INT6151 Machine Learning
Lecture 4 - SVM

Ta Viet Cuong

VNU-UET

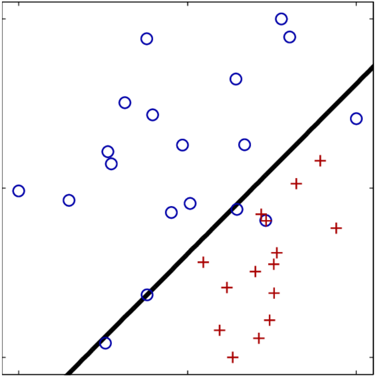
2024

Table of content

Recap: Logistic Regression

$$f(x) = w^T x + w_0$$

$$h(x) = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & \text{if } f(x) < 0 \end{cases}$$



Perceptron

The Perceptron[1] model infers the weight vector $\mathbf{w} \in \mathbb{R}^d$ and the value b representing the hyperplane

$$(H) : \{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$$

that separates the space $\mathbb{R}^d$ into two parts:
the positive class (positive, $y = +1$)

$$(+) : \{\mathbf{w}^T\mathbf{x} + b \geq 0\}$$

and the negative class (negative, $y = -1$)

$$(-) : \{\mathbf{w}^T\mathbf{x} + b < 0\}$$

[1] `https://en.wikipedia.org/wiki/Perceptron`

Linear separable

A dataset $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ in $\mathbb{R}^d \times \{-1, +1\}$ is **linearly separable** if $\exists \mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

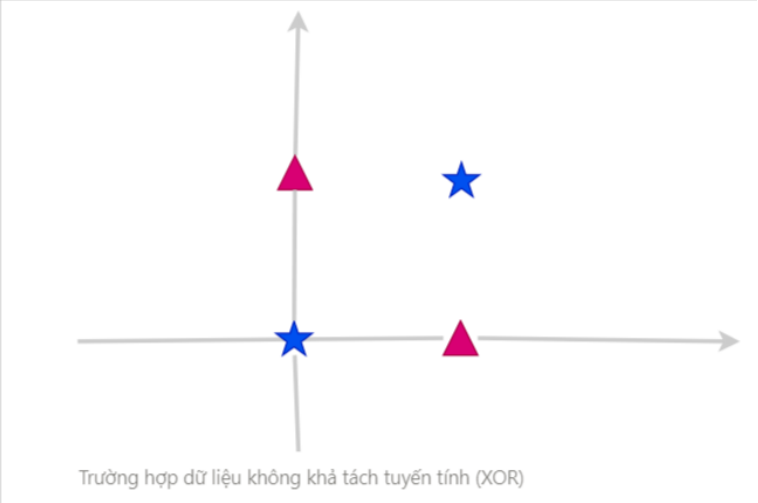
$$s_i = y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \forall i = 1, 2, \ldots, n$$

in other words, the hyperplane H completely separate the dataset D into negative and positive classes according to the labels y_i.

Mini exercises:

- Use a linear function to represent the AND function.
- Use a linear function to represent the OR function.
- Use linear functions to represent the XOR function.

Non-linear separable



Margin

Assume that D is linearly separable by a hyperplane H, The minimum distance from a data point to H is called as the margin of that hyperplane:

$$\delta = \min_{i=1}^{n} \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

Clearly, we have:

- If $\|\mathbf{w}\| = 1$ then $\delta = \min_{i=1}^{n} |\mathbf{w}^T \mathbf{x}_i + b| = \min_{i=1}^{n} y_i(\mathbf{w}^T x_i + b)$.
- If $\min_i |\mathbf{w}^T \mathbf{x}_i + b| = 1$ then $\delta = 1/\|\mathbf{w}\|$.

Perceptron: Training

Find **w** and b subject to: $s_i = y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \forall i = 1, \ldots, n$

$\text{Perceptron}(D)$

- Initialize $\mathbf{w}_{(0)} = 0, b_{(0)} = 0, t = 0$
- Iterate through the dataset multiple times, for each data sample $\mathbf{x}_i, y_i$
  - Calculate the score $s_i = y_i(\mathbf{w}_{(t)}^T \mathbf{x}_i + b_{(t)})$
  - If $s_i \geq 0$, skip (correctly classified)
  - If $s_i < 0$ (falsely classified), update $\mathbf{w}_{(t)}$ in the direction of the derivative $\frac{\partial s_i}{\partial \mathbf{w}}$ to increase s_i.

$$
\begin{aligned}
\mathbf{w}_{(t+1)} &\leftarrow \mathbf{w}_{(t)} + y_i \mathbf{x}_i \\
b_{(t+1)} &\leftarrow b_{(t)} + y_i \\
t &\leftarrow t + 1
\end{aligned}
$$

- Stop when correctly classifying all data: $s_i \geq 0, \forall i$.

