Classification with Sums of Separable Functions

Jochen Garcke

MATHEON and Technische Universität Berlin
Institut für Mathematik, MA 3-3
Straße des 17. Juni 136, 10623 Berlin, Germany
garcke@math.tu-berlin.de

Abstract. We present a novel approach for classification using a discretised function representation which is independent of the data locations. We construct the classifier as a sum of separable functions, extending the paradigm of separated representations. Such a representation can also be viewed as a low rank tensor product approximation. The central learning algorithm is linear in both the number of data points and the number of variables, and thus is suitable for large data sets in high dimensions. We show that our method achieves competitive results on several benchmark data sets which gives evidence for the utility of these representations.

1 Introduction

We consider the basic binary classification problem, where one starts from a set of labelled data,

$$D = \left\{(\underline{x}_j; y_j)\right\}_{j=1}^{N} = \left\{(x_1^j, \cdots, x_d^j; y_j)\right\}_{j=1}^{N}, \tag{1}$$

with $y_i \in \{-1, 1\}$ labelling the two classes and $\underline{x}$ a d-dimensional feature vector. There exist numerous algorithms for classification (see e.g. [1, 2]). Function based methods for classifying data construct a function $g(\underline{x})$ such that the sign of $g(\underline{x}_j)$ matches y_j for the given data, and the sign of $g(\underline{x})$ correctly predicts y for other $\underline{x}$. Since the data may contain errors, or may simply not provide enough information, one cannot expect to completely satisfy this goal, so one tries to minimise the classification error rates.

An approach using nonlinear functions in high dimensions typically has to address the curse of dimensionality, where the complexity of the function representation, which is the number of unknowns, typically grows exponentially with the dimensions. For example, support vector machines are based on a data centred function representation using kernels. One of the reasons for the success of this approach is that here the dimensionality turns up in the complexity of the kernel computation but not in the complexity of the function representation which essentially only depends on the number of data. One can view other approaches and how they address the curse of dimensionality in a similar fashion, e.g. for neural networks the dimension turns up in the perceptron.

In the last years numerical approaches using function representations based on a sparse tensor product approach were successfully used in high dimensions [3, 4]. These approaches are based, in an abstract fashion, on the approximation by a sum of separable functions,

$$g(\underline{x}) = \sum_{l=1}^{r} s_l \prod_{i=1}^{d} g_i^l(x_i)\,, \tag{2}$$

also known as low rank tensor decomposition or sparse tensor product approximation. The number of terms r is called the *(separation) rank*. Note that the coefficients s_l are solely for later (computational) convenience, so that one can scale the individual functions to $\|g_i^l\| = 1$ in some suitable function norm, e.g. the maximum norm. Since the s_l depend on the g_i^l they are not explicitly learned, but are derived values. If one applies a log to such a function with $r = 1$ the resulting approach is well known in the statistical literature as additive models.

Many methods are based on this formulation but differ in how they use it. Sparse grid methods are based on a multi-scale tensor product basis where the g_i^l are combined from a set of orthogonal functions. Here basis functions of small importance, justified by decay estimates exploiting hierarchical properties, are omitted [4]. In the statistics literature, representations of the form (2) appear under the names "parallel factorisation" or "canonical decomposition", see [5] for a review and further references. They are used primarily to analyse data on a grid, typically in $d = 3$. Since the goal is to interpret data, constraints on g_i^l, such as positivity when one is interested in probabilities, are often imposed. Similarly, since they only describe data on a grid, a general function is not built.

Following [6] we use sums of separable functions of the form (2) but without constraints such as orthogonality or positivity on the g_i^l. The resulting nonlinear approximation method is called a *separated representation* [3]. The functions g_i^l may be constrained to a *subspace*, but are not restricted to come from a particular *basis set*. This extra freedom allows one to find good approximations with surprisingly small r, and reveals a much richer structure than one would believe beforehand. Although there are at present no useful theorems on the size r needed for a general class of functions, there are examples where removing constraints produces expansions that are *exponentially* more efficient than one would expect a priori, i.e. $r = d$ instead of 2^d or $r = \log d$ instead of d. These example are discussed in detail in [3], we will sketch a few here as illustrations.

First, as a simple example, note that in the separated representation one can have a two-term representation

$$\prod_{i=1}^{d} \phi_i(x_i) + \prod_{i=1}^{d} (\phi_i(x_i) + \phi_{i+d}(x_i)) \tag{3}$$

where $\{\phi_j\}_{j=1}^{2d}$ form an orthonormal set. To represent the same function as (3) while requiring all factors to come from a master orthogonal set would force one to multiply out the second term and thus obtain a representation with 2^d terms. Thus a function that would have $r = 2^d$ in an orthogonal basis may be reduced to $r = 2$. Second, consider the additive model $\sum_{i=1}^{d} \phi_i(x_i)$, and note that it is equal to

$$\lim_{h \to 0} \frac{1}{2h} \left(\prod_{i=1}^{d} (1 + h\phi_i(x_i)) - \prod_{i=1}^{d} (1 - h\phi_i(x_i)) \right) . \tag{4}$$

Thus we can approximate a function that naively would have $r = d$ using only $r = 2$ by choosing h small enough for a given approximation error ϵ. This formula provides an example of converting addition to multiplication; it is connected to exponentiation, since one could use $\exp(\pm h\phi_i(x_i))$ instead of $1 \pm h\phi_i(x_i)$. Third, note that Gaussians are separable, since

$$\exp(-c\|\underline{x} - \underline{z}\|^2) = \prod_{i=1}^{d} \exp(-c(x_i - z_i)^2) \, . \tag{5}$$

By expanding a radial function in Gaussians, one can obtain a separated representation for it.

