BIOINFORMATICS

Vol. 00 no. 00 2006
Pages 1–1

PALMA: mRNA to Genome Alignments using Large Margin Algorithms

Uta Schulze,[1,2,*] Bettina Hepp,[3,*] Cheng Soon Ong,[1,4] and Gunnar Rätsch[1,†]

[1] Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen, Germany
[2] University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany
[3] Fraunhofer FIRST, Kekulèstr. 7, 12489 Berlin, Germany
[4] Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany

ABSTRACT

Motivation: Despite many years of research on how to properly align sequences in the presence of sequencing errors, alternative splicing and micro-exons, the correct alignment of mRNA sequences to genomic DNA is still a challenging task.

Results: We present a novel approach based on large margin learning that combines accurate splice site predictions with common sequence alignment techniques. By solving a convex optimization problem, our algorithm – called *PALMA* – tunes the parameters of the model such that true alignments score higher than other alignments. We study the accuracy of alignments of mRNAs containing artificially generated micro-exons to genomic DNA. In a carefully designed experiment, we show that our algorithm accurately identifies the intron boundaries as well as boundaries of the optimal local alignment. It outperforms all other methods: for 5702 artificially shortened EST sequences from *C. elegans* and human it correctly identifies the intron boundaries in all except two cases. The best other method is a recently proposed method called *exalin* which misaligns 37 of the sequences. Our method also demonstrates robustness to mutations, insertions and deletions, retaining accuracy even at high noise levels.

Availability: Datasets for training, evaluation and testing, additional results and a stand-alone alignment tool implemented in C++ and python are available at `http://www.fml.mpg.de/raetsch/projects/palma`.

Contact: Gunnar.Raetsch@tuebingen.mpg.de

1 INTRODUCTION

Many genomes have been sequenced recently. This is only a first step to understand what the genome actually encodes. Fortunately, most of them also come with rather large amounts of expressed sequence tags (ESTs; sequenced parts of mRNA), which help to accurately recognize genes and to identify the exon/intron boundaries as well as alternative splice forms (see Zhang and Gish (2006) and references therein).

Many methods for aligning ESTs to genomic DNA have been proposed, including approaches based on *blast* (Altschul *et al.*, 1990), *spliced alignments* (Gelfand *et al.*, 1996), *sim4* (Florea *et al.*, 1998), *GeneSeqer* (Usuka *et al.*, 2000), *Spidey* (Wheelan SJ, 2001), *blat* (Kent, 2002), an approach to find additional micro-exons (Volfovsky *et al.*, 2003) and most recently *exalin* (Zhang and Gish, 2006). The identification of exon/intron boundaries is important for finding the correct alignment. Therefore most approaches try to find an alignment that prefers splice site consensus signals at the ends of the identified introns (usually `GT/AG`, considerably less often `GC/AG` and in some organisms also `AT/AC`) that help to accurately identify these boundaries. This is done by employing either dynamic programming or sophisticated heuristics.

Zhang and Gish (2006) used an information theoretic approach to combine the two types of information available during alignment: the sequence similarity and splice site predictions. Given this model, dynamic programming is used to compute the maximum-log likelihood alignment. Our algorithm, called *PALMA*, is based on similar ideas as *exalin*. The main differences are (a) the modeling of splice sites using support vector machines instead of the commonly used position specific scoring matrices (PSSMs) (Berg and von Hippel, 1987; Stormo, 1988), (b) the inclusion of an intron length model and (c) a novel way of combining of the different types of information using a large margin based approach. In Rätsch *et al.* (2006b) we previously considered a global alignment algorithm. In this work we consider local alignments which is better suited for mRNA to genome alignments.

Our approach does not include any probabilistic models and hence does not return probabilities for a particular alignment. It is, however, able to very accurately and robustly align sequences as will be seen in the experimental section, where we consider the problem of aligning modified EST sequences to genomic DNA (*C. elegans* and human) using the most difficult setup: We generated artificially shortened internal exons (3-50nt) and added small to large amounts of noise in the mRNA sequences. We show that our method very accurately aligns the sequences while other methods fail as soon as the exons become too short or the amount of noise too large.

2 METHOD

The idea of our algorithm is to compute an alignment by dynamic programming that uses a scoring function. We tune the parameters of the scoring functions such that the true alignment does not only achieve a large score (to be "most likely"), but also that all other alignments score considerably lower than the true alignment (to obtain a "large margin between the alignments"). Similar ideas are used in other large margin algorithms such as Support Vector Machines (SVMs) (Vapnik, 1995; Müller *et al.*, 2001) and Boosting (Freund and Schapire, 1997; Meir and Rätsch, 2003). The resulting scoring function can then be maximized using dynamic programming in order to obtain the optimal alignment. Our method consists of three independent parts: the splice site prediction model, the dynamic programming algorithm and the optimization of the scoring function. We describe them in the following sections.

* authors contributed equally

† to whom correspondence should be addressed

Training the splice site model and also the large margin combination requires separate sequence data sets. For the splice site model, we used genes that were EST confirmed but without full length cDNA support (set 1). We consider a random subset of 40% of all cDNA confirmed genes without evidence for alternative splicing for training the large margin combination (set 2). The remaining 20% and 40% were used for hyper-parameter tuning (set 3) and final evaluation (set 4) respectively.

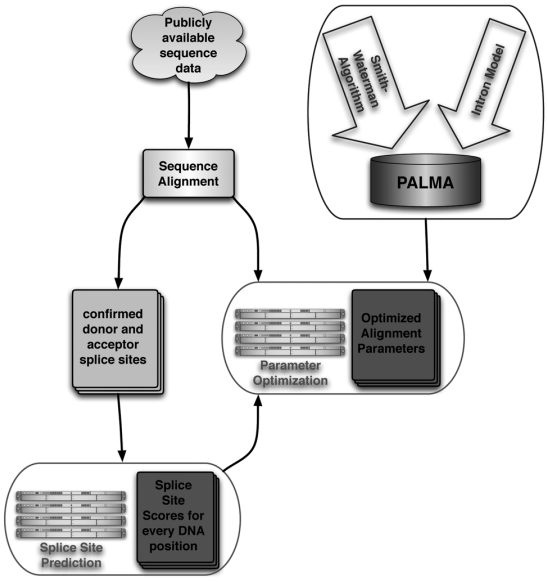

Fig. 1. *PALMA* generalizes the Smith-Waterman algorithm by including an intron model taking splice site predictions as well as intron length information into account. Publicly available sequence data of high quality is used to train PALMA. Confirmed donor and acceptor sites are used to train a SVM based splice site predictor. The information is then used to optimize the parameters used for alignment.

