시계열 데이터에 대한 클러스터링 성능 분석: Wavelet 과 Autoencoder 비교

황우성, 임효상
연세대학교 원주캠퍼스 컴퓨터정보통신공학부
e-mail: {zzong2006, hyosang}@yonsei.ac.kr

Clustering Performance Analysis for Time Series Data: Wavelet vs. Autoencoder

Woosung Hwang, Hyo-Sang Lim
Computer and Telecommunications Engineering Division, Yonsei University

요 약

시계열 데이터의 특징을 추출하여 분석하는 과정에서 시계열 데이터가 가지는 고차원성은 차원의 저주(Course of Dimensionality)로 인해 데이터내의 유효한 정보를 찾는데 어려움을 만든다. 이러한 문제를 해결하기 위해 차원 축소 기법(dimensionality reduction)이 널리 사용되고 있지만, 축소 과정에서 발생하는 정보의 희석으로 인하여 시계열 데이터에 대한 군집화(clustering)등을 수행하는데 있어서 성능의 변화를 가져온다. 본 논문은 이러한 현상을 관찰하기 위해 이산 웨이블릿 변환(Discrete Wavelet Transform:DWT)과 오토 인코더(AutoEncoder)를 차원 축소 기법으로 활용하여 시계열 데이터의 차원을 압축 한 뒤, 압축된 데이터를 K-평균(K-means) 알고리즘에 적용하여 군집화의 효율성을 비교하였다. 성능 비교 결과, DWT 는 압축된 차원수 그리고 오토인코더는 시계열 데이터에 대한 충분한 학습이 각각 보장된다면 좋은 군집화 성능을 보이는 것을 확인하였다.

1. 서론

시계열 데이터들이 가지는 고유한 특징을 이용하여 분석하는 것은 데이터 마이닝의 핵심 과제이다. 시계열이란 시간의 흐름에 따라 관측되는 데이터들의 수열을 의미한다. 군집화(Clustering)는 데이터 마이닝의 한 방법으로 분류되어 있지 않은 시계열 데이터를 분류하여 유효한 정보를 찾아내는데 사용될 수 있다[1]. 시계열 데이터의 군집화는 소비 패턴 조사, 의료 모니터링 그리고 유전자 및 ECG 데이터 분석 등에 응용된다[1].

시계열 데이터를 이용하여 군집화를 진행할 때, 고차원성에 의해, 시계열 데이터들 사이에 측정된 유사도가 가지는 의미가 희석되는 결과를 낳는다[2]. 이러한 문제를 해결하기 위해 이산 웨이블릿 변환(Discrete Wavelet Transform:DWT)와 오토인코더(Autoencoder)가 차원 압축 기법으로 활용될 수 있다[3][4]. 차원 축소된 시계열 데이터를 이용하여 군집화를 진행하는 경우, 적용된 차원 축소 기법에 의해 의도치 않게 군집화 성능이 변할 수 있다. 특히, 분류되지 않은 시계열 데이터들을 올바르게 군집화하는 것을 목적으로 하는 경우, 원래 데이터에 기반한 군집화 성능과 차원 축소된 데이터에 기반한 군집화의 성능을 비교할 필요성이 있다.

본 논문에서는 각각의 시계열 데이터가 가진 고차원 성분들을 DWT 와 오토인코더를 이용해 저차원으로 압축하여, 압축된 데이터가 군집화에 미치는 영향을 실루엣 점수(Silhouette score)를 이용해 수치적으로 분석한다. 그 결과 DWT 로 차원 압축된 데이터는 일관성 있고 오토인코더보다 상대적으로 높은 군집화 성능을 보이지만 일정 차원수 이상의 데이터 압축이 필요함을 발견하였다. 또한 오토인코더로 차원 압축된 데이터는 낮은 차원 수에서도 좋은 군집화 성능이 관찰 가능하지만 충분한 학습이 필요함을 발견하였다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구들을 살펴본다. 제 3 장에서는 DWT 와 오토인코더가 군집화 성능에 끼치는 영향을 비교하기 위한 알고리즘을 설명한다. 제 4 장에서는 비교 알고리즘을 사용한 실험 및 결과를 제시하고, 제 5 장에서는 결론을 맺는다.

