APRENDIZAJE SUPERVISADO

Docentes:

- **Dra. Ing. Karim Nemer**
- **Dr. Lic. José Robledo**

TERCERA CLASE

TERCERA CLASE: 01/07/2022

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

BOOSTING

- Método para hacer aprendizaje por "ensemble".
- En 1989 Vilian y Kerns plantean un algoritmo al que estimulan “boost” mediante el análisis de los errores cometidos.
- Combina un conjunto de clasificadores débiles para obtener un clasificador más poderoso.
- Los datos se siguen eligiendo en forma aleatoria.
- Es un método iterativo.
- Una vez obtenidos todos los resultados, se establece una jerarquía basada en el nivel de error, teniendo más peso el resultado del árbol que cometió el menor error.

BOOSTING

Algoritmo de Boosting

- Se definen las entradas, determinando cada una de las variables implicadas:

$$(x_i, y_i)$$

- Donde x_i son el conjunto de datos del árbol *i* y y_i su clasificación, tal que

$$y_i \in Y/Y = \{-1,1\} \approx \{\text{Error}, \text{Acierto}\}$$

- Se inicializa de forma uniforme, de tal manera que:

$$D_i = \frac{1}{m} \; \forall \; i = 1, \dots, m$$

- Durante la primera iteración se entrena el clasificador "débil", utilizando el conjunto de pesos D_i.

- Se obtiene la primera hipótesis $h_t : Y$. Se pretende obtener un error bajo de predicción

$$\varepsilon_t = Pr_{i\,D_i}[h_t(y) \neq y_i]$$

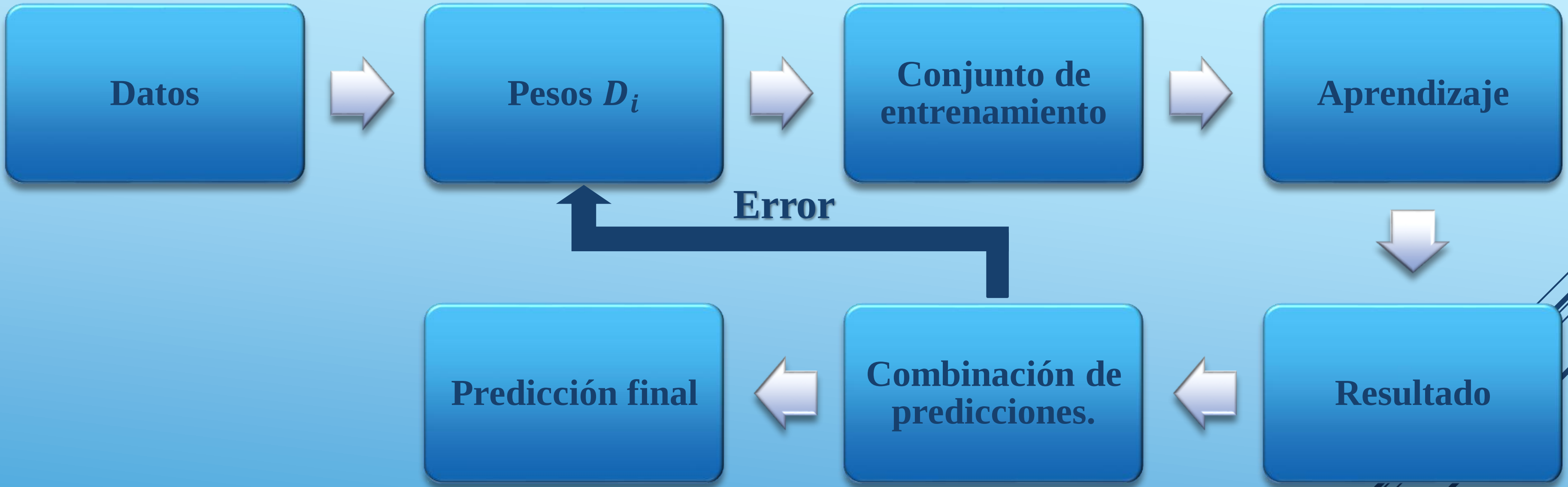
- Con los errores obtenidos se actualizan los D_i según lo siguiente:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \qquad D_{t+1}(i) = \frac{D_t(i)\exp\left(\alpha_t\, x_i\, h_t(y_i)\right)}{Z_t}$$

- Donde Z_t es un factor de normalización.

