UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
– BARCELONATECH

BACHELOR'S THESIS

Using Random Fourier Features with Random Forest

Author:
Albert RIBES

Supervisor:
Lluís A. BELANCHE

Computer Science

April 10, 2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONATECH

Abstract

Facultat d'Informàtica de Barcelona (FIB)
Computer Science

Bachelor Degree in Computer Science

Using Random Fourier Features with Random Forest

by Albert RIBES

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

- En 3 idiomas

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem to study

Supervised Learning uses statistical and mathematical models to predict a response variable from a classification or regression problem. Usually it does so by trying to minimise an error function or to minimise an accuracy function. The better a model is, the higher the accuracy it will obtain on new, unseen data.
Increasing the accuracy of the models is one of the main topics in the field. But it is not easy to achieve it. Usually, it comes at the cost of increasing the computation time to produce the model. Therefore, a trade-off needs to be made between the accuracy obtained and the training time.
In this project we study some new approaches to improve the trade-off of currently used Machine Learning methods. In particular, we study how Kernel approximation techniques could make some procedures feasible in very large datasets and also how to increase the accuracy of some models at the expense of some more time.

1.2 Project proposal

- Existe una batería de técnicas que son buenas, pero que nadie las ha combinado. Son:
  - Modelos simples
  - Ensembles
  - kernel trick
  - Aproximaciones de kernel
- La propuesta es combinar todo esto para mejorar el trade-off
- Sostenemos las siguientes hipótesis:
  - Se podría hacer un ensemble con modelos distintos a DT
  - Se puede aproximar una RBF-SVM pero con el coste de una lineal
  - RFF + Bootstrap quizá es demasiado aleatorio
  - Los modelos que no se basan en productos escalares no se beneficiarán tanto de usar RFF
- Lo que se hará en cada capítulo del trabajo

The current development of Machine Learning has opened many fronts and techniques trying to solve many of the difficulties that the field has.
One known issue is the Bias-Variance dilemma. While solving a classification or regression problem, the expected generalization error of the models is the sum of three error terms: the bias, the variance and the irreductible error. While the last one, as the name sugests, can not be reduced, since it is caused by the inherent random noise in the data, the other two terms seem to have an inverse proportion: trying to reduce one of them increases the other one most of the times. In an attempt to reduce both of them (or at least their sum) there have been developed some ensemble methods. While they tend to show good results, their usage is mostly restricted to a small set of models. This is because they tend to work better on unstable models.
Another advance in Machine Learning has been the usage of kernel methods. They are useful to transform the data into another dimension with better properties, such as a clear dividing margin between classes of data. They are very effective, but their high computational costs has caused their use to be limited to just some specific problems or with a small number of instances. There are some less expensive approaches to approximate this methods, but they are not widely used.
And then there is a collection of classical algorithms which tend to have the advantage of being very simple and straightforward, although they don't usually get the highests scores.
There is a collection of techniques that have shown some good results by their own, but still they haven't been tested in combination with the others. If we could mix some of these methods, maybe we could find new Machine Learning methods, with better accuracy or trade-off.
In this project, we try some combinations of currently known techniques which could produce better results or show new useful model designs. For the one hand, we try to extend the usage of ensemble methods to new basic

models. Currently, it doesn't make much sense to train an ensemble of Support Vector Machines or of Logistic Regression models, because they are so stable and most of the estimators would predict the same answer. We propose the use of random kernel approximations such as Random Fourier Features (RFF) or the Nyströmmethod to increase the unstability of these methods and thus be able to succesfully train and ansemble with them, hopefully increasing the accuracy of a single estimator.
On the other hand, the usage of these Random Kernel approximations could allow us to use some methods which right now are not accessible. Support Vector Machines cannot be used with non-linear Kernels such as the Radial Basis Function (RBF) kernel on big datasets, since the cost is very high. But if we transform the data to some space almost equivalent to the one of the RBF kernel, we can use a Linear Support Vector Machine, which is less expensive to train, and achieve a similar accuracy. In fact, with the usage of an ensemble, the results could be improved.
For this study, we have formulated some hypothesis and we try to confirm or refuse them based on the experimental results. The hypothesis are:

1. It is possible to achieve a similar accuracy to a SVM with the RBF kernel but with less training time by using RFF or the Nyströmmethod with a linear SVM.

2. It is possible to increase the accuracy of a Logistic Regression model or a SVM by training and ensemble of them with the usage of RFF and the Nyströmmethod.

3. Curently used bagging ensemble method could show a bad behaviour in combination with RFF of Nyströmmethod due to an excess of Randomness, caused by the Bootstrap together with the random mapping.

4. Basic algorithms which are not based on the dot product of the input data such as the Decision Tree wil not benefit so much of the usage of RFF and Nyströmthan those that do, like SVM or Logistic Regression.

In the following pages we show an experimental set up to check those hypothesis and the results of the experiments. In Chapter 2 we explain some of the Machine Learning concepts which are needed to understand the rest of the project. In 3 we explain with more detail how this study is developed and what experiments are executed. In 4 we show the results obtained with the experiments and discuss the hypothesis suggested based on them. In 5 we present the conclussions of this project and propose some future work related to this topic. Finally, in Chapter 6 we present a sustainability report of this project.

Chapter 2

Background Information and Theory

2.1 Machine Learning

- Una definición rápida
- Clasificación y regresión
- Cross-validation
- Qué son los datos de train y test, y por qué se hace esa partición
- Qué es el sobre-ajuste

Machine Leearning uses statistical and mathematical models to give answers to problems when there is no known formula of procedure to compute the answer. In the subfield of Supervised Learning, the objective is to predict a numerical or categorical variable in response to some input data, and the way of doing it is to feed the model with lots of different examples for which we already know the correct answer, and we expect the models to be able to predict the correct answer to instances that it hasn't seen before. When it does, we say that the model is able to generalize.
When a model is trained with some data, there is always a risk of overfiting. For a model to overfit means that it adjusts very well to the data that is has seen, but can't predict the correct answer to new, unseen data. This happens because it has not only from the relevant information, but also from the random noise that the data had, and so it can only memorise, but not generalize. For this reason, when a Machine Learning algorithm is trained the data is split in two subsets, a *Training dataset* and a *Testing dataset*. The training dataset will be used to train the model, while the Testing datasets will be used only to check it. If a model has generalized well, it will achieve a good accuracy score on both the training and the testing dataset, but if it has overfitted it will show good results in the training dataset and bad ones in the testing dataset.
Many models need some parameters to tune the behaviour of the algorithm. For example, some of them are used to adjust how much a model will fit to the data. We usually call these "hyperparameters". The correct value for them is not straightforward, and it is normally chosen with a process called "cross-validation". This process consists of splitting the training dataset in many subsets and check many possible values for the hyperparameters in order to

see which one gets a higher accuracy with unseen data.

2.2 Review de los principales modelos que existen

2.2.1 Decision Tree

- No se basa en productos escalares
- Es extremadamente rápido
- Es más fácil de interpretar que otros modelos
- Es extremadamente inestable
- Cuando se hace un Random Forest, se randommiza un poco, de modo que árboles distintos entrenados con los mismos datos pueden ser destintos
- Es un modelo no lineal

Decision Tree is a predictive model which uses the training data to build a tree where each node splits the data in two sets accoding to some feature, and the leafs contain the set of instances that belong to some class (in classification problems) or that has a similar numerical response variable (for regression problems).
To predict the answer to a new instance, it uses the features to "decide" the nodes to cross until it reaches a leaf. The response given is the most prevailing class in the leaf for classification problems, or the mean of the values of the rest of the instances in the leaf.
Decision Trees have the advantages that it is easy to interpret the tree produced and that it is very fast to build the tree. The way to avoid overfitting is to limit its growth.
These models are very unstable. This means that small differences in the training data can produce very different Decision Trees. This property is very useful to build an ensemble of estimators to produce better aswers. Random Forest is an algorithm that trains many Decision Trees with some sort of randomization.

2.2.2 Logistic Regression

2.2.3 Support Vector Machines

- Inicialmente pensadas para clasificación en 2 clases
- Pero se puede más clases con *one-vs-rest* y también hay formas de hacer regresión
- Se basa únicamente en el producto escalar de sus entradas

- Intenta separar los datos con un híper-plano
- Actualmente es poco eficiente usarlas porque su coste s cúbico con la cantidad de entradas.
- Las fórmulas que quiere optimizar

Support Vector Machine (SVM) is a model that finds in hyperplane that divides the data in two sets. In two-class classification problems, each side of the hyperplane contains the instances of each of the classes. It does so by converting the problem to an optimization one.
Given some data $D = \{\boldsymbol{\chi}, \boldsymbol{y}\}$, where $\boldsymbol{\chi} = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}, \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y} = \{-1, +1\}^n$, the optimization problem consists on finding $\boldsymbol{\alpha} \in \mathbb{R}^n$ the maximises

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j \tag{2.1}$$

subject to

$$0 \leq \alpha_i \leq C; \forall i \tag{2.2}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.3}$$

C is an hyperparameter to tune the ammount of penalization for missclassified instances.
If we compute

$$w = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \tag{2.4}$$

and

$$b = y_i - \boldsymbol{x}_i w \tag{2.5}$$

for any i so that $\alpha_i \neq 0$, we can compute the class of $\boldsymbol{x}_0$ with $sign(\boldsymbol{x}_0 w + b)$.
Note that this algorithm just uses the dot product of the input data, not the data itself. This property is allows us to use the Kernel Trick with them. See 2.4

2.3 Ensemble Methods

- Bagging
  - Inventado por Leo Breiman (referencia)
  - Pretende reducir el sesgo
  - Wikipedia dice que pretende reducir la varianza
  - Es el boosting el que pretende reducir el sesgo
  - Entrenamiento de los estimadores es independiente, se podría hacer en paralelo

  - Actualmente casi solo se usa con DT, debido a su inestabilidad

- Bootstrap
  - Intenta solucionar el problema de que para bagging es bueno que los estimadores sean distintos
  - Idealmente usaríamos un dataset distinto para cada estimador
  - Consiste en hacer un resalmpling con repetición
  - Si la cantidad de instancias del original es la misma que la de cada uno de los subconjuntos, se espera que la proporción de elementos úncos sea de $1 - \frac{1}{e} \approx 0.632$.
  - Si el conjunto original tiene n elementos, y tu haces un subconjunto de tamaño r, puedes esperar que la proporción de elementos del original que sí tienen presencia en el nuevo sea de $1 - e^{-\frac{r}{n}}$
- Random Forest

Ensemble methods are a technique used in Machine Learning to reduce the overall accuracy error of a basic classification or regression model. The idea is that a comitee of models is expected to learn better than a single one.
Some ensemble methods are focused on decreasing the error caused by the variance of the model. One example is *Bagging*. Others are focused on the bias error decreasing the bias error, like *Boosting*.
In Bagging, every model in the ensemble vote with equal weight. Thus, it is important to promote the variance among each of the models, since not doing it would be equivalent to training just one model. Ideally, one would train each of the models with totally different datasets, with no correlation between them. But in practice this is not always possible, because of a limited number of instances to train. One alternative is to use a technique called *Bootstrap*. Bootstrap allows to generate many different instances of a dataset by performing a resampling.
Given a dataset D of size n, Bootstrap generates m new datasets D_i of size n by sampling instances from D uniformly and with replacement. This means that some of the instances in D may be repeated in D_i, and others may not appear at all. With a large n, it is expected that each dataset D_i will contain 63.2% of the instances in D.
Theoretically Bagging could be used with any kind of method. However, for most of them Boostrap is not enought to decorrelate each of the estimator. In practice, Bagging is mostly used with Decision Tree, given that this method produces very different tree with a small variation in the data. Random Forest is an algorithm that trains many Decision Trees with a Bagging. Instead of building the tree in a deterministic way, in each split it choses a random subset of features on which to perform the separation. Besides, it lets the estimators overfit, since it has a positive impact in reducing the overall variance of the Forest.

