5월13일~5월19일

∙ 연구에 있어서 Principal Components Analysis에 대한 깊은 내용과 이해가 필요하여 집중적으로 공부 하였습니다. 지난주 연구에 이어 연구하였습니다.

**Step 5: Choosing components and forming a feature vector**

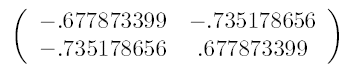
여기에 데이터 압축 및 축소된 차원의 개념이 들어있다. 이전 섹션에서 고유벡터와 고유값을 보면 고유값이 상당히 다른 값이라는 것을 알 수 있다. 사실, 고유치가 가장 높은 고유벡터가 데이터 세트의 주요 구성 요소인 것으로 나타났다. 우리의 예제에서, 큰 고유값을 갖는 고유벡터는 데이터의 중앙을 가리키는 고유벡터 였다. 이것은 데이터 차원 간의 가장 중요한 관계이다.

일반적으로, 일단 공분산 행렬로부터 고유벡터가 발견된다면 다음 단계는 가장 높은 값부터 가장 낮은 값까지 고유값에 의해 그들을 정렬하는 것이다. 이것은 중요한 순서대로 구성 요소를 제공한다. 우리는 덜 중요한 요소를 무시할 수 있다. 약간의 정보를 잃어 버리지만, 고유치가 작으면 많은 것을 잃지 않는다. 그래서 일부 구성 요소를 생략하면 최종 데이터 세트의 크기가 원본보다 작아진다. 정확히 말하면 데이터의 원래 n차원이 있고 n고유벡터와 고유값을 계산 한 다음 첫번째 고유벡터만 선택하면 최종 데이터 세트에는 p차원만 갖는다.

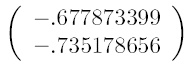
지금 벡터의 행렬에 대한 feature vector를 형성해야 한다. 이는 고유벡터 목록에서 유지하려는 고유벡터를 가져와서 행렬을 생성한다.



데이터 세트와 2개의 고유벡터가 있다는 사실을 감안할 때 두가지 선택이 있다. 우리는 두 고유벡터를 가진 특징 벡터를 만들 수 있다.



또는 더 작고 덜 중요한 구성요소를 제외하고 단일 열만 선택할 수도 있다.



**Step 5: Deriving the new data set**

이것은 PCA의 마지막 단계이며, 가장 쉬운 방법이기도 하다. 우리가 데이터에 보관하고자 하는 구성요서(고유벡터)를 선택하고 feature vector를 형성하여 벡터의 치환기를 가져와서 원래 데이터 세트의 왼쪽에 곱한다.



여기서 RowFeatureVector는 고유벡터가 행에 있고 상단에서 가장 중요한 고유벡터를 가지도록 전치된 행에 고유벡터가 있는 행렬이며, RowDataAdjust는 조정된 평균 데이터이다. 즉, 데이터 항목은 각 열에 있으며 각 행에는 별도의 차원이 있다. FinalData는 열에 데이터 항목이 있고 행에 차원이있는 최종 데이터의 집합이다.

이것이 우리에게 무엇을 줄까? 우리가 선택한 벡터의 관점에서만 원래의 데이터를 제공한다. 우리의 원래 데이터 세트에는 두 축 x와 y가 있었기 때문에 우리의 데이터는 그 축에 있었다. 원하는 두 축의 관점에서 데이터를 표현할 수 있다. 이 축이 수직이면 표현이 가장 효율적이다. 이것이 고유 벡터가 항상 서로 수직 인 것이 중요했던 이유이다. 여기서는 데이터를 축 x와 y의 관점에서 바꾸었고, 이제는 2 개의 고유 벡터에 관한 것이다. 새 데이터 세트의 차원이 감소된 경우 즉, 고유 벡터의 일부를 남겼다. 새로운 데이터는 우리가 유지하기로 결정한 벡터의 관점에서만 이다. 여기서 데이터에 각 feature vector를 사용하여 최종 변환을 수행했다. 결과를 좋은 표와 같은 형식으로 되돌리기 위해 결과의 전치를 가져 왔고 또한 최종 점을 구성 요소와 관련시키는 방법을 보여주기 위해 플로팅 되었다. 변환을 위해 두 개의 고유 벡터를 유지하는 경우 Figure 3.3에서 볼 수있는 데이터와 플롯을 얻었다. 이 플롯은 근본적으로 고유 데이터이며, 고유 벡터가 축이되도록 회전된다. 이것은 분해에서 정보를 잃어 버렸기 때문에 이해할 수 있다.

 여기서 만들 수 있는 다른 변환은 가장 큰 고유치를 갖는 고유 벡터만을 취하는 것이다. 그 결과의 데이터 표는 그림 3.4에 나와 있다. 예상대로 단일 차원만 있다. 이 데이터 세트를 두 개의 고유 벡터를 사용하여 생성 된 데이터 세트와 비교하면 이 데이터 세트가 다른 데이터 세트의 첫 번째 열에 있다는 것을 알 수 있다. 따라서 이 데이터를 플롯 하려면 1 차원이어야 하며 그림 3.3의 플롯에 있는 점의 x 위치와 정확히 일치해야 한다. 여기서는 다른 고유 벡터인 다른 모든 축을 효과적으로 버렸다.

그럼 여기서 뭘 했을까? 기본적으로 데이터를 변환하여 패턴 사이의 패턴으로 표현한다. 패턴은 데이터 간의 관계를 가장 잘 묘사하는 선이다. 이는 데이터 포인트를 각 라인의 기여도의 조합으로 분류했기 때문에 유용하다. 처음에 간단한 x와 y 축을 가졌다. 그리고 각 데이터 요소의 x 및 y 값은 실제로 그 점이 나머지 데이터와 어떤 관련이 있는지 정확하게 알려주지 않았다. 이제 데이터 요소의 값은 데이터 요소가 위치한 추세선의 위치 (즉 above / below)를 정확히 알려준다. 두 고유 벡터를 사용하는 변환의 경우, 여기서는 단순히 일반적인 축 대신에 고유 벡터의 관점에서 데이터를 변경했다. 그러나 단일 고유 벡터 분해는 더 작은 고유 벡터로 인한 기여를 제거하고 다른 하나의 관점에서만 데이터를 남겨 둔다.

