UNIVERSIDADE DE CAXIAS DO SUL

ÁREA DE CONHECIMENTOS DE CIÊNCIAS DA VIDA

INSTITUTO DE BIOTECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM BIOTECNOLOGIA

NÍVEL DE DOUTORADO

Predição de Regiões Promotoras em *Bacillus subtilis* através do uso de Redes Neurais Artificiais e Máquinas de Vetor de Suporte

Rafael Vieira Coelho

Caxias do Sul

2018

Rafael Vieira Coelho

Predição de Regiões Promotoras em *Bacillus subtilis* através do uso de Redes Neurais Artificiais e Máquinas de Vetor de Suporte

Tese apresentada ao programa de Pós-Graduação em
Biotecnologia da Universidade de Caxias do Sul,
visando a obtenção do grau de Doutor em Biotecnologia.

Orientadora: Profa. Dra. Ana Paula Longaray Delamare
Co-orientadora: Profa. Dra. Scheila de Avila e Silva

Caxias do Sul

2018

Dados Internacionais de Catalogação na Publicação (CIP)
Universidade de Caxias do Sul
UCS - BICE - Processamento Técnico

C672p Coelho, Rafael Vieira, 1984-

Predição de regiões promotoras em *Bacillus subtilis* através do uso de redes neurais artificiais e máquinas de vetor de suporte / Rafael Vieira Coelho. – 2018.

85 f. : il. ; 30 cm

Apresenta bibliografia.

Tese (Doutorado) - Universidade de Caxias do Sul, Programa de Pós-Graduação em Biotecnologia, 2018.

Orientação: Profa. Dra. Ana Paula Longaray Delamare.

Coorientação: Profa. Dra. Scheila de Avila e Silva.

1. Bacillus subtilis. 2. Biotecnologia. 3. Redes neurais (Computação). I. Delamare, Ana Paula Longaray, orient. II. Silva, Scheila de Avila e, coorient. III. Título.

CDU 2. ed.: 579.852.1

Índice para o catálogo sistemático:

1. Bacillus subtilis	579.852.1
2. Biotecnologia	60
3. Redes neurais (Computação)	004.032.26

Catalogação na fonte elaborada pela bibliotecária
Michele Fernanda Silveira da Silveira – CRB 10/2334

RAFAEL VIEIRA COELHO

PREDIÇÃO DE REGIÕES PROMOTORAS EM *Bacillus Subtilis* ATRAVÉS DO USO DE REDES NEURAIS ARTIFICIAIS E MÁQUINAS DE VETOR DE SUPORTE

Tese apresentada ao Programa de Pós-graduação em Biotecnologia da Universidade de Caxias do Sul, visando à obtenção do título de DoutoR em Biotecnologia.

Orientadora: Profa. Dra. Ana Paula Longaray Delamare

Co-orientadora: Profa. Dra. Scheila de Avila e Silva

TESE APROVADA EM 13 DE ABRIL DE 2018.

_____

Orientadora: Profa. Dra. Ana Paula Longaray Delamare

_____

Co-orientadora: Profa. Dra. Scheila de Avila e Silva

_____

Prof. Dr. Ney Lemke

_____

Prof. Dr. Marcio Dorn

_____

Prof. Dr. Daniel Luis Notari

O importante é não parar de questionar;

a curiosidade tem sua própria razão de existir.

Albert Einstein

AGRADECIMENTOS

À minha esposa Vanessa Bavaresco, pelo apoio constante em mais esta etapa de minha vida.

À minha orientadora, Profa. Dra. Ana Paula Longaray Delamare, e à minha co-orientadora, Profa. Dra. Scheila de Avila e Silva pelo apoio e contribuições realizadas ao longo da realização da tese.

Aos professor Dr. Sérgio Echeverrigaray pelos ensinamentos e contribuições.

LISTA DE ABREVIATURAS

AM	Aprendizado de Máquina
CMR	*Comprehensive Microbial Resource*
CSVM	*Central Support Vector Machine*
DBTBS	*Database of Transcriptional Regulation in Bacillus subtilis*
DREAM	*Dialogue on Reverse Engineering Assessment and Methods*
ECF	*Extracytoplasmic Function*
FN	Falso-negativos
FP	Falso-positivos
GRPF	*Gaussian Radial Basis Polynomials Function*
HPAM	*Hybrid Promoter Analysis Methodology*
LSSVM	*Least Square Support Vector Machine*
MLP	*Multilayer Perceptron*
MOM	Modelo Oculto de Markov
MOSS	*Multi-objective Scatter Search*
MPP	Matriz de Posições Ponderadas
NANS	*Nureka Artificial Neural Systems*
NCBI	*National Center for Biotechnology Information*
NPA	*Nearest Point Algorithm*
RBF	*Radial Basis Function*
RMS	*Root Mean Square*
RN	Rede Neural Artificial
RNAm	Ácido Ribonucleico Mensageiro
RNAp	Enzima RNA Polimerase
RS	Reconhecimento de Sinais
SAK	*Sequence Alignment Kernel*
SC	Sequência Consenso
SMO	*Sequential Minimal Optimization*
SOR	*Successive Over Relation*
SRM	*Structural Risk Minimization*
SVM	*Support Vector Machine*
TLS	Sítio de Início da Tradução
TSS	Sítio de Início de Transcrição
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos

LISTA DE SÍMBOLOS

W_{ji}	peso numérico de propagação da ativação entra a unidade j e a unidade i
n	número de entradas da rede neural artificial
a_{ij}	sinal de entrada da sinapse i conectada ao neurônio j
g	função de ativação
t	índice de passo de treinamento da rede neural artificial
$E()$	função de custo
x_i	valor de posição i do vetor de entrada
y_i	saída desejada para o padrão de entrada x na posição i
p	número de padrões de entrada
$\varphi_i(.)$	mapeamento do vetor de entrada x_j em um espaço de dimensão elevada
w	componentes do vetor de peso
b	termo de polarização do vetor
φ	função de mapeamento
x^T	transposta do vetor x
C	tamanho do lado do hipercubo

Sumário

Índice de Tabelas

Índice de Figuras

RESUMO

O processo de transcrição diz respeito à leitura da informação contida no DNA para geração do RNA mensageiro correspondente. Para iniciar o processo de transcrição de um determinado gene, a enzima RNA polimerase necessita reconhecer a região promotora, atuando assim na regulação da expressão dos genes. A literatura propõe diversos métodos computacionais para a predição de sequências promotoras, mas a maioria dos trabalhos concentra-se em bactérias Gram-negativas. O objetivo deste trabalho é predizer promotores em regiões intergênicas da bactéria *Bacillus subtilis* (Gram-positiva) através da aplicação de técnicas de aprendizado de máquina: Redes Neurais Artificias (RN) e Máquinas de Vetor de Suporte (SVM). O treinamento das RN foi realizado através do algoritmo *Multilayer Perceptron* (MLP) que se baseia na regra de aprendizagem por correção de erro (*backpropagation*). Já para SVM, destaca-se os *kernels* (faz o mapeamento no espaço de características para a identificação dos vetores de suporte ideais) *Radial Basis Function* (RBF) que utiliza uma função gaussiana; SIGMOID que utiliza uma função de tangente hiperbólica; e *Nu-Support Vector Classification* (Nu-SVC) que limita o custo de penalização entre 0 e 1. O primeiro passo do trabalho foi a coleta do genoma e dos promotores reconhecidos pelos fatores *sigma* da bactéria *Bacillus subtilis* a partir dos dados contidos em bancos de dados públicos. O processamento dos dados biológicos obtidos da bactéria *Bacillus subtilis* gerou 767 regiões promotoras, sendo a maioria encontrada a partir do fator Sigma SigA. Estes dados foram processados e utilizados como entrada na aplicação das técnicas de aprendizado de máquina RN e SVM. Desta forma, foi possível comparar o desempenho das duas soluções para o problema em questão. Em ambas as soluções foram usados os mesmos dados de entrada e validação cruzada (*k-cross validation*) de 5-*fold*. Os resultados são condizentes e competitivos com os encontrados na literatura, obtendo 93.20% e 95.63% de acurácia em sua predição com o SVM (combinando os *kernels* SIGMOID e RBF com o algoritmo Nu-SVC) e obtendo 98.57% e 97.69% de acurácia em sua predição com RN (MLP com 5 e 7 neurônios na camada oculta e 1 neurônio na camada de saída). A partir dos resultados obtidos, é possível afirmar que a predição, reconhecimento e caracterização de regiões promotoras de *Bacillus subtilis* pode ser realizado com sucesso tanto usando RN quanto SVM, embora RN tenha obtido melhor desempenho.

