ToBigs Curriculum

ToBigs 6기 장재석

Linear Discriminant Analysis

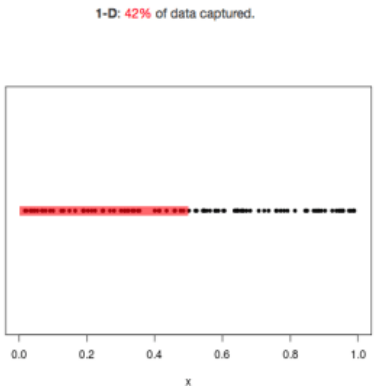
선형 판별 분석을 활용한
지도적 데이터 압축

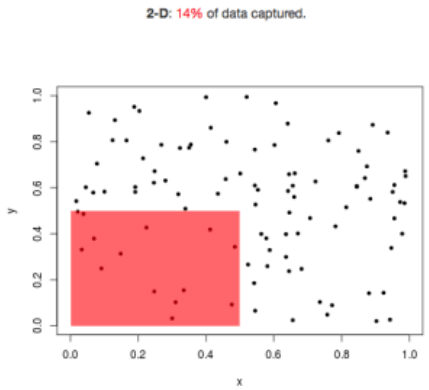
Contents

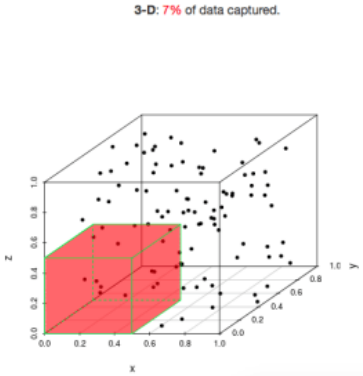
Unit 01 | 차원축소란?

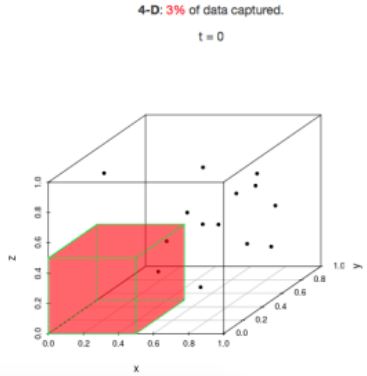
차원 축소, 차원의 저주에 관하여

차원이 증가하면 그것을 표현하기 위한 데이터 양이 기하급수적으로 증가한다.









왜 Feature가 필요할까?

머신러닝은 입력 데이터를 출력데이터로 대응시켜주는 블랙박스라고 생각해보자.
우리는 훈련데이터를 사용해서, 입력데이터의 함수인 블랙박스를 학습시키지만
항상 잘 학습되는 것은 아니다.

머신러닝의 성능은 데이터의 양과 질에 굉장히 의존적인 성질을 갖기 때문!

왜 Feature가 필요할까?

가장 이상적인 입력 데이터는 적당한 크기의 정확한 정보만 포함하기를 원한다.
그러나 풀고자 하는 문제에 대한 완벽한 사전지식을 갖기 어렵기 때문에,
머신러닝 기법을 적용하여 문제를 풀고자 한다.

데이터를 수집 한 후 어떤 feature가 유용한지 아닌지 확인이 필요!

Feature selection & Feature extraction

차원축소란 무엇인가?

고차원의 충분한 입력값을 갖고 있을 때, 모든 입력값의 feature들이 불필요할 수 있다.
따라서 관찰 대상들을 잘 설명할 수 있는 **잠재공간(latent space)**은
실제 관찰되어지는 공간(observation space) 보다 작을 수 있다.

즉 관찰 공간 위의 샘플들에 기반으로 잠재 공간을 파악하는 것을
차원축소(dimensionality reduction)라고 한다.

특징 선택 vs 특징 추출

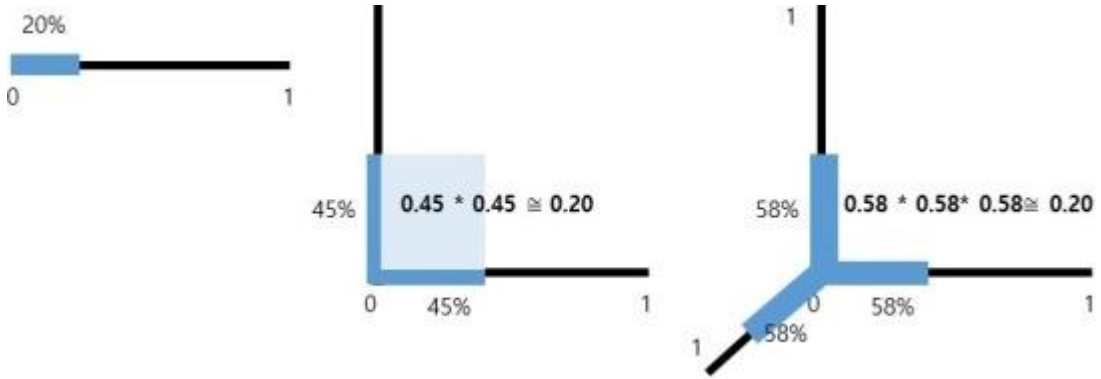
데이터의 차원을 줄이는 데는 **특징 선택(feature selection)**과 **특징 추출(feature extraction)**이 있다.

