19

Classifiers Based on Inverted Distances

Marcel Jirina and Marcel Jirina, Jr.
Institute of Computer Science AS CR,
Faculty of Biomedical Engineering,
Czech Technical University in Prague,
Czech Republic

1. Introduction

In this chapter we describe an elaborated yet simple classification method (IINC) that can outperform a range of standard classification methods of data mining, e.g. k-nearest neighbors, Naïve Bayes Classifiers' as well as SVM. In any case the method is an alternative to well-known and widely used classification methods.
There is a lot of classification methods, simpler or very sophisticated. Some standard methods of probability density estimate for classification are based on the nearest neighbors method which uses ratio k/V, where k is the number of points of a given class from the training set in a suitable ball of volume V with center at point x (Silverman, 1990; Duda et al., 2000; Cháves et al., 2001) sometimes denoted as query point. For probability density estimation by the k-nearest-neighbor (k-NN) method in E_n, the best value of k must be carefully tuned to find optimal results. Often used role of thumb is that k equals to square root of number of samples of the learning set. Nearest neighbors methods exhibit sometimes surprisingly good results see e.g. (Merz, 2010; Kim & Ghahramani, 2006). Bayesian methods form the other class of most reputable non-parametric methods (Duda et al., 2000; Kim & Ghahramani, 2006). Random trees or random forest approach belong among complex, but the best classification methods as well as neural networks of different types (Bock, 2004). The disadvantage of many these methods is the necessity to find proper set of internal parameters of the system. This problem is often solved by the use of genetic optimization as in the case of complex neural networks see e.g. (Hakl et al., 2002).
First, we will provide a shot overview of the basic idea of the IINC and its features and show a simple demonstative example of a pragmatic approach to a simple classification task. Second, we give a deeper mathematical insight into the method and finally we will demonstrate the power of the IINC on data sets from two well-known repository real-life tasks.

2. Classifier background - idea and motivation

In general, if we have estimates of the probability that a given sample (query point) belongs to a given class, we can easily construct a classifier. We just compare the individual probabilities and select the class with the highest probability. The presented IINC works in the same way, but the probabilities are estimated in a special way that is based on summing up the inverted indexes of neighbors.

We show a practical approach to the classification of data into two classes (extending the classifier to be able to classify to more than two classes will be then straightforward).
Let all samples of the learning set regardless of the class be sorted according to their distances from the query point x. Let indexes be assigned to these points so that index 1 is assigned to the nearest neighbor, index 2 to the second nearest neighbor etc.

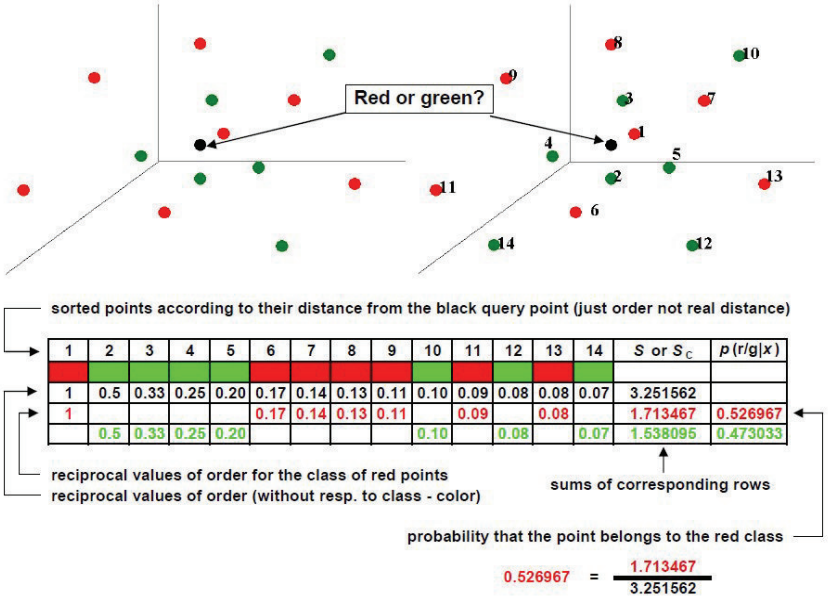
Let us compute sums $S_0(x) = \frac{1}{N_0} \sum_{i=1\,(c=0)}^{N} 1/i$ and $S_1(x) = \frac{1}{N_1} \sum_{i=1\,(c=1)}^{N} 1/i$, i.e. the sums of the reciprocals of the indexes of samples from class c = 0 and from class c = 1. N_0 and N_1 are the numbers of samples of class 0 and class 1, respectively, $N_0 + N_1 = N$, N is the total number of samples available.
The probability that point x belongs to class 0 is

$$p(c = 0 \mid x) \cong \frac{S_0(x)}{S_0(x) + S_1(x)}$$

and similarly the probability that point x belongs to class 1 is

$$p(c = 1 \mid x) \cong \frac{S_1(x)}{S_0(x) + S_1(x)} .$$

When a discriminant threshold θ is chosen (e.g. θ = 0.5), then if $p(c = 1 \mid x) \geq \theta$ point x is of class 1 else it is of class 0. This is the same procedure as in other classification approaches where the output is the estimation of probability (naïve Bayes) or any real valued variable (neural networks). The value of the threshold can be optimized with regard to the minimum classification error.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	S or S_c	$p(r/g\|x)$
1	0.5	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09	0.08	0.08	0.07	3.251562	
1					0.17	0.14	0.13	0.11		0.09		0.08		1.713467	0.526967
	0.5	0.33	0.25	0.20					0.10		0.08		0.07	1.538095	0.473033

Fig. 1. Example of a classification task for a two-class problem of spatial data

As shown, the IINC is very simple. It is based only on the sum of inverted indexes of the nearest neighbors. It opens the question whether it is as powerful as stated above.
In the above formulas the actual data do not appear directly, but are hidden behind the indexes that express their distance from a given sample (query point). To be able to get the indexes we have to sort the original data according to their distances from a particular sample. The only information we work with is their order, not the real distance! To compare distances we need proper metrics (just the L_1 (absolute) metrics yields the best results). And so on. In other words, there are many assumptions that have to be fulfilled (and fortunately they are fulfilled in standard classification tasks) to concentrate them into a simple presented classification algorithm IINC. To vindicate the correctness of the algorithm we offer a deeper mathematical insight into the IINC and demonstrate the IINC on real-life classification tasks.