2.1 Related Work

Two methods based on a similar idea have been independently proposed in Joachims *et al.* (2005) as well as Kececioglu and Kim (2006). Joachims *et al.* (2005) propose an algorithm related to support vector machines similar in spirit to our approach that can learn parameters for protein sequence alignments. Kececioglu and Kim (2006) presents an algorithm (based on the method from Gusfield *et al.* (1994)) that can simultaneously learn hundreds of parameters. Their approach called the "unique-optimal alignment" is equivalent to what is known as the "hard margin" approach in SVM literature; where the true alignment is constrained to score better than all other possible alignments. In contrast, the "soft margin" approach in Joachims *et al.* (2005) allows the true alignment to score worse than others but it penalizes the occurrence of such events. The "near-optimal alignment" approach in Kececioglu and Kim (2006), which attempts to allow suboptimal alignments in the training set, does not impose the concept of a margin at all.

Our optimization approach is very similar to that in Joachims *et al.* (2005), as we use the "soft margin" concept. In addition, unlike both previous approaches which were applied to protein sequence alignment, we explicitly model introns and splice sites. Furthermore, we explicitly regularize our parameters in an for the model appropriate manner, while Kececioglu and Kim (2006) does not regularize at all, and Joachims *et al.* (2005) performs standard SVM L_2-norm regularization. In essence, we utilize knowledge of the problem, i.e. mRNA to DNA alignments, to determine a better alignment model.

2.2 Splice Site Predictions

From the set of EST sequences (set 1) we extracted sequences of confirmed donor (intron start) and acceptor (intron end) splice sites (see Appendix for details). For acceptor splice sites we used a window of 80bp upstream to 60bp downstream of the site (on the DNA). For donor sites we used 60bp upstream and 80bp downstream. Also from these training sequences we extracted non-splice sites that are within an exon or intron of the sequence and have `AG` (acceptor) or `GT`/`GC` (donor) consensus. In order to recognize acceptor and donor splice sites, we trained two Support Vector Machine classifiers (Vapnik, 1995) with soft-margin using the so-called "weighted degree" kernel (Sonnenburg *et al.*, 2002; Rätsch *et al.*, 2006a; Rätsch *et al.*, 2007; Sonnenburg *et al.*, 2007). The kernel computes the similarity between two sequences $\mathbf{s}$ and $\mathbf{s}'$ by considering substrings occurring in both strings up to length d and their position. The kernel can be computed by:

$$\mathrm{k}(\mathbf{s},\mathbf{s}') = \sum_{j=1}^{d} v_j \sum_{i=1}^{N-j} \mathbf{I}(s_{[i,i+j]} = s'_{[i,i+j]}), \tag{1}$$

where $N = 140$ is the length of the sequence and $s_{[a,b]}$ denotes the substring of s from position a to (excluding) b. The function $\mathbf{I}$ is the indicator function, $\mathbf{I}(\mathrm{true}) = 1$, $\mathbf{I}(\mathrm{false}) = 0$ and the weights $v_j := d - j + 1$. We used a normalization of the kernel $\tilde{k}(\mathbf{s},\mathbf{s}') = \frac{k(\mathbf{s},\mathbf{s}')}{\sqrt{k(\mathbf{s},\mathbf{s})k(\mathbf{s}',\mathbf{s}')}}$ and $d = 22$ for the recognition of splice sites. Additionally, the regularization parameter of the Support Vector Machine was set to be $C = 2$ and $C = 3$ for acceptor and donor sites, respectively. All parameters (including the window size, regularization parameters etc) have been tuned on data set 2 (cf. Rätsch *et al.* (2005)).

Given a DNA sequence as target of an alignment we can now use the two SVMs to compute scores for each position with corresponding consensus[1] for being a splice acceptor or donor site, respectively. We consider the genomes of *C. elegans* and humans. U12 splicing in human is extremely rare and in *C. elegans* not present at all. We therefore do not consider `AT`/`AC` splice sites for our splice site model.

2.3 Smith-Waterman Alignments with Intron Model

The classical deterministic and exact alignment algorithm is the Smith-Waterman algorithm and is based on dynamic programming. Its running time is $\mathcal{O}(m \cdot n)$, where m is the length of the EST sequence S_E, and n is the length of the DNA sequence S_D. It builds up a $m \cdot n$ matrix and hence has the same space complexity.

The main idea of the algorithm is to compute a local alignment by determining the maximum over all alignments of all prefixes $S_E(1:i) := (S_E(1),\ldots,S_E(i))$ and $S_D(1:j) := (S_D(1),\ldots,S_D(j))$ of the two sequences S_E and S_D, while allowing for unaligned starts of the sequences (details below). An alignment is given by a sequence of pairs (a_r,b_r), $r = 1,\ldots,R$, where $R \leq m+n$ depends on the alignment and $a_r, b_r \in \Sigma := \{A, C, G, T, N, -\}$. A pair consists either of the two corresponding letters of the two sequences or a single letter in one sequence paired with a gap in the other sequence. The alignment is scored using a substitution matrix M, which we interpret as a function $M : \Sigma \times \Sigma \to \mathbb{R}$. Then the score for the alignment $\mathcal{A} = \{(a_r,b_r)\}_r$ is simply $\sum_r M(a_r,b_r)$.

We define $V(i_2,j_2)$ to be the score of the best alignment of prefixes $S_E(i_1:i_2)$ and $S_D(j_1:j_2)$ for the best choice of starting positions i_1 and j_1 on sequences E and D, respectively. The best local alignment can be obtained by finding the maximal entry in the matrix V determining i_2 and j_2 as the ends of the alignment. The matrix V can be computed using the following recurrence formula (for all $i = 1,\ldots,m$ and $j = 1,\ldots,n$):

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1,j-1) + M(S_E(i),S_D(j)) \\ V(i-1,j) + M(S_E(i),'-') \\ V(i,j-1) + M('-',S_D(j)) \end{cases} \tag{2}$$

[1] `AG` for acceptor sites and `GT` or `GC` for donor sites.

