A Penalized Likelihood based Pattern Classification Algorithm

Amir F. Atiya
Department of Computer Engineering
Cairo University
Giza, Egypt
amir@alumni.caltech.edu

Ahmed Al-Ani
Faculty of Engineering and Information Technology
University of Technology, Sydney
Australia
ahmed@eng.uts.edu.au

February 2, 2009

Abstract

Penalized likelihood is a general approach whereby an objective function is defined, consisting of the log likelihood of the data minus some term penalizing non-smooth solutions. Subsequently, this objective function is maximized, yielding a solution that achieves some sort of trade-off between the faithfulness and the smoothness of the fit. Most work on that topic focused on the regression problem, and there has been little work on the classification problem. In this paper we propose a new classification method using the concept of penalized likelihood (for the two class case). By proposing a novel penalty term based on the K-nearest neighbors, simple analytical derivations have led to an algorithm that is proved to converge to the global optimum. Moreover, this algorithm is very simple to implement and converges typically in two or three iterations. We also introduced two variants of the method by distance-weighting the K-nearest neighbor contributions, and by tackling the unbalanced class patterns situation. We performed extensive simulations experiments to compare the proposed method to several well-known classification methods. These simulations reveal that the proposed method achieves one of the top ranks in classification performance and with much smaller computation time than the other higher ranked methods.

1 Introduction

Penalized likelihood is a well-known nonlinear regression model based on the premise that a good model should possesses two indispensable properties: the goodness of fit and the smoothness of the

fit (Green [10], and Gu and Kim [13]). However, these two are primarily conflicting goals, and usually 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 Green and Silverman [11]):

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

One example of a penalized likelihood regression is the well-known regression spline model (Berry et al [5]). Most of the penalized regression work focused on finding a complete functional formulation and the optimization is performed mostly in the Hilbert space (see Wahba [28]).

In contrast to the regression framework, there is little work on extending it to the classification domain. For the classification problem the underlying function would then 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 [19], which was subsequently analyzed and extended in many other studies (see Gu [12], Lu et al [18], Wahba [28], [29], and Wahba et al [30]). 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 Loader [17] and see also the generalization to the multinomial logistic regression case in Cawley et al [6]), or the generalized additive model of Hastie and Tibshirani [14] (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 (Rasmussen and Williams [24]). 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 (Holmes and Adams [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 work 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 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 is guaranteed to converge to the maximum of the penalized likelihood function, and generally takes only around two or three iterations to converge. 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 data. 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, see [16]). On the other hand the computation time was much less than that of SVM and 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. In the next section we present the new approach. The following section details the approach for parameter estimation. Section 4 proposes some variants of the proposed approach. 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. The class membership y_m for training pattern x_m is defined as follows: it equals 1 if $x_m \in C_1$ and equals 0 if $x_m \in C_2$. In this work we consider only the two-class case. Let $P_m \equiv P(C_1|x_m)$ denote the posterior probability for class C_1. The purpose of the proposed method is to estimate the posterior probabilities P_m, 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

$$L = \prod_{m=1}^{M} P_m^{y_m} (1 - P_m)^{1-y_m} \tag{2}$$

Denote by $\mathcal{K}(x_m)$ as the set of K-nearest neighbors of point x_m (their indexes). We define a roughness function based on the square differences of the posteriors of neighboring data points. Specifically, it is given by

$$S = \frac{1}{K} \sum_{m=1}^{M+N} \sum_{m' \in \mathcal{K}(x_m)} (P_m - P_{m'})^2 \tag{3}$$

We define our overall objective function as a combination of the log-likelihood function and the roughness function:

$$\begin{aligned} J &= \log(L) - \lambda S & (4) \\ &= \sum_{m=1}^{M} \Big[y_m \log(P_m) + (1 - y_m) \log(1 - P_m) \Big] - \frac{\lambda}{K} \sum_{m=1}^{M+N} \sum_{m' \in \mathcal{K}(x_m)} (P_m - P_{m'})^2 & (5) \end{aligned}$$

The first term in the penalized log-likelihood J focuses on the goodness of fit aspect. It gauges how well that the considered P_m'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 S) will generally be smooth, and conversely a high S 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. Note that the testing patterns are also used in the expression for the smoothness function (the summation in S is over the entire data set). Even though they do not carry classification labels, they could be helpful in bridging the gaps between the training patterns to achieve a smoother fit. So in a way there is a semi-supervised element in the proposed approach. On the other hand, the summation for the log-likelihood function is over only the training set. The reason is that class labels are known only for the training set, but not for the test set.

3 The Proposed Algorithm

The goal is to solve the following maximization problem:

PROBLEM A): Maximize J (given by (5)) w.r.t. the variables: P_m, s.t. $0 \leq P_m \leq 1$, $m = 1, \ldots, M + N$.

It is easy to see that J is a convex function w.r.t. the P_m'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. Here is the algorithm:

1. Start with any initial choice e.g. $P_m = 0.5, \quad m = 1, \ldots, M+N$ (or another possible choice is $P_m = y_m, \quad m = 1, \ldots, M, \quad P_m = 0.5, \quad m = M+1, \ldots, M+N$).

2. For a number of iterations perform the following step:

3. For $m = 1$ to $M + N$:

   (a) If $x_m \in C_1$, then set:

$$P_m \equiv \frac{1}{2}\bar{P}_m + \frac{1}{2}\sqrt{(\bar{P}_m)^2 + \frac{2K}{\lambda(K + K_S)}} \tag{6}$$

   where

$$\bar{P}_m = \frac{1}{K + K_S}\Big[\sum_{m' \in \mathcal{K}(x_m)} P_{m'} + \sum_{m' \in \mathcal{S}(x_m)} P_{m'}\Big] \tag{7}$$

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

   (b) If $x_m \in C_2$, then set:

$$P_m \equiv 1 - \left[\frac{1 - \bar{P}_m}{2} + \frac{1}{2}\sqrt{(1 - \bar{P}_m)^2 + \frac{2K}{\lambda(K + K_S)}}\right] \tag{8}$$

   Note that $1 - \bar{P}_m$ here represents the neighborhood average of the posteriors of class C_2, i.e. the previous equation is the analogue of (6) but with tackling $1 - P_m$ instead of P_m.

   (c) If x_m is a test pattern, i.e. $M + 1 \leq m \leq M + N$, then set

$$P_m \equiv \bar{P}_m \tag{9}$$

   (d) Truncate if P_m goes out of the constraint box:

$$\text{Set } P_m = 1 \quad \text{if } P_m > 1 \quad \text{and set } P_m = 0 \quad \text{if } P_m < 0 \tag{10}$$

Essentially, what this algorithm performs is iterated local averaging of the posteriors (to obtain $\bar{P}_m$), and combining the resulting average in some way with the class membership (i.e. y_m) of the considered pattern (if known), through (6) and (8). Equations (6), (8), and (9) are basically the closed-form outcome of the one-variable search that is performed by cycling through all variables.

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_m's as the estimated posteriors of data points (whether training data or testing data). Recalling that $P_m \equiv P(C_1|x_m)$, then the final classification of a data point is estimated as class C_1 if $P_m > \frac{1}{2}$, otherwise it is class C_2.

