https://doi.org/10.33778/kcsa.2023.23.4.061

딥러닝 기반의 딥 클러스터링 방법에 대한 분석★

권 현*, 이 준**

요 약

클러스터링은 데이터의 정답값(실제값)이 없는 데이터를 기반으로 데이터의 특징벡터의 거리 기반 등으로 군집화를 하는 비지도학습 방법이다. 이 방법은 이미지, 텍스트, 음성 등 다양한 데이터에 대해서 라벨링이 없이 적용할 수 있다는 장점이 있다. 기존 클러스터링을 하기 위해 차원축소 기법을 적용하거나 특정 특징만을 추출하여 군집화하는 방법이 적용되었다. 하지만 딥러닝 기반 모델이 발전하면서 입력 데이터를 잠재 벡터로 표현하는 오토인코더, 생성 적대적 네트워크 등을 통해서 딥 클러스터링의 기술이 연구가 되고 있다. 본 연구에서, 딥러닝 기반의 딥 클러스터링 기법을 제안하였다. 이 방법에서 오토인코더를 이용하여 입력 데이터를 잠재 벡터로 변환하고 이 잠재 벡터를 클러스터 구조에 맞게 벡터 공간을 구성 및 k-평균 클러스터링을 하였다. 실험 환경으로 pytorch 머신러닝 라이브러리를 이용하여 데이터셋으로 MNIST와 Fashion-MNIST을 적용하였다. 모델로는 컨볼루션 신경망 기반인 오토인코더 모델을 사용하였다. 실험결과로 k가 10일 때, MNIST에 대해서 89.42% 정확도를 가졌으며 Fashion-MNIST에 대해서 56.64% 정확도를 가진다.

Analysis of deep learning-based deep clustering method

Hyun Kwon*, Jun Lee**

ABSTRACT

Clustering is an unsupervised learning method that involves grouping data based on features such as distance metrics, using data without known labels or ground truth values. This method has the advantage of being applicable to various types of data, including images, text, and audio, without the need for labeling. Traditional clustering techniques involve applying dimensionality reduction methods or extracting specific features to perform clustering. However, with the advancement of deep learning models, research on deep clustering techniques using techniques such as autoencoders and generative adversarial networks, which represent input data as latent vectors, has emerged. In this study, we propose a deep clustering technique based on deep learning. In this approach, we use an autoencoder to transform the input data into latent vectors, and then construct a vector space according to the cluster structure and perform k-means clustering. We conducted experiments using the MNIST and Fashion-MNIST datasets in the PyTorch machine learning library as the experimental environment. The model used is a convolutional neural network-based autoencoder model. The experimental results show an accuracy of 89.42% for MNIST and 56.64% for Fashion-MNIST when k is set to 10.

Key words : Clustering, Deep learning, Autoencoder, Latent vector

접수일(2023년 04월 25일), 수정일(2023년 05월 16일), 게재확정일(2023년 10월 06일)

★ 본 논문은 화랑대연구소의 2023년도(2023B1015) 저술활동비 지원을 받아 연구되었음.

* 육군사관학교 AI·데이터과학과 부교수(주저자)

** 호서대학교 게임소프트웨어학과 교수(교신저자)

1. 서 론

클러스터링(Clustering)[1]은 비지도학습 방법으로써 정답값(실제값)이 없는 데이터를 대상으로 특징벡터들의 거리를 측정하여 군집 형성하는 머신러닝 방법이다. 이 클러스터링 방법은 기존 지도학습 방법과 다르게 데이터셋의 정답값이 없기 때문에 데이터에 대한 별도의 라벨링 작업이 없이 적용 가능한 장점이 있다. 실제로 지도학습 방법을 적용하기 위해서 데이터의 정답값을 할당하는 라벨링 작업이 필요하며 상당한 시간과 에너지가 소요되기 때문에 지도학습 방법에 비해 상대적으로 클러스터링 방법이 실제 생활에 적용하기에 유리한 점이 있다.

클러스터링에서 중요한 것은 데이터를 특징벡터로 구성하는 것이다. 각 데이터를 어떤 벡터로 표현하느냐에 따라서 클러스터링의 성능에 영향을 미친다. 통상 차원 축소 방법을 적용하거나 특정 특징들만 추출하는 방법들이 적용되지만 최근 딥러닝 모델을 이용하여 데이터를 특징벡터로 압축하여 표현하는 방법들이 있다. 딥러닝 기반에서 특징벡터로 압축하는 방식으로 오토인코더(Autoencoder) 방법[8]이 사용된다. 오토인코더 방식은 인코더(encoder), 잠재 벡터(latent vector), 디코어(decoder)로 구성되어 있으며 인코더로 입력값을 압축하여 잠재 벡터를 생성한다. 이 잠재 벡터를 이용하여 디코어를 통해 다시 복원하여 원래 입력값과 동일한 출력값이 나오도록 인코더와 디코더의 가중치(파라미터)를 업데이트를 한다. 여러번의 학습과정을 통해서 인코더와 디코더의 가중치(파라미터)가 업데이트가 완료가 되면, 새로운 입력값에 대해서 인코더를 통해서 압축되어 생성된 잠재 벡터는 그 입력값의 특징을 잘 대변하는 벡터가 된다. 이 잠재 벡터를 입력 데이터의 특징벡터로 간주하여 클러스터링을 하면 효과적으로 잘 클러스터링이 된다. 딥러닝 기반 딥 클러스터링에서 한 개의 데이터의 표현은 오토인코더의 잠재 벡터 형태로 변환하여 표현한다.

