

Classification of genome data with n -gram models

Anton Antonov

Mathematica for Prediction blog

Mathematica for Prediction project at GitHub

January 2014

Introduction

In this document we consider the following problem:

Gene Sequence Classification Problem (GSCP): Given two genes, G_1 and G_2 , and a (relatively short) sub-sequence S from one of them tell from which gene the sub-sequence S is part of.

One way to derive a solution for this problem is to model each gene with an n -gram model, [2], and classify the sub-sequences according to the odds ratio test, [3], based on the Markov chain state transition probabilities. We are going to make classification experiments for different n 's of the n -gram model and use a type of Receiver Operating Characteristic (ROC) plot, [4], to judge which n is the best.

This approach can be also applied to text classification. We consider genes for conciseness and clarity.

This document demonstrates the usage of the *Mathematica* package for n -gram Markov chain models provided by the MathematicaForPrediction project at GitHub, see [1].

Mathematica has the function `GenomeData` that gives the DNA sequences for specified genes.

The approach

Our approach to answer GSCP is to define the algorithm named `NGramClassifier(k)` that takes as parameter $k \in \mathbb{N}$, $k > 2$, and then conduct a series of experiments to find the best k .

The algorithm `NGramClassifier(k)`

1. For each gene G_1 and G_2 create a k -gram, $(k - 1)$ order Markov chain model. Denote them NGM_1 and NGM_2 respectively. NGM_i tells what is the probability the sequence of characters $c_1 c_2 \dots c_{k-1}$ to be followed by the character c_k in G_i . We have $c_j \in \{A, C, G, T\}$, $j \in [1, k]$, $j \in \mathbb{N}$.

1.1. Each of the models i has a Markov chain state transition matrix M_i .

2. Partition the sub-sequence S into k -grams. I.e. if $S := \{c_1, c_2, c_3, \dots, c_m\}$ form the set

$$g_k(S) := \{\{c_1, \dots, c_k\}, \{c_2, \dots, c_{k+1}\}, \{c_3, \dots, c_{k+2}\}, \dots, \{c_{m-k+1}, \dots, c_m\}\}. \quad (1)$$

3. To each element of $g_k(S)$ we apply NGM_1 to obtain the sequence of probabilities:

$$\text{cp}_1 := \{p_1^1, p_2^1, \dots, p_{m-k}^1\}. \quad (2)$$

4. To each element of $g_k(S)$ we apply NGM_2 to obtain the sequence of probabilities:

$$\text{cp}_2 := \{p_1^2, p_2^2, \dots, p_{m-k}^2\}. \quad (3)$$

5. We compute the odds ratio test

$$\text{or}(S) := \prod_{i=1}^{m-k} \frac{p_i^1}{p_i^2}. \quad (4)$$

If $\text{or}(S) > 1$ we consider S to be part of the gene G_1 . If $\text{or}(S) < 1$ we consider S to be part of G_2 .

Tuning -- finding the best k for $\text{NGramClassifier}(k)$

1. For each $k \in [1, \dots, 10]$ calculate k -gram models for G_i . For these models use only the first 80% of the gene sequences. (In other words, for each G_i the training set is only 80% of G_i 's length.)
2. Using the last 20% of G_1 and G_2 derive a set of test sequences T by randomly picking G_i and the sub-sequence length within the range $\{l_{\min}, l_{\max}\}$.
3. For each $k \in [1, \dots, 10]$ calculate the classifications of the elements of T using $\text{NGramClassifier}(k)$ with the models calculated in step 1.
4. Calculate the True Positive Rate (TPR) and False Positive Rate (FPR) for each k in step 3.
5. Plot the points with coordinates $\{\{\text{TPR}_k, \text{FPR}_k\}\}_{k=1}^{10}$ and select the best k .

