Davidson Shahar – 203256128 – [davidson.shahar@gmail.com](mailto:davidson.shahar@gmail.com)

Sheffer Roee – XXX - [bluewaffles@gmail.com](mailto:bluewaffles@gmail.com)

**Project – introduction to machine learning**

How to run the script:

XXX

Results:

Average success rate (over snps), 3-fold cross validation.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Success Rate | .m file | Notes |
| K-Nearest Neighbors |  |  |  |
| SVM |  |  |  |
| SVM2 (pre-processing) |  |  |  |
| Decision Tree |  |  |  |
| Adaboost |  |  |  |
| Histogram\_alg + svm |  |  |  |

Few notes:

* 3-Cross-validation. 0-1 los function. Success of algorithm is avg of all snp's success.
* Boosting: algorithm per snp
* Analysis:
  + Corelations with other snp
  + Some snp's are hard for every algorithm

Algorithms

* K-Nearest Neighbors:
  + Predict snp i of person j: find the k nearest neighbors (L2 norm) from training set. Prediction according to their label.
  + Vector representation of missing snp: vector of 100 snp's before it, and 100 after it.
  + We found that snps has local affection on other snps (correlations). Thus, we want the closest snp's to affect the most on the missing one. The algorithm multiply the vector with 1-d Gaussian, weighting the near snps.
  + Parameters:
    - K – for KNN
    - Sigma – for the weighting Gaussian. Bigger sigma -> bigger effect of wide windows around the missing snp
* SVM:
  + Again, representing a missing snp, with vector of it's R nearest snp's from each side. i.e: 100. We noticed that near snp affect more than far, and therefore the Radius R.
  + For each missing snp, a different svm model was trained.
  + Using libsvm.
  + Parameters:
    - R – radius of window around the missing snp
    - Svm options – passed to libsvm
* SVM2:
  + The same as previous svm, but with smarter features selection.
  + During the training stage, For each missing snp, find it's correlations (including permutations of 0 1 2) with near snps. Take as features the best X correlated one.
  + Continue with libsvm.
  + Parameters:
    - R , svm options – like previous SVM algorithm
    - X – number of top correlated near snp's to take.
* Decision Tree:
  + Using matlab implementation: ClassifiacationTree
  + Create for each snp a decision tree, pruned to level 2 (matlab api).
  + Consider only snps that are within R window around the missing one.
  + Parameters:
    - R – radius
* Adaboost:
  + Our implementation.
  + Weak classifiers are: look at the snp in the index i, check if it's value is j.
  + For each snp: trained 3 binary classifier: 0 or else, 1 or else, 2 of else. On test sample, run the three of them, taking the label with biggest coefficient result (before taking only the sign).
  + Parameters:
    - R – radius
    - T – number of iterations for adaboost
* Histograms descriptor + SVM:
  + tried to find different features vectors for the SVM.
  + Histograms descriptor: taking a window of snps around the missing snp. Dividing it to buckets, and create a vector from the histograms of each bucket.
  + Parameters:
    - Width – of the window.
    - Slide interval – size of each bucket inside the window.

Submitted Files