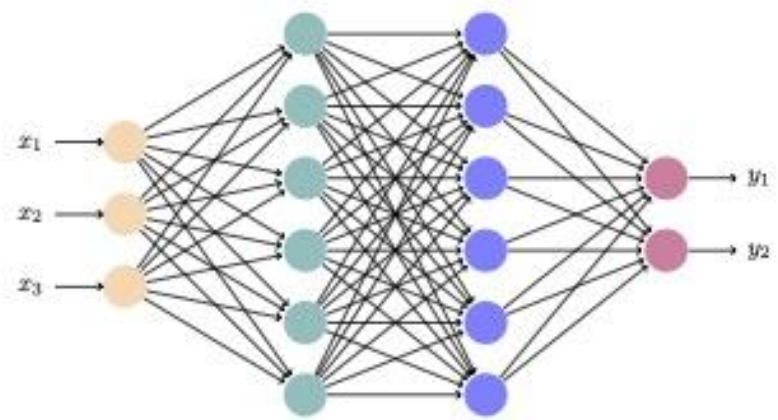
Fundamentos Matemáticos del Aprendizaje Profundo

1er cuat. 2025
Clase 2

Repaso

Una red neuronal codifica una función. Dadas las funciones de activación fijas para cada neurona, la función de salida depende de los pesos y sesgos w, b.



La salida de la red es $y = f_{w, b}(x)$.

Dada la función objetivo $z = \Phi(x)$ cómo elijo (w, b) para “aprenderla”.

Funciones de costo

Funciones de costo

Entradas de la red:

- unidimensional: x número real
- multidimensional: **x** = $(x_1,\ldots, x_n)$
- aleatorio: X variable aleatoria

Salidas de la red:

- unidimensional: y número real
- multidimensional: **y** = $(y_1,\ldots, y_k)$
- aleatorio: Y variable aleatoria

La red provee un mapa de entrada salida que depende de los parámetros de la misma

$$y = f_{w,\, b}(x)$$

w = pesos (weights)
b = sesgos (bias)

Funciones de costo

Dados una elección de parámetros y sesgos, w y b, se mide el error que se comete al intentar aproximar la función objetivo $z=\phi(x)$ por la salida de la red $y = f_{w,\, b}(x)$.

La función de costo depende de w y b: C(w, b).

Luego se buscan los valores de pesos y sesgos que minimicen el costo.

$$(w_{op},\, b_{op}) = \text{argmin } C(w,\, b)$$

El proceso de determinar los valores $(w_{op},\, b_{op})$ se denomina **aprendizaje**.

Ejemplos

1. Supremo o Máximo

$$C(w,b) = \sup_{x \in [0,1]} |f_{w,b}(x) - \phi(x)|.$$

En aplicaciones, se conoce ϕ en finitos puntos

$$z_i = \phi(x_i), \qquad 1 \leq i \leq n,$$

luego utilizamos la función costo

$$C(w,b) = \max_{1 \leq i \leq n} |f_{w,b}(x_i) - z_i|.$$

Ejemplos

2. Error L^2.

$$C(w,b) = \int_0^1 (f_{w,b}(x) - \phi(x))^2 \, dx.$$

Nuevamente, si conocemos al objetivo en finitos puntos

$$z_i = \phi(x_i), \qquad 1 \leq i \leq n,$$

obtenemos

$$C(w,b) = \sum_{i=1}^{n} (f_{w,b}(x_i) - z_i)^2 = \|f_{w,b}(\mathbf{x}) - \mathbf{z}\|^2$$

Ejemplos

3. Error cuadrático medio

X = variable aleatoria Y = $f_{w,\,b}(X)$ = variable aleatoria Z = función objetivo (v.a.)

$$C(w,b) = \mathbb{E}[(Y - Z)^2] = \mathbb{E}[(f_{w,b}(X) - Z)^2].$$

Veremos más adelante algoritmos para hallar (w_{op}, b_{op})

¿Qué ventajas tienen estas funciones de error que las hace útiles?

Error L^2 y error cuadrático medio

1. Tanto las funciones cuadrado integrable como las v.a. de cuadrado integrable forman un *Espacio de Hilbert* (spoiler: esto es muy bueno).

2. La salida de la red será la *mejor predicción de Z con la información dada por los datos*.

3. Si X y Z son v.a. independientes, entonces la mejor predicción es E[Z],

Ejemplos

4. Entropía cruzada: p, q densidades.

Función de verosimilitud negativa: $-\ell_q(x) = -\ln q(x).$

(cantidad de información que proporciona q)

Entropía cruzada de p con respecto a q

$$S(p,q) = \mathbb{E}^p[-\ell_q] = -\int_{\mathbb{R}} p(x) \ln q(x)\, dx.$$

Es la información proporcionada por q, pesada por la densidad p.

Observemos que si 0≤q≤1 (p.ej si q es discreta), entonces S(p,q)≥0.

