BIOINFORMATICS

Vol. 18 Suppl. 1 2002
Pages S268–S275

Marginalized kernels for biological sequences

Koji Tsuda, Taishin Kin and Kiyoshi Asai

Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-41-6 Aomi Koto-ku, Tokyo, 135-0064, Japan

Received on January 24, 2002; revised and accepted on April 1, 2002

ABSTRACT

Motivation: Kernel methods such as support vector machines require a kernel function between objects to be defined *a priori*. Several works have been done to derive kernels from probability distributions, e.g., the Fisher kernel. However, a general methodology to design a kernel is not fully developed.

Results: We propose a reasonable way of designing a kernel when objects are generated from latent variable models (e.g., HMM). First of all, a *joint* kernel is designed for complete data which include both visible and hidden variables. Then a *marginalized* kernel for visible data is obtained by taking the expectation with respect to hidden variables. We will show that the Fisher kernel is a special case of marginalized kernels, which gives another viewpoint to the Fisher kernel theory. Although our approach can be applied to any object, we particularly derive several marginalized kernels useful for biological sequences (e.g., DNA and proteins). The effectiveness of marginalized kernels is illustrated in the task of classifying bacterial gyrase subunit B (*gyrB*) amino acid sequences.

Contact: koji.tsuda@aist.go.jp

Keywords: kernel design; marginalized kernels; the Fisher kernel; biological sequence classification; string kernels.

INTRODUCTION

In kernel methods such as support vector machines (Müller *et al.*, 2001), a kernel function between two objects should be determined a priori. In supervised learning algorithms, the objective function to be optimized is clearly stated (e.g., the expected risk), so one should determine the kernel to optimize this function or its approximations (e.g., the leave-one-out error). However, in unsupervised learning algorithms such as clustering, the choice of kernel is quite subjective. The kernel is determined to reflect the user's notion of similarity, which cannot be justified nor falsified completely. However, it is not an easy task to describe your notion of similarity as a positive semidefinite kernel.

DNA and proteins are symbol sequences which may have different lengths. So they have similar characteristics to other symbol sequences such as texts (Frakes and Baeza-Yates, 1992). In order to measure similarity between such sequences, it is common to extract *count features*, which represent the number of each symbol contained in a sequence. Then, the similarity is obtained as the weighted dot product between these features, where a smaller weight is assigned for the symbols which appear frequently. Although this approach (i.e., vector space representation (Frakes and Baeza-Yates, 1992)) achieved a great success for texts, it is not appropriate for biological sequences. The primal reason is that the *context* changes frequently in one sequence. For example, a DNA sequence has coding and noncoding regions, whose statistical properties are quite different. The residuals 'A' in coding and noncoding regions have different meanings. Although it is difficult to determine the boundaries of the regions, they should be counted separately. If a sequence of hidden variables which describe the context are available, it would be easier to design a kernel (Figure 1). However, hidden variables are unknown in general, and have to be estimated.

In this paper, we propose a new reasonable way to design a kernel. First, a kernel between sequences is defined depending both on visible and hidden variables. Because this kernel requires both visible and hidden variables for calculation, we call it a *joint kernel*. Since hidden information is *assumed* to be available, the kernel can be designed depending on the hidden context of sequences. However, the problem is that such hidden information is actually not available. To cope with this problem, the posterior distribution of hidden variables are estimated by means of a probabilistic model such as HMM. Then, we obtain a *marginalized kernel* by taking expectation of the joint kernel with respect to hidden variables.

We will show that the Fisher kernel (Jaakkola and Haussler, 1999)—which has been successfully applied to many tasks e.g., protein classification (Jaakkola *et al.*, 2000; Karchin *et al.*, 2002) and promoter region detection (Pavlidis *et al.*, 2001)—is a special case of marginalized kernels. This reveals the joint kernel implicitly assumed in the Fisher kernel, which helps us to understand the Fisher kernel more in detail. For biological

© Oxford University Press 2002

h: 1 2 2 1 2 2 1 2 2

x: A C G G T T C A A

Fig. 1. A DNA sequence with hidden context information. Suppose the hidden variable ('h' in the figure) indicates e.g., coding/noncoding regions. If hidden variables are known, it would be much easier to design a kernel function between sequences.

sequences, we propose useful kernels, called *marginalized count kernels* (MCKs). In order to illustrate the effectiveness of our kernels, we will perform experiments to classify bacterial *gyrB* amino acid sequences (Kasai *et al.*, 2000). As a result, it is shown that MCKs compare favorably to the Fisher kernel.

METHODS

Marginalized kernels

Let us describe a visible variable as $x \in \mathcal{X}$, where the domain $\mathcal{X}$ is a finite set[†]. Our task is to define a kernel $K(x, x')$ between two visible variables x, x'. Suppose we have a hidden variable $h \in \mathcal{H}$, where $\mathcal{H}$ is a finite set. By utilizing the hidden information in h, the joint kernel $K_z(z, z')$ is designed between two combined variables $z = (x, h), z' = (x', h')$. The *marginalized* kernel in $\mathcal{X}$ is derived by taking the expectation with respect to hidden variables:

$$K(x, x') = \sum_{h \in \mathcal{H}} \sum_{h' \in \mathcal{H}} p(h|x) p(h'|x') K_z(z, z'). \quad (1)$$

The posterior distribution $p(h|x)$ is unknown in general, and has to be estimated from the data e.g., by HMMs. The calculation of K can be intractable when the cardinality of $\mathcal{H}$ is too large. However, for useful stochastic models such as HMMs, there are algorithms which enable efficient computation as shown in later sections. Since the class of positive semidefinite (Mercer) kernels are closed under addition and multiplication (Haussler, 1999), the marginalized kernel K is positive semidefinite as long as the joint kernel is positive semidefinite. In convolution kernels (Haussler, 1999), sub-kernels for the parts are aggregated into a kernel for the whole set. This is constrastive to our approach deriving the kernel for the part (i.e., visible variables) from the kernel for the whole set (i.e., visible and hidden variables).

