

Tema 4.1

Regularización sobre modelos

Deep Learning

Máster Oficial en Ingeniería Informática

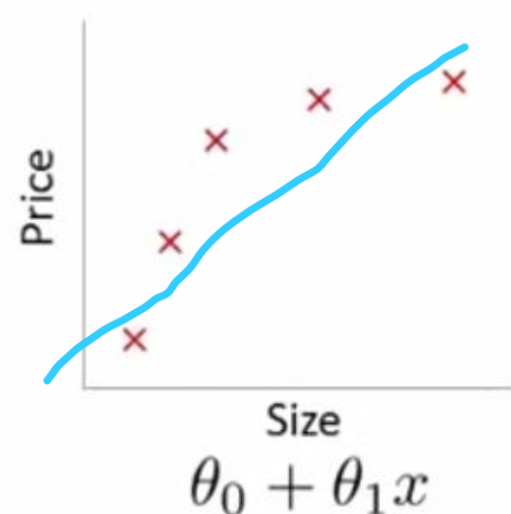
Universidad de Sevilla

Contenido

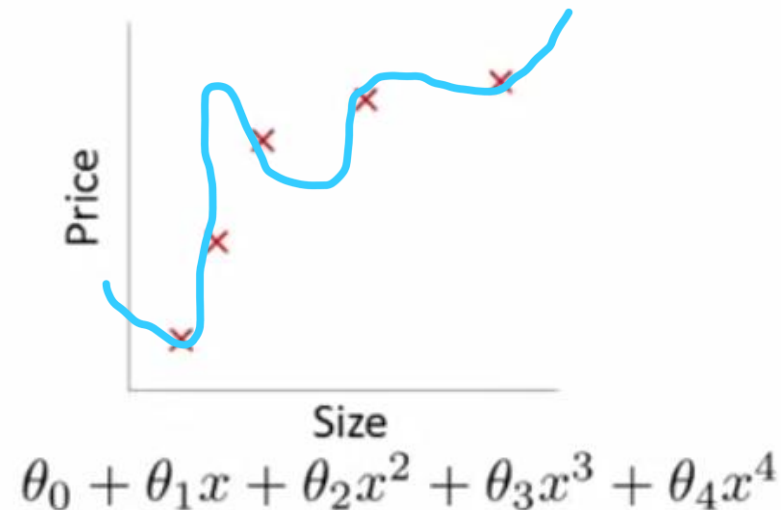
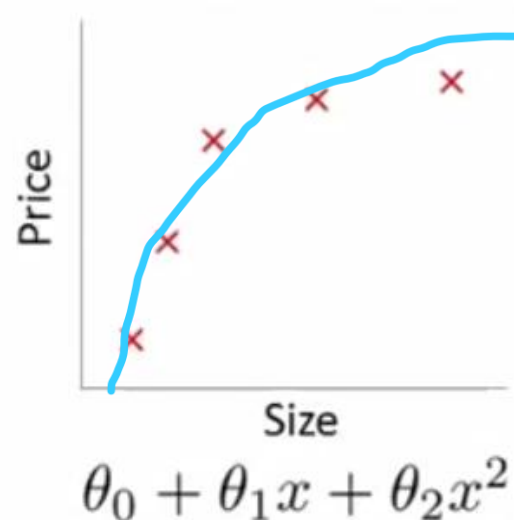
- Necesidad de regularización
- Penalización de parámetros
- Early stopping
- Ensemble
- Dropout
- Batch normalization

Necesidad de regularización

- En machine learning, buscamos modelos precisos a la vez que generalistas.
- El problema del **overfitting**:



Underfitting (high bias)



