DA-Final-4) Support Vector Machines (SVM)

1. SVM Introduction

1-1. SVM VS ANN

SVM
- Use "kernel trick" to learn nonlinear functions
- A few hyper-parameters to tune
- Learned in batch mode
  (한 묶음)

ANN
- Use multi-layers to learn nonlinear functions
- Many hyper-parameters to tune
- Can be learned incrementally

1-2. Polynomial classifiers

- Linear function: consider weights of individual features

$$f(\mathbf{x}) = b + \sum_{i=1}^{n} w_i x_i$$

+

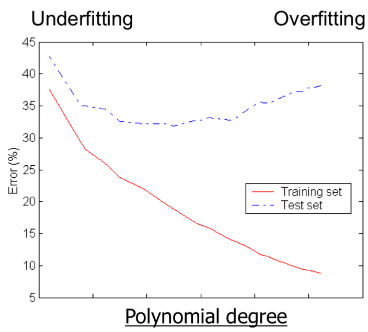
- Quadratic function: also consider weights of concatenations of two features

$$f(\mathbf{x}) = b + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j$$

- Polynomial function: also consider weights of concatenations of multiple features (*Degree p* denotes the size of concatenations)

$$f(\mathbf{x}) = b + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\, x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{j=1}^{n} w_{ij\,k} x_i x_j x_k + \cdots$$

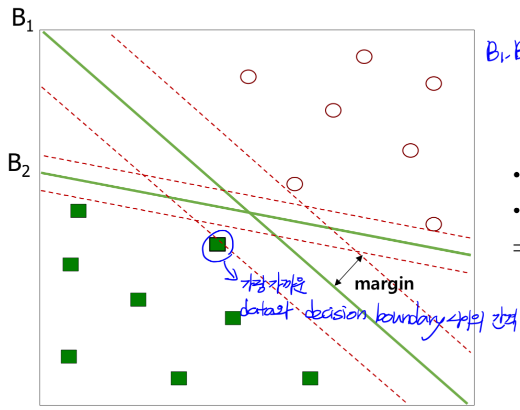
- *SVM learns a polynomial function with an arbitrary degree in linear time in terms of # of features using "kernel trick".*



model의 복잡도

2. Linear SVM

2-1. Linear SVM 개요



B_1, B_2 : Linear hyper-plane
(Decision boundary)

- Which of B_1 or B_2 is better? Generalization↑수록 좋다. for Unseen data
- How do you quantify the "goodness"?

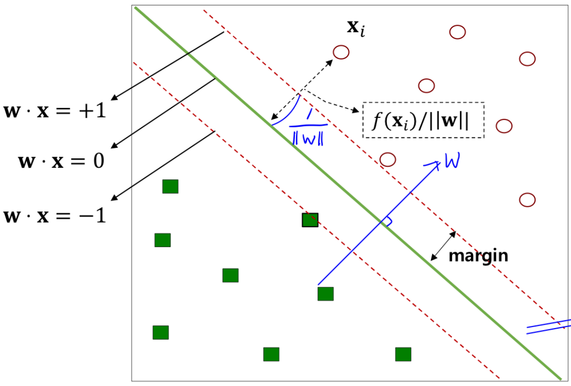
⇒The hyperplane that *maximizes the margin*
(Generalization의 하나의 척도)

- Given a labeled dataset $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ where $\mathbf{x}_i$ is a data vector, and y_i is its class label (+1 or -1)

- D is "linearly separable",
  - if there exists a linear function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ that correctly classifies every data vector in D, that is,
  - if there exists $\mathbf{w}$ that satisfies $(\mathbf{w} \cdot \mathbf{x}_i) y_i > 0$ for all $(\mathbf{x}_i, y_i) \in D$.
    (+) (+)
    (−) (−)

- Then, there exist $\mathbf{w}'$ that satisfies $(\mathbf{w}' \cdot \mathbf{x}_i) y_i \geq 1$ for all $(\mathbf{x}_i, y_i) \in D$?

2-2. Linear SVM 정의



$\frac{w\cdot x}{\|w\|} \geq \frac{1}{\|w\|}$

canonical!

