Improving Generalization by Data Categorization

Ling Li, Amrit Pratap, Hsuan-Tien Lin, and Yaser S. Abu-Mostafa

Learning Systems Group, California Institute of Technology, USA

Abstract. In most of the learning algorithms, examples in the training set are treated equally. Some examples, however, carry more reliable or critical information about the target than the others, and some may carry wrong information. According to their intrinsic margin, examples can be grouped into three categories: typical, critical, and noisy. We propose three methods, namely the selection cost, SVM confidence margin, and AdaBoost data weight, to automatically group training examples into these three categories. Experimental results on artificial datasets show that, although the three methods have quite different nature, they give similar and reasonable categorization. Results with real-world datasets further demonstrate that treating the three data categories differently in learning can improve generalization.

1 Introduction

Machine learning is an alternative approach to system design. Instead of the conventional way of mathematically modeling the system, the role of learning is to take a dataset of examples, such as input-output pairs from an unknown target function, and synthesize a hypothesis that best approximates the target. The dataset, acting as the information gateway between the target and the hypothesis, is thus at the heart of the learning process.

Generally, every example in the dataset is treated equally and no example is explicitly discarded. After all, each example carries its own piece of information about the target. However, if some of the examples are corrupted with noise, the information they provide would be misleading. In this case, it is better to identify and remove them, which can be performed either explicitly by an outlier detection preprocessing step [1], or implicitly in the learning algorithm via regularization [2].

Even in cases where all the examples are noiseless, there are situations in which we want to deal with examples differently. For instance, in large datasets which often contain redundant examples, a subset of informative examples that carries most of the information is usually more desirable for computational reasons [3]. In cases where none of the hypotheses can perfectly model the target, it is better to discard examples that cannot be classified correctly by any hypothesis as they may "confuse" the learning [4].

Most existing methods that treat examples differently tend to group examples into two categories based on different criteria: consistent vs. inconsistent (outliers) [1], easy vs. hard [5], typical vs. informative [3], etc. In this paper,

A. Jorge et al. (Eds.): PKDD 2005, LNAI 3721, pp. 157–168, 2005.

we introduce the concept of *intrinsic margin* as a criterion for grouping data and motivate the need to have three categories instead of two. We present three methods to automate the categorizing. We show that by treating examples in these three categories differently, we can improve the generalization performance of learning on real-world problems. In addition, the categorization can be used to reduce the dataset size without affecting the learning performance.

The paper is organized as follows. The formal framework of learning is defined in Sect. 2. Then in Sect. 3, we introduce the concept of data categorization and present our methods for automatic categorization. Results on artificial and real-world datasets are presented in Sects. 4 and 5. We finally conclude in Sect. 6.

2 Learning Systems

In learning problems, *examples* are in the form of input-output pair $(\mathbf{x}, y)$. We assume that the input vectors $\mathbf{x} \in \mathcal{X}$ are generated independently from an unknown probability distribution $P_{\mathcal{X}}$, and the output labels $y \in \mathcal{Y}$ are computed from $y = f(\mathbf{x})$. Here the unknown function $f\colon \mathcal{X} \to \mathcal{Y}$ is called the *target function*. In this paper, we shall only focus on binary classification problems, in which $\mathcal{Y} = \{-1, 1\}$. We further assume that f comes from thresholding an *intrinsic function* $f_r\colon \mathcal{X} \to \mathbb{R}$, i.e., $f(\mathbf{x}) = \operatorname{sign}(f_r(\mathbf{x}))$, where the magnitude of $f_r(\mathbf{x})$ corresponds to the reliability of the output $f(\mathbf{x})$. For example, if the target function $f(\mathbf{x})$ indicates whether a credit card should be issued to a person $\mathbf{x}$, the intrinsic function $f_r(\mathbf{x})$ could be the aligned credit score of the person.

For a hypothesis $g\colon \mathcal{X} \to \mathcal{Y}$ and an example $(\mathbf{x}, y)$, a commonly used error measure (loss function) is

$$e(g(\mathbf{x}), y) = [g(\mathbf{x}) \neq y] ,$$

where the Boolean test $[\cdot]$ is 1 if the condition is true and 0 otherwise. Then, for a target function f, we can define the *out-of-sample error* of g as

$$\pi(g) = \mathrm{E}_{\mathbf{x} \sim P_{\mathcal{X}}} \left[e(g(\mathbf{x}), f(\mathbf{x})) \right] .$$

The goal of learning is thus to choose a hypothesis g that has a low out-of-sample error $\pi(g)$ from a set of candidate hypotheses, namely the *learning model* $\mathcal{G}$.

