Support Vector Machine
for Classification and Regression

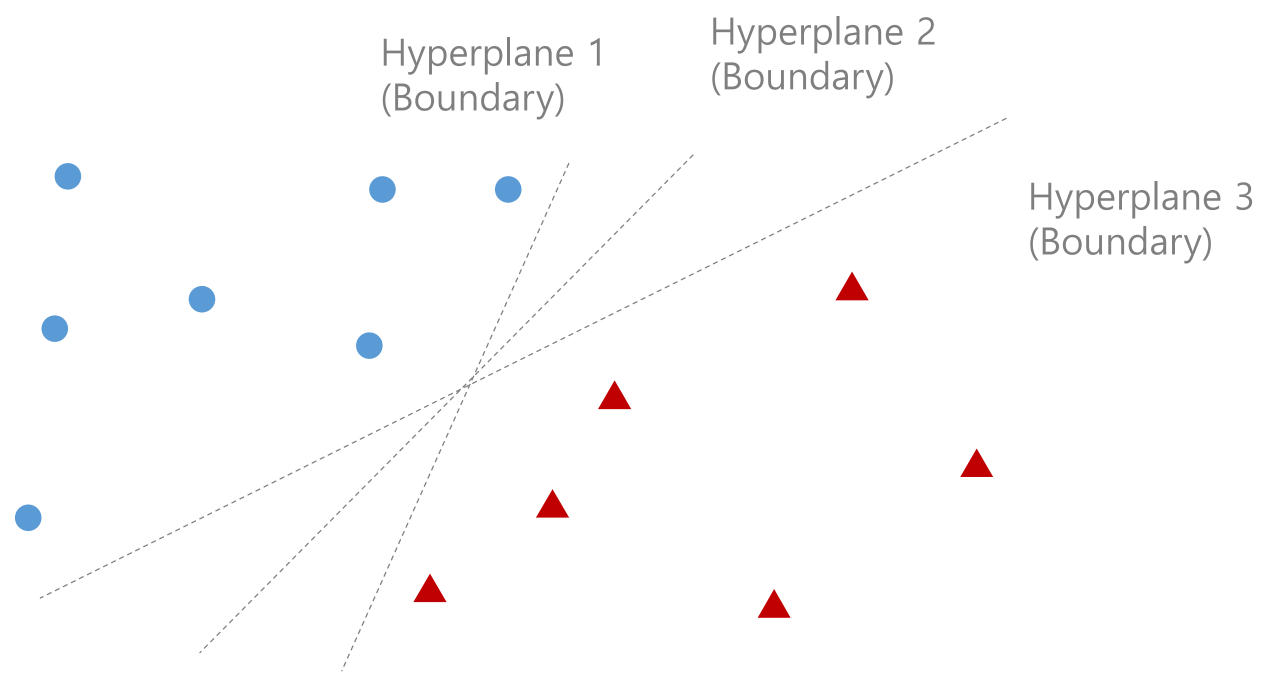
Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

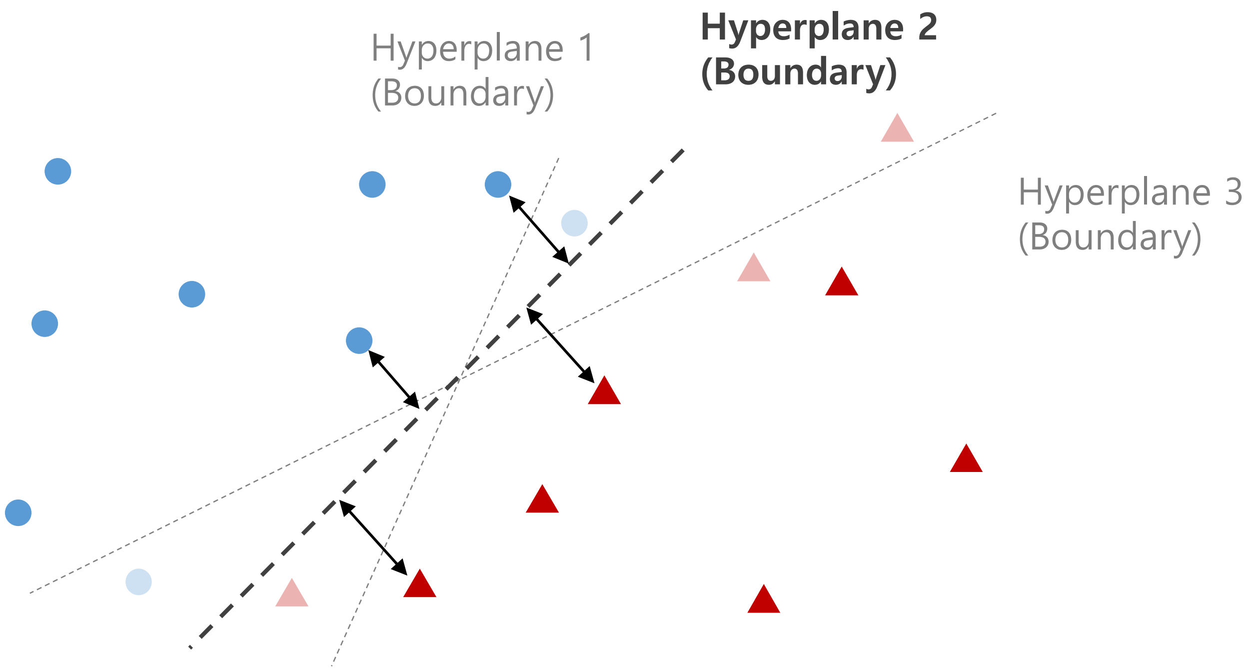
Support Vector Machine

- 데이터를 구분하는 경계면은 여러 개가 만들어질 수 있습니다. 모델의 품질 기준을 정확도로 이용할 경우 아래 세 경계면의 품질이 동일합니다.



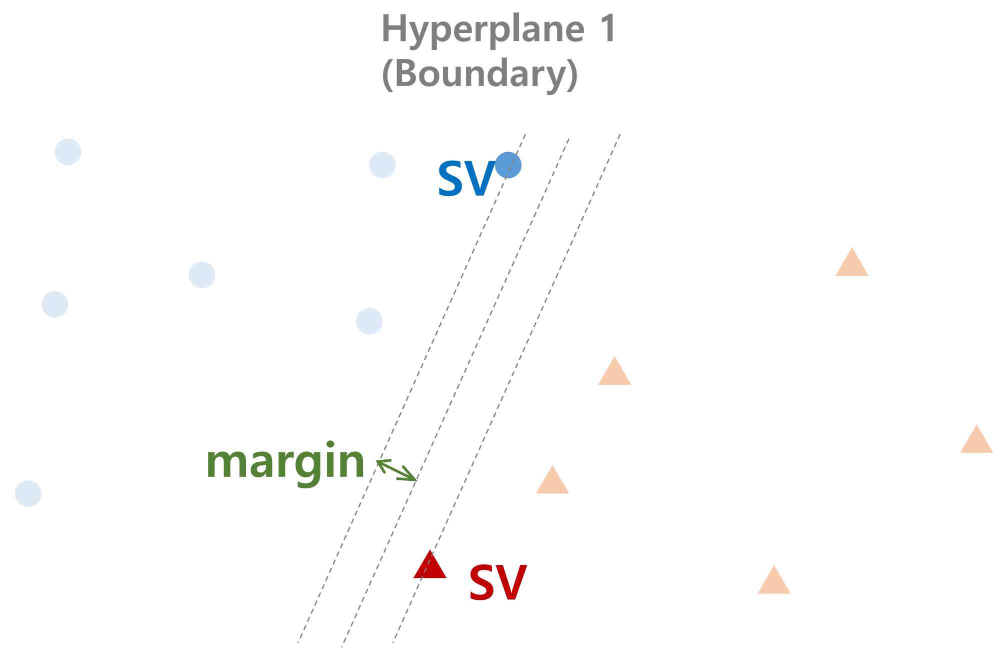
Support Vector Machine

- 새로운 데이터가 위치할 가능성을 고려한다면 학습데이터와의 거리가 충분히 떨어진 hyper-plane 2 가 적절합니다. SVM 은 다양한 경계면 중 학습데이터와의 거리가 충분히 떨어진 경계면을 탐색합니다.



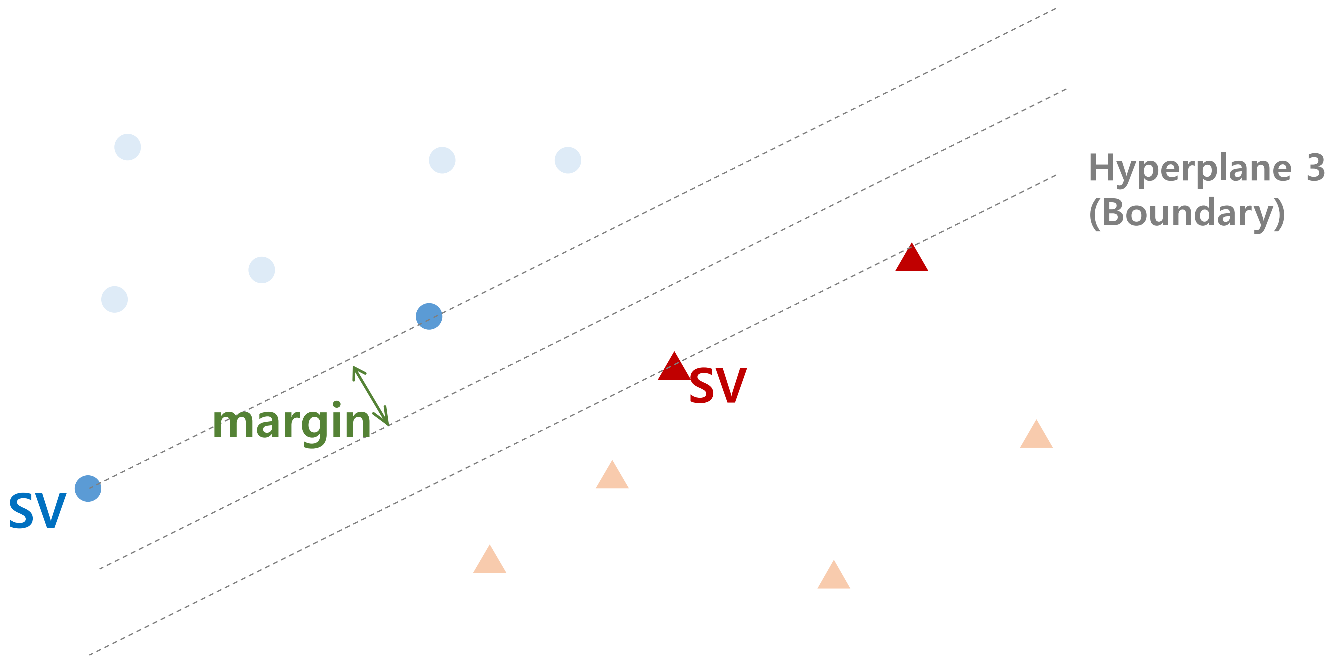
Support Vector Machine

- 클래스 별로 경계면에 가장 인접한 점을 support vectors (SV) 라 하며, 경계면과 SV 와의 거리를 margin 이라 합니다.