Marginalized kernel from Gaussian mixture

For intuitive understanding, we provide a simple example of marginalized kernels. Here we derive a marginalized kernel from Gaussian mixture with m-components. The visible variable is a point in the d-dimensional space $\boldsymbol{x} \in \Re^d$, and the hidden variable is an index of component $h \in \{1, \ldots, m\}$. The probabilistic model is written as $p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{h=1}^{m} p(h) q(\boldsymbol{x}|h, \boldsymbol{\mu}_h, A_h)$, where the hth component is a Gaussian distribution with mean $\boldsymbol{\mu}_h$ and covariance matrix A_h^{-1}:

$$q(\boldsymbol{x}|h, \boldsymbol{\mu}_h, A_h) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_h)^\top A_h (\boldsymbol{x} - \boldsymbol{\mu}_h))}{2\pi |A_h^{-1}|^{1/2}}. \quad (2)$$

Let us define the joint kernel as $K_z(z, z') = I(h = h')(\boldsymbol{x}^\top A_h \boldsymbol{x}')$, where $I(t)$ is the indicator function which is 1 if the condition t is true and 0 otherwise. Note that K_z is a positive semidefinite kernel. The marginalized kernel is obtained as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h=1}^{m} p(h|\boldsymbol{x}) p(h|\boldsymbol{x}') \boldsymbol{x}^\top A_h \boldsymbol{x}'. \quad (3)$$

We illustrate the shape of this kernel in Figure 2. Here, the kernel is converted to the distance in the feature space

$$D(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{K(\boldsymbol{x}, \boldsymbol{x}) + K(\boldsymbol{x}', \boldsymbol{x}') - 2K(\boldsymbol{x}, \boldsymbol{x}')}. \quad (4)$$

Fixing $\boldsymbol{x}'$ at a point, the contours of distance $D(\boldsymbol{x}, \boldsymbol{x}')$ are shown. When $\boldsymbol{x}'$ belongs to one cluster, the contour shape is similar to the shape of the cluster. The shape gradually changes when the point $\boldsymbol{x}'$ moves to one cluster to the other. In comparison with the Euclidean distance, this distance emphasizes the cluster structure. This kind of kernel is considered to be useful in visualizing cluster structure in a high dimensional space (Tipping, 1999).

Marginalized count kernel

Next we propose an important example of marginalized kernels for biological sequences. Let $\boldsymbol{x} = (x_1, \ldots, x_m)$, $x_i \in \{1, \ldots, n_x\}$ denote a symbol sequence of length m. Assume that each sequence can have a different length. One simple kernel for such sequences is the *count kernel*:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{n_x} c_k(\boldsymbol{x}) c_k(\boldsymbol{x}'), \quad (5)$$

where $c_k(\boldsymbol{x})$ is the number of symbol k normalized by the length:

$$c_k(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} I(x_i = k).$$

The count kernel is often used in text processing literatures (Frakes and Baeza-Yates, 1992), but it is not suitable for biological sequences because of frequent context

[†] Here we determine $\mathcal{X}$ as a finite set for simplicity, but the results in this paper can be easily extended to continuous domains.

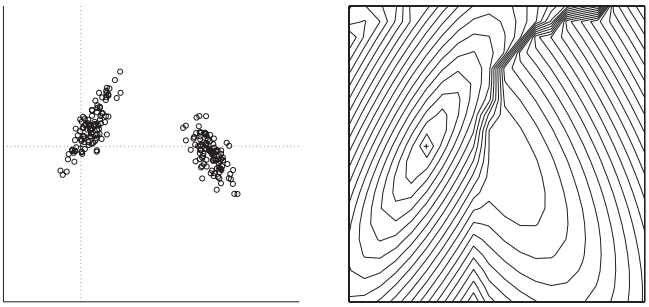

Fig. 2. Contours of the marginalized kernel-based distance from a specified point. The left figure shows sample points generated from the Gaussian mixture model. Here the crossing point of dotted lines indicates the central point from which the kernel-based distance (4) is measured. The right figure shows the distance contours. The contour shapes are adapted to the shape of cluster which the central point belongs to. Compared with the Euclidean distance, this distance emphasizes the cluster structure.

changes. We are going to extend the count kernel to include (hidden) context information.

