Machine Learning Explained

Lecture Notes of CS385, SJTU, taught by Prof. Quanshi Zhang

Summarized and Written by Tony Fang (galaxies@sjtu.edu.cn, tony.fang.galaxies@gmail.com)

Chapter 2. Support Vector Machine and Kernel Methods

2.1 Functional Margin

Logistic Regression with 0/1 Responses

Let $\{(X_i, y_i)\}_{i=1,2,\cdots,n}$ denote the set of training samples, $y_i \in \{0,1\}$. The basic idea of linear classification is to estimate a hyperplane that can separate two classes of samples. According to the idea of logistic regressions,

$$p(y_i = 1 \mid X_i) = p_i = \frac{1}{1 + e^{-X_i^T \beta}}$$

Hence, $p_i \geq 0.5$ if and only if $X_i^T \beta > 0$. And we can make the following clarifications.

(a) If $X_i^T \beta \gg 0$, then $p_i \approx 1$, the model predicts $y_i = 1$ with a high confidence;

(b) If $X_i^T \beta \ll 0$, then $p_i \approx 0$, the model predicts $y_i = 0$ with a high confidence;

(c) If $-\tau \leq X_i^T \beta \leq \tau$, then the prediction is sensitive to the noise since the confidence of the prediction is not-so-high.

Perceptron for Classification with +/- Responses.

Let $\{(X_i, y_i)\}_{i=1,2,\cdots,n}$ denote the set of training samples, $y_i \in \{\pm 1\}$. We use the weights vector w and the bias vector b to represent the perceptron instead of β, as introduced before. Hence,

$$\hat{y} = \text{sign}(w^T X_i + b)$$

The **functional margin** of (w, b) with respect to (X_i, y_i) is defined as

$$\gamma_i = y_i (w^T X_i + b)$$

We expect the functional margin to be large for confident classification. If $y_i = +1$, a large functional margin indicates $w^T x + b \gg 0$, *i.e.*, confidently assigning a positive label; if $y_i = -1$, a large functional margin indicates $w^T x + b \ll 0$, *i.e.*, confidently assigning a negative label.

As introduced before, perceptron is a deterministic version of logistic regression. Therefore, the gradient descent algorithm derived in section 1.12 can be modified into the following form:

$$w_{t+1} = w_t + \eta_t \sum_{i=1}^{n} \delta_i X_i y_i \,, \qquad b_{t+1} = b_t + \eta_t \sum_{i=1}^{n} \delta_i y_i$$

where $\delta_i = [y_i \neq \text{sign}(w_t^T x_i + b_t)]$ and $[\cdot]$ is the predicate function.

The gradient descent algorithm can be interpreted as the algorithm for the loss function

$$loss(w, b) = \sum_{i=1}^{n} \max\big(0, -y_i (w^T X_i + b)\big) = \sum_{i=1}^{n} \max(0, -\gamma_i)$$

When $-\gamma_i \leq 0$, that is, $\gamma_i \geq 0$, then no mistake is made, and we do not need to conduct punishment. When $-\gamma_i > 0$, then we need to force the algorithm "***learn from errors***".

Scaling.

Again, let us eliminate the influence of the scale. The model can have multiple solutions if we change the scale of w and b simultaneously by multiplying a fixed factor. This does not change the classification results, but may have huge influence on the margin. Therefore, we normalize the weight by dividing the L-2 norm of w.

$$w \leftarrow \frac{w}{\|w\|_2}, \qquad b \leftarrow \frac{b}{\|w\|_2}$$

where $\|w\|_2 = \sqrt{\sum_j w_j^2}$, and can be simplified to $\|w\|$. Hence, the modified functional margin is given as

$$\gamma_i = \frac{y_i (w^T X_i + b)}{\|w\|} = y_i \left(\left(\frac{w}{\|w\|} \right)^T X_i + \frac{b}{\|w\|} \right)$$

An interesting fact is $B_i = X_i - \gamma_i \cdot \left(\frac{w}{\|w\|}\right) y_i$ must localize on the decision boundary, that is, $w^T B_i + b = 0$. Just plug in the condition equation into the final formula is able to prove the fact.

2.2 Lagrange Multipliers and Karush-Kuhn-Tucker Conditions

For a simple optimization problem

$$\min_w f(w) \quad \text{s.t. } g(w) = 0$$

we define the Lagrangian as $L(w, \lambda) = f(w) + \lambda g(w)$. Lagrange multipliers tells us that now we only need to optimize the Lagrangian. Then, we just need to solve the following equations to obtain the optimal solution.

$$\frac{\partial L}{\partial w} = 0, \qquad \frac{\partial L}{\partial \lambda} = 0$$

Notice that the first equations ensures that we reach the minima of w (at least local minima), and the second equations ensures that we follow the limitation of $g(w) = 0$.

