1 Protein Classification via Kernel Matrix Completion

Taishin Kin
Tsuyoshi Kato
Koji Tsuda

The three-dimensional structure of a protein provides crucial information for predicting its function. However, as it is still a far more difficult and costly task to measure 3D coordinates of atoms in a protein than to sequence its amino acid composition, often we do not know the 3D structures of all the proteins at hand. Let us consider a kernel matrix that consists of kernel values representing protein similarities in terms of their 3D structures where some of the entries are *missing* because structural information about some proteins are not available whereas their amino acid sequences are readily available. This chapter proposes to estimate the missing entries by means of another kernel matrix derived from amino acid sequences. Basically, a parametric model is created from the sequence kernel matrix, and the missing entries of the structure's kernel matrix are estimated by fitting this model to existing entries. For model fitting, we adopt two algorithms: *single e-projection* and *em algorithm* based on the information geometry of positive definite matrices. For evaluating and demonstrating the performance of our method, we performed protein classification experiments by using support vector machines (SVMs). Our results show that these algorithms can effectively estimate the missing entries.

1.1 Introduction

Protein structure and sequence

One of the major issues in bioinformatics is the functional annotation of proteins. Proteins are molecules which play a variety of important roles (functions) in every living organism. The function of a protein is determined by its shape, which is usually called a 3D or tertiary structure, or more simply, structure. Therefore, protein structure is one of the major factors for investigating the mechanisms of proteins. The constant growth of a protein structure database such as the protein data bank (PDB) (Berman et al., 2000) might be proof of its importance. However, the protein structures are not always available because measuring 3D coordinates of every atom of a nano-scale molecule requires very expensive and intensive experiments. On the other hand, protein amino acid sequences are abundantly available, as shown by the rapid growth of databases such as Swiss-Prot (O'Donovan et al., 2002). We can find this fact with a very simple comparison: there are 129,463 sequences in Swiss-Prot and 19,375 structures in PDB as of this writing. Thus, ongoing research is endeavoring to achieve practical prediction of protein structures from their amino acid sequences (see figure 1.1 for a brief description of the relation between sequence and structure of a protein). Although varieties of prediction methods have been proposed, the prediction of an exact tertiary structure remains one of the most difficult problems because mechanisms behind the relation between structure and sequence are not fully clarified yet. Nevertheless it is highly probable that a sequence contains certain information to infer its structure.

In this chapter, we estimate the *relation* between two structures instead of estimating the structure itself, following the ideas in Tsuda et al. (2003). As in other chapters of this book, we represent n proteins by an $n \times n$ kernel matrix, which is a positive definite similarity matrix where the (i, j)th entry is the similarity between the ith protein and the jth protein. When we do not have structures for all proteins, we have to leave the entries of the kernel matrix for unavailable structures as missing. Obviously, due to missing entries, kernel learning algorithms such as support vector machines cannot be applied. Our aim is to complete the missing entries so that the learning algorithms can work on the completed kernel matrix. Basically we create a parametric model from another kernel matrix derived from sequences, and fit the model to the existing entries to complete the missing ones. Our algorithm is derived from a mathematical theory known as *information geometry* (Amari, 1995). Finally, we show promising results in protein classification experiments.

This chapter is organized as follows: Section 1.2 describes some definitions used in our algorithm. Section 1.3 introduces the information geometry to the space of positive definite matrices. Based on the geometric concepts, two algorithms for matrix approximation are presented in section 1.4. Then the protein structure classification experiment is described in section 1.5. We present our conclusions in section 1.6.

1.2 Kernel Matrix Completion

Kernel matrices for structure and sequence

Let us consider a kernel function as the similarity measure between two proteins. There are two types of such functions: k_{st} for structure similarity and k_{sq} for sequence similarity. We define the two matrices as follows:

- Structure kernel matrix D: $[D]_{ij} = k_{st}(x_i, x_j), \qquad i, j = 1, \cdots, l$

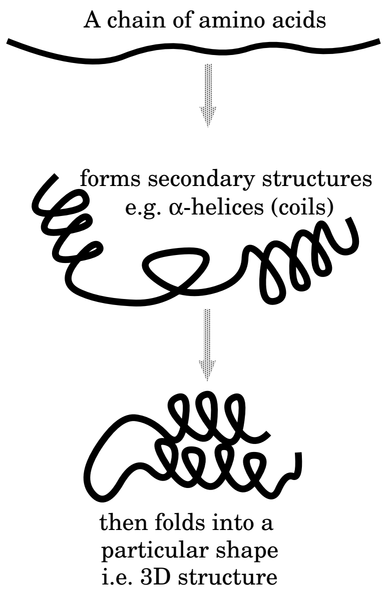

Figure 1.1 An overview of protein folding, which illustrates how an amino acid sequence folds into a protein molecule. A chain of amino acids is a product of translation of messenger RNA. The sequence forms secondary structures such as α helices (coils) or β sheets during its folding process. The secondary structure is the fundamental element of a protein structure. Finally, the sequence folds further into a specific shape which is the key property in determining the function of the protein. The shape of a protein is called its 3D structure.

