Machine Learning Basics Cheat Sheet

Codeium

2025-02-15

Machine Learning Basics Cheat Sheet

1. Types of Machine Learning

Supervised Learning

- Input → Output mappings using labeled data
- Types:
  - Classification (discrete output)
  - Regression (continuous output)
- Examples:

```
Classification:     Regression:
Email → Spam/Ham   House size → Price
Image → Cat/Dog    Age/BMI → Blood Pressure
```

Unsupervised Learning

- Finds patterns in unlabeled data
- Types:
  - Clustering
  - Dimensionality Reduction
  - Association
- Examples:

```
Clustering:        Association:
Customers → Groups  Shopping cart → Related items
Pixels → Segments   Netflix → Movie recommendations
```

Reinforcement Learning

- Agent learns through environment interaction
- Components:

```
State (S) → Action (A) → Reward (R)
Example: Chess game
S: Board position
A: Move piece
R: Win/Loss/Draw
```

2. Core Concepts

Model Evaluation Metrics

Classification Metrics

```
Confusion Matrix:
              Predicted
Actual    Positive  Negative
Positive    TP        FN
Negative    FP        TN

Accuracy = (TP + TN)/(TP + TN + FP + FN)
Precision = TP/(TP + FP)
Recall = TP/(TP + FN)
F1 Score = 2 × (Precision × Recall)/(Precision + Recall)
```

Regression Metrics

```
MSE = (1/n) × Σ(y_true - y_pred)²
RMSE = √MSE
MAE = (1/n) × Σ|y_true - y_pred|
R² = 1 - (MSE of model)/(MSE of baseline)
```

Cross-Validation

```
K-Fold (k=5):
Data: [1][2][3][4][5]
Fold 1: Test[1] Train[2,3,4,5]
Fold 2: Test[2] Train[1,3,4,5]
Fold 3: Test[3] Train[1,2,4,5]
Fold 4: Test[4] Train[1,2,3,5]
Fold 5: Test[5] Train[1,2,3,4]
```

3. Common Algorithms

Linear Regression

```
y = mx + b
Cost Function (MSE):
J = (1/n) × Σ(y_i - (mx_i + b))²

Gradient Descent Update:
m = m -   ×  J/ m
b = b -   ×  J/ b
```

Logistic Regression

```
P(y=1) = 1/(1 + e^(-z))
where z = wx + b

Cost Function:
J = -(1/n) × Σ(y_i × log(p_i) + (1-y_i) × log(1-p_i))
```

Decision Trees

```
Root
```

```
   Feature 1 < threshold
       Class A
       Feature 2 < threshold
           Class B
           Class C
   Class D

Splitting Criteria:
- Gini Impurity
- Information Gain
- Entropy
```

k-Nearest Neighbors (kNN)

```
For new point P:
1. Calculate distances to all training points
2. Find k closest points
3. Classification: majority vote
   Regression: average of k points

Distance Metrics:
- Euclidean: √Σ(x_i - y_i)²
- Manhattan: Σ|x_i - y_i|
```

Support Vector Machines (SVM)

```
Objective: Find hyperplane with maximum margin

Linear SVM:
w·x + b = 0 (hyperplane)
w·x + b   1 (positive class)
w·x + b   -1 (negative class)

Kernel Trick:
- Linear: K(x,y) = x·y
- RBF: K(x,y) = exp(- ||x-y||²)
```

4. Feature Engineering

Scaling Methods

```
Min-Max Scaling:
x_scaled = (x - x_min)/(x_max - x_min)

Standard Scaling:
x_scaled = (x -  )/

Robust Scaling:
x_scaled = (x - median)/(Q3 - Q1)
```

Encoding Categorical Variables

```
Label Encoding:
[red, blue, red] → [0, 1, 0]
```

```
One-Hot Encoding:
red  → [1, 0, 0]
blue → [0, 1, 0]
green→ [0, 0, 1]
```

Feature Selection

1. Filter Methods:
    - Correlation
    - Chi-square test
    - ANOVA
2. Wrapper Methods:
    - Forward Selection
    - Backward Elimination
    - Recursive Feature Elimination
3. Embedded Methods:
    - LASSO
    - Ridge
    - Elastic Net

5. Hyperparameter Tuning

Grid Search

```
params = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 15]
}
Total combinations: 3 × 3 = 9
```

Random Search

```
params = {
    'n_estimators': range(100, 300),
    'max_depth': range(5, 15)
}
Sample n random combinations
```

6. Common Problems & Solutions

Overfitting

Solutions: 1. More training data 2. Regularization (L1, L2) 3. Dropout 4. Early stopping 5. Cross-validation

Underfitting

Solutions: 1. More complex model 2. Better features 3. Reduce regularization 4. Train longer

Class Imbalance

Solutions: 1. Oversampling (SMOTE) 2. Undersampling 3. Class weights 4. Ensemble methods

7. Best Practices

Data Preprocessing

1. Handle missing values

2. Remove duplicates
3. Scale features
4. Handle outliers
5. Balance classes

Model Selection

1. Start simple
2. Use cross-validation
3. Consider computational cost
4. Check assumptions
5. Validate on holdout set

Model Deployment

1. Save model artifacts
2. Version control
3. Monitor performance
4. Set up pipelines
5. Plan for updates

8. Learning Curves

```
Overfitting:
Training Error: ↓↓↓
Validation Error: ↓↑↑

Underfitting:
Training Error: ↓→
Validation Error: ↓→
(Both high)

Good Fit:
Training Error: ↓→
Validation Error: ↓→
(Both low and close)
```

Remember: - Start with simple models - Use cross-validation - Feature engineering is crucial - Monitor both training and validation metrics - Consider computational resources - Document your process