Regresión y clasificación: especialización ML

miércoles, 16 de agosto de 2023 21:56

'x': Feature variable, variable de entrada. $\hat{y}$: predicción del modelo.

'y': target variable, variable de salida

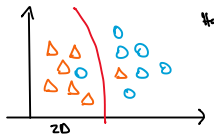
'm': número de ejemplos de entrada.

(x,y): Un único elemento del conjunto de entrenamiento.

$(x^{(m)}, y^{(m)})$: mismo ejemplo de entrenamiento.

Aprendizaje supervisado:

clase = categoría > salida

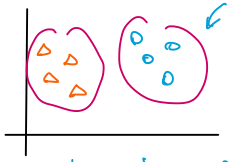


Hallar un límite que separa ambas clases

El límite me dice si es + o - probable que pertenezca a x o y clase.

Aprendizaje no supervisado

Datos no etiquetados, el algoritmo debe de hallar un patrón en los datos.



Clustering

Agrupa noticias parecidas

Agrupa sectores de clientes para mejorar atención (Market segmentation).

→ Detección de anomalías
→ Reducción de la dimensionalidad.
{ Comprimir un conjunto de datos en otro más pequeño perdiendo la menor información posible }

> Predecir un número {22, 1.7, 1.8} se denomina un problema de regresión.

- Número infinito de posibles salidas.

> Predecir una categoría o categoría discreta es un problema de clasificación

- Número reducido, discreto y finito de posibles salidas.

Funcionamiento

Regresion lineal (supervisado).

$f(x) = wx+b$ → bias (weight)

Regresion lineal univariante → solo 1 variable de entrada (x).

w: pendiente de la funcion
b: intercepto con el eje y

Se ajusta para que la linea pase aproximadamente o este cerca de todos los puntos.

$\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$ error → que tan lejos esta la prediccion del objetivo, del valor real.

(SE error cuadratico)

$\frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$ } MSE (error cuadratico medio)) Funcion de coste.

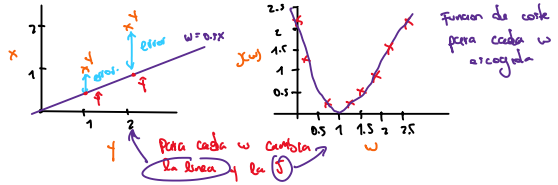
↳ Muy usada en todos los problemas de regresion

$\hat{y} = f_{(w,b)}(x^{(i)}) = wx^{(i)}+b$ > una prediccion es

- Hallar valores de w y b que hagan mas pequeña la funcion de coste.

Objetivo: minimize $J(w,b)$ (Funcion d'coste)

- Funcion de coste: nos dice que tambien funciona el modelo.
- Parametros del modelo son las variables que se pueden ajustar durante el entrenamiento.



Como depende de w y b la funcion de coste se ve de mejor manera en un grafico 3D, sin embargo, tambien se hace uso de un grafico de contorno para visualizar la funcion de coste.

Descenso de gradiente es un algoritmo que se usa para minimizar cualquier funcion de entrada.

- Comienza con una hipotesis inicial colocando los parametros como 0.
- En cada iteracion cambiara los parametros tratando de reducir la funcion de coste.
- Si quiero bajar ¿Que direccion tomo para poder bajar mas rapido? (Direccion del descenso mas pronunciado).
- En cada iteracion el algoritmo tratara de hallar la direccion del descenso mas pronunciado.
- Repetir hasta que la funcion converja, hallar un minimo local, punto en donde por cada paso los valores de los parametros no cambian mucho

→ En cada paso

$w := w - \alpha \frac{\partial}{\partial w} J(w,b)$ asignacion

$b := b - \alpha \frac{\partial}{\partial b} J(w,b)$ asignacion

w_{i-1}: Valor anterior de w (w_{i-1})

b_{i-1}: Valor anterior de b (b_{i-1})

α: tasa de aprendizaje

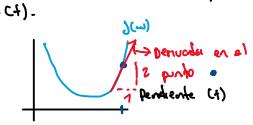
numero positivo entre 0 y 1, controla que tan grande es la distancia para volver a calcular el descenso del gradiente

$\frac{\partial}{\partial w} J(w,b)$: Derivada de la funcion de coste $J(w,b)$.

nos dice en que direccion debemos dar w, me ayuda a calcular la pendiente de la funcion, la direccion de la funcion

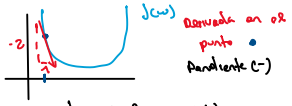
- Se deberan de actualizar los parametros

- Cuando la recta tangente (la derivada) apunta hacia arriba a la derecha la pendiente es positiva (+).