Assume that there is a sequence of hidden variables $\boldsymbol{h} = (h_1, \ldots, h_m)$, $h_i \in \{1, \ldots, n_h\}$. Define the combined sequence as

$$\boldsymbol{z} := (z_1, \ldots, z_m) = (\{x_1, h_1\}, \ldots, \{x_m, h_m\})$$

where each z_i can have $n_z = n_x n_h$ symbols. The count kernel for $\boldsymbol{z}$ can be defined as

$$K_z(\boldsymbol{z}, \boldsymbol{z}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} c_{k\ell}(\boldsymbol{z}) c_{k\ell}(\boldsymbol{z}'),$$
$$c_{k\ell}(\boldsymbol{z}) = \frac{1}{m} \sum_{i=1}^{m} I(x_i = k, h_i = \ell). \qquad (6)$$

When $\boldsymbol{h}$ is regarded as the context information, symbols are counted separately in each context (Figure 3). For example, if the sequences are DNA and the hidden contexts are exon/intron, the frequencies of 'A', 'C', 'G', 'T' are counted and compared separately for exon/intron.

Setting (6) as a joint kernel, the *marginalized count kernel* is defined as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h}'} p(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{h}'|\boldsymbol{x}') K_z(\boldsymbol{z}, \boldsymbol{z}'), \qquad (7)$$

where $\sum_{\boldsymbol{h}} = \sum_{h_1=1}^{n_h} \cdots \sum_{h_m=1}^{n_h}$. This kernel (7) is rewritten as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \gamma_{kl}(\boldsymbol{x}) \gamma_{kl}(\boldsymbol{x}'),$$

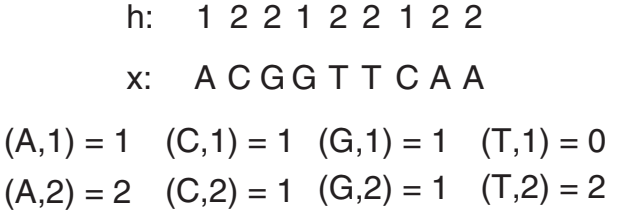

Fig. 3. Illustration of the marginalized count kernel (MCK1). Each feature is obtained by counting the number of combined symbols, and the joint kernel is defined as the dot product between these features. Finally, MCK1 is obtained by marginalizing the joint kernel.

where the marginalized counts $\gamma_{kl}(\boldsymbol{x})$ are described as

$$\gamma_{kl}(\boldsymbol{x}) = \frac{1}{m} \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}) \sum_{i=1}^{m} I(x_i = k, h_i = \ell)$$
$$= \frac{1}{m} \sum_{i=1}^{m} \sum_{h_i=1}^{n_h} p(h_i|\boldsymbol{x}) I(x_i = k, h_i = \ell).$$

Fortunately, the sum over all hidden variables $\boldsymbol{h}$ can be replaced by the sum over each h_i, which reduces the computational cost.

When the probability distribution $p(\boldsymbol{x})$ is represented as HMM, the posterior probability $p(h_i|\boldsymbol{x})$ is computed easily by the forward-backward algorithm (Durbin *et al.*, 1998). An HMM is described as

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{\boldsymbol{h}} q_{h_1} e_{h_1 x_1} [\prod_{i=2}^{m} a_{h_{i-1} h_i} e_{h_i x_i}] d_{h_m}, \qquad (8)$$

where the parameters $\boldsymbol{\theta} = \{a, e, q, d\}$ are transition probabilities, emission probabilities, initial state distribution and terminal state distribution, respectively. The forward and backward algorithms provide the following probabilities, respectively:

$$f_k(i) = p(x_1, \ldots, x_i, h_i = k),$$
$$b_k(i) = p(x_{i+1}, \ldots, x_m | h_i = k).$$

Then the posterior probability is described as

$$p(h_i = \ell|\boldsymbol{x}) = \frac{f_\ell(i) b_\ell(i)}{p(\boldsymbol{x})}$$

which is known as $\gamma_i(\ell)$ in HMM literatures (Rabiner, 1989).

Second-order marginalized count kernel

When adjacent relations between symbols have essential meanings, the count kernel is obviously not sufficient. In

such cases, it would be better to count the number of combinations of two adjacent symbols (Figure 4). The dot product of such counts is described as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{n_x} \sum_{k'=1}^{n_x} c_{kk'}(\boldsymbol{x}) c_{kk'}(\boldsymbol{x}'), \tag{9}$$

$$c_{kk'}(\boldsymbol{x}) = \frac{1}{m-1} \sum_{i=1}^{m-1} I(x_i = k) I(x_{i+1} = k').$$

We call it the *second order count kernel*. Incorporating hidden variables to (9), one can easily extend the marginalized count kernel to second order: Let us define

$$\delta^i_{k\ell k'\ell'} := I(x_i = k, h_i = \ell, x_{i+1} = k', h_{i+1} = \ell').$$

The joint kernel is described as

$$K_z(\boldsymbol{z}, \boldsymbol{z}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \sum_{k'=1}^{n_x} \sum_{\ell'=1}^{n_h} c_{k\ell k'\ell'}(\boldsymbol{z}) c_{k\ell k'\ell'}(\boldsymbol{z}'), \tag{10}$$

where $c_{k\ell k'\ell'}(\boldsymbol{z}) = \frac{1}{m-1} \sum_{i=1}^{m-1} \delta^i_{k\ell k'\ell'}$. The marginalized kernel of (10) is described as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \sum_{k'=1}^{n_x} \sum_{\ell'=1}^{n_h} v_{k\ell k'\ell'}(\boldsymbol{x})^\top v_{k\ell k'\ell'}(\boldsymbol{x}') \tag{11}$$

where

$$\begin{aligned} v_{k\ell k'\ell'}(\boldsymbol{x}) &= \frac{1}{m-1} \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}) \sum_{i=1}^{m-1} \delta^i_{k\ell k'\ell'} \\ &= \frac{1}{m-1} \sum_{i=1}^{m-1} \sum_{h_i=1}^{n_h} \sum_{h_{i+1}=1}^{n_h} p(h_i, h_{i+1}|\boldsymbol{x}) \delta^i_{k\ell k'\ell'}. \end{aligned}$$

