Machine Learning in Practice
#2-1: Machine Learning Overview

Sang-Hyun Yoon

Summer 2019

Outline

Deduction vs. Induction

Given an algorithmic problem $f : X \to Y$,

Deductive approach to an algorithmic problem

1. Construct an algorithm $\mathcal{A}$ for f
2. **Prove** that $\mathcal{A}(x) = f(x)$ for all $x \in X$ (logical deduction)

Inductive approach (when $f(\cdot)$ is unknown or ill-defined)

1. Guess an (imprecise) algorithm $\mathcal{A}$ for f
2. For a set $\{(x_i, y_i) \in X \times Y \mid i = 1, \ldots, n\}$ of sample input/output pairs, **check** if $\mathcal{A}(x_i) = y_i$ for all i
   - where $y_i = f(x_i)$
3. **Induce** that $\mathcal{A}(x) = f(x)$ for all $x \in X$!
   - in the belief that the example set is sufficiently representative..
   - crux of inductive reasoning, though mathematically unsound..
4. Claim that $\mathcal{A}$ is a correct algorithm for the problem!!

Diversion: Scientific Induction vs. Mathematical Induction

Scientific induction: example

- **Claim**: All odd numbers > 1 are prime.
- **Inductive argument**: 3 is prime, 5 is prime, 7 is prime, 9 is not prime, 11 is prime, 13 is prime. So the claim is true!
  - 9 must be an experimental error which can be ignored..

Mathematical induction:

- $(p(0) \ \wedge \ \forall n \geq 0, \ p(n) \Rightarrow p(n+1)) \ \Rightarrow \ \forall n \geq 0, \ p(n)$

  for all predicate $p : \mathbb{N} \rightarrow \{\mathtt{T}, \mathtt{F}\}$
- 위 명제는 Peano arithmetic의 공리 (axiom)
  - ZFC set theory로부터도 유도 가능
- 수학적 귀납법은 위 공리를 이용한 deduction으로 앞 슬라이드 기준의 induction이 전혀 아님
  - 수학적 논증에서는 (scientific) induction이 허용되지 않음

Inductive Reasoning

- Although not allowed in math theorems, scientific theories are built upon the inductive reasoning.
  - e.g. $F = m\ddot{x}$ still governs the cutting-edge technologies!
- Useful as long as reliable predictions are made.
- "Learning" is inductive in the context of AI.

Inductive approach (when $f(\cdot)$ is unknown or ill-defined)

1. Guess an (imprecise) algorithm $\mathcal{A}$ for f
2. For a set $\{(x_i, y_i) \in X \times Y \mid i = 1, ..., n\}$ of sample input/output pairs, **check** if $\mathcal{A}(x_i) = y_i$ for all i
   - where $y_i = f(x_i)$
3. **Induce** that $\mathcal{A}(x) = f(x)$ for all $x \in X$!
   - in the belief that the example set is sufficiently representative..
   - crux of inductive reasoning, though mathematically unsound..
4. Claim that $\mathcal{A}$ is a correct algorithm for the problem!!

Inductive Learning & Machine Learning (supervised case)

Learning about a problem $f : X \to Y$

- Given $T \subseteq X \times Y$ (a set of sample input/output pairs),
  (where $y = f(x)$ for each $(x, y) \in T$)
  find an algorithm $\mathcal{A}$ such that $\mathcal{A}(x) \cong y$ for all $(x, y) \in T$.
  - If $\mathcal{A}(x)$ fits y $(= f(x))$ for all $(x, y) \in T$,
    we can accept $\mathcal{A}$ as a good approximator of f.

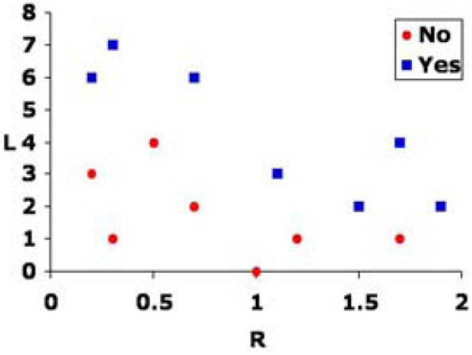
- Learning: sample inputs/outputs로 부터 algorithm $\mathcal{A}$를 찾기
  - sample inputs/outputs을 training data로 부름
  - training data로 부터 problem $f : X \to Y$를 **interpolation**하는 걸로 보면 됨

Manual Learning vs. Machine Learning

- Manual learning: 사람이 손으로 algorithm $\mathcal{A}$를 찾기
- Machine learning: 알고리즘으로 자동으로 algorithm $\mathcal{A}$ 찾기
  - Learning 알고리즘: algorithm $\mathcal{A}$를 찾는 meta-algorithm
- 우리의 목표는 manual learning이 아니고 machine learning

