Universidad Autónoma de Madrid
Escuela Politécnica Superior - Departamento de Ingeniería Informática

Algoritmos SVM para problemas sobre big data

Trabajo fin de máster para
Máster en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones

Yvonne Gala García
bajo la dirección de
José Ramón Dorronsoro Ibero

Madrid, 25 de septiembre de 2013

Índice general

Índice de figuras

Índice de cuadros

Resumen

Las Máquinas de Vectores Soporte es una de las técnicas más poderosas del aprendizaje automático cuya idea principal consiste en encontrar un separador lineal de las clases. Con una serie de transformaciones esta idea puede ser extendida para problemas en el que los datos no sean linealmente separables mediante la proyección a un espacio de dimensión superior, es decir, construimos un separador lineal en el espacio proyectado para unos datos que en el espacio de entrada no son linealmente separables. Esta técnica puede ser utilizada tanto para problemas de clasificación como de regresión.

La ventaja de las Máquinas de Vectores Soporte está en la sencillez de los modelos, así como su robustez y buena generalización para nuevos datos.

El algoritmo Sequential Minimal Optimization (SMO) ha sido el más extensamente utilizado, ya que es el que utiliza la librería LIBSVM. Consiste en resolver subproblemas del problema inicial para que el coste computacional sea menos costoso. A lo largo de los últimos años, se han desarrollado algoritmos que usando la información del gradiente obtienen una tasa de convergencia mucho menor que la de SMO, es decir, convergen en menos iteracciones a la soución óptima, como son Dual Coordinate Descent Method (DCD) o Stochastic sub-gradient Descent Method (Pegasos).

Por tanto en este trabajo fin de máster será estudiado el algoritmo y convergencia de SMO, DCD y Stochastic sub-gradient Descent Method. Además serán aplicados a problemas con mucho auge en la actualidad como son la predicción de energía solar y la clasificación de páginas web y en los que debido a la creciente importancia, cada día disponemos de más información, y por tanto más patrones siendo necesario el uso de algoritmos eficientes con una convergencia a la solución óptima en pocas iteracciones.

Agradecimientos

Este Trabajo Fin de Master sintetiza gran parte de mi trabajo realizado durante el último año con el grupo de investigación GAA, Escuela Politécnica Superior, Universidad Autónoma de Madrid. El trabajo ha sido patrocinado por la Beca de la Cátedra ADIC-UAM de modelado y predicción. Por tanto me gustaría agradecer el apoyo de la cátedra y a mi tutor, José Ramón Dorronsoro por su guía y por su apoyo durante este periodo de tiempo.

Estructura general

En este trabajo fin de máster hemos tratado de estudiar el estado del arte de los algoritmos de las Máquinas de Vectores Soporte. En la actualidad la mayoría de las investigaciones se centran en encontrar u optimizar los algoritmos ya existentes para hacerlos más eficaces. Muchas de estas investigaciones se encaminan usando la información del gradiente en cada coordenada para hacer la optimización más eficiente. A lo largo de este trabajo hemos estudiado la teoría básica de optimización y Máquinas de Vectores Soporte (SVM), algoritmos clásicos de las SVM como Sequential Minimal Optimization (SMO) y algoritmos más nuevos y optimizados como Dual Coordinate Descent Method (DCD) o Stochastic sub-gradient Descent Method. Por último hemos usados estos tres algoritmos en dos problemas diferentes el de predicción de radiación solar y el de clasificación de páginas web. Todo esto ha sido desarrollado de la siguiente manera:

En el primer capítulo vamos a explicar la teoría básica de optimización y de las Máquinas de Vectores Soporte, junto con el algoritmo SMO. SMO un método de descomposición para las SVM que reduce el coste computacional eligiendo subproblemas de menor tamaño y que es el utilizado e implementado en la famosa librería LIBSVM.

En el segundo capítulo hemos estudiado dos tipos de algoritmos online y batch, que utilizando la información del gradiente optimizan la convergencia del algoritmo, estos son; Dual Coordinate Descent Method (DCD) y Stochastic sub-gradient Descent Method.

El tercer capítulo está dedicado a los experimentos numéricos. En ellos hemos utilizado algunas de librerías disponibles para estos algoritmos, como LIBSVM para SMO, LIBLINEAR para DCD y Pegasos para el método Stochastic subgradient descent. También hemos implementando una búsqueda exhaustiva en rejilla con un conjunto de validación para dos problemas muy diferentes. El primero se trata de un problema de clasificación de predicción de radiación solar, el segundo de clasificación de páginas web.

Por último, el cuarto capítulo resume algunas de las conclusiones obtenidas en este trabajo fin de máster y posibles trabajos futuros.

Capítulo 1

Teoría clásica de las Máquinas de Vectores Soporte

Las Máquinas de Vectores de Soporte o Support Vector Machines (SVM) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik.

El objetivo de los problemas de clasificación que aplican este tipo algoritmos de aprendizaje supervisado es el siguiente; dado un conjunto de entrenamiento con sus etiquetas de clase, entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra o conjunto de test.

Las SVM son una de las técnicas más poderosas del aprendizaje automático. Consiste en construir un hiperplano en un espacio de dimensionalidad muy alta (o incluso infinita) que separe las clases que tenemos. Una buena separación entre las clases permitirá un clasificación correcta de la nueva muestra, es decir, necesitamos encontrar la máxima separación a los puntos más cercanos a este hiperplano.

La figura 1.0.1 muestra un ejemplo gráfico del hiperplano que separa la muestra en dos clases.

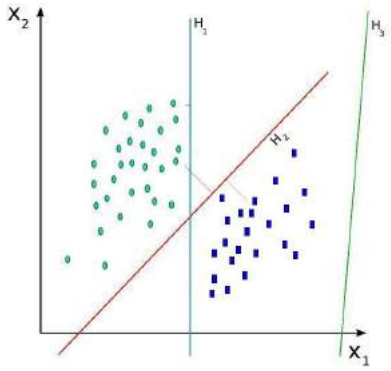

Figura 1.0.1: Ejemplo sencillo en el que tenemos varios hiperplanos y tratamos de encontrar aquel que mejor separe la muestra.

Por tanto, detrás de la teoría de las Máquinas de Vectores Soporte está maximizar la distancia del hiperplano a los puntos más cercanos de la muestra, por lo que antes

de profundizar más en ella es necesario explicar algunos conceptos de la teoría de optimización.

En el siguiente apartado vamos a explicar tanto conceptos básicos de optimización, como herramientas para convertir un problema primal a su correspondiente problema dual, condiciones de Kunh-Tucker y condiciones necesarias y suficientes para determinar la solución de un problema de optimización.

1.1. Optimización

La teoría de optimización [1] es una herramienta imprescindible en el desarrollo de la técnica usada por las Máquinas de Vectores Soporte, ésta es la razón por la que se justifica la presencia de este capítulo.

Un problema de optimización trata de encontrar el máximo o el mínimo de una función sujeta a una serie de restricciones que pueden ser de igualdad o desigualdad. Dependiendo de la función a optimizar y de las restricciones tenemos infinidad de tipos de problemas de optimización en los que podemos usar diferentes algoritmos. En esta sección vamos a centrarnos en los problemas de optimización convexa cuadráticos con restricciones lineales, es decir aquellos en los que la función objetivo es cuadrática y las restricciones son lineales.

1.1.1. Problema de optimización primal

En este apartado vamos a definir los conceptos de función convexa, problema de optimización primal y condición de Fermat.

Definicion 1. *Una función f, definida en $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ es convexa si para todo $x, y \in \mathbb{R}^n$ existe θ tal que:*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

La figura 1.1.1 muestra un ejemplo gráfico de función convexa.

Definicion 2 (Problema de optimización primal)**.** *Dadas las funciones convexas f, g_i, $i = 1, \cdots, k$, y h_i, $i = 1, \cdots, m$, definidas en un dominio $\Omega \subseteq \mathbb{R}^n$. El problema de optimización primal será el siguiente:*

$$\begin{array}{lll} \text{mín} & f(\mathbf{w}), & \mathbf{w} \in \Omega, \\ s.t & g_i(\mathbf{w}) \leq 0, & i = 1, \cdots, k, \\ & h_i(\mathbf{w}) = 0, & i = 1, \cdots, m, \end{array} \tag{1.1.1}$$

donde $f(\mathbf{w})$ es conocida como función objetivo, $h_i(\mathbf{w})$ son las restricciones de igualdad y $g_i(\mathbf{w})$ las restricciones de desigualdad.

La solución del problema de optimización convexa 1.1.1 viene dada por $\mathbf{w}^* \in R$ donde R se conoce como región factible y se define de la siguiente manera:

$$R = \{\mathbf{w} \in \Omega : g(\mathbf{w}) \leq 0, h(\mathbf{w}) = 0\}$$

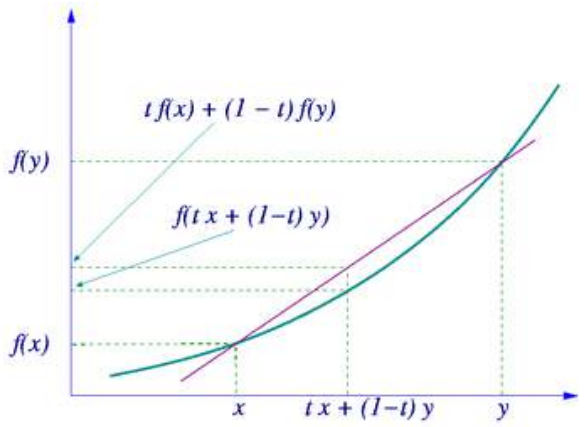

Figura 1.1.1: Figura que muestra un ejemplo de función convexa.

Definicion 3 (Mínimo local)**.** *Un mínimo w^* de f es local, si existe ϵ tal que w^*, para todo $w \in (w^* - \epsilon, w^* + \epsilon)$, $f(w) \leq f(w^*)$*

Definicion 4 (Mínimo global)**.** *Un mínimo w^* de f es global en un dominio D, si para todo $w \in D$ y $w^* \in D$, $f(w^*) \leq w$.*

En el caso del problema 1.1.1 en el que tenemos un problema de optimización convexa cuadrático, cualquier mínimo local será mínimo global, ya que el mínimo del problema es único por convexidad, y además siempre existirá por tratarse de un problema cuadrático. Es decir, la solución del problema 1.1.1 es aquel $\mathbf{w}$ que cumple las restricciones, y además es un mínimo global.

Esta es una de las ventajas de las Máquinas de Vectores Soporte frente a otros algoritmos del aprendizaje automático, y es que la solución siempre existe y es única.

Por último el teorema 1 nos dará las condiciones para que un punto $\mathbf{w}^*$ sea mínimo de una función f. Este teorema se conoce como Teorema de las condiciones de Fermat [1].

Teorema 1 (Condiciones de Fermat)**.** *Si una función es convexa y diferenciable, y además el gradiente es cero en un punto $\mathbf{w}^*$, entonces existe un mínimo de la función en ese punto. En otras palabras;*

$$\frac{\partial f(\mathbf{w}^*)}{\partial \mathbf{w}} = 0$$

junto con la convexidad de f es condición suficiente para que $\mathbf{w}^*$ sea un mínimo.

Al tratarse de un problema de optimización con restricciones no podemos usar las técnicas clásicas analíticas, por ello es necesario introducir la teoría Lagrangiana. Ésta nos permite resolver problemas de optimización con restricciones sin resolver explícitamente esas restricciones.

1.1.2. Teoría Lagrangiana

El principal uso de la teoría Lagrangiana [2] en este contexto es transformar un problema de optimización con restricciones de igualdad en uno con restricciones simples. Esta transformación será llevada a cabo a través de una nueva función que introduciremos conocida como función lagrangiana.

Definicion 5 (Lagrangiano)**.** *Dado un problema de optimización cuya función objetivo es $f(\mathbf{w})$, y las restricciones de igualdad son $h_i(\mathbf{w}) = 0, i = 0, ..., m$*

$$\begin{array}{ll} \text{mín} & f(\mathbf{w}), \qquad \mathbf{w} \in \Omega, \\ s.t & h_i(\mathbf{w}) = 0, \quad i = 1, \cdots, m \end{array} \tag{1.1.2}$$

definimos el lagrangiano de la función (1.1.2) *como:*

$$L(\mathbf{w}, \beta) = f(\mathbf{w}) + \sum_{i=1}^{m} \beta_i h_i(\mathbf{w}), \tag{1.1.3}$$

donde β_i son conocidos como los multiplicadores de Lagrange.

Como nuestro objetivo es minimizar la función lagrangiana, existe un teorema que nos garantiza la existencia de ese mínimo [1].

Teorema 2. *Una condición necesaria para que un punto $\mathbf{w}^*$ sea mínimo de la función $f(\mathbf{w})$ sujeta a $h_i(\mathbf{w}) = 0, i = 1, .., m$ con f y $h_i \in C^1$ es:*

$$\frac{\partial L(\boldsymbol{w}^*, \beta^*)}{\partial \mathbf{w}} = 0$$
$$\frac{\partial L(\mathbf{w}^*, \beta^*)}{\partial \beta} = 0$$

Las condiciones anteriores también son suficientes si $L(\mathbf{w}, \beta^*)$ es una función convexa de $\mathbf{w}$.

1.1.3. Formulación Dual del problema Primal

Dado un problema de optimización primal puede ser transformado a su correspondiente problema dual mediante la siguiente fórmula:

$$\begin{array}{ll} & \theta(\alpha, \beta) = inf_{\mathbf{w} \in \Omega} L(\mathbf{w}, \alpha, \beta) \\ s.t. & \alpha \geq 0, \end{array} \tag{1.1.4}$$

donde L es el lagrangiano de la función primal 1.1.2.

Como podemos observar, en este nuevo problema tratamos de encontrar el máximo de una función θ, que sólo dependerá de los multiplicadores de Lagrange, i.e. α_i y β_j para todo i, j. De hecho, la función θ se define como el ínfimo del lagrangiano sobre $\mathbf{w}$.

En la mayoría de las ocasiones resolver el problema dual resulta mucho más sencillo y en el caso de las SVM, tiene una serie de ventajas que explicaremos más adelante. De la formulación dual, obtenemos varios teoremas que serán útiles en la búsqueda del óptimo [1].

Teorema 3 (Teorema débil de la dualidad)**.** *Dado el problema de optimización 1.1.1 y su dual definido como 1.1.4, si* $\mathbf{w} \in \Omega$ *es un punto de la región factible del problema primal y* (α, β) *es una solución factible del dual, existe la siguiente relación;*

$$f(\mathbf{w}) \geq \theta(\alpha, \beta).$$

Esto se deduce inmediatamente de la definición 1.1.4 del problema dual ya que;

$$\theta(\alpha, \beta) = inf_{\mathbf{w}\in\Omega} L(\mathbf{w}, \alpha, \beta) \leq L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \alpha g(\mathbf{w}) + \beta h(\mathbf{w}) \leq f(\mathbf{w}).$$

Definicion 6 (Gap Dual)**.** *La diferencia entre el valor óptimo del problema primal* $\mathbf{w}^*$ *y de problema dual* (α^*, β^*) *es conocido como gap dual.*

Teorema 4 (Teorema fuerte de la dualidad)**.** *Dado el problema de optimización 1.1.1 y su dual definido como 1.1.4, si las restricciones son funciones afines, es decir, satisfacen* $f(x) = Ax + b$*, entonces el gap dual será cero. Es decir,*

$$f(\mathbf{w}^*) = \theta(\alpha^*, \beta^*),$$

con $f(\mathbf{w}^*)$ *y* $\theta(\alpha^*, \beta^*)$ *soluciones óptimas del problema primal y dual respectivamente.*

Resumiendo, el teorema débil de la dualidad da la relación entre la solución del problema dual y primal, donde el valor de la función objetivo en el problema dual es menor o igual que el valor de la función objetivo en el problema primal. Por el teorema fuerte de la dualidad sabemos que el valor la función de optimización del problema primal y dual coinciden en el óptimo.

1.1.4. Condiciones de Kuhn-Tucker

Hasta ahora los problemas que hemos tratado con la teoría Lagrangiana básica son problemas de optimización con restricciones de igualdad, como es el caso de 1.1.2. Las condiciones de Kuhn-Tucker sirven para generalizar los problemas de optimización de funciones convexas con restricciones de igualdad a problemas con restricciones desigualdad también, como en el problema (1.1.1).

$$\begin{array}{lll} \text{mín} & f(\mathbf{w}), & \mathbf{w} \in \Omega, \\ \text{s.t} & g_i(\mathbf{w}) \leq 0, & i = 1, \cdots, k, \\ & h_i(\mathbf{w}) = 0, & i = 1, \cdots, m, \end{array} \tag{1.1.5}$$

Para este caso definimos la nueva función lagrangiana como:

$$L(\mathbf{w}, \beta) = f(\mathbf{w}) + \sum_{i=1}^{k} \alpha_i g_i(\mathbf{w}) + \sum_{j=1}^{m} \beta_i h_i(\mathbf{w}), \tag{1.1.6}$$

donde α_i y β_j son los multiplicadores de Lagrange y se cumple que; f función convexa, g_i, h_i afines y $\alpha_i \geq 0$.

Para garantizar la existencia del mínimo tenemos un nuevo teorema, en las que se especifican las condiciones de Kuhn-Tucker.

Teorema 5 (Condiciones KKT)**.** *Una condición necesaria y suficiente para que un punto* $\mathbf{w}^*$ *sea óptimo es la existencia de* α^*, β^* *tal que:*

$$\begin{aligned}
\frac{\partial L(\mathbf{w}^*, \alpha^*, \beta^*)}{\partial \mathbf{w}} &= 0 \\
\frac{\partial L(\mathbf{w}^*, \alpha^*, \beta^*)}{\partial \beta} &= 0 \\
\alpha_i^* g_i(\mathbf{w}^*) &= 0, \quad i = 1, \cdots, k \\
g_i(\mathbf{w}^*) &\leq 0, \quad i = 1, \cdots, k \\
\alpha_i^* &\geq 0, \quad i = 1, \cdots, k.
\end{aligned}$$

La ecuación $\alpha_i^* g_i(\mathbf{w}^*) = 0, \quad i = 1, \cdots, k$ es conocida como la condición KKT complementaria.

Una vez definidas la formulación primal y dual del problema de optimización, la función lagrangiana y las condiciones KKT estamos en condiciones de comenzar a estudiar la teoría básica de las Máquinas de Vectores Soporte.

1.2. Máquinas de Vectores Soporte (SVM)

Las Máquinas de Vectores Soporte [1], en inglés Support Vector Machines (SVM), es una de los técnicas más poderosas del aprendizaje automático, que a pesar de su sencillez ha demostrado ser un algoritmo robusto y que generaliza bien en problemas de la vida real.

