Sign Language Classification Using Search Algorithms

Jing Xu JINGXU@CS.UCR.EDU
Jianfeng Yang JYANG@CS.UCR.EDU

Abstract

We concentrate on the problem of feature selection on sign language classification. The goal is to maximize the classification accuracy using a proper subset of features out of totally 22 given features. In this work, we employ two search methods: Hill Climbing approach and Random Walk approach, to select the features. We claim that both algorithms are easy-implemented but reasonable and efficient. We illustrate the power of our methods by successfully achieving a relatively high classification accuracy on a real dataset.

1. Introduction

Sign language is a language that uses manual movements to convey grammatical structure and meaning. A movement representing a sign or more exolicitly, a word, is usually fulfilled by different parts of a human being's body, for instance, fingers, wrists or heads. Whenever a sign is expressed, each part of the body shows a different movement respectively. Intuitively we can convert these movements into the corresponding time series. Hence for each word in the sign language, we have a set of time series describing some certain parts of the body. These time series thus construct the original whole feature space for the sign language.

Our goal is simply to maximize the classfication accuracy. Naturally, it is true that the more features we have, the better we know about the dataset. However, because of the existance of redundant information among the features, which we call "noise" here, we argue that not all of the information is useful. Moreover, some unrelated features would possible influence the correct decision and we might get much worse performance without pruning them. Thereby, using all these features together to construce our classification feature space is not a good idea.

Our aim is to find a combination of features to maximize the classification accuracy. Basically it is a feature seletion problem. Although it is a classical machine learning topic and a lot of effective approaches have been applied on it, such as Principle Component Analysis, Boosting algorithms and Genetic algorithms, etc. We do it in a differnt way. We consider the problem as an optimizing search problem essentially. The goal is thus trying to find such a subset of features which can describe the sign language most precisely and thus can classify the data most correctly.

In this work, we apply two approaches on searching the best subset of features, Hill Climbing and Random Walk. Our experiment results on a real dataset demonstrate that our models are innovative and effective.

The rest of paper is organized as follows. In section 2 we briefly review the prior work in this field. In section 3, we describe the two approaches we used. We show some experiment results in section 4. And we have some conclusion in section 5.

2. Related Work

Classification is a traditional research field in artificial intelligence. Given the training dataset which contains labeled data, a well-learned model should be able to predict the unlabeled testing data correctly. In the feature selection problem, we are trying to find a set of most representative features and using them to build the classification model.

Many learning methods have been proposed for feature selection. Principle Component Analysis method is trying to project the features onto the most important eigenvectors of the covariance matrix and transform the feature vectors into another axis. Boosting methods use the idea that initially set all the features as equally importance, that is, a same weight, and after each iteration of learning, the weights are adjusted to get a better predicate accuracy. Genetic algorithms

AI project for cs 205 in winter 2006

also show a good performance in this problem. Some other methods focus on entropy information of the features and try to extract a most independent subset of all the features.

While there has been a great variety of previous work, our work is novel in that using very simple ideas, we achieve a very good performance by using search methods. Although searching methods are widely used in machine learning problems, it is hard to build a robust search model on feature selection since there is not a definite and clear goal state here. The two methods we present here are efficient, and we argue believe that they can be adaptive to other similar problems easily.

3. Data Preprocessing

Figure 1 shows an example of the relationship between a sign language and the corresponding time series describing it.

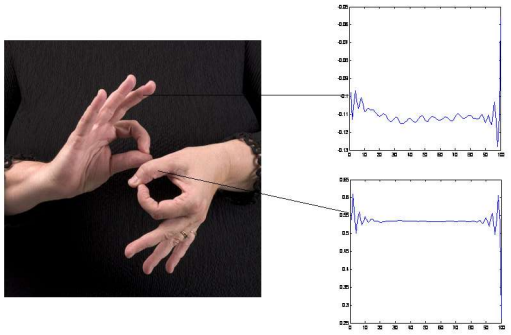

Figure 1. An example of a sign language and the corresponding time series describing the movements of this sign

Our data set contain 95 different sign language words, each of which is repeated 27 times and has 22 features. Each feature is a time series with 136 numbers. First, we divide the whole data set into two parts, the training set and the test set. The training set is used to select one set of features. And the test set is used to evaluate the performance of the set of selected features. These data have different value ranges because they are from different people and different time. So we normalize these data first. The data are normalized independently, which avoids the dependency among the dataset itself. Also we only take part of the data to do the project. The training set contains 95 words, each of which is repeated 6 times. The test set contains 95 words, each of which is repeated 3 times.

4. Approach and Experiment Results

We employ two search algorithms to select features here. After getting the proper set of features, we use Nearest Neighbor classification method to predicate the labels. The distance metric applied here is Euclidean distance.

