GRADIENT-BASED OPTIMIZATION OF HYPERPARAMETERS FOR BASE-PAIRING PROFILE LOCAL ALIGNMENT KERNELS

KENGO SATO[1,2,3] YUTAKA SAITO[3]
sato-kengo@aist.go.jp saito@dna.bio.keio.ac.jp

YASUBUMI SAKAKIBARA[3]
yasu@bio.keio.ac.jp

[1] *Japan Biological Informatics Consortium (JBIC), 2–45 Aomi, Koto-ku, Tokyo 135–8073, Japan*
[2] *Computational Biology Research Center (CBRC), National Institute of Advanced Industrial Science and Technology (AIST), 2–42 Aomi, Koto-ku, Tokyo 135–0064, Japan*
[3] *Department of Biosciences and Informatics, Keio University, 3–14–1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223–8522, Japan*

We have recently proposed novel kernel functions, called *base-pairing profile local alignment (BPLA) kernels* for discrimination and detection of functional RNA sequences using SVMs. We employ STRAL's scoring function which takes into account sequence similarities as well as upstream and downstream base-pairing probabilities, which enables us to model secondary structures of RNA sequences. In this paper, we develop a method for optimizing hyperparameters of BPLA kernels with respect to discrimination accuracy using a gradient-based optimization technique. Our experiments show that the proposed method can find a nearly optimal set of parameters much faster than the grid search on all parameter combinations.

Keywords: support vector machines; kernel functions; gradient decent; non-coding RNAs.

1. Introduction

Recent research has discovered crucial roles of non-coding RNAs (ncRNAs) in cells, including post-transcriptional gene regulation, maturation of mRNAs, rRNAs and tRNAs. Most functional ncRNAs form secondary structures with hydrogen bonds such as Watson-Crick base-pairs (`A-U` and `G-C`) and Wobble base-pairs (`G-U`). It is well-known that functions of ncRNAs are deeply related to their secondary structures rather than primary sequences. For example, transfer RNAs registered in the Rfam database [?] have only 44 % average sequence identity, whereas they are folded into conserved clover leaf secondary structures. Therefore, ncRNAs are modeled by means of their secondary structures for computational ncRNA analysis. One of such models is stochastic context-free grammars (SCFGs), which can represent RNA secondary structures without pseudoknots [?, ?]. These grammatical methods have succeeded in modeling typical secondary structures of RNAs, and are commonly used for structural alignment of RNA sequences. However, such stochastic models are not capable of discriminating the member sequences of RNA families from nonmembers with sufficiently high accuracy to detect non-coding RNA regions in genome sequences.

On the other hand, a compelling attraction of support vector machines (SVMs) and other kernel methods has been increasing in many research fields, including bioinformatics [**?**], because of their robustness. For example, Saigo *et al.* have proposed local alignment kernels for amino acid sequences [**?**, **?**], and their kernels with SVMs have outperformed other state-of-the-art methods in benchmarks for remote homology detection of amino acid sequences. Therefore, we have been considering using kernel methods, including SVMs, for the analysis of functional ncRNAs [**?**, **?**, **?**].

We have recently proposed novel kernel functions, called *base-pairing profile local alignment (BPLA) kernels* for discrimination and detection of functional RNA sequences using SVMs [?]. We extend the concept of local alignment kernels in such a way that it can handle RNA sequences using the scoring function in STRAL, which is a practical aligner for non-coding RNAs [**?**]. The local alignment kernels measure the similarity between two sequences by summing the scores over all possible local alignments with gaps, whereas STRAL's scoring function takes into account sequence similarities as well as upstream and downstream base-pairing probabilities, which enables us to model secondary structures of RNA sequences. Our experiments have proved powerful discrimination ability of BPLA kernels for functional RNA families. Furthermore, we have demonstrated the applicability of our kernels to the problem of genome-wide search of snoRNA families in the *C. elegans* genome, and have confirmed that the expression is valid in 14 out of 48 of our predicted candidates by using qRT-PCR (p-value $= 2.8 \times 10^{-32}$).

Our previous experiments showed that parameters for the BPLA kernels are quite different depending on what kind of data (flanking regions or not) and what family of sequences should be trained. Our previous study employed a brute force approach in which optimal parameters for each case were calibrated by the grid search on all parameter combinations from selected search space for each parameter. It is easily predicted that this approach should require exponential time in the number of parameters to be optimized.

