• Deep Learning

• Fundamentos de la IA

DuocUC®

ESCUELA DE
INFORMÁTICA Y
TELECOMUNICACIONES

Generalización

- Función de error
- Métricas de Desempeño
- Generalización

DuocUC®

Resumen

En clasificación, la función de salida recomendada es softmax, la cual genera un vector ($\hat{\mathbf{y}}$) tan largo como clases tenga que clasificar y los valores de cada posición representan una probabilidad de que sea esa clase.

Softmax se asegura que cada uno de los valores del vector esté entre 0 y 1 y que la suma de todas sus posiciones no sea > 1 (o 100%)

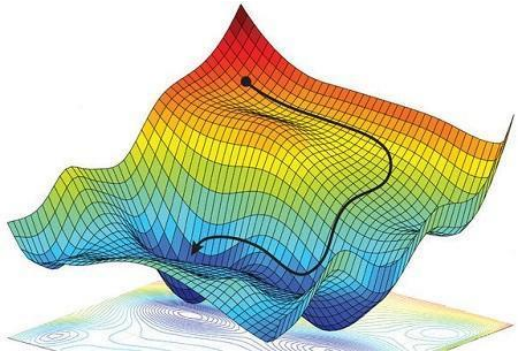
Por otro lado, $\mathbf{y}$ también es un vector del mismo largo que $\hat{\mathbf{y}}$, donde todas sus posiciones valen 0, excepto la posición que contiene la clase correcta, la cual se le asigna el valor 1.

La definición del descenso del gradiente: Pasar todos los ejemplos por la red, calcular el error promedio que la red está cometiendo con de todos los ejemplos y luego usar ese promedio como una función para actualizar los pesos de la red´.

El problema con el descenso del gradiente es que la cantidad de ejemplo podrían ser millones o billones y pasar todos estos ejemplos por la red para recién hacer un solo paso del descenso del gradiente es demasiado costoso computacionalmente.

En la práctica, el error se estima usando SGD (descenso estocástico del gradiente): Se escoge un grupo de ejemplos (batch) al azar. Este batch de ejemplos se pasa por la red y se calcula el error promedio solo con estos ejemplos. Finalmente se utiliza backpropagation solo con esos valores y descenso del gradiente para actualizar los parámetros, luego se repite cuantas veces se estime necesario.

En rigor, se desordenan todos los ejemplos, se dividen de acuerdo al tamaño del batch elegido, por ejemplo, si hay 100 ejemplos y se escoje un tamaño 5 de batch, se crearan 20 batches de 5 ejemplos.
Una vez que terminados de pasar los 20 batches, se completa 1 iteración o en jerga de Deep learning, 1 época (EPOCH)



