Tareas y Notas

Andrés Moguel López Jensen
Machine Learning
UIA
Otoño 2022

Índice

1. Tarea 1: Machine Learning

1. Aprendizaje no supervisado:
   El aprendizaje no supervisado (y la inteligencia artificial) se pueden utilizar en la detección de anomalías. Uno de los algoritmos más comunes para este uso es el *Density Based Scan Clustering* (DBSC). Este algoritmo ayuda a detectar ruido en los datos. De la manera más simple, el algoritmo crea agrupaciones o vecindarios de datos en un radio definido; los datos que queden fuera o no tengan grupo serán identificados como ruido y posibles anomalías.[1]

2. Aprendizaje supervisado:
   Encontré una aplicación de aprendizaje automático para poder descifrar señales utilizadas en partidos de béisbol. Esto es de un video de Mark Rober. La aplicación permite al usuario asignar letras a los distintos tipos de señales que le da el çoach"de tercera base a un corredor. Tras ingresar y asignar todas las señales, se procede a una fase de aprendizaje en donde el usuario ingresa el patrón de señales utilizado y si hubo o no intento de robo de base. Tras varias repeticiones de esto la aplicación luego comienza a predecir si habrá robo de base o no basado en la secuencia ingresada. [2]

3. Algoritmo Cocktail Party: Consiste en poder enfocarse en una sola voz o sonido a pesar de que exista ruido u otras voces alrededor o en el ambiente en el que se está presente. En el campo de la música se ha logrado separar voces de los ritmos y música de fondo utilizando machine learning y redes neuronales. Esto ha sido desarrollado en la Universidad de Surrey en el Reino Unido [3]

Referencias

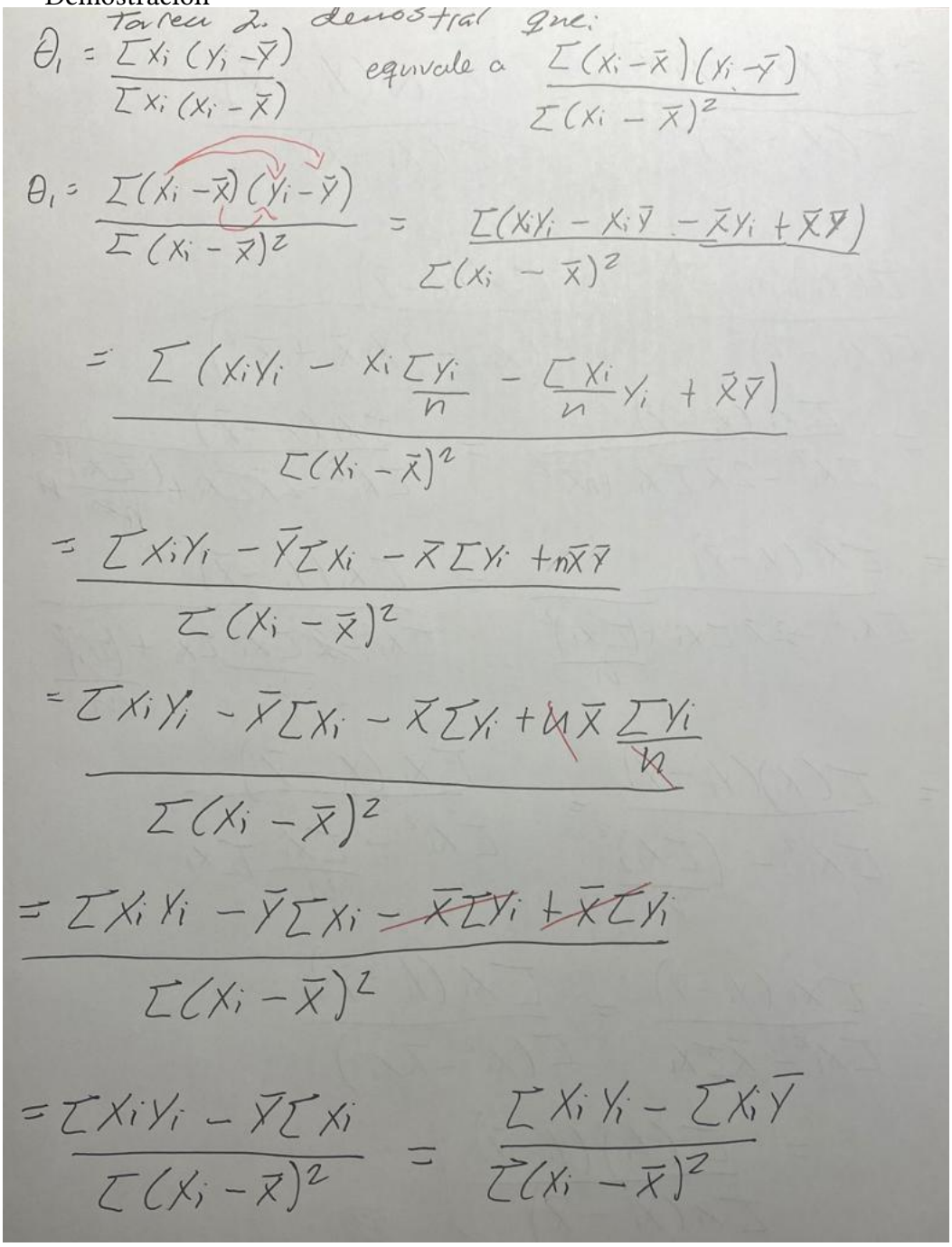
1. Cepeda, A., 2022. Aprendizaje no supervisado para la detección de anomalías. Centum Digital. Available at: https://centum.com/aprendizaje-no-supervisado-para-la-deteccion-de-anomalias/ [Accessed August 21, 2022].

2. Rober, M., 2019. Stealing baseball signs with a phone (machine learning). YouTube. Available at: https://www.youtube.com/watch?v=PmlRbfSavbI [Accessed August 21, 2022].

3. arXiv, E.T.from the, 2020. Deep Learning machine solves the cocktail party problem. MIT Technology Review. Available at: https://www.technologyreview.com/2015/04/29/168316/deep-learning-machine-solves-the-cocktail-party-problem/ [Accessed August 21, 2022].

