Mirtron SVM prediction – short manual

The following is a short instruction on how to run the SVM script for mirtron prediction. The code requires Matlab2010. It uses the LIBSVM package (which can be downloaded at http://www.csie.ntu.edu.tw/~cjlin/libsvm/#matlab). Before running the code, make sure that LIBSVM is installed and the location added to the path. Also, add the folder containing the .m files to the path. In this example the .m files are located in the folder `~/mirtrons/SVM/code/`.

```
>> addpath('~/mirtrons/SVM/code')
```

For this sample session we will use the pre-processed fly data, created by the function `organism_data`. (To preprocess the data, run `S = organism_data(n,d,L)` where `n` is a list with all intron names, `d` is a directory with .ct files, and `L` are names of sure mirtrons, likely mirtrons and possible mirtrons.) To load the preprocessed data, first add the folder with the data to the path, or `cd` to the folder. In this example the files are located in the folder `~/mirtrons/SVM/data/`.

```
>> cd ~/mirtrons/SVM/data/
```

Next, load the preprocessed fly data.

```
>> load fly
>> F

F =

                ss: [1420443x4 double]
              seqs: {86540x1 cell}
               mfe: [86540x1 double]
              ss25: [744996x4 double]
      names_unique: {27622x1 cell}
             names: {86540x1 cell}
            labels: [86540x1 double]
    ohang_features: [86540x36 double]
           ohangs: [36x2 double]
     ss25_features: [86540x9 double]
```

For each RNA structure `F.seqs` holds the RNA sequence, `F.mfe` the minimum free energy, `F.names` the name and `F.labels` a label representing the status of the structure (1=sure mirtron, 2=likely/new mirtron, 3=candidate mirtron and 0=not mirtron). `F.ohang_features` holds a binary vector representing the different overhang configurations and `F.ss25_features` holds a vector representing different features of the secondary structure up to the 25 nucleotide on the 3' end (number of bps in ss25, number of bulges in ss25, number of nucleotides in ss25 bulges, number AU bps in ss25, number GU bps in ss25, number GC bps in ss25, number 5' bulges in ss25, number 3' bulges in ss25, number interior loops in ss25). `F.ss` holds all secondary structures (the same information as in the .ct files). `F.ss25` holds the same type of information as `F.ss`, but only up to the 25th nucleotoide on 3' end.

```
>> load fly_training_mirtron_names
>> b

b =
```

```
    'CG6695_in5'
    'CG31772_in12'
    'Atet_in4'
    'VhaSFD_in3'
    'CG1718_in3'
    'CG18004_in2'
    'CG3860_in6'
    'CG31163_in17'
    'CG17274_in4'
    'Lerp_in6'
    'wts_in6'
    'CG2196_in3'
    'CG6432_in3'
    'opa1-like_in6'
```

The model is trained using the function `train_model`. In the example below we train the model on the set of known positive examples (`b`) and 100 randomly selected negative examples. Training the model might take a few minutes. The function returns the model `M`, a set of predictions `p`, the training set used `Ftrain` and all introns that passed the filters `Fkeep`. (For more details on how the SVM is trained, we refer to the Matlab code in the file `train_model.m`.)

```
>> [M,p,Ftrain,Fkeep]=train_model(F,b,100);
applying filters 1-3 to the data and collecting data ..
selecting 100 negative training introns from filtered data ...
.
.
.
```

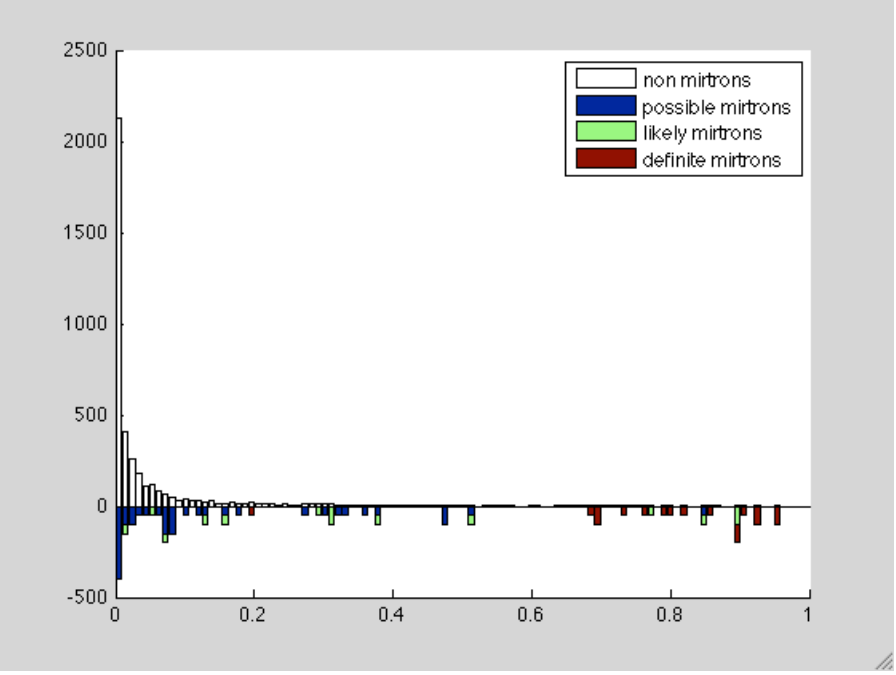
The structure `p` contains all predictions. For each intron, the name, label and prediction score are given. The scores are between 0 and 1, where higher scores mean higher probability that the intron is a mirtron. (Introns that did not pass the initial filtering step are assigned a score of -99.)

```
>> p

p =

    labels: [27622x1 double]
     preds: [27622x1 double]
     names: {27622x1 cell}

>> p.names(1:10)

ans =

    'CG7927_in2'
    'CG18815_in4'
    'CG3630_in3'
    'CG6370_in2'
    'CG6752_in9'
    'CG6695_in5'
    'CG31772_in12'
    'Atet_in4'
    'VhaSFD_in3'
    'CG1718_in3'
```

```
>> p.preds(1:10)

ans =

  -99.0000
  -99.0000
    0.0032
  -99.0000
  -99.0000
    0.9206
    0.9101
    0.8544
    0.8160
    0.7873

>> p.labels(1:10)

ans =

     0
     0
     0
     0
     0
     3
     3
     3
     3
     3
```

A quick way to visualize the results is through the function `plot_histogram`. In the plot below the x-axis shows the scores, and the y-axis the number of introns with different labels.

```
>> plot_histogram(p.preds,p.labels,0)
```



To run the model on a different set of introns (e.g. from a different species), use the function `test_model`:

```
>> [Tpred,Tkeep]=test_model(M,Ftrain,NewSpecies);
```

Here `M` is the previously trained model, `Ftrain` the data used to train the model, `NewSpecies` preprocessed data for another species (same format as `F` above). The function returns a set of predictions `Tpred` and all introns that passed the filters `Tkeep`.

For more information, we refer to the Matlab code. To find out more about a function, use help (e.g. `help train_model`, `help test_model` and `help organism_data`).