Machine learning - Support Vector Machine

Yonghoon Dong

April 14, 2024

Linear Regression

Linear models tackle the regression problem, assuming that there exists **a linear relation** between input $x \in \mathbb{R}^d$ and output $y \in \mathbb{R}$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d := \theta^T x$$

where θ_i is the parameters that parameterize the mapping from the input space $\mathcal{X}$ to output space $\mathcal{Y}$

Non-linear Regression : Basis function

Actually, there's no reason for us to insist on using only linear functions. Instead, we will use **basis functions** instead of just linear functions.

Non-linear Regression : Basis function

In mathematics, a basis function is an element of a particular **basis** for a function space. Every function in the function space can be represented as a linear combination of basis functions, just as every vector in a vector space can be represented as a linear combination of basis vectors.

Non-linear Regression : Basis function

Therefore, we can convert h_θ as follows

$$h_\theta(x) = \sum_{l=0}^{L} \theta_l \phi_l(x) := \theta^T \phi(x)$$

where $\phi_i : \mathbb{R}^d \to \mathbb{R}$ is a basis function and $\phi(x) = \Big(\phi_1(x), \phi_2(x), \ldots, \phi_L(x)\Big)^T$

Kernel Trick

If we define the loss function by least squares, the batch gradient descent update is

$$\theta \leftarrow \theta + \alpha \sum_{n=1}^{N} \left(y^{(n)} - \theta^T \phi(x^{(n)}) \right) \phi(x^{(n)})$$

Kernel Trick

Since everything is linear, we can express θ as a linear combination of basis functions. For simplicity, we assume the initial θ_0 as 0 where the numbers below theta represent the update index. Then we can express θ_n as a linear combination of basis functions

$$\theta_n = \sum_{n=1}^{N} \beta_n \phi(x^{(n)})$$

Kernel Trick

Then we can express θ_{n+1} as follows

$$\theta_{n+1} = \theta_n + \alpha \sum_{n=1}^{N} \Big(y^{(n)} - \theta^T \phi(x^{(n)}) \Big) \phi(x^{(n)}) \tag{1}$$

$$= \sum_{n=1}^{N} \beta_n \phi(x^{(n)}) + \alpha \sum_{n=1}^{N} \Big(y^{(n)} - \theta^T \phi(x^{(n)}) \Big) \phi(x^{(n)}) \tag{2}$$

$$= \sum_{n=1}^{N} \Big(\beta_n + \alpha \Big(y^{(n)} - \theta^T \phi(x^{(n)}) \Big) \Big) \phi(x^{(n)}) \tag{3}$$

Take $\beta_{n+1} = \beta_n + \alpha \Big(y^{(n)} - \theta^T \phi(x^{(n)}) \Big)$

Kernel Trick

Originally, our goal is to find θ_n. However, we already know that θ_n can be expressed as a sum of product of β_n and $\phi(x^{(n)})$. Therefore, the task of finding θ is reduced to finding β_n

$$\beta_n \leftarrow \beta_n + \alpha\Big(y^{(n)} - \theta^T \phi(x^{(n)})\Big) \tag{4}$$

$$\beta_n \leftarrow \beta_n + \alpha\Big(y^{(n)} - \sum_{m=1}^{N} \beta_m \phi(x^{(m)})^T \phi(x^{(n)})\Big) \tag{5}$$

Kernel Trick

Note that the pairwise inner products

$$\langle \phi(x^{(m)}), \phi(x^{(n)}) \rangle = \phi(x^{(m)})^T \phi(x^{(n)})$$

can be pre-calculated for all pairs of m, n

Kernel Trick : Example

Let $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, and let $\phi(x)$ be the vector that contains all the monomials of x with degree less than or equal to 3.

$$\phi(x) = (1, x_1, x_2, \ldots, x_d, x_1^2, x_1 x_2, \ldots, x_d^2, x_1^3, x_1^2 x_2 \ldots, x_d^3)^T$$

We can easily prove that the given basis functions are linearly independent.

Kernel Trick : Example

Then, we can pre-calculate $\langle \phi(x), \phi(z) \rangle$

$$\langle \phi(x), \phi(z) \rangle = 1 + \sum_{i \in \{1,\dots,d\}} x_i z_i + \sum_{i,j \in \{1,\dots,d\}} x_i x_j z_i z_j + \sum_{i,j,k \in \{1,\dots,d\}} x_i x_j x_k z_i z_j z_k \quad (6)$$

$$= 1 + (\sum_{i=1}^{d} x_i z_i) + (\sum_{i=1}^{d} x_i z_i)^2 + (\sum_{i=1}^{d} x_i z_i)^3 \quad (7)$$

$$= 1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3 \quad (8)$$

Kernel Trick

We define the **Kernel** $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that

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

where $\mathcal{X}$ is a input space. As we mentioned above, it can be pre-calculated.

