출처가 명시되지 않은 모든 자료(이미지 등)는 조성현 강사님 블로그 및 강의 자료 기반.

[머신러닝-분류]

[SVM]

딥러닝이 등장하기 이전 부흥했던 알고리즘이다. 앞서 보았던 KNN, Decision Tree 모두 집단을 분류하기 위해 결정 경계를 찾는 것이 중요했다. 결국, *분류* 문제는 여러 클래스 간 **경계**를 측정하는 것이 중요한 문제이다.

1. 분류 원리

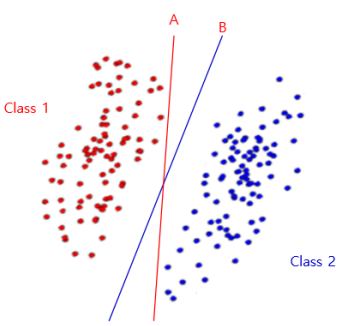
SVM 알고리즘은 분류 작업을 수행하기 위해, 훈련 데이터에서 **마진**을 최대로 하는 서포트 벡터를 찾는 알고리즘이다.

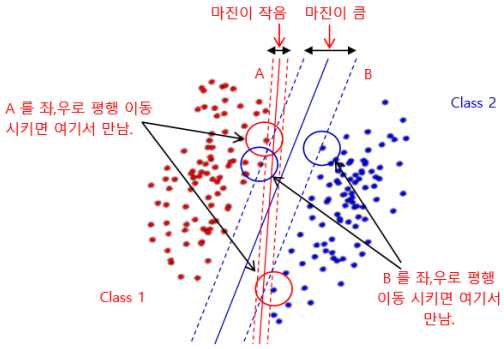
주요 개념

- 마진(Margin) : 클래스를 구분하는 결정 경계와 가장 가까운 훈련 데이터 사이의 거리.
- 서포트 벡터(Support Vector) : 클래스를 구분하는 결정 경계와 가장 가까운 훈련 데이터.

최대 마진

교재의 예시를 통해 최대 마진이 무엇인가 이해해 보자.





왼쪽 그림에서 빨간색 점들이 class 1에 속하고, 파란색 점들이 class 2에 속한다고 하자. 이 때 가상의 의사결정 경계를 A 또는 B로 그어 판단한다고 하자.

두 선 모두 훈련 데이터의 클래스를 제대로 구분하고 있다. 그러나 새로운 데이터가 들어왔을 때, 각각의 직선이 새로 들어온 데이터를 어떻게 구분할지, 그 효과는 동일하지 않다. **일반화 특성**이 달라지는 것이다.

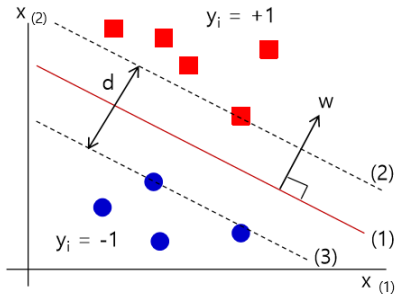
두 직선 A와 B를 각각 왼쪽과 오른쪽으로 이동시킨다고 하자. 이 때 A는 조금만 움직여도 각 클래스의 샘플과 맞닿지만, B는 움직일 수 있는 공간이 더 많기 때문이다. 이렇게 *결정 경계를 이동시킬 수 있는 폭*이 바로 **마진**이다.

두 직선 모두 훈련 데이터를 잘 분류한다는 것을 알고 있기 때문에, 마진이 더 큰 결정 경계 B가 더 적합한 결정 경계가 된다. 이런 방식으로 마진을 찾는 것이 **최대 마진**을 찾는 원리이다. 각 데이터를 구분할 수 있는 여러 후보 경계들이 모두 훈련 데이터를 정확하게 분류한다면, 마진이 클수록 일반화 오차가 낮아지기 때문에, 큰 마진을 갖는 결정 경계를 찾는 것이다. (반대로 말하면, 마진이 작을수록 훈련 데이터에 과대적합되기 쉽다는 의미이다.)

수학적 원리

SVM 알고리즘에서 결정 경계를 찾는 것은 선형 분리가 가능한 경우와 불가능한 경우로 나뉜다. 전자를 **하드 마진 분류**, 후자를 **소프트 마진 분류**라고 한다.

