Learning from Categorical and Numerical Imbalanced Data

By Hao Canh Nguyen

A thesis submitted to
School of Knowledge Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Knowledge Science
Graduate Program in Knowledge Science

Written under the direction of
Professor Tu Bao Ho

March, 2006

Learning from Categorical and Numerical Imbalanced Data

By Hao Canh Nguyen (450026)

A thesis submitted to
School of Knowledge Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Knowledge Science
Graduate Program in Knowledge Science

Written under the direction of
Professor Tu Bao Ho

and approved by
Professor Tu Bao Ho
Professor Kozo Sugiyama
Associate Professor Kenji Satou
Associate Professor Tsutomu Fujinami

February, 2006 (Submitted)

Abstract

Imbalanced data learning has recently begun to receive considerable attention from the research and industrial communities. Imbalanced data is problematic as traditional machine learning methods fail to achieve satisfactory results due to the skewed class distribution. Solutions to the problem generally use traditional machine learners to make a bias decision in favor of the smaller class. To make a bias decision, one need to have a good assumption of some kind of data distribution. The thesis proposes two methods to learn imbalanced data problem, one is a rule learner for categorical data using local data distributions and another is a family of sampling algorithms for numerical data using manifold modeling.

For categorical data, we deal with imbalanced data problem using example weighting to make a bias decision. Higher weights are assigned to small class examples to avoid being overshadowed by the large class ones. In this work, we introduce a scheme to weight examples of small class based solely on local data distributions. A rule learning algorithm is constructed taking the weighting scheme into account. The approach proves favorable performance to other rule learning systems. We conclude that local data distributions contain information that would be useful for the imbalanced data problem.

For numerical data, we explicitly model the distribution of small class data to make a bias decision. We utilize the flexibility of manifold modeling for the small class data distribution. Based on recent advances in manifold learning algorithms, we design basic sampling strategies to account for skewed class distribution by generating synthetic small class data. We combine these strategies to create a family of three sampling algorithms. Experimental evaluation shows that the proposed algorithms can learn effectively imbalanced data sets. We conclude that manifold is flexible and useful enough to account for the imbalanced data problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is a large number of applications of the fields of Machine Learning, Data Analysis and the like that emerged recently and encountered various theoretical difficulties. One of such difficulties is the problem of imbalanced data. In medical data analysis, it is common to have a few patients which are cancerous while the number of healthy ones are large. In network intrusion detections and fraudulent transaction detections, most of activities are regular while only a few are suspicious. In Information Retrieval, one is only interested in a piece of information in the ocean of available data. To learn those events with small number of occurrences is challenging but of primary interest. The problem is regarded as one of the ten challenges in Data Mining as compiled from opinions of many experts in the fields [3].

Traditionally, decision makers, including human and computer programs, would use statistical and computational models to describe the concepts underlining those events. However, when those events are rare compared to non-interested events, building models for the concepts becomes difficult. The reason is that most models for the concept of rarely occurred events render statistically insignificant due to the prevalence of other events. This phenomenon poses challenges to the community to develop theoretical foundations, methods and systems tailored to this type of applications.

Most methods for learning imbalanced data modify the existing traditional classifiers to adapt to this problem. In general, they have classifiers made a bias decision in favor of the small class. Researches on this problems usually focus on how to make such a bias decision, concretely on grounds and techniques to make such a decision. Various works have been proposed, but none of them becomes a de facto standard. Most of them work in isolation, based on different grounds and use different techniques but none of them proves successful in a wide range of domains.

Motivated by the challenges, the objective of this work is to derive machine learning approaches that deal with the imbalanced data problem in some concrete settings, trying to overcome drawbacks faced by previous works. Specifically, the scope of this work is to enhance classification methods in the application where the imbalanced data problem occurs. We design *methods* to make classifiers learn better the imbalanced data.

The contributions of this thesis are two works for two types of data representation. In these works, the proposed methods for classifying imbalanced data are meant to be general purposed, not bound to any specific application domains.

1. For categorical data, we introduce a rule learning algorithm of IDL for imbalanced data . The algorithm induces rules from data and is capable of weighting examples of the small class in order to account for the imbalanced data effect. The key point of this algorithm is that it assumes the imbalanced data effect occurs locally; local data distributions are used to estimate weights small class examples. The work shows that local data distributions are useful source to account for the imbalanced data effect [4].

2. For numerical data, we propose to use the notion of manifold learning [5] to model small class data distribution. The reason of using manifold modeling is that it is flexible, does not make any strict assumption but yet still useful enough. Based on the data distribution model, we derive a family of three algorithms that perturb training data by inflating the data distribution. This proves the utility of manifold notion in the imbalanced data problem.

The organization of this thesis is as follows. First we review some core background of the problem in Chapter 2. We then introduce our IDL algorithm to induce rule from imbalanced data in Chapter 3. Chapter 4 describes the sampling algorithms for numerical data. We sum up the thesis in Chapter 5.

Chapter 2

Background

In this section, we review some background knowledge of the main theme and provide insight into the problem we try to solve in a top down way. We first present the field of machine learning with two flavors of data and hypothesis representation. That is rule learning, which learns a set of rules. Its advantage is the expressiveness and human readability of the learnt concept. The second is Support Vector Machines, among the best off-the-shelf classification methods. Then we review some typical works dealing with imbalanced data problem, to highlight the problem's pervasiveness and the scatter of solutions.

2.1 Machine Learning

Ever since the computers were invented, people have wondered whether they may be able to learn. If we can program them to learn, the impact might be dramatic. One can imagine the effect if a computer can scan through patients' records and learn to be the best doctors. The houses could optimize energy usage and safety. Vehicles can autonomously drive us around. Softwares can help an human expert with rich knowledge base and intelligent assistance. Robot can do all the jobs that human can do. We can communicate with devices in natural languages. The perspective of computers can learn would be extremely promising.

Broadly speaking, machine learning is the study of computer programs that can learn though experiences. To be more precise, it is defined in [6] as follows.

Definition: *A computer program is said to learn from experiences E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experiences E.*

Table 2.1: Play tennis example

Day	Outlook	Temperature	Humidity	Windy	PlayTennis
1	sunny	hot	high	weak	No
2	sunny	hot	high	strong	No
3	overcast	hot	high	weak	Yes
4	rain	mild	high	weak	Yes
5	rain	cool	normal	weak	Yes
6	rain	cool	normal	strong	No
7	overcast	cool	normal	strong	Yes
8	sunny	mild	high	weak	No
9	sunny	cool	normal	weak	Yes
10	rain	mild	normal	weak	Yes
11	sunny	mild	normal	strong	Yes
12	overcast	mild	high	strong	Yes
13	overcast	hot	normal	weak	Yes
14	rain	mild	high	strong	No

The scope of machine learning is vast, but here we limit to some concrete settings. They are rule learning and the Support Vector Machines. We narrow ourselves to a specific task. We will work with the supervised classification problem. The type of experiences is in vector form (including categorical vectors). Performance measure is usually accuracy or F measure in the following chapters.

2.1.1 Rule Learning

When we are interested in the knowledge learnt by computers to be expressive and human readable, rule learning is a favorite choice. In this part, we review the algorithms to learn a set of if-then rules from categorical data.

Training data, which is the experience for computers to learn is in the form of categorical vectors. We take an example as in Table 2.1.

The first row in the table is the set of attributes of the data. The last attribute is the class attribute while the rest are descriptive attributes. Each of the following rows represents a training example. The computers are expected to learn from this data set to create a set of rules. The target is that the values of all descriptive attributes, the computers should be able to tell the value of the class attribute. In this case, we simplify the concept by introducing only learning propositional conjunctive rules. The rule set may look like follows.

- IF Outlook=overcast THEN PlayTennis=yes.
- IF Outlook=sunny and Humidity=high THEN PlayTennis=No.
- IF Outlook=rainy and Rain=weak THEN PlayTennis=yes.

How can the rule set be generated? Technically, the computers (programs) need to search though a space of possible rule sets (hypothesis space). Ideally, it should examine all rule sets, give a score to each set and finally return the highest score. There are two main ways of searching: sequential covering and simultaneous covering.

Sequential covering is described as follows. The program searches to learn only one rule at a time. Usually, the rule learnt is expected to cover a large enough number of examples with large enough precision (that is the correctly predicted cases over the total coverage). After a rule is learnt, the rule is put into the rule set, and covered examples are removed from training data. The process is repeated for the remaining training examples until some conditions hold. A representative technique of sequential covering is CN2 [7].

Simultaneous covering learn the entire set of rules as part of the single search. A common search procedure is divide and conquer as in decision tree induction. The training data are split into small data sets until each small data set is homogeneous according to some measures. Representative techniques in this line are C4.5 [8] and Ripper [9].

There are various issues involving any rule learning system. First is how to evaluate the goodness of a rule to stop searching for rules or to prefer one rule to another. The goodness of a rule is usually based on its coverage and purity (percentage of a class). Popular goodness measures are precision, coverage, information gain, gain ratio etc. Inductive bias, which is defined as the assumptions made on the ideal hypothesis, is used to choose one hypothesis instead of another. In rule learning algorithms, *simple* rule sets are usually preferred. The notion of simplicity comes from Occam's razor, minimum description length principle or so on.

In practice, it is exponentially expensive to search exhaustively the whole hypothesis space. Therefore, most of rule learning systems search for the rule set in a greedy manner. It is expected that greedy searches are stuck in some local minima. As a result, rule learning does not give high performance in general, specially in high dimensional domains as the expense for its expressiveness and understandability.

2.1.2 Support Vector Machines

Recently, most of major advances in the field of machine learning come from kernel methods in which Support Vector Machines algorithm (SVMs) is the most popular one. There are extremely huge number of publications related to SVMs. We recommend serious readers to the two books, one of Schlkopf and Smola [10] and the other of Showe-Taylor and Critianini [11]. Here we only revise the fundamental ideas behind the algorithm, omitting the mathematical and logarithmical sophistication.

At the first glance, SVMs can be viewed as using *kernel methods for large margin classification*. Kernel methods is a technique that provide efficient computation together with nonlinear transformation. A kernel function is used to transform data into the so-called feature space, and represent data not in vectorial form but as pairwise comparisons. This allows linear classification algorithms to work efficiently and can be made large margin. The algorithm can be summarized as follows.