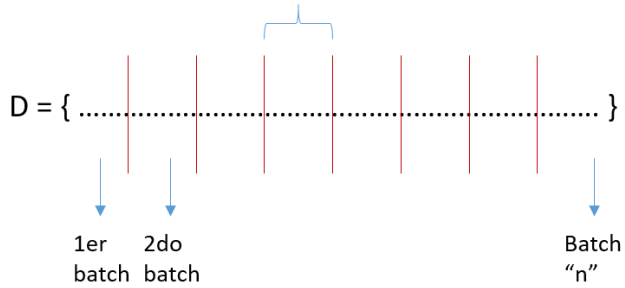
$$y = f(u)$$

$$u = f(x)$$

$$y = f(f(x))$$

$$\frac{dy}{dx} = \frac{dy}{du} \times \frac{du}{dx}$$

Tamaño del batch (batch size) es el número de ejemplos en un batch



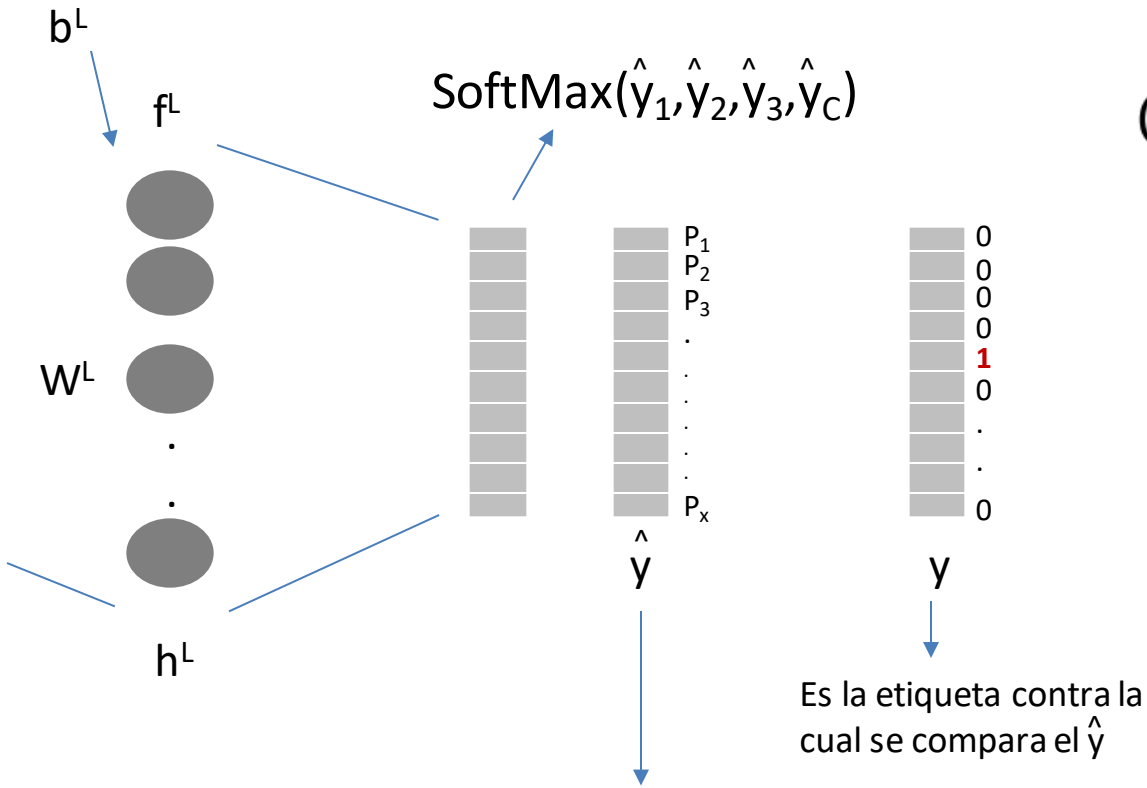
DuocUC®

Función de error $\mathcal{L}$

La función de error asegura que el entrenamiento sea consistente y bueno

Para clasificación es crossentropy, que compara 2 distribuciones de probabilidades, **y** vs $\hat{\mathbf{y}}$ (Para clasificación binaria, suele usarse sigmoid)

En la práctica, **y** es transformado a un vector que representa una distribución de probabilidad (one-hot), donde la posición correcta de la clase en este vector es 1 (100% de probabilidad) y el resto en 0 (0% de probabilidad)

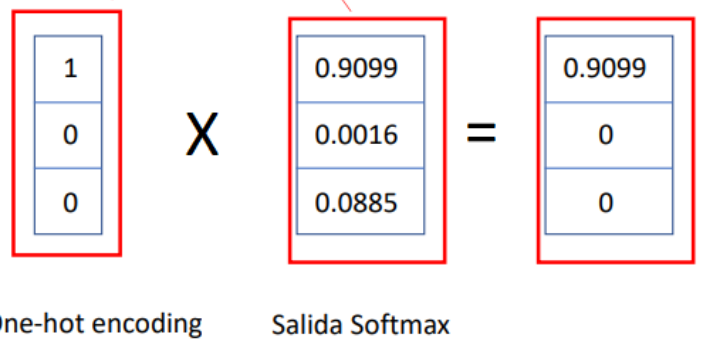


$$Vector_{(Salida)} = (\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_C)$$

El vector de salida $\hat{y}$ es una distribución de probabilidad sobre todas las clases definidas

