

Lecture: Introduction to Machine Learning

KTH AI Student

February 13, 2025

Introduction

- ▶ Machine Learning is a core aspect of modern AI.
- ▶ Famous for its ability to learn and improve from data without explicit programming.
- ▶ Goals of this section:
 - ▶ Understand the basics of Machine Learning.
 - ▶ Explore its types and applications.

What is Machine Learning?

- ▶ **Definition:** Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that enables systems to learn and improve from experience.
- ▶ **Core Idea:** Use data to make predictions or decisions.
- ▶ **Applications:**
 - ▶ Predicting house prices.
 - ▶ Detecting spam emails.
 - ▶ Recommending products.

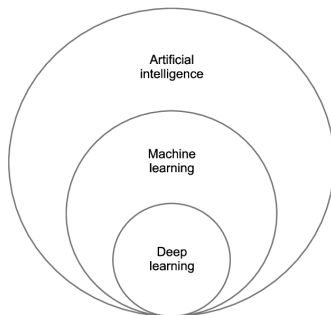


Figure: Machine Learning

Types of Machine Learning - Overview

- ▶ Machine Learning can be broadly categorized into three types:
 - ▶ Supervised Learning.
 - ▶ Unsupervised Learning.
 - ▶ Reinforcement Learning.

Supervised Learning

- ▶ Learns from labeled data.
- ▶ Examples:
 - ▶ Classification: Predicting if an email is spam or not.
 - ▶ Regression: Predicting house prices.
- ▶ Key formula: $y = f(X)$, where X are inputs, and y are outputs.

Unsupervised Learning

- ▶ Learns from unlabeled data.
- ▶ Example: Grouping customers based on purchasing behavior.
- ▶ Focuses on finding patterns and structures in data.

Reinforcement Learning

- ▶ Learns by interacting with the environment.
- ▶ Receives rewards for correct actions and penalties for incorrect ones.
- ▶ Example: Teaching a robot to walk.

Data Preprocessing Overview

- ▶ Essential step before training ML models.
- ▶ Tasks:
 - ▶ Load and clean data.
 - ▶ Handle missing values and categorical variables.
 - ▶ Normalize and split data into training and test sets.
- ▶ Example Dataset: Heart Disease dataset.

Steps in Data Preprocessing (Part 1)

- ▶ **Loading Data:** Use pandas to load datasets.
- ▶ **Handling Missing Values:** Remove or fill missing entries.

```
1 import pandas as pd
2
3 # Load dataset
4 data = pd.read_csv('heart_disease.csv')
5
6 # Display first few rows
7 print(data.head())
```

Steps in Data Preprocessing (Part 2)

- ▶ **Encoding Categorical Variables:** Convert categories to numerical values.
- ▶ **Splitting Data:** Divide into training and test sets, typically 80-20 split. We divide them to prevent overfitting.

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.preprocessing import LabelEncoder
3
4 # Encode categorical variables
5 label_encoder = LabelEncoder()
6 data['category'] = label_encoder.fit_transform(data['
    category'])
7
8 # Split data
9 X = data.drop('target', axis=1)
10 y = data['target']
11 X_train, X_test, y_train, y_test = train_test_split(X, y
    , test_size=0.2, random_state=42)
```

Overfitting and Underfitting

► **Overfitting:**

- Model learns the training data too well, including noise and outliers.
- High accuracy on training data but poor generalization to new data.
- Prevention:
 - Use simpler models.
 - Apply regularization techniques (e.g., L1, L2 regularization).
 - Use cross-validation to tune hyperparameters.

► **Underfitting:**

- Model is too simple to capture the underlying patterns in the data.
- Poor performance on both training and test data.
- Prevention:
 - Use more complex models.
 - Increase the number of features or use feature engineering.
 - Reduce regularization.

Advanced Preprocessing

- ▶ Normalize the data to improve model performance:
 - ▶ StandardScaler: Standardize features by removing the mean and scaling to unit variance.
- ▶ Feature Engineering:
 - ▶ Create new features by combining existing ones.
 - ▶ Example: Derive BMI from weight and height.

```
1 from sklearn.preprocessing import StandardScaler
2
3 # Normalize data
4 scaler = StandardScaler()
5 X_train = scaler.fit_transform(X_train)
6 X_test = scaler.transform(X_test)
7
8 # Feature engineering example
9 data['BMI'] = data['weight'] / (data['height'] / 100) **  
    2
```

Training a Simple Model: Perceptron

- ▶ **Perceptron:** Basic classification model.
- ▶ **Key Steps:**
 - ▶ Initialize model parameters.
 - ▶ Train model by updating weights.
 - ▶ Evaluate model accuracy on test data.