For a more general case of optimization problem

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_k(w) \le 0 \ (k = 1, 2, \cdots, K) \\ & h_l(w) = 0 \ (l = 1, 2, \cdots, L) \end{aligned}$$

we define the generalized Lagrangian as

$$L(w, \alpha, \beta) = f(w) + \sum_{k=1}^{K} \alpha_k g_k(w) + \sum_{l=1}^{L} \beta_l h_l(w)$$

Then, consider the quantity

$$V(w) = \max_{\alpha, \beta : \alpha_k \ge 0} L(w, \alpha, \beta)$$

If $g_k(w) > 0$ or $h_l(w) \ne 0$, then $V(w)$ must be infinity. Thus,

$$\min_w V(w) = \min_w \max_{\alpha, \beta : \alpha_k \ge 0} L(w, \alpha, \beta)$$

Let us consider the problem from another perspective.

$$V(\alpha, \beta) = \min_w L(w, \alpha, \beta)$$

Then,

$$\max_{\alpha, \beta : \alpha_k \ge 0} V(\alpha, \beta) = \max_{\alpha, \beta : \alpha_k \ge 0} \min_w L(w, \alpha, \beta)$$

which is the dual form of the former optimization problem. The duality theorem states that

$$\min_w V(w) \ge \max_{\alpha, \beta : \alpha_k \ge 0} V(\alpha, \beta)$$

which is proved by simply performing derivations from inner-min/max to outer-min/max as follows.

$$\min_w L(w, \alpha, \beta) \le L(w, \alpha, \beta) \implies \max_{\alpha, \beta : \alpha_k \ge 0} V(\alpha, \beta) = \max_{\alpha, \beta : \alpha_k \ge 0} \min_w L(w, \alpha, \beta) \le \max_{\alpha, \beta : \alpha_k \ge 0} L(w, \alpha, \beta) = V(w)$$

$$\implies \max_{\alpha, \beta : \alpha_k \ge 0} V(\alpha, \beta) \le \min_w V(w)$$

Now let us focus on the condition that ensures

$$\min_w V(w) = \max_{\alpha, \beta : \alpha_k \ge 0} V(\alpha, \beta)$$

which is termed as Karush-Kuhn-Tucker (KKT) condition, stating that the necessary condition of the optimal solution are as follows.

$$\frac{\partial}{\partial w} L(w, \alpha, \beta) = 0, \qquad \frac{\partial}{\partial \beta} L(w, \alpha, \beta) = 0$$

$$\forall k, \qquad \alpha_k g_k(w) = 0, g_k(w) \le 0, \alpha_k \ge 0$$

2.3 Margin Classifier Selection: Modeling and Transformation

Let us assume that all training samples can be correctly classified. Then, how to find the best classifiers among all the classifiers that can achieve 100% accuracy?

For all training samples, we usually focus on the most challenging one, *i.e.*, those with the minimum functional margin

$$\hat{\gamma} = \min_{i=1,2,\cdots,n} \gamma_i$$

Therefore, what we want to do is to maximize the minimum functional margin. The optimization problem can be written as follows. Notice that here we directly restrict the scale of w by setting limitation $\|w\| = 1$.

$$\begin{aligned} \max_{\tau,w,b} \quad & \tau \\ \text{s.t.} \quad & \forall i, \gamma_i = y_i(w^T X_i + b) \geq \tau \\ & \|w\| = 1 \end{aligned}$$

which can be solved using Lagrange multipliers with the loss function as

$$L(\tau, w, b, \alpha, \beta) = -\tau - \sum_{i=1}^{n} \alpha_i [y_i(w^T X_i + b) - \tau] + \beta(\|w\| - 1)$$

We re-write the optimization to absorb the condition $\|w\| = 1$ as follows.

$$\begin{aligned} \max_{\tau,w,b} \quad & \frac{\tau}{\|w\|} \\ \text{s.t.} \quad & \forall i, \gamma_i = y_i(w^T X_i + b) \geq \tau \end{aligned}$$

which can be solved similarly with the loss function as

$$L(\tau, w, b, \alpha) = -\frac{\tau}{\|w\|} - \sum_{i=1}^{n} \alpha_i [y_i(w^T X_i + b) - \tau]$$

Setting $\|w\| = 1$ is symmetric with setting $\tau = 1$, thus we get the following optimization.

$$\begin{aligned} \min_{w,b} \quad & \|w\| \\ \text{s.t.} \quad & \forall i, \gamma_i = y_i(w^T X_i + b) \geq 1 \end{aligned}$$

with the loss function as

$$L(w, b, \alpha) = \|w\| - \sum_{i=1}^{n} \alpha_i [y_i(w^T X_i + b) - 1]$$

Finally, the optimization can be transformed into an equivalent description by modifying its objective.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & \forall i, \gamma_i = y_i(w^T X_i + b) \geq 1 \end{aligned}$$

whose loss function is

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T X_i + b) - 1]$$

We transform the optimization into this version in order to make the gradient of the loss function more convenient and tractable in the following calculations.

