PRML과 부수적인 자료들을 공부하여 간단히 정리하였습니다.

이번 장에서는 SVM에 대해 공부할 것이다. SVM은

- classification, regression, novelty detection에 사용한다.
- decision machine이기 때문에 posterior probability를 알수는 없다.
- model parameter를 구하는데 있어 convex optimization problem이기에 any local solution이 global optimum이 된다.
- high dimension에서 분류할 때 좋은 generalization 성능을 보인다
- train이 quadratic programmin problem이다
- train data에 대해 fit하지만 generalization의 성능이 좋다.
- statistical learning theory라는 탄탄한 이론에 기반하여 만들어졌다.

7.1 Maximum Margin Classifiers (hard margin)

일단 몇 가지 가정을 하고 공부를 시작해보자.

- two class classification problem using linear model

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$$

- **train data가 linearly separable in feature space라고 가정**
    - 따라서 최소한 하나의 오류없이 100% 분류가능한 $\mathbf{w}, b$가 존재
- training data $(\mathbf{x}_1, ..., \mathbf{x}_n), (t_1, ..., t_n), t_n \in \{-1, 1\}$
- decision boundary : $y(\mathbf{x}) = 0$
- $y(\mathbf{x}_n) < 0 \rightarrow t_n = -1,\ y(\mathbf{x}_n) > 0 \rightarrow t_n = 1$
    - $\therefore y(\mathbf{x}_n)t_n > 0$ for all train data
- new data are classified by the sign of $y(\mathbf{x})$

train data가 lineary separable하므로 우리의 목표는 이를 나누는 hyperplane을 찾는 것이다. 그렇다면 가장 좋은 hyperplane을 어떻게 찾을까?

- SVM에서 **hyperplane (decision boundary)는 *margin* 을 최대화** 하도록 한다.
    - margin : smallest distance between the decision boundary and any of the samples
    - 이에 해당하는 그림을 찾아보면 이해하기 쉬울 것이다.
    - train set에서 margin 최대화 = generalization error 최소화 = good prediction

hyperplane과 data point $\mathbf{x}$의 거리는

$$\frac{|y(\mathbf{x})|}{||\mathbf{w}||}$$

인데 가정에 따라 모든 data point는 잘 분류되어 있다. (그 중에서 best hyperplane을 찾기). 따라서 $t_n y(\mathbf{x}_n) > 0$ 인 상황이므로

$$\frac{t_n y(\mathbf{x}_n)}{||\mathbf{w}||}$$

으로 나타낼 수 있다. 따라서 maximize margin solution은

$$\arg\max_{w,b}\{\frac{1}{||\mathbf{w}||} min_n[t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)]\}$$

이를 바로 풀기는 어려움이 있어서 약간의 변화를 준다.

- (가정) $t_n(\mathbf{w}^T\phi(x_n) + b) = 1$ for the point that is closest to the decision boundary.
    - 따라서 $t_n(\mathbf{w}^T\phi(x_n) + b) \geq 1$ 의 제약이 생긴다.
    - 대신 min 뒷 부분을 제외하고 optimization이 가능하다. (constraint가 생기지만)

이를 통해 우리는 (convex) optimization problem을 아래와 같이 정의할 수 있다.

- constraint : $t_n(\mathbf{w}^T\phi(x_n)+b) \geq 1$

$$\arg\min_{w,b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

+ 위 내용을 다른 방법으로 접근

- 고려대학교 김성범 교수님의 유투브 참조

위의 margin을 찾는 과정에 대해 조금 다른 derivation을 소개한다.

- $y(\mathbf{x})=1, y(\mathbf{x})=-1$ 위의 점을 각각 $\mathbf{x}^+, \mathbf{x}^-$ 라고 하자 : support vector, 즉 decision boundary와 가장 가까운 각 class 의 점들을 의미
  -$\mathbf{x}^-$을 움직여서 반대편 class $y(\mathbf{w})=-1$ 위의 점으로 만들 수 있다.
  - $\mathbf{x}^+ = \mathbf{x}^- + \alpha\mathbf{w}$

$$\begin{gathered}\mathbf{w}^T\mathbf{x}^+ + b = 1\\ \mathbf{w}^T(\mathbf{x}^- + \alpha\mathbf{w}) + b = 1\\ \mathbf{w}^T\mathbf{x}^- + b + \alpha\mathbf{w}^T\mathbf{w} = 1\\ \therefore \alpha = \frac{2}{\mathbf{w}^T\mathbf{w}}\end{gathered}$$

- margin = distance($\mathbf{x}^+, \mathbf{x}^-$) / 2

$$\begin{gathered}= \left\|\mathbf{x}^+ - \mathbf{x}^-\right\|_2/2\\ = \|\alpha\mathbf{w}\|_2/2\\ = \alpha\sqrt{\mathbf{w}^T\mathbf{w}}/2\\ = \frac{1}{\sqrt{\mathbf{w}^T\mathbf{w}}} = \frac{1}{\|\mathbf{w}\|_2}\end{gathered}$$

이렇게 약간 다른 과정을 통해 같은 결과를 얻을 수 있다.

