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 9, 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 solve

- Trade-off between accuracy and train time is not good

Machine Learning has shown it can be very useful when trying to predict a numerical or categorical variable based on some input data. It is able to define mathematical and statistical models which can help us on a lot of different fields, specially those that still need the presence of a human to take some decision.
However, in most of the situations a trade-off needs to be made between the ammount of precision in the predictions of the model (the accuracy) and the ammount of time the model needs to define the prediction function based on the data. There are many classification and regression problems that still require very powerful computers and a lot of "training time" in order to produce decent answers, and some of them still can't be solved with a sufficient level of accuracy.
A lot of research is being done in the scientific community trying to improve this trade-off. New models are defined, variations to old ones, techniques to approximate more complex methods, etc. As the field progresses, it is possible to deal with problems that where out the scope of Machile Learning. But still, a lot of work needs to be done.

1.2 Why is it important?

- Avances en este campo permitirían usarlo en otras ciencias como medicina, economía, sociedad
- Muchas tareas que ahora tiene que hacer un humano podría hacerlas una máquina, ahorrando tiempo y dinero

These days being able to learn from the data has many important applications. Big companies make use of Machine Learning techniques in order to be more efficient. Having good and cheap learning models helps them to have a better, faster and more efficient decision making.

1.3 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 highest 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

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

2.2.2 Logistic Regression

Hola que tal

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

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

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 híper-parámetro
  - σ más pequeño, más sobreajuste
  - γ más grande, más sobreajuste

2.5 Random Fourier Features

2.6 Nyström

Chapter 3

Project Development

3.1 General Idea

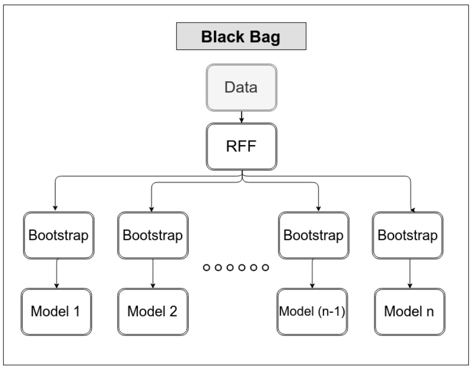
- Hemos visto que se puede sacar una aproximación aleatoria de la función implícita de un shift invariant kernel. Esto tiene 2 ventajas
  - Podemos transformar los datos directamente
  - Podemos producir pequeñas variaciones de un mismo dataset, todas ellas válidas
- Las 4 tipos de modelos que he definido. 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 Nyströmhas two main advantages related to this project:

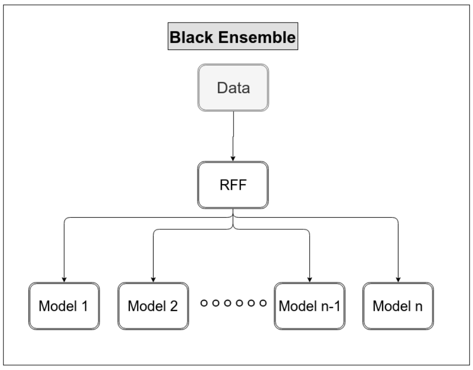
- We can use an explicit mapping of the data instead of the implicit one defined by the Kernel functions.
- We can produce many different datasets (all of them equally valid) from an original one, with the required number of features.

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

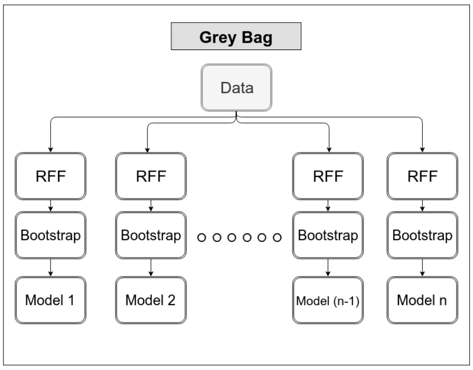
Since the Black Ensemble models will feed all the estimators with the same data, it only makes sense to use it with models with some randomness in the process. That's why we will barely use it here.
These models are based on what is done in Random Forest with the Decision Tree, and seem to be the logical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



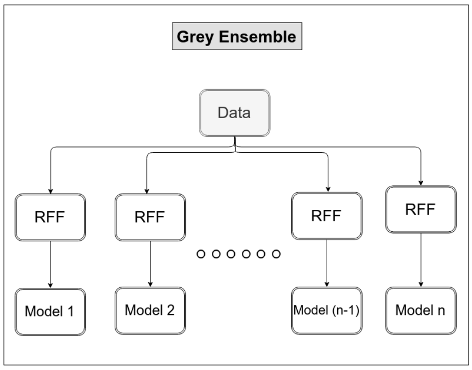
(A) Black Bag Model diagram. Each model recieves data from the same Random Mapping, but they take a different resampling with with replacement.



(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 apprixmation 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 doesn'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 Caputo et al., 2002 . 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 Nyströmperform 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

