Introduction to ML

DS 8015

Outline

- What is Machine Learning
- Why using Machine Learning
- Examples of Applications
- Types of Machine Learning Systems
- Main Challenges of Machine Learning
- Testing and Validating

What is Machine Learning

What is Machine Learning

- 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.

—Arthur Samuel, 1959

- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

—Tom Mitchell, 1997

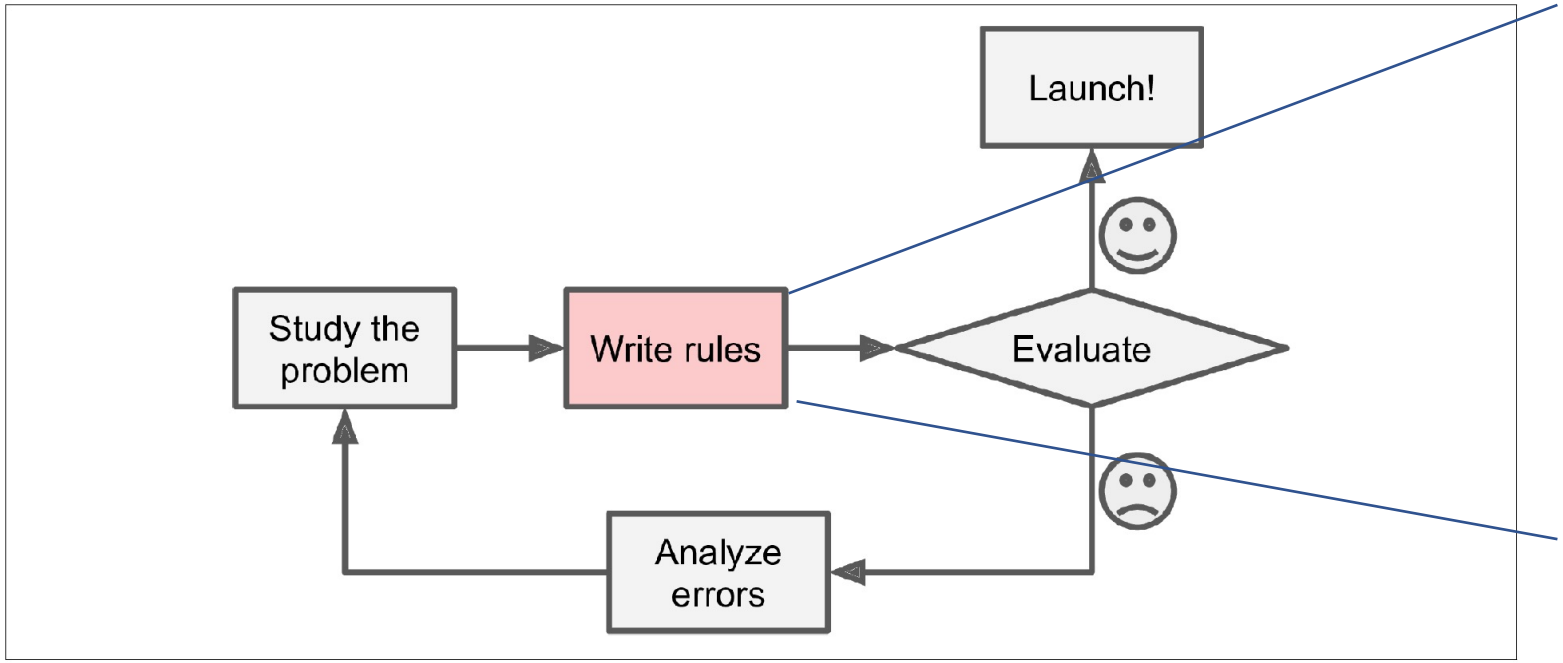
What is Machine Learning

Spam filter is a Machine Learning program that, given examples of spam emails (e.g., flagged by users) and examples of regular (non-spam, also called "ham") emails, can learn to flag spam.

- "**Given examples**" ➔ "training set": the system uses to learn
- Each **training example** ➔ "a training instance" (or sample).
- The **task T**: to flag spam for new emails,
- The **experience E**: the training data,
- The **performance measure P** needs to be defined;
  - E.g. the **ratio of correctly classified emails** (called "accuracy"), often used in **classification tasks**.