이를 *quadratic programming* problem이라고 한다.

- **quadratic programming** : minimize a quadratic function subject to a set of linear inequality constraints

이를 통해 Lagrange function을 만들어 parameter를 구해보자. 제약식이 있는 **maximum margin problem을 Lagrangian Primal문제로 변환** 시킨 것이다.

- **Lagrangian Primal**
  - $max_a min_{w,b} L(\mathbf{w}, b, \mathbf{a})$
  - Lagrange multipliers $a_n \geq 0$

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)-1\}$$

이 식을 각 parameter $\mathbf{w}, b$ 로 미분하면

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

이를 통해 maximum margin problem을 *dual representation* 으로 나타낼 수 있다.

- [과정]

첫번째 항

$$
\begin{aligned}
\frac{1}{2}\|\mathbf{w}\|_2^2 &= \frac{1}{2}\mathbf{w}^T\mathbf{w} \\
&= \frac{1}{2}\mathbf{w}^T\sum_{n=1} a_n t_n \phi(\mathbf{x}_n) \\
&= \frac{1}{2}\sum_{n=1} a_n t_n \mathbf{w}^T\phi(\mathbf{x}_n) \\
&= \frac{1}{2}\sum_{n=1} a_n t_n \sum_{m=1} a_m t_m \phi(\mathbf{x}_m)^T\phi(\mathbf{x}_n) \\
&= \frac{1}{2}\sum_{n=1}\sum_{m=1} a_n t_n a_m t_m \phi(\mathbf{x}_m)^T\phi(\mathbf{x}_n)
\end{aligned}
$$

두번째 항

$$
\begin{aligned}
&-\sum_{n=1}^{N} a_n\{t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)-1\} \\
&= -\sum_{n=1} a_n t_n \mathbf{w}^T\phi(\mathbf{x}_n) - b\sum_{n=1} a_n t_n + \sum_{n=1} a_n \\
&= -\sum_{n=1}\sum_{m=1} a_n a_m t_n t_m \phi(\mathbf{x}_m)^T\phi(\mathbf{x}_n) + \sum_{n=1} a_n
\end{aligned}
$$

따라서 최종적으로

- **Lagrangian dual**
  - constraint (제약식)
    - $a_n \geq 0$
    - $\sum_{n=1}^{N} a_n t_n = 0$

$$
L(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(\mathbf{x}_m)^T\phi(\mathbf{x}_n)
$$

위 식을 maximize하면 되고 이는 다시 a에 관한 quadratic problem이 된다.

- Lagrangian Dual problem으로 바뀐 것이다.
  - objective function is quadratic & constraint is linear
  - 따라서 여기서 Lagrangian Dual은 quadratic programming이 된다.
- 또한 kernel function을 사용할 수 있는 형태을 얻었다. (inner product)

우리는 $\mathbf{a}$를 구하여 $\mathbf{w}, b$ 모두 알 수 있고 prediction도 할 수 있다.

새로운 데이터를 분류하기 위해 $y(\mathbf{x})$의 sign을 알면 된다. 그리고 prediction을 위해 굳이 $\mathbf{w}, b$를 구하지 않고 아래의 식으로 prediction하면 된다.

$$
y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x})^T\phi(\mathbf{x}_n) + b
$$

그런데 $(\mathbf{w}, b, \mathbf{a})$가 Lagrangian dual problem의 최적해가 되기 위한 조건으로 *KKT condition* 을 만족해야 한다.

- KKT(Karuch-Kuhn-Tuker) condition
  - Stationarity
    - $\frac{\partial L(\mathbf{w},b,\mathbf{a})}{\partial \mathbf{w}} = 0, \frac{\partial L(\mathbf{w},b,\mathbf{a})}{\partial \mathbf{b}} = 0$
  - Primal feasibility
    - $t_n y(\mathbf{x}_n) - 1 \geq 0$

- Dual feasibility
  - $a_n \geq 0$
- Complementary slackness
  - $a_n\{t_n y(\mathbf{x}_n) - 1\} = 0$

이를 통해 SVM에서 중요한 내용을 생각할 수 있는데 모든 점은 두 가지 경우에 해당한다.

