MEMPUTE V3

Neuromorphic Hybrid Machine Learning

- Neuronet -

뉴로모픽 하이브리드 학습 머신은 인간의 사고 과정을 모방한 알고리즘과 심층 신경망을 결합하여 기존 신경망의 부족한 성능을 한층 강화한 학습 및 추론 엔진이다.

NeuroNet을 이상탐지에 적용한 테스트 결과

- 2만건 데이터 셈프링, 비정상 34건 나머지 정상 데이터
- 변별력을 위해 비 정상데이터를 기준으로 점수 산정

신경망 오토인코더 테스트 결과

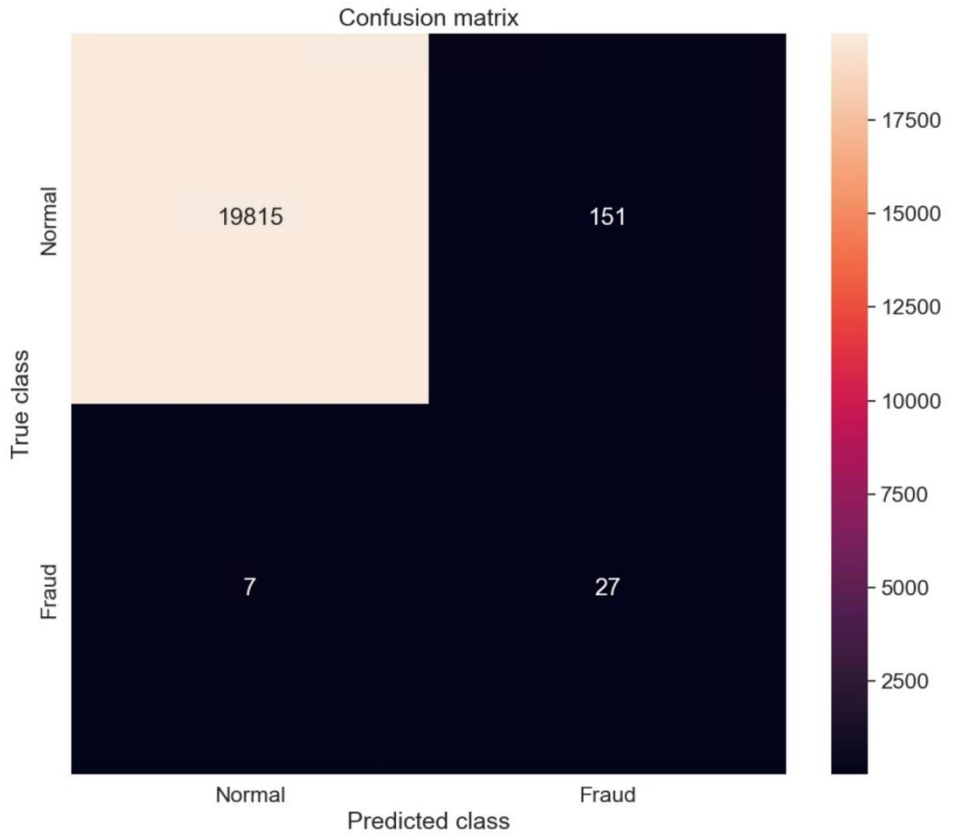
f1 score: 0.10485436893203885 recall: 0.056133056133056136 precision: 0.7941176470588235
normal-normal : 19512 normal-proud : 454 proud:normal : 7 proud:proud : 27

NeuroNet테스트 결과

f1 score: 0.12958963282937366 recall: 0.06993006993006994 precision: 0.8823529411764706
normal-normal : 19567 normal-proud : 399 proud:normal : 4 proud:proud : 30

f1 score: 0.7540983606557378 recall: 0.8518518518518519 precision: 0.6764705882352942
normal-normal : 19962 normal-proud : 4 proud:normal : 11 proud:proud : 23

f1 score: 0.3529411414413293 recall: 0.22689075611185652 precision: 0.7941176447231834
normal-normal : 19874 normal-proud : 92 proud:normal : 7 proud:proud : 27



신경망 오토 인코더 방식과 비교하여 뉴로넷은 하이퍼파라미터 조정에 따라 다양한 결과가 산출되고 모든 케이스에서 기존 방식을 크게 웃도는 예측 성능을 나타내며 향후 챗봇과 같은 다양한 시계열 분야에 적용할 경우 상당한 개선 결과가 기대된다.

