Assignment 2 CS170: Introduction to Artificial Intelligence

Andrew Pirelli Dr. Eamonn Keogh
SID# 861223915
apire001@ucr.edu
December 17th, 2018

In completing this assignment I consulted:

- Python 3.7.1 documentation. Here is the URL of the table of contents: https://docs.python.org/3/
- Numpy documentation, found here: http://www.numpy.org/
- Project 2 briefing and evaluation slides.
- Some of my previous CS171 (machine learning) assignments in order to help myself implement the algorithms and create the graphs.
- The class website https://www.cs.ucr.edu/~eamonn/cs170/ in order to find and download my assigned datasets for this project.

All important code is original. Unimportant subroutines that are not completely original are…

- Numpy matrix data structures and functions, imported from the numpy library
- Plotting functions, imported from the matplotlib.pyplot
- Some built in python functions for arrays, such as remove and append.

First page of code:

```
#libraries I need
import numpy as np
# function to load the data
def loadsparsedata(fn):
    X = np.loadtxt(fn)
    Y = X[:,0]
    Y = Y[:,np.newaxis]
    X = X[:,1:]
    return X, Y
#nearestneighbor function using euclidean distance
def nearestneighbor(X, Y, x):
    m, n = X.shape
    mindist = np.linalg.norm(x - X[0])
    mindex = 0
    for i in range(m):
        dist = np.linalg.norm(x - X[i])
        if(dist < mindist):
            mindist = dist
            mindex = i
    return Y[mindex]
#forward search functions
#looking to find 2 strong, 1 weak feature
#returns set of 5 best features
def forwardsearch(X,Y,fset,depth):
    if(depth >= 10):
        return fset
    else:
        print('On level ' + str(depth+1) + ' of the forward search tree')
        m, n = X.shape
        bestacc = 0.0
        bestfeat = -1
        maxerr = m
        for i in range(n):
            tempset = np.copy(fset)
            if i not in fset:
                print('--Considering adding the ' + str(i) + ' feature')
                tempset = np.append(tempset, i)
                trainX = generatematrix(X,Y,tempset)
```

Last Page of Code:

```
#third algorithm
def myalgorithm(X):
    m,n = X.shape
    numappear = np.zeros((n-1))
    for i in range(10):
        tempX = X
        np.random.shuffle(tempX)
        tempX = tempX[20:]
        tempY = tempX[:,0]
        tempY = tempY[:,np.newaxis]
        tempX = tempX[:,1:]
        emptyset = ([])
        fset,fsearch = forwardsearch(tempX,tempY,emptyset,numfeatacc,0)
        for j in range(len(fset)):
            a = int(fset[j])
            numappear[a] = numappear[a] + 1
    return numappear
#for my algorithm data and graphs
relX = np.append(a,b,1)
relX = np.append(relX,c,1)
ae = loocv(a,TrainY,200)
be = loocv(b,TrainY,200)
ab = np.append(a,b,1)
abe = loocv(ab,TrainY,200)
ce = loocv(c,TrainY,200)
relXe = loocv(relX,TrainY,200)
print('Feature 55 accuracy ' + str(ae))
print('Feature 40 accuracy ' + str(be))
print('Feature 46 accuracy ' + str(ce))
print('Feature 55 40 accuracy ' + str(abe))
print('Feature 55 40 46 accuracy ' + str(relXe))
defaultrate = np.count_nonzero(TrainY == 2)/m
print('Default rate is ' + str(defaultrate))
```

Note: My final program is about 3 pages of code consisting of helper functions, plots, search functions and other functions.

CS170_SMALLtestdata__79.txt Forward Search Printout (As specified by the Project 2 Briefing):

On level 1 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 1 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
--Considering adding the 7 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
On level 1 added feature 5 to current set to get accuracy 0.855
On level 2 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 1 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
--Considering adding the 7 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
On level 2 added feature 2 to current set to get accuracy 0.995
On level 3 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
--Considering adding the 7 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
On level 3 added feature 8 to current set to get accuracy 0.975
On level 4 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
On level 4 added feature 9 to current set to get accuracy 0.92
On level 5 of the forward search tree

