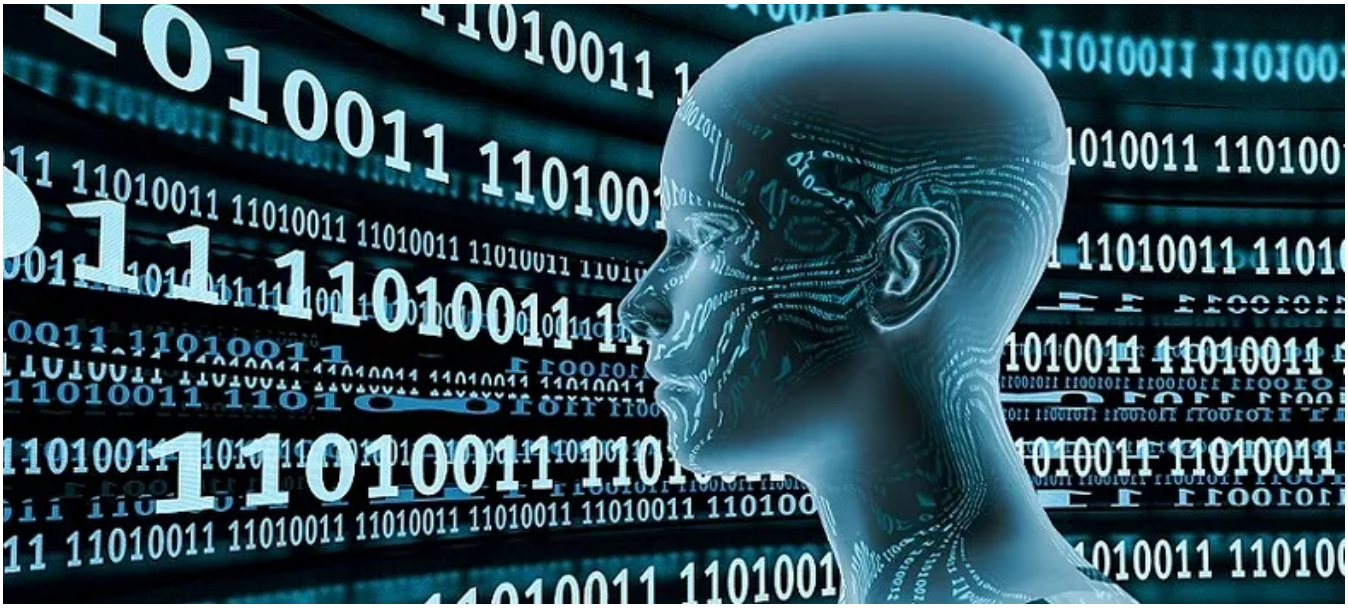
Este artículo presenta los conceptos básicos de la teoría del aprendizaje automático y establece los conceptos y técnicas comunes involucrados. Esta publicación está destinada a las personas que comienzan con el aprendizaje automático, lo que facilita seguir los conceptos básicos y familiarizarse con los conceptos básicos del aprendizaje automático.



Fuente

¿Qué es el aprendizaje automático?

En 1959, **Arthur Samuel** , un científico informático que fue pionero en el estudio de la inteligencia artificial, describió el aprendizaje automático como “el estudio que otorga a las computadoras la capacidad de aprender sin ser programadas explícitamente”.

El artículo seminal de Alan Turing (**Turing,** 1950) introdujo un estándar de referencia para demostrar la inteligencia de las máquinas, de modo que una máquina debe ser inteligente y receptiva de una manera que no se puede diferenciar de la de un ser humano.

> Machine Learning es una aplicación de inteligencia artificial en la que una computadora/máquina aprende de las experiencias pasadas (datos de entrada) y hace predicciones futuras. El

> rendimiento de dicho sistema debe ser al menos a nivel humano.

