Pattern Classification Using a Penalized Likelihood Method

Ahmed Al-Ani[1] and Amir F. Atiya[2]

[1] Faculty of Engineering and Information Technology, Univesity of Technology, Sydney, Australia
ahmed@eng.uts.edu.au
[2] Department of Computer Engineering, Cairo University, Giza, Egypt
amir@alumni.caltech.edu

Abstract. Penalized likelihood is a well-known theoretically justified approach that has recently attracted attention by the machine learning society. The objective function of the Penalized likelihood consists of the log likelihood of the data minus some term penalizing non-smooth solutions. Subsequently, maximizing this objective function would lead to some sort of trade-off between the faithfulness and the smoothness of the fit. There has been a lot of research to utilize penalized likelihood in regression, however, it is still to be thoroughly investigated in the pattern classification domain. We propose to use a penalty term based on the K-nearest neighbors and an iterative approach to estimate the posterior probabilities. In addition, instead of fixing the value of K for all pattern, we developed a variable K approach, where the number of neighbors can vary from one sample to another. The chosen value of K for a given testing sample is influenced by the K values of its surrounding training samples as well as the most successful K value of all training samples. Comparison with a number of well-known classification methods proved the potential of the proposed method.

1 Introduction

The basic concept behind penalized likelihood is that a good model should possesses two indispensable properties: the goodness of fit and the smoothness of the fit [1], [2]. However, as these two are primarily conflicting goals, a trade-off that suits the given application is pursued. The penalized likelihood approach seeks to achieve that trade-off by defining an overall objective function consisting of the log-likelihood of the data minus a roughness measure, and subsequently maximizing this objective function. The likelihood function is a measure of the faithfulness of the fit, while the roughness function is a penalty term that penalizes non-smooth solutions.

An example of the roughness function is the integral of the square of the second derivative of the function, leading to the following objective function (see [3]):

$$T = \log \text{ likelihood } - \lambda \int f''^2(x)dx \tag{1}$$

One example of a penalized likelihood regression is the well-known regression spline model [4]. Most of the penalized regression work focused on finding a complete functional formulation and the optimization is performed mostly in the Hilbert space [5].

For the classification problem the underlying function would be the class posterior probabilities. These are the functions which we attempt to estimate and for which we impose smoothness. Among the works considering penalized likelihood classification is the work of O'Sullivan et al [6], which was subsequently analyzed and extended in many other studies [7], [8], [5], [9], [10]. The basic idea of these approaches is to assume that the class posterior probability (considering a two-class case with classes C_1 and C_2) is modeled as a logit function applied to some (unrestricted) function. This is a mean to enforce the $[0, 1]$ bound on the posterior probability. In some of these works thin-plate spline is used as smoothness penalty, and in some others general smoothness penalties are used with the help of the theory of reproducing kernel hilbert spaces. The problem could be solved through a parametric representation, whose parameters are obtained through Newton-Raphson iteration. A related approach is to consider the logistic regression problem (which is essentially a two-class classification problem) in the framework of penalized likelihood regression (see [11] and see also the generalization to the multinomial logistic regression case in [12]), or the generalized additive model [13] (which also tackles in some way the penalized logistic regression problem).

A different methodology based on a Bayesian paradigm is the Gaussian process classification (GPC) approach [19]. While it does not have a penalized likelihood element in it, it enforces smoothness by defining a Bayesian prior that assigns a higher probability to smooth solutions. Again, imposing a logit function lead to intractable integrals that can only be approximated. Another related approach [15] uses the K-nearest neighbor class memberships in some way to describe the priors. It is a Bayesian approach, with the key parameters being attached some priors and these are then integrated out. Again, the integral is intractable and MCMC is proposed as a way to evaluate it.

