Kernel Methods

Stefanos Baros

1 Introduction

1.1 Basic idea

We embed instances into a high dimensional space using a mapping $\psi : \mathcal{X} \to \mathcal{F}$ which we call feature space and perform linear classification in that space wherein this task is easier. Whenever inner products of mapped instances appear $\langle \Psi(x_i), \Psi(x_j) \rangle$ we can use Kernel functions to compute these products efficiently without explicitly computing the inner product of these mapped instances. Embedding instances into a higher dimensional space with greater VC dimension can, in general, create sample complexity and computational issues. Support Vector Machines handle the sample complexity issue by searching for "large-margin" i.e., low-norm, separators. Kernel functions handle the computational efficiency issue by computing inner products of mapped instances efficiently.

1.2 Motivating example

We want to classify elements in $\mathbb{R}$ as follows. All x for which $|x| < 2$ should have label -1 while all x for which $|x| \geq 2$ should have label $+1$. Unfortunately, linear halfspaces in the instance space cannot help us classify these examples as their "expressive power" is restricted. Linear functions in the original instance space are not rich enough to help us separate examples. However, as mentioned above, we can map instances to a higher dimensional space and learn a halfspace in that space. Halfspaces in the new space can be more expressive (maybe nonlinear) with respect to the initial space. Here, we use the mapping:

$$\Psi : \mathbb{R} \to \mathbb{R}^2, \quad \Psi(x) = (x, x^2) \tag{1}$$

The range of Ψ defines the feature space. After applying Ψ to data we can use the halfspace:

$$h(x) = \text{sign}(x^2 - 5), \tag{2}$$

which is linear in the feature space seen as $h(x) = \text{sign}(\langle w, \Psi(x) \rangle - b)$ where $w = (0, 1), \quad b = 5$. With this halfspace, we can classify instances perfectly. Note though that, this halfspace is nonlinear on the original space.

1.3 Embedding Mappings

Goal. Our goal when choosing Ψ should be to make the initial distribution close to being linear in the feature space. This, of course, requires prior knowledge about the task! If we believe positive examples can be distinguished by some ellipse then we can define Ψ to be monomials up to order 2 or use a degree 2 polynomial Kernel. Often, it is convenient to use polynomial mappings. Thus, learning a k degree polynomial over $\mathbb{R}$ can be done by learning a linear mapping $p(x) = \langle x, \Psi(x) \rangle$ in the $(k+1)$-dimensional feature space.

$$p(x) = w_0 + w_1\Psi(x_1) + ... + w_m\Psi(x_m), \quad \Psi(x_i) \in \mathcal{F} \tag{3}$$

Polynomial-based classifiers yield much richer hypothesis classes than halfspaces.

Hilbert spaces. The mappings Ψ have to map instances into a Hilbert space in order for the inner products in the feature spaces to be well-defined. Simply stated, a Hilbert space is a *complete* vector space with an inner product measuring length.

Potential issues. We can enrich class of halfspaces by fist applying Ψ to map instances to a new dimensional space and then learn a halfspace there. However, if Ψ is a high dimensional space:

- we might need more samples to learn a halfspace in the feature space $\mathcal{F}$. SVM can help us deal with this issue via the concept of margin.
- performing calculations in a high dimensional space might be too costly. The Kernel trick here can be leveraged for this purpose.

So far, we talked about the benefits of mapping data to a new space. Interestingly, although calculations in the feature space might be too expensive in many machine learning problems the product of mapped instances $\langle \Psi(x_i), \Psi(x_j) \rangle$ appears and not the mapped instances themselves. This is very important as it enables one to use Kernels functions to compute these quantities efficiently. On the other hand, this would not be the case if the mapped instances $\Psi(x_i)$ appear isolated in these problems.

2 Kernel trick

2.1 Basic idea

Kernels are functions that describe inner products in the feature space. In simple terms, a Kernel is really a function that gives us the inner product of two vectors that lie in the feature space i.e.,

$$K(x, x') = \langle \Psi(x), \Psi(x') \rangle \tag{4}$$

Additionally, it also captures similarity between instances x and $x^{'}$.

2.2 Properties of Kernel functions

As mentioned above, a function is a Kernel function if it represents an inner product between $\Psi(x)$ and $\Psi(x^{'})$ for some mapping Ψ. It turns out that Kernel functions have some very useful properties. These are highlighted in the following lemma.

Lemma 1 ([1]). *A symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implements an inner product in some Hilbert space if and only if it is positive semidefinite namely, for all $x_1, ..., x_m$, the Gram matrix, $G_{i,j} = K(x_i, x_j)$ is a positive semidefinite matrix.*

Many algorithms rely on the values of Kernel functions over pair of domain points. The advantage can be summarized as:

- There is no need to specify points $\Psi(x_i)$ in $\mathcal{F}$ or expressing Ψ. The value of the inner product $\langle \Psi(x_i), \Psi(x_j) \rangle$ can be found efficiently via the Kernel $K(x_i, x_j)$.