In a data mining context one can view such a representation as a sum of different influences. Consider a case where several effects are responsible for the overall distinction into two classes. Each rank in (2) now plays the role of one such effect. This is a somewhat simplified view, as only products are considered as summands in our approach and the interaction between different attributes will not be just of product nature. On the other hand our algorithm optimises the non-linear separated formulation in all ranks at the same time and has therefore interaction between the different "effects". In the end the goal is not to compute summands describing effects, but to achieve the best compressed representation. As mentioned above, addition can be converted to multiplication (4) and additive representations of certain effects can therefore be represented more efficiently. Also note again, that for $r = 1$ (2) falls back to an additive model. Overall, there is some underlying structure present in machine learning applications which our approach exploits at least implicitly.

The representation (2) provides a rich class of functions from which to construct approximations, while also allowing algorithms that scale linearly in both N and d. It is especially appropriate for the case where the dimension d is large, but the underlying function that generated the data is fairly "simple". One can of course construct functions where the representation (2) would fail, in the sense that r must be very large. It appears, however, that such bad functions do not appear naturally. In some sense such bad functions are excluded by the very nature of the problem we consider. For example, consider a "complicated" function, whose discretisation would require a grid with M points in each direction and, thus, $N = M^d$ samples. Since M^d is impossibly large for even moderate values of M and d, we must assume $N \ll M^d$ and, therefore, the data cannot describe such a function to begin with.

In [6] it was shown that such representations are effective as regression functions. The goals of this paper are to present algorithms to construct classifiers of the form (2), and to give numerical evidence that such representations are worth using as classifiers as well, which from a function point of view possess quite different properties than regression functions.

To adapt the method to the classification problem we replace the least-squares error by other more suited loss functions. We use log-likelihood estimation, which in the sense of probability estimation is more appropriate and statistically sound for classification than a least squares approach [1, 2], and the hinge loss used for support vector machines in a smooth variant introduced in [7].

These loss functions make a different solution strategy necessary. Instead of an alternating least squares procedure, which at its core involves the solution of a linear equation system "living" in one dimension x_i, we now have a non-quadratic function to minimise using non-linear minimisation algorithms. We investigate both an alternating minimising procedure and a global minimisation which optimises in all dimensions simultaneously. For both we need to formulate a suitable regularised problem. This is necessary to avoid overfitting, but also for numerical stabilisation. We extend the approach from [6] and also investigate the use of $\|\nabla g(\underline{x})\|^2$ as a simple regularisation term to enforce smoothness.

In the following Section 2 we describe the regularised minimisation problem while Section 3 contains the employed algorithms. After presenting numerical results in Section 4 we conclude with an outlook.

2 Definition of the Problem

2.1 Loss Function

The representation of a function by sums of separable functions was studied recently for regression [6]. We now adapt this approach for loss functions preferred in classification problems, the overall aim is to minimise the expected classification error on test data. First is the negative log likelihood [1, 2]

$$\frac{1}{N}\sum_{j=1}^{N} L_{LL}(y_j, g(\underline{x}_j)) = \frac{1}{N}\sum_{j=1}^{N} \log\left(1 + \exp\left(-y_j g(\underline{x}_j)\right)\right) . \tag{6}$$

Note that we use the encoding $y_i \in \{-1, 1\}$ for the classes instead of the encoding $y_i \in \{0, 1\}$ often employed in the log likelihood approach.

As an alternative we also adapt the hinge loss used for support vector machines. Since this function is not differentiable at $yg(\underline{x}) = 1$ one would need a general descent method. Instead we use a smooth approximation to it proposed by [7], which is inspired by the Huber loss,

$$L_{HH}(y, t) = \begin{cases} 0 & \text{if } \; yt > 1 + h \\ \frac{(1+h-yt)^2}{4h} & \text{if } \; |1 - yt| \leq h \\ 1 - yt & \text{if } \; yt < 1 - h \end{cases} \tag{7}$$

where h is a parameter to be chosen, typically between 0.01 and 0.5; for $h = 0$ one obtains the hinge loss. Therefore one minimises in this case

$$\frac{1}{N}\sum_{j=1}^{N} L_{HH}(y_j, g(\underline{x}_j)). \tag{8}$$

Note that we are not actually minimising the hinge loss, but from a machine learning point of view there is, besides tractability, no reason to prefer the hinge loss over a smoothed version [7]. In other words, if one does not gain algorithmic advantages using the hinge loss, like for support vector machines in the dual optimisation, one can use a huberised version and expect comparable results.

In the following we describe our approach for the negative log likelihood (LL) loss function (6), but one can replace it at every step with the huberised hinge (HH) loss function (8).

