CS 4101 Artificial Intelligence – 2019/2020	Handout: Revision

A. Introduction Machine Learning:

- **Machine learning is a set of tools that, broadly speaking, allow us to "teach" computers how to perform tasks by providing examples of how they should be done.**
- **Machine learning is a data analytics technique that allows the system to learn from data, identify patterns and make decisions without being explicitly programmed or with minimal human intervention**
- **e.g.: Spam classification problem**
  If we want to write a program to distinguish between valid email messages and un wanted spam.
  - **Steps to solve this in a traditional way:**
    - **Write a set of simple rules (ex flagged messages that contains features like fake header or words like FREE,WON,WINNER…)**
    - **Drawbacks of this :**
      - **It is difficult**
      - **Many missed spam messages ,or many lost email**
      - **Spammers will adjust the way they send spam in order to trick these strategies**
    - **Conclusion: It is impossible task to implement a traditional program to perform this task. It is better to use machine learning algorithm.**
  - **Steps to solve in machine learning way :**
    - **Modern spam filters are "learned" from examples: we provide the learning algorithm with example emails which we have manually labeled as "ham" (valid email) or "spam" (unwanted email)**
    - **The algorithms learn to distinguish between them automatically**

Note : Often, Machine learning methods are broken into two phases:

- **Training: A model is learned from a collection of training data.**
- **Application: The model is used to make decisions about some new test data.**

Some definitions:(not important read only)

- **The Artificial Intelligence View. Learning is central to human knowledge and intelligence, and, likewise, it is also essential for building intelligent machines. Years of effort in AI has shown that trying to build intelligent computers by programming all the rules**

cannot be done; automatic learning is crucial. For example, we humans are not born with the ability to understand language — we learn it — and it makes sense to try to have computers learn language instead of trying to program it all it.

- **The Software Engineering View: Machine learning allows us to program computers by example, which can be easier than writing code the traditional way.**
- **The Stats View. Machine learning is the marriage of computer science and statistics: computational techniques are applied to statistical problems. Machine learning has been applied to a vast number of problems in many contexts, beyond the typical statistics problems. Machine learning is often designed with different considerations than statistics (e.g., speed is often more important than accuracy).**

B. Types of Machine Learning:

Main types are:

1) **Supervised Learning**
   - **The correct classes of the training data are known as shown in the figure 1.1 e.g., "spam" or "ham." In our previous example in introduction**
   - **The two most common types of supervised learning are classification (where the outputs are discrete labels, as in spam filtering) and regression (where the outputs are real-valued).**
   - **Supervised learning is used in applications where historical data predict likely events e.g.: fraud detection, email spam detection, diagnosis, image classification , risk management , score prediction**

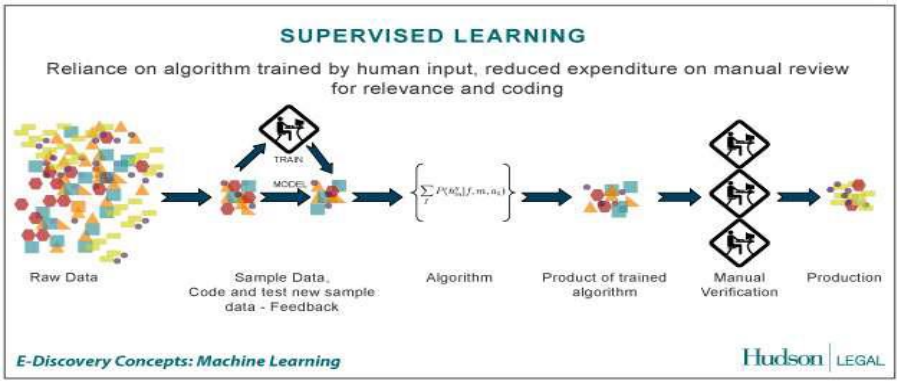


Figure 1.1

2) **Unsupervised learning**

- **The correct classes of the training data are not known, , in which we are given a collection of unlabeled data, which we wish to analyze and discover patterns within as shown in the figure 1.2**
- **The two most important examples are dimension reduction and clustering.**