--Considering adding the 0 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
--Considering adding the 9 feature
On level 5 added feature 10 to current set to get accuracy 0.89
On level 6 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
On level 6 added feature 7 to current set to get accuracy 0.89
On level 7 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 5 feature
On level 7 added feature 6 to current set to get accuracy 0.85
On level 8 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
On level 8 added feature 3 to current set to get accuracy 0.81
On level 9 of the forward search tree
--Considering adding the 0 feature
--Considering adding the 3 feature
On level 9 added feature 1 to current set to get accuracy 0.79
On level 10 of the forward search tree
--Considering adding the 3 feature
On level 10 added feature 4 to current set to get accuracy 0.795

CS170_SMALLtestdata__79.txt Backward Search Printout (As specified by the Project 2 Briefing):

On level 10 of the backward search tree
--Considering removing the 1 feature
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 4 feature
--Considering removing the 5 feature
--Considering removing the 6 feature
--Considering removing the 7 feature
--Considering removing the 8 feature
--Considering removing the 9 feature

--Considering removing the 10 feature
On level 10 removed feature 10 to current set to get accuracy 0.81
On level 9 of the backward search tree
--Considering removing the 1 feature
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 4 feature
--Considering removing the 5 feature
--Considering removing the 6 feature
--Considering removing the 7 feature
--Considering removing the 8 feature
--Considering removing the 9 feature
On level 9 removed feature 1 to current set to get accuracy 0.84
On level 8 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 4 feature
--Considering removing the 5 feature
--Considering removing the 6 feature
--Considering removing the 7 feature
--Considering removing the 8 feature
--Considering removing the 9 feature
On level 8 removed feature 8 to current set to get accuracy 0.855
On level 7 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 4 feature
--Considering removing the 5 feature
--Considering removing the 6 feature
--Considering removing the 7 feature
--Considering removing the 9 feature
On level 7 removed feature 7 to current set to get accuracy 0.88
On level 6 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 4 feature
--Considering removing the 5 feature
--Considering removing the 6 feature
--Considering removing the 9 feature
On level 6 removed feature 9 to current set to get accuracy 0.88
On level 5 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 4 feature
--Considering removing the 5 feature

--Considering removing the 6 feature
On level 5 removed feature 4 to current set to get accuracy 0.92
On level 4 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 5 feature
--Considering removing the 6 feature
On level 4 removed feature 6 to current set to get accuracy 0.94
On level 3 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 3 feature
--Considering removing the 5 feature
On level 3 removed feature 3 to current set to get accuracy 0.995
On level 2 of the backward search tree
--Considering removing the 2 feature
--Considering removing the 5 feature
On level 2 removed feature 2 to current set to get accuracy 0.855

CS170 Assignment 2 Write Up

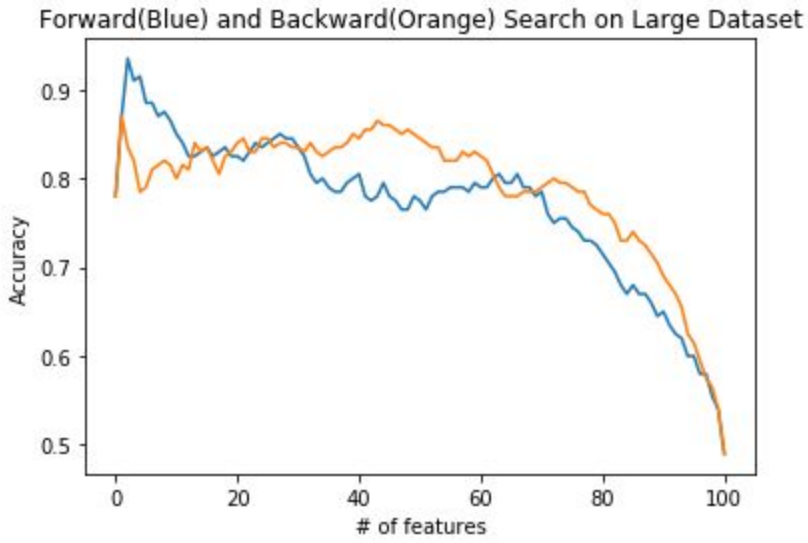
Andrew Pirelli, SID 861223915

Development Challenges:

Learning to use the proper numpy operations and functions in order to split and merge different parts of the matrices needed in order to perform the accuracy evaluations for the search functions was fairly challenging. I had many bugs which were very simple in hindsight, but took me a disproportionate amount of time to find and fix because I was unfamiliar with splitting and merging of matrices in Python. However, after completing this project I feel much more comfortable with matrix operations and broadcasting in Python, which will likely be a very useful skill in the future.

