Temas Selectos de Ciencia de Datos: Big Data

Diego Godinez Bravo

Centro de Investigación en Matemáticas

Maestría en Cómputo Estadístico

Octubre 2024

Máquina de Soporte Vectorial (SVMs)

El algoritmo Máquina de Soporte Vectorial (SVM) es un método de aprendizaje supervisado utilizado para problemas de clasificación y regresión. El objetivo principal es encontrar el hiperplano que separe de manera óptima un conjunto de datos en clases distintas.

Qué es un Hiperplano?

En un espacio de dimensión p, un **hiperplano** es un subespacio de dimensión $p-1$. De manera que, en un espacio de dos dimensiones, un hiperplano corresponde a una recta. En un espacio de tres dimensiones el hiperplano corresponde a un plano, es decir, un subespacio de dos dimensiones. Conforme aumenta el número de dimensiones ($p > 3$), pensar en la visualización de un hiperplano se vuelve más abstracto. Sin embargo, el concepto general se mantiene: un subespacio de dimensión $(p-1)$.

En un espacio de dos dimensiones, un hiperplano se define mediante la siguiente ecuación

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

la cual corresponde a la ecuación de una recta.

La expresión anterior se puede extender para p dimensiones

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

de manera que si un punto $X = (X_1, X_2, ..., X_p)^T$ satisface la ecuación, entonces X se encuentra en el hiperplano.

Supongamos que un punto X no satisface la ecuación del hiperplano, en su lugarse cumple que

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0$$

En este caso, podemos pensar que X se encuentra en un lado del hiperplano. Por otro lado, si

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0$$

entonces X se encuentra en el lado opuesto del hiperplano.

De este modo, podemos pensar en un hiperplano como un subespacio que divide el espacio p-dimensional en dos regiones.

Clasificación Utilizando un Hiperplano

Consideremos una matriz de datos X, la cual consiste de n observaciones en un espacio de p dimensiones

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

donde cada observación pertenece a una de dos clases, es decir, $y_1, ..., y_n \in \{-1, 1\}$, donde -1 representando una clase y 1 la otra.

Consideremos un vector de caracteristicas observado $x^* = (x_1^*, ..., x_p^*)^T$. Supongamos que existe un hiperplano capaz de separar las observaciones en la matriz X de acuerdo con sus etiquetas y_i. De manera que,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0 \text{ si } y_i = 1$$

y

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \text{ si } y_i = -1$$

En conclusión, si existe un hiperplano, puede utilizarse para clasificar las observaciones: cada observación se asignada a una clase en función del lado del hiperplano en el que se encuentre.

Las observaciones de prueba x^* se clasifican en función del signo de

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$$

Si un conjunto de datos puede ser clasificado utilizando un hiperplano, entonces existe un número infinito de hiperplanos que pueden ajustarse para cumplir este objetivo. Una opción es seleccionar el hiperplano que maximice la distancia a las observaciones de entrenamiento. En otras palabras, podemos utilizar la distancia perpendicular de cada observación hasta el hiperplano de separación; la menor de estas distancias se conoce como *margen*. El hiperplano de margen máximo es aquel para el cual el margen es mayor, es decir, el que tiene la mayor distancia mínima respecto a las observaciones de entrenamiento. Por lo tanto podemos clasificar las observaciones de acuerdo con el lado del hiperplano al que pertenezcan, lo que se conoce como 'Clasificador de Máximo Margen' (*Maximal Margin Classifier*).

En un clasificador de máximo margen, hay puntos en el espacio de características que están más cerca del hiperplano. Estas observaciones son esenciales porque "sostienen.[el] hiperplano, es decir, un ligero desplazamiento de ellas provocaría el movimiento del hiperplano. Por esta razón, se les conoce como 'vectores de soporte' (*support vectors*). Los vectores de soporte son fundamentales para la definición del margen, ya que determinan su posición y, por ende, influyen en el proceso de clasificación en el espacio de p dimensiones. En otras palabras, el hiperplano de margen máximo depende directamente de los vectores de soporte.

Construcción del Clasificador de Máximo Margen

El hiperplano de margen máximo es la solución al problema de optimización

$$\begin{aligned} &\text{maximize} \quad M \\ &\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \\ y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) &\geq M \quad \forall i = 1, \ldots, n. \end{aligned}$$

Las restricciones aseguran que cada observación se encuentre en el lado correcto del hiperplano a una distancia mínima de M del hiperplano. De manera que M representa el margen del hiperplano, y el problema elige los valores $\beta_0, \beta_1, ..., \beta_p$ que maximizan M.

Sin embargo, en la mayoría de los casos no existe un hiperplano que separe las clases de manera perfecta. Los 'clasificadores de vectores de soporte' permiten que algunas observaciones se encuentren en el lado incorrecto del margen o incluso del hiperplano. Este enfoque permite que ciertas observaciones de entrenamiento violen el margen sin comprometer el rendimiento general del clasificador.
De manera que el problema de optimización a resolver es ahora

$$\begin{aligned} &\text{maximize} \quad M \\ &\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \\ y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) &\geq M \quad \forall i = 1, \ldots, n \\ \epsilon_i \geq 0, &\sum_{i=1}^{n} \epsilon_i \leq C \end{aligned}$$

donde C es un parametro de ajuste no negativo.

