Lecture Outline

Reading: Chapter 6 Kernel Methods

This lecture covers

- Introduction
- Dual Representations
- Constructing Kernels
- Algorithm Summary

Note that project proposals are due Thursday, March 29.

6.0 Introduction

In Chapters 3 and 4, we considered linear parametric models for regression and classification in which the form of the mapping $y(\mathbf{x}, \mathbf{w})$ from input $\mathbf{x}$ to output y is governed by a vector $\mathbf{w}$ of adaptive parameters. During the learning phase, a set of training data is used either to obtain a point estimate of the parameter vector or to determine a posterior distribution over this vector. The training data is then discarded, and predictions for new inputs are based purely on the learned parameter vector $\mathbf{w}$. This approach is also used in nonlinear parametric models such as neural networks.

Figure 1: Block diagrams training and test stages

However, there is a class of pattern recognition techniques, in which the training data points, or a subset of them, are kept and used also during the prediction phase as introduced in Section 2.5. The goal of these methods is to estimate the probability density function or posterior probability. For example, the nearest neighbors classifier involved assigning to each new test vector, the same label as the closest example from the training set. The nearest neighbors classifier is an example of *memory-based* methods that involve storing the entire training set in order to make predictions for future data points.

Figure 2: Block diagram of memory-based classifier

Many linear parametric models can be re-cast into an equivalent 'dual representation' in which the predictions are also based on linear combinations of a *kernel function* evaluated at the training points. As we shall see, for models which are based on a fixed nonlinear *feature space* mapping $\boldsymbol{\phi}(\mathbf{x})$, the kernel function is given by

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}') \tag{1}$$

and arises naturally from the formulation. This kernel function is an inner (or scalar) product in the feature space. From this definition, we see that the kernel is a symmetric function of its arguments so that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. These techniques, known as kernel methods, are the subject of this chapter.

6.1 Dual Representations

Consider a linear regression model whose parameters are determined by minimizing a regularized sum-of-squares error function given by

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \\ &= \frac{1}{2}\mathbf{w}^T(\lambda\mathbf{I} + \boldsymbol{\Phi}^T\boldsymbol{\Phi})\mathbf{w} - \mathbf{w}^T\boldsymbol{\Phi}^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} \end{aligned} \tag{2}$$

where $\boldsymbol{\Phi}$ is the design matrix, whose nth row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^T$ and $\mathbf{t} = [t_1, \ldots, t_N]^T$ is the vector of targets.

The gradient of $J(\mathbf{w})$ with respect to $\mathbf{w}$ is given by

$$\nabla J(\mathbf{w}) = (\lambda\mathbf{I} + \boldsymbol{\Phi}^T\boldsymbol{\Phi})\mathbf{w} - \boldsymbol{\Phi}^T\mathbf{t} \tag{3}$$

and setting $\nabla J(\mathbf{w}) = \mathbf{0}$, we have

$$(\lambda\mathbf{I} + \boldsymbol{\Phi}^T\boldsymbol{\Phi})\mathbf{w} - \boldsymbol{\Phi}^T\mathbf{t} = \mathbf{0}. \tag{4}$$

Instead of the usual solution

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\mathbf{t}, \tag{5}$$

we rewrite (4) in a different form

$$\begin{aligned} \mathbf{w} &= -\frac{1}{\lambda}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{w} + \frac{1}{\lambda}\boldsymbol{\Phi}^T\mathbf{t} \\ &= -\frac{1}{\lambda}\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t}) \\ &= \boldsymbol{\Phi}^T\mathbf{a} \end{aligned} \tag{6}$$

where $\mathbf{a} \equiv -\frac{1}{\lambda}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t})$. In equivalent summation form we have,

$$\mathbf{w} = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) \tag{7}$$

where $a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$. We see in the summation form, the solution for $\mathbf{w}$ takes the form of a linear combination of the vectors $\boldsymbol{\phi}(\mathbf{x}_n)$, with coefficients that are functions of $\mathbf{w}$.

