ML /
~~Sist. de Recomendación~~

~~Sistemas no personalizados~~
~~Introducción a ML~~

KNN

Es un modelo que clasifica un punto como la clase de sus "k" vecinos más cercanos. En el caso de un problema de regresión se considera el promedio de los "k" vecinos.

KNN no necesita entrenamiento pero es muy lento a la hora de predecir.

A la hora de ~~[illegible]~~ buscar los "k" vecinos más cercanos podemos utilizar algunas estructuras de datos como:

- K-D Tree
- LSH (aproximación).
- V-P Trees
- Etc.

Hay que hallar el valor óptimo de k (k chico => overfitting, k grande => underfitting) se puede usar utilizando "leave one out cross validation".

Es muy importante que los atributos de los puntos estén normalizados. De lo contrario un atributo puede dominar las distancias.

Otro hiper-parametro es la función de distancia a usar. Algunos ejemplos:

- Euclideana
- Manhattan
- Coseno
- Jaccard.

Regresión lineal.

Es un modelo que intenta ajustar una linea a un conjunto de datos. Es decir, queremos hallar θ_0 y θ_1 que más se acerque a nuestros datos, según la ecuación de la recta:

$$y = \theta_1 x + \theta_0$$

Planteamos la hipótesis de regresión lineal:

$$h_\theta(\bar{x}) = \theta^T \bar{x} \quad \text{con } \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix} \text{ y } \bar{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Buscamos θ tal que el error entre $h_\theta(\bar{x}_i)$ y y_i sea mínimo.

ML 2

Descenso de gradiente.

El gradiente dice para donde aumenta más la función, si voy en sentido contrario voy a donde disminuye más. De esta forma utilizando el gradiente de nuestra función de costo (error) podemos minimizarla.

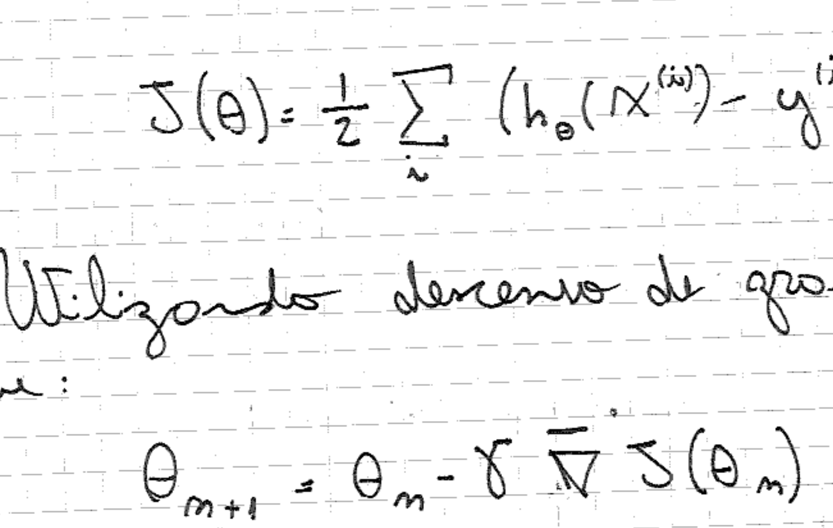
La función de error/costo para la regresión lineal es:

$$J(\theta) = \frac{1}{2}\sum_{i} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Utilizando descenso de gradiente llegamos a que:

$$\theta_{n+1} = \theta_n - \gamma \, \bar{\nabla} J(\theta_n).$$

Donde γ es el learning rate (hiperparámetro).



Perceptrón Clásico

Tenemos una función de activació (predicció).

$$f_\theta(x) = \begin{cases} 1 \text{ si } \bar{\theta}^T \bar{x} \geq 0 \\ 0 \text{ c.c} \end{cases}$$

Para actualizar $\bar{\theta}$ usamos:

$$\bar{\theta}^{t+1} = \bar{\theta}^{t} + \gamma \left(y_i - f_{\theta^t}(\bar{x}_i)\right) \bar{x}_i$$

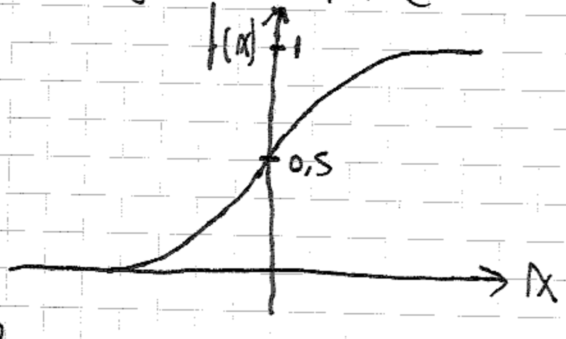
Esto se puede aplicar a un flujo de datos.