An alternative way is to apply the proposed algorithm only on the training set. Then, once converged, we obtain the P_m's of the test patterns using Eq. (9) (i.e. as the average P_m's of K-nearest the training patterns).

Unconstrained optimization algorithms that alternately optimize w.r.t. one variable at a time are known to converge to the true optimum for convex functions (see [4]). However, this is generally *not* the case when constraints are present, even if the feasible region is convex. So a proof of convergence for our case has to be established (because of the presence of the box constraint, i.e. all the P_m's have to be between 0 and 1). This is given in the following theorem.

Theorem: The algorithm described above converges to the true maximum for Problem A).

Proof: Let us arrange the P_i's in a vector P, and let $P^* = (P_1^*, \ldots, P_{N+M}^*)^T$ denote the optimal solution of PROBLEM A) above. Consider that in some iteration we are operating on the variable P_m. Assume for the time being that it is a training pattern and that it belongs to class C_1. Maximizing J w.r.t. P_m can be obtained by taking the derivative of J in (5) w.r.t. P_m and equating to zero. We get

$$\frac{y_m}{P_m} - \frac{1-y_m}{1-P_m} - \frac{2\lambda}{K} \sum_{m' \in \mathcal{K}(x_m)} (P_m - P_{m'}) - \frac{2\lambda}{K} \sum_{m' \in \mathcal{S}(x_m)} (P_m - P_{m'}) = 0 \tag{11}$$

Setting $y_m = 1$ (since the pattern is from C_1), we obtain a quadratic equation, whose solution is given by (6). A similar derivation applies for patterns from class C_2 or for test patterns, leading to (8) and (9) respectively.

For simplicity denote $N' = N + M$. Assume that the algorithm converges to a point $P^0 = (P_1^0, \ldots, P_{N'}^0)^T$ which is *not* the global maximum. We will then show that this leads to a contradiction.

Let us reshuffle the indexes of posterior vector such that the following is true:

$$P_1^0 = \ldots = P_{N_1}^0 = 1 \tag{12}$$

$$P_{N_1+1}^0 = \ldots = P_{N_1+N_2}^0 = 0, \tag{13}$$

$$0 < P_i^0 < 1, \quad \text{for} \quad i = N_1 + N_2 + 1, \ldots, N' \tag{14}$$

for some N_1 and N_2. Since, it converged at that point, the following is true (due to the convexity of the objective function J and the fact that for each dimension that corresponds to a border point the maximum should be beyond that point):

$$\left.\frac{\partial J}{\partial P_i}\right|_{P^0} \geq 0 \quad \text{if} \quad i = 1, \ldots, N_1 \tag{15}$$

$$\leq 0 \quad \text{if} \quad i = N_1 + 1, \ldots, N_1 + N_2 \tag{16}$$

$$= 0 \quad \text{if} \quad i = N_1 + N_2 + 1, \ldots, N' \tag{17}$$

Consider the line connecting P^0 with the true optimum P^*. Let u be the distance along that line starting from P^0, i.e. a point P on the line is given by

$$P = P^0 + u\frac{P^* - P^0}{\|P^* - P^0\|} \tag{18}$$

The derivative of J along that line, evaluated at the point P^0 (i.e. corresponding to $u = 0$) is given by

$$\left.\frac{\partial J}{\partial u}\right|_{u=0} = \sum_{i=1}^{N'} \left.\frac{\partial J}{\partial P_i}\right|_{P^0} \frac{P_i^* - P_i^0}{\|P^* - P^0\|} \tag{19}$$

But according to (15) the components for which $\frac{\partial J}{\partial P_i}$ are positive correspond to $P_i^0 = 1$ and hence $P_i^* - P_i^0 \leq 0$ because $P_i^* \leq 1$ (and $P_i^0 = 1$). Similarly the components for which $\frac{\partial J}{\partial P_i}$ are negative correspond to $P_i^0 = 0$ and hence $P_i^* - P_i^0 \geq 0$ because $P_i^* \geq 0$. Hence, from (19) $\left.\frac{\partial J}{\partial u}\right|_{u=0} \leq 0$. But, if P^* is the true maximum (with $J(P^*) > J(P^0)$), then this cannot happen due to the convexity of J (the derivative along the line from P^0 to the maximum P^* cannot start negative or zero, then turn positive).

4 Parameter Selection

From the formulation in the previous section one can see that there are two main parameters that have to be determined, namely λ and K . Both parameters control the degree of smoothness, with K determining neighborhood domain for which the roughness measure is estimated, while λ is the weight attached to the roughness measure.

To determine K we first propose a measure of the variation of the posterior as a function of K. First let $KNN(x_m)$ denote the K^{th} nearest neighbor for point x_m, and denote by $C(x_m)$ the class membership of the point x_m. Then, the new measure is given by

$$V(K) = \frac{|A_K|}{M} \tag{20}$$

where M is the size of the training set, and $|A_K|$ means the size of the set A_K, defined as

$$A_K = \{m \in \{1, \ldots, M\} | C(x_m) = C(KNN(x_m))\} \tag{21}$$

The intuitive meaning of this measure is as follows. For each point in the training set x_m, we examine its K^{th} nearest neighbor and check if its class agrees with that of x_m. The fraction of these data points (i.e. those whose class membership agrees with that of its K^{th} nearest neighbor) represents the new measure $V(K)$ that we seek. For the very nearest neighbors, for example considering $V(1)$, the measure is an approximation of how much patterns in the same neighborhood are expected to agree (in class membership). (It can be shown that $V(1)$ equals approximately

$E(P^2(C_1|x) + P^2(C_2|x))$.) By considering higher values of K, the posterior probabilities will gradually change, and the measure $V(K)$ would then deviate from $V(1)$. The reason is that the class membership can be considered as the outcome of a Bernoulli trial with underlying probability being $P(C_1|x)$. Then, two Bernoulli experiments with more different underlying probabilities will naturally tend to disagree more. Thus the difference of $V(K)$ from $V(1)$ is a measure of how the landscape of the posterior probabilities changes with varying K. It should therefore be a good guide in choosing K. Essentially, we keep examining larger K's until the difference between $V(K)$ and $V(1)$ is significant enough to indicate that the this value of K stretches too far to sample significantly different values of the posterior probabilities.

The function $V(K)$ is a bit noisy, so we need to smooth it with a moving average. Figure 4 shows an example of $V(K)$ for an artificial two-dimensional problem with two Gaussian densities (after smoothing using a moving average window of size 21). As can be seen the behavior of $V(K)$ decays as K increases.

The value of K that is selected for the proposed classification method is determined as follows. Evaluate a normalized version:

$$V'(K) = \frac{V(1) - V(K)}{V(1) - \min(V(j))} \tag{22}$$

Starting from $K = 1$, find the first value of K immediately before the point when $V'(K)$ goes below a threshold (call this threshold V_{th}). This is the value of K we should select.

