Marching learning – Study Note

What is Machine Learning?

Marching Learning is the science of programming computers so they can learn from data.

Most machine learning is about making predictions. This means that given a number of training examples, the system needs to be able to generalize to examples it has never seen before.

example:

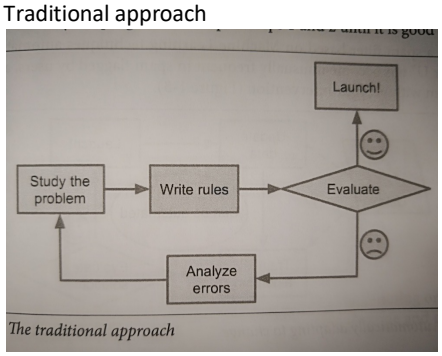
Spam filter

A computer program (Spam filter) is said to learn from experience E (Training data set) with respect to some task T (flag spam) and some performance measure P (Ratio of accuracy), if its performance on T, as measured by P, improves with experience E.

Why use Machine Learning?

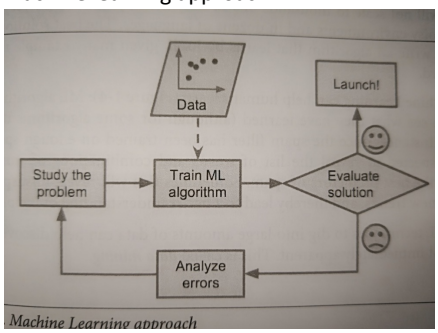
Problems for which existing solutions requires a lot of hand-tuning or long lists of rules.	eg: Spam filter
Complex problems for which there is no good solution at all using traditional approach.	eg: NLP
Fluctuating environments.	A ML system can adapt to new data.
Getting insights about complex problems and large amounts of data	To help humans learn

Traditional approach



The traditional approach

Machine learning approach



Machine Learning approach

Types of Machine Learning Systems

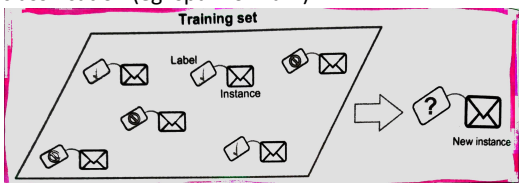
Some criteria to define ML categories

1. Supervised vs. Unsupervised: Whether or not they are trained with human supervision
2. Online vs. Batch: Whether or not they can learn incrementally on the fly
3. Instance-based vs. Model-based: Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training dat and build a predictive model, much like scientists do.

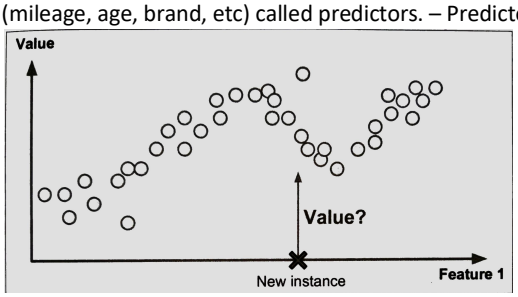
<u>Supervised vs. Unsupervised</u>

**According to the amount and type of supervision they get during training.*

- ***Supervised***
  - (1) The training data you feed to the algorithm includes the desired solutions, called Labels
  - (2) The typical supervised learning tasks are
    - a. classification (eg: spam or ham)



    - b. Regression (eg: Predict a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc) called predictors. – Predictors + Labels