2.4 Support Vectors

Apply Lagrange multipliers to the final optimization problem, and the Karush-Kuhn-Tucker condition tells us that

$$\alpha_i [y_i(w^T X_i + b) - 1] \geq 0$$

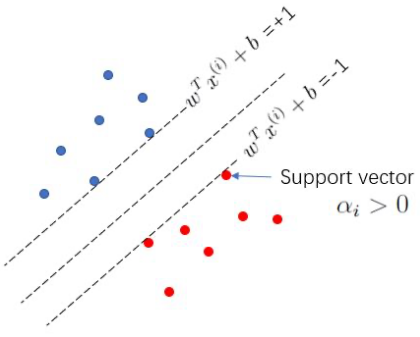
and

$$\alpha_i \geq 0, \qquad y_i(w^T X_i + b) - 1 \geq 0$$

Therefore, we may find some samples with $\alpha_i > 0$, which implies

$$y_i(w^T X_i + b) - 1 = 0$$

We call these samples **support vectors**. We can conclude the following properties of the support vectors.



1. Support vectors are usually much less than the training samples;
2. Support vectors usually correspond to samples that are difficult to classify, as the figure shown.

2.5 Margin Classifier Selection: Optimization

We construct the generalized Lagrangian as follows to learn the model.

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i[y_i(w^TX_i+b)-1]$$

Step 1. First fix α and learn w and b to minimize $L(w,b,\alpha)$. Karush-Kuhn-Tucker condition tells us that

$$\frac{\partial L(w,b,\alpha)}{\partial w}=w-\sum_{i=1}^{n}\alpha_i y_i X_i=0$$

and

$$\frac{\partial L(w,b,\alpha)}{\partial b}=-\sum_{i=1}^{n}\alpha_i y_i=0$$

Hence, we can derive that

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

and

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

Plug the results into the generalized Lagrangian, and we can get

$$\begin{aligned}L(w,b,\alpha)&=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left[y_i(w^TX_i+b)-1\right]\\&=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_jX_i^TX_j-\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_jX_i^TX_j-b\sum_{i=1}^{n}\alpha_iy_i+\sum_{i=1}^{n}\alpha_i\\&=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_jX_i^TX_j\end{aligned}$$

Step 2. Thus, re-write the objective as follows to optimize α.

$$\begin{aligned}\max_{\alpha}\quad & W(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_jX_i^TX_j\\\text{s.t.}\quad & \forall i,\alpha_i\geq 0\\&\sum_{i=1}^{n}\alpha_iy_i=0\end{aligned}$$

Since this is a quadratic programming problem, we can use the conventional algorithm to solve it. The data quantity is too large for the algorithm to perform efficiently. Sequential Minimal Optimization method[1] can be used to solve the optimization problem in an efficient way, which will be introduced in the appendix of this chapter. Notice that, the optimization process should follow the Karush-Kuhn-Tucker condition, that is,

$$\alpha_i\geq 0,\qquad y_i(w^TX_i+b)-1\geq 0,\qquad \alpha_i[y_i(w^TX_i+b)-1]\geq 0$$

Theoretically, we can use any support vector (X_i,y_i) to solve b from the property $y_i(w^TX_i+b)=1$ of support vectors. More robustly, if we have s support vectors, we can calculate b as follows.

$$b=\frac{1}{s}\sum_{\substack{i=1\\ \alpha_i>0}}^{n}\left(\frac{1}{y_i}-w^TX_i\right)=-\frac{1}{2}\left(\min_{i:y_i=+1}w^TX_i+\max_{i;y_i=-1}w^TX_i\right)$$

[1] Platt, John. "Sequential minimal optimization: A fast algorithm for training support vector machines." (1998).

In conclusion, w and b form the support vector machine (SVM) model $f(X_i) = w^T X_i + b$, and the classification is based on the perceptron model, that is,

$$\widehat{y_i} = \text{sign}\big(f(X_i)\big) = \text{sign}(w^T X_i + b)$$

For inference, we have $f(X_i) = \langle w, X_i \rangle + b$, where w only contains those support vectors with $\alpha_i > 0$.

2.6 Kernel-based SVM

Let X denote a sample vector, and let $\phi(X)$ be the vector after transformation ϕ. *e.g.*, if we set ϕ as

$$\phi(X) = \left[x_1, x_2, \cdots, x_p, x_1^2, x_2^2, \cdots, x_p^2, x_1^3, x_2^3, \cdots, x_p^3\right]$$

Then, the corresponding $\phi(X)$ is a $3p$-dimensional vector when X is a p-dimensional vector.