Forward and Backward Search Algorithm Analysis:

- I implemented both the forward and backward searches as specified in the project 2 briefing slides. Their printouts can be seen in the earlier pages of this report.
- For calculating accuracy, I used leave one out cross validation. This is the most extreme form of the k-fold cross validation, where k equals the number of objects in the dataset.
- The forward and backward search performed similarly on the small dataset, as seen above.
- However, the forward search proved to be much more useful than the backward search on the larger dataset. This was because I used nearest neighbor with euclidean distance as my evaluation function, meaning that all features were weighted equally (by distance) in determining the classification of the test object passed into the function. So when many features were being considered for the accuracy, the features which were actually relevant to correctly predicting the classification had a far smaller impact on the prediction for the test object.
- This lead to the backward search not being able to find one of the strong features. Since the forward search considers the impact of each feature individually, it had no trouble finding both of the strong features in both datasets. However it still had trouble finding the weaker features with a single search. I explain this further in the My Algorithm section and how I solved this problem.



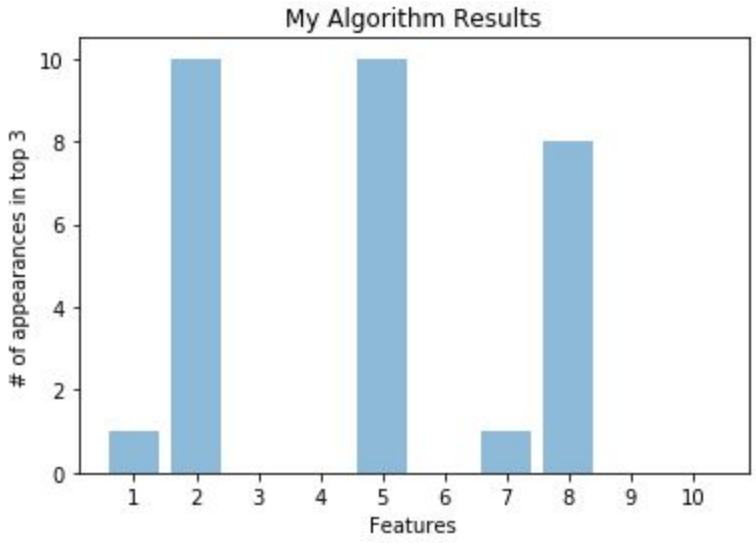
- The above figure shows the backward search on the large dataset missing one of the strong features, while the forward search did not miss any of the strong features. This is shown by the large difference in accuracy when the number of features in the search equals two.

My Algorithm:

- In order to find the weaker feature in both datasets, I needed a better algorithm than simply the forward or backward search.
- The problem was that the weaker feature's ability to classify the objects was sometimes overshadowed by the randomness of the useless features for the full set of data. This made the randomness of the useless features seem more useful than the weaker feature.
- To fix this, I decided to create an algorithm that would randomly shuffle the data, then trim the first 10% of examples from the training set.
- I then repeated the process above 10 times in order to see which features were best at classifying the data, then ranked them based on the number of top 3 by accuracy appearances they had for the small dataset, or number of top 5 by accuracy appearances they had in the large dataset.
- From there, I was able to determine which features were random/useless, and which feature was the weak feature that I was looking for.
- The graphics for these rankings are below the relevant features sections.

