

Parcial 2: Teoría y aprendizaje de mayo 2015 - I

• Modelos de aprendizaje y máquina:

OPCA: Es una técnica de reducción de dimensionalidad que busca proyectar datos a una alta dimensión o en un espacio a menor dimensión preservando la mayor cantidad de variancia posible. Se utiliza para compresión, visualización y eliminación de ruido.

• Perspectiva generativa:

Se asume que los datos originales están en una matriz: $X \in \mathbb{R}^{N \times P}$ donde:

N : Número de muestras P : Número de características (dim. Org.)

En PCA se asume que los datos se generan desde un espacio de bajo dimension: $Z \in \mathbb{R}^{N \times M}$, con $M < P$ usando una matriz V de componentes principales $W \in \mathbb{R}^{P \times M}$, con V reconstruye la matriz original como: $X = ZVU^T$ esto implica que $Z = XW$

• Problema de optimización:

Se busca la matriz W que minimice el error de reconstrucción

$$W^* = \underset{W}{\text{arg min}} \|X - XWW^T\|_F$$

s.t $WW^T = I$, las columnas w_i deben ser ortogonales.

• Perspectiva de variancia:

Otra forma de ver PCA es como el proceso de encontrar una proyección que maximiza la variancia de los datos proyectados.

Dado que los datos estan centrados en la media, su matriz de covarianza es:

$$\Sigma = \frac{1}{N} X^T X$$

La nueva formulación del problema de optimización es:

$$W = \underset{w}{\operatorname{arg\,max}} \operatorname{tr}(w^T \Sigma w)$$

$$\text{s.t. } WW^T = I$$

Esto significa que, se esta buscando los M vectores propios de Σ (mutua y covarianza) que maximicen la variancia.

UMAP:

UMAP construye un ponderado entre puntos x_n y $x_{n'}$, usando una probabilidad y conectividad:

$$P_{nn'} = C \exp \left(-\frac{\|x_n - x_{n'}\|^2 - P_n}{\sigma_n} \right)$$

Lénh: • P_n : distancia al vecino más cercano a x_n .

• σ_n : Escala adaptativa según el número de vecinos.

Para evitar que una mutua asimétrica de similitudes, UMAP simetriza:

$$\tilde{P}_{nn'} = P_{nn'} + P_{n'n} - P_{nn'} P_{n'n}, \text{ esto produce un grafo ponderado, no dirigido}$$

En el espacio a baja dimensión (con puntos z_n), se define una probabilidad $q_{nn'}$ usando una función tipo t-student de escala medida:

$$q_{nn'} = (1 + \alpha \|z_n - z_{n'}\|^2)^{-1}$$

los parámetros a y b controlan la forma de la distribución; generalmente se fijan en 1.

UMAP minimiza la entropía cruzada:

$$C(p, q) = \sum_{n \in N} (-\tilde{p}_{nn} \log(\tilde{q}_{nn}) - (1 - \tilde{p}_{nn}) \log(1 - \tilde{q}_{nn}))$$

• Naive Bayes Gaussiano:

Es un clasificador supervisado probabilístico basado en el teorema de Bayes, con una suposición de independencia "naive" entre las características, y una suposición adicional que cada característica x_i sigue una distribución normal dentro de cada clase A .

• Fórmula general (con independencia)

$$P(x, A) = P(x|A) P(A) = \prod_{j=1}^p P(x_j|A) \cdot P(A)$$

- $X = (x_1, x_2, \dots, x_p)$: Vector de características

- A : Clase

- $P(A)$: Probabilidad a priori de la clase A

- $P(x_j|A)$: Verosimilitud (likelihood) de la característica j dado que la clase es A

• Predicción:

$$\hat{y} = f(x) = \arg \max \left[\prod_{j=1}^p P(x_j|A_c) \cdot P(A_c) \right]$$

se busca la clase A_c que maximiza la probabilidad posterior.