이 논문에서 우리는 딥러닝 기반의 딥 클러스터링에 대한 성능분석을 하였다. 이 방법에서 컨볼루션 모델(Convolution neural network)[9]을 이용하여 입력 데이터를 잠재 벡터로 형성하고 이 잠재 벡터를 클러스터링하는 방법에 대해서 분석을 하였다. 기존 연구에서 클러스터링을 하기 위해 딥러닝 기반이 아닌 차원 축소 방법 등을 이용하여 사용자가 정한 특징들에 대해서만 벡터 공간에 표현을 해야 했기 때문에 사용자가 특징들을 잘못 선정할 경우, 클러스터링의 성능이 저하되는 문제가 있다. 하지만 딥러닝 모델을 기반으로 잠재 벡터로 차원을 축소하면 이러한 점을 해결할 수가 있다. 이 연구의 공헌점은 다음과 같다. 첫 번째로, 딥러닝 기반의 딥 클러스터링에 대한 구조와 동작 원리에 대해서 분석하였다. 또한, 국방 적용 가능 분야에 대해서도 분석을 하였다.

두 번째로, 딥러닝 모델을 통해서 입력데이터를 잠재 벡터를 변환하고 변화된 벡터를 K-mean 평균으로 시각화하고 성능을 측정하였다. 또한, 활성화 함수에 따른 성능차이에 대해서 분석을 하였다.

세 번째로, 딥러닝 기반의 딥 클러스터링의 성능 분석을 위해서 MNIST[10]와 Fashion-MNIST[11]에 대해서 분석을 하였고 타겟 모델은 컨볼루션 모델 기반의 오토인코더로 하였다.

이 논문의 나머지 구성은 다음과 같다. 2장에서 클러스터링에 관한 관련연구를 소개하고 3장에서 딥러닝 기반의 딥 클러스터링 방법에 대해서 설명한다. 4장에서 실험 환경, 실험 결과, 분석에 대해서 기술하였다. 마지막으로 5장에서 이 논문의 결론으로 구성되어 있다.

2. 관련연구

2.1 클러스터링의 분류

클러스터링의 분류는 대표적으로 k 평균 클러스터링(k-mean clustering)[2], 계층적 클러스터링(hirerachical clustering)[3], 밀도 기반 클러스터링(density-based clustering)[4]이 있다. k 평균 클러스터링 방법은 임의의 k개 군집 중심점을 설정한 후에 각 데이터마다 거리를 측

정하여 그 중 가장 가까운 군집 중심점에 해당하는 군집으로 데이터를 분류하는 방법이다. 이때, 거리는 유클리드 거리를 사용하여 각 좌표값의 차이 제곱근으로 거리를 계산한다. 보편적으로 많이 사용되는 클러스터링 방법이지만 k개의 군집을 설정해야하고 임의의 k개가 어디에서 시작되느냐에 따라 성능 차이가 발생하는 특징이 있다. 계층적 클러스터링은 상향식 접근 방법과 하향식 접근 방법이 있는데 보통 상향식 접근 방법을 적용한다. 나무모형(트리)[5] 구조처럼 각 데이터를 하나의 군집으로 설정하고 가장 가까운 군집끼리 병합하여 전체 데이터가 한 개의 군집이 될 때까지 반복한다. 이런 과정을 통해서 형성된 나무모형과 같은 구조를 덴드로그램(Dendrogram)[6]이라고 한다. 이 계층적 클러스터링 방법은 별도의 k개 설정이 없이 덴드로그램을 통해서 사용자가 원하는 군집의 개수를 분석할 수 있는 장점이 있다. 밀도 기반 클러스터링은 데이터 간의 밀도를 기반으로 군집하는 방식이다. 데이터의 특징 상 밀도가 높은 지역과 낮은 지역이 있는 불균형 데이터의 경우, 밀도 기반 클러스터링이 적합하다. 또한 이 클러스터링은 지정된 밀도 안에 들어오지 못하는 이상치(또는 잡음)를 군집하지 않는 장점도 있다. 밀도 기반 클러스터링의 동작 원리는 적절한 밀도 반경과 반경 안에 있어야 할 데이터의 최소수를 지정한다. 데이터로부터 밀도 반경에 있는 점 중에 코어점과 경계점을 구분하고 이에 속하지 않은 점들을 잡음으로 구분한다. 코어점 들 중에 특정 반경 안에 있을 경우 서로 연결하여 하나의 군집을 형성하는 방식으로 동작한다. 위와 같이 대표적으로 3가지 방식으로 클러스터링 기법을 적용한다.

