1 Linear regression

Mô hình: $\hat{y}(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$

Hợp lý cực đại

$p(t_n|\mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}(t_n|\mathbf{w}^T\mathbf{x}_n, \beta^{-1})$

$L(\mathbf{w}, \beta) = \beta E_D(\mathbf{w}) - \frac{N}{2}\log\beta + \frac{N}{2}\log(2\pi)$

với: $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\mathbf{x}_n)^2$

Nghiệm giải tích

Cực tiểu $E_D(\mathbf{w})$: $\mathbf{w}_{ML} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$

$\beta_{ML}^{-1} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w}_{ML}^T\mathbf{x}_n)^2$

Giải thuật lặp (Gradient Descent)

$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta\nabla E_D(\mathbf{w})$

Hồi quy cho quan hệ phi tuyến

$\mathbf{x} \to \boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x})]$

Đánh giá mô hình

Dự báo: $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}_{ML}$

Các độ đo

$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \hat{y}_n)^2$

$\text{RMSE} = \sqrt{\text{MSE}}$

Hạn chế quá khớp

Ridge Regression

$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\mathbf{x}_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$

Nghiệm: $\mathbf{w}_{ridge} = (\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$

LASSO: $L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\mathbf{x}_n)^2 + \lambda\sum_{m=1}^{M}|w_m|$

Dự báo cho nhiều biến

Với K biến đầu ra: $\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{T}$

Trong đó: $\mathbf{T} \in \mathbb{R}^{N\times K}, \mathbf{W} \in \mathbb{R}^{M\times K}$

2 Logistic regression

Hàm sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$

Mô hình dự báo:

$\hat{y} = p(C_1|x, w) = \sigma(w^T x)$

Nếu $\hat{y} \geq \lambda$ thì $x \in C_1$. Ngược lại, $x \in C_0$

Ước lượng tham số của mô hình

Xây dựng hàm mục tiêu

Với một điểm dữ liệu (x, y):

$p(y|x, w) = \hat{y}^y(1-\hat{y})^{1-y}$

Với N điểm dữ liệu:

$p(t|X, w) = \prod_{n=1}^{N}\hat{y}_n^{y_n}(1-\hat{y}_n)^{1-y_n}$

Negative log-likelihood: $L(w) = -\sum_{n=1}^{N}[y_n\log\hat{y}_n + (1-y_n)\log(1-\hat{y}_n)]$

Hàm này còn được gọi là **cross-entropy**.

Tìm hệ số của mô hình

$\nabla L(w) = \sum_{n=1}^{N}(\hat{y}_n - y_n)x_n = X^T(\hat{y} - y)$

Giải thuật lặp với đạo hàm bậc 2

Ma trận Hessian: $H = \nabla^2 L(w) = \sum_{n=1}^{N}\hat{y}_n(1-\hat{y}_n)x_n x_n^T = X^T R X$

R là ma trận đường chéo: $R_{nn} = \hat{y}_n(1-\hat{y}_n)$

Phương pháp sử dụng: Gradient Descent, Newton–Raphson, Iterative Re-weighted Least Squares (IRLS)

3 Softmax regression

Mỗi nhãn được mã hóa dưới dạng véctơ one-hot kích thước K.

Mô hình tuyến tính với Softmax

Mô hình dự báo

Ma trận tham số của mô hình:

$W = [w_1, w_2, \ldots w_K]^T \in \mathbb{R}^{K\times M}$

Các bước tính toán: $Z = XW^T \quad (N\times K)$,

$\hat{Y} = \text{softmax}(Z)$

Hàm softmax: $\hat{y}_k = \frac{\exp(z_k)}{\sum_{i=1}^{K}\exp(z_i)}$

Dự đoán

Nhãn dự đoán: prediction $= \arg\max_k \hat{y}_k$

Hàm mục tiêu và tối ưu

Hàm hợp lý

Xác suất của tập nhãn:

$p(t|X, W) = \prod_{n=1}^{N}\prod_{k=1}^{K}\hat{y}_{n,k}^{y_{n,k}}$

Hàm mất mát Cross-Entropy

Hàm mất mát được xây dựng bằng cách lấy log và đổi dấu:

$L(W) = -\sum_{n=1}^{N}\sum_{k=1}^{K}y_{n,k}\log(\hat{y}_{n,k})$

Mục tiêu là tìm W sao cho $L(W)$ đạt giá trị nhỏ nhất.

