Class Confidence Weighted *k*NN Algorithms for Imbalanced Data Sets

Wei Liu* and Sanjay Chawla

School of Information Technologies, University of Sydney
{wei.liu,sanjay.chawla}@sydney.edu.au

Abstract. In this paper, a novel k-nearest neighbors (kNN) weighting strategy is proposed for handling the problem of class imbalance. When dealing with highly imbalanced data, a salient drawback of existing kNN algorithms is that the class with more frequent samples tends to dominate the neighborhood of a test instance in spite of distance measurements, which leads to suboptimal classification performance on the minority class. To solve this problem, we propose `CCW` (class confidence weights) that uses the probability of attribute values given class labels to weight prototypes in kNN. The main advantage of `CCW` is that it is able to correct the inherent bias to majority class in existing kNN algorithms on any distance measurement. Theoretical analysis and comprehensive experiments confirm our claims.

1 Introduction

A data set is "imbalanced" if its dependent variable is categorical and the number of instances in one class is different from those in the other class. Learning from imbalanced data sets has been identified as one of the 10 most challenging problems in data mining research [1].

In the literature of solving class imbalance problems, data-oriented methods use sampling techniques to over-sample instances in the minor class or under-sample those in the major class, so that the resulting data is balanced. A typical example is the SMOTE method [2] which increases the number of minor class instances by creating synthetic samples. It has been recently proposed that using different weight degrees on the synthetic samples (so-called safe-level-SMOTE [3]) produces better accuracy than SMOTE. The focus of algorithm-oriented methods has been on extensions and modifications of existing classification algorithms so that they can be more effective in dealing with imbalanced data. For example, modifications of decision tree algorithms have been proposed to improve the standard C4.5, such as HDDT [4] and CCPDT [5].

KNN algorithms have been identified as one of the top ten most influential data mining algorithms [6] for their ability of producing simple but powerful

* The first author of this paper acknowledges the financial support of the Capital Markets CRC.

classifiers. The k neighbors that are the closest to a test instances are conventionally called prototypes. In this paper we use the concepts of "prototypes" and "instances" interchangeably.

There are several advanced kNN methods proposed in the recent literature. Weinberger et al. [7] learned Mahanalobis distance matrices for kNN classification by using semidefinite programming, a method which they call large margin nearest neighbor (LMNN) classification. Experimental results of LMNN show large improvements over conventional kNN and SVM. Min et al. [8] have proposed DNet which uses a non-linear feature mapping method pre-trained with Restricted Boltzmann Machines to achieve the goal of large-margin kNN classification. Recently, a new method WDkNN was introduced in [9] which discovers optimal weights for each instance in training phase which are taken into account during test phases. This method is demonstrated superior to other kNN algorithm including LPD [10], PW [11], A-NN [12] and WDNN [13].

In this paper, the model we propose is an algorithm-oriented method and we preserve all original information/distribution of the training data sets. More specifically, **the contributions** of this paper are as follows:

1. We express the mechanism of traditional kNN algorithms as equivalent to using only local *prior probabilities* to predict instances' labels, from which perspective we illustrate why many existing kNN algorithms have undesirable performance on imbalanced data sets;
2. We propose `CCW` (class confidence weights), the confidence (likelihood) of a prototype's attributes values given its class label, which transforms *prior probabilities* of to *posterior probabilities*. We demonstrate that this transformation makes the kNN classification rule analogous to using a *likelihood ratio test* in the neighborhood;
3. We propose two methods, mixture modeling and Bayesian networks, to efficiently estimate the value of `CCW`;

The rest of the paper is structured as follows. In Section 2 we review existing kNN algorithms and explain why they are flawed in learning from imbalanced data. We define `CCW` weighting strategy and justify its effectiveness in Section 3. `CCW` is estimated in Section 4. Section 5 reports experiments and Section 6 concludes the paper.

2 Existing *k*NN Classifiers

Given labeled training data $(\mathbf{x}_i,\ y_i)$ $(i = 1,...,n)$, where $\mathbf{x}_i \in \mathbb{R}^d$ are feature vectors, d is the number of features and $y_i \in \{c_1,\ c_2\}$ are binary class labels, kNN algorithm finds a group of k prototypes from the training set that are the closest to a test instance $\mathbf{x}_t$ by a certain distance measure (e.g. Euclidean distances), and estimates the test instance's label according to the predominance of a class in this neighborhood. When there is no weighting (NW) strategy, this majority voting mechanism can be expressed as:

$$\text{NW:} \quad y'_t = \arg\max_{c \in \{c_1, c_2\}} \sum_{\mathbf{x}_i \in \phi(\mathbf{x}_t)} I(y_i = c) \tag{1}$$

where y'_t is a predicted label, $I(\cdot)$ is an indicator function that returns 1 if its condition is true and 0 otherwise, and $\phi(\mathbf{x}_t)$ denotes the set of k training instances (prototypes) closest to $\mathbf{x}_t$. When the k neighbors vary widely in their distances and closer neighbors are more reliable, the neighbors are weighted by the multiplicative-inverse (MI) or the additive-inverse (AI) of their distances:

$$\text{MI:} \quad y'_t = \underset{c\in\{c_1,c_2\}}{\arg\max} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c) \cdot \frac{1}{dist(\mathbf{x}_t, \mathbf{x}_i)} \tag{2}$$

$$\text{AI:} \quad y'_t = \underset{c\in\{c_1,c_2\}}{\arg\max} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c) \cdot (1 - \frac{dist(\mathbf{x}_t, \mathbf{x}_i)}{dist_{max}}) \tag{3}$$

where $dist(\mathbf{x}_t, \mathbf{x}_i)$ represents the distance between the test point $\mathbf{x}_t$ and a prototype $\mathbf{x}_i$, and $dist_{max}$ is the maximum possible distance between two training instances in the feature space which normalizes $\frac{dist(\mathbf{x}_t,\mathbf{x}_i)}{dist_{max}}$ to the range of [0,1].

