1 Định nghĩa bài toán

1.1 Đầu vào

Cho tập dữ liệu gồm N mẫu:

- Tập các vector đặc trưng:

  $$\mathbf{x}_n = [x_{n,0}, x_{n,1}, \ldots, x_{n,D-1}]^T, \quad n = 1, \ldots, N$$

  với $x_{n,0} = 1$.

- Vector nhãn:

$$\mathbf{t} = [t_1, t_2, \ldots, t_N]^T$$

Dữ liệu được tổ chức dưới dạng:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \ldots & x_{1,D-1} \\ 1 & x_{2,1} & \ldots & x_{2,D-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \ldots & x_{N,D-1} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

1.2 Giả thiết

Giả sử nhãn được sinh theo mô hình:

$$t = h(\mathbf{x}) + \varepsilon \tag{1}$$

Trong đó:

- $h(\mathbf{x})$: hàm hồi quy tối ưu (chưa biết)
- $\varepsilon \sim \mathcal{N}(0, \sigma^2)$: nhiễu Gauss

Các mẫu dữ liệu và nhiễu được giả thiết là *i.i.d.*

1.3 Đầu ra

Mô hình hồi quy tuyến tính:

$$\hat{y}(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} \tag{2}$$

với:

$$\mathbf{w} = [w_0, w_1, \ldots, w_{D-1}]^T$$

2 Giải bài toán

2.1 Nguyên tắc chung

Bài toán được giải bằng cách:

- Xây dựng hàm hợp lý
- Cực đại hóa hợp lý (Maximum Likelihood)

2.2 Hợp lý cực đại

Với giả thiết nhiễu Gauss:

$$p(t_n | \mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}(t_n | \mathbf{w}^T \mathbf{x}_n, \beta^{-1}) \tag{3}$$

Hàm negative log-likelihood:

$$L(\mathbf{w}, \beta) = \beta E_D(\mathbf{w}) - \frac{N}{2} \log \beta + \frac{N}{2} \log(2\pi) \tag{4}$$

với:

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 \tag{5}$$

2.3 Nghiệm giải tích

Cực tiểu $E_D(\mathbf{w})$ cho nghiệm:

$$\mathbf{w}_{ML} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t} \tag{6}$$

Ước lượng phương sai nhiễu:

$$\beta_{ML}^{-1} = \frac{1}{N} \sum_{n=1}^{N} (t_n - \mathbf{w}_{ML}^T \mathbf{x}_n)^2 \tag{7}$$

2.4 Giải thuật lặp (Gradient Descent)

Cập nhật tham số:

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla E_D(\mathbf{w}) \tag{8}$$

3 Hồi quy cho quan hệ phi tuyến

Ánh xạ dữ liệu sang không gian đặc trưng mới:

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

Sau đó áp dụng hồi quy tuyến tính trên không gian mới.

4 Đánh giá mô hình

4.1 Dự báo

$$\hat{\mathbf{y}} = \mathbf{X} \mathbf{w}_{ML} \tag{9}$$

4.2 Các độ đo

Mean Squared Error:

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^{N} (t_n - \hat{y}_n)^2 \tag{10}$$

Root Mean Squared Error:

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{11}$$

5 Hạn chế quá khớp

5.1 Ridge Regression

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \tag{12}$$

Nghiệm:

$$\mathbf{w}_{ridge} = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t} \tag{13}$$

5.2 LASSO

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 + \lambda \sum_{m=1}^{M} |w_m| \tag{14}$$

6 Dự báo cho nhiều biến

Với K biến đầu ra:

$$\mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{T} \tag{15}$$

Trong đó:

- $\mathbf{T} \in \mathbb{R}^{N \times K}$
- $\mathbf{W} \in \mathbb{R}^{M \times K}$

7 Giới thiệu bài toán

Mục tiêu của bài toán là xây dựng một mô hình dự đoán từ tập huấn luyện để khi nhận một mẫu dữ liệu mới x, mô hình có thể dự đoán nhãn của mẫu đó.
Bài giảng này tập trung vào việc xây dựng **mô hình tuyến tính** cho bài toán phân loại hai lớp, trong đó đường biên phân lớp là tuyến tính.

