1. 딥러닝 기초

인공지능 스터디

2024/07/03 19:02 http://blog.naver.com/2439min/223500244056

1.1 딥러닝(Deep Learning)이란?

머신러닝(Machine Learning)의 하위 개념으로써 여러 층을 가진 인공신경망(Artificial Neural Network, ANN)을 사용하여 머신러닝 학습을 수행하는 것

기존의 머신러닝에서는 학습하려는 데이터의 여러 특징 중에서 어떤 특징을 추출할지를 사람이 직접 분석하고 판단해야만 한다.

하지만 딥러닝에서는 기계가 자동으로 학습하려는 데이터에서 특징을 추출하여 학습하게 된다.

딥러닝과 머신러닝의 가장 큰 차이점은 바로 기계의 자가 학습 여부로 볼 수 있다.

	기존 머신러닝	딥러닝
데이터 추출 판단	여러 데이터중에서 사람이 직접 판단하여 추출	기계가 자동으로 학습하려는 데이터를 판단하여 추출

딥러닝이란 기계가 자동으로 대규모 데이터에서 중요한 패턴 및 규칙을 학습하고, 이를 토대로 의사결정이나 예측 등을 수행하는 기술로 정의내릴 수 있다.

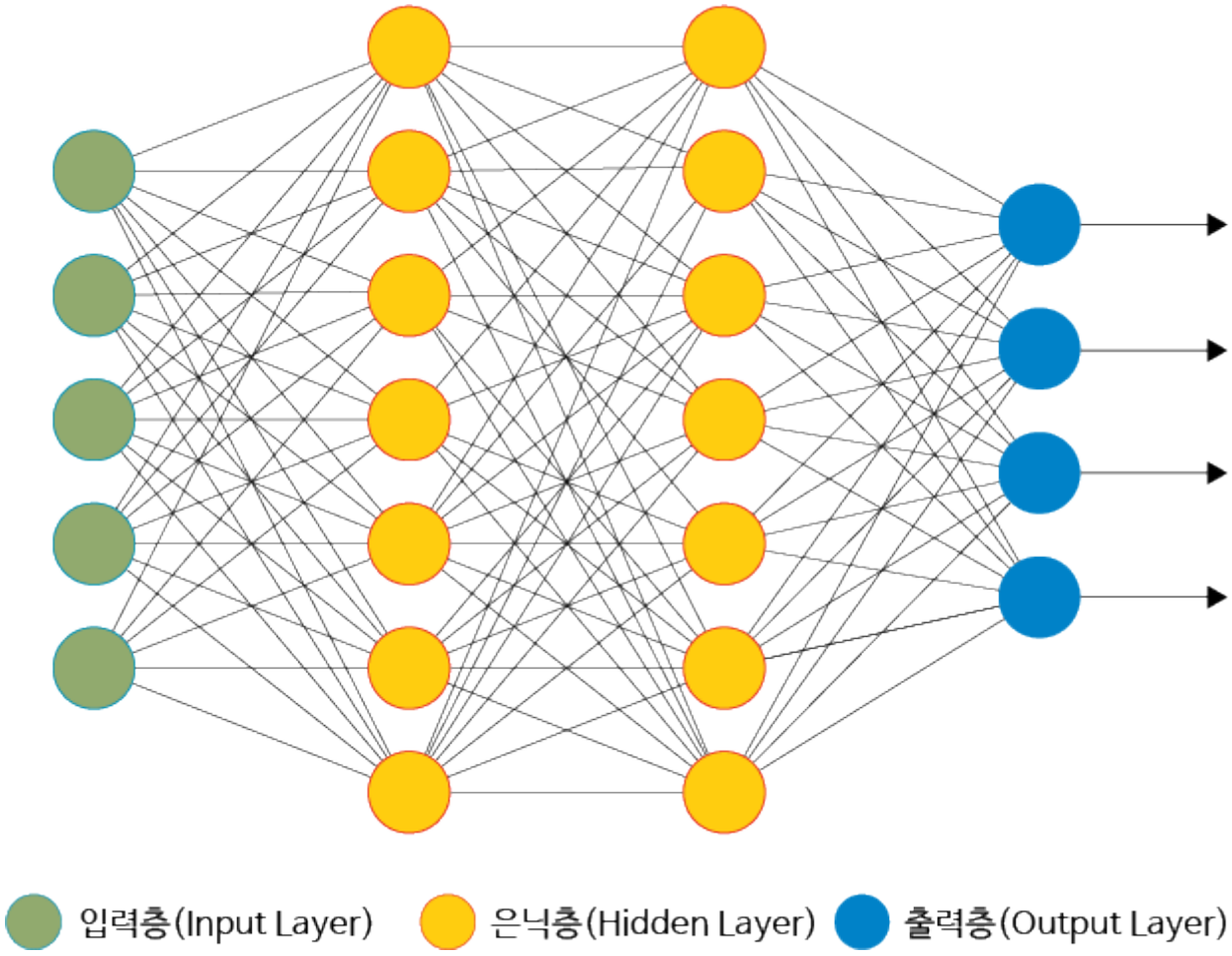
1.2 인공신경망(Artificial Neural Network, ANN)이란?