- Sequence kernel matrix M: $[M]_{ij} = k_{sq}(x_i, x_j), \qquad i, j = 1, \cdots, l$

where $[M]_{ij}$ is the (i, j)th element of a matrix M and x_i denotes the ith protein. The two matrices are assumed to be positive definite such that they are compatible with kernel methods.

Now the task is to estimate the missing entries of the structure matrix using the sequence matrix. We create a *parametric model* of admissible matrices from the sequence matrix, and estimate missing entries by fitting the model to existing entries. According to our previous paper (Tsuda et al., 2003), we define the parametric model as all *spectral variants* of the sequence matrix which have the same eigenvectors but different eigenvalues (Cristianini et al., 2002).

In order to fit a parametric model, the distance between two matrices has to be determined. A common way is to define the Euclidean distance between matrices (e.g., the Frobenius norm) and make use of the Euclidean geometry. Recently Vert and Kanehisa (2003) tackled the incomplete matrix approximation problem by means of kernel canonical correlation analysis (CCA). Also Cristianini et al. (2002) proposed a similarity measure called "alignment," which is basically the cosine between two matrices. In contrast to these methods, which are based on Euclidean geometry, this chapter follows an alternative way: we define the Kullback-

Kullback-Leibler divergence

Leibler (KL) divergence between two kernel matrices and make use of Riemannian information geometry (Ohara et al., 1996). The KL divergence is derived by relating a kernel matrix to a covariance matrix of Gaussian distribution. The primal advantage is that the KL divergence allows us to use the *em* algorithm (Amari, 1995) to approximate an incomplete kernel matrix. The e and m steps are formulated as convex programming problems; moreover, they can be solved analytically when spectral variants are used as a parametric model.

1.3 Information Geometry of Positive Definite Matrices

In this section, we introduce the information geometry of the space of positive definite matrices. Only necessary parts of the theory are presented here, so the reader is referred to Ohara et al. (1996) and Amari and Nagaoka (2000) for details.

Gaussian distribution

Let us define the set of all $d \times d$ positive definite matrices as $\mathcal{P}$. The first step is to relate a $d \times d$ positive definite matrix $P \in \mathcal{P}$ to the Gaussian distribution with mean 0 and covariance matrix P:

$$p(\mathbf{x}|P) = \frac{1}{(2\pi)^{d/2} \det P^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^\top P^{-1}\mathbf{x}\right). \qquad (1.1)$$

Exponential family

It is well known that the Gaussian distributions form an exponential family. An exponential family is a set of distributions that can be written in the following canonical form:

$$p(\mathbf{x}|\theta) = \exp(\theta^\top \mathbf{r}(\mathbf{x}) - \psi(\theta)),$$

where $\mathbf{r}(\mathbf{x})$ is the vector of sufficient statistics, $\theta \in \Re^\rho$ is called the natural parameter, and $\psi(\theta)$ is the normalization factor. When (1.1) is rewritten in the canonical form, we have the sufficient statistics as

$$\mathbf{r}(\mathbf{x}) = -\left(\frac{1}{2}x_1^2, \frac{1}{2}x_2^2, \ldots, \frac{1}{2}x_d^2, x_1x_2, x_2x_3, \ldots, x_{d-1}x_d\right)^\top,$$

and the natural parameter as

$$\theta = \left([P^{-1}]_{11}, [P^{-1}]_{22}, \ldots, [P^{-1}]_{dd}, [P^{-1}]_{12}, [P^{-1}]_{23}, \ldots, [P^{-1}]_{d-1,d}\right)^\top.$$

θ-coordinate system

From the viewpoint of information geometry, the natural parameter θ provides a coordinate system (Amari and Nagaoka, 2000) to specify a positive definite matrix P, which is called the θ-coordinate system (or the e-coordinate system). On the other hand, there is an alternative representation for the exponential family. Let us define the mean of $r_i(x)$ as η_i: For example, when $r_i(x) = x_s x_t$,

$$\eta_i = \int x_s x_t p(\mathbf{x}|\theta) d\mathbf{x} = P_{st}.$$

η-coordinate system

This new set of parameters η_i provides another coordinate system, called the η-coordinate system (or the m-coordinate system):

$$\eta = (P_{11}, P_{22}, \ldots, P_{dd}, P_{12}, P_{23}, \ldots, P_{d-1,d})^\top .$$

