Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. TR-146

An Automated Combination of Sequence Motif Kernels for Predicting Protein Subcellular Localization

Alexander Zien[1], Cheng Soon Ong[1] [2]

April 2006

[1] Max Planck Inst. for Biol. Cybernetics, Spemannstr. 38, Tübingen, Germany
[2] Friedrich Miescher Lab, Max Planck Soc., Spemannstr. 39, Tübingen, Germany

An Automated Combination of Sequence Motif Kernels for Predicting Protein Subcellular Localization

Alexander Zien, Cheng Soon Ong

Abstract. Protein subcellular localization is a crucial ingredient to many important inferences about cellular processes, including prediction of protein function and protein interactions. While many predictive computational tools have been proposed, they tend to have complicated architectures and require many design decisions from the developer.

We propose an elegant and fully automated approach to building a prediction system for protein subcellular localization. We propose a new class of protein sequence kernels which considers all motifs including motifs with gaps. This class of kernels allows the inclusion of pairwise amino acid distances into their computation. We further propose a multiclass support vector machine method which directly solves protein subcellular localization without resorting to the common approach of splitting the problem into several binary classification problems. To automatically search over families of possible amino acid motifs, we generalize our method to optimize over multiple kernels at the same time. We compare our automated approach to four other predictors on three different datasets.

1 Introduction

Support vector machines (SVMs) [5, 25] are nowadays in widespread and highly successful use for bioinformatics tasks. One example is the prediction of the subcellular localization of proteins, as exemplified by many approaches in the literature including [8, 11, 12, 17, 22, 32, 33].

SVMs are popular for two reasons. First, they exhibit very competitive classification performance, since overfitting is controlled by easily tunable regularization and the training is not hampered by any local minima. Second, SVMs can conveniently be adapted to the problem at hand by designing appropriate kernel functions to represent prior knowledge about the similarity between the examples of the problem at hand. The kernel function implicitly maps points to a feature space via an associated function Φ. In that space dot products are computed efficiently via $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. For more details, see for example [25].

Many SVM-based subcellular localization prediction methods employ the Gaussian RBF kernel (eg [12, 17, 22, 33]). Further, in many approaches the representation $\mathbf{x}$ of a data point is naturally divided into parts. This arises from the fact that different types of evidence are considered. For instance, in [22] different features are derived from the protein sequence: its composition in terms of single amino acids, of pairs of adjacent amino acids, of pairs of amino acids with one position gap inbetween them, and so on. As a second example, in [8, 13, 33] each protein is divided into parts, and a separate set of amino acid composition features is computed for each part.

While all features mentioned so far are compositions, ie histograms of subsequences, several approaches introduce features that relate to the primary sequence more indirectly and are of a different nature. Examples include the search for known motifs [11, 17] or PFAM domains [13], the use of PSI-BLAST profiles [32], and the use of PSI-BLAST similarities to other sequences [12]. In some cases, even SVMs or other classifiers are employed for feature generation or for motif detection [11, 17].

When more than one set of features have been defined and computed, the task becomes to combine the evidence they yield into a single final prediction. This is often done in complex, hand-crafted architectures that frequently consist of two layers of learning machines or decision systems. In principle, there seem to exist three possible strategies of dealing with this situation:[1]

[1]For simplicity, we describe all three strategies solely for the SVM setting. The first two of them similarly apply to any other machine learning method; however note that the third strategy is restricted to kernel methods.

1. Concatenating all feature sets into a single feature vector to be used in a single Gaussian kernel [8, 32].
2. Defining an individual kernel on each type of features, training an individual SVM on each kernel, and combining the SVM outputs or predictions, for example by a jury method or by another SVM [11, 12, 13, 17, 22, 33].
3. Defining an individual kernel on each set of features, combining them into a single kernel (for instance, by adding them), and training a single SVM on that kernel.

The third method has empirically been shown to be the most effective [23]. Using this way it is possible to build complex modular kernel functions by combining several simpler ones. However, this strategy has not yet been used for predicting the subcellular localization of proteins.

One difficulty with adding kernels is that doing so with uniform weights is not always optimal. An extreme example is the case that one kernel is not correlated with the labels at all – then giving it positive weight just increases the amount of noise that the SVM has to cope with. Multiple kernel learning (MKL) is a way of optimizing kernel weights. In [20] it has been shown to be useful for a binary classification of proteins related to subcellular localization, namely membrane versus non-membrane proteins. In addition to leading to good classification accuracies, MKL can also be useful for identifying biologically relevant features [27] (there, for the task of splice site identification). In this paper we extend MKL to the multi-class case and demonstrate how it can be applied to select a well-performing combination of kernels from a family of motif composition kernels.

In this paper, we develop two novel tools:

1. We define a class of sequence motif kernels on amino acid sequences in Section 2.
2. We derive a new optimization problem which solves the multiclass multiple kernel learning problem in Section 3.

In Section 4, we show that our elegant and fully automated method compares favorably with the current state of the art.

2 Motif Composition Kernels

We develop a family of kernels on sequences that is based on the occurence of motifs. This is a special case of a general class of kernels on probability measures [14]. Our family of kernels allows to take into account pairwise comparisons of amino acids (AAs) as well as sequence motifs.

2.1 Base kernels

Before we consider motifs consisting of several amino acids, we define a kernel on individual amino acids. This will be useful as an ingredient to the more complex motif kernel.

Let $\mathcal{A}$ be the set of 20 amino acids. A substitution matrix M consists of a real-valued element m_{ab} for each pair of amino acids a and b. It has been shown that every sensible substitution matrix M implies a matrix R of amino acid substitution probabilities via $m_{ab} = \frac{1}{\lambda} \log \frac{r_{ab}}{q_a q_b}$. Here q_a is the so-called background probability of a, ie its relative frequency of appearance in any protein sequence, and λ is a scaling factor for convenient representation of the matrix M. Given the constraints $\sum_a \sum_b r_{ab} = 1$ and $q_a = \sum_b r_{ab}$ and the symmetry of both M and R, R can be computed from M. We do so for the popular BLOSUM62 matrix [15] serving as our M.

The elements of the obtained R, being substitution probabilities, are positive, and thus R can be seen as a (complete) similarity graph between amino acids with weighted edges. From this we derive a positive definite kernel k^{AA} on the amino acids by taking the graph Laplacian:

$$k^{AA}(a,b) = \sum_c r_{ac} - r_{ab} \quad . \tag{1}$$

Note that other choices of kernels are possible. One alternative is the diffusion kernel, which is computed by taking the matrix exponential of a scalar multiple of R. In this context we prefer the graph Laplacian since it does not have any parameters to be adjusted.

We extend the AA-kernel to k-tuples of amino acids by simply adding kernel values over the components. For $s, t \in \mathcal{A}^k$ we define

$$k^{AA}(s,t) = \sum_{i=1}^{k} k^{AA}(s_i, t_i) \quad . \tag{2}$$

2.2 Motif compositions

Previous work has shown that the amio acid composition (AAC) of a sequence is a useful basis for classifying its subcellular localization [24]. An advantage of this set of features is that it is robust with respect to small errors in the sequences, as may be caused by automated determination from genomic DNA. In subsequent work the AAC has been refined in two directions.

First, instead of just considering the AAC of the entire protein sequence, it was calculated on different subsequences [8, 13, 17]. This makes sense since important indications of localization are not global. For example the targeting of a protein to the mitochondrion or to the chloroplast is indicated by an N-terminal signal peptide with specific properties (eg, pH or hydrophobicity) that are reflected by the AAC.