- 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

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

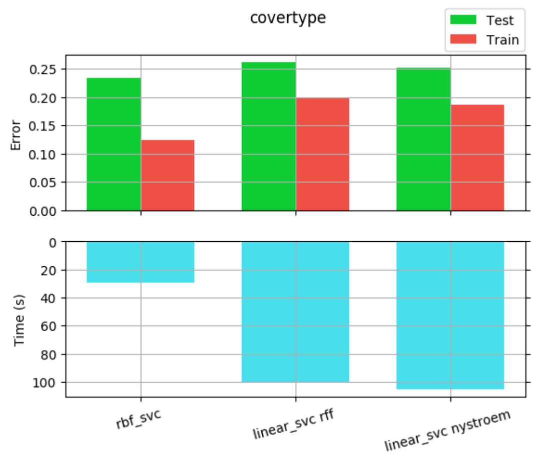
6.3.1 Impacto Personal

6.3.2 Impacto Social

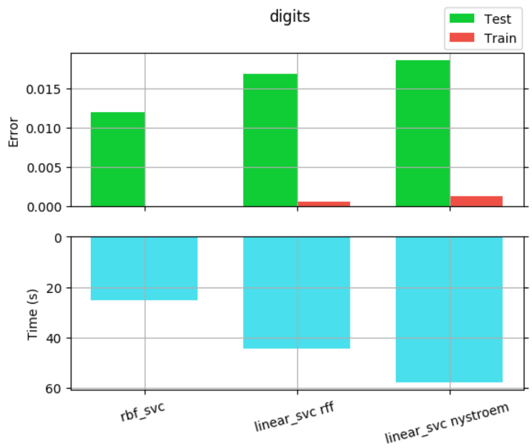
6.3.3 Riesgos Sociales

Appendix A

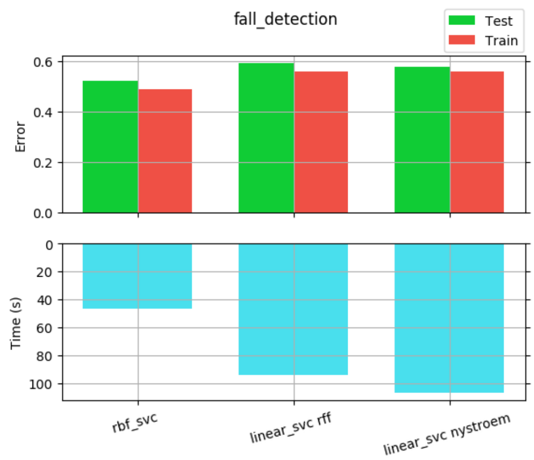
Results of experiment 1.1



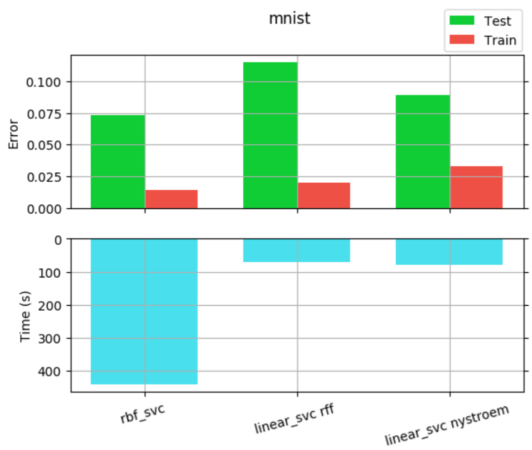
(A) prueba covertype



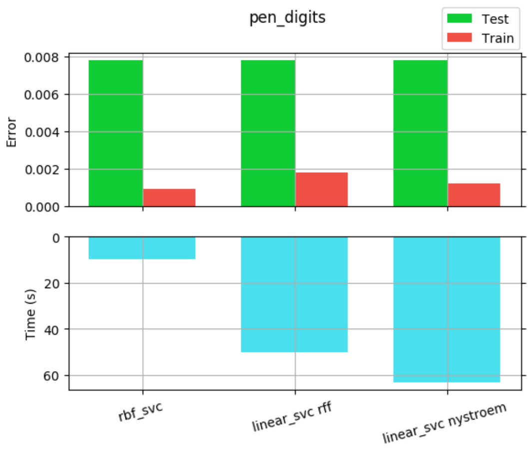
(B) prueba digits



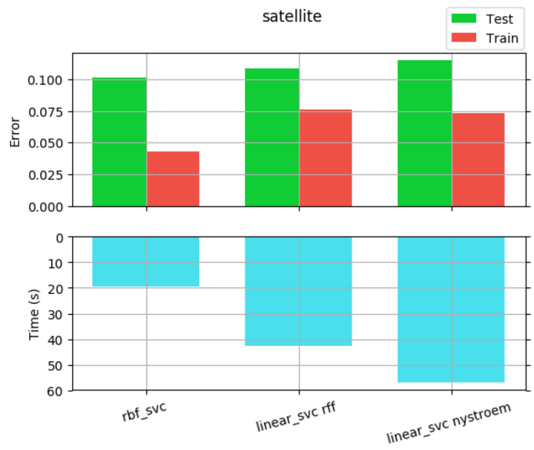
(A) prueba fall-detection



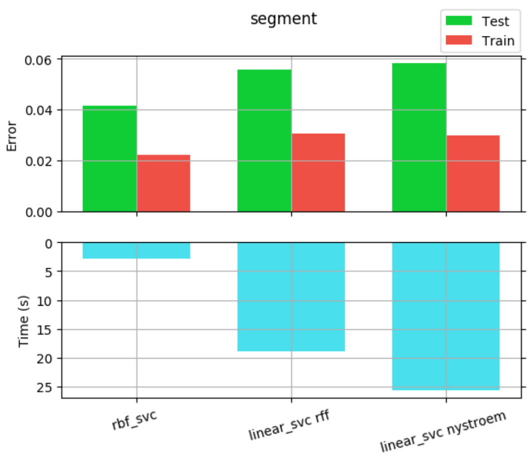
(B) prueba mnist



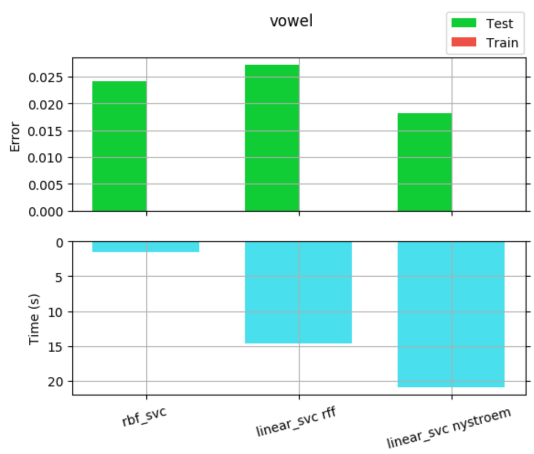
(A) prueba pen-digits



(B) prueba satellite



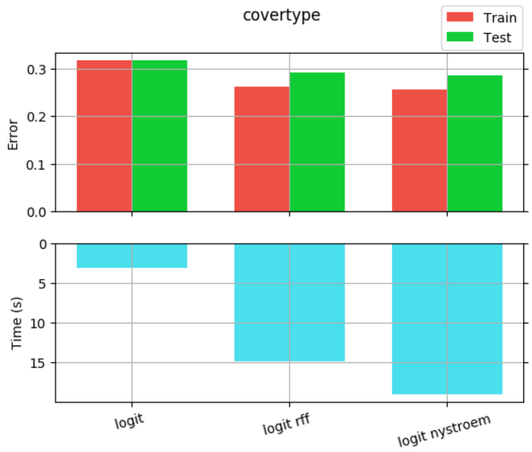