In this paper we propose a new penalized likelihood classification method for the two-class case. Rather than insisting on evaluating the posterior probability as a functional form (which makes it generally quite difficult), we evaluate it only for the points we need, that is for the training and the testing points. We use as a measure of roughness the sum of square difference between the posterior of a point and that of its K nearest neighbors. We therefore managed to avoid the use of the logit function, which in all above works was an obstacle to obtaining straightforward analytic solutions. We propose an iterative algorithm that converges to the maximum of the penalized likelihood function in few iterations. While we make use of some kind of pattern distance matrix like in the case of Gaussian process classification, the philosophy and the approach is quite different.

We tested the proposed method on a number of UCI benchmark datasets. As it turns out, it produces a classification performance beating many of the well-known methods (such as SVM and several other methods) and comparable to

GPC (it is generally believed that SVM and GPC are among the best two classification approaches [16]). On the other hand the computation time was much less than that of GPC. Another advantage of the method is that it is entirely based on distances between the training patterns (like the K nearest neighbor classifier and the GPC). So it can handle also non-numeric inputs, for example text inputs whereby some distance function can be defined. The proposed method is also very simple, consisting of only a simple iteration, and requiring little development time to implement it and no sophisticated optimization routines.

The paper is organized as follows. The proposed method is presented in the next section. The following section details the classification algorithm. In Section 5 we present the simulations results, followed by the conclusions section.

2 The Proposed Method

Let $x_m \in \mathcal{R}^L$ denote the feature vectors, with $x_1, \ldots, x_M$ denoting the training patterns, and $x_{M+1}, \ldots, x_{M+N}$ denoting the test patterns. In this work we consider only the two-class case. The class membership y_{gm} for class label g and training pattern x_m is defined as follows: it equals 1 if $x_m \in C_g$ and equals 0 otherwise, where $g \in \{1, 2\}$.

Let $P_{gm} \equiv P(C_g|x_m)$ denote the posterior probability for class C_g, and $\sum_{g=1}^{2} P_{gm} = 1$. The purpose of the proposed method is to estimate the posterior probabilities P_{gm}, both for the training set and the test set. Knowing the posterior probabilities will automatically determine the classification of the patterns. As we will shortly see, the posterior probabilities are obtained by defining the penalized likelihood function and subsequently maximizing it, leading to an iterative algorithm.

The likelihood of the data is given by

$$L = \prod_{m=1}^{M} \prod_{g=1}^{2} P_{gm}^{y_{gm}} \tag{2}$$

Denote by $\mathcal{K}(x_m)$ as the set of K_m-nearest neighbors of point x_m (their indexes), and K_m the size of $\mathcal{K}(x_m)$, which can vary from one pattern to another. We define a roughness function based on the square differences of the posteriors of neighboring data points. Specifically, it is given by

$$R = \sum_{m=1}^{M} \frac{1}{K_m} \sum_{m' \in \mathcal{K}(x_m)} (P_{hm} - P_{hm'})^2 \tag{3}$$

where h is the class that x_m belongs to (either 1 or 2), hence, $y_{hm} = 1$. Note that Eq. 2 can be written as $L = \prod_{m=1}^{M} P_{hm}$. We define our overall objective function as a combination of the log-likelihood function and the roughness function:

$$J = \log(L) - \lambda R \tag{4}$$

$$= \sum_{m=1}^{M} \log(P_{hm}) - \lambda \sum_{m=1}^{M} \frac{1}{K_m} \sum_{m' \in \mathcal{K}(x_m)} (P_{hm} - P_{hm'})^2 \tag{5}$$

The first term in the penalized log-likelihood J focuses on the goodness of fit aspect. It gauges how well that the considered P_{hm}'s fit the observed data (i.e. the given class memberships). The second term serves to penalize the roughness of the underlying posterior function. A posterior surface where its values for neighboring points are close (i.e. having low R) will generally be smooth, and conversely a high R is indicative of a rough or wiggly surface. The goal is to find the posterior probabilities that maximize the penalized log-likelihood J. We will therefore achieve a compromise between faithfully respecting the class memberships of the training data and the smoothness property of the posterior surface, with λ being the parameter that controls the degree of smoothness.

