Complete Machine Learning Study Guide

Table of Contents

---

1. Mathematical Foundations

1.1 Linear Algebra Essentials

Vectors and Matrices

- **Vector: An ordered list of numbers (features in ML)**
- **Matrix: 2D array of numbers (dataset representation)**
- **Dot product: $a \cdot b = \Sigma(a_i \times b_i)$ - measures similarity**
- **Matrix multiplication: Foundation of neural networks**

Key Operations:

```
Vector dot product: [1,2,3] · [4,5,6] = 1×4 + 2×5 + 3×6 = 32
Matrix-vector multiplication: Used in linear regression y = Xw + b
```

1.2 Calculus for Machine Learning

Derivatives:

- **Partial derivatives: $\partial f/\partial x$ - how function changes with respect to one variable**
- **Chain rule: Used in backpropagation**
- **Gradient: Vector of partial derivatives pointing in steepest ascent direction**

Optimization:

- **Gradient descent: $w = w - \alpha\nabla f(w)$**
- **Critical for finding minimum of loss functions**

1.3 Probability and Statistics

Probability Distributions:

- **Normal distribution: Bell curve, many ML assumptions**
- **Bernoulli: Binary outcomes (classification)**
- **Multinomial: Multiple categories**

Key Concepts:

- **Expected value: $E[X] = \Sigma P(x) \times x$**
- **Variance: $Var(X) = E[(X-\mu)^2]$**
- **Bayes' theorem: $P(A|B) = P(B|A) \times P(A)/P(B)$**

---

2. Introduction to Machine Learning

2.1 What is Machine Learning?

Machine Learning is the scientific discipline that explores algorithms that can learn from data and make predictions without being explicitly programmed for each task.

Tom Mitchell's Definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

2.2 Types of Machine Learning

Supervised Learning

- **Input: Training data with labels (X, y)**
- **Goal: Learn function $f: X \rightarrow y$**
- **Types:**
  - **Classification: Predict discrete categories (spam/not spam)**
  - **Regression: Predict continuous values (house prices)**

Unsupervised Learning

- **Input: Data without labels (X only)**
- **Goal: Discover hidden patterns**

- **Types:**
  - **Clustering: Group similar data points**
  - **Dimensionality Reduction: Reduce feature space**
  - **Association Rules: Find relationships between variables**

Reinforcement Learning

- **Input: Environment interactions**
- **Goal: Learn optimal actions to maximize reward**
- **Applications: Game playing, robotics, autonomous vehicles**

2.3 ML Workflow

1. **Problem Definition: What are we trying to predict/discover?**
2. **Data Collection: Gather relevant data**
3. **Data Preprocessing: Clean, transform, prepare data**
4. **Feature Engineering: Select/create relevant features**
5. **Model Selection: Choose appropriate algorithm**
6. **Training: Fit model to data**
7. **Evaluation: Assess model performance**
8. **Hyperparameter Tuning: Optimize model parameters**
9. **Deployment: Put model into production**
10. **Monitoring: Track performance over time**

2.4 Bias-Variance Tradeoff

Bias: Error from oversimplifying assumptions

- **High bias = underfitting**
- **Model too simple to capture true relationship**

Variance: Error from sensitivity to small changes in training data

- **High variance = overfitting**
- **Model memorizes training data, doesn't generalize**

Total Error = Bias^2 + Variance + Irreducible Error

The Tradeoff:

- **Simple models: High bias, low variance**
- **Complex models: Low bias, high variance**

- **Goal: Find optimal balance**

---

3. Data Preprocessing and Feature Engineering

3.1 Data Quality Issues

Missing Data

- **Types:**
  - **MCAR (Missing Completely at Random)**
  - **MAR (Missing at Random)**
  - **MNAR (Missing Not at Random)**
- **Solutions:**
  - **Deletion: Remove rows/columns with missing values**
  - **Imputation: Fill with mean, median, mode, or predicted values**
  - **Advanced: Multiple imputation, KNN imputation**

