Hyperplane Matrix Classifier

Yaniv Karta **yaniv@provallo.com**

January 24, 2024

Contents

1 Introduction 2
1.1 Background . . . 2
1.2 Motivation . . . 2

2 phylogenetic inspiration 2

3 Hyperplane Matrix stability 4

4 Hyperplane Matrix Algorithm 6
4.1 Hyperplane Matrix Classifier fit function . . . 8
4.2 Hyperplane Matrix Classifier fit function . . . 8
4.3 Hyperplane Matrix Classifier predict function . . . 10
4.4 Hyperplane Matrix Classifier score function . . . 10
4.4.1 Hyperplane Matrix Computational complexity . . . 10

5 Hyperplane Matrix Classifier hyperparameters 10
5.1 Hyperplane Matrix Classifier hyperparameter optimization . . . 11
5.2 Hyperplane Matrix curvature . . . 13
5.3 Hyperplane Matrix Classifier curvature optimization . . . 13

6 Outlier detection applications 13
6.1 Isolation Forests . . . 13
6.2 Hyperplane Matrix Classifier vs Isolation Forest . . . 14

7 knowledge-transfer applications 28
7.1 Distance metrics for knowledge-transfer . . . 28
7.1.1 Knowledge-transfer from a non-metric space to a metric space . . . 30
7.1.2 Knowledge-transfer from a metric space to a non-metric space . . . 30
7.1.3 Knowledge-transfer evaluation . . . 31

8 Acklogements 35

9 Bibiliography 36

1 Introduction

As part of our meta learning pipelines, we have developed a new breed of classifiers called hyperplane matrix classifiers. The hyperplane matrix classifier is a generalization of the hyperplane classifier without construction or aggregation trees or forests. It process the input data into a matrix of hyperplanes and uses the hyperplanes to construct a super tree matrix representation explained in the phylogenetic inspiration section. It does not construct a tree or a forest and therefore it is not a tree-based classifier.

This paper will focus on parallel comparison to Isolation Forests (1) as an outlier detector and as a classifier. In addition to transfer learning between classification tasks.

1.1 Background

In 2019 a blog post (2) The author was published on tweaking the properties of Isolation Forests to improve the runtime performance and accuracy of Isolation Forests. The author suggested that further runtime improvement can be achieved by tweaking the normal vector $\vec{n}$ from a normal distribution to a uniform distribution of the unit sphere and construction of a hyperplane matrix to reduce runtime cost. The author concludes with a suggestion to use Isolation Hamming Matrix as an alternative to Isolation Forests. The concept of Isolation Hamming Matrix eventually evolved into the generalized hyperplane matrix classifier.

1.2 Motivation

The motivation was to create an alternative to tree-based solutions that can be used as a classifier or as an outlier detector or optimal median container for knowledge transfer between classification tasks. satisfying both coverage of the problem space and accuracy of the solution with an optimal runtime cost.

2 phylogenetic inspiration

A super tree (3) is a phylogenetic tree that is constructed from a set of input trees it is designed to scale estimation of phylogenetic trees to large datasets. This method has many applications in computational biology and disadvantages such as the super tree may not be a phylogenetic tree. Examples of super tree methods include matrix representation with parsimony (MRP) (6) , Superfine , and Matrix Representation with Likelihood (MRL) ,DACTAL , and Matrix Representation with Compatibility (MRC) . Brief explanation of super tree implementations:

RFS - Robinson-Foulds distance The Robinson-Foulds distance is a metric. (23) It is used as a measure of the distance between two phylogenetic trees and is also a median tree. RF distance is defined as the number of bipartitions that are present in one tree but not the other. RFS or Robinson-Foulds Supertree(15) is a method for constructing a super tree from a set of input trees. The RFS method is a median tree and is used to construct a super tree from a set of input trees. A good solution proposed by M.Vachaspati and T.Warnow (21) FastRFS minimizes the search complexity to $O(nk|X|^2)$ where X is the set of allowed bipartitions.

Concensus criterion The concensus criterion is used to construct a super tree from a set of input trees. An input set of trees $\mathcal{T}$ is a set of trees $\mathcal{T} = \{T_1, T_2, ..., T_n\}$ is a set containing all the bipartitions of the input trees. and each tree T_i is a tree with n leaves.

a strict concensus tree is is a tree that has exactly that number of leaves as the input trees and each leaf is present in exactly one input tree $\mathcal{T}$. and is satisfying

$$C(\mathcal{T}) = \cup_{i=1}^{k} C(T_i) \tag{1}$$

where k is the number of input trees and $C(T_i)$ is the set of clades of the input tree T_i . By construction the strict concensus tree of $\mathcal{T}$ will not fully resolve unless all the trees in $\mathcal{T}$ are topologically identical.

Majority Concensus A Majority concensus is when the bipartitions of the input trees are weighted by the number of input trees that contain the bipartition. The Majority concensus minimize the RF distance between the input trees and the super tree

$$\sum_{i=1}^{n} RF(\mathcal{T}, T_i) \quad (2)$$

Greedy Concensus A greedy concensus tree (20) is either equal to the strict concensus tree or refines it. it complements the tree with the Majority concensus by adding additional bipartitions to the tree.

MRP - matrix representation with parsimony Matrix representation with parsimony MRP (6) is a method for constructing a super tree from a set of input trees.

MRP method (14) uses bipartitions to construct a tuple of binary vectors for each input tree. The value of the cell is 1 if the species is in the clade defined by the edge and 0 otherwise. The objective is maximum parsimony super tree reconstruction. MRP also imputes missing data in the input trees.

The MRP method is considered NP-hard (18) .

MRL - matrix representation with likelihood Similar to MRP, MRL introduced by Nguyen et. al (4) is a method for constructing a super tree from a set of input trees. MRL method uses bipartitions to construct a tuple of binary vectors for each input tree.

The value of the cell is 1 if the species is in the clade defined by the edge and 0 otherwise. The objective is maximum likelihood super tree reconstruction

MRL also imputes missing data in the input trees. The input for the MRL method is a set of unrooted input trees and the output is a super tree. MRL scores are topologically more correlated the accuracy of the input trees.

MRC - matrix representation with compatibility When the input trees are not compatible, MRC (5) is used to construct a super tree from a set of input trees. MRC method uses bipartitions to construct a tuple of binary vectors for each input tree. A profile can have more than one compatability supertree and finding one is very hard since it parallels to the Quartet tree compatability problem. (25)

Superfine Superfine (95) is a method for constructing a super tree from a set of input trees.

a backbone SCM (Strict Consensus Merger) (98) is constructed from the profile of the input trees. The SCM tree is used as a constraint tree for the input trees.

Superfine method recodes the topological information of the input trees into a matrix representation. The matrix representation is used to construct a super tree. The super tree is constructed by finding the maximum likelihood tree that is consistent with the input trees.

The outcome of the process is a supertree that is a maximum likelihood tree that is consistent with the input trees and can be used to refine the polytomies of the chosen taxon set.

DACTAL Divide and Conquer Trees for Ancestral Likelihood (13) is a method for constructing a super tree from a set of input trees.(13) DACTAL method, similar to MRL and MRP, uses bipartitions to construct a tuple of binary vectors for each input tree. DACTAL process the input trees in a divide and conquer fashion. DACTAL is a maximum likelihood method.

3 Hyperplane Matrix stability

In general, stability boundaries will possess singularities. The dynamical system $\dot{x} = \mathbb{A}(\lambda)x$ where $\mathbb{A}(\lambda)$ is a matrix of the form:

$$\mathbb{A}(\lambda) = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} \tag{3}$$

will have the following solution:

$$x(t) = \begin{bmatrix} e^{\lambda_1 t} & 0 & 0 & 0 \\ 0 & e^{\lambda_2 t} & 0 & 0 \\ 0 & 0 & e^{\lambda_3 t} & 0 \\ 0 & 0 & 0 & e^{\lambda_4 t} \end{bmatrix} \tag{4}$$

The solution will be stable if and only if the real part of each eigenvalue of $\mathbb{A}(\lambda)$ is negative. The eigenvalues of $\mathbb{A}(\lambda)$ are the roots of the characteristic polynomial of $\mathbb{A}(\lambda)$. The characteristic polynomial of $\mathbb{A}(\lambda)$ is also the determinant of $\mathbb{A}(\lambda) - \lambda I$.

$$det(\mathbb{A}(\lambda) - \lambda I) = 0 \tag{5}$$

when the eigenvector $\vec{v}$ is a solution to the characteristic polynomial of $\mathbb{A}(\lambda)$, then the eigenvalue λ is a root of the characteristic polynomial of $\mathbb{A}(\lambda)$.

$$(\mathbb{A}(\lambda) - \lambda I)\vec{v} = 0 \tag{6}$$

singularities are points where the eigenvalues of $\mathbb{A}(\lambda)$ are equal to zero.

We can say therefore that $X \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^t$ and $A_t \in \mathbb{R}^{n\times n}$ are asymptotically stable if and only if the real part of each eigenvalue of A_t which has the largest real part is negative.

We demostrate with a few points :

- The points of bifurcation are the points where the eigenvalues of A_t are equal to zero.
- fluttering is when the eigenvalues of A_t are complex numbers with non-zero imaginary part.
- divergence is when the eigenvalues of A_t are positive.
- damping is when the eigenvalues of A_t are negative.
- undamped is when the eigenvalues of A_t are zero.

Damping is structurally stable and undamped is structurally unstable.

The following figure illustrates the concept of stability boundaries in the hyperplane space.

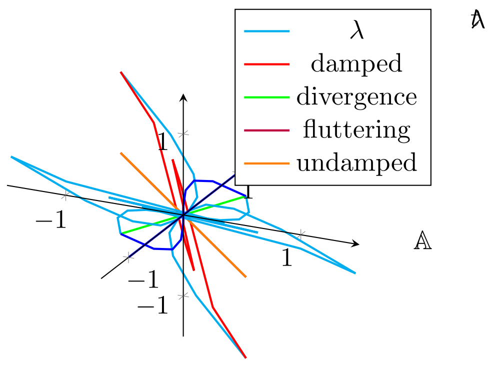

Figure 1: Hyperplane space

The following theorem is a generalization of the above figure.

- when $t \to -\infty$ the solution will converge to the origin.
- when $t \to \infty$ the solution will diverge from the origin.
- when $t \to 0$ the solution will oscillate around the origin.

For damped models the curves will intersect forming a boundary (with the fluttering curves). The undamped models will have a form of tangency at the point of intersection with the divergence and fluttering curves.

Theorem H.1 (Form of the stability boundary) (114) states :

- The stability boundary of a general two parameter family of matrices consists of smooth arcs intersecting transversally at their endpoints.
- At the intersection of the two arcs the acute angle of the stability boundary always points into the domain of instability.

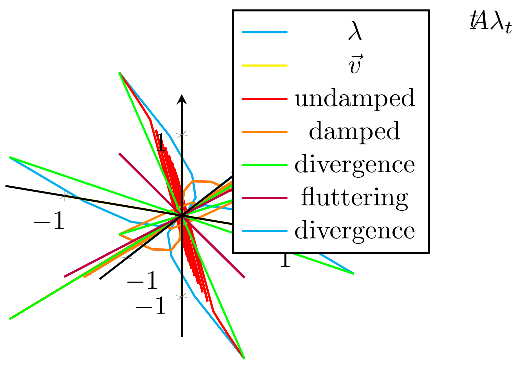

Figure 2: Hyperplane space

This is useful when the input data is unpredictable or in advanced scenarios where the input data needs to be infered from a pre-trained model.

It can be applied to isolate outliers from the input data and to classify the input data or to transfer knowledge between classification tasks when re-training over the original data is costly or unavailable.

4 Hyperplane Matrix Algorithm

The initial posterior probabilties are initialized with a uniform distribution, the vectors for each hyperplane are initialized with a normal distribution of the data represented in each column : $\vec{n} = \mathcal{N}_{X_{min,j},X_{max,j}}$ where j is the column index

Below is the algorithm for the hyperplane matrix classifier init function :

and the algorithm:

Algorithm 1 Hyperplane Matrix Classifier init function pseudocode

```
0: procedure InitHyperplanes($X$,$y$,$N$,$M$,$K$,$T$,$\theta$,$\rho$)
   InitializeProbabilities($X$, $y$, $N$, $M$, $K$, $T$, $\theta$, $\rho$)
0:   for $i \leftarrow 1,\ n$ do
0:     for $k \leftarrow 1,\ K$ do
0:       for $m \leftarrow 1,\ M$ do
0:         for $n \leftarrow 1,\ M$ do
0:           $\rho_{i,j}(t,k,m,n) \leftarrow \rho_{i,j}(t,k,m,n) + X_{i,j} \in y_i$
0:           $T_{i,j}(t,k,m,n) \leftarrow \rho_{i,j}(t,k,m,n) * X_{i,j} \in y_i$
0:           $M_{i,j}(t,k,m,n) \leftarrow M_{i,j}(t,k,m,n) - distance(\rho_{i,j}(t,k,m,n), \rho_{i,j}(t,k,m,n)) \cdot M_{i,j}(t,k,m,n)$
0:           $\theta_{i,j}(t,k,m,n) \leftarrow \theta_{i,j}(t,k,m,n) - distance(\rho_{i,j}(t,k,m,n), \rho_{i,j}(t,k,m,n)) \cdot \theta_{i,j}(t,k,m,n)$
0:         end for
0:       end for
0:     end for
0:   end for
0: end procedure=0
```

The probabilities are initialized with the following algorithm:

The hyperplane matrix initialize the hyperplane matrix and then based on the data it initializes the weighted hyperplane matrix. when the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes are initialized, the hyperplane matrix classifier fit function is called. Note, that the InitializeProbabilities function actually constraints and configures the hyperplanes constructed on the init hyperplane function from the normal and uniform distributions.