- La hipótesis final quedaría como:

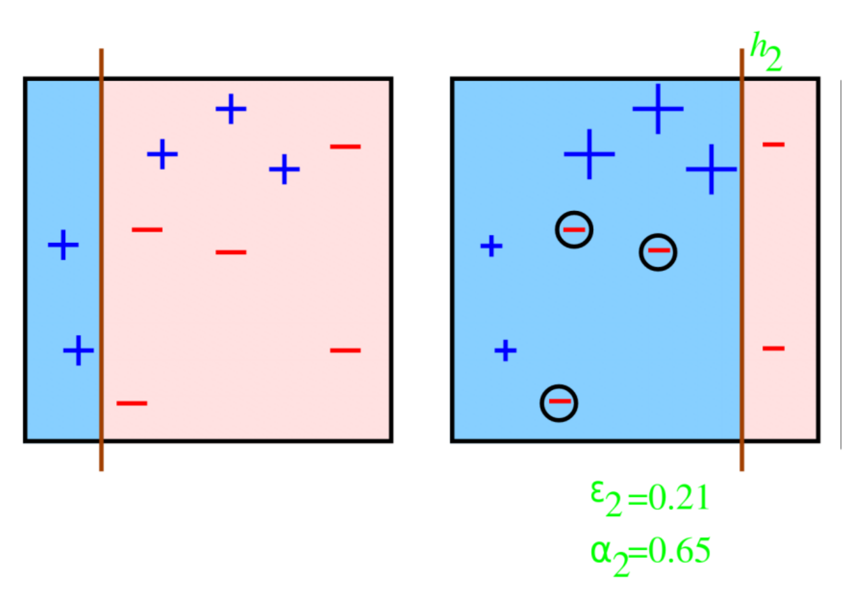
$$H(y) = signo\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$


Datos
Pesos D_i
Conjunto de entrenamiento
Aprendizaje
Error
Resultado
Combinación de predicciones.
Predicción final

Boosting

Tiene sus variantes, pero en forma general el Boosting tiene los siguientes pasos:

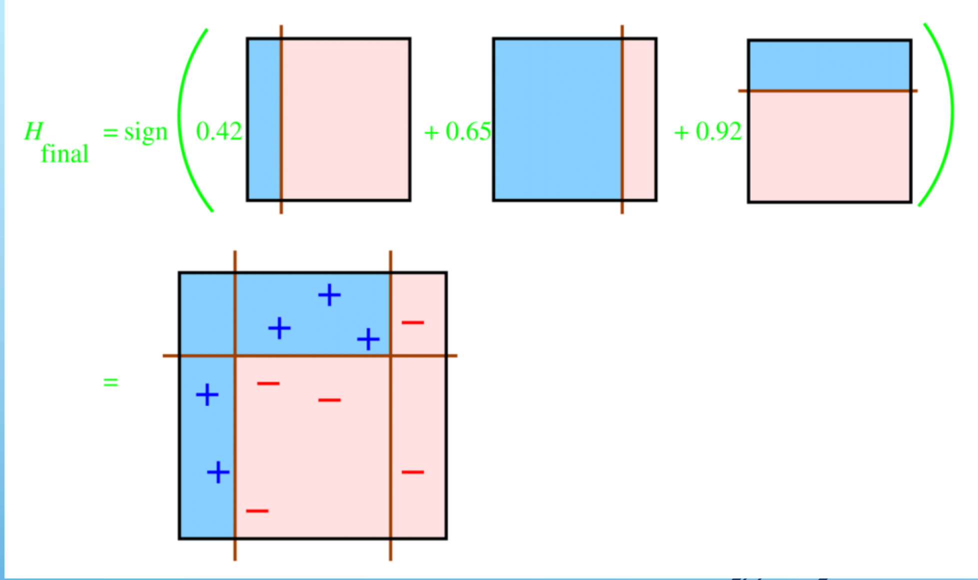
1. Se entrenan varios árboles de decisión con conjuntos aleatorios de datos. De forma individual.
2. Se le asigna la misma ponderación a cada set de datos de entrenamiento y se aplica al primer modelo de ML, llamado "*Modelo Base*".
3. El modelo hace predicciones a cada conjunto de datos.
4. El algoritmo de *Boosting* evalúa las predicciones y aumenta la ponderación de las muestras que presenten un error más significativo