2.2 Basis in One Dimension

Up to now we have not specified the representation for the one-dimensional functions g_i^l. For the above, and most of the following description, it is enough to assume that we are given a function space of (finite) dimension M_l in which to search for g_i^l. For example, one could choose polynomials of some degree, splines, or piecewise linear functions. This space may be different for each term l in the sum, each attribute i, and in general also for each (l, i) pair. We next choose some basis $\{\phi_k^l\}_{k=1}^{M_l}$ for this function space, but we emphasise that the main results are independent of the particular choice. The function g_i^l will be represented by the vector of its M_l coefficients $c_i^l(k)$ in its expansion into $\{\phi_k^l\}$:

$$g_i^l(x_i) = \sum_{k}^{M_l} c_i^l(k)\phi_k^l(x_i). \tag{9}$$

In our numerical experiments in Section 4 we use a multi-scale basis of tent functions on the interval $[0, 1]$, as was used e.g. in [6, 8]. On level 0 this consists of the functions 1 and x. On level 1 we additionally include the tent function of support 1 centred at 1/2, i.e. the line segments from $(0, 0)$ to $(1/2, 1)$ and then to $(1, 0)$. Level 2 adds two tent functions of width 1/2, centred at 1/4 and 3/4, etc. This function space consists of piecewise linear functions.

We will solve for the values of $c_i^l(k)$ for all i, l and k, so those are the free parameters with respect to which we minimise the error.

2.3 Avoiding Over-Fitting

There are two ways in which over-fitting can occur here. The first is when r is too large. Since r is the main complexity parameter, it is natural to take a parametric approach and choose r very low. As with all parametric methods, various more-or-less justified tests, or simple cross-validation, can be used to choose the appropriate r.

The second way over-fitting can occur is when there is over-fitting in the one-dimensional functions g_i^l. There are two natural ways to avoid over-fitting within this framework. One is again to use a parametric approach, and choose M small. Note that the discrete function space and the choice of its resolution can also be viewed in the context of regularisation by projection [9, 10].

The other way is to use a nonparametric approach and incorporate regularisation to encourage smoothness, as we describe next. We will use two different strategies for regularisation, one in regard to the full function g, and one in regard to the one-dimensional factors g_i^l. Furthermore, regularisation is usually also beneficial for the stability of the employed numerical solution strategy.

Global Regularisation One possibility is to add a weighted regularisation term to the loss function (6) or (8) which results in a new functional for the minimisation

$$\frac{1}{N}\sum_{j=1}^{N} L(y_j, g(\underline{x}_j)) + \lambda\|\mathcal{S}(g(\underline{x}))\|^2. \tag{10}$$

The particular choice of $\mathcal{S}$ depends on the chosen discrete function space, or, viewed in the kernel context, the particular choice of $\mathcal{S}$ corresponds, under certain conditions, to a reproducing Kernel Hilbert space (RKHS) and therefore defines a function space (which is then discretised). The regularisation parameter λ has to be chosen suitably as usual. Since we employ multi-scale linear functions as our basis for $g(\underline{x})$ we can only use first derivatives in $\mathcal{S}$. Therefore we use $\|\nabla g(\underline{x})\|^2$ as a simple regularisation term. Although this does not define a RKHS, it was shown to be a reasonable choice in [11, 12].

Regularisation of One Factor We will see in section 3.1 that for the alternating minimisation procedure the problem collapses to one-dimensional subproblems in coordinate direction x_i. Here one can use the global regularisation term from the last section. But one can also encourage smoothness of the function $g(\underline{x})$ just in regard to direction x_i and assume here for the minimisation that the other components have no influence on the smoothness of the function. Furthermore one enforces regularisation separately for each summand $g_i^l(x_i)$ in (2) and not combined. Note that we present the resulting form of regularisation in the following to completely formulate the problem setup. That this choice of the regularisation term has in particular numerical advantages will be clearer after the study of section 3.1, where the alternating minimisation procedure for the one-dimensional problems is explained.

The one-dimensional function $g_i^l(x_i)$ from (2) is using the basis functions ϕ_k^l – we assume the same basis in all dimensions here – and will be represented by M_l coefficients $c_i^l(k)$ according to (9). One now chooses a list of penalty weights γ_k^l and adds to the one dimension problem (13)

$$\lambda \sum_l s_l^2 \sum_k \gamma_k^l |c_i^l(k)|^2 \, . \tag{11}$$

This approach was taken in [6] with a particular choice of weights. We use a slight modification and choose the weights for one $g_i^l(x)$ by $\|\mathcal{S}(g_i^l(x))\|^2$. Due to the orthogonality of the basis we have $\int {\phi_k^l}'(x){\phi_{\tilde{k}}^l}'(x)dx = \delta_{k\tilde{k}}C(k)$, where the constant $C(k)$ depends on the size of the support of ϕ_k^l. Due to the choice of

$\|\mathcal{S}(\phi_k(x))\|^2$ basis functions with small support will be penalised stronger than those with large support. This will penalise large local variance much stronger than a change of the function over larger intervals.

Let us remark, that in the least-squares case the minimisation problem can be ill-posed [13], but if each coefficient is penalised by some value larger than zero the problem becomes well-posed; see [3] for discussion on controlling condition number in this way. We conjecture that the situation is similar when minimising log likelihood or the huberised hinge loss.

2.4 Sums of Separable Functions in the Learning Theory Context

This approach fits into the framework of Sobolev spaces, these were studied in the learning theory context for example in [14]. This then gives us an infinite function space, with bounds on its properties for learning, in which a discrete approximation takes place, in our case by the use of sums of separable functions.

