MLCheatsheet

1. Bài toán Gốc: Toán học (Theory)

1. Thiết lập:

- $x, w \in \mathbb{R}^{M-1}$; Siêu phẳng: $w^T x + b = 0$.

2. Dẫn dắt: Max Margin $\Leftrightarrow$ Min $\|w\| \Leftrightarrow$ Min $\frac{1}{2}\|w\|^2$.

3. Công thức Primal:

$$\boxed{\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{s.t. } t_n(w^T x_n + b) \geq 1, \forall n \end{aligned}}$$

2. Bài toán Gốc: Implementation (CVXOPT)

Solver: $\min \frac{1}{2}u^T P u + q^T u$ s.t $Gu \leq h$. **Biến:** $u = [\mathbf{w}; \mathbf{b}]_{(M\times 1)}$.

Mapping:

$$\boxed{\begin{aligned} &P = \text{diag}(\mathbf{1}, \ldots, \mathbf{1}, \mathbf{0})_{(M\times M)} \\ &q = \mathbf{0}_{(M\times 1)} \\ &G = -[\text{diag}(\mathbf{t}) \cdot \mathbf{X}, \quad \mathbf{t}]_{(N\times M)} \\ &h = -\mathbf{1}_{(N\times 1)} \end{aligned}}$$

3. Lagrangian & Điều kiện KKT

1. Hàm Lagrangian:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} \underbrace{\alpha_n}_{\geq 0} \underbrace{\{1 - t_n(w^T x_n + b)\}}_{\substack{\text{đúng: } \leq 0 \\ \text{sai: } > 0}}$$

$$= \frac{1}{2}\|w\|^2 + E_{\text{data}} \text{ (cost trên dữ liệu)}$$

Tính chất: Với $\alpha_n \geq 0$, (w^*, b^*) tối ưu:

$$\boxed{\mathcal{L}(w^*, b^*, \alpha) \leq \frac{1}{2}\|w^*\|^2}$$

2. Tính đạo hàm:

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

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

3. Điều kiện dừng (KKT-1):

- $\nabla_w \mathcal{L} = 0 \Rightarrow \boxed{w = \sum_{n=1}^{N} \alpha_n t_n x_n}$ (1)
- $\nabla_b \mathcal{L} = 0 \Rightarrow \boxed{\sum_{n=1}^{N} \alpha_n t_n = 0}$ (2)

4. Các điều kiện KKT khác:

- **(KKT-2)** Ràng buộc gốc: $1 - t_n(w^T x_n + b) \leq 0$
- **(KKT-3)** Ràng buộc dual: $\alpha_n \geq 0$, $\forall n$
- **(KKT-4)** Điều kiện bù: $\alpha_n\{1 - t_n(w^T x_n + b)\} = 0$

Ý nghĩa: Thỏa KKT $\Rightarrow (w, b, \alpha) = (w^*, b^*, \alpha^*)$

5. Tiêu chuẩn Slater (cho SVM):

Nếu phần trong của tập khả thi không rỗng thì:

$$\exists (w, b) \text{ s.t. } 1 - t_i(w^T x_i + b) < 0, \ \forall i$$

$$\Longrightarrow \text{strong duality } (p^* = d^*) \ \Leftrightarrow \ \text{duality gap} = 0$$

4. Bài toán Dual: Biến đổi (Math)

Thay (1), (2) vào $\mathcal{L}$:

A. Thay w vào $\frac{1}{2}w^T w$:

$$\frac{1}{2}\left(\sum \alpha_i t_i x_i\right)^T \left(\sum \alpha_j t_j x_j\right) = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j (x_i^T x_j) \ (*)$$

B. Thay w vào $-\sum \alpha_n t_n w^T x_n$:

$$-\sum_n \alpha_n t_n \underbrace{\left(\sum_m \alpha_m t_m x_m\right)^T}_{w^T} x_n = -2 \times (*)$$

C. Kết quả ($A + B = -A$):

$$\boxed{\begin{aligned} \max_{\alpha} g(\alpha) = &\sum_n \alpha_n - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j (x_i^T x_j) \\ \text{s.t.} \quad &\alpha_n \geq 0; \quad \sum \alpha_n t_n = 0 (g(\alpha) \text{ luôn là hàm lồi}) \end{aligned}}$$

5. Bài toán Dual: Implementation (CVXOPT)

Solver: $\min \frac{1}{2}\alpha^T P \alpha + q^T \alpha$ s.t $G\alpha \leq h, A\alpha = b$. **Biến:** $\alpha \in \mathbb{R}^N$.