Least squares with Kernel Trick

If we apply the kernel trick, the least square algorithm can be written as follows

1. Compute all values $\mathcal{K}(x^{(m)}, x^{(n)})$ for all $m, n \in \{1, \ldots, N\}$
2. Set the initial value of θ as 0
3. Iterate until it converges

$$\beta_n \leftarrow \beta_n + \alpha\left(y^{(n)} - \sum_{m=1}^{N} \beta_m K(x^{(m)}, x^{(n)})\right)$$

4. Predict of value y for a new input x

$$\theta^T \phi(x) = \sum_{n=1}^{N} \beta_n \phi(x^{(n)})^T \phi(x) = \sum_{n=1}^{N} \beta_n K(x^{(n)}, x)$$

Least squares with Kernel Trick

Note that since x is a new input, $K(x^{(n}, x)$ can not be pre-calculated.

Conditions for valid kernels

1. As we mentioned above, $K(x^m, x^{(n)})$ can be pre-calculated. So, we can define the **kernel matrix** such that

$$K_{mn} = K(x^m, x^{(n)})$$

2. If kernel is valid, the kernel matrix must be **symmetric**
3. If kernel is value, the kernel matrix must be positive semi-definite.

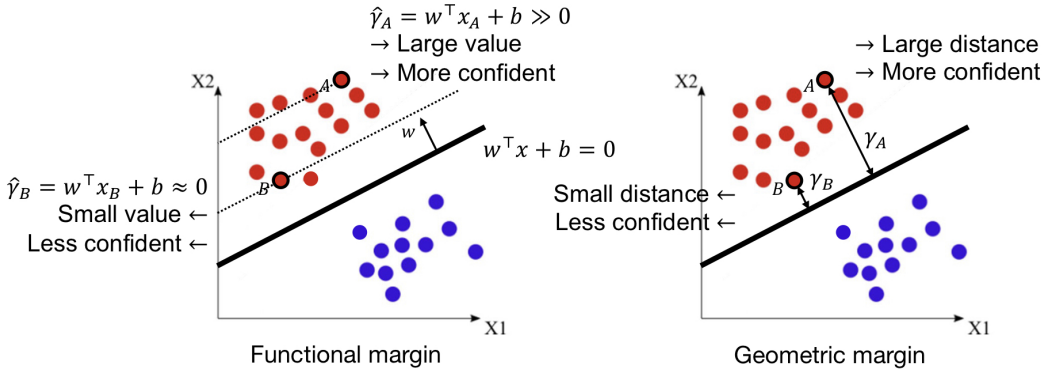
Conditions for valid kernels

Theorem (Mercer kernel)

For given $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *to be a valid kernel, it is necessary and sufficient that for any* $\{x^{(1)}, \ldots, x^{(N)}\}$ *the corresponding kernel matrix is* **symmetric and positive semi-definite**

Support Vector Machine : Purpose

Consider a case of binary classification problem. We want to find the decision boundary $W^T x + b$ i.e. We want to find the best affine function that separate the given classes.



Support Vector Machine : Two types of Margin

Actually, there are two different types of margin.

1. Functional margin : Related to the distance between the given point and the affine function, but its value can be affected by the norm of w.
2. Geometric margin : Actual distance between the given point and the affine function

Support Vector Machine : Functional Margin

Functional margin for a training example $(x^{(n)}, y^{(n)})$ is defined by

$$\hat{\gamma}^{(n)} = y^{(n)}(w^T x^{(n)} + b)$$

where $y^{(n)} \in \{-1, 1\}$ .To align with geometric meaning (distance of the given point and the line), w must point towards the side where the label is 1.

Support Vector Machine : Functional margin

Given a training set $S = \{(x^{(n)}, y^{(n)}\}$, we also define the function margin with respect to S as the smallest of the functional margins of the training examples

$$\hat{\gamma} = \min_{n=1,\dots,N} \hat{\gamma}^{(n)}$$

Caution

Note that it is different from the functional margin of the given point. It is just a new definition. Don't be confused.