딥러닝에서 가장 기본이 되는 개념은 바로 신경망(Neural Network)이다.

인간의 뇌 안의 뉴런의 연결 구조를 본떠 만든 네트워크 구조를 인공신경망(Artificial Neural Network, ANN)이라고 부른다.

인공신경망은 여러 뉴런이 서로 연결되어 있는 구조의 네트워크이며, 입력층(input layer)를 통해 학습하고자 하는 데이터를 입력받게 됩니다.

이렇게 입력된 데이터들은 여러 단계의 은닉층(hidden layer)을 지나면서 처리가 이루어져 출력층(output layer)을 통해 최종 결과가 출력되게 된다.



이러한 신경망을 3개 이상 중첩한 구조를 깊은 신경망(Deep Neural Network, DNN)이라고 부르며, 이를 활용한 머신러닝 학습을 특별히 딥러닝이라고 부르는 것입니다.

1.3 MLP(Multi-Layer Perceptron)란?

1.3.1 퍼셉트론의 구조

입력층과 출력층을 가진다

입력층은 연산을 하지 않으므로, 퍼트론은 단일 층 구조라고 간주한다

입력층은 여러 개의 노드, 출력층은 오직 한 개의 노드로 구성된다.

입력층 노드와 출력층 노드를 연결하는 에지는 가중치를 가진다

1.3.2 다층 퍼셉트론 (Multi- Layer perceptron)

퍼셉트론은 '선형 분류기' 라는 한계점이 있다. (선형 분리: 하나의 선을 그어, 2가지로 분류 가능하게 하는 것)

즉, 퍼셉트론은 XOR문제를 해결할 수 없다는 한계점이 존재한다는 것이다.

이를 극복하기 위해 '다층' 구조를 이용한 방안을 제시하였고, 1986년에 '다층퍼셉트론' 이론을 정립하며 '신경망' 이 부활되었다.

다층 퍼셉트론의 핵심 아이디어는 다음과 같다.

은닉층을 둔다

'시그모이드 활성함수' 를 도입한다 (시그모이드 활성함수: 연성(soft) 의사결정. 영역을 영역으로 변환한 다. 출력이 연속값이며 출력값을 신뢰도로 간주하여 더 융통성있는 의사결정이 가능하다. 때문에 딥러닝에 효과적이다)
퍼셉트론과의 차이점: 퍼셉트론은 '계단함수'를 활성함수로 사용한다 (계단함수: 경성 (hard) 의사결정. 영역을 점으로 변환한다)
오류 역전파 알고리즘을 사용한다 (오류역전파 알고리즘: 역방향으로 진행하면서 한 번에 한 층씩 그레이 디언트를 계산하고 가중치를 갱신하는 방식)

