Aprendizaje Supervisado

Dr. José Ramón Iglesias
DSP-ASIC BUILDER GROUP
Director Semillero TRIAC
Ingenieria Electronica
Universidad Popular del Cesar

Segunda Clase

Contenido de la Segunda Clase

- ✓Introducción al ML
- ✓Etapas en la aplicación del ML
- ✓Repaso:
  - ✓Regresión Lineal y Polinomial,
  - ✓Regresión Logística
  - ✓Perceptrón.
- ✓Support Vector Machines.
  - ✓SVC/SVR.
  - ✓Datos no linealmente separables.
  - ✓Función de costo.
- ✓Redes neuronales multicapa
- ✓Naive Bayes
- ✓Repaso:
  - ✓Árboles de decisión
- ✓Ensemble learning.
  - ✓Random Forest,
  - ✓Bagging, Boosting y Voting.
- ✓Sistemas de recomendación.
  - ✓Filtrado colaborativo.
- ✓“Buenas” prácticas en la aplicación de ML

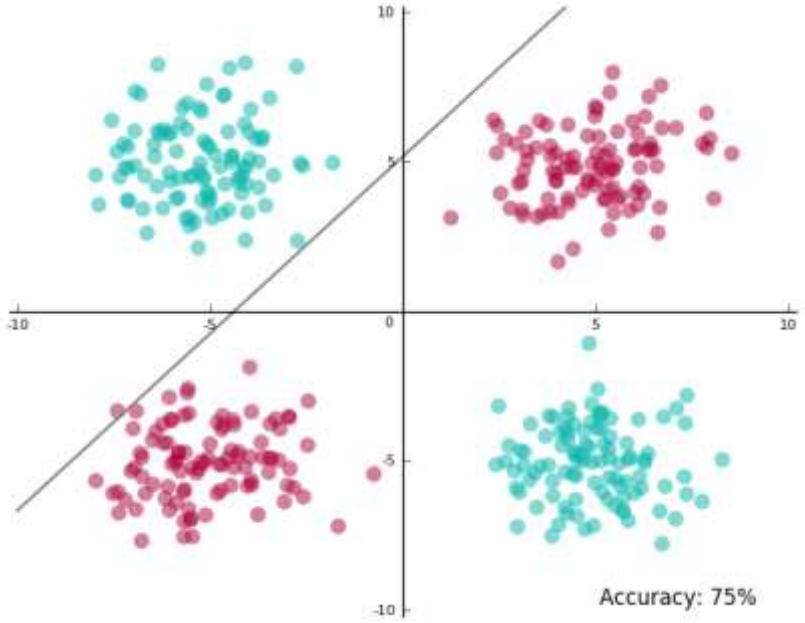
SVM con datos no linealmente separables

¿Qué hacer con datos no linealmente separables?

SVM es una técnica para separar los d
mediante un hiperplano.

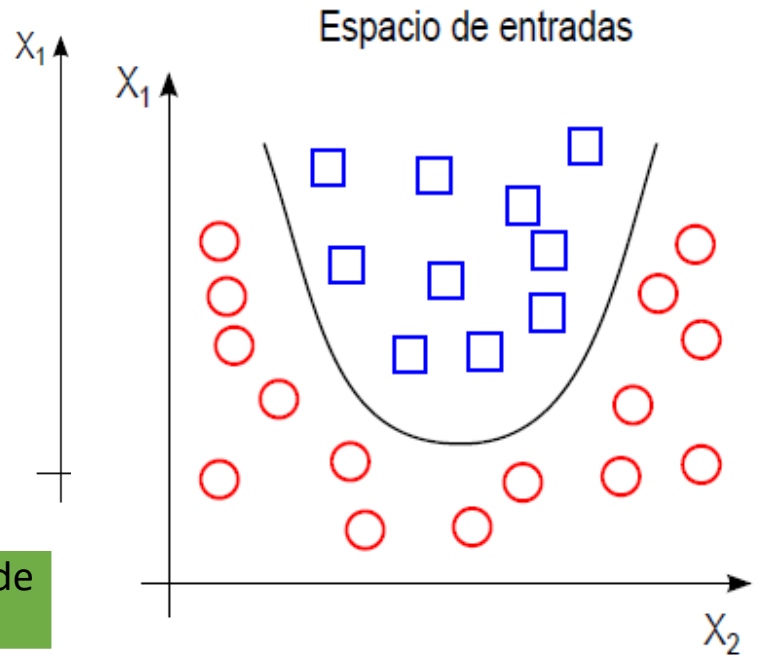
Si los datos no son linealmente separal
dicho hiperplano no existe.

Solución: Proyectar los datos a una dimensión donde sí sean linealmente separables.



Definición de espacios transformados

- Cuando los datos son no linealmente separables el Espacio de entradas
- Se definen “Espacios transformados de alta dimensionalidad”
- Para encontrar hiperplanos de separación óptima
- Cada uno de los espacios transformados se denominan “Espacio de características”
- El hiperplano en el espacio de características se transforma en una función no lineal en el espacio de entradas



¿Cómo se ve el plano proyectado?

