# Contents

1 Installation ........................................... 3

2 Statistical data depths .......................... 5
   2.1 Mahalanobis depth ............................... 5
   2.2 \( L^p \) depth and \( L^2 \) depth ..................... 6
   2.3 Spatial depth ....................................... 6
   2.4 Random projection depth ............................. 6
   2.5 Random halfspace depth ............................. 7
   2.6 Simplicial depth ..................................... 7
   2.7 Modified band depth ................................ 8

3 Depth-based classification .................. 9
   3.1 DD-S .................................................. 9
   3.2 DD-NCC ............................................... 10
   3.3 \( DD_{\alpha} \) .............................................. 10

4 Feature Selection ................................. 13
   4.1 Feature-class-correlation ............................ 13

5 References ........................................... 15

Bibliography ......................................... 17
DataDepth.jl is a Julia package for computing statistical data depths and classifiers which are based on the concept of data depths. Data depths are functions which measure how ‘deep’ a multivariate data point is, with respect to a given data matrix X. Three classifiers based on data depth are introduced: DD-S, DD-NCC and $DD_\alpha$. DD-S is a threshold classifier based on univariate data depth axis. DD-NCC classifies test points according to the highest data depth value for data points out of each class. The $DD_\alpha$ procedure was introduced in Lange et al. (2012) and trains data points on a so called DD-Plot. These are two-dimensional or higher, when the extended DD-Plot is used. An optimal separation line is calculated on this plot and training as well as test points are projected onto a perpendicular line to the separation line. The sign of this perpendicular line determines the outcome of predictions. Also, feature selection is implemented with a simple filter. Features with highest absolute feature-class-correlation are chosen for selection. This is recommended for high-dimensional data sets as some data depths are not computable anymore.

Contents:
The source code is available on Github:

- DataDepth.jl

To add the package from Julia:

```julia
Pkg.clone("https://github.com/robinsz/DataDepth.jl.git")
```
Let $\mathbf{z} \in \mathbb{R}^d$ and $X$ a random variable with distribution $P$. A data depth is defined as $D(\mathbf{z}|X) : \mathbb{R}^d \rightarrow [0, 1]$ and indicates how ‘deep’ a data point $\mathbf{z}$ is inside of $X$. High values correspond to ‘deeper’ points inside of $X$ and lower values indicate points which are further away from the center of $X$. The center and how the depth function $D(\mathbf{z}|X)$ behaves is defined by various depth functions. A depth function also should satisfy following properties:

- **D1** Affine invariance: $D(A\mathbf{z} + \mathbf{b}|AX + \mathbf{b}) = D(\mathbf{z}|X)$ for any $\mathbf{b} \in \mathbb{R}^d$ and regular matrix $A \in \mathbb{R}^{d \times d}$
- **D2** Null at infinity: $\lim_{\|\mathbf{z}\| \to \infty} D(\mathbf{z}|X) = 0$
- **D3** Monotone on rays: If a point $\mathbf{z}^\alpha$ has maximal depth, then for any $r$ in the unit sphere of $\mathbb{R}^d$ the function $\alpha \to D(\mathbf{z}^\alpha + \alpha r|X)$ weakly decreases with $\alpha > 0$
- **D4** Upper semicontinous: The upper level sets $D_\alpha(X) = \{\mathbf{z}|D(\mathbf{z}|X) \geq \alpha, \mathbf{z} \in \mathbb{R}^d\}$ are closed for all $\alpha$

Note that these properties are not satisfied by all data depths, but are rather desirable. In the following an empirical distribution is considered for $X$ with data points $\mathbf{x}_1, ..., \mathbf{x}_n$. For theoretical aspects, these are represented as a matrix $X \in \mathbb{R}^{d \times n}$. Refer to the note below for the representation in Julia.

**Note:** In depth functions of this Julia package, observations of data matrices $X$ are stored row-wise. Data points of which depth is calculated, are normal Julia (column) vectors.

### 2.1 Mahalanobis depth

The Mahalanobis depth $D^{mah}$ is defined as

$$D^{mah}(\mathbf{z}|X) = (1 + (\mathbf{z} - \mathbf{\bar{x}})^T\Sigma_X^{-1}(\mathbf{z} - \mathbf{\bar{x}}))^{-1}$$

where $\Sigma_X$ is the covariance matrix and $\mathbf{\bar{x}}$ is the mean vector of $X$. The Mahalanobis depth satisfies all properties **D1** to **D4**. Instead of the covariance matrix, more robust estimates like the MCD matrix can be used. Also, $D^{mah}$ belongs to the group of metric data depths.

