CS 170 Intro to Artificial Intelligence

Project 2

Feature Selection
with Nearest Neighbor

SID: 861310198
Tsung-Ying Chen

A printout of code (matlab)

feature_selection.m

```
function features = feature_selection(data)
data = trim_data(data);       % Trim 5% of dataset
data = z_normalize(data);    % Normalize data
searcher = choose_algo();    % Choose algorithm
print_dataset_info(data);
features = searcher(data);
end

function trimmed_data = trim_data(data)
sample_n = size(data,1);
k = randperm(sample_n);
trimmed_data = data(k(1:sample_n*0.95),:);
end

function print_dataset_info(data)
disp(['This dataset has ', num2str(size(data,2)-1), ' features with ',num2str(size(data,1)),' instances.'])
fprintf('\n');
end

function searcher = choose_algo()
algo = input('Type 1: Forward Selection\nType 2: Backward Elimination\nType 3: Exhaustive Search\n');
if algo == 1
      searcher = @forward_search;
elseif algo == 2
      searcher = @backward_search;
else if algo == 3
      searcher = @exhausive_search_3;
else
       searcher = @exhausive_search_2;
end
end

function data = z_normalize(original_data)
data = original_data;
for i = 1 : length(original_data(1,:))
      column_i = original_data(:,i);
      mean_col = mean(column_i);
      std_col = std(column_i);
      for j = 1 : length(column_i)
            data(j,i) = (data(j,i) - mean_col) / std_col;
      end
end
end
```

```
function data = range_normalize(original_data)
data = original_data;
for i = 1 : length(original_data(1,:))
      column_i = original_data(:,i);
      mini = min(column_i);
      maxi = max(column_i);
      for j = 1 : length(column_i)
            data(j,i) = (normalized_data(j,i) - mini) / maxi;
      end
end
end
```

forward_search.m

```
function best_features = forward_search(data)

current_set_of_features = []; % Initialize an empty set
current_accuracy_by_current_set = [];    % For plot
best_accuracy = 0; % For picking the best features

for i = 1 : size(data,2)-1
    feature_to_add_at_this_level = [];
    best_so_far_accuracy = 0;

    for k = 1 : size(data,2)-1
        if isempty(intersect(current_set_of_features,k)) % Only consider adding, if not already added.
            test_set_of_features = [current_set_of_features k];
            accuracy = leave_one_out_cross_validation(data,test_set_of_features);
            disp(['Using feature(s) ', num2str(test_set_of_features),' accuracy is ', num2str(accuracy)])

            if accuracy > best_so_far_accuracy
                best_so_far_accuracy = accuracy;
                feature_to_add_at_this_level = k;
            end
        end
    end

    %%% Add the best feature
    current_set_of_features(i) = feature_to_add_at_this_level;
    current_accuracy_by_current_set(i) = best_so_far_accuracy;
    disp(['On level ', num2str(i),' i added feature ', num2str(feature_to_add_at_this_level), ' to current set with
accuracy ', num2str(best_so_far_accuracy),])
    fprintf('\n');

    %%% Record the best subset
    if best_so_far_accuracy > best_accuracy
        best_features = current_set_of_features;
        best_accuracy = best_so_far_accuracy;
    end
end

disp(['Finished search!! The best feature subset is ', num2str(best_features), ', which has an accuracy of
',num2str(best_accuracy)])
plot_features_err(current_accuracy_by_current_set);

end
```

backward_search.m

```
function best_features = backward_search(data)

features = size(data,2)-1;
current_set_of_features = 1:features; % Initialize a full set
current_accuracy_by_current_set = zeros(1,features);

best_accuracy = leave_one_out_cross_validation(data, current_set_of_features); % Use full set as start
current_accuracy_by_current_set(features) = best_accuracy;
disp(['On level 1 Using feature(s) ', num2str(current_set_of_features), ' with accuracy ',
num2str(best_accuracy)])
fprintf('\n');

for i = 2 : features
    feature_to_remove_at_this_level = [];
    best_so_far_accuracy = 0;
    for k = 1 : size(data,2)-1
        if ~isempty(intersect(current_set_of_features, k)) % Only consider deleting, if not already removed.
            test_set_of_features = current_set_of_features;
            test_set_of_features(find(test_set_of_features == k)) = [];
            accuracy = leave_one_out_cross_validation(data,test_set_of_features);
            disp(['Using feature(s) ', num2str(test_set_of_features),' accuracy is ', num2str(accuracy)])

            if accuracy > best_so_far_accuracy
                best_so_far_accuracy = accuracy;
                feature_to_remove_at_this_level = k;
            end
        end
    end

    %%% remove the worst feature
    current_set_of_features(find(current_set_of_features == feature_to_remove_at_this_level)) = [];
    current_accuracy_by_current_set(features - i + 1) = best_so_far_accuracy;
    disp(['On level ', num2str(i),' i removed feature ', num2str(feature_to_remove_at_this_level), ' from current
set with accuracy ', num2str(best_so_far_accuracy)])
    fprintf('\n');

    %%%    Record the best subset
    if best_so_far_accuracy > best_accuracy
        best_features = current_set_of_features;
        best_accuracy = best_so_far_accuracy;
    end
end

disp(['Finished search!! The best feature subset is ', num2str(best_features), ', which has an accuracy of
',num2str(best_accuracy)])
end
```