However, $\pi(g)$ cannot be directly computed because the distribution $P_{\mathcal{X}}$ and the target function f are both unknown. The only information we can access is often limited in the *training set* $\mathcal{D}$, which consists of N examples $(\mathbf{x}_i, y_i)$, $i = 1..N$. Thus, instead of looking for a hypothesis g with low $\pi(g)$ values, a learning algorithm may try to find g that minimizes an estimator of $\pi(g)$. A commonly used estimator is the *in-sample error* $\nu(g)$ on the training set $\mathcal{D}$,

$$\nu(g) = \nu(g, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} e\left(g(\mathbf{x}_i), y_i\right) .$$

For a fixed hypothesis g, $\nu(g)$ is an unbiased estimator of $\pi(g)$, and when the size of $\mathcal{D}$ is large enough, statistical bounds guarantee that $\nu(g)$ and $\pi(g)$ would not differ by too much.

Note that the learning algorithm searches the whole learning model $\mathcal{G}$ for a suitable hypothesis rather than focusing on a fixed one. In this case, the probability that $\nu(g)$ and $\pi(g)$ differs for some $g \in \mathcal{G}$ gets magnified by the complexity of $\mathcal{G}$. Thus, the hypothesis found might fit the training set well while still having a high out-of-sample error [2]. This situation is called *overfitting*, which arises when good in-sample predictions do not relate to good out-of-sample predictions. The situation can become worse when the examples contain noise. Then, fitting the training set well means fitting the wrong information, which leads to bad out-of-sample predictions.

Learning algorithms often try to avoid overfitting through regularization [2]. Regularization usually enforces a trade-off between the complexity of $\mathcal{G}$ and the necessity to predict the training examples correctly. If we can characterize the usefulness of each training example, the learning algorithm can then be guided to focus on predicting important examples correctly, leading to a more meaningful regularization trade-off and thus a better generalization performance. This motivates our work to categorize the training examples.

3 Data Categorization

The purpose of data categorization is to group examples according to their usefulness to learning so that it is possible to treat them differently. Guyon et al. [3] grouped data into two categories, typical and informative. However, they found that the category of informative examples contained both useful examples and noisy ones. Thus, they needed human-based post-processing to eliminate the noisy examples. Similar problems are encountered in other methods that use two-group categorization. This shows that we need to have more than two categories. In this paper, we fit the need by having three categories: typical, critical and noisy.

Although all examples carry information about the target, they are not equal in the sense that some examples carry more useful information about the target than others, and some examples may misguide the learning algorithm. For instance, in classification problems, an example close to the class boundary gives more critical information than an example deep in the class territory. In addition, real-world data often contain mislabeled examples, which compromise the ability of the in-sample error to approximate the out-of-sample error and lead to bad generalization.

One way to categorize examples based on the above intuition is through the concept of *intrinsic margin*. For an example $(\mathbf{x}, y)$, its intrinsic margin is $y f_r(\mathbf{x})$, where f_r is the implicit intrinsic function defined in Sect. 2. Under some reasonable smoothness assumption, the intrinsic margin can be treated as a measure of how close the example is to the classification decision boundary. If the intrinsic margin is small positive, the example lies near the decision boundary and should be categorized as critical. If the margin is large positive, the example is far from the boundary and should be categorized as typical. Examples with negative intrinsic margin are mislabeled, and should be classified as noisy. Thus, we may use two thresholds, 0 and some small positive value, to partition the intrinsic margin and categorize the data.

In practical situations, it is impossible to calculate the intrinsic margin unless the intrinsic function is known. However, since we are only interested in thresholding the intrinsic margin, any monotonic function of the intrinsic margin can be used with appropriate thresholds. Next, we propose three different methods to estimate such functions for automatically categorizing the data.