Una definición más técnica dada por **Tom M. Mitchell** (1997): "Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y mide el desempeño P, si su desempeño en tareas en T, medido por P, mejora con la experiencia E." Ejemplo:

> **Un problema de aprendizaje de reconocimiento de escritura a mano:**
>
> **Tarea T :** reconocimiento y clasificación de palabras escritas a mano dentro de imágenes
> **Medida de desempeño P :** porcentaje de palabras correctamente clasificadas, precisión
> **Experiencia de capacitación E :** un conjunto de datos de palabras escritas a mano con clasificaciones dadas

Para realizar la tarea T, el sistema aprende del conjunto de datos proporcionado. Un conjunto de datos es una colección de muchos ejemplos. Un ejemplo es una colección de características.

Categorías de aprendizaje automático

El aprendizaje automático generalmente se clasifica en tres tipos: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje de refuerzo

Aprendizaje supervisado:

En el aprendizaje supervisado, la máquina experimenta los ejemplos junto con las etiquetas u objetivos para cada ejemplo. Las etiquetas en los datos ayudan al algoritmo a correlacionar las características.

Dos de las tareas de aprendizaje automático supervisado más comunes son **la clasificación** y **la regresión** .

> En problemas **de clasificación** la máquina debe aprender a predecir valores discretos. Es decir, la máquina debe predecir la categoría, clase o etiqueta más probable para los nuevos ejemplos. Las aplicaciones de la clasificación incluyen predecir si el precio de una acción subirá o bajará, o decidir si un artículo periodístico pertenece a la sección de política o de ocio.

En problemas **de regresión** la máquina debe predecir el valor de una variable de respuesta continua. Los ejemplos de problemas de regresión incluyen predecir las ventas de un nuevo producto o el salario de un trabajo en función de su descripción.

Aprendizaje sin supervisión:

Cuando tenemos datos sin clasificar y sin etiquetar, el sistema intenta descubrir patrones a partir de los datos. No hay una etiqueta o un objetivo dado para los ejemplos. Una tarea común es agrupar ejemplos similares llamados agrupación.

Aprendizaje reforzado:

El aprendizaje por refuerzo se refiere a algoritmos orientados a objetivos, que aprenden cómo lograr un objetivo complejo (meta) o maximizar a lo largo de una dimensión particular en muchos pasos. Este método permite que las máquinas y los agentes de software determinen automáticamente el comportamiento ideal dentro de un contexto específico para maximizar su rendimiento. Se requiere una retroalimentación de recompensa simple para que el agente sepa qué acción es mejor; esto se conoce como la señal de refuerzo. Por ejemplo, maximizar los puntos ganados en un juego en muchos movimientos.

Técnicas de aprendizaje automático supervisado

La regresión es una técnica utilizada para predecir el valor de una variable de respuesta (dependiente), a partir de una o más variables predictoras (independientes).

Las técnicas de regresión más utilizadas son: **Regresión lineal** y **Regresión logística** . Discutiremos la teoría detrás de estas dos técnicas destacadas junto con la explicación de muchos otros conceptos clave
como algoritmos , técnicas `Gradient-descent` involucradas en el aprendizaje automático. `Over-fit/Under-fit Error analysis Regularization Hyper-parameters Cross-validation`

Regresión lineal

En los problemas de regresión lineal, el objetivo es predecir una variable de valor real y a partir de un patrón dado X. En el caso de la regresión lineal, la salida es una función lineal de la entrada. Sea $\hat{y}$ la salida que predice nuestro modelo: $\hat{y} = WX+b$

Aquí X hay un vector (características de un ejemplo), W son los pesos (vector de parámetros) que determinan cómo cada característica afecta la predicción y b es un término de sesgo. Entonces, nuestra tarea T es predecir y a partir de X, ahora necesitamos medir el rendimiento P para saber qué tan bien funciona el modelo.

Ahora, para calcular el rendimiento del modelo, primero calculamos el error de cada ejemplo como: i

$$e_i = abs(\hat{y}_i - y_i)$$

tomamos el valor absoluto del error para tener en cuenta los valores de error positivos y negativos.

Finalmente, calculamos la media de todos los errores absolutos registrados (suma promedio de todos los errores absolutos).