Cross-entropy Loss

$$L(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$



Métricas de Desempeño

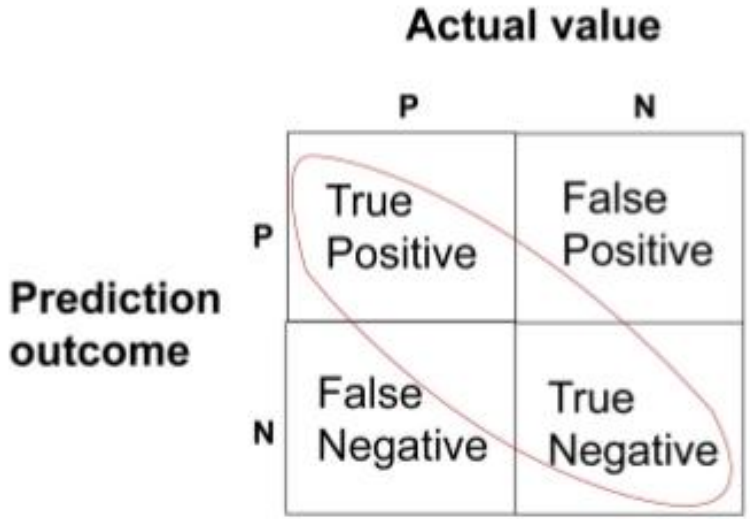
La métrica de desempeño más usada de las NN es la **Accuracy (Exactitud)**

Se calcula al mismo tiempo que la función de error, es decir, al hacer el forward de una muestra (ejemplo).

Se define como el % total de ejemplos clasificados correctamente

Funciona muy bien cuando las clases están equilibradas.

Con esta métrica, se toma la decisión si se continúa o no entrenando la red.



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Accuracy: Es el número de verdaderos positivos y verdaderos negativos dividido por el número de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos

Generalización

Al ajuste de los parámetros (W, b) se le conoce como **Optimización,** usando TRAINING SET

La **generalización** es el error promedio que la red comete en datos que **NO** son de entrenamiento, es decir, en datos que nunca haya visto, usando TESTING SET

En Deep learning, los datos de test NO se pueden usar para el entrenamiento y se debe cuidar que estos sean tan parecidos como se pueda a los que se usaran en la realidad.

En Deep learning, lo más importante no es el error de entrenamiento obtenido, sino que es el error de generalización. **El error promedio que comete la red con los datos de test**

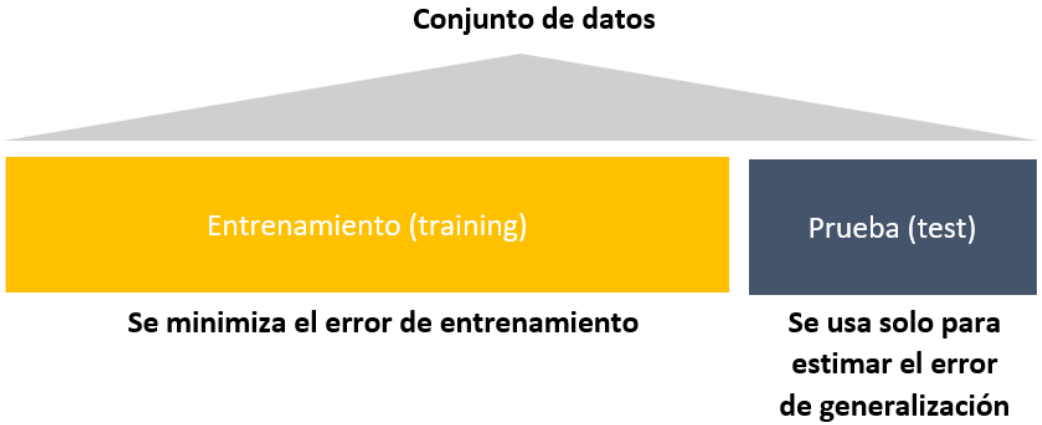
Para crear una buena generalización, en general hay que estresar la red.

Ejemplo, si existe un conjunto de datos de mala calidad y otro de buena de calidad, se entrena con el de mala calidad y se prueba con el bueno.

La motivación es que los datos de test sean lo más parecido a la realidad, lo que se va a encontrar cuando el modelo esté en producción.

La **capacidad** tiene que ver con todas las posibles funciones que la red será capaz de aprender, es decir, al crear una red, se define y fija la cantidad de neuronas y la cantidad de capas. Cada vez que los parámetros son ajustados, se crea una nueva red – la misma arquitectura – pero una nueva red. Hay una cantidad fija de funciones que se podrán aprender.

