Sistemas de Aprendizaje Automático

Conforme a contenidos del «Curso de Especialización en Inteligencia Artificial y Big Data»

Universidad de Castilla-La Mancha

Escuela Superior de Informática
Ciudad Real

Índice general

Listado de acrónimos

Capítulo 5

Otros Modelos de Aprendizaje Supervisado

Francisco P. Romero

5.1. Regresión Lineal

La **regresión lineal** es un enfoque sencillo de **aprendizaje supervisado** que se emplea para **predecir valores cuantitativos**. La regresión lineal es un enfoque ampliamente utilizado y muchos nuevos enfoques de aprendizaje estadístico son una generalización de esta.

La regresión lineal simple se define de la siguiente manera. Partimos de una observación X para predecir una respuesta cuantitativa Y. Suponemos que existe una relación lineal entre ambas (X e Y), esa relación se puede expresar de la siguiente forma:

$$Y = \beta_0 + \beta_1 X$$

donde β_0 y β_1 son dos constantes desconocidas a priori que representan la intersección (u ordenada en el origen) y la pendiente respectivamente. Juntas (β_0 y β_0) representas los coeficientes o parámetros del modelo. Una vez obtenidas las estimaciones de β_0 y β_1 (representadas como $\hat{\beta}_0$ y $\hat{\beta}_1$ partir de los datos de entrada podemos predecir $\hat{y}$ a partir de un valor particular de x), donde $\hat{y}$ indica una predicción de Y en base a $X = x$. Pero ¿cómo estimamos los coeficientes β_0 y β_1? Imaginemos los siguientes pares de observaciones:

$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

donde n representa el número de observaciones, de las cuales cada una es una medida de X y de Y. El objetivo es encontrar una intersección y una pendiente tales que el modelo lineal resultante se ajuste lo máximo posible a las n observaciones.

Para ello se empleará el enfoque de mínimos cuadrados. Dada $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_i$ la predicción de Y para i-ésimo valor de X y el error $e_i = y_i - \hat{y}_i$ que representa la diferencia entre el i-ésimo valor de la respuesta observada y la predicha. Definimos la suma del error cuadrático (*residual sum of squares*) como:

$$RSS = e_1^2 + e_2^2 + ... + e_n^2 = \sum_{i=1}^{n} (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

Empleamos el enfoque de mínimos cuadrados para seleccionar los parámetros $\hat{\beta}_0$ y $\hat{\beta}_1$ que minimicen el RSS:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

donde $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ e $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Para finalizar, recalcar que probablemente la verdadera relación de los datos de entrada no sea totalmente lineal ya que puede haber otras variables que causen variaciones en Y, por ello la función lineal queda reescrita de esta forma:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

donde ϵ es un error aleatorio irreducible para representar lo que no cubre el modelo [JWHT13].

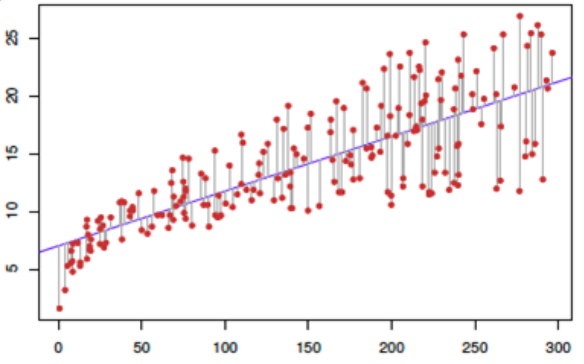

Figura 5.1: Ejemplo de regresión lineal simple ajustada con el enfoque de mínimos cuadrados.

Las hipótesis que debe cumplir este modelo de regresión lineal básico son las siguientes:

- **Esperanza matemática nula**: Para cada valor de X la perturbación tomará distintos valores de forma aleatoria, pero no tomará sistemáticamente valores positivos o negativos, sino que se supone tomará algunos valores mayores que 0 y otros menores que 0, de tal forma que su valor esperado sea 0.
- **Homocedasticidad**: Todos los términos de la perturbación tienen la misma varianza que es desconocida. La dispersión de cada ϵ_t en torno a su valor esperado es siempre la misma.
- **Incorrelación o independencia**: No existe correlación. Las covarianzas entre las distintas perturbaciones son nulas, lo que quiere decir que no están correlacionadas. Esto implica que el valor de la perturbación para cualquier observación muestral no viene influenciado por los valores de las perturbaciones correspondientes a otras observaciones muestrales.
- **Regresores no estocásticos**: Regresores deterministas, que no dependen de valores aleatorios.
- **Independencia lineal**: No existen relaciones lineales exactas entre los Regresores.
- $T > k+1$: Suponemos que no existen errores de especificación en el modelo, ni errores de medida en las variables explicativas.
- **Normalidad**: Las perturbaciones siguen una distribución normal

