An Expert Network for DNA Sequence Analysis

LiMin Fu, University of Florida, Gainesville

THE EXPERT NEURAL NETWORK SYSTEM THE AUTHOR HAS DEVELOPED FOR USE IN DNA SEQUENCE ANALYSIS COMBINES A TRADITIONAL SYMBOLIC EXPERT SYSTEM WITH AN ARTIFICIAL NEURAL NETWORK. THE RESULTING HYBRID SYSTEM CAN ACCURATELY MODEL UNDERLYING DOMAIN KNOWLEDGE TO IMPROVE ACCURACY, GENERALIZATION PERFORMANCE, AND INFORMATION-PROCESSING SPEED.

IN THE 45 YEARS SINCE THE DISCOVERY of DNA's helical structure, scientists have made great strides in exploring human DNA's structure and locating human genes. As part of that effort, advanced recombinant DNA and gene-mapping techniques developed over the last two decades have led to an unprecedented effort to map and sequence the entire human genome under the auspices of the Human Genome Project (see the sidebar).

The huge amount of data the HGP produces will require high-performance computing and more intelligent computer algorithms for analysis and inference, needs that the emerging field of computational molecular biology is addressing. Interest has been growing in exploring such AI tools as search algorithms, machine-learning techniques, and knowledge-based systems for DNA sequence analysis. Recently, the neural-network model has emerged as a promising AI technique in this regard because this approach might well embody important aspects of intelligence not captured by symbolic and statistical methods.[1]

An important direction in this work involves integrating multiple, fundamentally different AI approaches into single *hybrid intelligent systems*, which let each component perform the tasks for which it is best suited. Integrating symbolic knowledge into a neural network to create a *knowledge-based neural network* has quickly become an important hybrid-intelligence research area. Empirical observations indicate that such systems can outperform both neural-network and symbolic approaches. This integrated approach involves mapping a set of symbolic rules that encode available domain knowledge into the neural computing architecture[2–5] and requires innovative methods for extracting symbolic knowledge from a trained neural network.[3,5–8]

Called *expert networks* in some cases, these knowledge-based neural networks perform as well as human experts and often exhibit characteristics of a traditional symbolic expert system. In one implementation, the expert network combines a neural network's computational framework with an expert system's inference engine. We call a neural network that bases the activation function on the certainty factor model of Mycin-like systems a *CF-net*.[9] This article describes my successful attempt to apply the CF-net to DNA sequence analysis (see the "Expert network" sidebar for a description of the expert network model, its learning theory, and the neural-network pruning involved). As I'll show, the CF-net uses as the system input the DNA nucleotide sequence rather than predefined relative frequency measures as in existing exon-prediction systems.[1,10,11]

1094-7167/99/$10.00 © 1999 IEEE

The Human Genome Project

The Human Genome Project began as a joint initiative of the US's Department of Energy and National Institutes of Health that now brings scientific resources from around the world to bear on the problem of mapping and sequencing the entire human genome—the collection of all 100,000 human genes—by the year 2005. Officially begun in 1990, this project should be a great boon to human health, for example by improving genetic-level diagnosis and gene therapy.

Ultimately, this project may help usher in a new era of molecular medicine in which doctors do not so much treat symptoms as they do look into the deepest causes of diseases. If it can provide a thorough molecular-level understanding of how humans develop from embryo to adult, what makes us tick biologically, and how things can go wrong, this project may well lead to

- highly targeted pharmaceuticals that attack both heritable and communicative diseases at their molecular foundations;
- quick and more accurate diagnostic procedures that provide more timely treatment;
- deeper understanding of genetic susceptibilities to disease that when coupled with preventative therapies will thwart certain diseases; and
- the possibility of actually correcting certain genetic defects.

For more information regarding this project, see "From Maps to Medicine: About the Human Genome Research Project: at *http://www.nhgri. nih.gov/Policy_and_public_affairs/Communications/Publications/ Maps_to_medicine/about.html*; "History of the DOE Human Genome Program" at *http://www.er.doe.gov/production/ober/hugh_hist.html*; and "To Know Ourselves" at *http://www.lbl.gov/Publications/TKO*.

The DNA sequence analysis problem

A crucial problem in molecular biology is identifying new genes. Genes determine a biological being's development, appearance, and behavior, and are carried by the chromosomes in the cell nucleus. The number of chromosomes varies with the species. A chromosome's biochemical composition is characterized by a long string (sequence) of DNA nucleotides, which are huge biochemical molecules. Genes are biochemical units that are identified from the DNA sequence and carry hereditary characteristics from parent to offspring. *Genotype* refers to an individual's or organism's genetic constitution, while *phenotype* means the individual's observable characteristics or traits, as determined by the genotype and the environment. A *genome* is the collection of all an organism's genes.