En el ejemplo anterior, tomamos el conjunto de datos en dos dimensiones, y lo proyectamos a tres dimensiones con la siguiente ecuación:

$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2x_1\, x_2}$$

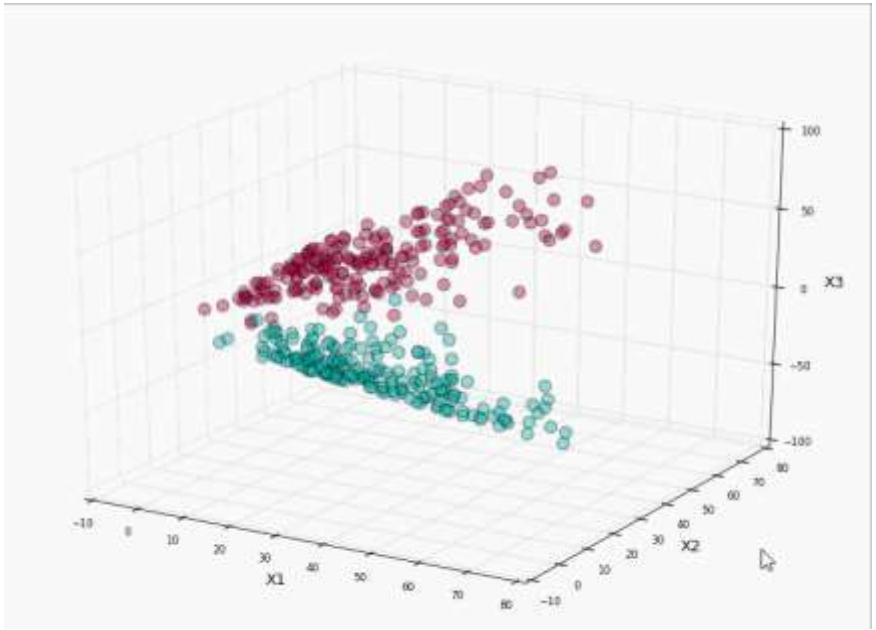


Imagen de https://blog.statsbot.co/

¿Cómo se ve el plano proyectado?

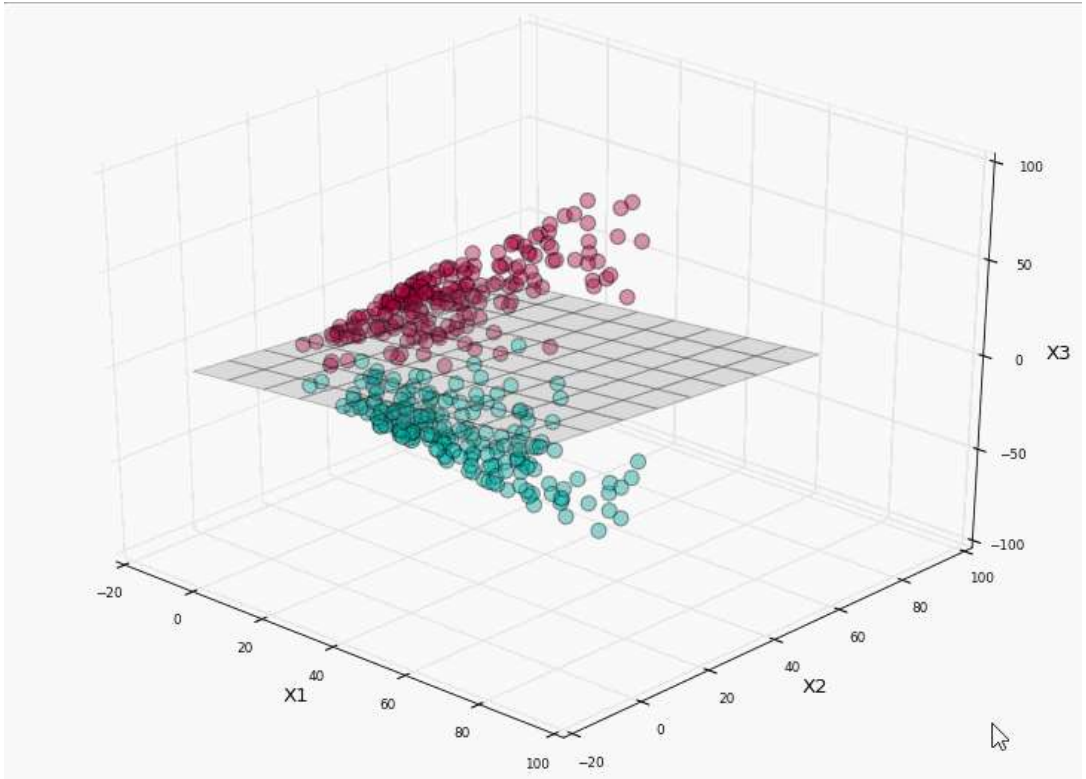


Imagen de https://blog.statsbot.co/

¿Y en el espacio de entradas?