2.4 The kernel trick

- Teorema de Bochner
- El kernel RBF
  - Su fórmula es ...
  - Equivalencia entre γ y σ
  - La noción de similitud que tiene
  - $\mathcal{H}$ es de dimensionalidad infinita
  - Permite ajustarse infinitamente a los datos, tuneando el híperparámetro
  - σ más pequeño, más sobreajuste
  - γ más grande, más sobreajuste

A Kernel is a function that equals to the inner product of inputs mapped into some Hilbert Space [a] , i.e:

$$\kappa(x, y) = \langle \phi(x)\phi(y) \rangle \tag{2.6}$$

They are interesting in Machine Learning because we don't need to know the explicit function $\phi(\cdot)$. In fact, $\phi(\cdot)$ could map the data to a Hilbert Space with infinite dimensions, and we could still compute $\phi(\boldsymbol{x})\phi(\boldsymbol{y})$ through the kernel κ
Support Vector Machines can benefit a lot of Kernel Functions. SVMs solve an optimization problem to maximise

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j \tag{2.7}$$

in order to find an hyperplane that separates the data points in two classes. But with some problems there may not exist such hyperplane, and so it would be needed to map the data to a different feature space. If we did that, then the function to maximise would be

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^\intercal \phi(\boldsymbol{x}_j) \tag{2.8}$$

As we said previously, SVMs don't work with the data points alone, but just with their inner products. Thus, a Kernel could be used to define the optimization problem as

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.9}$$

This approach has one big advantage: as long as the learning technique relies only on the inner product of the input, the underlying mapping $\phi(\cdot)$ does not need to be explicitly calculated and can, in fact, be unknown.

There are many known Kernels. One that is very popular is the Radial Basis Function Kernel, RBF. This kernel is defined as:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2} \tag{2.10}$$

γ is a free parameter. The value of this Kernel decreases with the euclidean distance of the parameters, so it can be interpreted as a measure of similarity. The feature space of this kernel has infinite number of dimensions.

[a] A Hilbert space is just a generalization of the Euclidean Space which contains the structure of an inner product that allows length and angle to be measured.

2.5 Random Fourier Features

2.6 Nyström

Chapter 3

Project Development

3.1 General Idea

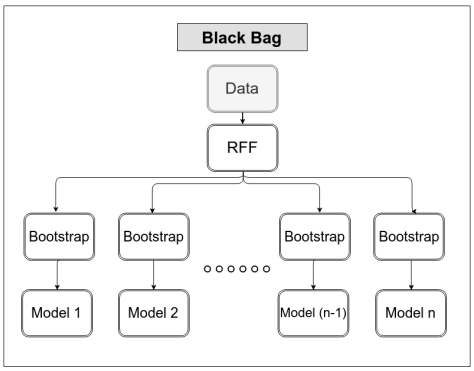
- Hemos visto que se puede sacar una aproximación aleatoria de la función implícita de un shift invariant kernel. Esto tiene 2 ventajas
  - Podemos transformar los datos directamente
  - Podemos producir pequeñas variaciones de un mismo dataset, todas ellas válidas
- Las 4 tipos de modelos que he definido. Referencia a la foto
- ¿Por qué he cogido estos 4 modelos? ¿No podrían haber sido otros? ¿Que tienen estos de bueno? Me he inspirado en Random Forest
- Hay por ahí algún paper que compara RFF y Nyström

Using RFF or Nyström has two main advantages related to this project:

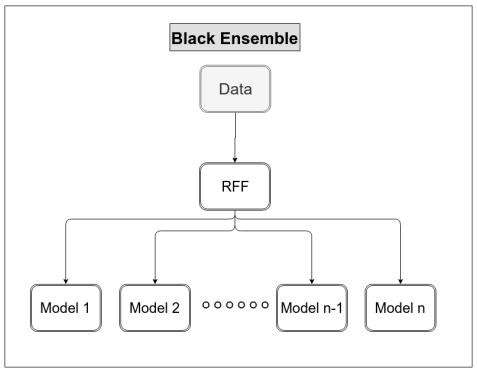
- We can use an explicit mapping of the data instead of the implicit one defined by the Kernel functions.
- We can produce many different datasets (all of them equally valid) from an original one, with the required number of features.

These advantages allow us to define some combinations of the Bagging Ensemble method with the Random Mappings. Depending on where do we place the Random Mapping in the ensemble process we can get two different approaches. If we understand the ensemble as a black box, on which we can only affect the inputs and the outputs of the box without affecting the rest of the process, we get what we have called the Black Box Model. If, on the contrary, we use the Random Mapping in the middle of the ensemble process, we get a White Box Model.
But we have defined two other models based on these ones. Given that we have assumed that maybe a Bootstrap in combination to the Random Mapping woud be too much randomness for the problem, we have defined also the models which doest perform the Bootstrap. We've called a "Bag" the models which do perform a Bootstrap on the data, and "Ensemble" to those that don't perform it. Thus, we have defined four models: "Black Bag", "White Bag", "Black Ensemble", and "White Ensemble". See 3.1

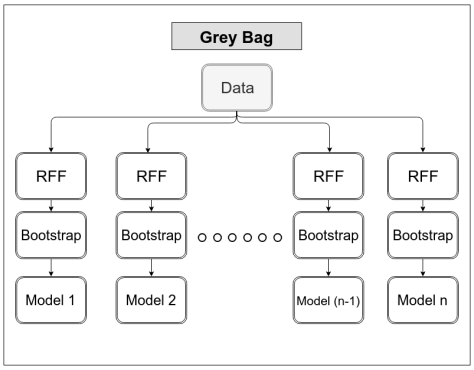
Since the Black Ensemble models will feed all the estimators with the same data, it only makes sense to use it with models with some randomness in the process. That's why we will barely use it here.
These models are based on what is done in Random Forest with the Decision Tree, and seem to be the logical ones to start with. That's why we have chosen to work with them.



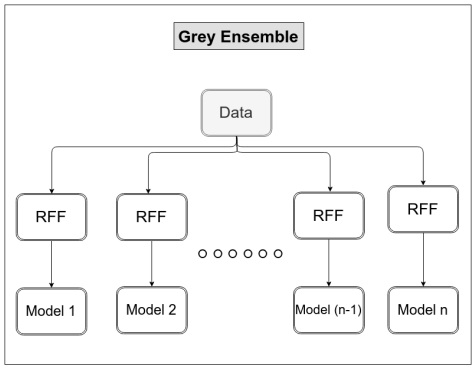
(A) Black Bag Model diagram. Each model recieves data from the same Random Mapping, but they take a different resampling with with replacement.



(B) Black Ensemble Model diagram. Each model recieves exactly the same data, from just one Random Mapping. This model doesn't make sense unless the basic model has some random process.



(C) White Bag Model diagram. Each models recieves data from a different Random Mapping, which has then had a resampling with replacement.



(D) White Ensemble Model diagram. Each model recieves data from a different Random Mapping.

FIGURE 3.1:
There are many ways to mix Ensemble Methods with Random Fourier Features. These are the ones studied in this project.

3.1.1 State of the art con las RFF

- Se ha trabajado poco con ellas. Solo he encontrado 2 usos:
  - Stacked kernel network (referencia): usarlas junto a una red neuronal para tener más niveles de aprendizaje no lineal
  - RFF with SVM (referencia): usar una SVM sin kernel con los datos mapeados usando RFF

3.1.2 State of the art con las Nyström

3.2 Hyper-parameters

- Existen los siguientes:
  - min-impurity-decrease para DT
  - C para SVM
  - gamma para RFF y Nyström
  - cantidad de features para RFF y Nyström
  - cantidad de estimadores para ensembles
- Hemos usado los siguientes valores:
  - Cantidad de features a 500
  - Cantidad de estimadores a 50
  - En modelos simples, el parámetro por crossvalidation
  - En modelos simples con RFF, el parámetro por crossvalidation y una gamma que sobreajuste
  - En modelos con ensemble, parámetros que sobreajusten y la gamma por crossvalidation
  - En RBF-SVM, la gamma por gamest y el parámetro por crossvalidation

With the models defined in this project there are many hyper-parameters to tune the models. These are the hyper-parameters that have been used in the experiments:

Number of features extracted from the kernel The higher this value is, the better the appriximation of the kernel function. We have fixed a value of 500, since this is not important as long as it is not too low.

Ammount of estimators Having a large number of estimator doen't affect negatively the accuracy obteined, but increases the computation time, so the ideal number depends on the computational resources available. For this project we have picked 50 estimators for each Bag/Ensemble.

Gamma parameter of the RBF Kernel A higher value will generate a higher overfit. There is a fast method to find a suitable value for this parameter, explained in [4] . We have chosen this estimation.

Parameters of the simple models Decision Tree use *min_impurity_decrease* to tune the overfit, and SVM use a penalty *C* to do the same. When we train these models without any ensemble, we use Cross-Validation to find a suitable value. When we train an ensemble of these models, as we want them to overfit we set *min_impurity_decrease* to 0 and *C* to 1000, whih is enoght to achieve it.

3.3 Hypothesis

1. Podemos aproximar bien una RBF-SVM
2. Puede tener sentido hacer ensembles con otros modelos a DT
3. RFF + Bootstrap puede ser malo
4. Si el modelo no se basa en productos escalares no se feneficiará tanto

El paper que hacer Linear SVM con RFF no está indexado en ningún sitio. Es solo un pdf que hay por la red

We had proposed these four hypothesis:

1. **It is possible to achieve an acuracy close to using the RBF Kernel but with a lower cost**

   When the number of instances available is too big it is not possible to use an SVM with the RBF kernel, because the cost is cubic with the number of instances, and the optimization problem is too complex. A linear Kernel needs less time, but it may not be suitable for some problems, since data may not be easy to separate.

   If we could first map the data to the new feature space, we could then feed a Linear SVM with it and have the same accuracy with less costs. But this can't be done with the RBF kernel, since the new feature space has infinite dimensions. However, with the use of RFF and Nyström, we can get an approximation of the feature space of the RBF. Using them with a Linear SVM could increase the accuracy on some datasets at almost the same cost.

2. **It could make sense to train ensembles of SVM and Logistic Regression algorithms**

   Since these models are very stable, having an ensemble of them is useless: all of them will allways predict the same answer. There are some methods to randomize a little bit the data, such as Bootstrap, but with these models it is not enough.

   Since RFF and Nyström generate a random mapping of the data, we can achieve a higher level of randomization of the data, while still being a good representation of the real data. Random Mapping can allow us to build ensembles with these two models, increasing the overall accuracy at the expense of some computation time.

3. **Bootstrap together with a Random Mapping may be too much randomization**

   With a simple mix of Bagging with RFF there are two different sources of randomness. For the one hand, Bootstrap generates a random sample of the data with replacement, and on the other hand, RFF and Nyström- perform a Random Mapping of the data to a different feature space.

