Project Report **Abhishek Srivastava**
CS205: Artificial Intelligence Student Id: 861307778

Problem Statement

Nearest neighbor algorithm is very simple, yet very competitive classification algorithm. But it is very sensitive to irrelevant features which can degrade the accuracy of the classifier.
Given a Dataset we have to find out the most importance features present in the data. We have to use following searches/methods to find out relevant features.

- Forward Selection
- Backward Elimination
- Our Original Algorithm.

Feature Selection

Feature selection is the process of selecting a subset of relevant features for the use of model construction.[1]

Feature selection techniques are used for following reasons:

- simplification of models to make them more interpretable.
- reduce training time.
- reduce overfitting.

In a large data set many features are either redundant or irrelevant and thus can be removed without incurring much loss of information.
Two of the most important methods to select subset of relevant features is:

- Sequential forward selection.
- Sequential backward selection.

Sequential Forward Selection

In this method we start with empty set. Sequentially we add the features which maximizes the accuracy when combined with the features that have already been selected. We do this till all features are selected then we select the subset which resulted in best accuracy.
Implementation
I used **fitcknn** method for knn training, **crossval** for "leave one out" validation and **kfoldloss** to calculate the loss which is subtracted by 1.00 to give the accuracy of the classifier.

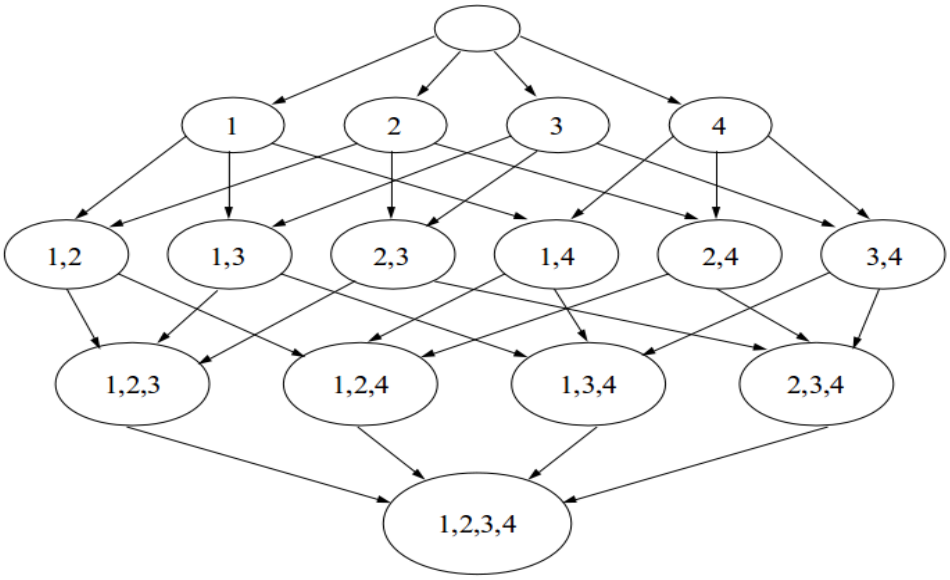

Figure 1: Possible path in feature selection.

Sequential Backward Selection

This method is similar but performed exactly opposite. we start with all features selected. Sequentially we remove each features and check the accuracy and select that which maximizes the accuracy when it is removed from the set. We do this till features set is empty and then we select the subset which gave the best accuracy.

Implementation

Same as in forward selection i used **fitcknn** method for knn training, **crossval** for "leave one out" validation and **kfoldloss** to calculate the loss which is subtracted by 1.00 to give the accuracy of the classifier. I used **horzcat** method as well for the concatenation of two matrices in horizontal manner.

Figure 1 shows all the possible path we can take during forward and backward selection for data with 4 features.

Abhishek's Solutions

Even though nearest neighbor is quite powerful but it is very slow because of number of comparisons done each points for classification. Computational cost grows exponentially with the increase in dimensions. To overcome this issue I utilized the power of decision trees, I choose boosting for this project.

I created decision tree for the whole data and then choose top 5 features of decision tree which it is using to make decisions. Then I calculated accuracy of each of the features and then selected top 2 features with better accuracy, since our data is strongly correlated to two features.

Implementation

I used **fitensemble** method to train a decision tree using X & Y. Then I used **predictorImportance** method to find out all the important decisions used in decision tree which gives a floating number for all the features. After sorting that array in descending order I picked top 5 decisions and then using **fitcknn** I calculated the accuracy of each features and then sorted the accuracy in descending order and picked top 2 features and calculated accuracy of that

Table 1: Performance of all 3 search algorithms in seconds.