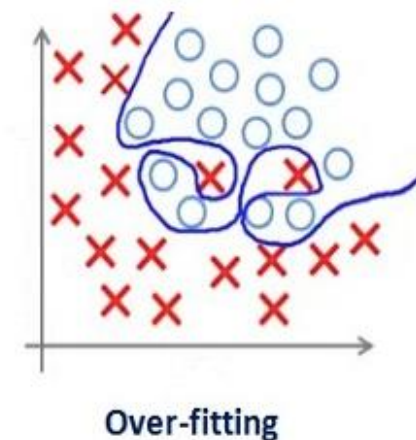
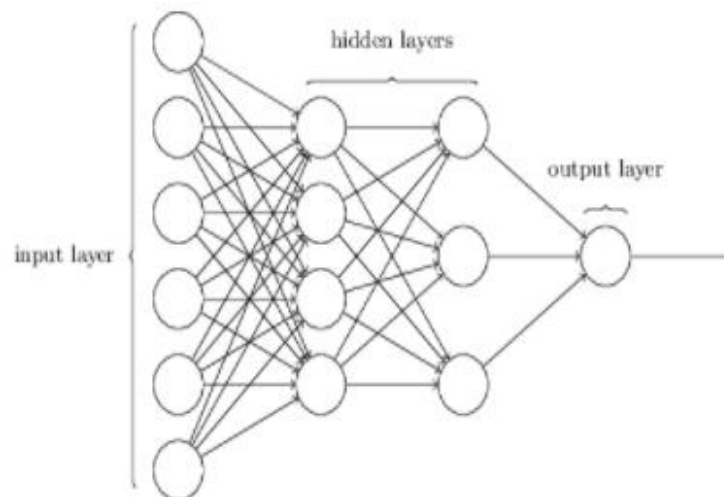
Overfitting (high variance)

Necesidad de regularización

- Las técnicas para atacar el overfitting son:
 - Reducir el número de características
 - Manualmente
 - Técnicas de reducción de dimensionalidad
 - Regularización
 - Mantenemos todas las características
 - Reducimos los valores de los parámetros del modelo
 - Funciona bien cuando tenemos muchas características y todas aportan un poco a predecir

Necesidad de regularización

- Técnicas de regularización ayudan a:
 - Combatir el overfitting
 - Generalizar mejor sobre los datos
 - Obtener modelos más simples
 - Obtener modelos más robustos



Penalización de parámetros

- Controlar la capacidad del modelo añadiendo una función de penalización a la función de coste:

$$\hat{J}(\theta) = J(\theta) + \lambda\Omega(\theta)$$

- Tendremos hipótesis más simples.
 - Distintos valores de los parámetros θ puede dar el mismo valor de pérdida, incluso si estos son valores muy extremos.
 - Si $x=[1,1,1,1]$, $w_1=[1,0,0,0]$ y $w_2=[0.25,0.25,0.25,0.25]$: $x*W_1=x*W_2=1$
- Menos dado a generar overfitting.

Penalización de parámetros

- Regularización L1 (lasso o dispersa):
 - Minimiza los parámetros del modelo calculando la suma de sus valores absolutos.
 - $\Omega_{L^1}(\theta) = \frac{1}{2} \|\theta\|_1 = \sum_i |\theta_i|$
- Regularización L2 (ridge, weight decay):
 - Minimiza los parámetros del modelo computando su norma euclídea.
 - $\Omega_{L^2}(\theta) = \frac{1}{2} \|\theta\|_2^2 = \sum_i |\theta_i|^2$

