7강: Kernels

SVM 복습
Optimal Margin Classifiers
Optimization objective (Dual Optimization Problem)
Kernels
Feature maps
Kernel trick
Properties of Kernel
How to make kernels
Regularizatation and non-seperable case

SVM 복습

- **label** :$y \in \{-1, 1\}$
- **classifier** : $h_{w,b}(x) = g(w^T x + b)$, $g(z) = \begin{cases} 1, & z \geq 0 \\ -1, & \text{otherwise} \end{cases}$
- **functional margins** : $\hat{\gamma}^{(i)} = y(i)(w^T x^{(i)} + b)$

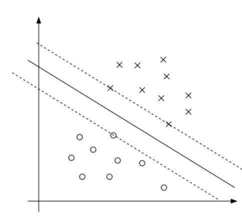
  if
  $y(i)(w^T x^{(i)} + b) > 0$ → prediction correct

  $\hat{\gamma} = \min_{i=1,...n} \hat{\gamma}^{(i)}$
- **geometric margins** : $\gamma^{(i)} = y^{(i)}((\frac{w}{\|w\|})^T x^{(i)} + \frac{b}{\|w\|})$
- note that if $\|w\| = 1$ → functional margin = geometric margin

→ $\gamma = \min_{i=1,...n} \gamma^{(i)}$

- **support vectors**

  : points with the **smallest margins**
  ; closest to the decision boundary



Optimal Margin Classifiers

우리의 목표는 **geometric margin을 최대화하는 결정경계를 찾는 것**이다.

이는 positive/negative training examples 간 분류를 **'gap'**을 통해서 분류하는 것과 같다.

training set이 **linearly separable 하다고 가정**할 때, 어떻게 해야 **maximal geometric margin**을 얻을 수 있을까 ?

우리는 다음과 같은 optimization problem을 제기할 수 있다. ~~(제거랄이 아니다! 제기할)~~

$$\max_{\gamma,w,b} \gamma \\ s.t.\ y^{(i)}(w^Tx^{(i)}+b) \geq \gamma,\ i=1,....,n \\ \|w\| = 1.$$

이 때, 제약 조건 $\|w\| = 1$로 인해 functional margin = geometric margin이 보장된다.

하지만, 동시에 $\|w\| = 1$은 non-convex하므로 우리가 최적화 소프트웨어로 풀 수 없다.

~~조금만 기다려보시라… 조물조물… 얍!~~

$$\max_{\hat{\gamma},w,b} \frac{\hat{\gamma}}{\|w\|} \\ s.t.\ y^{(i)}(w^Tx^{(i)}+b) \geq \hat{\gamma},\ i=1,....,n$$

최적화 문제를 다음과 같이 고치면 $\gamma = \frac{\hat{\gamma}}{\|w\|}$에 의해 geometric & functional margins 간 연관이 된다. 또한 성가신 제약조건 $\|w\| = 1$을 없앨 수 있다.

물론…. $\frac{\hat{\gamma}}{\|w\|}$이 아직 여전히 non-convex하다. 아직도 컴퓨터로 풀 수 없다! ~~(으아아악)~~

training set에 관한 w, b로 표현되는 functional margin에 scaling constraint를 도입하면 $\hat{\gamma} = 1$이고,
위의 제약조건에 이를 대입하면
$\frac{\hat{\gamma}}{\|w\|} = \frac{1}{\|w\|}$이 된다. 최적화 문제를 다음과 같이 바뀐다.

$$\min_{w,b} \frac{1}{2}\|w\|^2 \\ s.t.\ y^{(i)}(w^Tx^{(i)}+b) \geq 1,\ i=1,....,n$$

위의 경우라면 **convex quadratic objective**가 되고, 이 최적화 문제의 해는 **optimal margin classifier**가 된다.
해는
quadratic programming (QP) code를 통해 구현할 수 있다.

Optimization objective (Dual Optimization Problem)

$$\min_{w,b} \frac{1}{2}\|w\|^2 \\ s.t.\ y^{(i)}(w^Tx^{(i)}+b) \geq 1,\ i=1,....,n$$

위의 최적화 문제의 **제약 조건**을 다음과 같이 고칠 수 있다.

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \le 0$$

라그랑주 승수를 이용해 최적화 문제를 표현하면 다음과 같다.

(“ there’re only“α_i” but no “β_i” Lagrange multipliers, since the problem has <u>only inequality constraints</u>.)

이 문제의 dual form을 찾기 위해서는 $\mathcal{L}(w, b, \alpha)$을 w, b에 대해서 최소화(α는 고정)해야한다.

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)} = 0 \ \rightarrow w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$$

b에 대해 미분해 보면 $\frac{\partial}{\partial b}\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$임을 알 수 있다.

