DS 3BIM 2015-2016

Sergio Peignier Zapata

1 Markov chains (40 minutes or less)

A Markov chain is a stochastic process characterized by a sequence $(x_1, x_2, x_3, \ldots)$ of random variables taking values on a state space X. Let us call $(X_1, X_2, \ldots)$ the sequence of states actually taken by the random variables. This process undergoes transitions from one state to another on the space X. The transition probabilities between such states obey to the following Markov property : given the present state, the future and past states are independent:

$$P(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x \mid X_n = x_n)$$

Enough theory! In this exercice we will consider sequences of DNA nucleotides and thus we only have 4 states (A,C,G and T). Our markov chains will evaluate the probability to observe a given nucleotide knowing that we have observed a particular nucleotide on the previous index, e.g., we will evaluate the probability $P(C|T)$ to observe a C knowing that our previous observation was a T (conditional probability) and so on. Consequently we will have a matrix of transition probabilities ...

1.1 Problem statement

Choose and justify a suitable structure for the code that you will produce to solve the following problems.

- Load the Ebola and Narnavirus sequences from the FASTA files .fa (I do not want you to copy/paste the sequence!)
- Build the transitions matrices associated to each one of both virus sequences. This function should be able to work with any sequence (not only ATGC)!
- Plot both transition matrices (heatmap) ... are there major differences?

We will consider that the previous matrices are good models for the yeast virus Narnavirus sequence and for Ebola virus sequence respectivelly.

Let us define the likelihood of a sequence of states $S = (S_1, S_2, \ldots, S_n)$ with respect to a given model $\mathcal{M}$ as the probability to observe S according to the transition probabilites of $\mathcal{M}$. We estimate this function by computing the following product of probabilities $P(S_2|S_1, \mathcal{M}) \times P(S_3|S_2, \mathcal{M}) \times P(S_4|S_3, \mathcal{M}) \times \ldots \times P(S_{n-1}|S_n, \mathcal{M})$ For example imagine that according to $\mathcal{M}$ the probability to have a C after a T is equal to $P(C|T) = q$ and the probability to have a T after an C is equal to $P(T|C) = p$, then the likelihood of the sequence $S = CTCTCT$ is just $\mathcal{L}(S|\mathcal{M}) = p \times q \times p \times q \times p$. The Log-Likelihood is just the logarithm of the previous mesure which means that we will sum the logarithms of the probabilities to compute the Log-Likelihood.

- Compute the Log-Likelihoods of the unknown sequences with respect to each one of the previous models
- Which sequence is a Narnavirus virus one ... and an Ebola virus one?
- Why is it better to use the Log-Likelihood (Bonus: you can try to compute both the likellyhood and the log-likelihood and see want happens)?

2 Perceptron (40 minutes or less)

This algorithm is one of the first neural networks to be designed (during the 1950s). This algorithm is very simple and is used to do binary classification, i.e., the algorithm decides if an object described in D dimensions (just as in the k-means TP for iris) belongs to one class or the other (i.e., class 1 or -1). The perceptron uses a linear predictor function combining a set of real-valued weights $w = (w_1, w_2, \ldots, w_D)$ with the object features $x = (x_1, x_2, \ldots, x_3)$ to predict its class:

$$class(x) = \begin{cases} 1 & \text{if } (\sum_{j=0}^{D} w_j x_j) > 0 \\ -1 & \text{otherwise} \end{cases}$$

In order to make good predictions the perceptron only needs to compute the right weights w. Choose and justify a suitable structure for the code that you will produce to solve the following points. Please make the program the more general you can.

- Initialize the weights to a small random value (between $-b$ and b), with $b = 0.1$ and intialize $\alpha = 0.3$.

- For each dataset example x_i and its associated class membership y_i:
  - Calculate the predicted class $\gamma_i = class(x_i)$.
  - Update the weights: $w_j \leftarrow w_j + \alpha(y_i - \gamma_i) \times x_{i,j}$ for all features $0 \leq j \leq n$
- Program the perceptron algorithm and use it to classify the dataset sent by e-mail.
- Plot the confusion matrix.
- The perceptron has a parameter α. What does this parameter do?
- Bonus: the step related to the weights update is an classic optimization technique ... What is the name of this method? Which function is being optimized?

3 Random forest: Using a new library(40 minutes or less)

The Random Forest algorithm is a suppervised learning task that "learns" how to infer a function (a class memberships for example) from a set of labeled training examples. Each example is a pair $\langle x, y \rangle$, x is an object described in D dimensions (just as in the k-means TP for iris) and y is the desired output value (the class, e.g., the species for the iris dataset). In the ideal scenario the algorithm should be able to correctly determine the class labels for unseenobjects. Choose and justify a suitable structure for the code that you will produce (the structure can be very unsophisticated) to solve the following problems. Hint: Please do not reinvent the wheel and use pandas, scipy.stats and numpy ...

- Download the dataset wine.data at the website :
  http://archive.ics.uci.edu/ml/machine-learning-databases/wine/
- Load the dataset
- Shuffle the dataset rows
- Separate the first columns (known classes) from the other columns. Let us call this column Y and the rest of the dataset X.
- Replace the coordinate $x_{i,j}$ of each object $x_i \in X$ by its z-score, i.e., $z_{i,j} = \frac{x_{i,j} - \mu_{.,j}}{\sigma_{.,j}}$ where $\mu_{.,j}$ is the mean value along the dimension j and $\sigma_{.,j}$ is the standard deviation along the same dimension.
- Split X and Y in two parts, the first 100 rows will be used as a training set ($X_{training}$ and $Y_{training}$) and the remaining rows will be used as test set (X_{test} and Y_{test})
- Generate a random forest classifier using the class RandomForestClassifier from the library sklearn.ensemble.
- Train (fit) the classifier using the training set
- Test the classifier with the test set and output its quality (score)
- This algorithm has an important parameter, which is the number of trees in the forest. Repeat the three previous steps for different number of trees (1,10,20,30,40,50,60,70,80,90,100) and plot the scores as a function of the number of trees (make at least 3 plots to see the behaviour of the algorithm regardless of the stochastic effects).
- The random forest algorithm is known as an ensemble method ... why?