

2. DT KNN

Decision Tree

Impurity Measures:

- **Misclassification rate:** $i_E(t) = 1 - \max_c \pi_c$
- **Entropy (Shannon):** $i_H(t) = -\sum c_i \pi_{c_i} \log_2 \pi_{c_i}$
- **Gini index:** $i_G(t) = 1 - \sum c_i^2 \pi_{c_i}^2$

Greedy Optimization: Use $i_H(t)$ or $i_G(t)$ (not $i_E(t)$, which doesn't decrease impurity)

Information Gain:

$\cdot p_L = \frac{N_{left}}{N}$ $p_R = \frac{N_{right}}{N}$

$\cdot \Delta i = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R)$

Stopping Conditions: $i(t) = 0$ / d_{max} / $N_{node} < t_n$ / $\Delta i(s, t) < t_s$

LOOCV: Equivalent to N-fold cross-validation

KNN

Prediction: $\hat{y} = \arg \max_c \sum_{x_i \in N_k(x)} \mathbb{I}(y_i = c)$

Distance Metrics:

- **L1 (Manhattan):** $d(x_1, x_2) = \sum_d |x_{1d} - x_{2d}|$
- **L2 (Euclidean):** $d(x_1, x_2) = \sqrt{\sum_d (x_{1d} - x_{2d})^2}$
- **L ∞ :** $d(x_1, x_2) = \max_d |x_{1d} - x_{2d}|$

• **Cosine Similarity:** $\text{Sim}(x_1, x_2) = \frac{x_1^T x_2}{\|x_1\| \|x_2\|}$

• **Mahalanobis Distance:** $\sqrt{(x_1 - x_2)^T \Sigma^{-1} (x_1 - x_2)}$ (Σ : positive semi-definite, symmetric)

Weighted KNN (inverse distance weighting, closer points more important):

$\cdot \hat{y} = \arg \max_c \frac{1}{\sum_{x_i \in N_k(x)} d(x_i, x)} \sum_{x_i \in N_k(x)} \mathbb{I}(y_i = c)$

Hyperparameter Selection:

- k small \rightarrow overfitting
- k large \rightarrow underfitting
- Use odd number to avoid ties

Scale Issue: Normalization $x_i = \frac{x_i - \mu_i}{\sigma_i}$ (or use weighted distance)

Confusion Matrix

Ground \ Predict	**Y**	**X**
X	TP	FN
Y	FP	TN

Metrics:

- **Precision:** $\frac{TP}{P+FP}$
- **Sensitivity/Recall:** $\frac{TP}{P+FP+N}$
- **Accuracy:** $\frac{TP+TN}{P+FP+N+P+FN}$
- **F1 Score:** $\frac{2 \cdot \text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$

3. Prob Method

Probabilistic Inference

Maximum Likelihood Estimation (MLE):

$\cdot \theta_{MLE} = \arg \max_{\theta} p(D|\theta)$

$\cdot p(D|\theta) = \prod_{i=1}^N p(x_i|\theta)$

$\cdot E_{MLE} = -\ln p(D|\theta) = -\sum_{i=1}^N \ln p(x_i|\theta)$

$\cdot \theta_{MLE} = \frac{|T|}{|T|+|H|}$

Maximum A Posteriori (MAP):

$\cdot \theta_{MAP} = \arg \max_{\theta} p(\theta|D)$

$\cdot p(\theta|D) \propto p(D|\theta)p(\theta)$

$\cdot E_{MAP} = -(|T|+a-1) \ln \theta - (|H|+b-1) \ln (1-\theta)$

$\cdot \theta_{MAP} = \frac{|T|+a-1}{|T|+|H|+a+b-2}$

• When $a = b = 1$: $\theta_{MAP} = \theta_{MLE}$

Posterior: $P(\theta|D) = \text{Beta}(a\theta + |T|, b + |H|)$

Hoeffding's Inequality

$p(|\theta_{MLE} - \theta_{true}| \geq \epsilon) \leq 2e^{-2N\epsilon^2} \leq \delta$

Bayesian Models

Predictive Distribution: $p(f|D, a, b) = \int_0^1 p(f|\theta)p(\theta|D, a, b)d\theta$

Fully Bayesian: $\theta^* = \frac{|T|+a}{|T|+|H|+a+b} = \text{Ber}(f|\theta)$

Conjugate Priors

Bernoulli = Beta ($\Gamma(n) = (n-1)!$):

