MEJORAS A BACKPROPAGATION

Extensiones y variantes

Dos defectos principales de BP: *lento*

Mínimos locales

Algunas mejoras:

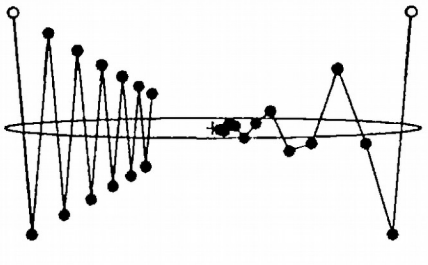
- *Momento*:

$$\Delta w_{pq}(t+1) = -\eta \frac{\partial E}{\partial w_{pq}} + \alpha \Delta w_{pq}(t)\,.$$

→ Si superficie de costo plana, acelera en un factor 1/(1-α)

$$\Delta w_{pq} \approx -\frac{\eta}{1-\alpha}\frac{\partial E}{\partial w_{pq}}$$

Si hay oscilaciones, las fluctuaciones son escaladas por η



- *Parámetros adaptivos*:

$$\Delta\eta = \begin{cases} +a & \text{si } \Delta E < 0 \text{ en los últimos pasos} \rightarrow \text{crece aritméticamente} \\ -b\eta & \text{si } \Delta E > 0 \rightarrow \text{decrece geométricamente} \\ 0 & \text{en otro caso} \end{cases}$$

Si $\Delta E > 0 \rightarrow \eta$ decrece $\rightarrow$ se anula la modificación
a = 0 hasta un paso exitoso (si se estaba usando momento)

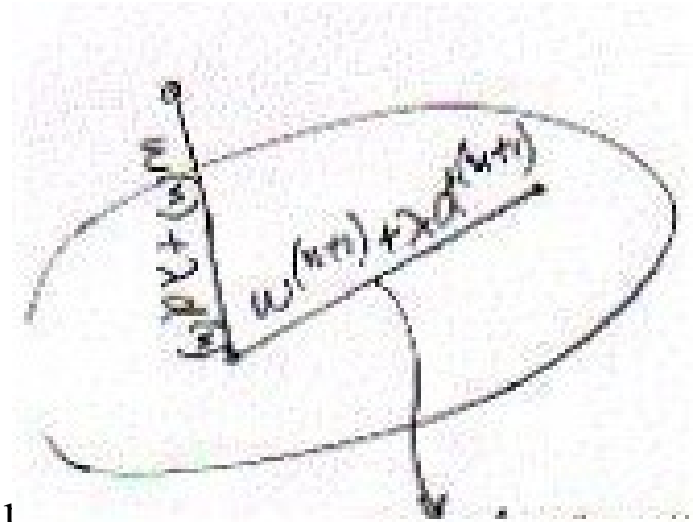
- *Otras técnicas determinísticas*: steepest descent
Gradientes conjugados
Quasi-Newton

- *Técnicas estocásticas*

Gradientes conjugados

$$\mathbf{d}^{\mathbf{new}} = -\nabla E^{\mathbf{new}} + \beta \mathbf{d}^{\mathbf{old}}$$

Cada nueva dirección es un compromiso entre la del gradiente y la anterior.
Condición: *β tal que d no cambie la componente del gradiente a lo largo de la dirección previa* (componente que ya era nula).



la dirección del gradiente debe ser ortogonal a la del paso anterior.

$$\mathbf{d}^{\text{old}} \cdot \nabla E(\mathbf{x}_0 + \lambda \mathbf{d}^{\text{new}}) = 0$$

$$0 = \frac{\partial}{\partial \lambda} E(\mathbf{x}_0 + \lambda \mathbf{d}^{\text{old}}) = \mathbf{d}^{\text{old}} \cdot \nabla E^{\text{new}}$$

$$\mathbf{d}^{\text{old}} \cdot \mathbf{H} \cdot \mathbf{d}^{\text{new}} = 0 .$$

→*el d nuevo y el anterior son vectores conjugados*

Fórmula de Polack-Ribière

$$\beta = \frac{(\nabla E^{\text{new}} - \nabla E^{\text{old}}) \cdot \nabla E^{\text{new}}}{(\nabla E^{\text{old}})^2}$$

