AIML - CS 337

Lecture 12: Kernels in SVM

26 Sept, 2022

Lecturer: Abir De *Scribe: Moiz, Poojan, Aanya, Makam*

In this lecture we were introduced to "kernel method". A method of applying SVMs to problems with non linear classification boundaries. The simple solution is to transform the input space to a high dimensional feature space and apply SVM in transformed space. Here we are introduced to kernels which are basically measure of similarity between two vectors.

1 Introduction

Remember, we had a question in midsem (question 2.d), which asked to provide a 1D transformation $\phi : \mathbb{R} \rightarrow \mathbb{R}$ so that SVM applied to the given dataset.

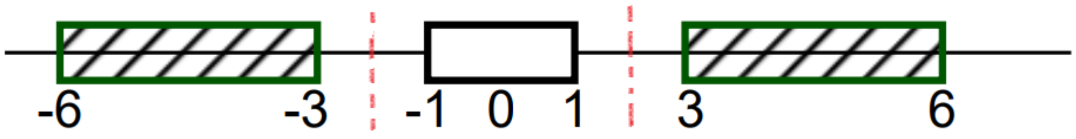

Figure 1: Input space : x

As it can be seen that SVM can not be applied to the initial input space but if we replace $x \rightarrow x^2$ i.e. our prediction will be $h(\phi(x))$ instead of $h(x)$, where $\phi(x) = x^2$ and h is the learnt classifier, it is possible to apply SVM in the transformed space as seen in the images. In this lecture we discover this paradigm in detail.

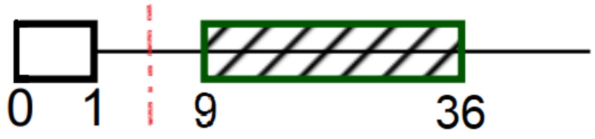

Figure 2: Transformed space : x^2

2 Transformations and Kernels

2.1 Non Linear Transformations

We have seen how to apply SVM paradigm to learning linear boundaries for classification. But what if the decision boundaries are not linear, a simple approach is to convert the given data into high-dimensional feature space. i.e. ,

$$\mathbf{x} \in \mathbb{R}^d \rightarrow \phi(\mathbf{x}) \in \mathbb{R}^{d'} \tag{1}$$

Now, we can train our model on $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle$, $\mathbf{w} \in \mathbb{R}^{d'}$.
Note : It is possible to train for above w if d' is small. If d' is infinite, then we cannot use usual training methods as they would take infinite amount of time. Thus instead of training for w we will try to compute $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle$ directly.

We wish to optimise the give expression:

$$\min_{w} f(\{y_i\}, \{\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle\}) + R(w) \tag{2}$$

where $f : \mathbb{R}^{2*m} \rightarrow \mathbb{R}$ corresponds to the loss function(m is number of samples) and $R : \mathbb{R}^d \rightarrow \mathbb{R}$ corresponds to the regularizer term, typically $\lambda \langle \mathbf{w}, \mathbf{w} \rangle$.
Finally our prediction would be:

$$\hat{y} = g(\langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle) \tag{3}$$

where $\mathbf{w}^*$ is a solution to the above optimization problem.

Now,

Claim 2.1. $\mathbf{w}^*$ *lies in the space spanned by* $\{\phi(\mathbf{x}_i)\}$*, equivalently* $\mathbf{w}^*$ *can be represented a linear combination* $\{\phi(\mathbf{x}_i)\}$.

Proof. Let $\mathbf{w}^*$ be an optimal solution of equation 2. We can rewrite $\mathbf{w}^*$ as:

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i) + \mathbf{u}$$

where $\langle \mathbf{u}, \phi(\mathbf{x}_i) \rangle = 0$ for all i.

Let $\mathbf{w}_1 = \mathbf{w}^* - \mathbf{u}$.

$$\begin{aligned}
R(\mathbf{w}^*) &= R(\mathbf{u} + \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i)) \\
&= \langle \mathbf{u} + \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i), \mathbf{u} + \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i) \rangle && \text{(taking } R(w) = \langle \mathbf{w}, \mathbf{w} \rangle\text{))} \\
&= R(\mathbf{u}) + R(\sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i)) + 2 * \langle \mathbf{u}, \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i) \rangle \\
&= R(\mathbf{u}) + R(\sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i)) \\
&\geq R(\sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i)) = R(\mathbf{w}_1)
\end{aligned}$$