We now provide an example to illustrate the benefit of working with Kernels [2]. For all $x, x' \in \mathbb{R}^2$:

$$K(x, x') = (xx' + c)^2 = (x_1 x_1' + x_2 x_2' + c)^2 = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1' x_2' \\ \sqrt{2c}x_1' \\ \sqrt{2c}x_2' \\ c \end{bmatrix}$$

$$= \Psi(x_1, x_2) \cdot \Psi(x_1', x_2') \tag{5}$$

The feature space here is 6-dimensional. Thus, it would be challenging to compute the product of $\Psi(x_1, x_2)$ and $\Psi(x_1', x_2')$ directly however, we can equivalently compute $(x_1 x_1' + x_2 x_2' + c)^2$ which involves inner products of 2-dimensional vectors.

2.3 Examples of Kernel functions

Two examples of Kernel functions are the Polynomial Kernels and the Gaussian Kernel. These are described below.

Polynomial Kernel. For all $x, x' \in \mathbb{R}^n$, all functions of the form:

$$K(x, x') = (1 + \langle x, x' \rangle)^k \tag{6}$$

can be written as:

$$K(x, x') = \langle \Psi(x), \Psi(x') \rangle \tag{7}$$

where k is the degree of the polynomial. It is easy to see here that, computing K, is $O(n)$ as we only need to compute $x_0 x_0', ..., x_n x_n'$. However, the dimension of Ψ is $O(n^k)$. Doing calculations in the feature space is thus expensive.

Gaussian Kernel. Let the original space be $\mathbb{R}$ where for $n \geq 0$ there exists an element of $\Psi(x)_n$ that equals $\frac{1}{\sqrt{n!}} e^{\frac{-x^2}{2}} x^n$. The Kernel function in this case is:

$$K(x, x') = e^{-\|x-x'\|^2} \tag{8}$$

Therefore, although $\Psi(x)$ is infinite dimensional the Gaussian Kernel gives us $\langle \Psi(x), \Psi(x') \rangle$ via a simple calculation. Further, when x and x' are away from each other $K(x, x') \approx 0$ while when they are close $K(x, x') \approx 1$. This example supports our earlier claim that Kernels describe similarity between instances. Gaussian Kernels are also called RBF—Radial Basis Functions.

3 General form of SVM optimization problems

Interestingly, all the SVM optimization problems we have derived so far have the following general form in the feature space:

$$\min_w f(\langle w, \Psi(x_1) \rangle, ..., \langle w, \Psi(x_m) \rangle) + R(\|w\|) \tag{9}$$

To see this consider the following two problems we have already seen.

3.1 Soft-SVM for homogeneous halfspaces

The main optimization problem (homogeneous case) in the feature space can be written as:

$$\min_w \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, \Psi(x_i) \rangle\} \tag{10}$$

Letting $\|w\| = a$ and $a_i = \langle w, \Psi(x_i) \rangle$ we finally end up with:

$$\min_w \lambda a^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i a_i\} \tag{11}$$

which is of course of the form (9) with:

$$R(a) = \lambda a^2 \tag{12}$$

$$f(a_1, ..., a_m) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i a_i\}. \tag{13}$$

3.2 Hard-SVM for nonhomogeneous halfspaces

The main optimization problem (nonhomogeneous case) in the feature space can be written as:

$$\begin{aligned} &\min_{w,b} \lambda \|w\|^2 \\ &\text{s.t} \quad y_i(\langle w, \Psi(x_i) \rangle + b \geq 1. \end{aligned} \tag{14}$$

Letting $\|w\| = a$ and $a_i = \langle w, \Psi(x_i)\rangle$ again allows to arrive at:

$$\min_w \lambda a^2 + f(a_1, ..., a_m), \tag{15}$$

where:

$$f(a_1, ..., a_m) = \begin{cases} 0, & if \quad \exists b \quad \text{s.t} \quad y_i(a_i + b) \geq 1, \forall i, \\ \infty, & \text{otherwise.} \end{cases} \tag{16}$$

This problem also has the general form (9).

3.3 Representer Theorem

In the previous section, we showed that standard SVM problems can take the general form (9). Why is that important though? Well, it turns out that the solution of (9) has a very nice property. This is revealed through the following theorem.

Theorem 1 ([1]). *Assume that Ψ is a mapping from $\mathcal{X}$ to a Hilbert space. Then there exists a vector $a \in \mathbb{R}^m$ such that $w = \sum_{i=1}^m a_i \Psi(x_i)$ is an optimal solution of* (9).

