CS6220 BDS 2018 Fall

Homework Assignment 3

(Programming Category)

Student Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Session: cs6220-A

You are given 4 types of programming problems. You only need to choose one of them as your second homework. For the problem with multiple options, you are only required to choose one option under the problem.

Feel free to choose any of your favorite programming languages, such as Java, C, Perl, Python, and so forth.

Post Date: Monday of Week 7

Due Date: midnight on Friday of Week 9

**Problem 1. Hand-on Experience with unsupervised auto-encoder**

You are asked to get hand-on experience with unsupervised auto-encoder, which is typically included in most of the deep learning frameworks, such as Tensorflow, PyTorch, OpenCV, etc.

Here are some code related readings.

1. https://rare-technologies.com/deep-learning-with-word2vec-and-gensim/
2. <https://rare-technologies.com/word2vec-in-python-part-two-optimizing/>
3. <http://www.deeplearningbook.org/contents/autoencoders.html>
4. <http://deeplearning.net/tutorial/dA.html>
5. <http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/>
6. <http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders>
7. <https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663>

**Option 1.1** Training an unsupervised auto-encoder model for Image classification or image denoising.

1. You can use the auto-encoder algorithms from one of your favorite deep learning software frameworks, such as Scikit-Learn, Keras, TensorFlow (google), Caffe, Torch/PyTorch, OpenCV (Intel), CNTK (Microsoft).

[https://scikit-learn.org/](https://scikit-learn.org/stable/)

<https://www.tensorflow.org>.

<https://caffe.berkeleyvision.org>

<http://torch.ch> or https://pytorch.org

<https://docs.opencv.org/master/index.html>

<https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/>

<https://docs.microsoft.com/en-us/cognitive-toolkit/setup-cntk-on-your-machine>

<https://keras.io>

2. The image dataset should be 50 images or larger and should have at least two classes of images, such dogs and cats. You can compose your dataset from public domain such as ImageNet or Kaggle.

3. Deliverable.

(a) Provide the dataset of 50 images that you have created for HW3 (option 1.1) and the URL of the public domain where you obtained your original dataset. Split your dataset for training and testing (8:2).

(b) Provide the description of your auto-encoder algorithm and the software package name and URL, from which you get your algorithm for training.

(c) Provide 2-3 screen shots of your training process. Elaborate on your training process and how did you improve the original (default) training process if any.

(d) Provide your test results for your trained model. You are recommended to use 10% of your training data for testing. For example, you have 40 images for training, then use additional 10 images out of your 50 images total for testing. Report the average performance, the best performance and the worst performance result. Elaborate your test results.

**Option 1.2**

The option is to use the auto-encoder to train an unsupervised self-identity model by tuning the latent space representation (such as #dimensions). For example, train at least two different auto-encoders on the same dataset with different random initial weight matrix or with different latent representation (code) space dimensions, say 10, 30, and compare the two models in terms of training and testing performance. You may recommend the best way to train a self-identity model.

A short tutorial is provided at <http://deeplearning.net/tutorial/dA.html> .

https://cs.stanford.edu/people/karpathy/convnetjs/

(1)You are asked to choose an image dataset of at least 50 images, say 25 dogs and 25 cats from ImageNet dataset or choose 50 images of two different classes of your favorite, 25 images each.

(2) Select the auto-encoder latent space representation with two different dimensions as your code. For example, 2 or 4 dimensions.

(3) For each latent space representation, train the self-identity model on your dataset with 80% training data and 20% testing data (5 images for each of the two classes)

(4) Report the accuracy and training time, test time for your two auto-encoder models.

(5) Discuss your comparison analysis on the results.

Deliverable:

1. provide URL of your open source code packages
2. Example screen shots of your execution process/environments
3. Input and output of your training model
4. Input and output of your prediction model (test cases)
5. Measurement tables as described in (1)~(4)
6. Your comparison analysis which can be text or table, which serves as your report on your experience and analysis.

