

a.) Consulte y presente el modelo y problema de optimización de los siguientes modelos de aprendizaje de máquina.

- PCA

• **Modelo matemático:** Dada una matriz de datos centrados

$X \in \mathbb{R}^{n \times p}$, PCA busca una transformación ortogonal $W \in \mathbb{R}^{p \times m}$

que proyecte los datos a un espacio de menor dimensión:

$$Z = XW$$

• **Problema de optimización:** Maximizar la varianza de los datos proyectados:

$$\max_w \text{tr}(W^T \Sigma W)$$

Sujeto a: $W^T W = I_m$

donde $\Sigma = \frac{1}{n} X^T X$ es la matriz de covarianza muestral

Esto significa que se está buscando los m vectores propios que maximicen la varianza.

- UMAP Es un método de reducción de dimensionalidad no lineal que preserva tanto la estructura global como local de los datos. Modela las relaciones entre puntos en el espacio original y el espacio de baja dimensión mediante teorías de variedades riemannianas y topología algebraica

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

Donde: ρ es la distancia entre los vecinos y $s_{\text{distancia}}$ es el tamaño de la muestra.

$P_n = \text{Distancia al vecino mas cercano a } x_n$

$\theta_n = \text{Escala adaptativa segun el numero de vecinos}$

Para evitar que una matriz simétrica de L sea singular,

UMAP simétrica:

$\tilde{P}_{nn} = \tilde{P}_{nn} + P_{nn} - P_{nn}P_{nn}$, esto produce un grafo ponderado no dir. gdp.

En el espacio de baja dimensión se define una probabilidad q_{nn} usando una función de tipo t-student de cota pesada:

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

los parámetros α y β controlan la forma de la dist. louron

$$(P; Q) = \sum_{n \neq n} (-\tilde{P}_{nn} \log(q_{nn}) - (1 - \tilde{P}_{nn}) \log(1 - q_{nn}))$$

- Naive Bayes Gaussian (GaussianNB)

Es un clasificador probabilístico basado en el teorema de Bayes, el cual asume independencia condicional entre los características dado el valor de la clase. Para cada clase, modelo las características con distribuciones normales. formula con independencia:

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

Donde:

$X = (X_1, X_2, \dots, X_p)$ → Vector de características

A = Clase

$P(A)$ = Probab. l. da a pri. or. de la clase pri. or.

$P(X_j|A)$ = Veros.m. l. luid de la caract. st. ca

Predicción:

$$\hat{y} = f(x) = \operatorname{argmax}_{A_c} \left[\sum_{j=1}^p P(X_j|A_c) \cdot P(A_c) \right]$$

Se busca la clase A_c que max.miza la probab. l. dad poster. or.

- SGD Class. fer: Es un clasificador supervisado que entrena un modelo linal. Actualiza los parámetros iterativamente usando un subconjunto de datos en cada paso.
- Modelo matemát. co: $f(x) = W^T x + b$

Donde: $x \in \mathbb{R}^p$ = Vector de características

$W \in \mathbb{R}^p$ = Pesos del modelo

$b \in \mathbb{R}$ = Sesgo (bias)

la predicción depende del signo de $f(x)$: - S:

- S: $f(x) > 0$ = predicción 1
 $f(x) \leq 0$ = predicción 0

El modelo traza una frontera de decisión el cual es un hiperparámetro que separa las clases

- Problema de optimización: El SGD classifier intenta minimizar la función de pérdida promedio sobre todos los datos:

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

Donde: L = función de pérdida

y_i = Etiquetado de la muestra i

x_i = Vector de características de la muestra j

$f(x_i)$ = Salida del modelo

$R(w)$ = Término de regulación

λ = Precio de regulación.

- RandomForestClassifier: Ensemble de árboles de decisión donde cada árbol se entrena con un subconjunto aleatorio de datos y características. La predicción final es el voto mayoritario de los árboles.

Cada árbol n entrena de manera independiente usando:

- Bootstrap: (muestras aleatorias con reemplazo)
- Selección aleatoria de características en cada decisión.

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

$$\text{Randomforest Classifier} = \{h_1(x), h_2(x), \dots, h_T(x)\}$$

Cada $h_T(x)$ es un árbol que predice una clase para una muestra x . La predicción final se hace por votación mayoritaria.

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

• Problema de Optimización: Aunque no optimiza una función global explícita, cada árbol s. incluye un problema de optimización greedy local al construir su estructura.
Para cada nodo del árbol se busca la mejor distribución de los datos que maximice la ganancia de la información.

- Problema de Optimización por nodo:

En cada nodo de un árbol, el algoritmo elige las características j y el umbral θ que maximiza la medida:

$$\text{Gra. impur. t4} = G(S) = 1 - \sum_{k=1}^{2^K} P^{2^k}$$

$$\text{Entropy} = H(S) = - \sum_{k=1}^C P_k \log(P_k), \quad P_k = \text{Proporción de clases en el nodo.}$$

- Gaussian Processes Classif. Es un clasif. r. o da basado en procesos gausianos que modela la distribución a posteriori de las funciones discutidas. Esto lo hace especialmente útil cuando se requiere una estimación bien calibrada de la incertidumbre.

