Using Random Fourier Features with Random Forest

Albert Ribes

Director: Lluís A. Belanche Muñoz

Computer Science

Grau en Enginyeria Informàtica

Computació

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Fecha de defensa

Contents

En los procedimientos que usan liblinear, cambiar el parámetro de *tolerancia*, y volver a poner la cantidad de iteraciones a su cantidad normal.
La gamest que tengo implementada está mal, porque eleva al cuadrado. Hay que cambiarlo

Ideas generales

El guión que me propuso LLuís es:

1. Problema que ataco

2. Por qué es importante
3. Qué propongo en mi TFG
4. Estado del arte en el problema que ataco
5. Nociones generales del tema
    - Machine Learning
    - Árboles
    - Logit
    - RFF
    - Nystroem
    - Bootstrap
    - Boosting
6. El trabajo propiamente dicho (explicar lo que voy a hacer)
7. Experimentos
8. Conclusiones y Trabajo futuro
9. Referencias
10. Apéndices

Parece que en algún paper se ha comprobado como no hay una diferencia significativa entre usar RFF y usar Nystroem
El paper [7] se comprueba como no hay una especial diferencia
En el paper [6] se comprueba como Nystroem es mejor cuando hay un eigen-gap más grande

1 Introduction

1.1 Problem to solve

Todavía no se consigue suficiente precisión con el Machine Learning

Se trata de encontrar un buen trade-off entre conseguir más precisión con el ML y el coste que tiene conseguirla.

1.2 Why is it important

Con precisión más alta se podría aplicar el machine learning en otros campos

Avances en este campo podrían pemitir usar el ML en campos donde ahora no llega, como la medicina, sociedad, economía. Las ventajas del ML son muchísimas, pero todavía no es posible explotarlas todas.

1.3 Project Proposal

Incrementar el accuracy que se puede conseguir con algunos problemas mezclando la técnica del bagging (y quizá del boosting) con el tema los RFF

Actualmente el bagging solo se usa con Decision Tree porque es muy inestable. Con lo que propongo aquí, podría ser factible usarlo con otros algoritmos más estables

Actualmente existe una batería de algoritmos y tecnologías que son muy útiles, pero que se están usando cada una por su cuenta, no se están combinando. Las que yo veo son:

- Los modelos de toda la vida: SVM, Logit, DT, etc.
- Los ensembles junto con el bagging
- Los kernel trick
- Las aproximaciones de los kernel, como RFF y Nystroem

Básicamente, la propuesta es combinar todos estos métodos que ya son muy buenos de por sí para conseguir un mejor trade-off entre la precisión y el tiempo de entrenamiento.

Poner las hipótesis de forma rápida

Los hipótesis son:

1. Se puede diseñar un algoritmo que permita que tenga sentido hacer un ensemble de modelos distintos a DT, como Logit o SVM
2. Se puede aproximar la precisión de una SVM con kernel no lineal con métodos más eficientes, sin perder demasiado la precisión. Esto ya se ha hecho en [12]
3. Las RFF quizá añaden demasiada aleatoriedad, y es necesario reducirla por otro lado
4. Los modelos que no se basan en productos lineales de las entradas no se beneficiarán tanto de las RFF

Chapter Explanation

Poner qué veremos en cada capítulo

En la sección 2 se da una explicación rápida de todos los conceptos que hace falta conocer para entender este trabajo. En la sección 3 se explica lo que se ha hecho en este trabajo. En particular, se explica la novedad que aporta este trabajo, las hipótesis que se han planteado con un poco más de detalle, con qué datasets se ha testeado y los procedimientos y preprocesado que se ha tenido con ellos y finalmente qué experimentos hemos diseñado. En la sección 4 se muestran los resultados de los experimentos y se estudian los mismos para finalmente contrastar las hipótesis que habíamos planteado. En la sección 5 se describen las conclusiones a las que se ha llegado con este proyecto y posibles expansiones que se pueden hacer. Finalmente, en la sección 6 se hace un estudio de la sostenibilidad de este proyecto.

2 Background

Más o menos, cada uno debería ocupar entre media y una página

2.1 Machine Learning

1. Qué es el ML
2. Subcampos que tiene el ML
3. Problemas y dificultades a los que se enfrenta

Los problemas a los que se enfrenta el ML son:

- Muchas veces no se tiene TODA la información necesaria para resolver un problema
- Los datos recogidos tienen un cierto error de precisión
- Intentamos aprender sobre TODA la población basándonos únicamente en una muestra
- Muchos de los problemas que intenta resolver son ambiguos o mal definidos. No siempre está claro que lo que hay en la imagen es un 1 y no un 7.
- Siempre corremos el peligro de sobre-ajustar.
- No tenemos un tiempo de entrenamiento infinito
- La forma de la fórmula implícita no la conocemos. Es decir, quizá estamos usando un polinomio cuando la fórmula implícita tiene logaritmos, y por lo tanto nunca seremos capaces de imitar su comportamiento con un polinomio.

