SVM Venn Machine with k-Means Clustering

Chenzhe Zhou, Ilia Nouretdinov, Zhiyuan Luo, and Alex Gammerman

Computer Learning Research Centre,
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK

Abstract. In this paper, we introduce a new method of designing Venn Machine taxonomy based on Support Vector Machines and k-means clustering for both binary and multi-class problems. We compare this algorithm to some other multi-probabilistic predictors including SVM Venn Machine with homogeneous intervals and a recently developed algorithm called Venn-ABERS predictor. These algorithms were tested on a range of real-world data sets. Experimental results are presented and discussed.

Keywords: Venn Machine, Support Vector Machine, k-means clustering.

1 Introduction

Classification is one of the major tasks in machine learning. It gives predictions for the new objects based on known properties learned from the training data set. However, most algorithms could only give single prediction (i.e. label). Demand of probabilistic prediction has arisen in view of the fact that sometimes we appreciate probabilities more than single predictions. A simple example is the probabilistic weather forecasting.

But in some area, single probabilistic prediction has not yet been enough. The term *multi-probabilities* is then brought to mind, namely, we announce several probability distributions for the new label rather than a solitary one. Venn predictor (or Venn Machine) is one of the multi-probabilistic classification systems [8]. There are many Venn predictors, each taxonomy used in the algorithm defines a Venn predictor even if the underlying algorithms are the same.

In our previous paper [10], we introduced a Venn predictor with Support Vector Machines (SVM) as its underlying algorithm, which converts numerical predictions of SVM into a taxonomy. That approach was applicable to any method that initially supplied predictions with prediction scores such as the distance to the hyperplane in SVM. Nonetheless, the process is very simple: all available scores are firstly sorted and then divided into several groups by equal-length intervals according to which interval the score lies. Each of these groups is a category. However, that approach could only be applied in binary cases. In this paper, we propose a method to generalize binary Venn Machine with SVM to a method capable for multi-class cases. Then we consider two alternative methods that may be more accurate: SVM Venn Machine with k-means clustering and Venn-ABERS predictor. These two algorithms are also applicable to any machine learning algorithms with prediction scores.

2 Methodology

In this section, two kinds of Venn predictors that use SVM as their underlying algorithm will be introduced together with our alternative methods. They are SVM Venn Machine with homogeneous intervals (VM-SVM-HI) generalized from the binary-only version of [10] together with our alternative method SVM Venn Machine with k-means clustering (VM-SVM-KM) and the Venn-ABERS predictor based on SVM (VA-SVM) proposed by Vladimir Vovk [7]. The former two algorithms could be implemented in both multi-class cases and binary cases, while VA-SVM could only deal with binary data sets.

2.1 Venn Machine

Venn Machine is a multi-probabilistic predictor described in [8]. The basic idea of Venn Machine is to divide every example into its corresponding category based on certain rules and then the frequencies of labels in the chosen category are used as probabilities for the new object's label. Taxonomy is the way how the examples are divided into categories. The underlying algorithm is the algorithm used in the taxonomy.

Assuming a standard machine learning classification problem: given a training set of examples z_1, z_2, ..., z_{n-1}. Each z_i consists of a pair of object x_i and label y_i. The possible labels y_i ($y_i \in \mathbf{Y}$) are finite. And we are also given a test object x_n. Our task is to predict the label y_n for the new object x_n and give the estimation of the likelihood that our prediction is correct.

Supposing we have a taxonomy A_n, consider a label $y \in \mathbf{Y}$ for the new object x_n. A_n assigns a category τ_i to an example z_i

$$\tau_i = A_n(\lbag z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n \rbag, z_i) \tag{1}$$

where n is the number of objects in the bag, $\tau_i \in \mathbf{T}$ is one of the finite categories and z_i is the pair (x_i, y_i), z_n is the pair (x_n, y).

Moreover, we assign z_i and z_j to the same category if and only if

$$A_n(\lbag z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n \rbag, z_i) = A_n(\lbag z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_n \rbag, z_j) \tag{2}$$

The category τ_n contains $z_n = (x_n, y)$. Let p_y be the empirical probability distribution of the labels in category τ_n.

$$p_y(y') := \frac{|\{(x^*, y^*) \in \tau_n : y^* = y'\}|}{|\tau_n|} \tag{3}$$

p_y is a probability distribution on $\mathbf{Y}$.