2. 관련 연구

2.1. 이산 웨이블릿 변환

시계열 데이터는 DWT 를 통해 시간-주파수 영역에서 다중해상도로 표현이 가능하다. 이를 위해 시계열 데이터는 일정 단계까지 웨이블릿을 이용해 분해

· 이 논문은 2018 년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. 2016-0-00179, 데이터 스트림 정제를 위한 지능형 샘플링 및 필터링 기술 개발).

된다. 분해를 통해 생성된 근사 계수(approximation coefficients)와 상세 계수(detailed coefficients)는 다양한 방식으로 시계열 데이터의 특징을 추출하는 차원 압축 기법에 활용할 수 있다[3].

2.2. 오토인코더

오토인코더는 신경망의 한 종류로써, 비지도 방식으로 주어진 데이터를 입력 받아 동일한 또는 유사한 데이터를 출력하도록 학습시킨다. 본 논문에서 사용된 적층 오토인코더는 인코더와 디코더로 구성되어 있는데, 오토인코더의 최중심 은닉 층, 즉, 인코더의 출력 층은 일반적으로 입력 층의 노드 개수보다 작다. 이러한 점을 이용하면 입력 데이터가 가지는 핵심 특징을 인코더에서 추출하는 압축 기법으로 활용할 수 있다[4].

2.3. K-평균(K-means) 알고리즘

K-평균 알고리즘은 군집화에 쓰이는 휴리스틱 방법이다. K-평균 알고리즘을 이용한 군집화를 수행할 때, 군집 내부 데이터들과 중심 사이의 거리를 최소화하기 위해 군집의 중심을 지속적으로 조정한다. K-평균 알고리즘을 통해 형성된 군집은, 군집 내부의 데이터 간에는 서로 높은 유사도를 가지고, 다른 군집에 있는 데이터 사이에는 낮은 유사도를 가지게 되는 특징을 가진다. 본 논문에서는 군집화 과정에서 시계열 데이터간 유사도를 측정하기 위해, K-평균 알고리즘의 특성을 고려하여, 이상적 측량값(metric)으로 유클리드 거리를 사용한다[5][6].

2.4. 실루엣 점수

실루엣 점수는 군집화 성능을 가늠할 수 있는 값으로, 군집화된 데이터들의 결과를 이용하여 그 성능을 단 하나의 점수로 표현한다[7]. 실루엣 점수가 높을수록 비슷한 특징을 지닌 데이터끼리 같은 군집 안에 잘 밀집해 있고 다른 특징을 지닌 데이터와는 다른 군집에 위치하여 멀리 떨어져 있다는 것을 의미하며, 따라서 군집화 성능이 좋은 것으로 판단된다. 본 논문은 시계열 데이터의 차원 압축으로 생성된 성분들이 K-평균 알고리즘의 군집화 성능에 기여하는 정도를 알아보기 위하여 실루엣 점수를 활용한다.