leave_one_out_cross_validation.m

```
function accuracy = leave_one_out_cross_validation(data, test_set_of_features)
Y = data(:, 1);
X = data(:, test_set_of_features + 1);
predY = [];
for i = 1:size(data,1)
	testX = X(i,:);
	trainX = X;	trainX(i,:) = [];
	trainY = Y;	trainY(i,:) = [];
	predY(i) = nearest_neighbor(trainX,trainY,testX);
end
accuracy = sum(predY == Y')/size(Y,1);
end

function predY = nearest_neighbor(trainX,trainY,testX)
distance_to_nearest_neighbor = inf;
for i = 1:size(trainX,1)
	i_X = trainX(i,:);
	distance_to_i = sqrt(sum((i_X - testX).^2)); % Euclidean

	if distance_to_i < distance_to_nearest_neighbor
		predY = trainY(i);	% i is the nearest neighbor
		distance_to_nearest_neighbor = distance_to_i;
	end
end
end
```

exhausive_search_2.m

```
function best_features = exhausive_search_2(data)
best_accuracy = 0;
for i = 1 : size(data,2)-2
	for j = i + 1 : size(data,2)-1
		test_set_of_features = [i j];
		accuracy =
leave_one_out_cross_validation(data,test_set_of_features);
		if accuracy > best_accuracy
			best_accuracy = accuracy;
			best_features = [i j];
		end
	end
end
end
```

exhausive_search_3.m

```
function best_features = exhausive_search_3(data)
best_accuracy = 0;
for i = 1 : size(data,2)-3
	for j = i + 1 : size(data,2)-2
		for k = j + 1 : size(data,2)-1
			test_set_of_features = [i j k];
			accuracy =
leave_one_out_cross_validation(data,test_set_of_features);
			if accuracy > best_accuracy
				best_accuracy = accuracy;
				best_features = [i j k];
			end
		end
	end
end
for i = 1 : size(data,2)-2
	for j = i + 1 : size(data,2)-1
		test_set_of_features = [i j];
		accuracy =
leave_one_out_cross_validation(data,test_set_of_features);
		if accuracy > best_accuracy
			best_accuracy = accuracy;
			best_features = [i j];
		end
	end
end
end
```