Why Using Machine Learning

Why Using Machine Learning

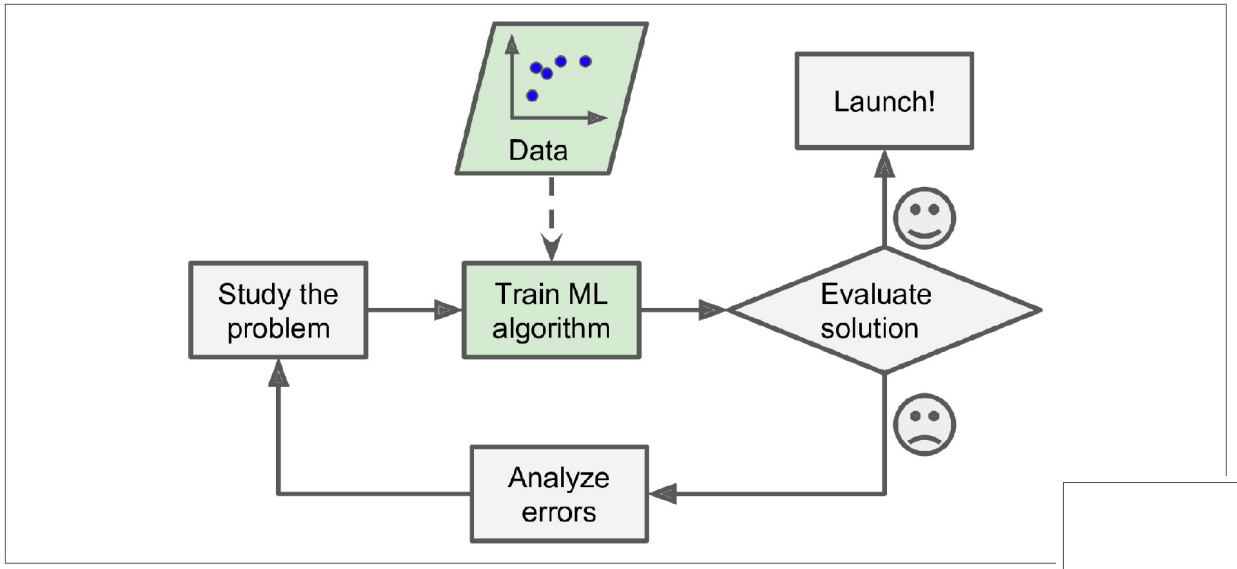


Traditional Approach

For difficult problems the list of rules will become very long and complex.
➔ Not Maintainable

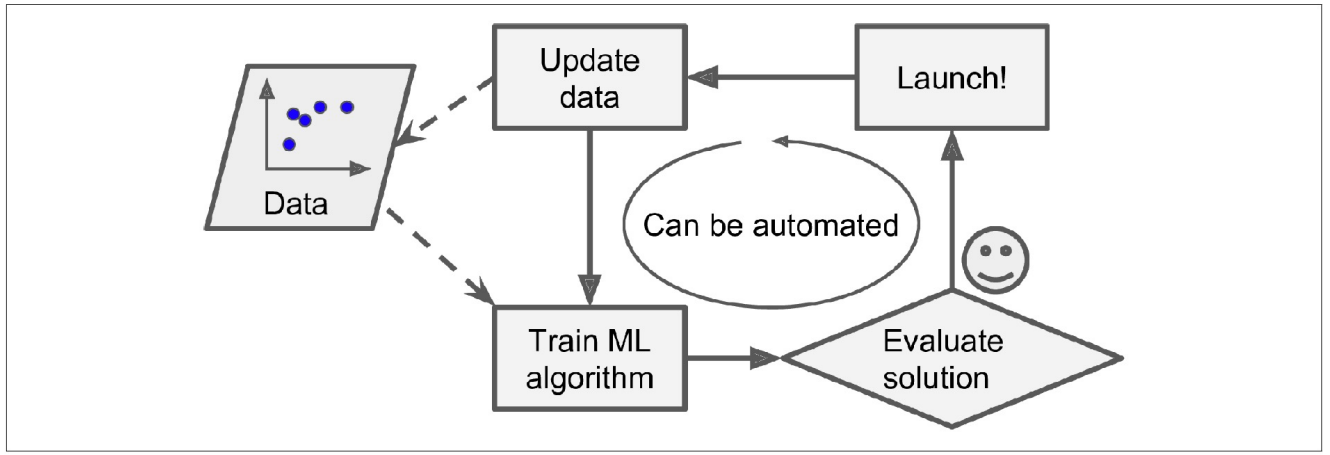
When spammers notice that all emails containing “4U” are blocked and change it to “For U”?
➔ You will write rules forever