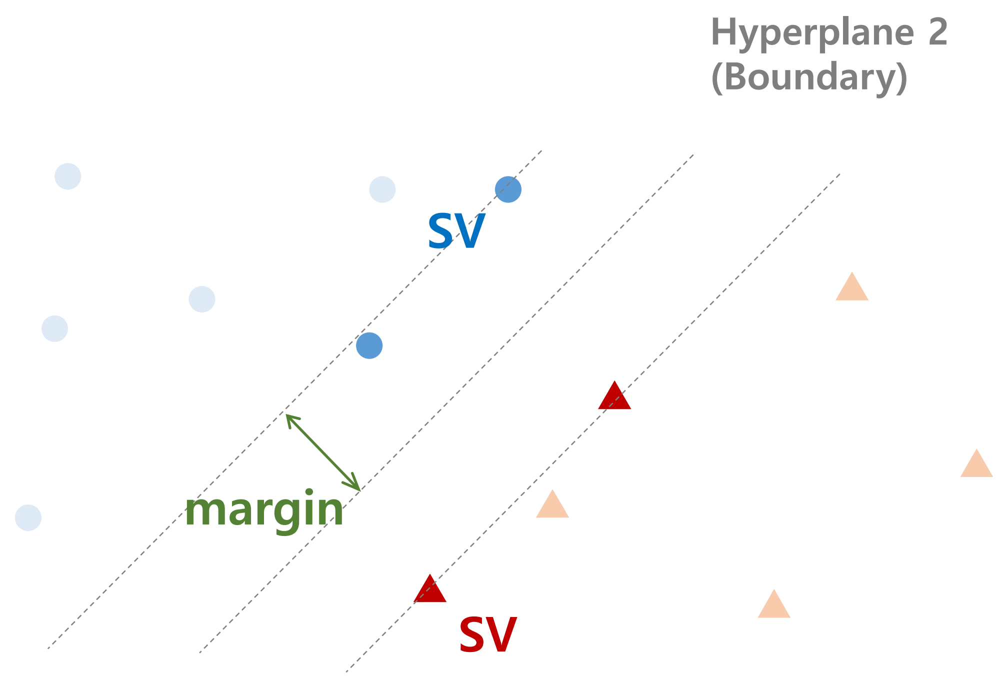
Support Vector Machine

- 여러 개의 경계면 중에서 margin이 가장 큰 경계면을 선택합니다.



Support Vector Machine

- 경계면 2의 margin 이 가장 큽니다. 이처럼 margin 이 가장 큰 경계면을 "maximal margin hyper-plane" 이라 합니다. SVM 은 "maximal margin classifier" 입니다.



Margin

수식의 이해가 필수는 아닙니다.
개념이 이해 되셨다면 유도과정은
넘어가도 괜찮습니다.

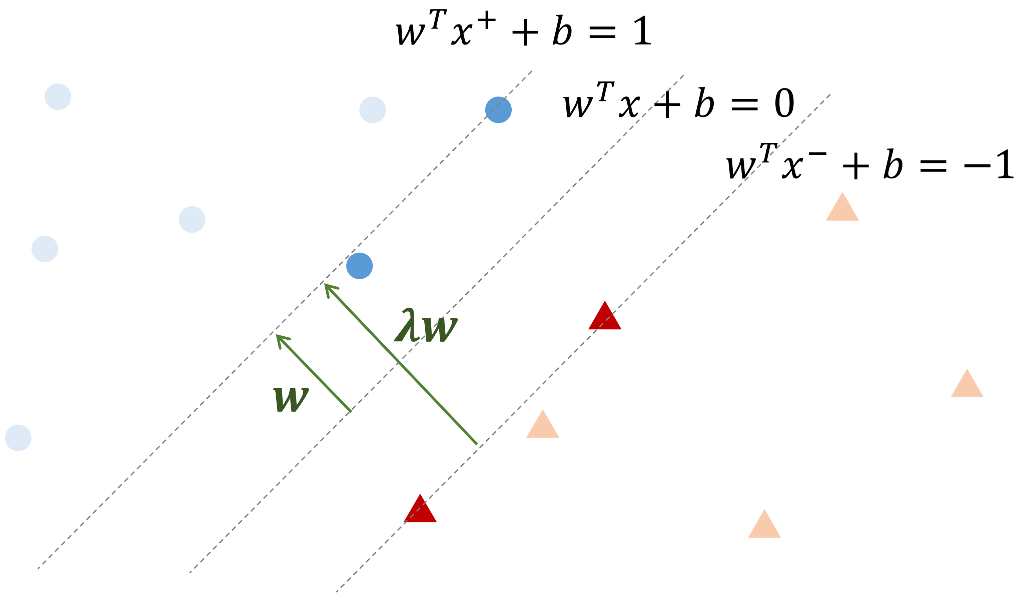
- $w^T x^- + b = -1$ 에서 w 방향으로 λ 만큼 이동하면 $w^T x + b = 1$ 에 도착합니다.
  - $x^+ = x^- + \lambda w$
  - $w^T x^+ + b = 1$

    $\rightarrow w^T(x^- + \lambda w) + b = 1$

    $\rightarrow w^T x^- + b + \lambda w^T w = 1$

    $\rightarrow \lambda w^T w = 2$

    $\rightarrow \lambda = \frac{2}{|w|_2^2}$



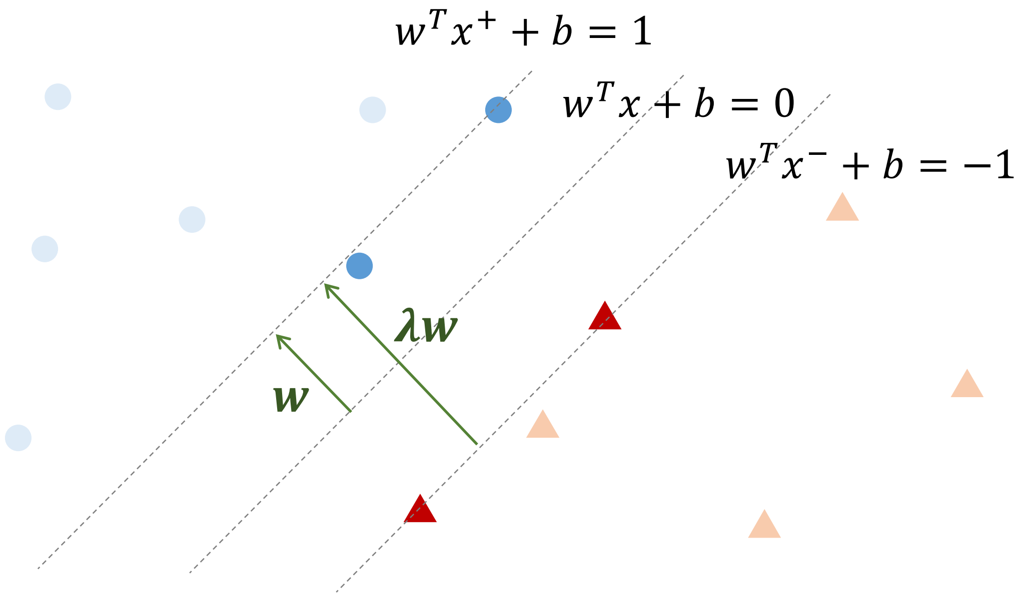
Margin

수식의 이해가 필수는 아닙니다.
개념이 이해 되셨다면 유도과정은
넘어가도 괜찮습니다.

- 두 클래스 간 경계의 넓이 (margin) 을 최대화하는 것은 다음처럼 표현할 수 있습니다.
  - maximize $\lambda = \frac{2}{|w|_2^2}$

    s.t. $\quad w^T x^+ + b \geq 1 \;\&\; w^T x^- + b \leq 1$
  - $\rightarrow$ minimize $\frac{1}{2}|w|_2^2$

    s.t. $\quad y_i(w^T x_i + b) \geq 1$



Solve objective

수식의 이해가 필수는 아닙니다.
개념이 이해 되셨다면 유도과정은
넘어가도 괜찮습니다.

- 제약조건이 포함된 목적함수의 최소값을 탐색하기 위하여 라그랑즈 승수법 (Lagrange Multiplier method) 을 이용합니다.

$$\text{minimize } L(w,b) = \frac{1}{2}|w|_2^2$$

$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1$$

$$\text{minimize } L(w,b,\alpha) = \frac{1}{2}|w|_2^2 - \sum_i \alpha_i (y_i(w^T x_i + b) - 1)$$

$$\text{s.t.} \quad a_i \geq 0$$

$$\frac{dL(w,b,\alpha)}{dw} = w - \sum_i \alpha_i y_i x_i = 0 \quad \rightarrow w^* = \sum_i \alpha_i y_i x_i$$

$$\frac{dL(w,b,\alpha)}{dw} = \sum_i \alpha_i y_i = 0$$

$$\text{s.t.} \quad a_i \geq 0$$

Solve objective

수식의 이해가 필수는 아닙니다.
개념이 이해 되셨다면 유도과정은
넘어가도 괜찮습니다.

- 제약조건이 포함된 목적함수의 최소값을 탐색하기 위하여 라그랑즈 승수법 (Lagrange Multiplier method) 을 이용합니다.

$$\text{minimize } L(w,b,\alpha) = \frac{1}{2}|w|_2^2 - \sum_i \alpha_i (y_i(w^T x_i + b) - 1)$$

$$\text{s.t.} \quad a_i \geq 0$$

$$\frac{1}{2}|w|_2^2 = \frac{1}{2}\sum_i w^{*T}\left(\sum_j \alpha_j y_j x_j\right) = \frac{1}{2}\sum_i \alpha_i y_i x_i \sum_j \alpha_j y_j x_j$$

$$= \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$(a+b+c)(d+e+f) = ad + ae + af + bd + be + bf + cd + ce + cf$$

$$\sum_i \alpha_i (y_i(w^T x_i + b) - 1) = \sum_i \alpha_i y_i w^T x_i + b\sum_i \alpha_i y_i - \sum_i \alpha_i$$

$$= \sum_i \alpha_i y_i \left(\sum_j \alpha_j y_j x_j\right)^T x_i - \sum_i \alpha_i$$

$$= \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_i \alpha_i$$

$$\sum_i \alpha_i y_i = 0$$

Solve objective

수식의 이해가 필수는 아닙니다.
개념이 이해 되셨다면 유도과정은
넘어가도 괜찮습니다.