Why do we need the transformation ϕ? We may use a more high-ordered feature vector to replace the original feature vector, which limits the SVM in the original workspace. Hence, let us calculate the inference process of SVM as follows.

$$w^T \phi(X_i) + b = \sum_{\substack{j=1 \\ \alpha_j > 0}}^{n} \alpha_j y_j \langle \phi(X_j), \phi(X_i) \rangle + b$$

If we define **kernel** $K(X_i, X_j) = \phi(X_i)^T \phi(X_j) = \langle \phi(X_i), \phi(X_j) \rangle$, then

$$w^T \phi(X_i) + b = \sum_{\substack{j=1 \\ \alpha_j > 0}}^{n} \alpha_j y_j K(X_j, X_i) + b$$

Therefore, we do not even need to calculate w, since b can be calculated as

$$b = \frac{1}{s} \sum_{\substack{i=1 \\ \alpha_i > 0}}^{n} \left(\frac{1}{y_i} - \sum_{j=1}^{n} \alpha_j y_j K(X_j, X_i) \right)$$

We can replace all $X_i^T X_j$ or $\langle X_i, X_j \rangle$ by $K(X_i, X_j)$ to generalize the SVM to the kernel space. Here we want to emphasize that sometimes $\phi(\cdot)$ may be difficult or even unable to compute, but we only need to define $K(\cdot, \cdot)$ in kernel-based SVM, *i.e.*, we may directly focus on $K(X_i, X_j)$ without considering the computation of $\phi(X_i)$.

2.7 Kernel

Mercer Theorem[2].

Let $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be given as the kernel function. Then, K is a valid (Mercer) kernel if and only if

(1) The kernel matrix is symmetric, that is, $K(X_i, X_j) = \phi(X_i)^T \phi(X_j) = \phi(X_j)^T \phi(X_i) = K(X_j, X_i)$;

(2) The kernel matrix is positive semi-definite, *i.e.*, if we define the kernel matrix K as an $n \times n$ matrix, where $K_{ij} = K(X_i, X_j)$, then for arbitrary vector z, we have

$$\begin{aligned}
z^T K z &= \sum_{i=1}^{n} \sum_{j=1}^{n} z_i K_{ij} z_j \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} z_i \phi(X_i)^T \phi(X_j) z_j \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} z_i z_j \sum_{k=1}^{p} \phi_k(X_i) \phi_k(X_j) \\
&= \sum_{k=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(z_i \phi_k(X_i)\right) \left(z_j \phi_k(X_j)\right) \\
&= \sum_{k=1}^{p} \left(\sum_{i=1}^{n} z_i \phi_k(X_i) \right)^2 \geq 0
\end{aligned}$$

---

[2] Mercer, J. "Functions ofpositive and negativetypeand theircommection with the theory ofintegral equations." *Philosophicol Trinsdictions ofthe Rogyal Society* (1909): 4-415.

RBF Kernel / Gaussian Kernel.

$$K(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right)$$

where $\sigma > 0$ is a hyper-parameter. The Gaussian kernel measures the similarity between X_i and X_j. If X_i is close to X_j, then $K(X_i, X_j) \to 1$; on the contrary, if X_i is far from X_j, $K(X_i, X_j) \to 0$.

The transformation ϕ for Gaussian kernel actually maps the initial feature to an infinite-dimensional space[3], since we can interpret $K(X_i, X_j)$ as follows. Here, let us assume $\sigma^2 = 1$ for convenience.

$$K(X_i, X_j) = \exp\left(-\frac{X_i^2 + X_j^2 - 2\langle X_i, X_j\rangle}{2}\right) = \exp\left(-\frac{X_i^2 + X_j^2}{2}\right)\exp\langle X_i, X_j\rangle$$

According to Taylor expansion, we have

$$K(X_i, X_j) = \exp\left(-\frac{X_i^2 + X_j^2}{2}\right)\exp\langle X_i, X_j\rangle = \exp\left(-\frac{X_i^2 + X_j^2}{2}\right)\sum_{k=1}^{+\infty}\frac{\langle X_i, X_j\rangle^k}{k!}$$

where $\langle X_i, X_j\rangle^k$ can enumerates every k-ordered polynomials formed by X_i and X_j. Since we enumerate k from 1 to infinite, we can get all the polynomials formed by X_i and X_j, which means that the transformation function is able to map X into an infinite space of $X, X^2, X^3, \cdots$. Hence, the Gaussian kernel is also called Radial Base Function (RBF) kernel.

Laplace Kernel.

$$K(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|}{\sigma}\right)$$

where $\sigma > 0$ is a hyper-parameter.

Linear Kernel.

$$K(X_i, X_j) = X_i^T X_j$$

which is the ordinary kernel we used in the previous discussions.

Simple Polynomial Kernel.

$$K(X_i, X_j) = \left(X_i^T X_j\right)^d$$

where $d \geq 1$ is a hyper-parameter.

Cosine Similarity Kernel.

$$K(X_i, X_j) = \frac{X_i^T X_j}{\|X_i\|\|X_j\|}$$

Sigmoid Kernel.

$$K(X_i, X_j) = \tanh\left(\alpha X_i^T X_j + c\right)$$

where

$$\tanh(a) = \frac{1 - \exp(-2a)}{1 + \exp(-2a)}$$

And $\alpha > 0, c < 0$ are hyper-parameters.

