16 Kernel Methods

In the previous chapter we described the SVM paradigm for learning halfspaces in high dimensional feature spaces. This enables us to enrich the expressive power of halfspaces by first mapping the data into a high dimensional feature space, and then learning a linear predictor in that space. This is similar to the AdaBoost algorithm, which learns a composition of a halfspace over base hypotheses. While this approach greatly extends the expressiveness of halfspace predictors, it raises both sample complexity and computational complexity challenges. In the previous chapter we tackled the sample complexity issue using the concept of margin. In this chapter we tackle the computational complexity challenge using the method of *kernels*.

We start the chapter by describing the idea of embedding the data into a high dimensional feature space. We then introduce the idea of kernels. A kernel is a type of a similarity measure between instances. The special property of kernel similarities is that they can be viewed as inner products in some Hilbert space (or Euclidean space of some high dimension) to which the instance space is virtually embedded. We introduce the "kernel trick" that enables computationally efficient implementation of learning, without explicitly handling the high dimensional representation of the domain instances. Kernel based learning algorithms, and in particular kernel-SVM, are very useful and popular machine learning tools. Their success may be attributed both to being flexible for accommodating domain specific prior knowledge and to having a well developed set of efficient implementation algorithms.

16.1 Embeddings into Feature Spaces

The expressive power of halfspaces is rather restricted – for example, the following training set is not separable by a halfspace.

Let the domain be the real line; consider the domain points $\{-10, -9, -8, \ldots, 0, 1, \ldots, 9, 10\}$ where the labels are $+1$ for all x such that $|x| > 2$ and -1 otherwise.

To make the class of halfspaces more expressive, we can first map the original instance space into another space (possibly of a higher dimension) and then learn a halfspace in that space. For example, consider the example mentioned previously. Instead of learning a halfspace in the original representation let us

first define a mapping $\psi : \mathbb{R} \to \mathbb{R}^2$ as follows:

$$\psi(x) = (x, x^2).$$

We use the term *feature space* to denote the range of ψ. After applying ψ the data can be easily explained using the halfspace $h(x) = \text{sign}(\langle \mathbf{w}, \psi(x) \rangle - b)$, where $\mathbf{w} = (0, 1)$ and $b = 5$.

The basic paradigm is as follows:

1. Given some domain set $\mathcal{X}$ and a learning task, choose a mapping $\psi : \mathcal{X} \to \mathcal{F}$, for some *feature space* $\mathcal{F}$, that will usually be $\mathbb{R}^n$ for some n (however, the range of such a mapping can be any *Hilbert space*, including such spaces of infinite dimension, as we will show later).
2. Given a sequence of labeled examples, $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, create the image sequence $\hat{S} = (\psi(\mathbf{x}_1), y_1), \ldots, (\psi(\mathbf{x}_m), y_m)$.
3. Train a linear predictor h over $\hat{S}$.
4. Predict the label of a test point, $\mathbf{x}$, to be $h(\psi(\mathbf{x}))$.

Note that, for every probability distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, we can readily define its image probability distribution $\mathcal{D}^\psi$ over $\mathcal{F} \times \mathcal{Y}$ by setting, for every subset $A \subseteq \mathcal{F} \times \mathcal{Y}$, $\mathcal{D}^\psi(A) = \mathcal{D}(\psi^{-1}(A))$.[1] It follows that for every predictor h over the feature space, $L_{\mathcal{D}^\psi}(h) = L_{\mathcal{D}}(h \circ \psi)$, where $h \circ \psi$ is the composition of h onto ψ.

The success of this learning paradigm depends on choosing a good ψ for a given learning task: that is, a ψ that will make the image of the data distribution (close to being) linearly separable in the feature space, thus making the resulting algorithm a good learner for a given task. Picking such an embedding requires prior knowledge about that task. However, often some generic mappings that enable us to enrich the class of halfspaces and extend its expressiveness are used. One notable example is polynomial mappings, which are a generalization of the ψ we have seen in the previous example.