Concerning λ, by performing extensive simulations using synthetic problems with Gaussian distributions with a variety of class overlap level, sizes of the training set, dimension, etc, we found that the best range of λ's is generally in the range from 0.2 to 0.6. One good value to select is the middle value, $\lambda = 0.4$. Similarly, based also on experiments on Gaussian problems, the good range for V_{th} (the threshold discussed above for the $V'(K)$ function) is from 0.6 to 0.7. We selected the value 0.67 and fixed it on that for all simulations on real data. It seems that one fixed choice of the two-parameter set obtained through this large collection of Gaussian problems has led to more robust results than tuning the parameters for each individual problem (using the training set). While the latter approach will make the parameter values more specific to the problem, it introduces estimation error due to the finite sample nature of the training set. This estimation error is more dominant in our situation perhaps because there is no strong performance sensitivity with respect the parameter values in the ranges considered above.

An alternative approach is to apply the so-called thick modeling approach (Granger and Jeon [9]). In this approach we design a number of classifiers, each for some specific parameter values in the good range (for example some combinations of $\lambda = 0.2,\ \lambda = 0.4, \lambda = 0.6$, and $V_{th} = 0.6,\ V_{th} = 0.65,\ V_{th} = 0.7$), and then pool their outputs. This means that once each classifier is designed (one classifier for each combination of λ/V_{th}), for each data point we take the estimated posteriors of the different classifiers and average them. By observing the extensive simulations on problems with the Gaussian distributions, we found that this latter approach did not provide improvement over the

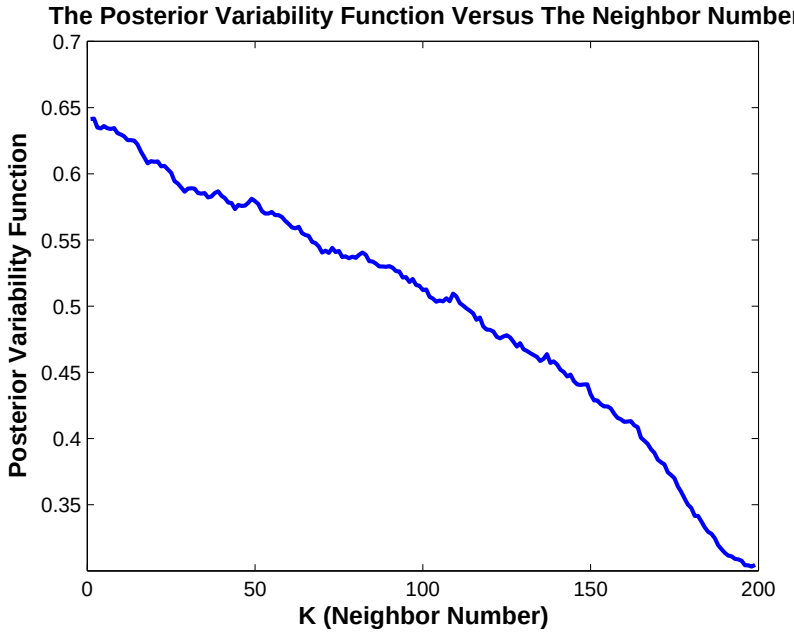

Figure 1:

single-parameter-value model (discussed above). Nevertheless, it is a useful approach to consider. Both discussed approaches yield generally better than other more sophisticated parameter selection procedures such as the K-fold validation procedure.

5 Variants of the Proposed Method

5.1 Distance Weighting

In the K nearest neighbor literature, distance-weighted versions have been reported to offer beneficial performance (see Atiya [2], Bailey et al [3], and Dudani [8]). In these approaches the neighbors of some point x_m are weighted according to their distance or their order, rather than treated equally as in the standard K-nearest neighbor method. We have also considered here the concept of weighted KNN, and propose the following weighting function:

$$v_k = \frac{d_{K+1} - d_k}{\sum_{j=1}^{K}(d_{K+1} - d_j)} K \tag{23}$$

where d_k denotes the distance between the considered point (say point x_m) and its k^{th} nearest neighbor. (Note that the weights sum to K rather than 1 to keep the correspondence with the standard K-nearest neighbor method, where all K weights equal 1.) The penalized log-likelihood becomes:

$$J = \sum_{m=1}^{M} \Big[y_m \log(P_m) + (1 - y_m) \log(1 - P_m) \Big] - \frac{\lambda}{K} \sum_{m=1}^{M+N} \sum_{m' \in \mathcal{K}(x_m)} w_{m,m'} (P_m - P_{m'})^2 \tag{24}$$

where $w_{m,m'}$ equals v_k if $x_{m'}$ is the k^{th} nearest neighbor of x_m, and equals zero if $x_{m'}$ is not among the K nearest neighbors of x_m.

The update in this situation will turn out to be the following. If $x_m \in C_1$, then

$$P_m \equiv \frac{1}{2} \bar{P}_m + \frac{1}{2} \sqrt{\bar{P}_m^2 + \frac{2K}{\lambda W}} \tag{25}$$

If $x_m \in C_2$ then update as:

$$P_m \equiv 1 - \left[\frac{1 - \bar{P}_m}{2} + \frac{1}{2} \sqrt{(1 - \bar{P}_m)^2 + \frac{2K}{\lambda W}} \right] \tag{26}$$

If x_m is a test pattern, then:

$$P_m \equiv \bar{P}_m \tag{27}$$

where

$$\bar{P}_m = \frac{1}{W} \Big[\sum_{m' \in \mathcal{K}(x_m)} w_{m,m'} P_{m'} + \sum_{m' \in \mathcal{S}(x_m)} w_{m',m} P_{m'} \Big] \tag{28}$$

$$W = \sum_{m' \in \mathcal{K}(x_m)} w_{m,m'} + \sum_{m' \in \mathcal{S}(x_m)} w_{m',m} \tag{29}$$

$$= K + \sum_{m' \in \mathcal{S}(x_m)} w_{m',m} \tag{30}$$

5.2 Class Pattern Balancing

In many classification problems the distribution of patterns among classes is not balanced. For instance, in medical diagnosis there may only be a small number of patients having a certain disease compared to a much larger number of persons that are tested. The receiver operating characteristics (ROC) has been recognized as an essential tool for the analysis of imbalanced datasets and has been widely used in the medical diagnosis field [27], and also in the pattern classification field in general [21]. The ROC curve is using the so-called *sensitivity* and *specificity*, as shown by the curve in Fig. 5.2. The mathematical definitions of sensitivity and specificity are given by (see also Table 1):

$$\text{Sensitivity} = \frac{\text{No. of true positives}}{\text{No. of true positives} + \text{No. of false negatives}} \tag{31}$$

$$\text{Specificity} = \frac{\text{No. of true negatives}}{\text{No. of true negatives} + \text{No. of false positives}} \tag{32}$$

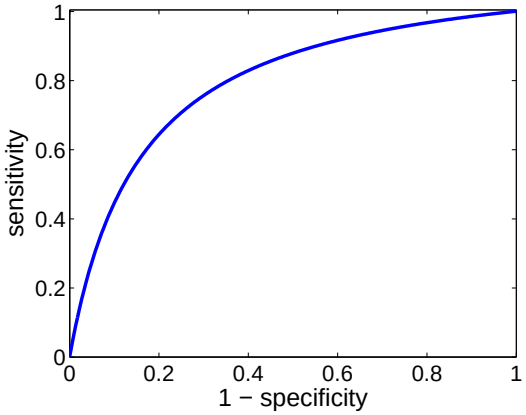

Figure 2: Example of an ROC curve

		Actual outcome	
		True	False
Classifier outcome	positive	True positive	False positive
	negative	False negative	True negative
		Sensitivity	Specificity

Table 1: Sensitivity and Specificity

If a classification algorithm merely attempts to maximize the classification accuracy (no. of true positives + no. of true negatives) without taking into consideration the individual accuracy of each class, then such a system may not be very beneficial. Hence, to better evaluate the performance of a classifier when dealing with imbalanced data, we are going to consider also the averaged class-wise accuracy, which reflects the trade-off between sensitivity and specificity. Many pattern classification algorithms, including the traditional K-nearest neighbor, do not take the class-wise accuracy into consideration. We propose in this section a procedure that attempts to address this issue. The procedure is implemented as follows:

- For every pattern x_m find the K-nearest neighbors that belong to only a specific class C_j ($j = 1,\ 2$). Compute the average of the distances from that point x_m to these K-nearest neighbors. Let that average be $\bar{d}_{j,m}$.