Let us consider the following curve $\theta(t)$ connecting two points θ_1 and θ_2 linearly in θ coordinates:

$$\theta(t) = t(\theta_2 - \theta_1) + \theta_1.$$

When written is the matrix form, this reads

$$P^{-1}(t) = t(P_2^{-1} - P_1^{-1}) + P_1^{-1}.$$

e-flat

This curve is regarded as a straight line from the exponential viewpoint and is called an exponential geodesic or e-geodesic. In particular, each coordinate curve $\theta_i = t$, $\theta_j = c_j \;\; (j \neq i)$ is an e-geodesic. When the e-geodesic between any two points in a manifold $\mathcal{S} \subseteq \mathcal{P}$ is included in $\mathcal{S}$, the manifold $\mathcal{S}$ is said to be e-flat. On the other hand, the mixture geodesic or m-geodesic is defined as

$$\eta(t) = t(\eta_2 - \eta_1) + \eta_1.$$

In the matrix form, this reads

$$P(t) = t(P_2 - P_1) + P_1.$$

m-flat

When the m-geodesic between any two points in $\mathcal{S}$ is included in $\mathcal{S}$, the manifold $\mathcal{S}$ is said to be m-flat.

In information geometry, the distance between probability distributions is defined as the KL divergence (Amari and Nagaoka, 2000):

$$KL(p, q) = \int p(x) \log \frac{p(x)}{q(x)} dx.$$

By relating a positive definite matrix to the covariance matrix of Gaussian (1.1), we have KL divergence for two matrices P, Q:

$$KL(P, Q) = tr(Q^{-1}P) + \log\det Q - \log\det P - d. \tag{1.2}$$

With respect to a manifold $\mathcal{S} \subseteq \mathcal{P}$ and a point $P \in \mathcal{P}$, the projection from P to $\mathcal{S}$ is defined as the point in $\mathcal{S}$ closest to P. Since the KL divergence is asymmetric, there are two kinds of projection:

- e-projection: $Q^* = \operatorname{argmin}_{Q \in \mathcal{S}} KL(Q, P)$.
- m-projection: $Q^* = \operatorname{argmin}_{Q \in \mathcal{S}} KL(P, Q)$.

It is known that the m-projection to an e-flat submanifold is unique, and the e-projection to an m-flat manifold is unique (Amari and Nagaoka, 2000).

1.4 Approximating an Incomplete Kernel Matrix

In this section, we describe the *em* algorithm to approximate an incomplete kernel matrix (Tsuda et al., 2003). Let $x_1, \ldots, x_\ell \in \mathcal{X}$ be the set of samples of interest. In supervised learning cases, this set includes both training and test sets; thus we are considering the transductive setting (Vapnik, 1998). Let us assume that the data are available for the first n samples, and unavailable for the remaining $m := \ell - n$ samples. Denote by K_I an $n \times n$ kernel matrix, which is derived from the data for the first n samples. In our experiments, K_I is derived from the protein 3D structures. The incomplete kernel matrix is described as

Incomplete kernel matrix

$$D = \begin{pmatrix} K_I & D_{vh} \\ D_{vh}^\top & D_{hh} \end{pmatrix}, \tag{1.3}$$

where D_{vh} is an $n \times m$ matrix and D_{hh} is an $m \times m$ symmetric matrix. Since D has missing entries, it cannot be presented as a point in $\mathcal{P}$. Instead, all the possible kernel matrices form a manifold

Data manifold

$$\mathcal{D} = \{D \mid D_{vh} \in \Re^{n\times m},\;\; D_{hh} \in \Re^{m\times m},\;\; D_{hh} = D_{hh}^\top,\;\; D \succ 0\},$$

where $D \succ 0$ means that D is strictly positive definite. We call it the *data manifold* as in the conventional EM algorithm (Ikeda et al., 1999). It is easy to verify that $\mathcal{D}$ is an m-flat manifold; hence, the e-projection to $\mathcal{D}$ is unique.

