Algorithms for the Hard Pre-Image Problem of String Kernels and the General Problem of String Prediction

Sébastien Giguère [1,2*] GIGUERE.SEBASTIEN@GMAIL.COM
Amélie Rolland [2*] AMELIE.ROLLAND.1@ULAVAL.CA
François Laviolette [2] FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA
Mario Marchand [2] MARIO.MARCHAND@IFT.ULAVAL.CA

[1] Institute for Research in Immunology and Cancer, University of Montreal, Montreal, Canada
[2] Department of Computer Science and Software Engineering, Laval University, Quebec, Canada
[*] These authors contributed equally to this work.

Abstract

We address the pre-image problem encountered in structured output prediction and the one of finding a string maximizing the prediction function of various kernel-based classifiers and regressors. We demonstrate that these problems reduce to a common combinatorial problem valid for many string kernels. For this problem, we propose an upper bound on the prediction function which has low computational complexity and which can be used in a branch and bound search algorithm to obtain optimal solutions. We also show that for many string kernels, the complexity of the problem increases significantly when the kernel is normalized.

On the optical word recognition task, the exact solution of the pre-image problem is shown to significantly improve the prediction accuracy in comparison with an approximation found by the best known heuristic. On the task of finding a string maximizing the prediction function of kernel-based classifiers and regressors, we highlight that existing methods can be biased toward long strings that contain many repeated symbols. We demonstrate that this bias is removed when using normalized kernels. Finally, we present results for the discovery of lead compounds in drug discovery. The source code can be found at https://github.com/a-ro/preimage.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

1. Introduction

This work addresses two combinatorial problems related to string prediction. For both problems, let $\mathcal{A}$ be the set of all symbols from an alphabet and $\mathcal{A}^*$ be the set of all possible strings of symbols from that alphabet.

1.1. Structured Output Pre-image

In the structured output prediction framework, the learner has access to a set $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\} \in \mathcal{X} \times \mathcal{Y}$ of input-output pairs. The input space $\mathcal{X}$ is arbitrary but we assume that the output space $\mathcal{Y}$ is the set $\mathcal{A}^*$ of strings from an alphabet $\mathcal{A}$. We assume the existence of an input feature map $\boldsymbol{\phi}_{\mathcal{X}} : \mathcal{X} \to \mathcal{H}_{\mathcal{X}}$ and an output feature map $\boldsymbol{\phi}_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{H}_{\mathcal{Y}}$, where both $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$ are high-dimensional vector spaces and, more generally, RKHS. The input kernel $K_{\mathcal{X}} : \mathcal{X}^2 \to \mathbb{R}$ is defined by the inner product between vectors in $\mathcal{H}_{\mathcal{X}}$, *i.e.*, $K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}), \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}') \rangle \; \forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2$. A similar definition holds for the output kernel $K_{\mathcal{Y}}$.

We consider predictors that are linear operators $\mathbf{W} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{Y}}$. Given any such $\mathbf{W}$, and any $\mathbf{x} \in \mathcal{X}$, the predicted output $\mathbf{y}^{\mathbf{W}}(\mathbf{x})$ of $\mathbf{W}$ on input $\mathbf{x}$ is given by solving the following *structured output pre-image problem*

$$\mathbf{y}^{\mathbf{W}}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} \left\| \boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}) - \mathbf{W} \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}) \right\| , \qquad (1)$$

where $\| \cdot \|$ denotes the L_2 norm in $\mathcal{H}_{\mathcal{Y}}$. Hence, the pre-image problem arises when trying to reconstruct the output from the output feature vector predicted by $\mathbf{W}$. A pre-image is said to be exact when there exists $\mathbf{y} \in \mathcal{Y}$ for which its feature vector $\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})$ is exactly equal to $\mathbf{W} \boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$. Unfortunately, an exact pre-image rarely exists for linear operators obtained by vector-valued ridge regression. Hence, in practice, we have to deal with a hard pre-image problem.

Let us consider normalized output feature vectors. Hence, given any $\boldsymbol{\phi}_{\mathcal{Y}}$, let us use $\widehat{\boldsymbol{\phi}}_{\mathcal{Y}}(\mathbf{y}) \stackrel{\text{def}}{=} \frac{\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})}{\|\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})\|}$. Then, equa-

tion (1) becomes

$$\mathbf{y}^{\mathbf{W}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \widehat{\boldsymbol{\phi}}_{\mathcal{Y}}(\mathbf{y}), \mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})\rangle \,. \tag{2}$$

Given any linear operator $\mathbf{W}$ obtained by vector-valued ridge regression, we have $\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(x) = \sum_{i=1}^{m}\sum_{j=1}^{m} \widehat{\boldsymbol{\phi}}_{\mathcal{Y}}(y_i)A_{i,j}K_{\mathcal{X}}(x_j, x)$ for some $m \times m$ matrix $\mathbf{A}$ (Cortes et al., 2007). In that case, $\mathbf{y}^{\mathbf{W}}(\mathbf{x})$ becomes

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y})}{\sqrt{K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y}_i)K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})}} A_{i,j}K_{\mathcal{X}}(\mathbf{x}_j, \mathbf{x}) \,. \tag{3}$$

As we will see, this paper presents a branch and bound algorithm for solving this combinatorial problem when the output kernel $K_{\mathcal{Y}}$ belongs to some family of string kernels.

1.2. String prediction for Classification and Regression

The second problem addressed in this paper consists of finding the string maximizing the prediction function of kernel-based classifiers and regressors. For example, finding the string maximizing the prediction function of a support vector machine consists of finding the positively-labeled string at greatest distance from the separating hyperplane. Alternatively, the string maximizing the prediction function of a kernel ridge regressor is the string achieving the greatest predicted real value.

