# INFO 8000 FINAL PROJECT

Professor: Kyle Johnsen

Student: Lei Lou

12/07/2017

## Introduction

Resveratrol is a stilbenoid, a type of natural phenol and a phytoalexin produced by several plants when the plant is under attack by pathogens such as bacteria or fungi. Resveratrol is widespread in our daily food such as grapes, peanuts and so on. This substance is so popular among the research area until now is mainly because Dipak Kumar Das's fraud that shows it can help cure some disease. While it is not confirmed to have any efficacy yet.

In this final project, we are going to accomplish two tasks.

First fart is to do a clustering among 23 given proteins, we decide to use the Hierarchical method to do it because of the following reasons. First, hierarchical can provide the whole tree-like structure so that we can backtrace to find out the sweet spot. Second, K-means is better at handling larger datasets, since it's result is too random on small datasets. Third, K-NN may get slow in high dimensions. We take total amount of AAs and atoms, the percentage of each AA and C,O,N,S atoms, the volume of each protein as our pointed features. We use 3-D grid to approximate volume calculation and then light up the point near each object,finally we calculate the lit point to get the volume.

In the second part we want to do a prediction in the a prediction model to classify proteins into resveratrol- or DNA- binding proteins. The main feature we use is the sequence of amino acid. Two proteins may be significantly different from each other due to their unlike arrangement. The different combination of proteins also attribute to distinct function. The following arrangement of this paper is algorithm, model selection, result, and discussion.

# Algorithm

## Data Processing

We used python 3 to parse the PDB file. We put our PDB file into two directories: res and dna. All the PDB files related to resveratrol are located in the “res” directory, while the PDB files of DNA's are in “dna” directory. The processes of open the file are similar in both tasks.

```
for filename in os.listdir("res"):
    with open("res/" + filename) as f:
        for line in f:
            if re.match("^ATOM.*", line):
                ...
```

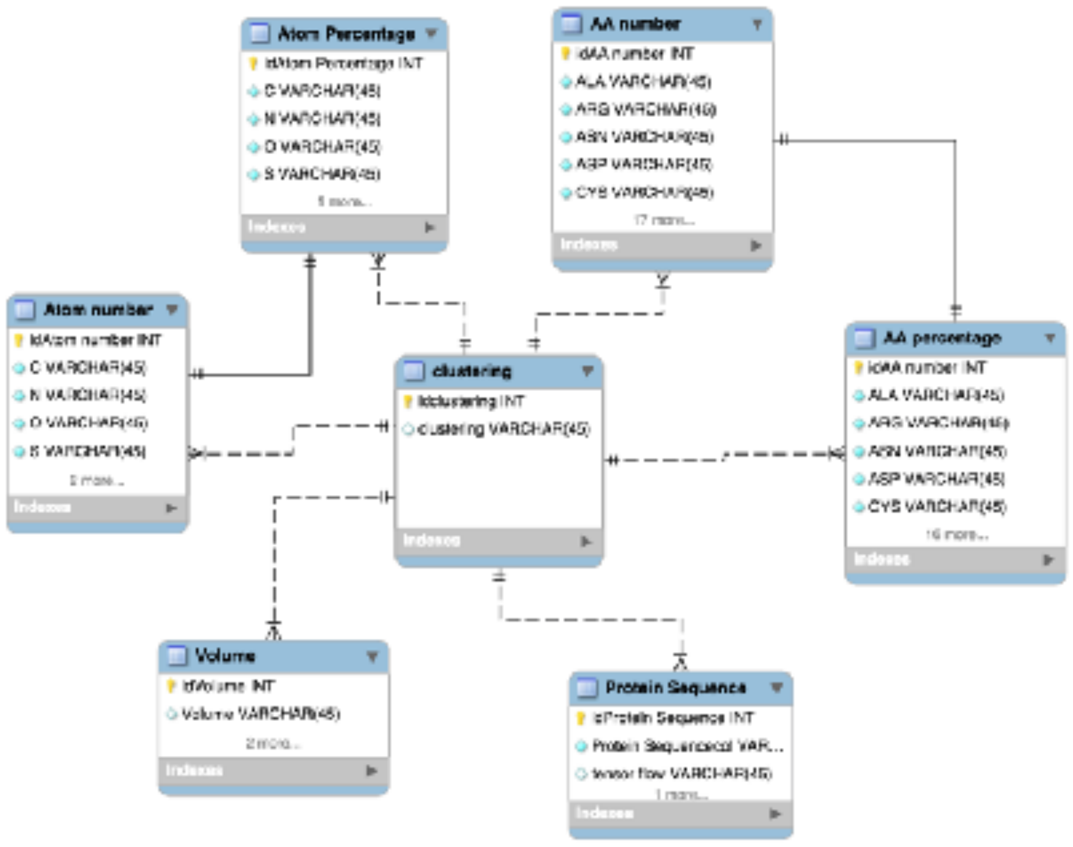
## Feature Selection

### Part 1

In task one, we chose a total number of 27 features, including 4 atom(C,O,N,S) and 20 amino acids. The names of features and the reason that we chose them are as the table below.