Si admitimos que los datos presentan estas hipótesis entonces el teorema de Gauss-Markov establece que el método de estimación de mínimos cuadrados va a producir estimadores óptimos, en el sentido que los parámetros estimados van a estar centrados y van a ser de mínima varianza. Este método intenta minimizar la suma de cuadrados de las diferencias en las ordenadas (residuos) entre los puntos generados por la función elegida y los correspondientes valores en los datos. Específicamente, se llama mínimos cuadrados promedio (LMS) cuando el número de datos medidos es 1 y se usa el método de descenso por gradiente para minimizar el residuo cuadrado. La estimación de mínimos cuadrados para modelos lineales es notoria por su falta de robustez frente a valores atípicos (outliers). Si la distribución de los atípicos es asimétrica, los estimadores pueden estar sesgados. En presencia de cualquier valor atípico, los estimadores mínimos cuadráticos son ineficientes y pueden serlo en extremo.

Para la construcción de estos modelos hay que tener muy en cuenta las relaciones entre las variables o características explicativas. Si hay variables que son combinación lineal de las demás, o bien están fuertemente correlacionadas se produce una inestabilidad en la solución del regresor que en general, producirá un aumento en su varianza y por ende de su error. En estos casos se impone la utilización de un método de selección de variables explicativas. A los problemas provocados por la fuerte correlación entre las variables explicativas se les llama multicolinealidad.

5.1.1. Modelos de regresión no lineales

Los modelos lineales pueden tener importantes limitaciones en relación al poder predictivo ya que **el supuesto de linealidad casi siempre se debe a una aproximación y muchas veces esa aproximación no se ajusta de forma adecuada a los datos**. Para conseguir una mejora sustancial en los modelos lineales hay que **relajar el supuesto de linealidad**, para ello existen extensiones de los modelos lineales que se exponen a continuación [JWHT13]:

- **Regresión polinomial**: amplía el modelo de regresión lineal añadiendo predictores adicionales elevando los existentes a una potencia, con ello se consigue un ajuste no lineal a los datos:

$$Y = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_n x_i^n + \epsilon_i$$

  donde ϵ_i es el término de error. Aquí de nuevo los coeficientes pueden estimarse fácilmente empleando los mínimos cuadrados como en la regresión lineal. La regresión polinomial tiene un inconveniente principal y es que para valores muy grandes de n la curva polinómica adopta formas muy raras y por consiguiente no es un buen generalizador. Normalmente se emplean n con valores no superiores a 4.

- **Funciones escalonadas o funciones a trozos**: a diferencia de la regresión polinomial que impone una estructura global en un modelo no lineal, las funciones escalonadas dividen el rango de X en intervalos al que se le ajusta una constante a cada intervalo. Esto es transformar una variable continua en una variable categórica ordenada. Para ello se definen una serie de puntos de corte con la que a partir de el rango de X se construyen $K+1$ variables nuevas, donde K es el número de cortes, estas variables también son conocidas como variables *dummy*.

  Una vez aplicada la transformación a los predictores, empleamos los mínimos cuadrados para ajustar un modelo lineal:

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + ... + \beta_k C_k(x_i) + \epsilon_i$$

  El principal problema de las funciones escalonadas, es seleccionar los puntos de corte.

- **Splines de regresión**: es una técnica de regresión que se emplea para suavizar funciones polinómicas a trozos.

  En lugar de entrenar un modelo de regresión polinómico con un alto grado para que se ajuste a los datos de entrada, la regresión polinómica a trozos implica ajustar polinomios de bajo grado debido a la separación de todo el rango de X en regiones. Esta división del conjunto de datos se llama nudos. El uso de más nudos conduce a polinomios a trozos más flexibles. Como definición general si colocamos K nudos diferentes a lo largo de X, entonces acabaremos ajustando $K+1$ polinomios diferentes.

