Learning with invariants

Master's Thesis

Author: Vladislav Nikolov Vasilev **Supervisor:** Dr. Oriol Pujol Vila

July 13, 2022

Department of Mathematics and Computer Science
University of Barcelona

Table of contents

Introduction

Motivation

- Classical machine learning methods consider no further information apart from the training samples.

Motivation

- Classical machine learning methods consider no further information apart from the training samples.
- But the data may have useful statistical information...

Motivation

- Classical machine learning methods consider no further information apart from the training samples.
- But the data may have useful statistical information...
- Enter the Learning Using Statistical Invariants (LUSI) paradigm!

Motivation

- Classical machine learning methods consider no further information apart from the training samples.
- But the data may have useful statistical information...
- Enter the Learning Using Statistical Invariants (LUSI) paradigm!
- Goal: Exploit statistical information of the training data in order to find better approximations of the goal function.

Goals of the project

1. Understand and further explore the application of the invariants to the learning problem.

Goals of the project

1. Understand and further explore the application of the invariants to the learning problem.
2. Propose new general use invariants that require no prior knowledge of the problem.

Goals of the project

1. Understand and further explore the application of the invariants to the learning problem.
2. Propose new general use invariants that require no prior knowledge of the problem.
3. Automatize the selection process of the most suitable invariants for a given problem.

Goals of the project

1. Understand and further explore the application of the invariants to the learning problem.
2. Propose new general use invariants that require no prior knowledge of the problem.
3. Automatize the selection process of the most suitable invariants for a given problem.
4. Extend the learning paradigm to multiclass classification problems.

Learning using statistical invariants

Strong and weak modes of convergence (I)

In a Hilbert space, the relationships between functions $f_1(x)$ and $f_2(x)$ have two numerical properties:

Strong and weak modes of convergence (I)

In a Hilbert space, the relationships between functions $f_1(x)$ and $f_2(x)$ have two numerical properties:

1. The distance between functions

$$\rho(f_1, f_2) = \|f_1(x) - f_2(x)\|$$

   which is defined by the metric of the L_2 space and

Strong and weak modes of convergence (I)

In a Hilbert space, the relationships between functions $f_1(x)$ and $f_2(x)$ have two numerical properties:

1. The distance between functions

$$\rho(f_1, f_2) = \|f_1(x) - f_2(x)\|$$

   which is defined by the metric of the L_2 space and

2. The inner product between functions

$$R(f_1, f_2) = \langle f_1(x), f_2(x) \rangle$$

   which has to satisfy the corresponding requirements.

Strong and weak modes of convergence (II)

These two properties imply two different modes of convergence:

Strong and weak modes of convergence (II)

These two properties imply two different modes of convergence:

1. Strong convergence (convergence in metrics):

$$\lim_{l \to \infty} \| P_l(y = 1|x) - P(y = 1|x) \| = 0 \quad \forall x$$

Strong and weak modes of convergence (II)

These two properties imply two different modes of convergence:

1. Strong convergence (convergence in metrics):

$$\lim_{l \to \infty} \| P_l(y = 1|x) - P(y = 1|x) \| = 0 \quad \forall x$$

2. Weak convergence (convergence in inner products):

$$\lim_{l \to \infty} \langle P_l(y = 1|x) - P(y = 1|x), \psi(x) \rangle = 0 \quad \forall \psi(x) \in L_2$$

The LUSI paradigm (I)

- Based on the weak mode of convergence.

The LUSI paradigm (I)

- Based on the weak mode of convergence.
- Replaces the infinite set of functions with a set of functions $\mathcal{P} = \{\psi_1(x), \ldots, \psi_m(x)\}$ called predicates.

The LUSI paradigm (I)

- Based on the weak mode of convergence.
- Replaces the infinite set of functions with a set of functions $\mathcal{P} = \{\psi_1(x), \dots, \psi_m(x)\}$ called predicates.
- The predicates describe important properties of the desired conditional probability function $P(y = 1|x)$.

The LUSI paradigm (I)

- Based on the weak mode of convergence.
- Replaces the infinite set of functions with a set of functions $\mathcal{P} = \{\psi_1(x), \ldots, \psi_m(x)\}$ called predicates.
- The predicates describe important properties of the desired conditional probability function $P(y = 1|x)$.
- These important properties are called invariants.

