Foundation of Machine Learning 1주차

정재헌, 우지수 / 2023.01.13

Computational Data Science LAB

CONTENTS

1. What is machine learning?
2. Elements of the ML Pipeline
3. Evaluating a Prediction Function
4. Other Sources of Test ≠ Deployment
5. Model Complexity & Overfitting
6. Introduction to Statistical Learning Theory

01 | What is Machine Learning?

What is Machine Learning for?

- prediction problem을 해결할 때 공통적인 주제:
  - ✓ **input** x가 주어지고,
  - ✓ "정확한"**output** y를 예측하는 것.

01 | What is Machine Learning?
Example

- 스팸메일 Detection
  - ✓ Input: email
  - ✓ Output: “SPAM” or “NOT SPAM”
  - ✓ **Binary classification(이진분류)**, **2개의** output에서 한가지를 선정하는 것.
- 의학적 진단
  - ✓ Input: 증상(fast breathing, nausea,…)
  - ✓ Output: 진단(flu, common cold,…)
  - ✓ **multiclass classification** problem: 몇 개의 **이산적인** output에서 한가지를 선정하는 것.
  - ✓ 불확실성을 표현하는 방법을 알고 싶은 경우: **Probabilistic classification** or **soft classification**(flu vs cold)
- 주식 가격예측
  - ✓ Input: 오늘 까지의 주식 가격 data
  - ✓ Output: 내일 주식 가격
  - ✓ **Regression(회귀)** problem, output이 연속적인 숫자들의 형태로 나옴.

01 | What is Machine Learning?

What is Machine Learning?

- What is not ML: **Rule-Based Approaches**
  - ✓ In medical diagnosis:
    1. 책을 읽거나 의사에게 문의하는 것(i.e. "**experts**").
    2. 진단 process를 이해하는 것.
    3. 위의 내용을 알고리즘을 구현한 것.(a "**rule-based system**")
  - ✓ It doesn't sound too bad, but this algorithm has several issues.
    1. **인력소모가** 큼.
    2. 예상치 못한 것에서는 **일반화할** 수 없음.
    3. **uncertainty**(불확실성)한 것은 처리하지 않음.

- What is ML: Not a Rule-Based Approaches
  - ✓ Machine "**learns**" on its own.
  - ✓ 우리는 그저"**training data** (input x , output y) pairs"를 제공할 뿐.
  - ✓ 이러한 형식으로 학습하는 것을 **supervised learning**라고 부름.

- The most different thing between Rule-Based Algorithms and ML is "**what is the subject**?".

01 | What is Machine Learning?

What is Machine Learning?

- Machine Learning Algorithm:
  - ✓ Input: training data
  - ✓ Output: “**prediction function**”
- The Prediction Function:
  - ✓ **Machine learning** helps find the **best prediction function**.
  - ✓ Input x와 output y 가 존재
- “**prediction function**”의 요소

02 | Elements of the ML Pipeline

Feeding Input to ML Algorithms

- Raw data has some of types like:
  - ✓ Text documents, Variable-length, time series, Image files, Sound recordings, DNA sequences

- **그러나** 대부분의 ML Prediction function은 input으로 다음과 같은 값을 원함.:
  - ✓ **고정된 length의 arrays of numbers**
  - ✓ `double[d]` – for the computer scientists
  - ✓ $\boldsymbol{R^d}$ – for the mathematicians

02 | Elements of the ML Pipeline
Feature Extraction

- Definition:
  - ✓ raw input x를 $\boldsymbol{R^d}$에 mapping하는 것은 **feature extraction** 또는 **featurization**라 부름.
  - ✓ $X \overset{x}{\rightarrow}$ `Feature Extraction` $\overset{\phi(x)}{\longrightarrow} R^d$
- Better features → 덜“**smart**”한 ML을 필요로 함.
- **Feature vectors** are often called **input vectors.**

