Part III Introduction to Machine Learning

\*What is learning?

Learning is **acquiring expertise** from **past experience**.

Ex. Rats learn to avoid poisonous food

Ex. Filter spam emails. Machine memorizes all previous emails that are labeled as spam and compare new emails with them. Learning by memorization is certainly not enough. We want machines to be able to label unseen emails. This ability is usually referred to as generalization, inductive reasoning, or inductive inference.

Generalization may result in false conclusion.

Ex. Many rituals in human history. /Human sacrifice for good years. /NBA players’ free throw rituals.

Ex. Rats learn to avoid poisonous food.

In 1996, a series of experiments were carried out

Rat -> food -> nausea -> avoid

\\_electrical shock/vocal signal -> not avoid

It seems that rats know a prior that food and nausea are correlated but food and the other two effects are uncorrelated.

We refer to this prior knowledge as inductive bias.

It turns out that biasing the learning process is inevitable for the success of learning algorithms.

Translating **domain expertise** into **learning bias** is an important theme in ML.

Strong bias: easy to train/learn, but less flexible.

\*Learning vs Programming

When do we need learning instead of directly program a computer?

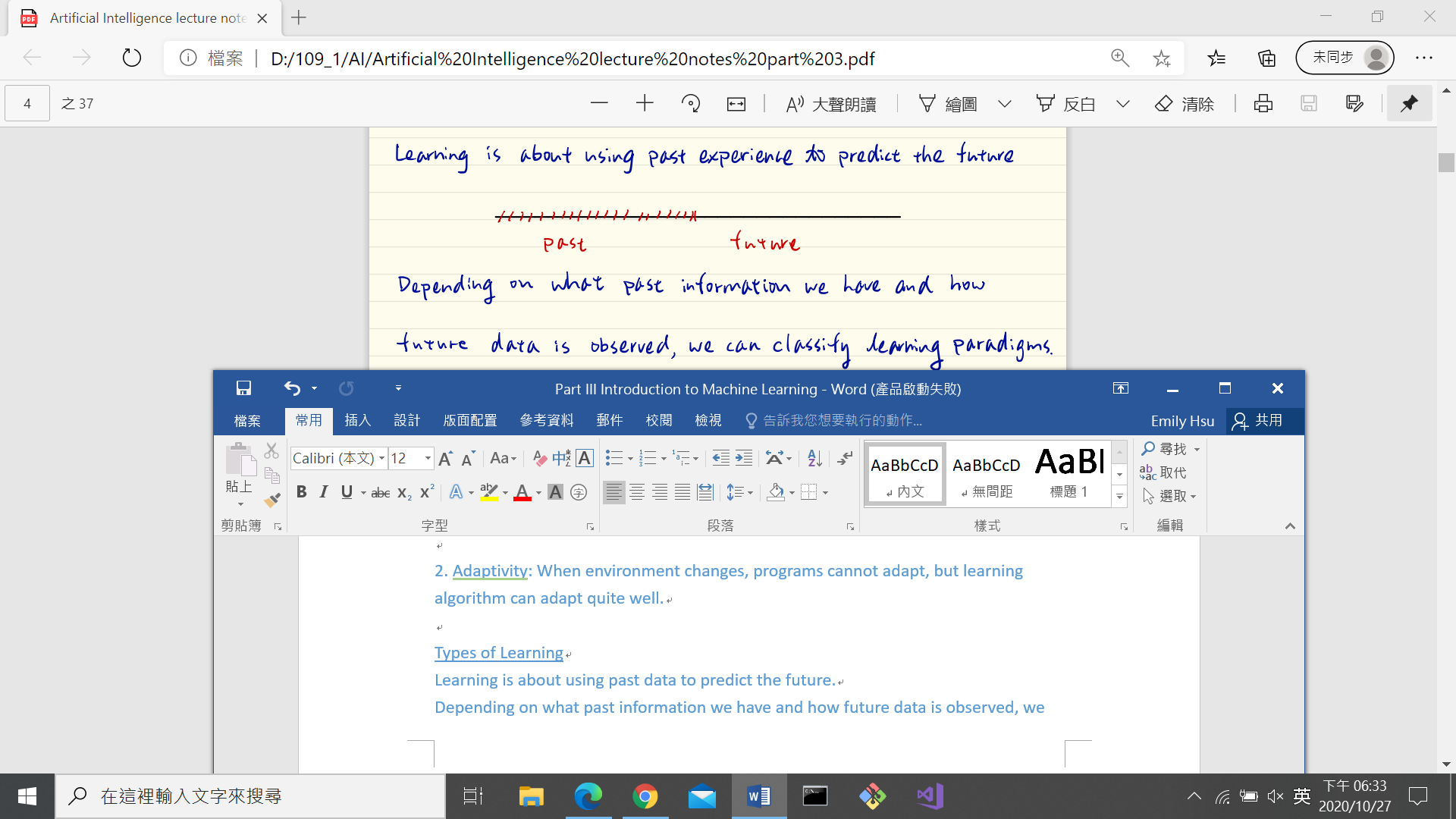
1. Tasks that are too complex: Many tasks are super complex so that we cannot extract a well-defined program. Some tasks are even beyond human capabilities.