Bayesian Pattern Probability Inference Engine

- Pattern Database Green Technology -

Pattern Database는 빈도수에 기반하여 패턴을 발견하고 베이지안 확률 추론을 수행하는 베이지안 패턴 확률 추론 머신이다. 학습과정은 주어진 데이터에 대하여 에포크 한번에 완결된다.
Pattern Database는 메모노드라는 분산 퓨전 관계형 데이터베이스위에서 구성된다. 모든 발견 패턴은 베이터베이스에 저장되고 추론에 사용된다. 메모노드 데이터베이스에 관한 내용은 www.memonode.com을 참고하고 언급된 모든 예제들은 깃허브https://github.com/mempute 에 있으며 **Pattern Database**에 관한 예제는 https://github.com/mempute/anormal 위치에 mempute_pattern_chain_database_example.ipynb 이 있다.

Pattern Database는 데이터에서 빈도수에 의해 규칙을 발견하고 이를 패턴화한다. 패턴들은 서로 경쟁하고 가장 높은 강도의 것이 **invoke**되어 추론에 사용된다.
Pattern Database는 목표값을 타겟팅 하기위해 목표값과의 오차로부터 역전파로 파라미터 오차를 조정하는 연역방식이 아니다. 데이터로 부터 라벨없이 빈도수에 의해 데이터를 가장 잘 설명하는 패턴을 발견하게 되며 이 과정은 귀납적이다. 목표값과의 연결 역시 가장 연결 강도가 높은 연결을 스스로 발견하고 사전 학습하므로써 추론시에 사후 예측한다. 이 모든 과정은 귀납식이다.
Pattern Database는 시간대 혹은 이벤트 등의 범주 단위 정도의 연관성이면 된다. 우리는 많은 경우에 입력에 잘 정합되는 목표값을 알지 못하며 이럴때 **Pattern Database**는 주어진 입력에 가장 가능성 높은 목표를 사후확률로 예측해 낸다. 연역방식인 신경망과 달리 귀납방식으로 추론함으로써 과적합의 위험없이 정확한 추론이 가능하다.
또한 멀티 소스대 타겟을 연결 학습하여 소스중 타겟에 정상성이 가장 높은 입력 요소를 추론하여 발견해 낸다.

이러한 기능성으로 **Pattern Database**는 예측, 분류, 타겟 라벨의 자동생성, 데이터 정상성 판별, 오류 데이터 필터링 등 빅데이터 처리의 모든 광범위한 영역에서 신경망뿐 아니라 이제까지 발명되어진 어떠한 데이터 처리 기술보다 다변적 기능을 강력하고도 범용적으로 수행할 수 있다.
또한 신경망과의 하이브리드 학습을 수행하여 신경망을 더 광범위한 영역에서 부스팅하여 기존에

신경망으로 해결하지 못했던 난제들을 해결할 수 있다.

타겟 라벨 생성은 높은 레벨의 패턴 추상화로 이루어 진다. 신경망 이나 기타 기술은 이러한 기능이 없으며 신경망은 잠재코드의 차원을 작게 줄이면 정보는 붕괴되어 의미없어 진다. 언어 모델에서 발전된 트랜스포머는 압축을 수행하지 않으며 컨볼루션 망은 압축을 위해 최대값만을 선택함으로써 많은 정보가 손실된다. 깃허브에 이상데이터 판별 예제에서 **Pattern Database**로 생성한 타겟 라벨으로 신경망과 하이브리드 학습을 진행한 결과를 보면 높은 수준의 제한된 라벨은 학습되어 판별 효과가 있었으나 더 넓은 범위의 낮은 레벨의 라벨은 목표값으로 학습되지 못하였다. 신경망은 매우 제한된 높은 정합성을 갖은 타겟 범주의 학습만 가능할 뿐이다. 언어의 경우 **one hot**으로 분류되어야 할 어휘 사이즈가 **10000**개도 쉽게 넘을 수 있고 신경망의 언어 모델은 이를 달성하나 언어라는 것은 문법도 있고 의미적으로도 우리는 정형된 패턴으로 사용하기에 가능한 것이다. 개나 고양이 분류에서 보듯 이러한 분류들도 또한 정상성이 매우 높은 데이터이다.