• Likelihood: $p(D|\theta) = \theta^k (1-\theta)^{n-k}$

• Prior: $\text{Beta}(\theta|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1}$

Poisson = Gamma:

• Likelihood: $p(D|\lambda) = \prod_{i=1}^N \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$

• Prior: $p(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)} \alpha^{-1} e^{-\beta\lambda}$

• Posterior: $p(\lambda|D) = \text{Gamma}(\lambda|\alpha + \sum x_i, \beta + N)$

Gaussian = Gaussian:

• Likelihood: $p(D|\mu) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$

• Prior: $p(\mu) = \mathcal{N}(\mu|\mu_0, \tau_0^2)$

Posterior parameters:

$\cdot \mu_{\text{post}} = (\mu_0\sigma^2 + N\bar{x}\tau_0^2)/(\sigma^2 + N\tau_0^2)$

$\cdot \tau_{\text{post}}^2 = 1/\left(\frac{1}{\tau_0^2} + \frac{N}{\sigma^2}\right)$

4. Linear Regression

Model: $y_i = f(x_i) + \epsilon_i$ $\epsilon_i \sim \mathcal{N}(0, \beta^{-1})$

Least Squares Error: $E_{LS} = \frac{\beta}{2} \sum_{i=1}^N (w^T x_i - y_i)^2$

Optimal Weight: $w^* = \arg \min_w E_{LS} = (X^T X)^{-1} X^T y = X^+ y$

Non-linear Data (Feature Transform):

$\cdot f(x) = w_0 + \sum_{j=1}^M w_j \phi_j(x) = w^T \Phi(x)$

$\cdot \Phi \in \mathbb{R}^{N \times (M+1)}$

$\cdot w^* = (\Phi^T \Phi)^{-1} \Phi^T y = \Phi^+ y$

Model Complexity:

- High variance \rightarrow overfit
- High bias \rightarrow underfit

Ridge Regression: $E_{ridge} = \frac{\beta}{2} \sum_{i=1}^N (w^T \Phi(x_i) - y_i)^2 + \frac{\lambda}{2} \|w\|_2^2$

Probabilistic Formulation

Likelihood: $y_i \sim \mathcal{N}(f_{\mu}(x_i), \beta^{-1})$, $p(y_i|X, w, \beta) = \prod_{i=1}^N p(y_i|f_{\mu}(x_i), \beta)$

Negative Log-Likelihood:

$\cdot E_{ML} = -\ln p(y|X, w, \beta)$

$\cdot E_{ML} = \frac{\beta}{2} \sum_{i=1}^N (w^T \Phi(x_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N \ln 2\pi}{2}$

Maximum Likelihood Estimators:

$\cdot w_{ML} = w_{LS} = \Phi^+ y$

$\cdot \frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{i=1}^N (w_{ML}^T \Phi(x_i) - y_i)^2$

Page 1: Supervised & Deep Learning (Sections 2-8)

With Gaussian Prior: $p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2} w^T w\right)$ (M : length of w)

MAP Estimation:

• $E_{MAP} = -\ln p(y|X, w, \beta) - \ln p(w|\alpha)$

• $E_{MAP} = \frac{\beta}{2} \sum_{i=1}^N (w^T \phi(x_i) - y_i)^2 + \frac{\alpha}{2} \|w\|_2^2$

• Equivalent to Ridge Regression where $\lambda = \frac{\alpha}{\beta}$

• $w_{\text{ridge}}^* = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$

Fully Bayesian Linear Regression

Posterior: $p(w|D) = \mathcal{N}(w|\mu, \Sigma)$

$\cdot \mu = \beta \Sigma \Phi^T y$

$\cdot \Sigma^{-1} = \alpha I + \beta \Phi^T \Phi$

Predictions:

• MLE: $p(y_{\text{new}}|x_{\text{new}}, w_{ML}, \beta_{ML}) = \mathcal{N}(y_{\text{new}}|w_{ML}^T \Phi(x_{\text{new}}), \beta_{ML}^{-1})$

• MAP: $p(y_{\text{new}}|x_{\text{new}}, w_{MAP}, \beta) = \mathcal{N}(y_{\text{new}}|w_{MAP}^T \Phi(x_{\text{new}}), \beta^{-1})$

• Fully Bayesian: $p(y_{\text{new}}|x_{\text{new}}, D) = \mathcal{N}(y_{\text{new}}|\mu^T \Phi(x_{\text{new}}), \beta^{-1} + \phi(x_{\text{new}})^T \Sigma \phi(x_{\text{new}}))$

