DDA3020 Notes

Course notes for CUHK-Shenzhen, Spring 2026

Hongyi Chen

Chapter 1: Introduction to ML

Included lectures

- Lecture 1: Course Overview
- Lecture 2: Probability & Information Theory
- Lecture 3: Linear Algebra Review
- Lecture 4: Basic Optimization Review

Lecture 1: Course Overview

Learning paradigms of ML: - **Supervised Learning:** Learn from teacher - **Unsupervised Learning:** Learn from oneself - **Reinforcement Learning:** Learn from interaction with environment (reward/punishment)

	Supervised Learning	**Unsupervised Learning**
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality Reduction

Lecture 2: Probability & Information Theory

1. Probability of discrete r.v.
2. Probability of continuous r.v.
3. Some popular distributions

3.1. **Bernoulli Distribution**

$x \in \{0, 1\} \rightarrow$ toss a coin, x = 1 (head) with probability μ, x = 0 (tail) with probability $1 - \mu$.

$$Bern(x|\mu) = \mu^x(1-\mu)^{1-x}$$

$$E[x] = \mu$$

$$Var[x] = \mu(1-\mu)$$

3.2. **Binomial Distribution**

Toss N coins, each follows Bernoulli distribution $p(x|\mu)$, count number of heads m.

$$Bin(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m}$$

$$E[m] = N\mu$$

$$Var[m] = N\mu(1-\mu)$$

3.3. **Gaussian/Normal Distribution**

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

For a D-dimensional vector **x**, the multivariate Gaussian distribution is defined as:

$$N(\mathbf{x}|\mu,\Sigma)=\frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}exp(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu))$$

where μ is the mean vector and Σ is the covariance matrix.

4. Information Theory

4.1. **Definition**
For a discrete r.v. with a finite alphabet, as follows:

$$X=\{x_1,x_2,...,x_n\}$$

We difine the Information as

$$I(x)=-logp(x)$$

Information measures the amount of "surprise" associated with an event. The less likely an event is, the more information it provides when it occurs.

4.2. **Entropy**
Entropy is the expected value of the information content of a random variable. It quantifies the uncertainty associated with a random variable.

$$H_p(X)=E[I(x_k)]=-\sum_{x_k\in X}p(x_k)logp(x_k)$$

Apparently, entropy is maximized when all outcomes are equally likely, i.e., $p(x_k)=\frac{1}{n}$ for all k.

4.3. **Cross Entropy**
Cross entropy is: when the real distribution is P, but we have to use another distribution Q to encode the information, how many bits are needed on average?

$$H_{p,q}(X)=-\sum_{x_k\in X}P(X=x_k)\cdot log(Q(X=x_k))$$

There are two properties of cross entropy:

- Non-negativity: $H_{P,Q}(X)\geq 0$
- Cross entropy is not smaller than entropy: $H_{P,Q}(X)\geq H_P(X)$, and the equality holds iff $P=Q$.

4.4. **Relative entropy: Kullback-Leibler (KL) Divergence** KL divergence measures the distance between two probability distributions.

$$D_{P,Q}(X)=\sum_{x_k\in X}P(X=x_k)log\frac{P(X=x_k)}{Q(X=x_k)}=H_{P,Q}(X)-H_P(X)$$

$$D_{P,Q}(X) = \int_{x \in X} p_X(x) log \left(\frac{p_X(x)}{q_X(x)} \right) dx$$

There are two properties of KL divergence:

- Non-negativity: $D_{P,Q}(X) \geq 0$
- Asymmetry: $D_{P,Q}(X) \neq D_{Q,P}(X)$

The cross entropy $H_{P,Q}(X)$ is the entropy of the distribution $H_P(X)$ plus the KL divergence $D_{P,Q}(X)$ between the two distributions.

$$H_{P,Q}(X) = H_P(X) + D_{P,Q}(X)$$

Lecture 3: Linear Algebra Review

1. Verctor, matrix, and their norms 1.1. Vector, matrix and their norms

   · Trace: $tr(A) = \sum_i A_{ii}$
   · Vector norms:
   – l_0 norm: $\|x\|_0 = \sum_i I(x_i \neq 0)$, number of nonzero elements in vector x.
   – l_1 norm: $\|x\|_1 = \sum_i |x_i|$
   – l_2 norm: $\|x\|_2 = \sqrt{\sum_i x_i^2}$
   – l_p norm: $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$
   · Matrix norms:
   – Frobenius norm: $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n x_{ij}^2} = \sqrt{tr(A^T A)}$
   – Spectral norm: $\|A\|_2 = \sigma_{max}(A)$, the largest singular value of A. (Singular Value Decomposition: $X = U\Sigma V^T$, $\Sigma = diag(\sigma_1, \sigma_2, ...)$)

2. Matrix inverse, determinant, independence 2.1. Matrix inverse For a square matrix A, if there exists a matrix B such that $AB = BA = I$, then B is called the inverse of A, denoted as A^{-1}.

   2.2. Determinant Via SVD:

   $$det(A) = \prod_i \sigma_i$$

   2.3. Linear Independence

   $$c_1 v_1 + c_2 v_2 + ... + c_n v_n = 0$$

   only holds for $c_1 = ... = c_n = 0$.

3. Systems of linear equations

   $$\mathbf{Xw} = \mathbf{y},$$

   where

   $$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

Lecture 4: Basic Optimization Review

1. Convex set 1.1. Affine line

$$x = \theta x_1 + (1-\theta)x_2, \theta \in R$$

1.2. Convex set

$$x = \theta x_1 + (1-\theta)x_2, \theta \in [0,1]$$

The convex set contains all line segments between any two points in the set.

