CS 439 Exam 02 Cheat Sheet

Linear Algebra Foundations

Vector Operations $\mathbf{v} \cdot \mathbf{w} = \sum v_i w_i$, $\|\mathbf{v}\| = \sqrt{\sum v_i^2}$
Orthogonal Two vectors orthogonal if $\mathbf{v} \cdot \mathbf{w} = 0$
Linear Independence Set $\{v_1, ..., v_n\}$ independent if no vector is linear combination of others.
Eigenvalues/Eigenvectors $A\mathbf{v} = \lambda\mathbf{v}$ where λ is eigenvalue, $\mathbf{v}$ is eigenvector.
Rank Max number of linearly independent rows or columns.
Span Set of all linear combinations of a set of vectors.
Basis Set of vectors that span a vector space.

SVD and PCA

SVD $X_{n\times m} = U_{n\times n}\Sigma_{n\times m}V^T_{m\times m}$ where U and V are orthogonal; Σ diagonal with singular values (square roots of eigenvalues of X^TX).
Rank-k Approximation Use first k singular values: $X_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$
PCA Dimensionality reduction by projecting onto principal components (directions of max variance). Steps: center data, compute covariance matrix, find eigenvalues/eigenvectors, sort by eigenvalues, select top k eigenvectors, project data.
Why? Reduces dimensionality, removes noise, improves computational efficiency.
Explained Variance Proportion of variance captured by each principal component: $\frac{\lambda_i}{\sum \lambda_j}$ where λ_i are eigenvalues.
Covariance Matrix $C = \frac{1}{n-1}X^TX$ where X is centered data matrix.

Probability & Bayes

Conditional Probability $P(A|B) = \frac{P(A\cap B)}{P(B)}$
(Marginal) Independence $P(A, B) = P(A) \cdot P(B)$ (test by checking if $P(A \cap B) = P(A)P(B)$)
Bayes Theorem $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ where $P(B) = P(B|A)P(A) + P(B|\neg A)P(\neg A)$ (law of total probability)
Marginal Probability $P(A) = \sum_i P(A, B_i) = \sum_i P(A|B_i)P(B_i)$
Joint Probability $P(A, B)$ vs $P(A) = \sum_B P(A, B)$
Conditional Independence $P(A, B|C) = P(A|C)P(B|C)$
Product Rule $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$
Chain Rule $P(A, B, C) = P(A|B, C)P(B|C)P(C)$

Naive Bayes Classification

Assumption Features conditionally independent given class: $P(X_1, ..., X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$. Can ignore denominator $P(X)$ when comparing classes.
Classification Rule $\hat{y} = \arg\max_y P(Y = y) \prod_{i=1}^{n} P(X_i|Y = y)$
Steps 1) Calculate priors: $P(Y) = \frac{\text{count}(Y)}{\text{total}}$. 2) Calculate likelihoods: $P(X_i|Y) = \frac{\text{count}(X_i \cap Y)}{\text{count}(Y)}$. 3) Compute scores (drop denominator). 4) Choose class with max score.
Laplace Smoothing Add 1 to counts to handle zero probabilities: $P(X_i|Y) = \frac{\text{count}(X_i \cap Y)+1}{\text{count}(Y)+|V|}$
Types of NB: Gaussian (continuous features), Multinomial (discrete counts), Bernoulli (binary features).