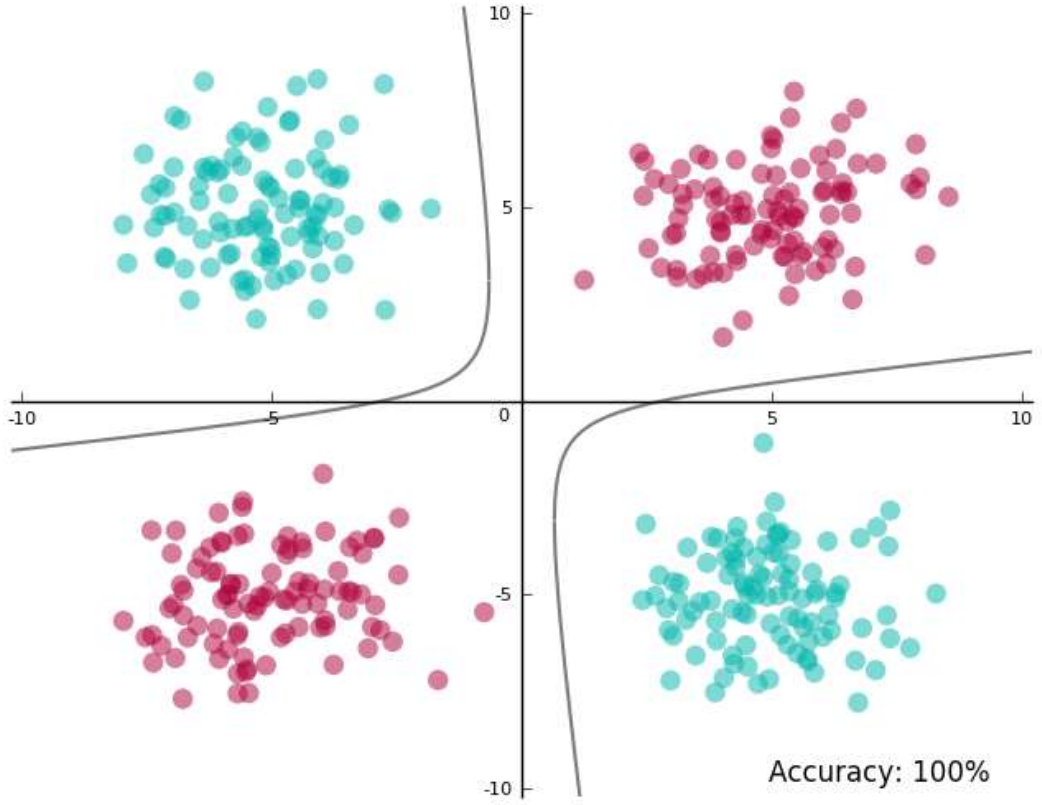


Imagen de https://blog.statsbot.co/

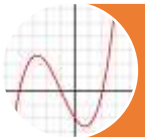
SVM: Kernels

- La manera en que el algoritmo de SVM realiza la proyección es mediante el uso de kernels.
- Las funciones de kernel toma dos puntos del espacio original (de entradas), y devuelve el producto punto en el espacio proyectado (espacio de características).
- Este producto punto es lo que la función de SVM necesita para calcular el costo.
- En el ejemplo anterior, el kernel es: $K(x_i, x_j) = \langle x_i, x_j \rangle^2$
- El proceso iterativo del SVM determina la dimensionalidad del espacio de características, pudiendo ser mucho mayor que el de entradas

¿Cómo elegir el kernel?

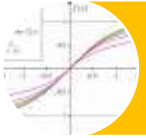
Este no es un problema trivial. Requiere mucho conocimiento matemático encontrar la proyección correcta.

En general, los frameworks más utilizados para hacer SVM tienen algunos kernels bastante comunes:



•**Polinomial**: $K(x,z) = (\langle x,z\rangle + c)^d$

Radial Basis Functions (RBF):
$K(x,z) = \exp\left(\frac{-(x-z)^2}{1\sigma^2}\right)$



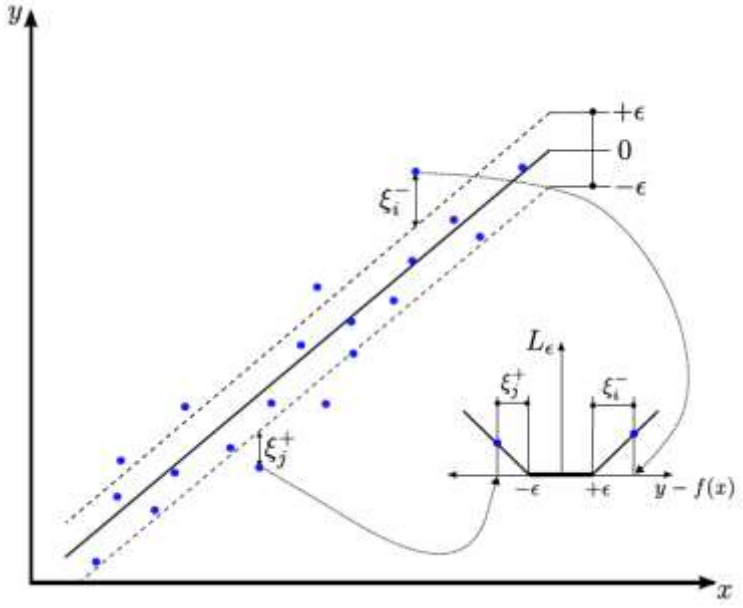
Sigmoid: $K(x,z) = \tanh(c\,\langle x,z\rangle + h)$

Support Vector Regression

Se basa en la idea de SVMs de buscar los vectores de soporte, pero en este caso el valor de y_i es un número real.

Utiliza necesariamente “márgenes blandos”, requiere un parámetro adicional ε para calcular la función de costo.

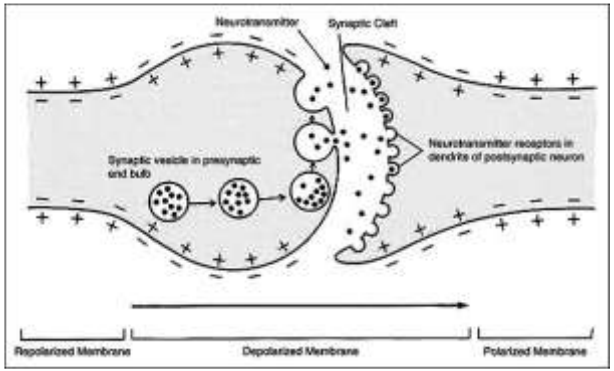
En general la regresión lineal es más popular, pero con el uso de kernels, se pueden lograr regresiones no lineales muy interesantes.

