Efficient Supervised Learning with the DD$\alpha$-classifier

Tatjana Lange$^1$, Karl Mosler$^2$, Pavlo Mozharovskyi$^2$

$^1$ Hochschule Merseburg, Geusaer Straße, 06217, Merseburg, Germany
$^2$ Universität zu Köln, Albertus-Magnus-Platz, 50923, Köln, Germany

tatjana.lange@hs-merseburg.de, {mosler,mozharovskyi}@statistik.uni-koeln.de

Abstract. To classify multidimensional data into $q \geq 2$ classes the new DD$\alpha$ classifier is proposed. This classifier first employs a proper data depth to represent the data by their DD-plot, which is a subset of the $q$-dimensional unit cube, and then separates the classes on this cube by means of a slightly modified alpha-procedure [8]. Specifically, we use two depth functions that are well computable in high dimensions, the zonoid depth [12] and the random Tukey depth [13]. The behavior of the procedure is explored with simulated as well as real data. The training of the DD$\alpha$-classifier proves to be faster than that of many known classification methods, while the error rates are of similar size.

Keywords
Pattern Recognition, Data Depth, $\alpha$-procedure, DD$\alpha$, Regularization

1 The task of pattern recognition

The basic task of pattern recognition consists in constructing a decision rule by which objects can be assigned to one of two given classes. The objects are characterized by a certain number of real-valued properties. The decision rule is based on a trainer’s statement that states for a training sample of objects, whether they belong to class $V_1$ or class $V_2$. Tab.1 shows an example of such a training sample.

<table>
<thead>
<tr>
<th>Object number</th>
<th>Properties</th>
<th>Assignment by trainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x_{11}$</td>
<td>$V_2$</td>
</tr>
<tr>
<td>2</td>
<td>$x_{21}$</td>
<td>$V_1$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$i$</td>
<td>$x_{ij}$</td>
<td>$V_1$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$l-1$</td>
<td>$x_{il}$</td>
<td>$V_2$</td>
</tr>
</tbody>
</table>

Note: Sometimes, we can also use combined properties $p_{(m+k)}$ which are constructed from original properties $p_q$ and $p_r$, e.g. $p_{(m+k)} = p_{(m+k)}(p_1, p_5) = p_1 p_5^2$

We search a separating hyperplane or a separating hypersurface for the initially measured and classified data (the training set) and we hope that this plane (or surface) will afterwards work automatically for other measured data and define without the trainer to which class $V_1$ or $V_2$ the object $x_{i,k}$ belongs.
2 Data depth and DD-plot for classification tasks

The solution of the problem that is proposed in this paper is based on the following three “corner stones”:

1. the data depth (DD),
2. the DD-plot,
3. the $\alpha$-procedure together $\text{DD}_{\alpha}$.

2.1 Data depth

The notion of data depth was proposed by Tukey [1] with the intention to avoid the following problem that we sometimes meet when we use the traditional parametric evaluation methods. This problem consists in the following:

a) When we know the density of a random variable together with the values of its distribution parameters, we can calculate its probability of taking values within a given interval. Thus, it is easy to construct the histogram from the distribution density (see fig. 2.1.1). Let’s consider this task as the “direct task”.

![Fig. 2.1.1: Direct task – from distribution density to histogram](image)

b) If we are aware of the parametric class of the distribution density but its parameters are unknown to us, it is also not very hard to estimate the parameters when we know the corresponding histogram that is based on a sufficient number of measured data.

c) But if we cannot assume a specific parametric class and if we have only a “small” number of data, then the inverse task of constructing the density function from the available small number of data may become an incorrect task as defined by Hadamard [2]: We may miss the right functional dependencies because our search “granularity” is too coarse and we may estimate the parameters of a completely wrong distribution (see fig 2.1.2).