Instead of working with the parameter vector $\mathbf{w}$, we can reformulate the error function in terms of the parameter vector $\mathbf{a}$, giving rise to a *dual representation*. If we substitute the solution in (6) into the cost

function in (2), we obtain

$$\begin{aligned} J(\mathbf{a}) &= \frac{1}{2}\mathbf{a}^T\boldsymbol{\Phi}(\lambda\mathbf{I}+\boldsymbol{\Phi}^T\boldsymbol{\Phi})\boldsymbol{\Phi}^T\mathbf{a}-\mathbf{a}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\mathbf{t}+\frac{1}{2}\mathbf{t}^T\mathbf{t} \\ &= \frac{1}{2}\mathbf{a}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\mathbf{a}-\mathbf{a}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\mathbf{t}+\frac{1}{2}\mathbf{t}^T\mathbf{t}+\frac{\lambda}{2}\mathbf{a}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\mathbf{a} \\ &= \frac{1}{2}\mathbf{a}^T\mathbf{K}\mathbf{K}\mathbf{a}-\mathbf{a}^T\mathbf{K}\mathbf{t}+\frac{1}{2}\mathbf{t}^T\mathbf{t}+\frac{\lambda}{2}\mathbf{a}^T\mathbf{K}\mathbf{a} \end{aligned} \tag{8}$$

where the *Gram* matrix $\mathbf{K}=\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ is an $N\times N$ symmetric matrix with elements

$$\begin{aligned} K_{nm} &= \boldsymbol{\phi}(\mathbf{x}_n)^T\boldsymbol{\phi}(\mathbf{x}_m) \\ &= k(\mathbf{x}_n,\mathbf{x}_m) \end{aligned} \tag{9}$$

with the scalar *kernel function* $k(\mathbf{x}_n,\mathbf{x}_m)$. Note as we will see, we don't necessarily have to compute $\boldsymbol{\phi}(\mathbf{x}_n)$ to get $\mathbf{K}$.

The solution to the dual representation can be found by completing the square

$$\mathbf{a} = (\mathbf{K}+\lambda\mathbf{I}_N)^{-1}\mathbf{t}. \tag{10}$$

We note that the solution is based on the kernel function.

If we substitute this back into the linear regression model, we obtain the following prediction for a new input $\mathbf{x}'$

$$\begin{aligned} y(\mathbf{x}') &= \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}') \\ &= \mathbf{a}^T\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}') \\ &= \boldsymbol{\phi}(\mathbf{x}')^T\boldsymbol{\Phi}^T\mathbf{a} \\ &= \mathbf{k}(\mathbf{x}')^T(\mathbf{K}+\lambda\mathbf{I}_N)^{-1}\mathbf{t} \\ &= \mathbf{k}(\mathbf{x}')^T\mathbf{a} \end{aligned} \tag{11}$$

where the vector $\mathbf{k}(\mathbf{x}')$ has elements $k_n(\mathbf{x}')=k(\mathbf{x}_n,\mathbf{x}')=\boldsymbol{\phi}(\mathbf{x}_n)^T\boldsymbol{\phi}(\mathbf{x}')$.

Figure 3: Block diagrams

Remarks

1. We see that the dual formulation allows the solution in (10) to be expressed entirely in terms of the kernel function $k(\mathbf{x}_n,\mathbf{x}_m)$.
2. In the dual formulation, we determine the parameter vector $\mathbf{a}$ by inverting an $N\times N$ matrix, whereas in the original parameter space formulation we had to invert an $M\times M$ matrix in order to determine $\mathbf{w}$. Because N (number of training data) is typically much larger than M (model order), the dual formulation does not seem to be particularly useful. However, the advantage of the dual formulation, as we shall see, is that it is expressed entirely in terms of the scalar kernel function $k(\mathbf{x}_n,\mathbf{x}_m)$. We can therefore work directly in terms of kernels and avoid the explicit introduction of the feature vector $\boldsymbol{\phi}(\mathbf{x}_n)$, which allows us implicitly to use functions of high, even infinite, dimensionality.

Example: The simplest example of a kernel function is obtained by considering the identity mapping for the feature space so that $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$, in which case $k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^T \mathbf{x}_m$. We shall refer to this as the *linear kernel*.

Example: There are numerous forms of kernel functions and many have the property of being a function only of the difference between the arguments, so that $k(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n - \mathbf{x}_m)$, which are known as *stationary* kernels because they are invariant to translations in input space.

Example: A further specialization involves *homogeneous* kernels, also known as *radial basis functions*, which depend only on the magnitude of the distance between the arguments so that $k(\mathbf{x}_n, \mathbf{x}_m) = k(\|\mathbf{x}_n - \mathbf{x}_m\|)$.