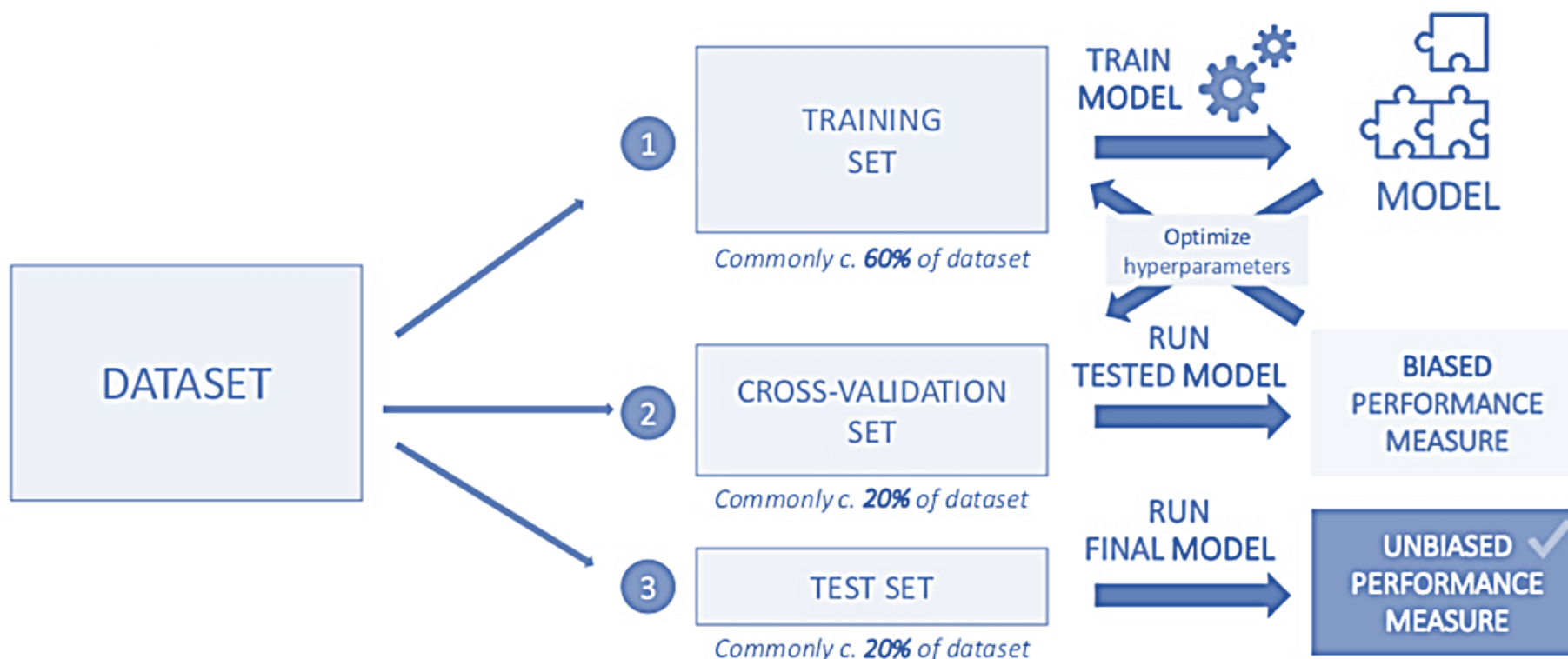
Penalización de parámetros

- Regularización L1 vs L2

	L1	L2
Coste computacional	Alto	Bajo
Solución única	No	Sí
Efecto sobre parámetros	Dispersos	Valores bajos
Con descenso gradiente	No siempre	Sí