ABSTRACT

The transcription process concerns reading the information contained in DNA to generate the corresponding messenger RNA. To initiate the transcription process of a given gene, RNA polymerase enzyme needs to recognize the promoter region, thereby regulating gene expression. Literature proposes several computational methods to predict promoter sequences, but most of them is focused on Gram-negative bacteria. Therefore, the objective of this work is to predict promoters in intergenic regions of the *Bacillus subtilis* bacterium (Gram-positive) through the application of machine learning techniques: Artificial Neural Networks (RN) and Support Vector Machines (SVM). The training of the RN was performed through the Multilayer Perceptron (MLP) algorithm that is based on the error correction learning rule (backpropagation). For SVM, the kernels (maps the characteristics space to identify ideal support vectors) that stands out are Radial Basis Function (RBF) that uses a Gaussian function; SIGMOID that uses a hyperbolic tangent function; and Nu-Support Vector Classification (Nu-SVC) that limits the penalty cost between 0 and 1. The first step was to obtain the genome and the promoters recognized by the Sigma factors of *Bacillus subtilis* from data in public data bases. Biological data gathered from Bacillus subtilis generated 767 promoter regions, being the majority found by Sigma SigA factor. These data were processed and used as input in RN and SVM machine learning techniques. Hence, it was possible to compare the efficiency of the two solutions. In both solutions, the same input data and 5-fold cross-validation were used. We obtained 93.20% and 95.63% accuracy in the SVM application (combining the SIGMOID and RBF kernels with the Nu-SVC algorithm). With RN (MLP with 5 and 7 neurons in the hidden layer and 1 neuron in the output layer), the best results were 98.57% and 97.69% accuracy. Both results are consistent and competitive when compared to those in literature. In addition, both solutions proved the reliability of the obtained data. Finally, it is possible to state that the prediction of *Bacillus subtilis* promoter regions can be successfully performed both using RN and SVM, although RN has obtained better performance.

1 INTRODUÇÃO

A bactéria *Bacillus subtilis* pertence ao gênero *Bacillus,* ordem *Eubacteria*, família *Bacilacea* (ZEIGLER et al., 2008). Trata-se de um bacilo encontrado normalmente no solo. Ele é utilizado com frequência em processos biotecnológicos industriais. Devido à capacidade de formação de esporos, é altamente resistente em condições adversas do meio ambiente (temperaturas extremas, radiação UV, etc.). Devido a sua fácil manipulação genética, trata-se do modelo de estudo de bactérias gram-positivas. Sendo assim, *B. subtilis* foi o objeto de estudo do presente trabalho.

Uma das áreas biológicas que pode avançar ainda mais através do auxílio da informática é o estudo da regulação de genes bacterianos, mais especificamente *B. subtilis* (REYS; MACEDO; DAMALIO, 2011). Para que um gene bacteriano possa ser transcrito, proteínas específicas chamadas fatores de transcrição devem localizar uma sequência no DNA chamada região promotora.

A regulação da expressão gênica em procariotos ocorre essencialmente no nível de transcrição. Embora proteínas de ligação ao DNA possam afetar a eficiência da transcrição, ela depende primordialmente das interações entre a enzima de transcrição RNA polimerase (RNAp) e as regiões promotoras (locais específicos de DNA). A holoenzima RNAp não apresenta afinidade específica pela sequência de DNA a não ser que ele seja associado à subunidade sigma (fatores de transcrição). Esta associação com o núcleo RNAp é transitória, deixando uma enzima do núcleo processante para alongar a transcrição iniciada (REYS *et al*., 2011).

Sendo assim, para iniciar a transcrição de uma região codificante, a enzima RNAp deve reconhecer a região promotora, a qual é constituída por uma sequência de nucleotídeos específicos que sinalizam e direcionam a transcrição do gene adjacente. A identificação destas regiões é relevante uma vez que os promotores

estão diretamente associados à expressão gênica e podem interferir na inibição ou ativação dos genes pós-genômica (RUFF; RECORD; ARTSIMOVITCH, 2015).

No entanto, existem alguns obstáculos que dificultam o desenvolvimento desta área de pesquisa, tornando-a desafiadora. As sequências promotoras são curtas e não completamente conservadas, tornando assim difícil sua localização e podendo gerar um alto número de falso-positivos. A predição precisa de regiões promotoras é uma questão chave para determinar padrões de expressão de genes e para entender redes de regulação genética (KREBS et al., 2017).

Foram encontrados na literatura diversos métodos computacionais que tem como objetivo a predição de regiões promotoras em procariotos: Análise Probabilística, Reconhecimento de Sinais (RS) e Aprendizado de Máquina (AM). No entanto, em procariotos, em sua grande maioria apenas as bactérias Gram-negativas são utilizadas como objeto de estudo tendo em vista a maior quantidade de dados disponíveis das mesmas. O presente trabalho visa melhorar o desempenho de soluções que tem como objetivo predizer e reconhecer promotores em regiões intergênicas da bactéria Gram-positiva *B. subtilis* através da aplicação de técnicas de aprendizado de máquina.

A metodologia do RS opera no reconhecimento de sinais relativamente conservados através de alinhamento e homologia entre promotores previamente identificados. As técnicas baseadas em reconhecimento de sinais apresentam limitações similares: (1) a variação dos nucleotídeos é grande; (2) assumem a independência entre bases adjacentes; (3) não permitem a presença de múltiplos elementos dos promotores, inserções, deleções ou espaço variável entre os elementos; e (4) o resultado pode variar de acordo com o método de alinhamento (XING et al., 2005).

Já a metodologia AM utiliza um conjunto de informações estruturais e funcionais disponíveis sobre as sequências promotoras para melhorar o reconhecimento automático e produzir hipóteses relevantes sobre as mesmas (BALDI; BRUNAK, 2001). Os trabalhos encontrados de AM podem ser divididos em: Modelos Ocultos de Markov (MOM), Redes Neurais Artificiais (RN) e Máquinas de Vetor de Suporte (SVM).

MOM tem como restrição o tamanho do conjunto de treinamento, visto a grande quantidade de parâmetros que precisam ser estimados. Além disso, há uma

dificuldade na incorporação de outras características dos promotores, em seu algoritmo, como a composição dos dinucleotídeos ou trinucleotídeos da sequência e informações sobre a estabilidade (REIS, 2005).

Segundo Mount (2013), RN são sistemas de aprendizado de máquina inspirados no funcionamento de redes neurais biológicas. Elas aprendem a partir dos exemplos e apresentam alguma capacidade de generalização do conjunto de treinamento. A vantagem desta solução é o fato delas puderem aprender a reconhecer padrões degenerados, imprecisos e incompletos; características comuns em regiões promotoras. Além disso, apresentam ótimo desempenho em grandes sequências genômicas. Como desvantagem, destaca-se a necessidade de sincronismo nos dados de entrada.

Por fim, o algoritmo das SVM também pode ser utilizado para classificações de padrões. Neste algoritmo, procura-se construir um hiperplano como superfície de decisão de tal forma que a margem de separação entre exemplos positivos e negativos seja máxima. As SVMs podem fornecer um bom desempenho de generalização em problemas de classificação de padrões, apesar de não incorporarem conhecimento do domínio do problema (LORENA; LEON; CARVALHO, 2007).

Sendo assim, as técnicas utilizadas no presente trabalho foram Redes Neurais Artificiais e Máquinas de Vetor de Suporte (implementação de *software* utilizando a biblioteca LibSVM) por serem as mais comumente aplicadas para este tipo de problema, apresentarem bom desempenho em sequências genômicas e reconhecerem padrões degenerados e não-conservados como sequências promotoras.

O restante do trabalho é organizado como segue. No Capítulo 2, são apresentados os objetivos do presente trabalho. Na sequência, no Capítulo 3 faz-se uma introdução aos conceitos básicos de biologia molecular (3.1) e das técnicas de aprendizado de máquina (3.2). Na Seção 3.3, é feita uma revisão dos principais trabalhos relacionados. Na sequência, no Capítulo 4, é descrita a forma como foi desenvolvido o presente trabalho e quais os experimentos realizados. Por fim, os resultados obtidos compõem o Capítulo 5 e o as conclusões obtidas a partir da tese compõem o Capítulo 6.