Spectral variants

Next let us define the parametric model to approximate D. Here the model is derived as the spectral variants of K_B, which is an $\ell \times \ell$ auxiliary kernel matrix derived from another information source. Let us decompose K_B as

$$K_B = \sum_{i=1}^{\ell} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top,$$

where λ_i and $\mathbf{v}_i$ are the ith eigenvalue and eigenvector, respectively. Define

$$M_i = \mathbf{v}_i \mathbf{v}_i^\top, \tag{1.4}$$

then all the spectral variants are represented as

$$\mathcal{M} = \{M \mid M = \sum_{j=1}^{\ell} \beta_j M_j,\;\; \beta \in \Re^\ell,\;\; M \succ 0\}$$

Model manifold

We call it the model manifold (Ikeda et al., 1999). For notational simplicity, we

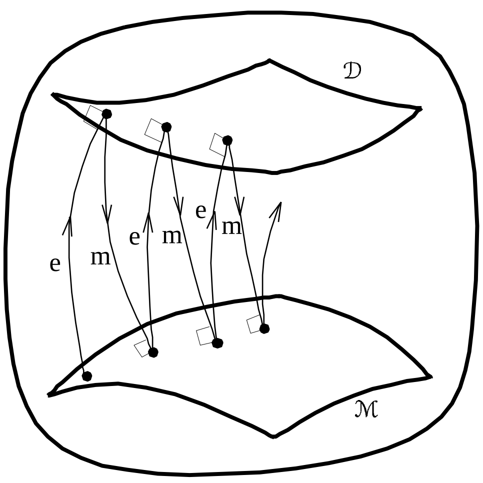

Figure 1.2 Information geometric picture of *em* algorithm. The data manifold $\mathcal{D}$ corresponds to the set of all completed matrices, whereas the model manifold $\mathcal{M}$ corresponds to the set of all spectral variants of a auxiliary matrix. The nearest points are found by gradually minimizing the KL divergence by repeating e- and m- projections.

choose a different parametrization of $\mathcal{M}$ [1]:

$$\mathcal{M} = \{M \mid M = (\sum_{j=1}^{\ell} b_j M_j)^{-1}, \ \ \mathbf{b} \in \Re^{\ell}, \ \ M \succ 0\}, \tag{1.5}$$

where $b_j = 1/\beta_j$. It is easily seen that the manifold $\mathcal{M}$ is e-flat and m-flat at the same time. Such a manifold is called dually flat.

Our approximation problem is formulated as finding the nearest points in two manifolds: Find $D \in \mathcal{D}$ and $M \in \mathcal{M}$ that minimize KL divergence $KL(D, M)$. In geometric terms, this problem is to find the nearest points between e-flat and m-flat manifolds. It is well known that such a problem can be solved by an alternating procedure called the *em* algorithm (Amari, 1995). The *em* algorithm gradually minimizes the KL divergence by repeating the e-step and m-step alternately (figure 1.2).

In the e-step, the following optimization problem is solved with a fixed M: Find $D \in \mathcal{D}$ that minimizes $KL(D, M)$ using (1.2). This is rewritten as follows: Find D_{vh} and D_{hh} that minimize

$$L_e = tr(DM^{-1}) - \log\det D, \tag{1.6}$$

1. $M^\top M^{-1} = (\sum \beta_j M_j)^\top (\sum 1/\beta_j M_j) = \sum \beta_j/\beta_j M_j^\top M_j = \sum M_j^\top M_j = I$.

subject to the constraint that $D \succ 0$. Notice that this constraint is not needed, because

$$\log \det D = \sum_{i=1}^{\ell} \log \mu_i,$$

where μ_i is the ith eigenvalue of D. Here $\log \det D$ is defined when all eigenvalues are positive. So, at the optimal solution, D is necessarily positive definite, because the KL divergence is infinite otherwise. As indicated by information geometry, this is a convex problem, which can readily be solved by any reasonable optimizer. Moreover the solution is obtained in a closed form: let us partition M^{-1} as

$$M^{-1} = \begin{pmatrix} S_{vv} & S_{vh} \\ S_{vh}^\top & S_{hh} \end{pmatrix}. \tag{1.7}$$

The solution to (1.6) is directly obtained by filling the missing entries in the matrix D with following forms:

$$D_{vh} = -K_I S_{vh} S_{hh}^{-1}, \tag{1.8}$$

$$D_{hh} = S_{hh}^{-1} + S_{hh}^{-1} S_{vh}^\top K_I S_{vh} S_{hh}^{-1}. \tag{1.9}$$