3. 비교 알고리즘 소개

차원 축소가 군집화에 미치는 영향을 비교하기 위해서 1) 시계열 데이터를 DWT 와 오토인코더를 사용하여 각각 차원 축소하고, 2) 그 결과를 K-평균 알고리즘을 사용하여 군집한 다음, 3) 군집화된 결과에 대해서 각각 실루엣 점수를 계산하여 비교하는 3 단계로 알고리즘을 구성하였다(그림 1). 첫 번째 단계에서는 먼저 주어진 시계열 데이터를 Z-정규화(Z-normalization) 한다(스텝 1-2). 각 시계열 데이터의 성분들의 평균이 0 그리고 표준 편차가 1 이 되도록 정규화 함으로써, 군집화를 진행할 때 K-평균 알고리즘이 시계열 데이터가 가지는 진폭보다 구조적 유사성에 집중하여 데이터를 분류하도록 한다[8]. 다음으로 정규화된 시계열 데이터를 DWT 와 오토인코더에 적용하여 차원 축소를 수행한다(스텝 3-12). DWT 에서는, 웨이블릿 기반의 다해상도 분석을 통해 분해된 시계열 데이터에서 근사 계수만을 취하여 차원 축소를 진행한다. 이때 입력 값인 분해 단계 $STEP_{DWT}$ 을 통해 근사 계수의 개수를 조절하여 축소될 차원수를 정한다. 오토인코더의 경우 먼저, 과적합(overfitting) 문제와 계산량 완화를 위해, 미리 설정된 세대 $epoch_{AE}$ 값과 배치(batch) 크기 B_{size} 를 이용하여 시계열 데이터를 오토인코더에 반복 학습시키고, 다시 학습데이터로 사용한 시계열 데이터와 학습된 오토인코더의 인코더를 이용하여 차원 축소된 데이터를 얻는다. 두 번째 단계에서는 차원 축소된 시계열 데이터에 K-평균 알고리즘을 적용해 군집화를 진행한다(스텝 13). 마지막으로 세 번째 단계에서는 군집화를 통해 얻어진 군집 번호 집합과 각 번호에 연관되는 원 시계열 데이터를 이용하여 실루엣 점수를 측정한다(스텝 14).

(그림 1) 성능 비교 알고리즘

성능 비교 알고리즘

입력:
(1) k 개의 시계열 데이터 $S_0, \dots, S_{k-1}$
(2) 군집화 군집 개수 n
(3) 이산 웨이블릿 변환의 분해 단계 $STEP_{DWT}$
(4) 오토인코더 세대 $epoch_{AE}$ 및 배치 크기 B_{size}
출력: 군집 개수 n에 따른 $S_0, \dots, S_{k-1}$의 실루엣 점수 $score_{DWT}$ 및 $score_{AE}$
알고리즘:
1. for i = 0 to k − 1:
2. 시계열 데이터 S_i 에 대해 Z-정규화 진행
3. for i = 0 to k − 1:
4. 시계열 데이터 S_i 를 이산 웨이블릿 변환하여 $STEP_{DWT}$ 단계까지 분해 진행
5. 이산 웨이블릿 변환 결과의 근사 계수들만 취하여 벡터 v_i 을 구성
6. for i = 0 to $epoch_{AE}$:
7. j = 0
8. while j < k:
9. 시계열 데이터S_0 … S_{k-1}중 j 에서 순차적으로 B_{size}만큼 택하여 오토인코더 학습
10. j += B_{size}
11. for i = 0 to k − 1:
12. 오토인코더의 인코더 부분을 통해 S_i를 압축하여 벡터 R_i 생성.
13. 벡터 v 와 R에 대하여 군집 개수 n개를 가지는 k-평균 군집화를 각각 수행하여 벡터 v의 할당 군집 번호 집합 $v\_labels\_ = \{a_0, \dots, a_{i-1}\}$ 와 벡터 R 의 할당 군집 번호 집합 $R\_labels\_ = \{b_0, \dots, b_{i-1}\}$ 생성
14. 생성된 집합 $v_labels_$과 원 시계열 데이터 그리고 집합 $R_labels_$과 원 시계열 데이터의 실루엣 점수 $score_{DWT}$ 및 $score_{AE}$를 각각 측정

4. 실험 결과 및 분석

4.1. 실험 데이터 및 환경 소개

실험 데이터로는 UCR Time Series Classification Archive 에서 제공하는 미리 분류된 두 개의 시계열 데이터를 사용하였다[9]. 첫 번째 시계열 데이터 Phoneme 는 구글 번역, 옥스포드 그리고 메리엄-웹스터 온라인 사전에서 각각 22,050, 44,100 그리고 11,025 의 샘플 주파수로 기록된 영어 발음이다. 총 39 개의 클래스로 분류되어 있고, 길이는 1024 이다. 두 번째 시계열 데이터 InsectWingBeatSound 는 센서를 통해 숫모기와 암모기의 소리를 기록한 것이다. 총 11 개의 클래스로 분류되어 있고 길이는 256 이다.