A trace of Forward Selection on dataset CS170Smalltestdata49

```
>> feature_selection(CS170Smalltestdata49)
Type 1: Forward Selection
Type 2: Backward Elimination
Type 3: Exhaustive Search
1
This dataset has 10 features with 100 instances.

Using feature(s) 1 accuracy is 0.59
Using feature(s) 2 accuracy is 0.66
Using feature(s) 3 accuracy is 0.71
Using feature(s) 4 accuracy is 0.62
Using feature(s) 5 accuracy is 0.69
Using feature(s) 6 accuracy is 0.65
Using feature(s) 7 accuracy is 0.61
Using feature(s) 8 accuracy is 0.65
Using feature(s) 9 accuracy is 0.88
Using feature(s) 10 accuracy is 0.73
On level 1 i added feature 9 to current set with accuracy 0.88

Using feature(s) 9  1 accuracy is 0.75
Using feature(s) 9  2 accuracy is 0.82
Using feature(s) 9  3 accuracy is 0.8
Using feature(s) 9  4 accuracy is 0.78
Using feature(s) 9  5 accuracy is 0.94
Using feature(s) 9  6 accuracy is 0.81
Using feature(s) 9  7 accuracy is 0.8
Using feature(s) 9  8 accuracy is 0.82
Using feature(s) 9  10 accuracy is 0.7
On level 2 i added feature 5 to current set with accuracy 0.94

Using feature(s) 9  5  1 accuracy is 0.81
Using feature(s) 9  5  2 accuracy is 0.83
Using feature(s) 9  5  3 accuracy is 0.88
Using feature(s) 9  5  4 accuracy is 0.78
Using feature(s) 9  5  6 accuracy is 0.88
Using feature(s) 9  5  7 accuracy is 0.82
Using feature(s) 9  5  8 accuracy is 0.86
Using feature(s) 9   5  10 accuracy is 0.8
On level 3 i added feature 3 to current set with accuracy 0.88

Using feature(s) 9  5  3  1 accuracy is 0.8
```

```
Using feature(s) 9  5  3  2 accuracy is 0.83
Using feature(s) 9  5  3  4 accuracy is 0.86
Using feature(s) 9  5  3  6 accuracy is 0.85
Using feature(s) 9  5  3  7 accuracy is 0.8
Using feature(s) 9  5  3  8 accuracy is 0.87
Using feature(s) 9  5  3 10 accuracy is 0.73
On level 4 i added feature 8 to current set with accuracy 0.87

Using feature(s) 9  5  3  8  1 accuracy is 0.79
Using feature(s) 9  5  3  8  2 accuracy is 0.83
Using feature(s) 9  5  3  8  4 accuracy is 0.9
Using feature(s) 9  5  3  8  6 accuracy is 0.77
Using feature(s) 9  5  3  8  7 accuracy is 0.76
Using feature(s) 9  5  3  8 10 accuracy is 0.77
On level 5 i added feature 4 to current set with accuracy 0.9

Using feature(s) 9  5  3  8  4  1 accuracy is 0.77
Using feature(s) 9  5  3  8  4  2 accuracy is 0.79
Using feature(s) 9  5  3  8  4  6 accuracy is 0.8
Using feature(s) 9  5  3  8  4  7 accuracy is 0.72
Using feature(s) 9  5  3  8  4 10 accuracy is 0.78
On level 6 i added feature 6 to current set with accuracy 0.8

Using feature(s) 9  5  3  8  4  6  1 accuracy is 0.7
Using feature(s) 9  5  3  8  4  6  2 accuracy is 0.74
Using feature(s) 9  5  3  8  4  6  7 accuracy is 0.7
Using feature(s) 9  5  3  8  4  6 10 accuracy is 0.74
On level 7 i added feature 2 to current set with accuracy 0.74

Using feature(s) 9  5  3  8  4  6  2  1 accuracy is 0.73
Using feature(s) 9  5  3  8  4  6  2  7 accuracy is 0.66
Using feature(s) 9  5  3  8  4  6  2 10 accuracy is 0.79
On level 8 i added feature 10 to current set with accuracy 0.79

Using feature(s) 9  5  3  8  4  6  2 10  1 accuracy is 0.74
Using feature(s) 9  5  3  8  4  6  2 10  7 accuracy is 0.74
On level 9 i added feature 1 to current set with accuracy 0.74

Using feature(s) 9  5  3  8  4  6  2 10  1  7 accuracy is 0.67
On level 10 i added feature 7 to current set with accuracy 0.67

Finished search!! The best feature subset is 9  5, which has an accuracy of
0.94
```

My algorithm: Exhaustive Search (2 versions: 2 features and 3 features)

My algorithm is a brute-force method:
exhausive_search_3: check all the combinations of any 2 and 3 features.
exhausive_search_2: check all the combinations of any 2.
This algorithm is based on the premise: the dataset has two strongly related features and one weakly related feature.

1. Time analysis

$$\begin{cases} \text{Forward and backward selection: } \frac{n(n+1)}{2}\times m = O(mn^2) \\ \text{Exhaustive search (3 features): } n(n-1)(n-2)\times m + n(n-1)\times m = mn(n-1)^2 = O(mn^3) \\ \text{Exhaustive search (2 features): } n(n-1)\times m = mn(n-1) = O(mn^2) \end{cases}$$

n: number of features; m: number of instances

In the case--large dataset: $n = 50 \Rightarrow \text{Forward: Exhaustive(3fs)} \approx 1:100$
Considering running times for doing trimmed dataset: as the results showed at next page, exhaustive search always get the same answer, so one time running is enough. Assume that we need to do 10 times forward selection to find the features, then $\text{Forward: Exhaustive(3fs)} \approx 1:10$.