• SGD Classifier:

Es un clasificador supervisado que entraña un modelo lineal (como regresión logística o SVM) usando descenso de gradientes estocástico.

$$\text{Modelo: } f(x) = w^T x + b$$

- $x \in \mathbb{R}^p$: Vector de características
- $w \in \mathbb{R}^p$: Pesos del modelo
- $b \in \mathbb{R}$: Sesgo (bias)

Predictión: Dependiendo el signo de $f(x)$:

- Si $f(x) > 0$: predice la clase 1
- Si $f(x) < 0$: predice la clase 0

El modelo lineal forma una frontera de decisión que es un hipoplano que separa las clases

Problema de optimización: El SGD clasifica intentando minimizar la función de pérdida promedio sobre todos los datos:

$$\min_{w, b} \frac{1}{n} \sum_{i=1}^n l(y_i, f(x_i)) + \lambda R(w)$$

Dond:

- $l(y_i, f(x_i))$: Función de pérdida

- y_i : Etiqueta real de la muestra i

- x_i : Vector de características de la muestra i

- $f(x_i)$: Salida del modelo ($w^T x_i + b$)

- $R(w)$: función de regularización (ej. $L_2 = \|w\|^2$, $L_1 = \|w\|$)

- λ : Factor de regularización

Cómo se resuelve?: En lugar de calcular el gradiente sobre todos los datos (como el descenso del gradiente clásico); SGD hace lo siguiente:

1. Toma una muestra aleatoria (x_i, y_i)

2. Calcula el gradiente de la pérdida con esa muestra:

$$\cdot \nabla w_i = \partial l_i / \partial w \quad \cdot \nabla b_i = \partial l_i / \partial b$$

→ Se actualiza w y b

$$w \leftarrow w - n \cdot \nabla w_i, \quad b \leftarrow b - n \cdot \nabla b_i$$

Donde: α es la tasa de aprendizaje

- (i) es la pérdida en la muestra

• Regresión logística:

Es un modelo lineal de clasificación binaria, sirviendo para predecir la probabilidad de que la probabilidad de que sea una muestra pertenezca a una clase.

Modelo: Dado un vector de características $x \in \mathbb{R}^p$, queremos predecir:

$$P(y=1) = \sigma(w^T x + b)$$

Donde:

- $w \in \mathbb{R}^p$: Vector de pesos

- $b \in \mathbb{R}$: sesgo (bias)

• $\sigma(z) = \frac{1}{1+e^{-z}}$: Función sigmoid o logística

Esto da una salida entre 0 y 1, interpretada como una probabilidad.

Función de pérdida: log-loss (cross-entropy)

La función de pérdida para una muestra (x_i, y_i) con $g_i = \sigma(w^T x_i + b)$ es:

$$L(x_i, y_i) = -y_i \log(g_i) - (1-y_i) \log(1-g_i)$$

Esto penaliza fuertemente las predicciones que están lejos de la etiqueta correcta.

Problema de optimización: Para un conjunto de entrenamiento de n muestras

$$\min \frac{1}{n} \sum_{i=1}^n [-y_i \log(g_i) - (1-y_i) \log(1-g_i)] + \lambda \|w\|^2$$

• El primer término es la pérdida promedio (log-loss)

• El segundo término es la regularización (L_2), que controla el subajuste.

Direcciones (Gradientes): Para una sola muestra, con $z = w^T x + b$

$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad L(w, b) = -y \log(\sigma(z)) - (1-y) \log(1-\sigma(z))$$

- Gradientes:
- Respecto a w_i : $\nabla_w L = \sigma(z) - y - x$
 - Respecto a b : $\nabla_b L = \sigma(z) - y$

Se aplica la regla de actualización por descenso de gradientes aplicada en SGD (clasefic.)

• Linear discriminant analysis (LDA)

LDA es un clasificador supervisado lineal que:

- Asume que los datos de cada clase siguen una distribución gaussiana con la misma matriz de covarianza
- Busca proyectar los datos en una dirección que maximice la separación entre clases (discriminación)
- Sí basa en principios probabilísticos y genera una proyección lineal.

Supuestos claves:

- Cada clase K genera datos $X \in \mathbb{R}^d$ con una distribución normal multivariante, $X \sim N(\mu_K, \Sigma)$, donde μ_K es la media de la clase y Σ es la misma matriz de covarianza para todas las clases

- La matriz de covarianza es igual para todas las clases: $\Sigma_1 = \Sigma_2 = \dots = \Sigma$

- Se conoce o estimó la probabilidad a priori de cada clase $P_l(x)$

Modelo de LDA: LDA predice la clase como:

$$g = \log \max_K \delta_K(x)$$

Donde: $\delta_K(x) = x^T \Sigma^{-1} \mu_K - \frac{1}{2} \mu_K^T \Sigma^{-1} \mu_K + \log P_l(x)$

- La función $\delta_K(x)$ es lineal en x , por lo tanto LDA es un modelo lineal

• K Neighbors Classifier: Es un clasificador supervisado no-paramétrico. Esto significa que no aprende una función o modelo durante el entrenamiento, sino que guarda todo el conjunto de entrenamiento.

