Machine Learning Fundamentals

Machine learning is a subset of artificial intelligence that focuses on building systems that learn from data. Instead of being explicitly programmed, these systems identify patterns and make decisions with minimal human intervention.

The three main categories of machine learning are:
1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Each category has its own set of algorithms, use cases, and evaluation criteria. Machine learning has become essential in modern software engineering.

Feature Engineering Techniques

Feature engineering is the process of using domain knowledge to create features that make machine learning algorithms work better. Good features can significantly improve model performance.

Common feature engineering techniques include:
- One-hot encoding for categorical variables
- Normalization and standardization of numerical features
- Polynomial features for capturing non-linear relationships
- Feature crossing for interaction effects
- Binning continuous variables into discrete buckets
- Log transformation for skewed distributions
- Target encoding for high-cardinality categoricals

Feature selection methods help reduce dimensionality:
- Filter methods (correlation, chi-squared)
- Wrapper methods (forward selection, backward elimination)
- Embedded methods (L1 regularization, tree importance)

Data Preprocessing Steps

Data preprocessing is a crucial step in any machine learning pipeline. Raw data is often incomplete, inconsistent, and contains errors.

Key preprocessing steps include:

1. Data Cleaning
   - Handle missing values (imputation, deletion)
   - Remove duplicates
   - Fix inconsistent formatting

2. Data Transformation
   - Scaling: MinMaxScaler, StandardScaler, RobustScaler
   - Encoding: Label encoding, One-hot encoding
   - Text preprocessing: tokenization, lemmatization, stopword removal

3. Data Integration
   - Merging datasets from multiple sources
   - Resolving schema conflicts
   - Entity resolution

4. Data Reduction
   - Dimensionality reduction (PCA, t-SNE)
   - Numerosity reduction (sampling, clustering)
   - Data compression

Advanced Data Preprocessing

Handling imbalanced datasets is a critical preprocessing challenge:
- Oversampling (SMOTE, ADASYN)
- Undersampling (Random, Tomek links)
- Hybrid approaches (SMOTETomek)
- Cost-sensitive learning

Data preprocessing for text:
- Tokenization breaks text into individual tokens
- Stemming reduces words to their root form
- Lemmatization converts words to their dictionary form
- TF-IDF weighting captures term importance
- Word embeddings create dense vector representations

Data preprocessing for images:
- Resizing and cropping
- Normalization (pixel value scaling)
- Data augmentation (rotation, flipping, color jittering)
- Histogram equalization

Gradient Descent Optimization

Gradient descent is the backbone of modern machine learning optimization. It iteratively adjusts model parameters to minimize a loss function.

Types of gradient descent:
1. Batch Gradient Descent - uses entire dataset per update
2. Stochastic Gradient Descent (SGD) - uses single sample per update
3. Mini-batch Gradient Descent - uses subset of data per update

Learning rate is the most critical hyperparameter:
- Too high: overshooting, divergence
- Too low: slow convergence, stuck in local minima

Learning rate schedules:
- Step decay
- Exponential decay
- Cosine annealing
- Warm-up strategies

Advanced Optimization Algorithms

Beyond basic SGD, several optimization algorithms improve convergence:

Momentum-based methods:
- SGD with Momentum: accumulates past gradients
- Nesterov Accelerated Gradient: look-ahead gradient computation

Adaptive learning rate methods:
- AdaGrad: adapts learning rate per parameter
- RMSProp: uses moving average of squared gradients
- Adam: combines momentum and adaptive learning rates
- AdamW: Adam with decoupled weight decay

Second-order methods:
- Newton's method: uses Hessian matrix
- L-BFGS: approximate second-order optimization
- Natural gradient descent

Gradient clipping prevents exploding gradients:
- Clip by value
- Clip by norm

Supervised Learning Algorithms

Supervised learning uses labeled data to learn a mapping from inputs to outputs.

Classification algorithms:
- Logistic Regression: linear decision boundary
- Support Vector Machines: maximum margin classifier
- Decision Trees: recursive feature splitting
- Random Forests: ensemble of decision trees
- Gradient Boosting: sequential tree building
- Neural Networks: deep learning classifiers

Regression algorithms:
- Linear Regression: ordinary least squares
- Ridge Regression: L2 regularization
- Lasso Regression: L1 regularization
- Elastic Net: combined L1 and L2
- Support Vector Regression
- Gradient Boosting Regression

Key concepts:
- Bias-variance tradeoff
- Cross-validation for model selection
- Hyperparameter tuning (grid search, random search, Bayesian optimization)

Decision Tree Classification

Decision trees are intuitive models that make predictions by learning simple decision rules inferred from data features.

Key algorithms:
- ID3: uses information gain (entropy)
- C4.5: handles continuous attributes, missing values
- CART: binary splits, supports regression

Splitting criteria:
- Information Gain (Entropy)
- Gini Impurity
- Variance Reduction (for regression)

Overfitting in decision trees:
- Pre-pruning: limit tree depth, minimum samples per leaf
- Post-pruning: cost-complexity pruning (CCP)
- Minimum impurity decrease

Ensemble methods built on decision trees:
- Random Forests: bagging + feature randomness
- Gradient Boosting: XGBoost, LightGBM, CatBoost
- AdaBoost: weighted sequential learning

Overfitting and Regularization

Overfitting occurs when a model learns noise in the training data rather than the underlying pattern. The model performs well on training data but poorly on unseen data.