Ước lượng tham số

Gradient Descent

Gradient của hàm mất mát đối với đầu vào softmax: $\frac{\partial L}{\partial z} = (\hat{y} - y)^T$

Gradient theo tham số: $\Delta W = (\hat{y} - y)^T x^T$

Cập nhật trọng số: $W \leftarrow W - \eta\Delta W$

4 MLP

Core idea: deep learning = nonlinear feature extraction + simple linear head.

Forward Pass

Let $h^{(0)} = x$; $h^{(l)} = \phi\left(W^{(l)}h^{(l-1)} + b^{(l)}\right)$, $l = 1, \ldots, L$ where $\phi(\cdot)$ is a nonlinear activation function.

Output Layer

Regression: $\hat{y} = W^{(L+1)}h^{(L)} + b^{(L+1)}$

Classification: $\hat{p}_k = \frac{\exp(w_k^\top h^{(L)} + b_k)}{\sum_j \exp(w_j^\top h^{(L)} + b_j)}$

Function Composition View

$f(x;\theta) = f^{(L+1)} \circ \phi \circ f^{(L)} \circ \cdots \circ \phi \circ f^{(1)}(x)$

Fully Connected (Linear) Layer

Single sample: $y = Wx + b$

Mini-batch $X \in \mathbb{R}^{B\times N}$: $Y = XW^\top + \mathbf{1}b^\top$

Activation Functions

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$

Tanh: $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

ReLU: $\text{ReLU}(z) = \max(0, z)$

Leaky ReLU: $\text{LReLU}(z) = \begin{cases} z, & z \geq 0 \\ \alpha z, & z < 0 \end{cases}$

SiLU (Swish): $\text{SiLU}(z) = z\sigma(z)$

5 Training ANN

Problem Setup Given a dataset $\{(x_i, y_i)\}_{i=1}^n$ and a model $\hat{y}_i = f_\theta(x_i)$, training aims to solve:

$\min_\theta \; L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \hat{y}_i)$.

Regression Losses:

$L_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

$L_{\text{MAE}} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$.

Huber Loss:

$$L_\delta(e_i) = \begin{cases} \frac{1}{2}e_i^2, & |e_i| \leq \delta, \\ \delta(|e_i| - \frac{1}{2}\delta), & |e_i| > \delta, \end{cases}$$

where $e_i = y_i - \hat{y}_i$.

Classification Losses

BCE: For $y_i \in \{0, 1\}$ and $p_i = \sigma(z_i)$: $L_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}[y_i\log p_i + (1-y_i)\log(1-p_i)]$.

Categorical Cross-Entropy: For K classes with one-hot labels:

$L_{\text{CE}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}y_{ik}\log p_{ik}$.

Training Process:

Forward: compute predictions and loss.

Backward: compute grads via backprop.

Update: update param with optimizer.

SGD: $\theta \leftarrow \theta - \eta\nabla_\theta L$.

Training Algorithm

Algorithm 1 SGD Training Procedure

```
Initialize parameters θ
for epoch = 1 to E do
    Shuffle dataset and create mini-batches
    for each mini-batch (X, y) do
        Forward pass
        Compute loss L
        Backward pass: compute ∇_θ L
        Update: θ ← θ − η∇_θ L
    end for
end for
```

SGD with Momentum:

$v_t = \mu v_{t-1} + g_t, \quad \theta \leftarrow \theta - \eta v_t$.

Adam: $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$,

$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$,

$\theta \leftarrow \theta - \eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$.

AdamW: decouples weight decay from the gradient update

6 SVM primal problem

Đầu vào

Ma trận dữ liệu: $X \in \mathbb{R}^{N\times(M-1)}$.