3. Mathematical background of IINC

Let us consider partial influences of individual points on the probability that point x is of class c. Each point of class c in the neighborhood of point x adds a little to the probability that point x is of class c, where $c \in \{0, 1\}$ is the class mark. This influence grows larger the closer the point considered is to point x and vice versa. This observation is based on the finding of (Cover & Hart, 1967) that the nearest neighbor has the largest influence on the proper estimation to what class point x belongs. Let us assume – for proof see (Jirina & Jirina, 2008) – that the influence to the probability that point x is of class c (the nearest neighbor of class c) is 1, the influence of the second nearest neighbor is 1/2, the influence of the third nearest neighbor is 1/3, etc. We show further that just these values of influence lead to improved classification. Let $p_1(c \mid x, r_i)$ be the probability that query point x is of class c if neighbor point number i is of the same class as point x, K is a constant that is used to normalize the probability that point x belongs to any class to 1:

For the first (nearest) point i = 1 $\quad p_1(c \mid x, r_1) = K.1$,

for the second point i = 2 $\quad p_1(c \mid x, r_2) = K\frac{1}{2}$,

and so on, generally for point No. i $\quad p_1(c \mid x, r_i) = K\frac{1}{i}$.

Individual points are independent and then we can sum up these probabilities. Thus we add the partial influences of k individual points together by summing up

$$p(c \mid x, r_k) = \sum_{i=1(c)}^{k} p_1(c \mid x, r_i) = K \sum_{i=1(c)}^{k} 1/i \cdot$$

The sum goes over indexes i for which the corresponding samples of the learning set are of class c. Let

$$S_c = \sum_{i=1(c)}^{k} 1/i$$

and let

$$S = \sum_{i=1}^{N} 1 / i$$

(This is, in fact, so-called harmonic number HN, the sum of truncated harmonic series.) The estimation of the probability that query point x belongs to class c is

$$p(x \mid c) = \frac{S_c}{S} .$$

The approach is based on the hypothesis that the influence, the weight of a neighbor, is proportional to the reciprocal of its order number just as it is to its distance from the query point.
The hypothesis above is equivalent to the assumption that the influence of individual points of the learning set is governed by Zipfian distribution (Zipf's law), (Zipf, 1968).
It is also possible to show that the use of 1/i has a close connection to the correlation integral and correlation dimension and thus to the dynamics and true data dimensionality of processes that generate the data we wish to separate. It generally leads to better classification.

3.1 Data and the learning set

Let us consider only two classes for a classification task. Let the learning set U of total N samples be given in the form of a matrix X^T with N rows and n columns. Each sample corresponds to one row of X^T and, at the same time, corresponds to a point in n-dimensional space R_n, where n is the sample space dimension. The learning set consists of points (rows, samples) of two classes $c \in \{0, 1\}$, i.e. each row (point or sample) belongs to one of these two classes. Then, the learning set can be formally described as $U = U_0 \cup U_1$, $U_0 \cap U_1 = \varnothing$, $U_c = \{x_{cs}\}$, $s = 1, 2, \ldots N_c$, $c \in \{0, 1\}$. Nc is the number of samples of class c, $N_0 + N_1 = N$, and $x_{cs}=\{x_{cs1}, x_{cs2},\ldots x_{csn}\}$ is the data sample of class c.
As we need to express which sample is closer to or further from a given point x, we can bind the index of the point of the learning set with its distance from point x. Therefore, let U be a learning set composed of points (patterns, samples) x_i, where i is the index of a point regardless of the class to which it belongs; x_i is the i-th nearest neighbor of point x. By the symbol i(c), we denote those indexes i for which point $x_i(c)$ belongs to class c.
As we need to work with metrics space we have to transform general data space to metric space. Therefore, we use normalized data, i.e. each variable x_{csj} (j fixed, s = 1, 2, ... N, c = 0 or 1 corresponds to the j-th column of matrix X^T) has zero mean and unit variance. The empirical means and variances of individual variables are computed from the whole learning set, i.e. regardless of the classes. Later they are used for the normalization of testing samples. We use Euclidean (L_2) and absolute (L_1) metrics here.

3.2 Mapping the distribution

First we introduce two important notions, the probability distribution mapping function and the distribution density mapping function. It is interesting that there is a close connection between the probability distribution mapping function and the correlation integral by Grassberger and Procaccia (Grassberger & Procaccia, 1983).
Let us have an example of a ball in an n-dimensional space containing uniformly distributed points over its volume. Let us divide the ball into concentric "peels" of the same volume.

Using the formula $r_i = \sqrt[n]{V_i / S(n)}$, which is, in fact, inverted formula for volume V_i of an n-dimensional ball of radius r_i, we obtain a quite interesting succession of radii corresponding to the individual volumes - peels. The symbol S(n) denotes the volume of a ball with unit radius in E_n; note $S(3) = 4/3\pi$. A mapping between the mean density ρ_i in an i-th peel and its radius r_i is $\rho_i = p(r_i)$; $p(r_i)$ is the mean probability density in the i-th ball peel with radius r_i. The probability distribution of points in the neighborhood of a query point x is thus simplified to the function $p(r_i)$ of a scalar variable r_i. We call this function a probability distribution mapping function D(x, r) and its partial differentiation with respect to r the distribution density mapping function d(x, r). Functions D(x, r) and d(x, r) for x fixed are, in fact, the probability distribution function and the probability density function of variable r, i.e. of distances of all points from the query point x. More exact definitions follow.