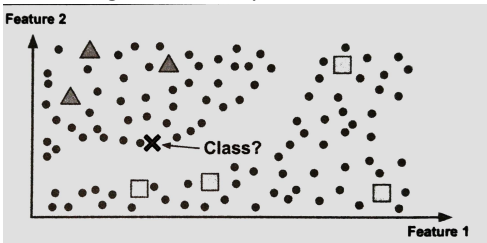


  - (3) Key supervised learning algorithms
    - a. K-Nearest Neighbors
    - b. Linear Regression
    - c. Logistic Regression
    - d. Support Vector Machines (SVMs)
    - e. Decision Trees and Random Forests
    - f. Neural Networks
- ***Unsupervised learning***
  - (1) The training data is unlabeled
  - (2) Key tasks
    - a. Visualization: 2D or 3D representation of the data
    - b. Clustering
    - c. Dimensionality reduction: to simplify the data without losing too much information; one way do this is to merge several correlate features into one.
    - d. Anomaly detection: eg: detecting unusual credit card transactions, catching manufacturing defects; automatically removing outliers from a dataset before feeding it to another learning algorithm;
    - e. Association rule learning: to dig into large amount of data and discover interesting relations between attributes; eg: people who purchase barbecue sauce and potato chips also tend to buy steak
  - (3) Key unsupervised learning algorithms
    - a. Clustering
      - i. K-Means
      - ii. Hierarchical Cluster Analysis (HCA)
      - iii. Expectation Maximization
    - b. Visualization and dimensionality reduction
      - i. Principal Component Analysis (PCA)
      - ii. Kernel PCA
      - iii. Locally-Linear Embedding (LLE)
      - iv. t-distributed Stochastic Neighbor Embedding (t-SNE)

c. Association rule learning
    i. Apriori
    ii. Eclat

- ***Semi-supervised learning***

(1) The training data is usually a lot of unlabeled data and a little bit of labeled data

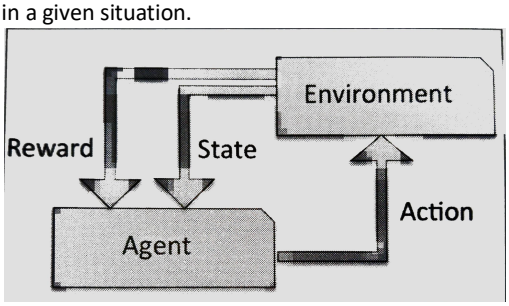


(2) Key tasks: Google photos as an example: Upload family photo to Google photos
    a. Use unsupervised machine learning algorithm (Clustering) to recognize that the same person A shows up in photos 1, 5, and 11, while another person B shows up in photos 2, 5, and 7.
    b. Then use supervised machine learning to give a label per person.

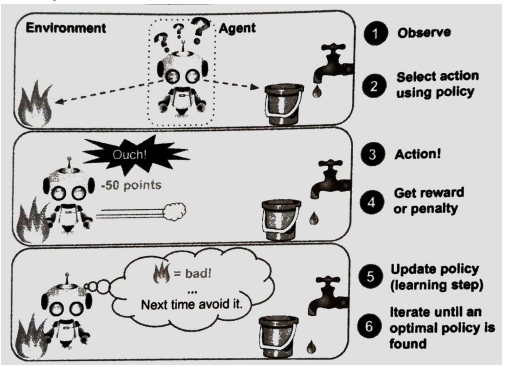
(3) Key algorithms
    a. DBNs (deep belief networks)
    b. RBMs (Boltzmann machines)

- ***Reinforcement learning***

(1) The learning system (called an *agent*) can observe the environment, select and perform actions, and get *rewards* in return (or penalties in the form of negative rewards). It must then learn by itself what is the best strategy, called 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.



(2) Example:



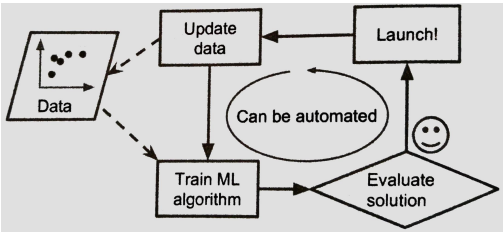
<u>Batch vs. Online learning</u>

**According to whether or not the system can learn incrementally from a stream of incoming data*

- ***Batch learning (offline learning)***

(1) Is incapable of learning incrementally: It must be trained using all the available data. (offline learning)