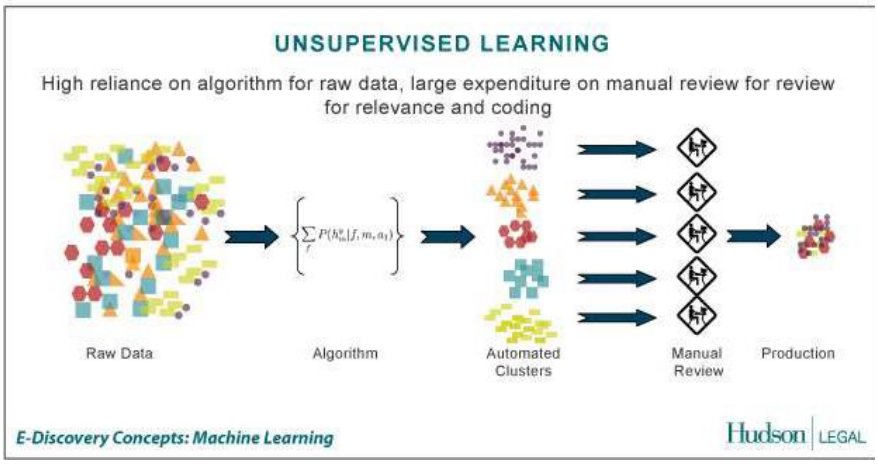


Figure 1.2

3) **Reinforcement learning :**

- **allows the machine or software agent to learn its behavior based on feedback from the environment. This behavior can be learnt once and for all, or keep on adapting as time goes by as shown in figure 1.3**
- **In other words, an agent (e.g., a robot or controller) seeks to learn the optimal actions to take based the outcomes of past actions.**

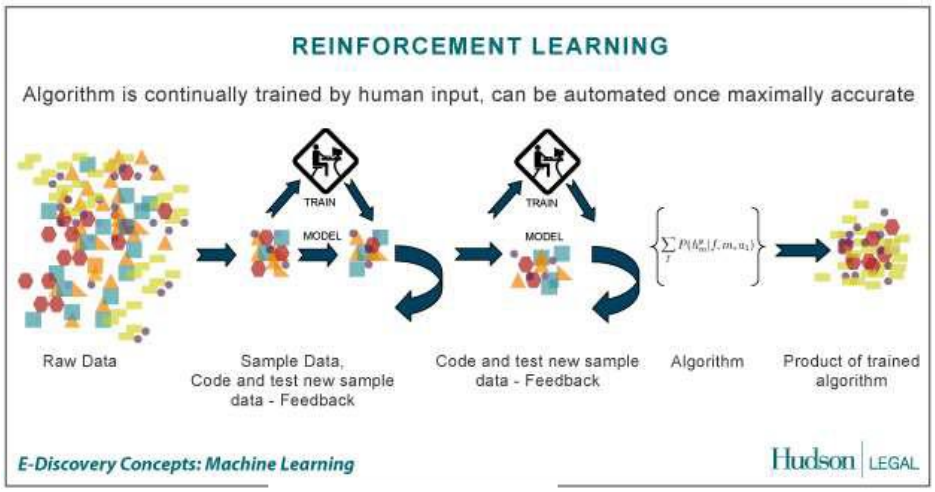
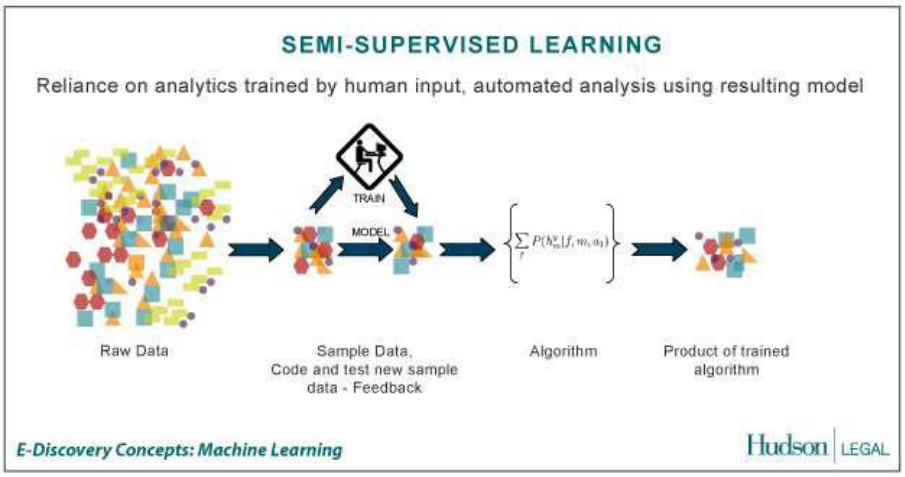