The recurrence is initialized with $V(0,0) := 0$, $V(i,0) := 0$ and $V(0,j) := 0$ for all $i = 1 \dots m$ and $j = 1 \dots n$. There are four possibilities:

- $S_E(1:i)$ and $S_D(1:j)$ are unaligned.
- $S_E(i)$ and $S_D(j)$ are aligned to each other (possibly a mismatch).
- $S_E(i)$ is aligned to a gap in the DNA sequence.
- $S_D(j)$ is aligned to a gap in the EST sequence.

In the original setting there are only these four possibilities and one can straightforwardly fill the matrix from left to right and top to bottom to finally compute the maximum over all elements in V. The optimal alignment can then be obtained by backtracking (Durbin *et al.*, 1998).

The Smith-Waterman algorithm only aligns the single bases of two sequences and does not distinguish between exons and introns – it essentially treats everything as exons. We therefore propose to extend the Smith-Waterman algorithm to better model introns: We allow an additional "intron transition" that is separately scored based on its length and the scores of splice sites at its ends. We denote by $f_I(i_1, i_2)$ the intron scoring function for an intron starting at i_1 and ending at i_2. The intron scoring function $f_I(i_1, i_2)$ is computed based on the intron length $i_2 - i_1$, the donor SVM output $g_{don}(i_1)$ for position i_1 and the acceptor SVM output $g_{acc}(i_2)$ for position i_2. During learning we determine three functions f_ℓ, f_{acc} and $f_{don} : \mathbb{R} \to \mathbb{R}$ to combine these three values:

$$f_I(i_1, i_2) = f_\ell(i_2 - i_1) + f_{don}(g_{don}(i_1)) + f_{acc}(g_{acc}(i_2)). \quad (3)$$

When there is no donor consensus at position i_1, then we define $f_{don}(g_{don}(i_1)) := -\infty$ (similarly for $f_{acc}(g_{acc}(i_2))$). Given the intron scoring function f_I we can now restate the recurrence formula (for all $i = 1, \dots, m$ and $j = 1, \dots, n$):

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1, j-1) + M(S_E(i), S_D(j)) \\ V(i-1, j) + M(S_E(i), '-') \\ V(i, j-1) + M('-', S_D(j)) \\ \max\limits_{j - L_{\max} \le k \le j-1} (V(i,k) + f_I(k,j)) \end{cases} \quad (4)$$

where we consider the additional possibility of an intron of maximal length $L_{\max}$ starting at position k and ending at j. Please note that the above recurrence formula is considerably more computationally expensive than the previous one: every step involves finding the optimal intron start ($\mathcal{O}(L_{\max})$), leading to the complexity of the dynamic programming algorithm of $\mathcal{O}(m \cdot n \cdot L_{\max})$. However, one only needs to consider those positions where the intron start and end exhibit the splice consensus sites and the splice site predictor scores are sufficiently large. Furthermore, if the intron length score only depends linearly on its length for introns longer than, say, L_{lin}, then the computational complexity can be reduced to $\mathcal{O}(m \cdot n \cdot L_{\text{lin}})$. Additional strategies for speeding up such algorithms and to reduce the amount of memory are discussed in Zhang and Gish (2006).

For completeness we need to extend our notation for alignments with introns. We use again alignment pairs $\mathcal{A} = \{(a_r, b_r)\}_r$, but extend the alphabet for a_r to $\Sigma \cup \{+\}$ ("intron sequence missing") and for b_r to $\Sigma \cup \{*\}$ ("intron sequence"). Note that b_r should only contain strings of length greater than one if $a_r = '+'$. Then the score $f(\mathcal{A})$ for an alignment $\mathcal{A}$ with intron is computed as before, i.e. $\sum_r M(a_r, b_r)$, except when $a_r = +$: In this case the intron score function $f_I(\cdot, \cdot)$ is used to score the corresponding intron.

2.4 Large Margin Combination

In the previous section we assumed that the functions f_{acc}, f_{don} and f_ℓ as well as the substitution matrix M were given. We now describe an algorithm to determine these parameters based on the training set of sequences and their true alignments.

Note that our proposed algorithm is two-layered: First one learns the splice site model and then how to combine the different pieces of information. In principle these two steps can be combined into one. Then the one-dimensional functions f_ℓ, f_{acc} and f_{don} can be replaced with linear combinations of kernel elements as similarly done in Altun *et al.* (2003). However, this makes training considerably slower and is not expected to improve the results in our case.

Since the three functions are one-dimensional, it suffices to use a simple piecewise linear model: Let s be the number of supporting points x_i (satisfying $x_i < x_{i+1}$) and y_i their values, then the piecewise linear function is defined by

$$f(x) = \begin{cases} y_1 & x \le x_1 \\ \frac{y_i(x_{i+1} - x) + y_{i+1}(x - x_i)}{x_{i+1} - x_i} & x_i \le x \le x_{i+1} \\ y_s & x \ge x_s \end{cases}. \quad (5)$$

After having appropriately chosen supporting points on the x-axis we only need to optimize the corresponding y-values. For f_{acc} and f_{don} we use 30 supporting points uniformly sampled between -5 and $+5$ (our SVM outputs are typically not larger). For f_ℓ we use 30 log-uniformly sampled supporting points between 30nt and 1000nt. Given the three functions and the substitution matrix, the alignment scoring function $f(\mathcal{A})$ is fully specified. Moreover, for a given alignment $\mathcal{A}$, it can be verified that $f(\mathcal{A})$ is linear in all parameters that we denote by $\boldsymbol{\theta}$, i.e. in the values of the substitution matrix and the y-values of the three piecewise linear functions, $\boldsymbol{\theta} := (\boldsymbol{\theta}_{acc}, \boldsymbol{\theta}_{don}, \boldsymbol{\theta}_\ell, \boldsymbol{\theta}_M)$.