Given a data set $x_i \in R^n$ is the set of training examples, which are vectors in an Euclidean space. An example x_i corresponds to a label y_i. In case $y_i \in \{+1, -1\}$, this is a binary classification problem, and if $y_i \in R$, this becomes a regression problem. The algorithm is expected to learn a function f such that $f(x_i) = y_i$. Ideally, f is expected to predict correctly label (or value) of future examples.

We start with the simplest case of f is a linear function of the examples for classification.

$$f(x) = w^T x + b \tag{2.1}$$

In the formula, $w \in R^n$, $sign(x) = 1$ if $x \geqslant 0$ and $sign(x) = -1$ if $x < 0$. We can visualize f in *Figure* 2.1. It is said from the figure that the point A is farthest from the decision boundary (the line in the figure), so it has the largest margin. In contrast, C is closest, so it has the lowest margin.

The intuition of margin is formalized as follows: Given a training example (x_i, y_i), functional margin $\widehat{\gamma_i}$ of decision boundary of f with respect to the examples is:

$$\widehat{\gamma_i} = y_i(w^T x + b) \tag{2.2}$$

In the margin, the term class labels y_i are included as decision values as well as class labels of f are negated through the decision boundary. As $y_i \in \{+1, -1\}$, including y_i into the margin would means: positive margin for correctly classified examples and negative for the incorrect ones. Functional margin of a set of examples is defined to be the worst

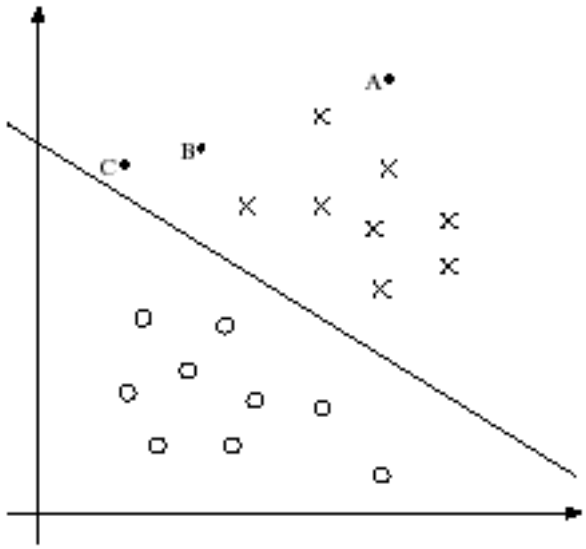


Figure 2.1: A linear classifier and margin intuition

margin with respect to all examples. Functional margin is sensitive to the scale of w and b. For example, if we use $2 * w$ and $2 * b$, we still get the same decision boundary with margins with respect to examples are all doubled. To measure the goodness of a classification, one needs to normalize this functional margins, called geometrical margin defined as follows:

$$\gamma_i = y_i \frac{w^T x + b}{\|w\|} \tag{2.3}$$

Margin is an intuition of confidence. When the margin is large, for the case of A in Figure 2.1, it is more confident that A belongs to the class, unlike C. Therefore, given a data set $S = \{(x_i, yi)\}_{i=1}^m$, it is natural to look for the classification function that has the largest margin, corresponding the fittest decision on the data. The optimal margin intuition in the computational learning theory is interpreted as the low capacity of the learnt function. One may find common philosophy of low capacity with Occam's razor of simplicity [12], Minimum Description Length principle [13], Kolmogorow complexity [14] and so on. The problem is then posed as a optimization problem as:

$$\begin{aligned} &maximize_{w,b}\ \gamma \\ &\quad s.t.\ y_i(w^t x_i + b) \geqslant \gamma, i = 1, ...m \\ &\quad \|w\| = 1. \end{aligned} \tag{2.4}$$

In this formula, we directly maximize the geometrical margin. The problem can also be

casted into:

$$\begin{aligned} minimize_{w,b}\ & \frac{1}{2}\|w\| \\ s.t.\ & y_i(w^t x_i + b) \geqslant 1, i = 1, ...m \end{aligned} \tag{2.5}$$

Now the problem becomes an optimization problem with convex quadratic objective function and linear constraints. There are many off-the-shelf QP solvers can do this task.

Using Lagrange's multiplier method, the Lagrangian is defined as:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\| - \sum_{i=1}^{m} \alpha_i [y_i(w^T x_i + b) - 1]. \tag{2.6}$$

Then, the maximal margin is now viewed as follows:

$$\begin{aligned} max_\alpha\ & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (x_i^T * x_j) \\ s.t.\ & \alpha_i \geqslant 0, i = 1, ...m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{2.7}$$

In fact, the formula 2.4 and 2.7 should give the same result. The former is called the *primal problem* and the latter is the *dual problem*. The key difference is as follows. The primal problem search for w and b. w is of the same dimension as input's. The dual problem search for α which has the dimension of number of training examples. In the space with too high dimension, the dual problem would be more efficient. However, that is not the key advantage of the dual problem, as so far we only mention the large margin aspect of SVMs. The key advantage of the dual problem is that it can be made to use kernels.

The role of kernels in the dual problem is that instead of using $x_i^T * x_j$, one can use a kernel function $k(x_i, x_i)$. If $k(x_i, x_i) = x_i^T * x_j$ then there is no differences. In this case, k is called a linear kernel. But one can make different k, which is proved to be equivalent to nonlinearly transforming data into another space, called *feature space*. The application of SVMs to practical problems is at how to choose a good kernel. Some general kernels of usual practice are:

1. Polynomial kernel: $k(x_i, x_i) = (x_i^T * x_j + 1)^d$
2. Radial basis function kernel (RBF): $k(x_i, x_i) = exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$

One may curious if these kernels work in practice. One glimpse of their utilities is as follows. If we use RBF kernel, the data set S's classes are always linearly separable in feature space.

SVMs can be refined to use *soft margin* to make it robust, accepting some training errors by introducing *slack* variables. In this case, SVMs algorithm no longer searches for the optimal margin, it has the name large margin. More details on SVMs can be looked up in the books.

One may notice the key distinction that makes SVMs algorithm perform well is that it has a global optimal solution, unlike aforementioned rule learning, which is greedy and likely to converge to local optimal of some objective functions. Another distinction of SVMs is that it can integrate capacity control into the optimization process, resulting in a regularized optimal solution.

2.2 Imbalanced data problem

Imbalanced dataset is one with very a skewed class distribution. For instance, in a binary classification problem, when one class accounts for only 2%, the other class has 98% of the examples. In such a situation, traditional classifiers would classify every example to the large class with an overall accuracy of 98%. However, the small class is not leant at all. In practice, the problem is encountered in various domains, for example, diagnoses of rare diseases [15], fraudulent transaction detection [16], oil spills in satellite images [17], rooftop detection [18], biological data [19], network intrusion detection [20], etc. In these examples, the small class is usually of primary interest; hence, an overall accuracy of 98% does not make any sense.

When class distribution is skewed, it is problematic as traditional classification methods usually make the fundamental assumption of equal class distribution. To deal with imbalanced date effect, one usually needs to make a bias decision in favor of the small class. The question remains is how to make a bias decision. Some examples of the ways to make a bias decision are: examples weighting and cost sensitive approaches, data perturbations by adding, removing or generating data and bias decision thresholds.

There are a number of works that deal with the imbalanced data problem. However, in our view, most of these works are conducted in isolation, based on different theory and use different assumptions. From that, they give different solutions. They study the imbalanced data problem in different domains. They give some insights into the problem but they fail the claim for the scope of their insights. For experimentation, in this line of research, there is not yet a standard benchmarking data resource. So far, there are no de facto standard methods to study comparatively. Most works solve some aspects of the problem, but to what extent is still a question. We review some of those works to let readers feel the scatter of this research direction.

Kubat and Matwin [17] selectively under-sampled the majority class while keeping the original population of the minority class. They have used the geometric mean as a performance measure for the classifier, which can be related to a single point on the ROC curve. The minority examples were divided into four categories: some noise overlapping the positive class decision region, borderline samples, redundant samples and safe samples. The borderline examples were detected using the Tomek links concept. Another related work proposed the SHRINK system that classifies an overlapping region of minority (positive) and majority (negative) classes as positive; it searches for the best positive region" [21].

Japkowicz [22] discussed the effect of imbalance in a dataset. She evaluated three

strategies: under-sampling, resampling and a recognition-based induction scheme. We focus on her sampling approaches. She experimented on artificial 1D data in order to easily measure and construct concept complexity. Two resampling methods were considered. Random resampling consisted of resampling the smaller class at random until it consisted of as many samples as the majority class and focused resampling" consisted of resampling only those minority examples that occurred on the boundary between the minority and majority classes. Random under-sampling was considered, which involved under-sampling the majority class samples at random until their numbers matched the number of minority class samples; focused under-sampling involved under-sampling the majority class samples lying further away. She noted that both the sampling approaches were effective, and she observed that using the sophisticated sampling techniques did not give any clear advantage in the domain considered.

Solberg and Solberg [23] considered the problem of imbalanced data sets in oil slick classification from SAR imagery. They used over-sampling and under-sampling techniques to improve the classification of oil slicks. Their training data had a distribution of 42 oil slicks and 2,471 look-alikes, giving a prior probability of 0.98 for look-alikes. This imbalance would lead the learner (without any appropriate loss functions or a methodology to modify priors) to classify almost all look-alikes correctly at the expense of misclassifying many of the oil slick samples. To overcome this imbalance problem, they over-sampled (with replacement) 100 samples from the oil slick, and they randomly sampled 100 samples from the non oil slick class to create a new dataset with equal probabilities. They learned a classifier tree on this balanced data set and achieved a 14% error rate on the oil slicks in a leave-one-out method for error estimation; on the look-alikes, they achieved an error rate of 4%.

Another approach that is of Domingos [24]. He compares the metacost approach to each of majority under-sampling and minority over-sampling. He finds that metacost improves over either, and that under-sampling is preferable to minority over-sampling. Error-based classifiers are made cost-sensitive. The probability of each class for each example is estimated, and the examples are relabeled optimally with respect to the misclassification costs. The relabeling of the examples expands the decision space as it creates new samples from which the classifier may learn.

A feed-forward neural network trained on an imbalanced dataset may not learn to discriminate enough between classes (DeRouin, Brown, Fausett, & Schneider, 1991). The authors proposed that the learning rate of the neural network be adapted to the statistics of class representation in the data. They calculated an attention factor from the

proportion of samples presented to the neural network for training. The learning rate of the network elements was adjusted based on the attention factor. They experimented on an artificially generated training set and on a real-world training set, both with multiple (more than two) classes. They compared this to the approach of replicating the minority class samples to balance the data set used for training. The classification accuracy on the minority class was improved.