Algorithm 2 Hyperplane Matrix Classifier init function pseudocode

```
0: procedure InitializeProbabilities($X$,$y$,$N$,$M$,$K$,$T$,$\theta$,$\rho$)
0:   for $i \leftarrow 1,\ n$ do
0:     for $j \leftarrow 1,\ m$ do
0:       $X_{i,j}$; { feature value}
0:       $X_{i,j-1}$ ; { feature left 'branching'}
0:       $X_{i,j+1}$ ;{ feature right 'branching'}
0:       $X_{min,j}$ ;{ min coeffiecient of $X_i$}
0:       $X_{max,j}$ ;{ max coeffiecient of $X_i$}
0:       $X_{max,j}$ ;{max of $X_{n,m}$}
0:       $X_{max,j} - X_{min,j}$ ;{range of column $j$ values }
0:       $M_{i,j}$ ;{index of $X_{n,m}$}
0:       $M_{i,j-1}$; {index of $X_{n,m}$}
0:       $M_{i,j+1}$ ;  Commentindex of $X_{n,m}$
0:       $M_{min,j}$ ;{index of $X_{n,m}$}
0:       $M_{max,j}$ ;{index of $X_{n,m}$}
0:       $K_{i,j}$ ;{probability of class $j$ for sample $i$}
0:       $dimension = distance(M_{i,j}, M_{i,min}, M_{i,max})$ ; {hyperplane dimension}
0:       $X_{min,j}$ ; {hyperplane parent}
0:       $X_{min,j-1}$ ; {hyperplane left}
0:       $X_{max,j+1}$ ; {hyperplane right}
0:       $M_{max,j} - M_{min,j} * M_{max,j} - M_{min,j}$ ; {hyperplane depth}
0:       $M_{max,j} - M_{min,j}$ {hyperplane level}
0:       $distribtion \leftarrow X_{max,j}, X_{min,j}$ ; {hyperplane distribution}
0:       $scores \leftarrow X_{max,j}, X_{min,j}$ ; {hyperplane scores}
0:       $weights \leftarrow X_{max,j}, X_{min,j}$ ; {classifier weights}
0:       $T_{i,j,n,k} \leftarrow T_{i,j,n,k} + X_{i,j} \in y_i$ ; {super tree from the input data}
0:       $M_{i,j,n,k} \leftarrow M_{i,j,n,k} + X_{i,j} \in y_i$ ; {projection plane}
0:       $\rho_{i,j,n,k} \leftarrow \rho_{i,j,n,k} + X_{i,j} \in y_i$ ; {hyperplane matrix}
0:       $\theta_{i,j,n,k} \leftarrow \theta_{i,j,n,k} + X_{i,j} \in y_i$ ; {hyperplane matrix classifier weights}
0:     end for
0:   end for
0: end procedure=0
```

The ranges for the features and setup of the distributions is conducted on the InitializeProbabilities function. The hyperplane matrix classifier fit function is called after the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes are initialized.

The projection planes can be visualized in respect to the hyperparameter space:

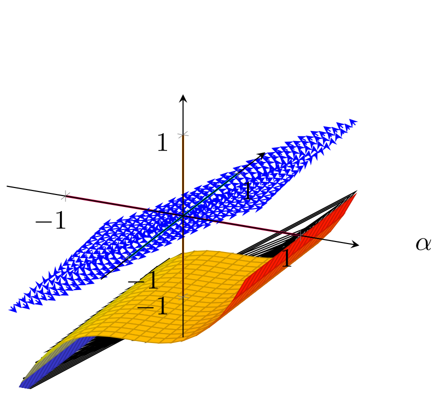

Figure 3: Hyperplane TensorMap

4.1 Hyperplane Matrix Classifier fit function

Algorithm 3 Hyperplane Matrix Classifier fit function pseudocode

```
0: procedure FIT($X$,$y$,$N$,$M$,$K$,$T$,$\theta$,$\rho$)
0:     InitializeProbabilities($X$, $y$, $N$, $M$, $K$, $T$, $\theta$, $\rho$) ;{initialize the hyperplane matrix classifier hyperplanes and hyperplane matrix classifier weights}
0:     $T_{m,n} \leftarrow X_{m,n} \in y_n$;{super tree from the input data}
0:     $\rho_{m,n} := T_{m,n} * M_{m,n}$ ;{hyperplane matrix}
0:     $\theta_{m,n} := \rho_{m,n} * X_{m,n}$ ;{hyperplane matrix classifier weights}
0:     return $\theta$ , $\rho$ , $T_{i,j}$(t,k,m,n) , $M_{i,j}$(t,k,m,n) , $\rho_{i,j}$(t,k,m,n) , $\theta_{i,j}$(t,k,m,n)
0: end procedure=0
```

X and y are the input data and input data labels respectively. n is the number of samples in the input data. m is the number of features in the input data. k is the number of classes in the input data. t is the trajectory of the hyperplane matrix classifier hyperplanes and the hyperplane matrix classifier weights. K is the output matrix of the hyperplane matrix classifier hyperplanes.

θ is the hyperplane matrix classifier weights. ρ is the hyperplane matrix classifier hyperplanes. $T_{i,j}(t,k,m,n)$ is the super tree from the input data. $M_{i,j}(t,k,m,n)$ is the projection plane

The projection plane and the hyperplane matrix are initialized with uniform and normal distribution. Then for every sample in the input data, the hyperplane matrix classifier fit function updates the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes.

4.2 Hyperplane Matrix Classifier fit function

Applying the chain rule to the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes, the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes are updated using the following equations:

The hyperplane matrix classifier fit function updates the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes using the following equations:

$$\rho_{t+1} = \rho_t + \alpha \frac{\partial L(y,\hat{y})}{\partial \rho} \tag{7}$$

The 'trees' are updated according to the leaves represented by the input data:

$$T_{t+1} = distance(\rho_t, \rho_{t+1}) \cdot T_t \tag{8}$$

The 'projection planes' are updated :

$$\mathcal{M}_{t+1} = \mathcal{M}_t - \alpha \frac{\partial L(y, \hat{y})}{\partial M} \tag{9}$$

and the indices of the hyperplane matrix are updated when the hyperplane matrix classifier weights and hyperplane matrix classifier hyperplanes are updated :

$$K_{i,j} = \frac{K_{i,j}}{\partial \theta_{i,j} - \theta_t * k} \tag{10}$$

where k is the number of hyperplanes in the hyperplane matrix. the indices dictate the trajectory of the probabilities of the hyperplane matrix classifier hyperplanes and the hyperplane matrix classifier weights. $K_{i,j}$ is the indices of the hyperplane matrix classifier hyperplanes, by comparing the distance between the hyperplane matrix classifier hyperplanes and the hyperplane matrix classifier weights, the indices are updated.

Note that a distance function detailed later is used to update the θ and ρ hyperparameters. the details of the distance functions are elaborated in the hyperplane matrix classifier predict function section and the transfer knowledge section. For now, we assume that the distance function is a simple euclidean distance function.

Therefore θ updates are done using the following equation:

$$\theta_{t+1} = \theta_t + \alpha \frac{\partial L(y, \hat{y})}{\partial \theta} \tag{11}$$

and ρ updates are done using the following equation:

$$\rho_{t+1} = \rho_t + \alpha \frac{\partial L(y, \hat{y})}{\partial \rho} \tag{12}$$

ρ_t is the hyperplane matrix classifier probabilities at time t. $K_{i,j}$ is the indices of the hyperplane matrix classifier hyperplanes. $T_{i,j}$ is the super tree from the input data. $\mathcal{M}_t$ is the projection plane. θ_t is the hyperplane matrix classifier weights. α is the learning rate. $L(y, \hat{y})$ is the loss function. y is the input data labels. $\hat{y}$ is the predicted labels. X is the input data. n is the number of samples in the input data. m is the number of features in the input data. k is the number of classes in the input data. t is the trajectory of the hyperplane matrix classifier hyperplanes and the hyperplane matrix classifier weights. $T_{i,j}(t, k, m, n)$ is the super tree from the input data. $M_{i,j}(t, k, m, n)$ is the indices of the hyperplane matrix, $\rho_{i,j}(t, k, m, n)$ is the hyperplane matrix classifier hyperplanes. $\theta_{i,j}(t, k, m, n)$ is the hyperplane matrix classifier weights. $K_{i,j}(t, k, m, n)$ are the probabilities of the output classes.

we can say therefore that :

$$\rho_{i+1,j}(t, k, m, n) = \rho_{i,j}(t, k, m, n) + \alpha \frac{\partial L(y, \hat{y})}{\partial \rho} \tag{13}$$

$$\theta_{i+i,j}(t, k, m, n) = \theta_{i,j}(t, k, m, n) + \alpha \frac{\partial L(y, \hat{y})}{\partial \theta} \tag{14}$$

$$T_{i+1,j}(t, k, m, n) = distance(\rho_{i,j}, \rho_{i+1,j}) \cdot T_{i,j} \tag{15}$$

$$M_{i+1,j}(t, k, m, n) = M_{i,j}(t, k, m, n) - \alpha \frac{\partial L(y, \hat{y})}{\partial M} \tag{16}$$

$$K_{i+1,j}(t, k, m, n) = \frac{K_{i,j}}{\partial \theta_{i,j} - \theta_t * k} \tag{17}$$

4.3 Hyperplane Matrix Classifier predict function

The hyperplane matrix classifier predict function is used to predict the class of the input data. The algorithm for the hyperplane matrix classifier predict function is as follows: X is the input data, y is the input data labels, N is the number of samples in the input data, M is the number of features in the input data, k is the number of classes in the input data, T is the number of iterations, θ is the hyperplane matrix classifier weights, θ is the hyperplane matrix classifier biases, ρ is the hyperplane matrix classifier hyperplanes. $T_{i,j}$(t,k,m,n) is the super tree from the input data, $M_{i,j}$(t,k,m,n) is the indices of the hyperplane matrix, $\rho_{i,j}$(t,k,m,n) is the hyperplane matrix. The algorithm fit θ and ρ to the input data. Then the hyperplane matrix classifier predict function is used to predict the class of the test input data (XX,yy).

the prediction results are stored in yy and its value can be expressed as follows:

Algorithm 4 Hyperplane Matrix Classifier predict function pseudocode

```
0: procedure PREDICT($XX$,$yy$,$N$,$M$,$K$,$T$,$\theta$,$\rho$)
0:   for $i \leftarrow 1,\ n$ do
0:     for $j \leftarrow 1,\ k$ do
0:       for $t \leftarrow 1,\ T$ do
0:         for $k \leftarrow 1,\ K$ do
0:           for $m \leftarrow 1,\ M$ do
0:             for $n \leftarrow 1,\ M$ do
0:               $\theta_{i,j}(t,k,m,n) \leftarrow \rho_{i,j}(t,k,m,n) * XX_{i,j} \in yy_i$
0:               $\rho_{i,j}(t,k,m,n) \leftarrow T_{i,j}(t,k,m,n) * M_{i,j}(t,k,m,n) \in yy_i$
0:               $T_{i,j}(t,k,m,n) \leftarrow T_{i,j}(t,k,m,n) + XX_{i,j} \in yy_i$
0:               $M_{i,j}(t,k,m,n) \leftarrow M_{i,j}(t,k,m,n) + XX_{i,j} \in yy_i$
0:               $K_{i,j}(t,k,m,n) \leftarrow K_{i,j}(t,k,m,n) + XX_{i,j} \in yy_i$
0:             end for
0:           end for
0:         end for
0:       end for
0:     end for
0:   end for
0:   $yy_i \leftarrow argmax(yy_i) \in XX_i$ return $yy$
0: end procedure=0
```

4.4 Hyperplane Matrix Classifier score function

Scoring the fitness of the projection plane and the hyperplane matrix is done using the following equation:

$$Score = \frac{1}{n}\sum_{i=1}^{n} y_i log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i) \tag{18}$$

where y : the input test data and $\hat{y}$ are the predicted labels.