1. **특징 선택** : 모든 특징의 부분 집합을 선택해서 간결한 특징 집합을 만드는 것
원본 데이터에서 불필요한 특징들(변수들)을 제거한다.

2. **특징 추출** : 원본 데이터의 특징들의 조합으로 새로운 특징을 생성하려고 시도.
예) PCA : 데이터로 부터 직교 주축을 찾고 모든 데이터를 해당 축에 투영시켜, 원본 특징들의 선형결합으로 이루어진 새로운 특징을 만든다.

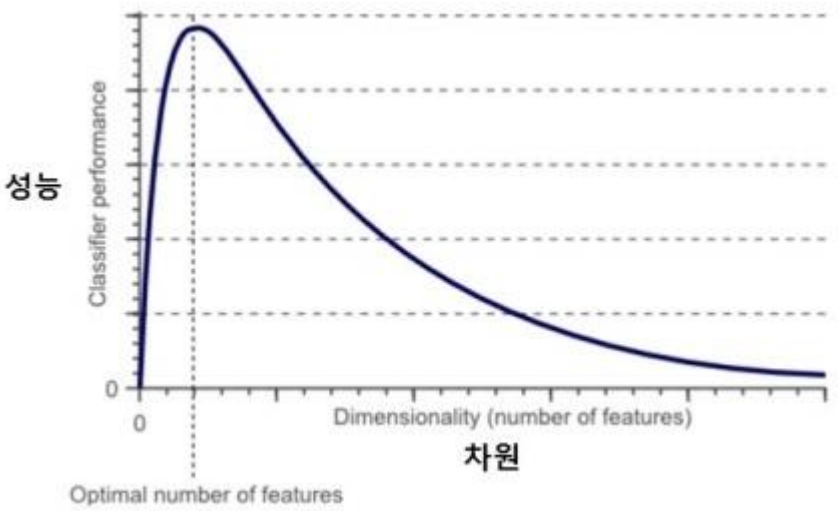
Unit 01 | 차원축소란?

차원의 저주



데이터의 차원이 증가하게 되면,
표현하는 데이터의 양은
지수적으로 증가하게 된다.
이는 계산비용이 기하급수적으로
증가한다는 것 뿐만 아니라,
데이터에서 정말 필요한
정보의 밀도가 감소한다는 뜻이다.

차원의 저주



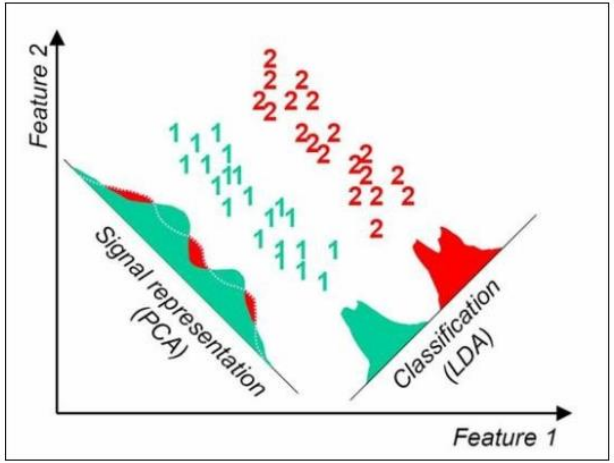
- 차원이 늘어나면 데이터가 공간을 설명하기 부족하게 된다.
- 적은 데이터로 공간을 설명해야 하기 때문에 **과적합**의 문제가 발생
- 모델의 성능이 떨어지므로 일반화가 안 된다.

⇒ **차원의 저주를 해결하기 위해서 모델을 regularization, dimensionality reduction, feature selection**

PCA의 단점

- PCA는 단순히 변환된 축이 최대 분산 방향과 정렬되도록 좌표회전을 수행
- 특징벡터의 클래스 라벨을 고려하지 않기 때문에, 클래스들의 구분성은 고려하지 않는다.
- 따라서 최대 분산방향이 특징 구분을 좋게 한다는 보장은 없다.
- PCA는 비선형인 패턴에는 적용하기 곤란하다.

LDA란?