Sequencing an entire genome is a systematic approach for unraveling the secret of life. However, knowing the DNA sequence does not necessarily mean understanding it. Some part of the sequence might control the expression of the genes of another part, for instance, and some genes might not be expressive at all. The same DNA sequence can produce very different proteins, depending on where the process begins.

Understanding the human genome, which consists of billions of nucleotides, poses an enormous, highly complex task and demands a heuristic approach to information processing. Scientists need to exploit existing background knowledge gathered from such fields as molecular genetics, biochemistry, and biophysics to facilitate this understanding process because this knowledge can rule out many incorrect interpretations of the DNA sequence and generate plausible hypotheses for experimental verification. Both heuristic and knowledge-based information processing lie at the heart of AI.

Gene expression involves the synthesis of RNA on the DNA template (transcription) and the synthesis of protein on the RNA template (translation). In eukaryotic cells (cells with visibly evident nuclei and organelles), a gene includes regions preceding and following the protein-coding region and consists of alternating segments of exons and introns. An *exon* is a nucleotide sequence that is expressed or translated into protein, whereas an *intron* is an intervening sequence that is transcribed (into RNA) but later eliminated from the transcript by splicing its adjacent exons. Therefore, only exons represent the mature gene. The splice junctions refer to the points where splicing takes place. Because the DNA sequence is ordered, a splice junction can be either exon-intron (EI) or intron-exon (IE) (see Figure 1).

Given a DNA sequence, the fundamental gene-identification issue is to determine the presence and location of genes in the sequence. Searching for special signal regions such as promoters (the initiation sites of transcription) or splice junctions is one approach. Measuring the features characteristic of protein coding from segment to segment is another. In either case, exon identification is an essential step toward eukaryotic gene modeling.

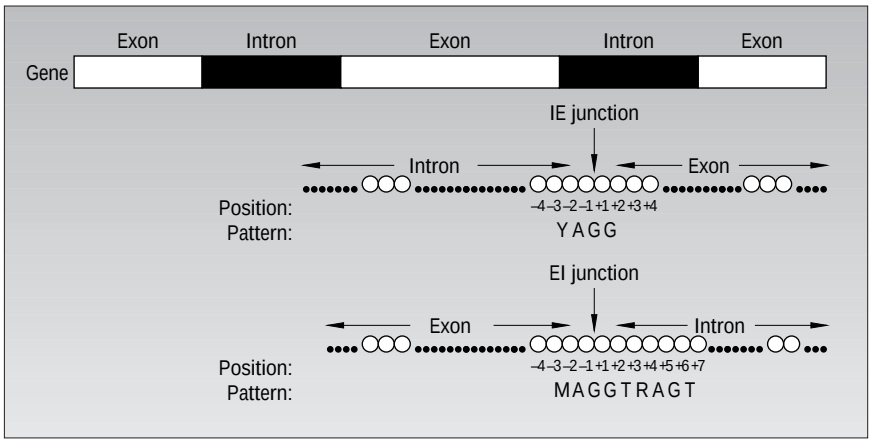

Figure 1. Splicing junctions of a gene.

Applying CF-Net to DNA sequence analysis

As we'll see, in applying hybrid-intelligent-system approaches to the problem of DNA sequence analysis, a reduced neural-network architecture improved the network-generalization performance. Furthermore, using the expert network for splice-junction recognition worked well for building a practical system for identifying exons in DNA sequences. In contrast to other knowledge-

based neural-network research,[2,4,5] I did not insert the prior domain theory into the neural network, which means that the neural network must discover the theory inductively from empirical data.

Data descriptions. The first data set used for this study concerns primate splice-junction gene sequences (see *ftp.ics.uci.edu*).[5] The data set contains 3,190 instances. Each instance consists of a sequence of 60 DNA nucleotides of four base types: A (adenine), G (guanine), T (thymine), and C (cytosine), and a label indicating one of the three possible classes: IE (an intron-exon boundary—called an acceptor), EI (an exon-intron boundary—called a donor), and N (neither IE nor EI). There are 768 instances in the IE category, 767 instances in the EI category, and the remaining 1,655 instances in the neither category.

The position in each instance sequence is specified relative to the boundary in the middle. There are 30 nucleotides before and following the boundary, corresponding to the positions −1 to −30 and +1 to +30. There is no position zero. Thus, given a position in the middle of a window of 60 DNA elements, an instance tells the presence of an IE, or EI, or neither boundary. At each position, a character denotes the nucleotide type. So, the sequence takes the form of AGCTGTTC-CGG.... In some cases, the base type is ambiguously specified, imparting some degree of missing information. For example, the character "R" refers to "A" or "G."

I sampled the second data set, which contains 50 human genes (DNA sequences), from the National Institute of Health's gene bank (*http://www.ncbi.nlm.nih.gov*). In contrast to the first data set, the second set's DNA sequences have variable lengths, with the numbers of nucleotides ranging from hundreds to a few thousand. Thus, one instance in the second set can generate numerous instances as defined in the first set.