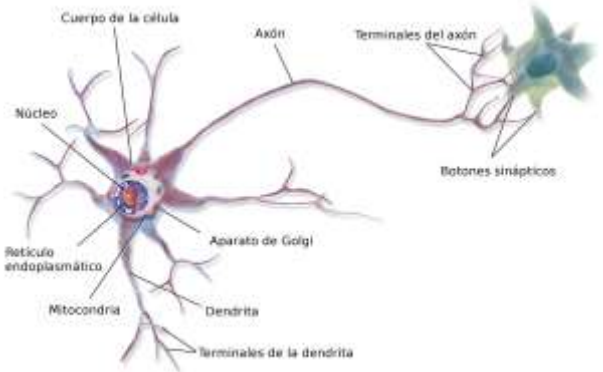


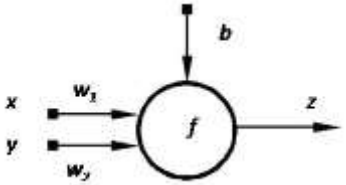
Demo Time (demo_5_kernels)

Redes Neuronales Artificiales (Introducción)

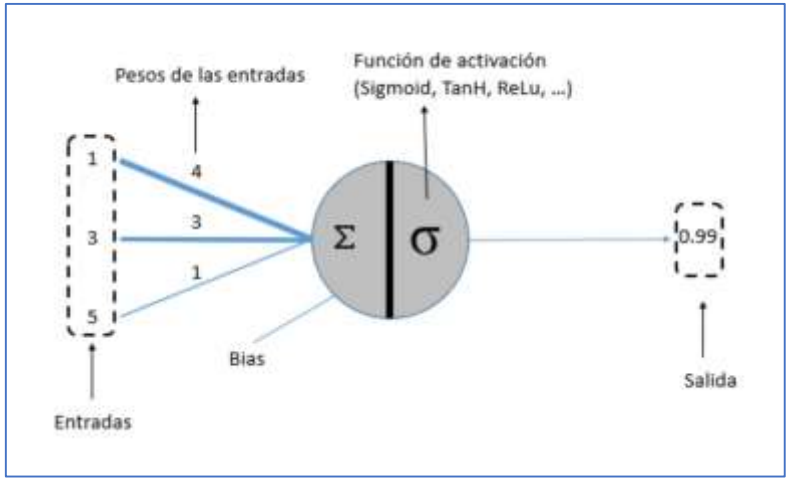
Redes Neuronales Artificiales

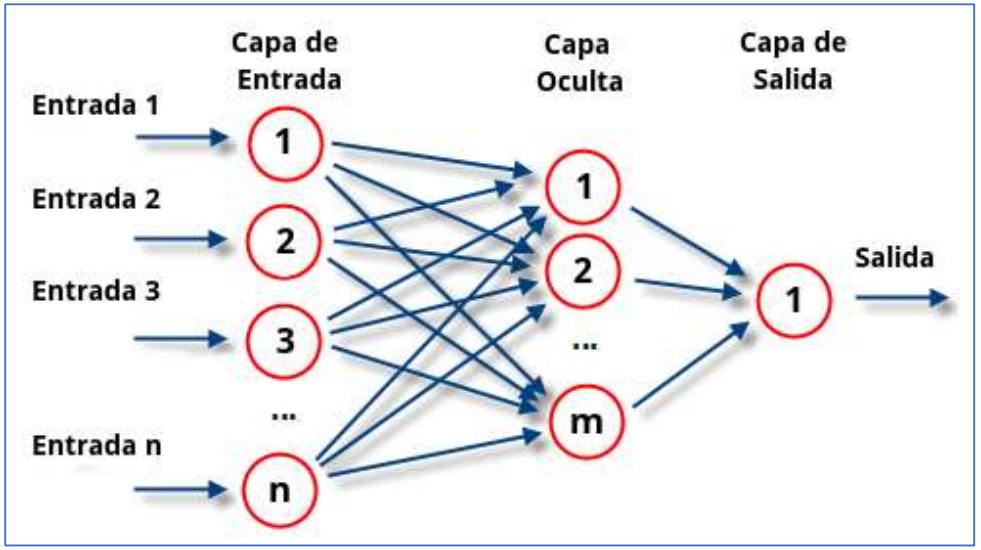




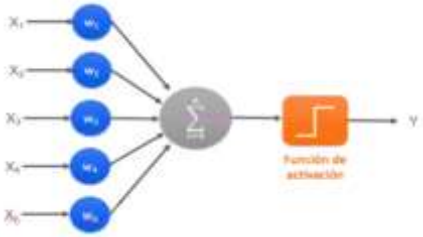


Partes de una red neuronal artificial

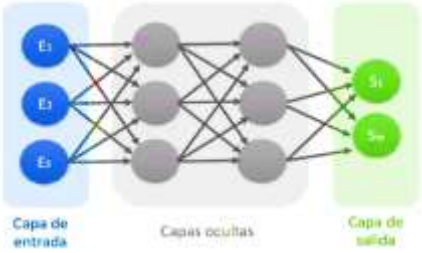




Tipos de redes neuronales artificiales

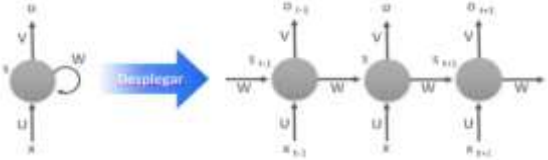


Perceptrón simple

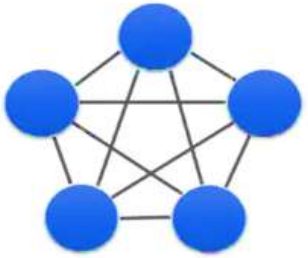


Perceptrón Multicapa

Red Convolucional



Red Neuronal Recurrente

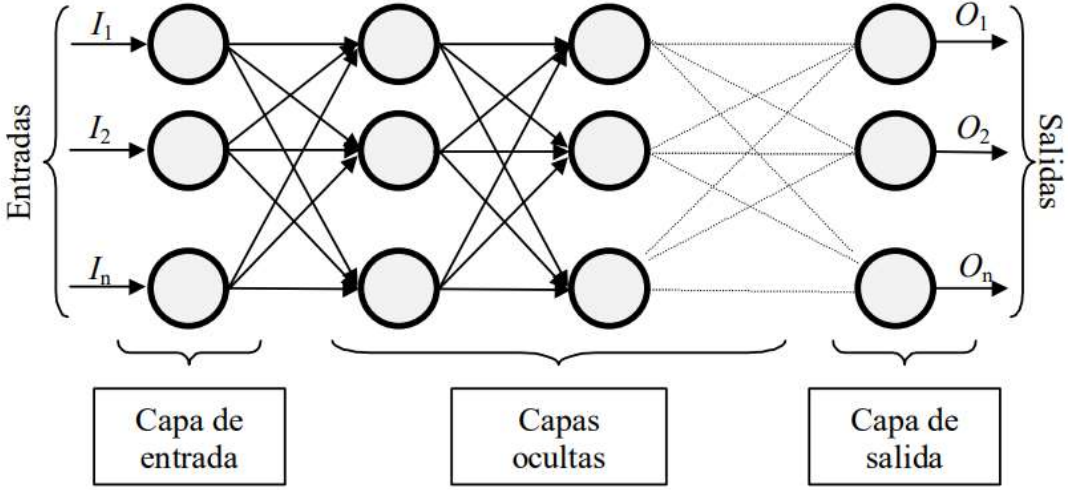


Redes de base radial

Red de Base Radial (RBF)

Tipos aprendizajes en redes neuronales artificiales

- **Supervisado**
  - Por corrección de errores
  - Estocástico
- **No supervisado**
  - Hebbiano
  - Competitivo y cooperativo
- **Por refuerzo**



¿En qué se aplican las redes neuronales artificiales?

Hay muchos tipos diferentes de redes neuronales; cada uno de los cuales tiene una aplicación particular más apropiada. Ejemplos:

- Biología
- Empresas
- Medioambiente
- Finanzas