Lewis and Catlett (1994) examined heterogeneous uncertainty sampling for supervised learning. This method is useful for training samples with uncertain classes. The training samples are labeled incrementally in two phases and the uncertain instances are passed on to the next phase. They modified C4.5 to include a loss ratio for determining the class values at the leaves. The class values were determined by comparison with a probability threshold of LR=(LR + 1), where LR is the loss ratio.

SMOTE is an up-sampling technique that can improve accuracy of classifier when used with down-sampling the large class [25]. It adds more examples to the small class to account for the imbalanced data effect. The synthetic examples are generated in the line segments between neighbor points randomly. In effect, this will increase the density within the small class region. When using with down-sampling, experimental evaluation shows improvements on accuracy as well as AUC measure. This work is related to our sampling technique, to be elaborated more later on.

Adaptive Conformal Mapping is used to make bias feature space transformation for SVMs [26]. This algorithm works directly on feature space by modifying the kernel function. The kernel function is modified using Conformal Mapping; the mapping that preserves local angles between elements and at the same time keeps the kernel to be valid (positive semidefinite). The transformation inflates spatial resolution around the small class examples. It is claimed that by doing so, classes are more separated and decision boundary are moved toward the large class.

As we claimed earlier, this research direction is quite scattered. However, in our two works in Chapter 3 and 4, we will try to distill the fundamental ideas behind these related works to compare and contrast to our proposed solutions.

Chapter 3

Rule Induction for Categorical Data

In this chapter, we introduce the work on rule induction from imbalanced categorical data. We first review the literature on how related works use weighting for the imbalanced data problem, then start describing our proposed algorithm from the next section.

As above, the reason that standard classifiers no longer give satisfying performance on such data sets is that they make a fundamental assumption of frequencies of classes being equally distributed. Adaptations to imbalanced data sets are usually by giving small class examples higher weights (maybe implicitly). A simple way is resampling, which duplicates small class examples or selects only a subsets of large class ones. Such approaches do not have high performance, as studied in [22], because examples are affected by class imbalance problem differently. SMOTE [25] combines synthetic example generation with downsampling, but the resampling degree is still not specified. Resampling to reflect relative weights between examples or classes remains an art.

It is believed to be more promising to weight examples differently, and various approaches have been proposed. Kubat et al. [21] insists large class examples in a mixed region are weighted zero as long as that increases performance measure. Learning on cluster basis is used [27] to weight examples accordingly. Learning decision trees (DT) [28] is made independent of class frequencies by using the Area Under the ROC Curve (AUC) as splitting criterion, which is equivalent to example weighting according to their distribution in the set of examples covered by the splitting nodes. A general way to weight examples optimally (in Bayes risk minimization sense) is using MetaCost [24], by bagging and then probability estimation. However, in highly imbalanced data sets, examples of small class are rarely to learn, making their optimal costs infinity or similarly high. Again, it is still a challenge to weight examples optimally for imbalanced data problem.

We propose a method to estimate the optimal weight of each small class example basing

solely on local data distributions. The intuition is that by looking more closely into local data distribution, we have more chance to reveal useful information about the effect of class imbalance. To this end, we first define the concept of *vicinity* that characterizes local data distribution and then determine examples' weights aiming at maximizing AUC[1] on the vicinity. The weight is integrated into a rule induction algorithm at rule pruning process.

The chapter is organized as follows. Section 1 is the foundation and formulation of our locally adaptive weighting scheme. Integration of the weighting scheme into our rule learning algorithm is described in section 2. In section 3, we show experimental evaluation of the scheme to other imbalanced data classifiers. Relation of the work to others is analyzed in section 4. Conclusions and future works are discussed in section 5.

[1]Our work on rule induction utilizes the current emerging research direction in machine learning of Receiver Operating Characteristics analysis (ROC analysis). The ROC analysis has appeared in the fields of signal detection and medical decision making for some time. It was introduced into machine learning community quite recently. In essence, ROC analysis is used to visualize performance of a classifier at all possible decision thresholds. Area under the ROC curve (AUC), calculated using ROC analysis, is one of a popular metric for classification in cost-sensitive setting and imprecise environments. Interested readers are recommended to read the following works for recent advances: [29], [30], [31] and [28].

3.1 Locally Adaptive Weighting Scheme

We approach the imbalanced data problem by giving a weight adaptively for each small class example, while keeping weights of large class examples a default value (i.e. 1). The key idea to weight each small class based on its local neighborhood (hence, it is locally adaptive), which is defined as the vicinity of the rule covering it. This section will define the concept of vicinity and derive the formulation of example weighting basing on vicinity using AUC as the criterion.

Vicinity: The intuition behind vicinity is as follows. Consider two rules $R_i, i = 1, 2$ with the same coverage for every class (R_i covers n_i, p_i examples from large and small class respectively, $n_1 = n_2, p_1 = p_2$). Conventionally, the two rules are evaluated as the same goodness (e.g., precision for small class $\frac{p_i}{p_i+n_i}$). Assume that we have some way to define a surround of a rule called neighborhood. If R_1 is likely to be pruned to a better one, then in its neighborhood, there must be some examples of the same class as the predicting class of R_1. On the other hand, R_2 is surrounded with examples from other classes, hence cannot be pruned to a better one. Our idea is to evaluate the two rules differently, R_1 to be higher than R_2, reflecting their abilities to be pruned. This different evaluation is based on the ground that, there is a set of examples in each rule's neighborhood, which makes the difference in pruning ability. Vicinity is meant to be this set of examples. We define vicinity based on the concept of k-vicinity.

Definition: *The distance from a rule to an example is the minimum number of attribute value pairs in body of the rule that need to be removed in order to make the rule cover the example.*

Definition: *A k-vicinity of a rule R for a training data set D is the set of examples in D that are of less or equal to a distance of k to the rule.*

$$k\text{-}vicinity(R) = \{x \mid x \in D, Distance(R, x) \leqslant k\} \tag{3.1}$$

K-vicinity is a subset the training data set, which is potentially covered by the rule after k steps of generalization. The smaller k is, the higher influence the examples in k-vicinity may have on generalization (pruning) ability of the rule. For example, 0-vicinity is the set of examples covered by the rule, m-vicinity is the whole data set if m is number of attribute-value pairs in the rule body. Set of all k-vicinities is a nested chain of subsets of the data set, meaning: *0-vicinity* $\subseteq$ *1-vicinity* $\subseteq$ ... $\subseteq$ *m-vicinity*. We define vicinity using this chain with weights. Formally, vicinity is a function f over k-vicinities (assuming there

exist up to m-vicinities).

$$vicinity = f\{\textit{1-vicinity}, \textit{2-vicinity}, ..., \textit{m-vicinity}\} \tag{3.2}$$

Estimating vicinity is a difficult task. However, there is a way around, which is to let vicinity of a rule remain a virtual concept. We only need to calculate *ratio of class distribution* in the vicinity for example weighting in the next section.

Example Weighting and Rule Evaluation: As vicinity is expected to contain examples that influence pruning ability of a rule, we use this assumption to define the *best* rule as the one giving optimal classification within its vicinity. Out idea is to weight examples in the vicinity such that optimal classification coincides with lowest misclassification cost. Defining optimal classification on a vicinity results in a locally adaptive weighting scheme.

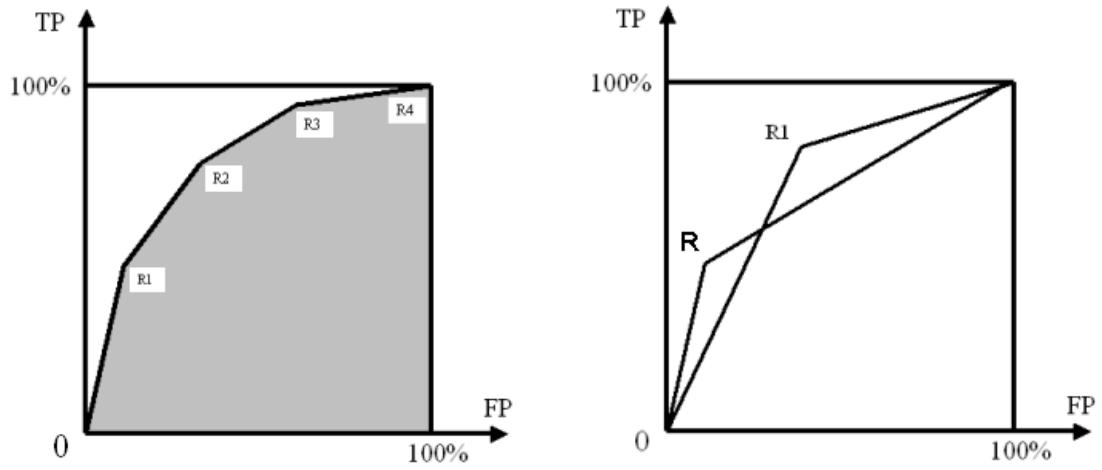
We define that optimal classification is the one that gives largest AUC. AUC is a popular metric to compare classifiers' performance [32, 29] when misclassification costs are unknown. When a classifier is a set of rules, as in Figure 3.1 (a), the ROC curve contains a set of line segments. Here, the classifier is assumed to have four rules, sorted in decreasing order of rules' precisions for a class. For simplicity, we assume that there is only one rule for small class in a vicinity. Then, the ROC curve of a classifier (by R or $R2$) in its vicinity would look like in Figure 3.1 (b). The classifier here consists of a rule (say R) and the default rule predicting large class. Suppose that R covers p small and n large class examples, and the vicinity contains P small and N large class examples. The rule evaluation metric, defined to be AUC above, is calculated [28] as:

$$AUC(R) = \frac{p}{2P} - \frac{n}{2N} + \frac{1}{2} \tag{3.3}$$

The above formula implies that, the weight of a small class example in this vicinity is $\frac{N}{P}$ when the weight of a large class example is the default value 1.

