Traditional Machine Learning

Theory and Implementation

UNIVERSITY OF CALGARY

Outline

- Learning Goals
- Traditional ML Concepts
- Popular ML algorithms
- Model Evaluation
- ML best practices
- Summary

UNIVERSITY OF CALGARY

Learning Goals

- **Understand** how machines learn patterns from data
- **Learn** the foundations of key ML algorithms
- **Apply** traditional ML methods to real-world problems
- **Evaluate** model performance using appropriate metrics
- **Recognize** when to use traditional ML vs. deep learning

What is Machine Learning?

> *"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed."*
>
> — Arthur Samuel (1959)

- **Modern Perspective**
  - ML algorithms learn patterns from data to make predictions or decisions
  - Focus on building models that improve performance through experience
  - Bridges statistics, computer science, and domain expertise
  - Examples: Email spam detection, Medical diagnosis assistance

UNIVERSITY OF CALGARY

Machine Learning Paradigms

Artificial Intelligence

Machine Learning

Supervised Learning

- Learns from labeled data
- Has target/output variable
- Types:
  - Regression(continuous)
  - Classification (discrete)

Unsupervised Learning

- Learns from unlabeled data
- Finds hidden patterns
- Types:
  - Clustering
  - Dimensionality reduction

Reinforcement Learning

- Learn through feedback
- No predefined data learn from environment
- Examples:
  - Game AI
  - Robotics

Deep Learning

Machine Learning Paradigms

Machine Learning

Supervised Learning

Regression

Predicting a continuous numerical value

Examples

- Linear Regression
- Polynomial Regression
- Ridge/Lasso Regression
- Decision Tree Regressor
- Random Forest Regressor
- Support Vector Regression (SVR)

Classification

Predicting a discrete category/class

Examples

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- Naive Bayes

Unsupervised Learning

Clustering

Grouping similar data points

Examples

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- Naive Bayes

Dimensionality Reduction

Reducing number of features

Examples

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- Naive Bayes

Traditional ML vs Deep Learning

Traditional ML

- Requires feature engineering
- Works well with small datasets
- More interpretable
- Faster training
- Less computational resources
- Examples: Linear Regression, SVM, Random Forest

Deep Learning

- Automatic feature learning
- Requires large datasets
- Less interpretable
- Longer training time
- GPU-intensive
- Examples: CNN, RNN, Transformers

Machine Learning Workflow

1 Problem Definition

- Define task type
- Set success metrics
- Business goals

2 Data Collection

- Gather data
- Explore patterns
- Check quality

3 Preprocessing

- Clean data
- Handle missing values
- Normalize features

4 Feature Engineering

- Create features
- Feature selection
- Encoding

5 Model Training

- Select algorithm
- Train/test split
- Fit model

6 Evaluation

- Test performance
- Tune parameters
- Iterate

Feature Engineering

The art of transforming raw data into meaningful features

Data Preprocessing

- Handle missing values
  - Removal, mean/median/mode imputation
- Handle outliers
  - Removal, capping, transformation
- Feature scaling
  - Normalization, standardization
- Splitting data
  - Train/Test/Validation

Feature Transformation

- Encode categorical
  - Label encoding, one-hot encoding
- Polynomial features
- Log, square root, or power transform
- Interaction features
  - Combining two or more features
- Dimensionality reduction
  - PCA, LDA

Feature Selection

- Filter methods
- Recursive Feature Elimination (RFE)
- Removing low-variance features
- Regularization methods
  - Lasso, Ridge
- Feature importance from models

💡 **Good features** often matter more than the algorithm!

Train/Test/Validation Split

- **Why Split Data?**
  - To evaluate how well our model generalizes to unseen data

- **Common Split Strategy:**
  - **Training Set (60-70%):** Learn patterns from data
  - **Validation Set (15-20%):** Tune hyperparameters
  - **Test Set (15-20%):** Final unbiased evaluation