2. Tarea 2: Regresión Lineal

Demostración



$$= \frac{\sum(x_i y_i - x_i \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{\sum x_i (y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Ya tenemos el numerador

Ahora el denominador

$$\frac{\sum(x_i)(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{\sum x_i (y_i - \bar{y})}{\sum(x_i^2 - 2x_i\bar{x} + \bar{x}^2)}$$

$$= \frac{\sum x_i (y_i - \bar{y})}{\sum x_i^2 - 2\bar{x}\sum x_i + n\bar{x}^2} = \frac{\sum x_i (y_i - \bar{y})}{\sum x_i^2 - 2\bar{x}\sum x_i + \frac{(\sum x_i)^2}{n^2}\, n}$$

$$= \frac{\sum x_i (y_i - \bar{y})}{\sum x_i^2 - 2\bar{x}\sum x_i + \frac{(\sum x_i)^2}{n}} = \frac{\sum x_i (y_i - \bar{y})}{\sum x_i^2 - \frac{2\sum x_i}{n}\sum x_i + \frac{(\sum x_i)^2}{n}}$$

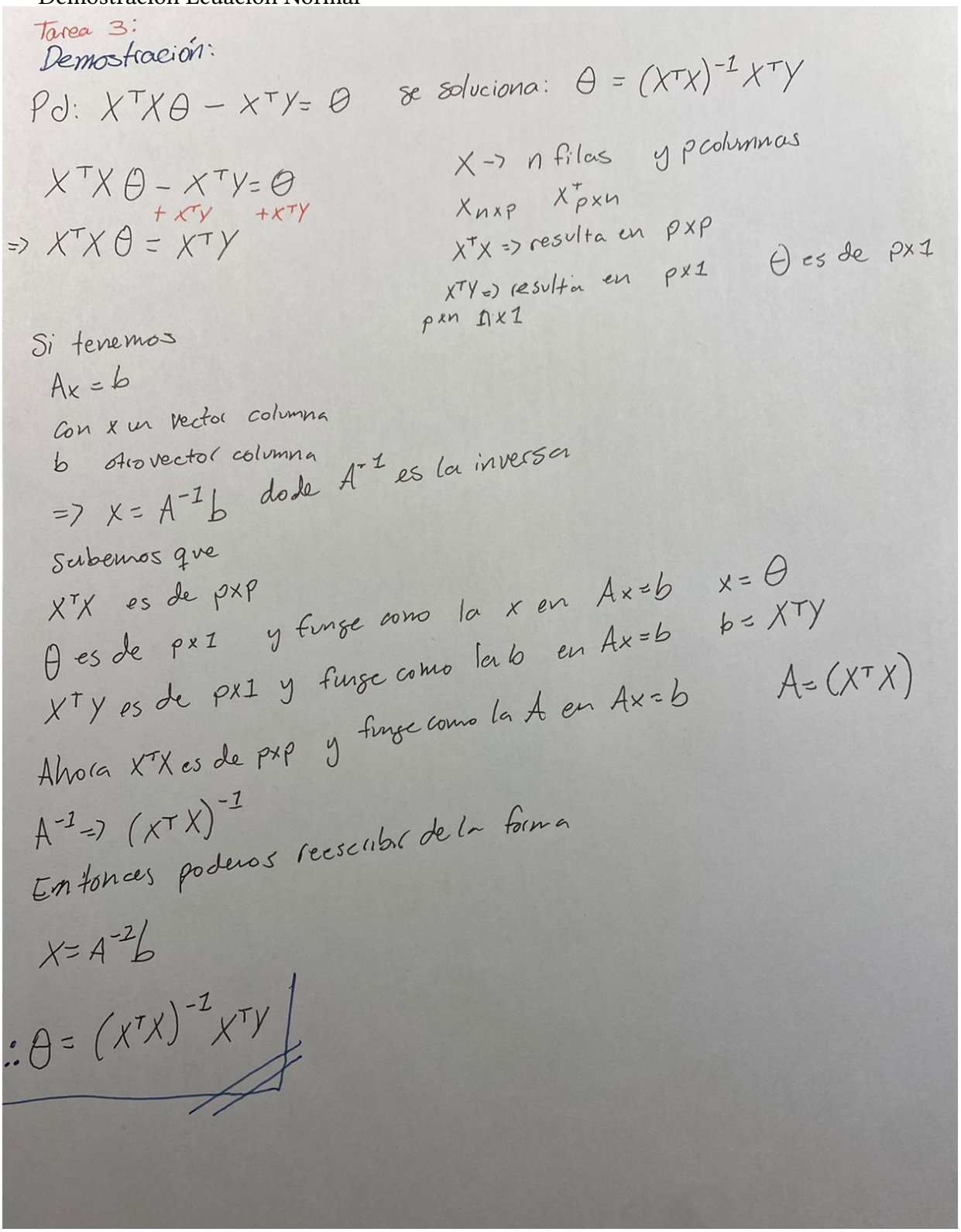
$$= \frac{\sum(x_i)(y_i - \bar{y})}{\sum x_i^2 - \frac{(\sum x_i)^2}{n}} = \frac{\sum x_i (y_i - \bar{y})}{\sum x_i^2 - \frac{\sum x_i}{n}\sum x_i}$$

$$= \frac{\sum x_i (y_i - \bar{y})}{\sum x_i^2 - \bar{x}\sum x_i} = \frac{\sum x_i (y_i - \bar{y})}{\sum(x_i^2 - \bar{x}x_i)}$$

$$= \frac{\sum(x_i)(y_i - \bar{y})}{\sum x_i(x_i - \bar{x})}$$

3. Tarea 3: Regresión Lineal Multivariable

Demostración Ecuación Normal



¿Qué pasa cuando la ecuación normal $(X^T X)$ no es invertible?
Cuando la ecuación normal no es invertible, se debe utilizar la inversa de Moore-Penrose.
Sea una matriz B de mxn la matriz inversa generalizada de Moore-Penrose es una matriz pseudo-inversa única de nxm denotada como B^+.
La inversa de Moore-Penrose satisface o cumple lo siguiente:

1. $BB^+B = B$
2. $B^+BB^+ = B^+$
3. $(BB^+)^H = BB^+$
4. $(B^+B)^H = B^+B$

Dónde B^H es la matriz conjugada traspuesta [1].

Feature Scaling en Gradiente Descendiente
Feature Scaling es una idea o concepto que se utiliza en el algoritmo del gradiente descendiente para asegurar o mantener a todos los parámetros estén todos en una misma escala.
Se utiliza el Feature Scaling para acelerar el proceso de convergencia del gradiente descendiente.