2.2 딥러닝 기반의 딥 클러스터링

딥러닝 기반의 딥 클러스터링의 분류는 딥 싱글 뷰 클러스터링(Deep single-view clustering) [12, 13], 세미-지도학습 기반의 딥 클러스터링(Deep clustering based on semi-supvervised learninig) [13, 14], 멀티 뷰 학습 기반 딥 클러스터링(Deep clustering based on multi-view learning)[15, 16], 전송 학습 기반 딥 클러스터링(Deep clustering based on trasnfer learning)[17]이 있다.

먼저, 딥 싱글 뷰 클러스터링 방법은 한가지 딥러닝 모델을 통해서 잠재 벡터를 생성하는 것을 의마한다. 하지만 이러한 방법은 해당 딥러닝 모델의 높은 상관관계가 있기 때문에 다른 딥러닝 모델에 적용하기가 제한될 수가 있다. 이러한 방법으로 오토인코더를 이용하는 방법, 생성 적대적 방법(Generatvie adversarial net)이 있다.

두 번째로 세미-지도학습 기반의 딥 클러스터링 방법은 부분적으로 정답값(실제값)이 있는 데이터가 있고 부분적으로 정답값이 없는 데이터에 적용될 수 있는 방법이다. 이 방법은 정답값이 있는 데이터를 이용한 방법으로 손실함수 구성시 정답값에 해당되는 제약조건의 정보를 한 개를 더 추가하여 손실함수를 구축하여 이 손실함수를 최소화하여 잠재 벡터를 생성하는 방법을 제안한다. 이 방법은 기존 정답값이 있는 데이터를 통해서 좀 더 향상된 딥 클러스터링의 성능을 가져올 수가 있다.

세 번째로, 멀티 뷰 학습 기반 딥 클러스터링은 서로다른 딥러닝 모델에서 입력 데이터에 대한 각각의 잠재 벡터를 구성한다. 이 방법은 다양한 모델에서 나온 입력 데이터에 대한 잠재 벡터를 이용하여 싱글 뷰 딥 클러스터링 기반에 각각 방법들을 앙상블로 구성한다.

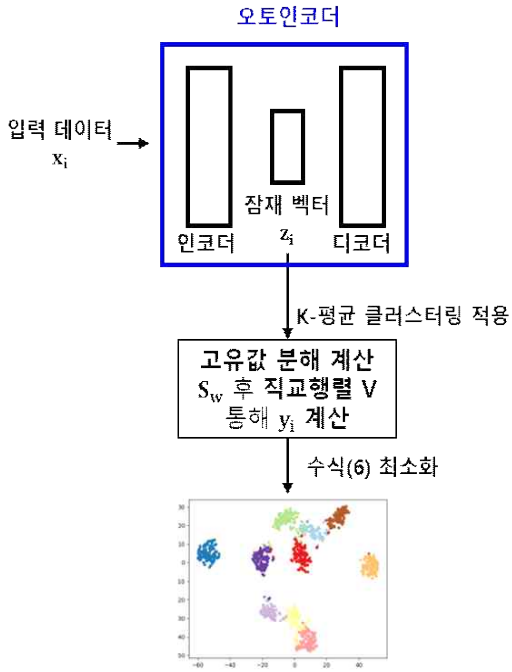
네 번째로, 전송 이전 기반 딥 클러스터링은 고차원이고 데이터의 양이 부족한 환경에서 추가적인 정보를 추가하여 딥 클러스터링의 성능을 개선한 방법이다. 두가지 종류의 잠재 벡터로

구성된 것이 있고 두가지 종류가 서로 유사성이 있다고 한다면 두가지 종류의 잠재 벡터를 동시에 사용하여 딥 클러스터링의 성능을 개선한 방법이다.

이 연구에서는 싱글 뷰 학습 기반 딥 클러스터링 방법[18]을 적용하여 오토인코더를 이용하여 잠재 벡터를 형성한다. 이 잠재 벡터를 클러스터 구조를 반영하기 위해 직교행렬을 곱하여 새로운 벡터로 표현 하고 k-평균 기법을 적용한 클러스터링 방법이다. 자세한 내용은 다음 3장에 설명되어 있다.

3. 제안방법

딥 클러스터링 방법으로 절차는 (그림 1)과 같이 입력데이터를 잠재 벡터로 변환하는 과정, k-평균 클러스터링 하는 과정, 클러스터를 최적화하는 과정으로 3가지 단계로 구성이 된다.



(그림 1) 딥 클러스터링의 구조

먼저, 입력 데이터를 잠재 벡터로 구성하는 과정에서 오토인코더를 이용한다. 오토인코더는 입력 데이터와 출력 데이터를 같게 하는 방법으로 입력 데이터를 인코더로 압축한 후에 잠재 벡터를 만든다. 이 잠재 벡터를 다시 디코더를 통해서 복원하여 입력 데이터와 같은 출력 데이터를 구성한다. 오토인코더의 수식은 아래와 같다.

$$\min L_1 = \sum_{i=1}^{n} \| x_i - \hat{x}_i \|^2 \quad (1)$$
$$= \sum_{i=1}^{n} \| x_i - g(f(x_i)) \|^2$$

여기서 x_i는 입력 데이터이고 $\hat{x}_i$은 출력 데이터이다. $g(\bullet)$는 디코더이고 $f(\bullet)$는 인코더이다. $f(x_i) = z_i$로 z_i는 잠재 벡터이고 n은 입력 데이터의 총 수이다. 손실함수 L_1를 최소화하여 오토인코더를 학습한다. 학습이 완료된 이후에 각 입력 데이터 x_i를 오토인코더의 인코더를 이용하여 잠재 벡터 z_i를 생성하여 차원 축소 등을 한다.

