Machine Learning Fundamentals

The first ML application that really became mainstream, improving the lives of hundreds of millions of people, took over the world back in the 1990s: the spam filter. Hundreds of ML applications that now quietly power hundreds of products and features that you use regularly: voice prompts, automatic translation, image search, product recommendations, and many more.

Machine learning is the science (and art) of programming computers so they can learn from data. [Machine learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

examples that the system uses to learn are called the training set. Each training example is called a training instance (or sample). The part of a machine learning system that learns and makes predictions is called a model. Neural networks and random forests are examples of models.

Machine learning is great for:

1. Problems for which existing solutions require a lot of fine-tuning or long lists of rules (a machine learning model can often simplify code and perform better than the traditional approach)
2. Complex problems for which using a traditional approach yields no good solution (the best machine learning techniques can perhaps find a solution)
3. Fluctuating environments (a machine learning system can easily be retrained on new data, always keeping it up to date)
4. Getting insights about complex problems and large amounts of data

Examples of Applications

- Detecting tumors in brain scans
- Analyzing images of products on a production line to automatically classify them
- Automatically classifying news articles
- Automatically flagging offensive comments on discussion forums
- Summarizing long documents automatically
- Creating a chatbot or a personal assistant
- Forecasting your company’s revenue next year, based on many performance metrics
- Making your app react to voice commands
- Detecting credit card fraud
- Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment
- Recommending a product that a client may be interested in, based on past purchases

The main types of machine learning (ML), based on the type of supervision or guidance the model receives during training:

1. Supervised Learning

In supervised learning, the model is trained on labeled data. This means that for every input, there is a corresponding correct output (label). The goal is for the model to learn the relationship between inputs and outputs so it can predict labels for new, unseen data.

- **Examples**:
  - Classification (e.g., identifying whether an email is spam or not)
  - Regression (e.g., predicting house prices)

2. Unsupervised Learning

In unsupervised learning, the model is given data without labels, and the goal is to discover patterns or relationships in the data. It often involves clustering, dimensionality reduction, or anomaly detection.

- **Examples**:
  - Clustering (e.g., grouping customers based on purchasing behavior)
  - Dimensionality reduction (e.g., reducing the number of variables in a dataset)
  - Anomaly Detection
  - Association Rule learning

3. Reinforcement Learning

In reinforcement learning, the model interacts with an environment and learns through trial and error, receiving feedback in the form of rewards or penalties. The aim is to maximize the cumulative reward over time by learning the best actions to take in different situations.

- **Examples**:
  - Game-playing agents (e.g., AlphaGo)
  - Robotics (e.g., robots learning to navigate a space)

4. Semi-Supervised Learning

Semi-supervised learning is a blend of supervised and unsupervised learning. In this approach, the model is trained on a small amount of labeled data and a large amount of unlabeled data. The goal is to improve learning efficiency by leveraging the unlabeled data to enhance the model's performance.

- **Use Case**: When labeling data is expensive or time-consuming (e.g., medical image classification where only some images are labeled by experts, but many unlabeled images exist).
- **Example**: Using a small set of labeled photos of animals, combined with a larger set of unlabeled animal images, to improve classification accuracy.

5. Self-Supervised Learning

Self-supervised learning is a relatively new approach, often used in deep learning, where the model learns from the data by generating its own labels. In this case, the learning task is designed in such a way that the model can generate labels from the input data itself (no human-provided labels are necessary). The model learns to predict parts of the data from other parts, essentially creating pseudo-labels.

- **Use Case**: This is widely used in tasks such as natural language processing (NLP) and computer vision, where huge amounts of unlabeled data are available.
- **Example**: Predicting missing parts of an image or predicting the next word in a sentence (like how GPT models are trained).

Transferring knowledge from one task to another is called transfer learning.

No Free Lunch (NFL) Theorem:

The NFL theorem states that if you make no assumptions about the data, then:

1. No model is inherently better than any other.
2. No model is guaranteed to work bette r than any other. In other words, without any prior knowledge or assumption s about the data, all models are equally good (or bad). This means that: There is no universal best model that works for all problems.
3. Model selection is crucial to find the best approach for a specific problem.

