데이터스트림에서 Exponential Histogram을 사용한 개념 변화 검출 기법

김만수*, 임효상
연세대학교 원주캠퍼스 컴퓨터정보통신공학부
e-mail:{mansookim, hyosang}@yonsei.ac.kr

A Method for Detecting Concept Drift in Data Stream by Using Exponential Histogram

Man-Soo Kim* and Hyo-Sang Lim
Computer and Telecommunications Engineering Division,
Yonsei University, Wonju

본 논문은 Exponential histogram을 사용하여 데이터스트림에서 개념 변화를 검출 하는 기법을 제안한다. 스트림 데이터와 같이 빠르게 증가하는 데이터에 대한 개념 변화를 찾는 것은 중요 문제이다. 기존에 사용하던 슬라이딩 윈도우 기반의 방법들은 과거의 데이터를 버렸지만, 제안하는 방법은 과거의 데이터를 효율적으로 저장하며, 윈도우의 크기를 변경 할 수 있는 방법을 제안한다. 실험을 통해 제안하는 방법에 대한 효율성과 정확성을 보인다.

1. 서론

최근 디지털 기기의 보급과 소셜 네트워크, 온라인 비즈니스 등의 영향으로 많은 양의 스트림 데이터가 생성되고 있다[1]. 스트림 데이터란 시간이 경과함에 따라 지속적으로 생성되는 데이터들의 시퀀스를 뜻한다. 스트림 데이터는 지속적으로 생성되는 대용량 형태를 띠기 때문에, 데이터를 빠르고 정확하게 분류하는 것이 중요하다. 하지만 스트림 데이터가 가지고 있는 고유한 특징 때문에 이는 쉽지 않은 문제이다[2]. 그 고유한 특징으로 개념 변화, 개념 진화, 지연된 라벨링 등이 있다.

본 논문에서는 위에서 언급한 특징 중 개념 변화를 검출하는 문제를 다룬다. 개념 변화란 지금까지 들어오던 데이터와 다른 개념(concept)을 가지는 데이터가 들어오는 것을 뜻한다. 여기서의 개념은 데이터의 분포, 통계 등이 될 수 있다. 본 논문은 개념 변화를 검출하는 방법을 제안한다.

기존 데이터 스트림에서는 질의 처리를 위해 슬라이딩 윈도우 방법을 사용한다. 슬라이딩 윈도우란 일정한 크기의 윈도우를 입력 데이터에 따라서 연속적으로 슬라이딩 시키는 것을 의미한다. 기존의 개념 변화 검출 방법에서는 이 슬라이딩 윈도우 방법을 개념 변화 검출에 적용하여 사용한다. 하지만 이 방법은 슬라이딩 윈도우 특성상 과거 데이터는 모두 버려진다는 문제가 발생한다. 과거 데이터를 보관하더라도 스트림 데이터의 양이 매우 많아 큰 메모리가 필요하다. 따라서 본 논문에서는 이를 해결하기 위한 방법으로 Exponential histogram[3]을 사용한 개념 변화 검출 기법을 제안한다. Exponential histogram은 오래된 데이터들은 병합(merge)하여 대략적인 모습을 유지하고, 상대적으로 최근 데이터는 시간에 따라 정확한 값을 저장하는 근사 기술이다. Exponential histogram을 통해 과거 데이터를 유지하면서 최근 데이터를 비교하는 개념 변화 검출이 가능하다. 또한 과거 데이터를 보관하기 때문에 슬라이딩 윈도우의 크기를 유동적으로 변화시켜 다양한 분석이 가능하다는 장점이 존재한다.

본 논문의 구성은 다음과 같다. 제2장에서는 관련 연구를 살펴본다. 제3장에서는 본 논문에서 제안하는 방법을 설명한다. 제4장에서는 제안하는 방법에 대한 실험을 통해 유용함을 입증하고, 제5장에서 결론을 맺는다.