3 The Proposed Algorithm

The goal is to solve the following maximization problem:

Maximize J (given by (5)) w.r.t. the variables: P_{gm}, s.t. $0 \leq P_{gm} \leq 1, m = 1, \ldots, M, g = \{1, 2\}$.

It is easy to see that J is a convex function w.r.t. the P_{gm}'s. Hence the problem has a unique maximum. The algorithm proposed below is based on cycling through all variables, each time optimizing w.r.t. only one of the variables (through a line search). In each step, the optimum w.r.t. one variable can be obtained analytically, as show below.

$$\frac{\partial J}{\partial P_{hm}} = \frac{1}{P_{hm}} - \frac{2\lambda}{K_m}\left[\sum_{m' \in \mathcal{K}(x_m)} (P_{hm} - P_{hm'}) + \sum_{m' \in \mathcal{S}(x_m)} (P_{hm} - P_{hm'})\right] \tag{6}$$

$$= \frac{1}{P_{hm}} - \frac{2\lambda(K_m + S_m)}{K_m} P_{hm} + \frac{2\lambda}{K_m}\left[\sum_{m' \in \mathcal{K}(x_m)} P_{hm'} + \sum_{m' \in \mathcal{S}(x_m)} P_{hm'}\right] \tag{7}$$

where $\mathcal{S}(x_m)$ is the set of patterns for which x_m is one of the neighbors, S_m is the length of $\mathcal{S}(x_m)$. To find the value of P_{hm} that maximizes J, we need to make the right hand side of Eq. 7 equals 0, which would lead to

$$1 - \left(\frac{2\lambda(K_m + S_m)}{K_m}\right) P_{hm}^2 + \left(\frac{2\lambda}{K_m}\left[\sum_{m' \in \mathcal{K}(x_m)} P_{hm'} + \sum_{m' \in \mathcal{S}(x_m)} P_{hm'}\right]\right) P_{hm} = 0 \tag{8}$$

$$P_{hm}^2 - \left(\frac{1}{K_m + S_m}\left[\sum_{m' \in \mathcal{K}(x_m)} P_{hm'} + \sum_{m' \in \mathcal{S}(x_m)} P_{hm'}\right]\right) P_{hm} - \frac{K_m}{2\lambda(K_m + S_m)} = 0 \tag{9}$$

Eq. 9 is a quadratic equation that can easily be solved. The algorithm of the proposed method is given below.

1. Start with any initial choice e.g. $P_{gm} = 0.5, \quad m = 1, \ldots, M, g =\in \{1, 2\}$.
2. While the change in the posteriors between the current and previous iteration is greater than a certain threshold ($Thresh$), execute step 3.
3. For each training pattern, $m = 1$ to M:
   (a) Set:

$$P_{hm} \equiv \frac{1}{2}\bar{P}_{hm} + \frac{1}{2}\sqrt{(\bar{P}_{hm})^2 + \frac{2K_m}{\lambda(K_m + S_m)}} \tag{10}$$

   where

$$\bar{P}_{hm} = \frac{1}{K_m + S_m}\Big[\sum_{m' \in \mathcal{K}(x_m)} P_{hm'} + \sum_{m' \in \mathcal{S}(x_m)} P_{hm'}\Big] \tag{11}$$

   where K_m is the number of nearest neighbors, $\mathcal{S}(x_m)$ is the set of data points for which x_m is one of the nearest neighbors, and S_m is the size of set $\mathcal{S}(x_m)$. Thus $\bar{P}_{hm}$ is the mean of the values of $P_{hm'}$ for some sort of neighborhood of points around x_m.