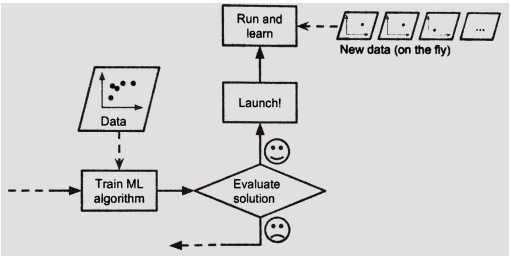
If you want a batch learning system to know about new data, you need to train a new version of the system from scratch on the full dataset (old data + new data)



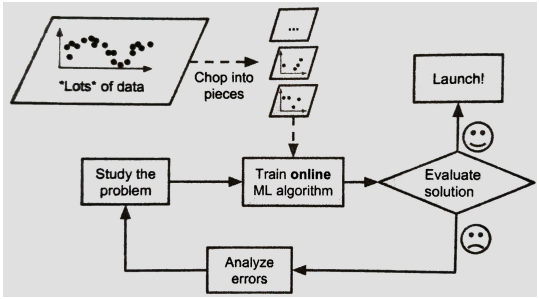
(2) Issues
- Takes time
- Requires a lot of computing resources

- ***Online learning***

a. You train the system incrementally by feeding it data instances sequentially, either individually or by small groups called mini-batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.



b. Usage
- Good for systems that receives data as a continuous flow (eg: stock prices) and need to adapt to change rapidly or autonomously.
- Good option for solutions with limited computing resource. Eg: don't to keep the training data unless you want to roll back …
- Good for solution which huge datasets that cannot fit in one machine's main memory (called out-of-core learning).



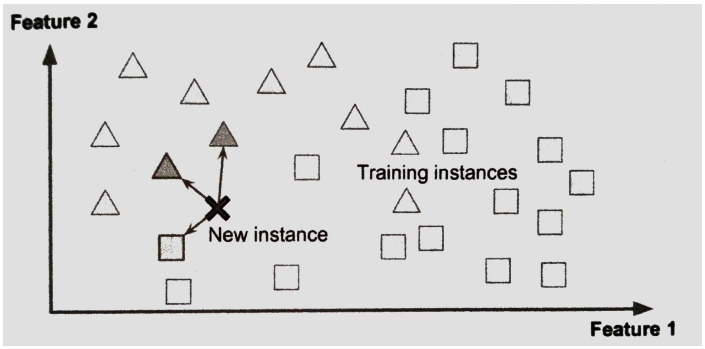
c. Issues
- Learning rate: How fast the system should adapt to changing data
  - High learning rate: rapidly adapt to new data, ten to quickly forget the old data
  - Low learning rate: the system will have more inertia. Learn slowly, but it also be less sensitive to noise in the new data or to sequences of non-representative data points
- If bad data is fed to the system, the system's performance will gradually decline. Need to monitor your system closely and promptly switch learning off (and possibly revert to a previously working state) if you detect a drop in performance

Instance-based vs. Model-based

**According to how it generalizes*

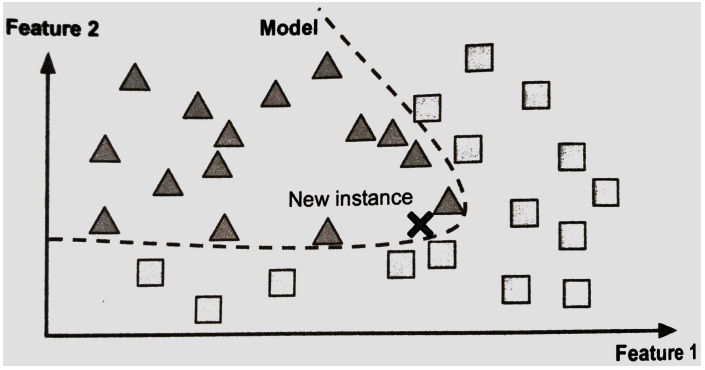
- ***Instance-based learning***

(1) Learn the examples by heart, then generalizes to new cases using a similarity measure



(2) similarity measure: eg: k-Nearest Neighbors regression

- ***Model-based learning***
  (1) Build a model of these examples, then use that model to make predictions



(2) Example: A simple linear model: life_satisfaction = Q0 + Q1 x GDP_per_capita
You feed the model your training data, and it finds the parameters that make the linear model fits best to your data. This is called "*Training the model*"