2. 관련 연구

2.1 개념 변화 관련 연구

개념 변화는 독립적인 확률 분포 p_i로부터 생성된 데이터 s_i들의 시퀀스 s가 존재 할 때, $p_i \neq p_{i+1}$이면 개념 변화가 발생한 것으로 판단한다. 이 때, $i+1$ 시점을 개념 변화가 발생한 시점이라 한다[4]. 기존의 개념 변화 검출 기법으로 윈도우를 사용한 개념 변화 탐지 알고리즘이 존재한다. 데이터 스트림 처리에 있어, 모든 지난 데이터들을 메모리에 저장하는 것은 비현실적이기 때문에, 두 개의 윈도우를 사용하여 변화를 탐지한다. 참고 윈도우(reference window)는 기존 데이터에 대한 윈도우이고, 현

재 윈도우(current window)는 새로 들어오는 데이터를 따라 슬라이딩 하는 윈도우를 의미한다[5]. 참고 윈도우는 개념 변화가 발생 된 경우 현재 윈도우의 위치로 업데이트하여 새로운 개념 변화 탐지를 수행하게 된다.

2.2 Exponential Histogram 관련 연구

Exponential histogram은 0과 1이 입력되는 데이터스트림 환경에서 최근 T초 동안 입력 된 1의 개수를 근사화해서 저장하는 데이터 구조이다[3]. 오래된 데이터 일수록 시간이 함께 병합되어 시간에 따라 대략적인 값을 나타내고, 최근 데이터 일수록 시간에 따라 명확한 값을 나타낸다. 이를 통해 메모리를 절약하며 효율적으로 데이터를 저장 할 수 있다. Exponential histogram은 입력된 0 또는 1의 데이터 중, 1의 데이터의 일정량을 모아서 버킷의 형태로 유지하고 0은 버린다. 버킷은 Exponential histogram을 유지하는 단위이고 버킷의 크기와 타임스탬프로 구성된다. 여기서 버킷의 크기는 해당 타임스탬프에 저장된 1의 개수가 된다. 타임스탬프는 해당 버킷에 포함된 데이터 중 가장 최근에 들어온 데이터의 입력 시간으로 설정된다. 아래 표 1은 시간t_1부터 t_3까지 연속적으로 1이 입력되었을 때, Exponential histogram의 상태를 나타낸다.

버킷의 크기	1	1	1
타임스탬프	t_1	t_2	t_3

표 1. t_3초 일 때의 Exponential histogram

표 1에서 볼 수 있듯이, 입력 데이터로 1이 연속적으로 들어오게 되면 Exponential histogram이 생성된다. 하나의 데이터가 입력 될 때마다 갱신이 일어난다. 이 때 입력되는 데이터는 1만 가능하다. 갱신 될 때 버킷이 병합이 되는데 이 조건은 사전에 주어진다. 동일한 데이터의 수가 병합조건과 같아지면 해당 데이터 중 가장 오래된 데이터가 병합되고, 이 때 타임스탬프는 병합 되는 데이터 중 최근 시간으로 정해진다. 아래 표 2는 병합조건이 4이고, 표 1의 상황에서 시간이 지나 시간t_4일 때 데이터 1이 들어와 병합과정 거친 결과를 보여준다. 이와 같은 과정을 거쳐서 Exponential histogram에는 입력 된 1이 모두 저장이 된다. 과거의 데이터는 병합 된 상태로 근사 된 데이터가 저장되고, 최근 데이터는 타임스탬프에 따라 정확하게 저장이 되게 된다.

버킷의 크기	2	1	1
타임스탬프	t_2	t_3	t_4

표 2. t_4초 일 때의 Exponential histogram

3. 제안하는 방법

3.1 슬라이딩 윈도우에 따른 Exponential histogram 구성

입력 데이터에 대한 개념 변화를 검출하기 위해 슬라이딩 윈도우와 Exponential histogram을 사용한다. Exponential histogram을 사용하기 때문에 입력되는 데이터는 0 또는 1이다. 두 개의 슬라이딩 되는 윈도우의 크기만큼 Exponential histogram에서 값을 가져와 비교하여 개념 변화를 탐지한다. 즉, 제안하는 방법에서의 개념 변화는 입력되는 데이터에서 0과 1의 비율인 분포가 급격이 바뀌는 순간이 개념 변화가 되었다고 판단 한다.

최초 데이터가 입력됨에 따라 첫 번째 윈도우인 w_1이 슬라이딩 된다. 입력 된 데이터가 0일 경우에는 데이터를 버리고, 1일 경우에는 w_1에 대한 Exponential histogram인 E_{w_1}를 생성한다. 데이터가 새롭게 입력되면 w_1은 이와 같은 과정을 반복하며 시간에 따라 일정하게 슬라이딩 된다. 윈도우가 슬라이딩 됨과 동시에 E_{w_1}도 지속적으로 병합조건에 따라 갱신된다.

