Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
BioSystems 87 (2007) 117–124
BioSystems
www.elsevier.com/locate/biosystems

Identifying DNA splice sites using hypernetworks with artificial molecular evolution

Jose L. Segovia-Juarez[a], Silvano Colombano[b,*], Denise Kirschner[a]

[a] *Department of Microbiology and Immunology, University of Michigan, Ann Arbor, MI, USA*
[b] *Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035, USA*

Received 28 February 2005; received in revised form 8 July 2006; accepted 15 July 2006

Abstract

Identifying DNA splice sites is a main task of gene hunting. We introduce the hyper-network architecture as a novel method for finding DNA splice sites. The hypernetwork architecture is a biologically inspired information processing system composed of networks of molecules forming cells, and a number of cells forming a tissue or organism. Its learning is based on molecular evolution. DNA examples taken from GenBank were translated into binary strings and fed into a hypernetwork for training. We performed experiments to explore the generalization performance of hypernetwork learning in this data set by two-fold cross validation. The hypernetwork generalization performance was comparable to well known classification algorithms. With the best hypernetwork obtained, including local information and heuristic rules, we built a system (HyperExon) to obtain splice site candidates. The HyperExon system outperformed leading splice recognition systems in the list of sequences tested.
© 2006 Elsevier Ireland Ltd. All rights reserved.

Keywords: DNA splice sites identification; Artificial evolution; Hypernetwork learning; Molecular networks

1. Introduction

Identification of DNA splice sites in large DNA databases is a main task of gene hunting. Genomes from most eukaryotic chromosomes contain both exon and intron sites. Exons are DNA sections that code for proteins, while introns are DNA sections that are removed following mRNA synthesis (Watson et al., 1987). The division between an exon and intron is called a splice junction (intron–exon, and exon–intron in Fig. 1). DNA is transcribed into pre-mRNA, then, during RNA processing, excision of introns, splice of exons, and synthesis of the *cap* and the *poly*(*A*) tail occurs.

Exons provide the information for mRNA to synthesize the peptide during the translation phase in the cytosol.

Gene identification in large DNA sequences involves the location of the start and stop DNA triplets (codons), exons, and introns. Typically the initial and ending sites of genes have well-defined patterns. Moreover, there are some rules for intron–exon boundaries that help to define splice sites, but this is insufficient to solve the issue completely. The problem is to discover nucleotide patterns that serve as true splice junctions in the DNA sequence over random DNA substrings. Thus, the task is to recognize exon–intron boundaries (EI or donor sites) and intron–exon boundaries (IE or acceptor sites) from large DNA sequences. This can be treated as a classification problem.

The precision of gene identification computing algorithms depends on the accuracy of locating exons and

* Corresponding author.
E-mail address: scolombano@mail.arc.nasa.gov (S. Colombano).

0303-2647/$ – see front matter © 2006 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2006.09.004

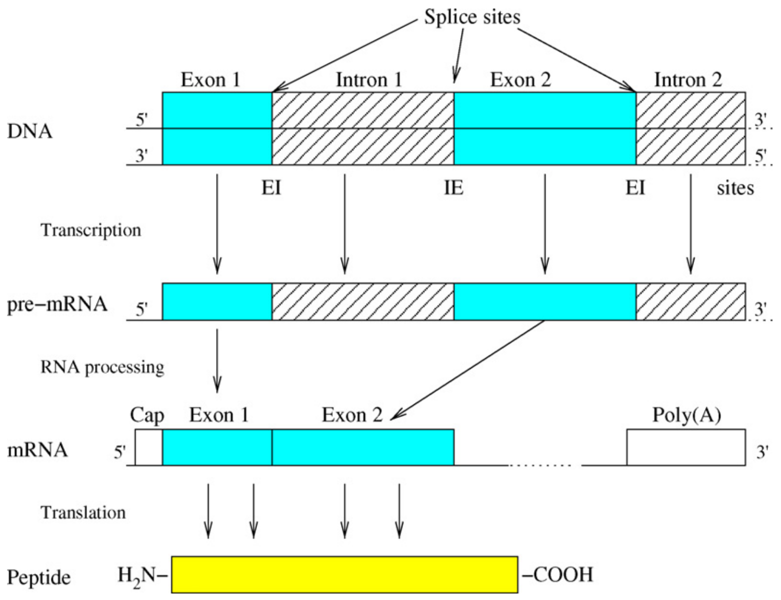

Fig. 1. DNA splice sites and their role in transcription and translation process of DNA processing and peptide synthesis. EI and IE are exon–intron and intron–exon sites, respectively.