(A) prueba segment



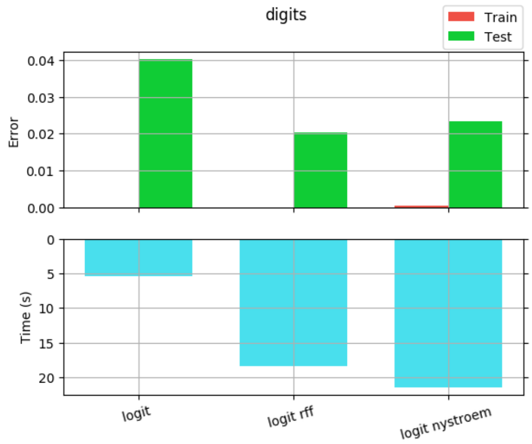
(B) prueba vowel

Appendix B

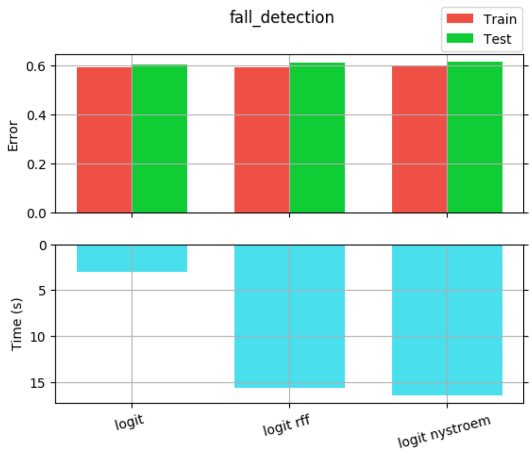
Results of experiment 2.1



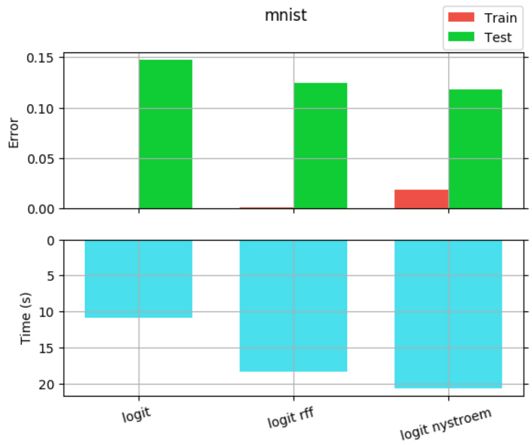
(A) prueba covertype



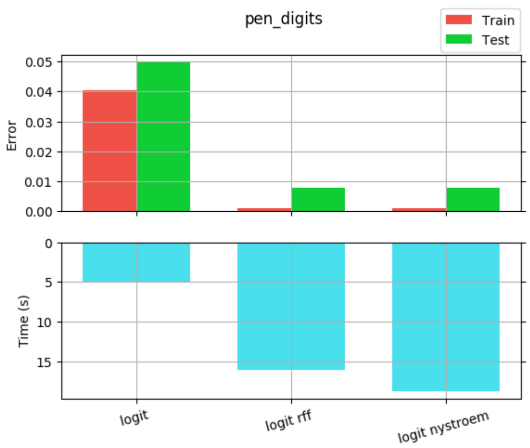
(B) prueba digits



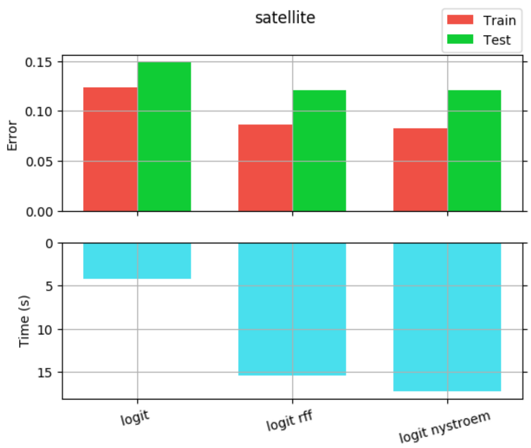
(A) prueba fall-detection



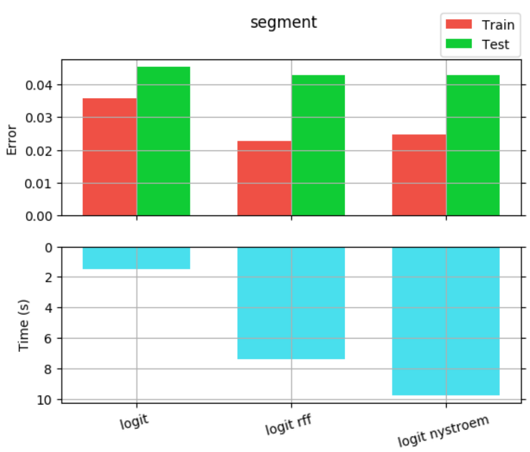
(B) prueba mnist



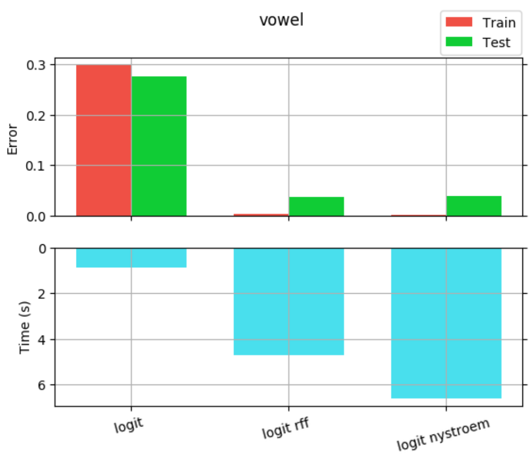
(A) prueba pen-digits



(B) prueba satellite



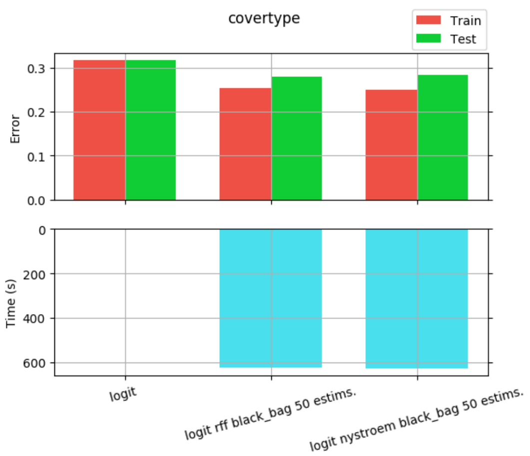
(A) prueba segment



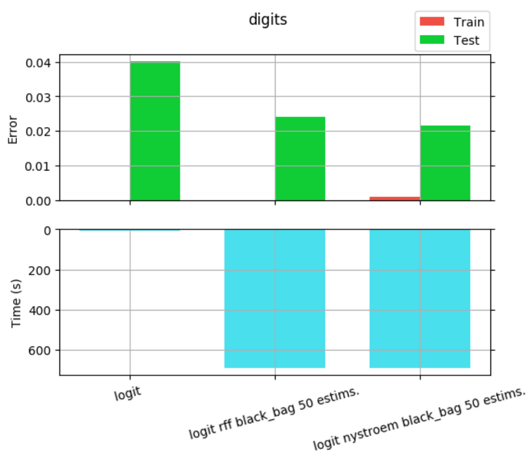
(B) prueba vowel

Appendix C

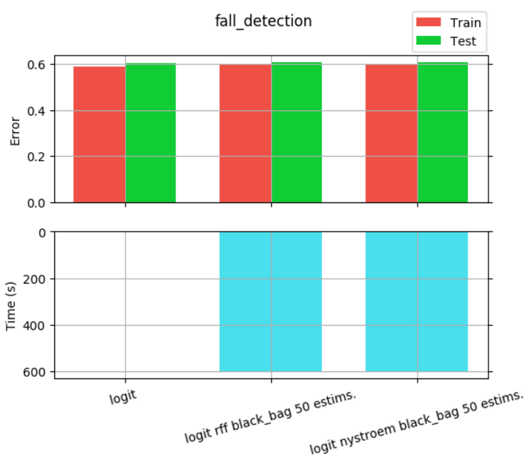
Results of experiment 2.2



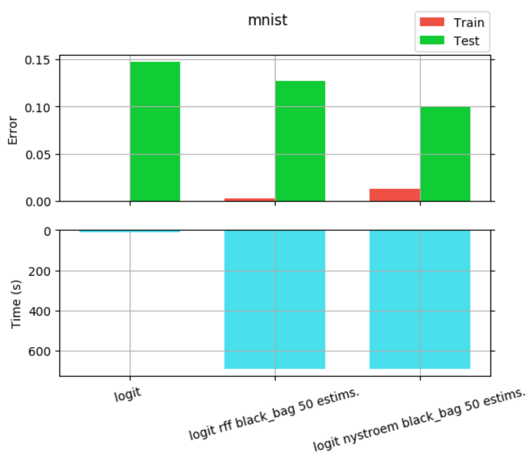