Digamos que x1 y x2 son parámetros en un problema a resolver por gradiente descendiente. Si a ambos parámetros los dividimos o .esçalamos.º normalizamos de cierta manera los datos por sus respectivas unidades, esto hará que la convergencia a un mínimo de la función de costo sea mucho más rápida. [2]

Mean Normalization es una alternativa para Feature Scaling. En este método se escala o se aumenta el tamaño de la media, dividiendo por el rango de valores del tamaño de los datos. [2]

Feature scaling es el proceso de traer todos los elementos de un problema de machine learning a una escala o rango similar. Se utiliza para normalizar el rango de variables independientes. Puede tener un efecto significativo para mejorar la eficiencia en el entrenamiento de un modelo de Machine Learning. [3]

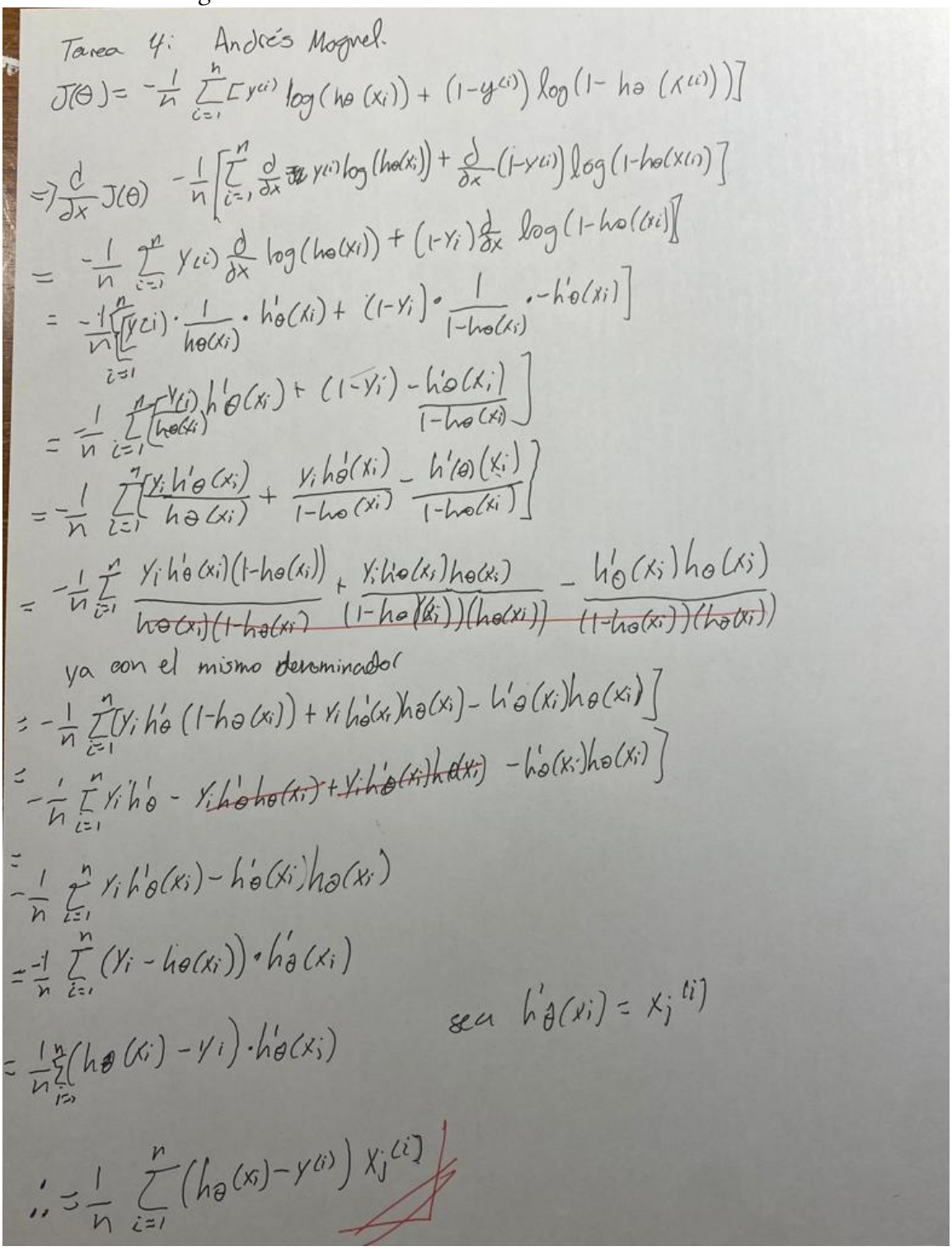
El mean normalization se ve de la siguiente manera:
$x^(mod) = (x - media(x))/(rango(x))$ [3].

Referencias

1. https://mathworld.wolfram.com/Moore-PenroseMatrixInverse.html
2. https://societyofai.medium.com/
3. https://medium.com/analytics-vidhya/mean-normalization-and-feature-scaling-a-simple-explanation-3b9be7bfd3e8