3.1 Selection Cost

Bad generalization arises when the in-sample error is a bad indicator of the out-of-sample error. A particular example $(\mathbf{x}, y)$ may deteriorate the generalization performance if its error is a bad indicator of the out-of-sample error. Based on this intuition, Nicholson [4] suggested to use the correlation coefficient between $e(g(\mathbf{x}), y)$ and $\pi(g)$ under a prior distribution $P_{\mathcal{G}}$ for g,

$$\begin{aligned}\rho(\mathbf{x}, y) &= \operatorname{corrcoef}_g\left[e(g(\mathbf{x}), y), \pi(g)\right] \\ &= \frac{\mathrm{E}_g\left[e(g(\mathbf{x}), y)\pi(g)\right] - \mathrm{E}_g\left[e(g(\mathbf{x}), y)\right]\mathrm{E}_g\left[\pi(g)\right]}{\sqrt{\operatorname{Var}_g\left[e(g(\mathbf{x}), y)\right]\operatorname{Var}_g\left[\pi(g)\right]}},\end{aligned}$$

to measure how well the individual error $e(g(\mathbf{x}), y)$ indicates $\pi(g)$. A positive correlation ρ indicates that if g has a low error on this example, it is likely to have a low out-of-sample error, too. This is formalized in Theorem 1.

Theorem 1. *If the learning model $\mathcal{G}$ is negation symmetric (i.e., $P_{\mathcal{G}}[g] = P_{\mathcal{G}}[-g]$ for any $g \in G$),*

$$\rho(\mathbf{x}, y) \propto \mathrm{E}_g\left[\pi(g) \mid g(\mathbf{x}) \neq y\right] - \mathrm{E}_g\left[\pi(g) \mid g(\mathbf{x}) = y\right], \tag{1}$$

where the proportional constant is positive and depends only on $\mathcal{G}$.

Proof. For a given example $(\mathbf{x}, y)$ and $P_{\mathcal{G}}$, let $p_i = \Pr\left[e(g(\mathbf{x}), y) = i\right]$ and $\pi_i = \mathrm{E}_g\left[\pi(g) \mid e(g(\mathbf{x}), y) = i\right]$ for $i = 0$, 1. We have $p_0 + p_1 = 1$, and

$$\begin{gathered}\mathrm{E}_g\left[e(g(\mathbf{x}), y)\pi(g)\right] = p_1\pi_1, \quad \mathrm{E}_g\left[\pi(g)\right] = p_0\pi_0 + p_1\pi_1, \\ \mathrm{E}_g\left[e(g(\mathbf{x}), y)\right] = p_1, \quad \operatorname{Var}_g\left[e(g(\mathbf{x}), y)\right] = p_0 p_1.\end{gathered}$$

Hence with the definition of $\rho(\mathbf{x}, y)$,

$$\rho(\mathbf{x}, y) = \frac{p_1\pi_1 - p_1\left(p_0\pi_0 + p_1\pi_1\right)}{\sqrt{\operatorname{Var}_g\left[e(g(\mathbf{x}), y)\right]\operatorname{Var}_g\left[\pi(g)\right]}} = \left(\pi_1 - \pi_0\right)\sqrt{\frac{p_0 p_1}{\operatorname{Var}_g\left[\pi(g)\right]}}.$$

When $\mathcal{G}$ is negation symmetric, it is trivial that $p_0 = p_1 = \frac{1}{2}$ for any $(\mathbf{x}, y)$. So the proportional ratio is a constant. □

The conditional expectation π_1 (π_0) is the expected out-of-sample error of hypotheses that predict $(\mathbf{x}, y)$ wrongly (correctly). In the learning process, we can select hypotheses that agree on $(\mathbf{x}, y)$ or those that do not. The difference between the two conditional expectations is thus the relative change in the average out-of-sample error, and is called the *selection cost*. If there were only one example to learn, a positive selection cost would imply that we should choose a hypothesis that agrees with it. When a dataset is concerned, the selection cost indicates, at least qualitatively, how desirable it is to classify the example correctly. Since the correlation ρ is just a scaled version of the selection cost, we will use the name selection cost for both quantities in the following text.