Also,

$$\begin{aligned}
\langle \mathbf{w}^*, \phi(\mathbf{x}_i) \rangle &= \langle \mathbf{u} + \sum_{j=1}^{m} \alpha_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \\
&= \langle \mathbf{u}, \phi(\mathbf{x}_i) \rangle + \langle \sum_{j=1}^{m} \alpha_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \\
&= \langle \sum_{j=1}^{m} \alpha_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \\
&= \langle \mathbf{w}_1, \phi(\mathbf{x}_i) \rangle
\end{aligned}$$

that implies,

$$f(\{y_i\}, \{\langle \mathbf{w}^*, \phi(\mathbf{x}_i) \rangle\}) = f(\{y_i\}, \{\langle \mathbf{w}_1, \phi(\mathbf{x}_i) \rangle\})$$

That implies $L(\mathbf{w}_1) \leq L(\mathbf{w}^*)$, where $L(w)$ is the loss defined in the optimization problem above(2). Since $\mathbf{w}^*$ realizes the minimum for $L(w)$, $\mathbf{u}$ must be 0 (for equality to hold i.e. $R(\mathbf{w}_1) = R(\mathbf{w}^*)$ Hence $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i)$.

QED □

2.2 Kernels

Thus from previous claim,

$$\langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle = \langle \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^{m} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$$

Therefore,we can train for $\{\alpha_i\}$ instead of $\mathbf{w}^*$ and more importantly we do not need to specify the potenially infinite dimensional transformation ϕ. It suffices to provide a function

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

This is called **Kernel**.

We could have different kernels like: $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$, $K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma^2}}$
There are conditions on what the kernel can be so that there is some ϕ for which $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$.
These conditions are given by mercer's theorem.

Theorem 2.2 (Mercers's Theorem). $K : [a, b] \times [a, b] \to \mathbb{R}$ *is a kernel if and only if :*

1. *Symmetric :* $K(x, y) = K(y, x)$ *for all* $x, y \in [a, b]$*, and*
2. *Positive Semi-definite :* $\sum_{j=1}^{n} \sum_{j=1}^{n} K(x_i, x_j)c_ic_j \geq 0$ *for all finite sequences of points* $\mathbf{x}_1, ..., \mathbf{x}_n$ *of* $[a, b]$ *and all choices of real numbers* $c_1, ..., c_n$

3 Kernel SVM

Let w be a vector lying in the span of $\{\phi(\mathbf{x}_i)\}$. It suffices to do the analysis of optimization problem for such w's, since optimal $\mathbf{w}^*$ follows the mentioned property.

$$\begin{aligned}\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle &= \langle \sum_{j=1}^{m} \alpha_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \\ &= \sum_{j=1}^{m} \alpha_j \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \\ &= \sum_{j=1}^{m} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i)\end{aligned}$$

Also,

$$\begin{aligned}\langle \mathbf{w}, \mathbf{w} \rangle &= \langle \sum_{j=1}^{m} \alpha_j \phi(\mathbf{x}_j), \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i) \rangle \\ &= \sum_{j=1}^{m} \sum_{i=1}^{m} \alpha_j \alpha_i \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \\ &= \sum_{j=1}^{m} \sum_{i=1}^{m} \alpha_j \alpha_i K(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}$$

Hence the optimization problem (2) can be expressed just using kernels without any transformation function. Let $\alpha = [\alpha_1, ...\alpha_m]^T$

$$\min_{\alpha \in \mathbb{R}^m} f(\{y_i\}, \{\sum_{j=1}^{m} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i)\}) + \lambda \sum_{j=1}^{m} \sum_{i=1}^{m} \alpha_j \alpha_i K(\mathbf{x}_i, \mathbf{x}_j) \tag{4}$$

Now we do not need the elements in the feature space (Transformed space) to solve the problem (4). The only thing we need to know is how to calculate inner products in the feature space, or equivalently, the kernel function. In fact, it would suffice to know the value of the $m \times m$ matrix $\mathbf{G}$ s.t. $\mathbf{G}[i][j] = K(\mathbf{x}_i, \mathbf{x}_j)$, known as the *Gram* matrix.
Hence we can rewrite the problem (4), taking the soft-SVM approach, as

$$\min_{\alpha \in \mathbb{R}^m} \frac{1}{m} \sum_{i=1}^{m} max\{0, 1 - y_i(\mathbf{G}\alpha)_i\} + \lambda \alpha^T \mathbf{G} \alpha \tag{5}$$

Now once we train for α we can predict for a new instance $\mathbf{x}$ as

$$\hat{y} = \langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{m} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

Concluding: The advantage of working with kernels rather than directly optimizing $\mathbf{w}$ in the feature space is that in some situations the dimension of the feature space is extremely large while implementing the kernel function is very simple.