위에서 구한 정보를 가지고 식을 정리해보면 우리는 다음과 같은 식을 얻는다.

$$\begin{aligned} \mathcal{L}(w, b, \alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{n} \alpha_i y^{(i)} \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} \end{aligned}$$

위의 w와 b에 관한 $\mathcal{L}$의 최소화 문제를 α에 대한 **dual optimization** 문제로 바꾸어 생각해볼 수 있다.

$$\begin{aligned} \max_{\alpha} \ W(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)} x^{(j)} \rangle \\ s.t. \ \ \alpha &\ge 0, i = 1, ...n \\ \sum_{i=1}^{n} \alpha_i y^{(i)} &= 0 \end{aligned}$$

- should also be able to verify that the conditions required for $p^* = d^*$

- the **KKT conditions** to hold are indeed satisfied in our optimization problem

이를 이용해서 w^Tx+b를 x와 training set의 **내적(inner product)**로 표현할 수 있다.

- algorithm **in terms of only inner products between input feature vectors**
  (not individual obs)
- original x값은 몰라도 된다→ 내적값만 알면 된다.
- estimation on parameters : requires only $\binom{n}{2}$ inner products

$$\begin{aligned} w^Tx+b &= (\sum_{i=1}^{n}\alpha_i y^{(i)}x^{(i)})^T x + b \\ &= \sum_{i=1}^{n}\alpha_i y^{(i)}\langle x^{(i)}, x\rangle + b \end{aligned}$$

이렇게 **내적의 관점에서 알고리즘을 표현할 수 있다는 특성**을 통해서 **kernel**을 이용한 분류 또한 진행할 수 있다.

> optimal margin classifier + kernel trick = SVM

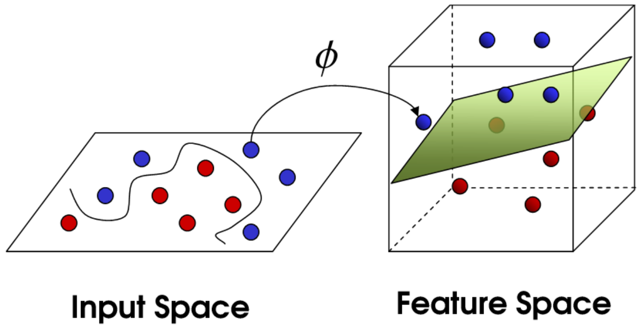
데이터 마이닝 수업에서 배웠던 Suppot Vector Machines

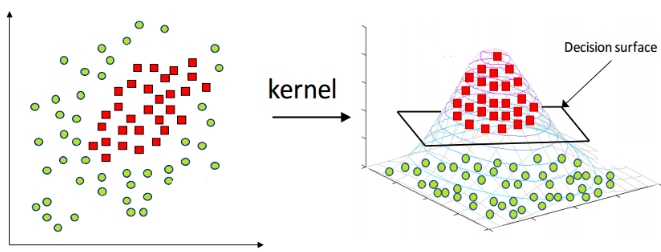
- *only support vectors affect the hyperplane*
  (else do not affect the hyperplane)
- Support vector classifier's decision rule is based on only a small subset of training data
- Estimated parameters : $\alpha_i \neq 0$ for support vectors ; $\alpha_i = 0$ for other obs
- Difference from LDA(GDA) : decision rule is based only on a small subset of training data → ***robust*** to obs far away from the hyperplane.

Kernels

선형 결정 경계를 그을 수 없는 경우는 어떡하면 좋을까?

원 데이터의 차원을 확장시킨 후 hyperplane을 찾으면 된다.





Feature maps

$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$ 라는 함수가 있다고 하자.

그리고 $\phi(x) = \mathbb{R} \rightarrow \mathbb{R}^4$를 다음과 같이 정의하자.

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4$$

$\theta \in \mathbb{R}^4 = \{\theta_0, \theta_1, \theta_2, \theta_3\}$라 하면 우리는 y를 $\theta^T \phi(x)$로 재정의할 수 있다.

x에 대한 cubic function은 $\phi(x)$에 대한 linear function으로 여길 수 있다. (발번역 어게인)

- ***attributes*** : original input values (x)
- ***feature variables*** : new quantities - When the original input is mapped to some new set of quantities $\phi(x)$
- ***feature map*** : ϕ - maps the attributes to the features

Kernel trick

~~you just activated my kernerl trick card~~

1. Write your whole algoritms in terms of $\langle x^{(i)}, x^{(j)} \rangle$
   (편의상 다음 단계에서는 내적을 그냥
   $\langle x, z \rangle$로 적겠음)
2. Let there will be mapping from $x \rightarrow \phi(x)$
   ; expanding
   x to a high dimensional (even be infinite) feature vector with x

   이 때 feature mapping의 값들을 내적하는데에는 두 가지 문제점이 있다.