**Option 1.3**

This 3rd option is to use ConvNetJS, a Javascript library for training deep neural networks in your browser.

Hint: read the short tutorial at https://cs.stanford.edu/people/karpathy/convnetjs/index.html

For example, you can use ConvNetJS to train a denoising auto-encoder. One example for training on MNIST is provided at <https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html>.

Feel free to choose other tasks instead of auto-encoder, such as image painting, visualization of 2D classification, and so forth.

The goal of this programming option is to learn by first-hand experience on how to train a deep neural network model and how to provide visual interpretation of the trained model in terms of its performance in accuracy and time.

Deliverable:

1. provide URL of your open source code packages
2. Example screen shots of your execution process/environments
3. Input and output of your training model(s)
4. Input and output of your prediction model (test cases)
5. Measurement tables for comparison and
6. Elaborate your comparison results.

**Problem 2. Hand-on Experience with deep learning using Word2Vec**

Training an unsupervised word2vec model for text retrieval. You can use the word2vec algorithms from one of your favorite deep learning software frameworks.

Easiest way to use Word2Vec is via the Gensim libarary for Python (tends to be slowish, even though it tries to use C optimizations like Cython, NumPy)

<https://radimrehurek.com/gensim/models/word2vec.html>

Original word2vec C code by Google

<https://code.google.com/archive/p/word2vec/>

* + includes the models and pre-trained embeddings
  + Pre-trained is good, because training takes a lot of data

Word Embedding Visualization   
<http://ronxin.github.io/wevi/>

You are asked to choose a text dataset with over 1000 unique words. You can compose your dataset from public domain such as Kaggle or use the dataset that is provided by the word2vec package you use.

**Deliverable.**

(a) Provide the dataset that you have created for HW3 (option 1.2) and the URL of the public domain where you obtained your original dataset.

(b) Provide the description of your word2vec algorithm and the software package name and URL, from which you get your algorithm for training.

(c) Provide 2-3 screen shots of your training process. Elaborate on your training process and how did you improve the default training process if any. Hint: you can vary the latent representation (code) space size from 10 to 30. If you have 10,000 words or more, then your latent code space of 300 might be reasonable.

(d) Provide your test results for your trained model. You are recommended to use at least 10 words for testing. Report the test result in terms of the best, the average and the worst performance. Elaborate your test results.

**Problem 3: Hand-on Experience with** **Unsupervised (Clustering) Ensemble**

Your task for this assignment is to select 3~5 clustering algorithms, such as different implementations of k-means, or different clustering algorithms (k-means, SVM, etc.), and run all of them on a chosen dataset and then use a clustering ensemble method, such as majority voting, or bagging technique, to show how much clustering ensemble can improve the learning accuracy. Concretely,

1. Select a dataset from the UCI repository (https://archive.ics.uci.edu/ml/datasets.php) or use a dataset of your own choice.
2. Determine how you will measure the quality of the clusters produced. Some reference on clustering evaluation can be found at <http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html>.
3. Select 3-5 algorithms (e.g., different implementations of K-means, SVM with different kernels) from a ML library, such as scikit-learn at https://scikit-learn.org/stable/ or <https://pypi.org/project/scikit-learn/0.21.1/>. You may also find them from

* Scikit-Learn (http://scikit-learn.org/stable/),
* Weka (<http://www.cs.waikato.ac.nz/ml/weka/>),
* Mahout (<http://mahout.apache.org/users/classification/>),
* R (<http://www.rdatamining.com>).

Discuss how you choose your unsupervised ensemble learning strategy and why.

