**우리는 실무와 같이 구현을 해봐야한다!**

**필요모듈 예시**

Model.py: Architecture가 정의된 클래스

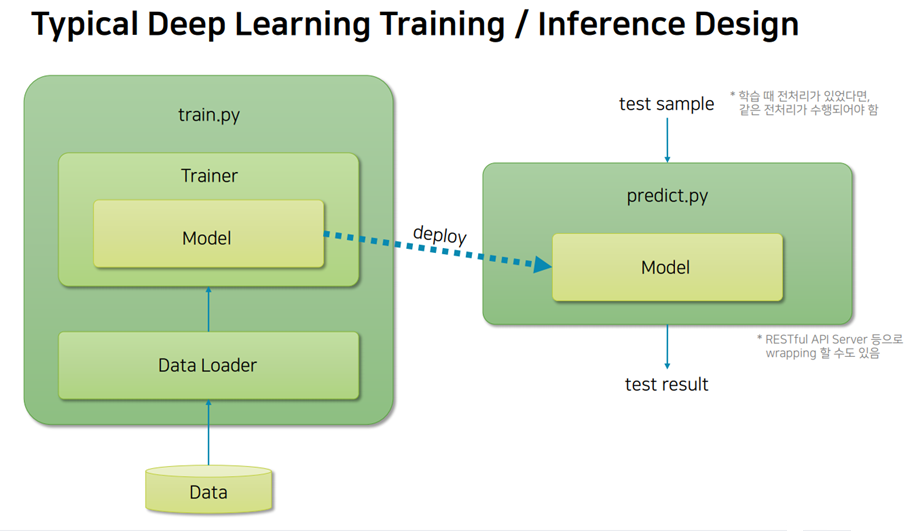
Train.py: model을 학습하기 위한 코드: epoch이 여기서 돈다!

Datalaoder.py: 데이터를 불러와 전처리를 수행하고, 신경망에 넣기 좋은 형태로 변환: Dataloader, Transforms등의 모듈을 여기서 사용한다

Train.py: 사용자로부터 Hyper-parameter을 받아 model과 trainer, Loader를 선언하고 학습: argparser를 여기서 한다. 또한 CLI환경에서 실행하는 모듈이기도 하다.

Predict.py: 사용자로부터 MODEL과 input을 받아 추론을 수행한다. 눈으로 보고 확인할 수 있다!

**Typical Deep Learning Training/ Inference Design**



Trainer에서 epoch을 전개하고, feed-forward, backpropagation을 진행한다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

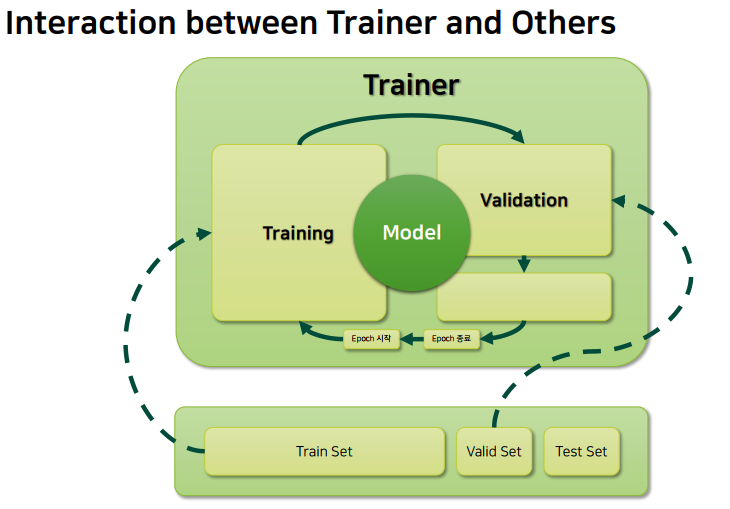
자동 생성된 설명

Epoch: 전데 데이터셋에 대한 한번 학습완료한 사태

Batch size: 한번의 배치마다 주는 데이터 샘플 size ex) 700개의 데이터를 7개로 나누면 batch size는 100, Iteration은 100 -> batch size \* Iteration = Data 수

Iteration: epoch을 나누어서 실행하는 횟수 (Training에서만 수행) ex) 10,000개 데이터를 1,000개씩 5번 실행하면 batch size = 1000, Iteration = 10, Epoch = 5.

gradient descent수행은 자신이 가지고 있는 웨이트 파라미터를 가지고 수행한다.



