Part 1: Classical Classification Algorithms

K-Nearest Neighbors (KNN)

- **Type:** Non-parametric, Instance-based (Lazy Learning).
- **Prediction Rule:** Assign the majority class among the k nearest neighbors (using a distance metric, typically Euclidean).
- **Pros**
  - Simple and easy to implement
  - No training phase (lazy learning)
  - Naturally handles non-linear decision boundaries
  - Effective for multi-class problems
- **Cons**
  - Slow inference: Computationally expensive at prediction time (calculates distance to all training points)
  - Memory intensive: Must store the entire training dataset
  - Sensitive to irrelevant or redundant features
  - Curse of dimensionality: Distance metrics lose meaning in very high dimensions; performance degrades with high-dimensional data
  - Requires careful choice of k, distance metric, and feature scaling

Bayes Optimal Classifier

- **Decision Rule:** $\hat{y} = \arg\max_y P(y|x)$
- **Pros**
  - Theoretically optimal (lowest possible error rate)
  - Provides true posterior probabilities
  - Serves as benchmark for all classifiers
- **Cons**
  - Intractable in practice (requires full knowledge of distributions)
  - Cannot be computed directly for most real problems

Naive Bayes (NB)

- **Type:** Generative Model ($P(x|y)$).
- **Decision Rule:** $\hat{y} = \arg\max_y P(y) \prod_i P(x_i|y)$ (using independence assumption)
- **Pros**
  - Extremely fast training and prediction
  - Performs surprisingly well even when independence assumption is violated
  - Works well with high-dimensional data (e.g., text/spam classification)
  - Robust to irrelevant features
  - Handles missing values naturally
  - Good for text classification (e.g., spam detection)
- **Cons**
  - Strong independence assumption: Assumes features are independent given the class (often violated)
  - Zero frequency problem: Requires smoothing (e.g., Laplace) for unseen features
  - Poor probability estimates (though good for classification; outputs often poorly calibrated and too extreme)

Linear Discriminant Analysis (LDA)

- **Type:** Generative Model.
- **Assumption:** Classes are Gaussian with shared covariance Σ.
- **Discriminant Function:** $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$
- **Decision Boundary:** Linear (due to shared covariance).
- **Pros**
  - Supervised dimensionality reduction: Projects data to maximize class separability (max dimensions = $C-1$); uses class labels
  - Optimal classifier if data is Gaussian with equal covariance matrices
  - Provides good class separation
  - Computationally efficient (closed-form solution)
- **Cons**
  - Strict assumptions: Requires Gaussian distribution and shared covariance matrix across classes
  - Sensitive to outliers
  - Limited to linear decision boundaries
  - Not suitable for non-Gaussian data

Naive Bayes vs. LDA

Feature	Naive Bayes	LDA
Assumption	Features are independent (Diagonal Covariance)	Features share correlations (Full Shared Covariance)
Similarities	Both assume Gaussian feature distributions (Gaussian NB); Both generative models; Both derive from Bayes theorem	
Differences	Faster and works better with limited data; Can have non-linear (quadratic) boundaries	Generally more accurate when correlations exist; Needs more data to estimate covariance matrix; Strictly linear boundaries
Data Efficiency	Better with small data	Needs more data to estimate covariance matrix

Logistic Regression (LR)

- **Type:** Discriminative Model ($P(y|x)$).
- **Model:** $P(y=1|x) = \sigma(w^T x + b) = \frac{1}{1+e^{-(w^T x + b)}}$ (Sigmoid function)
- **Pros**
  - Outputs well-calibrated probabilities (not just classifications)
  - Highly interpretable coefficients (feature importance)
  - Efficient training; easy to update with new data (using SGD)
  - Robust to noise
  - Works well when classes are linearly separable
- **Cons**
  - Inherently linear decision boundary (unless feature engineering)
  - Sensitive to outliers
  - Can overfit on high-dimensional data without regularization
  - Requires careful feature scaling

Logistic Regression vs. Generative Models (NB/LDA)

Feature	Logistic Regression	NB / LDA
Type	Discriminative (models $P(y\|x)$ directly)	Generative (models $P(x\|y)$ and $P(y)$)
Assumptions	Few assumptions about data distribution	Strong assumptions (Gaussian, Independence)

Feature	Logistic Regression	NB / LDA
Similarities	All provide probabilistic outputs; All can handle multi-class problems	
Performance	Generally higher accuracy with sufficient data; Often performs better in practice; Doesn't assume feature distribution	Better with missing data or very small datasets

Support Vector Machines (SVM)

