Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Biology and Chemistry 27 (2003) 555–564
Computational Biology and Chemistry
www.elsevier.com/locate/compbiolchem

Artificial neural networks for prediction of mycobacterial promoter sequences

Rupali N. Kalate [a,*], Sanjeev S. Tambe [b], Bhaskar D. Kulkarni [b]

[a] *Department of Biosciences and Informatics, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223-8522, Japan*
[b] *Chemical Engineering Division, National Chemical Laboratory, Pune 411 008, India*

Received 20 May 2003; received in revised form 16 September 2003; accepted 16 September 2003

Abstract

A multilayered feed-forward ANN architecture trained using the error-back-propagation (EBP) algorithm has been developed for predicting whether a given nucleotide sequence is a mycobacterial promoter sequence. Owing to the high prediction capability (≅97%) of the developed network model, it has been further used in conjunction with the caliper randomization (CR) approach for determining the structurally/functionally important regions in the promoter sequences. The results obtained thereby indicate that: (i) upstream region of −35 box, (ii) −35 region, (iii) spacer region and, (iv) −10 box, are important for mycobacterial promoters. The CR approach also suggests that the −38 to −29 region plays a significant role in determining whether a given sequence is a mycobacterial promoter. In essence, the present study establishes ANNs as a tool for predicting mycobacterial promoter sequences and determining structurally/functionally important sub-regions therein.

Keywords: Mycobacterial promoters; Error-back-propagation algorithm; Caliper randomization approach

1. Introduction

While *Mycobacteria* have a low transcription rate and a low RNA content per unit DNA (Harshey and Ramkrishnan, 1977), their genomes are rich in G+C content. Since the G+C content of a genome affects the codon usage and the promoter recognition sites in an organism (Nakayama et al., 1989; Ohama et al., 1987), it is expected that the transcription and translation signals in *Mycobacteria* may be different from those in other bacteria such as *E. coli*. Understanding the factors responsible for the low level of transcription and the possible mechanisms of regulation of gene expression in *Mycobacteria* necessitates examination of the structure of mycobacterial promoters and their transcription machinery.

Mulder et al. have listed −35 and −10 regions of a few mycobacterial promoters (Mulder et al., 1997). Some promoters from their compilation contain −35 and −10 regions resembling *E. coli* σ^{70} type promoters. Although *Mycobacteriophage I3* (Ramesh and Gopinathan, 1995) and *M. paratuberculosis* (Bannantine et al., 1997) promoters exhibit good sequence similarity with the *E. coli* promoters at the −35 consensus, they display significant variation in the −10 region. For promoters like *M. tuberculosis 85A* (Kremer et al., 1995), sequences at the −35 position are essential for transcription although their exact location may not be critical. Some mycobacterial promoters, for instance, *M. paratuberculosis* (Bannantine et al., 1997), have a high GC content in their −10 region as compared to the AT-rich −10 region of *E. coli* σ^{70} type. Possibly, promoters having a high GC content at −10 region are the true representatives of the mycobacterial type (Parbhane, 2000). An analysis of *M. smegmatis* and *M. tuberculosis* promoters by Bashyam et al. showed that the respective −10 regions are highly similar to those of *E. coli* σ^{70} promoters; however their −35 regions exhibit greater sequence variability (Bashyam et al., 1996). The stated feature contrasting the one observed by Ramesh and Gopinathan (Ramesh and Gopinathan, 1995) is however in agreement with that noticed by Kremer et al. (Kremer et al., 1995) for mycobacterial promoters, and by Strohl (Strohl, 1992) for *Streptomyces* promoters. *Streptomyces* promoters contain diverse sequences in their −35 regions and do not function in *E. coli* (Strohl, 1992). For

* Corresponding author. Tel./fax: +81-44-211-4750.
E-mail address: rupa_kalate@bio.keio.ac.jp (R.N. Kalate).

1476-9271/$ – see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compbiolchem.2003.09.004

mycobacterial promoters, where apparent conservation in −35 region is absent, many of them possess TG dinucleotide in the immediate upstream of the −10 region, and thus they are termed "extended −10 promoters". The large variations among the mycobacterial promoters characterized thus far suggest that the consensus sequences are not representative of all mycobacterial promoters. Consequently, a number of conflicting opinions regarding the presence and characteristics of consensus promoter sequences in the *Mycobacteria* have been aired in the literature (Mulder et al., 1997).

An important objective in molecular biology is analyzing the DNA sequences for their structural and functional motifs. Macromolecular binding to specific sites of DNA involves recognition of a specific sequence pattern. In some cases, this pattern may be very distinct while in others it may be diffused. During examination of the molecular binding sites in a DNA, conventionally a consensus is derived by aligning an ensemble of sequences recognized by a common macromolecule. It is often found that the sequence pattern is never completely conserved. Efforts have also been made to develop statistical algorithms for the sequence analysis and motif prediction by searching for homologous regions or by comparing the sequence information with a consensus sequence (O'Neill and Chiafari, 1989). This approach may fail or yield insufficiently accurate results when consensus sequences are difficult to define (O'Neill and Chiafari, 1989; O'Neill, 1989). Wide variations existing within individual promoter sequences are primarily responsible for the unsatisfactory results yielded by the promoter-site-searching algorithms that in essence perform statistical analysis (Mulligan et al., 1984; Mulligan and McClure, 1986). It can thus be inferred that recognition of mycobacterial promoter sequences and the important regions therein, require a powerful technique that is capable of unraveling those hidden pattern(s) in the promoter regions, which are difficult to identify manually. An artificial intelligence (AI) based modeling/classification paradigm known as 'artificial neural networks' (ANNs) possessing significant nonlinear pattern recognition and generalization capabilities has become available in the last decade. The error-back-propagation (EBP) method (Rumelhart et al., 1986; Rumelhart and McClelland, 1986) currently represents the most popular algorithm for training feedforward networks. Neural networks using the EBP training algorithm (hereafter referred to as EBPN) have been successfully used for various applications in biology involving nonlinear input–output modeling and classification (e.g. Poggio and Girosi, 1990; Wu, 1997; Schneider and Wrede, 1998; Parbhane et al., 1998; Mahadevan and Ghosh, 1994; Tolstrup et al., 1994; Bisant and Maizel, 1995; Uberbacher et al., 1996; Zupan and Gasteger, 1993). Lukashin et al. (1989) and Mahadevan and Ghosh (1994) have developed neural network models for promoter recognition. Accordingly, our objective in this paper is to demonstrate: (i) the utility of ANNs for differentiating (classifying) mycobacterial promoter sequences from random (non-promoter) sequences, and (ii) an ANN-based caliper randomization (CR) approach (Nair et al., 1994; Nair, 1997) for determining the structurally and functionally important regions within the mycobacterial promoter sequences.

2. System and methods

The simulation programs for network training and promoter prediction were written in FORTRAN-77 and compiled using the Microsoft FORTRAN 5.0 compiler for the IBM PC and compatibles.

2.1. Data

The data for EBPN training were taken from our own compilation of the mycobacterial promoters (refer Table 1) (Kalate et al., 2002). These promoter sequences were compiled from a large number (approximately 80) of studies; the bibliographical details of the reference material are available from the authors upon request. The compiled promoter data set contains a total of 125 mycobacterial promoters out of which 80 have their transcription start site (TSS) mapped while the remaining 45 sequences are putative promoters. The promoters with the mapped TSS contain sequence stretches between −50 and +10 bp with respect to the TSS; the sequence stretch for the putative promoters lies between 15 bp upstream region of −35 box and 20 bp downstream of the −10 region. Length-wise, the compiled promoter sequences show variations owing to: (i) non-uniform availability of the nucleotide sequence upstream of the −35 region and downstream of the −10 region in the original reference, and (ii) variations in the spacer length. The shortest and the longest of the compiled sequences are 34 and 71 nucleotides long, respectively. In a few cases, two or more different sequence frames are considered for the same gene on the basis of alternate consensus probability. Thus, an overall set comprising 135 mycobacterial promoter sequences has been employed in this study.