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}\cdot\mathbf{x} \geq +1 \\ -1 & \text{if } \mathbf{w}\cdot\mathbf{x} \leq -1 \end{cases}$$

$$\text{margin} = \frac{1}{\|\mathbf{w}\|}$$

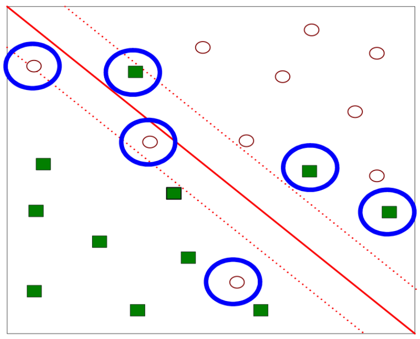
$(w\cdot x)y \geq 1$
for all x, y

즉 margin 영역에는 data가 존재X
를 의미!

- We want to maximize: margin $= \frac{1}{\|\mathbf{w}\|}$
- Instead, we minimize: $L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}$
- But subject to the following constraints:

  $(\mathbf{w}\cdot\mathbf{x}_i)y_i \geq 1$ for all $(\mathbf{x}_i, y_i) \in D$
- This is a constrained optimization problem.
- Numerical approaches to solve it (e.g. quadratic programming)

2-3. Linear SVM using Soft margin 정의



What if not linearly separable?

- Introduce slack variables ξ_i
- Minimize:

$$L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C\left(\sum_{i=1}^{m}\xi_i\right)$$

- Subject to:

$$\forall(\mathbf{x}_i, y_i) \in D: (\mathbf{w}\cdot\mathbf{x}_i)y_i \geq 1 - \xi_i \ (\xi_i \geq 0)$$

Primal form

- *Minimize:*

error term

$$L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C\left(\sum_{i=1}^{m}\xi_i\right)$$

- Subject to:

$$\forall(\mathbf{x}_i, y_i) \in D: (\mathbf{w}\cdot\mathbf{x}_i)y_i \geq 1 - \xi_i \ (\xi_i \geq 0)$$

: Binary classification constraints

Dual form

- *Maximize:*

$$M(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j$$

- Subject to:

$$C \geq \alpha_i \geq 0, \qquad \sum_{i=1}^{m}\alpha_i y_i = 0$$

- $\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i$

- Many of the α_i are zero, and $\mathbf{w}$ is a linear combination of a small number of data points (i.e. support vectors).
- $\mathbf{x}_i$ with non-zero α_i are called *support vectors*.

$$\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i$$

- The decision boundary is determined only by the support vectors
- Prediction on a new data $\mathbf{x}$:

$$f(\mathbf{x}) = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i\cdot\mathbf{x}$$

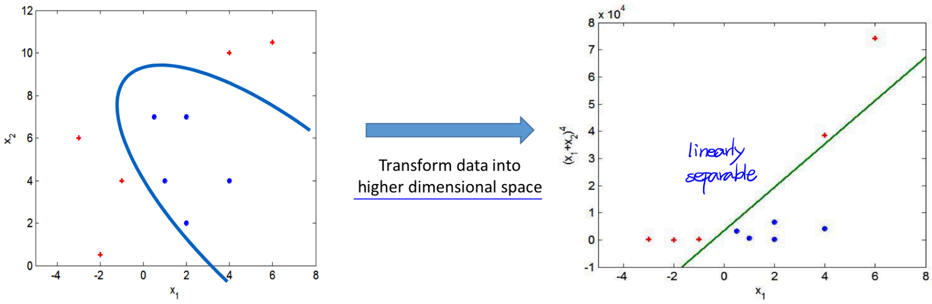
- SVM model: a list of support vectors and their coefficients α

- How C affects the model? $C\uparrow$ ··· difficult problem이 된다.

- Why transform to dual? Why not solving the primal?

3. Non-linear SVM

3-1. Non-linear SVM 개요



- Naive way → Inefficient.
  - Transform data into higher dimensional space.
  - Compute a linear function in the new feature space. (The linear boundary in the new space becomes a nonlinear in the original space.)

- SVM *kernel trick*
  - Does all of these without explicitly transforming data into higher dimensional space

3-2. Non-linear SVM using Kernel trick 정의

Dual form

- *Maximize:*

$$M(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- Subject to:

$$C \ge \alpha_i \ge 0, \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\boxed{K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)}$$

로 대체해 계산! : kernel trick

ϕ : higher dimension으로 mapping.