Weighted Linear Regression

Objective (with weight τ_i): $E_{\text{weighted}} = \frac{1}{2} \sum_{i=1}^N \tau_i (w^T \phi(x_i) - y_i)^2$

Optimal Weight: $w_{\text{weighted}} = (\Phi^T R \Phi)^{-1} \Phi^T R y$

5. Linear Classification

Zero-one Loss: $l_{01}(y, \hat{y}) = \sum_i \mathbb{I}(\hat{y}_i \neq y_i)$ (loss for incorrect predictions is 1)

Hyperplane: $f(x) = w^T x + w_0$

• Direction: w

• Distance from origin: $\frac{w_0}{\|w\|}$

Perceptron Update Rule (for each misclassified x_i):

$w \leftarrow \begin{cases} w + x_i & \text{if } y_i = 1 \\ w - x_i & \text{if } y_i = 0 \end{cases}$ $w_0 \leftarrow \begin{cases} w_0 + 1 & \text{if } y_i = 1 \\ w_0 - 1 & \text{if } y_i = 0 \end{cases}$

Probabilistic Generative Model:

• Prior: $y \sim \text{Categorical}(\theta)$, $p(y = c) = \theta_c = \frac{N_c}{N}$, $\sum_c \theta_c = 1$

• Class-conditional: $p(x|y = c) = \mathcal{N}(x|\mu_c, \Sigma)$ (assume Σ_c all equal)

Probabilistic Generative Models & Discriminant Analysis

Binary Classification:

$\cdot p(y = 1|x) = \sigma(a) = \frac{1}{1+e^{-a}}$ where $a = \ln \frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}$

$\cdot a = w^T x + w_0$

LDA (Linear Discriminant Analysis) (with shared covariance Σ_c):

$\cdot w = \Sigma^{-1} (\mu_1 - \mu_0)$

$\cdot w_0 = -\frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + \ln \frac{p(y=1)}{p(y=0)}$

• Thus $y|x \sim \text{Bernoulli}(\sigma(w^T x + w_0))$

Multi-class Classification:

$\cdot p(y = c|x) = \frac{p(x|y=c)p(y=c)}{\sum_{j=1}^C p(x|y=c_j)p(y=c_j)} = \frac{\exp(w_c^T x + w_{c0})}{\sum_{j=1}^C \exp(w_j^T x + w_{j0})}$

$\cdot w_c = \Sigma^{-1} \mu_c$

$\cdot w_{c0} = -\frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + \ln p(y = c)$

QDA (Quadratic Discriminant Analysis) (different covariances Σ_{c_i}):

$\cdot p(y = 1|x) = \sigma(a)$ where $a = x^T W_2 x + w_1^T x + w_0$

$\cdot W_2 = \frac{1}{2} \Sigma_0^{-1} - \Sigma_1^{-1}$

$\cdot w_1 = \Sigma_1^{-1} \mu_1 - \Sigma_0^{-1} \mu_0$

$\cdot w_0 = -\frac{1}{2} \mu_1^T \Sigma_1^{-1} \mu_1 + \frac{1}{2} \mu_0^T \Sigma_0^{-1} \mu_0 + \ln \frac{\pi_1}{\pi_0} + \frac{1}{2} \ln \frac{|\Sigma_0|}{|\Sigma_1|}$

Linear Discriminant Model: Logistic Regression

Binary Logistic Regression:

$\cdot p(y = 1|x) = \sigma(w^T x)$

$\cdot p(y = 0|x) = 1 - \sigma(w^T x)$

$\cdot p(y|w, x) = \prod_{i=1}^N \sigma(w^T x_i)^{y_i} (1 - \sigma(w^T x_i))^{1-y_i}$

Loss Function (Binary Cross Entropy):

$\cdot E(w) = -\sum_{i=1}^N (y_i \log \sigma(w^T x_i) + (1 - y_i) \log (1 - \sigma(w^T x_i)))$

• Regularization can be added: $\frac{\lambda}{2} \|w\|_2^2$

Multi-class (Softmax + Cross Entropy):

$\cdot E(w) = -\sum_{i=1}^N \sum_{c \neq i} y_{ic} \log \frac{\exp(w_c^T x)}{\sum_{c' \neq i} \exp(w_{c'}^T x)}$

$\cdot y_{ic} = 1$ iff sample $x \in c$ class

6. Optimization

Convexity

A function is convex if:

1. $f((1-t)x + ty) \leq (1-t)f(x) + tf(y)$ (any point between two points is lower than the line connecting them)