I used the first data set to train the neural network, learn domain knowledge, and evaluate the performance of the learning system, while I used the second data set to evaluate the performance of the EXON-ENet (described later) versus the Grail system for exon recognition.

Reduced-expert-network performance. The neural network has 241 input units, including the bias unit, one hidden layer of five hidden units, and one output. Each input DNA sequence for processing consists of 60 nucleotides. In the network, the nucleotide at each position is encoded by four input units designated by A, G, T, and C. Figure 2 shows the architectural view. The original neural network is fully connected between adjacent layers, while the reduced network is sparsely connected. I used the same neural network structure in both learning tasks (learning IE and EI). I wrote the software used here in Lisp and have published a C version in *Neural Networks in Computer Intelligence*.[3]

I evaluated generalization performance using twofold cross-validation. The twofold cross-validation error is the average test error, determined by randomly dividing the instances into two sets and using one as the training set, and the other as the test set, and then vice versa. In learning the IE junction, my evaluation considered all the data set's non-IE instances as negative instances.

Table 1 shows the results of the cross-validation experiments in the domain of splice-junction analysis. This table lists two types of errors: the mean-squared error between the actual network output and the target output value, and the classification error. Because the model's network output value is confined to the range of 0 to 1, the network classifies a given instance as a positive instance if the network output value is greater than 0.5, or else it is a negative instance. The MSE error and the classification error are generally in agreement, although in some cases a smaller MSE error might be associated with a slightly larger classification error. When the network output value is interpreted as the posterior probability of the output concept for a given instance, the MSE error indicates the accuracy of such probability estimation.

The reduced neural network has a better MSE error than the original network in both learning tasks (learning IE and EI junctions). For classification error, the reduced network performs better in learning EI junctions and about the same in learning IE junctions, compared with the original network.

In a previous study, the tenfold cross-validation classification error rates on 1,000 randomly drawn instances for learning EI and IE junctions were 0.024 and 0.026, respectively, with the original expert network. This result surpasses those on the same data set reported elsewhere:[5] Kbann (0.076, 0.085), MLP with backpropagation (0.057, 0.11), Pebls (0.082, 0.076), perceptron (0.16, 0.17), ID3 (0.11, 0.14), Cobweb (0.15, 0.095), and nearest-neighbor (0.12, 0.091), where the first number in the parenthesis is the EI error rate and the second is the IE error rate. Among these techniques, Kbann relies on knowledge and artificial neural networks, the MLP and the perceptron are artificial

Table 1. Cross-validation experiment results. (CV-error is cross-validation error, MSE is mean-squared error, and CLE is classification error).

		Experiment A				Experiment B					
		Train		Test		Train		Test		CV-error	
Network	Concept	MSE	CLE	MSE	CLE	MSE	CLE	MSE	CLE	MSE	CLE
Original	IE	0.028	0.023	0.033	0.029	0.026	0.023	0.034	0.036	0.034	0.033
Original	EI	0.022	0.021	0.022	0.026	0.018	0.014	0.026	0.033	0.024	0.030
Reduced	IE	0.025	0.028	0.028	0.034	0.024	0.026	0.029	0.033	0.029	0.034
Reduced	EI	0.023	0.030	0.021	0.024	0.017	0.025	0.023	0.028	0.022	0.026

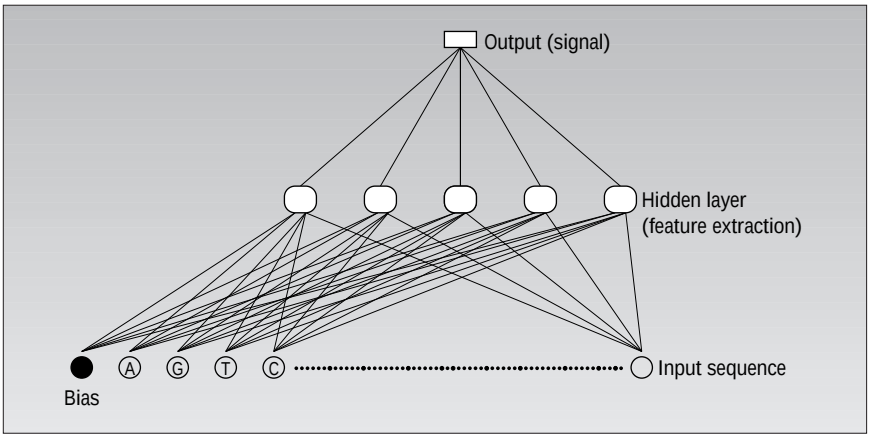

Figure 2. The expert network architecture for learning splicing junctions.

