Or Samimi Golan 305641045

Shajaf Gonen 204067268

Unless specified otherwise in an alternative, we generated our classifiers in the following form:

1. SNP-wise. Meaning, that we generated a classifier for each missing SNP in the test set (300 classifiers in total in each alternative). The classifier of each SNP is the one that achieved the best success rate for the SNP amongst all alternatives(see section 2).
2. Each sample in the train (test) set, is the 100 SNPs before and after the SNP to be learned (200 feature in the feature vector in total). The matching label of a sample, is of course the value of the SNP. (Known in the train set = {0, 1, 2}, unknown in the test set = {-1}). These details are generated out of 'extracted\_train' and 'extracted\_test'
3. The success rate for each SNP is calculated using 10-fold validation, with a 0-1 loss function.
4. The total success rate is the average of all SNPs success rate.

**Section 1 – Same classifier for each SNP**

At first , each of us generated a classifier, and tested it.

Alternative 1 - SVM – one vs one using libsvm

Libsvm handles mutli-class classification, using one-vs-one technique (learned in class).

By running: svmtrain, using the following options.

Success rates

Linear SVM

C = 1: 69.47%

C = 0.1: 72.62%

C = 0.01: 70.69%%

Polynomial SVM

C = 1, degree = 4: 73.25%

C = 0.1, degree = 10: 72.37%

C = 1, degree = 10: 72.30%

C = 100, degree = 10: 72.30%

Radial basis SVM

C = 1, gamma = 1/features\_number (default value)= 67.51%

C = 100000, gamma = 1/features\_number (default value) = 73.69%

Results: as seen the best total success rate achieved with Gaussian kernel – 73.69%.

Alternative 2 – Multiclass Adaboost

We modeled the missing SNP value as dependent on some of the neighbor SNPs. For each missing SNP, we want to find the most influencing neighbor SNPs and in what way are they influencing.

We created classifiers by the following way – for each neighbor SNP, we created 27 classifiers, each one is the function of influence the current neighbor on the missing SNP. For example – classifier 0-0-1 for neighbor 103 states that if neighbor 103 is 0, then missing SNP is 0, if 1 then 0, and if 2 then 1. There are in total 27 options for such function.

There are 300 X 27 classifiers and 600 examples for the adaboost input. The adaboost output are the the chosen classifiers and their coefficients(alphas). Notice that this case is special because there are 3 classes, instead of 2 in normal case. Therefore we need different α for updating the examples. This is because now if our error rate is for example 50%, then it is better than random but α would be 0.5.

The chosen α is according to the SAMME algorithm (Stagewise Additive Modeling using a Multi-class Exponential loss function):

where K is the number of classes, in this case – 3.

After the train, the predicted class is:

Results: The best total results were 73.2% success rate with 10 iterations, and taking only the 60 main neighbors SNPs.

Alternative 3 – K-NN

We implemented a K-NN algorithm.

The metric we used: the distance between samples is the percentage of different neighbor SNPs(Hamming distance).

Results: the best results achieved were with 10-NN algorithm that looks only on the 10 main neighbor SNPs – 74.3%.

**Section 2 – Different classifier types for different SNPs**

We understood that for each SNP a different learner might suit. This is because we saw that for SNP 1 for example the svm got the best result, but for SNP 2 it was adaboost.

Therefore we decided to combine our learners in the following way: for each SNP we chose the learner with the best results of the 10-fold cross validation. The prediction for the missing SNP was according to this learner's classifier.

Results: the total success rate for the combination of learners is 76.1%.

Results Summary:

|  |  |
| --- | --- |
| **Algorithm** | **Total success rate(%)** |
| SVM | 73.7 |
| Adaboost | 73.2 |
| K-NN | 74.3 |
| Combined learners | 76.1 |

**Section 3 – Script details:**

Script Files:

* go.m – runs the program.
* parse\_input.m – prepare the data for the next stages.
* k\_fold\_cross\_validation.m – performance scrore implementation.
* train\_classifier\_for\_single\_snp.m – trains the given learner.
  + For each learner there is a suitable train script.
* predict\_lables\_for\_single\_snp.m – predicts the missing SNP for all test data according to given learner method.
  + For each learner there is a suitable prediction script.
* get\_single\_snp\_best\_classifier.m – finds the best classifier amongst all learners as described in section 2.

Script Usage:

Run:

[predicted\_labels\_mat, success\_rate\_vec, used\_classifiers\_vec, avg\_success\_rate] = main();

Inside the main directory.

Script Output:

* predicted\_labels\_mat – 300 X400 matrix of prediction for each missing SNP for each example in the test sample. Also saved to ytest.mat
* success\_rate\_vec – vector containing the 10 fold cross validation success rate for each SNP.
* used\_classifiers\_vec – a vector containing the chosen classifier method(ie. 'adaboost'\'svm' …).
* avg\_success\_rate – the total success rate(average over all SNPs).