Train set: 훈련을 위해

Valid set: 파라미터 수정을 위해

Test set: 성능평가를 위해

# Feature: 샘플을 잘 설명하는 특징

딥러닝은 이 특징 추출을 최소한의 전처리를 통해(scale) 데이터에 모델에 넣고 수행하고, end- to end 방식으로 할수있다. 또한 사람이 발견하지 못하는 특징도 찾을 수 있지만, 해석하기가 어렵다.

End to end: Input -> Learning Algorithm -> output

Feature Vector: 각 특징을 모아서 하나의 vector로 만드는 것 -> Tabular dataset의 각 row도 이에 해당한다.

각 차원은 어떤 속성에 대한 level을 나타낸다.

feature vector를 통해 샘플 사이의 거리(유사도)를 계산할 수 있다. -> 자연어처리에서 word2vec

# One-hot Encoding:

의미를 갖는 integer대신 1개의 1과 n-1개의 0으로 이루어진 n차원의 벡터. 1이 하나고, 나머지가 0이다. – Categorical Value를 나타내기 위한 방법

서로다른 두 벡터는 항상 직교한다. (element wise의 곱이 0이 된다.)

Cosine similarity가 0. 따라서 두 샘플 사이의 유사도(거리)를 구할 수 없다.

->머신러닝 에서는 유사도 개념은 중요하지만, one-hot encoding은 불가능하다. Word embedding 을 통해 나중에 vector로 표현하고 유사도를 측정한다.

\* Categorical Value vs Continuous Value

- Categorical Value: 보통은 discrete value(이산확률변수), 단어 클래스(범위가 주어져 있다)

우리가 가지고 있는 의미를 반영하지 못한다. 값이 비슷해도 비슷한 의미를 갖지 않는다.

- Continuous Value: 키, 몸무게

-> 값이 비슷하면 비슷한 의미를 갖는다

# Sparse vs Dense Vector

Sparse Vector: Vector의 element가 대부분 0인경우. Ex) One got- Encoder vector.

Dense Vector: 반대개념 element가 대부분 1인경우.

# Motivation of Embedding Vectors

NLP에서는 단어는 categorical and discrete value의 속성을 가진다. 따라서 one-hot representation으로 표현한다. 하지만, 이는 실제 존재하는 단어 사이의 유사도를 표현할 수 없다

* Word Embedding Vectors

Word2vec 또는 DNN을 통해서 차원 축소 및 dense vector로 표현을 통해 단어 유사도를 표현할 수 있다!

Categorical Value는 One-hot Encoding을 통해 벡터로 표현된다. NLP에서 Sparse Vector는 벡터간 유사도 계산이 어렵다. 따라서 단어간의 유사도를 표현하기 위해서 Dense Vector로 표현할 필요가 있다(Vector Embedding: Word2vec, DNN).

AutoEncoder

딥러닝의 특징을 잘 반영한 architecture로, 인코더(encoder)와 디코더(decoder)를 통해 압축과 해제를 실행한다. 인코더는 입력(x) 정보를 최대한 보존하는 방식으로 손실 압출을 수행하고, 디코더는 중간 결과물(z)의 정보를 입력(x)와 같아지도록 압축 해제(복원)을 수행한다. 이 과정에서 필요 없는 정보를 버린다.

복원을 성공적으로 성공하기 위해, 오토 인코더는 특징을 추출하는 방법을 자동으로 학습한다.

