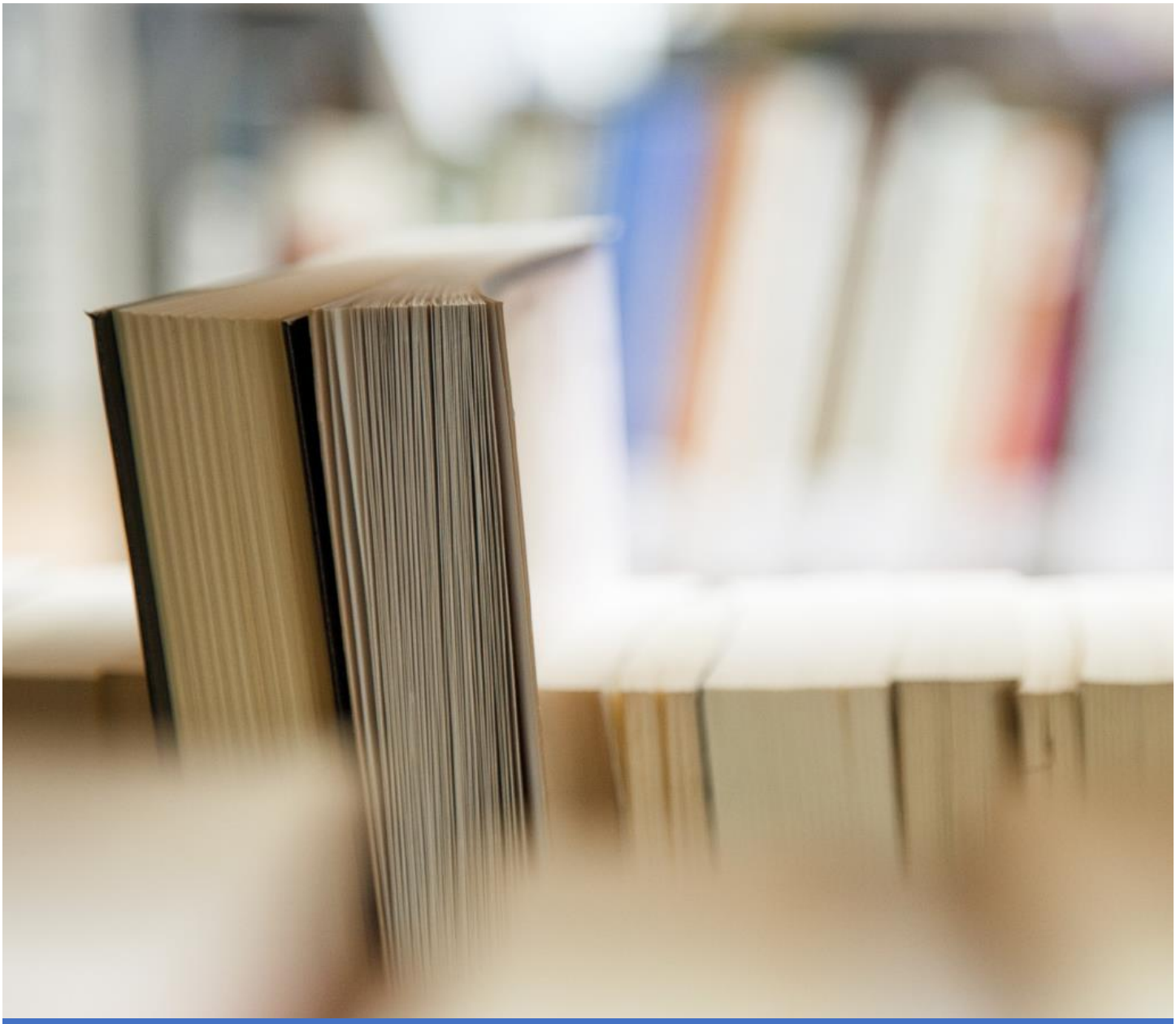


ENSEMBLE DEEP LEARNING FOR REGRESSION AND TIMESERIES FORECASTING

Paper Reproduction

2020021327 조경선[대학원석사과정재학 / 산업경영공학과]

12/6/20

BUSINESS ANALYTICS

Paper Summary

이 논문에서는 회귀 및 시계열 예측을 위해 처음으로 심층신뢰망(DBN) 앙상블이 제안되었다. 또 한 SVR (Support Vector Regression) 모델로 다양한 DBN 의 output 을 집계하였다는 점에서 의의가 있다.