Definition 1. Probability distribution mapping function D(x, r) of the query point x is function $D(x,r) = \int_{B(x,r)} p(z)dz$, where r is the distance from the query point and B(x, r) is a ball with center x and radius r.

Definition 2. Distribution density mapping function d(x, r) of the query point x is function $d(x,r) = \frac{\partial}{\partial r} D(x,r)$, where D(x, r) is a probability distribution mapping function of the query point x and radius r.

Note. When it is necessary to differentiate the class of a point in distance r from point x, we write D(x, r, c) or d(x, r, c).

3.3 Zipfian distribution (Zipf's law)

The Zipfian distribution (Zipf's law) (Zipf, 1968; Zipf-Mandelbrot, 2009) predicts that out of a population of N elements, the frequency of elements of rank k, f(i;s,N), is

$$f(i;s,N) = \frac{1/i^s}{\sum_{t=1}^{N} 1/t^s},$$

where N is the number of elements, i is their rank, s is the value of the exponent characterizing the distribution.

The law may also be written:

$$f(i;s,N) = \frac{1}{i^s H_{N,s}},$$

where HN, s is the N-th generalized harmonic number.

The simplest case of Zipf's law is a "1/f function". Given a set of Zipfian distributed frequencies of the occurrence of some objects, sorted from the most common to the least common, the second most common frequency will occur 1/2 as often as the first. The third most common frequency will occur 1/3 as often as the first. The n-th most common frequency will occur 1/i as often as the first. However, this cannot hold exactly, because items must occur an integer number of times: there cannot be 2.5 occurrences of anything. Nevertheless, over fairly wide ranges, and to a fairly good approximation, many natural phenomena obey Zipf's law. Note that in the case of a "1/f function", i.e. s = 1, N must be finite; otherwise the denominator is a sum of harmonic series, which is divergent. This is not true if exponent s exceeds 1, s > 1, then

$$\zeta(s) = \sum_{t=1}^{\infty} \frac{1}{t^s} < \infty \ ,$$

where ζ is Riemann's zeta function.
The original motivation of Zipf's law was a corpus of natural language utterances. The frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, which occurs twice as often as the fourth most frequent word, etc. In this example of the frequency of words in the English language, N is the number of words in the English language and, if we use the classic version of Zipf's law, the exponent s is 1. f(i; s,N) will then be the fraction of the time the i-th most common word occurs. It is easily seen that the distribution is normalized, i.e., the predicted frequencies sum to 1:

$$\sum_{i=1}^{N} f(i;s,N) = 1 \ .$$

3.4 Probability density estimation

As we mentioned above the classification method presented is based on estimation of a probability to which class point x of the data space belongs. The sum of inverted neighbors' indexes can be utilized for the probability estimation with advantage. In this section we give a deeper mathemeatical insight into the probability density estimation and and thus vindication of the method presented.
Let us assume that the best case for the distribution density estimation is the case of uniform distribution. This conjecture follows from generally accepted meaning (often implicit only) that best results are usually obtained in cases which are not too far from uniform distribution. For both classes distributed uniformly the probability that point x belongs to a class is given exactly by apriori probability. Then we are looking for a transformation by which we get the probability distribution mapping function linear and its derivative, the distribution density mapping function, constant.
Let indexes i be assigned to points (samples) of the learning set without respect to a given class so that i = 1 is assigned to the nearest neighbor of point x, i = 2 to the second nearest neighbor etc. We have finite learning set of size N samples and N_c samples of each class. The same number of samples of both classes is assumed without loss of generality in the theorem and proof as follows.
Theorem. Let the task of classification into two classes be given and let the best case for the distribution density estimation is the case of uniform distribution holds. Let the size of the learning set be N and let both classes have the same number of samples. Let i be the index of the i-th nearest neighbor of point x (without considering neighbor's class) and r_i be its distance from the point x. Then

$$p(c \mid x) = \lim_{N\to\infty} \frac{\sum_{i=1(c)}^{N} 1/i}{\sum_{i=1}^{N} 1/i} \qquad (1)$$

(the upper sum goes over indexes i for which the corresponding samples are of class c) is probability that point x belongs to class c.

Proof. For each query point x one can state the probability distribution mapping function D(x, r_i, c). We approximate this function so that it holds (K is a constant)

$$D(x, r_i^q, c) = K r_i^q$$

in the neighborhood of point x. Using derivation, according to variable z = r_i^q, we get $d(x, r_i^q, c) = K$. By the use of z = r_i^q, the space is mapped ("distorted") so that the distribution density mapping function is constant in the neighborhood of point x for any particular distribution. The particular distribution is characterized by particular value of the distribution mapping exponent q in point x. In this mapping the distribution of points of class c is uniform.