Como hemos explicado en la introducción de este capítulo, la técnica de las SVM consiste en construir un hiperplano en un espacio de dimensionalidad muy alta que separe las clases que tenemos. Esta técnica puede ser utilizada tanto en problemas de clasificación como de regresión. Una buena separación entre las clases permitirá un clasificación correcta.

En esta sección comenzaremos explicando el problema más sencillo de SVM en clasificación, es decir, aquel en el que los datos son linealmente separables, para después generalizarlo aquellos casos en los que no lo sean introduciendo una nueva variable que nos permita cometer ciertos errores. Posteriormente pasaremos a explicar las Máquinas de Vectores Soporte para regresión y por último introduciremos los conceptos de núcleo y proyección.

1.2.1. Caso linealmente separable

Dado que el objetivo de las SVM es buscar el hiperplano óptimo que mejor separe las clases, introduciremos el concepto de margen para conseguirlo. Este concepto sólo tiene sentido en los casos en los que datos sean linealmente separables, pero es la base para obtener el problema de optimización que nos dará el algoritmo de las SVM.

Definicion 7. *Si consideramos el conjunto de datos de entrenamiento como* $(\mathbf{x}_i, y_i), i = 1, ..., m$, *con* $\mathbf{x}_i \in \mathbb{R}^l$ *e* $y_i \in \{-1, 1\}$ *y suponemos que existe un hiperplano que separa esos datos dado por:*

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b.$$

Entonces, podemos definir el margen como la suma de las distancias de los puntos más cercanos al hiperplano, es decir:

$$\gamma = \frac{1}{2}(\frac{\mathbf{w}}{||\mathbf{w}||_2} \cdot \mathbf{x}^+ - \frac{\mathbf{w}}{||\mathbf{w}||_2} \cdot \mathbf{x}^-) = \frac{1}{2||\mathbf{w}||_2}(\mathbf{w} \cdot \mathbf{x}^+ - \mathbf{w} \cdot \mathbf{x}^-) = \frac{2}{||\mathbf{w}||_2},$$

donde $\mathbf{x}^+$ *y* $\mathbf{x}^-$ *representan los patrones pertenecientes a la clase positiva y negativa respectivamente.*

Por tanto, el objetivo es maximizar el margen $\gamma = \frac{2}{||\mathbf{w}||_2}$, matemáticamente esto es lo mismo que minimizar la siguiente función:

$$\text{mín} \frac{1}{2}||\mathbf{w}||^2 \tag{1.2.1}$$

sujeta a una serie de restricciones, que vienen de la condición de que las clases positivas ($y_i = 1$) deben estar separadas de las clases negativas ($y_i = -1$).

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{para } y_i = +1 \tag{1.2.2}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{para } y_i = -1 \tag{1.2.3}$$

y donde las restricciones 1.2.2 y 1.2.3 pueden ser unidas en una sóla de la siguiente manera:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i. \tag{1.2.4}$$

Finalmente, si juntamos la función 1.2.1 con la restricción 1.2.4 obtenemos el problema primal de optimización de las Máquinas de Vectores Soporte:

$$\begin{array}{ll} \min\limits_{\mathbf{w},b} & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t} & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \forall i. \end{array} \tag{1.2.5}$$

Por tanto nos encontramos ante un problema de optimización cuadrática convexa con restricciones lineales, en el que el hiperplano se obtiene minimizando la norma del vector de pesos $\mathbf{w}$.

Para minimizar la función usaremos la función lagrangiana para el problema 1.2.5, que por lo explicado en la sección previa la podemos caracterizar como:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1], \tag{1.2.6}$$

donde L debe ser minimizada sobre $\mathbf{w}$ y b, por tanto calculamos $\frac{\partial L}{\mathbf{w}}$ y $\frac{\partial L}{b}$ e igualamos a cero, obteniéndose las siguientes ecuaciones:

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i = 0$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{b} = \sum_{i=1}^{n} y_i \alpha_i = 0,$$

que sustituidas en la ecuación de Lagrangiano 1.2.6 nos dará:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i,j=1}^{l} y_i y j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{l} \alpha_i y_i b + \sum_{i=1}^{l} \alpha_i = \Theta(\alpha) \tag{1.2.7}$$

y por tanto, el problema dual para las SVM será:

$$\begin{array}{ll} \max\limits_{\alpha} & \Theta(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{s.t} & \alpha_i \geq 0 \quad \forall i \\ & \sum_i \alpha_i y_i = 0. \end{array} \tag{1.2.8}$$

Por último, la condición KKT complementaria será:

$$\alpha_i^* (y_i((\mathbf{w} \cdot \mathbf{x}_i + b^*) - 1)) = 0.$$

Una vez obtenida la solución óptima del problema dual (α^*, b^*) de las SVM 1.2.8 de las condiciones KKT y las derivadas parciales del lagrangiano, podemos obtener los pesos $\mathbf{w}^*$ del problema primal mediante la ecuación dada por $\frac{\partial L}{\mathbf{w}} = 0$:

$$\mathbf{w}^* = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i, \tag{1.2.9}$$

mientras que el término de bias se corresponderá con:

$$b^* = y_i - \mathbf{w}^* \cdot \mathbf{x}_i = y_i - \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x}_j.$$

También de la condición KKT complementaria observamos que si $\alpha_i^* > 0$ entonces

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b^*) = 1, \tag{1.2.10}$$

por tanto estos serán los puntos que están en el hiperplano. En resumen, por la ecuación 1.2.9 y de la condición 1.2.10 deducimos que los puntos que definen los pesos son aquellos que están en el hiperplano óptimo, es decir, aquellos en los que $\alpha_i^* > 0$ y por tanto se tiene 1.2.10. A estos puntos los llamaremos *vectores soporte*.

Finalmente el hiperplano óptimo puede representarse mediante α^* y b^* como:

$$f(\mathbf{x}, \mathbf{w}^*, b^*) = \sum_{i \in vs} y_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b^*.$$

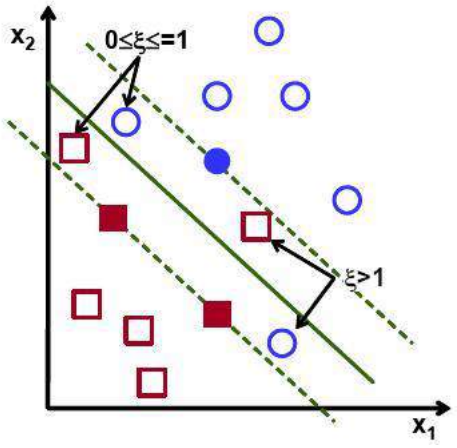

Figura 1.2.1: Ejemplo de problema no separable linealmente

1.2.2. Caso no separable linealmente

En la vida real, la mayoría de los problemas no tienen datos linealmente separables, lo que imposibilita la tarea de encontrar un hiperplano que separe perfectamente los datos. Para resolver este problema introducimos una nueva variable, a la que conocemos como variable de holgura y denotamos por ξ. El objetivo de esta nueva variable es hacer el modelo menos rígido y permitir ciertos errores, es decir, que algunos puntos de clase $+1$ sean clasificados como -1 y viceversa.

Al introducir una nueva variable, lógicamente, nuestro problema cambia, ya que tanto la función a optimizar como las restricciones no son las mismas. Las restricciones ahora tendrán la forma:

$$y_i(\mathbf{w}\cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \forall i \tag{1.2.11}$$

permitiendo un margen ξ_i de equivocación. La figura 1.2.1 nos muestra gráficamente el efecto de ξ_i en la separación de las clases.

Al mismo tiempo, vamos a penalizar ese error con un nuevo parámetro C, que es añadido a la función objetivo, de tal manera que a mayor C, mayor es la penalización que damos a los errores y por tanto permitimos menos. Por el contrario si C es pequeño, permitimos a nuestro modelo cometer más errores.

El valor de C va ser importante en este sentido, ya que si damos un valor demasiado grande, el modelo penalizará mucho los errores cometidos en el conjunto de entrenamiento y por tanto se producirá el *overfitting* o sobreaprendizaje, esto es, el modelo sobre aprende los datos de entrenamiento, ciñéndose a ellos, lo que produce que no haya una buena generalización y la clasificación en los nuevos datos de test no sea buena. Por otro lado, si C es muy pequeño, el modelo permitirá muchos errores y no será bueno, produciéndose el conocido *underfitting*.

Después de estas modificaciones, el problema de optimización obtenido para SVM en el que los datos no son linealmente separables será el siguiente:

$$\begin{array}{cc} \min\limits_{w,b} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t} & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0 \quad \forall i \\ & \xi_i \geq 0 \forall i, \end{array} \tag{1.2.12}$$

Por tanto vamos a resolver un problema en el que tratamos de encontrar el máximo margen, como en el caso de problemas con datos linealmente separables, pero permitiendo ciertos errores, por lo que $\sum_{i=1}^{l}\xi_i$ se convertirá en una cota superior de los errores que permitimos, ya que para que un error ocurra debe exceder ξ_i y por tanto la suma será una cota para el error total controlado por el parámetro C.

Como en el caso anterior al tratarse de un problema con restricciones obtendremos su función lagrangiana usando la teoría lagrangiana:

$$L(\mathbf{w}, b, \alpha, \beta) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{l}\beta_i\xi_i,$$

donde L debe ser minimizada sobre $\mathbf{w}$, ξ_i y b para obtener el dual. Calculamos $\frac{\partial L}{\mathbf{w}}$, $\frac{\partial L}{\xi_i}$ y $\frac{\partial L}{b}$ e igualamos a cero, obteniéndose las siguientes ecuaciones:

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \beta)}{\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} y_i\alpha_i\mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{l} y_i\alpha_i\mathbf{x}_i$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \beta)}{\xi_i} = C - \alpha_i - \beta_i = 0 \Rightarrow 0 \leq \alpha_i \leq C$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \beta))}{b} = \sum_{i=1}^{l} y_i\alpha_i = 0.$$

Finalmente obtenemos el siguiente problema dual del problema primal 1.2.12

$$\begin{array}{cc} \max\limits_{\alpha} & \Theta(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j \\ \text{s.t} & C \geq \alpha_i \geq 0 \quad \forall i \\ & \sum_i \alpha_i y_i = 0 \end{array} \tag{1.2.13}$$

y cuyas condiciones KKT son:

$$\alpha_i^*(y_i(\mathbf{x}_i\cdot\mathbf{w}^* + b^*) - 1 + \xi_i^*) = 0$$

$$C \geq \alpha_i^* \geq 0$$

$$\beta_i^* \geq 0$$

$$\beta_i^*\xi_i^* = 0 \Rightarrow (C - \alpha_i^*)\xi_i^* = 0.$$

Además, de las condiciones KKT observamos que:

Si $\alpha_i^* = 0 \Rightarrow \xi_i^* = 0 \Rightarrow y_i(\mathbf{w}^*\cdot\mathbf{x}_i + b^*) \geq 1$.

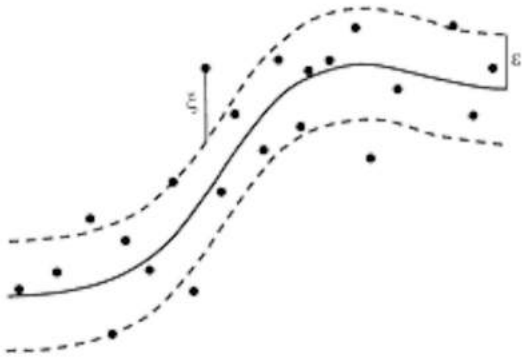

Figura 1.2.2: Ejemplo de banda ϵ no lineal

Si $C \geq \alpha_i \geq 0 \Rightarrow \xi_i^* = 0 \Rightarrow y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) = 1$

Si $\alpha_i^* = C$ y $\xi > 0 \Rightarrow y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) = 1 - \xi_i^* \leq 1$.

Por tanto, el problema dual de SVM y las condiciones KKT nos permiten definir los vectores soporte como aquellos que están en el margen, es decir, aquellos que $\alpha_i \neq 0$. En consecuencia, el número de vectores soporte nos dará una idea de la dificultad del problema.

1.2.3. Máquinas de Vectores Soporte para Regresión

Como hemos dicho anteriormente la teoría de las Máquinas de Vectores Soporte puede ser extendida a problemas de regresión. El objetivo en este caso es encontrar una función $f(x)$ con una desviación ϵ del "target"y_i para todos los datos.

La variable ϵ es conocida como la anchura de la banda y es deseable que no tenga un valor muy alto manteniendo un equilibrio con la complejidad del problema. Por tanto vamos a permitir ciertos errores para aquellos valores predichos que estén cerca del valor real. Para ello introducimos el concepto de función de pérdida ϵ que permite errores menores que ϵ del valor real.

En estos modelos también introducimos la variable de holgura ξ para permitir ciertos errores, que es acompañada por su correspondiente término de regularización C en la función objetivo.

La fórmula 1.2.14 nos da función de pérdida ϵ y la figura 1.2.3 y 1.2.3 muestran un ejemplo gráfico de la banda ϵ y el parámetro de holgura ξ con una función no lineal y lineal respectivamente.

$$L(\mathbf{x}, y; f) = |y - f(\mathbf{x})|_\epsilon = max(0, |y - f(\mathbf{x}| - \epsilon). \tag{1.2.14}$$

En cuanto al valor de ϵ, la anchura de la banda, es importante de manera similar a como lo era C en los problemas no linealmente separables. Si la banda ϵ es muy grande permitimos mucho error, ya que nuestra predicción estará más "lejos"del objetivo y_i.

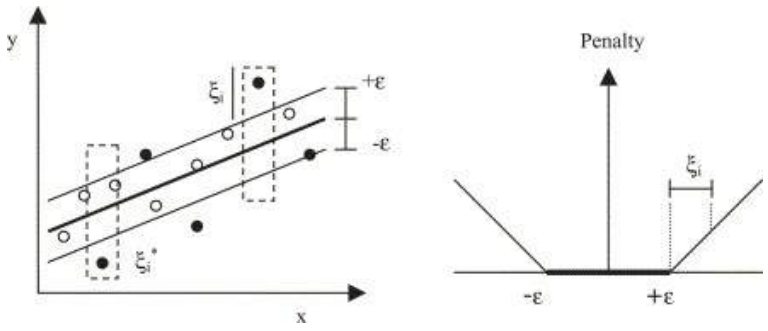

Figura 1.2.3: Gráfica que muestra cómo influyen ξ y ϵ gráficamente.

En cambio si ϵ es muy pequeño permitimos muy poco error y la complejidad del modelo aumenta rápidamente.

Por tanto existe una relación entre C y ϵ a la hora de buscar los parámetros óptimos del modelo. Si C es muy grande ϵ también lo será, ya que sino tendríamos modelos muy complicados, en los que se permitirian pocos errores, con riesgo de *overfitting*. En caso contrario si C es pequeño ϵ también debería serlo, ya que en caso contrario nuestro modelo sería permisivo con los errores y habría riesgo de *underfitting*.

El problema de optimización primal se reescribe ahora:

$$\begin{array}{rl} \min\limits_{w,b} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}(\xi_i + \hat{\xi}_i) \\ \text{s.t} & (\mathbf{x}_i \cdot \mathbf{w} + b) - y_i \leq \epsilon + \xi_i \quad \forall i \\ & y_i - (\mathbf{x}_i \cdot \mathbf{w} + b) \leq \epsilon + \hat{\xi}_i \quad \forall i \\ & \xi_i, \hat{\xi}_i \geq 0 \forall i, \end{array} \tag{1.2.15}$$

de donde obtenemos la función lagrangiana:

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}(\xi_i + \hat{\xi}_i) \\ &+ \sum_{i=1}^{l}\alpha_i[(\mathbf{w}\cdot\mathbf{x}_i + b) - y_i - \epsilon - \xi_i] \\ &+ \sum_{i=1}^{l}\hat{\alpha}_i[y_i - (\mathbf{w}\cdot\mathbf{x}_i + b) - \epsilon - \hat{\xi}_i] \\ &- \sum_{i=1}^{l}(\beta_i\xi_i + \hat{\beta}_i\hat{\xi}_i), \end{aligned}$$

y sus derivadas respecto a $\mathbf{w}$, ξ_i, $\hat{\xi}_i$ y b como:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} + \sum_{i=1}^{l}(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i = 0 \Rightarrow \mathbf{w} = -\sum_{i=1}^{l}(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0$$

$$\frac{\partial L}{\partial \hat{\xi}_i} = C - \hat{\alpha}_i - \hat{\beta}_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) = 0.$$

Finalmente, obtendremos el siguiente problema dual:

$$\begin{array}{ll} \max\limits_{\alpha} & \frac{1}{2}\sum_{i,j}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\mathbf{x}_i \cdot \mathbf{x}_j - \sum_i^l y_i(\hat{\alpha}_i - \alpha_i) + \sum_i^l \epsilon(\hat{\alpha}_i + \alpha_i) \\ \text{s.t} & \sum_i (\hat{\alpha}_i - \alpha_i) = 0 \\ & 0 \leq \alpha_i, \hat{\alpha}_i \leq C \end{array} \tag{1.2.16}$$

con sus correspondientes condiciones KKT.

$$\begin{gathered} \alpha_i^*[(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - y_i - \xi_i^* - \epsilon] = 0 \\ \alpha_i^*[y_i - (\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - \xi_i^* - \epsilon] = 0 \\ \alpha_i^* \geq 0 \\ \beta_i^* \geq 0 \\ \beta_i^* \xi_i^* = 0 \Rightarrow (C - \alpha_i^*)\ \xi_i^* = 0 \\ \hat{\beta}_i^* \hat{\xi}_i^* = 0 \Rightarrow (C - \hat{\alpha}_i^*)\ \hat{\xi}_i^* = 0. \end{gathered}$$

Por tanto, el valor de $\mathbf{w}$ se obtiene de la ecuación $\frac{\partial L}{\partial \mathbf{w}} = 0$, resultando la ecuación:

$$\mathbf{w}^* = \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i)\mathbf{x}_i$$

y el de b con las condiciones KKT.

1.2.4. Proyección a espacios de alta dimensión

Cuando el problema no es linealmente separable, muchas veces debido a que la dimensión del problema es baja, es necesario proyectar los datos originales a un espacio de dimensión superior o incluso infinita a través de una función ϕ, donde los datos sean linealmente separables. Por tanto, si el hiperplano separador es $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ en este caso se convierte en $f(\mathbf{x}) = \phi(\mathbf{w}) \cdot \mathbf{x} + b$, obteniéndose un separador lineal en el espacio proyectado para datos no linealmente separables en el espacio de entrada.

La figura 1.2.4 muestra un ejemplo sencillo de una proyección mediante una función $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$, es decir, datos en $\mathbb{R}^2$, donde no son linealmente separables a uno de dimensión 3 donde si lo son.

