

오늘 실습 코드

1. Hands on machine learning 책 16장 autoencoder 일부

2. 깃허브: <https://github.com/cm8908/>


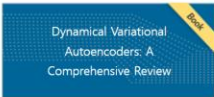
에서 Repositories 클릭 후 Anomaly-Detection...

Autoencoder와 관련하여 IEEE Xplore 검색 결과

Showing 1-25 of 18,209 results for **autoencoder** ×

☐ Conferences (12,759) ☐ Journals (5,077) ☐ Early Access Articles (247) ☐ Magazines (104) ☐ Books (22)

Publications You May Be Interested In: Hide Related



[View More Journals](#)

Show

☒ All Results
☐ Subscribed Content ?
☐ Open Access Only

Year ^

☒ Range ☐ Single Year

1989 2024

[Clear](#) [Apply](#)

Author ▼

Affiliation ▼

Publication Title ▼

☐ **Select All on Page**

Sort By Relevance ▼

☐ **SIFT Features for Deep and Variational Autoencoders: A Performance Comparison** 🔒

Fabian Barreto; Sushilkumar Yadav; Suprava Patnaik; Jignesh Sarvaiya
2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)
Year: 2020 | Conference Paper | Publisher: IEEE
Cited by: Papers (2)

[Abstract](#) [HTML](#) [PDF](#) [CC](#)

☐ **Laughter synthesis: A comparison between Variational autoencoder and Autoencoder** 🔒

Nadia Mansouri; Zied Lachiri
2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)
Year: 2020 | Conference Paper | Publisher: IEEE
Cited by: Papers (1)

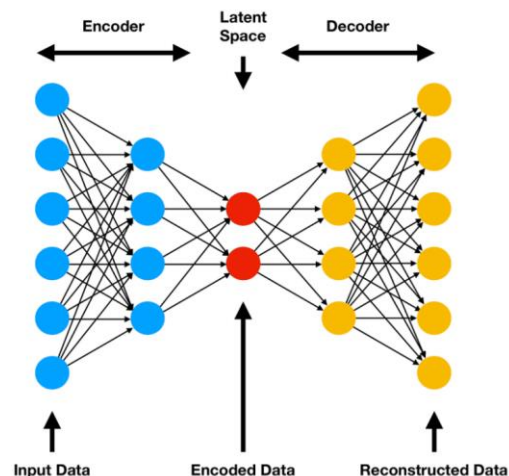
[Abstract](#) [HTML](#) [PDF](#) [CC](#)

1. Encoder란? 일반적으로 입력데이터를 특정 형태로 변환하는 것. 유용한 특징을 추출

- 주로 번역기와 같은 encoder-decoder 구조에서 사용
- Autoencoder에서도 encoder-decoder 구조를 사용

2. Autoencoder란? 자동적인 encoder. 즉, 자동으로 입력데이터를 특정 형태로 변환함

- 정의 1: 레이블이 없는 입력 데이터에 대하여 dense (compressed)한 표현을 학습할 수 있는 인공신경망.
- 정의 2: 입력 데이터에 대하여 supervision (지도)없이 효율적인 표현을 학습할 수 있는 인공 신경망



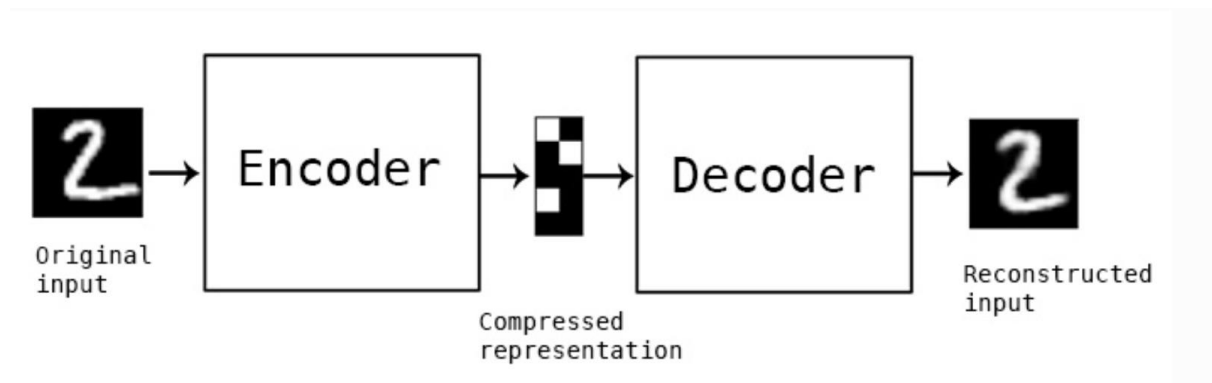
간단한 autoencoder 예

3. Latent representation (coding)? 입력 데이터의 압축된 표현으로 중간에 생성된 표현을 잠재 표현이라 함. 입력 데이터보다 대개의 경우 낮은 차원

4. Autoencoder의 구성

- 1) Encoder (recognition network): 입력데이터를 의미 있고 주로 저차원의 압축된 표현인 latent representation (잠재 표현)으로 encoding
- 2) Decoder (generative network): 잠재 표현을 다시 입력 데이터로 복원 (reconstruction). 원래의 입력 데이터와 유사하게