In that second framework, the learner has access to a set $\mathcal{S} = \{(\mathbf{y}_1, r_1), \ldots, (\mathbf{y}_m, r_m)\}$ of m training examples where $\mathbf{y}_i$ is a string and r_i is a scalar label. For example, $\mathbf{y}_i$ could be a peptide (a small sequence of amino acids) and r_i could quantify its ability to bind to a certain protein or to inhibit some biological process. Alternatively, r_i could represent one of two classes, namely $+1$ or -1. For both cases, the first step is to build a real-valued prediction function h whose value on any input $\mathbf{y}$ is given by

$$h(\mathbf{y}) = \sum_{i=1}^{m} \alpha_i K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y}) \,, \tag{4}$$

where the weight vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ is obtained by minimizing an objective function (such as those for support vector machines or kernel ridge regression), and where $K_{\mathcal{Y}}$ is assumed to be a string kernel. In the regression case, $h(\mathbf{y})$ is the predicted real-valued label for input $\mathbf{y}$. In the classification case, the predicted class on input $\mathbf{y}$ is given by $\operatorname{sgn}(h(\mathbf{y}))$. In both cases, given a predictor, we are interested at finding the input $\mathbf{y}$ maximizing the value of the prediction function $h(\mathbf{y})$, *i.e.*, when the inputs are strings of length ℓ, we want to solve

$$\mathbf{y}^{h} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{A}^{\ell}} h(\mathbf{y}) \,. \tag{5}$$

$\mathbf{y}$	$\mathbf{y}'$	$K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}')$	$\frac{K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}')}{\sqrt{K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})K_{\mathcal{Y}}(\mathbf{y}', \mathbf{y}')}}$
AAB	AAB	5	**1**
AAA	AAB	**6**	0.89

Table 1. Example using the 1-gram kernel where $K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}') > K_{\mathcal{Y}}(\mathbf{y}', \mathbf{y}')$ when $\mathbf{y} \neq \mathbf{y}'$. Here, $||\phi_{\mathcal{Y}}(\texttt{AAB})|| = \sqrt{5}$ and $||\phi_{\mathcal{Y}}(\texttt{AAA})|| = \sqrt{9}$.

Recently, an algorithm was proposed (Giguère et al., 2015) to solve this problem in polynomial time for any prediction function h in the form of Equation (4) when $K_{\mathcal{Y}}$ belongs to a family of unnormalized string kernels known as the Generic String (GS) kernel. Their approach consists of mapping this combinatorial problem to the problem of finding the longest (weighted) path in a directed acyclic graph. Because the graph is acyclic, this problem is solved by dynamic programming in $O(\ell|\mathcal{A}|^{n+1})$ time, where n is the maximum sub-string size considered by the GS kernel.

However, there exists an important problem for most string kernels which can bias the solution of Equation (5) in an undesirable way. Given two strings $\mathbf{y}$ and $\mathbf{y}'$ of the same length, the Euclidean norms of the feature vectors $\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})$ and $\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}')$ induced by an unnormalized string kernel can differ substantially, i.e. we can have $\sqrt{K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})} \gg \sqrt{K_{\mathcal{Y}}(\mathbf{y}', \mathbf{y}')}$ even if $\mathbf{y}$ and $\mathbf{y}'$ have the same length. Consider, for example, two strings AAAAA and ABCDE and the n-gram kernel with $n = 2$ (also known as the Spectrum kernel (Leslie et al., 2002)). String AAAAA has a norm of $\sqrt{4^2} = 4$, while ABCDE has a norm of $\sqrt{1^2 + 1^2 + 1^2 + 1^2} = 2$. Hence the norm of the feature vector induced by this kernel is sensitive to repetitions in the string. The norm is also influenced by the length of the string. This problem is shared by most string kernels that are based on n-gram counts or comparisons.

Note that we can also have $K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}') > K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})$ while $\mathbf{y} \neq \mathbf{y}'$. Table 1 shows a simple example when this happens. Consequently, $h(\mathbf{y})$ depends on the norm of $\boldsymbol{\phi}(\mathbf{y})$ and $\mathbf{y}^h$ is biased toward strings having a feature vector of large norm, for example, a long string having many n-gram repetitions. For most applications, this is not a desirable bias which can be removed by normalizing the kernel. In this case, the prediction function becomes

$$\widehat{h}(\mathbf{y}) = \sum_{i=1}^{m} \alpha_i \frac{K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y})}{\sqrt{K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y}_i)K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})}} \,. \tag{6}$$

Consequently, we are then interested at solving

$$\mathbf{y}^{\widehat{h}} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{A}^*} \widehat{h}(\mathbf{y}) \,. \tag{7}$$

We will see that, despite its similarity with optimization problem (5), the increase in computational complexity of problem (7) is striking.

1.3. Unified Optimization Problem

An important observation is that both Equation (7) and Equation (3) reduce to solving

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{A}^*}{\operatorname{argmax}} \frac{1}{\sqrt{K_\mathcal{Y}(\mathbf{y},\mathbf{y})}} \sum_{i=1}^{m} \beta_i K_\mathcal{Y}(\mathbf{y}_i,\mathbf{y}) \,, \quad (8)$$

where $\beta_i = \frac{\alpha_i}{\sqrt{K_\mathcal{Y}(\mathbf{y}_i,\mathbf{y}_i)}}$ for solving Equation (7), and $\beta_i = \beta_i(\mathbf{x}) = \sum_{j=1}^{m} \frac{A_{i,j}}{\sqrt{K_\mathcal{Y}(\mathbf{y}_i,\mathbf{y}_i)}} K_\mathcal{X}(\mathbf{x}_j,\mathbf{x})$ for solving Equation (3). Observe that for the pre-image problem, this makes the problem independent of the input kernel $K_\mathcal{X}$. Hence, the rest of this paper will focus on solving this unified problem defined by Equation (8).

The computational complexity of Problem (8) depends on the choice of string kernel for $K_\mathcal{Y}$. The use of the Generic String (GS) kernel (Giguère et al., 2013) is appealing since, depending on the chosen hyper-parameters, this kernel can be specialized to eight different kernels; namely, the Hamming kernel, the Blended Spectrum (Shawe-Taylor & Cristianini, 2004), the Radial Basis Function (RBF), the Oligo (Meinicke et al., 2004), and the Weighted degree (Rätsch & Sonnenburg, 2004). Hence, any advance in solving Equation (8) would also be applicable to these eight string kernels. The GS kernel was originally proposed for strings of amino acids by comparing them using their physico-chemical properties. Although we present the GS kernel in a bioinformatics context, the concept of similarity between the elements of a sequence is not limited to this field. For example, one could use phoneme similarity in speech recognition, words similarity in part of speech tagging or synonymous words in machine translation.