In practice, the selection cost of an example $(\mathbf{x}_i, y_i)$ is inaccessible because $\pi(g)$ cannot be computed. However, we may estimate $\pi(g)$ by the leave-one-out error[1] $\nu^{(i)}(g) = \nu\left(g, \mathcal{D} \setminus \{(\mathbf{x}_i, y_i)\}\right)$. The selection cost can then be estimated, by random sampling over the learning model, as the correlation coefficient between $e(g(\mathbf{x}_i), y_i)$ and $\nu^{(i)}(g)$. This works well even in the presence of noise [4].

Note that for Theorem 1 to be meaningful, the actual learning model has to be used to estimate the selection cost. Under suitable choices of model complexity, however, relaxing this requirement often does not affect the performance in experiments. In this paper, we shall use neural networks as our underlying model when computing the selection cost.

We categorize an example as typical if its selection cost is greater than a threshold t_c, noisy if the cost is less than a threshold t_n, and critical if the cost lies between the two thresholds. Since a negative selection cost implies that it is better to misclassify the example, zero is an ideal threshold to separate noisy and noiseless examples. We choose thresholds around zero, $t_c = 0.15$ and $t_n = -0.1$, to accommodate estimation errors. Better categorization may be obtained by further estimating the optimal thresholds; but for the illustration purpose of this paper, we use ad hoc thresholding for all our methods.

3.2 SVM Confidence Margin

The support vector machine (SVM) [2] is a learning algorithm that finds a large-confidence hyperplane classifier in the feature space. Such classifier is usually obtained by solving the Lagrange dual problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) - \sum_{i=1}^{N} \alpha_i \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C. \end{aligned}$$

Here the kernel $K\left(\mathbf{x}, \mathbf{x}'\right)$ is an inner product of the transformed $\mathbf{x}$ and $\mathbf{x}'$ in the feature space, and C is the regularization parameter. SVM predicts the label of $\mathbf{x}$ as the sign of $\tilde{g}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K\left(\mathbf{x}_i, \mathbf{x}\right) + b$. We call $y_i \tilde{g}(\mathbf{x}_i)$ the *confidence margin* of the example $(\mathbf{x}_i, y_i)$. This concept is related to the intrinsic margin, but comes specifically from the view of a learning algorithm.

---

[1] To avoid a positive bias in estimating ρ, which can be easily verified from a formula similar to (1), we do not use the in-sample error ν.

The Lagrange multiplier α_i and the confidence margin $y_i\tilde{g}(\mathbf{x}_i)$ are also closely related:

– When $\alpha_i = 0$, we have $y_i\tilde{g}(\mathbf{x}_i) \geq 1$. The example is typical because the confidence margin is large.
– When $\alpha_i > 0$, we have $y_i\tilde{g}(\mathbf{x}_i) \leq 1$. The example is a support vector and is informative for evaluating $\tilde{g}(\mathbf{x})$.

Guyon et al. [3] used the relative magnitude of α_i as a criterion for automated identification and elimination of noisy examples with $C = \infty$. Somehow they found that the criterion failed to distinguish critical examples from noisy ones cleanly, and hence they proposed human-based post-processing to manually eliminate the noisy ones. The situation becomes even more confusing when C is finite, simply because we cannot tell whether examples with $\alpha_i = C$ are critical or noisy without other means.

In this paper, we propose to use the confidence margin $y_i\tilde{g}(\mathbf{x}_i)$ as the criterion for categorization, which works well with a suitable choice of finite C. The ideal thresholds, according to the relationship between the confidence margin and α_i, would be $t_n = 0$ and $t_c = 1$. For robustness, we use slightly different ad hoc values $t_n = 0.05$ and $t_c = 0.95$. We apply the popular Gaussian kernel with grid-based parameter search [6], and train an SVM with the best parameter to compute the confidence margin.

3.3 AdaBoost Data Weight

AdaBoost [7] is an algorithm to improve the accuracy of any base learner by iteratively generating a linear ensemble of base hypotheses. During its iterations, some examples are more likely to be misclassified than others, and are thus "hard" to learn [5]. AdaBoost maintains a set of weights for the training examples and gradually focuses on hard examples by giving them higher weights. At iteration t, the ensemble $\tilde{g}_t(\mathbf{x}) = \sum_{s=1}^{t} \alpha_s h_s(\mathbf{x})$ is constructed, where h_s is a base hypothesis and α_s is the coefficient for h_s. The data weight $w_i^{(t)}$, proportional to $e^{-y_i\tilde{g}_t(\mathbf{x}_i)}$, is thus tightly related to the ensemble confidence margin $y_i\tilde{g}_t(\mathbf{x}_i)$, and shows how hard it is to get an example correct at iteration t [7]. For instance, noisy examples tend to get misclassified a lot by base hypotheses and would have very large weights for most of the iterations; Typical examples, on the contrary, are almost always classified correctly and would have small weights. Thus, the average weight over different iterations can be used for data categorization.