Outliers

- **Detection:**
  - **Statistical methods: Z-score, IQR**
  - **Visual methods: Box plots, scatter plots**
  - **Model-based: Isolation Forest, Local Outlier Factor**
- **Treatment:**
  - **Remove outliers**
  - **Transform data (log transformation)**
  - **Use robust algorithms (less sensitive to outliers)**

Inconsistent Data

- **Standardize formats (dates, text case)**
- **Handle duplicates**
- **Resolve conflicting information**

3.2 Data Transformation

Scaling

```
python
```

```
# Standardization (Z-score normalization)
X_scaled = (X - mean) / std
# Result: mean=0, std=1

# Min-Max scaling
X_scaled = (X - min) / (max - min)
# Result: range [0,1]
```

Encoding Categorical Variables

- **One-Hot Encoding: Create binary columns for each category**
- **Label Encoding: Assign integers to categories**
- **Target Encoding: Replace with target mean for each category**

3.3 Feature Engineering

Feature Selection

- **Filter methods: Statistical tests (correlation, chi-square)**
- **Wrapper methods: Forward/backward selection**
- **Embedded methods: L1 regularization (Lasso)**

Feature Creation

- **Polynomial features: x, x^2, x^3, $x_1 \times x_2$**
- **Binning: Convert continuous to categorical**
- **Domain-specific features: Based on problem knowledge**

---

4. Supervised Learning - Regression

4.1 Linear Regression

Simple Linear Regression

Model: $y = \beta_0 + \beta_1 x + \varepsilon$

Assumptions:

1. **Linearity: Relationship between X and y is linear**
2. **Independence: Observations are independent**
3. **Homoscedasticity: Constant variance of errors**
4. **Normality: Errors are normally distributed**

Multiple Linear Regression

Model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \varepsilon$

Matrix form: $y = X\beta + \varepsilon$

Parameter Estimation

Normal Equation (Closed Form): $\beta = (X^T X)^{-1} X^T y$

Gradient Descent (Iterative):

> Cost function: $J(\beta) = (1/2m) \Sigma(h\beta(x^{(i)}) - y^{(i)})^2$
> Update rule: $\beta := \beta - \alpha \partial J/\partial \beta$

Model Evaluation

- **R^2 (Coefficient of Determination): $R^2 = 1 - SS_{res}/SS_{tot}$**
- **Mean Squared Error (MSE): $MSE = (1/n) \Sigma(y - \hat{y})^2$**
- **Root Mean Squared Error (RMSE): $\sqrt{MSE}$**
- **Mean Absolute Error (MAE): $(1/n) \Sigma|y - \hat{y}|$**

4.2 Regularization

Ridge Regression (L2)

Cost function: $J(\beta) = MSE + \lambda \Sigma\beta_j^2$

- **Shrinks coefficients toward zero**
- **Handles multicollinearity**
- **Keeps all features**

Lasso Regression (L1)

Cost function: $J(\beta) = MSE + \lambda \Sigma|\beta_j|$

- **Can shrink coefficients to exactly zero**
- **Performs automatic feature selection**
- **Sparse solutions**

Elastic Net

Cost function: $J(\beta) = MSE + \lambda_1 \Sigma|\beta_j| + \lambda_2 \Sigma\beta_j^2$

- **Combines L1 and L2 penalties**
- **Balances feature selection and multicollinearity handling**

4.3 Polynomial Regression

Transform features: $x \rightarrow [1, x, x^2, x^3, ...]$

Benefits: Can model non-linear relationships Risks: Overfitting with high-degree polynomials

4.4 Logistic Regression

Binary Logistic Regression

Sigmoid function: $\sigma(z) = 1/(1 + e^{-z})$

Model: $P(y=1|x) = \sigma(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p)$

Log-odds (logit): $\log(p/(1-p)) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$

