



Aprendizaje Supervisado


Cristian Cardellino




Tercera Clase

Temario de la Clase

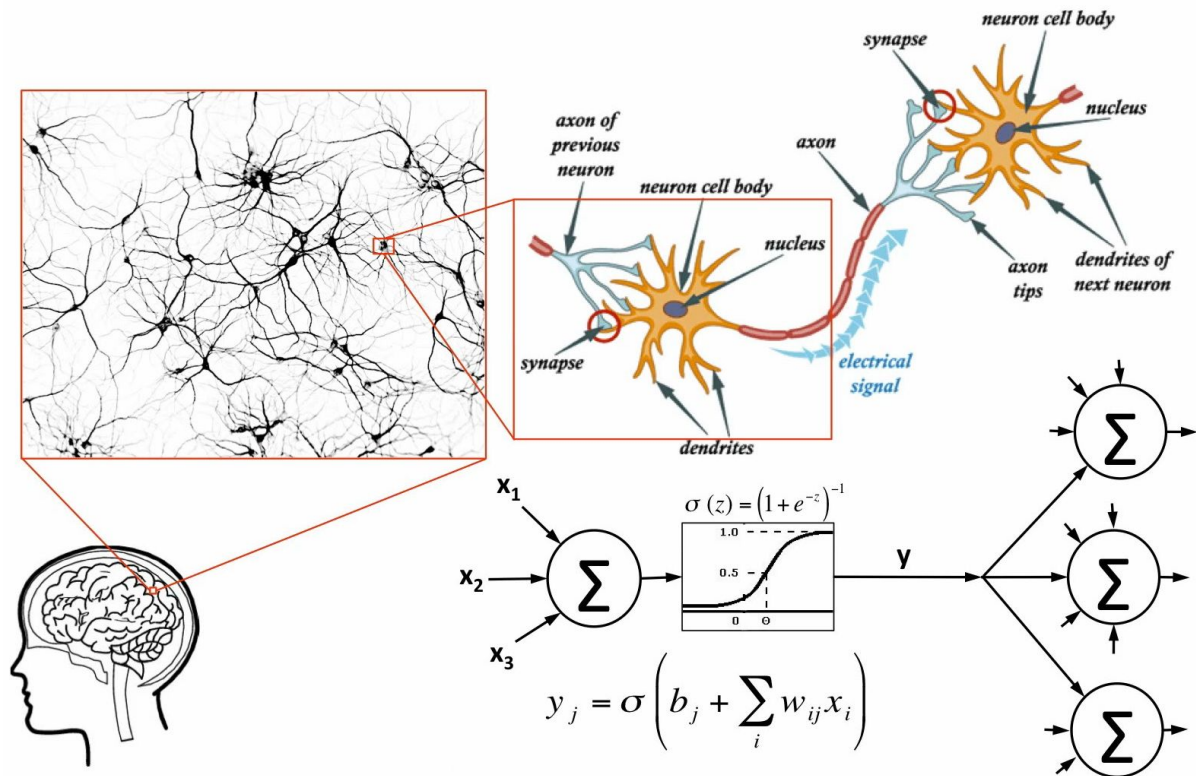
- ¿Qué es aprendizaje supervisado?
- Aprendizaje supervisado.
 - Repaso: Regresión Lineal y Polinomial, Regresión Logística, Naive Bayes.
- Support Vector Machines.
 - Repaso: Perceptrón.
 - SVC/SVR. Datos no linealmente separables. Función de costo.
- Ensemble learning.
 - Repaso: Decision Trees
 - Random Forest, Bagging, Boosting, Voting.
- Redes neuronales.
 - Perceptrón multicapa.
- Sistemas de recomendación.
 - Filtrado colaborativo.
- Prácticas de reproducibilidad



Redes Neuronales (Introducción)



Por qué redes "neuronales"?



Repaso de la Regresión Logística

- Dado x , el objetivo es encontrar: $\hat{y} = P(y = 1|x)$
- Cuál es la forma más sencilla de transformar un vector x ?