Cuando llega un nuevo punto x , el modelo:

1. Calcula la distancia entre x y todos los puntos del conjunto de entrenamiento.
2. Selecciona los K puntos más cercanos (los vecinos).
3. Asigna la clase más común entre los vecinos a x .

El modelo de KNN es simple, pero puede extenderse a la siguiente manera dado un conjunto de entrenamiento:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Y una nueva muestra x , la predicción es:

$$\hat{y} = \text{modo}\{y_i \mid x_i \in N_k(x)\}$$

Dónde:

- $N_k(x) \rightarrow$ Conjunto de los K vecinos más cercanos a x según alguna métrica (como distancia euclídea).
- modo \rightarrow Clase más frecuente entre los vecinos.

KNN no optimiza una función de costo explícita, pero puede entenderse como un método que minimiza el error de clasificación localmente.

$$\hat{y}(x) = \text{Odg max}_{\text{Clases}} \sum_{x_i \in N_k(x)} I(y_i = c)$$

Dónde:

- $I(y_i = c)$ es una función indicadora que vale 1 si el vecino x_i tiene clase c , y 0 si no.
- $\hat{y}(x)$ es la clase que KNN va a predecir para el punto x .
- $c \in \text{Clases}$: se evalúa cada clase posible ($y \in \{0, 1, \text{perro}, \text{gato}, \dots\}$)

- SVC: Support Vector Classifier

Es un clasificador supervisado que busca encontrar un hipoplano lineal que separe dos clases con el máximo margen posible.

Es una versión lineal del SVM (Support vector machine), pero implementado en un espacio más eficiente.

- El modelo lineal es:

$$f(x) = w^T x + b$$

- La predicción se hace como:

$$\hat{y} = \text{Sign}(w^T x + b)$$

Donde:

- w : Vector perpendicular al hipoplano (dirección al hipoplano)

- b : Sesgo (desplazamiento al hipoplano).

- x : Vector de características

• Cabe resaltar que el margen es la distancia entre el hipoplano y los puntos más cercanos a cada clase, llamados vectores de soporte.

- Problema de optimización (cujo separable)

- El problema es:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$\text{s.t. } y_i (w^T x_i + b) \geq 1, \text{ para todo } i$$

Donde:

- $w^T x_i + b$: Predicción del modelo

- y_i : Salida real

Si $y_i (w^T x_i + b) \geq 1$; significa que el punto está bien clasificado y suficientemente lejos al hipoplano (margen).

Maximizar el margen equivale a minimizar $\|w\|^2$.



- Problema de optimización (con datos no separables)

El problema es:

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

$$\text{s.t } y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- Dond:
- ξ_i : Cuánto se pierde que el punto se equivoca
 - C : hipiperámetro que controla el trade-off entre margen

Otros tipos de clasificación.

- Random Forest classifier:

El random forest es un ensamble de árboles de decisión que combina múltiples árboles para mejorar la generalización y robustez del clasificador. cada árbol es una sola decisión independiente usando:

- Bootstrap (muestra aleatorias con reemplazo)
- Selección aleatoria de características en cada división:

El modelo consiste en un conjunto de T árboles de decisión:

$$\text{Random Forest classifier: } \{h_1(x), h_2(x), \dots, h_T(x)\}$$

Cada $h_t(x)$ es un árbol que predice una clase para una muestra x . la predicción final se hace por votación mayoritaria (pura clasificación):

$$\hat{y} = \text{mode}(\{h_t(x)\}_{t=1}^T)$$

- Problema de optimización:

Aunque no optimiza una función global explícita, cada árbol individual resuelve un problema de optimización greedy local al construir su estructura. Para cada nodo del árbol, se busca la mejor división



h los datos que maximice la ganancia de información.