2 OBJETIVOS

O objetivo geral deste trabalho é predizer promotores em regiões intergênicas de *B. subtilis* através da aplicação de duas técnicas de aprendizado de máquina: redes neurais artificiais e máquinas de vetor de suporte.

2.1 OBJETIVOS ESPECÍFICOS

- Coletar dados gênicos de *B. subtilis* em bancos de dados públicos;
- Testar diferentes arquiteturas e encontrar a melhor arquitetura de redes neurais artificiais para o reconhecimento de promotores;
- Implementar uma solução que utilize máquinas de vetor de suporte para a predição de regiões promotoras da bactéria Gram-positiva *B. subtilis* e para a diminuição de falso-positivos;
- Definir o conjunto de parâmetros que devem ser utilizados nas máquinas de vetor de suporte para obter o melhor desempenho;
- Comparar os resultados obtidos com RN e SVM na predição de promotores de *B. subtilis* com ferramentas existentes na literatura através das métricas de desempenho acurácia, sensibilidade e especificidade.

3 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica está dividida em três partes: (3.1) fundamentos biológicos da transcrição gênica, (3.2) abordagens *in silico* de predição de promotores bacterianos e (3.3) trabalhos que aplicam RN e SVM na predição de promotores bacterianos.

3.1 FUNDAMENTOS BIOLÓGICOS DA TRANSCRIÇÃO GÊNICA EM PROCARIOTOS

As bactérias (procariotos) são organismos unicelulares que apresentam seu DNA livre na célula. Mesmo não possuindo membrana celular, a regulação em nível de transcrição é complexa. Estes organismos podem ser separados em dois grandes grupos com base na estrutura das suas paredes celulares e suas relações filogenéticas. Esta separação foi definida por Hans Christian Gram em 1884 através da diferença observada na constituição da parede celular (técnica de coloração diferencial de Gram). Desta forma, pôde-se distinguir dois grupos filogenéticos bacterianos com importantes diferenças morfológicas, genéticas e metabólicas, incluindo as sequências promotoras: (A) bactérias Gram-negativas e (B) bactérias Gram-positivas (RUFF *et al*., 2015).

Segundo Letunic e Bork (2016), existe uma distância filogenética entre *B. subtilis* e *E. coli* (exemplos modelo de Gram-positiva e Gram-negativa, respectivamente) tendo como base a análise do alinhamento de sequências e o cálculo da distância evolutiva. Wang e Sun (2009) também realizaram um estudo filogenético comparativo entre *B. subtilis* e *E. coli*. As sequências 16S de rDNA foram obtidas do banco de dados GenBank e foram realizados alinhamentos através do *software* ClustalX 1.83. Na sequência, foram realizados cálculos de similaridade entre pares de sequências 16S de rDNA através do *web-server* EzTaxon (http://www.eztaxon.org), gerando uma árvore contendo nove grupos. O grupo 1 (28 espécies) que continha *B. subtilis* está distante de *E. coli* baseado em sua filogenia, apresentando assim constituições gênicas diferentes (Figura 3.1).

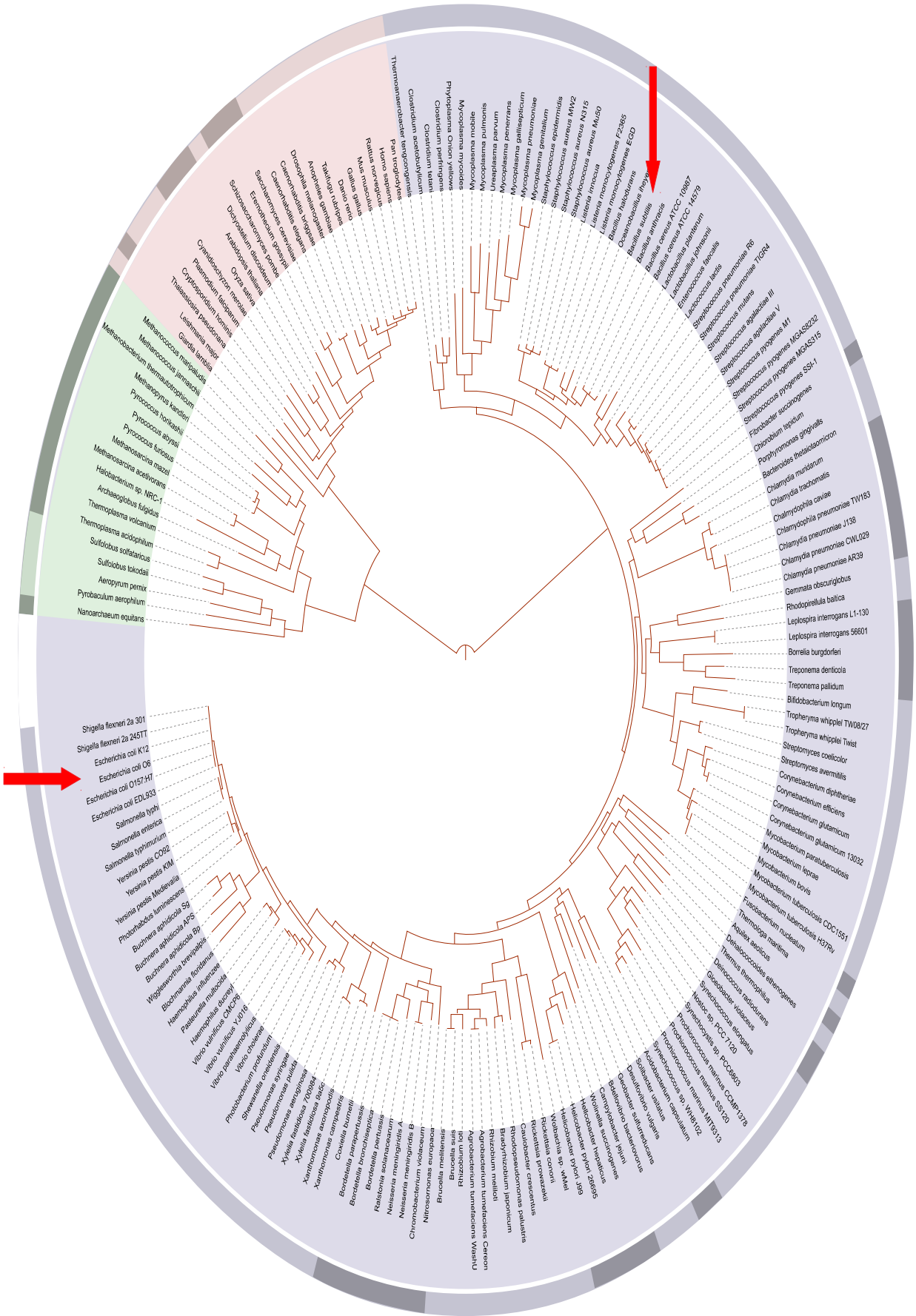


Figura 3.1: Árvore filogenética bacteriana gerada no iTOL (LETUNIC e BORK, 2016) em abril de 2018 .

Mais especificamente, este trabalho visa o estudo do DNA de *B. subtilis.* O DNA, material genético de todos os organismos celulares, é um polímero formado por resíduos denominados nucleotídeos. Os nucleotídeos são constituídos por fosfato, açúcar (desoxirribose) e uma base nitrogenada. Quatro bases nitrogenadas são encontradas no DNA: adenina (A) e guanina (G) – bases purinas; citosina (C) e timina (T) – bases pirimidinas. Na molécula de DNA, os resíduos nucleotídicos encontram-se ligados entre si através de ligações covalentes fosfodiéster formando uma cadeia estável. Por sua vez duas cadeias complementares são formadas, como pode ser observado na Figura 3.2 (KREBS et al., 2017). Cada nucleotídeo é complementado através de uma ligação da outra fita, seguindo as regras de pareamento A ↔ T (ligação fraca, duas pontes de hidrogênio) e C ↔ G (ligação forte, três pontes de hidrogênio). Mesmo que a ligação A ↔ T seja fraca, o alto número de ocorrências no DNA garante a união das cadeias. Além disso, as cadeias de DNA possuem polaridade, tendo um fósforo 5' (P) e um grupo hidroxila 3' (OH) nas extremidades. A sintetização de uma nova fita ocorre na direção 5'→ 3'.