Cuando dividimos el conjunto de entrenamiento en nudos y ajustamos funciones a cada subconjunto se pueden producir saltos, es decir, que la regresión no sea continua, para evitar esto se puede añadir la restricción de que la curva debe ser continua. Pero esa continuidad no garantiza un ajuste «natural» de la curva, para ello podemos añadir otra restricción y es que además de continua, la curva ha de estar suavizada. Esta ajuste continuo y suavizado a través de los nudos se llama spline. **La definición general de un spline de grado d, es un polinomio de grado d a trozos con continuidad en las derivadas hasta el grado $d-1$ en cada nudo**.

Todos los modelos de regresión estudiados en esta sección son **casos especiales** del enfoque conocido como **funciones base**. La idea subyacente es una familia de funciones o transformaciones aplicadas a los atributos X, por tanto, en lugar de ajustar un modelo lineal en X, se aplican esas transformaciones para adaptarlos a la no linealidad de los datos, por lo que se ajusta el modelo:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + ... + \beta_k b_k(x_i) + \epsilon_i$$

donde $b(X)$ son funciones base bien conocidas.

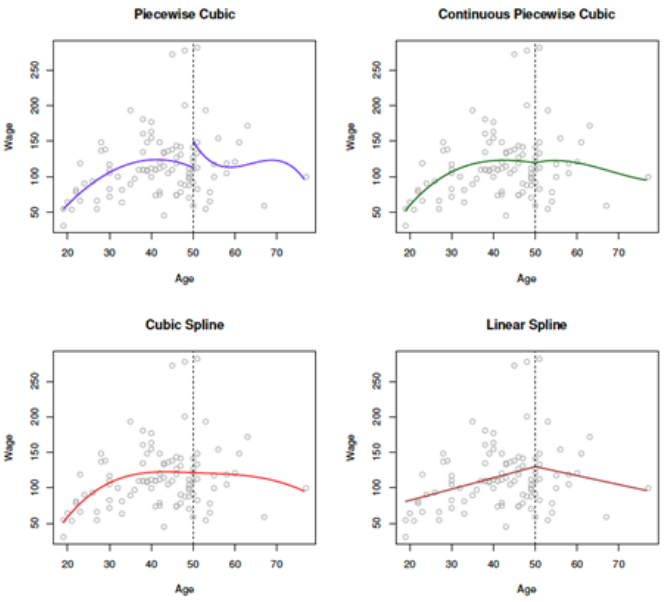

Figura 5.2: Regresiones ajustadas a la no linealidad con y sin splines.

5.2. Regresión logística

La regresión logística es similar a la regresión lineal, pero con la diferencia de que la regresión logística predice si algo pertenece a una clase o no en lugar de valores continuos como hace la regresión lineal.

La regresión logística calcula las probabilidades de pertenencia a una clase y lo hace utilizando tanto atributos discretos como continuos. Esa probabilidad está comprendida entre los valores 0 y 1 siendo 0 la clase negativa y 1 la clase positiva. Para ilustrar este razonamiento imaginemos que queremos predecir si un tumor es benigno o maligno en función de su tamaño. Definimos como clase positiva $(y = 1)$ tumor maligno y como clase negativa $(y = 0)$ tumor benigno.

Nuestro objetivo es estimar la probabilidad de que el tumor sea maligno a partir de su tamaño, matemáticamente $P(X) = P(Y = 1|X : \beta)$ definido como la probabilidad de que $y = 1$ dado una X parametrizada por β. Para representar esto podríamos emplear una regresión lineal de la siguiente forma:

$$P(X) = \beta_0 + \beta_1 X$$

pero esto tiene un problema y es que, si tratamos de ajustar una recta a una respuesta binaria codificada como 0 o 1, en principio se podría predecir $P(X) < 0$ o $P(X) > 1$ y para este tipo de problemas no sería adecuado. Para paliar este problema se puede modelar $P(X)$ usando una función cuya salida esté entre 0 y 1 para todos los valores de X.

En regresión logística se emplea la función sigmoide o función logística:

$$P(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Gracias a esta función se puede definir un umbral de pertenencia a la clase, así por ejemplo si $P(X) >= 0{,}5$ el ejemplo pertenece a la clase 1 y si $P(X) < 0{,}5$ pertenece a la clase cero [JWHT13] [TSK16].