The LUSI paradigm (II)

The main goal of this paradigm is to find an approximation of the conditional probability function that preserves the specified invariants. Mathematically, this can be expressed as:

$$\frac{1}{l}\sum_{i=1}^{l}\psi_s(x_i)P_l(y=1|x_i) \approx \frac{1}{l}\sum_{i=1}^{l}y_i\psi_s(x_i), \quad s=1,\ldots,m$$

The LUSI paradigm (II)

The main goal of this paradigm is to find an approximation of the conditional probability function that preserves the specified invariants. Mathematically, this can be expressed as:

predicates

$$\frac{1}{l}\sum_{i=1}^{l} \psi_s(x_i)\, P_l(y=1|x_i) \approx \frac{1}{l}\sum_{i=1}^{l} y_i\, \psi_s(x_i), \quad s = 1, \ldots, m$$

The LUSI paradigm (II)

The main goal of this paradigm is to find an approximation of the conditional probability function that preserves the specified invariants. Mathematically, this can be expressed as:

$$\frac{1}{l}\sum_{i=1}^{l} \overbrace{\psi_s(x_i)}^{\text{predicates}} \underbrace{P_l(y=1|x_i)}_{\text{prediction}} \approx \frac{1}{l}\sum_{i=1}^{l} y_i \, \overbrace{\psi_s(x_i)}^{\text{predicates}}, \quad s = 1, \dots, m$$

The LUSI paradigm (II)

The main goal of this paradigm is to find an approximation of the conditional probability function that preserves the specified invariants. Mathematically, this can be expressed as:

$$\frac{1}{l}\sum_{i=1}^{l} \underbrace{\psi_s(x_i)}_{\text{predicates}} \underbrace{P_l(y=1|x_i)}_{\text{prediction}} \approx \frac{1}{l}\sum_{i=1}^{l} \underbrace{y_i}_{\text{ground truth label}} \underbrace{\psi_s(x_i)}_{\text{predicates}}, \quad s = 1, \ldots, m$$

Invariant selection

The authors propose a method to select new invariants. Given a predicate ψ_{m+1}, we first evaluate the following expression:

$$\mathcal{T} = \frac{\left|\sum_{i=1}^{l} \psi_{m+1}(x_i) P_l^m(y=1|x_i) - \sum_{i=1}^{l} y_i \psi_{m+1}(x_i)\right|}{\sum_{i=1}^{l} y_i \psi_{m+1}(x_i)}$$

Invariant selection

The authors propose a method to select new invariants. Given a predicate ψ_{m+1}, we first evaluate the following expression:

approximation of cond. prob. func.
using m invariants

$$\mathcal{T} = \frac{\left|\sum_{i=1}^{l} \psi_{m+1}(x_i)\, P_l^m(y=1|x_i) - \sum_{i=1}^{l} y_i \psi_{m+1}(x_i)\right|}{\sum_{i=1}^{l} y_i \psi_{m+1}(x_i)}$$

Invariant selection

The authors propose a method to select new invariants. Given a predicate ψ_{m+1}, we first evaluate the following expression:

approximation of cond. prob. func.
using m invariants

$$\mathcal{T} = \frac{\left|\sum_{i=1}^{l} \psi_{m+1}(x_i)\, P_l^m(y = 1|x_i) - \sum_{i=1}^{l} y_i \psi_{m+1}(x_i)\right|}{\sum_{i=1}^{l} y_i \psi_{m+1}(x_i)}$$

If $\mathcal{T} \geq \delta$ for some small threshold δ, then the new invariant defined by predicate ψ_{m+1} is selected. Otherwise, it is disregarded.

Statistical invariants

- A statistical invariant is a specific realization of a predicate with statistical meaning.

Statistical invariants

- A statistical invariant is a specific realization of a predicate with statistical meaning.
- It captures some sort of statistical information of the data that has to be conserved when selecting the best candidate function.

Statistical invariants

- A statistical invariant is a specific realization of a predicate with statistical meaning.
- It captures some sort of statistical information of the data that has to be conserved when selecting the best candidate function.
- There are different types of statistical invariants:

Statistical invariants

- A statistical invariant is a specific realization of a predicate with statistical meaning.
- It captures some sort of statistical information of the data that has to be conserved when selecting the best candidate function.
- There are different types of statistical invariants:
  - Zeroth order invariant (proportion of positive class elements):

$$\psi_{z.o.}(x) = 1$$

Statistical invariants

- A statistical invariant is a specific realization of a predicate with statistical meaning.
- It captures some sort of statistical information of the data that has to be conserved when selecting the best candidate function.
- There are different types of statistical invariants:
  - Zeroth order invariant (proportion of positive class elements):

$$\psi_{z.o.}(x) = 1$$

  - First order invariant (mean or centroid of positive class elements):

$$\psi_{f.o.}(x) = x$$

Solving the learning problem (I)

In a Reproducing Kernel Hilbert Space (RKHS), the estimate of the conditional probability function can be computed as

$$f(x) = A^T \mathcal{K}(x) + c$$

Solving the learning problem (I)

In a Reproducing Kernel Hilbert Space (RKHS), the estimate of the conditional probability function can be computed as

vector of coefficients

$$f(x) = \boxed{A^T} \, \mathcal{K}(x) + c$$

Solving the learning problem (I)

In a Reproducing Kernel Hilbert Space (RKHS), the estimate of the conditional probability function can be computed as

vector of coefficients

$$f(x) = A^T \mathcal{K}(x) + c$$

vector of kernel functions
evaluated on training data

Solving the learning problem (I)

In a Reproducing Kernel Hilbert Space (RKHS), the estimate of the conditional probability function can be computed as

$$f(x) = \underbrace{\overbrace{A^T}^{\text{vector of coefficients}} \mathcal{K}(x)}_{\mathcal{K}(x):\ \text{vector of kernel functions evaluated on training data}} + \overbrace{c}^{\text{bias}}$$

Solving the learning problem (II)

Let us define the following elements:

Solving the learning problem (II)

Let us define the following elements:

- $Y = (y_1, \ldots, y_l)$ the vector containing the labels of the training set.