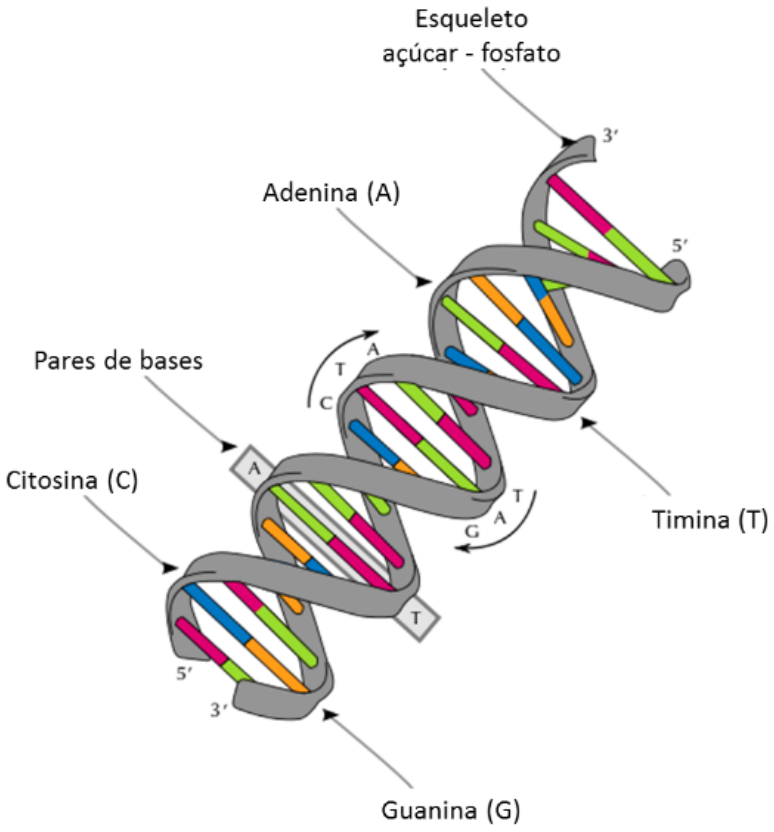


Figura 3.2: Estrutura do DNA (ácido desoxirribonucleico).

* Adaptado de Krebs et al. (2017).

Em um genoma, um gene é expresso através da cópia de sua informação em forma de RNA (ácido ribonucleico), informação esta que por sua vez é traduzida gerando uma proteína específica. Este processo é chamado de Dogma Central da Biologia Molecular e pode ser visto em detalhes na Figura 3.3 (KREBS *et al*., 2017).

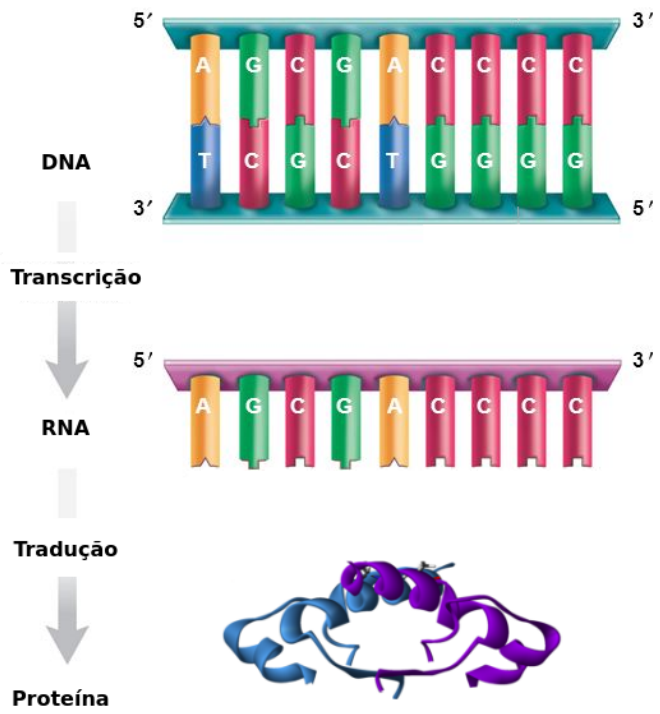


Figura 3.3: Dogma Central da Biologia Molecular.

* Adaptado de Mader e Windelspecht (2015).

A unidade de transcrição em procariotos, tanto *E. coli* quanto *B. subtilis*, representa todos os elementos que determinam a expressão de um gene. Ela é determinada pela presença da própria sequência da região codificante e por mais duas regiões importantes que sinalizam o início e o fim da unidade de transcrição. São elas: a região promotora e a região terminadora da transcrição (vide Figura 3.4).

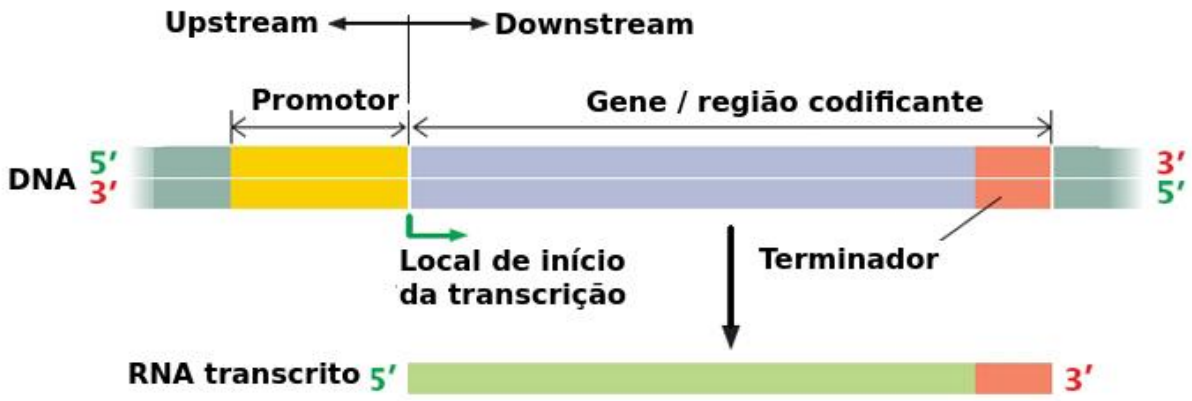


Figura 3.4: Organização da unidade de transcrição de um gene bacteriano.

* Adaptado de Reys et al. (2011)

Um promotor bacteriano típico de *E. coli* localiza-se aproximadamente *70pb* antes do ponto de início da transcrição gênica. Através da análise comparativa de vários promotores bacterianos, foram definidas duas sequências consenso: uma localizada a *-10pb (5'-TATAAT-3')* do ponto de início da transcrição e uma localizada a *-35pb (5'-TTGAC-3')* (NELSON e COX, 2011). Casey et al. (2014) realizaram um estudo no qual tentaram identificar esses elementos ao isolar os promotores de lacUV5 mutantes e observaram maior atividade em *B. subtilis*. As mutações que conferem o fenótipo desejado foram agrupadas entre os pares de base 14 e 18 e perto do local de iniciação da transcrição. A maioria das mutações tiveram pouco efeito sobre a atividade do promotor em *E. coli*, mas aumentaram a atividade de lacUV5 em até 28 vezes em *B. subtilis*. Eles concluíram que, embora as sequências nas regiões 35 e 10 altamente conservadas do promotor sejam importantes, sequências adicionais na região promotora contribuem para a expressão gênica em *B. subtilis*.

A regulação da expressão gênica em procariotos ocorre essencialmente no nível de transcrição. Embora proteínas de ligação ao DNA possam afetar a eficiência da transcrição, ela depende primordialmente das interações entre a enzima de transcrição RNA polimerase (RNAp) e as regiões promotoras (locais específicos de DNA). RNAp bacterianas podem ser isoladas de duas formas: (1) núcleo de RNAp, catalisa a polimerização de ribonucleotídeos para o complemento de RNA de uma sequência de DNA e (2) holoenzima RNAp, contém as sub-unidades da molécula do

núcleo mais um fator proteico (Sigma) que permite que a holoenzima reconheça elementos promotores e inicie a transcrição nesses locais.