Nhãn: $\mathbf{t} = [t_1, t_2, \ldots, t_N]^T, t_n \in \{-1, +1\}$

Mục tiêu Xác định đường biên quyết định: $\mathbf{w}^T\mathbf{x} + b = 0$ sao cho **lề (margin)** giữa hai lớp là lớn nhất.

Khoảng cách từ điểm đến đường thẳng

Siêu phẳng trong không gian đặc trưng $(M-1)$ chiều: $\mathbf{w}^T\mathbf{x} + b = 0$

Khoảng cách có dấu từ điểm $\mathbf{x}$ đến siêu phẳng: $d(\mathbf{x}) = \frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$

Khoảng cách hình học: $|d(\mathbf{x})| = \frac{|\mathbf{w}^T\mathbf{x}+b|}{\|\mathbf{w}\|}$

Hàm quyết định: $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

Quy tắc phân lớp: $\text{class}(\mathbf{x}) = \text{sign}(y(\mathbf{x}))$

Lề (Margin): $m_{\mathbf{w}} = \min_n \frac{t_n(\mathbf{w}^T\mathbf{x}_n+b)}{\|\mathbf{w}\|}$

Cực đại lề: Có thể chuẩn hóa sao cho: $t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \quad \forall n$

Hàm mục tiêu

Cực đại lề tương đương với bài toán: $\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2$ với ràng buộc:

$t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \quad n = 1, \ldots, N$

Bài toán gốc:

$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$

s.t. $\quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \quad \forall n$

Giải bằng thư viện CVXOPT

Dạng chuẩn: $\min_{\mathbf{x}}\frac{1}{2}\mathbf{x}^T K\mathbf{x} + \mathbf{p}^T\mathbf{x}$

s.t. $G\mathbf{x} \leq \mathbf{h}$. Trong đó: $\mathbf{x} = \begin{bmatrix}\mathbf{w} \\ b\end{bmatrix}$

7 SVM dual problem

Hàm Lagrangian

Hàm Lagrangian của bài: $L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N}\alpha_n\big[t_n(w^T x_n + b) - 1\big]$,

với $\alpha_n \geq 0, \quad n = 1, \ldots, N$.

Điều kiện KKT

(KKT-1) ĐK dừng: $\nabla_{w,b}L(w, b, \alpha) = 0$

(KKT-2) Ràng buộc gốc

(KKT-3) Ràng buộc đối ngẫu: $\alpha_n \geq 0$

(KKT-4) ĐK bù: $\alpha_n[1 - t_n(w^T x_n + b)] = 0$

Xây dựng hàm đối ngẫu

$\frac{\partial L}{\partial w} = w - \sum_{n=1}^{N}\alpha_n t_n x_n = 0$

$\frac{\partial L}{\partial b} = \sum_{n=1}^{N}\alpha_n t_n = 0$

Suy ra: $w = \sum_{n=1}^{N}\alpha_n t_n x_n$.

Thay vào Lagrangian, được hàm đối ngẫu: $g(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{r=1}^{N} \sum_{c=1}^{N} \alpha_r \alpha_c t_r t_c x_r^T x_c$

Bài toán đối ngẫu

$\alpha^* = \arg\min_\alpha \ \frac{1}{2}\alpha^T K \alpha - \mathbf{1}^T \alpha$

s.t. $\alpha_n \geq 0, n = 1, \ldots, N, \sum_{n=1}^{N} \alpha_n t_n = 0,$

trong đó $K_{rc} = t_r t_c x_r^T x_c$.

Tiêu chuẩn Slater

Vì tồn tại (w, b) sao cho $t_n(w^T x_n + b) > 1, \ \forall n$, nên bài toán thỏa tiêu chuẩn Slater. Do đó: $\min_{w,b} \max_\alpha L(w, b, \alpha) = \max_\alpha \min_{w,b} L(w, b, \alpha)$.

Suy ra **duality gap bằng 0**.

Công thức dự báo

Với tập véc-tơ hỗ trợ $S = \{n : \alpha_n > 0\}$, $y(x) = \sum_{n \in S} \alpha_n t_n x_n^T x + b$.

Nhãn dự báo: $\hat{y} = \text{sign}(y(x))$.

