

## 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

$$\mathbf{x} \rightarrow \phi(\mathbf{x}) = [\phi_0(\mathbf{x}), \dots, \phi_{M-1}(\mathbf{x})]$$

### Đánh giá mô hình

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

### Các độ đo

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

$$RMSE = \sqrt{MSE}$$

### Hạn chế quá khứ

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

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

### 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:

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

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  $(x, y)$ :

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

Với  $N$  điểm dữ liệu:

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

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

Hàm này còn được gọi là cross-entropy.

### Tìm hệ số của mô hình

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

### Giải thuật lặp với đạo hàm bậc 2

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

$R$  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 vectơ one-hot kích thước  $K$ .

### Mô hình tuyến tính với Softmax

### Mô hình dự báo

Ma trận tham số của mô hình:

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

Các bước tính toán:  $Z = \mathbf{X} \mathbf{W}^T$  ( $N \times K$ ),

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

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

### Dự đoán

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

### Output Layer

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

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

### Function Composition View

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

### Fully Connected (Linear) Layer

Single sample:  $y = Wx + b$

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

### 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:

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

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

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  $p_i = \sigma(z_i)$ :

$$L_{BCE} = -\frac{1}{n} \sum_{i=1}^n [y_i \log p_i + (1 - y_i) \log(1 - p_i)].$$

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

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

### 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 (SGD)

Initialize parameters  $\theta$ .

For epoch = 1 to  $E$  do:

Shuffle dataset and create mini-batches.

For each mini-batch  $(X, y)$  do:

Perform a forward pass.

Compute loss  $L$ .

Perform a backward pass and compute gradient  $\nabla_\theta L$ .

Update parameters:  $\theta \leftarrow \theta - \eta \nabla_\theta L$ .

### 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 gradient update

## 6 SVM primal problem

### Đầu vào

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

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

**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:  $\text{class}(\mathbf{x}) = \text{sign}(y(\mathbf{x}))$

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

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

### Hàm mục tiêu

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

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

### Bài toán gốc:

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

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

### Giải bằng thư viện CVXOPT

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

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

## 7 SVM dual problem

### Hàm Lagrangian

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

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

### Điều kiện KKT

**(KKT-1)** DK 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)** DK 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 Lagrangian, 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

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

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

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  $S = \{n : \alpha_n > 0\}$ ,  
 $y(x) = \sum_{n \in S} \alpha_n t_n x_n^T x + b$ .

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

### 8 SVM soft margin

#### Ràng buộc mới

$t_n(w^T x_n + b) \geq 1 - \xi_n, \quad n = 1, \dots, N$ ,  
 $\xi_n \geq 0$ .

#### Hàm mục tiêu mới

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

#### Bài toán tối ưu

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

#### 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

$$\begin{aligned} w = & \sum_{n=1}^N \alpha_n t_n x_n, \quad \sum_{n=1}^N \alpha_n t_n = 0, \\ 0 \leq \alpha_n \leq & C. \end{aligned}$$

#### Bài toán đối ngẫu

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{r=1}^N \sum_{c=1}^N \alpha_r \alpha_c t_r t_c x_r^T x_c \\ - & \sum_{n=1}^N \alpha_n \end{aligned}$$

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

#### Công thức dự báo

Sau khi tìm được  $\alpha$  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.

Nhân dự báo: label = 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

Bài toán đối ngẫu của SVM lè mềm:

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

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:  $b = \frac{1}{N_M} (t_M - K_{MS}[\alpha_S \odot t_S])^T \mathbf{1}$

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

Nhân dự đoán: label = sign( $y$ )

**Phương pháp Kernel:** Tính giá trị  $\langle \Phi(x_i), \Phi(x_j) \rangle$  thông qua 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

$$K_{Gram} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$$

#### Các kernel thông dụng

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:  $k(x, x') = \tanh(\gamma x^T x' + r)$

**Thiết kế Kernel:**  $k(x_i, x_j)$  lớn nếu  $x_i, x_j$  cùng lớp;  $k(x_i, x_j)$  nhỏ nếu khác lớp

### 10 PCA

Tập dữ liệu  $X \in \mathbb{R}^{N \times D}$ ;

**Mục tiêu:** Giảm số chiều từ  $D$  xuống  $M$  với  $M \ll D$ ; Các đặc trưng mới không còn tương quan tuyến tính.

**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:

$$S = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)(x_n - \mu)^T$$

$Au = \lambda u$ ,  $u$  là eigenvector,  $\lambda$  là eigenvalue.