- Compute a distance weighting function

$$Q = \frac{\sum_m \bar{d}_{2,m}}{\sum_m \bar{d}_{1,m} + \sum_m \bar{d}_{2,m}} \tag{33}$$

- Given a pattern x_m that needs to be classified, the K nearest neighbors are computed by re-weighting the distances according to Q. Specifically we multiply all distances from x_m to

the training patterns from class C_1 by Q. Also, we multiply all distances from x_m to the training patterns from class C_2 by $1 - Q$. The resulting distances are then sorted and the smallest K distances are selected to give the new K nearest neighbors.

According to this procedure, if class C_1 is underrepresented, then the distances $\bar{d}_{1,m}$ will tend to be higher than $\bar{d}_{2,m}$. The reason is that every pattern has to reach out further to get to its K nearest neighbors that are from the same class. Therefore, it is expected to have $Q < 0.5$ in that situation. Due to the re-weighting step, the lower value of Q will make the patterns that belong to C_1 more represented in the new K nearest neighbors. We use this procedure in selecting the K-nearest neighbors for the proposed penalized likelihood method. We experimented this procedure on some Gaussian problems, and found a noticeable improvement in averaged class-wise classification accuracy for the case of imbalanced class distribution. We are going to present results with and without this variant of the penalized likelihood classifier.

5.3 Multi-Class Case

The proposed approach has been developed primarily for the two-class case. It is a little hard to generalize it to the multi-class case. The optimization formulation does not yield to a straightforward or simple algorithm (as it did for the two-class case), due mainly to the more involved inequality constraints. One way to tackle the multi-class case is to divide the original problem into binary subtasks and repeatedly apply the two-class formulation on these. Approaches along this line have been investigated thoroughly in the literature for SVM's. SVM's are originally two-class classifiers, but extensions to the multi-class case include methods such as one versus all, one versus one, binary tree based approaches, etc. To avoid distraction, these multi-class extensions in conjunction with the proposed penalized likelihood method will not be investigated in this work, and will be tackled it in a future study.

6 Simulation Results

To test the performance of the proposed method, we have conducted a comparative study using a number of real-world benchmark problems. We have compared the performance of the proposed method to that of the following well-known methods:

- Bayes classifier (Duda et al [7], p. 168) with the class-conditional densities estimated according to the Parzen window density estimator (PARZEN) (Silverman [26]). A key parameter for the Parzen estimator is the width of kernels h. We used the value derived by [26] (Silverman's rule):

$$h = \hat{\sigma} \left[\frac{4}{(2L+1)I} \right]^{\frac{1}{L+4}} \tag{34}$$

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 mixture model classifier (GMM) [20]. It is basically a Bayes classifier with the class-conditional densities estimated as a mixture of Gaussian functions. We used the software developed by [20], where the mixture estimation algorithm is chosen to be the EM algorithm.

- Gaussian process classification using the expectation propagation approximation [24]. 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 [23].

- Support vector machines (SVM) (Scholkopf and Smola [25]). We tested two versions. A linear SVM (referred to as SVM1) implemented using the Liblinear SVM software[1], and a radial basis function SVM (referred to as SVM2) implemented using the OSUsvm toolbox[2]. 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). In addition to the traditional algorithm (KNN), we also used a weighted version (KNNw), where weights have been assigned to the neighbors based on their distance from the tested pattern (Eq. 23).

- 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 [33][3].

- Neighborhood components analysis (NCA). This algorithm attempts to maximize a stochastic variant of the leave-one-out KNN score on the training set [**?**][4].

The parameters of the proposed penalized likelihood classification method were selected as described in Section 4 (K is selected based on the posterior variability function with a smoothing window of size 3 and cut-off threshold of 0.67 for $V'(K)$, and $\lambda = 0.4$). We used the weighted distance version (described in Subsection 5.1) without class pattern balancing (abbreviated as PLC) and with the class pattern balancing modification that was described in Subsection 5.2 (referred to as PLCm). Table 6 lists the classification models considered in the comparison study, together with their abbreviations.

[1] obtained from http://www.csie.ntu.edu.tw/ cjlin/liblinear/

[2] obtained from http://downloads.sourceforge.net/svm/osu-svm-3.0.zip

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

[4] the NCA software is available at http://www.cs.berkeley.edu/ fowlkes/software/nca/

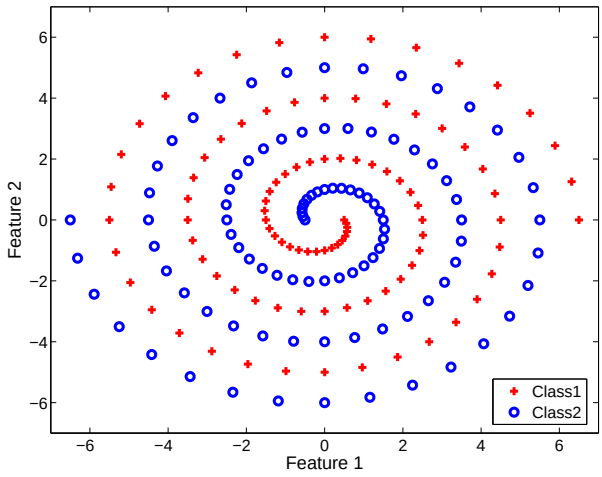

Figure 3: Two-spiral dataset

We tested all these competing methods on real-world pattern classification problems, mostly from the UCI repository [1]. We also tested those algorithms on the Brain Computer Interface (BCI) problem, which is a challenging biosignal driven application that utilizes the electroencephalogram (EEG) signal. The EEG is a recording of electrical activity originating from the brain. The EEG dataset used in this paper was taken from the Department of Medical Informatics in the University of Technology, Graz in Austria. The EEG signals were recorded from three right handed females who were asked to imagine right and left finger movements. A total of 406 trials were used, 208 for the left movement and 198 for right movement. Two channels were used here with five frequency bands extracted from each channel. More details on experiment set-up can be found in Ramoser et al [22]. In addition, the well-known two-spiral classification problem is also used. This dataset consists of points on two inter-wined spirals that cannot be linearly separated, as shown in Fig. 6. Table 6 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 the contraceptive method choice (cmc) dataset, which corresponds to a 3-class classification problem, we considered here distinguishing between classes 2 and 3 only (long term vs. short term contraceptive). For the teaching assistant dataset, which represents the evaluation of teaching performance according to 3 classes, we chose to combine classes 2 and 3, and hence form a two-class problem, i.e., low vs. medium or high. 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