Support Vector Machine : Functional Margin

One important observation is that the functional margin can be affected by the length of w. So we want to restrict the size of w to 1 so that we can interpret as a actual distance between the given point and the affine function. So, this becomes a new definition : **Geometric Margin**

Support Vector Machine : Geometric Margin

Geometric margin for a training example $(x^{(n)}, y^{(n)})$ is defined by

$$\gamma^{(n)} = y^{(n)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(n)} + \frac{b}{\|w\|} \right)$$

where $y^{(n)} \in \{-1, 1\}$. Actually, it is just a actual distance by scaling the functional margin.

Support Vector Machine : Geometric Margin

Similar to the functional margin case, we also define the geometric margin with respect to S as the smallest of the functional margins of the training examples

$$\gamma = \min_{n=1,\dots,N} \gamma^{(n)}$$

Caution

Note that it is different from the geometric margin of the given point. It is just a new definition. Don't be confused.

Support Vector Machine : Objective

We want to find the decision boundary that maximize the geometric margin, since it would reflect a very confident set of predictions on the training set.

Objective function

$$\underset{\gamma,w,b}{\text{maximize}}\ \gamma$$

subject to

$$y^{(n)}(w^T x^{(n)} + b) \geq \gamma, \quad n = 1, \ldots, N$$
$$\|w\| = 1$$

Support Vector Machine : Objective

However, $\|w\| = 1$ is non-convex i.e. Our objective function is hard to optimize compared to the convex case.

Revised objective function

$$\underset{\hat{\gamma},w,b}{\textit{maximize}} \ \frac{\hat{\gamma}}{\|w\|}$$

subject to

$$y^{(n)}(w^T x^{(n)} + b) \geq \hat{\gamma}, \quad n = 1, \ldots, N$$

Since $\gamma = \hat{\gamma}/\|w\|$, our actual objective is still to minimize the geometric margin.

Support Vector Machine : Objective

We can make $\hat{\gamma}$ to 1 by changing the norm of w. So, without loss of generality, we can regard it as 1.

Revised objective function

$$\underset{w,b}{maximize} \ \frac{1}{\|w\|}$$

subject to

$$y^{(n)}(w^T x^{(n)} + b) \geq 1, \quad n = 1, \ldots, N$$

Support Vector Machine : Objective

Maximizing $1/\|w\|$ is actually equivalent to minimizing $1/2\|w\|^2$ because they are monotone function.

Revised objective function

$$\underset{w,b}{minimize}\ \frac{1}{2}\|w\|^2$$

subject to

$$y^{(n)}(w^T x^{(n)} + b) \geq 1, \quad n = 1, \ldots, N$$

This objective function can be solved by **quadratic programming (QP)**. Therefore, we can find the optimal w and b very efficiently. Actually, there is a better approach than QP by using the concept of **duality**.

Optimization problem in standard form

$$\underset{x}{\text{minimize}}\ f_0(x)$$

subject to

$$f_i(x) \leq 0,\ i = 1, \ldots, m$$
$$h_i(x) = 0,\ i = 1, \ldots, p$$

- $x \in \mathbb{R}^d$ is the optimization variable
- $f_0 : \mathbb{R}^d \to R$ is the objective or cost function
- $f_i : \mathbb{R}^d \to R,\ i = 1, \ldots, m$ are the inequality constraint functions
- $h_i : \mathbb{R}^d \to R,\ i = 1, \ldots, p$ are the inequality constraint functions

Barrier method

Consider the inequality constrained problem

$$\underset{x}{minimize}\ f_0(x)$$

subject to

$$f_i(x) \leq 0, i = 1, \ldots, m$$

Barrier methods use a barrier term that approaches the infinite penalty function ϕ.

Barrier method

Let $\phi(x)$ be a function that is continuous on the interior of the feasible set, and that becomes unbounded as the boundary of the set is approached from its interior:

$$\phi(x) \to \infty \quad \text{as} \quad f_i(x) \to 0$$

There are two examples:

1. Logarithmic function

$$\phi(x) = \sum_{i=1}^{m} \log(f_i(x))$$

2. Inverse function

$$\phi(x) = -\sum_{i=1}^{m} \frac{1}{f_i(x)}$$

Barrier method

Now let μ be a positive scalar. Then $\mu\phi(x)$ will approach $\sigma(x)$ as μ approaches zero.
By adding a barrier term of the form $\mu\phi(x)$ to the objective, we obtain a 'barrier function'

