CS 429/529: Introduction to Machine Learning
Martin Cenek
Spring 2025 Due: NLT 23:59 3/12/2025

Homework 4:

For the previous exercise, you built a perceptron to classify the aurora images. Now, let's see how SVM would deal with this data, and if we can increase the classification accuracy of the resulting SVM using the kernel tricks from the class. Luckily, the SVM is a classifier that will differentiate between two classes of images using the 765 features that were extracted from each image and labeled as 1 for aurora images versus all other vector instances that represent the images without aurora (these feature vectors are labeled as -1).

Part 1:

1. Use the ML library from http://www.cs.cornell.edu/people/tj/svm_light/. Download, install and configure the SVM (works on Linux, Mac, Windows).

   The SVM engine takes a well formatted feature vector, so alter the data-set so you have two sets that are correctly labeled and split into training and validation sets. Note that some SVM packages do this for you. You might also want to check if the feature values must be re-scaled to 0-1 or test it if it makes any difference in the SVM performance.
   <line> .=. <target> <feature>:<value> <feature>:<value> ... <feature>:<value> # <extra stuff>
   <target> .=. +1 | -1 | 0 | <float>
   <feature> .=. <integer> | "qid"
   <value> .=. <float>
   <info> .=. <string>

   Example: -1 1:0.43 3:0.12 9284:0.2

2. Build a SVM to classify the aurora images, and save the resulting SVM model that you will use in step 3 below to measure the SVM performance. In your write-up report the SVM model details by annotating any useful information from the model (plain text file).

3. Evaluate the accuracy of your SVM by running all validation instances of the feature vectors on the SVM. Report the classification accuracy of the SVM.

4. Test the effect of one kernel on the SVM classification accuracy. You will have to re-run steps 2-3, so you will need to re-train the SVM with the selected kernel and re-test the SVM on your validation instances and report the accuracy.

Part 2:

Analyze the original extracted feature vectors from the images using PCA. This will tell us out of the 765 features, which features really matter (colloquially spoken). The PCA will return the variance ratio for each of the features (explained_variance_ratio_). Use the transformed and sorted features in the non-ascending order of variance ratio to train and evaluate SVM. Report the findings, how did your performance change as a function of conditionality reduction. Test the relationship between how many components after PCA transformation you keep and the SVM accuracy.

I had good success with Python's scikit learn decomposition library to do the PCA analysis: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html Feel free to use any

other PCA analysis toolkits to perform the analysis.

Graduate Students:
In addition to the PCA decomposition on the feature vector, perform the PCA analysis on the instances. In other words, reduce the set of training instances by only taking the instances that capture the most variance in the data. Give detail synopsis of your analysis in the write-up.

Part 3.
Repeat the part 1 of this assignment, but only use the first n features after the PCA transform. Report how much data you eliminated and what the SVM classification accuracy trade-off was. Did your SVM loose or gain accuracy only using the n features extracted from your images? You might consider making a table that records the number of features kept (and retained data variance) and the resulting accuracy.

What to submit:
On the moodle, submit an archive file (.zip or similar) with your (1) code base, (2) readme file with the instructions on the architecture, the compilation instructions and the usage (how and what the interface needs as its inputs) and (3) one to two page write-up in .pdf format of your findings. Please be as complete and thorough as you can be, use plots and tables summarize your findings.