다만 이문서의 두번째 파트인 **mempute** 신경망 버전의 강력한 시계열망인 제너릭 망으로는 생성된 라벨로 하이브리드 학습을 진행할수록 오차가 줄어들어 학습이 됨을 확인할 수 있다. 그러나 규칙이 희박한 자연데이터로부터 넓은 범위의 이산 라벨을 수렴시키는 것은 신경망에게는 매우 버거운 일로써 학습시간이 오래 걸린다. 사전학습은 인코더를 생성하는 과정인데 **Pattern Database**의 추상화된 라벨은 단일 혹은 몇 개 차원 정도로 입력에 대한 분류 태그를 제공하여 이를 타겟으로 인코더 학습시킬 경우 더 높은 사전학습 결과를 얻을 수 있을 것이다.

Pattern Database는 비지도 학습, 타겟 연결 학습, 사전학습, 전이학습, 증강학습 및 보상에 따른 패턴 강도를 조정으로 강화학습등 다양한 학습을 수행 할 수 있다.

Pattern Database는 학습의 결과로 발견된 패턴을 데이버테이스에 저장하고 계속적인 추가학습으로 인해 패턴이 데이터베이스에 누적됨에 따라 점점 더 정확하고 오류에 강한 추론이 가능하다. 이는 통계표본이 클수록 모집단과 유사해져 예측 결과가 정확해 지는 것과 같은 이유이다

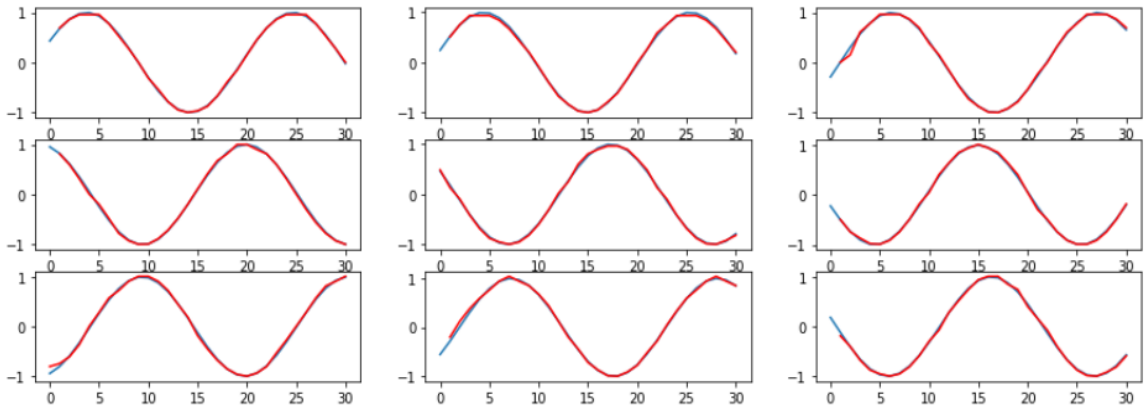
Pattern Database는 고성능의 대용량 분산 관계형 데이터베이스위에서 구동되어 대량의 패턴 학습 및 추론이 가능하며 분산 분할 학습과 패턴 병합을 하여 대량의 패턴 처리를 빠른 시간에 수행할 수 있다.