![Fig. 2.1.2: Inverse task – from histogram of measured data to distribution density](image)

1) That means, we cannot resort to asymptotics.
This is why Tukey proposed a completely different approach to the description of random variables – the non-parametric data depth:

The data depth example for just one variable shown in fig. 2.1.3 is probably not self-explaining enough. The nature of the data depth may become clearer with the help of the fig. 2.1.4 that shows an example for constructing the data depth for two variables.

Fig. 2.1.4: Random Tukey depth [1] for two variables

Fig. 2.1.5 shows us finally the data depth for two variable (two features, $d = 2$) and two classes ($0 = 2$).
It seems that we could directly use such a data representation for classification tasks by simply comparing the depth of a concrete data point in the two classes. But it appears that such an approach is not effective enough.

2.2 DD-plot

To construct a DD-plot, the original $d$-dimensional feature space is transformed into a $d$-dimensional space of $\theta$ classes ($\theta \leq d$). For example, the space of two features ($d = 2$) shown in fig. 2.2.1 is transformed into the space of two classes ($\theta = 2$) that is shown in fig. 2.2.2.

Again, we could perform the separation in the new space simply on the basis of the maximum depth. That means a point would be classified as belonging to that class in which it has maximum depth [3,4]. In this case the depth space would be partitioned into $\theta$ parts which are separated by $\theta$ secant hyperplanes. Such a method that is based on the maximum depth is a linear classification rule.

A non-linear classification rule was used, for example, in [5], where a separating polynomial line up to $3^{rd}$ degree was constructed for the space $[0,1]^2$, i.e. for the case $\theta = 2$ (see [6,7]).
One of the difficulties of these approaches appears when data points are located close to the zero-depth of one of the classes, i.e. when we have to deal with so-called outliers.

2.3. DDa- and $\alpha$-procedure

Let us, for example, consider a task, where we have to find a separating rule for two classes ($\theta = 2$) within a $d$-dimensional feature space. (This corresponds to the 2-dimensional DD-plot.)

This task can be solved with the $\alpha$-procedure [8] but also with the DDa-procedure.

All methods of pattern recognition, and thus also the $\alpha$-procedure and the DDa-procedure, start with the table of "measured" data that is shown in tab. 1.

In this table of original data the columns contain the values of the properties $p_1, p_2, \ldots$ of different objects. Each row represents one object, i.e. the row $i$ contains the values $x_{i1}, x_{i2}, \ldots$ of the properties $p_1, p_2, \ldots$ of the object $i$, where $i = 1, 2, 3, \ldots, l$.

$l$ is the number of objects of both classes $V_1$ and $V_2$ together, i.e. the number of data points in the $d$-dimensional space. The dimension of the property space is normally much higher than the number of searched classes, $d >> \theta$.

The common $\alpha$-procedure works directly with this table of original data, i.e. it takes the property values from the columns as they are.

The DDa-procedure differs from the $\alpha$-procedure in the following way: At the beginning it calculates the data depths of all objects with regard to the training classes and then works (instead of the original property values $x_{i1}, x_{i2}, \ldots$) with the corresponding pairs of data depths ($D_1(x_i), D_2(x_i)$), $\ldots$

In this way a new, transformed table is constructed. It has in principle the same form as the original table but new values in its columns that have a different semantic meaning.

Tab. 2 reflects our example, where we have $\theta = 2$ classes.

<table>
<thead>
<tr>
<th>Object number</th>
<th>Basic features / depth properties</th>
<th>Assignment by trainer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_1$ $D_1(x)$</td>
<td>$p_2$ $D_2(x)$</td>
</tr>
<tr>
<td>1</td>
<td>$D_1(x_1)$</td>
<td>$D_2(x_1)$</td>
</tr>
<tr>
<td>2</td>
<td>$D_1(x_2)$</td>
<td>$D_2(x_2)$</td>
</tr>
<tr>
<td>...</td>
<td>$D_1(x_i)$</td>
<td>$D_2(x_i)$</td>
</tr>
<tr>
<td>...</td>
<td>$D_1(x_l)$</td>
<td>$D_2(x_l)$</td>
</tr>
</tbody>
</table>

After that the $\alpha$-procedure is used for further treatment but it works now with the transferred table.