Machine Learning Approach

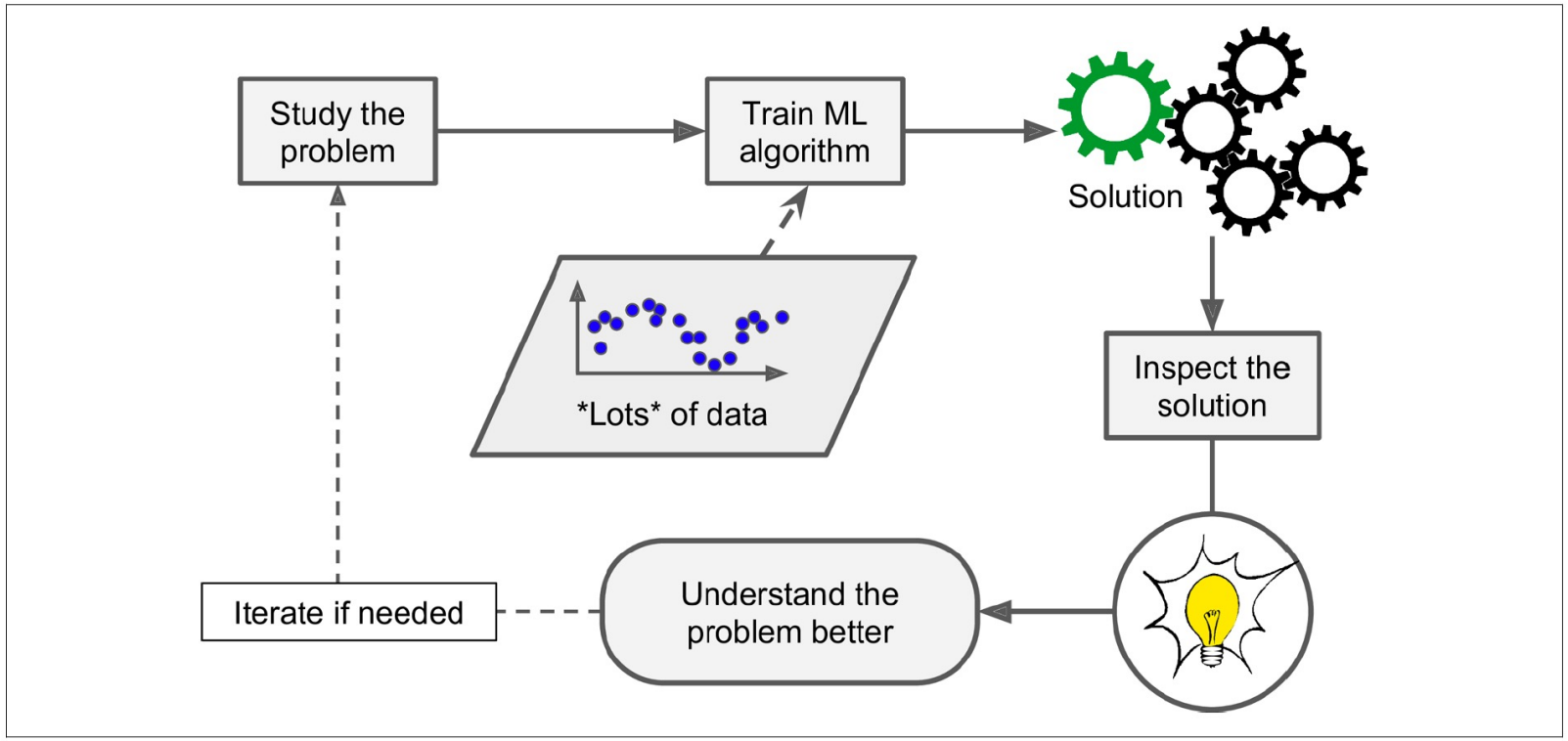


A ML based spam filter automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples compared to the ham examples

ML learning can be automated



ML Can Help Humans Learn



ML is Great For

- Problems for which existing solutions require a lot of fine-tuning or long lists of rules.
- Complex problems for which using a traditional approach yields no good solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

Examples of Applications

Examples of Applications

- Analyzing images of products on a production line to automatically classify them: **image classification**.
- Detecting tumors in brain scans: **semantic segmentation** (each pixel in the image is classified as we want to determine the exact location and shape of tumors).
- Automatically classifying news articles: **natural language processing** (NLP)
- Automatically flagging offensive comments on discussion forums: **text classification** (same NLP)
- Summarizing long documents automatically: **text summarization** (NLP)
- Creating a chatbot or a personal assistant: **natural language understanding** (NLU) and **question-answering**

Examples of Applications

- Forecasting your company's revenue next year, based on many performance metrics: **regression**
- Making your app react to voice commands: **speech recognition**
- Detecting credit card fraud or network traffic change: **anomaly detection**
- Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment: **clustering**
- Representing a complex, high-dimensional dataset in a clear and insightful diagram: **data visualization**
- Recommending a product that a client may be interested in, based on past purchases: **recommender system**
- Building an intelligent bot for a game: usually solved with **reinforcement learning**

Types of Machine Learning Systems

Types of Machine Learning Systems

Many different types of ML systems. Can be categorized using the criteria:

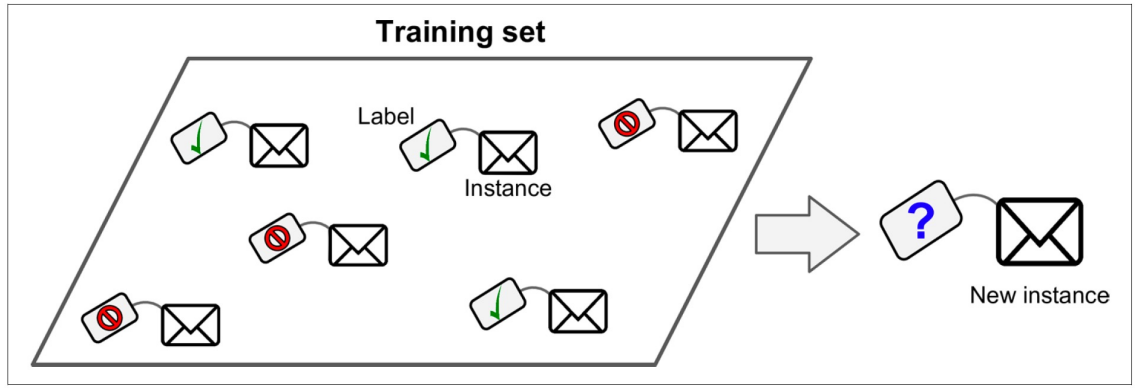
- Whether or not they are **trained with human supervision** (supervised, unsupervised, semisupervised, and Reinforcement Learning)
- Whether or not they **can learn incrementally on the fly** (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead by **detecting patterns** in the training data and building a predictive model (instance-based versus model-based learning)

Supervised/Unsupervised Learning

4 major learning categories

- Supervised
- Unsupervised
- Semisupervised
- Reinforcement

Supervised Learning



- The training set fed to the algorithm includes the desired solutions (*labels*)
- A typical supervised learning task is **classification** (e.g. spam filter)
- Another typical task is **regression:** predicting a target numeric value given a set of features (*predictors*)
- Some regression algorithms can be used for classification as well (e.g. Logistic Regression)