Linear Regression

Assumptions Linearity, independence, homoscedasticity, normality of errors.
Hypothesis $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n = \theta^T\mathbf{x}$
MSE (L2) $L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2$
MAE (L1) $L(\theta) = \frac{1}{n}\sum_{i=1}^{n}|h_\theta(\mathbf{x}^{(i)}) - y^{(i)})|$
Huber Loss
$L(\theta) = \sum_{i=1}^{n} \begin{cases} \frac{1}{2}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2 & \text{if } |h_\theta(\mathbf{x}^{(i)}) - y^{(i)}| \le \delta \\ \delta(|h_\theta(\mathbf{x}^{(i)}) - y^{(i)}| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$ It is quadratic for small errors and linear for large errors: robust to outliers.
Normal Equation $\theta = (X^TX)^{-1}X^T\mathbf{y}$ (closed-form solution)
Simple Linear Reg $y = \beta_0 + \beta_1 x$ where $\beta_1 = \frac{\text{cov}(X,Y)}{\text{var}(X)} = \frac{\sum(X-\bar{X})(Y-\bar{Y})}{\sum(X-\bar{X})^2}$ and $\beta_0 = \bar{Y} - \beta_1\bar{X}$
R^2 **Score** $R^2 = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2}$ (closer to 1 is better)
Model Eval Use train/test split (e.g., 80/20) or cross-validation to assess generalization. Look at residual plots for patterns. R^2 can be used to evaluate model performance.
Lin Reg Shortcut $\beta_1 = \frac{\text{cov}(X,Y)}{\text{var}(X)} = \frac{\sum(X-\bar{X})(Y-\bar{Y})}{\sum(X-\bar{X})^2} = \frac{\sum XY - n\bar{X}\bar{Y}}{\sum X^2 - n\bar{X}^2}$. $\beta_0 = \bar{Y} - \beta_1\bar{X}$
Worked Example: Points $(1, 2), (2, 3), (3, 5)$ We fit $y = \beta_0 + \beta_1 x$. Calculate $\beta_1 = \frac{(1-2)(2-10/3)+(2-2)(3-10/3)+(3-2)(5-10/3)}{(1-2)^2+(2-2)^2+(3-2)^2} = \frac{(-1)(-4/3)+0+(1)(5/3)}{1+0+1} = \frac{4/3+5/3}{2} = \frac{9/3}{2} = \frac{3}{2} = 1.5$. Then $\beta_0 = \bar{Y} - \beta_1\bar{X} = 10/3 - 1.5 * 2 = 10/3 - 3 = 1/3$.

Gradient Descent

Update Rule $\theta_j := \theta_j - \alpha\frac{\partial L}{\partial \theta_j}$ (simultaneously update all θ_j, i.e., calculate all gradients first)
For Linear Reg $\theta_j := \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)}$
Learning Rate (α) Too large → oscillation/divergence. Too small → slow convergence. Plot loss vs iterations to check.
Convergence Stop when $|L_{t+1} - L_t| < \epsilon$ or max iterations reached. Loss should decrease monotonically. Feature scaling $[-1, 1]$ or $[0, 1]$ helps.
Batch GD Use all training examples. **Stochastic GD** Use one example at a time. **Mini-batch GD** Use subset of examples.
Worked Example $f(\theta) = (\theta - 4)^2$, $\frac{df}{d\theta} = 2(\theta - 4)$. Start $\theta = 0$, $\alpha = 0.1$. Iter 1: $\theta = 0 - 0.1 * 2(0 - 4) = 0 + 0.8 = 0.8$. Iter 2: $\theta = 0.8 - 0.1 * 2(0.8 - 4) = 0.8 + 0.64 = 1.44$. Iter 3: $\theta = 1.44 - 0.1 * 2(1.44 - 4) = 1.44 + 0.512 = 1.952$. Continue until convergence.

Model Complexity & Regularization

Bias-Variance Tradeoff High bias (underfitting): model too simple. High variance (overfitting): model too complex, fits noise.
Regularization Add penalty to prevent overfitting: $L(\theta) = \text{MSE} + \lambda\sum_{j=1}^{n}\theta_j^2$ (don't penalize θ_0)
Why? Constrains model complexity to improve generalization.
L2 (Ridge) Penalty $= \lambda\sum\theta_j^2$ (shrinks coefficients)
L1 (Lasso) Penalty $= \lambda\sum|\theta_j|$ (feature selection, sets some θ to 0)
λ **Selection** $\lambda = 0$: no regularization (may overfit). Large λ: more regularization (may underfit). Use cross-validation. Try $0, .01, 0.1, 1, 10, 100$.
Train/Test Split Typically 80-20 or 70-30. Train on training set, evaluate on test set. Never train on test data!
Cross-Validation K-fold: split data into k subsets, train on $k-1$, test on 1, repeat k times, average results. Split into $60-80\%$ train, $10-20\%$ validation, $10-20\%$ test for hyperparameter tuning.
WorkFlow: 1) Split data into train/validation/test sets. 2) Train model on training set. 3) Tune hyperparameters using validation set. 4) Evaluate final model on test set. 5) Use cross-validation for more robust performance estimates.
Gradient Descent Regularization Update Rule: $\theta_j := \theta_j - \alpha\left(\frac{1}{n}\sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{n}\theta_j\right)$ (for L2 regularization, do not regularize θ_0)