2.2 Classification and Regression

2.3 Review de los principales modelos que existen

2.3.1 Decision Tree

- No se basa en productos escalares
- Es extremadamente rápido
- Es más fácil de interpretar que otros modelos
- Es extremadamente inestable
- Es un poco aleatorio: dos entrenamientos iguales pueden producir árboles distintos (contrastar si esto es solo con los RF)
- Es un modelo no lineal

2.3.2 Logistic Regression

2.3.3 SVM

- Originalmente estaban pensadas para clasificar en 2 clases, pero se han hecho expansiones. Con *one-vs-rest* se puede clasificar con más de dos clases, y también existe un modelo de SVM para hacer regresión.
- Se basa únicamente en el producto escalar de sus entradas
- Por si misma es un modelo lineal, incapaz de separar la mayoría de los datos
- Actualmente es muy poco eficiente usarlas porque su coste es cúbico con la cantidad de entradas.

2.4 Las técnicas ensembling

2.4.1 Bagging

- Inventado por Leo Breiman [4]
- Pretende reducir el sesgo
- Los estimadores se entrenan de forma independiente, se podría hacer en paralelo
- Actualmente casi que solo se usa con el DT, debido a su inestabilidad

2.4.2 Boosting

- Cita [11]
- Adaboost (adaptive boosting)
- El siguiente estimador es más probable que contenga los elementos que no se han predicho bien en el anterior
- Se trata de ir modificando los pesos que tiene cada una de las instancias
- El entrenamiento de los modelos es secuencial, a diferencia del bagging
- Enterarme de quien lo inventó, y para qué ámbitos es útil
- Está basado en la pregunta que planteó Kearns and Valiant de: "Can a set of weak learners create a single strong learner?"

2.5 El bootstrap

- En bagging es bueno que los estimadores estén poco relacionados entre ellos
- Idealmente, usaríamos un dataset distinto para cada uno de los estimadores, pero eso no siempre es posible
- Una alternativa es usar un resampling con repetición sobre cada uno de los estimadores para tener datasets un poco distintos entre ellos.
- Enterarme de la cantidad de elementos distintos que se espera que queden en el subconjunto, y quizá hablar de la cantidad de aleatoriedad
- 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 $1 - e^{-\frac{r}{n}}$

2.6 Las funciones kernel

A kernel is a function that equals to the inner product of inputs mapped into some Hilbert space, i.e:

$$\kappa(x, y) = \langle \phi(x), \phi(y) \rangle \tag{1}$$

As long as the learning technique relies only on the linear product of the inputs, the underlying mapping $\phi(\cdot)$ doest not need to be explicitly calculated and can, in fact, be unknown.
[12]

Las SVM encuentran un híper-plano que separa las instancias de un problema determinado en dos subconjuntos del espacio, y en el que cada subconjunto se identifica con las clases que se quieren discriminar. Este híper-plano busca maximizar la distancia mínima entre él mismo y las instancias (los vectores) de cada una de las clases. Para hacerlo, convierte el problema en uno de optimización.

Tenemos un conjunto de datos $D = \{\boldsymbol{\chi}, \boldsymbol{y}\}$, donde $\boldsymbol{\chi} = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y} = \{-1, +1\}^n$

Se requiere encontrar $\boldsymbol{\alpha} \in \mathbb{R}^n$ que maximice:

$$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{2}$$

Pero este procedimiento solamente es efectivo si se da el caso que todas las instancias de $\boldsymbol{\chi}$ son linealmente separables por un híper-plano en cada una de las dos clases.

Este no siempre es el caso, y por eso se suele realizar una transformación de los datos, que los lleven de un subespacio a otro, que normalmente tiene más dimensiones que el original y que se espera que sí permita separar linealmente los datos.

Entonces, si se define una función de transformación de espacio $z(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^D$, la función a optimizar sería:

$$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j z(\boldsymbol{x}_i)^T z(\boldsymbol{x}_j) \tag{3}$$

Estos cálculos únicamente trabajan con el producto escalar de los vectores, nunca con ellos directamente. Es por eso que si existiera una función:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = z(\boldsymbol{x})^T z(\boldsymbol{y}) \tag{4}$$

Se podría optimizar la función

$$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5}$$

sin tener jamás que calcular el producto escalar de los vectores. De hecho, la dimensionalidad del nuevo espacio vectorial podría ser infinita sin ningún problema. Lo único necesario sería que en ese nuevo espacio, que no tenemos por qué conocer, los vectores fueran linealmente separables.

Pues estas funcions κ existen, y se suelen llamar funciones kernel. Se usan especialmente con las SVM, pero se podrían usar en cualquier otro campo.

Algunas de las que existen son el kernel lineal ($\kappa(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{y} + c$), el polinómico ($\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T\boldsymbol{y}+c)^d$), el gausiano o RBF ($\kappa(\boldsymbol{x}, \boldsymbol{y}) = exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2})$), etc.

La SVM simple, la que no permite que nadie viole el margen y que no converge cuando los datos no son linealmente separables plantea este problema.
Encontrar los valores $\boldsymbol{\alpha}$ que maximizan:

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \tag{6}$$

Sujeto a que:

$$\alpha_i \geq 0; \forall i \tag{7}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{8}$$

$$\tag{9}$$

Y la solución se encontraría con:

$$w = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i \tag{10}$$

La b se podría encontrar resolviendo la equación

$$\alpha_i \left(\boldsymbol{y}_i \left(\boldsymbol{w}\boldsymbol{x}_i + b \right) - 1 \right) = 0; \forall i | \alpha_i \neq 0 \tag{11}$$