Supervised Learning

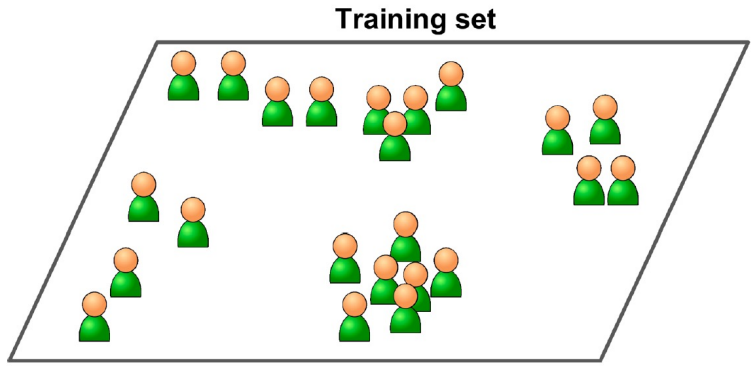
Regression problem: to predict a value given an input feature (or multiple features)

Some important supervised learning algorithms

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural Networks

Unsupervised Learning

The training data is unlabeled

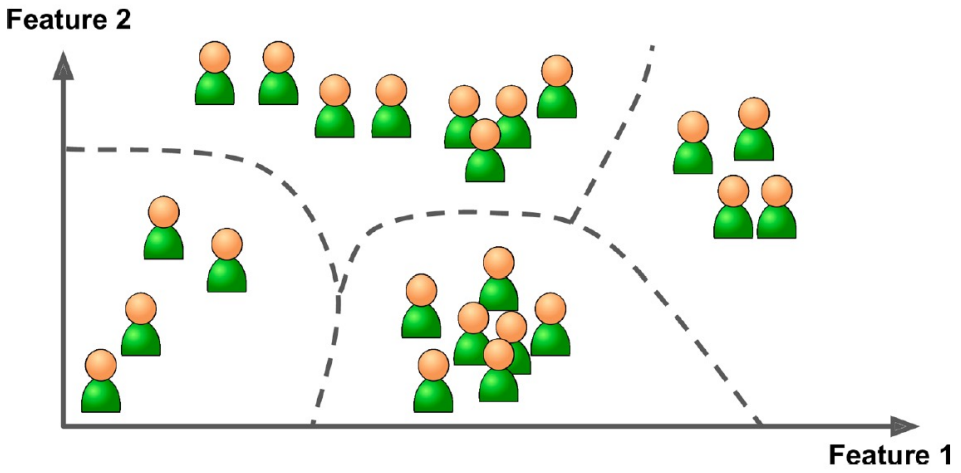


Some of the most important unsupervised learning algorithms

- Clustering
- Anomaly detection and novelty detection
- Visualization and dimensionality reduction
- Association rule learning