Repaso de la Regresión Logística

- Dado x , el objetivo es encontrar: $\hat{y} = P(y = 1|x)$
- Cuál es la forma más sencilla de transformar un vector x ?

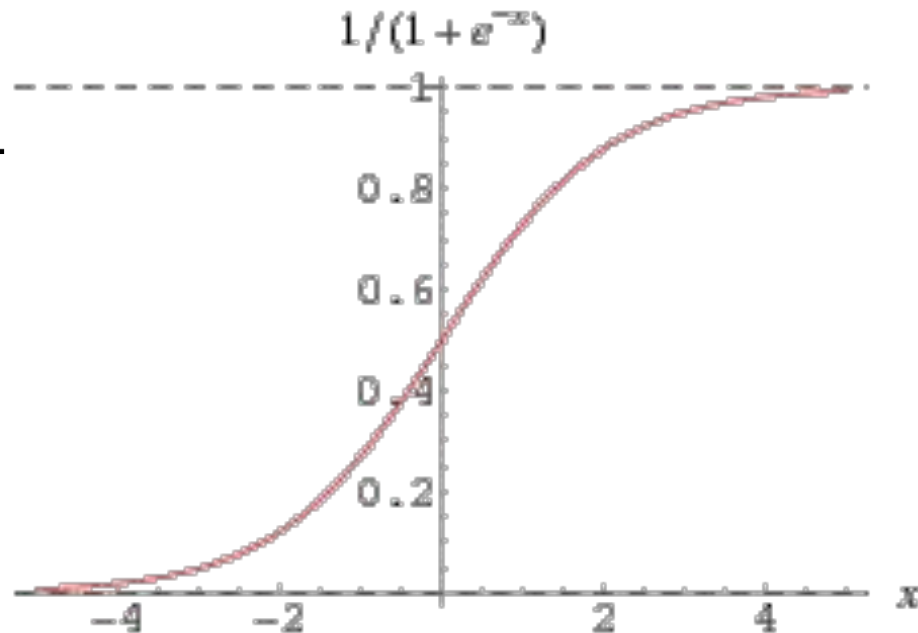
$$\hat{y} = w^T x + b$$

- Pero nos gustaría que \hat{y} fuese una probabilidad: $0 \leq \hat{y} \leq 1$

Repaso de la Regresión Logística: Sigmoid

Función de Sigmoid

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



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):

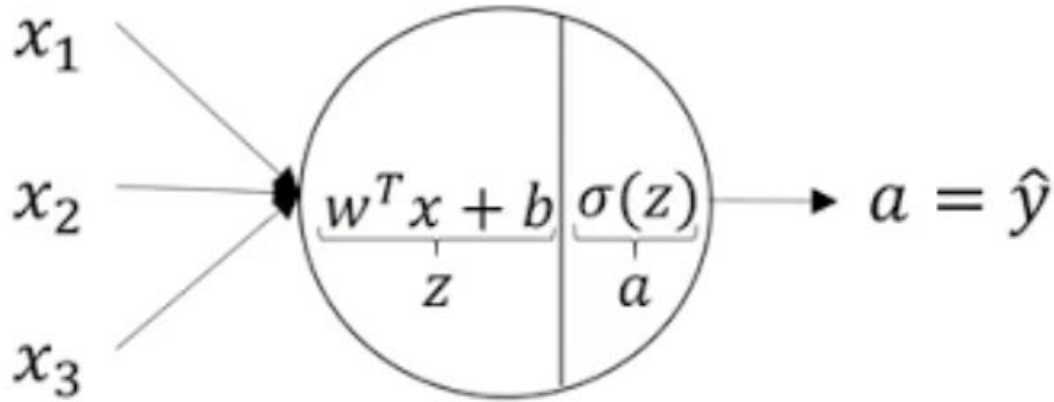
$$\mathcal{L}(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