텍스트, 폰트, 포스트잇 노트, 직사각형이(가) 표시된 사진

자동 생성된 설명 \* Encoder: 복원에 필요한 정보를 중심으로 손실압축을 수행. 필요 없는 정보(뻔한 특징)은 버릴 수 있다. Encoder내에서도 많은 layer들이 존재할 것이다. 각 layer의 결과물을 hidden vector라고 하며, 샘플들의 feature들을 가지고 있다. Layer(신경망)를 통과시키는 것은 입력공간(input space)에서 잠재공원(latent space)로의 맵핑 과정이다(고차원 공간 -> 저차원 공간). 이 hidden vector를 해석하는 것은 매우 어렵지만, 비슷한 sample은 비슷한 hidden representation을 가질것이다.

\* Bottleneck: 입력 x에 비하여 작은 차원으로 구성되어 있다. 정보의 선택과 압축이 발생하며, 차원에 따라 압축되는 정도를 결정한다. 그러므로 z는 입력(x)에 대한 feature vector라고 할 수 있다. 압축 효율이 높아야 하므로 입력에 비해 dense 하다. z는 x를 복원하기 위한 정보를 담고 있다.

\* Decoder: 압축된 중간 결과물 z를 바탕으로 최대한 x와 비슷하게 복원한다. (X^, 보통 MSE LOSS로 최적화를 수행한다.) 뻔한 정보는 주어지지 않더라도 알 수 있기 때문에 복원이 가능하다.

- 오토 인코더는 압축과 해제를 반복하며 특징 추출을 자동으로 학습한다. 필요 있는 정보와 필요 없는 정보를 구분할 수 있게 되는 것이다. (불필요한 feature을 삭제한다.)

- 인코더로부터 나온 중간 결과물(z)은 입력(x)에 대한 feature vector이다. – Embedding Vector

- 인코더에 통과시키는 것은 feature vector에 대한 embedding과정이라 볼 수 있다.

**확률적 관점에서의 딥러닝**

가상의 함수를 모사하여, 원하는 출력값을 반환하는 신경망의 파라미터를 만들자 – Gradient Descent, Back-propagation, Loss, Optimizer, Feature Vector… -> 이제는 우리의 생각을 확장할 때!

Before: 함수를 배우자! -> After: 확률 분포 함수를 배우자!

Why? 수학적으로 좀더 설명 가능, 불확실성까지 학습한다.

Neural Network는 확률 분포 함수를 모델링 할 수 있다. 이를 통해, 가상의 확률 분포함수p(y|x) (x가 주어졌을 때 y의 확률)을 근사한다.

따라서 우린 MLE(Maximum Likelihood Estimation), Cross Entropy, KL-Divergence, Log softmax, NLL(Negative Log Likelihood)를 학습해야 한다.

**1. 확률변수**

랜덤 변수는 확률을 이야기할 때 랜덤하게 발생하는 어떤 사건을 정의한다.

주사위 사건을 던졌을때(사건x) 주사위값이 3이 나왔다면, P는 확률을 의미하며, 괄호 안의 확률 변수가 특정 값을 가질때 확률값을 반환하는 함수다.

이 때, 확률 p는 0에서 1 사이다

이 때 확률 변수 x가 가질수 있는 N개의 값에 대한 확률을 모두 더하면 1이 된다.

**1-1 이산확률변수와 이산확률분포**

랜덤변수는 불연속적인 이산(discrete)값인 경우가 많다. 하나의 예로는 주사위가 된다.

라인, 도표, 평행, 직사각형이(가) 표시된 사진

자동 생성된 설명확률 질량함수

이처럼 불연속적인 랜덤 변수를다루는 이산 확률 분포는 확률 변수가 정의된 공간에서 나타날 수있다. **불연속적인 이산 확률 변수에 대한 확률 함수를 확률 질량함수라고한다.**

​이상적인 확률 변수를 같은 확률분포로는 베르누이 분포와 멀티눌리 분포가있다.

베르누이 분포(=이항분포): 0과 1 두개의 값만 가질 수 있다.

멀티눌리 분포(=다항분포): 주사위의 경우와 같이여러 개의 이산적인 값을 가질 수 있다.

​

**1-2 연속확률변수와 연속확률분포**

연속적인 값을 다루는 연속 확률 변수를 가지는 확률 분포를 생각할 수 있다. 이것을 연속확률 분포라 부른다. 예를들면 주사위가 정육면체가 아닌 아닌 하나의 '구'라고 상상하는 것이다. 이떄는 점을 예측하는것보단 영역(구간)에 대해 확률값을 계산하는것이 편하다.