Forecasting Models

(A) Support Vector Regression

사전 훈련된 CNN 은 feature 추출기로 사용한다. 논문에서는 softmax regression layers 가 제거된 AlexNet 과 VGG15 을 사용하였다. 훈련하는 동안 Convolution layer 를 고정하고 완전히 연결된 레이어만 훈련한다. D 차원의 feature 가 추출됐다면, 가우시안으로 부터 생성된 pseudo-negative data 를 feature 에 추가한다.

(B) Artificial Neural Network

원 데이터와 함께 pseudo-negative class data 가 합쳐졌기 때문에, 분류 네트워크의 input 은 배치사이즈 2 이다. Softmax regression layer 를 마지막에 사용하고 그 결과도 2 로 셋팅된다.

(C) Deep Learning Algorithms

딥 러닝 알고리즘은 분산 표현을 기반으로하는 기계 학습 방법이다. 딥 러닝은 여러 비선형 변환으로 구성된 구조를 사용하여 데이터의 학습하려고 한다. 자주 사용되는 모델은 DBN, CNN 및 SAE 이다.

- **Deep Belief Network**: DBN 은 여러 계층의 hidden 유닛으로 구성된 deep neural network 의 한 유형입니다. 각 레이어의 유닛간에 inner-connection 이 없다. DBN 은 unsupervised 방식으로 판별 특성을 추출하는 데 사용할 수 있습니다. 그런 다음 softmax 또는 SVM / SVR 과 같은 supervised 방식을 DBN 위에 추가 할 수 있다.

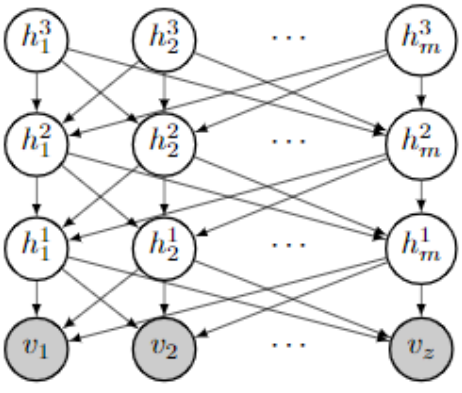


Fig. 2: Flowchart of a three-layer Deep Belief Network (DBN)

RBM 은 입력 데이터 세트에 대한 확률 분포를 학습 할 수있는 신경망이다. v_z는 visible layer unit, h_m는 hidden layer unit, $W_{m\times n}$은 hidden & visible unit 의 연결하는 가중치를 나타내고 b_z, c_m은 visible & hidden layer 의 오프셋을 각각 나타낸다.

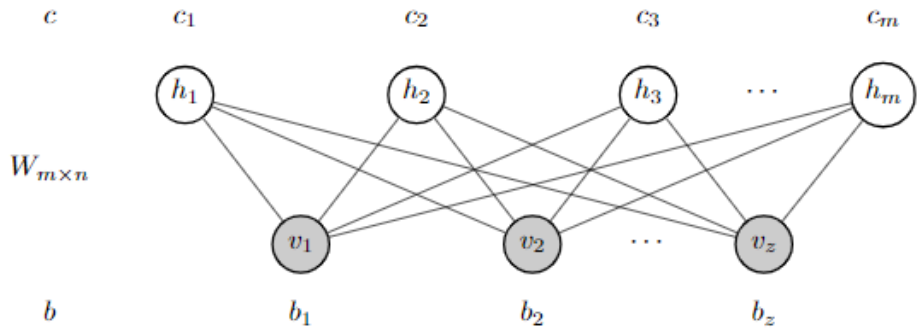


Fig. 3: Schematic Diagram of a Restricted Boltzmann Machine (RBM)

(D) Ensemble Method

앙상블 학습 방법은 여러 학습 알고리즘을 전략적으로 결합하여 더 나은 예측 성능을 얻기위한 기계 학습 프로세스다. 앙상블 방법은 다음과 같은 세 가지 이점이 있다.

1. 잘못된 모델을 선택할 위험을 줄일 수 있음.
2. 다른 시작점에서 많은 로컬 검색을 실행하여 결과를 향상 시킴
3. 함수는 여러 가설의 가중 합계로 더 잘 근사 할 수 있음

Proposed Ensemble Deep Learning Method

회귀 및 시계열 예측의 경우 back propagation 의 훈련 epoch 수가 변경되면 예측 결과가 다를 수 있다. 따라서 우리는 서로 다른 epoch 수로 훈련 된 FNN 에 의해 생성 된 모든 출력을 결합고 각 출력에 해당 가중치 값을 할당하여 전체 예측 출력 값을 계산할 수 있다.
논문에서는 서로 다른 수의 Epoch 를 사용하여 훈련 된 DBN 과 입력을 DBN 의 출력으로, 출력을 최종 예측으로 사용하는 SVR 로 구성된 딥 러닝 알고리즘 앙상블을 수행한다.
자세한 절차는 다음과 같다.

