Using A KNN and MOG Based Algorithm for Static Hand Posture Recognition

Abhishek Ranjan
aranjan@cs.toronto.edu
Department of Computer Science
University of Toronto,
Toronto, Canada.

Abstract

Several machine learning algorithms have been applied to the problem of static hand posture recognition. K-nearesr neighbor (KNN) performs very well in flexible posture recognition, but speed and memory requirements of the algorithm make it difficult to use in real time applications. In this paper we propose an approach to speed up the KNN without changing its behavior. We use the mixture of gaussians (MOG) to model the input training data and apply KNN on this model.

1 Introduction

The goal of static hand posture recognition is to classify the given hand posture data, represented by some features, into some predefined finite number of posture classes. Gesture recognition is a similar problem of classifying dynamic hand gestures. The challenges involved in these problems are to track hand positions accurately and classifying the data into postures. Recent advances in the tracking technologies (Vicon, Flock of Birds, CyberGlove etc.) and computer vision algorithms [3] (pages 1-4) have made it possible to get the hand postures tracked quite accurately.

Several machine learning approaches have been applied to the problem of classifying the posture/gesture data. In [5] (pages 3-6) neural nets have been used for posture recognition. A vision based mixture model for pointing gesture recognition has been used in [2] (pages 1002-1005). [4] (pages 2-3) proposes an algorithm for the fundamental problem of clustering the data for classification purposes and applies it to character recognition problem. The system described in [6] (pages 283-284) uses PCA and Multiple Discriminant Analysis for hand posture recognition. Most of these systems rely on vision based techniques for tracking and do not perform well when postures are allowed to be very flexible. Allowing flexibility in posture recognition results in various prototypes of a single posture. Depending on the subject's choice and ease, various instances of a single posture can lie far apart in the feature space. KNN algorithm has often been found to be successful and accurate in such cases [1] (page 417). But KNN algorithm uses all the training data at test time which makes it memory and time expensive. These time space requirements make the algorithm difficult to be used in realtime posture recognition applications such as posture/gesture based human computer interaction.

The nature of the flexible posture recognition problem suggests that various prototypes of one posture can be modeled as different clusters belonging to the same class. Based on these observations we propose an approach which combines KNN and clustering using MOG. The idea is to create an approximate model of the training data by using a mixture of gaussians at training time, and using KNN on this approximate model at test time. In the following sections we describe our experimental set up and formalize our approach. Further we analyze the results of the experiments and conclude.

2 Our approach

We define a set of postures as target classes and represent them in terms of some features. Our MOG based system has been illustrated using the terms of this setup.

2.1 Gesture and feature selection

Our system recognizes 3 postures: Pointing, Stopping and Picking. They are the target postures. These three postures are widely used in various human computer interaction techniques. We allow these postures to be very flexible and natural so that they can successfully be applied to a wide range of people and situations. Figure 1 shows some sample prototypes of the postures on which our system was trained.

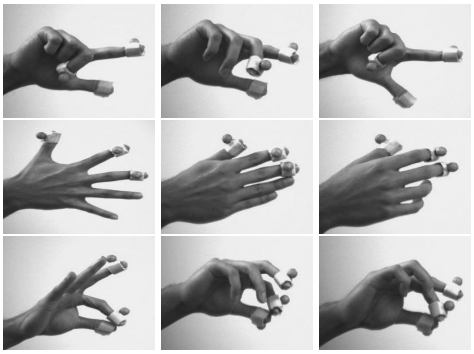

Figure 1: Variations of pointing, stopping and picking postures shown in three rows respectively

Based on these postures we created the feature vector $X = (x_1, x_2)$ with two components

1. Distance between thumb and index finger (named x_1).
2. Distance between index finger and middle finger (named x_2).

Figure 2 shows these two dimensions of the feature vector. This selection of feature vector makes the problem independent of the orientation of the hand in 3D.

2.2 Data collection

We used Vicon motion tracking system for getting 3D data. In this system we calibrate some infrared cameras to track some specific markers in 3D space. These markers are then put on the object which is to be tracked. The system reconstructs the 3D co-ordinates of the markers using various images taken by a bunch of cameras. In our experiments we have used 3 such markers to track 3 fingers (thumb, index and middle). We extract features

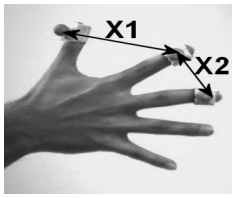

Figure 2: Feature vector $X = (x_1, x_2)$

from the raw data after which all data points lie in the 2D feature space described earlier. The target value y belongs to the posture set $\{c_1, c_2, c_3\}$, where each of the c_i corresponds to one of the three postures described earlier. For our experiments we collected 100 data samples for each of the three postures. Figure 3 shows the data distribution. Different posture data have been shown with different symbols.