Mapping:

$$\boxed{\begin{aligned} &P = \mathbf{K}_{\text{Gram}(N\times N)} \quad (K_{ij} = t_i t_j x_i^T x_j) \\ &q = -\mathbf{1}_{(N\times 1)} \quad (\text{Max } \Sigma \to \text{Min } -\Sigma) \\ &G = -I_{(N\times N)}; \ h = \mathbf{0}_{(N\times 1)} \\ &A = t^T; \ b = 0 \end{aligned}}$$

6. Soft Margin & Kernel

1. Soft Margin (ý tưởng): Cho phép một số điểm vi phạm margin bằng biến slack ξ_n, đổi lại bị phạt trong hàm mục tiêu.

2. Bài toán gốc (Primal – soft margin):

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

$$\text{s.t. } t_n(w^T x_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0$$

Ý nghĩa C: C lớn $\Rightarrow$ phạt mạnh (ít sai, lề hẹp); C nhỏ $\Rightarrow$ cho sai nhiều (lề rộng).

3. Dạng đối ngẫu (Dual):

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + C\sum_n \xi_n - \sum_n \alpha_n\{t_n(w^T x_n + b) - 1 + \xi_n\} - \sum_n \mu_n \xi_n$$

$$\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j \, x_i^T x_j$$

$$\text{s.t. } \boxed{0 \leq \alpha_n \leq C}, \quad \sum_{n=1}^{N} \alpha_n t_n = 0$$

Phân loại theo KKT:

- $\alpha_n = 0$: điểm ngoài margin (không support)
- $0 < \alpha_n < C$: support vector trên margin
- $\alpha_n = C$: support vector vi phạm / phân loại sai

4. Kernel Trick: Thay thế tích vô hướng trong không gian đặc trưng:

$$\langle \phi(x_i), \phi(x_j) \rangle \ \longrightarrow \ k(x_i, x_j)$$

$$k(x, z) = \phi(x)^\top \phi(z)$$

Hàm quyết định của SVM:

$$f(x) = \sum_{n \in SV} \alpha_n t_n \, k(x_n, x) + b, \qquad y = \text{sign}(f(x))$$

Deep Learning

Goal: Multiclass classification ($y \in \{1, \ldots, K\}$).

Model (Softmax):

$$p_{ik} = \frac{\exp(\mathbf{w}_k^\top \mathbf{x}_i + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x}_i + b_j)}$$

Loss (Categorical Cross-Entropy):

$$\mathcal{L}_{\text{CE}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log p_{ik} = -\frac{1}{n}\sum_{i=1}^{n} \log p_{i,y_i}$$

Prediction:

$$\hat{y}_i = \arg\max_k p_{ik}$$

Notes:

- $y_{ik} \in \{0, 1\}$: nhãn one-hot của mẫu i tại lớp k
- p_{i,y_i}: xác suất dự đoán của lớp đúng của mẫu i

Vectorized form:

- **Vector (single sample):** for $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{W} \in \mathbb{R}^{M\times N}$, $\mathbf{b} \in \mathbb{R}^M$
  $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \in \mathbb{R}^M$
- **Matrix (mini-batch):** for $\mathbf{X} \in \mathbb{R}^{B\times N}$ (rows are samples)
  $\mathbf{Y} = \mathbf{X}\mathbf{W}^\top + \mathbf{1}\mathbf{b}^\top \in \mathbb{R}^{B\times M}$ (broadcast $\mathbf{b}$ to all rows)

Activation functions:

Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \qquad \sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Tanh

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \tanh'(x) = 1 - \tanh^2(x)$$

ReLU

$$\text{ReLU}(x) = \max(0, x), \qquad \text{ReLU}'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x \leq 0 \end{cases}$$

Leaky ReLU

$$\text{LReLU}(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x \leq 0 \end{cases}, \qquad \text{LReLU}'(x) = \begin{cases} 1, & x \geq 0 \\ \alpha, & x \leq 0 \end{cases}$$

SiLU (Swish)

$$\text{SiLU}(x) = x\,\sigma(x), \qquad \text{SiLU}'(x) = \sigma(x) + x\,\sigma(x)(1 - \sigma(x))$$

Linear Regression Solution:

Model:

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b$$

Loss (MSE):

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

Closed-form Solution (Normal Equation):

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

Hồi quy Ridge:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y}$$

Hồi quy LASSO:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

$\mathbf{w}^*$ (không có nghiệm dạng đóng, cần giải bằng phương pháp lặp)

Hồi quy Logistic:

Mô hình dự báo:

$$\hat{y}_n = p(y_n = 1 \mid \mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}_n) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_n)}$$

Hàm mục tiêu (negative log-likelihood / cross-entropy):

$$\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N}\Big[y_n \log \hat{y}_n + (1 - y_n)\log(1 - \hat{y}_n)\Big]$$

Gradient của hàm mất mát:

$$\nabla \mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N}(\hat{y}_n - y_n)\mathbf{x}_n = \mathbf{X}^\top(\hat{\mathbf{y}} - \mathbf{y})$$

$\mathbf{w}^*$ (không có nghiệm dạng đóng, giải bằng phương pháp lặp)

