Víctor Quintero T. g. 976.089.

1. PCA (Principal Component Analysis)

Modelo: Busca una proyección lineal ortogonal $z = W^T x$

Problema de optimización: Encontrar la matriz de proyección W que maximiza la varianza de los datos proyectados

$Var(z) = W^T S W$ donde S es la matriz de covariante con $W^T W = I$.

Usando un multiplicador de Lagrange λ:

$$\mathcal{L}(W, \lambda) = W^T S W - \lambda(W^T W - 1)$$

Tomando la derivada con respecto a W e igualando a $\emptyset$

$$\frac{\partial \mathcal{L}}{\partial W} = 2SW - 2\lambda W = 0$$

$$SW = \lambda W$$

$$Var(z) = W^T S W = W^T \lambda W = \lambda W^T W = \lambda$$

2. UMAP (Uniform Manifold Approximation and Projection).

Modelo: Método no lineal que busca preservar la estructura topológica de los datos.

Problema de optimización: Minimizar la entropía cruzada entre las similitudes en el espacio de alta dimensión (P_{ij}) y la del espacio de baja dimensión (q_{ij})

$$\text{Min } CE(P, Q) = \sum_{i,j} \left[P_{ij} \log\left(\frac{P_{ij}}{q_{ij}}\right) + (1 - P_{ij}) \log\left(\frac{1 - P_{ij}}{1 - Q_{ij}}\right) \right]$$

Usa SGD (Descenso de Gradiente Estócastico) para encontrar las coordenadas de los puntos en el espacio de baja dimensión que minimizan esta divergencia.

Naive Bayes (GaussianNB).

Modelo: Modelo generativo basado en el teorema de Bayes con la suposición "ingenua" de I.I.D, independencia condicional de las características dada la clase.

$$p(y=k|x) \propto p(y=k) \prod_{i=1}^{p} p(x_i|y=k)$$

los parámetros a estimar son la media μ_{kc} y la varianza σ^2_{kc} para cada característica k y clase c. La media y la varianza muestral calculadas a partir de las muestras de entrenamiento que pertenecen a la clase c.

SGDClassifier.

Modelo: Método de optimización para modelos lineales como Regresión Logística o SVC lineal.

Problema de Optimización: Minimiza una función de pérdida $L(w)$ de forma iterativa. En cada paso, actualiza los pesos w usando el gradiente de una única muestra ó mini-lote.

$$w_{t+1} = w_t - \eta \nabla L(w_t; x_n; y_n)$$

Para un modelo de regresión con pérdida de error cuadrática

$$L = \frac{1}{2}(y_n - w^T x_n)^2$$

el gradiente es:

$$\nabla L = (y_n - w^T x_n) x_n$$

La regla de actualización es:

$$w_{t+1} = w_t + \eta (y_n - w_t^T x_n) x_n.$$

Regresión Logística

Modelo: En un modelo discriminativo que modela la probabilidad a posteriori de la clase k usando softmax.

$$p(y=k \mid x, W) = \frac{\exp(w_k^T x)}{\sum_{j=1}^{K} \exp(w_j^T x}$$

donde W es la matriz de pesos.

Problema de optimización: Minimizar la entropía cruzada negativa.

$$L(W) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln(y_{nk})$$

donde $y_{nk} = p(y=k \mid x_n, W)$.

Se busca el gradiente de L con respecto a los pesos de una clase j, w_j. El gradiente de softmax es:

$$\frac{\partial y_{nk}}{\partial w_j} = y_{nk}(\delta_{kj} - y_{nj}) x_n.$$

Sustituyendo y simplificando la derivada tenemos:

$$\frac{\partial L}{\partial w_j} = \sum_{n=1}^{N} (y_{nj} - t_{nj}) x_n$$

Aplicando Descenso de Gradiente:

$$w_j^{(new)} = w_j^{(old)} - \eta \frac{\partial L}{w_j}$$

LDA (Linear Discriminant Analysis).

Modelo: Es un modelo generativo. Asume que la densidad de probabilidad de cada clase k, $p(x|y=k)$ es Gaussiana. Todas las clases comparten la misma matriz de covarianza Σ pero tiene diferentes medias μ_k.

$$p(x|y=k) = \mathcal{N}(x|\mu_k, \Sigma)$$

Problema de optimización: Encontrar la proyección $y = w^T x$ que maximiza el criterio de Fisher:

$$J(w) = \frac{\text{Varianza inter-clase}}{\text{varianza intra-clase}} = \frac{w^T S_B w}{w^T S_W w}$$

derivando $J(w)$ con respecto a w e igualando a $\emptyset$

$$S_B w = J(w) S_W w$$

Reescribiendo como $(S_W^{-1} S_B) w = J(w) w$

K Neighbors Classifier (KNN).

Modelo: KNN es un modelo no paramétrico y basado en instancias (o "lazy learning")

Problema de optimización: Para una nueva muestra X_{new} se calcula la distancia entre X_{new} y cada una de las muestras X_n.

Se identifican las K muestras cuya distancias son las más pequeñas. Se calcula la moda y se etiquetan los vecinos.

SVC (Support Vector Classifier).

Modelo: Es un hiperplano que actúa como límite de decisión. Este se define por $w^T \phi(x) + b = 0$, donde w es el vector de pesos, b el sesgo y $\phi(x)$ es una función que mapea los datos de entrada a un espacio de mayor dimensión.

Problema de optimización: Encontrar el hiperplano que maximiza el margen y minimiza el error de clasificación ponderado por un hiperparámetro de regularización C.

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

$$\text{sujeto a } y_n(w^T\phi(x_n)+b) \geq 1 - \xi_n,$$

$$\text{y } \xi_n \geq 0, \ \forall n.$$

Random Forest Classifier

Modelo: Es un modelo de ensamble que consiste en una colección de múltiples árboles de decisión.

Problema de optimización.: Es un proceso local y codicioso aplicado a cada árbol que resuelve en cada nodo:

Encontrar la característica j y el umbral t que producen la división que maximiza la pureza de los nodos hijos.

esto es la maximización de la ganancia de información o la minimización de la impureza:

$$Gini(S) = 1 - \sum_{k=1}^{K} p_k^2$$

Gaussian Process Classifier.

Modelo: Es un modelo Bayesiano no paramétrico. Se especifica mediante la función de media $m(x)$ y una función de covarianza o kernel $k(x, x')$

$$f(x) \sim GP(m(x), k(x, x'))$$

donde:

$$p(y = 1 | x) = \sigma(f(x)).$$

Problema de optimización: Calcular la distribución predictiva posterior. Encontrar una aproximación tratable a la distribución posterior $p(f | X, y)$:

$$p(f | X, y) = \frac{p(y | f)\, p(f | X)}{p(y | X)}.$$

Clasificadores basados en deep learning.

Modelo: Composición de múltiples capas de transformaciones no lineales: $a_l = g(W_l a_{l-1} + b_l)$

Problema de optimización: Minimizar la función de pérdida (típicamente entropía cruzada) sobre un gran número de parámetros (W_l, b_l).

Se calcula a través de Backpropagation:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)}$$

donde $\delta_j^{(l)}$ es el error de la neurona j de la capa l, que se propaga de la capa de salida. Se realiza con Descenso de gradiente estocástico.