**depth_mahalanobis(x, data [; mcd])**

Calculates the Mahalanobis depth of $x$ w. r. t. matrix $data$. $mcd$ is a Bool field, where true means calculation with MCD estimate and false (default) means calculation with covariance matrix.
2.2 \( L^p \) depth and \( L^2 \) depth

The \( L^p \) depth is defined as

\[
D^{L^p}(\vec{z}|X) = (1 + \frac{1}{n} \sum_{i=1}^{n} \|\vec{z} - \vec{x}_i\|_p)^{-1}.
\]

It satisfies the properties D2 to D4, but not affine-invariance.

**depth\_Lp (x, data [; p])**

Calculates the \( L^p \) depth of \( x \) w. r. t. matrix \( data \). Parameter \( p \) is set to 2 as default.

An affine-invariant version of the \( L^2 \) depth is given as

\[
D^{L^2}(\vec{z}|X) = (1 + \frac{1}{n} \sum_{i=1}^{n} \|\vec{z} - \vec{x}_i\|_{\Sigma_X})^{-1}
\]

with the \( M \)-norm

\[
\|\vec{z}\|_M = \sqrt{\vec{z}^T M^{-1} \vec{z}}, \vec{z} \in \mathbb{R}^d.
\]

\( M \) is a positive definite \( d \times d \) matrix. Now, \( D^{L^2} \) satisfies D1 to D4. Again, \( \Sigma_X \) is the covariance matrix of \( X \) and instead, the robust MCD estimate can be used. Like \( D^{mah} \), the \( L^p \) and \( L^2 \) depths belong to the metric data depths.

**depth\_L2 (x, data [; mcd])**

Calculates the \( L^2 \) depth of \( x \) w. r. t. matrix \( data \). \( mcd \) is a \( \text{Bool} \) field, where \( \text{true} \) means calculation with MCD estimate and \( \text{false} \) (default) means calculation with covariance matrix.

2.3 Spatial depth

Using the idea of spatial quantiles (Chaudhuri, 1996), the affine-invariant spatial depth is defined as

\[
D^{spatial}(\vec{z}|X) = 1 - \|\frac{1}{n} \sum_{i=1}^{n} \nu(\Sigma_X^{-1/2}(\vec{z} - \vec{x}_i))\|
\]

where \( \nu(\vec{y}) = \frac{\vec{y}}{||\vec{y}||} \) for \( \vec{y} \neq \vec{0} \) and \( \nu(\vec{0}) = 0 \). Once again, \( \Sigma_X \) is the covariance matrix of \( X \) or alternatively the MCD estimate can be used. The spatial depth as defined above satisfies D1 to D4.

**depth\_spatial (x, data [; mcd])**

Calculates the spatial depth of \( x \) w. r. t. matrix \( data \). \( mcd \) is a \( \text{Bool} \) field, where \( \text{true} \) means calculation with MCD estimate and \( \text{false} \) (default) means calculation with covariance matrix.

2.4 Random projection depth

Random projection depth (Velasco-Forero, 2012) is defined as

\[
D^{proj}(\vec{z}|X, k) = (1 + O^{proj}(\vec{z}|X, k))^{-1}
\]

with

\[
O^{proj}(\vec{z}|X, k) = \max_{\vec{u} \in U} \frac{|\vec{u}^T \vec{z} - m(\vec{u}^T X)|}{MAD(\vec{u}^T X)}
\]

Chapter 2. Statistical data depths
and

\[ m(\vec{y}) = \text{median}\{\vec{y}, 1 \leq i \leq n\} \]
\[ \text{MAD}(\vec{y}) = \text{median}\{|\vec{y} - m(\vec{y})|, 1 \leq i \leq n\}. \]

Also, \( U = \{\vec{u}_1, \vec{u}_2, ..., \vec{u}_k\} \) is defined with \( \vec{u}_i \in S^{d-1} \). \( D^{Rprj} \) satisfies \( D_1 \) to \( D_4 \) and is also a member of the metric data depths. A crucial problem with \( D^{Rprj} \) is the choice of projection parameter \( k \). In this Julia package, the default value is \( k = 1000 \), but it can be easily increased to a higher number.

**depth_projection(\( x, data []; k \))**
Calculates the random projection depth of \( x \) w. r. t. matrix \( data \). \( k \) is the parameter for the count of random projections. As a default value, \( k = 1000 \) is set.