Solving the learning problem (II)

Let us define the following elements:

- $Y = (y_1, \ldots, y_l)$ the vector containing the labels of the training set.
- $K \in \mathbb{R}^{l \times l}$ the matrix with elements $K(x_i, x_j),\ i, j = 1, \ldots, l$.

Solving the learning problem (II)

Let us define the following elements:

- $Y = (y_1, \dots, y_l)$ the vector containing the labels of the training set.
- $K \in \mathbb{R}^{l \times l}$ the matrix with elements $K(x_i, x_j),\ i, j = 1, \dots, l$.
- $\Phi_s = (\psi_s(x_1), \dots, \psi_s(x_l))^T$ the vector obtained from evaluating the l points of the sample using predicate ψ_s.

Solving the learning problem (II)

Let us define the following elements:

- $Y = (y_1, \ldots, y_l)$ the vector containing the labels of the training set.
- $K \in \mathbb{R}^{l \times l}$ the matrix with elements $K(x_i, x_j),\ i, j = 1, \ldots, l$.
- $\Phi_s = (\psi_s(x_1), \ldots, \psi_s(x_l))^T$ the vector obtained from evaluating the l points of the sample using predicate ψ_s.
- $1_l = (1, \ldots, 1) \in \mathbb{R}^l$ a vector of ones.

Solving the learning problem (II)

Let us define the following elements:

- $Y = (y_1, \ldots, y_l)$ the vector containing the labels of the training set.
- $K \in \mathbb{R}^{l \times l}$ the matrix with elements $K(x_i, x_j),\ i, j = 1, \ldots, l$.
- $\Phi_s = (\psi_s(x_1), \ldots, \psi_s(x_l))^T$ the vector obtained from evaluating the l points of the sample using predicate ψ_s.
- $1_l = (1, \ldots, 1) \in \mathbb{R}^l$ a vector of ones.
- $V \in \mathbb{R}^{l \times l}$ a matrix called the V-matrix, which captures some geometric properties of the data.

Solving the learning problem (II)

Let us define the following elements:

- $Y = (y_1, \ldots, y_l)$ the vector containing the labels of the training set.
- $K \in \mathbb{R}^{l \times l}$ the matrix with elements $K(x_i, x_j),\ i, j = 1, \ldots, l$.
- $\Phi_s = (\psi_s(x_1), \ldots, \psi_s(x_l))^T$ the vector obtained from evaluating the l points of the sample using predicate ψ_s.
- $1_l = (1, \ldots, 1) \in \mathbb{R}^l$ a vector of ones.
- $V \in \mathbb{R}^{l \times l}$ a matrix called the V-matrix, which captures some geometric properties of the data.

Note

In the most simple case, the $V-$matrix can be replaced with the identity matrix.

Solving the learning problem (III)

We can formulate and solve a minimization problem subject to the invariants equality constraints that has a closed-form solution.

The coefficients vector A can be computed as

$$A = (A_V - cA_c) - \left(\sum_{s=1}^{m} \mu_s A_s \right)$$

Solving the learning problem (III)

We can formulate and solve a minimization problem subject to the invariants equality constraints that has a closed-form solution.

The coefficients vector A can be computed as

$$A = (A_V - cA_c) - \left(\sum_{s=1}^{m} \mu_s A_s \right)$$

where

$$A_V = (VK + \gamma I)^{-1} VY$$
$$A_c = (VK + \gamma I)^{-1} V 1_l$$
$$A_s = (VK + \gamma I)^{-1} \Phi_s, \quad s = 1, \ldots, n$$

Solving the learning problem (IV)

The values of c and the m coefficients μ_s can be obtained solving the following system of equations:

$$c[1_l^T VKA_c - 1_l^T V1_l] + \sum_{s=1}^{m} \mu_s [1_l^T VKA_s - 1_l^T \Phi_s] = [1_l^T VKA_V - 1_l^T VY]$$

$$c[A_c^T K\Phi_k - 1_l^T \Phi_k] + \sum_{s=1}^{m} \mu_s A_s^T K\Phi_k = [A_V^T K\Phi_k - Y^T \Phi_k], \quad k = 1, \dots, m$$

Overview of the LUSI algorithm

Step 1: Construct an estimate of the conditional probability function without considering the predicates.

Step 2: Find the maximal disagreement value $\mathcal{T}_s$ for vectors

$$\Phi_k = (\psi_k(x_1), \ldots, \psi_k(x_l))^T, \quad k = 1, \ldots, m$$

Step 3: If $\mathcal{T}_s > \delta$, add the invariant associated to the predicate ψ_s; otherwise stop.