2.4.1 Optimization For training we are given a set of N true alignments $\mathcal{A}_i^+$, $i = 1, \dots, N$. The goal is to find the parameters $\boldsymbol{\theta}$ of the alignment scoring function f such that the difference of scores $f_\theta(\mathcal{A}_i^+) - f_\theta(\mathcal{A}^-)$ is large for *all* wrong alignments $\mathcal{A}^- \neq \mathcal{A}_i^+$. This can be done by solving the following convex optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\xi} \ge \mathbf{0}, \boldsymbol{\theta}} \quad & \frac{1}{N} \sum_{i=1}^N \xi_i + \mathbf{P}(\boldsymbol{\theta}) \\ \text{s.t.} \quad & f_\theta(\mathcal{A}_i^+) - f_\theta(\mathcal{A}^-) \ge 1 - \xi_i \quad \forall i \text{ and } \mathcal{A}^- \neq \mathcal{A}_i^+. \end{aligned} \quad (6)$$

Here we introduced so-called slack-variables ξ_i to implement a soft-margin (Cortes and Vapnik, 1995), i.e. to allow for a few misaligned examples. Additionally, we use a regularization term to avoid over-fitting (see Section 2.4.2 for details). The above optimization problem has exponentially many constraints and cannot be easily solved directly. Instead one adopts a column generation technique (cf. Hettich and Kortanek (1993) and references therein) and for every true alignment one maintains a set of wrong alignments $\mathcal{A}_{i,j}^- \neq \mathcal{A}_i^+$, for all j. Initially this set is empty and elements are added iteratively. Given a set of wrong alignments one can now determine the intermediate optimal parameters $\boldsymbol{\theta}$ by solving:

$$\begin{aligned} \min_{\boldsymbol{\xi} \ge \mathbf{0}, \boldsymbol{\theta}} \quad & \frac{1}{N} \sum_{i=1}^N \xi_i + \mathbf{P}(\boldsymbol{\theta}) \\ \text{s.t.} \quad & f_\theta(\mathcal{A}_i^+) - f_\theta(\mathcal{A}_{i,j}^-) \ge 1 - \xi_i \quad \forall i, j. \end{aligned} \quad (7)$$

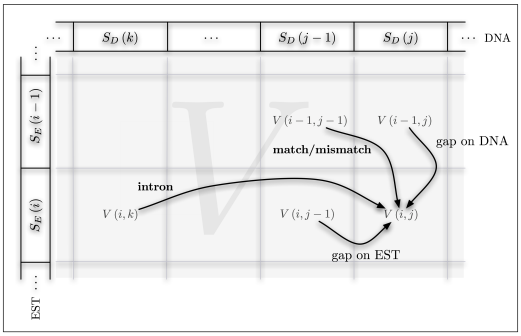

Fig. 2. A graphical representation of (4) showing the possible transitions to cell $V(i,j)$ in the alignment matrix. There is the additional possibility to start or finish the alignment at any position ("local alignment").

Using these parameters we can compute the best and the second best alignment, of which one must be wrong. We use the wrong one with the highest score for generating a new constraint in (7), i.e. we add it to the set of wrong alignments. This procedure is iterated and provably converges to the solution of (6) in a finite number of steps. In our application usually not more than 30 iterations are needed until all constraints are satisfied.

2.4.2 Regularization In empirical inference it is common to regularize the parameters in order not to over-fit. We implement this by adding a regularization term $\mathbf{P}(\boldsymbol{\theta})$ in (6). Recall that the parameter vector consists of four parts, and we define the regularization term as follows:

$$\mathbf{P}(\boldsymbol{\theta}) = C \left(\sum_{i=1}^{n-1} (\theta_{acc,i+1} - \theta_{acc,i})^2 + \sum_{i=1}^{n-1} (\theta_{don,i+1} - \theta_{don,i})^2 + \sum_{i=1}^{n-1} (\theta_{\ell,i+1} - \theta_{\ell,i})^2 + \sum_{a,b} M(a,b)^2 \right).$$

It implements the idea that the piecewise-linear functions should be smooth and the values in the substitution matrix small. The constant C is the regularization constant found by model selection and determines the trade-off between the number of training errors and the complexity of the model.

2.5 Whole Genome Alignments

Since the amount of computation and memory increases linearly with the length of the genomic sequence, it is not feasible to directly apply *PALMA* to align mRNAs to genomic sequences. *exalin* has the same problems and Zhang and Gish (2006) suggested to use *blast* to detect High-Scoring Pairs (HSPs) that can be used to significantly speedup the search. We follow a similar idea and use *blast* to identify HSPs with an e-value greater than 10^{-3}. In order to get complete alignments, we appropriately extend the HSPs' terminal ends and then apply *PALMA* to align this region with the query. Finally, the alignment with the greatest *PALMA* alignment score is chosen (among the several alignments generated for each HSP).

Both versions of our software are available for download at `http://www.fml.mpg.de/raetsch/projects/palma`. The software comes as a python package including scripts to align mRNA to DNA sequences or to perform a whole genome search. As output the tool provides the extended BLAT-like PSL format or the BED format for alignments against genomes.

The tool aligned all 4,358 cDNA sequences (4.7Mbases), with 14,058 *blast* hits against the *C. elegans* genome (version WS150, 100Mbases) in about 5 and 9 hours using the model without and with intron length information, respectively (on a 2.4 Ghz Athlon AMD64). This time included the whole genome prediction of splice sites took (about 30 minutes), aligning with *blast* (about 30 minutes) and processing the *blast* hits (1 or 2 seconds per hit on average).

3 RESULTS AND DISCUSSION

Most alignment algorithms work very well for aligning mRNA sequences against genomic DNA when query and target perfectly match and the matching blocks are long enough. In our experimental study we are interested in the most difficult cases, where most algorithms start to fail. If an algorithm works in such case we expect that it will also return correct alignments for easier cases. We consider two organisms, *C. elegans* and human, which have quite different characteristics. Introns in *C. elegans* are rather short (often only 50nt) and the splice sites are well conserved and relatively easy to recognize (data not shown). An intron on human DNA can span several hundred thousands of nucleotides and the splice sites are harder to recognize. Correct alignments to the human genome are considerably more challenging.