- 제약조건이 포함된 목적함수의 최소값을 탐색하기 위하여 라그랑즈 승수법 (Lagrange Multiplier method) 을 이용합니다.

$$\text{minimize } L(w, b, \alpha) = \frac{1}{2}|w|_2^2 - \sum_i \alpha_i (y_i(w^T x_i + b) - 1)$$

$$\text{s.t.} \quad a_i \geq 0$$

$$\frac{1}{2}|w|_2^2 = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\sum_i \alpha_i (y_i(w^T x_i + b) - 1) = \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_i \alpha_i$$

$$\text{minimize } L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t.} \quad a_i \geq 0$$

$$a_i = 0 \text{ or } y_i(w^T x_i + b) - 1 = 0$$

Solve objective

- w^* 와 b^* 가 학습데이터 x_i 의 선형조합 (가중합) 입니다.
- $a_i > 0$ 인 점은 $y_i(w^T x_i + b) - 1 = 0$ 입니다. 경계면에서 margin 만큼 떨어진 점입니다. 이를 support vectors 라 합니다.

$$\text{minimize } L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t.} \quad a_i \geq 0$$

$$a_i = 0 \text{ or } y_i(w^T x_i + b) - 1 = 0$$

$$w^* = \sum_i \alpha_i y_i x_i$$

$$b^* = 1 - w^T x_i^+$$

Support Vector Machine

- 직선의 경계면의 패러매터 w^*, b^* 를 얻었기 때문에 새로운 벡터 q 에 대하여 선형회귀식인 decision function, $f(q)$ 를 얻을 수 있습니다.

$$w^* = \sum_i \alpha_i y_i x_i$$

$$b^* = 1 - w^T x_i^+$$

$$f(q) = {w^*}^T q + b^* > 0$$

Support Vector Machine

- 새로운 벡터 q 가 입력되면 다음의 식을 이용하여 클래스를 판단합니다. $\alpha_i = 0$ 인 점은 영향을 주지 않습니다. SVM 은 학습 후 SV 와 이에 해당하는 a_i 를 모델에 저장합니다.
  - 벡터 q 와 SV 와 내적을 한 뒤, coefficient vector αy 를 내적한 선형판별식입니다.

$$f(q) = \sum_{i \in SV} \alpha_i y_i x_i^T q + b > 0$$

$$= (\alpha y)^T (SV^T q) + b$$

Support vectors 의 label y 와
이에 대한 가중치 $\boldsymbol{\alpha_i}$

Support vectors 와 query 와의
Inner product 에 의한 유사도 벡터 (representation)

Support Vector Machine

- SVM 은 q 와 SV 와의 내적을 이용하여 $|SV|$ 차원의 벡터로 representation 을 변환한 뒤, αy 벡터와 내적을하여 점수를 계산하는 선형판별을 수행합니다.
- SVM 의 $f(q)$ 결과는 확률이 아닌 점수입니다.

$$\text{Logistic}: f(q) = \frac{1}{1 + \exp(-\beta^T q)} > 0.5$$

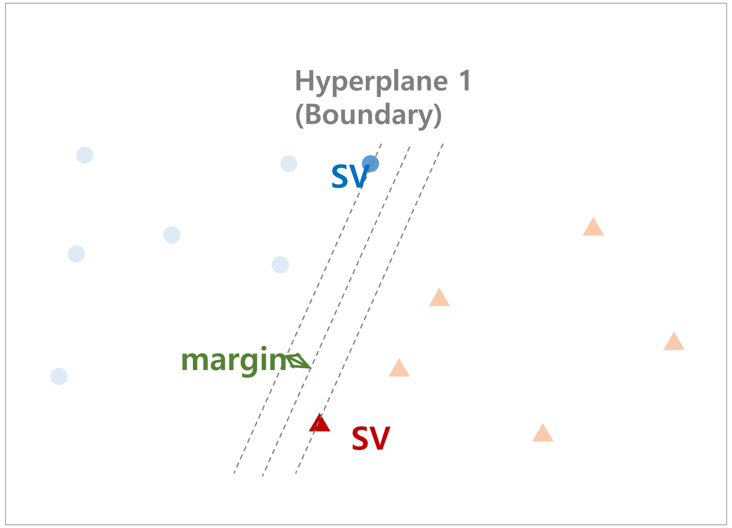
$$\text{SVM}: \ f(q) = \sum_{i \in SV} \alpha_i y_i x_i^T q + b > 0$$

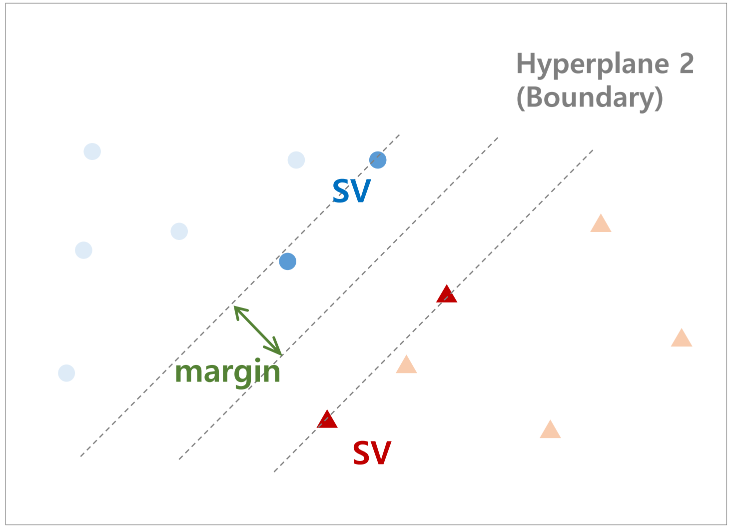
Support vectors 의 label y 와
이에 대한 가중치 $\boldsymbol{\alpha_i}$

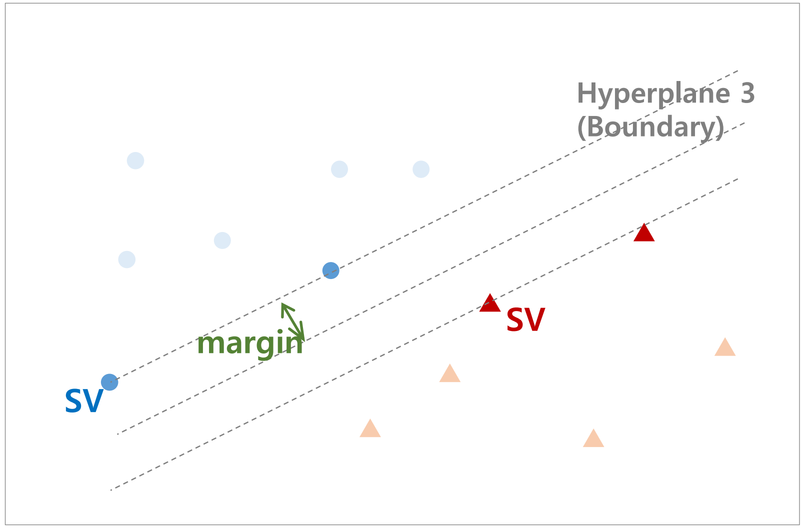
Support vectors 와 query 와의
Inner product 에 의한 유사도 벡터
(representation)

Support Vector Machine

- SVM 은 경계면, support vectors 선정, coefficient a_i 의 학습을 동시에 수행합니다.

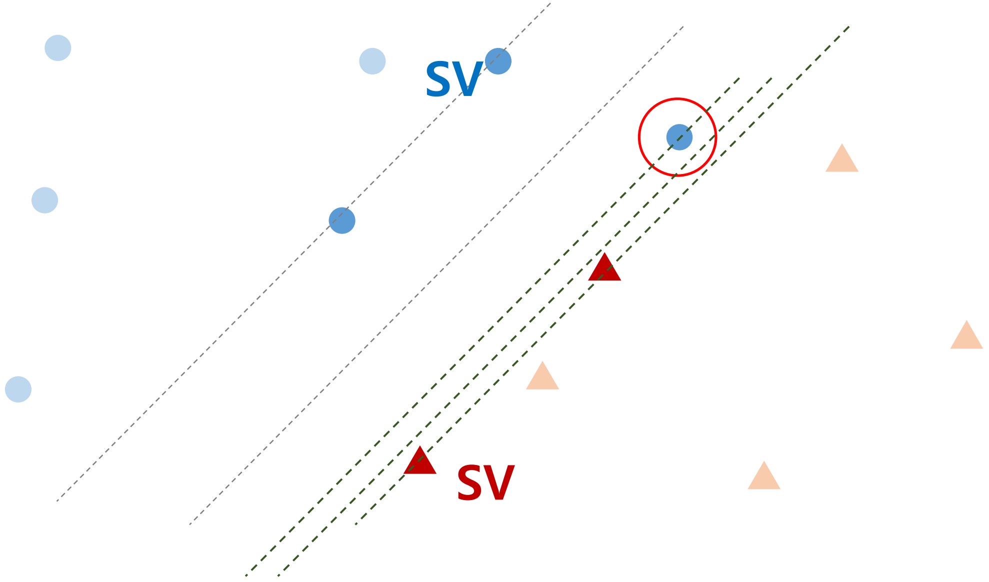






Soft margin

- Linear separable 인 경계면은 outliers 에 민감합니다.
  - 한 점만 무시한다면 margin 이 매우 커질 수 있습니다.



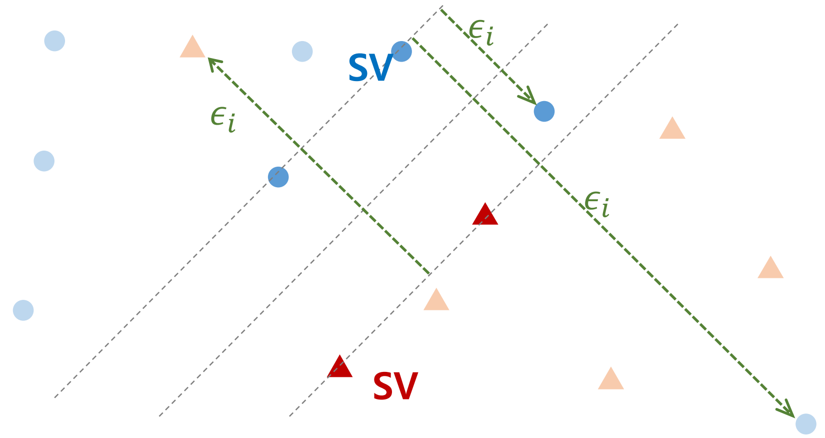
Soft margin

- 오류 ϵ_i 를 포함하는 새로운 비용함수를 정의할 수 있습니다.
  - C 는 ridge regression 처럼 regularization 역할을 합니다.
  - C 가 클수록 margin 이 작은, 오류가 덜 허용되는 경계면이 학습됩니다.