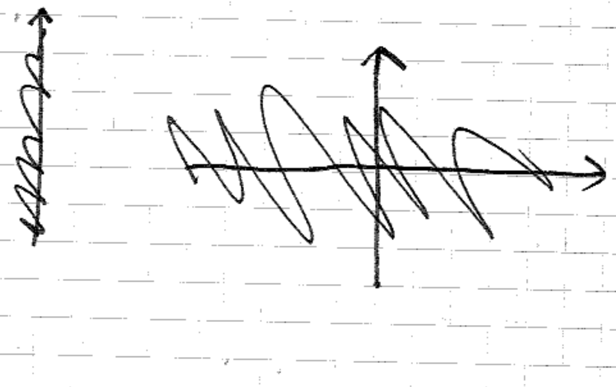
Regresión Logística

Mismo principio que la regresió lineal pero queremos predecir un valor discreto 0 o 1.

Agregamos la funció logística:

$$f(x) = \frac{1}{1+e^{-x}}$$





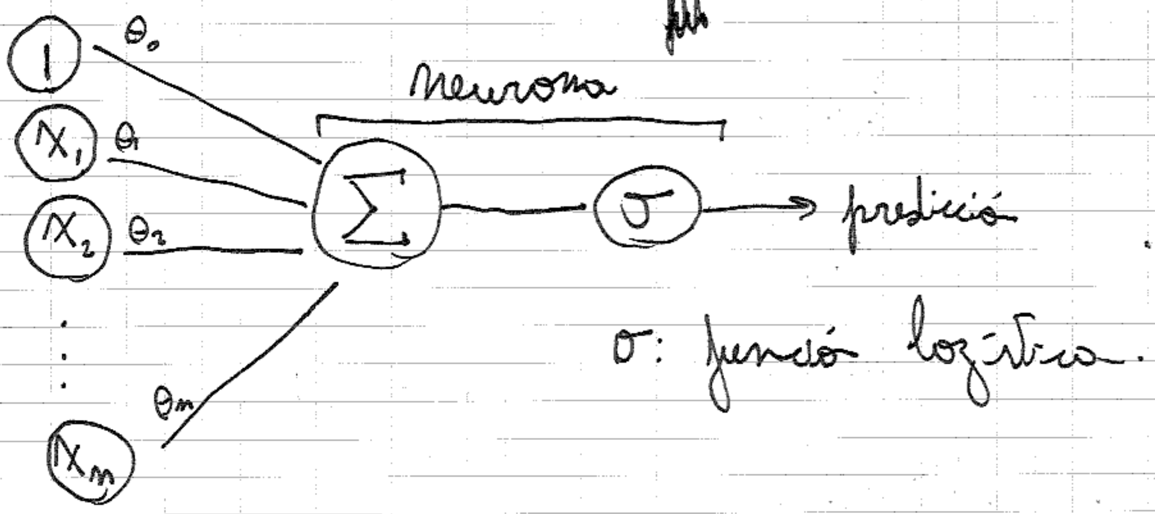
Entonces para nuestro modelo $P(y=1|\bar{x}) = \dfrac{1}{1+e^{-\bar{\theta}^T\bar{x}}}$

ML3

La función de costo para una regresión logística es:

$$J(\theta) = -\sum_{i} \left(y^{(i)} \log\left(h_\theta(\bar{x}^{(i)})\right) + (1-y^{(i)}) \log\left(1-h_\theta(\bar{x}^{(i)})\right) \right)$$

$$\bar{\nabla} J(\theta) = \sum_{i} \bar{x}^{(i)} \left(h_\theta(\bar{x}^{(i)}) - y^{(i)}\right)$$



σ: función logística.

Regresión Softmax.

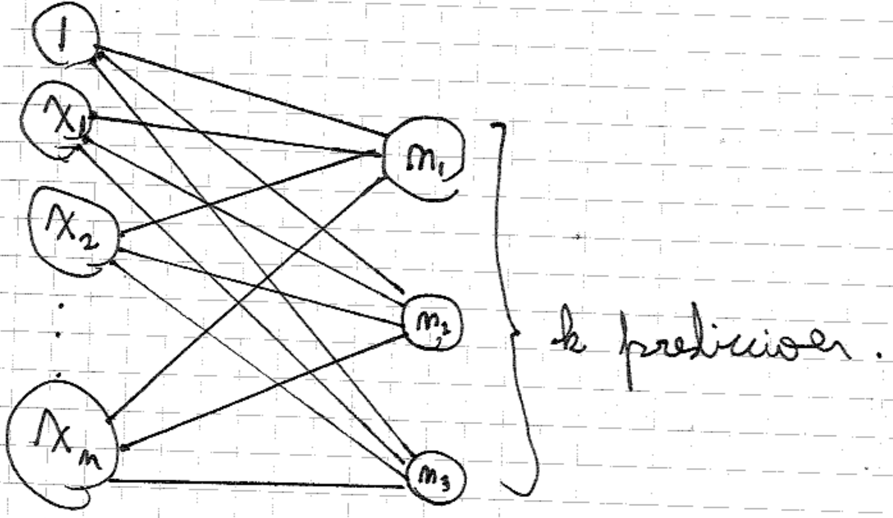
Es una generalización de la regresión logística para el caso de múltiples clases.