$$\beta_\mu(x) = f_0(x) + \mu\phi(x)$$

Barrier method

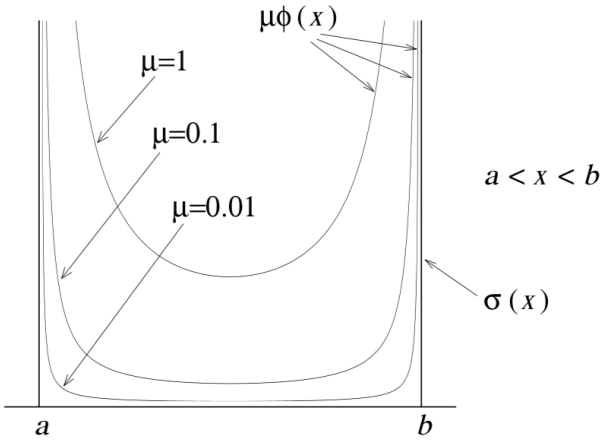


Figure: Illustration of barrier method

Lagrangian via barrier method

Let our original problem as follows

$$\underset{x}{\textit{minimize}}\ f_0(x)$$

subject to

$$f_i(x) \leq 0,\ i = 1, \ldots, m$$

Lagrangian via barrier method

One way to make this inequality constraint to equality constraint is using a penalty function.

$$\underset{x}{minimize}\left[f_0(x) + \sum_{i=1}^{m} P_i(f_i(x))\right]$$

where

$$P_i(x) = \begin{cases} \infty & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Lagrangian via barrier method

However, it has a problem because it is non-differentiable at $x = 0$. This problem can't be fixed even though we change ∞ to some large value. Therefore, we take a linear function as a penalty function.

$$P_i(x) = \lambda_i x$$

where $\lambda_i \geq 0$

Note

It is quite natural. If we choose x that doesn't satisfy the constraints, it rewards. On the other hand, if we choose x that doesn't satisfy the constraints, it penalize.

Lagrangian via barrier method

The problem is that it could change the optimal value. If we take the maximum of λ_i, we can act as a original penalty function. Therefore, the original function can be express as

$$\underset{x}{minimize}\left[f_0(x) + \underset{\lambda}{maximize}\sum_{i=1}^{m}\lambda_i f_i(x)\right] \tag{9}$$

$$= \underset{x}{minimize}\ \underset{\lambda}{maximize}\left[f_0(x) + \sum_{i=1}^{m}\lambda_i f_i(x)\right] \tag{10}$$

Caution

It is actually a min-max problem which means the order is very important.

Lagrangian via barrier method

Definition (Lagrangian)

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

where $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$

Definition (Primal objective)

$$\underset{x}{\textit{minimize}} \ \underset{\lambda,\nu}{\textit{maximize}} \ \mathcal{L}(x, \lambda, \nu)$$

Lagrange Dual function

Definition (Lagrange dual function)

$$
\begin{aligned}
g(\lambda, \nu) &= \inf_{x \in \mathcal{D}} \mathcal{L}(x, \lambda, \nu) \\
&= \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)
\end{aligned}
$$

where $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$

Lower bound property of Dual function

Theorem

If $\lambda \succeq 0$, then $g(\lambda, \nu) \leq p^*$ where p^* is a optimal value of the primal.

Proof.

If $\bar{x}$ is feasible and $\lambda \succeq 0$, then

$$f_0(\bar{x}) \geq \mathcal{L}(\bar{x}, \lambda, \nu) \geq \inf_{x \in \mathcal{D}} \mathcal{L}(x, \lambda, \nu) = g(\lambda, \nu)$$

minimizing over all feasible $\bar{x}$ gives $p^* \geq g(\lambda, \nu)$ □

Lagrange Dual problem

For each pair (λ, ν) with $\lambda \succeq 0$, the Lagrange dual function gives us a **lower bound** on the optimal value p^* of the optimization problem. Thus we have a lower bound that depends on some parameters λ, ν.

Lagrange Dual problem

The natural question is "What is the 'best' lower bound that can be obtained from the Lagrange dual function?". This leads to the optimization problem

Definition (Lagrange dual problem)

$$\text{maximize } g(\lambda, \nu)$$

subject to

$$\lambda \succeq 0$$

Weak and Strong Duality

1. Weak duality $d^* \leq p^*$ (where d^* is the optimal value of the dual problem)
   1. it always hold for any optimization problems
   2. it can be used to find non-trivial lower bounds for difficult problem
2. strong duality : $d^* = p^*$
   1. it does not hold in general
   2. (usually) holds for convex problems
   3. conditions that guarantee strong duality in convex problems are called **constraint qualifications**

Slater's condition

One simple constraint qualification is **Slater's condition**
Let our problem is as follows

$$\text{minimize } f_0(x)$$

subject to

$$f_i(x) \leq 0 \quad i = 1, \ldots, m \tag{11}$$

$$Ax = b \tag{12}$$

where $f_0, \ldots, f_m$ is a convex function.