The rule evaluation metric is used to compare different rules for search bias. However, it is not natural to compare AUC in different contexts (vicinities). Hence, we propose a comparison strategy that rule R_1 is considered better than rule R if and only if it gives higher AUC in the vicinity of R. Equivalently, R_1 is considered better than R if their AUC difference (in formula 3.4) is positive.

$$AUC(R_1) - AUC(R) = \tfrac{1}{2}(\tfrac{p_1-p}{P} - \tfrac{n_1-n}{N}) = \tfrac{1}{2}[(p_1-p)\tfrac{N}{P} - (n_1-n)]\tfrac{1}{N} \tag{3.4}$$



(a) (b)

Figure 3.1: The ROC space: (a) plot of a rule learner. (b) for rule comparison in a vicinity

For the purpose of comparing rules for search bias, it is sufficient to know $\frac{N}{P}$. This is the reason we allow vicinity to remain an abstract concept, while we heuristically estimate $\frac{N}{P}$ directly. From equation 3.2, we propose to estimate class distribution ratio $\frac{N}{P}$ in the vicinity to be (m is the number of k-vicinities):

$$\frac{N}{P} = \sum_{k=1}^{m} w_k * \frac{N_k}{P_k} \tag{3.5}$$

In this formula, $\frac{N_k}{P_k}$ is class distribution ratio in k-vicinity and w_k is its associated weight, with $\sum w_k = 1$. This just smooths out class distribution ratios of different k-vicinities to estimate that of the vicinity, in similar fashion to shrinkage estimator, to make it robust. The definition of vicinity is tunable by its weighting scheme, namely the set $\{w_k\}$. If w_k is large for small k, vicinity reflects more local information. If w_k is large for large k, vicinity is more global. If we want to define vicinity to be the whole data set, then set $w_m = 1$. Having tunable set of $\{w_k\}$ is a generalization of simple cost sensitive classification. In this algorithm, by default, we fix (but still can be changed by user):

$$w_k = \frac{1}{m}, k = \overline{1, m} \tag{3.6}$$

Discussion: The key point to make this example weighting scheme suitable to imbalanced data is the use of local neighborhood. Having a myopic view around a rule, we may have a better picture of how much the imbalance may hinder classification rules.

Examples far away from boundary of classes may not participate in any vicinity, also not affecting class discrimination (this is similar to the idea behind SVMs).

Input: D is training data.
Output: Rule sets.

IDL
1. Generate a candidate rule set
2. Prune rules from high coverage to low

GenerateRuleSet
1. Generate a decision tree
2. Stop when leaf nodes contain only example from one class
3. Extract the set of leaf nodes that contain only examples of small class
4. Convert those nodes into rules and return

PruneRules
1. Sort rules according to coverage
2. From high to low coverage rule do
3. Remove best *attribute value pair*
4. Until no more AUC is gained
5. Return pruned rules

Figure 3.2: IDL algorithm

3.2 IDL: Imbalanced Data Learner

IDL is a rule induction algorithm, which follows one-sided selection strategy to learn rules only for small class. It takes into account example weighting to effectively learn small class. Example weighting is integrated into rule induction process by using above rule evaluation metric (Formula 3.3). The algorithm consists of two steps. First, it generates a set of candidate rules for small class, which are meant to be complete and of high precision. This set is generated by growing a decision tree. Then it prunes these rules by greedily removing attribute value pairs to make them robust. The overall strategy is depicted in Figure 3.2.

In the candidate rule set generation step, IDL grows a decision tree and only stops when leaf nodes contain examples from one class. IDL uses recommendation from [33] to take impurity ($2\sqrt{p(1-p)}$) gain as splitting criterion. After fully grown, the set of leaf nodes that contain only small class examples are collected, turn into a set of rules. In the second step, the collected rules for small class are sorted in decreasing order of coverage. Starting from high coverage one, each rule is pruned by removing the *best* attribute value pair

(having highest AUC difference), according to formula 3.4. It stops when removing does not give improvement on either AUC or precision of the rule (calculated without taking weights into account) falls under certain threshold. The examples covered by a rule are marked to avoid the other rules to focus on them. If marked examples are covered again, only half of their weights are retained. This makes the rules to overlap, and obtains a large improvement on recall of the classifier. The threshold represents minimum precision a rule can get, reflecting the amount of noise in data. This is generally set by user. It is estimated in IDL as follows. First, set it to 80%, then do a 10-fold stratified cross-validation on the data set to estimate its difficulty to learn. Taking the F-measure on the small class, say f (in percent), then the threshold takes the value $max(50, f - 10)$.

In the first step, IDL constructs an unpruned decision tree, which is of $O(ea)$ time complexity, where e is number of examples and a is that of attributes. In the second step, suppose it generates k rules, each has maximum n_k attribute value pairs. As each pruning operation requires a pass of database to calculate class distribution ratio in vicinity, time complexity of this step is at most $O(ekn_k)$.

3.3 Experimental Evaluation

We empirically evaluate IDL on its ability to learn the small class. We compare it to other well known approaches. First is SMOTE-NC [25], nominal categorical version of the arguably best method (SMOTE) for learning imbalanced data. SMOTE is based on C4.5, as in its original report. As SMOTE is sensitive to its degree of sampling parameters, we run SMOTE on three degrees of small class upsampling, namely N=100%, N= 300% and N=700%, corresponding to multiplying the number of examples to two, four and eight (2, 2^2 and 2^3) times. We also compare to a general classifier of C4.5, with or without cost sensitive setting. In cost sensitive setting (C.S.), the relative cost is just the ratio of class distribution of the data set. Boosting is not only a general way to improve a performance of classifiers, it is also capable of enhancing imbalanced data learners [34], then AdaBoost over C4.5 is also compared. We use these classifiers from WEKA[2]. All algorithms run with their default parameters. We use F-measure on small class as performance criterion, instead of AUC measure because our algorithm does not attempt to learn rules for the large class, using AUC would be unfair. In the formula 3.7, pr is precision and rc is recall of the classifier.

$$F - measure = 2 * \frac{pr * rc}{pr + rc} \tag{3.7}$$

We evaluate those algorithms on selected fifteen UCI data sets [3], where smallest class is chosen to be the small class; the other classes are merged to be the large class. All missing values in data are filled. As the algorithm is for categorical data, all data sets are discretized. We split data sets with a ratio of 75-25 randomly in a stratified manner. Large parts are used for training and small parts are for testing. To be fair, all training and testing data sets are common to all classifiers. The Table 3.1 shows the result of testing on the small part of data. Columns are names, percentage of small class proceeded with class index and then classifiers (SMOTE is tested with three parameters). All numbers are in percentage. The last line shows average performance of on all data sets.

The table shows that our approach outperforms general classifiers by a large margin, beats three different parameters for SMOTE, and is better than SMOTE on average. IDL improves 11.74% in term of F measure on small class comparing to a standard classifier of C4.5. For cost sensitive setting of C4.5 (C.S), it also improves 3.81%. Comparing to AdaBoost over C4.5, IDL is 2.85% higher. This means that IDL is more suitable for imbalanced data than general classifiers. Comparing IDL to an imbalanced data

[2]www.cs.waikato.ac.nz/ml/weka/

[3]http://www.ics.uci.edu/ mlearn/MLRepository.html

Name	%	C4.5	C.S.	SMOTE				A.Boost	IDL
				100	300	700	average		
annealing1	11.0	73.9	62.3	77.4	71.0	66.7	71.7	70.6	96.2
car3	3.7	66.7	84.2	77.4	80.0	80.0	79.1	80.0	76.9
flare4	8.0	0.0	36.9	30.4	36.1	38.6	35.0	32.7	29.0
glass3	13.5	93.3	73.7	93.3	82.4	82.4	86.0	80.0	85.7
hypo0	5.0	85.7	81.9	85.7	83.1	83.1	84.0	84.6	84.6
inf0	6.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
krkopt16	0.9	85.7	82.8	88.4	88.4	90.1	89.0	84.1	87.1
krkopt4	0.7	58.0	69.1	66.7	71.8	66.7	68.4	61.3	75.9
led7	8.4	59.8	59.9	63.9	61.6	50.5	58.7	50.7	62.4
letter0	3.9	91.0	90.0	92.0	91.8	89.7	91.2	96.0	92.0
satimage3	9.7	51.5	52.8	57.9	51.8	51.3	53.7	57.9	50.3
segment5	14.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
sick1	6.5	82.8	69.6	82.8	81.8	73.8	79.5	82.8	80.9
vowel5	9.1	0.0	75.2	67.8	69.4	65.5	68.2	80.0	60.0
yeast4	3.4	0.0	30.8	33.3	44.4	40.0	39.2	21.1	43.5
Average	**6.94**	**63.23**	**71.16**	**74.59**	**74.24**	**71.89**	**73.57**	**72.12**	**74.97**

Table 3.1: Comparison of Classifiers on UCI data

learner of SMOTE (SMOTE-NC version), on average, IDL is also competitive to three parameter settings. The average performance of SMOTE in the three parameter settings is 1.40% lower than IDL. It is noteworthy that there is no systematical ways to determine resampling degree for SMOTE.

It is interesting to look into improvement of IDL over C4.5 in comparison with average improvement of that of SMOTE with the three parameters in Figure 3.3. X axis of is performance improvement of SMOTE (averaging all parameters) while y axis is for IDL. The set of points shows a near linear relation. This means that improvement of IDL is proportionate with that of SMOTE, meaning that IDL is consistently similar to SMOTE.

In summary, IDL gives superior performance over standard classifiers (including an ensemble method), which are not specialized for imbalanced data. Comparing to SMOTE, IDL performance is consistently similar, and slightly higher on average of all data sets. IDL also outperforms a boosting, a cost sensitive and a general classifier. This proves that IDL is strong for learning imbalanced data.

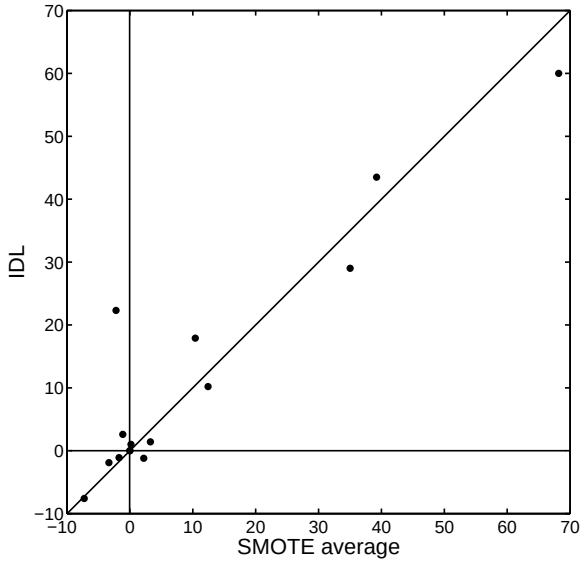


Figure 3.3: Improvement of IDL versus SMOTE

3.4 Related Works

A number of works also used data distribution in some sense to infer useful information for making decision. Ferri et al [28] propose a method to build DT, integration of cost model to make DT cost sensitive. It differentiates the process of tree growing, which is considered cost model independent, to the process of tree leaves labeling, which is inherently cost sensitive. Tree growing uses a splitting criterion of AUCSplit, that chooses the split that maximizes Area Under the ROC Curve within the considering node's context. After fully growing, decision trees are made cost sensitive in process of labeling leaves. The vicinity in our language can be viewed in this work to be the set of examples covered by the splitting nodes.