Therefore, we propose a gradient-based parameter optimization for BPLA kernels. This work is based on Keerthi *et al.* [**?**], in which gradients of accuracy measures (e.g. AUC) with respect to hyperparameters of kernel functions are calculated, then these hyperparameters are optimized with a numerical optimization algorithm such as the quasi-Newton's method. We develop a dynamic programming algorithm for calculating the gradients of BPLA kernels with respect to their parameters to be optimized by the Keerthi's method. Our experiments show that the proposed method can find a nearly optimal set of parameters much faster than the grid search on all parameter combinations.

This paper is organized as follows: in section 2, we first introduce BPLA kernels for functional RNA analysis and dynamic programming algorithms for calculating BPLA kernels and their gradient with respect to their hyperparameters. Then, we briefly describe the Keerthi's gradient-based optimization method. In section 3, computational experiments are shown to confirm the efficiency of our method, and we discuss our framework in section 4. Finally, we conclude this paper in section 5.

2. Methods

2.1. *Base-pairing profile local alignment kernels*

Local alignment kernels can calculate the similarity between a pair of sequences by taking into account only sequence homology [?]. However, since it is well-known that ncRNAs form secondary structures which are related to their functions, it is necessary to consider the secondary structures of ncRNAs in order to compare two RNA sequences. To do this, we have proposed stem kernels [?] and their approximation using directed acyclic graphs (DAGs) of ncRNAs [?]. The stem kernels can consider RNA secondary structures strictly, but require huge computational time for practical problems. Differing from them, BPLA kernels employ a kind of summary information of secondary structures, called *base-pairing profiles*, as well as STRAL, which is a practical aligner for non-coding RNAs [?].

For each sequence, we first calculate a base-pairing probability matrix using the McCaskill algorithm [**?**] or the inside-outside algorithm [**?**]. The base-pairing probability matrix for a sequence x consists of the base-pairing probabilities P_{ij} that the i-th and the j-th nucleotides of x form a base pair, which is defined as:

$$P_{ij} = \mathbb{E}[I_{ij}|x] = \sum_{y \in \mathcal{Y}(x)} p(y|x) I_{ij}(y), \tag{1}$$

where $\mathcal{Y}(x)$ is an ensemble of all possible secondary structures of x, $p(y|x)$ is the posterior probability of a secondary structure y given x, and $I_{ij}(y)$ is an indicator function, which equals 1 if the i-th and the j-th nucleotides form a base-pair in y and 0 otherwise. In this study, we employ the Vienna RNA package [**?**] for computing the expected counts (1) using the McCaskill algorithm.

Subsequently, for each position i of x, the base-pairing probabilities are summarized into tree kinds of sums: the probability $P_i^{left} = \sum_{j>i} P_{ij}$ that a pair is formed with one of the downstream nucleotides, the probability $P_i^{right} = \sum_{j<i} P_{ji}$ that a pair is formed with one of the upstream nucleotides, and the probability $P_i^{unpair} = 1 - (P_i^{left} + P_i^{right})$ that the nucleotide is unpaired. A probability distribution consisting of these three probabilities is called a *base-pairing profile* [**?**].

In accordance with STRAL [?], we define the match score between two nucleotides x_i and y_j using base-pairing profiles as follows:

$$\begin{aligned} S(x_i, y_j) =& \alpha(S_{struct}) + S_{seq} \\ =& \alpha\left(\sqrt{P_{x_i}^{left} P_{y_j}^{left}} + \sqrt{P_{x_i}^{right} P_{y_j}^{right}}\right) + \sqrt{P_{x_i}^{unpair} P_{y_j}^{unpair}} S'(x_i, y_j), \end{aligned} \tag{2}$$

```
Initialization:
for $i \in \{0, \ldots, |x|\}$ and $j \in \{0, \ldots, |y|\}$ do
    $M(i,0) = I_X(i,0) = I_Y(i,0) = R_X(i,0) = R_Y(i,0) = 0$
    $M(0,j) = I_X(0,j) = I_Y(0,j) = R_X(0,j) = R_Y(0,j) = 0$
end for
Iteration:
for $i \in \{1, \ldots, |x|\}$ and $j \in \{1, \ldots, |y|\}$ do
    $M(i,j) = e^{\beta S(x_i, y_j)}(1 + I_X(i-1, j-1) + I_Y(i-1, j-1) + M(i-1, j-1))$
    $I_X(i,j) = e^{\beta g} M(i-1, j) + e^{\beta e} I_X(i-1, j)$
    $I_Y(i,j) = e^{\beta g}(M(i, j-1) + I_X(i, j-1)) + e^{\beta e} I_Y(i, j-1)$
    $R_X(i,j) = M(i-1, j) + R_X(i-1, j)$
    $R_Y(i,j) = M(i, j-1) + R_X(i, j-1) + R_Y(i, j-1)$
end for
Termination:
$K(x,y) = 1 + R_X(|x|, |y|) + R_Y(|x|, |y|) + M(|x|, |y|)$
```