7.1 Đầu vào

7.1.1 Dữ liệu đầu vào

Ma trận dữ liệu:

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,(M-1)} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,(M-1)} \end{bmatrix}$$

- Mỗi hàng tương ứng với một điểm dữ liệu
- X có kích thước $N \times M$

7.1.2 Tập nhãn

- Bài toán có hai nhãn
- Mỗi nhãn được mã hóa bằng chỉ số $\{0, 1\}$

7.1.3 Nhãn dữ liệu

$$t = (t_1, t_2, \ldots, t_N)^T$$

7.1.4 Quy ước

- X: ma trận dữ liệu, kích thước $N \times M$
- t: vector nhãn
- y: biểu diễn dạng số của nhãn
- N: số điểm dữ liệu
- M: số đặc trưng
- $\hat{y}$: giá trị dự báo từ mô hình

8 Phương pháp xây dựng mô hình

8.1 Ý tưởng

- Hai lớp: C_0 và C_1
- Mô hình dự báo xác suất x thuộc lớp C_1
- Xác suất thuộc lớp C_0 là $1 - \hat{y}$

Bên trong mô hình:

1. Sử dụng mô hình tuyến tính:

$$z = w^T x$$

2. Đưa z qua hàm sigmoid

8.2 Hàm sigmoid

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

8.3 Mô hình dự báo

$$\hat{y} = p(C_1|x, w) = \sigma(w^T x) \tag{16}$$

Quy tắc phân lớp:

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

9 Ước lượng tham số của mô hình

9.1 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:

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

Sử dụng **negative log-likelihood**:

$$L(w) = -\sum_{n=1}^{N} [y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n)] \tag{17}$$

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

9.2 Tìm hệ số của mô hình

Gradient của hàm mất mát:

$$\nabla L(w) = \sum_{n=1}^{N} (\hat{y}_n - y_n) x_n = X^T (\hat{y} - y) \tag{18}$$

9.3 Giải thuật lặp với đạo hàm bậc 2

Ma trận Hessian:

$$H = \nabla^2 L(w) = \sum_{n=1}^{N} \hat{y}_n (1 - \hat{y}_n) x_n x_n^T = X^T R X \tag{19}$$

Trong đó 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)

10 Giới thiệu bài toán

Bài toán phân loại nhiều lớp (multi-class classification) nhằm xây dựng một mô hình dự đoán nhãn của dữ liệu đầu vào, trong đó mỗi mẫu dữ liệu thuộc về một trong K lớp rời rạc.
Mục tiêu của bài toán là học được một mô hình từ tập huấn luyện để dự đoán chính xác nhãn của các mẫu dữ liệu mới.

11 Dữ liệu đầu vào

11.1 Ma trận dữ liệu

Giả sử tập dữ liệu đầu vào được biểu diễn bởi ma trận:

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,(M-1)} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,(M-1)} \end{bmatrix}$$

trong đó $X \in \mathbb{R}^{N \times M}$.

11.2 Nhãn và biểu diễn one-hot

Mỗi nhãn được mã hóa dưới dạng véctơ one-hot kích thước K. Ví dụ với $K = 4$:

Nhãn	Chỉ số	One-hot
Chó	0	[1, 0, 0, 0]
Mèo	1	[0, 1, 0, 0]
Chuột	2	[0, 0, 1, 0]
Thỏ	3	[0, 0, 0, 1]

12 Mô hình tuyến tính với Softmax

12.1 Mô hình dự báo

Ma trận tham số của mô hình:

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_K^T \end{bmatrix} \in \mathbb{R}^{K \times M}$$

Các bước tính toán:

$$Z = XW^T \quad (N \times K) \tag{20}$$

$$\hat{Y} = \text{softmax}(Z) \tag{21}$$

Hàm softmax được định nghĩa:

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

12.2 Dự đoán

Nhãn dự đoán của mẫu dữ liệu được xác định bằng:

$$\text{prediction} = \arg\max_k \hat{y}_k$$

13 Hàm mục tiêu và tối ưu

13.1 Hàm hợp lý

Xác suất của tập nhãn:

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

13.2 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.