사전에 두 윈도우 사이의 시간 간격을 정의하고 w_1가 슬라이딩 되는 과정 중에 사전에 정의 된 시간 간격만큼 진행이 되었을 때 두 번째 윈도우인 w_2가 생성된다. 이 때 시간 간격이 윈도우의 크기보다 크거나 같으면 두 윈도우는 입력 데이터에 대해 중복된 구간을 슬라이딩 하지 않게(disjont) 된다. 만약 시간 간격이 윈도우의 크기보다 작다면 두 윈도우는 일정부분 겹쳐진 상태로 진행하게 된다. 이는 제안하는 방법에서 결과를 바꾸는데 큰 영향력을 끼친다. 따라서 시간간격과 윈도우의 사이즈는 서로 연관성을 가진다. 아래 그림 1은 스트림 데이터가 들어옴에 따라 w_1, w_2가 각각 슬라이딩 되는 것을 보여준다. 가장 위의 D는 시간에 따라 데이터가 들어오는 시퀀스를 의미하고, 아래의 두 개의 윈도우는 동일한 간격을 유지하며 슬라이딩 되는 것을 보여준다. 그림 1에서의 시간 간격은 윈도우 사이즈보다 작은 것을 보인다.

w_2 또한 w_1와 동일하게 슬라이딩 되며, w_2에 대한 E_{w_2}를 생성한다. w_2이 슬라이딩 되며 E_{w_1}와 동일한 방법으로 E_{w_2}은 갱신 된다. 즉, w_1와 w_2는 독립적으로 입력 데이터에 대해 Exponential histogram을 갱신해 나간다.

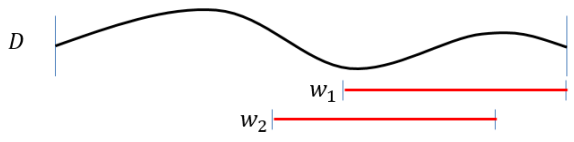


그림 1. 입력 된 데이터에 대한 슬라이딩 윈도우

3.2 개념 변화 검출 과정

개념 변화 검출을 위해 w_1, w_2에 의해 생성된 E_{w_1}, E_{w_2}을 사용한다. 구성된 두 개의 Exponential histogram에서 윈도우 크기만큼의 값을 가져와 두 값의 차이를 비교한다. 값을 가져 올 때에는 Exponential histogram의 가장 최근 타임스탬프에 저장 된 버킷 값부터 윈도우 사이즈의 크기만큼 지나간 과거 타임스탬프 사이의 모든 버킷 값을 더한다. 이 때, Exponential histogram의 근사화 특성상, 과거 저장된 버킷은 비교에 필요한 타임스탬프에 정확히 매핑이 안 될 수도 있다. 이때, Exponential histogram의 특성인 절대오류[6]를 이용한 추정 값을 사용하여 해당 타임스탬프 버킷의 크기에 1/2를 곱한 후 사용하게 된다.

E_{w_1}과 E_{w_2}에서 가져온 값들 간의 비교는 데이터가 들어온 후 각각의 Exponential histogram의 갱신이 끝나고 진행된다. Exponential histogram에 저장되는 값은 1의 개수이기 때문에 개념 변화를 확인하는 방법(measure)은 1에 대한 집계 값의 차이이다. 집계 값의 차이를 비교하는 방법으로는 각각의 Exponential histogram에서 얻은 값의 차이를 구해서 그 값이 사전에 정의된 임계값을 넘는지를 확인한다. 임계값은 개념 변화가 일어났는지를 판단하는 기준으로, 구한 값이 임계값을 넘으면 개념 변화가 일어났다고 판단을 한다. 제안하는 방법에서는 윈도우 크기에 대한 비율로 임계값을 설정하고 표기법으로 Θ을 사용한다. 개념 변화가 일어났을 때, 변화 후의 데이터 분포가 이전 데이터의 분포보다 높을 수도 있고 적을 수도 있기 때문에 차이 값을 절대 값으로 처리하여 계산한다. 데이터가 들어오는 순간마다 위의 검색과 비교 과정을 반복하여 연속적으로 개념 변화 구간을 찾게 된다.