Error absoluto medio (MAE) = Promedio de todos los errores absolutos

$$MAE = 1/m \sum_i abs(\hat{y}_i - y_i)$$

La forma más popular de medir el rendimiento del modelo es usar

Error cuadrático medio (MSE) : Promedio de las diferencias cuadráticas entre la predicción y la observación real.

$$MSE = 1/2m \sum_i (\hat{y}_i - y_i)^2$$

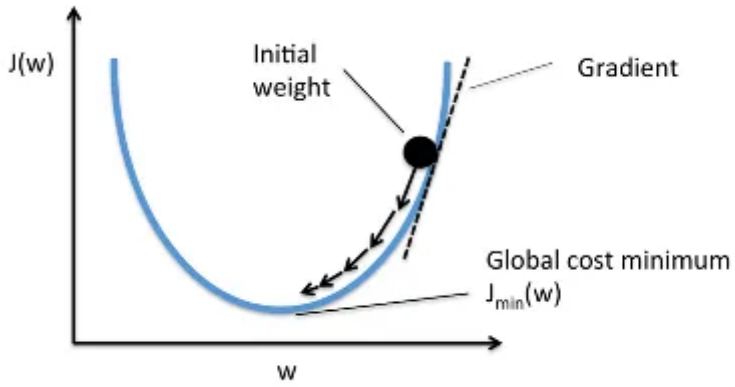
La media se reduce a la mitad (1/2) como una conveniencia para el cálculo del descenso del gradiente [discutido más adelante], ya que el término derivado de la función cuadrada cancelará el término 1/2. Para obtener más información sobre MAE vs MSE, consulte [1] y [2].

> El objetivo principal de entrenar el algoritmo ML es ajustar los pesos W para reducir el MAE o MSE.

Para minimizar el *error* , el modelo mientras experimenta los ejemplos del conjunto de entrenamiento, actualiza los parámetros del modelo `W`. Estos cálculos de error, cuando se grafican contra la función de costo, también se denominan **función de costo** , ya que determina el costo/penalización del modelo. Entonces, minimizar el error también se denomina minimización de la función de costo J. `W J(w)`

Algoritmo de descenso de gradiente:

Cuando graficamos la función de costo `J(w) vs w`. Se representa de la siguiente manera:



Como vemos en la curva, existe un valor de parámetros `W` que tiene el costo mínimo `Jmin`. Ahora necesitamos encontrar una manera de alcanzar este costo mínimo.

En el algoritmo de descenso de gradiente, comenzamos con parámetros de modelo aleatorios y calculamos el error para cada iteración de aprendizaje, seguimos actualizando los parámetros del modelo para acercarnos a los valores que resultan en un costo mínimo.

repetir hasta coste mínimo: {

$$w_j = w_j - \alpha \partial / \partial w_j J(W)$$

}

En la ecuación anterior, estamos actualizando los parámetros del modelo después de cada iteración. El segundo término de la ecuación calcula la pendiente o pendiente de la curva en cada iteración.

El gradiente de la función de costo se calcula como derivada parcial de la función de costo J con respecto a cada parámetro del modelo wj, j toma el valor del número de características $[1 \text{ to } n]$. α, *alpha* , es la tasa de aprendizaje, o qué tan rápido queremos avanzar hacia el mínimo. Si α es demasiado grande, podemos sobrepasarnos. Si α es demasiado pequeño, significa pequeños pasos de aprendizaje, por lo tanto, el tiempo total que tarda el modelo en observar todos los ejemplos será mayor.

Hay tres formas de hacer el descenso de gradiente:

Descenso de gradiente por lotes: utiliza todas las instancias de entrenamiento para actualizar los parámetros del modelo en cada iteración.

Descenso de gradiente de minilote: en lugar de usar todos los ejemplos, el descenso de gradiente de minilote divide el conjunto de entrenamiento en un tamaño más pequeño llamado lote indicado por 'b'. Por lo tanto, se utiliza un mini lote 'b' para actualizar los parámetros del modelo en cada iteración.