Given any pair $(\mathbf{y},\mathbf{y}')$ of strings, the value of the GS kernel is defined as

$$GS(\mathbf{y},\mathbf{y}',n,\sigma_p,\sigma_c) \stackrel{\text{def}}{=} \sum_{l=1}^{n} \sum_{i=0}^{|\mathbf{y}|-l} \sum_{j=0}^{|\mathbf{y}'|-l} \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-\|\boldsymbol{\psi}^l(y_{i+1},..,y_{i+l}) - \boldsymbol{\psi}^l(y'_{j+1},..,y'_{j+l})\|^2}{2\sigma_c^2}\right), \quad (9)$$

where n controls the maximum length of the compared l-grams, $\boldsymbol{\psi}^l : \mathcal{A}^l \to \mathbb{R}^{ld}$ encodes the physico-chemical properties of l-grams by mapping each of the l amino acids to a real-valued vector containing d components, σ_c controls the penalty incurred when the physico-chemical properties of two l-grams differ, and σ_p controls the penalty incurred when two l-grams are not sharing the same position in their respective strings.

Our ability to efficiently solve the Problem (8) could contribute to many discoveries. If $\widehat{h}$ predicts the binding affinity of a peptide to some protein involved in a certain disease, finding the peptide $\mathbf{y}^{\widehat{h}}$ having the highest predicted binding affinity could have significant impact on our ability to design new drugs. Alternatively, if $\widehat{h}$ predicts the anti-microbial activity of a peptide, finding the peptide $\mathbf{y}^{\widehat{h}}$ having the greatest predicted anti-microbial activity could be a valuable tool in the fight of new infectious diseases.

Most structured output prediction algorithms need to solve the pre-image for inference, some also during the training phase. The the accuracy of these algorithms will improve with a faster and more accurate pre-image solver.

In the next section, we present low computational complexity upper bounds on Equation (8). The bounds are later used in a branch and bound search to obtain exact solutions to the unified string prediction problem (8).

2. Method

The computational complexity of Problem (8) depends on the values of the parameters defining the GS kernel. Parameters σ_c and σ_p each control the variance of one Gaussian function. Whenever one of these parameters equals 0, the related Gaussian function becomes a Dirac delta function. These simpler cases allow for algorithmic optimizations and tighter bounds. For that reason, we consider separately three different cases; from the easiest to the hardest.

2.1. Case when $\sigma_p = 0$

In that case, $\exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right)$ is the identity function and is equal to 1 if $i = j$ and 0 otherwise. Recall that i and j are the positions of the l-grams in the strings $\mathbf{y}$ and $\mathbf{y}'$. Hence, only l-grams that are at the same position contribute to the GS kernel of Equation (9). We recover the Hamming kernel if ($\sigma_c = 0$ and $n = 1$), the Weighted Degree kernel (Rätsch & Sonnenburg, 2004) if ($\sigma_c = 0$ and $n > 0$), and, finally, a variant of the Weighted Degree (Toussaint et al., 2010) if ($\sigma_c > 0$ and $n > 0$).

All these kernels have the property that $K_\mathcal{Y}(\mathbf{y},\mathbf{y})$ is a constant c for all strings $\mathbf{y}$ of the same length. When searching for a string $\mathbf{y}$ having a fixed length ℓ, Equation (8) becomes

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{A}^\ell}{\operatorname{argmax}} \frac{1}{\sqrt{c}} \sum_{i=1}^{m} \beta_i K_\mathcal{Y}(\mathbf{y}_i,\mathbf{y}) \,, \quad (10)$$

with β_i defined as in Equation (8). In that case, we recover the problem studied by Giguère et al. (2015). Their approach uses a graph, similar to a De Bruijn graph, that has a source and a sink node. The graph has three major properties. First, a single source-sink path is associated to every possible string in $\mathcal{A}^\ell$. Next, the number of edges and arcs in the graph are dependant of $|\mathcal{A}^n|$, not $|\mathcal{A}^\ell|$. Finally, the length (or weight) of a path (a string) is

exactly $\sum_{i=1}^{m} \beta_i K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y})$. Consequently, the solution $\mathbf{y}^\star$ of Equation (8) is given by the longest path which can be found by dynamic programming since the graph is acyclic. For a string of unknown length, the procedure can be repeated and the string achieving the highest score is kept.

2.2. Case when $\sigma_p > 0$ and $\sigma_c = 0$

When $\sigma_p > 0$, the optimization problem becomes harder because $K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})$ is no longer constant for all $\mathbf{y} \in \mathcal{A}^\ell$. However, when $\sigma_c = 0$, $\exp\left(\frac{-\|\boldsymbol{\psi}^l(s) - \boldsymbol{\psi}^l(s')\|^2}{2\sigma_c^2}\right)$ in turn becomes the identity function and is equal to 1 if the l-grams s and s' are identical and 0 otherwise. For that reason, we recover the well known Blended Spectrum kernel (Shawe-Taylor & Cristianini, 2004) if $\sigma_p = \infty$ and the n-gram kernel if, in addition, we keep only the term $l = n$ in the summation over l in Equation (9). When σ_p is finite, we recover the Oligo kernel (Meinicke et al., 2004).

In these cases, the approach of Giguère et al. (2015) is no longer guaranteed to find the optimal solution of Problem (8). However, it can still be used to compute an upper bound on partial solutions in a branch and bound search.

Let us now describe the approach of Cortes et al. (2007) for the n-gram pre-image to which we will compare later. Then, we will explain the branch and bound algorithm we used. Next, we will demonstrate how a tight bound can be obtained when $\sigma_c = 0$. Later, in Section 2.3, we present a similar but looser bound for the case when $\sigma_c > 0$.

2.2.1. Eulerian Path Heuristic

Cortes et al. (2007) solved the exact pre-image problem of the n-gram kernel by finding an Eulerian circuit in a graph. To find $\mathbf{y}^{\mathbf{w}}(\mathbf{x})$, the components of the predicted feature vector $\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$ are first rounded to obtain a vector $\mathbf{z}$ of integer values. Each component of $\mathbf{z}$ should represent the number of times each of the possible n-grams appears in the string $\mathbf{y}^{\mathbf{w}}(\mathbf{x})$. Then, they define a graph $G_{\mathbf{z},n}$, similar to a De Bruijn graph, composed of a vertex for each possible $(n-1)$-gram in $\mathcal{A}^{n-1}$. Next, the number of edges between the vertex $a_1, .., a_{n-1}$ and the vertex $a_2, .., a_n$ is given by the count of the n-gram $a_1, .., a_n$ in $\mathbf{z}$. Finally, they predict a string by constructing the Eulerian circuit in $G_{\mathbf{z},n}$ (assuming that it exists). However, the exact pre-image does not exist when some components of $\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$ are not integers. Also, the rounded integer-valued vector $\mathbf{z}$ might not give an Eulerian graph. When facing this problem, the authors proposed to merge multiple paths together. There is thus no guarantee on the optimality of the predicted string. Finally, this heuristic is in $O(\ell + |\mathcal{A}|^n)$ if an Eulerian circuit exists.