4. 실험

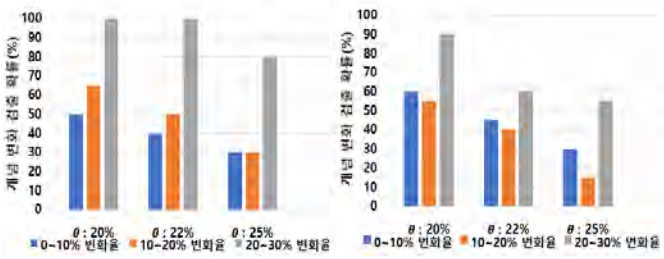
4.1 실험 데이터 및 환경

실험은 입력 데이터(0 또는 1)에 대한 개념 변화를 확인한다. 데이터 셋은 십 만개의 1과 0으로 이루어진 세 개의 데이터 셋으로 구성된다. 데이터 셋은 각각 ±0~10%, ±10~20%, ±20~30%의 분포 변화율을 가진다. 분포의 변화율이란 개념 변화가 발생하기 전 분포에서 발생 한 후의 분포로 변화할 때, 변화하는 정도를 뜻한다. 예를 들어, 최초에 데이터 0에 대한 1의 발생 비율이 50%인 분포로 시작한다고 가정한다. 이 때 분포의 변화율이 ±0~10%일 경우, 개념 변화가 발생 한 후의 분포는 40~60% 사이의 비율을 가질 수 있게 된다. 모든 데이터 셋은 20번 개념 변화가 발생하도록 생성한다. Exponential histogram의 병합 조건은 4로 설정하고, 2개의 윈도우 사이의 간격은 1000으로 설정한다. 개념 변화를 검출하는 Θ 값은 윈도우 사이즈 대비 20%, 22%, 25%로 바꿔 가며 실험을 진행한다.

4.2 실험 결과

실험은 데이터 셋에 있는 데이터가 1초에 1개씩 들어오도록 하고, 윈도우의 크기를 1000과 1500 일 때로 나누어서 수행한다. 첫 번째 실험은 개념 변화 검출 확률을 확인하는 실험이다. 개념 변화 검출 확률은 제안하는 방법이 입력 된 데이터의 분포가 변하는 부분을 얼마나 잘 찾아내는지를 나타낸다. 모든 데이터 셋은 20번의 분포 변화를 주었고 찾는 개수에 따라서 확률로 나타낸다. 두 번째 실험은 착오 해답(false alarm) 비율을 확인하는 실험이다. 여기서의 착오 해답이란 개념 변화가 일어나지 않았는데 일어났다고 잘못 판단하는 경우를 뜻한다. 즉, 두 개의 윈도우가 같은 분포를 가지는 구간에 존재하는데 개념 변화가 일어났다고 판단하는 순간이다. 이러한 경우에 대한 발생 횟수를 총 데이터의 개수인 십만 번 중 발생하는 횟수로 나눠서 비율을 구한다. 세 번째 실험은 실제 입력 데이터에서 개념 변화가 일어났을 때, 개념 변화를 최초 탐색하는 순간까지의 걸리는 시간을 알아보는 실험이다. 결과 값은 나온 값에 대한 평균으로 해당 값을 판단한다.

그림 2는 첫 번째 실험인 개념 변화 검출 확률에 대한 결과를 보여준다. 윈도우 크기가 1000일 경우, 0~10% 변화률과 10~20% 변화율에서 비슷한 수준의 검출률을 보인다. Θ 값이 커질수록 검출량이 줄어들기 때문에 확률이 동일해 진다. 반면 분포의 변화량이 큰 20~30%의 변화율에서는 Θ 값이 증가하여도 높은 검출량을 보인다. 분포의 변화량이 클수록 데이터의 값이 급속도로 바뀌기 때문이다. 윈도우의 크기가 1500으로 커질 경우, 윈도우의 크기가 1000일 때 보다 전반적으로 검출량이 줄어든다. 윈도우의 크기가 커질수록 윈도우가 중복되는 부분이 증가하기 때문에 검출 확률이 줄어든다.