1. Convolution Cơ Bản

1. Định nghĩa Convolution:

$$Y(u,v) = X * W = \sum_{i=-r}^{r} \sum_{j=-r}^{r} X(u-i, v-j) W(i,j)$$

với $r = \lfloor k/2 \rfloor$ (bán kính kernel)
Cross-correlation:

$$Y(u,v) = X \star W = \sum_{i=-r}^{r} \sum_{j=-r}^{r} X(u+i, v+j) W(i,j)$$

=> Hầu hết thư viện Deep Learning (CNN) gọi là convolution nhưng thực chất đang dùng cross-correlation
Kích thước output:

$$o_1 \times o_2 = (i_1 - k_1 + 1) \times (i_2 - k_2 + 1)$$

Thuật toán:
1. Xoay kernel $180^\circ \rightarrow \text{Rot}_{180}(W)$
2. Flatten kernel thành vector
3. Padding input (nếu cần)
4. Căn chỉnh kernel với góc trên-trái
5. Lấy sub-image, flatten và tính dot-product
6. Trượt kernel theo stride, lặp lại

2. Padding & Stride

1. Half Padding: Padding size: $p = \lfloor k/2 \rfloor$

Output: $i_1 \times i_2$ (giữ nguyên kích thước)

2. Full Padding: Padding size: $p = k - 1$

Output: $(i_1 + k_1 - 1) \times (i_2 + k_2 - 1)$

3. Non-unit Stride: Với stride s_1, s_2:

$$\text{Output: } \left\lfloor \frac{i_1 + 2p_1 - k_1}{s_1} \right\rfloor + 1 \times \left\lfloor \frac{i_2 + 2p_2 - k_2}{s_2} \right\rfloor + 1$$

4. Multi-channel Input: Input D channels $\rightarrow$ mỗi filter có D channels

$$y_{u,v} = \sum_{d=1}^{D} \sum_{i,j} W_{i,j}^{(d)} X_{u+i,v+j}^{(d)}$$

5. Multi-filters Layer: K filters $\rightarrow$ output có K channels

VD: Rotate 180° Kernel

Ma trận gốc W:

$$W = \begin{bmatrix} 1 & 0 & 2 \\ 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

1. Flip horizontal: $\begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 0 \end{bmatrix}$
2. Flip vertical: $\text{Rot}_{180}(W) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 2 & 0 & 1 \end{bmatrix}$

3. Backprop qua Conv2D

Forward: $Y = X * W$
Backward - Tính ΔW:

$$\boxed{\Delta W = \text{Rot}_{180}(\Delta Y) * X}$$

- $\text{Rot}_{180}(\Delta Y)$ làm input, no padding
- X làm kernel

Chi tiết:

$$\delta w_{ij} = \sum_{u,v} \delta y_{u,v} \cdot x_{u+i,v+j}$$

Backward - Tính ΔX:

$$\boxed{\Delta X = \text{Rot}_{180}(W) * \Delta Y}$$

- $\text{Rot}_{180}(W)$ làm input, full padding
- ΔY làm kernel

Chi tiết:

$$\delta x_{uv} = \sum_{i,j} \delta y_{ij} \cdot w_{u-i,v-j}$$

4. Matrix Form Convolution

Ký hiệu:
- $\mathbb{X}_{conv}$: matrix từ im2col (X thành columns)
- $\mathbb{W}_{flat}$: flatten kernel thành row
- $\mathbb{W}_{conv}$: kernel2row matrix

Forward:

$$X * W = \text{Reshape}_{o_1 \times o_2}(\mathbb{W}_{flat} \times \mathbb{X}_{conv})$$
$$= \text{Reshape}_{o_1 \times o_2}(\mathbb{W}_{conv} \times \mathbb{X}_{flat})$$

Backward - ΔW:

$$\Delta W = \text{Reshape}_{k_1 \times k_2}(\Delta \mathbb{Y}_{flat} \times \mathbb{X}_{conv}^T)$$

Backward - ΔX:

$$\Delta X = \text{Reshape}_{i_1 \times i_2}(\mathbb{W}_{conv}^T \times \Delta \mathbb{Y}_{flat})$$

Ưu điểm: Tận dụng phép nhân ma trận tối ưu

5. Fully-Connected Layer

Forward:

$$y = Wx + b$$

với $W \in \mathbb{R}^{N \times M}$, $x \in \mathbb{R}^M$, $b \in \mathbb{R}^N$
Số tham số: $M \times N + N$
Backward - ΔW:

$$\boxed{\Delta W = \Delta y \cdot x^T}$$
$$\delta w_{n,m} = \delta y_n \cdot x_m$$

Backward - Δb:

$$\boxed{\Delta b = \Delta y}$$

Backward - ΔX:

$$\boxed{\Delta X = W^T \cdot \Delta y}$$
$$\delta x_m = \sum_{n=1}^{N} \delta y_n \cdot w_{n,m}$$

