Northeastern University
EECE – Pattern Recognition and Machine Learning
April 2018, Emily Dutile, Tristan Sweeney

Project Proposal: Classification in High-Resolution Brain Scans

1 Goal

Use machine learning for foreground-background classification in high-resolution brain scans.

2 The Data

The dataset[1] is composed of csv files with the intensity of pixels in a 3D image. The last value is the label for the whole image. On each line there should be 21*21*7+1 values.

3 Prediction Program

Using MapReduce or Spark, we'll implement a parallel prediction program that uses the previously generated, persisted model to predict for each record in the unlabeled data whether the center pixel is foreground (1) or background (0). Stated differently, this program should replace the ? label in the unlabeled data by either 0 or 1, based on the models prediction.

4 Models

Random Forest, SVM, and k-NN. We will compare how to achieve the best possible accuracy by exploring the parameter combinations.

While uniform random partitioning into training and validation data often works well, there are applications where it is not sufficient. This dataset is one of those tricky cases. Assume we have two labeled records with centers (x, y, z) and (x+1, y+1, z+1), respectively. Uniform sampling might assign (x, y, z) to training, and (x+1, y+1, z+1) to the validation data. Clearly, the two neighborhoods and the labels of these two pixels are highly correlated. Hence a model that overfits to the training data will also do very well on validation pixel (x+1, y+1, z+1). To prevent overly optimistic validation accuracy caused by such correlations, we have to be more careful when partitioning the labeled data. A safe procedure for ensuring independent training and validation data is to partition by image. For example, training data come from images 1, 2, 3, and 6; while 4 is used for validation. Alternatively, you can also partition each image into two separate blocks, using one block for training data and the other for validation data.

Bagging trains each model in an ensemble using bootstrap sampling. All these bootstrap samples have to be taken from the training data. This is the right approach when training data is limited, but in this application, we can produce a training record for each image pixel (minus those too close to the boundary or with unknown labels), then produce even more through rotation and mirroring. Instead of bootstrap samples, we may consider producing a new training set for each model in the ensemble but this may or may not be difficult to achieve in Spark MLlib.

5 Evaluation and Testing

Following the standard way of training and testing a classification model, we will partition the labeled data into 3 separate sets: training data, validation data, and test data. This will be done randomly, such that most data goes into training (typically 60-80 percent), and less into the other two (typically 20-10 percet for each). The following steps will be repeated until we are satisfied with the model accuracy: set the model parameters to some values,

[1] `https://drive.google.com/drive/u/0/folders/1EJBgJFmp-FQf2czw9LGImo0hE020v0oo`

train the model on the training data using this setting, and then evaluate the model accuracy on the validation data.

We can use accuracy (ACC) for the final evaluation. In the final evaluation test set there are about 0.0057 percent foreground and 99.9943 percent background pixels.

The output file will be a single file with a single column and as many rows as there are test points. We'll replace all ? labels with either 1 or 0 and remove all columns except for the label column. All predictions will be stored in plain text, i.e., exactly as strings 1 or 0 (without the quotation marks).