CMap: DPeak Challenge

John Gardner (gardn999)

Introduction:

The purpose of this challenge is to identify the peaks in the fluorescence intensity (FI) distribution of two genes mixed together at a 2:1 ratio. For the ideal case, where two peaks are clearly present at the expected ratios, identifying their positions should be rather easy. However, this is quite often not the case. Situations such as the two peaks heavily overlapping, not being present at the expected ratios or not matching the UNI "truth" values can occur. Some examples of this are shown in the FI distribution plots at the end. A machine learning algorithm is useful for predicting these difficult cases. Since both speed and accuracy are important, a solution using random forest regression was chosen. It is a simple and fast machine learning algorithm. The main difficulty in achieving the best results is identifying the most effective set of features from the provided data to train with.

About me:

I am from the United States. My highest academic degree is a PhD In Physics from the University of Kansas. In addition to the prize money, the possibility that my solution could be useful for some important research was motivating. I found the challenge was fairly easy to get started with. The problem statement was clearly laid out and a basic solution resulting in a reasonable estimate for well behaved cases was straightforward. There were, however, plenty of challenges to overcome in trying to figure out how to most accurately predict the non-ideal cases.

Solution Overview:

The first main function of my solution is creating a model file by training a random forest algorithm with the provided data and the UNI "true" peak positions. The second is using this model file to make predictions on the FI distribution peak positions for other data sets. The majority of my time was spent determining the best features to use for training. An important benefit of my approach is that it runs very quickly. Not only is speed important to the final aggregated score, but it also makes it easy to quickly test the effect of adding new candidate values to the feature set.

Solution Details:

My solution is written in Java and composed of seven files. Program execution begins in Main.java. It includes methods for training, testing, making plots and accepting command-line arguments. The runCommandLine method is intended for the end user and accepts arguments for either training on data to produce the model file, "rfAll.gz", or making predictions using this model file. For training, this method requires the training set directories and UNI "truth" files. The arguments are: <DPK input directory>, <DPK truth file>, <LITMUS input directory> and <LITMUS truth file>. For making predictions, arguments are: <input directory>, <output directory> and <plate name>. See the text file, Readme.md, for complete running instructions. The file, RFRegressor.java, is the general purpose random forest algorithm used to produce the training model. An RFRegressor object is created for predicting both the low peak and high peak.

The remaining five files define the main types of objects representing the data. These are Set, Experiment, Barcode, Pair and Gene.

The Pair object contains the FI distribution values of a pair of genes for a specific Barcode and Experiment. It also calculates quantities used in feature selection and has a method for returning an array of those selected features. The four calculated quantities are: mean, median, lowPeak and highPeak. LowPeak and highPeak are simple estimates of the peak positions which are fairly accurate if the FI distribution is well behaved.

The Experiment object contains a map to all Pairs in the experiment and averaged values of the calculated quantities of the Pairs. The Barcode object contains averaged quantities for all Pairs with the same barcode. The Set object contains a map of all Barcodes and Experiments in the set and averaged quantities for all Pairs in the set. Peak predictions for all Pairs occur in the calcPeaks method of the Experiment object. The calcPeaks method in the Set object calls this method for each Experiment in the set. These peak predictions are organized into a map of Gene objects so that they can be easily output in order by gene id to the solution file, <plate name>. Each Gene object contains a map for all experiments of experiment id to predicted peak.