Recall that the prediction of a standard halfspace classifier on an instance $\mathbf{x}$ is based on the linear mapping $\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle$. We can generalize linear mappings to a polynomial mapping, $\mathbf{x} \mapsto p(\mathbf{x})$, where p is a multivariate polynomial of degree k. For simplicity, consider first the case in which $\mathbf{x}$ is 1 dimensional. In that case, $p(x) = \sum_{j=0}^{k} w_j x^j$, where $\mathbf{w} \in \mathbb{R}^{k+1}$ is the vector of coefficients of the polynomial we need to learn. We can rewrite $p(x) = \langle \mathbf{w}, \psi(x) \rangle$ where $\psi : \mathbb{R} \to \mathbb{R}^{k+1}$ is the mapping $x \mapsto (1,\ x,\ x^2,\ x^3, \ldots, x^k)$. It follows that learning a k degree polynomial over $\mathbb{R}$ can be done by learning a linear mapping in the $(k+1)$ dimensional feature space.

More generally, a degree k multivariate polynomial from $\mathbb{R}^n$ to $\mathbb{R}$ can be written as

$$p(\mathbf{x}) = \sum_{J \in [n]^r : r \leq k} w_J \prod_{i=1}^{r} x_{J_i}. \tag{16.1}$$

[1] This is defined for every A such that $\psi^{-1}(A)$ is measurable with respect to $\mathcal{D}$.

As before, we can rewrite $p(\mathbf{x}) = \langle \mathbf{w}, \psi(\mathbf{x}) \rangle$ where now $\psi : \mathbb{R}^n \to \mathbb{R}^d$ is such that for every $J \in [n]^r$, $r \leq k$, the coordinate of $\psi(\mathbf{x})$ associated with J is the monomial $\prod_{i=1}^{r} x_{J_i}$.

Naturally, polynomial-based classifiers yield much richer hypothesis classes than halfspaces. We have seen at the beginning of this chapter an example in which the training set, in its original domain ($\mathcal{X} = \mathbb{R}$), cannot be separable by a halfspace, but after the embedding $x \mapsto (x,\ x^2)$ it is perfectly separable. So, while the classifier is always linear in the feature space, it can have highly nonlinear behavior on the original space from which instances were sampled.

In general, we can choose any feature mapping ψ that maps the original instances into some *Hilbert space*.[2] The Euclidean space $\mathbb{R}^d$ is a Hilbert space for any finite d. But there are also infinite dimensional Hilbert spaces (as we shall see later on in this chapter).

The bottom line of this discussion is that we can enrich the class of halfspaces by first applying a nonlinear mapping, ψ, that maps the instance space into some feature space, and then learning a halfspace in that feature space. However, if the range of ψ is a high dimensional space we face two problems. First, the VC-dimension of halfspaces in $\mathbb{R}^n$ is $n + 1$, and therefore, if the range of ψ is very large, we need many more samples in order to learn a halfspace in the range of ψ. Second, from the computational point of view, performing calculations in the high dimensional space might be too costly. In fact, even the representation of the vector $\mathbf{w}$ in the feature space can be unrealistic. The first issue can be tackled using the paradigm of large margin (or low norm predictors), as we already discussed in the previous chapter in the context of the SVM algorithm. In the following section we address the computational issue.

16.2 The Kernel Trick

We have seen that embedding the input space into some high dimensional feature space makes halfspace learning more expressive. However, the computational complexity of such learning may still pose a serious hurdle – computing linear separators over very high dimensional data may be computationally expensive. The common solution to this concern is kernel based learning. The term "kernels" is used in this context to describe inner products in the feature space. Given an embedding ψ of some domain space $\mathcal{X}$ into some Hilbert space, we define the kernel function $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$. One can think of K as specifying similarity between instances and of the embedding ψ as mapping the domain set

[2] A Hilbert space is a vector space with an inner product, which is also complete. A space is complete if all Cauchy sequences in the space converge.
In our case, the norm $\|\mathbf{w}\|$ is defined by the inner product $\sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}$. The reason we require the range of ψ to be in a Hilbert space is that projections in a Hilbert space are well defined. In particular, if M is a linear subspace of a Hilbert space, then every $\mathbf{x}$ in the Hilbert space can be written as a sum $\mathbf{x} = \mathbf{u} + \mathbf{v}$ where $\mathbf{u} \in M$ and $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ for all $\mathbf{w} \in M$. We use this fact in the proof of the representer theorem given in the next section.