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

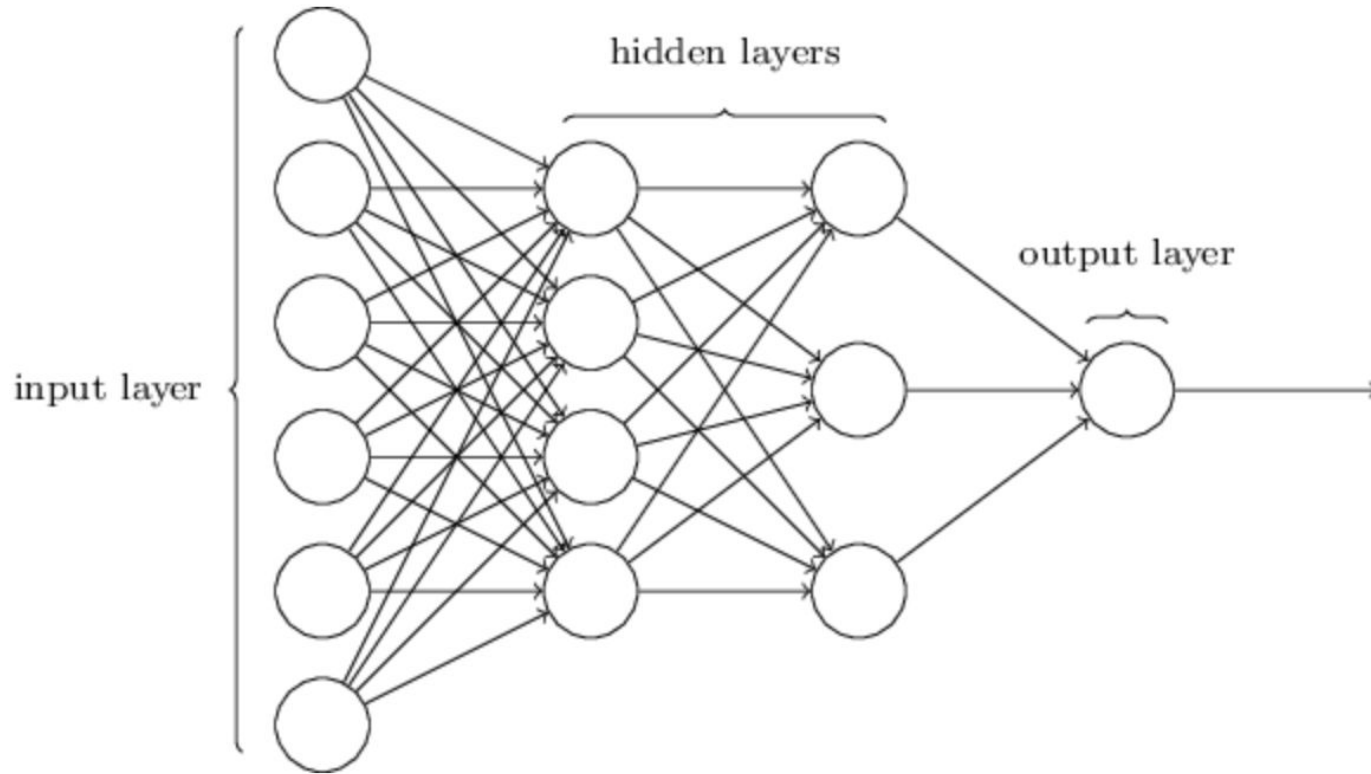
Para minimizarla, usamos descenso por gradientes. Necesitaremos:

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

La Regresión Logística como una neurona



Redes Neuronales



Redes Neuronales

Funciones de Activación:

- Sigmoid (como en la regresión logística)
- tanh: $\frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$
- Rectified Linear Unit (ReLU): $\max(0, x)$

Demo Time

(demo_10_neural_networks)

Softmax Regression (Multiple Classes)

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

Modificamos nuestra hipótesis:

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

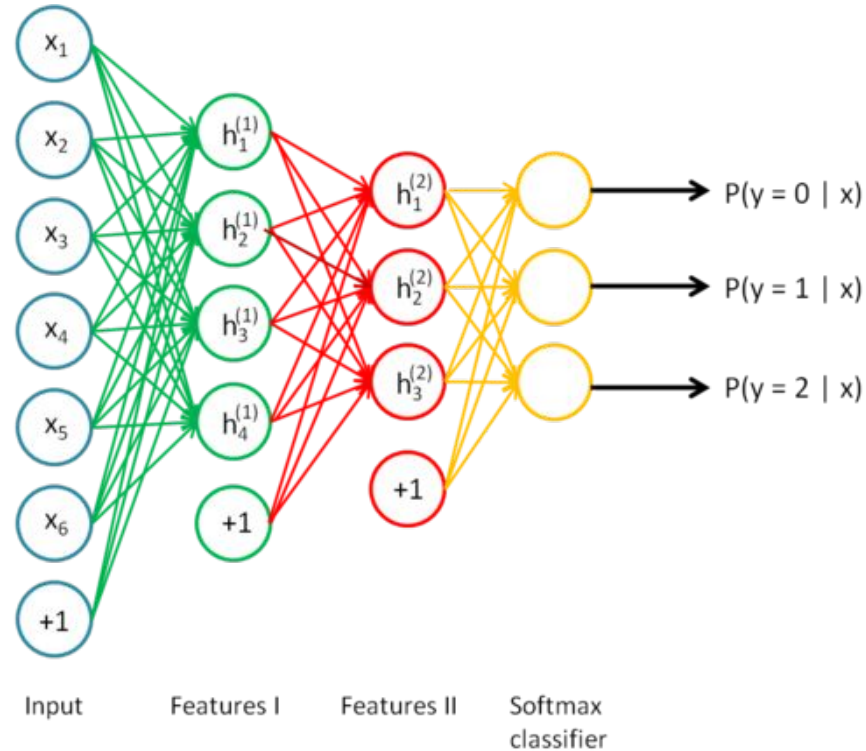
Softmax Regression: Cost Function

Cambia la función de costo:

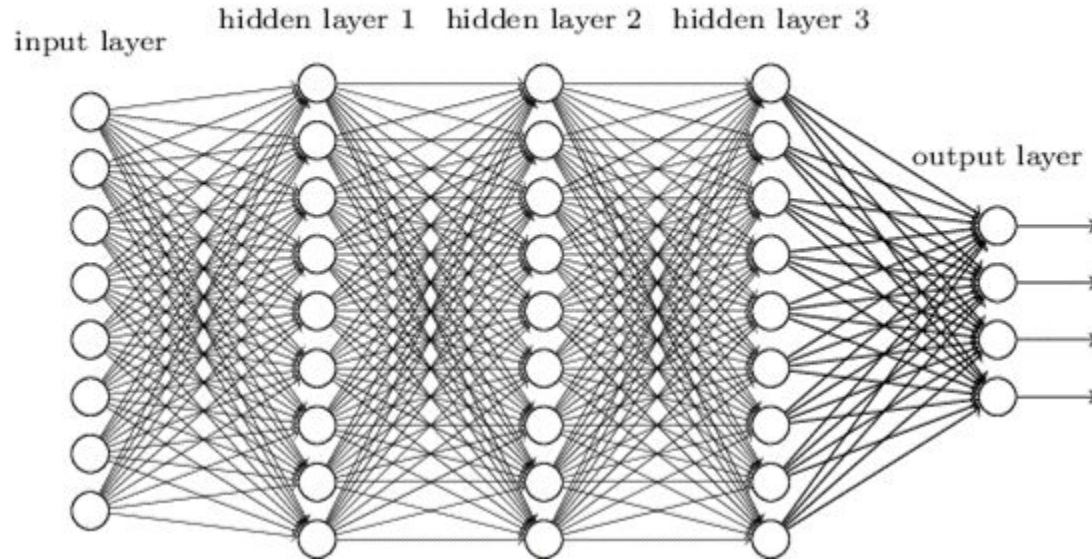
$$J(\theta) = - \left[\sum_{i=1}^m \sum_{k=1}^K \mathbf{1} \left\{ y^{(i)} = k \right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^K \exp(\theta^{(j)\top} x^{(i)})} \right]$$