### 2.5 Random halfspace depth

Halfspace depth, constructed with random projections, is defined as

\[ D^{Rloc}(\vec{z}|X, k) = \frac{1}{n} \min_{\vec{u} \in U} \sum_{i=1}^{n} I[\vec{u} \cdot \vec{x}_i \geq \vec{u} \cdot \vec{z}] \]

with \( I \) being the indicator function and \( U = \{\vec{u}_1, \vec{u}_2, ..., \vec{u}_k\} \) with \( \vec{u}_i \in S^{d-1} \). Random halfspace depth satisfies \( D_1 \) to \( D_4 \) and is called a combinatorical data depth. Note that the choice of \( k \) is once again crucial. This Julia package uses \( k = 1000 \) as default value. Another critical aspect of the halfspace depth is the fact that data points beyond the convex hull of \( X \) vanish, which means they have depth value zero. In certain classification scenarios this is a problem and calls for extra treatment.

**depth_halfspace(\( x, data []; k \))**
Calculates the random halfspace depth of \( x \) w. r. t. matrix \( data \). \( k \) is the parameter for the count of random projections. As a default value, \( k = 1000 \) is set. Note that this data depth also can be zero if \( x \) is outside the convex hull of \( data \).

### 2.6 Simplicial depth

Liu (1990) defines the Simplicial depth as follows:

\[ D^{sim}(\vec{z}|X) = \left(\frac{n}{d+1}\right)^{-1} \sum_{1 \leq i_1 < ... < i_{d+1} \leq n} I[\vec{z} \in S[\vec{x}_{i_1}, ..., \vec{x}_{i_{d+1}}]]. \]

Again, \( I \) is the indicator function and \( S[\vec{x}_{i_1}, ..., \vec{x}_{i_{d+1}}] \) the simplex of vectors \( \vec{x}_{i_1}, ..., \vec{x}_{i_{d+1}} \). The sample simplicial depth as defined above satisfies \( D_1 \) and \( D_2 \) but not stringently \( D_3 \) and \( D_4 \). Especially with discrete probability distributions, Zuo (2000) showed that \( D_3 \) and \( D_4 \) not always hold. This Julia package only implements a two-dimensional version of simplicial data depth which measures the depth via every possible triangle of bivariate data points in \( X \). Therefore, simplicial depth is also a combinatorial data depth.

**depth_simplicial(\( x, data \))**
Calculates the bivariate halfspace depth of \( x \) w. r. t. matrix \( data \). Note that this data depth also can be zero if \( x \) is outside the convex hull of \( data \). Simplicial depth so far is only implemented for bivariate data.
2.7 Modified band depth

As one of the functional data depths, the modified band depth is introduced as

\[
MBD(\vec{z} | \vec{X}) = \left( \frac{n}{2} \right)^{-1} \sum_{1 \leq i_1 < i_2 \leq n} \frac{1}{d} \sum_{k=1}^{d} \mathbb{I}[\min(\bar{x}^{(k)}_{i_1}, \bar{x}^{(k)}_{i_2}) \leq z^{(k)} \leq \max(\bar{x}^{(k)}_{i_1}, \bar{x}^{(k)}_{i_2})]
\]

with \( \bar{x}^{(k)} \) being the \( k \)-th component of a vector \( \bar{x}, k \in \{1, \ldots, d\} \). \( MBD \) satisfies \( D_1, D_2 \) and \( D_4 \). \( D_3 \) is a strict restriction for functional data. Also, the modified band depth is one of the data depths with a complexity which is more dependent on \( n \) than on \( d \), \( O(n^2 \cdot d) \) that is. This makes it suitable for data sets with \( n \ll d \) where other data depths like \( D_{mah}, D^L \) or \( D_{sim} \) fail.

**modified_band_depth** (x, data)

Calculates the modified band depth of \( x \) w. r. t. matrix \( data \).
This package implements three different ways to classify new data points and to train data based on the concept of statistical data depths.

**Note:** Observations of test data points are stored row-wise as well as training data.

### 3.1 DD-S

**Two classes** Let $\mathcal{Y} = \{c_1, c_2\}$ be the class labels of two classes and $X_1 = [\vec{x}_1, ..., \vec{x}_{n_1}] \in \mathbb{R}^{d \times n_1}$, $X_2 = [\vec{x}_{n_1+1}, ..., \vec{x}_{n_2}] \in \mathbb{R}^{d \times n_2}$ be data matrices that belong to those two classes. Moreover, $n = n_1 + n_2$. DD-S now projects every point $\vec{x}$ out of $X_1$ and $X_2$ onto two separate univariate axis with $D(\vec{x}_i | X_1)$ and $D(\vec{x}_i | X_2) \forall i = 1, ..., n$, with $D$ being any depth function. The result are two separate single threshold classifiers based on data depths.