Slater's condition

If there exists an $x \in \text{relint } \mathcal{D}$ such that

$$f_i(x) < 0 \quad i = 1, \ldots, m \tag{13}$$

$$Ax = b \tag{14}$$

Such a point is sometimes called **strictly feasible**

Caution

Slater's theorem states that strong duality holds if Slater's condition holds and the problem is convex.

KKT condition

We now assume that the functions f_i, h_i are differentiable and therefore have open domains. Let x^* and (λ^*, ν^*) be any primal and dual optimal points with zero duality gap.

KKT condition

Theorem (KKT condition)

1. *Primal constraints*

$$f_i(x^*) \leq 0 \quad i = 1, \ldots, m \quad h_i(x^*) = 0 \quad i = 1, \ldots, m \tag{15}$$

2. *Dual constraints:* $\lambda^* \succeq 0$
3. *Complementary slackness :* $\lambda_i f_i(x^*) = 0 \quad i = 1, \ldots, m$
4. *Stationary condition*

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0$$

Revisit support vector machine objective

The primal optimization problem in support vector machine problem is

Primal problem of support vector machine

$$\underset{w,b}{minimize}\ \frac{1}{2}\|w\|^2$$

subject to

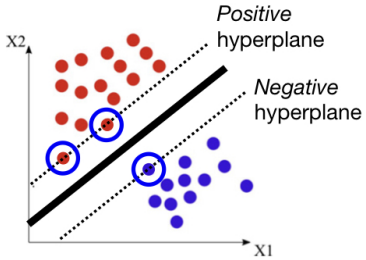
$$y^{(n)}(w^T x^{(n)} + b) \geq 1, \quad n = 1, \ldots, N$$

The constraint for each training example can be rewritten as follows

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

Revisit support vector machine objective

From the KKT complementary slackness condition, we will have $\lambda_i > 0$ only for the training examples that have functional margin exactly equal to one (i.e. $g_i(x) = 0$).



Support vectors

refer to the examples corresponding to constraints that hold with equality, $g_i(w) = 0$

Figure: Illustration of support vector

Lagrangian in support vector machine

The **Lagrangian** for our optimization problem can be constructed as follows

$$\mathcal{L}(w, b, \lambda) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} \lambda_n [y^{(n)}(w^T x^{(n)} + b) - 1]$$

KKT condition for support vector machine

By stationary condition,

$$\nabla_w \mathcal{L}(w, b, \lambda) = w - \sum_{n=1}^{N} \lambda_n y^{(n)} x^{(n)} = 0 \tag{16}$$

$$\Rightarrow w^* = \sum_{n=1}^{N} \lambda_n y^{(n)} x^{(n)} \tag{17}$$

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \lambda) = \sum_{n=1}^{N} \lambda_n y^{(n)} = 0 \tag{18}$$

KKT condition for support vector machine

By dual feasibility

$$a_n \geq 0, \quad n = 1, \ldots, N$$

By complementary slackness

$$\sum_{n=1}^{N} a_n y^{(n)} = 0$$

KKT condition for support vector machine

Instead of solving primal problem, we want to solve dual problem instead.

$$\underset{\lambda}{\text{maximize}} \; \underset{w,b}{\text{minimize}} \; \mathcal{L}(w, b, \lambda) \tag{19}$$

$$= \underset{\lambda}{\text{maximize}} \left[\sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{m,n=1}^{N} y^{(m)} y^{(n)} \lambda_m \lambda_n (x^{(m)})^T x^{(n)} - b \sum_{n=1}^{N} \lambda_n y^{(n)} \right] \tag{20}$$

$$= \underset{\lambda}{\text{maximize}} \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{m,n=1}^{N} y^{(m)} y^{(n)} \lambda_m \lambda_n (x^{(m)})^T x^{(n)} \tag{21}$$

subject to

$$\lambda_n \geq 0, \quad n = 1, \ldots, N \tag{22}$$

$$\sum_{n=1}^{N} \lambda_n y^{(n)} = 0 \tag{23}$$

Prediction of new input

Suppose we have fit our model's parameters to a training set, and then we can make a prediction at a new input x

$$(w^*)^T x + b = \left(\sum_{n=1}^{N} \lambda_n y^{(n)} x^{(n)} \right)^T x + b = \sum_{n=1}^{N} \lambda_n y^{(n)} \langle x^{(n)}, x \rangle + b$$

Caution

Note that λ_i will be zero except for the support vectors