3) MLP와 동일한 구조. 단, 입력 뉴런의 개수 = 출력 뉴런의 개수



4) 주의:

- 우리의 목적은 의미있는 잠재 표현을 찾아내는 것으로 단순히 입력을 출력에 copy만 하는 인공신경망을 원하지는 않음.
- 입력 데이터에 비해서 잠재 표현의 차원이 더 낮기 때문에 이를 **undercomplete**라고 표현하며 undercomplete autoencoder의 경우에는 단순히 입력을 copy해서 출력으로 내보낼 수 없음. **It is forced to learn the most important features in the input data and also drop the unimportant ones**

5) 뒤에서 보겠지만 입력 데이터의 잠재 표현을 더 잘 학습할 수 있게 추가적인 방법들도 있음

- 입력에 noise를 줄 수도 있고 잠재 표현의 크기를 줄이는 방법도 있다

5. Autoencoder의 학습 목표

- 입력 데이터만을 활용하는 비지도 학습으로 입력데이터와 복원된 데이터의 차이 (**reconstruction loss**)를 최소화하는 방향으로 학습
- 수식:

$$\text{MSE Loss} = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2$$

여기서:

- N 은 데이터 포인트의 총 수,
- x_i 는 입력 데이터의 i -번째 요소,
- \hat{x}_i 는 해당 입력 데이터에 대한 Autoencoder의 출력의 i -번째 요소를 나타냅니다.

6. (실습) 간단한 autoencoder build

- 파이토치(PyTorch)를 사용하여 간단한 오토인코더(Autoencoder)를 정의하는 것

- 784 -> 32 -> 784

1) 인코더: 입력 크기 784 (28x28), 활성화 함수: ReLU

2) 디코더: 인코더에서 압축된 특징을 다시 784차원으로 복원

3) forward 메서드: 주어진 입력 'x'를 인코더를 통과시켜 압축하고, 디코더를 통과시켜 다시 복원한 값을 반환

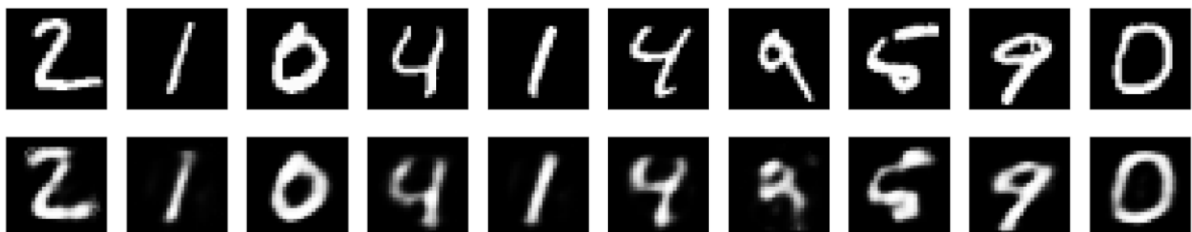
4) 하이퍼파라미터 설정

- encoding_dim: 압축된 특징의 차원으로 32차원

- adam optimizer

- BCELoss: binary cross entropy loss

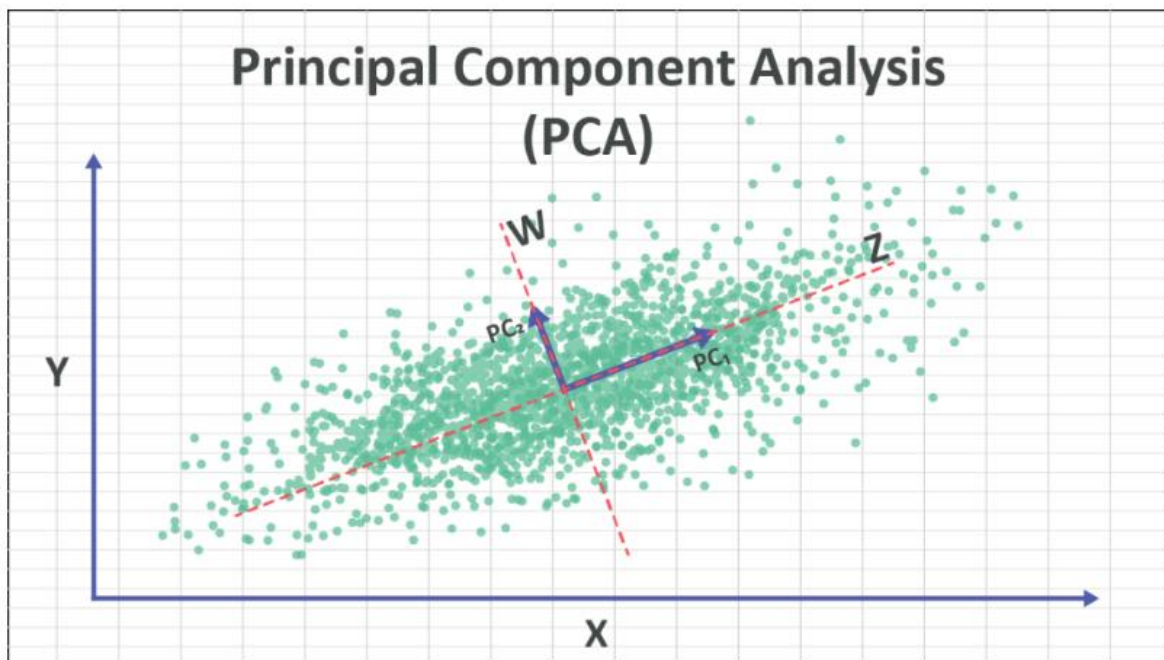
```
Epoch:1, Loss:0.2324
Epoch:2, Loss:0.1776
Epoch:3, Loss:0.1548
Epoch:4, Loss:0.1441
Epoch:5, Loss:0.1418
Epoch:6, Loss:0.1244
Epoch:7, Loss:0.1194
Epoch:8, Loss:0.1185
Epoch:9, Loss:0.1189
Epoch:10, Loss:0.1119
```