Now DD-S searches an optimal threshold $S$ on which minimizes misclassification rate of all ordered depth values along the resulting univariate axis. When the first threshold classifier contain all depth values $D(\vec{x}_i | X_1) \forall i = 1, ..., n$, a test point $\vec{z} \in \mathbb{R}^d$ gets the class label $c_1$ when $D(\vec{z} | X_1) \geq S$ yields, else it gets the label $c_2$. The second DD-S classifier is analogously defined, but with $X_2$ instead of $X_1$.

**More than two classes** For more than two classes, every possible combination of two classes is considered. This yields $\binom{|\mathcal{Y}|}{2}$ threshold classifiers. The resulting class for a new test point $\vec{z} \in \mathbb{R}^d$ is decided via majority vote. If the assignment of the result class label is not distinct, $c_1$ is chosen as label.

*single_threshold_classifier* (test_data, training_data, training_classlabels, depth_function [: classlabel_ind])

Classifies test data points *test_data* with training data *training_data* and corresponding class labels *training_classlabels*. *depth_function* determines which data depth is used for depth axis. Parameter *classlabel_ind* is needed for 2 class problems and can be 1 or 2, resulting in a different single threshold classifier. For more than 2 classes, this parameter doesn’t have to be set, so majority vote is triggered.
3.2 DD-NCC

Another approach to use data depths for classification is via maximal data depths w. r. t. each class. This is called DD-NCC (Data depth-Nearest centroid classifier). Let \(X_1, \ldots, X_l\) be data matrices that belong to \(l\) classes with labels \(\mathcal{Y} = \{c_1, \ldots, c_l\}\). A new point is then classified to the label \(c_k\) with \(k = \arg\max_j D(\vec{x}|X_j), j = 1, \ldots, l\).

\[\text{ddncc\_classify}(\text{test\_data}, \text{training\_data}, \text{training\_classlabels}, \text{depth\_function})\]

Classifies test data points \(\text{test\_data}\) with training data \(\text{training\_data}\) and corresponding class labels \(\text{training\_classlabels}\). \text{depth\_function} determines which data depth is used for depth axis.

3.3 \(DD\alpha\)

Two classes

The \(DD\alpha\) procedure is a classifier by Lange et al. (2012) based on so-called (extended) DD-Plots. A DD-Plot is based on two distinct classes and forms a two dimensional data-depth-dependent coordinate system. Let’s say \(X_1 = [\vec{x}_1, \ldots, \vec{x}_{n_1}] \in \mathbb{R}^{d \times n_1}\) and \(X_2 = [\vec{x}_{n_1+1}, \ldots, \vec{x}_{n_1+n_2}] \in \mathbb{R}^{d \times n_2}\) are two classes with \(n = n_1 + n_2\). Then, the DD-Plot is defined as

\[
DD = \left\{ \vec{d}_i \in [0, 1]^2 \bigg| \vec{d}_i = (D(\vec{x}_i|X_1), D(\vec{x}_i|X_2)), i = 1, \ldots, n \right\}.
\]

Moreover, DD-plots can be extended to higher dimensions by also using products and exponents to a certain degree \(p\). For two classes, this yields dimension \(r = \binom{p+2}{2}\). Let’s say \(p = 2\), then a 5-dimensional extended DD-Plot is obtained and is calculated as follows:

\[
EDD = \left\{ \vec{d}_i \in [0, 1]^5 \bigg| \vec{d}_i = (D(\vec{x}_i|X_1), D(\vec{x}_i|X_2), D(\vec{x}_i|X_1) \cdot D(\vec{x}_i|X_2), D(\vec{x}_i|X_1)^2, D(\vec{x}_i|X_2)^2), i = 1, \ldots, n \right\}.
\]

With \(p = 1\) the DD-Plot and extended DD-Plot are equal.

First of all, \(DD\alpha\) calculates \(EDD\) of the given training data. Then, every possible coordinate-pair \((\nu_1, \nu_2)\) out of \(EDD\) is chosen, although \(nu_1\) and \(nu_2\) have to depend on other classes. So, for example, a pair \((D(\vec{x}_i|X_1), D(\vec{x}_i|X_2)^2)\) is forbidden and a pair \((D(\vec{x}_i|X_1), D(\vec{x}_i|X_1) \cdot D(\vec{x}_i|X_2))\) is allowed. With those two coordinates chosen, a coordinate system with these and a separation line through the origin of it is constructed. This separation line is defined by an angle \(\alpha\) which minimizes average misclassification rate (AMR) in the \((\nu_1, \nu_2)\) plane. When the corresponding angle is within an interval, the middle value is chosen. Out of all \((\nu_1, \nu_2)\)-combinations the pair with minimal AMR defined by the corresponding optimal pair \((\nu_{1*}, \nu_{2*})\) and angle \(\alpha^{(1)} = \alpha_{\nu_{1*}, \nu_{2*}}\) is chosen. Now, a new line perpendicular to the separation line defined by \(\alpha^{(1)}\) is built as follows:

\[
\mu_i^{(i)} = \nu_1^{(i)} \cos \alpha^{(1)} - \nu_2^{(i)} \sin \alpha^{(1)}, i = 1, \ldots, n_1 + n_2.
\]

