Learning of Facial Gestures Using SVMs

Jacky Baltes, Stela Seo, Chi Tai Cheng, M.C. Lau, and John Anderson

Autonomous Agent Lab,
University of Manitoba,
Winnipeg, Manitoba,
Canada, R3T 2N2
j.baltes@cs.umanitoba.ca
http://www.cs.umanitoba.ca/~jacky

Abstract. This paper describes the implementation of a fast and accurate gesture recognition system. Image sequences are used to train a standard SVM to recognize Yes, No, and Neutral gestures from different users. We show that our system is able to detect facial gestures with more than 80% accuracy from even small input images.

Keywords: Facial Recognition, SVM, Machine Learning.

1 Introduction

In this paper, we are presenting a machine learning approach to user independent facial gesture recognition. The motivation is to learn a set of simple facial gestures that are nevertheless practical and often useful in human robot interaction.

A SVM is used to train the system to detect three different gestures: Yes, No, and Neutral as shown in Fig. 1. We investigate several pre-processing steps as described in 3.2 to avoid overfitting and to speed up the facial gesture detection.

2 Related Work

In recent years, advances in robot hardware and software has led to increased interest in the area of human robot interaction [5]. Many researchers have looked at various aspects of full body gesture recognition for human robot interaction [4].

Many researchers have investigated the use of computer vision in facial gesture recognition for human robot interaction, since vision provides a cheap and ubiquitous, but extremely flexible and powerful sensor [3]. However, whereas recognizing gestures and motions comes naturally to humans and animals, it is very difficult for computers and robots. This is especially true in unstructured environments where the lighting and background can not carefully be controlled [7]. Researchers use mid-level vision features (e.g., the output of region segmentation and matching or objects grouped through optical flow) to detect and classify gestures. These features are then grouped into gestures using finite state machines, particle filters, neural networks, and hidden Markov models.

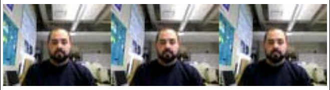
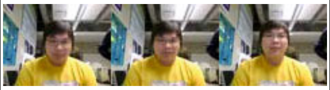
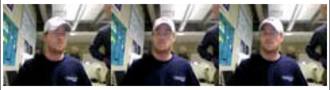
No	Yes
Neutral	Yes
No	 Neutral
 Yes	 No

Fig. 1. Sample Sequences of Gestures

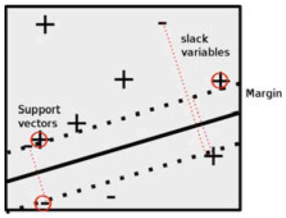


Fig. 2. Linear SVM example showing support vectors, margin, and slack variables

Several researchers have investigated the use of neural networks in recognizing facial gestures [2]. However, these systems are not in common use because their accuracy is not high enough in unstructured environments and without time consuming calibration.

In recent years, support vector machines (SVM) first introduced by Vapnik in 1995 have shown very good performance in a range of supervised classification problems [8]. Support vector machines try to learn a decision surface that maximizes the separation between positive and negative instances of the decision problem. This separation is called the margin and the instances that lie along the margin are called support vectors, which give SVMs their name. An example of a two-dimensional SVM is shown in Fig. 2.

The simplest form of SVM learns linear classifiers which break the training data, which is represented as an n dimensional input vector, into two classes using a $n-1$ dimensional hyperplane. These so-called hard margin Linear SVMs may fail if the training set cannot be separated by a hyperplane.

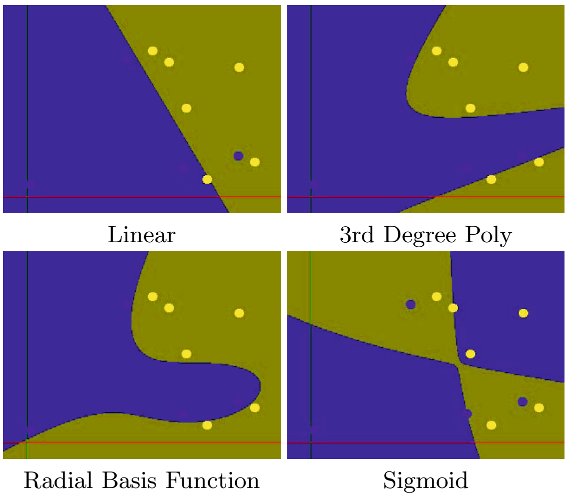


Fig. 3. SVM examples for different decision surfaces. Training data is shown as dots.