1.3.3 다층 퍼셉트론의 구조

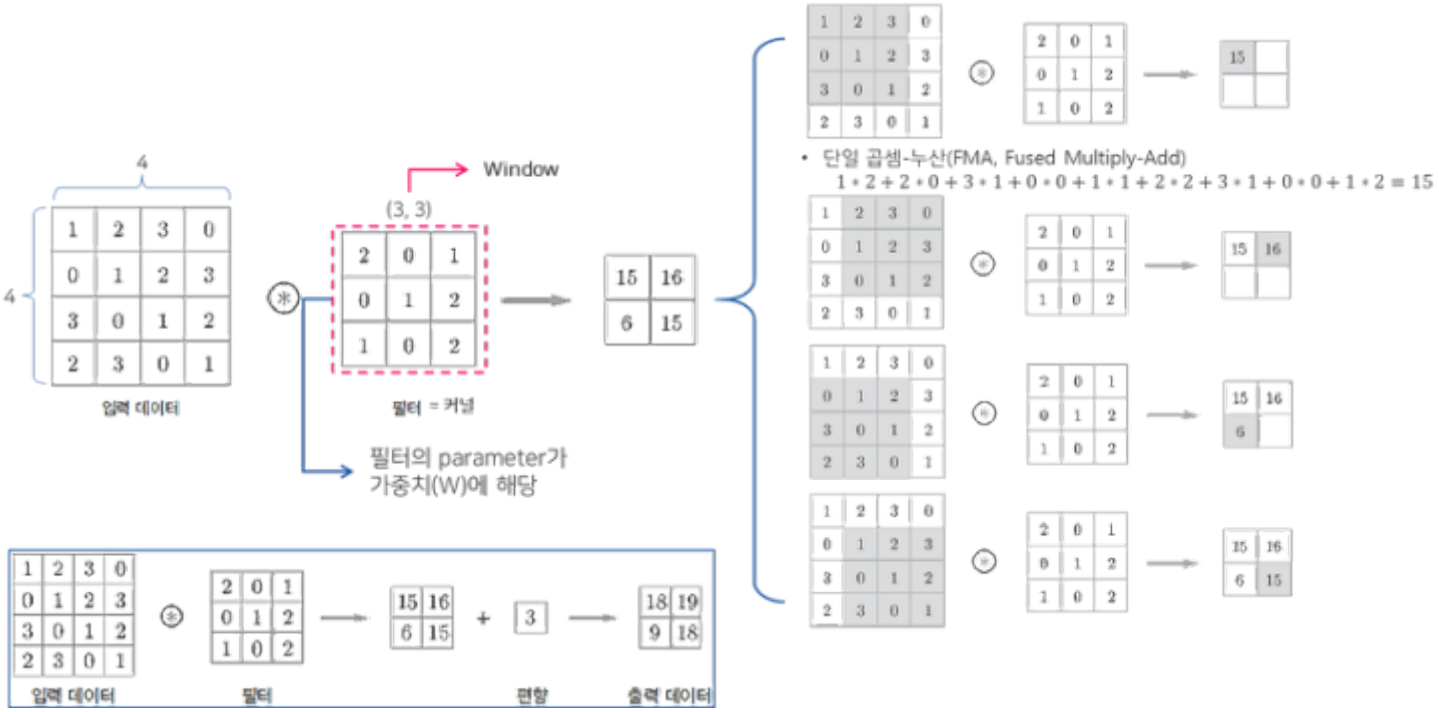
다층 퍼셉트론은 은닉층의 개수에 따라 구조가 정해진다.

ex) 2층 퍼셉트론: 은닉층 1개 /'입력층 - 은닉층 - 출력층' 의 2층 구조
3층 퍼셉트론: 은닉층 2개 /'입력층 - 은닉층- 은닉층 - 출력층' 의 3층 구조

여기서 '은닉층' 이란, '특징 추출기' 로서 특징 벡터를 분류에 더 유리한 새로운 특징 공간으로 변환하는 일을 한다.
즉, 출력을 내보내기 전까지 특징 공간을 분할하는 일을 하는 것이다. (현대 기계학습에서는 이를 '특징 학습' 이라고 한다)
딥러닝은 이러한 은닉층이 3개 이상으로, 더 많은 단계를 거쳐 특징 학습을 한다.