Note that examples with smaller average weights are usually more reliable. For consistency with the other two methods, we actually use the negative of the average weight to approximate the intrinsic margin. For a set of size N, the initial weight is $1/N$. Thus, we use $t_c = -1.05/N$ and $t_n = -2.1/N$ to categorize examples with average weights slightly above the initial weight as critical, and examples with even higher average weights as noisy. We observe that these ad hoc thresholds work well under a common AdaBoost setting: 1000 iterations with the decision stump as the base learner.

Merler et al. [5] used the concept of *misclassification ratio,* the fraction of times an example is misclassified when it is not in the training set, to detect hard examples. Since they used resampling instead of reweighting in AdaBoost, in each iteration, the training set is sampled based on the data weight, and the misclassification ratio of an example is computed only when the example is not picked for training. This method however has a drawback that hard examples tend to have large weights and so are almost always picked in the training set. Thus the misclassification ratio is computed from a very small number of cases and is not very reliable, especially for critical and noisy examples.

We compared the misclassification ratio and the average data weight, and found that the average data weight is a better indicator of the intrinsic margin, as seen from Fig. 2.

4 Experiments with Artificial Data

We first test our methods for data categorization on three artificial targets (details in Appendix A), for which the intrinsic function is known. For each target, a dataset of size 400 is randomly generated, and the outputs of 40 examples (the last 10% indices) are further flipped as injected outliers. The ability of each method to capture the intrinsic margin is examined in two steps, the two-category experiments and the three-category experiments.

4.1 Two-Category Experiments

As mentioned previously, we try to construct a measure which is monotonic in the intrinsic margin. The scatter plots of two measures used in our methods versus the intrinsic margin (Fig. 1) show the overall monotonic relationship. However, we also observe that the monotonicity is not perfectly honored locally, and the degree of inconsistency depends on the dataset and the method used.

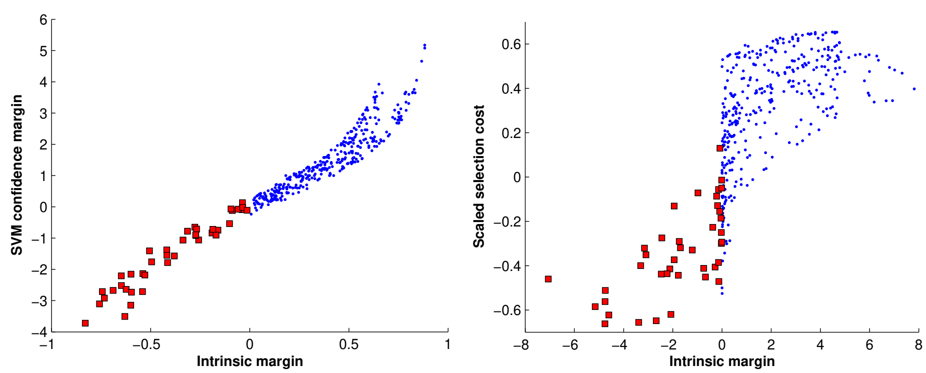

Fig. 1. Correlation between the measures and the intrinsic margin for the NNet dataset with the SVM confidence margin (**Left**) and the Sin dataset with the selection cost (**Right**). Noisy examples with negative intrinsic margins are shown as filled squares.