We call it the *second order marginalized count kernel*. As in the first order case, the posterior probability $p(h_i, h_i + 1|\boldsymbol{x})$ is obtained from forward and backward algorithms as

$$p(h_i = \ell, h_{i+1} = \ell'|\boldsymbol{x}) = a_{\ell\ell'} e_{\ell' x_{i+1}} \frac{f_\ell(i) b_{\ell'}(i+1)}{p(\boldsymbol{x})}.$$

Note that this quantity is well known as $\xi_i(\ell, \ell')$ parameter in Baum-Welch algorithm, which gives estimation of transition probabilities in HMMs. The second order marginalized count kernel is particularly useful, because it can utilize second order information as well as hidden context information. Higher order extension is straightforward, but not written here for brevity.

Connections to the Fisher kernel

In the following, we will show that the Fisher kernels (Jaakkola and Haussler, 1999) derived from latent variable models are described as marginalized kernels. This section will give a new analysis to explain the nature of the Fisher kernel.

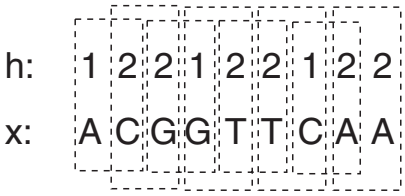

Fig. 4. Illustration of the second-order marginalized count kernel (MCK2). Each feature is obtained by counting the number of combinations of two adjacent symbols. The joint kernel is defined as the dot product between these features, and MCK2 is obtained by marginalization.

Definition of the Fisher kernel Assume a probabilistic model $p(x|\boldsymbol{\theta})$ is defined on $\mathcal{X}$, where $\boldsymbol{\theta}$ is a r-dimensional parameter vector. Let $\hat{\boldsymbol{\theta}}$ denote parameter values which are obtained by some learning algorithm (e.g., maximum likelihood). Then the Fisher kernel (FK) (Jaakkola and Haussler, 1999) between two objects is defined as

$$K_f(x, x') = \boldsymbol{s}(x, \hat{\boldsymbol{\theta}})^\top Z^{-1}(\hat{\boldsymbol{\theta}}) \boldsymbol{s}(x', \hat{\boldsymbol{\theta}}), \tag{12}$$

where $\boldsymbol{s}$ is the Fisher score

$$\begin{aligned} \boldsymbol{s}(x, \hat{\boldsymbol{\theta}}) &= \left(\frac{\partial}{\partial \theta_1} \log p(x|\hat{\boldsymbol{\theta}}), \ldots, \frac{\partial}{\partial \theta_r} \log p(x|\hat{\boldsymbol{\theta}}) \right)^\top \\ &:= \nabla_{\boldsymbol{\theta}} \log p(x|\hat{\boldsymbol{\theta}}), \end{aligned}$$

and Z is the Fisher information matrix:

$$\begin{aligned} Z(\hat{\boldsymbol{\theta}}) &= \mathrm{E}\left[\boldsymbol{s}(x, \hat{\boldsymbol{\theta}}) \boldsymbol{s}(x, \hat{\boldsymbol{\theta}})^\top \middle| \hat{\boldsymbol{\theta}} \right] \\ &= \sum_{x \in \mathcal{X}} p(x|\hat{\boldsymbol{\theta}}) \boldsymbol{s}(x, \hat{\boldsymbol{\theta}}) \boldsymbol{s}(x, \hat{\boldsymbol{\theta}})^\top. \end{aligned}$$

The Fisher kernel is a general method which can be applied for any objects. However, the Fisher kernel is particularly effective for biological sequences when combined with HMMs (Jaakkola *et al.*, 2000; Karchin *et al.*, 2002; Pavlidis *et al.*, 2001).

The Fisher Kernel from Latent Variable Models When a latent variable model $p(x|\boldsymbol{\theta}) = \sum_{h \in \mathcal{H}} p(x, h|\boldsymbol{\theta})$ is adopted, the Fisher score is described as

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log p(x|\hat{\boldsymbol{\theta}}) &= \frac{\sum_{h \in \mathcal{H}} \nabla_{\boldsymbol{\theta}} p(x, h|\hat{\boldsymbol{\theta}})}{p(x|\hat{\boldsymbol{\theta}})} \\ &= \sum_{h \in \mathcal{H}} \frac{p(x, h|\hat{\boldsymbol{\theta}})}{p(x|\hat{\boldsymbol{\theta}})} \frac{\nabla_{\boldsymbol{\theta}} p(x, h|\hat{\boldsymbol{\theta}})}{p(x, h|\hat{\boldsymbol{\theta}})} \\ &= \sum_{h \in \mathcal{H}} p(h|x, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}} \log p(x, h|\hat{\boldsymbol{\theta}}). \end{aligned}$$