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

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

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

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:  $\hat{U} = [u_1, u_2, \dots, u_M]$

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

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

**SVD:** Phân rã SVD:  $X = U S V^T$

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.

### 11 LDA

Tập dữ liệu:  $X \in \mathbb{R}^{N \times D}$ , với  $D$  thường rất lớn.  $t_k \in \{1, 2, \dots, C\}$  là nhãn lớp của điểm dữ liệu thứ  $k \Rightarrow$  Giảm số chiều từ  $D$  xuống  $M$ , với  $M \leq C - 1$ . Dữ liệu có độ phân tách giữa các lớp là lớn nhất.

#### Tâm của mỗi lớp

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

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

#### Trường hợp có $C$ lớp

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

#### Số chiều tối đa: $M \leq C - 1$

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

**Giải thuật LDA:** Tính  $S_B$  và  $S_W$ . Tính  $A = S_W^{-1} S_B$ . Thực hiện SVD hoặc eigen-decomposition. Chọn  $M$  eigenvector tương ứng với eigenvalue lớn nhất. Chiếu dữ liệu:  $\hat{X} = (X - \mu^T) W$ .

### 12 Ensemble

Variance of ensemble regression:

$$\text{Var}\left(\frac{1}{M} \sum_{m=1}^M \hat{y}^{(m)}\right) \approx \frac{1}{M^2} \sum_{m=1}^M \text{Var}(\hat{y}^{(m)})$$

#### Bagging (Bootstrap Aggregating)

Training dataset  $D = \{(x_i, y_i)\}_{i=1}^n$ ,

- (1) Draw  $M$  bootstrap datasets by sampling  $n$  points with replacement from  $D$ .
- (2) Train base learner on each bootstrap to obtain models  $h_1, h_2, \dots, h_M$ .
- (3) Combine predictions of all models:

$$\hat{y}(x) = \begin{cases} \frac{1}{M} \sum_{m=1}^M h_m(x), & \text{regression}, \\ \text{majority vote}, & \text{classification}. \end{cases}$$

**Random Forest:** (1) Sample bootstrap dataset from train set. (2) Grow tree by recursively splitting nodes. (3) At each split, randomly select subset of features  $F_{\text{sub}} \subset \{1, \dots, d\}$ . (4) Choose best split using only features in  $F_{\text{sub}}$ .  $|F_{\text{sub}}| = \sqrt{d}$  for classification,  $|F_{\text{sub}}| = d/3$  for regression.

**Boosting:** train models sequentially, where each model focuses on samples misclassified by previous ones. 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. final classifier is  $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$ , where  $\alpha_t$  is determined by weighted classification error of  $h_t$ .

**Gradient Boosting:** views ensemble construction as gradient descent in function space. Each iteration, new weak learner is fitted to negative gradient of loss function with respect to current predictions.

**Voting:** majority or probability averaging

**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 meta-model.

### 13 Genetic algorithm

A solution is encoded as a chromosome. Binary encoding Real-valued encoding Permutation encoding Tree encoding

#### Fitness Function

Evaluates quality of solution  $x$ . For minimization problems, a transformation is typically applied,  $f_{\text{max}}(x) = \frac{1}{1 + f_{\text{min}}(x)}$

**Selection:** Roulette wheel selection; Rank selection; Tournament selection; Elitism

**Crossover:** Single-point crossover; Two-point crossover; Uniform crossover; Arithmetic crossover (for real-valued encoding)

**Mutation:** Bit flipping for binary encoding; Gaussian or uniform noise for real-valued encoding; Swap or inversion for permutation encoding

**GA:** (1) Init population of  $N$  individuals (2) Eval fitness each individual (3) Repeat: (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 next generation (with optional elitism)

Break conditions: max # of gen, fitness convergence, stagnation, or time limits.

**Parameters and Tuning:** Population size ( $N = 20-200$ ); Crossover probability ( $p_c = 0.6-0.9$ ); Mutation probability ( $p_m = 0.001-0.1$ )

**Variants and Extensions:** RCGA; DE; GP; Multi-objective Genetic Algorithms (NSGA-II)