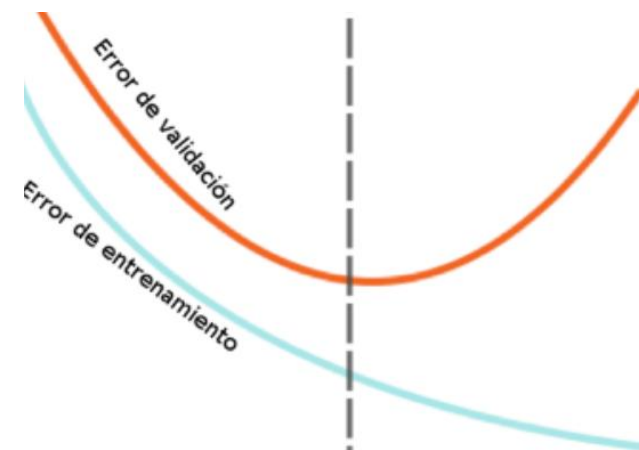
Early stopping

- Nuestro dataset está previamente dividido en tres partes principales



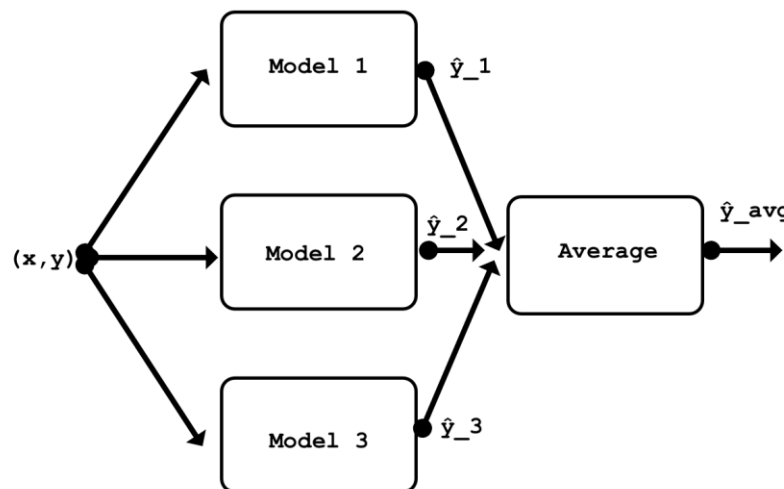
Early stopping

- **Idea:** detener el entrenamiento cuando el error cometido sobre el conjunto de validación crece.
- Requiere guardar una copia del mejor modelo obtenido
- Hiperparámetro: **p = paciencia**, número de evaluaciones sobre validación antes de detener el entrenamiento
- Estrategia popular por efectividad y simplicidad.
- Procedimiento:
 - Entrenar con early stopping usando training set
 - Entrenar usando training+validation set



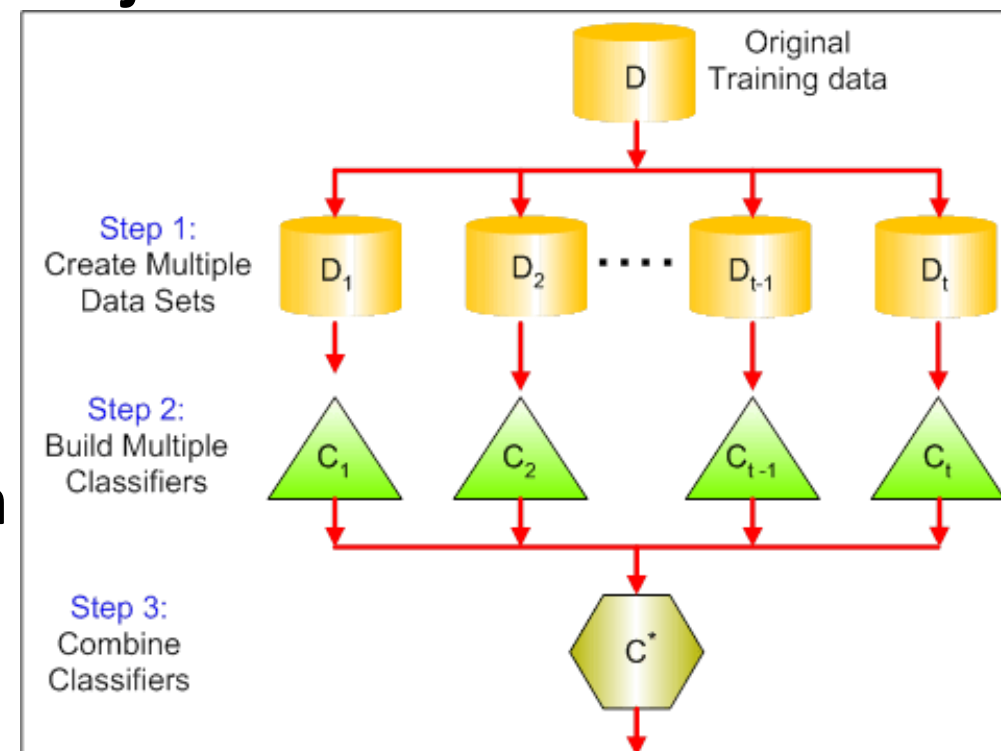
Ensemble

- El método de **ensamblado** aumenta la generalización combinando varios modelos:
 - **Model averaging**: entrenar varios modelos por separado y obtener la media (mayoría, máximo, suma...) de los votos de todos ellos para la predicción final.
 - Para obtener buenos resultados es necesario que la respuesta de los modelos **no este correlacionada**



Ensemble

- Para obtener diferentes modelos podemos modificar la **hipótesis**, la **función de evaluación**, el **optimizador** o el **conjunto de datos**.
- **Bagging (bootstrap aggregating)** se usan **k** conjuntos de datos diferentes.
 - Cada conjunto de datos se construye mediante muestreo con reemplazamiento.
 - De media tendremos 2/3 del original.
- Desventaja: mayor tiempo de computación y memoria.