2.2. Data representation for ANN-based classification

In ANN-based molecular sequence analyses, flexible sequence (network input) encoding schemes can be used for grasping the heterogeneous sequence features. Specifically, an individual nucleotide of a sequence can be represented using various coding strategies, such as *CODE-2*, *CODE-4* (Demeler and Zhou, 1991), *EIIP* code (Nair et al., 1994), and *wedge* and *twist* codes (Parbhane et al., 2000). In classification studies by Nair et al. (1994) and Parbhane et al. (2000), it is observed that the CODE-4 strategy (sparse encoding) fares better than the other input coding approaches. In the CODE-4 scheme, each nucleotide is represented using a set of four binary digits as given by: C = 0001; G = 0010; A = 0100; and T = 1000. On the other hand, the above-stated other coding schemes utilize smaller number of bits or

Table 1
Compilation of mycobacterial promoters

Number	Gene name	Sequence
		M. tuberculosis
1	*T3*	ATCGACGGCCACGGCTGGTCTAGGACGAGGTACCCGG(**TAACAT**)GCTGGGC[G]
2	*T6*	CCGTCCAGTCTGGCAGGCCGGAAACATCGGTCAGCAGA(**TAGGCT**)TTACCA[G]
3	*T26*	CTGCGAGCATCATATGCCGCGTGCGTGGTGATGCGGCAG(**GATGTT**)GGACC[A]
4	*T180*	GATCACTCCGAGCATGCGCCCATTGTTGTGCATAGGG(**CAGGAT**)GCCCTG[G]
5	*T101*	AGCGATCGCAGCCGACGTGATACCTGACCGTTGTTGA(**TAGTGT**)CGGCGGC[A]
6	*T119*	CCCCGTGCTCGTAGTAGGCGTCCAGCCGACCCGCCGC(**TACCAT**)GCACAAG[T]
7	*T125*	CCGAGGTAAGGACTGAGCATGGGCCCGATAAAGTGAC(**TATTAT**)GGATTTC[T]
8	*T129*	ACTCGCGGCAGATTACGCCGACGGTTCCTGGCGTGG(**TTCAAT**)ATTCGCCG[A]
9	*T130*	ACTCCAACAGGTCGATAACCTCCTGCGCCTGCTCGTC(**TATGCT**)GCGATCC[G]
10	*T150*	GACCCCCGCCACGTATTGACACTTTGCGACACGCTTT(**TATCAT**)TTTCCGA[C]
11	*recA*	TTCGGAGCAGCCGAC(**TTGTCA**)GTGGCTGTC(**TCTAGT**)GTCACGGCC[A]ACCGACCGAT
12	*rrnA P1*	GAGAACCTGGTGAGT(**CTCGGT**)GCCGAGATCGAACGGG(**TATGCT**)GTTAGGC[G]ACGGTCACCT
13	*gyrA*	GATGGGCGAGGACGT(**CGACGC**)GCGGCGCAGCTTTATCA(**CCCGCA**)ACGCCAA[G]GATGTTCGGT
14	*cpn60*	CCCCGGCGATCCCCG(**TGCTCA**)CCACGGGTGATTTCCGG(**GGCGGC**)ATGCGTT[A]GCGGACTAGC
15	*gyrB P1*	GATGTCCGACGCACG(**GCGCGG**)TTAGATGGGTAAAAACG(**AGGCCA**)GAAGATC[G]GCCCTGGCGC
16	*gyrB P3*	CAAGGGGCCTCGCCA(**TATTGC**)CGGTAGGGGTCCGCGCG(**ACACCT**)ACGGATA[A]CACGTCGATC
17	*85A*	GAAGTTGTGGTTGAC(**TACACG**)AGCACTGCCGGGCCCAG(**CGCCTG**)CAGTCTG[A]CCTAATTCAG
18	*85A*	CGCCCGAAGTTGTGG(**TTGACT**)ACACGAGCACTGCCGGGCCCAG(**CGCCTG**)CAGTCTG[A]CCTAATTCAG
19	*gyrB P2*	AGCGGTTGGCAACGA(**TGTGGT**)GCGATCGCTAAAGATCAC(**CGGGCC**)GGCACC[A]TCGTGGCGCA
20	*rrnA PCL1*	TGACCGAACCTGGTC(**TTGACT**)CCATTGCCGGATTTGTAT(**TAGACT**)GGCAGG[G]TTGCCCGAAA
21	*16S rRNA*	TGACCGAACCTGGTC(**TTGACT**)CCATTGCCGGATTTGTAT(**TAGACT**)GGCAGG[G]TTGCCCCGAA
22	*glnA*	TCGGCATGCCACCGG(**TTACGA**)TCTTGCCGACCATGGCCC(**CACAAT**)AGGGCCGGGG[A]GACCCGGCGT
23	*glnA*	CCACCGGTTACGATC(**TTGCCG**)ACCATGGCCC(**CACAAT**)AGGGCCGGGG[A]GACCCGGCGT
24	*katG* P_A	GGTCATCTACTGGGG(**TCTATG**)TCCTGATTGTTCGATATCC(**GACACT**)TCGCGATC[A]CATCCGTGAT
25	*katG* P_A	ATCTACTGGGGTCTA(**TGTCCT**)GATTGTTCGATATCC(**GACACT**)TCGCGATC[A]CATCCGTGAT
26	*katG* P_B	GAGGCGGAGGTCATC(**TACTGG**)GGTCTATGTCCTGATTGTTC(**GATATC**)CGACAC[T]TCGCGATCAC