Let us consider sum $\sum_{i=2}^{N} d(x, r_i^q, c) / r_i^q$. For this sum we have

$$\lim_{N\to\infty} \sum_{i=2}^{N} d(x, r_i^q, c) / r_i^q = p(c \mid x) \lim_{N\to\infty} \sum_{i=2}^{N} 1 / r_i^q$$

because d(x, r_i^q, c) = d(x, z, c) = p(c | x) for all i (uniform distribution has a constant density). By the use of $z_i = r_i^q$, the space is distorted so that the distribution density mapping function $d(x, z_i, c)$ is constant in the neighborhood of point x for any particular distribution. This local property we extend to wider neighborhood to have $d(x, r_i^q, c) = d(x, z_i, c)$ constant in the whole data space. For it the exponent q need not be a constant but can be a function q = q(i, c). Let $r_i^{q(i,c)} = k_1 i$ for all i of class c; k_1 is a constant. (From the last formula one could derive the q(i, c), but we need not it.) We rewrite the equation above in form

$$\lim_{N\to\infty} \sum_{i=2}^{N} d(x, r_i^{q(i,c)}, c) / r_i^{q(i,c)} = p(c \mid x) \lim_{N\to\infty} \sum_{i=2}^{N} 1 / r_i^{q(i,c)}$$

and then in form

$$\lim_{N\to\infty} \sum_{i=2}^{N} d(x, r_i^q, c) / i = p(c \mid x) \lim_{N\to\infty} \sum_{i=2}^{N} 1 / i .$$

Given the learning set, we have the space around point x "sampled" by individual points of the learning set. Let $p_c(r_i)$ be an a-posteriori probability point i in distance r_i from the query point x is of the class c. Then $p_c(r_i)$ is equal to 1 if point i is of class c and $p_c(r_i)$ is equal to zero, if the point is of the other class. Then the particular realization of $p(c \mid x) \sum_{i=2}^{N} 1 / i$ is sum $\sum_{i=2(c)}^{N} 1 / i$. Using this sum we can write

$$p(c \mid x) \lim_{N\to\infty} \sum_{i=2}^{N} 1 / i = \lim_{N\to\infty} \sum_{i=2(c)}^{N} 1 / i .$$

Dividing this equation by the limit of sum on the left hand side we get

$$p(c \mid x) = \frac{\lim\limits_{N\to\infty} \sum\limits_{i=2(c)}^{N} 1/i}{\lim\limits_{N\to\infty} \sum\limits_{i=2}^{N} 1/i}$$

and due to the same limit transition in numerator and in the denominator we can rewrite it in form (1). □

3.5. Generalization of the classifier

Here we generalize the classifier to cases of learning sets of different sizes for each class and for case of more than two classes. For different number of samples of one and the other class formula (1) has form

$$p(c \mid x) = \lim_{N\to\infty} \frac{\frac{1}{N_c} \sum\limits_{i=1(c)}^{N} 1/i}{\frac{1}{N_0} \sum\limits_{i=1(0)}^{N} 1/i + \frac{1}{N_1} \sum\limits_{i=1(1)}^{N} 1/i} \tag{2}$$

It is only recalculation of the relative representation of different numbers of samples of one and the other class. For C classes there is

$$p(c \mid x) = \lim_{N\to\infty} \frac{\frac{1}{N_c} \sum\limits_{i=2(c)}^{N} 1/r_i^q}{\sum\limits_{k=1}^{C} \frac{1}{N_k} \sum\limits_{i=2(c)}^{N} 1/r_i^q}.$$

It is interesting that formula (1) expresses Zipfian distribution (Zipf´s law) (Zipf, 1968) with Zipf´s exponent s = 1 (or eventually Zipf-Mandelbrot's law with zero additive parameter (Zipf, Mandelbrot, 2008)). It is easily seen that

$$\sum_{c=1}^{C} p(c \mid x) = \sum_{c=1}^{C} \lim_{N\to\infty} \frac{\sum\limits_{i=1(c)}^{N} 1/i}{\sum\limits_{i=1}^{N} 1/i} = 1$$

and p(c | x) is a "sum of relative frequencies of occurrence" of points of a given class c. A "relative frequencies of occurrence" of point i, i.e. of the i-th neighbor of query point x, is just

$$f(i;1,N) = \frac{1/i}{\sum\limits_{j=1}^{N} 1/j}.$$

In fact, f(i; s, N) is a probability mass function of Zipfian distribution. In our case p(c | x) is a sum of probability mass functions for all appearances of class c. We could discuss optimal value of Zipf´s exponent s, but as seen above s = 1 is just optimal value. In the context of our

findings this discrete distribution gets much broader role than its use in linguistics and psychology.

Example. Let us show a practical approach to construction a classifier that classifies to more than two classes and moreover it manages different numbers of patterns in the individual classes. In this example we use the well-know iris data by (Fischer, 1936). The task is to classify irises to three possible classes: Virginic, Versicolor and Setosa on the basis of their sepal and petal leaf width and length. There are totally 150 patterns (irises).

Let us chose one sample from the set as an unknown (test) pattern, say

Sepal Length	Sepal Width	Petal Length	Petal Width	Iris Type
5.9	3	5.1	1.8	Virginic

As we excluded one pattern from the available set of irises we have 149 (49, 50 and 50) patterns to our disposal for classifier construction.

The first step in our classifier construction is a normalization of the data (each individual feature is normalized independently) to zero mean and unit variance and consequently a normalization of the test pattern. Second, we calculate all (Euclidean) distances of the test pattern to all given patterns (149) and sort all the patterns in ascending order according to this distance. Further, a reciprocal value of order index is assigned to each pattern. In other words, 1 is assigned to the nearest pattern from our given pattern, ½ to the second nearest pattern and so on ... Finally, the 1/149 is assigned to the furthest pattern. As a further step we split the patterns with the assigned reciprocal indexes according to their class identifier and sum the particular values of the reciprocal indexes for the corresponding classes. We get the values

Virginic	Versicolor	Setosa
3.11377202	2.062620008	0.408121893

The sum of reciprocal values of indexes of all 149 patterns is 5.584513922. The ratios of these individual values to the number of patterns in the corresponding class is

Virginic	Versicolor	Setosa
3.11377202/49 = 0.063546368	2.062620008/50 = 0.0412524	0.408121893/50 = 0.008162438

After simple recalculation we finally get the probabilities in percentual representation

Virginic	Versicolor	Setosa
56.2550 %	36.5191 %	7.2259 %

On the basis of these results we can conclude that the given test pattern belongs to class 'Virginic' what has been assumed at the begining.