Signs of overfitting:
- Large gap between training and validation accuracy
- Model complexity is too high for the data
- Training loss decreases while validation loss increases

Regularization techniques:
1. L1 Regularization (Lasso): encourages sparsity
2. L2 Regularization (Ridge): penalizes large weights
3. Elastic Net: combines L1 and L2
4. Dropout: randomly deactivates neurons during training
5. Early Stopping: halt training when validation loss plateaus
6. Data Augmentation: increase effective training set size
7. Batch Normalization: normalizes layer inputs

The bias-variance tradeoff:
- High bias: underfitting (model too simple)
- High variance: overfitting (model too complex)
- Goal: find the sweet spot that minimizes total error

Neural Network Architecture

Neural networks are composed of layers of interconnected nodes (neurons).
Each connection has a weight, and each neuron has a bias and activation function.

Basic architecture components:
- Input layer: receives raw features
- Hidden layers: learn representations
- Output layer: produces predictions

Common layer types:
- Dense (fully connected): every neuron connects to every neuron in next layer
- Convolutional: learns spatial features using filters
- Recurrent: processes sequential data with memory
- Attention: weighs importance of different input parts

Activation functions:
- ReLU: max(0, x) ? most popular for hidden layers
- Sigmoid: outputs probability (0, 1)
- Tanh: outputs value (-1, 1)
- Softmax: multi-class probability distribution
- GELU: smooth approximation of ReLU

Deep Neural Network Architectures

Modern deep learning architectures have revolutionized AI:

Convolutional Neural Networks (CNNs):
- LeNet-5: pioneering CNN for digit recognition
- AlexNet: deep CNN that won ImageNet 2012
- VGGNet: very deep networks with small filters
- ResNet: skip connections for training very deep networks
- EfficientNet: compound scaling of depth, width, resolution

Recurrent Neural Networks (RNNs):
- Vanilla RNN: basic sequence processing
- LSTM: long short-term memory for long dependencies
- GRU: gated recurrent unit (simplified LSTM)
- Bidirectional RNNs: process sequence in both directions

Transformer Architecture:
- Self-attention mechanism
- Multi-head attention
- Positional encoding
- BERT, GPT, T5 family of models
- Vision Transformers (ViT)

Model Evaluation Metrics

Choosing the right evaluation metric is crucial for model assessment.

Classification metrics:
- Accuracy: correct predictions / total predictions
- Precision: true positives / (true positives + false positives)
- Recall: true positives / (true positives + false negatives)
- F1 Score: harmonic mean of precision and recall
- ROC-AUC: area under receiver operating characteristic curve
- Log Loss: cross-entropy loss for probabilistic predictions

Regression metrics:
- MSE: mean squared error
- RMSE: root mean squared error
- MAE: mean absolute error
- R-squared: coefficient of determination
- MAPE: mean absolute percentage error

Ranking metrics:
- Precision@K: precision in top-K results
- Recall@K: recall in top-K results
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gain (NDCG)

Cross Validation Strategy

Cross-validation provides robust estimates of model performance on unseen data.

K-Fold Cross-Validation:
- Split data into K equal folds
- Train on K-1 folds, validate on remaining fold
- Repeat K times, average results
- Typical K values: 5, 10

Stratified K-Fold:
- Maintains class distribution in each fold
- Essential for imbalanced datasets

Leave-One-Out Cross-Validation (LOOCV):
- K equals the number of samples
- Computationally expensive but low bias

Time Series Cross-Validation:
- Expanding window: growing training set
- Sliding window: fixed-size training set
- No random shuffling to preserve temporal order

Nested Cross-Validation:
- Outer loop: estimate model performance
- Inner loop: hyperparameter tuning
- Prevents information leakage

Dimensionality Reduction Methods

Dimensionality reduction transforms high-dimensional data into a lower-dimensional representation while preserving important information.

Linear methods:
- PCA (Principal Component Analysis): finds orthogonal directions of maximum variance
- LDA (Linear Discriminant Analysis): maximizes class separability
- SVD (Singular Value Decomposition): matrix factorization approach
- Factor Analysis: identifies latent factors

Non-linear methods:
- t-SNE: preserves local neighborhood structure (visualization)
- UMAP: uniform manifold approximation (faster than t-SNE)
- Autoencoders: neural network-based compression
- Kernel PCA: PCA with non-linear kernel functions
- Isomap: preserves geodesic distances

Feature selection methods:
- Mutual information
- Variance threshold
- Recursive feature elimination
- L1-based selection (Lasso)

Applications:
- Visualization of high-dimensional data
- Noise reduction
- Computational efficiency
- Avoiding curse of dimensionality

Deployment and Production ML

Taking a model from notebook to production requires engineering discipline.

Model serving:
- REST APIs (Flask, FastAPI)
- gRPC for low-latency serving
- Batch prediction pipelines
- Edge deployment (ONNX, TensorFlow Lite)

MLOps practices:
- Model versioning (MLflow, DVC)
- Experiment tracking
- CI/CD for ML pipelines
- A/B testing for model comparison
- Monitoring model drift

Infrastructure:
- Containerization (Docker)
- Orchestration (Kubernetes)
- Feature stores (Feast, Tecton)
- Model registries

Best practices:
- Reproducible training pipelines
- Data validation and schema enforcement
- Automated retraining triggers
- Canary deployments for gradual rollout