1. 입력 데이터 X 행렬을 사용하여 DBN 훈련.
2. Step size 100 인 back propagation 을 셋팅하고, epochs 를 100 에서 2000 으로 설정하여 20 개의 output 을 얻는다. DBN 은 20 번 다시 초기화됩니다 .
3. 모든 output 을 Xnew 로 셋팅하고 SVR 을 훈련시키는 데 사용한다.
4. 마지막으로 더 정확한 예측 결과를 얻는다.

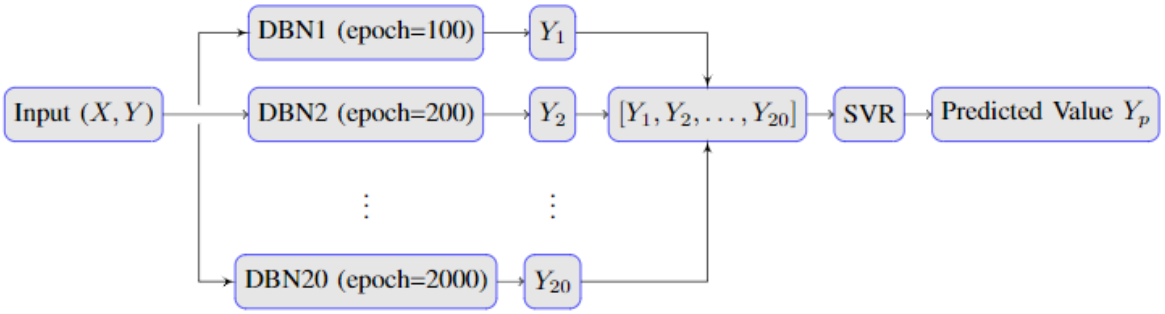


Fig. 4: Schematic Diagram of the proposed Ensemble Deep Learning Network

Reproduction

Data Preperation

```
import numpy as np

np.random.seed(1337)  # for reproducibility
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.metrics.regression import r2_score, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from dbn.tensorflow import SupervisedDBNRegression

# Loading dataset
boston = load_boston()
X, Y = boston.data, boston.target

# Splitting data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=1337)

# Data scaling
min_max_scaler = MinMaxScaler()
X_train = min_max_scaler.fit_transform(X_train)
```

DBN

```
# Training
regressor = SupervisedDBNRegression(hidden_layers_structure=[100],
                                    learning_rate_rbm=0.01,
                                    learning_rate=0.01,
                                    n_epochs_rbm=20,
                                    n_iter_backprop=200,
                                    batch_size=16,
                                    activation_function='relu')
regressor.fit(X_train, Y_train)

# Test
X_test = min_max_scaler.transform(X_test)
Y_pred = regressor.predict(X_test)
print('Done.\nR-squared: %f\nMSE: %f' % (r2_score(Y_test, Y_pred),
mean_squared_error(Y_test, Y_pred)))
```

SVR with DBN output

```
X_train_append = np.append(X_train, Y_train_Pred, axis=1)
Y_train_Pred=regressor.predict(X_train)
X_test_append = np.append(X_test, Y_test_Pred, axis=1)
Y_test_Pred=regressor.predict(X_test)

import pandas as pd
svr = SVR()
svr_linear = {'C': [0.01, 0.1, 1, 10],
              'kernel': ['linear']}
svr_others = {'C': [0.01, 0.1, 1, 10],
              'kernel': ['poly', 'rbf', 'sigmoid']}
parameters = [svr_linear, svr_others]
# parameters = {'C': [0.001, 0.01, 0.1, 1, 10, 25, 50, 100],
#               'kernel': ['poly','rbf','sigmoid','linear']}
grid_svr = GridSearchCV(svr, param_grid = parameters, cv = 5, verbose=2)
grid_svr.fit(X_train_append, Y_train)

result = pd.DataFrame(grid_svr.cv_results_['params'])
result['mean_test_score'] = grid_svr.cv_results_['mean_test_score']
result.sort_values(by='mean_test_score', ascending=False)

from sklearn.svm import SVR
svr_rbf = SVR(kernel='rbf', gamma=0.1, C=10)
svr_poly = SVR(kernel='poly', gamma=0.1, C=10)
svr_linear = SVR(kernel='linear', C=10)
```