Example: Learning about the problem BANKRUPTCY

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



L: #late payments / year
R: expenses / income

Learning = finding an algorithm $\mathcal{A} : \mathbb{R} \times \mathbb{R} \rightarrow \{\text{Yes, No}\}$ s.t.

$$\mathcal{A}(3, 0.2) = \text{No} \quad / \quad \mathcal{A}(1, 0.3) = \text{No}$$

$$\mathcal{A}(6, 0.2) = \text{Yes} \quad / \quad \mathcal{A}(2, 1.9) = \text{Yes} \ \ldots$$

Outline

Machine Learning Systems

Learning as an optimization problem (supervised case)

Given training data $T \subseteq X \times Y$ (set of sample input/output pairs),

- solution space: $\mathcal{H} = \{\mathcal{A} : X \to Y\}$
  - i.e. set of algorithms under consideration (e.g. neural networks)
- cost function: $C : \mathcal{H} \to \mathbb{R}$
  - e.g. $C(\mathcal{A}) = \sum_{(x,y)\in T} \|\mathcal{A}(x) - y\|^2$ (i.e. squared-error)
- goal: find $\mathcal{A}^* \triangleq \operatorname{argmin}_{\mathcal{A}\in\mathcal{H}} C(\mathcal{A})$

Each machine learning system is characterized by

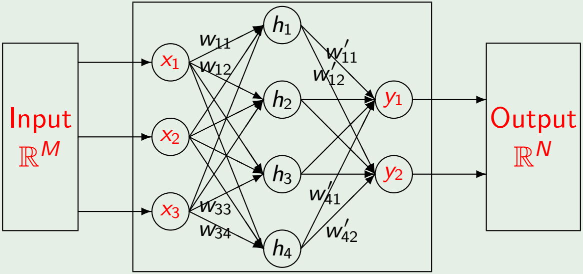
- $\mathcal{H}$: hypothesis space / representation / model architecture
- C : cost/evaluation/objective/scoring function
- how to compute argmin : optimization/learning algorithm

along with

- how to acquire data
  - 3 types: supervised / unsupervised / reinforcement

Machine Learning Systems: Example

Neural Networks



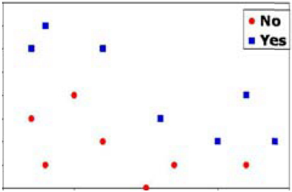
$$h_j = \sigma(\textstyle\sum_i w_{ij} x_i)$$
$$y_k = \sigma(\textstyle\sum_j w'_{jk} h_j)$$
where
$$\sigma(z) = 1/(1 + e^z)$$

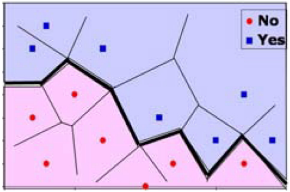
$$f_{\mathbf{w}} = (y_k)_k$$
$$\mathcal{H} \triangleq \{f_{\mathbf{w}} | \mathbf{w} = (w_{ij}, w'_{jk})\}$$

- $\mathcal{H}$: hypothesis space / representation / model architecture
  $$\mathcal{H} = \{f_{\mathbf{w}} \,|\, \mathbf{w} \in \mathbb{R}^{\text{very large}}\}$$
- C : cost/evaluation/objective/scoring function
  $$C : \mathcal{H} \to \mathbb{R} \,; \quad C(f) = \textstyle\sum_{(x,y) \in T} \|f(x) - y\|^2$$
- how to compute argmin : optimization/learning algorithm
  $$\mathbf{w} \Leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} C \; \left(\text{i.e. } w_{ij} \Leftarrow w_{ij} - \eta \frac{\partial C}{\partial w_{ij}} \text{ and } w'_{jk} \Leftarrow w'_{jk} - \eta \frac{\partial C}{\partial w'_{jk}} \right)$$
- Data acquisition type: supervised

Machine Learning Systems: Example

Nearest Neighbor





Given training data $T = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subseteq X \times Y$

- algorithm $\mathcal{A}$ "learnt" from T is

  $$\mathcal{A}(x) = y_i \text{ where } i \text{ is determined by}$$
  $$dist(x, x_i) = \min\{dist(x, x_j) \mid 1 \leq j \leq n\}$$

  - i.e. $\mathcal{A}$ is an algorithm for Voronoi diagram

- 즉, x에 "거리"가 가장 가까운 x_i를 선택해서 x_i에 대응되는 output y_i를 출력으로
- 위 그림은 $X = \mathbb{R}^2$, $Y = \{\mathtt{T}, \mathtt{F}\}$인 경우