Calcular el vector proyectado de atributos puede ser computacionalmente muy costoso si la dimensión de los atributos es muy alta, es por ello que mediante la observación de que en la ecuación 1.2.8 sólo aparecen los datos de entrenamiento como un producto escalar, $\mathbf{x}_i \cdot \mathbf{x}_j$, podemos utilizar el truco del núcleo. Por tanto en lugar de guardar el vector proyectado completo y después hacer el producto escalar, mediante el uso del truco del núcleo podemos obtener directamente el producto de los vectores proyectados mediante una función de núcleo K, siendo la reserva de memoria menos costosa.

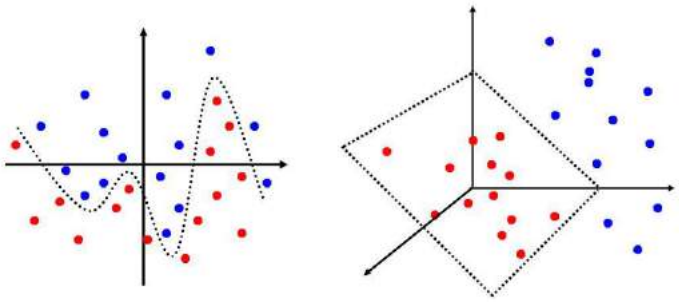

Figura 1.2.4: Proyección mediante una función $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$

Definicion 8 (Función de núcleo)**.** *Sea X el espacio de entrada, H el de características dotado de un producto interno y una función $F : X \longrightarrow H$, con H espacio inducido de Hilbert, se define la función de núcleo $K : X \times X \longrightarrow \mathbb{R}$ como:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j).$$

El problema con el que nos encontramos al usar esta técnica es que no cualquier función puede ser usada como núcleo, ya que no es posible encontrar una función K tal que $K(x, y) = \phi(x) \cdot \phi(y)$. El teorema de Mercer, sacado de**??** nos dará las condiciones que son necesarias para cumplir lo anterior.

Teorema 6 (Teorema de Mercer)**.** *Si una función escalar $k(\mathbf{x}_i, \mathbf{x}_j)$ es semidefinida positiva, existe una función*

$$\Phi : \mathbb{R}^d \longrightarrow F,$$

con F espacio de Hilbert, tal que $k(\mathbf{x}_i, \mathbf{x}_j)$ puede descomponerse como un producto escalar

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j).$$

Por tanto, necesitamos dar las definiciones de matriz y kernel semidefinidos positivos.

Definicion 9 (Matriz semidefinida positiva)**.** *Una matriz simétrica de tamaño $n \times n$ es semidefinida positiva si para todo vector $\mathbf{x} \in \mathbb{R}^n$ y $\mathbf{x} \neq 0$, tenemos que*

$$\mathbf{x}^T K \mathbf{x} \geq 0.$$

Definicion 10 (Kernel semidefinido positivo)**.** *Una función de kernel $k(x_i, x_j)$ es semidefinido positivo si para cualquier otra función $f \in L^2$ tenemos que:*

$$\int k(x_i, x_j) f(x_i) f(x_j) dx_i dx_j.$$

Para terminar vamos a listar algunos de los núcleos más utilizados:

- El núcleo lineal es el más sencillo, su expresión es;

$$k(x_i, x_j) = x_i \cdot x_j.$$

- El núcleo gausiano proyecta los vectores a un espacio de dimensión infinita y se define como;

$$k(x_i, x_j) = e^{\frac{-||x_i - x_j||^2}{2\sigma^2}}.$$

Ambos los usaremos en la experimentos númericos de la última sección.

1.2.5. Propiedades de las SVM

Las principales propiedades y ventajas de las Máquinas de Vectores Soporte y por las que su uso está tan extendido son las siguientes:

- Tiene una buena generalización con nuevos datos cuando el modelo está bien parametrizado.
- El proceso de entrenamiento no depende del número de atributos.
- Los modelos dependen de pocos parámetros, C, σ, ϵ. Por lo que la metamodelización es más fácil.
- Es un problema de optimización convexa cuadrático. Lo que significa que el mínimo es único.
- El modelo final suele ser sencillo, simplemente una combinación de un pocos vectores soporte.

1.3. Sequential Minimal Optimization (SMO)

En esta sección vamos a describir el algoritmo Sequential Minimal Optimization (SMO) [4, 8]. Cuando el número de datos que disponemos es muy grande, el problema cuadrático de las SVM puede ser difícil de resolver con las técnicas estándar descritas anteriormente.

El algoritmo SMO es un método de descomposición en que la idea es elegir subproblemas del problema inicial para que sea menos costoso computacionalmente. En el caso del problema de optimización cuadrática de las SVM implica al menos dos multiplicadores de Lagrange, ya que deben cumplir la restricción $\sum_{i=1}^{l} \alpha_i y_i = 0$, es decir, los subproblemas que elijamos deben tener al menos tamaño dos.

Podemos observar que la simplificación de SMO es que permite optimizar analíticamente. Por tanto, el algoritmo se compone de dos pasos; el primero es una técnica heurística para elegir los dos mejores multiplicadores; el segundo es una técnica analítica para optimizarlos.

Como hemos dicho en apartados anteriores, las máquinas de Vectores Soporte resuelven problemas tanto de clasificación como de regresión. En la mayoría de los trabajos, los algoritmos se explican desde el punto de vista de la clasificación, por

eso en este trabajo vamos a introducir una notación sacada de [4] que nos permita estudiar el algoritmo tanto para clasificación como para regresión.

Si tenemos la muestra $S = \{(\mathbf{x}_i, y_i) : i = 1, ..., N\}$ donde $y_i = \pm 1$ para clasificación y $R = \{(\mathbf{x}_i, t_i) : i = 1, ..., N\}$ donde $t_i \in \mathbb{R}$ para regresión, el problema que queremos optimizar será el siguiente:

$$\begin{array}{rc} P(\mathbf{w}, b, \xi) = \min\limits_{w,b} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t} & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq p_i - \xi_i \quad \forall i \\ & \xi_i \geq 0 \forall i \end{array} \tag{1.3.1}$$

Cuando p_i es 1 tenemos el clásico problema de SVM para clasificación.

Para problemas regresión redefinimos la muestra R como el triplete $S_R = \{(\mathbf{x}_i, y_i, pi) : i = 1, ..., 2N\}$, donde.

- Para todo $i = 1, ..., N$, tenemos que $y_i = 1$ y $p_i = t_i - \epsilon$.
- Para todo $i = N + 1, ..., 2N$, tenemos que $y_i = -1$ y $p_i = -t_i - \epsilon$.

Con lo que finalmente su correspondiente problema dual será:

$$\begin{array}{rc} \min\limits_{\alpha} & \mathcal{D}(\alpha) = \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j - \sum_i \alpha_i p_i \\ \text{s.t} & \alpha_i \geq 0 \quad \forall i \\ & \sum_i \alpha_i y_i = 0 \end{array} \tag{1.3.2}$$

Como hemos dicho anteriormente para aplicar SMO debemos actualizar al menos dos coeficientes en cada iteración, estos los denotaremos como L y U. Si queremos elegir el óptimo de L^t y U^t en la iteración t, sus coeficientes en la siguiente serán;

$$\begin{aligned} \alpha_{L^t}^{t+1} &= \alpha_{L^t}^t + \delta_{L^t}^t \\ \alpha_{U^t}^{t+1} &= \alpha_{U^t}^t + \delta_{U^t}^t \end{aligned}$$

mientras que el resto de los coeficientes α_j permanecen iguales.

Debido a la restricción $\sum_i \alpha_i y_i = 0$, la actualización de uno de los multiplicadores puede ser escrita en función del otro, de tal manera que: $\delta_{U^t}^t = -y_{U^t} y_{L^t} \delta_{L^t}^t$

Para elegir los índices U^t y L^t, se usa el criterio conocido en inglés como the most violating pair (MVP), es decir U^t y L^t viene determinado por las siguientes ecuaciones.

$$\begin{aligned} U^t &= \arg\max_{i\in\mathcal{I}_U^t}\{y_i\nabla\mathcal{D}_i^t\} \\ L^t &= \arg\min_{i\in\mathcal{I}_L^t}\{y_i\nabla\mathcal{D}_i^t\} \end{aligned} \tag{1.3.3}$$

donde

$$I_L = \{i : y_i = +1 \wedge \alpha_i < C\} \cup \{i : y_i = -1 \wedge \alpha_i > 0\}$$

$$I_U = \{i : y_i = +1 \wedge \alpha_i > 0\} \cup \{i : y_i = -1 \wedge \alpha_i < C\}$$

Si medimos la diferencia $\mathcal{D}$ entre dos iteraciones del algoritmo SMO tenemos que:

$$\mathcal{D}^t - \mathcal{D}^{t+1} = \delta^t_{L^t} y_{L^t} \Delta^t - \frac{(\delta^t_{L^t})^2}{2} \|\mathbf{x}_{L^t} - \mathbf{x}_{U^t}\|^2 = \psi(\delta^t_{L^t}) \tag{1.3.4}$$

siendo $\Delta^t = \mathbf{w}(\mathbf{x}_U - \mathbf{x}_L)$
El objetivo es conseguir la máxima diferencia entre dos iteraciones; esto es fácil de calcular mediante la derivada $\psi' = 0$ donde ψ es en la ecuación (1.3.4). Por tanto, obtenemos;

$$\overline{\delta}^t_{L^t} = y_{L^t} \frac{(y_{U^t} \nabla \mathcal{D}_{U^t}) - (y_{L^t} \nabla \mathcal{D}_{L^t})}{\|\mathbf{x}_{L^t} - \mathbf{x}_{U^t}\|^2} = y_{L^t} \frac{\Delta^t}{\|\mathbf{x}_{L^t} - \mathbf{x}_{U^t}\|^2}$$

Si sustituimos la ecuación (1.3.5) en (1.3.4), obtenemos:

$$\mathcal{D}^t - \mathcal{D}^{t+1} = \frac{(y_{U^t} \nabla \mathcal{D}_{U^t} - y_{L^t} \nabla \mathcal{D}_{L^t})^2}{2\|\mathbf{x}_{L^t} - \mathbf{x}_{U^t}\|^2} = \frac{(\Delta^t)^2}{2\|\mathbf{x}_{L^t} - \mathbf{x}_{U^t}\|^2} \tag{1.3.5}$$

Por tanto, sin tener en cuenta el denominador, los índices elegidos por la regla MVP ,L^t y U^t, obtenemos el máximo valor $\mathcal{D}^t - \mathcal{D}^{t+1}$, $\mathcal{D}^t$ decrece en cada iteración, como queríamos. Es decir la regla MVP coincide con la maximización de $\mathcal{D}^t - \mathcal{D}^{t+1}$.

Capítulo 2

Algoritmos de Máquinas de Vectores Soporte para grandes datos

La elección de este tema está motivada por el creciente interés en la investigación de técnicas que nos permitan trabajar con grandes volúmenes de datos y alta dimensionalidad. Cada día más la cantidad de información disponible aumenta vertiginosamente, aumentando a su vez tanto el número de patrones como la dimensionalidad de los mismos. Por tanto es importante desarrollar más técnicas potentes en este campo.

En esta sección trataremos de discutir las ventajas y desventajas de cada uno de los métodos que vamos a estudiar, estos son; Dual Coordinate Descent method (DCD) [9] y Stochastic Sub-gradient Descent for Primal Problem (Pegasos) [6]. DCD es un método batch que trata mediante un método iterativo de descenso coordinado de encontrar la solución en el problema dual. En cambio Pegasos es un método online que resuelve el problema primal y en el que se alternan dos pasos, un método de descenso por subgradiente aproximado y una proyección sobre un subconjunto.

La diferencia entre el aprendizaje batch y online es que en el aprendizaje batch el gradiente se calcula sobre el error muestral global; es decir, sobre toda la muestra En cambio en el aprendizaje online se calcula un gradiente local sobre el error muestral local; es decir, patrón a patrón. La ventaja del aprendizaje batch frente al online es que las condiciones de convergencia se entienden mejor. En cambio la ventaja del aprendizaje online es que generalmente es más rápido.

Nuestro objetivo, por tanto es encontrar un método eficiente, es decir, que el tiempo empleado para entrenar no sea muy alto y además tengamos una precisión razonable. Como hemos dicho antes, en esta sección vamos a estudiar dos de los principales algoritmos de las Máquinas de Vectores Soporte, Dual Coordinate Descent Method (DCD) y Stochastic Sub-gradient Descent for Primal Problem (Pegasos).

2.1. El método Dual Coordinate Descent (DCD)

Como ya se ha anunciado anteriormente, los problemas que encontramos en la vida real no tienen datos linealmente separables, haciendo necesario proyectarlos

mediante una función no lineal $\phi(x)$ a un espacio de dimensionalidad superior. Sin embargo, se ha observado que cuando los datos están en un espacio de atributos de alta dimensionalidad, los resultados del algoritmo son parecidos con y sin proyección, con la ventaja de que el tiempo de entrenamiento se reduce considerablemente. El método que vamos a describir a continuación, Dual Coordinate Descent [9], parte de esa observación para implementar su algoritmo.

Se trata de un algoritmo de descenso por coordenadas, que en cada iteracción minimiza un subproblema de una única variable; además resuelve el problema dual.

2.1.1. Algoritmo

DCD puede ser utilizado tanto en clasificación como en regresión y en nuestros experimentos vamos a utilizarlo en los dos tipos de problemas. Aunque son parecidos, explicaremos el algoritmo para ambos casos.

Clasificación

Nuevamente el problema que queremos resolver es el problema clásico de las Máquinas de Vectores Soporte para clasificación y cuya formulación primal nos es conocida. Por tanto, tenemos la función objetivo 2.1.1

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi(\mathbf{w};\mathbf{x}_i,y_i) \tag{2.1.1}$$

sujeta a ciertas restricciones y donde $\xi(\mathbf{w};\mathbf{x}_i,y_i)$ es la función de pérdida.

Tras una serie de cálculos, que hemos desarrollado en el capítulo 1 mediante la teoría lagrangiana obtenemos el dual de la ecuación del problema (2.1.1) como:

$$\begin{aligned} &\min_{\alpha} f(\alpha) = \tfrac{1}{2}\alpha^T\hat{Q}\alpha - \mathbf{e}^T\alpha \\ \text{s.t}\quad &0 \le \alpha_i \le U \quad \forall i, \end{aligned} \tag{2.1.2}$$

donde $\hat{Q} = Q + D$, con D matriz diagonal y $U = C$, $D_{ii} = 0$ en L1-SVM. $Q_{i,j} = y_i y_j \mathbf{x}_i^T\mathbf{x}_j$.

El algoritmo Dual Coordinate Descend (DCD) empieza con un punto inicial $\alpha^0 \in \mathbb{R}^l$, tras lo que se genera una secuencia de vectores $\{\alpha^k\}_{k=0}^{\infty}$.

Por tanto, en cada iteracción del algotimo necesitamosactualizar $\alpha^{k,i}$ a $\alpha^{k,i+1}$, siendo $\alpha^{k,i} = [\alpha_1^{k+1},...,\alpha_{i-1}^{k+1},\alpha_i^k,...,\alpha_l^k]$, resolveremos el subproblema de una variable:

$$\begin{aligned} &\min_{d} f(\alpha^{k,i} + de_i) \\ &\text{s.t } 0 \le \alpha^{k,i} + d \le U, \end{aligned} \tag{2.1.3}$$

donde $e_i = [0,..,0,1,0,..,0]^T$.

En este caso no tenemos la restricción $\sum \alpha_i x_i y_i = 0$ como en SMO, y por tanto el tamaño mínimo del subproblema elegido puede ser uno.

Podemos observar como la función objetivo del problema 2.1.3 se puede reescribir como una función cuadrática de d:

$$f(\alpha^{k,i} + de_i) = \frac{1}{2}\hat{Q}_{ii}d^2 + \nabla_i f(\alpha^{k,i})d + cte, \tag{2.1.4}$$

donde $\nabla_i f$ es la componente i del gradiente total de f, ∇f.

Por tanto, si derivamos e igualamos a cero la ecuación (2.1.4), observamos que tiene un mínimo en $d = -\frac{\nabla_i f(\alpha^{k,i})}{\hat{Q}_{ii}}$ y la actualización en cada paso puede ser expresada como sigue:

$$\alpha_i^{k,i+1} = \text{mín}(\text{máx}(\alpha_i^{k,i} - \frac{\nabla_i f(\alpha^{k,i})}{\hat{Q}_{ii}}, 0), U),$$

si $\hat{Q}_{ii} > 0$. Donde usamos el máximo y el mínimo para asegurarnos que $0 \leq \alpha_i^{k,i+1} =\leq C$.

Por tanto el algoritmo se puede describir como sigue a continuación:

Algorithm 1 DCD

Dado α y su correspodiente $\mathbf{w} = \sum_i y_i \alpha_i \mathbf{x}_i$.
Mientras α no sea óptimo:
for $n = 1 \ldots l$ **do**
(1) $G = y_i \mathbf{w}_i \mathbf{x}_i - 1 + D_{ii}\alpha_i$
(2)

$$PG = \begin{cases} \text{mín}(G, 0) & \text{si } \alpha_i = 0 \\ \text{máx}(G, 0) & \text{si } \alpha_i = U \\ G & \text{si } 0 < \alpha_i < U \end{cases}$$

(3) Si $|PG \neq 0|$,
$\hat{\alpha}_i \leftarrow \alpha_i$
$\alpha_i \leftarrow \text{mín}(\text{máx}(\alpha_i - \frac{G}{\hat{Q}_{ii}}, 0), U)$
$\mathbf{w} \leftarrow \mathbf{w} + (\alpha_i - \hat{\alpha}_i) y_i \mathbf{x}_i$
end for

En resumen, los principales cálculos que debemos hacer para implementar el algoritmo son $\hat{Q}_{ii}$ y $\nabla_i f(\alpha^{k,1})$, que vienen dados por las fórmulas 2.1.5 y 2.1.6 respectivamente:

$$\hat{Q}_{ii} = \mathbf{x}_i^T \mathbf{x}_i \tag{2.1.5}$$

$$\nabla_i f(\alpha^{k,1}) = (\hat{Q}\alpha)_i - 1 = \sum_{j=1}^{l} \hat{Q}_{ij}\alpha_j - 1 = \sum_{j=1}^{l} y_i y_j \mathbf{x}_i \mathbf{x}_j \alpha_j - 1. \tag{2.1.6}$$

Si $\hat{n}$ es el número de elementos no cero, el coste computacional para hacer cada evaluación del núcleo es $O(\hat{n})$, entonces el coste para obtener la componente fila i de la matriz de núcleos será $O(l\hat{n})$.