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

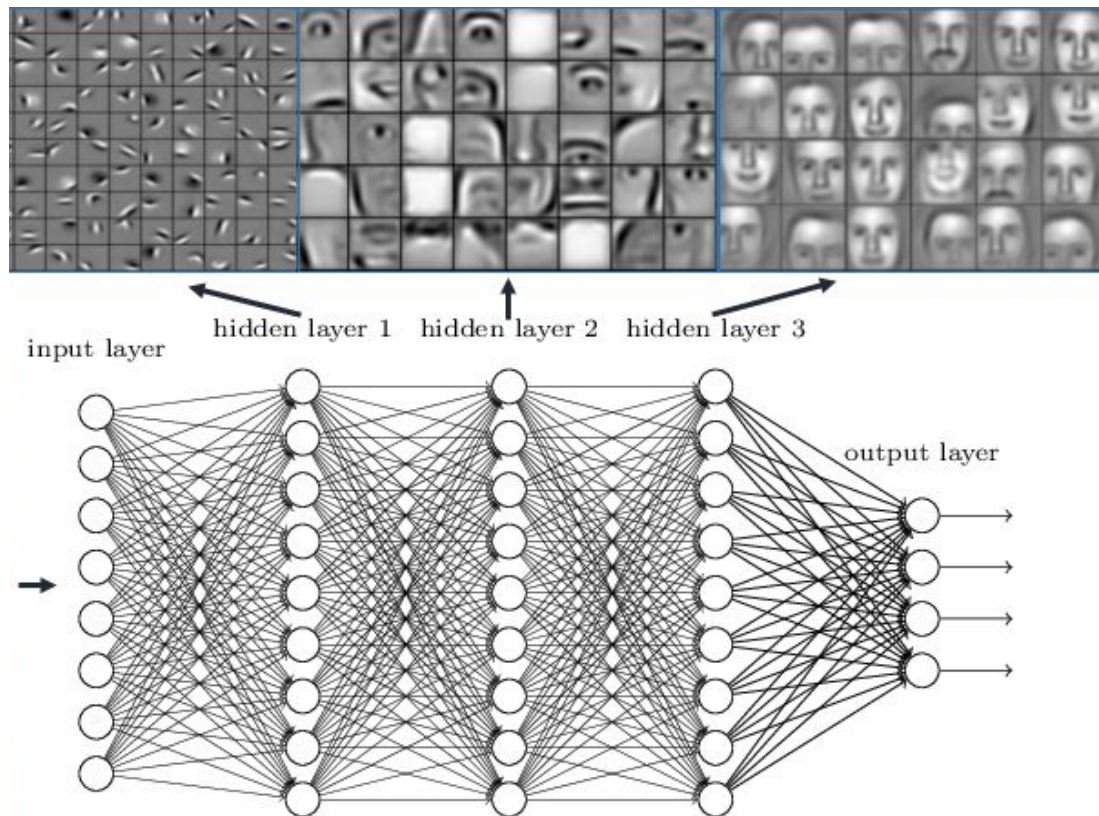
Multiclass Neural Networks



Redes Neuronales Profundas (Deep Learning)



Redes Neuronales Profundas



Redes Neuronales

Dataset:

- Train/Test/Validation



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

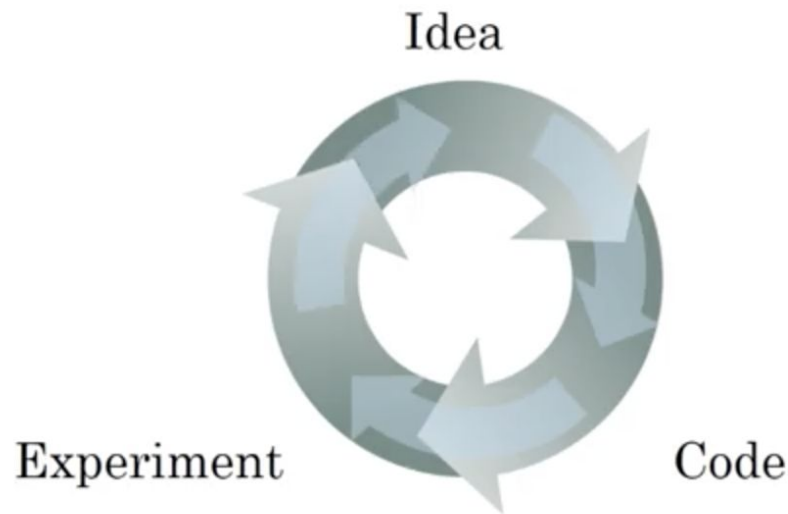
Overfitting (high variance):

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

Redes Neuronales

Cómo se determinan?

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



Redes Neuronales: Regularización

Logistic regression:

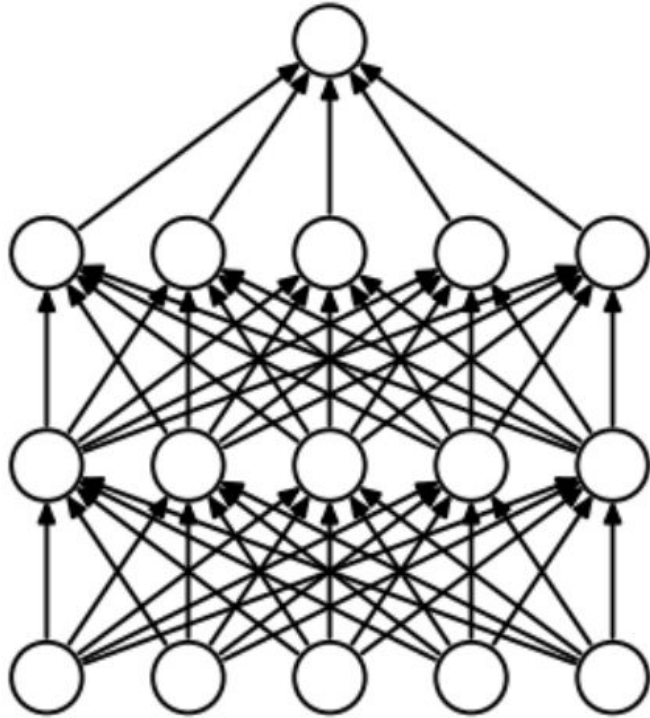
$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Neural network:

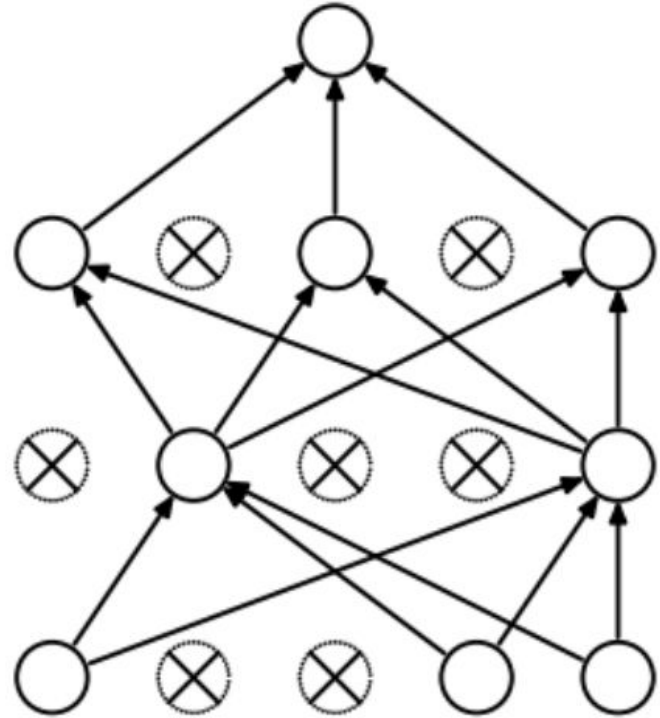
$$h_{\Theta}(x) \in \mathbb{R}^K \quad (h_{\Theta}(x))_i = i^{th} \text{ output}$$