1.4 합성곱 연산

데이터와 필터(또는 커널)의 모양을 (높이, 너비)로 나타내고, 윈도우(Window)라고 부른다. 여기서 입력 데이터는 (4, 4), 필터는 (3, 3)이고, 필터가 바로 Conv Layer의 가중치에 해당한다.
합성곱 연산은 필터의 윈도우를 일정한 간격으로 이동해가며 계산한다. 아래의 그림처럼, 합성곱 연산은 입력데이터와 필터간에 서로 대응하는 원소끼리 곱한 후 총합을 구하게 되며, 이것을 Fused Multiply-Add(FMA)라고한다. 마지막으로 편향(bias)은 필터를 적용한 후에 더해주게 된다.



1.5 비용함수(Cost Function), 손실함수란(Loss Function)?

1.5.1 손실함수(Loss Function)

손실함수(Loss Function)는 입력으로 받은 데이터를 하나하나 받아 실제값과 예측값 간의 오차를 계산하는 방식이다. 입력으로 받은 데이터를 하나씩 모두 오차를 계산하는 방식인 경우를 손실 함수라 부른다.

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty. For example:
- square loss $l(f(x_i|\theta), y_i) = (f(x_i|\theta) - y_i)^2$, used in linear regression
- hinge loss $l(f(x_i|\theta), y_i) = \max(0, 1 - f(x_i|\theta)y_i)$, used in SVM
- 0/1 loss $l(f(x_i|\theta), y_i) = 1 \iff f(x_i|\theta) \neq y_i$, used in theoretical analysis and definition of accuracy

1.5.2 비용함수(Cost Function)

비용함수(Cost Function)는 입력으로 받은 데이터를 모아서 오차를 계산하는 함수를 일컫는다. 입력으로 들어온 데이터를 기반으로 모든 데이터의 비용을 계산하는 방식이다. 풀 배치 형태로 한 번에 입력을 받아 계산하는 방식이다.

- **Cost function** is usually more general. It might be a sum of loss functions over your training set plus some model complexity penalty (regularization). For example:
- Mean Squared Error $MSE(\theta) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i|\theta) - y_i)^2$
- SVM cost function $SVM(\theta) = \|\theta\|^2 + C\sum_{i=1}^{N}\xi_i$ (there are additional constraints connecting ξ_i with C and with training set)

1.6 과적합이란?

1.6.1 과적합

머신 러닝에서 과적합은 알고리즘이 학습 데이터에 과하게 적합한 상태이거나 정확하게 일치할 때 발생하며 그 결과 모델이 학습 데이터가 아닌 다른 데이터에서 정확한 예측을 생성하거나 결론을 도출할 수 없게 된다.
머신 러닝 알고리즘이 구성되면 샘플 데이터 세트를 활용하여 모델을 학습시킵니다. 그러나 모델이 샘플 데이터를 너무 오래 학습하거나 모델이 너무 복잡하면 데이터 세트 내에서 '노이즈' 또는 관련 없는 정보를 학습하기 시작할 수 있다. 모델이 노이즈를 기억하고 학습 세트에 과하게 적합한 상태이면 '과적합'이 됩니다. 즉, 새로운 데이터를 효과적으로 일반화할 수 없게 된다. 모델이 새로운 데이터를 효과적으로 일반화할 수 없다면 의도한 분류 또는 예측 작업을 수행할 수 없다.