27	*katG* P_B	ACGAGGCGGAGGTCA(**TCTACT**)GGGGTCTATGTCCTGATTGTTC(**GATATC**)CGACAC[T]TCGCGATCAC
28	*katG* P_C	CCTGATTGTTCGATA(**TCCGAC**)ACTTCGCGATCACATCCGTGAT(**CACAGC**)CCGATAA[C]ACCAACTCCT
29	*katG* P_C	TTCGATATCCGACAC(**TTCGCG**)ATCACATCCGTGAT(**CACAGC**)CCGATAA[C]ACCAACTCCT
30	*purL*	CGGCTTGTCCGTTTC(**CACGCG**)GCCGCAGCGCGATGGGGCCTAGC(**TAGACT**)GCCTCC[G]TGATGTCTCC
31	*purC*	ATCTCATACCAGAGA(**TACCAG**)CACAGGGCGCCGTCGTGCGGCGGA(**TAGGCT**)GGCGTG[A]TGCGCCCCGC
32[a]	*groE*	CAGGAAGCAAGGGGGCG(**CCCTTG**)AGTGCTAGCACTCTCATGT(**ATAGAG**)TGCTAGATGGCAATCGGCTA
33[a]	*groE*	CAGGAAGCAAGGGGG(**CGCCCTTG**)AGTGCTAGCAC(**TCTCATGTATAGAG**)TGCTAGATGGCAATCGGCTA
34[a]	*ahpC*	TGTGATATATCACCT(**TTGCCT**)GACAGCGACTTCACGG(**TACGAT**)GGAATGTCGTAACCAAATGC
35[a]	*32 kDa*	ACATGCATGGATGCG(**TTGAGA**)TGAGGATGAGGGAAGC(**AAGAAT**)GCAGCTTGTTGACAGGGTTC
36[a]	*10kDa*	AAGCAAGGGGCGCCC(**TTGAGT**)GTCAGCACTCTCATGTA(**TAGAGT**)GCTAGATGGCAATCGGCTAA
37[a]	*10kDa*	AAGCAAGGGGCGCCC(**TTGAGT**)GTCAGCACTCTCATG(**TATAGA**)GTGCTAGATGGCAATCGGCT
38[a]	*10kDa*	AAGCAAGGGGCGCCC(**TTGAGT**)GTCAGCAC(**TCTCAT**)GTATAGAGTGCTAGATGGCA
39[a]	*65 kDa*	GCGTAAGTAGCGGGG(**TTGCCG**)TCACCCGGTGACCCCCG(**TTTCAT**)CCCCGATCCGGAGGAATCAC
40[a]	*mpt64*	GAGTCTGGTCAGGCA(**TCGTCG**)TCAGCAGCGCGATGCCC(**TATGTT**)TGTCGTCGACTCAGATATCG
41[a]	*metA*	TCCGGCCCCCGCGAT(**TTGGCG**)AGCTTCGTGCGTGTTCGG(**TAGCCT**)GGCATTTACCGACGCGGGGT
42[a]	*rpsL*	GCCGCAACGCCCGCT(**TTGACC**)TGCCAGACTGGCGGCGGG(**TATTGT**)GGTTGCTCGTGCCTGGCGGC
43[a]	*38 kDa*	CGTCGCCGGACTGTCGGGGGACGTCAAGGACGCCAAGCGCG(**GAAATT**)GAAGAGCACAGAAAGGTATG
44[a]	*ppgk*	CGGGCCGCAGTTTAAGGTGAGGGTCATCCACGTCTCGCCGAGGAGATTCGATGACCAGCAC
		M. bovis BCG
45	*hsp60 P2*	CGGTGCGGGGCTTCTTGCACTCGGCATAGGCGAGTGC(**TAAGAA**)TAACGTT[G]
46	*rRNA*	TGACCGAACCTGGTC(**TTGACT**)CCATTGCCGGATTTG(**TATTAG**)ACTGGCAGGGTTGCCCCGAA
47[a]	*ahpC*	TGTGATATATCACCT(**TTGCCT**)GACAGCGACTTCACGG(**TACGAT**)GGAATGTCGCAACCAAATGC
48[a]	*23K*	GAGTCTGGTCAGGCA(**TCGTCG**)TCAGCAGCGCGATGCCC(**TATGTT**)TGTCGTCGACTCAGATATCG
49[a]	*mpb64*	GAGTCTGGTCAGGCA(**TCGTCG**)TCAGCAGCGCGATGCCC(**TATGTT**)TGTCGTCGACTCAGATATCG
50[a]	*18K*	TGGCGTCCGAAACAC(**TTGAGG**)TGCGGCCCAGCAAGGGGC(**TACAGG**)TTTTTTCCTTCACCTACGGA
51	*64K*	GCGTAAGTAGCGGGG(**TTGCCG**)TCACCCGGTGACCCCCGG(**TTTCAT**)CCCCGATCCGGAGGAATCAC
52[a]	*rpsL*	GCCGCAACGCCCGCT(**TTGACC**)TGCCAGACTGGCGGCGGG(**TATTGT**)GGTTGCTCGTGCCTGGCGGC
53[a]	*Mpb70*	TGGCGTCCGAAACAC(**TTGAGG**)TGCGGCCCAGCAAGGGGC(**TACAGG**)TTTTTTCCTTCACCTACGGA
54[a]	*alpha*	CGACTTTCGCCCGAA(**TCGACA**)TTTGGCCTCCACACACGG(**TATGTT**)CTGGCCCGAGCACACGACGA
		M. leprae
55	*16S rRNA*	TAGTCAACCCGGGAC(**TTGACT**)CCTCTGCTGGATCTGT(**ATTAAT**)CTGGCTG[G]GTTGCCGAAG
56	*18Kda*	CTTGTCTATCACAAC(**TTGCAT**)CAATATATCGACCAGTG(**CTATAT**)CAAATCTA[T]GTAGTCAGGA
57	*18 Kda*	CTTGTCTATCACAAC(**TTGCAT**)CAATATATCGACCAGTGC(**TATATC**)AAATCTA[T]GTAGTCAGGA
58[a]	*28-kDa*	TCAATATAACCACTC(**TGGTCA**)CACTAACCATACTCG(**TACCAT**)CAACCGTGTGGGGCTAATCC
59[a]	*groE1*	AGCAGCGGGCCGGCC(**TTGAGT**)GCTAGCACTCGCGTGTA(**TAGAGT**)GCTAGATGGCAGTCGGCCAG
60[a]	*65 kd*	GAATTCCGGAA(**TTGCAC**)TCGCCTTAGGGGAGTGC(**TAAAAA**)TGATCCTGGCACTCGCGATC