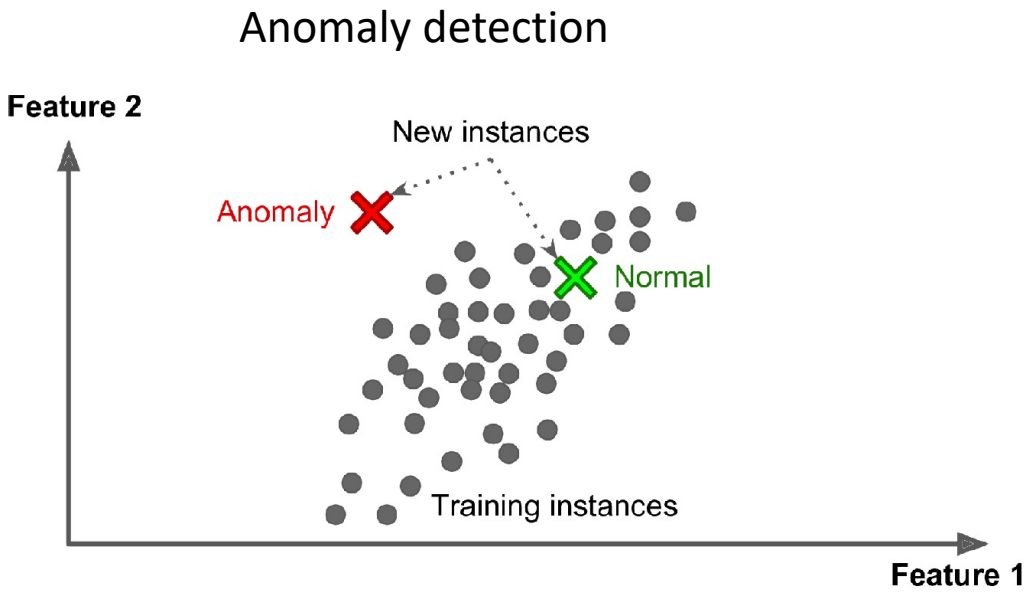
Unsupervised Learning – Some Algorithms

Detecting groups of similar visitors



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

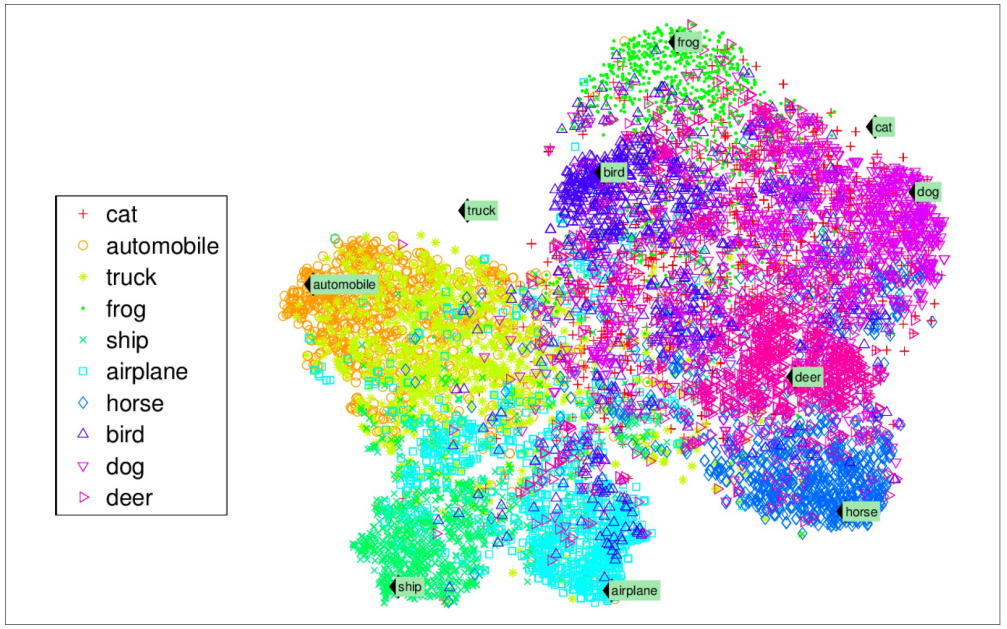
Unsupervised Learning – Some Algorithms



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

Unsupervised Learning – Some Algorithms

2D or 3D representation of data



t-SNE visualization highlighting semantic clusters

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

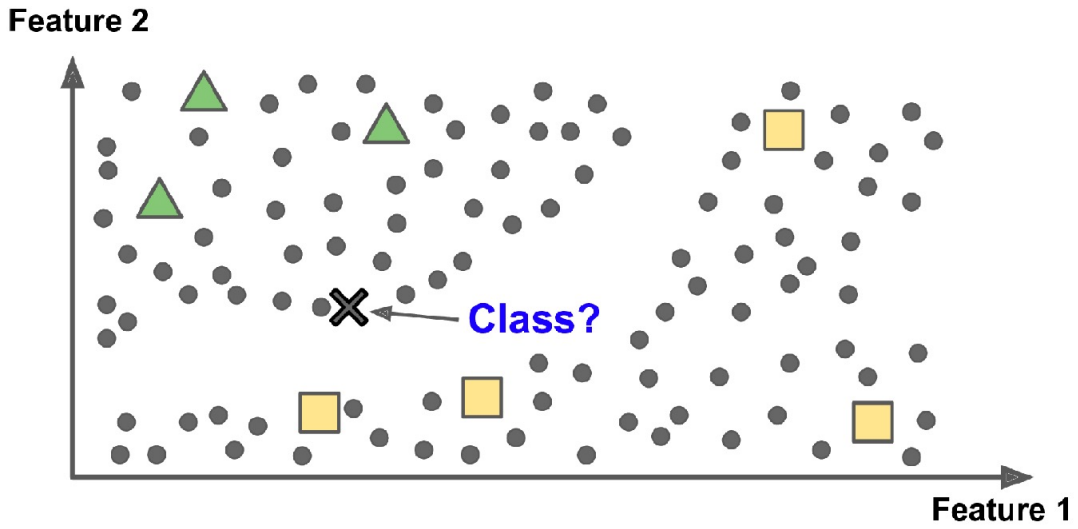
Unsupervised Learning – Some Algorithms

Goal: to dig into large amounts of data and discover interesting relations between attributes.

E.g., Running an association rule on your sales logs may reveal that people who purchase barbecue sauce and potato chips also tend to buy steak. Thus, you may want to place these items close to one another.

- Association rule learning
  - Apriori
  - Eclat

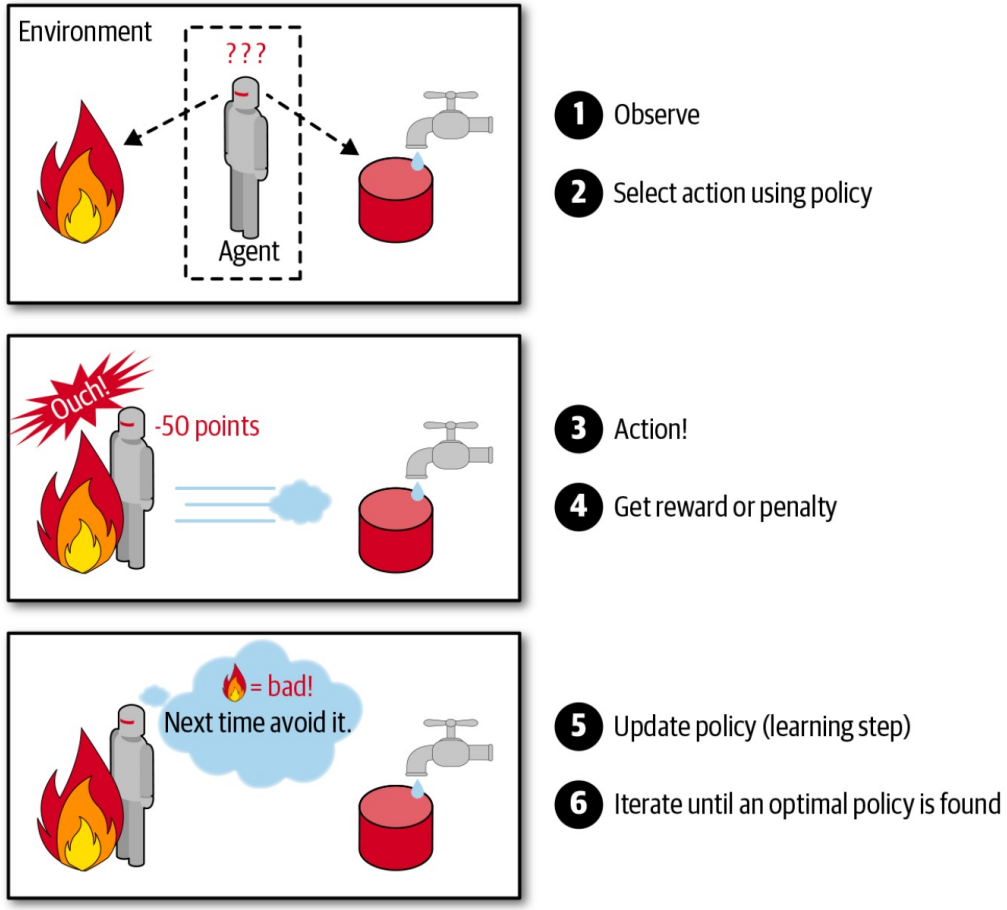
Semisupervised Learning



Two classes (squares and triangles)