(A) prueba covertype



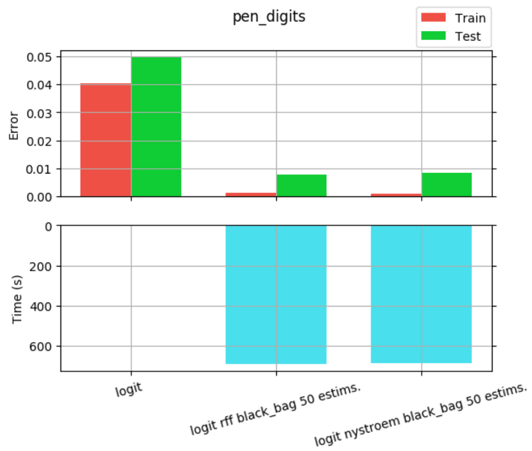
(B) prueba digits



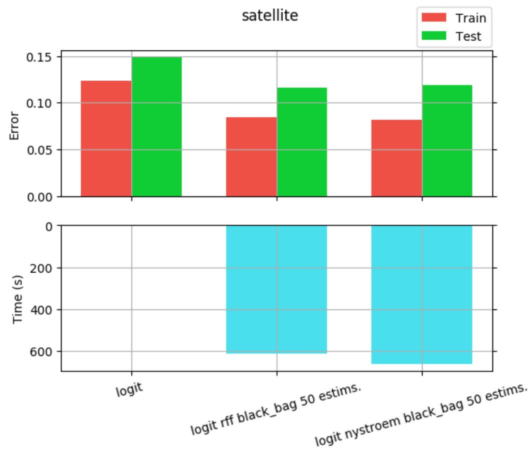
(A) prueba fall-detection



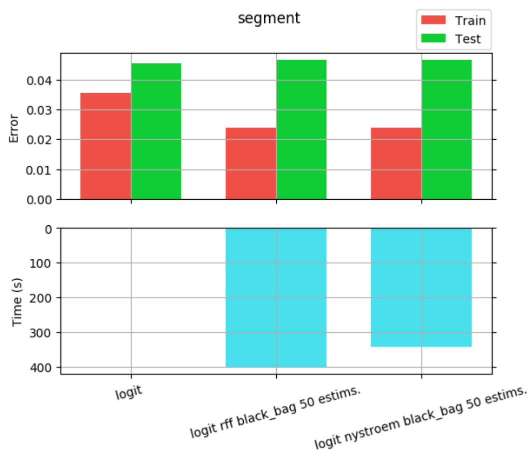
(B) prueba mnist



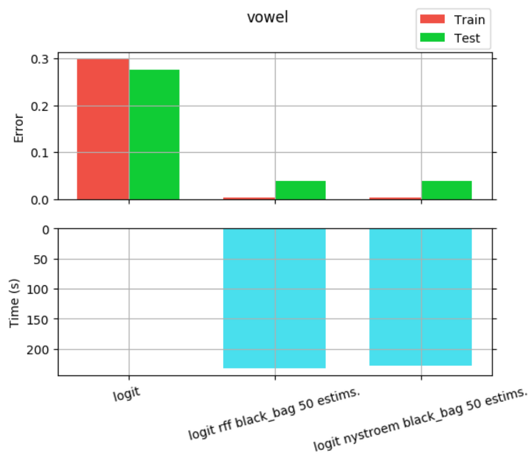
(A) prueba pen-digits



(B) prueba satellite



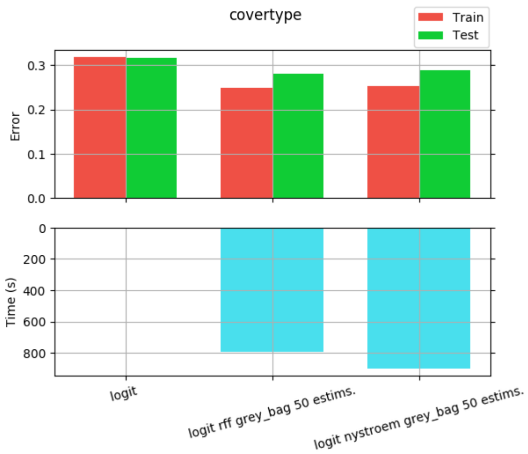
(A) prueba segment



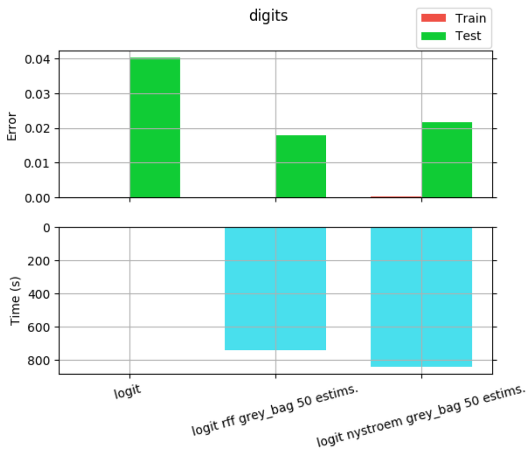
(B) prueba vowel

Appendix D

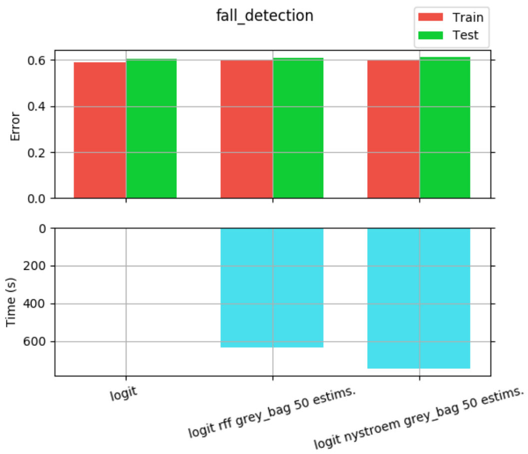
Results of experiment 2.3



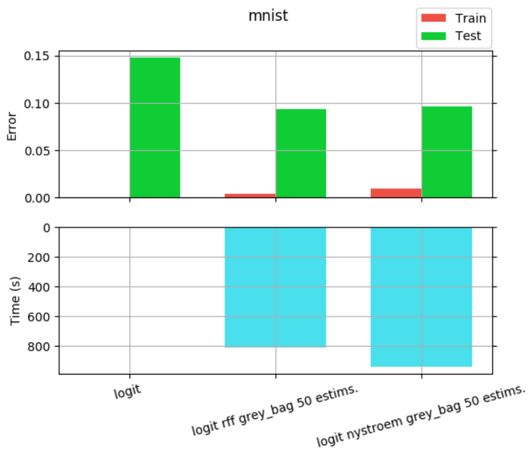
(A) prueba covertype



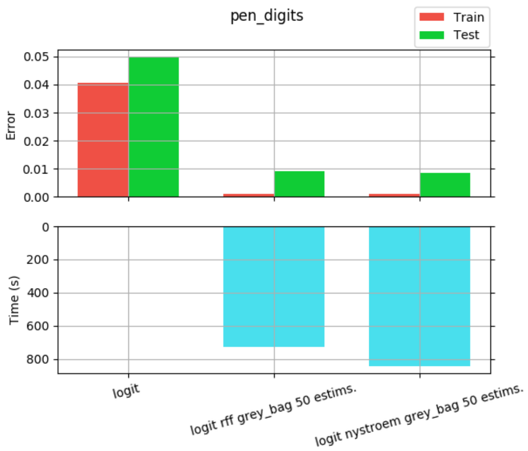
(B) prueba digits



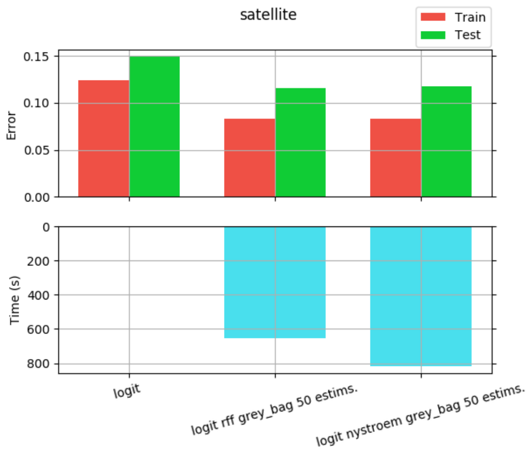
(A) prueba fall-detection



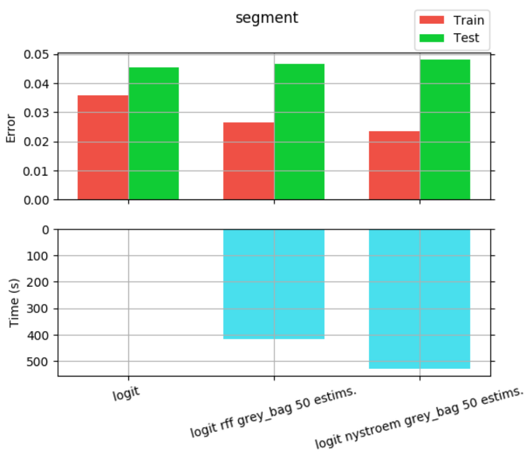
