



Día Day	Mes Month	Año Year

## 1. Análisis de componentes Principales (PCA)

Descripción del modelo:

PCA (Principal Component Analysis) es una técnica de reducción de dimensionalidad no supervisada cuyo objetivo principal es transformar un conjunto de variables posiblemente correlacionadas en un nuevo conjunto de variables ortogonales llamadas componentes principales. Estas componentes ~~tan mayor~~ retienen la mayor parte de la variancia de los datos originales.

Aplicaciones:

- Visualización de datos en 2D o 3D.
- Preprocesamiento para modelos de clasificación.
- Compresión de datos.

Problema de optimización:

PCA busca encontrar una base ortogonal que maximice la variancia de los datos proyectados. Formalmente, el problema se expresa como:

$$\max_{W \in \mathbb{R}^{d \times K}} \text{tr}(W^T X^T X W) \text{ sujeto a } W^T W = I_K$$

donde:

- $X \in \mathbb{R}^{n \times d}$  es la matriz de datos centrados,
- $K$  es la cantidad de componentes principales,
- $W$  contiene los vectores propios (autovectores) de la matriz de covarianza.

Dia Day	Mes Month	Año Year
------------	--------------	-------------

## 2. UMAP (uniform Manifold Approximation and projection).

Modelo:

UMAP es un método no lineal de reducción de dimensionalidad que preserva tanto la estructura local como global de los datos al construir un grafo en el espacio de alta dimensional y tratar de preservarlo en el espacio reducido.

Problema de optimización:

Minimizar la divergencia entre las distribuciones de pares de puntos en los espacios original y proyectado.

$$\min_Y C(Y) = \sum_{(i,j)} [p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1-p_{ij}) \log \frac{1-p_{ij}}{1-q_{ij}}]$$

Donde:

- $p_{ij}$  es la probabilidad de conexión entre  $i$  y  $j$  en el grafo original (alta dimensional).
- $q_{ij}$  es la probabilidad en la proyección (baja dimensional).
- $Y$  es la representación baja dimensional.

UMAP es estocástico y no lineal, lo que lo hace muy expresivo pero también dependiente de hiperparámetros como  $n$ -neighbors, min-dist, y metric.

Día Day	Mes Month	Año Year
------------	--------------	-------------

### 3. Naive Bayes (GaussianNB)

Descripción del modelo:

Naive Bayes es un modelo generativo probabilístico que asume la independencia condicional entre características dado el valor de la clase. La versión Gaussian del modelo, conocida como GaussianNB, supone además que cada característica sigue una distribución normal para cada clase.

Función de decisión:

A partir del teorema de Bayes, se define la probabilidad posterior de una clase dada una instancia.

$$P(y=c|x) \propto P(y=c) P(x_i|y=c)$$

En GaussianNB:

$$P(x_i|y=c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} \exp\left(-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}\right)$$

No hay un problema de optimización explícito.

Los parámetros del modelo ( $\mu_{ic}$ ,  $\sigma_{ic}^2$  y  $P(y=c)$ ) se estiman directamente con la media, varianza y frecuencia de ocurrencia en los datos de entrenamiento.

Ventajas:

- Muy rápido de entrenar
- Eficiencia en escenarios de muchas dimensiones.
- Robusto con pocos datos, aunque limitado por suposiciones fuertes de independencia.

## 4. SGDClassifier (Descenso de gradiente estocástico)

Descripción del modelo:

SGDClassifier es una implementación eficiente de clasificadores lineales entrenados mediante descenso de gradiente estocástico. Este método es particularmente útil para trabajar con conjuntos de datos muy grandes o en aprendizaje en línea.

Modelo general:

SGDClassifier puede adaptarse para fines con diferentes funciones de pérdida, como:

- Perceptrón: función escalón.
- SVM: pérdida hinge.
- Regresión logística: pérdida logarítmica.

Ejemplo de problema de optimización (regresión logística):

$$\min_{w, b} \frac{1}{n} \sum_{i=1}^n \log \left( 1 + e^{-y_i(w^T x_i + b)} \right) + \lambda \|w\|^2$$

donde:

- $\lambda$  es el parámetro de regularización L2 o L1.
- El optimizador recorre muestras aleatorias o mini-lotes, actualizando los parámetros en cada iteración.