Partial cumulative sums for individual classes are depicted in Fig. 2. It is obvious that the lines do not overlap in this example. It means that it does not matter how many nearest neighbors will be used for the pattern classification (probability determination to which class the pattern belongs). The only difference would take effect in the different values of the probabilities of the individual classes not in their order.

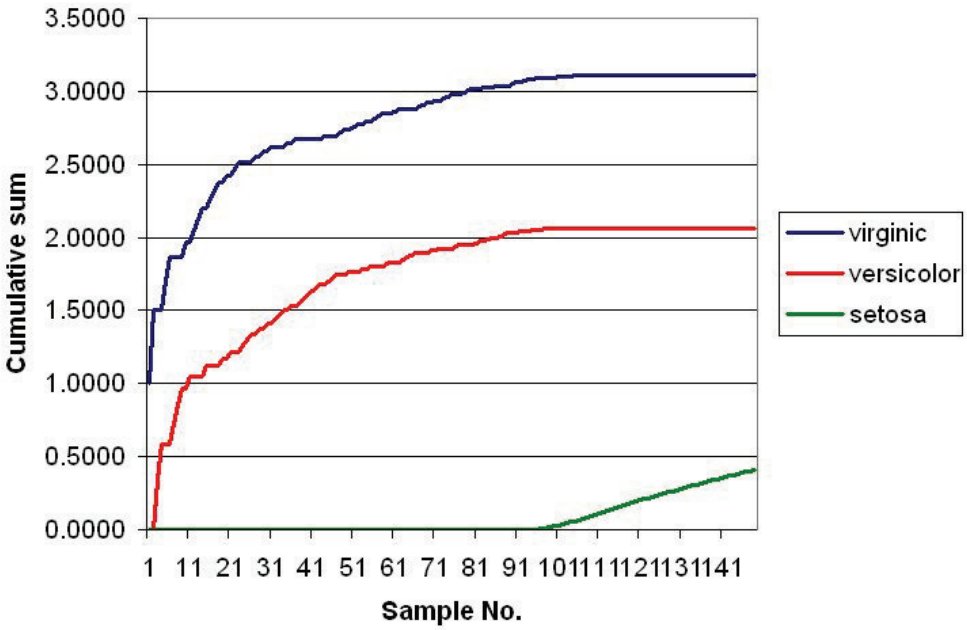


Fig. 2. Partial cumulative sums for individual classes

3.6 Measuring the distance

Usually distances are measured in Euclidean metric. On the experimental observation is seems that L_1 (absolute) metrics gives better results. At the same time, the larger p of L_p metric, the worse. The question arises why? We have no exact proof but point of view only, as follows.

Let us consider a metric written in a standard form

$$\lambda_i(a,b) = \sqrt[i]{\sum_{j=1}^{n} |b_i - a_i|^i} .$$

Let us formally rewrite this formula in form of scalar product using a vector which we will call weights

$$\lambda_i(a,b) = \sqrt[i]{(|b_1 - a_1|, |b_2 - a_2|, ..., |b_n - a_n|).(w_1, w_2, ..., w_n)} .$$

In our case, input arguments for the metric are coordinate differences $\delta_j = b_j - a_j$, j = 1, 2,..., n. Corresponding weight let be w_j. In Table 1 it can be seen that weights depend on the size of coordinate differences, and for L_1 metric only the weights are equal one to another. In other cases the larger the coordinate difference, the larger it's weight. There is also dependence on p of L_p and differences in the weights are the larger the larger p. The limit case is L_{max} metric.

Norm	Weights	distance $\sqrt[i]{\sum_{j=1}^{n}\lvert d_j\rvert w_j}$
L_1	$w_j = 1$	$\sqrt[1]{\sum_{j=1}^{n}\lvert d_j\rvert}$
L_2	$w_j = d_j$	$\sqrt[2]{\sum_{j=1}^{n} d_j^2}$
L_3	$w_j = d_{j2}$	$\sqrt[3]{\sum_{j=1}^{n}\lvert d_j^3\rvert}$
etc.	etc.	etc.
Lmax	$w_j = 1$ for maximal d_j $w_j = 0$ otherwise	$\max(\lvert d_j\rvert)$

Table 1. Metrics as Weighted Sum of Coordinate Differences.

It seems to hold that the only "fair" metric is L_1 as it gives to all coordinate differences the equal "chance" to influence the distances of neighbors and, in the end, their final relative positions and thus their ordering which influences the sums of reciprocals of neighbor's indexes for one and the other class.

3.7 Correspondence of the distribution mapping exponent to correlation dimension

It can be seen that for a fixed x the function D(x, r), r > 0 is monotonously non-decreasing from zero to one. Functions D(x, r) and d(x, r) for fixed x are one-dimensional analogs to the probability distribution function and the probability density function, respectively. In fact, D(x, r) is the distribution function of distances of points from the query point x and d(x, r) is the corresponding probability density function. So we can write p(c | x, r) = d(x, r, c). Moreover, D(x, r) resembles the correlation integral (Grassberger & Procaccia, 1983; Camastra & Vinciarelli, 2001). The correlation integral

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} h(r - |x_{.i} - x_{.j}|),$$

where $x_{.i}$ and $x_{.j}$ are points of the learning set without regard to class and h(.) is the Heaviside's step function, can be written in form (Camastra & Vinciarelli, 2001; Camastra, 2003)

$$C(r) = \lim_{N\to\infty} \frac{1}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} h(r - |x_{.i} - x_{.j}|).$$