Fórmula de Fletcher-Reeves (la original)

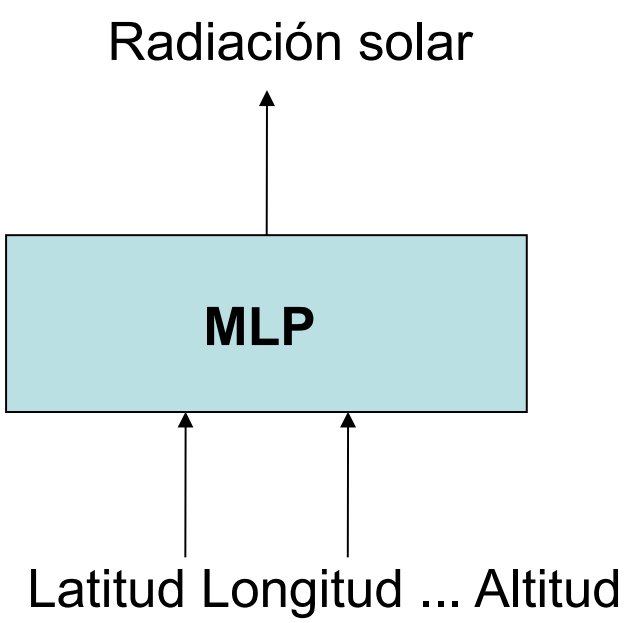
$$\beta = \frac{(\nabla E^{(n+1)})^2}{(\nabla E^{(n)})^2}$$

- Las últimas n (dimensión) direcciones son mutuamente conjugadas
- No requiere conocer H
- Superior a BP en general. Encuentra el mínimo de una superficie cuadrática en n pasos
- Computacionalmente más complejo. Sensible al λ de cada paso

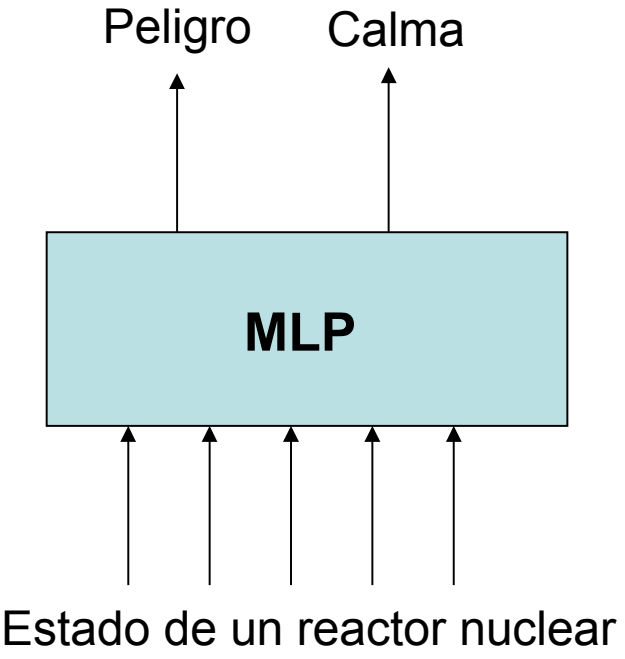
Perceptrón Multicapa

¿Para qué se puede usar un perceptrón multicapa?