Figure 1.3

There are many other types of machine learning as well, for example:(the semi supervised is important only ,the rest for reading)

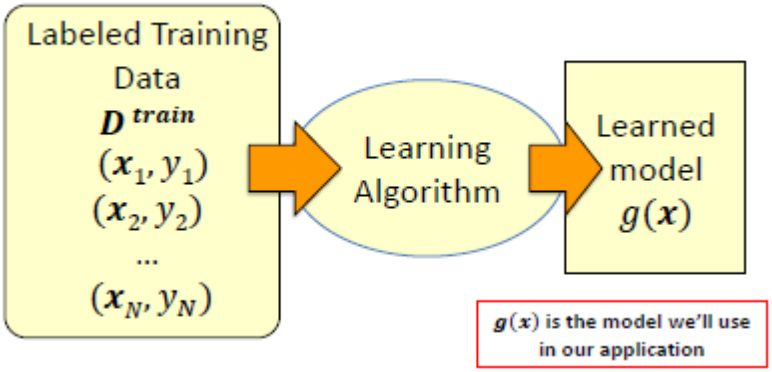
1. **Semi-supervised learning, in which only a subset of the training data is labeled**
   It is a mix of supervised and unsupervised learning as shown in figure



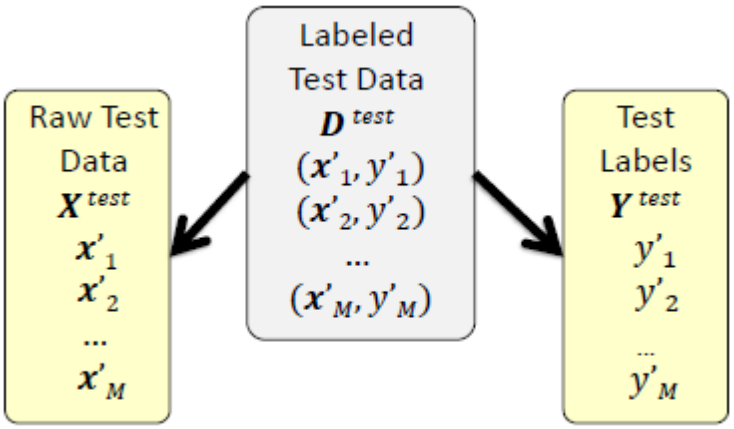
2. **Time-series forecasting, such as in financial markets**
3. **Anomaly detection such as used for fault-detection in factories and in surveillance**
4. **Active learning, in which obtaining data is expensive, and so an algorithm must determine which training data to acquire**

C. **<u>Supervised learning:</u>**

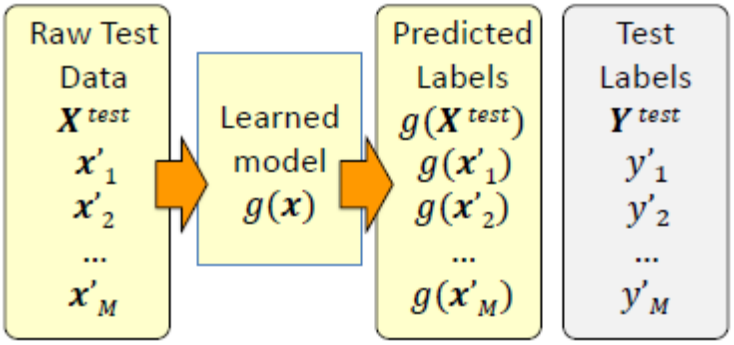
- **is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples.**
- **In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).**
- **A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances like shown in figure**
- **Example:**
  i. **Given a set of N training examples of form $\{(x_1,y_1),\ldots..,(x_n,y_n)\}$,where x is feature vector and y is its label**
  ii. **Learning algorithm seeks a function g to map x features into y label as shown in figure this is called training phase**



iii. Then, the learned model is tested by tested set that was separated from the dataset before training phase, separate test data like shown in the figure



Apply the model to raw test data and then evaluate by comparing predicated labels against the test labels as shown in the figure