두 번째로, k-평균 클러스터링을 이용하여 잠재 벡터 z_i에 대하여 k개 군집(Cluster)로 클러스터링을 한다. 각 잠재 벡터 z_i에 대한 k개의 군집 결정은 k개의 군집 중심점 $\mu_i (i = 1, \cdots, k)$과의 유클리드 거리로 가장 가까운 거리에 있는 군집 중심점의 군집으로 할당한다. k-mean 클러스터링에 대한 수식을 표현하면 아래와 같다.

$$\min L_2 = \sum_{i=1}^{k} \sum_{z_i \in C_i} \| z_i - \mu_i \|^2 \quad (2)$$

$$\mu_i = \frac{1}{|C_i|} \sum z_i \in C_i$$

여기서 z_i는 입력 데이터 x_i에 대한 잠재 벡터이고 μ_i는 i 번째 군집 중심점을 의미한다. C_i은 i 번째 군집에 할당된 잠재 벡터의 집합을 의미한다. k는 군집의 수를 의미한다.

잠재 벡터 z_i에 클러스터 구조를 반영하기 위해서, 잠재 벡터 z_i를 직교 행렬(orthogonal matrix) V를 이용하여 $Y = VZ$ 로 새로운 Y 벡터 공간을 생성한다. 여기서 Z는 잠재 벡터 공간이며 $Z = [z_1\ z_2\ z_3\ \cdots\ z_n]^T$ 이다. 수식 (2)를 아래

와 같이 변경할 수가 있다.

$$\min L_2 = \sum_{i=1}^{k} \sum_{z_i \in C_i} \| Vz_i - V\mu_i \|^2$$
$$= \sum_{i=1}^{k} \sum_{z_i \in C_i} (Vz_i - V\mu_i)^T (Vz_i - V\mu_i)$$
$$= \sum_{i=1}^{k} \sum_{z_i \in C_i} (z_i - \mu_i)^T V^T V (z_i - \mu_i)$$
$$= \sum_{i=1}^{k} \sum_{z_i \in C_i} \mathrm{Trace}((z_i - \mu_i)^T V^T V (z_i - \mu_i)) \quad (3)$$

여기서 마지막 단계에서 Trace방법을 사용하였다. 왜냐하면 행렬 사이즈가 1×1이기 때문에 적용이 가능하다. $V^T V = I$이기 때문에, 수식 (3)은 수식 (2)와 동일하다. 수식 (3)를 좀 더 다르게 정리하면 아래와 같다.

$$\min L_2 = \mathrm{Trace}(VS_wV) =$$
$$\mathrm{Trace}\left(V\left[\sum_{i=1}^{k} \sum_{z_i \in C_i} (z_i - \mu_i)^T (z_i - \mu_i)\right] V^T\right) \quad (4)$$

여기서 $S_w = \sum_{i=1}^{k} \sum_{z_i \in C_i} (z_i - \mu_i)^T (z_i - \mu_i)$

이고 k-평균의 각 클래스의 분산 합이다. 즉. 각 클래스 내의 잠재 벡터 z_i 들과 클래스의 평균 사이의 거리를 모두 더한 값이다. V는 직교 행렬이기 때문에 수식 (4)번은 정형적인 trace 최소화 문제로 변경이 가능하다. Rayleigh-Ritz 이론에 의거하여 S_w의 고윳벡터(eigenvectors)는 직교 행렬 V를 찾는 것과 관련이 있다. 고유값이 작아질수록 $Y = VZ$의 클러스터 구조에서 고유벡터 S_w의 중요성이 증가된다. S_w는 대칭적이고 직교 대각화가 가능하기 때문에 직교 행렬 V를 구할 수가 있다.

위의 내용을 정리하면, 잠재 벡터 z_i에 k-means를 이용하여 S_w를 얻는다. S_w의 고유값 분해를 통해서 V를 얻는다. $Y = VZ$를 통하여 새로운 새로운 Y 벡터 공간을 생성한다.

세 번째로, 두 번째 단계에서 구한 Y는 클러스터 구조에 대한 정보를 가지고 있다. 클러스터 구조 정보 측면에서 Y의 각 차원은 중요성이 있다. 수식 (3)을 아래와 같이 다시 변경할 수가 있다.

$$\min L_2 = \sum_{i=1}^{k} \sum_{y_i \in C_i} \| y_i - m_i \|^2 \quad (5)$$

여기서 $y_i = Vz_i$ 이고 $m_i = V\mu_i$이다. $Y = [y_1 y_2 \cdots y_n]^T$이다. 여기서 Y안에 클러스터 구조 정보를 향상시키기 위해서 최적화를 어떻게 할것인지가 중요하다. 엔트로피를 이용하여 크러스터 정보를 측정한다. 데이터의 엔트로피가 적으면 클러스터 구조의 정보가 높다. 그리드 알고리즘을 이용하여 각 Y의 마지막 차원의 군집 중심점에 가깝도록 하는 방식이다. 이 방법을 통해서 클러스터 구조 정보를 향상시키면서 최적화 할 수가 있따. 그리드 방법은 아래와 같다. 여기서 y_i 값은 $\grave{y}_i$으로 복사하고 m_i의 마지막 차원을 $\grave{y}_i$의 마지막 차원으로 대체한다. 수식으로 표현하면 아래와 같다.