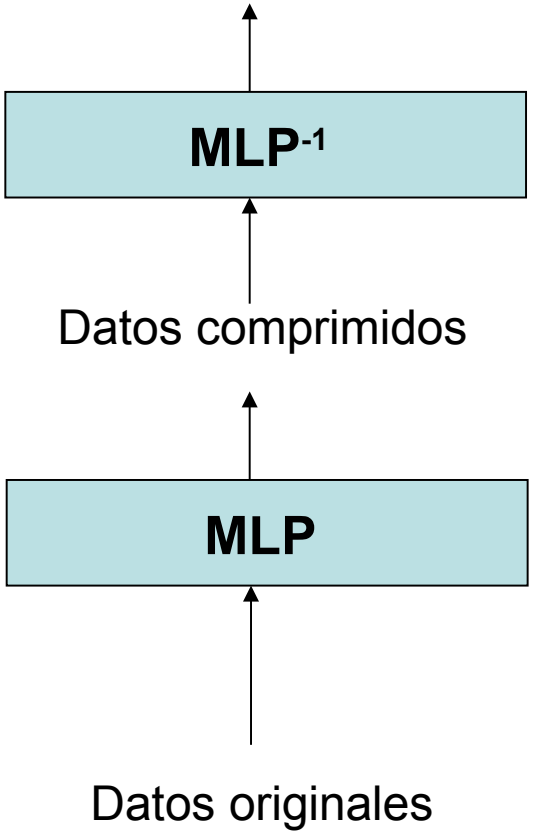
Regresión



Clasificación

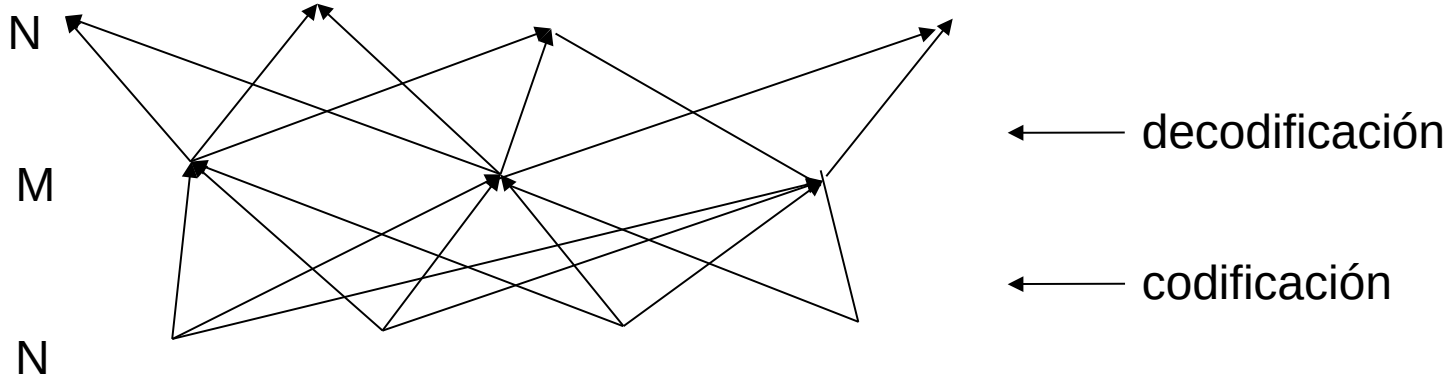


Compresión



Compresión

*Codificador (*encoder*)*: codificar información en las unidades ocultas



$\xi_i = \zeta_i = \delta_{i\mu}$ (el patrón μ sólo tiene prendida la unidad μ)

La capa intermedia codifica en binario el número de patrón

$\longrightarrow$ $M \geq log2(N)$ (condición)

A menor *M,* más eficiente la codificación.

BP puede encontrar esquemas alternativos, mediante valores no saturados en las unidades ocultas, resolviendo el problema incluso en casos en que esa cota no se cumple