Second, it was noted that features corresponding to more than a single amino acid can increase the prediction performance. This seems plausible since there exist a number of (rather short) known motifs that are important for subcellular targeting. Examples include the C-terminal targeting signal for microbodies (SKL), the C-terminal endoplasmatic reticulum targeting sequence (KDEL), and the bipartite nuclear targeting sequence (which consists of five basic amino acids, R or K, in a certain arrangement). Existing prediction methods that generalize the AAC to higher order compositions do so in at least two ways: [22] and [12] use composition of pairs of amino acids, possibly with fixed-length gaps between them. [33] consider distributions of k-length subsequences, where a reduced size alphabet is used to avoid combinatorial explosion of the feature space for large k.

Here we carry the generalization a bit further, by allowing for patterns consisting of any number k of amino acids in any (fixed) positional arrangement. For example, we could choose the frequencies of occurance of AA triplets with two positions gap between the first two and no gap between the second two, corresponding to a pattern $(0, 3, 4)$. More formally, we define a pattern d to be a list $(d_1, \ldots, d_k) \in \mathbf{N}^k$ of relative positions d_i with $d_1 = 0$ and $d_i < d_{i+1}$. We say that d is of order k and has length $|d| := d_k$.

For any given pattern, we can compute the empirical distribution of corresponding motifs from a given AA sequence. This is a histogram of occurrences of each possible k-mer sequence. The example above will result in a histogram of all possible 3-mers where each sequence is represented by the counts of the occurrences of each 3-mer with the specified gap. Note that the combinatorial explosion of possible motifs for increasing order k is not a real problem, because the number of motifs with positive probability is bounded by the protein length, and we employ sparse representations.

2.3 Motif composition kernels

The feature sets defined as described above are histograms, and after normalization they are probability distributions over discrete sets. While we can use standard kernels (like the Gaussian RBF) on these data, this would ignore the fact that they are not arbitrary vectors, but that the features are in fact probabilities. We prefer to resort to kernels that are specially designed for probability distributions [14]. These kernels have the added benefit of allowing us to model pairwise similarities between amino acids. To our knowledge, this is the first time such kernels have been applied to (protein) sequence analysis.

For brevity, we choose the Jensen-Shannon divergence kernel (corresponding to $\alpha = 1$ in [14]), which is based on a symmetric version of the Kullback-Liebler divergence of information theory. Applied to histograms on patterns of order k we have

$$\begin{aligned} k^{JS}(p, q) = & \sum_{s \in \mathcal{A}^k} \sum_{t \in \mathcal{A}^k} k^{AA}(s, t) \\ & \left(p(s) \log \frac{p(s)}{p(s) + q(t)} + q(s) \log \frac{q(s)}{p(s) + q(t)} \right) \quad , \end{aligned} \tag{3}$$

where p and q are the histograms corresponding to two sequences, and s and t are the amino acid motifs that the distributions range over. For this paper, we define the kernels between amino acids $k^{AA}(s, t)$ using the summed graph Laplacian defined in Equations (1) and (2).

Even using these choices, we are still left with a large number of possible patterns (as defined in Section 2.2) to consider. Instead of using an arbitrary combination of them, we develop a method which optimizes over any finite set of the kernels automatically to form the best predictor. In principle, our method is only limited by computational and storage limits.

3 Multiclass Multiple Kernel Learning

Recently there have been developments in machine learning on optimizing over a set of kernels while choosing the best predictor [1, 2, 6, 19, 21]. The algorithms consider a linear combination of kernels and optimize to find the best combination for the prediction task at hand. In this paper, we propose the first multiclass version of this approach. Furthermore, we use an $\mathcal{L}_1$ regularizer on the combination weights to promote sparsity. To our knowledge, this particular regularizer is also new.

3.1 Multiclass SVMs (m-SVMs)

We consider linear classifiers in a feature space $\mathcal{H}$ defined by a potentially non-linear map $\Phi : \mathcal{X} \to \mathcal{H}$. Binary SVMs learn a linear decision function $f(\cdot; \mathbf{w}, b) : \mathcal{X} \to \mathbf{R}$ defined by $f(\mathbf{x}; \mathbf{w}, b) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$. More precisely, training amounts to finding good values for the feature weights $\mathbf{w}$ and the bias b. The value $f(\mathbf{x}; \mathbf{w}, b)$ is often called the SVM output for $\mathbf{x}$. The sign of $f(\mathbf{x}; \mathbf{w}, b)$ corresponds to the predicted class in $\mathcal{Y} = \{+1, -1\}$. The magnitude of the output can be seen as a measure of confidence in the prediction.

This suggests a way of generalizing the learning machine to more than two classes. Let $\mathcal{Y}$ now be a set of $m > 2$ classes. We now consider m output functions, one for each class, that quantify the confidence in the corresponding prediction. To do so we follow the modelling used in [29]. The key idea of this approach is to use a joint feature map between data $\mathcal{X}$ and labels $\mathcal{Y}$ denoted by $\Phi(\mathbf{x}, y)$. The output function for a class $y \in \mathcal{Y}$ can then be defined as

$$f_y(\mathbf{x}; \mathbf{w}, \mathbf{b}) = \langle \mathbf{w}, \Phi(\mathbf{x}, y) \rangle + b_y \ , \tag{4}$$

with $\mathbf{b} = (b_1, \ldots, b_m)$. Thus, the predicted class y for a point $\mathbf{x}$ is chosen to maximize the confidence in the prediction:

$$\arg\max_{y \in \mathcal{Y}} f_y(\mathbf{x}; \mathbf{w}, \mathbf{b}) \ . \tag{5}$$

Training consequently amounts to finding parameters $\mathbf{w}$ and $\mathbf{b}$ that satisfy, at least to a large extent,

$$f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) \quad > \quad f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) \quad \forall u \in \mathcal{Y} - \{y_i\} \ ,$$

for data points $(\mathbf{x}_i, y_i)$. Hence for a general convex monotonically decreasing loss function ℓ, training (with $\mathcal{L}_2$ regularization, which enables kernelization) can be implemented by the following optimization problem:

$$\min_{\mathbf{w}, \mathbf{b}} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max_{u \neq y_i} \{ \ell_y \left(f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) - f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) \right) \} \ , \tag{6}$$

where n is the number of examples in the training set, and we write $u \neq y_i$ short for $u \in \mathcal{Y} - \{y_i\}$.

For the particular choice of the hinge loss, $\ell(t) = C \min\{0, 1 - t\}$, we call (6) and any derived equivalent optimization problem m-SVM (for m-class SVM), since it generalizes the standard binary SVM. Deriving the Lagrange dual (cf. [3]) of (6) yields an SVM style expansion of the hyperplane normal,

$$\mathbf{w} = \sum_i \sum_{u \in \mathcal{Y}} \alpha_{iu} \Phi(\mathbf{x}_i, u) \ . \tag{7}$$

This nice property also follows from the famous representer theorem (eg [25]). Here $\boldsymbol{\alpha} \in \mathbf{R}^{n \times m}$ is the solution of the following quadradic program (QP):

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}} & \frac{1}{2} \|\mathbf{w}(\boldsymbol{\alpha})\|^2 + \sum\limits_i \alpha_{iy_i} \\ \text{s.t.} & \forall i : -C \leq \alpha_{iy_i} \leq 0 \\ & \forall i : \forall u \neq y_i : \alpha_{iu} \geq 0 \\ & \forall i : \sum\limits_{u \in \mathcal{Y}} \alpha_{iu} = 0 \\ & \forall u \in \mathcal{Y} : \sum\limits_i \alpha_{iu} = 0 \ , \end{array} \tag{8}$$

where we write $\mathbf{w}(\boldsymbol{\alpha})$ in the objective function to stress that the right hand side of (7) should be plugged in. For Lagrange duals of versions with different losses (but without a bias $\mathbf{b}$) see [29]. In our formulation, the bias can be calculated according to the complementary slackness conditions [3] as follows. For a point i with $-C < \alpha_{iy_i} < 0$ we have $f_{y_i}(\mathbf{x}_i; \mathbf{w}, b) - f_u(\mathbf{x}_i; \mathbf{w}, b) = 1$ for all $u \in \mathcal{Y} - \{y_i\}$ with $\alpha_{iu} > 0$. For fixed $\boldsymbol{\alpha}$, this results in a linear equality system in $\mathbf{b}$ that can be solved, for instance, by least squares.