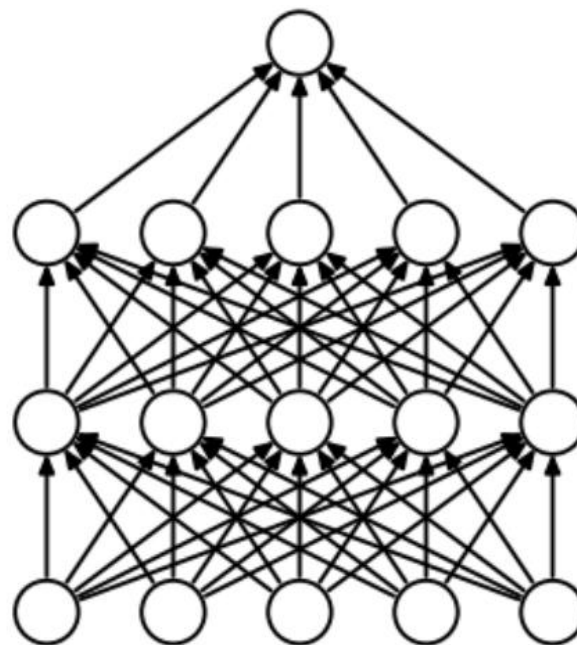


Dropout

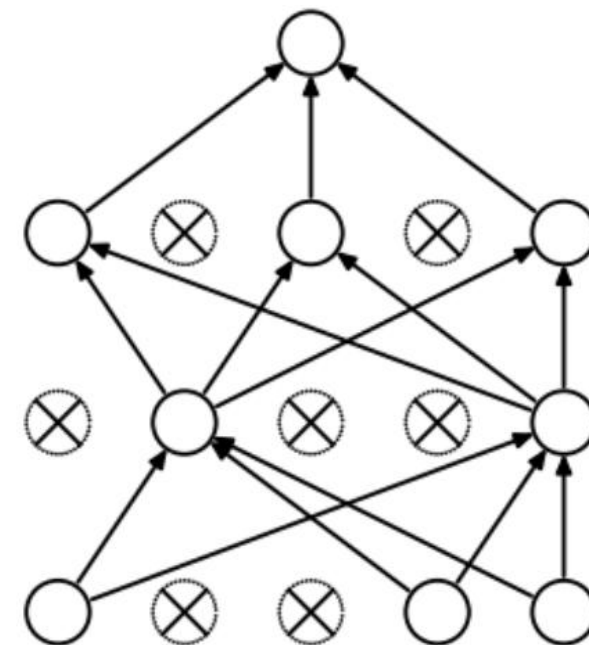
- **Idea:** aleatoriamente **poner a cero** algunas neuronas en la propagación hacia adelante

[[Srivastava et al 2014](#)]

- **Hiperparámetro: p**
 - Probabilidad de poner a cero
- Es decir, de media en una capa con L neuronas, se desactivan $L * p$ neuronas.