A commonly used extension of SVMs are soft-margin SVMs, that do try to both (a) maximize the margin, and (b) minimize the distance of misclassified instances to the boundary of the correct margin boundary. This distance is modelled using so-called slack variables.

SVMs have also been used with more complex decision surfaces such as second, third, or higher order polynomials, radial basis functions, and sigmoid functions. Figure 3 shows a randomly created two-dimensional sample problem. The training data is shown as blue and green dots and the decision surfaces are shown in the corresponding colour.

Other researchers have attempted to use SVMs to learn gestures using support vector machines [6].

3 Design

Implementing a SVM based machine learning system required the implementation of a quadratic programming problem solver, which is non-trivial. Luckily, a high quality open-source library `LIBSVM` was developed and made available as open source software by Chang, Chih-Chung and Lin, Chih-Jen [1].

`LIBSVM` has bindings to various programming languages include C,Java, and Python. It provides several optimizations and improvements to the original SVM model. It is fast, reliable, and well documented. We therefore chose to use `LIBSVM` as the SVM solver in our application. The system described in this paper was implemented in Java and the complete code can be downloaded from our lab website (http://aalab.cs.umanitoba.ca).

3.1 Conversion of Images into Instance Vectors

The images must be converted into real-valued vectors so that they can be used as inputs to an SVM. One issue is that intuitive conversions of the input to a real-valued scalar may result in greatly different scales for the different dimensions of the vector. For example, if we are trying to learn a classifier for predicting a disease then we may want to include body weight as well as amount of iron in the blood stream. However, typical body weight will be around 75 kg, whereas iron will be measured in milligrams. But classification using SVMs is based on the dot product of the instance and the weight vector, which corresponds to the angle between the instance and weight vector. Any change in the iron vector will be overshadowed by changes in the weight vector.

Therefore, scaling of the real-valued input vector is an important step that determines the success of the SVM. This is easily done in images, since the minimum and maximum of the different channels is known. We normalize each colour channel (red, green, or blue) of a pixel or brightness of a pixel into a value between 0 and 1, by dividing it with 255, the maximum value for each channel or brightness. The values of the difference images are normalized by dividing each value with two times the maximum and adding a constant offset of 0.5 to the value for colour or greyscale difference images respectively.

The vector for an instance is created from an image by joining pixels from the top left to the bottom right along each row.

3.2 Pre-processing

In most machine learning applications, too much data is just as bad as too little data. The reason for this is that too much input data often leads to overfitting because features that are not relevant to the target concept may occur in significant patterns by chance. This problem is called the curse of dimensionality.

We therefore investigated several pre-processing methods to reduce the size of the input from the original 240*60*3 dimensional vector. We considered four direct and two difference representation pre-processing steps as described in the following.

Direct Representation. Firstly, we converted the colour images to grey-scale images. Each pixel was replaced by the average brightness of the red, green, and blue channels.

Secondly, we sub-sampled the images by a factor of four, which resulted in 60*15 sized images. Each pixel in the sub-sampled image is the average of four adjacent pixels in the original image.

Difference Representation. We also investigated a difference representation for the input images. In that case, an image was created by subtracting the first image from the second image in the sequence and another image by subtracting the second image from the third image in the sequence. The two images were then combined into an input image. A sample difference image is shown in Fig. 4.