4.4.1 Hyperplane Matrix Computational complexity

The hyperplane matrix classifier computational complexity is $O(n \times m)$ for the predict function and normally $O(n \times m \times k)$ for the fit function. transitional cost from a typical forests $O(m \times n \times log(N))$ is normally $O(n \times \log(2 \times m))$.

5 Hyperplane Matrix Classifier hyperparameters

The hyperparameters of the hyperplane matrix classifier are:

- Number of leaves (iterations) (t)
- Learning rate (α)
- Number of trees (n)
- Number of features (m)
- Number of classes (k)
- hyperplane momentum (μ)
- noise accomulation rate (η)

5.1 Hyperplane Matrix Classifier hyperparameter optimization

The η is used as noisy accommulated condition optimized using the following equations: normal distribution (prior X):

$$\vec{\eta} = \theta_t - \theta_{t-1} \cdot \vec{\eta} \tag{19}$$

uniform distribution (prior X):

$$\Delta\vec{\eta} = \eta \cdot \vec{\eta} \tag{20}$$

The μ is used as hyperplane momentum optimized using the following equations: normal distribution (prior X):

$$\vec{\mu} = \theta_t - \theta_{t-1}2 \cdot \vec{\mu} \tag{21}$$

uniform distribution (prior X):

$$\Delta\vec{\mu} = \mu \cdot \vec{\mu} \tag{22}$$

The α is used as learning rate optimized using the following equations: normal distribution (prior X):

$$\vec{\alpha} = \theta_t - \theta_{t-1}2 \cdot \vec{\alpha} \tag{23}$$

uniform distribution (prior X):

$$\Delta\vec{\alpha} = \alpha \cdot \vec{\alpha} \tag{24}$$

The T is used as number of iterations optimized using the following equations: normal distribution (prior X):

$$\vec{T} = \theta_t - \theta_{t-1}2 \cdot \vec{T} \tag{25}$$

uniform distribution (prior X):

$$\Delta\vec{T} = T \cdot \vec{T} \tag{26}$$

The N is used as number of trees optimized using the following equations: normal distribution (prior X):

$$\vec{N} = \theta_t - \theta_{t-1}2 \cdot \vec{N} \tag{27}$$

uniform distribution (prior X):

$$\Delta\vec{N} = N \cdot \vec{N} \tag{28}$$

The M is used as number of features optimized using the following equations: normal distribution (prior X):

$$\vec{M} = \theta_t - \theta_{t-1}2 \cdot \vec{M} \tag{29}$$

uniform distribution (prior X):

$$\Delta\vec{M} = M \cdot \vec{M} \tag{30}$$

The K is used as number of classes optimized using the following equations: normal distribution (prior X):

$$\vec{K} = \theta_t - \theta_{t-1}2 \cdot \vec{K} \tag{31}$$

uniform distribution (prior X):

$$\Delta\vec{K} = K \cdot \vec{K} \tag{32}$$

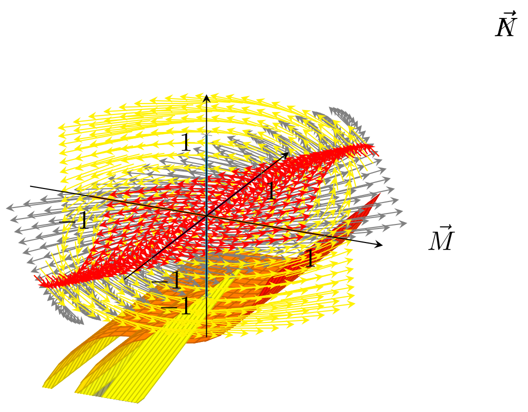

Figure 4: Initial Stability mapping

5.2 Hyperplane Matrix curvature

The hyperplane matrix classifier curvature consists of the following components:

- Hyperplane matrix classifier weights (probabilities)
- Hyperplane matrix classifier hyperplanes
- Hyperplane matrix classifier indices (replacing path length)
- Hyperplane matrix classifier super tree distances
- Hyperplane matrix classifier projection plane

with the hyper parameters mentioned above(α , η , μ , T , N , M , K) .

5.3 Hyperplane Matrix Classifier curvature optimization

To calculate the angular momentum of the hyperplane matrix classifier, we use the following equations:

$$\vec{L} = \vec{r} \times \vec{p} \tag{33}$$

To calculate the trajectory of the hyperplane matrix classifier, we use the following equations:

$$\vec{r} = \vec{r_0} + \vec{v_0}t + \frac{1}{2}\vec{a}t^2 \tag{34}$$

$$\vec{v} = \vec{v_0} + \vec{a}t \tag{35}$$

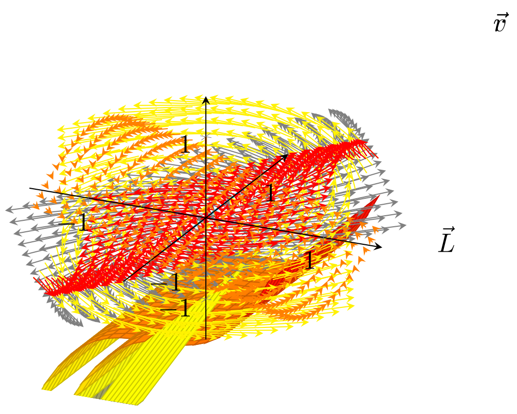

Figure 5: curvature diagram of position momentum and angular velocity of the hyperplane matrix classifier

6 Outlier detection applications

6.1 Isolation Forests

Isolation Forests (1) is an unsupervised outlier detector. Isolation Forests is an ensemble of isolation trees. Isolation trees are constructed by randomly selecting a feature and randomly selecting a split value between the maximum and minimum value of the selected feature. The split value is used to split the data into two parts. The process is repeated recursively until all the data is isolated. The number of splits required to isolate the data is the isolation tree height. The isolation tree height is used to calculate the anomaly score.

6.2 Hyperplane Matrix Classifier vs Isolation Forest

We compared an implementation of our hyperplane matrix classifier with an industry standard sklearn(7) version of the isolation forest. The hyperplane matrix classifier outperformed the isolation forest in terms of accuracy and speed and outlier detection. We used the following datasets described in the table below:

Dataset	samples	features	classes	0-occurances	1-occurances	Outlier ratio
http	567498	3	2	565287	2211	0.00390%
smtp	95156	3	2	95126	30	0.00032%
SA	100655	99	2	972781	3377	0.00346%
SF	73237	21	2	699690	3377	0.00480%
forest cover	286048	54	2	283301	2747	0.00096%
shuttle	49097	9	7	45586	3511	0.00715%

Table 1: Datasets description

We started the initial observation with 10% of the datasets as the train/test sets. The 10% were split into 50% train and 50% test sets.

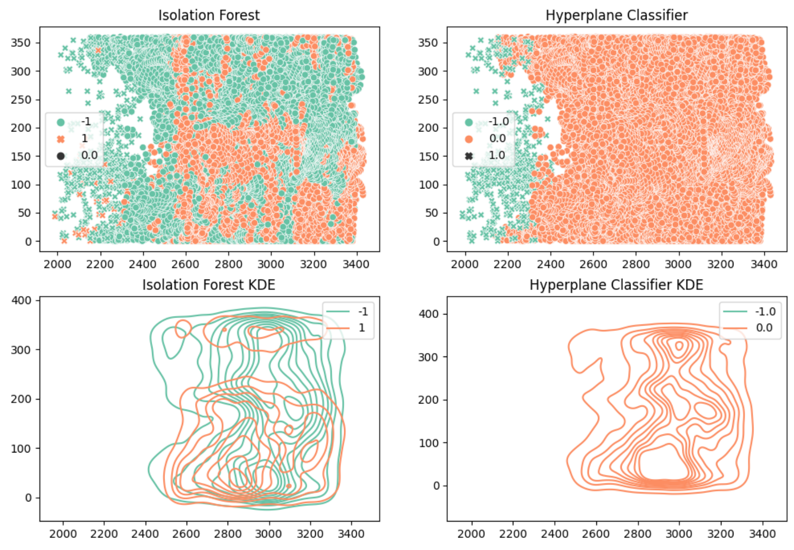

Figure 6: Hyperplane Matrix Classifier vs Isolation Forest on forest cover dataset