Ventajas:

- Escalable a millones de muestras.
- Adoptable a flujos de datos.
- Compatible con regularización y diferentes funciones de pérdida.



Dia Day	Mes Month	Año Year
------------	--------------	-------------

### 3. Regresión logística

Descripción del modelo:

la regresión logística es un modelo lineal para clasificación binaria (o multiclase con extensiones). Predice la probabilidad de pertenencia a una clase mediante la función sigmoidal aplicada a una combinación lineal de los características.

$$P(y=1 | X) = \frac{1}{1 + e^{-(w^T x + b)}}$$

Problema de optimización:

la función objetivo minimiza la entropía cruzada o pérdida logarítmica entre las predicciones y las verdaderas etiquetas:

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^n \log \left( 1 + e^{-y_i(w^T x_i + b)} \right) + \lambda \|w\|^2$$

- el término de regularización  $\lambda$  previene el sobreajuste ( $L_2$  por defecto).
- el problema es convexo, por lo que se garantiza una solución óptima global.

Ventajas:

- Fácil de interpretar.
- Rápido de entrenar.
- Buena rendimiento en problemas linealmente separables.

## K-Nearest Neighbors (KNeighbors classifier)

### Notación y espacio de trabajo:

- **ESPAZIO MÉTRICO**  $(\mathcal{X}, d)$  con  $d: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$   
que satisface simetría, identidad y desigualdad triangular.  
Tipicamente  $\mathcal{X} \subseteq \mathbb{R}^d$  con  $d(x, x') = \|x - x'\|_p$  (norma de Minkowski),  
pero los resultados valen para cualquier espacio métrico separable.

- **Distribución subyacente:** Sea un par aleatorio  $(X, Y) \sim P$  donde

$$X \in \mathcal{X}, Y \in \mathcal{Y} := \{1, \dots, C\}, C \geq 2$$

denotamos la densidad de clase posterior

$$\eta_c(x) := P(Y=c | X=x), \quad \eta(x) := (\eta_1(x), \dots, \eta_C(x)).$$

- **Muestra de entrenamiento**  $(i, i, d):$

$$D_N := \{(X_i, Y_i)\}_{i=1}^N \stackrel{i.i.d.}{\sim} P.$$

### ESTIMACIÓN K-NN DE LAS PROBABILIDADES POSTERIORES

Sea  $x \in \mathcal{X}$  un punto de consulta.

- **Conjunto vecinos**  $N_K(x) := \arg \top - K_1 \leq i \leq N d(X_i, x)$ .
- **Estimador de  $\eta_c(x)$  (votación uniforme)**

$$\hat{\eta}_c^{(k)}(x) = \frac{1}{K} \sum_{i \in N_K(x)} \mathbf{1}_{\{Y_i=c\}} = \frac{\#\{i \in N_K(x) : Y_i=c\}}{K}.$$

(versión ponderada: sustituir  $1/K$  por pesos  $w_i(x) \propto d(X_i, x)^{-1}$ )

### Regla de decisión K-NN

$$\hat{g}^{(k)}(x) := \arg \max_{c \in \mathcal{Y}} \hat{\eta}_c^{(k)}(x)$$

### Riesgo de clasificación

- **Riesgo verdadero de un clasificador  $g$ :**

$$R(g) := P(g(x) \neq Y) = E[\mathbf{1}_{\{g(x) \neq Y\}}].$$

• Riesgo de Bayes (óptimo teórico):

$$R^* := \inf_{g: \mathcal{X} \rightarrow \mathcal{Y}} R(g) = \mathbb{E} [1 - \max_c \gamma_c(x)].$$

• Para  $g = \hat{g}^{(k)}$  se escribe  $R_N^{(k)} := R(\hat{g}^{(k)})$ .

Propiedad de consistencia

Teorema (Cover & Hart 1967 - consistencia para  $k=1$ ):

Si  $\dim(\mathcal{X}) < \infty$  y  $P(x=x)=0$  para todo  $x$ , entonces

$$\lim_{N \rightarrow \infty} R_N^{(1)} \leq 2R^*(1-R^*) < 2R^*.$$

Teorema (Stone 1977 - consistencia universal):

Sea  $K = K(N)$  tal que

$$K(N) \xrightarrow[N \rightarrow \infty]{} \infty, \quad \frac{K(N)}{N} \xrightarrow[N \rightarrow \infty]{} 0$$

bajo dicha dirección y suponiendo  $\mathcal{X}$  métrico separable y  $\gamma_c$  medible,

$R_N^{(K(N))} \xrightarrow[N \rightarrow \infty]{P} R^*$ . El clasificador  $K\text{-NN}$  es universalmente consistente; puede

alcanzar el riesgo de Bayes si se imponer hipótesis paramétricas sobre  $\gamma$ .