2. Convex function 2.1. Definition f: $R^n \to R$ is convex if $domf$ is a convex set and for all $x_1, x_2 \in domf$ and

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta)f(x_2), \theta \in [0,1]$$

2.2. First-order condition If f is differentiable, then f is convex iff for all $x_1, x_2 \in domf$,

$$f(y) \geq f(x) + \nabla f(x)^T (y-x)$$

2.3. Second-order condition If f is twice differentiable, then f is convex iff for all $x \in domf$,

$$\nabla^2 f(x) \succeq 0$$

i.e., the Hessian matrix is positive semidefinite. 2.4. Jensen' s Inequality If f is convex, then for $0 \leq \theta \leq 1$,

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta)f(x_2)$$

More generally, for random variable X,

$$f(E[X]) \leq E[f(X)]$$

3. Convex Optimization Problem A convex optimization problem has the form:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, ..., m \\ & h_j(x) = 0, \quad j = 1, ..., p \end{array}$$

where $f_0, f_1, ..., f_m$ are convex functions and $h_1, ..., h_p$ are affine functions. 3.1. Local and Global Minimum **Theorem:** In a convex optimization problem, any local minimum is also a global minimum.

4. Unconstrained Minimization 4.1. Gradient Descent `Given a starting point x. repeat 1. Determine a descent direction delta_x 2. Choose a step size t > 0 3. Update x := x + t * delta_x until stopping criterion is satisfied.`

4.2. Line Search Method **Exact line search:** $t = argmin_{t \geq 0} f(x + t\Delta x)$ **Backtracking line search (inexact)** Starting at $t = 1$ Repeat $t := \beta * t$ until

$$f(x + t * \Delta x) \leq f(x) + \alpha * t * \nabla f(x)^T * \Delta x$$

5. Constrained Minimization: Lagrangian duality, KKT conditions 5.1. Lagrangian and Lagrange Dual Function Given a general constrained optimization problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, ..., m \\ & h_j(x) = 0, \quad j = 1, ..., p \end{array}$$

5.1. Lagrangian and Lagrange Dual Function The Lagrangian is defined as:

$$L(x, u, v) = f_0(x) + \sum_{i=1}^{m} u_i f_i(x) + \sum_{j=1}^{p} v_j h_j(x)$$

The Lagrange dual function is defined as:

$$g(u, v) = inf_x L(x, u, v)$$

The dual problem is:

$$\begin{array}{ll} \text{maximize} & g(u, v) \\ \text{subject to} & u \succeq 0 \end{array}$$

5.2. KKT Conditions A necessary condition for x^* to be optimal for general constrained optmization problem, a sufficient condition for x^* to be optimal for convex optimization problem:

- Stationarity: $0 \in \partial f(x) + \sum_{i=1}^{m} u_i \partial f_i(x) + \sum_{j=1}^{p} v_j \partial \ell_j(x)$
- Complementary slackness: $u_i f_i(x) = 0$ for all i
- Primal feasibility: $h_i(x) \leq 0; \ell_j(x) = 0$ for all i, j
- Dual feasibility: $u_i \geq 0$ for all i

Chapter 2: Classical Supervised Learning

Contents: Linear Regression, Logistic Regression, SVM, Decision Tree, Random Forest etc.

Included lectures

- Lecture 5: Linear Regression
- Lecture 6: Logistic Regression
- Lecture 7: Support Vector Machine (SVM)
- Lecture 8: Tree-based Methods

Lecture 5: Linear Regression

1. Basic Formulation Goal: Find a linear function that best fits the data points.

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

Obejective function (MSE):

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$$

2. Probabilistic Perspective Assume $y = \mathbf{w}^T\mathbf{x} + b + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$. Thus, the output y can be seen as a random variable, $p(y|\mathbf{x}, \mathbf{w}) = N(y|\mathbf{w}^T\mathbf{x} + b, \sigma^2)$.

Maximum Likelihood Estimation (MLE):

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w}} \log \mathcal{L}(\mathbf{w}; D),$$

$$\begin{aligned}\log \mathcal{L}(\mathbf{w}; D) &= \log\left(\prod_{i=1}^{m} p(y_i|\mathbf{x}_i, \mathbf{w})\right) = \sum_{i=1}^{m} \log \mathcal{N}(\mathbf{w}^\top\mathbf{x}_i, \sigma^2) \\ &= -m\log(\sigma(2\pi)^{\frac{1}{2}}) - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(y_i - \mathbf{w}^\top\mathbf{x}_i)^2.\end{aligned}$$

Removing constants w.r.t. $\mathbf{w}$, we have

$$\mathbf{w}_{MLE} = \arg\min_{\mathbf{w}} \sum_{i=1}^{m}(y_i - \mathbf{w}^\top\mathbf{x}_i)^2.$$

3. Analytical Solution The squared error function can be written compactly using $\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$ as:

$$\begin{aligned}J(\mathbf{w}) &= \mathbf{e}^\top\mathbf{e} = (\mathbf{X}\mathbf{w} - \mathbf{y})^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y}.\end{aligned}$$

Setting the gradient to zero, we have

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2\mathbf{X}^\top\mathbf{X}\mathbf{w} - 2\mathbf{X}^\top\mathbf{y} = 0.$$

Thus, if $\mathbf{X}^\top\mathbf{X}$ is invertible, the closed-form solution is

$$\text{Learning:} \quad \hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}.$$

$$\text{Prediction:} \quad \hat{y}(X_{\text{new}}) = \mathbf{X}_{\text{new}}\hat{\mathbf{w}}$$

4. Design Matrix with Bias Term

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

$$\mathbf{x}_i^T \mathbf{w} = [1, x_1, x_2, ..., x_d]_i \cdot [w_0, w_1, w_2, ..., w_d]^T$$

So, w_0 is the bias term.

5. Gradient Descent Method Update rule:

$$\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} J(\mathbf{w})$$

where η is the learning rate / step size, and

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}).$$

6. Closed-form vs. GD Closed-form: No hyperparameter or iteration, but needs matrix inversion $O(d^3 + m^3)$, not suitable for large-scale data. GD: $O(Tmd)$, works well when d is large.