introns in the genome database. Several approaches have been published: Noordewier et al. (1991) published KBANN that is a combination of neural networks and a knowledge based system; Xu et al. (1994) described GRAIL II, a hybrid of a rule-based system and neural networks; Kulp et al. (1996) and Henderson et al. (1997) used Hidden Markov Models to build the Genie architecture; Rampone (1998) described BRAIN using Boolean expressions, neural networks and discriminant analysis; and Fu (1999) used a symbolic system and a neural network in their MYCIN architecture. More recently, Pertea et al. (2001) implemented the "GeneSplicer" combining a Markov model with a decision tree; Howe et al. (2002) created GAZE, a dynamic programming based architecture; and Fogel et al. (2003) used for the first time an evolutionary neural network to address the problem. There are many other methods whose review is outside the scope of this paper. It is quite clear that the complexity and diversity of genomic structures in eukaryotic organisms will likely allow the use of several methods targeted to particular problems (Mathe et al., 2002).

Our approach is to develop an evolutionary method with the hypernetwork architecture of biological information processing (Segovia-Juarez and Conrad, 2001). This is a novel machine learning methodology that is just beginning to be applied to classification problems, and, as demonstrated here, is also suitable for addressing the splice site recognition problem. It uses a biologically inspired architecture that has representations of "molecules", "cells", and a set of cells forming a "tissue" or "organism". Molecules have a binary representation and molecular interactions are based on string matching. The dynamics occur through formation of networks of molecular interactions. The learning algorithm is based on simulated molecular evolution (see Section 2 below).

Here we demonstrate the learning ability of the hypernetwork architecture in solving the splice problem for a popular genome dataset; we study the generalization properties of the hypernetwork on this data, and the performance of this algorithm in comparison with others. Using the best network obtained during training, we implemented a splice site recognition system that was tested with a list of human DNA sequences, and finally we report our conclusions.

2. Methods

2.1. The hypernetwork architecture

The hypernetwork architecture is a biologically inspired learning model based on abstract molecules and molecular interactions that exhibits functional and organizational correlation with biological systems.

This is a multi-scale architecture that comprises three hierarchical levels: organism, cellular, and molecular. Molecules are an abstract of enzyme-like structures, and interactions occur as typical activation and inhibition processes. The representation of molecules and their interactions are comprised of binary strings and string matching, respectively. Molecules have an "excitatory receptor domain", an "inhibitory receptor domain",

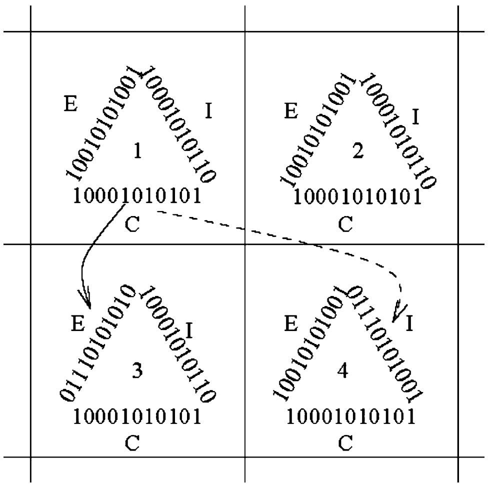

Fig. 2. Representation of four "molecules" in a cell. Each molecule has an excitatory receptor domain (E), inhibitory receptor domain (I), and catalytic domain (C). For the experiments, each domain is set to 20 bits (only some are shown). There is an example of activation of molecule nos. 1–3 (solid arrow), and inhibition of molecule nos. 1–4 (dashed arrow). Activations or inhibitions occurs when the catalytic domain of an activated molecule matches – above a threshold – the excitatory or inhibitory domain of the target molecule.

and a "catalytic domain", with 20 bits each, giving a total of 60 bits/molecule.