Utilidad: *compresión de imágenes*

Calibración de cámara de video

f(x,y) = (d,θ,h)

x,y coordenadas de la cámara (CCD)

d,θ distancia y ángulo al objeto

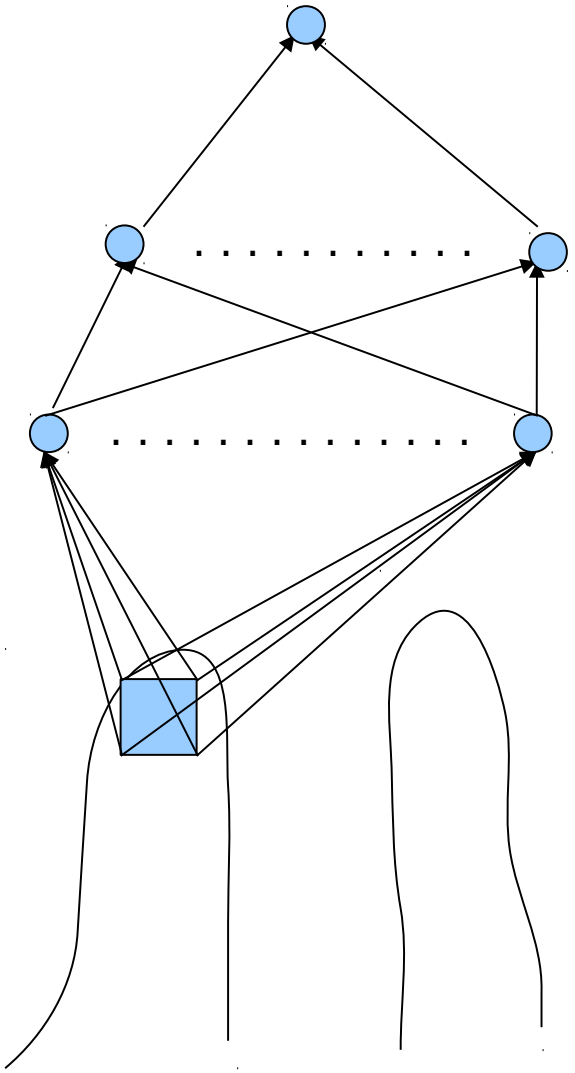
h tamaño del objeto

Se utiliza un perceptrón multicapa (una capa oculta) para aproximar la función f.

1) Muestrear f: $(d,\theta) \rightarrow (x,y)$

2) *"Invertir" f* $(x,y) \rightarrow (d,\theta)$

Reconocimiento de patrones



Salida

Capa oculta 2 (M2)

Capa Oculta 1 (M1)

Capa de Entrada (NxN)

Aprendizaje

I-*Generalización*: relación I/O correcta para patrones nunca enseñados

≠

II-*Memorización*: el sistema opera como una “look-up table”, el mapeo que la red computa no es (necesariamente) suave

≠

III-*Interpolación*: la red ajusta exactamente los patrones enseñados, la respuesta a nuevos estímulos es inestable (error no acotado)