7. Linear Regression with multiple outputs

$$\mathbf{f}_{\mathbf{W}}(\mathbf{X}) = \mathbf{X}\mathbf{W}$$

$$\begin{matrix} \text{Sample 1} \dashrightarrow \\ \vdots \\ \text{Sample m} \dashrightarrow \end{matrix} \underbrace{\begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix}}_{\mathbf{X}} \mathbf{W} = \underbrace{\begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,d} \end{bmatrix}}_{m \times (d+1)} \underbrace{\begin{bmatrix} w_{0,1} & \cdots & w_{0,h} \\ w_{1,1} & \cdots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,h} \end{bmatrix}}_{(d+1) \times h}$$

$$\begin{matrix} \text{Sample 1' s output} \dashrightarrow \\ \vdots \\ \text{Sample m' s output} \dashrightarrow \end{matrix} = \underbrace{\begin{bmatrix} f_{1,1} & \cdots & f_{1,h} \\ \vdots & \ddots & \vdots \\ f_{m,1} & \cdots & f_{m,h} \end{bmatrix}}_{m \times h}$$

$$\mathbf{J}(\mathbf{W}) = \text{trace}((E^T E)) = \text{trace}((\mathbf{X}\mathbf{W} - \mathbf{Y})^T (\mathbf{X}\mathbf{W} - \mathbf{Y}))$$

In general, the closed-form solution is

$$\text{Learning:} \quad \hat{\mathbf{W}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

$$\text{Prediction:} \quad \hat{\mathbf{Y}}(\mathbf{X}_{\text{new}}) = \mathbf{X}_{\text{new}} \hat{\mathbf{W}}$$

8. Linear Regression for Classification Data: $x_i, y_i{}_{i=1}^{m}$, with $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$. If $\mathcal{Y}$ is a finiteand discrete set, then the task corresponds to classification.

   · **Binary Classification** If $\mathbf{X^T X}$ is invertible, the closed-form solution is

   $$\text{Learning:} \quad \hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.\text{Prediction:} \quad \hat{y}(X_{\text{new}}) = \text{sgn}(\mathbf{X}_{\text{new}} \hat{\mathbf{w}}).$$

   · **Multi-class Classification**

   $$\text{Learning:} \quad \hat{\mathbf{W}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.\text{Prediction:} \quad \hat{\mathbf{Y}}(\mathbf{X}_{\text{new}}) = \arg \max_{i=1,\ldots,c} \mathbf{X}_{\text{new}} \hat{\mathbf{W}}.$$

   Each row in Y is a one-hot vector. For example, if there are 3 classes and the label is class 2, then the one-hot vector is [0,1,0].

9. Variants of Linear Regression 9.1. Ridge Regression (L2 Regularization) Motivation: We can not gaurantee that $\mathbf{X^T X}$ is invertible.

$$J(\mathbf{w}) = \sum_{i=1}^{m} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2$$

Closed-form solution:

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

$(X^T X + \lambda I)$ is always invertible when $\lambda > 0$.

9.2.Polynomial Regression Sometimes, the relationship between x and y is not linear.

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j x_j + \sum_{j=1}^{d} \sum_{k=j}^{d} w_{jk} x_j x_k + \cdots$$

See an example: Data $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$:

· $\{x_1 = 0, x_2 = 0\} \to \{y = -1\}$
· $\{x_1 = 1, x_2 = 1\} \to \{y = -1\}$
· $\{x_1 = 1, x_2 = 0\} \to \{y = +1\}$
· $\{x_1 = 0, x_2 = 1\} \to \{y = +1\}$

2^{nd} order polynomial model:

$$\mathbf{P} = \begin{bmatrix} 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$\hat{\mathbf{w}} = \mathbf{P}^\top(\mathbf{P}\mathbf{P}^\top)^{-1}\mathbf{y}$ (note: under determined linear system)

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 6 & 3 & 3 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

9.3. Lasso Regression (L1 Regularization)

$$J(\mathbf{w}) = \sum_{i=1}^{m}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda\|\mathbf{w}\|_1$$

9.4. Robust Regression

$$J(\mathbf{w}) = \sum_{i=1}^{m}|y_i - \mathbf{w}^\top \mathbf{x}_i|$$

Summary of Probabilistic Perspectives:

$p(y \mid \mathbf{x}, \mathbf{w})$	$p(\mathbf{w})$	regression method
Gaussian	Uniform	Least squares
Gaussian	Gaussian	Ridge regression
Gaussian	Laplace	Lasso regression
Laplace	Uniform	Robust regression
Student	Uniform	Robust regression

Lecture 6: Logistic Regression

1. Classification Sigmoid function:

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

$f_{\mathbf{w}}(\mathbf{x})$ is the probability of $y = 1$ given $\mathbf{x}$ and $\mathbf{w}$, which is defined as:

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

2. Logistic Regression Linear vs. Logistic Regression:

Aspect	Linear Regression	Logistic Regression
Loss Function	MSE (Mean Squared Error)	Cross-Entropy Loss
Formula	$\frac{1}{2m}\sum_{i=1}^{m}(f_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2$	$-y\log(f_{\mathbf{w}}(\mathbf{x})) - (1 - y)\log(1 - f_{\mathbf{w}}(\mathbf{x}))$
Output Range	$(-\infty, +\infty)$	$[0, 1]$ (probability)
Convexity	Convex	Non-convex with MSE; Convex with Cross-Entropy

Why Cross-Entropy for Logistic Regression?

1. **MSE Problems:**
   - Non-convexity
   - Gradient vanishing
2. **Cross-Entropy Advantages:**

$$\text{cost} = \begin{cases} -\log(f_{\mathbf{w}}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - f_{\mathbf{w}}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$

   - **Rewards correct predictions:** Loss approaches 0 when $f_{\mathbf{w}}(\mathbf{x})$ matches y.
   - **Penalizes errors:** Loss approaches ∞ when predictions are far from true labels, forcing rapid parameter correction.