2. $f(y) - f(x) \geq \frac{f((1-t)x + ty) - f(x)}{t}$

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

4. Hessian Matrix is positive semi-definite

Gradient Descent (Line Search)

1. $\Delta \theta = -\nabla f(\theta)$

2. $t^* = \arg \min_{t \geq 0} f(\theta + t \Delta \theta)$

3. $\theta = \theta + t^* \Delta \theta$

SGD (Stochastic Gradient Descent)

$\theta = \theta - r \cdot \nabla f(\theta)$ where r is learning rate

Decaying learning rate: $r = \alpha r$, $0 < \alpha < 1$

Momentum

$\cdot m_t = r \cdot \nabla f(\theta_t) + \gamma \cdot m_{t-1}$

$\cdot \theta_{t+1} = \theta_t - m_t$

Adam

$\cdot m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla f(\theta_t)$ (mean)

$\cdot v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla f(\theta_t))^2$ (variance)

$\cdot \hat{m}_t = \frac{m_t}{1 - \beta_1^t}$ $\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$

$\cdot \theta_{t+1} = \theta_t - \frac{r}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$

• Default values: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

Newton Method

Taylor expansion: $f(\theta_t + \delta) = f(\theta_t) + \delta^T \nabla f(\theta_t) + \frac{1}{2} \delta^T \nabla^2 f(\theta_t) \delta + \dots$

Update: $\theta_{t+1} = \theta_t - [\nabla^2 f(\theta_t)]^{-1} \nabla f(\theta_t)$

Mini-batch SGD

$\cdot \theta_{t+1} = \theta_t - r \cdot \frac{1}{|S|} \sum_{j \in S} \nabla L_j(\theta_t)$

• Batch size $\uparrow \rightarrow$ variance \uparrow

• Batch size $\uparrow \rightarrow$ computation time \uparrow

7. Deep Learning

Notation: w_{ijk} denotes weight from layer i , input node j , output node k

Architecture Types

• Feed-Forward Neural Network (FFNN)

• Multi-layered Perceptron (MLP)

Activation Functions

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

• ReLU: $\max(0, x)$

• ELU: $\begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

• tanh: $\tanh(x)$

• Leaky ReLU: $\max(0.1x, x)$

• Swish: $x \cdot \sigma(x)$

Target and Loss Functions

Target $f_P(y)$	x / \hat{y}	Final Layer	Loss
-----	-----	-----	-----
Binary (Bernoulli)	Sigmoid	BCE (Binary Cross Entropy)	
Discrete (Categorical)	Softmax	CE (Cross Entropy)	
Continuous (Gaussian)	Identity	Squared Error	

Weight Update Rule

$W^{(new)} = W^{(old)} - r \nabla_W E(W^{(old)})$

Backpropagation

Chain Rule: $\frac{\partial c}{\partial z} = \frac{\partial c}{\partial a} \frac{\partial a}{\partial z} + \frac{\partial c}{\partial b} \frac{\partial b}{\partial z}$

Gradient of vector: $\nabla_a c = \left(\frac{\partial c}{\partial a_1}, \frac{\partial c}{\partial a_2}, \dots, \frac{\partial c}{\partial a_m} \right)^T \in \mathbb{R}^{1 \times m}$

Vector chain rule: $\nabla_x c = \left(\frac{\partial a}{\partial x} \right)^T \nabla_a c$

Derivative Dimensions (Output vs. Input):

scalar	**scalar**	**vector**	**matrix**
scalar	scalar	vector	matrix
vector	vector	matrix	3-way tensor
matrix	matrix	3-way tensor	4-way tensor

Tensor notation: $\left(\frac{\partial a}{\partial W} \right)_{ijk} = \frac{\partial a_i}{\partial W_{jk}}$

8. CNN & Deep Learning Architecture

CNN Kernel Parameters: $L \times M \times C_{in} \times C_{out}$

Padding

• VALID: No padding, $D_{l+1} = (D_l - K) + 1$

• SAME: Add $P = \lfloor \frac{K-1}{2} \rfloor$ padding on each side to keep size

• FULL: Add $P = K - 1$ on each side to increase size

9. Support Vector Machines (SVM)

Margin: $\frac{1}{\|w\|}$
Constraints:
• $w^T x_i + b \geq 1$ for $y_i = +1$
• $w^T x_i + b \leq -1$ for $y_i = -1$
• Thus: $y_i(w^T x_i + b) - 1 \geq 0$ for $\forall x_i$
Optimization Problem:
• **Minimize:** $\frac{1}{2} w^T w$
• **Subject to:** $f_i(w, b) = y_i(w^T x_i + b) - 1 \geq 0$