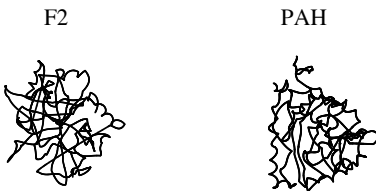
Experiments

The genes

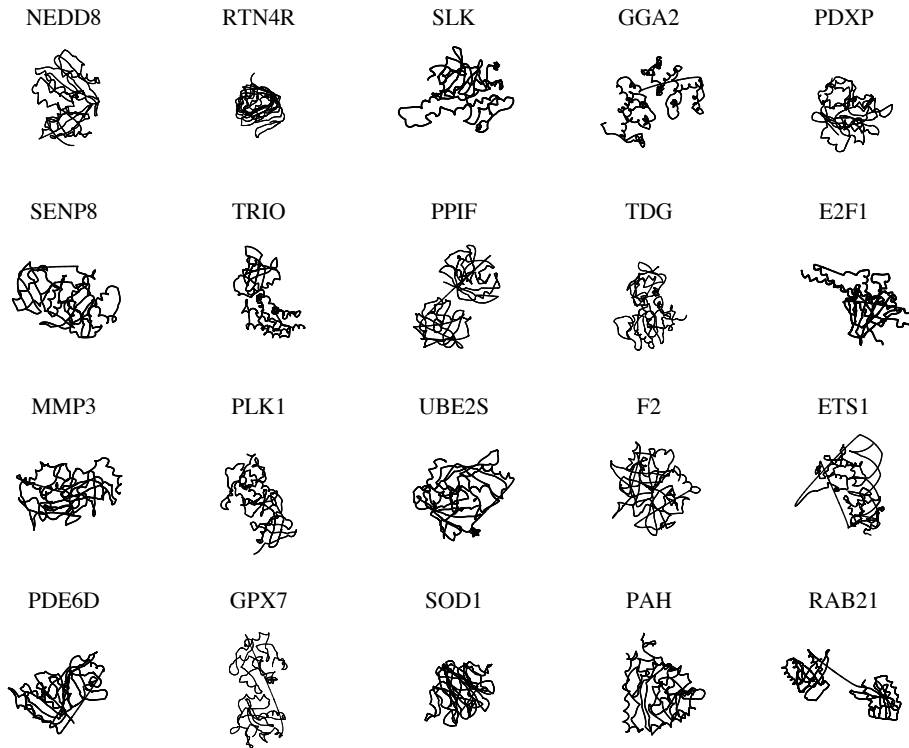
We are going to answer GSCP using the pair of genes “F2” and “ETS1”:



and the pair of genes “F2” and “PAH”:



The genes were picked from this human genome data table:



For more details and examples see *Mathematica's* help browser function page for `GenomeData`.

Package load

Load the package [1]:

```
Get["~/MathFiles/MathematicaForPrediction/NGramMarkovChains.m"]
```

Data and models

Each n -gram model has an array representing the n -gram probabilities and rules of converting gene component letters to indices. *Mathematica* has the symbol `DiscreteMarkovProcess` that has similar but different function and representation. The n -gram models built with [1] use high dimensional arrays instead of matrices. (Matrices are used only for the 2-gram models.)

For example,

```
NGramMarkovChainModel[
  Characters[GenomeData["F2", "FullSequence"]], 2, "ColumnStochastic" -> True]
NGramModel[SparseArray[<64>, {4, 4, 4}],
  Dispatch[{A -> 1, C -> 2, G -> 3, T -> 4}, -DispatchTables -],
  Dispatch[{1 -> A, 2 -> C, 3 -> G, 4 -> T}, -DispatchTables -]]
```

The following commands built the n -gram models for each of the genes.

```

geneName1 = "F2";
sGeneSeq1 = GenomeData[geneName1, "FullSequence"];
StringLength[sGeneSeq1]
20 301

Clear[ngm1]
AbsoluteTiming[
  Do[
    ngm1[morder] =
      NGramMarkovChainModel[Characters[StringTake[sGeneSeq1, {1, 10 000}]],
        morder, "ColumnStochastic" → True],
    {morder, 1, 10}]
]
{335.986954, Null}

geneName2 = "ETS1";
sGeneSeq2 = GenomeData[geneName2, "FullSequence"];
StringLength[sGeneSeq2]
63 502

Clear[ngm2]
AbsoluteTiming[
  Do[
    ngm2[morder] =
      NGramMarkovChainModel[Characters[StringTake[sGeneSeq2, {1, 10 000}]],
        morder, "ColumnStochastic" → True],
    {morder, 1, 10}]
]
{487.276162, Null}

geneName3 = "PAH";
sGeneSeq3 = GenomeData[geneName3, "FullSequence"];
StringLength[sGeneSeq3]
79 278

Clear[ngm3]
AbsoluteTiming[
  Do[
    ngm3[morder] =
      NGramMarkovChainModel[Characters[StringTake[sGeneSeq3, {1, 10 000}]],
        morder, "ColumnStochastic" → True],
    {morder, 1, 10}]
]
{402.186793, Null}