- Manufacturación
- Medicina
- Militares
- Análisis del clima
- Análisis de la superficie terrestre

Repaso de la regresión logística: Coste

Se disponen de *m* instancias: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

Se pretende predecir $\hat{y}^{(i)} \approx y^{(i)}$

Luego, buscamos minimizar (en regresión logística):

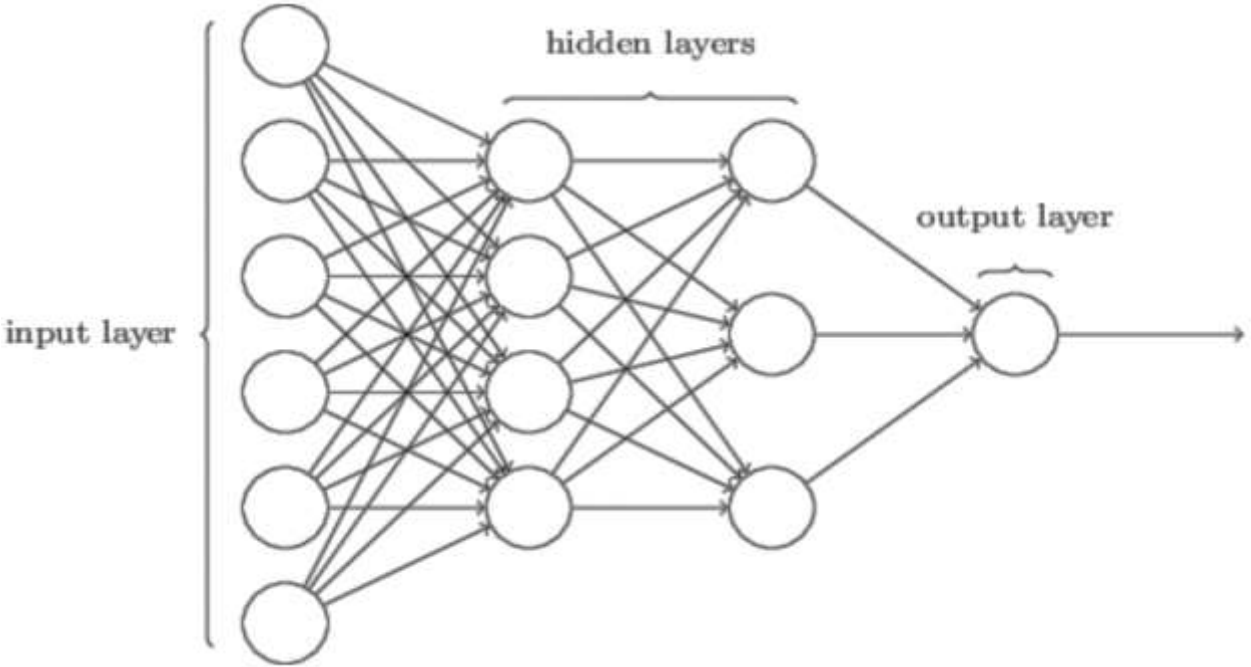
$$\mathcal{L}(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

Siendo entonces la función de coste: $\mathcal{J}(\omega, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Para minimizarla, usamos descenso por gradientes. Necesitaremos:

$$\mathcal{J}(\omega, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})$$

Redes Neuronales



Redes Neuronales

Funciones de Activación:

- **Sigmoid** (como en la regresión logística)

$$\hat{y} = g(x) = \frac{1}{1 + \exp\left(-(w^T x + b)\right)}$$

- **tanh**:

$$\hat{y} = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)}$$

- Rectified Linear Unit (**ReLU**):

$$\hat{y} = max(0, x)$$

Link

Regresión softmax (múltiples clases)

Buscamos predecir un vector de probabilidades (cada clase es una dimensión del vector).

