Discovery of Informative Unlabeled Data for Improved Learning

Weijun He[1], Xiaolei Huang[1], Gabriel Tsechpenakis[1], Dimitris Metaxas[1], and Carol Neidle[2]
[1]Center for Computational Biomedicine Imaging and Modeling,
Division of Computer and Information Sciences, Rutgers University, NJ, USA
[2]Department of Modern Foreign Languages and Literatures,
Boston University, MA, USA

Abstract

In computer vision, the acquisition of sufficient labeled data for training is often time-consuming. However, unlabeled data are conveniently available. The key problem is to discover and incorporate those informative and confidently predicted unlabeled data into the training set for improved learning. In this paper, we discover such unlabeled data by exploiting the locality property of the data. The locality property means that with high probability one example shares the same label with its near neighbors. Thus the label of those informative unlabeled data may be learned from their confidently predicted neighbors. In many computer vision applications, each individual feature set may not be sufficient by itself for the learning purpose. Therefore, unlabeled data can not be included in the co-training way. In addition, there are different types of feature sets with very different information in computer vision that can not be effectively combined and used within a single classifier. Our method combines the predictions from multiple classifiers on different feature sets. Based on these predictions, we discover informative unlabeled data with confident predictions by exploiting the locality property. We apply our learning framework to the segmentation of the fingerspelled signs from an American Sign Language (ASL) video sequence. The experimental results show that our method improves the segmentation accuracy by including the chosen unlabeled data.

1. Introduction

For many supervised learning algorithms in computer vision, the cost of acquiring labeled data is prohibitively high. This "High Initial Expenses" issue is discussed and tackled in [8] using the co-training method [2]. The assumptions in co-training are i) each example has two redundant but not completely correlated feature sets and ii) each feature set would be sufficient for learning if enough data were available. Under these assumptions, the predictions of one classifier on new unlabeled examples are expected to generate informative examples to enrich the training set of the other. However, these formal assumptions may not hold in applications of high complexity and data dimensionality, such as many computer vision applications.

A co-training algorithm using confidence-rated predictions is proposed in [8] and applied to the task of automobile detection in roadway surveillance video. The algorithm uses Adaboost [5, 11] to train two separate classifiers on two different feature sets, and assumes the margins of the two classifiers are only weakly related. The key idea is to add only those unlabeled examples which are confidently labeled by one classifier to the training set of the other classifier. The experiments showed improved performance using this co-training algorithm compared to using labeled examples alone, and empirically demonstrated that even two closely related classifiers can be co-trained effectively. Another algorithm that combines clustering and co-training to enhance text classification is proposed in [10]. The algorithm uses Support Vector Machines [4, 12] as predictors on separate feature sets, with one trained using data from the original feature space, the other one with new features that are derived by clustering both labeled and unlabeled data. However, in all these extensions of the co-training algorithm, we assume that the individual feature set would be sufficient for the learning purpose if enough data were available. This assumption may not hold in many computer vision applications. However, the spatio-temporal pattern among data, which is common in computer vision, is not well exploited. For instance, in the recognition of events in video sequences, the label of one image frame is highly related to the labels of its neighboring frames. So we can utilize the spatio-temporal pattern among the unlabeled data when combining labeled and unlabeled data in learning.

In this paper, we present a new learning framework to utilize the unlabeled data which have some kind of spatio-temporal locality property. The main idea is that the label of those informative unlabeled data may be discovered through their confidently predicted neighbors. Including into the training set these informative unlabeled data with labels learned from neighbors improves prediction accuracy of our classifiers. We also use multiple feature sets for each

example. Each feature set has different capability and is not sufficient by itself for the learning purpose. Thus, we train the classifiers on these feature sets separately and combine the results based on their prediction accuracy on a validation set.

We apply this new learning framework to the segmentation of the fingerspelled signs from an American Sign Language (ASL) video sequence. The key idea is to apply our classification method to a frame window of five, which serve as a single example because the inter-frame changes may reveal whether the sign is fingerspelling or not. Since the video sequences have the property of temporal locality, this new learning framework is well suited to exploit it.