8. Autoencoder와 차원 감소의 유사성

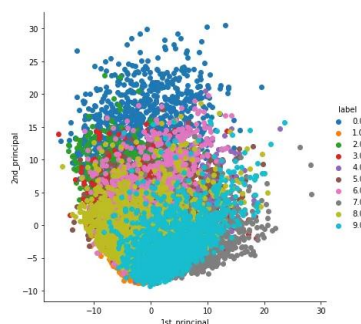
- Autoencoder는 입력 데이터를 그 보다 저차원의 latent (잠재) 표현으로 바꾼다
- 이는 PCA (principal component analysis)와 유사하지 않나?

Recall: PCA: '정보 손실'을 최소화하면 D 차원을 d 차원으로 변환 ($d < D$). 즉, 차원 축소.
목표: 차원 축소시에 변환된 데이터들의 분산을 최대한 하는 축을 찾음.



아래 MNIST dataset을 2차원으로 차원 축소 후에 시각화. Class간 overlapping이 많이 되어서 구분이 어려움

실제 고차원 데이터의 시각화 목적에서는 T-SNE가 많이 사용됨



t-SNE (t-Distributed Stochastic Neighbor Embedding)

- 각 데이터 포인트 간의 유사성을 보존하면서 저차원에서의 매핑을 수행
- 군집 간의 거리를 잘 유지

9. (실습) Autoencoder와 PCA의 유사성

- Linear activation만 사용, cost function은 MSE (mean square error)
- 이와 같은 경우 PCA를 수행하는 것과 같은 결과를 줌
- 3D dataset을 2D로 투영 (projection)시키는 예

구글에서 "hands on machine learning github" 로 검색


아래 Quick Start에서 "open in Colab" 클릭

Quick Start

Want to play with these notebooks online without having to install anything?

Use any of the following services (I recommended Colab or Kaggle, since they offer free GPUs and TPU:

WARNING: Please be aware that these services provide temporary environments: anything you do will be while, so make sure you download any data you care about.

-  [Open in Colab](#)

아래 코드 추가로 3D random data 시각화

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_train[:, 0], X_train[:, 1], X_train[:, 2], c='r', marker='o')

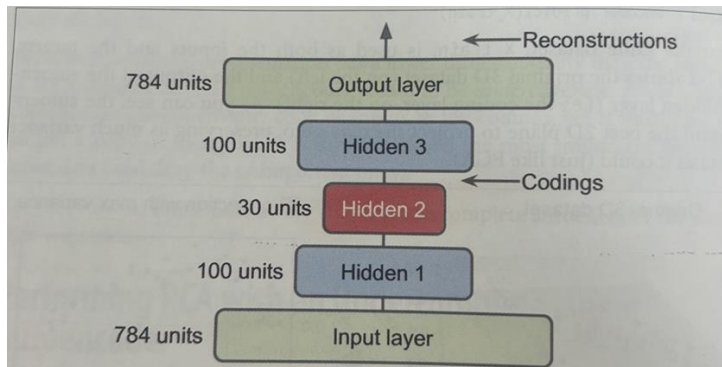
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')