Intuitively, an optimal solution of problems that have the general form (9) lies in the linear subspace defined by $\mathbf{span}\{\Psi(x_1), ..., \Psi(x_m)\}$. It is useful to go over the proof of the *Representer theorem* in order to get more insight why this is the case. We now detail the proof of this theorem which is based on a fundamental theorem from linear algebra that is given in the next section.

Proof. Since w appears in the inner products, it should be in the Hilbert space H where Ψ resides. We now let $w^\star$ be an optimal solution of (9). Then, since $\Psi(x_i) \in H$, $K = \text{span}\{\Psi(x_1), ..., \Psi(x_m)\}$ is a linear subspace of H. Thus, each element of H, including $w^\star$ can be expressed as:

$$w^\star = v + v' \tag{17}$$

where $v \in K$ and $v' \in K^\perp$. Equivalently:

$$w^\star = \sum_{i=1}^m a_i \Psi(x_i) + u, \tag{18}$$

where $\langle \Psi(x_i), u\rangle = 0, \ \forall i$. Thus, it holds:

$$\|w^\star\|^2 = \|w\| + \|u\|^2, \tag{19}$$

which yields $\|w\| \leq \|w^\star\|$. As the function R is nondecreasing we obtain:

$$R(\|w\|) \leq R(\|w^\star\|). \tag{20}$$

Finally, we can easily show that $\langle w^\star, \Psi(x_i)\rangle = \langle w, \Psi(x_i)\rangle$ which enables us to conclude that:

$$f(\langle w^\star, \Psi(x_1)\rangle, ..., \langle w^\star, \Psi(x_m)\rangle) = f(\langle w, \Psi(x_1)\rangle, ..., \langle w, \Psi(x_m)\rangle). \tag{21}$$

Combining (21) and (20) we see that the objective function value with w is less than or equal to the value with $w^\star$. As we assumed that $w^\star$ is an optimal solutions then so is w. □

> ***Intuition.*** Basically, in the above proof we have used the fact that $\text{span}\{\Psi(x_1), ..., \Psi(x_m)\}$ defines a linear subspace of the Hilbert space H. Hence, every element, including the optimal solution $w^\star$, of the Hilbert space can be written as $\sum_{i=1}^m a_i\Psi(x_i) + u$ where $\langle \Psi(x_i), u\rangle = 0$, $\forall i$. From that, we have finally shown that an optimal of (9) lies in the linear subspace $\text{span}\{\Psi(x_1), ..., \Psi(x_m)\}$ i.e., $w = \sum_{i=1}^m a_i\Psi(x_i)$.

One may wonder why is the above property of the solution is useful. Well, knowing that the optimal solution of all these problems can be written as a linear combination of the mapped instances allows us to rewrite all the problems of the form (9) in terms of Kernel functions that can be computed efficiently.

3.4 Orthogonal Decomposition of Hilbert Spaces

We state the main theorem below.

Theorem 2. *Assume that K is a closed subspace of the Hilbert space H. Then every $u \in H$ can uniquely be represented as a sum:*

$$u = v + v', \tag{22}$$

where $v \in K$ and $v' \in K^\perp$. This is written as $H = K \bigoplus K^\perp$.

4 Rewriting SVM Optimization Problems in Terms of Kernel Functions

Previously, we have proved that the solution halfspace of (9) has the form:

$$w = \sum_{i=1}^m a_i\Psi(x_i). \tag{23}$$

Using that, we can write:

$$\langle w, \Psi(x_i)\rangle = \left\langle \sum_{i=1}^m a_i\Psi(x_i), \Psi(x_i) \right\rangle = \sum_{i=1}^m a_j K(x_j, x_i). \tag{24}$$

Similarly we can obtain:

$$\|w\|^2 = \left\langle \sum_{j=1}^{m} a_j \Psi(x_j), \sum_{j=1}^{m} a_j \Psi(x_j) \right\rangle = \sum_{i,j=1}^{m} a_i a_j \left\langle \Psi(x_i), \Psi(x_j) \right\rangle = \sum_{i,j=1}^{m} a_i a_j K(x_i, x_j), \tag{25}$$

With all these, we can finally express (9) in the form:

$$\min_{a \in \mathbb{R}^m} f(\sum_{j=1}^{m} a_j K(x_j, x_1), ..., \sum_{j=1}^{m} a_j K(x_j, x_m)) + R(\sqrt{\sum_{i,j=1}^{m} a_i a_j K(x_i, x_j)}). \tag{26}$$

We now explain why expressing SVM optimization problems in the form (26) is useful.

> ***Remark.*** By observing (26), we can see that why do not need to have access to the elements in the feature space to solve these problems but only be able to compute their inner products which can be done efficiently via Kernel functions at different instances x_i, x_j. Thus, (23) allowed us to state these general problems in a new form that enables one to solve them in a computationally efficient manner.