- Labeling data is time consuming and expensive. There will be **many unlabeled** instances and **few labeled** instances
- E.g. Google Photos. Once photos are uploaded and some persons labelled it start recognizing the persons in other photos

Reinforcement Learning



- The learning system (*an agent*) in this context, can **observe** the environment, select and perform **actions**, and get **rewards** in return (or **penalties** in the form of negative rewards)
- Must learn by itself what is the best strategy (*a policy*) to get the most reward over time.
- A policy defines what action the agent should choose when it is in a given situation.

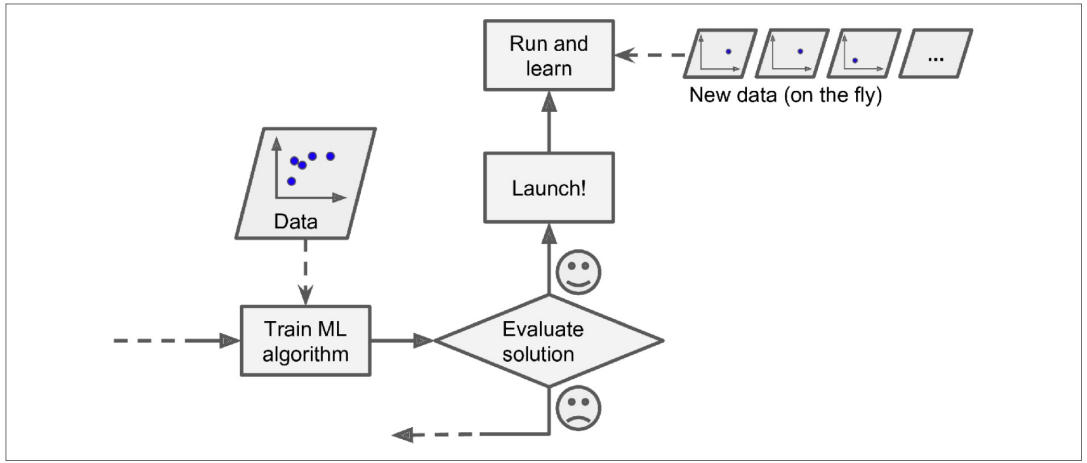
Batch/Online Learning

Batch Learning

- The system is incapable of learning incrementally: it must be trained using all the available data (lots of time and computing resources ➔ typically done offline)
- The whole process of training, evaluating, and launching an ML system can be automated (therefore even a batch learning can adapt to change)

Batch/Online Learning

Online Learning



An important parameter: **Learning Rate**
High learning rate:
- System rapidly adapts to new data
- Tend to quickly forget old data

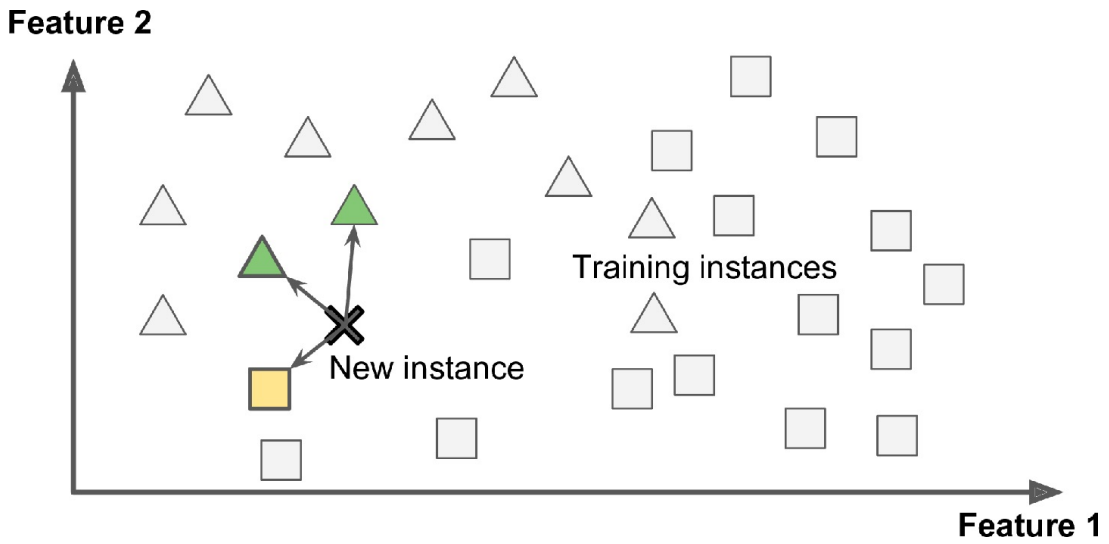
Low learning rate:
- System will have more inertia (learn slowly)
- Less sensitive to noise or to outliers in the data

- The system incrementally trained by feeding data instances sequentially (**individually** or in small groups called **mini-batches**)
- Good for
  - Systems that receive data as a continuous flow (e.g. stock prices) and need to adapt to change rapidly or autonomously
  - When you have limited computing resources

Instance-Based vs Model-Based Learning

- How the ML systems generalize
- Given a number of training examples, the system needs to be able to make good predictions for (**generalize to**) examples it has never seen before.
- Having a good performance measure on the training data is good, but insufficient; the true goal is to **perform well on new instances**.
- Two main approaches to generalization:
  - Instance-based learning,
  - Model-based learning.

