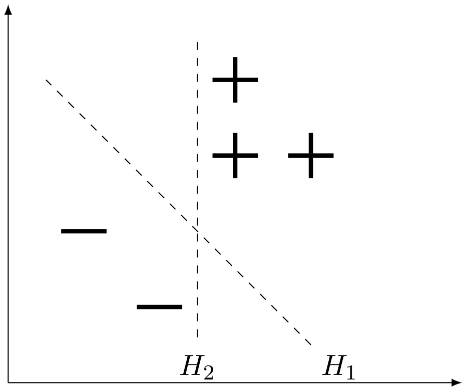
SUPPORT VECTOR MACHINES

Support Vector Machines (SVMs) are one of the most powerful supervised learning algorithms which work on the idea of finding the ‘safest’ separator hyperplane (one with the largest ‘margin’) to divide the labelled hyperspace.

1 Optimal Linear Separator

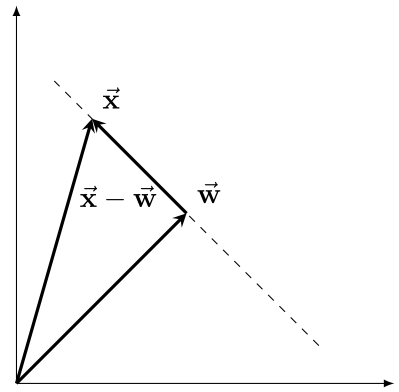
Given training data of the form $(x_1, y_1),\ (x_2, y_2),\ ...\ ,(x_n, y_n)$ where $y_i \in \{+1, -1\}\ \forall\ x_i \in \mathbf{X}$, what is the optimal linear separator that can separate the positives from the negatives?



It is clear that H_1 is a better linear separator as opposed to H_2. It is expected that H_1 will make fewer errors for predicting unseen points as comapred to H_2.

2 Derivation

Let $\vec{\mathbf{w}}$ be a vector perpendicular to a linear separator. The following figure visualizes the constraint that the separator is subjected to.



$$
\begin{aligned}
(\vec{\mathbf{x}} - \vec{\mathbf{w}}).\vec{\mathbf{w}} &= 0 \\
\vec{\mathbf{x}}.\vec{\mathbf{w}} - |\vec{\mathbf{w}}|^2 &= 0 \\
\vec{\mathbf{x}}.\vec{\mathbf{w}} - b &= 0
\end{aligned}
$$

Assume two hyperplanes parallel to the separator hyperplane, one on each side of the separator, such that they cut through the closest training point(s) on the corresponding side. The equations for these 'support' hyperplanes are,

$$
\begin{aligned}
\vec{\mathbf{x}_+}.\vec{\mathbf{w}} &= b + \delta \\
\vec{\mathbf{x}_-}.\vec{\mathbf{w}} &= b - \delta
\end{aligned}
$$

Note that the subscipts $+$ and $-$ denote the positive and negative side respectively. The equations are still satisfied if we scale $\vec{\mathbf{w}}$ and (b and δ) by the same amount. Hence, $\delta = 1$ is set to avoid this difficutly.

$$
\begin{aligned}
\vec{\mathbf{x}_+}.\vec{\mathbf{w}} &= b + 1 \\
\vec{\mathbf{x}_-}.\vec{\mathbf{w}} &= b - 1
\end{aligned}
$$

The length of margin between the $+$ support hyperplane and the $-$ support hyperplane is calculated as follows,

$$\begin{aligned} margin &= (\vec{\mathbf{x_+}} - \vec{\mathbf{x_-}}).\frac{\vec{\mathbf{w}}}{|\vec{\mathbf{w}}|} \\ &= \frac{2}{|\vec{\mathbf{w}}|} \end{aligned}$$

All positive and negative points can be assumed to follow,

$$\begin{aligned} \vec{\mathbf{x_+}}.\vec{\mathbf{w}} \geq b + 1 \\ \vec{\mathbf{x_-}}.\vec{\mathbf{w}} \leq b - 1 \end{aligned}$$

These equations can be combined together to form,

$$y \times (\vec{\mathbf{x}}.\vec{\mathbf{w}} - b) - 1 \geq 0$$

or,

$$y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 \geq 0 \; \forall \; i$$

The problem can be formulated as follows,

$$\text{maximize } \frac{2}{|\vec{\mathbf{w}}|} \text{ subject to } y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 \geq 0 \; \forall \; i$$

or,

$$\text{minimize } \frac{1}{2} \times |\vec{\mathbf{w}}|^2 \text{ subject to } y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 \geq 0 \; \forall \; i$$

This form is called the primal form and it is a quadratic optimization problem but not readily kernelized. The primal form is converted the dual form by using a Langrangian.

$$\mathscr{L}(\vec{\mathbf{w}},\ b,\ \boldsymbol{\alpha}) = \frac{1}{2} \times |\vec{\mathbf{w}}|^2 - \sum_{i=1}^{N} \alpha_i \times [y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1]$$

Taking derivative w.r.t $\vec{\mathbf{w}}$ and b, equating to zero and solving, these two equations arise.

$$\vec{\mathbf{w}}^* = \sum_{i=1}^{N} \alpha_i \times y_i \times \vec{\mathbf{x_i}} \tag{1}$$

$$\sum_{i=1}^{N} \alpha_i \times y_i = 0 \tag{2}$$

Now, for a point i on a support hyperplane,

$$\begin{aligned} y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}}^* - b^*) &= 1 \\ y_i \times y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}}^* - b^*) &= y_i \\ \vec{\mathbf{x_i}}.\vec{\mathbf{w}}^* - b^* &= y_i \\ b^* &= y_i - \vec{\mathbf{x_i}}.\vec{\mathbf{w}}^* \end{aligned}$$