Tasa de convergencia

Para dimensión  $d \geq 1$  y densidad de  $X$  acotada y lipschitziana:

$$\mathbb{E}[R_N^{(K)} - R^*] = O((K/N)^{1/d} + \frac{1}{\sqrt{K}}).$$

equilibrando con  $K \asymp N^{2/(d+2)}$  se obtiene

$$\mathbb{E}[R_N^{(K)} - R^*] = O(N^{-2/(d+2)}).$$

La tasa empeora exponencialmente con  $d$  (maldición de la dimensionalidad).

Sub-problema explícitos de optimización

Aunque  $K\text{-NN}$  no "entrena" parámetros, si se formulan problemas de optimización relativos a hiperparámetros o métricas:

• Objetivo: Selección de  $K$   
formulación matemática:  $K^* = \arg \min_{K \in \{1, \dots, K_{\max}\}} \hat{R}_{cv}(K)$  donde  $\hat{R}_{cv}$  es el riesgo estimado por validación cruzada.

Naturaleza: Optimización discreta unidimensional (generalmente búsqueda exhaustiva).

• Objetivo: Metric Learning (P. ej) LMMN)

formulación matemática: se busca  $A \succeq 0$  que minimice  $\sum_{i,j \in T} d_A(x_i, x_j) + \mu \sum_{i,j \in L} \mathbb{1}_{\{y_i = y_j \neq y_L\}} [L + d_A(x_i, x_j) - d_A(x_i, x_j)] +$  con  $d_A(x, x') = (x - x')^T A (x - x')$ .

Naturaleza: Problema convexo semidefinido método de gradiente proyectado o sub-gradiente.

• Objetivo: Ponderación de vecinos.

formulación matemática: Para pesos locales  $w_i(x) \geq 0$ :  $\min_{w(\cdot)} \mathbb{E}[\ell(g_w(x), Y)] + \lambda \|w\|^2$ .

Naturaleza: Variación funcional o paramétrica; conduce a estimadores "locally weighted".

## Support Vector classifier (SVC)

$$D_N = \{(x_i, y_i)\}_{i=1}^N, x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$$

Datos de entrenamiento.

$$f(x) = \langle w, x \rangle + b, \quad w \in \mathbb{R}^d, b \in \mathbb{R}$$

función de decisión lineal.

$$g(x) = \text{Sign}(f(x))$$

clasificador.

## Caso separable (hard-margin SVM)

Si los datos linealmente separables, existe  $(w, b)$  tal que

$$y_i(\langle w, x_i \rangle + b) \geq 1, \quad \forall i$$

Problema de optimización (primal)  $\min_{w,b} \frac{1}{2} \|w\|_2^2$

$$\text{Sujeto a } y_i(\langle w, x_i \rangle + b) \geq 1, \quad i=1, \dots, N.$$

Objetivo: maximizar el margen geométrico

$$\gamma = \min_i \frac{y_i(\langle w, x_i \rangle + b)}{\|w\|}$$

minimizar  $\frac{1}{2} \|w\|^2$

equivale a maximizar  $\gamma$ .

## Caso no separable (soft-margin C-SVC)

introducimos variables de holgura (slack)  $\xi_i \geq 0$ .

problema primal

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i$$

$$\text{Sujeto a } y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \quad i=1, \dots, N.$$

$$\xi_i \geq 0, \quad i=1, \dots, N.$$

- $C > 0$  regula el trade-off entre maximizar el margen y penalizar errores.

## Formulación dual y "kernel trick"

Sca  $\alpha = (\alpha_1, \dots, \alpha_N)^T$  el vector de multiplicadores de Lagrange.

