Lecture Notes

Lecture 10: Classification

- **Goal:** Predict a class label based on feature inputs (predictors).
- Models attempt to map attributes to class labels. Sometimes, multiple or no correct labels may exist due to noise or insufficient features.
- Even failed predictions give insights — "All models are wrong but some are useful".
- **Predictor Quality:**
  - Use correlation (e.g., Pearson or Spearman) to evaluate predictor relevance.
  - Spearman's rank correlation is used for ordinal or nonlinear data.
- **Data Types:**
  - **Nominal:** No inherent order (e.g., color, gender).
  - **Ordinal:** Ordered categories with unquantified distances (e.g., ratings).
- **Model Testing:**
  - Split data into training/testing sets.
  - Goal: generalize, not memorize (avoid overfitting).
  - Watch out for outliers and noise in data.
- **Instance-Based Classifiers:**
  - Store all training examples.
  - Match exact input for classification or output unknown.
- **K-Nearest Neighbor (KNN):**
  - Choose k, compute distance to all training points, classify by majority vote.
  - **Distances:** Euclidean, Manhattan, Hamming.
  - **Small k:** sensitive to noise. **Large k:** may blur class boundaries.
  - **Pros:** Intuitive, easy to implement. **Cons:** Slow, sensitive in high dimensions.

Lecture 11: Decision Trees

- Predict class labels via hierarchical yes/no questions.
- **Hunt's Algorithm:**
  - Recursively split data based on attributes to form pure nodes.
  - **Base cases:**
    - All data same class ⇒ predict that class.
    - Empty data ⇒ predict majority class.
- **Split Types:**
  - Binary: e.g., age > 30.
  - Multi-way: one branch per category value.
- **GINI Index:** Measures node impurity. Lower GINI = purer node.
- **Overfitting Solutions:**
  - **Early Stopping:** stop growing based on depth, node size, or GINI gain.
  - **Pruning:** remove subtrees post-training.

Lecture 12: Model Evaluation

- **Confusion Matrix:**

	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

- **Metrics:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$\text{Total Cost} = TP \cdot C_{TP} + FP \cdot C_{FP} + FN \cdot C_{FN} + TN \cdot C_{TN}$$

- **Evaluation Methods:**
  - Holdout: Train/test split (e.g., 75% train, 25% test).

  - K-Fold Cross Validation: Partition into k parts; rotate test fold.
  - Leave-One-Out: Special case of $k = n$.

- **Ensemble Methods:**
  - Combine multiple models to reduce error.
  - **Bagging:** Train models on bootstrap samples. Example: Random Forest.
  - **Boosting:** Sequentially correct mistakes of previous learners.

Lecture 13: Support Vector Machines (SVM)

- **Goal:** Find the widest margin (street) that separates two classes.
- **Decision Boundary:** $w^T x + b = 0$
- **Margin Width:** Inversely proportional to $\|w\|$.
- **Regularization Parameter C:**
  - $C > 1$: tight margin, less tolerant of errors $\Rightarrow$ overfitting risk.
  - $C < 1$: wider margin, more tolerant of errors $\Rightarrow$ generalization.
- **Soft Margin:** Allows some misclassifications to increase robustness.
- **Perceptron Learning:**
  - Update weights if point misclassified.
  - Adjust via: $w := w + \alpha y_i x_i$, $b := b + \alpha y_i$
- **Kernel Trick:**
  - Transform inputs to higher dimension implicitly using a kernel.
  - Examples: linear, polynomial, RBF kernels.

Lecture 14: Recommender Systems

- **Goal:** Recommend items to users using past data.
- **Challenges:** Scalability, cold start (new users/items), sparse data.
- **Approaches:**
  - **Neighborhood Methods:**
    * User-User similarity: recommend items liked by similar users.
    * Item-Item similarity: recommend items similar to those liked.

  - **Content-Based Filtering:**
    * Recommend based on item features (e.g., genre, keywords).
    * Use dot product of user-to-feature and feature-to-item matrices.
  - **Collaborative Filtering:**
    * Matrix factorization: $R_{ij} \approx P_i^T Q_j$
    * Alternating minimization of P and Q.

Lecture 15: Linear Regression

- **Goal:** Fit a linear model $y = X\beta + \epsilon$ to predict outputs.
- **Motivation:** Understand how y varies with x, identify trends, interpret relationships.
- **Assumptions:**
  - Linearity in parameters β.
  - Residuals ϵ are i.i.d. and normally distributed.

  s

- **Cost Function:** Sum of squared errors (SSE):

$$L(\beta) = \sum_{i=1}^{n} (y_i - X_i\beta)^2$$

- **Learning Methods:**
  - **Least Squares:** Minimize SSE to find $\hat{\beta}$.
  - **Maximum Likelihood Estimation:**
    * Assume $Y \sim \mathcal{N}(X\beta, \sigma^2 I)$
    * Maximize log-likelihood: $\log L(\beta) = -\frac{1}{2\sigma^2} \sum (y_i - X_i\beta)^2$
- **Overfitting:** Using overly complex models fits noise, not signal.