Se podría resolver con una sola i, pero es más estable hacer la media de todas.
Entonces, cuando tenemos un punto nuevo $\boldsymbol{x}$ y queremos saber a qué clase pertenece, calculamos $sgn(\boldsymbol{w}\boldsymbol{x} + b)$. Si es positivo es de la clase $+$, y si es negativo es de la case $-$. Se supone que nunca estará entre $-1 < x < 1$.
Si queremos permitir que haya vectores que cometan un cierto error podemos hacerlo añadiendo las variables ζ.
Ahora la función objetivo para minimizar ya no es $\frac{1}{2}\|\boldsymbol{w}\|^2$, sino que es:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_i \zeta_i \tag{12}$$

Y entonces el objetivo es maximizar

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \tag{13}$$

Sujeto a:

$$0 \leq \alpha_i \leq C; \forall i \tag{14}$$

$$\sum_i \alpha_i y_i = 0 \tag{15}$$

La solución es la misma que antes. La única diferencia es que ahora hay una cota superior para $\boldsymbol{\alpha}$.

¿Pero qué hacemos cuando la función de decisión no es una función lineal de los datos? Imagina que primero mapeáramos los datos a un (posiblemente de infinitas dimensiones) espacio euclideo $\mathcal{H}$, que vamos a llamar $\phi : \mathbb{R}^d \mapsto \mathcal{H}$. Puesto que los datos solo aparecen en forma de productos vectoriales, podríamos tener una función $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y})$ únicamente tendríamos que usar la función κ, y nunca tendríamos que conocer explícitamente la función ϕ. Un ejemplo es el kernel rbf, que es $\kappa(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}}$.

Ahora para hacer una predición ya no podemos usar el vector $\boldsymbol{w}$, porque está en otro espacio. Pero todavía podemos usar la fórmula $sign\left(\sum_{i=1}^{l} \alpha_i y_i \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x})\right) = sign\left(\sum_{i=1}^{l} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x})\right)$

2.6.1 El kernel RBF

El kernel RBF es una familia de funciones kernel. El nombre viene de *Radial Basis Function*. Esta familia de funciones tiene un parámetro σ, y son estas:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}; \sigma) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|}{2\sigma^2}} \tag{16}$$

A veces también lo expresan con una gamma (γ), con la equivalencia $\gamma = \frac{1}{2\sigma^2}$:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}; \gamma) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|} \tag{17}$$

Tiene una cierta noción de similaridad entre los dos vectores: cuanto más distintos son (cuanto mayor es la norma de su diferencia) más se aproxima a 0, y si son iguales es 1.

Se sabe que el feature space de este kernel tiene dimensionalidad infinita, y es de los kernel más utilizados.

(Me gustaría enterarme si siempre siempre siempre es dimensionalidad infinita, con cualquier valor de gamma).

Un valor de σ muy pequeño (muy cercano a 0) produce más sobre-ajuste, mientras que un valor más grande lo disminuye.

- Su fórmula es ...
- La equivalencia entre la σ y la γ
- La noción de similitud que tiene
- $\mathcal{H}$es de dimensionalidad infinita
- Permite ajustarse totalmente a los datos, tuneando el parámetro.

- σ pequeño, más sobre ajuste.

2.7 Las Random Fourier Features

$$\kappa(\lambda) \approx \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle \tag{18}$$

$$\omega_i \sim \kappa(\omega) \tag{19}$$

$$\phi(x) = \frac{1}{\sqrt{D}} \left[e^{-i\omega_1^\intercal x}, \ldots, e^{-i\omega_D^\intercal x} \right]^\intercal \tag{20}$$

Es una ténica que permite aproximar el feature space de un kernel. Sea κ un kernel, tal que

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = z(\boldsymbol{x})^T z(\boldsymbol{y}) \tag{21}$$

(Creo que no permite aproximar todos los kernel, solo los que cumplen una condición)

Donde $z(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^D$. En el caso particular del kernel RBF, $z(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^\infty$

Las Random Fourier Features permiten generar una función $f(\boldsymbol{x})$ que aproxima $z(\boldsymbol{x})$ con una dimensionalidad arbitraria, de manera que $f(\boldsymbol{x})f(\boldsymbol{y}) \approx \kappa(\boldsymbol{x}, \boldsymbol{y})$

Como el subespacio de $z(\boldsymbol{x})$ es de dimensionalidad infinita para algunos kernels como el RBF, $f(\boldsymbol{x})$ coje un subconjunto aleatorio de todas esas dimensiones, según la cantidad que se haya especificado. Esto permite generar varias imágenes aleatorias de distintas aproximaciones $f(\boldsymbol{x})$ para un mismo vector $\boldsymbol{x}$, y esto mismo es lo que se explota en este trabajo para generar aleatoriedad en los datos

2.8 Nystroem

Sobretodo en el ámbito de las SVM se utiliza el concepto de *Gramm matrix* de un kernel entrenar un modelo. Sea $\boldsymbol{\chi} = \{\boldsymbol{x}_1, \ldots \boldsymbol{x}_n\}$ un conjunto de datos y $\kappa(\boldsymbol{x}, \boldsymbol{y})$ un función kernel. La matriz de Gram G es de tamaño $n \times n$, y $G_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$

El cálculo de esta matriz es muy costoso en tiempo y en espacio, y por lo tanto no es factible para la mayoría de problemas de Machine Learning, que requieren grandes cantidades de datos.