3. Gradient Descent for Logistic Regression Learning $\mathbf{w}$ by minimizing the cost function:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} J(\mathbf{w}) = \arg\min_{\mathbf{w}} -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(f_{\mathbf{w}}(\mathbf{x}_i)) + (1 - y_i)\log(1 - f_{\mathbf{w}}(\mathbf{x}_i))]$$

Repeat the following steps until convergence:

$$\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} J(\mathbf{w})$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i$$

4. Multiclass Logistic Regression (Softmax Regression) Implementation Strategies

Strategy	One-vs-All (OvR)	Softmax Regression (Multinomial)
Core Idea	Train C binary classifiers (one per class).	Train one unified model for C classes.
Logic	Predict class with highest binary probability.	Output a normalized probability distribution.
Use Case	Overlapping categories (e.g., multi-tagging).	Mutually exclusive categories (single label).

Softmax Function For input $\mathbf{x}$, the probability of belonging to class j is:

$$P(y = j \mid \mathbf{x}; \mathbf{W}) = \frac{\exp(\mathbf{w}_j^T \mathbf{x})}{\sum_{c=1}^{C} \exp(\mathbf{w}_c^T \mathbf{x})}$$

Cost Function and Optimization Cross-Entropy Loss:

$$J(\mathbf{W}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{C} [\mathbb{I}(y_i = j) \log(f_{\mathbf{w}_j}(\mathbf{x}_i))]$$

Where $\mathbb{I}(\cdot)$ is the indicator function (effectively one-hot encoding).

Gradient Descent:

$$\nabla_{\mathbf{w}_j} J(\mathbf{W}) = \frac{1}{m} \sum_{i=1}^{m} [f_{\mathbf{w}_j}(\mathbf{x}_i) - \mathbb{I}(y_i = j)]\mathbf{x}_i$$

5. Regularization in Logistic Regression Add an L_2 norm penalty term to the original cost function $J(w)$:

$$\bar{J}(w) = J(w) + \frac{\lambda}{2m} \sum_{j=1}^{d} w_j^2$$

- λ **(Regularization parameter):** Controls the penalty strength. A larger λ makes the model simpler (prevents overfitting), but a value too large can cause underfitting.

- **No penalty for w_0:** The bias term w_0 is usually not regularized because it only translates the decision boundary and does not affect the model' s complexity.

Gradient Descent Update Rule: The parameter updates with regularization are slightly modified:

For w_0 (unchanged):

$$w_0 \leftarrow w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_w(x_i) - y_i) x_i^{(0)}$$

For w_j ($j > 0$):

$$w_j \leftarrow w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (f_w(x_i) - y_i) x_i^{(j)} + \frac{\lambda}{m} w_j \right]$$

Lecture 7: Support Vector Machine (SVM)

1. Two Types of Derivation Perspective 1: Geometric Intuition and Large Margin - The Distance from a Point to a Hyperplane: For a hyperplane defined by $\mathbf{w}^T\mathbf{x} + b = 0$, the distance from a point $\mathbf{x}$ to the hyperplane is given by:

$$\text{Distance} = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

- **Property of the Margin:** If we scale $\mathbf{w}$ and b by a positive constant c, the location of the hyperplane does not change:

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

- **Normalization:** We can normalize $\mathbf{w}$ and b such that the closest points (support vectors) satisfy:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$$

- **Maximizing the Margin:** Margin over all training data points is

$$\gamma = \min_i \frac{|f_{\mathbf{w},b}(\mathbf{x}_i)|}{\|\mathbf{w}\|}$$

Recall $y_i \in \{-1, +1\}$, we have:

$$\gamma = \min_i \frac{y_i f_{\mathbf{w},b}(\mathbf{x}_i)}{\|\mathbf{w}\|}$$

Maximize margin:

$$\max_{\mathbf{w},b} \gamma = \max_{\mathbf{w},b} \min_i \frac{y_i f_{\mathbf{w},b}(\mathbf{x}_i)}{\|\mathbf{w}\|} = \max_{\mathbf{w},b} \min_i \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Recall the fixed scaleing $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$, we have:

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

Subject to the constraints:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, ..., m$$

Perspective 2: Hinge Loss When $y_i = +1$, if the prediction value $f_{\mathbf{w},b}(\mathbf{x}_i) \geq 1$, the model is correct ans $Loss = 0$; if $f_{\mathbf{w},b}(\mathbf{x}_i) < 1$, there will be a linear penalty.

Hinge Loss:

$$\max(0, 1 - y_i f_{\mathbf{w},b}(\mathbf{x}_i))$$

2. Optimizing SVM by Lagrangian Duality

The objective function of SVM is:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i \end{aligned}$$

It can be transformed to:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0, \forall i \end{aligned}$$

The Lagrangian is:

$$L(\mathbf{w}, b, \mathbf{u}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m} u_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

The primal and dual optimal solutions should satisfy the KKT conditions: - **Stationarity:**

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_i^m \alpha_i y_i \mathbf{x}_i \frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i^m \alpha_i y_i = 0$$

- **Feasibility:**

$$\alpha_i \geq 0,\ 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0,\ \forall i$$

- **Complementary slackness:**

$$\alpha_i(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) = 0,\ \forall i$$

Replacing the stationary condition into Lagrange function, we have:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \alpha) &= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i^m \alpha_i - \sum_i^m \alpha_i y_i \left(\sum_j^m \alpha_j y_j \mathbf{x}_j\right)^\top \mathbf{x}_i - \sum_i^m \alpha_i y_i b \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i^m \alpha_i - \sum_{i,j}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - b\sum_i^m \alpha_i y_i \\ &= \sum_i^m \alpha_i - \frac{1}{2}\|\mathbf{w}\|^2 \\ &= \sum_i^m \alpha_i - \frac{1}{2}\sum_{i,j}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \end{aligned}$$

Thus, the dual problem is:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad & \alpha_i \geq 0, \forall i \\ & \sum_{i}^{m} \alpha_i y_i = 0 \end{aligned}$$

It can be solved by any off-the-shelf optimization solver.