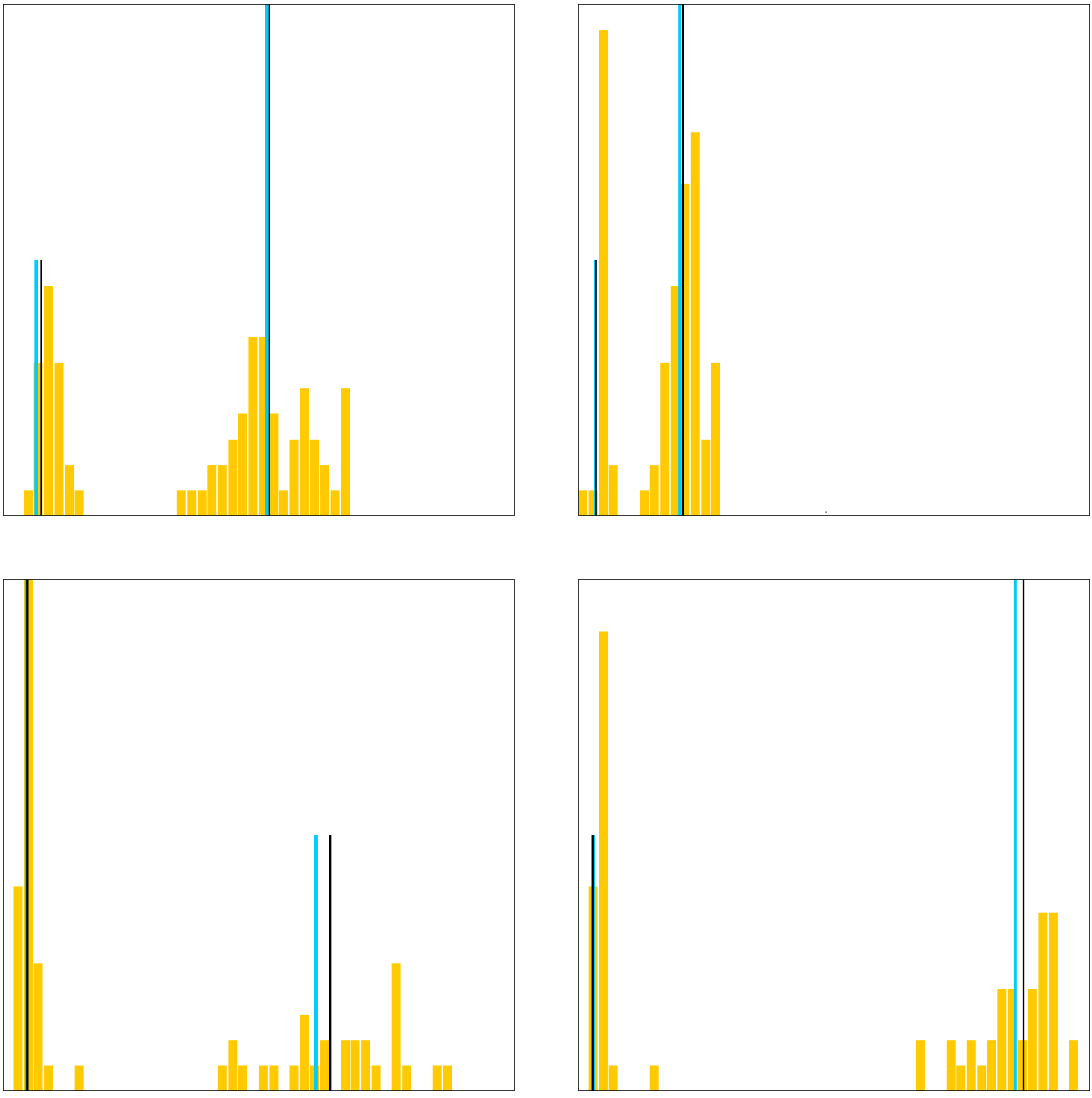
Feature Selection and Training:

There are a total of 60 features used in the random forest solution. The creation and selection of these features was the most time consuming part of the process. Many Pair only feature combinations were tried, but in the end just using the FI values directly worked best. Since the number of FI values varies, the actual values are stretched or compressed to 50. The 51rst feature is the total number of FI values in the Pair. The remaining 9 features involve various relationships of the averaged quantities in the Barcode, Experiment and Set for which the Pair belongs.

An offline scorer is provided for estimating the quality of the choice of features. For this process, the two data sets provided are trained separately to produce a model file for each. Then each model file is used to make predictions on the other set. This process was carried out in the method, runRFSplit, in Main.java. This provides a good lower bound estimate for the score. In general, training on both data sets should produce slightly better results due to the increased statistics. An upper bound on the score can be estimated by make predictions using the combined model, “rfAll.gz”. This is of course biased high due to training and predicting on the same data set. It is a good check to make anyway in order to make sure the final model file produces reasonable results.