Such an approximation of an element from a function space by a linear combination of functions from a given dictionary is much less studied in learning theory. From the perspective of approximation theory in [15] bounds for convergence rates for the problem of approximating a given function f from a Hilbert space H by means of greedy algorithms are given and applied to the statistical learning theory context.

In regard to approximation properties of our approach, some results are given in [3] which show how other approaches can be formulated in the form of (2), and how with increasing number of ranks r and increasing resolution M one can approximate a function from a Sobolev space of certain smoothness arbitrarily close. But the convergence order for these somewhat related approaches grows exponentially in d.

To give theoretical results for our approach a characterisation of functions with low separation rank (or tensors with a low rank decomposition) is needed, but currently there is no characterisation of this kind. The examples above and in [3] as well as the successful use of the related "parallel factorisation" or "canonical decomposition" in statistics show that there are surprising mechanisms that allow low separation rank. At this stage the lack of a complete theory should not prevent a study of sums of separable functions for learning.

3 Minimisation Procedures

There are many algorithms for solving least-squares problems using representations like (2) described in the literature, see [5, 16] for surveys. It is a non-trivial problem and there is active research to improve convergence and reduce the dependence on the starting guess.

These algorithms can be classified in three main groups: alternating algorithms, which update only a subset of the unknowns at each step; derivative-based methods, seeking an update for all the parameters simultaneously by successive approximations; and direct (non-iterative) methods. The latter cannot be applied in our setting.

3.1 Alternating Minimisation Procedure

For the least squares error this approach is well-known as alternating least-squares (ALS). The idea of partitioning the space of unknowns and solving the optimisation alternatingly in these partitions is known under several other names like coordinate descent and goes back at least to [17]. In the following we will call it alternating minimisation procedure (AMP).

Collapse to One-Dimensional Subproblems We now assume that an initial guess g of the form (2) is given, with some choice of representation for g_i^l. We fix the components in all directions but one, and so collapse to an one-dimensional problem. For simplicity we describe the case for direction $i = 1$, and so fix g_i^l for $i > 1$. We define the (fixed) partial products from the remaining directions by

$$p_j^l = s_l \prod_{i=2}^{d} g_i^l(x_i^j)\,, \qquad l = 1, \ldots, r, \quad j = 1, \ldots, N\,. \tag{12}$$

The loss (6) then reduces to

$$\frac{1}{N}\sum_{j=1}^{N} \log\left(1 + \exp\left(-y_j \sum_{l=1}^{r} p_j^l g_1^l(x_1^j)\right)\right)\,. \tag{13}$$

To minimise (13) we must solve a one-dimensional non-linear problem involving r one-dimensional functions g_1^l, each described by M_l coefficients. As a minimiser one has the choice under several algorithms. We did experiments with the quasi-Newton method BFGS, a non-linear CG-method, and a trust-region method (see e.g. [18]). One could numerically estimate the needed derivatives (and hessian for the trust-region method) of the loss function. But e.g. the derivative for the log-likelihood with respect to $c_1^l(k)$ can be given explicitly as

$$\frac{1}{N}\sum_{j=1}^{N} -y_j s_l \phi_k^l(x_1^j) \frac{\exp\left(-y_j \sum_{l=1}^{r} p_j^l g_1^l(x_1^j)\right)}{1 + \exp\left(-y_j \sum_{l=1}^{r} p_j^l g_1^l(x_1^j)\right)}\,, \tag{14}$$

and the derivative of the regularisation terms is straightforward.

The minimisation finishes once a suitable stopping criteria for the employed non-linear solver is fulfilled, e.g. the objective function does decrease smaller than a given threshold. Since we have an outer iteration the stopping criteria for the one-dimensional minimisation can be coarser than typically used. We then re-normalise g_1^l and incorporate the norm into s_l, this is not strictly necessary, we need not normalise at all; we do so only to prevent over/under-flows.

Before we describe the full algorithm including the iteration over the dimensions we now consider the computational cost bounds; exemplary for the BFGS-method. Here we assume $M_l = M$ for all l and that the cost to evaluate ϕ_k^l is $\mathcal{O}(1)$, which is the case for our choice of basis functions. Therefore the cost to evaluate a single g_i^l at a single point is $\mathcal{O}(M)$. The computation count for one iteration of BFGS is $\mathcal{O}(r^2M^2)$ plus the costs for the evaluation of the loss function and the gradient for the iteration update and in particular the line search [18]. Given the p_j^l, it costs $\mathcal{O}(rMN)$ to compute the loss function (13). To evaluate (14) we compute the fraction

$$\frac{\exp\left(-y_j\sum_{l=1}^r p_j^l g_1^l(x_1^j)\right)}{1+\exp\left(-y_j\sum_{l=1}^r p_j^l g_1^l(x_1^j)\right)} \tag{15}$$

once for each j, again in $\mathcal{O}(rMN)$. Using that (now fixed) value we compute the derivative with respect to a single $c_1^l(k)$ in $\mathcal{O}(N)$. Since we have rM different $c_1^l(k)$, the total complexity for the computation of the derivatives for the loss part is $\mathcal{O}(rMN)$.