Modificamos nuestra hipótesis:

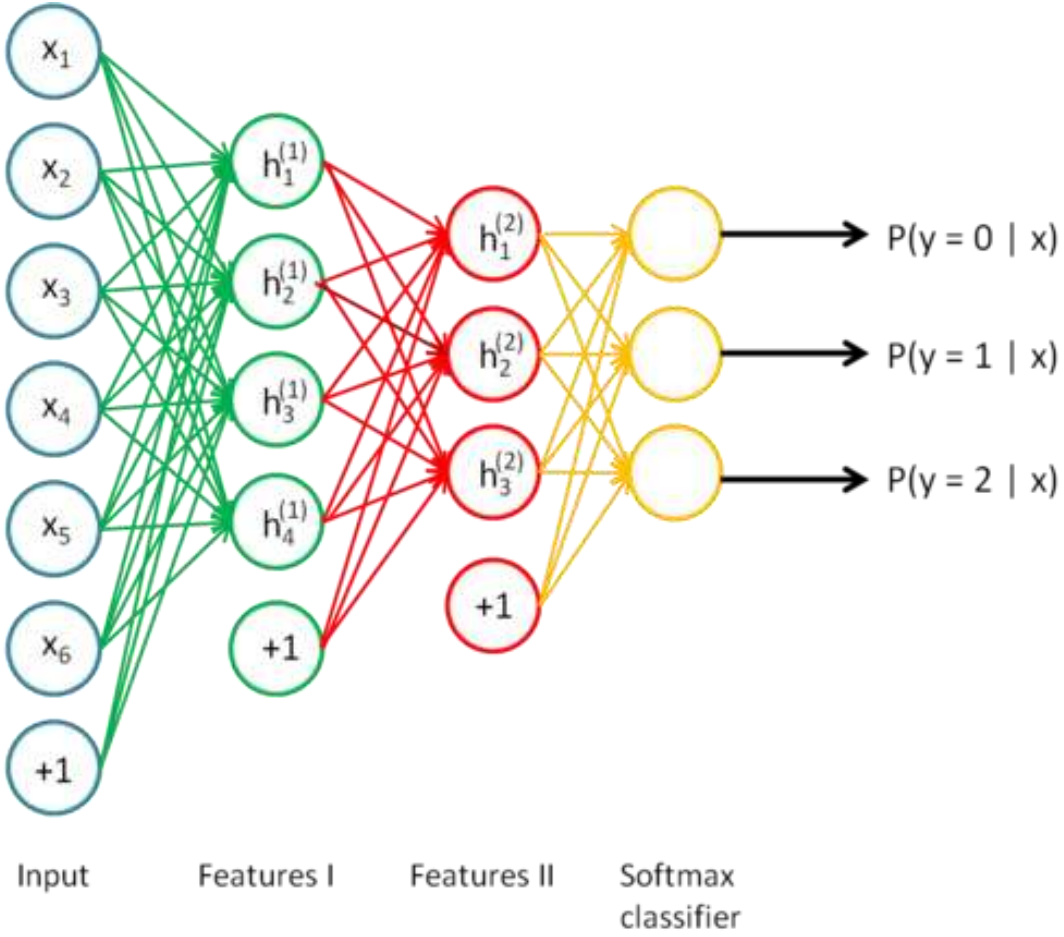
$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)T} x)} \begin{bmatrix} P(\theta^{(1)T}|x;\theta) \\ P(\theta^{(2)T}|x;\theta) \\ \vdots \\ P(\theta^{(K)T}|x;\theta) \end{bmatrix}$$

Cambia la función de costo:

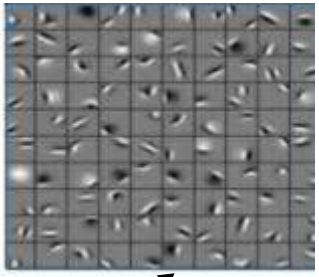
$$J(\theta) = -\underset{1\{y^{(i)}=k\}}{} \left[\sum_{i=1}^{m} \sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{\exp\left(\theta^{(k)T} x^{(i)}\right)}{\sum_{j=1}^{K} \exp(\theta^{(j)T} x^{(i)})} \right]$$

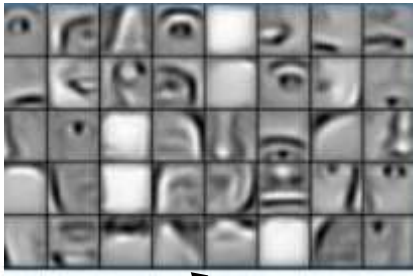
Donde el valor de $1\{y^{(i)} = k\}$ es igual a 1 si la condición entre {} se cumple y 0 en caso contrario.

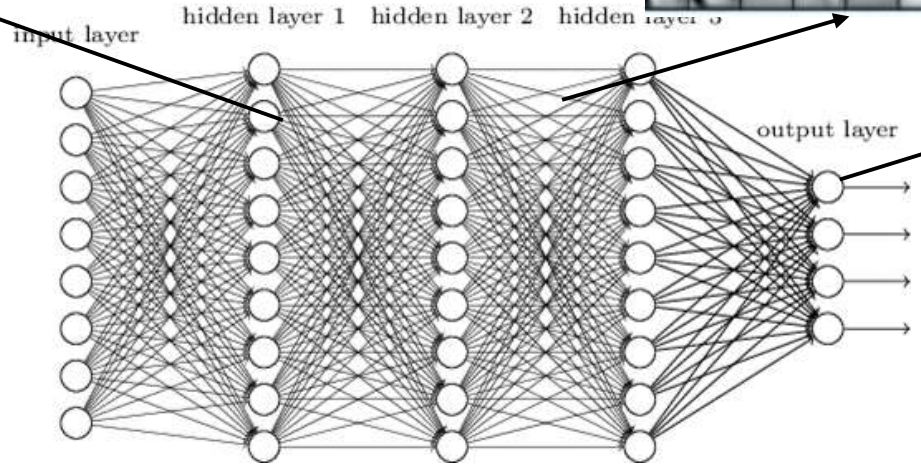
Redes neuronales multiclases



Redes neuronales profundas (deep learning)







Redes Neuronales

Dataset:

- Train/Test/Validation

- Ahora?
  - Muchísimos datos (>> 10.000.000 registros)