Rational Quadratic Kernel[4].

$$K(X_i, X_j) = 1 - \frac{\|X_i - X_j\|^2}{\|X_i - X_j\|^2 + c}$$

where $c > 0$ is a hyper-parameters. It can be used as an alternative to RBF kernel due to less computations.

Kernel Properties.

1. If K_1, K_2 are kernel functions, then for arbitrary positives γ_1, γ_2, $\gamma_1 K_1 + \gamma_2 K_2$ is a kernel function;
2. If K_1, K_2 are kernel functions, then $K_1 \otimes K_2 = K_1(X_i, X_j)K_2(X_i, X_j)$ is a kernel function;
3. If K is a kernel function, then for arbitrary function g, $g(X_i)K(X_i, X_j)g(X_j)$ is a kernel function.

---

[3] http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf

[4] http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

2.8 SVM with Outliers

The previous discussions based on an assumption that the data can be well separated. But when the data cannot be well separated, *i.e.*, there are some outliers in the data, we can formulate the objectives as follows.

$$\begin{aligned} \min_{\xi,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(w^T X_i + b) \geq 1 - \xi_i, \quad i = 1,2,\cdots,n \\ & \xi_i \geq 0, \quad i = 1,2,\cdots,n \end{aligned}$$

which is equivalent to

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

where $\max\big(0,1 - y_i(w^T X_i + b)\big)$ is the hinge loss introduced in Chapter 1, and C is a hyper-parameter. The hinge loss does not only penalize negative margins, it also penalizes margins less than 1.

Thus, the Langrangian is given as

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(w^T X_i + b) - 1 + \xi_i] - \sum_{i=1}^{n}\beta_i\xi_i$$

and the objective is

$$\min_{w,b,\xi:\xi_i\geq 0}\ \max_{\alpha,\beta:\alpha_i\geq 0,\beta_i\geq 0} L(w,b,\xi,\alpha,\beta)$$

According to the Karush-Kuhn-Tucker conditions, we know that

$$\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial w} = 0, \qquad \frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial b} = 0, \qquad \frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial \xi} = 0$$

which leads to

$$w = \sum_{i=1}^{n}\alpha_i y_i X_i\,, \qquad \sum_{i=1}^{n}\alpha_i y_i = 0, \qquad C - \beta_i - \alpha_i = 0\ (i = 1,2,\cdots,n)$$

Hence,

$$\begin{aligned} L(w,b,\xi,\alpha,\beta) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(w^T X_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i \\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j X_i^T X_j + \sum_{i=1}^{n}\xi_i[C - \alpha_i - \beta_i] \\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j X_i^T X_j \end{aligned}$$

Surprisingly, we derive the same result as result in section 2.5 where we do not consider about outliers. Notice that the objective function doesn't contain β_i, so we want to eliminate β_i completely. Thus consider the condition $C - \beta_i - \alpha_i = 0$ and the requirements $\beta_i \geq 0$, we can derive that

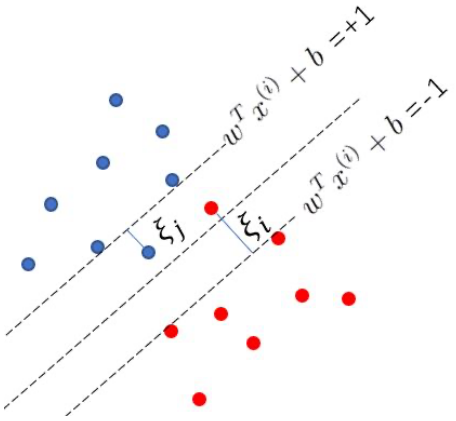
$$\alpha_i = C - \beta_i \leq C$$

Therefore, we only need to optimize with regard to α, shown as follows.

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j X_i^T X_j \\ \text{s.t.} \quad & \forall i, 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned}$$

which can also be solved using Sequential Minimal Optimization mentioned before. b can be calculated using the same method in section 2.5. Then we can analyze the result and make the following discussions.

1. If $\alpha_i = 0$, then $\beta_i = C - \alpha_i = C > 0$, we must have
$$\xi_i = 0$$
that is,
$$y_i(w^T X_i + b) > 1 - \xi_i = 1$$
2. If $\alpha_i = C > 0$, then $\beta_i = C - \alpha_i = 0$. Hence, we can have arbitrary ξ_i. Naturally, we will choose $\xi_i \geq 0$ that ensures
$$y_i(w^T X_i + b) = 1 - \xi_i$$
Therefore, we have
$$y_i(w^T X_i + b) < 1$$
3. If $0 < \alpha_i < C$, then $0 < \beta_i = C - \alpha_i < C$, we must have
$$\xi_i = 0, \qquad y_i(w^T X_i + b) = 1 - \xi_i$$
that is,
$$y_i(w^T X_i + b) = 1$$
which indicates that (X_i, y_i) is the support vector.