Lagrange duals demonstrate the significance of kernel functions. Since the QP can be expressed in terms of dot products of data points, instead of having to compute the feature map $\Phi(\mathbf{x}, y)$ explicitly it is sufficient for the training to compute the associated kernel function,

$$k((\mathbf{x}, y), (\mathbf{x}', y')) := \langle \Phi(\mathbf{x}, y), \Phi(\mathbf{x}', y') \rangle \quad , \tag{9}$$

which maps $(\mathcal{X} \times \mathcal{Y})^2$ to $\mathbf{R}$. Further, as the hyperplane normal $\mathbf{w}$ can be expressed as a linear combination of feature maps of data points 7, kernel function evaluations are also sufficient for prediction:

$$f_y(\mathbf{x}) = \sum_i \sum_{u \in \mathcal{Y}} \alpha_{iu} \langle \Phi(\mathbf{x}_i, u), \Phi(\mathbf{x}, y) \rangle + b_y \quad . \tag{10}$$

Multiclass classification can be considered the simplest learning problem with non-trivial structure on the outputs. Multiclass SVMs have been investigated for several years now [18, 31]. While it is still common practice to reduce it to a set of binary classification problems (eg. [9]), in general genuine multiclass approaches can be superior [7]. Further, they can naturally be extended to more complex output structures, eg sequences [28, 29].

3.2 Multiple kernel learning (MKL)

We can generalize the above m-SVM further to operate on $p \geq 1$ feature maps $\Phi_k(\mathbf{x}_i, y_i)$ (for $k = 1, \ldots, p$). For each feature map there will be a separate weight vector $\mathbf{w}_k$; for convenience, we will denote them jointly by $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_p)$. Here we consider linear combinations of feature maps of the form $\Phi(\mathbf{x}_i, y_i) = \sum_{k=1}^p \beta_k \Phi_k(\mathbf{x}_i, y_i)$ that are specified by mixture weights $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$. This gives rise to the following output functions:

$$f_y(\mathbf{x}; \mathbf{w}, \mathbf{b}, \boldsymbol{\beta}) \quad = \quad \sum_{k=1}^p \beta_k \langle \mathbf{w}_k, \Phi_k(x, y) \rangle + b_y \quad .$$

As before, an ideal combination of functions would correctly classify any point x, including the training points x_i. We thus aim at choosing $\mathbf{w}$ and $\boldsymbol{\beta}$ such that $f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}, \boldsymbol{\beta}) \geq f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b}, \boldsymbol{\beta})$ for all $u \in \mathcal{Y} - \{y_i\}$. The resulting optimization problem, a generalization of Equation (6), can be written like this:

$$\begin{aligned} \min_{\boldsymbol{\beta}, \mathbf{w}, \mathbf{b}, \xi} \quad & \frac{1}{2} \sum_{k=1}^p \beta_k \|\mathbf{w}_k\|^2 + \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \forall i : \xi_i = \max_{u \neq y_i} \ell_y \left(f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}, \boldsymbol{\beta}) - f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b}, \boldsymbol{\beta}) \right) \\ & \sum_{k=1}^p \beta_k = 1, \ \ \forall k : 0 \leq \beta_k \end{aligned} \tag{11}$$

Note that we constrain each β_k to be positive; this way, it is guaranteed that the combined kernel is positive definite. We also constrain the $\mathcal{L}_1$ norm of $\boldsymbol{\beta}$ to be unity. Otherwise, the regularizer on $\mathbf{w}$ would not be effective: it could be driven to zero without changing f by dividing $\mathbf{w}$ by any positive scalar while multiplying $\boldsymbol{\beta}$ with the same number.

By deriving the Lagrangian dual, we again find that at the optimum each hyperplane normal can be expanded in the training points,

$$\mathbf{w}_k \quad = \quad \sum_i \sum_{y \in \mathcal{Y}} \alpha_{iu} \Phi_k(\mathbf{x}_i, u) \quad . \tag{12}$$

From the general dual, we derive our specific optimization problem by using the popular SVM hinge loss, where we use the same loss for each class. While it is possible to use a different loss for each class, we focus on a single loss function to reduce the number of parameters in our problem.

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \gamma + \sum_i \alpha_{iy_i} \\
\text{s.t.} \quad & \forall i : -C \leq \alpha_{iy_i} \leq 0 \\
& \forall i : \forall u \neq y_i : \alpha_{iu} \geq 0 \\
& \forall i : \sum_{u \in \mathcal{Y}} \alpha_{iu} = 0 \\
& \forall u \in \mathcal{Y} : \sum_i \alpha_{iu} = 0 \\
& \forall k : \gamma \geq \frac{1}{2} \left\| \mathbf{w}_k(\boldsymbol{\alpha}) \right\|^2
\end{aligned} \tag{13}$$

For the case of a single kernel, $p = 1$, there is a single constraint in addition to those in problem (8). We further observe that at the optimum this constraint will always be at equality, so that we can substitute its right hand side back into the objective. This way, we exacly get back (8), demonstrating that our m-SVM is just a special case of the MKL formulation.

The problem (13) is a quadratically constrained linear program: its objective is linear, and so are all constraints but the last, which each consist of the square of a linear form in the optimization variables. In order to solve this optimization problem, we convert it into an equivalent semi-infinite linear program (SILP) formulation by a second (partial) dualization. We obtain:

$$\begin{aligned}
\max_{\boldsymbol{\beta}} \quad & \theta \\
\text{s.t.} \quad & \forall \boldsymbol{\alpha} \in \mathcal{S} : \;\; \theta \leq \frac{1}{2} \sum_k \beta_k \left\| \mathbf{w}_k(\boldsymbol{\alpha}) \right\|^2 + \sum_i \alpha_{iy_i} \\
& \forall k : \;\; 0 \leq \beta_k \\
& \sum_{k=1}^{p} \beta_k = 1 \;\; ,
\end{aligned} \tag{14}$$

where

$$\mathcal{S} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \forall i : -C \leq \alpha_{iy_i} \leq 0 \\ \forall i : \forall u \neq y_i : \alpha_{iu} \geq 0 \\ \forall i : \sum_{u \in \mathcal{Y}} \alpha_{iu} = 0 \\ \forall u \in \mathcal{Y} : \sum_i \alpha_{iu} = 0 \end{array} \right\} \tag{15}$$

is the set of admissable parametrizations for the first constraint. Contrary to (13), solving the problem (14) yields the values for $\boldsymbol{\beta}$, but not for $\boldsymbol{\alpha}$. However, once we know the optimal $\boldsymbol{\beta}$, the problem reduces to a m-SVM (8) with a correspondingly mixed kernel, and $\boldsymbol{\alpha}$ and $\mathbf{b}$ can be determined as described in the last section.

Observe that for a fixed $\boldsymbol{\beta}$, Equation (14) is equivalent to a quadratic program (QP) which is only slightly more complicated than a standard SVM. Furthermore, for fixed $\boldsymbol{\alpha}$, the optimization problem in $\boldsymbol{\beta}$ is a linear program (LP). However, the constraint on θ has to hold for every suitable $\boldsymbol{\alpha}$; hence the name (refering to the infinitely many constraints).

This suggest the use of a column generation strategy to solve (14): Solving the QP given by the constraints for a fixed $\boldsymbol{\beta}$ results in a particular $\boldsymbol{\alpha}$, which gives rise to a constraint on θ which is linear in $\boldsymbol{\beta}$. We alternate generating new constraints in this way, and solving the LP with the constraints collected so far. This procedure is known to converge [16, 27]. Hence our seemingly complicated problem of finding the best multiclass predictor with several kernels can be solved with off-the-shelf QP and LP solvers. In our implementation, we used CPLEX for both problems. Another way of achieving this is the use of semi-definite programing (SDP) [3, 19], which however is computationally more expensive. The above result extends the result of [27], and the details of the derivation can be found in the appendix.