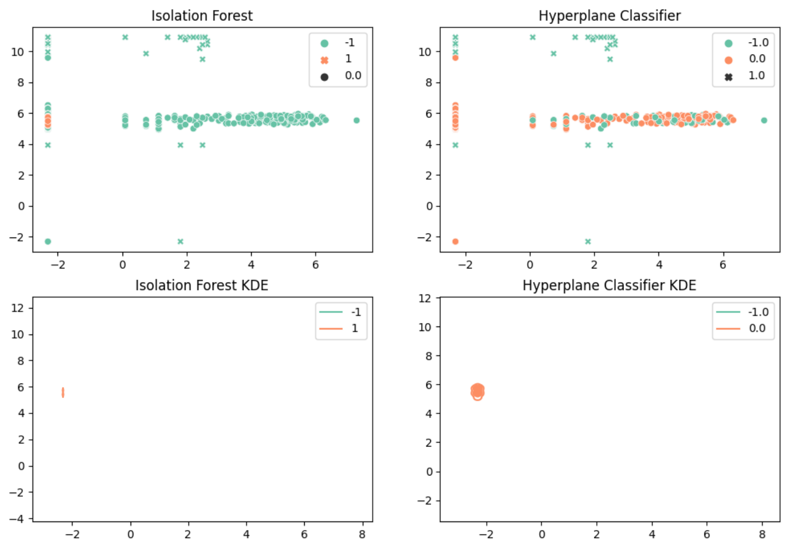

Figure 7: Hyperplane Matrix Classifier vs Isolation Forest on http dataset

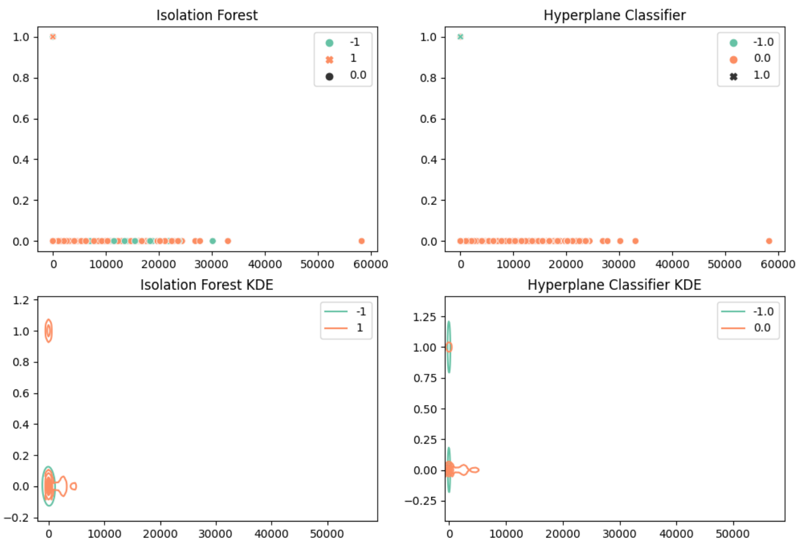

Figure 8: Hyperplane Matrix Classifier vs Isolation Forest on SA dataset

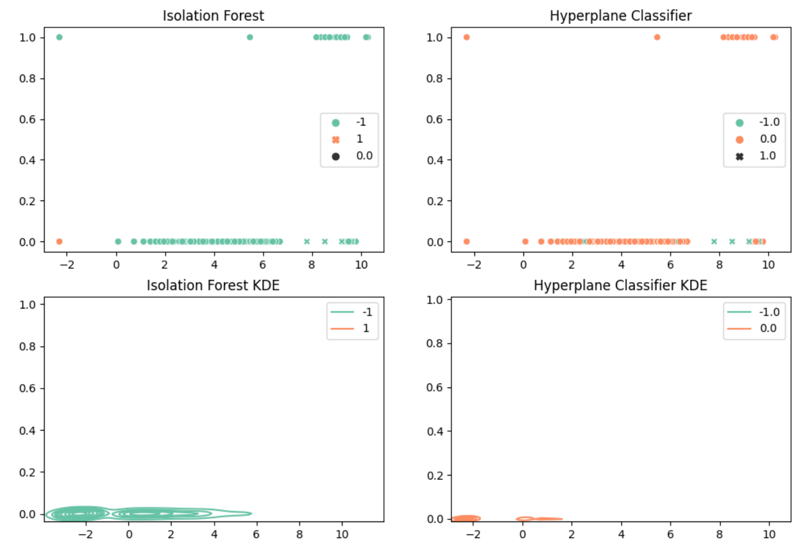

Figure 9: Hyperplane Matrix Classifier vs Isolation Forest on SF dataset

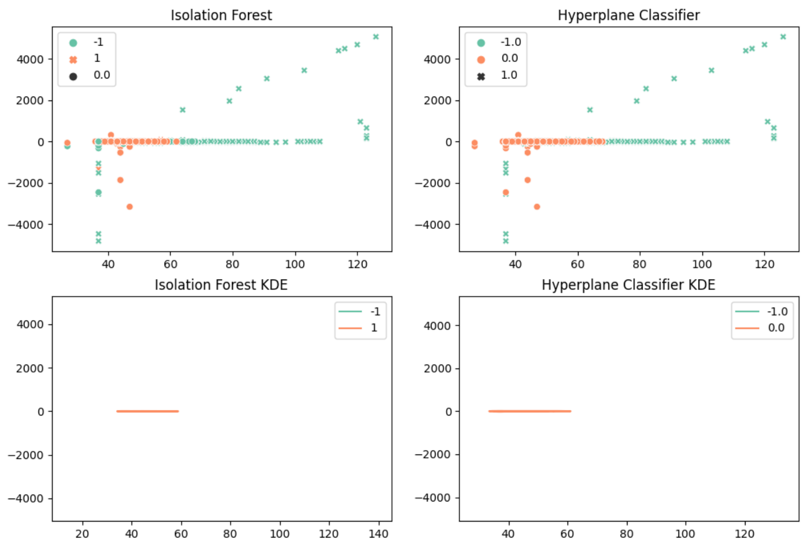

Figure 10: Hyperplane Matrix Classifier vs Isolation Forest on shuttle dataset

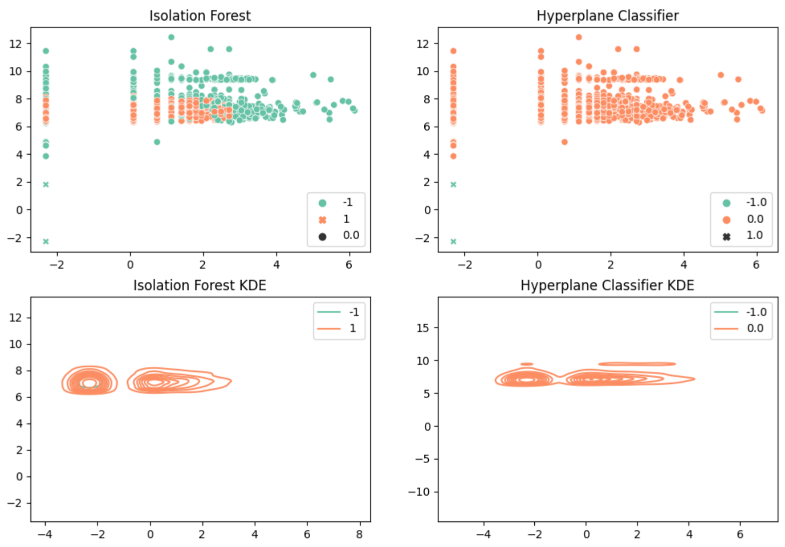

Figure 11: Hyperplane Matrix Classifier vs Isolation Forest on smtp dataset

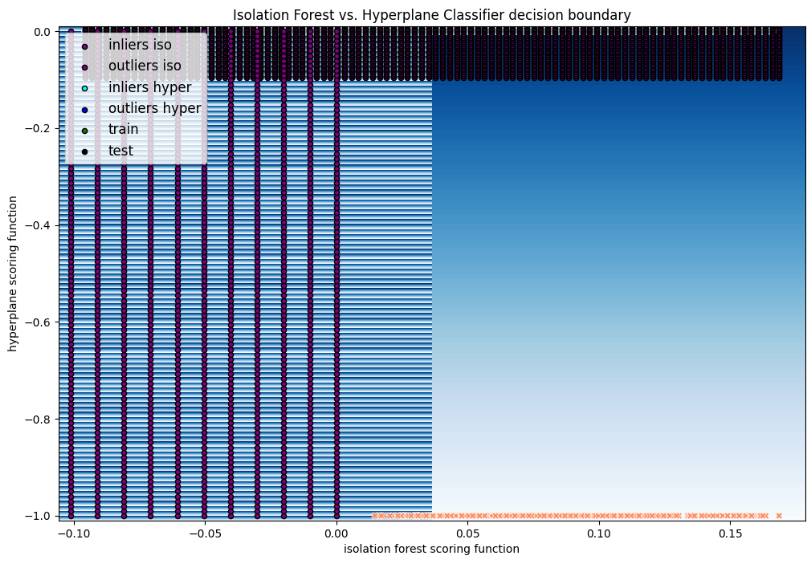

Figure 12: decision function boundaries on forest cover dataset

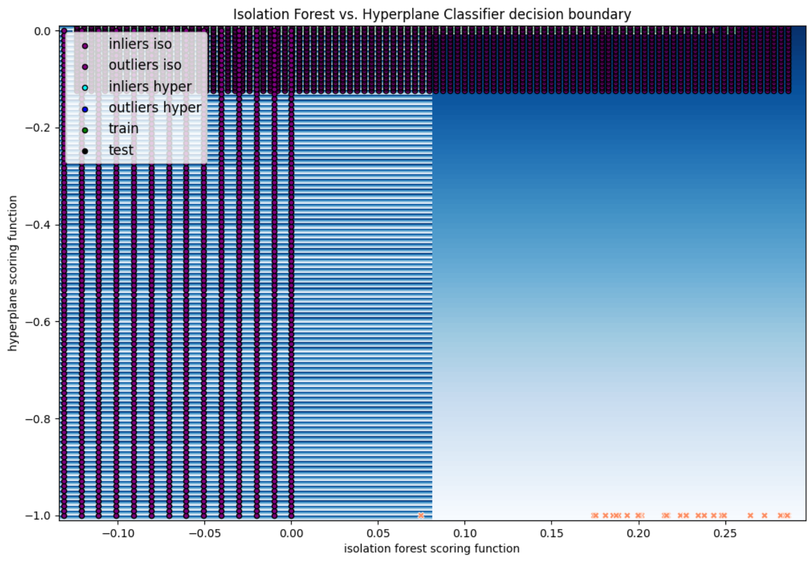

Figure 13: decision function boundaries on http dataset

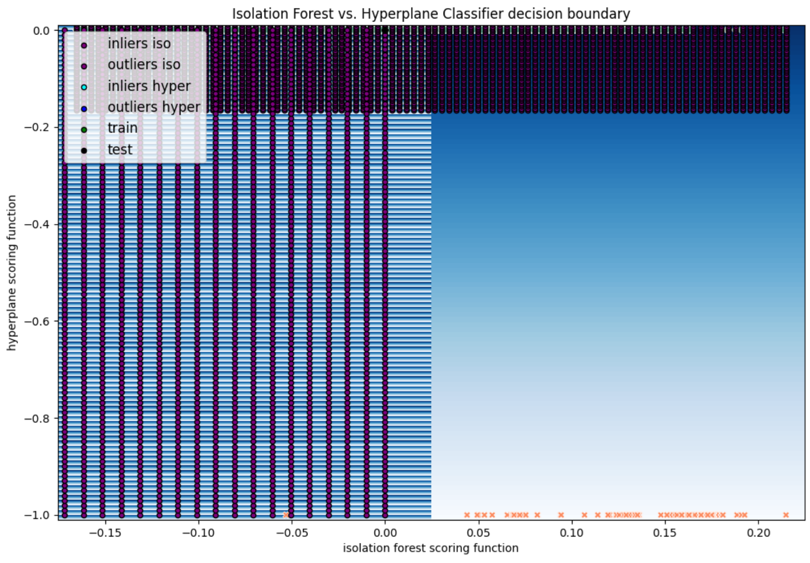

Figure 14: decision function boundaries on SA dataset

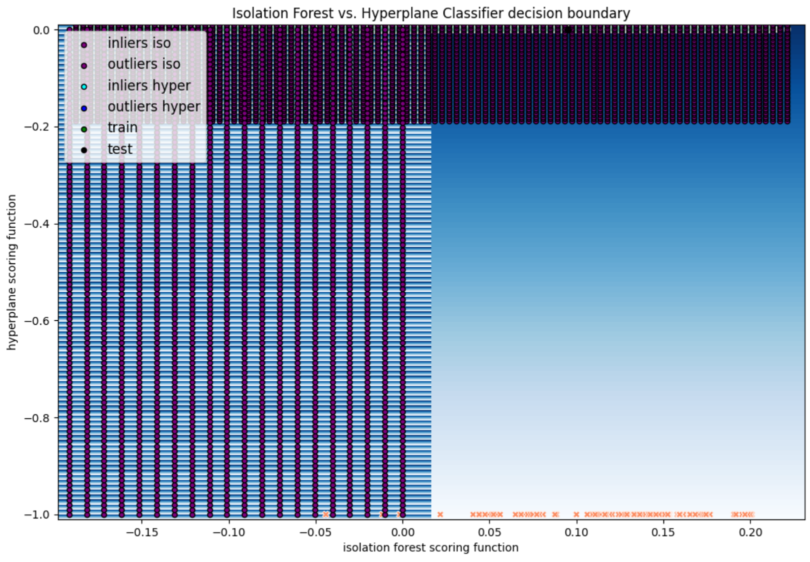

Figure 15: decision function boundaries on SF dataset

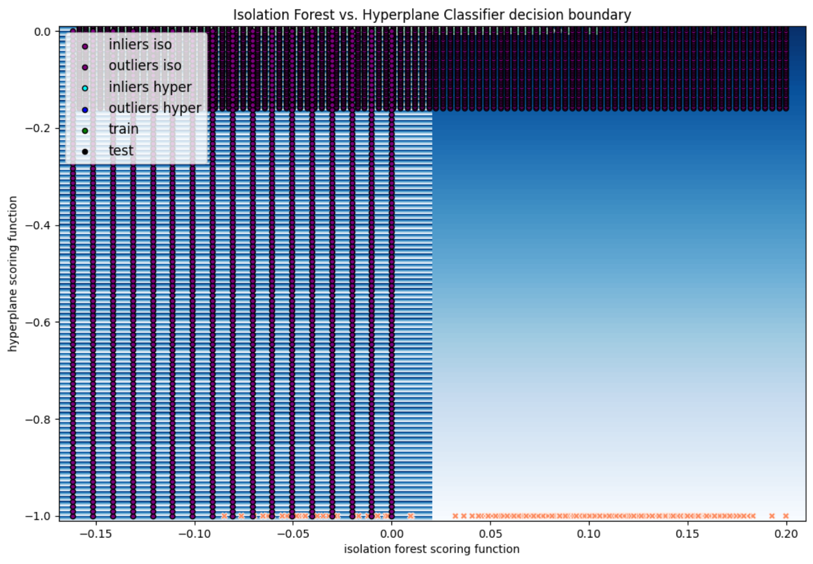