Métodos de Ensamblaje

Un método de ensamblaje es un enfoque que combina múltiples modelos "simples"para generar un único modelo de ensamblaje que puede ser muy potente. Estos modelos simples son conocidos como 'weak learners', ya que tienden a realizar predicciones débiles por sí solos.

Bagging

Bootstrap aggregation (*bagging*) es un procedimiento de uso general para reducir la varianza de un método de aprendizaje estadístico. Dado un conjunto de n observaciones independientes $Z_1, ..., Z_n$ cada una con varianza σ^2, la varianza de la media $\bar{Z}$ de las observaciones está determinada por σ^2/n. Es decir, promediar un conjunto de observaciones reduce la varianza. Por lo tanto, podríamos pensar que una forma de reducir la varianza y aumentar la precisión del conjunto de prueba de un método de aprendizaje estadístico sería tomar múltiples conjuntos de entrenamiento de la población, construir un modelo de predicción para cada conjunto de entrenamiento y promediar las predicciones resultantes.

Es decir, podemos calcular $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ utilizando B conjuntos de entrenamiento diferentes, promediar sus resultados y obtener un único modelo con una varianza reducida

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

Sin embargo, el problema principal radica en la falta de múltiples conjuntos de entrenamiento. Por lo tanto, podemos emplear el método de bootstrap que permite generar muestras repetidas a partir de un único conjunto de entrenamiento. De esta manera, se generan B diferentes muestras bootstrap de entrenamiento, y posteriormente se entrena el modelo utilizando cada una de estas B muestras para obtener $\hat{f}^{*b}(x)$. Finalmente, se promedian las predicciones

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Este enfoque es conocido como *bagging*.

Para aplicar el método a árboles de regresión, simplemente se generan B árboles utilizando B muestras bootstrap de entrenamiento y se promedian sus predicciones. A pesar de que cada árbol individual presenta una varianza alta y un sesgo pequeño, al promediar los B árboles se reduce la varianza. *Bagging* es un método eficaz que mejora significativamente la precisión al combinar múltiples árboles (cientos o incluso miles) en un solo procedimiento.

Random Forests

Al igual que *bagging*, se construye una serie de árboles de decisión a partir de muestras bootstrap del conjunto de entrenamiento. Supongamos que existe un predictor extremadamente fuerte en los datos, junto con varios predictores moderadamente fuertes. En este escenario, la mayoría de los árboles en la colección probablemente utilizarían este predictor como la raíz principal. Como resultado, todos los árboles generados tendrían estructuras similares, lo que provocaría que sus predicciones estén altamente correlacionadas. Promediar predicciones que son altamente correlacionadas no conduce a una reducción significativa de la varianza, a diferencia de lo que ocurre al promediar cantidades no correlacionadas. Es decir, en este caso, el método *bagging* no logra una reducción significativa de la varianza.

El método de Random Forests aborda este problema al forzar que cada división considere solo un subconjunto aleatorio de predictores. En promedio, $(p-m)/p$ de las divisiones no tomarán en cuenta el predictor más fuerte, lo que permite que otros predictores tengan la oportunidad de ser seleccionados. Este proceso de "decorrelación de árboles"genera un conjunto de árboles con estructuras diferentes,

lo que reduce la dependencia entre ellos. Al promediar las predicciones de estos árboles, se disminuye la variabilidad, proporcionando una estimación más confiable y robusta.

Redes Neuronales

Una Red Neuronal toma un vector de entrada $X = (X_1, X_2, ..., X_p)$ de p variables y construye una función no lineal $f(X)$ para predecir la respuesta Y. Las p características $X_1, X_2, ..., X_p$ constituyen la *capa de entrada*, donde cada una de ellas alimenta a las K unidades que componen la *capa oculta*. El modelo de la red neuronal de una sola capa se presenta en la siguiente manera

$$\begin{aligned} f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j) \end{aligned}$$

La *capa oculta* calcula las activaciones $A_k = h_k(X)$ que son transformaciones no lineales de combinaciones lineales de las entradas $X_1, X_2, ..., X_p$

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

donde $g(z)$ es una función de activación no lineal.

Podemos considerar cada una de estas activaciones A_k como una transformación diferente $h_k(X)$ de las características originales. Sin embargo, estas activaciones A_k no son observadas directamente. Las K activaciones de la *capa oculta* alimentan a la *capa de salida*, dando como resultado

$$f(X) = \beta_0 + \sum_{K}^{k=1} \beta_k A_k$$

un modelo de regresión lineal con K activaciones. Todos los parámetros $\beta_0, ..., \beta_k$ y $w_{10}, ..., w_{Kp}$ necesitan ser estimados a partir de los datos.