Nickerson et al [27] deal with the imbalanced problem by utilizing clustering. The guided resampling method is to equalize numbers of instances in each cluster. It is similar to our approach if we define vicinity to be cluster. Instead of guided sampling, we classify examples in a cost sensitive manner within the vicinity.

An in-depth analysis of the question why imbalanced data may prone error is conducted by Jo and Japkowicz [35]. They argue that it is not the class imbalance itself to degrade classifiers' performance. However, it is the likely cause for the problem of Small Disjuncts, which in turn causes degradation. In our approach, vicinity of examples in a small disjunct would contains a small number of examples of small class. This would likely leads to highly

skewed class distribution in this vicinity. Thus, the concept of vicinity can be used to detect small disjuncts. To some extent, the idea from our approach is implicitly aware of small disjunct problem.

Boosting uses the feedback of each round of learning to update weights accordingly [36, 34]. Initially, all weights are set equally. On each round of learning, the weights of incorrectly classified examples are increased so that, the learner is forced to focus on hard examples in the training set. Our approach estimates the weights based on their vicinity. In the end, either way also gives different weights to examples, the higher to the more difficult to be learnt. The difference is at how to measure the difficulty of learning each example, by trials in boosting or by a heuristics in ours. Our approach only produces one classifier instead of many in boosting.

3.5 Conclusion

We have proposed a method to weight examples for small class utilizing their local neighborhood information. A local is defined as a virtual concept of vicinity, while computation is based on k-vicinities. The algorithm is clearly better than general classifiers, including AdaBoost and C4.5, competitive to SMOTE while having the advantage of no resampling parameters required. From this, we can conclude that the local information around an example, like vicinity in this algorithm, is useful to weight it, in order to compensate to imbalanced data.

The clear limitation of this method is how to define weighting scheme for a vicinity. For the moment, its computational complexity is the main problem, which should be reduced for large data sets. Applying the weighting scheme to other classifiers for imbalanced data is a natural extension. Whether local data distribution can be used to improve classifiers in general is an open question.

Chapter 4

Sampling Algorithms for Numerical Data

This work is a family of sampling algorithms that add more synthetic examples to the small class in the data set. We first review the literature, unlike the last chapter, we describe in a different way. In this part, we review the assumptions behind related works, point out their limitations and proposed solutions for the imbalanced data problem.

The reason that traditional classifiers fail to learn the small class is that they tend to make the fundamental assumption of an equal class distribution. When dealing with imbalanced data, most approaches bend this assumption by introducing bias into traditional classification methods. The ground, on which bias is introduced, is usually some form of data distribution. One of the assumptions on the effect of skewed class distribution is that small class data is simply sparser. In this case, rebalancing class distribution, either by adding examples to small class (upsampling) or removing examples from the large class (downsampling), would be sufficient. Japkowicz in [22] observed that these sampling methods did not give a good performance. Simple cost-sensitive method, which gives distinct costs to classes, does not make much difference in various classification methods [33]. Similarly, SMOTE [25] generates synthetic data to add to the small class relies basically on this assumption.

Other methods make different assumptions to account for the imbalanced data problem. Nickerson et al in [27] claimed the imbalanced data effect happens within clusters in the data; rebalancing class distribution on clusters would solve the problem. The concept of Tomek link is used to downsample the large class [17]. Imbalanced data effect is attributed to small disjuncts, the ones with only a few small class examples [35]. Higher spatial resolution in feature space is given to small class [26] as small class is thought

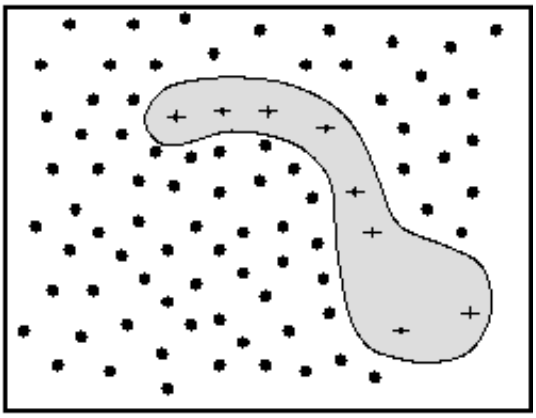


Figure 4.1: A manifold of small class

to occupy a small region. The common point among these methods is that one needs to make a good assumption of data distribution and design algorithms to make bias decisions accordingly.

In search of a good assumption for small class data, we used the notion of manifold. Manifold is a way to model nonlinear data distribution and to reveal hidden structures. The key distinction from previous approaches is that manifold is a *flexible* framework, which does not impose strict assumptions on small class data distribution. It does not require data to lie in clusters, linear subspaces or small disjuncts. *Figure* 4.1 demonstrates a manifold modeling of the small class. Having assumed the manifold structure of small class data, we deal with the imbalanced data problem by generating synthetic examples with two sampling strategies. Combining the strategies in different ways, we design three algorithms for imbalanced data. Empirical results show the significance of sampling strategies and merits of the algorithms.

In this work, we first review the fundamental idea behind manifold learning in Section 2. We then design two sampling strategies for imbalanced data by strengthening and expanding manifold structures in Section 3. In Section 4, we describe a family of three algorithms using those strategies in different ways. Then we evaluate those algorithms in comparison with other classifiers in Section 5. We then conclude the paper with some outlooks.

4.1 Manifold Learning

The driving force of manifold learning is for the problem of intrinsically low dimensional data that lies in a high dimensional space. Such problems are encountered across domains like Artificial Intelligence, Information Retrieval, Data Mining, Image Processing, Cognitive Science and so on [37]. The target of manifolds learning algorithms is to discover the low and meaningful dimensional representation of data [1]. The fundamental assumption in manifold learning algorithms is that data should lie on a manifold M, which is viewed as a Riemannian submanifold of the ambient Euclidean space and is globally isomorphic to a convex subset of a low dimensional space [38]. In this section, we review the key idea behind the manifold assumption and borrow it for the imbalanced data problem.

Recently, a new family of manifold learning algorithms has been proposed to characterize the intrinsic geometric structure of a manifold and embed it into a low (and hopefully meaningful) dimensional space. Representative algorithms are Isometric Feature Mapping (ISOMAP) [1] and Locally Linear Embedding (LLE) [2]. The frameworks of these algorithms are quite similar and can be unified as: constructing a neighborhood graph and distill information, then embedding the data into a low dimensional space preserving the information. The first step in the framework extracts information characterizing the manifold. After the abstraction process, the information is used to construct a low dimensional space representation of the original manifold. Various algorithms extract different information for computational purposes, but basically, the information is based on some neighborhood graphs. The ISOMAP algorithm, as in *Figure* 4.2, can be described as follows:

Given a data set $\{x_i\}_{i=1}^n, x_i \in R^d$, we wish to find a mapping $\phi : R^d \to R^{d'}$ such that the mapping preserves some desired information.

1. Determine which points are neighbors in the manifold based on distance between pairs of points $d(x_i, x_j)$. Two simple methods are connecting point within some fixed radius ε or connecting all k-nearest neighbors. The connections form a weighted graph G.

2. Estimate the geodesic distances $d_M(x_i, x_j)$ between all pairs of points on the manifold by the shortest path distances $d_G(x_i, x_j)$ from the graph G.

3. Use the Multidimensional Scaling method to construct an embedding in the lower dimensional space $y_i = \phi(x_i) \in R^{d'}$ by minimizing an error function, which tries to preserve geodesic distances.

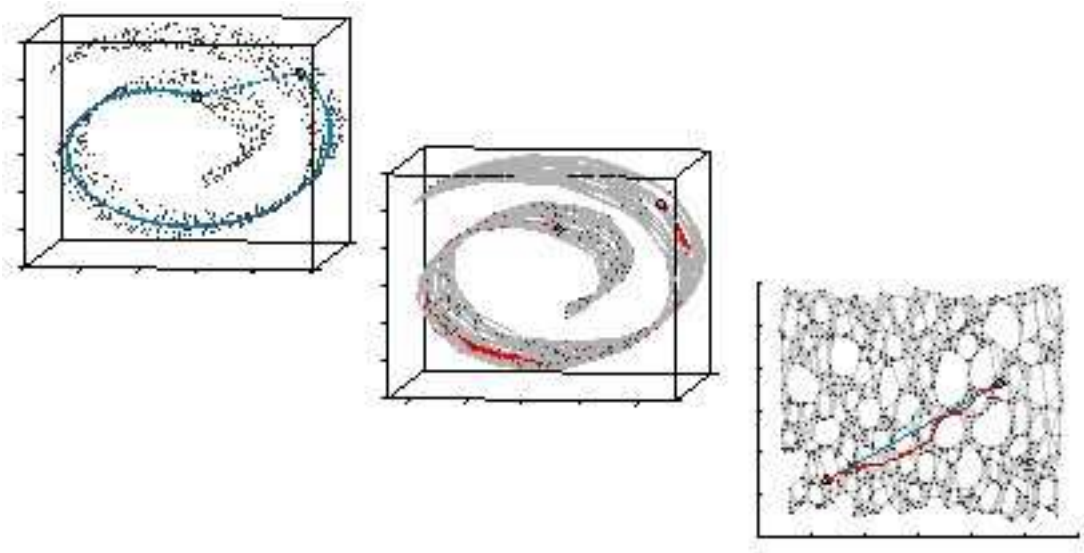


Figure 4.2: The ISOMAP algorithm [1]

LLE is slightly different that instead of preserving pairwise distances, it preserves linear coefficients that reconstruct each data point from its neighbors. The LLE algorithm can be viewed in *Figure* 4.3.

