1 Kernel-based Integration of Genomic Data using Semidefinite Programming

Gert R. G. Lanckriet
Department of Electrical Engineering and Computer Science
University of California Berkeley
Berkeley, CA, 94720
USA
gert@cs.berkeley.edu

Nello Cristianini
Department of Statistics
University of California Davis
Davis, CA, 95616
USA
nello@support-vector.net

Michael I. Jordan
Department of Statistics, Division of Computer Science
University of California Berkeley
Berkeley, CA, 94720
USA
jordan@cs.berkeley.edu

William Stafford Noble
Department of Genome Sciences
University of Washington
Seattle, WA, 98195
USA
noble@gs.washington.edu

An important challenge in bioinformatics is to leverage different descriptions of the same data set, each capturing different aspects of the data. Many such sources of information [about genes and proteins] are now available, such as sequence, expression, protein and regulation information. More data types are going to be available in the near future, such as array-based fitness profiles and protein-protein interaction data from mass spectrometry. Recent work in bioinformatics – such as gene function prediction, prediction of protein structure and localization, and inference of regulatory and metabolic networks – could benefit significantly from an approach that treats in a unified way the different types of information, merging them into a single representation, rather than only using the description that is judged to be the most relevant at hand.

This chapter describes a computational framework for integrating and drawing inferences from a collection of genome-wide measurements. Each data set is represented via a kernel function, which defines generalized similarity relationships between pairs of entities, such as genes or proteins. The kernel representation is both flexible and efficient, and provides a principled framework in which many types of data can be represented, including vectors, strings, trees and graphs. Furthermore, kernel functions derived from different types of data can be combined in a straightforward fashion—recent advances in the theory of kernel methods have provided efficient algorithms to perform such combinations in an optimal way. These methods formulate the problem of optimal kernel combination as a convex optimization problem that can be solved with semidefinite programming techniques.

After introducing the semidefinite programming techniques, we will use them to investigate the problem of predicting membrane proteins in yeast as well as predicting yeast functional classifications based on different types of information, including amino acid sequences, hydropathy profiles, gene expression data, known protein-protein interactions and known protein complexes. We show that a support vector machine (SVM) trained from all of these data, using the combined kernel, performs significantly better than the same algorithm trained on any single type of data, and significantly better than previously-described approaches.

1.1 Introduction

Much research in computational biology involves drawing statistically sound inferences from collections of data. For example, the function of an unannotated protein sequence can be predicted based on an observed similarity between that protein sequence and the sequence of a protein of known function. Related methodologies involve inferring related functions of two proteins if they occur in fused form in some other organism, if they co-occur in multiple species, if their corresponding mRNAs share similar expression patterns, or if the proteins interact with one another.

It seems natural that, while all such data sets contain important pieces of information about each gene or protein, the comparison and fusion of these data should produce a much more sophisticated picture of the relations among proteins, and a more detailed representation of each protein. Especially the recent availability of multiple types of genome-wide data that provide biologists with complementary views of a single genome, highlights the need for machine learning algorithms that unify these views and exploit this fused representation. Combining information from different sources contributes to forming a complete picture of the relations between the different components of a genome, enhancing the total information about the problem at hand.

In yeast, for example, for a given gene we typically know the protein it encodes, that protein's similarity to other proteins, its hydrophobicity profile, the mRNA expression levels associated with the given gene under hundreds of experimental conditions, the occurrences of known or inferred transcription factor binding sites in the upstream region of that gene, the identities of many of the proteins that interact with the given gene's protein product or form a complex with it. Each of these distinct data types provides one view of the molecular machinery of the cell. In the near future, research in bioinformatics will focus more and more heavily on methods of data fusion.

One problem with this approach, however, is that genomic data come in a wide variety of data formats: expression data are expressed as vectors or time series; protein sequence data as strings from a 20-symbol alphabet; gene sequences are strings from a different (4-symbol) alphabet; protein-protein interactions are best expressed as graphs, and so on.

This chapter presents a computational and statistical framework for integrating heterogeneous descriptions of the same set of genes, proteins or other entities. The approach relies on the use of kernel-based statistical learning methods that have already proven to be very useful tools in bioinformatics (Jaakkola et al., 1999; Brown et al., 2000; Furey et al., 2000; Zien et al., 2000). These methods represent the data by means of a kernel function, which defines similarities between pairs of genes, proteins, etc. Such similarities can be quite complex relations, implicitly capturing aspects of the underlying biological machinery. One reason for the success of kernel methods is that the kernel function takes relationships that are implicit in the data and makes them explicit, so that it is easier to detect patterns. Each kernel function thus extracts a specific type of information from a given data set, thereby providing a partial description or view of the data. The goal of this chapter is to find a kernel that best represents all of the information available for a given statistical learning task. Given many partial descriptions of the data, we solve the mathematical problem of combining them using a convex optimization method known as semidefinite programming (SDP) (Boyd et al., 1994; Nesterov and Nemirovsky, 1994; Vandenberghe and Boyd, 1996). This SDP-based approach (Lanckriet et al., 2002) yields a general methodology for combining many partial descriptions of data that is statistically sound, as well as computationally efficient and robust.

In order to demonstrate the feasibility of these methods, we describe two problems: identifying membrane proteins in yeast and predicting the function of yeast proteins. The first problem, identifying membrane proteins in yeast, is relevant considered that integral plasma membrane proteins serve several important communicative functions between the inside and the outside of the cell (Alberts et al., 1998). Classifying a protein as a membrane protein or not based on protein sequence is non-trivial and has been the subject of much previous research. The second problem, predicting the function of yeast proteins, is possibly even more relevant and has also been studied before.

Both problems are typical statistical learning problems in which a single type of feature derived from the protein sequence cannot provide the full story. We demonstrate that incorporating knowledge derived from the amino acid sequences, protein complex data, hydropathy profiles, gene expression data and known protein-protein interactions significantly improves classification performance relative to previously described methods and relative to our method trained on any single type of data.

We begin by describing related work. Afterwards, the main ideas of the kernel approach to pattern analysis are explained and semidefinite programming techniques introduced as an advanced instance of convex optimization. After presenting the necessary ingredients, we describe how different kernels defined on different data can be integrated using semidefinite programming techniques to provide a unified description. Finally, we describe the two biological applications of membrane protein recognition and protein function prediction in yeast. We define a number of kernels that are designed to capture different features of protein sequences, expression data, complex data and protein-protein interactions. We present computational experiments that demonstrate the validity and power of the kernel approach to data fusion, that outperforms the same kernel approach on any single type of data, as well as previously-described approaches.

1.2 Related Work

Considerable work has been devoted to the problem of automatically integrating genomic datasets, leveraging the interactions and correlations between them to obtain more refined and higher-level information. Previous research in this field can be divided into three classes of methods.

The first class treats each data type independently. Inferences are made separately from each data type, and an inference is deemed correct if the various data agree. This type of analysis has been used to validate, for example, gene expression and protein-protein interaction data (Ge et al., 2001; Grigoriev, 2001; Mrowka et al., 2003), to validate protein-protein interactions predicted using five different methods (von Mering et al., 2002), and to infer protein function (Marcotte et al., 1999). A slightly more complex approach combines multiple data sets using intersections and unions of the overlapping sets of predictions (Jansen et al., 2002).

The second formalism to represent heterogeneous data is to extract binary relations between genes from each data source, and represent them as graphs. As an example, sequence similarity, protein-protein interaction, gene co-expression or closeness in a metabolic pathway can be used to define binary relations between genes. Several groups have attempted to compare the resulting gene graphs using graph algorithms (Nakaya et al., 2001; Tanay et al., 2002), in particular to extract clusters of genes that share similarities with respect to different sorts of data.

The third class of techniques uses statistical methods to combine heterogeneous data. For example, Holmes and Bruno use a joint likelihood model to combine gene expression and upstream sequence data for finding significant gene clusters (Holmes and Bruno, 2000). Similarly, Deng et al. (2003b) use a maximum likelihood method to predict protein-protein interactions and protein function from three types of data. Alternatively, protein localization can be predicted by converting each data source into a conditional probabilistic model and integrating via Bayesian calculus (Drawid and Gerstein, 2000). The general formalism of graphical models, which includes Bayesian networks and Markov random fields as special cases, provides a systematic methodology for building such integrated probabilistic models. As an instance of this methodology, Deng et al. (2003a) developed a Markov random field model to predict yeast protein function. They found that the use of different sources of information indeed improved prediction accuracy when compared to using only one type of data.

This chapter describes a fourth type of data fusion technique, also statistical, but of a more nonparametric and discriminative flavor. The method, described in detail below, consists of representing each type of data independently as a matrix of kernel similarity values. These kernel matrices are then combined to make overall predictions. An early example of this approach, based on fixed sums of kernel matrices, showed that combinations of kernels can yield improved gene classification performance in yeast, relative to learning from a single kernel matrix (Pavlidis et al., 2001). The current work takes this methodology further—we use a *weighted* linear combination of kernels, and demonstrate how to estimate the kernel weights from the data. This yields not only predictions that reflect contributions from multiple data sources, but also yields an indication of the relative importance of these sources.

The graphical model formalism, as exemplified by the Markov random field model of Deng et al. (2003a), has several advantages in the biological setting. In particular, prior knowledge can be readily incorporated into such models, with standard Bayesian inference algorithms available to combine such knowledge with data. Moreover, the models are flexible, accommodating a variety of data types and providing a modular approach to combining multiple data sources. Classical discriminative statistical approaches, on the other hand, can provide superior performance in simple situations, by focusing explicitly on the boundary between classes, but tend to be significantly less flexible and less able to incorporate prior knowledge. As we discuss in this chapter, however, recent developments in kernel methods have yielded a general class of discriminative methods that readily accommodate non-standard data types (such as strings, trees and graphs), allow prior knowledge to be brought to bear, and provide general machinery for combining multiple data sources.