14 Ước lượng tham số

14.1 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:

$$W \leftarrow W - \eta \Delta W$$

trong đó η là hệ số học.

15 Mở rộng cho đường biên phi tuyến

Để xử lý dữ liệu không phân tách tuyến tính, có thể:

- Biến đổi đặc trưng sang không gian mới
- Sử dụng hàm cơ sở đa thức
- Áp dụng mạng nơ-ron để học đặc trưng

16 Kết luận

Mô hình tuyến tính kết hợp với softmax là một phương pháp nền tảng cho bài toán phân loại nhiều lớp, dễ cài đặt, hiệu quả và là cơ sở cho các mô hình học sâu phức tạp hơn.

17 Linear Regression and Logistic Regression: Recap

17.1 Linear Regression (Single Output)

Given an input feature vector $x \in \mathbb{R}^d$, linear regression predicts a scalar output:

$$\hat{y} = w^\top x + b \tag{22}$$

The model is trained by minimizing the Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{23}$$

Limitation: Only linear relationships can be modeled.

17.2 Linear Regression (Multiple Outputs)

For multiple outputs $y \in \mathbb{R}^m$:

$$\hat{y} = W^\top x + b, \quad W \in \mathbb{R}^{d\times m} \tag{24}$$

17.3 Logistic Regression (Binary Classification)

Logistic regression models the probability:

$$\hat{p} = P(y=1|x) = \sigma(w^\top x + b), \quad \sigma(z) = \frac{1}{1+e^{-z}} \tag{25}$$

Binary Cross-Entropy loss:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)] \tag{26}$$

17.4 Softmax Regression (Multiclass)

For K classes:

$$\hat{p}_k = \frac{\exp(w_k^\top x + b_k)}{\sum_{j=1}^{K}\exp(w_j^\top x + b_j)} \tag{27}$$

Key limitation of linear models: decision boundaries are linear and cannot represent complex nonlinear patterns.

18 MLP: Computational Architecture View

An MLP consists of:

- A **feature transformer**: stacked linear layers with nonlinear activations.
- An **output head**:
  - Linear head for regression
  - Logistic or softmax head for classification

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

19 MLP: Mathematical Model

19.1 Forward Pass

Let $h^{(0)} = x$. Each hidden layer computes:

$$h^{(l)} = \phi\left(W^{(l)}h^{(l-1)} + b^{(l)}\right), \quad l = 1, \dots, L \tag{28}$$

where $\phi(\cdot)$ is a nonlinear activation function.

19.2 Output Layer

Regression:

$$\hat{y} = W^{(L+1)}h^{(L)} + b^{(L+1)} \tag{29}$$

Classification (Softmax):

$$\hat{p}_k = \frac{\exp(w_k^\top h^{(L)} + b_k)}{\sum_j \exp(w_j^\top h^{(L)} + b_j)} \tag{30}$$

19.3 Function Composition View

An MLP is a composition of functions:

$$f(x;\theta) = f^{(L+1)} \circ \phi \circ f^{(L)} \circ \cdots \circ \phi \circ f^{(1)}(x) \tag{31}$$

More layers imply higher representation power.

20 MLP: Layers

20.1 Fully Connected (Linear) Layer

For a single sample:

$$y = Wx + b \tag{32}$$

For a mini-batch $X \in \mathbb{R}^{B\times N}$:

$$Y = XW^\top + \mathbf{1}b^\top \tag{33}$$

20.2 Activation Functions

Sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{34}$$

Tanh

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{35}$$

ReLU

$$\text{ReLU}(z) = \max(0, z) \tag{36}$$

Leaky ReLU

$$\text{LReLU}(z) = \begin{cases} z, & z \geq 0 \\ \alpha z, & z < 0 \end{cases} \tag{37}$$

SiLU (Swish)

$$\text{SiLU}(z) = z\sigma(z) \tag{38}$$

21 MLP: Summary

- MLP extends linear models by introducing nonlinear feature transformations.
- Architecture: stacked fully connected layers + activations.
- Can approximate complex nonlinear functions.
- Powerful but prone to overfitting without sufficient data or regularization.