Having tried every possible label for x_n, we get a Venn predictor. The predictor $P_n := \{p_y : y \in \mathbf{Y}\}$ is a multi-probabilistic predictor consists of K distributions, where $K = |\mathbf{Y}|$. Then we could calculate a $K \times K$ frequency matrix P. The *quality* of a column is the minimum entry of the column. Let the *best* column which has the highest *quality* be j_{best}. Then our predicted label is j_{best} and the interval of possibility that our prediction is correct is

$$[\min_{i=1,\ldots,K} P_{i,j_{best}} \ , \ \max_{i=1,\ldots,K} P_{i,j_{best}}] \tag{4}$$

Underlying Algorithm for Taxonomy. Any algorithm that generates or predicts a numeric score for the example could be implemented in our taxonomy. However, we mainly focus on Venn predictors with SVM as the underlying algorithm in this paper. The decision function in SVM is a kind of scoring functions. Therefore, we use the values derived from the decision function of SVM (i.e. the values prior to applying a sign function) as part of our design.

Homogenous Intervals. One of the simplest ways to design taxonomies is stated as follows. Firstly we use the training set to train an SVM and calculate the decision values ($d(x) = \langle w, x_i \rangle + d$) of all examples in the training set and the new object. Secondly the whole range of decision values obtained will be divided into several intervals of equal length. Each interval is a category and objects of which the decision values fall into the same interval are of the same category. This design was introduced in [10] and could only used in binary case. Now we will discuss the generalization and alternative to it.

Combined Decision Function. In multi-class cases, we will have several binary SVM classifiers regardless of whether One-vs-One or One-vs-All approach is used. A scheme for multi-class SVM using One-vs-All approach was developed by Lambrou et al. in [5], which uses the largest decision value as the score. Generally, One-vs-One SVM is more efficient in accuracy than One-vs-All SVM. Therefore, we need to develop a new function to combine the outputs of all One-vs-One SVM classifiers and transform them into a single prediction score which could be used by Venn Machine. We call such function a *Combined Decision Function.*

For a data set with k possible labels: $\{0, 1, \ldots, k-1\}$, there are $k(k-1)/2$ binary SVM if we use One-vs-One approach. For each possible label, there are $k-1$ related SVM decision functions. Then we use (5) to calculate the combined decision function $D(x)$ for the new example x,

$$D(x) = \hat{y} + \frac{1}{k-1} \sum_{i=0, i \neq \hat{y}}^{k-1} N(f_{\hat{y}i}(x)) \tag{5}$$

where $\hat{y}$ is the overall predicted label done by max-wins voting strategy in One-vs-One SVM, $f_{\hat{y}i}(x)$ is the decision function of SVM classifier on $\hat{y}$-vs-i, N is a function that does the normalized transformation to $[0, 1]$. Another point we need to declare here is that in $f_{\hat{y}i}(x)$ we always put $\hat{y}$ before i which means we need to apply an opposite operation when $\hat{y}$ is greater than i. Since the examples of label $\hat{y}$ are treated as negative examples in i-vs-$\hat{y}$ classifier of a binary SVM.

This function firstly selects all $k-1$ related SVM and applies an opposite operation if $\hat{y}$ is not treated as the positive class in the binary SVM classifier. Then it does the normalisation to transform the values into $[0, 1]$. Finally, we output the arithmetic mean of them added with $\hat{y}$ as the combined decision value of new example x. The reason that adding $\hat{y}$ to the arithmetic mean is that it could prevent the decision values of different classes stack at the same area.

Dividing Intervals by k-Means Clustering. Instead of dividing the intervals homogeneously, we came up with a new dividing scheme, which uses k-means clustering [4, 6] to divide all decision values.

k-means clustering is a cluster analysis method which aims to divide n objects into k clusters in which each object belongs to the cluster with the nearest mean. Given a set of objects $(x_1, x_2, \ldots, x_n)$, where each object $x_i \in \mathbf{R}^d$ is a d-dimensional real vector, k-means clustering aims to partition the n objects into k sets $(k \leq n)$ $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimise the within-cluster sum of squares (WCSS):

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \tag{6}$$

where μ_i is the mean value of points in S_i.

In our design, dimension d is fixed to "1", while the number of clusters is equal to the number of possible labels. So the heuristic algorithm we used could be described as below.