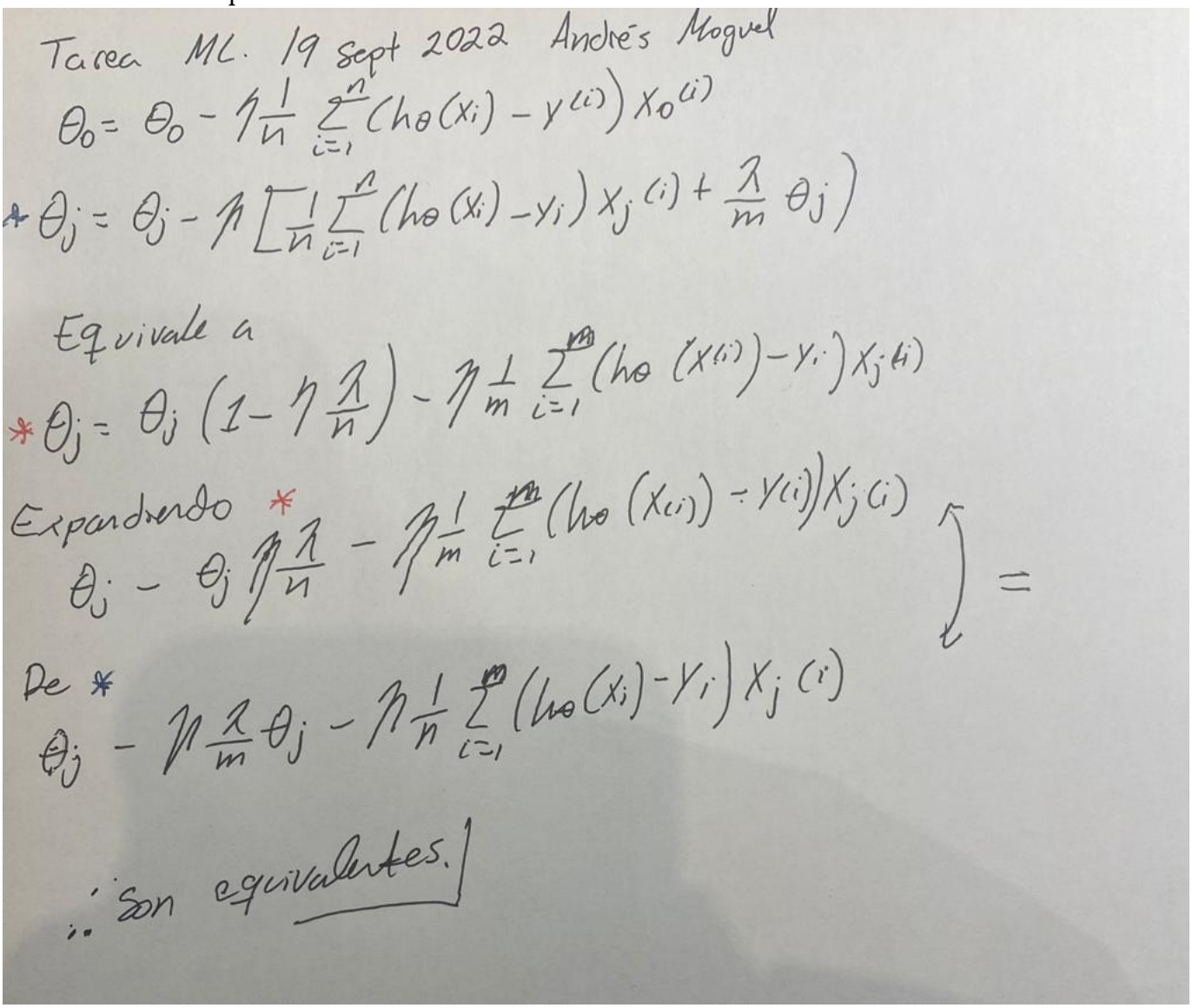
4. Tarea 4: Regresión Logística

Demostración gradiente descendiente:



5. Tarea 5: Regularización

Demostración equivalencia de ecuaciones:



¿Cómo se vería la formula normal con la regularización?

Determinar la forma final matricial de la respuesta al problema de regresión lineal multivariable.

Se tiene lo siguiente tomando la forma matricial sin regularización:

$$\overline{\theta} = (X^T X)^{-1} X^T \overline{y}$$

Al añadir el término de regularización con λ tenemos:

$$J(\overline{\theta}) = \frac{1}{n} \sum_{t=1}^{n} \frac{1}{2} \left[y^{(t)} - \overline{\theta} \cdot \overline{x}^{(t)} \right]^2 + \frac{\lambda}{2} \|\overline{\theta}\|^2$$

Derivando e igualando a cero se tiene:

$$\frac{1}{n} \sum_{t=1}^{n} \overline{x}^{(t)} \left(\overline{x}^{(t)} \right)^T \overline{\theta} + \lambda \overline{\theta} = \frac{1}{n} \sum_{t=1}^{n} \overline{x}^{(t)} y^{(t)}$$

De forma matricial se ve de la siguiente manera:

$$\frac{1}{n} X^T X \overline{\theta} + \lambda \overline{\theta} = \frac{1}{n} X^T \overline{y}$$

$$\left[\frac{1}{n} X^T X + \lambda I \right] \overline{\theta} = \frac{1}{n} X^T \overline{y}$$

Finalmente se resuelve para lambda y de forma final matricial tenemos:

$$\overline{\theta} = \left[\frac{1}{n} X^T X + \lambda I \right]^{-1} \frac{1}{n} X^T \overline{y}$$

Referencias

https://web.mit.edu/zoya/www/linearRegression.pdf

6. Tarea 6: Métricas de Evaluación – Regresión

¿Cómo se ve el Error Cuadrático Medio desde la perspectiva de estimar un parámetro? Escribir la forma matemática

Definición:
El error cuadrático medio de un estimador $\hat{\theta}$ de un parámetro θ es la función de θ definida por: $E[\theta - \hat{\theta}^2]$ [1]

Investigar en que consiste el R cuadrado ajustado
La R^2 ajustada es una medida corregida de bondad de ajuste o de precisión del modelo para los modelos lineales. Identifica el porcentaje de varianza que se explica por las variables regresoras o entradas. El R^2 va incrementando al ir añadiendo variables regresoras al modelo, esto puede llevar a un sobre ajuste. La R^2 ajustada solamente incrementa si la variable regresora que se añade al modelo es útil y ayuda a mejorar el modelo. La R^2 ajustada se calcula dividiendo el error cuadrático medio residual por el error cuadrático total y le restamos este resultado a 1 es decir: $R^2_{Adj} = 1 - \frac{MS_{res}}{MS_T}$ [2]

Referencias

1. http://people.missouristate.edu/songfengzheng/teaching/mth541/lecture %20notes/evaluation.pdf
2. https://www.ibm.com/docs/es/cognos-analytics/11.2.0?topic=terms-adjusted-r-squared

7. Tarea 7: Bayes Ingenuo

Gaussian Naïve Bayes

Diferencias con el método base de Bayes Ingenuo

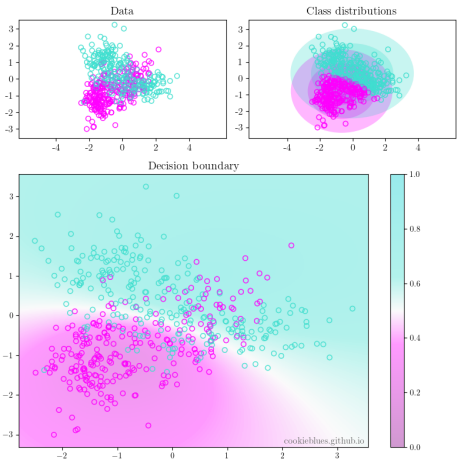
El método de Gaussian Naïve Bayes asume que cada clase sigue una distribución gaussiana o normal. En Naïve Bayes no se asume ningún tipo de distribución. Ambos modelos asumen que las características son independientes entre si. En Gaussian Naïve Bayes se asume que las distribuciones condicionales $(x_i|C_k)$ se distribuyen normales [1]:
$(x|C_k, \mu_k, \Sigma_k) = N(x|\mu_c, \Sigma_c)$
Entonces la hipótesis o clasificación queda como:
$argmax_k P(x|C_k, \mu_k, \Sigma_k)$

Cambios en la formulación

Para el modelo tenemos [2]:
$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} exp(-\frac{(x_i-\mu_k)^2}{2\sigma_k^2})$
Los parámetros μ_k y σ_k se estiman con máxima verosimilitud
Para la derivación y entrenamiento se tiene [1]:
$\pi_k = \frac{N_k}{N}$
$\mu_k = \frac{1}{N_c} \sum_{i=1}^{N} C_k * x_i$
$\sigma_k = diag(\frac{1}{N_c} \sum_{i=1}^{N} C_k * (x_i - \mu_k)(x_i - \mu_k)^T)$
donde $diag$ se refiere que cada valor fuera de la diagonal se iguala a 0.