Implications :

1. Assumptions are necessary : To prefer one model over another, you need to make some reasonable assumption s about the data.
2. Model evaluation is essential: Since you can't evaluate all models, you need to s elect a few reasonable ones and compare their performance.
3. No silver bullet: There is no single model that works for all problems; each problem requires careful consideration and evaluation.

Batch Learning (Offline Learning)

- **Definition**: In batch learning, the model is trained using the entire dataset at once, or in large chunks (batches). The learning process occurs periodically and the model is updated only after processing the entire dataset or large portions of it.
- **Key Features**:
  - **Static Dataset**: The model is trained on a fixed dataset, and it doesn't change over time.
  - **Computational Cost**: Requires significant computational resources because the entire dataset (or a large batch) needs to be processed at once.
  - **Efficiency**: Best suited when the dataset is large but can fit into memory, or when there is enough time to train before deployment.
  - **Updates**: The model is updated periodically (e.g., daily, weekly), but not continuously.

- Models performance decades over time because the world grows , this phenomenon is called model rot or data drift. Retrain the model regularly

Online Learning

- **Definition**: In online learning, the model is trained incrementally as data arrives, updating continuously or after every new data point (or small batch). The model learns in real-time without waiting for a large dataset.
- **Key Features**:
  - **Dynamic Dataset**: The model adapts as new data is made available, which makes it suitable for situations where the data distribution might change over time (non-stationary data).
  - **Computational Efficiency**: Requires less memory and computing power because data is processed incrementally, instead of all at once.
  - **Adaptability**: Online learning can handle streaming data and is good for systems that need to react quickly to changes in the environment or incoming data.
  - **Updates**: The model is updated after each data point or small batch, which makes it responsive to real-time data changes.
  - It can be used to train huge datasets that cannot fit in the memory (out of core learning) by dividing the datasets into mini batches.
- One important parameter of online learning systems is how fast they should adapt to changing data: this is called the learning rate. If you set a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data
- Conversely, if you set a low learning rate, the system will have more inertia; that is, it will learn more slowly, but it will also be less sensitive to noise in the new data or to sequences of non representative data points

1. Instance-Based Learning (Memory-Based Learning)

- **Definition**: In instance-based learning, the model simply memorizes the training data and uses it directly to make predictions for new inputs. It doesn't create an explicit model or generalize beyond the training data; instead, predictions are made by comparing new examples to stored instances from the training set.
- **How it Works**:
  - When a new input is encountered, the algorithm looks for the closest or most similar instances in the training data.
  - Predictions are made based on these stored instances (for example, by averaging the target values of the nearest instances).