• Problema de optimización por nodo:

En cada nodo h un árbol, el algoritmo elige la característica j y el umbral θ que maximizan alguna medida de impureza:

• Gini impurity: $G(s) = 1 - \sum_{k=1}^K p_k^2$

• Entropy: $H(s) = - \sum_{k=1}^K p_k \log(p_k)$, p_k : Proporción de clase k en el nodo

• Puro de optimización local

Para cada división candidata (c_j, θ) , se evalúa:

$$Gini(c_j, \theta) = \text{Impureza}(s) - \left(\frac{|S_L|}{|S|} \text{impureza}(S_L) + \frac{|S_R|}{|S|} \text{impureza}(S_R) \right)$$

se selecciona (c_j^*, θ^*) que maximiza esa ganancia.

• Gaussian Process Classifier

Es un modelo bayesiano no paramétrico que asume que la función de función subyacente es una multiplicación de un proceso gaussiano.

Esto lo hace especialmente útil cuando se requiere una estimación bien calibrada de la incertidumbre.

• Problema de optimización:

Iniciar la distribución posterior al vector f inicial

$f = [f(x_1), \dots, f(x_N)]^T$, y predecir en nuevos puntos.

para 1: Hacer la predicción

$$p(y|f) = \prod_{i=1}^N \Phi(y_i | f_i) \quad (\text{esto usando } \psi(\{-1, +1\}) \text{ para simplificar})$$

Paso 2: Primera aproximación sobre f

$f \sim N(0, \kappa)$ donde $\kappa \in \mathbb{R}^{N \times N}$ es la matriz de covariancia del ruido ϵ . (x_i, x_j)

Como la función posterior: $p(f|x,y) \propto p(y|f) \cdot N(f|0, \kappa)$ no es gaussiana

(por la función sigmoid), no se puede calcular exactamente.

Para esto se approxima por Laplace, así:

se approxima el posterior $p(f|x,y)$ por una Gaussiana ajustada alrededor del nodo de posterior (MAP)

Como tal el problema de optimización:

$$f^* = \arg \max_f \log p(y|f) - \frac{1}{2} f^\top \kappa^{-1} f$$

- Primer término: log-verosimilitud (ejm: logit o probit)

- Segundo término: regularización por la prior gaussiana.

Este problema se resuelve usando métodos de optimización o segundo orden como Newton-Raphson o Laplace method.

- Clasificadores basados en Deep Learning:

- CNN (Convolutional neural network)

El objetivo es aprender una función de clasificación:

$$f_\theta: \mathbb{R}^{1 \times 16 \times 16} \rightarrow \{0, 1, \dots, 9\}$$

Donde:

- $x_i \in \mathbb{R}^{1 \times 16 \times 16}$: imagen en escala de grises de 16x16 pixeles

- $y_i \in \{0, 1, \dots, 9\}$: Clase verdadera (etiqueta) de la imagen

- $f_\theta(x)$: red neuronal convolucional parametrizada por θ , que produce logits (puntuaciones sin normalizar) para cada clase.

La salida de la red se convierte en probabilidades con softmax:

$$\hat{y}_i = \text{softmax}(\theta(x_i)) \in [0, 1]^D$$

Problema de optimización

El entrenamiento busca minimizar la función de pérdida de entropía cruzada (cross-entropy) sobre el conjunto de entrenamiento:

$$\theta^* = \underset{\theta}{\operatorname{arg\min}} \quad L(\theta) = - \sum_{i=1}^N \log \hat{y}_{i,i} = - \sum_{i=1}^N \log \left(\frac{e^{z_{i,i}} y_i}{\sum_{l=1}^D e^{z_{i,l}}} \right)$$

Donde:

- $\hat{y}_{i,i}, y_i$: probabilidad predicha para la clase verdadera.
- $z_{i,l}$: logit (output unts d softmax) para clase l .
- θ : todos los parámetros (pesos y sesgos) del modelo.
- N : # de ejemplos en el conjunto de entrenamiento.

Evaluación del clasificador

Después del entrenamiento, el modelo es evaluado en el conjunto de prueba mediante:

- Exactitud (Accuracy): $\text{Acc.} = \frac{\# \text{ predicciones correctas}}{N_{\text{test}}}$
- F1-score macro: media del F1 por clase
- Matriz de confusión: distribución de errores por clase
- Curva ROC (macro-avergae OvR): desempeño de clasificación multiclas en términos de sensibilidad vs especificidad.