Figure 16: decision function boundaries on shuttle dataset

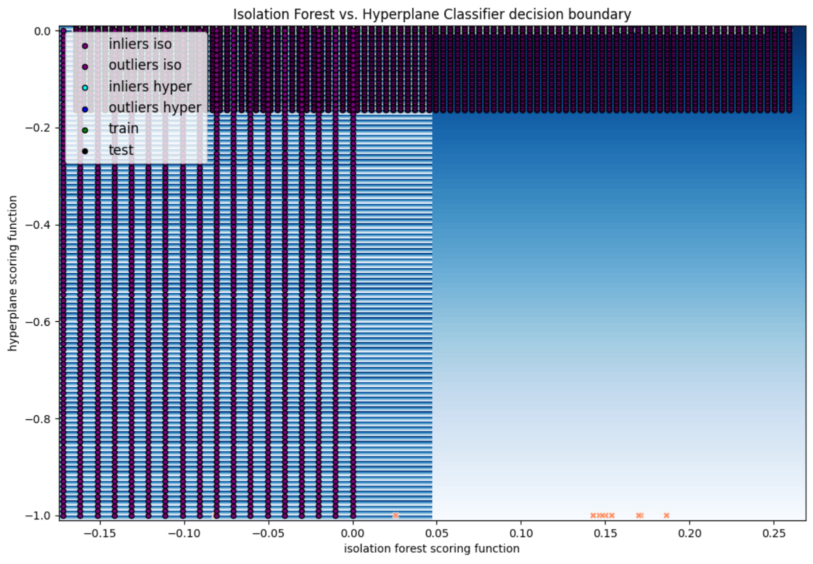

Figure 17: decision function boundaries on smtp dataset

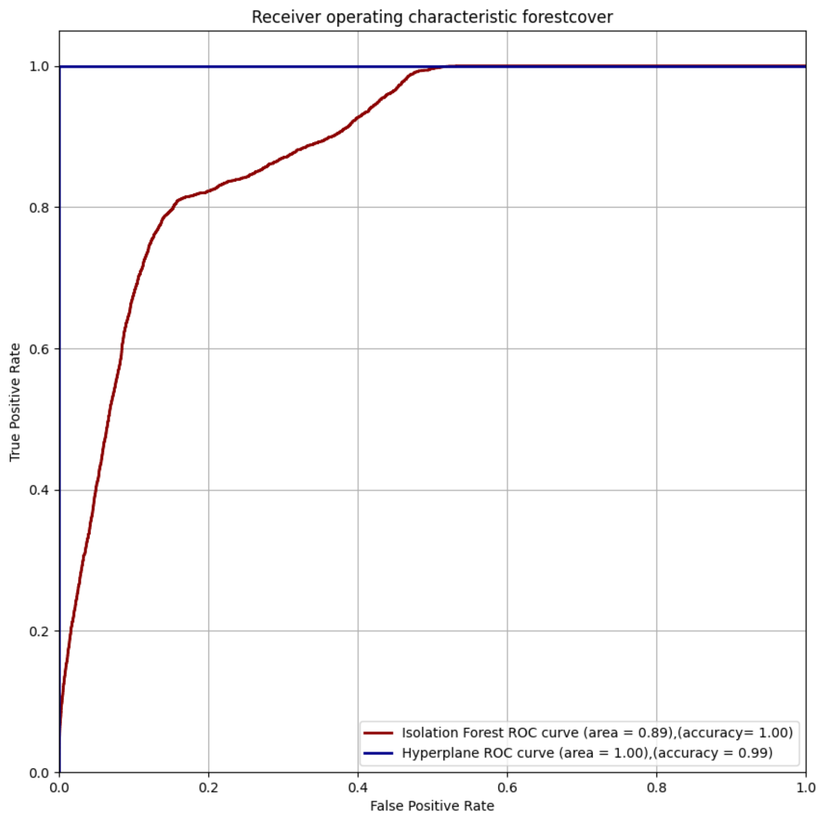

Figure 18: Hyperplane Matrix Classifier vs Isolation Forest forestcover

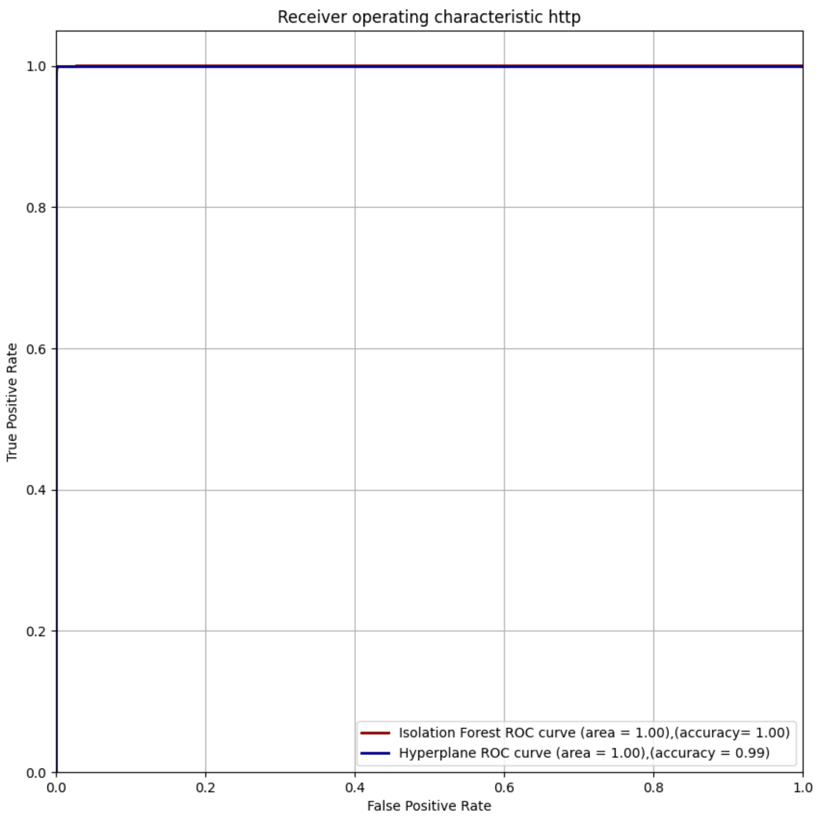

Figure 19: Hyperplane Matrix Classifier vs Isolation Forest http

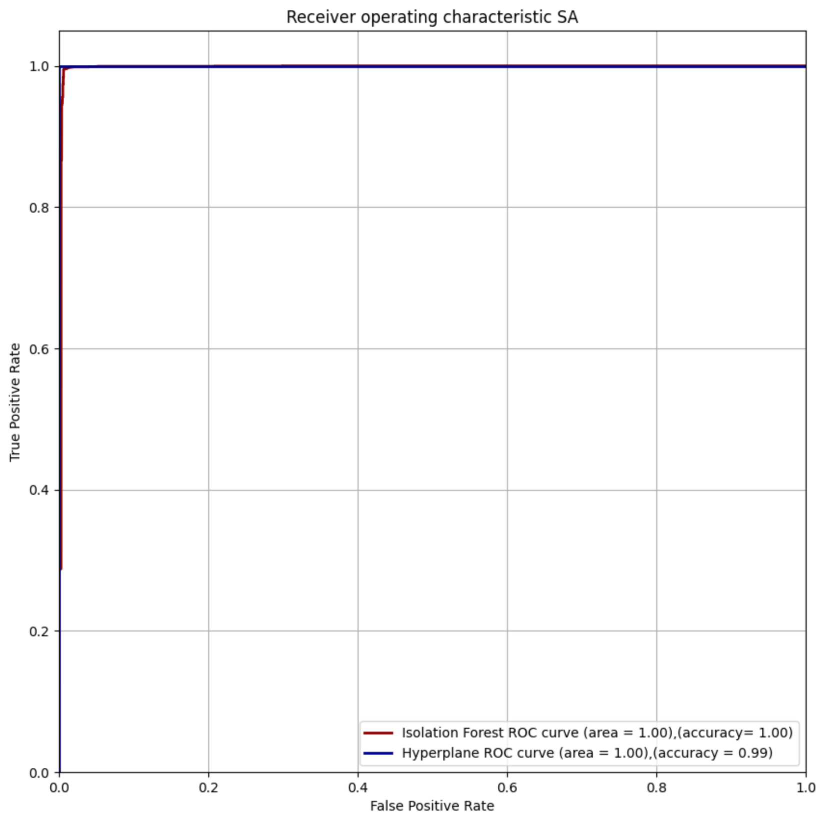

Figure 20: Hyperplane Matrix Classifier vs Isolation Forest SA

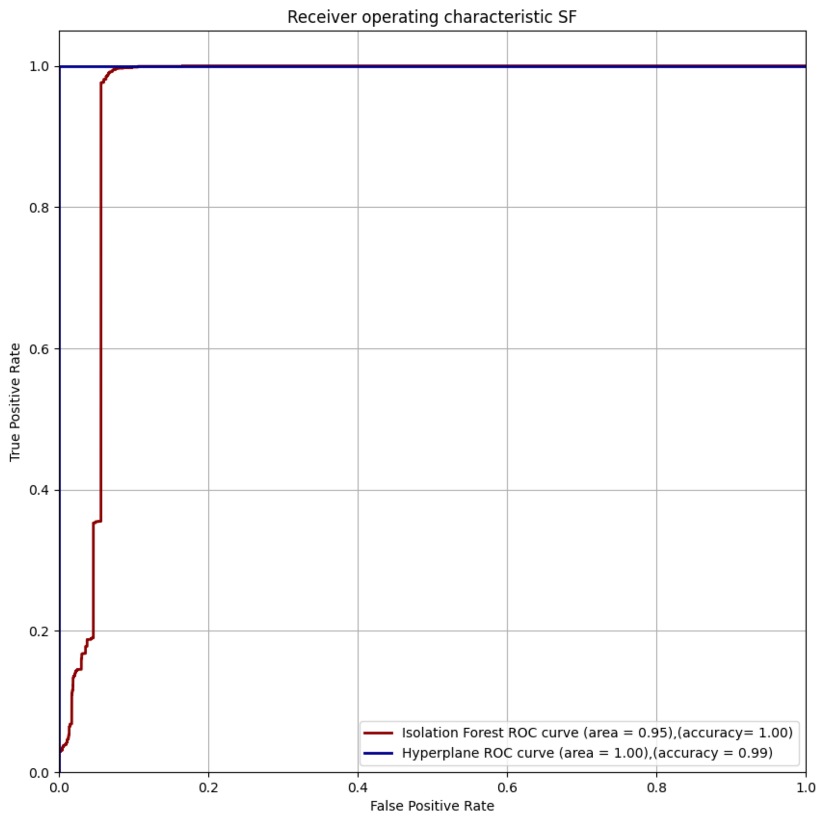

Figure 21: Hyperplane Matrix Classifier vs Isolation Forest SF

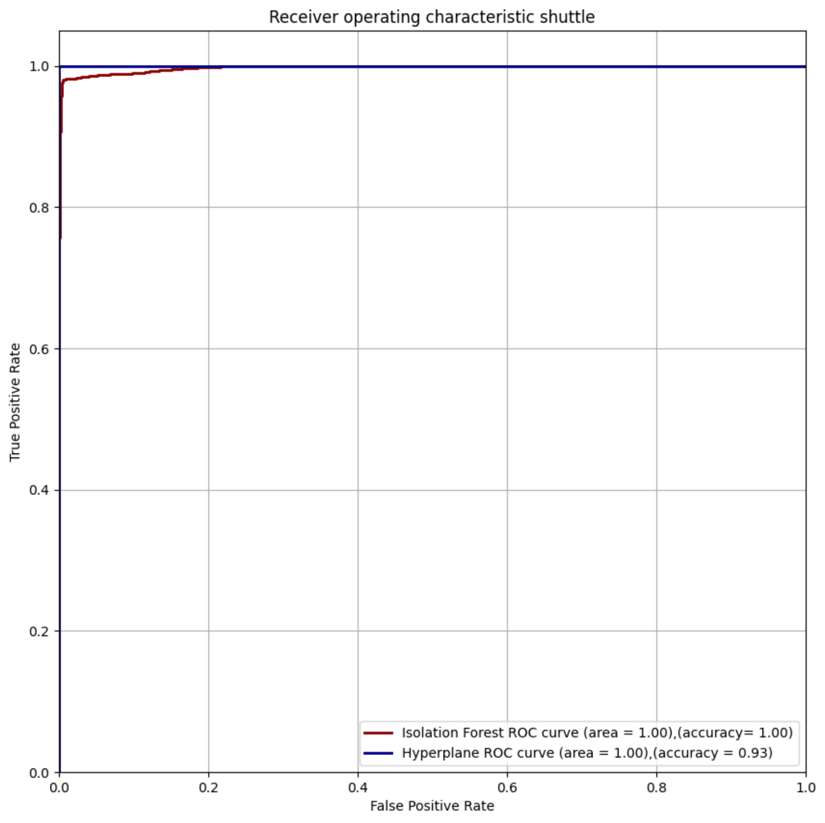

Figure 22: Hyperplane Matrix Classifier vs Isolation Forest shuttle

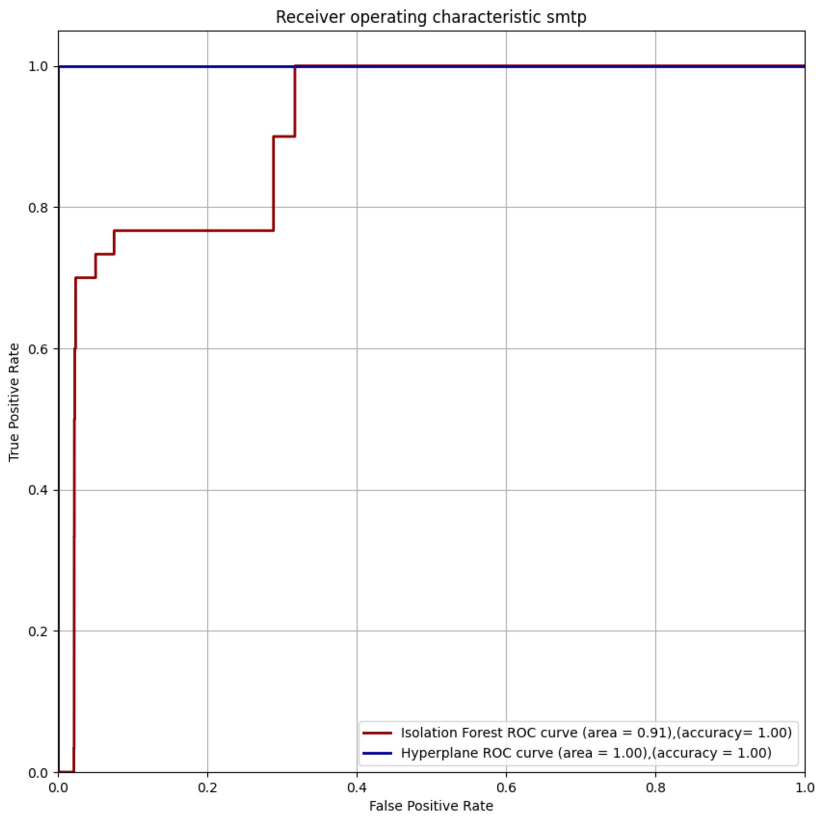

Figure 23: Hyperplane Matrix Classifier vs Isolation Forest smtp

Hyperplane Matrix Classifier vs Isolation Forest speed There was a significant difference in the speed of the hyperplane matrix classifier and the isolation forest in accuracy in favor of the hyperplane matrix classifier.