El método Nystroem consiste en aproximar esta matriz de Gram con un subconjunto aleatorio de los datos que sea adecuado sin afectar negativamente la precisión de la solución

2.9 Cross-validation

3 Project Development

3.1 General Idea

La idea general un poco desarroyada

Hemos visto que podemos sacar una aproximación aleatoria de la función implícita de un kernel. Esto tiene básicamente 2 ventajas:

- Podemos transformar los datos directamente
- Podemos producir pequeñas variaciones de un mismo dataset, todas ellas válidas.

Black Bag Datos → RFF → Bootstrap → Modelos

Grey Bag

Black Ensemble

Grey Ensemble

La figura 1 muestra los tipos de caja

Las funciones kernel son funciones que se pueden expresar de la forma:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) \tag{22}$$

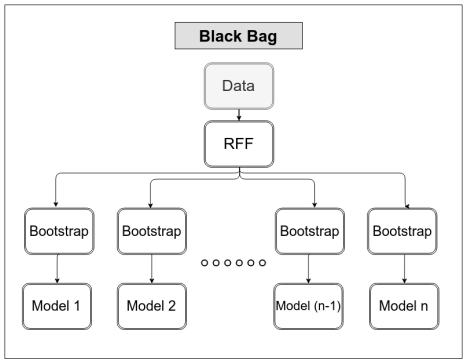
Es decir, como producto escalar de una función de sus parámetros. Un kernel muy popular es el RBF (*Radial Basis Function*) gausiano, que es este:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}; \sigma) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|}{2\sigma^2}} \tag{23}$$

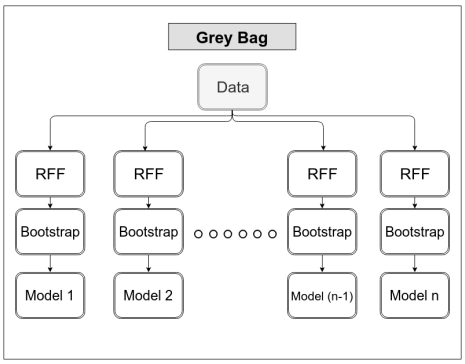
La función implícita ϕ ($\mathcal{L} \mapsto \mathcal{H}$) de este kernel tiene una dimensionalidad infinita, y se sabe que para cualquier conjunto de datos se puede encontrar un kernel RBF κ tal que su función implícita ϕ es capaz de separarlos mediante un híper-plano.

A pesar de que la función ϕ tiene dimensionalidad infinita ($\mathcal{H} \equiv \mathbb{R}^\infty$) es posible extraer una aproximación aleatoria de la misma con una precisión arbitraria, mediante el uso de *Random Fourier Features* [10] (RFF). Otra técnica que también se puede utilizar es el método Nystroem. Con estos métodos se puede extraer $\psi(\boldsymbol{x}) \approx \phi(\boldsymbol{x})$ y usarlos para lo que haga falta.

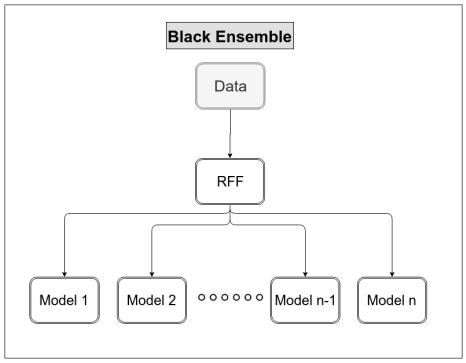
La extracción de estas aproximaciones se ha usado con anterioridad junto con métodos de redes neuronales, y ha mostrado muy buenos resultados. En este nosotros tratamos de usarlas con otros modelos. En particular, hemos estudiado los modelos de Decision Tree, Logit y LinearSVC, en combinación con varios tipos de ensemble.



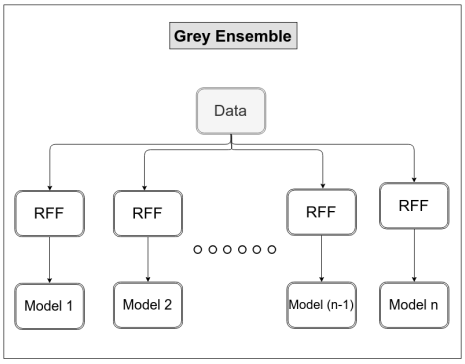
(a) Black Bag



(b) Grey Bag



(c) Black Ensemble: every model is trained with the same data



(d) Grey Ensemble

Figure 1: Box Models

El uso de ensembles está muy extendido junto con los Decision Tree. Esto se debe a que éste es un modelo muy inestable, y una pequeña alteración en los datos puede producir resultados muy distintos. Estas condiciones son idoneas para hacer un comité de Decision Trees, entrenarlos con datos ligeramente distintos y elegir la solución qué más árboles hayan predicho.

Pero este procedimiento no tiene ningún sentido hacerlo con modelos que no son inestables. Si los modelos no son inestables, la mayoría de los estimadores responderán la misma solución, y no servirá para nada haber entrenado tantos modelos distintos. Es como si en un comité de expertos todos ellos opinaran igual: para eso no necesitamos todo un comité, con un solo experto nos habría bastado.

La técnica de *bagging* utiliza el *bootstrap* para generar datasets distintos para cada uno de los estimadores. Consiste en hacer un remuestreo de los datos con repetición para cada uno de los estimadores. Esta diferenciación que se produce es suficiente para los Decision Tree, pero es demasiado leve con los métodos más

estables, como Logit y LinearSVC.

