Intro to Machine Learning

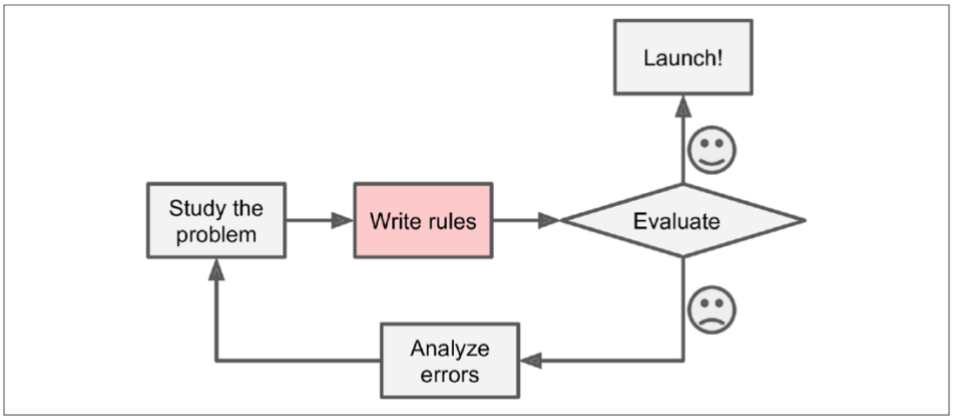
Summary of the The Machine Learning Landscape:

What Is Machine Learning?

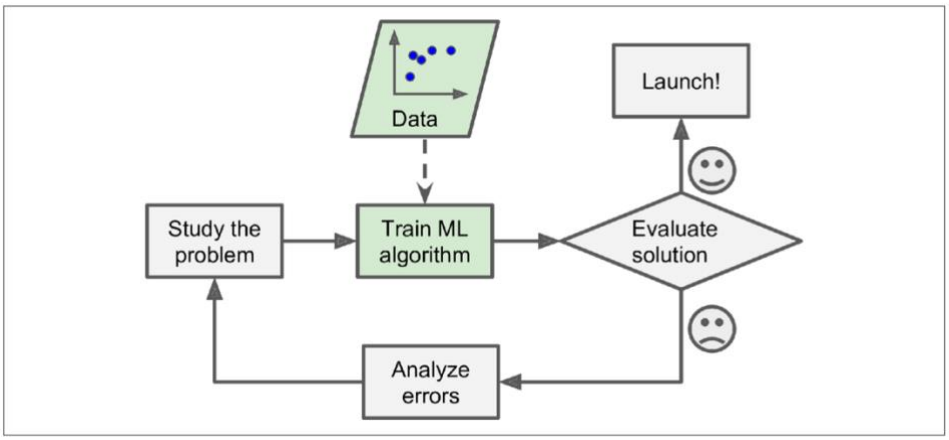
Machine Learning is the science (and art) of programming computers so they can *learn from data*.

Ex: Spam filter: ML program that, given examples of spam emails (e.g., flagged by users) and examples of regular (nonspam, also called "ham") emails, can learn to flag spam. The examples that the system uses to learn are called the *train- ing set*. Each training example is called a *training instance* (or *sample*). In this case, the task *T* is to flag spam for new emails, the experience *E* is the *training data*, and the performance measure *P* needs to be defined; for example, you can use the ratio of correctly classified emails.

Why Use Machine Learning?



The traditional approach, if the problem is difficult, your program will likely become a long list of complex rules—pretty hard to maintain



Can be automated, no need for writing rules forever. ML techniques automatically notices that "For U" has become unusually frequent in spam flagged by users

Machine Learning is great for:

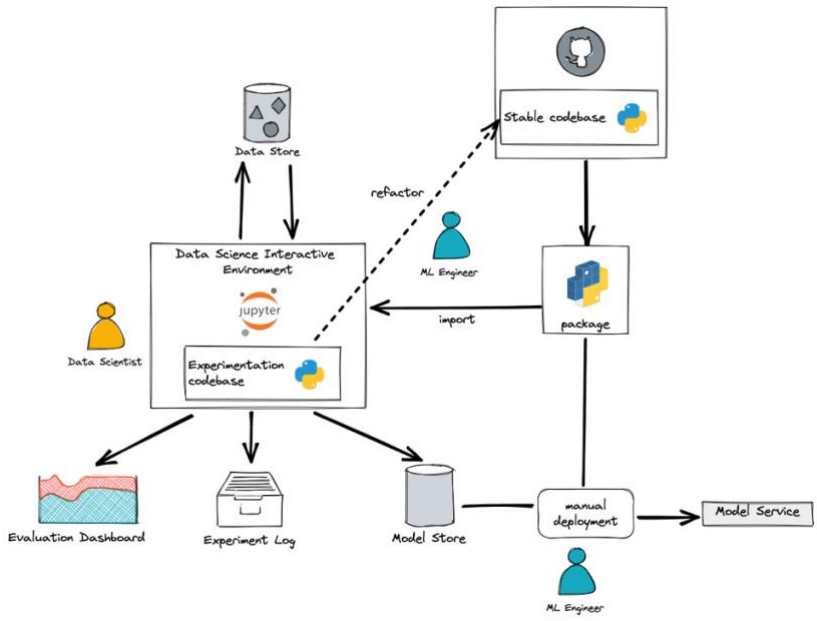
- Problems for which existing solutions require a lot of fine-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.
- Complex problems for which using a traditional approach yields no good solution: the best Machine Learning techniques can perhaps find a solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

Examples of it:

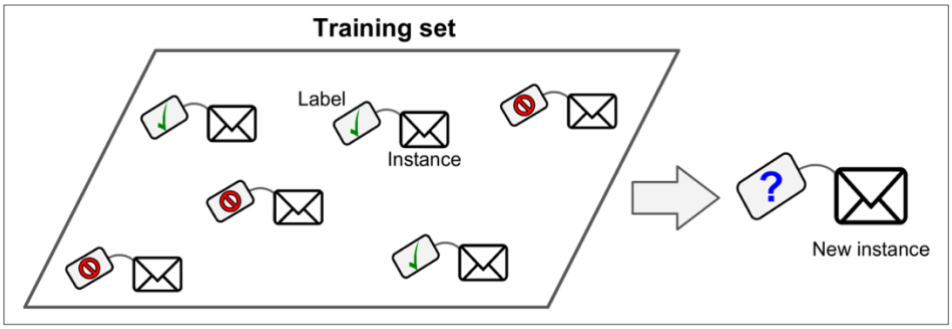
- *Automatically classifying news articles – NLP*
- *Automatically flagging offensive comments on discussion forums – NLP*
- *Detecting credit card fraud- A*nomaly detection

Types of Machine Learning Systems

- Human supervision: (supervised, unsuper- vised, semisupervised, and Reinforcement Learning)
- Learn incrementally on the fly (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model-based learning)



Supervised/Unsupervised Learning



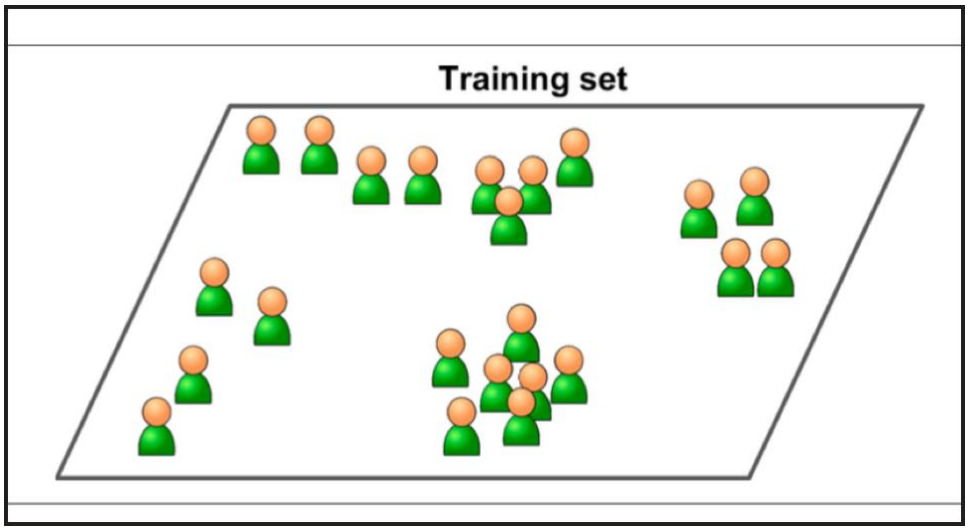
- In *supervised learning*, the training set you feed to the algorithm includes the desired solutions, called *labels*

