# COMP 140: Stock Prediction (Writeup)

Due on October 11, 2017 at 8:00 PM

*Rixner*

**Joel Abraham**

# 1 Recipe

The inputs of the markov_chain() function are a sequence of numbers assigned to the variable *data* (corresponding to the bins 0-3) and a number greater than or equal to 1 assigned to the variable *order*, representing the order of the desired markov chain. The function outputs an $n^{th}$ order markov chain, where $n = order$ represented as a mapping between $n$-tuples (corresponding to the previous $n$ states) and a mapping between integers (representing a certain possible next state) and real numbers between 0 and 1 (representing the probability of reaching the corresponding state). ***Note that the notation $f(x)$ will be used throughout this assignment to refer to the value that $x$ maps to under the mapping $f$.***

```
procedure MARKOV_CHAIN(data, order)
    model ← empty mapping
    arr ← empty sequence of integers
    index ← −1
    for all element ∈ data do
        index ← index + 1
        if index < order then
            arr ← arr + element
        else
            tup ← tuple with elements of arr
            if tup is not in the domain of model then
                model(tup) ← empty mapping between integers and real-valued probabilities
            end if
            if tup is not in the domain of model then
                model(tup)(element) ← 0
            end if
            model(tup)(element) ← model(tup)(element) + 1
            arr ← arr + element
            remove first element of arr
        end if
    end for
    for all n-tuples entry in the domain of model do
        total ← 0.0
        for all integers state in the domain of model(entry) do
            total ← total + model(entry)(state)
        end for
        for all integers state in the domain of model(entry) do
            model(entry)(state) ← model(entry)(state)/total
        end for
    end for
    return model
end procedure
```

The predict function takes the following as inputs: markov model, *model*, represented as a mapping between tuples and mappings between integer valued states and real-valued probabilities; a sequence of integers, *prev*, representing the previous states with length $n$ where $n$ is the order of the markov model, *model*; an integer, *num*, representing the number of future states to predict. The predict function outputs a sequence of integers of length *num*, representing the predicted states for the next *num* days.

```
procedure PREDICT(model, last, num)
    prev ← copy of last
    predicted ← empty sequence of integers
    index ← −1
    while index < num do
        next ← 0
        tup ← tuple with elements of last
        if tup is not in the domain of model then
            next ← random integer between 0 and 3, inclusive
        else
            entry ← model(tup)
            total ← 0
            random_number ← random real value between 0 and 1
            for all integers state in the domain of entry do
                total ← total + entry(key)
                if random_number <= total then
                    next ← key
                    break
                end if
            end for
        end if
        predicted ← predicted + next
        prev ← prev + next
        remove first element of prev
    end while
    return predicted
end procedure
```

## 2 Discussion

Discuss the results of the experiments you ran in 4.B (be sure to include the results). In particular, discuss the following questions:

1. What is the order of the model that works best for each stock/index? If the orders are not the same, discuss why that might be the case.
2. Which stock/index can you predict with the lowest error? Based on the plots of the day-to-day change in stocks and the histogram of bins, can you guess why that stock/index is easiest to predict?
3. Given that we have divided the day-to-day price change into 4 bins, how many possible states are there in an $n^{th}$ order Markov chain for predicting the change in stock price?
4. The training data we gave you covers two years of data, with 502 data points per stock/index. Is that enough data that it is possible to see all possible states in an $n^{th}$ order Markov chain? What are the constraints on $n$? How do you think it would affect the accuracy of the model if there were not enough data?

**Solution**

1. The model predicts with the least error for stock FSLR with order 5, for stock GOOG with order 1, and for stock DJIA with order 3. This is likely since highly volatile stocks require higher order markov models in order to predict accurately, as low order markov models would be skewed by noise; only as order increases does the impact of noise decrease. Thus, since FSLR is more volatile than DJIA which is more volatile than GOOG (based on daily change and variations in the bin histogram), it is logical that the model for FSLR required a higher order than that for DJIA which required a higher order than that for GOOG.
2. GOOG is predicted with the lowest error, since it is the most stable of the three stocks given. GOOG has the most homogenous bin histogram as well as the least volatile daily change. As such, since stability implies predictability, a given state for GOOG is likely to be similar to the previous state, which is easily predictable by a markov model.
3. Since the model is of order $n$, a state is represented by a tuple with $n$ values, where each value has 4 possibilities. Thus, there are $4^n$ possible states.
4. THere is theoretically enough data to see all possible states in a $3^{rd}$ order Markov chain, but not for a $5^{th}$ order markov chain. Thus, for $n \leq \frac{\log 502}{\log 4} \approx 4$ we can see all possible states. Without enough data, the model would be inaccurate since it would predict certain next states with high certainty based off of insufficient empirical data, leading to false confidence.