File Name	Forward Selection	Backward Elimination	Abhishek Algorithm
cs_205_small52	44.394922	45.179802	5.628990
cs_205_small64	45.445308	45.206639	5.671263
cs_205_small65	45.423482	45.278004	6.011677
cs_205_large52	3587.343066	3688.891623	5.844109
cs_205_large64	3717.966384	3889.944463	5.880867
cs_205_large65	3626.637648	3726.015276	5.801674

subset.[2, 3]

Explanation

I tried to implement a method which reduces the time to find feature subset substantially thats why I picked boosting(**AdaBoost** to be specific). Then I used boosting to find the important features. The reason for improvement in time can be seen with respect to the forward selection and backward elimination is due to reduction in training decision trees. Decision trees complexity does not increases with the increase in features in data. **AdaBoost** training process selects only those features known to improve the the model, reducing dimensionality and potentially improving execution time as irrelevant features do not need to be computed. [4]

Table 1 shows the time comparison between forward selection, backward elimination and my implemented algorithm. And it is clear that my algorithm improves time for both small and large data set substantially.

Testing & Analysis

Testing

For this project I evaluated my implementation of all 3 algorithms on 6 different datasets: **cs_205_small52**, **cs_205_small64**, **cs_205_small65**, **cs_205_large52**, **cs_205_large64**, **cs_205_large65**. I evaluated feature selected after running the algorithms, accuracy of the final feature selected and time taken to complete the specific algorithm.

Time Analysis

For all the small data sets with 10 features Forward selection takes an average of approx 45 seconds to run, Backward elimination takes about 45 seconds to select features. My algorithm for all 3 small datasets takes about 6 seconds to do feature selection.

For the large data sets with 100 features time increases exponentially. Forward selection takes on an average about 1 hour to complete feature selection. Backward selection also takes about 1 hours and 15 minutes to complete on average. So backward elimination takes a little more time for feature selection than forward selection. My algorithm performs the same as it for the small datasets, it takes on average 6 seconds to do feature selection.

Table 1 shows the performance of all 3 search algorithms ran on 6 different datasets.

Feature & Accuracy Analysis

For the small dataset forward feature selection resulted in subset of 2-3 features for all the data sets. For backward elimination dataset 52 & 64 resulted in subset of 2 features whereas dataset 65 gave subset of 9 features. My algorithm only gives subset of 2 best features since we have assumed that 2 feature is strongly co-related with the results.

For large dataset forward selection resulted in subset of different lengths dataset 52 gave subset

Table 2: Features Selected by 3 search algorithms with accuracy result.

File Name	Forward Selection	Backward Elimination	Abhishek Algorithm
cs_205_small52	{10, 2, 3 }(92.0 %)	{3, 10 }(90.0 %)	{10, 6 }(84.0 %)
cs_205_small64	{6, 10 }(92.0 %)	{6, 10 }(92.0 %)	{6, 10 }(92.0 %)
cs_205_small65	{4, 8 }(98.0 %)	{1, 2, 4, 5, 6, 7, 9, 10 }(82.0 %)	{4, 9 }(80.0 %)
cs_205_large52	{21, 38, 50, 62, 74, 37 }(94.0 %)	{1, 3, 4 . . . 98, 99 }(91.0 %)	{38, 62 }(96.0 %)
cs_205_large64	{2, 82, 67, 87, 55 }(93.0 %)	{8, 11, 37 . . . 96, 99 }(93.0 %)	{82, 46 }(80.0 %)
cs_205_large65	{50, 16, 71 }(95.0 %)	{9, 12, 19 . . . 97, 98 }(97.0 %)	{50, 5 }(87.0 %)

of 6 features, dataset 64 gave subset of 5 features and dataset 65 gave subset of 3 features. For backward elimination all large data set resulted in vary large subset of features.

