

Distinguishing Mislabeled Data from Correctly Labeled Data in Classifier Design

Sundara Venkataraman Dimitris Metaxas Dmitriy Fradkin Casimir Kulikowski
Ilya Muchnik

{sundara,dnm,dfradkin,kulikows}@cs.rutgers.edu, muchnik@dimacs.rutgers.edu

Department of Computer Science
Rutgers, The State University of New Jersey
110 Frelinghuysen Rd., Piscataway, NJ 08854-8019

Abstract

We have developed a method for distinguishing between correctly labeled and mislabeled data sampled from video sequences and used in the construction of a facial expression recognition classifier. The novelty of our approach lies in training a single, optimal classifier type (a Support Vector Machine, or SVM) on multiple representations of the data, involving different “discriminating” subspaces. Results of a preliminary study on the discrimination of “high stress” vs. “low stress” facial expression data by this method confirms that our novel approach is able to distinguish subproblems where labeling is highly reliable from those where mislabeling can lead to high error rates. In helping detect data sub-samples which yield misleading classification results, the method is also a rapid, highly efficient cross-validated approach for eliminating outliers.

1. Introduction

In many applications it is necessary to construct a classifier from unreliable training data. This situation can arise not only from mistakes in the labeling process, but also from noisy or inadequately sampled data which may not be sufficiently representative of a correctly labeled complete sample. This often happens when objects are described by some physical measurement obtained as results of experiments, or when features are extracted by only partially-automated process. For example, in speech recognition a human expert has to identify and label parts of a speech signal containing specific sounds. Different experts may do so differently, and even experts may incorrectly specify the boundaries of the signal, or choose non-representative examples.

There is a large amount of literature on detecting outliers, defined as points that have low probability under an assumed model of the data, or that are isolated from neighbors [2], [11]. However, mislabeled or corrupted data will

not necessarily behave in a way consistent with being an outlier, especially if it constitutes a relatively large part of the available data.

The problem of dealing with mislabeled data, and in particular of constructing a classifier from such data, has been approached from a number of different directions. A lot of work has been done on detecting mislabeled data for k Nearest Neighbor Classifiers (kNN), which are particularly sensitive to noise and errors. Several methods are discussed in [14]. Neural Nets (NN) are used to remove or relabel suspicious examples in [10].

Removing points with consistently high weights during training of support vector machines (SVM) has been reported to improve training speed and classification performance of acoustic speech models [8]. A related approach is to remove points whose inclusion in the training set makes its description, for example by a logical rule, significantly more complicated. This approach was effective for noise removal in the diagnosis of coronary artery disease [7].

A very general approach to finding mislabeled data was suggested by Brodley and Friedl in [3]. A number of independent classifiers of different types are trained and evaluated on the available data using cross-validation. Each classifier thus makes a prediction about each point. The points for which the prediction of at least one classifier (consensus filtering) or majority of classifiers (majority vote) do not match the label are not used in training of the final classifier. This approach has been shown to be effective for Part-of-Speech Tagged Corpora [1]. Extensions of this idea that use ensemble-based methods such as bagging and boosting are examined in [15]. It was found that majority vote cross-validation and bagging consensus filters gave showed best results.

In our experiments, instead of using several different types classifiers, or training classifiers on different subsets of the data, we train the same type of classifier (linear SVM) on the data using different parts of the feature space. Intuitively, “discriminating” subspaces are the subspaces that do

not overlap and where the classifier has accuracy comparable to that in the original feature space. In general, such subspaces may be difficult to find explicitly, but in our case they were suggested by the structure of the data, as will be seen in Section 3.3. This approach uses only a single type of classifier and does not require any data sampling. We simply select the subspaces that we are going to use for building classifiers.

For domains where the structure of the feature space is unknown, the use of SVM with different kernels provides an alternative to our method, since use of different kernels amounts to implicitly projecting data into different “kernel spaces.” However we do not examine this approach in the current paper.

