Support Vector Machines

Quanshi Zhang

March 27, 2020

Contents

Some contents are from `http://cs229.stanford.edu/notes/cs229-notes3.pdf`.
All the figures are taken from the web.

1 Support vector machines

1.1 Margin and support vectors

Logistic regression

Let $D = \{X_i, y_i\}_{i=1,\ldots,n}$ 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 the two classes of samples.

$p(y_i = 1|X_i)$ is modeled as

$$p(y_i = 1|X_i) = p_i = \text{sigmoid}(X_i^\top \beta) = \frac{e^{X_i^\top \beta}}{1 + e^{-X_i^\top \beta}} = \frac{1}{1 + e^{-X_i^\top \beta}},$$

If $p_i \geq 0.5$, if and only if $X_i^\top \beta \geq 0$.

- If $X_i^\top \beta \gg 0$, p_i approximates 1, and the model predicts $y_i = 1$ with a high confidence.
- If $X_i^\top \beta \ll 0$, p_i approximates 0, and the model predicts $y_i = 0$ with a high confidence.
- If $-\tau \leq X_i^\top \beta \leq \tau$, then the prediction result is sensitive to the noise.

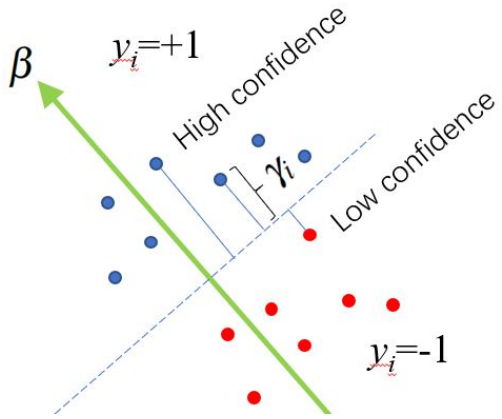

Figure 1: Linear classification

For classification

Notation: We use $y_i \in \{-1,+1\}$ instead of $\{0,1\}$ to denote the class label. In addition, we use the linear weight w and the bias term b to replace β. Here, we can write the classifier as

$$\begin{aligned}
\hat{y}_i &= sign(X_i^\top \beta) \\
&= sign(X_{i1}\beta_1 + X_{i2}\beta_2 + \cdots + X_{ip}\beta_p) \\
&= sign(\beta_1 + X_{i2}\beta_2 + \cdots + X_{ip}\beta_p), \qquad \text{because } X_{i1} = 1 \\
X_i &\leftarrow [X_{i2}, X_{i3}, \ldots, X_{ip}]^\top \\
b &\leftarrow \beta_1 \\
w &\leftarrow [\beta_2, \beta_3, \ldots, \beta_p]^\top
\end{aligned}$$

Thus, we get

$$\hat{y} = sign(w^\top x + b) = \begin{cases} +1, & w^\top x + b \geq 0 \\ -1, & w^\top x + b < 0 \end{cases}$$

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

$$\gamma_i = y_i(w^\top x_i + b)$$

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

Review of the Perceptron
A deterministic version of the logistic regression is the perceptron model $y_i = \text{sign}(w^\top x_i + b)$, where $y_i \in \{+1,-1\}$. The gradient-descend learning algorithm can be modified into the perceptron algorithm. Starting from $\beta_0 = 0$,

$$w^{(t+1)} = w^{(t)} + \sum_{i=1}^{n} \delta_i y_i x_i,$$

where $\delta_i = 1(y_i \neq \text{sign}(w^\top x_i + b))$ to determine whether $w^{(t)}$ makes a mistake in classifying y_i.
The algorithm can be considered the gradient descent algorithm for the loss function

$$loss(w,b) = \sum_{i=1}^{n} \max(0, -y_i(w^\top x_i + b)) = \sum_{i=1}^{n} \max(0, -margin_i),$$

where $\max(0, -margin_i) = 0$ if $margin_i \geq 0$, i.e., no mistake is made, and $\max(0, -margin_i) = -margin_i$ if $margin_i < 0$. Again the algorithm learns from the mistakes.