plt.title('3D Scatter Plot')
plt.show()
```

10. Stacked (deep) autoencoder

- 여러 개의 hidden layer를 갖는 오토인코더
- 중간 hidden layer를 기준으로 대부분 대칭구조

< MNIST dataset에 대한 stacked autoencoder 예 >



11. (실습) Stack autoencoder for MNIST dataset

- keras 사용.
- - original image와 autoencoder 학습 후에 reconstruction된 image들 보여줌
- SELU (scaled exponential linear unit) 활성화 함수 사용
- 손실 함수로는 이진 교차 엔트로피(binary crossentropy)를 사용하고, 최적화기로는 확률적 경사 하강법(SGD)을 사용
- 20 epoch 학습



12. Fashion MNIST dataset 소개

- Fashion MNIST는 10개의 다른 의류 class에 속하는 흑백 28x28 크기의 패션 관련 이미지로 구성
- 각 image는 10개의 class 중 하나에 속함. 각 class는 다양한 의류 아이템을 나타냄
- MNIST 와 유사하지만 좀 더 복잡한 image를 다루기 위해 사용, 분류기 benchmark

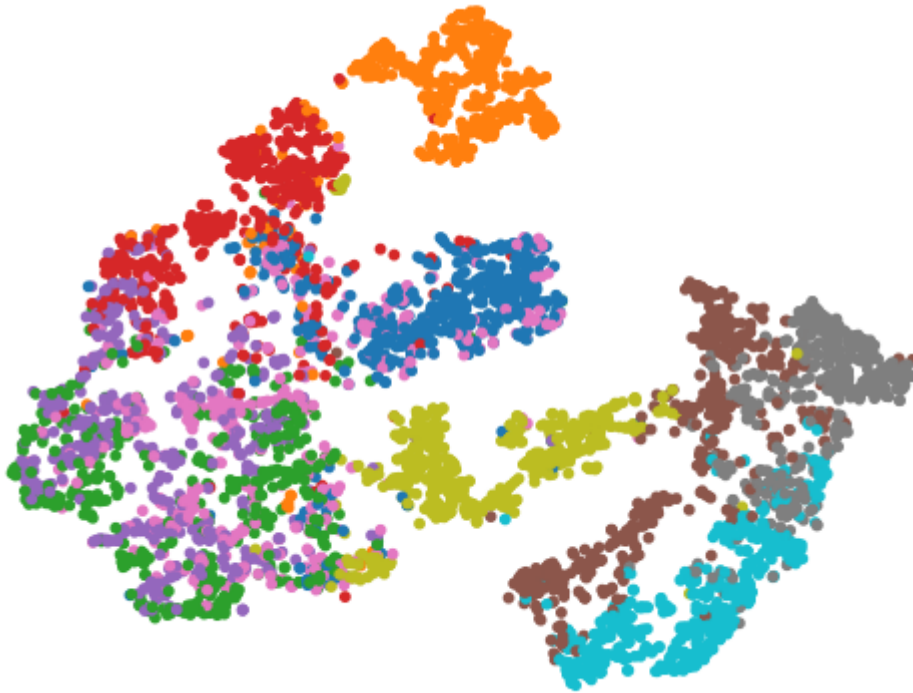
1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

각 이미지는 흑백(1채널)이며, 해상도는 28x28 픽셀'



13. (실습) Visualizing Fashion MNIST

- T-SNE 사용



13. Autoencoder에서의 tying (sharing) weights

- 위와 같이 대칭적인 autoencoder에서 사용되는テクニック
- 인코더와 디코더 간의 가중치를 공유. 즉, 인코더와 디코더의 가중치 행렬이 같아짐
- Why? 학습 가능한 모델의 파라미터 수를 감소시킴. 더 적은 데이터로도 학습되거나, 계산 및 메모리 자원 절약
- 자연어 처리등에서도 사용

14. (실습) Tying weights

It is common to tie the weights of the encoder and the decoder, by simply using the transpose of the encoder's weights as the decoder weights. For this, we need to use a

custom layer.



16. Pre-training (사전 훈련)

- 주로 large dataset에서 일반적인 특징을 학습한 후에 더 작은 데이터셋이나 특정 작업에 대해서 fine-tuning하는 방법을 포함
- 초기 가중치를 설정하고 모델의 초기 특성을 학습함으로써, 학습의 안정성을 높이고 성능을 개선
- 장점: 1) 데이터 부족 문제 해결 2) 초기 가중치 설정을 통한 학습 초기화 3) 특정 작업에 맞게 fine-tuning을 통한 학습 시간과 데이터 양 절약
- 한가지 방법은 large 비지도 학습 데이터셋에 대해서 모델을 미리 훈련
- 최근 여러가지 computer vision과 같은 task에서 pre-training의 중요성이 많이 연구됨
- 여기서는 오토인코더를 사용한 pre-training 보여줌

17. Unsupervised pretraining using stacked autoencoder (스택 오토인코더를 사용한 비지도 사전 학습)

- Label이 없는 large 데이터셋이 있으면 모든 데이터로 autoencoder 학습
- 실제 classification과 같은 문제에 **autoencoder의 lower (하위) layer를 재사용**
- Lower layer: input에 가까운 hidden layer를 부르는 말