Pero los RFF y Nystroem abren una nueva puerta. Puesto que son aproximaciones aleatorias de un conjunto infinito, podemos sacar tantos mapeos distintos como queramos de los datos originales, y por lo tanto podemos diferenciar todavía más los datasets generados para cada uno de los estimadores.

Además de todo esto, hay una ventaja adicional: entrenar una *Support Vector Machine* (SVM) con kernel lineal es más barato que entrenar una no lineal, por ejemplo una que use RBF. Si usamos una SVM lineal, pero en vez de entrenarla con los datos originales la entrenamos con los datos $\psi(\boldsymbol{x})$, tenemos un coste similar al de entrenar una SVM lineal pero con una precisión equiparable a una RBF-SVM. Esto ya se ha hecho antes.

Existen varias formas de combinar las RFF con los métodos ensembles. Básicamente, hay dos parámetros que podemos elegir: qué tipo de ensemble usar y en qué momento usar las RFF.

Cuando se combina un ensemble con los RFF, básicamente hay dos momentos en los que se puede usar el mapeo. Un momento es nada más empezar, antes de que el ensemble haya visto los datos, y el ensemble trabaja normalmente, solo que en vez de recibir los datos originales recibe un mapeo de los mismos. Este método se abstrae completamente de lo que hace el ensemble, y lo trata como una caja negra. *Black Box*.

El otro método consiste en usar el RFF, no nada más empezar y el mismo para todos los estimadores, sino justo antes del estimador: se hace un mapeo nuevo para cada uno de los estimadores. Este método ya se mete dentro de lo que es un ensemble, y por tanto diremos que es de caja blanca (*White Box*).

Pero se sabe que se obtienen mejores resultados cuando hay bastante diversidad entre los estimadores del ensemble. Se nos presentan ahora dos formas de crear diversidad en el ensemble. Una de ellas es la forma clásica, mediante el bootstrap, que ha mostrado muy buenos resultados con el *Decision Tree*. Pero ahora podemos usar también la aleatoriedad de los RFF para generar esa diversidad. Entonces tenemos dos opciones: usar los RFF ellos solos o usarlos junto con el Bootstrap. A usarlos junto con el bootstrap le llamaremos un *Bagging*, mientras que si no usamos bootstrap le llamaemos un *Ensemble*.

Tenemos entonces varias combinaciones entre manos:

Black Bag Black Box model con Bagging. Primero se hace un mapeo de los datos y después se hace un bootstrap con ellos para cada uno de los estimadores. Si los estimadores son *Decision Tree* es los mismo que un *Random Forest*, pero no con los datos originales, sino con el mapeo.

White Bag White Box model con Bagging. Primero se hace un bootstrap de los datos, para cada uno de los modelos, y después para cada uno de ellos se hace un mapeo de los datos.

White Ensemble White Box model sin baging. Se hace un mapeo para cada uno de los estimadores, todos ellos usando todos los datos originales.

El *Black Ensemble* no tiene ningún sentido hacerlo, porque en ese caso todos los estimadores recibirían exactamente los mismos datos, y por lo tanto todos producirían exactamente los mismos resutados, a no ser que tuvieran algún tipo de aleatoriedad, como los Decision Tree. A pesar de que tengamos ese caso particular con los DT, no lo vamos a tratar.

Y luego, por supuesto, haremos pruebas con un modelo simple usando los RFF, sin usar ningún tipo de ensemble.

3.1.1 State of the art con las RFF

Se ha trabajado muy poco con ellas. Básicamente, solo se le ha dado dos usos:

Stacked kernel network [15] Usarlas junto con una red neuronal para tener más niveles de aprendizaje no lineal.

RFF with SVM [12] Usar una SVM sin kernel con los datos mapeados usando RFF

3.2 Hyperparameters

Existen los siguientes parámetros:

- En DT, min-impurity-decrease
- En SVM, la C
- EN RFF y en Nystroem, la gamma
- en RFF y en Nystroem, la cantidad de features
- En los ensembles, la cantidad de estimadores

Entonces, hemos usado el siguiente procedimiento:

- La cantidad de features la hemos fijado a 500, aquí ya se estanca
- La cantidad de estimadores la hemos fijado a 50
- En modelos simples, se estima el parámetro por cross-validation
- En modelos simples con RFF, se estima el parámetro por cross-validation y se pone una gamam que sobre-ajuste
- En modelos con ensemble, hemos fijado híper-parámetros que sobre-ajusten y se estima la gamma por cross-validation
- En SVM con RBF, se usa la gamma de gamest, y se estima C por cross-validation

Existen los siguiente híper-parámetros:

Decision Tree

El algoritmo de Python de DT tiene muchísimos híperparámetros: la máxima profundidad, el mínimo de hojas, mínimo de instancias para hacer split, mínimo decrecimiento de la impureza de un nodo para hacer split, etc.

Trabajar con todos ellos no era viable, de modo que únicamente hemos considerado el *min_impurity_decrease*. Todos los demás los hemos dejado que sobreajusten. De esta manera el modelo es más parecido al de R, y facilita hacer comparaciones

Logit

Se puede hacer Logit con regularización de w o sin hacerla, pero la implementación en Python que tenía disponible obligaba a hacerla. Como decidimos que no queríamos hacer regularización, hemos puesto un valor que hace que la regularización sea mínima.

Support Vector Machine

Tiene un parámetro C para regular la importancia que se le da a los vectores que quedan fuera del margen. Es el que hemos considerado.