- Problema de optimización: Maximiza la verosimilitud marginal.

Paso 1: Modelo de predicción

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

Paso 2: Prv:or. gausiana sobre f

$f \sim N(0, K)$ donde K pertenece $\mathbb{R}^{N \times N}$ es la matriz de covarianza

Como la función posterior: $P(x_n | f) \propto P(y | f) \cdot N(f | \mu, \sigma^2)$
no es gaussiana (por la función sigmoidal no se puede calcular exactamente).

Pero podemos aproximar por búsquedas de lo siguiente modo:

Se approxima el posteror: $P(f | x, y)$ por una gaussiana ajustada alrededor del nodo y posteror: (MAP)

Como tal el problema de optimización quedará:

$$f^* = \arg \max_f \log P(y | f) - \frac{1}{2} f^T \Sigma^{-1} f$$

- Primer término: log-verosimilitud (sgm)
- Segundo término: regularización por la prior gaussiana.
- Clas. f. nodo basado en deep learning (CNN)

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 16×16 pixels
- * $y_i \in \{0, 1, \dots, 9\}$: Clase verdadera (etiqueta) de la imagen

A $f_\theta \in$: red neuronal convolucional parametrizada por θ para cada clase

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

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

Problema de Optimización

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

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^N \log \hat{y}_i, y_i = - \sum_{i=1}^N \log \left(\frac{e^{z_i, y_i}}{\sum_{t=1}^n e^{z_i, t}} \right)$$

donde:

\hat{y}_i, y_i : Probabilidad predicha para la clase verdadera.

z_i, t : logit (output de softmax)

θ : Todos los parámetros (pesos y sesgos) del modelo

N : Número de ejemplos en el conjunto de entrenamiento.

Evaluación del clasificador

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

$$\text{Accuracy} = \frac{\text{Acc. # de predicciones correctas}}{N_{\text{test}}}$$

* F_1 -score: media de f_1 por clase

* Matriz de confusión: distribución de errores por clase

Evaluación del Clasificador.

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

* (Accuracy): $\frac{\text{Acc. } \# \text{ de predicciones correctas}}{N_{\text{test}}}$

* f_1 -score macro: media de f_1 por clase.

* Matriz de confusión: distribución de errores por clase

* Curva ROC (macro-average OvR): desempeño de clasificación multiclase en términos de sensibilidad vs especificidad

Logistic Regression

es un modelo que se utiliza para clasificación binaria. modela la probabilidad de que una muestra pertenezca a una clase usando la función sigmoidal.

$$P(Y=1|X) = \frac{1}{1+e^{-\theta^T X}}$$

Método de optimización

El objetivo es encontrar los parámetros θ que maximicen la verosimilitud de los datos, lo que es equivalente a minimizar la entropía cruzada binaria (log-loss negative)

$$\min_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m [y_i \log(h_\theta(x_i)) + (1-y_i) \log(1-h_\theta(x_i))]$$

Linear Discriminant Analysis (LDA)

LDA es un clasificador lineal que proyecta datos en un subespacio de menor dimensión de manera que se maximicen la separación entre las clases. Asumiendo que todas las clases comparten la misma matriz de covarianza. Se basa en el teorema de Bayes para estimar la probabilidad posterior y asignar una clase a una nueva observación x .

$$P(x|Y=k) \sim N(\mu_k, \Sigma)$$

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

Problema de optimización:

LDA no se entrena por gradientes sino que estima los parámetros para maximizar verosimilitud

- estimar Matrix M
- estimar Matrix de Covarianza Compartida Σ
- estimar Proporciones T_{ik}

También puede formularse como un problema de maximizar la varianza de varianza entre clases y dentro de clases (criterio fisher)

$$\underset{w}{\text{Max}} \ J(w) = \frac{w^T S_B w}{w^T S_w w}$$

K-Nearest Neighbors Classification:

es un modelo no paramétrico de aprendizaje basado en instancias (lazy learning).

En lugar de construir un modelo explícito durante el entrenamiento almacena el conjunto de entrenamiento completo y realiza la predicción mediante la búsqueda de los K vecinos más cercanos a un punto de cálculo (aproximando usando distancia euclídea)

$$\hat{y} = \text{mod}(\{ y_i | x_i \in kNN(x) \})$$

Problema de optimización

No tiene un problema de optimización como tal. Sin embargo, el "entrenamiento" consiste en almacenar los datos y posteriormente preprocesar (normalización, reducción dimensional)

La única optimización es seleccionar el hiperparámetro K y las métricas de distancia mediante validación cruzada (cross validation)

SVC - Support Vector Classification

es un clasificador supervisado que encuentra un hipoplano de separación óptima entre clases, maximizando el margen entre los ejemplos de diferentes clases.

$$f(x) = \mathbf{w}^T \mathbf{x} + b$$

Problema de optimización

Caso lineal (soft margin)

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

sujeto a: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

ξ_i : variables de holgura (permiten errores)

C : Parámetros de penalización

Caso no lineal usa Kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ se resuelve el problema de forma dual

$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

sujeto a: $0 \leq \alpha_i \leq C \quad \sum_i \alpha_i y_i = 0$

Un vez resuelto, el clasificador se expresa como

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

con \mathbf{x}_i son vectores de soporte