   (b) Truncate if P_{hm} goes out of the constraint box, set:

$$P_{hm} = 1 \quad \text{if } P_{hm} > 1, \quad \text{or} \quad P_{hm} = 0 \quad \text{if } P_{hm} < 0 \tag{12}$$

   (c) Let $f \neq h, f \in \{1, 2\}$, set:

$$P_{fm} = 1 - P_{hm} \tag{13}$$

   (d) Let the set of possible K-nearest neighbor values $\mathcal{KK} = \{k_1, k_2, \ldots, k_n\}$, calculate the error associated with using k_j-nearest neighbors, $j = 1 : n$, and set K_m to the value that minimizes the error, E_{mj}, as follows

$$\begin{aligned} E_{mj} &= 1 - \text{mean}(P_{h\mathcal{KK}_j(x_m)}) \\ K_m &= \arg\min_j \,(E_{mj}) \end{aligned} \tag{14}$$

   where $\mathcal{KK}_j(x_m)$ is the set of k_j-nearest neighbors of x_m. One possible choice is to use $\mathcal{KK} = \{3, 5, 7, \ldots, 25\}$.
4. For each test pattern, $m = M + 1$ to $M + N$:
   (a) Find the value of K_m:

$$K_m = \begin{cases} \arg\min_j \,(\text{mean}(E_{\mathcal{Q}_m j})) \text{ if } \min\,(\text{mean}(E_{\mathcal{Q}_m j})) < T_E \\ \arg\min_j \,(\text{mean}(E_{\mathcal{M}j})) \;\; \text{otherwise} \end{cases} \tag{15}$$

   where $\mathcal{Q}_m$ represents the set of training patterns that surrounds x_m (local neighborhood), while $\mathcal{M}$ represents the set of all training patterns. T_E is a threshold, which can be the mean of $E_{\mathcal{M}j}$.

   (b) Calculate the posterior probabilities $P_{gm}, g \in \{1, 2\}$:

$$P_{gm} = \text{mean}(P_{g\mathcal{K}(x_m)}) \tag{16}$$

This algorithm performs an iterated estimation of the posteriors through Eq. 10, which is basically the closed-form outcome of the one-variable search that is performed by cycling through all variables. Eq. 14 shows how the value of K_m can vary from one pattern to another. This can lead to a higher value of J when compared to using a fixed K for all patterns. The iterations should carry on until the change in the posteriors from one cycle till the next is small. Once the algorithm converges, we use the obtained final values of the P_{gm}'s as the estimated posteriors of training data points, which will be used to estimate the posteriors of the test data points. The rationale behind Eq. 15 is to first check the error of the local neighborhood. If the error is small, then the neighborhood would influence the choice of K_m. Otherwise, use the value of K that, on average, is most reliable over all training samples. Recalling that $P_{gm} \equiv P(C_g|x_m)$, then the final classification of a test data point is estimated as class C_1 if $P_{1m} > P_{2m}$, otherwise it is class C_2.

4 Simulation Results

A number of benchmark datasets were used to test the performance of the proposed method. We have compared the performance of the proposed method to that of the following well-known classification methods:

– Bayes classifier ([17], p. 168) with the class-conditional densities estimated according to the Parzen window density estimator (PARZEN) [18]. A key parameter for the Parzen estimator is the width of kernels h. We used the value derived in [18] (Silverman's rule):

$$h = \hat{\sigma} \left[\frac{4}{(2L+1)I} \right]^{\frac{1}{L+4}} \tag{17}$$

where $\hat{\sigma}^2 \equiv \sum_{i=1}^{L} S_{ii}/L$ denotes the mean of the diagonal of the sample covariance matrix S, L is the dimension of the space, and I is the number of data points (we used Gaussian kernels).

– Gaussian process classification using the expectation propagation approximation [19]. We used the non-optimized (GPC) and optimized (GPCo) versions. The latter attempts to approximate the integrals in the Gaussian process classification formula. We used the software available in [14].

– Support vector machines (SVM) (Scholkopf and Smola [20]). We used a radial basis function SVM implemented using the OSUsvm toolbox[1]. The values of C and γ for the latter are set using a K-fold validation procedure (we used five-fold validation and allowed C and γ to range between [0.5, 1.5]).