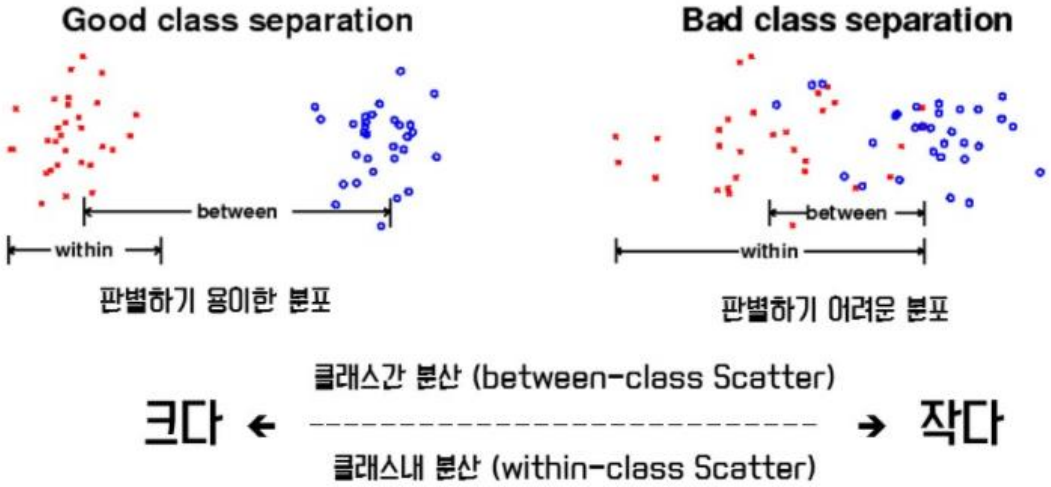
선형판별분석은 PCA와 마찬가지로 피처 압축 기법이다.
전체적인 개념은 상당히 유사하지만, LDA는 PCA와 달리
최대분산의 수직을 찾는 것이 아니라 지도적 방식으로
데이터의 분포를 학습하여 분리를 최적화하는
피처부분공간을 찾은 뒤, 학습된 결정경계에 따라
데이터를 분류하는 것이 목표이다.

LDA란?

즉 PCA가 데이터의 전체적인 분포를 참고하여 새로운 basis를 설정하고, 그 축에 맞게 데이터를 새롭게 projection하는 것이 목표라면, LDA는 지도적인 방법으로 basis를 찾아서 그 축을 분리에 이용한 뒤, 최적의 분리를 완성시킨 후에 projection하는 것이 목표

⇒ 일반적인 상황에서는 PCA보다는 LDA가 성능이 좋은 것으로 알려져 있다.
⇒ LDA의 기본적인 방법은 class들의 mean 값들의 차이는 최대화하고,
classLH의 variance는 최소화하는 벡터 $\vec{w}$ w 를 찾는 것이다

LDA란?



-특징 공간 상에서 클래스 분리를
최대화하는 주축으로 사상(projection)
선형공간으로 차원을 축소하는 방법

-클래스간 분산(between-class)과
클래스내 분산(within-class)의 비율을
최대화하는 방식으로 데이터에 대한
특징 벡터의 차원을 축소

LDA에 대한 수식적 접근

P차원의 입력벡터 x를 w라는 벡터(축)에 projection 한 후 생성되는 1차원의 좌표값을 아래와 같이 y라고 정의한다.
각각 N1, N2개의 관측치를 갖는 C1과 C2 두 범주에 대해서 원래 입력공간에서 각 범주의 중심 벡터를 m1, m2라고 하자.

$$y = \vec{w}^T \vec{x}$$

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n$$

$$m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

$$m_2 - m_1 = w^T (m_2 - m_1)$$

$$m_k = w^T m_k$$

Projection후에 두 범주의 중심이 멀리 떨어져 위치하는 벡터 w를 찾아야 한다. 관계식은 다음과 같다.

LDA에 대한 수식적 접근

Projection 이후에 각 범주에 속한 관측치들은 해당 범주 중심에 가까이 있을 수록 좋기 때문에 분산을 작게 해야 한다. 분산의 식은 아래의 s_k^2 이며, LDA가 데이터를 잘 분류하기 위해서는 두 범주의 중심은 최대화하고, 분산을 최소화해야 한다. 따라서 이를 목적함수 J(w)로 나타내면 다음과 같이 나타 낼 수 있다.

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_W w}$$

S_w : within − class scatter
S_B : $between - class\ scatter$

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

LDA에 대한 수식적 접근

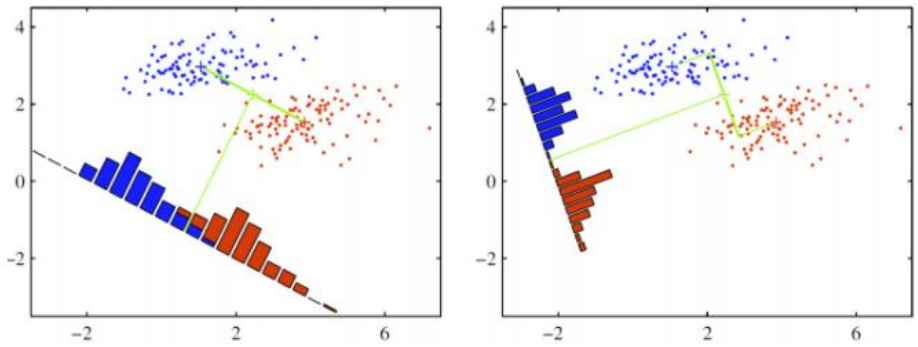
최종적으로 앞의 장에서의 목적함수 J(w)는 w에 대해 미분한 값이 0이 되는 지점에서 최대값을 갖고 아래 식과 같이 나온다.