$$\text{minimize } L(w,b) = \frac{1}{2}|w|_2^2 + C\sum_i \epsilon_i$$

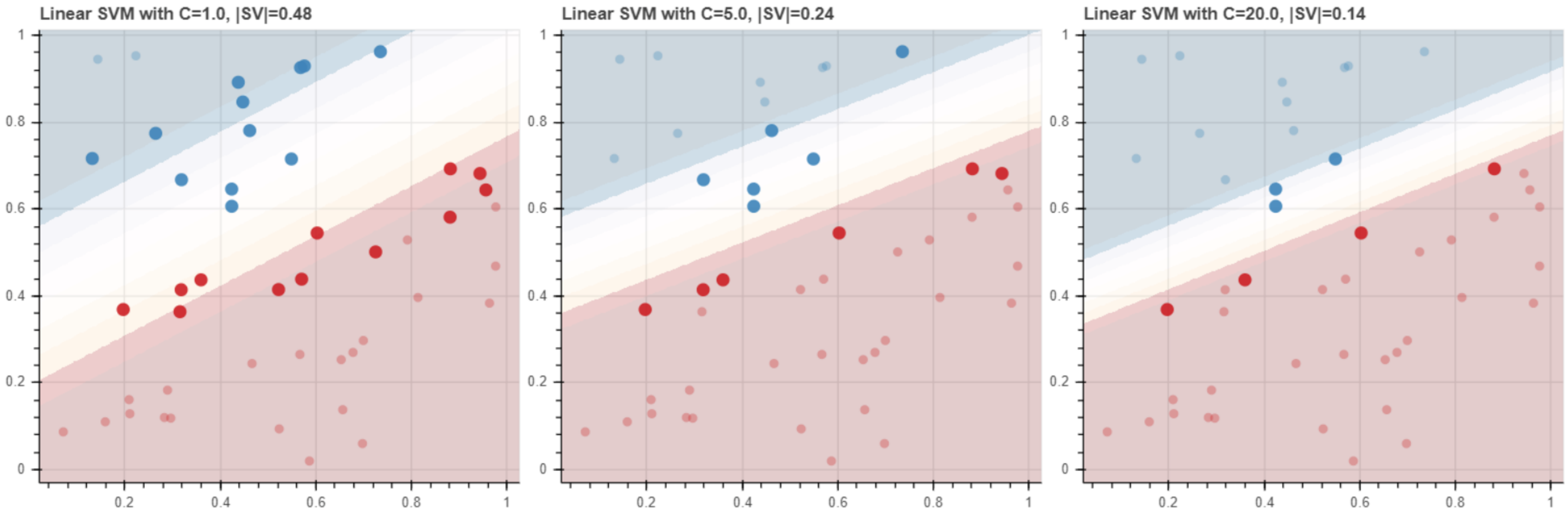
$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \epsilon_i$$

$$\epsilon_i \geq 0$$



Soft margin

- C 가 작으면 학습데이터의 과적합을 방지할 수 있습니다.



Soft margin

- C 를 조절하여 학습데이터의 과적합을 방지할 수 있습니다.

sklearn.svm.SVC

class sklearn.svm.SVC(*C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

Parameters:

C : float, optional (default=1.0)

Penalty parameter C of the error term.

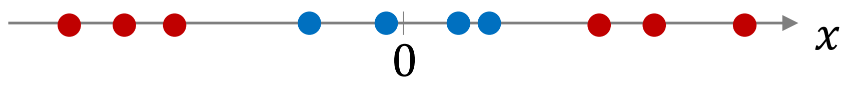
kernel : string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is

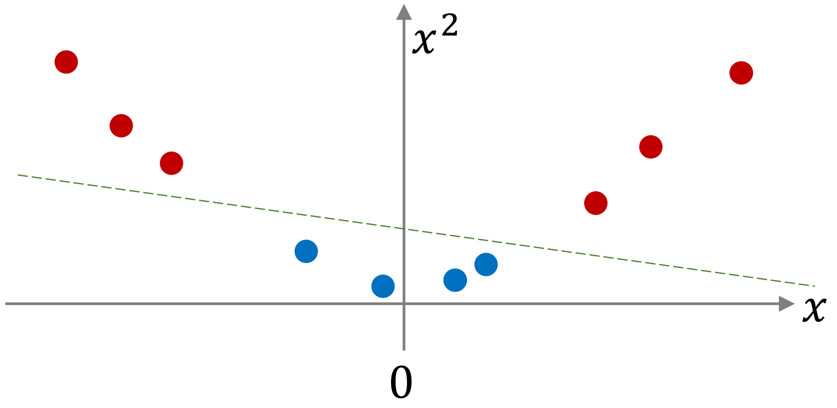
Polynomial kernel SVM

- 선형분리가 않되는 1차원 데이터를 2차원 polynomial features 로 표현하면 선형분리가 이뤄집니다. 데이터를 고차원 벡터로 변환한 뒤 SVM 모델을 학습합니다.

Linear inseparable with x



Linear separable with (x, x^2)



Polynomial kernel SVM

- SVM 은 모든 학습데이터 x_i, x_j 간의 내적을 이용하여 w, b, α 를 학습합니다. x_i 를 $\Phi(x_i)$, feature vectors 로 변환하여도 이는 동일합니다.

$$\text{minimize } L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t.} \quad a_i \geq 0$$

$$a_i = 0 \text{ or } y_i(w^T x_i + b) - 1 = 0$$

$$f(q) = \sum_{i \in SV} \alpha_i y_i x_i^T q + b > 0$$

$$\text{minimize } L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$

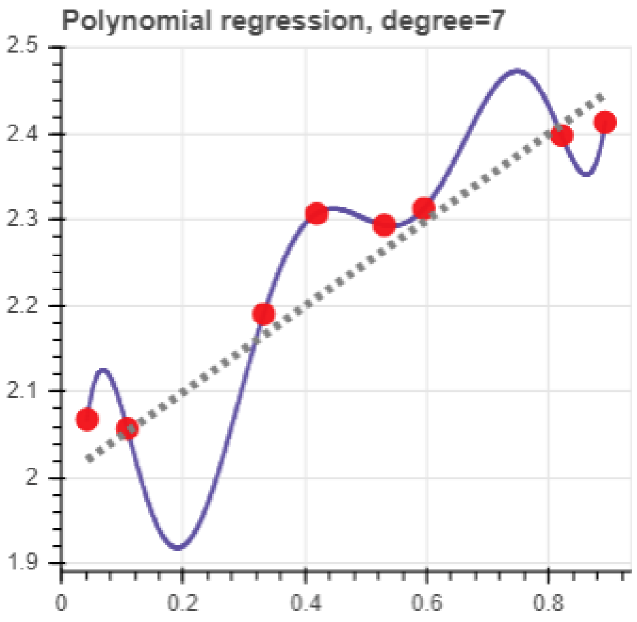
$$\text{s.t.} \quad a_i \geq 0$$

$$a_i = 0 \text{ or } y_i(w^T \Phi(x_i) + b) - 1 = 0$$

$$f(q) = \sum_{i \in SV} \alpha_i y_i \Phi(x_i)^T \Phi(q) + b > 0$$

Polynomial kernel SVM

- Feature representation 을 바꾸면 선형 모델이 잘 맞을 가능성이 높습니다. Polynomial feature space 에서 정확히 일치하는 ($R^2 = 1$) 회귀식은 원 공간에서 곡선이었습니다.
- 고차원 공간의 직선/평면에 해당하는 저차원 공간의 점들은 곡선/곡면일 수 있습니다.

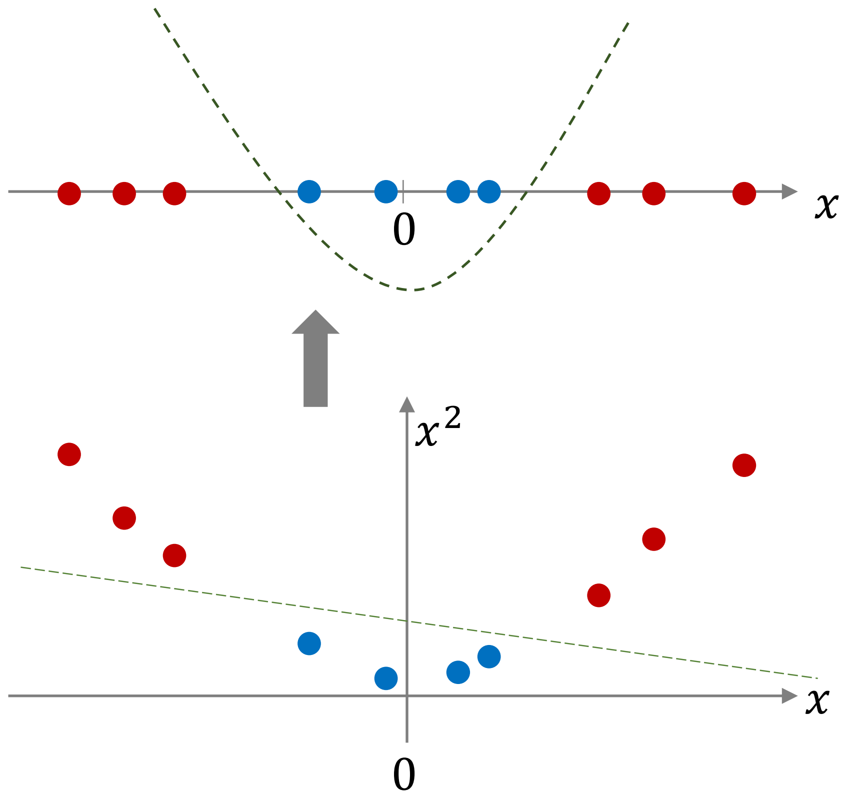


Polynomial kernel SVM

- 2 차원 공간에서의 직선의 경계면은 1차원 공간에서 곡선의 경계면일 수 있습니다.

Linear inseparable with x

Linear separable with (x, x^2)



Polynomial kernel SVM

- Feature space 에서의 내적 $\Phi(x_i)^T\Phi(x_j)$ 을 커널함수 $K(x_i, x_j)$ 라 정의합니다.
  - Polynomial kernel : $K(x_i, x_j) = (x_i^T x_j + r)^d$

$$\text{minimize } L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \Phi(x_i)^T\Phi(x_j)$$

$$\text{s.t.} \quad a_i \geq 0$$

$$a_i = 0 \text{ or } y_i(w^T\Phi(x_i) + b) - 1 = 0$$

$$f(q) = \sum_{i\in SV} \alpha_i y_i \Phi(x_i)^T\Phi(q) + b > 0$$

$$f(q) = \sum_{i\in SV} \alpha_i y_i K(x_i, q) + b > 0$$

Polynomial kernel SVM

- 1 차원 x_i, x_j 에 $r = \frac{1}{2}, d = 2$ 을 적용하면 3차원 feature space 에서의 내적과 같습니다.
- $x_i \rightarrow \left(x_i^2, x_i, \frac{1}{2}\right)$ 변환을 하지 않고도 $\Phi(x_i)^T \Phi(x_j)$ 의 값을 계산할 수 있습니다.

$$K(x_i, x_j) = \left(x_i \times x_j + \frac{1}{2}\right)^2 = x_i^2 x_j^2 + x_i x_j + \frac{1}{4} = \left(x_i^2, x_i, \frac{1}{2}\right)^T \left(x_j^2, x_j, \frac{1}{2}\right)$$

Kernel SVM

수식의 이해가 필수는 아닙니다.
개념이 이해 되셨다면 정의는
넘어가도 괜찮습니다.

- Kernel functions 은 Mercer's theorem 을 만족하는 함수입니다.