While MI and AI solve the problem of large distance variance among k neighbors, their effects become insignificant if the neighborhood of a test point is considerably dense, and one of the class (or both classes) is over-represented by its samples – since in this scenario all of the k neighbors are close to the test point and the difference among their distances is not discriminative [9].

2.1 Handling imbalanced data

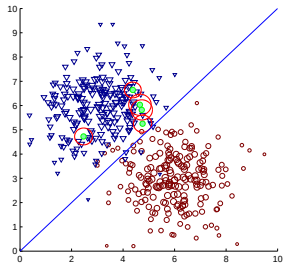
Given the definition of the conventional kNN algorithm, we now explain its drawback in dealing with imbalanced data sets. The majority voting in Eq. 1 can be rewritten as the following equivalent maximization problem:

$$\begin{aligned} & y'_t = \underset{c\in\{c_1,c_2\}}{\arg\max} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c) \\ \Rightarrow \quad & \max \{ \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c_1), \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c_2) \} \\ = \quad & \max \{ \frac{\sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c_1)}{k}, \frac{\sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c_2)}{k} \} \\ = \quad & \max \{ p_t(c_1), p_t(c_2) \} \end{aligned} \tag{4}$$

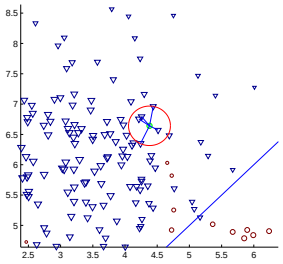
where $p_t(c_1)$ and $p_t(c_2)$ represent the proportion of class c_1 and c_2 appearing in $\phi(\mathbf{x}_t)$ – the k-neighborhood of $\mathbf{x}_t$. If we integrate this kNN classification rule into Bayes's theorem, treat $\phi(\mathbf{x}_t)$ as the *sample space* and treat $p_t(c_1)$ and $p_t(c_2)$ as *priors*[1] of two classes in this sample space, Eq. 4 intuitively illustrates that the classification mechanism of kNN is based on finding the class label that has a higher *prior* value.

This suggests that traditional kNN uses only the *prior* information to estimate class labels, which has suboptimal classification performance on the minority class when the data set is highly imbalanced. Suppose c_1 is the dominating

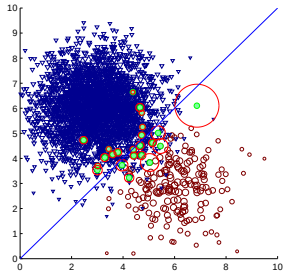
[1] We note that $p_t(c_1)$ and $p_t(c_2)$ are *conditioned* (on $\mathbf{x}_t$) in the sample space of the overall training data, but *unconditioned* in the sample space of $\phi(\mathbf{x}_t)$.



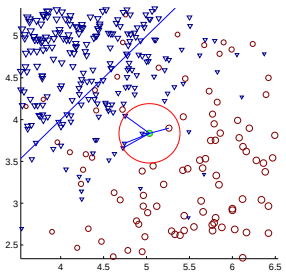
(a) Balanced data full view



(b) Balanced data regional view



(c) Imbalanced data full view



(d) Imbalanced data regional view

Fig. 1: Performance of conventional kNN ($k = 5$) on synthetic data. When data is balanced, all misclassifications of circular points are made on the upper left side of an optimal linear classification boundary; but when data is imbalanced, misclassifications of circular points appear on both sides of the boundary.

class label, it is expected that the inequality $p_t(c_1) \gg p_t(c_2)$ holds true in most regions of the feature space. Especially in the overlap regions of two class labels, kNN always tends to be biased towards c_1. Moreover, because the dominating class is likely to be over-represented in the overlap regions, "distance weighting" strategies such as WI and AI are ineffective in correcting this bias.

Figure 1 shows an example where kNN is performed by using Euclidean distance measure for $k = 5$. Samples of positive and negative classes are generated from Gaussian distributions with mean $[\mu_1^{pos}, \mu_2^{pos}] = [6, 3]$ and $[\mu_1^{neg}, \mu_2^{neg}] = [3, 6]$ respectively and a common standard deviation I (the identity matrix). The (blue) triangles are samples of the negative/majority class, the (red) unfilled circles are those of the positive/minority class, and the (green) filled circles indicate the positive samples incorrectly classified by the conventional kNN algorithm. The straight line in the middle of two clusters suggests a classification boundary built by an ideal linear classifier. Figure 1(a) and 1(c) give global

overall views of kNN classifications, while Figure 1(b) and 1(d) are their corresponding "zoom-in" subspaces that focus on a particular misclassified positive sample. Imbalanced data is sampled under the class ratio of Pos:Neg = 1:10.