4.1 Soft-SVM Optimization Problem (Homogeneous Case) in Terms of The Gram Matrix

By noting that G, also known as *Gram* matrix, involves the values of the Kernel function evaluated at different instances specifically, $G_{i,j} = K(x_i, x_j)$, we can express the Soft-SVM optimization problem (homogeneous case) compactly as:

$$\min_{a \in \mathbb{R}^m} \lambda a^T G a + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i (Ga)_i\}. \tag{27}$$

Above, $(Ga)_i$ denotes the i-th element of the vector Ga. The clasiffier $h(x) = \langle w, \Psi(x) \rangle$, can be expressed as a sum of Kernel function evaluations:

$$h(x) = \sum_{j=1}^{m} a_j K(x_j, x). \tag{28}$$

Thus, the prediction for an instance x, will be based only on the values of Kernels on training set instances x_j and new instance x.

5 Implementing Soft-SVM with Kernels

In this section, we will show how we can transform the Stochastic Gradient Descent algorithm for solving Soft-SVM in terms of Kernels. The *traditional*

SGD algorithm in the feature space is given below

Traditional SGD algorithm

- Parameter: T
- Initialize: $\theta^{(1)} = 0$
- For $t = 1, ..., T$ update $w^{(t)} = \frac{1}{\lambda t}\theta^{(t)}$
- Choose i uniformly at random from $[m]$
- If $(y_i \left\langle w^{(t)}, \Psi(x_i) \right\rangle < 1)$
  - Set $\theta_i^{(t+1)} = \theta_i^{(t)} + y_i \Psi(x_i)$
- Else
  - Set $\theta_i^{(t+1)} = \theta_i^{(t)}$
- Output: $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$

To recast this algorithm in terms of Kernels we start from the equation $w^{(t)} = \sum_{i=1}^{m} a_j^{(t)} \Psi(x_j)$. With this, we can reduce the condition $(y_i \left\langle w^{(t)}, \Psi(x_i) \right\rangle < 1)$ to:

$$y_i \sum_{j=1}^{m} a_j K(x_j, x_i) < 1. \tag{29}$$

Further, using $w^{(t)} = \frac{1}{\lambda t}\theta^{(t)}$ and the above expression for $w^{(t)}$ we arrive at:

$$\theta^{(t)} = \sum_{j=1}^{m} (a_j^{(t)} \lambda t) \Psi(x_j). \tag{30}$$

Letting $\beta_j^{(t)} = a_j^{(t)} \lambda t$ allows us to write:

$$\theta^{(t)} = \sum_{j=1}^{m} \beta_j^{(t)} \Psi(x_j). \tag{31}$$

We now focus on the update rule $\theta_i^{(t+1)} = \theta_i^{(t)} + y_i \Psi(x_i)$. This can be expanded using $\theta_i^{(t)}$ as:

$$\theta_i^{(t+1)} = \sum_{j=1}^{m} \beta_j^{(t)} \Psi(x_j) + y_i \Psi(x_i) = \sum_{j=1}^{m} \beta_j^{(t+1)} \Psi(x_j), \tag{32}$$

where:

$$\beta_i^{(t+1)} = \beta_i^{(t)} + y_i, \quad \text{for picked } i, \tag{33}$$

$$\beta_j^{(t+1)} = \beta_j^{(t)}, \quad \forall j \neq i. \tag{34}$$

Similarly, the update rule $\theta_i^{(t+1)} = \theta_i^{(t)}$ leads to:

$$\beta_i^{(t+1)} = \beta_i^{(t)}, \quad \text{for picked } i, \tag{35}$$

$$\beta_j^{(t+1)} = \beta_j^{(t)}, \quad \forall j \neq i. \tag{36}$$

By taking into account all these equivalent expressions, we can rewrite the algorithm provided above as follows:

SGD for Soft-SVM with Kernels

- Parameter: T
- Initialize: $\beta^{(1)} = 0$
- For $t = 1, ..., T$ update $a^{(t)} = \frac{\beta^{(t)}}{\lambda t}$
- Choose i uniformly at random from $[m]$
- For all $j \neq i$ set $\beta_j^{(t+1)} = \beta_j^{(t)}$
- If $y_i \sum_{j=1}^{m} a_j K(x_j, x_i) < 1$
  - Set $\beta_i^{(t+1)} = \beta_i^{(t)} + y_i$
- Else
  - Set $\beta_i^{(t+1)} = \beta_i^{(t)}$
- Output: $\overline{w} = \sum_{j=1}^{m} \overline{a}_j \Psi(x_j)$ where $\overline{a}_j = \frac{1}{T} \sum_{t=1}^{T} a_j(t)$

Conclusions. In these notes, we introduced Kernel functions and some of their useful properties. We also showed how we can solve many machine learning problems by mapping instances into high dimensional spaces (feature spaces) and perform linear classification there where it might be easier.

References

[1] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[2] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.