Let us now shortly describe the $\alpha$-procedure [8]:

First, we perform some pre-selection, taking into further considerations only those properties $p_q, q = 1, \ldots, m$, whose values are completely separated or have some overlap as shown in fig. 2.3.1.

![Fig. 2.3.1: Classification by a single property $p_q$](image)
Next, we introduce a **discrimination power** or **separating power**, which is defined for a **single property** as:

\[ F(p_q) = \frac{\omega_q}{l}, \]

where \( \omega_q \) is the number of objects outside the intersection area of the property \( p_q \) and \( l \) - the lengths of the training set (sample), i.e. the number of objects.

We set a minimum admissible discrimination power \( F_{min} \), and at the first step select any property as a possible **feature**, whose discrimination power exceeds the minimum admissible one:

\[ F(p_q) > F_{min} \]

For the synthesis of the space, we select step-by-step those features that have best discrimination power. Each new feature shall increase the number of correctly classified objects. For this, we use the following definition of the **discrimination power** of a feature, selected at step \( k \):

\[ F(x_k) = \frac{\omega_k - \omega_{k-1}}{l} = \frac{\Delta\omega_k}{l}, \quad \omega_0 = 0, \]

where \( \omega_{k-1} \) is the accumulated number of correctly classified objects before the \( k \)-th feature was selected and \( \omega_k \) is the same after it was selected.

At Stage 1 we select a property having best discrimination power as a basis feature \( f_0 \) (= first axis) and represent the objects by their values on this axis; see fig. 2.3.2.

![Fig. 2.3.2: \( \alpha \)-procedure, stage 1](image)

At stage 2 we add a second property \( p_k \) to the coordinate system and project the objects to the plane that is spanned by the axes \( f_0 \) and \( p_k \). In this plane a ray originating from the origin is rotated up to the point where the **projections** of the objects onto this ray provide the best separation of the objects. The resulting ray, characterized by its rotation angle \( \alpha \), defines a possible new axis. We repeat this procedure for all remaining properties and select the property that gives the best separation of the objects on its rotated axis, which is denoted as \( \tilde{f}_1 \). This axis is taken as the first **new feature**, and the respective plane as the first **repère**; see fig. 2.3.3.
At Stage 3 we regard another property $p_j$ that has not been used so far and define the position of the objects in a new plane that is built by the axes $\tilde{f}_1$ and $p_j$. Again we consider a ray in this plane and turn it around the origin by the angle $\alpha$ until the projections of the objects onto this axis give the best separation. We repeat this procedure for all remaining properties and select the best one, which, together with $\tilde{f}_1$ forms the second repère (fig. 2.3.4).

In our simple example this feature already leads to a faultless separation of the objects. If all properties have been used but no complete separation of all objects reached, a special stopping criterion as described in [9] is to be used.

After $n$ steps the normal of the separating hyperplane is given by
\[
\left( \prod_{k=2}^{n} \cos \alpha_k^0, -\sin \alpha_2^0, \prod_{k=3}^{n} \cos \alpha_k^0, ..., -\sin \alpha_q^0, \prod_{k=q+1}^{n} \cos \alpha_k^0, ..., -\sin \alpha_n^0 \right); \quad n \leq n_0,
\]
where $\alpha_k^0$ denotes the angle $\alpha$ that is best in step $k$, $k = 2, ..., n$. Due to the fact that the formula shown above is stepwise calculated, the underlying features must be assigned backwards in practical classification.

For example, the separation decision plane and the decomposition of its normal vector are shown in Fig. 2.3.5 and 2.3.6.
3 Simulations and applications

To explore the specific potentials of the $\alpha$-procedure we apply it to simulated data. Besides the application to the original data, the $\alpha$-procedure can also be applied to properly transformed data; in particular, it has been successfully used to classify data on the basis of their DD-plot, see [10,11]. Especially, we have used two depth functions that are well computable in high dimensions, the zonoid depth [12] and the random Tukey depth [13].