Como se observa en la función logística los parámetros β_0 y β_0 son desconocidos, y deben ser estimados basándose en los datos de entrenamiento disponibles. Si recordamos en el apartado anterior se usó el enfoque de mínimos cuadrados para estimar los parámetros de la regresión lineal. En este caso empleamos el enfoque de máxima verosimilitud cuya idea subyacente es la siguiente: se trata de buscar los valores de $\hat{\beta_0}$ y $\hat{\beta_1}$ tal que al introducirlos en la función logística se obtenga un número cercano a uno para todos los pacientes con tumor maligno y un número cercano a 0 para aquellos con tumor benigno. En la Figura 5.3 se puede ver el ejemplo de clasificación de tumores en función de su tamaño empleando regresión logística y regresión lineal. Se observa que la regresión lineal no se ajusta al problema, ya que si consideramos que lo que se encuentre a la izquierda de la regresión lineal es un tumor maligno y lo que se encuentra a la derecha benigno, se observa que tumores malignos los clasificaría como benignos.

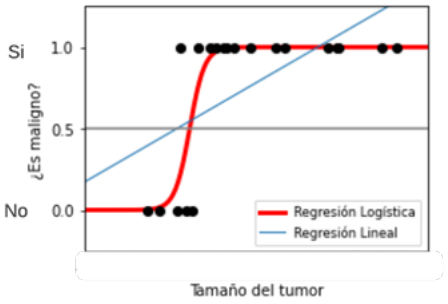

Figura 5.3: Regresión logística vs regresión lineal para la clasificación del tipo de tumor en función del tamaño.

Un coeficiente positivo ($b_k > 0$) para una característica, indica una correlación positiva con la clase predicha, mientras que coeficientes negativos ($b_k < 0$) indican una relación negativa. Los valores absolutos más altos del coeficiente para un campo da lugar a un mayor impacto en las predicciones de ese campo. Esto no debe interpretarse erróneamente como importancia del campo importancia del campo debido a varias razones:

- La importancia del campo en la regresión logística puede definirse como la contribución de un campo a la probabilidad final de la clase, que depende no sólo de la probabilidad de la clase. de clase final, que no sólo depende del coeficiente del campo, sino también de las interacciones con el resto de los campos de entrada. con el resto de los campos de entrada. Dado que el modelo asume la independencia entre las distintas entradas los coeficientes pueden considerarse medidas absolutas de la importancia de los campos sólo cuando todas las entradas son independientes, pero a menudo no es así. En muchos conjuntos de datos reales, el impacto de un campo concreto también depende de los valores de otros campos.
- Las diferentes magnitudes de los campos hacen que los coeficientes sean incomparables. Los coeficientes de campos con diferentes magnitudes, por ejemplo, el salario y la edad, no son comparables, ya que tenderán a ser mayores para los campos con escalas más pequeñas. Cambiar la escala del campo cambiaría el valor del coeficiente. BigML escala automáticamente sus campos numéricos. (Véase la subsección 4.4.8.)
- Los campos pueden ser multicolineales. Si dos campos están muy correlacionados, pueden sustituirse efectivamente el uno por el otro durante el entrenamiento del modelo, por lo que la importancia del campo podría dividirse entre los dos. El mayor sea el número de campos en el conjunto de datos, más probable será que los campos sean multicolineales.

Tras la realización del proceso de entrenamiento y una vez de que se dispone de los coeficientes, se puede utilizar el modelo para hacer predicciones para nuevas instancias. La regresión logística siempre devuelve una probabilidad por clase. La clase clase con la mayor probabilidad será la clase predicha. Teniendo en cuenta las fórmulas anteriores, para un conjunto dado de valores de entrada, $(X_1, X_2, \dots X_k)$, se obtendrán dos probabilidades, una por clase, por ejemplo, $p_1 = 85\,\%$ y $p_2 = 15\,\%$. Cuando hay más de dos clases, lo habitual es ofrecer las probabilidades normalizadas para que la suma de todas las probabilidades de cada predicción de instancia sea igual al 100 %.

Hay que tener en cuenta que por definición, los datos de entrada $(X_1, X_2, \dots X_k)$ en la fórmula de regresión logística deben ser valores numéricos. Sin embargo, se podrá manejar cualquier tipo de atributos aplicando un conjunto de transformaciones a los campos categóricos, de texto y de elementos.