실험을 진행한 하드웨어 환경 및 소프트웨어 환경 표 1 과 같다.

<표 1> 실험 하드웨어 및 소프트웨어 환경

CPU	Intel® Core(TM) i7-4790K 4.00GHz
RAM	DDR3 8GB 1600MHz
GPU	GeForce GTX 1080Ti GDDR5 11GB
OS	Ubuntu 16.04.3 LTS
DWT	pywavelet[10]
오토인코더	Keras[11] & Tensorflow
K-평균 군집화	Scikit-learn[12]

실험에 사용된 DWT 및 오토인코더의 구성은 각각 표 2, 표 3 과 같다. DWT 에서는 웨이블릿의 한 종류인 하(Haar) 웨이블릿을 사용하여 주어진 시계열 데이터를 변환하여 다해상도 분해하였다. 오토인코더에서는 각 계층에 활성화 함수 Parametric Rectified Linear Unit(PReLU)를 적용하여 Gradient Vanishing 과 Dying ReLU 문제를 완화하였다[13][14]. 실험에 사용할 시계열 데이터의 수량적 한계를 고려하여 오토인코더 훈련을 위한 학습 횟수 $epoch_{AE}$와 배치 크기 B_{size}를 달리하였다. 그리고 차원 압축 정도에 따른 군집화 성능의 변화를 확인하기 위해, DWT 와 오토인코더에서는 분해 단계를 조절하여 생성될 근사 계수의 개수와 인코더의 출력 노드 개수를 각각 지정하였다.

<표 2> 실험 DWT 구성

웨이블릿 종류	Haar
시계열 데이터	분해 단계 $STEP_{DWT}$ (압축된 차원수)
Phoneme	3(128), 4(64), 5(32), 6(16)
InsectWingbeatSound	2(64), 3(32), 4(16), 5(8)

<표 3> 실험 오토인코더 구성

데이터	Phoneme	InsectWingbeat
데이터 길이 S	1024	256
계층 별 노드 개수	$S, S/2, S/4, D, S/4, S/2, S$	
압축된 차원수 D	16, 32, 64, 128	32, 16, 8, 4
학습률	0.005	
활성화 함수	Parametric Rectified Linear Unit	
최적화 함수	Adamax	
오차 함수	Mean Squared Error	
세대($epoch_{AE}$)	100	130
배치 크기(B_{size})	32	64

DWT 와 오토인코더를 통해 차원 압축된 시계열 데이터는 K-평균 알고리즘을 이용해 최소 2, 최대 100 개의 군집을 형성한다. 군집화의 수행 시간을 단축시키고 군집 중심이 극소값(local minimum)에 수렴하는 문제를 완화하기 위해, 군집 중심의 초기값 설정에 K-평균++ 알고리즘을 사용하였다[15].

실험에 사용된 시계열 데이터는 미리 분류된 데이터이므로, 분류된 클래스 개수만큼의 군집 개수를 정하여 군집화를 진행하는 것이 최적인지 확인하기 위해, 각 시계열 데이터의 클래스 개수인 39(Phoneme) 및 11(InsectWingbeatSound)개를 군집 개수로 정하여 비교 알고리즘으로 실루엣 점수를 측정한다. 동시에 군집 개수를 100 개까지 설정하고 군집화를 진행하였을 때 비교 알고리즘을 통해 얻을 수 있는 실루엣 점수의 최대치를 측정한다. 실루엣 점수를 측정할 때 원 시계열 데이터 간의 거리를 계산하기 위해 유클리드 거리를 사용하였다.