02 | Elements of the ML Pipeline

Feature Template: One-Hot Encoding

- 범주형 변수는 일반적으로 string형식의 문자열이 존재.
  - ✓ ML algorithm은 input 값에 문자열 값을 지원하지 않음.
  - ✓ 따라서 문자열 값을 숫자로 대체.**(categorical variable encoding)**

1. **one-hot encoding:**
   - ➢ **Nonzero** value를 가지는 **binary feature**의 집합.
2. **Dummy encoding:**
   - ➢ **Dummy** encoding 또한 **binary feature**를 사용한다.
   - ➢ 변수의 범주 수(k)와 동일한 수의 더미 변수를 만드는 대신, 더미 인코딩은 **k-1 더미 변수**를 사용.

02 | Elements of the ML Pipeline

Labeled Data vs Unlabeled Data

Ftr1	Ftr2	···	Y
0	1.54	···	False
1	-1.9	···	True
0	2.3	···	False

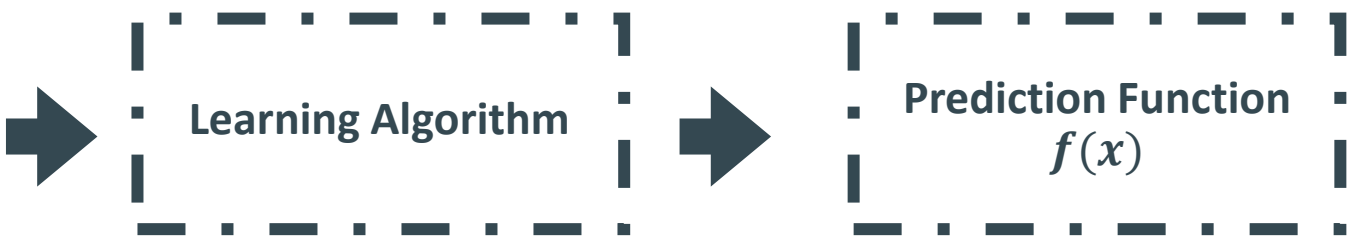
Ftr1	Ftr2	···	Y
0	1.54	···	?
1	-1.9	···	?
0	2.3	···	?

- Labeled Data
  - ✓ 각 행은“**example**” 또는 “**labeled datum**”.
  - ✓ The last column is the **output** or “**label**” column.
- Unlabeled Data
  - ✓ 마지막 missing labels을 predict하기 원함.

02 | Elements of the ML Pipeline
Learning Algorithm

- A **learning algorithm** has
  - ✓ **input**: labeled data (i.e. the training set)
  - ✓ **output**: a prediction function

Ftr1	Ftr2	⋯	Y
0	1.54	⋯	False
1	−1.9	⋯	True
0	2.3	⋯	False

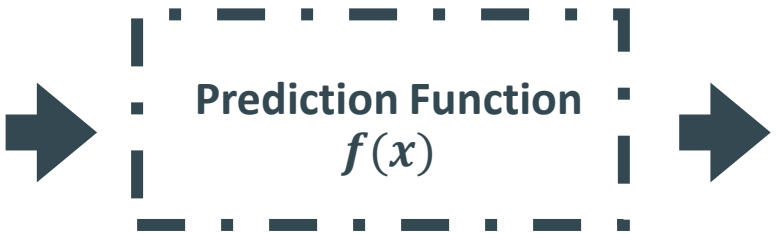


02 | Elements of the ML Pipeline

Prediction Functions

Ftr1	Ftr2	⋯
0	1.54	⋯
1	-1.9	⋯
0	2.3	⋯

[Unlabeled] Input Data



Y
False
True
False

[Predictions]

- A **prediction function** has
  - ✓ **input**: a feature vector (a.k.a. “input vector”)
  - ✓ **output**: a “label” (a.k.a. “prediction”, “response”, “action”, or “output”).
- 평가는 어떻게 진행할지.