Considering that if we set $w \leftarrow 3w$, $b \leftarrow 3b$, then the model will not change the classification result, because the classification result only depends on the sign of $w^\top x + b$. However, this changes the functional margin $\gamma_i \leftarrow 3\gamma_i$.

Thus, we normalize the weight

$$w \leftarrow \frac{w}{\|w\|_2}$$

where $\|w\|_2$ denotes the L-2 norm of w

$$\|w\|_2 = \sqrt{\sum_j w_j^2}$$

Thus, the modified functional margin is given as

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

In this case, the point $X_i - \gamma_i \frac{w}{\|w\|} y_i$ must localize on the decision boundary.
Proof

$$w^\top \left(X_i - \gamma_i \frac{w}{\|w\|} y_i \right) + b = (w^\top X_i + b) - y_i \frac{w^\top (w^\top X_i + b) w}{\|w\|^2} = (w^\top X_i + b) - (w^\top X_i + b) = 0, \qquad \text{//because } y_i^2 = 1$$

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

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

1.2 Margin classifier

Let us assume that all training samples $\{(X_i, y_i)\}$ can be correctly classified. Then, given all classifiers that achieve 100% accuracy, which is the best margin classifier?

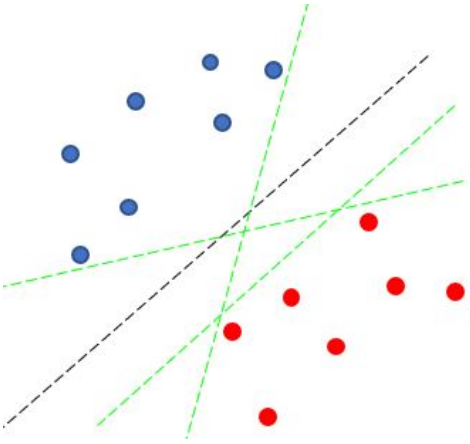

Figure 2: Which is the best margin classifier?

We focus on the following objective

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

in order to maximize the margin.

To simplify the computation, we can rewrite the objective as

$$\max_{\tau,w,b} \frac{\tau}{\|w\|}$$
$$\text{s.t.} \qquad \forall i,\ y_i(w^\top X_i + b) \geq \tau$$

Here $\hat{\gamma} = \frac{\tau}{\|w\|}$.

In previous paragraphs, we control the scale of w to $\|w\| = 1$. In fact, we alternatively control τ instead of w to simplify the computation. It is because changing the scale of w does not affect the classification performance.

For example, we may set

$$\tau = 1$$

and learn the optimal w and b to maximize $\frac{\tau}{\|w\|}$ with respect to $\forall i,\ y_i(w^\top X_i + b) \geq \tau$. The above objective is equivalent to

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \qquad \forall i,\ y_i(w^\top X_i + b) \geq 1$$

Lagrange multipliers

Let us consider the following problem.

$$\min_{w} f(w)$$
$$\text{s.t.} \qquad h_k(w) = 0, k = 1, 2, ..., K$$

We can minimize the following loss for optimization.

$$L(w, \lambda) = f(w) + \sum_{k=1}^{K} \lambda_k h_k(w)$$

Here, λ_k is termed the lagrange multiplier. The minimization of the above loss function requires

$$\frac{\partial L}{\partial w} = 0, \qquad \forall k,\ \frac{\partial L}{\partial \lambda_k} = 0$$

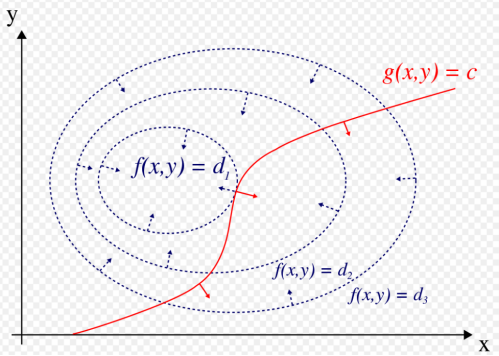

Figure 3: Energy landscape of f and g. $w = [x, y]^\top$.

