Complete Machine Learning Algorithms & Evaluation Cheat Sheet

Table of Contents

---

1. Supervised Learning Algorithms {#supervised-learning}

1.1 Linear Regression

Purpose: Predict continuous values by fitting a linear relationship between features and target.

Mathematics:

- Model: $y = \beta_0 + \beta_1x_1 + \beta_2x_2 + ... + \beta_nx_n + \varepsilon$
- Loss Function: $MSE = (1/n)\Sigma(y_i - \hat{y}_i)^2$
- Solution: $\beta = (X'X)^{-1}X'y$ (Normal Equation)

Variants:

- **Ridge Regression (L2)**: Adds $\lambda\Sigma\beta_i^2$ penalty to prevent overfitting
- **Lasso Regression (L1)**: Adds $\lambda\Sigma|\beta_i|$ penalty for feature selection
- **Elastic Net**: Combines L1 and L2 penalties

When to Use:

- Linear relationships between features and target
- Interpretability is important
- Baseline model for regression tasks

1.2 Logistic Regression

Purpose: Binary/multiclass classification using sigmoid/softmax function.

Mathematics:

- Binary: $P(y=1|x) = 1/(1 + e^{(-\beta'x)})$
- Loss: Cross-entropy = $-\Sigma[y \log(p) + (1-y)\log(1-p)]$
- Optimization: Gradient Descent or Newton-Raphson

Key Concepts:

- Odds Ratio: $p/(1-p)$
- Log Odds: $\log(p/(1-p)) = \beta'x$
- Decision Boundary: Linear in feature space

Extensions:

- Multinomial Logistic Regression (Softmax)
- Ordinal Logistic Regression

1.3 Decision Trees

Purpose: Non-linear classification/regression through recursive partitioning.

Algorithms:

- **CART (Classification and Regression Trees)**
  - Gini Impurity: $G = 1 - \Sigma p_i^2$
  - Entropy: $H = -\Sigma p_i \log_2(p_i)$
  - MSE for regression
- **ID3/C4.5**: Uses Information Gain
  - $IG(S,A) = H(S) - \Sigma(|S_v|/|S|)H(S_v)$

Pruning Methods:

- Pre-pruning: Max depth, min samples split
- Post-pruning: Cost complexity pruning

Advantages: Interpretable, handles non-linearity, no scaling needed **Disadvantages**: Overfitting, unstable, biased to high-cardinality features

1.4 Support Vector Machines (SVM)

Purpose: Find optimal hyperplane maximizing margin between classes.

Mathematics:

- Linear SVM: $\min(1/2)||w||^2$ subject to $y_i(w'x_i + b) \geq 1$
- Soft Margin: Adds slack variables ξ_i with penalty C
- Dual Form: Uses Lagrange multipliers α_i

Kernel Trick:

- Linear: $K(x_i,x_j) = x_i'x_j$
- Polynomial: $K(x_i,x_j) = (\gamma x_i'x_j + r)^d$
- RBF: $K(x_i,x_j) = \exp(-\gamma||x_i-x_j||^2)$
- Sigmoid: $K(x_i,x_j) = \tanh(\gamma x_i'x_j + r)$

Key Parameters:

- C: Regularization (trade-off between margin and misclassification)
- γ (gamma): Kernel coefficient
- kernel: Type of kernel function

1.5 k-Nearest Neighbors (k-NN)

Purpose: Instance-based learning using proximity to training examples.

Algorithm:

1. Calculate distance to all training points
2. Select k nearest neighbors
3. Vote (classification) or average (regression)

Distance Metrics:

- Euclidean: $d = \sqrt{\Sigma(x_i-y_i)^2}$
- Manhattan: $d = \Sigma|x_i-y_i|$
- Minkowski: $d = (\Sigma|x_i-y_i|^p)^{(1/p)}$
- Cosine: similarity = $(x\cdot y)/(||x||||y||)$

Considerations:

- k selection: Cross-validation, odd k for binary classification
- Curse of dimensionality
- Computational cost: O(nd) for each prediction
- Feature scaling crucial

1.6 Naive Bayes

Purpose: Probabilistic classifier based on Bayes' theorem with independence assumption.