Most important algoritms:

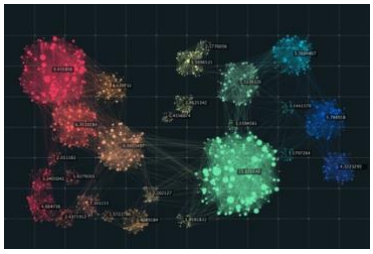
- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks[2]

Unsupervised learning

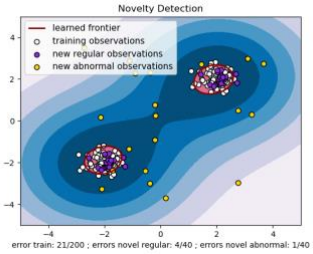
- In *unsupervised learning*, as you might guess, the training data is unlabeled



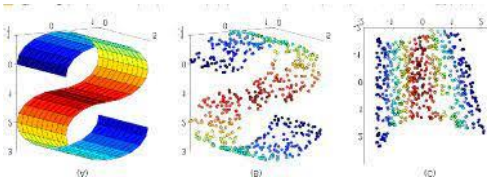
Important algorithms:



Clustering
—K-Means
—DBSCAN
—Hierarchical Cluster Analysis (HCA)



Anomaly detection and novelty detection
— One-class SVM
— Isolation Forest



Visualization and dimensionality reduction
— Principal Component Analysis (PCA)
— Kernel PCA
— Locally Linear Embedding (LLE)
— t-Distributed Stochastic Neighbor Embedding (t-SNE)

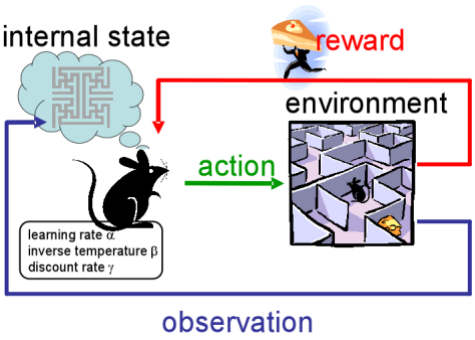
Association rule learning
— Apriori
— Eclat

Semisupervised learning:

Mobile clustering, your phone recognizes people, you label them

Reinforcement Learning

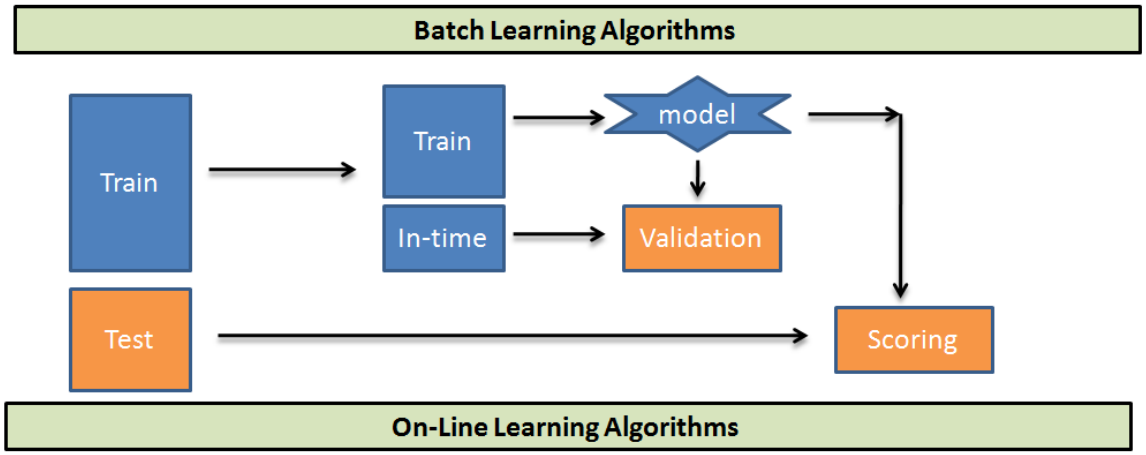
Learns through penalites, to get the policy or best strategy. Ex: AI playing videogames