So, the Fisher kernel is described as a marginalized kernel

$$K_f(x,x') = \nabla_{\boldsymbol{\theta}} \log p(x|\hat{\boldsymbol{\theta}})^\top Z(\hat{\boldsymbol{\theta}})^{-1} \nabla_{\boldsymbol{\theta}} \log p(x'|\hat{\boldsymbol{\theta}}) := \sum_{h\in\mathcal{H}} \sum_{h'\in\mathcal{H}} p(h|x,\hat{\boldsymbol{\theta}}) p(h'|x',\hat{\boldsymbol{\theta}}) K_z(z,z'), \tag{13}$$

where the joint kernel is described as $K_z(z,z') = \nabla_{\boldsymbol{\theta}} \log p(x,h|\hat{\boldsymbol{\theta}})^\top Z(\hat{\boldsymbol{\theta}})^{-1} \nabla_{\boldsymbol{\theta}} \log p(x',h'|\hat{\boldsymbol{\theta}})$. Thus the Fisher kernel is one special case in the class of marginalized kernels. One characteristic aspect of the Fisher kernel is that the joint kernel is determined by the probabilistic model, while, in our approach, the joint kernel is designed to fit user's purposes. Since the joint kernel of the Fisher kernel is not suitable for every purpose, you have to check whether it fits your purpose or not. If not, the joint kernel should be engineered.

The Fisher kernel from HMM In this section, we derive the Fisher kernel from HMM (8) and discuss its connection to marginalized count kernels. The joint distribution of HMM is described as

$$p(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta}) = q_{h_1} e_{h_1 x_1} [\prod_{i=2}^{m} a_{h_{i-1}h_i} e_{h_i x_i}] d_{h_m}.$$

As in the literature (Jaakkola *et al.*, 2000), we take the derivatives with respect to emission probabilities e only:

$$\frac{\partial}{\partial e_{\ell k}} \log p(\boldsymbol{x},\boldsymbol{h}|\hat{\boldsymbol{\theta}}) = \frac{m c_{k\ell}(\boldsymbol{z})}{\hat{e}_{\ell k}} - m \sum_{k=1}^{n_x} c_{k\ell}(\boldsymbol{z}), \tag{14}$$

where $\hat{e}_{\ell k}$ is the estimated emission probability and $c_{k\ell}(\boldsymbol{z})$ is defined as (6). Note that the second term of (14) comes from the constraint of emission probabilities $\sum_{k=1}^{n_x} e_{\ell k} = 1$. If we do not use the Fisher information matrix as in (Jaakkola *et al.*, 2000), the joint kernel is described as

$$\nabla_{\boldsymbol{e}} \log p(\boldsymbol{x},\boldsymbol{h}|\hat{\boldsymbol{\theta}})^\top \nabla_{\boldsymbol{e}} \log p(\boldsymbol{x}',\boldsymbol{h}'|\hat{\boldsymbol{\theta}}). \tag{15}$$

This is rewritten as

$$\sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \frac{m m'}{\hat{e}_{\ell k}^2} (c_{k\ell}(\boldsymbol{z}) - \bar{c}_{k\ell}(\boldsymbol{z}))(c_{k\ell}(\boldsymbol{z}') - \bar{c}_{k\ell}(\boldsymbol{z}')), \tag{16}$$

where $\bar{c}_{k\ell}(\boldsymbol{z}) = \hat{e}_{\ell k} \sum_{k'=1}^{n_x} c_{k'\ell}(\boldsymbol{z})$. This has a similar form to the count kernel (6), however the count is centralized and the dot product is taken with respect to the weight $\alpha_{k\ell} = \frac{m m'}{\hat{e}_{\ell k}^2}$. The weight is dependent on the length m, so a proper normalization is needed for the Fisher kernel. Since $e_{\ell k}$ represent the emission probability that symbol k is produced from state ℓ, the weight becomes large when the symbol k is rarely produced from state ℓ. It makes sense, because the cooccurence of a rare symbol is a strong clue of high similarity. However this weight is still argueable, because a huge weight can appear when $e_{\ell k}$ is very small.

DISCUSSION

In the previous section, we derived the Fisher kernel only from emission probabilities. However, if you take the derivatives of transition probabilities as well, you obtain a different joint kernel from the one shown in (16). How should we choose the subset of parameters to take derivatives? More generally, you can derive a new parameter as a function of a subset of original parameters. How about taking the derivative with respect to the new parameter?

As suggested in this example, one theoretical problem about the Fisher kernel is that it depends not only on the distribution itself, but also the parametrization which a user has *intentionally* chosen. Consider two parametric models:

$$p(x,h|\boldsymbol{\theta}),\ \boldsymbol{\theta}\in\Re^{r_1}, \quad p(x,h|\boldsymbol{\mu}),\ \boldsymbol{\mu}\in\Re^{r_2}, \quad r_1 \neq r_2.$$

Let us assume that a joint distribution $p(x,h)$ is represented by two different parametric models:

$$p(x,h) = p(x,h|\hat{\boldsymbol{\theta}}) = p(x,h|\hat{\boldsymbol{\mu}}).$$

In general, the Fisher kernels derived from $p(x|\hat{\boldsymbol{\theta}})$ and $p(x|\hat{\boldsymbol{\mu}})$ are different although the underlying distribution is the same[‡]. Since there is no admitted way to choose proper parametrization so far, it is basically determined by trial and error.