1.6.2 과적합 방지 방법

- 조기 종료: 모델이 모델 내의 노이즈를 학습하기 전에 학습을 종료한다. 이 접근 방식은 학습 프로세스를 너무 빨리 종료하여 과소적합이라는 반대 문제를 일으킬 위험이 있다. 과소적합과 과적합 사이의 '최적점'을 찾는 것이 궁극적인 목표이다.
- 더 많은 데이터로 학습: 더 많은 데이터를 포함하도록 학습 세트를 확장하면 입력 변수와 출력 변수 간의 우세한 관계를 구문 분석할 수 있는 더 많은 기회를 제공하여 모델의 정확도를 높일 수 있다. 즉, 이는 정제되고 관련성 높은 데이터가 모델에 주입될 때 더 효과적인 방법이다. 그렇지 않은 경우 모델에 복잡성이 계속 추가되어 과적합이 발생할 수 있다.
- 데이터 증대: 정제되고 관련성이 높은 데이터를 학습 데이터에 주입하는 것이 바람직하지만 때로는 모델 안정성을 향상하기 위해 노이즈가 있는 데이터가 추가된다. 하지만 이 방법은 신중하게 사용해야 한다.
- 특징 선택: 모델을 구축할 때 특정 결과를 예측하는 데 사용되는 여러 매개 변수 또는 특징이 있지만 이러한 특징은 다른 특징과 중복되는 경우가 많다. 특징 선택은 학습 데이터 내에서 가장 중요한 특징을 식별한 다음 관련성이 없거나 중복되는 특징을 제거하는 과정이다. 일반적으로 차원 축소로 오인되지만 이는 다르다. 그러나 두 방법 모두 모델을 단순화하여 데이터의 우세 추이를 설정하는 데 도움이 된다.
- 정규화: 모델이 너무 복잡할 때 과적합이 발생하면 특징 수를 줄이는 것이 합리적이다. 모델에서 어떤 특징을 제거해야 할지 모르는 경우 정규화 방법이 특히 유용할 수 있습니다. 정규화는 계수가 더 큰 입력 매개변수에 '페널티'를 적용하여 모델의 분산량을 제한한다. 라쏘 정규화, 릿지 회귀, 드롭아웃 등 다양한 정규화 방법이 있지만 모

두 데이터 내의 노이즈를 식별하고 줄이는 것을 목표로 한다.

- 앙상블 방법: 앙상블 학습법은 분류기 세트(예: 의사 결정 트리)로 구성되며 가장 인기 있는 결과를 식별하기 위해 예측이 집계된다. 가장 잘 알려진 앙상블 방법으로 배깅(Bagging)과 부스팅(Boosting)이 있다. 배깅에서는 학습 세트에 있는 데이터의 무작위 샘플이 대체를 통해 선택된다. 즉, 개별 데이터 포인트를 두 번 이상 선택할 수 있다. 여러 데이터 샘플이 생성된 후 이러한 모델은 독립적으로 학습되며 작업 유형(예: 회귀 또는 분류)에 따라 이러한 예측의 평균 또는 대부분을 통해 더 정확한 추정치를 산출한다. 이는 일반적으로 노이즈가 있는 데이터 세트 내의 분산을 줄이는 데 사용된다.

3. 데이터 전처리

인공지능 스터디

2024/07/03 20:40 http://blog.naver.com/2439min/223500328155

3.1 데이터 전처리(Data preprocessing)란?

데이터 전처리란, 데이터 분석을 위해 수집한 데이터를 분석에 적합한 형태로 가공하는 과정이다. 데이터 전처리를 통 해 불필요한 데이터를 제거하고, 결측치나 이상치를 처리하여 데이터의 질을 향상시킬 수 있다. 이렇게 가공된 데이 터는 분석 모델을 구축하고 결과를 도출하는 데에 더욱 유용하게 활용될 수 있다.

3.2 데이터 전처리의 중요성

- 잡음 (noise): 측정 과정에서의 오류 값.
- 이상치(outlier): 일반적인 데이터에 비해 다른 특성을 가지는 튀는 값.
- 부적합(inconsistent): 모순된 잘못된 데이터 값.
- 결측치(missing value): 누락된 데이터 값
- 중복(duplicated): 중복되는 데이터 값.

이러한 값들이 많으면 데이터를 적절히 분석하는 데 어려움이 있기에, 필요 없는 값들은 삭제하고 부족한 값들은 적절한 값으로 채워 넣는 등 다양한 절차가 필요하다. 데이터의 질이 좋지 않으면 분석 결과도 부정확하게 나올 수 있다. 따라서 정확한 분석 결과를 얻기 위해 데이터 전처리는 반드시 필요한 작업이다.