Classifier	Abbreviation
Parzen window Bayes classifier	Parzen
Gaussian mixture model Bayes classifier	GMM
Gaussian process classifier (using EP, non-optimized)	GPC
Gaussian process classifier (using EP, non-optimized)	GPCo
Support vector machines (linear)	SVM1
Support vector machines (RBF kernel)	SVM2
K-nearest neighbor	KNN
Weighted K-nearest neighbor	KNNw
Evidential K-nearest neighbor	KNNds
Neighborhood components analysis	NCA
Proposed penalized likelihood classifier (without class balancing)	PLC
Proposed penalized likelihood classifier (with class balancing)	PLCm

Table 2: The classification models used in the comparison, and their abbreviations

Dataset	# Attributes	# Patterns	Class distribution
Aus. Credit	14	690	0.44/0.56
Ger. Credit	24	1000	0.70/0.30
Cylender bands	30	350	0.62/0.38
Blood transfusion	4	748	0.24/0.76
cancer	9	683	0.65/0.35
census income	14	>1500	0.75/0.25
contracep. meth. choi.	9	844	0.39/0.61
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
musk	166	476	0.57/0.43
parkinsons	22	195	0.75/0.25
pima	8	768	0.35/0.65
sonar	60	208	0.53/0.47
Teaching Assistant	5	161	0.68/0.32
Two Spiral	2	194	0.50/0.50
WDBC	30	569	0.63/0.37
EEG	10	406	0.49/0.51

Table 3: Datasets used to evaluate the performance of classifiers

the test sets of the 20 runs.

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.
- **Mean class-wise accuracy (Acw).** This measure is more suitable for imbalanced classes, as it is calculated by averaging the accuracies of class 1 and class 2. In other words, it is the average of the sensitivity and the specificity.
- **Estimated standard deviation of the accuracy.** It is calculated by dividing the standard deviation of Acc (or Acw) 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:

$$\exp \frac{\sum_{i=1}^{n} \log(ai/bi)}{n} = \sqrt[n]{\prod_{i=1}^{n} ai/bi} \tag{35}$$

  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 [32, 31].

Table 6 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 NCA. However, the results indicate that PLC is quite similar to KNNw and it is not significantly better than the Gaussian process and SVM, particularly GPCo and SVM2. As mentioned earlier, these two classifiers are considered in the literature to be among the best classification approaches. The table also indicates that KNNw is better than KNN in classifying most of the datasets. It makes sense to have the effects of the different neighbors taper off as we move far away. Similarly GPCo is found to be better than GPC and SVM2 is slightly better than SVM1.

In order to present a more detailed analysis of the classification results, Table 6 gives more comparison measures. The first row of the table represents the 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 SVM2. 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, GMM, KNN, KNNw, KNNds and NCA. In fact the geometric error ratio also ranks PLC second after GPCo. On the other hand, the win-tie-loss favors the Gaussian process and support vector machine variants over PLC. However, as seen from the p-value numbers from among these classifiers only the outperformance of only GPCo is significant. It is worth mentioning that for the hill/valley and musk datasets, the GMM classifier could not produce any results, as there was an error in estimating the probabilities. Hence, for this particular classifier the remaining 19 datasets were only used in the measures of Table 6.

As shown in Table 6 the class distribution of many of the datasets used are unbalanced. Hence, to give a better indication about the performance of the various classifiers, the mean class-wise accuracy (Acw) is used, as shown in Table 6. It is clear from this figure that PLCm has managed to achieve a higher bullet/open circle ratio than that of Table 6. With the exception of GPCo, the significant outperformance of PLCm over the remaining classifiers is quite obvious. The results presented in Table 6 further prove the superiority of PLCm. The mean accuracy clearly favors PLCm over all other methods, including GPCo, with close to 2% improvment over its closest rival. The geometric error ratio of all methods with respect to PLCm is greater than 1, i.e., the rest of the methods achieved worse error than PLCm. The win-tie-loss also favors PLCm, while GPCo being very close (11 vs. 10).

The above results indicate that the performance of PLCm and is quite close to that of GPCo, and the both outperform the remaining classification methods. So, we thought it would be important to compare these two classifiers in terms of computational complexity. Table 6 shows the computation time of both GPCo and PLCm for all considered datasets. The table indicates that PLCm is considerably faster than GPCo, which represents a great advantage for the proposed algorithm.

In summary, the proposed PLC method offers a performance commensurate with the top classification methods, including the optimized expectation propagation Gaussian process, but with much smaller computational requirements. As such, it can be ranked among the top binary classification

	Parzen	GMM	GPC	GPCo	SVM1	SVM2
AusCred	84.06±0.64	83.12±1.18	85.98±0.59	85.91±0.53	85.54±0.51	85.00±0.56
GerCred	68.52±0.38•	64.25±3.57•	76.90±0.51○	77.15±0.60○	77.55±0.51○	75.17±0.47○
bands	67.71±1.21•	72.64±1.07	74.21±0.99○	75.29±1.06○	70.79±0.89	73.21±0.75○
Btrans	77.33±0.40	76.93±0.68	76.47±0.31	78.17±0.54○	76.50±0.27	76.17±0.19
cancer	96.39±0.35	95.62±0.31•	96.93±0.31	96.93±0.31	96.72±0.32	96.75±0.32
census	79.63±0.57	73.47±1.35•	81.78±0.55	83.63±0.52○	82.35±0.54○	82.40±0.44○
cmc	62.46±0.72	57.04±1.86•	63.93±0.63○	63.55±0.62○	63.11±0.73	62.93±0.67
EEG	63.84±1.08	63.78±0.99	66.59±1.18	68.23±1.06○	66.77±0.99	67.26±1.22
haber	71.80±0.60○	66.89±1.32	74.18±0.64○	74.59±0.75○	74.75±0.73○	72.87±0.44○
heart	79.91±1.30•	82.36±0.79	82.74±1.05	84.81±0.68	83.77±1.05	84.43±0.78
heartS	79.17±1.32	77.41±0.92	82.13±0.94	84.26±1.00○	84.07±1.06○	80.83±1.14
hill	53.72±0.72•		49.59±0.66•	50.74±0.46•	60.00±0.53○	49.30±0.59•
ion	87.14±0.66	87.14±1.18	88.93±0.64	95.07±0.66○	86.50±1.03	93.93±0.58○
mamm	80.19±0.65	80.74±0.71	82.50±0.65○	83.09±0.66○	82.41±0.70○	82.69±0.80○
musk	85.11±0.81		86.84±0.90	90.00±0.68○	84.63±0.75	90.84±0.67○
parkinson	94.74±0.78	83.46±0.94•	83.97±1.00•	91.67±0.98	83.33±0.78•	87.44±0.89•
pima	72.05±0.79	68.12±1.15•	75.32±0.61	76.33±0.61○	75.62±0.54	75.88±0.56○
sonar	83.05±1.45	65.12±3.27•	83.90±1.22	83.41±1.27	76.34±1.24•	86.10±1.19
TeachAs	73.83±1.42○	69.83±1.29	68.83±1.22	72.67±1.50	68.67±1.66	68.33±1.12
TwoSpiral	33.68±1.44•	43.16±1.09•	48.68±1.59•	50.53±0.31•	50.92±1.75•	48.95±1.40•
WDBC	96.73±0.31	95.18±0.35•	97.35±0.31	97.57±0.31	97.43±0.25	97.26±0.29