Descenso de gradiente estocástico (SGD): actualiza los parámetros utilizando solo una única instancia de entrenamiento en cada iteración. La instancia de entrenamiento generalmente se selecciona al azar. El descenso de gradiente estocástico a menudo se prefiere para optimizar las funciones de costos cuando hay cientos de miles de instancias de entrenamiento o más, ya que convergerá más rápidamente que el descenso de gradiente por lotes [3].

Regresión logística

En algunos problemas, la variable de respuesta no se distribuye normalmente. Por ejemplo, el lanzamiento de una moneda puede tener dos resultados: cara o cruz. La distribución de Bernoulli describe la distribución de probabilidad de una variable aleatoria que puede tomar el caso positivo con probabilidad P o el caso negativo con probabilidad $1-P$. Si la variable de respuesta representa una probabilidad, debe estar restringida al rango $\{0,1\}$.

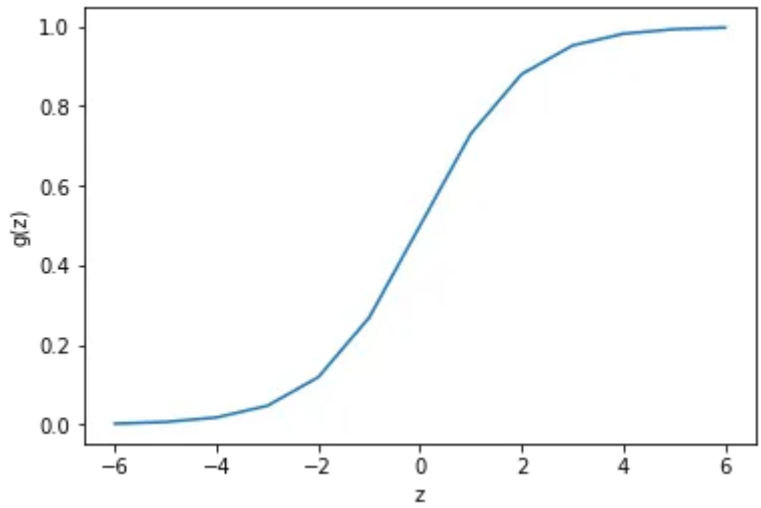
En la regresión logística, la variable de respuesta describe la probabilidad de que el resultado sea el caso positivo. Si la variable de respuesta es igual o supera un umbral de discriminación, se predice la clase positiva; de lo contrario, se predice la clase negativa.

La variable de respuesta se modela como una función de una combinación lineal de las variables de entrada utilizando la función logística.

Dado que nuestra hipótesis $\hat{y}$ tiene que satisfacer , esto se puede lograr conectando la función logística o "Función sigmoidea" $0 \leq \hat{y} \leq 1$

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

La función $g(z)$ asigna cualquier número real al $(0, 1)$ intervalo, lo que la hace útil para transformar una función de valor arbitrario en una función más adecuada para la clasificación. La siguiente es una gráfica del valor de la función sigmoidea para el rango $\{-6,6\}$:



Ahora, volviendo a nuestro problema de regresión logística, supongamos que z es una función lineal de una sola variable explicativa x. Entonces podemos expresar z de la siguiente manera:

$$z=w_0+w_1x$$

Y la función logística ahora se puede escribir como:

$$g(x)=\frac{1}{1+e^{-(w_0+w_1x)}}$$

Tenga en cuenta que `g(x)` se interpreta como la probabilidad de la variable dependiente.