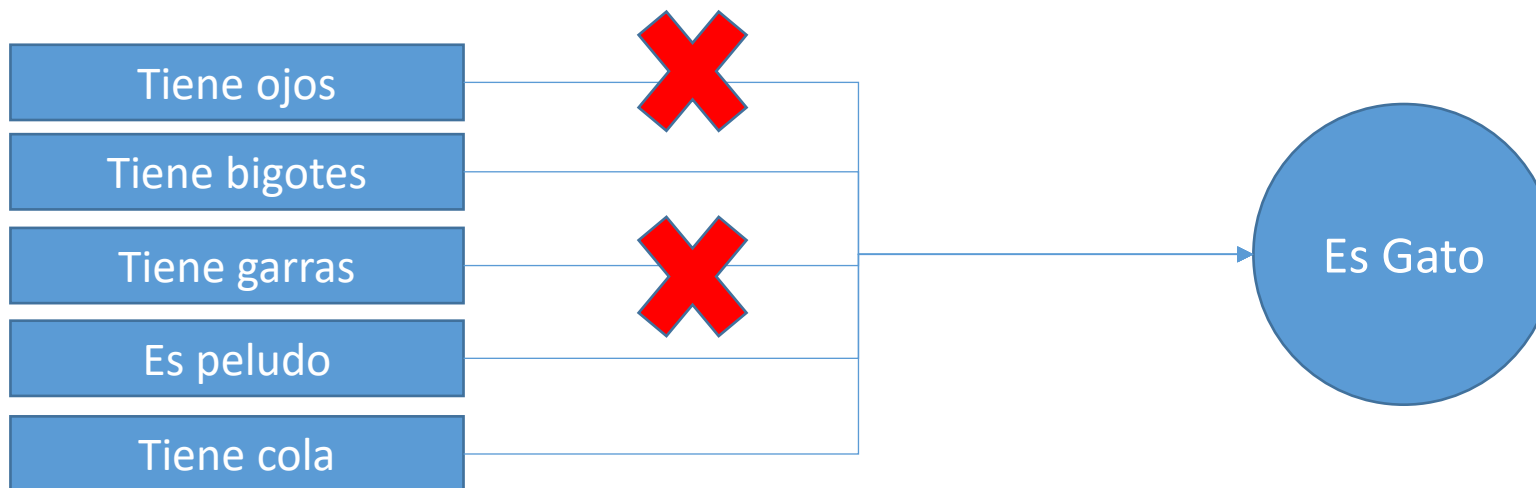
(a) Standard Neural Net



(b) After applying dropout.

Dropout

- Fuerza la red tener una representación más redundante
 - La red encuentra otros “caminos” dentro de la red para llegar a la misma conclusión



Dropout

- **En la fase de entrenamiento:**

- Por cada capa oculta, por cada ejemplo de entrenamiento, por cada iteración, ignorar una fracción aleatoria (p) de nodos.

- **En la fase de test:**

- Usar todas las activaciones, pero reducir los valores en un factor p
- ¿Por qué? Para contar las activaciones desestimadas durante el entrenamiento:
 - Con $p=0,5$, usar todas las entradas en la fase de propagación hacia adelante inflaría las activaciones por 2 de lo que la red estaba acostumbrada durante el entrenamiento. Tenemos que compensarlo escalando las activaciones a $\frac{1}{2}$.