Extra: interpretaciones gráficas

Si se llega a implementar en Python pueden resultar las siguientes gráficas [1]:



Referencias

1. Hrouda-Rasmussen, S. (2021, May 7). How (Gaussian) Naive Bayes Works. Towards Data Science. Retrieved September 26, 2022, from https://towardsdatascience.com/

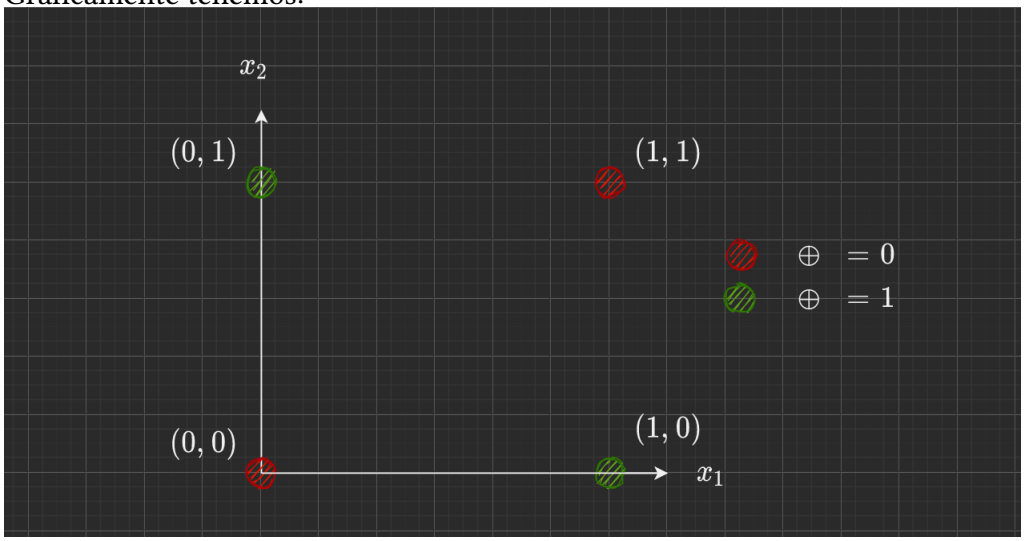
2. Scikit Learn. (n.d.). 1.9. naive Bayes. scikit. Retrieved September 26, 2022, from https://scikit-learn.org/stable/modules/naivebayes.html

8. Tarea 8: Introducción a las Redes Neuronales

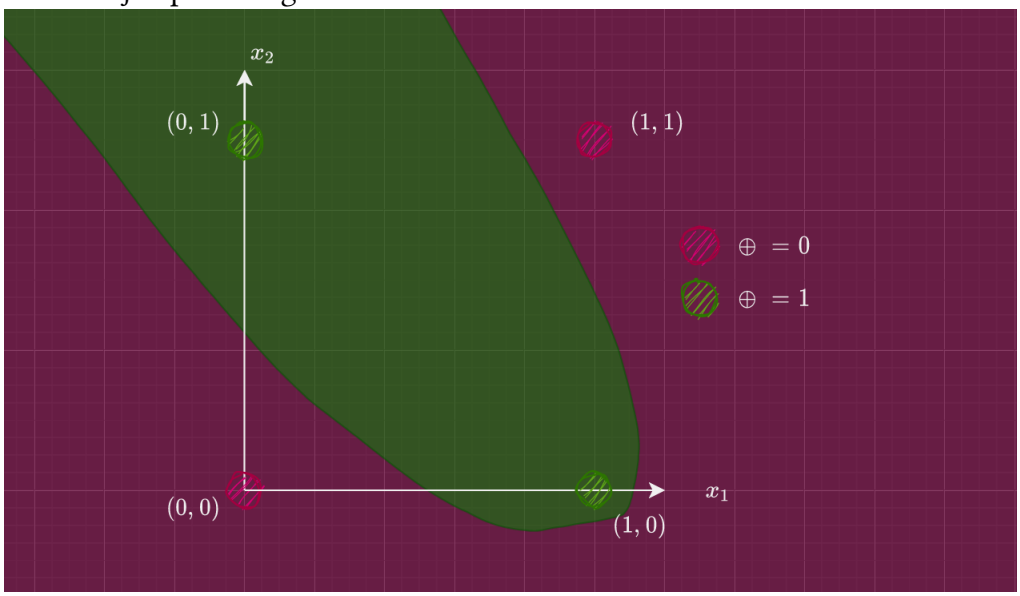
Neurona de McCulloch y Pitts

1. El problema de determinar XOR con la neuroma de McCulloch y Pitts
   La neurona de McCulloch y Pitts solamente podía hacer divisiones de manera lineal; es un margen de decisión lineal. XOR requiere de un margen de decisión no lineal (ver pregunta 2 con la gráfica). Para poder hacer decisiones de manera no lineal se requiere una red neuronal, no se puede hacer solamente con una neurona. En 1969 Marvin Minsky, en *Perceptron's – An Introduction to Computational Geometry* argumenta que los perceptrones y la neurona de McCulloch y Pitts no funciona ya que no puede resolver problemas no lineales como la función XOR. [1]

2. ¿Cómo se ve gráficamente el XOR?
   Gráficamente tenemos:

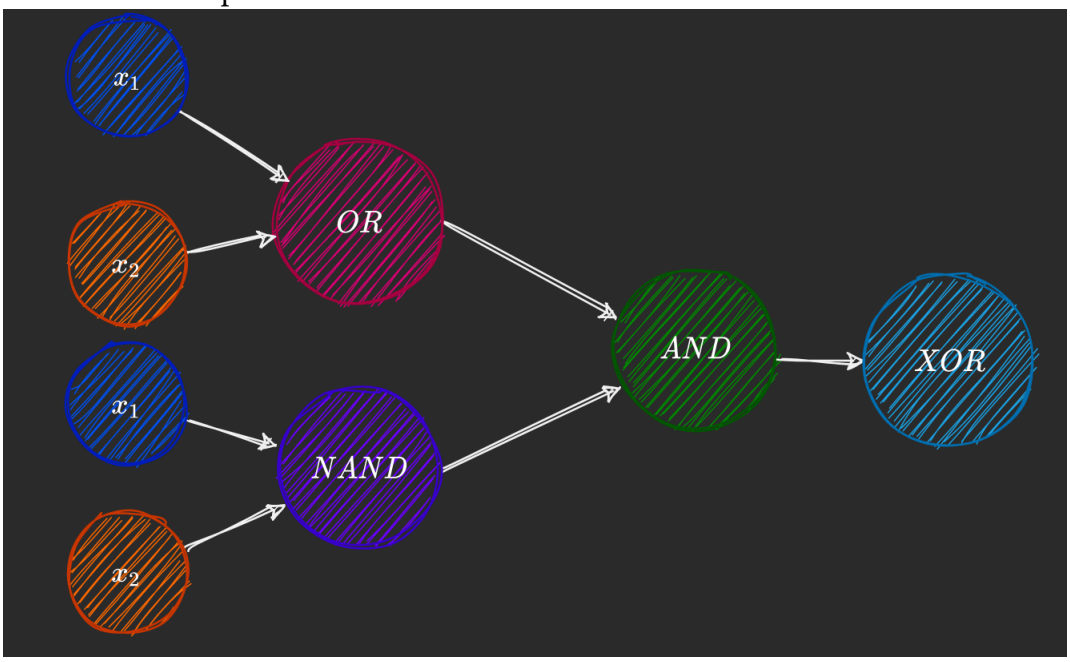


Fuente: [2]
Con un ejemplo de algoritmo de decisión:



Fuente [2]

3. ¿Cómo se resuelve el problema del XOR con la neurona de McCulloch y Pitts?
   La función XOR es una combinación de las funciones OR y NAND. Entonces, para resolver el problema de XOR con la neurona de McCulloch y Pitts primero se tienen que resolver u obtener las salidas de las funciones OR y NAND con la neurona de McCulloch y Pitts y luego meter dichas salidas como entradas a la función AND con la neurona de McCulloch y Pitts. Recordemos que XOR se puede ver como: OR AND NAND [2].
   Gráficamente se puede ver como:



   Fuente [2]

4. ¿Qué causó este problema en el campo de las redes neuronales y en la inteligencia artificial?
   Tras la publicación de Minsky, se pensaba que las redes neuronales y la inteligencia artificial servían de poco o nada para resolver problemas más complejos. Esto llevó a una época oscura conocido como el invierno de la inteligencia artificial. Durante este periodo se paralizó el mundo de la Inteligencia Artificial ya que se le dejó de invertir dinero, tiempo y fondos y hubo poco o nada de investigación en este campo. [1]

Referencias

1. Dharanikota, D. (2018, March 8). History of neural networks. LinkedIn. Retrieved October 15, 2022, from https://www.linkedin.com/pulse/history-neural-networks-datta-dharanikota

2. Karajgi, A. (2021, July 21). How neural networks solve the XOR problem. Medium. Retrieved October 15, 2022, from https://towardsdatascience.com/how-neural-networks-solve-the-xor-problem-59763136bdd7

Ejercicio XNOR

Tercera 7 ML

XNOR

Tenemos las funciones

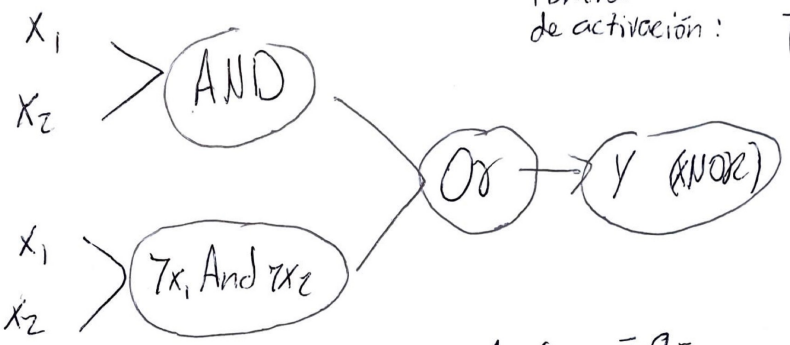
And: $-30+20x_1+20x_2$

$\neg x_1$ And $\neg x_2$: $10-20x_1-20x_2$

Or: $-10+20x_1+20x_2$

$x_1, x_2 \in \{0,1\}$

Función de activación: $\frac{1}{1+e^{-w^T x}}$



Sea AND $= a_1$ y $\neg x_1$ And $\neg x_2 = a_2$ y XNOR $= Y$

Tenemos

x_1	x_2	a_1	a_2	y
1	1	1	0	1
1	0	0	0	0
0	1	0	0	0
0	0	0	1	1

entran x_1, x_2

Para a_1 (AND)

$\{1,1\} = \frac{1}{1+e^{-(-30+20+20)}} \approx 1$

$\{0,1\} = \frac{1}{1+e^{-(-30+0+20)}} \approx 0$

$(1,0) = \frac{1}{1+e^{-(-30+20+0)}} \approx 0$

$\{0,0\} = \frac{1}{1+e^{-(-30+0+0)}} \approx 0$

entran x_1, x_2

Para a_2 ($\neg x_1$ AND $\neg x_2$)

$\{1,1\} = \frac{1}{1+e^{-(10-20-20)}} \approx 0$

$\{1,0\} = \frac{1}{1+e^{-(10-20-0)}} \approx 0$

$\{0,1\} = \frac{1}{1+e^{-(10-0-20)}} \approx 0$

$(0,0) = \frac{1}{1+e^{-(10-0-0)}} \approx 1$

Para Y (OR) entran a_1 y a_2

$\{1,0\} = \frac{1}{1+e^{-(-10+20+0)}} \approx 1$

$\{0,0\} = \frac{1}{1+e^{-(-10+0+0)}} \approx 0$

$\{0,1\} = \frac{1}{1+e^{-(-10+0+20)}} \approx 1$

9. Tarea 9: Redes Neuronales – Backpropagation

Ejercicio Backpropagation

Ejercicio Backpropagation

Sea $J(W) = \frac{1}{2}\sum_{k}(t_k - a_k)^2$ determinar las reglas de actualización mediante backpropagation