| | |
|---|---|
| **Total Number of Atom** | If two proteins are so different from each other, there’s a chance that the total number are total different. |
| **Total Number of Amino Acid** | Amino acids are the component of proteins. So to tell the difference between two proteins, total number of amino acid may provide some essential information. |

| | |
|---|---|
| **Atom Percentage** | We chose four atoms and calculated their percentage (Carbon, Oxygen, Nitrogen and Sulfur). Some PDB files may ignore Hydrogen so we don't use it. |
| **Amino Acid Percentage** | We calculated the percentage for all 20 amino acids. Proteins in the same category usually have similar amino acid in order to perform similar functionality. |
| **Volume** | At first we wanted to use shape as a feature. But that requires knowledge of 3D model comparison. So we choose the volume. Proteins performs similar functionalities usually would have the similar size. |



### Part 2

We only used one feature in task 2, which is the sequence of amino acid. Because protein is generated one by one from amino acid, so the sequence of amino acid can reveal information about the protein, such as structure, functionality and more. We tried to follow the principle of nature to learn how to classify proteins. When we parsed the PDB file, we appended the amino acid one by one to the end of the list. Later we fed the list to our machine learning model to train and verify the result.

## Model Selection

### Part 1

We used hierarchical clustering method as our model for task one. Hierarchical can provide a complete tree structure of the data, so that we can find out where the breakpoint is. The reason we didn't choose K-means due to the fact K-means is better at handling larger datasets. In our project, the dataset is too small so if we used K-means, then the dots would be too distributed to tell whether they are truly belonged to the same group. K-NN is the model that will compute the distance of each two data. If we want to expand our datasets in the future, this model would not be effective enough.
Part of the core codes are listed below.

```
name = []
attr = []
for filename in os.listdir("res"):
    with open("res/" + filename) as f:
        name.append(filename[:-4])

        feature = [0] * 27
        old_resSeq = ""
        AMINOACID = {"ALA":6, "ARG":7, "ASN":8, "ASP":9, "CYS":10,
                "GLU":11, "GLN":12, "GLY":13, "HIS":14, "ILE":15,
                "LEU":16, "LYS":17, "MET":18, "PHE":19, "PRO":20,
                "SER":21, "THR":22, "TRP":23, "TYR":24, "VAL":25}
        grid = []
        for line in f:
            if re.match("^ATOM.*", line):
                if line[21] == "B":
                    continue
                feature[0] += 1
                resSeq = line[22:26]
                if resSeq != old_resSeq:
                    old_resSeq = resSeq
                    feature[1] += 1
                if line[76:78].strip() == "C":
                    feature[2] += 1
                if line[76:78].strip() == "O":
                    feature[3] += 1
                if line[76:78].strip() == "N":
                    feature[4] += 1
                if line[76:78].strip() == "S":
                    feature[5] += 1
                if len(line[17:20].strip()) == 3:
                    feature[AMINOACID[line[17:20].strip()]] += 1
                grid.append(str(int(float(line[30:38].strip())/10))
+","+str(int(float(line[38:46].strip())/10))+","+str(int(float(line[46:54].strip())/10)))
        grid = list(set(grid))
        feature[26] = len(grid)

        attr.append(feature)

hierarchical(name, attr)
```

## Part 2

As we mentioned above, we value the sequence of amino acids. However, not all the machine learnings are capable of handling

sequencing data. That's why we implemented Long Short-Term Memory(LSTM).
LSTM belongs to neural network. But instead of being fed by a list of un-sequenced data, LSTM accepts a string of sequenced data. This characteristic is been given because of the "memory" state in the LSTM cell. This unique characteristic makes our goal doable.

```
with tf.Session() as sess:
    sess.run(init)

    for step in range(1, training_steps + 1):
        batch_x, batch_y, batch_seqlen = dataSets.train_next(batch_size)

        # Run optimization op (backprop)
        sess.run(optimizer, feed_dict={x: batch_x, y: batch_y, seqlen: batch_seqlen})

    print("Optimization Finished!")

    # Do the test
    test_data, test_label, test_seqlen = dataSets.getTestData()
    print("Testing Accuracy:", sess.run(accuracy, feed_dict={x: test_data, y: test_label, seqlen:
test_seqlen})
```