We use the solved α back into the stationary condition:

$$\mathbf{w} = \sum_{i}^{m} \alpha_i y_i \mathbf{x}_i$$

For any support vector $\mathbf{x}_j$ (i.e., $\alpha_j > 0, j \in \mathcal{S}$), we have:

$$y_j(\mathbf{w}^T \mathbf{x}_j + b) = 1 \Rightarrow y_j(\sum_{i}^{m} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j + b) = 1$$

Product y_j for both sides of the above equation, and utilizing $y_j \cdot y_j = 1$, we have:

$$\sum_{i}^{m} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j + b = y_j \Rightarrow b = \frac{1}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} (y_j - \sum_{i}^{m} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j)$$

Prediction:

$$\mathbf{w}^T \mathbf{x} + b = \sum_{i}^{m} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + \frac{1}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} (y_j - \sum_{i}^{m} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j)$$

If $\mathbf{w}^T \mathbf{x} + b > 0$, then the predicted class of $\mathbf{x}$ is $+1$, otherwise -1.

The prediction is correct iff $y(\mathbf{w}^T \mathbf{x} + b) > 0$.

Lecture 8: Tree-based Methods

前面学过的模型大多是参数模型：训练时先选定一个整个输入空间共享的模型形式，再用全部训练数据去学一组固定参数；测试时所有样本都用同一套模型、同一套参数。

它们的问题是：

- 模型形式由人预先假设，可能和真实关系差很远；
- 决策过程通常不够直观，不容易解释。

引入**非参数模型** Typical nonparametric models include k-nearest neighbors (KNN) and decision tree (DT).

1. Classification Tree 对同一训练集，可能有很多棵零训练误差的树。

One input variable each step: how to choose? * Random * Least-values * Most-values * Impurity Measure (选择使不纯度下降最多，即信息增益最大的属性)

Impurity 不纯度：可用 Entropy 衡量。

$$\text{Info}(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

Information Gain 信息增益：选择使不纯度下降最多的属性。

$$\text{Gain}(S, A) = \text{Info}(S) - \text{Info}(S|A)$$

$$\text{Info}(S|A) = \sum_{v=1}^{V} \frac{|D_v|}{|S|} \text{Info}(D_v)$$

也可以用 Gini Index 衡量 impurity。

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

$$\text{Gini}(S|A) = \sum_{v=1}^{V} \frac{|D_v|}{|S|} \text{Gini}(D_v)$$

$$\Delta\text{Gini}(S) = \text{Gini}(S) - \text{Gini}(S|A)$$

2. Regression Tree 回归树和分类树构造方式很像，只是分类树关心类别纯度；回归树关心数值预测误差。

叶节点预测值：$\bar{y}_c = \frac{\sum_{i\in c} y_i}{N_c}$ 回归树通过 MSE/SSE 来评价效果：某个节点的 MSE：$e_c = \frac{1}{N_c}\sum_{i\in c}(y_i - \bar{y}_c)^2$ 整棵树一般看总 SSE：$S = \sum_{c\in\text{leaves(Tree)}}\sum_{i\in c}(y_i - \bar{y}_c)^2$

算法：1. 从包含全部样本的单个节点开始；2. 对每个节点，尝试所有变量、所有二分阈值；3. 选择使 SSE 降低最多的划分；4. 若降低不够大、或子节点样本过少、或特征已无差异，则停止；5. 对新子节点继续递归。

也就是说，对于某个值 w_1 把根节点分成左右节点 c_L, c_R，则

$$\bar{y}_L = \frac{\sum_{i\in c_L} y_i}{N_L}, \quad \bar{y}_R = \frac{\sum_{i\in c_R} y_i}{N_R} S_{w_1} = \sum_{i\in c_L}(y_i - \bar{y}_L)^2 + \sum_{i\in c_R}(y_i - \bar{y}_R)^2 w_1^* = \arg\max_{w_1}(S - S_{w_1})$$

3. Pruning 过程：1. 在训练集上先长一棵比较深的树；2. 然后反复评估 “把某个子树剪掉” 是否提升验证集表现；3. 贪心地剪掉最有利于验证集准确率的节点；4. 直到继续剪枝会带来伤害。

4. Ensemble Methods Bagging

第一步：Bootstrap 重采样 * 从训练集有放回抽样，得到多个不同的训练子集。

第二步：每个子集训练一棵过度生长的树

第三步：聚合预测 * 分类：多数表决 * 回归：平均值

Random Forest 在 Bagging 的基础上，每次分裂节点时不在全部 N 个特征里找最优划分，而是随机选择 m 个特征进行划分，增加树之间的差异性。

Chapter 3: Deep Learning

Contents: Neural Networks(MLP,CNN), RNN, Transformer etc.

Included lectures

- Lecture 9: Neural Networks
- Lecture 10: Convolutional Neural Networks

Lecture 9: Neural Networks

1. Introduction 神经单元: $y = \sigma(\mathbf{w}^T\mathbf{x} + b)$

Some Activation Functions: * ReLU: $\sigma(z) = \max(0, z)$ * Soft ReLU: $\sigma(z) = \log(1 + e^z)$ * Sigmoid/Logistic: $\sigma(z) = \frac{1}{1+e^{-z}}$ * Tanh: $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

2. Perceptron 能表示 AND、OR、NOT 等线性可分的函数，但不能表示 XOR 等线性不可分的函数。Formula: $y = \text{sgn}(\mathbf{w}^T\mathbf{x} + b)$ Objective function: $J(\mathbf{w}, b) = -\sum_{i=1}^{m} y_i(\mathbf{w}^T\mathbf{x}_i + b)$ Gradient Descent: $\mathbf{w} \leftarrow \mathbf{w} + \eta(y - t)\mathbf{x}$