Lagrangian Dual Function
• **Dual Function:** $g(\alpha) = \min_{a \in \mathbb{R}^d} (f_0(a) + \sum_{i=1}^M \alpha_i f_i(a))$
• **Lagrangian:** $L(\theta, \alpha) = f_0(\theta) + \sum_{i=1}^M \alpha_i f_i(\theta)$
• **Conditions:** $\alpha_i \geq 0$ and $f_i(\theta) \leq 0$

SVM Optimization Steps
1. Calculate Lagrangian:
• $L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^N \alpha_i [y_i(w^T x_i + b) - 1]$
2. Minimize L:
• $\frac{\partial L}{\partial w} = w - \sum_{i=1}^N \alpha_i y_i x_i \stackrel{!}{=} 0$
• $\frac{\partial L}{\partial b} = \sum_{i=1}^N \alpha_i y_i \stackrel{!}{=} 0$
• $\Rightarrow w = \sum_{i=1}^N \alpha_i y_i x_i$
3. Dual Problem:
• $g(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j x_i^T x_j$
• **Note:** $x_i^T x_j$ can be replaced by Kernel $\phi(x_i, x_j)$
• **w.r.t:** $\alpha_i \geq 0, \quad \sum_{i=1}^N \alpha_i y_i = 0$
• $w = \sum_{i=1}^N \alpha_i y_i x_i$
• $b = \frac{1}{\sum_{i=1}^N \alpha_i} w^T x_i = y_i - w^T x_i$
• $\therefore h(x) = \text{sign}(\sum_{i \in S} \alpha_i y_i x_i^T x + b)$

Soft SVM (relaxed margin):
• **Constraint:** $y_i(w^T x_i + b) \geq 1 - \xi_i \quad (\xi_i \geq 0)$
• **Minimize:** $f_0(w, b, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i$
• **w.r.t:**
• $y_i(w^T x_i + b) - 1 + \xi_i \geq 0$
• $\xi_i \geq 0$
• $(C \rightarrow \infty: \text{hard margin})$

Soft Margin SVM Derivation
Lagrangian Formulation:
 $L(w, b, \xi, \alpha, \mu) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^N \mu_i \xi_i$

2. Minimize L (Gradients):
• $\frac{\partial L}{\partial w} = w - \sum_{i=1}^N \alpha_i y_i x_i \stackrel{!}{=} 0$
• $\frac{\partial L}{\partial b} = \sum_{i=1}^N \alpha_i y_i \stackrel{!}{=} 0$
• $\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i \stackrel{!}{=} 0 \Rightarrow \alpha_i = C - \mu_i$
• **Since** $\mu_i \geq 0$ **and** $\alpha_i \geq 0$: **Box Constraint** $\alpha_i \in [0, C]$
3. Maximize Dual Function:
 $g(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j x_i^T x_j$

Subject to:
• $\sum_{i=1}^N \alpha_i y_i = 0$
• $0 \leq \alpha_i \leq C$
Interpretation of α_i :
• **If** $0 < \alpha_i < C$: $\xi_i = 0$ (point exactly on margin)
• **If** $\alpha_i = C$: $\xi_i > 0$ (point violates margin)
• **Larger** $C \rightarrow$ **less tolerance for points inside margin**

Hinge Loss:
 $\frac{1}{2} w^T w + C \sum_{i=1}^N \max\{0, 1 - y_i(w^T x_i + b)\}$

Kernel Methods
Definition: $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
Prediction: $h(x) = \text{sign}(\sum_{j \in S} \alpha_j y_j k(x_i, x) + b)$
Kernel Matrix Properties: Must be symmetric positive semi-definite
Valid Kernel Construction Rules:
1. **Sum:** $k(x_1, x_2) = k_1 + k_2$
2. **Scaling:** $k(x_1, x_2) = c k_1$ with $c > 0$
3. **Product:** $k(x_1, x_2) = k_1 \cdot k_2$
4. **Transformation:** $k(x_1, x_2) = k_3(\phi(x_1), \phi(x_2))$
5. **Matrix Scaling:** $k(x_1, x_2) = z^T A x_2$ where A is symmetric positive semi-definite
Common Examples:
• **Polynomial:** $k(a, b) = (a^T b)^n$ or $(a^T b + 1)^P$
• **Gaussian (RBF):** $k(a, b) = \exp\left(-\frac{\|a-b\|^2}{2\sigma^2}\right)$