Update trọng số:

$$W \leftarrow W - \alpha \Delta W, \quad b \leftarrow b - \alpha \Delta b$$

6. Conv vs FC

Fully-Connected:
- Dense connectivity: mỗi output nối với mọi input
- Mỗi output có bộ weights riêng
- Số tham số: $(i_1 \times i_2) \times (o_1 \times o_2) + (o_1 \times o_2)$
- VD: input 3×3, output 2×2 $\rightarrow$ 9×4 + 4 = 40 params

Convolution:
- Sparse connectivity: output chỉ nối vùng local
- Parameter sharing: tất cả output dùng chung weights
- Số tham số: $k_1 \times k_2 \times D + 1$ (với D channels)
- VD: kernel 2×2 $\rightarrow$ 4 + 1 = 5 params

Ưu điểm Conv: Giảm tham số, học local patterns, translation invariant

7. Pooling Layer

Mục đích:
- Downsampling feature maps
- Loại bỏ redundant features
- Giảm overfitting

Max Pooling:

$$y_{u,v} = \max_{(i,j) \in \text{window}} x_{i,j}$$

Average Pooling:

$$y_{u,v} = \frac{1}{k_1 \times k_2} \sum_{(i,j) \in \text{window}} x_{i,j}$$

Output size (stride = 1):

$$(i_1 - k_1 + 1) \times (i_2 - k_2 + 1)$$

Output size (stride = s):

$$\left\lfloor \frac{i_1 - k_1}{s_1} \right\rfloor + 1 \times \left\lfloor \frac{i_2 - k_2}{s_2} \right\rfloor + 1$$

Thường dùng: window 2×2 hoặc 3×3, stride = 2

8. Gradient Descent

Forward Pass: Tính output từ input qua các layers
Backward Pass (Backpropagation): Tính gradients theo chain rule:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial w}$$

Batch Gradient Descent:

$$\Delta w = \frac{1}{m} \sum_{i=1}^{m} \Delta w^{(i)}$$
$$w \leftarrow w - \alpha \Delta w$$

với α là learning rate, m là batch size
Quy trình:
1. Forward qua tất cả samples trong batch
2. Backward tính gradients cho từng sample
3. Trung bình gradients
4. Update weights

1. Kiến thức Toán nền tảng

1. Hình chiếu: Chiếu vector x lên vector u: $l = \frac{u^T x}{\|u\|}$.

2. Dạng bậc hai & Đạo hàm:

- $u = [u_1, u_2, \ldots, u_N]^T$, $v = [v_1, v_2, \ldots, v_M]^T$.
- $A = [a_{ij}]_{N \times M}$.
- $u^T A v = \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} u_i^T v_j$. ($aij$ và $u_i^T v_j$ là vô hướng).
- $\frac{\partial u^T A u}{\partial u} = 2Au$ (với A đối xứng).

3. Eigenvectors & Eigenvalues:

$Au = \lambda u \quad \Leftrightarrow \quad (A - \lambda I)u = 0$(3.7)

$=>$ $\det(A - \lambda I) = 0$(3.8)

Giải 3.8 để tìm λ_i (có tối đa N eigenvalues), thay vào 3.7 để tìm u_i.

4. Tính chất eigenvectors và eigenvalues S:

- Ma trận A là đối xứng chứa các số thực.
- Có N eigenvector trực giao và N eigenvalue thực không âm.
- Hạng của A bằng số eigenvalue lớn hơn 0.
- Định thức của A là tích các eigenvalue. Trace của A là tổng các eigenvalue.
- Khi hạng của A là N, ma trận A^-1 có N eigenvalue = $1/\lambda_i$. (i = 1,...,N)

2. Giới thiệu & Mục tiêu PCA

1. Đầu vào:

- Ma trận dữ liệu $X(N \times D)$ (N mẫu, D chiều). Thường D rất lớn.
- Độ phức tạp tính toán tăng lên: ma trận Kgram, kernel, hiệp phương sai, khoảng cách giữa các điểm dữ liệu, số nơron lớp ẩn tăng. Và cần mô hình phức tạp hơn, điểm dữ liệu lớn hơn dễ gây overfit.

2. Mục tiêu:

- Giảm số chiều $D \rightarrow M$ sao cho $M \ll D$.
- Đặc trưng mới không tương quan (uncorrelated).
- Giữ lại thông tin quan trọng nhất (Variance lớn nhất).