⚠ **Critical Rule: NEVER train on test data!**

Linear Regression

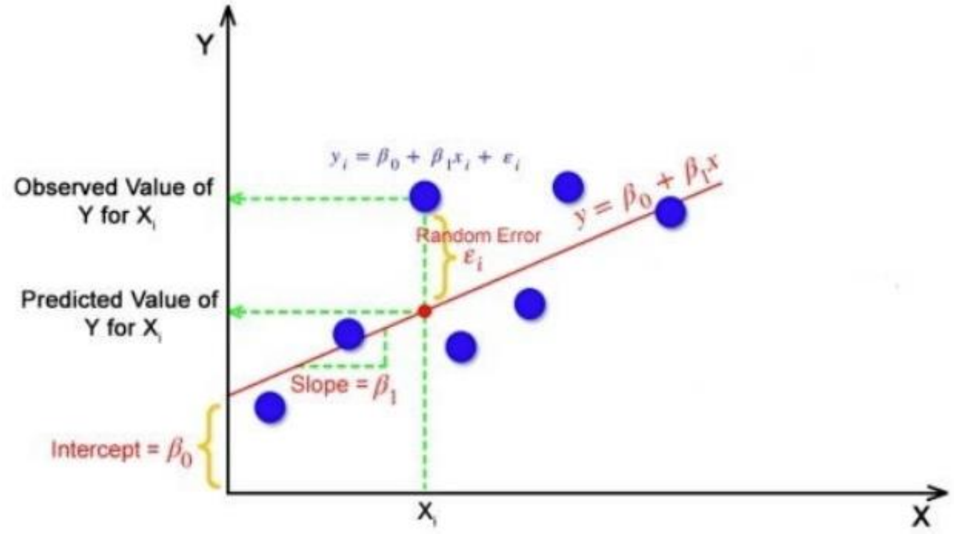
- Goal: Models linear relationship between features and target

Model Formulation

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Objective: Minimize Sum of Squared Errors

$$\text{Loss}(\beta) = \Sigma(y_i - \hat{y}_i)^2$$



✓ Pros: Simple, low cost, fast, interpretable

✗ Cons: Assumes linearity, Sensitive to outliers

Applications: Price prediction • Sales forecasting • Trend analysis • Risk assessment

UNIVERSITY OF CALGARY

Logistic Regression

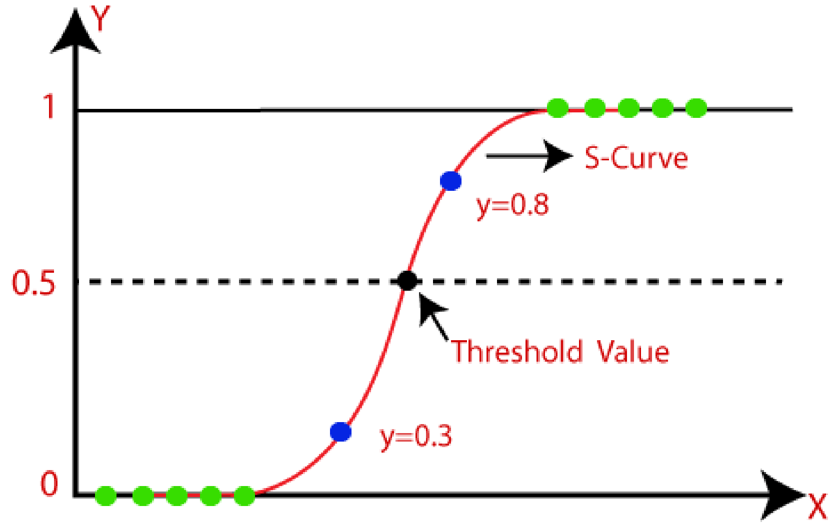
- Goal: Classify data into discrete categories (typically binary: 0 or 1)

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

where $z = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$
Output: $P(y=1|x) \in [0, 1]$