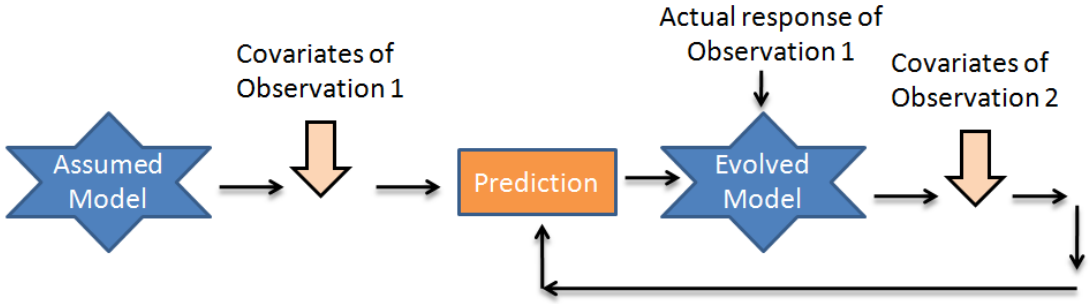


Batch and Online Learning

Batch Learning: Trains on the entire dataset at once, requires significant resources, and cannot learn incrementally. Once deployed, it does not update its learning unless retrained with the complete new dataset.

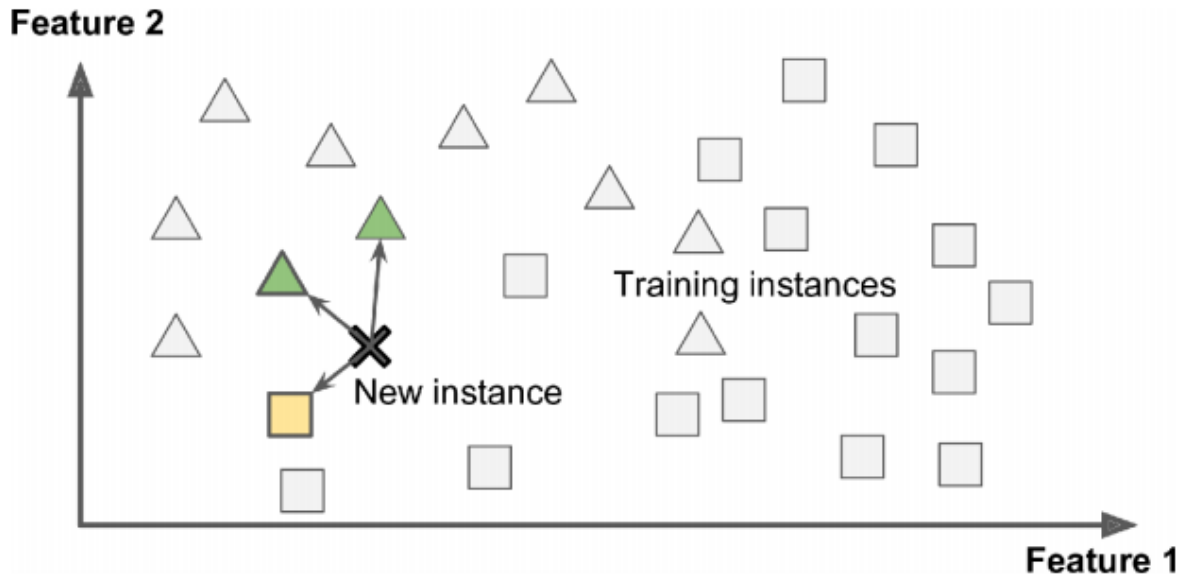
Online Learning: Learns incrementally from new data as it arrives, suitable for environments with continuous data flow. It's resource-efficient but sensitive to the quality of incoming data, requiring ongoing monitoring and adjustments.