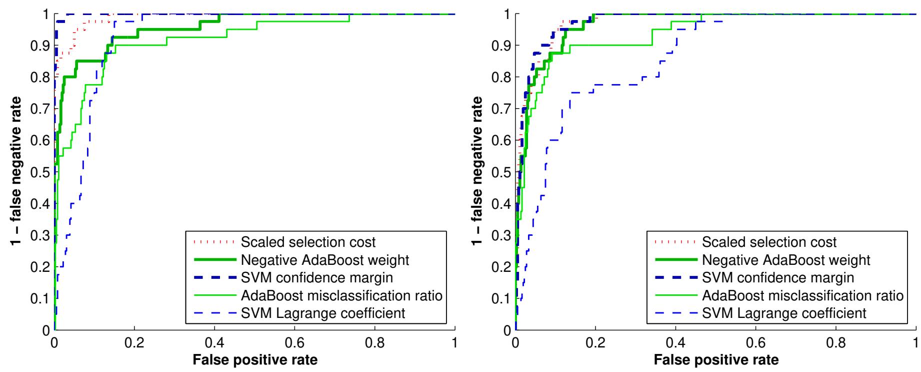

Fig. 2. ROC curves comparing the performance of all the methods on artificial datasets NNet (**Left**) and Sin (**Right**).

The scatter plots also show that our methods can reasonably group the data into noiseless (including typical and critical) and noisy categories, since the mislabeled examples are mostly cluttered in the bottom half of the plots. Figure 2 shows the receiver operating characteristic (ROC) curves for such categorization, where the false positive rate is the portion of noiseless examples being categorized as noisy. The ROC curves of two other methods, namely, the SVM Lagrange coefficient [3] and the AdaBoost misclassification ratio [5], are also plotted. These curves show that our methods, SVM confidence margin and AdaBoost data weight, surround larger area underneath and hence are much better than the related methods in literature for two-group categorization.

4.2 Three-Category Experiments

To visually study the nature of the data categorization obtained by the three methods, we design *fingerprint plots*, in which examples are positioned according to their intrinsic value $f_r(x_i)$ on the vertical axis and their index i in the dataset on the horizontal axis. Examples are also marked as typical, critical, and noisy, as assigned by the categorization method.

An ideal fingerprint plot would have the last 10% of the examples, which are mislabeled, categorized as noisy. The plot should also have a band of critical examples around the zero value. Figure 3 shows the fingerprint plot for the NNet dataset with the selection cost as the categorization method. Since the target function is in the model for estimating the selection cost, the categorization is near perfect. Figure 4 shows fingerprint plots for two other cases, where we do not have perfect categorization, and some of the critical examples are categorized as outliers and vice versa. This is partly due to the ad hoc thresholding used to categorize the examples. Similar results are obtained for the other combinations of dataset and method.

Figure 5 shows the categorization results for the two 2-D datasets visually. First, we notice that almost all the mislabeled examples are detected as noisy

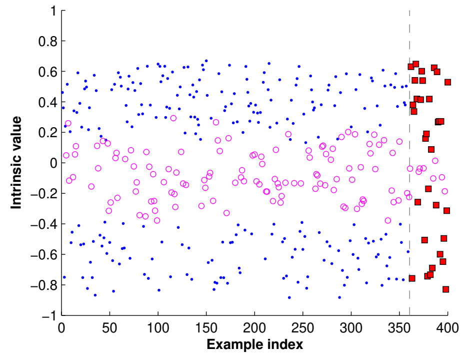

Fig. 3. Fingerprint plot of the NNet dataset with the selection cost. Critical and noisy examples are shown as empty circles and filled squares, respectively.

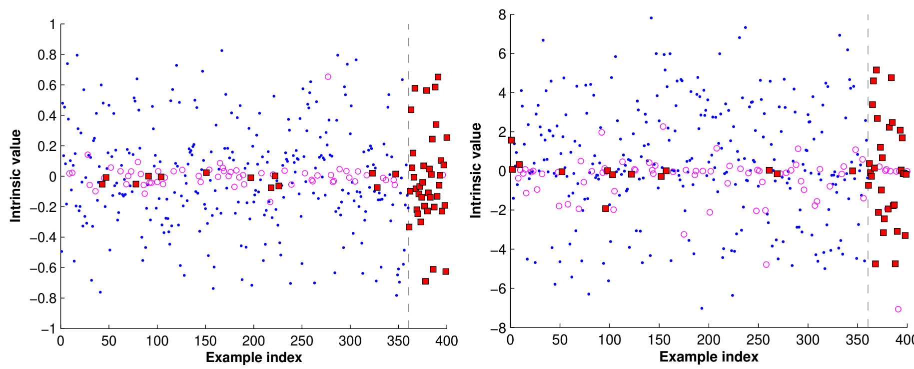

Fig. 4. Fingerprint plots of the Yin-Yang dataset with the SVM confidence margin (**Left**), and the Sin dataset with the AdaBoost data weight (**Right**).