Fig. 1. The dynamic programming of the BPLA kernel.

where α is a weight parameter for structural information, and S' is a substitution scoring function between two nucleotides. In this study, we use a modified RIBOSUM 85-60 [?] as the substitution scoring matrix, in which its smallest eigenvalue is subtracted from each of its diagonal elements in order to make it positive semidefinite. Then, we employ equation (2) as the match scores for the local alignment kernels between two sequences x and y, which is defined as follows:

$$K(x,y) = \sum_{\pi \in \Pi(x,y)} \exp \beta s(x,y,\pi), \tag{3}$$

where $\beta \geq 0$ is a concentration parameter, $\Pi(x,y)$ is a set of all possible local alignments of x and y, and $s(x,y,\pi)$ is a score of alignment π defined by:

$$s(x,y,\pi) = \sum_{a,b} n_{a,b}(x,y,\pi) \cdot S(a,b) - n_g(x,y,\pi) \cdot g - n_e(x,y,\pi) \cdot e, \tag{4}$$

where $n_{a,b}(x,y,\pi)$ is the number of times that nucleotides a is aligned with nucleotides b in π, $n_g(x,y,\pi)$ and $n_e(x,y,\pi)$ are the numbers of gap opens and gap extensions, respectively, and g and e are penalties for gap opens and gap extensions, respectively. We call equation (3) with the match scoring function (2) the *base-pairing profile local alignment (BPLA) kernel.* Equation (3) can be computed by the dynamic programming technique shown as Figure 1.

The hyperparameter optimization algorithm, described in Section 2.2, employs the gradient of $K(x,y)$ with respect to parameters α, β, g and e, which can also be computed by the dynamic programming. Figure 2 shows the dynamic programming of calculating the derivative with respect to α. The derivatives with respect to the other parameters can be computed as well as α.

In order to avoid length bias, we usually normalize kernel values. Therefore, we would like to optimize hyperparameters with respect to the normalized kernels K_n by using its derivative K'_n:

```
Initialization:
for $i \in \{0, \dots, |x|\}$ and $j \in \{0, \dots, |y|\}$ do
    $\frac{\partial}{\partial\alpha}M(i,0) = \frac{\partial}{\partial\alpha}I_X(i,0) = \frac{\partial}{\partial\alpha}I_Y(i,0) = \frac{\partial}{\partial\alpha}R_X(i,0) = \frac{\partial}{\partial\alpha}R_Y(i,0) = 0$
    $\frac{\partial}{\partial\alpha}M(0,j) = \frac{\partial}{\partial\alpha}I_X(0,j) = \frac{\partial}{\partial\alpha}I_Y(0,j) = \frac{\partial}{\partial\alpha}R_X(0,j) = \frac{\partial}{\partial\alpha}R_Y(0,j) = 0$
end for
Iteration:
for $i \in \{1, \dots, |x|\}$ and $j \in \{1, \dots, |y|\}$ do
    $\frac{\partial}{\partial\alpha}M(i,j) = \beta M(i,j)\left(\sqrt{P^{left}_{x_i}P^{left}_{y_j}} + \sqrt{P^{right}_{x_i}P^{right}_{y_j}}\right)$
            $+e^{\beta S(x_i,y_j)}(\frac{\partial}{\partial\alpha}I_X(i-1,j-1) + \frac{\partial}{\partial\alpha}I_Y(i-1,j-1) + \frac{\partial}{\partial\alpha}M(i-1,j-1))$
    $\frac{\partial}{\partial\alpha}I_X(i,j) = e^{\beta g}\frac{\partial}{\partial\alpha}M(i-1,j) + e^{\beta e}\frac{\partial}{\partial\alpha}I_X(i-1,j)$
    $\frac{\partial}{\partial\alpha}I_Y(i,j) = e^{\beta g}(\frac{\partial}{\partial\alpha}M(i,j-1)\frac{\partial}{\partial\alpha}I_X(i,j-1)) + e^{\beta e}\frac{\partial}{\partial\alpha}I_Y(i,j-1)$
    $\frac{\partial}{\partial\alpha}R_X(i,j) = \frac{\partial}{\partial\alpha}M(i-1,j) + \frac{\partial}{\partial\alpha}R_X(i-1,j)$
    $\frac{\partial}{\partial\alpha}R_Y(i,j) = \frac{\partial}{\partial\alpha}M(i,j-1) + \frac{\partial}{\partial\alpha}R_X(i,j-1) + \frac{\partial}{\partial\alpha}R_Y(i,j-1)$
end for
Termination:
$\frac{\partial}{\partial\alpha}K(x,y) = \frac{\partial}{\partial\alpha}R_X(|x|,|y|) + \frac{\partial}{\partial\alpha}R_Y(|x|,|y|) + \frac{\partial}{\partial\alpha}M(|x|,|y|)$
```

