Linear Algebra & SVD/PCA

Vectors & Matrices:
- Vector: $\vec{v} = (v_1, v_2, ..., v_n)$
- Matrix: $A_{m \times n}$
- Transpose: $A^\top$
- Dot product: $\vec{a} \cdot \vec{b} = \sum_i a_i b_i$

Eigenvalues/Eigenvectors:
- $A\vec{x} = \lambda\vec{x}$
- Principal directions (PCA): Variance maximization.
- SVD: $A = U\Sigma V^\top$

U, V: orthogonal; Σ: singular values. Reduces dimensionality.
PCA Steps:
1. Standardize data. 2. Compute covariance. 3. Eigendecomposition. 4. Select top-k directions.

Probability & Bayes

Basic Rules:
- $P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A|evidence) = \frac{P(evidence|A)P(A)}{P(evidence)}$
- Independent: $P(A \cap B) = P(A)P(B)$

Random Variables: Discrete or continuous.
Expectation: $E[X] = \sum xP(x)$ or $\int xp(x)dx$

Bayes' Theorem Example:
If $P(G) = 0.5$, $P(B) = 0.5$, $P(C|G) = 0.6$, $P(C|B) = 0.1$, then $P(G|C) = \frac{P(C|G)P(G)}{P(C|G)P(G)+P(C|B)P(B)}$

Naive Bayes Classification

Assumes: features independent conditioned on class.
Classify x: Compute $P(C_k|x) \propto \prod_j P(x_j|C_k)P(C_k)$
Pick class k maximizing $P(C_k|x)$.

Example: Classify email as spam/not spam:
$P(spam|x) \propto P(w_1|spam)...P(w_n|spam)P(spam)$

Linear Regression

Model: $h_\theta(x) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$
Loss: $L(\theta) = \frac{1}{n}\sum(h_\theta(x^{(i)}) - y^{(i)})^2$
Normal Equation: $\theta = (X^\top X)^{-1}X^\top y$
Assumptions: Linearity, constant variance, independence.

Overfitting: Model fits training noise.
Underfitting: Model can't capture patterns.

Gradient Descent

Iteratively minimize loss $L(\theta)$.
Update: $\theta_j \leftarrow \theta_j - \alpha\frac{\partial L}{\partial \theta_j}$
Stochastic: Uses single data point per step.
Batch: Uses all data per step.
Convergence: Choose α (step size) carefully.

Model Complexity & Regularization

Bias: Error from overly simplistic assumptions.
Variance: Error from model sensitivity to fluctuations.
Tradeoff: High bias $\rightarrow$ underfit; high variance $\rightarrow$ overfit.
Regularization: Penalizes large θ values.
- Ridge (L2): $L(\theta) + \lambda\sum_{j=1}^n \theta_j^2$
- Lasso (L1): $L(\theta) + \lambda\sum_{j=1}^n |\theta_j|$

Choose λ by cross-validation.
Do not regularize θ_0!

Cross-validation: Split training data into k folds; ensures better generalization.

Logistic Regression

Classifies: $y \in \{0, 1\}$
Sigmoid activation: $g(z) = \frac{1}{1+e^{-z}}$
Hypothesis: $h_\theta(x) = g(\theta^\top x)$
Probability Interpretation: $h_\theta(x) = P(y = 1|x, \theta)$

Decision boundary: $\theta^\top x = 0$
Loss: $L(\theta) = -\frac{1}{n}\sum[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$

Classification Metrics

Confusion Matrix: TP, FP, TN, FN.
Precision: $\frac{TP}{TP+FP}$
Recall: $\frac{TP}{TP+FN}$
Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$
F1 Score: $2\frac{Precision \cdot Recall}{Precision+Recall}$
Tune threshold for precision/recall tradeoff.

Multiclass Classification

OvA: Train k binary classifiers for k classes.
Classifier i: $h_\theta^{(i)}(x) = P(y = i|x)$
Predict: $\arg\max_i h_\theta^{(i)}(x)$

K-means Clustering

Unsupervised learning: No labels.
Input: $x_1, ..., x_m \in R^n$, k clusters.
1. Randomly initialize centers $\mu_1, ..., \mu_k$
2. Assign each point to closest center.
3. Update centers to cluster means.
4. Repeat until convergence.

Objective: Minimize $J = \sum_{i=1}^m ||x^{(i)} - \mu_{c^{(i)}}||^2$
K-means++: Smarter initialization picks distant points as centers.
Elbow method: Plot loss vs k; choose at "elbow".