$\mathcal{X}$ into a space where these similarities are realized as inner products. It turns out that many learning algorithms for halfspaces can be carried out just on the basis of the values of the kernel function over pairs of domain points. The main advantage of such algorithms is that they implement linear separators in high dimensional feature spaces without having to specify points in that space or expressing the embedding ψ explicitly. The remainder of this section is devoted to constructing such algorithms.

In the previous chapter we saw that regularizing the norm of $\mathbf{w}$ yields a small sample complexity even if the dimensionality of the feature space is high. Interestingly, as we show later, regularizing the norm of $\mathbf{w}$ is also helpful in overcoming the computational problem. To do so, first note that all versions of the SVM optimization problem we have derived in the previous chapter are instances of the following general problem:

$$\min_{\mathbf{w}} \left(f\left(\langle \mathbf{w}, \psi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \psi(\mathbf{x}_m) \rangle \right) + R(\|\mathbf{w}\|) \right), \tag{16.2}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is an arbitrary function and $R : \mathbb{R}_+ \to \mathbb{R}$ is a monotonically nondecreasing function. For example, Soft-SVM for homogenous halfspaces (Equation (15.6)) can be derived from Equation (16.2) by letting $R(a) = \lambda a^2$ and $f(a_1, \ldots, a_m) = \frac{1}{m} \sum_i \max\{0, 1 - y_i a_i\}$. Similarly, Hard-SVM for nonhomogenous halfspaces (Equation (15.2)) can be derived from Equation (16.2) by letting $R(a) = a^2$ and letting $f(a_1, \ldots, a_m)$ be 0 if there exists b such that $y_i(a_i + b) \geq 1$ for all i, and $f(a_1, \ldots, a_m) = \infty$ otherwise.

The following theorem shows that there exists an optimal solution of Equation (16.2) that lies in the span of $\{\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_m)\}$.

THEOREM 16.1 (Representer Theorem) *Assume that ψ is a mapping from $\mathcal{X}$ to a Hilbert space. Then, there exists a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ such that $\mathbf{w} = \sum_{i=1}^m \alpha_i \psi(\mathbf{x}_i)$ is an optimal solution of Equation (16.2).*

Proof Let $\mathbf{w}^\star$ be an optimal solution of Equation (16.2). Because $\mathbf{w}^\star$ is an element of a Hilbert space, we can rewrite $\mathbf{w}^\star$ as

$$\mathbf{w}^\star = \sum_{i=1}^m \alpha_i \psi(\mathbf{x}_i) + \mathbf{u},$$

where $\langle \mathbf{u}, \psi(\mathbf{x}_i) \rangle = 0$ for all i. Set $\mathbf{w} = \mathbf{w}^\star - \mathbf{u}$. Clearly, $\|\mathbf{w}^\star\|^2 = \|\mathbf{w}\|^2 + \|\mathbf{u}\|^2$, thus $\|\mathbf{w}\| \leq \|\mathbf{w}^\star\|$. Since R is nondecreasing we obtain that $R(\|\mathbf{w}\|) \leq R(\|\mathbf{w}^\star\|)$. Additionally, for all i we have that

$$\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle = \langle \mathbf{w}^\star - \mathbf{u}, \psi(\mathbf{x}_i) \rangle = \langle \mathbf{w}^\star, \psi(\mathbf{x}_i) \rangle,$$

hence

$$f\left(\langle \mathbf{w}, \psi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \psi(\mathbf{x}_m) \rangle \right) = f\left(\langle \mathbf{w}^\star, \psi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}^\star, \psi(\mathbf{x}_m) \rangle \right).$$

We have shown that the objective of Equation (16.2) at $\mathbf{w}$ cannot be larger than the objective at $\mathbf{w}^\star$ and therefore $\mathbf{w}$ is also an optimal solution. Since $\mathbf{w} = \sum_{i=1}^m \alpha_i \psi(\mathbf{x}_i)$ we conclude our proof. □

On the basis of the representer theorem we can optimize Equation (16.2) with respect to the coefficients $\boldsymbol{\alpha}$ instead of the coefficients $\mathbf{w}$ as follows. Writing $\mathbf{w} = \sum_{j=1}^{m} \alpha_j \psi(\mathbf{x}_j)$ we have that for all i

$$\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle = \left\langle \sum_j \alpha_j \psi(\mathbf{x}_j), \psi(\mathbf{x}_i) \right\rangle = \sum_{j=1}^{m} \alpha_j \langle \psi(\mathbf{x}_j), \psi(\mathbf{x}_i) \rangle.$$