How to understand Lagrange multipliers?

Let us consider

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

Let us slightly change w by Δw along the curve of $g(w) = 0$. Then, the orientation of Δw must be orthogonal to the gradient $\frac{\partial g}{\partial w}$; otherwise, $g(w + \Delta w) \neq 0$, *i.e.*

$$\forall \Delta w, \quad \Delta w^\top \frac{\partial g}{\partial w} = 0$$

On the other hand, let w be a solution to $\min_w f(w)$ s.t. $g(w) = 0$, then

$$\forall \Delta w, \quad \Delta w^\top \frac{\partial f}{\partial w} = 0, \qquad \text{s.t.} \quad \Delta w^\top \frac{\partial g}{\partial w} = 0;$$

otherwise, w does not satisfy $\min_w f(w)$ s.t. $g(w) = 0$.

Thus, if we can find a scalar λ that ensures

$$\frac{\partial f}{\partial w} + \lambda \frac{\partial g}{\partial w} = 0, \qquad \text{Imply that } \frac{\partial L}{\partial w} = 0$$
$$g(w) = 0, \qquad \text{Imply that } \frac{\partial L}{\partial \lambda} = 0$$

then we find a solution to $\min_w f(w)$ s.t. $g(w) = 0$.

Thus, we optimize

$$L(w, \lambda) = f(w) + \lambda g(w)$$

and ensure

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

Then, let us consider a more generalized case with both equality constraints and inequality constraints.

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

Accordingly, we define the generalized Lagrangian

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

Then, consider the quantity

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

If either $g_k(w) > 0$ or $h_l(w) \neq 0$, it is easy to prove $V(w) = +\infty$, *i.e.*

$$V(w) = \begin{cases} f(w), & \forall k,\ g_k(w) \leq 0,\ \forall l,\ h_l(w) = 0 \\ +\infty, & \text{otherwise} \end{cases}$$

Thus

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

We can view the following problem from a slightly different perspective. Let us define

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

Then, we construct a dual optimization problem.

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

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

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

Proof

$$\begin{aligned}
& \forall w,\alpha,\beta:\alpha_k\geq 0, \quad \min_w L(w,\alpha,\beta) \leq L(w,\alpha,\beta) \\
\Rightarrow \quad & \forall w, \quad \max_{\alpha,\beta:\alpha_k\geq 0} \min_w L(w,\alpha,\beta) \leq \max_{\alpha,\beta:\alpha_k\geq 0} L(w,\alpha,\beta) \\
\Rightarrow \quad & \max_{\alpha,\beta:\alpha_k\geq 0} \min_w L(w,\alpha,\beta) \leq \min_w \max_{\alpha,\beta:\alpha_k\geq 0} L(w,\alpha,\beta)
\end{aligned}$$

Then, let us discuss the condition that ensures

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

which is termed **Karush-Kuhn-Tucker (KKT) conditions**, as follows.

$$\begin{aligned}
\forall p, \quad \frac{\partial}{\partial w_p} L(w,\alpha,\beta) &= 0 \\
\forall l, \quad \frac{\partial}{\partial \beta_l} L(w,\alpha,\beta) &= 0 \\
\forall k, \quad \alpha_k g_k(w) &= 0 \\
\forall k, \quad g_k(w) &\leq 0 \\
\forall k, \quad \alpha_k &\geq 0
\end{aligned}$$

Learning and support vectors

We can rewrite the objective of learning the classifier as

$$\begin{aligned}
& \min_{w,b} \frac{1}{2}\|w\|^2 \\
\text{s.t.} \quad & \forall i, \ -y_i(w^\top X_i + b) + 1 \leq 0
\end{aligned}$$

According to the KKT condition, we will have $\alpha_i > 0$ for training samples that ensures $-y_i(w^\top X_i + b) + 1 = 0$. We call these samples **support vectors**.

- Support vectors are usually much less than training samples.
- Support vectors usually correspond to samples that are difficult to classify.