(shown as ■), while very few of them are wrongly categorized as critical (□). Some clean examples, mostly those around the decision boundary, are also categorized as noisy (●), partly explained with the ad hoc thresholding. Secondly, we can see that most of the identified critical examples (○ or □) are around the boundary or the outliers, which is desired since examples there do provide critical information about the target.

5 Real-World Data

When the dataset has been categorized, it is possible to treat different data categories differently in learning. For example, we can remove the noisy examples and also emphasize the critical examples, and intuitively this would help

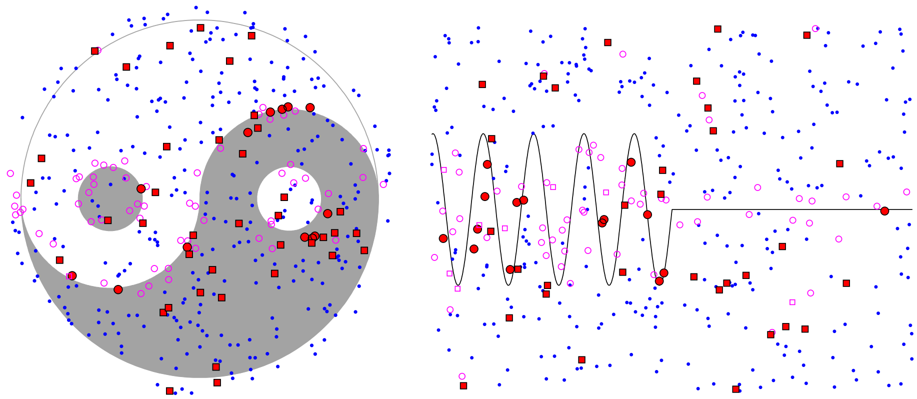

Fig. 5. 2-D categorization with the SVM confidence margin on artificial datasets Yin-Yang (**Left**) and Sin (**Right**). The 10% mislabeled examples are shown in squares and the 90% correctly labeled ones are shown in dots or circles. The three categories are shown as dots (typical), empty circles or squares (critical), and filled circles or squares (noisy).

learning.[2] To demonstrate that such a simple intuition based on our data categorization methods can be quite useful in practice, we carry out experiments on seven datasets[3] from the UCI machine learning repository [8]. Their input features are normalized to the range $[-1, 1]$. Each dataset is randomly split into training and testing parts with 60% of the data for training and rest for testing. The data categorization methods[4] are applied to the training set, and noisy examples are then removed. We further emphasize the critical examples by giving them twice the weight of the typical ones. A 500-iteration AdaBoost of decision stumps is used to learn both the full training set and the filtered one. The test error averaged over 100 runs is reported in Table 1 together with its standard error. It is observed that removing the noisy examples and emphasizing the critical ones almost always reduces the test error. The exceptions with the selection cost method should be due to a possible mismatch of complexity between the underlying model of the selection cost and the learning model used in AdaBoost.

Although details are not included here, we also observed that the test error, when we just removed the noisy examples, was not statistically different from

[2] We do not flip the noisy examples since the categorization may not be perfect. If a noiseless example is marked as noisy, flipping it brings a relatively high risk. So removing the noisy examples would be a safer choice.

[3] They are australian (Statlog: Australian Credit Approval), breast (Wisconsin Breast Cancer), cleveland (Heart Disease), german (Statlog: German Credit), heart (Statlog: Heart Disease), pima (Pima Indians Diabetes), and votes84 (Congressional Voting Records), with incomplete records removed.

[4] Note that the feed-forward neural networks for estimating the selection cost have one hidden layer of 15 neurons.