Loss Function:
Cross-Entropy Loss

$L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$



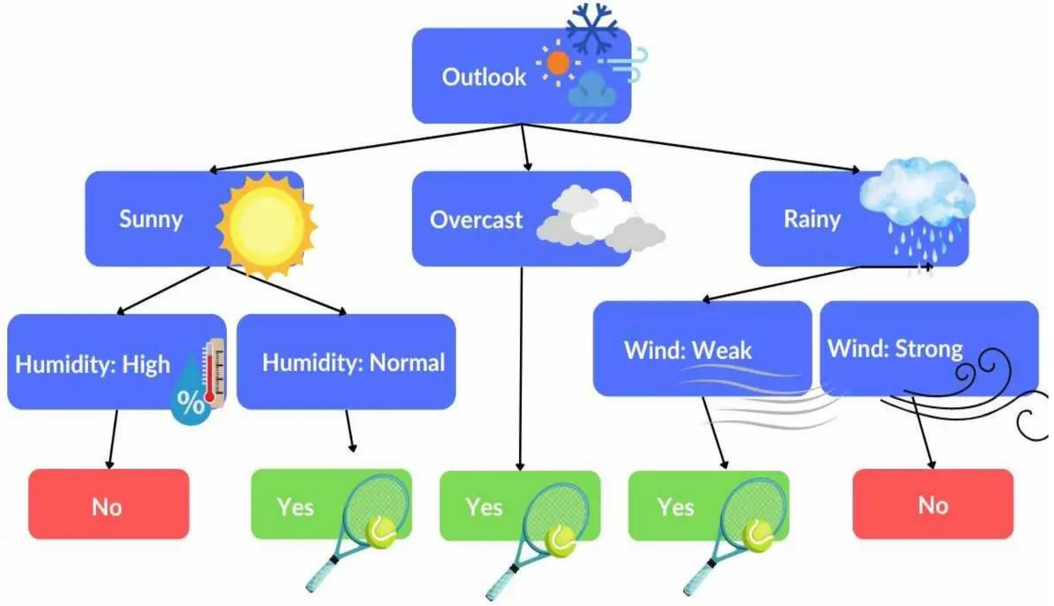
✓ Pros: Probabilistic, interpretable

✗ Cons: Linear boundary

Applications: Email Spam Detection • Medical diagnosis • Credit risk

Decision Trees

- Goal: Create a tree-like model of decisions for classification or regression
  - Splits data recursively based on feature values
    - Internal nodes = feature tests
    - Branches = test outcomes
    - Leaves = predicted class or value
- **Splitting Criteria**
  - **Gini Impurity:** Measures node impurity
    - Gini = $1 - \Sigma(p_i)^2$
  - **Entropy (Information Gain):** Measures information
    - Entropy = $-\Sigma\, p_i \log_2(p_i)$



✓ Pros: Interpretable, Handles non-linear data

✗ Cons: Prone to overfitting, Unstable to small data changes

Random Forests

- Goal: Ensemble of decision trees - majority vote wins
- **How it Works?**
  - **Bootstrap Sampling:** Random samples with replacement
  - **Random Feature Selection:** Each split uses subset of features
  - **Train Multiple Trees:** Each tree on different sample
  - **Aggregate Predictions:** Majority vote or average