$$\min L_2 = \sum_{i=1}^{k} \sum_{y_i \in C_i} \| y_i - \grave{y}_i \|^2 \quad (6)$$

여기서 미니 배치를 업데이트 하는 방식을 적용한다. 위의 수식을 최적화 함으로써 잠재 벡터 z_i에 대한 새로운 공간을 만들어 낼 수가 있다. 그리고 나서 k-mean를 통해서 군집을 찾고 각 군집 중심점을 찾는다. 특정 기준을 만족할 때까지 위의 과정을 반복한다.

4. 실험 및 평가

딥 클러스터링을 구현하기 위해서 실험환경으로 pytorch 머신러닝 라이브러리[19]를 사용하였으며, 서버는 Intel(R) Core(TM) i3-7100 CPU @ 3.90GHz와 GPU는 GeForce GTX 1050을 사용하였다.

4.1 데이터셋

데이터셋은 MNIST[10]와 Fashion-MNIST[11]을 사용하였다. MNIST는 손글씨 데이터셋으로 0부터 9까지의 숫자데이터셋이다. 흑백이미지이며

가로 28 픽셀, 세로 28 픽셀로 구성되어 있다. 훈련데이터셋은 6만개가 있으며 테스트 데이터셋은 1만개가 있다. Fashion-MNIST는 패션 관련 데이터셋으로 운동화, 셔츠, 바지, 가방, 신발 등에 10가지 종류로 구성되어 있따. 흑백이미지이며 가로 28 픽셀, 세로 28 픽셀로 구성되어 있다. 훈련데이터셋은 6만개가 있으며 테스트 데이터셋은 1만개가 있다.

4.2 오토인코더

입력 데이터를 잠재 벡터(Latent vector)로 변환 시키기 위해서 오토인코더를 학습할 필요가 있다. 오토인코더의 구조는 표 1과 같다. 입력 데이터는 인코더를 통해서 잠재 벡터를 생성한다. 디코더는 잠재 벡터를 이용하여 데이터를 복원하여 입력 데이터와 출력 값이 같게 하도록 오토인코더를 학습한다. 학습이 완료된 후에는 각 입력 데이터 n개에 대하여 잠재 벡터 n개의 값을 생성하는 과정이 있다.

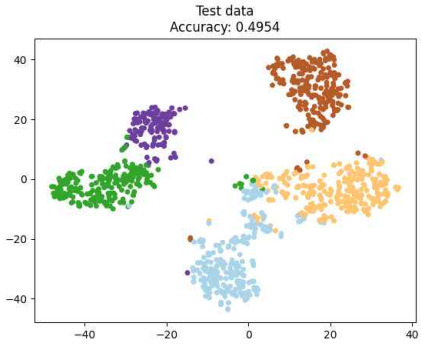
<표 1> 오토인코더의 구조

각 층의 제원	구조
Convolutional + ReLU	필터 5×5, 출력 32
Convolutional + ReLU	필터 5×5, 출력 64
Convolutional + ReLU	필터 5×5, 출력 128
Fully connected layer	입력 3×3×128, 출력 10
Latent vector	10
Fully connected layer	입력 10, 출력 3×3×128
Convolutional + ReLU	필터 5×5, 출력 64
Convolutional + ReLU	필터 5×5, 출력 32
Fully connected layer	입력 5×5×32, 출력 28×28

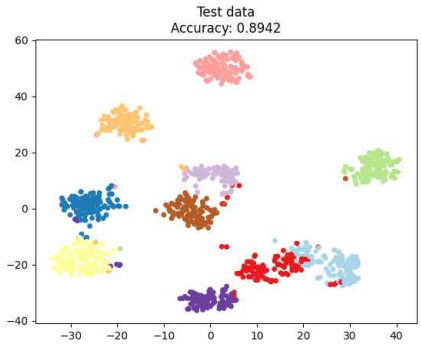
4.3 실험결과

이 섹션에서 k개에 따른 MNIST와 Fashion-MNIST에 대한 딥 클러스터링의 시각화 및 epoech에 따른 정확도를 보여준다. 또한, 활성화 함수에 따른 성능에 대한 분석을 하였다.

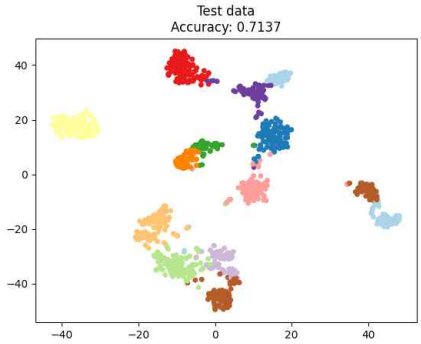
(그림 2)는 MNIST에서 k개에 따른 딥 클러스터링의 실험 결과를 보여준다. 그림에서 보면 특징공간 상에 특성에 맞게 잘 클러스터링이 된 것을 볼 수가 있다. k가 10일 때, 89.42% 정확도로 잘 클러스터링된 것을 볼 수가 있고 각 군집별로 거리도 적절하게 잘 배치된 것을 볼 수가 있다.