Perceptron: Convergence

Theorem: If there exists $\mathbf{w}_\star, b_\star$ such that $y_i(\mathbf{w}_\star^T \mathbf{x}_i + b) \geq \delta > 0$, then the maximum number of updates of Perceptron algorithm[2] is

$$t \leq \frac{R^2(\|\mathbf{w}^\star\|^2 + b^2)}{\delta^2}$$

where R is the radius of the dataset, $R^2 = \max_i \|\mathbf{x}_i\|^2 + 1$.

[2] minimum margin equals $\delta / \|\mathbf{w}_\star\| = \delta$ if $\|\mathbf{w}_\star\| = 1$

Perceptron: Convergence

Proof: For convenience, let $\mathbf{v} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}, \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$ then $\mathbf{v}^T\mathbf{z} = \mathbf{w}^T\mathbf{x} + b$. After each update we have

$$\begin{aligned}
\|\mathbf{v}_{(t)}\|^2 &= \|\mathbf{v}_{(t-1)} + y_i\mathbf{z}_i\|^2 && (1)\\
&= \|\mathbf{v}_{(t-1)}\|^2 + \underbrace{y_i^2}_{1}\|\mathbf{z}_i\|^2 + 2\underbrace{y_i\mathbf{v}_{(t-1)}^T\mathbf{z}_i}_{s_i<0} && (2)\\
&\leq \|\mathbf{v}_{(t-1)}\|^2 + R^2 \leq tR^2 && (3)
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\|\mathbf{v}_\star\| \cdot \|\mathbf{v}_{(t)}\| &\geq \mathbf{v}_\star^T\mathbf{v}_{(t)} = \mathbf{v}_\star^T(\mathbf{v}_{(t-1)} + y_i\mathbf{z}_i) && (4)\\
&\geq \mathbf{v}_\star^T\mathbf{v}_{(t-1)} + \delta \geq t\delta && (5)
\end{aligned}$$

$$\Rightarrow \|\mathbf{v}_\star\| \geq \frac{t\delta}{R\sqrt{t}} = \frac{\sqrt{t}\delta}{R} \qquad \text{Divide by sqrt of (3)} \qquad (6)$$

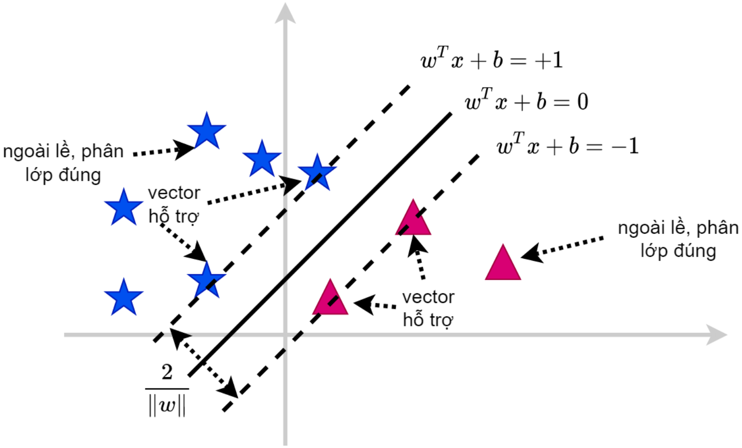
Rearrange the inequality (6), we conclude the proof.