Os fatores sigma (σ) são reguladores do processo de transcrição e compõem uma classe de proteínas que é composta pelas sub-unidades α1, α2, β, β' e ω. A holoenzima RNAp não apresenta afinidade específica pela sequência de DNA a não ser que ele seja associado à subunidade σ, capaz de reconhecer promotores gênicos específicos. No entanto, esta associação com o núcleo RNAp é transitória, deixando uma enzima do núcleo processante para alongar a transcrição iniciada. Sendo assim, para iniciar a transcrição de uma região codificante, a enzima RNAp deve reconhecer a região promotora, a qual é constituída por uma sequência de nucleotídeos específicos que sinalizam e direcionam a transcrição do gene adjacente. À medida que a RNA polimerase avança (vide Figura 3.5), uma molécula de RNA transcrita é gerada (REYS *et al*., 2011).

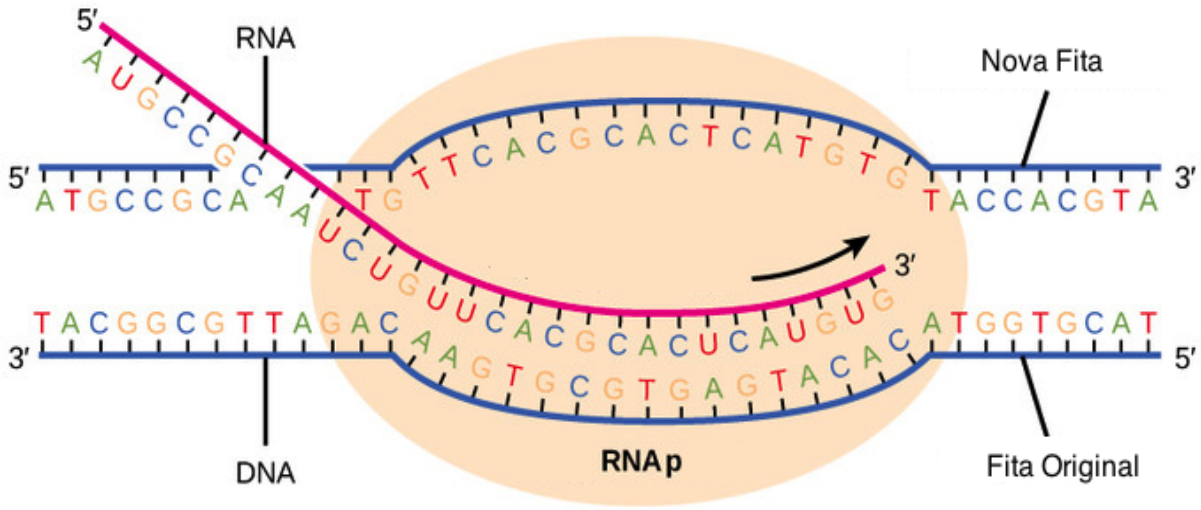


Figura 3.5: Ligação da RNA Polimerase ao DNA.

* Adaptado de Reys et al. (2011).

Os primeiros fatores Sigma alternativos de *B. subtilis* foram descobertos com base em suas qualidades bioquímicas. Seus genes estruturais foram revelados em estudos posteriores. A partir de análise de homologia em sequências de proteínas foram estabelecidos genes reguladores clonados como seqüências de codificação para fatores Sigma hipotéticos (REYS *et al*., 2011).

O processo de esporulação desta bactéria é coordenado pelos Sigmas sigE, sigF, sigG, sigH e sigK. Já a formação de flagelos depende do Sigma sigD. Os fatores Sigma descobertos com base na atividade bioquímica (por exemplo, sigA) foram designados inicialmente com base na sua massa aparente quando submetidos a eletroforese em géis de poliacrilamida desnaturantes.

Além disso, influências históricas complicaram a nomeação de genes do fator Sigma. A sequência do promotor alvo e a função de σ^A, por exemplo, são semelhantes às do factor Sigma principal de *E. coli* (σ^{70}). Sendo assim, o gene estrutural de σ^A teve a designação genética utilizada por *E. coli,* rpoD. Diversos fatores Sigma foram identificados e designados pelos locais de esporulação como por exemplo σ^H que é codificado por spoOH. O fator σ^A é o principal Sigma presente no crescimento vegetativo de *B. subtilis*. E-σ^A foi a primeira holoenzima isolada de *B. subtilis*, sendo purificada por técnicas cromatográficas de isolamento de enzimas. A adição de σ^A purificada ao núcleo de RNAp estimula a transcrição *in vitro* a partir de DNA de *B. subtilis*.

Embora possa parecer que grande parte da bioquímica estudada em *E. coli* seja diretamente aplicável à *B. subtilis*, as RNAp dessas bactérias não são idênticas. Em *E. coli,* o maior polipeptídeo do núcleo (β') é uma subunidade que pode se ligar ao DNA e a segunda maior sub-unidade (β) é o alvo de inibidores de iniciação rifampicina e estreptolidigina. Já em *B. subtilis*, ocorrem mutações de resistência à rifampicina na maior subunidade. As mutações que conferem resistência à estreptolidigina (antibiótico que atua na RNAp) não se mapeiam para esta sub-unidade, mas são encontradas no gene da outra subunidade, β. Além disso, existe uma sub-unidade RNAp única em *B. subtilis,* δ. Experimentos *in vitro* demonstraram que a adição da proteína delta a reações de transcrição reduz a iniciação inespecífica por RNAp contendo Sigmas codificados por bacteriófago (KREBS *et al*., 2017).

Promotores de *E. coli e B. subtilis* possuem similaridades e diferenças. Promotores transcritos pelos fatores $E\sigma^A$ ou $E\sigma^{70}$ possuem as seguintes similaridades: (a) sequências conservadas nos hexâmeros *35* e *10,* (b) distância entre os dois hexâmeros e (c) posição do sítio de início de transcrição. No entanto, genes estruturais ligados a alguns promotores de *E. coli* (por exemplo, o gene *lacUV5*) não conseguem ser transcritos pela RNA polimerase de *B. subtilis*. Além

disso, estudos bioquímicos demonstram que *B. subtilis* é menos tolerante ao desvio em relação ao consenso de *12pb*, diferentemente do que *E. coli* (YAMADA et al., 1991; O'NEILL, 1991). Por fim, existem seqüências cis-regulatórias localizadas imediatamente ao montante da região −35 (elementos UP, *upstream elements*) que também podem afetar o reconhecimento e a atividade dos promotores bacterianos. Este tipo de sequência apresenta 20pb de comprimento e o consenso é $^{-59}$NNAAAWWTWTTTTNNNAAANNN^{-38}, onde W pode ser substituído por A ou T e N pode representar qualquer base. Os elementos UP podem ser divididos em duas sub-regiões distintas: distal (centralizada na posição -52pb) e proximal (centralizada na posição −42pb). Estas são reconhecidas pelo domínio C-terminal de uma das duas sub-unidades α da holoenzima RNAp, estimulando assim a transcrição (HOOK-BARNARD e HINTON, 2007).

Os promotores possuem características estruturais próprias (conformação), diferentes das regiões não-promotoras, que também podem ser incorporadas nos estudos destes elementos, tais como a curvatura e a estabilidade dos promotores. A curvatura do DNA está envolvida em processos biológicos como a transcrição do DNA. Em bactérias, a curvatura é mais acentuada na região antecedente ao promotor. Para realizar o cálculo de curvatura do DNA, analisam-se os ângulos de enovelamento, torção e inclinação dos nucleotídeos. As principais metodologias existentes para a obtenção destes valores são: (1) eletroforese em gel de agarose, (2) análise do ângulo dos dinucleotídeos AA e (3) cristalografia de raio-x (KOZOBAY-AVRAHAM *et al*., 2008). Já a estabilidade é uma propriedade que depende primordialmente da soma de interações entre os dinucleotídeos da sequência. A estabilidade geral para um oligonucleotídeo pode ser descoberta a partir da contribuição relativa de cada interação vizinha mais próxima no DNA (KANHERE *et al*., 2005).

Percebe-se que o reconhecimento de um promotor bacteriano é um ponto de regulação da expressão gênica já que este reconhecimento determina o início do gene, o momento e a quantidade que será expressa pela célula (RUFF *et al*., 2015).