03 | Evaluating a Prediction Function

Evaluating a Single Prediction: The Loss Function

- Evaluating a **Prediction Function**
  - ✓ Very **important part** of machine learning.
- **loss function**은 "**target**" output으로부터 예측이 얼마나 떨어져 있는지 **점수를 매김**.
  - ✓ Big Loss = Bad Error
  - ✓ Small Loss = Minor Error
  - ✓ Zero Loss = No Error

1. Classification loss or "0/1 Loss"
   - ✓ Loss is 1 if prediction is wrong.
   - ✓ Loss is 0 if prediction is correct.
2. Square loss for regression
   - ✓ loss = $(predicted - target)^2$

03 | Evaluating a Prediction Function

Evaluating a Prediction Function

- ML algorithm은 prediction function을 제공.
  - ✓ “Average classification loss on training data was 0.01.”
  - ✓ Loss가 2% 미만이니 괜찮다 라고 말할 수 없음.
  - ✓ 위 prediction function은 training data에서만 훈련됨.
  - ✓ Prediction function은 **new input**에 대해 잘 수행되어야 한다.
- A “**test set**” is labeled data that is **independent** of **training data**.

03 | Evaluating a Prediction Function

Train/Deploy vs Train/Test vs Train/Validate

1. **Train/Deploy:**
   - ✓ 모든 labeled data으로 모델 생성.
2. **Train/Test:**
   - ✓ split data로 모델 생성.
   - ✓ split data는 random하게 특히 sorted data를 쓰면 **부정확**(편향성).
   - ✓ **Times series**는 random하게 하면 안됨, **시간 상관관계** 존재.
3. **Train/Validation:**
   - ✓ Test set과 비슷하지만 Train data를 한 번 더 split한다고 생각.

[Deployment Set]

[Test Set]

[Validation Set]

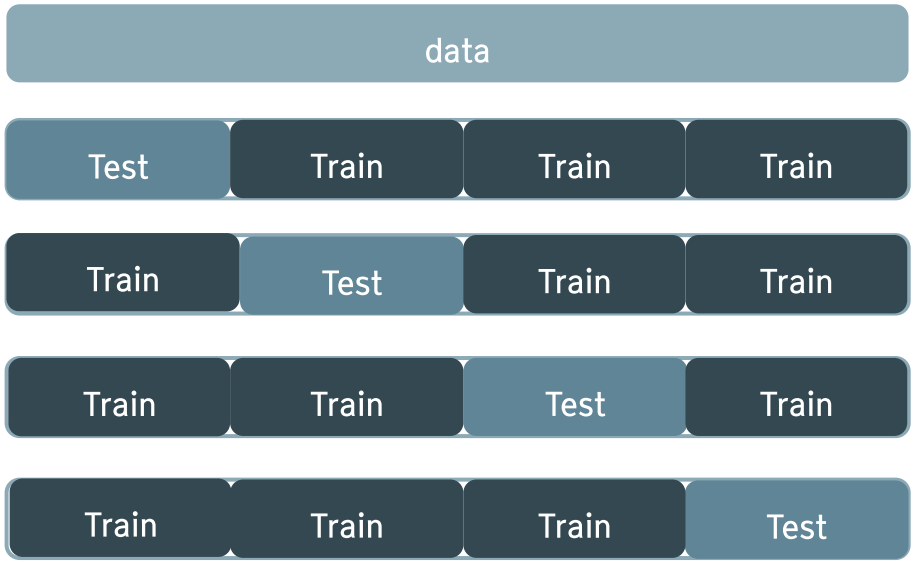
03 | Evaluating a Prediction Function

K-Fold Cross Validation

- **test set**이 너무 작아 **좋은 성능**의 추정을 얻지 못한다고 가정.
  - ✓ 총 데이터 개수가 적은 데이터 셋에 대해 정확도 향상 가능.
  - ✓ 전체 데이터에 대해서 한번씩 검증을 해볼 수 있음.

