KERNEL MODELS AND SUPPORT VECTOR MACHINES

Kazushi Ikeda
Nara Institute of Science and Technology, Ikoma, Nara, Japan

Keywords: kernel, margin, statistical learning theory

Contents

Summary

Kernel models are a method for introducing nonlinear preprocessing to linear models. A kernel function determines a feature space called a reproducing kernel Hilbert space. The solution of an optimization problem with kernel models is expressed as a linear combination of given examples thanks to the representer theorem. This means that no explicit feature vectors are required and hence kernel models are applicable to even sequences or graphs. The first half of this chapter introduces the readers to the fundamentals of kernel models such as the kernel trick and the representer theorem.

The most successful application of kernel models is support vector machines (SVMs). SVMs are derived based on the statistical learning theory, which does not assume any statistical model beforehand and leads to margin maximization for higher generalization ability. Although an SVM is constructed so that only the upper bound of the generalization error in a worst case is minimized, it performs better than conventional methods on average. Another advantage of SVMs is its convexity, that is, it has a unique solution differently from other hierarchical learning models. The idea of margin maximization has been applied to regression, density estimation and other areas. The second half of this chapter introduces the readers to the derivation of SVMs and their properties as well as several variations.

1. Introduction

The simplest model of a neuron is Perceptron that calculates the weighted sum of input signals and outputs +1 when the sum exceeds a threshold and −1 otherwise (Minsky and

Papert, 1969). The model attracted much attention as a binary classifier or dichotomy with an adaptive algorithm for a given dataset called the Perceptron learning rule because it has good properties such as the convergence theorem that assures finite times of iterations for a linearly separable dataset and the cycling theorem that proves periodical solutions otherwise.

One way to introduce a nonlinear preprocessing to Perceptron is the multi-layer Perceptron model (MLP), where the threshold function is replaced with a sigmoid function such as the hyperbolic tangent and the preprocessor is autonomously acquired through learning. This model is simple and works very well but it suffers a lot of local optimal solutions and plateaus in a gradient learning method when the cost function is given as the sum of squared errors.

An alternative to the MLP is a kernel model. The preprocessor of a kernel model must be fixed *a priori* but it has a large variety of choices. In fact, we need only the inner product of preprocessed data, without describing them in an explicit form. This enables the feature space to have even infinite dimensions, or treat graphs directly, which leads to a wide range of applications. Nowadays, kernel models are applied to not only classifiers but also regression problems and other linear signal processing fields.

From the mathematical viewpoint, a kernel model is based on its positive (or nonnegative) definiteness. Positive definite kernels appeared in the literature early in the twentieth century and their general theory was established as the reproducing kernel Hilbert space in 1950's. However, the idea of kernel models has become popular after it was introduced as a nonlinearity for support vector machines or SVMs in 1990's. An SVM is a linear dichotomy like Perceptron but it maximizes the margin, that is, the minimum distance of examples from the discriminative boundary. The idea of margin maximization is based on Vapnik's statistical learning theory that minimizes the structural risk in the framework of the probably approximately correct (PAC) learning introduced by Valiant in 1984.

One of the good properties of SVMs is the structural risk minimization or SRM. The prediction error by SVMs is theoretically bounded even in the worst case and is often much lower than the bound or that of MLPs. Another pro is the uniqueness of the solution. An SVM results in a convex quadratic programming problem that has a unique solution. Moreover, recent development of computer science and engineering such as inner point methods makes the solvable size larger and larger. Thanks to such superiority, SVMs have attracted much attention and their theoretical properties as well as their variants have been well studied.

2. Kernel Function and Feature Space

A kernel method maps an input vector $\mathbf{x} \in X$ to the corresponding feature vector $\mathbf{f}\left(\mathbf{x}\right)$ in a high-dimensional feature space F and processes it linearly, such as Perceptron, the linear regression or the principal component analysis (PCA). Let us consider the Perceptron learning here as a simple example.

Given the t -th example $(\mathbf{x}(t), y(t))$ where $\mathbf{x}(t) \in \{\mathbf{x}_k\}_{k=1}^n$ is the t-th input vector and $y(t)$ is the true label for the input vector, the Perceptron learning rule in the feature space F is formulated as

$$\begin{aligned} &\mathbf{w}(t) = \mathbf{w}(t-1) + y(t)\mathbf{f}(\mathbf{x}(t)) && \text{if } y(t) \neq \operatorname{sgn}\left[\mathbf{w}^{\mathrm{T}}(t)\mathbf{f}(\mathbf{x}(t))\right], \\ &\mathbf{w}(t) = \mathbf{w}(t-1) && \text{otherwise,} \end{aligned} \tag{1}$$

where $\mathbf{w}(t)$ is the weight vector of Perceptron at iteration t and T denotes the transpose. Here, we assume that the weight vector can be expressed as a weighted sum of inputs, that is,