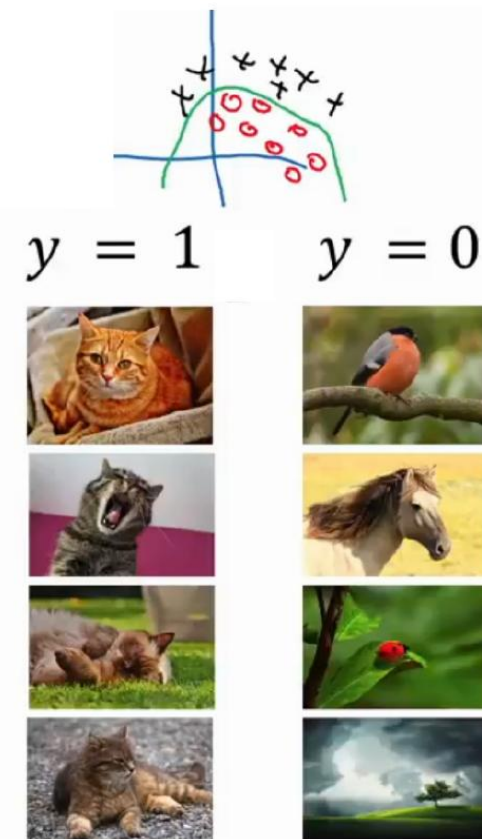
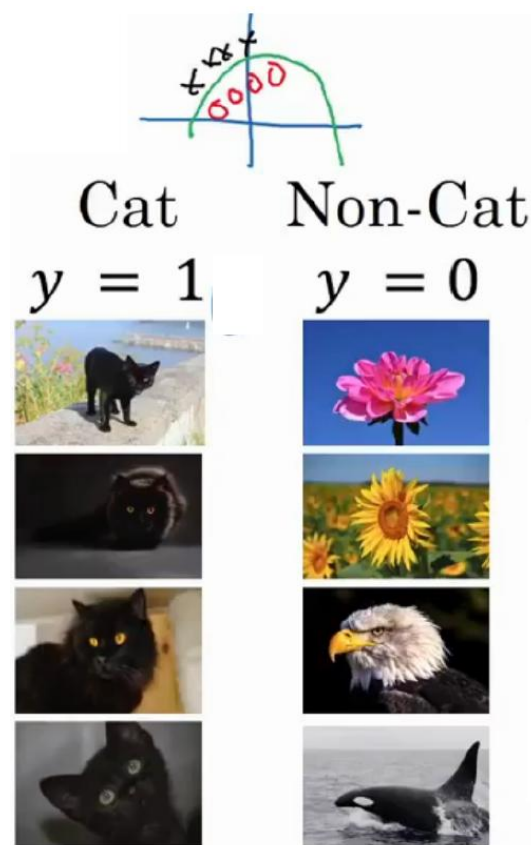
- **Dropout invertido:** se escala por $1/p$ solo en entrenamiento

Dropout

- Dropout equivale a entrenar un **gran ensamblado de modelos que comparten parámetros**.
 - Cada filtro nodos es un modelo que se entrena por solo esa iteración.
 - Con L neuronas en total en capas ocultas, tenemos 2^L posibles modelos.
 - No es exactamente igual a bagging, donde cada modelo se entrena por separado.
- **Ventajas:**
 - Funciona bien, barato de aplicar, no se limita a ningún tipo de modelo, compatible con el resto de técnicas de regularización.
- Un valor bueno de **p es 0,5**.

Batch normalization

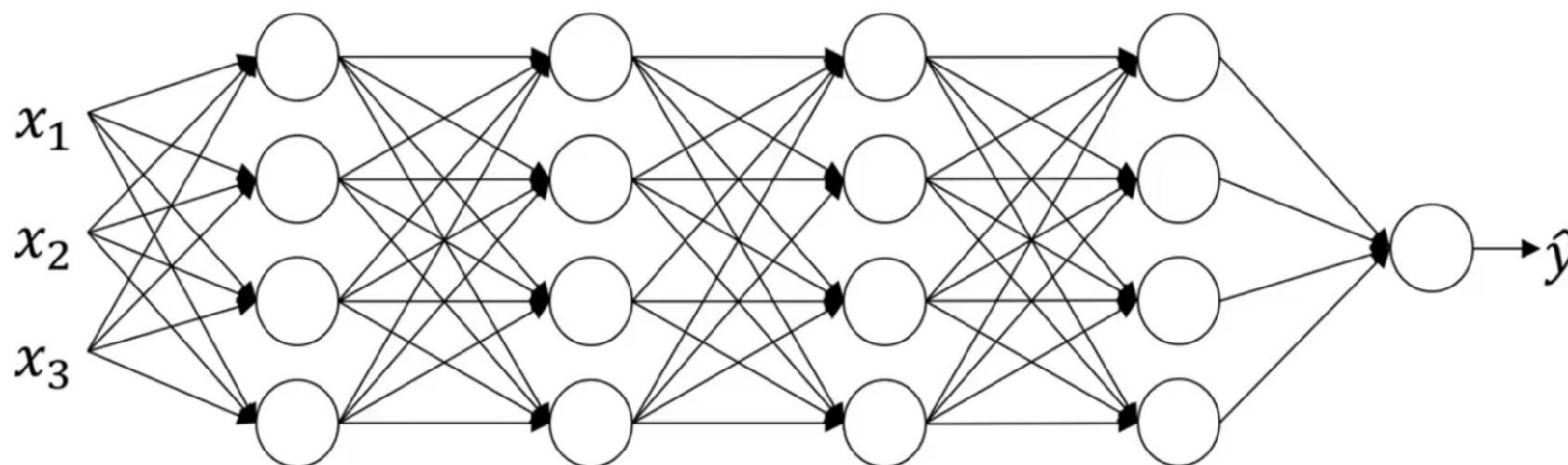
- Hemos visto la importancia de normalizar los datos:
 - La actualización de los pesos depende de las entradas!
- **Covariate shift problem:**
Cambio de distribución de datos puede hacer que nuestro modelo no generalice bien.