$$(w^T S_B w) S_W w = (w^T S_W w) S_B w \longrightarrow \begin{matrix} S_W w = \lambda S_B w \\ {S_B}^{-1} S_W w = \lambda w \end{matrix}$$

괄호 안은 스칼라임으로
다음과 같은 식으로 변환

위의 식은 최종적으로 $AX = \lambda X$ 의 형태이므로 새로운 축 w는 S_B의 역행렬과 S_W를 내적한 행렬의 고유벡터!
이 행렬을 고유값 분해하여 새로운 축 w를 구할 수 있다.

⇒ 새로운 데이터가 들어왔을 때 이를 w와 내적해 각각의 스코어를 낼 수 있으며, 그 스코어가 일정 값보다 크면 C1 범주, 작으면 C2 점주로 분류하게 된다.



오른쪽 그림과 같이 분류가 잘 되는 쪽으로
정사영 시키고 싶다 => Fisher의 해법

Fisher's LDA

projection 시킨 데이터들에서
같은 클래스에 속하는 데이터들의 variance는 최대한 줄이고 ($\sigma_{within}\sigma_{within}$),
각 데이터들의 평균 값들의 variance는 최대한 키워서 ($\sigma_{between}\sigma_{between}$)

-> 클래스들끼리 최대한 멀리 떨어지게 만드는 것이다

이를 수식으로 표현하면 다음과 같은 수식을 얻을 수 있다.

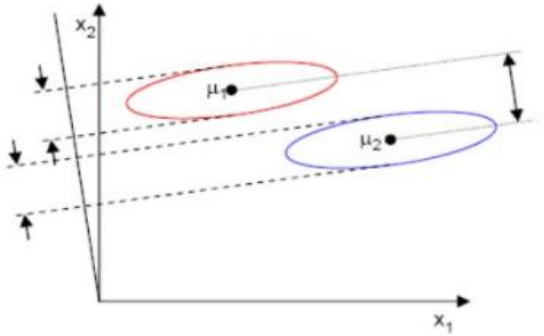
$$S = \frac{\sigma^2_{between}}{\sigma^2_{within}}$$

Unit 02 | LDA

각 클래스들에 대하여 분산은 다음과 같이 주어지며

$$\widetilde{S}_i^2 = \sum_{y \in \omega_i} (y - \widetilde{\mu}_i)^2 = \sum_{y \in \omega_i} (y - \widetilde{\mu}_i)(y - \widetilde{\mu}_i)^T$$

사영 표본들의 클래스내 분산(within-class scatter) ➔ $\left(\widetilde{S}_1^2 + \widetilde{S}_2^2\right)$



Fisher의 선형판별은 다음의 목적함수를
최대화 하는 선형함수 w^Tx 에 해당한다.

$$J(\mathbf{w}) = \frac{\left|\widetilde{\boldsymbol{\mu}}_1 - \widetilde{\boldsymbol{\mu}}_2\right|^2}{\widetilde{S}_1^2 + \widetilde{S}_2^2}$$

$\left|\widetilde{\boldsymbol{\mu}}_1 - \widetilde{\boldsymbol{\mu}}_2\right|^2$ ➔ 최대

$\widetilde{S}_1^2 + \widetilde{S}_2^2$ ➔ 최소

따라서, Fisher의 선형판별식은 동일한 클래스의
표본들은 인접하게 사영이 취해지고,
동시에 클래스 간의 사영은 중심이 가능한 멀리 떨
어지게 하는 변환행렬(w)를 찾아내는 것이다.

어떻게 변환행렬(w)에 대한 함수로 표현되는 위의 목적함수 J(w)를 최대화하는 변환 행렬 w을
찾을 것인가?

Unit 02 | LDA

최적의 사영 w 를 구하기 위해서는 J(w) 를 w 에 대한 함수로 표현해야 한다.
다차원 특징 공간에서 스캐터(scatter) 행렬은 사영 상에서 분산과 동일한 형태

$$\mathbf{S}_i^2 = \sum_{\mathbf{x}\in\omega_i} (\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)^T$$

클래스내 공분산 행렬 → S_w $\mathbf{S}_1^2 + \mathbf{S}_2^2 = \mathbf{S}_w$

사영된 y의 공분산을 특징벡터 x의 공분산 행렬의 함수로 다음과 같이 표현된다.