1.3 Kernel Methods

Kernel methods work by embedding data items (genes, proteins, etc.) into a vector space $\mathcal{F}$, called a *feature space*, and searching for linear relations in such a space. This embedding is defined implicitly, by specifying an inner product for the feature space via a positive semidefinite *kernel function*: $k(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$, where $\Phi(\mathbf{x}_1)$ and $\Phi(\mathbf{x}_2)$ are the embeddings of data items $\mathbf{x}_1$ and $\mathbf{x}_2$. Note that if all we require in order to find those linear relations are inner products, then we do not need to have an explicit representation of the mapping Φ, nor do we even need to know the nature of the feature space. It suffices to be able to evaluate the kernel function, which is often much easier than computing the coordinates of the points explicitly. Evaluating the kernel on all pairs of data items yields a symmetric, positive semidefinite matrix K known as the *kernel matrix*, which can be regarded as a matrix of generalized similarity measures among the data points.

The kernel-based binary classification algorithm that we will describe in this chapter, the *1-norm soft margin support vector machine* (SVM) (Boser et al., 1992; Schölkopf and Smola, 2002), forms a linear discriminant boundary in feature space $\mathcal{F}$, $f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$, where $\mathbf{w} \in \mathcal{F}$ and $b \in \mathbb{R}$. Given a labelled sample $S_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{w}$ and b are optimized to maximize the distance ("margin") between the positive and negative class, allowing misclassifications (therefore "soft margin"):

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i \qquad (1.1)\\ \text{subject to} \quad & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

where C is a regularization parameter, trading off error against margin. By considering the corresponding dual problem of (1.1), one can prove (see, e.g., Schölkopf and Smola, 2002) that the weight vector can be expressed as $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i)$, where the support values α_i are solutions of the following dual *quadratic program* (QP):

$$\max_{\boldsymbol{\alpha}} \quad 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y}) K \mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \ : \ C \geq \boldsymbol{\alpha} \geq 0, \ \ \boldsymbol{\alpha}^T\mathbf{y} = 0. \qquad (1.2)$$

The first stage of processing in a kernel method is thus to reduce the data by computing the kernel matrix. Given this matrix, and given the labels y_i, we can throw away the original data; the problem of fitting the SVM to data reduces to an optimization procedure that is based entirely on the kernel matrix and the labels. Different kernels correspond to different embeddings of the data and thus can be viewed as capturing different notions of similarity. For example, in a space derived from amino acid sequences, two genes that are close to one another will have protein products with very similar amino acid sequences. This amino acid space would be quite different from a space derived from microarray gene expression measurements, in which closeness would indicate similarity of the expression profiles of the genes. Finally, an unlabelled data item $\mathbf{x}_{new}$ can be classified by computing the linear function

$$f(\mathbf{x}_{new}) = \mathbf{w}^T \Phi(\mathbf{x}_{new}) + b = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_{new}) + b.$$

If $f(\mathbf{x}_{new})$ is positive, then we classify $\mathbf{x}_{new}$ as belonging to class $+1$; otherwise, we classify $\mathbf{x}_{new}$ as belonging to class -1.

1.4 Semidefinite Programming (SDP)

In this section we review the basic definition of semidefinite programming as well as some important concepts and key results. Details and proofs can be found in Boyd and Vandenberghe (2001).

Semidefinite programming (Nesterov and Nemirovsky, 1994; Vandenberghe and Boyd, 1996; Boyd and Vandenberghe, 2001) deals with the optimization of convex functions over the convex cone[1] of symmetric, positive semidefinite matrices

$$\mathcal{P} = \left\{ X \in \mathbb{R}^{p \times p} \mid X = X^T, X \succeq 0 \right\},$$

or affine subsets of this cone. As explained before, every positive semidefinite and symmetric matrix is a kernel matrix, and conversely, every kernel matrix is symmetric and positive semidefinite. Therefore $\mathcal{P}$ can be viewed as a search space for possible kernel matrices. This consideration leads to the key problem addressed in this chapter — we wish to specify a convex cost function that will enable us to learn the optimal kernel matrix within $\mathcal{P}$ using semidefinite programming.

1.4.1 Definition of Semidefinite Programming

A *linear matrix inequality,* abbreviated LMI, is a constraint of the form

$$F(\mathbf{u}) := F_0 + u_1 F_1 + \ldots + u_q F_q \preceq 0.$$

Here, $\mathbf{u}$ is the vector of decision variables, and $F_0, \ldots, F_q$ are given symmetric $p \times p$ matrices. The notation $F(\mathbf{u}) \preceq 0$ means that the symmetric matrix F is negative semidefinite. Note that such a constraint is in general a *nonlinear* constraint; the term "linear" in the name LMI merely emphasizes that F is affine in $\mathbf{u}$. Perhaps the most important feature of an LMI constraint is its convexity: the set of $\mathbf{u}$ that satisfy the LMI is a convex set.

An LMI constraint can be seen as an *infinite* set of scalar, affine constraints. Indeed, for a given $\mathbf{u}$, $F(\mathbf{u}) \preceq 0$ if and only if $\mathbf{z}^T F(\mathbf{u}) \mathbf{z} \leq 0$ for every $\mathbf{z}$; every constraint indexed by $\mathbf{z}$ is an affine inequality, in the ordinary sense, i.e., the left-hand side of the inequality is a scalar, composed of a linear term in $\mathbf{u}$ and a constant

1. $S \subseteq \mathbb{R}^d$ is a convex cone iff $\forall \mathbf{x}, \mathbf{y} \in S, \forall \lambda, \mu \geq 0\ : \lambda \mathbf{x} + \mu \mathbf{y} \in S$.

term. Alternatively, using a standard result from linear algebra, we may state the constraint $F(\mathbf{u}) \preceq 0$ as

$$\forall Z \in \mathcal{P} \ : \ \mathrm{trace}(F(\mathbf{u})Z) \leq 0. \tag{1.3}$$

This can be seen by writing down the spectral decomposition of Z and using that $\mathbf{z}^T F(\mathbf{u})\mathbf{z} \leq 0$ for every $\mathbf{z}$.

A semidefinite program (SDP) is an optimization problem with a linear objective, and linear matrix inequality and affine equality constraints.

Definition 1.1
A semidefinite program is a problem of the form

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{c}^T\mathbf{u} \\ \text{subject to} \quad & F^j(\mathbf{u}) = F_0^j + u_1 F_1^j + \ldots + u_q F_q^j \preceq 0, \quad j = 1, \ldots, L \\ & A\mathbf{u} = \mathbf{b}, \end{aligned} \tag{1.4}$$

where $\mathbf{u} \in \mathbb{R}^q$ is the vector of decision variables, $\mathbf{c} \in \mathbb{R}^q$ is the objective vector, and matrices $F_i^j = (F_i^j)^T \in \mathbb{R}^{p \times p}$ are given.

By convexity of their LMI constraints, SDPs are convex optimization problems. The usefulness of the SDP formalism stems from two important facts. First, despite the seemingly very specialized form of SDPs, they arise in a host of applications; second, there exist "interior-point" algorithms to globally solve SDPs that have extremely good theoretical and practical computational efficiency (Vandenberghe and Boyd, 1996).

One very useful tool to reduce a problem to an SDP is the so-called Schur complement lemma, which will be invoked later in this chapter.

Lemma 1.2
Schur complement lemma Consider the partitioned symmetric matrix

$$X = X^T = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix},$$

where A, C are square and symmetric. If $\det(A) \neq 0$, we define the Schur complement of A in X by the matrix $S = C - B^T A^{-1} B$. The Schur complement lemma states that if $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$.

To illustrate how this lemma can be used to cast a nonlinear convex optimization problem as an SDP, consider the following result:

Lemma 1.3
The quadratically constrained quadratic program (QCQP)

$$\begin{aligned} \min_{\mathbf{u}} \quad & f_0(\mathbf{u}) \\ \text{subject to} \quad & f_i(\mathbf{u}) \leq 0, \quad i = 1, \ldots, M, \end{aligned} \tag{1.5}$$

with $f_i(\mathbf{u}) \triangleq (A_i\mathbf{u}+\mathbf{b}_i)^T(A_i\mathbf{u}+\mathbf{b}_i) - \mathbf{c}_i^T\mathbf{u} - d_i$, is equivalent to the semidefinite programming problem:

$$\begin{aligned} \min_{\mathbf{u},t} \quad & t \\ \text{subject to} \quad & \begin{pmatrix} I & A_0\mathbf{u}+\mathbf{b_0} \\ (A_0\mathbf{u}+\mathbf{b_0})^T & \mathbf{c_0}^T\mathbf{u}+d_0+t \end{pmatrix} \succeq 0, \\ & \begin{pmatrix} I & A_i\mathbf{u}+\mathbf{b}_i \\ (A_i\mathbf{u}+\mathbf{b}_i)^T & \mathbf{c}_i^T\mathbf{u}+d_i \end{pmatrix} \succeq 0, \quad i=1,\ldots,M. \end{aligned} \tag{1.6}$$

This can be seen by rewriting the QCQP (1.5) as:

$$\begin{aligned} \min_{\mathbf{u},t} \quad & t \\ \text{subject to} \quad & t - f_0(\mathbf{u}) \geq 0, \\ & -f_i(\mathbf{u}) \geq 0, \quad i=1,\ldots,M. \end{aligned}$$

Note that for a fixed and feasible $\mathbf{u}$, $t = f_0\,(\mathbf{u})$ is the optimal solution. The convex quadratic inequality $t - f_0(\mathbf{u}) = (t+\mathbf{c_0}^T\mathbf{u}+d_0) - (A_0\mathbf{u}+\mathbf{b_0})^T I^{-1}(A_0\mathbf{u}+\mathbf{b_0}) \geq 0$ is now equivalent to the following LMI, using the Schur complement Lemma 1.2:

$$\begin{pmatrix} I & A_0\mathbf{u}+\mathbf{b_0} \\ (A_0\mathbf{u}+\mathbf{b_0})^T & \mathbf{c_0}^T\mathbf{u}+d_0+t \end{pmatrix} \succeq 0.$$

Similar steps for the other quadratic inequality constraints finally yield (1.6), an SDP in standard form (1.4), equivalent to (1.5). This shows that a QCQP can be cast as an SDP. Of course, in practice a QCQP should not be solved using general-purpose SDP solvers, since the particular structure of the problem at hand can be efficiently exploited. The above does show that QCQPs, and in particular, linear programming problems, belong to the SDP family.