2.2.2. Branch and Bound

A branch and bound algorithm starts by dividing the search space into disjoint subspaces. For example, one subspace could be all strings ending with the string `DE`. For a maximization problem, an upper bound on the best achievable solution is computed for each subspace. Then, the subspace with the highest upper bound is further divided. Finally, the search in a subspace stops when the subspace can no longer be divided (a leaf is reached in the search tree), or when the upper bound value is lower than the value of an already achieved solution (*i.e.*, an already reached leaf in the search tree). A branch and bound approach can thus avoid exploring a large part of the search space.

The search algorithm used here differ slightly from a standard branch and bound. It alternates between a branch and bound phase and a greedy phase. The later is important to ensure that leaves of the search tree are quickly visited. This allows good but sub-optimal solutions to be returned by the algorithm if the allowed computational time expires. Whenever a node is visited, the bound is computed for all its children and they are added to a priority queue accordingly. This greedy process is repeated until a leaf is reached. Then, the node with the largest bound is visited and the greedy process starts again. At all time, the best solution found so far is kept and the search stops when the bound of the node on top of the priority queue is smaller than the value of the best solution. The search algorithm is detailed in the Supplementary Material.

2.2.3. Upper bound when $\sigma_p > 0$ and $\sigma_c = 0$

An upper bound for a maximization problem takes as input a partial solution and returns an upper bound of the maximum achievable with that partial solution. In our setting, a partial solution $\mathbf{y}' = y'_1, .., y'_p$ is the last p characters of a possibly longer string of ℓ characters: $y_1, .., y_{(\ell-p)}, y'_1, .., y'_p$. That way, we can define $\mathcal{A}^{\ell-p} \times \{\mathbf{y}'\}$ as the set of all possible strings of length ℓ ending with $y'_1, \ldots, y'_p$. Our goal is to have a function F that upper bounds Equation (8) for every string in $\mathcal{A}^{\ell-p} \times \{\mathbf{y}'\}$. In other words,

$$F(\mathbf{y}', \ell) \geq \max_{\mathbf{y} \in \mathcal{A}^{\ell-p} \times \{\mathbf{y}'\}} \frac{1}{\sqrt{K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})}} \sum_{i=1}^{m} \beta_i K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y})\,. \tag{11}$$

To do so, let $F(\mathbf{y}', \ell) \stackrel{\text{def}}{=} \frac{1}{\sqrt{f(\mathbf{y}', \ell)}} g(\mathbf{y}', \ell)$, where

$$f(\mathbf{y}', \ell) \leq \min_{\mathbf{y} \in \mathcal{A}^{\ell-p} \times \{\mathbf{y}'\}} K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}) \tag{12}$$

and

$$g(\mathbf{y}', \ell) \geq \max_{\mathbf{y} \in \mathcal{A}^{\ell-p} \times \{\mathbf{y}'\}} \sum_{i=1}^{m} \beta_i K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y})\,. \tag{13}$$

Observe that the right hand side of the upper bound $g(\mathbf{y}', \ell)$ is the same optimization problem as in Equation (5). For that reason, the approach of (Giguère et al., 2015) can be used for the computation of that bound. Their approach uses a weight table $W_{|\mathcal{A}|^n \times (\ell-n+1)}$ and a dynamic programming table $T_{|\mathcal{A}|^n \times (\ell-n+1)}$ to compute the longest path in a graph and it is relatively easy to modify their algorithm to return the tables instead of the solution. In that way, given a string with suffix $\mathbf{y}'$, it is possible to determine, by accessing T, the value of the prefix from $\mathcal{A}^{\ell-p}$ maximizing the right hand side of Equation (13). This value is given by the row corresponding to the n-gram $\mathbf{y}'_1, ..., \mathbf{y}'_n$ and the column $\ell - |\mathbf{y}'|$ in T. To this we add the weights of all the n-grams of $\mathbf{y}'_2, ..., \mathbf{y}'_p$, respectively located at the columns $\ell - |\mathbf{y}'| + 1$ to $\ell - n$ in W. Thus, the algorithm in $\mathcal{O}((\ell - n)|\mathcal{A}|^{n+1})$ only needs to be executed once, before the branch and bound search, to obtain the value of $g(\mathbf{y}', \ell)$ in constant time for any $\mathbf{y}'$. Finally, g is the smallest possible upper bound and is exact since there always exists a string $\mathbf{y} \in \mathcal{A}^{\ell-p} \times \{\mathbf{y}'\}$ with $g(\mathbf{y}, \ell) = \sum_{i=1}^{m} \beta_i K_{\mathcal{Y}}(\mathbf{y}_i, \mathbf{y})$.

Let us demonstrate how to obtain the lower bound $f(\mathbf{y}', \ell)$ when $K_{\mathcal{Y}}$ is the GS kernel with hyper-parameters n , σ_p and σ_c. The bound is defined as follows (more details are given in Supplementary Material):

$$f(\mathbf{y}', \ell) \stackrel{\text{def}}{=} GS(\mathbf{y}', \mathbf{y}', n, \sigma_p, \sigma_c) + 2YY'(\mathbf{y}', \ell, n, \sigma_p, \sigma_c) + YY(\ell - |\mathbf{y}'|, n, \sigma_p, \sigma_c)\,, \quad (14)$$

where

$$YY'(\mathbf{y}', \ell, n, \sigma_p, \sigma_c) = \sum_{l=1}^{n} \sum_{i=0}^{\ell-|\mathbf{y}'|-1} \min_{\mathbf{y} \in \mathcal{A}^l} \sum_{j=0}^{|\mathbf{y}'|-l} \exp\left(\frac{-(i-j+|\mathbf{y}'|-\ell)^2}{2\sigma_p^2}\right) \times I(y_1, .., y_l = y'_{j+1}, .., y'_{j+l})\,, \quad (15)$$