It can be seen (Camastra & Vinciarelli, 2001; Camastra, 2003) that correlation integral is a distribution function of all binate distances among the given data points. The probability distribution mapping function is a distribution function of distances from one fixed point. In the case of a finite number of points N, there is N(N - 1)/2 binate distances and from them one can construct the empirical correlation integral. Similarly, for each point there are N - 1 distances and from these N - 1 distances one can construct the empirical probability

distribution mapping function. There are exactly N such functions and the mean of these functions gives the correlation integral. This applies also for the limit for the number of points N going to infinity.
On the other hand there are essential differences. The probability distribution mapping function is a local feature dependent on the position of point x. It also includes the boundary effects (Arya et al., 1996) of a true data set. The correlation integral is a feature of a fractal or data generated process and should not depend on the position of a particular point considered or on the size of the data set at hand.
In a log-log graph of the correlation integral, i.e. the graph of dependence of C on r, the slope gives the correlation dimension ν. In the log-log graph of the probability distribution mapping function D(x, r) the curve is also close to a monotonously and nearly linearly growing function. The slope (derivative) is given by a constant parameter. Let us denote this parameter q and call it the distribution mapping exponent. This parameter is rather close but generally different from ν.
The linear part of the log-log graph means

$$\log C(r) = a + \log \nu$$

where a is a constant, and then $C(r) = ar^{\nu}$. Thus C(r) grows linearly with variable r^{ν}.
Similarly the probability distribution mapping function grows linearly with r_q at least in the neighborhood of point x. Its derivative, the distribution density mapping function, is constant there. We will use this finding in the next section.

4. Demonstrations of the IINC on real-life tasks

4.1 Tasks from UCI machine learning repository

The classification ability of the IINC presented here was tested using real-life tasks from UCI Machine Learning Repository (Asuncion & Newman, 2007). Four tasks of classification into two classes for which data from previous tests were known were selected: "German", "Heart", "Adult", and "Ionosphere".
The task "German" decides whether a client is good or bad to be lent money to. In this data errors are weighted so that not to lend money to good a client means error weight 1, and lending money to a bad client means error weight 5.
The task "Heart" indicates the absence or presence of a heart disease in a patient.
The task "Adult" determines whether a person earns over $ 50000 a year.
For the task "Ionosphere" the targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not show this; their signals pass through the ionosphere.
We do not describe these tasks in detail here as all can be found in (Asuncion & Newman, 2007). For each task the same approach to testing and evaluation was used as described in (Asuncion & Newman, 2007). Especially splitting the data set into two disjoint subsets, the learning set and the testing set, and the use of cross validation were the same as in (Asuncion & Newman, 2007). For our method the discriminant threshold was tuned accordingly.
The testing should show the classification ability of IINC method for some tasks and also show its classification ability relatively to other published methods and results for the same data sets.
In Table 2 the results are shown together with the results of other methods as given in (Asuncion & Newman, 2007). For each task the methods were sorted according to the classification error, the method with the best – the smallest - error first.

"German"		**"Heart"**	
Algorithm	Error	Algorithm	Error
IINC	0.1580	**IINC**	0.1519
SVM	0.297	Bayes	0.374
Discrim	0.535	Discrim	0.393
LogDisc	0.538	LogDisc	0.396
Castle	0.583	Alloc80	0.407
Alloc80	0.584	SVM	0.411
Dipol92	0.599	QuaDisc	0.422
Smart	0.601	Castle	0.441
Cal	0.603	Cal5	0.444
Cart	0.613	Cart	0.452
QuaDisc	0.619	Cascade	0.467
KNN	0.694	KNN	0.478
Default	0.700	Smart	0.478
Bayes	0.703	Dipol92	0.507
IndCart	0.761	Itrule	0.515
Back Prop	0.772	Bay Tree	0.526
BayTree	0.778	Default	0.560
Cn2	0.856	BackProp	0.574
"Adult"		**"Ionosphere"**	
Algorithm	Error	Algorithm	Error
FSS Naive Bayes	0.1405	IB3	0.0330
NBTree	0.1410	**IINC**	0.0331
C4.5-auto	0.1446	backprop	0.0400
IDTM (Decision table)	0.1446	Ross Quinlan's C4	0.0600
HOODG	0.1482	nearest neighbor	0.0790
C4.5 rules	0.1494	"non-linear" perceptron	0.0800
OC1	0.1504	"linear" perceptron	0.0930
C4.5	0.1554	SVM	0.1400
Voted ID3 (0.6)	0.1564		
SVM	0.1590		
CN2	0.1600		
Naïve-Bayes	0.1612		
IINC	0.1617		
Voted ID3 (0.8)	0.1647		
T2	0.1684		
1R	0.1954		
Nearest-neighbor (4)	0.2035		
Nearest-neighbor (2)	0.2142		

Table 2. Comparison of the classification error of IINC method for different tasks with results of other classifiers as given in (Asuncion & Newman, 2007).