Features selected for **cs_205_large52** from backward elimination: 1, 3, 4, 10, 11, 13, 14, 15, 17, 20, 22, 26, 27, 28, 29, 32, 39, 40, 41, 42, 46, 47, 48, 51, 52, 57, 58, 60, 64, 65, 69, 74, 76, 79, 80, 81, 86, 87, 89, 90, 91, 92, 96, 97, 98, 99

Features selected for **cs_205_large64** from backward elimination: 8, 11, 37, 40, 46, 49, 53, 59, 61, 69, 71, 77, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 94, 96, 99

Features selected for **cs_205_large65** from backward elimination: 9, 12, 19, 21, 26, 29, 31, 43, 47, 49, 55, 56, 57, 58, 67, 70, 71, 72, 77, 79, 83, 85, 86, 88, 89, 90, 91, 95, 96, 97, 98

My algorithm gave 2 features for all large datasets. But the accuracy for all them was in par with the accuracy from forward selection subset and backward elimination subset for dataset 65 & 64 except for dataset 52, where my algorithm took 6 seconds to select feature which gave accuracy of 96% which was better than accuracy of the subsets selected from both forward selection and backward elimination.

Table 2 shows all the features selected for all 3 search algorithms on 6 different datasets.

Final Evaluation

Nearest neighbor algorithm is very powerful tool for the feature selection as well as for classification. But with increase in the dimension computational cost increases exponentially. So in conclusion we should use nearest neighbor algorithm when the number of features is less for the dataset. If the number of features is quite large then we should explore other options for feature selection for example **decision trees, PCA** etc.