1.4.2 Duality

An important principle in optimization—perhaps even the most important principle—is that of *duality*. To illustrate duality in the case of an SDP, we will first review basic concepts in duality theory and then show how they can be extended to semidefinite programming. In particular, duality will give insights into optimality conditions for the semidefinite program.

Consider an optimization problem with n variables and m scalar constraints

$$\begin{aligned} \min_{\mathbf{u}} \quad & f_0(\mathbf{u}) \\ \text{subject to} \quad & f_i(\mathbf{u}) \leq 0, \quad i=1,\ldots,m, \end{aligned} \tag{1.7}$$

where $\mathbf{u} \in \mathbb{R}^n$. In the context of duality, problem (1.7) is called the *primal problem*; we denote its optimal value p^*. For now, we do not assume convexity.

Definition 1.4
Lagrangian The *Lagrangian* $\mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R}$ corresponding to the minimization problem (1.7) is defined as

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = f_0(\mathbf{u}) + \lambda_1 f_1(\mathbf{u}) + \ldots + \lambda_m f_m(\mathbf{u}).$$

The $\lambda_i \in \mathbb{R},\ i = 1, \ldots, m$ are called *Lagrange multipliers* or *dual variables.*

One can now notice that

$$h(\mathbf{u}) = \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = \begin{cases} f_0(\mathbf{u}) & \text{if } f_i(\mathbf{u}) \leq 0,\ i = 1, \ldots, m \\ +\infty & \text{otherwise.} \end{cases} \tag{1.8}$$

So, the function $h(\mathbf{u})$ coincides with the objective $f_0(\mathbf{u})$ in regions where the constraints $f_i(\mathbf{u}) \leq 0,\ i = 1, \ldots, m$ are satisfied and $h(\mathbf{u}) = +\infty$ in infeasible regions. In other words, h acts as a "barrier" of the feasible set of the primal problem. Thus we can as well use $h(\mathbf{u})$ as objective function and rewrite the original primal problem (1.7) as an *unconstrained* optimization problem:

$$p^* = \min_{\mathbf{u}} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}). \tag{1.9}$$

The notion of weak duality amounts to exchanging the "min" and "max" operators in the above formulation, resulting in a lower bound on the optimal value of the primal problem. Strong duality refers to the case where this exchange can be done without altering the value of the result: the lower bound is actually equal to the optimal value p^*. While weak duality always hold, even if the primal problem (1.9) is not convex, strong duality may not hold. However, for a large class of generic convex problems, strong duality holds.

Lemma 1.5
Weak duality For all functions $f_0, f_1, \ldots, f_m$ in (1.7), not necessarily convex, we can exchange the max and the min and get a lower bound on p^*:

$$d^* = \max_{\boldsymbol{\lambda} \geq 0} \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) \leq \min_{\mathbf{u}} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = p^*.$$

The objective function of the maximization problem is now called the (Lagrange) dual function.

Definition 1.6
(Lagrange) dual function The *(Lagrange) dual function* $g : \mathbb{R}^m \to \mathbb{R}$ is defined as

$$\begin{aligned} g(\boldsymbol{\lambda}) &= \min_{\mathbf{u}}\ \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) \\ &= \min_{\mathbf{u}}\ f_0(\mathbf{u}) + \lambda_1 f_1(\mathbf{u}) + \ldots + \lambda_m f_m(\mathbf{u}). \end{aligned} \tag{1.10}$$

Furthermore $g(\boldsymbol{\lambda})$ is concave, even if the $f_i(\mathbf{u})$ are not convex.

The concavity can easily be seen by considering first that for a given $\mathbf{u}$, $\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda})$ is an affine function of $\boldsymbol{\lambda}$ and hence is a concave function. Since $g(\boldsymbol{\lambda})$ is the pointwise minimum of such concave functions, it is concave.

Definition 1.7
Lagrange dual problem The *Lagrange dual problem* is defined as

$$d^* = \max_{\boldsymbol{\lambda} \geq 0} g(\boldsymbol{\lambda}).$$

Since $g(\boldsymbol{\lambda})$ is concave, this will always be a convex optimization problem, even if the primal is not. By *weak duality*, we always have $d^* \leq p^*$, even for non-convex problems. The value $p^* - d^*$ is called the duality gap. For **convex** problems, we usually (although not always) have *strong duality* at the optimum, i.e.,

$$d^* = p^*,$$

which is also referred to as a *zero duality gap.* For convex problems, a sufficient condition for zero duality gap is provided by *Slater's condition*:

Lemma 1.8
Slater's condition If the primal problem (1.7) is convex and is **strictly feasible**, i.e., $\exists\ \mathbf{u}_0\ :\ f_i(\mathbf{u}_0) < 0,\ i = 1, \ldots, m$, then

$$p^* = d^*.$$

1.4.3 SDP Duality and Optimality Conditions

Consider for simplicity the case of an SDP with a single LMI constraint, and no affine equalities:

$$p^* = \min_{\mathbf{u}} \mathbf{c}^T\mathbf{u} \text{ subject to } F(\mathbf{u}) = F_0 + u_1F_1 + \ldots u_qF_q \preceq 0. \tag{1.11}$$

The general case of multiple LMI constraints and affine equalities can be handled by elimination of the latter and using block-diagonal matrices to represent the former as a single LMI.

The classical Lagrange duality theory outlined in the previous section does not directly apply here, since we are not dealing with finitely many constraints in scalar form; as noted earlier, the LMI constraint involves an infinite number of such constraints, of the form (1.3). One way to handle such constraints is to introduce a Lagrangian of the form

$$\mathcal{L}(\mathbf{u}, Z) = \mathbf{c}^T\mathbf{u} + \text{trace}(ZF(\mathbf{u})),$$

where the dual variable Z is now a symmetric matrix, of same size as $F(\mathbf{u})$. We can check that such a Lagrange function fulfills the same role assigned to the function defined in Definition 1.4 for the case with scalar constraints. Indeed, if we define

$h(\mathbf{u}) = \max_{Z \succeq 0} \mathcal{L}(\mathbf{u}, Z)$ then

$$h(\mathbf{u}) = \max_{Z \succeq 0} \mathcal{L}(\mathbf{u}, Z) = \begin{cases} \mathbf{c}^T \mathbf{u} & \text{if } F(\mathbf{u}) \preceq 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{1.12}$$

Thus, $h(\mathbf{u})$ is a barrier for the primal SDP (1.11), that is, it coincides with the objective of (1.11) on its feasible set, and is infinite otherwise. Notice that to the LMI constraint we now associate a multiplier *matrix*, which will be constrained to the positive semidefinite cone.

In the above, we made use of the fact that, for a given symmetric matrix F,

$$\Theta(F) := \sup_{Z \succeq 0} \ \mathrm{trace}(ZF)$$

is $+\infty$ if F has a positive eigenvalue, and zero if F is negative semidefinite. This property is obvious for diagonal matrices, since in that case the variable Z can be constrained to be diagonal without loss of generality. The general case follows from the fact that if F has the eigenvalue decomposition $F = U \Lambda U^T$, where Λ is a diagonal matrix containing the eigenvalues of F, and U is orthogonal, then $\mathrm{trace}(ZF) = \mathrm{trace}(Z' \Lambda)$, where $Z' = U^T Z U$ spans the positive semidefinite cone whenever Z does.

Using the above Lagrangian, one can cast the original problem (1.11) as an unconstrained optimization problem:

$$p^* = \min_{\mathbf{u}} \max_{Z \succeq 0} \mathcal{L}(\mathbf{u}, Z).$$

By weak duality, we obtain a lower bound on p^* by exchanging the min and max:

$$d^* = \max_{Z \succeq 0} \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, Z) \leq \min_{\mathbf{u}} \max_{Z \succeq 0} \mathcal{L}(\mathbf{u}, Z) = p^*.$$

The inner minimization problem is easily solved analytically, due to the special structure of the SDP. We obtain a closed form for the (Lagrange) dual function:

$$\begin{aligned} g(Z) = \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, Z) &= \min_{\mathbf{u}} \mathbf{c}^T \mathbf{u} + \mathrm{trace}(ZF_0) + \sum_{i=1}^{q} u_i \ \mathrm{trace}(ZF_i) \\ &= \begin{cases} \mathrm{trace}(ZF_0) & \text{if } c_i = -\mathrm{trace}(ZF_i), \ i = 1, \ldots, q \\ -\infty & \text{otherwise.} \end{cases} \end{aligned}$$

The dual problem can be explicitly stated as follows:

$$\begin{aligned} d^* &= \max_{Z \succeq 0} \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, Z) \\ &= \max_{Z} \mathrm{trace}(ZF_0) \ \text{ subject to } \ Z \succeq 0, \ c_i = -\mathrm{trace}(ZF_i), \ i = 1, \ldots, q. \end{aligned} \tag{1.13}$$

We observe that the above problem is an SDP, with a single LMI constraint and q affine equalities in the matrix dual variable Z.

While weak duality always holds, strong duality may not, even for SDPs. Not surprisingly, a Slater-type condition ensures strong duality. Precisely, if the primal

SDP (1.11) is strictly feasible, that is, there exist a $\mathbf{u}_0$ such that $F(\mathbf{u}_0) \prec 0$, then $p^* = d^*$. If, in addition, the dual problem is also strictly feasible, meaning that there exist $Z \succ 0$ such that $c_i = \text{trace}(ZF_i)$, $i = 1, \ldots, q$, then both primal and dual optimal values are attained by some optimal pair $(\mathbf{u}^*, Z^*)$. In that case, we can characterize such optimal pairs as follows. In view of the equality constraints of the dual problem, the duality gap can be expressed as

$$\begin{aligned} p^* - d^* &= \mathbf{c}^T\mathbf{u}^* - \text{trace}(Z^*F_0) \\ &= -\text{trace}(Z^*F(\mathbf{u}^*)). \end{aligned}$$

A zero duality gap is equivalent to $\text{trace}(Z^*F(\mathbf{u}^*)) = 0$, which in turn is equivalent to $Z^*F(\mathbf{u}^*) = O$, where O denotes the zero matrix, since the product of a positive semidefinite and a negative semidefinite matrix has zero trace if and only if it is zero.