In the m-step, the following optimization problem is solved with D being fixed: Find $M \in \mathcal{M}$ that minimizes $KL(D, M)$. This is rewritten as follows: Find $\mathbf{b} \in \Re^\ell$ that minimizes

$$L_m = \sum_{j=1}^{\ell} b_j tr(M_j D) - \log \det(\sum_{j=1}^{\ell} b_j M_j) \tag{1.10}$$

subject to the constraint that $\sum_{j=1}^{\ell} b_j M_j \succ 0$. Notice that this constraint can be ignored as well. When $\{M_j\}_{j=1}^{\ell}$ are defined as (1.4), the closed-form solution of (1.10) is obtained as

$$b_i = 1/tr(M_i D), \qquad i = 1, \ldots, \ell. \tag{1.11}$$

For detailed derivation of (1.8), (1.9), and (1.11), the reader is referred to Tsuda et al. (2003).

1.5 Protein Structure Classification Experiment

We perform protein classification experiments by using our kernel completion algorithms. Here we use a fully curated database of protein structures called SCOP (Murzin et al., 1995), where proteins are classified into categories such that both structural and evolutionary relatedness are reflected. The categories are organized hierarchically as the following levels: class, fold, superfamily, and family. At the finest level, a family contains proteins with clear evolutionary relationship.

A set of proteins in a superfamily is considered to have the same evolutionary origin but it is not detectable at the level of sequences. Those in the same fold have major structural similarity, but evolutionary origins may be different. In this experiment, we used the following three superfamilies: NAD(P)-binding Rossmann-fold domains (6 families, 105 proteins) and (trans)glycosidases (4 families, 62 proteins). We classified the proteins in a superfamily into families by using our algorithms. In the case of (trans)glycosidases, 62 proteins are classified into 6 classes. Additionally, we used TIM beta/alpha-barrel protein fold (4 superfamilies, 90 proteins), where each protein is classified into one of superfamilies.

1.5.1 Kernels

We now describe how to obtain the $n \times n$ incomplete matrix K_I and the $l \times l$ auxiliary matrix K_B in this experiment.

The Incomplete Matrix K_I The incomplete matrix is obtained by structural similarities of proteins. The structural similarities are computed using results of MATRAS (Kawabata and Nishikawa, 2000) which is a software to measure a structural similarity of proteins. MATRAS yields several values to measure the similarity. We use values of *Rdis* which is the normalization (ranges from 0 to 100) of *ScDIS*, a similarity score for inter residual distances. We compute *Rdis* for every pair of proteins, which gives an $n \times n$ similarity matrix S_I. Since S_I is not positive definite, we modified S_I to be positive definite by cutting off non-positive eigenvectors (Roth et al., 2003). Let λ_i, ν_i denote the ith eigenvalue and eigenvector of S_I, respectively. Then, K_I is obtained as

$$K_I = \sum_{i:\lambda_i > 0} \lambda_i \nu_i \nu_i^\top .$$

The Base Matrix K_B As the base matrix K_B, we use the second-order marginalized count kernel (MCK) (Tsuda et al., 2002) in which the entry represents sequence similarities between two proteins. MCK is a general framework that includes Fisher kernel (Jaakkola and Haussler, 1999). Second-order MCK is shown to be more efficient than Fisher kernel for the protein structure classification problem (Tsuda et al., 2002). MCK exploits parameters of a latent variable model for its kernel computation. The latent variable model is used to represent implicit features of proteins such as secondary structures (i.e. α helices and β sheets). Here we use a hidden Markov model (HMM) which has a tri state full-connection network. Although such a network is not supported by any biological knowledge, it is supposed to capture implicit features of proteins. In order to build a kernel matrix derived from a homogeneous probability distribution, we train the HMM with all the protein sequences. Therefore, no class-specific information is explicitly given to the HMM.

1.5.2 Kernel Completion Methods

Other than the *em* algorithm presented in the previous section, we applied two simpler methods, namely *single e-projection* and *k-linear interpolation.* Let us describe details of them.

Single e-Projection The method *single e-projection* performs only one e-projection from K_B to the data manifold D; it does not do m-projection. So D can be obtained very fast since the method needs no iterations. However, we found that this method does not work well when the diagonal elements of D and M differ significantly. So we normalize K_I as follows:

$$K_I' := AK_IA \tag{1.12}$$

where A is the $n \times n$ diagonal matrix with entries

$$[A]_{ii} = \sqrt{\frac{[K_B]_{ii}}{[K_I]_{ii}}}. \tag{1.13}$$

The transformation in (1.12) adjusts the norms of feature vectors while the angles between feature vectors are kept the same. Notice that the *em* algorithm does not need normalization, because additional variables b_i can automatically absorb the difference of norms.