- **Type:** Discriminative (Max Margin).
- **Hard Margin Objective:** Maximize margin $\frac{2}{\|w\|}$ subject to $y_i(w^T x_i + b) \geq 1$
- **Pros**
  - Max margin: Theoretically robust to overfitting
  - Effective in high-dimensional spaces (even when $d > n$)
  - Memory efficient (defined only by support vectors)
  - Works well with clear margin separation
- **Cons**
  - No direct probability estimates (requires Platt scaling)
  - Sensitive to noise, parameter tuning (C), and choice
  - Poor performance with overlapping classes
  - Training computationally expensive on large datasets

Kernel SVM

- **Kernel Trick:** Maps data to higher dimension for non-linear boundaries.
- **Pros**
  - Handles non-linear data
  - Flexible with different kernels
  - Powerful for complex boundaries
- **Cons**
  - Computationally expensive ($O(n^2)$ to $O(n^3)$; kernel matrix)
  - Harder to interpret
  - Risk of overfitting with wrong kernel

Kernel SVM vs. Linear SVM

Feature	Linear SVM	Kernel SVM
Similarities	Both maximize margin; Both use support vectors	
Differences	Faster, simpler	Handles non-linear data but slower

Part 2: Regression & Optimization

Linear Regression Variants

1. **Standard Linear Regression**
   - Objective: Minimize $\sum(y_i - \hat{y}_i)^2$ (Sum of Squared Errors, SSE).
   - **Pros:** Simple and interpretable; Fast training/prediction; Closed-form solution; Good baseline
   - **Cons:** Assumes linear relationship; Sensitive to outliers; Unstable with correlated features/multicollinearity; Poor with non-linear data
2. **Ridge Regression (L2)**
   - Objective: Minimize $\sum(y_i - \hat{y}_i)^2 + \lambda\|w\|_2^2$
   - **Pros:** Handles multicollinearity; Reduces overfitting; Stable with noisy data
   - **Cons:** Doesn't perform feature selection; Bias introduced; Requires tuning lambda; Shrinks coefficients but keeps all

3. **LASSO Regression (L1)**
    - Objective: Minimize $\sum(y_i - \hat{y}_i)^2 + \lambda\|w\|_1$
    - **Pros:** Induces sparsity (sets some weights to zero); Acts as feature selection; Good for high-dimensional data; Handles sparse solutions
    - **Cons:** Unstable with correlated features (can select only one from group)

Ridge vs. LASSO Regression

Feature	Ridge	LASSO
Similarities	Both regularized linear regression; Both shrink coefficients	
Differences	L2 penalty (shrinks but keeps all)	L1 penalty (can zero out coefficients)

Non-linear Regression

- **Pros**
    - Can model complex relationships
    - Flexible function forms
- **Cons**
    - Risk of overfitting
    - Harder to interpret
    - Computationally expensive

Kernel Ridge Regression

- **Pros**
    - Combines kernel trick with ridge
    - Handles non-linear data
    - Closed-form solution
- **Cons**
    - $O(n^3)$ complexity
    - No sparsity

Kernel Ridge vs. Ridge Regression

Feature	Ridge	Kernel Ridge
Similarities	Both use L2 regularization	
Differences	Linear only	Non-linear in input space

Support Vector Regression

- **Pros**
    - Robust to outliers (epsilon tube)
    - Handles non-linear via kernels
    - Good generalization
- **Cons**
    - Sensitive to parameters
    - Slow training on large data

Gradient Descent (GD) Variants

Variant	Batch Size	Pros	Cons
Batch GD	All Data	Stable convergence; exact gradient; Guaranteed convergence to minimum for convex functions	Slow on large datasets; High memory usage; Can get stuck in saddle/flat regions
Stochastic GD	Sample	Fast updates; Escapes local minima; Low memory; Good for online learning	Noisy convergence; May not converge exactly; Requires learning rate decay/tuning
Mini-Batch GD	N Samples	Best of both worlds: Balances speed and stability; Utilizes GPU vectorization; Lower variance than SGD; Good convergence properties	Requires tuning batch size; Still some noise in updates

Batch GD vs. Mini-batch GD vs. Stochastic GD

- **Similarities:** All minimize the same objective function; All use gradient information
- **Differences:** Batch: stable but slow; Mini-batch: compromise (most common in practice); Stochastic: fast but noisy

Coordinate Descent

- **Pros**
  - Simple implementation
  - Fast for sparse problems
  - Effective for L1 regularization (Lasso)
- **Cons**
  - May converge slowly
  - Order of coordinates matters
  - Not parallelizable easily