To summarize, consider the SDP (1.11) and its Lagrange dual (1.13). If either problem is strictly feasible, then they share the same optimal value. If both problems are strictly feasible, then the optimal values of both problems are attained and coincide. In this case, a primal-dual pair $(\mathbf{u}^*, Z^*)$ is optimal if and only if

$$\begin{aligned} &F(\mathbf{u}^*) \preceq 0, \\ &Z^* \succeq 0, \\ &c_i = -\text{trace}(Z^*F_i), \quad i = 1, \ldots, q, \\ &Z^*F(\mathbf{u}^*) = O. \end{aligned}$$

The above conditions represent the expression of the general Karush-Kuhn-Tucker (KKT) conditions in the semidefinite programming setting. The first three sets of conditions express that $\mathbf{u}^*$ and Z^* are feasible for their respective problems; the last condition expresses a complementarity condition.

For a pair of strictly feasible primal-dual SDPs, solving the primal minimization problem is equivalent to maximizing the dual problem and both can thus be considered simultaneously. Algorithms indeed make use of this relationship and use the duality gap as a stopping criterion. A general-purpose program such as SeDuMi (Sturm, 1999) handles those problems efficiently. This code uses interior-point methods for SDP (Nesterov and Nemirovsky, 1994); these methods have a worst-case complexity of $O(q^2p^{2.5})$ for the general problem (1.11). In practice, problem structure can be exploited for great computational savings: e.g., when $F(\mathbf{u}) \in \mathbb{R}^{p\times p}$ consists of L diagonal blocks of size p_i, $i = 1, \ldots, L$, these methods have a worst-case complexity of $O(q^2(\sum_{i=1}^{L} p_i^2)p^{0.5})$ (Vandenberghe and Boyd, 1996).

1.5 Kernel Methods for Data Fusion

Given multiple related data sets (e.g., gene expression, protein sequence, and protein-protein interaction data), each kernel function produces, for the yeast genome, a square matrix in which each entry encodes a particular notion of similarity of one yeast protein to another. Implicitly, each matrix also defines an embedding of the proteins in a feature space. Thus, the kernel representation casts heterogeneous data—variable-length amino acid strings, real-valued gene expression data, a graph of protein-protein interactions—into the common format of kernel matrices.

The kernel formalism also allows these various matrices to be combined. Basic algebraic operations such as addition, multiplication and exponentiation preserve the key property of positive semidefiniteness, and thus allow a simple but powerful algebra of kernels (Berg et al., 1984). For example, given two kernels K_1 and K_2, inducing the embeddings $\Phi_1(\mathbf{x})$ and $\Phi_2(\mathbf{x})$, respectively, it is possible to define the kernel $K = K_1 + K_2$, inducing the embedding $\Phi(\mathbf{x}) = [\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x})]$. Of even greater interest, we can consider parameterized combinations of kernels. In this chapter, given a set of kernels $\mathcal{K} = \{K_1, K_2, \ldots, K_m\}$, we will form the linear combination

$$K = \sum_{i=1}^{m} \mu_i K_i. \tag{1.14}$$

As we have discussed, fitting an SVM to a single data source involves solving the quadratic program (1.2) based on the kernel matrix and the labels. It is possible to extend this optimization problem not only to find optimal discriminant boundaries but also to find optimal values of the coefficients μ_i in (1.14) for problems involving multiple kernels (Lanckriet et al., 2002). In the case of the 1-norm soft margin SVM, we want to minimize the same cost function (1.1), now with respect to both the discriminant boundary and the μ_i. Since the primal problem (1.1) is convex and strictly feasible, strong duality holds for (1.1) and (1.2) according to Lemma 1.8:

$$\begin{aligned} \omega_{S1}(K) &= \mathbf{w}_*^T \mathbf{w}_* + C \sum_{i=1}^{n} \xi_{i,*} \qquad (1.15) \\ &= \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T \mathbf{e} - \boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y}) K \mathrm{diag}(\mathbf{y}) \boldsymbol{\alpha} \;:\; C \geq \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T \mathbf{y} = 0. \end{aligned}$$

where $\mathbf{e}$ is a vector containing ones and $\mathbf{w}_*$ and $\xi_{i,*}$ the optimal values of the primal variables $\mathbf{w}$ and ξ_i. Training an SVM for a given kernel $K \succeq 0$ yields the minimal value (1.15) of (1.1) which is obviously a function of the particular choice of K, as is expressed explicitly in (1.15) as a dual problem. Let us now optimize this quantity with respect to the kernel matrix $K = \sum_{i=1}^{m} \mu_i K_i$, i.e., let us try to find the weights $\boldsymbol{\mu} \in \mathbb{R}^m$ for which the corresponding embedding shows minimal $\omega_{S1}(K)$, keeping the trace of K constant:

$$\min_{\boldsymbol{\mu} \in \mathbb{R}^m, K \succeq 0} \omega_{S1}(K) \qquad \text{s.t. } \mathrm{trace}(K) = c, \quad K = \sum_{i=1}^{m} \mu_i K_i, \tag{1.16}$$

where c is a constant. Note that the constraint $K \succeq 0$, emerges very naturally because the optimal kernel matrix must indeed come from the cone of positive semidefinite matrices. We first notice a fundamental property of the quantity

$\omega_{S1}(K)$, a property that is crucial for the remainder of this discussion:

Proposition 1.9
The quantity

$$\omega_{S1}(K) = \max_{\boldsymbol{\alpha}} \ 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y}) K \mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \ : \ C \geq \boldsymbol{\alpha} \geq 0, \ \ \boldsymbol{\alpha}^T\mathbf{y} = 0,$$

is convex in K.

This is easily seen by considering first that $2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y}) K \mathrm{diag}(\mathbf{y}))\boldsymbol{\alpha}$ is an affine function of (the entries of) K, and hence is a convex function as well. Secondly, we notice that $\omega_{S1}(K)$ is the pointwise maximum of such convex functions and is thus convex. This last statement is illustrated in a discrete case in figure 1.1. It shows how the pointwise maximum of two functions is convex. This can be extended for an infinite set of functions, e.g., indexed by $\boldsymbol{\alpha}$ in this case.

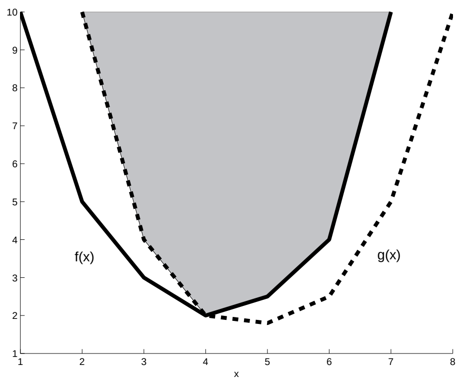

Figure 1.1 *Given two convex functions $f(x)$ and $g(x)$. Their pointwise maximum* $\max\{f(x), g(x)\}$ *will also be convex, as can easily be seen from the convexity of the shaded area (called the epigraph).*

Problem (1.16) is now a convex optimization problem. The following theorem shows that, for $K = \sum_{i=1}^m \mu_i K_i$, this problem can be cast as an SDP:

Theorem 1.10
Given a labelled sample $S_n = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ with corresponding set of labels $\mathbf{y} \in \mathbb{R}^n$ and a set of kernel matrices $\{K_i\}_{i=1}^m$, the kernel matrix $K = \sum_{i=1}^m \mu_i K_i$ that optimizes (1.16) can be found by solving the following convex

optimization problem which is a *semidefinite program (SDP)*:

$$\begin{aligned}
\min_{\boldsymbol{\mu},t,\lambda,\boldsymbol{\nu},\boldsymbol{\delta}} \quad & t \qquad\qquad (1.17)\\
\text{subject to} \quad & \operatorname{trace}\left(\sum_{i=1}^m \mu_i K_i\right) = c,\\
& \sum_{i=1}^m \mu_i K_i \succeq 0,\\
& \begin{pmatrix} \operatorname{diag}(\mathbf{y})(\sum_{i=1}^m \mu_i K_i)\operatorname{diag}(\mathbf{y}) & \mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y}\\ (\mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y})^T & t-2C\boldsymbol{\delta}^T\mathbf{e}\end{pmatrix} \succeq 0,\\
& \boldsymbol{\nu},\boldsymbol{\delta} \geq 0.
\end{aligned}$$

Proof After substitution of $\omega_{S1}(K)$ as defined in (1.15), (1.16) becomes:

$$\begin{aligned}
\min_{\boldsymbol{\mu}\in\mathbb{R}^m, K\succeq 0} \max_{\boldsymbol{\alpha}} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y})\boldsymbol{\alpha}\\
\text{subject to} \quad & C \geq \boldsymbol{\alpha} \geq 0,\ \ \boldsymbol{\alpha}^T\mathbf{y}=0,\ \ \operatorname{trace}(K)=c,\ \ K=\sum_{i=1}^m \mu_i K_i, \qquad (1.18)
\end{aligned}$$

with c a constant. Assume that $K \succ 0$, hence $\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y}) \succ 0$ (the following can be extended to the general semidefinite case). We note that $\omega_{S1}(K)$ is convex in K (because of Proposition 1.9) and thus in $\boldsymbol{\mu}$, since K is a linear function of $\boldsymbol{\mu}$. Given the convex constraints in (1.18), the optimization problem is thus certainly convex in $\boldsymbol{\mu}$. We write this as:

$$\begin{aligned}
\min_{\boldsymbol{\mu}\in\mathbb{R}^m, K\succeq 0, t} t \ : \ & t \geq \max_{\boldsymbol{\alpha}}\ 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y})\boldsymbol{\alpha},\\
& C \geq \boldsymbol{\alpha} \geq 0,\ \ \boldsymbol{\alpha}^T\mathbf{y}=0,\ \ \operatorname{trace}(K)=c,\ \ K=\sum_{i=1}^m \mu_i K_i. \qquad (1.19)
\end{aligned}$$

We will now express $t \geq \max_{\boldsymbol{\alpha}}\ 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y})\boldsymbol{\alpha}$ as an LMI using duality. In particular, we express the constraint using the dual minimization problem. This will allow us to drop the minimization and use the Schur complement lemma to obtain an LMI. We explain this now in more detail.