3.3 데이터 전처리 절차 및 방법

3.3.1 데이터 전처리 절차

1. 데이터 수집
2. 데이터 정제 (결측치, 이상치 처리)
3. 데이터 변환 (날짜, 문자열 등)
4. 데이터 필터링 (조건에 따른 데이터 추출)
5. 데이터 정렬 (sort_values 함수 사용법)
6. 데이터 그룹화 (groupby 함수 사용법)
7. 데이터 변환 함수 (apply, map, applymap 등)
8. 데이터 피벗 (pivot_table 함수 사용법)
9. 데이터 병합 (merge 함수 사용법)

10. 데이터 분할 (split 함수 사용법)
11. 데이터 샘플링 (sample 함수 사용법)
12. 데이터 집계 (agg함수 사용법)
13. 데이터 시각화 (matplotlib, seaborn 등)

3.3.2데이터 전처리 방법

- 결측치 처리: 데이터에서 빠진 값이 있을 경우, 해당 값을 대체하거나 삭제하여 데이터의 일관성을 유지한다.
- 이상치 처리: 데이터에서 이상한 값이 있을 경우, 해당 값을 대체하거나 삭제하여 분석 결과에 영향을 미치는 오류 를 방지한다.
- 데이터 정규화: 서로 다른 스케일의 데이터를 비교 분석하기 위해, 데이터 값을 일정한 범위로 조정한다.
- 데이터 인코딩: 텍스트 데이터를 컴퓨터가 이해할 수 있는 형태로 변환한다. (예: 원-핫 인코딩)
- 데이터 통합: 여러 개의 데이터를 하나의 데이터로 통합하여 분석에 용이하게 한다.
- 데이터 분할: 분석에 필요한 부분 데이터를 추출하여, 불필요한 데이터를 제거한다.
- 데이터 정렬: 분석에 필요한 순서대로 데이터를 정렬한다.
- 데이터 그룹화: 데이터를 그룹별로 분류하고, 각 그룹에 대한 통계 정보를 추출한다. (예: 그룹별 평균, 합계, 표준편차 등)
- 데이터 변환 함수: apply, map, applymap 등의 함수를 사용하여 데이터 값을 변환한다.
- 데이터 피벗: pivot_table 함수를 사용하여, 행과 열을 바꾸거나, 그룹별 집계 정보를 표현한다.
- 데이터 병합: merge 함수를 사용하여, 여러 개의 데이터를 하나로 병합한다.
- 데이터 분할: split 함수를 사용하여, 데이터를 분할하고, 분할된 데이터를 분석한다.
- 데이터 샘플링: sample 함수를 사용하여, 샘플 데이터를 추출하고, 추출된 데이터를 분석한다.
- 데이터 집계: agg 함수를 사용하여, 그룹별 집계 정보를 추출한다.
- 데이터 시각화: matplotlib, seaborn 등의 시각화 라이브러리를 사용하여, 데이터를 시각화하고, 분석 결과를 쉽게 이해할 수 있도록 도와준다.

3.4 데이터 전처리 기초 문법

https://wikidocs.net/book/4764

파이썬 데이터전처리 실습

책 소개 데이터분석에서 데이터 전처리하는 과정이 80% 이상을 차지합니다. 이 책은 데이터 분석에서 자주 사용하는

자료구조에 대한 설명과 실습 예제를 정리하여 제공합…

wikidocs.net

2. Pytouch 다루기

인공지능 스터디

2024/07/03 19:26 http://blog.naver.com/2439min/223500265599

2.1 환경설정

2.1.1 Pytouch란

Pytouch는 오픈소스 머신 러닝 라이브러리로 자연어 처리를 비롯해 이미지 프로세싱과 같은 애플리케이션을 위해 사용된다.

Python에 친화적인 라이브러리로 간결하고 구현이 빨리되며, 텐서플로우보다 사용자가 익히기 훨씬 쉽다. 학습 및 추론 속도가 빠르고 Define by Run 방식을 기반으로 하여 실시간 결괏값을 시각화할 수 있다.