Result

```
import time
results = []
print ('-----------')
print ('SVR - linear')
print ('-----------')
results.append(launch_model('SVR - linear', svr_linear, X_train, Y_train,
X_test, Y_test))

from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import GridSearchCV
def launch_model(name,model, X_train, y_train, X_test, y_test):
    start = time.time()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    ypred_train = model.predict(X_train)
    print ('MSE test', mean_absolute_error(y_test, y_pred))
    r_2 = model.score(X_test, y_test)
    print ('R^2 test', r_2)
    print('Tiempo de ejecución: {0:.2f} segundos.'.format(time.time() -
start))
    return name + '($R^2={:.3f}$)'.format(r_2), np.array(y_test), y_pred


def plot(results):

    fig, plts = plt.subplots(nrows=len(results), figsize=(8, 8))
    fig.canvas.set_window_title('Predicting')

    for subplot, (title, y, y_pred) in zip(plts, results):
        subplot.set_xticklabels(())
        subplot.set_yticklabels(())
        subplot.set_title(title)
        subplot.plot(y, label='actual')
        subplot.plot(y_pred, label='predicted')
        subplot.fill_between(
            # Generate X values [0, 1, 2, ..., len(y)-2, len(y)-1]
            np.arange(0, len(y), 1),
            y,
            y_pred,
            color='r',
            alpha=0.2
        )

        subplot.axvline(len(y) // 2, linestyle='--', color='0', alpha=0.2)
        subplot.legend()

    fig.tight_layout()
    plt.show()
    plt.savefig('plot.png')
    plt.close()
```

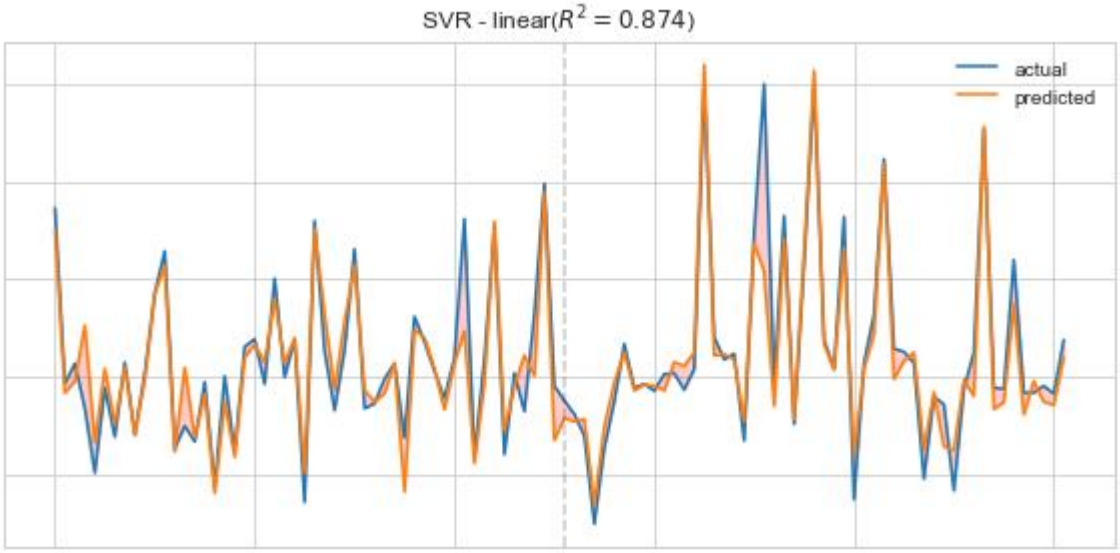
Result And Comparison

Data: sklearn 에서 제공하는 Boston 데이터를 활용하여 Boston 집값 Data

MSE 와 R-squared 를 통해 성능예측을 비교하였다.

SVR 에 비하여 DBN 을 사용하였을 때 R-Squared 값이 향상되었고, 두모델의 앙상블을 이용하였을 때 더 나은 지표를 얻는 것을 확인 할 수 있었다.

	SVR_{linear} (C=10)	DBN	DBN+SVR_{linear} (C=10)
MSE	3.3243	11.0055	2.0564
R-squared	0.6551	0.8671	0.8737



Conclusion

논문에서 제시한 것 처럼 단일 모델보다 앙상블 모델에서 더 나은 결과를 도출할 수 있음을 확인하였다.

아쉬운 점은 논문에서는 SVR 의 커널 종류에 대한 자세한 정보를 얻을 수 없었따. 추후에 SVR 의 kernel 종류를 변경하면서 얻어지는 결과도 비교해보면 좋을 것 같다.

또한 시계열 데이터인만큼 Test 와 Train 데이터의 추출방법에 따라 결과가 많이 달라짐을 확인할 수 있었다. 이 점도 보완하여 더 추가적인 실험을 해 볼 생각이다.