Define the Lagrangian of the maximization problem (1.2) by

$$\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\nu},\lambda,\boldsymbol{\delta}) = 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y})\boldsymbol{\alpha} + 2\boldsymbol{\nu}^T\boldsymbol{\alpha} + 2\lambda\mathbf{y}^T\boldsymbol{\alpha} + 2\boldsymbol{\delta}^T(C\mathbf{e}-\boldsymbol{\alpha}),$$

where $\lambda \in \mathbb{R}$ and $\boldsymbol{\nu},\boldsymbol{\delta} \in \mathbb{R}^n$. By duality, we have

$$\omega_{S1}(K) = \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\nu}\geq 0,\boldsymbol{\delta}\geq 0,\lambda} \mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\nu},\lambda,\boldsymbol{\delta}) = \min_{\boldsymbol{\nu}\geq 0,\boldsymbol{\delta}\geq 0,\lambda} \max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\nu},\lambda,\boldsymbol{\delta}),$$

where $\boldsymbol{\nu} \geq 0 \Leftrightarrow \nu_i \geq 0$ for $i = 1,\ldots,n$, similarly for $\boldsymbol{\delta}$. Since $\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y}) \succ 0$, at the optimum, we have

$$\boldsymbol{\alpha} = (\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y}))^{-1}(\mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y}),$$

and can form the dual problem

$$\omega_{S1}(K) = \min_{\boldsymbol{\nu},\, \boldsymbol{\delta},\, \lambda} \ (\mathbf{e}{+}\boldsymbol{\nu}{-}\boldsymbol{\delta}{+}\lambda\mathbf{y})^T\left(\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y})\right)^{-1}(\mathbf{e}{+}\boldsymbol{\nu}{-}\boldsymbol{\delta}{+}\lambda\mathbf{y}){+}2C\boldsymbol{\delta}^T\mathbf{e} \ : \ \boldsymbol{\nu} \geq 0,\ \boldsymbol{\delta} \geq 0.$$

We obtain that for any $t > 0$, the constraint $\omega_{S1}(K) \leq t$ is true if and only if there exist $\boldsymbol{\nu} \geq 0$, $\boldsymbol{\delta} > 0$ and λ such that

$$(\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T\left(\operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y})\right)^{-1}(\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y}) + 2C\boldsymbol{\delta}^T\mathbf{e} \leq t,$$

or, equivalently (using the Schur complement lemma), such that

$$\begin{pmatrix} \operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y}) & \mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T & t - 2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0$$

holds. Taking this into account, (1.19) can be expressed as:

$$\begin{aligned} \min_{\boldsymbol{\mu}\in\mathbb{R}^m, K, t, \lambda, \boldsymbol{\nu}, \boldsymbol{\delta}} \quad & t \\ \text{subject to} \quad & \operatorname{trace}(K) = c, \\ & K = \sum_{i=1}^m \mu_i K_i \succeq 0, \\ & \begin{pmatrix} \operatorname{diag}(\mathbf{y})K\operatorname{diag}(\mathbf{y}) & \mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T & t - 2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0, \\ & \boldsymbol{\delta} \geq 0, \end{aligned} \tag{1.20}$$

which yields (1.17) after substituting $K = \sum_{i=1}^m \mu_i K_i$ to eliminate K. Notice that $\boldsymbol{\nu} \geq 0 \Leftrightarrow \operatorname{diag}(\boldsymbol{\nu}) \succeq 0$, and thus an LMI; similarly for $\boldsymbol{\delta} \geq 0$. ■

Notice that the optimization problem (1.17) is an SDP in the standard form (1.4). This leads to a general method for learning the optimal combination of kernel matrices as a semidefinite programming problem, which can be solved via efficient interior-point algorithms (Vandenberghe and Boyd, 1996). Although efficient, these algorithms will still have a worst-case complexity $O(n^{4.5})$ in this particular case, according to the complexity results mentioned in Section 1.4.3.

In this discussion, the K_i are positive semidefinite by construction; thus $K \succeq 0$ is automatically satisfied if the weights μ_i are constrained to be non-negative. We will now point out an additional advantage of the restriction $\boldsymbol{\mu} \geq 0$: it will allow us to cast the SDP (1.17) as a *quadratically constrained quadratic program (QCQP)*, which has beneficial computational effects by lowering the efficiency of the computation to $O(n^3)$ in terms of the number of data points. Also, the constraint can result in improved numerical stability—it prevents the algorithm from using large weights with opposite sign that cancel. Finally, Lanckriet et al. (2002) shows that the constraint also yields better estimates of the generalization performance of these algorithms.

Solving the original learning problem (1.16) subject to the extra constraint $\boldsymbol{\mu} \geq 0$ yields:

$$\begin{aligned} \min_{\boldsymbol{\mu}\in\mathbb{R}^m, K}\ \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})K\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathrm{trace}(K) = c, \\ & K \succeq 0, \\ & K = \sum_{i=1}^m \mu_i K_i, \\ & \boldsymbol{\mu} \geq 0, \end{aligned}$$

when $\omega_{S1}(K)$ is expressed using (1.15). We can omit the second constraint, because this is implied by the last two constraints, since $K_i \succeq 0$. If we let $\mathrm{trace}(K_i) = r_i$, where $\mathbf{r} \in \mathbb{R}^m$, the problem reduces to:

$$\begin{aligned} \min_{\boldsymbol{\mu}}\ \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})\left(\sum_{i=1}^m \mu_i K_i\right)\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\mu}^T\mathbf{r} = c, \\ & \boldsymbol{\mu} \geq 0. \end{aligned}$$

We can write this as:

$$\begin{aligned} & \min_{\boldsymbol{\mu}\,:\,\boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c}\ \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})\left(\sum_{i=1}^m \mu_i K_i\right)\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \\ = & \min_{\boldsymbol{\mu}\,:\,\boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c}\ \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})K_i\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \\ = & \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0}\ \min_{\boldsymbol{\mu}\,:\,\boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})K_i\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha}, \end{aligned}$$

The interchange of the order of the minimization and the maximization is justified by standard results in convex optimization (see, e.g., Boyd and Vandenberghe, 2001) since the objective is convex in $\boldsymbol{\mu}$ (it is linear in $\boldsymbol{\mu}$) and concave in $\boldsymbol{\alpha}$, and because the minimization problem is strictly feasible in $\boldsymbol{\mu}$ and the maximization problem strictly feasible in $\boldsymbol{\alpha}$. We thus obtain:

$$\begin{aligned} & \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0}\ \min_{\boldsymbol{\mu}\,:\,\boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})K_i\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} \\ = & \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0}\left[2\boldsymbol{\alpha}^T\mathbf{e} - \max_{\boldsymbol{\mu}\,:\,\boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c}\left(\sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})K_i\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha}\right)\right] \\ = & \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0}\left[2\boldsymbol{\alpha}^T\mathbf{e} - \max_i\left(\frac{c}{r_i}\boldsymbol{\alpha}^T\mathrm{diag}(\mathbf{y})K_i\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha}\right)\right]. \end{aligned}$$

Finally, this can be reformulated as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - ct && (1.21)\\ \text{subject to} \quad & t \geq \frac{1}{r_i}\boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y}) K_i \mathrm{diag}(\mathbf{y})\boldsymbol{\alpha}, \quad i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T\mathbf{y} = 0, \\ & C \geq \boldsymbol{\alpha} \geq 0, \end{aligned}$$

or, when the K_i are normalized ($[K_i]_{jj} = 1$, $j = 1, \ldots, n$, such that $r_i = n$):

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - ct && (1.22)\\ \text{subject to} \quad & t \geq \frac{1}{n}\boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y}) K_i \mathrm{diag}(\mathbf{y})\boldsymbol{\alpha}, \quad i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T\mathbf{y} = 0, \\ & C \geq \boldsymbol{\alpha} \geq 0. \end{aligned}$$

This problem is a convex optimization problem, more precisely a *quadratically constrained quadratic program (QCQP)* (Boyd and Vandenberghe, 2001). Thus, the SDP (1.17) can be cast as a QCQP, which improves the efficiency of the computation to $O(n^3)$ in terms of the number of data points. The optimal weights μ_i, $i = 1, \ldots, m$, can be recovered from the primal-dual solution found by standard software such as SeDuMi (Sturm, 1999).

Thus, by solving a QCQP, we are able to find an adaptive combination of kernel matrices—and thus an adaptive combination of heterogeneous information sources—that solves our classification problem. The output of our procedure is a set of weights μ_i and a discriminant function based on these weights. We obtain a classification decision that merges information encoded in the various kernel matrices, and we obtain weights μ_i that reflect the relative importance of these information sources.

1.6 Two Biological Experiments

In this section, we illustrate the kernel-based approach for fusing heterogeneous genomic data using semidefinite programming for 2 biologically relevant problems: membrane protein prediction and protein function prediction in yeast. More details can be found in Lanckriet et al. (2003), for membrane protein recognition, and in Lanckriet et al. (2004), for the protein function classification.

1.6.1 Membrane Protein Classification

Membrane proteins are proteins that anchor in one of various membranes in the cell. Many membrane proteins serve important communicative functions. Generally, each membrane protein passes through the membrane several times. The transmembrane

Table 1.1 Kernel functions. The table lists the seven kernels used to compare proteins, the data on which they are defined, and the method for computing similarities. The final kernel, K_{RND}, is included as a control. All kernels matrices, along with the data from which they were generated, are available at `noble.gs.washington.edu/sdp-svm`.

Kernel	**Data**	**Similarity measure**
K_{SW}	protein sequences	Smith-Waterman
K_B	protein sequences	BLAST
K_{HMM}	protein sequences	Pfam HMM
K_{FFT}	hydropathy profile	FFT
K_{LI}	protein interactions	linear kernel
K_D	protein interactions	diffusion kernel
K_E	gene expression	radial basis kernel
K_{RND}	random numbers	radial basis kernel

regions of the amino acid sequence are typically hydrophobic, whereas the non-membrane portions are hydrophilic. This specific hydrophobicity profile of the protein allows it to anchor itself in the cell membrane.