3. Multilayer Feedforward Neural Networks 一个多层前馈神经网络包含: - 输入层 input layer - 一个或多个隐藏层 hidden layer(s) - 输出层 output layer

它的结构特点：- 信息只沿一个方向传播：从输入到输出；- 相邻层之间全连接；- 同层神经元之间没有连接；- 非相邻层之间没有连接，也没有 skip connection。

```
x -> h -> y
```

多层模型:

$$h = g_2(\mathbf{W}^T\mathbf{x} + \mathbf{c})y = g_1(w^T h + b)$$

4. Backpropagation 在计算图上应用链式法则

Lecture 10: Convolutional Neural Networks

1. Covolution Layer Convolution operation: $y = w * x + b$ 每一个 filter（卷积核）会产生一张 activation map（特征图），每个位置的值是卷积核和输入图像对应区域的加权和。

若输入是：32×32×3，用 6 个 5×5×3 的 filters，stride=1，且无 padding，那么每个 filter 生成一张 28×28 的图。最后堆叠得到：28×28×6

输出深度 = filter 个数。

卷积层的空间尺寸计算公式：

$$\frac{N - F + 2P}{S} + 1$$

其中，N 是输入的空间尺寸，F 是 filter 的空间尺寸，P 是 padding 的像素数，S 是 stride。

卷积层参数量设输入是：$W_1 \times H_1 \times C$ 卷积层有 4 个超参数：* K：滤波器个数 * F：滤波器尺寸 * S：stride * P：padding

则输出为：$W_2 \times H_2 \times K$ 其中：

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

参数量为：$F^2 \times C \times K + K$ 即：* $F^2 \times C \times K$ 个权重 * K 个偏置

2. Pooling Layer MAX pooling：取局部区域的最大值，保留纹理特征。

3. Fully Connected Layer 全连接层：每个神经元与上一层的所有神经元相连，输出一个标量。

Chapter 4: Unsupervised Learning

Contents: K-means, GMM, EM, PCA etc.

Included lectures

- Lecture 11: RNNs
- Lecture 12/13: Over/Under-fitting, Bias - Variance Tradeoff and Performance Evaluation
- Lecture 14: Intro to Unsupervised Learning
- Lecture 15: K-means Clustering
- Lecture 16: Gaussian Mixture Models and Expectation-Maximization Algorithm

Lecture 11: RNNs

1. Basic RNN RNN is a type of neural network designed for sequential data.
序列数据的核心特征是：顺序重要、长度可变。

” 当前时刻的表示，不仅取决于当前输入，还取决于过去的状态。

$$h_t = f(h_{t-1}, x_t)$$

这里，h_t 是当前时刻的隐藏状态，h_{t-1} 是前一时刻的隐藏状态，x_t 是当前时刻的输入，f 是一个非线性函数（如 tanh 或 ReLU）。

e.g.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) y_t = W_{hy}h_t$$

RNN 参数共享：- W_{hh}：隐藏状态之间的权重矩阵 - W_{xh}：输入到隐藏状态的权重矩阵 - W_{hy}：隐藏状态到输出的权重矩阵。

Cost function:

$$E(\theta) = \frac{1}{T}\sum_{t=1}^{T} L(y_t, \hat{y}_t)$$

where $\theta = W, W'$.

Training

RNN 的优化仍然是 “反向传播 + 梯度下降”，但由于网络跨越时间展开，因此要使用 Backpropagation Through Time，BPTT。

实际上一般使用 Truncated BPTT （截断 BPTT）来减少计算量，即只反向传播固定数量的时间步。

问题：RNN 的梯度在时间上反向传播时，会不断乘上 W_{hh}。- 如果 W_{hh} 的特征值小于 1，梯度会逐渐变小，导致梯度消失；- 如果特征值大于 1，梯度会逐渐变大，导致梯度爆炸。

-> RNN 难以捕捉 long-term dependencies.

2. LSTM (optional) LSTM（Long Short-Term Memory）解决传统 RNN 的梯度消失问题。

加入了三个门控机制：- **遗忘门**：决定保留多少过去的信息。

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- **输入门**：决定当前输入的重要性。

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

- **输出门**：决定输出多少当前状态的信息。

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

LSTM 还引入了一个新的状态变量 C_t，称为细胞状态，用于存储长期信息。

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$

3. Limitations of RNN

- **计算效率**：RNN 需要逐步处理序列数据，无法并行化，导致训练时间较长。
- **长距离依赖**：虽然 LSTM 和 GRU 改善了这一问题，但在处理非常长的序列时仍然存在困难。

4. Transformer Transformer 的核心是 self-attention 机制，它允许模型在处理序列时关注不同位置的信息。- **Query, Key, Value**：每个输入位置都会生成三个向量：

$$Q = W_Q x, K = W_K x, V = W_V x$$

- **Attention Score**：计算 Query 和 Key 之间的相似度：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Lecture 12/13: Over/Under-fitting, Bias - Variance Tradeoff and Performance Evaluation

Bias-Variance Tradeoff 假设数据生成过程为

$$y = f(x) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2)$$

机器学习的目标是基于训练集 D，通过某些学习算法 $\mathcal{A}$（如 SVM，Logistic Regression），学习一个 hypothesis function h。即 $h_D = \mathcal{A}(D)$。

Expected Hypothesis Function (given $\mathcal{A}$):

$$\bar{h} = \mathbb{E}_D[h_D]\mathbb{E}_{y|x}[y] = f(x)$$

即在所有可能的训练集 D 上，学习算法 $\mathcal{A}$ 学习到的假设函数的平均。

Expected test error:

$$\mathbb{E}_{(x,y),D}[(h_D(x) - y)^2] = \mathbb{E}_D[(h_D(x) - \bar{h}(x))^2] + \mathbb{E}[(\bar{h}(x) - f(x))^2] + \mathbb{E}[(f(x) - y)^2]$$

即 = Variance + Bias^2 + Noise

- 模型复杂度越高，Variance 越大，Bias 越小。
- 模型复杂度越低，Variance 越小，Bias 越大。

Cross Validation How to tune the hyper-parameters of a model?