Table 1. Test error (%) of AdaBoost with 500 iterations

dataset	full dataset	selection cost	SVM margin	AdaBoost weight
australian	16.65 ± 0.19	15.23 ± 0.20	14.83 ± 0.18	13.92 ± 0.16
breast	4.70 ± 0.11	6.44 ± 0.13	3.40 ± 0.10	3.32 ± 0.10
cleveland	21.64 ± 0.31	18.24 ± 0.30	18.91 ± 0.29	18.56 ± 0.30
german	26.11 ± 0.20	30.12 ± 0.15	24.59 ± 0.20	24.68 ± 0.22
heart	21.93 ± 0.43	17.33 ± 0.34	17.59 ± 0.32	18.52 ± 0.37
pima	26.14 ± 0.20	35.16 ± 0.20	24.02 ± 0.19	25.15 ± 0.20
votes84	5.20 ± 0.14	6.45 ± 0.17	5.03 ± 0.13	4.91 ± 0.13

the case when we also emphasized the critical examples. However, we found that removing the critical examples almost always increased the test error, and removing as much as 50% of the typical examples did not affect the test error by much. This clearly shows the distinction between the three categories.

6 Conclusion

We proposed the concept of grouping data into typical, critical, and noisy categories according to the intrinsic margin, and presented three methods to automatically carry out the categorization. The three methods, rooted from different parts of learning theory, are quite different in the models they use and the way they approximate the intrinsic margin. However, they still gave similar categorization results on three artificial datasets, which established that the concept is independent of the methods. The categorization results can be used in conjunction with a large variety of learning algorithms for improving the generalization. The results on the UCI datasets with AdaBoost as the learning algorithm demonstrated the applicability of the methods in real-world problems. In addition, the categorization can also be used to reduce the dataset size without affecting the learning performance.

Further work needs to be done to estimate the optimal thresholds from the dataset (say, using a validation set [5]), to better utilize the categorization in learning, and to extend the framework for regression problems.

A Artificial Targets

We used three artificial target functions in the paper.

3-5-1 NNet. This is a feed-forward neural network with 3 inputs, 5 neurons in the hidden layer, and 1 output neuron. All neurons use tanh (sigmoid) as the transfer function. The weights and thresholds are randomly picked with a Gaussian distribution $\mathcal{N}(0, 0.7^2)$. The continuous output from the output neuron is used as the intrinsic value f_r.

Yin-Yang. A round plate centered at $(0,0)$ in $\mathbb{R}^2$ is partitioned into two classes (see Fig. 5). The "Yang" (white) class includes all points (x_1, x_2) that satisfy

$$(d_+ \le r) \vee \left(r < d_- \le \tfrac{R}{2}\right) \vee \left(x_2 > 0 \wedge d_+ > \tfrac{R}{2}\right),$$

where the radius of the plate is $R = 1$, the radius of two small circles is $r = 0.18$, $d_+ = \sqrt{(x_1 - \frac{R}{2})^2 + x_2^2}$, and $d_- = \sqrt{(x_1 + \frac{R}{2})^2 + x_2^2}$. Points out of the plate belong to the Yang class if its $x_2 > 0$. For each example, we use its Euclidean distance to the nearest boundary as its intrinsic margin.

Sin. The Sin target in [5] is also used in this paper (see Fig. 5). It partitions $[-10, 10] \times [-5, 5]$ into two class regions, and the boundary is

$$x_2 = \begin{cases} 2 \sin 3x_1, & \text{if } x_1 < 0; \\ 0, & \text{if } x_1 \ge 0. \end{cases}$$

As in the Yin-Yang target, the distance to the nearest boundary is used as the intrinsic margin.

Acknowledgment

We thank Anelia Angelova, Marcelo Medeiros, Carlos Pedreira, David Soloveichik and the anonymous reviewers for helpful discussions. This work was mainly done in 2003 and was supported by the Caltech Center for Neuromorphic Systems Engineering under the US NSF Cooperative Agreement EEC-9402726.

References

1. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. Artificial Intelligence Review **22** (2004) 85–126
2. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag, Berlin (1995)
3. Guyon, I., Matić, N., Vapnik, V.: Discovering informative patterns and data cleaning. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press / MIT Press, Cambridge, MA (1996) 181–203
4. Nicholson, A.: Generalization Error Estimates and Training Data Valuation. PhD thesis, California Institute of Technology (2002)
5. Merler, S., Caprile, B., Furlanello, C.: Bias-variance control via hard points shaving. International Journal of Pattern Recognition and Artificial Intelligence **18** (2004) 891–903
6. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Technical report, National Taiwan University (2003)
7. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference. (1996) 148–156
8. Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998) Downloadable at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.