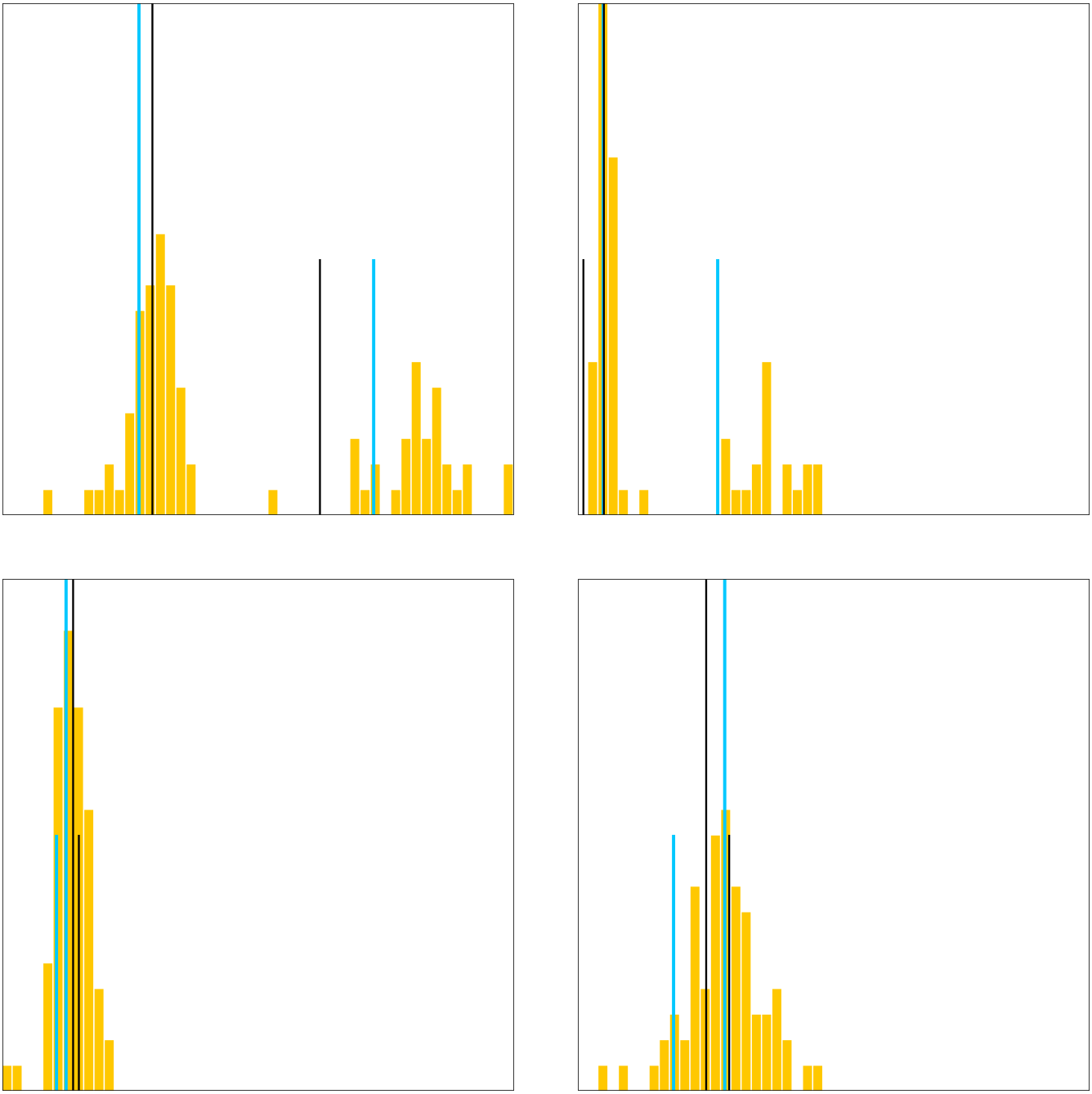
Potential Improvements:

Increasing the number of trees used in the random forest (Main.java line 10) should produce a minor accuracy improvement, but with a speed reduction. Training with more data sets should improve results. This is not just due to increased statistics, but also it would add more potential to develop features sensitive to set wide systematic bias. Also, I feel that there is still room for improvement of the feature set selection in general.



Well matched FI Distributions:

UNI “truth” peaks are black, predicted peaks are blue
low peaks are half height, high peaks are full height



Problematic FI Distributions:

UNI “truth” peaks are black, predicted peaks are blue
low peaks are half height, high peaks are full height