What can we learn from these manifold learning algorithms? The algorithms directly model the manifold of data points by constructing a neighborhood graph. Information on the graph, such as geodesic distances or linear coefficients to reconstruct each data point, characterizes the manifold. These algorithms are capable of modeling nonlinear manifolds. Manifold modeling is flexible in the sense that it does not make any strict assumption of data distribution like clusters, linear subspaces, Gaussian mixtures and so on. This motivates us to use manifold to model the small class in imbalanced data. The reason is that in imbalanced data, the small class is difficult to learn due to its shortage of data and may not exhibit any regularity. Therefore, the manifold assumption would be weak enough to for imbalanced data.

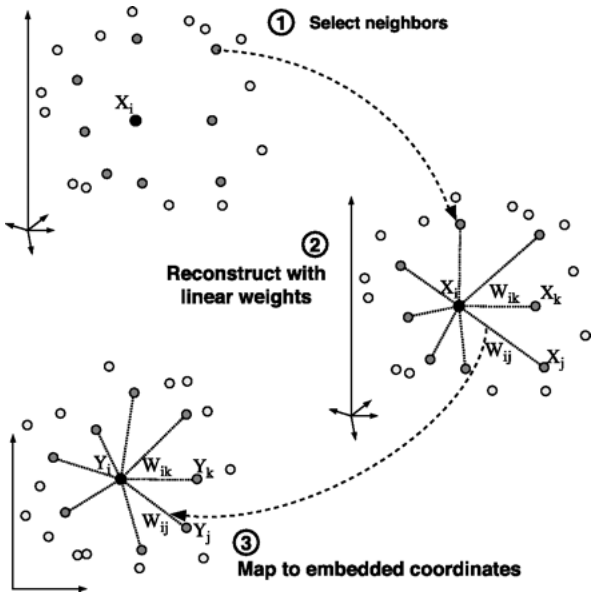


Figure 4.3: The Locally Linear Embedding algorithm [2]

4.2 Sampling Strategies

Having assumed that small class examples lie in a manifold, the first step in a manifold learning framework could be used to extract relevant information of the manifold to deal with imbalanced data. As the small class is short of training examples, it is expected that the manifold would be represented by an inefficient number of examples. Therefore, we use the manifold assumption to generate more synthetic training examples to add to the small class in order to account for the imbalanced data problem. This section describes two ways to generate synthetic examples: in-class sampling to *enhance* the manifold structure and out-class sampling to *expand* the manifold structure.

4.2.1 In-class Sampling

Our method for modeling the manifold of the small class follows the common framework of manifold learning as ISOMAP and LLE. To enhance the manifold structure, the strategy generates synthetic examples for the small class with the requirement that synthetic examples should lie in the manifold. Therefore, it is natural to choose synthetic examples as points in the line segment connecting nearest neighbors. The in-class sampling strategy is described in *Figure* 4.4.

The strategy is different from SMOTE [25] in the sense that it is fully deterministic,

Input: D^+ is set of small class examples, $x_i \in D^+$
Parameter: k is number of nearest neighbors,
n is sampling degree
Output: Synthetic examples S^+

1. *Look for x_i's k-nearest neighbors in D^+ :*
   $NB^+(x_i) \subset D^+$,$|NB^+(x_i)| = k$
2. *Choose from its k-nearest neighbors n examples with the largest distances to x_i:*
   $nNB^+(x_i) \subset NB^+(x_i)$, $|nNB^+(x_i)| = n$
3. *For each chosen neighbor, generate a synthetic examples the middle point of the line segment between it and x_i:*
   $\forall x_j \in nNB^+(x_i), x_{ij} = \frac{x_i + x_j}{2} \rightarrow x_{ij} \in S^+$

Figure 4.4: In-class Sampling Strategy

while SMOTE chooses among k-nearest neighbors randomly and generate synthetic examples randomly in the line segments. The idea of generating synthetic examples of to make data denser was also used in [39]. However, in-class sampling suffers from two properties that would be limitations for learning imbalanced data.

Property 1: *The synthetic examples generated by in-class sampling always lie inside the convex Hull of the original small class examples.*

The proof of this property is straight from the convexity of convex Hull: all line segments connecting points inside the convex Hull lie entirely within the convex Hull. In case the shortage of the small class data causes the shrinkage of the ideal convex Hull, only in-class sampling strategy would be insufficient.

Property 2: *The synthetic examples may reduce the expected (bias-corrected) variance of small class data.*

Proof: Denote the set of small class examples $D^+ = \{x_i\}_{i=1}^n$. Then mean of the set is $\overline{x} = \frac{\sum_{i=1}^n x_i}{n}$ and (bias-corrected) variance is: $var_1 = var(D^+) = \frac{\sum_{i=1}^n (x_i - \overline{x})^2}{n-1}$. Denote the set of p generated synthetic examples is $S^+ = \{x_i\}_{n+1}^{n+p}$. The new mean of all small class examples now is $\overline{x'} = \frac{\sum_{i=1}^{n+p} x_i}{n+p}$. The variance of the new small class data is $var_2 = var(D^+ \cup S^+) = \frac{\sum_{i=1}^{n+p} (x_i - \overline{x'})^2}{n+p-1}$. Denote $d = min\|x_i - x_j\|, 1 \leqslant i < j \leqslant n$ and $l = \|\overline{x} - \overline{x'}\|$.

The way in-class sampling generate synthetic examples is: $x_{n+m} = \frac{x_i + x_j}{2}$, then for any $x, (x_i - x)^2 + (x_j - x)^2 = 2(x_{n+m} - x)^2 + \frac{(x_i - x_j)^2}{2} \geqslant 2(x_{n+m} - x)^2 + \frac{d^2}{2}$. If we assume that i, j are random indices in $\{1..n\}$, then the expected value of $\sum_{m=1}^p (x_{n+m} - \overline{x})^2 \leqslant$

$\frac{p}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 - \frac{p}{2}d^2$.

Then we have:

$$\begin{aligned}
(n+p-1) * var_2 &= \sum_{i=1}^{n+p}(x_i - \overline{x'})^2 \\
&= \sum_{i=1}^{n+p}(x_i - \overline{x})^2 - (n+p)l^2 \\
&\leqslant \frac{n+p}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 - \frac{p}{2}d^2 - (n+p)l^2 \\
var_2 &\leqslant \frac{(n+p)(n-1)}{n(n+p-1)} * var_1 - \frac{(n+p)l^2 + \frac{p}{2}d^2}{n+p-1}
\end{aligned} \tag{4.1}$$

$$var_2 < (1 - \frac{p}{n(n+p-1)}) * var_1 \tag{4.2}$$

$$var_2 < var_1. \ \square$$

In-class sampling suffers from these limitations. They are unwanted for the imbalanced data problem as for the shortage of data, the learnt manifold may be shrunken down, at least in convex Hull and (bias corrected) variance senses. We need a sampling strategy to account for these limitations.

4.2.2 Out-class sampling

The previous section proves that in-class sampling does not increase the convex Hull, or the (bias-corrected) variance of small class data. However, it is reasonable to think that the shortage of data for the small class may shrink down the learned manifold. It is necessary to introduce new synthetic examples to compensate for this effect and hope it better reflects an ideal small class data distribution. The effect of shrinking a manifold would move class boundary toward the small class, therefore we wish to expand the manifold toward the boundary of classes. However, detecting the boundary of classes would be hard and algorithm specific. A way around this is to look for nearest neighbors from the other classes (the large class in binary classification problems). Therefore, we expand the manifold of small class by generating synthetic examples linking each small class example to its nearest neighbors in the large class. We call this out-class sampling as in *Figure* 4.5.

By default, we set $\epsilon = \frac{1}{3}$. This means that the generated examples are at one third of the way from the small class examples to their neighbors in the other class. The strategy

Input: $x_i \in D^+$ is a small class examples,
D^- is set of large class examples.
Parameter: k is number of nearest neighbors,
n is sampling degree,
ϵ is expansion degree.
Output: Synthetic examples S^+.

1. *Look for x_i's k-nearest neighbors in D^+*
   $NB^-(x_i) \subset D^-, |NB^-(x_i)| = k$
2. *Choose from its k-nearest neighbors n examples*
   with the smallest distances to x_i:
   $nBN^-(x_i) \subset NB^-(x_i)$, $|nNB^-(x_i)| = n$
3. *For each chosen neighbor, generate a synthetic example as a point in the line segment between it and x_i:*

$\forall x_j \in nNB^-(x_i), x_{ij} = (1-\epsilon)x_i + \epsilon x_j \rightarrow x_{ij} \in S^+$

Figure 4.5: Out-class Sampling Strategy

generates examples in the line segment between a small class example and one of its neighbors from the large class. This will push the class boundary toward the large class and expand the small class region, overcoming the two limitations of in-class sampling. In the next section, we will show how to combine these sampling strategies to learn imbalanced data.

Input: D^+ is set of small class examples,
D^- is set of large class examples.
Parameter: k is number of nearest neighbors,
inn is degree of in-class sampling,
outn is degree of out-class sampling.
Output: Synthetic examples S^+

For each $x_i \in D^+$*:*
1. In-class sampling with sampling degree inn
2. Out-class sampling with sampling degree outn

Figure 4.6: Monolithic algorithm

4.3 Manifold Sampling Algorithms

In this section, we describe three algorithms that use sampling strategies for the imbalanced data problem. The algorithms differ in the way they deploy those sampling strategies. These are: the Monolithic, Adaptive and Selective algorithms. The Monolithic algorithm simply combines in-class sampling and out-class sampling. The Adaptive algorithm uses the two sampling strategies adaptively depending on the example being considered. The Selective algorithm guesses when an example being considered needs to be sampled. In all these algorithms, the sampling degrees of both in-class and out-class are parameters to be set by users.

4.3.1 Monolithic Algorithm

A natural way to combine the two sampling strategies is to use both of them. The Monolithic algorithm uses both in-class sampling and out-class sampling for each small class example. It is summarized in *Figure* 4.6.

For each small class example, there will be $inn + outn$ synthetic examples generated around it.

4.3.2 Adaptive Algorithm

It may not be desirable to treat every small class example the same way. Some examples may be on the boundary between classes, others may lie well inside the region of the class. In such a case, it is better to sample them in different ways. Therefore, we present an

Input: D^+ is set of small class examples,
D^- is set of large class examples.
Parameter: k is number of nearest neighbors,
n is total degree of sampling.
Output: Synthetic examples S^+.