```
1 from sklearn.linear_model import Perceptron
2 from sklearn.metrics import accuracy_score
3
4 # Initialize and train model
5 model = Perceptron()
6 model.fit(X_train, y_train)
7
8 # Predict and evaluate
9 y_pred = model.predict(X_test)
10 accuracy = accuracy_score(y_test, y_pred)
11 print(f'Accuracy: {accuracy}')
```

Perceptron Visualization

- ▶ Example of decision boundary plot.
- ▶ Helps in understanding how the model separates classes.

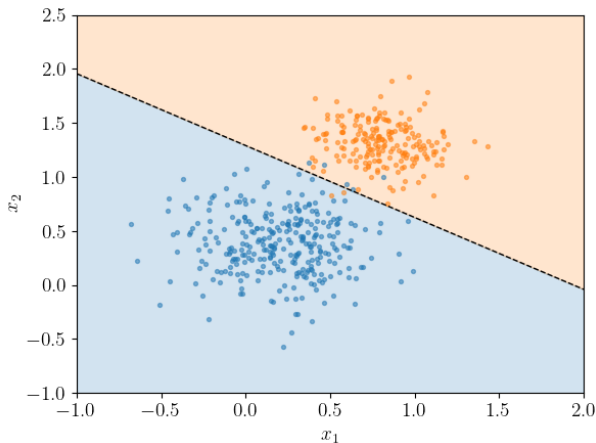


Figure: Decision Boundary of Perceptron Model

Advanced Classification Models (Part 1)

- ▶ Test various models using scikit-learn:
 - ▶ Logistic Regression.
 - ▶ Support Vector Machines.

```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.svm import SVC
3
4 # Logistic Regression
5 log_reg = LogisticRegression()
6 log_reg.fit(X_train, y_train)
7 log_reg_pred = log_reg.predict(X_test)
8
9 # Support Vector Machine
10 svm = SVC()
11 svm.fit(X_train, y_train)
12 svm_pred = svm.predict(X_test)
```

Advanced Classification Models (Part 2)

- ▶ Test more advanced models:
 - ▶ Decision Trees.
 - ▶ Random Forests.

```
1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.ensemble import RandomForestClassifier
3
4 # Decision Tree
5 tree = DecisionTreeClassifier()
6 tree.fit(X_train, y_train)
7 tree_pred = tree.predict(X_test)
8
9 # Random Forest
10 forest = RandomForestClassifier()
11 forest.fit(X_train, y_train)
12 forest_pred = forest.predict(X_test)
```


Evaluation Metrics for Classification

► Metrics to evaluate model performance:

- Accuracy: $\frac{\text{Correct Predictions}}{\text{Total Predictions}}$.
- Precision: $\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$.
- Recall: $\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$.
- F1-score: $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

```
1 from sklearn.metrics import precision_score,
   recall_score, f1_score
2
3 # Calculate metrics
4 precision = precision_score(y_test, y_pred)
5 recall = recall_score(y_test, y_pred)
6 f1 = f1_score(y_test, y_pred)
7
8 print(f'Precision: {precision}')
9 print(f'Recall: {recall}')
10 print(f'F1 Score: {f1}')
```

Model Evaluation Metrics

- ▶ **Accuracy:** Proportion of correct predictions.
- ▶ **Precision:** Correct positive predictions out of total predicted positives.
- ▶ **Recall:** Correct positive predictions out of actual positives.
- ▶ **F1-Score:** Harmonic mean of precision and recall.

What is a Hyperparameter?

- ▶ **Definition:** Hyperparameters are parameters whose values are set before the learning process begins.
- ▶ **Examples:**
 - ▶ Learning rate in gradient descent.
 - ▶ Number of trees in a random forest.
 - ▶ Number of clusters in K-Means.
- ▶ **Importance:**
 - ▶ They control the training process and model complexity.
 - ▶ Proper tuning can significantly improve model performance.

Hyperparameter Tuning

- ▶ **Grid Search:** Explore combinations of parameters to find the best configuration.
- ▶ **Cross-Validation:** Evaluate model on different data splits to ensure reliability.

```
1 from sklearn.model_selection import GridSearchCV
2
3 # Define parameter grid
4 param_grid = {
5     'C': [0.1, 1, 10],
6     'kernel': ['linear', 'rbf']
7 }
8
9 # Initialize Grid Search
10 grid_search = GridSearchCV(SVC(), param_grid, cv=5)
11 grid_search.fit(X_train, y_train)
12
13 # Best parameters
14 print(f'Best Parameters: {grid_search.best_params_}')
```

Overview of Regression Models

- ▶ Examples of models:
 - ▶ Linear Regression.
 - ▶ Support Vector Machines for Regression.
 - ▶ Decision Trees.
 - ▶ Random Forests.

Evaluating Regression Models

► Metrics to assess model performance:

► Mean Squared Error. $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$.

► Coefficient of Determination. $R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$.