HARD 마진



$$(1)\ w_1 x_{(1)} + w_2 x_{(2)} + b = 0 \rightarrow w \cdot x + b = 0$$

$$(2)\ w \cdot x + b = k \qquad (k > 0)$$

$$(3)\ w \cdot x + b = -k$$

$$y = \begin{cases} +1 \leftarrow & w \cdot x + b > 0 \\ -1 \leftarrow & w \cdot x + b < 0 \end{cases}$$

$$(4)\ d = \frac{2k}{\| w \|} \quad \longleftarrow \text{ d 를 최대화 = w 를 최소화}$$

$$(5)\ y_i(w \cdot x_i + b) \geq k \quad (i = 1, 2, \ldots N) \longleftarrow \text{제한 조건}$$

$$(6)\ \operatorname{argmin} \frac{1}{2} \| w \|^2, \quad constraint : y_i(w \cdot x_i + b) \geq k$$

$$(7)\ L_p = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{N} \lambda_i \{ y_i(w \cdot x_i + b) - k \}$$

Primal Lagrange (최소화)

$$(8)\ \frac{\partial L_p}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$(9)\ \frac{\partial L_p}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

Dual Lagrange (최대화)

$$(10)\ L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i \cdot x_j$$

위와 같은 최대 마진의 결정 경계를 찾는 과정을 최적화 원리를 통해 이해할 수 있다.

선형으로 표현되는 직선의 방정식인 결정 경계를 수식으로 나타내면 (1)과 같다.

편의상, 결정 경계 위쪽의 샘플을 양성 영역, 음성 영역이라 하자. 아래 쪽의 샘플을 음성 샘플이라 하자. 교재 그림에서는 (2) 영역의 샘플이 양성 샘플, (3) 영역의 샘플이 음성 샘플이 되는 것이다. 양성 영역의 서포트 벡터와 만나는 결정 경계와 음성 영역의 서포트 벡터와 만나는 결정 경계를 각각 수식으로 나타내면 수학 식으로 나타내면 (2)와 (3)의 식이 된다.

(2)에서 (3)을 빼면 두 직선 사이의 거리가 나오고, 이것이 바로 마진이 된다. 위의 식에서 마진은 (4) 식의 `d`가 된다. 결정 경계가 양성 샘플과 음성 샘플을 모두 분류한다는 전제가 있어야 하므로, 최대 마진을 찾는 문제는 **(5)의 제약조건을 만족하는 선에서 (4)의 `d`를 최대화하는 최적화 문제**가 된다.

(4)의 `d`를 최대화하기 위해서는 분모에 있는 행렬식 `||w||`를 최소화해야 한다.

$$||W|| = (W^T \times W)^{\frac{1}{2}}$$

수학과 님의 도움에 의하면 위와 같다고 합니다.. 행렬의 크기를 구한 거라고..

그리고 위의 식에 의해 풀어야 할 최적화 문제를 나타내면 (6)에서와 같이

$$\frac{1}{2} \times ||w||^2$$

를 최소화하는 문제가 된다. (7)에서의 식과 같아지는 것이다.

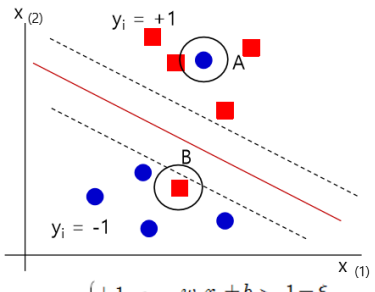
그런데 듀얼 라그랑지안 법칙(범위를 넘는다고 판단해 듣기만 하고 기록은 하지 않는다.)에 의해 (7)의 최소화 문제는 (10)의 최대화 문제와 같다.(라그랑지안 승수를 알 수 없으므로, 라그랑지안 승수를 목적함수에 둔다고 한다. 둘 모두 최적화 조건이 (8)과 (9)로 동일하다.) 쿼드라틱 프로그래밍(이 역시 해당 과정의 범위를 넘는다고 판단해 기록은 하지 않는다.)

결과적으로 (10)의 최적화 문제를 수학적으로 풀어내면 최대 마진을 갖는 선형 결정 경계를 찾을 수 있게 된다.