1. training을 k개의 fold로 나눔.
2. (k-1)개의 training data, 마지막 한 개는 **validation data**로 지정.
3. 모델을 생성하고 예측한 다음 **error**를 추출.
   - ✓ k개의 정확도 산출, k개의 결과물의 평균값이 k-fold 모델의 성능.

- 데이터를 어떻게 split해야 하는지.



04 | Other Source of Test ≠ Deployment

Data Leakage, Sample bias

- **Data Leakage(데이터 누출)**:
  - ✓ Information about labels **sneaks** into features(레이블에 대한 정보가 기능으로 몰래 들어갈 때).
  - ✓ e.g. 코로나 확진 여부를 예측할 때 코로나 확진자 지원금 수령여부를 feature값으로 넣는 경우.
  - ✓ 변수가 **target 값의 정보**를 가지고 있을 경우 **과적합**이 발생.

- **Sample bias(표본 치우침)**:
  - ✓ **모집단의** 일부가 다른 feature보다 sample에서 선택될 가능성이 높을 때 발생.
  - ✓ 즉, **편향된** 표본의 결과는 표본과 특성을 공유하는 모집단으로만 **일반화**될 수 있다.

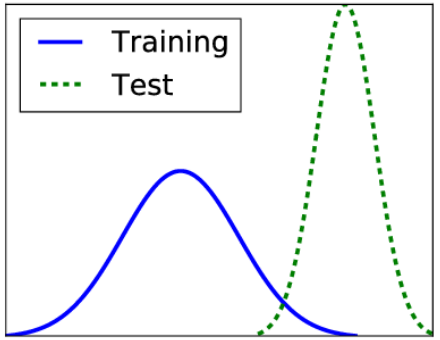
04 | Other Source of Test ≠ Deployment

Nonstationarity

- Nonstationarity(비정상성):

1. Covariate Shift: **공변량 분포의 변화**
   - ✓ Training data와 Test data의 **Distribution**이 다른 경우에 **발생**.
   - ✓ e.g. 젊은 얼굴로 얼굴 인식 알고리즘을 구축했는데 나이 든 얼굴이 input인 경우.

2. Concept Drift: **개념의 변화**
   - ✓ **시간의 경과**에 따른 input의 변화에 따라 output을 **수정**해야한다.
   - ✓ e.g. 스팸 메일을 검출하는 머신러닝 모델 구축 -> 처음엔 높은 정답률 -> 반년 후에는 거의 검출 못 함.
   - ✓ **이유** : 반년 동안 스팸 메일을 보내는 쪽에서 새로운 스팸 방법을 고안. 독립변수는 그대로인데 **종속변수가 변화.**



[Covariate Shift]

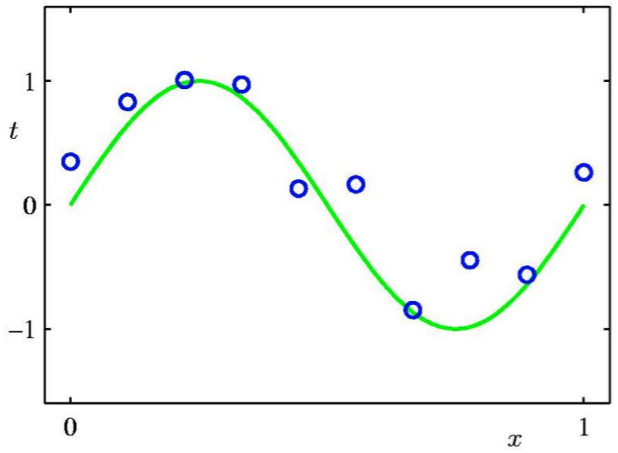
- 예측함수로 만들었을 때 문제점.