K-fold Cross Validation:

```
| fold 1 | fold 2 | fold 3 | fold 4 | fold 5 | test |
```

1. 将训练数据划分为 K 个子集。
2. 每次用其中 1 份做 validation，其余 K-1 份做 train.
3. 重复 K 次，计算平均 validation performance.
4. 选择平均 performance 最好的 hyper-parameters.

为什么比分成 train/validation/test 更好? 因为它降低了 “随机切一刀” 的偶然性。

如果只做一次 train/val split，某些样本恰好被分进 validation，可能会让结果偏高或偏低。K-fold 的思想是:

让每个样本都轮流做一次验证集样本。

这样对数据的利用更充分，也更稳定。

但成本也高。

Evaluation Metrics for Regression

- **Mean Absolute Error (MAE)**: 平均绝对误差

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

- **Mean Squared Error (MSE)**: 平均平方误差

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

Evaluation Metrics for Classification Confusion Matrix | | Predicted Positive | Predicted Negative | |———-—|———————-|———————-| | Actual Positive | True Positive (TP) | False Negative (FN) | | Actual Negative | False Positive (FP) | True Negative (TN) |

- **Accuracy**: 准确率 (所有样本中，有多少被正确分类。)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**: 精确率 (所有被模型预测成正类的样本中，真正是正类的比例。)

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: 召回率 (所有真实正类中，被模型成功找出来的比例。)

$$Recall = \frac{TP}{TP + FN}$$

- **Cost-sensitive metrics**: 不同类型的错误有不同的成本。对于两种错误：FP 和 FN，分配不同的权重。

$$Cost = \frac{C_{p,p} \cdot TP + C_{n,n} \cdot TN}{C_{p,p} \cdot TP + C_{n,n} \cdot TN + C_{p,n} \cdot FP + C_{n,p} \cdot FN}$$

可以先对 Confusion Matrix 作 Normalization：| | Predicted Positive | Predicted Negative | |———-—|———————-|———————-| | Actual Positive | TPR | FNR | | Actual Negative | FPR | TNR |

- (True Positive Rate) TPR = TP/(TP + FN) = Recall
- (False Negative Rate) FNR = FN/(TP + FN)
- (False Positive Rate) FPR = FP/(FP + TN)
- (True Negative Rate) TNR = TN/(FP + TN)

每一个阈值设置，都对应一组不同的 FPR 和 FNR，这样的点称为一个 operating point

随着阈值从 0 变到 1，FPR 呈下降趋势，FNR 呈上升趋势。他们的交点称为 Equal Error Rate (EER)。

Graph DET curve: Detection Error Tradeoff curve, 以 FPR（x 轴）和 FNR（y 轴）为坐标的曲线。- DET 越低，模型性能越好。

ROC curve: Receiver Operating Characteristic curve, 以 FPR （x 轴）和 TPR （y 轴）为坐标的曲线。- ROC 越接近左上角，模型性能越好。

AUC: Area Under the Curve, ROC 曲线下的面积，越大越好。

AUC 可以看作模型给一个随机正样本打分高于一个随机负样本的概率。（衡量排序能力，AUC = 1 -> 完美模型）

- Scale-invariant：输出分数范围改变，不影响 AUC
- Threshold-invariant：阈值如何选，不影响 AUC。

$$AUC = \frac{1}{m^+m^-}\sum_{i=1}^{m^+}\sum_{j=1}^{m^-}u(e_{ij})u(e_{ij}) = \begin{cases}1, & \text{if } s_i^+ > s_j^- \\ 0.5, & \text{if } s_i^+ = s_j^- \\ 0, & \text{if } s_i^+ < s_j^-\end{cases}$$

Lecture 14: Intro to Unsupervised Learning

1. Definition

Unsupervised Learning	Supervised Learning
无标签数据 $D = (x_i)_{i=1}^N$ 从数据中发现结构、模式、分布等	有标签数据 $D = (x_i, y_i)_{i=1}^N$ 学习输入与输出之间的映射关系

2. Main Approaches

- **Clustering:** 把数据点按相似性分成若干组, 使组内点相互接近、组间点相互远离。(e.g. K-means, DBSCAN)
- **Dimensionality Reduction:** 将高维数据映射到低维空间，同时尽量保留原始数据的结构和信息。(e.g. PCA, t-SNE)
- **Density Estimation:** 估计数据的概率分布，常用于异常检测和生成模型。(e.g. Gaussian Mixture Models)
- **Autoencoders:** 通过神经网络学习数据的压缩表示，常用于降维和生成任务。
- **Self-supervised Learning:** 利用数据本身的结构进行无监督学习，常用于预训练和特征提取。

Lecture 15: K-means Clustering

K-means clustering minimizes withincluster variances (squared Euclidean distances).

本质：把 n 个样本分成 k 组，使得每个样本属于离它最近的那个簇中心所在的组。

算法步骤

1. 选定 K，随机初始化 K 个质心（centroids）
2. 赋值步骤（Assignment）：把每个样本分配给离它最近的质心
3. 更新步骤（Refitting）：重新计算每个簇的质心（簇内所有点的均值）
4. 重复 2-3 直到分配不再改变

数学本质 K-means minimize the within-cluster variance：

$$\min_{c,r} \mathcal{J}(c,r) = \min_{c,r} \sum_{i=1}^{n} \sum_{k=1}^{K} r_{i,k} \|x_i - c_k\|^2$$

$$\text{s.t. } r_i \in \{0,1\}^{n \times K}, \sum_{k=1}^{K} r_{i,k} = 1$$

where $r_{i,k} = 1$ if x_i is assigned to cluster k.