Este coste computacional puede ser reducido en el caso de usar el núcleo lineal, ya que definimos $\mathbf{w}$ como:

$$\mathbf{w} = \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j, \tag{2.1.7}$$

y por tanto el gradiente de la fórmula 2.1.6 viene dado por la fórmula 2.1.8:

$$\nabla f(\alpha) = y_i \mathbf{w}^T \mathbf{x}_i - 1 + D_{ii}\alpha_i. \tag{2.1.8}$$

Por tanto, si $\hat{n}$ es de nuevo el número de elementos no cero por patrón, el coste principal del algoritmo para actualizar α_i será evaluar la ecuación 2.1.8; este coste es $O(\hat{n})$ Además, para calcular 2.1.8 debemos mantener el gradiente respecto a $\mathbf{w}$ y por tanto necesitaremos usar la ecuación 2.1.7, siendo el coste total $O(l\hat{n})$.

Sin embargo, si el valor actual es $\hat{\alpha}_i$ y queremos obtener α_i, podemos mantener $\mathbf{w}$ con la siguiente ecuación:

$$\mathbf{w} \leftarrow \mathbf{w} + (\alpha_i - \hat{\alpha}_i) y_i \mathbf{x}_i \tag{2.1.9}$$

Por tanto, el coste computacional para mantener $\mathbf{w}$ por 2.1.9 se reduce a $O(\hat{n})$ y el número de operaciones es $O(\hat{n})$ para calcular 2.1.8. Finalmente, el coste por iteración, es decir, pasar de α^k a α^{k+1} es $O(l\hat{n})$.

Regresión

Como hemos explicado en la sección previa de SVR muchos problemas del mundo real no son de clasificación sino de regresión y por tanto es importante obtener algoritmos que mejoren nuestros resultados para este tipo de problemas. Por esta razón vamos a desarrollar la teoría dada para DCD en clasificación para problemas de regresión. La idea para obtener el algoritmo DCD para regresión es tomada de [10].

Como en el caso anterior necesitamos resolver el siguiente problema de SVR.

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \left(\xi(\mathbf{w}; \mathbf{x}_i, y_i) + \hat{\xi}(\mathbf{w}; \mathbf{x}_i, y_i)\right) \tag{2.1.10}$$

sujeto a algunas restricciones donde $\xi(\mathbf{w}; \mathbf{x}_i, y_i)$ and $\hat{\xi}(\mathbf{w}; \mathbf{x}_i, y_i)$ son las funciones de pérdida. Para la ecuación (2.1.10) el problema dual es:

$$\begin{array}{ll} \max\limits_{\alpha} & \frac{1}{2}\sum_{i,j}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\mathbf{x}_i \cdot \mathbf{x}_j - \sum_i^l y_i(\hat{\alpha}_i - \alpha_i) + \sum_i^l \epsilon(\hat{\alpha}_i + \alpha_i) \\ \text{s.t} & \sum_i (\hat{\alpha}_i - \alpha_i) = 0 \\ & 0 \leq \alpha_i, \hat{\alpha}_i \leq C, \end{array} \tag{2.1.11}$$

Podemos simplificar la notación del problema dual de SVR 2.1.11 mediante $\beta_i = \alpha_i - \hat{\alpha}_i$ de la siguiente manera:

$$\begin{array}{ll} \min\limits_{\beta} f(\beta) = \frac{1}{2}\beta^T \hat{Q}\beta - y^T\beta + \sum |\beta_i| \\ \text{s.t} \qquad\qquad -C \leq \beta_i \leq C \quad \forall i, \end{array} \tag{2.1.12}$$

donde cada β_i viene dado por $\beta_i = \alpha_i - \hat{\alpha}_i$ y $\sum_i^l y_i(\hat{\alpha}_i - \alpha_i) = \sum_i^l |\beta_i|$ por una propiedad del óptimo del problema dual en las Máquinas de Vectores Soporte para regresión que satisface:

$$\alpha_i \hat{\alpha}_i = 0 \forall i. \tag{2.1.13}$$

ya que si α_i o $\hat{\alpha}_i$ es positivo en el óptimo, entonces el otro coeficiente sea cero al final de las iteraciones.

Por tanto, por la ecuación 2.1.13 y $\alpha_i, \hat{\alpha}_i \geq 0$ tenemos que en el óptimo:

$$\alpha_i + \hat{\alpha}_i = |\alpha_i - \hat{\alpha}_i|,$$

lo que se demuestra fácilmente comoc sigue:

$$\alpha_i \hat{\alpha}_i = 0 \Rightarrow \begin{cases} \alpha_i \geq 0 \wedge \hat{\alpha}_i = 0 & \text{si } |\alpha_i - \hat{\alpha}_i| = |\alpha_i| = \alpha_i = \alpha_i + \hat{\alpha}_i \\ \hat{\alpha}_i \geq 0 \wedge \alpha_i = 0 & \text{si } |\alpha_i - \hat{\alpha}_i| = |-\hat{\alpha}_i| = \hat{\alpha}_i = \alpha_i + \hat{\alpha}_i. \end{cases}$$

Igual que en clasificación, el método Dual Coordinate Descend (DCD) empieza en un punto inicial $\beta^0 \in \mathbb{R}^l$ y genera una secuencia de vectores $\{\beta^k\}_{k=0}^{\infty}$. Para actualizar $\beta^{k,i}$ a $\beta^{k,i+1}$, siendo $\beta^{k,i} = [\beta_1^{k+1}, ..., \beta_{i-1}^{k+1}, \beta_i^k, ..., \beta_l^k]$, resolveremos el subproblema de una variable:

$$\begin{array}{l} \min\limits_{d} f(\beta^{k,i} + de_i) \\ \text{s.t} \ -U \leq \beta^{k,i} + d \leq U \end{array} \tag{2.1.14}$$

donde $e_i = [0, ..,0, 1, 0, ..,0]^T$.

Además la función objetivo puede ser expresada como:

$$f(\beta^{k,i} + de_i) = \frac{1}{2}\hat{Q}_{ii}d^2 + \nabla_i f(\beta^{k,i})d + cte. \tag{2.1.15}$$

Por tanto, si $\hat{Q}_{ii} > 0$, la ecuación (2.1.15) tiene un mínimo en $d = -\frac{\nabla_i f(\beta^{k,i})}{\hat{Q}_{ii}}$ y la actualización en cada paso puede ser expresada como sigue:

$$\beta_i^{k,i+1} = \text{mín}(\text{máx}(\beta_i^{k,i} - \frac{\nabla_i f(\beta^{k,i})}{\hat{Q}_{ii}}, 0), U).$$

El algoritmo final para regresión en DCD es similar al obtenido para clasificación, ya es igual salvo por que en lugar de α_i tenemos $\beta_i = \alpha_i - \hat{\alpha}_i$.

2.1.2. Extensiones

Algunas extensiones del algoritmo pueden mejorar la convergencia del mismo. Estas son:

1. Permutación aleatoria de los subproblemas.
2. Contracción de las variables.
3. Aprendizaje online.

Permutación aleatoria de los subproblemas

Algunos resultados, como los obtenidos en [12], muestran que resolver subproblemas del problema total en un orden arbitrario mejora la convergencia, por esto permutamos los subíndices $1, ..., l$ a $\pi(1), ..., \pi(l)$.

El algoritmo es muy similar, generando una secuencia de vectores $\alpha^{k,i}$ con:

$$\alpha_t^{k,i} = \begin{cases} \alpha_t^{k+1} & \text{si } \pi^{-1}(t) < i \\ \alpha_t^k & \text{si } \pi^{-1}(t) \geq i. \end{cases}$$

Por tanto, simplemente debemos cambiar actualización de $\alpha^{k,i}$ a $\alpha^{k,i+1}$ por la fórmula:

$$\alpha_t^{k,i+1} = \alpha_t^{k,i} + \arg \min_{0 \leq \alpha_t^{k,i} + d \leq U} f(\alpha^{k,i} + de_i) \text{ si } \pi_k^{-1}(t) = i.$$

Contracción o "Shrinking"

Si α_i es 0 o U en varias iteraciones, el valor de α_i se mantendrá en el mismo valor. La técnica de shrinking reduce el tamaño del problema de optimización eliminando estas variables que están en el borde y valen 0 o U, ya que es muy probable que se matengan en el mismo valor.

Aprendizaje online

En algunas aplicaciones, el número de patrones es alto, por tanto ir a lo largo de todos los α_i con $i = 1, ..., l$ tiene un coste computacional alto en cada iteración exterior.

En su lugar, podemos elegir un índice aleatorio i_k y hacer una actualización sólo de α_{ik} en la k-th iteración exterior.

2.1.3. Comparación con métodos previos

Los métodos de descenso coordinado están relacionados con los métodos de descomposición para SVM no lineales. En esta sección vamos a mostrar las diferencias principales con estos métodos.

Métodos de descomposición para SVM no lineales

Los métodos de descomposición son una de las técnicas más populares para SVM no lineales con un gran volumen de datos. Resuelven sub-problemas de pocas variables del problema total, por lo que sólo necesitamos un pequeño número de columnas de la matriz de núcleos. En los primeros métodos de descomposición, las variables elegidas en cada iteración para optimizar se escogen mediante la técnica heurística de MVP, most violating pair, como en SMO.

Tras algunas mejoras se decidió usar la información del gradiente para hacer la selección de las variables. La ventaja es que el gradiente nos da información de la dirección de descenso, además mantener el gradiente completo no supone un coste computacional extra, como se discute a continuación.

Para resolver el subproblema 2.1.4, usaremos la ecuación 2.1.6 para calcular $\nabla_i f(\alpha)$. Si $\hat{n}$ es el número de elementos no cero, el coste computacional para hacer cada evaluación del núcleo es $O(\hat{n})$, y entonces el coste para obtener la componente fila i de la matriz de núcleos será $O(l\hat{n})$. Por otro lado, después de actualizar $\alpha_i^{k,i}$ a $\alpha_i^{k,i+1}$ obtenemos $\hat{Q}_{:,i}$, por lo que el nuevo gradiente se calcula con la siguiente ecuación:

$$\nabla f(\alpha^{k,i+1}) = \nabla f(\alpha^{k,i}) + \hat{Q}_{:,i}(\alpha_i^{k,i+1} - \alpha_i^{k,i}), \tag{2.1.16}$$

donde $\hat{Q}_{:,i}$ es la i-ésima columna de $\hat{Q}$, siendo el coste computacional de calcular $\hat{Q}_{:,i}$ $O(l\hat{n})$ y el de 2.1.16 $O(l)$. Por tanto, mantener el gradiente completo de Q no supondrá un coste computacional extra, pero sí tiene la ventaja de que el número de iteraciones es menor, es decir, tenemos una convergencia más rápida debido al uso de la información del gradiente para elegir las variables de los subproblemas.

Recordemos que hemos comenzado el capítulo observando que la ventaja de datos de grandes dimensiones, es que estos suelen dar los mismos resultados con o sin proyección, y por tanto no es necesario usar el truco del núcleo para proyectarlo a un espacio de dimensión superior, lo que supone una parte importante del coste computacional.

Por tanto para el núcleo lineal, el coste principal del algoritmo para actualizar α_i será evaluar la ecuación 2.1.8, que es $O(\hat{n})$ de calcular el producto escalar $\mathbf{w}^T\mathbf{x}_i$. Si queremos mantener el gradiente $\nabla_i f(\alpha)$ necesitaremos evaluar la ecuación 2.1.8 l veces siendo el coste total $O(l\hat{n})$.

Si comparamos los métodos previos de descomposición y DCD podemos deducir que para SVM no lineales es mejor usar métodos de descomposición que actualicen el gradiente completo. Para SVM lineales, con l grande, el coste por iteración es más pequeño si no mantenemos el gradiente completo, siendo DCD más rápido que los métodos clásicos de descomposición en estos casos.

Métodos lineales

Los métodos lineales están relacionados a los métodos estocásticos de descenso por gradiente mediante la siguiente fórmula;

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}}(\mathbf{x}_i, y_i),$$

donde $\nabla_{\mathbf{w}}(\mathbf{x}_i, y_i)$ es el subgradiente aproximado de la función objetivo y η es la tasa de aprendizaje.Por tanto la actualización es muy similar a la ecuación de DCD 2.1.9.

Ambos algoritmos usan un sólo patrón $\mathbf{x}_i$, pero toman diferentes direcciones que son $y_i\mathbf{x}_i$ en el caso de DCD y $\nabla_w(y_i, \mathbf{x}_i)$ para los métodos de descenso estocástico por gradiente.

Por tanto, podemos ver DCD como un tipo de método estocástico de descenso. La ventaja está en que no es necesario elegir la tasa de aprendizaje, que viene dada por la ecuación 2.1.9 y que suele ser un problema, ya que si es demasiado grande puede que sobrepasemos el óptimo, y por otro lado, si es muy pequeña, el paso que daremos en cada iteración será muy pequeño y el algoritmo será muy lento.

POr último si definimos la ϵ-precisión como:

Definicion 11 (ϵ-precisión)**.** *Si α^* es un mínimo de f, una solución α tiene ϵ-precisión si:*

$$f(\alpha) \leq f(\alpha^*) + \epsilon$$

es decir, la diferencia con el óptimo es menor que ϵ.

La tasa de convergencia del algoritmo DCD para obtener una solución con ϵ-precisión es $O(log(\frac{1}{\epsilon}))$. Para demostrar esta tasa de convergencia necesitaremos un teorema que enunciamos y demostramos a continuación.

2.2. Pegasos

Pegasos [6] es un algoritmo que alterna un método estocástico de descenso por subgradiente estimado (i.e. el gradiente elegido de un subconjunto) en el problema primal, con un paso de proyección (i.e. una proyección sobre un subconjunto). El algoritmo realiza T iteracciones y usa un subconjunto k de patrones del conjunto total S en cada iteración para estimar el subgradiente.

2.2.1. Algoritmo

Como en las secciones anteriores, sea S el conjunto de entrenamiento con $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ donde $\mathbf{x}_i \in \mathbb{R}^n$ y $y_i \in [-1, 1]$, la función objetivo del problema de optimización primal es:

$$\min_{\mathbf{w}} \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{m}\sum_{(x,y)\in S} l(\mathbf{w}; (\mathbf{x}, y)), \tag{2.2.1}$$

donde $l(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - (y\mathbf{w} \cdot \mathbf{x})\}$ sujeta a algunas restricciones.

Inicialmente, escogeremos $\mathbf{w}_1$, siendo cualquier vector cuya norma valga al menos $\frac{1}{\sqrt{\lambda}}$, donde λ es el parámetro de regularización de las SVM. Este parámetro será de una de las entradas del algoritmo junto con T y k, que son el número de iteraciones del algoritmo y el número de patrones utilizados para calcular el subgradiente respectivamente.

En cada iteración del algoritmo primero elegiremos un subconjunto $A_t \subseteq S$ de tamaño k, tal que reemplazaremos la función objetivo del problema (2.2.1) por la función objetivo aproximada:

$$f(\mathbf{w}, A_t) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{k}\sum_{(x,y)\in A_t} l(\mathbf{w}; (\mathbf{x}, y)).$$

Luego definimos la tasa de aprendizaje como $\eta_t = \frac{1}{\lambda t}$ y A_t^+ como el conjunto de patrones para los que $\mathbf{w}$ tiene una función de pérdida cero.

Finalmente escalamos $\mathbf{w}_t$ con $(1 - \eta_t\lambda)$ y para todo $(x, y) \in A_t^+$ añadimos a $\mathbf{w}$ el vector $\frac{y\eta_t}{k}x$. Por tanto el paso puede ser reescrito como:

$$\mathbf{w}_{t+1/2} = \mathbf{w}_t - \eta_t\nabla_t, \tag{2.2.2}$$

donde ∇_t se define como:

$$\nabla_t = \lambda\mathbf{w}_t - \frac{1}{|A_t|}\sum_{(x,y)\in A_t^+} y\mathbf{x}, \tag{2.2.3}$$

obteniendo $\mathbf{w}_{t+1/2}$ como:

$$\mathbf{w}_{t+1/2} = (1 - \eta_t\lambda)\mathbf{w}_t + \frac{\eta_t}{k}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x}.$$

Por último proyectamos $\mathbf{w}_{t+1/2}$ en el subconjunto:

$$B = \{\mathbf{w} : ||\mathbf{w}|| \leq \frac{1}{\sqrt{\lambda}}\} \tag{2.2.4}$$

es decir, $\mathbf{w}_{t+1}$ se obtiene escalando $\mathbf{w}_{t+1/2}$ con mín$\{1, 1/(\sqrt{\lambda}||\mathbf{w}_{t+1/2}||)\}$.

Por tanto el algoritmo se puede describir como en el pseudo código 2.2.1:

Algorithm 2 Pegasos

Input S, λ, T y k.
Elegimos $\mathbf{w}_1$ tal que $||\mathbf{w}_1 \leq \frac{1}{\lambda}$:
for $n = 1 \dots T$ **do**
(1) Elegimos $A_t \subseteq S$, donde $|A_t| = k$.
(2) Definimos $A_t^+ = \{(\mathbf{x}, y) \in A_t : y\mathbf{w}_t\mathbf{x} < 1\}$
(3) $\eta_t = \frac{1}{\lambda t}$
(4) $\mathbf{w}_{t+1/2} = (1 - \eta_t\lambda)\mathbf{w}_t + \frac{\eta_t}{k}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x}$
(5) $\mathbf{w}_{t+1} = \text{mín}\{1, \frac{1/\sqrt{\lambda}}{||\mathbf{w}_{t+1/2}||}||\}\mathbf{w}_{t+1/2}$
end for
Output $\mathbf{w}_{T+1}$

Finalmente observamos, que si elegimos A_t como toda la muestra S en cada iteración tendremos un método de subgradiente proyectado. Por otro lado, si escogemos A_t como un solo patrón, obtendremos una variante de los métodos estocásticos de descenso por gradiente.

2.2.2. Extensiones

Como todos los algoritmos para grandes volúmenes de datos, las primeras implementaciones se reducen a problemas donde no tenemos término de bias y usamos el kernel lineal. Estas suposiciones se permiten porque en un espacio de dimensionalidad alta el término de bias es despreciable y el algoritmo funciona más o menos igual con o sin proyección. En este apartado vamos a ampliar el algoritmo para estos dos casos:

1. Incorporar el término de bias.
2. Uso de núcleo.

Incorporar el término de bias

El término de bias tiene una relevancia especial cuando las clases están desequilibradas. Para solucionar este problema existen tres maneras para incorporar el término de bias.

1. La primera de ellas es la más sencilla y simplemente añade una característica más que será constante a cada patrón $\mathbf{x}$, por tanto si la dimensión de nuestros patrones era n, se aumentará $n + 1$.