	KNN	KNNw	KNNds	NCA	PLC	PLCm
AusCred	85.62±0.55	85.00±0.64	84.38±0.55	83.88±0.72	85.29±0.62	85.40±0.58
GerCred	72.03±0.42	72.32±0.37	71.37±0.51	70.90±0.84	71.75±0.38	67.00±0.82•
bands	70.14±1.03	69.29±1.02	67.71±1.16•	65.00±1.85•	70.79±0.75	71.86±0.96
Btrans	78.17±0.47○	77.67±0.63	73.80±0.62•	76.33±0.57	76.33±0.62	71.27±0.76•
cancer	96.93±0.34	96.61±0.32	97.15±0.28	97.19±0.38	97.08±0.33	96.79±0.31
census	80.35±0.55	79.90±0.64	78.38±0.60•	79.10±0.71	80.28±0.58	73.40±0.73•
cmc	61.36±0.78	60.83±0.81	62.07±0.80	60.33±1.30	61.39±0.64	60.98±0.63
EEG	63.35±1.10	62.99±1.00	62.20±0.99	60.30±1.56•	64.94±1.01	64.57±1.02
haber	74.67±0.72○	72.87±0.75○	67.70±0.85	74.18±0.58○	67.54±1.10	66.72±1.36
heart	81.32±0.96	80.94±1.34	80.75±1.27	80.57±1.46	83.58±0.66	80.47±1.24•
heartS	81.48±1.08	80.93±1.04	78.61±1.10	75.74±1.40•	79.54±0.96	79.26±1.03
hill	55.00±0.97	56.07±0.94	55.50±1.11	52.69±0.69•	57.11±0.77	57.60±0.82
ion	84.79±0.56•	86.43±0.79	88.71±0.54	84.93±1.03•	87.50±0.65	94.93±0.70○
mamm	80.90±0.59	80.25±0.72	79.17±0.74	82.22±0.76○	79.44±0.90	78.83±0.73
musk	82.16±1.06•	84.37±0.99	82.42±1.02•	67.32±1.47•	85.47±0.92	87.21±0.83
parkinson	92.05±1.27	94.23±0.83	93.21±1.04	82.69±1.49•	92.95±0.93	87.69±1.24•
pima	72.99±0.71	73.21±0.92	72.76±0.89	73.02±1.05	73.73±0.79	72.53±0.71
sonar	81.59±1.33	85.00±1.22	82.07±1.19	70.49±2.16•	84.76±1.13	85.85±0.96
TeachAs	68.50±1.42	71.50±1.15	70.50±1.53	56.83±2.35•	69.00±1.69	68.17±2.06
TwoSpiral	74.74±2.10•	83.68±1.41•	74.74±2.10•	85.26±3.93	88.03±1.14	87.63±1.08
WDBC	96.68±0.37	96.55±0.44	97.12±0.35	96.73±0.36	97.39±0.32	96.68±0.33

Table 4: Classification accuracy and estimated standard deviation for the considered classifiers. The abbreviations of the considered classifiers can be found in Table 6.

	Parz	GMM	GPC	GPCo	SVM1	SVM2	KNN	KNNw	KNNds	NCA	PLC	PLCm
Mean acc.	75.76	73.56	77.51	**79.22**	77.51	77.99	77.85	78.60	77.16	75.03	78.76	77.85
Parz												
$\dot{r}$		0.85	1.053	1.192	1.03	1.109	1.039	1.086	1.03	0.928	1.116	1.077
s		14-1-4	4-0-17	3-0-18	6-0-15	4-0-17	8-0-13	7-0-14	11-1-9	13-1-7	6-0-15	9-0-12
p		0.031	0.007	0.001	0.078	0.007	0.383	0.189	0.824	0.263	0.078	0.664
GMM												
$\dot{r}$			1.243	1.403	1.209	1.292	1.238	1.289	1.223	1.13	1.321	1.261
s			2-0-17	0-0-19	5-0-14	2-0-17	5-0-14	5-0-14	5-0-14	8-0-11	4-0-15	7-0-12
p			0.001	0	0.064	0.001	0.064	0.064	0.064	0.648	0.019	0.359
GPC												
$\dot{r}$				1.132	0.979	1.054	0.987	1.032	0.978	0.881	1.06	1.023
s				3-1-17	10-0-11	11-0-10	15-1-5	15-0-6	16-0-5	17-1-3	13-0-8	15-0-6
p				0.003	1	1	0.041	0.078	0.027	0.003	0.383	0.078
GPCo												
$\dot{r}$					0.865	0.931	0.872	0.911	0.864	0.779	0.937	0.904
s					17-0-4	19-0-2	15-2-4	17-0-4	17-0-4	18-0-3	16-0-5	18-0-3
p					0.007	0	0.019	0.007	0.007	0.001	0.027	0.001
SVM1												
$\dot{r}$						1.077	1.009	1.054	0.999	0.9	1.083	1.045
s						10-0-11	15-0-6	16-0-5	15-0-6	19-0-2	13-1-7	14-0-7
p						1	0.078	0.027	0.078	0	0.263	0.189
SVM2												
$\dot{r}$							0.937	0.979	0.928	0.836	1.006	0.971
s							12-0-9	13-2-6	16-0-5	16-0-5	13-0-8	15-0-6
p							0.664	0.167	0.027	0.027	0.383	0.078
KNN												
$\dot{r}$								1.045	0.991	0.893	1.074	1.036
s								12-0-9	11-1-9	15-0-6	7-0-14	13-1-7
p								0.664	0.824	0.078	0.189	0.263
KNNw												
$\dot{r}$									0.948	0.854	1.028	0.992
s									17-0-4	16-0-5	8-0-13	10-0-11
p									0.007	0.027	0.383	1
KNNds												
$\dot{r}$										0.901	1.084	1.046
s										14-0-7	6-0-15	12-0-9
p										0.189	0.078	0.664
NCA												
$\dot{r}$											1.203	1.161
s											3-1-17	9-0-12
p											0.003	0.664
PLC												
$\dot{r}$												0.965
s												15-0-6
p												0.078

Table 5: Comparison of averaged classification accuracy, geometric error, win-tie-loss, and p-value of the sign test across all the used datasets. The abbreviations of the considered classifiers can be found in Table 6.