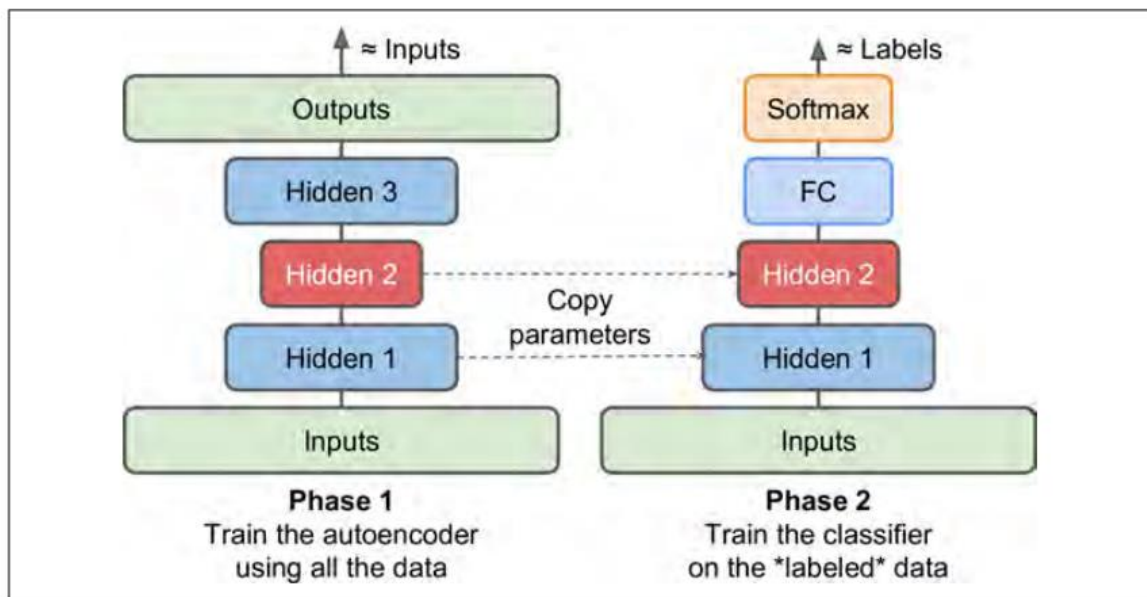


Figure 15-8. Unsupervised pretraining using autoencoders

- 왜 lower layer를 재사용? Lower layer에서 추출된 특성이 더 단순하거나 일반적인 특성을 나타냄
- upper (상위) layer: 고수준의 의미 파악 또는 작업과 관련된 특징을 학습

18. Denoising autoencoder

- 입력 데이터에 임의의 노이즈 추가, 오토인코더가 이 노이즈를 제거하고 정상적인 입력을 복원하도록 학습. 단, 학습 시에만 사용
- **Goal: 입력을 출력에 copy하는 것을 막고 패턴 추출**
- 또는, Dropout 방법 사용, 즉, 훈련 중에 무작위로 선택된 일부 뉴런을 비활성화 시킴
- **Goal: 더 강력하고 일반화된 특성 학습하고 overfitting 줄임**

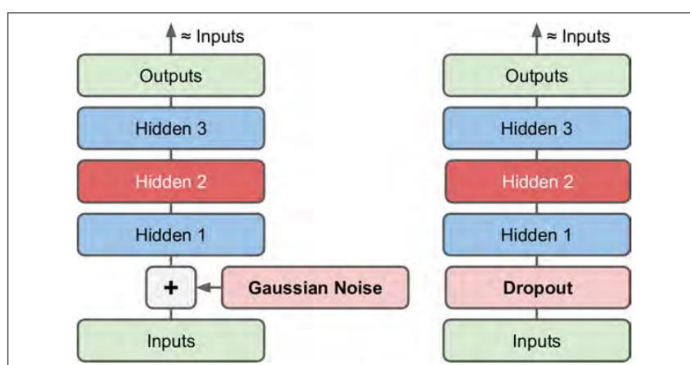
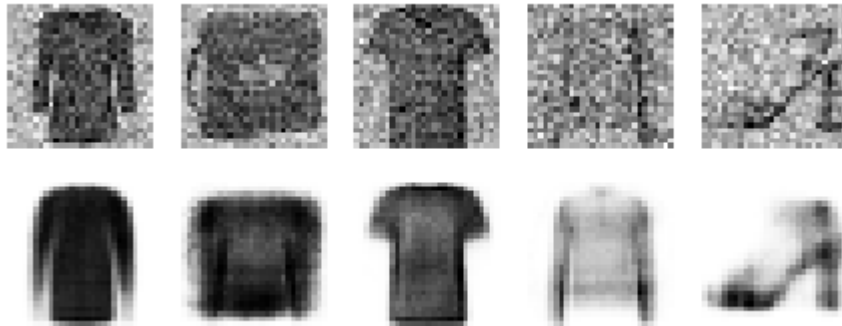


Figure 15-9. Denoising autoencoders, with Gaussian noise (left) or dropout (right)

19. (실습) fashion MNIST dataset을 활용한 stacked denoising autoencoder

- (위) Gaussian noise 더함 (아래) reconstruction



- (위) Dropout 사용 (아래) reconstruction



19. Outlier detection이란?

- dataset에서 일반적인 패턴에서 벗어나는 이상치를 찾는 데이터 마이닝 기술. 이상치는 대부분의 데이터 포인트와는 다르게 동작하는 관측치로 정의됨

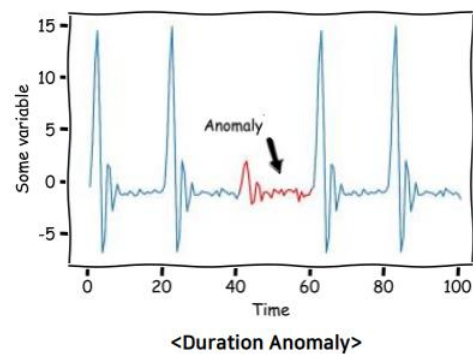
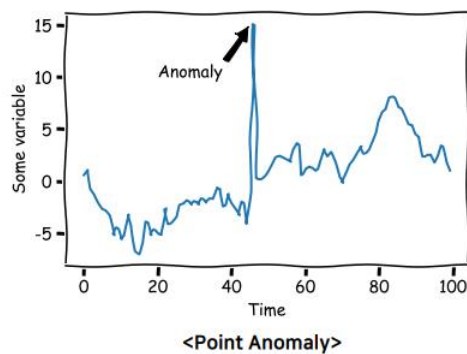
- 응용 예

- 1) **사기 탐지(Fraud Detection)**: 금융 거래나 신용 카드 사용과 같은 금융 활동에서 이상치를 감지하여 사기 행위를 식별
- 2) **제조업에서의 품질 관리(Quality Control in Manufacturing)**: 제조 과정에서 제품 불량을 감지하고, 이상치를 찾아 공정을 개선
- 3) **네트워크 보안**: 이상치 감지는 네트워크에서의 이상 행위를 식별하고 보안 위협을 탐지하는 데 사용

20. Autoencoder를 사용한 이상치 감지 (outlier detection)

- 기본 idea: 정상 시계열 데이터로 autoencoder 학습 후에 새로운 test 데이터에 대해서 reconstruction error를 사용해서 정상 패턴에서 벗어난 이상치 탐지

1. 시점 이상: 비정상적인 값에 도달한 단일 시점 또는 연속적인 시점의 집합
2. 기간 이상: 특정 구간에서 이상 현상 발견



유명한 논문: LSTM-based encoder-decoder for multisensory anomaly detection, ICML 2016.

LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, Gautam Shroff
{MALHOTRA.PANKAJ, ANUSHA.RAMAKRISHNAN, GAURANGI.ANAND, LOVEKESH.VIG, PUNEET.A, GAUTAM.SHROFF}@TCS.COM
TCS Research, New Delhi, India

Abstract