RANSAC (Random Sample Consensus)

- **Purpose:** Robust fitting of models (e.g., lines, homographies) in the presence of many outliers.
- **Pros:** Extremely robust to outliers/high outlier ratios; Simple concept
- **Cons:** Non-deterministic; Requires threshold parameter and many iterations; No guarantee of optimal solution

Part 3: Unsupervised Learning & Clustering

Clustering Algorithms

- **Exhaustive Clustering**
  - **Pros:** Finds optimal clustering
  - **Cons:** Computationally infeasible
- **K-Means**
  - Objective: Minimize $\sum_{k=1}^{K}\sum_{i\in C_k}\|x_i-\mu_k\|^2$ (Within-cluster sum of squares)
  - Hard assignment; Assumes spherical clusters of similar size.
  - **Pros:** Simple and fast; Scales well; Easy to implement
  - **Cons:** Sensitive to initialization and outliers; Requires choosing k
- **Gaussian Mixture Models (GMM)**
  - Probability: $p(x)=\sum_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)$
  - Soft assignment (probabilistic); Assumes elliptical clusters; Uses Expectation-Maximization (EM).
  - **Pros:** Handles elliptical clusters; Can model complex distributions; Provides probabilities
  - **Cons:** Sensitive to initialization; Can overfit; Computationally heavier; Slower

K-Means vs. Gaussian Mixture Model

Feature	K-Means	GMM
Similarities	Both centroid-based; Use EM-like optimization	
Differences	Hard assignment, spherical clusters	Soft assignment, elliptical clusters

Expectation-Maximization (EM) Algorithm

- **Pros**
  - Handles latent variables
  - Monotonic improvement
  - Flexible for many models
- **Cons**
  - Local optima
  - Slow convergence
  - Sensitive to initialization

Part 4: Dimensionality Reduction & Feature Processing

Dimensionality Reduction

- **Pros**
  - Reduces computation
  - Removes noise/redundancy
  - Helps visualization
  - Mitigates curse of dimensionality
- **Cons**
  - Loss of information
  - Harder interpretation

Feature Selection

- **Pros**
  - Maintains interpretability
  - Reduces overfitting
  - Faster training
- **Cons**
  - May miss combined effects
  - Computationally expensive (some methods)

Dimensionality Reduction vs. Feature Selection

Feature	Feature Selection	Dimensionality Reduction
Similarities	Both reduce features	
Differences	Subset of original features	New transformed features

Linear Dimensionality Reduction

- **Pros**
  - Fast and simple
  - Interpretable
- **Cons**
  - Assumes linear relationships
  - May miss complex structures

Singular Value Decomposition (SVD)

- **Pros**
  - Numerically stable
  - Basis for many methods
  - Handles any matrix
- **Cons**
  - Linear only
  - Computationally intensive for large matrices
- The mathematical matrix factorization technique that underpins PCA; More numerically stable than eigendecomposition.

Principal Component Analysis (PCA)

- **Unsupervised.**
- Finds orthogonal directions of maximum variance (eigenvectors of covariance matrix, ordered by eigenvalues).
- **Pros**
  - Maximizes variance
  - Orthogonal components
  - Reduces dimensions effectively
- **Cons**
  - Sensitive to scaling
  - Assumes linear correlations
  - Interprets variance as importance ("Variance" does not always equal "Information")

PCA vs. SVD vs. Dimensionality Reduction

- **Similarities:** PCA uses SVD; All linear techniques
- **Differences:** SVD: general matrix decomposition; PCA: SVD on centered data for variance maximization; Dimensionality reduction: broader category

Feature Normalization / Data Whitening

- **Pros**
  - Improves convergence speed/optimization
  - Prevents feature dominance
  - Essential for distance-based algorithms
  - Decorrelates features; Equalizes variances
- **Cons**
  - Can leak test information if not careful
  - May distort data distribution/amplify noise
  - Assumes linear correlations
  - Computationally expensive (whitening)

Output Transformation

- **Pros**
  - Can stabilize variance
  - Make data more normal-like
  - Help meet model assumptions
- **Cons**
  - Harder interpretation
  - Need inverse transform for predictions

Part 5: Deep Learning

Architectures

- **Multi-Layer Perceptron (MLP)**
  - Dense connections; Good for tabular data.
  - **Pros**
    * Universal approximator
    * Handles non-linear data
    * Flexible architecture
  - **Cons**
    * Black box
    * Prone to overfitting on images
    * Requires much data
- **Convolutional Neural Network (CNN)**
  - **Inductive Bias:** Translation invariance and locality; Parameter sharing.
  - **Pros**
    * Excellent for images/grid data
    * Learns hierarchical features
    * Efficient due to parameter sharing
  - **Cons**
    * Requires large data
    * Computationally expensive
    * Less effective on non-spatial data