Boosting: Ejemplo (Parte 1)

Boosting: Ejemplo (Parte 2)

5. También asigna una ponderación basada en el rendimiento del modelo
6. Un modelo con pocos o ningún error tendrá una ponderación mucho más alta que los demás
7. El algoritmo pasa los datos ponderados al siguiente árbol de decisión
8. El algoritmo repite los pasos del 4 al 7 hasta que el error de entrenamiento esté por debajo de lo aceptado



Source: http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Los principales tipos de Boosting son

- ***Boosting Adaptativo***
  - Fue uno de los primeros desarrollados
  - Se adapta y autocorrige en cada iteración
  - Es menos sensible que otros tipos de Boosting.
  - No funciona bien cuando existe correlación entre las features
  - Apto para problemas de clasificación

Tipos de boosting: Boosting Adaptativo

Tipos de boosting: Boosting por Gradiente

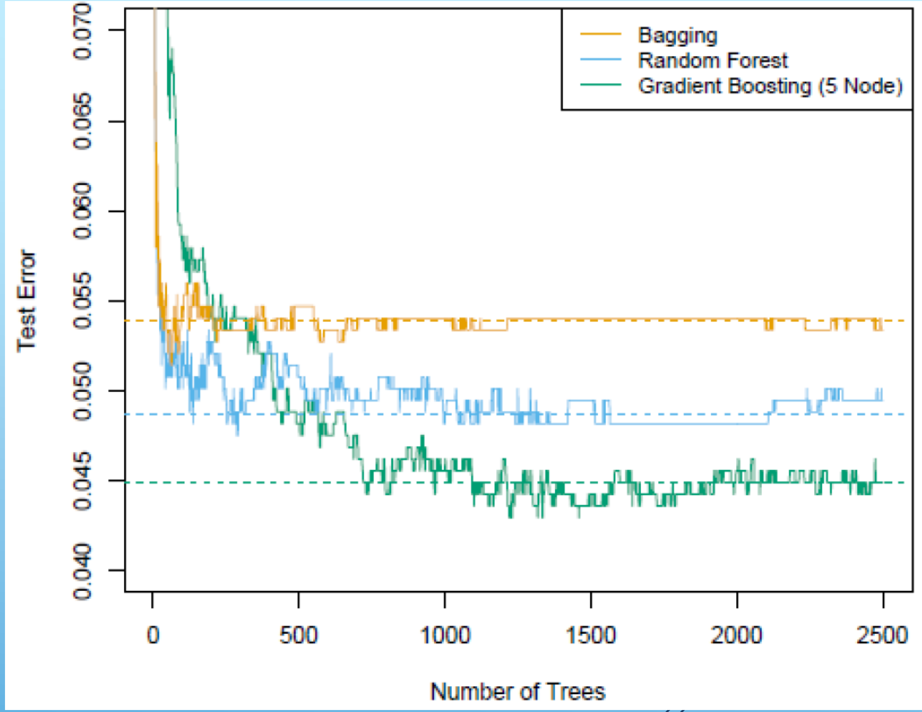
- ***Boosting por gradiente***
  - Similar al adaptativo.
  - No asigna ponderación a los que clasifican de forma incorrecta
  - Se utiliza una función de pérdida para que el nuevo paso sea más eficiente que el anterior
  - Presenta soluciones más efectivas que el anterior
  - Sirve para problemas de clasificación y regresión