Step 4: Find a new approximation of the conditional probability function and go back to **Step 2**; otherwise stop.

Main results and limitations

Contributions

Main results and limitations

Contributions

- Good overall results.

Main results and limitations

Contributions

- Good overall results.
- It can reduce the number of necessary training examples in order to get a good approximation of the conditional probability function.

Main results and limitations

Contributions

- Good overall results.
- It can reduce the number of necessary training examples in order to get a good approximation of the conditional probability function.

Limitations

Main results and limitations

Contributions

- Good overall results.
- It can reduce the number of necessary training examples in order to get a good approximation of the conditional probability function.

Limitations

- Invariants are problem specific.

Main results and limitations

Contributions

- Good overall results.
- It can reduce the number of necessary training examples in order to get a good approximation of the conditional probability function.

Limitations

- Invariants are problem specific.
- Invariant selection can sometimes be a "black-art".

Main results and limitations

Contributions

- Good overall results.
- It can reduce the number of necessary training examples in order to get a good approximation of the conditional probability function.

Limitations

- Invariants are problem specific.
- Invariant selection can sometimes be a "black-art".
- Invariants only consider statistical information of the positive class, hence it cannot be directly applied to multiclass classification problems.

Proposals

Overview

In order to deal with some of the limitations of the original work, we propose

Overview

In order to deal with some of the limitations of the original work, we propose

- Two new general use invariants based on random processes: random projections and random hyperplanes.

Overview

In order to deal with some of the limitations of the original work, we propose

- Two new general use invariants based on random processes: random projections and random hyperplanes.
- An extension of the original algorithm to consider the negative class and all classes in multiclass classification problems using Error Correcting Output Codes (ECOC).

Random projections (I)

- Based on the first order invariant.

Random projections (I)

- Based on the first order invariant.
- Projects the data into a new dimensional space, as if it was viewed from a particular point in the original space.

Random projections (I)

- Based on the first order invariant.
- Projects the data into a new dimensional space, as if it was viewed from a particular point in the original space.
- Preserve the centroid of the positive class in this new space.

Random projections (I)

- Based on the first order invariant.
- Projects the data into a new dimensional space, as if it was viewed from a particular point in the original space.
- Preserve the centroid of the positive class in this new space.
- Mathematically, this can be expressed as:

$$\psi_{r.p.}(x) = xp$$

Random projections (I)

- Based on the first order invariant.
- Projects the data into a new dimensional space, as if it was viewed from a particular point in the original space.
- Preserve the centroid of the positive class in this new space.
- Mathematically, this can be expressed as:

$p \sim \mathcal{N}(\mu, \Sigma)$, projection vector

$$\psi_{r.p.}(x) = x\ p$$

Random projections (II)

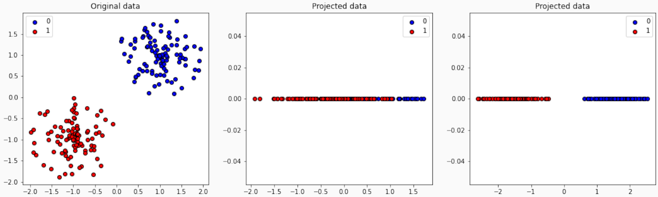


Figure 1: Examples of the random projection invariant.

Random hyperplanes (I)

- Based on the zeroth order invariant.

Random hyperplanes (I)

- Based on the zeroth order invariant.
- Split the data using a hyperplane passing through one of the points of the sample.

Random hyperplanes (I)

- Based on the zeroth order invariant.
- Split the data using a hyperplane passing through one of the points of the sample.
- The elements that fall on the right side of the hyperplane will be considered as the positive samples, whereas the ones on the left as the negative ones.

Random hyperplanes (I)

- Based on the zeroth order invariant.
- Split the data using a hyperplane passing through one of the points of the sample.
- The elements that fall on the right side of the hyperplane will be considered as the positive samples, whereas the ones on the left as the negative ones.
- Preserve the proportion of elements of the positive class that fall on the right side of the hyperplane.

Random hyperplanes (I)

- Based on the zeroth order invariant.
- Split the data using a hyperplane passing through one of the points of the sample.
- The elements that fall on the right side of the hyperplane will be considered as the positive samples, whereas the ones on the left as the negative ones.
- Preserve the proportion of elements of the positive class that fall on the right side of the hyperplane.
- Mathematically, this can be expressed as:

$$\psi_{r.h.}(x) = \begin{cases} 1 & \text{if } (x - x_c)n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Random hyperplanes (I)

- Based on the zeroth order invariant.
- Split the data using a hyperplane passing through one of the points of the sample.
- The elements that fall on the right side of the hyperplane will be considered as the positive samples, whereas the ones on the left as the negative ones.
- Preserve the proportion of elements of the positive class that fall on the right side of the hyperplane.
- Mathematically, this can be expressed as:

$$\psi_{r.h.}(x) = \begin{cases} 1 & \text{if } (x - \overbrace{\boxed{x_c}}^{x_c \in X})n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Random hyperplanes (I)