SOFT 마진

그러나 데이터의 분포를 보았을 때, 위와 같이 선형 분리가 가능하지 않은 경우도 있다. 데이터가 어떻게 하더라도 선형 분리가 되지 않는다면, 분리의 기준을 완화하여 선형 경계를 벗어난 데이터도 인정하는 방향으로 경계를 찾는 과정을 바꿔야 할 것이다.

이를 위해 **슬랙 변수**를 도입한다. 즉, 슬랙 변수는 선형적으로 구분되지 않는 데이터에서 선형 제약의 조건을 완화하기 위해 도입된 변수이다. 선형적으로 완벽히 분리되지는 못하지만, 적절하게 손해를 보면서(몇 개 데이터의 분류 오차를 용인하면서) 최적의 분류 경계를 찾고자 한다.



$$(24)\ y = \begin{cases} +1 & \leftarrow\ w \cdot x_i + b \geq 1 - \xi_i \\ -1 & \leftarrow\ w \cdot x_i + b \leq -1 + \xi_i \end{cases} \quad (\xi_i > 0)$$

- Slack 변수 (ξ_i)를 추가하여 경계를 벗어나는 것을 허용.
- 점 A와 B는 경계를 벗어나 있음. $(\xi_A > 0,\ \xi_B > 0)$
- $y_A = -1, \xi_A = 10$ 이라면, $w \cdot x_A + b \leq 9$ 가 되어 제한 조건에 위배되지 않을 수 있음.

$$(25)\ \underset{\xi_i > 0}{argmin} \frac{1}{2} \| w \|^2 + C\sum_{i=1}^{N}\xi_i$$

$$(26)\ L_p = \frac{1}{2} \| w \|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\{y_i(w \cdot x_i + b) - 1 + \xi_i\} - \sum_{i=1}^{N}\mu_i\xi_i$$

$$(27)\ \frac{\partial L_p}{\partial w} = 0 \longrightarrow w = \sum_{i=1}^{N}\lambda_i y_i x_i$$

$$(28)\ \frac{\partial L_p}{\partial b} = 0 \longrightarrow \boxed{\sum_{i=1}^{N}\lambda_i y_i = 0} \leftarrow \text{제한 조건}$$

$$(29)\ \frac{\partial L_p}{\partial \xi_i} = 0 \rightarrow \lambda_i + \mu_i = C \rightarrow \boxed{0 \leq \lambda_i \leq C}$$

- Dual Lagrange(최대화) – 선형 분리 가능한 경우와 동일함.
- 제한 조건만 다름

$$(30)\ L_D = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i \cdot x_j$$

하드 마진 분류에서의 목적식에 slack 변수로 C를 추가한다. 목적 함수의 loss에 대한 벌점 항이라고 보자. 이 벌점항을 적절히 조절한다면, 몇 개의 데이터가 경계를 벗어나더라도, 다른 데이터들은 경계를 벗어나지 않도록 할 수 있다.

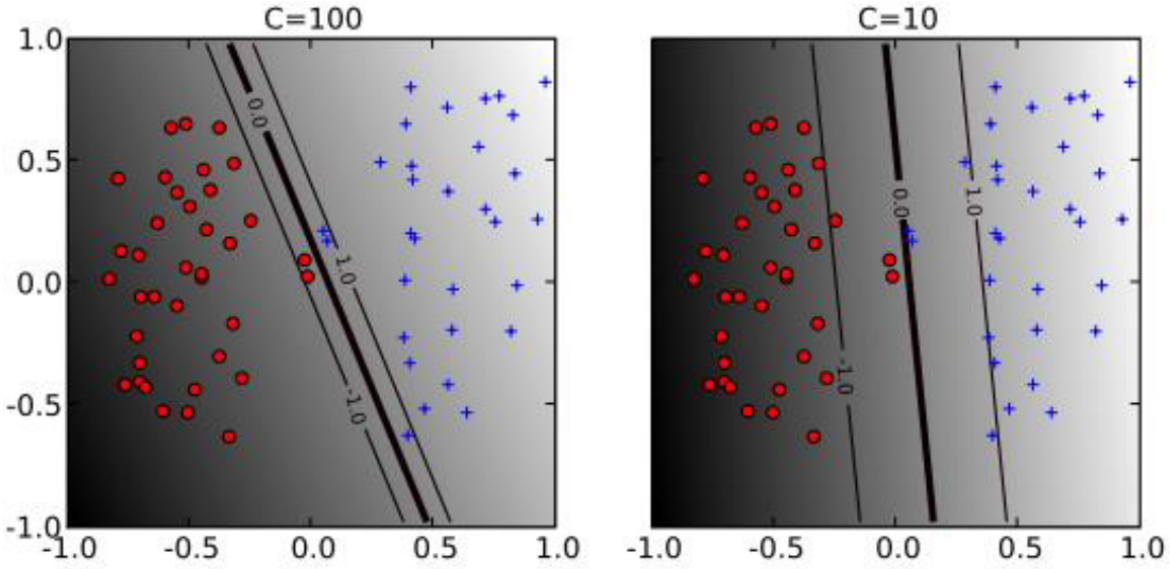
수학적 과정을 전부 공부하는 것은, 역시 범위를 넘어간다고 본다.