자동 미분 모듈, 최적화 모듈, 이미지 처리 모듈, 오디오 처리 모듈 등 머신 러닝 및 딥 러닝을 위한 다양한 모듈을 제공한다. 또한, PyTorch는 클라우드 플랫폼(Amazon Web Services, Google Cloud Platform 등)에서도 손쉽게 적용할 수 있다.

2.1.2 Anaconda 설치

아나콘다(Anaconda)를 패키지 관리자로 사용하여 운영체제 내에 가상 Python 환경을 설정한다.

아나콘다는 전 세계적으로 2,500만 명이 넘는 사용자를 보유한 가장 인기 있는 Python 배포 플랫폼입이다.
클라우드 기반 저장소를 검색하여 7,500개 이상의 데이터 과학 및 머신 러닝, 딥 러닝 패키지를 쉽게 설치할 수 있다.
아나콘다를 사용하면 서로 간섭 없이 개별적으로 유지 관리하고 실행할 수 있는 여러 실행 환경을 쉽게 관리할 수 있다.

2.1.3 PyTorch CPU 설치

아나콘다(Anaconda)를 통해 PyTorch를 설치할 수 있다.

pytorch는 다양한 모듈을 제공하며, 대표적으로 다음과 같은 라이브러리가 있습니다.

1. pytorch : 자동 미분 시스템에 구축된 심층 신경망 라이브러리

2. torchvision : 컴퓨터 비전을 위한 이미지 변환 라이브러리
3. torchaudio : 오디오 및 신호 처리를 위한 라이브러리

PyTorch CPU(cpuonly)는 다음과 같은 명령어를 통해 설치가 가능하다.
conda install pytorch torchvision torchaudio cpuonly -c pytorch

2.1.4 Pytorch GPU 설치

PyTorch GPU는 아나콘다(Anaconda) 이외에도 CUDA를 설치해야 활용이 가능하다.
또한, CUDA 설치 조건에는 특정 조건을 만족하는 GPU만 사용이 가능하다.
현재 PyTorch를 설치하려는 GPU의 사양을 확인한다.
CUDA Compute Capability가 3.5 이상의 GPU를 사용하고 있는지 확인한다.
만약, 3.0 이상의 GPU라면, CUDA 10.2까지 적용이 가능하다.
그 미만의 CUDA Compute Capability라면 PyTorch GPU를 적용할 수 없다.
또한, 현재 사용하고 있는 GPU의 드라이버를 최신 버전으로 업데이트한다.

- CUDA Toolkit

CUDA Toolkit은 GPU 가속화 애플리케이션 개발에 필요한 라이브러리를 제공한다.
GPU 가속화 라이브러리, 디버깅 및 최적화 툴, 컴파일러 등을 제공한다.
이 라이브러리를 활용하여 딥 러닝 알고리즘들을 사용하기 쉽게 도와준다.
CUDA Toolkit을 설치하기 전에, PyTorch GPU에서 지원하는 CUDA Toolkit 버전을 확인해야 한다.

현재, PyTorch GPU에서 지원하는 CUDA Toolkit은 CUDA 10.2, CUDA 11.3이다.
GPU의 CUDA Compute Capability에 맞는 CUDA Toolkit으로 설치한다.

- NVIDIA cuDNN

NVIDIA CUDA 심층 신경망 라이브러리(cuDNN)는 심층 신경망을 위한 GPU 가속 프리미티브(GPU-accelerated library of primitives) 라이브러리이다.
설치한 CUDA 버전과 호환되는 압축 파일을 다운로드하여, NVIDIA GPU Computing Toolkit이 설치된 경로로 파일을 덮어 씌운다.
NVIDIA GPU Computing Toolkit/CUDA/{Version}의 경로이다.
파일을 모두 덮어 씌웠다면 환경 변수에 경로(Path)를 등록한다.

- Tip : CUDA 11.3의 경우 Download cuDNN v8.2.1 (June 7th, 2021), for CUDA 11.x로 설치한다.