- ***Boosting por Gradiente Extremo***
  - Similar al por gradiente.
  - Incrementa la velocidad computacional del anterior.
  - Se realiza procesamiento en paralelo.
  - Se puede utilizar para manejar grandes conjuntos de datos.
  - Sus características clave son la paralelización, la computación distribuida, la optimización de la memoria caché y el procesamiento fuera del núcleo.
  - Sirve para problemas de clasificación y regresión

Tipos de boosting: Boosting por Gradiente Extremo

Ventajas y desventajas del Boosting

Ventajas:	**Desventajas**
Poca variabilidad	No estudia todas las características.
Eficacia computacional	Se necesitan muchos árboles
Sesgo reducido	Implementación en tiempo real
Fácil de implementación	Vulnerabilidad a datos atípicos



Demo time (demo_9_boosting)

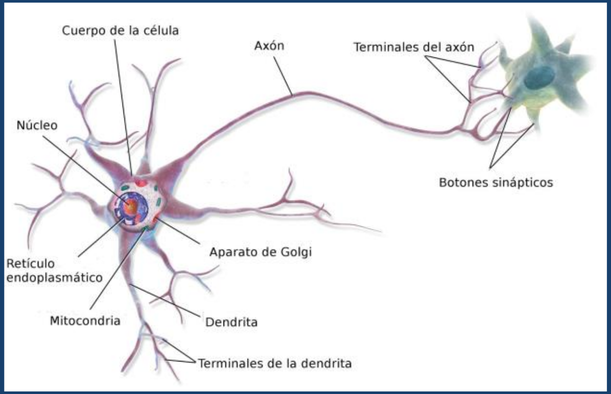
VOTING

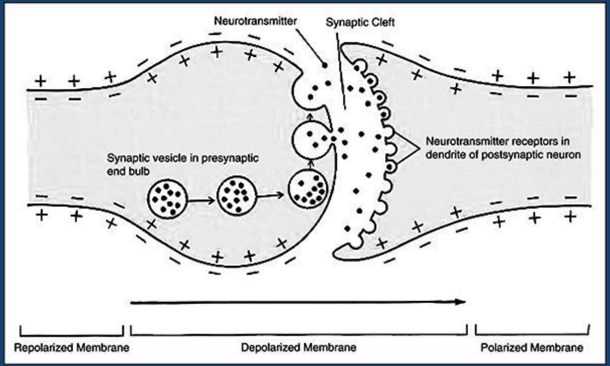
VOTING

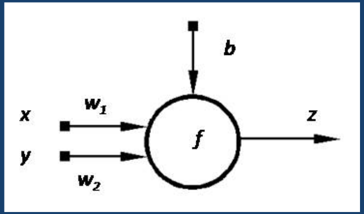
- Clasificador compuesto de varios clasificadores.
- Cada clasificador que lo compone se entrena sobre el conjunto de datos.
- El clasificador de “voting” final simplemente elige la clase que tuvo “más votos” de parte de los clasificadores que lo componen.
- La votación puede ser “hard” (simplemente se cuenta la cantidad de votos para una clase) o “soft” (se usa la probabilidad).