5.2.1. Características representativas

Algunas características representativas de la regresión logística son las siguientes [TSK16]:

- **Pueden extenderse fácilmente a clasificación multiclase**. Esto se conoce como **regresión logística multinomial**.
- **Los parámetros (coeficientes) aprendidos en la regresión logística pueden emplearse para entender las relaciones entre los atributos y la clase**.
- **La regresión logística es robusta frente a espacios de alta dimensionalidad** ya que no implica cálculo de densidades o distancias que son muy costosos.
- **La regresión logística es robusta frente a atributos irrelevantes y frente a atributos redundantes**. Sin embargo, la presencia de estos atributos en entornos de alta dimensionalidad puede llevar al sobreaprendizaje.
- Por último, **la regresión logística no puede manejar instancias con valores perdidos**, ya que como hemos visto las probabilidades se calculan mediante la suma ponderada de todos los atributos.

La *binary crossentropy loss function* (función de pérdida binaria de entropía cruzada) calcula la pérdida de un ejemplo mal clasificado calculando la siguiente media:

$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

donde $\hat{y}_i$ es el i-ésimo valor escalar en la salida del modelo, y_i y *output size* es el número de valores escalares en la salida del modelo.

Listado 5.1: Implementación de la Regresión Logística en Python

```
class LogisticRegression:
  def __init__(self, learning_rate=0.1, number_of_iterations=1000, verbose=False):
    self.learning_rate = learning_rate
    self.number_of_iterations = number_of_iterations
    self.weights = None
    self.bias = None
    self.loss = []

  # Inicialización de pesos y parámetos
  def initialize_parameters(self, n): # n ->número de características
    self.weights = np.zeros(n)
    self.bias = 0

  #Calcula la función de pérdida
  def binary_cross_entropy(self, y, y_hat):
    m = y.shape[0]
    return -(1/m) * np.sum(y*np.log(y_hat) + (1-y)*np.log(1-y_hat))

  def sigmoid(self, z):
    return 1 / (1 + np.exp(-z))

  def fit(self, X, y):
    m = X.shape[0] # numero de elementos de entrenamient
    n = X.shape[1] # numero de características
    # inicializar los valores de los pesos
    self.initialize_parameters(n)
    for i in range(0, self.number_of_iterations):

      # calcular resultado utilizando la función lineal y la sigmoide
      z = np.dot(X, self.weights) + self.bias
      y_hat = self.sigmoid(z) + 0.00001 # añadir 0.00001 para evitar 0 en el logaritmo

      # calcular la pérdida
      loss = self.binary_cross_entropy(y, y_hat)
      self.loss.append(loss)

      # calcular las derivadas parciales
      dw = (1/m)*(2*np.dot(X.T, (y_hat-y)))
      db = (1/m)*(np.sum(y_hat-y))

      # actualizar peso y coeficiente independiente
      self.weights -= self.learning_rate*dw
      self.bias -= self.learning_rate*db

  # calcular las probabilidades de cada clase a partir de lo aprendido
  def predict_proba(self, X):
    z = np.dot(X, self.weights) + self.bias
    return self.sigmoid(z)

  # predecir el resultado final basándonos en el umbral y las probabilidades de cada
      clase
  def predict(self, X, theta=0.5): # theta ->umbral
    probabilities = self.predict_proba(X)
    predictions = [1 if i>theta else 0 for i in probabilities]
    return predictions
```

5.2.2. Proceso

Los pasos a seguir para realizar regresión logística son los siguientes:

- **Búsqueda de variables correlacionadas con la variable objetivo**. Con el fin de seleccionar las variables relevantes al problema, es decir, aquellas que más aportan en la explicación de la variable objetivo.
- **Discretización de variables numéricas**: Con el fin de mejorar el rendimiento del modelo es necesario discretizar las variables contínuas de entre las seleccionadas en el paso anterior. Para ello se pueden utilizan diferentes métodos: división por intervalos, por densidad, etc.
- **Entrenamiento del modelo inicial** (sin interacciones): En este caso se utiliza el principio jerárquico, que consiste en que si en el modelo de regresión logística se incluye un término cualquiera, todos sus términos de menor orden deben de permanecer en el modelo, y que si se elimina un término cualquiera, todos los términos de mayor orden en los que intervenga también deben sacarse del modelo. Para ello, se construye un modelo inicial solamente con las variables explicativas que han resultado significativas y posteriormente se analiza si las interacciones entre ellas son significativas.
- **Entrenamiento del modelo** (con interacciones): Una vez que se obtiene un modelo válido con los efectos principales de las variables, se analiza si hay interacciones entre ellas que resulten significativas y se mejore por lo tanto la explicación de la variable objetivo.
- **Entrenamiento del modelo jerárquico** (sin selección de variables): Ya sabemos cuáles son las variables que más aportan en la explicación de la variable dependiente. Ahora bien, en el modelo obtenido en el paso anterior, puede haber variables para los que es significativa su interacción con otra variable pero no lo es su efecto principal. Por lo tanto, se puede ajustar un modelo sin selección de variables, donde estimemos los coeficientes de las siguientes variables que nos interesa incluir en el modelo.
- **Elección del mejor modelo**: Después de estimar diferentes modelos es necesario elegir el mejor de ellos lo cual se realizará a partir del uso de las métricas de éxito mencionadas anteriormente.
- **Evaluación**: Consiste en medir los resultados del modelo seleccionado de entre los calculados con el conjunto seleccionado para la evaluación o test. De esta forma se asegura que el modelo no se ajusta exclusivamente a los datos utilizados en su construcción. Se contrastará la capacidad predictiva del mismo con dichos datos de test, de forma que con este conjunto de datos se tiene que llegar a las mismas conclusiones que las que obtuvieron con los datos de entrenamiento. En caso de ser así, ya se tendría el modelo definitivo construido a partir de la regresión logística.