这个问题同时优化两类变量（分配 r 和质心 c），联合优化很难，于是用 **坐标下降（Coordinate Descent）** 轮流固定一个优化另一个：

当固定 c 时，最优的 r 是将每个样本分配给最近的质心。即 $k^* = \arg\min_k \|x_i - c_k\|^2$。

当固定 r 时，最优的 $c_k = \frac{\sum_{i=1}^{n} r_{i,k} x_i}{\sum_{i=1}^{n} r_{i,k}}$（簇内样本均值）。

这正是 K-means 算法的赋值步骤和更新步骤。

Performance Evaluation of Clustering

1. Internal Evaluation Metrics.

   只使用数据本身的信息来评估聚类质量。(Silhouette Coefficient).

   对每个样本计算：

   - a: 样本与同簇内其他样本的平均距离
   - b: 样本与最近的其他簇内样本的平均距离
   - Silhouette Coefficient: $s = \frac{b-a}{\max(a,b)}$，范围 [-1, 1]，越接近 1 越好。

2. External Evaluation Metrics.

   需要已知真实标签。

RI 统计所有点对中” 分类决策一致” 的比例（同簇同类 + 不同簇不同类），ARI 是对 RI 的随机校正版本，解决了 RI 对随机聚类也可能较高的问题。

- a：在 X 中同簇，在 Y 中也同簇
- b：在 X 中不同簇，在 Y 中也不同簇
- c：在 X 中同簇，在 Y 中不同簇
- d：在 X 中不同簇，在 Y 中同簇
- $RI = \frac{a+b}{a+b+c+d}$ （一致的样本对/总样本对），取值在 [0, 1]，越接近 1 越好。

ARI：对 RI 做 chance adjustment，解决了 RI 对随机聚类也可能较高的问题。

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

其中 $E[RI]$ 是随机聚类的期望 RI，$\max(RI)$ 是 RI 的最大值（当聚类完全正确时）。ARI 的取值范围是 [-1, 1]，越接近 1 越好。

Lecture 16: Gaussian Mixture Models and Expectation-Maximization Algorithm

A. Gaussian Mixture Models (GMM) 对于普通分类问题：$ p(x,z) = p(z) p(x|z) $ 其中 z 是类别标签，$p(z)$ 是类别的先验分布。

但对于聚类问题：z 未知。对 z 求和：

$$p(x) = \sum_z p(x, z) = \sum_z p(z)p(x|z)$$

这就是 Mixture Model。

1. Gaussian Mixture Model 假设每个簇都是 Gaussian 分布：

   $$p(x|z = k) = \mathcal{N}(x|\mu_k, \Sigma_k)$$

   整体：

   $$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

   其中 π_k 是第 k 个簇的权重，满足 $\sum_{k=1}^{K} \pi_k = 1$。

   $$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

2. GMM 为什么比 K-means 更强大?

- K-means 只能捕捉球状簇，而 GMM 可以捕捉任意形状的簇。
- GMM 提供了每个样本属于每个簇的概率，而 K-means 只能给出硬分配。

3. **参数估计** 目标：求 $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ 使得数据的似然函数最大化：

   $$\mathcal{L} = \log L(\Theta) = \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x^{(n)}|\mu_k, \Sigma_k)\right)$$

   where $\Theta = \{\pi, \mu, \Sigma\}$.

$$\frac{\partial \mathcal{L}}{\partial \mu_k} = 0 \implies \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)} \mid \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}^{(n)} \mid \mu_j, \Sigma_j)}}_{\gamma_k^{(n)}} \Sigma_k^{-1}(\mathbf{x}^{(n)} - \mu_k) = 0$$

引入 responsibility 责任度 $\gamma_k^{(n)}$:

$$\gamma_k^{(n)} = \frac{\pi_k \mathcal{N}(x^{(n)}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x^{(n)}|\mu_j, \Sigma_j)}$$

$\gamma_k^{(n)}$ 表示样本 $x^{(n)}$ 属于簇 k 的后验概率。

4. **参数更新共识** 有了责任度后，可以得出以下更新公式：

- 有效样本数：$N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$
- 均值更新：$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} x^{(n)}$
- 协方差更新：$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (x^{(n)} - \mu_k)(x^{(n)} - \mu_k)^T$
- 混合系数更新：$\pi_k = \frac{N_k}{N}$

5. **交替更新算法** 第一步：计算责任度

$$\gamma_k^{(n)} = \frac{\pi_k \mathcal{N}(x^{(n)} | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x^{(n)} | \mu_j, \Sigma_j)}$$

第二步：更新参数

$$N_k = \sum_{n=1}^{N} \gamma_k^{(n)} \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} x^{(n)} \Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (x^{(n)} - \mu_k)(x^{(n)} - \mu_k)^T \pi_k = \frac{N_k}{N}$$

不断重复直到对数似然收敛。

6. Comparison with K-means

- K- Means
  - Assignment step：把每个样本分给最近的簇中心（硬分配）
  - Refitting step：更新簇中心为簇内样本的均值
- GMM
  - E-step：计算每个样本属于每个簇的后验概率（软分配）
  - M-step：根据责任度更新簇的参数（均值、协方差、混合系数）

B. Expectation-Maximization (EM) Algorithm

1. GMM 的隐变量视角 We introduce a latent variable z for each sample, where z_i indicates which component generated x_i.

$$z \sim \text{Categorical}(\pi) p(z = k) = \pi_k p(x | z = k) = \mathcal{N}(x | \mu_k, \Sigma_k)$$

于是：

$$p(x) = \sum_{k=1}^{K} p(z = k) p(x | z = k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)$$

这样就把 GMM 解释成一个含离散隐变量的生成模型。

$z^{(n)}$ 不知道，难以用 MLE 直接求解参数。

2. Jensen’s Inequality Suppose f is a convex function, and X is a random variable, then:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$

If f is concave, the inequality is reversed:

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$$

- E-step：先猜每个样本来自各个簇的概率
- M-step：假设这些概率是真的，再重新估计参数