Como w esta positivo se le resta (-)

- Cuando la recta tangente (la derivada) apunta hacia abajo a la derecha (arriba-izquierda) la pendiente es negativa (-).



Como w es negativa se le suma (+)

- Al elegir un α pequeño se decrementa el costo de forma muy lenta.

- Si α es muy grande se puede sobrepasar el mínimo local de la función, puede llegar a no converger o incluso divergir.

- Si ya se esta en un mínimo local la pendiente es 0 por lo que, 'w' quedara sin cambios por hacer.

- A medida que se acerca al mínimo local la pendiente (derivada) sera cada vez menor, y por ende, el paso también será menor.

- Batch gradient descent: En cada paso del descenso de gradiente usamos todos los datos de entrenamiento.

$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$

$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$

$w = tmp\_w$

$b = tmp\_b$

Multiples variables

x_j = j^{th} feature

n = # de features

$\vec{x}^{(i)}$ = features del i^{th} ejemplo de entrenamiento.

$x_j^{(i)}$ = valor del feature j en el i^{th} ejemplo de entrenamiento. (i → fila, j → columna)

Regresion logistica

$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n + b$

$f_{w,b}(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$

$\vec{w} = [w_1\ w_2\ w_3\ \dots\ w_n]$ (vector)

$\vec{x} = [x_1\ x_2\ x_3\ \dots\ x_n]$ (vector) features.

$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ (· → producto punto)

Vectorizacion: convertir los parametros a vectores (Numpy)

✓ Hace el codigo mas corto

✓ Hace que el codigo sea mas eficiente (realiza operaciones en paralelo).

→ coge todos los valores de los vectores y realiza los calculos correspondientes en paralelo.

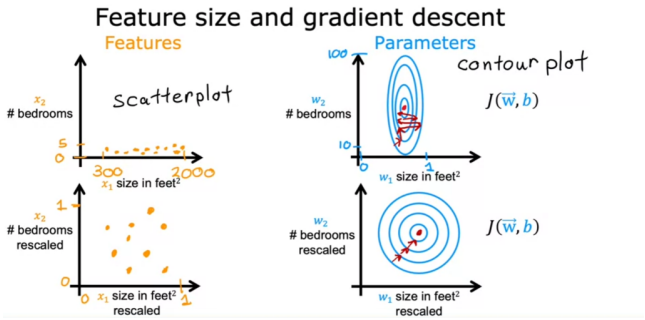
! en el descenso de gradiente con n parametros
el tiene que actualizar los n valores del vector $\vec{w}$ y luego la "b" n veces para minimizar la funcion de coste.

▲ Hay otro algoritmo que sirve para hallar w y b solo en regresion lineal y sus iteraciones: ecuacion normal.

Desventajas:
✓ lento cuando el # de parametros es alto (>10,000)

Algunas bibliotecas de ML pueden usarlo para resolver w y b 'under the hood'

Feature scaling: tecnica.
Al tener datos de distintas magnitudes el contorno de las funciones asociadas a los mismos puede ser muy grande dada la varianza en su magnitud, en este caso es mejor realizar un escalado de los datos, para que todos tengan la misma base y magnitud.
El escalado de parametros permite que el modelo converja rapidamente.



Escalamiento: Forma 1
coges el maximo de c/u de las caracteristicas y dividir c/u de los valores de la caracteristica con su respectivo maximo.

Escalamiento: Forma 2 (Mean normalization).
se reescalan las caracteristicas para q' sus valores queden centrados alrededor del 0, sus valores pueden ser + o −, suele estar entre +1 y −1.

✓ Halla la media (μ → mu)

✓ Halla el maximo y minimo del intervalo.

✓ $x_1 = \frac{x_1 - \mu(\text{media})}{\max - \min}$; para c/u de los datos del feature

Escalamiento: Forma 3 (Z score)

✓ Calcular la desviacion estandard (Std), y la media (μ) de la columna.

✓ $x_1 = \frac{x_1 - \mu}{\text{Std}}$; para c/u de los datos del feature.

! se debe de tratar de q luego de escalar el rango del feature oscile lo mas cerca posible de −1 a 1.

cuidado de rangos muy pequeños: ($-0.001 \leq x \leq 0.001$)

y de valores muy grandes (-∞ ≤ x ≤ ∞)

Para saber si el descenso del gradiente funciona se puede graficar los valores de J (función de coste) para cada de las iteraciones, esta curva se llama 'learning curve'

J debería de decrecer en cada de las iteraciones
si J crece quiere decir que la tasa de aprendizaje es muy alta

Cuando la curva deja de crecer el descenso del gradiente ha convergido.