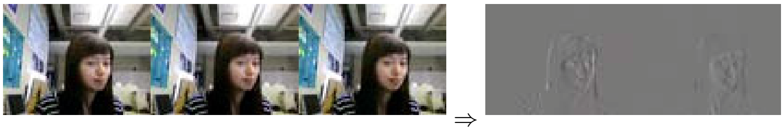


Fig. 4. Sample difference image

4 Evaluation

We evaluate the performance of the system by creating a database of eight different subjects. Each subject was asked to show three gestures (Yes, No, and Neutral) for approximately 30 seconds each.

It is difficult to recognize gestures using adjacent frames in the video sequence since there is only very small differences between successive frames in a video stream at high frame rates. For example, the difference between images is only 33.30ms at 30 frames per second. We therefore create frame sequences by selecting three images that are spaced approximately 200ms apart and discarding the intermediate frames. This results in three images where the first and the middle and the middle and the end frame are 200ms apart.

Approximately one second after the instruction is given to the subject we start to record video and extract approximately 70 frame sequences from the video randomly. This results in approximately 70 frame sequences for each of three gestures (Yes, No, Neutral) for eight subjects or a total of 1572 frame sequences in our database.

To avoid biasing the sequences, the selection and order of subjects was done at random and all other factors (e.g., time of day) was kept as constant as practical.

The algorithms were trained and their generalization ability tested using a n percent holdout method. That is, n percent of the sequences were selected at random and used for training.

SVM only provide binary decision procedures. We therefore use the following common approach to solving the multi-variate decision problem. Three separate SVMs are trained for Yes, No, and Neutral respectively. For example, the SVM for Yes uses sequences of Yes as positive examples and all other sequences (No, Neutral) as negative examples. Similarly we train the No and Neutral SVMs.

After training, the system classifies all 1572 sequences in the database. We measured the accuracy of the system as the percentage of correctly classified sequences in the database.

An sequence is classified by computing its classification using the Yes, No, and Neutral SVM and by comparing the strength of the classifier in the positive region. The sequence is classified as the gesture associated with the SVM that returns the maximum in the positive region.

4.1 Pre-processing

First, we compare the performance of the SVM using various pre-processing steps described in section 3: (a) colour image 80x60, (b) colour sub-sampled image

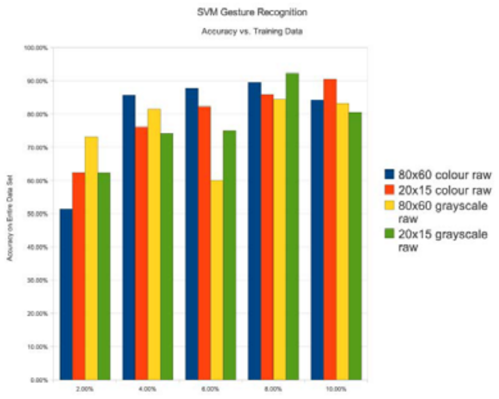


Fig. 5. Accuracy vs. training instances grouped by pre-processing method

20x15, (c) grey-scale 80x60 image, (d) grey-scale sub-sampled image (20x15), (e) difference image 80x60, (f) difference sub-sampled image 20x15.

Figure 5 shows the accuracy of the SVM gesture recognizer with different number of training instances. The influence of the number of training instances is discussed in Subsec 4.2.

The entries for 6%, 8%, and 10% show that there is some variation in the pre-processing methods and that no clear winner emerges. In general the larger image size methods (80x60) seem to perform better than the sub-sampled image methods (20x15), but that difference is not statistically significant. With sufficient training instances, the performance of the four methods described above is above 80%, which is suitable for our application.

4.2 Number of Training Instances

The first characteristic we investigated is the number of training examples needed to train an accurate classifier. Figure 5 shows the accuracy of various pre-processing methods dependent on the number of training examples.

The number of training samples is an important parameter of the algorithm since the run time and storage complexity of an SVM algorithm grows with the square of the number of training samples.

We trained the SVM gesture recognizer using an increasing number of training samples from 2% to 10% of the dataset. This corresponds to 31 to 155 training images out of a total 1572 images in the data set. The accuracy was evaluated using the full 1572 images.

