PSTk-Classifier 1.0
Software for classifying DNA sequences based on oligomer frequencies

Daniel Dalevi

September 16, 2010

1 Introduction

PSTk-Classifier is a software written in C++ for classifying DNA using a Bayesian approach. Different underlying models can be selected — Naive (Nk), Markov (Mk) and Variable Length Markov (VLMK). The classifier works by first constructing profiles for all groups using fasta-files directly. The profiles are kept in a directory. Then sample sequences (in a multifasta file) can be scored against the profiles and a high-score list will be presented. Note, currently only a flat prior is implemented. Contact author if you need to specify a different prior.

2 Methods for pruning the Prediction Suffix Tree (PST)

All ideas use a bottom-up algorithm that starts with a big PST that overfits data. This tree is gradually pruned by removing leaf-nodes until a certain criterion is fulfilled. The method is either that of [**?**] where a cut-off is specified by the user, or, a method that prunes the tree until it has a certain number of parameters. The latter [**?**] works by assigning all leaves a score ($\Delta_{v,w}$) which depends on the Küllback-Liebler distance between its own next-symbol-probabilities and the next those of the parent. If we denote the context string v of the leaf-node i_v and let w be the $|v|-1$ suffix of v, then

$$\Delta_{v,w} = \sum_{a \in \Gamma} p[a|v] log\Big(\frac{p[a|v]}{p[a|w]}\Big) N(v) \tag{1}$$

Where Γ is the alphabet and $N(u)$ is the count of word u. The leaf-nodes with the smallest deltas are removed until the tree has approximately "-npar" parameters.

3 Installation

1. Download the compressed archive *classifier.tar.gz*

2. Unpack

   ```
   tar xvzf classifier.tar.gz
   ```

3. Change to the main-directory

   ```
   cd classifier
   ```

4. Build the binaries

   ```
   make
   ```

5. The binaries will be find in the directory *bin*

4 How to use the program

The software does not have a graphical user interface and only function from the command line. Once installed you may test that the Classifier is proper built by typing

```
./Classifier -h
```

The software will print all options and a short description on what they do.

4.1 Input files

The input file for the profiles, using the ”-crr” option, is a single-fasta file that looks like this,

```
>gene-name or other information
AACCCTTGAATA
ATAAAA
```

The length of the lines should not be longer than 1000 characters (because off the implementation using the getline function in C++). If they are the string may be truncated and you miss out on the last part of the sequence. The program linebreaker can be installed if you want to convert files. To compile linebreaker go to the *src* directory and type.

The file for scoring against the profiles, using the ”-sap” option, is a multifasta-file, which is the same as above but having one or several entries. It looks like this.

```
>gene-name 1
AACCCTTGAATA
>gene-name 2
AACCAAAGAAAA
ATCCCTT
>gene-name 3
GGGGAAAAAATTT
TTAAAA
```

```
c++ -O2 -o linebreaker linebreaker.cpp
```

The syntax is

```
./linebreaker infile.fasta > myoutfile.fasta
```

4.2 Options

All options are run from command-line by adding a "-" sign prior to the option. For example, order_estimator -crr. All Boolean are false by default and set to true using the specifier without the word true. All other specifiers need a value.

alp (Integer) Alphabet. Currently only Binary (0), DNA(1) and Peptide (2). DNA is default. If you wish to use another one, contact author

c_c (Boolean) Use if you want to specify a constant cut-off ("-nc") for training in the method of [**?**].

ipf (String) A postfix that is shared between all input files to process

ipwd (String) Input directory

opf (String) A postfix added to output files

osf (String) A prefix added to output files

isf (String) A prefix that is shared among all input files

opwd (String) Output directory

nc (Integer) Constant cut-off for pruning tree (5.0)

kmax (Integer) Maximum depth of PST, ie. longest possible memory

k (Integer) Order k in fixed Markov model, or, word-length - 1 if a naïve model.

m (Integer) Model. fixed order Markov model (0), variable length Markov model (1) and naïve model (2).

minc (Integer) Constraint on nodes. This is the minimal count a node needs to be in the PST.

npar (Integer) Prune tree until having this many parameters.

f (String) Specify the name of a fasta-file to process Many "-f" can be used.

f_f (String) Specifying a file including names of all files to be processed.

format (Boolean) False by default. The output format in variable option. Currently two possible formats. One a format suitable for running the program s̈eq_genänd another which simply prints the a PST tree similar to the format in R. The latter is default.

rnd (Boolean) Randomly sample a region of length "-l" from the input file. Specify seed using "-s". If no seed, clock will be used and hard to reproduce result.

s (Integer) Seed for random number generator.

l (Integer) Length of sequence when randomly samples from fasta file. Used in "-sap" option (1000)

crr (Boolean) Create a random profile tree of 1 - "frac" percent of the sequence. Remaining part will be saved in a file specified with "off"

ffn (Boolean) Used to concatenate regions when training using a multi-fasta file. Examples of the *.ffn format can be found on the GenBank ftp-server.

frac (Float) Fraction to use for testing, the remaining sequence is used for the training-set used in building the profile.

aic (Boolean) Print AIC value for PST (option not tested!).

sap (Boolean) Score sequence(s) against profiles created using "-crr" option.

nocum (Boolean) Do not use accumulated probabilities in PST. Do not use this option.

nofna (Boolean) Do not output a fasta-file with the training sequence.

nsamp (Integer) Number of samples of random sequences drawn from sequence "-seq" specified in "-sap" option negative number=whole sequence will be used (-1)

print (Boolean) Draw tree-structure a profile-file which you specify with "-f"

pseudo (Boolean) Use pseudo-counts when next-symbol counts are zero.