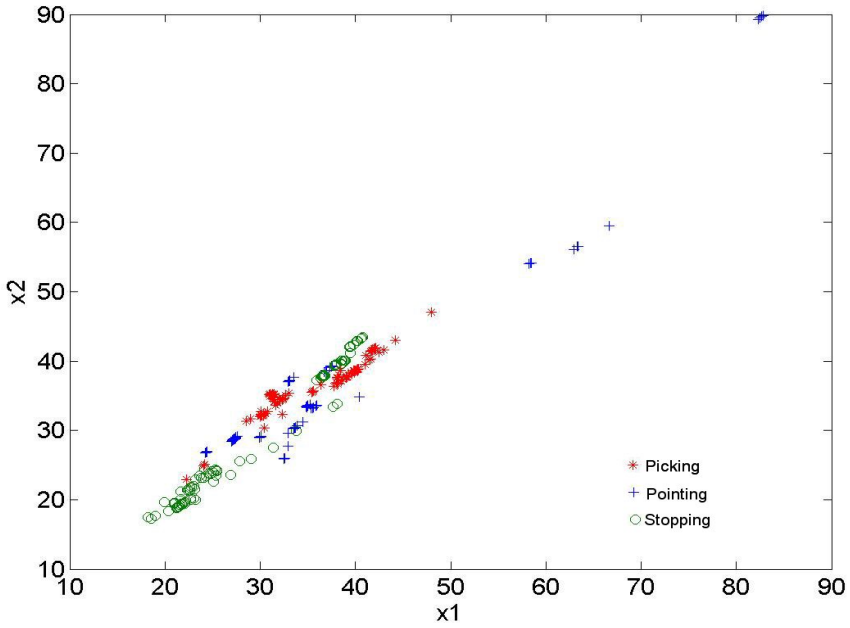

Figure 3: Data points in 2D feature space. 3 classes of data are shown using different symbols

2.3 Algorithm

We describe our approach with respect to our experimental setup, but it can also be generalized to other classification problems. We denote the set of target postures as $C = \{c_1, c_2, c_3\}$, where c_1 represents pointing posture, c_2 represents stopping and c_3 represents picking. We have input training data set $D = \{X_1, X_2, \ldots X_n\}$, where each X_i is a 2D feature vector. The corresponding target values are $Y = \{y_1, y_2, \ldots y_n\}, y_i \in C$. The algorithm takes as input D, Y, two parameters r, k and test points X^{test}. The output

is the test target values Y^{test}. We further divide the input training data into three input classes D_1, D_2, D_3 where $D_i = \{X_j | j \in \{1, 2 \ldots, n\}, y_j = c_i\}$, which means that we put all pointing posture data in the input class D_1, stopping posture data in D_2 and picking posture data in D_3. With these notations our approach can be described in the following steps:

1. **Training**: Using EM algorithm we fit a mixture of r gaussians to each of the training class D_i. In our experiments, during the training we find the parameters of total $3r$ gaussians fitted to the training data. We represent this set of gaussians as $G = \{G_{D_1 1}, G_{D_1 2}, \ldots, G_{D_1 r}, G_{D_2 1}, \ldots, G_{D_2 r}, G_{D_3 1}, \ldots, G_{D_3 r}\}$, where $G_{D_i j}$ is jth gaussian fitted to training class D_i.
2. **Testing**: At test time we have a set of $3r$ gaussians and a test data set $X_{test} = \{X_1, \ldots, X_m\}$. We find the distance matrix $M_{3r \times m}$ similar to that in KNN. But in this case, instead of using Euclidean distance and all the training data points we calculate Mahalanobis distance ($d_m(N, x)$)[1] of each test data point from each of the gaussians. Each entry of the distance matrix M can be defined as $M(i, j) = d_m(G_i, X_j), G_i \in G, X_j \in X_{test}$. The use of Mahalanobis distance takes care of the correlation between different features. Moreover, this allows more general class boundaries such as ellipses and straight lines. In our experiments we computed $3r$ distances for each data point. Now, for each test point, we find k nearest gaussians in Mahalanobis sense. Each of the k gaussians votes for one of the target values c_1, c_2 or c_3 which it belongs to. In the notation used so far this means that a gaussian $G_{D_i j}$ votes for target c_j. The algorithm outputs the target value of the test point as the target value with the maximum number of votes.

If the size of the input data set is n then at test time naive KNN calculates n distances for each of the test data point. Our approach calculates $3r$ distances for each data point. If r is selected to be less than $n/3$ then our approach certainly does less number of distance calculations at the test time and requires smaller amount of memory. Finding an appropriately small value of r could be done using validation data.