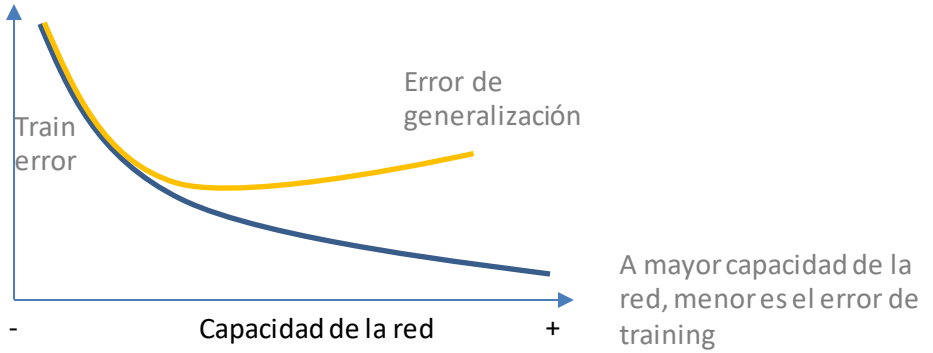
(Existe un tercer grupo de ejemplos, este se llama Development test en Deep learning, conocido también como validation test, el cual es utilizado para buscar cuál es la capacidad correcta de la red)



Generalización

La capacidad de una red se controla (aumenta o disminuye) con los hiperparámetros:

- Más cantidad de capas
- Más cantidad de neuronas por capas
- Tasa de aprendizaje
- Tiempo entrenamiento (cant. épocas)
- Funciones de activación
- Tamaño del batch
- Regularización



Hay que tener un cuidado especial, en general, cuando se le da mayor capacidad a la red.

Baja el error de entrenamiento, pero aumenta el error de generalización.

Finalmente, es el error de generalización el que interesa minimizar.

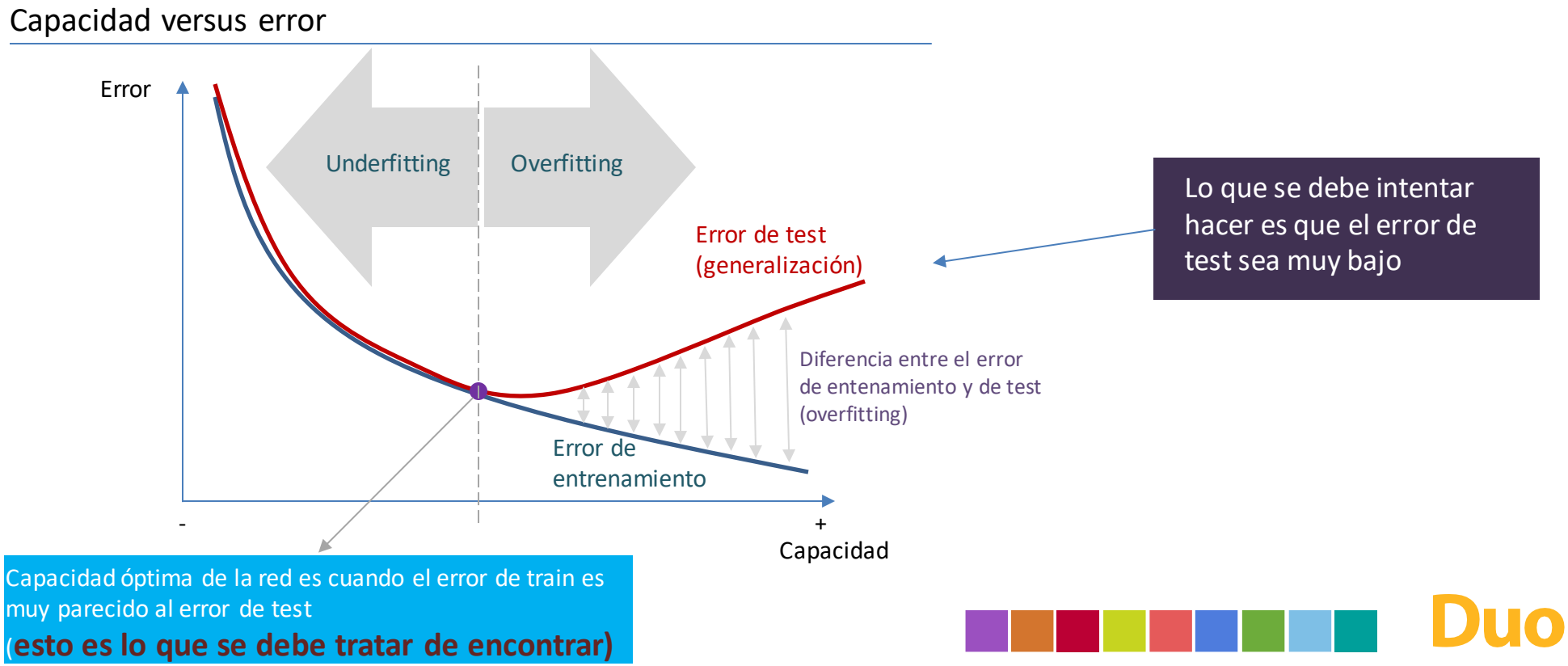
Se puede caer en **overfitting y underfitting**