A symmetric function $K(x, y)$ can be expressed as an inner product

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

for some ϕ if and only if $K(x, y)$ is positive semidefinite, i.e.

$$\int K(x, y) g(x) g(y) dx dy \geq 0 \qquad \forall g$$

or, equivalently:

$$\begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots \\ K(x_2, x_1) & \ddots & \\ \vdots & & \end{bmatrix} \text{ is psd for any collection } \{x_1 \ldots x_n\}$$

Therefore you can either explicitly map the data with a ϕ and take the dot product, or you can take any kernel and use it right away, without knowing nor caring what ϕ looks like. For example:

- Gaussian Kernel: $K(x, y) = e^{\frac{1}{2}||x-y||^2}$
- Spectrum Kernel: count the number of substrings in common. It is a kernel since it is a dot product between vectors of indicators of all the substrings.

https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf

Kernel SVM

- 자주 이용되는 커널 함수는 다음과 같습니다.
  - Linear kernel : $K(x_i, x_j) = x_i^T x_j$
  - Polynomial kernel : $K(x_i, x_j) = (x_i^T x_j + r)^d$
  - Radial Basis (Gaussian) kernel : $K(x_i, x_j) = \exp\left(-\gamma |x_i - x_j|^2\right)$
  - Sigmoid kernel : $K(x_i, x_j) = \tanh(\gamma(x_i^T x_j) + \theta)$

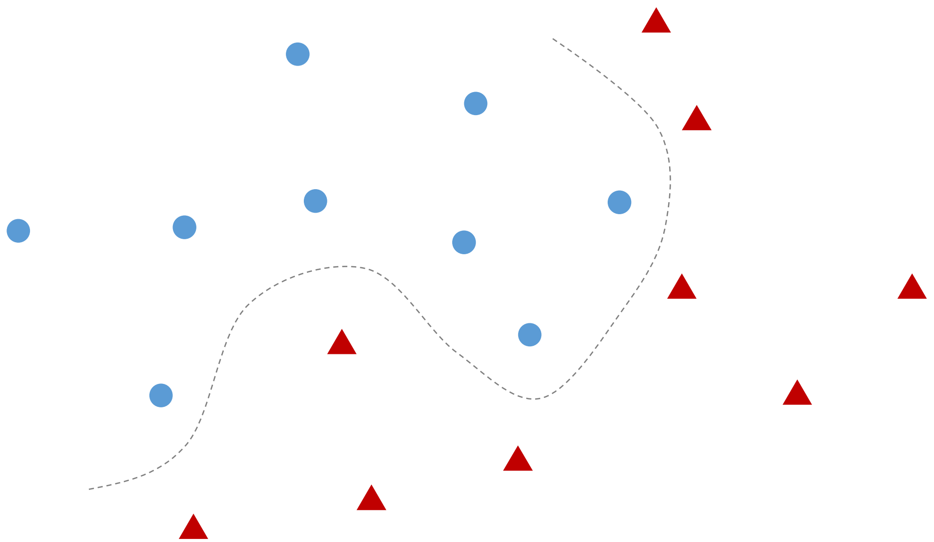
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

RBF kernel SVM

- Kernel SVM 의 판별식은 유사도를 정의하는 부분만 바뀝니다.

$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \, K(x_i, q) + b$$

$$K(x_i, q) = exp\left(-\gamma |x_i - q|^2\right)$$

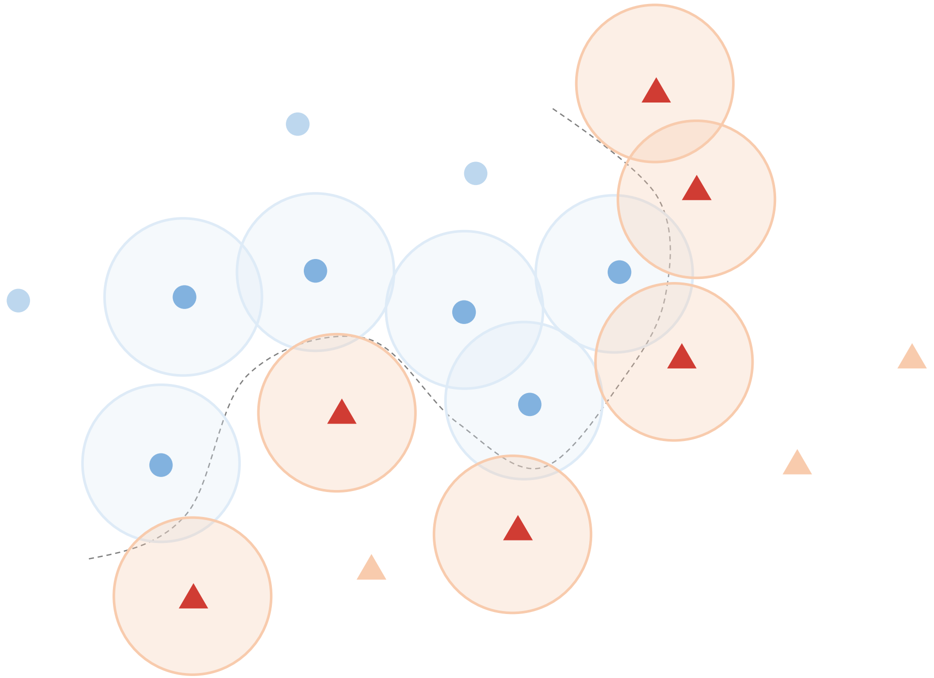


RBF kernel SVM

- RBF kernel 은 q 가 SV 와 Gaussian 으로 얼마나 가까운지를 정의합니다.

$$\boldsymbol{f(q) = \sum_{i \in SV} \alpha_i \, y_i \, K(x_i, q) + b}$$

$$\boldsymbol{K(x_i, q) = exp\left(-\gamma |x_i - q|^2\right)}$$

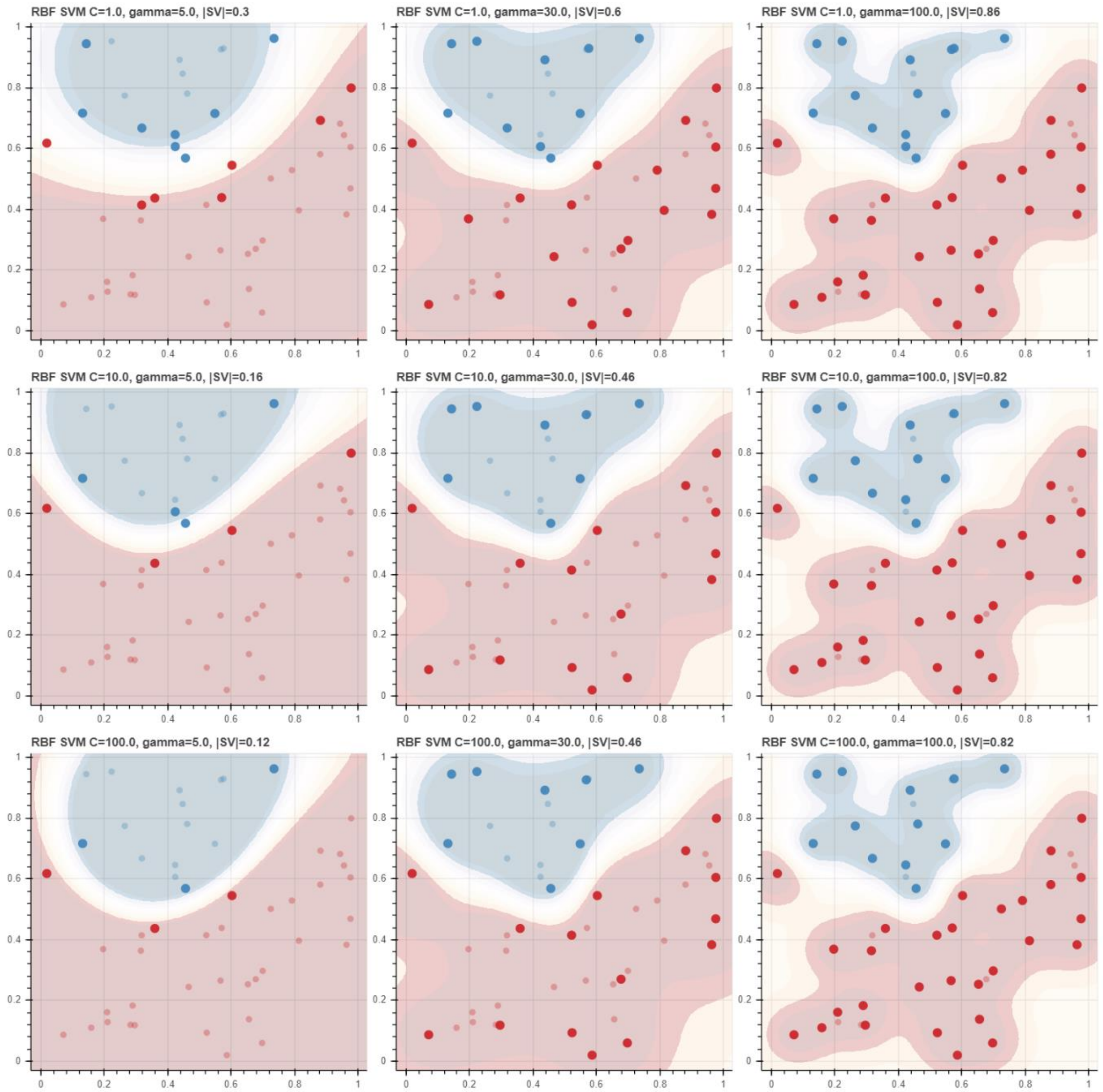


RBF kernel SVM

- RBF kernel 은 무한 차원 공간에서의 벡터 내적으로 알려져 있습니다. Feature space 가 무한차원이기 때문에 선형 분리가 가능한 공간이 만들어질 가능성이 높습니다.
  (자세한 내용은 appendix 의 Tayler expansion 을 살펴보시기 바랍니다)

RBF kernel SVM

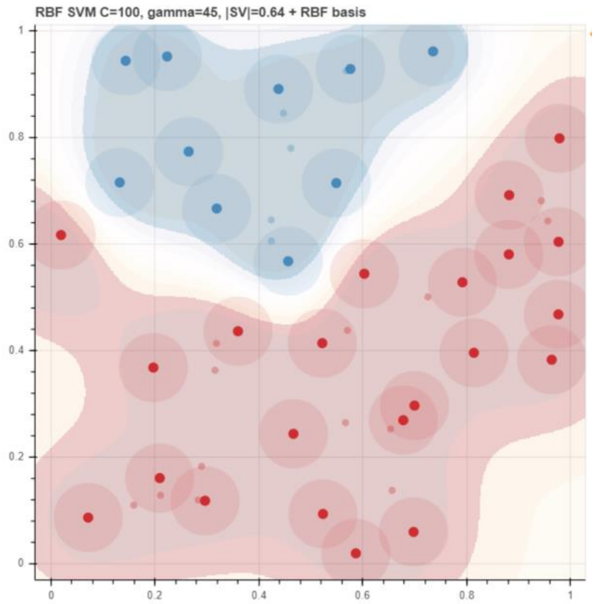
- C 와 γ 를 조절하면 SV 의 개수와 경계면의 모양을 조절할 수 있습니다.
- γ 가 작으면 많은 학습데이터를 외우는 현상이 발견됩니다.
- 데이터의 분포가 단순하고 γ 가 크다면 적은 수의 데이터로 경계면을 잘 표현할 수 있습니다.