The two regularisation alternatives have different costs. To compute the global regularisation for the regularised loss (10) one needs $\mathcal{O}(r^2M^2)$ operations. The prime of the regularisation needs $\mathcal{O}(r^2M)$ each time, it simplifies for the employed multi-scale linear basis. The contribution of the other dimensions to the regularisation term in (10) can be computed once at the beginning of the minimisation and needs $\mathcal{O}(d^2r^2M^2)$ If we denote the number of BFGS iterations by S and take it all together the cost to minimise the one-dimensional problem with the global regularisation is

$$\mathcal{O}((r^2M^2+rMN)S+d^2r^2M^2). \tag{16}$$

The simpler regularisation term (11) only needs $\mathcal{O}(rM)$ operations for the evaluation of the regularised loss function and its prime. Which gives a total cost of

$$\mathcal{O}((r^2M^2+rMN)S). \tag{17}$$

Alternating Improvement If we can solve the one-dimensional subproblems, then we can iteratively solve such problems to reduce the loss (6). The alternating strategy [3, 5, 16] is to loop through the directions $i = 1, \ldots, d$. One then repeats this alternating process and monitors the change in the loss (6), or the regularised loss (10), to detect convergence. It is certainly possible to hit local minima. Even when we approach the true minima, we have no reason to expect any better than linear convergence.

To account for the computational cost to set up these problems we assume that the number of AMP iterations is K. The cost to compute all p_j^l for a single i is then $\mathcal{O}(rdMN)$. It would appear that we have cost $\mathcal{O}(rd^2MNK)$ in the outer loop through the d directions. However, when we switch from, say, $i = 1$ to $i = 2$, we can simply update p_j^l by multiplying it by $g_1^l(x_1^j)/g_2^l(x_2^j)$, at cost $\mathcal{O}(rMN)$. The total cost for the update in the AMP formulation (without the cost for solving the one-dimensional subproblems) is thus

$$\mathcal{O}(drMNK)\,. \tag{18}$$

If we incorporate this algorithm into the overall method and account for the total cost we get

$$\mathcal{O}(K(dr^2M^2 + drMN)S). \tag{19}$$

for the local regularisation from Section 2.3. The cost is linear in both d and N, and so the method is feasible for large data sets in high dimensions.

Using the global regularisation from Section 2.3 we observe a complexity of

$$\mathcal{O}(K[(dr^2M^2 + drMN)S + d^3r^2M^2]). \tag{20}$$

Again linear in N, but in parts cubic in d, although the inner non-linear solver is linear in d. Nevertheless, the complexity still suggests the method for large data sets in high dimensions.

The computational complexity for a non-linear CG-method or the trust-region method in regard to N and d are similar. Only the evaluation of the regularised loss functions, their prime and hessian (for the trust-region method) depends on these and there we have a linear scaling in N and d for the regularisation (11) and linear in N and cubic in d for the regularisation (10).

The number of iterations needed in the non-linear solvers is the remaining important computational aspect. These will depend implicitly on the complexity of the function and therefore on the number of data. At this point we have not investigated the non-linear solvers in detail but use well tested and publically available implementations. Line-search procedures adopted to the problem and suitable pre-conditioners for the non-linear CG approach are needed for a fully efficient scheme. But for now we focus on the investigation of the accuracy and representation power of our approach for classification problems. Therefore it is enough to solve the non-linear problems to a sufficient degree in reasonable time.

Also note that we expect that after a few steps of the alternating procedure we will have good starting values for the non-linear minimisation.

3.2 Global Minimisation Procedure

Alternating algorithms are in particular attractive for the least square loss because at their inner core a linear equation system needs to be solved. We here use other loss functions and therefore have to use a non-linear minimisation procedure anyway, therefore one can consider treating the full problem (10) directly, as it is also often used for least squares minimisation [5, 16]. In the following we will call it global minimisation procedure (GMP).

The amount of data and the dimensionality now have a different influence on the computational complexity. Again we focus on the BFGS-algorithm. The number of unknowns of our representation is drM, therefore the cost for one BFGS-iteration is of the order $\mathcal{O}(d^2r^2M^2)$ plus the cost for evaluating the regularised loss function (10) and its prime. It costs $\mathcal{O}(drMN)$ to evaluate the loss function and $\mathcal{O}(d^2r^2M^2)$ for the regularisation term. The same holds for one partial derivative of which there are drM. In total we have for the complexity of the algorithm

$$\mathcal{O}((d^2r^2M^2N + d^2r^3M^3)S), \tag{21}$$

where S is the number of BFGS-iterations.

Although the global minimisation procedure has the larger order of computational complexity it can be competitive if the number of iterations S is small. But for this often special care has to taken in the line search procedure. Furthermore, implementing such an algorithm is often more cumbersome, especially in regard to the needed derivative. As we will see in the numerical results using standard implementations of non-linear solver does not achieve the wanted accuracy. Here further investigation of the procedures and their adaptation to the particular loss function and problem setup are necessary, but out of scope of this paper.

4 Numerical Results

In this section we give numerical results for several benchmark problems. Our goal is to demonstrate that the representation (2) is powerful enough to build good classifiers.

We compare against data used in the benchmark study [19], where the classification methods support vector machines with RBF-kernel (svm), classification trees, linear discriminant analysis, quadratic discriminant analysis, neural networks, generalised linear models (glm), multinomial logit models, nearest neighbours (nn), learning vector quantisation (lvq), flexible discriminant analysis, mixture discriminant analysis, bagging, double bagging (dbagg), random forests (rForst), and multiple additive regression trees were compared empirically[1].