Training data 획득 방식에 따른 3가지 분류

Supervised learning (가장 널리 사용)

- Training data T를 미리 모두 주고 최적의 $\mathcal{A} \in \mathcal{H}$를 찾음
- Input과 더불어 output도 주어져야 하는 현실적 어려움이
- Classification, regression 문제에 주로 사용

Unsupervised learning

- Input에 대응되는 output이 주어지지 않는 경우
- Clustering 문제, feature 추출, 차원 줄이기에 사용

Reinforcement learning

- Supervised learning의 일반화로 environment와 interaction 하면서 reward를 받는 형태 (cf. off-line vs. on-line algorithm)

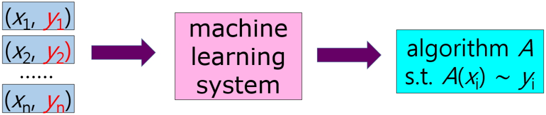
각 learning 기법들은 2가지 이상 방식으로 사용가능

- 예: AlphaGo의 convolutional neural network은 supervised와 reinforcement 방식 두가지를 사용

Supervised vs. Unsupervised

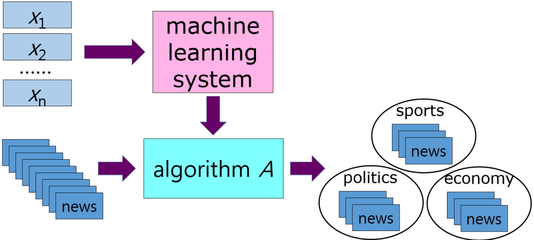
Supervised learning (지도 학습) (가장 널리 사용)

- Input과 output을 모두 주고 최적의 $\mathcal{A} \in \mathcal{H}$를 찾음
- Classification, **regression** 문제에 주로 사용



Unsupervised learning (비지도 학습)

- Input에 대응되는 output이 주어지지 않는 경우
- **Clustering** 문제, feature 추출, 차원 줄이기에 사용



Classification vs. Regression 문제 (supervised case)

Recall: Learning about a problem $f : X \to Y$ (supervised case)

- Given $T \subseteq X \times Y$ (a set of sample input/output pairs),
  (where $y = f(x)$ for each $(x, y) \in T$)

  find an algo $\mathcal{A} \in \mathcal{H}$ such that $\mathcal{A}(x) \cong y$ for all $(x, y) \in T$.
  - If $\mathcal{A}(x)$ fits y $(= f(x))$ for all $(x, y) \in T$,
    we can accept $\mathcal{A}$ as a good approximator of f.
  - Any better criteria? (Refer to "No free lunch theorem")

Classification problem: when Y is finite/discrete

- f : {photo of the Simpsons} $\to$ {"Homer", "Marge", "Bart"}

Regression problem: when Y is continuous

- f : {바둑 configuration} $\to \mathbb{R}$; $f(C) = C$에서 이길 확율

Foundational Issues (supervised case)

Consider a machine learning system for a problem $f : X \to Y$:

- $\mathcal{H}$: hypothesis space / model architecture / representation
- $C : \mathcal{H} \to \mathbb{R}$: cost/objective function
- optimization/learning algorithm to find $\operatorname{argmin}_{\mathcal{A} \in \mathcal{H}} C(\mathcal{A})$

Representability (related to $\mathcal{H}$)

Given any (training data) $T \subseteq X \times Y$, does there exist $\mathcal{A}_T \in \mathcal{H}$ s.t. $\mathcal{A}_T(x) \cong y$ for all $(x, y) \in T$?

Learnability (related optimization/learning algorithm)

If there exists such $\mathcal{A}_T \in \mathcal{H}$, can $\mathcal{A}_T$ be computed (in poly time)?

Generalizability (related to training data T)

For a $(x, y) \notin T$, $\mathcal{A}_T(x) \cong y$?

- Depends on the representativeness of T w.r.t. the problem f
  - Garbage In, Garbage Out (Refer to "No free lunch theorem")

Outline

On Representability

Recall: Representability **(related to $\mathcal{H}$)**

Given any (training data) $T \subseteq X \times Y$, does there exist $\mathcal{A}_T \in \mathcal{H}$ s.t. $\mathcal{A}_T(x) \cong y$ for all $(x, y) \in T$?