Capacidad de una red y el impacto en la generalización

Overfitting es cuando la diferencia entre el error de entrenamiento y el error de test es muy grande (Error de entrenamiento baja y Error de test sube), En otras palabras, cuando se observa el error con un dato que no se está entrenando, el error es más alto

Underfitting es cuando el error de entrenamiento es muy grande
Con redes neuronales, "nunca" se debería estar en underfitting, basado en el teorema de universal de aproximación, a menos que los datos sean muy, pero muy malos (ejemplo, pésimamente etiquetados)

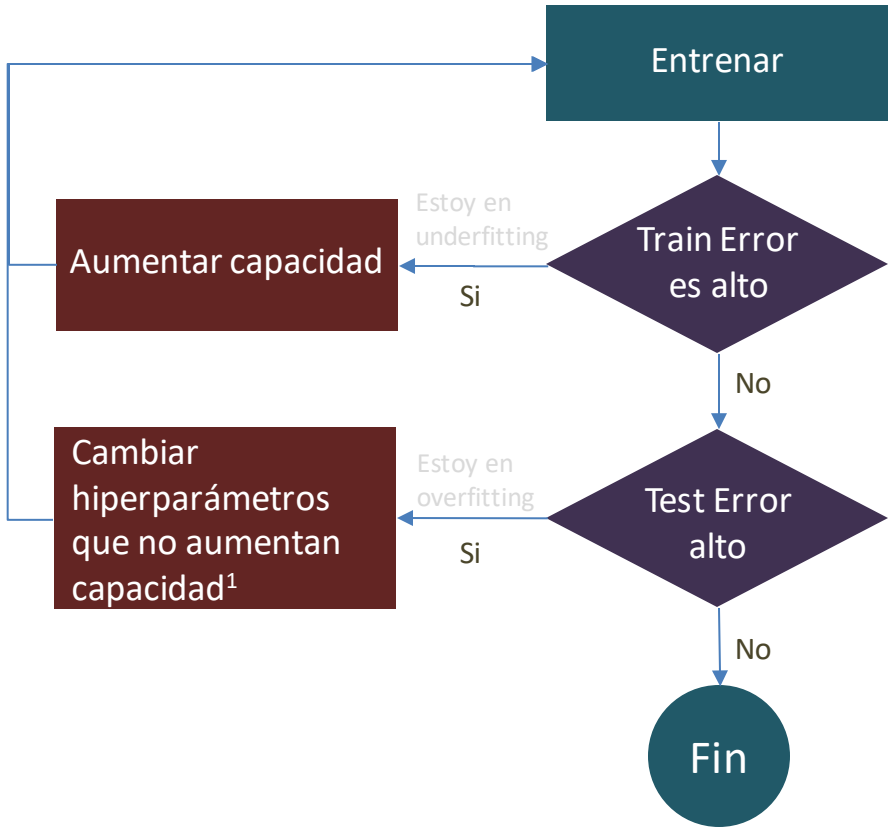
En general, siempre el error de entrenamiento es más bajo que el error de test



Pasos (tips) prácticos para mejorar la generalización en Deep learning

Este algoritmo es un excelente camino para mejorar la generalización de la red

Es importante saber que en Deep learning, conseguir más datos o ejemplos, en general es lo más caro, cuesta dinero



- Este “loop” podría ser caro (cómputo, dinero (por los datos adicionales a conseguir), tiempo para entrenar)
- En general, se debiera tender a una red con gran capacidad (tener un modelo tan grande como pueda)
- Aumentar la regularización, que es un término que no hemos visto todavía

DuocUC®