TALLER 29 may - 5 jun

CV Clasificación LFW:

1) Naive Bayes Gaussian:

Red Bayesiana simple, donde Y(Clase) es el nodo padre de un conjunto X_i que representa una evidencia. Si tenemos un vector de evidencia $X=(X_1, X_2, \dots, X_d)$ determina la prob. de que pertenezca a una de las clases y, valores que puede tomar Y.

Distribución Gaussiana de datos. Por medio de train estimamos μ_y y θ_y^2 de cada clase.

Se empieza guardando Priors en una clase, preestablecidos o estimados con clases. Los Priors responden a las tablas de prob. para Y, $p(y)$. También se estima tabla condicional para cada X_i, que tienen padre a Y. Así $P(X_i|Y)$, gaussianas por eso: μ_y y θ_y^2 ; así:

$$p(X_i = x_i \mid Y = y) = \frac{1}{\sqrt{2\pi\theta^2}} e^{\left(\frac{-x_i - \mu_y}{2\theta_y^2}\right)^2}$$

Probabilidad conjunta:

$$P(Y=y, X_1=x_1, \dots, X_d=x_d) = P(Y=y) \prod_{i=1}^{d} P(X_i = x_i \mid Y=y)$$

Decidir sobre clase, se elegirá la clase que maximice la probabilid. estimada:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(Y=y, X_1=x_1, \dots, X_d=x_d)$$

SGDClassifier:

Un método de optimización. Para minimizar un loss, SGD actualiza los parámetros de forma iterativa

Dataset:

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

Objetivo encontrar un vector de pesos $w \in \mathbb{R}^d$ y bias $b \in \mathbb{R}$ que minimice la pérdida logística regularizada:

- Logistic Loss (Clasificación binaria):

$$P(y_i \mid x_i) = \frac{1}{1 + e^{(-y_i(w^T x_i + b))}}$$

$$\ell(y_i, w^T x_i + b) = \log\left(1 + e^{(-y_i(w^T x_i + b))}\right)$$

- función objetivo con regularización: (Prevenir sobreajuste)

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

- Cálculo de gradiente:

$$\nabla_w J = -\frac{1}{N} \sum_{i=1}^{N} \frac{y_i x_i}{1 + \exp(y_i(w^T x_i + b))} + \lambda w$$

Para el bias b:

$$\frac{\partial J}{\partial b} = -\frac{1}{N} \sum_{i=1}^{N} \frac{y_i}{1 + \exp(y_i(w^T x_i + b))}$$

- Descenso de gradiente estocástico:
  No calcula el gradiente completo, 1 ejemplo a la vez:
  para (x_t, y_t), es:

$$w \leftarrow w - \eta_t \left(\frac{-y_t x_t}{1 + \exp(y_t(w^T x_t + b))} + \lambda w \right)$$

$$b \leftarrow b - \eta_t \left(\frac{-y_t}{1 + \exp(y_t(w^T x_t + b))} \right)$$

(η_t: tasa de aprendizaje en t)

SGD resuelve el siguiente problema de regresión logística regularizada:

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

Optimiza iterativamente, actualizando pesos después de cada muestra

Logistic Regression: (usando SGD)

Dataset: $\{(x_i, y_i)\}_{i=1}^{N}$; input $x_i \in \mathbb{R}^d$; labels $y_i \in \{0,1\}$

- Logistic model (Sigmoid Activation)
  Probabilidad de class 1:

$$P(y=1 \mid x) = \sigma(w^T x + b) = \frac{1}{1 + \exp(-w^T x - b)}$$

$$\hat{y} = \begin{cases} 1, & \sigma(w^T x + b) > 0{,}5 \\ 0, & \text{e.o.c} \end{cases}$$

- Log-likelihood (forma maximizada)
  Likelihood para dataset:

$$\ell(w,b) = \prod_{i=1}^{N} \sigma(w^T x_i + b)^{y_i} (1 - \sigma(w^T x_i + b))^{1-y_i}$$

Log-Likelihood:

$$\log(\ell(w,b)) = \sum_{i=1}^{N} (y_i \log(\sigma(w^T x_i + b)) + (1 - y_i) \log(1 - \sigma(w^T x_i + b)))$$

- Forma minimizada (Negative Log-likelihood):

$$J(w,b) = -\sum_{i=1}^{N} (y_i \log(\sigma(w^T x_i + b)) + (1 - y_i) \log(1 - \sigma(w^T x_i + b)))$$

- Vector con Regularización:
  $y_i \in \{-1, 1\}$, simplifica la expresión loss

$$J(w,b) = \sum_{i=1}^{N} \log(1 + \exp(-y_i(w^T x_i + b)))$$

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

($\frac{\lambda}{2}$ regularización)

Linear Discriminant Analysis (LDA):

- Distribuciones de clase:

$$x \mid y = c \sim N(\mu_c, \Sigma)$$

(Covarianza en común)