- 문제 f에 성격에 따라 따라 잘 표현해주는 모델구조 $\mathcal{H}$가 따로 있음
- 예: f가 image 분류 문제인 경우 CNN이 잘 맞음이 실험적으로 알려짐
- 예: 다차원 자료 분류 문제에 대해 nearest neighbor나 linear regression을 적용한다면..
- 모델구조 하나만으로 표현이 잘 안되는 문제 f도 존재하며 이 경우 여러 모델을 혼합(model ensemble)해서 사용하기도 함

On Generalizability

Recall: Generalizability (related to training data T)

For a $(x, y) \notin T$, $\mathcal{A}_T(x) \cong y$?

- Depends on the representativeness of T w.r.t. the problem f
  - Garbage In, Garbage Out (Refer to "No free lunch theorem")

- Training data T가 문제 f를 잘 반영해주는 "대표성 있는" 자료여야 T를 바탕으로 얻은 $\mathcal{A}_T$도 f와 유사해짐
  - 예: f가 얼굴 인식 문제인 경우 T에 남자 어린이 사진만 포함되어 있고, 성인 또는 여자 사진이 드문 경우..
- 문제 f와 잘 맞지 않는 모델 구조 $\mathcal{H}$를 사용한다면 generalizability를 따지기 전에 representability에서 걸림
  - generalizability를 논하기 위해서는 representability에서 문제가 없는 것이 선행되어야 함

Occam's Razor

Occam's Razor

- "필요하지 않은 경우까지 가정하면 안된다"
- "간단한 논리로 설명가능하면, 복잡한 논리를 세우지 말라"
- "같은 현상을 설명하는 두 개의 주장이 있다면, 간단한 쪽을 선택하라"

현대 (실험적) 과학을 구성하는 기본적 지침 (예: $F = ma$)

근거

- 모델이 불필요하게 복잡하면 overfitting 발생
- 예: 일차 선형 함수로 근사화 할 수 있는 $T \subseteq \mathbb{R}^2$를 100차 다항 함수로 근사화 한다면.. (twiddling lots of knobs)

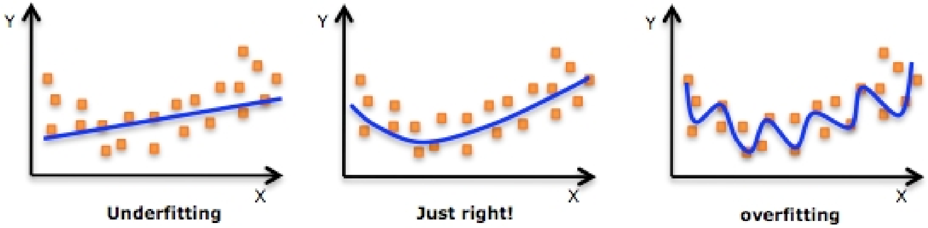
Representability와 generalizability 모두와 연관됨

- 지나치게 간단하면 representability에 문제가 생김
- 필요 이상으로 복잡하면 overfitting 발생

Occam's Razor

Representability와 generalizability 모두와 연관됨

- 지나치게 간단하면 representability에 문제가 생김
- 필요 이상으로 복잡하면 overfitting 발생



Cross Validation

Recall: Generalizability (related to training data T)

For a $(x, y) \notin T$, $\mathcal{A}_T(x) \cong y$?

- Depends on the representativeness of T w.r.t. the problem f

- 가용한 training data $T \subseteq X \times Y$를 모두 사용해서 T를 완벽하게 반영하는 $\mathcal{A}_T$를 구한 경우 어떤 문제가?
- Overfitting으로 인한 generalizability 문제 발생 여부를 알 수 없게 됨
  - T외에는 f와 관계된 자료가 없음
- T를 적절히 training용과 testing (for generalizability)용으로 분할하여 사용하면 위 문제는 해결 가능
  - 분할 후 training-testing을 여러번 하면 더욱 바람직
- Generalizability에 문제가 있음이 감지되면 모델구조를 적절히 바꿔주면 됨

On Learnability

Reall: Learnability (related optimization/learning algorithm)

If there exists such $\mathcal{A}_T \in \mathcal{H}$, can $\mathcal{A}_T$ be computed (in poly time)?

- Cost 함수 $C : \mathcal{H} \to \mathbb{R}$을 최소화하는 $\mathcal{A}_T$를 찾는 것은 대부분 경우 intractable
  - $\mathcal{H}$가 neural networks인 경우 NP-hard
- 하지만, 모델구조와 cost 함수는 어차피 정확하지 않으므로 굳이 cost 함수를 최소화 하는 $\mathcal{A}_T \in \mathcal{H}$를 찾을 필요는 없음
- "적당히" 최적에 가까운 $\mathcal{A}_T$를 찾아주면 충분