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{Sujeto a } 0 \leq \alpha_i \leq C, \quad i=1, \dots, N,$$

$$\sum_{i=1}^N \alpha_i y_i = 0$$

Kernel positivo definido

$$K(x, x') = \langle \phi(x), \phi(x') \rangle_H,$$

donde  $\phi: \mathcal{X} \rightarrow \mathbb{H}$  es una inmersión en un espacio de Hilbert reproducing Kernel (RKHS).

del teorema del representante se deduce

$$w = \sum_{i=1}^N \alpha_i y_i \phi(x_i)$$

función de decisión en el espacio kernel

$$f(x) = \sum_{i=1}^N \alpha_i y_i K(x_i, x) + b \Rightarrow g(x) = \text{sign}(f(x))$$

los puntos con  $0 < \alpha_i < C$  son los vectores de soporte ( $\alpha_i > 0$  soportan el hiperplano óptimo).

condiciones de Karush - Kuhn - Tucker (KKT)

Para cada  $i$ : 1)  $\alpha_i \geq 0$ ,  $\gamma_i \geq 0$  2)  $\alpha_i [y_i f(x_i) - 1 + \gamma_i] = 0$   
3)  $C - \alpha_i \geq 0$ ,  $(C - \alpha_i) \gamma_i = 0$  4)  $\sum_i \alpha_i y_i = 0$

las KKT permiten: - determinar  $b$  mediante cualquier vector de soporte con  $0 < \alpha_i < C$ .

- caracterizar la región de margen:

- $0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1$
- $\alpha_i = C \Rightarrow y_i f(x_i) \leq 1$  (punto mal clasificado o en interior del margen)
- $\alpha_i = 0 \Rightarrow y_i f(x_i) \geq 1$  (punto correctamente clasificado y fuera del margen).

Perspectiva de pérdidas y regularización

el problema primal soft-margin es equivalente a minimizar el riesgo empírico con pérdida hinge regularizado por la norma de  $w$ :

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \ell_{\text{hinge}}(y_i, f(x_i)),$$

donde

$$\ell_{\text{hinge}}(y, t) = \max \{0, 1 - yt\}.$$

## Gorantías de generalización basadas en margen.

Sea  $\gamma$  el margen geométrico logrado y  $R$  un radio tal que  $\|x_i\| \leq R$  para todos los  $i$ .

- Cota de Vapnik (lineal, separable)

$$R(g) \leq \frac{4}{N} \left( \frac{R^2}{\gamma^2} \right), \text{ con alta probabilidad.}$$

(indica que un margen grande reduce el riesgo de generalización).

- VC-dimension: si los datos se encapsulan en una bola de radio  $R$  y se consigue margen  $\gamma$ , la VC-dimension satisface

$$d_{VC} \leq \min \left\{ \frac{R^2}{\gamma^2}, d \right\} + 1.$$

## Predicción de hiperparámetros y kernels

- Parámetro de penalización  $C$ : controla la constante de regularización

$$\lambda = \frac{1}{2C} \quad \text{Busca} \quad C^* = \arg \min_{C>0} \hat{R}_{CV}(C).$$

- Parámetros de kernel (p.ej.  $\sigma$  en RBF):

$$(\sigma^*, C^*) = \arg \min_{\sigma, C} \hat{R}_{CV}(\sigma, C).$$

Al ser problemas de optimización continuos no convexos respecto a  $(\sigma, C)$ , se resuelven con búsqueda en rejilla o métodos Bayesianos; la fase de ajuste ocurre fuera del problema convexamente optimizado del SVC.

## Complejidad computacional teórica.

• Entrenamiento: resolver el dual requiere  $O(N^2)$  memoria, y en general, entre  $O(N^2)$  y  $O(N^3)$  operaciones; algoritmos SMO y variantes chunking reducen el coste práctico.

• Predicción:  $f(x) = \sum_i \epsilon_{SV} \alpha_i y_i k(x_i, x) + b$ . complejidad  $O(|SV|dk)$ , donde  $dk$  es el coste de evaluar  $k$ .

## Random Forest Classifier

$$D_N = \{(x_i, y_i)\}_{i=1}^N \stackrel{i.i.d}{\sim} P, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{1, \dots, c\}.$$

denotemos por  $H$  la clase de clasificadores árbol (CART) que se describen abajo.

### Clasificador individual: Árbol de decisión CART

Sea  $T \in H$  un árbol binario de particiones recursivas del espacio  $\mathbb{R}^d$ .