The main contributions of this paper are: (1) a new learning framework for discovering informative unlabeled data to improve learning by exploiting spatio-temporal pattern among data; (2) extension of the co-training method to multiple feature sets using the boosting-like weighting.

2. Methodology

There are two levels of learning in our method. First, classification of the unlabeled data can be learned from the predictions from the classifiers based on different feature sets. According to our framework, different types of classifiers can be used. We combine the predictions from these classifiers based on their respective prediction accuracy on a validation data set in a boosting-like procedure. Second, the data can themselves learn their labels from a weighted voting from their near neighbors since the locality property is assumed.

The data are in four categories: training data, validation data, unlabeled data and testing data. The validation data are used to measure the prediction accuracy of each classifier, and these measurements are then used to evaluate the boosting-like weights for each classifier.

2.1. Learning Framework

For each unlabeled data set, we first combine the predictions of classifiers based on their prediction accuracy on the validation data. For this purpose, we use a boosting-like method described below. Then we "smooth" the weighted prediction using a discrete Gaussian kernel. Finally we select the examples that are most confidently labeled as positive or negative and add them into the training data. These data may include some of the *salt-and-pepper noise*, which are actually wrongly labeled (with high probability) by the current classifier. So the classifiers can improve their performance by including these data along with the correct labels.

Fig. 1 illustrates our learning framework. Here each small box represents the prediction for an unlabeled example. Our method can be applied to data where the locality property exists.

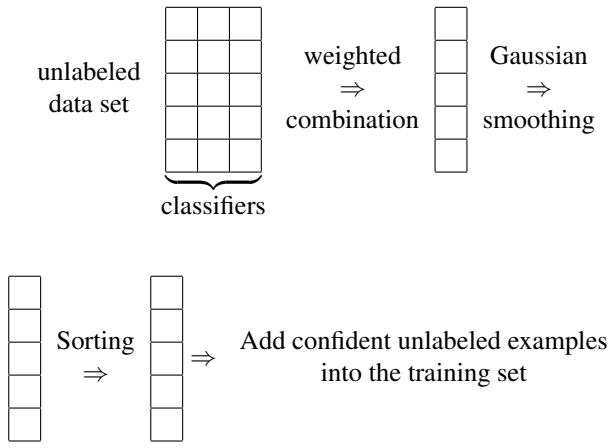


Figure 1: Our learning framework

In the first pass we combine the predictions from different classifiers using boosting-like weights. In the second pass we smooth the result with a discrete Gaussian kernel. Our aim is to learn an example's label from its near neighbors (the voting result from its neighbors may indicate a possible prediction error in the *salt-and-pepper* data). Finally we include those confident unlabeled examples into the training set and retrain the classifiers.

Let $\mathcal{L}_i(X), i = 1, \ldots, n$ denote the predictions of classifier i on data X and $\mathcal{L}(X) = (\mathcal{L}_i(X), i = 1, \ldots, n)$. Denote $\mathcal{W}$ and $\mathcal{S}$ as the boosting-like weighting and Gaussian smoothing operators respectively (we will define these operators in the following subsections). We define operators $\mathcal{F}$ as:

$$\mathcal{F}(X) = \mathcal{S}(\mathcal{W}(\mathcal{L}(X))) \tag{1}$$

We sort the unlabeled data $u_j \in U$ according to the value of $\mathcal{F}$ and include the p most confident positive examples and n most confident negative examples into the training set.

After we re-train the classifier $i = 1, \ldots, n$ on the enriched training set, our final hypothesis $\mathcal{H}$ is:

$$\mathcal{H}(X) = \text{ sign } (\mathcal{F}(X)) \tag{2}$$

Fig. 2 is the pseudo-code description of our learning framework.