Table 1 (*Continued*)

Number	Gene name	Sequence
61[a]	*36k*	GTTGGG(**TTTCCT**)CTCGGAGGGCGCACCGC(**TACGTT**)AGCGGGATG
62	*SOD*	GG(**TGGGCG**)CGATCATGGCGCAGCGTT(**GATTAT**)GCTAGTCG
63	*rpsL*	CGCCGTTGGGTCGCT(**TTGACC**)TGCCCGAGCAGGGACGGG(**TATTGT**)GTTTCTCGTTCCTGACGGCT
		M. smegmatis
64	*alrA*	GTCTGCGGCCTCTGG(**GACAAT**)GGGCGCC[G]GAGATTATGA
65	*S4*	AAGCCGAATCGAGACCTTTTGGGTTCGTACACACTTGCTT(**TATAAG**)CCTC[G]
66	*S5*	AACAAGATTCCGTTAATCGTGTCTGGTGGAGCTGGTGG(**TAAGCT**)TGATCC[G]
67	*S6*	CATCGATTTTAAATTTTTGA(**TAGAG**T)GCAAATA[A]
68	*S12*	ACCTCGTTATGCTTCTGGCTATTTTTGATCAACTTT(**TATACA**)TGGGCGGT[T]
69	*S14*	TCAAGCACCCAAGCCAACATGGTTGTAGTAGTCGTTT(**TACCAT**)GTGTACC[T]
70	*S16*	TCCACGCGAACCGCTTCGGCGTGCCCCGTTTTCCCTGT(**TATAAT**)ATCGGC[G]
71	*S18*	GATCATTGTCTTCTGTTGTCTTTCGTA(**TAAAGT**)TGTTACT[G]
72	*S19*	TTTGATGTAGCCAAAGGCTCTCACCACCTGAGCCATGA(**TAGTAT**)CCATCC[C]
73	*S21*	ACATGGCATTTTTCATTTAAAACAGGACTCAGGTGG(**TATGGT**)TGACATCG[A]
74	*S30*	GATCAGCTATGTTCTTCAGTAAAATTTCGGC(**TATATG**)TTGGT[G]
75	*S33*	GATCCGCTCTTCTTATGATGCCAGTTATGGTATC(**TATGGT**)TATC[G]
76	*S35*	AACTAAAGTATGTGCCGTAATTGACAGTGTTCTAGAT(**TATGAT**)GCTGCAT[C]
77	*S65*	GGCACAGCTCGAAGTTCTACTACATGGCTTGCTGAA(TCCAGT)CACATTAC[T]
78	*S69*	ATCACGATGTCTTCATGCTTGGCTTCAATGCTCCGGTC(**TACAAT**)CAGTTC[A]
79	*S119*	GATCAAGAAGCCAATGATTTGT(**TAAACG**)CAATTAAT[G]
80	*gyrB*	CAGAATCGGTGCTGT(**CGCTAT**)CTCGCGG(**TAGACT**)GGACGAC[G]GATCTCAGGC
81	*recA*	AGAGTTCGACCGGAC(**TTGTCG**)GTGGTCTGC(**TCTAAC**)GTCACGGCC[A]ACCGATCGGA
82	*ask*	GT(**TTGCCC**)GCCGCGGCGCCC(**CACGAT**)GAACCGC[A]CGGGCTGACG
83	*acetamidase*	GGCCGGCGTTCACCC(**TTGACT**)TTTATTTTCATCTGGA(**TATATT**)TCGGGT[G]AATGGAAAGG
84	*rrnB*	CTCTGACCTGGGGAT(**TTGACT**)CCCAGTTTCCAAGGACG(**TAACTT**)ATTCCAG[G]TCAGAGCGAC
85	*rrnA P1*	GAAAACCTGGTCAGC(**CTCGGA**)GCCGAGATCGAGAGAG(**TAAGCT**)CGTAG[G]AAGCAAGACC
86	*rrnA P2*	CTCTGACCAGGCGAT(**TTGCAA**)TCGCGACGAACCTCGTAT(**TATCTT**)TATGAA[G]TCGCCGCGGA
87	*rrnA P3*	CCGGGCCAGAGCGAC(**TTGACA**)AGCCAGCCGAGATCGTAC(**TAAGCT**)GGCGAG[G]TTGCCTCAGA
88	*rrnA PCL1*	CCGGTCCAGAGCGAC(**TTGACA**)AGCCAGACAAAGCAGTAT(**TAAGCT**)GGCAGG[G]TTGCCCCAAA
89	*rpsL*	CCGCCGTGCACGAGT(**TTGTTT**)CGTCGCGGTCGCCCCTGG(**TATTGT**)GGTGGATC[G]TGCCTGGCCC
90	*rpsL*	CGTGCACGAGTTTGT(**TTCGTC**)GCGGTCGCCCCTGGTAT(**TGTGGT**)GGATC[G]TGCCTGGCCCGAAA
91[a]	*ahpC*	TGTGATATATCACCT(**TTGCCT**)GACAGCGACTTCACGG(**CACGAT**)GGAATGTCGCAACCAAATGC
		M. paratuberculosis
92	*pAJB303*	GACGACGAGGGCGG(**TGGCGT**)CGCCGGTGTAGCCGAA(**CGGCAC**)GTGCGCG[T]AGGCCCAGAT
93	*pAJB86*	CCACCTTACTCCCGA(**TGACGT**)TGCACGGCTGGGATTAA(**CGGTCC**)GCGTGC[T]CCAGGAGACA
94	*pAJB125*	GCAACGAGCGCATCA(**TTAAAG**)ATCGANGGCGCCGGGNT(**CATGTC**)CCTTCAC[C]CCGCCCAGCT
95	*pAJB300*	TCGAGTTCAAGACCC(**TGACGC**)TGGCCGACCTCGGCGCG(**CAGCCG**)ACCGCGC[A]GCGGTGCACG
96	*pJB305*	ATCCGGACGGGCAGT(**TGTTGG**)AGTTTCTGTCGGACGGT(**TGGTTG**)GCGGCAT[T]TCCGGCGAGG
97	*pAJB304*	CACCAGGTACACGCC(**AAGGAC**)AACGGCCGTATCCGGTA(**CCAACG**)GGTGTGC[G]AGCTGGACGG
98	P_{AN}	CTGGTGAAGGGTGAA(**TCGACA**)GGTACACACAGCCGCCA(**TACACT**)TCGCTTC[A]TGCCCTTACG
99	*pAJB73*	GATCGGTG(**TGCCGC**)TTGAACCGGCCCAGCTCCCG(**CTCCAG**)GGTGACG[T]GCTCGAGCTC
100[a]	*pAJB301*	GATCTGGCGGGCGG(**TCCAGT**)ACACCGCGAGTTCGCGCACG(**CTGGCC**)GGCAGCGTCTTGGACGCCCG
		M. fortuitum
101	*repA*	GAGCTCGTGTCGGACCATACACCGGTGATTAATCGTGG(**TCTACT**)ACCAAG[C]
102	*rrnA PCL1*	CCAGGATGATGCAAC(**TTGACT**)TGCCGGCAAGATTCGAAT(**TAAGCT**)GGCGGG[G]TTGCCCCAAA
103[a]	*rrnA P1*	GAAAACCTGTTGAGC(**CTCGGA**)GCCGAGATCGAAAGAG(**TAGGGT**)CGTAAACAGCAGTCCGGGCC
104[a]	*rrnA P2a*	CGCTGACCAGCCGAT(**TTGACC**)TTGTAGGCAGGCCCGCGC(**TAATCT**)TTTGAAGTCGCGCGGAGCGG
105[a]	*rrnA P2b*	CCGGGCCAGAGCGAC(**TTGACA**)AGCCAGCCGAGATCGTAC(**TAAGCT**)GGCGAGGTTGCCTCAGACCG
106[a]	*rrnA P3*	CAGGATGATGCAACT(**TGACTT**)GCCGGCAAGATTCGAATT(**AAGCTG**)GCGGGGTTGCCCCAAAACAG
		M. phlei
107	*rrnA PCL1*	ACTGGGGACGAGGTC(**TTGACG**)CCCCTGATCAGATCGGTA(**TAGACT**)GGCAGG[G]TTGCCCGAAA
108	*rrnA P1*	GAGAACCTCCGCAGT(**CTCGGC**)GCCGAGATCGAGAGGG(**TCGCCT**)GAAACATGCCGTTTACCTGC
109	*rrnA P2*	AGGGGACCCCCCTTT(**TTGACT**)CCGCTCAGACGTGGGC(**TATTCT**)TCTAACCACAAGCCCAACGC
110	*rrnA P3*	CTGGGGACGAGGTCT(**TGACGC**)CCCTGATCAGATCGGTAT(**AGACTG**)GCAGGGTTGCCCGAAAGCAA
		Mycobacteriophage I3
111[a]	*pKGR25*	CCTGTACACCCTCGC(**TGCACT**)CGCCGAGGACAAG(**CACTAT**)CGCCCCGACGTCCCGGCCTGG
112[a]	*pKGR9*	ACCACGAGCACCCGG(**TCGTCA**)GGACTGCGACACTCGA(**TGTTGT**)AGACGCACTGGTGCAGCATG
113[a]	*pKGR38*	ATCTGGTCGACCTGC(**TCGACG**)AGGTCGATCATCTTCT(**TCATCT**)CGCCGAACGGGATGCCCTGG
114	*ORF1*	ACCTCATGGAGCACT(**TCGAGG**)TCACTGAGCACGCCCA(**CGAACT**)ACGAGAGGCCGTGGGACTGG
115[a]	*ORF2*	TACTTTTTGTACCGT(**TCGACA**)CCAGCGGTTTCCGCTTCCTTGC(**CAATCT**)CCTGCAAACAAACCACAATG
116[a]	*pKGR1*	ACACAGACCAGGAGC(**TCGACA**)TGACCGCCACCGCCCCCTACAGCG(**TCATCT**)GGTTCGAAGGCACCCCGGAT
		Mycobacteriophage L5
117	*71 P2*	TACCTGTCACAAGGT(**TTGCTA**)CCGAGTGGGGCAGGCCGC(**TACATT**)TACGACC[G]CGTAACGCCA
118	*71* P_{left}	TTTGCGATTAGGGC(**TTGACA**)GCCACCCGGCCAGTAGTG(**CATTCT**)TGTGTC[A]CCGCAGCAGC