연속 확률 분포에 대해서는 확률 질량함수 대신 확률 밀도함수를 통해 확률변수로 정의된 공간에 대해 확률값이 아닌 확률 밀도를 정의한다.

확률밀도 p(x)는 꼭 1보다 작을 필요는 없으며, p(x)를 적분한 값은 항상 1이다.

확률 분포 함수를 연속적인 확률변수, 즉 연속 확률 변수에 대한 확률 밀도 함수라고 한다.

정규분포라고 불리는 가우시안 분포가 좋은 예이다.

라인이(가) 표시된 사진

자동 생성된 설명확률 밀도 함수

연속적인 확률 분포함수에서 확률값이란 보통 어떠한 구간의 값을 말한다.

가우시안 분포: 정규분포: 어떤 구간의 넓이: 함수의 넓이가 확률분포: 연속확률분포

|  |  |  |
| --- | --- | --- |
| 이산적인 확률 변수 = 확률 질량 함수 | 베르누이(Bernoulli's) 확률분포 = 이항분포 | 0과 1(함수값의 확률) |
| 멀티누이(Multinoulli) 확률분포 = 다항분포 | 여러 개의 이산값(함수값이 확률) |
| 연속 확률 변수 = 확률 밀도 함수 | 가우시안 확률분포 = 정규분포 | 어떤 구간의 넓이(함수의 넓이가 확률) |

​**2. 결합확률, 독립확률**

결합 확률이란 두개 이상의 사건이 동시에 일어날 확률이다.

주사위 두개를 던지개 되었을때, A=3, B=2가 일어날 확률을 보자

주사위 두개를 던질 확률​

*A*=3, *B*=2가나올확률​, 는 P(A∩B)와 비슷한 개념이다.

이때, 2개의 주사위를 던지는 경우에는 서로에게 영향을 끼치지 않는다. 이것을 독립이라 한다.

독립확률일때는 다음과 같은 식이 따른다.

**3. 조건부확률**

조건부확률은 머신러닝과 딥러닝에서 가장 중요하다. 주어지는 문제 task가 확률 기반하기 때문이다. 다만 독립시행과 달리 조건부 확률은 하나의 확률변수가 주어졌을 때 다른 확률 변수에 대한 확률분포다. 여기서 는 사건B가 주어졌을 때 A가 일어날 확률이다.

​

**\*베이즈정리(Bauyes Theorem) : 데이터 B가 주어졌을 때, 가설 A확률**

두 확률 변수의 사전 확률과 사후 확률 사이의 관계를 나타내는 정리다. 사전 확률로 사후 확률을 구할 수 있다.

이 식을 도출할 수 있다. 간단하게 위에 식 중, P(A∩B) = P(B|A)\*P(A) = P(A|B)\*P(B)이다.

여기서 P(B|A)\*P(A) = P(A|B)\*P(B)을 이항 시키면가 된다.

여기서 P(A|B)는 사후확률(posterior), P(A)는 사전확률(prior)이다. P(B)는 그냥 사건 B다. 조건부 확률이 아닌 상수 같은 느낌이다.

**3. MLE(Maximum Likelihood Estimation, 최대우도법)**

우리는 가상의 확률분포를 잘 모사하는 확률분포 파라미터(θ)를 찾고싶다. 목표 확률로부터 데이터를 수집한 후, 데이터를 잘 설명하는 파라미터를 찾자! -> **데이터를 잘 설명하는 확률 분포의 파라미터를 찾자!**

Likelihood라는 값을 통해 얼마나 잘 설명하는 변수인지 알 수 있다. Gradient accent를 통해 찾자!

**MLE는 주어진 데이터가 특정 확률 분포에서 가장 높은 확률값을 갖도록 모델의 파라미터를 조정, 추정하여 가능도를 최대화하는 방법이다.** MLE는 입력으로 주어진 확률분포(파라미터 )가 데이터를 얼마나 잘 설명하는지 나타내는 점수(Likelihood)를 출력한다. 데이터가 해당 확률 분포에서 높은 확률 값을 가진다.