It is possible that for some models, this is too much randomization of the data, and it coud have a bad effect on the learning process.

4. **Decision Tree does not benefit from RFF and Nyströmas much as Logit and SVM do**

   Kernels were originally used on Support Vector Machines because they were a fast way to implicitly compute the inner product of two vectors in a feature space where data was separable by an hyper-plane. They were useful because SVM just needed the inner products of their input to work.

   RFF and Nyströmare ways to explicitly compute an approximation of that mapping, which doesn't necessarily fits the requirements of Decision Tree, which has nothing to do with the inned products. That's the reason why Decision Tree may not benefit so much of these Random Mappings.

3.3.1 Experiments Proposal

1. Hipótesis: Aproximar RBF-SVM
   1.1. Comparar una RBF-SVM con SVM normal que use RFF
2. Hipótesis: Ensembles con otros
   2.1. Logit normal vs. Logit con RFF
   2.2. Logit normal vs. Logit con RFF Black Bag
   2.3. Logit normal vs. Logit con RFF Grey Bag
   2.4. Logit normal vs. Logit con RFF Grey Ensemble
   2.5. Linear-SVM vs Linear-SVM con RFF
   2.6. Linear-SVM vs Linear-SVM con RFF Black Bag
   2.7. Linear-SVM vs Linear-SVM con RFF Grey Bag
   2.8. Linear-SVM vs Linear-SVM con RFF Grey Ensemble
3. Hipótesis: RFF + Bootstrap
   3.1. Logit con RFF Grey Bag vs Logit con RFF Grey Ensemble
   3.2. Logit con RFF Black Bag vs Logit con RFF Black Ensemble (los dos con un solo estimador)
   3.3. Linear-SVM con RFF Grey Bag vs Linear-SVM con RFF Grey Ensemble
   3.4. Linear-SVM con RFF Black Bag vs Linear-SVM con RFF Black Ensemble (los
4. Hipótesis: DT + RFF
   - DT vs DT con RFF
   - DT vs DT con RFF Black Bag

TABLE 3.1: Information on the datasets used in this project

Dataset	N. Instances	N. Features	N. Classes
Covertype[3]	4900	12	7
Digits[2]	5000	10	10
Fall Detection[10]	5000	6	6
MNIST[6]	5000	717	10
Pen Digits[1]	5000	16	10
Satellite[7]	5000	36	6
Segment[8]	2310	19	7
Vowel[5]	990	11	10
Fashion MNIST[9]	70000	784	10

- DT vs DT con RFF Black Ensemble
- DT vs DT con RFF Grey Bag
- DT vs DT con RFF Grey Ensemble

In order to be able to accept or refuse the hypothesis previously proposed, we have defined a set of experiments.

3.4 Datasets

- 8 Datasets
- Normalizados
- Únicamente tienen variables numéricas, no categóricas
- Únicamente problemas de clasificación
- Algunas cosas particulares que he hecho:
  - Mezclar datos de train y de test para luego hacer mi propia separación
  - Cuando había poca presencia de una clase, hacer un resampling para igualar las cantidades
  - No trabajar cosas como el skiwness o los outliers
  - Eliminar columnas en las que todo eran 0
  - Reducir el conjunto de instancias

We have chosen 9 different datasets to run the experiments proposed. All of them are from classification problems since that was the scope of the project. All the data has been normalized to have a mean of 0 and a variance of 1. Since the kernel RBF can only work with numerical data, all categorical variables

have been converted to integers. Here follows some of the preprocessing that has been done on the datasets:

Covertype In the original dataset the classes were not balanced. So, we performed a resampling of the data, taking 700 random instances of each of the clases. There were 40 binary columns that were converted to a single numerical variable.

MNIST We took a subset of 5000 instances from the original dataset. Since in the new sample some of the columns contained only zeros, they were removed.

Chapter 4

Experimental Results

4.1 Enfrentar resultados 2 a 2

Exp 1-1

- Nunca jamás conseguimos sacar mejores resultados que una RBF-SVM
- Pero los resultados que conseguimos son bastante parecidos en algunos casos
- En el experimento que hamos hecho, los tiempos de los métodos lineales son mucho mayores que los de la RBF-SVM
- Pero hay que atribuirlo al pequeño tamaño del dataset que hemos usado. En un dataset más grande sí se notaría
- Efectivamente, he realizado el experimento en fashion-mnist, y no hay comparación

Exp 2-1

- En general sí conseguimos mejorar sustancialmente un logit
- Pero los tiempos son un poco mayores
- Con datasets mayores, es de esperar que los tiempos no se disparen demasiado

Exp 2-2

- En los mismos datasets en los que un solo logit no había conseguido hacer nada, aquí tampoco han hecho nada
- Entre logit normal y logit con ensemble, la precisión es más o menos la misma, pero el tiempo es mucho mayor
- En el caso logit, realmente puede deberse a que no hemos hecho nada para sobreajustar más.

2-3

- Exactamente igual que el anterior

2-4

- Exactamente igual que el anterior

2-5

- Conseguimos mejorar sustancialmente la precisión de la mayoría de datasets
- Los tiempos son mayores, pero todavía son aceptables
- Suponemos que en datasets más grandes los tiempo serían más parecidos

2-6

- El dataset que se resistía se sigue resistiendo cuando hacemos un ensemble
- Ahora los tiempo sí que són muchísimo más grandes, quizá no sale a cuenta hacer ensemble
- Para ver si sale a cuenta hacer ensemble, habrá que verlo más adelante

2-7

- Exactamente igual que el anterior

4.2 2-8

- Exactamente igual que el anterior

3-1

- No podemos decir que haya una ensemble que claramente sea mejor que los demás
- Da la impresión que en algunos casos el grey es mejor que los otros, pero no es nada concluyente

3-2

- Da la impresión que los black ensembles son un poco mejor que los grey, pero no es nada concluyente

3-3

- Los resultados no son nada concluyentes

3-4

- Exactamente igual que el anterior

4-1

- Usar rff con un solo árbol no es nada beneficioso
- Tanto los errores como los tiempos son más altos

4-2

- En algunos casos ha mejorado sustancialmente, y en otros no
- Pero estamos comparando un solo árbol con 50: claramente los 50 tendrían que ser siempre mejores, y ese no es el caso
- Los tiempos no tienen ninguna comparación
- En algunos casos, incluso empeora el usar un ensemble

4-3

- Exactamente igual que el anterior

4-4

- Exactamente igual que el anterior

4-5

- Exactamente igual que el anterior

4.3 Contrastar hipótesis con resultados

Chapter 5

Conclusion and Future Directions

- Problemas de regresión
- Aproximar otros kernels a RBF
- Ver el comportamiento con problemas que no sean tan bonitos (con missings, clases desbalanceadas, etc)
- Otros tipos de ensembles, como el boosting

- Todo lo que hemos hecho solo sirve con datasets muy grandes. Para pequeños, en general salimos perdiendo
- Hemos conseuido hacerle un boost a logit
- Si usamos RFF, en general no sale nada a cuenta hacer un ensemble. No se mejora demasiado
- Sí que podemos aproximar una RBF-SVM con una lineal a coste lineal

En general los únicos éxitos de este trabajo son:

- Ahora podemos hacer un ensemble de logit y de svm, que antes no se podría
- también hemos conseguido mejorar un poco un solo logit y svm
- Hemos aprendido que da igual el tipo de ensemble que cojamos