Table 1 (*Continued*)

Number	Gene name	Sequence
119	*71 P1*	ACAACTGAATATGGT(**TCCGCA**)GACGCAACTAAATTAGGGG(**TATCCT**)TGACA[G]GCACCAACAT
		M. avium
120[a]	*Avi-3*	GCCGGCGATCGTGGG(**CTGATA**)AGTCTTATCGGGCATAC(**TATAAG**)TGTAGTGGGAAATATCACCT
121[a]	*pLR7*	AGCCTTGTTGGCGGC(**CAACTG**)CCGGACGATCGCGGCGGC(**CATCGT**)CCTCGAGCTCGGCCCCGTGC
		M. neoaurum
122	*rrnA PCL1*	GCGAGACAGAGAAGC(**TTGACT**)CGCCAGACAAGATAGTT(**TAAGCT**)GGCAGG[G]TTGCCCCGAA
123[a]	*rrnA P1*	GAAAACCTGGTCAGC(**TTGGGC**)GCCGGGATCGAGCGAG(**TACACT**)CGTAAGAGACCGGTCGAGTG
124[a]	*rrnA P3*	GCGAGACAGAGAAGC(**TTGACT**)CGCCAGACAAGATAGTT(**TAAGCT**)GGCAGGGTTGCCCCGAAACG
125[a]	*rrnA P2*	CTCTGACCAGCGGAT(**TTGACT**)TCCGAAGGCACAAAGTTC(**TAATCT**)TTTGAAGTCGCCGCGGGGAG
		M. abscessus
126	*rrnA P4*	GCCAAAACCGGGAAT(**TTGACT**)CAGGTTCACGAACTTGA(**TACGGT**)TTCCGA[G]CGCCCGAAAG
127	*rrnA P1*	GGCGGGTCTAGTGGC(**GGACGG**)CGTCACAGAGGTATACGA(**TGTGTT**)TCATATCG[A]CCGCGGTTAC
128	*rrnA PCL1*	GCCCCCGACCCGAAG(**TTGACT**)CAAGTTCATTGGACTTGG(**TACAGT**)GGTCGG[G]TTGCCCTGAA
129	*rrnA P2*	GCCAAAACCGGGAAT(**TTGACT**)CAAGTTCACCGAACTTGA(**TACGGT**)TTCC[A]AGTCGCTCGG
130	*rrnA P3*	GCCAAAACCGGGAAT(**TTGACT**)CAAGTTCACCGAACTTGA(**TACGGT**)TTCCAA[G]TCGCTCGGAA
		M. chelonae
131	*rrnA P2*	CCAAAACCCGGAGTT(**TGACTC**)AAGTTCACCGAACTTGA(**TCGGTT**)CCCGG[G]CCGCTTACAA
132	*rrnA P1*	GGCGGGGTTAGTGGC(**GGATGG**)CGTCACCGAGGTATACGA(**TGTGTT**)TCATATC[G]ACCGCGGTTA
133	*rrnA PCL1*	CCCCAGAACCCGAAG(**TTGACT**)CAAGTTCATTGGACTTGG(**TACAGT**)GGTCGG[G]TTGCCCTGAA
134	*rrnA P3*	GCCAAAACCGGGAAT(**TTGACT**)CAAGTTCACCGAACTTGA(**TCGGTT**)TCCCA[G]CCGCCCGAAA
135	rrnA P4	GCCAAAACCGGGAAT(**TTGACT**)CAAGTTCACCGAACTTGA(**TACGGT**)TTCCGA[G]CCGCCCGAAA

[a] Promoter sequences which were part of test data set.

real numbers. For instance, mononucleotide representation schemes such as CODE-2 and EIIP respectively use two binary digits and a single electron ion interaction potential value for describing a nucleotide. Dinucleotide based wedge and twist codes use a single non-binary value to represent a nucleotide pair. Since CODE-4 requires maximum number (i.e., four) of input nodes to represent a single nucleotide, it produces a large-sized network as compared to other coding schemes. A large-sized network consequently increases the number of network weights (adjustable network parameters). CODE-4 scheme is inherently biased, since the vectors representing each of the four nucleotides are orthogonal (Demeler and Zhou, 1991), thus, CODE-4 encoding scheme outperforms other encoding schemes in classification application. Large parameter space increases the risk of overfitting, but the beneficial effects of the unbiased encoding apparently outweigh this in most cases. Hence, in the present classification study, the CODE-4 scheme has been preferred for mononucleotide representation.

Reese has developed a neural network model of the structural and compositional properties of a eukaryotic core promoter region of the *Drosophila melanogaster* genome. The model uses a special case of feed-forward neural network: time-delay architecture. The structure of this model allows for variable spacing between functional binding sites, which is known to play a key role in the transcription initiation process (Reese, 2001). For *E. coli* promoter sequences, Mahadevan and Ghosh employed a three-module approach with 98% classification accuracy (Mahadevan and Ghosh, 1994). In their methodology, the first neural net module predicts the consensus boxes; the second module aligns the promoters to a length of 65 bases, and the third neural net module classifies the entire sequence of 65 bases while taking care of the possible interdependencies among the bases in the promoters. It is important to note that in the present study, the perfectly aligned promoter sequences are not being used as the network input. All the promoter sequences were fed to the net in an ungapped left-adjusted fashion, as shown in Fig. 1. Consequently, the input sequence data do not require introduction of gaps for perfect alignment. The advantage of this approach is that it allows analysis of sequences where alignment is difficult or impossible to define.

2.3. Neural network simulation

The EBP network architecture used in this study is shown in Fig. 1. Training simulations for the network were performed on a 486 AT equipped with the math co-processor. The EBPN training comprises: (i) presenting the network with an input pattern (sequence) from the example set, (ii) calculating the network output by propagating the input pattern through the hidden and output layers, (iii) computation of prediction error [difference between the desired (target) output and the actual network output], and (iv) utilization of the prediction error value to update the network weights with a view of minimizing the prespecified error function, usually the *root-mean-squared-error* (RMSE). Steps (i) and (ii) of this procedure are termed "forward pass" and steps (iii) and (iv) are termed the "reverse pass" through the network architecture.