For each $x_i \in D^+$*:*
1.Calculate average distances to nearest neighbors
$pnd(x_i) = \overline{d(x_i, x_j)}, x_j \in NB^+(x_i)$
$nnd(x_i) = \overline{d(x_i, x_j)}, x_j \in NB^-(x_i)$
$inn = round(n * \frac{nnd}{pnd+nnd})$
$outn = n - inn$
2. In-class sampling with sampling degree inn
3. Out-class sample with sampling degree outn

Figure 4.7: Adaptive algorithm

adaptive way of generating synthetic examples, called the Adaptive algorithm, based on the following criteria:

- If a small class example is near the boundary of classes, use more out-class sampling to expand the boundary of small class
- If a small class example is well inside the class region, use more in-class sampling to increase the density of the class region.

The algorithm is described in *Figure* 4.7. In the algorithm, there is only one parameter (except for the number of nearest neighbors), which is the total degree of sampling for each small class example. We use the relative ratio of average distance to the small and the other class to determine how close an examples to the class boundary. The algorithm calculates the degree of in-class and out-class sampling in an adaptive manner in step *1.*

4.3.3 Selective Algorithm

Another method that does not treat every (small class) example in the same way is to select only some of them for sampling. Some examples may lie inside the region of the other class; they might be noise and should not be used for sampling. The Selective algorithm is based on the criteria:

Input: D^+ is set of small class examples,
D^- is set of large class examples.
Parameter: k is number of nearest neighbors,
inn is degree of in-class sampling,
outn is degree of out-class sampling.
Output: Synthetic examples S^+.

For each $x_i \in D^+$*:*
1. Calculate average distances to nearest neighbors
$pnd(x_i) = \overline{d(x_i, x_j)}, x_j \in NB^+(x_i)$
$nnd(x_i) = \overline{d(x_i, x_j)}, x_j \in NB^-(x_i)$
2. If $nnd > pnd$*, continue next* x
3. In-class sampling with sampling degree inn
4. Out-class sample with sampling degree outn

Figure 4.8: Selective algorithm

- If an example is near the boundary of classes or in the region of its class, use both sampling strategies to enhance and expand the manifold structures
- If an example is well inside the other class region, do not sample

The algorithm is described in *Figure* 4.8. The algorithm has two sampling parameters, in-class and out-class sampling degrees as in the Monolithic algorithm. By default, we use a heuristics to detect the examples lying closer to the large class than the small class in step 1 and 2. The heuristics means that when an example is too close to the other class, it is not used for sampling.

4.4 Experimental Evaluation

We evaluate the ability of the proposed algorithms to learn imbalanced data , which could also be the reasonability of assuming manifold structures in various domains. The base learner in our experiments was Support Vector Machines [10], due to its high performance on a vast number of domains. However, the approaches are meant to be general, not bound to any specific classification method. We showed the ability to learn a small class with F1-measure. It is defined as:

$$F\ measure = \frac{2pr * rc}{pr + pc} \tag{4.3}$$

where pr and rc respectively are the precision and recall of the learner on the small class. We deliberately choose domains in which data potentially has manifold structures.

1. Text data: Reuters-21578[1] and 20 newsgroups[2]
2. Image data: USPS optical character recognition data[3]

Text data is a popular test bed for manifold learning methods. In these databases, we selected one class in turn to be the small class and merged all the other classes to create the large class. USPS data is also a popular dataset for learning manifold.

We used SVMs from the LIBSVM[4] package. We evaluated the effectiveness of the proposed algorithms by comparing their performance against SVMs itself and SMOTE on top of SVMs. Table 4.1, Table 4.2 and Table 4.5 show the statistics of the data set (number of the small class training examples, its percentage and number of the small class testing examples). The F measure results of each algorithm in a column, namely the plain SVMs, followed by the SMOTE, Monolithic, Adaptive and Selective algorithms.

Parameters for the algorithms were chosen as follows: For SVMs on text classification, we chose linear kernel. Regularization parameter C was selected from the highest F measure using a cross-validation, C=10 for Reuters-21578 and C=5 for 20 newsgroups. For USPS, we used Gaussian (RBF) kernel with C=10, $gamma$=0.0075. The other algorithms, i.e. SMOTE and our sampling algorithms, used the same SVMs parameters. Number of nearest neighbors k=5 for all experiments. For the sampling degrees (in-class and out-class sampling) of our approaches, as noted previously, they are free parameters; we just run with all parameters and select the highest performance. In practice, one can

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2] http://people.csail.mit.edu/jrennie/20Newsgroups/

[3] http://kernel-machines.org/

[4] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

Table 4.1: Statistics of Reuters-21578 data. SVMs with linear kernel, C=10.

CLASS	#TRAIN	%TRAIN	#TEST
ACQ	1569	28.6	696
CRUDE	253	4.6	212
EARN	2840	51.8	1083
GRAIN	41	0.7	10
INTEREST	190	3.5	81
MONEY-FX	206	3.8	87
SHIP	108	2.0	36
TRADE	251	4.6	75
TOTAL	5485	100%	2189

select the best parameters by using cross-validation and the like.

4.4.1 Experimental Results on Text Domain

For the text classification databases of Reuters-21578 and 20 newsgroups, we carried out standard preprocessing as follows. First, to simplify the experiments, we filtered out all documents belonging to more than one class (multiple-labelled). Then, all non-letter characters were filtered. Short words of less than three characters and stop words were removed. We then applied Porter's stemming[5] to the remaining words. The words that appeared in fewer than three documents were also removed. These steps left Reuters-21578 data with 4172 terms, and 20 newsgroups with 17835 terms. For Reuters-21578, in the end, we chose only eight categories, which gave us large enough number of documents for the experiment. Train-test splitting was recommended by our data sources. Finally, documents were represented using TFIDF.

For the Reuters-21578 data in Table 4.3, on average, SMOTE gives a little higher F-measure than plain SVMs (1.159% higher). However, all of the proposed algorithms give much higher results than SVMs (8.747% for Monolithic, 8.310% for Adaptive and 5.880% for Selective). Moreover, our algorithms also show a significant improvement over SMOTE (7.588%, 7.151% and 4.721%). This confirms our observation about the limitations of SMOTE and proves that out-class sampling strategy is necessary.

Results for the 20 newsgroups data are shown in Table 4.4. We can see that, in most cases, our proposed algorithms give some improvement over plain SVMs, and a slight improvement over SMOTE. On average, the improvements of those algorithms over SVMs

[5]http://www.tartarus.org/martin/PorterStemmer/

Table 4.2: Statistics of 20 newsgroup data.

CLASS	#TRAIN	%TRAIN	#TEST
ATHEISM	480	4.3	319
GRAPHIC	584	5.2	389
MS-WINDOWS	572	5.1	394
PC.HARDWARE	590	5.2	392
MAC.HARDWARE	578	5.1	385
WINDOWS.X	593	5.3	392
FORSALE	585	5.2	390
AUTOS	594	5.3	395
MOTORCYCLES	598	5.3	398
BASEBALL	597	5.3	397
HOCKEY	600	5.3	399
CRYPT	595	5.3	396
ELECTRONICS	591	5.2	393
MED	594	5.3	396
SPACE	593	5.3	394
CHRISTIAN	598	5.3	398
GUNS	545	4.8	364
MIDEAST	564	5.0	376
POLITICS	465	4.1	310
RELIGION	377	3.3	251
TOTAL	11293	100%	7528

Table 4.3: F-measure results on Reuters-21578 data. SVMs with linear kernel, C=10.

CLASS	SVMS	SMOTE	MONOLITHIC	ADAPTIVE	SELLECTIVE
ACQ	94.43	94.55	95.55	94.43	94.53
CRUDE	87.78	90.26	94.07	93.28	92.37
EARN	74.15	74.51	97.16	97.11	82.90
GRAIN	88.89	88.89	95.23	95.23	95.23
INTEREST	75.38	79.10	85.18	84.47	86.27
MONEY-FX	78.15	79.74	81.87	80.65	81.01
SHIP	70.18	70.18	82.86	82.26	81.01
TRADE	87.41	88.41	95.42	95.42	90.90
AVERAGE	82.046	83.205	90.793	90.356	87.926

are: SMOTE: 4.083%, Monolithic: 5.049%, Adaptive: 4.611% and Selective: 4.286%. One may conclude that in these data sets, it is likely that out-class sampling may not be useful, that only in-class sampling would be enough. However, using out-class sampling does not damage performance as we can see that the approaches relying on it still give higher F-measures.

4.4.2 Experimental Results on USPS

USPS data consists of normalized gray scale images of hand written digits from "0" to "9". It is a popular test bed for evaluating and benchmarking classifiers, often used in manifold learning works. The data statistics and F-measures results on USPS are summarized in *Table* 4.6. We can see that SMOTE improves 0.214% over plain SVMs, the Monolithic algorithm improves 2.90%, the Adaptive algorithm improves 0.209% and the Selective algorithm improves 0.410%. It is noteworthy that the accuracy of SVMs on USPS is very high; therefore, it is unlikely that USPS data suffers heavily from the imbalanced data problem. This explains why those improvements are quite marginal. However, one can see the difference among algorithms and on average, our proposed algorithms also show better F-measure results than SMOTE and plain SVMs.

4.4.3 Summary

In summary, we have compared our proposed algorithms with plain SVMs and a sampling algorithm of SMOTE. We used the domains in which the data potential has manifolds, as our method based on the manifold assumption. It was shown that our proposed algorithms give improvements consistently over the plain SVMs. The algorithms can

Table 4.4: F-measure results on 20 newsgroup data. SVMs with linear kernel, C=5.