RBF kernel SVM

- 경계면은 radial basis 로 표현된 새로운 feature representation 과 와 αy 의 내적입니다.
- SV 와 가까이 위치하지 않더라도 $f(q)$ 를 이용하여 판별은 가능합니다.
- 하지만 SV 와 매우 멀리 떨어진 점은 $f(q)$ 의 크기가 작을 가능성이 높습니다.

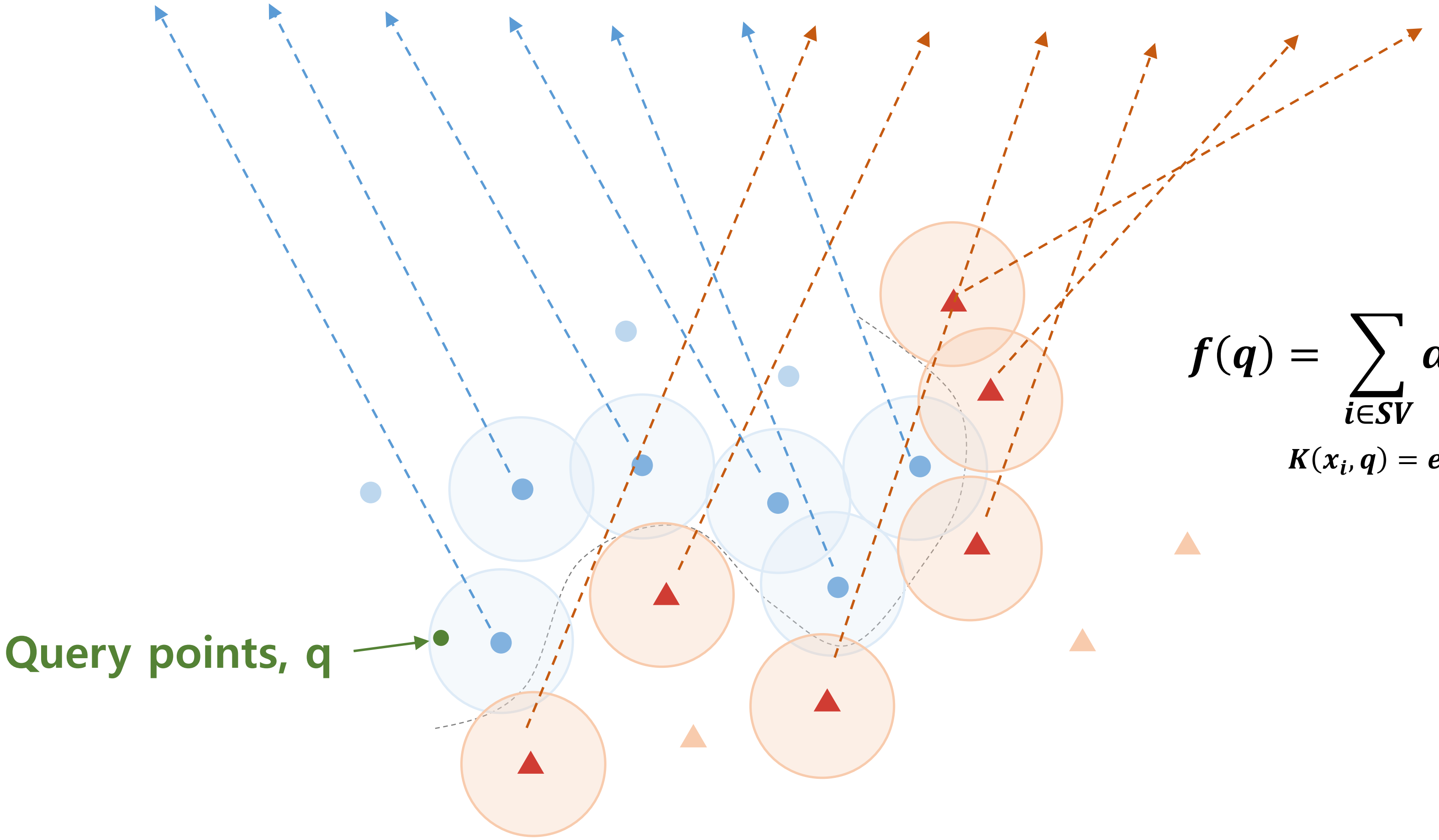
$$\boldsymbol{f(q)} = \sum_{i \in SV} \boldsymbol{\alpha_i\, y_i\, K(x_i, q) + b}$$



RBF kernel SVM

Kernel vector(?) ~= Similarity vector(?), $K(x_i, q)$

SV_1	SV_2	SV_3	SV_4	SV_5	SV_6	SV_7	SV_8	SV_9	SV_{10}	SV_{11}	SV_{12}
0.3	0.0001	0	0	0	0	0.0001	0	0	0	0	0

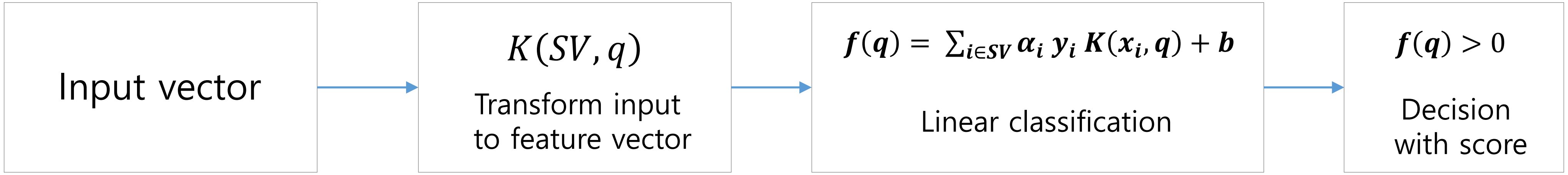


$$f(q) = \sum_{i \in SV} \alpha_i \, y_i \, K(x_i, q) + b$$

$$K(x_i, q) = exp\left(-\gamma |x_i - q|^2\right)$$

Support Vector Machine

- SVM 은 1 layer neural network 처럼 한 번 feature vector 를 변환한 뒤, 선형 판별을 수행합니다. 그 결과값은 확률이 아니며 판별 점수입니다.



Support Vector Machine

- SVM 은 학습 시 support vectors 가 아닌 점들을 학습 품질 평가에 이용하지 않습니다. Margin 이 크고 error 의 합이 작기를 원합니다. 하지만 Softmax 는 모든 학습데이터를 이용하여 likelihood (NLL loss) 를 평가합니다.
- 학습데이터의 개수가 적을 때에는 대체로 SVM 계열의 성능이 좋다고 알려져 있습니다.

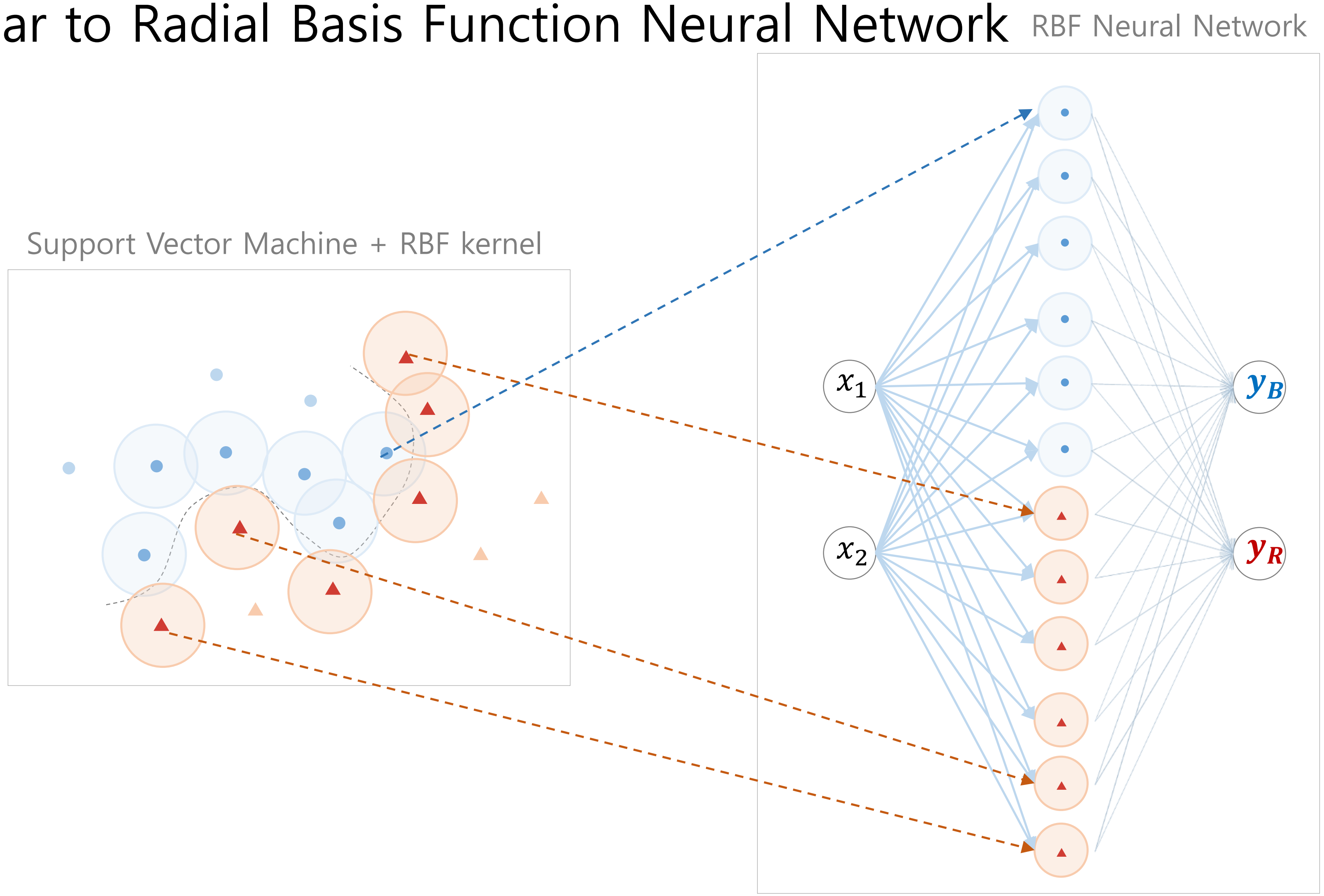
Support Vector Machine

- Sparse data 의 경우 공통된 features 의 유무를 유사도로 이용하는 linear kernel 이 RBF kernel 보다 좋은 성능, 효율성을 보인다고 알려져 있습니다.

y	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
0	5	3													
0	3	2		5		1			2						
0	2	4		4											
0	5		2				4	5	3						
0	1		1		2										
0	4			1											
1	2								2					2	
1	3							5					4	4	
1	5								1	1		3			
1	1								2			2			3
1	3								4		1		2	1	1
1	2								4				1		2

SVM as neural network

- Similar to Radial Basis Function Neural Network



Multiclass classification using SVM

- SVM 은 이진 분류 (binary classification) 만을 지원합니다. 클래스의 개수가 3 이상일 경우, 두 가지 전략을 이용할 수 있습니다.
  - one-vs.-one (ovo) :

    매 클래스마다 SVM 모델을 학습합니다. $\frac{\#c(\#c-1)}{2}$ 개의 모델을 학습합니다.
  - one-vs.-others (ovr) :

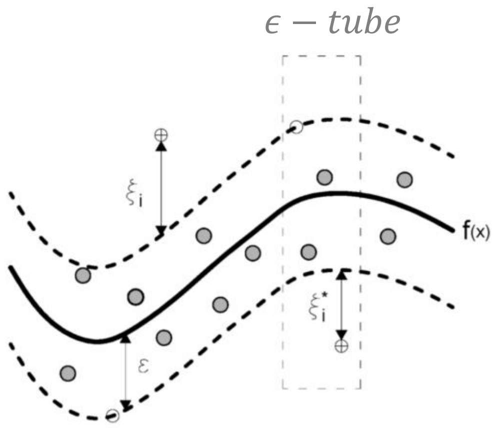
    한 클래스와 나머지 클래스를 구분하는 $\#c$ 개의 모델을 학습합니다.