In this study, a logistic sigmoid transfer function is used at the hidden and output nodes to represent the non-linearity. During training of an EBPN, the task of RMSE minimization is accomplished by adjusting the network weights using a gradient descent technique namely the *generalized delta rule* (GDR) (Rumelhart et al., 1986). To test generalization

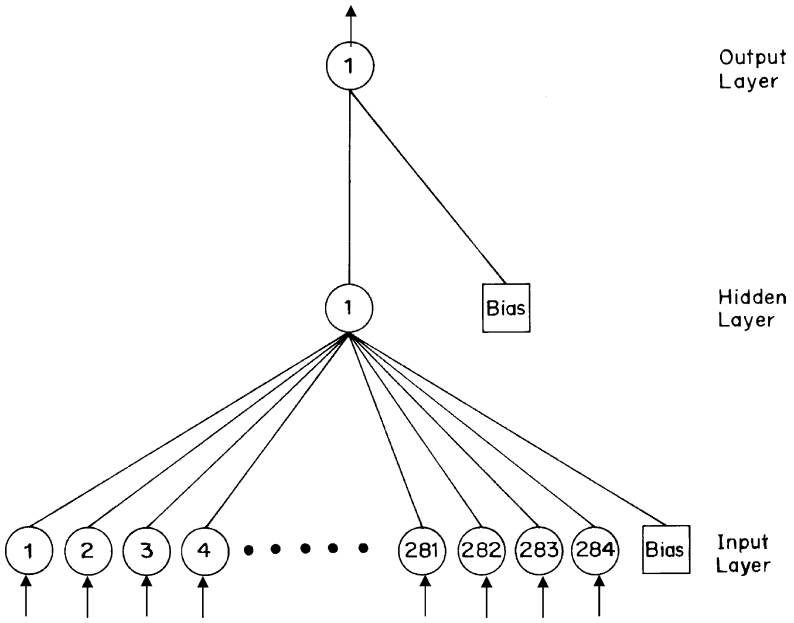

Fig. 1. Schematic of the optimized EBP neural network used in the study (containing 284 neurons in the input layer and a single neuron each in the hidden and output layers).

ability of network, its classification performance is checked at the end of each training epoch by computing the RMSE with respect to the test set; the network weights that result into smallest RMSE for the test set are taken to be optimal since such a weight set exhibits best classification performance. Since 'more-than-necessary' hidden neurons also result in overtraining, the above-described training procedure is repeated by assuming varying number of hidden nodes in the network architecture. The optimal network architecture is the one, which houses just adequate number of hidden neurons and whose weight set (termed "optimal weight set") results in the least RMSE magnitude for the test set. The detailed description of obtaining an optimal network structure and associated weight set can be found, e.g., in Freeman and Skapura (1992), and Tambe et al. (1996). For training an EBPN, the GDR algorithm for weight-updation makes use of two adjustable parameters namely, the learning rate (η) and momentum coefficient (α). In practice, values of both the GDR parameters are selected heuristically so as to obtain a network possessing good generalization ability.

Towards developing an optimal EBPN, the compiled promoter data set (135 sequences) was partitioned into training and test sets comprising 95 and 40 sequences, respectively. While partitioning the network data into training and test set, extreme care had been taken. Each promoter sequence was analyzed for various features like their total length, −35 and −10 region, spacer length, length between −10 region and TSS, nucleotide present at the TSS, %(A+T) and %(G+C) content of the promoter. Sequences representing different features were made as a part of training set. Majority of putative promoter sequences were part of test data set. All promoter sequences were aligned using ClustalW software (http://www.ebi.ac.uk/clustalw/#). The purpose of using ClustalW software was to discriminate between homologous promoter sequences. Carefully observing the results obtained from multiple sequence alignment and studying various sequence features, entire sequence compilation was distributed in training and test set. We had taken care that training and test sets do not contain copies of homologous sequences. In order that the EBPN differentiates promoter sequences from the non-promoter ones, the training and test sets must also include non-promoter sequences. Since the compiled promoter sequences were of varying length (i.e. varying in length from 34 bp long to 71 bp long), we randomly generated non-promoter sequences varying in length from 34 to 71 nucleotide long, wherein probability of occurrence of either A, T, G or C was equal to 0.25. The random sequences thus created were added to the promoter sequences in the training and test sets in 1:3 ratio. Thus the training and test sets comprised 380 and 160 sequences, respectively. For network training, the input data vectors (fragments coded in CODE-4) need to be of same size. Thus, the shorter fragments (i.e. < 71 bp) were uniformly padded with 0.01 till each fragment was 284 ($= 71 \times 4$) elements long. The resulting training and test sets can be viewed as matrices of size (380×284) and (160×284), respectively.

The EBPN architecture (Fig. 1) used for classifying the promoter sequences consists of 284 nodes in the input layer, and a single node in the output layer for representing whether the input sequence is a mycobacterial promoter. Accordingly, the target output for a promoter sequence was chosen to be unity and for a non-promoter, the target output was zero. For a given input sequence if the network output lies between 0.5 and 1.0 then the sequence is assumed to be a promoter, otherwise (i.e. network output < 0.5) it is a non-promoter.

3. Results and discussions

The training and test sets each comprising promoter and random sequences were utilized for obtaining an optimal network architecture—and the optimized weight set thereof—by following the network optimization procedure described earlier. The optimal network so developed, contains a single neuron in its hidden layer; increasing the number of hidden neurons beyond one did not increase the classification accuracy of the trained network. The RMSE profiles corresponding to the training and test sets for the optimized network are shown in Fig. 2. It was observed that the weights at the 318th training epoch ($\eta = 0.6$, $\alpha = 0.4$) correspond to the minimum RMSE (highest classification accuracy) with respect to the test set; thus these weights were taken as optimal. The optimal EBPN could correctly classify all the 380 sequences in the training set (100% classification accuracy). That is, the network could indeed differentiate between 95 promoter sequences and 285 random sequences. Moreover, the network correctly classified 155 sequences in the test set comprising 160 sequences (96.9% classification accuracy). It was also witnessed that the network did not predict any false positive i.e. none of the random sequences in the training/test sets were classified as mycobacterial promoter sequences.

To check whether the trained network has correctly identified mycobacterial promoters from random sequences just because they have high G+C content compared to random nucleotide sequences, we performed a control study for the networks utilizing equal composition of A, T, G, and C in their random sequences. The average G+C content of mycobacterial promoters from our compilation listed in Table 1 is 56%. Herein, we generated new 500 random sequences whose G+C content was kept as 56%. These new input sequences are then used to predict whether the given sequence is mycobacterial promoter or not. For this purpose, optimal weights obtained originally were utilized. It is observed that the trained network correctly identifies 498 random sequences out of 500 random sequences (classification accuracy = 99.6%). We also performed another control study wherein we tested the network's prediction capability on the non-promoter sequences of various lengths from *M. tuberculosis* genome. It is seen that trained network correctly identifies non-promoter sequences with 96% classification accuracy. This means trained network correctly classifies mycobacterial promoter sequences based on not only its G+C content, but also other sequence features like −35 box, −10 region, and spacer region, etc.

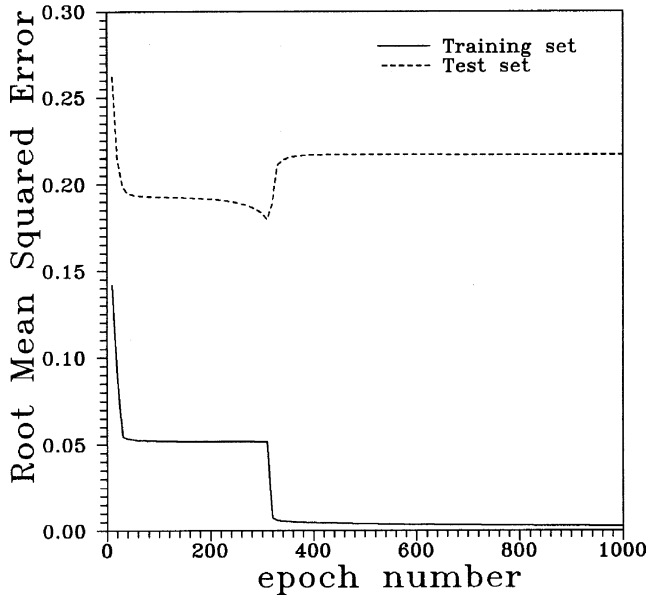

Fig. 2. RMSE profiles corresponding to the training and test data sets.