Automatic convergence test:
Es como un stopper, si J(w,b) ≤ ε (stopper)
declarar convergencia.

Como elegir la tasa de aprendizaje (α)
✓ El coste debería de bajar en cada de las iteraciones.

✓ Bajo alpha (α) a un número muy pequeño y ver si es el coste baja.
↳ Depuración.

✓ Verificar con distintos valores de alpha y trazar la función de coste.
α = [0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3...]

Feature engineering: Transformar o combinar las características (datos) originales para facilitar que el algoritmo prediga con más precisión

Polynomial Regression: Ajusta curvas del orden x^2, x^3, x^n
puede comenzar con una curva cuadrática y continuar con una función cubica ($w_1x + w_2x^2 + w_3x^3 + b$)

! El escalado de parametros es de suma importancia en estos modelos de x^2, x^3 para no introducir sesgo en el modelo y que los parametros tengan rangos comparables.

Clasificación binaria: Problemas en los cuales solo hay 2 posibles clases - categorías de salida.

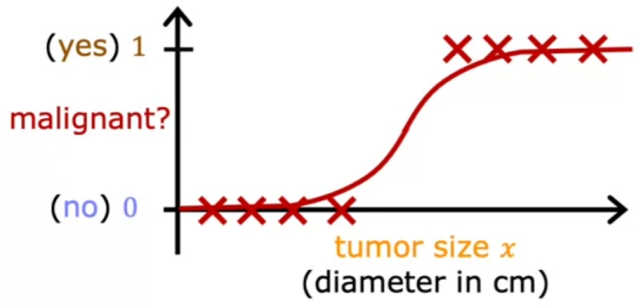
Falso | Verdadero ← clase positiva = presencia

Clase negativa = ausencia

! Regresión lineal no es binaria en clasificación ya que calcula la linea q' mejor se ajusta a los datos recalculando la frontera de decisión.

Regresión logística (Clasificación binaria)

✓ Se ajusta una curva en forma de 'S' al conjunto de datos.



✓ El algoritmo devuelve una probabilidad asociada al dato dado (0-1)

! la probabilidad de que la clase sea igual a 1 dado un determinado $\vec{x}$.

Funcion sigmoide o logistica:

$g(z) = \frac{1}{1+e^{-z}} \quad \hat{=} \quad y = \frac{1}{1+e^{-x}}$

Matematica de la reg. logistica

$z = \vec{w} \cdot \vec{x} + b$ y Reg. lineal

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

$f_{\vec{w},b}(\vec{x}) = P(y=1 \mid \vec{x}; \vec{w}, b)$

Probabilidad que "y" sea 1, dado un input $\vec{x}$, y parametros $\vec{w}$ y b.

$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z})$

$= \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$

Limite de decision: Neutral al predecir si y es 0 o 1.

$z = \vec{w} \cdot \vec{x} + b = 0$, este limite puede ser lineal

o polinomico.

! El limite de decision puede cambiar, puede ser 0.5 asi como puede ser 0.9; depende de que tan sensible se quiera al modelo.

Funcion de costo:

Al aplicar MSE (error cuadratico medio) a la funcion de la regresion logistica me da una funcion no convexa, con diversos minimos locales, causando en ocasiones dar descenso de gradiente.

$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(f_{\vec{w},b}(\vec{x}^{(i)})) & \text{si } y^{(i)} = 1 \\ -\log(1 - f_{\vec{w},b}(\vec{x}^{(i)})) & \text{si } y^{(i)} = 0 \end{cases}$

Funcion de perdida

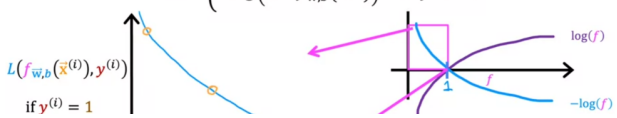
funcion de perdida: ver que tan bien o mal el algoritmo actua para un dato en particular.

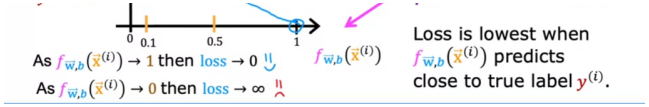
funcion de costo: Que tan bien el algoritmo lo hizo para un conjunto de datos (Promedio de todas las funciones de perdida)

! la perdida es mas baja cuando $f_{\vec{w},b}(\vec{x}^{(i)})$ predice cerca del valor real $y^{(i)}$

Logistic loss function

$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$

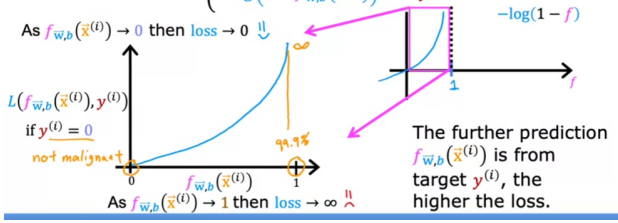




Loss is lowest when $f_{\vec{w},b}(\vec{x}^{(i)})$ predicts close to true label $y^{(i)}$.