05 | Model Complexity & Overfitting

Polynomial Curve Fitting

- Toy Example:
  - ✓ Green line is **truth**; Blue points are our **train data.**

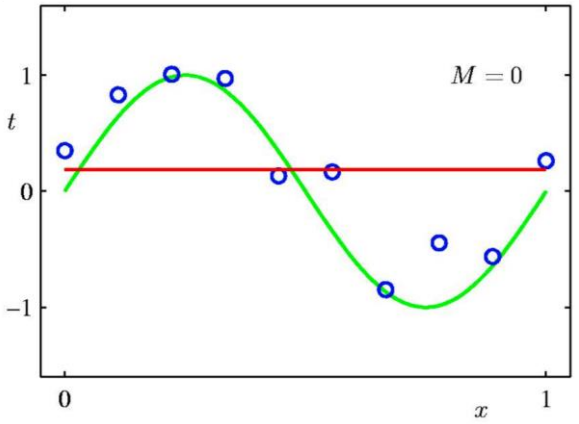
- Curve Fitting **다항식**:
  - ✓ $f\ (x) = \ w_0 + w_1 x\ + w_2 x^2 + \cdots + w_M x^M$
  - ✓ 이 함수가 **학습을 진행**하면서 $w_0, w_1, \ldots, w_M$의 **parameter 행렬**을 반환.
  - ✓ 위 **parameter**와 차수 **M**을 통해서 **Prediction function**을 생성.
    (이 때, 차수M은 **hyperparameter**이다.)



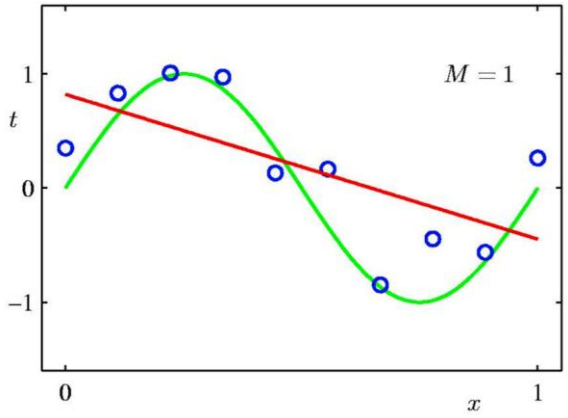
[Toy Example]

05 | Model Complexity & Overfitting

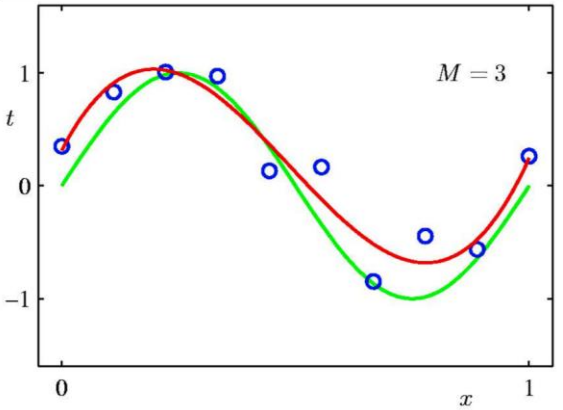
Example: Polynomial Curve Fitting



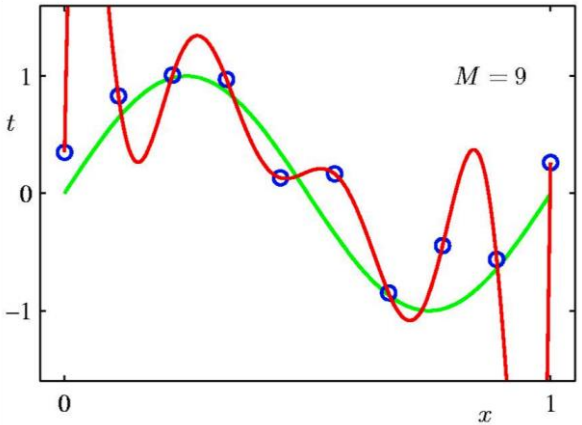
Fit with M=0 → Underfit



Fit with M=1 → Underfit



Fit with M=3 → Good!