- impureza de nodo: para un nodo que contiene el subconjunto

$S \subset D_N$  definimos

$$I(S) = \begin{cases} 1 - \sum_{c=1}^C \hat{P}_c^2 & \text{(índice Gini),} \\ - \sum_{c=1}^C \hat{P}_c \log \hat{P}_c & \text{(entropía de Shannon),} \end{cases}$$

$\hat{P}_c = \frac{1}{|S|} \sum_{(x,y) \in S} \mathbb{1}_{\{y=c\}}$

- criterio de división: en un nodo se considera un conjunto aleatorio  $F \subset \{1, \dots, d\}$  (ver §3) y, para cada atr. bufo  $j \in F$  y umbral  $s \in R$ , se evalúa.

$$\Delta I(j, s) = I(S) - \frac{|S_L|}{|S|} I(S_L) - \frac{|S_R|}{|S|} I(S_R),$$

donde  $S_L = \{(x, y) \in S : x^{(j)} \leq s\}$  y  $S_R = S \setminus S_L$ .

problema de optimización Local

$$(j^*, s^*) = \arg \max_{j \in F, s} \Delta I(j, s)$$

e) proceso continua recursivamente hasta que se verifica un criterio de parada (profundidad máxima, impureza mínima o número mínimo de ejemplos).

- predicción del árbol: cada hoja  $l$  almacena la distribución empírica

ii. l. Para  $x$  que cae en  $l(x)$ :  $T(x) = \arg \max_c \pi_L^*(c)$ .

## Aleatorización: Bootstrap + Sub-espacio de atributos

Para cada árbol  $m=1, \dots, M$ :

- Bootstrap: se genera una muestra  $D_N^{(m)}$  de tamaño  $N$  por remuestreo con reemplazo  $D_N$ .
- Selección de atributos por nodo: en cada nodo interno se elige uniformemente un subconjunto  $f^{(m)} \subset \{1, \dots, d\}$  de tamaño  $d_{try}$  (típicamente  $d_{try} \approx \sqrt{d}$ ). El criterio de división se restringe a  $f^{(m)}$ .

## Bosque aleatorio

Sean  $T_1, \dots, T_M$  i.i.d Q árboles entrenados como arriba (condicionados a  $D_N$ ). La función de voto del bosque es

$$\hat{\eta}_c^{(M)}(x) = \frac{1}{M} \sum_{m=1}^M \mathbb{1}_{\{T_m(x) = c\}}, \quad \hat{g}_M(x) = \arg \max_c \hat{\eta}_c^{(M)}(x).$$

## Sub-Problemas de optimización en Random forest

### Nivel - Nodo

Parámetro a ajustar: corte  $(j, s)$  (atributo + umbral)

Objetivo de la optimización:  $(j^*, s^*) = \arg \max_{j,s} \Delta I(j, s)$

Naturaleza del problema: Mixto (discreto - continuo). Suelta resolverse por búsqueda exhaustiva sobre un conjunto finito de umbrales  $s$ .

### Nivel - Árbol

Parámetros a ajustar: profundidad máxima, mínimo de muestras por hoja, poda costo-complejidad.

Objetivo de la optimización:  $\min_{\text{árbol}} I_{\text{tot}} + \lambda |H|$

Naturaleza del problema:  $|H|$

## Nivel - Bosque

parametros a ajustar: Número de árboles  $M$ , tamaño del sub-espacio de atributos  $d_{try}$ , uso de bootstrap/sub-muestra.

objetivo de optimización:  $(M^*, d_{try}^*) = \arg \min_{M, d_{try}} \widehat{R}_{CV}(M, d_{try})$   
(error out-of-bag).

Naturaleza del problema: optimización discreta basada en validación cruzada o estimador oob.