Algunos resultados

PROPOSICIÓN 3.2.1. *Se tiene la desigualdad* $S(p,q) \geq H(p)$, *donde* $H(p)$ *es la* entropía de Shannon *definida como*

$$H(p) = S(p,p) = -\mathbb{E}^p[\ell_p] = -\int_{\mathbb{R}} p(x) \ln p(x)\, dx.$$

COROLARIO 3.2.2. *Sea* X *una variable aleatoria sobre* $\mathbb{R}$ *con densidad* p. *Asumimos que* X *tiene esperanza y varianza finita. Entonces*

$$H(p) \leq \frac{1}{2} \ln(2\pi e \mathbb{V}(X)),$$

donde $\mathbb{V}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$ *es la varianza.*

Observación: Entropía → clasificación, L^2 → regresión.

Entrenamiento

Entrenamiento

El conjunto de datos (x_i, z_i), $i \in I$, se divide en 3 partes

- Entrenamiento (aprox 70% de los datos)
- Testeo (aprox 20% de los datos)
- Validación (el 10% restante)

Se procede aproximadamente de esta manera:

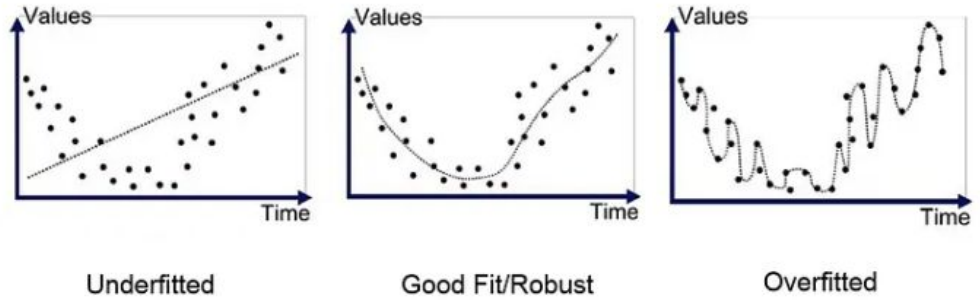
1. Se minimiza la función de costo sobre los datos de entrenamiento y se obtienen pesos y sesgos óptimos para este conjunto (w_e, b_e).
2. Se evalúa el costo sobre para ese valor de los parámetros sobre los datos de testeo. Se espera que el costo sea mayor, dado que los parámetros fueron optimizados para otros datos.

Entrenamiento (2)

Se tienen las siguientes alternativas:

a) Si los errores de entrenamiento y testeo son ambos pequeños, la red “generaliza bien”.
b) Si el error de entrenamiento es pequeño, pero el error de testeo es grande, la red no generaliza y probablemente se tenga un problema de overfitting ⇒ regularizar el costo o modificar la arquitectura.
c) Ambos errores son grandes ⇒ subajuste. Nueva arquitectura aumentando los parámetros.

Sobreajuste (Overfitting) - Subajuste (Underfitting)



Para evitar overfitting, es conveniente limitar el tamaño de los pesos (que no sean arbitrariamente grandes). Una forma es *penalizar*

$$C_\lambda(w, b) = C(w, b) + \lambda\, |(w,b)|^2$$

(λ es un “hiperparámetro”)

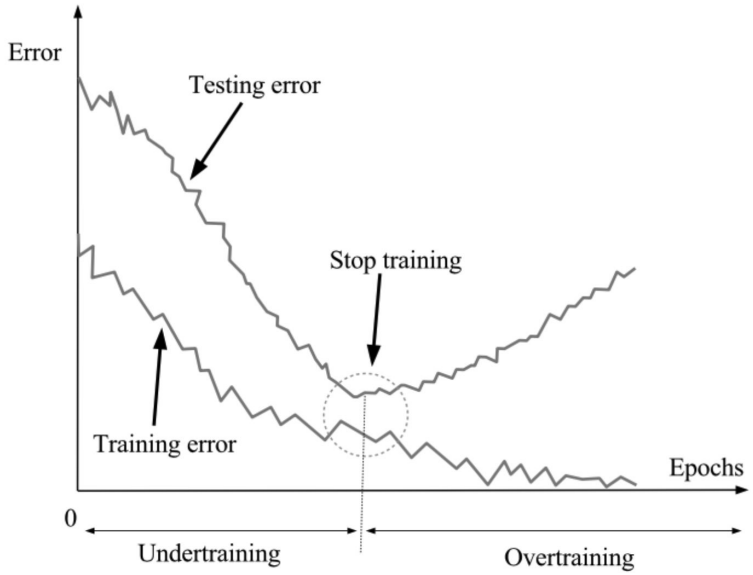
Entrenamiento (3)

El conjunto de validación es usado para ajustar los *hiperparámetros*.

> “Finding the optimal hyperparameters is more like an art rather than science, depending on the scientist’s experience, and we shall not deal with it here”

Entrenamiento (4)

¿Cuántos ciclos de entrenamiento son necesarios?



Resumen

- El proceso de aprendizaje consiste en minimizar la función de costo

$$C(w, b) = |f_{w, b}(x) - z|$$

- Se parten los datos en:
  - Entrenamiento
  - Testeo
  - Validación
- Entrenamiento+Testeo: la red generaliza o no.
- Problemas de overfitting → penalizar. Uso de un hiperparámetro λ. Con los datos de validación se ajusta este valor.