1. Run each of them on the same dataset and compare their results using your quality metrics. Note that by varying the initial points, you can obtain different implementations of a k-means algorithm.
2. Design a clustering ensemble algorithm, such as majority voting based bagging strategy or boosting method, to show whether and how much clustering ensemble can improve the learning accuracy. NOTE: you should make comparison on each of the 3~5 algorithms and their ensemble for comparison under different metrics.
3. Discuss how you choose your ensemble learning strategy and why.
4. Write a brief report to:

* Describe the dataset and your quality metrics.
* Describe your experiment setup such as how you preprocessed the data (if any), how you chose the parameters for the selected algorithms, and why.
* Present the experiment results for all six methods (4~5 individual clustering and 1 clustering ensemble) in a tabular or chart format for easy comparison.
* Discuss the insights and conclusions from your experiments.  For example, do different clustering methods make a difference in terms of quality or performance for the particular dataset you selected?  Why does clustering ensemble can improve the learning quality?

7. Deliverable.

* One tar or zip file that contains your source files (if available), the executable, a readme file explaining how to compile/run your program.
* The input dataset file or URL
* The output file for the test dataset screen shots of your execution process.
* Runtime statistics in excel plots or tabular format.
* Visual display of your ensemble learning and tuning parameters and the parameter settings in the first two iterations of the parameter tuning. Also discuss how your algorithm sets its convergence condition, such as error function and threshold.
* Report in ppt/word/pdf.

**Hint:** If you have a huge dataset that cannot fit into the main memory, then it is also interesting for you to use random sampling techniques to generate 3~5 sets of samples and use the same clustering algorithms (1~2) on all 3~5 different sample sets and then ensemble the results. Thus, you can replace the first three steps by using data ensembles and still carry out step 4~7 for your HW3.

**Problem 3. Hand-on Experience with** **Supervised** (**Classification) Ensemble**

Your task for this assignment is to choose random forest with 5 randomly generated decision trees or to select 4~5 classification algorithms, and run all of them on a chosen dataset and then use a classification ensemble method, such as majority voting, to show how much an ensemble method can improve the learning accuracy.

1. Select a dataset from the UCI repository (https://archive.ics.uci.edu/ml/datasets.php) or use a dataset of your own choice.
2. Determine how you will measure the quality of the classification results produced.
3. Choose either random forest and generate at least 5 different random decision trees with good diversity. Or Choose 3~5 different implementations of one classifier, such as the C4.5 decision tree classifier or Naïve Bayesian classifier or SVM or any other classifier that you are familiar with. You can find them from Select 3-5 algorithms (e.g., different implementations of K-means, SVM with different kernels) from a ML library, such as
   * + scikit-learn at https://scikit-learn.org/stable/ or <https://pypi.org/project/scikit-learn/0.21.1/>,
     + Weka (<http://www.cs.waikato.ac.nz/ml/weka/>),
     + Mahout (<http://mahout.apache.org/users/classification/>) or
     + R (<http://www.rdatamining.com>).
4. Discuss how you choose your ensemble learning strategy and why.
5. Evaluate the individual classifiers using the chosen dataset from UCI repository. One example is the [mushroom dataset](http://archive.ics.uci.edu/ml/datasets/Mushroom) from UCI repository. The training dataset contains 7423 records and the test dataset 701 records.  The first attribute is the class of each record and the rest 21 attributes are categorical attributes.
6. Write a brief report that include the following: Present and discuss the results of your experiments on the chosen dataset with each of the chosen individual classifiers and the classification ensemble; and discuss the experiences and lessons you have learned from the experimentation.
7. Deliverable:

* One tar or zip file that contains your source files (if available), the executable, a readme file explaining how to compile/run your program.
* The output file for the test dataset screen shots of your execution process.
* Runtime statistics in excel plots or tabular format.
* Visual display of your ensemble learning and tuning parameters and the parameter settings in the first two iterations of the parameter tuning. Also discuss how your algorithm sets its convergence condition, such as error function and threshold.
* Report in pdf/word/ppt.

**Hint:** If you have a huge dataset that cannot fit into the main memory, then it is also interesting for you to use random sampling techniques to generate 3~5 sets of samples and use the same classification algorithms (1~2) on all 3~5 different sample sets and then ensemble the results. Thus, you can replace the first three steps by using data-driven ensembles and still carry out the steps 4~7 for your HW3.