As we can see from Figure 1(a) and 1(b), when data is balanced all of the misclassified positive samples are on the upper left side of the classification boundary, and are always surrounded by only negative samples. But when data is imbalanced (Figure 1(c) and 1(d)), misclassifications of positives appear on both sides of the boundary. This is because the negative class is over-represented and dominates much larger regions than the positive class. The incorrectly classified positive point in Figure 1(d) is surrounded by 4 negative and 1 positive neighbors, with a negative neighbor being the closest prototype to the test point. In this scenario, distances weighting strategies (e.g. MI and AI) cannot be helpful to correct the bias to negative class. In the next section, we introduce `CCW` and explain how it can solve such problems and correct the bias.

3 `CCW` weighted kNN

To improve the existing kNN rule, we introduce `CCW` to capture the probability (confidence) of attributes values given a class label. We define `CCW` on a training instance i as follows:

$$w_i^{\texttt{CCW}} = p(\mathbf{x}_i|y_i), \tag{5}$$

where $\mathbf{x}_i$ and y_i represent the attribute vector and the class label of instances i. Then the resulting classification rule integrated with `CCW` is:

$$\texttt{CCW}: \quad y'_t = \arg\max_{c\in\{c_1,c_2\}} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c)\cdot w_i^{\texttt{CCW}}, \tag{6}$$

and by applying it into distance weighting schemes MI and AI we obtain:

$$\texttt{CCW}^{\texttt{MI}}: \quad y'_t = \arg\max_{c\in\{c_1,c_2\}} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c)\frac{1}{dist(\mathbf{x}_t,\mathbf{x}_i)}\cdot p(\mathbf{x}_i|y_i) \tag{7}$$

$$\texttt{CCW}^{\texttt{AI}}: \quad y'_t = \arg\max_{c\in\{c_1,c_2\}} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c)(1-\frac{dist(\mathbf{x}_t,\mathbf{x}_i)}{dist_{max}})\cdot p(\mathbf{x}_i|y_i) \tag{8}$$

With the integration of `CCW`, the maximization problem in Eq. 4 becomes:

$$\begin{aligned}
& y'_t = \arg\max_{c\in\{c_1,c_2\}} \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} I(y_i = c)\cdot p(\mathbf{x}_i|y_i) \\
\Rightarrow\ & max\ \{\ \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} \frac{I(y_i=c_1)}{k} p(\mathbf{x}_i|y_i=c_1),\ \sum_{\mathbf{x}_i\in\phi(\mathbf{x}_t)} \frac{I(y_i=c_2)}{k} p(\mathbf{x}_i|y_i=c_2)\ \} \\
=\ & max\ \{\ p_t(c_1)p(\mathbf{x}_i|y_i=c_1)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)},\ p_t(c_2)p(\mathbf{x}_i|y_i=c_2)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)}\ \} \\
=\ & max\ \{\ p_t(\mathbf{x}_i,c_1)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)},\ p_t(\mathbf{x}_i,c_2)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)}\} \\
=\ & max\ \{\ p_t(c_1|\mathbf{x}_i)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)},\ p_t(c_2|\mathbf{x}_i)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)}\ \}
\end{aligned} \tag{9}$$

where $p_t(c|\mathbf{x}_i)_{\mathbf{x}_i\in\phi(\mathbf{x}_t)}$ represents the probability of $\mathbf{x}_t$ belonging to class c given the attribute values of all prototypes in $\phi(\mathbf{x}_t)$. Comparisons between Eq. 4 and

Eq. 9 demonstrate that the use of CCW changes the bases of kNN rule from using *priors* to *posteriors*: **while conventional kNN directly uses the probabilities (proportions) of class labels among the k prototypes, we use conditional probabilities of classes given the values of the k prototypes' feature vectors**. The change from *priors* to *posteriors* is easy to understand since CCW behaves just like the notion of *likelihood* in Bayes' theorem.

3.1 Justification of CCW

Since CCW is equivalent to the notion of *likelihood* in Bayes' theorem, in this subsection we demonstrate how the rationale of using CCW-based kNN rule can be interpreted by *likelihood ratio tests*.

We assume c_1 is the majority class and define the null hypothesis (H_0) as "$\mathbf{x}_t$ belonging to c_1", and the alternative hypothesis (H_1) as "$\mathbf{x}_t$ belonging to c_2". Assume among $\phi(\mathbf{x}_t)$, the first j neighbors are from c_1 and the other $k-j$ ones are from c_2. We obtain the likelihood of H_0 (L_0) and H_1 (L_1) from:

$$L_0 = \sum_{i=1}^{j} p(\mathbf{x}_i|y_i = c_1)_{\mathbf{x}_i \in \phi(\mathbf{x}_t)}, \quad L_1 = \sum_{i=j+1}^{k} p(\mathbf{x}_i|y_i = c_2)_{\mathbf{x}_i \in \phi(\mathbf{x}_t)}$$