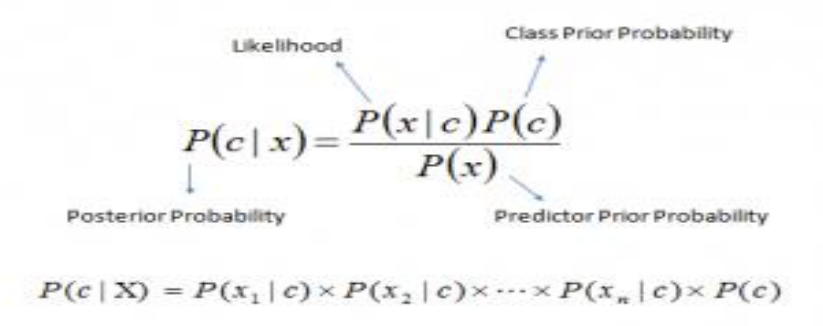
- **Some approaches and algorithms:**
  - **Naive bayes classifier**
  - **Nearest Neighbor Algorithm**
  - **Decision tree learning**
  - **Artificial neural network**
  - **Support vector machines**
  - **Random Forests**
- **Some applications:**
  - **Bioinformatics**
  - **Database marketing**
  - **Handwriting recognition**
  - **Information retrieval**
  - **Learning to rank**
  - **Object recognition in computer vision**
  - **Optical character recognition**
  - **Spam detection**
  - **Pattern recognition**
  - **Speech recognition**

C.1 <u>Naive bayes classifier</u>

- **When to use:**
  - **Moderate or large training set available**
  - **Attributes that describe instances are conditionally independent given classification**
- **Successful applications:**
  - **Diagnosis**
  - **Classifying text documents**
- **A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.**
- **Even if these features are related to each other, a Naive Bayes classifier would consider all of these properties independently when calculating the probability of a particular outcome.**
- **A Naive Bayesian model is easy to build and useful for massive datasets. It's simple, and is known to outperform even highly sophisticated classification methods**
- **To calculate the probability that an event will occur, given that another event has already occurred**
- **To calculate the probability of an outcome given the value of some variable**
- **to calculate the probability of a hypothesis(h) being true, given our prior knowledge(d), we use Bayes' Theorem as follows in the equation:**

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



Above:

- **P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).**
- **P(c) is the prior probability of class.**
- **P(x|c) is the likelihood which is the probability of predictor given class.**
- **P(x) is the prior probability of predictor.**

Note : Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

- **Estimating $P(x_i|c_j)$ instead of $P(x_1,x_2,\ldots,x_n|c_j)$ greatly reduce the number of parameters (and the data sparseness).**
- **The learning step in Naïve Bayes consists of estimating and $P(c_j)$ based on the frequencies in the training data**
- **An unseen instance is classified by computing the class that maximizes the posterior**
- **When conditioned independence is satisfied, Naïve Bayes corresponds to MAP classification.**

- **Example: 'Play Tennis'**

  Given the following Dataset :

  - Predict: : For the day <sunny, cool, high, strong>, what's the play prediction?

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

- Answer:
  - Based on the examples in the table, classify the following day x:
    x=(Outl=Sunny, Temp=Cool, Hum=High, Wind=strong)
  - That means: Play tennis or not?
    - P1 = P(Outl=Sunny, Temp=Cool, Hum=High, Wind=strong | Play Tennis = yes)
    - P2 = P(Outl=Sunny, Temp=Cool, Hum=High, Wind=strong | Play Tennis = no)
    - Max {P1 , P2}
    - H = {yes , no}
  - Learning Phase
    $P(\text{Play}=Yes) = 9/14$, $P(\text{Play}=No) = 5/14$

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

  - Test Phase
    - Given a new instance, predict its label

x'=(Outlook=*Sunny,* Temperature=*Cool,* Humidity=*High,* Wind=*Strong*)

    - Look up tables achieved in the learning phrase

P(Outlook=*Sunny*|Play=*Yes*) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