   ① $\phi(x)$로 feature space를 만드는 연산량이 엄청나다(그리고 $\phi(x)$**를 정의하는 것 자체가 어렵다**)

   ② **내적** $\langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$**의 연산량**이 엄청 나다

   ① 고차원 변환 + ② 내적 연산량 문제를 동시에 해결하고자 고안된 것이 바로 **kernel**이다.

3. Find a way to compute $K(x, z) = \phi(x)\phi(z)$
4. Replace $\langle x, z \rangle$ in algorithm with $K(x, z)$

위의 방법 처럼 $K(x, z)$을 계산할 수 있다면 $\phi(x)$**와 그 내적을 직접 계산할 필요가 없이 항상 kernel을 통해 연산을 수행**하면 된다.

Properties of Kernel

몇 가지 예시를 통해서 커널 함수의 특징을 알아보자.

$x, z \in \mathbb{R}^d$라 할 때, 커널 함수 $K(\cdot, \cdot)$~~제법 귀여운 물개 같이 생겼다~~를 다음과 같이 정의해 보자:

$$K(x, z) = (x^T z)^2$$

이는 다음과 같이도 쓸 수 있다.

$$\begin{aligned} K(x, z) &= (\sum_{i=1}^{d} x_i z_i)(\sum_{i=1}^{d} x_j z_j) \\ &= \sum_{i=1}^{d} \sum_{i=1}^{d} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{d} (x_i x_j)(z_i z_j) \end{aligned}$$

따라서, $K(x, z) = \phi(x)\phi(z)$ 는 아래처럼 주어진 feature mapping ϕ에 대응하는 커널함수임을 알 수 있다.

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

이 때, $d = 3$ 일 때, 다차원의 $\phi(x)$는 $O(d^2)$ time의 computation을 요구하는 반면,

$K(x, z)$의 경우 $O(d)$ time의 computation을 요구한다. (linear in the dimension of the input attributes)

다른 연관된 예도 보자. $K(\cdot, \cdot)$를 다음과 같이 정의한다고 하자.

$$\begin{aligned} K(x, z) &= (x^Tz + c)^2 \\ &= \sum_{i,j=1}^{d}(x_ix_j)(z_iz_j) + \sum_{i=1}^{d}(\sqrt{2c}x_i)(\sqrt{2c}x_j) + c^2 \end{aligned}$$

이 경우 K 는 아래처럼 주어진 feature mapping ϕ에 대응하는 커널함수임을 알 수 있다.

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}$$

$K(x, z) = (x^Tz + c)^2$는 $\binom{d+k}{k}$차의 featrue space로 mapping하는 $\phi(x)$와 상응한다.

$\phi(x)$의 경우 $O(d^k)$ time인 반면 , $K(x, z)$는 $O(d)$ time만 걸린다.

이렇게 커널 함수를 이용한 효과적인 연산방법이 있기 때문에! ϕ**의 연산을 직접 해줄 필요는 없다.**

How to make kernels

Kernel의 기하학적 의미 → criteria

커널함수는 결국 **두 벡터의 내적**을 나타내는 함수로, **기하학적으로 cosine 유사도를 의미**한다.
(이 때문에
similarity function이라고도 불린다.)

$$a \cdot b = \|a\|\|b\|cos\theta$$

이를 고려한다면 다음과 같은 기준이 요구된다고 할 수 있다.

- x & z are ***similar*** →$K(x,z)$=$\phi^T(x)\phi(z)$ are ***large***
- x & z are ***dissimilar*** → $K(x,z) = \phi^T(x)\phi(z)$ are ***small***

Conditions for valid kernels

1. Necessary Conditions

만약 K가 valid kernel이라면

- **symmetric** ($\because K_{ij} = K(x_{(i)}, x_{(j)}) = \phi(x_{(i)})^T\phi(x_{(j)}) = \phi(x_{(j)})^T\phi(x_{(i)}) = K(x_{(j)}, x_{(i)}) = K_{ji}$)
- $K \geq 0$: K is **positive semi-definite**