A molecule can be activated or inhibited by neighbor molecules through its excitatory or inhibitory domain, respectively (see Fig. 2). The interactions are dynamically formed by the catalytic site of a molecule and the excitatory or inhibitory sites of the target molecule. An interaction is formed if the binary matching is above a threshold value lasting just one time step.

Molecules are placed in cells, which are modeled by cellular automata, and an organized group of cells forms an organism (see Fig. 3). Each cell has three types of molecules: receptor molecules that gather information from the environment or from external molecules, effector molecules that interact with receptor molecules of target cells, and internal molecules.

Cell to cell interactions are produced by the effector–receptor molecules of the cells. Biosystems have several examples of excitatory as well as inhibitory effector–receptor interactions. For example, in the immune system, inhibitory receptors on lymphoid and myeloid cells are very important in modulating the immune response. The disruption of inhibitory receptor activity results in fatal autoimmune disorders (Ravetch and Lanier, 2000).

The hypernetwork is organized with four layers of cells: input cells, that get information from the environment, two layers of internal cells, and a layer of output cells from which information is obtained. This organization was derived empirically. Since the tissue has 60 input cells, we used small input cell sizes to improve the simulation time. Other types of architectures, and their impact on learning are currently being investigated.

The hypernetwork receives environmental influences from its input cells, and delivers a signal from its output cells. Internal cells interact with input and output cells. The input vector is split into two bit strings, and each substring activates a receptor molecule in each input cell (complementary to its first two bits), triggering cascades of molecular interactions inside the cells of the organism. The cascade of interactions continue through the network until an effector molecule of an output cell is activated, or 15 time steps passed after the initial feeding of each input vector. At this point, the output is obtained from the state of readout structures on the output cells. For every output cell we check the states of the molecules where the readout structures resided. If the molecule where the readout structure resides was activated during the simulation, then the output of the corresponding cell is a "1", otherwise the output is "0". The output vector is formed by concatenation of output cell states. Therefore, a given input vector will produce a specific output vector.

2.2. *The learning algorithm*

Hypernetwork organisms learn classification tasks by an adaptive algorithm based on molecular evolution. This is a bottom-up approach. Molecular entities form dynamic networks of interactions that affect the organismic level and allow an organism to perform a selected task (produce a desired output). An organism is reproduced with random molecular mutations (variation), and selection is used to remove those organisms with the least appropriate molecule structures, leaving those that remain to serve as parents for the next generation of solutions for the problem to be solved. For the described results we used a population of two organisms, since our software was restricted to such number. Current implementations, however, allows the use of up to 1000 individuals or more.

During "variation", molecules have a small chance of mutation ("molecular mutation probability"), and if the molecule is selected, a percentage of its bits are changed ("percentage of intramolecular rearrangement") randomly choosing between "0" or "1". These parameters, as well as activation and inhibition threshold values are set at the beginning of the simulation and do not change during its course. Once an organism is formed, we feed it with each input vector, trigger the cascades of interactions in the hypernetwork and we obtain the output vector. We find the Hamming distance between the desired output vector (D) and the output vector (O) obtained from the network. The sum of the distances is used to find the training performance of the generation (performance measure). This process is iterated for a number of generations, or until the system fully learns the task. The learning algorithm is shown in Fig. 4.