Similarly,

$$\|\mathbf{w}\|^2 = \left\langle \sum_j \alpha_j \psi(\mathbf{x}_j), \sum_j \alpha_j \psi(\mathbf{x}_j) \right\rangle = \sum_{i,j=1}^{m} \alpha_i \alpha_j \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle.$$

Let $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ be a function that implements the kernel function with respect to the embedding ψ. Instead of solving Equation (16.2) we can solve the equivalent problem

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} f\left(\sum_{j=1}^{m} \alpha_j K(\mathbf{x}_j, \mathbf{x}_1), \ldots, \sum_{j=1}^{m} \alpha_j K(\mathbf{x}_j, \mathbf{x}_m) \right) + R\left(\sqrt{\sum_{i,j=1}^{m} \alpha_i \alpha_j K(\mathbf{x}_j, \mathbf{x}_i)} \right). \tag{16.3}$$

To solve the optimization problem given in Equation (16.3), we do not need any direct access to elements in the feature space. The only thing we should know is how to calculate inner products in the feature space, or equivalently, to calculate the kernel function. In fact, to solve Equation (16.3) we solely need to know the value of the $m \times m$ matrix G s.t. $G_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, which is often called the *Gram* matrix.

In particular, specifying the preceding to the Soft-SVM problem given in Equation (15.6), we can rewrite the problem as

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \left(\lambda \boldsymbol{\alpha}^T G \boldsymbol{\alpha} + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y_i (G\boldsymbol{\alpha})_i\right\} \right), \tag{16.4}$$

where $(G\boldsymbol{\alpha})_i$ is the i'th element of the vector obtained by multiplying the Gram matrix G by the vector $\boldsymbol{\alpha}$. Note that Equation (16.4) can be written as quadratic programming and hence can be solved efficiently. In the next section we describe an even simpler algorithm for solving Soft-SVM with kernels.

Once we learn the coefficients $\boldsymbol{\alpha}$ we can calculate the prediction on a new instance by

$$\langle \mathbf{w}, \psi(\mathbf{x}) \rangle = \sum_{j=1}^{m} \alpha_j \langle \psi(\mathbf{x}_j), \psi(\mathbf{x}) \rangle = \sum_{j=1}^{m} \alpha_j K(\mathbf{x}_j, \mathbf{x}).$$

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. A few examples are given in the following.

Example 16.1 (Polynomial Kernels) The k degree polynomial kernel is defined to be

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

Now we will show that this is indeed a kernel function. That is, we will show that there exists a mapping ψ from the original space to some higher dimensional space for which $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$. For simplicity, denote $x_0 = x'_0 = 1$. Then, we have

$$\begin{aligned}
K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k &= (1 + \langle \mathbf{x}, \mathbf{x}' \rangle) \cdot \cdots \cdot (1 + \langle \mathbf{x}, \mathbf{x}' \rangle) \\
&= \left(\sum_{j=0}^{n} x_j x'_j \right) \cdot \cdots \cdot \left(\sum_{j=0}^{n} x_j x'_j \right) \\
&= \sum_{J \in \{0,1,\ldots,n\}^k} \prod_{i=1}^{k} x_{J_i} x'_{J_i} \\
&= \sum_{J \in \{0,1,\ldots,n\}^k} \prod_{i=1}^{k} x_{J_i} \prod_{i=1}^{k} x'_{J_i}.
\end{aligned}$$

Now, if we define $\psi : \mathbb{R}^n \to \mathbb{R}^{(n+1)^k}$ such that for $J \in \{0,\ 1, \ldots, n\}^k$ there is an element of $\psi(\mathbf{x})$ that equals $\prod_{i=1}^{k} x_{J_i}$, we obtain that

$$K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle.$$

Since ψ contains all the monomials up to degree k, a halfspace over the range of ψ corresponds to a polynomial predictor of degree k over the original space. Hence, learning a halfspace with a k degree polynomial kernel enables us to learn polynomial predictors of degree k over the original space.