- Based on the zeroth order invariant.
- Split the data using a hyperplane passing through one of the points of the sample.
- The elements that fall on the right side of the hyperplane will be considered as the positive samples, whereas the ones on the left as the negative ones.
- Preserve the proportion of elements of the positive class that fall on the right side of the hyperplane.
- Mathematically, this can be expressed as:

$$\psi_{r.h.}(x) = \begin{cases} 1 & \text{if } (x - \underbrace{x_c}_{x_c \in X})\underbrace{n}_{n \sim \mathcal{N}(\mu, \Sigma),\ \text{normal vector}} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Random hyperplanes (II)

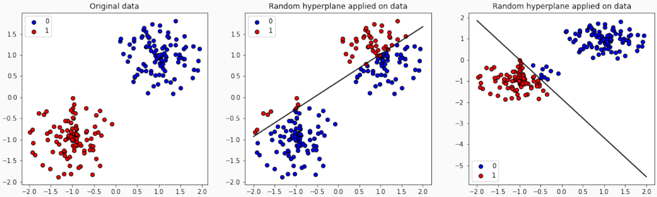


Figure 2: Examples of the random hyperplane invariant.

Considerations

The original LUSI algorithm has to be modified to take into account the stochastic nature of these new invariants. Specifically, the following changes have to be introduced:

Considerations

The original LUSI algorithm has to be modified to take into account the stochastic nature of these new invariants. Specifically, the following changes have to be introduced:

1. Generate new invariants at each iteration.

Considerations

The original LUSI algorithm has to be modified to take into account the stochastic nature of these new invariants. Specifically, the following changes have to be introduced:

1. Generate new invariants at each iteration.
2. Allow the algorithm to try new invariants when no invariant is selected.

Considerations

The original LUSI algorithm has to be modified to take into account the stochastic nature of these new invariants. Specifically, the following changes have to be introduced:

1. Generate new invariants at each iteration.
2. Allow the algorithm to try new invariants when no invariant is selected.
3. Limit the number of tries so that the algorithm does not end up in an infinite loop.

Extending LUSI with ECOC (I)

- In order to deal with both the negative class and multiclass classification problems, we can extend LUSI using the Error Correcting Output Codes (ECOC) framework.

Extending LUSI with ECOC (I)

- In order to deal with both the negative class and multiclass classification problems, we can extend LUSI using the Error Correcting Output Codes (ECOC) framework.
- The ECOC framework is used to transform multiclass classification problems into binary ones.

Extending LUSI with ECOC (I)

- In order to deal with both the negative class and multiclass classification problems, we can extend LUSI using the Error Correcting Output Codes (ECOC) framework.
- The ECOC framework is used to transform multiclass classification problems into binary ones.
- For each one of the N_c classes we create a codeword of length n. These codewords are organized as the rows of a matrix $M \in \{0, 1\}^{N_c \times n}$ called the coding matrix.

Extending LUSI with ECOC (I)

- In order to deal with both the negative class and multiclass classification problems, we can extend LUSI using the Error Correcting Output Codes (ECOC) framework.
- The ECOC framework is used to transform multiclass classification problems into binary ones.
- For each one of the N_c classes we create a codeword of length n. These codewords are organized as the rows of a matrix $M \in \{0,1\}^{N_c \times n}$ called the coding matrix.
- Each column of the matrix is treated as an individual classification problem and a binary classifier is fit with the transformed data.

Extending LUSI with ECOC (II)

	h_1	h_2	h_3	h_4	h_5
C1	1	0	0	0	0
C2	0	1	0	0	0
C3	0	0	1	0	0
C4	0	0	0	1	0
C5	0	0	0	0	1

	h_1	h_2	h_3	h_4	h_5
C1	1	1	1	0	0
C2	0	1	1	0	1
C3	1	0	0	1	0
C4	0	1	0	1	0
C5	0	0	1	0	1

Table 1: Two examples of coding matrices. The left one represents a special coding called one-against-all.

Extending LUSI with ECOC (III)

Let $f(x) = (f_1(x), \dots, f_n(x))$ be the vector containing the predictions for each one of the problems. In order to obtain the final output class, we have to decode this vector, which can be done using the following expression:

$$\hat{y} = \operatorname*{argmin}_r d(M_r, f(x))$$

Extending LUSI with ECOC (III)

Let $f(x) = (f_1(x), \ldots, f_n(x))$ be the vector containing the predictions for each one of the problems. In order to obtain the final output class, we have to decode this vector, which can be done using the following expression:

distance metric (Hamming, Euclidean, etc.)

$$\hat{y} = \underset{r}{\operatorname{argmin}}\ \boxed{d}\,(M_r, f(x))$$

Extending LUSI with ECOC (III)