Asegurarse que test / validation vienen de la misma distribución

Redes neuronales: sesgo y varianza

Underfitting (high bias):

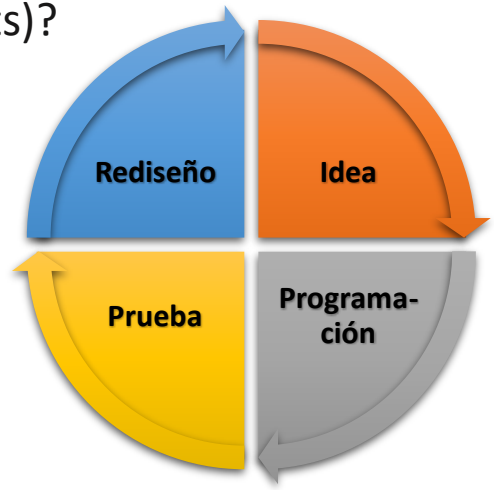
- Ampliar la red
- Cambiar la arquitectura de la red

Overfitting (high variance):

- Agregar más datos
- Regularización
- Cambiar la arquitectura de la red

¿Cómo se determinan …

- el número de capas ocultas (hidden layers)?
- el número de unidades (units)?
- qué función de activación usar?



Redes neuronales: regularización

- Penalización de pesos grandes
- Ej. Regresión Logística

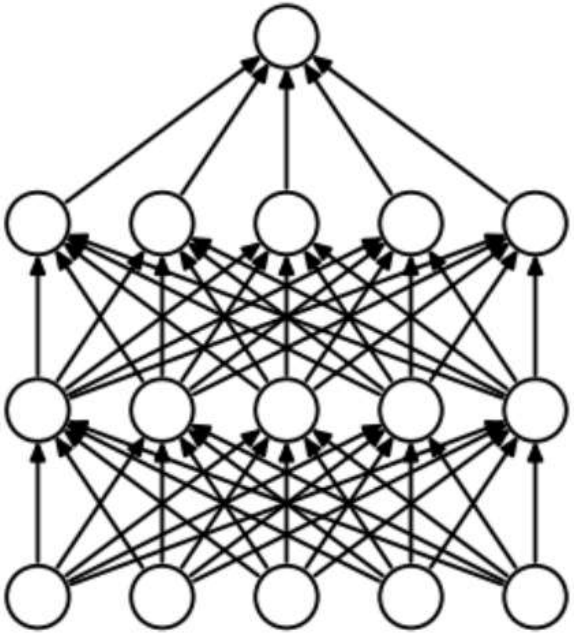
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

En la red neuronal:

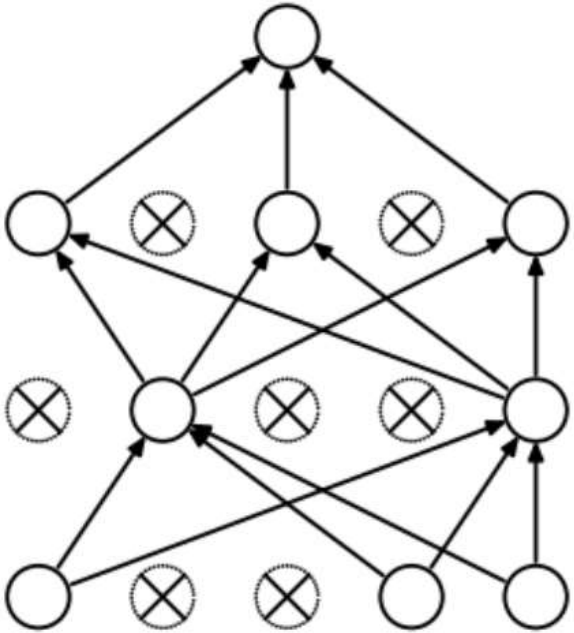
$$h_\Theta(x) \in \mathbb{R}^K \quad \left(h_\Theta(x)\right)_i = i^{th}\ salida$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right) \log\left(1 - h_\Theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\Theta_{ij}^{(l)}\right)^2$$

Redes Neuronales: Dropout



(a) Standard Neural Net



(b) After applying dropout.

Demo time (demo_6_neural_networks)

Naive Bayes

Naive Bayes

- •Es un clasificador basado en el teorema de Bayes, con una asunción "*inocente*" sobre los datos.
- Son "*naive*" porque se asume que las variables predictoras son independientes entre sí.
- Es muy sencillo de programar y entender.
- Sirve mucho como línea base de comparación, puede tener resultados que sobrepasan a algoritmos mucho más complejos.
- Es rápido de entrenar y funciona con datos de mucha dimensionalidad (e.g. es muy útil a la hora de clasificar documentos).

Teorema de Bayes

El algoritmo de "***Naive Bayes***" está fuertemente ligado al teorema de Bayes.

El Teorema de Bayes establece:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

El teorema establece que se puede encontrar la probabilidad de **A** (e.g. una clase objetivo) dada la ocurrencia de **B** (e.g. un conjunto de features). Es decir, **B** es la evidencia y **A** es la hipótesis.

Naive Bayes: Asunciones

1. Bajo el teorema de Bayes, la principal asunción es que los atributos son independientes entre sí. Esto significa que la presencia de una característica no afecta a las otras. Esta asunción es "***naive***", por eso el nombre del algoritmo.
2. Una segunda asunción, es que todos los atributos tienen el mismo efecto en la salida del algoritmo.