Maximum Likelihood Estimation

Likelihood: $L(\beta) = \Pi P(y_i|x_i)^{y_i} (1-P(y_i|x_i))^{(1-y_i)}$

Log-likelihood: $\ell(\beta) = \Sigma[y_i \log(p_i) + (1-y_i) \log(1-p_i)]$

Multiclass Extensions

- **One-vs-Rest: Train binary classifiers for each class**
- **Multinomial Logistic: Direct multiclass extension using softmax**

---

5. Supervised Learning - Classification

5.1 Classification Fundamentals

Types of Classification

- **Binary Classification: Two classes (spam/not spam)**
- **Multiclass Classification: Multiple classes (iris species)**
- **Multilabel Classification: Multiple labels per instance**

Decision Boundaries

- **Linear classifiers: Straight lines/hyperplanes**
- **Non-linear classifiers: Curved boundaries**

5.2 Decision Trees

How Decision Trees Work

1. **Start with entire dataset**

2. **Find best split (feature + threshold)**
3. **Split data into subsets**
4. **Repeat recursively until stopping criterion**

Splitting Criteria

Information Gain (ID3):

- **Based on entropy reduction**
- **Entropy: $H(S) = -\Sigma\ p_i \log_2(p_i)$**
- **Information Gain: IG = H(parent) - Σ (|child|/|parent|) × H(child)**

Gini Impurity (CART):

- **Gini: $G = 1 - \Sigma\ p_i^2$**
- **Measures probability of misclassification**

Advantages and Disadvantages

Advantages:

- **Interpretable (white box)**
- **Handles both numerical and categorical features**
- **No need for feature scaling**
- **Automatic feature selection**

Disadvantages:

- **Prone to overfitting**
- **Unstable (small data changes → different tree)**
- **Biased toward features with many levels**

5.3 Ensemble Methods

Random Forest

Algorithm:

1. **Bootstrap sampling (bagging)**
2. **Train decision trees on each sample**
3. **Random feature selection at each split**
4. **Average predictions (regression) or vote (classification)**

Advantages:

- **Reduces overfitting**
- **Handles missing values**
- **Provides feature importance**
- **Works well out-of-the-box**

Gradient Boosting

Algorithm:

1. **Start with simple model**
2. **Train next model to predict residuals**
3. **Add to ensemble with small weight**
4. **Repeat until convergence**

AdaBoost (Adaptive Boosting):

- **Focuses on misclassified examples**
- **Increases weights of difficult examples**

XGBoost/LightGBM:

- **Optimized gradient boosting implementations**
- **Handle large datasets efficiently**

5.4 Support Vector Machines (From your notes)

Linear SVM

- **Find hyperplane that maximizes margin**
- **Support vectors: Points closest to decision boundary**
- **Optimization problem: Minimize $||w||^2/2$ subject to $y_i(w \cdot x_i + b) \geq 1$**

Soft Margin SVM

- **Allow some misclassifications with slack variables ξ_i**
- **Cost function: $(1/2)||w||^2 + C \Sigma\xi_i$**
- **C parameter balances margin size vs misclassifications**

Kernel Trick

- **Transform data to higher dimensions: $\varphi(x)$**
- **Kernel function: $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$**
- **Common kernels: Polynomial, RBF (Gaussian), Sigmoid**

5.5 k-Nearest Neighbors (From your notes)

Algorithm

1. Calculate distance to all training points
2. Find k nearest neighbors
3. Classify based on majority vote (or weighted vote)

Distance Metrics

- Euclidean: $\sqrt{\Sigma(x_i - y_i)^2}$
- Manhattan: $\Sigma|x_i - y_i|$
- Cosine: $\cos(\theta) = (a \cdot b)/(||a|| \, ||b||)$

Choosing k

- Small k: More sensitive to noise
- Large k: Smoother boundaries but may lose local patterns
- Rule of thumb: $k = \sqrt{n}$ (where n is number of samples)