Let $f(x) = (f_1(x), \ldots, f_n(x))$ be the vector containing the predictions for each one of the problems. In order to obtain the final output class, we have to decode this vector, which can be done using the following expression:

distance metric (Hamming, Euclidean, etc.)

$$\hat{y} = \underset{r}{\operatorname{argmin}}\ d\left(M_r, f(x)\right)$$

$r-$th row of M

Extending LUSI with ECOC (III)

Let $f(x) = (f_1(x), \ldots, f_n(x))$ be the vector containing the predictions for each one of the problems. In order to obtain the final output class, we have to decode this vector, which can be done using the following expression:

distance metric (Hamming, Euclidean, etc.)

$$\hat{y} = \underset{r}{\operatorname{argmin}}\ d(M_r, f(x))$$

$r-$th row of M

In our proposal we use a vector of predicted probabilities and the Euclidean distance.

Experimentation and results

Overview

Experimentation with two types of datasets with different goals in mind:

Overview

Experimentation with two types of datasets with different goals in mind:

- Toy datasets

Overview

Experimentation with two types of datasets with different goals in mind:

- Toy datasets
    - Compare the different invariants.

Overview

Experimentation with two types of datasets with different goals in mind:

- Toy datasets
  - Compare the different invariants.
  - Compare the original algorithm with the ECOC version.

Overview

Experimentation with two types of datasets with different goals in mind:

- Toy datasets
    - Compare the different invariants.
    - Compare the original algorithm with the ECOC version.
    - Find interesting properties of the invariants.

Overview

Experimentation with two types of datasets with different goals in mind:

- Toy datasets
    - Compare the different invariants.
    - Compare the original algorithm with the ECOC version.
    - Find interesting properties of the invariants.
- Real datasets

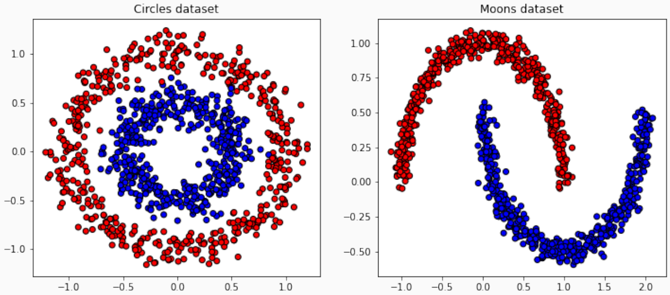
Overview

Experimentation with two types of datasets with different goals in mind:

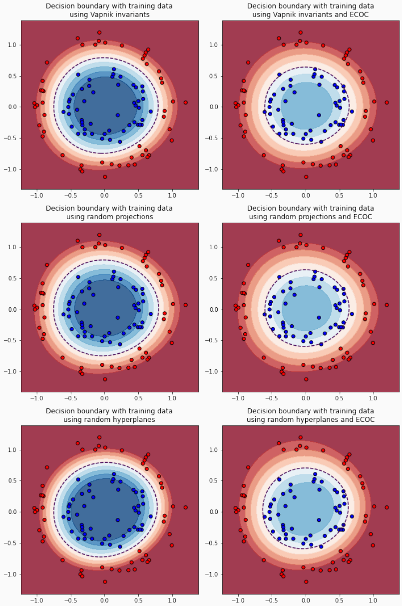
- Toy datasets
    - Compare the different invariants.
    - Compare the original algorithm with the ECOC version.
    - Find interesting properties of the invariants.
- Real datasets
    - Assess the quality of the proposed invariants.

Experimentation with toy datasets

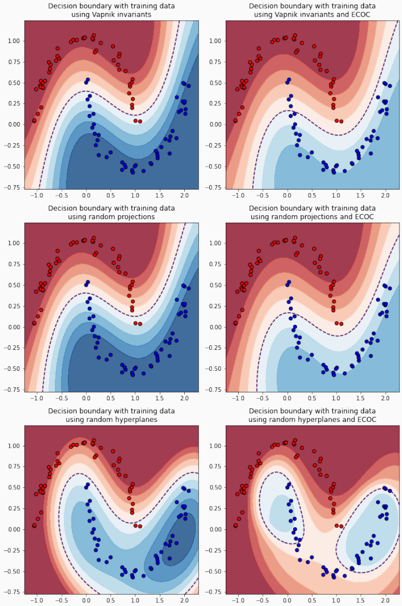
Compare Vapnik's invariants (zeroth and first order) with random projections and random hyperplanes on the Circles and Moons datasets.