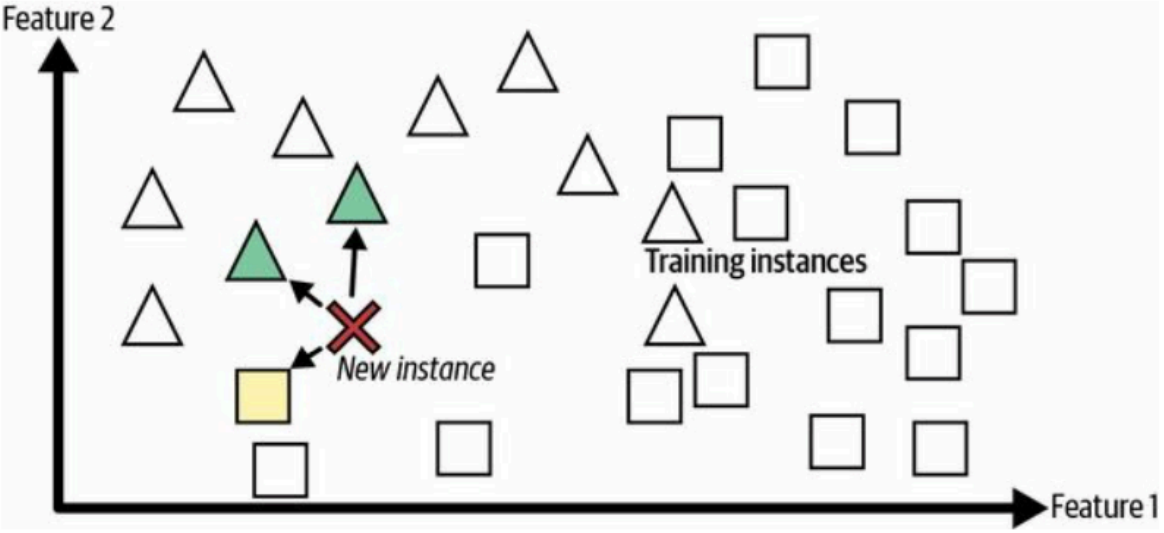

Figure 1-16. Instance-based learning

2. Model-Based Learning

- **Definition**: In model-based learning, the algorithm creates a generalized model during training by finding patterns in the data. This model is then used to make predictions without needing to refer to the training data directly.

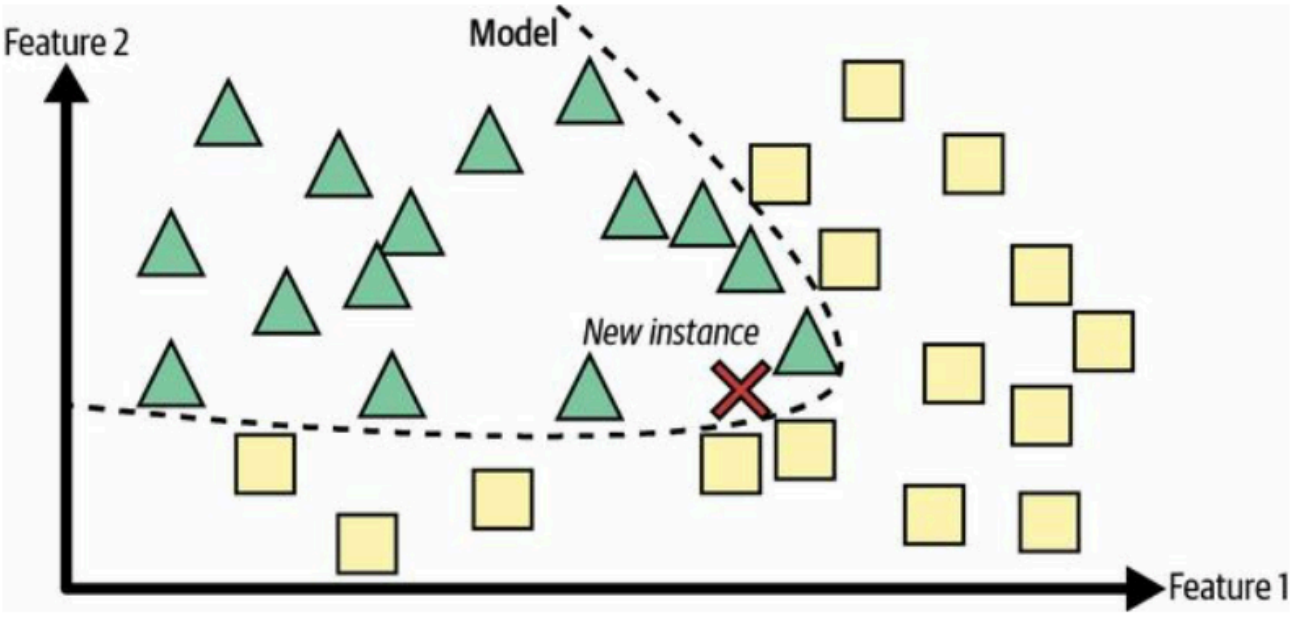

Figure 1-17. Model-based learning

- **How it Works**:
  - The model tries to learn a mapping from input features to target labels by optimizing certain criteria (e.g., minimizing error, maximizing likelihood).
  - Once trained, the model summarizes the knowledge learned from the training data and uses it to make predictions for new inputs.