Reescribimos el caso binario como:

$$P(x|\theta) = \begin{pmatrix} \exp(\bar{\theta}^{(1)T}\bar{x}) \\ \exp(\bar{\theta}^{(2)T}\bar{x}) \\ \vdots \\ \exp(\bar{\theta}^{(k)T}\bar{x}) \end{pmatrix} \frac{1}{\sum_{i=1}^{k} \exp(\bar{\theta}^{(i)T}\bar{x})}$$

Ahora hay múltiples vectores $\Theta^{(i)}$ (uno para cada clase) y cada par del vector salida representa la probabi de que el elemento sea de esa clase.

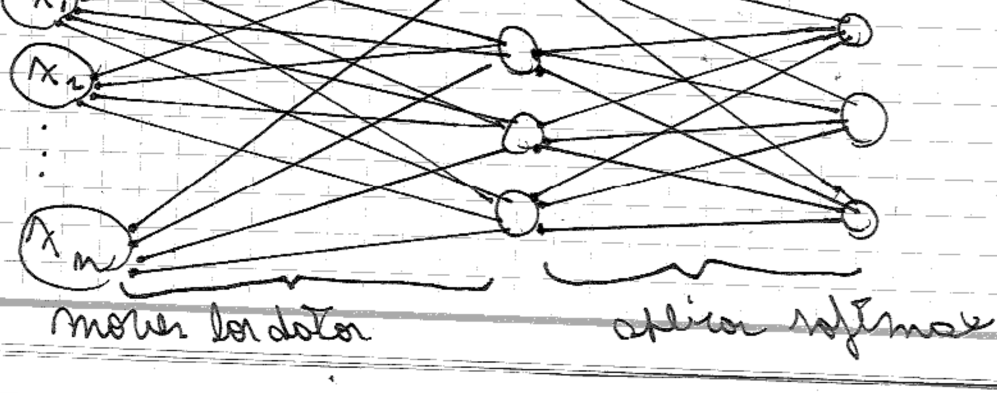
$$J(\Theta) = -\sum_{i} \sum_{k=1}^{K} \mathbb{1}\{y^{(i)}=k\} \frac{\exp(\bar{\Theta}^{(k)T}\bar{x}^{(i)})}{\sum_{j=1}^{K} \exp(\bar{\Theta}^{(j)T}\bar{x}^{(i)})}.$$



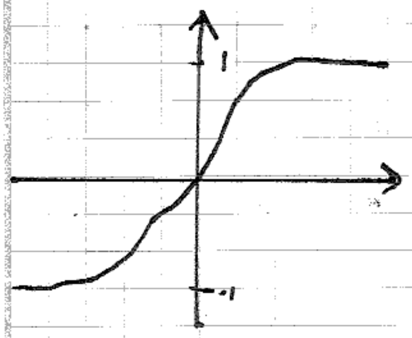
m: neurona (suma + logística)

Perceptrón multicapa:

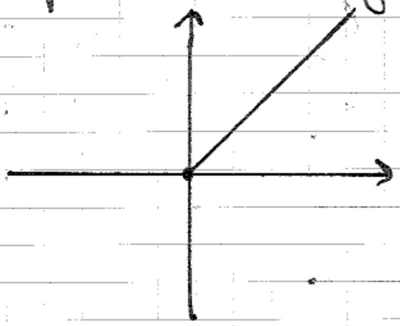
Busca resolver problemas que no pueden separarse linealmente. Para hacer esto se los proyecta a dimensiones mayores donde sí pueden ser separados.



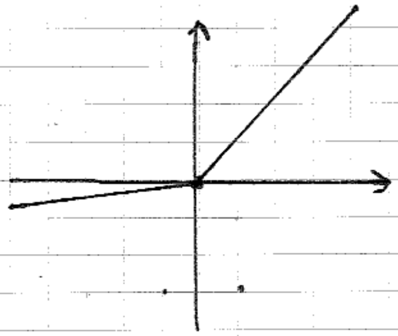
En vez de usar la sigmoide ~~[illegible]~~ para la salida de la neurona se pueden utilizar otras como:



Tanh



RELU: $\max\{0, x\}$.



LEAKY RELU.

$$\bar{x} \xrightarrow[f_1]{} f_1(\bar{x}) \xrightarrow[f_2]{} f_2(f_1(\bar{x}))$$

Se pueden utilizar más de dos capas si es necesario.