Multiclass Classification Strategies
• **1 vs n classification:** Look at maximum distance
• **1 vs 1 classification:** Look at majority vote

10. Dimension Reduction PCA SVD

Dimension Reduction (PCA)
Transformation: $\tilde{x} = \tilde{x} \cdot F \quad \Sigma_{\tilde{x}} = F^T \Sigma_x F$

1. Centering Data
 $\tilde{x}_i = x_i - \bar{x}$ where $\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_d \end{bmatrix} = \frac{1}{N} X^T \cdot 1_N$
2. Variance and Covariance
• **Variance:** $\text{Var}(X_j) = \frac{1}{N} X_j^T X_j - \bar{x}_j^2$
• **Covariance:** $\text{Cov}(X_i, X_j) = \frac{1}{N} X_i^T X_j - \bar{x}_i \bar{x}_j$
• **Covariance Matrix:** $\Sigma_X = \frac{1}{N} \tilde{X}^T \tilde{X}$ (symmetric)
3. Eigen-decomposition
 $\Sigma_{\tilde{X}} = \Gamma \Lambda \Gamma^T$ (where Λ is diagonal)
4. Transformation
• $Y = \tilde{X} \cdot \Gamma$
• $Y_{\text{reduced}} = \tilde{X} \cdot \Gamma_{\text{truncated}}$
• **Criteria:** Retain 90% of variance: $\sum_{i=1}^k \lambda_i \geq 0.9 \sum_{i=1}^d \lambda_i$
• **Complexity:** $O(nd^2 + d^3)$
5. Iterative Eigenvector Calculation
Power Iteration: $v \leftarrow \frac{A \cdot v}{\|A \cdot v\|}$ (converges to eigenvector with largest eigenvalue)

Page 2: SVM, Unsupervised & Misc (Sections 9-12)

Singular Value Decomposition (SVD)

Goal: Find best low-rank approximation of matrix A
Frobenius Norm Objective: $\|A - B\|_F^2 = \sum_{i,j=1}^n \sum_{k=1}^p (a_{ij} - b_{ij})^2$
Complexity: $O(n \cdot d^2)$ or $O(n^2 \cdot d)$
Decomposition: $A = U \Sigma V^T$ where:
• $U \in \mathbb{R}^{n \times r}$ (user-to-concept similarity)
• $\Sigma \in \mathbb{R}^{r \times r}$
• $V \in \mathbb{R}^{d \times r}$ (item-to-concept similarity)
Rank-2 Decomposition: $A = (\sigma_1 \cdot u_1 \cdot v_1^T) + (\sigma_2 \cdot u_2 \cdot v_2^T)$
Using SVD for Dimensionality Reduction
Projection: $P = U \Sigma$ or $P = A \cdot V$
Retain 90% energy: $\sum_{k=1}^k \sigma_i^2 \geq 0.9 \sum_{i=1}^n \sigma_i^2$
Relationship to Eigenvectors:
• V contains eigenvectors of $X^T X$
• U contains eigenvectors of $X X^T$

Matrix Factorization (MF)

1. Fundamentals & Metrics
RMSE: $RMSE = \sqrt{\frac{1}{|S|} \sum (r_{ui} - \hat{r}_{ui})^2}$
SSE: $SSE = \sum (r_{ui} - [U \Sigma V^T]_{ui})^2$
Decomposition: $R = U \Sigma V^T \approx Q \cdot P^T$
Prediction: $\hat{r}_{ui} = q_u \cdot p_i^T$
2. Alternating Optimization
1. Initialize: $P_0, Q_0, t = 0$
2. Update P: $P_{t+1} = \text{argmin}_P f(P, Q_t)$
• **Closed form:** $p_i^T = \left(\frac{1}{|S_{q,i}|} \sum q_u^T q_u \right)^{-1} \cdot \frac{1}{|S_{q,i}|} \sum q_u^T r_{ui}$
3. Update Q: $Q_{t+1} = \text{argmin}_Q f(P_{t+1}, Q)$
4. Repeat until convergence
3. Stochastic Gradient Descent (SGD)
• $e_{ui} \leftarrow r_{ui} - q_u \cdot p_i^T$
• $q_u \leftarrow q_u + 2r(e_{ui} p_i)$
• $p_i \leftarrow p_i + 2r(e_{ui} q_u)$ (r : learning rate)
4. Extensions: Bias & Regularization
Regularized Objective:
 $\sum (r_{ui} - q_u \cdot p_i^T)^2 + \lambda_1 \sum \|q_u\|^2 + \lambda_2 \sum \|p_i\|^2$
Full Loss with Bias:
 $L = \sum (r_{ui} - (q_u \mu_i^T + b_u + b_i + b) \cdot p_i^T)^2 + \lambda_1 \sum \|q_u\|^2 + \lambda_2 \sum \|p_i\|^2$
SGD Updates (with Bias & Regularization):
1. $e_{ui} = r_{ui} - q_u \cdot p_i^T$
2. $q_u \leftarrow q_u + 2r(e_{ui} p_i - \lambda_1 q_u)$
3. $p_i \leftarrow p_i + 2r(e_{ui} q_u - \lambda_2 p_i)$
4. $b_u \leftarrow b_u + 2r e_{ui}$
5. $b_i \leftarrow b_i + 2r e_{ui}$
6. $b \leftarrow \frac{1}{|S|} \sum r_{ui}$ (Global Bias)