3.3 Decomposable Kernels

The multiclass SVM is defined in a very general problem setting, in which the feature maps Φ_k (and accordingly, also the kernels k_k) are defined on pairs $(\mathbf{x}, y)$ of data points and labels. However, the motif compostion kernels that we define in Section 2 are only defined on the points $\mathbf{x}$. We now extend those kernels to pairs $(\mathbf{x}, y)$.

Let $k_\mathcal{X}$ be a kernel on $\mathcal{X}$, ie $k_\mathcal{X} : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$, and let $k_\mathcal{Y}$ be a kernel on $\mathcal{Y}$, ie $k_\mathcal{Y} : \mathcal{Y} \times \mathcal{Y} \to \mathbf{R}$. Then a joint kernel $k_{\mathcal{X}\times\mathcal{Y}}$ can be defined by

$$k_{\mathcal{X}\times\mathcal{Y}}\left((\mathbf{x}, y), (\mathbf{x}', y')\right) = k_\mathcal{X}(\mathbf{x}, \mathbf{x}') \cdot k_\mathcal{Y}(y, y') \ . \tag{16}$$

This structure is reflected in the feature space: the joint feature space will be the tensor product of the features space on $\mathcal{X}$ with that on $\mathcal{Y}$.

In our experiments, we will use (16) with the matching kernel (or identity kernel) on $\mathcal{Y}$, ie $k_\mathcal{Y}(y, y') = \delta_{yy'}$. This kernel imposes the least structure on the classes. Note that available prior knowledge on pairwise relationships between classes could be taken into account by defining a corresponding $k_\mathcal{Y}$.

With a kernel that decomposes as in (16), the quadratic terms in the optimization problems (8, 12, 13) simplify to a Kronecker product of kernel matrices on $\mathcal{X}$ and $\mathcal{Y}$. Further, for our choice of the matching kernel on $\mathcal{Y}$, we get

$$\begin{aligned} \|\mathbf{w}\|^2 &= \sum_i \sum_j \sum_{u\in\mathcal{Y}} \sum_{v\in\mathcal{Y}} \alpha_{iu}\alpha_{jv} \left\langle \Phi(\mathbf{x}_i, u), \Phi(\mathbf{x}_j, v) \right\rangle \\ &= \sum_i \sum_j k_\mathcal{X}(\mathbf{x}_i, \mathbf{x}_j) \sum_{u\in\mathcal{Y}} \alpha_{iu}\alpha_{ju} \ . \end{aligned} \tag{17}$$

Thus the joint kernel matrix is sparse (with n^2m non-zero elements) and can be arranged into a block-diagonal form. This structure allows to save memory and computation time.

4 Experiments

In this section, we perform two types of experiments to compare our proposed method to current state of the art methods. The first (Section 4.3) is to demonstrate that the kernels proposed in Section 2 discover useful motifs for predicting subcellular location from the associated amino acid sequence. The second experiment (Section 4.4) uses a subset of the data from [20] to compare our proposed multiclass multiple kernel learning algorithm to their related binary classification multiple kernel learning algorithm.

4.1 Performance Measures

To be clear about the definitions of the various performance measures used to report multiclass results, we collect all the definitions in this section. Most measures are defined with respect to a particular class, say A, and can be calculated from the corresponding confusion matrix:

		Predicted Label	
		A	$\neg A$
Actual	A	True Positive (TP)	False Negative (FN)
Label	$\neg A$	False Positive (FP)	True Negative (TN)

From the confusion matrix above, we can define the various performance measures, including the Matthews Correlation Coefficient (MCC).

Measure	Formula
Accuracy	$\frac{(TP+TN)}{(TP+TN+FP+FN)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall / Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+TP)}$
MCC	$\frac{TP\,TN-FP\,FN}{\sqrt{(TP+FN)(TP+FP)(TN+FP)(TN+FN)}}$

4.2 Experimental Protocol

In the optimization problem described by Equation (14), we have only one parameter to choose, namely the regularization parameter C. Following [4], for each kernel k_k its default value of the regularization parameter C_k is computed to be the inverse of the variance of the points in feature space. Then the global default value of C is defined to be their geometric mean, that is $C_0 := \left(\prod_{k=1}^p C_k\right)^{(1/p)}$. The protocol for all our experiments is as follows:

1. Ten random splits into 80% training / 20% test data are prepared.

2. For each training set, the parameter C is chosen using 5-fold or 2-fold cross validation on the training set only. We searched over a grid of values $\{1/27, 1/9, 1/3, 1, 3, 9, 27\}$ relative to the default C_0.

3. The figure of merit for choosing C is the area under receiver operating characteristic (auROC) for binary classification and the F1 score for all the multiclass tasks. The best C is chosen using the performance indicator on the validation (hold out) set.

4. Using the best value of C, we solve Equation (14) using the full training set and predict the labels of the test set.

In order to compare with existing methods, we compute several different measures of performance, each corresponding to the method we are comparing to. This also demonstrates the robustness of our method with respect to various figures of merit.

4.3 Comparing with other localization predictors

To investigate the usefulness of the kernel proposed in Section 2, we compare our approach with four other approaches on three different datasets. Here, we focus our investigations on the kernel, and we investigate our proposed algorithm in Section 4.4. Observe that it is not possible to evaluate the many patterns available to the histogram kernels using traditional methods, hence our proposed algorithm is crucial.

For a comparative evaluation of the performance of our approach we utilize two different datasets, for which results with other methods are reported in the literature. The first is the data set used for training and evaluating TargetP [10]. It consists of two subsets for plants and non-plants. The second dataset is a database of bacterial proteins, PSORTdb [11].

4.3.1 Comparison on TargetP dataset

The original plant dataset of TargetP [10] is divided into five classes: chloroplast (ch), mitochondria (mi), secretory pathway (SP), cytoplasm (cy), and nucleus (nuc). However, in many reported results, cy and nuc are fused into a single class "other" (OT). We retain the original class distributions throughout the model selection and training phases. However, for comparison purposes, we merge the nuclear and cytoplasmic proteins during the test phase.

In our experiments with the plant data, we use 81 kernels, corresponding to motifs up to length 6 with up to 2 gaps. Each random split contains 14 cross validation optimizations (2-fold cross-validation to select from 7 values of C), with an additional larger optimization at the end. The computation time for each split is roughly 10 hours on a 2.4Ghz AMD64 machine.

From Table 1, our performance is comparable to TargetP but worse than TargetLoc. Our method has much higher specificity but suffers from lower sensitivity. The features most often selected for classification as seen in Table 2 are at the N-terminus of the protein and are 6-mers. This implies that the localization signals in plants may contain rather long motifs.