Although neglected for many years, the kernel concept was re-introduced into machine learning in the context of large-margin classifiers giving rise to the technique of *support vector machines*. Since then there has been considerable interest in this topic.

Algorithm Summary

Training

Given: Input training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and associated targets $\{t_1, \ldots, t_N\}$.

Steps:

1 Choose a valid kernel $k(\mathbf{x}_n, \mathbf{x}_m)$ and goto Step 3b or choose a feature space mapping $\boldsymbol{\phi}(\mathbf{x})$ and goto Step 2.

2 Compute the design matrix $\boldsymbol{\Phi}$ $(N \times D)$ whose nth row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^T$.

3a Compute the Gram matrix $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ $(N \times N)$. Goto Step 4

3b Compute the Gram matrix $\mathbf{K}$ $(N \times N)$ where $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$.

4 Compute the optimal solution $\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{t}$ $(N \times 1)$ to the regularized sum-of-squares error function where λ is the regularization term, $\mathbf{I}_N$ is the $N \times N$ identity matrix, and $\mathbf{t} = [t_1, \ldots, t_N]^T$.

Examples of feature space mapping include identity function $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$ or polynomials $\boldsymbol{\phi}(x) = \begin{bmatrix} x^0 & x^1 & \ldots & x^M \end{bmatrix}^T$ (1-D input data).

Examples of kernel functions include the radial basis function (RBF) $k(\mathbf{x}_n, \mathbf{x}_m) = \|\mathbf{x}_n - \mathbf{x}_m\|$ or Gaussian kernel $k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(-\|\mathbf{x}_n - \mathbf{x}_m\|^2 / 2\sigma^2\right)$.

Testing

Given: Input test point $\mathbf{x}'$.

Step:

1 Compute $y(\mathbf{x}') = \mathbf{k}(\mathbf{x}')^T \mathbf{a}$ where $k_n(\mathbf{x}') = k(\mathbf{x}_n, \mathbf{x}')$ or $k_n(\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}')$, $1 \leq n \leq N$.

6.2 Constructing Kernels

The concept of a kernel formulated as an inner product in a feature space allows us to build interesting extensions of many well-known algorithms by making use of the *kernel trick*, also known as *kernel substitution*. The general idea is that, if we have an algorithm formulated in such a way that the input vector $\mathbf{x}$ enters only in the form of scalar products, then we can replace that scalar product with some other choice of kernel.

In order to exploit kernel substitution, we need to be able to construct valid kernel functions, i.e. must correspond to an inner (or scalar) product in feature space.

Approach 1: Choose a feature space mapping $\boldsymbol{\phi}(\mathbf{x})$ and then use this to find the corresponding kernel, as is illustrated in Fig. 1. Here the kernel function is defined for a 1-D input space by

$$\begin{aligned} k(x_n, x_m) &= \boldsymbol{\phi}(x_n)^T \boldsymbol{\phi}(x_m) \\ &= \sum_{i=1}^{M} \phi_i(x_n)\phi_i(x_m) \end{aligned} \tag{12}$$

where $\phi_i(x)$ are the basis functions.

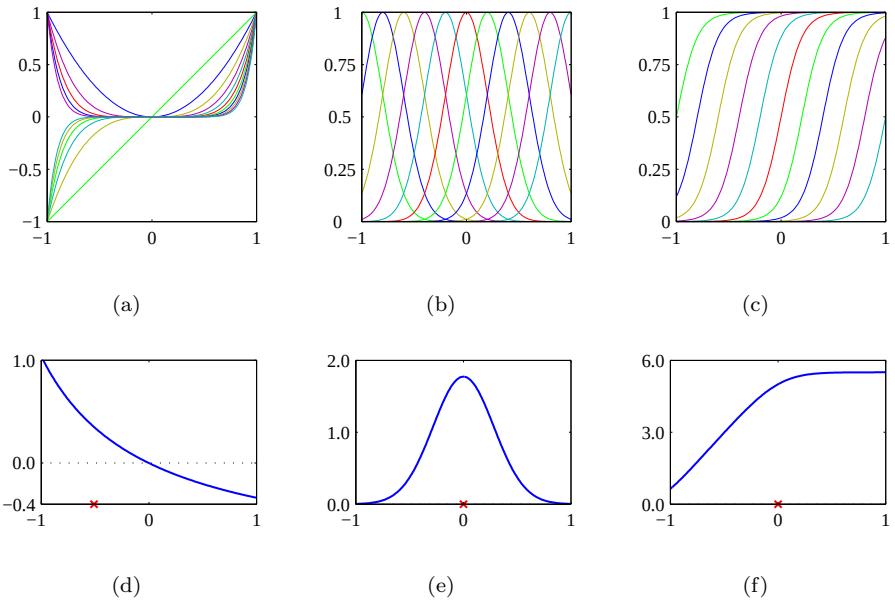

Figure 1: First row shows example $\phi_j(x)$ and the second row shows corresponding $k(x, x')$ (x' is red $\times$ and we sweep x)