2.2. Weighted Combination

Since we do not assume that each classifier on different feature sets is strong enough by itself for classification, we need some voting mechanism to combine the predictions from different classifiers. Voting classification algorithms, such as Bagging and AdaBoost [3, 5, 9, 11], improve the accuracy by combining the (weak) classifiers. An empirical

1. Input parameters: standard deviation σ of discrete Gaussian kernel
2. Train Classifier $i, i = 1, \ldots, n$ on feature set i
3. ϵ_i is the prediction error on validation data.
4. $\alpha_i = \frac{1}{2}\ln(\frac{1-\epsilon_i}{\epsilon_i})/C$
5. For each unlabeled set
6. Predict: $L = \mathcal{L}(x)$
7. Weighted combination: $W = \mathcal{W}(L)$
8. Gaussian smoothing $S = W * G_\sigma$
9. Sorting on S
10. Add confident unlabeled data to the training set
11. Re-train Classifier $i, i = 1, \ldots, n$
12. Update ϵ_i and α_i
13. EndFor
14. Output: Classifier i, ϵ_i and $\alpha_i, i = 1, \ldots, n$

Figure 2: Pseudo-code of our learning framework

comparison of voting classification algorithms is given in [1].

In our learning framework, since different feature sets have different discriminative capabilities, we adopt the boosting-like weighting of [11] for the separate classifiers. Instead of evaluating the classifiers on the training set, we keep a separate validation set for evaluation, which is more accurate.

Suppose prediction error on the validation set for classifier $i, i = 1, \ldots, n$ is ϵ_i, then the boosting-like weighting operator $\mathcal{W}$ is defined as:

$$\mathcal{W} = \sum_{i=1}^{n} \alpha_i \times \mathcal{L}_i, \tag{3}$$

where

$$\alpha_i = \frac{1}{2}\ln(\frac{1-\epsilon_i}{\epsilon_i})/C \tag{4}$$

and C is a normalization factor such that $\sum_{i=1}^{n} \alpha_i = 1$.

2.3. Discover Informative Unlabeled Data

If we just add into the training set the unlabeled data which are predicted with high confidence, they do not improve a lot the prediction accuracy of the classifiers since they are not informative. We tackle this problem of "informative" unlabeled data by the idea of learning from their neighbors. We assume that the data have spatio-temporal relationships, particularly the locality property, i.e., the label of one example is highly related to that of its neighbors. We exploit this locality property and label those informative but not confidently predicted examples among non-informative but confidently predicted examples. Thus, our method combines the ideas from both classification and segmentation since segmentation is not done on the original data; instead, segmentation is applied to the weighted labels (or weighted margins, if the predictions are margins).

Our method can find many applications as long as the data has the locality property. Actually this locality property has been well exploited in the compression schemes, but not as well in the learning methods. The locality property can be exploited to discover those informative but not confidently predicted unlabeled data. Suppose one example is not confidently predicted but most of its neighbors are confidently predicted to belong to a specific class; we label this example with the same class because of the locality property. Including such unlabeled examples will improve the prediction accuracy of each classifier by enriching the training data set.

Suppose we are doing a two-class classification problem (class +1 and class -1). The prediction of one example can be in the range of $[-1, +1]$. The closer the prediction to $+1$, the more probable the example belongs to class $+1$, and vice versa. We can use Gaussian smoothing to approximately model the locality property as:

$$\mathcal{S}(W) = W * G_\sigma, \tag{5}$$

where G_σ is a discrete Gaussian kernel with standard deviation σ and $*$ is the convolution operator.

Here a discrete Gaussian kernel is used to express the locality property. We apply Gaussian smoothing to the prediction results instead of the original data. The *salt-and-pepper* data are examples wrongly labeled (with high probability) by the current classifier. After smoothing, these errors are corrected. So if we include those examples with the correct labels into the training data, the new classifier will learn from these errors and then improve its prediction accuracy.

After the predictions are smoothed with Gaussian kernels, we select those confidently predicted examples for inclusion into the training set. First we sort the unlabeled examples according to the smoothed predictions. Then we select the p most confident positive and n most negative examples and add them into the training set.