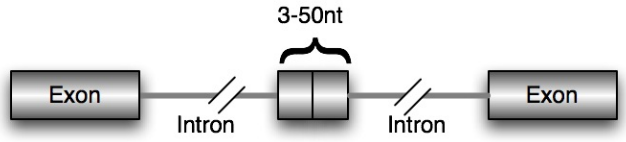

Fig. 3. For our simulation study we considered exon triples, where we artificially shorted the second exon to 3-50nt. Most alignment algorithms fail to reliably identify such short exons.

3.1 Experimental Setup

We evaluate our proposed method, *PALMA*, and other methods including *exalin*, *sim4* & *blat*. We consider the alignment of mRNA sequence fragments containing three exons, where we artificially shortened the middle exon (final length of 3-50nt; see Figure 3 and Appendix for details) in order to make the problem harder.[2] Artificially generating the data has the benefit of knowing exactly what the correct alignment has to be. Additionally, we add varying amounts of noise ($p = 0\%, 1\%, 5\%, 10\%$ of random mutations, deletions or insertions) to the query sequence. Finally, we replace a part of the DNA or mRNA sequence at its terminal ends with random sequence leading to a shortened correct alignment. This allows us to determine how well the methods perform in finding the correct *local alignment* including its boundaries. During evaluation we count how often the methods correctly identify the alignments. The evaluation is done on a separate test set which was not used during training of our method (set 4, cf. Appendix). This test set consists of 4358 sequences for *C. elegans* and 1344 sequences for human, giving a total of 5702 alignments in our evaluation.

The model selection for the splice site predictors have been performed on separate validation sets (set 2). Model selection of regularization parameter C in our method (cf. Section 2.4.2) was done by simple validation on a separate validation set (set 3). While the method was trained on noise-free data, we applied it to the noisy versions during validation since otherwise the validation error rate was very close to zero, almost independently of the choice of C. We determined $C = 0.001$ as optimal regularization constant. This choice performs well over a wide range of noise levels. However, for high noise levels there are better choices (see supplementary information on the project web site). Hence, if the noise level is known in advance the one should use the corresponding optimal value for C.

3.2 *PALMA* versus other alignment tools

Figure 4 shows the alignment error rates for different methods on the sequences from *C. elegans* and human. Here we count an alignment as correct, if all intron boundaries are predicted correctly. We can observe that there are drastic differences between the methods. Almost all methods perform reasonably well when the query perfectly matches the target – with the exception of *sim4* having problems to correctly align a relatively large part of the sequences. For *blat* and *sim4* the error rates drastically increase when the query sequence is noisy. For *blat* this behavior is expected as the matching heuristic requires long blocks of identity. Only *exalin* and *PALMA* have low error rates for relatively high noise levels. When deleting, inserting or mutating up to 10% of the query sequence, *PALMA* still identifies 97.8% of all introns correctly, while the other methods achieve less than 90% accuracy.

[2] We excluded exons of length two since *exalin* was not able to predict them.

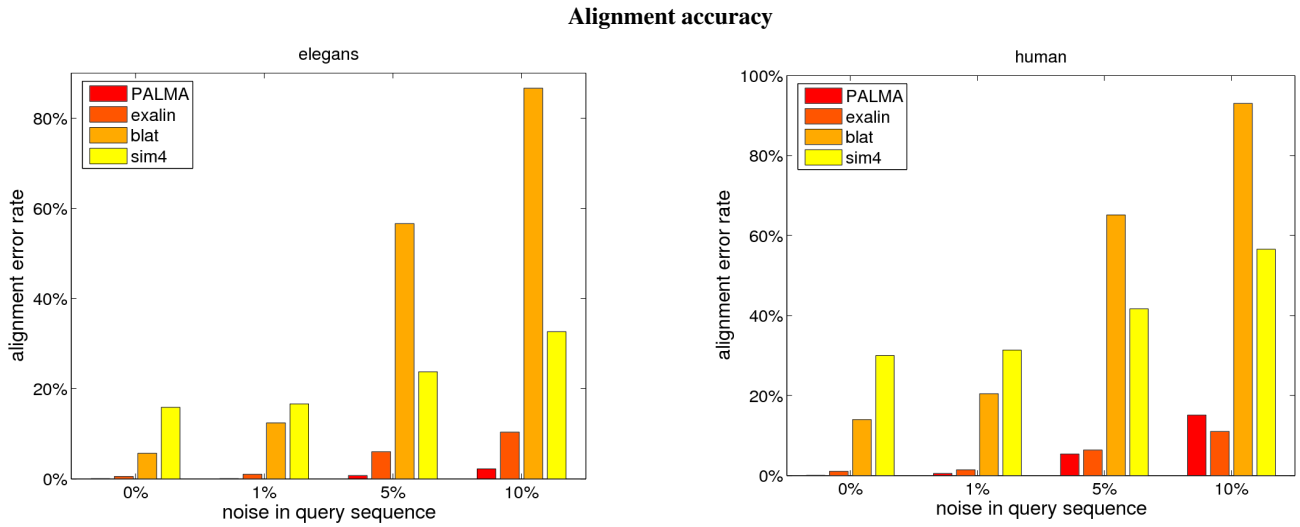

Fig. 4. Comparison of different methods for aligning mRNAs to genomic DNA: *C. elegans* on the left and human on the right. We considered the particularly difficult task of aligning exon triples with short middle exons (3-50nt) in the presence of different amounts of noise. An alignment is declared as correct if the intron boundaries, i.e. all splice sites, are correct. *PALMA* has significantly lower error rates than all other methods throughout all noise levels (exception: human with noise levels; see main text for details).