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_{kj}}$$

$$\frac{\partial J}{\partial a}\left[\frac{1}{2}(t_k - a_k)^2\right] = \frac{1}{4}(t_k - a_k)(-1)$$

$$= -\frac{1}{4}(t_k - a_k)$$

$$a = \frac{1}{1+e^{-z}} \Rightarrow \frac{\partial}{\partial z} = \frac{-e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(-\frac{1}{1+e^{-z}} + 1\right)$$

$$= (\sigma(z))(1-\sigma(z))$$

$$= a_k(1-a_k)$$

$$= a_k(1-a_k)$$

$$z = w_{kj}a_j + b_j$$

$$\frac{\partial z}{\partial w_{kj}} = a_j$$

$$\frac{\partial J}{\partial w} = -\frac{1}{4}(t_k - a_k)\left[(a_k)(1-a_k)\right][a_j]$$

$$\underbrace{-\frac{1}{4}(t_k - a_k)(a_k - a_k^2)a_j}_{\partial x}$$

$$\Delta w_{ji} \propto -\left[\left[-\frac{1}{4}(t_k - a_k)(a_k - a_k^2)\right]w_{kj}\right]\left[a_j(1-a_j)\right]a_i \Rightarrow \sigma'(z_i)$$

Reglas $\Rightarrow$

$$\partial^{(l)} w \partial^{(l+1)} \sigma'(z^{(l)})$$

con $\partial^{(l+1)} = -\frac{1}{4}(t_k - a_k)(a_k - a_k^2)$

Teorema de Convergencia del Perceptrón

Antes de plantear el Teorema de Convergencia del Perceptrón se da la siguiente definición:

Definición: Dos conjuntos de A y B son linealmente separables en un espacio n-dimensional si existen $n+1$ números reales $w_1, ..., w_k, \theta$ de manera que cada punto $(x_1, ..., x_n) \in A$ satisface $\sum_{j=1}^{n} w_j x_j >= \theta$ y cada punto $(x_1, ..., x_n \in B$ satisface $\sum_{j=1}^{n} w_j x_j < \theta$

Teorema de Convergencia:
Si el conjunto de patrones de entrenamiento $x^1, z^1, x^2, z^2, ..., x^p, z^p$ es linealmente separable entonces el Perceptrón simple encuentra una solución en un número finito de de iteraciones, es decir, consigue que la salida de la red coincida con la salida deseada para cada uno de los patrones de entrenamiento.

Demostración:

Como los patrones son linealmente separables existirán unos valores $w_1, ..., w_n, \theta^*$ tal que

$$\sum_{j=1}^{n} w_j^* x_j > \theta^* \text{ para los patrones de la clase 1}$$

$$\underbrace{\sum_{j=1}^{n} w_j^* x_j}_{\textit{salida deseada}} < \theta^* \text{ para los patrones de la clase 0}$$

Supongamos que en la iteración k la red tiene que modificar los pesos sinápticos según la regla de aprendizaje, ya que la salida de la red $y(k)$ no coincide con la salida deseada $z(k)$, es decir, $y(k) \neq z(k)$.

Tendremos que:

$$\sum_{j=1}^{n+1}(w_j(k+1) - w_j^*)^2 = \sum_{j=1}^{n+1}(w_j(k) + \eta[z(k) - y(k)]x_j(k) - w_j^*)^2 =$$

$$= \sum_{j=1}^{n+1}(w_j(k) - w_j^* + \eta[z(k) - y(k)]x_j(k))^2 =$$

$$= \sum_{j=1}^{n+1}(w_j(k) - w_j^*)^2 + \eta^2[z(k) - y(k)]^2 \sum_{j=1}^{n+1} x_j(k)^2 + 2\eta[z(k) - y(k)] \sum_{j=1}^{n+1}(w_j(k) - w_j^*)x_j(k) =$$

$$= \sum_{j=1}^{n+1}(w_j(k) - w_j^*)^2 + \eta^2[z(k) - y(k)]^2 \sum_{j=1}^{n+1} x_j(k)^2 + 2\eta[z(k) - y(k)](\sum_{j=1}^{n+1}(w_j(k)x_j(k)) -$$

$$2\eta[z(k) - y(k)](\sum_{j=1}^{n+1}(w_j^* x_j(k))$$

Observamos que $[z(k) - y(k)] \sum_{j=1}^{n+1} w_j(k)x_j(k) < 0$,

ya que si $\sum_{j=1}^{n+1} w_j(k)x_j(k) > 0 \implies y(k) = 1$ y como la salida es incorrecta $(y(k) \neq z(k))$ tiene que ser $z(k) = -1$.

y si $\sum_{j=1}^{n+1} w_j(k)x_j(k) < 0 \implies y(k) = -1$ y como la salida es incorrecta $(y(k) \neq z(k))$ tiene que ser $z(k) = 1$.

Este término se puede escribir de la forma $-2|\sum_{j=1}^{n+1} w_j(k)x_j(k)|$.

Y el término $[z(k)-y(k)]\sum_{j=1}^{n+1} w_j^* x_j(k) > 0$,

ya que si $\sum_{j=1}^{n+1} w_j^* x_j(k) > 0 \implies$ la salida deseada $z(k) = 1$ y la salida incorrecta de la red tiene que ser $y(k) = -1$.

y si $\sum_{j=1}^{n+1} w_j^* x_j(k) < 0 \implies z(k) = -1$ y la salida incorrecta de la red tiene que ser $y(k) = 1$.

Este término se puede escribir de la forma $2|\sum_{j=1}^{n+1} w_j^* x_j(k)|$.