3. Bài toán PCA

1. Hai quan điểm về PCA:

- Khi chiếu dữ liệu lên không gian con, phương sai của dữ liệu là lớn nhất: $argmax_u \frac{1}{N} \sum_{k=1}^{N} x_{u,k}^2$
- Hoặc khi chiếu dữ liệu lên không gian con, khoảng cách từ điểm dữ liệu đến không gian con là nhỏ nhất: $argmin_u \sum_{k=1}^{N} d_{u,k}^2$

2. Cực đại hóa phương sai:

- Tìm vector u đầu tiên cho ra phương sai lớn nhất. (Tối ưu có ràng buộc và dùng Lagrangian để giải)
- Giả sử đã tìm ra M vector đầu tiên: Các vector đều có chiều dài 1 và trực giao nhau đôi một.
- Tìm vector thứ M+1 sao cho trực giao với M vector trước và có phương sai lớn nhất.

Công thức tính định thức ma trận

Ma trận 2×2:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Ma trận 3×3 (quy tắc Sarrus):

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = aei + bfg + cdh - ceg - bdi - afh$$

a. Khi k = 1:

- Mục tiêu: Tìm u để phương sai của dữ liệu chiếu lên u là lớn nhất.
- Điểm trung tâm của dữ liệu: $\mu = \frac{1}{N} \sum_{n=1}^{N} x_n$. $z_n = x_n - \mu$ là điểm tương ứng của x_n sau khi dịch về μ.
- Variance của dữ liệu chiếu lên u: $\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} (u^T z_n)^2 = \frac{1}{N} \sum_{n=1}^{N} (u^T z_n)(z_n^T u) = \frac{1}{N} \sum_{n=1}^{N} u^T [z_n z_n^T] u = \frac{1}{N} \sum_{n=1}^{N} u^T [(x_n - \mu)(x_n - \mu)^T] u = u^T S u$.
- Covariance matrix (ma trận hiệp phương sai): $S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T = \frac{1}{N} Z^T Z = \frac{1}{N} (X - \mu)^T (X - \mu)$. S là ma trận vuông, đối xứng $D \times D$, bán định dương.
- Bài toán tối ưu: $argmax_u u^T S u$ với ràng buộc $u^T u = 1$. (S bán định dương => hàm mục tiêu lồi. Giải bằng pp nhân tử Lagrange).
- Lagrangian: $\mathcal{L}(u, \lambda) = u^T S u - \lambda(u^T u - 1)$. Đạo hàm theo u và đặt bằng 0: $\frac{\partial \mathcal{L}}{\partial u} = 0 \Rightarrow Su - \lambda u = 0 \Rightarrow Su = \lambda u$.
- Bài toán trở thành bài toán tìm eigenvector và eigenvalue của ma trận S: $Su_1 = \lambda_1 u_1$. Nghiệm u_1 là eigenvector cho variance lớn nhất λ_1.
- Tính chất của eigenvalues và eigenvectors của S: $\lambda_1 = u_1^T S u_1$ (phương sai lớn nhất).

b. Khi k = M:

- Bài toán tối ưu: $u_M = argmax_u u^T S u$ với ràng buộc $u^T u = 1$ và $u^T u_k = 0$ với $k = 1, 2, \ldots, (M-1)$.
- Mục tiêu: Tìm vector u_M sao cho trực giao với tất cả các vector trước và khi chiếu dữ liệu lên hướng này có phương sai lớn nhất (Vẫn $\leq \lambda_1, \lambda_2, \ldots, \lambda_{M-1}$). Độ lớn của u_M là 1.
- Giả thiết: Đã tìm được cặp u_k, λ_k với $k = 2, \ldots, M$ từ bài toán tối ưu ở trên.

c. Khi k = M+1:

- Bài toán tối ưu và mục tiêu: Tương tự như trên nhưng thay M bằng M+1 và M-1 bằng M. Cần suy ra bài toán tối ưu ở đây có nghiệm.
- Lagrangian: $\mathcal{L}(u, \lambda, \mu) = u^T S u - \lambda(u^T u - 1) - \sum_{k=1}^{M} \mu_k (u^T u_k)$.
- Đạo hàm theo u: $\frac{\partial \mathcal{L}}{\partial u} = 2Su - 2\lambda u - \sum_{k=1}^{M} \mu_k u_k$.
- Điểm làm lagrangian cực tiểu là nghiệm khi đạo hàm bằng 0: $2Su - 2\lambda u - \sum_{k=1}^{M} \mu_k u_k = 0$.
- Ở phương trình này, $\mu_k = 0$ vì u trực giao với tất cả u_k. Vậy ta nhân hai vế với u_k và có lại phương trình $Su = \lambda u$.

2. Giải thuật thu giảm số chiều:

- Tính ma trận hiệp phương sai và tìm các cặp u_k, λ_k (numpy.linalg.eig, numpy.linalg.eigh).
- Chọn M eigenvector $e_1, e_2, \ldots, e_M$. Chọn M bằng kiểm thử chéo.
- Chiếu dữ liệu gốc lên M vector để thu dữ liệu mới.