Pattern Database는 신경망의 강력한 전처리기로서도 사용될 수 있다. 입력데이터에서 타겟을 가장 잘 설명하는 요소를 정확하게 분리하내는 과정은 사람의 판단이나 개입이 필요없어 데이터 학습의 모든 과정을 자동화시킬 수 있다. 이와 같은 기능성으로 **Pattern Database**는 다양한 분야에서 범용적으로 사용될 수 있을뿐만 아니라 이상 탐지나 개인화 추천 시스템등 일반적으로 기존 기계학습 방법으로 효과가 적거나 결과가 애매한 분야에 탁월한 성능을 기대 할 수 있고 추론 결

과에 대한 확률 해석 설명을 제공할 수 있다.

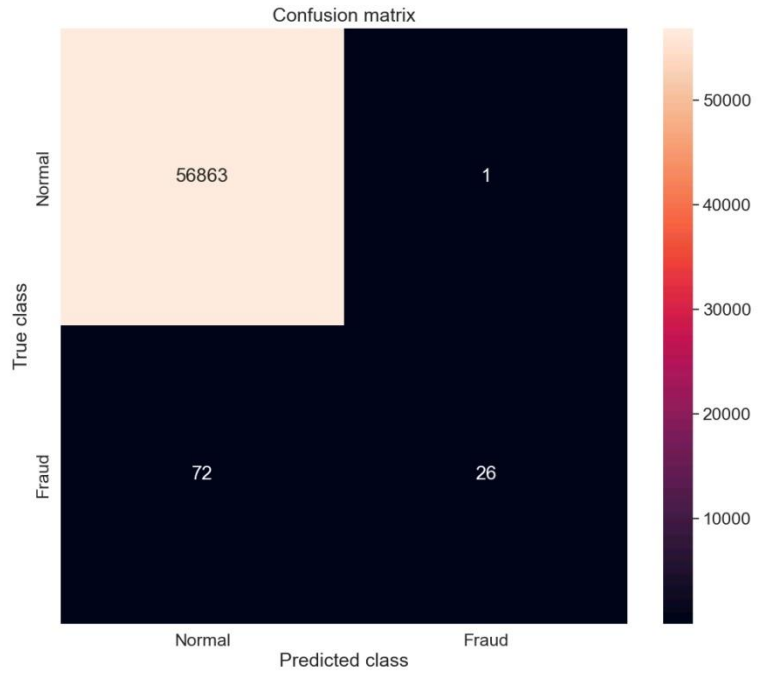
간단한 Pattern Database적용 예

사인 곡선 예측

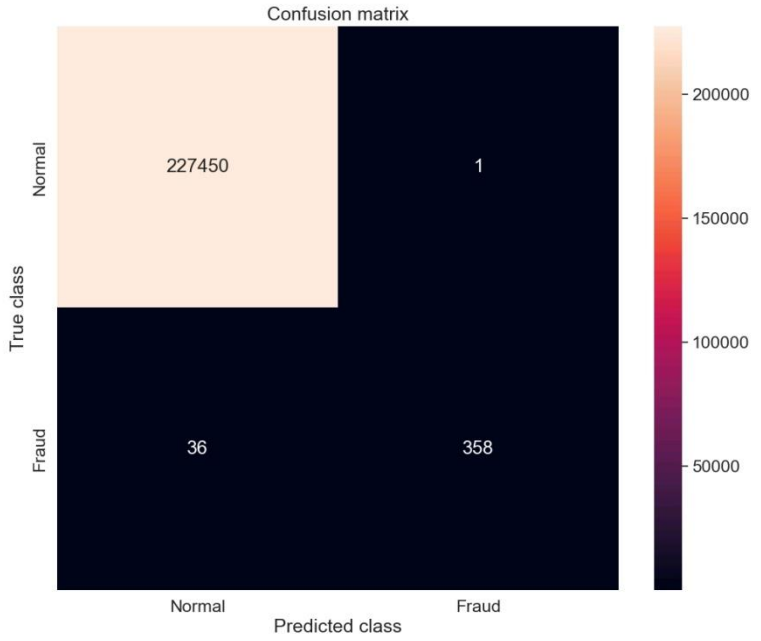


아마존 주가 예측

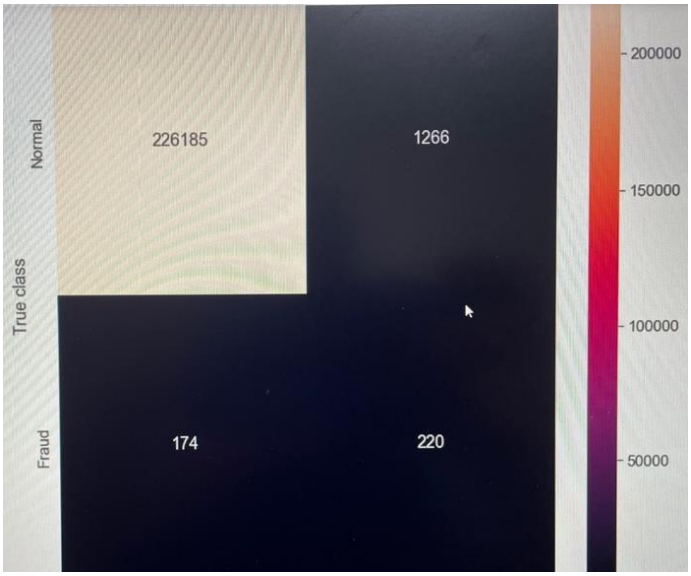
이상 탐지 데이터 예측



이상 탐지 학습 데이터 예측



신경망 학습 데이터 예측



Anormal detection(이상 탐지)는 비정상 데이터가 적어 신경망으로는 지도 학습을 하지 못하고 오토인코더로 학습하는데 테스트 데이터 예측도 정확도가 낮을뿐 아니라 학습데이터를 예측한 결과도 마찬가지이다. 신경망의 목적이 일반화에 있어 당연한 결과인데 이상 탐지의 경우 정상과 비정상의 데이터 비율이 비대칭이어서 학습 데이터가 축적되어도 최총 정확도가 향상될 수 없다.
학습 데이터를 예측한다는 것은 학습데이터에서 가능도 높은 패턴을 발견해 내는 일반화 과정뿐만 아니라 이러한 고 가능도 패턴을 데이터베이스에 직접적으로 계속 누적함으로써 경험치를 높혀 정확도가 지속적으로 향상되는 것을 의미한다.