3. Segmenting fingerspelled signs from an ASL video sequence

In American Sign Language (ASL) and other sign languages, signs such as names and other borrowings from the spoken language, are expressed by fingerspelling [13]. In fingerspelling, the hand shapes correspond to the letters of the alphabet. Thus, the recognition methods for fingerspelling [6] are quite different from those used for other type of signs. Therefore segmenting the fingerspelling phase from the continuous signing phase in an ASL video is a

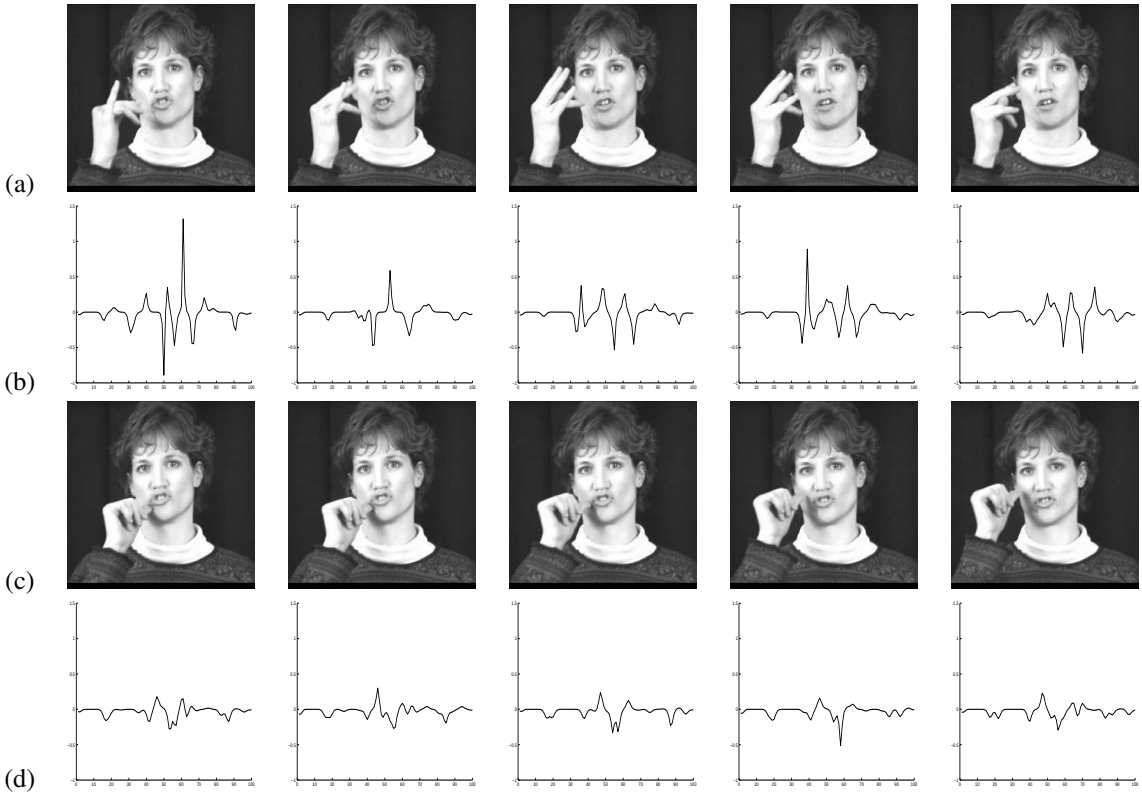


Figure 3: (a, b) fingerspelling of "John" and the corresponding hand contour curvature. The changes of curvature (Euclidean distance) between successive frames are: [2.2469, 1.6506, 1.8364, 1.6119]; (c, d) continuous sign "yesterday" and the corresponding hand contour curvature. The changes are: [0.4863, 0.7447, 0.6425, 0.6404].

step that must be completed before we apply the recognition methods. This segmentation problem can be tackled through a classification method. Given an ASL video sequence, we want to classify every frame as fingerspelling or not.

Fig. 3(a) shows a five-frame sequence during a fingerspelling phase (the word "John"). Fig. 3(c) shows the continuous signing "yesterday". We notice that some discriminative features are the styles of the inter-frame changes. For example, more rapid movements of individual fingers are generally found in fingerspelling. Given that the hand shapes for letters may also be used in other types of signs, the inclusion of dynamic information is essential. For example, in Fig. 3(c), we can not tell whether a single frame corresponds to the continuous sign "yesterday" or the fingerspelling letter "a" (which has similar hand shape). We know fingerspelling involves rapid movement of individual fingers, which causes the rapid change of hand contours. Since the consecutive five frames in Fig. 3(c) have little change of the hand contours, we can conclude that it is for the continuous sign "yesterday". So in our learning method, the features sets should encode both the static information like the hand shape and the dynamic information like the motion of hand.