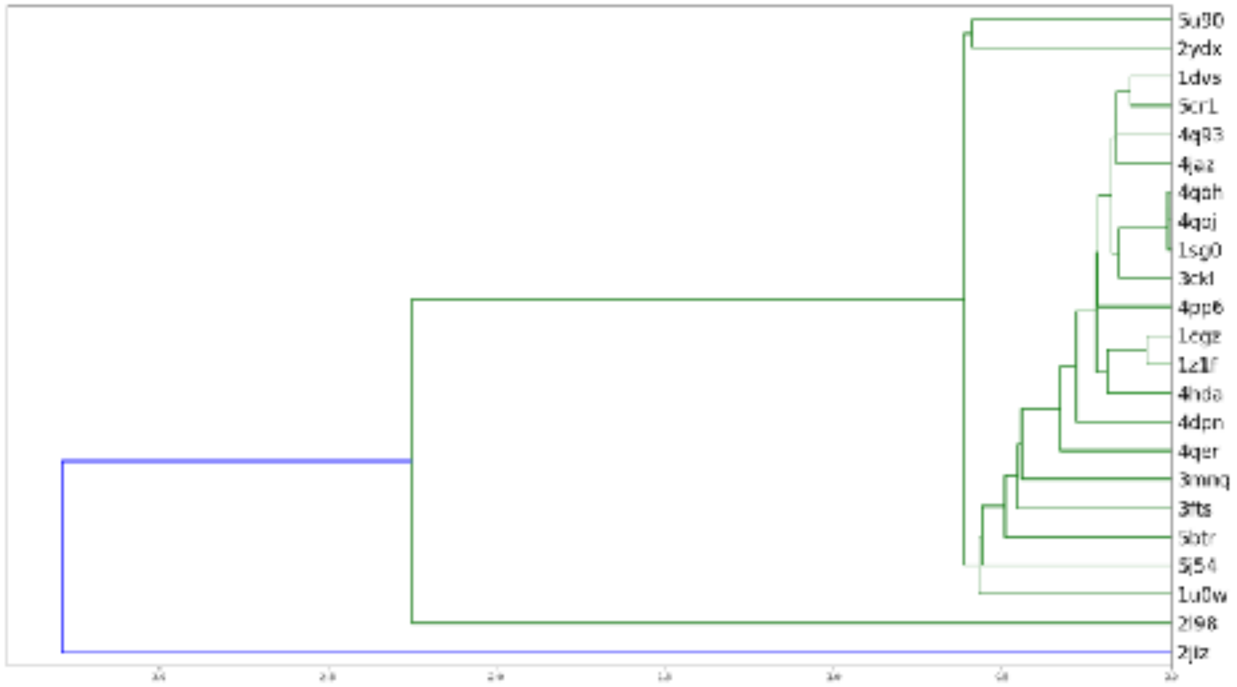
# Results and Discussion

For both parts, the programs are running in the following system specifications.

- CPU: Intel i7-6800K @ 3.40GHz
- Memory: 16GB
- GPU: Nvidia 1080 Ti 11GB
- OS: Ubuntu 16.04
- Machine Learning Framework: TensorFlow

## Results

### Part 1



X-axis represents the distance between two nodes.

Y-axis represents the names of the proteins.
So we can tell from the graph, distance around 0.6 is where big difference begin to appear. So to get the right cluster, we probably need to cut the tree around 0.6.

### Part 2

The accuracy of the training result depends on how many iterations the model is trained and how fast the model learns. The test accuracy in our three runs are listed below.

| | **Run 1** | **Run 2** | **Run 3** |
|---|---|---|---|
| **Iterations** | 10,000 | 100,000 | 1,000,000 |
| **Learning rate** | 0.001 | 0.0001 | 0.0001 |
| **Test Accuracy** | 60% | 80% | 80% |

## Performance

For task 1, because hierarchical method is pretty fast and also our datasets are small, the result generated immediately.
But for task 2, the time is highly depended on how well the the model is trained and how fast the model learns. The table below shows the time in our three runs.

| | **Run 1** | **Run 2** | **Run 3** |
|---|---|---|---|
| **Iterations** | 10,000 | 100,000 | 1,000,000 |
| **Learning rate** | 0.001 | 0.0001 | 0.0001 |
| **Time** | ~4 minutes | ~27 minutes | ~4 hours 20 minutes |

## Validation

The validation is only been used in task 2. The process of the validation is processed by TensorFlow. As we proceeded in training, the accuracy of validation increases meaning the model in learning. However at some point, the accuracy stops around 83%, which we think is the limit of LSTM model on this small datasets. LSTM may perform better given a larger datasets.