Then the likelihood ratio test statistic can be written as:

$$\Lambda = \frac{L_0}{L_1} = \frac{\sum_{i=1}^{j} p(\mathbf{x}_i|y_i = c_1)_{\mathbf{x}_i \in \phi(\mathbf{x}_t)}}{\sum_{i=j+1}^{k} p(\mathbf{x}_i|y_i = c_2)_{\mathbf{x}_i \in \phi(\mathbf{x}_t)}} \tag{10}$$

Note that the numerator and the denominator in the fraction of Eq. 10 correspond to the two terms of the maximization problem in Eq. 9. It is essential to ensure the majority class does not have higher priority than the minority in imbalanced data, so we choose "$\Lambda = 1$" as the rejection threshold. Then the mechanism of using Eq. 9 as the kNN classification rule is equivalent to "predict $\mathbf{x}_t$ to be c_2 when $\Lambda \leq 1$" (reject H_0), and "predict $\mathbf{x}_t$ to be c_1 when $\Lambda > 1$" (do not reject H_0).

Example 1. We reuse the example in Figure 1. The size of triangles/circles is proportional to their CCW weights: the larger the size of a triangle/cirle, the greater the weight of that instance; and the smaller the lower the weight. In Figure 1(d), the misclassified positive instance has four negative-class neighbors with CCW weights 0.0245, 0.0173, 0.0171 and 0.0139, and has one positive-class neighbor of weight 0.1691. Then the total negative-class weight is 0.0728 and the total positive-class weight is 0.1691, and the CCW ratio is $\frac{0.0728}{0.1691} < 1$ which gives a label prediction to the positive (minority) class. So even though the closest prototype to the test instance comes from the wrong class which also dominates the test instance's neighborhood, a CCW weighted kNN can still correctly classify this actual positive test instances.

4 Estimations of CCW weights

In this section we briefly introduce how we employ mixture modeling and Bayesian networks to estimate CCW weights.

4.1 Mixture models

In the formulation of mixture models, the training data is assumed follow a q-component finite mixture distribution with probability density function (*pdf*):

$$p(\mathbf{x}|\theta) = \sum_{m=1}^{q} \alpha_m p(\mathbf{x}|\theta_m) \tag{11}$$

where $\mathbf{x}$ is a sample of training data whose *pdf* is demanded, α_m represents mixing probabilities, θ_m defines the mth component, and $\widehat{\theta} \equiv \{\theta_1,...,\theta_q, \alpha_1,...,\alpha_q\}$ is the complete set of parameters specifying the mixture model. Given training data Ω, the log-likelihood of a q-component mixture distribution is: $\log p(\Omega|\widehat{\theta}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i|\widehat{\theta}) = \sum_{i=1}^{n} \log \sum_{m=1}^{q} p(\mathbf{x}_i|\theta_m)$. Then the maximum likelihood (ML) estimate $\widehat{\theta}_{ML} = \arg\max_\theta \log p(\Omega|\theta)$ can be found analytically. We use the expectation-maximization (EM) algorithm to solve ML and then apply the estimated $\widehat{\theta}$ into Eq. 11 to find the *pdf* of all instances in training data set as their corresponding `CCW` weights.

Example 2. We reuse the example in Figure 1, but now we assume the underlying distribution parameters (i.e. the mean and variance matrixes) that generate the two classes of data are unknown. We apply training samples into ML estimation, solve for $\widehat{\theta}$ by EM algorithm, and then use Eq. 11 to estimate the *pdf* of training instances which are used as their `CCW` weights. The estimated weights (and their effects) of the neighbors of the originally misclassified positive sample in Figure 1(d) are shown in Example 1.

4.2 Bayesian networks

While mixture modeling deals with numerical features, Bayesian networks can be used to estimate `CCW` when feature values are categorical. The task of learning a Bayesian network is to (*i*) build a directed acyclic graph (DAG) over Ω, and (*ii*) learn a set of (conditional) probability tables $\{p(\omega|pa(\omega)), \omega \in \Omega\}$ where $pa(\omega)$ represents the set of parents of ω in the DAG. From these conditional distributions one can recover the joint probability distribution over Ω by using $p(\Omega) = \prod_{i=1}^{d+1} p(\omega_i|pa(\omega_i))$.

In brief, we learn and build the structure of the DAG by employing K2 algorithm [14] which in the worst case has an overall time complexity of $O(n^2)$, one "n" for the number of features and another "n" for the number of training instances. Then we estimate the conditional probability tables directly from training data. After obtaining the joint distributions $p(\Omega)$, the `CCW` weight of a training instance i can be easily obtained from $w_i^{\texttt{CCW}} = p(\mathbf{x}_i|y_i) \propto \frac{p(\Omega)}{p(y_i)}$ where $p(y_i)$ is the proportion of class y_i among the entire training data.

5 Experiments and Analysis

In this section, we analyze and compare the performance of `CCW`-based kNN against existing kNN algorithms, other *algorithm-oriented* state of the art ap-

Table 1: Details of imbalanced data sets and comparisons of kNN algorithms on weighting strategies for $k = 1$.