$$YY(d, n, \sigma_p, \sigma_c) = \sum_{l=1}^{n} \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} S(i, j, l, d, \sigma_p, \sigma_c), \quad (16)$$

$$S(i, j, l, d, \sigma_p, \sigma_c) = \begin{cases} 1 & \text{if } i = j, \\ \exp\left(\frac{-d^2}{2\sigma_p^2}\right) & \text{if } |i-j| \in \{|\mathcal{A}|^l, 2|\mathcal{A}|^l, 3|\mathcal{A}|^l, ...\}, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that when $|\mathcal{A}| > \ell$, YY' is zero and YY is $n(\ell - |\mathbf{y}'|)$. In contrast with g, the lower bound f is not always attained since it can underestimate the value of the string $\mathbf{y} \in \mathcal{A}^{\ell-p} \times \{y'_1, \ldots, y'_p\}$ minimizing $K_{\mathcal{Y}}(\mathbf{y}, \mathbf{y})$.

2.3. Case when $\sigma_p > 0$ and $\sigma_c > 0$

When $\sigma_p > 0$ and $\sigma_c > 0$, the functions YY' and S are modified to take in account the similarity between the n-grams. In that case,

$$YY'(\mathbf{y}', \ell, n, \sigma_p, \sigma_c) = \sum_{l=1}^{n} \sum_{i=0}^{\ell-|\mathbf{y}'|-1} \min_{\mathbf{y} \in \mathcal{A}^l} \sum_{j=0}^{|\mathbf{y}'|-l} \exp\left(\frac{-(i-j+|\mathbf{y}'|-\ell)^2}{2\sigma_p^2}\right) \times \exp\left(\frac{-\|\boldsymbol{\psi}^l(y_1,..,y_l) - \boldsymbol{\psi}^l(y'_{j+1},..,y'_{j+l})\|^2}{2\sigma_c^2}\right), \quad (17)$$

$$S(i, j, l, d, \sigma_p, \sigma_c) = \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-l(D(i,j))}{2\sigma_c^2}\right)$$

and

$$D(i, j) = \begin{cases} 0 & \text{if } i = j, \\ \max\limits_{(a,a') \in \mathcal{A}^2} ||\boldsymbol{\psi}(a) - \boldsymbol{\psi}(a')||^2 & \text{otherwise.} \end{cases}$$

The computational complexity of YY is thus unchanged since the computation to identify the couple (a, a') only has to be done once for an alphabet.

3. Results and Discussion

We report the results of our approach on two distinct problems. The first one corresponds to the structured output pre-image problem and consists of predicting the word associated to an image representation. The second one corresponds to the prediction function maximization problem and consists of finding the peptide with the greatest predicted bioactivity value.

3.1. Pre-image string prediction

In the first set of experiments, we compared our approach with the one proposed by Cortes et al. (2007) to solve the pre-image problem on the optical word recognition task (Taskar et al., 2004). This task consists of predicting the handwritten word contained in a binary pixel image. We used the same 10 folds division as (Taskar et al., 2004), where each fold has approximately 600 training and 5500 testing examples. The performances were measured according to the percentage of errors on words (0/1 risk), the percentage of errors on letters (letter risk), and the percentage of character edits (insertions, deletions, substitutions) required to change the predicted words into the correct words (Levenshtein risk). The results were averaged over the 10 folds.

For both approaches, we used multiple-output ridge regression (Cortes et al., 2007) (with parameter λ) as the learning algorithm and the polynomial kernel of degree d. The parameters λ and d were chosen using standard cross-validation in each fold. The parameters σ_p and n of the GS kernel were fixed to obtain the Hamming ($\sigma_p = 0$, $n = 1$), the Weighted degree ($\sigma_p = 0$, $n = 2, 3$), and the n-gram

	0/1 risk	Letter risk	Levenshtein risk
$\mathbf{y}^\star_{\text{Hamming}}$	8.10 $\pm.62$	4.97 $\pm.56$	4.95 $\pm.55$
$\mathbf{y}^\star_{\text{Weighted Degree-2}}$	5.77 ±1.1	3.80 $\pm.81$	3.78 $\pm.79$
$\mathbf{y}^\star_{\text{Weighted Degree-3}}$	**5.19** $\pm.91$	**3.75** $\pm.77$	**3.73** $\pm.76$
$\mathbf{y}^\star_{\text{GS}}$	5.22 $\pm.93$	3.78 $\pm.76$	3.75 $\pm.75$
$\mathbf{y}^\star_{\text{2-gram}}$	18.82 $\pm.72$	12.38 $\pm.54$	9.76 $\pm.50$
$\text{Eulerian}_{\text{2-gram}}$	20.86 $\pm.67$	13.66 $\pm.43$	10.93 $\pm.45$
$\mathbf{y}^\star_{\text{3-gram}}$	6.18 ±1.2	4.26 $\pm.80$	4.17 $\pm.85$
$\text{Eulerian}_{\text{3-gram}}$	6.95 $\pm.80$	4.84 $\pm.71$	4.59 $\pm.69$

Table 2. Empirical results on the optical word recognition task when $|\mathbf{y}|$ is known at prediction time.

	0/1 risk	Levenshtein risk
$\mathbf{y}^\star_{\text{Hamming}}$	8.25 $\pm.63$	5.15 $\pm.54$
$\mathbf{y}^\star_{\text{Weighted Degree-2}}$	6.01 ±1.1	**3.78** $\pm.60$
$\mathbf{y}^\star_{\text{Weighted Degree-3}}$	**5.46** $\pm.95$	3.79 $\pm.58$
$\mathbf{y}^\star_{\text{GS}}$	5.47 $\pm.96$	3.93 $\pm.73$
$\mathbf{y}^\star_{\text{2-gram}}$	22.29 $\pm.85$	11.67 $\pm.42$
$\text{Eulerian}_{\text{2-gram}}$	64.11 ±2.0	32.62 ±1.6
$\mathbf{y}^\star_{\text{3-gram}}$	9.07 $\pm.65$	6.01 $\pm.50$
$\text{Eulerian}_{\text{3-gram}}$	34.76 ±2.5	17.25 ±1.2

Table 3. Empirical results on the optical word recognition task when $|\mathbf{y}|$ is unknown at prediction time.