(a) k=5인 MNIST 분류 결과



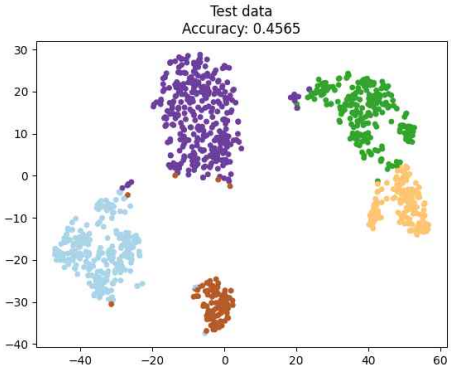
(b) k=10인 MNIST 분류 결과



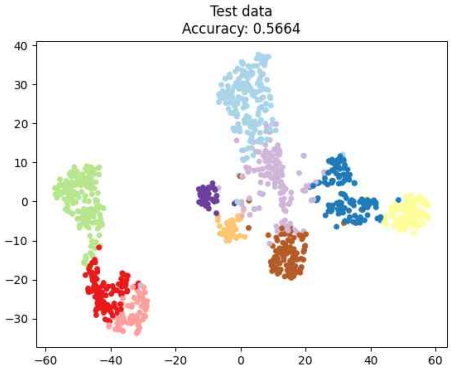
(c) k=15인 MNIST 분류 결과

(그림 2) MNIST에서 k에 따른 딥 클러스터링의 정확도와 시각화

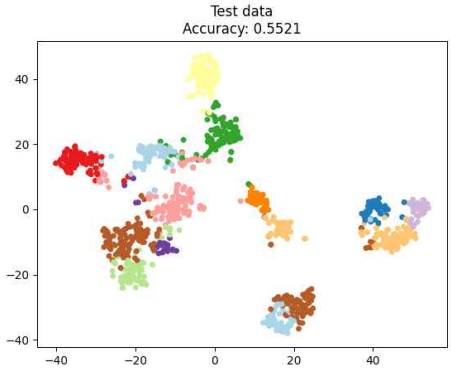
(그림 3)는 Fashion-MNIST에서 k개에 따른 딥 클러스터링의 실험 결과를 보여준다. 그림에서 보면 특징공간 상에 특성에 맞게 잘 클러스터링이 된 것을 볼 수가 있다. k가 10일 때, 56.64%의 성능이 되는 것을 볼 수가 있다. 상대적으로 Fashion-MNIST의 성능이 MNIST에 비해 떨어진 것을 볼 수가 있다. 이는 Fashion-MNIST 상에서 T-shirt와 pullover, shirt가 유사한 클래스가 있기 때문이다. 또한 sandal, sneaker, ankle boot 모형도 서로 유사하게 되어 있어서 데이터 특징상 유사성이 있기 때문에 상대적으로 숫자 데이터에 비해서 성능이 저하된 것을 볼 수가 있었다.



(a) k=5인 Fashion-MNIST 분류 결과



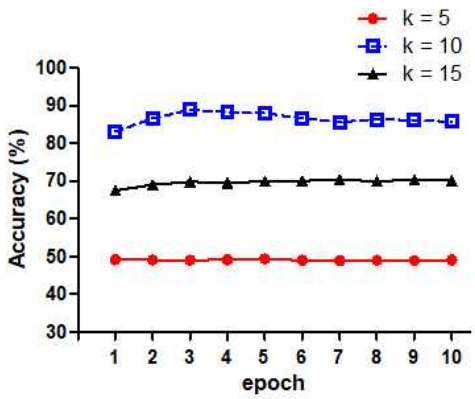
(b) k=10인 Fashion-MNIST 분류 결과



(c) k=15인 Fashion-MNIST 분류 결과

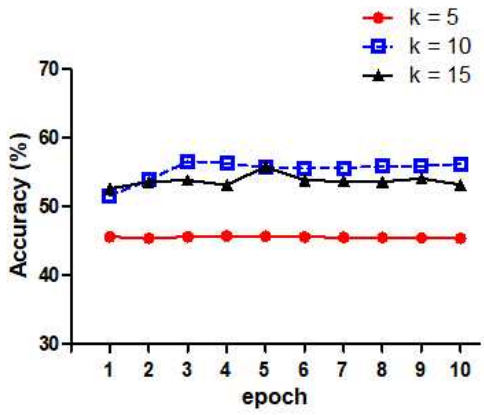
(그림 3) Fashion-MNIST에서 k에 따른 딥 클러스터링의 정확도와 시각화

(그림 4)는 MNIST에서 epoch에 따른 딥 클러스터링의 정확도(accuracy)를 보여준다. 그림에서 보면 k가 10일 때 가장 좋은 성능이 나타난 것을 볼 수가 있고 epoch이 성능변화도 3 이상이면 거의 일정하게 유지된 것을 볼 수가 있었다.