Name	#Inst	#Att	MinClass	CovVar	Area Under Precision-Recall Curve					
					NW	MI	CCW^{MI}	AI	CCW^{AI}	WDkNN
KDDCup'09[7]:										
Appetency	50000	278	1.76%	4653.2	.022(4)	.021(5)	.028(2)	.021(5)	.035(1)	.023(3)
Churn	50000	278	7.16%	3669.5	.077(3)	.069(5)	.077(2)	.069(5)	.093(1)	.074(4)
Upselling	50000	278	8.12%	3506.9	.116(6)	.124(4)	.169(2)	.124(4)	.169(1)	.166(3)
Agnostic-vs-Prior[8]:										
Ada.agnostic	4562	48	24.81%	1157.5	.441(6)	.442(4)	.520(2)	.442(4)	.609(1)	.518(3)
Ada.prior	4562	15	24.81%	1157.5	.443(4)	.433(5)	.518(3)	.433(5)	.606(1)	.552(2)
Sylva.agnostic	14395	213	6.15%	11069.1	.672(6)	.745(4)	.790(2)	.745(4)	.797(1)	.774(3)
Sylva.prior	14395	108	6.15%	11069.1	.853(6)	.906(4)	.941(2)	.906(4)	.945(1)	.907(3)
StatLib[9]:										
BrazilTourism	412	9	3.88%	350.4	.064(6)	.111(4)	.132(2)	.111(4)	.187(1)	.123(3)
Marketing	364	33	8.52%	250.5	.106(6)	.118(4)	.152(1)	.118(4)	.152(2)	.128(3)
Backache	180	33	13.89%	93.8	.196(6)	.254(4)	.318(2)	.254(4)	.319(1)	.307(3)
BioMed	209	9	35.89%	16.6	.776(6)	.831(4)	.874(2)	.831(4)	.887(1)	.872(3)
Schizo	340	15	47.94%	0.5	.562(4)	.534(5)	.578(3)	.534(5)	.599(1)	.586(2)
Text Mining [15]:										
Fbis	2463	2001	1.54%	2313.3	.082(6)	.107(4)	.119(2)	.107(4)	.117(3)	.124(1)
Re0	1504	2887	0.73%	1460.3	.423(6)	.503(5)	.561(2)	.503(4)	.563(1)	.559(3)
Re1	1657	3759	0.78%	1605.4	.360(1)	.315(5)	.346(2)	.315(5)	.346(2)	.335(4)
Tr12	313	5805	9.27%	207.7	.450(6)	.491(4)	.498(1)	.491(3)	.490(5)	.497(2)
Tr23	204	5833	5.39%	162.3	.098(6)	.122(4)	.136(1)	.122(4)	.128(3)	.134(2)
UCI [16]:										
Arrhythmia	452	263	2.88%	401.5	.083(6)	.114(4)	.145(2)	.114(4)	.136(3)	.159(1)
Balance	625	5	7.84%	444.3	.064(1)	.063(4)	.063(4)	.064(2)	.064(3)	.061(6)
Cleveland	303	14	45.54%	2.4	.714(6)	.754(4)	.831(2)	.754(4)	.846(1)	.760(3)
Cmc	1473	10	22.61%	442.1	.299(6)	.303(5)	.318(2)	.305(4)	.357(1)	.315(3)
Credit	690	16	44.49%	8.3	.746(6)	.751(4)	.846(2)	.751(4)	.867(1)	.791(3)
Ecoli	336	8	5.95%	260.7	.681(4)	.669(5)	.743(2)	.669(5)	.78(1)	.707(3)
German	1000	21	30.0%	160.0	.407(6)	.427(4)	.503(2)	.427(4)	.509(1)	.492(3)
Heart	270	14	44.44%	3.3	.696(6)	.758(4)	.818(2)	.758(4)	.826(1)	.790(3)
Hepatitis	155	20	20.65%	53.4	.397(6)	.430(4)	.555(2)	.430(4)	.569(1)	.531(3)
Hungarian	294	13	36.05%	22.8	.640(6)	.659(4)	.781(2)	.659(4)	.815(1)	.681(3)
Ionosphere	351	34	35.9%	27.9	.785(6)	.874(5)	.903(2)	.884(3)	.911(1)	.882(4)
Ipums	7019	60	0.81%	6792.8	.056(6)	.062(4)	.087(1)	.062(5)	.087(2)	.078(3)
Pima	768	9	34.9%	70.1	.505(6)	.508(4)	.587(2)	.508(4)	.618(1)	.533(3)
Primary	339	18	4.13%	285.3	.168(6)	.222(4)	.265(1)	.217(5)	.224(3)	.246(2)
Average Rank					5.18	4.18	1.93	4.03	1.53	2.84
Friedman Tests					✓ 7E-7	✓ 8E-6	**Base**	✓ 2E-5	–	✓ 4E-5
Friedman Tests					✓ 3E-6	✓ 2E-6	–	✓ 9E-6	**Base**	✓ 2E-4

proaches (i.e. WDkNN[2], LMNN[3], DNet[4], CCPDT[5] and HDDT[6]) and *data-oriented* methods (i.e. safe-level-SMOTE). We note that since WDkNN has been demonstrated (in [9]) better than LPD, PW, A-NN and WDNN, in our experiments we include only the more superior WDkNN among them. CCPDT and HDDT are pruned by Fisher's exact test (as recommended in [5]). All experiments are carried out using 5×2 folds cross-validations, and the final results are the average of the repeated runs.