Fit with M=9 → Overfit

- 차수 M은 **Model Complexity**를 control.
  - ✓ 차수가 크면 클수록 더 “complex”한 Prediction Function을 따름.
- **Overfitting**
  - ✓ Training의 성능은 **좋지만** test/validation의 성능이 **안좋은 경우**
  - ✓ 모델의 **complexity**를 줄이거나 더 많은 **training data**를 집어 넣음.
- 평가하는 방법이 무엇이 있는지.

06 | Introduction to Statistical Learning Theory

Decision Theory: High Level View

- 데이터 사이언스에서는 **의사결정(Decision)**을 만들고 **행동(Action)**을 취하고 **출력물(Output)**을 생성해낸다.
- **Actions :**
  - ✓ Action은 우리의 시스템에 의해 생산되는 것의 일반적인 용어.
  - ✓ 함수 그 자체라고 생각.
- **Evaluation Criterion**(평가기준):
  - ✓ Decision Theory가 최적의 다양한 정의에서 **"최적의"** 행동을 찾는 것에 관한 것.
- 문제를 더 잘 이해함에 따라 **Fomalization**(형식화)는 점진적으로 발전할 수 있다.

Decision Theory: High Level View

- Sequence of Events
  - ✓ 많은 문제들은 다음과 같이 **Fomalization**(형식화)됨.
    1. input x 관찰
    2. Take action a
    3. outcome y 관찰
    4. Outcome과 관련해 action을 **평가하는 방법**: $\ell(a, y)$

	X	A	Y
Linear regression	R^d	R^d	R^d
logistic regression	R^d	Number between $\begin{cases}1\\0\end{cases}$	$\begin{cases}1\\0\end{cases}$
support vector machine	R^d	Number of score	$\begin{cases}1\\0\\-1\end{cases}$

06 | Introduction to Statistical Learning Theory

Decision Theory: High Level View

- Some Formalization
  - ✓ The Spaces
    1. X: *input space*
    2. Y: *outcome space*
    3. A: *action space*

- Decision Function
  - ✓ **decision function** (or **prediction function**)은 input($x \in X$)을 취하고 action($a \in A$)을 생성.
  - ✓ $\mathrm{F}: \begin{cases} X \to A \\ x \to f(x) \end{cases}$

- Loss Function
  - ✓ **loss function**은 action($a \in A$)을 평가.
  - ✓ $\ell : \begin{cases} A \times Y \to R \\ (a, y) \to \ell(a, y) \end{cases}$

- 우리는 **action space**를 정의하고 이를 평가하는 것이 목표.

Statistical Learning Theory

- **통계 학습 이론을 위한 설정**
  - ✓ data generating distribution $\boldsymbol{P_{XY}}$가 있다고 가정.
  - ✓ P_{XY}는 모든 input/output 쌍 (x, y)를 $\boldsymbol{i.i.d.}$**(독립성, 등분산성)**로 생성.
  - ✓ $Decision\ function\ \boldsymbol{f(x)}$가 평균적으로 잘되려면 $loss\ function\ \ell(\boldsymbol{f(x)}, \boldsymbol{y})$가 보통 **작아야** 함.

- **The Risk Functional:**
  - ✓ $\boldsymbol{P_{XY}}$분포에서 추출한 새로운 **sample**(x, y)에 대한 $loss$의 **기댓값**을 의미.
  - ✓ The **risk** of a decision function $f : X \rightarrow A$ 로 가고

$$R(f) = \mathbb{E}\ell(f(x), y)$$ 의 형태.

