Carpeta 4, Cuaderno 6

◆ Ejercicio 1. Consulte y presente el modelo y problema de optimización de los siguientes clasificadores

✳ ✳ Naive Bayes Gaussian

Se suponen datos i.i.d y distribución normal, donde la función de probabilidad esta dada como:

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

Para clasificar una nueva instancia x se calcula la probabilidad posterior y se usa el teorema de Bayes,

$$P(y|x) = \frac{P(y)\prod_i P(x_i|y)}{P(x)}$$

Problema de optimización.
Se estiman los parametros $\mu_{i,y}$ y $\sigma_{i,y}^2$

La media como media muestral, así:

$$\mu_{i,y} = \frac{1}{N_y}\sum_{x_i \in \text{clase } y} x_i \quad ; \; N_y = \text{total de muestras en } y$$

La varianza de cada caracteristica x_i

$$\sigma_{i,y}^2 = \frac{1}{N_y}\sum_{x_i \in \text{clase } y} (x_i - \mu_{i,y})^2$$

No hay un proceso explicito de optimización en cuanto a minimizar una función de costo.
El modelo estima los parametros de la distribución gaussiana para cada caracteristica y clase a partir de los datos.

✳ ✳ SGD Classifier

Es una implementación de la maquina de vectores de soporte (SVM) y de otros clasificadores lineales mediante la técnica de descenso de gradiente estocastico (SGD)

Al ser un modelo lineal, predice la clase de x como una función lineal, así:

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

Permite ajustar varios tipos de modelos, incluyendo:

- SVM. Maximiza el margen entre clases
- Regresion Logistica. Clasificacion binaria y multiclase
- Hinge loss. Clasificacion SVM
- Log loss. Para regresion logistica
- Perceptron. Clasificacion lineal

Problema de optimización
Utiliza el metodo de descenso de gradiente estocástico, que es eficiente para grandes conjuntos de datos y puede manejar datos de flujo.

La función de perdida depende del modelo, los mas comunes son:

$$\text{Hinge loss} = \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

$$\text{Log loss} = -\sum_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Donde, p_i = Probabilidad predicha de x_i
y_i = Etiqueta de x_i

Descenso de gradiente estocastico
Actualiza los parametros del modelo en cada iteracion utilizando un subconjunto aleatorio (mini-lote) de los datos de entrenamiento.

La regla de actualización básica es:

$$w \leftarrow w - \eta \nabla_w \text{loss}(w, x, y)$$

Donde, η = tasa de aprendizaje
∇_w Loss = gradiente de la función de perdida respecto a los pesos

Regularización
Para evitar sobreajuste, los mas comunes son:

- L1 Regularizacion (Lasso) . Proporcional a la suma de los valores absolutos de los pesos
- L2 Regularizacion (Ridge) . Proporcional a la suma de los cuadrados de los pesos

La función de perdida con regularización se puede escribir como:

$$\text{Loss}_{\text{total}} = \text{Loss}_{\text{original}} + \lambda \|w\|_2^2$$

Donde, λ = parametro de regularización y $\|w\|_2^2$ es la penalización L2

Logistic Regression

Predice la probabilidad de que x pertenezca a una etiqueta especifica
La probabilidad de la clase positiva se modela mediante la función Sigmoide (o logística):

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

Función de perdida (Pérdida logistica o Cross entropy loss)

$$Log\ loss = -[Y \log(P) + (1-Y)\log(1-P)]$$

Donde, P = Probabilidad predicha para la clase positiva
Y = Etiqueta verdadera

Clasificación multiclase (varias etiquetas)
Se utiliza una extension llamada regresion logistica multinomial, se predice una probabilidad para cada clase usando la función Softmax

$$P(Y=K|X) = \frac{\exp(w_K^T x + b_K)}{\sum_j \exp(w_j^T x + b_j)}$$

Donde, k = clase especifica
$\sum_j$ = sobre el total de clases

Problema de optimización
Consiste en encontrar los parámetros W y b que minimicen la funcion de perdida

Funcion de perdida total para un conjunto de datos

$$Log\ loss_{total} = -\frac{1}{N}\sum_{i=1}^{N}[Y_i \log(P_i) + (1-Y_i)\log(1-P_i)]$$

Donde, N = total de datos
Y_i = etiqueta verdadera

Logistic Regression utiliza la misma regularización de SGDClassifier

Optimización

Se realiza tipicamente mediante métodos de optimización como:

Descenso de gradiente (Estocastico)
Newton's method
L-BFGS (Limited-memory Broyden-fletcher-Goldfarb-Shanno)

** Linear Discriminant Analysis

LDA se basa en el principio de encontrar una combinacion lineal de las caracteristicas que mejor separa las clases de datos

Función de Separación
LDA busca encontrar eigenvectores y eigenvalores que corresponden a las direcciones de maxima separación entre clases.