Because the hydrophobicity profile of a membrane protein is critical to its function, this profile is better conserved in evolution than the specific amino acid sequence. Therefore, classical methods for determining whether a protein spans a membrane (Chen and Rost, 2002) depend upon a *hydropathy profile*, which plots the hydrophobicity of the amino acids along the protein (Engleman et al., 1986; Black and Mould, 1991; Hopp and Woods, 1981). In this subsection, we build on these classical methods by developing a kernel function that is based on the low-frequency alternation of hydrophobic and hydrophilic regions in membrane proteins. However, we also demonstrate that the hydropathy profile provides only partial evidence for transmembrane regions. Additional information is gleaned from sequence homology and from protein-protein interactions.

Note that, in general, membrane protein prediction consists of predicting the locations of multiple transmembrane regions within a single protein. In this example, however, for the purposes of demonstrating the SDP method, we focus on the binary prediction task of differentiating between membrane and non-membrane proteins.

1.6.1.1 Kernels for membrane protein prediction

For the task of membrane protein classification we experiment with seven kernel matrices derived from three different types of data: four from the primary protein sequence, two from protein-protein interaction data, and one from mRNA expression data. These are summarized in Table 1.1.

1.6.1.2 Protein sequence: Smith-Waterman, BLAST and Pfam HMM kernels

A homolog of a membrane protein is likely also to be located in the membrane. Therefore, we define three kernel matrices based upon standard homology detection methods. The first two sequence-based kernel matrices (K_{SW} and K_B) are generated using the BLAST (Altschul et al., 1990) and Smith-Waterman (SW) (Smith and Waterman, 1981) pairwise sequence comparison algorithms, as described previously (Liao and Noble, 2002). Because matrices of BLAST or Smith-Waterman scores are not necessarily positive semidefinite, we represent each protein as a vector of scores (BLAST and SW log E-values, respectively) against all other proteins. Defining the similarity between proteins as the inner product between the score vectors (the so-called empirical kernel map, Tsuda, 1999) leads to a valid kernel matrix, one for the BLAST score and one for the SW score. Note that including in the comparison set proteins with unknown subcellular locations allows the kernel to exploit this unlabelled data. The third kernel matrix (K_{HMM}) is a generalization of the previous pairwise comparison-based matrices in which the pairwise comparison scores are replaced by expectation values derived from hidden Markov models in the Pfam database (Sonnhammer et al., 1997). These similarity measures are not specific to the membrane protein classification task.

1.6.1.3 Protein sequence: FFT kernel

In contrast, the fourth sequence-based kernel matrix (K_{FFT}) directly incorporates information about hydrophobicity patterns, which are known to be useful in identifying membrane proteins. The kernel uses hydropathy profiles generated from the Kyte-Doolittle index (Kyte and Doolittle, 1982). This kernel compares the frequency content of the hydropathy profiles of the two proteins. After pre-filtering the hydropathy profiles, their Fourier transforms (describing the frequency content) are computed using an FFT algorithm. The frequency contents of different profiles are compared by applying a Gaussian kernel function, $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-||\mathbf{x}_1 - \mathbf{x}_2||^2/2\sigma)$ with width $\sigma = 10$, to the corresponding vectors of FFT values. This kernel detects periodicities in the hydropathy profile, a feature that is relevant to the identification of membrane proteins and complementary to the previous, homology-based kernels.

1.6.1.4 Protein interactions: linear and diffusion kernels

We expect information about protein-protein interactions to be informative in this context for two reasons. First, hydrophobic molecules or regions of molecules tend to interact with each other. Second, transmembrane proteins are often involved in signaling pathways, and therefore different membrane proteins are likely to interact with a similar class of molecules upstream and downstream in these pathways (e.g., hormones upstream or kinases downstream). The two protein interaction kernels are generated using medium- and high-confidence interactions from a database of known interactions (von Mering et al., 2002). These interactions can be represented as an interaction matrix, in which rows and columns correspond to proteins, and binary entries indicate whether the two proteins interact.

The first interaction kernel matrix (K_{LI}) is comprised of linear interactions, i.e., inner products of rows and columns from the centered, binary interaction matrix. The more similar the interaction pattern (corresponding to a row or column from the interaction matrix) is for a pair of proteins, the larger the inner product will be.

An alternative way to represent the same interaction data is to consider the proteins as nodes in a large graph. In this graph, two proteins are linked when they interact and otherwise not. Kondor and Lafferty (2002) propose a general method for establishing similarities between the nodes of a graph, based on a random walk on the graph. This method efficiently accounts for all possible paths connecting two nodes, and for the lengths of those paths. Nodes that are connected by shorter paths or by many paths are considered more similar. The resulting *diffusion kernel* generates the second interaction kernel matrix (K_D).

An appealing characteristic of the diffusion kernel is its ability, like the empirical kernel map, to exploit unlabelled data. In order to compute the diffusion kernel, a graph is constructed using all known protein-protein interactions, including interactions involving proteins whose subcellular locations are unknown. Therefore, the diffusion process includes interactions involving unlabelled proteins, even though the kernel matrix only contains entries for labelled proteins. This allows two labelled proteins to be considered close to one another if they both interact with an unlabelled protein.

1.6.1.5 Gene expression: radial basis kernel

Finally, we also include a kernel constructed entirely from microarray gene expression measurements. A collection of 441 distinct experiments was downloaded from the Stanford Microarray Database (`genome-www.stanford.edu/microarray`). This data provides us with a 441-element expression vector characterizing each gene. A Gaussian kernel matrix (K_E) is computed from these vectors by applying a Gaussian kernel function with width $\sigma = 100$ to each pair of 441-element vectors, characterizing a pair of genes. Note that we do not expect that gene expression will be particularly useful for the membrane classification task. We do not need to make this decision *a priori*, however; as explained in the following section, our method is able to provide an *a posteriori* measure of how useful a data source is relative to the other sources of data. We thus include the expression kernel in our experiments to test this aspect of the method.

1.6.1.6 Experimental design

In order to test our kernel-based approach in the setting of membrane protein classification, we use as a gold standard the annotations provided by the Munich

Information Center for Protein Sequences Comprehensive Yeast Genome Database (CYGD) (Mewes et al., 2000). The CYGD assigns subcellular locations to 2318 yeast proteins, of which 497 belong to various membrane protein classes. The remaining approximately 4000 yeast proteins have uncertain location and are therefore not used in these experiments.

The primary input to the classification algorithm is the collection of kernel matrices from Table 1.1. Using the SDP techniques described above, we find an optimal combination of the seven kernel matrices, and the resulting matrix is used to train an SVM classifier.

For comparison with the SDP/SVM learning algorithm, we consider several classical biological methods that are commonly used to determine whether a Kyte-Doolittle plot corresponds to a membrane protein, as well as a state-of-the-art technique using hidden Markov models (HMMs) to predict transmembrane helices in proteins (Krogh et al., 2001; Chen and Rost, 2002). The first method relies on the observation that the average hydrophobicity of membrane proteins tends to be higher than that of non-membrane proteins, because the transmembrane regions are more hydrophobic. We therefore define f_1 as the average hydrophobicity, normalized by the length of the protein. We will compare the classification performance of our statistical learning algorithm with this metric.

Clearly, however, f_1 is too simplistic. For example, protein regions that are not transmembrane only induce noise in f_1. Therefore, an alternative metric filters the hydrophobicity plot with a low-pass filter and then computes the number, the height and the width of those peaks above a certain threshold (Chen and Rost, 2002). The filter is intended to smooth out periodic effects. We implement two such filters, choosing values for the filter order and the threshold based on Chen and Rost (2002). In particular, we define f_2 as the area under the 7th-order low-pass filtered Kyte-Doolittle plot and above a threshold value 2, normalized by the length of the protein. Similarly, f_3 is the corresponding area using a 20th-order filter and a threshold of 1.6.

Finally, the Transmembrane HMM (TMHMM) web server (`www.cbs.dtu.dk/services/TMHMM`) is used to make predictions for each protein. In Krogh et al. (2001), transmembrane proteins are identified by TMHMM using three different metrics: the expected number of amino acids in transmembrane helices, the number of transmembrane helices predicted by the N-best algorithm, and the expected number of transmembrane helices. Only the first two of these metrics are provided in the TMHMM output. Accordingly, we produce two lists of proteins, ranked by the number of predicted transmembrane helices (T_{PH}) and by the expected number of residues in transmembrane helices (T_{ENR}).

Each algorithm's performance is measured by splitting the data into a training and test set in a ratio of 80/20. We report the receiver operating characteristic (ROC) score, which is the area under a curve that plots true positive rate as a function of false positive rate for differing classification thresholds (Hanley and McNeil, 1982; Gribskov and Robinson, 1996). The ROC score measures the overall quality of the ranking induced by the classifier, rather than the quality of a single

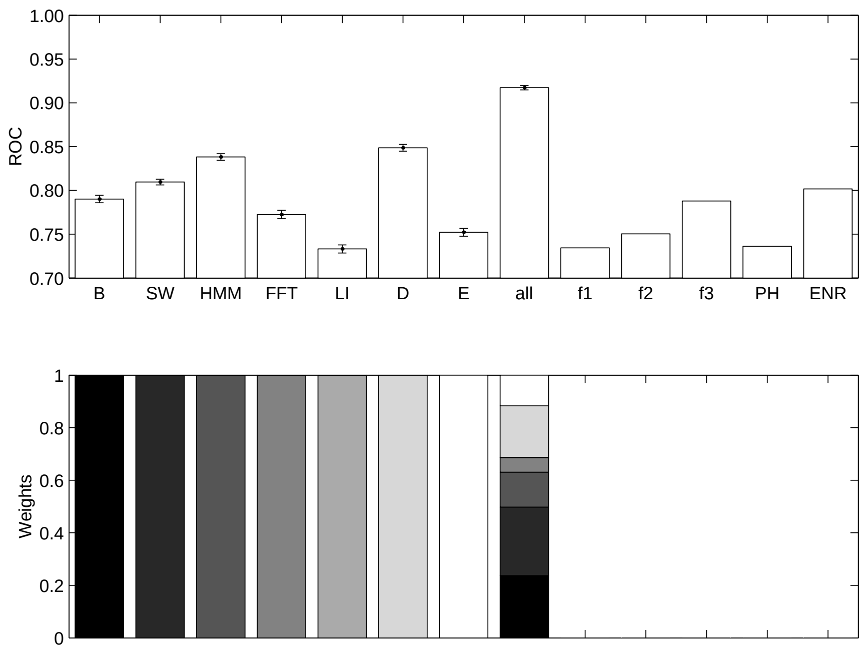

Figure 1.2 Combining data sets yields better classification performance. The height of each bar is proportional to the ROC score of the given membrane protein classification method. The bars labelled *B* to *E* and *all* correspond to SDP/SVM methods, the bars labelled *f* are hydropathy profile metrics, and the bars labelled *PH* and *ENR* refer to the TMHMM methods as defined in the text. Error bars indicate standard error across 30 random train/test splits. The heights of the grey level bars below each plot indicate the relative weight of the different kernel matrices in the optimal linear combination.

point in that ranking. An ROC score of 0.5 corresponds to random guessing, and an ROC score of 1.0 implies that the algorithm succeeded in putting all of the positive examples before all of the negatives. Each experiment is repeated 30 times with different random splits in order to estimate the variance of the performance values.