   La desventaja de este método es que entonces nuestra función a minimizar cambiará ligeramente a:

   $$||\mathbf{w}||^2 + b^2.$$

   Sin embargo, la ventaja es que no será necesario modificar el algoritmo.

2. En la segunda introducimos b en la función de pérdida sin penalizar por b, obteniendo la siguiente ecuación:

   $$l((\mathbf{w}, b); (\mathbf{w}, b)) = \text{máx}\{0, 1 - y(\mathbf{w}\mathbf{x} + b)\}.$$

   La ventaja será que $\mathbf{w}$ permanece intacto y el subgradiente respecto a b es fácil de calcular. La desventaja es que entonces f deja de ser fuertemente convexa, como queríamos, y se convierte en una función lineal por trozos en la dirección de b.

3. Por último, el otro modo, y más complicado, es usar la siguiente función:

   $$\min_w \frac{\lambda}{2}||\mathbf{w}||^2 + g(\mathbf{w}; S),$$

   donde $g(\mathbf{w}; S) = \text{mín}_b \frac{1}{m} \sum_{(x,y)\in S} [1 - (y\mathbf{w} \cdot \mathbf{x} + b)]_+$.

   En cada iteración del algoritmo tratamos de buscar el subgradiente de $f(\mathbf{w}; A_t)$ y sustraerlo (multiplicado por η_t) de $\mathbf{w}_t$, proyectando el vector resultante. El problema es como encontrar el subgradiente $g(\mathbf{w}; A_t)$, ya que éste está definido a través de un problema de minimización sobre b.

Uso de núcleos de Mercer

Una de las ventajas de SVM es su habilidad de tratar problemas no lineales mediante el uso de núcleos que satisfacen las condiciones de Mercer, explicadas en el capítulo 1. Para que esto sea posible debemos resolver el problema dual cuya solución óptima $\mathbf{w}^*$ se expresa como una combinación de los $\mathbf{x}_i$.

En esta ampliación del algoritmo usamos el truco del núcleo a pesar de resolver el problema primal mediante la observación siguiente:

Si inicializamos $\mathbf{w}_1$ como un vector de ceros, en cada iteración del algoritmo tenemos que;

$$\mathbf{w}_t = \sum_{i \in I_t} \alpha_i \mathbf{x}_i$$

I_t es un subjunto de $\{1, ..., m\}$, entonces si calculamos

$$\mathbf{w}_t \cdot \mathbf{x}_t = \sum_{i \in I_t} \alpha_i \mathbf{x}_i \cdot \mathbf{x}_t$$

y evaluamos la norma de $\mathbf{w}_t$, tenemos:

$$||\mathbf{w}||^2 = \sum_{i \in I_t} \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j,$$

y por tanto, podemos aplicar el truco del núcleo.

Por último, la convergencia de Pegasos a una solución con ϵ-precisión será $\tilde{O} = (\frac{1}{\lambda \epsilon})$. Para demostrar esta convergencia del algoritmo necesitaremos una serie de lemas y teoremas que enunciaremos y demostraremos a continuación:

Capítulo 3

Experimentos Numéricos

3.1. Introducción

En esta sección vamos a realizar experimentos con problemas de clasificación y regresión mediante técnicas de aprendizaje automático, en concreto Support Vector Machines (SVM), tanto para clasificación (Support Vector Classification, SVC) como para regresión (Support Vector Regression, SVR). Nuestro objetivo será comparar los algoritmos estudiados en los capítulos 1 y 2 para SVM, es decir, determinar un algoritmo eficiente con una precisión razonable.

Para ello, vamos a tratar dos tipos de problemas. En primer lugar estudiaremos un problema relacionado con la predicción de energía solar, es decir, nos encontramos ante un problema de regresión, cuyo objetivo es mejorar las predicciones dadas por el European Centre for Medium-Range Weather Forecasts (ECMWF). El segundo problema tratará sobre la clasificación de páginas web por temas. La elección de estos dos temas se justifica por el creciente interés tanto en la predicción de energías renovables, siendo las energías del futuro, como en clasificación de páginas web, debido al inmenso crecimiento de la web y su información, que hace necesaria una buena clasificación por temas.

Para realizar el estudio vamos a comparar tres enfoques algorítmicos diferentes. Dos de ellos los usaremos en el problema de regresión; éstos son Sequential Minimal Optimization (SMO), algoritmo que está detrás de la librería LIBSVM [1] y un método de descenso dual coordinado, Dual Coordinate Descent method (DCD), algoritmo usado por la librería LIBLINEAR [2]. Por último para clasificación vamos a comparar estos dos algoritmos junto con un método estocástico de descenso por subgradiente, Stochastic sub-gradient Descent method, que es el implementado por Pegasos [3].

Compararemos LIBSVM vs LIBLINEAR en problemas de regresión y estos dos con Pegasos para clasificación. Por último obtener un error pequeño o una buena precisión será otro de nuestros objetivos; para ello realizaremos una metamodelización de los parámetros de los diferentes modelos, que se espera que mejore resultados.

[1] LIBSVM: A library for support vector machine. Chang, Chih-Chung and Lin, Chih-Jen. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[2] LIBLINEAR: A Library for Large Linear Classification. Rong-En Fan and Kai-Wei Chang and Cho-Jui Hsieh and Xiang-Rui Wang and Chih-Jen Lin

[3] A Library for Primal Estimated sub-GrAdient SOlver for SVM. Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. http://www.cs.huji.ac.il/ shais/code/

3.2. Predicción de radiación en estaciones individuales

El primer problema que queremos resolver es de predicción de radiación solar. Nuestro objetivo es mejorar las predicciones dadas por ECMWF en 2 puntos concretos de la Península Ibérica, situados en Alicante y A Coruña. Hemos elegido estas dos localidades por tratarse de puntos de la Península con diferentes climas. En el tema de la energía solar la nubosidad juega un papel muy importante, siendo más complicado la predicción de aquellos puntos en los que hay más nubosidad, como en nuestro caso es A Coruña. Por tanto, elegir 2 puntos tan diferentes nos dará más riqueza en los resultados.

Para empezar vamos a construir los modelos más sencillos posibles de Support Vector Regression, es decir aquellos con núcleo lineal. Tras este paso vamos a utilizar el núcleo gaussiano con el fin de conseguir una mejor separación de los datos.

Por último haremos una ampliación del problema anterior, en la que introduciremos en los patrones datos ECMWF de toda la Península para ampliar la dimensionalidad, usando las librerías LIBSVM y LIBLINEAR. En ellos vamos a tener dos tipos de problemas según la salida de los modelos, aquellos en los que tratamos de hacer predicciones por estaciones con datos de toda la península y los que trataremos de predecir la radiación media peninsular. Al tratarse de un problema diferente lo expondremos en una sección aparte.

3.2.1. Descripción de los datos

Para el estudio los datos de predicción son proporcionados por European Centre for Medium-Range Weather Forecasts (ECMWF), mientras que los datos de radiación real han sido cedidos por Red Eléctrica Española (REE).

De ECMWF tenemos radiación acumulada y nubosidad media en intervalos de 3 horas, para horizontes entre 0 y 72 horas (3 días), dados en una rejilla de 0.25 para toda la Península. Por tanto, las predicciones dadas por ECMWF son radiaciones trihorarias acumuladas para una latitud y longitud determinadas en una rejilla.

Por otro lado los datos reales vienen dados por estaciones solares, al menos una por provincia, en intervalos de una hora de 5 a 21 horas solares. Por falta o incompletitud de los datos nos quedaremos con 31 provincias y una estación por cada una, quedando el mapa de puntos en los que tenemos medidas de radiación como se muestra en la figura 3.2.1. Aunque para estos primeros experimentos sólo necesitaremos datos de las estaciones de Alicante y A Coruña, este mapa nos será útil para posteriores experimentos.

Antes de realizar el estudio lo primero que debemos hacer es unificar los datos. Trataremos con predicciones trihorarias en las horas en las que hay radiación, es decir, de 6, 9, 12, 15, 18, 21 horas solares, en las dos estaciones que hemos elegido. Como hemos dicho antes, nuestro objetivo es predecir para dos provincias, Alicante y A Coruña, en ECMWF tenemos puntos en una rejilla, por lo que usaremos como predicción ECMWF el punto más cercano de la rejilla de ECMWF para Alicante y A Coruña. En el caso de la radiación real, REE proporciona datos directamente en las estaciones solares.

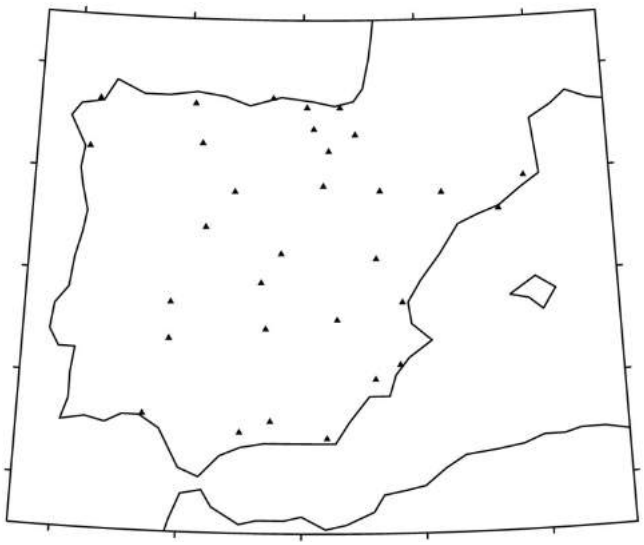

Figura 3.2.1: Mapa de las estaciones solares de las que tenemos datos de radiación real

Conj. test	**Conj. Validación**	**Conj. train**
m	$m-1$	Desde $m-13$ a $m-2$
Enero 2011	Diciembre 2010	Diciembre 2009-Noviembre 2010

Cuadro 3.2.1: Ejemplo del conjunto de entrenamiento, validación y test

El intervalo de fechas que hemos utilizado para construir los modelos comprende desde diciembre de 2009 a junio de 2011, construyendo modelos para cada uno de los 6 meses del año 2011. Para dicha construcción usaremos 3 conjuntos, entrenamiento, prueba y validación; éste último lo usaremos para la metamodelización, es decir, la búsqueda de los parámetros óptimos del modelo. Dado que en radiación es importante la serie temporal de los datos, pues no es lo mismo la radiación en enero que en junio, la obtención de los conjuntos de entrenamiento y validación no será hecha de manera aleatoria.

En concreto, el conjunto de validación será justo el anterior al mes al que queremos construir el modelo, ya que consideramos que ofrecerá una buena representación del mes a modelar. El conjunto de entrenamiento será de un año, 12 meses, desde el mes anterior al de validación hasta un año anterior. Por tanto, para construir el modelo del mes m usaremos como conjunto de validación el mes $m-1$ y desde $m-13$ hasta $m-2$ para el conjunto de entrenamiento. Por ejemplo, en enero de 2011, el conjunto de entrenamiento será desde diciembre de 2009 a noviembre de 2010, el conjunto de validación diciembre de 2010 y así sucesivamente para los otros 5 meses a modelar. La tabla 3.2.1 muestra un resumen de los conjuntos utilizados para construir el modelo.

Habría otras opciones para construir los conjuntos de entrenamiento, validación y prueba. La más interesante podría ser usar el mes m de 2011 para predecir y el mismo mes m de 2010 para validar, dejando el resto para el conjunto de entrenamiento. Por ejemplo, construir modelos para enero de 2011 y validar con enero de 2010.

En la misma línea, hemos decidido no usar validación cruzada, ya que podríamos

	Horizonte 1					...	Horizonte 6				
	P1		P2	P3	P4	...	P1	P2	P3	P4	
Rad. Real	DSSR	TCC	...							DSSR	TCC

Cuadro 3.2.2: Esquema de los patrones para modelos diarios

	Horizonte							
	Punto 1		Punto 2		Punto 3		Punto 4	
Rad. Real H1	DSSR	TCC	DSSR	TCC	DSSR	TCC	DSSR	TCC
Rad. Real H2	DSSR	TCC	DSSR	TCC	DSSR	TCC	DSSR	TCC
...								
Rad. Real H6	DSSR	TCC	DSSR	TCC	DSSR	TCC	DSSR	TCC

Cuadro 3.2.3: Esquema de los patrones para modelos trihorarios

usar días del año que no sean interesantes para validar el mes del que estamos construyendo el modelo.

3.2.2. Descripción de los patrones

Construimos patrones con los 4 puntos de la rejilla de ECMWF más cercanos a la latitud y longitud que hemos determinado para las estaciones de Alicante y A Coruña, siendo $(-8{,}42, 43{,}37)$ en A Coruña y $(-0{,}55, 38{,}28)$ en Alicante. Además escogemos 2 variables, Downward Surface Solar Radiation (DSSR) y Average Total Cloud Cover (TCC), predichas por el ECMWF.

Primero realizaremos predicciones diarias, para lo que construiremos los patrones con las 2 características recién citadas y los 4 puntos de la rejilla de ECMWF más cercanos que rodean a la longitud y latitud de la estación correspondiente a esa provincia. Usaremos las horas con radiación solar en intervalos de 3 horas, es decir, las predicciones para las 6, 9, 12, 15, 18 y 21 horas solares, obteniéndose finalmente $2 \times 4 \times 6 = 48$ atributos y 365 patrones en el conjunto de entrenamiento. Para el objetivo, o "target", escogemos la radiación real diaria, para lo que sumamos la radiación horaria proporcionada por REE entre las 5 y las 21 horas. La tabla 3.2.2 muestra un esquema de cómo se distribuyen los patrones para las predicciones diarias.

Nuestro segundo planteamiento será el de realizar predicciones trihorarias, construyendo los patrones como hemos descrito anteriormente y escogiendo como "target"la radiación real acumulada dada por REE para ese intervalo trihorario, ese día y esa longitud y latitud. Por tanto la dimensión de los patrones será de 4 puntos $\times$ 2 características $= 8$ atributos y tendremos $365 \times 6 = 2190$ patrones en el conjunto de entrenamiento. La tabla 3.2.3 muestra un esquema de como se distribuyen los patrones para las predicciones trihorarias.

3.2.3. Transformaciones

Para poder comparar los modelos necesitamos hacer una serie de transformaciones para homogeneizar los datos, ya que algunos son diarios y otros trihorarios. En el caso de querer comparar los modelos, SVR-3H y SVR-D y las predicciones de ECMWF de manera diaria primero debemos agregar las predicciones trihorarias dadas por los

modelos de SVR-3H a diarias, así como las predicciones de ECMWF, que vienen dadas de manera trihoraria. En cambio necesitaremos desagregar las predicciones diarias dadas por los modelos SVR diarios a predicciones trihorarias para poder comparar los resultados SVR-D, SVR-3H y ECMWF de manera trihoraria. Ambos procedimientos los explicamos a continuación.

2. Agregar predicciones trihorarias para conseguir predicciones diarias

En los primeros resultados compararemos radiación diaria. Como tendremos predicciones trihorarias del ECMWF y de nuestro modelo SVR-3H, necesitamos obtener la radiación para cada día a partir de las predicciones trihorarias de los modelos de ECMWF y SVR-3H.

Para un modelo general M, sus predicciones trihorarias $\widehat{I}_{d,3k}^{3;M}$ se convierten en predicciones diarias sumándolas. Esto es,

$$\widehat{I}_d^M = \sum_2^7 \widehat{I}_{d,3k}^{3;M} \tag{3.2.1}$$

En nuestro caso tendremos $M = E$ para las predicciones del ECMWF, $M = 3SL$ para las las predicciones de una SVR lineal trihoraria, $M = DSL$ para las predicciones de los modelos de la SVR lineal diaria y así sucesivamente.

3. Desagregar a predicciones diarias a predicciones trihorarias

Para desagregar las predicciones diarias de un modelo general M en valores trihorarios, usaremos el porcentaje que asigna ECMWF a cada intervalo trihorario del día en cuestión. Esto es, la predicción diaria $\widehat{I}_d^{D,M}$ de un modelo M da lugar a las predicciones trihorarias

$$\widehat{I}_{d,3k}^{3;M} = \frac{\widehat{I}_{d,3k}^{3;E}}{\widehat{I}_d^E}\widehat{I}_d^{D,M}, \tag{3.2.2}$$

donde ahora $M = DSL$ para las predicciones de una SVR diaria, y $\widehat{I}_d^{D,M}$ es la predicción diaria del modelo M para el día d, $\widehat{I}_d^E$ es la diaria de ECMWF para el día d y $\widehat{I}_{d,3k}^{3;E}$ es la predicción trihoraria acumulada de ECMWF para el día d a en el intervalo $[3h : 3h + 3]$.

3.2.4. Modelos lineales y gaussianos

Como hemos explicado en el capítulo 1, en la vida real podemos tener problemas con datos linealmente separables o no. Las SVM tienen una generalización para casos en los que los datos no sean linealmente separables. Esta generalización consiste en transformar los datos a un espacio de dimensión superior en el cual éstos sean linealmente separables mediante una función adecuada, esto es, usando una función de núcleo.

	A Coruña				Alicante			
	C_D	C_{3H}	ϵ_D	ϵ_{3H}	C_D	C_{3H}	ϵ_D	ϵ_{3H}
En	2^6	2^{14}	2^9	2^{-1}	2^5	2^5	2^{-1}	2^3
Feb	2^9	2^5	2^3	2^3	2^5	2^5	2^8	2^3
Mar	2^5	2^5	2^{-1}	2^5	2^5	2^5	2^3	2^5
Abr	2^5	2^{10}	2^{-1}	2^7	2^5	2^{12}	2^7	2^7
May	2^{14}	2^5	2^7	2^7	2^{15}	2^6	2^5	2^5
Jun	2^8	2^5	2^7	2^5	2^{14}	2^{12}	2^1	2

Cuadro 3.2.4: Parámetros óptimos para Alicante y A Coruña en el modelo lineal

Los modelos más sencillos que podemos construir son aquellos en los usamos el núcleo lineal. En ellos la función utilizada en el núcleo es el producto escalar, obteniéndose como función de proyección la función identidad, por tanto la más sencilla. La ventaja de estos modelos, además de la sencillez de la función de proyección, se encuentra en la metamodelización, donde no tenemos que optimizar el parámetro σ. En cambio, la desventaja será que al tratarse de problemas de dimensión baja, puede que los modelos no sean eficaces.

Por otro lado, si queremos usar modelos no lineales, el núcleo más comúnmente utilizado es el gaussiano. Ambas son opciones que podemos modificar en la librería LIBSVM y por lo tanto para nuestros experimentos vamos a usar modelos lineales y gaussianos.

A continuación explicamos el procedimiento utilizado en cada uno de los dos casos para metamodelizar, dando los resultados de los parámetros óptimos, y para obtener los modelos.