2.9 Structural Risk, Empirical Risk and Regularization

In previous section, we have derived that the optimization problem of SVM with outliers is equivalent to

$$\min_{\xi,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i(w^T X_i + b)\right)$$

where $\max\left(0, 1 - y_i(w^T X_i + b)\right)$ is hinge loss. The loss of SVM with outliers has the form of the loss of ridge regression, which will be introduced in Chapter 3.2. So let us rewrite the optimization into a general form.

$$\min_{f} \Omega(f) + C\sum_{i=1}^{n} L(f(X_i), y_i)$$

where $L(\hat{y}_i, y_i)$ is the loss function, and the first term only relates to the model f. Actually, we often call the first term $\Omega(f)$, which is used to describe some certain properties of f, the **structural risk**; and the second term is called the **empirical risk**, which is used to describe the consistent of the model predictions and the ground-truth labels. C is used as a trade-off variable between the structural risk and the empirical risk.

In fact, $\Omega(f)$ describes the model that we want to get, which is also used to describe the desired property of the model f, and it can reduce the risk of over-fitting on the training data. From this perspective, the previous objective is called "**regularization**" problem, which will be further discussed in Chapter 3, and $\Omega(f)$ is the regularization term while C is the regularization constant. We usually let $\Omega(f)$ be L_p norm defined as follows.

$$\mathrm{L}_p(x) = \|x\|_{\ell_p} = \left(\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}}$$

Noteworthily, L_0 norm is specially defined as $\mathrm{L}_0(x) = |\{x_i \neq 0 \mid i = 1,2,\cdots,n\}|$.

The L_p norm usually corresponds to the "distance" between the origin and x. For example, the L_1 norm is the Manhattan distance between origin and x, *i.e.*,

$$\|x\|_{\ell_1} = \mathrm{L}_1(x) = \sum_{i=1}^{n} x_i$$

while L_2 norm is the Euclidean distance between origin and x, *i.e.*,

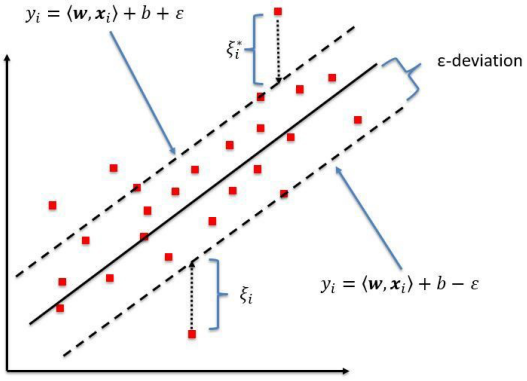
$$\|x\|_{\ell_2} = \mathrm{L}_2(x) = \sqrt{\sum_{i=1}^{n} x_i^2} \overset{\text{def}}{=} \|x\|^2$$

and L_∞ norm is the Chebyshev distance between origin and x, *i.e.*,

$$\|x\|_{\ell_\infty} = \mathrm{L}_\infty(x) = \max_{i=1,2,\cdots,n} x_i$$

2.10 Support Vector Regression

Given training samples $\{(X_i, y_i)\}_{i=1,2,\cdots,n}$, and the regression problem wants to learn a regression model with the form of $f(X_i) = w^T X_i + b$ to predict y_i. The conventional regression problem use the difference between the predicted value and the ground-truth to calculate loss function, but Support Vector Regression (SVR) supposes that we allow at least ε difference between the predicted value and the ground-truth. The range that SVR accepts is called ***ε*-deviation**. Hence, we rewrite the objective of SVR as follows.



$$\begin{aligned} \min_{\xi,\hat{\xi},w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \hat{\xi}_i\right) \\ \text{s.t.} \quad & -\varepsilon - \hat{\xi}_i \le y_i - w^T X_i - b \le \varepsilon + \xi_i, \quad i = 1, 2, \cdots, n \\ & \xi_i \ge 0,\ \hat{\xi}_i \ge 0, \quad i = 1, 2, \cdots, n \end{aligned}$$

which is equivalent to

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} L_\varepsilon(w^T X_i + b, y_i)$$

where C is the regularization constant, and $L_\varepsilon(\hat{y}, y)$ is the ***ε*-intensive loss function** defined as follows.

$$L_\varepsilon(\hat{y}, y) = \begin{cases} 0 & |\hat{y} - y| \le \varepsilon \\ |\hat{y} - y| - \varepsilon & |\hat{y} - y| > \varepsilon \end{cases}$$

We use the same methods to solve the optimization problem by constructing generalized Langrangian $L(w, b, \xi, \hat{\xi}, \alpha, \hat{\alpha}, \beta, \hat{\beta})$ as follows.