$$\mathbf{w}(t) = \sum_{k=1}^{n} \alpha_k(t)\mathbf{f}(\mathbf{x}_k) \tag{2}$$

Then, the algorithm (1) is simply rewritten as

$$\begin{aligned} &\alpha_k(t) = \alpha_k(t-1) + y(t) && \text{if } y(t) \neq \operatorname{sgn}\left[\mathbf{w}^{\mathrm{T}}(t)\mathbf{f}(\mathbf{x}(t))\right], \mathbf{x}(t) = \mathbf{x}_k \\ &\alpha_k(t) = \alpha_k(t-1) && \text{otherwise,} \end{aligned} \tag{3}$$

for all $k = 1, \rightleftharpoons, n$. This is called the kernel Perceptron. Here, the inner product of $\mathbf{w}(t)$ and $\mathbf{f}(\mathbf{x})$ for an arbitrary $\mathbf{x} \in X$ is calculated as

$$\mathbf{w}^{\mathrm{T}}(t)\mathbf{f}(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k(t)\mathbf{f}^{\mathrm{T}}(\mathbf{x}_k)\mathbf{f}(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k(t)K(\mathbf{x}_k, \mathbf{x}). \tag{4}$$

where $K(\mathbf{x}_k, \mathbf{x}) = \mathbf{f}^{\mathrm{T}}(\mathbf{x}_k)\mathbf{f}(\mathbf{x})$ is called a kernel function. Hence, although we do not know the feature vector $\mathbf{f}(\mathbf{x})$ itself, we can calculate the inner product. This makes the feature vector very high-dimensional or even infinite without increase of computational complexity, which is called the **kernel trick**.

One example of kernel functions is the p -th order polynomial kernel $K(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}' + 1\right)^p$. When $\mathbf{x} \in R^2$ and $n = 2$, a possible feature vector is

$$\mathbf{f}(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right)^{\mathrm{T}}, \tag{5}$$

and hence $\mathbf{w}^{\mathrm{T}}(t)\mathbf{f}(\mathbf{x})$ can express any polynomial function with order two or less. Another popular kernel function is Gaussian kernel $K(\mathbf{x},\mathbf{x}')=\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$, whose feature vector is essentially the Fourier transform as shown below:

Suppose that the feature of $\mathbf{x}$ indexed by $\omega\in R$ is $\exp(-i\omega x)$ where ω and x are one-dimensional for simplicity. Then, the inner product of $\exp(-i\omega x)$ and $\exp(-i\omega x')$ is calculated as

$$\begin{aligned} K(x,x') &= \frac{1}{\sqrt{2\pi}}\int \exp(-i\omega x)\exp(-i\omega x')\exp\left(-\frac{\omega^2}{2}\right)d\omega \\ &= \frac{1}{\sqrt{2\pi}}\int \exp\left(-\frac{(\omega-i(x-x'))^2}{2}\right)\exp\left(-\frac{(x-x')^2}{2}\right)d\omega \qquad (6) \\ &= \exp\left(-\frac{(x-x')^2}{2}\right) \end{aligned}$$

where $\exp\left(-\omega^2/2\right)\Big/\sqrt{2\pi}$ is a measure of the space of ω.

An advantage of kernel models is that we need not consider the feature space for a kernel function explicitly. This property makes kernel models possible to apply to the spaces of statistical models, sequences, and even graphs. In fact, Mercer's theorem assures the existence of an inner product space F for any positive semi definite (or non-negative definite) kernel function $K(\cdot,\cdot)$, where a symmetric continuous function $K(\cdot,\cdot)$ is said to be positive semi definite if and only if it satisfies

$$\sum_{k=1}^{n}\sum_{m=1}^{n} c_k c_m K(\mathbf{x}_k,\mathbf{x}_m)\geq 0 \qquad (7)$$

for any vector $\{\mathbf{x}_k\}_{k=1}^{n}$, any real number $\{c_k\}_{k=1}^{n}$ and any positive integer n. The Gaussian kernel, for example, has a feature space thanks to this property although it is difficult to describe explicitly.

From the practical viewpoint, however, proving a kernel function being positive semi definite is more difficult than constructing a feature space explicitly. Hence, popular kernel functions are simple ones such as the polynomial kernel or Gaussian kernel, or their combinations. In fact, when two functions K_1 and K_2 are positive semi definite, so are the following functions:

$$c_1K_1+c_2K_2 \text{ for } c_1,c_2\geq 0,\quad K_1K_2,\quad \exp(K_1),\quad g(\mathbf{x})K_1(\mathbf{x},\mathbf{x}')g(\mathbf{x}'). \qquad (8)$$

The polynomial kernel as well as Gaussian kernel are proven to be positive semi definite using (8).

What kernel should we use? The answer strongly depends on the application we consider but an alternative approach is “kernel learning”, that is, automatic selection of optimal kernel from data. One typical example is to optimize the coefficients $\{c_i\}_{i=1}^{j}$ of a weighted sum of kernels, $\sum_{i=1}^{j} c_i K_i$, so that the performance of the learning machine becomes best.