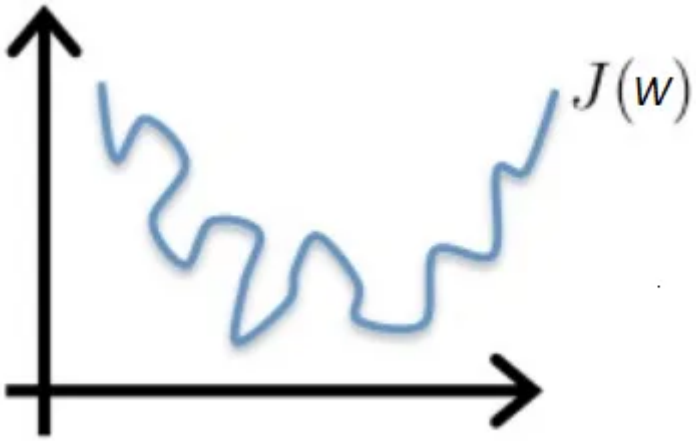
`g(x) = 0.7`, nos da una probabilidad del 70 % de que nuestra salida sea 1. Nuestra probabilidad de que nuestra predicción sea 0 es solo el complemento de nuestra probabilidad de que sea 1 (por ejemplo, si la probabilidad de que sea 1 es del 70 %, entonces la probabilidad de que sea 0 es 30%).

La entrada a la función sigmoidea `'g'` no necesita ser una función lineal. Puede muy bien ser un círculo o cualquier forma.

$$z=(w_0+w_1x_1^2+w_2x_2^2)$$

función de costo

No podemos usar la misma función de costo que usamos para la regresión lineal porque la función sigmoidea hará que la salida sea ondulada, causando muchos óptimos locales. En otras palabras, no será una función convexa.



Función de costo no convexa

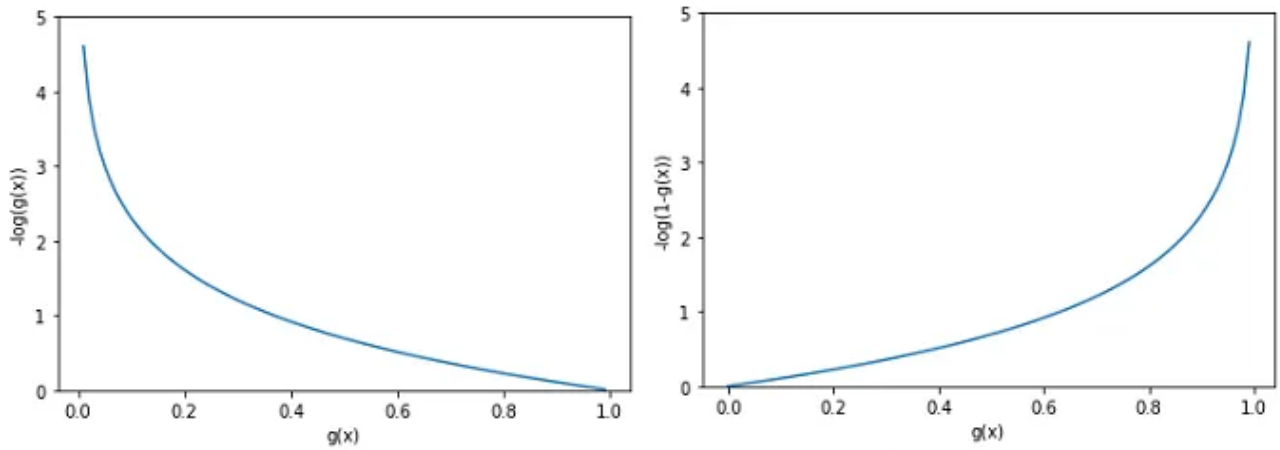
Para asegurar que la función de costo sea convexa (y por lo tanto asegurar la convergencia al mínimo global), la función de costo se transforma usando el logaritmo de la función sigmoidea. La función de costo para la regresión logística se ve así:

$$J(w)=\frac{1}{m}\sum_i Cost(g(x^{(i)}),y^{(i)})$$

$$Cost(g(x^{(i)}),y^{(i)}) = -log(g(x^{(i)})) \quad if \quad y=1, \qquad Cost(g(x^{(i)}),y^{(i)}) = -log(1-g(x^{(i)}) \quad if \quad y=0$$