```
1 from sklearn.metrics import mean_squared_error, r2_score
2
3 # Calculate metrics
4 mse = mean_squared_error(y_test, y_pred)
5 r2 = r2_score(y_test, y_pred)
6
7 print(f'Mean Squared Error: {mse}')
8 print(f'R^2 Score: {r2}')
```

Visualizing Regression Results

- ▶ Compare predicted values against true values.
- ▶ Use scatter plots for analysis.

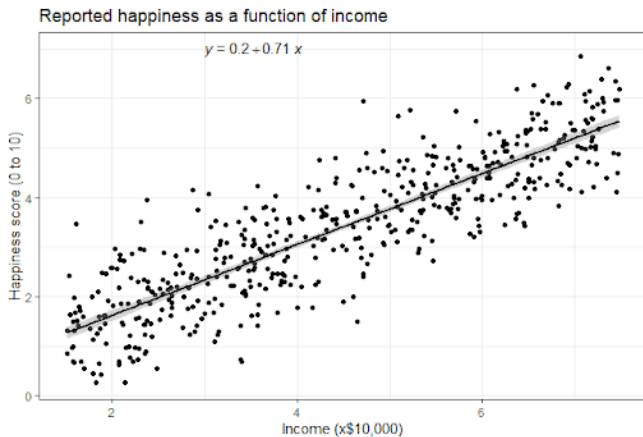


Figure: Regression Results Visualization

Introduction to Clustering

- ▶ **Definition:** Group similar data points based on features.
- ▶ **Applications:**
 - ▶ Customer segmentation.
 - ▶ Image segmentation.

Examples of Clustering Algorithms

- ▶ Common algorithms:
 - ▶ K-Means.
 - ▶ DBSCAN.
 - ▶ Agglomerative Clustering.
 - ▶ Gaussian Mixture Models.

```
1 from sklearn.cluster import KMeans
2
3 # Initialize and fit K-Means
4 kmeans = KMeans(n_clusters=3)
5 kmeans.fit(X)
6
7 # Predict clusters
8 clusters = kmeans.predict(X)
9
10 # Cluster centers
11 centers = kmeans.cluster_centers_
```

Visualizing Clusters

- ▶ Plot clusters using scatter plots with different colors.
- ▶ Analyze cluster characteristics and separability.

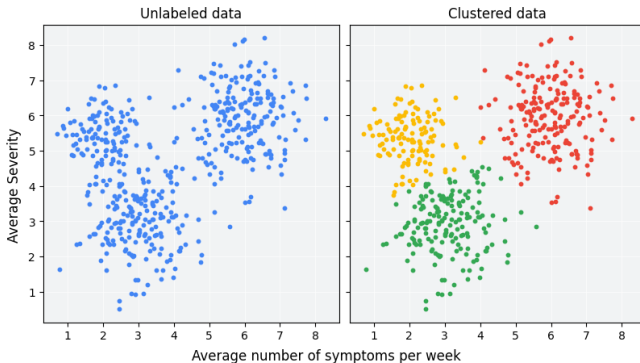


Figure: Cluster Visualization

Summary

- ▶ Recap of all tasks and key concepts:
 - ▶ Introduction to Machine Learning.
 - ▶ Data preprocessing steps.
 - ▶ Training and evaluating classification and regression models.
 - ▶ Clustering and its applications.
- ▶ Importance of proper evaluation and tuning in ML pipelines.

Resources

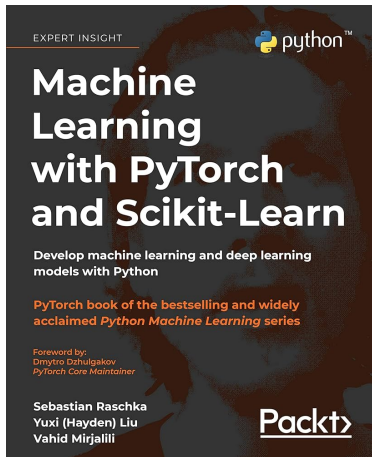


Figure: This one good for code + basic theory

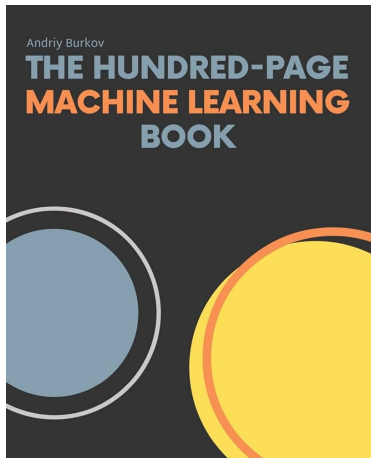


Figure: This one good for theory

Resources

coursera

Machine Learning Specialization

DeepLearning.AI | Stanford | ONLINE



Figure: You can get the certification for free if you say you are a student

Resources

You can also take the following courses at KTH:

- ▶ **Foundations of Machine Learning (DD1420)**: Introduces the basic principles and algorithms in machine learning.
- ▶ **Artificial Intelligence (DD2380)**: Covers fundamental AI concepts and techniques.
- ▶ **Machine Learning, Advanced Course (DD2434)**: Delves deeper into advanced machine learning methodologies and applications.
- ▶ **Deep Learning in Data Science (DD2424)**: Focuses on deep learning techniques for data science, emphasizing neural networks and large-scale data analysis.
- ▶ **Artificial Neural Networks and Deep Architectures (DD2437)**: Provides an in-depth understanding of neural network architectures and training methods.