Fig. 2. The dynamic programming of the derivative of the BPLA kernel with respect to α.

$$\begin{aligned}
K_n(x,y) =& \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}} \\
K_n'(x,y) =& \frac{K'(x,y)}{\sqrt{K(x,x)K(y,y)}} - \frac{K_n(x,y)}{2}\left(\frac{K'(x,x)}{K(x,x)} + \frac{K'(y,y)}{K(y,y)}\right).
\end{aligned}$$

2.2. *Gradient-based optimization for SVMs*

Tuning hyperparameters of kernel functions is known as an important task to achieve enough accuracy in practical problems. Although brute force approaches like the grid search are usually employed for such tasks, huge computational time will obviously be required if the search space of hyperparameters is large. Keerthi *et al.* have proposed a more efficient method which optimizes hyperparameters via the gradient-based optimization [?].

We assume that n training data $\{(x_i, y_i)\}_{i=0}^n$ $(y_i \in \{-1, +1\})$ are given. In SVM training, we optimize the parameters w and b which satisfy the following condition:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_i l(o_i, y_i),$$

where l is a loss function such as the hinge-loss $l(o_i, y_i) = \max\{0, 1 - y_i o_i\}$, C is a weight for the loss function, and o_i is the output from the SVM for the training data:

$$o_i = w \cdot \phi(x_i) - b = \sum_j \alpha_j y_j K(x_i, x_j) - b.$$

Here, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ is called a kernel function which can implicitly define a high-dimensional mapping ϕ. The terms α_js constitute a dual representation for w. We can obtain a linear system from the Karush-Kuhn-Tucker (KKT) condition:

$$P \begin{pmatrix} \alpha \\ b \end{pmatrix} = q. \tag{5}$$

The training data and their coefficients α are divided into three parts: a part I_0 for $\alpha_i = 0$, a part I_u for $0 < \alpha_i < C$, and a part I_c for $\alpha_i = C$. The linear system (5) can be rewritten as follows:

$$\alpha_0 = 0, \quad \alpha_c = Ce_c, \quad \begin{pmatrix} \Omega_{uu} & -y_u \\ -y_u^T & 0 \end{pmatrix} \begin{pmatrix} \alpha_u \\ b \end{pmatrix} = \begin{pmatrix} e_u - \Omega_{uc}\alpha_c \\ y_c^T \alpha_c \end{pmatrix},$$

where $\Omega_{ij} = y_i y_j K_{ij}$ and e_u is a unit vector. These α and b can be computed by solving quadratic programming.

For given validation data $\{(\bar{x}_l, \bar{y}_l)\}$, we can calculate:

$$\bar{o}_l = \sum_i \alpha_i y_i K(\bar{x}_l, x_i) - b,$$

and can rewrite it with a matrix form:

$$\bar{o}_l = \psi_l^T \beta,$$

where β is a vector containing α and b, and ψ_l is a vector containing $y_i K_{li}, i = 1, \dots, n$ and -1 as the last element (corresponding to b).

Our objective is to maximize f (which is a kind of accuracy measures such as specificity, sensitivity, F-value, or AUC) with respect to hyperparameters θ of a given kernel function K. Here, $\dot{f}$ denotes a derivative of f with respect to θ, that is, $\dot{f} = \frac{\partial f}{\partial \theta}$, which can be computed as:

$$\dot{f} = \sum_l \delta_l \dot{\bar{o}}_l = \sum_l \delta_l \left(\psi_l^T P^{-1} \left(\dot{q} - \dot{P}\beta \right) + \dot{\psi_l}^T \beta \right) = d^T \left(\dot{q} - \dot{P}\beta \right) + \left(\sum_l \delta_l \dot{\psi}_l \right)^T \beta, \tag{6}$$

where d is the solution of

$$P^T d = \left(\sum_l \delta_l \psi_l \right), \tag{7}$$

and

$$\delta_l = \frac{\partial f}{\partial \bar{o}_l}. \tag{8}$$

The steps for calculating the derivative $\dot{f}$ is as follows. First, compute δ_l from (8). Then, solve (7) for d. Equation (7) can efficiently be solved by the conjugate gradient method. Then, compute $\dot{f}$ for each θ from (6). After f and its gradients $\dot{f}$ are computed, we can apply the gradient-based optimization by the L-BFGS [**?**] (or the L-BFGS-B [**?**] with box-constraints).