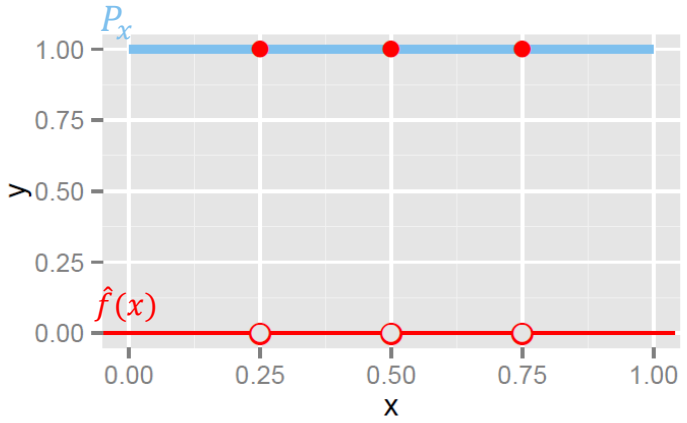
- 우리는 P_{xy}를 모르기 때문에(**모집단**을 모른다는 의미) **Risk function** 을 계산할 수 없다.
  - ✓ **Sample** data를 통해 추정 할 수 밖에 없음: [data set]$\boldsymbol{D_n} = ((x_1, y_1), \ldots, (x_n, y_n))\ i.i.d.\ from\ \boldsymbol{P_{XY}}$
- The **Bayes Decision Function:**
  - ✓ 기존에 가지고 있던 사전 정보를 활용하려 의사 결정을 할 때에 사용되는 이론.
  - ✓ 가능한 모든 함수 중에서 **최소한의 위험**을 달성하는 함수. ($f^*: X \to A$)
  - ✓ $\boldsymbol{f^*} = \underset{f}{\operatorname{argmin}}\, R(f)$

- Example 1: **Least Squares Regression(최소제곱법)**
  - ✓ Square loss :
    - $\ell(a, y) = (a - y)^2$ (Evaluate action in relation to the outcome: $\ell(a, y)$)
  - ✓ mean square risk:
    - $R(f) = \mathbb{E}[(f(x) - y)^2]$

- Example 2: **Multiclass Classification (다중 클래스 분류)**
  - ✓ 0-1 loss: $\ell(a, y) = 1(a \neq y) \coloneqq \begin{cases} 1\,, & if\ a \neq y \\ 0, & o.w. \end{cases}$
  - ✓ Misclassification error rate의 risk: $\mathrm{R}(f) = \mathbb{E}[1(f(x) \neq y)] = 0 * \mathbb{P}(f(x) = y) + 1 * \mathbb{P}(f(x) \neq y)$
    $= \mathbb{P}(f(x) \neq y)$

- **Empirical Risk (실증적 위험)**
  - ✓ P_{XY}그 자체로는 계산할 수 없음.
  - ✓ Let $D_n = ((x_1, y_1), \dots, (x_n, y_n))$ be drawn $i.i.d.$ from P_{XY}.
  - ✓ $\widehat{\boldsymbol{R}}_{\boldsymbol{n}}(\boldsymbol{f}) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$
  - ✓ $\lim_{n\to\infty} \hat{R}_n(f) = R(f)$ (큰 수의 법칙에 의거)
- Empirical Risk **Minimization**($\hat{f}$)
  - ✓ $\hat{\boldsymbol{f}} = \operatorname{argmin}_{f} \hat{R}_n(f)$

- Empirical Risk Minimization($\hat{f}$)
  - ✓ $\hat{f} = \underset{f}{\operatorname{argmin}}\, \hat{R}_n(f)$

