Final Project Proposal

Dogs vs. Cats classification problem

Data:

Data is available at https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data

The training data folder contains 25,000 images of dogs and cats. Each image in this folder has the label as part of the filename. The test folder contains 12,500 images, named according to a numeric id. For each image in the test set, we need to predict if the image is a dog (1 = dog, -1 = cat).

Task:

Build a dog vs cat classifier. A dog vs cat classifier takes input as an image of a dog or cat, and the output will be a statement about whether the input is a picture of a cat or a picture of a dog. This is a supervised learning task, because every picture has a label either dog or cat. Compare the performance of the classifiers that are built by several different algorithms and choose the best performance classifier to implement.

Metrics:

$$Misclassification\ rate = \frac{1}{n}\Sigma\, I(y_i \neq y_i')$$

where

- n is the number of images in the data subset
- y_i' is the predicted label of the image, 1 for dog, -1 for cat
- y_i is the true label of the image
- $I(y_i \neq y_i')$ is a function that if $y_i \neq y_i'$ is true, then the function equal to 1, equal to 0 otherwise

A smaller misclassification rate is better in general.

Methodology:

1. Use Tensor flow to modify the format of input images and load pre-trained neural network to figure out the weights on the second-to-last layer.
2. Learn classifiers by RBF kernel SVM, Random Forest and AdaBoost by decision stumps
3. Use k fold Cross Validation and Grid Search to find the optimal parameters for the algorithms

Experiments:

We have the parameters for each algorithm:
RBF kernel SVM: margin parameter γ and bandwidth σ
Random Forest: number m of axes in R_d
AdaBoost: Number M of weak learners

To evaluate the performance of the algorithms/techniques,
firstly, evaluate the performance of the parameters of the algorithms: plot the performance metrics for different parameter values on both training set and validation set to check if the parameter value chosen is optimal. (The optimal value should be achieved when the log loss on both training set and validation set is fairly small and stable, notice that the training error is not necessary to be minimal, overfitting may occur);
secondly, evaluate the performance of the algorithms: compare the performance of different algorithms by plotting the performance metrics for the various algorithms with their optimal parameter values on testing data. The algorithm with lower testing error will have better performance.