$$\begin{aligned}
\widetilde{\mathbf{S}}_i^{\,2} &= \sum_{\mathbf{y}\in\omega_i} (\mathbf{y}-\widetilde{\boldsymbol{\mu}}_i)(\mathbf{y}-\widetilde{\boldsymbol{\mu}}_i)^T \\
&= \sum_{\mathbf{x}\in\omega_i} (\mathbf{w}^T\mathbf{x}-\mathbf{w}^T\boldsymbol{\mu}_i)(\mathbf{w}^T\mathbf{x}-\mathbf{w}^T\boldsymbol{\mu}_i)^T \\
&= \sum_{\mathbf{x}\in\omega_i} \mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)^T\mathbf{w} \\
&= \mathbf{w}^T\mathbf{S}_i^2\mathbf{w}
\end{aligned}$$

따라서 $\widetilde{S}_1^{\,2} + \widetilde{S}_2^{\,2} = w^T S_w w$

Unit 02 | LDA

마찬가지로 사영된 평균들의 간의 차이 (분산)를 원래의 특징공간에서의 평균들 간의 차이 (분산)으로 다음과 같이 동일한 형태로 표현된다.

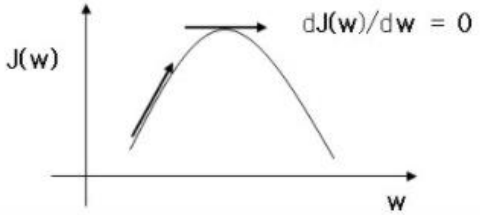
클래스간 공분산(between-class scatter) 행렬

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T\mu_1 - w^T\mu_2)^2$$
$$= w^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = w^T S_B w$$

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{(\tilde{S}_1^2 + \tilde{S}_2^2)} \Rightarrow J(w) = \frac{|w^T S_B w|}{|w^T S_w w|}$$

S_B 와 S_W로 표현된 Fisher의 기준 ➔ 이 목적함수를 최대로 하는 변환행렬 w 를 어떻게 찾을 것인가?

J(w)의 최대값을 찾기 위해서는 w 에 대하여 미분한 식을 0 으로 놓고, 이를 만족하는 w 를 구하면 된다.



선형판별분석(LDA)에 의한 변환행렬 구성 및 분류 방법

1. 클래스간 분산(between-class scatter) S_B를 구한다.

$$\mathbf{S}_B = \sum_{i=1}^{c} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \qquad \boldsymbol{\mu} = \frac{1}{N}\sum_{\forall \mathbf{x}}^{c} \mathbf{x} = \frac{1}{N}\sum_{\mathbf{x}\in\omega} N_i\boldsymbol{\mu}_i \qquad \boldsymbol{\mu}_i = \frac{1}{N_i}\sum_{\mathbf{x}\in\omega_i}\mathbf{x}$$

2. 각각의 클래스내 분산 (within-class scatter) S_{Wi}를 구한다.

$$\mathbf{S}_{W_i} = \sum_{\mathbf{x}\in\omega_i}(\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}} \qquad \left(\mathbf{S}_W = \sum_{i}^{C}\mathbf{S}_{W_i}\right) \leftarrow \text{클래스-독립의 경우}$$

3. 각각의 $\mathbf{S}_{W_i}^{-1}\mathbf{S}_B$ 에 대한 고유값 분석을 수행한다.

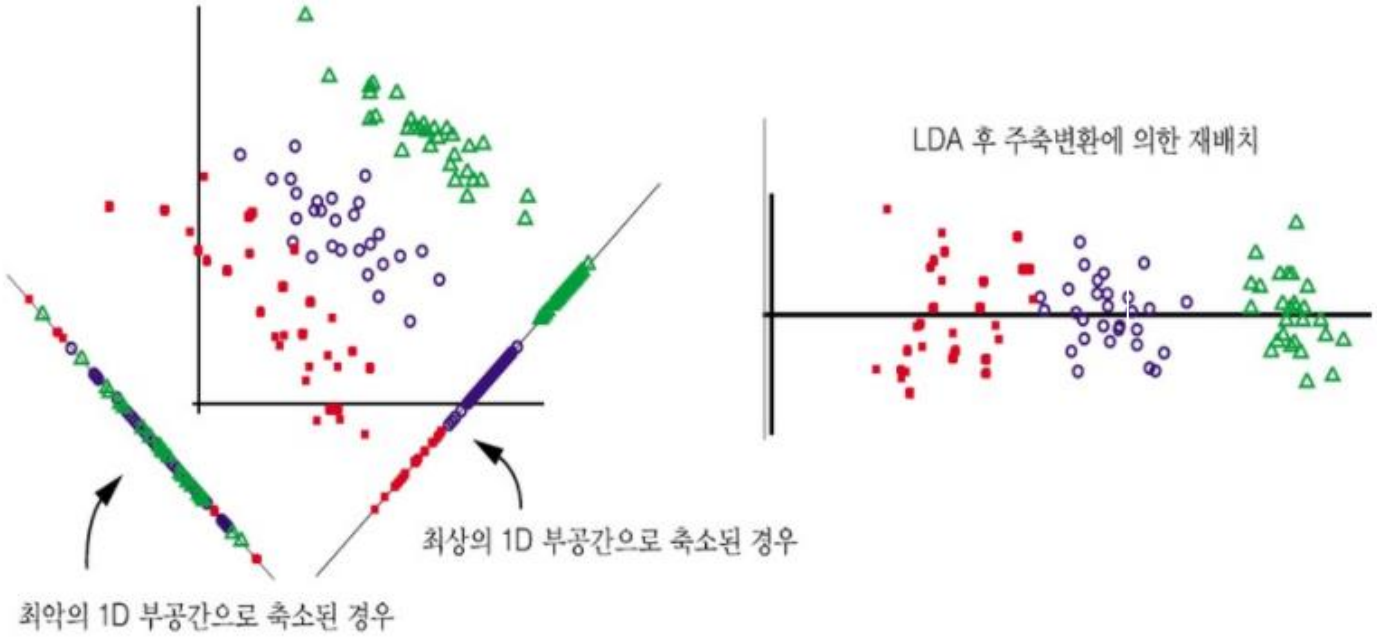
   i클래스의 고유치들 중에서 q 개의 가장 큰 고유값 $(\lambda_1,\ldots,\lambda_q)$에 해당하는 고유벡터 $(u_1,\ldots,u_q)$를 선택하여 이를 열로 하는 i-클래스의 변환행렬 W_i 를 구성한다.