## Test Accuracy

Test accuracy is only used in task 2. We divided the 46 datasets into 36 training set and 10 testing set according to the ratio of 80/20. We did the test after the training process and feed the model with 10 testing set which is unknown to it.
The accuracy is 80% on the testing.
The main reason we think that caused the failure is small datasets, which didn't give LSTM enough knowledge to learn about the sequencing. And also, the LSTM we used only read first 200 amino acids. We could let it reads more but more feeding means more physical memory and in our system, we could not handle the whole sequence which has about 15 thousand amino acid.

## Future Work

- **Expand datasets to more proteins**
    - In this project, there are only 23 resveratrol binding proteins, which is a small number. Therefore, our algorithm can carry out a reasonable and efficient result. To verify our methodology more, a data set with more proteins is required.
- **Can use the "shape" as one of the features**
    - At first, we tried to use the shape as one features. However, it is complicated to define the shape from the raw data. We might need to construct the proteins to figure out the 3D outline. Even after we understand the shape of certain atom,

it is still difficult to define the score of shape and which shapes are similar to each other.

- **Electronic charge**

    - Charged group of protein function as an important role in the protein interaction, such as  the formation of protein–protein and protein–nucleic acid complexes, denaturation of proteins at high and low pH values, and association of receptors with charged ligands. It illustrated the electronic charge is the basic physical characteristic and it should be considered.

- **Adding the sequence of AA**

    - The machine learning we use is the atom level, we can also change it to the AA level, and maybe later the second structure level. And this will help ensurance the accuracy since it is in the combination of different level of consideration which we usually research a protein's structure like this.

# Supplementary material

## Part 1

```
import os
import re

from scipy.cluster import hierarchy
import matplotlib.pyplot as plt
from sklearn.preprocessing import normalize
import numpy as np

'''
attr has following features
    0. atom count
    1. aa count
    2. atom C %
    3. atom O %
    4. atom N %
    5. atom S %
    6 - 25. aa %
    26. volume
'''
def main():
    name = []
    attr = []
    for filename in os.listdir("res"):
        with open("res/" + filename) as f:
            name.append(filename[:-4])

            feature = [0] * 27
            old_resSeq = ""
            AMINOACID = {"ALA":6, "ARG":7, "ASN":8, "ASP":9, "CYS":10,
                         "GLU":11, "GLN":12, "GLY":13, "HIS":14, "ILE":15,
                         "LEU":16, "LYS":17, "MET":18, "PHE":19, "PRO":20,
                         "SER":21, "THR":22, "TRP":23, "TYR":24, "VAL":25}
            grid = []
            for line in f:
                if re.match("^ATOM.*", line):
                    if line[21] == "B":
                        continue
```

## Part 2

```
from __future__ import print_function

import tensorflow as tf
import os
import re
import random


class fetchData(object):
    def __init__(self, ratio = 0.8):
        self.wholedata = []
        self.trainset = []
        self.testset = []
        self.maxSeqLen = 0
        self.uniqueWord = []
        limit = 200
        for filename in os.listdir("res"):
            with open("res/" + filename) as f:
                old_resSeq = ""
                aaseq = []
                for line in f:
                    if re.match("^ATOM.*", line):
                        if line[21] == "B":
                            continue
                        resSeq = line[22:26]
                        if resSeq != old_resSeq:
                            old_resSeq = resSeq
                            if len(line[17:20].strip()) == 3:
                                aaseq.append(line[17:20])
                self.maxSeqLen = max(len(aaseq), self.maxSeqLen)
                if len(aaseq) > limit:
                    aaseq = aaseq[:limit]
                self.wholedata.append(self.dataSet(aaseq, [1., 0.], len(aaseq)))
                self.uniqueWord += aaseq
        for filename in os.listdir("dna"):
            with open("dna/" + filename) as f:
                old_resSeq = ""
                aaseq = []
                for line in f:
                    if re.match("^ATOM.*", line):
                        if line[21] == "B":
                            continue
                        resSeq = line[22:26]
```

## Links

https://www.rcsb.org/pdb/home/home.do
https://www.tensorflow.org
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
http://www.ks.uiuc.edu/Research/vmd/
https://www.schrodinger.com/maestro