위 이상탐지의 예를 살펴보면 차원 축소 압축이 2만분의 1로 된 것으로 압축은 일반화의 한 과정이고 학습데이터 예측이 이 정도 높은 일반화로도 높은 정확도로 나온 결과이고 테스트 데이터 예측도 같은 압축으로 나온 결과로서 예측되지 못한 비정상 케이스는 학습 데이터에는 없는 패턴이다. 또한 패턴 데이터베이스와 신경망을 하이브리드 형태로 증강 학습법을 수행한다면 state of the art를 달성 할 수 있을 것이다.
다음은 이상 탐지 예제의 패턴 추론과정의 확률 계산을 출력한 것으로서 이상인데 정상으로 판별하여 추론이 틀린 경우를 설명한다. prior_v는 목표값이고 0은 정상, 1은 이상 임을 나타내고 이상으로 예측해야하나 posterior(사후확률)을 보면 정상 출력이 -10.810163, 이상 출력이 -18.120785 으로서 정상 목표값의 확률이 더 높아 정상으로 판별한 것을 나타낸다. 여기서 pid는 목표값의 아이디, prior st는 사전확률, likely는 event(사건)하에서 발생되는 prior확률인 가능도이다.

event-prior probable result: pid(4249817) prior_v(0.000000) posterior(-10.810163) likely(-10.809829) prior(-0.000334) prior st(2993) prior ast(2994)

event-prior probable result: pid(4249833) prior_v(1.000000) posterior(-18.120785) likely(-10.116419) prior(-8.004366) prior st(1) prior ast(2994)

NEURONET

- MEMPUTE NEUROMORPHIC HYBRID MACHINE LEARNING -

- 메뉴얼 -

뉴로넷의 API 는 파이썬과 C 가 1:1 매칭되며 본 문서는 파이썬을 기준으로 한다.