Redes Neuronales Artificiales (Introducción)

Redes Neuronales Artificiales







Es un sistema que:

- Pertenece al Machine Learning
- Aprende mediante la experiencia
- Mediante la evaluación del error obtenido, se adapta para minimizarlo
- Permite resolver problemas complejos
- Aprende a través de la aproximación de funciones no lineales con muchas incógnitas

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:

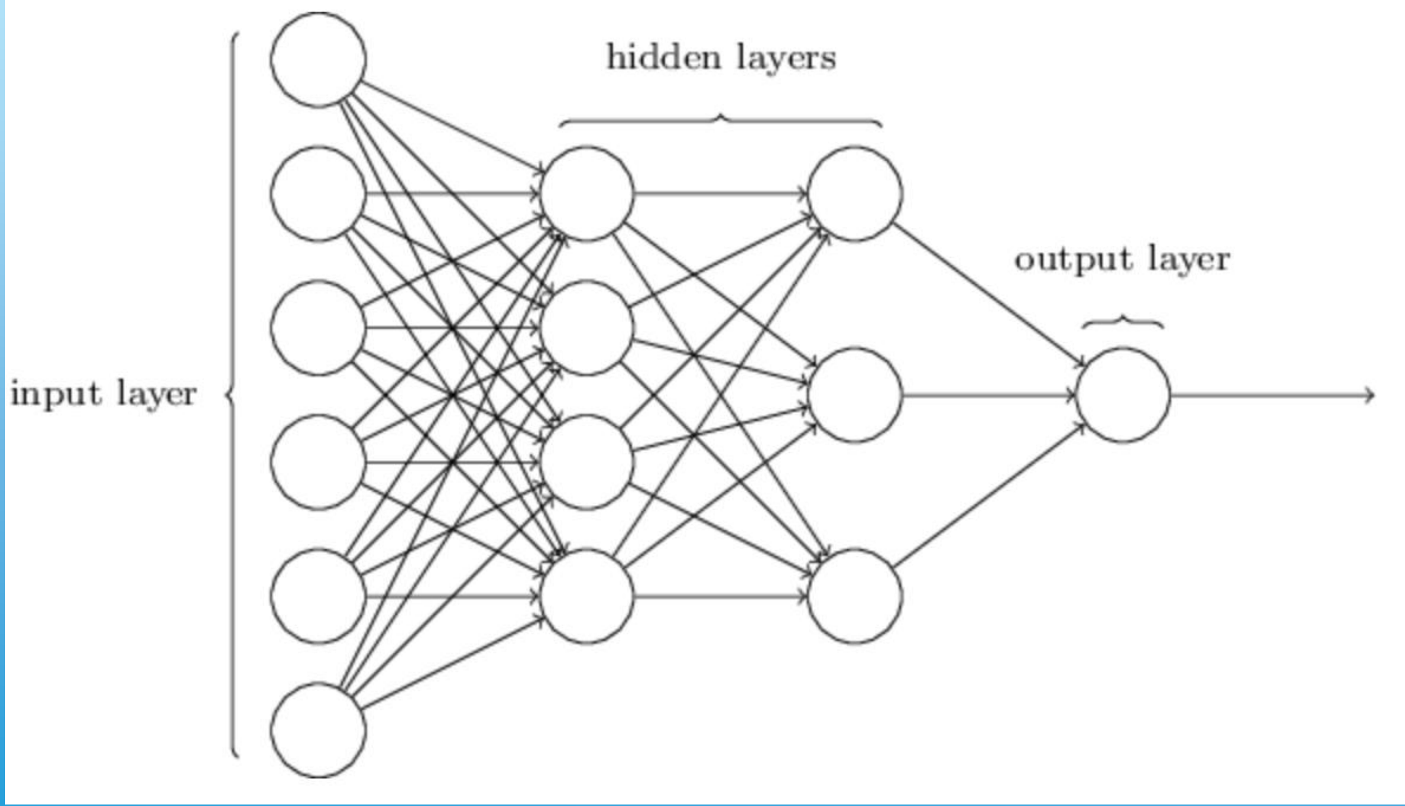
$$\mathcal{J}(\omega, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Para minimizarla, usamos descenso por gradientes. Necesitaremos:

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

Repaso de la regresión logística: Coste

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):

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

Regresión softmax (múltiples clases)

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

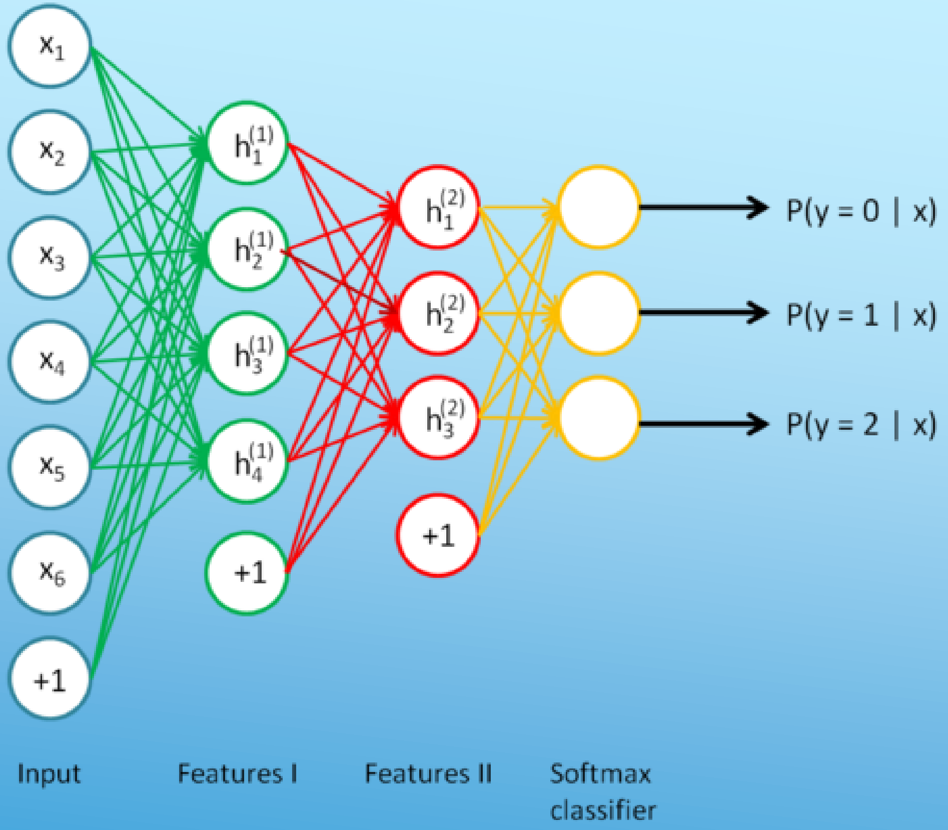
Modificamos nuestra hipótesis:

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

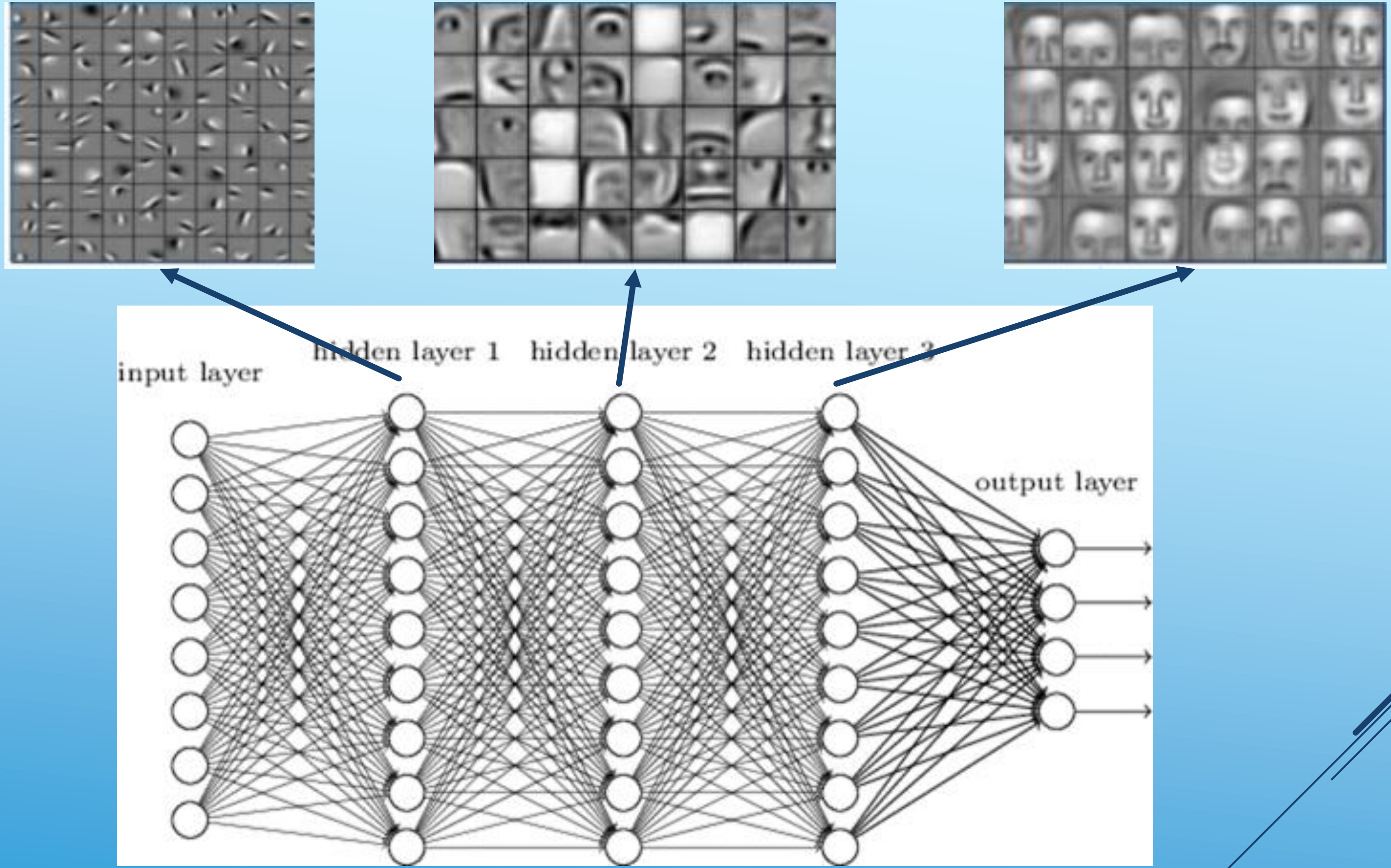
Cambia la función de costo:

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

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):

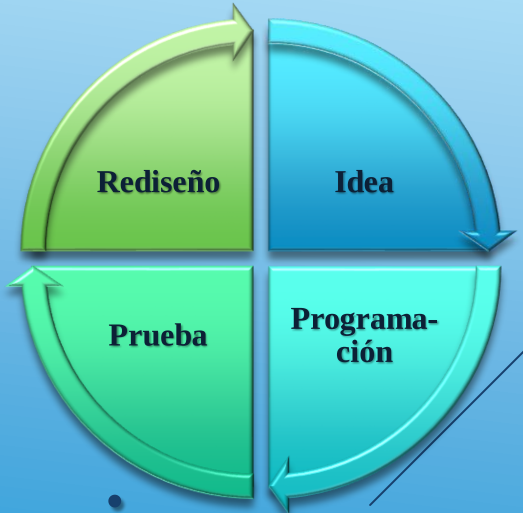
- Ampliar la red
- Cambiar la arquitectura de la red

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?