– K-nearest neighbor classifier. The value of K was set using a five-fold validation process (only odd numbers that range between 3 and 25 were considered).

– Evidential K-nearest neighbor (KNNds). This algorithm is based on the Dempster-Shafer theory of evidence taking into account the distance and class label information of the neighbors for generating soft decision vectors [21][2].

[1] Obtained from http://downloads.sourceforge.net/svm/osu-svm-3.0.zip

[2] The KNNds software is available at http://www.hds.utc.fr/ tdenoeux/software.htm

We tested all these competing methods on real-world pattern classification problems, mostly from the UCI repository [22]. We also tested those algorithms on the well-known two-spiral classification problem. This dataset consists of points on two inter-wined spirals that cannot be linearly separated, as shown in Fig. 1. Table 1 summarizes the characteristics of the datasets used in this paper.

Patterns that consist of missing values were removed from the datasets. In certain cases, attributes that consist of many missing values were excluded to minimize the number of removed patterns. Categorical attributes were changed to attributes with integer values to enable the chosen algorithms to handle them. For all considered problems the input attributes are first scaled so that they lie in a suitable range. We used 80% of the data as a training set, and the remaining 20% as a test set. We performed 20 runs for each method, each run with a different random train/test partition. Then we average the classification accuracies on the test sets of the 20 runs.

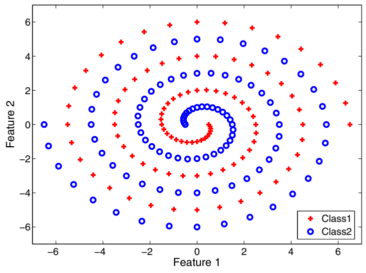

Fig. 1. Two-spiral dataset

Table 1. Datasets used to evaluate the performance of classifiers

Dataset	# Attributes	# Patterns	Class distribution
Two Spiral	2	194	0.50/0.50
Ger. Credit	24	1000	0.70/0.30
Cylinder bands	30	350	0.62/0.38
Blood transfusion	4	748	0.24/0.76
cancer	9	683	0.65/0.35
Haberman's survival	3	306	0.73/0.27
heart	22	267	0.79/0.21
heart SPECT	13	270	0.55/0.45
hill-valley	100	606	0.51/0.49
ionosphere	33	351	0.64/0.36
mammographic	5	814	0.48/0.52
monk	6	432	0.50/0.50
Parkinson	22	195	0.75/0.25
pima	8	768	0.35/0.65
sonar	60	208	0.53/0.47
Tic-tac	9	958	0.65/0.35
wdbc	30	569	0.63/0.37

In order to compare the performance of the various algorithms mentioned above, we used the following measures:

– **Mean classification accuracy (Acc).** This measure gives a general indication about the performance of each classifier.
– **Estimated standard deviation of the accuracy.** It is calculated by dividing the standard deviation of Acc by the square root of the number of runs.
– **Significance test.** A two-tailed paired t-test is performed with significance level of $\alpha = 0.05$. This indicates if there is a significant difference in the performance of two classifiers.
– **Geometric mean error ratio.** For the two classifiers that have errors $a_1, a_2 \ldots, a_n$ and $b_1, b_2 \ldots, b_n$ respectively (n represents the number of runs), the geometric error ratio is:

$$\exp \frac{\sum_{i=1}^{n} \log(ai/bi)}{n} = \sqrt[n]{\prod_{i=1}^{n} ai/bi} \tag{18}$$

This measure reflects the relative performance of one classifier with respect to another. If the outcome is less than 1, then it is an indication that the first classifier outperforms the second classifier in terms of error reduction.
– **Win-Tie-Loss.** This is an important measure, where the three values are the number of datasets for which classifier a obtained better, equal, or worse performance outcomes than classifier b.
– **Sign test.** The p-values of a two-tailed sign test based on the win-tie-loss record. if p is significantly low, then one can conclude that it is unlikely that the outcome was obtained by chance, i.e., the difference between the two classifiers is significant. On the other hand, a higher p value indicates that the two classifiers are not significantly different.