Que se puede escribir como:

$$Cost(g(x^{(i)}),y^{(i)}) = -y^{(i)}log(g(x^{(i)})) - (1-y^{(i)})log(1-g(x^{(i)}))$$



cost for y = 1 cost for y =0

Entonces, la función de costo para la regresión logística es:

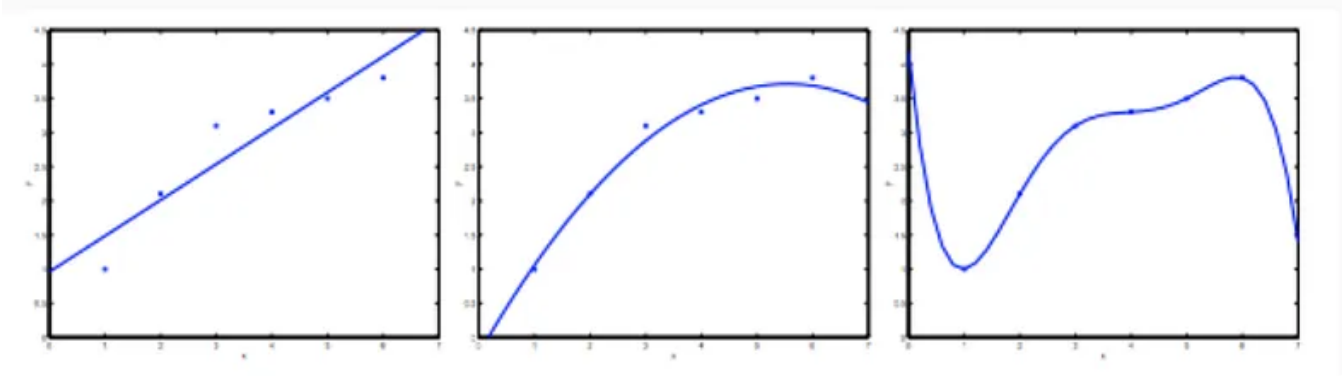
$$J(w) = -\frac{1}{m}\sum_{i} y^{(i)}log(g(x^{(i)})) + (1-y^{(i)})log(1-g(x^{(i)}))$$

Dado que la función de costo es una función convexa, podemos ejecutar el algoritmo de descenso de gradiente para encontrar el costo mínimo.

Ajuste insuficiente y ajuste excesivo

Intentamos que el algoritmo de aprendizaje automático se ajuste a los datos de entrada aumentando o disminuyendo la capacidad de los modelos. En problemas de regresión lineal, aumentamos o disminuimos el grado de los polinomios.

Considere el problema de predecir y a partir de $x \in R$. La siguiente figura a la izquierda muestra el resultado de ajustar una línea a un conjunto de datos. Dado que los datos no se encuentran en línea recta, el ajuste no es muy bueno (figura del lado izquierdo).



Para aumentar la capacidad del modelo, agregamos otra característica al agregarle un término x^2 . Esto produce un mejor ajuste (figura central). Pero si seguimos haciéndolo (x^5 , polinomio de quinto orden, figura del lado derecho), es posible que podamos ajustar mejor los datos, pero no generalizaremos bien para nuevos datos. La primera cifra representa un ajuste insuficiente y la última cifra representa un ajuste excesivo.

Debajo del ajuste:

Cuando el modelo tiene menos funciones y, por lo tanto, no puede aprender muy bien de los datos. Este modelo tiene un alto sesgo.

Sobreajuste:

Cuando el modelo tiene funciones complejas y, por lo tanto, puede ajustarse muy bien a los datos, pero no puede generalizar para predecir nuevos datos. Este modelo tiene una alta varianza.