3 Experimental studies and results

We analyzed the behavior of our approach and KNN on the posture data. We randomly picked 60% data from each class for training, 20% for validation and 20% for testing. The performance of KNN has been analyzed by varying the value of parameter k. In figure 4 for a fixed training set, training and validation errors have been plotted against various k values. We selected the k value for the test time based on the validation error. Final test error was found to be 5%.

In the MOG based approach, EM algorithm has been used to fit r gaussians to each of the target class data. The number of gaussians, r, and the KNN parameter, k, have been estimated by using validation data set. We fitted 1, 2, 3 and 4 gaussians to each of the target class data. Figure 6 shows different target classes with various gaussians (1 standard deviation) fitted along with the data. We fix the r value at training time and run our algorithm on the validation set for various values of k. Figure 5 shows the plot of this process. Each of the plots in figure 5 has 4 curves. Each curve shows the error rate as a function of k value for a fixed number of gaussians (labels on the curves) fitted to each training class at the training time. From left side plot in figure 5 we can see that one gaussian ($r = 1$) is a poor estimate of the whole class cluster because for various k values the error rate is very high. As the r value is increased, the performance improves. We can expect it to behave

[1] Mahalanobis distance $d_m(N, x)$ of a vector x from a gaussian $N(\mu, \Sigma)$ is $(x-\mu)\Sigma^{-1}(x-\mu)^T$.

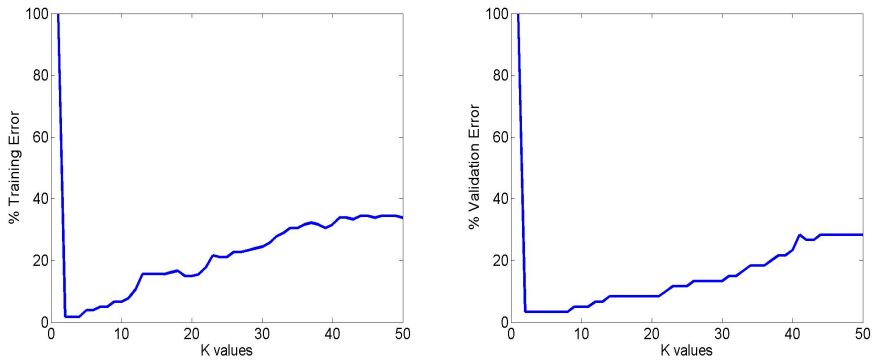

Figure 4: KNN training error (left) and validation error (right) plotted against k value

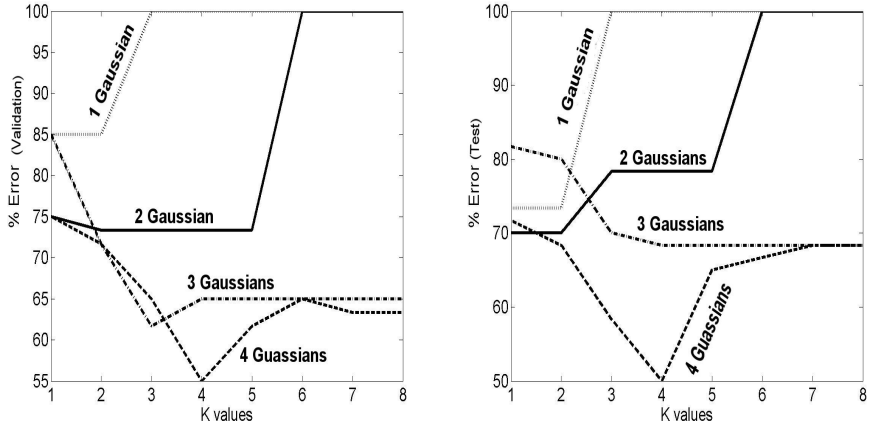

Figure 5: Validation error (left) and test error (right) plotted against k values for different number of gaussians (r values) fitted to the data. Solid, dotted, dash-dot and dash-dash represent 1,2,3 and 4 gaussians respectively.

similar to the KNN as the r value approaches the number of data points in the target data class. The plots in figure 5 and 4 show that for a fixed number of gaussians and varying k value the behavior of our approach is similar to that of KNN. Using the validation data set we estimate the value of the parameters k and r. In our experiments we got the best results with $r = 4$ and $k = 4$. The test error plots for various fitted gaussians and varying k have been shown in the plot on the right side in figure 5.

4 Conclusion and Future Directions

The success of KNN in classifying data has motivated several attempts to speed up the KNN algorithm without losing its accuracy. In this paper we proposed an approach based on mixture of gaussians to speed up the KNN algorithm. Depending on the parameters, the behavior of our algorithm has been found to be similar to that of KNN.