As in [19], we measure the classification performance using the prediction error. Ten-fold cross-validation was performed ten-times; we report the means and medians of the test set error rates of all 100 runs, whereas the standard deviation and inter-quartile range are computed with regard to the ten-fold results. For comparison we give the best result from the benchmark study and note the rank of our approach in comparison to the other methods used.

The separation rank r, the discretisation level of the multi-scale basis (which correlates with the basis size M) and the size of the regularisation parameter were selected similar to [19]: we split the training data 2:1, train on the first two thirds and evaluate on the last third to select good parameters. With these we learn on all training data and evaluate on the as-yet-unseen test data. Note that depending on the problem we used up to level 4 of the multi-scale basis and rank $r = 7$, although often $r \leq 4$ was sufficient.

It was observed that the test error is relatively unaffected by the value of h in the huberised hinge loss, as long as it is not too large when it resembles more the L_2 loss [7]. In our experiments we observed this behaviour as well and use a fixed $h = 0.05$.

Pre-processing of the data consists of omitting missing values, like in [19], and scaling all data to $[0, 1]^d$. We concentrate in this paper on data sets with metric attributes. Therefore we use the five synthetic data sets and five real ones,

[1] The data is available from (`http://www.ci.tuwien.ac.at/~meyer/benchdata/`).

Table 1. Results on low dimensional synthetic data sets for both loss functions and (A)lternating and (G)lobabl minimisation procedures. We give the mean (with standard deviation) and median (with inter-quartile range).

	CIRCLE					SPIRALS			
	LL - A	LL - G	HH - A	HH - G		LL - A	LL - G	HH - A	HH - G
mean	**2.22**	2.66	2.45	3.65	mean	**0.25**	0.53	1.03	2.79
	(0.46)	(0.39)	(0.47)	(0.47)		(0.10)	(0.28)	(0.22)	(0.57)
median	**1.95**	2.35	2.30	3.45	median	**0.20**	**0.20**	0.75	2.30
	(0.61)	(0.60)	(0.80)	(0.49)		(0.16)	(0.44)	(0.29)	(0.93)

including one with some categorical variables which were transformed into binary attributes. For binary attributes i we only use a linear function, i.e. $M_i = 2$.

4.1 Alternating and Global Minimisation

First we remark on the empirical behaviour of the two different minimisation procedures. We consider the circle in a square (CIRCLE) and the two noisy spirals (SPIRALS) data sets from [19], both are two dimensional. One might expect that in this low dimensional case there would not be too much difference in the behaviour of the two minimisation procedures. But already here the global optimisation procedure does not cope with the problem and produces worse results. This does not depend on the employed non-linear solver.

To be more precise. Using the same minimisation procedure in a standard implementation using standard line search procedures the alternating minimisation procedure achieves better results than the global minimisation procedure. This observations does not depend on the non-linear solver nor which public implementation was used. We expect that with a detailed investigation of the minimisation problem and an adaption of the line search procedure the global minimisation will perform better. There is active research in this regard in the least-squares context [5, 16, 20].

In any case, not only does the alternating minimisation procedure show the better order of computational complexity, it also achieves better correctness rates as we can see in Table 1.

Somewhat promising are further results using a few iterations of the alternating procedure to compute a good starting point for the global minimiser. For the spiral data set we achieve in this way a median of 0.1 (0.075) and mean of 0.22 (0.13) using the log-likelihood as a loss function. Although the huberised hinge loss does not benefit from such an approach for this data set; nor do we observe such an improvement for the circle data set.

For higher dimensional data sets the global minimisation performs, as one would expect after these results, even worse. We therefore abstain form giving detailed results for the global minimisation procedure in the following.

We also use the two dimensional spiral data set to illustrate the approach. In Figure 1 we show the components $g_i^l(x_i)$ of the solution for one instance of

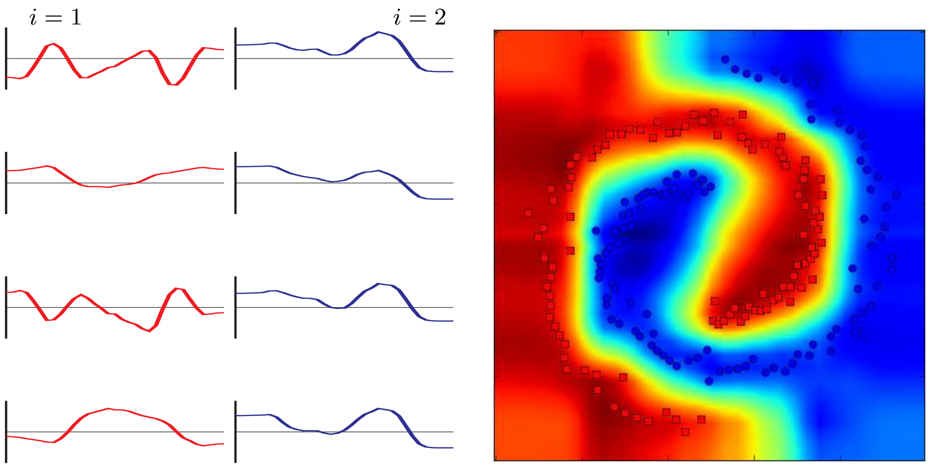

Fig. 1. Classifier with $r = 4$ using the multi-scale basis with level five produced using the negative log likelihood error on one instance of the spiral data set. Left: The function of the form (2) with $r = 4$ using the multi-scale basis with level five. Each subplot shows a $g_i^l(x_i)$. The magnitude s_l has been distributed. Right: Value of the classifier over the two-dimensional domain.

the data set and the resulting classifier. One might not expect that with just the sum of four product functions such a complicated classifier can be obtained.