```

Classification example

Here is an example of classification using steps 3,4, and 5 of the algorithm NGramClassifier described above.

```

sampleToGuess = Characters[sGeneSeq1][[11 200 ;; 11 200 + 100]];
StringJoin@sampleToGuess
AGCAGGCATATCTAGGGGAGGAGCACCGCAGGCTGGGGGCATGGCAGGCACTAAGGCCCTGAGGTGGGAGCACTC\
TTGGCTTGTCTGGGGAGCAGTAGGGA

(* step 3 *)
cp1 = ngm1[2][[1]][Sequence@@(# /. ngm1[2][[2]])] & /@
  Partition[sampleToGuess, 2 + 1, 1];
(* step 4 *)
cp2 = ngm2[2][[1]][Sequence@@(# /. ngm2[2][[2]])] & /@
  Partition[sampleToGuess, 2 + 1, 1];
(* step 5 *)
Apply[Times, cp1 / cp2]
"sub-sequence of " <> If[% >= 1, geneName1, geneName2]
269.471

sub-sequence of F2

```

Classification tuning runs F2 vs. ETSI

Generate test sub-sequences:

```

gTests12 =
  Table[
    {offset, slen, gseq} = Flatten@
      {RandomInteger[Floor /@ {0.8 Min[StringLength /@ {sGeneSeq1, sGeneSeq2}],
        0.95 Min[StringLength /@ {sGeneSeq1, sGeneSeq2}]}], 1],
      RandomInteger[{100, 120}, 1], RandomChoice[{1, 2}]}];
    sampleToGuess = Characters[If[gseq == 1, sGeneSeq1, sGeneSeq2]][[
      offset ;; offset + slen]];
    {offset, slen, gseq, sampleToGuess}},
    {2000}];

```

Calculate classifications:

```

classRes12 =
  Table[
    Map[
      cp1 = ngm1[morder][[1]][Sequence@@(# /. ngm1[morder][[2]])] & /@
        Partition[#[-1], morder + 1, 1];
      cp2 = ngm2[morder][[1]][Sequence@@(# /. ngm2[morder][[2]])] & /@
        Partition[#[-1], morder + 1, 1];
      cp1 = cp1 /. {0 -> 10^-8, 0. -> 10^-8.};
      cp2 = cp2 /. {0 -> 10^-8, 0. -> 10^-8.};
      Append[Most[#], Apply[Times, cp1 / cp2]] &,
      gTests12],
    {morder, 1, 10}];

```

Classification tuning runs F2 vs. PAH

Generate test sub-sequences:

```

gTests13 =
  Table[ (
    {offset, slen, gseq} = Flatten@
      {RandomInteger[Floor /@ {0.8 Min[StringLength /@ {sGeneSeq1, sGeneSeq3}],
        0.95 Min[StringLength /@ {sGeneSeq1, sGeneSeq3}]]}, 1],
      RandomInteger[{100, 120}, 1], RandomChoice[{1, 2}]}];
    sampleToGuess = Characters[If[gseq == 1, sGeneSeq1, sGeneSeq3]][
      offset ;; offset + slen];
    {offset, slen, gseq, sampleToGuess}},
    {2000}];

```

Calculate classifications:

```

classRes13 =
  Table[
    Map[ (
      cp1 = ngm1[morder][[1]][Sequence@@ (# /. ngm1[morder][[2]])] & /@
        Partition[#[[-1]], morder + 1, 1];
      cp3 = ngm3[morder][[1]][Sequence@@ (# /. ngm3[morder][[2]])] & /@
        Partition[#[[-1]], morder + 1, 1];
      cp1 = cp1 /. {0 -> 10^-8, 0. -> 10^-8.};
      cp3 = cp3 /. {0 -> 10^-8, 0. -> 10^-8.};
      Append[Most[#[[1]], Apply[Times, cp1 / cp3]]] &,
        gTests13],
    {morder, 1, 10}];