[2] We implement CCW-based kNNs and WDkNN inside Weka environment [17].
[3] The code is obtained from `www.cse.wustl.edu/~kilian/Downloads/LMNN.html`.
[4] The code is obtained from `www.cs.toronto.edu/~cuty/DNetkNN_code.zip`.
[5] The code is obtained from `www.cs.usyd.edu.au/~weiliu/CCPDT_src.zip`.
[6] The code is obtained from `www.nd.edu/~dial/software/hddt.tar.gz`.

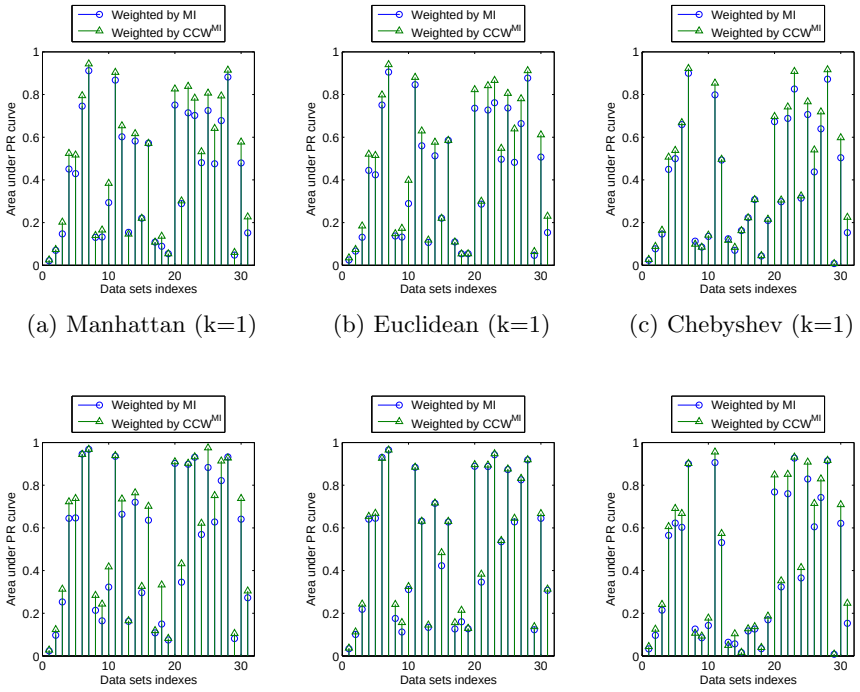
Table 2: Performance of kNN weighting strategies when $k = 11$.