Note that here the complexity of implementing K is $O(n)$ while the dimension of the feature space is on the order of n^k.

Example 16.2 (Gaussian Kernel) Let the original instance space be $\mathbb{R}$ and consider the mapping ψ where for each nonnegative integer $n \geq 0$ there exists an element $\psi(x)_n$ that equals $\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} x^n$. Then,

$$\begin{aligned}
\langle \psi(x),\ \psi(x') \rangle &= \sum_{n=0}^{\infty} \left(\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} x^n \right) \left(\frac{1}{\sqrt{n!}} e^{-\frac{(x')^2}{2}} (x')^n \right) \\
&= e^{-\frac{x^2 + (x')^2}{2}} \sum_{n=0}^{\infty} \left(\frac{(xx')^n}{n!} \right) \\
&= e^{-\frac{\|x - x'\|^2}{2}}.
\end{aligned}$$

Here the feature space is of infinite dimension while evaluating the kernel is very

simple. More generally, given a scalar $\sigma > 0$, the Gaussian kernel is defined to be

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma}}.$$

Intuitively, the Gaussian kernel sets the inner product in the feature space between $\mathbf{x}, \mathbf{x}'$ to be close to zero if the instances are far away from each other (in the original domain) and close to 1 if they are close. σ is a parameter that controls the scale determining what we mean by "close." It is easy to verify that K implements an inner product in a space in which for any n and any monomial of order k there exists an element of $\psi(\mathbf{x})$ that equals $\frac{1}{\sqrt{n!}} e^{-\frac{\|\mathbf{x}\|^2}{2}} \prod_{i=1}^{n} x_{J_i}$. Hence, we can learn any polynomial predictor over the original space by using a Gaussian kernel.

Recall that the VC-dimension of the class of all polynomial predictors is infinite (see Exercise 12). There is no contradiction, because the sample complexity required to learn with Gaussian kernels depends on the margin in the feature space, which will be large if we are lucky, but can in general be arbitrarily small.

The Gaussian kernel is also called the RBF kernel, for "Radial Basis Functions."

16.2.1 Kernels as a Way to Express Prior Knowledge

As we discussed previously, a feature mapping, ψ, may be viewed as expanding the class of linear classifiers to a richer class (corresponding to linear classifiers over the feature space). However, as discussed in the book so far, the suitability of any hypothesis class to a given learning task depends on the nature of that task. One can therefore think of an embedding ψ as a way to express and utilize prior knowledge about the problem at hand. For example, if we believe that positive examples can be distinguished by some ellipse, we can define ψ to be all the monomials up to order 2, or use a degree 2 polynomial kernel.

As a more realistic example, consider the task of learning to find a sequence of characters ("signature") in a file that indicates whether it contains a virus or not. Formally, let $\mathcal{X}_d$ be the set of all strings of length at most d over some alphabet set Σ. The hypothesis class that one wishes to learn is $\mathcal{H} = \{h_v : v \in \mathcal{X}_d\}$, where, for a string $x \in \mathcal{X}_d$, $h_v(x)$ is 1 iff v is a substring of x (and $h_v(x) = -1$ otherwise). Let us show how using an appropriate embedding this class can be realized by linear classifiers over the resulting feature space. Consider a mapping ψ to a space $\mathbb{R}^s$ where $s = |\mathcal{X}_d|$, so that each coordinate of $\psi(x)$ corresponds to some string v and indicates whether v is a substring of x (that is, for every $x \in \mathcal{X}_d$, $\psi(x)$ is a vector in $\{0,1\}^{|\mathcal{X}_d|}$). Note that the dimension of this feature space is exponential in d. It is not hard to see that every member of the class $\mathcal{H}$ can be realized by composing a linear classifier over $\psi(x)$, and, moreover, by such a halfspace whose norm is 1 and that attains a margin of 1 (see Exercise 1). Furthermore, for every $x \in \mathcal{X}$, $\|\psi(x)\| = O(d)$. So, overall, it is learnable using SVM with a sample

complexity that is polynomial in d. However, the dimension of the feature space is exponential in d so a direct implementation of SVM over the feature space is problematic. Luckily, it is easy to calculate the inner product in the feature space (i.e., the kernel function) without explicitly mapping instances into the feature space. Indeed, $K(x, x')$ is simply the number of common substrings of x and x', which can be easily calculated in time polynomial in d.