11. Dimension Reduction Neighbor Graph Method

Neighbor Graph Method
Preserve local structure
Plaintext

t-SNE

High-dimensional similarity:
 $p_{ij} = \frac{\exp(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2})}$
• **Data** X
• $p_{ii} = 0, p_{ij} = \frac{p_{ji} + p_{ij}}{2N}$
• $p_{ij} = p_{ji}$
Low-dimensional similarity:
 $q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k (1 + \|y_i - y_k\|^2)^{-1}}$
• **Target** y
• $q_{ii} = 0$
Next: minimize KL Divergence
 $\sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}} = KL(P||Q)$
• $KL(P||Q) \geq 0, = 0$ iff $P = Q$
Divergence Behavior:
Plaintext

Plaintext

Autoencoder

Plaintext

Notes:
• Extract key features and reconstruct; equivalent to only taking $\text{rank}(X) = L$
• **Formula:** $X W_1 W_2 = \hat{X}$
• $W_1^T W_2 = W^* = \Gamma$
• Γ is the largest L eigenvectors of $X^T X$.
12. Clustering
K-means: Objective (Distortion Measure)
• **Formula:**
 $J(X, Z, \mu) = \sum_{i=1}^n \sum_{k=1}^K z_{ik} \|x_i - \mu_k\|^2$
• **Indicator Variable:**
• $z_i = \{0, 0, 1, 0, 0, 0\}$ implies sample i belongs to class 3.
Lloyd's Algorithm (Alternating Optimization)
1. **Initialize:** All centroids μ_i .
2. **Assignment:** Assign cluster indicators based on the nearest neighbor.
• **(Note:** Whichever centroid a point is closest to, it belongs to that cluster.)

3. Update:
• $\mu_k = \frac{1}{N_k} \sum_{i=1}^N z_{ik} x_i$
• **Where** $N_k = \sum_{i=1}^N z_{ik}$
• **Loop:** Repeat until convergence.
K-means ++
1. Randomly select one data point as μ_1 .
2. Calculate the distance of the remaining points to it: $\|x_i - \mu_1\|^2$.
3. Sample the next center point, with probability proportional to the distance size.
4. Re-calculate $D_i^2 = \min(\|x_i - \mu_1\|^2, \|x_i - \mu_2\|^2, \dots)$.
5. Repeat steps 3 and 4 until k centers are chosen.

Gaussian Mixture Model (GMM)

Model Definition:
 $p(x|\theta) = \sum_{k=1}^K p(x|z, \theta) p(z|\theta)$
• z is latent variables
• **Optimization Goal:** $\theta^* = \text{argmax}_{\theta} p(x|\theta)$
Distributions:
1. **Prior:** $p(z|\theta) = \text{Cat}(\pi) \quad // \quad \{\pi_1, \pi_2, \dots, \pi_K\}$
• **Constraint:** $\sum_{k=1}^K \pi_k = 1$
2. **Likelihood:** $p(x|z = k, \theta) = \mathcal{N}(x|\mu_k, \Sigma_k)$
Parameters:
Thus $\theta = \{\pi, \mu, \Sigma\}$
• π : k parameters (effectively $k - 1$)
• $\mu \in \mathbb{R}^d$: $k \cdot d$ parameters
• $\Sigma_k \in \mathbb{R}^{d \times d}$: $k \cdot d^2$ parameters (or $k \cdot \frac{d(d+1)}{2}$ due to symmetry)