8 SVM soft margin

Ràng buộc mới

$$t_n(w^T x_n + b) \geq 1 - \xi_n, \quad n = 1, \ldots, N, \tag{1}$$

$$\xi_n \geq 0. \tag{2}$$

Hàm mục tiêu mới

$f_0(w, b, \xi) = \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n$, trong đó $C > 0$ là siêu tham số điều chỉnh mức phạt.

Bài toán tối ưu

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n \\ \text{s.t.} \quad & t_n(w^T x_n + b) \geq 1 - \xi_n, \\ & \xi_n \geq 0, \quad n = 1, \ldots, N. \end{aligned}$$

Bài toán đối ngẫu

Hàm Lagrangian

$$\begin{aligned} L(w, b, \xi, \alpha, \mu) = & \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n \\ & - \sum_{n=1}^{N} \alpha_n [t_n(w^T x_n + b) - 1 + \xi_n] - \sum_{n=1}^{N} \mu_n \xi_n. \end{aligned}$$

Điều kiện KKT

$$w = \sum_{n=1}^{N} \alpha_n t_n x_n, \tag{3}$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0, \tag{4}$$

$$0 \leq \alpha_n \leq C. \tag{5}$$

Bài toán đối ngẫu

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{r=1}^{N} \sum_{c=1}^{N} \alpha_r \alpha_c t_r t_c x_r^T x_c - \sum_{n=1}^{N} \alpha_n \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \\ & \sum_{n=1}^{N} \alpha_n t_n = 0. \end{aligned}$$

Công thức dự báo

Sau khi tìm được α và b, hàm quyết định là: $y(x) = \sum_{n \in S} \alpha_n t_n x_n^T x + b$, trong đó S là tập các véc-tơ hỗ trợ.

Nhãn dự báo: label $= \text{sign}(y(x))$.

Cài đặt với CVXOPT

Bài toán đối ngẫu có dạng chuẩn: $\min_\alpha \ \frac{1}{2}\alpha^T K \alpha + p^T \alpha$ s.t. $G\alpha \leq h$, $A\alpha = b$.

Các ràng buộc hộp $0 \leq \alpha_n \leq C$ được mã hóa trong ma trận G và h.

9 SVM kernel

SVM hai lớp với lề mềm Xét tập huấn luyện $\{(x_i, t_i)\}_{i=1}^N$ với $t_i \in \{-1, +1\}$. Bài toán đối ngẫu của SVM lề mềm được viết dưới dạng:

$\alpha^* = \arg\min_\alpha \frac{1}{2}\alpha^T K \alpha + p^T \alpha$

với các ràng buộc: $G\alpha \leq h, \qquad A\alpha = b$

Trong đó, ma trận kernel K được xác định bởi tích vô hướng giữa các điểm dữ liệu.

Dự báo Giá trị bias b được ước lượng bởi: $b = \frac{1}{N_M}\left(t_M - K_{MS}[\alpha_S \odot t_S]\right)^T \mathbf{1}$

Hàm dự báo: $y = K_{BS}[\alpha_S \odot t_S] + b$

Nhãn dự đoán: label $= \text{sign}(y)$

Khi đường biên giới phi tuyến Trong nhiều bài toán thực tế, dữ liệu không thể phân tách tuyến tính. Giải pháp là ánh xạ dữ liệu thông qua hàm trích đặc trưng: $\Phi : \mathbb{R}^d \to \mathcal{H}$

Tuy nhiên, việc tính trực tiếp $\Phi(x)$ có thể tốn kém hoặc không khả thi khi không gian đặc trưng có số chiều rất lớn hoặc vô hạn.

Phương pháp Kernel Phương pháp kernel cho phép tính: $\langle \Phi(x_i), \Phi(x_j) \rangle$ mà không cần biết tường minh $\Phi(x)$, thông qua một hàm kernel: $k(x_i, x_j)$

Điều kiện Mercer Một hàm $k(x_i, x_j)$ là kernel hợp lệ nếu:

- Đối xứng: $k(x_i, x_j) = k(x_j, x_i)$
- Bán định dương: $\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geq 0$

Huấn luyện và dự báo với Kernel Ma trận Gram kernel: $K_{Gram} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$

Việc sử dụng kernel đảm bảo bài toán tối ưu là lồi và có nghiệm toàn cục.