1. k initial means values are randomly generated within the data domain.
2. k clusters are created (or reassigned) by associating every object with the nearest mean value.
3. The centroid of each of the k clusters becomes the new mean value.
4. Steps 2 and 3 are repeated until the change of WCSS (6) between two states declines to be less than $\epsilon = 10^{-4}$.

Having applied k-means clustering, we divided the decision values into categories which could be used to calculate the matrix for new examples and make the probabilistic predictions as the standard Venn Machine does.

2.2 Venn-ABERS Predictor

Venn-ABERS predictor is a recently developed algorithm for multi-probabilistic prediction. It is modified from Zadrozny and Elkan's procedure of probability forecasting [9], which cannot be well calibrated. The modification introduced Venn predictors into the procedure to overcome the problem of potentially weak calibration as a result of the fact that Venn predictors are always well calibrated and guaranteed to be well calibrated under the exchangeability assumption. The basic idea of pre-trained Venn-ABERS predictor is that the training set is split into two parts: the proper training set and the calibration set. The proper training set is used to train the learning machine and predict the label for new examples, while the calibration set is used to calculate the probabilistic outputs for the predicted labels. The calibration set will be turned into a monotonically increasing set in this algorithm according to [1].

Before we discuss Venn-ABERS predictor, there are some notions to be introduced yet. First notion is the term "*scoring algorithm*". Scoring algorithm is an algorithm that trains a classifier on the training set and uses the classifier to output a prediction score $s(x)$ for the new example x and predicts the label of x to be "1" if and only if $s(x) \geq c$ (c is a fixed threshold). So s is hereby called the *scoring function*. Many machine learning algorithms for classification are scoring algorithms. In our case, as what SVM defines, the decision function of SVM is a scoring function, since we assign a new example the positive label "+1" if and only if its decision value is greater than zero and vice versa for the negative label. The second notion is "*isotonic calibrator*", which is a monotonically increasing function on the set $\{s(x_1), \ldots, s(x_l)\}$ that maximizes the likelihood

$$\prod_{i=1}^{l} p_i, \text{ where } p_i := \begin{cases} g(s(x_i)) & \text{if } y_i = 1 \\ 1 - g(s(x_i)) & \text{if } y_i = 0 \end{cases} \tag{7}$$

this function g is unique and can be found by using the "pool-adjacent violators algorithm" (PAVA) introduced in [1].

The workflow of Venn-ABERS predictor is as follows. Assuming a standard binary machine learning problem: a training set of examples $z_1, z_2, \ldots, z_l$. Each z_i consists of a pair of object x_i, and label y_i. The possible labels are binary, that is, $y \in \{0, 1\}$. And we are also given a test object x. Our task is to predict the label y for the new object x and give the estimation of the likelihood that our prediction is correct.

Let us split the training set $\lbag z_1, z_2, \ldots, z_l \rbag$ into two parts: the proper training set $\lbag z_1, z_2, \ldots, z_m \rbag$ of size m ($m < l$) and the calibration set $\lbag z_{m+1}, z_{m+2}, \ldots, z_l \rbag$. And $s : \mathbf{X} \to \mathbb{R}$ is the scoring function of training set $\lbag z_1, z_2, \ldots, z_m \rbag$. Given a new example x, we have two calibrators. Let g_0 be the isotonic calibrator for $\lbag (s(x_{m+1}), y_{m+1}), (s(x_{m+2}), y_{m+2}), \ldots, (s(x_l), y_l), (s(x), 0) \rbag$, g_1 be the calibrator for $\lbag (s(x_{m+1}), y_{m+1}), (s(x_{m+2}), y_{m+2}), \ldots, (s(x_l), y_l), (s(x), 1) \rbag$.

To achieve the isotonic calibrator, we do the followings according to the definition of PAVA. First we arrange the pairs $(s(x_i), y_i)$ in the increasing order according to the values of score function $s(x_i)$. Having obtained a binary sequence consisting of labels y_i, we applied PAVA to find the increasing sequence of them. The final isotonic calibrator g is a function mapping the increasing scores to the increasing sequence (i.e. probabilities). As the score increases, the object is more likely to be "1" in correlation with the increasing sequence.

Then the multi-probability prediction outputs for that the predicted label should be "1" is $\{p_0, p_1\}$, where $p_0 := g_0(s(x))$ and $p_1 := g_1(s(x))$. And for the reason that we need to predict the probability for the prediction label is correct, we should transform the bounds $\{p_0, p_1\}$ to $\{1 - p_1, 1 - p_0\}$ when the predicted label is "0".