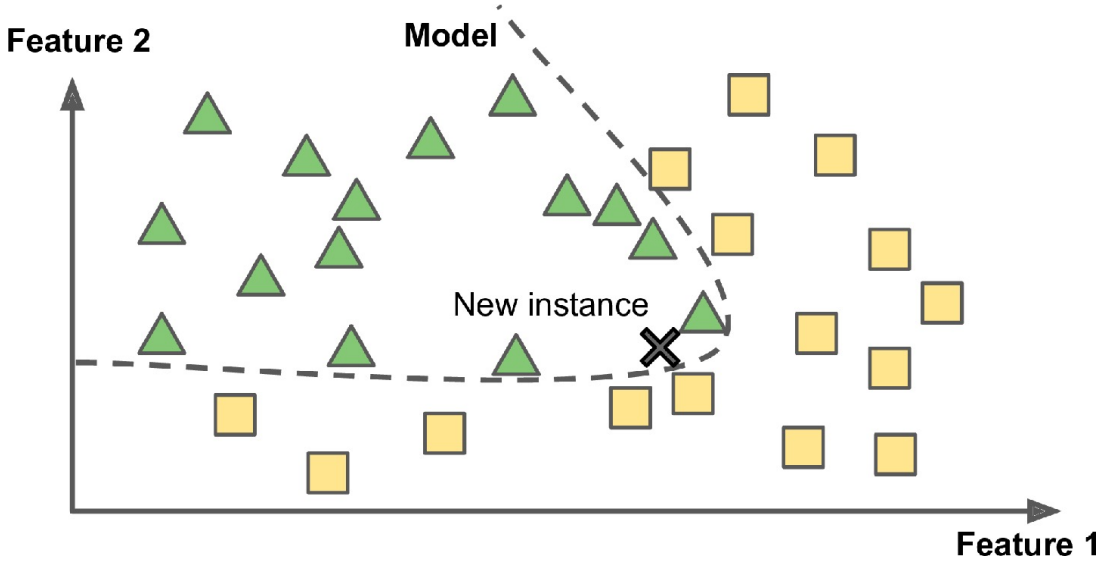
Instance-Based Learning



New instance classified as **triangle** because the majority of the most similar instances belong to that clas

- The system learns the examples by heart, then generalizes to new cases by using a **similarity measure** to compare them to the learned examples
- E.g. similarity measure between two emails can be a count of words they have in common

Model-Based Learning



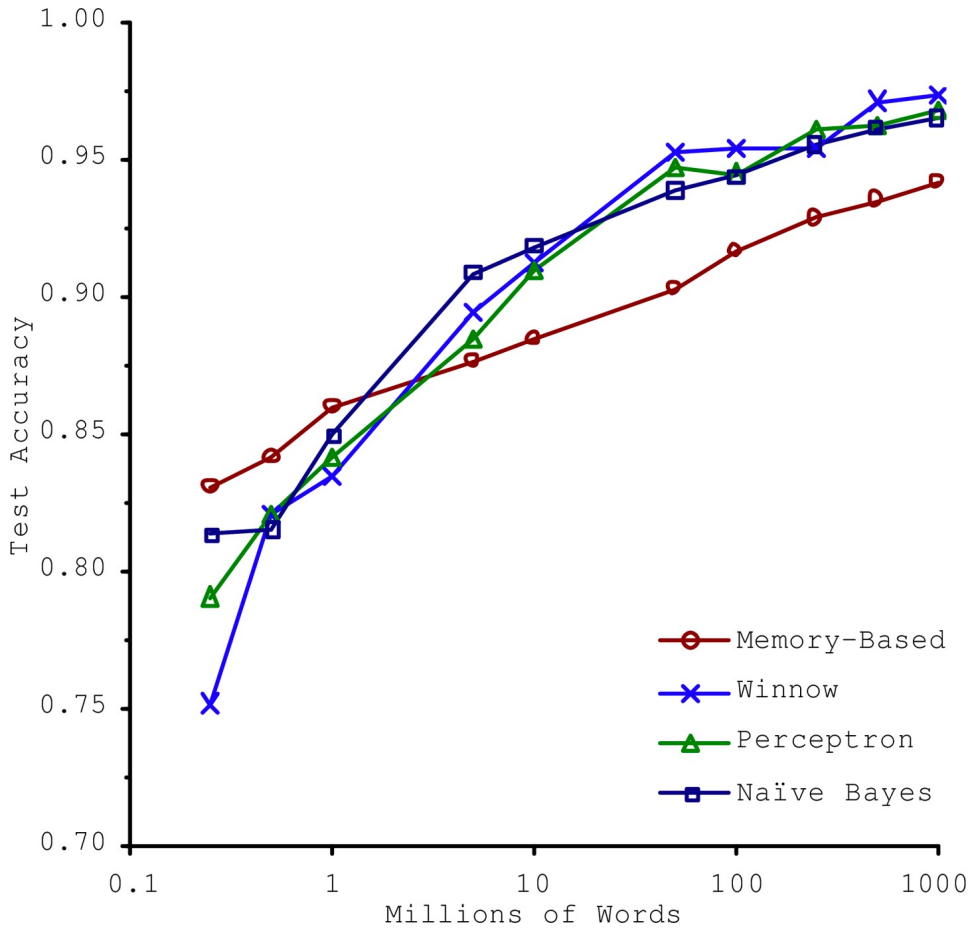
- Building a model of the examples and then to use that model to make predictions