> 제약조건이 늘어나기 때문에 라그랑지안 승수가 2개 필요하게 된다. 일단, 강의 내용을 다 이해하기 보다는 슬랙 변수가 없는 두 두 개의 듀얼 라그랑지안 문제를 푼다고 생각하고 넘어가자.

다만, 벌점항의 도입으로 인해 마진을 최대화할 것인지, 섞인 비율을 최소화할 것인지의 **의사 결정(내지는 취사 선택?!) 문제**가 발생한다는 것을 알아두자.

마진을 넓히면 넓힐수록 마진 내에 제대로 분류되지 않는 데이터는 많아질 수밖에 없다. 당연히, 마진을 줄이면서 제대로 분류되지 않는 데이터를 줄일 수도 있다. 그러나 이 경우 새로운 데이터가 들어왔을 때 일반화가 될 수 있을지에 대해 의문이 발생한다.

수식으로 표현한 라그랑지안 최적화 문제의 식에서, **C를 크게 한다**는 이야기는 *마진을 크게 신경쓰지 않고 섞이는 것에 대한 벌점을 더 크게 주어 섞인 데이터의 비율을 최소화* 하겠다는 말이 된다.



C의 크기에 따라 마진이 어떻게 달라지는 지 확인하자.

그림 출처 : https://ratsgo.github.io/machine%20learning/2017/05/29/SVM2/

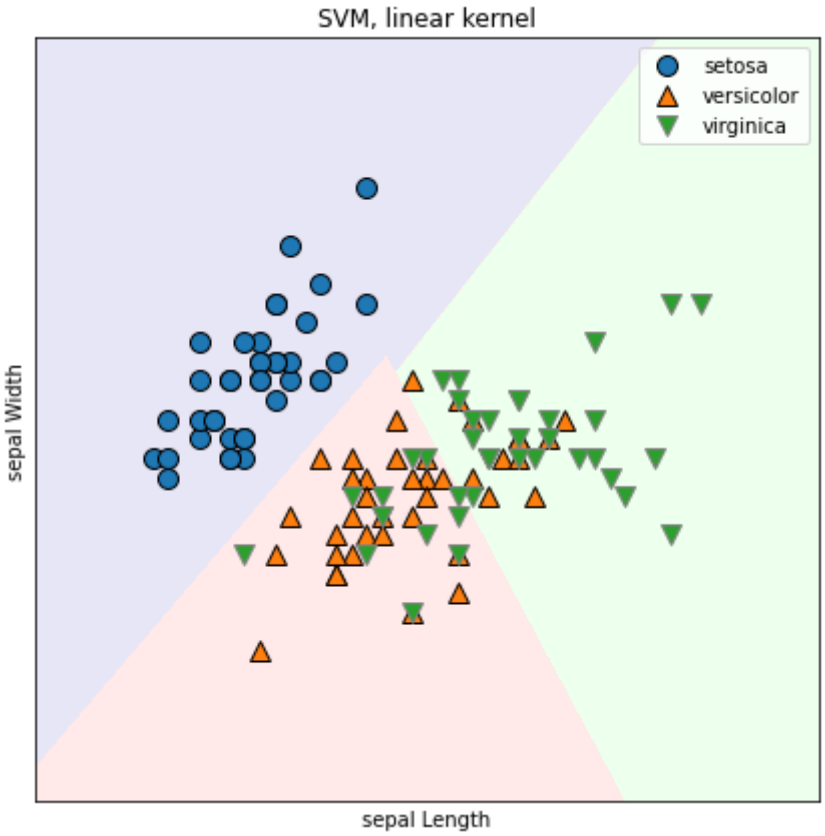
결국 선형적 분리가 불가능한 상황에서 C는 조절 변수의 역할을 하고, 그 크기를 조절하는 문제는 전적으로 분석자의 판단에 달리게 된다.