Main challenges in machine learning

1. Insufficient Quantity of Training Data

- **Challenge**: Machine learning models, especially complex ones like neural networks, require large amounts of data to generalize well. When the training dataset is too small, the model may fail to capture the underlying patterns.
- **Impact**: Small datasets can lead to overfitting (the model memorizes the data instead of learning patterns) or underfitting (the model fails to capture important trends).
- **Solution**:
  - Use **data augmentation** techniques to artificially increase the size of your dataset.
  - Apply **transfer learning**, where a pre-trained model is fine-tuned on your smaller dataset.
  - Generate synthetic data or explore more data sources.

---

2. Non-Representative Training Data

- **Challenge**: If the training data doesn't adequately represent the problem space or the real-world scenarios the model will encounter, the model will struggle to generalize.
- **Impact**: The model performs well on the training data but poorly on new, unseen data due to bias introduced by the training set.
- **Solution**:
  - Ensure that the data collected covers the full range of scenarios, classes, or cases.
  - Apply **sampling techniques** to balance the dataset and prevent biases.
  - Perform **cross-validation** to detect biases early in the modeling process.

---

3. Poor Quality of Data

- **Challenge**: Data often contains noise, errors, missing values, or inconsistencies that can reduce the quality of your model.
- **Impact**: A model trained on poor-quality data may produce inaccurate predictions or misleading patterns, resulting in degraded performance.
- **Solution**:
  - Implement a rigorous **data cleaning** process to handle missing values, outliers, and noisy data.
  - Normalize or standardize data, and perform **data imputation** for missing values.
  - Use **domain expertise** to correct errors and inconsistencies.

---

4. Irrelevant Features

- **Challenge**: Including irrelevant or redundant features (features that don't contribute to predicting the output) can confuse the model and lead to poor performance.
- **Impact**: Irrelevant features can increase model complexity, leading to overfitting and reduced generalization.

- **Solution**:
  - Apply **feature selection** techniques like recursive feature elimination or using domain knowledge to remove irrelevant features.
  - Use **dimensionality reduction** techniques like Principal Component Analysis (PCA) to eliminate redundant information.

---

5. Overfitting the Data

- **Challenge**: Overfitting occurs when the model learns not only the underlying patterns in the training data but also the noise or minor fluctuations, making it too complex.
- **Impact**: The model performs well on the training data but fails to generalize to unseen data, resulting in poor test performance.
- **Solution**:
  - Use **regularization techniques** (e.g., L1/L2 regularization, dropout in neural networks) to penalize overly complex models.
  - Collect more data if possible or reduce model complexity.
  - Apply **cross-validation** to check for overfitting during the training process.
  - Consider **early stopping** during training to avoid excessive fitting.

---

6. Under-fitting

- **Challenge**: Under-fitting occurs when the model is too simple to capture the underlying structure of the data. This often happens if the model isn't complex enough, the training time is too short, or the features used aren't informative.
- **Impact**: The model performs poorly on both the training data and unseen data, indicating that it hasn't learned the relevant patterns.
- **Solution**:
  - Use a more complex model that can capture the intricacies of the data.
  - Improve feature selection and engineering to make the data more informative.
  - Increase training time or tune model hyper-parameters.

Testing and Validation

1. Training Set

- **Definition**: The training set is the portion of the dataset used to train the machine learning model. The model learns the underlying patterns and relationships from this data.
- **Purpose**: To allow the model to learn and adjust its parameters (e.g., weights in neural networks) so that it can make predictions based on input data.
- **Size**: Typically the largest portion of the data, often 60-80% of the total dataset.

- **Usage**: The model sees both the input features and the corresponding labels (for supervised learning), and iteratively updates its parameters to minimize error.