In this particular work, no goal state is defined since we can not explicitly know how high the accuracy it can approach. Obviously no 100% accuracy can be achieved on any algorithm. Also if we set a threshold on the accuracy and claim that whenever the accuracy exceeds this threshold, it gets a goal state, this will also bring us to a dilema that how high the threshold should be. Also, we have a lot of alternative ways of intializing the start state. Given 22 features in total, we can either use part of them or all of them. The score function we employed to measure the state here is the prediction accuracy.

We conjecture that the change of accuracy according to the number of features should be look like what is shown in Figure 2.

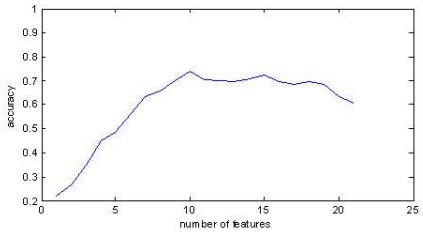

Figure 2.

In the following two subsections, we will first look into two approaches separately. After that we will compare these two methods and draw some conclusions.

4.1. Hill Climbing Search Algorithm

Hill climbing search algorithm is essentially a greedy choice approach. The basic idea is that we branch out all the child nodes of the current state and pick the one which has the highest score value, which is prediction

accuracy. The pseudo code for this approach is as follows.

Algorithm 1 Hill-Climbing Algorithm

```
function Hill-Climbing returns a solution state
inputs: problem // a problem
local variables: current; next // a node
current ⇐ Make-Node(Initial-State[problem] //initialize as 0 feature or 22 features
loop
  next ⇐ a highest-valued successor of current
  if Value[next]<Value[current] then
    then return current
  end if
  current ⇐ next
end loop
```

In this method, we have two choice of state initialization: **forward** and **backward**. We can either start with 0 feature, then andomly pick one feature and expand the whole search tree on it. Each time we expand the current best node, and find the next best node, and do it repeatedly. Hence, in each iteration, we always add one feature and guarantee that the accuracy is increasing. Whenever we come to a state in which no child nodes of it can improve the current accuracy, the iteration terminates. Besides, we can also use all 22 features as the start state and each time we delete a feature under the condition that this deletion improves the accuracy. Both methods will reach a final state in which no addition or deletion can improve the accuracy. This final state is thus used as our classification model.

Figure 3 plots the accuracy we get on training dataset and test dataset seperately using 0 feature as the initial state, that is, forward method.

Figure 4 shows us the detailed process of forward search in hill climbing.

Figure 5 plots the accuracy we get on training dataset and test dataset seperately using 22 features as the intial state, that is, backward method.

Figure 6 shows us the detailed process of backward search in hill climbing.

We can see from the above figures that the best combination of features are not the whole set. It verifies our conjecture that the noisy features here will have bad effect on the prediction and thus we should prune these irrelavent features. Using hill climbing method, we successfully achieve accuracies as high as 82%.

Unfortunately, since we are using greedy method, and we cannot guarantee that the final state we get is a global maximum state. We cannot avoid going into a local maxima here.

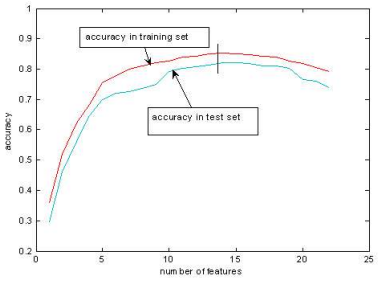

Figure 3. accuracy plots by forwarding in hill climbing.

test set accuracy	feature adding list	training set accuracy
0.29	17	0.36
0.46	17 15	0.52
0.56	17 15 6	0.62
0.64	17 15 6 19	0.68
0.694	17 15 6 19 14	0.75
0.72	17 15 6 19 14 4	0.78
0.72	17 15 6 19 14 4 22	0.80
0.73	17 15 6 19 14 4 22 5	0.81
0.74	17 15 6 19 14 4 22 5 9	0.81
0.79	17 15 6 19 14 4 22 5 9 12	0.82
0.80	17 15 6 19 14 4 22 5 9 12 21	0.83
0.80	17 15 6 19 14 4 22 5 9 12 21 20	0.84
0.81	17 15 6 19 14 4 22 5 9 12 21 20 16	0.85
0.81	17 15 6 19 14 4 22 5 9 12 21 20 16 13	0.85
0.82	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18	0.85
0.81	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2	0.84
0.81	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2 7	0.84
0.81	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2 7 10	0.84
0.80	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2 7 10 3	0.82
0.76	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2 7 10 3 8	0.81
0.76	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2 7 10 3 8 1	0.80
0.73	17 15 6 19 14 4 22 5 9 12 21 20 16 13 18 2 7 10 3 8 1 11	0.79