5.6 Naive Bayes

Bayes' Theorem

$$P(\text{class}|\text{features}) = P(\text{features}|\text{class}) \times P(\text{class}) / P(\text{features})$$

Naive Assumption

Features are conditionally independent given the class:

$$P(x_1,x_2,...,x_n|\text{class}) = P(x_1|\text{class}) \times P(x_2|\text{class}) \times ... \times P(x_n|\text{class})$$

Types

- Gaussian NB: Features follow normal distribution
- Multinomial NB: For count data (text classification)
- Bernoulli NB: For binary features

---

6. Model Evaluation and Validation

6.1 Train/Validation/Test Split

Training Set (60%): Used to train the model Validation Set (20%): Used for hyperparameter tuning Test Set (20%): Final evaluation of model performance

6.2 Cross-Validation (From your notes)

k-Fold Cross-Validation

1. Split data into k equal parts
2. Train on k-1 folds, test on remaining fold
3. Repeat k times
4. Average the results

Leave-One-Out Cross-Validation

- Special case where k = n (number of samples)
- Computationally expensive but unbiased

Stratified Cross-Validation

- Maintains class distribution in each fold
- Important for imbalanced datasets

6.3 Classification Metrics (From your notes - excellent coverage)

Confusion Matrix

```
          Predicted
         Pos   Neg
Actual  Pos  TP    FN
     Neg  FP    TN
```

Key Metrics

- Accuracy: (TP + TN) / (TP + TN + FP + FN)
- Precision: TP / (TP + FP) - Of predicted positives, how many are correct?
- Recall (Sensitivity): TP / (TP + FN) - Of actual positives, how many did we catch?
- Specificity: TN / (TN + FP) - Of actual negatives, how many did we correctly identify?
- F1-Score: 2 × (Precision × Recall) / (Precision + Recall)

ROC and AUC (From your notes)

- ROC Curve: True Positive Rate vs False Positive Rate
- AUC: Area Under the ROC Curve
- AUC = 1: Perfect classifier
- AUC = 0.5: Random classifier

6.4 Regression Metrics

- **Mean Squared Error (MSE): Average of squared differences**
- **Root Mean Squared Error (RMSE): √MSE, same units as target**
- **Mean Absolute Error (MAE): Average of absolute differences**
- **R^2 Score: Proportion of variance explained by model**

6.5 Hyperparameter Tuning

Grid Search

- **Define parameter grid**
- **Try all combinations**
- **Computationally expensive but thorough**

Random Search

- **Random sampling from parameter distributions**
- **Often more efficient than grid search**

Bayesian Optimization

- **Uses probability to model the objective function**
- **More efficient for expensive evaluations**

---

7. Unsupervised Learning

7.1 Clustering

k-Means Clustering

Algorithm:

1. **Initialize k cluster centers randomly**
2. **Assign each point to nearest center**
3. **Update centers to mean of assigned points**
4. **Repeat until convergence**

Objective: Minimize Within-Cluster Sum of Squares (WCSS) $WCSS = \Sigma \Sigma ||x_i - \mu_j||^2$

Choosing k:

- **Elbow method: Plot WCSS vs k, look for "elbow"**
- **Silhouette analysis: Measure cluster separation**

Advantages:

- **Simple and fast**
- **Works well with spherical clusters**

Disadvantages:

- **Must specify k**
- **Sensitive to initialization**
- **Assumes spherical clusters of similar size**

Hierarchical Clustering

Agglomerative (Bottom-up):

1. **Start with each point as its own cluster**
2. **Merge closest clusters**
3. **Repeat until one cluster remains**

Linkage Criteria:

- **Single: Minimum distance between clusters**
- **Complete: Maximum distance between clusters**
- **Average: Average distance between clusters**
- **Ward: Minimize within-cluster variance**

Dendrogram: Tree diagram showing cluster hierarchy

DBSCAN (Density-Based)

Parameters:

- **ε (epsilon): Maximum distance between neighbors**
- **MinPts: Minimum points to form cluster**

Algorithm:

1. **For each point, find neighbors within ε**
2. **If neighbors ≥ MinPts, start new cluster**
3. **Recursively add neighbors to cluster**
4. **Points not in any cluster are noise**

Advantages:

- **Automatically determines number of clusters**
- **Can find arbitrarily shaped clusters**
- **Robust to outliers**

7.2 Dimensionality Reduction

Principal Component Analysis (PCA)

Goal: Find directions of maximum variance in data

Steps:

1. **Standardize data (zero mean, unit variance)**
2. **Compute covariance matrix**
3. **Find eigenvalues and eigenvectors**
4. **Sort by eigenvalue (largest first)**
5. **Select top k eigenvectors (principal components)**
6. **Transform data: X_new = X × PC**

Explained Variance Ratio: Proportion of total variance explained by each component

Applications:

- **Data compression**
- **Visualization (reduce to 2D/3D)**
- **Noise reduction**
- **Feature extraction**

t-SNE (t-Distributed Stochastic Neighbor Embedding)

Purpose: Nonlinear dimensionality reduction for visualization

Algorithm:

1. **Calculate pairwise similarities in high dimension**
2. **Map to low dimension (usually 2D)**
3. **Minimize divergence between similarity distributions**

Use Cases:

- **Visualizing high-dimensional data**
- **Cluster analysis**
- **Feature exploration**

UMAP (Uniform Manifold Approximation and Projection)

Advantages over t-SNE:

- **Preserves global structure better**
- **Faster computation**
- **More stable results**

7.3 Association Rules

Market Basket Analysis

Find relationships between items purchased together

Key Metrics:

- **Support: Frequency of itemset in transactions**
- **Confidence: P(B|A) - If A, then B**
- **Lift: Confidence / P(B) - Strength of association**

Apriori Algorithm

1. **Find frequent individual items**
2. **Generate candidate pairs**
3. **Count support for candidates**
4. **Repeat for larger itemsets**

---

8. Neural Networks and Deep Learning

8.1 Neural Network Fundamentals (From your notes)

Perceptron

Mathematical Model:

$$z = w_1x_1 + w_2x_2 + ... + w_nx_n + b$$
$$y = activation(z)$$

Activation Functions (From your notes)

- **Linear: f(x) = x**
- **Step: f(x) = 1 if x > 0, else 0**
- **Sigmoid: f(x) = 1/(1 + e^{-x})**

- **Tanh: $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$**
- **ReLU: $f(x) = \max(0, x)$**

8.2 Feedforward Networks

Architecture

- **Input Layer: Receives features**
- **Hidden Layers: Process information**
- **Output Layer: Produces predictions**

Universal Approximation Theorem

A feedforward network with a single hidden layer can approximate any continuous function, given enough neurons.

8.3 Backpropagation (From your notes - excellent coverage)

The algorithm for training neural networks by computing gradients and updating weights.

Key Steps:

1. **Forward pass: Compute predictions**
2. **Calculate loss**
3. **Backward pass: Compute gradients**
4. **Update weights: $w := w - \alpha \nabla w$**

8.4 Deep Learning Regularization

Dropout

Randomly set some neurons to zero during training

- **Prevents overfitting**
- **Forces network to learn robust features**

Batch Normalization

Normalize inputs to each layer

- **Accelerates training**
- **Reduces internal covariate shift**

Early Stopping

Stop training when validation performance stops improving

8.5 Convolutional Neural Networks (From your notes - comprehensive)

Convolution Operation

- **Filters/Kernels: Learn local patterns**
- **Feature Maps: Output of convolution**
- **Parameter Sharing: Same weights used across image**
- **Local Connectivity: Each neuron connects to local region**

CNN Architecture

1. **Convolutional Layers: Feature extraction**
2. **Pooling Layers: Downsampling**
3. **Fully Connected Layers: Classification**