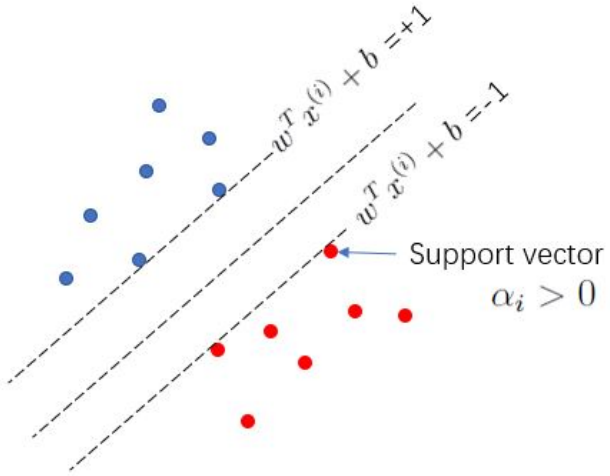

Figure 4: Support vectors

Optimization

We construct the Lagrangian to learn the model, and the loss function is given as

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \Big[y_i(w^\top X_i + b) - 1 \Big], \qquad \min_{w,b} \max_{\alpha:\alpha_i \geq 0} L(w,b,\alpha)$$

where we only consider α, and there does not exist the β term.

- **Step 1:** First, fix α, learn w and b to minimize $L(w,b,\alpha)$

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

  Considering $\frac{\partial L(w,b,\alpha)}{\partial w} = 0$, we get

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

  Then, let us consider b

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

  Considering $\frac{\partial L(w,b,\alpha)}{\partial b} = 0$, we get

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

Considering $w = \sum_{i=1}^n \alpha_i y_i X_i$ and $\sum_{i=1}^n \alpha_i y_i = 0$, we get

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^n \alpha_i \Big[y_i (w^\top X_i + b) - 1 \Big] \\ &= \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j X_i^\top X_j - \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j X_i^\top X_j - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i \\ &= \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j X_i^\top X_j - b \sum_{i=1}^n \alpha_i y_i \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j X_i^\top X_j \end{aligned}$$

- **Step 2:** Thus, we can rewrite the objective as follows to optimize α.

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

  Sophisticated methods can be used to optimize the above equation.

  Given w, we get the optimal $b = -\frac{1}{2}(\min_{i:y_i=+1} w^\top X_i + \max_{i:y_i=-1} w^\top X_i)$.

- **Step 3:** Go back to **Step 1**.

Thus, we can rewrite the inference.

$$\begin{aligned} w^\top X_i + b &= \left(\sum_{j=1}^n \alpha_j y_j X_j \right)^\top X_i + b \\ &= \sum_{j=1}^n \alpha_j y_j \langle X_j, X_i \rangle + b \\ &= \sum_{j:\alpha_j > 0} \alpha_j y_j \langle X_j, X_i \rangle + b \\ &= \langle w, X_i \rangle + b \qquad \text{using support vectors for inference} \end{aligned}$$

where

$$w = \sum_{j:\alpha_j > 0} \alpha_j y_j X_j.$$

1.3 Kernel-based SVM

How to mimic a cubic function?

Let x denote a feature vector (we omit the subscript i of X_i for clarity).

$$\phi(x) = [x_1, x_2, \ldots, x_p, x_1^2, x_2^2, \ldots, x_p^2, x_1^3, x_2^3, \ldots, x_p^3]^\top$$

Thus, the corresponding w is a $3p$-dimensional vector. The linear SVM can be represented as

$$w^\top \phi(x) + b = \sum_{i=1}^{p} (w_{3i-2} x_i + w_{3i-1} x_i^2 + w_{3i} x_i^3) + b$$

Sometimes, we need to use $\phi(x)$ instead of x as features to deal with non-linear classification problems.

$$\begin{aligned} w^\top \phi(X_i) + b &= \left(\sum_{j=1}^{n} \alpha_j y_j \phi(X_j) \right)^\top \phi(X_i) + b \\ &= \sum_{j=1}^{n} \alpha_j y_j \langle \phi(X_j), \phi(X_i) \rangle + b \\ &= \sum_{j:\alpha_j > 0} \alpha_j y_j \langle \phi(X_j), \phi(X_i) \rangle + b \\ &= \sum_{j:\alpha_j > 0} \alpha_j y_j K(X_j, X_i) + b \end{aligned}$$

where we define

$$K(X_i, X_j) \stackrel{\text{def}}{=} \phi(X_i)^\top \phi(X_j)$$

as the **Kernel**.