También hay otros híperparámetros:

Cantidad de features Se puede sacar una cantidad arbitraria de features, cuanto mayor mejor se aproxima al kernel. Después de muchas pruebas, vimos que con 500 features ya no se puede mejorar mucho más, de modo que hemos hecho todas las pruebas usando exactamente 500 features.

Gammas Tanto el RFF como el Nystroem tienen un parámetro *gamma*, que es el del kernel que quieren simular, el RBF.

Cantidad de estimadores de los ensembles Se sabe que incrementear la cantidad de estimadores no empeora la precisión del modelo, pero coger un número demasiado alto incrementa el tiempo de entrenamiento. Hemos fijado este parámeto a 500.

El procedimiento que hemos usado para encontrar los híperparámetros para los experimentos ha sido el siguiente:

Hemos usado crossvalidation, pero hemos intentado que éste fuera solo de un híperparámetro para no incrementear demasiado el coste. Por lo tanto, hemos tenido que hacer algunas simplificaciones.

En los experimentos que implican los modelos simples, sin ningún añadido, se ha hecho crossvalidation con el único parámetro que tienen: DT el min_impurity _decrease, Logit no tiene ninguno (pues hemos forzado la regulación al mínimo) y SVM tiene la C. Hemos hecho crossvalidation con esos.

Cuando usamos algún sampler, también tenemos una gamma que encontrar. En esos casos, puede ser que estemos usando un modelo simple, o un ensemble

de modelos. Cuando se trate de un modelo simple usaremos una gamma que sobreajuste, mientras que haremos crossvalidation con sus híper-parámetros.

Cuando se hace un ensemble, hay que dejar que cada uno de los estimadores sobreajusten. Para ello usaremos unos híperparámetros que sobreajusten, mientras que haremos crossvalidation con la gamma.

3.3 Hypothesis

Poner la hipótesis 2 veces: la primera con muy poco detalle, sin palabras técnicas, en la introducción. Indicando que mas tarde se van a precisar

La segunda será más técnica.

1. Podemos conseguir una precisión similar a SVM con RBF con un coste mucho más pequeño, aproximando el RBF con la RFF
2. Podemos hacer que tenga sentido hacer un ensemble de otros modelos (como Logit o SVM) si usamos RFF. Podemos hacer que un ensemble de estos modelos sea mejor que el modelo solo
3. RFF + Bootstrap puede ser demasiada aleatoriedad, y perjudicará a algunos modelos.
4. Los modelos que no se basen en el producto escalar de las entradas no se van a beneficiar tanto de usar las RFF.

- Con las RFF podemos mejorar la precisión de un Logit o de una SVM haciendo un ensemble con ellos.
- Es posible conseguir un rendimiento parecido al de una RBF-SVM con un coste mucho menor usando las RFF
- Bootstrap + RFF puede generar demasiada aleatoriedad en algunos problemas, y perjudicar la precisión del algoritmo
- Por la naturaleza misma del DT, no se beneficiará demasiado de usar los RFF. Estudiar si un método que no se basa en productos escalares se puede beneficiar de usar Random Fourier Features. No hablar en particular del dt, sino de los que no se basan en producto escalar.
- Tenemos tres tipos de ensembles: Black Bag, Grey Bag y White Bag. El White bag será el mejor.

3.3.1 Planteamiento de los experimentos

Según las hipótesis que he planteado, únicamente hace falta medir el tiempo para la hipótesis 1. ¿Vale la pena medir tiempo con lo otro? ¿Aunque sea solo para comentarlo?

Para contrastar hipótesis 1

(Emular RBF-SVM)

1. Hacer una SVM con el RBF original y otra SVM con las RFF. Ver la precisión que consiguen con cada dataset y también el tiempo que han tardado en llegar. Las pruebas también se pueden hacer con Nystroem

Para contrastar hipótesis 2

(Ensembles con otros modelos) Primero hay que saber cuál de los 3 tipos de ensembles es el mejor, para poder compararlos con el normal
Ver primero si a un solo estimador le beneficia, y luego ver si con ensemble es mejor

1. Un Logit normal contra un logit con RFF
2. Un Logit normal contra un logit con RFF Black Bag
3. Un Logit normal contra un logit con RFF Grey Bag
4. Un Logit normal contra un logit con RFF Grey Ensemble

---

5. Un SVM lineal normal contra un SVM lineal con RFF
6. Un SVM lineal normal contra un SVM lineal con RFF Black Bag
7. Un SVM lineal normal contra un SVM lineal con RFF Grey Bag
8. Un SVM lineal normal contra un SVM lineal con RFF Grey Ensemble

Para contrastar hipótesis 3

(RFF + Bootstrap malo)

1. Un Logit con RFF Grey Bag contra un Logit con RFF Grey Bag
2. Un Logit con RFF Black Ensemble con un solo estimador contra un Logit con RFF Grey Ensemble con un solo estimador

---

3. Un SVM lineal con RFF Grey Bag contra un SVM lineal con RFF Grey Ensemble
4. Un SVM lineal con RFF Black Bag con un solo estimador contra un Logit con RFF Black Ensemble con un solo estimador

Para contrastar hipótesis 4

RFF + Modelos no dot product malo En el caso del DT quizá sí que tiene sentido el Black Ensemble, porque tiene una parte aleatoria.