22 Giới thiệu bài toán

22.1 Đầu vào

Dữ liệu huấn luyện gồm:

- Ma trận dữ liệu:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,(M-1)} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,(M-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,(M-1)} \end{bmatrix}$$

  với $X \in \mathbb{R}^{N\times(M-1)}$.

- Véc-tơ nhãn:

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}, \quad t_n \in \{-1, +1\}$$

22.2 Giả thiết

Dữ liệu thuộc hai lớp có thể **phân tách tuyến tính**, tức tồn tại một siêu phẳng sao cho các điểm mang nhãn $+1$ và -1 nằm ở hai phía khác nhau.

22.3 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.

22.4 SVM với `scikit-learn`

Ví dụ sử dụng SVM trong `scikit-learn`:

```
from sklearn import svm
X = [[0, 0], [1, 1]]
y = [0, 1]
clf = svm.SVC()
clf.fit(X, y)
clf.predict([[2., 2.]])
```

23 Phương pháp xây dựng bộ phân lớp

23.1 Nguyên tắc

Quy trình gồm ba bước chính:

1. Chuyển về bài toán tối ưu có ràng buộc với mục tiêu cực đại lề (bài toán gốc).
2. Chuyển sang bài toán đối ngẫu (Dual problem).
3. Giải bài toán tối ưu bằng các thư viện tối ưu lồi như `CVXOPT`.

24 Bài toán gốc (Primal Problem)

24.1 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:

$$|d(\mathbf{x})| = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

24.2 Hàm quyết định

Hàm quyết định:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Quy tắc phân lớp:

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

24.3 Lề (Margin)

Lề trên tập huấn luyện:

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

24.4 Cực đại lề

Có thể chuẩn hóa sao cho:

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

24.5 Hàm mục tiêu

Cực đại lề tương đương với bài toán:

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

với ràng buộc:

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

24.6 Bài toán gốc

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{s.t.} \quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \quad \forall n$$

24.7 Giải bằng thư viện CVXOPT

Dạng chuẩn:

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T K\mathbf{x} + \mathbf{p}^T\mathbf{x} \quad \text{s.t. } G\mathbf{x} \leq \mathbf{h}$$

Trong đó:

$$\mathbf{x} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

25 Tổng kết và Câu hỏi

25.1 Tổng kết

- Số biến: M
- Số ràng buộc: N
- Phù hợp khi $M \ll N$
- Áp dụng cho dữ liệu khả tách tuyến tính

25.2 Câu hỏi

1. Véc-tơ $\mathbf{w}$ có ý nghĩa gì trong SVM?
2. Vì sao nhãn $\{-1, +1\}$ thuận lợi?
3. Độ rộng của lề bằng bao nhiêu?
4. Viết dạng ma trận của bài toán gốc.

Tài liệu tham khảo

- C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer.
- N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines*.
- `https://scikit-learn.org/stable/modules/svm.html`

26 Bài toán gốc

Cho tập dữ liệu huấn luyện

$$\{(x_n, t_n)\}_{n=1}^N, \quad x_n \in \mathbb{R}^M,\ t_n \in \{-1, +1\}.$$

Bài toán SVM hai lớp, khả tách tuyến tính được phát biểu như sau:

$$(w^*, b^*) = \arg\min_{w,b} \ \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad t_n(w^T x_n + b) \geq 1, \quad n = 1, \ldots, N. \tag{39}$$

Đây là một bài toán tối ưu lồi với:

- Hàm mục tiêu lồi và khả vi
- Các ràng buộc bất đẳng thức lồi

27 Kiến thức liên quan

27.1 Tối ưu lồi

Bài toán (39) thuộc lớp *convex optimization*. Theo [**?**], nếu thỏa tiêu chuẩn Slater thì tồn tại *strong duality*.

27.2 Máy véc-tơ hỗ trợ

SVM tìm siêu phẳng phân lớp có *biên lớn nhất*, được xác định bởi các *véc-tơ hỗ trợ* (support vectors).