Expert network

We can formally define a neural network model using a four-tuple $<S, W, L, F>$, where W is a set of weights (ω_{ji}), L is a set of learning operations (l_{op}), F is a set of network activation functions (f_{act}), and S is further defined by a pair $<N, C>$, where N is a set of nodes (n_i) and C a set of connections. Assume n nodes in the network. We have the following definitions:

$$N = \{n_i | 1 \le n\}$$
$$C \subseteq \{(n_i \to n_j) | 1 \le i, j \le n\}$$
$$W \subseteq \{(n_i \to n_j, w_{ji}) | 1 \le i, j \le n\}$$
$$L = \{l_{op1}, l_{op2}, ...\}$$
$$F = \{f_{act1}, f_{act2}, ...\}.$$

Each learning operation transforms one neural network to another by weight adaptation or structural change.

Under this formal definition, the expert network model for our application works as follows. The network is a two-layer, feedforward, fully connected architecture with one hidden layer $L = \{l_{bp}\}$, where l_{bp} is the standard backpropagation algorithm.[1] $F = \{f_{CF}\}$ where f_{CF} is the activation function based on the certainty factor model of Mycin-like expert systems.[2,3] The weight values are confined to the range from –1 to 1 and the weights between the hidden and the output layer are further restricted to nonnegative values: $-1 \le w_{ji} \le 1$ for all $1 \le i, j \le n$, where w_{ji} is the weight associated with the connection pointing from node i to node j, but $0 \le w_{ji} \le 1$ if node j is an output unit. Unlike hidden units, all output units have no, or zero, bias: $w_{j,b} = 0$ if node j is an output unit where b designates the bias unit. The activation levels are in the range of –1 to 1: $-1 \le o_i \le 1$ for all $1 \le i \le n$ where o_i denotes the activation level at node i. I have adopted the CF-based activation function because it can improve the neural network's generalization capability of the neural network.[3]

Learning theory

The Vapnik-Chervonenkis dimension forms an important basis for measuring the capacity of a machine-learning or pattern-classification system.[4,5]

Given a set of instances S, a concept hypothesis h in H can partition the set into two groups: the instances in one group satisfy the hypothesis h and those in the other group do not. The partition is called the dichotomy induced by h. H can be a class of {0, 1}-valued functions F in which a dichotomy induced by f is a partition of S into two disjoint subsets S^+ and S^- such that $f(\mathbf{x}) = 1$ for $\mathbf{x} \in S^+$ and $f(\mathbf{x}) = 0$ for $\mathbf{x} \in S^-$. The maximum number of dichotomies induced by hypotheses in H on any set of m instances is the *growth function* of H with respect to m. The Vapnik-Chervonenkis dimension (VCdim) of H is the largest m such that the corresponding growth function is equal to 2^m. That is, H can induce all possible dichotomies of m instances drawn from the instance space if and only if the VCdim of H is m. In this way, the VCdim of H measures the capacity of H.

The network's *generalization error* is the difference between the generalization on the training data (which forms an estimate of the true generalization) and the generalization on the actual problem. Because the network tends to fit the training data, the generalization with respect to it will be overly optimistic. However, in many cases, the generalization error can be bounded (a worst-case analysis), and this bound can be made arbitrarily small by increasing the number of training instances. Vapnik and Chervonenkis show that a useful bound can be established when the number of training instances exceeds the VCdim.[4] Assume the hard-limiting activation function. The VCdim of a one-hidden-layer perceptron with full connectivity between the layers is in the range

$$2\lfloor N_h/2 \rfloor d \le VCdim \le 2N_w \log(eN_n),$$

where $\lfloor \cdot \rfloor$ is the *floor* operation that returns the largest integer less than its argument, N_h is the number of hidden units, N_w is the number of weights, N_n is the number of computing nodes in the network, e is the base of the natural logarithm, and d is the number of input units.[6,7] The upper bound holds no matter what the number of layers and the connectivity. As a rule of thumb, the number of weights can give a rough estimate of the VCdim. The VCdim of a neural network using the sigmoid activation function is actually larger than that in the hard-limiting case.

What if the neural network uses the CF-based activation function? Experience with expert systems shows that the system's conclusion is insensitive to the change of CFs up to ±0.2.[8] This observation suggests that we can quantize continuous CF values into a small number of intervals with little change in the system's conclusion. Weight quantizability gives a different perspective to compute the hypothesis space's cardinality and hence the neural network's learning capacity. Elsewhere, I provide the first analysis of this case and demonstrate empirically that the CF-based neural network can generalize better from a limited number of training instances than the traditional sigmoid-function neural network.[3]