We optimize the hyperparameters of BPLA kernels by means of AUC, that is, $f = auc$ defined as:

$$auc = \frac{1}{n^+n^-} \sum_{i:\bar{y}_i=+1} \sum_{j:\bar{y}_j=-1} \mathrm{sign}(\bar{o}_i - \bar{o}_j), \tag{9}$$

where n^+ and n^- are the numbers of the positive and negative examples in the validation dataset, respectively. The step function $\mathrm{sign}(x)$ is approximated by using the sigmoidal function $s(x) = \frac{1}{1+\exp(-\sigma x)}$ to make (9) differentiable.

3. Results

In order to illustrate the efficiency of our gradient-based optimization, we compared the proposed method with the brute-force grid search. In each experiment, we first calculated AUC for all the parameter combinations in the predefined search space, and then performed our optimization procedure with the corresponding box constraints to evaluate whether it could approach the AUC value at the optimal grid point. We used the same data set as in our previous paper [?], which includes five ncRNA families taken from the Rfam database [?]. For each family, 100 sequences were chosen as positive samples in such a way that the pairwise identity is not above 80% for any pair of sequences, and as negative samples, 100 randomly shuffled sequences with the same dinucleotide composition as the positives were generated. AUC was calculated using four-fold cross-validation, and its gradient was computed by summing over the four data partitions.

In our first experiment, we attempted to simultaneously optimize all the four parameters α, β, g, and e in BPLA kernels, as well as C in the loss functions of SVMs. To do this, we defined the search space with the five-dimensional box constraints: $0 < \alpha \leq 20$, $0 < \beta \leq 0.2$, $0 < g \leq 30$, $0 < e \leq 1$, and $0 < C \leq 40$. Subsequently, we carried out the grid search in which we tried six values for each parameter at even intervals, *i.e.* $\alpha \in \{1 \times 10^{-9}, 4, 8, 12, 16, 20\}$, $\beta \in \{1 \times 10^{-9}, 0.04, 0.08, 0.12, 0.16, 0.20\}$, and similarly for the others. Then, we performed the gradient-based optimization. Since the proposed method might fall into one of local maxima, resulting AUC depends on initial values of the parameters. Therefore, we executed the optimization procedure from various start points. These start points were chosen by combining the first or third quantile of each parameter, *i.e.* $\alpha \in \{5, 15\}$, $\beta \in \{0.05, 0.15\}$, and similarly for the others, and we evaluated the mean AUC value of the 2^5 trials.

Table 1 presents the experimental results. Although the grid search attained the best discrimination for all the ncRNA families tested, it took the huge computational time because of brute-force calculation for 7776 ($= 6^5$) combinations of the parameters. Note that even the relatively coarse grid with only six values for one dimension required such massive computation. This clearly shows that the grid search is not a practical way to adapt the parameters of BPLA kernels against various kinds of data set. On the other hand, our gradient-based optimization successfully found the nearly optimal set of the parameters much faster than the grid

Table 1. Comparison between the gradient-based optimization and the grid search.

	gradient-based optimization			grid search		
Family	N	AUC	time (h)	N	AUC	time (h)
5S rRNA	24	0.997 (0.956)	1.8	7776	0.997	149.0
tRNA	15	0.993 (0.957)	0.6	7776	0.995	67.5
C/D snoRNA	25	0.843 (0.773)	1.8	7776	0.887	124.8
H/ACA snoRNA	23	0.879 (0.832)	3.5	7776	0.906	248.0
miRNA	20	0.995 (0.971)	1.1	7776	0.995	89.0

Note: Family: name of the target ncRNA family. N: number of parameter sets tried. AUC: area under the ROC curve, the value in parentheses is AUC calculated at the start point of optimization. time: computational time on a 2.8 GHz AMD Opteron processor. In the columns of the gradient-based optimization, mean values of 2^5 trials from the different start points are shown.

search. The AUC was substantially improved compared to the start point on average, and convergence was obtained within a few dozen of iterations. In particular, we achieved the comparable discrimination ability to the grid search for 5S rRNAs, tRNAs and miRNAs showing the effectiveness of the proposed method.