Using GMM as a Generative Model
1. **Sample class** according to $\pi = \{\pi_1, \pi_2, \dots, \pi_K\}$
2. **Generate** x according to $x \sim \mathcal{N}(\mu_k, \Sigma_k)$ (Probability Density)
Marginal Probability:
 $p(x|\pi, \mu, \Sigma) = \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k)$

Log-Likelihood:
 $\log p(X|\pi, \mu, \Sigma) = \sum_i \log \left(\sum_k \pi_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k) \right)$

Inference
Calculating the responsibility of component k for observation i :
 $p(z_{ik} = 1|x_i, \pi, \mu, \Sigma) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$

Notation:
 $\gamma(z_{ik}) = p(z_{ik} = 1|x_i, \pi, \mu, \Sigma)$
Expectation Maximization (EM) for GMM

Learning Process:
1. **Initialize:** $\pi^0, \mu_1^0 \dots \mu_K^0, \Sigma_1^0 \dots \Sigma_K^0$
2. **E-step (Expectation):**
Calculate the responsibility (posterior probability):
 $\gamma(z_{ik}) = \frac{\pi_k^t \mathcal{N}(x_i|\mu_k^t, \Sigma_k^t)}{\sum_{j=1}^K \pi_j^t \mathcal{N}(x_i|\mu_j^t, \Sigma_j^t)}$
• **Note:** E-step evaluates the posterior $p(z_i|z, \theta^t)$.
3. **M-step (Maximization):**
Update parameters to maximize expected joint probability:
• $\mu_k^{t+1} = \frac{1}{N_k} \sum_{i=1}^N \gamma(z_{ik}) x_i$
• $\Sigma_k^{t+1} = \frac{1}{N_k} \sum_{i=1}^N \gamma(z_{ik}) (x_i - \mu_k^{t+1})(x_i - \mu_k^{t+1})^T$
• $\pi_k^{t+1} = \frac{N_k}{N}$ where $N_k = \sum_{i=1}^N \gamma(z_{ik})$
4. Repeat steps 2 and 3 until convergence
Theoretical Objective:
• **M-step Goal:** $\theta^{t+1} = \text{argmax}_{\theta} E_{z \sim p(z|\theta^t)} [\log p(X, z|\theta)]$
• **Decomposition:** $\log p(X|\theta) = L(q, \theta) + KL(q||p(z|\theta))$
• $\log p(X|\theta) \geq L(q, \theta)$ (Lower Bound)
• $L(q, \theta) = E_{z \sim q} [\log \frac{p(X, z|\theta)}{q(z)}] = \dots + H(q)$

Model Selection & Hierarchical Clustering

Choosing K (Number of Clusters):
1. **Bayesian Information Criterion (BIC)**
• **Formula:** $BIC = M \log N - 2 \log L$
• $M = K(1 + D + D^2)$ (Number of parameters)
• N : Sample size
• L : Observed log-likelihood
• **Goal:** Minimize BIC.
2. **AIC (Akaike Information Criterion)**
• **Formula:** $AIC = 2M - 2 \log L$
3. **Agglomerative Clustering (Hierarchical)**
• **Method:** Merge clusters recursively.
• **Description:** Merge classes gradually based on distance.
Plaintext

$\{A\} \{B\} \{C\} \{D\} \quad \leftarrow$ Merging from bottom up

Misc

Convex Functions

Properties of Convex Functions:
1. $g(x) = g_1(x) + g_2(x)$ (Sum of convex functions is convex)
2. $g(x) = \max\{g_1(x), g_2(x)\}$ (Pointwise maximum is convex)
3. $g(x) = c \cdot g_1(x)$ where $c \geq 0$ (Non-negative scaling)
4. $g(x) = c \cdot f_1(x)$ where $c \leq 0$ and $f_1(x)$ is concave
5. $g(x) = g_1(Ax + b)$ (Affine transformation)
Isotropic Gaussian & Priors
Isotropic Gaussian:
(Refers to Gaussian distributions with covariance proportional to identity, i.e., spherical)
Priors:
• **Laplace Prior:** $\exp(-\frac{\|w\|_1}{\lambda})$
• **Gaussian Prior:** $\exp(-\frac{\|w\|_2^2}{2\sigma^2})$
Key Differences:
• Laplace encourages sparsity (results in sparse solutions, like L1 regularization).
• Gaussian results in solutions around the mean (like L2 regularization).
• Laplace results in solutions around the median.