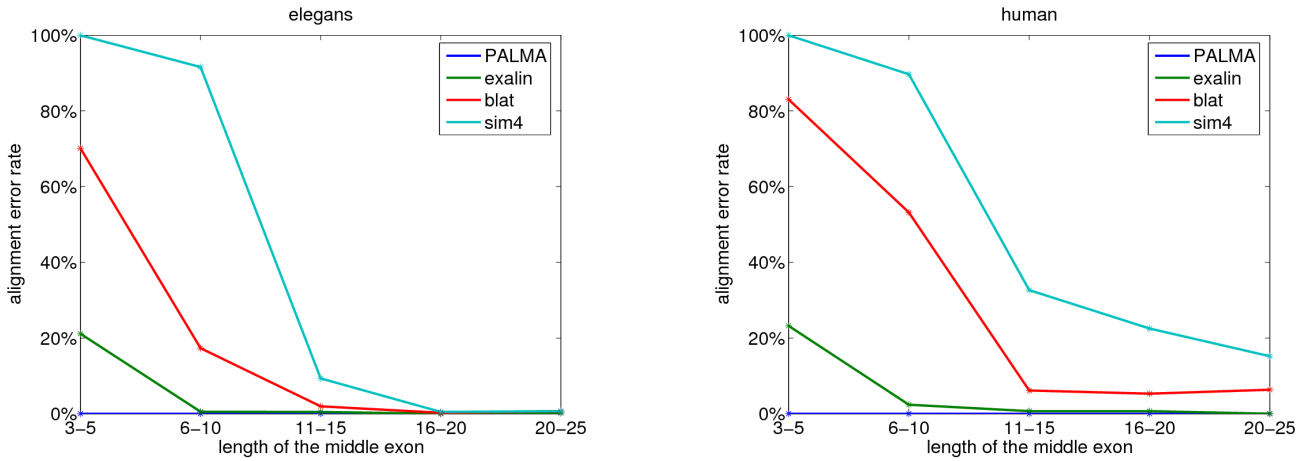

Fig. 5. Comparison of different methods for aligning mRNAs to genomic DNA: *C. elegans* on the left and human on the right. The error rates at different lengths of the middle exon are shown for the data with no noise. For *C. elegans* all methods perform almost perfectly for exons longer than 20nt. However, for human, there is some residual error for *blat* and *sim4* even for longer exons. For both organisms, the different algorithms begin to fail as the middle exons become shorter, except for *PALMA*.

We have also considered error rates for the whole alignment (not shown, but displayed on the supplementary website). Here we counted an alignment as correct, if the intron boundaries as well as the 5' and 3' ends are exactly correct. Not entirely surprising, we observe that it is increasingly difficult to identify the correct ends of the alignment if the amount of noise is increased. While *blat*, *exalin* and *PALMA* perform about equally good for low noise levels, *sim4* often fails to identify the correct start or end of the alignment.[3]

Finally, Figure 5 illustrates one of the main reasons for the rather poor performance of the other methods. Shown is the accuracy (intron boundaries) of aligning exon triples with middle exons of varying length (3-5nt, 6-10nt, 11-15nt, 16-20nt, 20-25nt). We can observe that *sim4* is most affected by the shortness of the exons, closely followed by *blat*. For very short exons ($\leq$5nt) *exalin* produces wrong alignments in a significant number of cases. *PALMA*'s performance is essentially unaffected by the length of the exon (at least in the noise-free case): Out of the 5702 sequences (*C. elegans* and human), *PALMA* only failed on two sequences to produce the correct alignment.

3.3 Splice site scoring and intron length modeling affect performance

We investigate the different features used by *PALMA* and how they contribute to the improvement of the alignment accuracy. The same data was used as before (*C. elegans* only). The error rates for full *PALMA*, *PALMA* with splice site scores disabled, *PALMA* with

[3] We observed that the alignments are up to 15 nucleotides too short on each end.

the intron length model disabled and with both disabled are shown in Figure 6. For comparison we also show the results of *exalin*.

The higher the noise level, the less information is actually available in the query mRNA sequence. Hence, the splice sites help more to identify the introns, as can also be observed in Figure 6. But also in the low noise case, the splice site predictions also help to accurately identify very short exons that may be matched to several positions in the intronic regions (cf. Figure 6). If splice site information not available, then the error rate increases quite drastically (large effects for noise $\geq$5%).

If the splice site information is available, the intron length only contributes little to reducing the error rate. For higher noise levels ($\geq$2%) we even find that the intron length information can be harmful, which can be explained that *PALMA* was trained on noise free sequences. Hence, it might not have learned good parameters for balancing the matches with the intron length for the noisy case. Furthermore, taking the intron length into account is computationally rather expensive, i.e. $\mathcal{O}(m \cdot n \cdot L_{\text{lin}})$ instead of $\mathcal{O}(m \cdot n)$. In practice, we therefore suggest to use *PALMA* with splice site predictions, but without intron length model, leading to only a small loss of accuracy compared to the best version.

Finally, the question remains why *exalin* is performing worse than *PALMA* without intron and splice site model, which we found very surprising. It turns out that exalin often has problems aligning the terminal ends of the sequences (see Figures 5 and 4 and tables in supplementary material). Moreover, it often fails to identify very short exons (length 2-6nt.), in particular it does not predict exons of length shorter than three. Finally, we noticed that for higher noise levels in a few percent of the cases *exalin* produces alignments where the intron boundaries are shifted by 2nt, even when there is no consensus. All these issues lead to a considerably worse performance of *exalin* compared to *PALMA*.

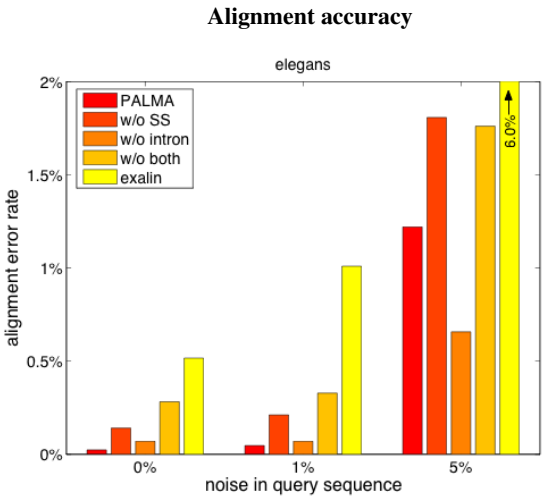

Fig. 6. Shown is the accuracy (considering intron boundaries only) of *PALMA* and *exalin* (as in Figure 5), as well as *PALMA* with three different handicaps: with the splice site model disabled (w/o SS), with the intron model set to a constant function (w/o intron), and with both disabled (w/o both).

3.4 Illustration of Learning Result

Figures 7 and 8 show the optimized parameters determined by our algorithm. For the piece-wise linear functions for acceptor and donor scores in Figure 8 we obtain smooth sigmoid-shaped functions ("differences between very large or very small score values do not matter"). The same figure displays the piece-wise linear function for scoring intron lengths. We observe that the very short and very long introns are penalized. Interestingly, the maximum does not correspond to the most frequent exon length (50nt in *C. elegans*).