Mechanical devices such as engines, vehicles, aircrafts, etc., are typically instrumented with numerous sensors to capture the behavior and health of the machine. However, there are often external factors or variables which are not captured by sensors leading to time-series which are inherently unpredictable. For instance, man-

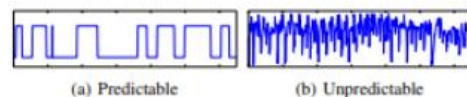


Figure 1. Readings for a manual control sensor.

example, a laden machine behaves differently from an un-

1) **입력 데이터 준비:** 정상 시계열 데이터를 사용해서 autoencoder 학습

2) **Autoencoder 구조 설정:** RNN/LSTM/Transformer와 같은 시계열 데이터에 적합한 autoencoder의 구조 선택

3) **Reconstruction loss 정의:** 목적 함수로 입력과 재구성된 출력 간의 손실을 측정하는 reconstruction loss 정의. 주로 MSE 사용

4) **Autoencoder 학습:** 정상 데이터 만을 사용하여 autoencoder 학습. 학습이 끝나면 모델은 주어진 입력을 정상 패턴으로 잘 복원하는 능력을 갖게 됨

5) **test data로 autoencoder 평가:** 학습한 autoencoder를 사용하여 test 데이터를 사용하여 reconstruction 시도 후 reconstruction error 계산

6) **Threshold 설정:** 정상 데이터의 reconstruction error 분포를 고려하여 임계값 설정. 임계값 이상의 reconstruction error를 갖는 데이터는 이상치로 간주

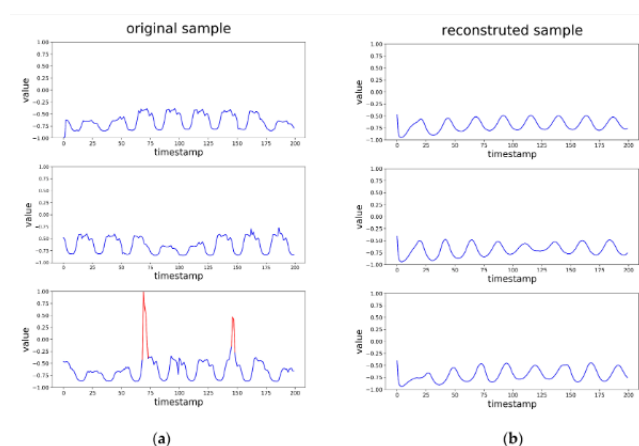
정상 데이터로 autoencoder로 학습 후에

(왼쪽) 새로운 test data 입력

(오른쪽) 왼쪽의 입력에 대한 reconstruction

Outlier 탐지: 왼쪽과 오른쪽의 차이가 너무 크면, 즉 threshold 이상이면

젤 아래 데이터와 같이



CHUNGNAM NATIONAL UNIVERSITY
충남대학교

차량 운행 데이터를 활용한 딥러닝 기반 고장 유형 분류 및 예측

Deep learning-based failure classification and prediction using vehicle operation data

손유지¹, 정민석¹, 김나영¹, 박소현¹, 김영국¹, 이규환²
¹Youji Sohn⁰¹, Minseok Jeong¹, Nayoung Kim¹, Sohyeon Park¹, Young-Kuk Kim¹, Gyuhyeon Lee²
¹Department of Computer Science & Engineering, Chungnam National University
²INFOCAR

Abstract

본 논문에서는 시동 꺼짐 등 차량의 심각한 고장이 발생하기 전에 이를 운전자에게 미리 알려서 이에 대비할 수 있는 서비스를 제안한다. 기존에는 차량의 고장을 수동으로 진단하는 서비스만 제공되었으며 선행된 차량 고장 예측 연구에서는 고장코드의 유형을 분류하지 않고 1가지 고장코드를 선정하여 그 고장코드를 기준으로 정상인지 아닌지만을 예측하였다. 본 연구에서는 선행연구에서 개선된 성능을 보였던 LSTM-AutoEncoder 모델로 1차적인 이상 탐지를 수행하였으며 다음 단계로 MLP Classifier를 통해 5가지의 고장코드 중 어떠한 고장코드의 흐름을 보이는지 예측하는 실험을 진행하였다. 전체적인 정확도는 93%의 성능을 보였으며 정확도보다 더 의미 있는 성능평가치인 F1-score는 라벨별로 상이하였으나 평균 약 79%의 성능을 나타냈다.