Hard margin SVM

The SVM algorithm is inspired by the Perceptron algorithm with the idea of finding the separating hyperplane having the largest margin as follows:

$$\begin{aligned} \max_{\mathbf{w},b} \quad & \delta \\ \text{Subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \delta\|\mathbf{w}\| > 0, \forall i \end{aligned}$$

Hard margin SVM



Hard margin SVM

Without loss of generality, we can always choose $\mathbf{w}, b$ such that at the data points lying on the margin we have (see figure)

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$$

Then $\delta = 1/\|\mathbf{w}\|$ so the optimization problem becomes the following equivalent problem

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{Subject to} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i \end{aligned}$$

Soft margin SVM

The hard margin SVM requires the data set D to be linearly separable. The soft margin SVM is defined with the variable ξ as:

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{Subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \forall i \\
& \xi_i \geq 0, \forall i
\end{aligned}
$$

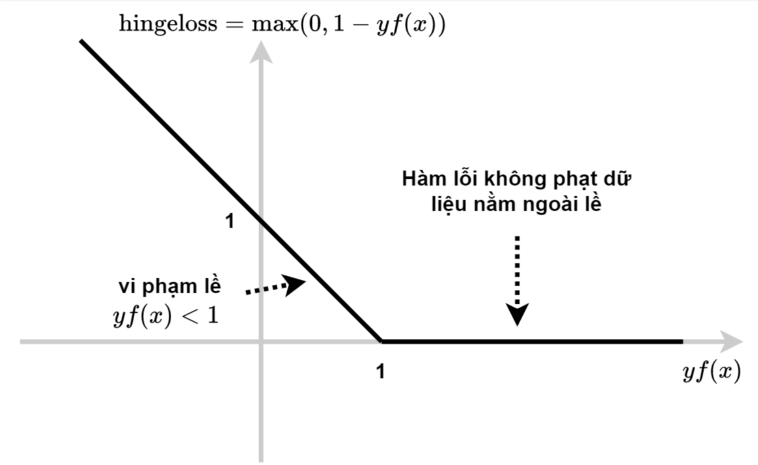
where C is a hyper parameter. And, the optimal solution must satisfy $\xi_i \geq 0$ and $\xi_i \geq 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$, we always have:

$$\xi_i = \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) = \mathrm{HingeLoss}(\mathbf{w}^T\mathbf{x}_i + b, y_i)$$

with Hinge loss function

$$\mathrm{HingeLoss}(s, y) = \max(0, 1 - y \cdot s)$$

Hinge loss



$\text{hingeloss} = \max(0, 1 - yf(x))$
1
Hàm lỗi không phạt dữ
liệu nằm ngoài lề
vi phạm lề
$yf(x) < 1$
1
$yf(x)$

Soft margin SVM

The optimization problem becomes

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

where

- The quantity $\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$ is the total error (margin) of the solution
- The quantity $\frac{1}{2}\|\mathbf{w}\|^2$ also known as the regularization term has the effect of limiting the "freedom" of the model, preventing the model from being too strong and overfit. For SVM, the non-overfitting model must have large margins.

Dual formulation

Using non-negative Lagrangian multipliers[3] ($\alpha_i \geq 0, \beta_i \geq 0$), the Lagrangian function of the soft margin optimization problem is

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + b)) + \sum_{i=1}^{n}\beta_i(-\xi_i)$$

At the optimal solution we have the derivative

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} y_i\alpha_i = 0 \Rightarrow \sum_{i=1}^{n} y_i\alpha_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \Rightarrow \alpha_i + \beta_i = C$$

[3] https://en.wikipedia.org/wiki/Lagrange_multiplier

Dual formulation

So, we have

$$\mathcal{L} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left\| \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \right\|^2 = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}_j) y_j \alpha_j$$

The dual optimization problem becomes [4]

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}_j) y_j \alpha_j$$

$$\text{Subject to} \quad 0 \leq \alpha_i \leq C, \forall i$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

[4] still having a 2nd order objective function

Karush-Kuhn-Tucker conditions

The relationship between the optimal solution of the two original problems and the dual problem is expressed by the following KKT conditions [5]:

$$\begin{aligned}
\mathbf{w} &= \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\
0 &= \alpha_i (1 - \xi_i - y_i(\mathbf{w}^T \mathbf{x}_i + b)) \\
0 &= (C - \alpha_i)\xi_i
\end{aligned}$$