Data	Class	Our Method			TargetP			TargetLoc		
		SE	SP	MCC	SE	SP	MCC	SE	SP	MCC
plant	ch	78.8±8.9	96.1±1.7	75.5±6.2	85	69	72	88	76	78
	mi	88.3±4.0	89.8±2.3	77.7±3.6	82	90	77	87	94	84
	SP	93.5±3.6	96.8±1.2	90.2±2.8	91	95	90	93	97	93
	OT	76.3±4.5	96.4±1.5	75.0±5.3	85	78	77	92	84	86
nonplant	mi	83.6±3.7	96.2±1.1	77.6±4.8	89	67	73	91	77	81
	SP	90.0±2.6	94.9±1.2	83.9±1.9	96	92	92	95	92	91
	OT	91.2±1.4	91.0±1.4	81.5±1.4	88	97	82	91	97	86

Table 1: Comparing with TargetP and TargetLoc on sensitivity (SE), specificity (SP) and Matthew's Correlation Coefficient (MCC) on the plant dataset. The classes are chloroplast (ch), mitochondria (mi), secretory pathway (SP) and other (OT).

subsequence	pattern	times selected	mean β_k
$[1, 15]$	(0,1,2,3,5,6)	10	0.204
$[1, 60]$	(0,1,2,3,5,6)	10	0.165
$[1, 15]$	(0,1,2,4,5,6)	10	0.131
$[100, \infty]$	(0,1,2,4)	9	0.085
$[1, \infty]$	(0,1,2,3)	8	0.082
$[1, \infty]$	(0,1,2,4)	6	0.059
$[1, 15]$	(0,1,2,3,4,5)	7	0.056
$[1, 60]$	(0,1,2,3,4,6)	7	0.055
$[1, 15]$	(0,1,3,4,5,6)	7	0.052
$[1, 100]$	(0,1,2,4)	8	0.044
$[1, 100]$	(0,1,3,4)	6	0.031
$[100, \infty]$	(0,1,2,3)	3	0.028
$[1, 15]$	(0,1,2,3,4)	2	0.006
$[1, \infty]$	(0,2,3,4)	1	0.002
$[1, 15]$	(0,2,3,4,5,6)	1	0.001

Table 2: Kernels selected in the ten repetitions of experiments on the plant dataset, sorted by importance as indicated by the averaged coefficient β_k. Note that the selection is very consistent across the repetitions, and that only a small fraction of kernels obtained a positive weight in any repetition. The first column shows the considered region of the protein, starting with 1 at the N-terminus; ∞ means that the region extends to the C-terminus. The second column shows the pattern associated with the kernel.

4.3.2 Comparison on PSORTdb dataset

We use sequences and localizations of proteins in Gram-positive bacteria as obtained from PSORTdb [11]. We only consider singly localized proteins. For the Gram positive bacteria, we compute 50 kernels corresponding to motifs up to length 5 with up to 2 gaps. PSORTb has the option of withholding a prediction when it is uncertain about the localization. To compare our performance with the results in [11], we estimate the proportion of "unknown" prediction from their supplementary website to be 13% for the Gram positive bacteria. Then, we compute probabilistic outputs from our method by using the softmax function, that is $\frac{\exp f(x)}{\sum_u \exp f(u)}$. We then discard the most uncertain predictions based on the fraction obtained above. The results of Table 3 are the mean and standard deviations of this reduced test set.

Comparing the results of our method versus PSORTb shows that we significantly better at class EC, but significantly worse at class CW. We perform comparably in the other two classes. It is important to note that PSORTb uses various sources of information in a two layer architecture while our method directly uses the sequence information only. We also analyze the motifs most often selected for classification, which is shown in Table 4. Of the 50 kernels, these are the only ones with non-zero weight in our method. More detailed analysis of the top features show that 2-mers on the C-Terminal end of the protein are useful for classification, and 4-mers on the N-terminal end of the protein are useful.

Data	Class	Our Method			PSORTb v2.0		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
PSORT+	C	85.7±5.5	97.3±2.5	91.1±3.4	97.1	86.6	91.6
	CM	97.8±3.0	87.6±7.6	92.2±4.4	96.9	91.3	94.0
	CW	88.5±12.8	75.4±14.0	80.0±8.3	94.7	88.5	91.5
	EC	86.1±7.4	84.4±5.8	85.1±5.5	93.9	67.8	78.7

Table 3: Comparing with PSORTb v2.0 on singly located proteins from PSORTdb on Gram positive bacteria. The classes are cytoplasm (C), cytoplasmic membrane (CM), extracellular (EC) and cell wall (CW).

4.4 Comparing with a previous/binary MKL approach

We investigate the performance of the proposed algorithm by using previously defined kernels computed for the detection of membrane proteins [20]. Since we generalize their binary classification method to multiclass, we also generalize the data by splitting the class of non-membrane proteins. Using the MIPS classifications of yeast protein

subsequence	**pattern**	**times selected**	**mean β_k**
$[60, \infty]$	(0,2)	10	0.359
$[60, \infty]$	(0,3)	10	0.139
$[1, 15]$	(0,1,3,4)	10	0.125
$[1, 60]$	(0,1,2,4)	10	0.092
$[60, \infty]$	(0,1)	8	0.085
$[1, 15]$	(0,1,2,4)	8	0.068
$[1, 100]$	(0,1,3,4)	6	0.046
$[1, 100]$	(0,1,2,4)	6	0.046
$[1, 60]$	(0,2,3,4)	4	0.021
$[1, 60]$	(0,1,3,4)	2	0.007
$[1, 100]$	(0,2,3,4)	1	0.005
$[1, \infty]$	(0,1)	1	0.004
$[1, 15]$	(0,1,2,3)	2	0.002

Table 4: Analoguous to Table 2, but for gram-positive bacteria for PSORTb.

subcellular localization [30] (version of 14 Nov 2005), we selected the three localizations with the largest numbers of singly localized proteins from the subset of non-membrane proteins. Hence our dataset contains 284 cytoplasmic proteins, 228 nuclear proteins, 140 mitochondrial proteins, and 497 membrane proteins (corresponding to the ones in [20]).

To directly compare the two algorithms, we merged the three classes of non-membrane proteins and applied our multiclass algorithm to the two class problem. We use exactly the same 7 kernel matrices provided by [20]. The results in Table 5 show that we perform slightly worse than the binary classifier. This may have been caused by the significantly smaller dataset (we have 1149 examples while the original data had 2318 examples).

Method	Accuracy	AROC	TP1FP
Our Method	87.9±1.4	87.1±2.8	21.3±14.1
Lanckreit et. al.	88.7±0.2	92.2±0.2	36.1±2.0

Table 5: Membrane versus non-membrane proteins: comparing with a previous MKL approach for test set accuracy, area under the receiver operating characteristic curve (AROC) and the percentage true positives at one percent false positives (TP1FP).

However, our algorithm can distinguish between the non-membrane proteins as well, as shown in Table 6. The definition of recall score is the same as the sensitivity score in Section 4.3.

Class	Accuracy	Precision	Recall	F1-Score
C	89.6±1.5	79.8±4.9	80.5±5.0	80.0±3.2
N	91.9±1.3	80.8±6.7	82.8±3.2	81.6±3.0
mi	92.0±1.8	79.0±6.4	56.0±11.9	65.1±9.4
mem	88.0±1.6	84.9±2.7	90.5±2.4	87.6±1.9

Table 6: Performance on yeast data, classifying the proteins into the cytoplasm (C), nucleus (N), mitochondria (mi), and membrane (mem).

5 Conclusion

We have proposed a family of histogram-based motif kernels for amino acid sequences. We then optimize over the set of kernels using a multiclass multiple kernel learning method. We demonstrate that our method performs comparably to the current state of the art in protein subcellular localization. Further, this is already achieved with only using information from the amico acid sequence, while our method offers a principled way of integrating other data types (cf. [20, 23]).

As has been shown before [26], multiple kernel learning can be used to identify individual features that are relevant to the classification. This makes the results interpretable and may aid in getting insight into biological mechanisms. The idea of this approach is to represent the kernel by a sum of subkernels and to learn a weight (importance) for each of them. In principle, we could proceed in a similar way for our motif kernels, in order to identify individual relevant motifs. However, subdividing the Jensen-Shannon kernel may be difficult, and simpler kernels might have to be used.

Finally, the proposed approach is very general, and could be beneficial for other multiclass bioinformatics prediction problems. Since the derivation works with arbitrary monotonically decreasing convex loss functions, class-dependent (and even data point-dependent) loss can easily be implemented. This provides another possibility for piping prior knowledge on the classes or the data into the learning process.