3. Singular Value Decomposition (SVD):

- SVD là kĩ thuật tổng quát của PCA.
- PCA: Tính ma trận hiệp phương sai S rồi phân rã eigen trên S.
- SVD: Phân rã SVD trên chính ma trận gốc (X), quá trình phân rã ổn định hơn.
- Ma trận X được SVD phân rã thành $X = USV^T$. Với $U(N \times N)$: có cột là eigenvectors của XX^T. $V(D \times D)$: có cột là eigenvectors của $X^T X$. $S(N \times D)$: ma trận có đường chéo chính là singular values xếp từ lớn đến nhỏ. (Singular Value là căn bậc 2 của eigenvalue)

Giải thuật

- Tính $Z = X - m^T$ với m là total mean = $\frac{1}{N} \sum_{n=1}^{N} x_n$.
- Dùng SVD phân rã $Z = USV^T$ (numpy.linalg.svd).
- Chọn M vectors đầu tiên của V tương ứng M singular values lớn nhất tạo $\hat{V}$.
- Chiếu dữ liệu Z lên M eigenvectors: $X_{pca} = Z\hat{V}$.

4. Tổng kết

PCA:

- $A = UDU^T = \sum_{i=1}^{r} \lambda_i u_i u_i^T$.
- A: ma trận hiệp phương sai.
- r: hạng của A.
- Thu giảm chiều: Chỉ giữ lại M eigenvector và chiếu X lên các vector này.

SVD:

- $A = USV^T = \sum_{i=1}^{r} s_i u_i v_i^T$.
- A: ma trận bất kỳ.
- r: hạng của A.
- Thu giảm chiều: Chỉ giữ lại M eigenvector và xấp xỉ X lên các vector này.

5. PCA qua API

```
1. import numpy as np
2. from sklearn.decomposition import PCA
3. X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
4. pca = PCA(n_components=2)
5. pca.fit(X)
6. print(pca.explained_variance_ratio_)
7. print(pca.singular_values_)
```

1. Giới thiệu về LDA

1. Đầu vào:

- X có kích thước $N \times D$, D rất lớn.
- t_k là nhãn lớp, có C lớp khác nhau.

2. Mục tiêu:

- Giảm số chiều từ D xuống M và $M \leq C - 1$.
- Dữ liệu sau khi giảm có tính phân tách cao nhất.

2. LDA qua API (Scikit-learn)

1. from sklearn import datasets
2. from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
3. iris = datasets.load_iris()
4. X = iris.data
5. y = iris.target
6. target_names = iris.target_names
7. lda = LinearDiscriminantAnalysis(n_components=2)
8. X_r2 = lda.fit(X, y).transform(X)

3. Bài toán tối ưu

1. Các thành phần:

- Tâm của dữ liệu thuộc mỗi lớp: $m_1 = \frac{1}{N_1}\sum_{n\in C_1} x_n$, $m_2 = \frac{1}{N_2}\sum_{n\in C_2} x_n$
- Khoảng cách giữa 2 tâm khi chiếu lên vector w: $d_{12} = w^T S_B w$
- Between-class covariance matrix: $S_B = (m_2 - m_1)(m_1 - m_2)^T$
- Variance của lớp 1 sau khi chiếu lên w: $\sigma_1^2 = w^T S_{W1} w$
- Within-class covariance matrix: $S_{W1} = \sum_{n\in C_1}(x_n - m_1)(x_n - m_1)^T$
- Within-class covariance matrix của cả 2 lớp: $S_W = S_{W1} + S_{W2}$
- Within-class variance: $\sigma^2 = w^T S_W w$

2. Hàm mục tiêu (Quan điểm của Fisher):

Tìm hướng $w^* = \arg\max_w J(w)$ để cực đại hóa tỷ số:

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

3. Tìm nghiệm:

- Đạo hàm $\nabla_w J = 0 <=> S_B w = \lambda S_W w$, $\lambda = \frac{w^T S_B w}{w^T S_W w}$
- => Phương trình eigenvector: $S_W^{-1} S_B w = \lambda w$
- Chúng ta chỉ cần hướng của w, nên dùng $S_B w = (m_2 - m_1)(m_2 - m_1)^T w$
- Hướng của vector w là $S_W^{-1}(m_2 - m_1)$; Cần chuẩn hóa w để có độ dài đơn vị.

4. Trường hợp có C lớp

1. Các thành phần:

- Danh sách các vector cần tìm: $w_1, w_2, \ldots, w_M$.
- Ma trận W (DxM) chứa các vector: $W = [w_1, w_2, \ldots, w_M]$.
- Between-class var: $trace(W^T S_B W) = \sum_{m=1}^{M} w_m^T S_B w_m$.
- Within-class var: $trace(W^T S_W W) = \sum_{m=1}^{M} w_m^T S_W w_m$.