$$\mathbf{W}_i = [\mathbf{u}_1 \ldots \mathbf{u}_q]$$

4. 각각의 훈련자료들을 변환한다. $\mathbf{y} = \mathbf{W}_i^T\mathbf{x}$

5. 임의의 시험자료들을 변환하여(using W_i^T), 변환된 훈련자료의 평균과 비교하여 클래스를 결정

Unit 02 | LDA



[그림 11-2] LDA에 의한 차원 축소

LDA가 잘 작동하기 위해서?

데이터의 패턴, 데이터의 내재적 속성 자체를 밝혀내는 것이 중요할 때에는
PCA (eigenface와 같이 이미지 분석같은 경우)가 더 잘 작동하고,
LDA는 다음과 같은 조건이 있을 때 잘 작동한다.

1. 데이터가 정규 분포(다변량 정규분포)를 따라야 한다.
2. 각각의 분류들은 동일한 공분산 행렬을 갖는다.
3. 피처들은 통계적으로 상호 독립적이다.

QDA : Quadratic Discriminant Analysis

LDA는 관측치들이 특정 클래스에 공통인 공분산 행렬을 다변량 정규분포로부터 추출되었다 가정한다.
이때 LDA는 각 클래스의 공분산이 동일하다고 가정하지만, QDA는 각 클래스의 공분산 행렬이 다르다고 가정한다.
단 QDA또한 LDA 처럼 클래스의 관측치들이 정규분포를 따른다 가정하고, 모수에 추정치를 대입한다.

이분산 가정하에 QDA 분류기는 추정값을 대입해 이때 그 값을 가장 크게 하는 클래스에 X=x 관측치를 할당한다.
이때 x에 대한 2차 함수 형태로 나타나 QDA라고 한다.

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k\end{aligned}$$

QDA : Quadratic Discriminant Analysis

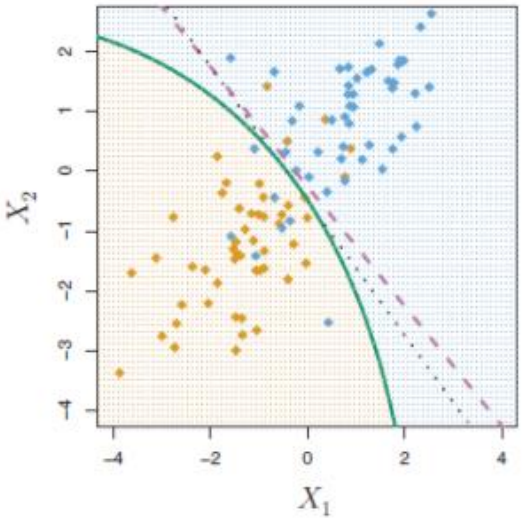
Bias-Variance Trade off 관점에서의 LDA와 QDA

LDA와 QDA의 가장 큰 차이점은 클래스의 공분산에 대한 가정이다.
예를 들어 LDA는 p개의 설명변수가 있을 때 공분산 행렬 추정은 p(p+1)/2개의 모수의 추정을 요구하는 반면에
QDA는 클래스의 개수가 k라고 했을때 k * p(p+1)/2개를 추정 해야 한다.