(B) prueba mnist



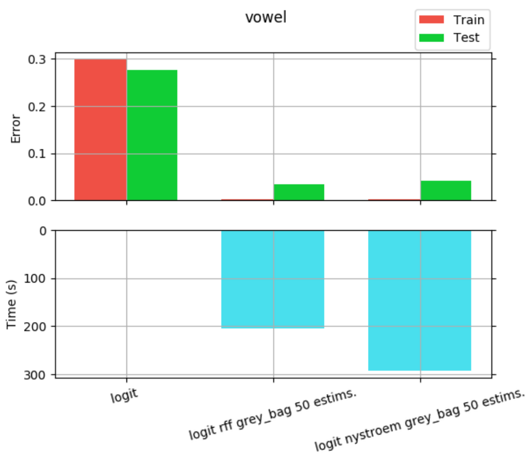
(A) prueba pen-digits



(B) prueba satellite



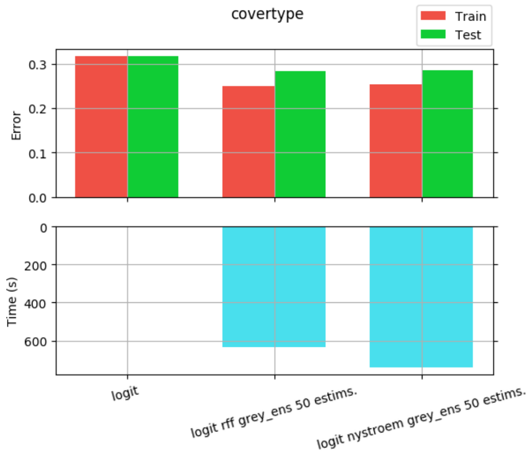
(A) prueba segment



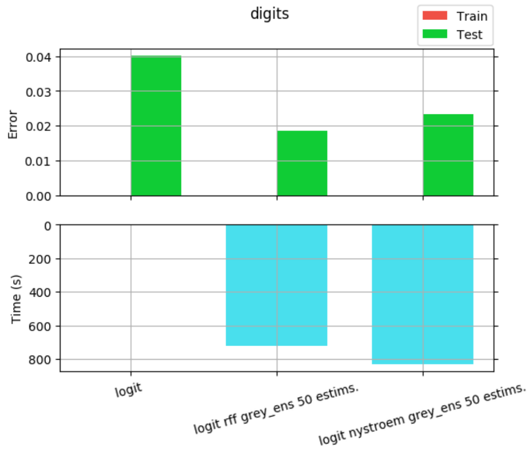
(B) prueba vowel

Appendix E

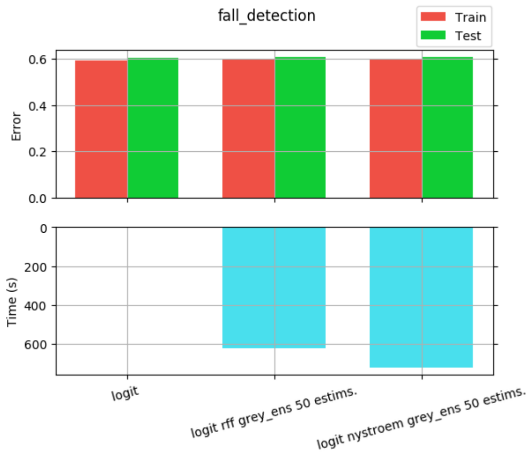
Results of experiment 2.4



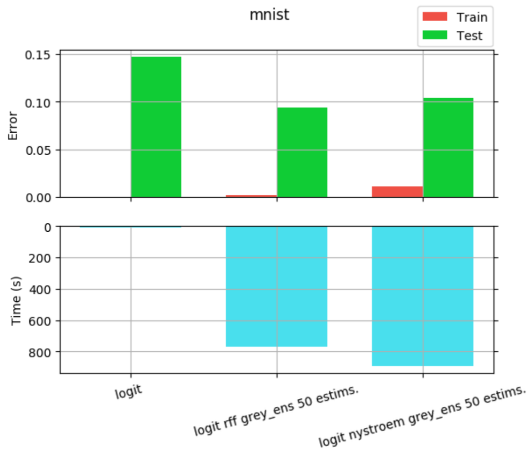
(A) prueba covertype



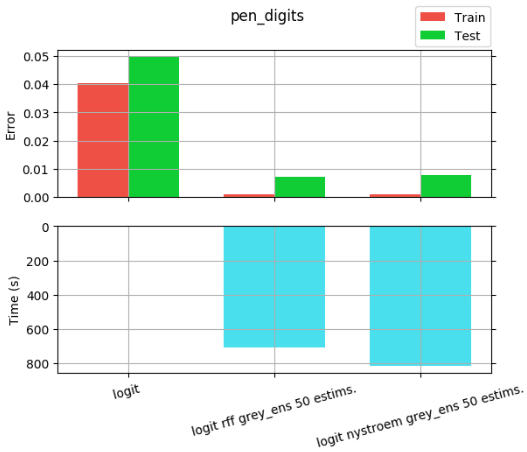
(B) prueba digits



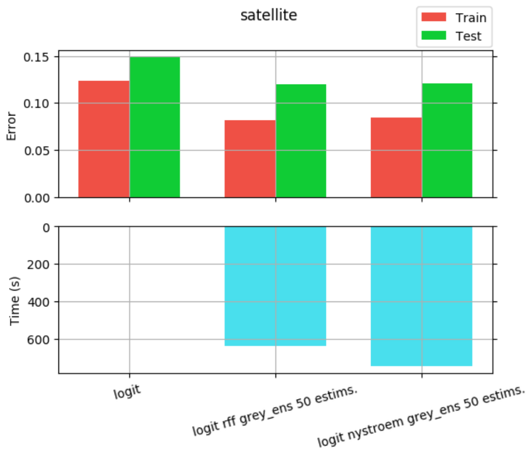
(A) prueba fall-detection



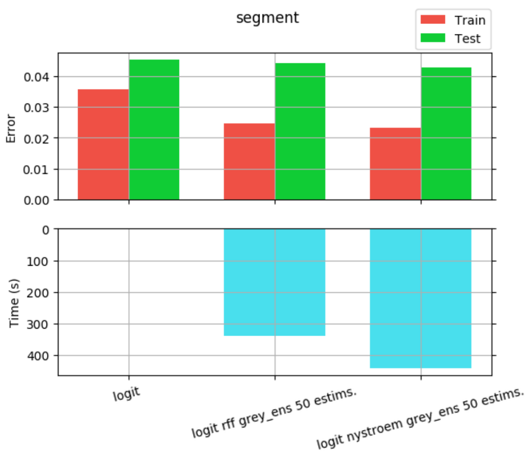
(B) prueba mnist



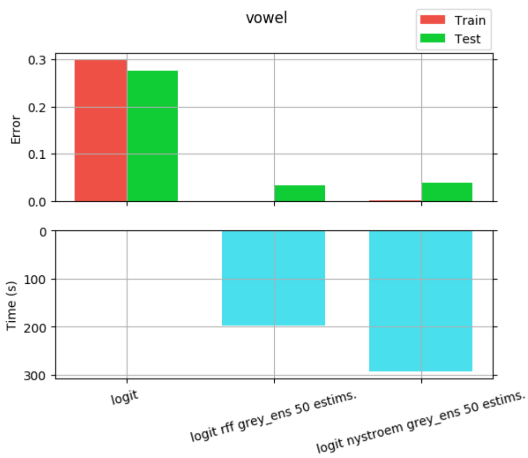
(A) prueba pen-digits



(B) prueba satellite



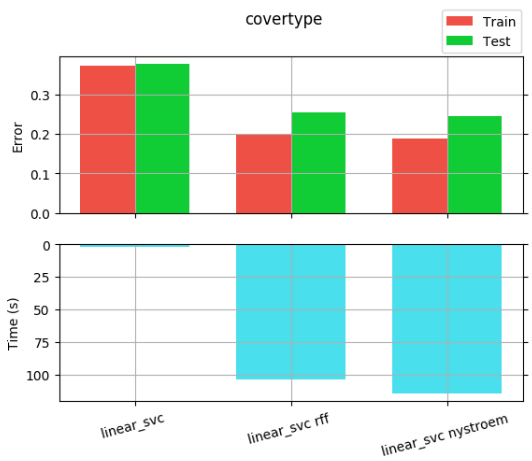
(A) prueba segment