Mathematics:

- Bayes' Theorem: $P(y|X) = P(X|y)P(y)/P(X)$
- Naive Assumption: $P(X|y) = \prod P(x_i|y)$

Variants:

- **Gaussian NB**: $P(x_i|y) \sim N(\mu_y, \sigma_y^2)$
- **Multinomial NB**: For discrete features (text classification)
- **Bernoulli NB**: For binary features

Laplace Smoothing: Add α to avoid zero probabilities

---

2. Unsupervised Learning Algorithms {#unsupervised-learning}

2.1 K-Means Clustering

Purpose: Partition data into k clusters minimizing within-cluster variance.

Algorithm:

1. Initialize k centroids randomly
2. Assign points to nearest centroid
3. Update centroids as cluster means
4. Repeat until convergence

Objective Function: $J = \Sigma\Sigma||x_i - \mu_j||^2$ (inertia)

Initialization Methods:

- Random
- K-means++: Probabilistic selection for better initialization
- Multiple runs with different seeds

Limitations:

- Assumes spherical clusters
- Sensitive to outliers
- Must specify k
- Local minima issues

2.2 Hierarchical Clustering

Purpose: Build hierarchy of clusters using agglomerative or divisive approach.

Linkage Criteria:

- Single: min distance between clusters
- Complete: max distance between clusters
- Average: mean distance between all pairs
- Ward: minimize within-cluster variance

Distance Metrics: Euclidean, Manhattan, Cosine

Output: Dendrogram for visualizing hierarchy

2.3 DBSCAN (Density-Based Spatial Clustering)

Purpose: Find arbitrary-shaped clusters based on density.

Parameters:

- ε (eps): Maximum distance between points in neighborhood
- MinPts: Minimum points to form dense region

Point Types:

- Core points: Have ≥ MinPts within ε
- Border points: Within ε of core point
- Noise points: Neither core nor border

Advantages: Finds arbitrary shapes, robust to outliers, no k needed

2.4 Gaussian Mixture Models (GMM)

Purpose: Probabilistic model assuming data from mixture of Gaussians.

Mathematics:

- Model: $p(x) = \Sigma \pi_k \, N(x|\mu_k, \Sigma_k)$
- EM Algorithm:
  - E-step: Calculate responsibilities
  - M-step: Update parameters (π, μ, Σ)

Covariance Types:

- Full: Each component has own covariance matrix

- Diagonal: Diagonal covariance matrices
- Spherical: $\Sigma = \sigma^2 I$
- Tied: All components share same covariance

2.5 Principal Component Analysis (PCA)

Purpose: Linear dimensionality reduction maximizing variance.

Algorithm:

1. Standardize data
2. Compute covariance matrix
3. Calculate eigenvectors/eigenvalues
4. Select top k components
5. Transform data

Mathematics:

- Covariance: $C = (1/n)X'X$
- Components: Principal axes of variation
- Explained Variance Ratio: $\lambda_i/\Sigma\lambda_j$

Variants:

- Kernel PCA: Non-linear using kernel trick
- Sparse PCA: L1 penalty for interpretability
- Incremental PCA: For large datasets

2.6 t-SNE (t-Distributed Stochastic Neighbor Embedding)

Purpose: Non-linear dimensionality reduction for visualization.

Key Concepts:

- Preserves local structure
- Uses Student's t-distribution in low dimensions
- Perplexity parameter: Balance between local/global aspects

Limitations:

- Computationally expensive $O(n^2)$
- Non-deterministic
- Cannot project new data

- Mainly for visualization (2D/3D)

2.7 Autoencoders

Purpose: Neural network for unsupervised representation learning.