The coordinates \((\nu_{1*}, \nu_{2*})\) are now completely replaced by \(\mu_1\). With that line, a new \((\mu_1, \nu)\)-plane is constructed with \(\nu\) being one of the unused coordinates left in \(EDD\). All steps above are repeated to build another optimal \(\alpha^{(2)} = \alpha_{\mu_1, \nu*}\) with a corresponding \(\mu_2\) which then replaces \((\mu_1, \nu^{*})\). These steps are repeated until the AMR doesn’t change anymore or there are no coordinates in \(EDD\) left. In the end, the resulting line is a polynomial consisting of data depths w. r. t. \(X_1\) and \(X_2\), as well as products and exponents of them. A new data point for testing is then labelled according to the sign of the resulting polynomial.
More than two classes With more than two classes, every possible two-class-combination is considered and the algorithm above is used. In the end, a new test point will be assigned via majority vote.

Outsiders When using halfspace or simplicial depth, the occurrence of outsiders is possible. Outsiders are point which lie in the origin of $DD$ or $EDD$. This can happen if a new point is outside of the convex hull of training points. Because separation is done by a line through the origin, $DD\alpha$ has no decision which class is correct. In recent literature there are some ways to handle this problem. This Julia package uses 3NN (k-nearest-neighbor with $k = 3$) as a treatment for those points, done in the original feature space, not depth-transformed in $DD$ or $EDD$.

ddalpa_classify(test_data, training_data, training_classlabels, depth_function [:p])

Classifies test data points test_data with training data training_data and corresponding class labels training_classlabels. depth_function determines which data depth is used for depth axis. Parameter p determines the dimension of extended DD-Plot for separation. Note that some data depths (especially spatial depth) perform better with higher p. The default value is p = 2.
Feature Selection

In this Julia package, feature selection methods are implemented to handle problems with high-dimensional data. These usually are: High complexity of inverse covariance matrix and an extremely high number of $k$ projections when using random projection or random halfspace depth. So far, the only implemented feature selection method, is selecting the $t < d$ ‘best’ features by absolute feature-class-correlation.

4.1 Feature-class-correlation

Let $\vec{v}_i \in \mathbb{R}^n$, $i = 1, \ldots, d$ be the feature vectors and $(x_j, y_j) \forall j = 1, \ldots, n$ the training data with $y_j \in \mathcal{Y}$ and numerical class labels.

Then, the feature-class-correlation is calculated in the following way:

$$ r_{\vec{v}_i} = \frac{\sum_{k=1}^{n} (\vec{v}_i^{(k)} - \overline{v}_i)(y_k - \overline{y})}{\sqrt{\sum_{k=1}^{n} (\vec{v}_i^{(k)} - \overline{v}_i)^2 \cdot \sum_{k=1}^{n} (y_k - \overline{y})^2} } $$

with $\vec{v}_i^{(k)}$ being the $k$-th component of $\vec{v}_i$, $\overline{v}_i$ the middle value of $\vec{v}_i$ and $\overline{y}$ the middle value of all numerical class labels.

Now, the $t < d$ highest absolute values out of $r_{\vec{v}_i}$, $i = 1, \ldots, d$ are chosen as new features.

**feature_selection_fcc** *(data_set, classlabels [: k]*)

Selects the $k$ ‘best’ features from given data matrix data_set with class labels classlabels in terms of absolute feature-class-correlation as defined above. Default value for $k$ is 50.
CHAPTER 5

References


D

dalpha_classify() (built-in function), 11
ddncc_classify() (built-in function), 10
depth_halfspace() (built-in function), 7
depth_L2() (built-in function), 6
depth_Lp() (built-in function), 6
depth_mahalanobis() (built-in function), 5
depth_projection() (built-in function), 7
depth_simplicial() (built-in function), 7
depth_spatial() (built-in function), 6

F

feature_selection_fcc() (built-in function), 13

M

modified_band_depth() (built-in function), 8

S

single_threshold_classifier() (built-in function), 9