We encode five consecutive frames as a single training example to determine whether the middle frame is fingerspelling or not. In our experiment, we track the movement of the dominant hand and extract the hand contour using Snakes [7]. We use the following two feature sets for our learning framework.

Curvature of the hand contour in the middle frame of a five-frame video segment. Let $C_i^l = (x_i^l, y_i^l), i = 1, \ldots, 100$ denote 100 equally spaced hand contour points of frame l. The curvature $K_i^l, i = 1, \ldots, 100$, is defined as:

$$K_i^l = \frac{\dot{x}_i^l \ddot{y}_i^l - \ddot{x}_i^l \dot{y}_i^l}{[(\dot{x}_i^l)^2 + (\dot{y}_i^l)^2]^{3/2}}, \tag{6}$$

where $\dot{x}_i^l$, $\dot{y}_i^l$, $\ddot{x}_i^l$ and $\ddot{y}_i^l$ are the discrete approximation of the first and second order derivatives for contour point i on frame l. The curvature function is obtained after arranging the hand contour points in a clockwise order, always starting from the same position of the hand.

Changes of curvature of the hand contour between five consecutive frames (4 feature variables). Since point correspondence of the hand contour between two successive frames are approximately maintained, the total change

of curvature can be computed as the Euclidean distance between two curvatures.

$$dK^l = \left(\sum_{i=1}^{100} \left(K_i^{l+1} - K_i^l \right)^2 \right)^{1/2} \quad (7)$$

Thus, the two feature sets for a video segment of five frames $(l-2, l-1, l, l+1, l+2)$ are $\{(K_i^l, i = 1, \ldots, 100), (dK^{l-2}, dK^{l-1}, dK^l, dK^{l+1})\ \}$. The curvature feature set K describes the static configuration of the hand while the change dK describes the motion of hand.

Fig. 3(b,d) show the hand contour curvatures for Fig. 3(a,c) respectively. We can see that curvature of fingerspelling is quite different from that of continuous signing. This is generally the case since fingerspelling involve a lot of individual finger movement, which results in more curved contour. Another observation is that the changes of curvature between successive frames are larger in fingerspelling than in continuous signing. This result is caused by the fast finger movement in fingerspelling.

However, we notice neither of these two feature sets is strong enough by itself for a good classifier. These two feature sets have different prediction accuracy and are not tightly correlated to each other. So we use weighted combination of the predictions of the classifiers which are trained on these two feature sets separately.

4. Experiments

We use Support Vector Machines (SVM) as the base classifier on each feature set. A polynomial (degree = 3) kernel is used. The standard deviation and support of the discrete Gaussian kernel are 2 and 4 respectively in our experiments. The ASL video data are from the National Center for Sign Language and Gesture Resources at Boston University. In our experiments, we use three labeled videos and choose same number of positive examples and negative examples as the original training data set. One labeled video is left as the validation data set and one unlabeled video is used as the unlabeled data set. We include 8 most postive unlabeled examples and 8 most negative unlabeled examples into the training set and retrain each classifier. We test our learning framework on two separate labeled testing sets to show the effectiveness of including the chosen unlabeled data. The average length of each video sequence is 40 frames.

In experiment 1 (Fig. 4), the testing video includes one fingerspelling phase (word "John" frames 8-15). In experiment 2 (Fig. 5), the testing video includes two fingerspelling phases (words "Mary" frames 8-16 and "John" frames 24-33). Both figures show that after including the chosen unlabeled sequence, the predictions in the final stage get closer to true labels. We can also see that how the classifier improves the prediction accuracy through learning from its neighbors (temporal locality property). For example, in Fig. 4(d1), the weighted prediction of frame 11 is below zero (nonfingerfelling), which is not correct. However the predictions of all its near neighbors are +1 (fingerspelling). After smoothing with Gaussian kernel in Fig. 4(e1), the classifier corrects this frame as a fingerspelling frame. A similar example can also be found in frame 11 of Fig. 5(d1,e1). Those examples are wrongly predicted by the weighted prediction. So including these examples along with their correct labels will help improve the classifiers.