In (13) we represented the Fisher kernel as a function of the joint kernel and posterior probabilities of hidden variables. While the joint kernel is not invariant to parametrization,

$$\nabla_{\boldsymbol{\theta}} \log p(x,h|\hat{\boldsymbol{\theta}})^\top Z(\hat{\boldsymbol{\theta}})^{-1} \nabla_{\boldsymbol{\theta}} \log p(x',h'|\hat{\boldsymbol{\theta}}) \neq \nabla_{\boldsymbol{\mu}} \log p(x,h|\hat{\boldsymbol{\mu}})^\top Z(\hat{\boldsymbol{\mu}})^{-1} \nabla_{\boldsymbol{\mu}} \log p(x',h'|\hat{\boldsymbol{\mu}}),$$

the posterior probabilities are invariant,

$$p(h|x,\hat{\boldsymbol{\theta}}) = p(h|x,\hat{\boldsymbol{\mu}}).$$

Therefore *choosing the parametrization amounts to choosing the joint kernel.* You may be able to derive the joint kernel to fit your purpose by changing parametrization. However, in our opinion, this is an awkward and indirect way.

In the Fisher kernel scheme, you have to control two things (i.e., joint kernel and posterior probability) simultanously by the choice of a parametric model. In our opinion, there is no need to control them in such an unified manner, because this scheme is sometimes too restrictive.

[‡] When there is one-to-one correspondence between two parameter spaces around $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\mu}}$, the Fisher kernel is invariant to parametrization because of the Fisher information matrix (Jaakkola and Haussler, 1999). However it is not the case in general.

For example, when you would like to incorporate the second-order information into the Fisher kernel, you have to use second-order HMMs (Durbin *et al.*, 1998). Since the number of parameters of the second-order HMM is much larger, it would be difficult to learn the parameters reliably with a small sample set. This drawback is caused by the fact that the joint kernel is tied to the probabilistic model. In our approach, the joint kernel and the probabilistic model are completely separated, so you can utilize second order information with a first order HMM as in the second order marginalized count kernel.

RESULTS

In this section, we illustrate the performance of marginalized kernels in classification experiments using bacterial *gyrB* amino acid sequences. *gyrB*—gyrase subunit B—is a DNA topoisomerase (type II) which plays essential roles in fundamental mechanisms of living organisms such as DNA replication, transcription, recombination and repair etc. One more important feature of *gyrB* is its capability of being an evolutionary and taxonomic marker alternating popular 16S rRNA (Kasai *et al.*, 1998). Our dataset consists of 84 amino acid sequences of *gyrB* from five genera in *Actinobacteria* which are *Corynebacterium*, *Mycobacterium*, *Gordonia*, *Nocardia* and *Rhodococcus*, respectively (Kasai *et al.*, 2000). For brevity these genera will be called genus 1 to 5, respectively. The number of sequences in each genus is listed as 9, 32, 15, 14 and 14. The sequences are, by their nature, quite similar in terms of sequence similarity. Pairwise identity for each sequence is at least 62 and 99% at most. For computing distance matrix based on the sequence similarity, one can use the BLAST scores (Altschul *et al.*, 1990). However, since such scores cannot directly be converted into positive semidefinite kernels, kernel methods cannot be applied to them in principle.

In order to investigate how well the kernels reflect underlying genera, we performed two kinds of experiments—clustering and supervised classification. The following kernels are compared:

- CK1: Count kernel (5)
- CK2: Second-order count kernel (9)
- FK: Fisher kernel (16)
- MCK1: Marginalized count kernel (7)
- MCK2: Second-order marginalized count kernel (11)

As the first experiment, K-Means clustering is performed in feature spaces corresponding to kernels (see Müller *et al.* (2001) for details). The number of clusters is determined as five (i.e., the true number). In FK and MCKs, we used complete-connection HMMs with 3, 5 and 7 states. Note that FK is normalized by the sequence lengths. In training HMMs, all 84 sequences are used. One can also train HMMs in a classwise manner (Tsuda *et al.*, 2002). However, we did not do so because the number of sequences is not large enough. For evaluating clusters, we used the adjusted Rand index (ARI) (Yeung and Ruzzo, 2001). The advantage of this index is that you can compare two partitions whose number of clusters are different. The ARI becomes 1 if the partitions are completely correct. Also, the expectation of the ARI is 0 when partitions are randomly determined.

The kernel matrices by FK and MCKs are shown in Figure 5. Additionally, the ideal kernel is shown for reference, where $K(\boldsymbol{x}, \boldsymbol{x}')$ is 1 for any two sequences in the same genus, and -1 otherwise. Here, the number of HMM states is three in all cases. For fair visualization, each kernel matrix is normalized in the same manner: First, the kernel matrix is 'centralized' as $K_c := K - 1_n K - K 1_n + 1_n K 1_n$ where 1_n is the $n \times n$ matrix whose elements are all $1/n$. Here n denotes the number of sequences, i.e., $n = 84$ in this experiment. Then, K_c is normalized by the Frobenious norm as $K_c / \|K_c\|_F$. As seen in the figure, MCK2 is the best to recover the underlying structure. This result is quantitatively shown by ARI in Figure 6, where CK1 and CK2 correspond to the MCKs with only one HMM state. Notably the Fisher kernel was worse than MCK1, which shows that the joint kernel of the Fisher kernel (16) is not appropriate for this task.