	Parzen	GMM	GPC	GPCo	SVM1	SVM2
AusCred	83.93±0.64•	83.04±1.12•	85.89±0.61	85.87±0.55	85.96±0.48	85.18±0.56
GerCred	61.45±0.48•	59.25±3.24	67.29±0.62	67.94±0.74○	69.30±0.69○	64.17±0.67
bands	65.27±1.27•	68.25±1.20	70.28±1.11	72.47±1.12	66.89±1.06•	69.10±0.88
Btrans	57.01±0.65•	63.78±1.10	53.21±0.44•	58.65±0.76•	52.89±0.40•	50.77±0.38•
cancer	95.85±0.44•	95.79±0.31•	96.78±0.37	96.85±0.36	96.46±0.37•	96.76±0.36
census	73.45±0.68	72.87±0.79•	74.38±0.70	77.47±0.70○	75.03±0.70	71.92±0.58•
cmc	60.36±0.80	58.21±1.38	60.88±0.65	60.30±0.63	59.83±0.76	59.42±0.70
EEG	63.87±1.08	63.84±0.99	66.66±1.18	68.27±1.06○	66.85±1.00	67.48±1.22
haber	52.49±0.74•	53.29±1.26•	53.50±0.88•	57.10±1.04	55.30±1.01•	49.49±0.28•
heart	70.38±1.76	71.76±1.34	68.31±1.53	69.62±1.24	73.83±1.85	67.70±1.40
heartS	78.69±1.38	77.02±0.88	81.56±0.96	84.04±0.96○	83.81±1.02○	80.15±1.15
hill	52.87±0.72•		48.76±0.66•	50.25±0.42•	59.02±0.54	48.39±0.58•
ion	82.93±0.91•	87.56±1.06•	85.52±0.81•	94.12±0.77	82.26±1.36•	93.23±0.63
mamm	80.34±0.65	80.87±0.72	82.64±0.65○	83.12±0.66○	82.55±0.69○	82.85±0.79○
musk	85.86±0.77		86.91±0.89	89.63±0.70○	84.46±0.80	90.07±0.72○
parkinson	94.01±1.10○	67.75±1.83•	71.86±1.77•	87.03±1.56	72.74±1.46•	76.32±1.75•
pima	67.08±1.03•	64.42±1.23•	70.01±0.73•	71.83±0.71	70.24±0.71•	70.22±0.71•
sonar	82.72±1.49	64.84±3.63•	83.67±1.25	83.33±1.29	76.32±1.22•	85.95±1.20
TeachAs	67.50±1.80	69.25±1.54	61.37±1.48	66.75±1.95	61.38±1.64	62.13±1.91
TwoSpiral	33.68±1.44•	43.16±1.09•	48.68±1.59•	50.53±0.31•	50.92±1.75•	48.95±1.40•
WDBC	96.20±0.37	95.38±0.34	96.72±0.42	97.29±0.35	96.84±0.32	96.87±0.36

	KNN	KNNw	KNNds	NCA	PLC	PLCm
AusCred	85.62±0.57	84.95±0.66	84.26±0.56	83.71±0.75•	85.27±0.63	85.71±0.59
GerCred	59.18±0.54•	61.83±0.48•	62.65±0.60•	61.90±1.31•	61.56±0.49•	65.48±0.90
bands	64.41±1.21•	64.71±1.22•	64.09±1.26•	61.34±2.04•	66.65±0.85•	70.06±1.08
Btrans	61.36±0.82•	61.98±0.99•	60.76±1.01•	52.79±0.70•	61.29±0.91•	66.32±1.08
cancer	96.70±0.41	96.36±0.39•	97.71±0.23	97.19±0.41	97.18±0.38	97.43±0.25
census	73.77±0.73	73.56±0.84	72.22±0.76•	72.03±1.04•	73.56±0.75	75.29±0.70
cmc	58.87±0.80	58.45±0.87•	59.97±0.83	56.58±1.37•	58.85±0.80	61.01±0.75
EEG	63.44±1.09	63.07±1.00	62.31±0.99	60.32±1.56•	64.99±1.01	64.63±1.02
haber	57.06±1.04	55.93±1.05•	54.05±0.72•	53.70±1.00•	52.93±1.03•	59.82±1.44
heart	72.45±1.75	70.19±1.60	66.39±2.42	67.27±3.13	68.17±1.26	68.89±1.68
heartS	80.85±1.11	80.27±1.07	78.23±1.14	75.19±1.43•	79.02±0.98	79.04±1.03
hill	54.95±0.97	56.04±0.95	55.50±1.11	51.71±0.73•	57.09±0.77	57.46±0.83
ion	79.72±0.78•	82.16±1.07•	85.22±0.72•	81.21±1.27•	83.74±0.86•	94.99±0.71
mamm	81.00±0.59○	80.31±0.72	79.19±0.74	82.25±0.75○	79.49±0.90	79.04±0.73
musk	83.20±0.99•	85.27±0.93	83.47±0.96•	66.90±1.49•	85.34±0.91	86.74±0.87
parkinson	89.74±1.88	94.32±1.23○	91.50±1.65	72.80±2.13•	89.85±1.52	88.61±1.43
pima	67.40±0.72•	68.62±1.04•	68.76±1.03•	68.15±1.28•	69.40±0.91•	72.33±0.76
sonar	81.14±1.32•	84.52±1.26	81.59±1.19•	70.47±2.16•	84.41±1.15	86.01±0.96
TeachAs	62.88±1.67	66.13±1.39	64.25±1.67	49.50±2.81•	62.87±1.94	66.88±2.35
TwoSpiral	74.74±2.10•	83.68±1.41•	74.74±2.10•	85.26±3.93	88.03±1.14	87.63±1.08
WDBC	95.92±0.45	96.01±0.51	96.52±0.46	95.91±0.45	96.88±0.40	96.36±0.40

Table 6: Averaged class-wise accuracy and estimated standard deviation for the considered classifiers. The abbreviations of the considered classifiers can be found in Table 6.

	Parz	GMM	GPC	GPCo	SVM1	SVM2	KNN	KNNw	KNNds	NCA	PLC	PLCm
Mean acc.	71.71	73.56	72.14	74.88	72.52	72.24	73.54	74.68	73.49	69.82	74.60	**76.65**
Parz												
$\dot{r}$		0.897	0.998	1.155	0.99	1.043	1.024	1.095	1.053	0.902	1.103	1.217
s		11-0-8	5-0-16	5-0-16	7-0-14	8-0-13	10-0-11	9-0-12	11-0-10	15-0-6	6-0-15	4-0-17
p		0.648	0.027	0.027	0.189	0.383	1	0.664	1	0.078	0.078	0.007
GMM												
$\dot{r}$			1.113	1.289	1.099	1.152	1.151	1.23	1.186	1.041	1.239	1.372
s			4-0-15	3-0-16	4-0-15	5-0-14	6-0-13	7-0-12	8-0-11	9-0-10	7-0-12	3-0-16
p			0.019	0.004	0.019	0.064	0.167	0.359	0.648	1	0.359	0.004
GPC												
$\dot{r}$				1.157	0.991	1.045	1.025	1.097	1.055	0.903	1.105	1.219
s				3-0-18	8-0-13	11-0-10	14-0-7	13-0-8	14-0-7	16-0-5	13-0-8	8-0-13
p				0.001	0.383	1	0.189	0.383	0.189	0.027	0.383	0.383
GPCo												
$\dot{r}$					0.857	0.904	0.886	0.948	0.912	0.781	0.955	1.054
s					16-0-5	19-0-2	16-0-5	15-0-6	16-0-5	18-0-3	15-0-6	10-0-11
p					0.027	0	0.027	0.078	0.027	0.001	0.078	1
SVM1												
$\dot{r}$						1.054	1.034	1.106	1.064	0.911	1.115	1.229
s						11-0-10	14-0-7	14-0-7	13-0-8	18-0-3	12-0-9	8-0-13
p						1	0.189	0.189	0.383	0.001	0.664	0.383
SVM2												
$\dot{r}$							0.981	1.049	1.009	0.864	1.057	1.166
s							11-0-10	12-0-9	12-0-9	15-0-6	10-0-11	5-0-16
p							1	0.664	0.664	0.078	1	0.027
KNN												
$\dot{r}$								1.07	1.028	0.881	1.078	1.189
s								9-0-12	9-1-11	15-0-6	9-0-12	4-0-17
p								0.664	0.824	0.078	0.664	0.007
KNNw												
$\dot{r}$									0.962	0.824	1.008	1.111
s									15-0-6	17-0-4	9-1-11	4-0-17
p									0.078	0.007	0.824	0.007
KNNds												
$\dot{r}$										0.857	1.048	1.156
s										18-0-3	7-0-14	4-0-17
p										0.001	0.189	0.007
NCA												
$\dot{r}$											1.223	1.349
s											4-0-17	1-0-20
p											0.007	0
PLC												
$\dot{r}$												1.103
s												5-0-16
p												0.027