($\sigma_p = \infty$, $n = 2, 3$) kernels. Also, to obtain the n-gram kernel, we fixed $l = n$ in the summation over l in Equations (9), (15), and (16). Otherwise, substrings of length 1 to n would contribute to the kernel, in which case, this is known as the Blended Spectrum kernel. Finally, we also present results obtained using the GS kernel where σ_c, σ_p and n are chosen by cross-validation. Note that for all experiments, the branch and bound search was limited to 30 seconds by prediction.

The pre-image heuristic of Cortes et al. (2007) is based on Eulerian paths and is only valid for the n-gram kernel. We used the n-gram with $n = 2, 3$ when comparing with this approach. Also, for non-unique pre-images, the solution was randomly chosen among the equivalent solutions.

3.1.1. Case when the length is known

We first considered the case where ℓ, the length of $\mathbf{y}$, is known at prediction time. For the Eulerian path heuristic, we normalized and rounded the predicted n-gram counts to obtain the correct number of n-grams in $\mathbf{y}$. The results for this setting are summarized in the second and third part of Table 2. Despite using the same learning algorithm and n-gram kernel, the accuracies of both methods differ significantly. The heuristic of rounding the predicted n-gram counts into integer values probably explains the decrease in accuracy. This suggests that, when possible, the exact computation of the pre-image is preferable.

The best results were obtained when using the Weighted degree kernel with $n = 3$. This suggests that, for this task, the positions of the n-gram in the words are important. Previously lacking a pre-image algorithm for this kernel, this is the first time it has been used for optical word recognition. Interestingly, the Weighted degree is among the cases where its pre-image is computable in polynomial time.

3.1.2. Case when the length is unknown

The second setting consists of a more difficult pre-image problem where ℓ, the length of the output word $\mathbf{y}$, is unknown at prediction time. For this task, the search algorithm was modified to take as parameters the minimum length ℓ_{min} and the maximum length ℓ_{max} observed in the training set instead of the exact length ℓ. We then created a priority queue for each of the $\ell_{\text{max}} - \ell_{\text{min}} + 1$ lengths. At every iteration, the best node of each priority queue was explored. Only the best solution over all lengths was kept. This allowed the algorithm to stop exploring the solutions for a specific length if the node on top of that priority queue had a smaller bound than the best solution found.

For the Eulerian path algorithm, we followed the procedure as explained in Cortes et al. (2007). That is, we chose a threshold t_j for each n-gram and rounded to one the count of the j-th predicted n-gram if $(\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}))_j > t_j$. Each threshold t_j was selected using the training set in a way that $\sum_{i=1}^{m} I\left[(\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}_i))_j > t_j\right] = \sum_{i=1}^{m} \boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}_i)_j$. When no components of $\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x})$ were above the thresholds, we rounded to one the $\ell_{\text{min}} - n + 1$ highest $(\mathbf{W}\boldsymbol{\phi}_{\mathcal{X}}(\mathbf{x}))_j$.

Table 3 reports the results for this setting. A decrease in accuracy was observed for both approaches, but is substantially greater for the Eulerian path algorithm. This could be explained by the difficulty of finding the thresholds for which all test examples would have the correct predicted number of n-grams. As for the branch and bound algorithm, the increased number of possible solutions over all lengths made it harder to find the exact pre-image during the 30 seconds allowed for the search. However, as seen in Table 3, the solutions returned by the best-first scheme of the branch and bound generally outperformed the approximations made by the Eulerian path algorithm.

Note that, for this setting, Cortes et al. (2007) have reported a 0/1 risk of 34.7 ± 2.3 by using a different kernel over the input $\mathbf{x}$, and a 0/1 risk of 24.4 ± 1.5 by combining the predictions obtained with different values of n. However, even with these improvements, the proposed approach still achieve better accuracy. Given the popularity of the n-gram kernel, the fact that it is outperformed by the Weighted Degree and other kernels (see Tables 2 and 3) is noteworthy.

Predictor	BPPs	CAMPs
h	**0.6134**	0.5916
$\widehat{h}$	0.6128	**0.5918**
$h_{\sigma_p=\infty}$	0.5111	0.5222
$\widehat{h}_{\sigma_p=\infty}$	**0.5589**	**0.5351**

Table 4. 10-fold cross-validation R^2 of the predictors on the BPPs and CAMPs datasets. Computed using the union of the 10 validation sets.

3.1.3. Execution time analysis

For the Hamming and the Weighted degree, with $n = 2$, it took an average of 15 milliseconds (ms) per prediction in both settings. For the Weighted degree, when $n = 3$, the time increased to 35 ms and 50 ms in the first and second setting respectively. The branch and bound search for the n-gram, with $n = 2, 3$, was allowed a maximum of 30 seconds. In the known length case, the search ended in the allowed time 99.94% of the time, with an average of 159 ms per prediction for the 2-gram and 70 ms for the 3-gram. In the unknown length case, the branch and bound was forced to terminate most of the time. However, as shown in Table 3, the correct output string was generally found. The branch and bound for the unknown length case was executed on one core of an Intel Xeon X5560 CPUs (2.8GHz). All other results were computed on one core of an Intel Core i7 (1.9GHz).

In comparison with the results of Table 2, the initial depth first search guided by the bounds gave a $0/1$ loss of 44.48% for the 2-gram and 7.94% for the 3-gram. Demonstrating the usefulness of the full branch and bound procedure.

3.2. String prediction for Classification and Regression

We followed the protocol suggested in Giguère et al. (2015) and used the same two bioactivity regression datasets they used. The first contains 101 cationic antimicrobial pentadecapeptides (CAMPs) (Wade & Englund, 2002). The second dataset consists of 31 bradykinin-potentiating pentapeptides (BPPs) (Ufkes et al., 1982).

As in Giguère et al. (2015), we used kernel ridge regression as the learning algorithm. Except when stated otherwise, all hyper-parameters for the GS kernel (n, σ_c, σ_p) and the kernel ridge regression (λ) were chosen by standard cross-validation. For each dataset, we learned two predictors, the first denoted by h uses an unnormalized kernel (the predictor used in Giguère et al. (2015)). The second, denoted by $\widehat{h}$, was trained using a normalized kernel. For some experiments, all hyper-parameters except $\sigma_p = \infty$ were chosen by cross-validation. When this is the case, we will refer to the two predictors by $h_{\sigma_p=\infty}$ and $\widehat{h}_{\sigma_p=\infty}$. The R^2 (coefficient of determination) of all predictors are shown in Table 4.