Chapter 6

Sustainability Report

6.1 Environmental

6.2 Economic

6.3 Social

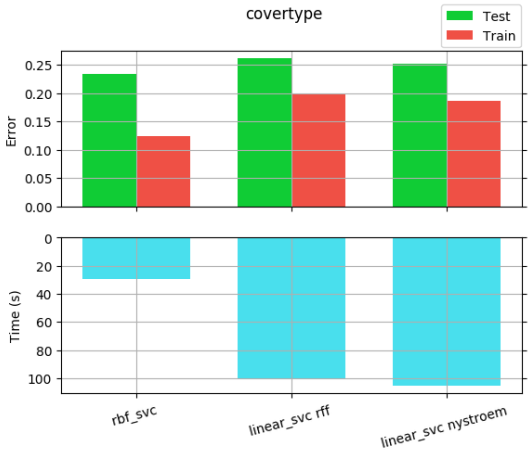
6.3.1 Impacto Personal

6.3.2 Impacto Social

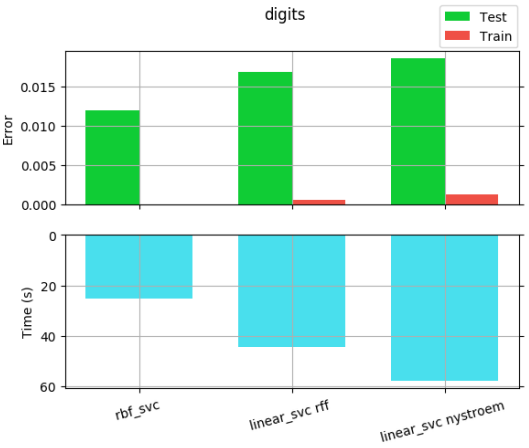
6.3.3 Riesgos Sociales

Appendix A

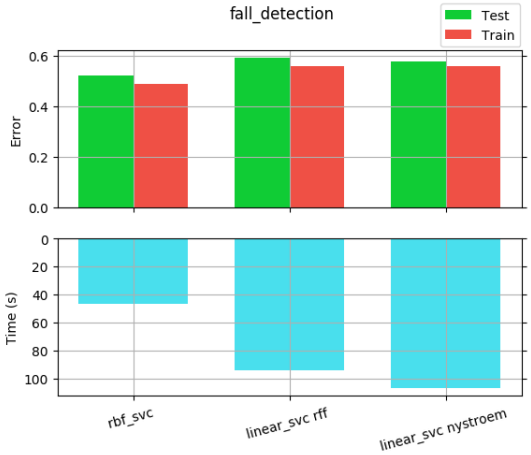
Results of experiment 1.1



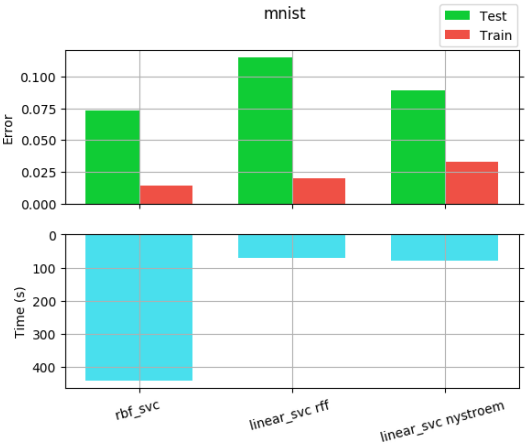
(A) prueba covertype



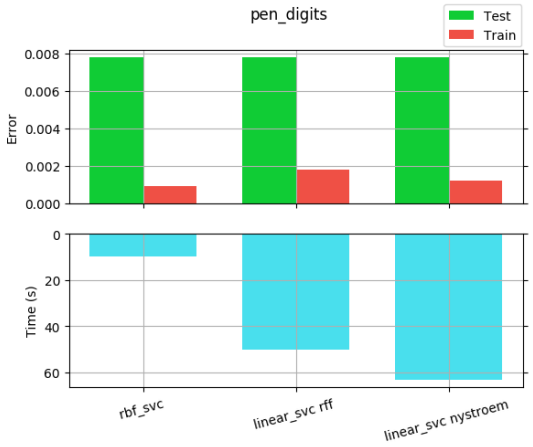
(B) prueba digits



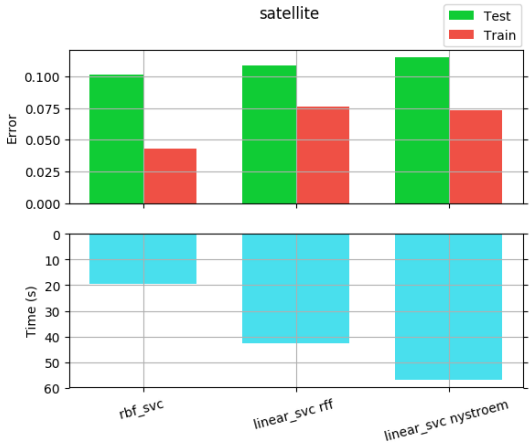
(A) prueba fall-detection



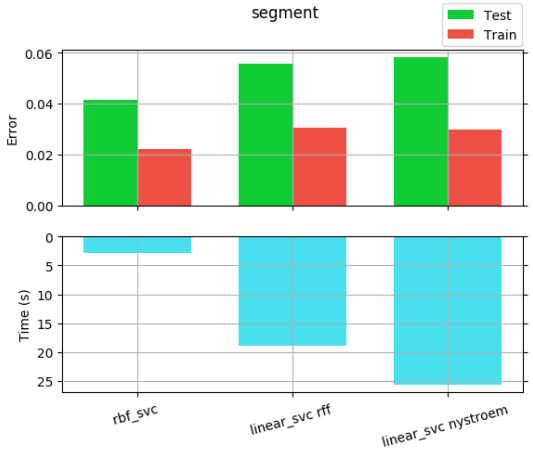
(B) prueba mnist



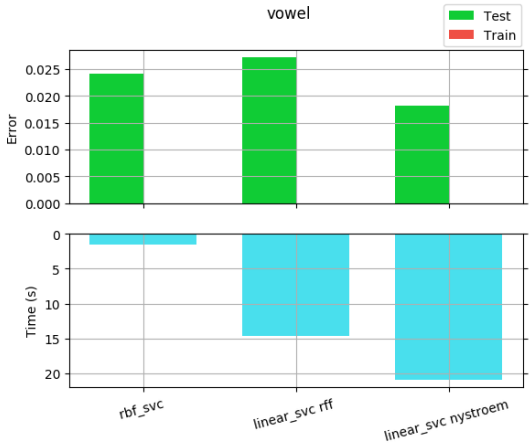
(A) prueba pen-digits



(B) prueba satellite



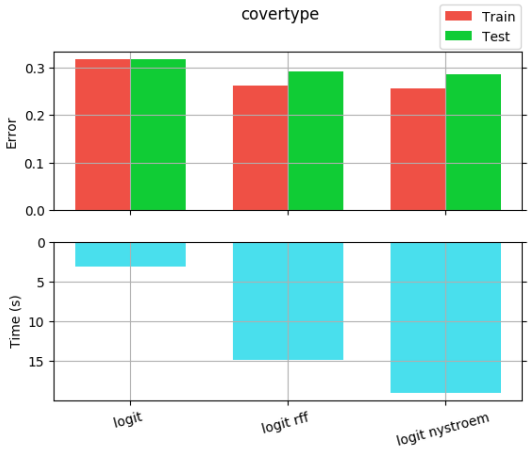
(A) prueba segment



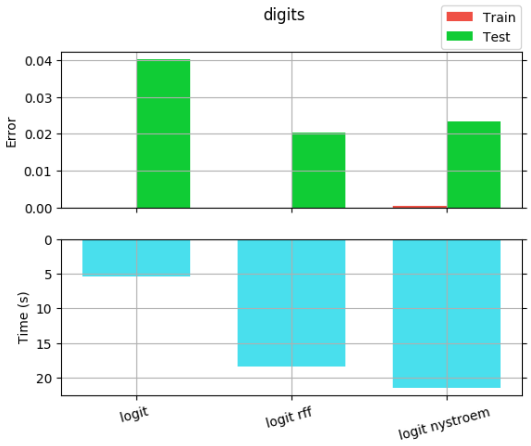
(B) prueba vowel

Appendix B

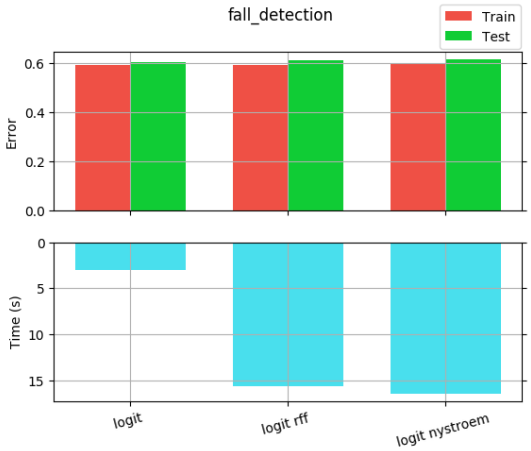
Results of experiment 2.1



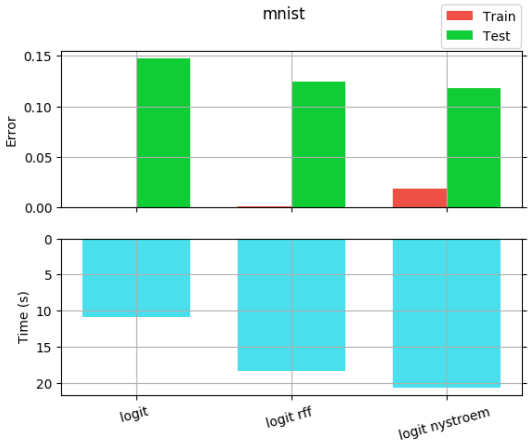
(A) prueba covertype



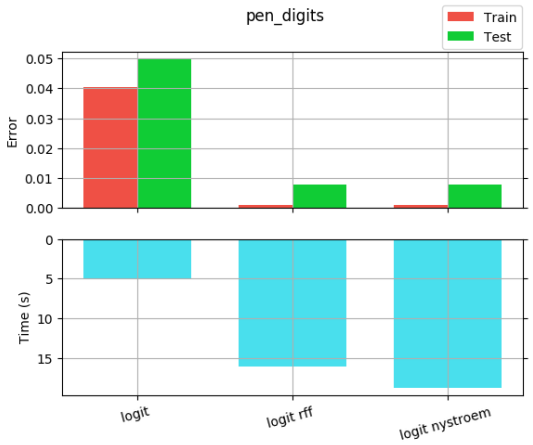
(B) prueba digits



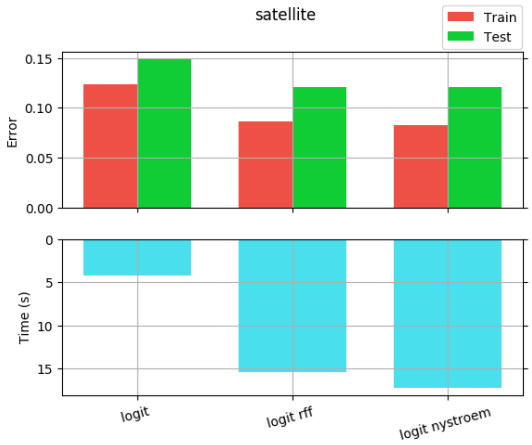
(A) prueba fall-detection



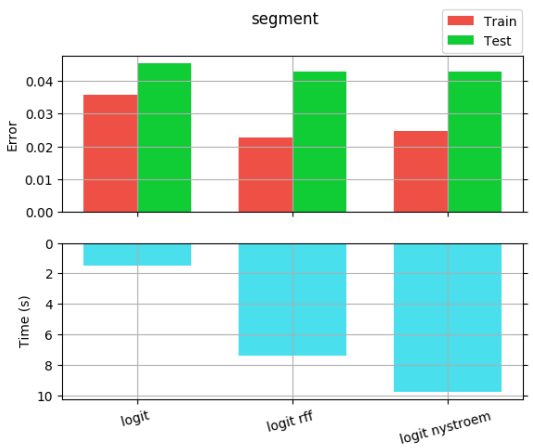
(B) prueba mnist



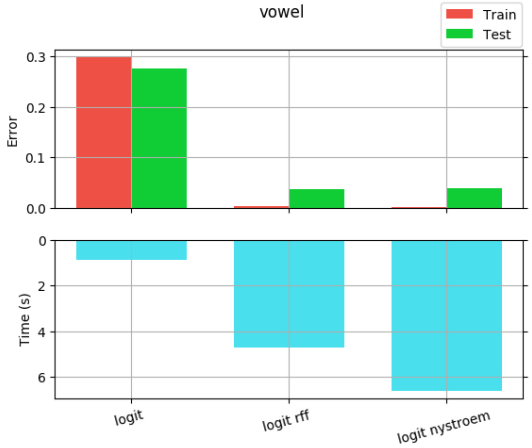
(A) prueba pen-digits



(B) prueba satellite



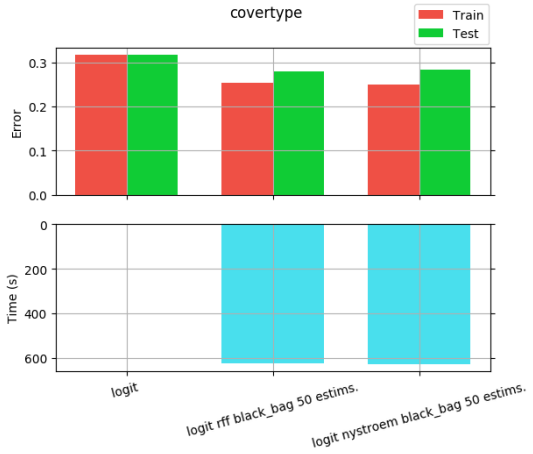
(A) prueba segment



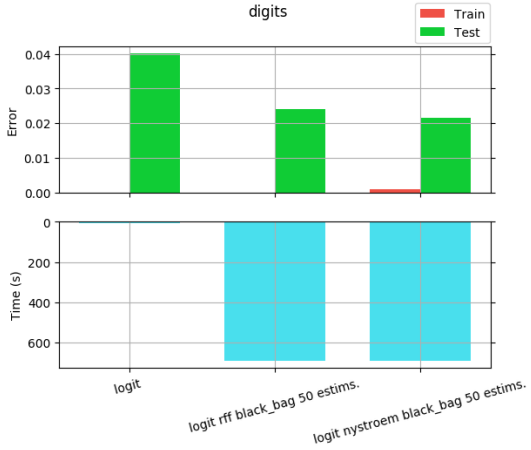