Given neural network R with a single output, assume that the connection weight is quantizable into q levels. Each weight setting is associated with a mathematical function that the network uses to classify objects. Given N_w adjustable weights, the cardinality of the hypothesis space H of all possible distinct weight settings is q^{Nw}, which bounds from above the number of possible functions computed by the network under the quantization scheme. Let $VCdim(R) = v$. There exists a set of v distinct instances for which 2^v distinct functions are required to form all possible dichotomies. Therefore,

$$VCdim(R) = v \le \log|H| = N_w \log q.$$

In the defined expert network with a single output

$$N_w = (d + 1)N_h + N_h,$$

where the first term counts the number of weights between the input and the hidden layer and the second term the number of weights between the hidden layer and the output. Recall that d is the input dimension and N_h is the number of hidden units. Combining similar terms yields

$$N_w = (d + 2)N_h.$$

The result follows:

$$VCdim(R) \le (d + 2)N_h \log q.$$

We might then ask how many training instances are required for valid generalization. S*ample complexity* refers to the number of random examples needed for a learning system to produce a hypothesis that is correct. Using Valiant's notion,[9] we say that a class of target concepts is "learnable" if the following condition is met: For every concept in the class and with any probability distribution on the instance space, there exists a polynomial time algorithm that can produce a hypothesis such that its probability of error has a small upper bound. The probability of error is relative to the distribution of instances and is defined as the probability of instances that are either in the hypothesis and not in the target concept or in the target concept and not in the hypothesis. When the error is small, the hypothesis is a good approximation to the target concept.

Valiant's model for learning is also known as the PAC (Probably Approximately Correct) model. A hypothesis f is said to be approximately correct with accuracy $\in$ if $E(f) \le \in$ where E is the function returning the

probability of error. We say a learning program is probably approximately correct with probability δ and accuracy $\in$ if $Prob(E(f) > \in) < \delta$ where f is the hypothesis output by the program and *Prob* is the probability function.

Given the preceding analysis on the growth function and VCdim, we can determine the number of training instances required for good generalization. Eric Baum and David Haussler[6] show that if $m \geq O[(N_W/\in) \log (N_n/\in)]$ (assuming $0 < \in \leq 1/8$) random examples are used to train a perceptron with N_n computing nodes and N_w weights, the network will correctly classify a fraction of $1 - \in$ of future examples drawn from the same distribution.

Given a feedforward multilayer neural network R with N_w adjustable weights, suppose the connection weight is quantizable into q levels. I have shown[3] that the network trained on a set of m random instances is probably approximately correct with probability δ and accuracy $\in$ if

$$m \geq (\ln(1/\delta) + N_w \ln q)/\ln(1/1 - \in).$$

We can derive the sample complexity of the defined expert network from this result using the equality

$$N_w = (d + 2)N_h.$$

It follows that the expert network trained on a set of m random instances is probably approximately correct with probability δ and accuracy $\in$ if

$$m \geq (\ln(1/\delta) + (d + 2)N_h \ln q)/\ln(1/1 - \in).$$

Reduced network architecture

One approach to preventing data overfitting and improving neural network generalization is to simplify and regularize the trained neural network. For instance, we can build in an a priori bias in favor of simple models where there are not too many strong interactions between the model variables. *Weight decay* implements this idea by introducing an extra term into the error function so that weights not reinforced in the error-minimization process are decayed to a small value.

Network pruning improves network generalization. The idea is to prune connections or nodes that are relatively less important to network performance and tend to cause some degree of overfitting to the training data. Indices are available for measuring how sensitive the change in the node activation or in the weight value is to the network output. This would reflect the criticality of each node or weight in the network. The least critical nodes or weights are the candidates for pruning.[2]

The certainty-factor-based neural network takes a simple procedure for network pruning. After training, this network deletes all weights with magnitude below 0.2. We can justify this procedure by the empirical observation that certainty factors can be perturbed in a small range up to 0.2 with little effect in the quality of the network output. Another round of backpropagation then follows the pruning operation to ensure that the error criterion is minimized.

That a pruned neural network can generalize better than its parent network is not only an empirical observation but can be shown theoretically. If a pruned network learns the same number of randomly drawn training instances as its parent network, the former provides more accurate generalization than the latter from the PAC learning perspective.

Theorem 1. A feedforward multilayer neural network R has N_w adjustable weights. Suppose the connection weights are quantizable into q levels. The network that learns a set of m random instances is probably approximately correct with probability δ and accuracy $\in$ then

$$\in > 1 - (\delta / q^{Nw})^{1/m}.$$

Proof. Let H be the space of all possible weight settings for network R. For a function f implemented by a certain weight setting in H that is not correct, $E(f) > \in$ where E is the function returning the probability of error. So, the probability of f being correct on a single instance is $\leq 1 - \in$ and the probability of f being correct on all m instances is $\leq (1 - \in)^m$. Then the probability that there exists a function implementable by a weight setting in H that is consistent with the m instances but is not approximately correct is at most $|H|(1 - \in)^m$. Solving the constraint $|H|(1 - \in)^m < \delta$, we obtain

$$\in > 1 - (\delta / |H|)^{1/m}.$$

The result follows because under the quantization scheme, $|H| = q^{Nw}$.