We report here on the result of our approach in the context of the problem of training a facial expression recognition system. The data consists of 2-second segments of a longer video sequence. The features characterize the displacement and deformation of a mask that was manually matched to the face and automatically tracked through the segment. This situation is very similar to the speech recognition example. While the correct label for the long sequence is known, the short segment, selected by a non-expert, may not be representative. Furthermore, there may be mistakes in the feature extraction process. Therefore, when trying to build a classifier from the data collected in this way, there can be many unreliable subsamples in the data, mixed with the reliable ones.

Our paper gives a general methodology for determining whether we are dealing with unreliable samples of data, and how these can be filtered from the reliable samples based on classifier performance. We also discuss applicability of our method to recognizing facial expressions and finding reliable data subsets for this problem.

2. A general approach to unreliable data

For classification problems data can be considered reliable if we can build a classifier based on this data that will have high cross-validation accuracy. This assumes that the data uses an adequate representation, and that an appropriate class of classifiers has been chosen. In k -fold cross-validation, data is partitioned in k non-overlapping sets, and the system is trained on $(k - 1)$ sets and evaluated on the remaining “left-out” set. This is repeated using each of the k sets for evaluation. This method is efficient in its use of the training data. Furthermore, it produces an estimate of the classifier’s performance on each object. Literature on cross-validation, its statistical properties and applications to pattern recognition is extensive (ex. [9], [16]).

To build a reliable classifier one needs reliable data and labeling. A few bad objects do not significantly affect the cross-validation accuracy. However, when unreliable data

subsamples are anticipated, they can have a significant impact on the classifier.

Let us assume that all of our data is correct. We can build multiple classifiers on subspaces of the data and in the original space, and estimate their accuracy with cross-validation. Combining results of cross-validation across the classifiers, we obtain, for each object, a statistic on how often it gets misclassified. Good reliable data can be identified as subsets of objects that are all correctly classified in the same (or even in all) subspaces. Once a candidate reliable subset is identified, we can check its quality by building a classifier on it in the original space and performing cross-validation. The results should be better on the whole data.

Once a good initial subset of reliable data is found, it can be improved by trying to add or remove groups of points and evaluating the classifier on the adjusted set. Alternatively, semi-supervised learning methods can be applied, with “good” set labels viewed as true, and “bad” set objects viewed as unlabeled. The task is not to classify all objects, but to extend the scope of the reliable classification. However, analysis of these approaches lies beyond the scope of this paper. In the following sections we only demonstrate that our method for finding initial reliable subsets can be successful.

3. Application to facial expression recognition

Facial Expression Recognition from short image sequences (video) is an important task for human-computer interaction and virtual reality applications. Multiple studies of different methods have been conducted ([17], [13], [6], [12]). We have short video segments of facial expression, labeled by the emotion of the long sequence from which a particular segment is taken. We would like to use these segments to construct a classifier capable of correctly labeling new sequences from new people (i.e. people who were not present in the training sequences). Our approach views each sequence of frames as a single object, and constructs a new representation for the sequences based on the dynamics of the frames. The classifier design and data validation are performed on this new representation.

This section first describes how data was obtained and how the representation of the sequences was constructed and explains possible sources of errors in the training data. The structure of our classifier is then briefly discussed.

3.1. Image sequence preprocessing

Data collection involved videotaping peoples’ faces as they were undergoing various tests devised by psychiatrists. The tests were devised to be of varying difficulty and would induce stress in the subjects. The stress manifested on the

face is captured by the cameras. The video data thus obtained is used for face tracking. The subjects express a wide range of facial movements such as raising eyebrows, twisting lip, half smiles, etc.

For our experiments we had 28 image sequences belonging to two classes of 14 people (1 sequence for each class from each person). The whole image sequence lasts about 30 minutes. In a practical application, a facial expression has to be recognized much faster. Intervals lasting approximately 2 seconds were chosen by a non-expert from each sequence, as the most indicative of the expression. The segments selected varied in length from 44 to 727 frames, with the total number of frames in the 28 sequences being 7236. We note that even when the whole sequence is labeled by a particular expression, it is possible that the expression is not clearly present for some subintervals of time.