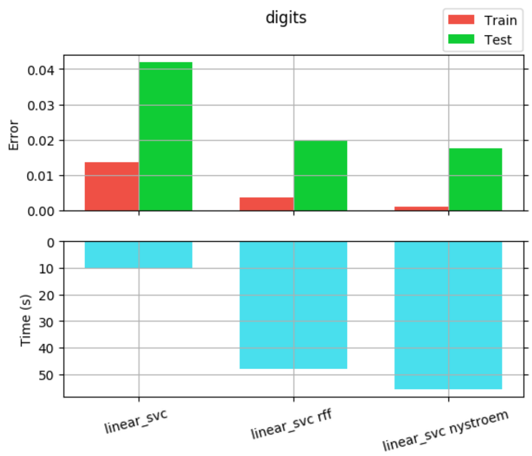
(B) prueba vowel

Appendix F

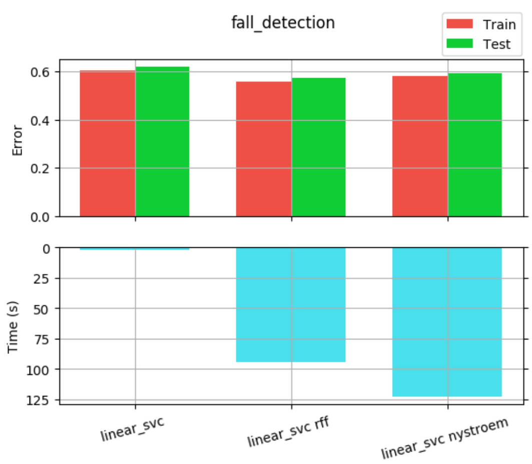
Results of experiment 2.5



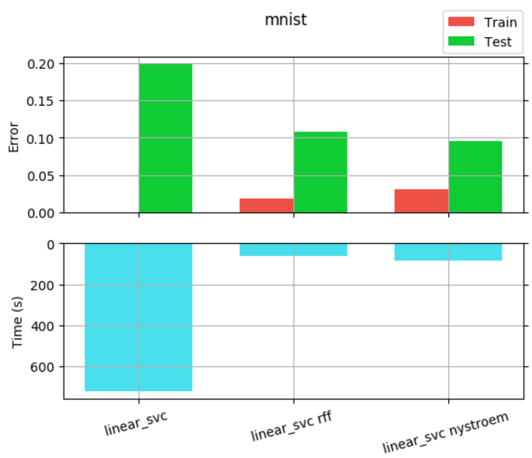
(A) prueba covertype



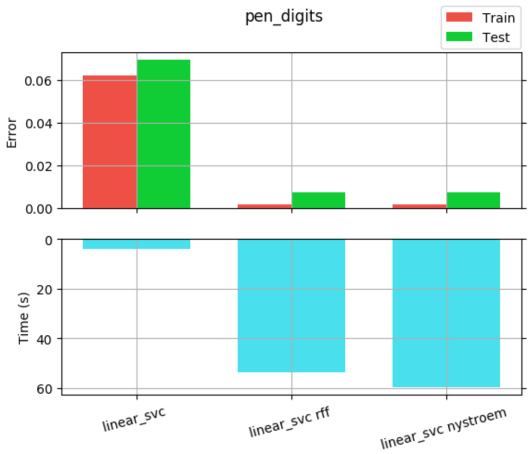
(B) prueba digits



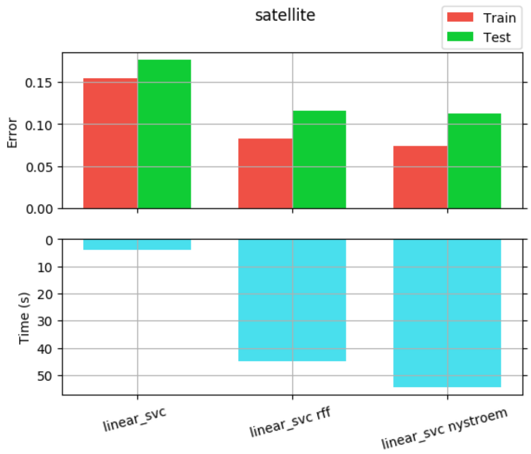
(A) prueba fall-detection



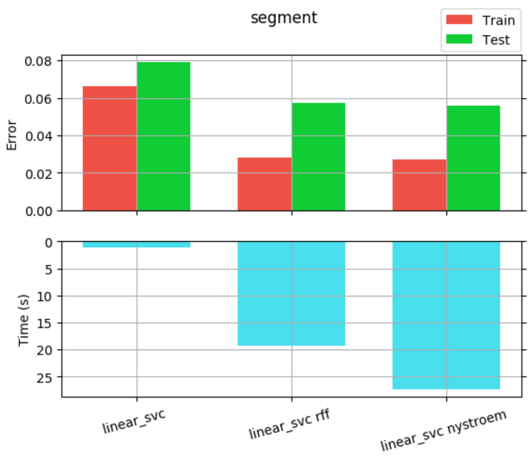
(B) prueba mnist



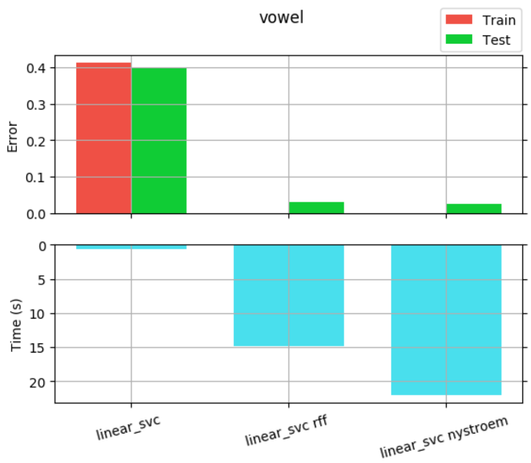
(A) prueba pen-digits



(B) prueba satellite



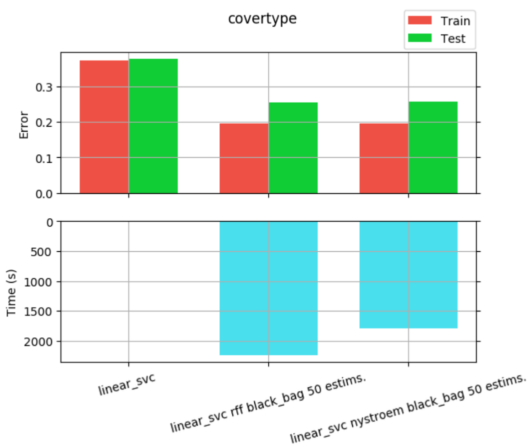
(A) prueba segment



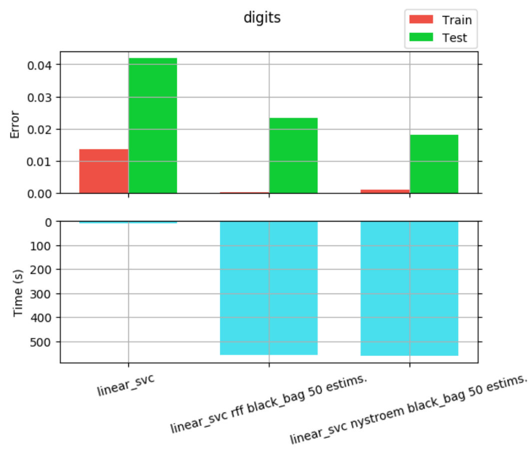
(B) prueba vowel

Appendix G

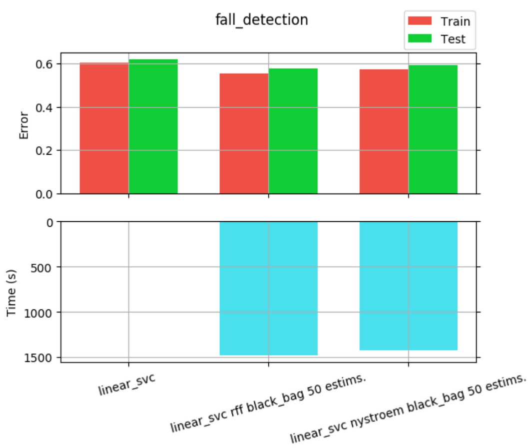
Results of experiment 2.6



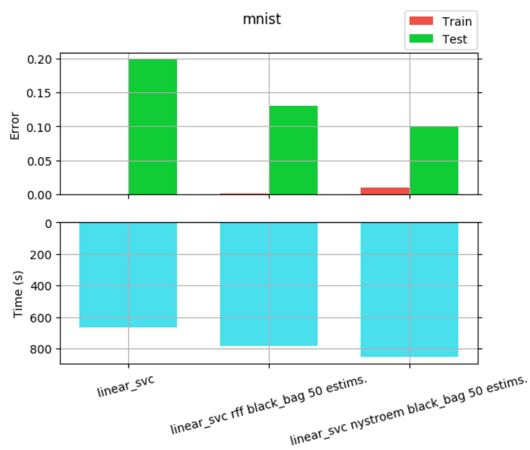
(A) prueba covertype