The $\alpha$-procedure (applied in the original space (a-pr.(1)) and the extended space using polynomials of degree 2 (a-pr.(2)) and 3 (a-pr.(3))) is contrasted with the following nine classifiers: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), $k$-nearest neighbors classification (KNN), maximum depth classification based on Mahalanobis (MM), simplicial (MS), and halfspace (MH) depth, and DD-classification with the same depths (DM, DS and DH, correspondingly; see Li et al. (2012) for details), and to the DDa-classifier. Six simulation alternatives are used; each time a sample of 400 objects (200 from each class) is used as a training sample and 1000 objects (500 from each class) to evaluate the classifier’s performance (= classification error). First, normal location (two classes originate from $N\left(\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 4 \end{bmatrix}\right)$ and $N\left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 4 \\ 10 \end{bmatrix}\right)$, see fig. 3.1, left) and normal location-scale (the second class has covariance $\begin{bmatrix} 4 & 4 \\ 4 & 16 \end{bmatrix}$, see fig. 3.1, middle) alternatives are tried.

To test the proposed machinery for robustness properties we challenge it using a contaminated normal distribution, where the first class of the training sample of the normal location and location-scale alternatives considered above contains 10% objects originating from $N\left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 4 \\ 1 \end{bmatrix}\right)$ (see fig. 3.1, right and fig. 3.2, left correspondingly).

Other robustness aspects are demonstrated with a pair of Cauchy distributions forming a similar location-scale alternative, see Fig. 9, middle. Settings with exponential distributions (Exp(1),Exp(1)) vs. (Exp(1)+1,Exp(1)+1), see fig. 4.2) conclude the simulation study.
The $\alpha$-procedure performs fairly well for normal alternatives and shows remarkable performance for the robust alternatives considered. Though it works well on exponential settings as well, its goodness appears to depend on the size of the extended feature space. The last choice can be made by using either external information or cross-validation techniques.

4. Conclusion

The practical use of parametric statistical methods in so-called inverse tasks, among them in pattern recognition, is often difficult due to missing, unreliable or poorly founded information about the structure of the parametric dependency. As known, this shortage can be sometimes compensated by re-formulating the task into a task of minimizing a main functional, where this functional is represented as distance (the metrics can be different) between the solution of the direct task and the measured data.

Together with that a correction is performed with the help of regularization or a stabilizing functional using additional information. This is the so-called Tikhonov’s regularization [2]. Sometimes the regularization can be substituted or completed by validation.

The construction of the main and regularizing functional can be substituted by a “selection of solution” with the help of a criterion that owns a regularizating or validating component, as, for example, in:

- MGDH [14, 15, 16, 17, 18, 19]
- SVM [20]
- Selection according to Akaike [21, 22]

As an alternative to the approaches described above non-parametric methods can be used for inverse tasks.

The $D\alpha$-procedure is non-parametric in two respects:

- First, it uses the notion of depth of points in $d$-dimensional space.
- Second, the algorithm of $\alpha$-procedure itself is non-parametric – in contrast, for example, to the regression-based separation of patterns.

As we can see from the description of the $\alpha$-procedure, it is invariant with regard to the class belonging; i.e. the invariant is just the statement of the trainer. The procedure selects –stepwise – a planar projection and determines an angle $\alpha$ that yields a separation of the projected points like the univariate (= Tukey) median.

The $d$-dimension of the original properties can be very high. For a powerful acceleration of the $\alpha$-procedure in place of the original data the depth-transformed data can be used. Here only $\theta$ depth properties have to be calculated from $d$ original properties, $d \gg \theta$.

The application of the $\alpha$-procedure to the transformed depth-table of properties (see tab. 2) is a regularization or validation operation on the depth representation of the data.
References