This example also demonstrates how feature mapping enables us to use halfspaces for nonvectorial domains.

16.2.2 Characterizing Kernel Functions*

As we have discussed in the previous section, we can think of the specification of the kernel matrix as a way to express prior knowledge. Consider a given similarity function of the form $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Is it a valid kernel function? That is, does it represent an inner product between $\psi(\mathbf{x})$ and $\psi(\mathbf{x}')$ for some feature mapping ψ? The following lemma gives a sufficient and necessary condition.

LEMMA 16.2 *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 $\mathbf{x}_1, \ldots, \mathbf{x}_m$, the Gram matrix, $G_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, is a positive semidefinite matrix.*

Proof It is trivial to see that if K implements an inner product in some Hilbert space then the Gram matrix is positive semidefinite. For the other direction, define the space of functions over $\mathcal{X}$ as $\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$. For each $\mathbf{x} \in \mathcal{X}$ let $\psi(\mathbf{x})$ be the function $\mathbf{x} \mapsto K(\cdot, \mathbf{x})$. Define a vector space by taking all linear combinations of elements of the form $K(\cdot, \mathbf{x})$. Define an inner product on this vector space to be

$$\left\langle \sum_i \alpha_i K(\cdot, \mathbf{x}_i), \sum_j \beta_j K(\cdot, \mathbf{x}'_j) \right\rangle = \sum_{i,j} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{x}'_j).$$

This is a valid inner product since it is symmetric (because K is symmetric), it is linear (immediate), and it is positive definite (it is easy to see that $K(\mathbf{x}, \mathbf{x}) \geq 0$ with equality only for $\psi(\mathbf{x})$ being the zero function). Clearly,

$$\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = \langle K(\cdot, \mathbf{x}), K(\cdot, \mathbf{x}') \rangle = K(\mathbf{x}, \mathbf{x}'),$$

which concludes our proof. □

16.3 Implementing Soft-SVM with Kernels

Next, we turn to solving Soft-SVM with kernels. While we could have designed an algorithm for solving Equation (16.4), there is an even simpler approach that

directly tackles the Soft-SVM optimization problem in the feature space,

$$\min_{\mathbf{w}} \left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle\} \right), \tag{16.5}$$

while only using kernel evaluations. The basic observation is that the vector $\mathbf{w}^{(t)}$ maintained by the SGD procedure we have described in Section 15.5 is always in the linear span of $\{\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_m)\}$. Therefore, rather than maintaining $\mathbf{w}^{(t)}$ we can maintain the corresponding coefficients $\boldsymbol{\alpha}$.

Formally, let K be the kernel function, namely, for all $\mathbf{x}, \mathbf{x}'$, $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}')\rangle$. We shall maintain two vectors in $\mathbb{R}^m$, corresponding to two vectors $\boldsymbol{\theta}^{(t)}$ and $\mathbf{w}^{(t)}$ defined in the SGD procedure of Section 15.5. That is, $\boldsymbol{\beta}^{(t)}$ will be a vector such that

$$\boldsymbol{\theta}^{(t)} = \sum_{j=1}^{m} \beta_j^{(t)} \psi(\mathbf{x}_j) \tag{16.6}$$

and $\boldsymbol{\alpha}^{(t)}$ be such that

$$\mathbf{w}^{(t)} = \sum_{j=1}^{m} \alpha_j^{(t)} \psi(\mathbf{x}_j). \tag{16.7}$$

The vectors $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are updated according to the following procedure.

SGD for Solving Soft-SVM with Kernels

Goal: Solve Equation (16.5)
parameter: T
Initialize: $\boldsymbol{\beta}^{(1)} = \mathbf{0}$
for $t = 1, \ldots, T$
  Let $\boldsymbol{\alpha}^{(t)} = \frac{1}{\lambda t}\boldsymbol{\beta}^{(t)}$
  Choose i uniformly at random from $[m]$
  For all $j \neq i$ set $\beta_j^{(t+1)} = \beta_j^{(t)}$
  If $(y_i \sum_{j=1}^{m} \alpha_j^{(t)} K(\mathbf{x}_j, \mathbf{x}_i) < 1)$
    Set $\beta_i^{(t+1)} = \beta_i^{(t)} + y_i$
  Else
    Set $\beta_i^{(t+1)} = \beta_i^{(t)}$
Output: $\bar{\mathbf{w}} = \sum_{j=1}^{m} \bar{\alpha}_j \psi(\mathbf{x}_j)$ where $\bar{\boldsymbol{\alpha}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\alpha}^{(t)}$

The following lemma shows that the preceding implementation is equivalent to running the SGD procedure described in Section 15.5 on the feature space.