As can be seen, the performance of the system is poor if only two percent of the data set is used as training data, but improves quickly. There is no statistical difference in the performance of the system with 8% (125 images) or 10% (157 images) of training instances.

The effect of increasing the number of training instances on the runtime is shown in Fig. 6.

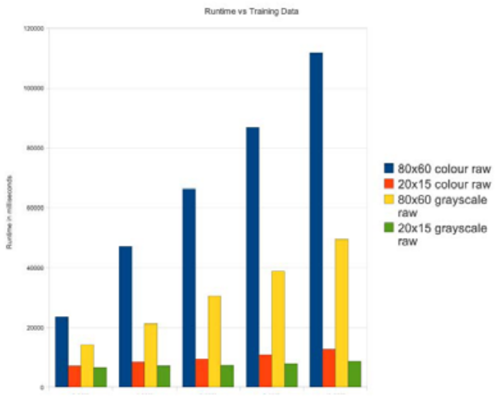


Fig. 6. Runtime vs. training instances grouped by pre-processing method

As can be seen, the runtime grows steeply with increasing the number of training samples for the large image size. Reducing the size of the image greatly reduces the runtime of our algorithm from 1 minute 51 seconds to 8.64 seconds.

4.3 Difference Representation

As can be seen in Fig. 7, the performance of the difference methods is worse than expected. The accuracy on the entire data set is only 40%.

This is due to the fact that as in most robotics applications, calculating the difference between sensor signals greatly increases the suscebitability to noise. So small noisy inputs are amplified by calculating the derivative of the input signal. In this case, the few detected true motion pixels are overshadowed by background noise.

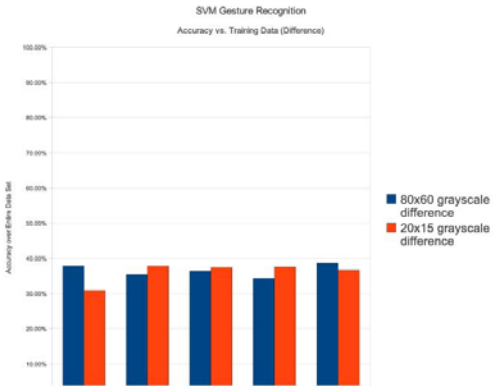


Fig. 7. Accuracy vs. training instances for difference representations

5 Conclusion

We describe a fast and practical SVM learning system for detecting facial gestures indicating Yes, No, or Neural expressions. It is shown that even small images of size 20x15 are suitable to detect the gesture. This means that the system can be combined with a face detector to watch multiple users in a single view.

References

1. Chang, C.-C., Lin, C.-J.: Libsvm: a library for support vector machines. Computer, 1–30 (2001)
2. Hagg, J., Rkl, B., Akan, B., Asplund, L.: Gesture recognition using evolution strategy neural network, pp. 245–248. IEEE, Los Alamitos (2008)
3. Hasanuzzaman, M., Ampornaramveth, V., Bhuiyan, M.A., Shirai, Y., Ueno, H.: Real-time vision-based gesture recognition for human robot interaction. In: 2004 IEEE International Conference on Robotics and Biomimetics, pp. 413–418 (2004)
4. Lee, S.-W.: Automatic gesture recognition for intelligent human-robot interaction. In: 7th International Conference on Automatic Face and Gesture Recognition FGR 2006, pp. 645–650 (2006)
5. Mitra, S., Acharya, T.: Gesture recognition: A survey. IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews 37(3), 311–324 (2007)
6. Oshita, M., Matsunaga, T.: Automatic learning of gesture recognition model using som and svm. Advances in Visual Computing, 751–759 (2010)
7. Valibeik, S., Yang, G.-Z.: Segmentation and Tracking for Vision Based Human Robot Interaction. IEEE, Los Alamitos (2008)
8. Vapnik, V.N.: An overview of statistical learning theory. IEEE Transactions on Neural Networks 10(5), 988–999 (1999)