1.6.1.7 Results

We performed computational experiments which study the performance of the SDP/SVM approach as a function of the number of data sources, compare the performance of the method to classical biological methods and state-of-the-art techniques for membrane protein classification, and study the robustness of the method to the presence of noise.

The results from the first three experiments are summarized in Figure 1.2. The plot illustrates that SDP/SVM learns significantly better from the heterogeneous data than from any single data type. The mean ROC score using all seven kernel matrices (0.9174 ± 0.0025) is significantly higher than the best ROC score using only one matrix (0.8487 ± 0.0039 using the diffusion kernel). This improvement corresponds to a change in test set accuracy of 7.3%, from 81.3% to 88.6%.

As expected, the sequence-based kernels yield good individual performance. This is evident from the ROC scores. Furthermore, when all seven matrices are used at once, the SDP assigns relatively large weights to the sequence-based kernels. These weights are as follows: $\mu_B = 1.66$, $\mu_{SW} = 1.83$, $\mu_{HMM} = 0.93$, $\mu_{FFT} = 0.39$, $\mu_{LI} = 0.01$, $\mu_D = 1.37$ and $\mu_E = 0.82$ (note that for ease of interpretation, we scale the weights such that their sum is equal to the number m of kernel matrices). Thus, two of the three kernel matrices that receive weights larger than 1 are derived from the amino acid sequence. The Smith-Waterman kernel yields better results than the BLAST kernel, reflecting the fact that BLAST is a heuristic search procedure, whereas the Smith-Waterman algorithm guarantees finding the optimal local alignment of two sequences.

The results also show that the interaction-based diffusion kernel is more informative than the expression kernel. Not only has the diffusion kernel an individual ROC score which is significantly higher than the expression kernel, the SDP also assigns a weight of 1.37 to the diffusion kernel, whereas the expression kernel receives a weight of 0.82. Accordingly, removing the diffusion kernel reduces the ROC score from 0.9174 to 0.8984, whereas removing the expression kernel has a smaller effect, leading to a ROC score of 0.9033. Further description of the results obtained when various subsets of kernels are used is provided in Lanckriet et al. (2003).

Figure 1.2 also compares the membrane protein classification performance of the SDP/SVM method with that of previously described techniques. The results confirm that using learning in this context dramatically improves the results relative to the simple hydropathy profile approach. Also, the SDP/SVM improves upon the performance of the TMHMM approach, even when the SVM algorithm uses only the sequence data K_{SW} or K_{HMM} (ROC of 0.8096 ± 0.0033 or 0.8382 ± 0.0038 versus 0.8018, respectively).

While the SDP/SVM algorithm is a discriminative method that attempts to find a decision boundary that separates positive and negative instances of membrane proteins, the TMHMM is a generative method that simply attempts to model the membrane proteins. As an illustration of the difference, it is known that the TMHMM tends to yield false positives for sequences containing signal peptides—hydrophobic sequences in the N-terminal regions of proteins (Chen and Rost, 2002). The SDP/SVM approach tends to avoid these false positives, because signal peptides appear among the negative instances in the training set. Indeed, as shown in Lanckriet et al. (2003), signal peptides tend to be highly ranked by the TMHMM, and are more uniformly spread within the SDP/SVM rankings.

Finally, in order to test the robustness of our approach, a second experiment was performed in which a randomly generated kernel matrix K_{RND} was included among the kernel matrices used as input to our algorithm. This kernel matrix was generated by sampling 100-element vectors for each protein, where each component

Table 1.2 Functional categories. The table lists the 13 CYGD functional classifications used in these experiments. The class listed as "others" is a combination of four smaller classes: (1) cellular communication/signal transduction mechanism, (2) protein activity regulation, (3) protein with binding function or cofactor requirement (structural or catalytic) and (4) transposable elements, viral and plasmid proteins.

	Category	Size		Category	Size
1	metabolism	1048	8	cell rescue, defense & virulence	264
2	energy	242	9	interaction w/ cell. envt.	193
3	cell cycle & DNA processing	600	10	cell fate	411
4	transcription	753	11	control of cell. organization	192
5	protein synthesis	335	12	transport facilitation	306
6	protein fate	578	13	others	81
7	cellular transp. & transp. mech.	479			

of each vector was sampled independently from a standard normal distribution, and then computing inner products of the 100-element vectors to form K_{RND}. A control classifier trained using only the random kernel yields an ROC score of 0.5, indicating that K_{RND} is indeed uninformative for the classification problem at hand. More importantly, when a classifier is trained using all seven real kernels plus K_{RND}, SDP assigns the random kernel a weight that is close to zero. Thus, the ROC score derived from seven matrices does not change when the random matrix is added, indicating that the method is robust to the presence of noisy, irrelevant data.

1.6.2 Yeast Function Prediction

As a second test for our kernel-based approach, we follow the experimental paradigm of Deng et al. (2003a). The task is predicting functional classifications associated with yeast proteins, and we use as a gold standard the functional catalogue provided by the MIPS Comprehensive Yeast Genome Database (CYGD—`mips.gsf.de/proj/yeast`). The top-level categories in the functional hierarchy produce 13 classes (see Table 1.2). These 13 classes contain 3588 proteins; the remaining yeast proteins have uncertain function and are therefore not used in evaluating the classifier. Because a given protein can belong to several functional classes, we cast the prediction problem as 13 binary classification tasks, one for each functional class.

The primary input to the classification algorithm is a collection of kernel matrices representing different types of data. In order to compare the SDP/SVM approach to the MRF method of Deng *et al.*, we perform two variants of the experiment: one in which the five kernels are restricted to contain precisely the same binary information as used by the MRF method, and a second experiment in which two of the kernels use richer representations and a sixth kernel is added.

1.6.2.1 Kernels for protein function prediction

For the first kernel, the domain structure of each protein is summarized using the mapping provided by SwissProt v7.5 (`us.expasy.org/sprot`) from protein sequences to Pfam domains (`pfam.wustl.edu`). Each protein is characterized by a 4950-bit vector, in which each bit represents the presence or absence of one Pfam domain. The kernel function K_{Pfam} is simply the inner product applied to these vectors. This bit vector representation was used by the MRF method. In the second experiment, the domain representation is enriched by adding additional domains (Pfam 9.0 contains 5724 domains) and by replacing the binary scoring with log E-values derived by comparing the HMMs with a given protein using the HMMER software toolkit (`hmmer.wustl.edu`).

Three kernels are derived from CYGD information regarding three different types of protein interactions: protein-protein interactions, genetic interactions, and co-participation in a protein complex, as determined by tandem affinity purification (TAP). All three data sets can be represented as graphs, with proteins as nodes and interactions as edges. As explained before, each interaction graph allows to establish similarities among proteins through the construction of a corresponding diffusion kernel. This generates three interaction kernel matrices, K_{Gen}, K_{Phys} and K_{TAP}. Because direct physical interaction is not necessarily guaranteed when two proteins participate in a complex, a smaller diffusion constant — this parameter is required to construct a diffusion kernel (see Kondor and Lafferty, 2002) — is used to construct K_{TAP}, i.e., $\tau = 1$ instead of $\tau = 5$ for the others.

The fifth kernel is generated using 77 cell cycle gene expression measurements per gene (Spellman et al., 1998). Two genes with similar expression profiles are likely to have similar functions; accordingly, Deng *et al.* convert the expression matrix to a square binary matrix in which a 1 indicates that the corresponding pair of expression profiles exhibits a Pearson correlation greater than 0.8. We use this matrix to form a diffusion kernel K_{Exp}. In the second experiment, a Gaussian kernel is defined directly on the expression profiles: for expression profiles $\mathbf{x}_1$ and $\mathbf{x}_2$, the kernel is $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-||\mathbf{x}_1 - \mathbf{x}_2||^2/2\sigma)$ with width $\sigma = 0.5$.

In the second experiment, we construct one additional kernel matrix by applying the Smith-Waterman pairwise sequence comparison algorithm (Smith and Waterman, 1981) to the yeast protein sequences. Each protein is represented as a vector of Smith-Waterman log E-values, computed with respect to all 6355 yeast genes. The kernel matrix K_{SW} is computed using an inner product applied to pairs of these vectors. This matrix is complementary to the Pfam domain matrix, capturing sequence similarities among yeast genes, rather than similarities with respect to the Pfam database.