28 Bài toán đối ngẫu

28.1 Hàm Lagrangian

Hàm Lagrangian của bài toán (39) là:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} \alpha_n \big[t_n(w^T x_n + b) - 1\big], \tag{40}$$

với

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

28.2 Điều kiện KKT

Bài toán thỏa hệ điều kiện KKT:

- **(KKT-1)** Điều kiện 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)** Điều kiện bù:

$$\alpha_n[1 - t_n(w^T x_n + b)] = 0$$

29 Xây dựng hàm đối ngẫu

Lấy đạo hàm theo w và b:

$$\frac{\partial L}{\partial w} = w - \sum_{n=1}^{N} \alpha_n t_n x_n = 0 \tag{41}$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{N} \alpha_n t_n = 0 \tag{42}$$

Suy ra:

$$w = \sum_{n=1}^{N} \alpha_n t_n x_n. \tag{43}$$

Thay vào Lagrangian, ta thu được hàm đối ngẫu:

$$g(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{r=1}^{N} \sum_{c=1}^{N} \alpha_r \alpha_c t_r t_c x_r^T x_c. \tag{44}$$

30 Bài toán đối ngẫu

Bài toán đối ngẫu tương đương:

$$\begin{aligned} \alpha^* = \arg\min_{\alpha} \ & \frac{1}{2} \alpha^T K \alpha - \mathbf{1}^T \alpha \\ \text{s.t.} \quad & \alpha_n \geq 0, \quad n = 1, \ldots, N, \\ & \sum_{n=1}^{N} \alpha_n t_n = 0, \end{aligned} \tag{45}$$

trong đó

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

31 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**.

32 Công thức dự báo

Với tập véc-tơ hỗ trợ $S = \{n : \alpha_n > 0\}$,

$$y(x) = \sum_{n \in S} \alpha_n t_n x_n^T x + b. \tag{46}$$

Nhãn dự báo:

$$\hat{y} = \text{sign}(y(x)).$$

33 Tổng kết

- Bài toán SVM gốc là bài toán tối ưu lồi
- Bài toán đối ngẫu có dạng quy hoạch toàn phương
- Nghiệm phụ thuộc vào một số ít véc-tơ hỗ trợ
- Huấn luyện và dự báo chỉ cần tích vô hướng

34 Ôn lại: Hai lớp khả tách tuyến tính

Cho tập dữ liệu huấn luyện

$$\{(x_n, t_n)\}_{n=1}^{N}, \quad x_n \in \mathbb{R}^M, \ t_n \in \{-1, +1\}.$$

Bài toán SVM hard-margin được phát biểu như sau:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & t_n(w^T x_n + b) \geq 1, \quad n = 1, \ldots, N. \end{aligned} \tag{47}$$

Mô hình này chỉ áp dụng khi dữ liệu khả tách tuyến tính.

35 Giới thiệu bài toán không khả tách tuyến tính

Trong thực tế, dữ liệu thường **không khả tách tuyến tính**. Khi đó:

- Tập nghiệm khả thi của hard-margin là rỗng.
- Không tồn tại w, b thỏa tất cả các ràng buộc.

Do đó, cần mở rộng mô hình bằng cách cho phép một số điểm vi phạm ràng buộc.

36 Nguyên tắc Soft Margin

Ý tưởng chính:

1. Nới lỏng ràng buộc bằng biến phạt ξ_n.
2. Phạt các điểm nằm sai vị trí thông qua hàm mục tiêu.

Mô hình này được gọi là **Soft Margin SVM**.

37 Bài toán gốc với lề mềm

37.1 Ràng buộc mới

$$t_n(w^T x_n + b) \geq 1 - \xi_n, \quad n = 1, \ldots, N, \tag{48}$$

$$\xi_n \geq 0. \tag{49}$$

37.2 Hàm mục tiêu mới

$$f_0(w, b, \xi) = \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n, \tag{50}$$

trong đó $C > 0$ là siêu tham số điều chỉnh mức phạt.