Acknowledgement

We thank Alex Smola for stimulating discussions, and Gunnar Rätsch for practical help with optimization. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Appendix: Deriving the Optimization Problem

We prove the dualization for the multiple kernel case. Recall that n denotes the number of data points, and there are $m := |\mathcal{Y}|$ classes and p kernels. The single kernel case falls out as special case with $p = 1$.

For later use in the proof we recall a lemma about conjugate functions and perspective functions [3].

Lemma 1 (Perspective of a conjugate function) *For $\boldsymbol{\alpha}, \mathbf{t}, \boldsymbol{\eta} \in \mathbf{R}^{n\times m}$ such that $\boldsymbol{\eta} \geq 0$,*

$$-\eta_{iu}\ell_y^*\left(-\frac{\alpha_{iu}}{\eta_{iu}}\right) = \inf_{t_{iu}}\left(\alpha_{iu}t_{iu} + \eta_{iu}\ell_y(t_{iu})\right)$$

for all $i = 1, \dots, n$ and $u \in \mathcal{Y}$.

Proof Recall the definitions of conjugate functions and perspective functions. For any $f : \mathbf{R}^n \to \mathbf{R}$ the conjugate dual $f^* : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is given by:

$$f^*(q) = \sup_{t\in\mathbf{R}^n}\left(q^\top t - f(t)\right).$$

For $f : \mathbf{R}^n \to \mathbf{R}$ the perspective of f is the function $g : \mathbf{R}^{n+1} \to \mathbf{R}$ defined by

$$g(\alpha, \eta) = \eta f(\alpha/\eta)$$

for $\eta > 0$.

Hence for $\eta_{iu} > 0$, the lemma above follows from the definition of the perspective function of the conjugate function. From the right hand side,

$$\begin{aligned}\inf_{t_{iu}}(\alpha_{iu}t_{iu} + \eta_{iu}\ell_y(t_{iu})) &= \eta_{iu}\inf_{t_{iu}}\left(\tfrac{\alpha_{iu}}{\eta_{iu}}t_{iu} + \ell_y(t_{iu})\right)\\ &= -\eta_{iu}\sup_{t_{iu}}\left(-\tfrac{\alpha_{iu}}{\eta_{iu}}t_{iu} - \ell_y(t_{iu})\right)\\ &= -\eta_{iu}\ell_y^*\left(-\tfrac{\alpha_{iu}}{\eta_{iu}}\right).\end{aligned}$$

Some care has to be taken when $\boldsymbol{\eta} \geq 0$ but $\eta_{iu} = 0$ for some i and u. In this case, observe from the right hand side that if $\alpha_{iu} = 0$ and $\eta_{iu} = 0$ then $-\eta_{iu}\ell_y^*\left(-\frac{\alpha_{iu}}{\eta_{iu}}\right) = 0$, and if $\alpha_{iu} \neq 0$ and $\eta_{iu} = 0$ then $-\eta_{iu}\ell_y^*\left(-\frac{\alpha_{iu}}{\eta_{iu}}\right) = -\infty$. Hence the function $-\eta_{iu}\ell_y^*\left(-\frac{\alpha_{iu}}{\eta_{iu}}\right)$ is well defined for all $\boldsymbol{\eta} \geq 0$. ■

Theorem 2 *The dual of the multiclass multiple kernel learning problem*

$$\begin{aligned}\min_{\boldsymbol{\beta},\mathbf{w},\mathbf{b}} \quad & \frac{1}{2}\sum_{k=1}^{p}\beta_k\|\mathbf{w}_k\|^2 + \sum_{i=1}^{n}\max_{u\neq y_i}\ell_y\left(f_{y_i}(\mathbf{x}_i;\mathbf{w},\mathbf{b},\boldsymbol{\beta}) - f_u(\mathbf{x}_i;\mathbf{w},\mathbf{b},\boldsymbol{\beta})\right)\\ \text{s.t.} \quad & \sum_{k=1}^{p}\beta_k = 1,\ \ \forall k : 0 \leq \beta_k\end{aligned} \tag{18}$$

is equivalent to

$$\mathbf{w}_k = \sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \left(\Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right) \tag{19}$$

where $\tilde{\alpha}$ is deterimined by

$$\begin{aligned} \min_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \quad & \gamma + \sum_i \sum_{u \neq y_i} \eta_{iu} \ell_y^* \left(-\frac{\tilde{\alpha}_{iu}}{\eta_{iu}} \right) \\ \text{s.t.} \quad & \forall k : \gamma \geq \frac{1}{2} \sum_i \sum_j \sum_{u \neq y_i} \sum_{v \neq y_j} \tilde{\alpha}_{iu} \tilde{\alpha}_{jv} \left\langle \Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u), \Phi_k(\mathbf{x}_j, y_j) - \Phi_k(\mathbf{x}_j, v) \right\rangle, \\ & \forall i : \forall u \neq y_i : 0 \leq \eta_{iu}, \\ & \forall i : 1 = \sum_{u \neq y_i} \eta_{iu}, \\ & \forall v : 0 = \sum_i (1 - \delta_{y_i v}) \tilde{\alpha}_{iv} - \sum_i \delta_{y_i v} \sum_{u \neq y_i} \tilde{\alpha}_{iu} \end{aligned} \tag{20}$$

with $\gamma \in \mathbf{R}$, $\tilde{\boldsymbol{\alpha}} \in \mathbf{R}^{n \times m}$, $\boldsymbol{\eta} \in \mathbf{R}^{n \times m}$.

Proof [of Theorem 2] We first note that (18) can eqivalently be rewritten as

$$\begin{aligned} \min_{\boldsymbol{\beta}, \mathbf{w}, \mathbf{b}, \mathbf{s}, \mathbf{t}} \quad & \frac{1}{2} \sum_k \beta_k \|\mathbf{w}_k\|^2 + \sum_i s_i \\ \text{s.t.} \quad & \sum_k \beta_k = 1, \quad \forall k : 0 \leq \beta_k, \\ & \forall i : \forall u \neq y_i : t_{iu} = \sum_k \beta_k \left\langle \mathbf{w}_k, \Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right\rangle + b_{y_i} - b_u, \\ & \forall i : \forall u \neq y_i : s_i \geq \ell_y(t_{iu}). \end{aligned} \tag{21}$$

The Lagrangian of this is given by

$$\begin{aligned} \mathcal{L} = \quad & \frac{1}{2} \sum_k \beta_k \|\mathbf{w}_k\|^2 + \sum_i s_i + \gamma \left(\sum_k \beta_k - 1 \right) - \sum_k \epsilon_k \beta_k \\ & + \sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \left(t_{iu} - \sum_k \beta_k \left\langle \mathbf{w}_k, \Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right\rangle - b_{y_i} + b_u \right) \\ & + \sum_i \sum_{u \neq y_i} \eta_{iu} \left(\ell_y(t_{iu}) - s_i \right), \end{aligned} \tag{22}$$

with Lagrange variables $\tilde{\boldsymbol{\alpha}} \in \mathbf{R}^{m \times \mathcal{Y}}$, $\mathbf{0} \leq \boldsymbol{\epsilon} \in \mathbf{R}^p$, and $\mathbf{0} \leq \boldsymbol{\eta} \in \mathbf{R}^{m \times \mathcal{Y}}$.