Các kernel thông dụng Một số kernel phổ biến trong thực tế:

- **Linear**: $k(x, x') = x^T x'$
- **Polynomial**: $k(x, x') = (\gamma x^T x' + r)^d$
- **RBF (Gaussian)**: $k(x, x') = \exp(-\gamma\|x - x'\|^2)$
- **Sigmoid**: $k(x, x') = \tanh(\gamma x^T x' + r)$

Kernel RBF tương ứng với không gian đặc trưng vô hạn chiều.

Thiết kế Kernel Việc thiết kế kernel phụ thuộc mạnh vào kiến thức miền (domain knowledge), với mục tiêu:

- $k(x_i, x_j)$ lớn nếu x_i, x_j cùng lớp
- $k(x_i, x_j)$ nhỏ nếu khác lớp

Minh họa Các thí nghiệm với kernel đa thức và kernel RBF cho thấy khả năng phi tuyến hóa đường biên phân lớp một cách hiệu quả.

10 PCA

Mô tả bài toán Giả sử tập dữ liệu đầu vào: $X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,D} \end{bmatrix}$

trong đó:

- $X \in \mathbb{R}^{N \times D}$;
- D rất lớn và các chiều có tương quan với nhau.

Mục tiêu:

1. Giảm số chiều từ D xuống M với $M \ll D$;
2. Các đặc trưng mới không còn tương quan tuyến tính.

Kiến thức toán học liên quan Phương sai và hiệp phương sai Trung bình của dữ liệu: $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$

Dữ liệu được chuẩn hóa: $z_n = x_n - \mu$

Ma trận hiệp phương sai: $S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)(x_n - \mu)^T$

Eigenvalue và Eigenvector Với ma trận vuông A, bài toán eigen: $Au = \lambda u$ trong đó u là eigenvector và λ là eigenvalue.

Cơ sở lý luận của PCA PCA có thể được nhìn theo hai cách:

- Cực đại hóa phương sai của dữ liệu sau khi chiếu;
- Cực tiểu hóa sai số phục hồi dữ liệu.

Hai cách tiếp cận này là tương đương về mặt toán học.

Cực đại hóa phương sai Với một vectơ đơn vị u, phương sai của dữ liệu khi chiếu lên u là: $\sigma^2 = u^T S u$

Bài toán tối ưu: $\max_u \quad u^T S u$ s.t. $u^T u = 1$

Sử dụng nhân tử Lagrange dẫn đến: $Su = \lambda u$

Do đó:

- Các trục chính của PCA là các eigenvector của S;
- Phương sai tương ứng là các eigenvalue.

Thu giảm số chiều Chọn M eigenvector tương ứng với M eigenvalue lớn nhất, tạo thành ma trận: $\hat{U} = [u_1, u_2, \ldots, u_M]$

Chiếu dữ liệu: $X_{\text{PCA}} = (X - \mu^T)\hat{U}$

Phục hồi xấp xỉ: $\hat{X} = \mu^T + X_{\text{PCA}}\hat{U}^T$

PCA qua API Ví dụ PCA trong `scikit-learn`:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X)
print(pca.explained_variance_ratio_)
```

Singular Value Decomposition (SVD) Phân rã SVD: $X = USV^T$

- PCA thực hiện eigen-decomposition trên ma trận hiệp phương sai;

- SVD phân rã trực tiếp trên ma trận dữ liệu và có độ ổn định số cao hơn.

Ứng dụng của PCA

- Nén dữ liệu;
- Trực quan hóa dữ liệu nhiều chiều;
- Tiền xử lý cho học máy;
- Nhận dạng mẫu và xử lý ảnh.

11 LDA

Giới thiệu về LDA

Đầu vào

Cho tập dữ liệu: $X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,D} \end{bmatrix}$, $t = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$

Trong đó:

- $X \in \mathbb{R}^{N \times D}$, với D thường rất lớn.
- $t_k \in \{1, 2, \ldots, C\}$ là nhãn lớp của điểm dữ liệu thứ k.

Mục tiêu

Mục tiêu của LDA là:

1. Giảm số chiều từ D xuống M, với $M \leq C - 1$.
2. Dữ liệu sau khi chiếu có độ phân tách giữa các lớp là lớn nhất.

LDA qua API

Ví dụ sử dụng `Scikit-learn`:

```
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()
X = iris.data
y = iris.target

lda = LinearDiscriminantAnalysis(n_components=2)
X_r = lda.fit(X, y).transform(X)
```

Bài toán tối ưu

Tâm của mỗi lớp

Với lớp k: $\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in C_k} \mathbf{x}_n$

Between-class scatter matrix

$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$

Within-class scatter matrix

$S_W = \sum_{k=1}^{C} \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T$

Hàm mục tiêu của Fisher

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

Mục tiêu: $\mathbf{w}^* = \arg\max_{\mathbf{w}} J(\mathbf{w})$

Tìm nghiệm

Giải bài toán tối ưu dẫn đến phương trình trị riêng: $S_W^{-1} S_B \mathbf{w} = \lambda \mathbf{w}$

Hướng chiếu tối ưu là: $\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

Trường hợp có C lớp

Hàm mục tiêu tổng quát

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

Số chiều tối đa

Số chiều tối đa có thể chọn là: $M \leq C - 1$

Do ma trận S_B có hạng tối đa là $C - 1$.

Giải thuật LDA

1. Tính S_B và S_W.
2. Tính $A = S_W^{-1} S_B$.
3. Thực hiện SVD hoặc eigen-decomposition.
4. Chọn M eigenvector tương ứng với eigenvalue lớn nhất.
5. Chiếu dữ liệu: $\hat{X} = (X - m^T)W$.

12 Ensemble

Bias–Variance Perspective

Prediction error can be decomposed into bias and variance. Bias results from overly simplistic assumptions, while variance reflects sensitivity to training data fluctuations. Ensemble learning aims to reduce variance by averaging predictions of multiple weakly correlated models. For regression, the variance of an ensemble predictor can be approximated as $\text{Var}\left(\frac{1}{M}\sum_{m=1}^{M} \hat{y}^{(m)}\right) \approx \frac{1}{M^2}\sum_{m=1}^{M} \text{Var}(\hat{y}^{(m)})$.

Bagging (Bootstrap Aggregating)

Bagging is designed to reduce variance, particularly for unstable learners such as decision trees.

Method Description

Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, Bagging constructs an ensemble of M models as follows:

1. Draw M bootstrap datasets $D_1, D_2, \ldots, D_M$ by sampling n points with replacement from D.
2. Train a base learner on each bootstrap dataset to obtain models $h_1, h_2, \ldots, h_M$.
3. Combine predictions of all models: $\hat{y}(x) = \begin{cases} \frac{1}{M}\sum_{m=1}^{M} h_m(x), & \text{regression,} \\ \text{majority vote,} & \text{classification.} \end{cases}$

Properties

Bagging is simple, highly parallelizable, and effective at variance reduction. However, it requires storing many models and results in slower inference.

Random Forest

Random Forest extends Bagging by introducing randomness in feature selection. Each tree is trained on a bootstrap dataset, and at each split only a random subset of features is considered.

Training Procedure

For each tree:

1. Sample a bootstrap dataset from the training set.
2. Grow a decision tree by recursively splitting nodes.
3. At each split, randomly select a subset of features $F_{\text{sub}} \subset \{1, \ldots, d\}$.
4. Choose the best split using only features in F_{sub}.

Typical choices are $|F_{\text{sub}}| = \sqrt{d}$ for classification and $|F_{\text{sub}}| = d/3$ for regression.

Boosting

Boosting methods train models sequentially, where each model focuses on samples misclassified by previous ones. Unlike Bagging, Boosting can reduce both bias and variance.

AdaBoost

For binary classification with $y_i \in \{-1, +1\}$, AdaBoost maintains a weight distribution over training samples. At iteration t, a weak learner h_t is trained using weighted data. The final classifier is $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$, where α_t is determined by the weighted classification error of h_t.

Gradient Boosting

Gradient Boosting views ensemble construction as gradient descent in function space. At each iteration, a new weak learner is fitted to the negative gradient of the loss function with respect to current predictions.

Voting and Stacking

Voting

Voting combines predictions of multiple models using a fixed rule such as majority voting or probability averaging. It is simple and often serves as a strong baseline.