Introduction

- 차량의 갑작스러운 고장은 운전자들에게 불편과 안전 문제를 가져올 수 있으며 주행 중 고장이 발생하게 되면 교통사고의 위험성도 증가할 수 있음
- 최근에는 차량과 통신이 가능한 OBD2 기기를 차량에 장착하여 차량의 현재 상태 정보와 고장 진단 결과를 운전자에게 알려주는 서비스만 제공되고 있음
- 차량의 고장이 발생한 이후에 운전자에게 이를 알리게 되면 운전자가 해당 고장으로 인한 피해를 받을 가능성이 존재하기 때문에 차량 고장 발생 이전에 전조증상을 감지하여 운전자에게 알려줄 수 있는 시스템의 필요성이 대두되었음
- 따라서 고장 이전에 발생하는 데이터의 이상 흐름을 감지하여 고장이 발생하기 이전에 예측하고 이를 운전자에게 미리 고장을 알려 적절한 선제 조치를 하게 함으로써 운전자의 안전을 확보할 수 있고 갑작스러운 고장으로 인한 피해를 줄이는 것이 목표

Related Study

- 다양한 분야에서 예지보전 연구가 진행되고 있으나 단순히 정상 상황과 정상이 아닌 상황으로 구분되는 이진 분류 연구가 대부분임
- 또한, 선행연구[1]에서 차량 데이터를 활용하여 고장 진단 및 예측을 진행하여 94.4%의 정확도와 83%의 Macro F1-score를 보였으나 발생해도 주행에 미미한 영향을 주는 한 가지 유형의 고장코드만 예측하였다는 한계점이 있었음
- 본 연구에서는 다양한 고장을 예측하는 서비스를 제안하기 위해 먼저 시계열 특성을 가진 차량 주행 데이터를 활용하여 이상치가 발생하는 시점을 탐지하기 위해 비지도 학습 기반의 딥러닝 모델인 LSTM-AutoEncoder를 사용하여 이상치 여부를 파악하고 MLP(Multi-Layer Perceptron)를 추가적으로 사용하여 주행 중 발생할 경우 차량 정지로 이어질 수 있는 위험한 고장 유형 중에 어떠한 고장인지 판별하고자 함

Failure Prediction Model

- '인포카' 앱을 통해 수집된 실제 주행 데이터를 사용하여 각 주행 데이터는 차량 센서 값들의 흐름이며 차량 주행 시작한 이후 차량의 속도, 엔진 회전수, 냉각수 관련 정보, 내부 압력, 흡입되는 공기의 온도 등의 약 30개의 운행 정보가 차량 주행 중론 미전까지 1초 단위로 수집되어 저장
- 티켓으로는 수백 개 이상의 고장코드 중에서 주행 중 발생하였을 때 적절한 조치를 취하지 않고 지속적으로 주행하게 된다면 차량 정지나 시동 꺼짐과 같이 심각한 상황으로 이어질 수 있는 5가지 고장코드를 선정하였으며 선정된 고장코드의 의미는 표 1과 같음
- 차량 운행 데이터는 약 30개의 운행 정보 중 정확도가 95% 이상인 항목은 제거하고 타겟이 되는 고장코드 5가지와 관련이 있는 운행 정보를 선정하여 최종적으로 26개의 항목이 사용하였음
- Standard Scaler를 통해 각 항목별 평균이 0, 표준편차가 1인 분포로 변환

| 고장코드 | 의미 |
|-------|-----------------------------|
| P0035 | 크랭크 샤프트 위치 센서 회로 이상 |
| P0122 | 낮은 스로틀 액추에이터 포지션 센서 회로의 신호값 |
| P0135 | 산소 센서 이터 회로 이상 |
| P0339 | 캠캠크루크 위치 센서 B 회로 오작동 |
| P0562 | 낮은 전압 시스템 |

Step 1. LSTM-AutoEncoder

- 주어진 데이터의 95%가 정상인 경우이므로 비지도 학습 기반으로 정상 라벨 데이터를 활용하여 학습하는 LSTM 레이어 기반의 AutoEncoder 모델 구조로 1차적인 이상 탐지를 진행

Step. 2 MLP Classifier

- LSTM-AutoEncoder를 거쳐 정상 혹은 고장으로 예측된 데이터는 MLP를 통해 5가지 유형 중 어떤 고장코드의 흐름을 보이는지 예측하게 됨

Experiments & Results

