Basic Machine Learning: Supervised Learning, 강의 내용 정리

Index

Overview

Machine Learning은 어떤 것일까?

- Algorithm이란?
  - 명령어의 집합이며 명령어들을 하나하나 실행했을 때, 우리의 문제를 해결해주는 것
- Traditionally
  1. input이 주어지면
  2. 프로그램을 통해
  3. **output**을 구한다.
- Machine learning
  1. 데이터가 주어지면(input and output)
  2. machine learning algorithm을 훈련시켜
  3. input을 넣으면 원하는 output이 나오는 **프로그램**을 구한다.

Supervised Learning - Overview

supervised learning을 할 때, 주어지는 것은 무엇이고 어떤 과정을 거쳐 결론을 내는지 알아보자.

- Provided:
  - a set of N input-output 'training' examples
  - A per-example loss function
  - Evaluation sets: validation and test examples
- What we must decide:
  - Hypothesis sets
    - architecture와 그에 속한 parameter로 구성된 space
  - Optimization algorithm
    - 어떻게 학습할지를 결정
- Supervised learning finds an appropriate algorithm/model automatically
  1. Training
  - $\hat{M}_m = argmin_{M \in H_m} \sum_{n=1}^{N} l(M(x_n), y_n)$

        - 한 방법에 대해 적합한(파라미터가 잘 조절된) 모델을 찾는다.
    2. Model Selection
    - $\hat{M} = argmin_{M \in \{\hat{M}_1, \hat{M}_2, \cdots, \hat{M}_M\}} \sum_{(x,y) \in D_{val}} l(M(x), y)$
        - 좋은 모델을 찾는다.
    3. Reporting
    - Report how well the best model would work using the test set loss.
    - $R(\hat{M}) \sim \frac{1}{|D_{test}|} \sum_{(x,y) \in D_{test}} l(\hat{M}(x), y)$

    - 중요한 부분은 1, 2, 3을 진행하는 데이터들이 구분되어야 한다는 것이다.

- It result in an algorithm $\hat{M}$ with an expected performance of $R(\hat{M})$

Three points to consider both in research and in practice

1. How do we **decide/design** a hypothesis set?
    - 문제에 적합한 방법들은 어떤 것들이 있고 무엇을 사용할 지
2. How do we decide a **loss function**?
    - 알려진 loss function 외에 문제에 적합한 loss function은 어떻게 정의할지?
3. How do we **optimize** the loss function?
    - 그러한 loss function을 어떻게 optimization할지?

강의에서는 딥러닝에 한정해서 설명한다.

Hypothesis Set

neural network는 무엇이며, 우리가 신경써야 하는 부분은 무엇일지 알아보자.

neural network는 An directed acyclic graph라고 할 수 있는데, 각 leaf node부터 계산을 시작하여 root node까지 forward computation을 한다. 현재 pytorch, tensorflow등을 통해 각 노드들의 계산 혹은 미분은 추상화되어 잘 정의되어 있기 때문에 이 패키지를 사용하는 입장에서는 neural network, 즉, an directed acyclic graph를 design하고 hypothesis set을 만드는 것이 중요하다고 할 수 있다.

Note

- Forward computation(구조를 사용하는 것): how you "use" a trained neural network.
- Implication in practice
    - Naturally supports high-level abstraction
    - Object-oriented paradigm fits well.
        - Base classes: variable (input/output) node, operation node, ...
        - Define the internal various types of variables and operations by inheritance
    - Maximal code reusability
- You define a hypothesis set by designing a directed acyclic graph.
- The hypothesis space is then a set of all possible parameter settings.

Loss Function - Preview

train한다는 것은 문제에 맞는 loss function의 값이 최대한 작아지게 하는 것이라고 할 수 있다. 하지만 존재하는 loss function은 해결하고자 하는 문제에 이상적으로 적합하지 않는 경우가 많기 때문에 문제에 따라 loss function을 문제에 맞게 정의해야하는 순간들이 많다. 하지만 output이 distribution based라면 negative log-probability를 통해 loss function을 자연스럽게 정의할 수 있다.