Stacking

Stacking trains a meta-learner on predictions of base models. To avoid overfitting, cross-validation is used to generate out-of-fold predictions, which are then used as inputs for the meta-model.

Comparison and Practical Considerations

Bagging is most effective for high-variance models, Boosting can significantly improve accuracy but is sensitive to noise, and Stacking offers the greatest flexibility at the cost of increased complexity. In practice, Random Forests and Gradient Boosting are strong default choices for tabular data.

13 Genetic algorithm

Key Components of Genetic Algorithms

Representation (Encoding)

A solution is encoded as a chromosome. Common encoding methods include:

- **Binary encoding**: chromosomes consist of bits (0 or 1)
- **Real-valued encoding**: genes are real numbers
- **Permutation encoding**: chromosomes represent ordered sequences

- **Tree encoding**: solutions are tree structures (used in genetic programming)

Fitness Function
The fitness function $f(x)$ evaluates the quality of a solution x. For minimization problems, a transformation is typically applied, such as: $f_{\max}(x) = \frac{1}{1+f_{\min}(x)}$
The fitness function should be computationally efficient, as it is evaluated many times during the evolutionary process.
Selection
Selection chooses parent solutions based on fitness. Popular methods include:

- Roulette wheel selection
- Rank selection
- Tournament selection
- Elitism

Elitism ensures that the best individuals are preserved across generations.
Crossover
Crossover combines genetic material from two parents to produce offspring. Common techniques include:

- Single-point crossover
- Two-point crossover
- Uniform crossover
- Arithmetic crossover (for real-valued encoding)

Mutation
Mutation introduces random changes to maintain population diversity. Typical mutation strategies include:

- Bit flipping for binary encoding
- Gaussian or uniform noise for real-valued encoding
- Swap or inversion for permutation encoding

Genetic Algorithm Framework
A standard genetic algorithm follows these steps:

1. Initialize a population of N individuals
2. Evaluate fitness of each individual
3. Repeat until a termination condition is met:
   (a) Select parents based on fitness
   (b) Apply crossover to generate offspring
   (c) Apply mutation to offspring
   (d) Evaluate fitness of new individuals
   (e) Form the next generation (with optional elitism)
4. Return the best solution found

Termination conditions may include a maximum number of generations, fitness convergence, stagnation, or time limits.
Simple Example
Consider maximizing the function: $f(x) = x^2, \quad x \in \{0, 1, \ldots, 15\}$
Binary encoding with 4 bits is used. An example initial population is shown in Table **??**.

After selection, crossover, and mutation, the population gradually converges toward the optimal solution $x = 15$.
Parameters and Tuning
Key parameters include:

- Population size (N)
- Crossover probability (p_c)
- Mutation probability (p_m)

Typical values are:

- $N = 20$–200
- $p_c = 0.6$–0.9
- $p_m = 0.001$–0.1

Parameter selection is problem-dependent and often requires empirical tuning.
Advantages and Disadvantages
Advantages

- Global search capability
- No requirement for gradient information
- Parallelizable structure
- Flexible solution representation

Disadvantages

- No guarantee of global optimality
- Computationally expensive
- Sensitive to parameter settings
- Risk of premature convergence

Applications
Genetic Algorithms have been successfully applied in:

- Combinatorial optimization (TSP, scheduling)
- Machine learning (feature selection, hyperparameter tuning)
- Engineering design (antennas, circuits)
- Bioinformatics (protein structure prediction)
- Game AI and procedural content generation

Variants and Extensions
Popular GA variants include:

- Real-Coded Genetic Algorithms (RCGA)
- Differential Evolution (DE)
- Genetic Programming (GP)
- Multi-objective Genetic Algorithms (e.g., NSGA-II)

These variants extend GA to continuous, programmatic, and multi-objective optimization problems.