4.2 Comprehensive tests

Data sets ready for a run with a classifier were prepared by Paredes and Vidal and are available on the net (Lucas & Algoval, 2008). We used all data sets in this corpus. Each task consists of 50 pairs of training and testing sets corresponding to 50-fold cross validation. For DNA data (Paredes, 2008), Letter data (Letter recognition (Asuncion & Newman, 2007)), and Satimage (Statlog Landsat Satellite (Asuncion & Newman, 2007)) the single partition into training and testing sets according to specification in (Asuncion & Newman, 2007) was used. We also added the popular Iris data set (Asuncion & Newman, 2007) with ten-fold cross validation. In Table 3 the results obtained by different methods are summarized. The methods are as follows:

L_2	The nearest neighbor method, data by (Paredes & Vidal, 2006)
1-NN L_2	The nearest neighbor method computed by authors
sqrt-NN L_2	The k-NN method witt k equal to square root of the number of samples of the learning set computed by authors
Bayes 10	The Bayes naive method with ten bins histograms, computed by authors
CDM	The learning weighted metrics method with class dependent Mahalanobis, data by (Paredes & Vidal, 2006)
CW	The learning weighted metrics method with class dependent weighting by (Paredes & Vidal, 2006), data by (Paredes & Vidal, 2006)
PW	The learning weighted metrics method with prototype dependent weighting by (Paredes & Vidal, 2006), data by (Paredes & Vidal, 2006)
CPW	The learning weighted metrics method with class and prototype - dependent weighting by (Paredes & Vidal, 2006), data by (Paredes & Vidal, 2006)
posit. L_1	The learning weighted metrics method (Jirina & Jirina, 2008) with positions weighting and Manhattan L_1 metrics
posit. L_2	The learning weighted metrics method (Jirina & Jirina, 2008) with positions weighting and Euclidean L_2 metrics
diff. L_1	The learning weighted metrics method (Jirina & Jirina, 2008) with coordinate differences weighting and Manhattan L_1 metrics
diff. L_2	The learning weighted metrics method (Jirina & Jirina, 2008) with coordinate differences weighting and Euclidean L_2 metrics
IINC L_1	The method presented here with Manhattan L_1 metrics
IINC L_2	The method presented here with Euclidean L_2 metrics

In Table 3 in each row the best result is denoted by bold numerals. Furthermore, in the last column, the values for IINC better with L_2 metrics than with L_1 metrics are shown in italics. There are 6 such cases out of a total of 24.

Task\Method Dataset \	L_2	1-NN L_2	sqrt-NN L2	Bayes 10	SVM	CDM	CW	PW	CPW	posit. L_1	posit. L_2	diff. L_1	diff. L_2	**IINC L_1**	**IINC L_2**
Australian	34.37	20.73	15.50	13.88	35.99	18.19	17.37	16.95	16.83	17.64	19.00	17.86	21.51	**13.31**	14.75
Balance	25.26	23.61	32.06	15.17	45.48	35.15	17.98	13.44	17.6	17.85	**16.17**	34.48	37.74	32.58	*30.80*
Cancer	4.75	5.07	3.25	2.68	16..34	8.76	3.69	3.32	3.53	17.70	**3.18**	26.46	26.49	3.28	3.48
Diabetes	32.25	29.48	26.46	**24.19**	29.64	32.47	30.23	27.39	27.33	34.90	26.49	34.90	34.90	26.21	*25.52*
Dna	23.4	25.72	34.06	6.66		15	4.72	6.49	**4.21**	20.83	24.37	42.24	41.57	27.82	31.03
German	33.85	32.76	30.90	24.97	27.25	32.15	27.99	28.32	**27.29**	29.02	29.23	29.87	30.00	30.91	31.13
Glass	**27.23**	32.72	42.10	47.37		32.9	28.52	26.28	27.48	43.43	30.29	46.89	43.77	33.01	35.18
Heart	42.18	25.11	16.89	17.44	38.89	22.55	22.34	18.94	19.82	19.04	21.56	21.37	22.52	17.96	***17.93***
Ionosphere	*19.03*	14.05	14.70	**9.26**						29.39	17.58	29.70	30.03	10.82	14.81
Iris	*6.91*	5.91	7.91	9.82	6.55					**4.91**	6.91	25.82	11.82	7.91	***4.91***
Led17	*20.5*	11.50	**0.12**	0.00						7.64	2.67	24.78	37.72	0.46	*0.45*
Letter	4.35	4.80	18.70	28.98	40.53	6.3	3.15	4.6	**4.2**	6.23	5.90	7.95	8.05	4.85	4.98
Liver	37.7	39.59	41.48	39.42	37.68	39.32	40.22	**36.22**	36.95	40.96	42.00	40.70	40.43	38.29	39.13
Monkey1	*2.01*	2.01	9.27	28.01	23.54					2.01	2.82	**1.45**	1.47	4.79	4.79
Phoneme	*18.01*	11.83	20.71	21.47	21.71					14.72	**14.61**	29.27	29.27	17.55	18.06
Satimage	10.6	10.65	15.20	19.15	44.85	14.7	11.7	**8.8**	9.05	11.40	11.70	76.95	75.90	11.00	11.55
Segmen	*11.81*	3.81	11.41	9.85						5.18	5.35	9.96	10.62	**4.12**	5.05
Sonar	*31.4*	18.37	32.51	31.46						21.11	21.89	46.63	46.63	**19.89**	22.85
Vehicle	35.52	30.51	31.51	38.40		32.11	29.38	**29.31**	28.09	30.48	31.01	36.83	34.96	29.40	*29.34*
Vote	8.79	8.74	9.60	9.70		6.97	6.61	5.51	**5.26**	7.97	7.45	7.17	11.98	8.52	8.89
Vowel	1.52	**1.19**	46.68	26.64		1.67	1.36	1.68	1.24	3.52	3.89	5.55	6.17	2.73	2.74
Waveform 21	24.1	23.73	**14.71**	19.26						18.50	18.63	25.56	25.19	16.15	16.38
Waveform 40	31.66	28.22	**16.24**	20.31						20.50	22.61	32.25	32.78	17.59	18.08
Wine	24.14	5.42	6.15	4.50		2.6	1.44	1.35	**1.24**	5.27	6.06	72.04	67.42	4.24	5.66

Table 3. Classification error rates for different datasets and different approaches. Empty cells denote not available data. For legend see text above

5. Conclusion

The IINC seems to provide better classification then other classifiers in most tasks even though it is not the best all the time. This could make it a welcome alternative to standard classification methods.