4.2 Results on Benchmark Data

We give the results of our experiments in Table 2 for the synthetic data sets and Table 3 for the real ones. Our current code is a somewhat experimental python implementation, whose purpose is to produce the approximation results that are the main point of this paper. For one run on a data set the computational time varied between a few seconds and a few hundred seconds, depending on the data set, the rank, the degree and the regularisation parameter. In python, loops with numerical computations are known to be inefficient and the algorithm consists of a fair amount of loops over the data or the basis functions. We expect that a proper implementation would decrease the runtime significantly. Therefore we abstain for now from giving more detailed numbers for the computational times, but will followup once a scaleable implementation is available.

Overall the log likelihood estimation performs somewhat better than the huberised hinge loss, the latter might benefit from the h in its definition (7) chosen depending on the data. The run time difference between these two loss functions in our experiments did not appear to be significant.

The simple local regularisation method performs slightly better than the global regularisation. For the twonorm, bupa liver and credit data set it results in smaller misclassification rates, while for the spirals and threenorm data set the

Table 2. Results on synthetic data from the study [19]. We give the mean (with standard deviation) and median (with inter-quartile range) for the best results from [19], our approach, and our rank in comparison to all 17 approaches used. We use log likelihood estimation (LL), the huberised hinge loss (HH) and both forms of regularisation.

			LL-Reg. (11)		HH-Reg. (11)		LL-Reg. (10)		HH-Reg. (10)	
data set	best	other	SumSep	rank	SumSep	rank	SumSep	rank	SumSep	rank
mean	2.66	svm	2.26	1	2.39	1	**2.22**	1	2.45	1
CIRCLE			(0.44)		(0.44)		(0.46)		(0.47)	
median	2.50	svm	2.00	1	2.10	1	**1.95**	1	2.30	1
	(0.49)		(0.41)		(0.62)		(0.61)		(0.80)	
mean	0.17	nn	1.04	3	1.19	3	**0.25**	2	1.03	3
SPIRALS			(0.18)		(0.22)		(0.10)		(0.22)	
median	0.10	nn	0.90	3	1.10	3	**0.20**	2	0.75	3
	(0.07)		(0.22)		(0.31)		(0.16)		(0.29)	
mean	2.82	svm	**3.61**	5	4.08	5	12.04	17	6.37	12
TWONORM			(0.34)		(0.34)		(7.85)		(1.34)	
median	2.70	svm	**3.40**	5	3.85	5	5.50	10	5.90	11
	(0.20)		(0.40)		(0.54)		(8.93)		(0.59)	
mean	14.17	lvq	18.94	9	19.21	9	**14.45**	2	15.62	2
THREENORM			(0.98)		(0.60)		(0.40)		(0.69)	
median	13.70	lvq	18.95	8	19.10	9	**14.40**	2	15.40	2
	(0.77)		(0.98)		(0.50)		(0.79)		(1.22)	
mean	3.58	svm	5.44	2	5.92	2	**4.85**	2	5.43	2
RINGNORM			(0.58)		(2.64)		(0.31)		(0.32)	
median	2.90	svm	4.80	2	4.90	2	**4.70**	2	5.30	2
	(0.70)		(0.75)		(0.61)		(0.33)		(0.31)	

global regularisation method is better. In particular for some of the real data the minimisation procedure had problems to converge for the global regularisation; again, better adapted non-linear solution strategies might improve the results. The good performance of the simpler local regularisation is an indication that the limitation to a discrete function representation has a large effect in the avoidance of overfitting, known as regularisation by projection in other fields [9, 10].

In comparison to the 17 other methods our approach achieves very competitive results, here we look at the median as is done in [19]. For all data sets at least one variant of our approach is in the top five and for eight of the data sets we are in the top three. For six data sets, the majority, at least one version of our procedure achieved better results than a support vector machine, which was the best method in the referenced study [19].

5 Outlook

We described a new classification algorithm using sums of separable functions to represent the classifier. Numerical evidence shows that typical data sets can be efficiently described by this approach, which is the main message of this paper.

Table 3. Results on real data from the study [19]. We give the mean (with standard deviation) and median (with inter-quartile range) for the best results from [19], our approach, and our rank in comparison to all 17 approaches used. We use log likelihood estimation (LL), the huberised hinge loss (HH) and both forms of regularisation.