This theorem provides an estimate of the lower bound of the generalization error of the neural network for a given probability δ and the number of training instance m under the PAC model of learning. Suppose we define

$$\in_l = 1 - (\delta / q^{Nw})^{1/m},$$

where $\in_l$ refers to the lower bound of the generalization error from the PAC learning perspective. Then we have the following result.

Corollary 2. A neural network R_1 with N_1 weights reduces to a smaller neural network R_2 with N_2 weights. $N_2 < N_1$. If both networks learn the same m random instances, network R_2 has a better generalization accuracy than network R_1 with respect to $\in_l$. That is, $\in_l(R_2) < \in_l(R_1)$.

References

1. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representation by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1, MIT Press, Cambridge, Mass., 1986.
2. L.M. Fu, *Neural Networks in Computer Intelligence*, McGraw Hill, New York, 1994.
3. L.M. Fu, "Learning in Certainty Factor Based Multilayer Neural Networks for Classification," *IEEE Trans. Neural Networks*, Vol. 9, No. 1, 1998, pp. 151–158.
4. V.N. Vapnik and A.Y. Chervonenkis, "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities," *Theory of Probability and Its Applications*, Vol. 16, No. 2, 1971, pp. 264–280.
5. A. Blumer et al., "Learnability and the Vapnik-Chervonenkis Dimension," *J. ACM*, Vol. 36, No. 4, 1989, pp. 929–965.
6. E.B. Baum and D. Haussler, "What Size Net Gives Valid Generalization," *Neural Computation*, Vol. 1, 1989, pp. 151–160.
7. D.R. Hush and B.G. Horne, "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, Jan. 1993, pp. 8–39.
8. B.G. Buchanan and E.H. Shortliffe, eds., *Rule-Based Expert Systems*, Addison-Wesley, Reading, Mass., 1984.
9. L.G. Valiant, "A Theory of the Learnable," *Comm. ACM*, Vol. 27, No. 11, 1984, pp. 1134–1142.

neural networks, ID3 is based on decision trees, Cobweb is based on probabilistic theory, and the nearest-neighbor algorithm and its weighted version (Pebls) are related to instance-based reasoning.

The reduced neural network also offers a nearly tenfold information-processing speedup because it has about one-tenth of the number of the original network's connection weights.

Exon-recognition-system performance. Once the system recognizes the splice junctions, recognizing exons is a simple matter because an exon is associated with a pair of IE and EI borders. However, as research indicates, the splice-junction recognition procedure often produces numerous false-positive results.[1] Fortunately, content-based checking can remove as many as 90% of false exons. Each hypothetical exon is subject to the evaluation of its protein-coding potential; if the potential is low, it is removed.

This study used the trained expert network to infer IE or EI boundaries at each position in the DNA sequence on the basis of the nucleotide composition in its adjacent 60 positions (30 on either side). Each recognized IE or EI junction received a score based on the network output's activation level that reflected the posterior probability of the output concept for the input sequence.

I then used the following heuristics to process recognized IE and EI boundaries in generating a given DNA sequence's exon locations. This approach recognized exons by pairing each recognized IE border with the most adjacent EI border down the sequence. If two IE borders shared the same EI border, it removed the one with a lower score as well as exons of length less than 30 bases. In this way, I built the computer program EXON-ENet for exon recognition, which I then evaluated by comparing it with Grail,[1] a heavily used government-sponsored Internet server for DNA sequence analysis and gene modeling. Unlike Grail, the EXON-ENet uses no sophisticated content-based knowledge or features for rejecting implausible exons.

For this study, the EXON-ENet predicted IE and EI junctions and used them as the basis to predict exons for the gene sequences in the second data set. The Grail server processes requests and returns results via electronic mail. The results presented here were generated by Grail-2 (version 2). Table 2 shows the exons identified by Grail and the EXON-ENet corresponding to the real exons documented in the gene bank. An exon counts as correctly identified if at least one junction (IE or EI) is exactly identified or if the overlapped length exceeds half the average length of the candidate and the real exons. I used this fuzzy criterion because the problem is typically noisy. Such studies typically measure performance level by the recognition and false-positive rates, with recognition rate being the number of real exons identified over the total number of real exons, and false-positive rate the number of falsely identified exons over the total number of identified exons.

In this study, the average recognition rate and the false-positive rate for Grail are 0.55 and 0.08; those for the EXON-ENet are 0.85 and 0.15. Thus, the EXON-ENet outperforms Grail by a large margin in terms of the recognition rate. EXON-Enet's higher false-positive rate arises because it does not analyze the protein-coding potential for exon candidates whereas Grail does. Also, the region of 30 base pairs on each side of the splice junction does not contain enough information for the EXON-ENet to learn protein-coding features.