An example of a hypernetwork organism is shown in Fig. 3. This organism was used to perform the experiments reported here. The input was translated from nucleotide notation (A, T, G, C) into binary, which was used to form the binary input vector, and the figure shows a sample of all possible cell to cell interactions. The parameters for the system are: threshold for activation = 60%, threshold for inhibition = 60%, molecular mutation probability = 0.009, percent-

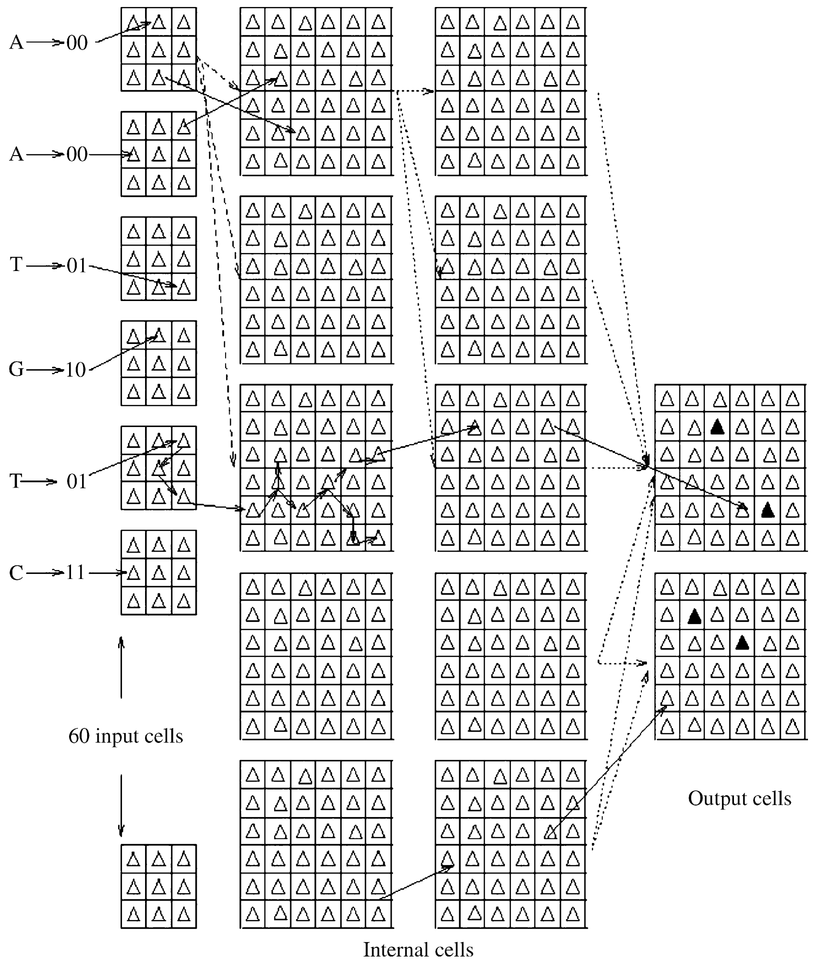

Fig. 3. The hypernetwork organism used for training. The organism has 60 input cells, two layers of internal cells with five cells each, and two output cells. Each input cell has nine molecules comprised of four receptor, two effector, and three internal molecules. Internal and output cells have 36 molecules comprised of four receptor, four effector, and 28 internal molecules. Each layer has the potential to be fully connected to the adjacent layer (dotted lines). The cell to cell interactions are dynamically formed by effector molecules to receptors of target cells. The cascade of interactions (solid lines are a sample) is originated by the input vectors applied to the input cells. A two-bit substring from the input vector, activates a receptor molecule in each input cell, triggering a cascade of molecular interactions through the organism. Each output cell will answer "1" or "0" based on the state of their readout structures (two per cell, shown filled).

age of intra-molecular rearrangement = 40%, molecular size (shape) = 20 bits/domain. The parameter file and the source code of the training and testing programs can be requested from the corresponding author.

During training Hamming distance metric was used. With this metric we give credit to partial matching on the output. However, during testing we counted only 100% correct answers as positive, otherwise the answer was negative. Accuracy was defined as the total of true positive cases divided by the total number of instances.

2.3. Data description

We used primate splice-junction DNA sequences taken from GenBank. This dataset has been used by several authors. Even though there are other gene splice datasets, this dataset was chosen as it allows comparison with many other machine learning algorithms.