We find the stationary points by setting the partial derivatives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k} = \beta_k \mathbf{w}_k - \sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \beta_k \left(\Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right), \tag{23}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_k} = \frac{1}{2} \|\mathbf{w}_k\|^2 - \sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \left\langle \mathbf{w}_k, \Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right\rangle + \gamma - \varepsilon_k, \tag{24}$$

$$\frac{\partial \mathcal{L}}{\partial s_i} = 1 - \sum_{u \neq y_i} \eta_{iu}, \tag{25}$$

$$\frac{\partial \mathcal{L}}{\partial b_v} = - \sum_i \delta_{y_i v} \sum_{u \neq y_i} \tilde{\alpha}_{iu} + \sum_i \sum_{u \neq y_i} \delta_{uv} \tilde{\alpha}_{iu} \tag{26}$$

to zero, yielding:

$$\begin{aligned}
\forall k: \quad \mathbf{w}_k &= \sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \left(\Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right), \\
\forall k: \quad \varepsilon_k &= \gamma - \frac{1}{2}\|\mathbf{w}_k\|^2, \\
\forall i: \quad 1 &= \sum_{u \neq y_i} \eta_{iu}, \\
\forall v: \quad 0 &= \sum_i (1-\delta_{y_i v})\tilde{\alpha}_{iv} - \sum_i \delta_{y_i v} \sum_{u \neq y_i} \tilde{\alpha}_{iu}.
\end{aligned} \tag{27}$$

Plugging these equations back into the Lagrangian, it simplifies considerably, and the problem that remains to be solved is the following:

$$\begin{aligned}
\max_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \min_{\mathbf{t}} \quad & \textstyle\sum_i \sum_{u \neq y_i} \left(\eta_{iu} \ell_y(t_{iu}) + \tilde{\alpha}_{iu} t_{iu} \right) - \gamma, \\
\text{s.t.} \quad & \forall k: \gamma \geq \frac{1}{2}\|\mathbf{w}_k\|^2, \\
& \forall i: \forall u: 0 \leq \eta_{iu}, \\
& \forall i: 1 = \sum_{u \neq y_i} \eta_{iu}, \\
& \forall v: 0 = \sum_i (1-\delta_{y_i v})\tilde{\alpha}_{iv} - \sum_i \delta_{y_i v} \sum_{u \neq y_i} \tilde{\alpha}_{iu} \quad .
\end{aligned} \tag{28}$$

We plug Lemma 1 into (28) and get the optimization problem:

$$\begin{aligned}
& \max_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \min_{\mathbf{t}} \quad \sum_i \sum_{u \neq y_i} \left(\eta_{iu} \ell_y(t_{iu}) + \tilde{\alpha}_{iu} t_{iu} \right) - \gamma \\
\Leftrightarrow \quad & \max_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \quad \sum_i \sum_{u \neq y_i} \inf_{t_{iu}} \left(\eta_{iu} \ell_y(t_{iu}) + \tilde{\alpha}_{iu} t_{iu} \right) - \gamma \\
\Leftrightarrow \quad & \max_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \quad \sum_i \sum_{u \neq y_i} -\eta_{iu} \ell_y^* \left(-\frac{\tilde{\alpha}_{iu}}{\eta_{iu}} \right) - \gamma \\
\Leftrightarrow \quad & \min_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \quad \gamma + \sum_i \sum_{u \neq y_i} \eta_{iu} \ell_y^* \left(-\frac{\tilde{\alpha}_{iu}}{\eta_{iu}} \right)
\end{aligned} \tag{29}$$

which is subject to the same constraints as above. ■

Corollary 3 *When choosing the hinge loss, $\ell(t) := C \max(0, \Delta_{iu} - t)$, the optimum $\mathbf{w}$ of (21) can be computed as*

$$\forall k: \mathbf{w}_k = \sum_i \sum_{u \in \mathcal{Y}} \alpha_{iu} \Phi_k(\mathbf{x}_i, u) \quad , \tag{30}$$

where $\boldsymbol{\alpha} \in \mathbf{R}^{n \times \mathcal{Y}}$ is the solution of the following quadratically constrained linear program:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \gamma + \sum_i \alpha_{iy_i} \\
\text{s.t.} \quad & \forall i: -C \leq \Delta_{iu}\alpha_{iy_i} \leq 0 \\
& \forall i: \forall u \neq y_i: \Delta_{iu}\alpha_{iu} \geq 0 \\
& \forall i: \sum_{u \in \mathcal{Y}} \Delta_{iu}\alpha_{iu} = 0 \\
& \forall u: \sum_i \Delta_{iu}\alpha_{iu} = 0 \\
& \forall k: \gamma \geq \frac{1}{2} \sum_i \sum_j \sum_u \sum_v \alpha_{iu}\alpha_{jv} \left\langle \Phi_k(\mathbf{x}_i, u), \Phi_k(\mathbf{x}_j, v) \right\rangle
\end{aligned} \tag{31}$$

Proof We apply Theorem 2 with the observation that the conjugate function of the margin-scaled hinge loss $\ell_y(t) := C_{iu} \max(0, \Delta_{iu} - t)$ is given by

$$\ell_y^*(\nu) = \begin{cases} \Delta_{iu}\alpha_{iu} & -C_{iu} \leq \nu \leq 0 \\ \infty & \text{otherwise} \end{cases} \quad . \tag{32}$$

This yields

$$\mathbf{w}_k = \sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \left(\Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right). \tag{33}$$

where $\tilde{\alpha}$ is deterimined by

$$\begin{aligned}
\min_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \quad & \gamma - \sum_i \sum_{u \neq y_i} \Delta_{iu} \tilde{\alpha}_{iu} \\
\text{s.t.} \quad & \forall k : \gamma \geq \frac{1}{2} \left(\sum_i \sum_{u \neq y_i} \tilde{\alpha}_{iu} \left(\Phi_k(\mathbf{x}_i, y_i) - \Phi_k(\mathbf{x}_i, u) \right) \right)^2, \\
& \forall i : \forall u \neq y_i : 0 \leq \Delta_{iu} \tilde{\alpha}_{iu} \leq \eta_{iu} C, \\
& \forall i : \forall u \neq y_i : 0 \leq \eta_{iu}, \\
& \forall i : 1 = \sum_{u \neq y_i} \eta_{iu}, \\
& \forall v : 0 = \sum_i (1 - \delta_{y_i v}) \tilde{\alpha}_{iv} - \sum_i \delta_{y_i v} \sum_{u \neq y_i} \tilde{\alpha}_{iu}
\end{aligned} \tag{34}$$

For the next step, we assume $C_{iu} \equiv C$ for all i and u. We note the following equality of constraints, and apply it for every i.

$$\begin{aligned}
& \left\{ (\tilde{\alpha}_{iu})_{u \in \mathcal{Y}} \;\middle|\; \begin{array}{c} \forall u \neq y_i : 0 \leq \Delta_{iu} \tilde{\alpha}_{iu} \leq \eta_{iu} C \\ \forall u \neq y_i : 0 \leq \eta_{iu} \\ 1 = \sum_{u \neq y_i} \eta_{iu} \end{array} \right\} \\
= \quad & \left\{ (\tilde{\alpha}_{iu})_{u \in \mathcal{Y}} \;\middle|\; \begin{array}{c} \forall u \neq y_i : 0 \leq \Delta_{iu} \tilde{\alpha}_{iu} \\ \sum_{u \neq y_i} \Delta_{iu} \tilde{\alpha}_{iu} \leq C \end{array} \right\} \\
= \quad & \left\{ (\alpha_{iu})_{u \in \mathcal{Y}} \;\middle|\; \begin{array}{c} \sum_u \Delta_{iu} \alpha_{iu} = 0 \\ -C \leq \Delta_{i y_i} \alpha_{i y_i} \leq 0 \\ \forall u \neq y_i : \Delta_{iu} \alpha_{iu} \geq 0 \end{array} \right\}
\end{aligned} \tag{35}$$

where the last line is expressed in terms of $\boldsymbol{\alpha} \in \mathbf{R}^{n \times \mathcal{Y}}$ defined by

$$\alpha_{iu} \quad = \quad \begin{cases} \tilde{\alpha}_{iu} & \text{if } u \neq y_i \\ -\sum_{v \neq y_i} \tilde{\alpha}_{iv} & \text{if } u = y_i \end{cases} \tag{36}$$