Predicción lineal

Tras aplicar las técnicas de Support Vector Regression (SVR) con núcleo lineal obtenemos en cada caso un modelo que nos dará la predicción trihoraria y diaria, a los que denominamos con la nomenclatura lSVR-3H y lSVR-D respectivamente.

Para intentar obtener modelos óptimos vamos a proceder a realizar una metamodelización de los parámetros de las SVR lineales, es decir, C, parámetro de regularización y ϵ, factor de tolerancia. Para ello, usamos la técnica de búsqueda exhaustiva en una rejilla de 2 dimensiones, donde consideramos valores entre $C \in [2^5 : 2^{15}]$ con paso 2^1 y $\epsilon \in [2^{-1} : 2^{15}]$ con paso 2^2.

La tabla 3.2.4 muestra los parámetros óptimos para las dos provincias, Alicante y A Coruña, para cada uno de los modelos con núcleo lineal, tanto para la predicción diaria, lSVR-D, como la trihoraria, lSVR-3H. En este caso el subíndice D se refiere a los parámetros de los modelos diarios, SVR-D, mientras que el subíndice $3H$ se refiere a los modelos trihorarios, SVR-3H.

Como podemos ver en la tabla algunos de los valores óptimos que obtenemos se encuentran en el borde de la malla que tenemos. Esto significa que si ampliamos la malla quizás halla algún otro valor óptimo. En este caso hemos trabajado con esta malla, ya que en radiación los modelos cambian mucho de un mes a otro y es por esto que algunos meses tienen sus parámetros en el borde. Por tanto, encontrar

	A Coruña						Alicante					
	C_D	C_{3H}	σ_D	σ_{3H}	ϵ_D	ϵ_{3H}	C_D	C_{3H}	σ_D	σ_{3H}	ϵ_D	ϵ_{3H}
En	2^{15}	2^5	2^{-14}	2^{-7}	2^1	2^3	2^{10}	2^5	2^{-7}	2^{-7}	2^7	2^3
Feb	2^7	2^5	2^{-6}	2^{-7}	2^3	2^1	2^{11}	2^5	2^{-8}	2^{-7}	2^7	2^1
Mar	2^{15}	2^6	2^{-14}	2^{-9}	2^7	2^3	2^8	2^6	2^{-7}	2^{-9}	2^5	2^3
Abr	2^{14}	2^{15}	2^{-11}	2^{-7}	2^7	2^{11}	2^6	2^{15}	2^{-5}	2^{-7}	2^5	2^5
May	2^{11}	2^{14}	2^{-5}	2^{-3}	2^{-1}	2^5	2^9	2^{14}	2^{-6}	2^{-3}	2^7	2^5
Jun	2^{15}	2^{15}	2^{-14}	2^{-7}	2^7	2^{-1}	2^{15}	2^{15}	2^{-15}	2^{-7}	2^7	2^{-1}

Cuadro 3.2.5: Parámetros óptimos para Alicante y A Coruña en el modelo gaussiano

una malla centrada para cada parámetro y cada modelo sería mucho más costoso computacionalmente.

Predicción gaussiana

Como en el caso anterior en el que usábamos las técnicas de Support Vector Regression (SVR) con núcleo lineal, en este nuevo experimento usaremos el núcleo gaussiano para las SVR. Realizaremos nuevamente modelos que nos darán las predicciones trihorarias y diarias, a los que denominamos con la nomenclatura gSVR-3H y gSVR-D respectivamente.

También usaremos la librería LIBSVM usando el núcleo gaussiano y realizaremos una metamodelización para mejorar nuestra predicciones, aunque en este caso tendremos una malla de tridimensional para buscar C, σ y ϵ. Donde consideramos valores entre $C \in [2^5 : 2^{15}]$ con paso 2^1, $\sigma \in [2^{-18} : 2^3]$ con paso 2^2, siendo σ la anchura del núcleo gaussiano, y $\epsilon \in [2^{-1} : 2^{15}]$ con paso 2^2.

La tabla 3.2.5 muestra los parámetros óptimos para las dos estaciones, Alicante y A Coruña, para cada uno de los 6 modelos con núcleo gaussiano para predicciones diarias y trihoraias, cuya notación será gSVR-D y gSVR-3H respectivamente. Nuevamente, el subíndice D se refiere a los parámetros de los modelos diarios, gSVR-D, mientras que el subíndice $3H$ se refiere a los modelos trihorarios, gSVR-3H.

3.2.5. Resultados

En este apartado vamos a evaluar los modelos que hemos explicado en los apartados anteriores. Recordamos que los experimentos que hemos realizado en radiación son los siguientes:

- Predecir radiación para Alicante y A Coruña con núcleo lineal.
  - Modelos radiación diaria.
  - Modelos radiación trihoraria.
- Predecir radiación para Alicante y A Coruña con núcleo gaussiano.

- Modelos radiación diaria.
- Modelos radiación trihoraria.

Compararemos el error tanto del ECMWF como el de nuestra predicción diaria dada por los modelos lSVR-D y gSVR-D y trihoraria de los modelos lSVR-3H y gSVR-3H. En el siguiente apartado nos referiremos a modelos que nos dan predicción diaria con SVR-D si nos referimos tanto a modelos lineales como gaussianos y SVR-3H en el caso de los modelos trihorarios.

Para construir los modelos de estos experimentos hemos usado la librería LIBSVM. En primer lugar hemos trabajado con el núcleo lineal, el más sencillo de todos y luego hemos pasado al gaussiano para proyectar los datos a un espacio de dimensión infinita, donde nos aseguramos que son linealmente separables. Con estos dos experimentos esperamos que el núcleo gaussiano de mejores resultados aunque el coste computacional resulte más alto; de todas formas, al tratarse de problemas de baja dimensión, el coste computacional no será significativo.

El procedimiento utilizado para construir modelos es el mismo que en el caso lineal que en el gaussiano, usamos la misma librería, LIBSVM e igual construcción de patrones. Lo único que cambia es la metamodelización, donde en lugar de usar una malla bidimensional, tendremos una malla en tridimensional para agregar el parámetro sigma σ, que es la anchura de la función de núcleo gaussiana.

Por último evaluaremos los modelos mediante el Error Medio Absoluto, MAE, usando las medidas reales de radiación dadas por Red Eléctrica Española (REE). Mostraremos los resultados obtenidos para todos los modelos, indicando primero cómo se van a estimar los distintos MAEs y dando luego los correspondientes resultados.

A continuación vamos a comparar los modelos SVR-D, SVR-3H y ECMWF desde dos perspectivas diferentes. La primera será una comparación diaria y la segunda trihoraria.

Radiación diaria

En esta primera perspectiva vamos a comparar las salidas diarias de los modelos.

Si definimos ND_m como el número de días que componen el mes m, la fórmula 3.2.3 nos da el MAE de la predicción diaria de todos los modelos:

$$e_D^E(m) = \frac{1}{ND_m} \sum_{d=1}^{ND_m} |I_d - \widehat{I}_d^M|. \tag{3.2.3}$$

donde $\widehat{I}_d^M$ representan las predicciones diarias para el día d dadas por los modelos y I_d es la radiación real diaria para el día d. Como se aprecia, el MAE se calcula como el promedio mensual de los MAEs de cada día del mes.

Aplicando la fórmula 3.2.1 para agregar las predicciones trihorarias de SVR-3H y las dadas por ECMWF a predicciones diarias y 3.2.3 a la salida de nuestros modelos y las predicciones dadas por ECMWF para calcular el MAE obtenemos la tabla de errores 3.2.6. En esta tabla comparamos nuestra predicción diaria dada por el modelo

Cuadro 3.2.6: MAE de las predicciones diarias de ECMWF agregado y modelos SVR-D y SVR-3H agregado con núcleo lineal y gaussiano.

	A Coruña						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	151.90	198.99	250.70	227.03	272.79	357.13	243.09
lSVR-D	150.10	132.11	185.49	231.62	257.32	363.24	219.98
gSVR-D	**117.51**	**124.96**	**146.78**	250.93	261.64	**337.32**	**206.53**
lSVR-3H	136.60	142.45	179.90	**215.83**	**224.06**	389.97	214.80
gSVR-3H	126.60	133.50	168.20	246.55	262.27	370.27	217.90
	Alicante						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	127.96	103.88	284.56	233.82	246.72	233.26	205.03
lSVR-D	132.80	99.29	**220.78**	184.34	199.53	261.34	183.01
gSVR-D	124.26	111.38	238.92	182.10	206.18	225.48	181.39
lSVR-3H	114.17	95.54	234.29	203.12	211.32	238.12	182.76
gSVR-3H	**107.58**	**93.99**	256.94	**181.02**	**195.41**	**221.17**	**176.02**

SVR-D y las predicciones trihorarias desagregada a trihoraria, SVR-3H y ECMWF, tanto en el caso lineal como gaussiano.

Como puede ser observado siempre uno de nuestros modelos, lSVR-D, lSVR-3H, gSVR-D o gSVR-3H mejora la predicción dada por ECMWF para todos los meses en las dos provincias. También podemos observar que en general funcionan mejor para Alicante los modelos trihorarios, mientras que en A Coruña los diarios, además los modelos con núcleo gaussiano suelen dar mejores resultados como aquellos modelos con núcleo lineal, como cabía esperar.

De todos modos no hay un comportamiento claro en ninguno de los casos, ya que en predicción de energía solar hay mucha variabilidad de las condiciones atmosféricas, que influyen en gran medida en el modelo y por tanto en la predicción y error final, tanto geográfica como temporal, siendo difícil sacar una misma conclusión para todos los modelos de cada mes y cada provincia.

Radiación trihoraria

En la segunda parte de este experimento vamos a comparar la salida trihoraria de los modelos. La fórmula 3.2.4 nos dará el MAE para la predicción del ECMWF y los modelos lSVR-3H y gSVR-3H;

$$e^E_{3H}(m) = \frac{1}{ND_m} \sum_{d=1}^{ND_m} \frac{1}{6} \sum_{k=2}^{7} |I^3_{d,3k} - \widehat{I}^{3,M}_{d,3k}| \tag{3.2.4}$$

donde $\widehat{I}^{3,M}_{d,3k}$ representa las predicciones trihorarias para el día d a la hora $3k$ con $k = 2, ..., 7$ dadas del modelo M, y $I^3_{d,3k}$ es la radiación real trihoraria para el día d a la hora $3k$ con $k = 2, ..., 7$.

Aplicando la fórmula 3.2.2 para desagregar las predicciones diarias de SVR-D y s y 3.2.4 a la salida de nuestros modelos y las predicciones dadas por ECMWF para

Cuadro 3.2.7: MAE de las predicciones trihorarias de ECMWF y modelos SVR-D desagregado y SVR-3H con núcleo lineal y gaussiano.

	A Coruña						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	39.06	50.57	66.58	71.19	79.47	88.64	65.92
lSVR-D	37.97	40.78	54.55	68.14	80.75	87.04	61.54
gSVR-D	**34.01**	**39.64**	**51.45**	67.76	80.07	**84.39**	59.55
lSVR-3H	35.89	44.32	56.96	72.65	84.95	89.19	63.99
gSVR-3H	34.22	41.85	52.17	**64.03**	**72.92**	87.00	**58.70**
	Alicante						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	29.21	33.81	57.73	47.71	53.09	50.28	45.30
lSVR-D	29.81	32.44	52.38	44.22	**50.26**	54.49	**43.93**
gSVR-D	28.68	32.82	54.60	**44.04**	53.35	51.76	44.21
lSVR-3H	**27.75**	31.87	**52.06**	45.77	53.08	52.68	**43.87**
gSVR-3H	28.38	**30.49**	57.78	45.09	52.03	**50.16**	**43.99**

calcular el MAE obtenemos la tabla de errores 3.2.7.

En esta tabla comparamos tanto nuestra predicción trihoraria dada por el modelo SVR-3H como la predicción diaria desagregada a trihoraria con el ECMWF, siendo mejor en media la predicción trihoraria para valores trihorarios como cabía esperar, aunque la mejora no llegue a ser significativa, pero siempre mejor que las predicciones de ECMWF. Nuevamente los modelos gaussianos dan mejores resultados, sobre todo en A Coruña, siendo menos concluyente en Alicante donde todos los modelos cometen un error más parecido, probablemente por tratarse de un punto geográfico con mejores condiciones atmosféricas.

3.3. Predicción con patrones peninsulares

3.3.1. Predicción por estación

Para profundizar un poco más en los experimentos relacionados con energía solar queremos aumentar la dimensión de los patrones que utilizamos y realizar pruebas con LIBLINEAR. Para ello extendemos la información de DSSR y TCC a todos los puntos de la Península. Por tanto, nuestro nuevo problema consiste en dar predicciones para Alicante y A Coruña, pero teniendo información de toda la Península en los atributos.

En este caso vamos a utilizar sólo modelos lineales, ya que como hemos dicho previamente LIBLINEAR se basa en la observación de que para patrones de altas dimensiones los datos estarán a efectos prácticos separados linealmente y por tanto no necesitemos hacer una proyección de los datos a un espacio de dimensión superior, siendo la función del núcleo la lineal.

Siguiendo el esquema anterior vamos a realizar predicciones diarias y trihorarias. En el caso de predicciones diarias, la dimensión está determinada por 6 horas del día, 2 características y 1995 puntos en la rejilla dada por ECMWF, obteniéndose en

Cuadro 3.3.1: Parámetros óptimos para Alicante y A Coruña para LIBSVM y LIBLINEAR en modelos diarios

	A Coruña		Alicante	
	C	ϵ	C	ϵ
En	2^{-5}	2^{5}	2^{-3}	2^{5}
Feb	2^{-1}	2^{9}	2^{-4}	2^{5}
Mar	2^{-3}	2^{7}	2^{-3}	2^{1}
Abr	2^{-3}	2^{-1}	2^{1}	2^{3}
May	2^{-2}	2^{3}	2^{1}	2^{-1}
Jun	2^{0}	2^{-1}	2^{-2}	2^{7}

total $6 \times 2 \times 1995 = 23940$ atributos y 365 patrones en el conjunto entrenamiento. En cambio, en el caso de predicciones trihorarias tendremos $2 \times 1995 = 3990$ atributos y $365 \times 6 = 2190$ patrones de entrenamiento.

El tratamiento y rango de los datos es el mismo que en el caso anterior. El algoritmo utilizado para predecir vuelve a ser Máquinas de Vectores Soporte para regresión, pero en este caso vamos a utilizar dos librerías diferentes para compararlas; estas son LIBSVM y LIBLINEAR, aplicando también técnicas de metamodelización para mejorar los modelos.

Como podemos observar, la diferencia en este experimento radica en la dimension de los patrones y en las librerias usadas. LIBSVM, usa SMO, mientras que LIBLINEAR utiliza DCD, explicados en los capítulos 1 y 2 respectivamente. Además LIBLINEAR se basa en la observación de que para alta dimensionalidad es probable que los datos estén linealmente separados y por tanto no es necesario el uso de un núcleo para pasar los datos iniciales a un espacio de dimensionalidad más alta, lo que reduce el tiempo de entrenamiento. Por tanto, para que los resultados de las dos librerías sean comparables, en LIBSVM usaremos el núcleo lineal ya que LIBLINEAR usa modelos lineales, siendo bidimensional la rejilla utilizada para metamodelizar estos experimentos. En consecuencia buscaremos los parámetros C y ϵ, donde $C \in [2^{-5} : 2^{5}]$ con paso 2^2 y $\epsilon \in [2^{-1} : 2^{15}]$ con paso 2^2. En este caso hemos cambiado la rejilla ya que al tener patrones de toda la península el modelo no debería penalizar tanto los errores, y los parámetros óptimos los hemos encontrado en otro rango.

Las tablas 3.3.1 y 3.3.2 muestran los parámetros óptimos de Coruña y Alicante obtenidos LIBLINEAR para modelos diarios y trihorarios respectivamente. Hemos decidido usar la libreria LIBLINEAR para metamodelizar ya que un estudio previo de los tiempos muestra que LIBLINEAR es más rápido que LIBSVM y como se ha desarrollado en los capítulos 1 y 2 ambos resuelven el mismo problema.

El tiempo empleado para entrenar LIBSVM es de 50 segundos en modelos trihorarios y 15 segundos en modelos diarios. En cambio LIBLINEAR necesitará 15 segundos en modelos trihorarios y 2 segundos en modelos diarios. Por tanto al hacer la búsqueda exhaustiva en rejilla, teniendo 9 puntos para C y 11 para ϵ nos darán un total de 99 puntos y por tanto la diferencia de tiempo necesario para entrenar y predecir haciendo una búsqueda exhaustiva en rejilla empieza a ser mayor.

Por último los resultados también serán medidos mediante el MAE y comparados

Cuadro 3.3.2: Parámetros óptimos para Alicante y A Coruña para LIBSVM y LIBLINEAR en modelos trihorarios

	A Coruña		Alicante	
	C	ϵ	C	ϵ
En	2^{-2}	2^{-1}	2^{-2}	2^{3}
Feb	2^{-2}	2^{3}	2^{0}	2^{1}
Mar	2^{-1}	2^{3}	2^{-2}	2^{1}
Abr	2^{-1}	2^{3}	2^{0}	2^{-1}
May	2^{0}	2^{-1}	2^{1}	2^{1}
Jun	2^{1}	2^{5}	2^{-1}	2^{1}

Cuadro 3.3.3: MAE de las predicciones diarias de ECMWF, SVR-D y SVR-3H en Alicante y A Coruña con información de toda la Península con LIBLINEAR y LIBSVM.

	A Coruña						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	151.90	198.99	250.70	**227.03**	272.79	357.13	243.09
D-SVR-liblinear	**126.74**	173.99	238.68	309.49	298.36	355.70	250.49
D-SVR-libsvm	132.09	158.58	**220.37**	311.09	284.68	**335.07**	240.31
3H-SVR-liblinear	159.79	**148.74**	235.42	268.43	257.16	373.76	240.55
3H-SVR-libsvm	153.60	162.35	234.10	235.08	**254.76**	366.44	**234.39**
	Alicante						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	127.96	**103.88**	284.56	233.82	**246.72**	233.26	205.03
D-SVR-liblinear	**127.79**	152.64	259.37	192.20	296.29	231.92	210.03
D-SVR-libsvm	133.33	153.85	**256.19**	195.16	264.23	**222.92**	204.28
3H-SVR-liblinear	132.98	109.17	269.15	**188.89**	291.69	223.55	202.57
3H-SVR-libsvm	124.42	106.00	265.32	198.47	263.09	223.60	**196.82**

con el ECMWF. Los resultados para Alicante y A Coruña se muestran en la tabla 3.3.3, donde D-SVR-libsvm y D-SVR-liblinear se refiere a los modelos diarios dados por la librería LIBSVM y LIBLINEAR respectivamente y 3H-SVR-libsvm y 3H-SVR-liblinear se refiere a los modelos trihorarios dados por la librería LIBSVM y LIBLINEAR respectivamente.