(B) prueba vowel

Appendix C

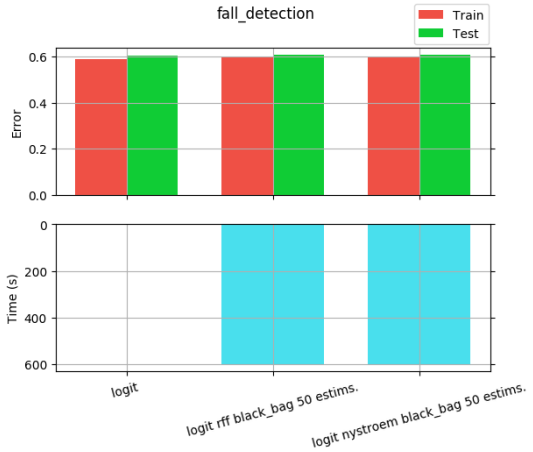
Results of experiment 2.2



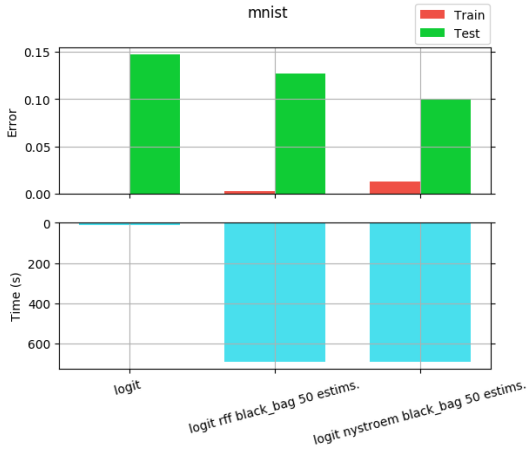
(A) prueba covertype



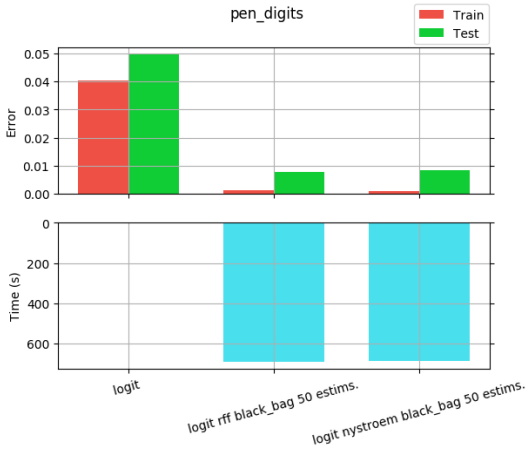
(B) prueba digits



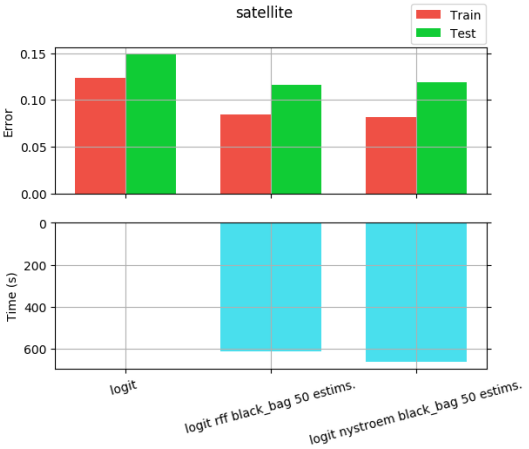
(A) prueba fall-detection



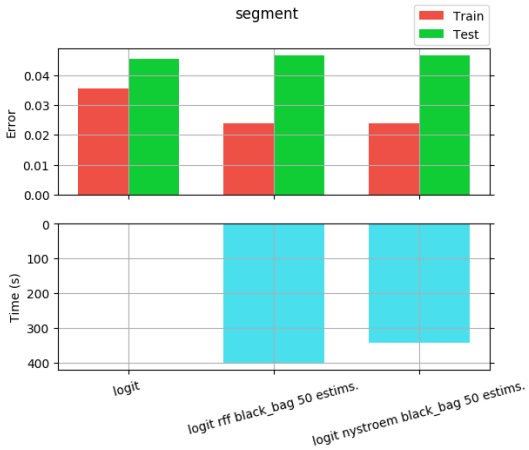
(B) prueba mnist



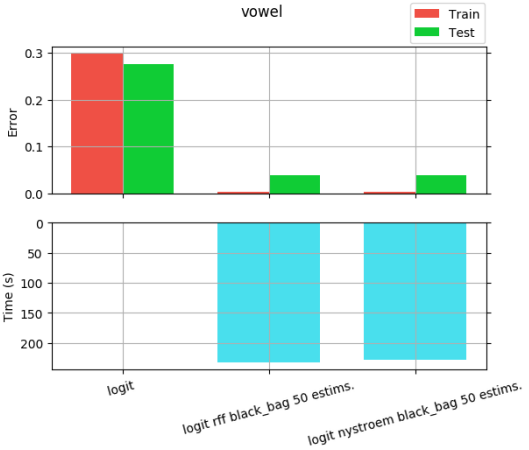
(A) prueba pen-digits



(B) prueba satellite



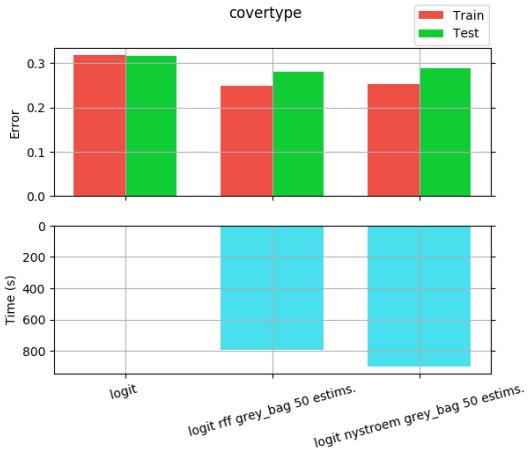
(A) prueba segment



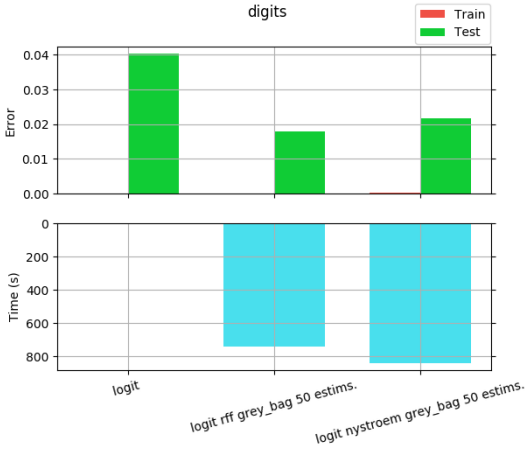
(B) prueba vowel

Appendix D

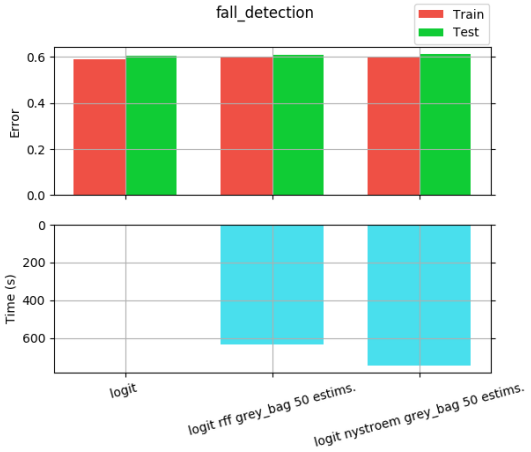
Results of experiment 2.3



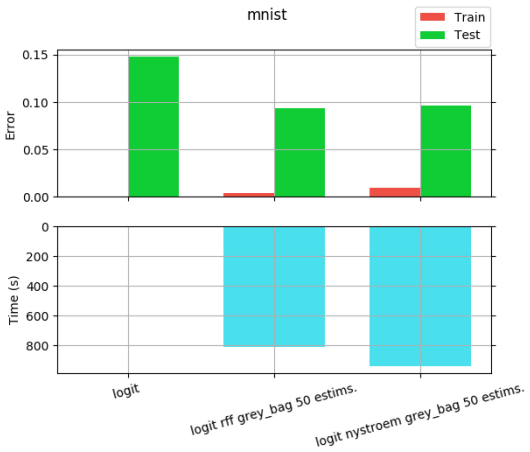
(A) prueba covertype



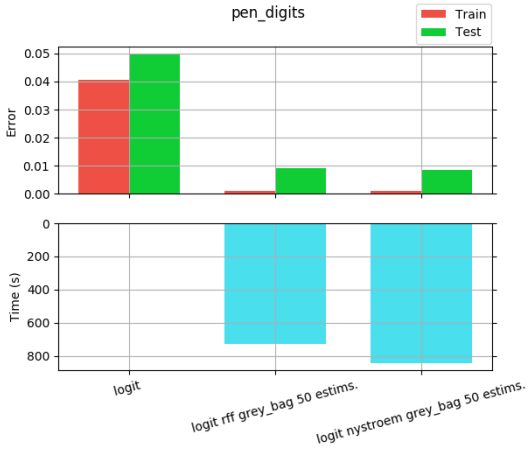
(B) prueba digits



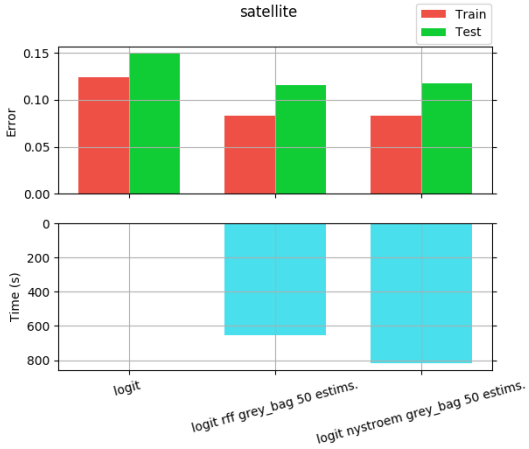
(A) prueba fall-detection



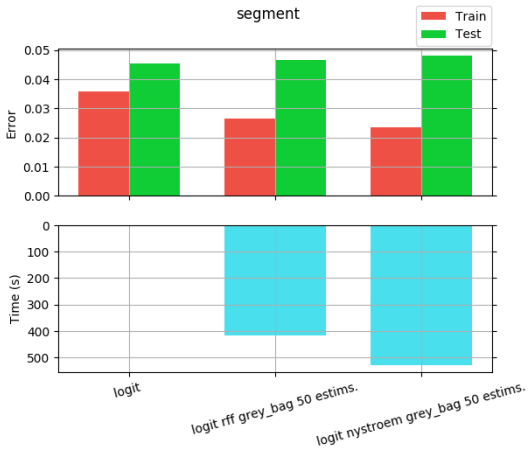
(B) prueba mnist



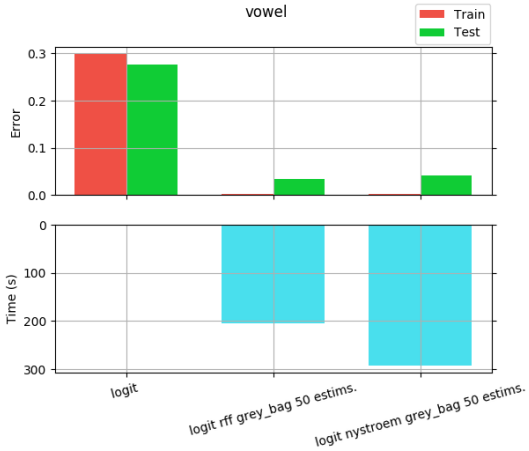
(A) prueba pen-digits



(B) prueba satellite



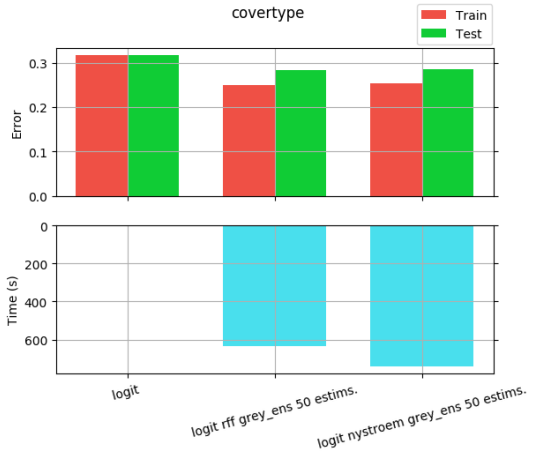
(A) prueba segment



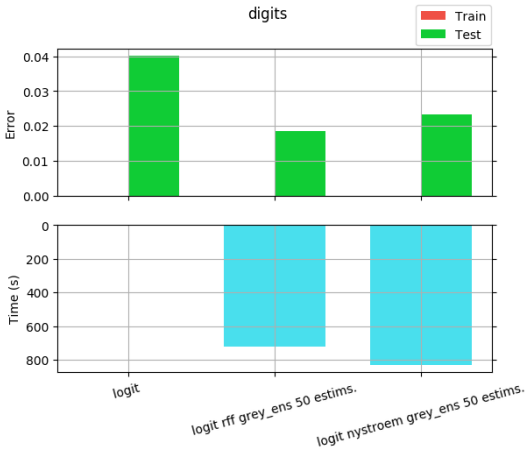
(B) prueba vowel

Appendix E

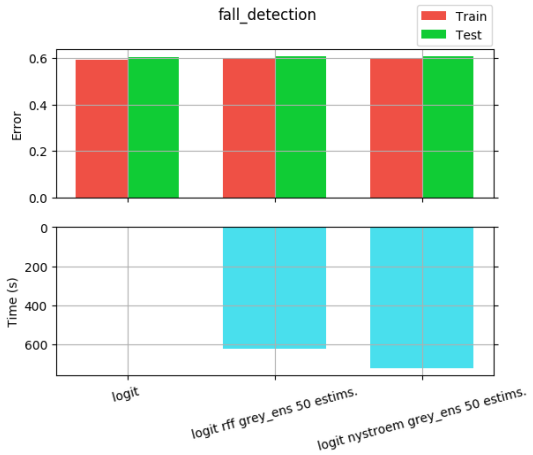
Results of experiment 2.4



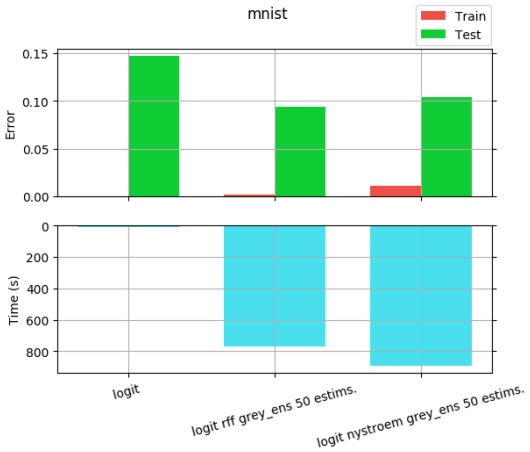
(A) prueba covertype



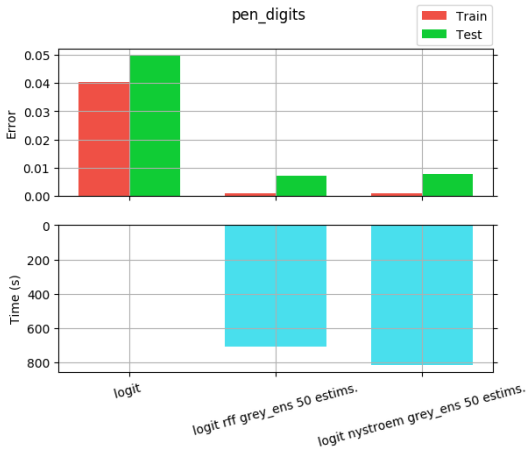