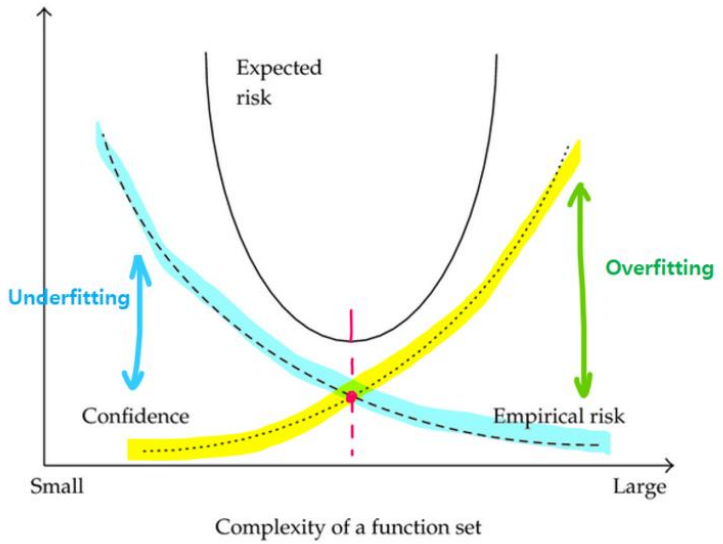


- $\boldsymbol{P_x} = Uniform[0,1],\ y = 1$ (i.e. Y는 언제나 1)
- $\boldsymbol{D_3}$={$(x_1, y_1), (x_2, y_2), (x_3, y_3)$} (from $\boldsymbol{P_x}$)(i.e. sample을 3개로 뽑음.)
- $\hat{f}(x) = 1(x \in \{0.25, 0.5, 0.75\}) = \begin{cases} 1, if\ x \in \{0.25, .5, .75\} \\ 0, \qquad\qquad o.w. \end{cases}$
- $\hat{f}(x)$에서 x =0.25, 0.5, 0.75 일 때 는 y =1이고 나머지는 0으로 학습된다.
- 0/1 loss에서 $\hat{f}$의 **Empirical Risk**(0/1 loss: $\ell(a, y) = 1(a \neq y) \coloneqq \begin{cases} 1, & if\ a \neq y \\ 0, & o.w. \end{cases}$):
  - ✓ $\begin{cases} \hat{R}_n(\hat{f}) = \frac{1}{n}\sum \ell(\hat{f}(x_i), 1), & if\ x_i \in \{0.25, .5, .75\} \\ \hat{R}_n(\hat{f}) = \frac{1}{n}\sum \ell(\hat{f}(x_i), 0), & o.w. \end{cases}$ 이므로 **Empirical Risk=0**이라 할 수 있다.

0/1 loss에서 $\hat{f}$의 Risk:

- ✓ $\hat{f}$의 모집단 P_X(Test data)가 train 3점에 대해서만 맞고 다 틀리기에 **Risk=1**이라 할 수 있다.

- Empirical Risk Minimization(**ERM**)
  - ✓ ERM은 **train data**에 대해서 학습했기에 test data에 대해 **overfitting**이 발생하게 된다.
  - ✓ One **approach**: "**Constrained ERM**":
    - ➢ 모든 의사결정 기능에 대한 경험적 위험을 최소화하는 대신,
    - ➢ **가설 공간**(Hypothesis Space)이라고 하는 **특정 부분 집합**으로 제한.
    - ➢ 제한함으로써 Overfitting을 방지하게됨.
  - ✓ **가설 공간**(Hypothesis Space):
    - ➢ 어떤 문제를 해결하는데 필요한 가능성 있는 **가설 후보군**의 집합을 의미.
    - ➢ A hypothesis space $\mathbb{F}$ is a **set** of functions mapping $X \to A$.



Statistical Learning Theory

- Constrained Empirical Risk Minimization(**CERM**)
  - ✓ Hypothesis space $\mathbb{F}$, a **set** of **[decision] functions mapping** $X \to A$.
  - ✓ **Empirical** risk minimizer (ERM) in $\mathbb{F}$ is

$$\hat{f}_n = \underset{f \in \mathbb{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

  - ✓ **Risk** minimizer in $\mathbb{F}$ is $f_{\mathbb{F}}^* \in \mathbb{F}$, where

$$f_{\mathbb{F}}^* = \underset{f \in \mathbb{F}}{\operatorname{argmin}} \mathbb{E}\ell(f(x), y)$$

Q&A

감사합니다.