3 Experimental Results

To compare our algorithm to SVM Venn Machine with homogeneous intervals and Venn-ABERS predictor, we used eight data sets from the real world which could be easily obtained from UCI Repository (`http://archive.ics.uci.edu/ml/`) except that SVMguide1 is obtained from the website of LibSVM [3]. The data sets we

Table 1. Main characteristics for each data set

Data Set	# of Objects	# of Features	# of Classes	Training Set Size	Testing Set Size
WBC	683	10	2	400	283
SVMguide1	7089	4	2	3089	4000
Splice	3175	60	2	1000	2175
Satimage	6435	36	6	4435	2000
Segment	2310	19	7	1500	810
DNA	3186	180	3	2000	1186
Wine	178	13	3	100	78
Vehicle	846	18	4	500	346

used in this paper could be divided into two parts based on their number of classes. The details of these data sets are summarised in Table 1.

3.1 Experimental Settings

For VM-SVM-KM, the number of clusters and the initial means, which are the two key features of k-means clustering, are often regarded as its biggest drawbacks. The number of clusters is an input parameter: an inappropriate choice of k may yield poor results. That is why, when performing k-means clustering, it is important to run diagnostic checks for determining the number of clusters in the data set. The choice of initial means might lead the convergence to a local minimum which may produce counterintuitive results. A good design of a combined decision function could make it easier to avoid these two drawbacks.

To have a more intuitive view of our combined decision function described in (5), we applied the algorithm to Satimage data set and plotted the histogram in Fig. 1, roughly representing the distribution of the decision values.

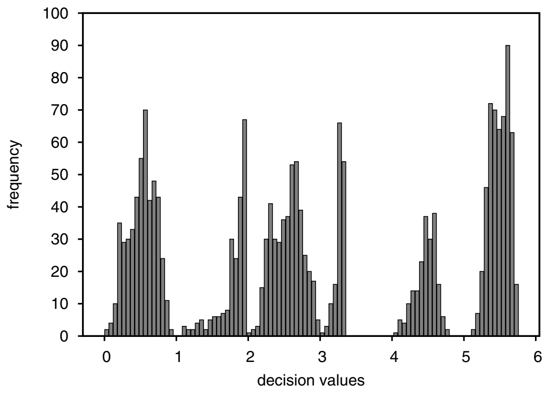


Fig. 1. Histogram of combined decision values for the Satimage data set

It can be seen obviously from the figure that there were 6 clusters in the data set, the exact number of the possible labels. The reason for this is that the decision function spreads out the values into $(0, k)$ by adding the most possible labels. Furthermore, each cluster i $(i = 1, 2, \ldots, k)$ is approximately within the range of $(i - 1, i)$, which means we could choose the initial means from each range to avoid the local minimum trap as much as possible and speed up the convergence process. We conducted k-means clustering to these decision values and calculated the 6 centroids: $0.63, 1.91, 2.64, 3.33, 4.57, 5.59$. The result seems to be a reasonable reflection of the histogram.

Then we could come to our decision that we set the number of clusters the same as the number of possible labels and we choose the initial means as $0.5, 1.5, \ldots, k - 0.5$ if the possible labels are $0, 1, \ldots, k - 1$.

Additionally, we need to notice that k-means clustering uses Euclidean distance as a metric and variance as a measure of cluster scatter, which makes it tend to produce equal-sized clusters. Since data is split halfway between cluster means, this can lead to suboptimal splits as some objects will be attributed to the incorrect cluster, especially for unbalanced data set as Satimage data set.

Except all the settings for the underlying algorithm, we need another setting for VA-SVM. It is the size of the calibration set. Having given careful consideration to both accuracy and narrowness of the bounds, we decided to take 30% of the whole data set as the calibration set, And the calibration set was stratified selected from the whole training set, which means the distribution of classes in the calibration set was the same as in the training set.

Although the size of proper training set in VA-SVM is smaller comparing to the size of training set in our algorithms, this is still a fair comparison because we use the same original training set for all algorithms, otherwise VA-SVM will need extra examples for probabilistic predictions. We also noticed that Venn-ABERS predictor is an inductive Venn predictor while Venn Machine is a transductive Venn predictor. The gap between inductive and transductive learning algorithms are not distinguishable in our offline setting. Because in offline setting, we use the fixed predictors to make predictions for testing set. Furthermore, in VA-SVM we repeat the computations of isotonic calibrators for each testing object which still involve all examples in calibration set.