The original dataset we chose is found at the UCI repository of machine learning databases.[1] This data is also part of the Statlog dataset (Michie et al., 1994). The data set obtained is comprised of 3186 entries. Each entry is formed by 60 nucleotides and one attribute to label the class. The splice site is in the middle of the DNA sequence, so there are 30 nucleotides before and after the splice site. Each instance can belong to one of three classes: EI (exon–intron boundary), IE (intron–exon boundary) or *N* (neither of them).

[1] http://www.datalab.uci.edu/data/mldb-sgi/data/.

```
1: Input data: The Input vectors (I) of length $k$, and Desired output vectors (D).
2: Initialize an organism with molecules created randomly ($O_{best}$).
3: **repeat**
4:    *Variation:* reproduce the organism ($O_{best}$) with mutation to generate a mutant.
5:    **for** The mutant **do**
6:        **for** Every pair (input vector $I_i$, desired output vector $D_i$), from **n** vectors **do**
7:    Read input vector $I_i$ into **input cells**
8:        **repeat**
9:    Propagate interactions through cells
10:       **until** Effectors of output cells are activated or 15 time steps passed since input cells
      activation.
11:   Read output vector $O_i$ from the **output cells**
12:   **end for**
13:   Evaluate the performance of the mutant: $P_o = 1 - (\sum_{i=1}^{n} (O_i - D_i)*100/k)/(100*n)$.
14:   **end for**
15:   *Selection:* select the performer $O_{best}$ (between the parent and the mutant).
16: **until** ($P_{best}$ = 1) or termination condition
```

Fig. 4. The variation-selection algorithm for hypernetwork learning. Modified from Segovia-Juarez and Colombano (2006).

Noordewier et al. (1991) and Towell and Shavlik (1992) used these data with a knowledge-based neural network named KBANN; Fu (1999) used these data with a system that combined a symbolic expert system and a feed-forward multilayer neural network; Rampone (1998) used the dataset with an algorithm based on complex relevance rules.

Since the input and output of the hypernetwork are in binary, we encoded DNA data and their classes into a binary representation. The binary representation of DNA nucleotides A, T, G and C were "00", "01", "10", and "11", respectively. The splice site classes: exon–intron (donors), intron–exon (acceptors), and neither, were encoded into "10", "01", and "11", respectively. Although other data representations may be used, this representation was selected for simplicity.

From the 3186 entries, the dataset was already split into two sets. A set for training with 2000 instances, and a set for validation with 1186 instances. In our experiments we used the same partitions as the train and validation sets.

3. Experiments

We ran experiments to study the effect of the size of the training set on generalization, the overall generalization with a two-fold cross validation experiment. Then we built a splice recognition system.

3.1. Generalization

We studied the effect of the size of the training set on generalization accuracy, which is an important feature of the algorithm since we wanted to know both how well the learning architecture captured the features during training, and how well it performed when the size of the input data increased. We also report its performance over time.

From the training data set of 2000 entries, we randomly obtained five subsets of 100, 400, and 1000 entries as shown in Table 1. We also trained the hypernetwork organism with the complete training set of 2000 entries. For the experiments A–C, we ran only up to 30,000 generations due to time limitations, and for experiment D we ran up to 50,000 generations.

Since the learning algorithm is stochastic, we ran it twice with each subset obtained, and evaluated the accuracy of the best trained hypernetwork with the validation set. Then, we evaluated the average performance of the five subsets. The number of generations was constrained by computer time and it was evaluated from preliminary experiments with the training set. We did not use the validation set during training in any way.

3.2. Two-fold cross validation

We estimated the overall generalization performance of the hypernetwork architecture by using two-fold cross validation. We randomly divided the 3186 dataset entries into two sets of 1593 entries each, and used one as the training set and the other as the validation set, and vice versa. We ran the simulations twice due to the stochastic nature of the algorithm, for 80,000 generations, and with the best network found in each set we evaluated its performance on the validation set.