Architecture:

- Encoder: Compresses input to latent representation
- Decoder: Reconstructs input from latent space
- Loss: Reconstruction error (MSE, Cross-entropy)

Variants:

- Denoising Autoencoders
- Sparse Autoencoders
- Variational Autoencoders (VAE)
- Contractive Autoencoders

---

3. Semi-Supervised Learning {#semi-supervised-learning}

3.1 Self-Training

Algorithm:

1. Train on labeled data
2. Predict on unlabeled data
3. Add confident predictions to training set
4. Retrain and repeat

3.2 Co-Training

Requirements: Two independent feature sets **Process**: Train two classifiers, each labels data for the other

3.3 Graph-Based Methods

Approach: Propagate labels through graph of data points **Examples**: Label Propagation, Label Spreading

---

4. Reinforcement Learning {#reinforcement-learning}

4.1 Core Concepts

- **Agent**: Decision maker
- **Environment**: External system
- **State (s)**: Current situation

- **Action (a)**: Agent's decision
- **Reward (r)**: Feedback signal
- **Policy (π)**: Mapping states to actions
- **Value Function**: Expected future reward

4.2 Q-Learning

Update Rule: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$

4.3 Deep Q-Networks (DQN)

Innovations:

- Neural network approximates Q-function
- Experience replay buffer
- Target network for stability

4.4 Policy Gradient Methods

Objective: Maximize expected reward **Examples**: REINFORCE, Actor-Critic, PPO, TRPO

---

5. Ensemble Methods {#ensemble-methods}

5.1 Bagging (Bootstrap Aggregating)

Purpose: Reduce variance through averaging.

Algorithm:

1. Create bootstrap samples
2. Train model on each sample
3. Average predictions (regression) or vote (classification)

Example: Random Forest

- Multiple decision trees
- Random feature selection at each split
- Out-of-Bag (OOB) error estimation

5.2 Boosting

Purpose: Reduce bias by combining weak learners sequentially.

AdaBoost:

- Weight misclassified samples higher

- Combine weighted weak learners
- Exponential loss function

Gradient Boosting:

- Fit new model to residuals
- Additive model: $F(x) = \Sigma f_m(x)$
- Can use any differentiable loss

XGBoost Improvements:

- Regularization terms
- Second-order approximation
- Column subsampling
- Parallel processing

LightGBM Features:

- Leaf-wise growth
- Gradient-based One-Side Sampling
- Exclusive Feature Bundling

5.3 Stacking (Stacked Generalization)

Process:

1. Train base models on training data
2. Use predictions as features for meta-model
3. Meta-model learns optimal combination

Key Considerations:

- Use cross-validation to generate base predictions
- Diverse base models improve performance

---

6. Deep Learning Algorithms {#deep-learning}

6.1 Feedforward Neural Networks

Components:

- Input layer
- Hidden layers with activation functions
- Output layer

- Backpropagation for training

Activation Functions:

- ReLU: f(x) = max(0,x)
- Sigmoid: f(x) = 1/(1+e^(-x))
- Tanh: f(x) = (e^x - e^(-x))/(e^x + e^(-x))
- Leaky ReLU: f(x) = max(αx,x)
- GELU, Swish, Mish (modern variants)

6.2 Convolutional Neural Networks (CNN)

Key Layers:

- Convolutional: Feature detection with filters
- Pooling: Downsampling (Max, Average)
- Fully Connected: Classification/regression

Important Concepts:

- Stride, Padding, Receptive Field
- Transfer Learning
- Data Augmentation

Architectures: LeNet, AlexNet, VGG, ResNet, Inception, EfficientNet

6.3 Recurrent Neural Networks (RNN)

Purpose: Sequential data processing with memory.