The mycobacterial promoter sequence compilation along with its bibliographic details, optimal parameters (weights) of the ANN model and the computational procedure for predicting the promoter sequences, are available upon request from the authors.

3.1. Analysis using caliper randomization strategy

The above-described classification results in essence indicate that the optimized EBPN model possesses good capability of differentiating between a mycobacterial promoter sequence and a random sequence. In other words, the network model has satisfactorily captured the hidden features that impart mycobacterial promoter characteristic to a given nucleotide sequence. It can be inferred further that the network model could now be utilized to identify important sub-regions in a promoter sequence. Towards this goal, we employ the CR approach wherein each single mycobacterial promoter sequence is randomized in parts and applied to the trained network to examine whether the sequence still retains its promoter characteristic. In here, we present results pertaining only to the mycobacterial promoters whose TSS is mapped experimentally. The other type of compiled promoter sequences, namely "putative" promoters are called so since they comprise possible consensus boxes. However, the fact that their TSS is not mapped experimentally may lead to erroneous conclusions about mycobacterial transcription machinery. For this reason, the putative promoters are excluded from analysis via CR approach. If the network classifies the partly randomized sequence to be a non-promoter, then it can be concluded that the randomized region of the original promoter sequence governs its promoter functioning. For testing this hypothesis, the trained network was presented with mycobacterial promoter sequences randomized at fixed caliper lengths. Specifically, a fixed-sized caliper window of 10 nucleotides (approximately one turn of the helix) is chosen for randomization, which is moved from one end of the sequence to the other, in an overlapping fashion (refer Fig. 3). Thus from a single promoter sequence of 71 nucleotides, 62 sequences each containing a different randomized sub-region (window) could be formed. For sequences having their length

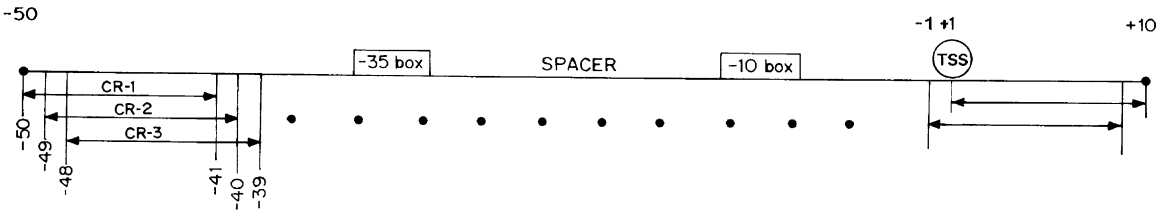

Fig. 3. CR scheme; CR-i refers to *i*th caliper window.

shorter than 71 bp long, the region undergoing randomization process is only the true sequence content and not the padded portion of the sequence. Upon randomizing all the promoter sequences (one at a time) in the training and test sets in this manner, the resulting partially randomized promoter sequences were applied to the optimized EBPN for predicting whether they maintain their promoter characteristic. Each partially randomized mycobacterial promoter sequence was examined separately with respect to its TSS. The location of caliper window which is affecting promoter features of partially randomized mycobacterial promoter sequence is calculated with respect to TSS of that particular promoter.

The classification results in respect of the partially randomized mycobacterial promoter sequences—whose TSS is known—are portrayed in Fig. 4. In the figure, it is observed that depending upon the starting location of the randomized window, the resulting sequences are classified as non-promoters to varying extent. It can thus be opined that the starting location of the randomized window plays an important role while classifying a randomized promoter sequence. More importantly, it is noticed that when the starting position for the randomized caliper window lies in the −42 to −35 region, then the resulting sequences are predominantly classified as non-promoters. This observation suggests that the nucleotide content and its arrangement in the calipers located in the −42 to −35 region are critical for mycobacterial promoters. When the caliper windows covering the spacer region and the −10 box are randomized, the original mycobacterial promoter sequences loose their promoter features. However, in this case the percentage of randomized sequences classified as non-promoters is not as high as that when caliper windows located in the −42 to −35 region are randomized. Thus, it is possible to infer that: (i) the −35 box and its upstream region play a critical role in mycobacterial promoter functioning, (ii) −10 box and spacer region also contribute towards mycobacterial promoter characteristics, and (iii) for promoter recognition the −10 region is not as important as the −35 region.

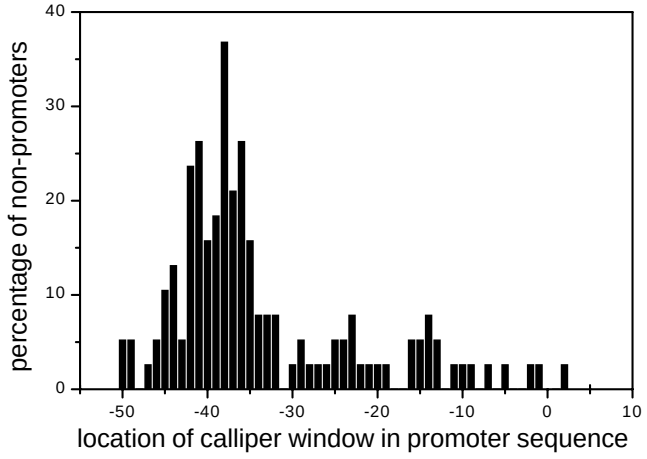

Fig. 4. Classification results in respect of partially randomized mycobacterial promoter sequences. The X-axis refers to the location of 10 nucleotide-sized caliper window and Y-axis refers to the percentage of randomized promoters classified as non-promoters.

In Fig. 4, it is clearly noticed that the caliper window starting at location −38, when randomized, results in the highest percentage (i.e. 37%) for non-promoters. This observation suggests that the −38 to −29 region is most influential in determining whether a given compiled sequence is a mycobacterial promoter or not. For an in-depth scrutiny of the −38 to −29 region, it was divided into two sub-regions viz., −38 to −34 and −33 to −29, following which each of the two sub-regions was separately randomized. Upon randomizing all the promoter sequences in this manner, they were subjected to classification using the optimal EBPN. The results of such an analysis show that 57% of the sequences require randomization of the entire −38 to −29 region to alter their classification from promoters to non-promoters. It was also noticed that randomization of the −38 to −34 region and −33 to −29 region changes 36% and 7% of the original promoter sequences, respectively, to non-promoters.

Since the −38 to −29 region of the mycobacterial promoter sequences seems more influential in imparting them the promoter characteristics, it is of interest to study the nature of consensus nucleotide pattern for this sequence stretch. Towards this objective, all the mycobacterial promoters from the compilation in Table 1, were aligned with respect to their TSS and examined carefully to identify the consensus pattern in the −38 to −29 region. Thus, the consensus nucleotide pattern observed in the −38 to −29 region is: A_{31} C_{30} T_{43} T_{49} G_{44} G_{27} C_{34} C_{37} T_{37} C_{40}. Here it is seen that while the −38 to −34 region comprises a single ‘A’ and two ‘T’s, the −33 to −29 region is GC-rich. This observation suggests that the comparatively higher AT

content in the −38 to −34 region assumes special significance for mycobacterial promoters. The −38 to −29 region is also analyzed for purine/pyrimidine consensus pattern. Thus, purine (R) and pyrimidine (Y) consensus for −38 to −29 region is: R_{57} Y_{54} Y_{65} Y_{65} R_{52} R_{51} Y_{52} Y_{62} Y_{56} Y_{58}.