따라서, 이후 강의를 통해 distribution-based loss functions를 정의하는 방법을 배우자

Probability in 5 minutes

확률의 기초를 학습하자.

이벤트 셋이란, 확률 변수란, 확률이란, 확률의 특성이란?

- An "event set" Ω contains all possible events: $\Omega = \{e_1, e_2, \cdots, e_D\}$
  - Discrete: when there are a finite number of events $|\Omega| < \infty$
  - Continuous: when there are infinitely many events $|\Omega| = \infty$
- A "random variable" X could take any one of these events: $X \in \Omega$
- A probability of an event: $p(X = e_i)$
  - How likely would the i-th event happen?
  - How often has the i-th event occur relative to the other events?
- Properties
  1. Non-negative: $p(X = e_i) \geq 0$
  2. Unit volume: $\sum_{e \in \Omega} p(X = e) = 1$

Multiple random variables란, joint probability란, conditional probability란, mariginal probability란?

- Multiple random variables: consider two here - X, Y
- A joint probability $p(Y = e_j^Y, X = e_i^X)$
  - How likely would e_j^Y and e_i^X happen together?
- A conditional probability $p(Y = e_j^Y | X = e_i^X)$
  - Given e_i^X, how likely would e_j^Y happen?
  - The chance of both happening together divided by that of e_i^X happening regardless of whether e_j^Y happend:
    - $p(Y|X) = \frac{p(X,Y)}{p(X)} \iff p(X,Y) = p(Y|X)p(X)$
- Probability function $p(X)$ returns a probability of X(marginal probability)
- A marginal probability $p(Y = e_j^Y)$
  - Regardless of what happen to X, how likely is e_j^Y?
    - $p(Y = e_j^Y) = \sum_{e \in \Omega_x} p(Y = e_j^Y, X = e)$
- malginalization
  - 동전 예시, 첫번째 동전의 결과값: X, 두번째 동전의 결과값: Y
  - 알고 있는 것 $p(X, Y)$, 구하고 싶은 것 $p(Y)$

Loss Function

neural network가 conditional distribution을 output으로 출력한다면, negative log probability를 통해 loss function을 잘 정의할 수 있다.

output으로 다음과 같은 distribution을 만들수 있다.

- Binary classification: Bernoulli distribution
- Multiclass classification: Categorical distribution
- Linear regression: Gaussian distribution
- Multimodel linear regression: Mixture of Gaussians

output을 어떻게 하면 distribution으로 만들수 있을까?

- Bernoulli distribution(output: sigmoid), Categorical distribution(output: softmax), ...

Loss Function - negative log-probability

neural network의 output이 conditional distribution $p_\theta(y|x)$을 출력한다면 우리는 자연스럽게 loss function을 정의할 수 있다.

당연하게도 x_n이 주어졌을 때, $p_\theta(y_n|x_n)$이 높아지는 θ를 구하는 게 목적이 되고 이를 수식으로 표현하면 다음과 같다.

- $argmax_\theta \; log \, p_\theta(D) = argmax_\theta \sum_{n=1}^{N} log \, p_\theta(y_n|x_n)$
- 이때 $p_\theta(y_n|x_n)$의 합을 최대화해야하는데 이는 $-p_\theta(y_n|x_n)$의 합을 최소화하는 것과 같다 (negative log-probabilities). 이를 통해 마지막으로 loss function을 수식화하면 다음과 같다.
- A loss function is the sum of negative log-probabilities of correct answers.

  $L(\theta) = \sum_{n=1}^{N} l(M_\theta(x_n), y_n) = -\sum_{n=1}^{N} log \, p_\theta(y_n|x_n)$

Optimization Methods

output이 distribution output이 되게 한다면, loss function이 negative log-loss로 계산되게 할 수 있다. 이제 문제는 "이 loss function을 어떻게 minimization하는가"이다.