Approach 2: We can construct the kernel functions directly, however, we must ensure that the function is a valid kernel, i.e. that it corresponds to a scalar product in some feature space.

Example: Let,

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2 \tag{13}$$

and 2-D input $\mathbf{x} = (x_1, x_2)^T$ and $\mathbf{z} = (z_1, z_2)^T$. Is $k(\cdot, \cdot)$ a valid kernel function?

Expanding we have

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \begin{bmatrix} x_1^2 & \sqrt{2}x_1 x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} z_1^2 \\ \sqrt{2}z_1 z_2 \\ z_2^2 \end{bmatrix} \\ &= \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{z}) \end{aligned} \tag{14}$$

where $\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. Thus for the given feature mapping $\boldsymbol{\phi}(\mathbf{x})$, we have an inner product in feature space and our kernel is valid. Note that our input is 2-D but the feature vector is 3-D–this allows us to push problems into higher dimensional spaces where classification may be easier.

We need a simple way to test whether a function constitutes a valid kernel without having to construct the function $\boldsymbol{\phi}(\mathbf{x})$ explicitly as shown in the above example. A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel is that the Gram matrix $\mathbf{K}$, whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$ should be positive semidefinite[1] for all possible choices of the set $\{\mathbf{x}_n\}$.

Approach 3: One powerful technique for constructing new kernels is to build them using simpler kernels as building blocks. On p. 296 there is a list of properties showing how new valid kernels can be constructed from simpler kernels. Equipped with these properties, we can now embark on the construction of more complex kernels appropriate to specific applications. We require that the kernel $k(\mathbf{x}, \mathbf{x}')$ be symmetric and $\mathbf{K}$ positive semidefinite and that it expresses the appropriate form of similarity between $\mathbf{x}$ and $\mathbf{x}'$ according to the intended application.

Example: Consider the valid, linear kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^T\mathbf{x}_m. \tag{15}$$

From (6.13) $[k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')]$ and (6.16) $[k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))]$, we can obtain another valid kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(\mathbf{x}_n^T\mathbf{x}_m/\sigma^2\right) \tag{16}$$

where σ^2 is a constant.

From (6.14) $[k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')]$ with $f(\mathbf{x}) = \exp(-\mathbf{x}^T\mathbf{x}/2\sigma^2)$, we can obtain another valid kernel

$$\begin{aligned} k(\mathbf{x}_n, \mathbf{x}_m) &= \exp\left(-\mathbf{x}_n^T\mathbf{x}_n/2\sigma^2\right)\exp\left(\mathbf{x}_n^T\mathbf{x}_m/\sigma^2\right)\exp\left(-\mathbf{x}_m^T\mathbf{x}_m/2\sigma^2\right) \\ &= \exp\left(-\|\mathbf{x}_n - \mathbf{x}_m\|^2/2\sigma^2\right). \end{aligned} \tag{17}$$

This kernel is known as the 'Gaussian' or the Radial Basis Function (RBF) kernel. Note that in this context it is not interpreted as a pdf.

An important contribution to arise from the kernel viewpoint has been the extension to inputs that are symbolic, rather than simply vectors of real numbers. Kernel functions can be defined over objects as diverse as graphs, sets, strings, and text documents. Consider, for instance, a fixed set and define a nonvectorial space consisting of all possible subsets of this set. If A_1 and A_2 are two such subsets then one simple choice of kernel would be

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|} \tag{18}$$

where $A_1 \cap A_2$ denotes the intersection of sets A_1 and A_2 and $|A|$ denotes the number of elements in A. Hence, $|A_1 \cap A_2|$ is the number of common elements in the set. This is a valid kernel because it can be shown to correspond to an inner product in a feature space.

[1] A matrix is *positive semidefinite* if $\mathbf{y}^T\mathbf{A}\mathbf{y} \geq 0$ for all $\mathbf{y}$.