2. Hàm mục tiêu và tìm nghiệm:

- Hàm mục tiêu: $J(W) = \frac{trace(W^T S_B W)}{trace(W^T S_W W)}$.
- Mục tiêu: $W^* = \arg\max_W J(W)$.
- Đạo hàm $\nabla_W J = 0 <=> S_B W = \lambda S_W W$.
- PT eigenvector: $S_W^{-1} S_B W = \lambda W$.
- Điều kiện để S_W khả đảo: X phải có ít nhất D điểm độc lập tuyến tính và $N \geq D$.
- M chiều tìm được là M eigenvector của $S_W^{-1} S_B$ ứng với M eigenvalue lớn nhất.

3. Tính toán các ma trận:

- Tâm của tất cả các điểm dữ liệu: $m = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{1}{N}\sum_{k=1}^{C} N_k m_k$
- Between-class scatter matrix: $S_B = \sum_{k=1}^{C} N_k (m_k - m)(m_k - m)^T$
- Within-class scatter matrix: $S_W = \sum_{k=1}^{C}\sum_{n\in C_k}(x_n - m_k)(x_n - m_k)^T$

Tìm ma trận chiếu W để tối đa hóa tỷ số định thức:

4. Cách lựa chọn số chiều M:

- Khi có C lớp, M lớn nhất là C-1 vì hạng S_B tối đa là C-1.
- **Vì:** Ma trận S_B là tổng có trọng số của C ma trận có hạng là 1.
- Bất kỳ m_k nào cũng có thể tính qua C-1 vector m_k còn lại.
- và $N_k = N - \sum_{i=1}^{C|i\neq k} N_i$.

5. Giải thuật:

- Tính S_B và S_W.
- Tính $A = S_W^{-1} S_B$.
- Sử dụng SVD để tìm ra M, M < C - 1, vector w_m với m = 1, 2, ..., M.
- Tính $Z = X - m^T$ (cần broadcast) với m là total mean = $\frac{1}{N}\sum_{n=1}^{N} x_n$.
- Dữ liệu sau khi thu giảm chiều: $\hat{X} = ZW$ với W chứa M vector w_m.

1. Genetic Algorithm (GA)

1. Khái niệm cốt lõi:

- **Học là tìm kiếm:** Tìm giả thuyết tốt nhất trong không gian giả thuyết thông qua các thế hệ.
- Thế hệ giả thuyết tiếp theo được tạo ra từ đột biến và lai ghép các giả thuyết tốt của thế hệ hiện tại.

2. Ưu điểm:

- Phương pháp bền vững (robust) để thích nghi.
- Xử lý được không gian giả thuyết chứa các phần tương tác phức tạp.
- Dễ dàng song song hóa (Parallelized).

3. Giải thuật GA tiêu chuẩn:

- **B1. Khởi tạo:** Quần thể P gồm p giả thuyết ngẫu nhiên.
- **B2. Đánh giá:** Tính $Fitness(h)$ cho mỗi $h \in P$.
- **B3. While** $\max_{h\in P} Fitness(h) < Fitness\_threshold$ **do:**
  - Tạo thế hệ mới (New Generation).
  - Đánh giá lại $Fitness$.

2. Tạo thế hệ mới

1. Chọn lọc (Selection):

- Chọn $(1 - r)p$ giả thuyết từ P để thêm vào thế hệ mới.
- **Xác suất chọn (Probabilistic Selection):**

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{h\in P} Fitness(h)}$$

(Giả thuyết tốt hơn có cơ hội được chọn cao hơn).

2. Lai ghép (Crossover):

- Chọn $(r/2)p$ cặp giả thuyết từ P dựa theo $Pr(h)$.
- Với mỗi cặp (h_1, h_2), áp dụng toán tử lai ghép để tạo 2 cá thể con.
- Thêm tất cả con vào thế hệ mới.

3. Đột biến (Mutation):

- Chọn ngẫu nhiên $m\%$ các giả thuyết vừa thêm vào với phân phối đều.
- Với mỗi giả thuyết, đảo ngược (invert) 1 bit ngẫu nhiên trong biểu diễn của nó.

3. Biểu diễn Giả thuyết

1. Biểu diễn Luật phân loại (Classification Rule):

- Dạng bit string: IF expr(A1)∧...∧expr(An) THEN C = c.
- Mã hóa thành chuỗi bit (Bit string).
- Độ dài chuỗi bit phụ thuộc vào số lượng thuộc tính và số giá trị có thể có của chúng.

2. Ví dụ Mã hóa (Example):

- Thuộc tính *Wind* (Strong, Weak) → 2 bit.
- `Wind = Strong` → `10`; `Wind = Weak` → `01`; Không quan tâm → `11`.
- Chuỗi đầy đủ ghép các thuộc tính: `Outlook | Wind | PlayTennis`.
- Ví dụ: `111` (Outlook) `10` (Strong) `10` (Yes) → `1111010`.