1. Un DT normal contra un DT con RFF
2. Un RF normal contra un DT con RFF Black Bag
3. Un RF normal contra un DT con RFF Black Ensemble
4. Un RF normal contra un DT con RFF Grey Bag
5. Un RF normal contra un DT con RFF Grey Ensemble

3.4 Datasets

- 8 datasets distintos
- Están normalizados
- Solo trabajo con variables numéricas. Además, los kernels con categóricas no tienen mucho sentido
- Las cosas particulares que he hecho con los datasets:
  - Mezclar datos de train y test para luego hacer mi propia separación
  - Cuando había poca presencia de una clase, hacer resampling con la misma cantidad
  - No trabajar cosas como el skewness, y outliers
  - Eliminar columnas en las que todo eran 0
  - Reducir el conjunto de instancias

He enfocado el trabajo únicamente con problemas de clasificación. He hecho pruebas con 8 datasets distintos.

Todos ellos los he normalizado a media 0 y varianza 1, y he usado dos tercios para train y un tercio para test.

Por limitaciones de las implementaciones de los algoritmos que tengo, no podía trabajar con variables categóricas, de modo que algunas de ellas he tenido que convertirlas a float.

Algunos datasets me los daban separados en 2 subconjuntos, uno para train y el otro para test. Yo los he mezclado, y ya si eso he hecho la separación, aleatoria, por mi cuenta más tarde.

En algunos casos me daban muchísimas instancias, y yo no necesitaba tantas, y por lo tanto he cogido un subconjunto

En algunos casos el dataset no estaba bien balanceado, había muchas instancias de un tipo y pocas de otro. Yo he hecho un subconjunto para cada clase, todos con la misma cantidad de instancias. Esto lo he hecho porque el objetivo de este trabajo no tiene nada que ver con clases mal balanceadas.

3.4.1 Pen Digits

[See 1]

Distinguir entre los 10 dígitos (0-9) de un conjunto de imágenes. El dataset se ha generado cogiendo las coordenadas x e y del trazo hecho por una persona para dibujar ese número e interpolando 8 puntos normalmente espaciados en todo el trazo del dibujo.

La tabla con la información es 1

3.4.2 Covertype

[See 3]

Identificar el tipo de terreno usando información como la elevation, el slope, la horizontal distance to nearest surface water features, etc.

En el dataset original había un atributo con 40 columnas binarias que identificaba el tipo de tierra, con la Soil Type Designation. Estas 40 columnas las he convertido a una sola variable, con números del 1 al 40.

La tabla con la información es 2

3.4.3 Satellite

[See 13]

La página de este dataset dice que son 7 clases, pero una de ellas no tiene ninguna presencia, y por lo tanto yo no la he contado para nada.

Cada instancia es una parcela de 3×3 pixels y para cada pixel nos dan 4 números, cada uno de ellos es el color que se ha captado con 4 different spectral bands. Por eso son 36: $9 \times 4 = 36$.

Lo que se predice es el tipo de terreno que contenía ese pixel, como plantación de algodón,

La tabla con la información es 3

3.4.4 Vowel

[See 5]

Me daban los datos separados por quien había dicho cada vocal, y yo lo he mezclado todo.

Se trata de ver cual de las 11 vocales que tiene el idioma inglés es la que se ha pronunciado. Para ello se usan 10 atributos.

La tabla con la información es 4

3.4.5 Fall Detection

[See 9]

Se trata de identificar cuando una persona se ha caido al suelo o en qué otro estado está (de pie, tumbado, caminando).

La tabla con la información es 5

3.4.6 MNIST

[See 8]

La tabla con la información es 6

3.4.7 Segment

[See 14]

La tabla con la información es 7

3.4.8 Digits

[See 2]

La tabla con la información es 8

Name	pen-digits
# Features	16
# Instances	10992
# Classes	10

Table 1: Info. of pen-digits

Name	covertype
# Features	12
# Instances	4900
# Classes	7

Table 2: Info. of covertype

Name	satellite
# Features	36
# Instances	6435
# Classes	6

Table 3: Info. of satellite

Name	vowel
# Features	10
# Instances	990
# Classes	11

Table 4: Info. of vowel

Name	fall-detection
# Features	6
# Instances	16382
# Classes	6

Table 5: Info. of fall-detection

Name	mnist
# Features	12
# Instances	4900
# Classes	7

Table 6: Info. of mnist

Name	segment
# Features	717
# Instances	5000
# Classes	10

Table 7: Info. of segment

Name	digits
# Features	64
# Instances	5620
# Classes	10

Table 8: Info. of digits

4 Experimental Results

1. Plantear la hipótesis
2. Plantear los experimentos que habría que hacer
3. Mostrar los resultados que he obtenido
4. Discutir temas 2 a 2
5. Contrastar las hipótesis, y ver si se han refutado o support

4.1 Enfrentar resultados 2 a 2

Los experimentos ya los tengo planteados 2 a 2. Entonces, se trata de comentar cada uno de los experimentos?