Relevant Features For CS170_SMALLtestdata__79.txt:

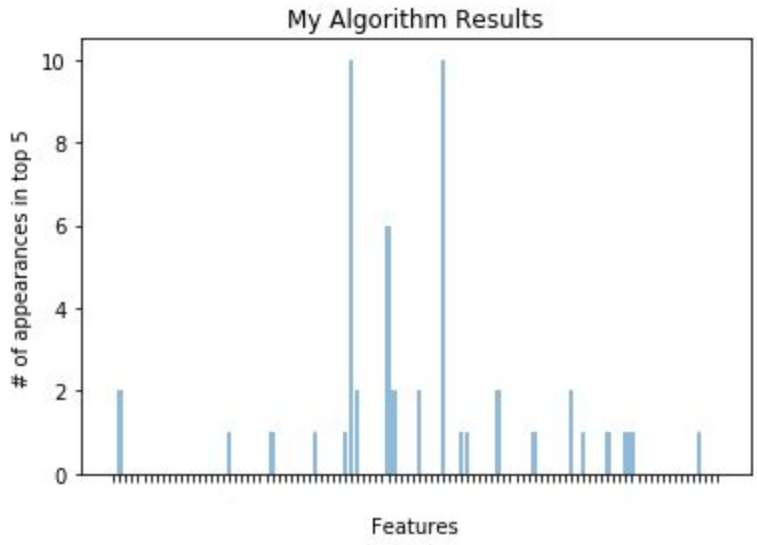
- I am assuming that the features are numbered 1 to n, as shown in the slides.
- For context, the default rate on this dataset is 0.875



- The bar graph showing the number of appearances in the top 3 per each feature over 10 iterations can be seen above.
- The strong features are [5 2], this shown by their accuracies from the forward/backward searches, as well as from my algorithm:
    - Features [5] and [2] consistently appeared 10 times in the top 3 features by accuracy over the 10 iterations on the randomly 10% trimmed datasets when I executed my algorithm.
    - The feature set [5] has an accuracy of 0.855
    - The feature set [2] has an accuracy of 0.815
    - The feature set [5 2] has an accuracy of 0.995
- The weak feature was [8], according to my algorithm above. This is because:
    - Feature [8] had the most appearances in the top 3 features by accuracy over the 10 iterations on the randomly 10% trimmed datasets, at 8 appearances.
    - The next most likely features to be the weak feature have only 1 appearance. This makes it seem that feature [8] is almost certainly the weak feature in this dataset.
    - The feature set [8] has an accuracy of 0.76
    - The feature set [5 2 8] has an accuracy of 0.975

Relevant Features For CS170_LARGEtestdata__11.txt:

- I am assuming that the features are numbered 1 to n, as shown in the slides.
- For context, the default rate on this dataset is 0.78



- The bar graph showing the number of appearances in the top 3 per each feature over 10 iterations can be seen above. Unfortunately I could not figure out how to properly scale the feature numbers at the bottom. However the values on the bar graph are still correct, and the features are in ascending order (1, 2, 3, ...).
- The strong features are [55 40], this shown by their accuracies from the forward/backward searches, as well as from my algorithm:
    - Features [55] and [40] consistently appeared 10 times in the top 3 features by accuracy over the 10 iterations on the randomly 10% trimmed datasets when I executed my algorithm.
    - The feature set [55] has an accuracy of 0.87
    - The feature set [40] has an accuracy of 0.705
    - The feature set [55 40] has an accuracy of 0.935
- The weak feature was [46], according to my algorithm above. This is because:
    - Feature [46] had the most appearances in the top 3 features by accuracy over the 10 iterations on the randomly 10% trimmed datasets, at 6 appearances.
    - The next most likely features to be the weak feature have only 2 appearances. This makes it seem that feature [46] is almost certainly the weak feature in this dataset.
    - The feature set [46] has an accuracy of 0.695
    - The feature set [55 40 46] has an accuracy of 0.91

Conclusion:

- The forward search is much more practical than the backward search on larger datasets where the majority of features are random and useless. However, both searches performed similarly on the smaller dataset.
- The weak feature was still not obvious in the forward search due to the randomness of the useless features making them seem better on that specific dataset. I minimized this problem in my algorithm by testing 10 subsets of the original dataset where a random 10 percent of the original data was removed. I then counted top appearances by accuracy to get a better idea of what the weaker feature was.
- This also decisively showed which features were the strong features, as they had a 100 percent appearance rate in the top features by accuracy on both datasets.
- The strong features for CS170_SMALLtestdata__79.txt were [5 2], while the weak feature was [8].
- The strong features for CS170_LARGEtestdata__11.txt were [55 40], while the weak feature was [46].