

Theoretical and practical metagenomic approaches to viral discovery

Practical Session: Random Forest Classifier and viral application

Kevin Lamkiewicz, Manja Marz

23.10.2019

European Virus Bioinformatics Center

How to: Implement the Random Forest Classifier

RANDOM FOREST

```
1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn import datasets
3 # we know this already...
4 iris = datasets.load_iris()
5 data = iris.data
6 target = iris.target
7
```

RANDOM FOREST

```
1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn import datasets
3 # we know this already...
4 iris = datasets.load_iris()
5 data = iris.data
6 target = iris.target
7
8
9 # create the classifier object
10 # n_estimators: number of trees for the forest
11 # max_depth: maximum depth of one tree
12 rfc = RandomForestClassifier(n_estimators=100, max_depth=2)
13 rfc = rfc.fit(data, target)
```

Training our model to identify viral elements

WE WANT TO IDENTIFY VIRAL PRE-MiRNAs

Our task:

In the `miRBase` database are currently around 320 viral pre-miRNAs. We want to use them in to train a machine learning model that can distinguish between viral pre-miRNAs and other sequences.

WE WANT TO IDENTIFY VIRAL PRE-MiRNAs

Our task:

In the `miRBase` database are currently around 320 viral pre-miRNAs. We want to use them in to train a machine learning model that can distinguish between viral pre-miRNAs and other sequences.

I already prepared...

- ▶ a file with all precursors
- ▶ a file with our negative data
- ▶ some utilities we need (reading the data, ...)

WE WANT TO IDENTIFY VIRAL PRE-MiRNAs

Our task:

In the `miRBase` database are currently around 320 viral pre-miRNAs. We want to use them in to train a machine learning model that can distinguish between viral pre-miRNAs and other sequences.

I already prepared...

- ▶ a file with all precursors
- ▶ a file with our negative data
- ▶ some utilities we need (reading the data, ...)
- ▶ Features?

EXERCISE

EXERCISE

Your task

Have a look at the `rf_classifier.py` script. Try to understand as much of it as possible.

Then, create a Random Forest Classifier and train the model with the data – `precursor.fa` and `negative_set.fa`

Let us discuss our results

FIRST THE BASICS

```
1 # first we need to translate the sequences into something
2 # that can be used by the machine learning algorithm:
3
```

FIRST THE BASICS

```
1 # first we need to translate the sequences into something
2 # that can be used by the machine learning algorithm:
3
4 import numpy as np
5
```

FIRST THE BASICS

```
1 # first we need to translate the sequences into something
2 # that can be used by the machine learning algorithm:
3
4 import numpy as np
5
6 # read in the data
7 trainingsData = read_training_set(positive, negative)
8
```

FIRST THE BASICS

```
1 # first we need to translate the sequences into something
2 # that can be used by the machine learning algorithm:
3
4 import numpy as np
5
6 # read in the data
7 trainingsData = read_training_set(positive, negative)
8
9 # transform the sequence into features
10 trainingSet, targets = transform_data(trainingsData)
```

FUNCTION DEFINITIONS

```
1 def transform_data(data):  
2     # this will be our data structure  
3     trainingSet = []  
4     targets = []  
5
```


FUNCTION DEFINITIONS

```
1 def transform_data(data):
2     # this will be our data structure
3     trainingSet = []
4     targets = []
5
6     # iterate over all sequences
7     for dataPoint in data:
8         sequence = dataPoint[0]
9         group = dataPoint[1]
10        # extract the features
11        length = len(sequence)
12        ...
13
```

FUNCTION DEFINITIONS

```
1 def transform_data(data):
2     # this will be our data structure
3     trainingSet = []
4     targets = []
5
6     # iterate over all sequences
7     for dataPoint in data:
8         sequence = dataPoint[0]
9         group = dataPoint[1]
10        # extract the features
11        length = len(sequence)
12        ...
13
14        # store everything in our array
15        vector = [length, ...]
16        trainingSet.append(vector)
17        targets.append(group)
18
```

FUNCTION DEFINITIONS

```
1 def transform_data(data):
2     # this will be our data structure
3     trainingSet = []
4     targets = []
5
6     # iterate over all sequences
7     for dataPoint in data:
8         sequence = dataPoint[0]
9         group = dataPoint[1]
10        # extract the features
11        length = len(sequence)
12        ...
13
14        # store everything in our array
15        vector = [length, ...]
16        trainingSet.append(vector)
17        targets.append(group)
18
19    # return the data within a numpy array
20    return(np.array(trainingSet, dtype=float), targets)
```

PERFORMANCE OF OUR MODEL

```
1 # we know this already. we split the data into training and test sets
2 data_training, data_test, target_training, target_test =
3     train_test_split(trainingSet, targets, test_size=0.2)
4
5 # creating the random forest classifier and already fitting the training data
6 rfc = RandomForestClassifier(n_estimators=100, max_depth=2).
7     fit(data_training, target_training)
8 # prediction of the test set
9 prediction = rfc.predict(data_test)
10 # output the confusion matrix to evaluate the model
11 print()
12 print(metrics.confusion_matrix(target_test, prediction))
13 print(metrics.accuracy_score(target_test, prediction))
```

COFFEE BREAK