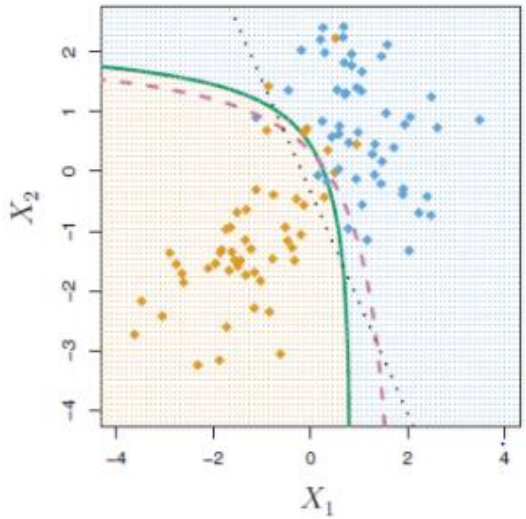
⇒ 결론적으로 LDA는 QDA에 비해서 선형적인 모형을 갖기 때문에 유연성은 떨어지지만 **낮은 분산(Variance)을 갖는다.**
⇒ 그러나 만약 데이터가 LDA의 가정에 맞지 않으면 LDA의 **편향(Bias)는 굉장히 높아진다.**

⇒ **따라서 훈련 데이터가 상대적으로 적어 분산을 줄이는 것이 중요하다면, LDA를 선택하는 것이 좋으며,**
훈련데이터가 충분히 크고 분류기의 분산이 주요 관심사가 아니거나 클래스에 대한 분산이 공통적이 아니면 QDA를 사용

공분산이 공통인 경우와 아닌 경우의 그래프



〈공분산이 공통인 경우〉



〈공분산이 다른 경우〉

좌측 그림에서 X1, X2의 사이의 상관계수가 공통적인 그림에서는 LDA가 더 잘 분류하는 반면에,

우측 그림에서는 변수들간의 상관계수가 각각 다를 경우에는 QDA가 LDA보다 더 정확하게 경계를 근사한다.

Kernel PCA : 커널 주성분 분석

비선형 데이터의 매핑을 위한 커널을 사용한 주성분 분석으로써, 회귀분석과
대부분의 분류 알고리즘에서 데이터를 선형분리 가능하다는 가정을 한다.
그러나 데이터의 모양이 선형이 아니라 비선형인 경우에는?

⇒ PCA, LDA와 같은 방법론은 기본적으로 선형변환 기법이므로
선형으로 분리 불가능한 데이터에 대해서는 적당하지 않다!

Kernel PCA (KPCA) : 비선형 데이터에 대해서 차원을 축소하기 위한 방법!

SVM에서의 커널 기법을 생각해보면, 원래의 D차원의 데이터를 K차원으로 (D < K) 매핑하는 하나의 함수를 가정하자. 이 때 이 커널을 활용하여 데이터를 고차원으로 변환하는 비선형 매핑을 수행한 뒤, 일반적인 PCA를 통해서 데이터들이 선형분리가 가능한 저차원 공간으로 다시 투영해보자 -> **커널 PCA의 기본적인 생각!!**

⇒ 기존의 PCA를 비지도로 학습시킬 때,주어진 데이터를 우리가 정한 basis에 projection시키고, 각 basis들에 사영된 값이 새로운 feature가 되는 구조이다. 이때 kernel trick을 이용해서 nonlinear한 projection으로 바뀌게 된다.

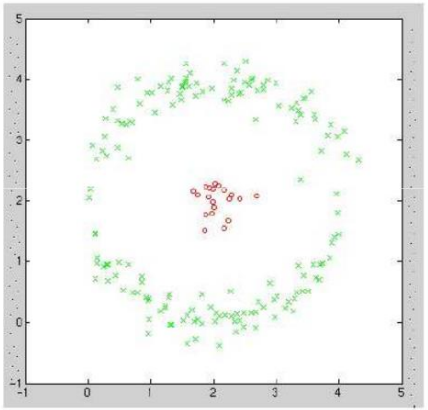
PCA + kernel function

PCA는 피처간의 관계를 공분산 행렬로 표현하였다. 이때 공분산은 피쳐간의 내적을 통해서 표현되는 데, 내적이란 두 데이터의 유사도를 의미하기도 한다.
따라서 두 벡터 사이를 내적하는 부분의 함수를 데이터를 고차원으로 mapping하는 kernel 함수로 대체하여, 두 데이터의 상관관계를 표현해보자.

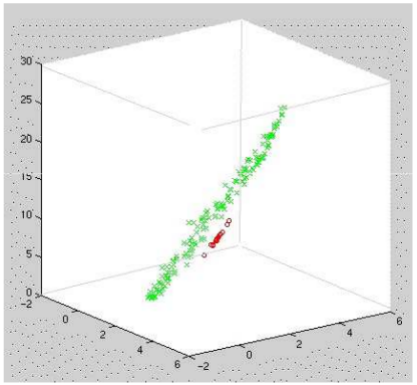
⇒ 즉 벡터의 내적이 아닌, kernel로 표현하게 되면 kernel 함수에 따른 non-linear한 표현이 가능해지며, 유사도라는 개념을 보다 범용적으로 사용가능

Kernel function

Example in $\mathbb{R}^2$



Example: High-Dimensional Mapping



We can make the problem linearly separable by a simple mapping

$$\Phi : \mathbf{R}^2 \to \mathbf{R}^3$$

$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$

Kernel Trick

- High-dimensional mapping can seriously increase computation time.
- Can we get around this problem and still get the benefit of high-D?
- Yes! Kernel Trick

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- Given *any* algorithm that can be expressed solely in terms of dot products, this trick allows us to construct different nonlinear versions of it.