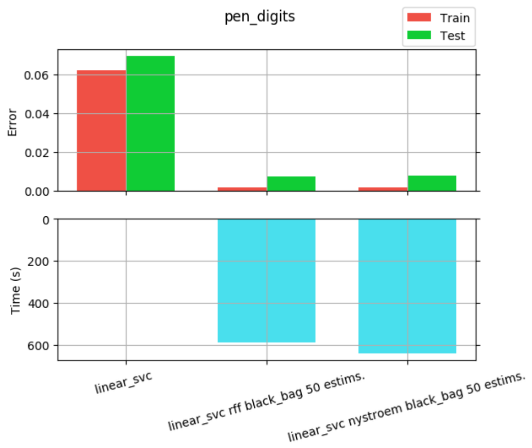
(B) prueba digits



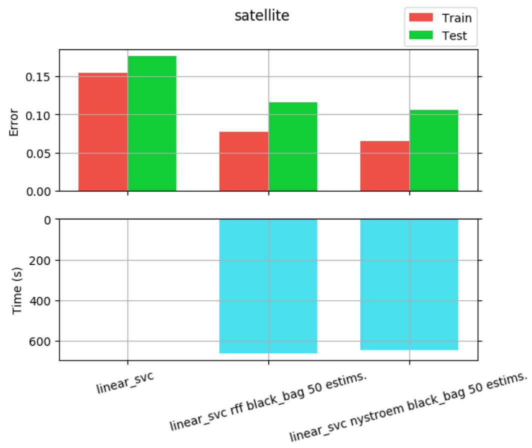
(A) prueba fall-detection



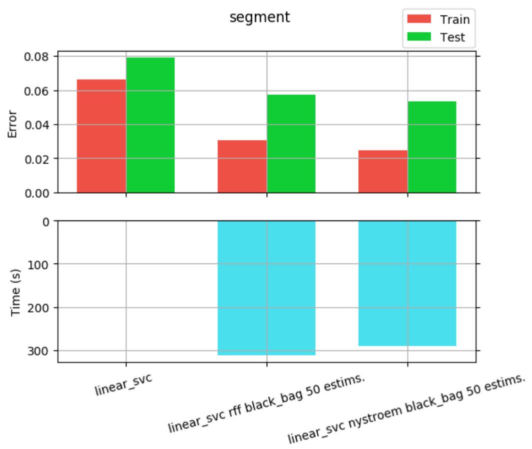
(B) prueba mnist



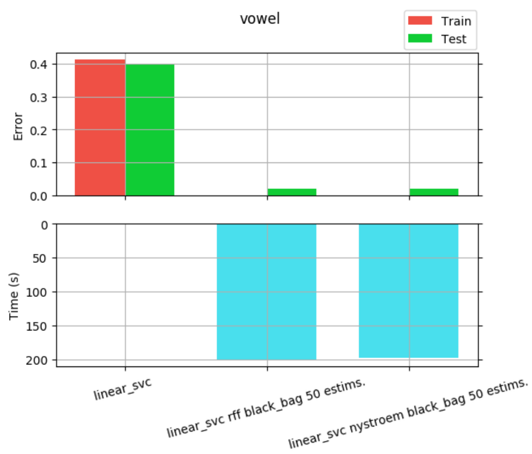
(A) prueba pen-digits



(B) prueba satellite



(A) prueba segment

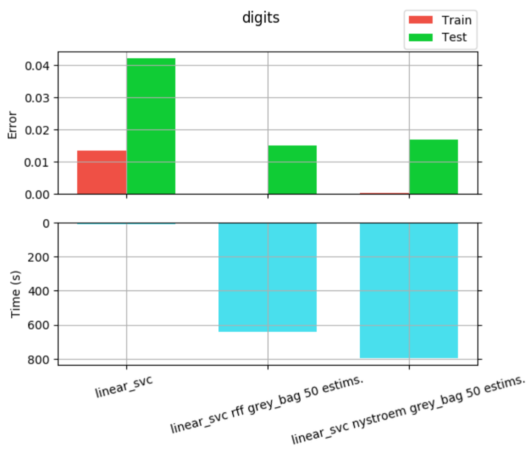


(B) prueba vowel

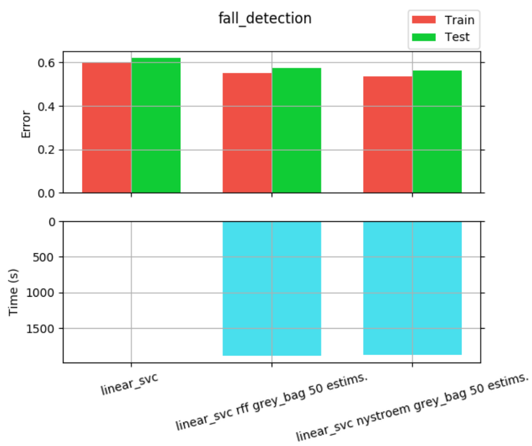
Appendix H

Results of experiment 2.7

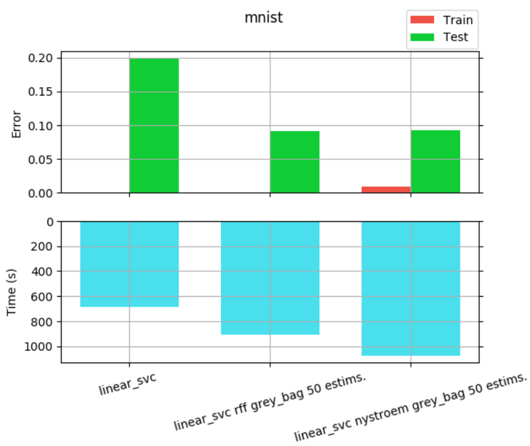
(A) prueba covertype



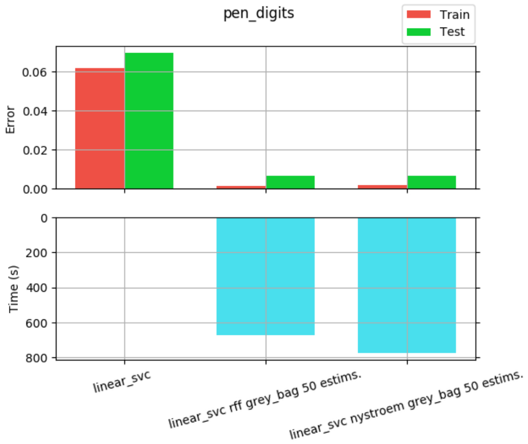
(B) prueba digits



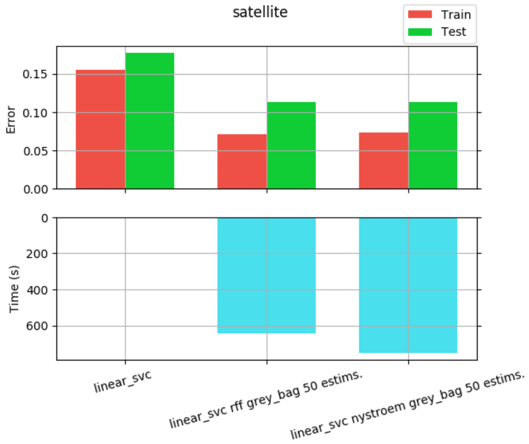
(A) prueba fall-detection



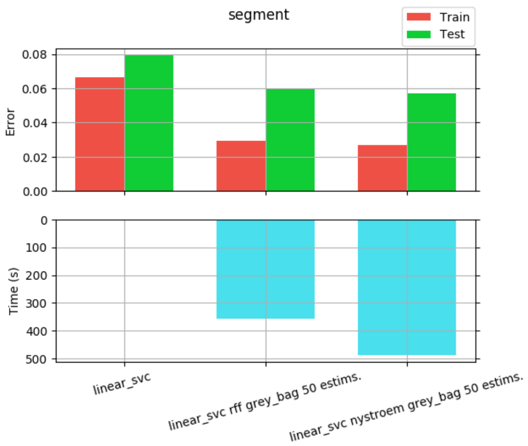
(B) prueba mnist



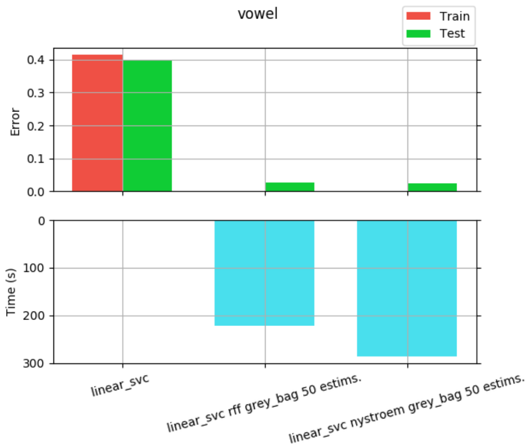