Ex. Investments, medical, recommendation system

2. Adaptivity: When environment changes, programs cannot adapt, but learning algorithm can adapt quite well.

Types of Learning

Learning is about using past data to predict the future.



Depending on what past information we have and how future data is observed, we can classify learning paradigms.

1. Supervised(Classification) vs Unsupervised(Clustering)

In supervised learning, we usually have correct answers for training data; In each training data, there is significant information which is missing in test data. The goal is to predict this information.

Ex: In learning to filter spam emails, the emails used in training/learning stage are labeled by human users as spam/not spam. We then use learned algorithm to predict whether a new email is spam or not. Note that this newly labeled email.

In unsupervised learning, there is no difference between training and test data. The algorithm tries to come up with some summary or compressed version of data.

Ex. Anomaly detection tries to identify unusual behavior from a large data set.

Ex. Clustering tries to partition a large data set into subsets where data in a subset shares common properties.

Something in between: Training data contains some information that test data does not. But the algorithm aims at predicting even more information.

Ex. In teaching AI play GO or Chess, training data reveals the ultimate result (win or lose) but the algorithm is expected to predict next step.

Investigated under the name Reinforcement Learning.

2. Active vs Passive

Depending on the role played by the learner, we can distinguish between “active” and “passive” learner.

Active learners are those who interact with the environment at training time by posing queries or performing experiments.

Passive learners only observe information provided by the environment.

Ex: Filter spam emails usually involve passive learners.

3. Statistical vs Adversarial

In statistical learning, training data are assumed to be generated by some random processes.

Sometimes, there are some adversarial teacher who generates misleading data.

4. Online vs Batch

In online learning, learner has to respond online.

Ex: Investment of stock market. The learner may become expert over time, but may make costly mistakes in the process.

One outstanding example is the multi-armed bandit problem. In batch learning, the learner would have access to a large training data set and can start predicting after sufficient exploration.

In this part, we will:

* Introduce the statistical machine learning framework.

Supervised + Statistical + batch + passive learning

* Introduce the idea of PAC learning
* Talk about some classic algorithms

Statistical Learning Framework

\*Mathematical model

Input

Domain set: An arbitrary set, , containing all the possible objects that we may want to label.

Label set: A set containing all the possible labels.

Training data: is a finite sequence of pairs in

, where each pair is a labeled domain point.

Output

The learner is equipped with a prediction rule . This function is called a predictor or a classifier. This predictor will give any a label .

Data generation

We assume that training data is generated according to a probability distribution D.

Note that D comes from the nature and the learner does NOT know any information about D.

We assume that there is a correct labeling function that gives every a correct label . Again, the learner does NOT know f.

Measure of error

We define the error of prediction rule h to be

That is, the probability of randomly choosing that results in

is usually called generalization error, the true error, or the risk.

: Generalization error

D: true distribution

f: true labeling function

h: prediction function

S: training data

\*Empirical Risk

Generalization error is unknown to the learner because the learner is oblivious to D and f.

An alternative is to focus on what is available.

Empirical risk

Let S denote the training data set and let be a prediction based on S.

We define the empirical risk as

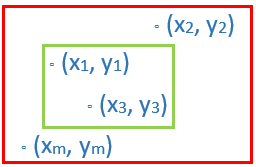
i.e., fraction of error when we apply the learned prediction to training data.

The learning paradigm that tries to minimize empirical risk is called empirical risk minimization (ERM).

Problem of ERM: Overfitting

Overfitting refers to the scenario where our algorithm fits the training data set too well so that it loses generalizability.

Ex.

Suppose D is uniform over the area inside red square.

maps every inside the green square to 1, and 0 otherwise. Given a training data set , a perfect ERM algorithm is

This will have because it fits perfectly. But it can be easily shown that every prediction function assigning 1 only finite times suffers from generalization of

\*ERM with Inductive Bias

One way to guarantee that ERM does not overfit is to introduce inductive bias.

This will make sure that as long as the prediction works well with training data, it will be highly likely to work well for test data.

We therefore apply ERM to a restricted set of predictors.

Let be a hypothesis class that contains a bunch of prediction functions .