As explained in Section 2.1, whenever $\sigma_p = 0$, the solutions $\mathbf{y}^h$ and $\mathbf{y}^{\widehat{h}}$ are the same. The motivations behind presenting results obtained from the predictors $h_{\sigma_p=\infty}$ and $\widehat{h}_{\sigma_p=\infty}$ are two folds. First, it highlights the regime in which $\mathbf{y}^h$ and $\mathbf{y}^{\widehat{h}}$ differ the most. Second, many kernels, including the n-gram and the Blended Spectrum, are only obtained when $\sigma_p = \infty$. These kernels are commonly used, it this thus important to evaluate the proposed approach when using these even if they were not selected by cross-validation for the specific task studied here.

3.2.1. Comparing the solutions found

The first experiment compares the peptides found by both approaches under two different settings: the first uses h and $\widehat{h}$, while the second uses $h_{\sigma_p=\infty}$ and $\widehat{h}_{\sigma_p=\infty}$.

In the first setting, the method of Giguère et al. (2015) was used to identify $\mathbf{y}^h \in \mathcal{A}^{15}$ and $\mathbf{y}^h \in \mathcal{A}^5$, respectively for the CAMPs and BPPs dataset. The branch and bound was used to identify $\mathbf{y}^{\widehat{h}} \in \mathcal{A}^{15}$ and $\mathbf{y}^{\widehat{h}} \in \mathcal{A}^5$. The results are shown in the first half of Table 5. On the two datasets, both methods identified the same peptides. On the CAMPs and the BPPs datasets, the values of σ_p, chosen by cross-validation, are respectively 0.8 and 0.2. Since the values of σ_p are so close to 0, it is normal that both methods found the same solutions. This also suggests that the method of Giguère et al. (2015) offers some resistance to variation in the norm of $\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y})$ when σ_p is small.

In the second setting, we used the predictors $h_{\sigma_p=\infty}$ and $\widehat{h}_{\sigma_p=\infty}$.The solutions found by both methods are shown in the second half of Table 5. In that setting, the solution $\mathbf{y}^{h_{\sigma_p=\infty}}$ contains many n-gram repetitions. On the BPPs dataset, the solution contained two times the 2-gram `WA`. On the CAMPs dataset, the solution is basically the repetition of the 2-grams `FK` and `KI`. Hence, in situations where the norm of feature vectors can vary a lot, h favors strings having repetitions, thus a feature vector of large norm. However, this bias is generally unjustified and not biologically founded for peptides. In contrast, the solution $\mathbf{y}^{\widehat{h}_{\sigma_p=\infty}}$ found by the branch and bound contains no repetition for the BPPs dataset and fewer for the CAMPs dataset. Finally, the solutions found by the branch and bound share many substructures with the most bioactive peptides of the training sets.

3.2.2. Comparing the norms and the lengths

The second experiment quantitatively compares $||\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}^{h_{\sigma_p=\infty}})||$ with $||\boldsymbol{\phi}_{\mathcal{Y}}(\mathbf{y}^{\widehat{h}_{\sigma_p=\infty}})||$. To make the comparison more informative, we use the fact that the method of Giguère et al. (2015) can output a rank of the k sequences having the greatest predicted value, such that $[h(\mathbf{y}^h_1) \geq h(\mathbf{y}^h_2) \geq \ldots \geq h(\mathbf{y}^h_k)]$. This is also possible for

Method	BPPs	CAMPs
$\mathbf{y}^{h}$	IEWAK	WWKWWKRLRRLFLLV
$\mathbf{y}^{\widehat{h}}$	IEWAK	WWKWWKRLRRLFLLV
$\mathbf{y}^{h_{\sigma_p=\infty}}$	**WA**K**WA**	**FK**KI**FK**KI**FK**KI**FK**F
$\mathbf{y}^{\widehat{h}_{\sigma_p=\infty}}$	VEWAK	WKKI**FK**KIWKFRV**FK**

Table 5. Predicted peptides with highest bioactivity on BPPs and CAMPs datasets

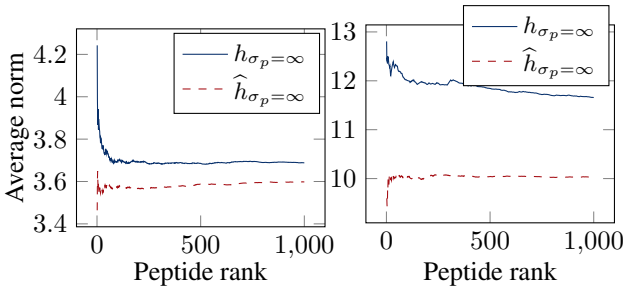

Figure 1. Cumulative moving average of $||\phi_{\mathcal{Y}}(\mathbf{y})||$ for the $1,000$ peptides with highest predicted bioactivities for the BPPs (left) and the CAMPs (right) datasets.

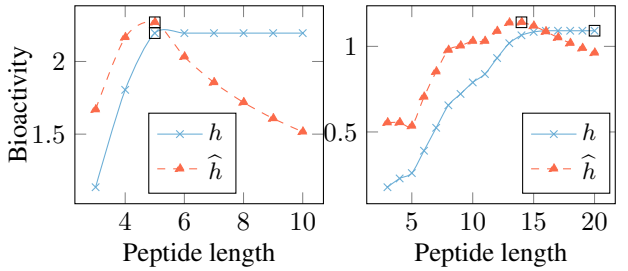

Figure 2. Maximum predicted bioactivity as a function of the peptide length for the BPPs (left) and the CAMPs (right) datasets.

the branch and bound by stopping the search only when the bound on top of the priority queue is lower than the k-th string found. For both datasets, we found the 1000 top peptides maximizing the un-normalized predictor $h_{\sigma_p=\infty}$ and the normalized predictor $\widehat{h}_{\sigma_p=\infty}$. Then, we compared the cumulative moving average norm of the feature vectors for all peptides.

The results on both datasets are shown in Figure 1. We observe that the branch and bound favored solutions with feature vectors of smaller norm.