Table 7: Comparison of averaged class-wise accuracy, geometric error, win-tie-loss, and p-value of the sign test across all the used datasets. The abbreviations of the considered classifiers can be found in Table 6.

	AusCred	GerCred	bands	Btrans	cancer	census	cmc	EEG	haber	heart	
GPCo	102.9	534.8	18.16	155.8	136.2	2722	211.0	25.86	12.70	12.72	
PLCm	1.177	3.347	0.382	1.299	1.072	9.416	1.684	0.413	0.255	0.291	

	heartS	hill	ion	mamm	musk	parkin	pima	sonar	teach	TwoSpiral	WDBC
GPCo	9.233	24.28	24.01	185.1	50.14	6.336	176.0	7.044	2.115	1.784	76.78
PLCm	0.241	1.569	0.462	1.413	1.397	0.156	1.305	0.200	0.108	0.135	0.927

Table 8: Execution Time for GPCo and PLCm, measured in CPU time (sec). This time includes training time and testing time

algorithms.

7 Conclusion

In this paper we have developed a new classification method based on the penalized likelihood concept. The new method is based on defining a roughness term based on the K-nearest neighbors. We have developed an algorithm that is guaranteed to converge to the global optimum. We have also developed variants of the proposed method that can handle aspects such as unbalanced class distribution. The proposed method was compared with several existing classification methods. It gave a performance competitive with the top model, but with a much less computational time. We therefore believe that the proposed approach offers superior performance and speed advantages, and as such it should be one of the major contenders to be tested or used in any binary classification task.

References

[1] Asuncion, D. J., 2007. UCI Machine Learning Repository, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[2] Atiya, A. F. Estimating the posterior probabilities using the k-nearest neighbor rule. *Neural Computation 17* (2006), 731–740.

[3] Bailey, T., and Jain, A. A note on distance-weighted k-nearest neighbor rules. *IEEE Trans. Systems, Man, Cybernetics 8* (1978), 311–313.

[4] Bazaraa, M., and Shetty, C. *Nonlinear Programming.* John Wiley & Sons, 1979.

[5] Berry, S. M., Carroll, R. J., and Ruppert, D. Bayesian smoothing and regression splines for measurement error problems. *J. Amer. Statist. Assoc. 97*, 457 (2002), 160–169.

[6] Cawley, G., Talbot, N. L., , and Girolami, M. Sparse multinomial logistic regression via bayesian l1 regularisation. *Proceedings NIPS* (2007), 209–216.

[7] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification, 2nd Edition.* Wiley Interscience, 2000.

[8] Dudani, S. The distance weighted k-nearest-neighbor rule. *IEEE Trans. Systems, Man, Cybernetics 6* (1976), 325–327.

[9] Granger, C. W. J., and Jeon, Y. Thick modeling. *Economic Modeling 21*, 2 (2004), 323–343.

[10] Green, P. Penalized likelihood, 1999. In Encyclopedia of Statistical Sciences, Update Volume 3.

[11] Green, P. J., and Silverman, B. W. *Nonparametric Regression and Generalized Linear Models: a Roughness Penalty Approach.* Chapman and Hall, London, 1994.

[12] Gu, C. Cross-validating non-gaussian data. *J. Comput. Graph. Statist. 1* (1992), 169–179.

[13] Gu, C., and Kim, Y.-J. Penalized likelihood regression: general formulation and efficient approximation. *Canadian Journal of Statistics 29* (2002).

[14] Hastie, T., and Tibshirani, R. *Generalized Additive Models.* Chapman and Hall, 1990.

[15] Holmes, C. C., and Adams, N. M. A probabilistic nearest neighbour method for statistical pattern recognition. *Journal Royal Statistical Society B 64* (2002), 295–306.

[16] Jensen, R., Erdogmus, D., Principe, J. C., and Eltoft, T. The laplacian classifier. *IEEE Trans. Signal Processing 55*, 7 (2007), 3262–3271.

[17] Loader, C. *Local Regression and Likelihood.* Springer-Verlag, 1999.

[18] Lu, F., Hill, G. C., Wahba, G., and Desiati, P. Signal probability estimation with penalized likelihood method on weighted data, 2005. Department of Statistics, University of Wisconsin, Technical Report, No. 1106.

[19] O'Sullivan, F., Yandell, B., and Raynor, W. Automatic smoothing of regression functions in generalized linear models. *J. Amer. Statist, Assoc. 81* (1986), 96–103.

[20] Paalanen, P., Kamarainen, J., and Kalviainen, H. http://www.it.lut.fi/project/gmmbayes/, 2007.

[21] Provost, F., and Fawcett, T. Robust classification of imprecise environments. *Machine Learning 42* (2001), 203–231.

[22] Ramoser, H., Muller-Gerking, J., and Pfurtscheller, G. Optimal spatial filtering of single trial eeg during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering 8* (2000), 441–446.

[23] Rasmussen, C. E., 2007. http://www.GaussianProcess.org/gpml/code/index.html.

[24] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[25] Scholkopf, B., and Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[26] Silverman, B. W. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[27] Swets, J. Measuring the accuracy of diagnostic systems. *Science 240* (1988), 1285–1293.

[28] Wahba, G. *Spline Models for Observational Data*. SIAM, 1990.

[29] Wahba, G. Soft and hard classification by reproducing kernel hilbert space methods. *Proc. Nat. Acad. Sciences 99* (2002), 16524–16530.

[30] Wahba, G., Gu, C., Wang, Y., and Chappell, R. Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance, 1993. Department of Statistics, University of Wisconsin, Technical Report, No. 899.

[31] Webb, G. Multiboosting: a technique for combining boosting and wagging. *Machine Learning 40* (2000), 159–196.

[32] Zhang, C.-X., and Zhang, J.-S. Rotboost: a technique for combining roataion forest and adaboost. *Pattern Recognition Letters 29* (2008), 1524–1536.

[33] Zouhal, L., and Denoeux, T. An evidence-theoretic k-nn rule with parameter optimization. *IEEE Trans. Syst. Man Cyber 28* (1988), 263–271.