3.2. Frame representation

The choice of the features used to represent individual frames is extremely important. On the one hand, extracting such features automatically is a computationally intensive task. On the other hand, while no method will work well with poor features the only way of determining the quality of a feature set is by observing the performance of a recognition system. Practically all of the papers cited above used different features in their recognition systems.

Face tracking was accomplished by a cue integration algorithm. The algorithm integrates cues from edge information, point trackers and optical flow information from the images. A 3D face model which is capable of producing the deformations mentioned above is fit to the subject's face at the starting frame of the tracking. Once the model is fit, the tracking algorithm is run on the images which produces the rigid and deformation parameters. Currently, we have parameters for eyebrow raising, lip stretching, lip curving and jaw movement. This is in addition to the rigid translation and rotation parameters.

In the end, each sequence is an ordered set of frames $S_j = \{x_j^1, \dots, x_j^{|S_j|}\}$, where each frame is represented as a vector in 14 dimensional space ($x_j^t \in R^d, d = 14$).

3.3. Sequence representation and classifier design

Once all frames in each 2-second sequence are represented in the feature space of Section 3.2, we construct vector representations of individual sequences.

We use an HMM methodology in an unsupervised setting to accomplish two tasks. First, by training the HMM on each sequence, we obtain a partition of each sequence into sets of frames associated with different states of the HMM. Secondly, we use the parameters of the HMM associated

with each state, or with the whole HMM, as a new representation of a corresponding set of frames or the whole sequence. Since the new representation consists of vectors of the same dimensionality, we use Support Vector Machines (SVM) [5] to classify sequences in the new feature space.

For each sequence S_i in the training set we construct an HMM H_i with specified number of states, and the Gaussian model for the distribution of observations in each state. We represent the sequence by the vector of parameters of the states of HMM denoted by $r(S_i)$. For example, given a choice of representation $j = 0, \dots, n$, we represent each sequence S_i by a vector $r_j(S_i)$:

$$r_j(S_i) = \begin{cases} \mu_{ij}, & j = \overline{1, n} \\ v_i = [\mu_{i1}, \mu_{i2}, \dots, \mu_{in}], & j = 0 \end{cases} \quad (1)$$

where μ_{ij} is the mean of the distribution of observations at the state j of HMM H_i . (Note that representations $r_j(S_i)$ for $j = \overline{1, n}$ can be seen as a set of non-overlapping subspaces of representation $r_0(S_i) = v_i$). We then train SVM classifier on a set of representative vectors $r(S_i)$.

For a test sequence S_t , we construct its HMM model H_t and its representative vector $r(S_t)$, and apply the SVM classifier to it. The high-level pseudocode for training and testing of the classifiers is given in Algorithm 1 and Algorithm 2.

Algorithm 1 Training Algorithm

Require: A set of k sequences $S_i, i = \overline{1, k}$, number of states $n > 1$.

- 1: **for** $i = 1, \dots, k$ **do**
 - 2: Construct an HMM H_i with n states, based on sequence S_i .
 - 3: Compute representative vector $r(S_i)$.
 - 4: **end for**
 - 5: Train SVM R on the set of vectors $r(S_i), i = 1, \dots, k$.
 - 6: Output resulting classifier R .
-

Algorithm 2 Classification Algorithm

Require: A sequence S_t , number of states $n > 1$.

- 1: Construct an HMM H_t with n states, based on sequence S_t
 - 2: Compute representative vector $r(S_t)$.
 - 3: Return the label given by classifier R to vector $r(S_t)$.
-

4. Validation of facial expression recognition system

For facial expression recognition we can assume that we have several sequences for each person. The cost of obtaining the training data is high. In order to estimate the performance of our system on a new sequence (from a new

Representation	$n = 2$	$n = 3$	$n = 4$
v_i	18 64.3%	19 67.9%	18 64.3%
μ_{i1}	16 57.1%	17 60.7%	18 64.3%
μ_{i2}	15 53.6%	14 50.0%	17 60.7%
μ_{i3}		16 57.1%	17 60.7%
μ_{i4}			17 60.7%

Table 1. Results of cross-validation over all data (28 sequences).