The second experiment was organized to visually demonstrate how our method works in the parameter space. The idea was that we would show a trajectory of optimization on the heat map of AUC. For this purpose, we focused only on α and β, and fixed the other parameters. Thus, we defined the same box constraints for α and β as in the first experiment, whereas we consistently used $g = 27$, $e = 1$, and $C = 1$ for the others. We calculated AUC over the finer grid with 100 values for each of α and β, and initiated the gradient-based optimization. Figure 3 shows the corresponding heat map and the trajectory plot. We present the typical results for the C/D snoRNA data set. The parameters α and β were moved sharply to the direction of higher AUC, and well calibrated along with the iterations. Indeed, the optimization in this case reached the region of highest AUC, and successfully converged at the global optima in the parameter space.

4. Discussion

Our current implementation of the hyperparameter optimizer for BPLA kernels does not optimize the substitution scoring function for RNAs (S' in (2)) although derivatives of substitution scores can easily be calculated. In fact, Saigo *et al.* have proposed a method for optimizing substitution scores for local alignment kernels using the gradient decent with respect to a confidence score for remote homology search [?] (not with respect to the SVM classification). However, the optimization of the substitution scoring matrix using the L-BFGS-B, which our method employed, cannot hold the positive semidefiniteness of S' and BPLA kernels, which is the essential basis of kernel methods. This kind of optimization problems can be solved by a method for semidefinite programming (SDP) [**?**]. Because of small effect of changing substitution scoring matrices in our method (data not shown), we fixed the substitution scoring matrix to a modified RIBOSUM 85-60 [?], in which its

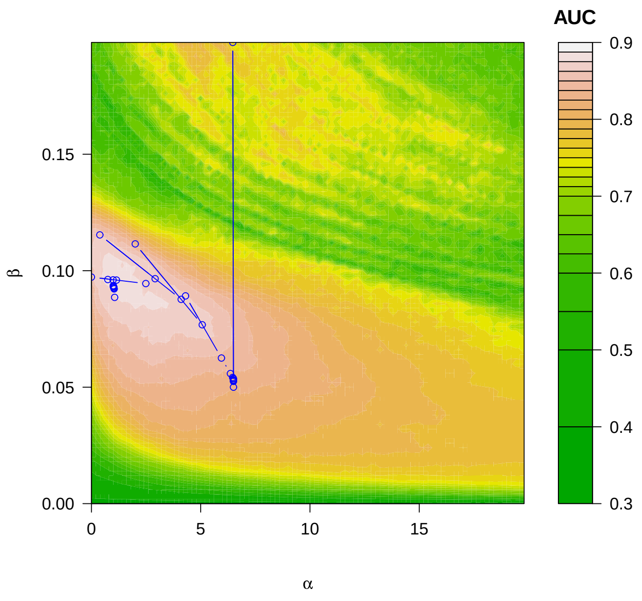

Fig. 3. Trajectory of parameter optimization on the heat map of AUC. AUC values evaluated for the C/D snoRNA data set are represented by gradation over the parameter space. The optimization was initiated from $(\alpha, \beta) = (6.50, 0.05)$, and terminated at $(\alpha, \beta) = (1.01, 0.09)$ after 35 iterations.

smallest eigenvalue is subtracted from each of its diagonal elements in order to make it positive semidefinite.

5. Conclusion

We have developed a method for optimizing hyperparameters of BPLA kernels with respect to discrimination accuracy using a gradient-based optimization technique. Our experiments showed that the proposed method can find a nearly optimal set of parameters much faster than the grid search on all parameter combinations.

We are planing to apply BPLA kernels to comparative analysis of functional RNAs, which can consider the sequence conservation between species as well as the structural conservation. For this purpose, the profile-profile comparison should be implemented.

Acknowledgment

This work was supported in part by a grant from "Functional RNA Project" funded by the New Energy and Industrial Technology Development Organization (NEDO) of Japan, and was also supported in part by Grant-in-Aid for Scientific Research on Priority Area "Comparative Genomics" No. 17018029 from the Ministry of Education, Culture, Sports, Science and Technology of Japan. We thank our colleagues from the RNA Informatics Team at the Computational Biology Research Center (CBRC) for fruitful discussions.