In order to see how genera are separated by introducing the second order information and hidden variables, we performed the following supervised classification experiments as well. First, we pick up two genera out of three genera (3, 4, 5). Genera 1 and 2 were not used because they can be separated easily by all kernels. The sequences of two genera are randomly divided into 25% training and 75% testing samples. Kernels are compared due to the test error by the kernel Fisher discriminant analysis (KFDA) (Roth and Steinhage, 2000), which compares favorably with the SVM in many benchmarks. Note that the regularization parameter ϵ of KFDA (Roth and Steinhage, 2000) is determined such that the test error is minimized[§]. The test errors of five kernels are shown in Table 1. The second order kernels (i.e., CK2 and MCK2) were significantly better than the first order kernels. This result coincides with the common understanding that higher order information of protein sequences is essential for classification and structure prediction (e.g., Asai *et al.* (1993)). Comparing CK2 and MCK2, MCK2 always performed better, which indicates that incorporating hidden variables (i.e., context information) is meaningful at least in this task.

[§] For regularization parameter ϵ, 10 equally spaced points on the log scale are taken from $[10^{-4}, 10^{2}]$. Among these candidates, the optimal one is chosen.

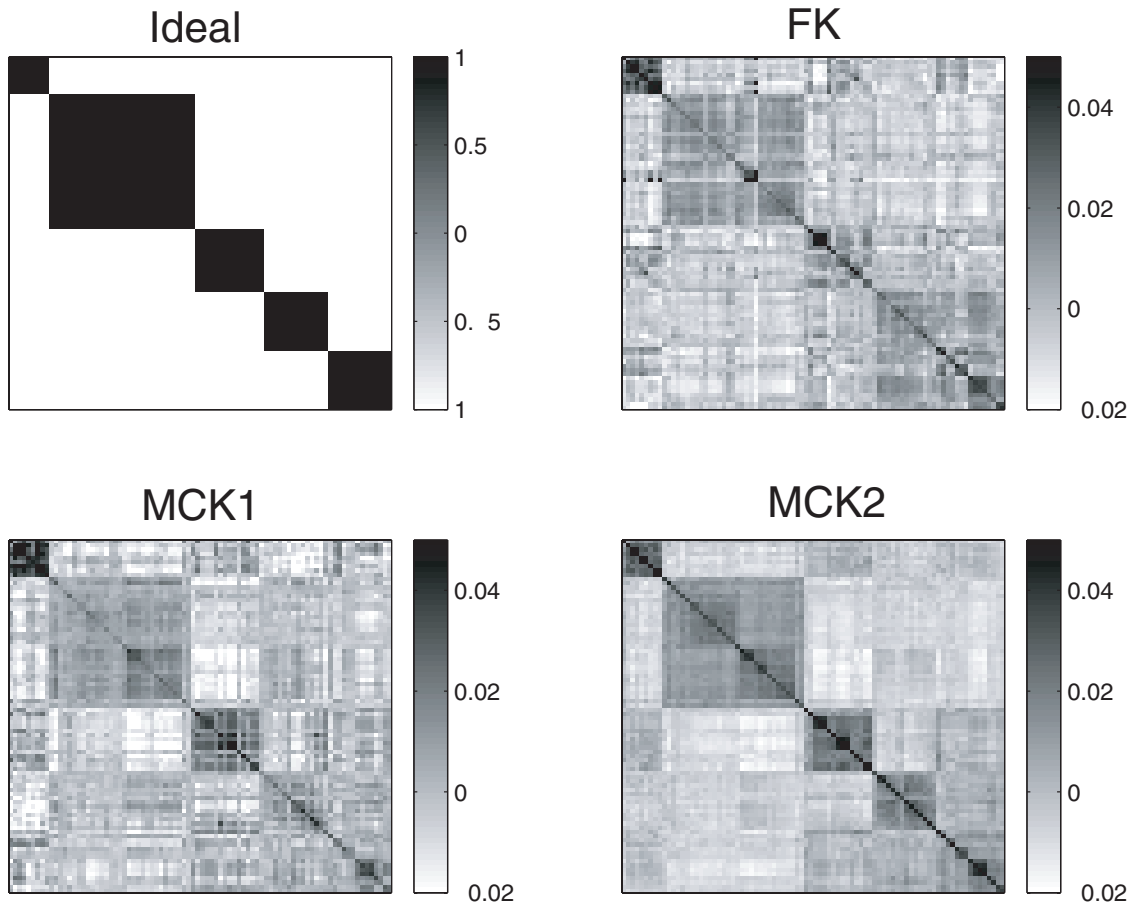

Fig. 5. (Upperleft) Ideal kernel matrix to illustrate the true clusters. (Upperright) Kernel matrix of the Fisher kernel. (Lowerleft) Kernel matrix of the first-order marginalized count kernel. (Lowerright) Kernel matrix of the second-order marginalized count kernel.

Table 1. Mean error rates (%) of supervised classification between two bacterial genera ([·] shows the standard deviation). The best result in each task is written in bold face

Genera	CK1	CK2	FK	MCK1	MCK2
3–4	24.5 [9.67]	9.10 [7.87]	10.4 [9.15]	12.8 [9.85]	**8.48** [7.76]
3–5	12.7 [8.93]	6.43 [7.76]	10.9 [10.1]	10.4 [8.17]	**5.71** [7.72]
4–5	25.6 [13.0]	13.5 [15.5]	23.1 [14.3]	20.0 [14.6]	**11.6** [14.6]

CONCLUSION

In this paper, we proposed marginalized kernels, which provide a new reasonable way to design a kernel from latent variable models. The Fisher kernel was described as a special case of marginalized kernels, which have added a new aspect to the Fisher kernel theory. Finally, we showed that marginalized count kernels perform well in protein classification experiments.