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \hat{\xi}_i) + \sum_{i=1}^{n}\hat{\alpha}_i[w^T X_i + b - y_i - \varepsilon - \hat{\xi}_i] + \sum_{i=1}^{n}\alpha_i[y_i - w^T X_i - b - \varepsilon - \xi_i] - \sum_{i=1}^{n}\beta_i\xi_i - \sum_{i=1}^{n}\hat{\beta}_i\hat{\xi}_i$$

According to the Karush-Kuhn-Tucker condition,

$$\frac{\partial L(w, b, \xi, \hat{\xi}, \alpha, \hat{\alpha}, \beta, \hat{\beta})}{\partial w} = 0, \qquad \frac{\partial L(w, b, \xi, \hat{\xi}, \alpha, \hat{\alpha}, \beta, \hat{\beta})}{\partial b} = 0,$$
$$\frac{\partial L(w, b, \xi, \hat{\xi}, \alpha, \hat{\alpha}, \beta, \hat{\beta})}{\partial \xi} = 0, \qquad \frac{\partial L(w, b, \xi, \hat{\xi}, \alpha, \hat{\alpha}, \beta, \hat{\beta})}{\partial \hat{\xi}} = 0$$

that is,

$$w = \sum_{i=1}^{n}(\hat{\alpha}_i - \alpha_i)X_i, \qquad \sum_{i=1}^{n}(\hat{\alpha}_i - \alpha_i) = 0$$
$$\alpha_i + \beta_i = \hat{\alpha}_i + \hat{\beta}_i = C, \qquad i = 1,2,\cdots,n$$

Hence, after some calculation, we can derive that

$$L(w, b, \xi, \hat{\xi}, \alpha, \hat{\alpha}, \beta, \hat{\beta}) = \sum_{i=1}^{n}\left(y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon(\hat{\alpha}_i + \alpha_i)\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)X_i^T X_j$$

Thus, re-write the objective as follows to optimize α and $\hat{\alpha}$.

$$\begin{aligned} \max_{\alpha,\hat{\alpha}} \quad & W(\alpha, \hat{\alpha}) = \sum_{i=1}^{n}(y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon(\hat{\alpha}_i + \alpha_i)) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)X_i^T X_j \\ \text{s.t.} \quad & \forall i, 0 \le \alpha_i, \hat{\alpha}_i \le C \\ & \sum_{i=1}^{n}(\hat{\alpha}_i - \alpha_i)y_i = 0 \end{aligned}$$

Still, we can use Sequential Minimal Optimization algorithm to calculate the optimal α and $\hat{\alpha}$. Notice that, the optimization process should follow the Karush-Kuhn-Tucker condition, that is,

$$\begin{cases} \hat{\alpha}_i\left(w^T X_i + b - y_i - \varepsilon - \hat{\xi}_i\right) = 0 \\ \alpha_i(y_i - w^T X_i - b - \varepsilon - \xi_i) = 0 \\ \hat{\beta}_i\hat{\xi}_i = (C - \hat{\alpha}_i)\hat{\xi}_i = 0 \\ \beta_i\xi_i = (C - \alpha_i)\xi_i = 0 \end{cases}$$

Then, we can analyze the result and make the following discussions.

1. At least one of α_i and $\hat{\alpha}_i$ is 0, since $w^T X_i + b - y_i - \varepsilon - \hat{\xi}_i = 0$ and $y_i - w^T X_i - b - \varepsilon - \xi_i = 0$ cannot hold in the same time. Similarly, at least one of ξ_i and $\hat{\xi}_i$ is 0.
2. If $\alpha_i > 0$ and $\hat{\alpha}_i = 0$, then y_i falls in $[w^T X_i + b + \varepsilon, +\infty)$. Moreover, if $0 < \alpha_i < C$, then
$$y_i = w^T X_i + b + \varepsilon$$
that is, sample (X_i, y_i) is on the down-margin of ε-deviation.
3. If $\hat{\alpha}_i > 0$ and $\alpha_i = 0$, then y_i falls in $(-\infty, w^T X_i + b - \varepsilon]$. Moreover, if $0 < \hat{\alpha}_i < C$, then
$$y_i = w^T X_i + b - \varepsilon$$
that is, sample (X_i, y_i) is on the up-margin of ε-deviation.
4. If both α_i and $\hat{\alpha}_i$ are 0, then y_i falls in $[w^T X_i + b - \varepsilon, w^T X_i + b + \varepsilon]$, *i.e.*, the ε-deviation.

Then, the inference phase of SVR model is

$$f(X_i) = w^T X_i + b = \sum_{j=1}^{n}\left(\hat{\alpha}_j - \alpha_j\right)X_j^T X_i + b$$

Hence, those samples (X_i, y_i) with $\hat{\alpha}_i - \alpha_i \neq 0$ are the support vectors, which falls out of the ε-deviation. Notice here, the definition of support vectors has a slight difference with the previous definition – it is not referenced to the samples on the margin, but samples out of ε-deviation!