Datasets	Area Under Precision-Recall Curve									
	MI	CCW^{MI}	AI	CCW^{AI}	SMOTE	WDkNN	LMNN	DNet	CCPDT	HDDT
Appetency	.033(8)	.037(4)	.036(6)	.043(1)	.040(3)	.036(5)	.035(7)	.042(2)	.024(10)	.025(9)
Churn	.101(7)	.113(2)	.101(6)	.115(1)	.108(4)	.100(8)	.107(5)	.111(3)	.092(10)	.099(9)
Upselling	.219(8)	.243(5)	.218(9)	.241(6)	.288(3)	.212(10)	.231(7)	.264(4)	.443(1)	.437(2)
Ada.agnostic	.641(9)	.654(5)	.646(8)	.652(6)	.689(3)	.636(10)	.648(7)	.670(4)	.723(1)	.691(2)
Ada.prior	.645(8)	.669(2)	.654(7)	.668(3)	.661(5)	.639(9)	.657(6)	.664(4)	.682(1)	.605(10)
Sylva.agnostic	.930(2)	.926(8)	.930(3)	.925(9)	.928(6)	.922(10)	.928(4)	.926(7)	.934(1)	.928(5)
Sylva.prior	.965(4)	.965(2)	.965(6)	.965(4)	.904(10)	.974(1)	.965(3)	.935(9)	.946(8)	.954(7)
BrazilTourism	.176(9)	.242(1)	.232(5)	.241(2)	.233(4)	.184(8)	.209(6)	.237(3)	.152(10)	.199(7)
Marketing	.112(10)	.157(2)	.113(9)	.161(1)	.124(8)	.150(3)	.134(5)	.142(4)	.130(6)	.125(7)
Backache	.311(7)	.325(3)	.307(8)	.328(2)	.317(6)	.330(1)	.318(5)	.322(4)	.227(9)	.154(10)
BioMed	.884(5)	.885(3)	.858(7)	.844(8)	.910(2)	.911(1)	.884(4)	.877(6)	.780(10)	.812(9)
Schizo	.632(6)	.632(4)	.626(7)	.617(8)	.561(10)	.663(3)	.632(5)	.589(9)	.807(2)	.846(1)
Fbis	.134(10)	.145(5)	.135(9)	.141(6)	.341(3)	.136(8)	.140(7)	.241(4)	.363(2)	.384(1)
Re0	.715(3)	.717(1)	.705(5)	.709(4)	.695(7)	.683(8)	.716(2)	.702(6)	.573(9)	.540(10)
Re1	.423(7)	.484(1)	.434(6)	.475(4)	.479(2)	.343(8)	.454(5)	.477(3)	.274(9)	.274(9)
Tr12	.628(6)	.631(4)	.624(7)	.601(8)	.585(10)	.735(3)	.629(5)	.593(9)	.946(1)	.946(1)
Tr23	.127(8)	.156(3)	.123(10)	.156(3)	.124(9)	.128(7)	.141(5)	.140(6)	.619(2)	.699(1)
Arrhythmia	.160(7)	.214(4)	.167(6)	.229(3)	.083(10)	.134(9)	.187(5)	.156(8)	.346(2)	.385(1)
Balance	.127(7)	.130(5)	.145(2)	.149(1)	.135(4)	.091(9)	.129(6)	.142(3)	.092(8)	.089(10)
Cleveland	.889(8)	.897(2)	.890(6)	.897(1)	.889(7)	.895(3)	.893(5)	.893(4)	.806(10)	.846(9)
Cmc	.346(9)	.383(2)	.357(7)	.384(1)	.358(6)	.341(10)	.365(5)	.371(4)	.356(8)	.380(3)
Credit	.888(7)	.895(2)	.887(8)	.894(3)	.891(5)	.903(1)	.891(6)	.893(4)	.871(9)	.868(10)
Ecoli	.943(3)	.948(1)	.938(5)	.941(4)	.926(7)	.920(8)	.945(2)	.933(6)	.566(10)	.584(9)
German	.535(7)	.541(2)	.533(8)	.537(4)	.536(6)	.561(1)	.538(3)	.537(5)	.493(9)	.464(10)
Heart	.873(7)	.876(4)	.873(8)	.876(5)	.878(2)	.883(1)	.875(6)	.877(3)	.828(9)	.784(10)
Hepatitis	.628(6)	.646(1)	.630(5)	.645(2)	.625(8)	.626(7)	.637(3)	.635(4)	.458(9)	.413(10)
Hungarian	.825(5)	.832(1)	.823(7)	.831(2)	.819(8)	.826(4)	.829(3)	.825(6)	.815(9)	.767(10)
Ionosphere	.919(4)	.919(2)	.916(7)	.918(5)	.916(7)	.956(1)	.919(3)	.917(6)	.894(9)	.891(10)
Ipums	.123(8)	.138(4)	.123(7)	.140(2)	.136(5)	.170(1)	.130(6)	.138(3)	.037(9)	.020(10)
Pima	.645(7)	.667(1)	.644(8)	.665(2)	.657(4)	.655(6)	.656(5)	.661(3)	.587(10)	.613(9)
Primary	.308(5)	.314(2)	.271(8)	.279(7)	.310(4)	.347(1)	.311(3)	.294(6)	.170(10)	.183(9)
Average Rank	6.5	2.78	6.59	3.71	5.59	5.18	4.68	4.78	6.68	6.9
Friedman	✓2E-7	**Base**	✓1E-6	–	0.1060	✓0.002	✓2E-7	✓0.007	✓0.019	✓0.007
Friedman	✓0.011	–	✓4E-5	**Base**	0.1060	✓0.007	✓0.007	✓0.048	✓0.019	✓0.007

We select 31 data sets from KDDCup'09[7], agnostic vs. prior competition[8], StatLib[9], text mining [15], and UCI repository [16]. For multiple-label data sets, we keep the smallest label as the positive class, and combine all the other labels as the negative class. Details of the data sets are shown in Table 1. Besides the proportion of the minor class in a data set, we also present the coefficient of variation (CovVar) [18] to measure imbalance. CovVar is defined as the ratio of the standard deviation and the mean of the class counts in data sets.

The metric of AUC-PR (area under precision-recall curve) has been reported in [19] better than AUC-ROC (area under ROC curve) on imbalanced data. A curve dominates in ROC space if and only if it dominates in PR space, and classifiers that are more superior in terms of AUC-PR are definitely more superior in terms of AUC-ROC, but not vice versa [19]. Hence we use the more informative metric of AUC-PR for classifier comparisons.

[7] http://www.kddcup-orange.com/data.php
[8] http://www.agnostic.inf.ethz.ch
[9] http://lib.stat.cmu.edu/



(a) Manhattan (k=1) (b) Euclidean (k=1) (c) Chebyshev (k=1)

(d) Manhattan (k=11) (e) Euclidean (k=11) (f) Chebyshev (k=11)

Fig. 2: Classification improvements from `CCW` on Manhattan distance (ℓ_1 norm), Euclidean distance (ℓ_2 norm) and Chebyshev distance (ℓ_∞ norm).

5.1 Comparisons among NN algorithms

In this experiment we compare `CCW` with existing kNN algorithm using Euclidean distance on $k = 1$. When $k = 1$, apparently all kNNs that use the same distance measure have exactly the same prediction on a test instances. However the effects of `CCW` weights generate different probabilities of being positive/negative for each test instance, and hence produce different AUC-PR values.

While there are various ways to compare classifiers across multiple data sets, we adopt the strategy proposed by [20] that evaluates classifiers by ranks. In Table 1 the kNN classifiers in comparison are ranked on each data set by the value of their AUC-PR, with ranking of 1 being the best. We perform Friedman tests on the sequences of ranks between different classifiers. In Friedman tests, p–values that are lower than 0.05 reject the hypothesis with 95% confidence that the ranks of classifiers in comparison are not statistically different. Numbers in parentheses of Table 1 are the ranks of classifiers on each data set, and a ✓ sign in Friedman tests suggests classifiers in comparison are significantly different.