Naive Bayes: Utilizando el Teorema de Bayes

En base a lo establecido, se puede utilizar el teorema de Bayes para calcular la probabilidad de una **clase *y*** de la siguiente manera:

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

Donde y representa la clase y X representa el vector de atributos:

$$X = (x_1, x_2, \dots, x_n)$$

Naive Bayes: Utilizando el Teorema de Bayes

Utilizando sustitución en la fórmula anterior obtenemos:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)\, P(x_2|y)\ \dots P(x_n|y)\, P(y)}{P(x_1)\, P(x_2)\ \dots P(x_n)}$$

Dado que el denominador es estático para todas las entradas del conjunto de datos se puede ignorar y se establece una proporcionalidad:

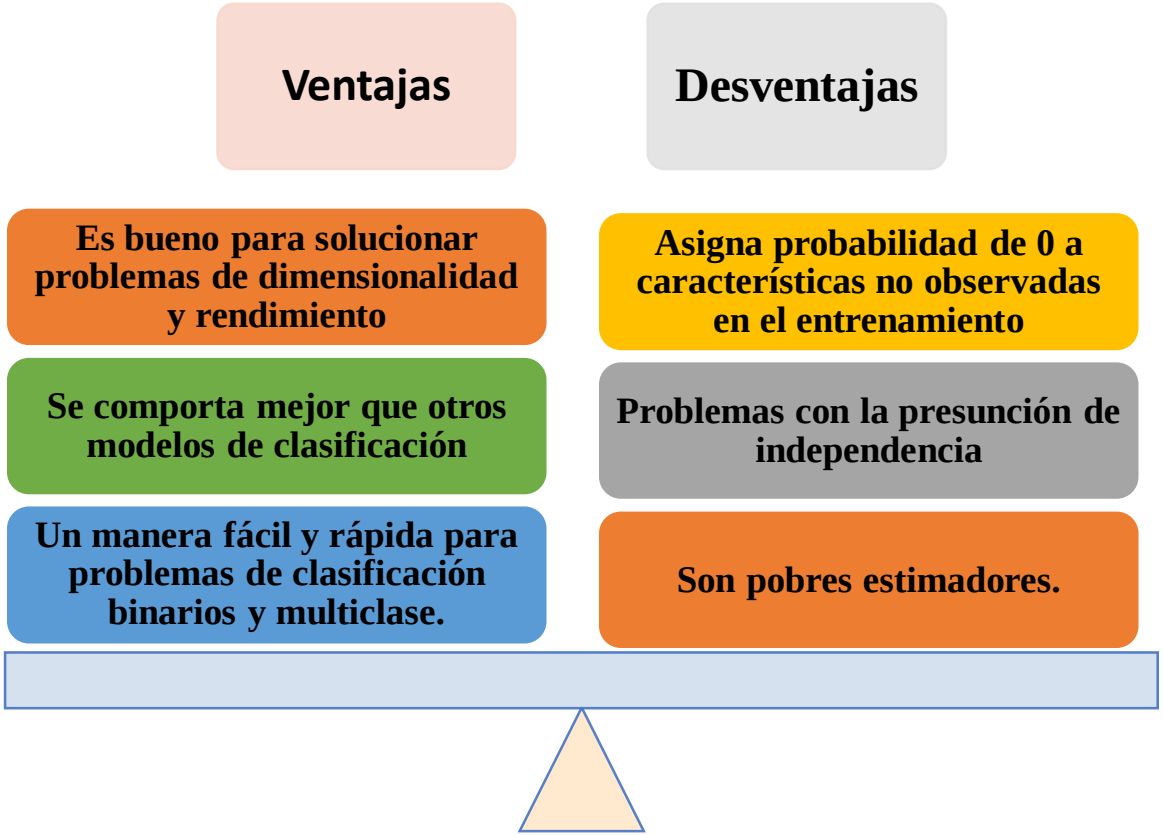
$$P(y|x_1, \dots, x_x)\ \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

En base a lo visto previamente, la predicción de la clase objetivo es sencillamente aquella con mayor probabilidad: $\boldsymbol{y = argmax_y\, P(y)\ \prod_{i=1}^{n} P(x_i|y)}$

Naive Bayes: Algoritmo

- Convertir el conjunto de datos en tablas de frecuencias.
- Crear una tabla de probabilidad calculando las correspondientes de cada evento.
- Calcular la probabilidad para cada clase
$$\boldsymbol{P(y) / \forall\, y \in Y}$$
- Calcular la probabilidad condicional de cada atributo dada cada clase:
$$\boldsymbol{P(x_i|y) / 0 \leq i \leq n\,, \forall\, y \in Y}$$
- Calcular la clase de acuerdo a la que maximice la probabilidad (o, en este caso la proporción): $\boldsymbol{y = argmax_y\, P(y)\ \prod_{i=1}^{n} P(x_i|y)}$

Puntos fuertes y débiles de Naive Bayes



Naive Bayes: Tipos de algoritmos

Bernoulli Naive Bayes: Para casos donde los atributos son variables binarias (e.g. si una palabra ocurre o no en un documento).

Multinomial Naive Bayes: Para casos donde los atributos representan frecuencias (e.g. la cantidad de veces que una palabra ocurre en un documento).

Gaussian Naive Bayes: Para casos donde los atributos toman valores continuos, se asume que los valores son muestras de una distribución gaussiana (esto se usa para calcular las probabilidades condicionales en el algoritmo).

Naive Bayes: Relación con Regresión Logística

Naive Bayes es un modelo **generativo**, es decir, que intenta modelar la probabilidad $p(y|\boldsymbol{x})$ donde "y" representa la etiqueta y "$\boldsymbol{x}$" representa los atributos. A grandes rasgos, trata de generar los atributos del modelo dada la etiqueta.

La **Regresión Logística** es un modelo **discriminativo**, es decir, que intenta modelar la probabilidad $p(y|\boldsymbol{x})$. A grandes rasgos, determina la etiqueta "y" a partir del vector de atributos "$\boldsymbol{x}$". Esto quiere decir que no le hace falta hacer ninguna asunción sobre $p(\boldsymbol{x})$ ya que no es necesaria para modelar.

Demo Time (demo_7_naive_bayes)

Dudas … Consultas