In Table 1, we can see that after including the chosen unlabeled examples, both the base classifiers and final classifier improve the prediction accuracy. This shows the effectiveness of our approach. We can also see that classifier 2 has higher prediction accuracy than classifier 1. In Table 2, our learning method assigns higher weight to classifier 2 after including the unlabeled data. This shows that the unlabeled data not only help improve the individual classifiers, but also help assign more accurate weight to each base classifier. In Tables 3 and 4, the same experiments are done without the smoothing step, that is, no locality property is utilized. The results show that for experiment 2, the final prediction accuracy does not improve with the inclusion of unlabeled examples. This demonstrates that the utilization of locality property is essential in our learning framework.

Table 1: Prediction Accuracy

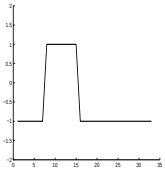
Experiment #	Classifier (on K)	Classifier (on dK)	Final
1. w/o unlabeled data	0.606	0.818	0.758
1. w/ unlabeled data	0.788	0.939	0.879
2. w/o unlabeled data	0.500	0.725	0.625
2. w/ unlabeled data	0.525	0.800	0.925

Table 2: Weights for combination

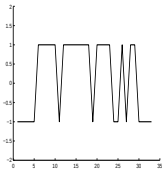
Experiment #	Classifier (on K)	Classifier (on dK)
1. w/o unlabeled data	0.633	0.367
1. w/ unlabeled data	0.567	0.463
2. w/o unlabeled data	0.602	0.398
2. w/ unlabeled data	0.332	0.668

Table 3: Prediction Accuracy (without smoothing)

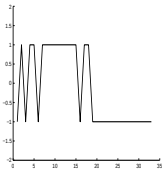
Experiment #	Classifier (on K)	Classifier (on dK)	Final
1. w/o unlabeled data	0.606	0.818	0.606
1. w/ unlabeled data	0.788	0.909	0.909
2. w/o unlabeled data	0.500	0.725	0.500
2. w/ unlabeled data	0.500	0.750	0.500



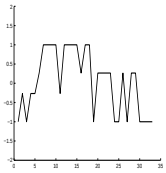
(a) true labels $C(X)$ for testing data X



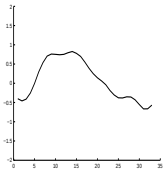
(b0) $\mathcal{L}_1(X)$



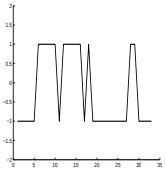
(c0) $\mathcal{L}_2(X)$



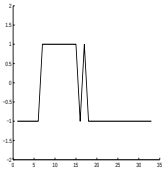
(d0) $\mathcal{W}(\mathcal{L}(X))$



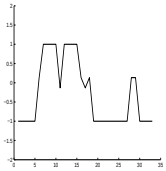
(e0) $\mathcal{S}(\mathcal{W}(\mathcal{L}(X)))$



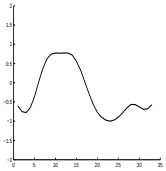
(b1) $\mathcal{L}_1(X)$



(c1) $\mathcal{L}_2(X)$



(d1) $\mathcal{W}(\mathcal{L}(X))$



(e1) $\mathcal{S}(\mathcal{W}(\mathcal{L}(X)))$

Figure 4: Experiment 1. y-axis: high(+1) denotes fingerspelling , low(-1) denotes continuous signing; x-axis: frames number 1-33. (a) Labeled testing data with one fingerspelling phase (word "John" frames 8-15); (b0)-(e0) predictions of the SVM classifier based on curvature, the SVM classifier based on curvature changes, weighted combination and Gaussian smoothing with only the training data; (b1)-(e1) respective predictions after including the chosen unlabeled data

Table 4: Weights for combination (without smoothing)

Experiment #	Classifier (on K)	Classifier (on dK)
1. w/o unlabeled data	0.633	0.367
1. w/ unlabeled data	0.500	0.500
2. w/o unlabeled data	0.602	0.398
2. w/ unlabeled data	0.602	0.398

5. Summary and Conclusions

The main idea in this paper is to discover those informative but not confidently predicted unlabeled data by exploiting the locality property of the data found in many computer vision applications. Those informative unlabeled data learn their labels through their neighbors and including those data into the training set helps enrich the training set, and thus improves the prediction accuracy of the final classifier. We present a learning framework in this paper that utilize this idea. This learning framework naturally extends the co-training method to multiple feature sets without the assumption that each feature set is sufficient for the learning purpose. The experimental results show that the prediction accuracy of the final classifier is improved by including the chosen unlabeled data.