Code

```
function feature_selection()
    clc;
    fprintf('Welcome to Abhishek Feature Selection Algorithm\n');
    filename = input('Type the name of the file to test: ','s');
    Data = load (filename);
    Y = Data(:,1);
    X = Data(:,2:end);
```

```
    fprintf('\nType the number of the algorithm you want to run\n');
    fprintf('1) Forward Selection\n');
    fprintf('2) Backward Elimination\n');
    fprintf('3) Abhishek Special Algorithm\n');
    ip = input('Input: ');
    [m n] = size(X);
    fprintf('\n\nThis dataset has %d features (not including the class attribute),
             with %d instances.\n',n,m);
    fprintf('\n');
    fprintf('Please wait while I normalize the data....');
    X_n = normr(X);
    fprintf('Done!\n\n');
    model = fitcknn(X,Y);
    cvmodel = crossval(model,'KFold',m);
    kfloss = kfoldLoss(cvmodel);
    cvacc = 1.00 - kfloss;
    fprintf('Running nearest neighbor with add %d features, using "leaving-one-out" evaluation,
             I get an accuracy of %.1f %%\n\n',n,cvacc*100);
    fprintf('Beginning Search\n\n');
    tic;
    if ip == 1
        forward_selection(X,Y);
    elseif ip == 2
        backward_selection(X,Y);
    elseif ip == 3
        abhi_selection(X,Y);
    end
    toc;
end

function forward_selection(X,Y)
    [m n] = size(X);
    test = 0;
    temp = 0;
    bestf = 0;
    bestacc = 0;
    tempacc = 0;
    for i = 1:n
        mdl = fitcknn(X(:,i),Y);
        cvmdl = crossval(mdl,'KFold',m);
        kloss = kfoldLoss(cvmdl);
        acc = 1.00 - kloss;
        fprintf('Using feature(s) {%d} accuracy is %.1f %%\n',i,acc*100);
        if tempacc < acc
            bestf = i;
            bestacc = acc;
            tempacc = acc;
        end
    end
    fprintf('Feature set {%d} was best accuracy is %.1f %%\n\n',bestf,bestacc*100);
    feature = 0;
    temp = bestf;
    while size(temp,2) < n-1
        tempacc = 0;
        if size(temp,2)==1
            test = X(:,temp(1));
        else
            test = [test X(:,temp(size(temp,2)))];
        end
        for i = 1:n
```

```
            if ~ismember(i, temp(:))
                mdl = fitcknn([test X(:,i)],Y);
                cvmdl = crossval(mdl,'KFold',m);
                kloss = kfoldLoss(cvmdl);
                acc = 1.00 - kloss;
                g = sprintf('%d ', [temp i]);
                fprintf('Using feature(s) {%s} accuracy is %.1f %%\n',g,acc*100);
                if tempacc < acc
                    feature = i;
                    pg = g;
                    tempacc = acc;
                end
            end
        end
        temp = [temp feature];
        if bestacc < tempacc
            bestf = [bestf feature];
            bestacc = tempacc;
        else
            fprintf('\nWarning, Accuracy has decreased!
                        Continuing search in case of local maxima\n');
        end
        fprintf('\nFeature set {%s} was best accuracy is %.1f %%\n\n',pg,tempacc*100);
    end
    bg = sprintf('%d ', bestf);
    fprintf('\nFinished search!! The best feature subset {%s}
                which has accuracy of %.1f %%\n',bg,bestacc*100);
end

function backward_selection(X,Y)
    [m n] = size(X);
    temp = 1:n;
    test = X;
    bestf = temp;
    bestacc = 0;
    mdl = fitcknn(X,Y);
    cvmdl = crossval(mdl,'KFold',m);
    kloss = kfoldLoss(cvmdl);
    acc = 1.00 - kloss;
    ag = sprintf('%d ', temp);
    fprintf('Using feature(s) {%s} accuracy is %.1f %%\n',ag,acc*100);
    bestacc = acc;
    fprintf('\nFeature set {%s} was best accuracy is %.1f %%\n\n',ag,bestacc*100);
    feature = 0;
    while size(temp,2)>1
        tempacc = 0;
        for i = 1:size(temp,2)
            test = horzcat(X(:,1:temp(i)-1),X(:,temp(i)+1:end));
            ttemp = horzcat(temp(:,1:i-1),temp(:,i+1:end));
            mdl = fitcknn(test,Y);
            cvmdl = crossval(mdl,'KFold',m);
            kloss = kfoldLoss(cvmdl);
            acc = 1.00 - kloss;
            bg = sprintf('%d ', ttemp);
            fprintf('Using feature(s) {%s} accuracy is %.1f %%\n',bg,acc*100);
            if tempacc < acc
                feature = i;
                pg = bg;
                tempacc = acc;
            end
```

```
        end
        X(:,temp(feature)) = 0;
        temp = horzcat(temp(:,1:feature-1),temp(:,feature+1:end));
        if bestacc < tempacc
            bestf = temp;
            bestacc = tempacc;
        else
            fprintf('\nWarning, Accuracy has decreased!
                        Continuing search in case of local maxima\n');
        end
        fprintf('\nFeature set {%s} was best accuracy is %.1f %%\n\n',pg,tempacc*100);
    end
    bg = sprintf('%d ', bestf);
    fprintf('\nFinished search!! The best feature subset {%s}
                which has accuracy of %.1f %%\n',bg,bestacc*100);
end

function abhi_selection(X,Y)
    fprintf('\nRunning...Boosting Algorithm..!!!\n\nPlease wait..!!\n\n');
    [m n] = size(X);
    ens1 = fitensemble(X,Y,'AdaBoostM1',100,'Tree');
    imp1 = predictorImportance(ens1);
    [sortedValues,sortIndex] = sort(imp1(:),'descend');
    maxIndex = sortIndex(1:5);
    fprintf('Boosting Finished..!!\n\n');
    ag = sprintf('%d ', maxIndex);
    fprintf('Top 5 features selected from Boosting: %s\n\n',ag);
    fprintf('Calculating Accuracy for each features\n\n');
    acc = 0;
    for i = 1:size(maxIndex,1)
        test = X(:,maxIndex(i));
        mdl = fitcknn( test,Y);
        cvmdl = crossval(mdl,'KFold',m);
        kloss = kfoldLoss(cvmdl);
        acc = [acc 1.00 - kloss];
        fprintf('Using feature(s) {%d} accuracy is %.1f %%\n',maxIndex(i),acc(size(acc,2))*100);
    end
    fprintf('\nSelecting top 2 features with best accuracy\n\n');
    [sortedValues2,sortFeature] = sort(acc(:,2:end),'descend');
    maxFeature = sortFeature(1:2);
    sample = [X(:,maxIndex(maxFeature(1))) X(:,maxIndex(maxFeature(2)))];
    selFeature = [maxIndex(maxFeature(1)) maxIndex(maxFeature(2))];
    mdl2 = fitcknn(sample,Y);
    cvmdl2 = crossval(mdl2,'KFold',m);
    kloss2 = kfoldLoss(cvmdl2);
    acc2 = 1.00 - kloss2;
    g = sprintf('%d ', selFeature);
    fprintf('\nFinished search!! The best feature subset {%s}
                which has accuracy of %.1f %%\n',g,acc2*100);
end
```

References

[1] Feature Selection Wikipedia.(https://en.wikipedia.org/wiki/Feature_selection).

[2] kNN Classifier MATLAB documentation.(https://www.mathworks.com/help/stats/fitcknn.html).

[3] Decision tree MATLAB documentation.(https://www.mathworks.com/help/stats/fitensemble.html).

[4] AdaBoost Wikipedia.(https://en.wikipedia.org/wiki/AdaBoost).