2. It could give better results
In the beginning, I used this method to check the answer for forward and backward selection. Somehow, I found that the accuracy from brute-force search for 3 features could be 3% above the accuracy from forward selection. For example, on dataset-- CS170BIGtestdata75, even I used trim-5%-data and ran it more than 10 times, it still cannot get the best feature subset that the exhaustive search got. Therefore, since forward and backward selection may not get the best features combination, the exhaustive search for 3 features is a choice for the better results.

3. It could be faster
If two strong features with more than 90% accuracy is good enough for feature selection (since forward search is not guaranteed to find the weak feature), we should use exhaustive search for 3 features instead of forward search. Because it always find the two strong features and it's faster:

$$\text{Forward: Exhaustive(2fs)} = \frac{n+1}{2} : n-1 \Rightarrow when\ n > 3, \text{Exhaustive(2fs)}\ is\ faster$$

Compare the search algorithms

Dataset	Forward Selection		Backward Elimination		Exhaustive Search (3fs)	
	feature set	accuracy	feature set	accuracy	feature set	accuracy
CS170Smalltestdata49	{9,5}	0.94	{5,9}	0.94	{5,9}	0.94
CS170BIGtestdata75	{45,15,3,6}	0.92	[1]	0.91	{3,15,43}	0.95
Trimmed large75	{45,15,3,2}	0.926	[2]	0.895	{3,15,43}	0.947
	{45,15,3,23}	0.916	[3]	0.905	{3,15,43}	0.937
	{45,15,3,19}	0.905	[4]	0.905	{3,15,43}	0.947
	{45,15,3,6}	0.926	[5]	0.895	{3,15,43}	0.947
	{45,15,3,20}	0.937	[6]	0.916	{3,15,43}	0.947
CS170BIGtestdata100	{47,15,11}	0.95	[7]	0.91	{11,15,47}	0.95
Trimmed large100	{47,15,11}	0.947	[8]	0.905	{11,15,47}	0.947
	{6,47,15}	0.937	[9]	0.916	{11,15,47}	0.937
	{47,15,6}	0.947	[10]	0.937	{11,15,47}	0.947
	{47,15}	0.947	[11]	0.905	{11,15,47}	0.947
	{47,15,6}	0.937	[12]	0.853	{11,15,47}	0.937

Backward on Trimmed 5% large75: {15}x6, {6,25,39}x5

1. {4, 5, **6**, 8, 11, **15**, 19, 22, 23, 24, **25**, 27, 29, 30, 31, 34, 36, 39, 43, 45, 49}
2. {4, **6**, 11, 12, **15**, **25**, 26, 28, 29, 30, 31, 34, 35, 36, 37, **39**, 42, 43, 44, 45, 47, 48, 50}
3. {2, 4, 5, **6**, 12, **15**, 22, 24, **25**, 28, 30, 35, 36, 37, **39**, 43, 47, 48, 50}
4. {5, **15**, 29, 34, **39**, 49}
5. {5, **6**, 9, 11, 12, **15**, **25**, 29, 34, **39**, 41, 43, 45, 49, 50}
6. {4, **6**, **15**, 16, 24, **25**, 30, 35, 36, 37, **39**, 44, 45, 48, 50}

Backward on Trimmed 5% large100: {41}x6, {20,29,30,32,34,39,44,47,49}x5

7. {7, 9, 11, 13, 14, 15, 17, 18, **20**, 23, 24, 26, 28, **29**, **32**, 33, **34**, 37, 38, **39**, **41**, 43, **44**, 46, 47, **49**, 50}
8. {1, 12, 18, **20**, 23, 24, **30**, **32**, 35, **39**, 40, **41**, 42, 43, **44**, **49**}
9. {3, 7, 9, 10, 11, 13, 15, 16, 18, **20**, 22, 25, 26, 28, **29**, **30**,31,**32**,33,**34**,35,37,38,40,**41**,**44**,46,47,**49**,50}
10. {1, 3, 6, 14, 15, 16, 17, **20**, 22, 23, 26, 27, **29**, **30**, 31, **32**, 33, **34**, 36, 38, **39**, 40, **41**, **44**, 47, **49**, 50}
11. {2, 3, 6, 7, 9, 10, 14, 16, 17, 18, **20**, 22, 26, **29**, **30**, 31, **32**, 33, **34**, **39**, 40, **41**, 45, 47, **49**, 50}
12. {4, 15, 23, 24, 27, **29**, **30**, **34**, 35, 36, 37, **39**, 40, **41**, 42, 43, **44**, 45, 47, 48}