For detailed description of these measures the reader is referred to [23], [24].

The PLC algorithm described in section 3 only needed few iterations to converge for all of the 17 datasets when we set *Thresh* to 0.01. Table 2 shows the average classification accuracy of the competing methods with the estimated standard deviation. It also shows if (PLC) is significantly different from other classifiers from a statistical viewpoint. For a given dataset, if PLC is significantly better than a certain classifier, then a bullet is displayed next to that classifier's result. On the other hand, an open circle indicates that the classifier is significantly better than PLC. A quick glance at the table would show that there are more bullets than open circles. PLC is found to be particularly better than Parzen, GMM, KNNds and KNN. However, the results indicate that PLC is not significantly better than the Gaussian process and SVM, particularly GPCo. As mentioned earlier, these two classifiers are considered in the literature to be among the best classification approaches.

In order to present a more detailed analysis of the classification results, Table 3 presents other comparison measures. The first row of the table represents the

Table 2. Classification accuracy and estimated standard deviation for the considered classifiers

	Parzen	GPC	GPCo	SVM
2Spiral	33.16 ± 1.31●	47.63 ± 1.51●	50.39 ± 0.29●	47.63 ± 1.61 ●
GerCr	68.75 ± 0.65●	76.78 ± 0.57○	77.50 ± 0.58○	75.90 ± 0.46○
bands	68.57 ± 0.84	74.00 ± 1.41○	75.00 ± 1.29○	73.14 ± 1.22○
Btrans	76.97 ± 0.39	76.73 ± 0.36	77.7 ± 0.41	75.97 ± 0.15●
canc	95.84 ± 0.36●	96.82 ± 0.23	96.57 ± 0.32	96.57 ± 0.23
haber	71.48 ± 0.59●	72.87 ± 0.70	73.28 ± 0.77	72.54 ± 0.50
heart	80.19 ± 1.31●	83.96 ± 0.86○	84.15 ± 1.00	83.68 ± 0.95
heartS	80.19 ± 0.87	83.24 ± 0.83○	83.70 ± 1.00○	83.06 ± 0.89○
hill	53.51 ± 0.74●	50.45 ± 0.58●	51.78 ± 0.51●	49.96 ± 0.57●
ion	88.21 ± 0.8○	89.57 ± 0.65○	96.07 ± 0.52○	94.29 ± 0.55○
mamm	79.20 ± 0.67	80.68 ± 0.58○	82.01 ± 0.56○	80.19 ± 0.62
monk	76.86 ± 0.95●	80.99 ± 1.00	94.65 ± 0.82○	91.10 ± 0.49○
parkin	95.00 ± 0.71	83.72 ± 1.10●	93.08 ± 1.10	87.82 ± 1.08●
pima	73.21 ± 0.50	76.33 ± 0.61○	76.79 ± 0.61○	76.66 ± 0.57○
sonar	87.32 ± 1.14	84.63 ± 1.30●	83.66 ± 1.41●	86.34 ± 1.33
tic-tac	87.49 ± 0.38	83.48 ± 0.70●	96.18 ± 0.36○	95.71 ± 0.29○
wdbc	97.52 ± 0.31	97.65 ± 0.31	97.26 ± 0.28	97.92 ± 0.27○