최대우도추정법에서는 특정한 통계 모델과 변수에서 데이터가 지금의 모습과 같이 나타날 확률을 극대화할 수 있는 파라미터(모수)의 값을 찾는다. 최대우도추정 문제를 기호를 통해 다음과 같이 간단히 나타낼 수 있다.

즉, 주어진 데이터를 잘 설명하기 위해서 MLE가능도를 최대화하도록 θ를 추정하는 것이다.

여기서 θ는 모수, 파라미터를 뜻하며 이것을 측정하는 것이다. MLE는 이산적 확률변수, 연속확률변수 둘 다 쓰인다.

이처럼 가능도(Likelihood)는 주어진 데이터 = {x\_1, x\_2, ..., x\_n}를 설명하는 확률분포 파라미터 θ에 대한 함수로 다음과 같이 표현한다. 따라서 이산 확률변수를 갖는 확률 분포에서 확률값 자체가 가능도로 표현할 수 있으며, 연속확률 변수를 갖는 확률분포의 경우에는 확률밀도값이 가능도를 나타낸다. 서로 독립인경우 n번의 시행을 거쳐 얻은 데이터(x\_1, x\_2 ... x\_n)에 대한 가능도는 다음과 같이 나타난다. 이때, Likelihood에서 가장 높은 값을 MLE라고한다.

이때 여기에 로그를 취할 수 있다.

수학적으로 조건부표기는 같다

이처럼 로그를 취하게 되면, 추후 소수점이 너무 작게 표현되어 언더플로(underflow)현상이 발생하는것을 방지할 수 있고, 덧셈 연산은 곱셈 연산보다 빠르므로 여러 가지 이점이다. 가장 큰 이점은 가우시안 분포에서의 지수를 제거할 수 있다는 것이다. 우리는 가능도에 로그를 취하여 로그 가능도를 최대화하도록 한다. MLE는 신경망 또한 확률분포함수이기 때문에, 신경망의 마지막 층의 소프트맥스을 통해서, 연산 y의 결과값은 클래스별 확률 값에 대한 분포 y^을 반환한다.

**NLL(Negative Log Likeligood) : 계산의 편리성을 위해 사용!**

만약 여기에 -1을 곱하면 최소화 문제로 치환할 수 있다. 이때, -1이 곱해진 로그 가능도를 음의 로그 가능도NLL(Negative Log Likelihood)라고 부른다. NLL값을 최소화하면 MLE를 하는것과 같은 효과를 얻을 수 있다. MLE최대화 = NLL최소화 -> 오차최소화

​

다중분류 문제는 softmax, 이중분류 문제는 sigmoid

​

∇: 델 연산자는 미분 연산자와 마찬가지로 그래디언트를 하나의 연산자로 바라본 것이다. 델 연산자는 벡터 미적분학에서 많이 쓰이는 연산자로써 나블라 기호로 표현하며 함수의 발산이나 회전 등을 나타내는데 사용된다. 신경망 가중치 파라미터 θ가 훈련 데이터를 잘 설명하도록 경사하강법을 통해 MLE를 수행하여 학습한다.

​

요약하자면, MLE를 통해 수집한 데이터셋을 가장 잘 설명하는 확률분포의 파라미터를 추정한다.

Neural network도 확률 분포 함수이므로 MLE를 통해 파라미터를 찾는다.

방법으로는 MLE를 최대화하는 방법(Gradient Accent을 통해)과, NLL을 최소화하는 방법(Gradient Descent를 통해)이 있다. Gradient Descent를 하기위해선 미분이 필요하다. 이것을 효율적으로 하기위해 backpropagation을 활용한다.

수식은 다음과 같다

NLL minimize 하는것은 Cross Entropy를 minimize 하는것과 같다.

**4. 정보이론에서 정보량Entropy, Cross Entropy(크로스 엔트로피)**

정보이론이란 데이터를 정량화하기 위한 응용 수학이다. 정보이론 또한 신경망과 밀접한 연관이 있다. 흔히 정보량이라는 표현을 사용한다. 정보량은 불확실성 또는 놀람의 정도를 나타낸다. 정보량이 높다는것은 어떤일이 일어날 확률이 낮은것을 의미한다.