Our future work includes: 1) for the learning framework, using confidence-rated base classifiers (If a classifier can predict a confidence value, instead of just -1 and $+1$, then the weighted combination will be more accurate); 2) for the fingerspelling segmentation, including both hands in the learning framework since ASL is usually performed by both hands (dominant and non-dominant).

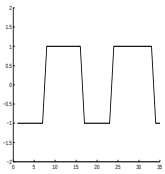
References

[1] E. Bauer and R. Kohavi. An empirical comparison of voting classification problems: Bagging, boosting and variants. *Machine Learning*, 36:105–142, 1999.

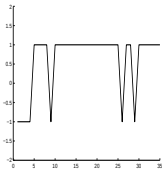
[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of the 1998 Conf. on Computational Learning Theory*, pages 92–100, 1998.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

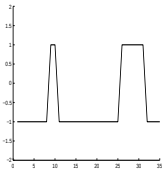
[4] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other kernel-based methods*. Cambridge University Press, 2000.



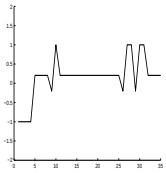
(a) true labels $C(X)$ for testing data X



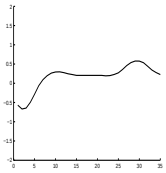
(b0) $\mathcal{L}_1(X)$



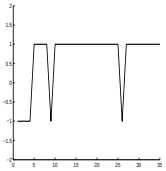
(c0) $\mathcal{L}_2(X)$



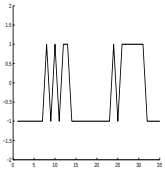
(d0) $\mathcal{W}(\mathcal{L}(X))$



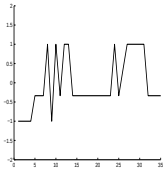
(e0) $\mathcal{S}(\mathcal{W}(\mathcal{L}(X)))$



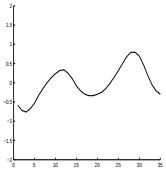
(b1) $\mathcal{L}_1(X)$



(c1) $\mathcal{L}_2(X)$



(d1) $\mathcal{W}(\mathcal{L}(X))$



(e1) $\mathcal{S}(\mathcal{W}(\mathcal{L}(X)))$

Figure 5: Experiment 2. y-axis: high(+1) denotes fingerspelling , low(-1) denotes continuous signing; x-axis: frames number 1-40).(a) Testing data with two fingerspelling phase (word "Mary" frames 8-16 and word "John" frames 24-33); (b0)-(e0) predictions of the SVM classifier based on curvature, the SVM classifier based on curvature changes, weighted combination and Gaussian smoothing with only the training data; (b1)-(e1) predictions after including the chosen unlabeled data

[5] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[6] K. Grobel and H. Hienz. Video-based recognition of fingerspelling in real time. In *Workshops Bildverarbeitung fur die Medizin*, pages 197–331, 1996.

[7] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[8] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual dectectors using co-training. In *Proc. of the Int'l Conf. on Computer Vision*, pages 626–633, 2003.

[9] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proc. AAAI-96 Fourteenth National Conf. on Artificial Intelligence*, pages 725–730, 1996.

[10] B. Raskutti, H. Ferra, and A. Kowalczyk. Combining clustering and co-training to enhance text classification using unlabelled data. In *Proc. of the eighth ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, pages 620–625, 2002.

[11] R. E. Schapire. A brief introduction to boosting. In *Proc. of the 16th Int'l Joint Conf. on Artificial Intelligence*, pages 1401–1406, 1999.

[12] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[13] S. Wilcox. *The phonetics of Fingerspelling*. John Benjamins Publishing Co., 1992.