3. Representer Theorem

In the previous section, we assumed that the weight vector could be expressed as a weighted sum of inputs, that is,

$$\mathbf{w}(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k K(\mathbf{x}, \mathbf{x}_k). \tag{9}$$

A general theory to justify the above assumption is the Representer Theorem, which says the solution of the optimization problem below has a solution $\mathbf{w}$ of the form in (9):

$$\min_{\mathbf{w}\in H, c\in R^j} L\left(\{\mathbf{x}_k\}_{k=1}^{n}, \{y_k\}_{k=1}^{n}, \left\{ \mathbf{w}(\mathbf{x}_k) + \sum_{i=1}^{j} c_i h_i(\mathbf{x}_k) \right\}_{k=1}^{n} \right) + \Psi(\|\mathbf{w}\|), \tag{10}$$

where $\{(\mathbf{x}_k, y_k)\}_{k=1}^{n}$ is a dataset, K is a kernel function of $\mathbf{x}$, H is the reproducing kernel Hilbert space of K, $\{h_i\}_{i=1}^{j}$ is a set of basis functions of $\mathbf{x}$, and $\Psi(\cdot)$ is a monotonically increasing function (regularization term). Intuitively speaking, any component in the complement space $H_0^{\perp}$ of $H_0 = \mathrm{span}\left(\{K(\cdot, \mathbf{x}_k)\}_{k=1}^{n}\right)$ in $\mathbf{w}$ does not change the cost L but only increases the normalization term Ψ.

One simple example of L is

$$\min_{\mathbf{w}\in H} \frac{1}{n} \sum_{k=1}^{n} \left(y_k - \mathbf{w}(x_k) \right)^2 + \lambda \|\mathbf{w}\|_H^2 \tag{11}$$

and another example is the support vector machine as shown later. Thanks to the kernel trick, the computational complexity of kernel methods does not depend on the dimensions of the feature space but only on the number of examples n.

-

-

-

Bibliography

Minsky M.L., Papert S.A. (1969). Perceptrons, MIT Press, Cambridge, MA. [This is a comprehensive textbook for perceptrons].

Rumelhert D.E., McClelland J.L. (1986). *Parallel Distributed Processing*, MIT Press, Cambridge, MA. [This presents a theory for multi-layer perceptrons and their applications].

Fukumizu K (2010). *Introduction to Kernel Methods*, Asakura-Shoten, Tokyo. [This presents a theoretical background of kernel methods for beginners. Note that it is written in Japanese].

Watanabe S (2001). Algebraic Analysis for Nonidentifiable Learning Machines. *Neural Computation*, **13**(4), 899-933. [This is an epoch-making paper introducing algebraic geometry to machine learning theory].

Vapnik V.N. (1995). *The Nature of Statistical Learning Theory*, Springer, Heidelberg. [This is the first book introducing support vector machines].

Valiant L.G. (1984). A Theory of Learnable. *Communications of the ACM*, **27**(11), 1134-1142. [This paper presents a framework of the PAC learning].

Kwok J.T., Tsang I.W. (2004). The Pre-Image Problem in Kernel Methods, *IEEE Transactions on Neural Networks*, **15**(6), 1517-1525. [This gives a concrete method to cope with the pre-image problem of kernel methods].
Schoelkopf B., Smola A.J., Williamson R.C., Bartlett P.L. (2000). New Support Vector Algorithms, *Neural Computation*, **12**, 1207-1245. [This paper introduces the framework of the ν -SVMs].

Bennett K.P., Bredensteiner E.J. (2000). Duality and Geometry in SVM Classifiers, *Proc. ICML*, 57-64. [This is the first paper to show another view of the ν -SVMs, that is, the convex hulls of data].

Ikeda K. (2004). An Asymptotic Statistical Theory of Polynomial Kernel Methods. *Neural Computation*, **16**(8), 1705-1719. [This paper solves the contradiction of the theoretical results on SVMs].

Ikeda K., Aoishi T. (2005). An Asymptotic Statistical Analysis of Support Vector Machines with Soft Margins, *Neural Computation*, **18**, 251-259. [This paper gives a theoretical analysis of the effect of soft margins].

Schoelkopf B., Platt J.C., Shawe-Taylor J., Smola A.J. (2001). Estimating the Support of a High-Dimensional Distribution, *Neural Computation*, **13**, 1443-1471. [This paper applies the idea of support vector macchines to a density estimation by formulating it as a one-class SVM].

Biographical Sketch

Kazushi Ikeda received the B.E., M.E. and Ph.D. degrees in Mathematical Engineering and Information Physics from the University of Tokyo, Tokyo, Japan, in 1989, 1991 and 1994, respectively. He then joined the Department of Electrical and Computer Engineering, Kanazawa University, Kanazawa, Japan, as an Assistant Professor and moved to the Department of Systems Science, Kyoto University, Kyoto, Japan, as an Associate Professor, in 1998. Since 2008, he has been a Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests include machine learning theory such as support vector machines and information geometry, and applications to adaptive systems and brain informatics. He is currently the Editor-in-Chief of Journal of Japanese Neural Network Society (JNNS), an Action Editor of Neural Networks (Elsevier) and an Associate Editor of IEEE Transactions on Neural Networks and Learning Systems and IEICE Transactions on Information and Systems. He has served as a member of Board of Governors of JNNS.