예를들어 누군나 뻔히 예측하거나 재현할수 있는것으로는 작은 정보만으로도 표현할수 있다. 일어날 확률이 낮은일에 대한 것일수록 많은 정보를 갖고 있다. 확률이 낮은 사건에 대해 서술한 문장이 맞을수록 그 정보는 굉장히 소중해진다.

쉽게 말해서! 말도 안 되는 일이 일어난다 -> 정보량이 많다는 것이다.

예를들면,

해가 동쪽에서 뜬다 -> 정보량이 낮다!

해가 서쪽에서 뜬다 -> 정보량이 높다!

위 수식은 정보량의 수식이다.

​

**4-2 엔트로피(정보량의 평균)**

정보량의 평균(기댓값)을 취할수 있는데 이를 엔트로피(Entropy)라고 한다

엔트로피는 분포의 대략적인 모양이 얼마나 펴져있는지, 뾰족한지를 가늠해볼수 있는 척도다. 보통 앤트로피가 작으면 작을수록 그 분포는 뾰족한 모양을, 엔트로피가 크면 평평한 모양을 가진다

라인, 디자인이(가) 표시된 사진

자동 생성된 설명

왼쪽 그림은 엔트로피(정보량이) 많을때, 오른쪽은 엔트로피(정보량이)가 적은 그래프다.

여기서 크로스 엔트로피가 추가된다(Cross Entropy)

크로스 엔트로피란 분포함수 P에서 샘플링한 x를 통해 분포함수 Q의 평균 정보량을 나타낸 것이다. 교차라는 단어에서 알수있듯이, 다른분포 P를 사용해 대상 분포 Q의 엔트로피를 측정한다.

-log를 취했기 때문에, 분포 P와 Q가 비슷한 모양일수록 교차 엔트로피 H(P, Q)는 더 작은 값을 갖는다. 분류문제에서 cross entropy손실함수를 사용하여, 손실함수의 값이 최소가 되도록 경사하강법을 통해 신경망을 훈련한다. 알아내고 싶은 실제 확률 분포 P에서 샘플링한 데이터 X를 통해 신경망으로 구쉉된 확률 분포 에 넣어 교차 엔트로피가 최소가 되도록 한다.

다만 이것은 확률값을 바로 구할 수 있는 이산 확률 변수를 다루는 확률 분포에만 해당된다. 연속 확률 분포의 경우에는 MSE(Mean Square Error(평균 제곱 오차) 손실함수를 사용해야 한다. cross entropy를 통해서 구하는것이 틀린것은 아니지만, 샘플에 대한 확률값을 구할 수 없기 떄문에 이산확률분포에만 교차 엔트로피를 수행한다.

​

이산확률변수: Cross Entropy

연속확률변수: MSE(Mean Square Error)

연속확률변수, 이산확률변수: MLE(Maximum Likelihood Estimation)

**4. MSE (Mean Square Error)**

평균 제곱 오차(Mean Squared Error, MSE)는 주어진 데이터에 대한 예측값과 실제 값 사이의 평균 제곱 차이를 측정하는 지표입니다. 주로 회귀 문제에서 모델의 예측 성능을 평가하는 데 사용된다. n개의 데이터 샘플이 있고, 예측값을 y\_i^, 실게값은 y\_i라고 하면,

신경망은 확률분포를 정의하는 파라미터 θ를 갖는 확률분포함수이고, 알고자 하는 실제 확률 분포에 근사하기 위해 교차 엔트로피를 사용하지만, 이것은 이산확률변수에 국한된다.

MSE를 사용하는 것은 신경망이 표현(반환)하는 연속 확률 분포가 가우시안 분포를 따른다는 가정을 따른다. 가우시안 분포를정의하면 평균과 표준편차가 따른다.