## Gaussian PROCESS classifier (GPC)

Datos:  $D_N = \{(x_i, y_i)\}_{i=1}^N$ , con  $x_i \in \mathbb{R}^d$  y  $y_i \in \{-1, +1\}$ .

objetivo: estimar la función de decisión  $g(x) = \text{sign}(f(x))$ , donde  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  es un proceso latente.

priori: proceso gaussiano sobre el latente

$$f(\cdot) \sim GP(0, K_\theta(\cdot, \cdot)),$$

donde •  $K_\theta(\cdot, \cdot)$  es un núcleo positivo-definido (P. ej. RBF isotrópico)  $K_\theta(x, x') = \sigma_f^2 \exp[-\frac{1}{2} \|x - x'\|^2 / \ell^2]$   
•  $\theta = (\ell, \sigma, f, \dots)$  son hiperparámetros a aprender.

para el conjunto de entradas  $X = [x_1, x_2, \dots, x_N]^T$ :

$$f := [f(x_1), \dots, f(x_N)]^T \sim N(0, K_{NN}), \quad K_{NN}[i, j] = K_\theta(x_i, x_j)$$

## Likelihood (clasificación)

se escribe una función de enlace sigmoidal  $\sigma(\cdot)$ :

- probit  $\sigma(t) = \Phi(t)$  (CDF de  $N(0, 1)$ )
- logística  $\sigma(t) = 1/(1 + e^{-t})$

Probabilidad condicional:

$$P(y_i | f_i) = \sigma(y_i, f_i), \quad P(y | f) = \prod_{i=1}^n \sigma(y_i | f_i).$$

## Posterior sobre el latente

$$\text{El posterior exacto } p(f|y, X, \theta) = \frac{p(y|f) N(f|0, K_{NN})}{p(y|X, \theta)}$$

No es gaussiano  $\Rightarrow$  se requiere aproximación.

## Laplace Approximation

Modo (MAP)  $f^* = \arg \max_f [\log p(y|f) - \frac{1}{2} f^T K_{NN}^{-1} f]$

Problema de optimización concavo (sigmoide log-concava).  
resuelto por Newton-Raphson.

### Aproximación gaussiana

$$p(f|y) \approx N(f^*, A^{-1}), \quad A = K_{NN}^{-1} + W$$

$$\text{con } W = \text{diag}(w_1, \dots, w_N) \text{ y } w_i = -\partial^2 \log \sigma(y_i; f_i) / \partial f_i^2 |_{f_i=f_i^*}$$

## Predictión

para un punto nuevo  $x_*$

distribución predictiva latente  $f_* | y \sim N(\mu_*, \nu_*)$ ,

$$\mu_* = K_{*}^T K_{NN}^{-1} f^*, \quad \nu_* = K_{**} - K_{*}^T (K_{NN}^{-1} - K_{NN}^{-1} A^{-1} K_{NN}^{-1}) K_{*},$$

$$\text{donde } K_{*} = [k_\theta(x_i, x_*)]_{i=1}^N \text{ y } K_{**} = k_\theta(x_*, x_*)$$

## Probabilidad de clase

$$\Pr(Y_* = +1 | x_*) = \int \sigma(f_*) N(f_* | \mu_*, \nu_*) df_*$$

integral univariada que admite forma cerrada para el probit  
y se evalúa numéricamente (cuadratura de Gauss-Hermite)  
para la logística.

## Decision

$$g(x_*) = \text{Sign}(2 \Pr(Y_* = +1) - 1)$$

Optimización de hiperparámetros - se maximiza la log-marginal likelihood aproximada (evidencia).

$$\text{Log } p(y|X, \theta) \approx \underbrace{\log p(y|f^*)}_{\text{verosim}} - \frac{1}{2} f^{*T} K_{NN}^{-1} f^*$$

$$- \frac{1}{2} \log |K_{NN}| + \frac{1}{2} \log |\Lambda|.$$

## Clasificadores basados en Deep Learning

Datos y Notación  $D_N = \{(x_i, y_i)\}_{i=1}^N, x_i \in \mathbb{R}^d, y_i \in \{1, \dots, C\}$

denotemos por  $f(x; w) : \mathbb{R}^d \rightarrow \mathbb{R}^C$

una red neuronal profunda (multicapa) parametrizada por pesos  $w$ .