k-Linear Interpolation As an alternative method, we make use of the nearest sequences for completing missing entries. Suppose the structure is missing for the r-th protein ($n+1 \leq r \leq \ell$), and let us estimate the entries of the structure kernel d_{ri} for $i = 1, \dots, n$. First, we identify the k-neighbors of protein r in terms of the sequence kernel M, that is, sort the entries m_{rj} ($j = 1, \cdots, n$) and take the k largest ones. Let us denote the identified indices as $j_1, \cdots, j_k$. Then, the structure kernel d_{ri} is determined as

$$d_{ri} = \begin{cases} \frac{1}{k}\sum_{a=1}^{k} d_{j_a,i} & i = 1 \cdots n \\ \frac{1}{k^2}\sum_{a=1}^{k}\sum_{b=1}^{k} d_{j_a,i} d_{j_b,i} & i = n+1 \cdots \ell \end{cases}$$

This amounts to estimating the feature vector of protein r as the center of gravity of other feature vectors corresponding to $j_1, \cdots, j_k$. We call this method *k-linear interpolation.* In the following, we chose $k = 3$ as a result of preliminary cross-validation experiments.

1.5.3 Experimental Design

For evaluating the performance of kernel completion methods, we observe the accuracies of the SVM in the following experiments. Given a complete kernel matrix of structures, we remove randomly chosen rows/columns. The fraction of removed rows/columns is changed from 10% to 90% by a 10-point step. After completing the missing entries with one of the completion methods, the whole set of samples is randomly shuffled and divided into 50% training and 50% test sets. The accuracies of SVM are computed on these training and test sets. The regularization parameter C is determined through five fold cross-validations on the training set. When there are more than two classes, the classification problem is interpreted as several two-class problems by means of the one-against-all scheme, then several different classifiers are joined based on the largest-score-wins criteria.

Each experiment is iterated 100 times to average random effects. Classification accuracies solely using the sequence kernel K_B are computed to serve as references for comparing the algorithms discussed here.

1.5.4 Results

Several kernel matrices are illustrated in figure 1.3 so that the reader can grasp a visual intuition of how the kernel matrix is completed by the different algorithms. The SVM accuracies of the three completion algorithms on each task are shown in figure 1.4. The *em* algorithm performs the best when the fraction of missing entries is relatively small ($< 50\%$). In two data sets, namely the NAD(P) and TIM barrels, it significantly outperforms single e-projection. However, as the fraction of missing entries increases, the accuracy suddenly falls down. The reason is deemed as *overfitting*, because the increase in missing entries also increases the number of parameters. Single e-projection performs constantly well on all three data sets. It shows its best performances when the missing fraction is very large. However, when the missing fraction is small, it performs poorer than *em* algorithm. This is considered as the effect of *early stopping*. As often observed in neural network training, stopping the optimization before convergence sometimes avoids overfitting (Haykin, 1998). Finally, the k-linear interpolation performed quite well in two data sets [NAD(P) and glycosidases], although it is a simple heuristic. However, its accuracy was always worse than one of the two other principled methods.

Dotted flat lines in the figures show the accuracies of the sequence kernels. Thus the completion does not make sense if the accuracy is lower than this level. In all three experiments, the accuracies of completed matrices are better than that level until the 80% to 90% missing fraction. Obviously it is promising result because it implies that only partial information of structure can enhance the whole classification performance significantly.

1.6 Conclusion

In this chapter, we presented an algorithm for compensating an incomplete kernel matrix by utilizing a base kernel matrix of another information source. The algorithm is based on information geometry which provides metrics for the space of kernel matrices. Our algorithm can be a powerful tool in many situations where one information source is precise but expensive and the other source is noisy but cheap. Thanks to our algorithm, we can get a better kernel matrix by combining a cheaper complete matrix with a more precise incomplete matrix. One such situation is proteins, from which we can extract the sequences easily but the structures are costly. The experimental results on the protein data set reveal a remarkable performance of our method. Nevertheless, it is worth noting other parametric models for the model manifold $\mathcal{M}$. For example, since all eigenvalues are fixed in *single e-projection* while all eigenvalues are adaptive in *em algorithm*, the other model might be one that permits a part of eigenvalues to be adaptive.

Although we only discussed the case in which two information sources were available, it is interesting to consider a method which utilizes more information sources than two. For example, we may be able to make use of other information sources like class labels. In future works, we look forward to developing information geometric methods to combine multiple kernel matrices.