- Prediction on a new data $\mathbf{x}$:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}$$

dot product ⇝ cosine simularity

: scalar 반환

- Suppose $\mathbf{x} \in \mathbb{R}^2$ and $\phi(\cdot)$ is given as follows $(\mathbb{R}^2 \to \mathbb{R}^6)$

$$\phi\left(\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- A dot product in the feature space is:

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- So, no need to transform by $\phi(\cdot)$ at all if we use the following kernel function (*kernel trick*):

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$$

- It results in learning a *quadratic function*.
- A *polynomial function with degree p* will be learned if we use the following kernel function:

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^p$$

- But linear time만 소요 x·y 계산 후 결과값만 p승해 하므로.

모든 pair를 계산하므로 최종적으로 quadratic time 소요

- *Maximize:*

$$M(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

- Subject to:

$$C \geq \alpha_i \geq 0, \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

- Prediction on a new data $\mathbf{x}$:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$

- No $\phi(\cdot)$ at all.
- $K\left(\mathbf{x}_i, \mathbf{x}_j\right)$, i.e. the dot product, is just kind of similarity measure: $K\left(\mathbf{x}_i, \mathbf{x}_j\right)$ returns higher value if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar to each other.
- Data appears only as $K\left(\mathbf{x}_i, \mathbf{x}_j\right)$ (a similarity function) => There is no restriction on the form of $\mathbf{x}$, and it can be something beyond a vector (e.g. sequence, tree, graph).

3-3. Kernel functions

- Polynomial kernel:

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^p$$

→ p tuning 쉽다 (p=1, 2, ⋯ ⇒ error 체크.)

- Radial basis function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}$$

→ 이론적으로 모든 형태의 boundary 구성 가능.

- As γ moves, the boundary shape changes, and an unartful (서투른) tuning of it could result in overfitting or underfitting.
- Not all similarity measure can be used as kernel function.
  - Kernel function needs to satisfy Mercer function, i.e. the function must be "positive definite".
  - The m-by-m kernel matrix (symmetric matrix), where the (i, j)-th entry is the $K(\mathbf{x}_i, \mathbf{x}_j)$, is always positive definite.

4. SVM implementation

4-1. Hyper-parameter tuning

→ 결과 도출에도 linear time 소요되는 것으로 유명 ···LibLinear

linear SVM

- Soft margin parameter C:
  - SVM becomes "soft" as C approaches to zero.
  - SVM becomes "hard" as C goes up.
- Polynomial kernel parameter p in $K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^p$:
  - As p increases, the boundary becomes more complex.
  - Start with $p = 1$ and increase p by 1.
  - The generalization performance will stop improving at some point.
- (Radial Basis Function) RBF kernel parameter γ in $K(\mathbf{x},\mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$:
  - No good heuristics.
  - Start with a small number like 2^{-6} and multiply by 2 at each iteration up to 2^6.
  - Once you find a good range of γ, you can tune it more finely within the range.

4-2. SVM 구현 예시

- LIBSVM
  - Highly optimized implementation
  - Provide interfaces to many other languages or tools.
  - Support various types of SVM such as multi-class classification, nu-SVM, one-class SVM, etc.
  - Support very fast linear SVM, i.e. LibLinear.
- SVM-light
  - One of the earliest implementations
  - Support ranking SVM.

5. Multi-class classification SVM

5-1. Multi-class classification SVM 개요

- A multiclass classification can be reduced into a set of binary classifications.
- Two representative ways:
  1. One-to-all
  2. Pairwise coupling (or one-to-one): more popularly used in SVM

One-to-all

- For k class classification, create k binary classifiers.
  - Each binary classifier is trained from one class as positive and the others as negative.
  - Classify new object by taking the majority vote of the classifiers.
- Less number of binary classifiers will be created.
- Size of training set for each classifier is larger.

Pairwise coupling (or one-to-one)

- Create a binary classifier for each pair from k classes.
  - $\binom{k}{2}$ binary classifiers will be created.
  - Each classifier is constructed from two classes of data – one as positive and the other as negative.
  - Classify new object by taking the majority vote of the classifiers.
- More number of binary classifiers will be created.
- Size of training set for each classifier is smaller.

일반적으로 SVM에서 one-to-one의 성능↑

속도 : smaller training

정확도 : skewed data?