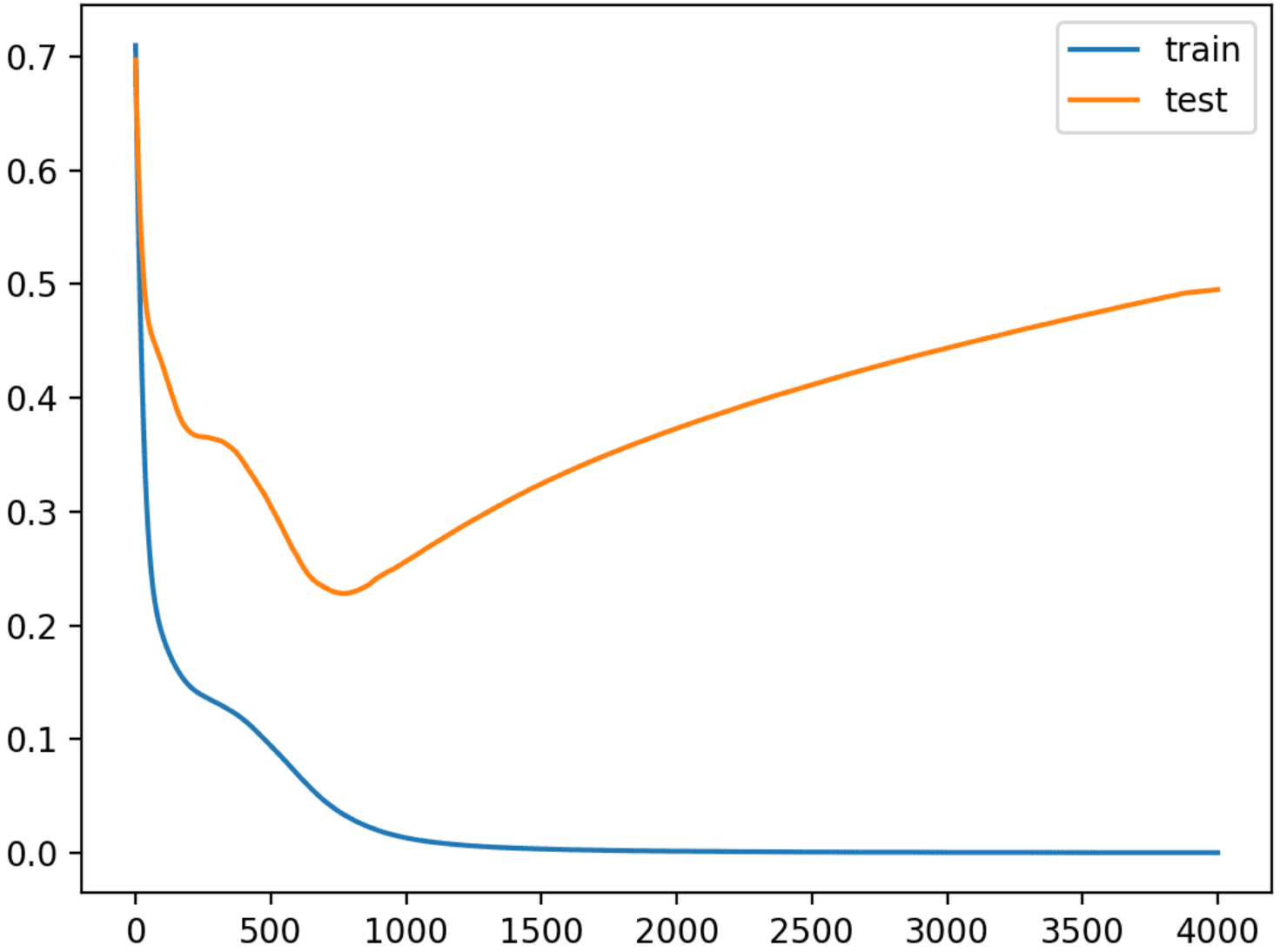
Para que I no degenere en II --> prevenir el sobreentrenamiento

Para que I no degenere en III --> Principio de la *Navaja de Occam* para selección del modelo: elegir la función más simple en ausencia de información a priori en contrario