3.2 Comparisons and Results

For binary cases, we applied VM-SVM-KM, VM-SVM-HI and VA-SVM to the data sets in the offline setting. While for multi-class cases, we only applied VM-SVM-KM and VM-SVM-HI to the data sets in both offline setting and online setting. Hence, there were three comparisons described as below. All the SVMs in these algorithms were using RBF kernel. Additionally, the parameters of SVM for each data set, including cost C and σ in RBF kernel, were determined by grid search on the training set and retained the same over corresponding algorithms respectively. The algorithms were compared in terms of their accuracies and probabilistic outputs in these data sets. In addition, we calculated the Brier scores (introduced in [2]) of the mean of the probabilistic bounds as evaluation for binary data sets.

The experimental results of VM-SVM-KM compared with VM-SVM-HI and VA-SVM are shown in Table 2.

Table 2. The offline accuracy and probability results on the binary data set

Data Set	Taxonomy	Accuracy	Prob. Outputs	Brier Score
WBC	VM-SVM-KM	97.53%	[86.34%,98.94%]	0.0325
	VM-SVM-HI	97.22%	[83.63%,98.70%]	0.0369
	VA-SVM	97.17%	[85.67%,95.97%]	0.0315
SVMguide1	VM-SVM-KM	96.93%	[91.27%,98.42%]	0.0362
	VM-SVM-HI	95.79%	[89.59%,98.97%]	0.0406
	VA-SVM	95.95%	[93.67%,96.29%]	0.0370
Splice	VM-SVM-KM	90.21%	[82.44%,96.07%]	0.0884
	VM-SVM-HI	89.52%	[80.15%,97.35%]	0.0939
	VA-SVM	89.15%	[83.40%,88.32%]	0.0878

The comparison results of our algorithm against VM-SVM-HI for all multi-class data sets in the offline setting are shown in Table 3.

Table 3. The offline accuracy and probability results on the multi-class data set

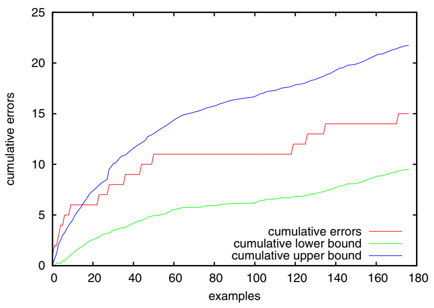
Data Set	Algorithm	Accuracy	Probabilistic Outputs
Satimage	VM-SVM-HI	84.18%	[75.48%,96.92%]
	VM-SVM-KM	86.56%	[81.18%,93.33%]
Segment	VM-SVM-HI	90.88%	[74.61%,96.68%]
	VM-SVM-KM	91.65%	[75.64%,95.60%]
DNA	VM-SVM-HI	94.34%	[81.39%,98.07%]
	VM-SVM-KM	96.65%	[87.25%,99.48%]
Wine	VM-SVM-HI	92.30%	[81.19%,97.11%]
	VM-SVM-KM	96.11%	[86.53%,98.47%]
Vehicle	VM-SVM-HI	67.63%	[56.42%,77.48%]
	VM-SVM-KM	69.15%	[60.65%,79.20%]

And the results for the online setting are shown in Table 4.

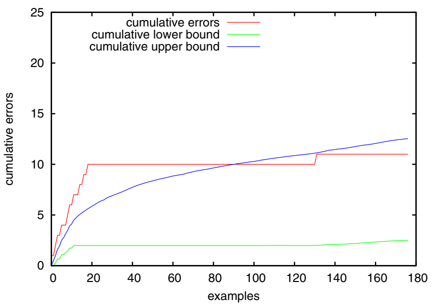
In order to giving a more intuitive comparison, we also give figures on online performance for Wine data set in Fig. 2.

Table 4. The online accuracy and probability results on the multi-class data set

Data Set	Algorithm	Accuracy	Probabilistic Outputs
Satimage	VM-SVM-HI	80.94%	[80.20%,81.69%]
	VM-SVM-KM	83.40%	[83.24%,83.86%]
Segment	VM-SVM-HI	88.40%	[88.92%,93.20%]
	VM-SVM-KM	89.96%	[90.11%,91.50%]
DNA	VM-SVM-HI	89.12%	[88.46%,89.86%]
	VM-SVM-KM	89.70%	[89.25%,90.48%]
Wine	VM-SVM-HI	91.53%	[87.57%,94.35%]
	VM-SVM-KM	93.22%	[91.67%,96.87%]
Vehicle	VM-SVM-HI	66.04%	[70.24%,72.17%]
	VM-SVM-KM	67.83%	[69.48%,71.02%]



(a) VM-SVM-HI



(b) VM-SVM-KM

Fig. 2. Comparison of online performances for the Wine data set

4 Discussion and Conclusion

From the results shown in Table 2 which comparing our method to VM-SVM-HI and VA-SVM, we could draw the following conclusions.