Batch normalization

- Motivación

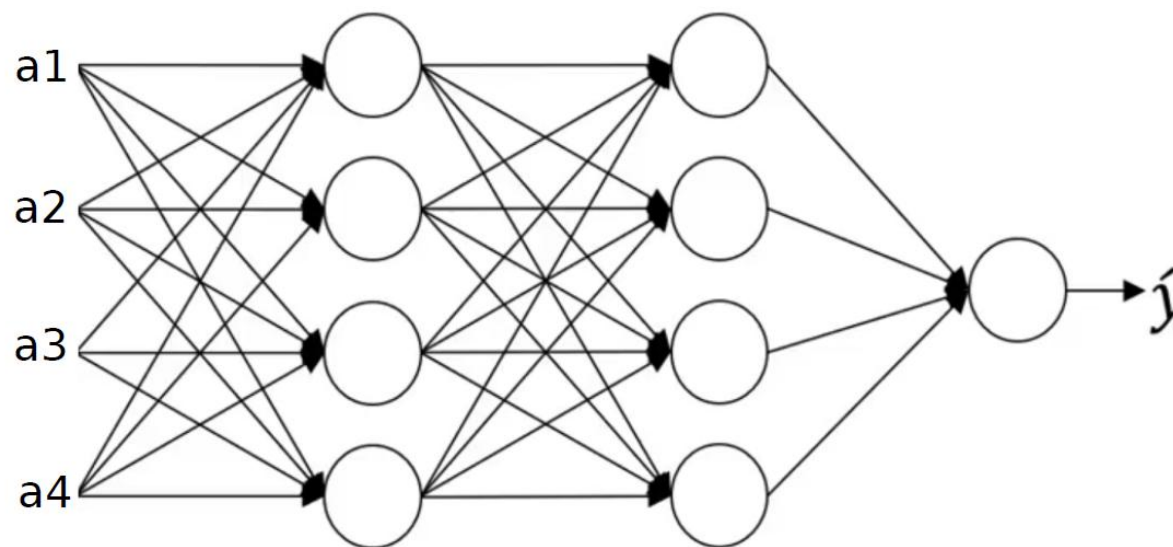
- Supongamos una red perceptrón multicapa,
- Centrémonos en la tercera capa.



Batch normalization

- **Motivación**

- La salida de la segunda capa (entrada de la tercera) cambiará constantemente durante el entrenamiento.
- **Idea:** reducir la cantidad de cambios normalizando los valores.



Batch normalization

- Introduce nuevos parámetros a aprender:
 - γ factor de escalado
 - β factor de desplazamiento
 - Pueden determinar si es necesario normalizar o no.
- Se suele aplicar antes de función de activación:
 - Evitar problema *vanishing/exploding gradient*.

[[Ioffe y Szegedy 2015](#)]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch normalization

- Ventajas:
 - Las redes son menos sensibles a los valores iniciales.
 - Se pueden usar un learning rate mayor.
 - Reduce el tiempo de entrenamiento, convergencia más rápida.
 - Funciona como un regularizador, reduciendo la necesidad de usar otras técnicas de regularización.
- Desventaja: la predicciones son mas lentas.

Recapitulación

- Importancia de evitar el sobreajuste con técnicas de regularización
- **Regularización L1/L2**
- **Early Stopping** para detener el entrenamiento cuando haya pérdida de generalización
- **Ensamblado** de modelos (p.ej. bagging) para una respuesta más robusta.
- **Dropout** para una red más robusta → Fácil de usar