MLP vs. CNN

Feature	MLP	CNN
Similarities	Both deep neural networks; Both use backpropagation	
Differences	Fully connected	Local connectivity, weight sharing

Regularization & Training Techniques

- **Dropout**
  - Randomly deactivates neurons during training; Prevents co-adaptation (ensemble effect).
  - **Pros**
    * Effective regularization
    * Reduces co-adaptation
  - **Cons**
    * Increases training time
    * Requires larger networks
    * Inference needs scaling
- **Early Stopping**
  - Stops training when validation error rises.
  - **Pros**
    * Prevents overfitting
    * Saves training time
    * Simple to implement
  - **Cons**
    * Requires validation set

    * May stop too early
    * Sensitive to patience parameter

- **Data Augmentation**
  - Artificially expands dataset (flip, rotate, crop).
  - **Pros**
    * Increases effective dataset size
    * Improves generalization
    * Reduces overfitting
    * Cost-effective; Crucial for computer vision
  - **Cons**
    * Can introduce unrealistic samples
    * Increases training time
    * Domain-specific design needed
- **Batch Normalization / Whitening**
  - Normalizes layer inputs (zero mean, unit variance).
  - **Pros**
    * Stabilizes learning
    * Allows higher learning rates
  - **Cons** (see Data Whitening above)

Early Stopping vs. Dropout

Feature	Early Stopping	Dropout
Similarities	Both prevent overfitting; Both regularization techniques	
Differences	Stops training early	Random neuron drop during training

Weight Initialization

- **Methods**
  - Zero (bad)
  - Random normal/uniform
  - Xavier/Glorot (for sigmoid/tanh)
  - He (for ReLU)
  - Orthogonal

Hyperparameter Tuning

- **Procedures**
  - Grid search
  - Random search
  - Bayesian optimization
  - Manual tuning
  - Cross-validation

Learning Curve Interpretation

- **Accuracy still going up:** Model has capacity to learn more → Needs more data or longer training
- **Huge gap between train and validation accuracy:** High variance/overfitting → Needs regularization or more data
- **Little gap between train and validation:** Low variance → If both low: high bias/underfitting

Part 6: Computer Vision & Image Processing

Tasks

- **Object Detection:** Locates and classifies objects with bounding boxes (e.g., YOLO, Faster R-CNN)
- **Semantic Segmentation:** Pixel-level classification (all instances of same class share label)
- **Instance Segmentation:** Pixel-level classification distinguishing individual objects (different instances of same class)

Image Restoration & Processing

- **Denoising:** Removing noise (e.g., Filtering, wavelet, deep learning like DnCNN)
  - **Pros:** Improves image quality; Enhances downstream tasks
  - **Cons:** May blur details; Computationally intensive (deep methods)
- **Deblurring:** Removes motion/camera shake blur (harder than denoising)
- **Super-Resolution:** Upscaling images; GANs often used to hallucinate details
  - **Pros:** Enhances details
  - **Cons:** Can introduce artifacts
- **Colorization:** Adds color to grayscale
  - **Pros:** Enhances old photos
  - **Cons:** Subjective; Can be inaccurate
- **Compression:** Reduces file size
  - **Pros:** Storage/transmission efficiency
  - **Cons:** Loss of quality (lossy)
- **Fusion:** Combines multiple images (multi-modal)
  - **Pros:** Richer information
  - **Cons:** Alignment issues

Quality Metrics

- **MSE/PSNR**
  - **Pros:** Simple, differentiable; Good for optimization
  - **Cons:** Poor correlation with human perception (penalizes slight shifts heavily); Sensitive to outliers
- **SSIM (Structural Similarity)**
  - Captures luminance, contrast, and structure.
  - **Pros:** Closer to human vision; Better perceptual quality
  - **Cons:** More complex; Sensitive to scaling/alignment
- **LPIPS (Learned Perceptual Image Patch Similarity)**
  - Uses deep network features.
  - **Pros:** Best match for human perception
  - **Cons:** Computationally heavy; Requires pretrained network

MSE vs. SSIM

Feature	MSE	SSIM
Similarities	Both full-reference metrics	
Differences	Pixel-wise error	Structural/perceptual similarity

General Notes

- **Model Comparison:** Use multiple metrics (accuracy, speed, interpretability) and validation techniques