```

ROC rates definitions

The following definitions are of different classification success functions as described in [4].

```

(* true positive rate *)
TPR[i_, classRes_] :=
  Count[classRes[[i]], {_, _, 1, r_} /; r > 1] / Count[classRes[[i]], {_, _, 1, _}] // N

(* true negative rate *)
TNR[i_, classRes_] :=
  Count[classRes[[i]], {_, _, 2, r_} /; r < 1] / Count[classRes[[i]], {_, _, 2, _}] // N

(* false positive rate *)
FPR[i_, classRes_] :=
  Count[classRes[[i]], {_, _, 2, r_} /; r > 1] / Count[classRes[[i]], {_, _, 2, _}] // N

(* false negative rate *)
FNR[i_, classRes_] :=
  Count[classRes[[i]], {_, _, 1, r_} /; r < 1] / Count[classRes[[i]], {_, _, 1, _}] // N

```

The ROC points plot function is define to resemble the ROC curve plot in [4].

```

In[768]:= ROCPointsGraph[tbl : {[_Integer, _?NumberQ, _?NumberQ] ..}, ratesType_String,
  geneName1_String, geneName2_String, opts : OptionsPattern[]] :=
  Graphics[{Gray, Dashed, Line[{0, 0}, {1, 1}], PointSize[0.02],
    Darker[Red], Tooltip[Point[Rest[#]], First[#]] & /@tbl,
    Darker[Blue], Text[First[#], Rest[#], {-2, -2}] & /@tbl},
  PlotRange -> {{0, 1}, {0, 1}}, Frame -> True, Axes -> False,
  AspectRatio -> 1, FrameLabel -> Map[Style[#, FontSize -> 16] &,
    {"false " <> ratesType <> " rate", "true " <> ratesType <> " rate"}],
  GridLines -> {FindDivisions[{0, 1}, 10], FindDivisions[{0, 1}, 10]},
  GridLinesStyle -> Directive[Gray, Dashed], PlotLabel -> Style[
    "ROC for genome classification\n by n-gram models\n " <> geneName1 <>
    " vs. " <> geneName2, FontFamily -> "Times", FontSize -> 16], opts];

```

ROC for F2 vs. ETSI

Compute and plot the “standard” ROC with positive rates and its dual with negative rates.

```

In[801]:= tblP = Table[{i, FPR[i, classRes12], TPR[i, classRes12]}, {i, 1, 10}];

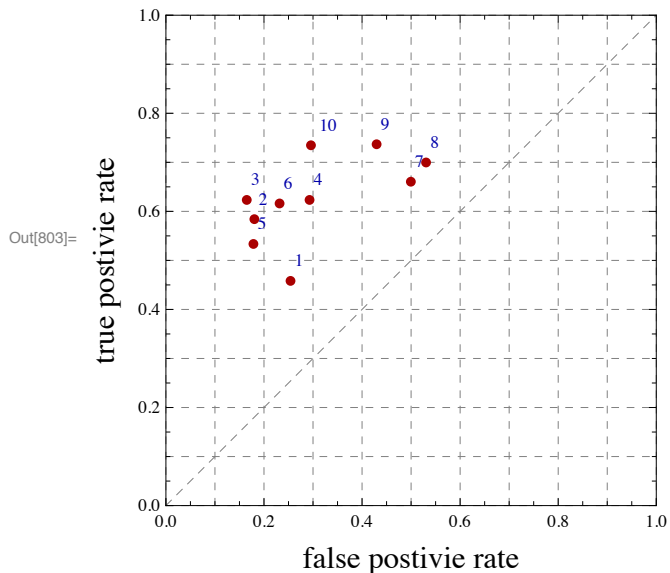
```

```

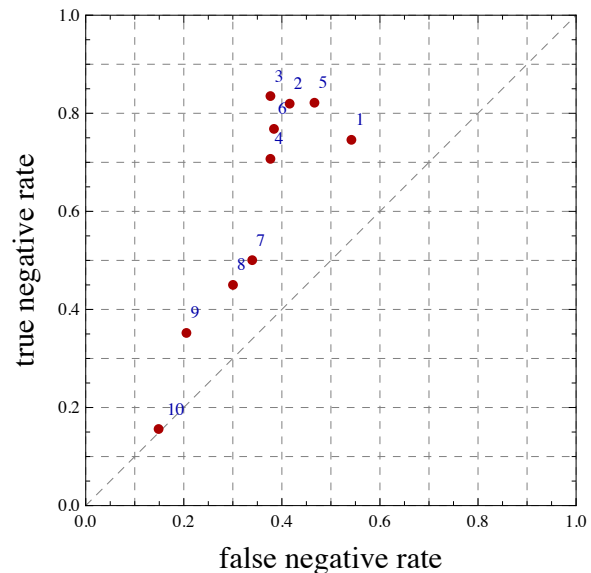
In[802]:= tblN = Table[{i, FNR[i, classRes12], TNR[i, classRes12]}, {i, 1, 10}];