4.2. 실험 결과

DWT 와 오토인코더의 차원 축소가 군집화 성능에 미치는 영향을 확인한다. 이를 위해 본 실험에서는 차원 축소 기법과 압축된 차원 수에 따른 실루엣 점수, 그리고 차원 축소를 진행하지 않은 원 시계열 데이터를 군집화하여 얻어진 실루엣 점수를 비교하여 분석한다. 시계열 데이터 Phoneme 을 이용한 실험 결과는 DWT 와 오토인코더 각각에 대해서 표 4, 표 5 와 같다. 시계열 데이터 InsectWingbeatSound 를 이용한 실험 결과는 각각 표 6, 표 7 와 같다. 원 시계열 데이터에 대해 군집화를 진행한 실험 결과는 표 8 과 같다.

먼저 시계열 데이터의 종류에 상관없이, DWT 을 이용한 차원 압축에서는 압축된 차원수가 증가할수록 실루엣 점수가 상승하면서 최대 실루엣 점수 및 원 시계열 데이터의 실루엣 점수에 근접하는 결과를 얻었다. 동시에 최대 실루엣 점수와 연관된 군집 개수는 지정된 군집 개수로 사용된 각 시계열 데이터의 클래스 개수에 근접하였다.

시계열 데이터 Phoneme 를 오토인코더로 차원 축소 할 경우 압축된 차원수가 증가하더라도 실루엣 점수의 상승 효과가 DWT 에 비해 상대적으로 적었다. 최대 실루엣 점수와 관련된 군집 개수의 경우, 압축된 차원수가 큰 영향을 주지 않음을 확인할 수 있다. 반대로 시계열 데이터 InsectWingbeatSound 를 이용한 오토인코더 실험 결과에서는 압축된 차원수가 증가할수록 실루엣 점수가 DWT 의 실험결과와 같은 경향을 보인다.

오토인코더의 실험 결과의 경우, 손실 값이 낮아질수록 실루엣 점수가 높아지는 것을 확인할 수 있다. 손실 값이란 오토인코더의 오차 함수로 계산된 값으로써, 본 실험에서는 0 에 가까울수록 오토인코더가 차원 압축된 데이터를 원 시계열 데이터에 유사하게

디코더로 재구성 할 수 있다는 의미 정보로 활용한다. 이를 통해 오토인코더가 입력된 시계열 데이터에 대해 충분히 훈련되지 못한다면, 인코더에서 출력되는 차원 압축된 데이터가 가지는 원 시계열 데이터의 대표적 특징이 희석될 것이고, 결과적으로 실루엣 점수를 감소하게 만드는 결과로 해석된다. 반대로 오토인코더에서 시계열 데이터를 충분히 훈련시킬 수 있다면, 압축된 차원수가 일정 수 이상 보장되어야 향상된 실루엣 점수를 보이는 DWT 와 달리, 상대적으로 적은 차원수를 이용하여도 보다 많은 차원수에서 얻을 수 있는 실루엣 점수와 근접한 점수를 측정할 수 있었다. 이러한 결과로 볼 때, 적은 수의 압축된 차원에서는, 데이터의 종류 및 훈련 정도에 따라 DWT 보다 오토인코더가 군집화 성능에 대한 기여도가 더 높을 수 있을 것으로 판단된다.