- 실제 데이터로 정상 주행 데이터 약 5백47만 개, P0035 주행 데이터 약 3만 개, P0122 주행 데이터 약 8만 개, P0135 주행 데이터 약 6만 3천 개, P0339 주행 데이터 약 8만 개, P0562 주행 데이터 약 4만 개를 사용
- 정상 라벨 데이터는 82로 나누어 80%에 해당하는 데이터를 LSTM-AutoEncoder 모델에 사용하였으며 각 고장코드도 라벨링 된 데이터들과 20%의 정상 라벨 데이터를 사용하여 MLP Classifier에 학습
- 수학적 정렬을 적용하고 메모리를 효율적으로 사용하기 위해 범주형 데이터에 해당하는 정상, P0035, P0122, P0135, P0339, P0562 라벨을 각각 0, 1, 2, 3, 4, 5로 변환하여 사용

| | precision | recall | f1-score | accuracy |
|----------|-----------|--------|----------|----------|
| 0(정상) | 0.96 | 0.98 | 0.97 | 0.93 |
| 1(P0035) | 0.86 | 0.60 | 0.71 | |
| 2(P0122) | 0.77 | 0.90 | 0.83 | |
| 3(P0135) | 0.91 | 0.83 | 0.87 | |
| 4(P0339) | 0.82 | 0.84 | 0.83 | |
| 5(P0562) | 0.96 | 0.38 | 0.54 | |

- 전체 데이터 중 정답을 맞춘 비율을 의미하는 정확도는 약 93%이지만 데이터가 불균형 할 경우에는 정확도에 성능을 평가하기 어렵다는 단점이 있으므로 모델이 고장이라고 예측한 것 중에 실제 고장인 데이터의 비율을 의미하는 정밀도와 실제 고장인 데이터 중 모델이 고장이라고 예측한 것의 비율을 의미하는 재현율을 활용한 F1-score를 성능평가 지표로 사용
- 정상 상태를 의미하는 라벨 0의 F1-score는 97%로 가장 우수한 성능을 보임
- 고장코드 P0562를 의미하는 라벨 5는 재현율이 38%로 상대적으로 낮아 54%의 F1-score를 기록하였으며 6개의 클래스 중 가장 낮은 성능을 보임
- 매코 평균은 각 클래스의 F1-score를 단순 평균한 값으로 79%의 성능을 보였고, 가장 평균은 각 클래스의 샘플 수를 고려하여 계산된 결과로 93%를 나타냈음

Conclusion

- 차량 운행 데이터를 사용하여 단지 고장 여부만을 예측하였던 기존 연구를 발전시켜 고장 상태일 경우에 어떠한 고장 유형 상태를 의미하는지 예측하는 다중 분류 태스크를 수행하였음
- 고장 유형별로 대응 방안이 다르므로 운전자에게 적절한 대응 방안을 제안하기 위해서는 높은 성능이 클래스별로 일관해야 하나 여전히 모델의 라벨별 예측 성능이 일정하지 않다는 한계점 존재
- 성능 불균형 문제는 실제 운영 환경에서의 고장 예측 성능에도 영향을 줄 수 있으므로 향후 연구에서는 이 문제점을 개선하기 위한 방안을 모색 필요
- 실제 환경에서는 여러 개의 고장코드가 동시에 발생하는 경우도 존재하기 때문에 2개 이상의 고장 코드도 동시에 예측하는 태스크에 관한 연구도 진행할 계획

Reference

[1] 손유지, et al. 차량 데이터를 활용한 LSTM-AutoEncoder 기반 차량 고장 예측 모델의 성능 개선. 대한전자공학회 학계학술대회논문집, pp. 2084-2087, 2023.

21. (실습) Autoencoder를 사용한 이상 감지 (outlier detection)

22. Autoencoder 모델의 한계 및 개선

- 정상 데이터만으로 학습하기 때문에, 다른 데이터가 들어와도 training set과 비슷하게 만드는 overfitting 문제가 발생하기 쉬움
- 이와 같은 문제를 해결하고자 variational autoencoder, autoencoder ensemble 방법과 GAN 기반의 outlier detection 등 다양한 방법이 등장