data set		best	other	LL-Reg. (11) SumSep	rank	HH-Reg. (11) SumSep	rank	LL-Reg. (10) SumSep	rank	HH-Reg. (10) SumSep	rank
CANCER	mean	2.28	rForst	3.30 (0.24)	5	**3.21** (0.26)	4	3.33 (0.22)	5	3.66 (0.49)	6
	median	1.49 (0.16)	rForst	2.94 (0.25)	4	**2.92** (0.33)	3	**2.92** (0.34)	3	2.94 (0.71)	4
LIVER	mean	27.04	rForst	26.63 (1.48)	1	**26.58** (0.98)	1	31.74 (1.50)	8	30.66 (1.86)	7
	median	27.02 (2.15)	rForst	**25.71** (2.52)	1	**25.71** (1.48)	1	30.56 (2.80)	6	30.56 (3.24)	6
CREDIT	mean	22.65	dbagg	**23.70** (0.67)	9	24.73 (0.87)	10	27.11 (1.40)	11	26.43 (1.10)	11
	median	22.77 (0.20)	5 appr.	**22.77** (0.57)	1	24.25 (0.87)	10	26.87 (2.40)	13	26.73 (0.72)	12
IONOSPHERE	mean	5.93	svm	**8.01** (0.84)	4	9.03 (0.63)	6	16.11 (1.67)	15	11.49 (2.23)	8
	median	5.71 (0.70)	2 appr.	**8.57** (1.35)	4	**8.57** (0.95)	4	**8.57** (3.21)	4	**8.57** (4.32)	4
DIABETIS	mean	22.37	2 appr.	23.37 (0.59)	5	23.35 (0.52)	5	**23.08** (0.71)	5	24.10 (0.73)	9
	median	22.08 (0.26)	4 appr.	**22.08** (0.65)	1	23.23 (0.80)	5	22.72 (0.94)	5	23.38 (0.41)	6

There are several extensions and generalisations possible. Foremost, instead of the standard and publically available non-linear minimisation algorithms a method more specifically tuned to the problem could be used, e.g. similar to [20, 21]. Second, the approach can easily be extended for categorical attributes x_j. One can use a basis of vectors rather than functions, and index their coordinates by the categories of such an attribute. Third, if one formulates the loss (13) using something besides negative log likelihood or the hinge loss used in support vector machines one obtains a similar nonlinear optimisation problem which can be treated analogously. Fourth, note that in [6] it was shown how to extend the regression algorithm to vector-valued regression functions. By letting the vector represent the probabilities of the data point being in the different classes, one obtains a multi-class classifier that produces probabilities rather than a decision on the class. Suitable multi-class loss functions for support vector machines or penalised likelihood estimation can be found in [22]. Finally, one can use different one-dimensional spaces for different attributes and for different l. For example, if one assumes an almost normal distribution underlying the data, one might use a suitable Gaussian for the $l = 0$ term, but another basis for other l to approximate necessary adjustments.

Acknowledgements The author cordially thanks Martin Mohlenkamp (Ohio University) for helpful discussions and suggestions.

References

1. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
2. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2001)
3. Beylkin, G., Mohlenkamp, M.J.: Algorithms for numerical analysis in high dimensions. SIAM J. Sci. Comput. **26**(6) (July 2005) 2133–2159
4. Bungartz, H.J., Griebel, M.: Sparse grids. Acta Numer. **13** (2004) 147–269
5. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3) (September 2009) 455–500
6. Beylkin, G., Garcke, J., Mohlenkamp, M.J.: Multivariate regression and machine learning with sums of separable functions. SIAM Journal on Scientific Computing **31**(3) (2009) 1840–1857
7. Chapelle, O.: Training a support vector machine in the primal. Neural Computation **19**(5) (2007) 1155–1178
8. Garcke, J., Griebel, M., Thess, M.: Data mining with sparse grids. Computing **67**(3) (2001) 225–253
9. Engl, H., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Kluwer (1996)
10. Natterer, F.: Regularisierung schlecht gestellter Probleme durch Projektionsverfahren. Numer. Math. **28** (1977) 329–341
11. Garcke, J.: Regression with the optimised combination technique. In Cohen, W., Moore, A., eds.: Proceedings of the 23rd ICML '06, New York, NY, USA, ACM Press (2006) 321–328
12. Garcke, J., Hegland, M.: Fitting multidimensional data using gradient penalties and the sparse grid combination technique. Computing **84**(1-2) (April 2009) 1–25
13. de Silva, V., Lim, L.H.: Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM J. Matrix Anal. Appl. **30**(3) (2008) 1084–1127
14. Cucker, F., Smale, S.: On the mathematical foundations of learning. Bulletin of the AMS **39**(1) (2001) 1–49
15. Barron, A.R., Cohen, A., Dahmen, W., Devore, R.A.: Approximation and learning by greedy algorithms. Ann. Stat. **36**(1) (2008) 64–94
16. Tomasi, G., Bro, R.: A comparison of algorithms for fitting the parafac model. Computational Statistics and Data Analysis **50**(7) (2006) 1700 – 1734
17. Yates, F.: The analysis of replicated experiments when the field results are incomplete. Emp. J. Exp. Agric. **1** (1933) 129–142
18. Nocedal, J., Wright, S.J.: Numerical optimization. 2nd ed. Springer (2006)
19. Meyer, D., Leisch, F., Hornik, K.: The support vector machine under test. Neurocomputing **55** (2003) 169–186
20. Espig, M.: Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen. PhD thesis, Universität Leipzig (2008)
21. Espig, M., Hackbusch, W., Rohwedder, T., Schneider, R.: Variational calculus with sums of elementary tensors of fixed rank. Numerische Mathematik (2010, submitted) Preprint 52/2009, MPI for Mathematics in the Sciences.
22. Wahba, G.: Soft and hard classification by reproducing kernel Hilbert space methods. Proc. Natl. Acad. Sci. USA **99**(26) (2002) 16524–16530