The success rate of our approach depends heavily on fitting mixture of gaussians. If training

data is very noisy then gaussians fitted will have larger covariances and that could cause very bad classification with KNN. Moreover, it would be interesting to study the effect of fitting different number of gaussians to different target classes.

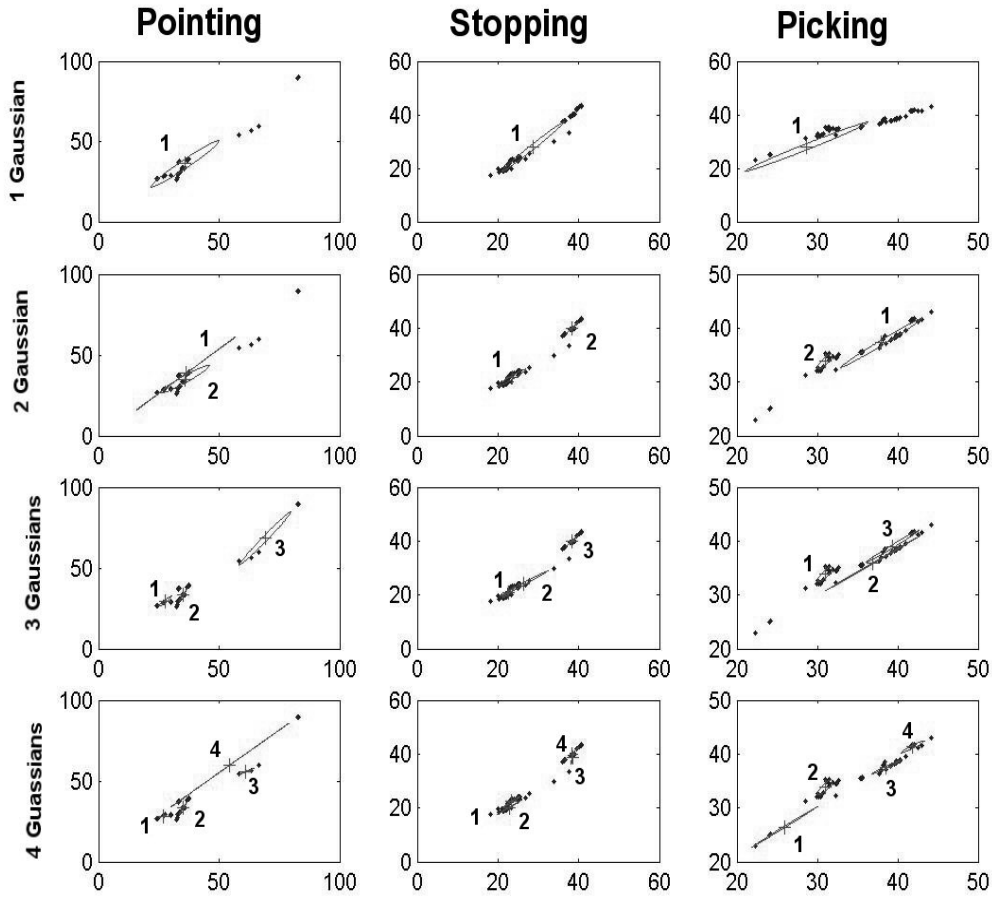

Figure 6: Different number of gaussians fitted to the 3 classes of the data

References

[1] Hastie, T., Tibshirani, R., Friedman, J. (2001) *The Elements of Statistical Learning.*(Pages 411-430) New York: Springer-verlag.

[2] Jojic, N., Brumitt, B., Meyers, B., Harris, S. & Huang, T., (2000) *Detection and Estimation of Pointing Gestures in Dense Disparity Maps*, in Proc. of International Conference on Automatic Face and Gesture Recognition, 468-474, 2000.

[3] Watson, R., (1993) *A Survey of Gesture Recognition Techniques* Trinity College, Dublin, TCD-CS-93-11.

[4] Sun, F., Omachi, S., Aso, H., (1999) *An Algorithm for Estimating Mixture Distribution of High Dimensional Vectors And Its Application to Character Recognition*, Proceedings of The 11th Scandinavian Conference on Image Analysis (SCIA'99), pp.267-274, June 1999.

[5] Bohm, K., Broll, W., and Sokolewicz, M. *Dynamic Gesture Recognition using Neural Networks: A Fundament for Advanced Interaction Construction*. In Proceedings of IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology 1994 (EI'94). San Jos (February 1994), SPIE, 336-346.

[6] Deng, J. W. & Tsui, H. T., *A Novel Two-layer PCA/MDA Scheme for Hand Posture Recognition* in The 5th International Conference on Automatic Face and Gesture Recognition, May 20-21, 2002 , Washington, DC , USA .pp. 294-299.