4.2 Contrastar hipótesis con resultados

5 Conclusion and Future Work

De momento, parece que algunos problemas sí que se benefician de esto, mientas que otros no lo hacen

Future work

- Problemas de regresión
- Aproximar kernels distintos a rbf
- Ver el comportamiento con problemas reales, que tienen missings, clases mal balanceadas, etc.
- Otros tipos de ensembles, como el boosting
- Generalización de bootstrap que regule la cantidad de aleatoriedad
- Procedimiento para sacar la cantidad promedio de ruido que tiene un problema

	Project development	**Exploitation**	**Risks**
Environmental	Consumption design	Ecological Footprint	Environmental risks
Economic	Project bill	Viability plan	Economic risks
Social	Personal impact	Social impact	Social risks

- El trabajo se ha centrado en problemas de clasificación, pero no hay ningún motivo para que no se pueda aplicar el regresión. Se ha omitodo por simplificar
- Aquella teoría de que quizá se puede regular la cantidad de aleatoriedad que añade el bootstrap, y quizá inventar un bootstrap con un parámetro para regular la cantidad de aleatoriedad
- Pensar en aquella teoría de que quizá se puede inventar un procedimiento para, dato un problema determinado con sus datos, sacar un número que sea representativo de la cantidad promedio de ruido que tiene. Puesto que quizá es útil para este proyecto conocer la cantidad de aleatoriedad que tienen los datos, para que se pueda regular

6 Sustainability Report

Consumo del equipo CO2 Materiales Peligrosos Materiales que vienen de zonas de conflicto Qué sabemos de nuestros proveedores, y si la fabricación de la maquinaria se ha hecho en una instalación sin riesgo, ni para las personas ni para la naturaleza

El impacto que tiene el proyecto directa e indirectamente en la gente Ha sido diseñado pensando en cómo se ha de reciclar y reparar, siguiendo conceptos de economía circular?

6.1 Environmental

El impacto medio-ambiental de mi TFG ¿Cuál es el coste medio-ambiental de los productos TIC? ¿Cuantos recursos son necesarios para fabricar un dispositivo? ¿Cuánto consumen estos dispositivos durante su vida útil? ¿Cuantos resuduos se generan para fabricarlos? ¿Qué hacemos de los dispositivos una vez ha terminado su vida útil? ¿Los tiramos y contaminamos el medio de nuestro entorno? ¿Los enviamos al tercer mundo, y contaminamos el suyo?

La huella ecológica se puede medir, por ejemplo, en Kilovatios hora y en emisiones de CO2

La huella ecológica que tendrá el proyecto durante toda su vida útil ¿Mi TFG contribuirá a reducir el consumo energético y la generación de resudios? ¿O los incrementará?

6.2 Economic

Costos de portarlo a terme i assegurar la seva pervivència

Estimar los recursos materiales y humanos necesarios para la realización del proyecto. Sería como preparar la factura que se le pasará al cliente, teniendo en cuenta que se terminará en un término bien definido

Una planificación temporal y Una explicación de si he pensado algún proceso para reducir costes

Estimar las desviaciones que he tenido respecto a las planificaciones iniciales.

Para evaluarlo durante su vida útil, he de hacer un pequeño estudio de mercado, y analizar en qué se diferencia de los ya existentes, si se mejora en algún aspecto, o no. Reflexionar sobre si mi producto tendrá mercado o no, y sobretodo, explicarlo

Además de estudiar los costes, plantearse si es posible reducirlos de alguna manera

Plantear posibles escenarios que por razones de limitaciones en el tiempo y de recursos no tendré en cuenta, pero que podrían perjudicar la viabilidad económica del proyecto

Suele salir a unos 20000 € el TFG

6.3 Social

Implicaciones sociales, tanto sobre el colectivo al que se dirige el proyecto como sobre otros colectivos

6.3.1 Impacto personal

En qué me ha afectado a mí, y a mi entorno cercano, la realización de este proyecto. En qué me ha cambiado la vida, o si ha cambiado mi visión sobre el tema. ¿Ha hecho que me de cuenta de situaciones que antes ignoraba?

6.3.2 Impacto social

Implicaciones que la realización de este proyecto puede tener sobre la sociedad. Hay que identificar al colectivo de los afectados. Los colectivos pueden ser: los propietarios, los gestores del proyecto, los trabajadores, los proveedores, los consumidores (usuarios directos), o terceros (usuarios indirectos o pasivos) Puedo consultar los estándares del GRI

6.3.3 Riesgos sociales

Explicar posibles escenarios probables, pero no significativos, que no puedo abordar por falta de tiempo o de recursos o de capacidad, y que podrían perjudicar a personas relacionadas con mi proyecto

References

[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] Leo Breiman. "Bagging Predictors". In: (Sept. 1994). URL: https://www.stat.berkeley.edu/~breiman/bagging.pdf.

[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] Rong Jin et al. "Improved Bounds for the Nyström Method with Application to Kernel Classification". In: (2011). URL: https://arxiv.org/pdf/1111.2262.pdf.

[7] Quoc Le, Tamás Sarlós, and Alex Smola. "Fastfood — Approximating Kernel Expansions in Loglinear Time". In: (2013). URL: https://www.robots.ox.ac.uk/~vgg/rg/papers/le__icml2013__fastfood.pdf.

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

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

[10] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines". In: (2007).

[11] Robert E. Schapire. "The Strength of Weak Learnability". In: (1990). URL: http://rob.schapire.net/papers/strengthofweak.pdf.

[12] Yauheni Selivonchyk. "Speeding up Support Vector Machines with Random Fourier Features". In: (2016). URL: http://cscubs.cs.uni-bonn.de/2016/proceedings/paper-04.pdf.

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

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

[15] Shuai Zhang et al. "Stacked Kernel Network". In: (Nov. 2017). URL: https://arxiv.org/pdf/1711.09219.pdf.