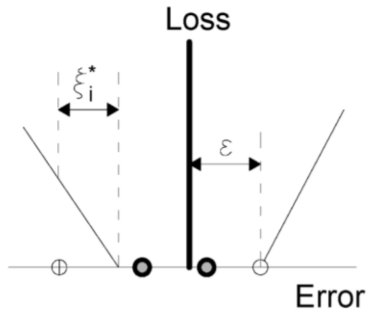
Support Vector Regression

- SVM 은 분류 뿐 아니라 회귀 문제에도 이용될 수 있습니다. 분류 모델을 Support Vector Classifier (SVC) 라 하며, 회귀 모델을 Support Vector Regression (SVR) 이라 합니다.

$$f(q) = w^T q + b = \sum_{i \in SV} \alpha_i y_i K(x_i, q) + b$$

$$-\epsilon \leq f(x) - y \leq \epsilon$$





Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). Support vector regression machines. In Advances in neural information processing systems (pp. 155-161).

Support Vector Regression

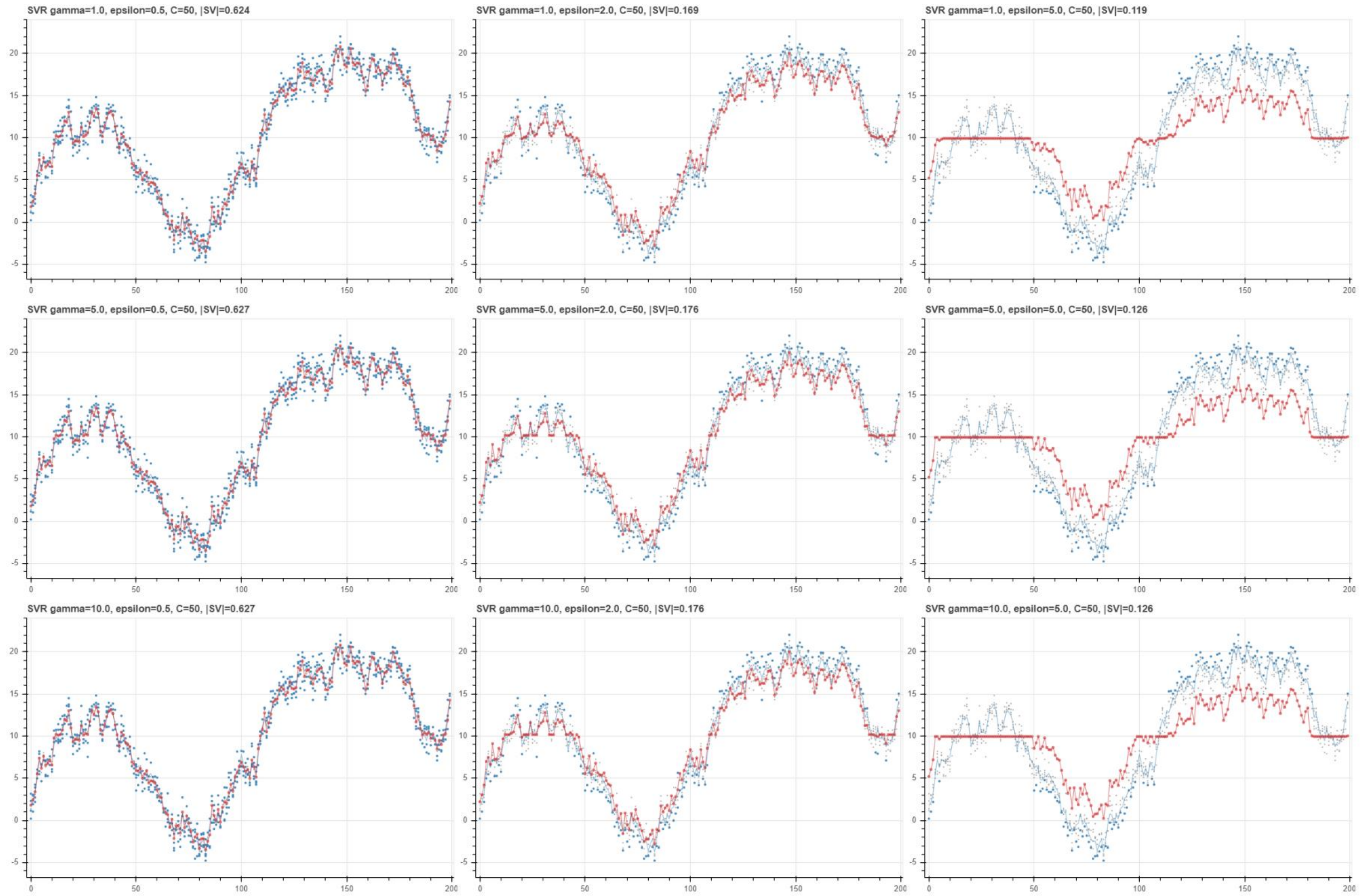
- ϵ-tube 내에 포함된 점은 loss 가 없습니다. 최대한 많은 점을 포함할 수 있는 feature space 를 만듭니다. 이 역시 support vectors 로 이뤄진 공간입니다.

$$\text{minimize } L(w, b, \alpha) = \frac{1}{2}|w|_2^2 + C\sum_i(\zeta_i^+ + \zeta_i^-)$$

$$\text{s.t.}\begin{cases} y_i - w^T x_i - b \le \epsilon + \zeta_i^+ \\ \epsilon + \zeta_i^- \le w^T x + b - y \\ \zeta_i^+, \zeta_i^- \ge 0 \end{cases}$$

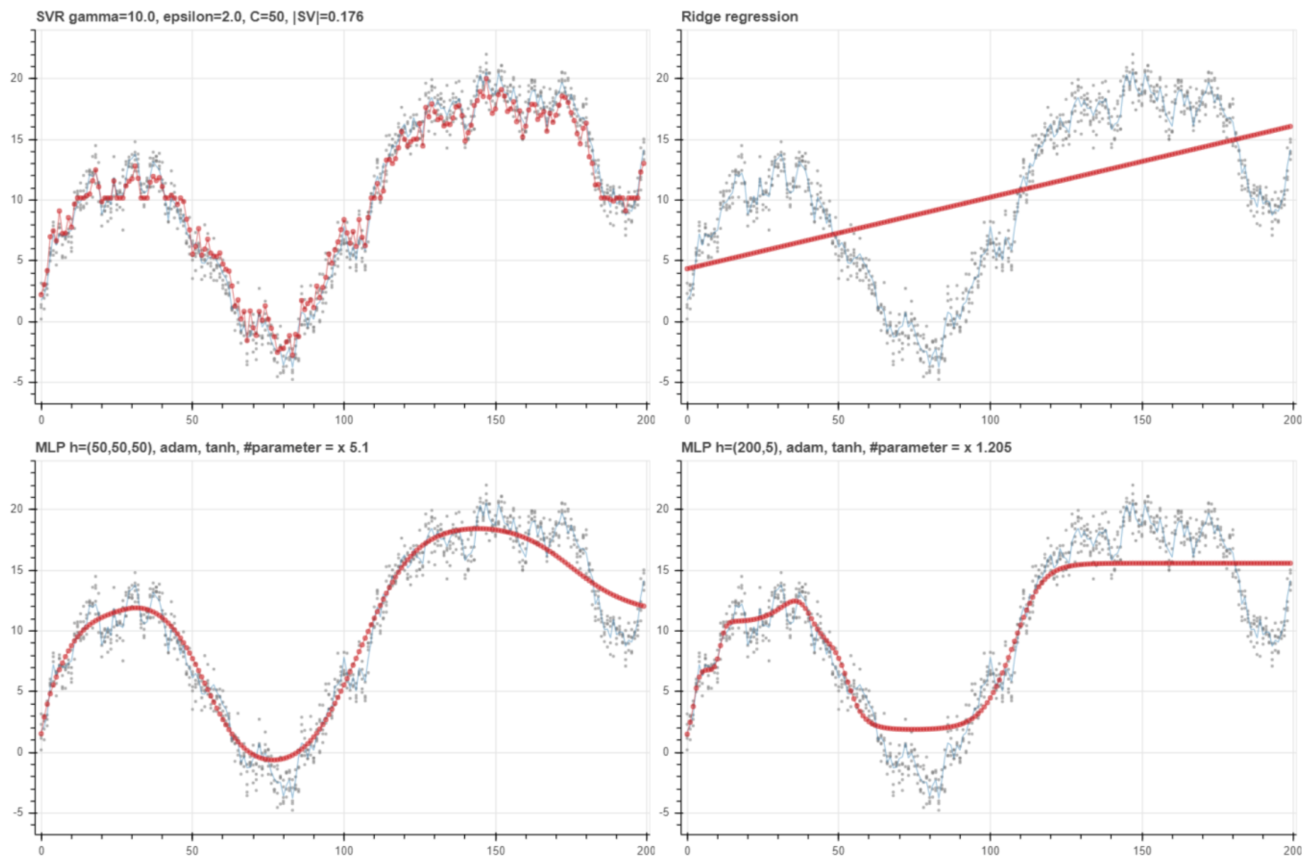
Support Vector Regression

- 커널함수를 정의하는 패러매터와 C 에 의하여 성능이 달라집니다.



