Linear Classifier for Number Recognition

Project Report

Project Report
Revision 1
© NDSU Team AI

Dinesh Singh, Meng Chao and Prateek Rajan

Abstract

The paper is about using the linear classifier to categorize the given gesture to predefined classes between 0-9. The core idea is to utilize a set of features that can uniquely identify an input gesture. The main research focus in our project was about deciding on these set of features. Once this feature set was defined we train the system on some example gestures for each class (10 for this specific project). Once this training is done the classifier can map any input gesture to a class. The rest of this report will talk about how we defined the features, what are the properties they define & how the actual classification works. Towards the end of this report we will talk about the problems that we faced along the path & what is the future scope of this project.

Table of Content

Introduction

Motivation

Core of our application is motivated by Dean Rubine's paper "Specifying Gestures by Example". The paper talks extensively about recognizing gestures. This can be considered as a classic example of human computer interaction. Rubine quotes" The intuitiveness and power of this gesture hints at the great potential of gestural interfaces for improving input from people to machines, historically the bottleneck in human computer interaction." Rubine suggested that more precisely the number of features that defines the characteristics of the gesture, more the accuracy will be. So we have tried to introduce our own features to define the characteristics of numbers from 0 to 9. At the end of this report we will be showing the accuracy of our classifier.

Basic Idea

The idea is to develop/Implement a model which recognizes gestures. Gestures are hand markings entered by a mouse or stylus. In our project we are working to recognize numbers (0-9). The project can be extended to recognize alphabets or any other gestures. The gesture will be entered using a mouse by a user and the application will return the class to which this gesture (number) entered by the user, much like a handwriting recognition application. The input data (Test Data will be trained with existing data in the application (Training Data) to find a closest match. The GUI will be developed in Java. MATLAB might be used to implement some of the core mathematical features or concepts.

So now the question comes; How can it be done? How to make a computer understand which number is entered by the user and which class of numbers it belong to?

This problem can be solved by defining certain features of different gestures. In our case as we are only recognizing numbers, we can define physical features of numbers ranging from 0-9. For Example:

Straightness: How straight the number is? 1 & 7 will have straightness values higher than 0 or 8

Similarly sufficient number of features must be defined to cover all the gestures. Features must be mathematically definable and more the features are the more accuracy can be produced. But if too many features are selected and implemented then it may also complicate the whole process.

Linear Classifier:

In the field of machine learning, the goal of statistical classification is to use an object's characteristics to identify which class (or group) it belongs to. A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics.

If the input feature vector to the classifier is a real vector $\vec{x}$, then the output score is

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right),$$

Where $\vec{w}$ a real vector of weights and f is is a function that converts the dot product of the two vectors into the desired output. (In other words, $\vec{w}$ is a one-form or linear functional mapping $\vec{x}$ onto R.) The weight vector $\vec{w}$ is learned from a set of labeled training samples. Often *f* is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex *f* might give the probability that an item belongs to a certain class.

Wikipedia.com

Getting Started

Features We Used

Distance Feature

This feature calculates the distance between initial and final points of the sketch.

Let's assume (x1, y1) being initial points and (x2, y2) are final point. Then distance between final and initial points are calculated as

D= Square Root [(x2-x1)^2 +(y2-y1)^2]

This feature is very useful in calculating other features like Cosine and Sine of the initial angle of the sketch.

Also distance between initial and final points of 1 will be much more than 0 unless someone has not sketched a very small 1. We can use this feature to distinguish between a straight sketch and a sketch which is almost a circle.

SINE & COSINE Feature

Sine and Cosine features calculate the Cosine and Sine of the initial angle of the sketch between the initial and final point of the sketch.

Sine Feature is calculated as:

Sine A = (y2-y1)/Distance Feature

Similarly, Cosine Feature is calculated as:

Cosine A(x2-x1)/Distance Feature

Line Test Feature

The basic idea is to use this feature to check the straightness of a sketch. First a list square line is fitted to the sketch points and then orthogonal distance squared is calculated between the fitted line and sketch points. Now this distance is then divided by the sketch length to compute least square error. We determine that the sketch is a line if the least square error is below a certain threshold.

NDDE Feature

Normalized Distance Between Direction Extremes: - We first take two points; one with highest direction value that is change of y value over x and second point with the lowest direction value. Then the sketch length between these two points is calculated. This length is divided by the entire sketch length and the result is the percentage of the sketch that lied between the two direction extremes.