Table 1
Experimental setting

Set	No. of training instances	No. of subsets	Total no. of generations
Experiment A	100	5	30,000
Experiment B	400	5	30,000
Experiment C	1000	5	30,000
Experiment D	2000	–	50,000

Table 2
Generalization results

Set	Table 100 generations (s)	Average generations	Average train performance	CI (±)	Average test accuracy	CI (±)
A	10	24,576	0.998	0.002	0.760	0.033
B	33	30,000	0.988	0.004	0.887	0.014
C	76	30,000	0.978	0.003	0.912	0.013
D	147	30,000	0.975	–	0.922	–
D	147	50,000	0.977	–	0.927	–

CI is the confidence interval (95% confidence), $n = 5$.

3.3. *The HyperExon recognition system*

We built the first version of a splice site recognition system (HyperExon), by using the best hypernetwork obtained during training, from experiment D. The hypernetwork learned global information from the length of the input string, 30 nucleotides at each side of the splice site. In addition to the hypernetwork, and in order to refine the output, the system also needed both information near the splice site and heuristic rules to create exon candidates from the list of splice sites obtained.

Local information was obtained from patterns of nucleotides near the splice sites using nucleotide occurrences frequencies (Stephens and Schneider, 1992). We employed a nucleotide frequency matrix from Stephens and Schneider (1992), for the locations starting at −5 to 4 of the exon–intron sites, and locations −13 to 4 of the intron–exon sites. The information present in each location was used to calculate a nucleotide consensus index. We averaged the consensus index with the global information obtained from the hypernetwork to get scored splice site candidates with weights between 0.0 and 1. Thus, we filtered a large proportion of the false positives that splice recognition procedures often produce (Fu, 1999; Xu et al., 1996). The HyperExon recognition system[2] was tested on 50 human loci obtained from Fu (1999). The data chosen was independent from the training data so that there would be no bias on the results of the study. The 50 loci had a total of 84 introns and 36,927 base pairs (bp) in length. About 20% of the loci had less than 200 bp, and 22% had more than 1500 bp.

We compared the performance of the splice site candidates HyperExon generated with Gene-Splicer (Pertea et al., 2001) and NetGene2 (Brunak et al., 1991; Hebsgaard et al., 1996), leading splice site prediction systems.We used their default parameter settings and their human databases. According to Pertea et al. (2001), GeneSplicer outperformed other recognition systems such as NetGene2, HSPL, NNSPlice, GENIO, and SpliceView.

In order to have a precise measure of the splice site candidates, we counted as positives only the sites that had exactly the same location as in the real exon. We used two performance measures: sensitivity (S_n) and specificity (S_p). Sensitivity (S_n) is the number of true positives divided by total number of cases to predict (true positives + false negatives), and Specificity (S_p) is the number of true positives divided by total predicted (true positives + false positives) (Snyder and Stormo, 1995).

4. Results and discussion

4.1. *The relationship between training set size and generalization*

The optimal networks learn the patterns hidden in the data with a representative training data set. The results of the hypernetwork generalization are shown in Table 2. Testing was performed after training, with the stored hypernetwork structures. The hypernetwork trained with 100 instances did not generalize well, obtaining average testing accuracy of 0.760 ± 0.033. Training with 400 instances (set B), the hypernetwork showed a testing accuracy of 0.887 ± 0.013. However, the hypernetwork achieved acceptable generalization when trained with half of the 2000 training instances (set C), obtaining testing accuracy of 0.912 ± 0.013. Thus, we observed an improvement in learning with the size of the training sets, and a linear relationship of the computer time the algorithm took while training a number of vectors.

Experiment D shows at 50,000 generations the best hypernetwork testing accuracy was 0.927. Using the same training and validation sets, Michie et al. (1994) reported a test accuracy of 0.959 with the radial basis function (RBF) method, accuracy of 0.912 with backpropagation, 0.905 with bayesian trees, and 0.854 with *k*-NN. Thus, the hypernetwork architecture outperformed some well-known algorithms, even though the current parameter setting could be further optimized.