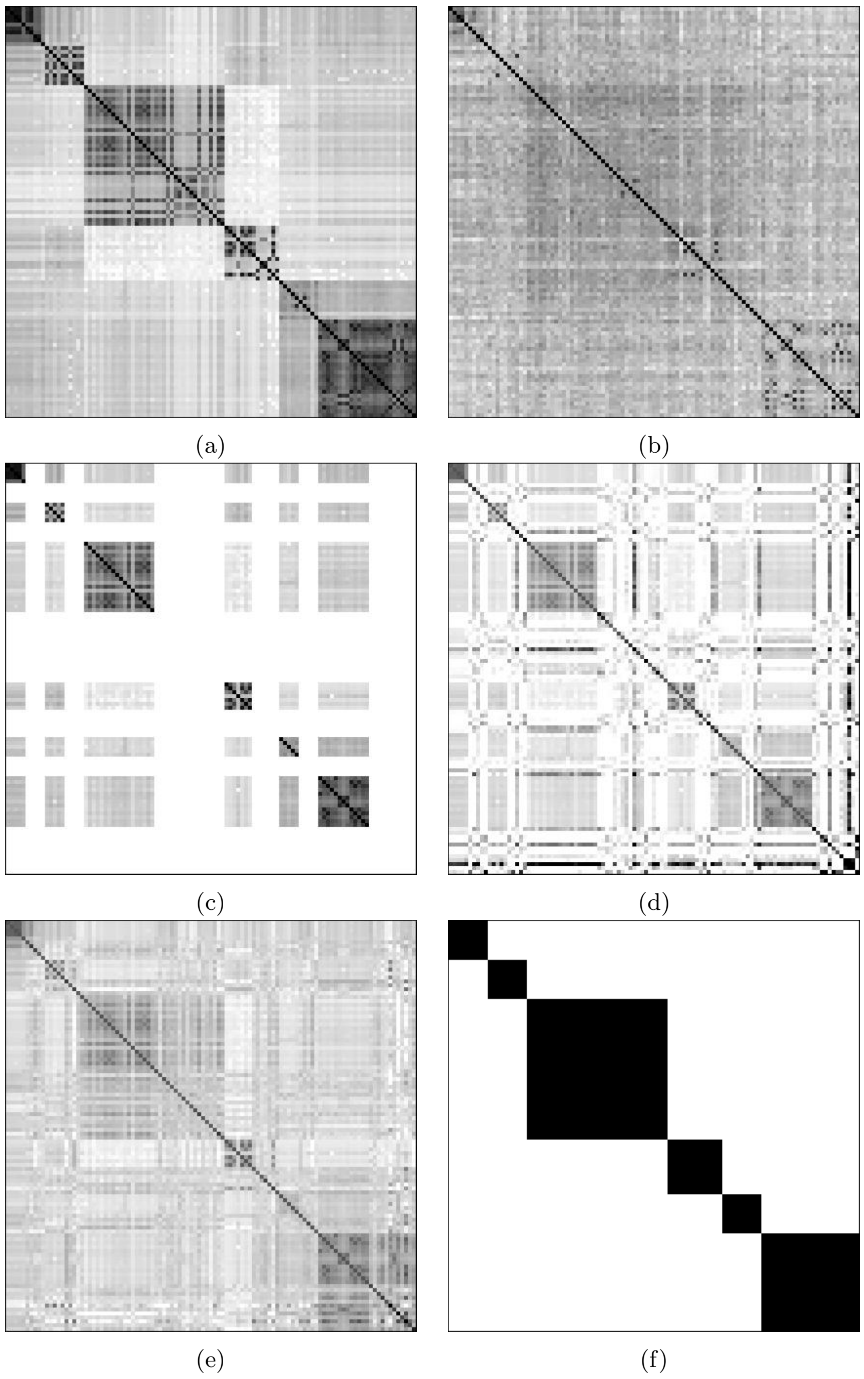

Figure 1.3 Raster of 62×62 kernel matrices for (trans)glycosidases. (a) Complete matrix of structure similarity (D_c). (b) Complete matrix of sequence similarity (M). (c) D with 50% missing elements. (d) D where missing elements are filled by using *single e-projection*. (e) Another D filled by using *em algorithm*. (f) The ideal kernel matrix showing three classes of this protein superfamily.