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

Redes Neuronales: Dropout



(a) Standard Neural Net



(b) After applying dropout.

Sistemas de Recomendación (Introducción)

Sistemas de Recomendación: ejemplos

albert einstein



All



Images



Videos



News



Books

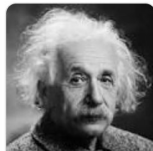


More

Settings

Tools

Related to Albert Einstein and Isaac Newton



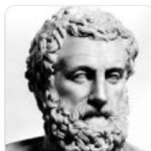
Albert Einstein



Galileo Galilei



Leonardo da Vinci



Aristotle



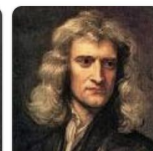
Thomas Edison



Charles Darwin



Nikola Tesla



Isaac Newton

Basado en tu última visita [Ver historial](#)



\$ 609⁹⁹

6x \$ 101,67 sin interés



\$ 459⁹⁹



\$ 119⁹⁸

6x \$ 20 sin interés



\$ 549



\$ 249⁹⁹

Sistemas de Recomendación: objetivos

Los sistemas de recomendación se centran en:

- *items*, para eCommerce
- *contenido*, para eLearning, noticias
- *links*, para navegación

en general, se habla de ítems como término genérico.

El objetivo general de estos sistemas es el de guiar a los usuarios a tomar decisiones.

Tipos de Sistemas de Recomendación

- **Basados en contenido:** tratan el problema de manera específica para cada usuario y "aprenden" una clasificación de lo que a un usuario le gusta basado en las características (contenido) de un ítem.
- **Filtros colaborativos:** se basan en la idea que los usuarios que coinciden en el pasado lo harán en el futuro y que les gustarán ítems similares.
- **Basados en el conocimiento:** el usuario proporciona explícitamente parámetros del producto que quiere
- **Basados en cuestiones demográficas**
- **Sistemas híbridos**

Filtros colaborativos

Combina usuarios con intereses similares

- Se podría requerir de muchos usuarios para que las posibilidades de encontrar un par con intereses en común.
- Tiene que existir un método sencillo para reflejar los intereses de los usuarios.
- Se necesita de un algoritmo eficiente para combinar a los usuarios con gustos similares.

Filtros colaborativos: cómo funcionan?

INPUT

Usuarios generan puntuaciones a un conjunto de ítems (implícita o explícitamente): matriz de puntuaciones usuario-ítem

OUTPUT

- Predicción (numérica) indicando cuánto a un usuario le gusta un ítem
- Una lista con N ítems recomendados por usuario

Filtros colaborativos: Vecindarios basado en usuarios

Dado un usuario (Alice) y un ítem que todavía no se ha asignado un puntaje, se debe:

- encontrar el conjunto de usuarios que más se parecen a Alice (usuarios a los que les gustan ítems similares) y han puntuado el objeto
- usar sus puntuaciones para predecir si a Alice le gustará el ítem
- aplicar este proceso sobre todos los objetos que Alice no ha puntuado y recomendar los que tienen mayor puntuación

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Similitud entre usuarios

Coeficiente de correlación (Pearson correlation)

$$\text{sim}(a, b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}}$$

donde:

- a y b son usuarios;
- $r_{a,p}$ es la puntuación del usuario a al ítem p ;
- y P es el conjunto de ítems que ya han sido puntuados por a y b .