Dataset	Exhaustive Search (2 features)	
	feature set	accuracy
CS170Smalltestdata49	{5,9}	0.94
CS170BIGtestdata75	{3,15}	0.93
CS170BIGtestdata100	{15,47}	0.93

As the result showed, we can conclude the following points:

1. Choose a fast algorithm on small datasets
   All the four algorithms get the same answer, so we can choose a fast algorithm like forward selection when the number of features is not a lot.

2. Forward selection can get the two strong features but may not get the weak feature
   On CS170BIGtestdata75, forward selection always got {3,15,45}. However, feature 45 is different from exhaustive search (3 features). Which means forward selection may not get the weak feature.

3. Backward Elimination may be hard to find strong features on large dataset
   The best accuracy subset from backward elimination is always big. The number of features could be 10~25. On CS170BIGtestdata75, the most frequent features which is 4 features include only one strong feature. And on CS170BIGtestdata100, even the most frequent features which is 11 features are only one strong feature included. Moreover, the accuracy from backward elimination is often less than the other two algorithms. Thus, backward elimination may be fast but not an effective algorithm to do the feature selection.

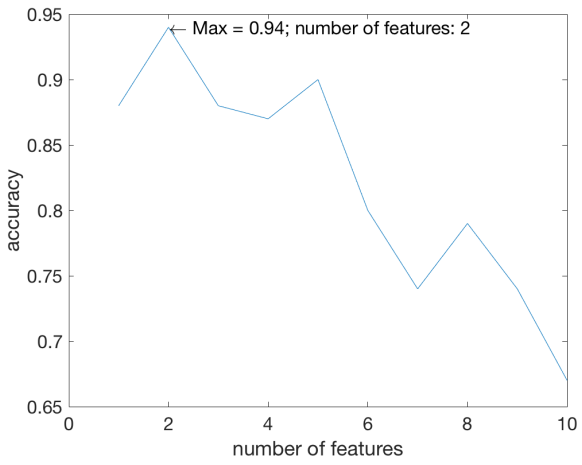
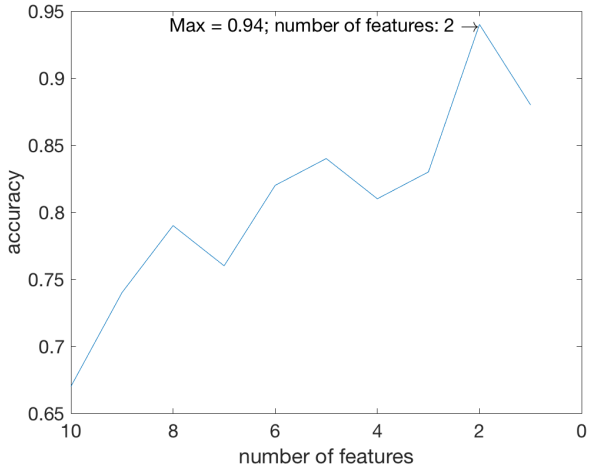
4. Exhaustive search seems always get the best features subset
   The features subsets from exhaustive search (3fs) are always the same and the accuracy keeps stable. On the effective perspective, exhaustive search is a pretty good algorithm.