[2] The HyperExon program, testing files, performance results files, and additional information can be found at http://malthus.micro.med.umich.edu/hypernet/hyperexon.

Table 3
Results of the two-fold cross validation experiment

Train set	Training performance	Testing accuracy			
		Acceptors	Donors	Neither	Total
A	0.9802	0.9179	0.9117	0.9554	0.936
B	0.9836	0.9177	0.9026	0.9563	0.934
Average	0.9819				0.935

Table 4
Performance of HyperExon and GeneSplicer in finding the correct acceptor and donor sites

	Acceptor sites			Donor sites		
	HE	GS	NG2	HE	GS	NG2
True positives	51	40	54	52	44	41
False positives	111	101	284	76	57	78
Sensitivity (S_n)	0.61	0.48	0.64	0.62	0.52	0.49
Specificity (S_p)	0.31	0.28	0.16	0.41	0.43	0.34

Total no. of cases to predict = 84. HE: HyperExon; GS: GeneSplicer; NG2: NetGene2.

4.2. Two-fold cross validation results

In addition to the experiments reported in Section 4.1, we performed two-fold cross validation. Table 3 shows the two-fold cross validation results from the hypernetwork generalization performance. The average testing accuracy was 0.935, with very small difference between of each set (A and B). Each run of the experiment took about 20–22 h of computer time on a Xeon 2.6 GHz machine.

4.3. The performance of the HyperExon recognition system

Table 4 shows the results of evaluating the set of loci with the HyperExon, GeneSplicer, and NetGene2 systems. A good recognition system will have high sensitivity (S_n) and high specificity (S_p).

The sensitivity of HyperExon outperformed GeneSplicer in both acceptor and donor sites, indicating that HyperExon was able to obtain a higher number of true positives than GeneSplicer, with both having about the same specificity level.

The sensitivity of NetGene2 and HyperExon were about the same in the acceptor sites, but NetGene2 obtained half the specificity of HyperExon. In general, an increase in specificity will usually result in a decrease in sensitivity. Therefore, if NetGene2 increases the specificity of acceptors to the same level of HyperExon values, its sensitivity will be lower. On the donor sites, HyperExon outperformed NetGene2 in both sensitivity and specificity values.

4.4. Future work

Hypernetwork dynamics is based on string matching and the formation of interaction cascades. Its evolutionary learning algorithm is based on mutations on the molecular binary strings. There are no weights to maintain as in standard neural networks. These features make this novel architecture a great candidate for hardware implementation in field programmable gate arrays (FPGA).

Hypernetworks can evolve in different structural levels, in terms of size and interactions. We have preliminary unpublished results that show that if we evolve other attributes in a population of hypernetworks it could result in the improvement of learning performance. Further research will explore these capabilities.

The HyperExon recognition system could be improved by training with a large annotated database, adding structural coding information and rules for creating exon models, among other features.

5. Concluding remarks

The main goal of this paper was to show that the hypernetwork architecture, a biologically inspired information processing system, exhibits generalization performance comparable with other algorithms and is suitable for DNA splice recognition systems. Using the best hypernetwork obtained during training, including local information, we built HyperExon, a system to obtain splice site candidates. The HyperExon system outperformed leading splice recognition systems in the list of sequences tested.

Biologically inspired methods and technologies, such as the ones proposed here, demonstrate the potential of stronger connections between computer science and biology, both in the development of new algorithms and possibly in new hardware approaches.

Acknowledgments

The authors would like to thank Gary Fogel and anonymous reviewers for helpful comments on this article.

References

Brunak, S., Engelbrecht, J., Knudsen, S., 1991. Prediction of human mRNA donor and acceptor sites from the DNA sequence. J. Mol. Biol. 220, 49–65.

Fogel, G.B., Challapilla, K., Fogel, D.B., 2003. Identification of coding regions in DNA sequences using evolved neural networks. In: Fogel, G.B., Corne, D.W. (Eds.), Evolutionary Computation in Bioinformatics. Morgan Kaufmann Publishers, Amsterdam, Boston, London, New York, pp. 195–218.