<표 4> DWT 실험 결과 (Phoneme)

압축된 차원수	16	32	64	128
실루엣 점수 (지정된 군집 개수)	−0.01 (39)	0.009 (39)	0.018 (39)	0.017 (39)
최대 실루엣 점수 (군집 개수)	0.014 (2)	0.013 (19)	0.019 (41)	0.019 (47)

<표 5> 오토인코더 실험 결과 (Phoneme)

압축된 차원수	16	32	64	128
손실 값	0.691	0.667	0.661	0.639
실루엣 점수 (지정된 군집 개수)	−0.01 (39)	−0.01 (39)	−0.003 (39)	−0.003 (39)
최대 실루엣 점수 (군집 개수)	0.008 (2)	0.007 (2)	0.006 (2)	0.006 (2)

<표 6> DWT 실험 결과 (InsectWingbeatSound)

압축된 차원수	8	16	32	64
실루엣 점수 (지정된 군집 개수)	0.076 (11)	0.168 (11)	0.188 (11)	0.189 (11)
최대 실루엣 점수 (군집 개수)	0.130 (5)	0.189 (8)	0.198 (9)	0.198 (9)

<표 7> 오토인코더 실험 결과 (InsectWingbeatSound)

압축된 차원수	4	8	16	32
손실 값	0.532	0.446	0.394	0.331
실루엣 점수 (지정된 군집 개수)	0.151 (11)	0.146 (11)	0.166 (11)	0.181 (11)
최대 실루엣 점수 (군집 개수)	0.169 (8)	0.178 (8)	0.177 (9)	0.186 (12)

<표 8> 원 시계열 데이터에 대한 군집화 결과

시계열 데이터	Phoneme	InsectWingbeatSound
데이터 길이	1024	256
실루엣 점수 (지정된 군집 개수)	0.0212 (39)	0.1898 (11)
최대 실루엣 점수 (군집 개수)	0.0212 (39)	0.1994 (9)

5. 결론

본 논문에서는 DWT 와 오토인코더를 통하여 차원 압축된 시계열 데이터가 가지는 특징이 K-평균 알고리즘으로 수행된 군집화의 성능에 기여하는 정도를 관찰하였다. 관찰을 위해 차원 압축된 데이터의 군집화 결과에서 얻어지는 군집 번호 집합과 원 시계열 데이터의 실루엣 점수를 측정하여 비교하는 방법을 사용하였다. 오토인코더에 비해 훈련 시간이 필요하지 않은 DWT 는, 일정 기준 차원수 이상으로 시계열 데이터를 압축하여 군집화를 진행한다면 오토인코더보다 안정적이고 향상된 실루엣 점수를 보였다. 이에 반해, 기준 이하로 압축된 데이터는 충분히 학습된 오토인코더를 통해 DTW 보다 높은 실루엣 점수를 얻을 수 있음을 확인하였다.

참고문헌

[1] T. Warren Liao (2005) “Clustering of time series data—a survey” Pattern Recognition 38(11): 1857-1874.

[2] Charu C. Aggrawal, et al. (2001) “On the Surprising Behavior of Distance Metrics in High Dimensional Space” ICDT 2001: 420-434.

[3] Fabian Mörchen (2003) “Time series feature extraction for data mining using DWT and DFT” Technical Report No. 33.

[4] G. E. Hinton and R. R. Salakhutdinov (2006) “Reducing the Dimensionality of Data with Neural Networks” Science, 313(5786):504–507.

[5] Vit Niennattrakul, et al. (2007) “On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping” MUE 2007: 733-738

[6] Eamonn J. Keogh, et al. (2003) “On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration” Data Min. Knowl. Discov. 7(4): 349-371.

[7] Rousseeuw, et al. (1987) “Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis” J. Comput. Appl. Math. 20: 53-65.

[8] Thanawin Rakthanmanon, et al. (2012) “Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping” KDD 2012: 262-270.

[9] http://www.cs.ucr.edu/~eamonn/time_series_data/

[10] https://pywavelets.readthedocs.io/en/latest/index.html

[11] https://keras.io/

[12] http://scikit-learn.org/stable/modules/clustering.html

[13] Lie Xu, et al. (2016) “Deep Sparse Rectifier Neural Networks” IWAENC 2016: 1-5.

[14] Kaiming He, et al. (2015) “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification” ICCV 2015: 1026-1034

[15] David Arthur, et al. (2007) “k-means++: The Advantages of Careful Seeding” SODA 2007: 1027-1035