37.3 Bài toán tối ưu

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n \\ \text{s.t.} \quad & t_n(w^T x_n + b) \geq 1 - \xi_n, \\ & \xi_n \geq 0, \quad n = 1, \ldots, N. \end{aligned} \tag{51}$$

38 Bài toán đối ngẫu

38.1 Hàm Lagrangian

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \alpha_n [t_n(w^T x_n + b) - 1 + \tag{52}$$

38.2 Điều kiện KKT

$$w = \sum_{n=1}^{N} \alpha_n t_n x_n, \tag{53}$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0, \tag{54}$$

$$0 \leq \alpha_n \leq C. \tag{55}$$

38.3 Bài toán đối ngẫu

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{r=1}^{N} \sum_{c=1}^{N} \alpha_r \alpha_c t_r t_c x_r^T x_c - \sum_{n=1}^{N} \alpha_n \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \\ & \sum_{n=1}^{N} \alpha_n t_n = 0. \end{aligned} \tag{56}$$

39 Công thức dự báo

Sau khi tìm được α và b, hàm quyết định là:

$$y(x) = \sum_{n \in S} \alpha_n t_n x_n^T x + b, \tag{57}$$

trong đó S là tập các véc-tơ hỗ trợ.
Nhãn dự báo:

$$\text{label} = \text{sign}(y(x)).$$

40 Cài đặt với CVXOPT

Bài toán đối ngẫu có dạng chuẩn:

$$\min_{\alpha} \quad \frac{1}{2}\alpha^T K \alpha + p^T \alpha \quad \text{s.t.} \quad G\alpha \leq h, \ A\alpha = b. \tag{58}$$

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

41 Kết luận

Bài báo đã trình bày:

- Mô hình SVM soft-margin cho dữ liệu không khả tách tuyến tính.
- Dẫn xuất bài toán đối ngẫu và điều kiện KKT.
- Công thức dự báo và hướng cài đặt bằng CVXOPT.

Soft Margin SVM là nền tảng quan trọng cho các phương pháp kernel và SVM phi tuyến.

42 Giới thiệu

SVM là phương pháp học có giám sát, nổi bật nhờ khả năng tổng quát hóa tốt. Ý tưởng chính của SVM là tìm siêu phẳng phân tách hai lớp dữ liệu với biên lớn nhất. Khi dữ liệu không phân tách tuyến tính, SVM có thể được mở rộng thông qua *phương pháp kernel*.

43 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:

$$\alpha^* = \arg\min_{\alpha} \frac{1}{2}\alpha^T K \alpha + p^T \alpha \tag{59}$$

với các ràng buộc:

$$G\alpha \leq h, \qquad A\alpha = b \tag{60}$$

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

43.1 Dự báo

Giá trị bias b được ước lượng bởi:

$$b = \frac{1}{N_M}\left(t_M - K_{MS}[\alpha_S \odot t_S]\right)^T \mathbf{1} \tag{61}$$

Hàm dự báo:

$$y = K_{BS}[\alpha_S \odot t_S] + b \tag{62}$$

Nhãn dự đoán:

$$\text{label} = \text{sign}(y) \tag{63}$$

44 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.

45 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:

$$k(x_i, x_j)$$

45.1 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:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geq 0$$

46 Huấn luyện và dự báo với Kernel

Ma trận Gram kernel:

$$K_{Gram} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix} \tag{64}$$

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

47 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.

48 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

49 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.

50 Kết luận

Bài báo đã trình bày SVM với lề mềm và phương pháp kernel, cho phép mở rộng khả năng phân lớp sang các bài toán phi tuyến. Phương pháp kernel giúp tránh việc tính toán trực tiếp không gian đặc trưng, đồng thời vẫn đảm bảo tính tối ưu của bài toán.

51 Giới thiệu

Trong nhiều bài toán học máy, dữ liệu đầu vào thường có số chiều rất lớn. Điều này dẫn đến các vấn đề như:

- Độ phức tạp tính toán tăng cao;
- Khoảng cách giữa các điểm dữ liệu trở nên kém ý nghĩa;
- Nguy cơ quá khớp (overfitting).