5.3. Máquinas de Vectores de Soporte (*Support Vector Machine*)

Las Máquinas de Soporte Vectorial o Support Vector Machine (SVM) es un algoritmo de aprendizaje estadístico que apareció por primera vez en 1992 en un artículo escrito por Vladimir Vapnik y sus colegas Bernhard Boser e Isabelle Guyon, aunque los fundamentos en los que se basa existen desde los años 60 [HPK11]. Las SVM han demostrado tener un buen rendimiento a la hora de clasificar, y son consideradas uno de los mejores clasificadores de «caja negra» (no explicables).

La SVM es una generalización de un clasificador más simple, llamado clasificador de Margen Máximo (Maximal Margin Classifier) fundamentado en el concepto de **hiperplano**, pero ¿qué es un hiperplano? **En un espacio n-dimensional, un hiperplano se define como el subespacio afín de dimensión** $n^{\smile}1$. Por ejemplo, si tenemos un espacio bidimensional, aquí el hiperplano sería un subespacio de una dimensión, es decir, una línea. En tres dimensiones, el hiperplano sería un subespacio de dos dimensiones, y así con el resto de las dimensiones, aunque para dimensiones mayores que tres, ya es más difícil de imaginar.

5.3.1. Clasificador de margen máximo (*Maximal Margin Classifier*)

En el caso de que nuestros datos puedan ser separados por un hiperplano, entonces existirá un número infinito de hiperplanos que puedan separar nuestros datos sin entrar en contacto con ninguna de las observaciones, entonces, ¿qué hiperplano empleamos para separar nuestros datos? Pues una forma es emplear el hiperplano de margen máximo, que es, el hiperplano de separación entre las dos clases que está más alejado de las observaciones de entrenamiento (mayor distancia mínima entre las observaciones). Con este hiperplano podemos clasificar un ejemplo de prueba basándonos únicamente en el lado del hiperplano en el que se encuentra, esto es, el clasificador de margen máximo. Como se observa en la Figura 5.4, el hiperplano es dependiente de las observaciones que se encuentran equidistantes (conocidas como vectores de soporte) y cualquier cambio en estas observaciones cambiarían el hiperplano de margen máximo, mientras que un cambio en el resto de las variables no afectaría al hiperplano (a no ser que atravesasen la frontera que describe el hiperplano) [JWHT13].

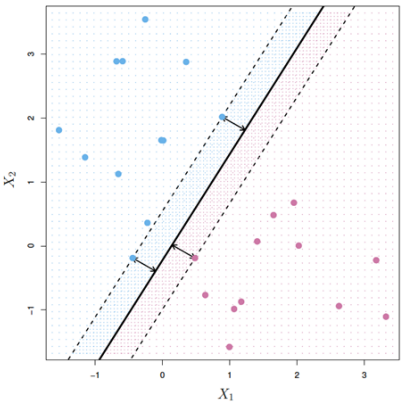

Figura 5.4: Hiperplano de margen máximo y vectores de soporte.

5.3.2. Clasificadores de soporte vectorial (*Support Vector Classifiers*)

Como se puede imaginar el lector, en muchas ocasiones no es posible separar el conjunto de entrenamiento en dos clases de una forma tan sencilla, porque no existe un hiperplano de separación. Pero se puede extender el concepto de hiperplano para que este casi separe las clases, llamado de margen suave. Esta generalización del caso claramente separable a uno no separable mediante un hiperplano de margen máximo se conoce como clasificador de soporte vectorial (*Support Vector Classifier*).