CLASS	SVMS	SMOTE	MONOLITHIC	ADAPTIVE	SELECTIVE
ATHEISM	64.43	72.04	72.04	69.85	70.61
GRAPHIC	68.91	72.14	72.35	71.89	72.09
MS-WINDOWS	55.41	61.59	64.07	64.06	64.97
PC.HARDWARE	61.63	65.07	65.07	65.43	66.07
MAC.HARDWARE	68.98	72.23	72.66	72.84	72.27
WINDOWS.X	70.74	74.66	74.93	74.93	73.33
FORSALE	75.77	79.44	81.45	81.17	80.69
AUTOS	80.47	82.65	82.65	82.53	82.42
MOTORCYCLES	87.99	88.80	88.80	88.00	88.83
BASEBALL	85.47	86.59	87.86	87.60	86.84
HOCKEY	94.22	94.53	94.59	94.42	94.54
CRYPT	86.62	88.8	88.8	87.76	88.62
ELECTRONICS	56.36	65.28	65.28	64.19	65.71
MED	76.52	80.8	83.81	83.6	81.10
SPACE	83.76	86.58	86.58	86.16	86.78
CHRISTIAN	77.47	79.62	82.1	82.1	81.36
GUNS	70.83	73.48	74.73	74.27	73.93
MIDEAST	78	83.13	83.67	83.04	79.62
POLITICS	56.32	61.19	64.31	63.49	62.00
RELIGION	40.36	53.3	55.48	55.15	54.19
AVERAGE	72.013	76.096	77.062	76.624	76.299

Table 4.5: Statistics of USPS data

CLASS	#TRAIN	%TRAIN	#TEST
0	1194	16.4	359
1	1005	13.8	264
2	731	10	198
3	658	9	166
4	652	8.9	200
5	556	7.6	160
6	664	9.1	170
7	645	8.9	147
8	542	7.4	166
9	644	9.1	177
TOTAL	7291	100%	2007

Table 4.6: F-measure results on USPS data. SVMs with Gaussian kernel, C=10, $gamma$=0.0075.

CLASS	SVMS	SMOTE	MONOLITHIC	ADAPTIVE	SELECTIVE
0	97.92	97.92	97.92	97.92	98.06
1	97.31	97.71	98.47	98.29	98.29
2	92.51	93.61	93.61	93.3	93.57
3	94.77	94.77	94.77	94.77	95.44
4	92.73	92.96	92.96	93.03	93.23
5	93.33	93.42	93.42	93.33	93.67
6	97.35	97.35	97.35	97.35	97.35
7	96.17	96.17	96.17	96.17	96.17
8	92.92	92.92	92.92	92.92	93.33
9	95.21	95.53	95.53	95.23	95.21
AVERAGE	95.022	95.236	95.312	95.231	95.432

also give significantly higher results than SMOTE. In the worst case, the algorithms still comparable to SMOTE. These experiments confirm our claim for the limitations of in-class sampling and SMOTE. They also show the merit of manifold assumption. As imbalanced data is difficult due to the shortage of training examples, such an assumption is not too strict but also beneficial.

4.5 Conclusion

In this work, we used the flexible notion of manifold to represent small class data distribution. Relying on the manifold leaning methods, we developed sampling strategies, which can be interpreted as strengthening or expanding the manifold to account for imbalanced data. We designed a family of three algorithms using those sampling strategies. In evaluation on the text and image domains, our algorithms showed significant improvements in their ability to learn the small class compared with the base learner of SVMs and another sampling method of SMOTE.

It is confirmed that the notion of manifold is flexible enough for data distribution of the small class. When used appropriately, it can help to express some regularity in the data distribution. It is also confirmed that the limitations of in-class sampling and SMOTE are among the effects of imbalanced data, which is an insight into the problem. This work inherits the limitations of manifold learning methods, relying on the success of manifold learning methods to model data distribution. The current work can be refined adaptively at the point of constructing neighborhood graphs. To go beyond the current work, we think of using more generative models integrating with manifold learning algorithms.

Chapter 5

Final Discussions

Imbalanced data is a challenging problem faced in many application domains of machine learning and data mining. It is a classification problem when class distributions are skewed. We proposed to two methods to learn imbalanced data in two settings. For categorical data, we proposed the rule learning algorithm of IDL, which use an example weighting technique to learn rules for the small class. For numerical imbalanced data, we proposed a family of three algorithms that sample the data to account for the imbalanced data effect. The methods show improvements on its ability to learn the small class in comparison with sibling methods.

The two methods for the two settings are related, but not directly applicable to each other. As discussed earlier, each method has its own drawbacks and future improvements. For the future work, we try to generalize the two methods into a common framework. Another future work is to apply these methods to concrete applications.

Chapter 6

Publications

1. Ho, T.B., Nguyen, C.H., Dam, H.C. (2004). Crystal Structure Determination using Data Mining Methods, *5th International Symposium on Knowledge and Systems Sciences*, Ishikawa 10-12 November, pp. 81-86.

2. Nguyen, C.H. and Ho, T.B. (2005). An Imbalanced Data Rule Learner, *Proceedings of the* 9^{th} *European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD 2005*. Porto, Portugal, pp. 617-624.

3. Ho, T.B., Nguyen, C.H., Kawasaki, S., Le, S.Q., Takabayashi, K. (2006). Exploiting Temporal Relations in Mining Hepatitis Data, *Journal of New Generation Computing*, Ohmsha, Ltd. and Springer-Verlag (submitted).

4. Nguyen, C.H., Ho, T.B. (2006). Learning Imbalanced Data with Manifold Assumption. (submitted).

Acknowledgements

I wish to show my sincere appreciation to Japan Ministry of Education, Culture, Sports, Science and Technology for its generous financial support for my study here. Without such a support, I would not be able to complete this work now.

I am indebted to Professor Tu Bao Ho, who guided me though all those years, encouraged me to pursue various research directions and showed me in detail how to get the works done in a formal way.

I wish to thank my family back home for understanding my absence and negligence, for managing themselves everything that I should have done and for their everlasting support for me to pursue my own interest.

I would like to thank Japan Advanced Institute of Science and Technology for its friendly staffs who helped me though all paper work's burdens, for nice and flexible faculties, for various interesting activities, for a fantastic working environment and for supporting me in so many things during my memorable time here.

I am grateful to the people around me who have made my time here so enjoyable. Thank the guys for hanging with me around the area, for playing soccer and letting me yell at you. Many thanks to the drinking and sport clubs.

It would be a mistake if I failed to thank the ladies who make my life interesting. Thanks for the time. Thanks for the sweet and sour memories, they are all a part of my life.

I am sure I have forgotten to thank many of you here. Please forgive me and remember that you are always in my heart.

Bibliography

[1] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–23, Dec 2000.

[2] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–6, Dec 2000.

[3] Qiang Yang. 10 challenging problems for data mining. In *http://www.cs.ust.hk/ qyang/10probs.ppt*, 2005.

[4] Canh Hao Nguyen and Tu Bao Ho. An imbalanced rule learner. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 617–624, 2005.

[5] Christopher JC Burges. *Geometric Methods for Feature Extraction and Dimensional Reduction*. 2005.

[6] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[7] Peter Clark and Tim Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[8] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[9] William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 9–12 1995. Morgan Kaufmann.

[10] B. Schlkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[11] John Shawe-Taylor and Nello Cristianini. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[12] Vladimir N. Vapnik. *The statistical learning theory*. Wiley, 1998. VAP v 98:1 1.Ex.

[13] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. Information Theory*, 44(6):2743–2760, 1998.

[14] Ming Li and Paul Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[15] S. Tsumoto. The common medical data sets to compare and evaluate kdd methods. *Journal of Japanese Society for Artificial Intelligence*, 15(5):751–758, 2000.

[16] Tom Fawcett and Foster Provost. Combining data mining and machine learning for effective user profiling. In Simoudis, Han, and Fayyad, editors, *The Second International Conference on Knowledge Discovery and Data Mining*, pages 8–13. AAAI Press, 1996.

[17] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215, 1998.

[18] M Maloof, P. Langlay, and R. Nevatia. Generalizing over aspect and location for rooftop location. In *IEEE Workshop on Applications of Computer Vision*, 1998.

[19] S. H. Muggleton, C. H. Bryant, and A. Srinivasan. Measuring performance when positives are rare: Relative advantage versus predictive accuracy — a biological case-study. In *European Conference on Machine Learning*, pages 300–312, 2000.

[20] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava, and V. Kumar. "evaluation of outlier detection schemes for detecting network intrusions". In *Third SIAM International Conference on Data Mining*, May 2003.

[21] M. Kubat and S. Marvin. Addressing the curse of unbalanced training sets: One-sided selection. In *Fourteenth International Conference on Machine Learning ICML'97*, pages 179–186, 1997.

[22] N. Japkowicz. The class imbalance problems: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, volume 1, pages 111–117, 2000.

[23] A. Solberg and R. Solberg. A large-scale evaluation of features for automatic detection of oil spills in ers sar images. In *International Geoscience and Remote Sensing Symposium*, pages 1484–1486, 1996.

[24] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[25] Nitest V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sapling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[26] G. Wu and E. Chang. Adaptive feature-space conformal transformation for imbalanced-data learning. In *Proceedings of IEEE International Conference on Machine Learning*, pages 816–823, 2003.

[27] A. Nickerson, N. Japkowicz, and E. Milios. Using unsupervised learning to guide re-sampling in imbalanced data sets. In *Eighth International Workshop on AI and Statitsics*, pages 261–265, 2001.

[28] Cesar Ferri, Peter Flach, and Jose Hernandez-Orallo. Learning decision trees using the area under the roc curve. In Claude Sammut, editor, *Nineteenth International Conference on Machine Learning ICML'02*. Morgan Kaufmann, July 2002.

[29] J. Furnkranz and P.A. Flash. An analysis of rule evaluation metrics. In *The Twentieth International Conference on Machine Learning (ICML'03)*, pages 202–209. AAAI Press, January 2003.

[30] David J. Hand and Robert J. Till. A simple generalization of the area under the ROC curve to multiple class classification problems. *Machine Learning*, 45(2):171–186, nov 2001.

[31] P.A. Flach. The geometry of roc space: understanding machine learning metrics through roc isometrics. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, pages 194–201. AAAI Press, January 2003.

[32] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

[33] C. Elkan. The foundations of cost-sensitive learning. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 973–978, 2001.

[34] Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. Predicting rare classes: can boosting make any weak learner strong? In *Proceedings of the eighth ACM international conference on Knowledge discovery and data mining*, pages 297–306. ACM Press, 2002.

[35] Taeho Jo and N Japkowicz. Class imbalances and small disjunts. *SIGKDD Explorations*, 6(1), June 2004.

[36] Robert E. Schapire. A brief introduction to boosting. In *IJCAI*, pages 1401–1406, 1999.

[37] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[38] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, 100(10):5591–5596, 2003.

[39] T.M. Ha and H. Bunke. Off-line handwritten numeral recognition by perturbation method. *IEEE transactions PAMI*, 19(5):535–539, May 1997.