As we can see, both $\texttt{CCW}^{\text{MI}}$ and $\texttt{CCW}^{\text{AI}}$ (the "Base" classifiers) are significantly better than existing methods of NW, MI, AI and WDkNN.

5.2 Comparisons among kNN algorithms

In this experiment, we compare kNN algorithms on $k > 1$. Without losing generality, we set a common number $k = 11$ for all kNN classifiers. As shown in Table 2, both $\texttt{CCW}^{\text{MI}}$ and $\texttt{CCW}^{\text{AI}}$ significantly outperforms MI, AI, WDkNN, LMNN, DNet, CCPDT and HDDT.

In the comparison with over-sampling techniques, we focus on MI equipped with safe-level-SMOTE [3] method, shown as "SMOTE" in Table 2. The results we obtained from CCW classifiers are comparable to (better but not significant than) the over-sampling technique under 95% confidence. This observation suggests that if one uses CCW he can obtain results comparable to the cutting-edge sampling technique, so the extra computational cost of data sampling before training can be saved.

5.3 Effects of distance metrics

While in all previous experiments kNN classifiers are performed under Euclidean distance (ℓ_2 norm), in this subsection we provide empirical results that demonstrate the superiority of CCW methods on other distance metrics such as Manhattan distance (ℓ_1 norm) and Chebyshev distance (ℓ_∞ norm). Due to page limits, here we only present the comparisons of "$\texttt{CCW}^{\text{MI}}$ vs. MI". As we can see from Figure 2, $\texttt{CCW}^{\text{MI}}$ can improve MI on all three distance metrics.

6 Conclusions and Future Work

The main focus of this paper is on improving existing kNN algorithms and make them robust to imbalanced data sets. We have shown that conventional kNN algorithms are akin in using only *prior probabilities* of the neighborhood of a test instance to estimate its class labels, which leads to suboptimal performance when dealing with imbalanced data sets.

We have proposed CCW, the likelihood of attribute values given a class label, to weight prototypes before taking them into effect. The use of CCW transforms the original kNN rule of using *prior probabilities* to their corresponding *posteriors*. We have shown that this transformation has the ability of correcting the inherent bias towards majority class in existing kNN algorithms.

We have applied two methods (mixture modeling and Bayesian networks) to estimate training instances' CCW weights, and their effectiveness is confirmed by synthetic examples and comprehensive experiments. When learning Bayesian networks, we construct network structures by applying the K2 algorithm which has an overall time complexity of $O(n^2)$.

In future our plan is to extend the idea of CCW to multiple-label classification problems. We also plan to explore the use of CCW on other supervised learning algorithms such as support vector machines etc.

References

1. Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology and Decision Making **5**(4) (2006) 597–604
2. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE. Journal of Artificial Intelligence Research **16**(1) (2002) 321–357
3. Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Safe-Level-SMOTE. Advances in Knowledge Discovery and Data Mining (PAKDD) (2009) 475–482
4. Cieslak, D., Chawla, N.: Learning Decision Trees for Unbalanced Data. In: Proceedings of ECML PKDD 2008 Part I. 241–256
5. Liu, W., Chawla, S., Cieslak, D., Chawla, N.: A Robust Decision Tree Algorithms for Imbalanced Data Sets. In: Proceedings of the Tenth SIAM International Conference on Data Mining. (2010) 766–777
6. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., et al.: Top 10 algorithms in data mining. Knowledge and Information Systems **14**(1) (2008) 1–37
7. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research **10** (2009) 207–244
8. Min, R., Stanley, D.A., Yuan, Z., Bonner, A., Zhang, Z.: A deep non-linear feature mapping for large-margin knn classification. In: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining. 357–366
9. Yang, T., Cao, L., Zhang, C.: A Novel Prototype Reduction Method for the K-Nearest Neighbor Algrithms with K $\geq$ 1. Advances in Knowledge Discovery and Data Mining (PAKDD 2010 Part II) (2010) 89–100
10. Paredes, R., Vidal, E.: Learning prototypes and distances. Pattern Recognition **39**(2) (2006) 180–188
11. Paredes, R., Vidal, E.: Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Transactions on Pattern Analysis and Machine Intelligence (2006) 1100–1110
12. Wang, J., Neskovic, P., Cooper, L.: Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recognition Letters **28**(2) (2007) 207–213
13. Jahromi, M.Z., Parvinnia, E., John, R.: A method of learning weighted similarity function to improve the performance of nearest neighbor. Information Sciences **179**(17) (2009) 2964–2973
14. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probablistic networks from data. Machine Learning **9**(4) (1992) 309–347
15. Han, E., Karypis, G.: Centroid-based document classification. Principles of Data Mining and Knowledge Discovery (2000) 116–123
16. Asuncion, A., Newman, D.: UCI Machine Learning Repository (2007)
17. Witten, I., Frank, E.: Data mining: practical machine learning tools and techniques with Java implementations. ACM SIGMOD Record **31**(1) (2002) 76–77
18. Hendricks, W., Robey, K.: The sampling distribution of the coefficient of variation. The Annals of Mathematical Statistics **7**(3) (1936) 129–132
19. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning. (2006) 233–240
20. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7** (2006) 1–30