Main challenges of machine learning

For ML, the main task is to select a learning algorithm and train it on some data. So the two things that can go wrong are
- "Bad algorithm"
- "Bad data"

Bad data:

1. Insufficient quantity of training data
   a. Microsoft researcher found that very different ML algorithms performed almost identically well once they were given enough data
   b. The results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development
2. Nonrepresentative training data
   a. Sample is too small -> sampling noise
   b. Sampling method is flawed -> sampling bias
3. Poor-quality data
   a. If training data is full of errors, outliers, and noise, it will make it harder for the system to detect the underlying patterns
   b. Most data scientists spend a significant part of their time to clean up your training data
4. Irrelevant features
   a. If the training data contains enough relevant features and not too many irrelevant ones
   b. A good set of features is critical to ML success

    c. Feature engineering
        i. Feature selection
        ii. Feature extraction – combining existing features to produce a more useful one
        iii. Creating new features by gathering new data

Bad algorithms

5. Overfitting the training data
    a. It means that the model preforms well on the training data, but It does not generalize well – the model is too complex
        i. Eg: you are visiting a foreign country and the taxi driver rips you off. You might be tempted to say that all taxi drivers in that country are thieves.
        ii. If the training set is noisy, or if it is too small (which introduces sampling noise), then the model is likely to detect patterns in the noise itself.
    b. Regularization: constraining a model to make it simpler and reduce the risk of overfitting
        i. Find the right balance between *fitting the data perfectly* and *keeping the model simple enough* to ensure that it will generalize well.
        ii. Hyperparameter: is a parameter of a learning algorithm (not of the model); it is not affected by the learning algorithm itself; It must be set prior to training and remains constant during training.
6. Underfitting the training data
    a. It means the model is too *simple* to learn the underlying structure of the data.
    b. The main options to fix this program:
        i. Selecting a more powerful model, with more parameters
        ii. Feeding better features to the learning algorithm (feature engineering)
        iii. Reducing the constraints on the model (eg: reducing the regularization hyperparameter)

Testing and validating

1. Testing
    a. Split your data into two sets:
        i. Training set 80%
        ii. Test set 20% -> error rate: Generalization error (or out-of-sample error)
    b. If training error is low, but the generalization error is high -> overfitting the training data
    c. Apply regularization to avoid overfitting -> multiple model and multiple hyperparameter -> if still use test set to measure generalization error -> you will adapt the model and hyperparameters by measure the generalization error multiple times on the test set, which will be unlikely to perform well on new data

2. Validating
    a. Have a second holdout set – the validation set;
        i. You train multiple models which various hyperparameters using the training set
        ii. You select the model and hyperparameters that perform best on the validation set
        iii. When you are happy with your model and you run a single final test against the test set to get an estimate of the generalization error.
    b. Alternative way – Cross validation
        i. Split training set into complimentary subsets
        ii. Each model is trained against a different combination of these subsets and validated against the remaining parts
        iii. Once the model type and hyperparameters have been selected, a final model is trained using these hyperparameters on the full training set,
        iv. The generalized error is measured on the test set.

Stepping back – A quick summary of all above

1. Machine learning is about making machines get better at some task by learning from data, instead of having to explicitly code rules
2. There are many different types of ML systems
    a. Supervise or not
    b. Batch or online
    c. Instance-based or Model-based
3. In a ML project
    a. you gather data in a training set,

    b. you feed the training set to a learning algorithm
    c. If the algorithm is model-based it tunes some parameters to fit the model to the training set
    d. If the algorithm is instance-based, it just learns the examples by heart and uses a similarity measure to generalize to new instances
4. The system will not perform well if
    a. Training set is too small
    b. Data is not representative
    c. Polluted with irrelevant features (garbage in, garbage out)
    d. The model is too simple (underfit) or too complex (overfit)
5. Testing and validating
    a. Testing (split data into training set 80% and test set 20%)
    b. Validating – handling overfitting issue
        i. Having a separate validation set
        ii. Cross validation