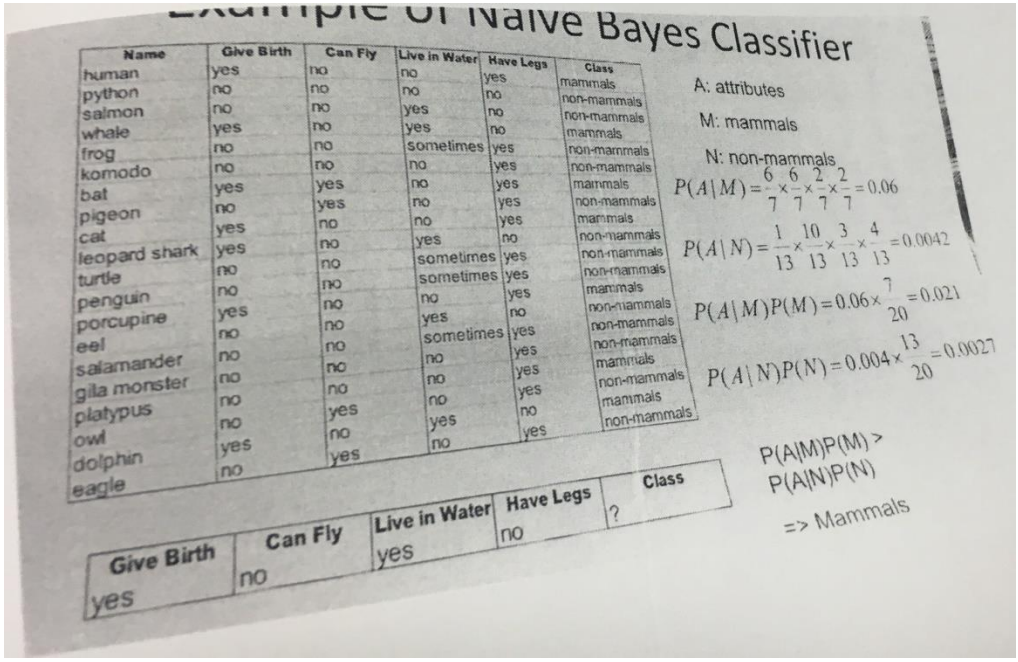
P(Outlook=S*unny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

    - Decision making with the MAP rule

- P(*Yes*|**x'**) ≈ [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053
- P(*No*|**x'**) ≈ [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|**x**') < P(*No*|**x**'), we label **x**' to be "*No*".

- Example from previous lectures (Exam 2018-2019):



- Code for naïve Bayes classifier (for read only):
  - The conditional probability for a feature value given the class label can also be estimated from the data. Specifically, those data examples that belong to a given class, and one data distribution per variable. This means that if there are *K* classes and *n* variables, that *k * n* different probability distributions must be created and maintained.
  - A different approach is required depending on the data type of each feature. Specifically, the data is used to estimate the parameters of one of three standard probability distributions.
  - In the case of categorical variables, such as counts or labels, a multinomial distribution can be used. If the variables are binary, such as yes/no or true/false, a binomial distribution can be used. If a variable is numerical, such as a measurement, often a Gaussian distribution is used.
    - **Binary**: Binomial distribution.
    - **Categorical**: Multinomial distribution.
    - **Numeric**: Gaussian distribution.
  - These three distributions are so common that the Naive Bayes implementation is often named after the distribution. For example:
    - **Binomial Naive Bayes**: Naive Bayes that uses a binomial distribution.
    - **Multinomial Naive Bayes**: Naive Bayes that uses a multinomial distribution.
    - **Gaussian Naive Bayes**: Naive Bayes that uses a Gaussian distribution.

Main steps:

```
# importing required libraries
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# read the train and test dataset
train_data = pd.read_csv('train-data.csv')
test_data = pd.read_csv('test-data.csv')



# seperate the independent and target variable on training data
train_x = train_data.drop(columns=['Survived'],axis=1)
train_y = train_data['Survived']

# seperate the independent and target variable on testing data
test_x = test_data.drop(columns=['Survived'],axis=1)
test_y = test_data['Survived']

'''
Create the object of the Naive Bayes model
You can also add other parameters and test your code here
Some parameters are : var_smoothing
Documentation of sklearn GaussianNB:

https://scikit-
learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.h
tml

 '''
model = GaussianNB()

# fit the model with the training data
model.fit(train_x,train_y)

# predict the target on the train dataset
predict_train = model.predict(train_x)
print('Target on train data',predict_train)

# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('accuracy_score on train dataset : ', accuracy_train)
```

```
# predict the target on the test dataset
predict_test = model.predict(test_x)
print('Target on test data',predict_test)

# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('accuracy_score on test dataset : ', accuracy_test)
```