En este algoritmo se permite no solo que algunas observaciones estén dentro del margen, si no que también estén en lado incorrecto del hiperplano (estas últimas estarían mal clasificadas). Esto le da al algoritmo una mayor robustez frente al sobreaprendizaje a no depender únicamente de unas pocas observaciones.

5.3.3. Maquinas de soporte vectorial (Support Vector Machines)

En el mundo real es difícil encontrar datos que sean separables linealmente. Con el fin de resolver este problema nace la Máquina de Soporte Vectorial. La idea que subyacente es la siguiente, se necesita transformar los datos para poder buscar un hiperplano que separe las clases de forma lineal para ello se pueden asignar los datos a un espacio más complejo con características no lineales (por ejemplo, aplicando un polinomio cuadrado), donde poder usar un clasificador lineal en ese espacio, es decir, se trasladan los datos a otro espacio de características linealmente separables. Esto nos evita tener que expresar funciones no lineales muy complejas,

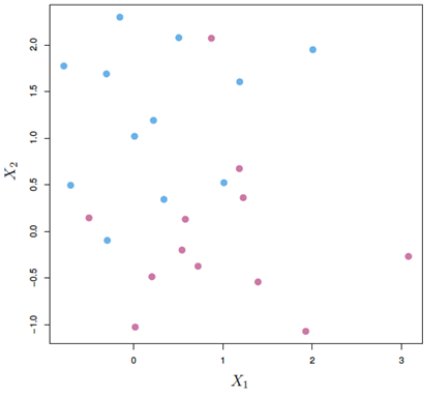

Figura 5.5: Distribución de puntos en el plano, no separables por un hiperplano.

pero, es fácil observar que al ampliar el espacio de características se puede acabar con un gran número de estas lo que implicaría cálculos inmanejables y complicaría la obtención de funciones de decisión generales (que no sobreajusten) por la alta dimensionalidad [TSK16].

Para solucionar este problema entran en juego los kernels, que no es simplemente más que un enfoque computacional eficiente para poder operar en espacios de alta dimensionalidad. Más formalmente, una función kernel devuelve el valor del producto escalar entre las imágenes de dos argumentos, es decir, todos los cálculos se realizan en el espacio de entrada original (que tiene una dimensionalidad potencialmente menor). Existen varios tipos de kernel [JWHT13]:

- Kernel polinomial:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

  donde d es el grado del polinomio.

- Kernel de base radial:

$$K(x_i, x_{i'}) = e^{-\alpha \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

  donde α es una constante positiva.

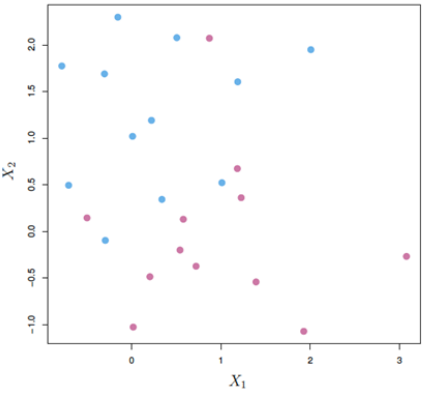

Figura 5.6: SVM con kernel de base radial.

5.3.4. Extensiones del SVM

Inicialmente hemos partido de un problema de clasificación binaria (los objetos pertenecen a dos grupos). Esta técnica ha sido extendida para problemas de clasificación de varios grupos. Una primera forma de extender el SVM binario a multi grupos es construir clasificadores binarios para todos los pares posibles de grupos. El objeto es clasificado en la clase más frecuentemente predicha. (" 'Max Wins'). La otra opción es extender la formulación problema C-SVM para varias clases. Este método se le conocee como `spoc-svc`.

Otra aplicación del SVM es the **detección de novedades** o clasificación en una única clase. En esencia lo que se trata es detectar otuliers en una muestra. Se construye una frontera esférica alrededor de los datos y se detecta una fracción de outliers. Estos métodos trabajan con el parámetro ν que controla una cota superior a la fracción de outliers detectados.