	KNN	KNNds	PLC
2Spiral	75.13 ± 2.10●	75.13 ± 2.10●	79.61 ± 1.09
GerCr	72.48 ± 0.34●	72.15 ± 0.74●	73.88 ± 0.40
bands	68.36 ± 1.30	67.57 ± 0.99	68.93 ± 1.14
Btrans	77.90 ± 0.45	73.27 ± 0.72●	77.5 ± 0.58
canc	96.64 ± 0.31	96.61 ± 0.27	96.68 ± 0.29
haber	73.11 ± 0.68	70.33 ± 1.01●	72.87 ± 0.69
heart	81.79 ± 1.17	80.28 ± 1.43	82.36 ± 1.13
heartS	81.76 ± 1.03	77.78 ± 1.07●	81.48 ± 0.92
hill	54.38 ± 0.77●	55.95 ± 1.12●	59.09 ± 1.00
ion	84.71 ± 0.69●	89.86 ± 0.74○	86.36 ± 0.68
mamm	79.26 ± 0.70	77.10 ± 0.47●	79.66 ± 0.62
monk	83.72 ± 0.87	83.95 ± 0.80○	82.50 ± 0.87
parkin	92.56 ± 1.03	92.69 ± 1.04	93.85 ± 1.08
pima	73.64 ± 0.67	73.99 ± 0.47	74.09 ± 0.58
sonar	83.66 ± 1.05●	83.78 ± 1.01●	87.56 ± 1.20
tic-tac	88.04 ± 0.48○	82.64 ± 0.48●	86.86 ± 0.57
wdbc	96.81 ± 0.31	96.99 ± 0.30	97.17 ± 0.26

mean accuracy across all the datasets. According to this measure PLC is found to be the second best classifier, after GPCo, outperforming all remaining classifiers, including SVM. The table also presents pair-wise comparisons between the classifiers according to their geometric error ratio ($\dot{r}$), and the win-tie-loss (s). Also shown is the p-value of the sign test for the win-tie-loss (p). According to these measures, PLC outperformed Parzen, KNN, and KNNds. In fact the geometric error ratio indicates that PLC is slightly better than GPC and not too different from SVM. On the other hand, the win-tie-loss favors the Gaussian

Table 3. Comparison of averaged classification accuracy, geometric error, win-tie-loss, and p-value of the sign test across all the used datasets

	Parzen	GPC	GPCo	SVM	KNN	KNNds	PLC
Mean Acc.	77.26	78.80	81.75	80.50	80.23	79.42	81.20
Parzen							
$\dot{r}$		1.016	1.333	1.212	1.052	1.020	1.112
s		5-0-12	4-0-13	4-0-13	5-0-12	9-0-8	4-0-13
p		0.1435	0.049	0.049	0.1435	1.000	0.049
GPC							
$\dot{r}$			1.311	1.193	1.035	1.003	1.094
s			3-0-14	9-1-7	10-0-7	12-0-5	9-1-7
p			0.0127	0.8036	0.6291	0.1435	0.8036
GPCo							
$\dot{r}$				0.910	0.789	0.765	0.834
s				14-0-3	12-1-4	13-0-4	12-0-5
p				0.0127	0.0768	0.049	0.1435
SVM							
$\dot{r}$					0.868	0.841	0.917
s					11-0-6	13-0-4	10-0-7
p					0.3323	0.049	0.6291
KNN							
$\dot{r}$						0.969	1.057
s						9-1-7	5-0-12
p						0.8036	0.1435
KNNds							
$\dot{r}$							1.090
s							2-0-15
p							0.0023

Table 4. Execution Time for GPCo and PLC, measured in CPU time (sec). This time includes training time and testing time.

	2Spiral	GerCred	bands	Btrans	cancer	haber	heart	heartS	hill
GPCo	1.63	882.85	22.77	360.82	314.43	11.75	9.58	7.63	82.67
PLC	0.19	2.73	0.56	1.40	1.43	0.36	0.35	0.31	6.06

	ion	mamm	monk	parkin	pima	sonar	tic-tac	wdbc
GPCo	59.81	530.17	106.57	9.31	469.99	10.39	1229.33	238.80
PLC	1.09	3.73	1.25	0.38	3.67	0.62	5.55	2.80

process and support vector machine over PLC. However, as seen from the p-value measure, only GPCo is significantly better than PLC.