En sus inicios, la función sigmoide de activación fue ampliamente utilizada

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

Sin embargo, en la actualidad se prefiere la función de activación *ReLU* (*rectified linear unit*), definida como

$$g(z) = (z)_+ = \begin{cases} 0 \text{ if } z < 0 \\ z \text{ otherwise} \end{cases}$$

la cual puede ser calculada y almacenada de manera más eficiente que la función de activación sigmoide.

De manera general, el modelo deriva K nuevas características al calcular K combinaciones lineales diferentes de X, luego transforma cada una de ellas mediante una función de activación $g(\ )$. El modelo resultante es lineal en relación con estas variables derivadas, es decir, la salida del modelo es una combinación lineal de estas nuevas características.

El añadir una función de activación no lineal permite que el modelo capture relaciones no lineales complejas y efectos de interacción. Estas funciones de activación no lineales son fundamentales, ya que sin ellas el modelo resultante se limitaría a ser un modelo lineal simple.

Ajustar un modelo de red neuronal implica la estimación de los parámetros desconocidos $\beta_0, ..., \beta_k$ y $w_{10}, ..., w_{Kp}$. En el caso de una respuesta cuantitativa, los parámetros se seleccionan minimizando

$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

Redes Neuronales Convolucionales (CNNs)

Las Redes Neuronales Convolucionales (CNNs) imitan en cierta medida la forma en que los humanos clasifican imágenes al reconocer características o patrones específicos en cualquier parte de la imagen que distinguen cada clase de objeto en particular. Es decir, el modelo toma una imagen e identifica sus características locales o características de bajo nivel, las cuales combina para crear características compuestas o características de alto nivel. De manera que la presencia o ausencia de estas características de nivel superior contribuye a la probabilidad de cualquier clase de salida dada.

La estructura de una red neuronal convolucional consta de dos tipos especializados de *capas ocultas*, llamadas *capas convolucionales* (*convolution layers*) y *capas de agrupamiento* (*pooling layers*).

Convolution Layers

Una *capa de convolución* se conforma por una gran cantidad de *filtros de convolución*, donde cada uno de ellos se encarga de determinar si una característica local particular está presente en la imagen. Un *filtro de convolución* se basa en una operación llamada *convolución*, la cual consiste en multiplicar repetidamente los elementos de la matriz para posteriormente sumar los resultados.

Consideremos una imagen de dimensión 4×3

$$Image = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix},$$

y un filtro de 2×2

$$Convolution\ Filter = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

Cuando realizamos la *convolución* obtenemos una imagen convolucionada

$$Convolved\ Image = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & d\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

De manera que la imagen convolucionada resalta regiones de la imagen original que comparten características con el filtro de convolución. Es decir, la imagen convolucionada destaca las regiones de la imagen original que corresponden a las características que el filtro busca, lo que facilita la identificación de patrones o características específicas.

Normalmente se aplica la función de activación *ReLU* a la imagen convolucionada. Este paso se considera como una capa adicional en el modelo, llamada *capa detectora*.

Pooling Layers

La *capa de agrupamiento* proporciona una forma de condensar una imagen grande en una imagen más pequeña. Existen distintas formas de realizar el agrupamiento, una de ellas es el agrupamiento máximo (*max pooling*), la cual resume cada bloque de píxeles no superpuesto de tamaño 2×2 utilizando el valor máximo del bloque. Reduciendo el tamaño de la imagen por un factor de 2 en cada dirección

$$Max\ pool \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

Arquitectura de una Red Neuronal Convolucional

Consideremos un conjunto de datos que consta de imágenes de tamaño 32×32.

La arquitectura de una *CNN* comienza en la capa de entrada, donde se presenta el mapa de características tridimensional de una imagen a color, en la que el eje de los canales representa cada color mediante un mapa de características bidimensional de píxeles de 32×32.

La primera capa de convolución aplica múltiples filtros de convolución, donde cada uno de ellos produce un nuevo canal en la primera capa oculta, cada uno de los cuales es un mapa de características de 32×32 (después de aplicar algún tipo de padding en los bordes). Esto resulta en una nueva "imagen", es decir, un mapa de características que contiene considerablemente más canales que los tres canales de color de la entrada.

Después de la primera capa de convolución, se incorpora una capa de agrupamiento (*max-pool layer*), que reduce las dimensiones de los mapas de características en cada canal por un factor de 4 (dos en cada dimensión). Cada capa de convolución subsecuente toma como entrada el mapa de características resultante de la capa anterior y lo trata como una imagen de múltiples canales, utilizando filtros que tienen tantos canales como el mapa de características de entrada.

A medida que los mapas de características se reducen en tamaño después de cada capa de agrupamiento, generalmente se aumenta el número de filtros en la siguiente capa de convolución para compensar esta reducción. Este proceso se repite hasta que el agrupamiento ha reducido cada mapa de características a solo unos pocos píxeles en cada dimensión.

Referencias

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning with applications in Python*. Springer Texts in Statistics. https://doi.org/10.1007/978-3-031-38747-0