LEMMA 16.3 *Let $\hat{\mathbf{w}}$ be the output of the SGD procedure described in Section 15.5, when applied on the feature space, and let $\bar{\mathbf{w}} = \sum_{j=1}^{m} \bar{\alpha}_j \psi(\mathbf{x}_j)$ be the output of applying SGD with kernels. Then $\bar{\mathbf{w}} = \hat{\mathbf{w}}$.*

Proof We will show that for every t Equation (16.6) holds, where $\boldsymbol{\theta}^{(t)}$ is the result of running the SGD procedure described in Section 15.5 in the feature

space. By the definition of $\boldsymbol{\alpha}^{(t)} = \frac{1}{\lambda t}\boldsymbol{\beta}^{(t)}$ and $\mathbf{w}^{(t)} = \frac{1}{\lambda t}\boldsymbol{\theta}^{(t)}$, this claim implies that Equation (16.7) also holds, and the proof of our lemma will follow. To prove that Equation (16.6) holds we use a simple inductive argument. For $t = 1$ the claim trivially holds. Assume it holds for $t \geq 1$. Then,

$$y_i \left\langle \mathbf{w}^{(t)}, \psi(\mathbf{x}_i) \right\rangle = y_i \left\langle \sum_j \alpha_j^{(t)} \psi(\mathbf{x}_j), \psi(\mathbf{x}_i) \right\rangle = y_i \sum_{j=1}^{m} \alpha_j^{(t)} K(\mathbf{x}_j, \mathbf{x}_i).$$

Hence, the condition in the two algorithms is equivalent and if we update $\boldsymbol{\theta}$ we have

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + y_i \psi(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j^{(t)} \psi(\mathbf{x}_j) + y_i \psi(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j^{(t+1)} \psi(\mathbf{x}_j),$$

which concludes our proof. □

16.4 Summary

Mappings from the given domain to some higher dimensional space, on which a halfspace predictor is used, can be highly powerful. We benefit from a rich and complex hypothesis class, yet need to solve the problems of high sample and computational complexities. In Chapter 10, we discussed the AdaBoost algorithm, which faces these challenges by using a weak learner: Even though we're in a very high dimensional space, we have an "oracle" that bestows on us a single good coordinate to work with on each iteration. In this chapter we introduced a different approach, the kernel trick. The idea is that in order to find a halfspace predictor in the high dimensional space, we do not need to know the representation of instances in that space, but rather the values of inner products between the mapped instances. Calculating inner products between instances in the high dimensional space without using their representation in that space is done using kernel functions. We have also shown how the SGD algorithm can be implemented using kernels.

The ideas of feature mapping and the kernel trick allow us to use the framework of halfspaces and linear predictors for nonvectorial data. We demonstrated how kernels can be used to learn predictors over the domain of strings.

We presented the applicability of the kernel trick in SVM. However, the kernel trick can be applied in many other algorithms. A few examples are given as exercises.

This chapter ends the series of chapters on linear predictors and convex problems. The next two chapters deal with completely different types of hypothesis classes.

16.5 Bibliographic Remarks

In the context of SVM, the kernel-trick has been introduced in Boser et al. (1992). See also Aizerman, Braverman & Rozonoer (1964). The observation that the kernel-trick can be applied whenever an algorithm only relies on inner products was first stated by Schölkopf, Smola & Müller (1998). The proof of the representer theorem is given in (Schölkopf, Herbrich, Smola & Williamson 2000, Schölkopf, Herbrich & Smola 2001). The conditions stated in Lemma 16.2 are simplification of conditions due to Mercer. Many useful kernel functions have been introduced in the literature for various applications. We refer the reader to Schölkopf & Smola (2002).