µ(뮤): 평균

σ(시그마): 시그마

∀(for all): 모든 x에 대하여

​

그러므로 신경망이 분류 문제에서 softmax함수를 활용하여 이산확률분를 반환했던것처럼, 연속 확률 분포를 따를 경우 평균과 표준편차를 반환하면 가우시안 분포를 반환한다. 즉 신경망은 샘플 x\_i를 입력으로 받아 가우시안 분포의 를 반환한다.

 = 가우시안 분포,​

*θ*는*μ*를정의하는 신경망 파라미터​

Φ는함수 *σ*를정의하는 신경망 파라미터​

여기서 가우시안 분포 함수는 다음과 같다.

이것을 다시 적용하면 훈련하고자 하는 신경망 함수 f는 다음과 같이 표현이 가능하다

이 수식을 NLL(Negative Log Likelihood, 음의 가능도 함수)로 바꿀 수 있다.

이때 가우시안 분포의 σ가 상수라고 가정하면, 관심있는 값이 μ라고 할때 미분을 취하면 다음값을 얻는다.

즉 신경망은 앞의 손실 함수를 최소화하도록 경사하강법을 통해 훈련된다.

따라서 손실함수는 다음과 같다.

이것은 기존의 MSE와 비교하면 매우 유사하다

즉, 기존의 이산 확률분포는x로부터 확률값을 얻을 수 있기 때문에, cross entropy를 적용하여 실제 확률 분포에 신경망 확률 분포를 근사할 수 있다. 하지만, 연속 확률 분포는 확률값을 알 수 없기 때문에, 신경망의 출력값이 가우시안 분포 μ라는 가정을 하면 MSE를 통해 실제 확률 분포에 신경망으로 정의된 확률 분포를 근사할 수 있다.

1. Read & Split (Data set)

1 - read&split tabular dataset (e.g tsv, txt, csv file)

2 - shuffling before split into train/valid/test set

3 - split tarin, valid, test set

1. Preprocessing (Data set)

1 - remove unnecessary rows.

2 - Standard or Min/Max scaling based on training set(train set에대해서만)

1. Iterator (DataLoader)

1 - Suffle for epoch

2 - Get tensor chunk with mini-batch size

3 - yield the mini-batch for each Iterator

미니배치로 나누고 학습진행, 매 epoch마다 shuffling, 예전에는 shuffling split, prerocessing을 직접했다. 이것들을 파이토치가 해줄것이다! Trainer는 Pytorch의 Ignite가 도와줄 것이다. (Dataloader로부터 미니배치에 넣어주고 for문을 돌아준다) Trainer, Engine, Events, Callback, Metrics등이 제공된다.

Before – 주어진 데이터를 통해서 출력값을 가지는 함수를 모사하자!

주어진 입력(x)에 대해서 원하는 출력(y)을 반환하도록 손실함수를 최소화하는 파라미터(θ)를 찾자!

Gradient descent를 수행하기 위해 back-propagation을 수행한다.

After – 주어진 데이터를 통해서 출력값을 가지는 확률분포 함수를 모사하자!

확률 분포 P(x)와 p(y|x)로부터 데이터를 수집하여, 해당 데이터를 잘 설명하는 확률 분포 함수의 파라미터(θ) 찾자: logP(y|x; θ) – MLE, NLL, Gradient Descent using backpropagation

정보이론 관점에서 두 분포를 비슷하게 만들자! Minimize Cross Entropy, KL-Divergence

Geometric perspective

데이터는 저차원의 manifold에 분포하고 있으며, 여기에 노이즈가 추가된다.

Manifold: 고차원상의 데이터가 존재하고 있는 모양. 이것을 저차원 공간으로 매핑

데이터는 고차원 공간에 존재하고 있는데, 고차원 공간에 균등하게 있는 것이 아니라 저차원의 manifold에 분포하고 있다. Manifold를 배울 수 있다면, 더 낮은 차원으로 효율적인 매핑이 가능하다.

고차원 공간의 샘플들이 저차원 다양체(manifold)의 형태로 분포해 있다. 고차원 공간에서의 두점 사이의 거리는 저차원 공간으로의 맵핑 후 거리와 다르다. (증명은 되지 않아 가설이지만, 기정사실)