```

ROC for genome classification
by n-gram models
F2 vs. ETS1



ROC for genome classification
by n-gram models
F2 vs. ETS1



ROC for F2 vs. PAH

Compute and plot the “standard” ROC with positive rates and its dual with negative rates.

```

In[797]:= tblP = Table[{i, FPR[i, classRes13], TPR[i, classRes13]}, {i, 1, 10}];

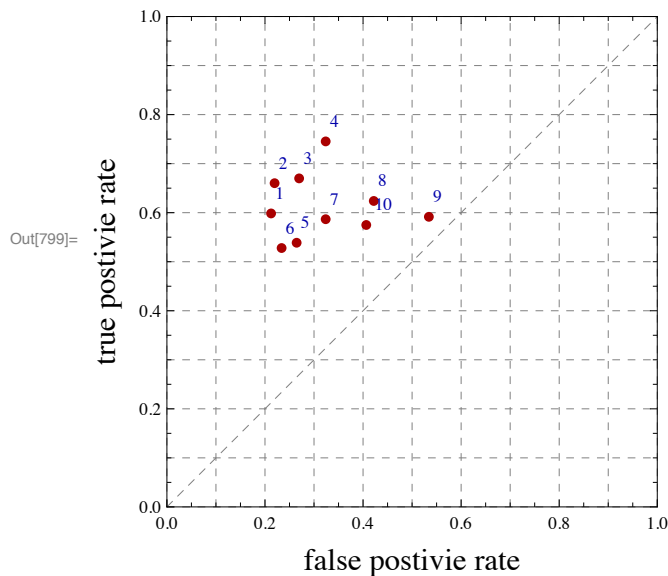
```

```

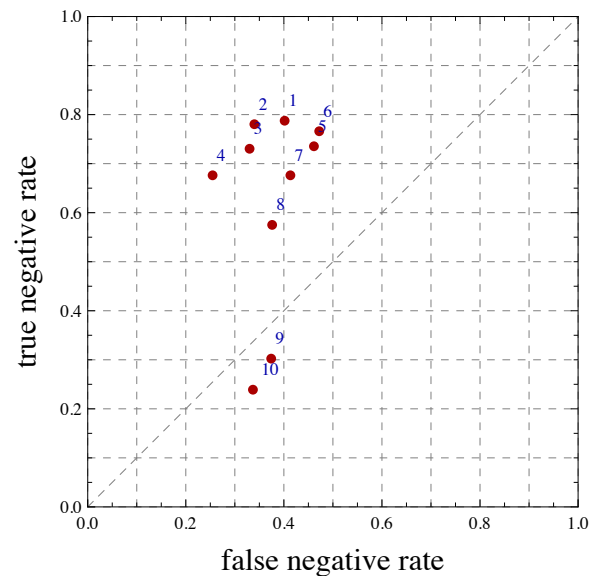
In[798]:= tblN = Table[{i, FNR[i, classRes13], TNR[i, classRes13]}, {i, 1, 10}];

```

ROC for genome classification
by n -gram models
F2 vs. PAH



ROC for genome classification
by n -gram models
F2 vs. PAH



Conclusions

From the ROC plots in the previous section we can see that the best classification results for GSCP with the genes “F2” and “ETS1” are obtained with Markov chain order 3 or with 4-grams. The 3-grams are almost as good.

A natural extension of the experiments described is to repeat them for other pairs of genes and across different lengths of sub-sequences. In this way we can derive more general conclusions for the best n -gram length in the algorithm NGramClassifier.

Repeating the experiments for the genes “F2” and “PAH” showed that using 3-grams gives best results for GSCP.

References

- [1] Anton Antonov, N -gram Markov chains *Mathematica* package, source code at GitHub, <https://github.com/antononcube/MathematicaForPrediction>, package NGramMarkovChains.m, (2014).
- [2] Wikipedia entry, n -gram, <http://en.wikipedia.org/wiki/N-gram>.
- [3] Wikipedia entry, Odds ratio, http://en.wikipedia.org/wiki/Odds_ratio.
- [4] Wikipedia entry, Receiver operating characteristic, http://en.wikipedia.org/wiki/Receiver_operating_characteristic.