The last experiment compares the ability of each method to handle strings of different lengths. This experiment is similar to the optical word recognition case where the length of the output word is unknown. Using both methods, we found $\mathbf{y}^{h} \in \mathcal{A}^{\ell}$ and $\mathbf{y}^{\widehat{h}} \in \mathcal{A}^{\ell}$ for different string lengths ℓ.

The results for both datasets are shown in Figure 2. The maximum bioactivity of h as a function of ℓ, increases, then stabilizes. In contrast, the maximum bioactivity of $\widehat{h}$ peaks near the optimal length, then decreases. This coincides more closely with our current understanding of the biology. This synthetic experiment suffices to highlight that when kernels are not normalized the solution $\mathbf{y}^{h}$ will be biased towards the longest possible string, a bias that $\mathbf{y}^{\widehat{h}}$ does not suffer from. Finally, the same comparison is done for $h_{\sigma_p=\infty}$ and $\widehat{h}_{\sigma_p=\infty}$ in the Supplementary Material.

3.2.3. Execution Time Analysis

The branch and bound search for the $\widehat{h}$ predictor took 5 seconds and 48 seconds respectively for the BPPs and CAMPs datasets. The time increased to 10 seconds and 23 minutes to find the 1000 best peptides. The search for the $\widehat{h}_{\sigma_p=\infty}$ predictor on the BPPs dataset took 8 seconds for the best peptide and 15 seconds for the 1000 best peptides. The branch and bound did not terminate for the $\widehat{h}_{\sigma_p=\infty}$ predictor on the CAMPs dataset. However, the best peptide found after 10 minutes was the same as after 20 hours, suggesting that proving optimality might be the crux of the problem.

4. Conclusion

First, we demonstrated that the structured output pre-image problem and the string prediction problem with normalized kernels both reduce to a unique combinatorial problem. We showed that for some string kernels, it is sometimes possible to solve this problem in polynomial time. When it is not possible, we proposed a low computational complexity upper bound for this problem. We demonstrated the usefulness of the bound in a branch and bound algorithm.

On the practical task of optical word recognition, the proposed approach significantly outperforms an existing pre-image heuristic whether the lengths of the words in the testing set are known or unknown. Also, empirical results show that the method proposed by Giguère et al. (2015) is biased toward strings having a feature with a large norm, a problem common to most string kernels. In these situations, the proposed method was shown to overcome this bias and new applications in structured output and string prediction are thus expected.

The $g(\mathbf{y}', \ell)$ bound was shown to be optimal. However, the $f(\mathbf{y}', \ell)$ bound could benefit from improvement, specifically in the case when $\sigma_p > 0$ and $\sigma_c > 0$. Finally, to further improve $F(\mathbf{y}', \ell)$, one could explore dual decomposition and Lagrangian relaxation methods (Fisher, 2004; Rush & Collins, 2014) to exploit the difference between the solution approaching $g(\mathbf{y}', \ell)$ and the one approaching $f(\mathbf{y}', \ell)$.

Acknowledgments

The authors would like to thank Claude-Guy Quimper for his insightful recommendations on the branch and bound. This work was supported in part by NSERC Discovery grants (FL:262067 , MM: 122405), by Fonds de recherche du Québec (FRQNT) (FL, MM: 2013-PR-166708), by Compute Canada, and by an award to Michael Tyers from the Ministère de l'enseignement supérieur, de la recherche, de la science et de la technologie du Québec through Génome Québec. AR is recipient of a Master's Scholarship from the FRQNT.

References

Cortes, Corinna, Mohri, Mehryar, and Weston, Jason. A general regression framework for learning string-to-string mappings. In Bakır, Gökhan, Hofmann, Thomas, Schölkopf, Bernhard, Smola, Alexander J., Taskar, Ben, and Vishwanathan, S. (eds.), *Predicting Structured Data*, chapter 8, pp. 143–168. MIT Press, Cambridge, MA, 2007.

Fisher, Marshall L. The lagrangian relaxation method for solving integer programming problems. *Management science*, 50(12_supplement):1861–1871, 2004.

Giguère, Sébastien, Marchand, Mario, Laviolette, François, Drouin, Alexandre, and Corbeil, Jacques. Learning a peptide-protein binding affinity predictor with kernel ridge regression. *BMC Bioinformatics*, 14, 2013.

Giguère, Sébastien, Laviolette, François, Marchand, Mario, Tremblay, Denise, Moineau, Sylvain, Liang, Xinxia, Biron, Éric, and Corbeil, Jacques. Machine learning assisted design of highly active peptides for drug discovery. *PLoS Comput Biol*, 11(4):e1004074, 04 2015. doi: 10.1371/journal.pcbi.1004074.

Leslie, Christina S, Eskin, Eleazar, and Noble, William Stafford. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pp. 566–575. World Scientific, 2002.

Meinicke, P., Tech, M., Morgenstern, B., and Merkl, R. Oligo kernels for datamining on biological sequences: A case study on prokaryotic translation initiation sites. *BMC Bioinformatics*, 5, 2004.

Rätsch, Gunnar and Sonnenburg, Sören. Accurate Splice Site Detection for Caenorhabditis elegans. In B and Vert, J. P. (eds.), *Kernel Methods in Computational Biology*, pp. 277–298. MIT Press, 2004.

Rush, Alexander M and Collins, Michael. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *arXiv preprint arXiv:1405.5208*, 2014.

Shawe-Taylor, John and Cristianini, Nello. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Taskar, Ben, Guestrin, Carlos, and Koller, Daphne. Max-margin Markov networks. In Thrun, Sebastian, Saul, Lawrence, and Schölkopf, Bernhard (eds.), *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Toussaint, Nora, Widmer, Christian, Kohlbacher, Oliver, and Rätsch, Gunnar. Exploiting physico-chemical properties in string kernels. *BMC bioinformatics*, 11(Suppl 8):S7, 2010.

Ufkes, Jan G.R., Visser, Berend J., Heuver, Gerritdina, Wynne, Herman J., and Meer, Cornelis Van Der. Further studies on the structure-activity relationships of bradykinin-potentiating peptides. *European Journal of Pharmacology*, 79(12):155 – 158, 1982.

Wade, David and Englund, Jukka. Synthetic antibiotic peptides database. *Protein and peptide letters*, 9(1):53–57, 2002.