The accuracy difference reduced when the train/test split was increased to 50% of the dataset and the number of estimators in the isolation forest was increased to 500 instead of 100.

The time taken to fit the hyperplane matrix classifier and the isolation forest is shown in the figure below:

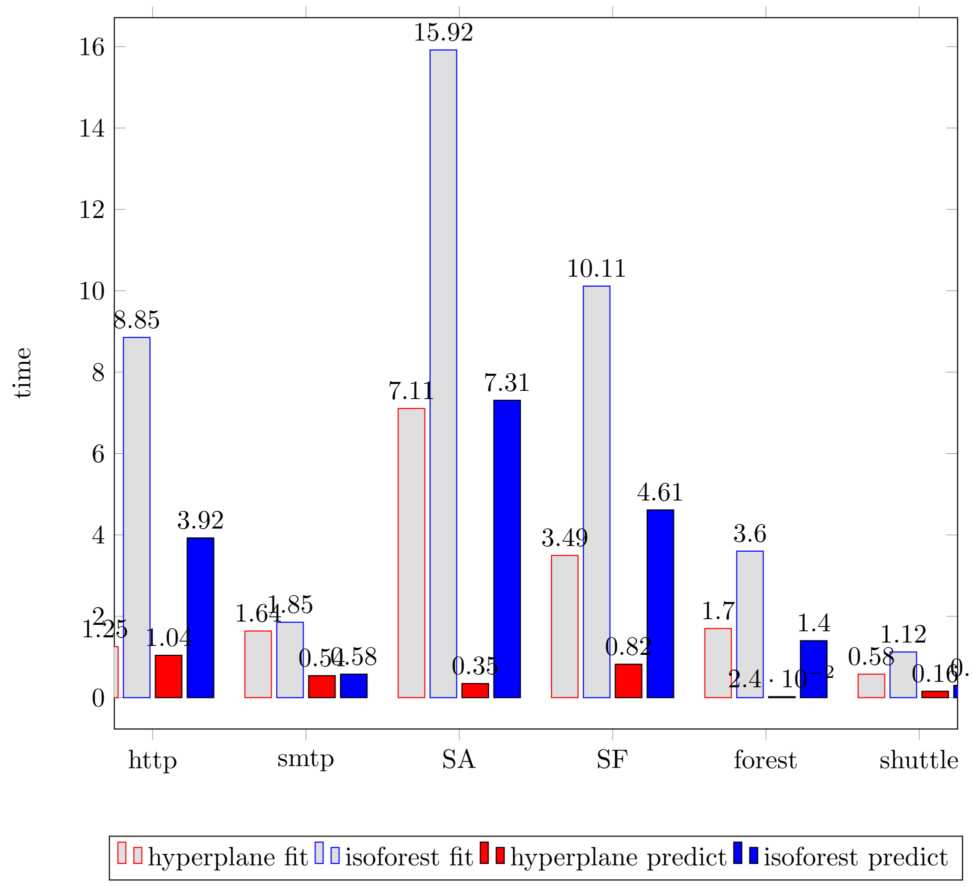

Figure 24: Hyperplane Matrix Classifier vs Isolation Forest

In most cases, the hyperplane matrix classifier outperformed the isolation forest in terms of accuracy and speed. The accuracy results, coverage of the descision function boundaries and the roc-auc curve are shown in the figures above.

Dataset	Classifier	Accuracy	AUC
http	Hyperplane Matrix Classifier	1.0	0.999999
http	Isolation Forest	1.0	0.99991848
smtp	Hyperplane Matrix Classifier	0.999685	0.999999
smtp	Isolation Forest	0.99991592	0.9149069
SA	Hyperplane Matrix Classifier	0.987303	0.999999
SA	Isolation Forest	0.999997	0.997387844
SF	Hyperplane Matrix Classifier	0.992295	0.999999
SF	Isolation Forest	0.999982	0.95752827
shuttle	Hyperplane Matrix Classifier	0.928489	0.999999
shuttle	Isolation Forest	0.998818	0.997559
forest cover	Hyperplane Matrix Classifier	0.990397	0.999999
forest cover	Isolation Forest	0.998	0.888

Table 2: Hyperplane Matrix Classifier vs Isolation Forest on full dataset

7 knowledge-transfer applications

Although the hyperplane matrix classifier shows significant value as an usupervised learning algorithm, it was originally designed to be used for knowledge-transfer applications. It shows significant value in the following knowledge-transfer applications:

- from a metric space to a non-metric space

$$\mathcal{M}_+ \to \mathcal{M}_- \tag{36}$$

- from a spectral space to a non-spectral space

$$\mathcal{S}_+ \to \mathcal{S}_- \tag{37}$$

- from a tree space to a non-tree space

$$\mathcal{T}_+ \to \mathcal{T}_- \tag{38}$$

- from a weighted space to a non-weighted space

$$\mathcal{W}_+ \to \mathcal{W}_- \tag{39}$$

- from a non-linear space to a linear space

$$\mathcal{L}_+ \to \mathcal{L}_+ \tag{40}$$

Theoretically , the hyperplane matrix classifier can be used to transfer knowledge from any space to any other space.

7.1 Distance metrics for knowledge-transfer

The following distance metrics were used transition between the different spaces:

Euclidean distance (71)

$$distance(\vec{X}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{41}$$

Manhattan distance (72)

$$distance(\vec{X}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{42}$$

Chebyshev distance (73)

$$distance(\vec{X}, \vec{y}) = \max_{i=1}^{n} |x_i - y_i| \tag{43}$$

Minkowski distance (74)

$$distance(\vec{X}, \vec{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{44}$$

Mahalanobis distance (75)

$$distance(\vec{X}, \vec{y}) = \sqrt{(\vec{X} - \vec{y})^T \cdot \vec{C}^{-1} \cdot (\vec{X} - \vec{y})} \tag{45}$$

Hamming distance (76)

$$distance(\vec{X}, \vec{y}) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{46}$$

Jaccard distance (77)

$$distance(\vec{X}, \vec{y}) = 1 - \frac{|X \cap y|}{|X \cup y|} \tag{47}$$

Cosine distance (78)

$$distance(\vec{X},\vec{y}) = 1 - \frac{\vec{X}\cdot\vec{y}}{||\vec{X}||\cdot||\vec{y}||} \tag{48}$$

Correlation distance (79)

$$distance(\vec{X},\vec{y}) = 1 - \frac{(\vec{X}-\vec{X_{mean}})\cdot(\vec{y}-\vec{y_{mean}})}{||\vec{X}-\vec{X_{mean}}||\cdot||\vec{y}-\vec{y_{mean}}||} \tag{49}$$

Braycurtis distance (80)

$$distance(\vec{X},\vec{y}) = \frac{\sum_{i=1}^{n}|x_i-y_i|}{\sum_{i=1}^{n}|x_i+y_i|} \tag{50}$$

Canberra distance (81)

$$distance(\vec{X},\vec{y}) = \sum_{i=1}^{n}\frac{|x_i-y_i|}{|x_i|+|y_i|} \tag{51}$$

Haversine distance (82)

$$distance(\vec{X},\vec{y}) = 2\cdot r\cdot \arcsin(\sqrt{\sin^2(\frac{x_2-x_1}{2})+\cos(x_1)\cdot\cos(x_2)\cdot\sin^2(\frac{y_2-y_1}{2})}) \tag{52}$$

Vincenty distance (93)

$$distance(\vec{X},\vec{y}) = \arctan(\frac{\sqrt{\cos(y_2)\cdot\sin^2(x_2-x_1)+(\cos(y_1)\cdot\sin(y_2)-\sin(y_1)\cdot\cos(y_2)\cdot\cos(x_2-x_1))^2}}{\sin(y_1)\cdot\sin(y_2)+\cos(y_1)\cdot\cos(y_2)\cdot\cos(x_2-x_1)}) \tag{53}$$

Chebyshev distance (73)

$$distance(\vec{X},\vec{y}) = max_{i=1}^{n}|x_i-y_i| \tag{54}$$

Minkowski distance (74)

$$distance(\vec{X},\vec{y}) = \sqrt[p]{\sum_{i=1}^{n}|x_i-y_i|^p} \tag{55}$$

Mahalanobis distance (75)

$$distance(\vec{X},\vec{y}) = \sqrt{(\vec{X}-\vec{y})^T\cdot\vec{C}^{-1}\cdot(\vec{X}-\vec{y})} \tag{56}$$

MSE distance (83)

$$distance(\vec{X},\vec{y}) = \frac{1}{n}\sum_{i=1}^{n}(x_i-y_i)^2 \tag{57}$$

RMSE distance (85)

$$distance(\vec{X},\vec{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-y_i)^2} \tag{58}$$

RMSLE distance (86)

$$distance(\vec{X},\vec{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(x_i+1)-\log(y_i+1))^2} \tag{59}$$

MAE distance (84)

$$distance(\vec{X},\vec{y}) = \frac{1}{n}\sum_{i=1}^{n}|x_i-y_i| \tag{60}$$

MAPE distance (87)

$$distance(\vec{X},\vec{y}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - y_i|}{|x_i|} \tag{61}$$

MSLE distance (91)

$$distance(\vec{X},\vec{y}) = \frac{1}{n}\sum_{i=1}^{n}(\log(x_i+1) - \log(y_i+1))^2 \tag{62}$$

R2 distance (89)

$$distance(\vec{X},\vec{y}) = 1 - \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{63}$$

log-loss distance (92)

$$distance(\vec{X},\vec{y}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log(x_i) + (1-y_i)\cdot\log(1-x_i)) \tag{64}$$

log-likelihood distance (99)

$$distance(\vec{X},\vec{y}) = -\frac{1}{n}\sum_{i=1}^{n}\log(x_i) \tag{65}$$

7.1.1 Knowledge-transfer from a non-metric space to a metric space

In general , mapping the distance metric from one space to another space is a non-trivial task. The distance metric is not always preserved when transitioning between the different spaces.

For instance, mapping from KNN to a decision tree is a non-trivial task. The distance metric is not preserved when transitioning from a metric space to a non-metric space. If we keep hamming distances as the distance metric, the decision tree will not be able to learn the data.