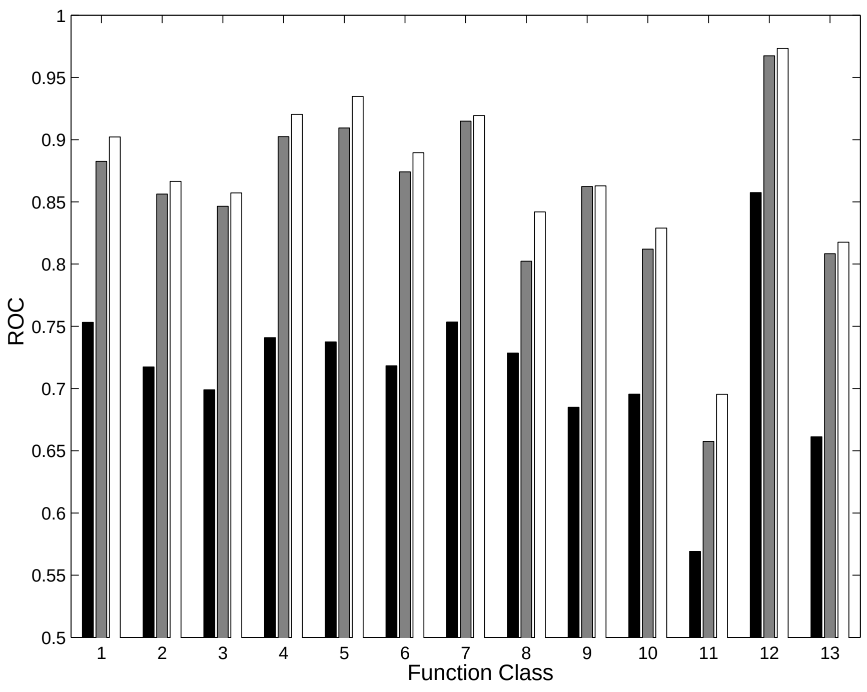

Figure 1.3 Classification performance for the 13 functional classes. The height of each bar is proportional to the ROC score. The standard deviation across the 15 experiments is usually 0.01 or smaller, so most of the depicted differences are significant. Black bars correspond to the MRF method of Deng *et al.*; grey bars correspond to the SDP/SVM method using five kernels computed on binary data, and white bars correspond to the SDP/SVM using the enriched Pfam kernel and replacing the expression kernel with the SW kernel.

1.6.2.2 Results

Each algorithm's performance is measured by performing 5-fold cross-validation three times. For a given split, we again evaluate each classifier by reporting the receiver operating characteristic (ROC) score on the test set. For each classification, we measure 15 ROC scores (three 5-fold splits), which allows us to estimate the variance of the score.

The experimental results are summarized in Figure 1.3. The figure shows that, for each of the 13 classifications, the ROC score of the SDP/SVM method is better than that of the MRF method. Overall, the mean ROC improves from 0.715 to 0.854. The improvement is consistent and statistically significant across all 13 classes. An additional improvement, though not as large, is gained by replacing the expression and Pfam kernels with their enriched versions. The most improvement is offered by using the enriched Pfam kernel and replacing the expression kernel with the

Table 1.3 Kernel weights and ROC scores for the transport facilitation class. The table shows, for both experiments, the mean weight associated with each kernel, as well as the ROC score resulting from learning the classification using only that kernel. The final row lists the ROC score using all kernels.

Kernel	Binary data		Enriched kernels	
	Weight	ROC	Weight	ROC
K_{Pfam}	2.21	.9331	1.58	.9461
K_{Gen}	0.18	.6093	0.21	.6093
K_{Phys}	0.94	.6655	1.01	.6655
K_{TAP}	0.74	.6499	0.49	.6499
K_{Exp}	0.93	.5457	—	.7126
K_{SW}	—	—	1.72	.9180
all	—	.9674	—	.9733

Smith-Waterman kernel. The resulting mean ROC is 0.870. Again, the improvement occurs in every class, although some class-specific differences are not statistically significant.

Table 1.3 provides detailed results for a single functional classification, the transport facilitation class. The weight assigned to each kernel indicates the importance that the SDP/SVM procedure assigns to that kernel. The Pfam and Smith-Waterman kernels yield the largest weights, as well as the largest individual ROC scores. Note that the combination of kernels performs significantly better than any single kernel. Results for the other twelve classifications are similar.

1.7 Discussion

We have described a general method for combining heterogeneous genome-wide data sets in the setting of kernel-based statistical learning algorithms, and we have demonstrated an application of this method to the problems of classifying yeast membrane proteins and protein function prediction in yeast. The resulting SDP/SVM algorithm yields significant improvement relative to an SVM trained from any single data type, relative to both state-of-the-art and classical biological methods for membrane protein prediction as well as relative to a previously proposed graphical model approach for fusing heterogeneous genomic data. More, the performance of the algorithm improves consistently in our experiments as additional genome-wide data sets are added to the kernel representation, if the added data contain complementary information.

Kernel-based statistical learning methods have a number of general virtues as tools for biological data analysis. First, the kernel framework accommodates not only the vectorial and matrix data that are familiar in classical statistical analysis, but also more exotic data types such as strings, trees and graphs. The ability to handle such data is clearly essential in the biological domain. Second, kernels provide significant opportunities for the incorporation of more specific biological knowledge, as we have seen with the FFT kernel and the Pfam kernel, and unlabelled data, as in the diffusion and Smith-Waterman kernels. Third, the growing suite of kernel-based data analysis algorithms require only that data be reduced to a kernel matrix; this creates opportunities for standardization. Finally, as we have shown here, the reduction of heterogeneous data types to the common format of kernel matrices allows the development of general tools for combining multiple data types. Kernel matrices are required only to respect the constraint of positive semidefiniteness, and thus the powerful technique of semidefinite programming can be exploited to derive general procedures for combining data of heterogeneous format and origin.

We thus envision the development of general libraries of kernel matrices for biological data, such as those that we have provided at `noble.gs.washington.edu/sdp-svm`, that summarize the statistically-relevant features of primary data, encapsulate biological knowledge, and serve as inputs to a wide variety of subsequent data analyses. Indeed, given the appropriate kernel matrices, the methods that we have described here are applicable to problems such as the prediction of protein metabolic, regulatory and other functional classes, the prediction of protein subcellular locations, and the prediction of protein-protein interactions.

References

B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Essential cell biology: an introduction to the molecular biology of the cell.* Garland Science Publishing, 1998.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

C. Berg, C. J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions.* Springer, New York, NY, 1984.

S. D. Black and D. R. Mould. Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications. *Anal. Biochem.*, 193: 72–82, 1991.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM, Philadelphia, PA, 1994. ISBN 0-89871-334-X.

S. Boyd and L. Vandenberghe. Convex optimization. Course notes for EE364, Stanford University. Available at `http://www.stanford.edu/class/ee364`, August 2001.

M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, Jr. M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *PNAS*, 97(1):262–267, 2000.

C. P. Chen and B. Rost. State-of-the-art in membrane protein prediction. *Applied Bioinformatics*, 1(1):21–35, 2002.

M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins. In *RECOMB*, pages 95–103, 2003a.

M. Deng, F. Sun, and T. Chen. Assessment of the reliability of protein-protein interactions and protein function prediction. In *PSB*, pages 140–151, 2003b.

A. Drawid and M. Gerstein. A Bayesian system integrating expression data with sequence patterns for localizing proteins: comprehensive application to the yeast genome. *J. Mol. Biol.*, 301:1059–1075, 2000.

D. M. Engleman, T. A. Steitz, and A. Goldman. Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. *Ann. Rev. Biophys.*

Biophys. Chem., 15:321–353, 1986.

T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

H. Ge, Z. Liu, G. Church, and M. Vidal. Correlation between transcriptome and interactome mapping data from *Saccharomyces cerevisiae*. *Nature Genetics*, 29: 482–486, 2001.

M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.

A. Grigoriev. A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage T7 and the yeast *S*accharomyces cerevisiae. *Nucleic Acids Res.*, 29:3513–3519, 2001.

J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

I. Holmes and W. J. Bruno. Finding regulatory elements using joint likelihoods for sequence and expression profile data. In *ISMB*, pages 202–210, 2000.

T. P. Hopp and K. R. Woods. Prediction of protein antigenic determinants from amino acid sequences. *Proc. Natl. Acad. Sci. USA*, 78:3824–3828, 1981.

T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *ISMB*, pages 149–158, Menlo Park, CA, 1999. AAAI Press.

R. Jansen, N. Lan, J. Qian, and M. Gerstein. Integration of genomic datasets to predict protein complexes in yeast. *Journal of Structural and Functional Genomics*, 2:71–81, 2002.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In C. Sammut and A. Hoffmann, editors, *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, 2002.

A. Krogh, B. Larsson, G. von Heijne, and E. L. L. Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: Application to complete genomes. *Journal of Molecular Biology*, 305(3):567–580, 2001.

J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157:105–132, 1982.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A framework for genomic data fusion and its application to membrane protein prediction. Technical Report 03-1273, University of California, Berkeley, Division of Computer Science, 2003.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In C. Sammut and A. Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning*. Morgan Kaufmann, 2002.

G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *PSB*, 2004.

L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *RECOMB*, pages 225–232, 2002.

E. M. Marcotte, M. Pellegrini, M. J. Thompson, T. O. Yeates, and D. Eisenberg. A combined algorithm for genome-wide prediction of protein function. *Nature*, 402(6757):83–86, 1999.

H. W. Mewes, D. Frishman, C. Gruber, B. Geier, D. Haase, A. Kaps, K. Lemcke, G. Mannhaupt, F. Pfeiffer, C Schüller, S. Stocker, and B. Weil. MIPS: a database for genomes and protein sequences. *Nucleic Acids Res.*, 28(1):37–40, 2000.

R. Mrowka, W. Lieberneister, and D. Holste. Does mapping reveal correlation between gene expression and protein-protein interaction? *Nature Genetics*, 33: 15–16, 2003.

Akihiro Nakaya, Susumu Goto, and Minoru Kanehisa. Extraction of correlated gene clusters by multiple graph comparison. In H. Matsuda, S. Miyano, T. Takagi, and L. Wong, editors, *Genome Informatics 2001*, pages 44–53. Universal Academy Press, 2001.

Y. Nesterov and A. Nemirovsky. *Interior point polynomial methods in convex programming: Theory and applications.* SIAM, Philadelphia, PA, 1994.

P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy. Gene functional classification from heterogeneous data. In *RECOMB*, pages 242–248, 2001.

B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

E. Sonnhammer, S. Eddy, and R. Durbin. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28(3):405–420, 1997.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell*, 9:3273–3297, 1998.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Special issue on Interior Point Methods (CD supplement with software).

Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:S136–S144, 2002.

K. Tsuda. Support vector classification with asymmetric kernel function. In M. Verleysen, editor, *Proceedings ESANN*, pages 183–188, 1999.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1): 49–95, 1996.

C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Olivier, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

A. Zien, G. Rätch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.