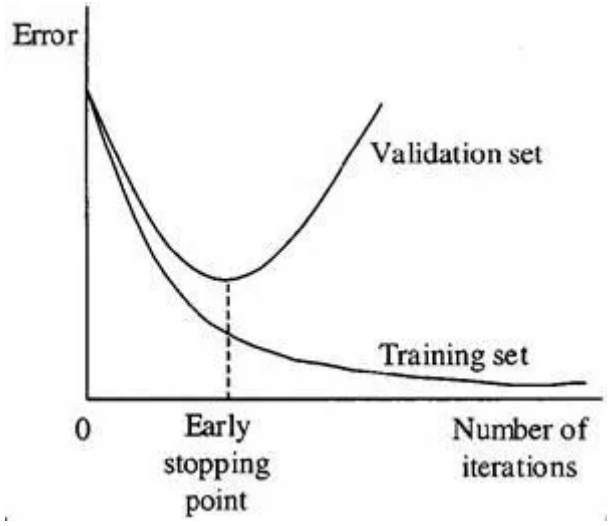
Hay tres opciones principales para abordar el problema del ajuste excesivo:

1. **Reduzca el número de funciones:** seleccione manualmente qué funciones conservar. Al hacerlo, es posible que nos perdamos información importante, si descartamos algunas funciones.

2. **Regularización:** mantenga todas las funciones, pero reduzca la magnitud de los pesos W. La regularización funciona bien cuando tenemos muchas funciones ligeramente útiles.

3. **Detención anticipada:** cuando estamos entrenando un algoritmo de aprendizaje de forma iterativa, como el uso del descenso de gradiente, podemos medir qué tan bien se desempeña cada iteración del modelo. Hasta cierto número de iteraciones, cada iteración mejora el modelo. Sin embargo, después de ese punto, la capacidad del modelo para generalizar puede

debilitarse a medida que comienza a sobreajustarse a los datos de entrenamiento.



regularización

La regularización se puede aplicar tanto a la regresión lineal como a la logística agregando un término de penalización a la función de error para evitar que los coeficientes o pesos alcancen valores grandes.

Regresión lineal con regularización

El término de penalización más simple toma la forma de una suma de cuadrados de todos los coeficientes, lo que lleva a una función de error de regresión lineal modificada:

$$J(w)=\frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)}-y^{(i)})^2 \;+\; \frac{\lambda}{2n}\sum_{j=1}^{n}w_j^2$$

donde lambda es nuestro parámetro de regularización.

Ahora, para minimizar el error, usamos el algoritmo de descenso de gradiente. Seguimos actualizando los parámetros del modelo para acercarnos a los valores que resultan en un costo mínimo.

repetir hasta la convergencia (con regularización): {

$$w_0 := w_0 \;-\; \alpha\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)}-y^{(i)})\;x_0^{(i)}$$

$$w_j := w_j - \alpha[(\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)}-y^{(i)})x_j^{(i)}) + \frac{\lambda}{m}w_j], \ \epsilon \ \{1,2..n\}$$

}

Con alguna manipulación, la ecuación anterior también se puede representar como:

$$w_j := w_j(1-\alpha)\frac{\lambda}{m} - \alpha\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)}-y^{(i)})x_j^{(i)}, \ \epsilon \ \{1,2..n\}$$

El primer término de la ecuación anterior,

$$1-\alpha\frac{\lambda}{m}$$

siempre será menor que 1. Intuitivamente, puede verlo como una reducción del valor del coeficiente en cierta cantidad en cada actualización.

Regresión Logística con Regularización

La función de costo de la regresión logística con Regularización es:

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}y^{(i)}log(g(x^{(i)})) + (1-y^{(i)})log(1-g(x^{(i)})) + \frac{\lambda}{2n}\sum_{j=1}^{n}w_j^{\,2}$$

repetir hasta la convergencia (con regularización): {

$$w_0 := w_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(g(x^{(i)})-y^{(i)})x_0^{(i)}$$

$$w_j := w_j - \alpha[(\frac{1}{m}\sum_{i=1}^{m}(g(x^{(i)})-y^{(i)})x_j^{(i)}) + \frac{\lambda}{m}w_j], \quad j \ \epsilon \ \{1,2..n\}$$

}

Regularización L1 y L2

El término de regularización utilizado en las ecuaciones anteriores se llama regularización L2 o Ridge.

$$\frac{\lambda}{2n}\sum_{j=1}^{n} w_j{}^2$$

La penalización L2 tiene como objetivo minimizar la magnitud al cuadrado de los pesos.