Priors de clase $P(y=c) = \pi_c$

• • •

- Regla de Bayes y función discriminante:

$$P(y=c|x) = \frac{P(x|y=c)\pi_c}{P(x)}$$

Log-posterior lleva a la función discriminante:

$$\delta_c(x) = x^T \Sigma^{-1} \mu_c - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c$$

$$\hat{y} = \arg\max_c \delta_c(x)$$

- Estimación de parámetros:
Media de clase:

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} x_i$$

Covarianza:

$$\hat{\Sigma} = \frac{1}{N-K} \sum_{c=1}^{K} \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$$

Priors de clase:

$$\hat{\pi}_c = \frac{N_c}{N}$$

- Optimización:
Sobre direcciones $f_n \in \mathbb{R}^d$ que maximizan la separación de clases:

$$\max_w \frac{w^T S_B w}{w^T S_w w}$$

Dispersión entre clases:

$$S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$$

Dispersión dentro de la clase:

$$S_w = \Sigma_0 + \Sigma_1$$

- Problema generalizado de valores propios:
Cociente de Rayleigh, que maximiza en:

$$w \propto S_w^{-1}(\mu_1 - \mu_0)$$

Para multiclases:

$$S_B w = \lambda S_w w$$

$$\max_w \frac{w^T (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T \Sigma_w} \longrightarrow w = \Sigma^{-1}(\mu_1 - \mu_0)$$

LDA multiclase generaliza esto mediante vectores propios

$$S_w^{-1} S_B$$

KNeighborsClassifier:

Busca predecir $\hat{y}$ encontrando la etiqueta mayoritaria entre los k vecinos más cercanos de x en train.

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

test $x \in \mathbb{R}^d$

- Cálculo de distancia:

$$D(x, x_i) = \|x - x_i\|_2 = \sqrt{(x-x_i)^T(x-x_i)}$$

k vecinos cercanos $N_k(x) \subset \{1, \dots, N\}$

$N_k(x) \longrightarrow$ Más pequeña $D(x, x_i)$ de k

- Regla predictiva (Mayoría)

$$\hat{y} = \arg\max_{c \in \{1,\dots,C\}} \sum_{i \in N_k(x)} 1(y_i = c)$$

- KNN ponderado (opcional):

$$\hat{y} = \arg\max_c \sum_{i \in N_k(x)} \frac{1}{D(x, x_i)} \cdot 1(y_i = c)$$

- Cálculo de costo (al predecir):

$$\arg\min_{N_k(x)} \sum_{i \in N_k(x)} D(x, x_i)$$

Linear SVC:

Busca un hiperplano $w^T x + b = 0$ que separa las 2 clases con el máximo márgen.

- Loss function hinge:
Penaliza los puntos mal clasificados y que violan el margen:

$$\ell(y_i, f(x_i)) = \max(0, 1 - y_i(w^T x_i + b))$$

Loss es 0 cuando $y_i(w^T x_i + b) \geq 1$

- Riesgo empírico regularizado (Primordial):
función de costo:

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

$\|w\|^2$ impone la maximización de márgenes (pesos pequeños $\rightarrow$ Margen grande)

C es un hiperparámetro que controla el equilibrio entre errores de margen y clasificación.

- Optimización (primordial):

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

SVC:

Función de decisión:

$$f(x) = \text{Sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right)$$

kernel function

multiplicadores lagrange

- Problema primordial (SVM margen suave):

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

Parámetro para misclasificación

Sujeto a $y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \; , \; \xi_i \geq 0$

Característica de transformación vía kernel

- Problema de derivación dual:

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

Sujeto a $0 \leq \alpha_i \leq C$

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

- Función de decisión final:

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

Random Forest Classifier:

Tenemos M árboles de decisión:

$$\hat{y} = \text{majority\_vote}(T_1(x), T_2(x), \ldots, T_M(x))$$

Cada árbol:
- Muestreo de datos con sustitución
- Seleccionar un subconjunto aleatorio de entidades en c/división
- Cultivar un árbol usando criterios codiciosos (Gini, entropía)

- Bootstrap Sample:
Para training $D = \{(x_i, y_i)\}_{i=1}^{N}$, genera un sample D_m de tamaño N

- Crecimiento de árboles:
Seleccionar atributo j y dividir s, tal que se divida en left/right maximizando la separación de clase.

- Criterio de división:

$$\max_{j,s} \Delta I = I(D) - \left(\frac{|D_{left}|}{|D|} I(D_{left}) + \frac{|D_{right}|}{|D|} I(D_{right}) \right)$$

- $I(\cdot)$: Medida de impurezas (Gini, Entropía)
- D_{left}, D_{right}: divide
- Intenta todos los s en j seleccionados

Gini impureza:

$$I(D) = 1 - \sum_{c} p_c^2, \quad p_c = \frac{1}{|D|} \sum_{i \in D} 1(y_i = c)$$

Entropía:

$$I(D) = -\sum_{c} p_c \log p_c$$

- Repetir recursivamente:
Repetir hasta que:
- Máximo de profundidad
- Mínimos samples por nodo
- Hojas puras (todos los samples de misma clase)

$$\hat{y}(x) = \arg\max_{c} \sum_{m=1}^{M} 1(T_m(x) = c)$$