Logistic Regression

Sigmoid Function $g(z) = \frac{1}{1+e^{-z}}$ (output between 0 and 1). $g(0) = 0.5$, $g(z) = 1 - g(-z)$, $g'(z) = g(z)(1 - g(z))$
Hypothesis $h_\theta(\mathbf{x}) = g(\theta^T\mathbf{x}) = \frac{1}{1+e^{-\theta^T\mathbf{x}}}$
Interpretation $h_\theta(\mathbf{x}) = P(y = 1|\mathbf{x}, \theta)$ (probability of class 1)
Decision Rule Predict $y = 1$ if $h_\theta(\mathbf{x}) \ge 0.5$ (equivalently, $\theta^T\mathbf{x} \ge 0$). Predict $y = 0$ if $h_\theta(\mathbf{x}) < 0.5$.
Logistic Loss $L(h, y) = -y\log(h) - (1 - y)\log(1 - h)$ or $L = -\frac{1}{n}\sum[y^{(i)}\log(h(\mathbf{x}^{(i)})) + (1 - y^{(i)})\log(1 - h(\mathbf{x}^{(i)}))]$ used because MSE is not convex for logistic regression.
Decision Boundary Line/curve where $h_\theta(\mathbf{x}) = 0.5$, i.e., $\theta^T\mathbf{x} = 0$. For 2D: $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$.
Non-linear Decision Boundary Use polynomial features (e.g., x_1^2, x_2^2, x_1x_2) to create non-linear boundaries.
Gradient Descent $\theta_j := \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)}$ (same form as linear regression!)
Worked Example Data: (1,0), (2,0), (3,1), (4,1). Start $\theta = [0, 0]$, $\alpha = 0.1$. Iter 1: Compute h_θ for each point, calculate gradients, update θ. Repeat until convergence.

Classification Metrics

Confusion Matrix TP (true pos), FP (false pos), TN (true neg), FN (false neg)
Precision Accuracy of positive predictions: $\frac{TP}{TP+FP}$ (how many predicted positives are correct?)
Recall (Sensitivity) Coverage of actual positives: $\frac{TP}{TP+FN}$ (how many actual positives did we catch?)
Accuracy Overall correctness: $\frac{TP+TN}{TP+TN+FP+FN}$
F1-Score Harmonic mean of precision and recall: $F1 = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$
Specificity True negative rate: $\frac{TN}{TN+FP}$
Trade-off High precision → fewer false positives (conservative). High recall → fewer false negatives (aggressive).
Use Case Accuracy good for balanced classes. Precision: when false positives costly. Recall: when false negatives costly. F1: balance precision and recall when classes imbalanced.
ROC AUC ROC: Plot TPR (recall) vs FPR ($\frac{FP}{FP+TN}$) at different thresholds. AUC: Area under ROC curve (closer to 1 is better).

Multiclass Classification

One-vs-All (OvA) For k classes, train k binary classifiers. Classifier i: class i (is positive) vs. all others (negative). Predict: $\hat{y} = \arg\max_i h_\theta^{(i)}(\mathbf{x})$.

Softmax Regression $P(y = k|\mathbf{x}) = \frac{e^{\theta_k^T \mathbf{x}}}{\sum_{j=1}^K e^{\theta_j^T \mathbf{x}}}$ (generalizes logistic regression to multiple classes)

Example: For 10 digit classes (0-9), train 10 classifiers in OvA or use softmax regression to predict digit.

K-means Clustering

Algorithm 1) Initialize: choose k random centers $\mu_1, ..., \mu_k$. 2) Assignment: assign each point to nearest center $c^{(i)} = \arg\min_j \|\mathbf{x}^{(i)} - \mu_j\|$. 3) Update: recompute centers as mean of assigned points $\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}^{(i)}$. 4) Repeat 2-3 until convergence.

Loss Function $J = \sum_{i=1}^m \|\mathbf{x}^{(i)} - \mu_{c(i)}\|^2$ (minimize sum of squared distances)

Convergence Loss decreases monotonically. Stop when change $< \epsilon$ or max iterations. May converge to local optimum (sensitive to initialization).

Distance Metrics Euclidean: $\sqrt{\sum (x_i - y_i)^2}$. Manhattan: $\sum |x_i - y_i|$. Cosine: $1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$.

Initialization Sensitivity Different initial centers can lead to different clusters. Run multiple times with different seeds and choose best result.