2. OVR 방식

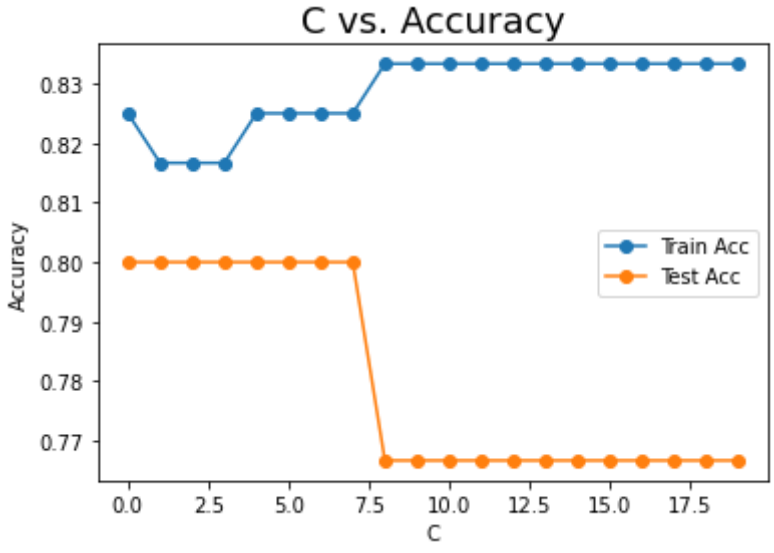
SVM은 기본적으로 이진 분류에 적용되는 알고리즘이다. 따라서 클래스가 여러 개인 경우, OVR 방식(One vs Rest)을 따라 분류한다. 하나의 클래스를 +1이라고 놓고, 나머지를 -1이라고 간주해 결정 경계를 찾고, -1로 분류된 데이터에 대해 같은 과정을 반복하는 방식이다.

실습 1. Iris Dataset

- `SVC` : 사이킷런에서 SVM 알고리즘을 구현한다. `kernel = linear` 옵션을 설정하면 선형 SVM 알고리즘을 적용해 분류할 수 있다.
- `mglearn` : SVM 시각화. 시각화를 위해 2개의 feature만 사용한다.



조절 변수 C를 변경하며 accuracy를 관찰한 결과를 시각화하면 다음과 같다.



3. 비선형 : 커널 트릭

선형 분리가 불가능한 경우, 커널 트릭을 써서 분류할 수도 있다. 딥러닝 등장 이전 SVM 알고리즘이 인기가 많았던 이유도 여기에 있다. 결국 비선형 문제를 풀어내야 하는데, 딥러닝이 부흥하기 전에 SVM이 커널 기법을 사용해 비선형 분류 문제를 풀어낼 수 있었기 때문이다.