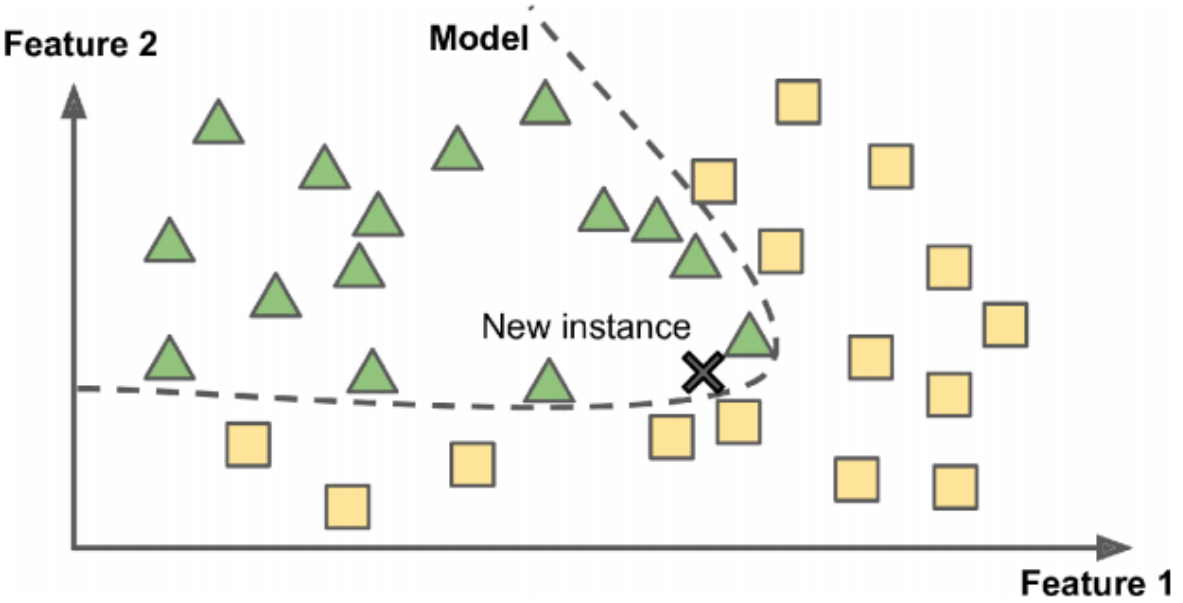
Instance-Based Versus Model-Based Learning

Instance-Based Learning: This method memorizes specific examples and uses them for future predictions based on similarity. For example, a spam filter learns by storing known spam emails and flags new emails similar to these examples. It's like identifying an animal by comparing it to similar ones in a photo album. EX: KNN



Model-Based Learning: This approach creates a general model from data to make predictions. For instance, a model predicting life satisfaction based on GDP per capita analyzes past data to find a pattern, then applies this pattern to predict new cases. It's like using a formula to estimate house prices based on features like size and location.

EX: Linear Regression



Challenges of Machine Learning

Insufficient Quantity of Training Data: Machine Learning algorithms often require large amounts of data. While a toddler might learn to recognize an apple with a few examples, machine learning systems might need thousands or millions of examples, especially for complex tasks like image or speech recognition. More data generally leads to better performance