Como podemos observar, en este caso nuestros modelos no son mucho mejores que los de ECMWF, llegando incluso a ser peores que los de ECMWF para algunos meses. Esto puede ser debido a que mucha de la información que estamos metiendo en los atributos es poco relevante. Por ejemplo, la radiación y nubosidad de Almería no es relevante para predecir la de A Coruña, llegando a ser perjudicial. Los resultados obtenidos no muestran un claro ganador pero si podemos ver que en MAE medio los mejores modelos son los trihorarios.

3.3.2. Modelos de radiación media peninsular

Por último, y para terminar con los experimentos relacionados con energía solar, igual que hemos hecho en el apartado 3.3, queremos aumentar la dimensión pero

Cuadro 3.3.4: MAE de las predicciones trihorarias de ECMWF, SVR-D y SVR-3H en Alicante y A Coruña con información de toda la Península con LIBLINEAR y LIBSVM.

	A Coruña						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	39.06	50.57	66.58	71.19	79.47	88.64	65.92
D-SVR-liblinear	34.70	44.72	59.67	78.38	88.74	87.81	65.67
D-SVR-libsvm	33.92	43.33	58.48	78.40	87.55	86.09	64.63
3H-SVR-liblinear	35.86	**40.95**	**55.20**	58.14	79.66	84.94	59.13
3H-SVR-libsvm	**32.82**	42.03	55.47	**56.04**	**77.76**	**84.48**	**58.10**
	Alicante						
	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	**29.21**	**33.81**	57.73	47.71	**53.09**	**50.28**	**45.30**
D-SVR-liblinear	30.51	35.99	57.02	47.56	64.54	57.29	48.82
D-SVR-libsvm	31.05	36.60	**56.01**	47.70	58.64	55.81	47.63
3H-SVR-liblinear	30.34	36.41	56.41	**47.09**	65.58	53.84	48.28
3H-SVR-libsvm	30.01	35.69	56.54	47.71	61.67	54.10	47.62

Cuadro 3.3.5: Parámetros óptimos de los modelos de predicción media total peninsular con LIBSVM y LIBLINEAR en modelos diarios

	C	ϵ
En	2^{13}	2^{11}
Feb	2^{9}	2^{11}
Mar	2^{5}	2^{13}
Abr	2^{5}	2^{3}
May	2^{9}	2^{11}
Jun	2^{9}	2^{11}

planteando otro problema. Para ello usaremos la información de DSSR y TCC en todos los puntos de la Península, con la diferencia de que en este caso vamos a predecir la radiación media de la Península diaria, donde entendemos por la radiación media diaria vendrá dada por la suma de la radiación en cada una de las estaciones que tenemos en el mapa 3.2.1 a lo largo de las horas con radiación, esto desde 6 a 21 horas, y dividido entre el número de estaciones que tenemos.

La dimensión de los atributos y el número de patrones es el mismo que en el problema del apartado anterior. Las características de los patrones son las mismas, DSSR y TCC para cada uno de los puntos de la rejilla de península, cambiando solamente el "target": en lugar de predecir la radiación para una estación determinada trataremos de dar la radiación media peninsular.

El rango de datos también se mantiene, así como la técnica aplicada, SVR con LIBSVM y LIBLINEAR, mejorando los modelos con metamodelización, donde consideramos valores entre $C \in [2^5 : 2^{15}]$ con paso 2^1 y $\epsilon \in [2^{-1} : 2^{15}]$ con paso 2^2 para modelos diarios y $C \in [2^{-5} : 2^5]$ con paso 2^1 y $\epsilon \in [2^{-1} : 2^{15}]$ en modelos trihorarios. Las tablas 3.3.5 y 3.3.6 dan los parámetros óptimos para predicción medias totales con LIBSVM y LIBLINEAR en modelos diarios y modelos trihorarios respectivamente.

Cuadro 3.3.6: Parámetros óptimos de los modelos de predicción media total peninsular con LIBSVM y LIBLINEAR en modelos trihorarios

	C	ϵ
En	2^1	2^{-1}
Feb	2^1	2^{-1}
Mar	2^2	2^3
Abr	2^4	2^2
May	2^4	2^9
Jun	2^5	2^1

Cuadro 3.3.7: MAE para la predicción media total diaria peninsular dadas por los modelos de ECMWF y SVR con LIBSVM y LIBLINEAR

	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	97.34	67.15	160.68	176.68	133.48	**98.18**	122.25
SVR-Liblinear	**76.72**	58.81	135.47	124.52	88.18	119.80	100.58
SVR-Libsvm	83.16	**55.47**	**114.08**	**117.69**	**85.98**	104.01	**93.40**

Para el caso de la predicción de la radiación media de la península obtenemos las tablas 3.3.7 y 3.3.8 que muestran el MAE de los modelos peninsulares medios diarios y trihorarios con LIBLINEAR y LIBSVM.

Podemos observar como la predicción mejora bastante, lo que se puede deber a que al predecir la radiación total de toda la península los errores se compensan más comparando con las predicciones dadas en puntos determinados.

3.4. Desagregación horaria

De cara a un uso práctico es claro que la predicción de interés es la de la radiación horaria, que vamos a obtener en una primera aproximación desagregando las predicciones trihorarias mediante el modelo Clear Sky de Bird [14], que nos dará la radiación directa horizontal (DHR) para una hora t en el día d para una determinada longitud y latitud como función de $c(t;d)$. Por tanto, la radiación acumulada entre la hora $h-1$ y la h será:

$$a(h;d) = \int_{h-1}^{h} c(t;d)dt \simeq \frac{1}{M}\sum_{i=1}^{M} c\left(h-1+\frac{i}{M};d\right),$$

donde M determina el paso que vamos a dar para la integración, en este experimento en concreto será 10.

Para desagregar la radiación acumulada trihoraria $\widehat{I}^3_{d,h}$ como la suma $\widehat{I}^3_{d,h} = \widehat{I}_{d,h} + \widehat{I}_{d,h-1} + \widehat{I}_{d,h-2}$ de los valores de radiación horaria, derivamos la predicción horaria $\widehat{I}_{d,3k-2}, \widehat{I}_{d,3k-1}, \widehat{I}_{d,h}$ de la trihoraria $\widehat{I}^3_{d,3k}$ con la siguiente fórmula:

$$\widehat{I}_{d,3k-j} = \frac{a(3k-j;d)}{A(3k;d)} I^3_{d,3k},$$

Cuadro 3.3.8: MAE para la predicción media total trihoraria peninsular dadas por los modelos de ECMWF y SVR con LIBSVM y LIBLINEAR

	En	Feb	Mar	Abr	May	Jun	Media
ECMWF	20.98	27.74	34.29	33.31	28.54	25.68	28.42
SVR-Liblinear	**14.47**	**21.80**	**26.82**	**23.48**	**26.90**	**23.13**	**22.77**
SVR-Libsvm	14.82	21.77	26.84	26.23	29.50	24.99	23.53

donde $k = 2, \ldots, 7$ y $A(h;d) = a(h-2;d) + a(h-1;d) + a(h;d) = \int_{h-3}^{h} c(t;d)dt$. Asegurándose que: $\widehat{I}^3_{d,3k} = \widehat{I}_{d,3k} + \widehat{I}_{d,3k-1} + \widehat{I}_{d,3k-2}$.

Este modelo lo aplicaremos intentando dar una predicción más fina de nuestros resultados, es decir, las predicciones horarias de SVR-3H, SVR-D y ECMWF. Para hacer la desagregación horaria serán necesario dos pasos; obtener la predicción trihoraria a partir de la predicción diaria, como hemos descrito anteriormente y desagregar a predicciones horarias mediante la curva Clear Sky tal y como hemos explicado previamente.

A continuación vamos a mostrar una serie de tablas donde desagregamos mediante el modelo Clear Sky cada uno de los modelos anteriores y obtenemos el MAE. La relación de tablas es la siguiente:

- La tabla 3.4.1 muestra los resultados del MAE para predicciones horarias de los modelos con núcleo lineal con patrones de los 4 puntos más cercanos lSVR-3H, lSVR-D, gSVR-3H y gSVR-D para Alicante y A Coruña.

- La tabla 3.4.2 muestra los resultados del MAE para predicciones horarias de los modelos con núcleo lineal con patrones de los de toda la península SVR-3H y SVR-D para Alicante y A Coruña con LIBLINEAR y LIBSVM.

Cuadro 3.4.1: MAE para predicciones horarias ECMWF, lSVR-3H, lSVR-D, gSVR-3H y gSVR-D.

	A Coruña						
	Jan	Feb	Mar	Apr	May	Jun	Ave
ECMWF	19.55	22.92	29.28	32.64	37.74	39.17	30.22
lSVR-D	19.29	19.82	25.69	32.02	38.52	38.46	28.97
gSVR-D	**17.58**	**19.11**	**24.54**	31.39	37.78	**37.29**	27.95
lSVR-3H	18.38	20.94	26.50	33.17	39.79	39.57	29.72
gSVR-3H	17.71	19.85	24.56	**30.72**	**35.11**	38.31	**27.71**
	Alicante						
	Jan	Feb	Mar	Apr	May	Jun	Ave
ECMWF	17.80	17.66	27.44	24.36	27.33	**24.68**	23.21
lSVR-D	16.95	17.47	**25.44**	22.94	**26.15**	26.79	22.62
gSVR-D	16.84	17.53	26.03	**22.90**	27.46	25.87	22.77
lSVR-3H	17.04	17.20	26.60	25.97	27.64	27.64	25.70
gSVR-3H	**16.57**	**16.80**	27.43	23.14	26.66	24.85	**22.57**

Cuadro 3.4.2: MAE para predicciones horarias ECMWF, SVR-3H y SVR-D con LIBLINEAR y LIBSVM

	A Coruña						
	Jan	Feb	Mar	Apr	May	Jun	Ave
ECMWF	19.55	22.92	29.28	32.64	**37.74**	39.17	30.22
SVR-D-liblinear	18.18	21.56	27.61	35.91	40.85	38.46	30.43
SVR-D-libsvm	18.78	21.20	27.12	35.95	40.54	37.86	30.24
SVR-3H-liblinear	18.13	**20.82**	26.67	31.38	38.86	39.18	29.17
SVR-3H-libsvm	**17.69**	20.91	**26.56**	**30.27**	38.52	**39.10**	**28.84**
	Alicante						
	Jan	Feb	Mar	Apr	May	Jun	Ave
ECMWF	17.80	**17.66**	27.44	24.36	**27.33**	**24.68**	**23.21**
SVR-D-liblinear	**17.27**	18.78	27.26	**23.37**	30.67	28.39	24.29
SVR-D-libsvm	17.28	19.03	**27.13**	23.52	29.17	27.71	23.97
SVR-3H-liblinear	17.32	18.38	27.39	24.41	32.00	26.28	24.30
SVR-3H-libsvm	17.41	18.46	27.50	25.06	30.90	26.32	24.28

Como podemos observar, si aplicamos el modelo clear sky para desagregar los errores la tendencia de los MAE se suele mantener en ambas tablas. Es decir, en el caso de la comparación de errores de núcleo gaussiano y lineal, tabla 3.4.1, en la mayoría de los casos el núcleo gaussiano consigue mejores resultados incluso después de desagregar las predicciones. En la tabla 3.4.2 podemos ver que para A Coruña son mejores los modelos trihorarios y en Alicante dependerá del mes. En todo caso los modelos clear sky actuales no tienen en cuenta muchas variables meteorológicas en el modelo físico, por lo que hoy por hoy no dan una buena representación de días nublados, este puede ser un punto de partida para refinar y mejorar las predicciones.

3.5. Conclusiones sobre los experimentos de radiación

Tras el auge de las energías renovables cada vez es más necesaria la integración e investigación de buenas técnicas de predicción. En este sentido el aprendizaje automático juega un papel importante siendo las Máquinas de Vectores Soporte para regresión una de las técnicas utilizada para ello. Nuestros experimentos han mostrado muy buenos resultados, mejorando las predicciones dadas por ECMWF como pretendíamos. La metamodelización es clave para mejorar los modelos y disminuir los errores.

En los primeros experimentos con predicciones trihorarias y diarias no hay un claro vencedor, ya que depende del mes y la provincia. Sí que podemos observar que en los meses dónde es más difícil predecir por la nubosidad, como puede ser la primavera, en general funcionan mejor los modelos diarios SVR-D ya que los errores se pueden compensar a lo largo del día. Por último, queda claro por los errores, que las predicciones en Alicante son mejores que en A Coruña, donde a pesar de tener menos radiación los MAE son más altos.

Para los experimentos posteriores en los que incluimos información de toda la península para construir modelos para Alicante y A Coruña, hemos podido observar que introducir demasiada información en los patrones puede ser perjudicial, ya que introducen ruido en el modelo y nos dan peores resultados que los previos. Haciendo un buen ajuste de los hiperparámetros del modelo podemos conseguir buenos resultados. En este caso además hemos usado la librería LIBLINEAR para hacer la búsqueda de los hiperparámetros ya que es más rápida.

En cuanto a los modelos para la predicción de la radiación total de la Península, como hemos dicho arriba, los resultados mejoran bastante, debido a que lo errores se compensan y que tenemos más información en los patrones.

Como hemos dicho previamente, no tenemos resultados concluyentes en todos los casos ya que en predicción de energía solar hay mucha variabilidad de las condiciones atmosféricas, que influyen en gran medida en el modelo y por tanto en la predicción y error final, tanto geográfica como temporal, siendo difícil sacar una misma conclusión para todos los modelos de cada mes y cada provincia.

3.6. Clasificación de páginas web

El segundo tipo de problema que queremos resolver es el de clasificación de páginas web. Como hemos dicho anteriormente, hoy en día la cantidad de información a la que se puede acceder en la web está aumentando rápidamente debido al desarrollo de la World Wide Web (WWW). Ésta es la razón por la que la clasificación de páginas web tiene una importancia especial. El objetivo en este tipo de problemas es, dada una página web, asignarle una determinada categoría dependiendo del contenido de dicha página.

En este caso vamos a tratar de clasificar entre 2 de las 16 categorías dadas en ODP [1]. Open Directory Project (ODP), es un proyecto colaborativo, en el que editores voluntarios listan y categorizan enlaces a páginas web. Por tanto nuestro problema será dada una url tratar de determinar si se trata de una página relacionada con arte o con negocios, las dos categorías elegidas para el experimento.

La minería web trata de descubrir patrones en las páginas web que sean útil para la clasificación de las mismas. Es un área del conocimiento en el que confluyen varias ramas, estadísticas, aprendizaje automático, recuperación de información, etc. En este proceso tenemos una serie de pasos que son los siguientes:

- **Obtención de los datos**.
- **Preprocesamiento**: en el que ocurre la limpieza de datos y filtrados.
- **Minería de datos**: extracción de información y representación de los mismos para obtener los patrones.
- **Clasificación**: uso de técnicas del aprendizaje automático para la clasificación de páginas web.

[1] Open Directory Project, http://www.dmoz.org/

De los 4 pasos recién citados nosotros nos centramos en los dos últimos. El primero consiste en la extracción de los datos e información. En este caso se estudia cuál de la múltiple información proporcionada por una página web es relevante o no para nuestra clasificación, como tags, keywords, tittle....componiendo de esta manera los atributos de nuestros patrones.

El segundo pertenece al área de la clasificación, en la que se pueden utilizar muchas técnicas del Aprendizaje Automático, y en la que las Máquinas de Vectores Soporte han demostrado buenos resultados por ser un algoritmo robusto y que puede tratar con problemas de alta dimensionalidad, gracias a la amplia investigación que hay en este campo. Para nuestros experimentos, la parte en la que profundizaremos más será esta última y más concretamente en los algoritmos que mejor funcionan en problemas de alta dimensionalidad.

3.6.1. Extracción de datos

Existen dos técnicas que son utilizadas especialmente en el proceso de extracción de datos y composición de los atributos. Ambas tratan de buscar las palabras con su frecuencia para dar un peso mayor aquellas que tienen más frecuencia, ya que probablemente serán más relevantes en el documento o url. Por tanto, la extracción de datos y composición de atributos se valen de técnicas estadísticas para evaluar o medir la relevancia que puede tener una palabra en un texto o página web. Estas dos técnicas son Bag of Words o bolsa de palabras y Basket Market Analysis o análisis de la cesta de la compra.

Bag of Words

Dado un conjunto de documentos, se determina la frecuencia de aparición de cada palabra del conjunto en cada uno de los documentos, sin importar el orden en la que éstas aparecen. Este algoritmo hace 2 suposiciones:

- La probabilidad de una palabra es independiente del orden en que aparece, es decir, el orden de aparición no es importante.
- La longitud del documento es independiente de su clase.

Como primer paso en este proceso, suele tener lugar un preproceso en el que se eliminan ciertas palabras, raíces, etc. La información obtenida puede ser más o menos refinada en función de los filtros previos que se apliquen sobre los textos, éstos son;

- **Tokenización**: eliminación de signos de puntuación y espacio en blanco.
- **Eliminación de stop words**: generalmente determinantes, conjunciones, preposiciones, etc... palabras muy frecuentes de bajo valor semántico.
- **Stemming**: se trata de reducir de las palabras a su raíz. De esta manera permite contar como un mismo término independientemente de su número, género o de formas verbales distintas.

- **Estandarización de las palabras**: las palabras irregulares son estandarizadas a una única forma. Por ejemplo, palabras diferentes en inglés americano e inglés británico.

Haciendo las suposiciones anteriores y aplicando los filtros propuestos, tenemos un conjunto de documentos D, donde cada documento d_n, con $d_n \in D$ está representado como un conjunto de palabras t_i de nuestro vocabulario V, es decir $t_i \in V$.

Por tanto, si n es el número de documentos y m es el número de palabras de nuestro vocabulario, tenemos que:

$$D = \{d_1, ..., d_n\}$$
$$V = \{t_1, ...t_m\}$$

Tras tener representados los documentos o páginas web como un conjunto de palabras o "bag of words", aplicamos técnicas estadísticas de conteo para obtener el peso de una palabra en el documento o en el conjunto total de documentos, lo que nos permitirá obtener los atributos de nuestros patrones. Por tanto a cada término o palabra se le asigna un peso w_{ij} en el documento d_j.

$$d_j = \{w_{1n}, w_{2n}, ..., w_{mn}\}$$

donde

$$w_{ij} = \begin{cases} f_{ij} & \text{si } t_i \in d_j \\ 0 & \text{en otro caso.} \end{cases}$$

siendo f_{ij} lal frecuencia de palabra i, t_i, en el documento j, d_j, es decir, f_{ij} es el número de veces que aparece la palabra en el documento.