(그림 4) MNIST에서 epoch에 따른 딥 클러스터링의 정확도 결과

(그림 5)는 Fashion-MNIST에서 epoch에 따른 딥클러스터링의 정확도를 보여준다. 그림에서 보면, k가 10일 때 성능이 가장 좋은 것을 볼 수가 있다. Fashion-MNIST에서 각 클래스별 유사한 형태에 데이터가 있어서 데이터 특성상 정확도가 MNIST에 비해 떨어지는 것을 볼 수가 있었다.

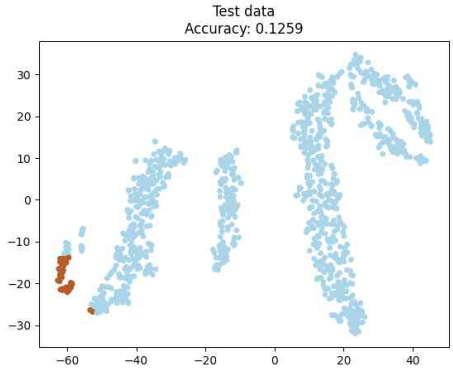


(그림 5) Fashion-MNIST에서 epoch에 따른 딥 클러스터링의 정확도 결과

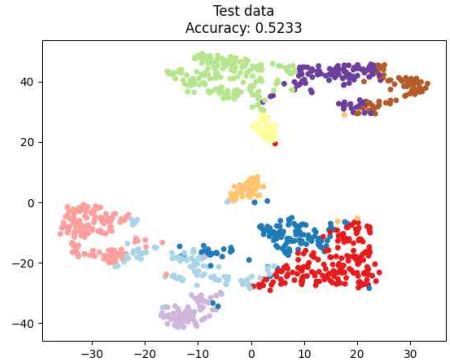
(그림 6)는 k가 10일 때, MNIST와 Fashion-MNIST에서 ReLU와 Sigmoid 활성화 함수를 적용하였을 때 정확도를 보여준다. ReLU 활성화 함수를 사용하였을 때 MNIST와 Fashion-MNIST 성능이 좋은 것을 볼 수가 있다. 또한, MNIST에 비해 상대적으로 Fashion-MNIST는 성능저하가 많이 발생하지 않았다.

(그림 6) k가 10일 때, MNIST와 Fashion-MNIST에서 활성화 함수를 ReLU와 Sigmoid를 적용 시 정확도 (k=10)

(그림 7)과 (그림 8)은 k가 10일 때, MNIST와 Fashion-MNIST에 대하여 활성화 함수 Sigmoid를 적용하였을 때 결과를 보여준다. 특히, 오토인코더를 통하여 잠재 벡터 모습이 MNIST에서는 잘 반영 안되는 것을 볼 수가 있다. 따라서 활성화함수는 ReLU 함수가 더 적절한 것을 볼 수가 있었다.



(그림 7) MNIST에서 활성화 함수를 Sigmoid를 적용한 시각화 결과 (k=10)



(그림 8) Fashion-MNIST에서 활성화 함수를 Sigmoid를 적용한 시각화 결과 (k=10)

4.4 분석결과

딥러닝 기반의 딥 클러스터링은 입력 데이터를 다른 벡터로 표현하여 군집의 성능을 개선 시킨 방법이다. 기존에는 차원 축소를 하거나 특정 특징만 추출하여 일부 정보를 인의적으로 제거하는 방식을 적용하였다. 하지만 딥러닝 기반으로 잠재 벡터로 압축이 가능하고 직교 행렬 등으로 클러스터의 구조를 고려한 벡터공간으로 표현함으로써 차원이 높은 데이터를 잘 군집화 하는 것을 실험적으로 볼 수가 있었다.

기존 연구와의 성능 비교측면에서, Deep Transformation-Invariant Clustering (DTI clustering) [20]으로 픽셀단위에서 직접적으로 변환하고 클러스터링을 실시하는 방법으로 2020년에 NIPS 발표

된 방법이다. 이 방법인 DTI clustering에서 Fashion-MNIST에서 61.2% 정확도와 SVHN에서 36.4% 정확도를 갖는다. 하지만 제안한 방법은 이에 비해 Fashion-MNIST에서 89.42% 정확도와 SVHN에서 56.64% 정확도로 성능이 향상된 것을 볼 수가 있다.

데이터셋 측면에서, MNIST에서 딥 클러스터링에 대한 성능이 좋은 것을 볼 수가 있었고 epoch의 수는 3이상일 경우 거의 유사한 특징을 볼 수가 있었다. 활성화 함수는 ReLU 함수가 적절하였으며 Sigmoid 함수 적용시 성능 저하가 생기는 것을 볼 수가 있었다. 반면에 Fashion-MNIST는 전체적으로 MNIST에 비해 성능이 저하된 것을 볼 수가 있었다. Fashion-MNIST 데이텃세에서 특정 클래스 간에 유사성이 높아서 비지도학습에서 성능에 영향을 미쳤으며 k가 10일 때 56.64% 정확도로 잘 유지하는 것을 볼 수가 있었다.

k 측면에서 적절한 k를 설정하는 것이 중요하였다. k개 선정은 실험적으로 적절한 k를 찾는 방법이 중요하였고 MNIST와 Fashion-MNIST 클래스가 10개이기 때문에 k가 10일 때 성능이 가장 좋은 것을 볼 수가 있었다.