---

2. Validation Set

- **Definition**: The validation set is used during the model training process to tune hyper-parameters and evaluate the model's performance on unseen data before the final test. It acts as a middle ground between the training and test sets.
- **Purpose**: To tune the model, select the best hyper-parameters, and avoid overfitting. It helps in decisions like adjusting learning rates, regularization terms, or the number of layers in a neural network.
- **Size**: Usually 10-20% of the total dataset.
- **Usage**: After each training iteration, the model is evaluated on the validation set to check how well it generalizes to unseen data. This helps in preventing overfitting to the training data.

---

3. Test Set

- **Definition**: The test set is a completely independent portion of the dataset that is not used during training or hyper-parameter tuning. It is used only once the model has been fully trained and tuned.
- **Purpose**: To assess the model’s true performance on new, unseen data and give an unbiased estimate of how well it will generalize to the real world.
- **Size**: Typically 10-20% of the total dataset.
- **Usage**: The test set is only used after training and validation are complete. It serves as the final check for how well the model will perform in deployment.

---

4. Generalization Error

- **Definition**: Generalization error refers to the difference between the model’s performance on the training data and its performance on new, unseen data (such as the validation or test set). It measures how well the model generalizes to unseen data.
- **Low Generalization Error**: If the model performs well on both the training and test sets, it has successfully learned the underlying patterns without overfitting.
- **High Generalization Error**: If the model performs well on the training set but poorly on the test set, it means the model has overfitted the training data and hasn't learned to generalize.
- **Purpose**: To quantify how likely the model is to make accurate predictions on new data in real-world scenarios.

Hyperparameters in Machine Learning

Hyperparameters are parameters whose values are set before the learning process begins, and they control the behavior of the learning algorithm itself. Unlike model parameters (which the model learns from the data during training), hyperparameters are external settings that influence the performance and accuracy of the model

Key Characteristics of Hyperparameters:

1. **Set Before Training**: Hyperparameters must be defined before training the model. They are not learned by the model but are chosen through experimentation or prior knowledge.
2. **Control the Learning Process**: Hyperparameters dictate how the model learns from the data, such as how fast it learns, the model's complexity, and how it handles data.
3. **Affect Model Performance**: The right choice of hyperparameters can significantly improve model performance, while poor choices can lead to underfitting, overfitting, or slow training.

Examples of Common Hyperparameters:

1. **Learning Rate (for optimization algorithms like gradient descent)**:
   - Controls how fast the model updates its parameters.
   - **Low learning rate**: Slow learning but potentially more accurate.
   - **High learning rate**: Faster training but may overshoot the optimal values.
2. **Number of Epochs** (for neural networks):
   - Defines how many times the learning algorithm will work through the entire training dataset.
   - **More epochs**: Can lead to better learning, but too many epochs can cause overfitting.
3. **Batch Size** (for batch learning):
   - The number of training examples the model processes before updating its parameters.
   - **Larger batch size**: More stable updates but slower iteration.
   - **Smaller batch size**: Faster iterations but less stable updates.
4. **Regularization Parameter** (e.g., L1 or L2 regularization):
   - Prevents overfitting by penalizing large weights in the model.
   - **Higher regularization**: Forces the model to simplify (reduce complexity).
   - **Lower regularization**: Allows the model to capture more complex patterns but risks overfitting.
5. **Number of Hidden Layers and Units** (for neural networks):
   - Controls the architecture of a neural network.
   - **More layers/units**: Allows the model to learn more complex representations but increases the risk of overfitting.

Hyperparameter Tuning

Since hyperparameters are set manually and can significantly impact model performance, choosing the right values requires tuning. Common techniques for hyperparameter tuning include:

1. **Grid Search**: Systematically tests all possible combinations of a set of hyperparameters and selects the one with the best performance.
2. **Random Search**: Randomly samples combinations of hyperparameters to find the best configuration.
3. **Bayesian Optimization**: Uses probabilistic models to select hyperparameters more intelligently based on past evaluations.
4. **Cross-Validation**: Evaluates the model's performance for different hyperparameter values by splitting the data into several folds and using each fold for validation at least once.