The number of falsely identified junctions grows as the sequence under processing gets longer, which leads to a combinatorial problem in generating potential exons. This problem is not so severe in this study because the genes under test all have short sequences of

Table 2. The exon regions identified by Grail and the EXON-ENet in correspondence with the real exons

DNA sequence	Locus	Length (bp)	Exon (real)	Exon (Grail)	Exon (Enet)
1	HSCD36G8	107	[31, 77]	NF	[31, 77]
2	HSCD36G15	684	[31, 654]	NF	NF
3	HSFASX567	1814	[308, 369]	[308, 369]	[128, 369]
			[522, 584]	NF	[522, 584]
			[1768, ≥1814]	[1768, 1813]	NF
4	HSZ75172	792	[1, 270]	[12, 270]	[73, 270]
			[517, 792]	[537, 791]	[534, 733]
5	HSIGMAAA	1029	[524, 571]	NF	NF
			[675, ≥1029]	[677, 1028]	[673, 820]
6	HSMUCIN5B1	700	[≤1, 526]	NF	[437, 526]
7	HSMUCIN5B2	3616	[266, 447]	[266, 447]	[266, 447]
			[733, 904]	[733, 916]	NF
			[2021, 2280]	[2021, 2280]	[2021, 2280]
			[2623, 2809]	[2623, 2809]	[2778, 2809]
			[2975, 3200]	[2975, 3200]	[2975, 3200]
			[3315, 3490]	NF	[3315, 3490]
8	HSMUCIN5B3	1426	[101, 170]	NF	[101, 170]
			[616, 716]	[616, 773]	[456, 773]
			[1186, 1217]	NF	[1186, 1217]
9	HSSORD01	249	[≤155, 220]	[70, 220]	[52, 220]
10	HSSORD02	167	[56, 89]	NF	[56, 89]
11	HSSORD03	276	[48, 212]	[62, 212]	[48, 181]
12	HSSORD04	278	[67, 226]	[67, 239]	NF
13	HSSORD05	194	[41, 159]	[89, 193]	[41, 159]
14	HSSORD06	171	[63, 128]	NF	[63, 128]
15	HSSORD07	275	[65, 240]	[65, 248]	[124, 248]
16	HSSORD08	1377	[105, 226]	[105, 226]	[105, 226]
			[1155, ≥1377]	NF	NF
17	AB004057	752	[85, 194]	NF	[85, 194]
			[296, 458]	NF	[296, 458]
			[547, 653]	NF	NF
18	HSKAMS01	1086	[736, 814]	NF	[623, 814]
19	HSKAMS02	261	[136, 217]	NF	[136, 217]
20	HSKAMS03	434	[211, 293]	[231, 293]	[211, 293]
21	HSKAMS04	490	[166, 238]	[166, 238]	[166, 238]
22	HSKAMS05	283	[76, 200]	[76, 200]	[76, 200]
23	HSKAMS06	197	[42, 116]	NF	[42, 116]
24	HSKAMS07	413	[114, 215]	[114, 253]	[114, 215]
25	HSKAMS08	953	[178, 381]	[178, 411]	[178, 383]
			[655, 738]	[655, 744]	[655, 717]
26	HSPPP1R2E1	476	[20, 439]	NF	[240, 439]
27	HSPPP1R2E2	198	[35, 142]	NF	[35, 142]

no more than a few thousand base pairs. However, without using protein-coding features, the EXON-ENet cannot very well predict exons in long gene sequences, while Grail does better on long sequences.

IN THE FUTURE, WE CAN IMPROVE splice-junction prediction by exploiting already detected high-order statistical dependencies or correlations of the signal sequences. Explicit representation of such information in the input enables the neural network to explore deeper statistical structures on this basis rather than from scratch. Work is underway to integrate such features into the EXON-ENet.[1,10,11]

documented (bp: base pair; NF: Not found).