Arquitectura abstracta Sea  $L$  el número de capas ocultas para la capa  $\ell = 1, \dots, L$ :

$$\begin{aligned} h^{(0)} &= x, \\ z^{(\ell)} &= w^{(\ell)} h^{(\ell-1)} + b^{(\ell)} \\ h^{(\ell)} &= \phi^{(\ell)}(z^{(\ell)}), \end{aligned}$$

donde: •  $w^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$  y  $b^{(\ell)} \in \mathbb{R}^{n_\ell}$

•  $\phi^{(\ell)}$  es la activación (ReLU, GELU, tanh, etc).

logits:  $z^{(L+1)} = w^{(L+1)} h^{(L)} + b^{(L+1)}$

Salida soft-max:

$$p_c(x, w) = \frac{\exp(z_c^{(L+1)})}{\sum_{k=1}^C \exp(z_k^{(L+1)})}.$$

## función de Perdida y Regularización

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^N [-\log p_{y_i}(x_i; w)] + \lambda \mathcal{J}_2(w)$$

- Primer término: Entropía cruzada (riesgo empírico).
- $\mathcal{J}_2(w)$ : Picamente  $\frac{1}{2} \sum_{\ell} \|w^{(\ell)}\|_F^2$  (Penalización L2) o norma L1.

- $\lambda > 0$ : hiperparámetro de regularización (weight-decay).

## Problema de optimización principal

$$w^* = \arg \min_w L(w).$$

Espacio paramétrico de alta dimensión (millones de entradas); el problema no es convexo.

## Algoritmos de entrenamiento

- gradiente estocástico:  $w_{t+1} = w_t - \eta_t \nabla_w \hat{L}_{B_t}(w_t)$ , donde  $B_t$  es un minibatch,  $\eta_t$  la tasa de aprendizaje.
- Momento de primer y segundo orden (Adam)
 
$$m_{t+1} = \beta_1 m_t + (1-\beta_1) \nabla w, \quad v_{t+1} = \beta_2 v_t + (1-\beta_2) (\nabla w)^2,$$
 y actualización con normalización  $(v_{t+1})^{1/2}$ .
- Back-propagation → uso de la regla de la cadena para propagar  $\frac{\partial L}{\partial z^{(l)}}$  hacia capas anteriores

## Estrategias de regularización adicionales

- Dropout ( $p$ ) → multiplica estocásticamente activaciones por Bernoulli ( $1-p$ ); en media equivale a un ensamble y añade término KL en la evidencia.
- Batch Normalization: Re-escaliza  $z^{(l)}$  para tener varianza unitaria; puede verse como reparametrización que suaviza el paisaje de  $L$ .
- Early-Stopping: minimiza  $L$  solo hasta que el error de validación alcanza mínimo: aproximación implícita a penalización sobre la norma de  $w$ .

## Propiedades de representación y capacidad

- Universalidad: redes con una capa oculta de tamaño finito  
Pueden aproximar funciones continuas en compactos ( $\epsilon$ -precisión)-  
Teorema de aproximación universal.

- Sobre parametrización:  $VC\text{-dim} \approx \sum_{l=1}^L (n_l n_{l-1})$  (cota superior); redes profundas pueden tener VC-dim mayor que el número de parámetros, debido a relu-composiciones.

## Garantías de generalización

- Margenes (Neyshabur et al.)

$$R(g) \leq \hat{R}_\gamma(w) + \tilde{O}\left(\frac{\|w\|_{2,1}^2 L}{\gamma^2 N}\right),$$

donde  $\gamma$  es margen de la última capa,  $\|w\|_{2,1}$  la norma producto.

- PAC Bayes: para distribución posterior  $Q$  sobre  $w$  (ruido

Gaussiano de varianza  $\sigma^2$ ):

$$R(Q) \leq \hat{R}(Q) + \sqrt{\frac{KL(Q||P) + \ln \frac{2\sqrt{N}}{\delta}}{2(N-1)}}$$

Sugiere que flat minima (bajo KL) generalizan mejor.

## Ejemplos de arquitecturas especializadas

Imágenes  $\rightarrow$  Red convolucional (CNN)  $\rightarrow$  Transformación base:  
 $f(x) = FC(Pool(\sigma(W^{(2)} * \sigma(W^{(1)} * x)))$

Secuencias  $\rightarrow$  Transformer  $\rightarrow$  T.base: atención escalada softmax  
 $(\frac{QK^T}{\sqrt{d}}) V$

Grafos  $\rightarrow$  GNN (GCN, GAT)  $\rightarrow$  Agregación  $h_v^{(k+1)} = \sigma(W^{(k)} @ \cup_{u \in N(v)} h_u^{(k)})$

Cada una se entrena bajo la misma pérdida de entropía cruzada pero con pesos y topologías distintas.