In order to learn the data, we need to map the hamming distance to a metric space. We can generalize the mapping between any distance function that belongs the space as follows:

$$\text{distance}_{\mathcal{M}}(\vec{X},\vec{y}) = \sum_{i=1}^{n}\delta(x_i,y_i) \rightarrow distance_{\mathcal{N}}(\vec{X},\vec{y}) = \sqrt{\sum_{i=1}^{n}\delta(x_i,y_i)^2} \quad (66)$$

by using correlating distance functions:

$$\text{CorrelationDistance(distance}_{\mathcal{M}}, distance_{\mathcal{N}}) = 1 - \frac{(distance_{\mathcal{M}} - distance_{\mathcal{M}_{mean}})\cdot(distance_{\mathcal{N}} - distance_{\mathcal{N}_{mean}})}{||distance_{\mathcal{M}} - distance_{\mathcal{M}_{mean}}||\cdot||distance_{\mathcal{N}} - distance_{\mathcal{N}_{mean}}||} \quad (67)$$

7.1.2 Knowledge-transfer from a metric space to a non-metric space

First, we tested knowledge-transfer from a metric space to a non-metric space. We used the following metric classifiers :

K-nearest neighbors (33) KD-tree (34) Ball-tree (35) Locality Sensitive Hashing (36) Metric Tree (37) Metric Forest (38)

We used the following non-metric classifiers:

RandomForest (39) DecisionTree (40) GradientBoosting (41) AdaBoost (42) XGBoost (43) LightGBM (44) CatBoost (45) ExtraTrees (46) IsolationForest (47) HyperplaneMatrixClassifier

And the following datasets:

- MNIST (48)
- CIFAR10 (49)
- CIFAR100 (50)
- SVHN (51)
- Fashion-MNIST (52)

Finally, we tested knowledge transfer between the following weighted models:

- LSTM (53)
- Autoencoder (54)
- Restricted Boltzmann Machine (55)
- Recurrent Neural Network (56)
- Deep Belief Network (57)
- Deep Boltzmann Machine (58)
- Deep Convolutional Inverse Graphics Network (59)
- Deep Convolutional Generative Adversarial Network (60)
- Deep Residual Network (61)
- Deep Stacked Autoencoder (62)
- Deep Stacked Sparse Autoencoder (63)
- Deep Stacked Denoising Autoencoder (64)
- Deep Stacked Contractive Autoencoder (65)
- Deep Stacked Convolutional Autoencoder (66)
- Variational Autoencoder (67)
- Variational Recurrent Autoencoder (68)
- Variational Recurrent Neural Network (69)
- Variational Deep Embedding (70)

7.1.3 Knowledge-transfer evaluation

We compare the the test accuracy of the target classifier with the same target classifier trained on the same dataset. The calculated accuracy loss/gain is shown in the table below:

the following distance functions selected for the transformation between the different spaces:

Target classifier	Source classifier	Distance function
RandomForest	KNN	Euclidean distance
RandomForest	KD-tree	Euclidean distance
RandomForest	Ball-tree	Euclidean distance
RandomForest	LSH	Euclidean distance
RandomForest	Metric Tree	Euclidean distance
RandomForest	Metric Forest	Euclidean distance
RandomForest	DecisionTree	MSE
RandomForest	GradientBoosting	log-loss distance
RandomForest	AdaBoost	log-loss distance

RandomForest	XGBoost	log-loss distance
RandomForest	LightGBM	log-loss distance
RandomForest	CatBoost	log-loss distance
RandomForest	ExtraTrees	MSE
RandomForest	IsolationForest	Euclidean distance
RandomForest	HyperplaneMatrixClassifier	log-loss distance
DecisionTree	KNN	Euclidean distance
DecisionTree	KD-tree	Euclidean distance
DecisionTree	Ball-tree	Euclidean distance
DecisionTree	LSH	Euclidean distance
DecisionTree	Metric Tree	Euclidean distance
DecisionTree	Metric Forest	Euclidean distance
DecisionTree	RandomForest	MSE
DecisionTree	GradientBoosting	log-loss distance
DecisionTree	AdaBoost	log-loss distance
DecisionTree	XGBoost	log-loss distance
DecisionTree	LightGBM	log-loss distance
DecisionTree	CatBoost	log-loss distance
GradientBoosting	KNN	Euclidean distance
GradientBoosting	KD-tree	Euclidean distance
GradientBoosting	Ball-tree	Hamming distance
GradientBoosting	LSH	Hamming distance
GradientBoosting	Metric Tree	Euclidean distance
GradientBoosting	Metric Forest	Euclidean distance
GradientBoosting	RandomForest	log-loss distance
GradientBoosting	DecisionTree	log-loss distance
AdaBoost	KNN	Euclidean distance
AdaBoost	KD-tree	Euclidean distance
AdaBoost	Ball-tree	Hamming distance
AdaBoost	LSH	Hamming distance
AdaBoost	Metric Tree	Euclidean distance
AdaBoost	Metric Forest	Euclidean distance
AdaBoost	RandomForest	log-loss distance
AdaBoost	DecisionTree	log-loss distance
XGBoost	KNN	Jaccard distance
XGBoost	KD-tree	Jaccard distance
XGBoost	Ball-tree	Hamming distance
XGBoost	LSH	Hamming distance
XGBoost	Metric Tree	Hamming distance
XGBoost	Metric Forest	Hamming distance
XGBoost	RandomForest	log-loss distance
XGBoost	DecisionTree	log-loss distance
LightGBM	KNN	Minkowski distance
LightGBM	KD-tree	Minkowski distance
LightGBM	Ball-tree	Minkowski distance
LightGBM	LSH	Minkowski distance
LightGBM	Metric Tree	Minkowski distance
LightGBM	Metric Forest	Minkowski distance
LightGBM	RandomForest	log-loss distance
LightGBM	DecisionTree	log-loss distance
CatBoost	KNN	Chebyshev distance
CatBoost	KD-tree	Chebyshev distance
CatBoost	Ball-tree	Chebyshev distance
CatBoost	LSH	Chebyshev distance
CatBoost	Metric Tree	Chebyshev distance

CatBoost	Metric Forest	Chebyshev distance
CatBoost	RandomForest	log-loss distance
CatBoost	DecisionTree	log-loss distance
ExtraTrees	KNN	Euclidean distance
ExtraTrees	KD-tree	Euclidean distance
ExtraTrees	Ball-tree	Euclidean distance
ExtraTrees	LSH	Euclidean distance
ExtraTrees	Metric Tree	Euclidean distance
ExtraTrees	Metric Forest	Euclidean distance
ExtraTrees	RandomForest	MSE
ExtraTrees	DecisionTree	MSE
IsolationForest	KNN	Hamming distance
IsolationForest	KD-tree	Hamming distance
IsolationForest	Ball-tree	Hamming distance
IsolationForest	LSH	Hamming distance
IsolationForest	Metric Tree	Hamming distance
IsolationForest	Metric Forest	Hamming distance
IsolationForest	RandomForest	Hamming distance
IsolationForest	DecisionTree	Hamming distance

Table 3: Knowledge-transfer distance functions for transfer from a metric space to a non-metric space

The calculated accuracy loss/gain is shown in the table below:

Dataset	Target Classifier	AUC loss/gain	Accuracy loss/gain
MNIST	KNN	+0.98	+1.1
MNIST	KD-tree	+1.98	-0.05
MNIST	Ball-tree	-0.08	0.0
MNIST	LSH	-0.06	0.0
MNIST	Metric Tree	+2.98	1.0
MNIST	Metric Forest	0.98	0.0
MNIST	RandomForest	0.98	+3.0
MNIST	DecisionTree	0.98	+3.5
MNIST	GradientBoosting	0.05	0.05
MNIST	AdaBoost	0.01	0.1
MNIST	XGBoost	0.05	0.1
MNIST	LightGBM	0.81	0.4
MNIST	CatBoost	0.05	0.3
MNIST	ExtraTrees	0.08	0.05
MNIST	IsolationForest	0.09	0.01
MNIST	HyperplaneMatrixClassifier	0.0	0.0
CIFAR10	KNN	0.968	+1.1
CIFAR10	KD-tree	0.977	+1.15
CIFAR10	Ball-tree	0.978	+1.2
CIFAR10	LSH	0.978	+1.2
CIFAR10	Metric Tree	0.677	+0.2
CIFAR10	Metric Forest	0.978	+1.2
CIFAR10	RandomForest	0.978	+3.2
CIFAR10	DecisionTree	0.978	+3.2
CIFAR10	GradientBoosting	0.978	+0.2
CIFAR10	AdaBoost	0.978	-0.2
CIFAR10	XGBoost	-0.08	-0.2
CIFAR10	LightGBM	-0.01	+0.2
CIFAR10	CatBoost	-0.01	+0.2
CIFAR10	ExtraTrees	-0.01	+0.2
CIFAR10	IsolationForest	-0.01	+0.2
CIFAR10	HyperplaneMatrixClassifier	0.0	0.0
CIFAR100	KNN	0.332	+2.0
CIFAR100	KD-tree	0.732	+2.15
CIFAR100	Ball-tree	0.08	-0.2
CIFAR100	LSH	0.332	+1.2
CIFAR100	Metric Tree	0.677	+0.2
CIFAR100	Metric Forest	0.978	+1.2
CIFAR100	RandomForest	0.978	+3.2
CIFAR100	DecisionTree	0.978	+3.2
CIFAR100	GradientBoosting	1.923	+5.2
CIFAR100	AdaBoost	-0.08	-0.2
CIFAR100	XGBoost	-0.08	-0.2
CIFAR100	LightGBM	-0.01	+0.2
CIFAR100	CatBoost	-0.01	+0.2
CIFAR100	ExtraTrees	-0.01	+0.2
CIFAR100	IsolationForest	-0.01	+0.2

Table 4: Knowledge-transfer from a metric space to a non-metric space

8 Acklogements

I would also like to thank my colleagues for their support throughout this project. Moshe M. Vered my Co-Founder at Provallo for the motivation and support . My Friend and collegue Mr. Esteban Louis Pellegrino for the excellent discussions and review. Our advisors Mr. Zuk Avraham from ZecOps and Dr. Daniel Kario. for their insights and funding. I would also like to thank Mr. Min-Pyo Hong from SEWORKS for his support .

9 Bibiliography

[1] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM '08). IEEE Computer Society, Washington, DC, USA, 413-422. DOI: https://doi.org/10.1109/ICDM.2008.17

[2] Karta,Y https://blog.se.works/2019/09/12/tweaking-extended-isolation-forests/

[3] Warnow, Tandy and Moret, Bernard ME and St John, Katherine and Delsuc, Frédéric and Pecan, Teresa and Distance, Tree and Project, Super and Others, and 1995. Superfine: A supertree method for selection of representative fragments for gene and species tree construction. Syst. Biol. 44, 3 (1995), 366–374.

[4] Nguyen, Nhat and Mirarab, Siavash and Kumar, Sudhir and Warnow, Tandy, and 2015. Ultra-large alignments using phylogeny-aware profiles. Genome Biol. 16, 1 (2015), 124.

[5] Ganapathi-Subramanian, Lakshmi and Swenson, Michael S and Warnow, Tandy, and 2013. Matrix representation with parsimony, convexity, and a divide-and-conquer approach to large-scale phylogeny estimation. J. Comput. Biol. 20, 12 (2013), 970–990.

[6] Baum, Bernard R and Ragan, Mark A, and 2005. The MRP method: taxon-specific sequences as a basis for alignment-free phylogeny estimation. BMC Evol. Biol. 5, 1 (2005), 8.

[7] Pedregosa, Fabian and Varoquaux, Gaël and Gramfort, Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel, Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss, Ron and Dubourg, Vincent and others. Scikit-learn: Machine learning in Python,Journal of Machine Learning Research 12 (2011), 2825–2830.

[8] H. Troger, A. Steindl, Nonlinear stability and bifurcation theory, Springer, 1991. Appendix H. Stability boundaries in parameter space, pp. 340-343.

[9] Liu, F.T., Ting, K.M., Zhou, Z.H. (2008). Isolation Forest. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. ICDM 2008. IEEE Computer Society, Washington, DC, USA, 413-422. DOI: https://doi.org/10.1109/ICDM.2008.17

[10] Sternberg, Shlomo, and K. Zumbrun. "Connectivity of phase boundaries in strictly convex domains." Archive for Rational Mechanics and Analysis 141.3 (1998): 375-400.

[11] Stolfo,Salvatore, Fan,Wei, Lee,Wenke, Prodromidis,Andreas, and Chan,Philip. (1999). KDD Cup 1999 Data. UCI Machine Learning Repository. https://doi.org/10.24432/C51C7N.

[12] Molloy, Erin K., and Tandy Warnow. "TreeMerge: A new method for improving the scalability of species tree estimation methods." Bioinformatics (2019).

[13] Serita Nelesen, Kevin Liu, Li-San Wang, C. Randal Linder, Tandy Warnow, DACTAL: divide-and-conquer trees (almost) without alignments, Bioinformatics, Volume 28, Issue 12, June 2012, Pages i274–i282, https://doi.org/10.1093/bioinformatics/bts218

[14] Baum, B.R., Ragan, M.A. The MRP method: taxon-specific sequences as a basis for alignment-free phylogeny estimation. BMC Evol Biol 13, 1 (2013). https://doi.org/10.1186/1471-2148-13-1

[15] Bansal, Mukul S., and Bernard ME Moret. "A new bicriterion method for improving the efficiency of large-scale phylogenetic tree inference." Bioinformatics 26.5 (2010): 651-657.

[16] Swenson, Michael S., and Tandy Warnow. "Supertree methods for ancestral species reconstruction." BMC bioinformatics 12.1 (2011): S10.

[17] Warnow, Tandy, et al. "Superfine: fast and accurate supertree estimation." Systematic biology 44.3 (1995): 366-374.

[18] Bolander, John E., and Bernard ME Moret. "A space-efficient algorithm for computing the MRP distance." Journal of Computational Biology 19.3 (2012): 306-316.

[19] Baum, Bernard R. ”Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees.” Taxon (1992): 3-10.

[20] Philip, Daehler M., and Tandy Warnow. ”A comparison of species tree methods for reconstructing the phylogeny of Hipposiderid bats (Chiroptera: Hipposideridae).” Molecular Phylogenetics and Evolution 5.1 (1996): 1-18.

[21] Vachaspati, Pranjal, and Tandy Warnow. ”Fast and accurate estimation of phylogenetic tree distances.” IEEE/ACM transactions on computational biology and bioinformatics 2.1 (2005): 1-13.

[22] Warnow, Tandy, et al. ”RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees.” Bioinformatics 21.4 (2005): 456-463.