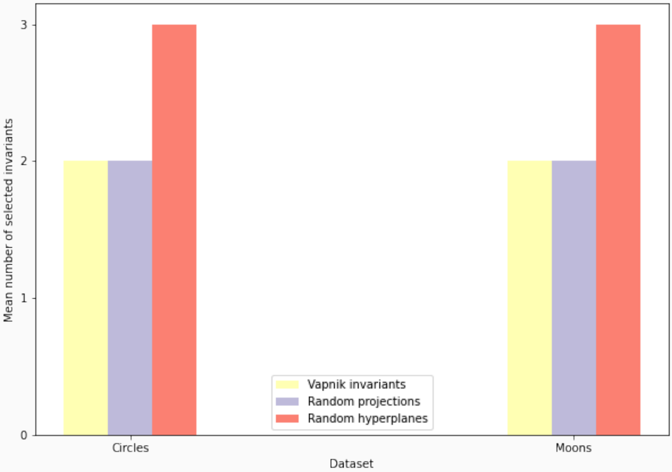
Comparing LUSI with the ECOC version (I)



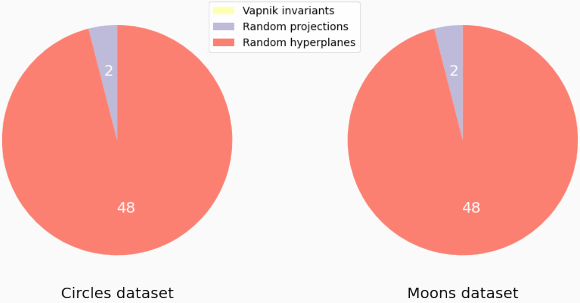
Comparing LUSI with the ECOC version (II)



Comparing the number of selected invariants



Exploring the bias towards certain types of invariants



Experimentation with real datasets

- Assess the quality of the invariants on 5 multiclass classification problems.

Experimentation with real datasets

- Assess the quality of the invariants on 5 multiclass classification problems.
- Compare 4 models using different types of invariants: no invariants, Vapnik's invariants, random projections and random hyperplanes.

Experimentation with real datasets

- Assess the quality of the invariants on 5 multiclass classification problems.
- Compare 4 models using different types of invariants: no invariants, Vapnik's invariants, random projections and random hyperplanes.
- Evaluate the models using different subsamples of the training data: 100%, 50% and 10% of the training partition.

Selected datasets

Problem	Num. examples	Attributes	Classes
Balance Scale	625	4	3
Ecoli	336	8	8
Glass	214	9	6
Iris	150	4	3
Yeast	1484	8	10

Results

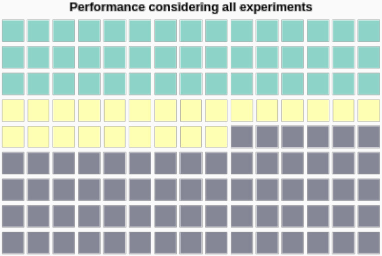
Dataset	Baseline model	Vapnik invariants	Random projections	Random hyperplanes
Balance Scale	$91.47 \pm 0.38\%$	$91.47 \pm 0.38\%$	$\mathbf{91.73 \pm 0.38}\%$	$90.13 \pm 1.51\%$
Ecoli	$85.29 \pm 2.08\%$	$\mathbf{85.29 \pm 1.20}\%$	$84.15 \pm 2.05\%$	$75.98 \pm 6.28\%$
Glass	$\mathbf{72.87 \pm 7.91}\%$	$\mathbf{72.87 \pm 7.91}\%$	$72.09 \pm 7.44\%$	$71.06 \pm 7.60\%$
Iris	$\mathbf{96.67 \pm 0.00}\%$	$95.56 \pm 1.57\%$	$95.19 \pm 1.66\%$	$88.52 \pm 11.56\%$
Yeast	$53.76 \pm 1.04\%$	$\mathbf{53.87 \pm 1.20}\%$	$52.53 \pm 5.44\%$	$50.39 \pm 5.33\%$

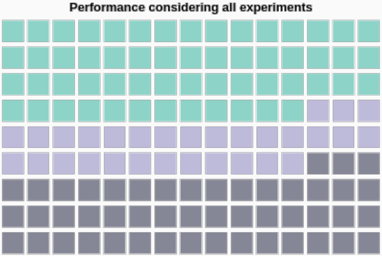
Table 2: Results using 100% of the training data.

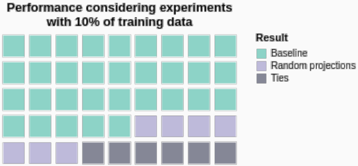
Dataset	Baseline model	Vapnik invariants	Random projections	Random hyperplanes
Balance Scale	$\mathbf{88.80 \pm 1.73}\%$	$87.73 \pm 2.10\%$	$83.73 \pm 7.17\%$	$76.00 \pm 12.77\%$
Ecoli	$71.57 \pm 1.39\%$	$\mathbf{72.55 \pm 4.22}\%$	$67.32 \pm 4.03\%$	$63.89 \pm 9.61\%$
Glass	$\mathbf{56.59 \pm 4.78}\%$	$45.74 \pm 6.67\%$	$52.45 \pm 7.03\%$	$45.48 \pm 9.50\%$
Iris	$\mathbf{93.33 \pm 2.72}\%$	$90.00 \pm 4.71\%$	$77.78 \pm 22.93\%$	$84.81 \pm 9.04\%$
Yeast	$\mathbf{47.92 \pm 2.08}\%$	$47.92 \pm 2.14\%$	$45.68 \pm 4.60\%$	$37.82 \pm 7.92\%$

Table 3: Results using a subsample of 10% of the training data.