Existe otra regularización llamada L1 o Lasso:

$$\frac{\lambda}{2n}\sum_{j=1}^{n} |w_j|$$

La penalización L1 tiene como objetivo minimizar el valor absoluto de los pesos

Diferencia entre L1 y L2

L2 reduce todos los coeficientes en las mismas proporciones pero no elimina ninguno, mientras que L1 puede reducir algunos coeficientes a cero, realizando así la selección de características. Para más detalles lea esto .

hiperparámetros

Los hiperparámetros son parámetros de "nivel superior" que describen información estructural sobre un modelo que debe decidirse antes de ajustar los parámetros del modelo, ejemplos de hiperparámetros que discutimos hasta ahora: tasa de aprendizaje *alfa* , regularización *lambda*.

Validación cruzada

El proceso para seleccionar los valores óptimos de los hiperparámetros se denomina selección de modelo. si reutilizamos el mismo conjunto de datos de prueba una y otra vez durante la selección del modelo, se convertirá en parte de nuestros datos de entrenamiento y, por lo tanto, es más probable que el modelo se sobreajuste.

El conjunto de datos general se divide en:

1. el conjunto de datos de entrenamiento
2. conjunto de datos de validación
3. conjunto de datos de prueba.

El conjunto de entrenamiento se usa para ajustar los diferentes modelos, y el rendimiento en el conjunto de validación se usa luego para la selección del

modelo. La ventaja de mantener un conjunto de prueba que el modelo no haya visto antes durante los pasos de entrenamiento y selección del modelo es que evitamos sobreajustar el modelo y el modelo puede generalizar mejor a datos no vistos.

En muchas aplicaciones, sin embargo, el suministro de datos para el entrenamiento y las pruebas será limitado y, para construir buenos modelos, deseamos utilizar la mayor cantidad posible de datos disponibles para el entrenamiento. Sin embargo, si el conjunto de validación es pequeño, dará una estimación relativamente ruidosa del rendimiento predictivo. Una solución a este dilema es utilizar la validación cruzada, que se ilustra en la Figura siguiente.

A continuación, los pasos de validación cruzada se toman desde aquí y se agregan aquí para completar.

Validación cruzada paso a paso:

Estos son los pasos para seleccionar hiperparámetros utilizando la validación cruzada K-fold:

1. Divide tus datos de entrenamiento en K = 4 partes iguales, o "pliegues".
2. Elija un conjunto de hiperparámetros que desee optimizar.
3. Entrena tu modelo con ese conjunto de hiperparámetros en los primeros 3 pliegues.
4. Evalúelo en el cuarto pliegue, o en el pliegue de "retención".
5. Repita los pasos (3) y (4) K (4) veces con el mismo conjunto de hiperparámetros, cada vez sosteniendo un pliegue diferente.

6. Agregue el rendimiento en los 4 pliegues. Esta es su métrica de rendimiento para el conjunto de hiperparámetros.

7. Repita los pasos (2) a (6) para todos los conjuntos de hiperparámetros que desee considerar.

La validación cruzada nos permite ajustar los hiperparámetros solo con nuestro conjunto de entrenamiento. Esto nos permite mantener el conjunto de prueba como un conjunto de datos verdaderamente invisible para seleccionar el modelo final.

Conclusión

Hemos cubierto algunos de los conceptos clave en el campo del aprendizaje automático, comenzando con la definición de aprendizaje automático y luego cubriendo diferentes tipos de técnicas de aprendizaje automático. Discutimos la teoría detrás de las técnicas de regresión más comunes (lineal y logística) junto con otros conceptos clave del aprendizaje automático.

Gracias por leer.

Referencias

[1] https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[2] https://towardsdatascience.com/ml-notes-why-the-least-square-error-bf27fdd9a721

[3] https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

[4] https://elitedatascience.com/machine-learning-iteration#micro

Apr