The optimized substitution matrices (Figure 7) are essentially diagonal, which is not surprising as there was no preference for substitutions in our data. Comparing the acceptor and donor scores in Figure 8 with the substitution matrices we observe that the difference between a weak and a strong splice site is worth about 3-5 matches. Furthermore, since we use the Smith-Waterman algorithm, the terminal ends are determined by the balance between scores for insertions or deletions and matching positions as well as between mismatches and matches. Whenever, the score at the ends are below 0, the alignment is shortened. For *C. elegans* one allows for about one gap per 4 matches, while for human it is about one gap in 5-9 nucleotides.

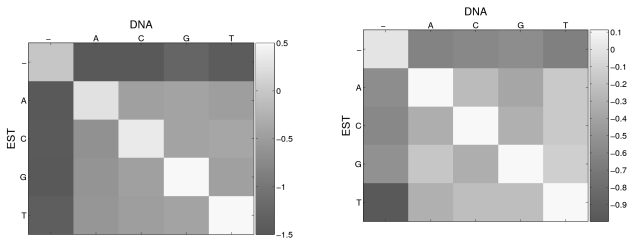

Fig. 7. Shown is the optimized substitution matrix for elegans (left) and human (right): matches score high and gaps low. Mismatch scores are all close to zero and do not contribute much.

4 CONCLUSION

We have proposed a new alignment algorithm that computes the optimal alignment of mRNA sequences to genomic DNA. It exploits very accurate kernel-based splice site predictions approaches and makes use of an intron length model. For combining the different pieces of information, we have developed a novel optimization method that simultaneously optimizes parameters such as the substitution matrix, intron length penalty function etc. The algorithm is based on maximizing the margin between true and wrong alignment by solving a convex optimization problem. In a thorough simulation study on aligning sequences from *C. elegans* and human with very short exons and varying amounts of noise we have shown that our method achieves significantly lower error rates than other methods. This indicates that the proposed method would be more effective than current approaches for discovering micro-exons, i.e. exons between 3-25nt in length. This is especially true in the presence of sequencing errors or mutations which may render current approaches inaccurate. In addition, by combining it with other methods such as *blast* we can reduce the computational cost in order to apply our method for alignments of ESTs to whole genomes.

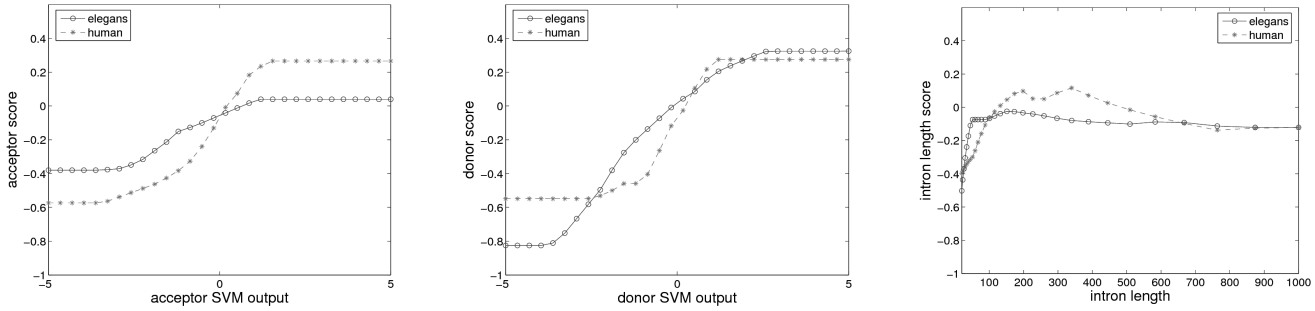

Fig. 8. *PALMA*'s optimized functions f_{acc} and f_{don} scoring acceptor and donor SVM outputs. The right figure shows is the optimized intron length scoring function f_ℓ.

ACKNOWLEDGMENTS

We thank D. Huson, G. Myers, A. Zien, S. Sonnenburg, and K.-R. Müller for helpful comments and suggestions on this work. Additionally, we thank G. Schweikert and N. Toussaint for proofreading the manuscript.

APPENDIX

Processing of Sequence Databases

Sequences from C. elegans*:* We collected all known *C. elegans* ESTs from Wormbase (Harris *et al.*, 2004) (release WS120; 236,893 sequences) and dbEST (Boguski and Tolstoshev, 1993) (as of February 22, 2004; 231,096 sequences). Using *blat* (Kent, 2002) we aligned them against the genomic DNA (release WS120). The alignment was used to confirm exons and introns. We refined the alignment by correcting typical sequencing errors, for instance by removing minor insertions and deletions. If an intron did not exhibit the consensus GT/AG or GC/AG at the 5' and 3' ends, then we tried to achieve this by shifting the boundaries up to 2 base pairs (bp). If this still did not lead to the consensus, we split the sequence into two parts and considered each subsequence separately. In a next step we merged consistent alignments, if they shared at least one complete exon or intron. This led to a set of 124,442 unique EST-based sequences.

We repeated the above procedure with all known cDNAs from Wormbase (release WS120; 4,855 sequences). These sequences only contain the coding part of the mRNA. We used their ends as annotation for start and stop codons.

We clustered the sequences in order to obtain independent training, validation and test sets. In the beginning each of the above EST and cDNA sequences were in a separate cluster. We iteratively joined clusters, if any two sequences from distinct clusters match to the same genomic location (this includes many forms of alternative splicing). We obtained 21,086 clusters, while 4072 clusters contained at least one cDNA.

For *set 1* we chose all clusters not containing a cDNA (17215), for *set 2* we chose 40% of the clusters containing at least one cDNA (1536). For *set 3* we used 20% of clusters with cDNA (775). The remaining 40% of clusters with at least one cDNA (1,560) were used as *set 4*. Sets 2-4 were filtered to remove confirmed alternative splice forms. This left 1,177 cDNA sequences for *testing* in set 4 with an average of 4.8 exons per gene and 2,313bp from the 5' to the 3' end.