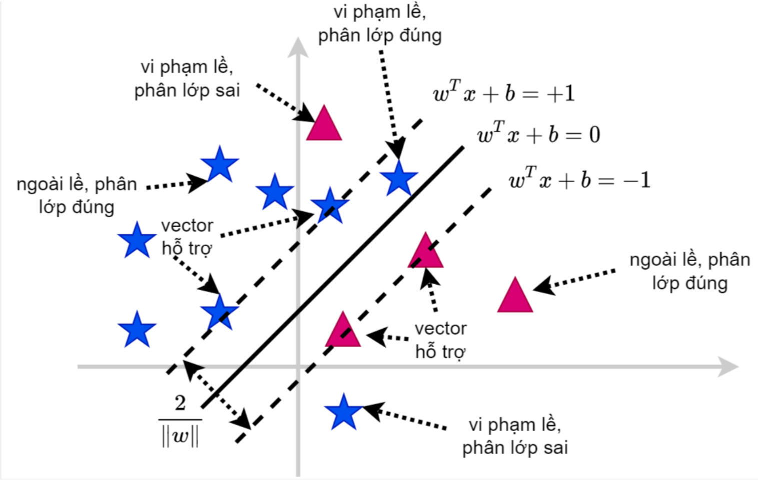
[5] in addition to the constraints of the original problem and the dual problem

- If $\alpha_i = 0$ then $C\xi_i = 0 \Rightarrow \xi_i = 0$ and $s_i = y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$, so $\mathbf{x}_i, y_i$ is outside the margin of the hyperplane. Furthermore $\alpha_i = 0$ means that this data sample does not contribute to the calculation of $\mathbf{w}$.
- If $0 < \alpha_i < C$ then $\xi_i = 0$ and $s_i = y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$, so $\mathbf{x}_i, y_i$ lies on the margin of separating hyperplane, we call these support vectors (support the margin of the hyperplane). Using support vectors, we can calculate

$$b = y_i - \mathbf{w}^T\mathbf{x}_i = y_i - \sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j^T\mathbf{x}_i)$$

- If $\alpha_i = C$ then $\xi_i \geq 0$ and $s_i = y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 1$, so $\mathbf{x}_i, y_i$ violates the margin (lying on the wrong side with respect to the margin) of the hyperplane, but it is still possible to classify true or false. For these data samples, we incur a "penalty" of a quantity ξ_i (hinge error function) in the objective function.

KKT condition



KKT condition

To solve the optimization problem

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

we can use the method downhill by derivative. where the derivative of the Hinge error function is

$$\nabla_{\mathbf{w}} \max(0, 1 - y \cdot s) = \begin{cases} -y\nabla_{\mathbf{w}} s, & y \cdot s < 1 \\ 0, & y \cdot s \geq 1 \end{cases}$$

KKT condition

Update $\mathbf{w}$ becomes

$$\mathbf{w} \leftarrow (1-\lambda)\mathbf{w} + \lambda \sum_{i:y_i(\mathbf{w}^T\mathbf{x}_i+b)<1} y_i\mathbf{x}_i$$
$$b \leftarrow b + \lambda \sum_{i:y_i(\mathbf{w}^T\mathbf{x}_i+b)<1} y_i$$

In case we put the learning samples into training in turn, the update is similar to the update formula of Perceptron, but the update is performed when there is a violation of the margin.

$$\mathbf{w} \leftarrow \mathbf{w} + \lambda\mathbb{I}(y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1)y_i\mathbf{x}_i$$
$$b \leftarrow b + \lambda\mathbb{I}(y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1)y_i$$

$\mathrm{SVMPrimal}(D, \lambda)$

- Initialize $\mathbf{w}_{(0)} = 0, b_{(0)} = 0, t = 0$
- Iterate through the dataset multiple times, for each data sample
  - Calculate the score $s_i = y_i(\mathbf{w}_{(t)}^T \mathbf{x}_i + b_{(t)})$
  - If $s_i \geq 1$, skip (no margin violation)
  - If $s_i < 1$ (margin violation), update $\mathbf{w}_{(t)}$ in the direction of the derivative $\frac{\partial s_i}{\partial \mathbf{w}}$ to increase s_i

$$
\begin{aligned}
\mathbf{w}_{(t+1)} &\leftarrow (1 - \lambda)\mathbf{w}_{(t)} + \lambda y_i \mathbf{x}_i \\
b_{(t+1)} &\leftarrow b_{(t)} + \lambda y_i \\
t &\leftarrow t + 1
\end{aligned}
$$

Solving the dual optimization problem

Dual optimization problem

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}_j) y_j \alpha_j$$

$$\text{Subject to} \quad 0 \le \alpha_i \le C, \forall i$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

is a convex optimization problem of order 2 with box-type constraints ($0 \le \alpha_i \le C$). Optimal software packages like CPLEX can solve the above problem with a small number of constraints (number of data samples).