3.2 TIPOS DE ABORDAGENS COMPUTACIONAIS PARA PREDIÇÃO DE PROMOTORES

Encontram-se na literatura diversas abordagens computacionais para reconhecer regiões promotoras bacterianas. As principais técnicas podem ser classificadas como baseadas em Reconhecimento de Sinais (RS) ou em Aprendizado de Máquina (AM). A primeira metodologia opera no reconhecimento de sinais relativamente conservados através de alinhamento e homologia entre promotores previamente identificados. Já a segunda utiliza um conjunto de informações estruturais e funcionais disponíveis sobre as sequências promotoras para melhorar o reconhecimento automático e produzir hipóteses relevantes sobre as mesmas.

Os trabalhos encontrados na literatura de reconhecimento de sinais são os seguintes: Sequência Consenso (SC) (LISSER e MARGALIT, 1993) e Matriz de Posições Ponderadas (MPP) (HERTZ e STORMO, 1999; XING *et al*., 2005). O método Sequência Consenso consiste em alinhar um conjunto de sequências identificadas previamente como promotora e, posteriormente, pesquisar por regiões conservadas em seu conteúdo. Cada coluna no alinhamento fornece a variação encontrada nesta posição do promotor. Já no caso das Matrizes de Posições Ponderadas, assume-se que cada linha da matriz corresponde a um dos quatro nucleotídeos e cada coluna a um alinhamento. Os elementos da matriz são os pesos utilizados para pontuar uma sequência teste, conforme uma medida que quantifica a aderência ao modelo. Calcula-se então a pontuação pela soma dos pesos de cada letra (nucleotídeo) alinhada em cada posição.

Segundo Song *et al*. (2007), todas as técnicas baseadas em reconhecimento de sinais apresentam limitações similares: (1) a variação dos nucleotídeos é grande; (2) assumem a independência entre bases adjacentes; (3) não permitem a presença de múltiplos elementos dos promotores, inserções, deleções ou espaço variável entre os elementos; e (4) o resultado pode variar de acordo com o método de alinhamento.

Enquanto isto, os trabalhos encontrados de aprendizado de máquina podem ser divididos em: Modelos Ocultos de Markov (MOM) (HAYKIN, 1998), Redes Neurais Artificiais (DEMELER e ZHOU, 1991; O'NEILL, 1991; MAHADEVAN e

GHOSH, 1994; PEDERSEN e ENGELBRECHT, 1995; OPPON, 2000; COTIK *et al*., 2005; BURDEN *et al*., 2005; SANTOS *et al*., 2008; DE AVILA E SILVA et al., 2011; MENG *et al*., 2013; SOLOVYEV e UMAROV, 2016; KUMAR e BANSAL, 2017) e Máquinas de Vetor de Suporte (GORDON *et al*., 2003; KIRYU *et al*., 2005; GORDON *et al*., 2006; POLAT e GUNES, 2007; ZANATY, 2012; DE JONG *et al*., 2012; MEYSMAN *et al*., 2014; SIWO *et al*., 2016). Estes trabalhos serão apresentados com maior detalhamento na seção 3.3.

Um Modelo Oculto de Markov é um modelo estatístico no qual o sistema modelado é assumido como um processo de Markov com parâmetros desconhecidos, e o desafio é determinar os parâmetros ocultos a partir dos parâmetros observáveis. Os parâmetros extraídos do modelo podem então ser usados para realizar novas análises, realimentando assim o sistema. A vantagem desta metodologia é a capacidade de capturar regularidades em sequências de caracteres, considerando a variação nos símbolos observados em cada estado. No entanto, segundo Reis (2005), uma das restrições deste tipo de solução é o tamanho do conjunto de treinamento, visto a grande quantidade de parâmetros que precisam ser estimados. Além disso, há uma dificuldade na incorporação de outras características dos promotores, em seu algoritmo, como a composição dos dinucleotídeos ou trinucleotídeos da sequência e informações sobre a estabilidade.

As Redes Neurais Artificiais são sistemas de aprendizado de máquina inspirados no funcionamento de redes neurais biológicas. Segundo Wu e Mclarty (2000), elas aprendem a partir dos exemplos e apresentam alguma capacidade de generalização do conjunto de treinamento. A vantagem desta solução é o fato delas puderem aprender a reconhecer padrões degenerados, imprecisos e incompletos. Estas são características dos promotores, o que é essencial para o presente trabalho. Além disso, segundo Cotik *et al*. (2005), apresentam ótimo desempenho em grandes sequências genômicas. Como desvantagem, destaca-se a necessidade de sincronismo nos dados de entrada (alinhamento de sequências).

O algoritmo das Máquinas de Vetor de Suporte também pode ser utilizado para classificações de padrões. Neste algoritmo, procura-se construir um hiperplano como superfície de decisão de tal forma que a margem de separação entre exemplos positivos e negativos seja máxima. As SVMs podem fornecer um bom desempenho de generalização em problemas de classificação de padrões, apesar

de não incorporarem conhecimento do domínio do problema (HAYKIN, 1998). Na Tabela 3.1, apresenta-se uma comparação entre as vantagens e desvantagens de usar cada uma das abordagens computacionais citadas previamente.

Tabela 3.1: Comparação entre as abordagens computacionais para reconhecimento de regiões promotoras.

Metodologia	Técnica	Vantagens	Desvantagens
Reconhecimento de Sinais (RS)	Sequencia Consenso (SC)	- simplicidade.	- alta variação dos nucleotídeos. - assume a independência entre bases adjacentes.
	Matrizes de Posições Ponderadas (MPP)	- eficiente em sequências pequenas.	- não permite múltiplos elementos dos promotores. - varia de acordo com o método de alinhamento.
Aprendizado de Máquina (AM)	Modelo Oculto de Markov (MOM)	- captura regularidades em sequências de caracteres.	- conjunto de treinamentos grandes. - incorporação de características dos promotores.
	Máquinas de Vetor de Suporte (SVM)	- desempenho de generalização na classificação.	- não incorpora conhecimento do domínio do problema.
	Rede Neural Artificial (RN)	- reconhecer padrões degenerados. - desempenho em sequências genômicas.	- sincronismo nos dados de entrada (alinhamento de sequências).

3.2.1 Redes Neurais Artificiais (RN)

As redes neurais artificiais foram originalmente desenvolvidas com o objetivo de modelar o processamento de informação e aprendizagem do cérebro (BERNARDA, 2007). Segundo Rebouças (2011), as estruturas baseadas em redes neurais são generalizações de modelos lineares que são apropriadas para a identificação de sistemas não-lineares. Elas podem ser utilizadas com diversos propósitos: classificação de padrões, filtragem de sinais, análise de imagens, identificação, síntese e reconhecimento de fala, *interface* adaptativa entre humanos e sistemas físicos complexos e aproximação de funções (BALDI e BRUNAK, 2001).

As redes neurais consistem em camadas de unidades de processamento com conexões entre elas. Trata-se de estruturas paralelas, distribuídas, constituídas por unidades simples de processamento conhecida como neurônios artificiais, que

computam determinadas funções matemáticas, geralmente não-lineares, a partir de entradas recebidas. Estas unidades são dispostas paralelamente constituindo camadas e são interligadas com as unidades de camadas vizinhas por conexões geralmente associadas a pesos.

Por analogia às interligações dos neurônios no cérebro humano, as conexões entre unidades de uma rede neural são chamadas “sinapses”, sendo os pesos denominados “pesos sinápticos”. Assim como os neurônios reais, eles têm conexões de entrada (dendritos) e conexões de saída (axônios). Cada neurônio processa a informação recebida, gerando assim um sinal de saída. Haykin (1998) afirma que existem duas diferenças fundamentais entre neurônios reais e uma unidade da rede neural:

- A saída de um neurônio biológico é um pulso de sinal modulado (série de pulsos de amplitude fixa) com sua frequência mudando em resposta aos sinais recebidos pelos dendritos, enquanto que o neurônio artificial tem como saída um número;
- a saída de um neurônio biológico muda continuamente com o tempo e já a de um artificial apresenta mudanças somente em um intervalo discreto de tempo, conforme é ilustrado na Figura 3.6.