- $a_n > 0 \; and \; t_n y(\mathbf{x}_n) - 1 = 0$
- $a_n = 0 \; and \; t_n y(\mathbf{x}_n) - 1 \neq 0$

근데 a_n**이 0인 경우는 decision boundary를 만드는 것과 prediction에 아무런 영향을 주지 않는다.** 영향을 주는 점들은 *support vector* 라고 부른다. 이들은 $t_n y(x_n) = 1$ 이고 maximum margin hyperplane의 위에 있는 점들이다. (각 class별로 decision boundary와 가장 가까운 점에 관한 hyperplane)

- support vector만으로 optimal hyperplane(decision boundary)를 구할 수 있다. 그래서 sparse kernel machine이라고 부르는 것이다.

7.1.1 Overlapping class distributions (soft margin)

이전까지 우리는 완전히 separable이 가능한 경우에서 모델링하였다. 하지만 이런 경우는 드물다. overlap되는 경우에 우리는 penalty를 주고 misclassification이 된 경우가 존재하는 모델을 만드는 것이다. 이를 *soft margin* 이라고 부른다. 이 penalty를 위해 각 data point마다 slack variable을 생각한다.

- slack variable (penalty)
  - $\xi_n = 0$: 데이터가 잘 분류되고 margin 밖에 위치
  - 이외의 경우 : $\xi_n = |t_n - y(x_n)|$ (틀린 거리만큼 penalty)
    - $0 < \xi_n < 1$: 잘 분류되었지만 margin 범위 안에 위치
    - $\xi_n = 1$: data가 +, - boundary 위에 위치
    - $\xi_n > 1$: 잘못 분류

이에 따라 $t_n y(x_n) \geq 1 - \xi_n$ 으로 constraint가 바뀐다(error를 허용하는). 따라서 우리는 minimize

- constraint : $t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1 - \xi_n$

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\mathbf{w}\|_2^2$$

하게 된다.

- 여기서 C는 trade-off 관계를 컨트롤하는 파라미터이다.
  - C가 커지면 error를 많이 허용하지 않으므로 overfit
  - C가 작으면 error를 많이 허용하므로 underfit

이 식을 위에서 했던 대로 Lagrange로 계산하면 위의 dual representation과 같은 결과가 나온다. 이번에는 Lagrange multiplyer가 두 개가 필요하다. 다른 점은 constraint, KKT 가 달라진다.

- Lagrangian Primal
  - $max_{a,\gamma} min_{w,b,\xi} L(\mathbf{w}, b, \mathbf{a}, \xi, \gamma)$
  - Lagrange multipliers $a_n \gamma_n \geq 0$

$$L(\mathbf{w}, b, \mathbf{a}, \xi, \gamma)$$
$$= \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{n=1} \xi_n - \sum_{n=1} a_n \{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1 + \xi_n\} - \sum_{n=1} \gamma_n \xi_n$$

이 식을 각 parameter $\mathbf{w}, b, \xi$ 로 미분하면

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

$$C - a_n - \gamma_n = 0$$

- Lagrange dual
  - constraint (제약식)
    - $a_n \geq 0$
    - $\sum_{n=1}^{N} a_n t_n = 0$

$$L(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

- KKT(Karuch-Kuhn-Tuker) condition
  - $\frac{\partial L(\mathbf{w},b,\mathbf{a})}{\partial \mathbf{w}} = 0, \frac{\partial L(\mathbf{w},b,\mathbf{a})}{\partial \mathbf{b}} = 0$
  - $C - a_n - \gamma_n = 0$
  - $a_n\{t_n y(\mathbf{x}_n) - 1 + \xi\} = 0, \gamma_n \xi_n = 0$