16.6 Exercises

1. Consider the task of finding a sequence of characters in a file, as described in Section 16.2.1. Show that every member of the class $\mathcal{H}$ can be realized by composing a linear classifier over $\psi(x)$, whose norm is 1 and that attains a margin of 1.
2. **Kernelized Perceptron:** Show how to run the Perceptron algorithm while only accessing the instances via the kernel function. *Hint:* The derivation is similar to the derivation of implementing SGD with kernels.
3. **Kernel Ridge Regression:** The ridge regression problem, with a feature mapping ψ, is the problem of finding a vector $\mathbf{w}$ that minimizes the function

$$f(\mathbf{w}) = \lambda \|\mathbf{w}\|^2 + \frac{1}{2m}\sum_{i=1}^{m}(\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle - y_i)^2, \tag{16.8}$$

   and then returning the predictor

$$h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle.$$

   Show how to implement the ridge regression algorithm with kernels.

   Hint: The representer theorem tells us that there exists a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ such that $\sum_{i=1}^m \alpha_i \psi(\mathbf{x}_i)$ is a minimizer of Equation (16.8).

   1. Let G be the Gram matrix with regard to S and K. That is, $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Define $g : \mathbb{R}^m \to \mathbb{R}$ by

$$g(\boldsymbol{\alpha}) = \lambda \cdot \boldsymbol{\alpha}^T G \boldsymbol{\alpha} + \frac{1}{2m}\sum_{i=1}^{m}(\langle \boldsymbol{\alpha}, G_{\cdot,i}\rangle - y_i)^2, \tag{16.9}$$

      where $G_{\cdot,i}$ is the i'th column of G. Show that if $\boldsymbol{\alpha}^*$ minimizes Equation (16.9) then $\mathbf{w}^* = \sum_{i=1}^m \alpha_i^* \psi(\mathbf{x}_i)$ is a minimizer of f.
   2. Find a closed form expression for $\boldsymbol{\alpha}^*$.
4. Let N be any positive integer. For every $x, x' \in \{1, \ldots, N\}$ define

$$K(x, x') = \min\{x, x'\}.$$

Prove that K is a valid kernel; namely, find a mapping $\psi : \{1, \ldots, N\} \to H$ where H is some Hilbert space, such that

$$\forall x, x' \in \{1, \ldots, N\},\ K(x, x') = \langle \psi(x), \psi(x') \rangle.$$

5. A supermarket manager would like to learn which of his customers have babies on the basis of their shopping carts. Specifically, he sampled i.i.d. customers, where for customer i, let $x_i \subset \{1, \ldots, d\}$ denote the subset of items the customer bought, and let $y_i \in \{\pm 1\}$ be the label indicating whether this customer has a baby. As prior knowledge, the manager knows that there are k items such that the label is determined to be 1 iff the customer bought at least one of these k items. Of course, the identity of these k items is not known (otherwise, there was nothing to learn). In addition, according to the store regulation, each customer can buy at most s items. Help the manager to design a learning algorithm such that both its time complexity and its sample complexity are polynomial in s, k, and $1/\epsilon$.
6. Let $\mathcal{X}$ be an instance set and let ψ be a feature mapping of $\mathcal{X}$ into some Hilbert feature space V. Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel function that implements inner products in the feature space V.

   Consider the binary classification algorithm that predicts the label of an unseen instance according to the class with the closest average. Formally, given a training sequence $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, for every $y \in \{\pm 1\}$ we define

$$c_y = \frac{1}{m_y} \sum_{i: y_i = y} \psi(\mathbf{x}_i).$$

   where $m_y = |\{i : y_i = y\}|$. We assume that m_+ and m_- are nonzero. Then, the algorithm outputs the following decision rule:

$$h(\mathbf{x}) = \begin{cases} 1 & \|\psi(\mathbf{x}) - c_+\| \leq \|\psi(\mathbf{x}) - c_-\| \\ 0 & \text{otherwise.} \end{cases}$$

   1. Let $\mathbf{w} = c_+ - c_-$ and let $b = \frac{1}{2}(\|c_-\|^2 - \|c_+\|^2)$. Show that

$$h(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle + b).$$

   2. Show how to express $h(\mathbf{x})$ on the basis of the kernel function, and without accessing individual entries of $\psi(\mathbf{x})$ or $\mathbf{w}$.