Popular Kernels

Gaussian $K(\vec{x}, \vec{x}') = \exp\left(-\beta \|\vec{x} - \vec{x}'\|^2\right)$

Polynomial $K(\vec{x}, \vec{x}') = (1 + \vec{x} \cdot \vec{x}')^p$

Hyperbolic tangent $K(\vec{x}, \vec{x}') = \tanh(\vec{x} \cdot \vec{x}' + \delta)$

Summary of kernel PCA

- Pick a kernel
- Construct the normalized kernel matrix of the data (dimension m x m):

$$\widetilde{\mathrm{K}} = \mathrm{K} - 2\mathbf{1}_{1/n}\mathrm{K} + \mathbf{1}_{1/n}\mathrm{K}\mathbf{1}_{1/n}$$

- Solve an eigenvalue problem:

$$\widetilde{\mathrm{K}}\alpha_i = \lambda_i \alpha_i$$

- For any data point (new or old), we can represent it as

$$y_j = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i), \quad \mathrm{j} = 1,..,d$$

Normalized kernel matrix 에 관한 수식

$$\mathrm{K}\alpha_j = n\lambda_j \alpha_{\mathrm{j}}$$ Alpha는 eigenvector의 계수

$$v_j^T v_j = 1 \;\Rightarrow\; \sum_{k=1}^{n}\sum_{l=1}^{n} \alpha_{jl}\alpha_{jk}\phi(x_l)^T\phi(x_k) = 1 \;\Rightarrow\; \alpha_{\mathrm{j}}^{\mathrm{T}}\mathrm{K}\alpha_{\mathrm{j}} = 1$$

기존의 pca수식에 kernel을 추가하고, 고유벡터의 계수에 대해서 normalize 조건

$$\lambda_j n \alpha_j^T \alpha_j = 1, \quad \forall j \qquad \phi(x)^T v_j = \sum_{i=1}^{n} \alpha_{ji}\phi(x)^T\phi(x_i) = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i)$$

새로운 x가 있을때, PCA처럼 사영시킨다. (using normalizing condition)

- The corresponding kernel is:

$$\widetilde{K}(x_i, x_j) = \widetilde{\phi}(x_i)^T \widetilde{\phi}(x_j)$$

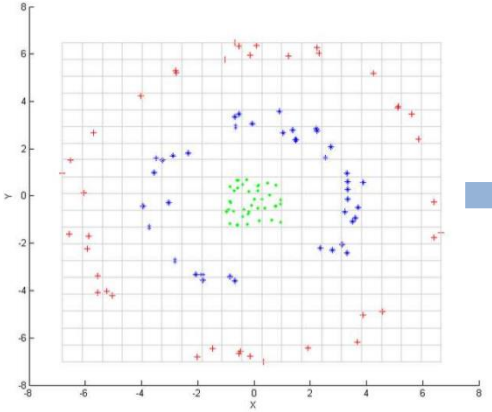
$$= \left(\phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)^T \left(\phi(x_j) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)$$

$$= K(x_i, x_j) - \frac{1}{n}\sum_{k=1}^{n} K(x_i, x_k) - \frac{1}{n}\sum_{k=1}^{n} K(x_j, x_k) + \frac{1}{n^2}\sum_{l,k=1}^{n} K(x_l, x_k)$$

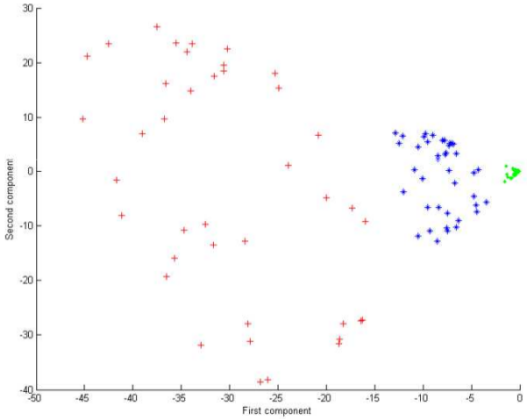
최종적으로 kernel을 추가하여 feature space에 관한 식을 도출

KPCA의 결과

Example: Input Points



Example: KPCA



실제로 KPCA를 적용한 결과
비선형 데이터를 잘 분류할 수 있다.
그러나 다른 곳에 일반화하여
활용에는 한계가 생길 수 있다.
또한 **kernel matrix를 연산해야** 하기에
만약 데이터가 많다면 계산하는데
어려움이 있을 수 있다.

들어주셔서 감사합니다.