Representation	$n = 2$	$n = 3$	$n = 4$
v_i	8 57.1%	8 57.1%	8 57.1%
μ_{i1}	9 64.3%	9 64.3%	9 64.3%
μ_{i2}	7 50.0%	7 50.0%	7 50.0%
μ_{i3}		8 57.1%	6 42.9%
μ_{i4}			8 57.1%

Table 3. Results of cross-validation over unreliable data (14 sequences).

Representation	$n = 2$	$n = 3$	$n = 4$
v_i	11 78.6%	10 71.4%	11 78.6%
μ_{i1}	12 85.7%	12 85.7%	11 78.6%
μ_{i2}	13 92.9%	10 71.4%	10 71.4%
μ_{i3}		13 92.9%	12 85.7%
μ_{i4}			13 92.9%

Table 2. Results of cross-validation over “good” data (14 sequences).

person), it seems reasonable to use “leave-one-person-out” cross-validation. In other words, we train our system on all sequences except for the ones produced by a single person. Then we evaluate our system on the sequences of that one person. We repeat this for all persons for whom we have the data. This approach should provide us with a realistic estimate of the performance of our system on a sequence from a new person. Furthermore, we obtain classification results on each sequence in the training data. This should enable us to analyze any mistakes and to improve our understanding of the problem and of the behavior of our system.

Since we can construct multiple representations for each sequence by choosing a number of states of HMM, and a specific state, as in (1), we can train classifiers and evaluate their accuracy using each of these representations. We thus obtain a number of predictions for each sequence. Since the size of the data is small, and the process of obtaining the data is expensive, we have to be careful in eliminating the data. An additional consideration is that we don’t want to use sequences of a person unless both of them are reliable. Therefore, we cannot use consensus filters or majority votes as in [3]. We keep all of person’s sequences if both of them have greater than 50% accuracy. Otherwise, all the sequences of that person are removed.

5. Experimental results

In all of our experiments, we use a linear SVM mode of BSVM [4] (parameter settings “-s 2 -t 0”). We experi-

mented with an HMM consisting of $n = 2, 3, 4$ states.

Initially we applied leave-one-person-out cross-validation to all data. The results can be seen in Table 1. The column header contains the number of states of the HMM. The rows correspond to the sequence representation used (see Section 3.3). The cells contains the number (out of 28) and percentage of sequences classified correctly by SVM using the corresponding representation. The results are rather consistent. The accuracy is approximately 16-18 out of 28 for parameters and features, which yields 57% – 64%. The result is clearly better than random guessing, but is not particularly good.

We decided to examine where the mistakes are made. By combining results of all experiments, i.e. over all values of n and choices of representation, we obtained evidence that for 7 out of the 14 persons, one of the sequences was almost always labeled incorrectly. More specifically, we had results of cross-validation prediction for using each of the 12 possible representations on each pair of sequences. For the 7 people one of these sequences would be misclassified at least 10 out of 12 times. We kept the sequences from the other subjects in a “good” set. We then repeated the leave-one-person-out cross-validation only on the 14 sequences of the “good” set. These results are in Table 2. With a considerable reduction of the training set size, the accuracy improved very notably: 10-13 out of 14, or 71% to 93%. This suggests that the sequences in the “bad” set were not simply difficult, but misleading. The sequences in the good set are in fact very consistent with each other in terms of how facial expression is produced.

In order to check whether the “bad” sequences in fact constitute another coherent group we repeated cross-validation experiments on that set. The poor results (Table 3) all yield 50% – 64% performance, and suggest that the sequences in that set do not form a coherent group.