The method of classification based on probability estimation proposed here consists in finding that each point of class c in the neighborhood of the query point x adds a little to the probability that point x is of class c, where c is the class mark. We proved that the influence to the probability that point x is of class c if the nearest neighbor of class c is 1, the influence of the second nearest neighbor is ½, the influence of the third nearest neighbor is 1/3 etc. We sum up these influences so that the sum goes over indexes i for which the corresponding samples of the learning set are of class c. In the case of two classes we get two numbers S_0 and S_1 which together give the sum of N first elements of harmonic series S = 1 + ½ + 1/3 + ¼ + … + 1/N. The estimation of the probability that the query point x belongs to class c is then $p(x \mid c) = \frac{S_c}{S}$.

The proof that ratio of sums mentioned gives just probability that the query point is of that class uses the notion of distance but no explicit metrics is specified. It was also found experimentally that it is usually better to measure distance by L_1 rather than standard L_2 metrics.

There is no problem with convergence of the method and the curse of dimensionality. The computational complexity grows at most linearly with dimensionality and quadratically or less with the learning set size depending on the sorting algorithm used.

6. Acknowledgements

This work was supported by the Ministry of Education of the Czech Republic under the project Center of Applied Cybernetics No. 1M0567, and No. MSM6840770012 Transdisciplinary Research in the Field of Biomedical Engineering II.

7. References

Arya, S., Mount, D. M., Narayan, O. (1996), Accounting for boundary effects in nearest neighbor searching, *Discrete and Computational Geometry*, Vol. 16, pp. 155-176.

Asuncion, A., Newman, D. J., (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science

Bock, R. K. (2004). Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research*, Vol. A 516, pp. 511-528

Camastra, F. (2003), Data dimensionality estimation methods: a survey. *Pattern recognition* Vol. 36, pp. 2945-2954.

Camastra, P., Vinciarelli, A. (2001), Intristic Dimension Estimation of Data: An Approach based on Grassberger-Procaccia's Algorithm. *Neural Processing Letters*, Vol. 14, No. 1, pp. 27-34.

Cháves, E., Figueroa, K., Navarro, G. (2001). A Fast Algorithm for the all k Nearest Neighbors Problem in General Metric Spaces. Escuela da Ciencias Fisicas y Matematicas, Universidad Michacana, Morelia, Michoacan, Mexico, 2001. Available:
http://garota.fismat.umich.mx/~elchavez/publica/.

Cover, T. M., Hart, P. E. (1967). Nearest neighbor Pattern Classification. *IEEE Transactions in Information Theory*, pp. 23-27, Vol. IT-13, No. 1

Duda, R. O., Hart, P. E., Stork, D. G. (2000). Pattern classification, Second Edition, John Wiley and Sons, Inc., New York.

Fisher, R.A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* , 7: 179–188.

Grassberger, P., Procaccia, I. (1983), Measuring the Strangeness of Strange Attractors. *Physica*, Vol. 9D, pp. 189-208.

Hakl, F., Hlaváček, M., Kalous, R. (2002). Application of Neural Networks Optimized by Genetic Algorithms to Higgs Boson Search. The 6th World Multi-Conference on Systemics, Cybernetics and Informatics. Proceedings. (Ed.: Callaos N., Margenstern M., Sanchez B.) Vol. : 11. Computer Science II. - Orlando, IIIS 2002, pp. 55-59

Hakl, F., Jirina, M., Richter-Was, E. (2005). Hadronic tau's identification using artificial neural network. *ATLAS Physics Communication*, ATL-COM-PHYS-2005-044, CERN, Geneve, Switzerland

Jiřina, M. and Jiřina, M., Jr. (2008). Classifier Based on Inverted Indexes of Neighbors II – Theory and Appendix, *Technical Report*, Institute of Computer Science AS CR, No. V-1041, Prague, Czech Republic

Jiřina, M., Jiřina, M., Jr. (2008). Learning Weighted Metrics Method with a Nonsmooth Learning Process. *Technical report,* V-1026, Institute of Computer Science AS CR, 15pp, Prague, Czech Republic

Kim, H. C., Ghahramani, Z. (2006). Bayesian Gaussian Process Classification with the EM-EP Algorithm. IEEE Trans. on pattern analysis and machine intelligence, pp. 1948-1959, Vol. 28, No. 12

Lucas, S. M., Algoval (2008). Algorithm Evaluation over the Web, [online], 2008, [cited November 23, 2008]. Available: <http://algoval.essex.ac.uk/data/vector/UCI/>

Merz, C. J., Murphy, P. M., Aha, D. W. (2010). UCI Repository of Machine Learning Databases. Dept. of Information and Computer Science, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn/MLSummary.html

Paredes, R. (2008). CPW: Class and Prototype Weights learning, [online], 2008, [cited November 23, 2008]. Available:
<http://www.dsic.upv.es/~rparedes/research/CPW/index.html>

Paredes, R., Vidal, E. (2006). Learning Weighted Metrics to Minimize Nearest Neighbor Classification Error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1100-1110, Vol. 20, No. 7

Silverman, B. W. (1990). Density estimation for statistics and data analysis, Chapman and Hall, London.

Zipf, G. K. (1968), The Psycho-Biology of Language. An Introduction to Dynamic Philology. The MIT Press. (Eventually: http://en.wikipedia.org/wiki/Zipf's_law)
Zipf–Mandelbrot law, [online], 2009, [cited January 28, 2009]. Available: http://en.wikipedia.org/wiki/Zipf-Mandelbrot_law