**CHAPTER 1: The Machine Learning Landscape**

This chapter provides a high-level overview of fundamental Machine Learning concepts and jargon, without much code, serving as a map before exploring the ML continent.

**What Is Machine Learning?**

* Machine Learning (ML) is defined as the science and art of programming computers to learn from data.
* Tom Mitchell (1997) provided an engineering-oriented definition: A computer program learns from experience E with respect to task T and performance measure P if its performance on T, as measured by P, improves with experience E.
* **Example: Spam Filter**
  + A spam filter is an ML program that learns to flag spam using examples of spam and regular (ham) emails.
  + The examples used for learning are the **training set**.
  + Each example in the training set is a **training instance** (or sample).
  + The task (**T**) is to flag spam for new emails.
  + The experience (**E**) is the training data.
  + The performance measure (**P**) must be defined, such as the ratio of correctly classified emails (called **accuracy** for classification tasks).

**Why Use Machine Learning?**

* ML allows systems to automatically learn patterns from data, unlike traditional programming which requires writing long lists of complex rules.
* For example, a traditional spam filter requires manual updates for new spam patterns like "For U" replacing "4U". An ML-based spam filter automatically notices such changes and adapts.
* ML is effective for problems that are too complex for traditional approaches or have no known algorithm, such as speech recognition.
* Applying ML to large amounts of data can uncover hidden patterns, a process called **data mining**.
* **Machine Learning is great for:**
  + Problems where solutions require extensive manual tuning or complex rules.
  + Complex problems with no good traditional solution.
  + Fluctuating environments, as ML systems can adapt to new data.
  + Gaining insights from complex problems and large datasets.

**Types of Machine Learning Systems**

ML systems can be classified based on various criteria:

1. **Supervision during training:**
   * **Supervised learning:** The training data includes desired solutions, called **labels**.
     + Tasks include **classification** (e.g., spam filter - spam or ham) and **regression** (predicting a target numeric value, e.g., car price, given features/predictors).
     + Some regression algorithms (like Logistic Regression) can also be used for classification by outputting probabilities.
     + Important supervised algorithms: k-Nearest Neighbors, Linear Regression, Logistic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests, Neural networks.
   * **Unsupervised learning:** Training data is unlabeled; the system learns without a teacher.
     + Tasks include **Clustering** (grouping similar instances, e.g., customer segmentation). Algorithms: K-Means, DBSCAN, Hierarchical Cluster Analysis (HCA).
     + **Anomaly detection** and **novelty detection** (learning what "normal" data looks like to detect abnormal instances). Algorithms: One-class SVM, Isolation Forest.
     + **Visualization** and **dimensionality reduction** (simplifying data without losing much information). **Feature extraction** (merging correlated features) is one way to do this. Algorithms: Principal Component Analysis (PCA), Kernel PCA, Locally-Linear Embedding (LLE), t-distributed Stochastic Neighbor Embedding (t-SNE).
     + **Association rule learning** (finding relationships between items). Algorithms: Apriori, Eclat.
   * **Semisupervised learning:** Algorithms combine unsupervised and supervised techniques, often using unlabeled data and a small amount of labeled data. Figure 1-11 shows this. Deep Belief Networks (DBNs) are an example.
   * **Reinforcement Learning:** A learning system (agent) observes an environment, selects actions, gets rewards/penalties, and learns a **policy** (strategy) to maximize reward over time. Examples: robots learning to walk, AlphaGo playing Go.
2. **Incremental learning capability:**

**Batch learning:** The system is trained using all available data offline and cannot learn incrementally. To incorporate new data, a new system version must be trained from scratch on the full dataset. This process can be automated. In **batch learning**, we use .fit() once on all data.

**Online learning:** The system is trained incrementally by feeding instances sequentially or in mini-batches, learning on the fly. It can adapt to rapidly changing data. If data is huge, online learning can be an alternative to splitting batch work across servers. In **online learning**, we use .partial\_fit() repeatedly on small chunks, mimicking a stream of data.

1. **Generalization method:**
   * **Instance-based learning:** The system learns examples by heart and generalizes to new cases by comparing them to learned instances using a similarity measure.
   * **Model-based learning:** The system builds a model from the training examples and uses that model to make predictions. This involves studying the data, selecting a model, **training the model** (finding parameters minimizing a cost function), and then using the model for prediction (inference).

**Main Challenges of Machine Learning**

Problems can arise from "bad data" or "bad algorithm".

* **Bad Data:**
  + **Insufficient Quantity of Training Data:** More data is generally needed than humans require to learn concepts.
  + **Nonrepresentative Training Data:** Training data must be representative of the cases to which you want to generalize. Nonrepresentative data can lead to inaccurate models.
  + **Poor Quality Data:** Training data containing errors, outliers, or noise can hinder learning. Requires data cleaning. Handling missing feature values is an example.
  + **Irrelevant Features:** The training data needs enough relevant features and not too many irrelevant ones ("garbage in, garbage out"). **Feature engineering** is crucial for finding good features, involving feature selection, feature extraction (dimensionality reduction algorithms can help), and creating new features.
* **Bad Algorithm:**
  + **Overfitting the Training Data:** The model performs well on the training data but generalizes poorly to new instances (like a human overgeneralizing from a single bad experience).
    - Solutions include: selecting a simpler model, reducing the number of features, or constraining the model using **regularization**.
  + **Underfitting:** (Implicitly a problem mentioned in relation to overfitting and regularization, although not explicitly listed as a main challenge in the source text provided). A model that is too simple or too regularized may fail to capture the underlying patterns in the data.

**Evaluating and Fine-Tuning**

* Evaluating the performance of a trained model is a critical step.
* Tuning **hyperparameters** (parameters of the learning algorithm itself, set prior to training, not affected by the algorithm) is an important part of building an ML system.

**Typical ML Project Workflow (as introduced in Chapter 1)**

* Gather data into a **training set**.
* Feed the training set to a learning algorithm.
* If **model-based**, the algorithm tunes parameters to fit the model to the training set.
* If **instance-based**, the algorithm learns examples and generalizes by comparing new instances.
* Evaluate and fine-tune the system.
* Make predictions on new cases.

**What is bias-variance trade-off?**

**Bias = error from assumptions, Variance = error from sensitivity. Balance both.**