The accuracy results of classifiers trained on the “good” when applied to the “bad” set were all around 50%. We also trained classifiers on the “bad” set and applied them to the good set. The resulting accuracy was in the range 42%-57%. This indicates that the sequences in the “bad” set are not

merely the marginal points, since then a classifier trained on that set would give a good classification performance on the “good” set despite a possibly poor cross-validation performance.

The analysis we conducted illustrates a possible approach to selection of the training data to be used in constructing a good facial expression recognition system.

6. Conclusion

We have suggested a general approach to extracting reliable data from a set of objects that are possibly mislabeled or incorrectly represented. Our approach trains the same classifier on different representations of the data, that can be thought of as different subspaces of the same representation, to obtain statistics on labeling of each point. This sets it apart from existing methods that use different subsets of data to train classifiers or use different types of classifiers [3], [15].

The results of experiments with facial expression recognition data show that this approach is successful in finding a large subset of good objects. This, in effect, shows that the method works as a highly efficient cross-validated approach for eliminating classification outliers. We are currently carrying out experiments with more extensive facial recognition datasets to investigate more deeply the properties of our approach.

A detailed study on a number of benchmark datasets would be necessary to compare our approach with other existing methods and to determine under what circumstances a particular method for finding mislabeled data should be used by itself or in conjunction with other methods.

7. Acknowledgments

The authors thank the anonymous reviewers for their comments. Ilya Muchnik was partially supported through the NSF grant CCR 035398. Dmitriy Fradkin thanks the KD-D group for its support through National Science Foundation grant number EIA-0087022 to Rutgers University. The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency.

References

- [1] H. Berthelsen and B. Megyesi. Ensemble of classifiers for noise detection in pos tagged corpora. In *Proceedings of Third International Workshop on Text, Speech and Dialogue*, 2000.
- [2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *Proceedings of ACM SIGMOD*, pages 93–104, 2000.
- [3] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–187, 1999.
- [4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [5] N. Christianini and J. Shawe-Taylor. *An Introduction to support vector machines and other kernelbased learning methods*. Cambridge University Press, 2000.
- [6] I. Cohen, N. Sebe, L. Chen, A. Garg, and T. S. Huang. Facial expression recognition from video sequences: Temporal and static modeling. *Computer Vision and Image Understanding*, 91(1-2), July-August 2003.
- [7] D. Gamberger, N. Lavrač, and C. Grošelj. Experiments with noise filtering in a medical domain. In *Proceedings of ICML*, pages 143–151, 1999.
- [8] A. Ganapathiraju and J. Picone. Support vector machines for automatic data cleanup. In *Proceedings of the International Conference of Spoken Language Processing*, pages 210–213, 2000.
- [9] J. S. U. Hjorth. *Computer Intensive Statistical Methods: Validation, Model Selection and Bootstrap*. Chapman and Hall, New York, USA, 1994.
- [10] Y. Jiang and Z.-H. Zhou. Editing training data for knn classifiers with neural network ensemble. In *Proceedings of ISNN'04*, 2004.
- [11] E. M. Knorr and R. T. Ng. A unified notion of outliers: Properties and computation. In *Knowledge Discovery and Data Mining*, pages 219–222, 1997.
- [12] P. Michel and R. E. Kaliouby. Real time facial expression recognition in video using support vector machines. In *Proceedings of ICMI'03*, pages 258–264, 2003.
- [13] T. Otsuka and J. Ohya. Recognizing multiple persons' facial expressions using hmm based on automatic extraction of significant frames from image sequences. In *Proceedings of ICIP'97*, October 1997.
- [14] J. Sanchez, R. Barandela, A. Marques, R. Alejo, and J. Badesnas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24:1015–1022, 2003.
- [15] S. Verbaeten and A. V. Assche. Ensemble methods for noise elimination in classification problems. In *Proceedings of 4th International Workshop on Multiple Classifier Systems*, 2003.
- [16] S. M. Weiss and C. A. Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc., 1991.
- [17] Y. Yacoob and L. S. Davis. Recognizing human facial expressions from long image sequences using optical flow. *IEEE PAMI*, 18(6):636–642, June 1996.