(B) prueba digits



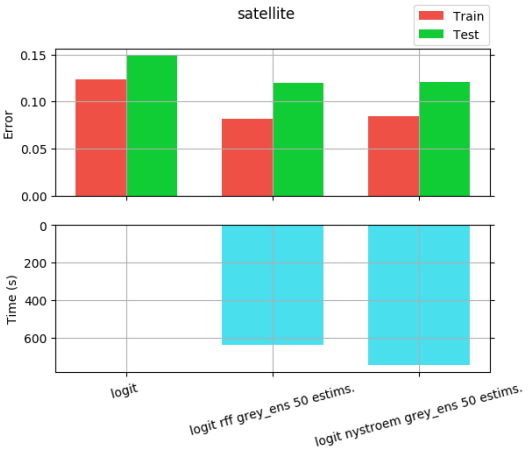
(A) prueba fall-detection



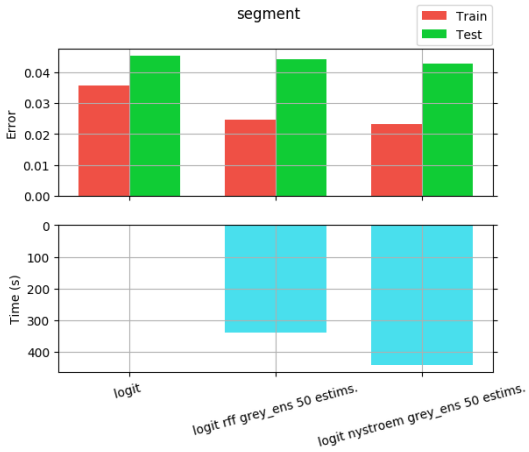
(B) prueba mnist



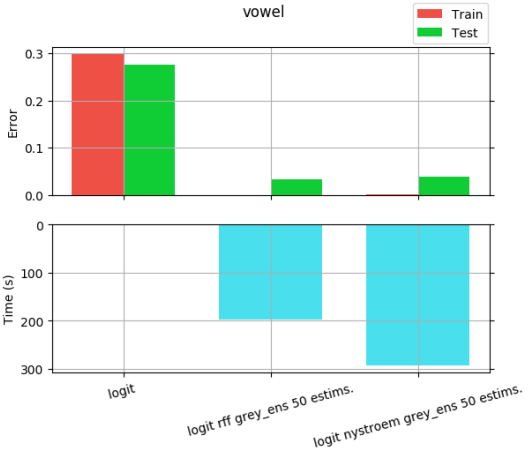
(A) prueba pen-digits



(B) prueba satellite



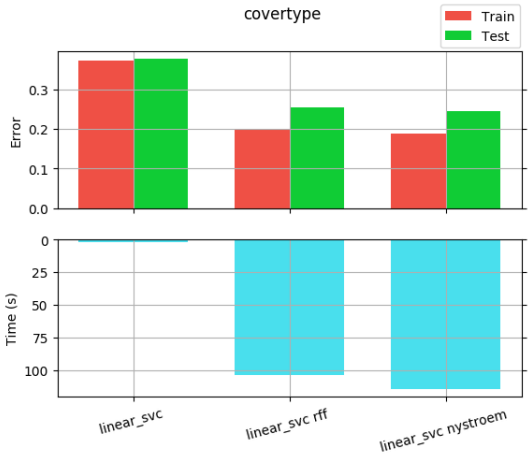
(A) prueba segment



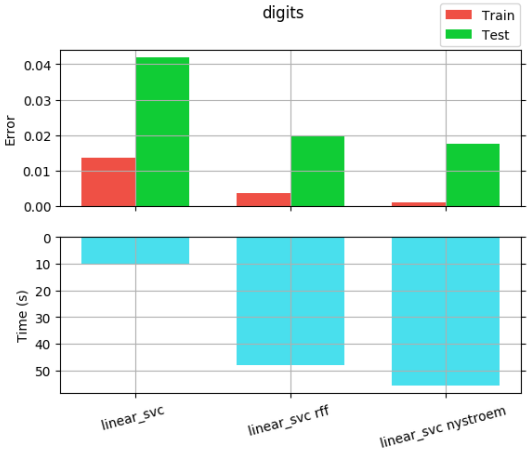
(B) prueba vowel

Appendix F

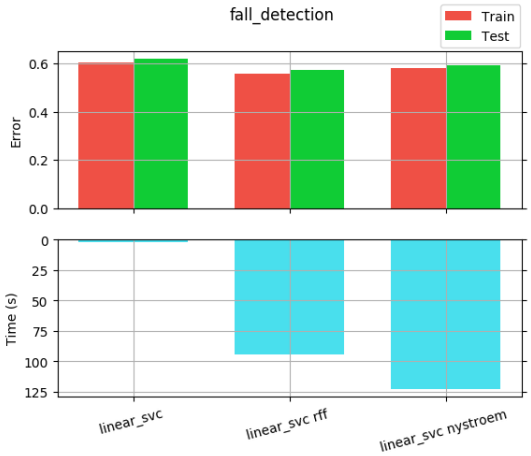
Results of experiment 2.5



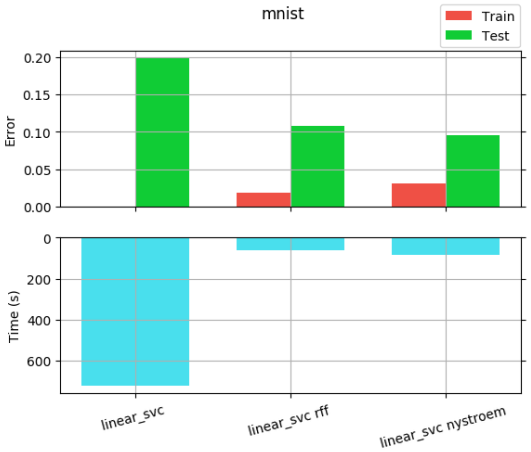
(A) prueba covertype



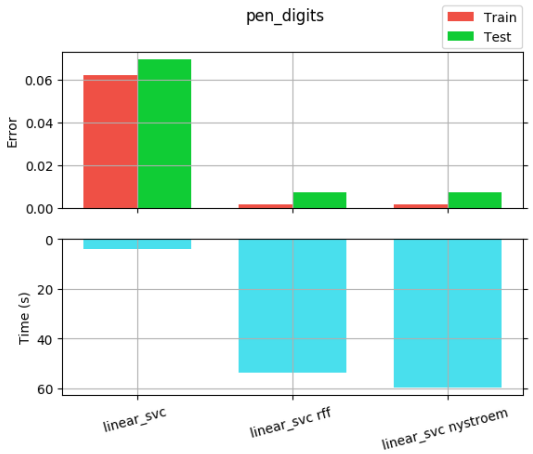
(B) prueba digits



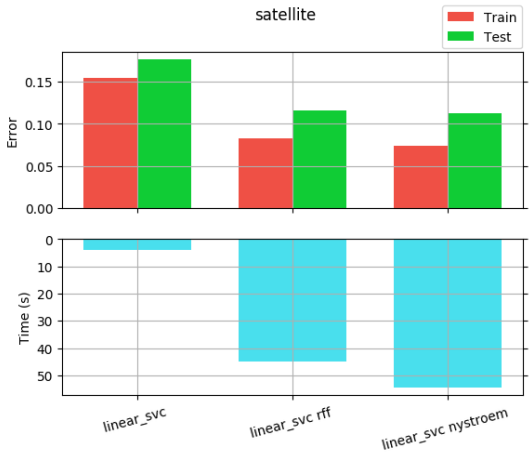
(A) prueba fall-detection



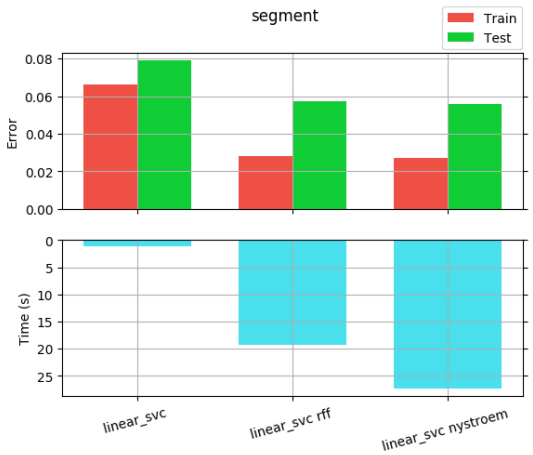
(B) prueba mnist



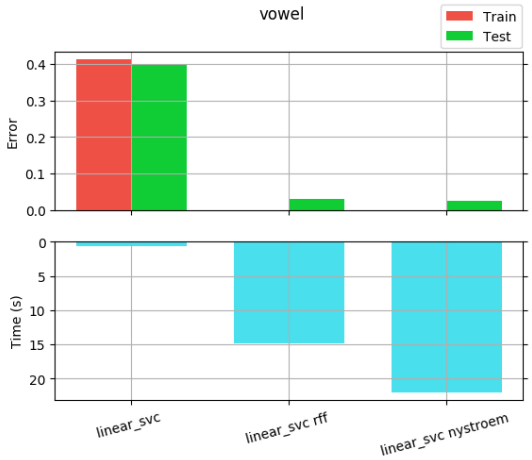
(A) prueba pen-digits



(B) prueba satellite



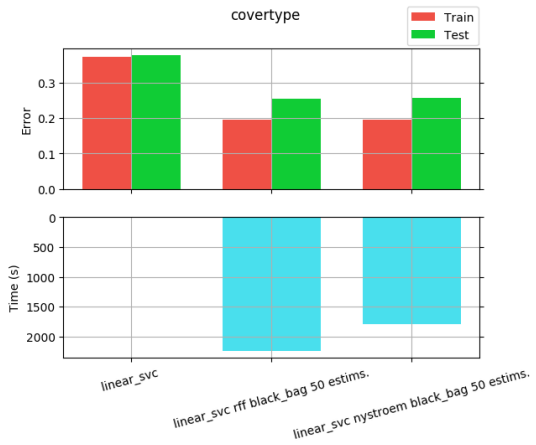
(A) prueba segment



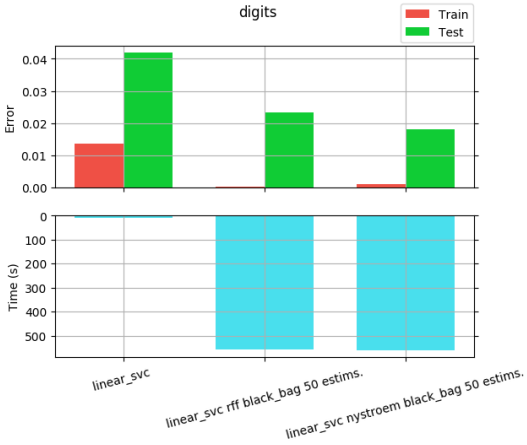
(B) prueba vowel

Appendix G

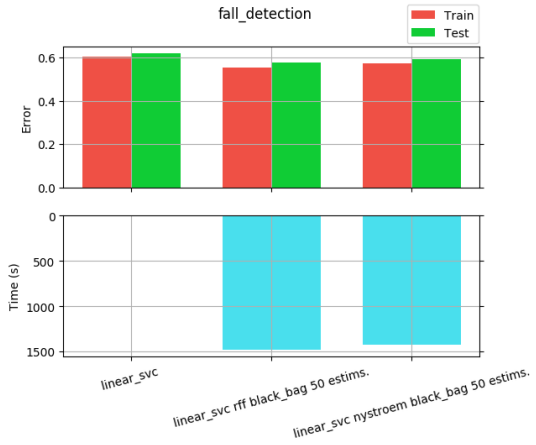
Results of experiment 2.6



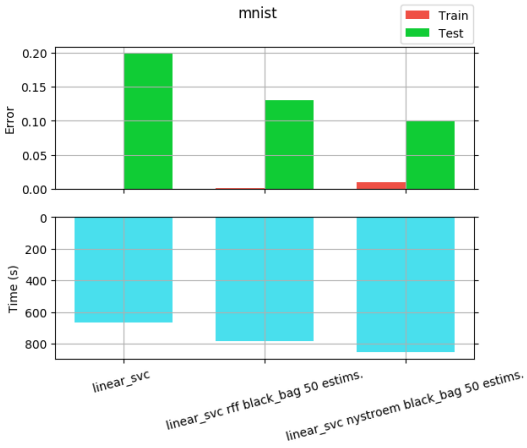
(A) prueba covertype



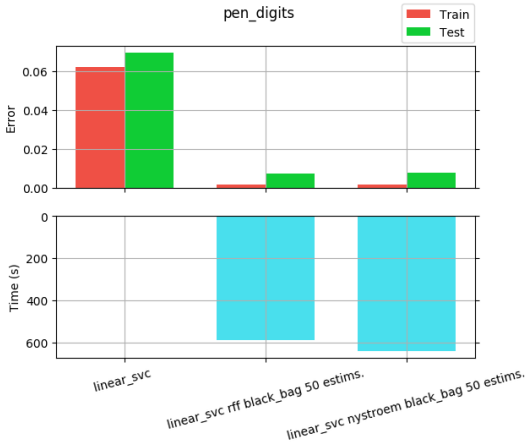
(B) prueba digits



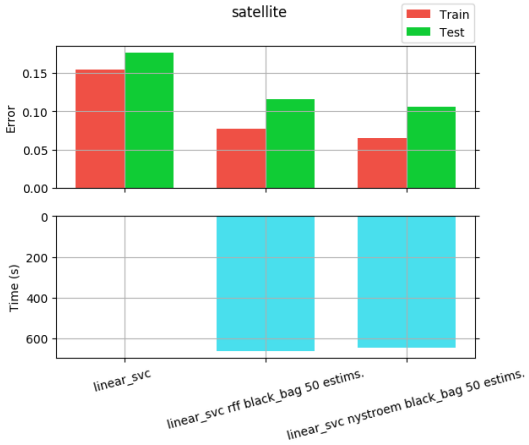
(A) prueba fall-detection



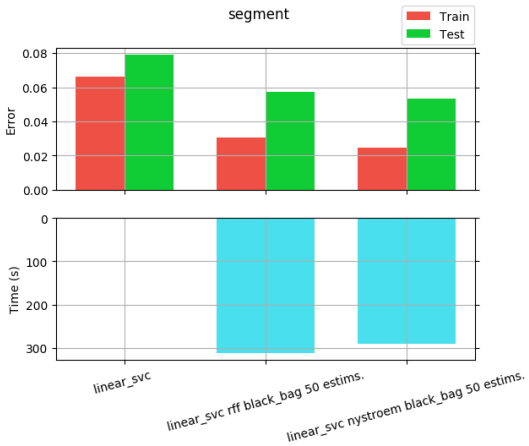
(B) prueba mnist



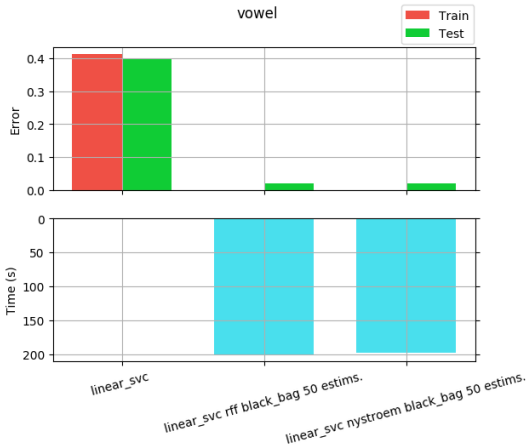
(A) prueba pen-digits