Similitud entre usuarios

$$\text{sim}(a, b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}}$$

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

$\text{sim}(\text{Alice}, \text{User1}) = 0.85$

$\text{sim}(\text{Alice}, \text{User2}) = 0.70$

$\text{sim}(\text{Alice}, \text{User3}) = 0$

$\text{sim}(\text{Alice}, \text{User4}) = -0.79$

Similitud entre usuarios

Coeficiente de correlación

$$\text{pred}(a, p) = \bar{r}_a + \frac{\sum_{b \in N} \text{sim}(a, b)(r_{b,p} - \bar{r}_b)}{\sum_{b \in N} \text{sim}(a, b)}$$

donde:

- N es el conjunto de "vecinos"
- p es un ítem

$$\text{pred}(\text{Alice}, \text{item5}) = 4 + \frac{0.85(3-2.4)+0.7(5-3.8)}{0.85+0.7}$$

Filtros colaborativos: Vecindarios basado en ítems

Se usan las semejanzas entre los ítems (y no entre los usuarios) para hacer las predicciones. Por ejemplo, busquemos los ítems parecidos al *Item5*. Ahora usamos las puntuaciones que Alice le asignó a tales ítems para darle un valor al *Item5*.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Similitud entre ítems

Distancia (ajustada) del coseno

$$\text{sim}(a, b) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

donde:

- a y b son ítems;
- $r_{u,a}$ es la puntuación del usuario u al ítem a ;
- y U es el conjunto de usuarios que han puntuado a los ítems a y b .

Similitud entre ítems

$$\text{pred}(u, item_j) = \bar{r}_u + \frac{\sum_{item_i \in N} \text{sim}(item_i, item_j) r_{u, item_i}}{\sum_{item_i \in N} \text{sim}(item_i, item_j)}$$

donde:

- N es el conjunto de "vecinos"
- u es un usuario

$$\text{pred}(\text{Alice}, item_5) = \bar{r}_{\text{Alice}} + \frac{\sum_{item_i \in N} \text{sim}(item_i, item_5) r_{\text{Alice}, item_i}}{\sum_{item_i \in N} \text{sim}(item_i, item_5)}$$

Filtros colaborativos: Preprocesamiento

Para que estos sistemas de recomendación sean escalables, es necesario aprender el **modelo "offline"**.

- Calcular offline todos las similitudes entre pares de objetos
- Calcular la predicción en tiempo real (pred de la slide anterior)
- El vecindario (N) suele ser bastante pequeño (el usuario puntúa pocos objetos)

Este preprocesamiento funciona en CF por ítems (y no en CF por usuarios): las similitudes entre ítems suelen ser más estables que entre usuarios.

Filtros colaborativos: Preprocesamiento

Requerimientos de memoria: si consideramos n ítems, tendremos n^2 similitudes.

En la práctica, la matriz es dispersa (muchos pares de ítems que no tienen similitud).

Se suelen usar reducciones fijando un umbral mínimo de "co-ratings": se eliminan ítems que tienen pocas puntuaciones comunes, usando al menos n' usuarios.

Se puede también limitar el tamaño del vecindario (N).

Filtros colaborativos: Problemas

Cold start problem:

- ¿Cómo recomendamos nuevos ítems?
- ¿Qué le recomendamos a usuarios nuevos?

Soluciones inmediatas:

- Forzar a los usuarios a puntuar un conjunto de objetos (-1)
- Emplear otro método para la estos casos (basado en contenido, información demográfica o recomendaciones no personalizadas)

Filtros colaborativos: Problemas

Problemas con los vecindarios:

El conjunto de usuarios ítems similares puede ser muy pequeño. Lo que produce malas predicciones.

Alternativas:

- CF recursivo
 - Transitividad entre vecinos

Demo Time

(demo_11_recommender_systems)



Fin de la tercera clase