K-means++ Better initialization: choose centers with probability proportional to distance from existing centers. Formula: $P(x) = \frac{D(x)^2}{\sum D(x_i)^2}$ where $D(x)$ is distance to nearest center.

Choosing k Elbow method: plot loss vs. k, look for "elbow" where improvement slows.

Hierarchical Clustering

Agglomerative (Bottom-up) Start with each point as own cluster. Merge closest pair. Repeat until single cluster or desired k clusters.

Divisive (Top-down) Start with all points in one cluster. Recursively split until each point is own cluster or desired k clusters.

Linkage Methods Single: $\min\{d(x, y) : x \in C_1, y \in C_2\}$ (min distance between any two points). Complete: $\max\{d(x, y) : x \in C_1, y \in C_2\}$ (max distance). Average: $\frac{1}{|C_1||C_2|} \sum_{x \in C_1} \sum_{y \in C_2} d(x, y)$ (avg distance).

Dendrogram Tree diagram showing merge history. Height of merge indicates distance. Cut at desired height to get k clusters.

Advantages No need to specify k upfront. Captures nested clusters. Can visualize cluster hierarchy. Useful for small datasets.

Feature Engineering

Feature Selection Remove irrelevant/redundant features. Methods: filter (correlation), wrapper (forward/backward selection), embedded (regularization).

Feature Extraction Create new features from existing: polynomial features ($x^2, x^3, x_1 x_2$), domain-specific (ratios, differences).

Scaling/Normalization Z-score: $z = \frac{x - \mu}{\sigma}$. Min-max: $x' = \frac{x - \min}{max - min}$. Important for gradient descent and distance-based methods.

One-Hot Encoding Convert categorical to binary features. Category with n values $\to$ n binary features.

Important Concepts

Overfitting Model fits training data too well, performs poorly on test data. Solutions: regularization, more data, simpler model, cross-validation.

Underfitting Model too simple, performs poorly on both training and test data. Solutions: more features, more complex model, reduce regularization.

Parametric Models Fixed number of parameters (linear/logistic regression). Assumptions about data distribution.

Non-parametric Models Number of parameters grows with data (k-NN, decision trees). Fewer assumptions.

Supervised Learning Training data has labels (classification, regression).

Unsupervised Learning Training data has no labels (clustering, dimensionality reduction).

Learning Curves Plot of training/test error vs. training set size. Diagnose bias/variance issues.

Hyperparameter Tuning Use grid search or random search with cross-validation to find best hyperparameters (e.g., λ, k in K-means).

Curse of Dimensionality As dimensions increase, data becomes sparse. Distance metrics become less meaningful. Use dimensionality reduction (PCA) or feature selection.

Common Mistakes to Avoid

× Forgetting to calculate $P(B)$ in Bayes (use law of total probability)
× Not dropping denominator in Naive Bayes comparisons
× Penalizing θ_0 in regularization
× Not updating gradient descent parameters simultaneously
× Thinking K-means loss can increase (it can't!)
× Training on test data or using test data for hyperparameter tuning
× Forgetting to standardize features for distance-based methods
× Data Leakage, using future information in training

Quick Formula Reference

Bayes: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
Naive Bayes: $P(Y|\mathbf{X}) \propto P(Y) \prod P(X_i|Y)$
Linear Reg: $h = \theta^T \mathbf{x}$, MSE $= \frac{1}{n} \sum (h - y)^2$
Gradient: $\theta := \theta - \alpha \nabla L$
Regularized: $L = \text{MSE} + \lambda \sum \theta^2$
Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$
Logistic: $h = g(\theta^T \mathbf{x})$, Loss $= -y \log(h) - (1 - y) \log(1 - h)$
Precision: $\frac{TP}{TP+FP}$, **Recall:** $\frac{TP}{TP+FN}$
K-means: $J = \sum \|\mathbf{x} - \mu_c\|^2$
Normal Equation: $\theta = (X^T X)^{-1} X^T \mathbf{y}$
Normal Equation with L2 Reg : $\theta = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$
R^2 : $1 - \frac{\sum (y - \hat{y})^2}{\sum (y - \bar{y})^2}$
Softmax : $P(y = k|\mathbf{x}) = \frac{e^{\theta_k^T \mathbf{x}}}{\sum_j e^{\theta_j^T \mathbf{x}}}$