Hiện tượng này thường được gọi là *Curse of Dimensionality*. PCA được đề xuất nhằm giải quyết vấn đề trên bằng cách thu giảm số chiều dữ liệu.

52 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.

53 Kiến thức toán học liên quan

53.1 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:

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

53.2 Eigenvalue và Eigenvector

Với ma trận vuông A, bài toán eigen:

$$Au = \lambda u$$

trong đó u là eigenvector và λ là eigenvalue.

54 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.

55 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:

$$\max_u \quad u^T S u \quad \text{s.t.} \quad u^T u = 1$$

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.

56 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:

$$\hat{X} = \mu^T + X_{\text{PCA}}\hat{U}^T$$

57 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_)
```

58 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.

59 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.

60 Kết luận

PCA là một phương pháp mạnh mẽ và hiệu quả cho bài toán thu giảm số chiều. Dựa trên nền tảng đại số tuyến tính, PCA giúp loại bỏ nhiễu, giảm độ phức tạp và cải thiện hiệu suất của các mô hình học máy.

61 Giới thiệu về LDA

Linear Discriminant Analysis (LDA) là một phương pháp thu giảm số chiều có giám sát, trong đó thông tin nhãn của dữ liệu được sử dụng để tìm ra không gian chiếu sao cho các lớp được phân tách tốt nhất.

61.1 Đầu vào

Cho tập dữ liệu:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,D} \end{bmatrix}, \quad t = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

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.

61.2 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.

62 LDA qua API

Ví dụ sử dụng `Scikit-learn`:

```
from sklearn import datasets
from sklearn.discriminant_analysis imp

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

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

63 Bài toán tối ưu

63.1 Tâm của mỗi lớp

Với lớp k:

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

63.2 Between-class scatter matrix

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

63.3 Within-class scatter matrix

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

63.4 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:

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

64 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:

$$\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

65 Trường hợp có C lớp

65.1 Hàm mục tiêu tổng quát

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

65.2 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 $C - 1$.

65.3 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: $\hat{X} = (X - m^T)W$.

66 Kết luận

Linear Discriminant Analysis là một phương pháp mạnh mẽ cho thu giảm số chiều có giám sát, đặc biệt hiệu quả khi mục tiêu là phân loại. LDA không chỉ giúp giảm độ phức tạp tính toán mà còn cải thiện hiệu năng của mô hình học máy.

67 Introduction

Single learning models often suffer from limited representational power or high sensitivity to training data. These issues can lead to high bias, high variance, or both. Ensemble methods address these limitations by combining multiple learners into a single predictive model, following the principle known as the *wisdom of the crowd*.

68 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

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

69 Bagging (Bootstrap Aggregating)

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

69.1 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:

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

69.2 Properties

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

70 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.

70.1 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.

71 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.

71.1 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.

71.2 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.

72 Voting and Stacking

72.1 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.

72.2 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.

73 Comparison and Practical Consideration

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.

74 Conclusion

Ensemble methods provide a principled way to improve predictive performance by combining multiple models. Understanding their assumptions and trade-offs is essential for effective application in real-world machine learning problems.

75 Introduction

Genetic Algorithm (GA) is a metaheuristic optimization algorithm inspired by the process of natural selection. It was developed by John Holland in the 1970s and belongs to the broader class of Evolutionary Algorithms (EA).
GA is particularly effective when:

- The search space is large or complex
- The objective function is non-differentiable or discontinuous
- Multiple local optima exist
- Classical optimization techniques are ineffective

Unlike gradient-based methods, GA does not require derivative information and performs a parallel exploration of the search space.

76 Key Components of Genetic Algorithms

76.1 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)

76.2 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.

76.3 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.

76.4 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)

76.5 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

77 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.

78 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 $x = 15$.

79 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.

80 Advantages and Disadvantages

80.1 Advantages

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

80.2 Disadvantages

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

81 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

82 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.

83 Conclusion

Genetic Algorithms provide a powerful and flexible optimization framework for complex problems where traditional methods fail. Although not guaranteed to find the global optimum, proper encoding and parameter tuning enable GA to produce high-quality approximate solutions efficiently.