✓ Pros: More accurate, less overfitting

✗ Cons: Less interpretable

K-Means Clustering

- Goal: Partition data into K clusters
- **Steps:**

1. Initialize: Choose K random centroids

2. Assignment: Assign each point to nearest centroid

3. Update: Recalculate centroids as mean of assigned points

4. Repeat: Until convergence

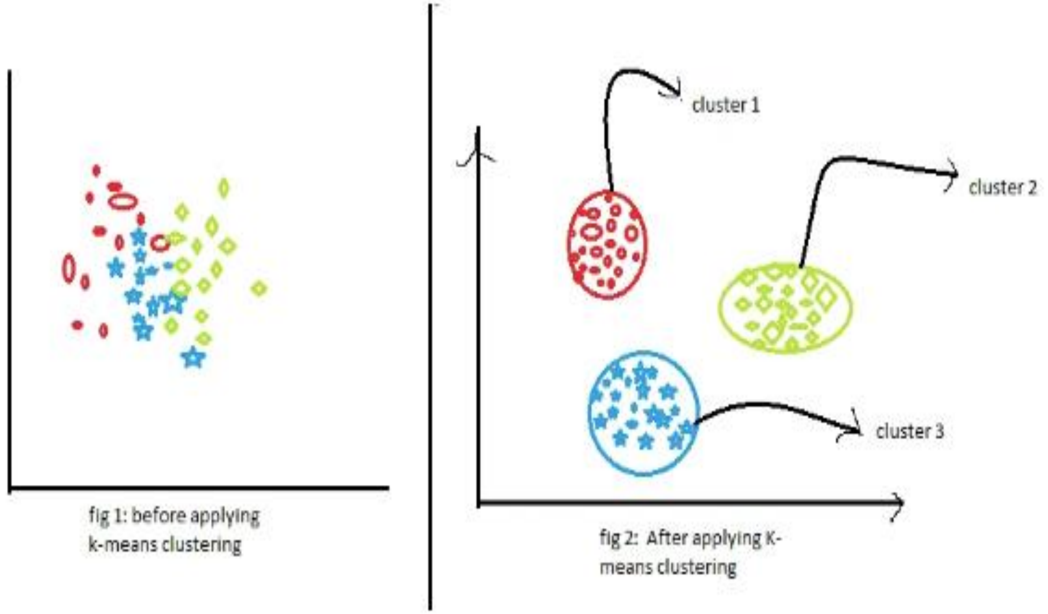


fig 1: before applying k-means clustering

fig 2: After applying K-means clustering

✓ Pros: Fast, scalable

✗ Cons: Must choose K

When to Use Which Algorithm?

Algorithm	Best For	Pros	Cons
Linear Regression	Linear relationships	Fast, interpretable	Assumes linearity
Logistic Regression	Binary classification	Simple, probabilistic	Linear boundary
Decision Trees	Non-linear, interpretability	Easy to visualize	Overfitting
Random Forest	Complex data	Robust, accurate	Less interpretable
SVM	High-dimensional	Effective in high dims	Slow on large data
K-Nearest Neighbors	Instance-based learning	Simple, no training	Slow prediction, memory intensive
K-Means	Segmentation	Simple, fast	Needs k specified

Model Evaluation

Measuring and Validating Model Performance

Why Evaluate?

- Assess model performance
- Compare different algorithms
- Detect overfitting/underfitting
- Make informed decisions

What to Evaluate?

- Prediction accuracy
- Generalization ability
- Computational efficiency
- Model interpretability

How to Evaluate?

- Appropriate metrics
- Cross-validation
- Train/test splits
- Confusion matrix analysis

Confusion Matrix

Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	TP	FP
	Negative	FN	TN

Definitions

True Positive (TP)

Correctly predicted positive

True Negative (TN)

Correctly predicted negative

False Positive (FP)

Incorrectly predicted positive (Type I Error)

False Negative (FN)

Incorrectly predicted negative (Type II Error)

Foundation for Classification Metrics

Classification Metrics

Accuracy
Formula: (TP + TN) / (TP + TN + FP + FN)

Use when: Balanced datasets

⚠ Misleading for imbalanced data

Precision
Formula: TP / (TP + FP)

Use when: Cost of FP is high

⚠ Ignores false negatives

Recall (Sensitivity)
Formula: TP / (TP + FN)

Use when: Cost of FN is high

⚠ Ignores false positives

F1-Score
Formula: 2 × (Precision × Recall) / (Precision + Recall)

Use when: Need balance between precision & recall

⚠ Equal weight to both metrics

Specificity
Formula: TN / (TN + FP)

Use when: True negative rate matters

⚠ Complements sensitivity

ROC-AUC
Formula: Area Under ROC Curve

Use when: Overall model discrimination ability

⚠ Not suitable for imbalanced data

Regression Metrics

Mean Absolute Error (MAE)

$MAE = (1/n)\ \Sigma\ |y_i - \hat{y}_i|$

Average absolute difference between predicted and actual

✓ Easy to interpret, same units as target ⚠ Doesn't penalize large errors heavily

[0, ∞), Lower is better

Mean Squared Error (MSE)

$MSE = (1/n)\ \Sigma\ (y_i - \hat{y}_i)^2$

Average of squared differences

✓ Penalizes large errors more ⚠ Units are squared, sensitive to outliers

[0, ∞), Lower is better

Root Mean Squared Error (RMSE)

$RMSE = \sqrt{MSE}$

Square root of MSE, same units as target

✓ Interpretable, penalizes large errors ⚠ Sensitive to outliers

[0, ∞), Lower is better

R^2 Score (Coefficient of Determination)

$R^2 = 1 - (SS\_res / SS\_tot)$

Proportion of variance explained by model

✓ Normalized, easy to interpret ⚠ Can be negative for poor models

(-∞, 1], Higher is better, 1 = perfect

Mean Absolute Percentage Error (MAPE)

$MAPE = (100/n)\ \Sigma\ |(y_i - \hat{y}_i) / y_i|$

Average percentage error

✓ Scale-independent, interpretable ⚠ Undefined when $y_i = 0$, biased toward low predictions

[0, ∞), Lower is better

Adjusted R^2

$Adj\ R^2 = 1 - [(1-R^2)(n-1)/(n-p-1)]$

R^2 adjusted for number of predictors

✓ Accounts for model complexity ⚠ More complex to calculate

(-∞, 1], Higher is better

Measuring Continuous Prediction Quality

Cross-Validation

Why Cross-Validation?