커널 트릭 : 공간 변환

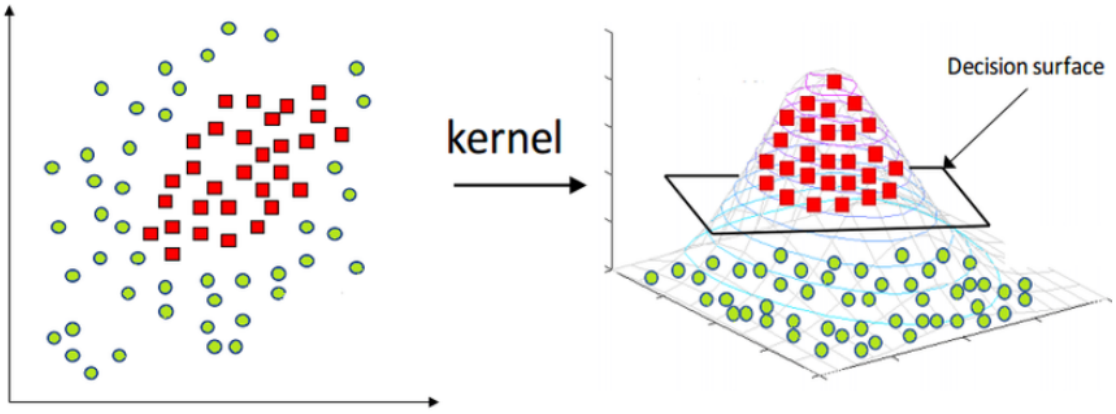


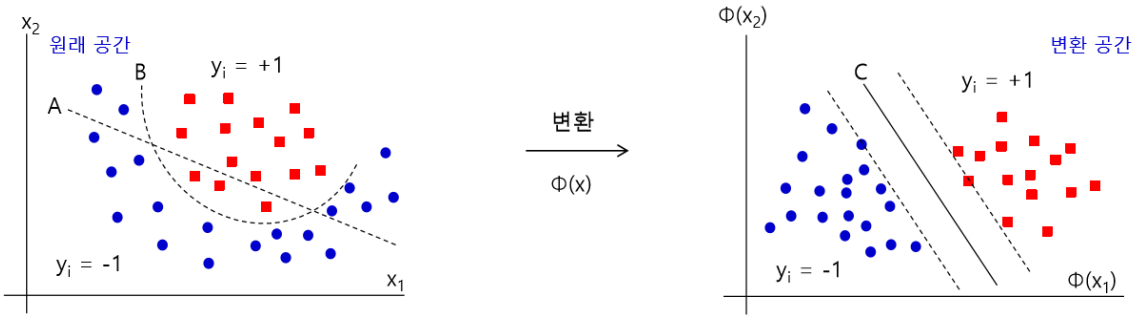
그림 출처 : https://m.blog.naver.com/PostView.nhn?blogId=sanghan1990&logNo=221136439100&proxyReferer=https:%2F%2Fwww.google.com%2F

위의 그림에서와 같은 데이터가 있다고 상상해 보자. 왼쪽 그림을 보면, 이 데이터는 2차원 공간에서는 선형 분리가 불가능하다. 그런데 이 데이터들이 3차원 공간에서 오른쪽과 같이 분포해 있다고 한다면, 두 데이터를 분류하는 결정 면을 찾아낼 수 있다. (직선이 아니라 면이기 때문에, 결정 경계가 아니라 *hyper plane*이다.)

> 종이를 구긴다고 생각하자!

위와 같이 선형 분리가 불가능한 경우여도 데이터들의 차원을 적절하게 변환할 수 있다면 데이터 간 결정 경계 역할을 하는 hyper plane을 찾아 분류 문제를 풀어낼 수 있을 것이다. *이론적으로는*, 어떠한 데이터라도, **무한 차원으로 보내면 선형 분리가 가능**해진다.

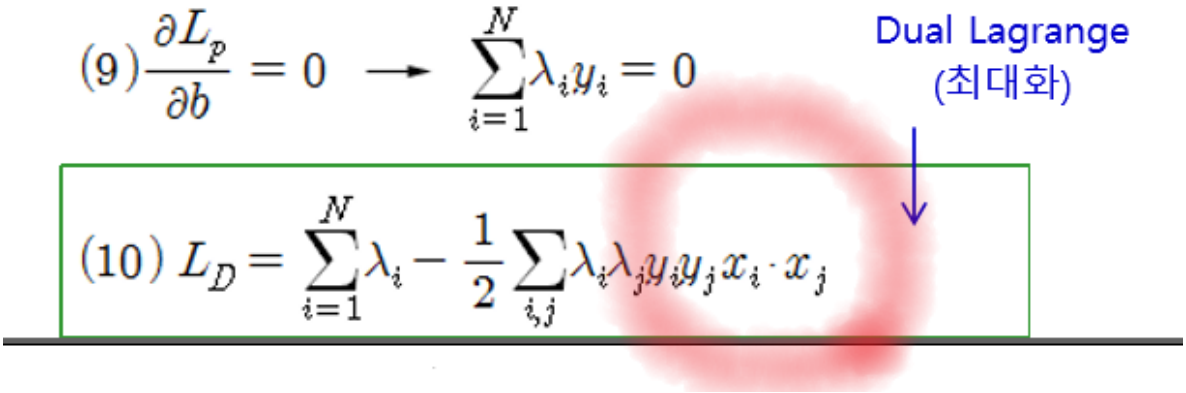
수학적 원리



데이터들을 또 다른 차원의 공간으로 변환하는 공간 변환 함수 `phi`를 생각해 보자.