Por lo tanto, prescindiendo de un término negativo a la derecha de la expresión, tenemos que:

$$\sum_{j=1}^{n+1}(w_j(k+1)-w_j^*)^2 \leq \sum_{j=1}^{n+1}(w_j(k)-w_j^*)^2 + 4\eta^2\sum_{j=1}^{n+1} x_j(k)^2 - 4\eta|\sum_{j=1}^{n+1} w_j(k)x_j(k)|$$

Sea

$$\begin{cases} D(k+1) = \sum_{j=1}^{n+1}(w_j(k+1)-w_j^*)^2 \\ D(k) = \sum_{j=1}^{n+1}(w_j(k)-w_j^*)^2 \\ T = \text{mín}_{1\leq k\leq p}\{|\sum_{j=1}^{n+1} w_j^* x_j(k)|\} \end{cases}$$

Como $\sum_{j=1}^{n+1} x_j(k)^2 = ||\mathbf{x}(k)||^2 = n+1$, se tiene la siguiente desigualdad

$$D(k+1) \leq D(k) + 4\eta^2(n+1) - 4\eta T$$

$$D(k+1) \leq D(k) + 4\eta[\eta(n+1) - T]$$

Si tomamos $\eta(n+1) - T < 0$, es decir, $\eta < \frac{T}{n+1}$, entonces$D(k+1) \leq D(k)$.

Esto significa que eligiendo un valor de η tal que $0 < \eta < \frac{T}{n+1}$ hace que $D(k)$ disminuya al menos en la cantidad constante $4\eta[\eta(n+1) - T]$ en cada iteración, con corrección.

Si el número de iteraciones con corrección fuese infinito entonces llegaríamos al absurdo de alcanzar en un momento determinado un valor negativo para el término $D(k)$ que evidentemente no puede ser negativo.

□

Referencias

https://zaguan.unizar.es/record/69205/files/TAZ-TFG-2018-148.pdf

10. Tarea 10: K-Means

K-means ++ método de inicialización

Inicializar los centroides de los clústeres de manera aleatoria lleva al problema de sensitividad de la inicialización. Esto lleva a que en ciertas ocasiones los clústeres formados no sean los correctos o los deseados; que no se formen bien. Existen dos métodos para evitar este problema al iniciar los centroides:

1. Repetir K-medias: este método fue revisado en clase y se elige finalmente el clúster que tenga el menor costo o valor de la función de error J.

2. K-means ++: este es una técnica inteligente de inicialización y luego se prosigue con el algoritmo tradicional de k-medias.

Para inicializar los centroides utilizando este método se utilizan los siguientes pasos:

1. Elegir el primer punto o centroide c_1 de manera aleatoria.

2. Calcular la distancia de todos los puntos del conjunto de datos y el centroide seleccionado. La distancia de cada punto x_i del centroide más lejano se puede calcular de la siguiente manera:

$$d_i = max_{(j:1 \mapsto m)} ||x_i - C_j||^2$$

3. Sea d_i la distancia del punto x_i más lejano del centroide y sea m el número de centroides ya seleccionados

4. El punto x_i más lejano dado por la distancia d_i es asignado como el nuevo centroide que tiene mayor probabilidad proporcional a d_i

5. Repetir los pasos 2 a 4 hasta encontrar los k centroides requeridos

Referencias

https://towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca

11. Tarea 11: PCA y SVD

SVD:

Con la Descomposición en Valores singulares de la matriz de covarianza una matriz se puede descomponer en 3 matrices: U, Σ y V. Donde U es la matriz de eigen vectores que salen de $A^T A$ y Σ es la matriz de covarianza [1].

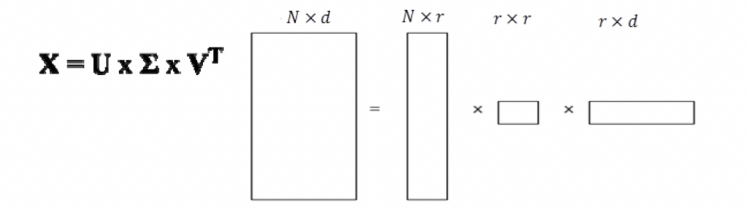
Se puede descomponer una matriz A de la siguiente manera [1]:

$$A = U\Sigma V^T$$

La matriz se puede descomponer aún más de la siguiente manera [1]:

$$\boldsymbol{A = u_1 \sum_1 v_1^T + u_2 \sum_2 v_2^T + \cdots + u_r \sum_r v_r^T}$$

Gráficamente SVD se ve de la siguiente manera [2]:



Relación entre SVD y PCA

Teorema:

Sea $A = U\Sigma V^T$ el SVD de una matriz A de Nxd y sea $C = \frac{1}{N-1} X^T X$ la matriz de dimensiones dxd de covarianza. Los eigenvectores de C son los mismos que los *right singular vectors* (vectores derechos singulares) de A [2].

Demostración:

$$X^T X = V\Sigma U^T U\Sigma V^T = V\Sigma\Sigma V^T = V\Sigma^2 V^T$$
$$C = V\frac{\Sigma^2}{N-1}V^T$$

Pero C es una matriz simétrica y entonces:

$$C = V\Lambda V^T$$

Por lo tanto, los eigenvectores de la matriz de covarianza son los mismos de la matriz V (vectores derechos singulares) y los eigenvalores de C se pueden calcular de los valores singulares de la siguiente manera [2]:

$$\lambda_i = \frac{\sigma_i^2}{N-1}$$

Visto de una manera similar se puede plantear de esta forma [1]:

$$\boldsymbol{A^T A} = (n-1)\boldsymbol{S} = \boldsymbol{V\Sigma^2 V^T}$$
$$\boldsymbol{S} = \boldsymbol{V}\left(\frac{\boldsymbol{\Sigma^2}}{n-1}\right)\boldsymbol{V^T}$$
$$\boldsymbol{\lambda} = \frac{\boldsymbol{\sigma^2}}{n-1}$$

Donde para una Matriz X, el k-ésimo componente principal es el vector derecho singular de la matriz de covarianza de X correspondiente al k-ésimo valor más grande de los valores singulares. Finalmente para la selección de los componentes principales para elegir el valor de k, es decir, cuantos componentes principales, se elige el valor más pequeño tal que:

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 1 - \alpha$$

Donde S es la matriz de covarianzas (solo se suman los elementos de la diagonal) es decir que la suma de las varianzas de los k componentes respecto a las varianzas de todos los componentes iniciales (n) son mayores o iguales a $1 - \alpha$ donde esto especifica cuánta variabilidad de los datos se busca mantener tras hacer la reducción de dimensionalidad. Es decir que queremos las menos dimensiones posibles, el k más pequeño, que nos maximice la variabilidad de los datos [3].

Referencias

1. https://towardsdatascience.com/singular-value-decomposition-and-its-applications-in-principal-component-analysis-5b7a5f08d0bd

2. https://people.cs.pitt.edu/ milos/courses/cs3750-Fall2014/lectures/class9.pdf

3. https://towardsdatascience.com/how-to-select-the-best-number-of-principal-components-for-the-dataset-287e64b14c6d