Figure 4. Forward in hill climbing. Each row shows the set of features and the corresponding accuracy on training dataset and testing dataset

4.2. Random Walk Search Algorithm

In this approach, we extend the flexibility of choosing the states. Instead of building a searching tree here, we build a searching graph. More detailedly, whenever we come to the current state, we have three alternative choice to pick the next state: **add**,**delete**,**replace** a feature. This three choice is randomly picked. The most exciting trick here is that we don't expand the all the possibility child nodes, instead, we pick one child and check whether the new accuracy is better than the current we have, if so we never expand any

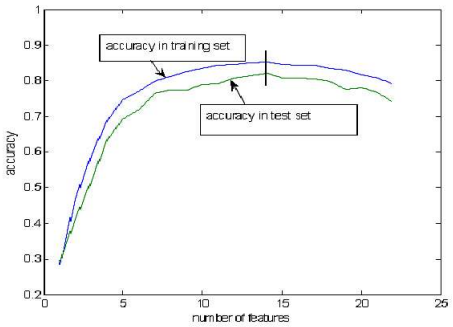

Figure 5.

test set accuracy	feature striking list	training set accuracy
0.73	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7 6 2 5 3 11 10	0.79
0.76	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7 6 2 5 3 11	0.80
0.78	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7 6 2 5 3	0.81
0.77	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7 6 2 5	0.83
0.79	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7 6 2	0.83
0.80	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7 6	0.84
0.80	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18 7	0.84
0.80	22 15 19 14 12 4 16 9 21 17 1 20 13 8 18	0.84
0.81	22 15 19 14 12 4 16 9 21 17 1 20 13 8	0.85
0.81	22 15 19 14 12 4 16 9 21 17 1 20 13	0.85
0.80	22 15 19 14 12 4 16 9 21 17 1 20	0.84
0.79	22 15 19 14 12 4 16 9 21 17 1	0.84
0.78	22 15 19 14 12 4 16 9 21 17	0.83
0.77	22 15 19 14 12 4 16 9 21	0.82
0.77	22 15 19 14 12 4 16 9	0.81
0.76	22 15 19 14 12 4 16	0.79
0.72	22 15 19 14 12 4	0.77
0.69	22 15 19 14 12	0.74
0.63	22 15 19 14	0.68
0.52	22 15 19	0.59
0.41	22 15	0.46
0.29	22	0.28

Figure 6.

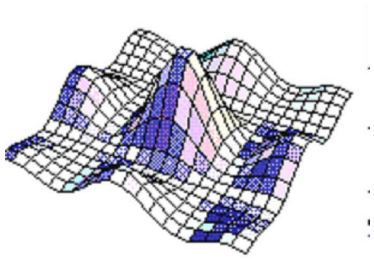

Figure 7. An example of local maximum state

mode nodes and directly go to that state. Otherwise, we pick another node and repeat the work above. So in each iteration, we only test a few nodes or even one node in the best case. This saves us a lot of time.

And the final state here is defined as no transition can improve the current accuracy and we claim that this is the local maximum state we get.

Figure 8 demonstrates one example of how state transition works.

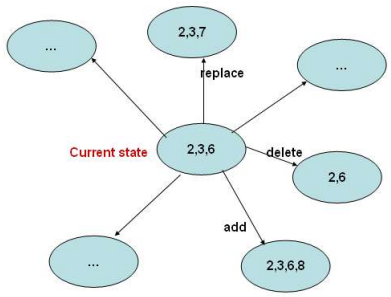

Figure 8. an example of how random walk method works

The following is the pseudo code for random walk approach.

Similarly as we do in hill climbing methods, we use **forward** and **backward** directions to expand our searching graph. Figure 9 and Figure 10 show the accuracy plot. Since in random walk, although the start can be fixed, for instance, 0 and 22 features here, since each step is randomly chosen, the accuracy plots can be very different. So we run our algorithm on the same dataset with same initial state for several times. Excitingly, all these random steps "walk" to a similar best accuracy.

Furthermore, we can randomly choose any combination of features as the start state. Figure 11 shows us that although the start accuracy can be very different according to the start state it randomly picks, the final local maximum accuracies each random iteration get are still very close.

We also use the resulting state of hill climbing approach gets as the start state and Figure 12 plots this experiment results.

Figure 13 shows us the detailed process of random walk search. After each step, a higher accuracy is guaranteed.