Then, b can be calculated using arbitrary sample (X_i, y_i) that is on the margin of ε-deviation by

$$b = y_i - w^T X_i - \varepsilon = y_i - \sum_{j=1}^{n}\left(\hat{\alpha}_j - \alpha_j\right)X_j^T X_i - \varepsilon$$

or

$$b = y_i - w^T X_i + \varepsilon = y_i - \sum_{j=1}^{n}\left(\hat{\alpha}_j - \alpha_j\right)X_j^T X_i + \varepsilon$$

according to which side of the margin the sample falls in. More robustly, b can be calculated as the average result of the previous formula from all the samples on the margin of ε-deviation.

Kernel-based SVR is similar to the ordinary SVR. We can rewrite the inference of SVR model as

$$f(X_i) = w^T\phi(X_i) + b = \sum_{j=1}^{n}\left(\hat{\alpha}_j - \alpha_j\right)K\left(X_j, X_i\right) + b$$

Also, we do not need to calculate the value of w, and we can calculate b as follows. Notice that here we only focus on the samples on the up-margin, since the samples on the down-margin are similar.

$$b = y_i - w^T\phi(X_i) + \varepsilon = y_i - \sum_{j=1}^{n}\left(\hat{\alpha}_j - \alpha_j\right)K\left(X_j, X_i\right) + \varepsilon$$

2.11 Kernel Methods

Given training samples $\{(X_i, y_i)\}_{i=1,2,\cdots,n}$, both SVM and SVR learn a model that is represented as the linear combination of the kernel function. Generally, we have the following representation theorem.

Representation Theorem.

Let $\mathbb{H}$ be the reproducing kernel Hilbert space (RKHS), $\|h\|_{\mathbb{H}}$ denote the norm of h in $\mathbb{H}$. Then, for any monotonically increasing function $\Omega\colon [0, +\infty] \to \mathbb{R}$ and arbitrary non-negative loss function $\mathcal{L}\colon \mathbb{R}^n \to [0, +\infty]$, the optimal solution of the optimization problem

$$\min_{h\in\mathbb{H}} F(h) = \Omega(\|h\|_{\mathbb{H}}) + \mathcal{L}\left((X_1, y_1), (X_2, y_2), \cdots, (X_n, y_n)\right)$$

can be written as

$$h^*(X) = \sum_{i=1}^{n} \alpha_i K(X, X_i)$$

The representation theorem shows that for most of the optimization problem with general regularization term and loss function, the optimal solution can be represented as the linear combination of kernel functions.

Kernel Methods.

The methods based on kernel functions are called kernel methods. Basically, we can introduce kernel function to make the model non-linear, *e.g.*, we can generalize the Linear Discriminative Analysis (LDA) model introduced in section 1.13 to Kernelized Linear Discriminative Analysis (KLDA) model. For more details of this example, you can check 'machine learning' book[5] yourselves.

Appendix: The Sequential Minimal Optimization Algorithm.

The Sequential Minimal Optimization (SMO) algorithm is based on a simple idea: let all other parameters except α_i fixed, then calculate the minimal value with regard to α_i. Since we have the constraint that

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

we must set another parameter α_j free while keep others fixed. Therefore, after initialization, the Sequential Minimal Optimization algorithm keeps executing the following steps until convergence.

(1) Choose one parameter pair (α_i, α_j) that needs to be updated;

(2) Keep other parameters fixed and solve the optimization problem of α_i and α_j, and update the result.

Notice that if one of α_i, α_j violates the Karush-Kuhn-Tucker condition, the objective will increase after updating parameters[6]. Intuitively, if a parameter violates Karush-Kuhn-Tucker condition severely, then the objective will increase a lot after updating. Therefore, the Sequential Minimal Optimization first choose the parameter that violates Karush-Kuhn-Tucker condition to the greatest degree. The other parameters should choose one with the fastest increasing speed, but it is impractical since the complexity is too high. Thus, the Sequential Minimal Optimization algorithm uses a heuristic method: maximize the distance between two chosen samples, which may bring large improvement to the objectives.

Now, we revise the optimization problem that we need to solve, which is shown as follows.

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j X_i^T X_j \\ \text{s.t.} \quad & \forall i, 0 \le \alpha_i \le C \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

If we keep α_i and α_j fixed, then the constraints can be rewritten to

$$\alpha_i y_i + \alpha_j y_j = c = -\sum_{\substack{k=1 \\ k \neq i,j}}^{n} \alpha_k y_k$$

Hence, we can use α_i to represent α_j, and plug the result in the objective, then we can get a single-variable quadratic optimization problem with only constraint $0 \le \alpha_i \le C$, which can be solved easily.

---

[5] Zhou Zhi-Hua, *Machine Learning*, Tsinghua University Press, 2016.

[6] Osuna, Edgar, Robert Freund, and Federico Girosit. "Training support vector machines: an application to face detection." *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997.