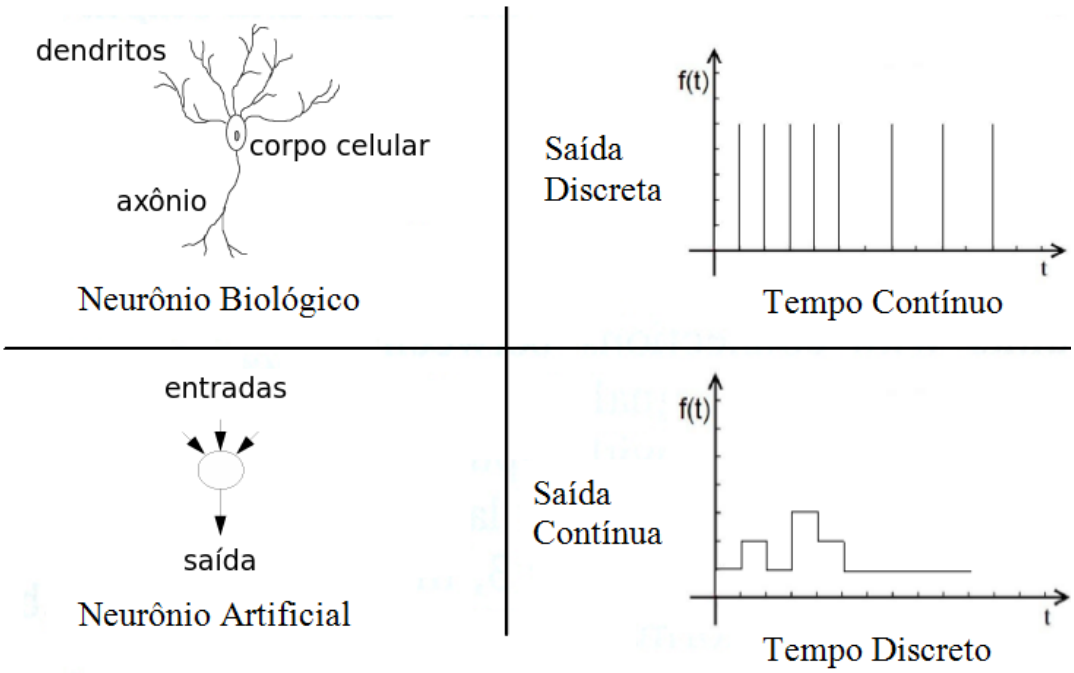


Figura 3.6: Analogia entre neurônios biológicos e artificiais.

* Adaptado de Wu e Mclarty (2000).

O treinamento de uma rede neural visa atribuir valores a um conjunto de pesos (inicializados aleatoriamente), aplicar os dados de entrada à rede e verificar como esta responde a determinados conjuntos de pesos. Caso o desempenho seja insatisfatório, os pesos devem ser modificados pelo algoritmo algoritmo da arquitetura e deve-se repetir o procedimento. Este procedimento é repetido até os pesos proporcionarem à rede a capacidade de representar o problema em questão, normalmente representado pelo alcance de um critério de parada pré-especificado (WU e MCLARTY, 2000). Uma vez treinada, os pesos são fixados e a rede pode ser empregada como um modelo, estimando saídas a partir de um conjunto de dados de entrada.

O procedimento utilizado para realizar o processo de aprendizagem procura modificar os pesos sinápticos de forma a alcançar o objetivo desejado. Os algoritmos de aprendizado podem ser supervisionados ou não-supervisionados, embora eles possam ser usados em conjunto dependendo da arquitetura.

O treinamento supervisionado é acompanhado pela apresentação de uma sequência no vetor de treinamento associada com um vetor de saída alvo. Um algoritmo de aprendizado supervisionado tem o objetivo de minimizar a diferença entre o valor de saída da rede (y_i) e o valor desejado através do auxílio de um especialista (h_w) (WU e MCLARTY, 2000). Uma típica função de erro a ser minimizada é representada pela Equação 3.11:

$$E = \sum_{i=1}^{n} (y_i - h_w(x))^2 \qquad (3.11)$$

Enquanto isto, o treinamento não-supervisionado não possui o auxílio de um especialista externo para verificar o processo de aprendizado. Neste tipo de rede, modificam-se os pesos até que os vetores mais similares sejam designados ao mesmo grupo de saída (*clusterização)*, o qual é representado por um vetor-exemplo.

O treinamento das redes neurais pode ser realizado através de um algoritmo que se baseia na regra de aprendizagem por correção de erro. Denomina-se algoritmo de retropropagação (*backpropagation*) (RUMELHART *et al*., 1988). Trata-se do algoritmo mais difundido para treinamento supervisionado de redes MLP

(*Multilayer Perceptron*) (RUMELHART *et al*., 1988). O algoritmo é dividido em dois processos: (1) propagação (*feedfoward*) e (2) retropropagação (*backpropagation*).

A propagação (passo 1) do sinal funcional a partir das entradas trata do envio do sinal funcional pela rede até a geração de uma saída, mantendo-se fixos os pesos das sinapses. Já na retropropagação do erro (passo 2), a saída é comparada com um alvo produzindo um sinal de erro. Este sinal propaga-se da saída para entrada, e os pesos são ajustados de maneira a minimizar o erro.

A cada passo *t* do treinamento, calcula-se o sinal de erro de cada neurônio *j* de saída da rede MLP. Para uma rede neural com *i* neurônios de saída, a função de custo é definida como o somatório dos *i* sinais de erro (Equação 3.1).

$$E(t)=\frac{1}{2}\sum_{j=1}^{i} e^{2}_{j}(t) \qquad (3.1)$$

A correção a ser aplicada a cada peso de um neurônio *j* de saída é dada pela derivada parcial de *E(t)* em relação ao peso em questão (Equação 3.2). Ou seja, é calculada utilizando-se a regra da cadeia pelo método do gradiente para minimização do erro (HAYKIN, 1998).

$$\Delta W_{ji}(t)=-\eta\,\partial E(t)/\partial W_{ji}(t)=\eta e_{j}(t)\varphi'_{j}(v_{j}(t))\,x_{i}(t) \qquad (3.2)$$

A derivada da função de ativação do neurônio é dada pela expressão $\varphi'_{j}(v_{j}(t))$. Sendo assim, é possível calcular o gradiente local agrupando o erro ($e_{j}(t)$) e a derivada da função de ativação ($\delta_{j}(t)$), Equação 3.3.

$$\delta_{j}(t)=\varphi'_{j}(v_{j}(t))e_{j}(t) \qquad (3.3)$$

Embora não seja viável calcular os erros dos neurônios ocultos diretamente, eles também são responsáveis pelo erro total cometido na saída da rede. Desta forma, o sinal de erro deve ser retropropagado, tornando possível o ajuste dos pesos nas camadas ocultas. Sendo o neurônio *j* um neurônio oculto, ligado à neurônios *k* da camada de saída, o algoritmo calcula o gradiente local (Equação 3.4).

$$\delta_j(t) = \varphi'_j(v_j(t)) \sum_k \delta_k(t) w_{kj}(t) \qquad (3.4)$$

O gradiente local para um neurônio *j* oculto é calculado através do somatório dos gradientes locais de cada neurônio *k* da camada seguinte, ponderados pelas ligações sinápticas que ligam cada neurônio *k* ao neurônio *j*. Trata-se de uma estimativa determinada recursivamente em termos dos sinais de erro de todos os neurônios da camada posterior. Para ajustar os pesos da camada oculta, deve-se realizar o computo da Equação 3.5, utilizando-se a Equação 3.4 para $\delta_j(t)$.

$$\Delta W_{ji}(t) = -\eta \delta_j(t) x_i(t) \qquad (3.5)$$

Os neurônios são conectados por vínculos orientados, como pode ser observado na Figura 3.7. Assim, pode-se representar uma rede neural como um grafo direcionado com peso (MOUNT, 2013). Um vínculo da unidade *j* para a unidade *i* serve para propagar a ativação a_j desde *j* até *i*. Cada vínculo também tem um peso numérico w_{ij} associado a ele, o qual determina a intensidade e o sinal da conexão.