Our work provides a general framework from which diverse kernels are expected to be constructed. In future works, we would like to derive useful kernels as many as possible not only in bioinformatics but also in other areas in information technology.

ACKNOWLEDGEMENT

The authors gratefully acknowledge that the bacterial *gyrB* amino acid sequences are offered by courtesy of Identification and Classification of Bacteria (ICB)

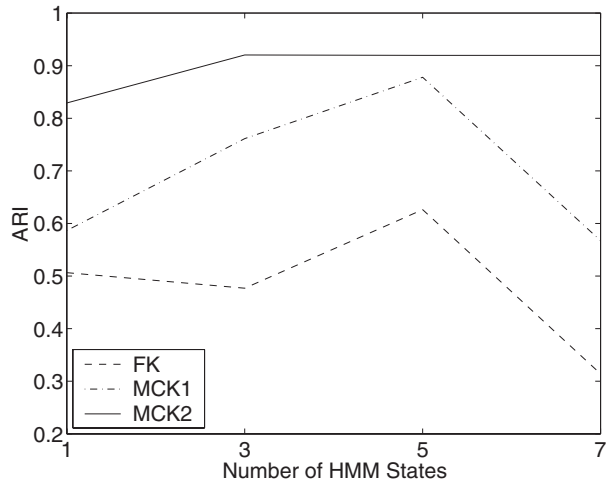

Fig. 6. Evaluation of kernels in clustering in terms of the adjusted Rand index (ARI). The x-axis corresponds to the number of states in HMM, from which the kernels are derived. The Fisher kernel (FK), the marginalized count kernels of first-order (MCK1) and second-order (MCK2) are compared. Note that the count kernels of first-order (CK1) and second-order (CK2) correspond to MCK1 and MCK2 at one HMM state, respectively.

database team (Watanabe *et al.*, 2001). The authors would like to thank S.-I.Amari, K.-R.Müller, G.Rätsch, M.Kawanabe, H.Nakahara and S.Sonnenburg for fruitful discussions.

REFERENCES

Altschul,S., Gish,W., Myers,E. and Lipman,D. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Asai,K., Hayamizu,S. and Handa,K. (1993) Prediction of protein secondary structure by the hidden Markov model. *CABIOS* (currently Bioinformatics), **9**, 141–146.

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

Frakes,W. and Baeza-Yates,R., (eds.) (1992) *Information Retrieval: Data Structure and Algorithms*. Prentice Hall.

Haussler,D. (1999) Convolution kernels on discrete structures. *Technical Report UCSC-CRL-99-10*. UC Santa Cruz.

Jaakkola,T., Diekhans,M. and Haussler,D. (2000) A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.*, **7**, 95–114.

Jaakkola,T. and Haussler,D. (1999) Exploiting generative models in discriminative classifiers. In Kearns,M., Solla,S. and Cohn,D. (eds), *Advances in Neural Information Processing Systems 11*. pp. 487–493.

Karchin,R., Karplus,K. and Haussler,D. (2002) Classifying G-protein coupled receptors with support vector machines. *Bioinformatics*, **18**, 147–159.

Kasai,H., Bairoch,A., Watanabe,K., Isono,K., Harayama,S., Gasteiger,E. and Yamamoto,S. (1998) Construction of the gyrB database for the identification and classification of bacteria. *Genome Informatics 1998*. Universal Academic Press, pp. 13–21.

Kasai,H., Ezaki,T. and Harayama,S. (2000) Differentiation of phylogenetically related slowly growing mycrobacteria by their gyrB sequences. *J. Clin. Microbiol.*, **38**, 301–308.

Müller,K.-R., Mika,S., Rätsch,G., Tsuda,K. and Schölkopf,B. (2001) An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Networks*, **12**, 181–201.

Pavlidis,P., Furey,T., Liberto,M., Haussler,D. and Grundy,W. (2001) Promotor region-based classification of genes. In *Proceedings PSB 2001*. pp. 151–163.

Rabiner,L. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, **77**, 257–285.

Roth,V. and Steinhage,V. (2000) Nonlinear discriminant analysis using kernel functions. In Solla,S., Leen,T. and Müller,K.-R. (eds), *Advances in Neural Information Processing Systems 12*. MIT Press, pp. 568–574.

Tipping,M. (1999) Deriving cluster analytic distance functions from gaussian mixture models. In Willshaw,D. and Murray,A. (eds), *Proceedings of ICANN'99*. IEE Press, pp. 815–820.

Tsuda,K., Kawanabe,M., Rätsch,G., Sonnenburg,S. and Müller,K.-R. (2002) A new discriminative kernel from probabilistic models. *Neural Comput.*, In press.

Watanabe,K., Nelson,J., Harayama,S. and Kasai,H. (2001) ICB database: the gyrB database for identification and classification of bacteria. *Nucleic Acids Res.*, **29**, 344–345.

Yeung,K. and Ruzzo,W. (2001) Principal component analysis for clustering gene expression data. *Bioinformatics*, **17**, 763–774.