pruning (Boolean) Pruning algorithm. The method in [**?**] (KL), Peres-Shields method (PS). These are only implemented for PSTs. KL is default.

revcomp (Boolean) Use when also consider reverse complementary strand

scan (Boolean) Score against one-self. Specify a window of size "-win" to use this option.

slen (Integer) Step increment in "-scan" (1)

4.3 Examples

4.3.1 Creating profiles

The profiles are a set of files that you create which corresponds to the groups a sequence can be classified to. For example, you may want to use Escherichia coli K12 and 10 other organisms. PSTk-classifier will save the profile to a file with the postfix ".tree" which looks like this:

```
Name: bnrf1EB
Date: 15:27:48, 24 Oct, 2005
Tree: Fixed
Alphabet: DNA
Number(nodes): 5
Number(parameters): 12
Node: 0 # [ 743 1195 1232 783 ] 0 [ 0 0 0 0 ][ 1 2 3 4  ]
Node: 1 A [ 130 283 244 86 ] 744 [ 130 219 247 148 ][ -1 -1 -1 -1  ]
Node: 2 C [ 219 380 367 229 ] 1195 [ 283 380 262 270 ][ -1 -1 -1 -1  ]
Node: 3 G [ 247 262 417 306 ] 1231 [ 244 367 417 203 ][ -1 -1 -1 -1  ]
Node: 4 T [ 148 270 203 162 ] 783 [ 86 229 306 162 ][ -1 -1 -1 -1  ]
```

The format has the following tags,

Name The name from the header which is copied from the fasta.

Date Time when created

Tree The model used which currently is Fixed, PST or Naïve.

Number(nodes) Number of nodes in the tree

Number(parameters) The number of free parameters in the model.

Node One per each node specifying the string obtained walking from the node to the head. # indicates head-node. The first nine integers have keeps track of different counts around the context and the last four integers specify if this node is connected to another node. If it is that nodes number is given otherwise "-1"

The syntax for creating the *.tree files differ depending on which model that is used. Below you find an example of each.

4.3.2 Naïve model

Use this option if you want to use a Naïve model of dinucleotides ($k = 1 = wordlength - 1$).

```
./classifier -crr -m 0 -k 1  -f data/sol_backbone_pb10 -ipwd data -ipf .fna
```

A file named sol_backbone_pb10.tree will be created.

4.3.3 Fixed order Markov model

Same syntax as Naïve model.

4.3.4 Variable length Markov model: PST

This is an example where we want to create a PST from the file sol_back_bone_pb10.fna which is located in the directory named data. We want the output to have prefix 60_ and be located in a directory named profiles.

```
daniel@linux:> ./classifier -crr -f sol_backbone_pb10 -ipf .fna -ipwd data
-pseudo -osf 60_ -npar 60 -minc 10 -kmax 9 -frac 0.1 -s 222
-nofna
```

A file named 60_ sol_backbone_pb10.tree will be created.

4.3.5 Printing a profile to screen

Sometimes you may want to see how the PST looks like. This can be done with the "-print" option. Syntax:

```
daniel@linux:> ./classifier -print -f profiles/60_sol_backbone_pb10.tree
[#]--(2391 4218 4054 2036 | 12699)
+---[A]--(575 704 645 467 | 2391)
      +---[AA]--(83 201 227 64 | 575) -T
      +---[CA]--(257 187 246 188 | 878) -T
      +---[TA]--(18 104 20 29 | 171) -T
+---[C]--(878 1101 1572 668 | 4219)
      +---[AC]--(130 218 276 80 | 704) -T
      +---[CC]--(229 200 436 236 | 1101)
            +---[GCC]--(111 105 229 85 | 530)
                  +---[GGCC]--(49 33 82 35 | 199)
                        +---[TGGCC]--(1 16 28 12 | 57) -T
      +---[GC]--(345 530 501 259 | 1635) -T
      +---[TC]--(174 153 359 92 | 778)
            +---[ATC]--(59 48 92 28 | 227)
                  +---[GATC]--(20 18 27 17 | 82)
                        +---[CGATC]--(14 17 1 13 | 45) -T
+---[G]--(767 1635 1080 571 | 4053)
      +---[GG]--(148 572 192 168 | 1080) -T
      +---[TG]--(145 229 277 106 | 757) -T
+---[T]--(171 778 757 330 | 2036)
      +---[AT]--(20 227 171 49 | 467) -T
      +---[CT]--(74 202 272 120 | 668) -T
```

The next-symbol counts for the specific context ([]) are shown within brackets() with the right-most value being the total count. The T indicates that it is a terminal node.

4.3.6 Score a sequence against created profiles

Here we wish to score the sequence in the fasta file mygene.fna mygenes.ffn against the profiles listed in list.txt to see wish is the most likely candidate. Before we started we put all the profiles in the directory named *profiles*.

```
./classifier -sap -s 333 -f list.txt -ipf .tree -ipwd profiles -isf m2_
-seq mygene.fna -nsamp -1 -k 2 -m 0 -revcomp
```

Note, you have to use the same profiles. That is, you cannot score against one profile of Markov type and another of Naïve type. The file list.txt looks like this:

```
sol_backbone_pb10
sol_backbone_r751
Escherichia_coli_K12
Bacillus_subtilis
```

Where the names are the names of the files without the prefix specified with ”-isf” and the postfix specified with ”-ipf”. So in the profile directory we have these files:

```
m2_sol_backbone_pb10.tree
m2_sol_backbone_r751.tree
m2_Escherichia_coli_K12.tree
m2_Bacillus_subtilis.tree
```

5 Copyright

6 Reference

Please cite [**?**] if you use this software in any work of research.