DNA sequence	Locus	Length (bp)	Exon (real)	Exon (Grail)	Exon (Enet)
28	HSPPP1R2E3	160	[25, 102]	NF	[?, 102]
29	HSPPP1R2E4	130	[16, 110]	NF	[?, 110]
30	HSGNBPB39	376	[36, 237]	[100, 237]	[17, 174]
31	HSGNBPB38	165	[56, 122]	NF	[26, 122]
32	HSGNBPB37	290	[77, 239]	[77, 211]	[77, 239]
33	HSGNBPB36	150	[38, 101]	NF	[38, 76]
34	HSGNBPB35	741	[575, 681]	NF	[575, 713]
35	HSGNBPB34	975	[166, 204]	NF	[166, 204]
36	HSGNBPB32	440	[69, 233]	NF	[69, 174]
37	HSGNBPB31	779	[191, 400]	NF	[254, 399]
38	HSGNBPB11	1085	[928, ≥1034]	[928, 1034]	NF
39	HSTCRHJC	650	[21, 63]	NF	NF
			[178, 488]	NF	[178, 292]
40	HSTCRHJA	517	[38, 349]	NF	[38, 415]
41	HSFGF8S1	265	[≤96, 127]	[96, 127]	[34, 127]
			[220, 256]	NF	[220, 256]
42	HSFGF8S2	256	[46, 226]	[79, 226]	[150, 226]
43	HSFGF8S3	167	[16, 122]	NF	[16, 122]
44	HSFGF8S4	402	[29, ≥319]	NF	NF
45	HSLICAM	418	[82, 197]	[90, 197]	[90, 256]
			[401, ≥418]	NF	NF
46	D83261S1	870	[669, 760]	[650, 760]	[16, 760]
47	D83261S2	1192	[56, 140]	[65, 140]	[56, 140]
			[234, 333]	[234, 333]	NF
			[837, 982]	[837, 982]	[837, 982]
48	D83261S3	2471	[338, 414]	[338, 414]	[338, 409]
			[544, 610]	[544, 610]	[544, 610]
			[747, 808]	NF	[672, 808]
			[1075, 1151]	[1033, 1151]	[1033, 1153]
			[1535, 1589]	[1535, 1589]	[1420, 1589]
			[1737, 1785]	[1737, 1785]	[1737, 1785]
			[1883, 1953]	[1883, 1960]	[1883, 1960]
			[2063, 2173]	[2073, 2173]	[2063, 2094]
			[2278, 2471]	NF	NF
49	HSIGG1CP	2802	[212, 505]	[212, 318]	[212, 505]
			[897, 941]	NF	[692, 941]
			[1060, 1389]	[1113, 1389]	[1060, 1198]
			[1487, 1809]	[1487, 1793]	NF
50	HSAPOA2G	2928	[912, 945]	NF	[905, 945]
			[1115, 1190]	[1115, 1190]	[1115, 1382]
			[1484, 1616]	[1504, 1616]	[1540, 1616]
			[2012, 2241]	[2012, 2129]	[1829, 2277]

Acknowledgments

The US National Science Foundation has supported this work under the grants IRI-9707333 and ECS-9615909.

References

1. Y. Xu et al., "Grail: A Multi-Agent Neural Network System for Gene Identification," *Proc. IEEE*, IEEE Press, Piscataway, N.J., Vol. 84, No. 10, 1996, pp. 1544–1552.
2. L.M. Fu, "Knowledge-Based Connectionism for Revising Domain Theories," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 23, No. 1, 1993, pp. 173–182.
3. L.M. Fu, *Neural Networks in Computer Intelligence*, McGraw Hill, New York, 1994.
4. G.G. Towell, J.W. Shavlik, and M.O. Noordewier, "Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks," *Proc. AAAI-90*, AAAI Press, Menlo Park, Calif., 1990, pp. 861–866.
5. G.G. Towell and J.W. Shavlik, "Knowledge-Based Artificial Neural Networks," *Artificial Intelligence*, Vol. 70, Nos. 1–2, 1994, pp. 119–165.
6. R. Andrews, J. Diederich, and A.B. Tickle, "Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," *Knowledge-Based Systems*, Vol. 8, No. 6, 1995, pp. 373–389.
7. L.M. Fu, "Rule Generation from Neural Networks," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 243, No. 8, 1994, pp. 1114–1124.
8. K. Saito and R. Nakano, "Medical Diagnostic Expert System Based on PDP Model," *Proc. IEEE Int'l Conf. Neural Networks*, IEEE Press, 1988, pp. 255–262.
9. B.G. Buchanan and E.H. Shortliffe, eds., *Rule-Based Expert Systems*, Addison-Wesley, Reading, Mass., 1984.
10. V.V. Solovyev, A.A. Salamov, and C.B. Lawrence, "Predicting Internal Exons by Oligonucleotide Composition and Discriminant Analysis of Spliceable Open Reading Frames," *Nucleic Acids Research*, Vol. 22, No. 24, 1994, pp. 5156–5163.
11. M.Q. Zhang, "Identification of Protein Coding Regions in the Human Genome by Quadratic Discriminant Analysis," *Proc. Nat'l Acad. Science*, National Academy of Science, Washington, D.C., Vol. 94, 1997, pp. 565–568.

LiMin Fu is an associate professor of Computer and Information Science and Engineering at the University of Florida at Gainesville. His research interests include rule extraction from trained neural networks and the integration of knowledge-based and neural-network principles. He created the Domrul system, which accurately discovers domain rules from a fraction of domain instances. He received his PhD and MS from Stanford, both in electrical engineering, and his BM (MD) in medicine from National Taiwan University. He wrote *Neural Networks in Computer Intelligence* (McGraw Hill, New York, 1994). He is the chairman and organizer of the First International Symposium on Integrating Knowledge and Neural Heuristics and an associate editor of *IEEE Transactions on Neural Networks*. Contact him at the Dept. of Computer and Information Science and Eng., 301 CSE, Univ. of Florida, Gainesville, FL 32611; fu@cise.ufl.edu;http://www.cis.ufl.edu/ ~fu.