The above results indicate that GPCo is the only classifier that is significantly better from a statistical point of view. So, it would be important to compare these two classifiers in terms of computational complexity. Table 4 shows the computation time of both GPCo and PLC for all considered datasets. The table indicates that PLC is considerably faster than GPCo, which represents a great advantage for the proposed algorithm.

5 Conclusion

A new classification method based on penalized likelihood concept is presented in this paper. The method is based on defining a roughness term based on the K-nearest neighbors. We have developed an algorithm that converges to the global optimum in only few iterations. We have also proposed to allow the value of K to vary from one pattern to another, which proved to be useful in maximizing the objective function. When compared with several well-known classification methods, the proposed classifier achieved a performance competitive with the top models, but with less computational time. As such, the proposed approach can be ranked among the top binary classification algorithms.

References

1. Green, P.: Penalized likelihood. In: Encyclopedia of Statistical Sciences, Update vol. 3 (1999)
2. Gu, C., Kim, Y.-J.: Penalized likelihood regression: general formulation and efficient approximation. Canadian Journal of Statistics 29 (2002)
3. Green, P.J., Silverman, B.W.: Nonparametric Regression and Generalized Linear Models: a Roughness Penalty Approach. Chapman and Hall, London (1994)
4. Berry, S.M., Carroll, R.J., Ruppert, D.: Bayesian smoothing and regression splines for measurement error problems. J. Amer. Statist. Assoc. 97, 160–169 (2002)
5. Wahba, G.: Spline Models for Observational Data. SIAM, Philadelphia (1990)
6. OSullivan, F., Yandell, B., Raynor, W.: Automatic smoothing of regression functions in generalized linear models. J. Amer. Statist, Assoc. 81, 96–103 (1986)
7. Gu, C.: Cross-validating non-gaussian data. J. Comput. Graph. Statist. 1, 169–179 (1992)
8. Lu, F., Hill, G.C., Wahba, G., Desiati, P.: Signal probability estimation with penalized likelihood method on weighted data, Technical Report, No. 1106. Department of Statistics, University of Wisconsin (2005)
9. Wahba, G.: Soft and hard classification by reproducing kernel hilbert space methods. Proc. Nat. Acad. Sciences 99, 16524–16530 (2002)
10. Wahba, G., Gu, C., Wang, Y., Chappell, R.: Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance, Technical Report, No. 899. Department of Statistics, University of Wisconsin (1993)
11. Loader, C.: Local Regression and Likelihood. Springer, Heidelberg (1999)
12. Cawley, G., Talbot, N.L., Girolami, M.: Sparse multinomial logistic regression via bayesian l1 regularisation. In: Proceedings NIPS, pp. 209–216 (2007)
13. Hastie, T., Tibshirani, R.: Generalized Additive Models. Chapman and Hall, Boca Raton (1990)
14. Rasmussen, C.E. (2007), `http://www.GaussianProcess.org/gpml/code/index.html`
15. Holmes, C.C., Adams, N.M.: A probabilistic nearest neighbour method for statistical pattern recognition. Journal Royal Statistical Society B 64, 295–306 (2002)

16. Jensen, R., Erdogmus, D., Principe, J.C., Eltoft, T.: The laplacian classifier. IEEE Trans. Signal Processing 55, 3262–3271 (2007)
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, Hoboken (2000)
18. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, Boca Raton (1986)
19. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (2005)
20. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
21. Zouhal, L., Denoeux, T.: An evidence-theoretic k-nn rule with parameter optimization. IEEE Trans. Syst. Man Cyber. 28, 263–271 (1998)
22. Asuncion, D.J.: UCI Machine Learning Repository (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
23. Zhang, C.-X., Zhang, J.-S.: Rotboost: a technique for combining roataion forest and adaboost. Pattern Recognition Letters 29, 1524–1536 (2008)
24. Webb, G.: Multiboosting: a technique for combining boosting and wagging. Machine Learning 40, 159–196 (2000)