Training, Classification & Evaluation

1. Training ML Models

Model training is the process of teaching an algorithm to make predictions by learning patterns from data. The algorithm adjusts its internal parameters to minimize prediction errors.

Key Components of Training

Training Data

- Dataset used to teach the model
- Consists of input features (X) and target outputs (y)
- Should be representative of the problem domain
- Typically 60-80% of total available data

Loss Function

- Measures how wrong the model's predictions are
- Different problems use different loss functions
- Common examples: Mean Squared Error (MSE), Cross-entropy
- Goal: minimize this function during training

Optimization Algorithm

- Method used to minimize the loss function
- Gradient Descent is the most common approach
- Updates model parameters iteratively
- Learning rate controls step size of updates

Training Process Steps

1. **Initialize** model parameters randomly
2. **Forward Pass**: Make predictions on training data
3. **Calculate Loss**: Compare predictions to actual values
4. **Backward Pass**: Calculate gradients of loss function
5. **Update Parameters**: Adjust weights using optimization algorithm
6. **Repeat** until convergence or maximum iterations reached

Important Training Concepts

Epochs

- One complete pass through the entire training dataset

- Models typically train for multiple epochs
- Too few: underfitting; too many: overfitting

Batch Size

- Number of samples processed before updating parameters
- Batch Gradient Descent: entire dataset
- Stochastic: single sample
- Mini-batch: subset of data (most common)

Learning Rate

- Controls how much parameters change with each update
- Too high: may overshoot optimal solution
- Too low: slow convergence
- Often requires tuning for best results

2. Classification with Logistic Regression

Logistic regression is a **classification** algorithm used to predict categorical outcomes. It estimates the probability that an instance belongs to a particular class.

Mathematical Foundation

Sigmoid Function

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

- Maps any real number to value between 0 and 1
- Creates smooth S-shaped curve
- Output interpreted as probability

Linear Combination

$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$

- β_0: intercept (bias term)
- β_1, β_2, ..., β_n: feature weights
- x_1, x_2, ..., x_n: input features

Prediction Formula

$P(y=1|x) = \sigma(z) = 1 / (1 + e^{(-(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n))})$

Binary Classification

- Predicts probability of positive class (class 1)
- Decision threshold typically set at 0.5
- If $P(y=1|x) \geq 0.5 \rightarrow$ predict class 1
- If $P(y=1|x) < 0.5 \rightarrow$ predict class 0

Multi-class Classification

One-vs-Rest (OvR)

- Train separate binary classifier for each class
- Each classifier: "this class" vs "all other classes"
- Predict class with highest probability

Multinomial Logistic Regression

- Direct extension to multiple classes
- Uses softmax function instead of sigmoid
- More efficient than OvR approach

Cost Function

Log-Likelihood (Cross-entropy)

$$\text{Cost} = -[y \cdot \log(h) + (1-y) \cdot \log(1-h)]$$

- y: actual label (0 or 1)
- h: predicted probability
- Penalizes confident wrong predictions heavily
- Convex function (guarantees global minimum)

Advantages of Logistic Regression

- Simple and fast to train
- No tuning of hyperparameters required
- Provides probability estimates
- Less prone to overfitting with low-dimensional data
- Good baseline for classification problems

Limitations

- Assumes linear relationship between features and log-odds
- Sensitive to outliers
- Requires large sample sizes for stable results
- Can struggle with complex relationships

3. Model Evaluation with Accuracy

Accuracy measures the proportion of correct predictions out of total predictions made.

Formula

Accuracy = (Correct Predictions) / (Total Predictions)
Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

Confusion Matrix

Actual vs predicted classifications:

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

When Accuracy Works Well

- Balanced datasets (equal class distribution)
- When all misclassification costs are equal
- Binary classification with similar importance for both classes
- As a quick initial assessment metric

Limitations of Accuracy

Class Imbalance Problem

- Example: 95% of emails are not spam
- Model predicting "not spam" for everything gets 95% accuracy
- But completely fails to identify actual spam
- Accuracy can be misleading in such cases

Equal Weighting Assumption

- Treats all errors as equally important
- In medical diagnosis: missing cancer (FN) worse than false alarm (FP)
- Accuracy doesn't capture these different costs

Alternative Metrics to Consider

Precision

Precision = TP / (TP + FP)

- Of predicted positives, how many were correct?
- Important when false positives are costly

Recall (Sensitivity)

Recall = TP / (TP + FN)

- Of actual positives, how many were found?
- Important when false negatives are costly

F1-Score

F1 = 2 × (Precision × Recall) / (Precision + Recall)

- Harmonic mean of precision and recall
- Balances both metrics
- Good for imbalanced datasets

Cross-Validation for Robust Evaluation

K-Fold Cross-Validation

1. Split data into k equal parts (folds)
2. Train on k-1 folds, test on remaining fold
3. Repeat k times, each fold used as test set once
4. Average results across all folds

Benefits

- More reliable estimate of model performance
- Uses all data for both training and testing
- Reduces variance in performance estimates
- Helps detect overfitting

Best Practices for Model Evaluation

Train-Validation-Test Split

- Training set (60%): Train the model
- Validation set (20%): Tune hyperparameters
- Test set (20%): Final unbiased evaluation

Multiple Metrics

- Don't rely on accuracy alone
- Consider precision, recall, F1-score
- Use ROC curves for probability thresholds
- Choose metrics based on problem requirements

Baseline Comparison

- Compare against simple baselines
- Random classifier performance
- Most frequent class classifier
- Helps contextualize model performance

Summary

Training ML Models involves iteratively adjusting parameters to minimize prediction errors using optimization algorithms like gradient descent. Key considerations include choosing appropriate loss functions, learning rates, and avoiding overfitting.

Logistic Regression transforms linear combinations of features through the sigmoid function to produce probabilities for classification. It's simple, interpretable, and effective for linearly separable problems.

Accuracy measures overall correctness but can be misleading with imbalanced datasets. Always consider multiple evaluation metrics and use cross-validation for robust performance assessment.