First, VM-SVM-KM performed better in accuracy among these three data sets nevertheless the increases were small. Furthermore, VA-SVM used 30% of the training set as the calibration set which did not participate in the training of classifiers; hence it may lead to worse results. Second, the accuracies of VA-SVM slightly outnumbered the upper bound in WBC and Splice data sets, which could be due to the offline setting. Third, the probability bounds of VA-SVM were the narrowest while VM-SVM-HI had the widest bounds and VM-SVM-KM was in-between. This is the advantage of VA-SVM in view of our preference for narrow bounds. It is also backed by the Brier scores results: VA-SVM and VM-SVM-KM had close Brier scores while VM-SVM-HI had the worst results.

Another point is that VA-SVM does not calibrate their predicted label according to the probability, more specifically it is an algorithm that generates the probabilities from the scores only, while our algorithm gives predictions based on the highest likelihood. Except the improvement in accuracy, VM-SVM-KM is easy to configure because the number of clusters is the only input parameter of this algorithm which is equal to the number of classes.

From the results presented in Table 3 and Table 4 where the performance of VM-SVM-KM is compared with VM-SVM-HI in both offline and online setting, we can discover the following points.

First, it can be observed that all accuracies were within the probabilistic outputs in the offline setting, while in the online setting the accuracies exceeded the bounds in Segment and Vehicle data sets. Second, after implementing the k-means clustering, the accuracies are improved in both settings. However, in the offline setting, the improvements ranged from 0.8% to 3.8% depending on the data sets. In the online setting, the difference between these two algorithms became smaller, only 0.6% to 2.5%. Third, probability bounds become narrower after applying the k-means clustering, mostly benefiting from the rise of lower bounds. An intuitive comparison is shown in Fig. 2. The cumulative errors and cumulative error bounds in the figures all decreased after implementing k-means clustering, and the bounds became narrower in the meantime.

In summary, the improvement in each of the eight data sets was not significant which is due to the consistency of these data sets. Nevertheless, we still believe that SVM Venn Machine with k-means clustering is better when compared with homogeneous intervals since it could yield better accuracy and narrower bounds. However, in comparison with Venn-ABERS predictor, our algorithm is good on accuracy and weak on narrowness of the bounds. Despite that, our algorithm is easier to set up, and it predicts the most likely label while Venn-ABERS predictor only generates the probabilities.

References

1. Ayer, M., Brunk, H.D., Ewing, G.M., Reid, W.T., Silverman, E.: An empirical distribution function for sampling with incomplete information. Ann. Math. Statist. 26(4), 641–647 (1955)
2. Brier, G.W.: Verification of forecasts expressed in terms of probability. Monthly Weather Review 78(1), 1–3 (1950)
3. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 27 (2011)
4. Forgy, E.W.: Cluster analysis of multivariate data: Efficiency vs interpretability of classifications. Biometrics 21, 768–769 (1965)
5. Lambrou, A., Papadopoulos, H., Nouretdinov, I., Gammerman, A.: Reliable probability estimates based on support vector machines for large multiclass datasets. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H., Karatzas, K., Sioutas, S. (eds.) AIAI 2012, Part II. IFIP AICT, vol. 382, pp. 182–191. Springer, Heidelberg (2012)
6. Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory 28, 129–137 (1982)
7. Vovk, V.: Venn predictors and isotonic regression. CoRR abs/1211.0025 (2012)
8. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer-Verlag New York, Inc., Secaucus (2005)
9. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 694–699. ACM (2002)
10. Zhou, C., Nouretdinov, I., Luo, Z., Adamskiy, D., Randell, L., Coldham, N., Gammerman, A.: A comparison of venn machine with platt's method in probabilistic outputs. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H. (eds.) EANN/AIAI 2011, Part II. IFIP AICT, vol. 364, pp. 483–490. Springer, Heidelberg (2011)