파이썬 클라이언트 API

neuronet

- 주어진 이름으로 신경망을 생성한다.

neurogate

- Pattern network database 에 패턴망을 생성한다.

close_net

- 신경망 및 패턴망 패쇄.

loadnet

- 뉴로넷 로드.

update_param

- 파라미터를 변경한다.

xtrain

- 뉴로넷 학습.

connect_train

- 패턴 데이터베이스에서 타겟 패턴 연결 학습, 타겟 패턴 연결이 필요없으면 생략.

xpredict

- 뉴로넷 추론.

neuronet(param, gid, name)

- param: 파라미터 딕셔너리
- gid: GPU index
- name: 뉴로넷 이름
- return: 뉴로넷 핸들

neurogate(net, stmt, name, learning_rate, x, y, init)

- net: neuronet()으로 생성된 뉴로넷 핸들
- stmt: pattern net database 핸들

- name: 패턴 망 이름
- learning_rate: 패턴 생성 학습률
- x: 입력값
- y: 목표값
- init: 패턴망 생성이면 1, 로드이면 0

xtrain(net, x, xother, tunning, epoch, relay = 0, decord = 0)
- net: neuronet()으로 생성된 뉴로넷 핸들
- x: 오토인코딩 학습할 학습 데이터
- xother: 디코더 학습할 데이터, x 와 동일하면 None
- tunning: 레벨 학습이면 0, 전체 튜닝 학습이면 1
- epoch: 에포크
- relay: 이전 학습 레벨을 계속하여 학습할려면 1, 아니면 0
- decord: 1 이면 디코더 학습, 0 이면 인코더 학습
- return: next level sequence

connect_train(net, x, y)
- net: neuronet()으로 생성된 뉴로넷 핸들
- x: xtrain() 리턴된 다음 레벨 시퀀스
- y: 패턴망에서 연결 학습될 타겟 데이터

xpredict(net, x, decord = 0)
- net: neuronet()으로 생성된 뉴로넷 핸들
- x: 디코딩 추론할 데이터
- decord: 1 이면 디코더 예측, 0 이면 패턴 네트워크 예측
- return: 추론 결과

update_param (net, param)
- net: neuronet()으로 생성된 뉴로넷 핸들
- param: 변경할 파라미터 딕셔너리

ex) 뉴로넷 생성

```
    import mempute as mp
    from morphic import *

    train_params = dict(
                input_size=inp_size,
                hidden_sz=16,
                learn_rate=1e-4,
                drop_rate=0.1,
                signid_mse = mse,
                wgt_save = 0,
                layer_norm = False,
                n_epair = 3, #encode depth 개수, 1 부터 시작
                residual = 1,
                on_schedule = False,
                dtype = mp.tfloat,
                levelhold = 0.7,
                tunehold = 0.98,
                seed = 0,
                decay = 0.0001,
                decode_active = 1,
                regression = 0,
                dec_lev_learn = 1,
                decode_infeat = dec_infeat_other,
                size_embed = dec_infeat_other + 1 if embedding else 0, #+1 은
    패딩 추가
                batch_size=batch_size)

train_params = default_param(**train_params)
param = param2array(**train_params)
 net = mp.neuronet(param, gid, model_name)
 mp.neurogate(net, stmt, model_name, sensor, x_train, y_train, 1)
 mp.close_net(net)
```

ex) 뉴로넷 로드

```
net, param = mp.loadnet(gid, model_name)
mp.neurogate(net, stmt, model_name, sensor, x_train, y_train, 0)
```

ex) 뉴로넷 학습

```
reent = np.expand_dims(x_train, -1)
for i in range(level):
    reent = mp.xtrain(net, reent, xtrain_other, tuning, epoch, relay)

 mp.connect_train(net, reent, y_train)#나머지 상위 및 연결 학습
```