Given a training data set, ERMH learner uses ERM to select a such that

Such restrictions are called inductive bias.

arg是變元（即自變量argument）的英文縮寫

arg min就是使後面這個式子達到最小值時的變量的取值

arg max 就是使後面這個式子達到最大值時的變量的取值

例如函數F(x,y):

arg min F(x,y)就是指當F(x,y)取得最小值時，變量x,y的取值

arg max F(x,y)就是指當F(x,y)取得最大值時，變量x,y的取值

Intuitively, choosing a more restricted H better protects us against overfitting but also causes a stronger bias.

(因為如果有很多可以選擇(less restricted H)，則可能更選到太過於貼合training data set的h，造成overfitting)

Finite Hypothesis Classes

One simplest type of restrictions is to constrain the size of H. One can show that under this constraint ERMH will never overfit.

We make the following assumption for now:

Assumption (Realizability) There exists such that i.e., there exist a function in that can perfectly mimic with probability 1.

This implies that for randomly sample , labeled by , w.p. 1.

(Note that here, h\* can depend on )

This assumption implies that w.p. 1.

But how about ?

We analyze this when data in are iid (independent and identically distributed) sampled according to .

For , we denote it by

Recall that is the prediction function learned according to randomly chosen training data set S.

The corresponding risk is random. i.e., some weird things could happen.

Ex: We happen to sample non-spam emails. Since no spam email ever shows up, would decide to label every new email as non-spam. If probability of spam is , we would end up mis-labeling with probability .

Let the probability of getting non-representative data be and call the confidence parameter.

Also, with finite m, we cannot guarantee perfect prediction. Thus, define the accuracy parameter .

For a pair , we want to find such that the probability that the generalization error is smaller than is within the confidence parameter .

i.e.,

Linear Predictors

Linear predictions

half space linear regression logistic regression

linear programming perception algorithm least square algorithm

Stochastic gradient descent

Define: (Affine) An affine function is a linear function with translation

where

Define the class of affine functions

Remark:

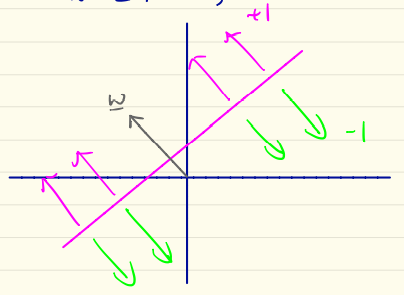
1) It is convenient to write

2) Within the family of linear predictor, different predictors apply different

Half Space: A hypothesis class designed for binary classification, i.e.

Def: (Half space)

Geometrically, determines a half space that is a shifted version of the half space perpendicular to



For half spaces, realizability condition is also referred to as the separability condition.

Linear programming: (for Solving ERM)

Let be labeled training set.

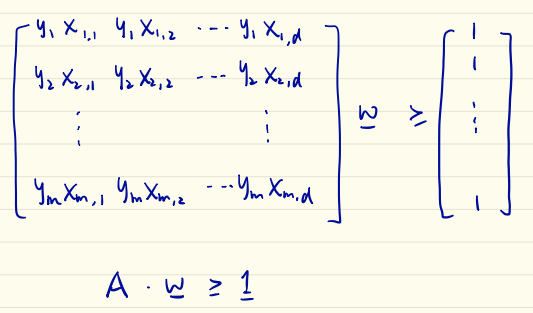
We are looking for such that

Such always exists under separability condition.

Let w\* be such a and let

normalization:

Hence, there exists such that



Linear programming:

**Set u=0, v=1 LP solver(套現有的)**

Linear Regression

In linear regression, . The hypothesis class of linear regression is simply .

(Predict height.)

Def: (Square loss & MSE)

One commonly used loss function is the square loss.

Under this loss function, the empirical risk becomes the famous mean squared error (MSE)

The problem of ERM becomes minimum mean square error (MMSE)

Least Square (Gauss 1795 Legendre 1805)

Let and let

Clearly,

線性代數: Orthogonal Projection

Logistic Regression

Soft information/decision vs hard information/decision

In logistic regression, we set . It is usually used for binary classification and indicates the probability of label of x being 1.

**Convex**

Stochastic Gradient Descent (SGD)

Iterative algorithm

Gradient Descent: T iterations (在維度小、offline的問題表現好)

Objective: minimize f(w)

Initialize: w’=0

Update: wt+1=wt-

Output: 最後一個 或 平均(全部/最後10個/最後20個…)

Intuition: gradient f(w) points in the direction of greatest rate of increase of f around w.

SGD

In SGD, we replace gradient f(wt) by a random vector Vt st.