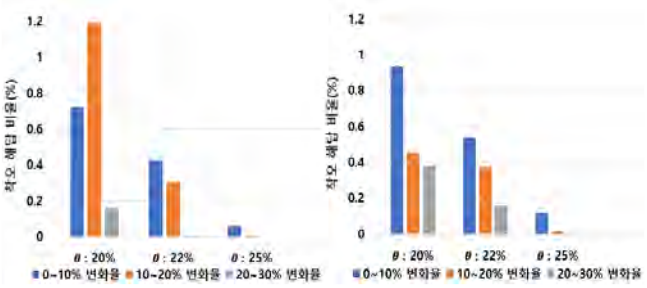


(a) 윈도우 크기=1000 (b) 윈도우 크기=1500

그림 2. 개념 변화 검출 확률

그림 3은 착오 해답 비율을 보여준다. 윈도우 크기가 1000일 경우, Θ 값이 커질수록 착오 해답 비율이 줄어든다. 이는 Θ 값이 증가되면서 검출량 자체가 줄어들기 때문에 잘못 검출되는 비율도 줄어든다. 높은 변화율을 가지는 20~30% 변화율의 경우, 잘못 검출되는 경우가 0%로 확인된다. 착오 해답의 경우 Θ 값의 변화가 다른 조건보다 큰 영향을 준다. 윈도우 크기가 1500일 경우에도, 윈도우 크기가 1000일 때와 비슷한 경향을 보인다. Θ 값이 작을 때에는 낮은 변화율 구간에서 윈도우 크기에 대해 큰

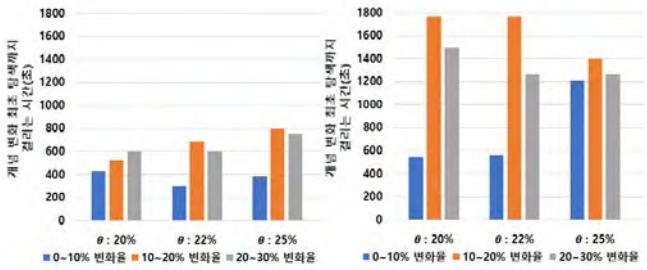
변화를 보였지만, Θ 값이 클 때에는 윈도우 크기에 큰 영향을 받지 않는다.



(a) 윈도우 크기=1000 (b) 윈도우 크기=1500

그림 3. 착오 해답 비율

그림 4는 개념 변화가 발생 했을 때, 그 부분을 최초로 탐색하기까지의 걸리는 시간을 나타낸다. 윈도우 크기가 1000일 경우, Θ 값이 증가함에 따라서 탐색까지 걸리는 시간이 증가하는 경향을 보인다. 윈도우 크기가 1500일 경우에는 윈도우 크기가 1000 경우와 비교하여 현저히 상승한다. 이는 개념 변화를 검출 할 확률이 줄어듦에 따라 연쇄적으로 시간도 증가하는 것으로 판단된다.



(a) 윈도우 크기=1000 (b) 윈도우 크기=1500

그림 4. 개념 변화 최초 탐색까지 걸리는 시간

5. 결론

본 논문에서는 데이터스트림에서 Exponential histogram을 이용하여 개념 변화를 검출하는 방법을 제안하였다. 먼저 슬라이딩 윈도우에 대해 Exponential histogram을 구성하였고, 이 값들을 이용해 개념 변화를 확인하였다. 각각의 슬라이딩 윈도우에 대한 Exponential histogram을 보유함으로서 윈도우의 크기를 동적으로 변경시킬 수 있다는 장점이 있다. 향후 연구로는 Exponential histogram을 생성하는 시간을 줄이는 효율적인 방법을 연구하고, 변경된 분포의 지속시간도 고려하는 방법 등을 연구할 예정이다.

참고문헌

[1] A. Haque, L. Khan, M. Baron, B. Thuraisingham and C. Aggarwal, "Efficient handling of concept drift and concept evolution over Stream Data," 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, pp.481-492, 2016.

[2] A. Haque, L. Khan, and M. Baron, "Semi supervised adaptive framework for classifying evolving data stream", In Advances in Knowledge Discovery and Data Mining, volume 9078 of Lecture Notes in Computer Science, Springer International Publishing, pp.383-394.

[3] Mayur Datar, Aristides Gionis, Piotr Indyk, Rajeev Motwani, "Maintaining stream statistics over sliding windows", Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, pp.635-644.

[4] Indr˙e Zliobait˙e, "Learning under Concept Drift: an Overview", Technical report, Faculty of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania, 2009.

[5] Kifer, Daniel, Shai Ben-David, and Johannes Gehrke. "Detecting change in data streams.", In Very Large Databases(VLDB), 2004.

[6]최영환, "데이터스트림에서의 효율적인 임의 시간구간 기본 집계 질의 처리", 연세대학교 대학원 전산학과 석사 학위논문, 2014.