This feature is very important for the project as it can clearly distinguish between round (arc) and polyline (straight shapes with sharp angles).

For curved shapes the highest and lowest direction values will be near the sketch's endpoints and will have high NDDE value. While polyline shapes like 4 or 7 will have one or more spikes in their direction graph. These spikes in the shapes

cause the highest and lowest direction points to be away from the end points of the sketch. These shapes will have a lower NDDE value then the curved sketch.

DCR Feature

Direction Change Ratio:- This is another important feature in recognition process. This feature is very useful in recognizing spikes or sharp turns in the sketch. This feature is computed by dividing the maximum change in direction by average change in direction. DCR values for a polyline shape will be higher than the DCR values of a curved sketch.

Intersection Feature

This feature checks if the sketch traverses the same point twice. This feature is very important in representing characteristics of number 8, 6 and 0. In all of these cases there is a point which is traversed at different instances.

Theta Feature

It is the total angle traversed in the sketch. This is calculated by adding all the angles between successive sketch points. This feature provides us the total orientation of the sketch.

Recognition Process: The Fun Part

To recognize an input gesture the same set of features are calculated for it and the system looks in the training set for the closest match.

We created examples for all of the classes (0-9) need to be recognized.

Then we calculate feature vectors for all the classes and hence we obtained a feature matrix of 10x5 dimensions for each class.

Then we calculate mean feature vector for each class.

Next we calculate co-variance matrix for each class from feature matrix of each class.

Then we add up all the co-variance matrices of all the class to get a common co-variance matrix.
Then the common co-variance matrix is inversed and the weights are calculated by the following formula:

$$w_{\hat{c}j} = \sum_{i=1}^{F} (\Sigma^{-1})_{ij} \bar{f}_{\hat{c}i} \qquad 1 \le j \le F$$

Where f(bar) is the mean feature vector of given class and ‘w’ is its calculated weight.

The base weight of each class can be calculated by following formula:

$$w_{c0} = -\frac{1}{2}\sum_{i=1}^{F} w_{ci}\bar{f}_{ci}$$

Where 'w' is the weight of individual class and f(bar) being its mean feature vector.

Now 'v' is calculated, which is the evaluation of all the weights for all the classes. The sketch belongs to the class whose 'Vc' is maximum.

$$v_c = w_{c0} + \sum_{i=1}^{F} w_{ci} f_i \qquad 0 \leq c < C$$

Output Analysis

Gesture Class	Number of Inputs	Recognized	Not Recognized	Accuracy
0	10	10	0	100%
1	10	8	2	80%
2	10	9	1	90%
3	10	6	4	60%
4	10	9	1	90%
5	10	5	5	50%
6	10	9	1	90%
7	10	8	2	80%
8	10	7	3	70%
9	10	4	6	40%

Total accuracy: - Accuracy achieved for the whole training set / Maximum accuracy possible

10+8+9+6+9+5+9+8+7+4 / 100 = 75/100= 75%

Conclusion & Future Scope

Linear classifiers are most basic type of recognition systems that can be utilized to categorize a gesture to a predefined set of gesture classes. Ex. our classifier can be used to recognize alphabets (A-Z or a-z). The only difference that would come would be in the feature set that can be used to allow that kind of recognition. Our classifier was able to recognize 75 % of all the test gestures that were tested on the system. The highest recognition percentage was observed for 0, 2, 4 & 6 having recognition percentage as 100, 90, 90 & 90 respectively. The lowest was observed for 9 & 3 with 40% & 60% recognition percentage respectively. The main reason of such a variation in the accuracy for different numbers was that our feature set was not unique enough to distinguish between all the numbers. With a more efficient feature set this accuracy can be increased manifolds. In the future we can research more about the features that would support such high level of recognition. In addition to that we are also looking forward to extend this project to recognize characters. Last but not the least we look forward to compare this classification technique with neural networks & see which score over the other.

References

i. **Specifying Gestures by Example** by 'Dean Rubine' of Information Technology Center at 'Carnegie Mellon University'.

ii. **PaleoSketch: Accurate Primitive Sketch Recognition and Beautification** by Brandon Paulson and Tracy Hammond at TAMU.

iii. What!?! No Rubine Features?: Using Geometric-based Features to Produce Normalized Confidence Values for Sketch Recognition.