3. Hàm Fitness: $Fitness(h) = (correct(h))^2$ Với $correct(h)$ là phần trăm số điểm dữ liệu phân loại đúng bởi giả thuyết h.

4. Các Toán tử Lai ghép Chi tiết

1. Lai ghép một điểm (Single-point):

- Chọn 1 điểm cắt ngẫu nhiên trên chuỗi.
- Tráo đổi phần đuôi của 2 cha mẹ cho nhau.
- Ví dụ: 010|00 và 111|11 → 010|11 và 111|00.

2. Lai ghép hai điểm (Two-point):

- Chọn 2 điểm cắt.
- Tráo đổi đoạn giữa 2 điểm cắt đó.
- Ví dụ : 01|00|10 và 11|11|01 → 01|11|10 và 11|00|01.

3. Lai ghép đồng nhất (Uniform):

- Xét từng vị trí bit.
- Bit của con được chọn từ cha hoặc mẹ với xác suất như nhau.
- Ví dụ: 11101001 và 00001010 → 10001000 và 01110110.

4. Chuỗi bit độ dài thay đổi (Variable-length):

- Áp dụng khi giả thuyết có số lượng luật khác nhau hoặc cấu trúc động.
- Cần căn chỉnh các đoạn gen tương ứng (ví dụ: A_1, A_2, C) trước khi lai ghép để đảm bảo tính hợp lệ của con sinh ra.
- Ví dụ: A1|A2|C và A1|C|A2 → A1|A2|A2 và A1|C|C.

1. Tổng quan Ensemble Learning

1. Định nghĩa:

- Ensemble Learning là phương pháp kết hợp nhiều mô hình học máy để tạo ra một mô hình dự đoán tốt hơn.

2. Các hướng tiếp cận chính:

- **Averaging (Trung bình hóa):** Huấn luyện nhiều mô hình độc lập rồi lấy trung bình dự đoán (VD: Bagging).
- **Sequential (Tuần tự):** Huấn luyện chuỗi các mô hình, mô hình sau sửa lỗi cho mô hình trước (VD: Boosting).
- **Selection (Chọn lựa):** Chọn một mô hình tốt nhất để dự đoán dựa trên đầu vào cụ thể.

2. Bagging (Bootstrap Aggregating)

1. Bootstrap Data Sets: Từ tập dữ liệu gốc X (N mẫu), tạo ra tập X_B bằng cách lấy mẫu ngẫu nhiên N lần **có hoàn lại** sao cho một số điểm có thể lặp lại trong X_B, một số có thể vắng mặt.

2. Quy trình Huấn luyện:

- Tạo M tập dữ liệu bootstrap khác nhau.
- Huấn luyện mô hình trên mỗi tập dữ liệu để có M mô hình cơ sở $y_m(x)$ cho dataset m.

3. Kết hợp (Committee Prediction):

- Lấy trung bình cộng các dự đoán (cho bài toán hồi quy):

$$y_{COM}(x) = \frac{1}{M}\sum_{m=1}^{M} y_m(x)$$

4. Hiệu quả lý thuyết: Nếu lỗi của các mô hình có trung bình bằng 0 và không tương quan, sai số bình phương trung bình của Bagging sẽ giảm đi M lần so với mô hình đơn lẻ.

3. Boosting (AdaBoost)

1. Nguyên lý:

- Huấn luyện tuần tự nhiều mô hình cơ sở.
- Mỗi mô hình tập trung weight cao hơn vào các điểm dữ liệu mà các mô hình trước đó phân loại sai.
- Kết quả cuối cùng là sự bầu chọn đa số có trọng số của tất cả mô hình cơ sở.

2. Thuật toán AdaBoost (Chi tiết):

- B1. Khởi tạo: Trọng số $w_n^{(1)} = 1/N$ cho mỗi điểm dữ liệu.
- B2. Lặp $m = 1 \dots M$:
  - Huấn luyện $y_m(x)$ cực tiểu hóa lỗi có trọng số $J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)$.
  - Tính tỉ lệ lỗi: $\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$.
  - Tính hệ số tín nhiệm: $\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$.
  - Cập nhật trọng số:

$$w_n^{(m+1)} = w_n^{(m)} \exp\{-\alpha_m I(y_m(x_n) \neq t_n)\}$$

- B3. Dự đoán bằng mô hình cuối cùng:

$$Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$$

4. Các phương pháp khác

- Đưa ra kết quả chỉ khi quá nửa số mô hình cơ sở đồng ý.
- Xác suất ensemble chọn kết quả đúng là phân phối nhị thức: $\sum_{k=T/2+1}^{T} \binom{T}{k} p^k (1-p)^{T-k}$ với p là xác suất mô hình cơ sở đúng và T là số mô hình cơ sở.
- Nếu p > 0.5, xác suất này tăng dần về 1 khi T tăng đến vô cùng