- Penalización de pesos grandes
- Ej. Regresión Logística

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

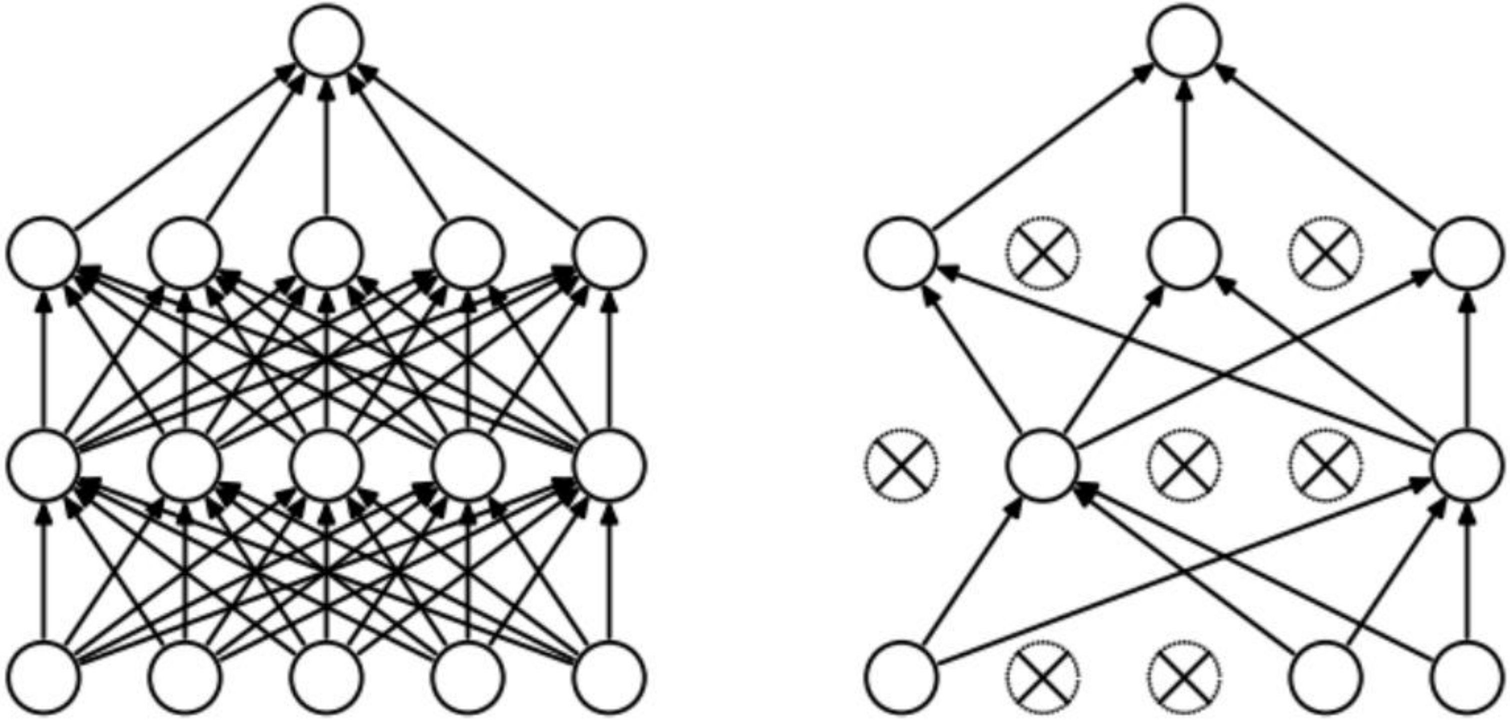
En la red neuronal:

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

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

Redes neuronales: regularización

Redes Neuronales: Dropout



(a) Standard Neural Net

(b) After applying dropout.

Demo time (demo_10_neural_networks)

FIN DE LA TERCERA CLASE