La función objetivo es maximizar el cociente:

$$J(W) = \frac{W^T S_B W}{W^T S_W W}$$

Donde, S_B = Matriz de covarianza entre clases (variabilidad entre clases)
S_W = Matriz de covarianza entro de clases (Variabilidad en cada clase)

La dimensionalidad del espacio proyectado es $C-1$, donde C es el total de clases
En el espacio proyectado, LDA utiliza un clasificador lineal para asignar nuevas instancias a las clases

Problema de optimización

Cálculo de las matrices de covarianza

$$S_B = \sum_{c=1}^{C} N_c (\mu_c - \mu)(\mu_c - \mu)^T$$

$$S_W = \sum_{c=1}^{C} \sum_{x_i \in C_c} (x_i - \mu_c)(x_i - \mu_c)^T$$

Donde, N_c = total de muestras en la clase C
μ = media global de todas las clases

Se convierte en encontrar los vectores propios w y los valores propios que maximizan la razón:

$$S_B W = \lambda S_W W$$

Reducción de dimensionalidad:
Los datos originales se proyectan en el espacio definido por los vectores propios, se seleccionan los vectores propios correspondientes a los mayores valores propios

Clasificación:
En el espacio reducido, se usa un clasificador lineal para asignar nuevas instancias a las clases

✳ ✳ KNeighborsClassifier

Clasifica una nueva instancia basandose en las etiquetas de los K vecinos más cercanos en el espacio de caracteristicas

Distancia
La distancia más utilizada es la distancia euclidiana

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

Otras funciones de distancia son:

- Distancia de Manhattan (L1)
- Distancia de Minkowski

Problema de optimización
No tiene problema de optimización tradicional

KNN implica:

- Procesamiento de datos _ Normalización / Estandarización (Es importante que las caracteristicas tengan escalas similares)
- Selección del hiperparametro K _ Generalmente se utiliza validación cruzada para seleccionar el valor óptimo de K que minimiza el error de clasificación en el conjunto de validación
- Distancia y metodo de busqueda _ Para bases de datos grandes, se usa estructuras de datos eficientes como arboles KD o búsqueda de vecinos aproximada

✳ ✳ Linear SVC

Se utiliza para encontrar un hiperplano en el espacio de caracteristicas que separa las clases con el margen más amplio posible

Problema de optimización
Se busca maximizar el margen entre las clases, mientras se minimiza la penalización por errores de clasificación, La función de pérdida es la hinge loss con regularización

La función de pérdida que se minimiza es la pérdida de margen con regularización L2

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i(w^T x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

Optimización
Linear SVC utiliza el algoritmo de optimización de coordenadas y metodos de Gradiente para resolver el problema de optimización

** SVC

Puede manejar tanto problemas lineales como no lineales mediante el uso de diferentes núcleos (Kernels)

Problema de optimización
Parametros a fijar

- W (vectores de pesos)
- b (término de sesgo)
- C (Parámetro de regularización)

Para el caso lineal

$$L(w,b) = \frac{1}{2}\|w\|^2 - C\sum_{i=1}^{N} \xi_i$$

Sujeto A;

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad ; \quad \xi_i \geq 0$$

Donde, ξ_i = Variables de holgura que permiten cierta flexibilidad en la separación de clase

Para el caso no lineal
utiliza una función de núcleo $K(x_i, x_j)$ que transforma los datos a un espacio de características de mayor dimensión

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

Sujeta a:

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

Donde, α_i = Son los multiplicadores de lagrange asociado con cada punto de datos.

Hay varias Kernels que se utilizan: Lineal, Polinomico, RBF, Sigmoide

** RandomForestClassifier

Combina múltiples árboles de decisión para realizar la clasificación. Utiliza la técnica de bosque aleatorio (Random forest), Cada arbol realiza una predicción y la clase final es determinada por la mayoría de votos de todos los árboles.

Se basa en dos técnicas principales

- Bootstrap Aggregation (Bagging): Cada árbol se entrena con una muestra aleatoria del conjunto de datos de entrenamiento, mediante muestreo con reemplazo
- Aleatorización en la construcción de arboles: en cada nodo de un árbol solo se considera un subconjunto aleatorio de las características para encontrar el mejor split.

Problema de optimización
Minimizar el error de clasificación en el conjunto de datos de prueba mediante la combinación de múltiples árboles de decisión.

Parámetros a ajustar

- Numero de arboles (n_estimators). un numero mayor de árboles mejora la precision pero aumenta el tiempo de computación
- Numero maximo de características. (mejor division en cada nodo)
- Profundidad maxima. (evitar sobre ajuste)
- Número mínimo de muestras para dividir un nodo. (previene arboles sobre ajustados)
- Número mínimo de muestras en un nodo hoja (reducir sobre ajuste)

Para cada árbol en el bosque

Optimizar función de costo = Gini impurity o Entropía

$$\text{Gini Impuriti} = 1 - \sum_{i=1}^{K} P_i^2$$

$$\text{Entropia} = -\sum_{i=1}^{K} P_i \log(P_i)$$

Donde, P_i proporción de muestras de la clase i en el nodo