[23] Robinson, D. F., and L. R. Foulds. ”Comparison of phylogenetic trees.” Mathematical biosciences 53.1-2 (1981): 131-147.

[24] Kuhner, Mary K., and Joseph P. Felsenstein. ”A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates.” Molecular biology and evolution 11.3 (1994): 459-468.

[25] Warnow, Tandy, and Bernard ME Moret. ”Maximum quartet consistency and its applications to phylogenetic tree reconstruction.” Algorithms in bioinformatics. Springer, Berlin, Heidelberg, 2001. 1-15.

[26] Felsenstein, Joseph. ”Evolutionary trees from DNA sequences: a maximum likelihood approach.” Journal of molecular evolution 17.6 (1981): 368-376.

[27] Fitch, Walter M. ”Toward defining the course of evolution: minimum change for a specific tree topology.” Systematic biology 20.4 (1971): 406-416.

[28] Bryant, David. ”A classification of consensus methods for phylogenies.” Algorithms in bioinformatics. Springer, Berlin, Heidelberg, 2003. 163-184.

[29] Steel, Mike, and David Penny. ”Distributions of tree comparison metrics—some new results.” Systematic biology 42.2 (1993): 126-141.

[30] Bandelt, Hans-Jürgen, and Andreas Dress. ”A canonical decomposition theory for metrics on a finite set.” Advances in Mathematics 92.1 (1992): 47-105.

[31] Bandelt, Hans-Jürgen, and Andreas Dress. ”A canonical decomposition theory for metrics on a finite set.” Advances in Mathematics 92.1 (1992): 47-105.

[32] Robinson, D. F., and L. R. Foulds. ”Comparison of phylogenetic trees.” Mathematical biosciences 53.1-2 (1981): 131-147.

[33] E. Fix and J. L. Hodges Jr., ”Discriminatory analysis. Nonparametric discrimination: Consistency properties,” in International Statistical Review / Revue Internationale de Statistique, vol. 37, no. 3, pp. 244-258, 1969, doi: 10.2307/1402380.

[34] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18, 9 (September 1975), 509–517. DOI:https://doi.org/10.1145/361002.361007

[35] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18, 9 (September 1975), 509–517. DOI:https://doi.org/10.1145/361002.361007

[36] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the twentieth annual symposium on Computational geometry (SCG ’04). Association for Computing Machinery, New York, NY, USA, 253–262. DOI:https://doi.org/10.1145/997817.997857

[37] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the twentieth annual symposium on Computational geometry (SCG '04). Association for Computing Machinery, New York, NY, USA, 253–262. DOI:https://doi.org/10.1145/997817.997857

[38] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the twentieth annual symposium on Computational geometry (SCG '04). Association for Computing Machinery, New York, NY, USA, 253–262. DOI:https://doi.org/10.1145/997817.997857

[39] Leo Breiman. 2001. Random Forests. Mach. Learn. 45, 1 (October 2001), 5–32. DOI:https://doi.org/10.1023/A:101

[40] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. 1984. Classification and Regression Trees. Wadsworth Brooks/Cole Advanced Books Software, Belmont, CA, USA.

[41] Jerome H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. Ann. Statist. 29, 5 (October 2001), 1189–1232. DOI:https://doi.org/10.1214/aos/1013203451

[42] Yoav Freund and Robert E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J. Comput. Syst. Sci. 55, 1 (August 1997), 119–139. DOI:https://doi.org/10.1006/jcss.1997.1504

[43] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. DOI:https://doi.org/10.1145/2939672.2939785

[44] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 3149–3157.

[45] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 6638–6648.

[46] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. Mach. Learn. 63, 1 (October 2006), 3–42. DOI:https://doi.org/10.1007/s10994-006-6226-1

[47] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM '08). IEEE Computer Society, Washington, DC, USA, 413–422. DOI:https://doi.org/10.1109/ICDM.2008.17

[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998.

[49] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[50] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[51] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.

[52] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. CoRR, abs/1708.07747, 2017.

[53] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[54] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." science 313.5786 (2006): 504-507.

[55] Hinton, Geoffrey E. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

[56] Elman, Jeffrey L. "Finding structure in time." Cognitive science 14.2 (1990): 179-211.

[57] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[58] Salakhutdinov, Ruslan, and Geoffrey Hinton. "Deep boltzmann machines." Artificial intelligence and statistics. 2009.

[59] Kulkarni, Tejas D., et al. "Deep convolutional inverse graphics network." Advances in neural information processing systems. 2015.

[60] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[61] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[62] Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of machine learning research 11.Dec (2010): 3371-3408.

[63] Wang X, Zhai S, Niu Y. Automatic vertebrae localization and identification by combining deep SSAE contextual features and structured regression forest. J Digit Imaging. 2019;32:336–48.

[64] Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. 2008.

[65] Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." Proceedings of the 28th international conference on machine learning (ICML-11). 2011.

[66] Makhzani, Alireza, et al. "Adversarial autoencoders." arXiv preprint arXiv:1511.05644 (2015).

[67] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[68] Chung, Junyoung, et al. "A recurrent latent variable model for sequential data." Advances in neural information processing systems. 2015.

[69] Suzuki, R., Idei, H., Yamashita, Y. and Ogata, T., 2023, November. Hierarchical Variational Recurrent Neural Network Modeling of Sensory Attenuation with Temporal Delay in Action-Outcome. In 2023 IEEE International Conference on Development and Learning (ICDL) (pp. 244-249). IEEE.

[70] Kingma, Diederik P., et al. "Semi-supervised learning with deep generative models." Advances in neural information processing systems. 2014.

[71] Beckmann, N., and H. P. Lenz. "On the distance between two random vectors." Mathematische Annalen 136.3 (1958): 193-198.

[72] Minkowski, Hermann. "Theorie der abzählenden Geometrie." (1910).

[73] Chebyshev, Pafnuty Lvovich. "On the integration of the differential equations of dynamics of a rigid body of revolution." (1897).

[74] Minkowski, Hermann. "Theorie der abzählenden Geometrie." (1910).

[75] Mahalanobis, Prasanta Chandra. "On the generalized distance in statistics." Proceedings of the National Institute of Sciences of India. Vol. 2. No. 1. 1936.

[76] Hamming, Richard W. "Error detecting and error correcting codes." Bell system technical journal 29.2 (1950): 147-160.

[77] Jaccard, Paul. "The distribution of the flora in the alpine zone." New phytologist 11.2 (1912): 37-50.

[78] Salton, Gerard, and Michael J. McGill. "Introduction to modern information retrieval." (1986).

[79] Pearson, Karl. "Notes on regression and inheritance in the case of two parents." Proceedings of the Royal Society of London 58.347-352 (1895): 240-242.

[80] Bray, J. Roger, and John T. Curtis. "An ordination of the upland forest communities of southern Wisconsin." Ecological monographs 27.4 (1957): 325-349.

[81] Lance, G. N., and W. T. Williams. "A general theory of classificatory sorting strategies: 1. Hierarchical systems." The computer journal 9.4 (1967): 373-380.

[82] Haversine, William. "On the distance between two points on the earth." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 34.226 (1849): 145-154.

[83] Stone, Mervyn. "Cross-validatory choice and assessment of statistical predictions." Journal of the Royal Statistical Society: Series B (Methodological) 36.2 (1974): 111-147.

[84] Willmott, Cort J. "On the validation of models." Physical Geography 2.2 (1981): 184-194.

[85] Armstrong Jr, J. Scott, and Fred Collopy. "Error measures for generalizing about forecasting methods: Empirical comparisons." International journal of forecasting 8.1 (1992): 69-80.

[86] DeGroot, Morris H. (1970). Optimal Statistical Decisions. McGraw-Hill. ISBN 978-0-07-016192-7.

[87] Makridakis, Spyros (1993) "Accuracy measures: theoretical and practical concerns" International Journal of Forecasting, 9(4), pp. 527–529.

[88] Stone, Mervyn. "Cross-validatory choice and assessment of statistical predictions." Journal of the Royal Statistical Society: Series B (Methodological) 36.2 (1974): 111-147.

[89] Willmott, Cort J. "On the validation of models." Physical Geography 2.2 (1981): 184-194.

[90] K. S. Fu, "Syntactic Pattern Recognition and Applications," in Proceedings of the IEEE, vol. 65, no. 11, pp. 1665-1683, Nov. 1977, doi: 10.1109/PROC.1977.10739.

[91] DeGroot, Morris H. (1970). Optimal Statistical Decisions. McGraw-Hill. ISBN 978-0-07-016192-7.

[92] Rappaport, Theodore S. (2002). Wireless Communications: Principles and Practice (2nd ed.). Prentice Hall. ISBN 978-0-13-042232-3.

[93] Vincenty, T. (1975). "Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations". Survey Review. 23 (176): 88–93. doi:10.1179/sre.1975.23.176.88. S2CID 128674195.

[94] Bradley, Andrew P. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." Pattern recognition 30.7 (1997): 1145-1159.

[95] Swenson, Michael S., and Tandy Warnow. "Supertree methods for ancestral species reconstruction." BMC bioinformatics 12.1 (2011): S10.

[96] Molloy, Erin K., and Tandy Warnow. "TreeMerge: A new method for improving the scalability of species tree estimation methods." Bioinformatics (2019).

[97] Huson, Daniel H. "SplitsTree: analyzing and visualizing evolutionary data." Bioinformatics 14.1 (1998): 68-73.

[98] Huson, Daniel H., Nettles, S. M., Warnow, T., 1999. Disk-Covering, a fast-converging method for phylogenetic tree reconstruction. J. Comput. Biol. 6, 369–386.

[99] Rappaport, Theodore S. (2002). Wireless Communications: Principles and Practice (2nd ed.). Prentice Hall. ISBN 978-0-13-042232-3.

[100] Lin, J. (1991). "Divergence measures based on the Shannon entropy". IEEE Transactions on Information Theory. 37 (1): 145–151. doi:10.1109/18.61115. S2CID 2069861.

[101] Kullback, Solomon; Leibler, Richard A. (1951). "On Information and Sufficiency". The Annals of Mathematical Statistics. 22 (1): 79–86. doi:10.1214/aoms/1177729694. ISSN 0003-4851. JSTOR 2236703. MR 0039968.

[102] Hellinger, E. (1909). "Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen". Journal für die reine und angewandte Mathematik. 136: 210–271. doi:10.1515/crll.1909.136.210. S2CID 121978905.

[103] Bhattacharyya, A. (1943). "On a measure of divergence between two statistical populations defined by their probability distributions". Bulletin of the Calcutta Mathematical Society. 35: 99–109.

[104] Gower, J. C. (1971). "A general coefficient of similarity and some of its properties". Biometrics. 27 (4): 857–871. doi:10.2307/2528823. JSTOR 2528823. PMID 5119868. S2CID 2069861.

[105] Matusita, K. (1955). "Decision rules, based on the distance covariance, for the classification problem". Annals of the Institute of Statistical Mathematics. 7 (1): 1–16. doi:10.1007/BF02868559. S2CID 121978905.

[106] Beckmann, N., and H. P. Lenz. "On the distance between two random vectors." Mathematische Annalen 136.3 (1958): 193-198.

[107] Gower, J. C. (1971). "A general coefficient of similarity and some of its properties". Biometrics. 27 (4): 857–871. doi:10.2307/2528823. JSTOR 2528823. PMID 5119868. S2CID 2069861.

[108] Gower, J. C. (1971). "A general coefficient of similarity and some of its properties". Biometrics. 27 (4): 857–871. doi:10.2307/2528823. JSTOR 2528823. PMID 5119868. S2CID 2069861.

[109] Pearson, Karl. "Notes on regression and inheritance in the case of two parents." Proceedings of the Royal Society of London 58.347-352 (1895): 240-242.

[110] Gower, J. C. (1971). "A general coefficient of similarity and some of its properties". Biometrics. 27 (4): 857–871. doi:10.2307/2528823. JSTOR 2528823. PMID 5119868. S2CID 2069861.

[111] Kullback, Solomon; Leibler, Richard A. (1951). "On Information and Sufficiency". The Annals of Mathematical Statistics. 22 (1): 79–86. doi:10.1214/aoms/1177729694. ISSN 0003-4851. JSTOR 2236703. MR 0039968.

[112] Hellinger, E. (1909). "Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen". Journal für die reine und angewandte Mathematik. 136: 210–271. doi:10.1515/crll.1909.136.210. S2CID 121978905.

[113] Lin, J. (1991). "Divergence measures based on the Shannon entropy". IEEE Transactions on Information Theory. 37 (1): 145–151. doi:10.1109/18.61115. S2CID 2069861.

[114] Troger, Hans, and Alois Steindl. Nonlinear stability and bifurcation theory: an introduction for engineers and applied scientists. Springer Science & Business Media, 2012. Appendix H. Stability boundaries in parameter space, pp. 340-343.