Using the results of the CR analysis, it is possible to get an insight into the sub-regions of the promoter sequences, which upon randomization were classified as non-promoters. Accordingly, a detailed examination of the randomized promoters was undertaken. It revealed that the mycobacterial promoters that were subjected to randomization (and were subsequently classified as non-promoters) in: (i) upstream region of −35 box, (ii) −35 region, (iii) spacer region, and (iv) −10 region, show resemblance to *E. coli* σ^{70} type promoters. More specifically, it is noticed that 32 mycobacterial promoters are sensitive to randomization within −38 to −29 region. Among these, 20 (64%) promoters exhibit resemblance to typical *E. coli* σ^{70} type; the remaining 12 (36%) belong to a typical mycobacterial type (GC rich −10 region) for their consensus sequence pattern.

4. Conclusion

To conclude, the results presented in this study suggest that ANNs can be gainfully employed for mycobacterial promoter sequence prediction. In view of the good performance of the optimized ANN in capturing the local and global features in the promoter sequences, it is possible to use them as feature detectors for locating the functionally important regions. The results of the CR strategy indicate that the network is indeed capable of acquiring the knowledge of regions that are structurally and functionally important. Additionally, the CR analysis results show that the method can be exploited in deriving consensus for other functionally important regions wherein weak consensus sequence pattern is observed.

References

Bannantine, J.P., Barletta, R.G., Thoen, C.O., Andrews, R.E., Jr., 1997. Identification of *Mycobacterium paratuberculosis* gene expression signals. Microbiology 143, 921–928.

Bashyam, M.D., Kaushal, D., Das Gupta, S.K., Tyagi, A.K., 1996. A study of mycobacterial transcriptional apparatus: identification of novel features in promoter elements. J. Bacteriol. 178, 4847–4853.

Bisant, D., Maizel, J., 1995. Identification of ribosome binding sites in *Escherichia coli* using neural network models. Nucleic Acids Res. 23, 1632–1639.

Demeler, B., Zhou, G., 1991. Neural network optimization for *E. coli* promoter prediction. Nucleic Acids Res. 19, 1593–1599.

Freeman, J.A., Skapura, D.M., 1992. Neural Networks Algorithms, Applications, and Programming Techniques Addison-Wesley, Reading, MA.

Harshey, R.M., Ramkrishnan, T., 1977. Rate of ribonucleic acid chain growth in *Mycobacterium tuberculosis* H37Rv. J. Bacteriol. 129, 616–622.

Kalate, R.N., Kulkarni, B.D., Nagaraja, V., 2002. Analysis of DNA curvature distribution in mycobacterial promoters using theoretical models. Biophys. Chem. 99, 77–97.

Kremer, L., Baulard, A., Estaquier, J., Content, J., Capron, A., Locht, C., 1995. Analysis of the *Mycobacterium tuberculosis 85A* antigen promoter region. J. Bacteriol. 177, 642–653.

Lukashin, A.V., Anshelevich, V.V., Amirikyan, B.R., Gragerov, A.I., Frank-Kamenetskii, M.D., 1989. Neural network models for promoter recognition. J. Biomol. Struct. Dyn. 6, 1123–1133.

Mahadevan, I., Ghosh, I., 1994. Analysis of *E. coli* promoter structures using neural networks. Nucleic Acids Res. 22, 2158–2165.

Mulder, M.A., Zappe, H., Steyn, L.M., 1997. Mycobacterial promoters. Tuber. Lung Dis. 78, 211–223.

Mulligan, M.E., Hawley, D.K., Entriken, R., McClure, W.R., 1984. *Escherichia coli* promoter sequences predict in vitro RNA polymerase selectivity. Nucleic Acids Res. 12, 789–800.

Mulligan, M., McClure, W.R., 1986. Analysis of the occurrence of promoter-sites in DNA. Nucleic Acids Res. 14, 109–126.

Nakayama, M., Fujita, N., Ohama, T., Osawa, S., Ishihama, A., 1989. *Micrococcus luteus*, a bacterium with a high genomic G+C content, contains *Escherichia coli*-type promoters. Mol. Gen. Genet. 218, 384–389.

Nair, T.M., 1997. Caliper randomization: an artificial neural network based analysis of *E. coli* ribosome binding sites. J. Biomol. Struct. Dyn. 15, 611–617.

Nair, T.M., Tambe, S.S., Kulkarni, B.D., 1994. Application of artificial neural networks for prokaryotic transcription terminator prediction. FEBS Lett. 346, 273–277.

Ohama, T., Yamao, F., Muto, A., Osawa, S., 1987. Organization and codon usage of the streptomycin operon in *Micrococcus luteus*, a bacterium with a high genomic G+C content. J. Bacteriol. 169, 4770–4777.

O'Neill, M.C., 1989. *Escherichia coli* promoters. I. Consensus as it relates to spacing class, specificity, repeat substructure, and three dimensional organization. J. Biol. Chem. 264, 5522–5530.

O'Neill, M.C., Chiafari, F., 1989. *Escherichia coli* promoters. II. A spacing class-dependent promoter search protocol. J. Biol. Chem. 264, 5531–5534.

Parbhane, R.V. (2000) Analysis of DNA Sequences: Modeling Sequence Dependent Features and their Biological Roles (Dissertation). University of Pune, Pune (India).

Parbhane, R.V., Tambe, S.S., Kulkarni, B.D., 1998. Analysis of DNA curvature using artificial neural networks. Bioinformatics 14, 131–138.

Parbhane, R.V., Tambe, S.S., Kulkarni, B.D., 2000. ANN modeling of DNA sequences: new strategies using DNA shape code. Comput. Chem. 24, 699–711.

Poggio, T., Girosi, F., 1990. Regularization algorithms for learning that are equivalent to multilayered networks. Science 247, 978–990.

Ramesh, G., Gopinathan, K.P., 1995. Cloning and characterization of *Mycobacteriophage I3* promoters. Indian J. Biochem. Biophys. 32, 361–367.

Reese, M.G., 2001. Application of a time-delay neural network to the annotation of the *Drosophila melanogaster* genome. Comput. Chem. 1, 51–56.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536.

Rumelhart, D.E., McClelland, J.L., 1986. Parallel and Distributed Processing: Explorations in the Microstructure of Cognition MIT Press-Cambridge, MA.

Schneider, G., Wrede, P., 1998. Artificial neural networks for computer-based molecular design. Prog. Biophys. Mol. Biol. 70, 175–222.

Strohl, W.R., 1992. Compilation and analysis of DNA sequences associated with apparent streptomycete promoters. Nucleic Acids Res. 20, 961–974.

Tambe, S.S., Kulkarni, B.D., Deshpande, P.B., 1996. Elements of Artificial Neural Networks With Selected Applications in Chemical Engineering and Chemical and Biological Sciences, Simulation and Advanced Controls, Louisville.

Tolstrup, N., Engelbrecht, T.J., Brunak, S., 1994. Neural network model of the genetic code is strongly correlated to the GES scale of amino acid transfer free energies. J. Mol. Biol. 243, 816–820.

Uberbacher, E.C., Xu, Y., Mural, R.J., 1996. Discovering and understanding genes in human DNA sequence using GRAIL. Methods Enzymol. 266, 259–281.

Wu, C.H., 1997. Artificial neural networks for molecular sequence analysis. Comput. Chem. 21, 237–256.

Zupan, J., Gasteger, J., 1993. Neural networks in chemistry. Angew. Chem. Int. Ed. Engl. 32, 503–527.