6. Ranking SVM

6-1. Ranking SVM 개요

- Notations: ordering 순서 비교
  - $(\mathbf{x}_i \succ \mathbf{x}_j)$: $\mathbf{x}_i$ is ranked higher than $\mathbf{x}_j$ according to some ordering.
  - $(\mathbf{x}_i, \mathbf{x}_j) \in R$: $\mathbf{x}_i$ is ranked higher than $\mathbf{x}_j$ according to an ordering R
  - $R' = \{(\mathbf{x}_i \succ \mathbf{x}_j)\}$ is a partial ordering of data given to user.
  - R^* is the desired global ordering of data which is unknown to user.

- Ranking model
  - Input: a set of partial orders R'
  - Output: $f(\mathbf{x})$ such that $f(\mathbf{x}_i) > f(\mathbf{x}_j)$ for all $(\mathbf{x}_i, \mathbf{x}_j) \in R'$ ⇒ NDCG, recall로 measure
    scoring
- Linear function: $f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}$
  - $\forall(\mathbf{x}_i, \mathbf{x}_j) \in R'$: $f(\mathbf{x}_i) > f(\mathbf{x}_j) \Leftrightarrow \mathbf{w}\cdot\mathbf{x}_i > \mathbf{w}\cdot\mathbf{x}_j$
  - If there exists such $\mathbf{w}$, then R' is *linearly rankable.*
- Objective considering generalization
  - Input: a set of partial orders R' ($\subset R^*$)
  - Output: $f(\mathbf{x})$ such that $f(\mathbf{x}_i) > f(\mathbf{x}_j) \Leftrightarrow \mathbf{w}\cdot\mathbf{x}_i > \mathbf{w}\cdot\mathbf{x}_j$ for all $(\mathbf{x}_i, \mathbf{x}_j) \in R^*$ (not R')

"We want to train f (or compute $\mathbf{w}$) from partial orders R' ($\subset R^*$), such that f is concordant (일치하는) with R' and also generalize well beyond R' to order unseen data with respect to R^*"

6-2. Ranking SVM 정의

- Minimize:

$$L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C\left(\sum_{i=1}^{m} \xi_i\right)$$

- Subject to:

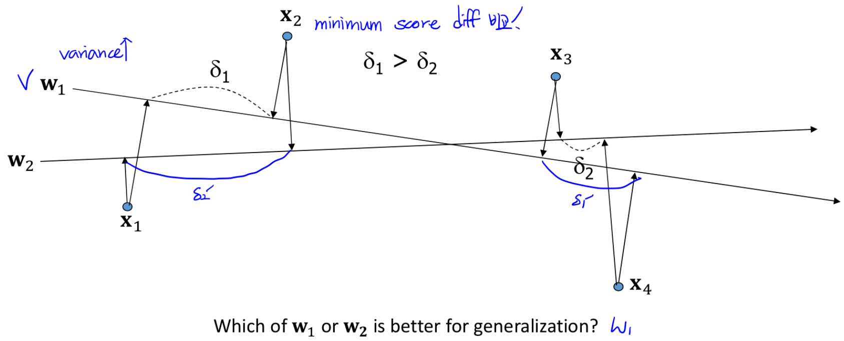
$$\forall(\mathbf{x}_i, \mathbf{x}_j) \in R': \mathbf{w}\cdot\mathbf{x}_i \geq \mathbf{w}\cdot\mathbf{x}_j + \underbrace{1 - \xi_{ij}}_{\text{margin (ranking score의 차이가 1이상)} + \varepsilon} \quad (\xi_{ij} \geq 0)$$

i.e.

$$\forall(\mathbf{x}_i, \mathbf{x}_j) \in R': \mathbf{w}\cdot(\mathbf{x}_i - \mathbf{x}_j) \geq 1 - \xi_{ij} \quad (\xi_{ij} \geq 0)$$

$$\forall(\mathbf{x}_i, \mathbf{x}_j) \in R': \mathbf{w}\cdot\mathbf{x}_i - \mathbf{w}\cdot\mathbf{x}_j \geq 1 - \xi_{ij} \quad (\xi_{ij} \geq 0)$$

$$(\mathbf{x}_1 \succ \mathbf{x}_2 \succ \mathbf{x}_3 \succ \mathbf{x}_4) \in R'$$



Which of $\mathbf{w}_1$ or $\mathbf{w}_2$ is better for generalization? w_1

minimum ranking score difference를 최대화해야한다

- Can we apply the kernel trick to learn nonlinear ranking functions? (O)
- RankSVM implementation => SVM light