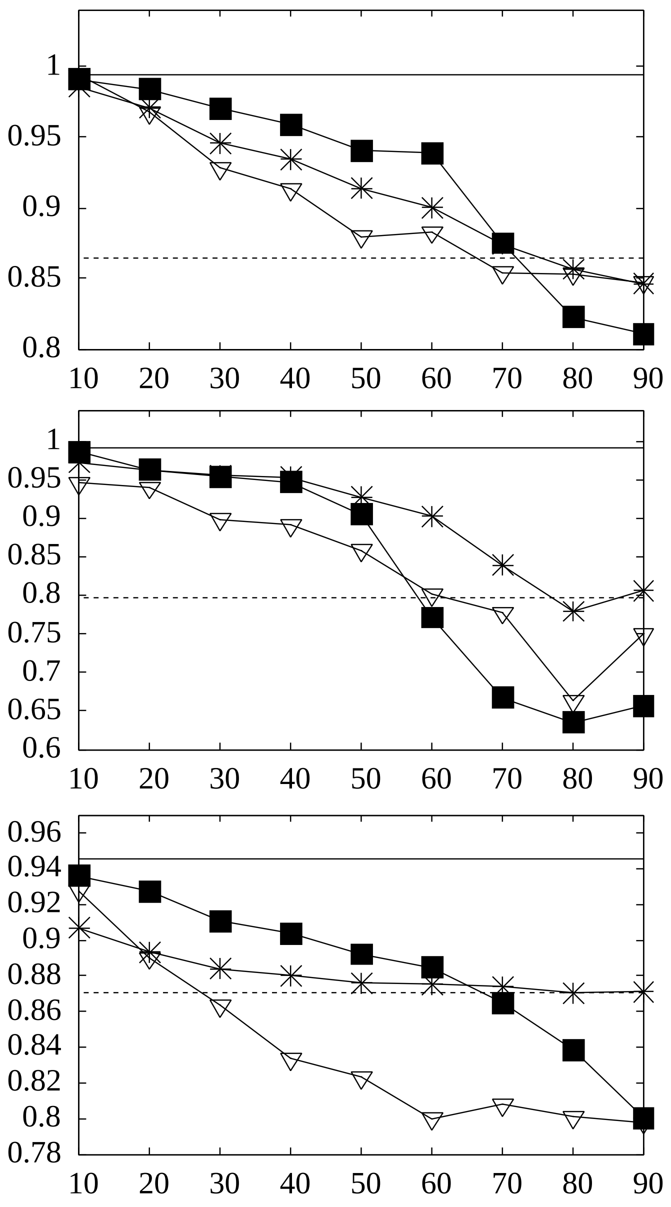

Figure 1.4 Classification accuracies for NAD(P)-binding Rossmann fold domains (top), (trans) glycosidases (middle), and TIM beta/alpha-barrels (bottom). The horizontal axis represents fraction of missing values. Three curves are shown in each figure: *square*, *em* algorithm; *star*, single *e*-projection; and *triangle*, 3-linear interpolation. The solid horizontal line indicates the accuracy without missing values. The dashed line indicates the accuracy using sequences only.

References

S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 9:1379–1408, 1995.

S.-I. Amari and H. Nagaoka. *Methods of Information Geometry.* Oxford, UK, Oxford University Press, 2000.

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 367–373. Cambridge, MA, MIT Press, 2002.

S. Haykin. *Neural Networks : A Comprehensive Foundation.* New York, Macmillan, second edition, 1998.

S. Ikeda, S. Amari, and H. Nakahara. Convergence of the wake-sleep algorithm. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 239–245. Cambridge, MA, MIT Press, 1999.

T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493. Cambridge, MA, MIT Press, 1999.

T. Kawabata and K. Nishikawa. Protein tertiary structure comparison using the Markov transition model of evolution. *Proteins*, 41:108–122, 2000.

A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

C. O'Donovan, M.J. Martin, A. Gattiker, E. Gasteiger, A. Bairoch, and R. Apweiler. High-quality protein knowledge resource: Swiss-Prot and TrEMBL. *Briefings in Bioinformatics*, 3:275–284, 2002.

A. Ohara, N. Suda, and S. Amari. Dualistic differential geometry of positive definite matrices and its applications to related problems. *Linear Algebra and Its Applications*, pages 31–53, 1996.

V. Roth, J. Laub, J.M. Buhmann, and K.-R. Müller. Going metric: Denoising pairwise data. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 817–824. Cambridge, MA, MIT Press, 2003.

K. Tsuda, S. Akaho, and K. Asai. The em algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, May 2003.

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18:S268–S275, 2002.

V. N. Vapnik. *Statistical Learning Theory.* Adaptive and Learning Systems for Signal Processing, Communications, and Control. New York, Wiley, 1998.

J.-P. Vert and M. Kanehisa. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1425–1432. Cambridge, MA, MIT Press, 2003.