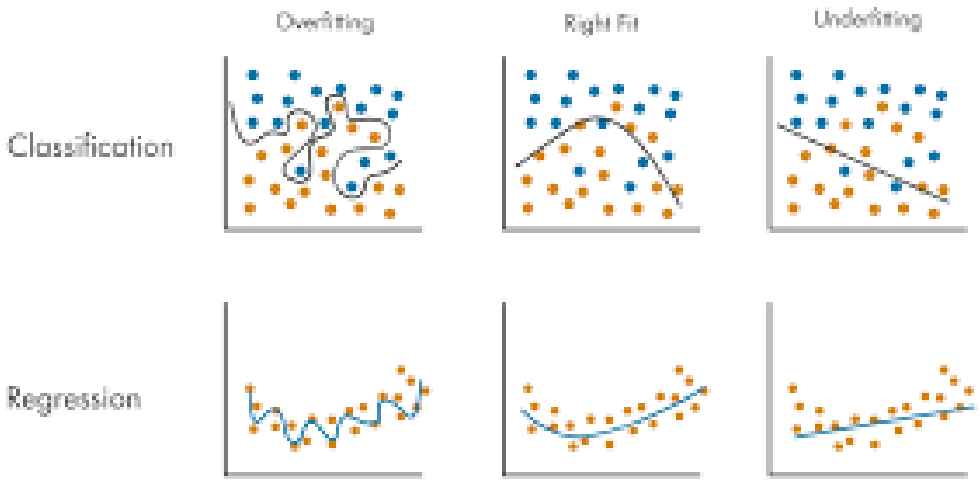
Nonrepresentative Training Data: For effective generalization, the training data must accurately represent the full scope of the problem. Nonrepresentative data leads to models that perform poorly on real-world data. An example is a linear model trained on incomplete country data for predicting happiness, which fails for very poor or very rich countries (too simplistic).

Poor-Quality Data: Data full of errors, outliers, biased and noise hampers the learning process, making it difficult for algorithms to discern underlying patterns. Cleaning and preprocessing the data is often a necessary step.

Irrelevant Features: Successful machine learning depends on identifying and using relevant features while excluding irrelevant ones. This process, feature engineering, involves feature selection, extraction, and sometimes creating new features.

Overfitting the Training Data: This happens when a model is too complex and learns the noise in the data rather than the actual pattern. It performs well on training data but poorly on new, unseen data. Solutions include simplifying the model, getting more data, and reducing noise in the data.

Underfitting the Training Data: This occurs when a model is too simple to capture the complexity of the data, resulting in poor performance even on the training data. Solutions include selecting a more powerful model, improving feature engineering, or reducing constraints on the model.

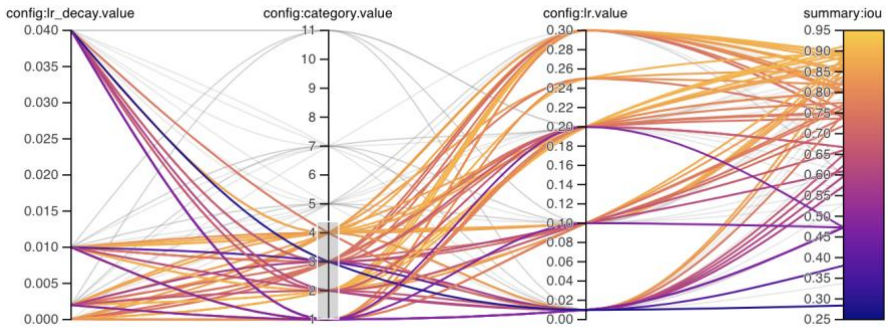


Testing and Validating

The primary method to assess a model's generalization is by using a separate test set. Common practice is to split data into a training set (used to train the model) and a test set (used to evaluate the model). The error on the test set is the generalization error. If a model performs well on training data but poorly on test data, it's likely overfitting.

Data Splitting Ratios: Typically, 80% of data is used for training and 20% for testing. The exact split can vary depending on the dataset's size, the more data the smaller the testing set can be.

Hyperparameter Tuning and Model Selection: To choose between models or set hyperparameters, holdout validation is used. This involves holding back part of the training set as a validation set. Models are trained on the reduced training set and evaluated on the validation set. The best-performing model is then retrained on the full training set.



Cross-Validation: For more accurate model evaluation, especially when the validation set is small, repeated cross-validation is used. It involves multiple rounds of training and validation with different subsets. However, this increases training time.

No Free Lunch Theorem: This theorem posits that no single model is universally superior. Model choice depends on the specific data and task. Assumptions about the data guide the selection of a reasonable set of models to evaluate.