이번에도 마찬가지로 solution의 특징을 보면

- $a_n\{t_n y(\mathbf{x}_n) - 1 + \xi\} = 0,\ \gamma_n \xi_n = 0,\ a_n = C - \gamma_n$
  - $a_n = 0 \Rightarrow \gamma_n = C \Rightarrow \xi_n = 0 \Rightarrow t_n y(\mathbf{x}_n) - 1 \neq 0$
    - $\mathbf{x}_n$ 이 +,- boundary 보다 멀리 잘 분류되었다.
  - $0 < a_n < C \Rightarrow \gamma_n > 0 \Rightarrow \xi_n = 0, \gamma_n \xi_n = 0 \Rightarrow t_n y(\mathbf{x}_n) - 1 = 0$
    - $\mathbf{x}_n$ 이 +,- boundary 위에 있다. (support vector)
  - $a_n = C \Rightarrow \gamma_n = 0 \Rightarrow \xi_n > 0 \Rightarrow t_n y(\mathbf{x}_n) - 1 = -a_n \xi_n \neq 0$
    - $\mathbf{x}_n$ 이 +,- boundary 과 decision boundary 사이에 존재한다. (margin의 범위에 존재, 이들도 support vector라고 함)

이제 kernel method for nonlinear classification에 대해 살펴보자. 우리의 위에서 dual problem을 통해 kernel function의 사용가능성을 파악할 수 있었다. kernel을 사용함으로서 nonlinear decision boundary를 만드는데 있어서 inner product $\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$ 이 아니라 $k(\mathbf{x}_n, \mathbf{x}_m)$으로 계산할 수 있다.

예시를 통해 kernel의 유용성을 알아보자.

- $\mathbf{x}, \mathbf{z}$는 2차원 vector
- kernel function : $k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})$ 라고 하자.

$$\begin{aligned} \phi(\mathbf{x})^T \phi(\mathbf{z}) = (1\ \sqrt{2}x_1\ \sqrt{2}x_2\ x_1^2\ \sqrt{2}x_1 x_2\ x_2^2)(1\ \sqrt{2}z_1\ \sqrt{2}z_2\ z_1^2\ \sqrt{2}z_1 z_2\ z_2^2)^T \\ = (1 + x_1 z_1 + x_2 z_2)^2 = (1 + \mathbf{x}^T \mathbf{z})^2 \end{aligned}$$

이처럼

- **kernel을 통해 기존의 $\mathbf{x}$을 nonlinear하게 만들어서 decision boundary를 만들 수 있어서 classification을 더 잘 할 수 있는 것이다.** (여기서는 처음부터 $\phi(\mathbf{x})$를 사용하였지만 지금까지의 모든 과정을 $\mathbf{x}$ 라고 생각하면 kernel을 통해 nonlinear하게 만들었다고 이해할 수 있다.)
- **기존의 data를 explicit하게 $\phi(\mathbf{x})$ (nonlinear하게) 으로 만든 뒤에 inner product로 계산하는 것이 아니라 kernel function을 통해 같은 결과를 만들 수 있기에 explicit하게 몰라도 되고 상당히 computationally 좋다.**

가장 많이 쓰이는 kernel은

- Gaussian Kernel (Radial basis function Kernel)

$$k(\mathbf{x}, \mathbf{z}) = exp(\frac{-\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2})$$

7.1.2 Comparison to logistic regression

그림을 찾아보는 것을 추천한다.

- SVM loss function : Hinge loss $\xi_j = (1-(wx_j+b)y_j)_+$
- logistic loss function : Log loss $\xi_j = -\log(1+exp(wx_j+b)y_j)$

$$\because \theta_{MLE} = \arg\max \sum log(P(T_i\ X_i;\theta))$$
$$= \arg\max \sum \{Y_i X_i \theta - \log(1+exp(X_i\theta))\}$$

Hinge loss는 correct한 경우 penalty가 0이 되지만 log loss는 corret한 경우에도 0이 되지 않는다. log loss가 좀 덜 극단적인 것 같다.

7.1.3 Multiclass SVM

다양한 방법이 있다. 하지만 다들 한계점이 있는 것으로 보인다.

7.1.4 SVMs for regression

어렵지 않다. 정리는 생략하고자 한다.

- ϵ-insensitive error function

$$E_\epsilon((y(x)-t)) = \begin{Bmatrix} 0, \quad if\ |y(x)-t| < \epsilon \\ |y(x)-t|-\epsilon,\ otherwise \end{Bmatrix},$$

아래의 식을 최소화한다.

$$C\sum_{n=1}^{N} E_\epsilon((y(x)-t)) + \frac{1}{2}\|\mathbf{w}\|^2$$

7.2 Relevance vector machines

SVM은 확률적인 요소가 없다. 이를 위해 Bayesian SVM이라고 할 수 있는 RVM이 있다. 하지만 나중에 필요하면 다시 공부하고자 한다.