In random walk approach, we also achieve an 82% accuracy on almost all the iteration. However, in this method, local maximum problem is not avoided ei-

Algorithm 2 Random-Walk Algorithm

function Random-Walk
inputs: *problem*
outputs: *a solution state*
local variables: *current; next* // node
current⇐Make-Node(Initial-State[*problem*] //make random intial state
loop
next ⇐ **Transit**(*current*)
if Value[next]<Value[current] **then**
then return *current*
end if
current ⇐ *next*
end loop

function Transit **returns** a transition of current state
inputs: *current* // a node
outputs: *next* // a node
next ⇐ **Add** or **Delete** or **Replace** one feature of *current*
end

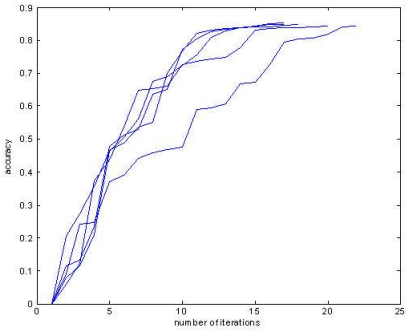

Figure 9. random walk method using 0 feature as start state

ther. For the purpose of decreasing the loss of generality, we should run the random walk for many times and check whether each results are same or similar. Thanks to the high efficiency of this approach, each iteration takes only a little time. So the total time of random walk is not as high as you may imagine.

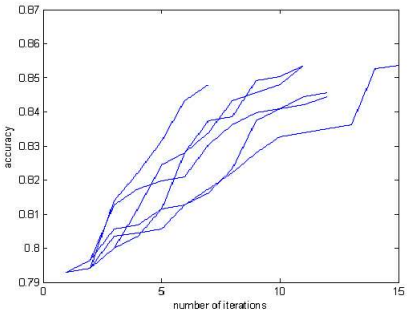

Figure 10. random walk method using 22 features as start state

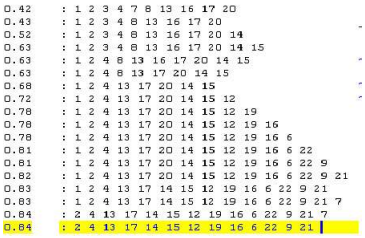

Figure 11. random walk method starts from random initial states

5. Results Analysis

In this section, we show some interesting results we get and we do some comparison of the two approaches.

First, we display the final sets of features we get using hill climbing approach in the following table. **F** denote forward and **B** denotes backward.

```
F [1 4 8 9 12 13 14 15 16 17 19 20 21 22]
B [4 5 6 9 12 13 14 15 16 17 19 20 21 22]
```

These two opposite ways share as much as 12 features.

Second, in random walk method, the commonly selected features of each initialization way are also listed in the following tables. **00 22 HC RD** denote start-

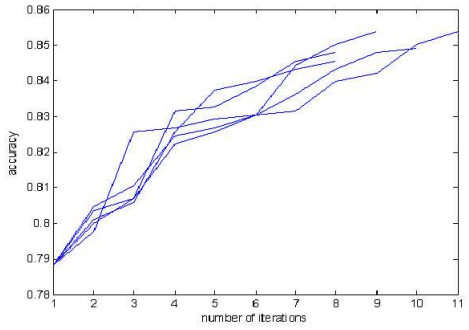

Figure 12. random walk method starts from the resulting state of hill climbing

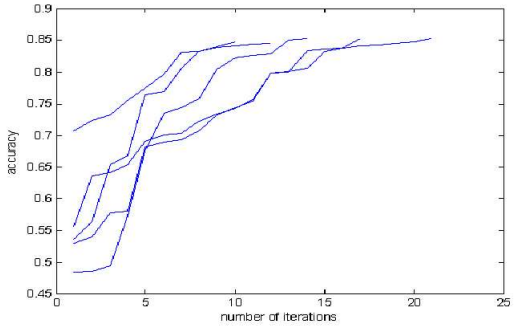

Figure 13.

ing from 0 feature, 22 features, resulting state of hill climbing and randomly picked features respectively.

```
00: [4 9 12 13 14 15 17 19 21 22]
22: [4 9 12 13 14 15 17 19 21 22]
HC: [4 9 12 14 15 17 19 21 22]
RD: [4 9 12 13 14 15 17 19 21]
```

It is also amazing that these four ways themselves share a lot of common features. Moreover, they together share a lot of common parts.

Based on the above results, we claim that with the two approaches, we get almost the same combination of features. To some extent, we can assume that this set of features is good enough.

6. conclusion

In this paper, we present two searching method on feature selection. We employ hill climbing method and random walk method and both of them successfully extract a set of useful features thereby classifying the sign language with a relatively high accuracy. In the future, we will extend our work to these interesting fields:1. Analyze what these selected features stand for to see whether they make sense; 2. Compare the results of this project with those of other methods.

Acknowledgments

Thanks to Dr. Eamonn Keogh for providing us the dataset and very useful advice.