Inserting (1) and (2) back into langragian, the dual form takes the form of,

$$\begin{aligned} \text{maximize } \mathscr{L}(\vec{\mathbf{w}},\ b,\ \boldsymbol{\alpha}) = & \sum_{i=1}^{N} \alpha_i \\ & - \frac{1}{2} \times \sum_{i,j} \alpha_i \times \alpha_j \times y_i \times y_j \times \vec{\mathbf{x_i}}.\vec{\mathbf{x_j}} \\ \text{subject to } \sum_{i=1}^{N} \alpha_i \times y_i = 0 \quad & \text{and} \\ \alpha_i >= 0\ \forall\ i & \end{aligned}$$

The dual formation is also a quadratic optimization problem but is readily kernelized through the substitution $\vec{\mathbf{x_i}}.\vec{\mathbf{x_j}} \rightarrow k(\mathbf{x_i}, \mathbf{x_j})$.

The primal problem is convex and the dual formation is concave and due to duality theroy the duality gap is zero meaning the solutions for both (minima for primal and maxima for dual) the problems are the same.

An interesting equation which holds at the solution is,

$$\alpha_i \times [y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1] = 0\ \forall\ i$$

At the points not on the support hyperplanes, $y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 > 0$ or $y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 < 0$ and hence α_i must be zero for each such point. On the other hand, for the points on the support hyperplanes, $y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 = 0$ so $\alpha_i \geq 0$ for each such point.

The training vectors on the support hyperplanes for which $\alpha_i > 0$ are called support vectors.

3 Non Linearly Separable Case

The equations for the non separable case are,

$$\begin{aligned} \vec{\mathbf{x_i}}.\vec{\mathbf{w}} &\geq b + 1 - \epsilon_i \ \forall \ y_i = +1 \\ \vec{\mathbf{x_i}}.\vec{\mathbf{w}} &\leq b - 1 + \epsilon_i \ \forall \ y_i = -1 \\ \epsilon_i &\geq 0 \ \forall \ i \end{aligned}$$

The variables ϵ_is allow for violation of the constraint. Hence, we must penalize for that. One penalty function is $C \times \sum\limits_i \epsilon_i$ where C is the tradeoff between margin and penalty. To be on the wrong side of the hyperplane ϵ_i will be greater than one, so $\sum\limits_i \epsilon_i$ is an upper bound on number of violations.

The primal problem thus becomes,

$$\begin{aligned} \text{minimize } & \frac{1}{2} \times |\vec{\mathbf{w}}|^2 + C \times \sum_i \epsilon_i \\ \text{subject to } & y_i \times (\vec{\mathbf{x_i}}.\vec{\mathbf{w}} - b) - 1 + \epsilon_i \geq 0 \ \forall \ i \ \mathit{and} \\ & \epsilon_i \geq 0 \ \forall \ i \end{aligned}$$

The dual formulation is also surprisingly similar,

$$\begin{aligned} \text{maximize } \mathscr{L}(\vec{\mathbf{w}},\ b,\ \boldsymbol{\alpha}) = & \sum_{i=1}^{N} \alpha_i \\ & - \frac{1}{2} \times \sum_{i,j} \alpha_i \times \alpha_j \times y_i \times y_j \times \vec{\mathbf{x_i}}.\vec{\mathbf{x_j}} \\ \text{subject to } \sum_{i=1}^{N} \alpha_i \times y_i = 0 \quad & \text{and} \\ 0 <= \alpha_i <= C \ \forall \ i & \end{aligned}$$

The multipliers α_is are now constrained by C. Note that the dual formation is still readily kernelized.

The Kernel Trick

Kernel functions are functions which let you calculate dot product in higher dimensions given vectors in lower dimensions. Sincel SVMs use only dot products, kernel function effectively solve the problem in higher dimensions without expliclity transforming each point into the higher dimension. This leads SVMs to be very time efficient algorithms even when solving the problem in higher dimensions than the input space.

Say there is a $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ transformation function ϕ such that $\phi(i, j) = (i^2, \sqrt{2} \times i \times j, j^2)$, the kernel function $K(\vec{\mathbf{x}}, \vec{\mathbf{z}}) = (\vec{\mathbf{x}}.\vec{\mathbf{z}})^2$ works because,

$$\begin{aligned}
\phi(\vec{\mathbf{x}}).\phi(\vec{\mathbf{z}}) &= (x_i^2, \sqrt{2} \times x_i \times x_j, x_j^2).(z_i^2, \sqrt{2} \times z_i \times z_j, z_j^2) \\
&= x_i^2 \times z_i^2 + 2 \times x_i \times x_j \times z_i \times z_j + x_j^2 \times z_j^2 \\
&= (x_i \times z_i + x_j \times z_j)^2 \\
&= (\vec{\mathbf{x}}.\vec{\mathbf{z}})^2 \\
&= K(\vec{\mathbf{x}}, \vec{\mathbf{z}})
\end{aligned}$$

4 Comments

- SVMs are helpful in tasks like text classification, image classification, digit recognition etc.
- SVMs can work with limited amount of training data and high dimensional data effectively.
- The maximal margin approach reduces overfitting allowing SVMs to generalize better.
- SVMs are typically quite robust against noise.