Variants:

- **Vanilla RNN**: $h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$
- **LSTM**: Gates control information flow (forget, input, output)
- **GRU**: Simplified LSTM with reset and update gates

Issues: Vanishing/exploding gradients, long-term dependencies

6.4 Transformer Architecture

Key Innovation: Self-attention mechanism

Components:

- Multi-head attention

- Positional encoding
- Feed-forward networks
- Layer normalization

Applications: BERT, GPT, T5, Vision Transformer

6.5 Generative Models

Generative Adversarial Networks (GANs):

- Generator: Creates fake samples
- Discriminator: Distinguishes real/fake
- Minimax game optimization

Variational Autoencoders (VAE):

- Encoder: Maps to latent distribution
- Decoder: Generates from latent space
- KL divergence regularization

Diffusion Models:

- Forward process: Add noise gradually
- Reverse process: Denoise to generate

---

7. Evaluation Metrics {#evaluation-metrics}

7.1 Classification Metrics

Confusion Matrix Components:

- True Positives (TP), True Negatives (TN)
- False Positives (FP), False Negatives (FN)

Basic Metrics:

- **Accuracy**: (TP + TN) / Total
- **Precision**: TP / (TP + FP)
- **Recall (Sensitivity)**: TP / (TP + FN)
- **Specificity**: TN / (TN + FP)
- **F1 Score**: 2 × (Precision × Recall) / (Precision + Recall)

Advanced Metrics:

- **F-beta Score**: Weighted F1 score
- **Matthews Correlation Coefficient (MCC)**: Balanced measure for imbalanced data
- **Cohen's Kappa**: Agreement correcting for chance
- **ROC-AUC**: Area under ROC curve (TPR vs FPR)
- **PR-AUC**: Area under Precision-Recall curve

Multi-class Metrics:

- Micro-averaging: Aggregate contributions
- Macro-averaging: Average per-class metrics
- Weighted-averaging: Class-frequency weighted

7.2 Regression Metrics

Error-based Metrics:

- **Mean Absolute Error (MAE)**: $(1/n)\Sigma|y_i - \hat{y}_i|$
- **Mean Squared Error (MSE)**: $(1/n)\Sigma(y_i - \hat{y}_i)^2$
- **Root Mean Squared Error (RMSE)**: $\sqrt{MSE}$
- **Mean Absolute Percentage Error (MAPE)**: $(100/n)\Sigma|y_i - \hat{y}_i|/|y_i|$

Correlation-based Metrics:

- **R^2 (Coefficient of Determination)**: 1 - (SS_res/SS_tot)
- **Adjusted R^2**: Penalizes for number of features
- **Pearson Correlation**: Linear correlation coefficient

7.3 Clustering Metrics

Internal Metrics (no ground truth):

- **Silhouette Score**: Cohesion vs separation (-1 to 1)
- **Davies-Bouldin Index**: Ratio of within-cluster to between-cluster distance
- **Calinski-Harabasz Index**: Ratio of between-group to within-group dispersion

External Metrics (with ground truth):

- **Adjusted Rand Index (ARI)**: Chance-corrected Rand index
- **Normalized Mutual Information (NMI)**: Information-theoretic measure
- **Fowlkes-Mallows Score**: Geometric mean of precision and recall

7.4 Ranking Metrics

- **Mean Average Precision (MAP)**
- **Normalized Discounted Cumulative Gain (NDCG)**
- **Mean Reciprocal Rank (MRR)**

---

8. Model Selection & Validation {#model-selection}

8.1 Train-Test Split

Basic Split:

- Training: 60-80%
- Test: 20-40%
- Stratified split for imbalanced data

8.2 Cross-Validation Techniques

k-Fold Cross-Validation:

- Split data into k folds
- Train on k-1, validate on 1
- Repeat k times, average results

Variations:

- **Stratified k-Fold**: Preserves class distribution
- **Group k-Fold**: Groups stay together
- **Time Series Split**: Respects temporal order
- **Leave-One-Out (LOO)**: k = n samples

8.3 Hyperparameter Tuning

Grid Search:

- Exhaustive search over parameter grid
- Computationally expensive but thorough

Random Search:

- Sample random combinations
- Often more efficient than grid search

Bayesian Optimization:

- Build probabilistic model of objective

- Intelligently select next parameters
- Examples: Gaussian Processes, Tree-structured Parzen Estimators

Advanced Methods:

- Genetic Algorithms
- Hyperband
- Population Based Training

8.4 Model Selection Criteria

Information Criteria:

- **AIC (Akaike)**: -2ln(L) + 2k
- **BIC (Bayesian)**: -2ln(L) + k×ln(n)
- Lower is better, penalizes complexity

Bias-Variance Tradeoff:

- High Bias: Underfitting, too simple
- High Variance: Overfitting, too complex
- Goal: Balance both

---

9. Feature Engineering & Selection {#feature-engineering}

9.1 Feature Creation

Numerical Features:

- Polynomial features
- Interaction terms
- Binning/Discretization
- Log/exponential transforms
- Domain-specific calculations

Categorical Features:

- One-hot encoding
- Label encoding
- Target encoding
- Frequency encoding
- Binary encoding

Text Features:

- Bag of Words (BoW)
- TF-IDF
- Word embeddings (Word2Vec, GloVe)
- N-grams
- Topic modeling (LDA)

Time Features:

- Lag features
- Rolling statistics
- Seasonal decomposition
- Fourier features

9.2 Feature Selection Methods

Filter Methods:

- Correlation coefficient
- Chi-square test
- Mutual information
- ANOVA F-statistic

Wrapper Methods:

- Forward selection
- Backward elimination
- Recursive Feature Elimination (RFE)

Embedded Methods:

- L1 regularization (Lasso)
- Tree-based feature importance
- Permutation importance

9.3 Feature Scaling

Standardization: $z = (x - \mu) / \sigma$ **Min-Max Normalization**: $x' = (x - min) / (max - min)$ **Robust Scaling**: Use median and IQR **Unit Vector Scaling**: $x' = x / ||x||$

---

10. Optimization Algorithms {#optimization}

10.1 Gradient Descent Variants

Batch Gradient Descent:

- Uses entire dataset
- Stable but slow convergence

Stochastic Gradient Descent (SGD):

- Updates per sample
- Noisy but faster

Mini-batch Gradient Descent:

- Balance between batch and SGD
- Most commonly used

10.2 Advanced Optimizers

Momentum:

- Accelerates in consistent directions
- $v = \beta v + (1-\beta)\nabla f$
- $\theta = \theta - \alpha v$

AdaGrad:

- Adaptive learning rates per parameter
- Accumulates squared gradients

RMSprop:

- Fixes AdaGrad's diminishing learning rates
- Exponential moving average of squared gradients

Adam (Adaptive Moment Estimation):

- Combines momentum and RMSprop
- Bias correction for initial steps
- Most popular optimizer

Variants:

- AdamW: Decoupled weight decay
- NAdam: Nesterov momentum in Adam
- RAdam: Rectified Adam

- LAMB: Layer-wise adaptive moments

10.3 Learning Rate Scheduling

Step Decay: Reduce by factor every k epochs **Exponential Decay**: lr = lr_0 × e^(-kt) **Cosine Annealing**: Cyclical learning rates **ReduceLROnPlateau**: Reduce when metric plateaus **Warm Restarts**: Reset to initial LR periodically

---

Best Practices Summary

Model Selection Guidelines

1. **Linear Models**: When relationships are approximately linear, interpretability needed
2. **Tree-based**: Non-linear patterns, mixed data types, feature importance needed
3. **SVM**: High-dimensional data, clear margin of separation
4. **Neural Networks**: Complex patterns, large datasets, unstructured data
5. **Ensemble Methods**: When single models insufficient, competition settings

Common Pitfalls to Avoid

1. Data leakage between train/test sets
2. Not handling class imbalance
3. Ignoring multicollinearity
4. Overfitting to validation set
5. Not considering computational constraints
6. Ignoring temporal aspects in time series
7. Using inappropriate metrics

Performance Optimization Tips

1. Start simple, increase complexity gradually
2. Ensure proper data preprocessing
3. Use cross-validation for reliable estimates
4. Monitor both training and validation metrics
5. Implement early stopping for iterative methods
6. Consider ensemble methods for final boost
7. Profile code for computational bottlenecks

When to Stop Improving

1. Validation performance plateaus

2. Marginal gains not worth complexity
3. Computational budget exceeded
4. Business requirements met
5. Risk of overfitting to test set