COGS 260, Spring 2018: Assignment 2

April 16, 2018

Due: April 29, 2018 11:59 PM PDT
Late policy: Every 10% of the total received points will be deducted for every extra day past due.

Instructions

1. You are required to use MNIST dataset for all the questions in this assignment [1].
2. Reference materials and some useful codes can be found at the end of the document.
3. Submit the ipython notebooks you used for this project in pdf format.
4. Write a report including: a) abstract, b) method, c) experiment, d) discussion, and e) references. You can follow leading conferences like CVPR (`http://cvpr2016.thecvf.com/submission/main_conference/author_guidelines`), NIPS (`https://papers.nips.cc/`), or ICML (`http://icml.cc/2016/?page_id=151`). All of those formats are readily available through `www.sharelatex.com`. For the homework assignments, the report can be brief and you can try to summarize your experiments, results, and main findings in a concise way.
5. Zip and upload related code and ipython notebooks as part of your submission on TritonEd in order for the TA to reproduce your result.
6. You are supposed to provide brief quantitative details and analysis of experiments whenever asked to report the details of the experiment. Please DO NOT write long paragraphs. You are encouraged to use plots and images to explain your results. Please report the random seed you used for your experiments. You can encouraged to print out your network structure in the ipython notebook if you choose to use Keras. There is no need to include that graph in your final latex report.

In this assignment you are going to compare the performance of different methods of image recognition on MNIST dataset. Divide the dataset into training and test set and use the same training and test set for all parts of the assignment. You may (cleverly) select a subset of the dataset to overcome computational challenges (if any).

1 Convolutional Neural Network [25]

For this question, choose a deep learning library (Keras, Tensorflow or Torch, etc) [2, 3] that you can prototype your network structures easily and efficiently. MNIST is a standard tutorial dataset for almost all deep learning libraries [4, 5], try to take a look and use the off-the-shelf data preprocessing pipelines instead of dealing with that yourself.

Build a convolutional neural network, train it and plot the training and test loss (not error) with the number of iterations. Also report the network architecture used and the classification accuracy on the test set for the experiment. You may play with different network architectures (LeNet, AlexNet, VGGNet, ResNet etc.), different hyper-parameters, and different optimizers like Stochastic Gradient Descent, Adaptive Gradient, RMSprop and/or Nesterov's Accelerated Gradient etc.

A template of LeNet using Keras has been provided. You are free to build on top of this code. If you choose to build your own network on top of the template provided, please report on at least one other CNN architecture.

2 1-Nearest Neighbor [25]

Using the raw pixel values (28×28) as feature vector, use 1-Nearest Neighbor to train a model and report its performance on the test set and the confusion matrix. Also report other details of the experiments like the distance functions used, prototype selection (if you have used), etc.

3 Support Vector Machines [25]

Again using the same feature vector, use SVM to train a model and report its performance on the test set and the confusion matrix. Report other relevant details of the experiments. You can use stochastic gradient descent based SVM in python, which can be found in scikit learn package [6].

4 Spatial Pyramid Matching [25]

Use traditional visual recognition pipeline (handcrafted SIFT local descriptor, bag-of-words codebook building and spatial pyramid matching [7], and classifier training) and report the performance on the test set and the confusion matrix. Report the details of the experiments. A link to the python code can be found under references [8]. Report your configuration details.

5 Extra Credit: Deep Belief Nets [5]

This section is optional but will fetch you extra credits if you do it. Unlike the above methods which are discriminative models, DBNs (Deep Belief

Nets) are generative models. The code for training a DBN is available in here (http://www.cs.toronto.edu/˜hinton/ MatlabForSciencePaper.html) but you are free to use codes from other sources or write your own code. Train a DBN and project the activations of the last layer in two-dimensional space using t-SNE or Multidimensonal Scaling (use matlab/python library for t-SNE and MDS) and clearly label different digits in different colors. You should observe data points with the same class label clustered together. Report your results and other details of experiments like network architecture, training procedure, etc.

References

[1] The MNIST Database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[2] Deep Learning Software Links. [Online]. Available: http://deeplearning.net/software_links/

[3] Deep Belief Networks Code. [Online]. Available: http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html

[4] Tensorflow MNIST Tutorial. [Online]. Available: https://www.tensorflow.org/get_started/mnist/pros

[5] Torch MNIST Demo. [Online]. Available: https://github.com/torch/demos/tree/master/train-a-digit-classifier

[6] SDG SVM. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier

[7] Spatial Pyramid Matching. [Online]. Available: http://slazebni.cs.illinois.edu/slides/ima_poster.pdf

[8] Spatial Pyramid Matching, Python Code. [Online]. Available: https://github.com/wihoho/Image-Recognition