Sequences from Human: We followed the same protocol using the version hg17 of the human genome and sequences from dbEST (as of July 14, 2005) and cDNAs from the RIKEN and MIPS cDNA collections. We obtained four sets as before. The first set was used to train the splice site recognizers (29748 clusters), the second set (1558 clusters) was use for computing the alignment parameters, the third set (780 clusters) for model selection and the fourth set (1560 clusters) for final evaluation. As before sets 2-4 were filtered to remove confirmed alternative splice forms leaving considerably few sequences as for *C. elegans*, which additionally often where single exon genes.

Artificial Micro-exon Dataset

Based on sets 2-4 for *C. elegans* and human described in the last section we created sets of consecutive exon triples from the confirmed transcripts in these sets. This lead to 4604, 2257 and 4358 triples for *C. elegans* as well as 1277, 669 and 1344 for human. In a first processing step we shortened the middle exons to a random length between 2nt and 50nt (uniformly distributed). To do so, we removed the correct number of nucleotides from the center of the middle exon – from the query as well as the DNA. This leaves the splice sites mostly functional. Since exalin was not able to detect exons shorter than 3nt., we exclude those from the further analysis. In a second step we added varying amounts of noise. For a given noise level p and a query sequence of length L, we first replaced $p \cdot L/3$ positions with a random letter ($\Sigma = \{A, C, G, T, N\}$). Then we deleted the same number of non-overlapping positions in the sequence and added the same number of random nucleotides at random positions. We used $p = 0\%, 1\%, 5\%, 10\%$. Finally, we randomly removed or replaced up to 15nt at the terminal ends. Replacement was done using a random string making sure that the first replaced letter is a mismatch.

REFERENCES

Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal Molecular Biology*, **215**(3), 403–10.

Altun, Y., Tsochantaridis, I., and Hofmann, T. (2003). Hidden markov support vector machines. In *Proc. 20th International Conference on Machine Learning*.

Berg, O. and von Hippel, P. (1987). Selection of dna binding sites by regulatory proteins. statistical-mechanical theory and application to operators and promoters. *J. Mol. Biol.*, **193**, 723–750.

Boguski, M. and Tolstoshev, T. L. C. (1993). dbEST–database for "expressed sequence tags". *Nat Genet.*, **4**(4), 332–3.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, **20**, 273–297.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic models of protein and nucleic acids*. Cambridge University Press. 7th edition.

Florea, L., Hartzell, G., Zhang, Z., Rubin, G., and Miller, W. (1998). A computer program for aligning a cdna sequence with a genomic dna sequence. *Genome Research*, **8**, 967–974.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139.

Gelfand, M., Mironov, A., and Pevzner, P. (1996). Gene recognition via spliced sequence alignment. *Proc. Natl. Acad. Sci.*, **93**(17), 9061–6.

Gusfield, D., Balasubramanian, K., and Naor, D. (1994). Parametric optimization of sequence alignment. *Algorithmica*, **12**, 312–326.

Harris, T., Chen, N., Cunningham, F., *et al.* (2004). Wormbase: a multi-species resource for nematode biology and genomics. *Nucl. Acids Res.*, **32**. Database issue:D411-7.

Hettich, R. and Kortanek, K. (1993). Semi-infinite programming: Theory, methods and applications. *SIAM Review*, **3**, 380–429.

Joachims, T., Galor, T., and Elber, R. (2005). Learning to align sequences: A maximum-margin approach. In B. Leimkuhler, C. Chipot, R. Elber, A. Laaksonen, and A. Mark, editors, *New Algorithms for Macromolecular Simulation*, number 49 in LNCS, pages 57–71. Springer.

Kececioglu, J. and Kim, E. (2006). Simple and fast inverse alignment. In *RECOMB*, pages 441–455.

Kent, W. J. (2002). BLAT–the BLAST-like alignment tool. *Genome Res*, **12**(4), 656–664.

Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNCS, pages 119–184. Springer.

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, **12**(2), 181–201.

Rätsch, G., Sonnenburg, S., and Schölkopf, B. (2005). RASE: recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics*, **21**(Suppl. 1), i369–i377.

Rätsch, G., Sonnenburg, S., and Schäfer, C. (2006a). Learning interpretable svms for biological sequence classification. *BMC Bioinformatics*, **7**(Suppl 1), S9.

Rätsch, G., Hepp, B., Schulze, U., and Ong, C. (2006b). Palma: Perfect alignments using large margin algorithms. In D. Huson, O. Kohlbacher, A. Lupas, K. Nieselt, and A. Zell, editors, *German Conference on Bioinformatics*, volume P-83 of *Lecture Notes in Informatics*, pages 104–113, Tübingen, Germany. Springer.

Rätsch, G., Sonnenburg, S., Srinivasan, J., Witte, H., Müller, K.-R., Sommer, R., and Schölkopf, B. (2007). Improving the c. elegans genome annotation using machine learning. *PLoS Computational Biology*, **3**(2), e20. http://dx.doi.org/10.1371/journal.pcbi.0030020.eor.

Sonnenburg, S., Rätsch, G., Jagota, A., and Müller, K.-R. (2002). New methods for splice-site recognition. In *Proc. International Conference on Artificial Neural Networks*.

Sonnenburg, S., Philips, P., Schweikert, G., and Rätsch, G. (2007). Accurate splice site recognition using svms. *BMC Bioinformatics*. accepted.

Stormo, G. (1988). Computer methods for analyzing sequence recognition of nucleic acids. *Annu. Rev. Biophys. Biophys. Chem*, **17**, 241–263.

Usuka, J., Zhu, W., and Brendel, V. (2000). Optimal spliced alignment of homologous cdna to a genomic dna template. *Bioinformatics*, **16**(3), 203–211.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer Verlag, New York.

Volfovsky, N., Haas, B., and Salzberg, S. (2003). Computational discovery of internal micro-exons. *Genome Research*, **13**, 1216–1221.

Wheelan SJ, Church DM, O. J. (2001). Spidey: a tool for mrna-to-genomic alignments. *Genome Research*, **11**(11), 1952–7.

Zhang, M. and Gish, W. (2006). Improved spliced alignment from an information theoretic approach. *Bioinformatics*, **22**(1), 13–20.