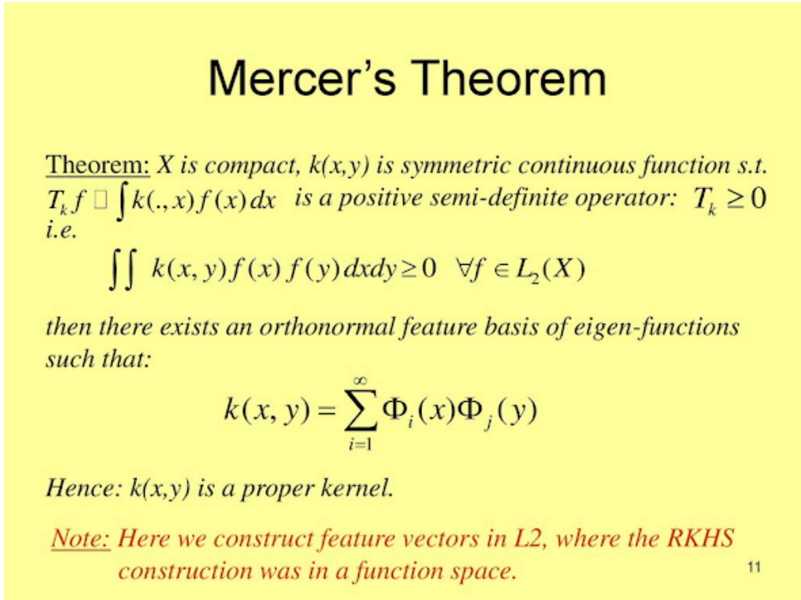
임의의 z에 대해서

$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\ &= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)})^T \phi_k(x^{(j)}) z_j \\ &= \sum_i \sum_j \sum_k z_i \phi_k(x^{(i)})^T \phi_k(x^{(j)}) z_j \\ &= \sum_k (\sum_i z_i \phi_k(x^{(i)}))^2 \\ &\geq 0 \end{aligned}$$

2. Sufficient Conditions

앞의 두 필요조건이 곧 충분 조건이기도 하다.

이는 Mercer’s Theorem 덕분이다.



- Theorem (Mercer) : Let K : $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any$\{x_{(1)}, ..., x_{(n)}\}, (n < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

요약하자면

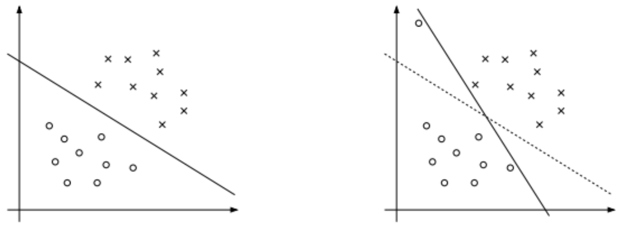
- **Kernel 함수 K가** 실수 scalar 를 출력하는 **continuous function**일 때
- Kernel 함수값으로 만든 행렬이 **Symmetric** (대칭행렬)이고
- **Positive semi-definite** (대각원소>0)라면

$K(x_i, x_j) = K(x_j, x_i) = \langle \phi(x_i), \phi(x_j) \rangle$**를 만족하는 mapping ϕ가 존재**한다.

Type of Kernel function $K(\cdot, \cdot)$

- d^{th} Degree polynomials :$(1 + \langle x_i, z_i \rangle)^d$
- Radial Basis : $exp(-\gamma ||x_i - z_i||^2)$
- Neaural Network : $tanh(K_1 \langle x_i, z_i \rangle + K_2)$
- gaussian : $exp(-\frac{||x-z||^2}{2\sigma^2})$

Regularizatation and non-seperable case



왼쪽의 그림의 경우 optimal margin classifier를 보여준다.

outlier 데이터 값이 추가되면 결정 경계의 기울기가 급격하게 변화하면서
margin의 길이도 훨씬 작아지는 것을 볼 수 있다.

알고리즘을 이상치에 덜 민감하게, 그리고 non-linearly sepearable한 데이터셋에도 잘 작동하게 하기 위해서 우리는 optimization을 아래의 형태로 정의할 수 있다. (using l_1 regularization)

$$\min_{\gamma,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$s.t.\ y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i,\ i = 1, ..., n$$
$$\xi_i \geq 0\ , i = 1, ..., n$$

위의 식은 오분류를 허용하는 식으로, C는 얼마 만큼 오분류를 허용할지 결정하는 하이퍼 파라미터이다.

라그랑지안을 적용하면 다음과 같다.

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i] - \sum_{i=1}^{n} r_i\xi_i$$

α_i, r_i는 Lagrange multipliers (constrained to be ≥ 0) 이다.

아까의 방법처럼 **dual form**을 구하면

$$\max_{\alpha}\ W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}x^{(j)}\rangle$$
$$s.t.\ \ 0 \leq \alpha_i \leq C, i = 1, ...n$$
$$\sum_{i=1}^{n}\alpha_i y^{(i)} = 0$$

역시 w가 α_i에 관해서 정의될 수 있다.

05.07-Support-Vector-Machines.html