Fu, L., 1999. An expert network for DNA sequence analysis. Hybrid Intell. Syst. 14 (1), 65–71.

Hebsgaard, S., Korning, P., Tolstrup, N., Engelbrecht, J., Rouze, P., Brunak, S., 1996. Splice site prediction in *Arabidopsis thaliana* DNA by combining local and global sequence information. Nucleic Acids Res. 24 (17), 3439–3452.

Henderson, J., Salzberg, S., Fasman, K., 1997. Finding genes in DNA with a hidden Markov model. J. Comput. Biol. 4 (2), 127–141.

Howe, K.L., Chothia, T., Durbin, R., 2002. Gaze: a generic framework for the integration of gene-prediction data by dynamic programming. Genome Res. 12 (9), 1418–1427.

Kulp, D., Haussler, D., Reese, M.G., Eeckman, F.H., 1996. A generalized hidden Markov model for the recognition of human genes in DNA. In: Proc. Int. Conf. Intell. Syst. Mol. Biol., vol. 4, AAAI Press, pp. 134–141.

Mathe, C., Schiex, T., Rouze, P., 2002. Current methods of gene prediction, their strengths and weaknesses. Nucleic Acids Res. 30 (19), 4103–4117.

Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J., 1994. Machine Learning, Neural and Statistical Classification. Ellis Horwood.

Noordewier, M.O., Towell, G.G., Shavlik, J.W., 1991. Training knowledge-based neural networks to recognize genes in DNA sequences. In: Lippmann, R., Moody, J., Touretzky, D. (Eds.), Advances in Neural Information Processing Systems, vol. 3. Morgan Kaufmann Publishers, pp. 530–536.

Pertea, M., Lin, X., Salzberg, S.L., 2001. GeneSplicer: a new computational method for splice site prediction. Nucleic Acids Res. 29 (5), 1185–1190.

Rampone, S., 1998. Splice-junction recognition on gene sequences (DNA) by BRAIN learning algorithm. Bioinformatics 14 (8), 676–684.

Ravetch, J.V., Lanier, L.L., 2000. Immune inhibitory receptors. Science 290 (5489), 84–89.

Segovia-Juarez, J., Colombano, S., 2006. A molecule interaction based architecture for evolutionary learning and biocomputing, submitted for publication.

Segovia-Juarez, J., Conrad, M., 2001. Learning with the molecular-based hypernetwork model. In: Kim, J.-H., Byoung-Tak, Z., Fogel, G., Kuscu, I. (Eds.), Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2. IEEE Press, COEX, Seoul, Korea, pp. 1177–1182.

Snyder, E.E., Stormo, G.D., 1995. Identification of protein coding regions in genomic DNA. J. Mol. Biol. 248 (1), 1–18.

Stephens, R.M., Schneider, T.D., 1992. Features of spliceosome evolution and function inferred from an analysis of the information at human splice sites. J. Mol. Biol. 228, 1124–1136.

Towell, G., Shavlik, J., 1992. Interpretation of artificial neural networks mapping knowledge-based neural networks into rules. In: Moody, J., Hanson, S., Lippmann, R. (Eds.), Advances in Neural Information Processing Systems, vol. 4. Morgan Kaufmann, pp. 977–984.

Watson, J.D., Hopkins, N.H., Roberts, J.W., Steitz, J., Weiner, A.W., 1987. Molecular Biology of the Gene, 4th ed. Pearson Education.

Xu, Y., Einstein, J., Mural, R., Shah, M., Uberbacher, E., 1994. An improved system for exon recognition and gene modeling in human DNA sequences. In: The Second International Conference on Intelligent Systems for Molecular Biology, vol. 2, pp. 376–384.

Xu, Y., Mural, R., Einstein, J.R., Shah, M., Uberbacher, E.C., 1996. GRAIL: a multi-agent neural network system for gene identification. Proc. IEEE 84 (10), 1544–1552.