군사적 활용 측면에서, 국방부에 있는 데이터 중에 이미지 데이터, 텍스트 데이터, 음성 데이터 등이 있지만 라벨링 작업이 없는 데이터들이 많이 있다. 아직 데이터센터 등에 구축이 되지 않아서 이러한 데이터에 대한 분석 등을 할 필요가 있다. 또한, 실시간 식별된 적 탐지하였을 때, 어떤 패턴 등이 있어 유사한 군집끼리 클러스터링 할 필요가 있다. 이러한 환경에서 딥 클러스터링을 적용하게 되면 각 데이터에 따라서 군집화하고 분류하는 데 도움이 될 것이라고 판단이 된다.

5. 결 론

이 논문에서 우리는 딥러닝 기반의 딥 클러스터링에 대한 성능분석을 하였다. 이 방법에서 컨볼루션 모델(Convolution neural network)기반의 오토인코더를 이용하여 입력 데이터를 잠재 벡터로 형성하고 이 잠재 벡터를 클러스터링하는 방법에 대해서 분석을 하였다. 실험결과로 k가 10일 때, MNIST에 대해서 89.42% 정확도를 가지고 Fashon-MNIST에 56.64% 정확도를 가졌다. 데이터의 라벨링이 없이 군집에 대해서 성능이 있는 것을 볼 수 있었다.

참고문헌

[1] Ezugwu, Absalom E., et al. "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects." Engineering Applications of Artificial Intelligence 110 (2022): 104743.

[2] Zada, Islam, et al. "Performance evaluation of simple K-mean and parallel K-mean clustering algorithms: big data business process management concept." Mobile Information Systems 2022 (2022): 1-15.

[3] Varshney, Ayush K., Pranab K. Muhuri, and QM Danish Lohani. "PIFHC: The probabilistic intuitionistic fuzzy hierarchical clustering algorithm." Applied Soft Computing 120 (2022): 108584.

[4] Shu, Hua, et al. "Density-based clustering for bivariate-flow data." International Journal of Geographical Information Science 36.9 (2022): 1809-1829.

[5] Myles, Anthony J., et al. "An introduction to decision tree modeling." Journal of Chemometrics: A Journal of the Chemometrics Society 18.6 (2004): 275-285.

[6] Podani, János, and Dénes Schmera. "On dendrogram based measures of functional diversity." Oikos 115.1 (2006): 179-185.

[7] Cunningham, Pádraig. "Dimension reduction." Machine learning techniques for multimedia: Case studies on organization and retrieval (2008): 91-112.

[8] Bank, Dor, Noam Koenigstein, and Raja Giryes. "Autoencoders." arXiv preprint arXiv:

2003.05991 (2020).

[9] Yamashita, Rikiya, et al. "Convolutional neural networks: an overview and application in radiology." Insights into imaging 9 (2018): 611-629.

[10] Liu, Jie, and Chenghua Fu. "MNIST data set recognition research based on TensorFlow framework." International Core Journal of Engineering 7.2 (2021): 410-414.

[11] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[12] Bengio Yoshua, Courville Aaron, and Vincent Pascal. Representation learning: a review and new perspectives. TPAMI, 35(8):1798 - 1828, 2013.

[13] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504 - 507, 2006.

[14] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In AISTATS, volume 2005, pages 57 - 64. Citeseer, 2005.

[15] Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clusieee t neur net lear. In ICML, pages 129 - 136, 2009.

[16] Yazhou Ren, Shudong Huang, Peng Zhao, Minghao Han, and Zenglin Xu. Self-paced and auto-weighted multi-view clustering. Neurocomputing, 383:248 - 256, 2019.

[17] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. KDE, 22(10):1345 - 1359, 2010.

[18] Guo, Wengang, Kaiyan Lin, and Wei Ye. "Deep embedded K-means clustering." 2021 International Conference on Data Mining Workshops (ICDMW). IEEE, 2021.

[19] Imambi, Sagar, Kolla Bhanu Prakash, and G. R. Kanagachidambaresan. "PyTorch." Programming with TensorFlow: Solution for Edge Computing Applications (2021): 87-104.

[20] Monnier, Tom, Thibault Groueix, and Mathieu Aubry. "Deep transformation-invariant clustering." Advances in Neural Information Processing Systems 33 (2020): 7945-7955.

〔저자소개〕

권 현 (Hyun Kwon)
2010년 2월 육군사관학교 이학사
2015년 8월 KAIST 전산학부 공학석사
2020년 2월 KAIST 전산학부 공학박사
email : hkwon.cs@gmail.com

이 준 (Jum Lee)
2004년 2월 건국대학교 컴퓨터공학부 공학사
2006년 2월 건국대학교 컴퓨터정보통신공학과 공학석사
2012년 2월 건국대학교 신기술융합학과 공학박사
email : jun.lee.mistra@gmail.com