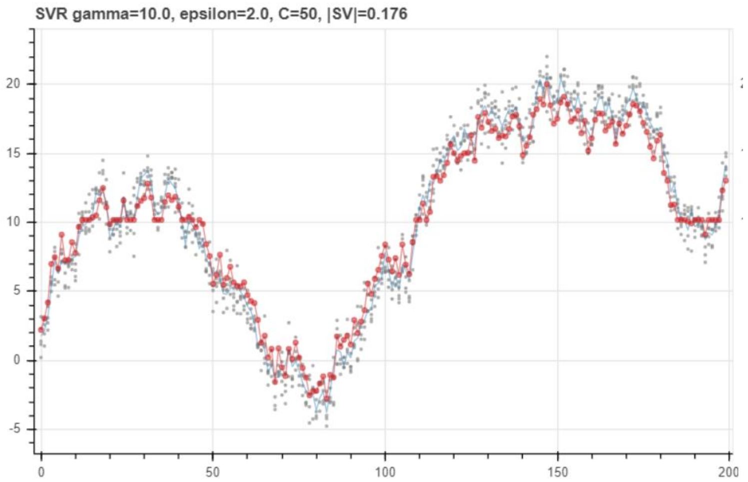
Support Vector Regression

- SVR 은 매끄러운 회귀선보다는 데이터 분포를 모사하는 선을 학습합니다.

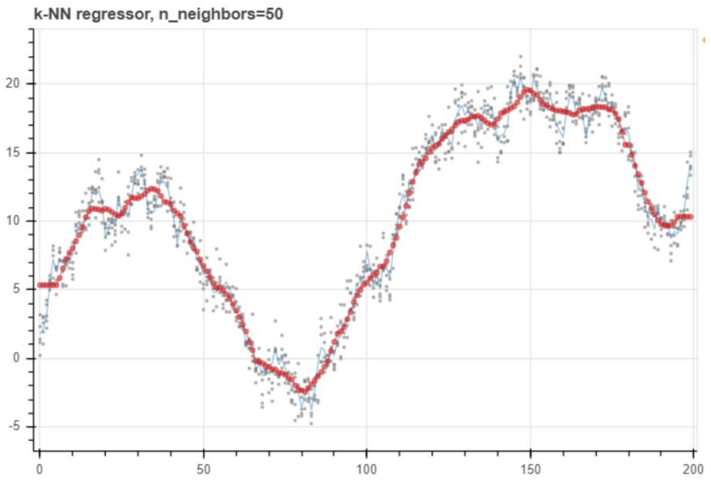


Support Vector Regression

- SVR 의 경향은 k-NN regression 의 경향과 비슷합니다.
  - $f(q) = \sum_{i \in SV} \alpha_i y_i K(x_i, q) + b$
  - SVM 은 두 종류 α_i 와 $K(x_i, q)$ 의 weights 를 지니는 instance model 입니다.



SVR example



k-NN example

Appendix
Function approximation using Tayler expansion

수식의 이해가 필수는 아닙니다.

개념이 이해 되셨다면 유도과정은 넘어가도 괜찮습니만?

XGBoost 를 공부할 때 또 등장하는 개념이라 알고 가시면 큰 도움이 됩니다.

Tayler's theorem

- 미분 가능한 함수 $f(x)$ 는 한 점 a 에서의 d 계도 미분값과 d 차 다항함수의 합으로 근사할 수 있습니다.
- 고차항의 다항함수의 합으로 표현할수록 근사함수의 오류는 줄어들며,
- 한 점 a 에서 멀어질수록 근사함수의 오류는 증가합니다.
- d 차 다항함수를 d 계도 다항함수의 합으로 표현하면 원 함수가 복원되며, 무한번 미분 가능한 함수는 무한 차원의 다항함수의 합으로 근사할 수 있습니다.

Tayler's theorem

- $g(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \frac{(x-a)^3}{3!}f'''(a) + \ \ldots$

  $= f(a) + \sum_{i=1}^{n} \frac{(x-a)^i}{i!} f^{(i)}(a)$

- $g(x+a) = f(a) + x\,f'(a) + \frac{x^2}{2!}f''(a) + \frac{x^3}{3!}f'''(a) + \ \ldots$

  $= \ f(a) + \sum_{i}^{n} \frac{x^i}{i!} f^{(i)}(a)$

다항함수의 근사

- $f(x) = x^3 + 2x^2 + 3x^1 + 4$
  - $f^{(1)} = 3x^2 + 4x + 3$
  - $f^{(2)} = 6x + 4$
  - $f^{(3)} = 6$

- $g(x)|_{a=1} = f(1) + (x-1)f^{(1)}(1) + \frac{(x-1)^2}{2}f^{(2)}(1) + \frac{(x-1)^3}{3\times 2}f^{(3)}(1)$

$$= 10 + 10(x-1) + 5(x-1)^2 + (x-1)^3$$

$$= x^3 + 2x^2 + 3x^1 + 4$$

다항함수의 근사

- $f(x) = x^3 + 2x^2 + 3x^1 + 4$
  - $f^{(1)} = 3x^2 + 4x + 3$
  - $f^{(2)} = 6x + 4$
  - $f^{(3)} = 6$

- $g(x)|_{a=1} \cong\ f(1) + (x-1)f^{(1)}(1) + \frac{(x-1)^2}{2}f^{(2)}(1)$

  $= 10 + 10(x-1) + 5(x-1)^2$

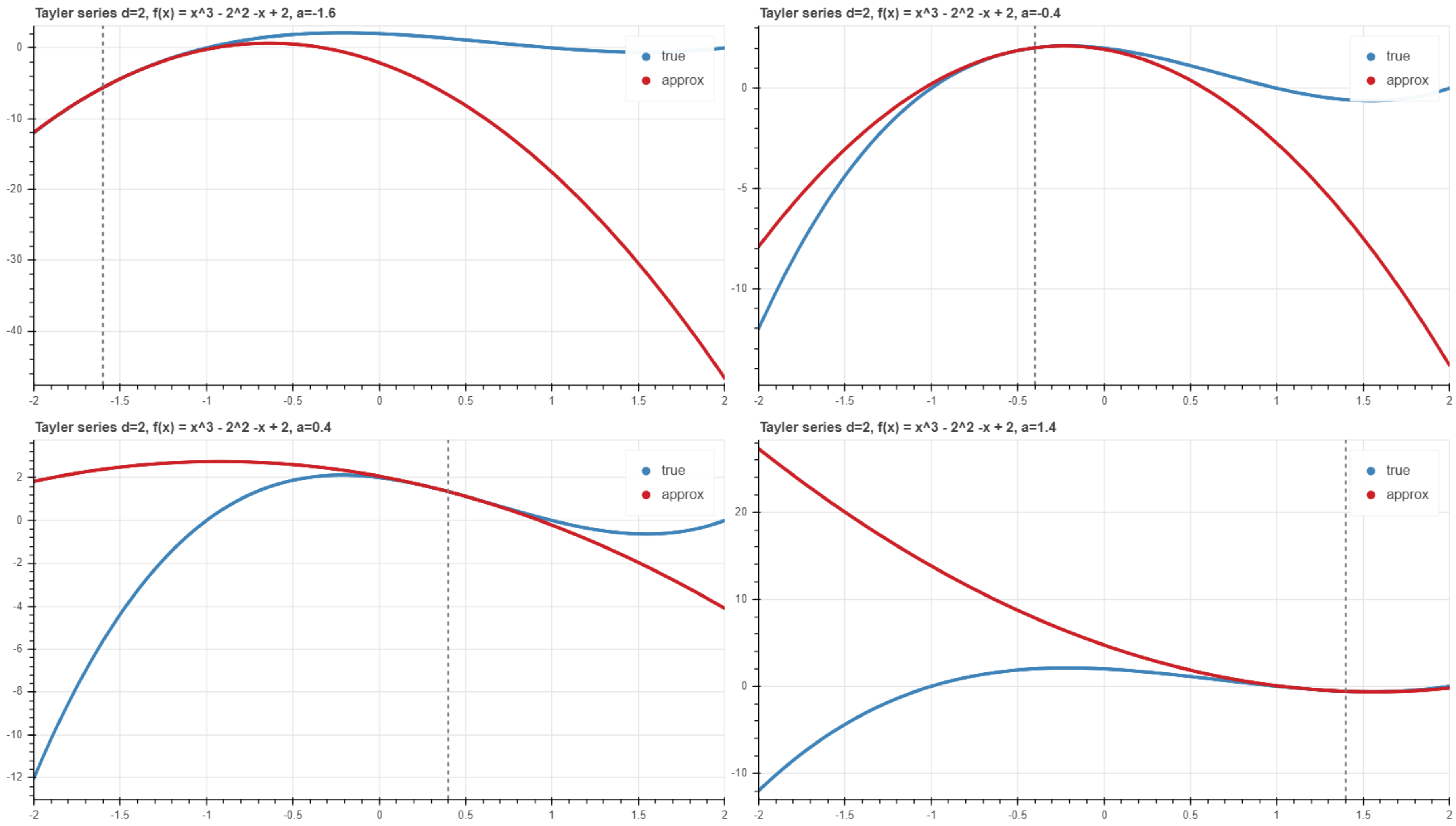
  $= 5x^2 + 5$

다항함수의 근사

- 3차 다항함수에 3차 Tayler expansion 을 적용하면 원 함수가 복원됩니다.
- 하지만 3차 다항함수에 2차 Tayler expansion 을 적용하면 a 에서 멀어질수록 오차가 커지는 2차 근사다항함수가 얻어집니다.

다항함수의 근사

- $f(x) = x^3 - 2x^2 - x^1 + 2$ 의 근사식



지수함수의 근사

- $f(x) = e^x$
  - $f^{(i)} = e^x$

- $f(x) = f(0) + xf^{(1)}(0) + \frac{x^2}{2!}f^{(2)}(0) + \ \ldots$

$$= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ \ldots$$

Tayler‘s expansion 의 쓰임

- 함수 $f(x)$ 가 주어진 구간 (domain) 에서 최소값을 가짐을 알지만, 함수가 복잡하여 도함수 $f'(x) = 0$ 를 만족하는 x^* 를 찾기 어려울 때,

  $f(x) \cong f(a) + (x-a)f^{(1)}(a) + \frac{(x-a)^2}{2} f^{(2)}(a)$ 의 최소값을 지니는 지점으로 이를 근사

- $f'(x) \cong f^{(1)}(a) + (x-a)f^{(2)}(a) = 0$

  $\rightarrow\ x^* \cong \frac{f^{(1)}(a) - af^{(2)}(a)}{f^{(2)}(a)}$

RBF kernel 의 근사

$$K(x_i, x_j) = \exp\left(-\gamma\left|x_i - x_j\right|^2\right), \text{ let } \gamma = \frac{1}{2}$$

$$= \exp\left(-\frac{1}{2}\left(x_i^2 + x_j^2\right) + x_i x_j\right)$$

$$= \exp\left(-\frac{x_i^2 + x_j^2}{2}\right)\exp(x_i x_j)$$

$$= \exp\left(-\frac{x_i^2 + x_j^2}{2}\right)\left(1 + x_i x_j + \frac{(x_i x_j)^2}{2!} + \frac{(x_i x_j)^3}{3!} + \ \dots\right)$$

$$f(x) = e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ \dots$$

$$= \exp\left(-\frac{x_i^2 + x_j^2}{2}\right)\left(1, x_i, \frac{{x_i}^2}{\sqrt{2!}}, \frac{{x_i}^3}{\sqrt{3!}}, \dots\right)^T\left(1, x_j, \frac{{x_j}^2}{\sqrt{2!}}, \frac{{x_j}^3}{\sqrt{3!}}, \dots\right)$$