이 함수가 무엇인지는 중요하지 않다. 강의의 범위를 벗어난다. 공간 변환을 통해 선형 분리가 가능해진다고 할 때, 그 핵심은 벡터의 내적에 있다.

잠시 위로 돌아가 최대 마진을 결정하기 위한 목적 함수를 보자.



$$(9)\ \frac{\partial L_p}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$(10)\ L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i \cdot x_j$$

라그랑지안 최적화 문제에 각 데이터 샘플 벡터의 내적이 있음을 확인할 수 있다.

공간을 변환해 더 고차원의 공간이 된다 하여도,

- 그 고차원의 공간 상에서 데이터 샘플의 벡터에 공간 변환함수를 적용한 후,
- 그 내적을 취한 값이,
- 2차원 공간 상에서 내적을 취했을 때의 기존 값과 동일하다면,

원래 공간에서의 데이터 내적과 고차원으로 변환된 공간에서의 내적을 적절하게 mapping할 수 있다.

요컨대, 변환된 공간에서 듀얼 라그랑지 문제를 풀기 위해 다음의 관계를 만족하는 공간 변환 함수만 찾으면 되는 것이다.

$$K(u, v) = \Phi(u) \cdot \Phi(v) = g(u \cdot v)$$

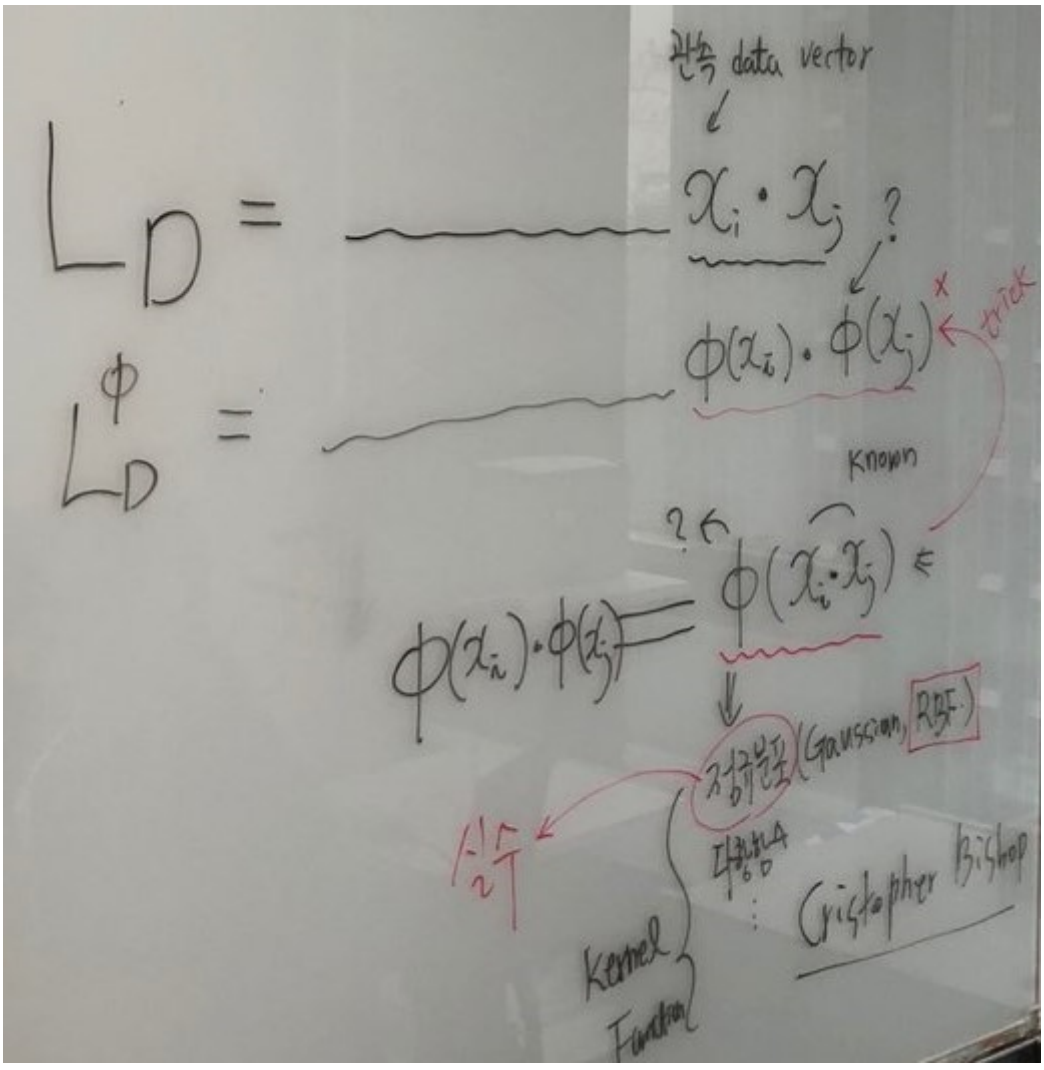
K는 커널 함수를 의미한다.