Factores que influyen en la generalización

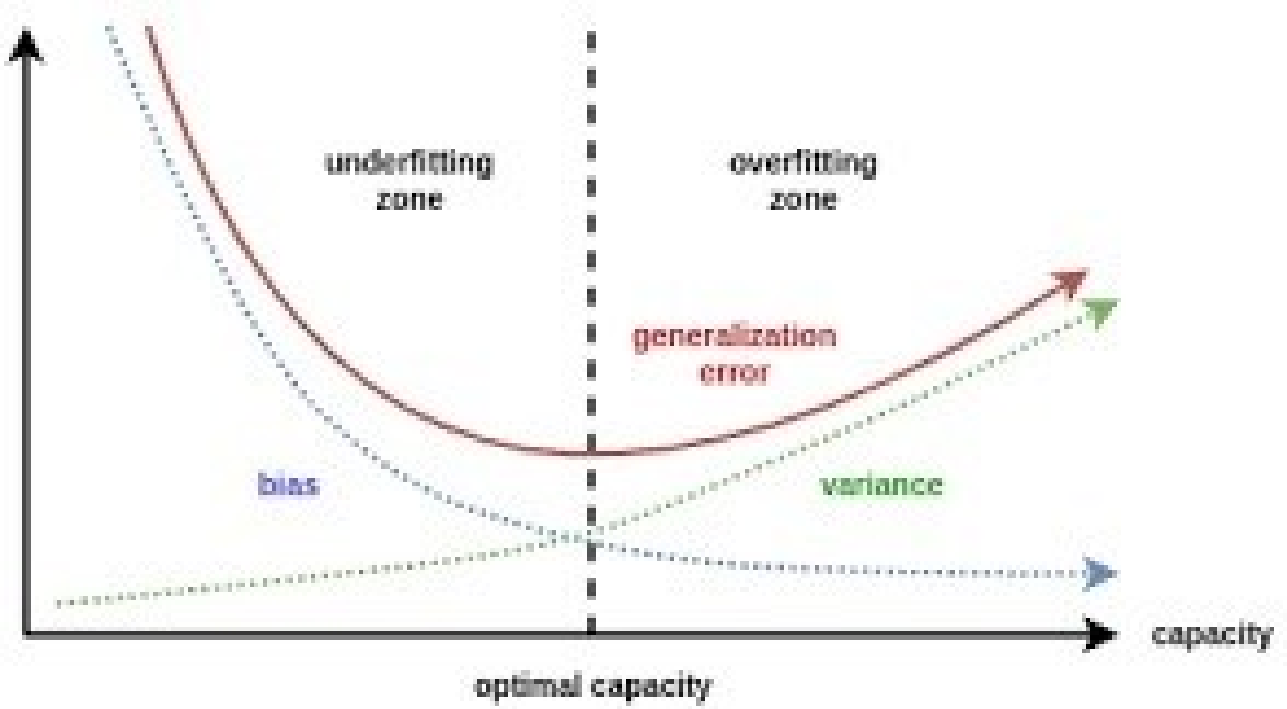
1) Tamaño y eficiencia del conj. de entrenamiento
2) Arquitectura de la red
3) Complejidad del problema

Early stopping



El dilema sesgo-varianza

Sesgo → incompleta estimación de parámetros → subentrenamiento (underfitting)
Varianza → sensibilidad ante fluctuaciones del conjunto de entrenamiento → sobreentrenamiento (overfitting)



→ Principio de la navaja de Okham: equilibrio complejidad/universalidad

El dilema sesgo-varianza

Sesgo alto → gran diferencia entre el valor medio predicho por el modelo y el valor medio real.
→ modelo demasiado simple que no ha aprendido las relaciones relevantes entre las variables disponibles y la variable a predecir → underfitting o subajuste → errores de entrenamiento y test altos.

Ejemplo: ajustar un comportamiento complejo y no lineal con una línea recta

Varianza alta → modelo demasiado complejo, presta atención en exceso a las peculiaridades del subconjunto empleado (que es variable, variando el modelo resultante)
→ overfitting → predicciones pobres ante datos no vistos (testeo)

Ejemplo: una curva que pasa por todos los datos de entrenamiento.

Buscamos un modelo que:
- haya aprendido de los datos que se le han proporcionado (sesgo y error de entrenamiento bajos)
- capaz de generalizar ante nuevos datos (varianza y error de test bajos).

Si intentamos disminuir el sesgo aumentando la complejidad del modelo, aumentará la varianza.
Si intentamos minimizar la varianza disminuyendo la complejidad, aumentará el sesgo.

Validación Cruzada

Conjunto de datos

- *Entrenamiento*
    1) estimación del modelo (= entrenamiento)
    2) evaluación de su performance (10-20 % de 1)

3) *Testeo*: desconocido a priori para el diseñador (idealmente)

1) diferentes estructuras candidato
→ 2) elección de la mejor
→ entrenamiento con el conjunto completo
→ 3) evaluación de la capacidad de generalización de la red resultante

Una performance uniforme ante distintas validaciones cruzadas indica un buen modelo