We may simply replace $\langle X_i, X_j \rangle$ with $K(X_i, X_j)$ in all previous equations.

Sometimes, $K(X_i, X_j)$ is cheap to compute, but $\phi(X_i)$ may be expensive to compute. For example,

$$K(X_i, X_j) \stackrel{\text{def}}{=} (X_i^\top X_j)^2$$

we get

$$\begin{aligned} K(X_i, X_j) &= \left(\sum_{k=1}^{p} X_{ik} X_{jk} \right) \left(\sum_{k=1}^{p} X_{ik} X_{jk} \right) \\ &= \sum_{k=1}^{p} \sum_{l=1}^{p} X_{ik} X_{jk} X_{il} X_{jl} \\ &= \sum_{k,l} (X_{ik} X_{il})(X_{jk} X_{jl}) \end{aligned}$$

Thus, we can get

$$\phi(X_i) = \begin{bmatrix} X_{i1}X_{i1} \\ X_{i1}X_{i2} \\ \vdots \\ X_{i1}X_{ip} \\ X_{i2}X_{i1} \\ X_{i2}X_{i2} \\ \vdots \\ X_{i2}X_{ip} \\ \vdots \\ \vdots \\ X_{ip}X_{i1} \\ X_{ip}X_{i2} \\ \vdots \\ X_{ip}X_{ip} \end{bmatrix}$$

The cost of calculating $\phi(X_i)$ is $O(p^2)$, but the cost of calculating $K(X_i, X_j)$ is $O(p)$.

For the kernel of

$$\begin{aligned} K(X_i, X_j) &= (X_i^\top X_j + c)^2 \\ &= \sum_{k,l}(X_{ik}X_{il})(X_{jk}X_{jl}) + \sum_{k=1}^{p}(\sqrt{2c}X_{ik})(\sqrt{2c}X_{jk}) + c^2 \end{aligned}$$

Thus, we can get

$$\phi(X_i) = \begin{bmatrix} X_{i1}X_{i1} \\ X_{i1}X_{i2} \\ \vdots \\ X_{i1}X_{ip} \\ X_{i2}X_{i1} \\ X_{i2}X_{i2} \\ \vdots \\ X_{i2}X_{ip} \\ \vdots \\ \vdots \\ X_{ip}X_{i1} \\ X_{ip}X_{i2} \\ \vdots \\ X_{ip}X_{ip} \\ \sqrt{2c}X_{i1} \\ \sqrt{2c}X_{i2} \\ \vdots \\ \sqrt{2c}X_{ip} \\ c \end{bmatrix}$$

Therefore, we may directly focus on $K(X_i, X_j)$ without considering the computation of $\phi(X_i)$.

Gaussian kernel

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

The Gaussian kernel measures the similarity between X_i and X_j.

- If X_i is close to X_j, then $K(X_i, X_j) \to 1$.
- If X_i is far away from X_j, then $K(X_i, X_j) \to 0$.

How to examine whether a function is valid kernel? *I.e.* how to check whether or not there exists a certain feature $\phi(X_i)$ to ensure $K(X_i, X_j) = \phi(X_i)^\top \phi(X_j)$?

Theorem (Mercer)
Let $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be given. Then, the necessary and sufficient requirement for K to be a valid (Mercer) kernel is that

- The kernel matrix is symmetric.

$$K(X_i, X_j) = \phi(X_i)^\top \phi(X_j) = \phi(X_j)^\top \phi(X_i) = K(X_j, X_i)$$

- The kernel matrix is positive semi-definite.

