scatter() function in the matplotlib package to visualize the result. The two clusters must be in different colors. You are required to implement two types of GMM, for covType = "diag" and covType = "full".

- (Q8) Implement the Gaussian Mixture Model for **Dataset 2**(use first 13 columns in the file as input) and use scatter() function in the matplotlib package to visualize the result.

  Note: In this problem, only use the first two features as input to scatter() function. The two clusters must be in different colors. Set covType = "diag".

- (Q9) Use the purity function coded in (Q5) to evaluate the results of (Q7) and (Q8).

3.5 How will your code be checked?

We will run the following command: "python clustering.py DatasetDirectoryFullPath" and your code should print the following results:

- ONE of the scatter plots (either (Q3),(Q7) or (Q8))
- k knee-finding plot in (Q4)
- ALL purityMetric values for results obtained in (Q3),(Q7) and (Q8)

3.6 Submission

Please submit your source code as "clustering.py" and PDF report containing your written answers via collab. In the report, you should include the following contents:

- ALL scatter plots generated in (Q3),(Q7) and (Q8)
- k knee-finding plot in (Q4)
- ALL purityMetric values for results obtained in (Q3),(Q7) and (Q8)