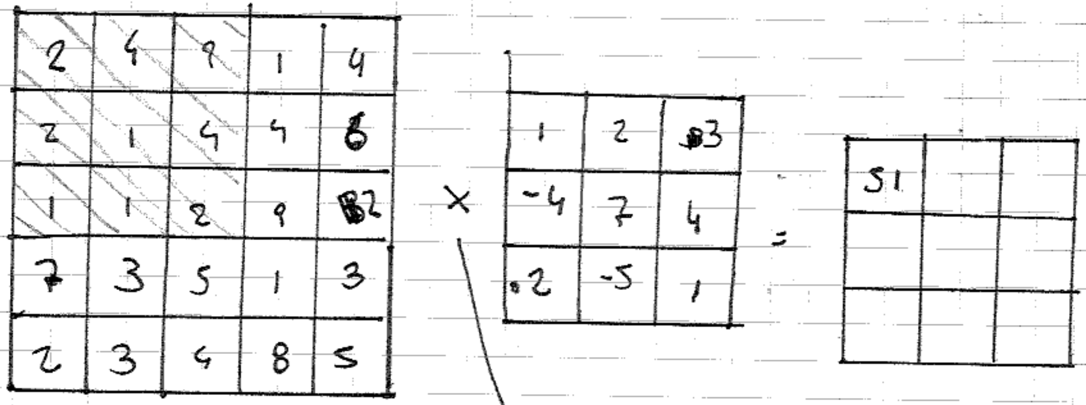
Para entrenar estas redes se utiliza "back-propagation" donde se propaga el error obtenido al final utilizando la regla de la cadena.

$$J(\theta) \leftarrow \frac{\partial \text{Costo}}{\partial \text{Capa}_{n-1}} = \frac{\partial \text{Costo}}{\partial \text{Capa}_n} \cdot \frac{\partial \text{Capa}_n}{\partial \text{Capa}_{n-1}}$$

Redes convolucionales

Se utilizan mayormente para reconocimiento de imágenes y resuelven el problema de cómo representar las imágenes para darlas a la red.

Se utilizan filtros especiales que van reduciendo el tamaño de la imagen.

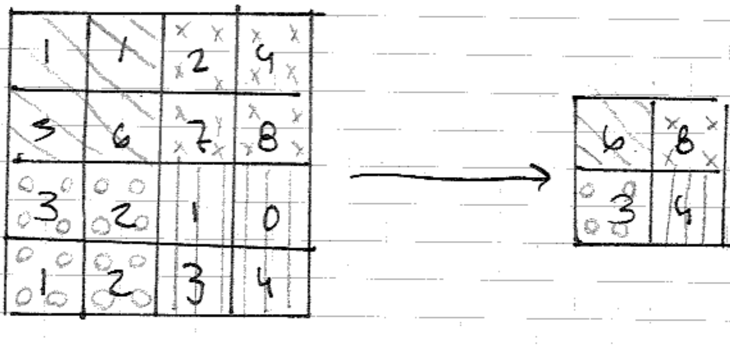


$$\begin{bmatrix} 2 & 4 & 9 & 1 & 4 \\ 2 & 1 & 4 & 4 & 6 \\ 1 & 1 & 2 & 9 & 2 \\ 7 & 3 & 5 & 1 & 3 \\ 2 & 3 & 4 & 8 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ -4 & 7 & 4 \\ 2 & -5 & 1 \end{bmatrix} = \begin{bmatrix} 51 & & \\ & & \\ & & \end{bmatrix}$$

Hiperparámetros:
- Cantidad de filtros - Padding.
- Tamaño del filtro
- Stride (paso

→ se hace elemento a elemento y después se suma.

Luego se ~~[illegible]~~ están las etapas de "max-pooling" donde se agarra el mayor elemento en un área:



$$\begin{bmatrix} 1 & 1 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 6 & 8 \\ 3 & 4 \end{bmatrix}$$

Se combinan etapas de convolució y max-pooling para ir reduciendo la imagen y luego se alimenta a una capa "densa" de red neuronal.

Conv1d

Se puede aplicar este concepto de

ML S

convolución a texto si se representan con un vector.
Para esto usamos embeddings y vectorizamos el texto para poder aplicarle estos filtros.

Redes Neuronales Profundas.

Buscan abstraerse mejor del problema utilizando más de dos capas.
Cada capa se abstrae más de los datos iniciales, obteniendo mejores representaciones de los datos.

Se presentan algunas dificultades al entrenar si las capas no aprenden al mismo ritmo (vanishing gradient problem), pero si se las entrena bien, pueden llegar a tener muchos mejores resultados.

Para evitar overfitting se puede utilizar:

- Dropout
- Batch Normalization.
- Regularization.