Main Challenges of ML

Main Challenges of ML

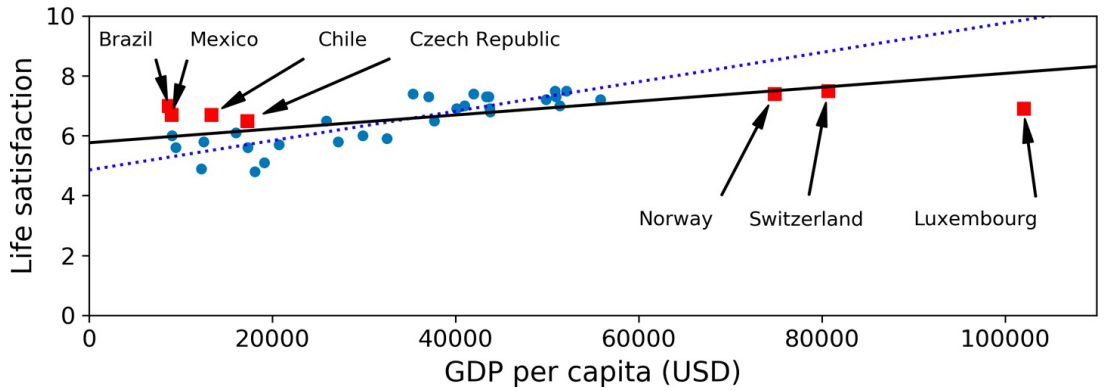
- Insufficient quantity of training data
- Nonrepresentative training data
- Poor-quality data
- Irrelevant features
- Overfitting the training data
- Underfitting the training data

Insufficient Quantity of Training Data



- In 2001, M Banko and E Brill showed that very different ML algorithms performed almost identically well on a complex problem of natural language disambiguation once they were given enough data

Nonrepresentative Training Data



- In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to
- Without the red data points the dotted line is the trained model. With the red data points added the model changes to the solid line.
- If the sample is too small, there will be a **sampling noise**
- Even very large samples can be nonrepresentative if the sampling method is flawed (called **sampling bias**)

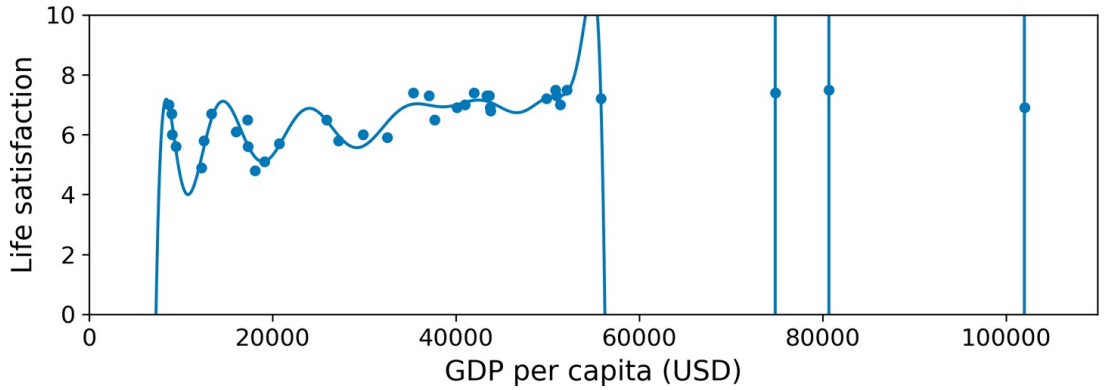
Poor Data Quality

- If the training data is full of errors, outliers, and noise it is harder for the system to detect the underlying pattern.
- Cleaning up the training data will improve the performance
- When to clean up:
  - Some instances are clearly outliers
  - Some instances are missing a few features

Irrelevant Features

- The system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones
- Feature engineering (coming up with a good set of features to train on) involves:
  - Feature selection (selecting the most useful features)
  - Feature extraction (combining existing features to produce a more useful one)
  - Creating new features by gathering new data

Overfitting the Training Data



High-degree polynomial life satisfaction model overfitting the training data

- The model performs well on the training data, but it does not generalize well

Underfitting the Training Data

- Occurs when the model is too simple to learn the underlying structure of the data
- Main options to fix the problem:
  - Selecting a more powerful model, with more parameters
  - Feeding better features to the learning algorithm (feature engineering)
  - Reducing the constraints on the model

Testing & Validating

Testing & Validating

- A way to know how well a model will generalize to new cases is to try it out on new cases
- A better option is to split the data into two sets: **training set** and **test set**.
  - Training set is used to train the model, test set is used to test the performance
  - The error rate on new cases is called the **generalization error** (or **out-of-sample** error)
  - If training error is low but generalization error is high → overfitting the training data

Hyperparameter Tuning and Model Selection

- How to decide which model to use?
  - One option: train multiple models and compare how well they generalize using the test set
- One model generalizes better but you want to apply some regularization to avoid overfitting. How to choose the value of the regularization parameter?
  - Option: train 100 different models using 100 different values for this hyperparameter. Select the best performing model and hyperparameter that generalizes with lowest error.
  - May not perform good in production: because model and parameters are chosen to perform the best on the particular test set.