8.6 Transfer Learning (From your notes)

Using pre-trained models and adapting them to new tasks:

1. **Load pre-trained model**
2. **Freeze early layers**
3. **Add custom classifier**
4. **Fine-tune on new data**

---

9. Advanced Topics

9.1 Time Series Analysis

Components

- **Trend: Long-term direction**
- **Seasonality: Regular patterns**
- **Cyclical: Irregular patterns**
- **Noise: Random fluctuations**

Methods

- **ARIMA: Autoregressive Integrated Moving Average**
- **Exponential Smoothing: Weighted averages**
- **Neural Networks: RNNs, LSTMs for sequence modeling**

9.2 Natural Language Processing Basics

Text Preprocessing

1. **Tokenization: Split text into words**
2. **Lowercasing: Normalize case**
3. **Remove stopwords: Filter common words**
4. **Stemming/Lemmatization: Reduce to root forms**

Feature Extraction

- **Bag of Words: Count word frequencies**
- **TF-IDF: Term frequency × Inverse document frequency**
- **Word Embeddings: Dense vector representations**

9.3 Recommender Systems

Collaborative Filtering

- **User-based: Find similar users**
- **Item-based: Find similar items**
- **Matrix Factorization: Decompose user-item matrix**

Content-Based Filtering

Recommend based on item features and user preferences

Hybrid Systems

Combine collaborative and content-based approaches

9.4 Model Interpretability

Feature Importance

- **Permutation Importance: Measure performance drop when feature is shuffled**
- **Tree-based Importance: From decision trees/random forests**

SHAP (SHapley Additive exPlanations)

Unified framework for explaining individual predictions

LIME (Local Interpretable Model-agnostic Explanations)

Explain individual predictions by learning local surrogate models

---

10. Practical Implementation

10.1 Scikit-learn (From your notes - good coverage)

Workflow

python

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Preprocess
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train model
model = RandomForestClassifier()
model.fit(X_train_scaled, y_train)

# Evaluate
predictions = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, predictions)
```

10.2 Deep Learning Frameworks

TensorFlow/Keras

python

```python
```

```python
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Dense(64, activation='relu', input_shape=(input_dim,)),
    layers.Dropout(0.5),
    layers.Dense(32, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100, validation_split=0.2)
```

PyTorch

```python
import torch
import torch.nn as nn

class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

10.3 MLOps Basics

Model Deployment

- **REST APIs: Flask, FastAPI**
- **Cloud Platforms: AWS SageMaker, Google Cloud AI Platform**
- **Containerization: Docker, Kubernetes**

Model Monitoring

- **Performance Monitoring: Track accuracy, latency**
- **Data Drift: Monitor input data changes**
- **Model Drift: Monitor prediction quality over time**

Version Control

- **Code: Git**
- **Data: DVC (Data Version Control)**
- **Models: MLflow, Weights & Biases**

Study Tips and Best Practices

1. Learning Sequence

Follow this order for optimal understanding:

1. **Mathematical foundations → Data preprocessing → Linear models**
2. **Classification algorithms → Evaluation metrics → Ensemble methods**
3. **Unsupervised learning → Neural networks → Advanced topics**

2. Hands-on Practice

- **Implement algorithms from scratch to understand mechanics**
- **Use real datasets from Kaggle, UCI ML Repository**
- **Start with simple datasets, gradually increase complexity**

3. Common Pitfalls

- **Data leakage: Future information in training data**
- **Overfitting: Model memorizes training data**
- **Improper validation: Using test set for model selection**
- **Ignoring domain knowledge: Not leveraging expert insights**

4. Project Ideas

- **Beginner: Iris classification, Boston housing prices**
- **Intermediate: Movie recommendation, sentiment analysis**
- **Advanced: Computer vision projects, time series forecasting**

Remember: Understanding the intuition behind algorithms is more important than memorizing formulas. Focus on when and why to use each technique, not just how they work mathematically.