PROJECT SPECIFICATION
IMAGE CLASSIFICATION

CMU 10-601: Machine Learning (Fall 2015)
http://www.cs.cmu.edu/~10601b/
OUT: Oct. 13, 2015
Project Proposal due: Oct. 22, 2015, 10:30 AM
Midway Report due: Nov. 24, 2015, 10:30 AM
Final Report and code due: Dec. 17, 2015, 10:30 AM

1 Project Description

1.1 Overview

In this project, you will apply the machine learning techniques learned in this course to a real world dataset. Up until now, you have focused on applying a specific algorithm to carefully preprocessed toy datasets. This project will require that you make decisions such as which classifier to use, which features to use, how to train the classifier, etc. You will find that real world datasets are more difficult to work with, and that careful tuning is required to get optimal performance.

1.2 Dataset

In this project you will work with a labeled dataset of images. The data for the project can be downloaded at http://www.cs.toronto.edu/~kriz/cifar.html. The website completely describes the composition of the dataset, how it's divided into training and test sets, links to download the dataset for use with MATLAB and Python, and the layout of the data in the files.

YOUR TASK is to write a classifier which can correctly predict the label of a given image. To receive full marks, it will not be sufficient to naively implement one of the classification algorithms from class; you will need to implement some more sophisticated methods of feature generation, classification, or both. For instance, you might experiment with various techniques to combat overfitting, or you might use a dimensionality reduction technique to generate better features.

1.3 Grading and Deliverables

- **Proposal [5 pts; due 10/22]**

  This is a half-page document that should briefly outline your plan for solving the problem. Things to consider: How will you convert the images to features? Which classifiers will you use (list at least 3 classifiers which you'll try, potentially including techniques to enhance the performance of those classifiers)? How will you train them? This document should also state who is in your group.

- **Midway Report [20 pts; due 11/24]**

  This is a 3- to 4-page document that should describe your progress on solving the problem. Consider the midway report as a partially complete final report. The midway report should therefore follow the same format as the final report, but not all sections need to be complete.

  Note that by the midway report, we expect you to have successfully submitted a working classifier to Autolab that achieves better-than-baseline classification accuracy. This classifier can be a simple implementation of one of the algorithms from class, but you should explain in the report what enhancements or more sophisticated methods you intend to try.

- **Final Report [40 pts; due 12/17]**

This is a 8-page document that describes how you solved the problem. It should contain detailed discussion about which methods you tried, results on their comparative performance, and a discussion about which methods worked best and why. It should follow the following simple outline: Introduction, Methods, Results, Conclusion, and a short paragraph on which team member contributed to which part of the project.

- **Classification Accuracy [35 pts]**

  In addition to the written deliverables, you will submit your classifier to Autolab, which will use it to classify a hidden test set. Your classification accuracy on this hidden test set will contribute to your final grade.

- **Competition [BONUS 10 pts]**

  To make things a little more exciting, this project is a competition. Your classification accuracy will be displayed on the class leaderboard using the handle you selected when you first logged in to Autolab. Students who are on top of the leaderboard at the start of each lecture may receive prizes, and the top 5 groups at the conclusion of the competition will receive bonus marks.

Please use NIPS document templates for your midterm and final reports.

1.4 Autolab Instructions

The project can be divided roughly into two parts: (1) Converting the images into features and training a model on those features, and (2) using this learned model to classify new data. We have split these tasks into two separate files:

1.4.1 train.m

This will be a MATLAB/Octave program to (a) extract features from the images, an (b) train a model on the extracted features. Your **train.m** should implement a function

```
[Model] = train(X, Y)
```

where `X` contains the training data, `Y` contains the training labels corresponding to `X`, and `Model` is a struct with all the parameters that define the model (you may design the struct to contain any parameters that are useful to your model). The dimensions of `X` are $N_{train} \times D$ and dimensions of `Y` are $N_{train} \times 1$. where N_{train} is the number of data points in the training data and D is the dimension of each point (number of features).

1.4.2 classify.m

This will be a MATLAB/Octave program to apply the model on test data. **classify.m** should implement a function

```
[Y] = classify(Model, X)
```

where `X` is the test data, `Model` is the model learnt during training, and `Y` contains the predicted labels of the data points in `X`. The dimensions of `X` are $N_{test} \times D$ and dimensions of `Y` are $N_{test} \times 1$. where N_{test} is the number of data points in the test data and D is the dimension of each point (number of features).

Your submission may contain other files implementing helper functions (such as functions that help you extract features).

You may use built-in MATLAB/Octave functions to extract features from the images, but NOT to do the actual model learning and classification, we expect you to implement the machine learning algorithm by yourself.

2 Techniques

We have learned (and will learn) a lot of both supervised and unsupervised machine learning techniques in this course. For this project, you are allowed to use any techniques taught in the classes. Also, in order to achieve a successful performance on our task (i.e., image classification), it is also very important which *feature* you use in training/testing. This may require some understanding of work in the field of computer vision. Some suggested techniques and popular image features that you may want to review/investigate are listed in the below.

1. A few classification techniques that you may want to investigate:
   - Logistic regression
   - Neural network
   - Decision tree
   - Support vector machine
   - K-nearest neighbor
2. A few techniques for generating/engineering features that you may want to investigate:
   - Raw intensity
   - Histogram of oriented gradients (HOG)
   - Scale invariant feature transform (SIFT)
   - Bag-of-words
   - Unsupervised learning algorithms (such as clustering)
   - Dimensionality reduction
3. A few other techniques that may turn out to be useful:
   - Cross-validation
   - Regularization

Some of the previous work that have achieved the state-of-the-art performances are listed in the reference ([2, 7, 6, 5, 4, 1, 3]), and there are many more! Please feel free to explore more out in the world. The classification performance can heavily depend on which feature is used, however, remember that our purpose for this project is more on making use of efficient machine learning algorithms learned throughout the course, not on making use of fancy image features.

References

[1] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[4] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[5] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[6] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[7] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer, 2010.