C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.3₩bin
C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.3₩extras₩CUPTI₩libx64
C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.3₩include
만약, C:₩Program Files 경로에 11.3 버전으로 설치했다면 위와 같이 경로를 추가한다.
총 세 개의 경로를 환경 변수에 추가한다.

setx path "%PATH%;C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.3₩bin"
setx path "%PATH%;C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.3₩extras₩CUPTI₩libx64"
setx path "%PATH%;C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.3₩include"
커맨드 창(cmd)에서 경로를 등록하는 경우 위와 같이 입력할 수 있다.
커맨드 창에 path를 입력하여 경로가 정상적으로 등록됐는지 확인한다.
경로가 정상적으로 등록됐다면, path 명령어에서 출력되는 결괏값에서 추가된 경로를 확인할 수 있다.

Conda Install
conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch
cudatoolkit={Version}의 형태로 PyTorch GPU를 설치한다.
Version은 현재 컴퓨터에 설치한 CUDA Toolkit 버전을 추가한다.

2.1.5결과 확인
```
import torch
print(torch.__version__)
print(torch.cuda.is_available())
```

결과
```
1.11.0
True
```

현재 PyTorch의 버전과 GPU 사용 가능 여부가 출력된다.
만약, GPU 버전으로 설치하였는데 torch.cuda.is_available()의 값이 거짓(False) 값으로 나온다면 커맨드 창에서 nvcc --version을 입력한다.
마지막 줄의 Cuda compilation tools, release {Version}을 확인하여 CUDA가 정상적으로 설치되었는지 확인한다.
또한, cuDNN의 패치 경로와 환경 변수를 확인한다.

2.2 구성요소

1. torch

- 메인 네임스페이스로 텐서 등의 다양한 수학 함수가 포함되어 있다.
- NumPy와 유사한 구조를 가진다.

2. torch.autograd

- 자동 미분 기능을 제공하는 라이브러리이다.

3. torch.nn

- 신경망 구축을 위한 데이터 구조나 레이어를 정의한다.
- 예를 들어 RNN, LSTM과 같은 레이어, ReLU와 같은 활성화 함수, MSELoss와 같은 손실 함수들이 있다.

4. torch.multiprocessing

- 병렬처리 기능을 제공하는 라이브러리이다.

5. torch.optim

- SGD(Stochastic Gradient Descent)를 중심으로 한 파라미터 최적화 알고리즘을 제공한다.

6. torch.utils

- 데이터 조작 등 유틸리티 기능 제공한다.

7. torch.onnx

- ONNX(Open Neural Network Exchange)의 포맷으로 모델을 익스포트할 때 사용한다.
- ONNX는 서로 다른 딥 러닝 프레임워크 간에 모델을 공유할 때 사용하는 포맷이다.

2.3 문법

https://tutorials.pytorch.kr/beginner/basics/intro.html

파이토치(PyTorch) 기본 익히기

파이토치(PyTorch) 기본 익히기|| 빠른 시작|| 텐서(Tensor)|| Dataset과 Dataloader|| 변형(Transform)|| 신경망 모델 구성하기|| Autograd|| 최적화(Optimization)|| 모델 저장하고 불러오기 Authors: Suraj Subramanian, Seth Juarez, Cassie Breviu, Dmitry Soshnikov, Ari Bornstein 번역: 박정환 대부분의 머신러닝 워크플로우는 데이터 작업과 모델 생성, 모델 매개변수 최적화, 학습된 모델 저장이 포함됩니다...

tutorials.pytorch.kr

2.4 특징

- 설치가 간편하다.

- 이해와 디버깅이 쉬운 직관적이고 간결한 코드로 구성된다.
- Define by Run 방식을 기반으로 한 실시간 결과값을 시각화
- 파이썬 라이브러리(Numpy, Scipy, Cython)와 높은 호환성
- Winograd Convolution Algorithm 기본 적용을 통한 빠른 모델 훈련이 가능하다.
- 모델 그래프를 만들 때 고정상태가 아니기 때문에 언제든지 데이터에 따라 조절이 가능하다.
- Numpy스러운 Tensor연산이 GPU로도 가능하다.
- 학습 및 추론 속도가 빠르고 다루기 쉽다.