Logistic loss function

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$



The further prediction $f_{\vec{w},b}(\vec{x}^{(i)})$ is from target $y^{(i)}$, the higher the loss.

! Costo — pérdida

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)$$

$$\begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{si } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{si } y^{(i)} = 0 \end{cases}$$

! Este costo genera una función convexa, por lo que el descenso de gradiente puede alcanzar un mínimo global.

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)$$

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)\right]$$

Gradient descent

cost

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

repeat {

$j = 1 \dots n$

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w},b) \qquad \frac{\partial}{\partial w_j}J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$b = b - \alpha\frac{\partial}{\partial b}J(\vec{w},b) \qquad \frac{\partial}{\partial b}J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

} simultaneous updates

Underfitting: Es cuando el modelo no se ajusta bien a los datos de entrenamiento.

→ El algoritmo tiene alto sesgo (bias).

Bias
- El algoritmo está sesgado al reconocer a una población, etnia, etc.
- El algoritmo no se adapta bien al conjunto de datos.

Generalización: Que el modelo se vaya bien predicciendo

datos no vistos anteriormente.

Overfitting: Se ajusta demasiado bien al conjunto de datos y no generaliza haciendo que sea malo al predecir nuevos datos; el algoritmo tiene una alta varianza.

Abordando el overfitting:

1. Recopilar mas datos de entrenamiento.
2. Ver si se pueden incluir/excluir caracteristicas. (feature selection).

¡Muchos features + datos insuficientes = Overfit.!

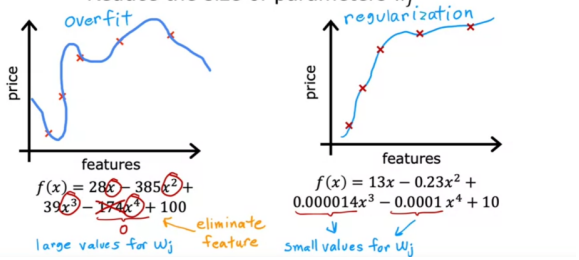
" El algoritmo descarta parte de la información sobre las cosas.

3. Regularización: Reduce el impacto de las features sin llegar al punto de eliminarlas.

✓ Fomentar que el algoritmo baje los valores de los parametros (w_j) sin exigir que se eliminen (se pongan como 0)

Regularization

Reduce the size of parameters w_j



Solo se regularizan w_1 - w_n y no b, no hay practicamente diferencia si se regularizar b o no.

Regularización:

✓ Minimizar w_1 - w_n al hacerlas ≈ 0.

✓ Entre mas bajos los valores de w mas simple el modelo, incluso algunos w pueden ser 0 eliminando una caracteristica.

✓ Entre mas simple el modelo, menos probable que se de sobreajuste.

✓ Se añade a la funcion de costo $J(\vec{w},b)$ todos los valores de $\vec{w}$ para que se aproximen a 0.

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

λ: "lambda" es el parametro de regularizacion
Representa que tan importante o no es la regularizacion con respecto a minimizar el error, es por esto que balancea ambas metas.

¿si es 0 no se esta regularizando

Si α (tasa de aprendizaje muy grande) se pone mucho antes en regularizar, por lo que $\vec{w} \approx 0$ y $f(x) \approx b$ resultando en una línea horizontal.

Regularized linear regression

$$J(\vec{w},b) = \min_{w,b}\left[\frac{1}{2m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

$$\frac{\partial}{\partial w_j} J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m} w_j$$

$$\frac{\partial}{\partial b} J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$ No se regulariza b

En cada iteración se multiplica w_j por un número menor a 1 (por muy poco) haciendo que los valores de $\vec{w}$ bajen en c/u de las iteraciones.

$$w_j\left(1 - \alpha\frac{\lambda}{m}\right)$$ — parám. regularización

tasa aprendizaje — tamaño del conjunto de entrenamiento

Regularized logistic regression:

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(f_{\vec{w},b}(\vec{x}^{(i)})) + (1-y^{(i)})\log(1-f_{\vec{w},b}(\vec{x}^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$\min \vec{w},b \rightarrow w_j, b$