- hypothesis space에는 무수히 많은 model들이 있는데, 모든 것을 다 시도해보고 최적인 것을 고르기는 어려움.
- 따라서, 아무 곳을 선택한 후에 loss를 낮추는 방향으로 최적화를 진행
  - Local, Iterative Optimization: Random Guided Search
    - 장점: 어떤 비용함수를 사용해도 상관 없음.
    - 단점: 차원이 커질 수로 사용하기 어려움
  - Gradient-based Optimization:
    - 미분을 통해 최적화 할 방향을 정한다.
    - 장점: local minimum에 빠질 수 있지만 그래도 loss를 확실하게 낮출 수 있음.
    - 단점: Random Guided search에 비해서 탐색영역이 작음. 학습률에 따라 최적의 값으로 갈 수도 있고 못갈 수도 있음

Backpropagation

optimization을 진행하기 위해서는 이제 gradient를 구하는 것이 목적이다.

architecture의 loss function은 결국 input을 변수로 가지는 함수와 이를 변수로 갖는 함수들의 합성이라고 이해할 수 있다. 이 합성함수를 미분한다면 chain rule를 사용할 것이고 이는 op node의 derivatives만 안다면 합성함수의 미분값을 구하는게 어려운 일이 아니다.

pytorch나 tensorflow에서 제공하는 각 op node의 derivatives 덕분에 loss function's derivatives를 구하는 것이 사용자입장에서는 더 이상 어려운 일이 아니게 되었다($O(n)$의 비용 where N: # of node).

Gradient-Based Optimization

train 데이터 전부를 보고 Backpropagation을 통해 loss funtion을 낮추는 gradient를 계산하기엔 연산상의 문제가 있기 때문에 gradient를 구하는(추정하는) 방식인 stochastic gradient descent가 제안되었다.

- stochastic gradient descent: Approximate the full loss function (the sum of per-examples losses) using only a small random subset of training examples:
  - $\nabla L \approx \frac{1}{N'} \sum_{n=1}^{N'} \nabla l(M(x_n, y_n))$
- Unbiased estimate of the full gradient.
- Extremely efficient de facto standard practice.

Stochastic gradient descent in practice

1. Grab a random subset of M training examples
   - $D' = \{(x_1, y_1), \cdots, (x_{N'}, y_{N'}\}$
2. Compute the minibatch gradient
3. Update the parameters
   - $\theta \leftarrow \theta + \eta \nabla(\theta; D')$
4. Repeat until the validation loss stops improving.
   - validation의 loss가 더 떨어지지 않으면 early stop을 해야한다.
   - An efficient way to prevent overfitting
     - Overfitting: the training loss is low, but the validation loss is not
     - The most serious problem in statistical machine learning
     - one of the solutions: **Early-stop** based on the validation loss

- 추가로 θ가 업데이트 되는 정도를 결정하는 learning rate를 구하는 방법도 많은 연구가 되어있다. 너무 작으면 학습이 너무 오래 걸리고 너무 크면 최적점을 뛰어넘을 수도 있기 때문이다.
  - 한가지 해결방안은 Adaptive learning rate을 구하는 것이다.
    - Adam, Adadelta, etc, ...
- 최적의 learning rate를 찾기 위해서 여러 learning rate를 시험해보는 것도 중요하지만 초반 prototype을 만들 때는 adam, adadelta 등을 통해 다른 가능성을 지우는 것도 중요함.

Summary

Supervised Learning with Neural Networks

1. How do we decide/design a **hypothesis set**?
   - Design a network architecture as a directed acyclic graph
2. How do we decide a **loss function**?
   - Frame the problem as a conditional distribution modelling
   - The per-example loss function is a negative log-probability of a correct answer
3. How do we **optimize** the loss function?
   - Automatic backpropagation: no manual gradient derivation
   - Stochastic gradient descent with early stopping [and adaptive learning rate]