El SVM se ha aplicado al problema de regresión. En este caso las variables y_i dejan de tener el valor ± 1 y toman números reales. El objetivo es obtener la función (de regresión) de decisión $f(\mathbf{x})$. Se ha empleado como **función de pérdida** en lugar de los mínimos cuadrado, $\|y - f(x)\|_\varepsilon = \max\{0, \|y - f(x)\| - \varepsilon\}$. Con esta función se crea un tubo alrededor de los datos predichos y se intenta encajarlo dentro de las observaciones.

5.3.5. SVM en Scikit-Learn

Scikit-learn cuenta con diferentes implementaciones de las Support Vector Machines. La principal es **SVC** aunque existe una alternativa **NuSVC**, similar pero aceptan conjuntos de parámetros ligeramente diferentes y tienen formulaciones matemáticas distintas (por ejemplo un párametro para controlar el número de vectores de soporte). Por otro lado, **LinearSVC** ofrece una implementación más rápida para el caso de uso de un *kernel* lineal.

Los parámetros más habituales son los siguientes:

- **Kernel**: La función principal del núcleo es transformar los datos de entrada del conjunto de datos en la forma requerida. Hay varios tipos de funciones, como la lineal, la polinómica y la de base radial (RBF). Las funciones polinómicas y RBF son útiles para los hiperplanos no lineales. Los núcleos polinómicos y RBF calculan la línea de separación en la dimensión superior. En algunas aplicaciones, se sugiere utilizar un kernel más complejo para separar las clases que son curvas o no lineales. Esta transformación puede conducir a clasificadores más precisos.
- **Regularización**: El parámetro C se utiliza para establecer el mecanismo de margen/regularización. C es el parámetro de penalización, que representa la mala clasificación o el término de error. El término de error o clasificación errónea establece el nivel de error asumible en el proceso de optimización. De esta forma se controla el equilibrio entre el límite de decisión y el término de clasificación errónea. Un valor menor de C crea un hiperplano de margen pequeño y un valor mayor de C crea un hiperplano de margen mayor.
- **Gamma**: Modela la parametrización del kernel. Un valor más bajo de *Gamma* se ajustará de forma imprecisa al conjunto de datos de entrenamiento, mientras que un valor más alto de gamma se ajustará exactamente al conjunto de datos de entrenamiento, lo que provoca un sobreajuste. Es decir, se puede decir que un valor bajo de gamma considera sólo los puntos cercanos en el cálculo de la línea de separación, mientras que el valor de gamma considera todos los puntos de datos en el cálculo de la línea de separación.

Listado 5.2: Utilización del clasificador SVC con optimización de hiperparámetros en Grid

```
1 from sklearn import svm, datasets
2 from sklearn.model_selection import GridSearchCV
3 iris = datasets.load_iris()
4 parameters = {'kernel':('linear', 'rbf'),
5               'C':[1, 10],
6               'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1]}
7 svc = svm.SVC()
8 clf = GridSearchCV(svc, parameters)
9 clf.fit(iris.data, iris.target)
```

Listado 5.3: Utilización del clasificador SVC con scikit-learn.

```
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

# preproceso: de imagenes a vectores
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
# division entre train y test
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.8, shuffle=False)
# crear/definir el clasificador
clf = svm.SVC(gamma=0.001)
# entrenamient del modelo
clf.fit(X_train, y_train)
# prediccion sobre test
predicted = clf.predict(X_test)
# Visualización
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, prediction in zip(axes, X_test, predicted):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title(f'Prediction: {prediction}')
# Evaluación
print(f"Classification report for classifier {clf}:\n"
      f"{metrics.classification_report(y_test, predicted)}\n")
```

5.3.6. Ventajas e inconvenientes

Las principales ventajas de las SVM son las siguientes:

- El modelo está basado en unos pocos vectores de soporte lo cual hace que sea ligero y ocupe poca memoria. Esto es, una vez entrenado el modelo, la fase de predicción es muy rápida.
- Como sólo se ven afectados por los puntos cercanos al margen, funcionan bien con datos de alta dimensión.
- Su integración con métodos de transformación kernel los hace muy versátiles, capaces de adaptarse a muchos tipos de datos.

Sus desventajas principales son:

- El escalado con el número de muestras N es $O[N^3]$ en el peor de los casos, o $O[N^2]$ para implementaciones eficientes. Para un gran número de muestras de entrenamiento, este coste computacional puede ser prohibitivo.
- Los resultados no tienen una interpretación probabilística directa.
- Los resultados dependen en gran medida de una elección adecuada del parámetro de suavización C .

Bibliografía

[HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[TSK16] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.