Para este paso podemos tener en cuenta solamente la frecuencia de cada palabra en el documento, obteniéndose así los pesos de cada palabra y en consecuencia aquellas con un peso más alto serán las palabras claves o "keywords".

Mejorando estas medidas podemos hacer *tf-idf*, en el que obtenemos el peso, w_{ij}, de una palabra t_i en el documento d_j como:

$$w_{ij} = tf_{ij} \times idf_i$$

siendo

- tf_{ij} la frecuencia del término

$$tf_{ij} = \frac{f_{ij}}{max\{f_{1j}, ..., f_{mj}\}}$$

- idf_i el inverso de la frecuencia

$$idf_i = log\frac{n}{df_i}$$

  con df_i la frecuencia de documentos d_j en los que el términos t_i aparece.

Finalmente, en este algoritmo se representa cada documento como un conjunto de palabras a las que se le asigna un peso, según su importancia mediante diferentes fórmulas, como puede ser frecuencia o tf-idf.

Basket analysis

Uno de los procedimientos más utilizados en minería web en clasificación de texto o páginas web para la generación de atributos es "Basket analysis". Se trata de una técnica de modelado en base a la teoría de la cesta de la compra, que consiste en que si una persona compra un determinado grupo de productos, es más probable que compre algún producto de un conjunto determinado.

Aplicado al caso de minería web y páginas web encontrar la concurrencia entre ciertas palabras puede ser interesante ya que nos da la posibilidad de encontrar patrones en el lenguaje a la hora de clasificar.

Si lo aplicamos al caso concreto de páginas web, tenemos los siguientes elementos:

- I es el conjunto de palabras clave en todas las páginas web.
- n es el número de páginas web que tenemos.
- I_n es el conjunto de palabras clave en la página web n.
- D es el conjunto de documentos o páginas web.

El procedimiento consiste en establecer un umbral y determinar qué productos lo superan por frecuencia. Para ello consideramos el ratio de las transacciones en las que aparece ese producto entre el número total de transacciones. En nuestro caso, las páginas web en las que aparece una palabra, entre el número total de páginas web. Todas aquellas palabras que superen el umbral serán las más frecuentes. Es decir si f_i es la frecuencia del término t_i.

$$f_i = \frac{|t_i : t_i \in I_j \forall j \in [1, n]|}{|D|}$$

En una segunda parte del algoritmo, tratamos de crear de generar reglas de asociación que tienen un nivel de confianza mayor que un umbral especificado. En esta parte no nos centraremos en las reglas de asociación entre las palabras, ya que no lo tenemos en cuenta para nuestro estudio, pero si nos parece interesante nombrarlo en tanto que puede ser un trabajo a futuro.

3.6.2. Obtención de datos y construcción de patrones

En nuestro caso, lo primero que hemos hecho es la recolección y tratamiento de los datos. Para la recolección hemos obtenido una base de datos de ODP que contiene urls, la descripción de la web y la categoría ODP a la que pertenece.

De ella hemos sustraído aquellas categorías que nos interesaban, ya que no vamos a clasificar todas las categorías ODP, obteniendo finalmente 8871 urls. Dentro de cada categoría hay una jerarquía de subcategorías; en nuestro caso sólo hemos tenido en cuenta la categoría principal para facilitar el análisis, ya que nuestro interés principal es el comportamiento de los algoritmos de SVM.

En nuestro caso, para el tratamiento de los datos hemos usado dos APIS de Textwise [2]. La primera es un API de conceptos, en la cual dada la url nos devuelve

[2] http://www.textwise.com/

	$\mathbf{t_1}$	$\mathbf{t_2}$	...	$\mathbf{t_m}$
$\mathbf{clase_{d_1}}$	w_{11}	w_{12}	...	w_{1m}
$\mathbf{clase_{d_2}}$	w_{21}	w_{22}	...	w_{2m}
	...			
$\mathbf{clase_{d_n}}$	w_{n1}	w_{n2}	...	w_{nm}

Cuadro 3.6.1: Esquema de los patrones para clasificación web

las keyword, más o menos 4 o 5 por url, dándonos el peso de dicha palabra dentro de la página web, con esto construimos los atributos de nuestros patrones, es decir, obtenemos un diccionario para las urls. El peso se determina mediante la frecuencia de esas 4 ó 5 palabras dentro de la url.

La segunda API es para categorización, es decir, dada una url devuelve su categoría ODP, con ella obtendremos el objetivo para cada patrón o url.

Tras este procedimiento hemos procedido a la construcción de los patrones. En los casos descritos anteriormente los algoritmos nos acaban dando un vocabulario V de términos t_i, dónde a cada término t_i se le asocia un peso w_{ij} en el documento d_j. Por tanto nuestro conjunto de atributos estará compuesto por cada uno de los elementos, términos, del conjunto V sin repeticiones. A su vez cada patrón será un documento d_j distinto, en nuestro caso, una página web, en donde a cada atributo se le asigna el peso de ese término en ese documento. Por último la clase será asignada según la categoría a la que pertenezca. La tabla 3.6.1 muestra cómo sería nuestra matriz de patrones.

Finalmente, una vez obtenidos los 8871 patrones con sus clases comenzamos con la parte de aprendizaje automático. Observamos que los patrones clasificados como arte son 6108, mientras que negocios serían 2763, es decir el 68,85 % de la muestra son urls relacionadas con temas de arte y el 31,15 % con negocios.

A continuación procedemos a separar el conjunto de entrenamiento y prueba asignando el 75 % de la muestra para entrenamiento y el 25 % restante para el conjunto de prueba, siendo 6653 y 2218 patrones de entrenamiento y prueba respectivamente. Esta división está hecha de manera aleatoria mediante una distribución uniforme, asegurándonos de tener la proporción similar a la del conjunto completo en cada uno de los conjuntos de prueba y entrenamiento. Por tanto la dimensión del conjunto de entrenamiento tendrá 6653 patrones 22363 atributos, mientras que el conjunto de prueba se compondrá de 2218 patrones 22363 atributos, es decir, podemos hablar de un conjunto de alta dimensionalidad.

3.6.3. Construcción de modelos

Como tenemos un problema de clasificación usaremos los 3 algoritmos para construir los de modelos de SVM. Estos algoritmos son:

- Sequencial Minimization Optimization (SMO) de la libreria LIBSVM.
- Dual Coordinate Descent (DCD) de la libreria LIBLINEAR.
- Stochastic sub-gradient Descent de la libreria Pegasos.

Al tratarse de un problema de alta dimensionalidad esperaremos que estos dos últimos sean más eficientes ya que LIBLINEAR tiene una tasa de convergencia más rápida que la de LIBSVM, y Pegasos, aunque converge más lentamente utiliza un aprendizaje online, por el que el tiempo que emplea para entrenar y precedir es menor, como hemos visto en las secciones anteriores. Por ello, en cada caso vamos a comparar la precisión y el tiempo para entrenar y predecir del algoritmo.

Para mejorar los resultados de nuestro modelo hemos hecho una metamodelización como en el caso de radiación. La diferencia en este caso radica en que sólo necesitamos una rejilla de una dimensión, ya que sólo necesitamos optimizar el parámetro de regularización C. C está en el rango $[2^{-1} : 2^{5}]$ con pasos de 2^{1}, por tanto tenemos 6 iteraciones. Para metamodelizar hemos usado la librería LIBLINEA, que como veremos en la sección de resultados, ha resultado ser mucho más rápida que LIBSVM.

3.6.4. Resultados

Al tratarse de un problema de clasificación usamos diferentes medidas para medir la bondad de nuestro modelo. En este caso tenemos sensitividad, especificidad, accuracy (tasa de acerto), accuracy balanceada, precisión y recall.

- **Sensitividad**: Fracción de positivos reconocidos de la clase positivos. Está dado por la siguiente fórmula.

$$Sen = \frac{TP}{TP + FN}$$

- **Especifidad**: Fracción de negativos reconocidos de la clase negativos. Está dado por la siguiente fórmula.

$$Esp = \frac{TN}{TN + FP}$$

La sensibilidad es la fracción de verdaderos positivos y la especificidad la fracción de verdaderos negativos.

- **Accuracy**: De total de la muestra que fracción clasifico bien tanto de positivos como de negativos.

$$Acc = \frac{TN + TP}{P + N}$$

- **Accuracy balanceada**: Como en el caso de la accuracy mide la fracción que clasifico bien tanto de positivos como de negativos, pero teniendo en cuenta que las clases están desbalanceadas y por tanto no es lo mismo equivocarse con los positivos con los negativos.

$$Acc_B = \frac{Sen + Esp}{2}$$

Cuadro 3.6.2: Resultados de la clasificación web para LIBSVM, LIBLINEAR y Pegasos

	Acc	Acc(b)	Sens	Espec	Prec	Recall	Tiempo
Liblinear	78,40 %	66,43 %	33,66 %	99,20 %	33,66 %	13,63 %	185
Libsvm	79,44 %	68,45 %	38,35 %	98,55 %	38,35 %	15,32 %	6918
Pegasos	85,08 %	79,57 %	64,49 %	94,65 %	64,49 %	24,06 %	235

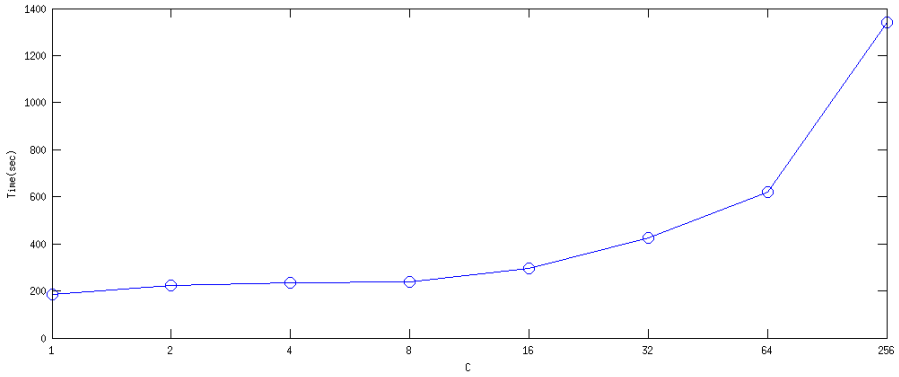

Figura 3.6.1: Gráfica que muestra el parámetro de regularización C frente al tiempo de entrenamiento en segundos

- **Precisión**: Sobre los clasificado como positivos que fracción realmente lo son.

$$Prec = \frac{TP}{TP + FP}$$

- **Recall**: Mide sobre los aciertos cuantos son positivos. Por tanto, un valor alto de recall significa que el algoritmo recupera más casos negativos que positivos.

$$Recall = \frac{TP}{TP + TN}$$

La tabla 3.6.2 muestra los resultados de la clasificación web para LIBSVM, LIBLINEAR y Pegasos.

Para mejorar estos modelos, hemos aplicado búsqueda exhaustiva en rejilla con LIBLINEAR que ha demostrado ser el algoritmo más rápido de los tres.

Por último, a lo largo de los experimentos hemos observado como según aumenta la complejidad del modelo, es decir, C es más grande, el tiempo de entrenamiento aumenta. Para comprobarlo obtenemos la curva que muestra el tiempo de entrenamiento frente al tamaño de C. La gráfica 3.6.1 muestra lo que hemos descrito anteriormente donde C se representa el eje de ordenadas y toma valores entre 2^0 y 2^7.

En esta gráfica observamos como el tiempo aumenta exponencialmente según aumenta la complejidad del modelo.

3.6.5. Conclusiones

En los resultados de la sección previa podemos observar que hemos obtenido son bastante buenos ya que tenemos una precisión de casi del 80 % para LIBLINEAR y

LIBSVM y de un 85 % para Pegasos. Como cabía esperar los resultados en LIBLINEAR y LIBSVM son parecidos ya que ambos resuelven el mismo problema dual. En cambio podemos observar ciertas diferencias con Pegasos, ay que en primer lugar resuelve el problema primal y segundo es un método de aprendizaje online. La propia libreria implenta este aprendizaje online cogiendo uno a uno los patrones.

En este caso, la elección del parámetro C y la metamodelización no ha jugado un papel muy importante pero si que podemos ver en la gráfica 3.6.1 que tiene una gran influencia en el tiempo que es necesario para entrenar los modelos.

Aplicando técnicas de minería web podríamos conseguir más información relevante sobre cada url y por tanto que los patrones tuvieran más información, mejorando los modelos y los resultados.

Capítulo 4

Conclusiones

4.1. Discusión

El objetivo de este trabajo fin de máster ha sido estudiar el estado del arte de las máquinas de vectores soporte tanto en clasificación como en regresión. Para ello hemos analizado la teoría básica de optimización y la teoría clásica de SVM, posteriormente hemos procedido a profundizar en tres de los algoritmos más importantes hoy en día, Sequential Minimal Algorithm, Dual Coordinate Descend Method, y Stochastic sub-gradient descent.

Según hemos concluido en la teoría el método Dual Coordinate Descend convergerá al óptimo en menos iteracciones ya que tiene una tasa de convergencia a una solución con -precisión de $O(log(\frac{1}{\epsilon}))$, mientras que la del método Stochastic subgradient descen es $\tilde{O} = (\frac{1}{\lambda\epsilon})$. Sin embargo aunque la tasa de convergencia al óptimo sea opeor en Pegasos, puede ser un algoritmo muy rápido porque su idea está basada en un aprendizaje online.

En el transcurso de este trabajo hemos utilizado librerías como LIBSVM, LIBLINEAR y Pegasos para cada uno de los tres algoritmos. Hemos implementado una búsqueda exhaustiva en rejilla usando un conjunto de validación en vez de validación cruzada para los experimentos relacionados con la energía solar.

En la parte experimental hemos trabajado con dos problemas completamente diferentes. El primero consiste en un problema de regresión para predecir energía solar. El segundo se trata en un problema de clasificación en el que clasificamos páginas web en dos categorías según su contenido. En esta sección hemos tratado de estudiar que algoritmos son más eficientes y más precisos en cada caso. En el primer problema hemos realizado varios enfoques para mejorar las predicciones dadas por el ECWMF en Alicante y A Coruña. En un principio hemos trabajado sólo con LIBSVM implementando enfoques tanto diarios como trihorarios con patrones compuestos por puntos cercanos a estas dos provincias. En un segundo experimento hemos añadido patrones de toda la Península, también con enfoques diarios y trihorarios y usando dos librerías LIBSVM y LIBLINEAR. Los enfoques trihorarios han resultado mejores en aquellos sitios donde la nubosidad es más baja, pero no se puede concluir que siempre sea así siempre, ya que en predicción de energía solar la variabilidad atmosférica es muy importante. Por último para este problema hemos realizado construido modelos para predecir la radiación total media de la península.

En el segundo problema, hemos hecho una clasificación de páginas web según dos categorías arte y negocios. Para ello hemos usado algunas de las técnicas de minería web para extraer las palabras clave y así poder construir los patrones. Además hemos utilizado tres librerías diferentes LIBSVM, LIBLINEAR y Pegasos, para Sequencial Minimal Algorithm, Dual Coordinate Descend Method, y Stochastic sub-gradient descent, respectivamente. Analizando en cada caso la precisión y el tiempo empleado para entrenar y predecir. Hemos podido concluir que LIBLINEAR es la más rápida y LIBSVM es mucho más lenta para esta cantidad de datos. También hemos analizado cómo influye la complejidad del modelo, es decir el valor de C, en el tiempo que LIBLINEAR emplea para entrenar y predecir.

Por tanto en este trabajo fin de máster hemos analizado los algoritmos más importantes en Máquinas de Vectores Soporte y hemos realizado experimentos con dos problemas muy diferentes que pongan de relieve la potencia de estos algoritmos sobre de Dual Coordinate Descend Method (DCD).

4.2. Trabajos futuros

Como trabajos futuros trataremos de hacer una implementación de Stochastic sub-gradient descent para regresión, ya que el software disponible actualmente es sólo para clasificación.

En cuanto a la parte experimental trataremos de mejorar la metamodelización ya que algunos de los parámetros podrían ser mejor optimizados, así como mejorar nuestras predicciones usando otros conjuntos de validación o incluso haciendo modelos para diferentes horas del día. Por último en la desagregación horaria realizaremos un estudio de modelo clear sky más nuevos que tengan en cuenta más variables meteorológicas.

En los experimentos de clasificación de páginas web, nuestro interés para mejorar los modelos se centrará en la parte de minería web, haciendo un análisis más exhaustivo de las diferentes técnicas para la extracción de los patrones.

Bibliografía

[1] Nello Cristianini and John Shawe-Taylor. An Introduction to Support Vector Machines. Cambridge University Press, 2000.

[2] D. Bertsekas. Constrained optimization and Langrange multiplier methods. Athena Scientific. 1996.

[3] J. Smola and B. Scholkopf. A Tutorial on Support Vector Regression. 2003.

[4] J. López, J.R. Dorronsoro. A Simple Proof of Convergence of the SMO Algorithm for Different SVM Variants. IEEE Transactions on Neural Networks and Learning Systems. 23 (7) (2012), 1142-1147.

[5] J. López, J.R. Dorronsoro. A Simple Proof of the Convergence of the SMO Algorithm for Linearly Separable Problems.

[6] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. ICML '07, Proceedings of the 24th International Conference on Machine Learning, pages 807–814, New York, NY, USA, 2007.

[7] A. Barbero. Efficient Optimization Methods for Regularized Learning: Support Vector Machines and Total–Variation Regularization. PhD thesis. Universidad Autónoma de Madrid, 2011.

[8] John C. Platt. Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[9] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML), 2008.

[10] Chia-Hua Ho and Chih-Jen Lin. Large-scale Linear Support Vector Regression. Journal of Machine Learning Research 13, pages 3323-3348, 2012.

[11] Peng, Xinjun. Projection Support Vector Regression Algorithms for Data Regression. Shangai Normai University.

[12] Chang k. Hsieh C. and Lin C. Coordinate descent method for large-scale L2-loss linear SVM. Journal of Machine Learning Research, 9, 1369 - 1368, 2008.

[13] Z. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. J. Optim. Theory Appl., 72, 7-35, 1992.

[14] Bird, R. E. and Hulstrom, R.L. Simplified Clear Sky Model for Direct and Diffuse Insolation on Horizontal Surfaces. Solar Energy Research Institute, Golden, CO. SERI/TR-642-761. 1981.

[15] S. Aixin, L. Ee-Peng and Wee-Keong Ng. Web Classification Using Support Vector Machine. Proceedings of the 4th international workshop on Web information and data management, Pages 96 - 99.

[16] M. Tsukada, T. Washio and H. Motoda. Automatic Web-Page Classification by Using Machine Learning Methods.