Hierarchical Clustering

Agglomerative: Start with each point in its own cluster; iteratively merge closest clusters.
Linkage methods:

- **Single**: Closest pair.
- **Complete**: Furthest pair.
- **Average**: Mean of all pairs.

Produces dendrogram showing cluster hierarchy.
Cut tree at desired depth to select k clusters.

Feature Engineering

- **Selection**: Remove irrelevant/redundant features. - **Extraction**: Build new features (e.g. PCA). - **Scaling**: Standardize (zero mean, unit variance). - **Encoding**: Convert categoricals (one-hot, ordinal). - **Handling missing values**: Impute or drop.

Exam Tips & Common Pitfalls

- Always split data into train/test sets (never look at test until the end!). - Use cross-validation for hyperparameter tuning. - Regularization cures overfitting, not underfitting. - Don't regularize intercept term. - When multiclass, use one-vs-all for logistic regression. - For K-means, rerun multiple times to avoid poor local minima.
Maximum Likelihood Estimation (MLE):
Choose parameters maximizing probability of observed data:
$\theta_{MLE} = \arg\max_\theta P(\mathbf{y}|\mathbf{X}, \theta)$
Covariance Matrix (PCA): $C = \frac{1}{n}\sum_{i=1}^n (x^{(i)} - \mu)(x^{(i)} - \mu)^\top$

Softmax for Multiclass Logistic:
$softmax(z_k) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$
Predict class with highest probability.
Regularization in Gradient Descent:
Update: $\theta_j \leftarrow \theta_j - \alpha\left[\frac{\partial L}{\partial \theta_j} + \lambda \cdot \theta_j\right]$
Penalizes large parameter values.
Loss Functions:
- Linear regression: MSE.
- Logistic regression: Cross-entropy.
- K-means: Sum of squared errors.

Hyperparameters:
- Set before training (λ, learning rate, k, degree, epochs). - Use grid search, cross-validation to choose.
Overfitting symptoms: Training error $\ll$ test error.
Underfitting: High error everywhere.
Dimensionality Reduction: Reduce features, mitigate curse of dimensionality.
Gradient Check: Numerically approximate derivative to check implementation.

Quick Formula Reference

Gradient: $\nabla_\theta L(\theta)$
Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$
Softmax: $softmax(z_k) = \frac{e^{z_k}}{\sum_j e^{z_j}}$
Covariance: $Cov(X,Y) = E[(X - E[X])(Y - E[Y])]$
Standardization: $x^{(i)}_{std} = \frac{x^{(i)}-\mu}{\sigma}$
Loss: $L(\theta)$

Worked Example: Ridge Regression

Minimize $L(\theta) = \frac{1}{n}\sum(x_i - 4\theta)^2 + \lambda(\theta - 3)^2$
Solution: $\theta = \frac{4\bar{x}+3\lambda n}{16+\lambda n}$
As $\lambda \to 0$: normal equation.
As $\lambda \to \infty$: $\theta \to 3$.

Worked Example: Logistic Regression

Given features $x = [1, 0.7, 0.5]$, θ,
$h_\theta(x) = g(\theta^\top x)$.
If $h_\theta(x) = 0.8$, $P(y = 1|x) = 0.8$

Worked Example: K-means

Points: (0,0),(0,1),(1,0),(2,2),(3,3); $k = 2$;
1. Random centers.
2. Assign clusters.
3. Update means.
4. Repeat until cluster assignments don't change.

Worked Example: Hierarchical (Single Linkage)

Points: A(0,0), B(1,0), C(0,1), D(4,4), E(5,4)
1. Compute pairwise distances.
2. Merge closest points.
3. Repeat until single cluster. 4. Visualize with dendrogram.

Precision-Recall Example

Actual positives: 10. Actual negatives: 10.
Predicted TP=8, FP=3, TN=7, FN=2.
Precision=$8/11 \approx 0.73$
Recall=$8/10 = 0.8$
Accuracy=$15/20 = 0.75$

Clustering Tips

- K-means works best for spherical clusters. - Hierarchical: Use average linkage for balanced clusters. - Outliers: K-means++ avoids picking them as centers.

Feature Engineering Tricks

- Use PCA after scaling. - One-hot encoding for categoricals. - Standardize features for regularized models.

General ML Strategy

1. Choose model. 2. Split train/test. 3. Feature engineer. 4. Hyperparameter tune (CV). 5. Train/finalize. 6. Evaluate only once on test.

Common Pitfalls

- Using test data during training - Not scaling before regularization (L1/L2) - Ignoring precision vs. recall when classes imbalanced - Only checking accuracy (use confusion matrix, F1, etc.)