커널 트릭에 '트릭'이라는 이름이 붙은 이유가 여기에 있다. 고차원의 공간에 데이터를 매핑해 경계면을 찾아낸다는 아이디어는 좋지만, 아무리 고차원의 공간으로 데이터를 변환할 수 있더라도, 차원이 증가하면 데이터를 매핑하고, 매핑된 공간에서의 내적을 계산하는 데에 엄청난 계산량이 필요하다. 커널 트릭은 **위의 조건을 만족하는 함수만 찾는다면, 고차원 공간 상에서 모든 계산을 수행하지 않더라도 같은 효과를 낼 수 있게** 만들어 준다.

이러한 커널 트릭을 만족하는 함수들만 찾아내면, 기존의 최적화 문제가 아래와 같이 바뀌게 된다.

$$(31)\ L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$$

위의 식에서와 달라진 부분을 보자. 공간 변환 함수 간 내적을 계산하면, 라그랑지안 최적화 문제를 풀 수 있다! 커널 트릭의 힘이다. 2차원의 공간을 또 다른 어떤 공간으로 변환하는 함수(Kernel Function)만 알 수 있다면, 데이터를 알고 있으니까 내적만 구하면 가능해 진다.



> *강사님의 가르침*
>
> 커널 함수가 어떻게 동작하는지, 공간 변환 함수가 어떤 건지는 알 수도 없고, 알려고 하지도 말자. 범위를 넘어도 한참 넘는다. 그냥 커널 트릭이 뭔지 그 작동 원리만 이해하자.
>
> 어쨌든 딥러닝이 나오면서 SVM도 다 옛날의 학문이 되었다. 딥러닝이 태생적으로 비선형 데이터 문제를 다루기 위해 태어난 학문이기 때문에, 머신러닝으로 다루려고 하는 건 예전의 방식이다.

사이킷런에서는 SVC 함수를 사용합니다. 선형분리일 때는 일단 커널 리니어로 모델 빌드하고, 학습 데이터로 fit. fit 과정에서 람다 QT 알고리즘으로 찾아 내고. 늘 했던 대로 테스트 데이터를 넣어서 성능을 평가하면 된다. 카테고리가 3개이므로 내부적으로 2번 수행한다.

위와 같은 성질을 만족하는 공간 변환 함수를 커널 함수라고 한다. 널리 사용되는 커널로 다음과 같은 것들이 있다. (출처 : 데이터사이언스스쿨)

- 다항 커널 (Polynomial Kernel)

$$k(x1, x2) = (\gamma(xT1x2) + \theta)^d$$

- RBF(Radial Basis Function) 또는 가우시안 커널(Gaussian Kernel)

$$k(x1, x2) = exp(-\gamma||x1 - x2||^2)$$

- 시그모이드 커널 (Sigmoid Kernel)

$$k(x1, x2) = tanh(\gamma(x_1^T x2) + \theta)$$

참고로, 앞에서 소개했던 선형 SVM은 `γ = 1`, `θ = 0`, `d = 2` 인 다항 커널 함수를 사용한다고 볼 수 있다.

실습 2. Iris Dataset

사이킷런의 SVM 클래스는 `kernel` 인수를 지정하여 커널을 지정할 수 있도록 한다. `gamma`와 `C` 파라미터를 조정하며 최적의 분류 결과를 찾고, 그 결과를 시각화하여 비교하면 다음과 같다.

```
optimal Gamma = 0.10
optimal C = 0.20
best TestSet Accuracy = 0.80
Model: SVC(C=0.2, gamma=0.1)
SVM rbf, train_accuracy: 81.66666666666667%
SVM rbf, test_accuracy: 80.0%
```

gamma = 1.0, C = 5.0	gamma = 0.1, C = 0.2