Plugging both into (33) and (34), we obtain

$$\mathbf{w}_k = \sum_i \left(\left(\sum_{u \neq y_i} \tilde{\alpha}_{iu} \right) \Phi_k(\mathbf{x}_i, y_i) - \sum_{u \neq y_i} \tilde{\alpha}_{iu} \Phi_k(\mathbf{x}_i, u) \right) = \sum_i \sum_u \alpha_{iu} \Phi_k(\mathbf{x}_i, u) \tag{37}$$

and

$$\begin{aligned}
\min_{\boldsymbol{\eta}, \tilde{\boldsymbol{\alpha}}, \gamma} \quad & \gamma - \sum_i \sum_{u \neq y_i} \Delta_{iu} \alpha_{iu} \\
\text{s.t.} \quad & \forall k : \gamma \geq \frac{1}{2} \left(\sum_i \sum_u \alpha_{iu} \Phi_k(\mathbf{x}_i, u) \right)^2, \\
& \forall i : -C \leq \Delta_{i y_i} \alpha_{i y_i} \leq 0 \\
& \forall i : \forall u \neq y_i : \Delta_{iu} \alpha_{iu} \geq 0 \\
& \forall i : \sum_u \Delta_{iu} \alpha_{iu} = 0, \\
& \forall u : \sum_i \Delta_{iu} \alpha_{iu} = 0
\end{aligned} \tag{38}$$

■

Theorem 4 *The QCLP multiclass multiple kernel learning problem defined by Equation* (13)*, is equivalent to the SILP*

$$\begin{aligned}
\max_{\boldsymbol{\beta}} \quad & \theta \\
\text{s.t.} \quad & \forall \boldsymbol{\alpha} \in \mathcal{S}: \;\; \theta \leq \frac{1}{2} \sum_k \beta_k \left\| \mathbf{w}_k(\boldsymbol{\alpha}) \right\|^2 + \sum_i \alpha_{iy_i} \\
& \forall k: \;\; 0 \leq \beta_k \\
& \sum_{k=1}^{p} \beta_k = 1 \;\; ,
\end{aligned} \tag{39}$$

where

$$\mathcal{S} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \forall i : -C \leq \alpha_{iy_i} \leq 0 \\ \forall i : \forall u \neq y_i : \alpha_{iu} \geq 0 \\ \forall i : \sum_{u \in \mathcal{Y}} \alpha_{iu} = 0 \\ \forall u \in \mathcal{Y} : \sum_i \alpha_{iu} = 0 \end{array} \right\} \tag{40}$$

is the set of admissable parametrizations for the first constraint.

Proof We partially dualise Equation (13) with respect to γ while keeping all the other constraints. Considering only the objective and the constraint containing γ, we have the Lagrangian

$$\mathcal{L}(\gamma, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \gamma - \sum_{i=1}^{n} \alpha_{iy_i} + \sum_{k=1}^{p} \beta_k \left(\frac{1}{2} \|w_k\|^2 - \gamma \right),$$

where $\boldsymbol{\beta} \geq 0$. Differentiating this with respect to γ, we obtain

$$\frac{\partial \mathcal{L}}{\partial \gamma} = 1 - \sum_{k=1}^{p} \beta_k .$$

Setting this to zero, we obtain the result. ■

References

[1] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple Kernel Learning, Conic Duality, and the SMO Algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[2] Olivier Bousquet and Daniel J.L. Herrmann. On the complexity of learning the kernel matrix. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 399–406, Cambridge, MA, 2003. MIT Press.

[3] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] O. Chapelle and A. Zien. Semi-Supervised Classification by Low Density Separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

[5] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20:273–297, 1995.

[6] Koby Crammer, Joseph Keshet, and Yoram Singer. Kernel design using boosting. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 537–544, Cambridge, MA, 2003. MIT Press.

[7] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[8] Qinghua Cui, Tianzi Jiang, Bing Liu, and Songde Ma. Esub8: A novel tool to predict protein subcellular localizations in eukaryotic organisms. *BMC Bioinformatics*, 5(66), 2004.

[9] K. Duan and S.S. Keerthi. Which is the best multiclass SVM method? An empirical study. Technical report, National University of Singapore, 2003. submitted to NIPS 03.

[10] Olof Emanuelsson, Henrik Nielsen, Søren Brunak, and Gunnar von Heijne. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of Molecular Biology*, 300:1005–1016, 2000.

[11] J. L. Gardy, M. R. Laird, F. Chen, S. Rey, C. J. Walsh, M. Ester, and F. S. L. Brinkman. PSORTb v.2.0: expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis. *Bioinfomatics*, 21:617–623, 2004.

[12] Aarti Garg, Manoj Bhasin, and Gajendra P.S. Raghava. Support vector machine-based method for subcellular localization of human proteins using amino acid composition, their order, and similarity search. *The Journal of Biological Chemistry*, 280(15):14427–14432, 2005.

[13] Chittibabu Guda and Shankar Subramaniam. TARGET: a new method for predicting protein subcellular localization in eukaryotes. *Bioinformatics*, 21(21):3963–3969, 2005.

[14] M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In R. Cowell Ghahramani, Z., editor, *Proceedings of AISTATS 2005*, pages 136–143, 2005.

[15] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, pages 10915–10919, 1992.

[16] R. Hettich and K. O. Kortanek. Semi-Infinite Programming: Theory, Methods, and Applications. *SIAM Review*, 35(3):380–429, September 1993.

[17] Annette Höglund, Pierre Dönnes, Torsten Blum, Hans-Werner Adolph, and Oliver Kohlbacher. MultiLoc: prediction of protein subcellular localization using N-terminal targeting sequences, sequence motifs, and amino acid composition. *Bioinfomatics*, 2006.

[18] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[19] Gert Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[20] Gert R. G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinfomatics*, 20(16):2626–2635, 2004.

[21] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

[22] K. J. Park and M. Kanehisa. Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics*, 19(13):1656–1663, Sep 2003.

[23] P. Pavlidis, J. Weston, J. Cai, and W. S. Noble. Learning gene functional classifications from multiple data types. *Journal of Computational Biology*, 9(2):401–411, 2002.

[24] A. Reinhardt and T. Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26:2230–2236, 1998.

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

[26] S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable SVMs for biological sequence classification. In S. Miyano, J. Mesirov, S. Kasif, S. Istrail, P. Pevzner, and M. Waterman, editors, *Research in Computational Molecular Biology*, pages 389–407. Springer Verlag, 2005.

[27] Sören Sonnenburg, Gunnar Rätsch, and Christin Schäfer. A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processings Systems*, 2005.

[28] B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *Neural Information Processing Systems Conference (NIPS03)*, December 2003.

[29] Ioannis Tsochantaridis, Thomas Hormann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and sturcutured output spaces. In *Proceedings of the 16th International Conference on Machine Learning*, 2004.

[30] Güldener U, Münsterkötter M, Kastenmüller G, Strack N, van Helden J, Lemer C, Richelles J, Wodak SJ, Garcia-Martinez J, Perez-Ortin JE, Michael H, Kaps A, Talla E, Dujon B, Andre B, Souciet JL, De Montigny J, Bon E, Gaillardin C, and Mewes HW. CYGD: the comprehensive yeast genome database. *Nucleic Acids Research*, 33:D364–368, 2005. Database issue.

[31] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *European Symposium on Artificial Neural Networks.*, 1999.

[32] Dan Xie, Ao Li, Minghui Wang, Zhewen Fan, and Huanqing Feng. LOCSVMPSI: a web server for subcellular localization of eukaryotic proteins using SVM and profile of PSI-BLAST. *Nucleic Acids Research*, 33:W105–W110, 2005.

[33] Chin-Sheng Yu, Chih-Jen Lin, and Jenn-Kang Hwang. Predicting subcellular localization of proteins for gram-negative bacteria by support vector machines based on n-peptide compositions. *Protein Science*, 13:1402–1406, 2004.