When the number of data samples is large, it is solved by converting the above optimization problem into a series of easier optimization problems (Osuna, Freund, Girosi theorems), e.g., SMO (Sequential Minimal Optimization) algorithm.

SMO algorithm

The main steps of the SMO algorithm are
Initialization

$$u_i = \mathbf{w}^T \mathbf{x}_i + b = \sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j^T \mathbf{x}_i) + b$$

$$s_i = y_i u_i$$

SMO algorithm

1. Find a Lagrange multiplier α_i that violates the following EZ conditions

$$\alpha_i = 0 \Rightarrow s_i \geq 1$$
$$0 < \alpha_i < C \Rightarrow s_i = 1$$
$$\alpha_i = C \Rightarrow s_i \leq 1$$

2. Find another Lagrange factor α_j, optimize separately the pair (α_i, α_j), fix the other factors

- Update b, get a data with $0 < \alpha_i < C$

$$b = y_i - \mathbf{w}^T \mathbf{x}_i = y_i - \sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j^T \mathbf{x}_i)$$

3. Update $u_i, i = 1, 2, \ldots, n$

$$u_i = \sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j^T \mathbf{x}_i) + b$$

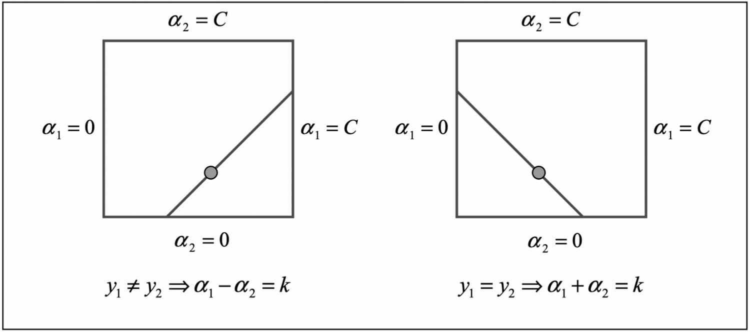
4. Repeat steps 1, 2, 3, 4 until all EZ conditions are satisfied

SMO algorithm

In steps 3, 4, only 2 values (α_i, α_j) change so it can be updated very quickly with complexity $O(n)$, especially when we have stored the dot products $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ into a matrix of size $n \times n$.

Once the pair (α_i, α_j) is selected and the other factors fixed, the problem becomes a problem of optimizing the quadratic function of a variable. Therefore, the optimal solution can be calculated using the explicit formula.

SMO algorithm



Kernel method

We see in the dual problem statement as well as the dual problem solving, all calculations are based on the scalar product $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. A groundbreaking idea for nonlinearization of linear models is to use a nonlinear map $\phi : \mathbb{R}^d \to \mathcal{X}$ where $\mathcal{X}$ is a new space (which can have infinite dimensions - the function space). Then, all solutions to the duality problem in the new space only need to calculate $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. The kernel function $\kappa(\cdot, \cdot)$ can be understood as a "window" to work on the $\mathcal{X}$ space without directly computing the ϕ (we only need the existence of ϕ without calculating ϕ).

Kernel method

The new space duality problem becomes

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}_j) y_j \alpha_j$$

$$\text{Subject to} \quad 0 \le \alpha_i \le C, \forall i$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

Similarly, the SMO algorithm also only needs to replace the dot product matrix with the matrix of the multiplication function $\kappa(\cdot,\cdot)$. Almost "for free", we have the nonlinear model through the multiplication function

$$\mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{j=1}^{n} \alpha_j y_j \kappa(\mathbf{x}_j, \mathbf{x}) + b$$

Some kernel functions

The design of the multiplication function is a familiar topic in the anthropomorphic approach. Many kinds of multiplication functions have been developed for different data types such as numeric, string, tree, graph data. Multiplication functions can also be combined to create new kernel functions.

- Linear kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$$

- polynomial kerne

$$\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- radial basis function

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

- Laplace function

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

- Tanh function

$$\kappa(\mathbf{x}, \mathbf{y}) = \tanh(\gamma \mathbf{x}^T \mathbf{y} + r)$$

Readings

- String Kernels
- Kernel Methods for Graphs