(A) prueba pen-digits



(B) prueba satellite



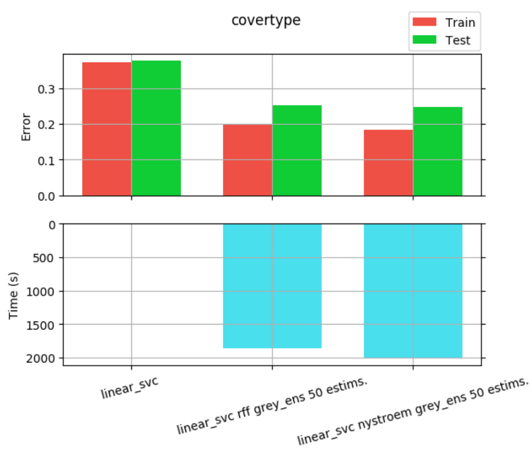
(A) prueba segment



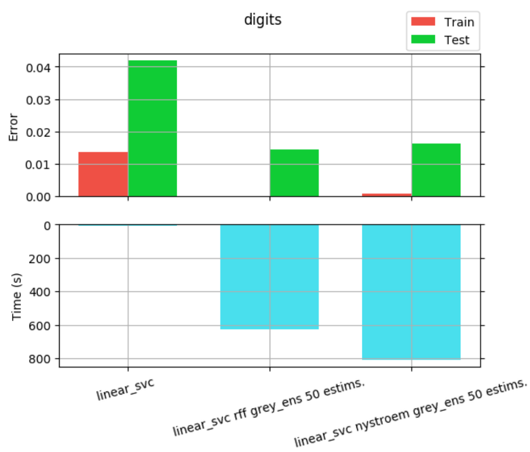
(B) prueba vowel

Appendix I

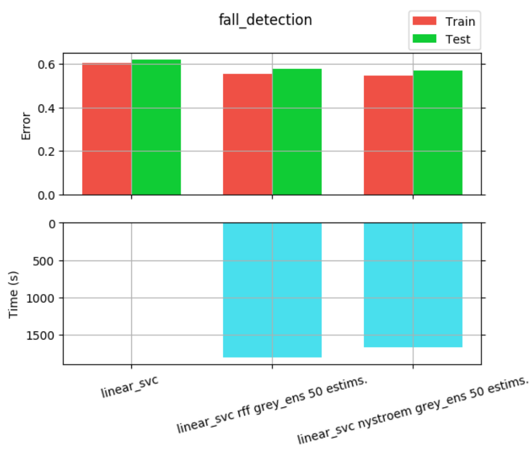
Results of experiment 2.8



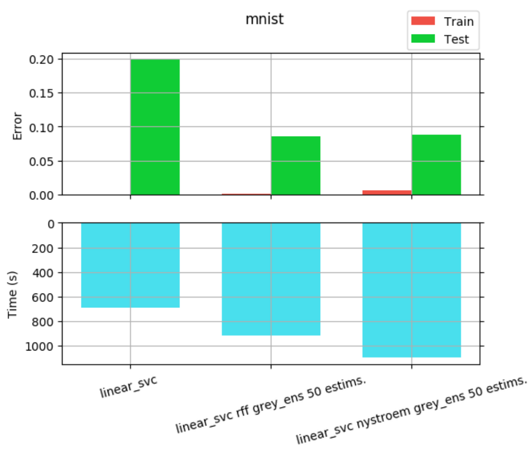
(A) prueba covertype



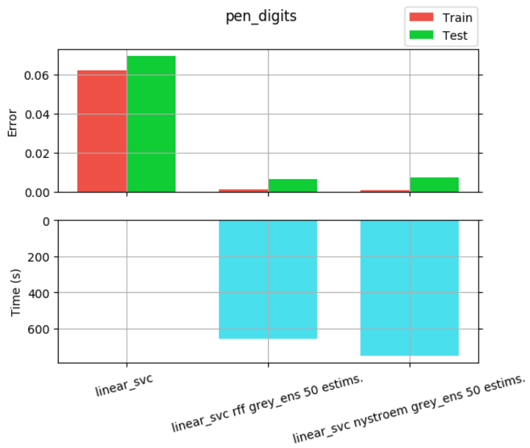
(B) prueba digits



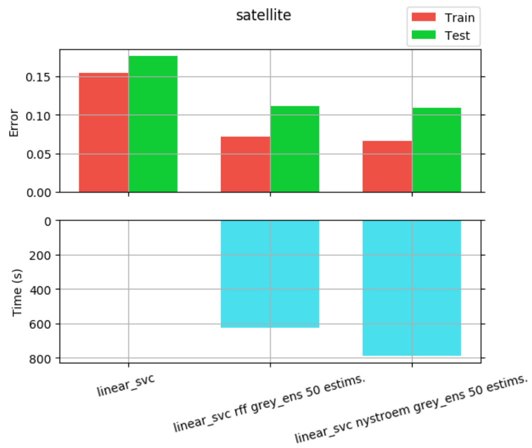
(A) prueba fall-detection



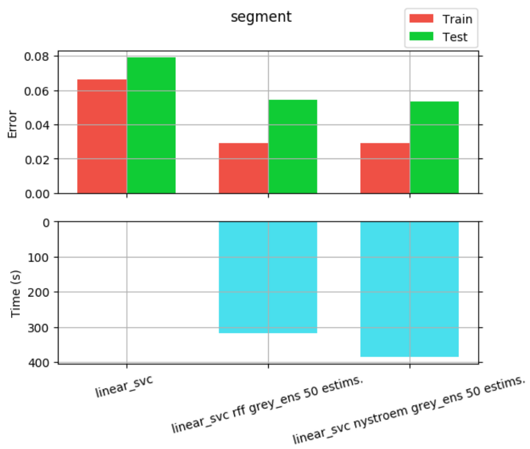
(B) prueba mnist



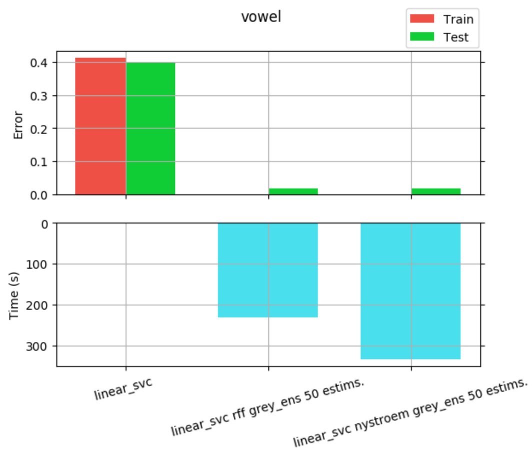
(A) prueba pen-digits



(B) prueba satellite



(A) prueba segment



(B) prueba vowel

Bibliography

Caputo, Barbara et al. (2002). “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.