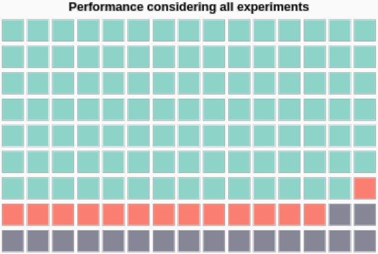
Comparing the invariants on individual problems (I)

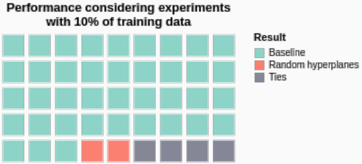


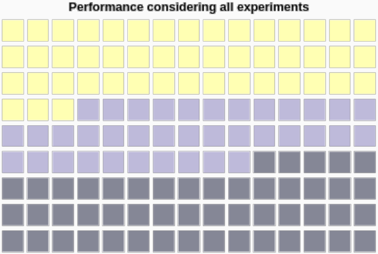




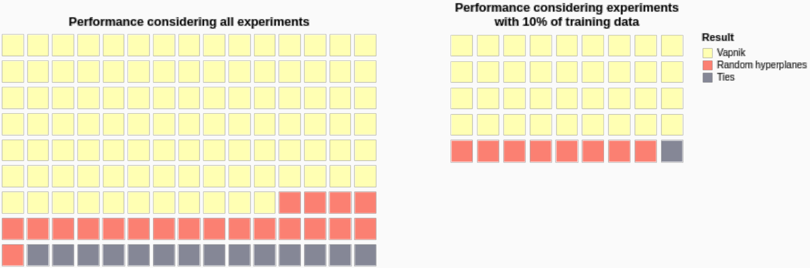
Comparing the invariants on individual problems (II)

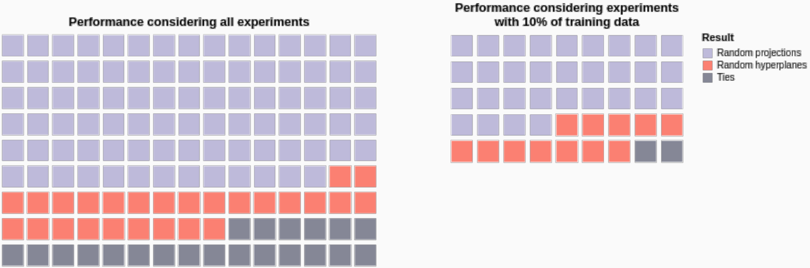






Comparing the invariants on individual problems (III)





Conclusions

Conclusions

✓ Propose new general use invariants that require no prior knowledge of the problem.

Conclusions

✓ Propose new general use invariants that require no prior knowledge of the problem.
  - Random projections are slightly worse than Vapnik's invariants.

Conclusions

✓ Propose new general use invariants that require no prior knowledge of the problem.

- Random projections are slightly worse than Vapnik's invariants.
- Random hyperplanes are outperformed by all the other types of invariants.

Conclusions

✓ Propose new general use invariants that require no prior knowledge of the problem.

- Random projections are slightly worse than Vapnik's invariants.
- Random hyperplanes are outperformed by all the other types of invariants.

✗ Automatized the selection process of the most suitable invariants for a given problem.

Conclusions

✓ Propose new general use invariants that require no prior knowledge of the problem.
  - Random projections are slightly worse than Vapnik's invariants.
  - Random hyperplanes are outperformed by all the other types of invariants.

✗ Automatized the selection process of the most suitable invariants for a given problem.

✓ Extend the learning paradigm to multiclass classification problems.

Conclusions

- ✓ Propose new general use invariants that require no prior knowledge of the problem.
  - Random projections are slightly worse than Vapnik's invariants.
  - Random hyperplanes are outperformed by all the other types of invariants.
- ✗ Automatized the selection process of the most suitable invariants for a given problem.
- ✓ Extend the learning paradigm to multiclass classification problems.
  - Created a small software module containing the different invariants and both the original and ECOC version of LUSI.

Future work

- Reformulate the optimization problem so that it can be solved by using an iterative algorithm.

Future work

- Reformulate the optimization problem so that it can be solved by using an iterative algorithm.
- Explore higher order invariants.

Future work

- Reformulate the optimization problem so that it can be solved by using an iterative algorithm.
- Explore higher order invariants.
- Explore invariants for other application domains (images, text, etc.).

Future work

- Reformulate the optimization problem so that it can be solved by using an iterative algorithm.
- Explore higher order invariants.
- Explore invariants for other application domains (images, text, etc.).
- Improve random projections by constructing an orthogonal space from random vectors.

Thank you for your attention!

Questions?