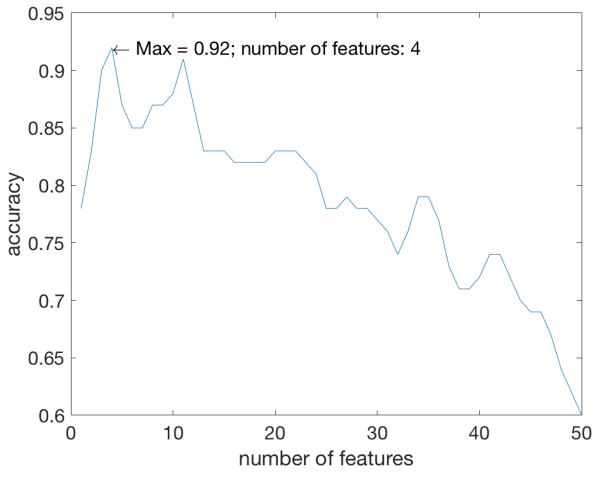
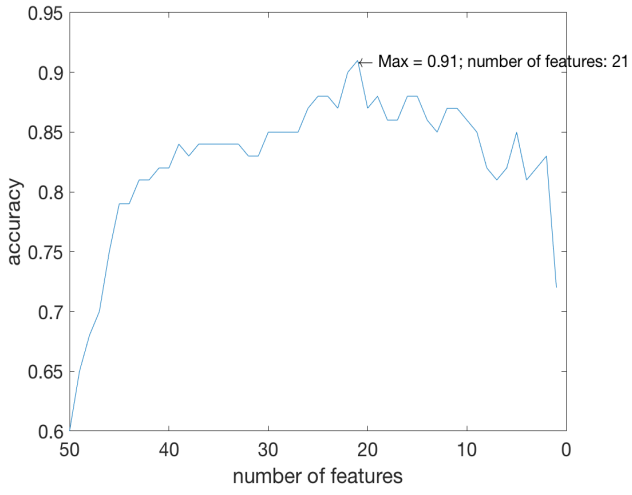
5. The exhaustive search seems to be a better choice
   On dataset CS170BIGtestdata75, the answer from forward selection is {45,15,3,6} with accuracy 0.92, the answer from exhaustive search (2fs) is {3,15} with accuracy 0.93, and the answer from exhaustive search (3fs) is {3,15,43} with accuracy 0.95. The forward selection never got feature 43, which means it never get the best subset. Therefore, the exhaustive search is better since the 2-

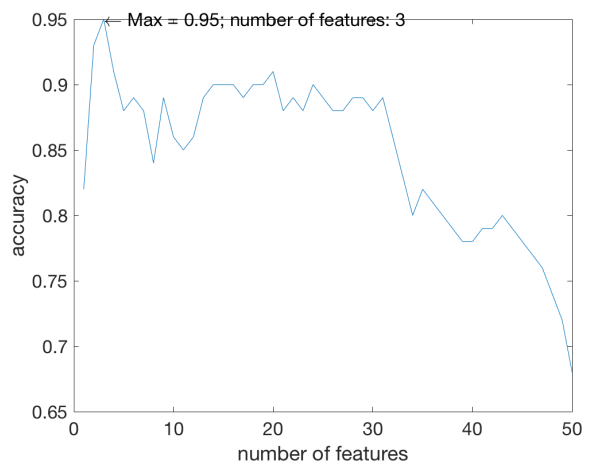
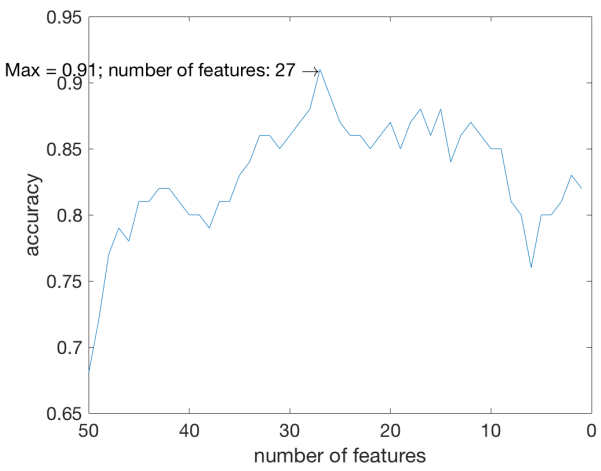
feature version is faster and gets the 2 strong features and the 3-feature version is guaranteed to get the best answer.

6. Forward selection may be a better choice if the number of key features is more than 3.

Number of features vs. Accuracy

CS170Smalltestdata49 {5,9}	
Forward Selection	Backward Elimination
	

CS170BIGtestdata75 {3,15,43}	
Forward Selection	Backward Elimination
	

CS170BIGtestdata100 {11,15,47}	
Forward Selection	Backward Elimination
	

According to the graphs showed, we can conclude the following points:

1. Most of the features are useless.

2. The reason that the backward elimination gets the highest accuracy with more than 20 features is that some useless features accidentally forms a good combination for accuracy. So the key features may be removed at some point.

3. The distribution of the data in these cases are more fit to forward selection. However, if we have a data distribution as below, the backward elimination should perform better because it should have at least two features to separate different classes. And forward selection doesn't work at the first level.