  We define the **Kernel matrix** K as a $n \times n$ matrix, where $K_{ij} = K(X_i, X_j)$. For arbitrary vector z, we get

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

1.4 Common kernels

- **RBF kernel:**

$$K(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right), \qquad \text{where } \sigma \text{ is the hyper-parameter.}$$

- **Simple polynomial kernel:**

$$K(X_i, X_j) = (X_i^\top X_j)^d, \qquad \text{where } d \text{ is the hyper-parameter.}$$

- **Cosine similarity kernel:**

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

- **Sigmoid kernel:**

$$K(X_i, X_j) = \tanh(\alpha X_i^\top X_j + c)$$

where

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

α and c are hyper-parameters.

Use the cross validation to choose the best hyper-parameter (enumerating hyper-parameters).

1.5 With outliers

In this subsection, we will discuss the case when the data cannot be well separated. We formulate the objective as follows.

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

This is equivalent to

$$\min_{\xi,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \underbrace{\max(0, 1 - y_i(w^\top X_i + b))}_{\text{Hinge loss}},$$

where $\max(0, 1 - y_i(w^\top X_i + b))$ is usually called the hinge loss. The hinge loss does not only penalize the negative margins $y_i X_i^\top \beta$, it also penalizes margins less than 1.

Thus, the Lagrangian is given as

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

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

Since $\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial w} = 0$ and $\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial b} = 0$, we can obtain the following dual form of the problem to optimize α.

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

the KKT conditions are

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad y_i(w^\top X_i + b) > 1 \\ \alpha_i = C \quad &\Rightarrow \quad y_i(w^\top X_i + b) < 1 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad y_i(w^\top X_i + b) = 1 \end{aligned}$$

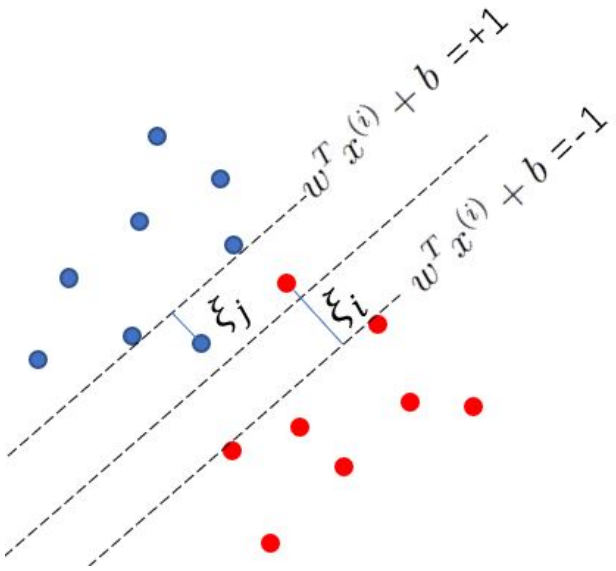

Figure 5: ξ_i

1.6 Examples

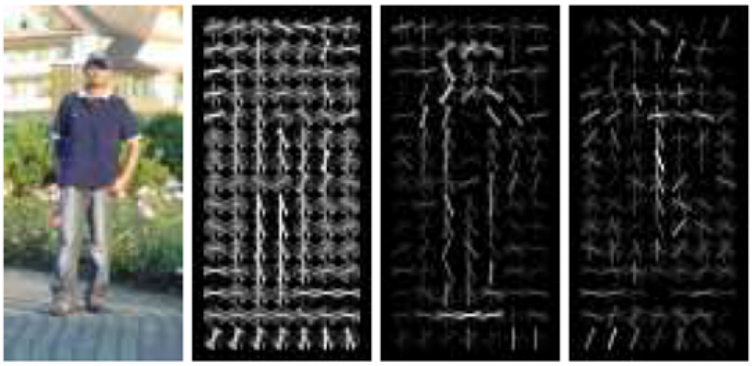

Figure 6: Linear SVM classification based on HOG features. *Dalal et al., Histograms of Oriented Gradients for Human Detection, in CVPR 2015*

- Positive weight dimensions correspond to common patterns among positive samples (persons).
- Negative weight dimensions correspond to common patterns among negative samples (background).
- Small weight dimensions ($|w_i| \rightarrow 0$) correspond to feature dimensions that are unrelated to the task.