CNEP Instruction Manual

This manual is for program version that accepts the following set of files:

1. Features from .bed files
2. Labels from conserved elements file
3. Exon to be removed from exon file
4. Chromosome lengths file (e.g. hg19.chrom.sizes)

Training

createFeatureVector.c

The program first builds individual feature (row) vectors for each feature file and then it combines them along with labels in a design (training) matrix in liblinear format (to be used for training with liblinear package). The following options are available for labels: pi, omega, phastCons and GERP.

This program also has an option to generate training sample by randomly picking positions from all chromosomes except the one that's held out for test. It saves those to a file to be read by the rest of the code, so the same samples could be read subsequently on the next code iteration that is not regenerating samples (if needed). Each sample file is set to containing 1 million samples (rows) in the two column format, where first column identifies chromosome number and the second column sampled position on that chromosome. Sampling is proportional to chromosome size.

This code builds feature vectors from user supplied files in .bed format, which contains three columns. In the first column is chromosome number, in the second column is start position (0 based inclusive), and in the third column is end position (0 based exclusive) of the measured peak. Each .bed file is treated as a single feature.

The program also requires the user to provide a list of file names of .bed files in .txt format. The reason for this is that the previous version of the code was more versatile and enabled reading of files from various folders. For the same reason, the program requires total number of features to be supplied as a constant. Use ListFeatures.txt as an example:

This file is located in:
/u/home/o/ogrujic/project-ernst/zarlab/sourceDataFiles

createFeatureVector.c is located in:
/u/home/o/ogrujic/project-ernst/zarlab/createFeatureVector/

createFeatureVector.c requires the user to specify number of features and location of the folder where feature files in .bed format are located (update constant in .c code):

```
#define TOTAL_NUM_OF_FEATURES 10836
#define FEATURE_LIST_ALL "/u/home/o/ogrujic/project-
ernst/zarlab/sourceDataFiles/ListFeatures.txt"
```

```
#define FEATURE_FOLDER_ALL "/u/home/o/ogrujic/project-ernst/zarlab/sourceDataFiles/"
```

Another constant that can be updated is the one indicating what the individual (row) vector file will be called:

```
#define SINGLE_FEATURE_VECTOR "/train_x_"
```

Also, you might want to provide updated file containing chromosome lengths (this file obtained by the popular tool, 1st column has chr# 2nd column contains length):

```
#define CHROMOSOME_LENGTHS_FILE "hg19.chrom.sizes"
```

After the constant is updated, the c. file needs to be recompiled using the following command:
gcc createFeatureVector.c -lm -o createFeatureVector

Sample script for running the program runFeatureVectorSingle.sh (also located in /u/home/o/ogrujic/project-ernst/zarlab/createFeatureVector/) can be edited according to the following options:
filename part[0-12] **test_chromosome**[0-23] **sampleNumber training_sample_file feature_vector_folder label_set**[1-4]

filename is the name of training vector file to be called (e.g. train_x_liblinear). Note: the file will have an extension added by the program based on the sample number provided (e.g. train_x_liblinear_82m)

part should be left at 0 (it was originally set for individual feature vectors to be created in parallel)

test chromosome number should be specified if new training sample is being created, 0 otherwise

sampleNumber number of training sample (e.g. 82) Note: originally, for each test chromosome there were 100 samples numbered 1 through 100, each sample file containing 1 million positions

training_sample_file what training sample would be called if being created or what it is called if it was already created Note: sampleNumber will be added as extension (e.g. sampleTrainingPositions_82m)

feature_vector_folder A folder where the program would create and place individual (row) vectors for each feature and where the program would look for the same files in order to build training matrix (e.g. ../train_x_rows) Important Note: sampleNumber will be added as extension (e.g. the program will place and look for files in: ../train_x_rows _82m), so when you're creating this folder, create it with extension: _#m where # denotes sample number

label_set number indicates the conserved element set for labels according to the following chart:
1 Pi
2 Omega
3 phastCons
4 GERP
Note: Conserved element files are referenced in the following constants and their location needs to be updated:

```
#define PI_FILE "/u/home/o/ogrujic/project-
ernst/zarlab/sourceDataFiles/hg19_29way_pi_lods_elements_12mers.chr_specific.fdr_0.1_with
_scores.txt"
```

```
#define OMEGA_FILE "/u/home/o/ogrujic/project-
ernst/zarlab/sourceDataFiles/hg19_29way_omega_lods_elements_12mers.chr_specific.fdr_0.1_w
ith_scores.txt"
#define PHASTCONS_FILE "/u/home/o/ogrujic/project-
ernst/zarlab/sourceDataFiles/phastconselements_100_hg19.txt"
#define GERP_FILE "/u/home/o/ogrujic/project-ernst/zarlab/sourceDataFiles/GERP/hg19_chr"
```

Note: GTF file was used to create EXON_FILE

```
#define GTF_FILE "/u/home/o/ogrujic/project-
ernst/zarlab/sourceDataFiles/gencode.v19.annotation.gtf"
#define EXON_FILE "/u/home/o/ogrujic/project-
ernst/zarlab/sourceDataFiles/exon_pos_v19.txt"
```

Note: This program can be run four times in parallel in order to efficiently create four training matrices, one per set of labels. In that case, when the method is applied genomewide, four sets of predictions will be generated. Those four sets can be averaged so that a single prediction score (CNEP) per base in the genome is obtained.

Here's an example how to call a single instance of the program (without the above mentioned script):

./createFeatureVector train_x_liblinear 0 0 82 sampleTrainingPositions ../train_x_rows 1

Training using Liblinear Package for Logistic Regression

Liblinear was downloaded and installed here:
/u/home/o/ogrujic/project-ernst/zarlab/liblinear-2.1

There are sample .sh scripts how to run it, here's an example of one:

```
./train -s 6 -B 1 -c 1 train_liblinear_m train_liblinear_m.model
```

In the example, train_liblinear_m is the training matrix and `train_liblinear_m.model is the resulting model file.`

Applying CNEP Model Genomewide

scorePredictions.c

Predictions can be applied genomewide using scorePredictions.c and several .py files to calculate and plot ROC and PRC values, all located here:

/u/home/o/ogrujic/project-ernst/zarlab/scorePredictions

.c file can be recompiled if needed using:
gcc scorePredictionsAllBP.c -lm -o scorePredictionsAllBP

Sample script file how to run it is called: runScorePredictionsSingle.sh

Averaging Predictions of CNEP Method

MakeCNEPAverage.java

To average predictions, the following file is located here:
/u/project/ernst/ernst/OLIVERA/CNEPAVERAGECODE

The usage is:
java -classpath . MakeCNEPAverage scorefile chrmappingfile outdir

scorefile is a file that contains one line per directory that contains files that should be averaged where files in each directory should be named scores_CHRKEY.wig or scores_CHRKEY.wig.gz

chrmapping is a file that contains one line per directory where each line has a chromosome key and name delimited by a tab

outdir is the output directory

CNEP Predictions

The final CNEP predictions in BigWig format are at:
https://ernst.cass.idre.ucla.edu/public/conservationPredictions/cnep.bw