(B) prueba satellite



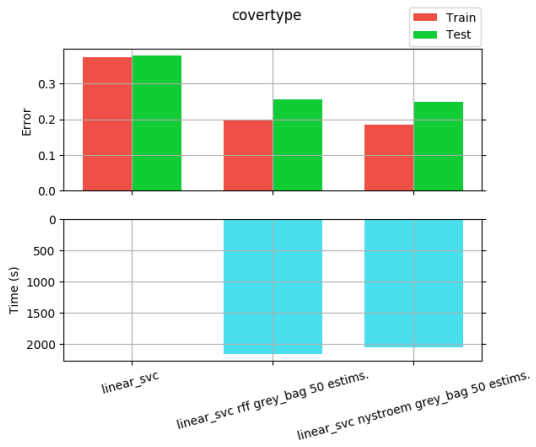
(A) prueba segment



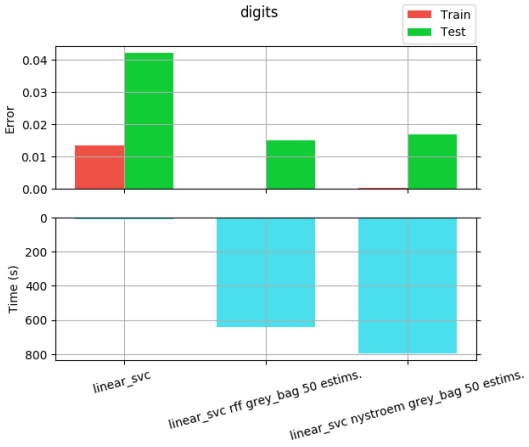
(B) prueba vowel

Appendix H

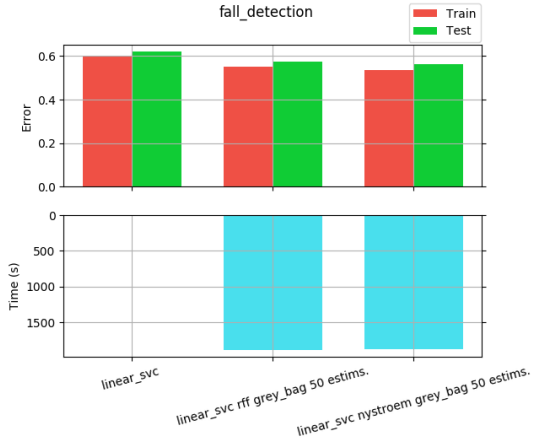
Results of experiment 2.7



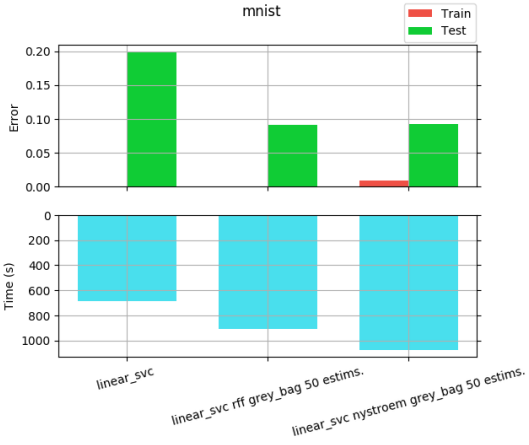
(A) prueba covertype



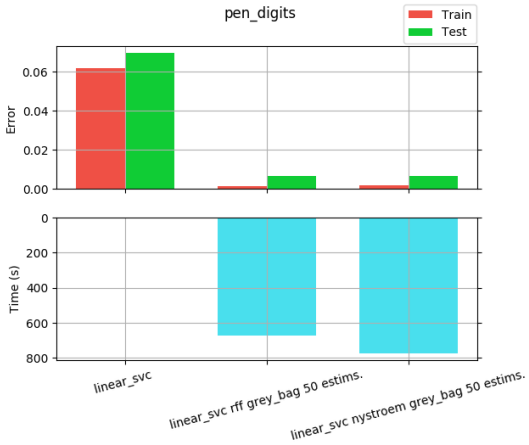
(B) prueba digits



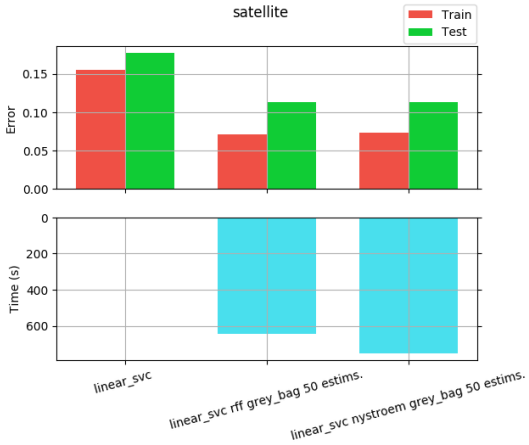
(A) prueba fall-detection



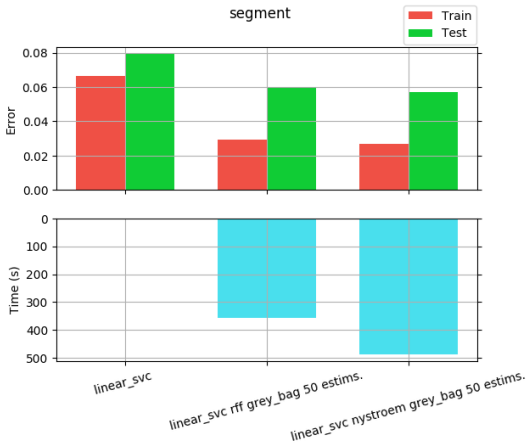
(B) prueba mnist



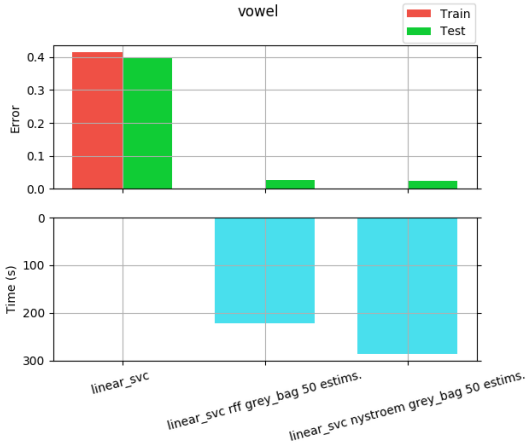
(A) prueba pen-digits



(B) prueba satellite



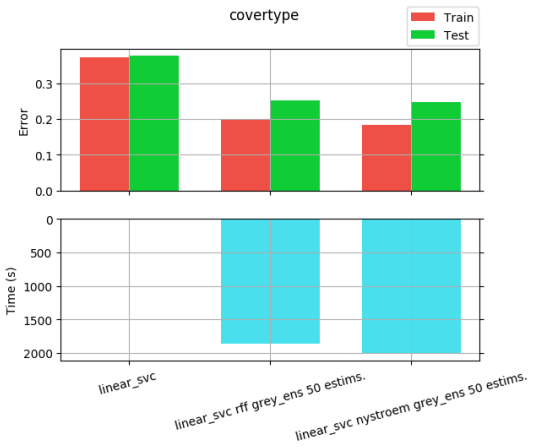
(A) prueba segment



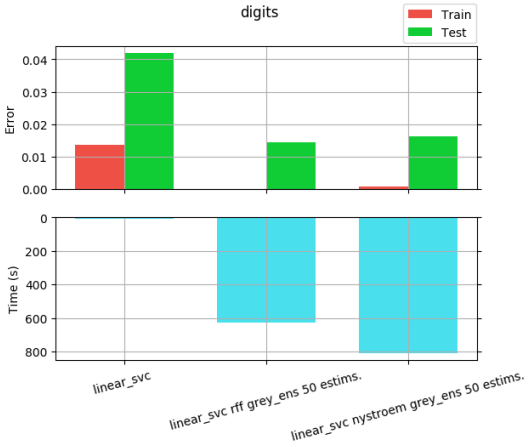
(B) prueba vowel

Appendix I

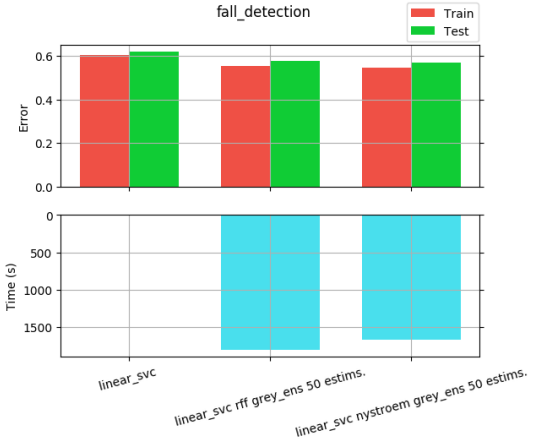
Results of experiment 2.8



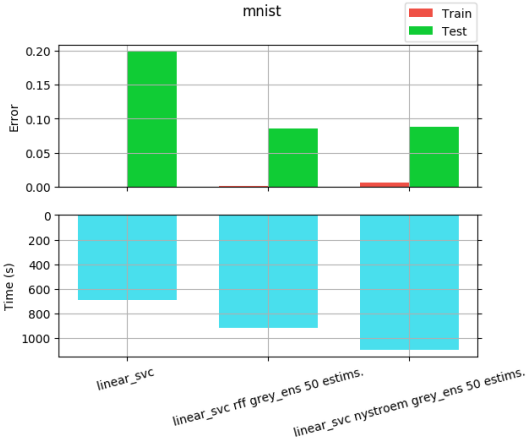
(A) prueba covertype



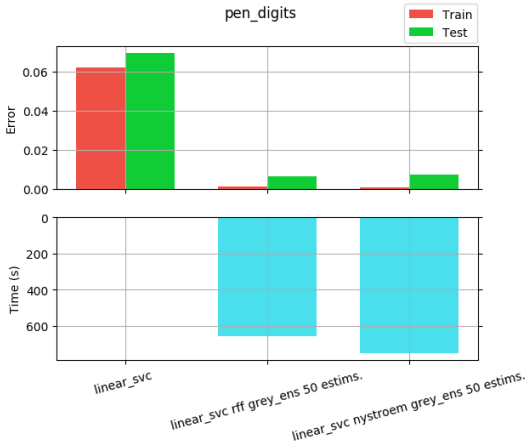
(B) prueba digits



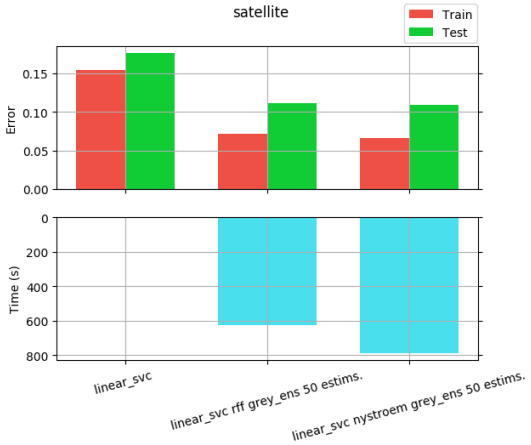
(A) prueba fall-detection



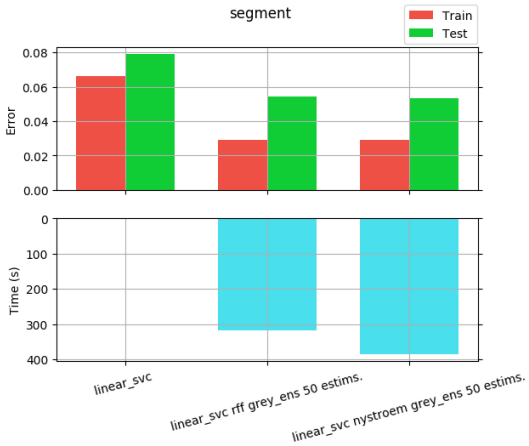
(B) prueba mnist



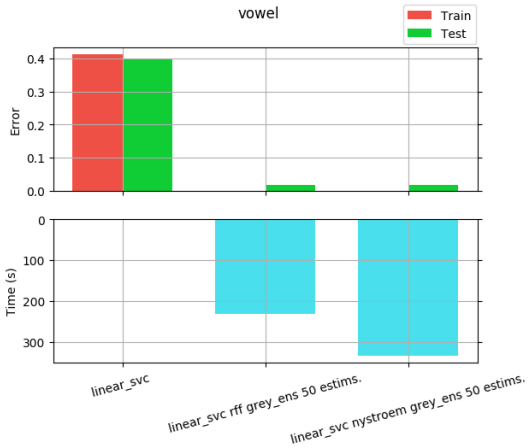
(A) prueba pen-digits



(B) prueba satellite



(A) prueba segment



(B) prueba vowel

Bibliography

[1] E. Alpaydin and Fevzi. Alimoglu. *Pen-Based Recognition of Handwritten Digits*. July 1998. URL: https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.

[2] E. Alpaydin and C. Kaynak. *Optical Recognition of Handwritten Digits*. July 1998. URL: http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits.

[3] Jock A. Blackard, Dr. Denis J. Dean, and Dr. Charles W. Anderson. *Forest Covertype data*. Aug. 1998. URL: https://archive.ics.uci.edu/ml/datasets/covertype.

[4] Barbara Caputo et al. "Appearance-based object recognition using SVMs: which kernel should I use?" In: *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*. Vol. 2002. 2002.

[5] David Deterding, Mahesan Niranjan, and Tony Robinson. *Vowel Recognition (Deterding data)*. URL: https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data).

[6] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *MNIST database*. URL: http://yann.lecun.com/exdb/mnist/.

[7] Ashwin Srinivasan. *Statlog (Landsat Satellite)*. Feb. 1993. URL: https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite).

[8] University of Massachusetts Vision Group. *Image Segmentation data*. Nov. 1990. URL: http://archive.ics.uci.edu/ml/datasets/image+segmentation.

[9] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[10] Özdemir, Ahmet Turan, and Billur Barshan. *Detecting Falls with Wearable Sensors Using Machine Learning Techniques*. 2017. URL: https://www.kaggle.com/pitasr/falldata.