- Maximizes use of limited data
- More reliable performance estimates
- Reduces variance in evaluation
- Helps detect overfitting

K-Fold Cross-Validation

1. Split data into K equal folds
2. For each fold i:
   - Train on K-1 folds
   - Validate on fold i
3. Average performance across K folds

Typical K: 5 or 10

Use: Standard approach for most problems

Stratified K-Fold

1. Maintain class distribution in each fold
2. Ensures each fold is representative
3. Same process as K-Fold
4. Especially important for imbalanced data

Use: Classification with imbalanced classes

Leave-One-Out (LOO)

1. K = n (number of samples)
2. Each sample used once as validation
3. Very computationally expensive
4. Nearly unbiased estimator

Use: Small datasets, when computational cost is acceptable

Time Series Split

1. Respects temporal ordering
2. Train on past, test on future
3. No shuffling of data
4. Prevents data leakage

Use: Time-dependent data (stock prices, weather)

ROC Curve and AUC

What is ROC Curve?

Plot of True Positive Rate (TPR) vs False Positive Rate (FPR) at different classification thresholds

TPR (Recall/Sensitivity):
TPR = TP / (TP + FN)

FPR:
FPR = FP / (FP + TN)

AUC (Area Under Curve)

- AUC = 1.0: Perfect classifier
- AUC = 0.9-1.0: Excellent
- AUC = 0.8-0.9: Good
- AUC = 0.7-0.8: Fair
- AUC = 0.5-0.7: Poor
- AUC = 0.5: Random classifier
- AUC < 0.5: Worse than random

Advantages

- Threshold-independent evaluation
- Good for comparing models
- Works well for balanced datasets
- Visualizes trade-off between TPR and FPR
- Single metric (AUC) summarizes performance

Limitations

⚠ Misleading for highly imbalanced datasets

⚠ Doesn't reflect real-world costs of errors

⚠ May hide poor performance on minority class

Alternative for Imbalanced Data: Precision-Recall Curve

Bias-Variance Tradeoff

Underfitting
High Bias

Characteristics:
- Model too simple
- Fails to capture patterns
- Poor training performance
- Poor test performance
- High training error
- High validation error

Solutions:
- Increase model complexity
- Add more features
- Reduce regularization
- Train longer

Good Fit
Optimal Balance

Characteristics:
- Right model complexity
- Captures true patterns
- Good training performance
- Good test performance
- Low training error
- Low validation error

Solutions:
- This is the goal!
- Monitor performance
- Continue validation
- Deploy with confidence

Overfitting
High Variance

Characteristics:
- Model too complex
- Memorizes training data
- Excellent training performance
- Poor test performance
- Very low training error
- High validation error

Solutions:
- Reduce model complexity
- Add more training data
- Use regularization
- Early stopping

Total Error = Bias^2 + Variance + Irreducible Error

Machine Learning Best Practices

Data Management

- Split before preprocessing
- Use stratified splits
- Document sources
- Handle missing values

Model Development

- Start simple
- Establish baseline
- Use cross-validation
- Track experiments

Evaluation & Deployment

- Align metrics to goals
- Test on held-out data
- Monitor data drift
- Document assumptions

Common Pitfalls to Avoid

Data Leakage
Example: Preprocessing before split

Look-Ahead Bias
Example: Using future information

Selection Bias
Example: Non-representative data

Class Imbalance
Example: 99% negative, 1% positive

Wrong Metrics
Example: Accuracy for imbalanced data

Overfitting to Validation
Example: Excessive tuning

Key Takeaways

🎯 **Machine Learning = Learning from Data**

- Algorithms discover patterns without explicit programming

🤖 **Choose the Right Algorithm**

- No single algorithm is best for all problems
  - Linear/Logistic Regression for simple relationships
  - Decision Trees/Random Forests for complex, non-linear data

⚖️ **Balance Bias and Variance**

- Too simple → High Bias (underfitting)
- Too complex → High Variance (overfitting)

📊 **Evaluate Properly**

- Use appropriate metrics for your problem
- Always validate on unseen test data

Thank you!

UNIVERSITY OF CALGARY