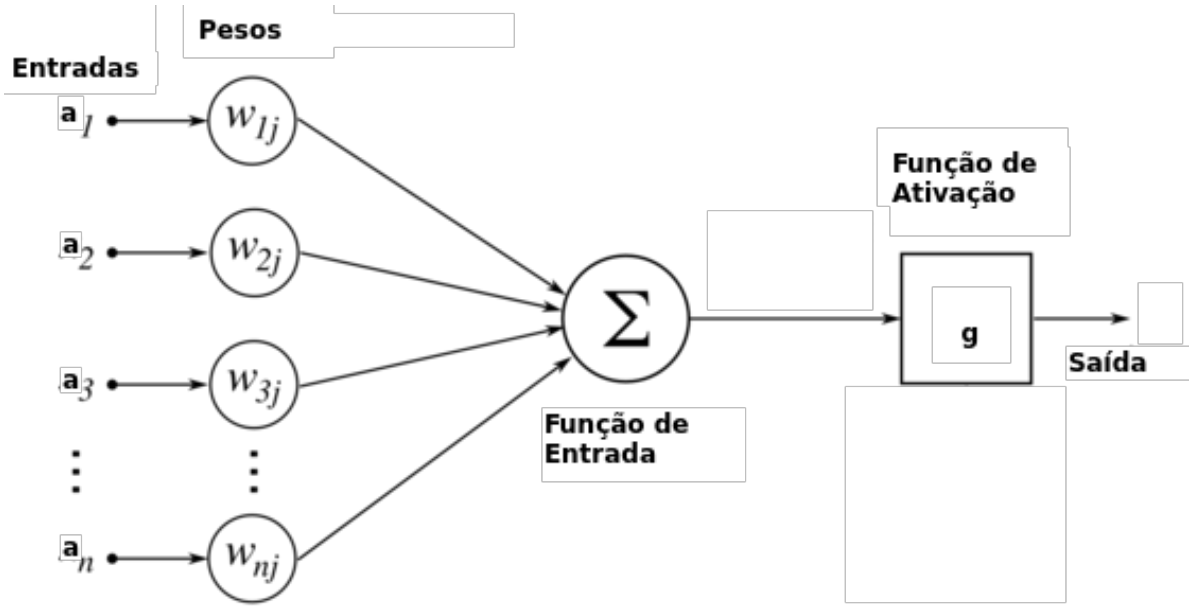


Figura 3.7: Modelo de um neurônio artificial.

* Adaptado de Mount (2013).

Cada unidade *i* calcula inicialmente uma soma ponderada de suas *n* entradas, como pode ser observado na Equação 3.6. Especificamente, um sinal a_j na entrada da sinapse *i* conectada ao neurônio *j* é multiplicada pelo peso sináptico w_{ij} (RUSSEL e NORVIG, 2003).

$$in_i = \sum_{j=0}^{n} w_{ij} a_j \qquad (3.6)$$

Então é aplicada uma função de ativação *g* a este somatório para derivar a saída (Equação 3.7). A função de ativação *g* restringe a amplitude do sinal de saída ao condicionar a ativação do sinal à ultrapassagem de um determinado limiar pelo valor da soma ponderada das entradas. Os intervalos típicos de normalização de saída são [0,1] e [-1, 1].

$$a_i = g(in_i) = g(\sum_{j=0}^{n} w_{ij} a_j) \qquad (3.7)$$

A função de ativação é projetada para atender duas aspirações: *(i)* a unidade de estar ativa (próxima de *+1*) quando as entradas positivas forem recebidas e negativas (próxima a 0) quando as entradas não-esperadas forem recebidas; *(ii)* a ativação precisa ser não-linear, caso contrário a RN inteira entrará em colapso, tornando-se uma função linear simples (RUSSEL e NORVIG, 2003).

Os tipos mais comuns de funções de ativação podem ser vistas na Figura 3.8: (a) Função Linear, (b) Função Limiar (degrau), (c) Função Rampa e (d) Função Sigmóide. A função linear é definida por uma reta: $F(v_k) = \alpha v_k$, sendo α um escalar positivo que indica a inclinação da reta e v_k um número real. Já na função Limiar, a saída do neurônio é igual a zero, quando seu valor for negativo e um, quando seu valor for positivo, como pode ser observado na Equação 3.8. Ressalta-se que o valor de ativação *v* é composto pelo combinador linear e pelo *bias,* $v(j) = \sum_{m}^{i=1} w_{ij} x_i + b$.

$$F(v_k) = \begin{matrix} 1, v \geq 0 \\ 0, v < 0 \end{matrix} \qquad (3.8)$$

A função rampa é definida pela Equação 3.9. Nesta função, os valores máximo e mínimo da saída do neurônio são *+1* e *-1*. Os valores de saída podem variar de acordo com o intervalo [-a, +a], sendo *a* diferente de zero.

$$F(v_k) = \begin{matrix} +1, u \geq a \\ v_k, -a < v_k < a \\ -1, v_k \leq -a \end{matrix} \qquad (3.9)$$

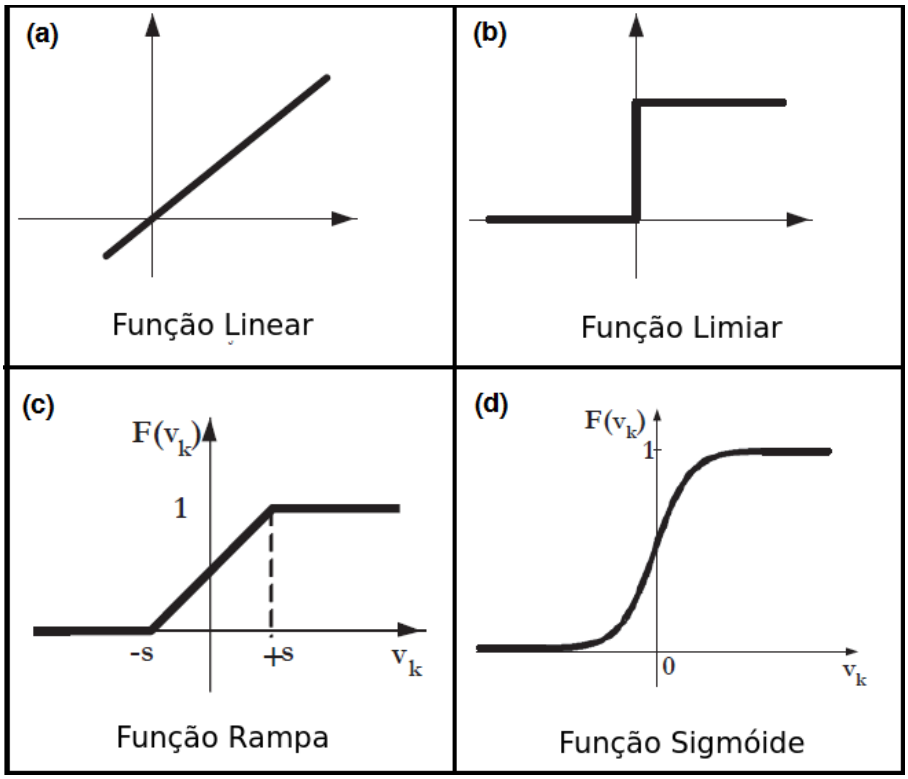


Figura 3.8: Tipos de funções de ativação de um neurônio.

* Adaptado de Russel e Norvig (2003).

Por fim, a função sigmóide é o tipo de função de ativação mais utilizado em redes neurais artificiais. Ela é definida como uma função crescente, que apresenta um balanço entre o comportamento linear e não-linear. Um exemplo de função sigmóide é a função tangente hiperbólica, definida pela Equação 3.10. Onde *a* é o parâmetro de inclinação da função sigmóide e *v* é o valor de ativação do neurônio.

$$F(v_k)=(1-e^{-av})/(1+e^{+av}) \qquad (3.10)$$

Esta função, ao contrário da função limiar, pode assumir todos os valores entre *0* e *1*. A representação mais utilizada para esta função é a função logística, definida pela Equação 3.11:

$$F(v_k)=1/(1+e^{-av}) \qquad (3.11)$$

Além da definição de qual função de ativação deve ser utilizada, deve-se definir também a quantidade de camadas da rede neural. Redes neurais com múltiplas camadas contêm um conjunto de neurônios de entrada, camada(s) intermediária(s) ou oculta(s) e uma camada de saída. Redes com mais de uma camada são denominadas *perceptrons* de múltiplas camadas (*Multilayer Perceptron* - MLP). Como exemplo, a Figura 3.9 apresenta uma MLP com *80* entradas, *1* camadas intermediárias com *8* neurônios cada e uma camada de saída com um neurônio, produzindo uma única informação de saída.

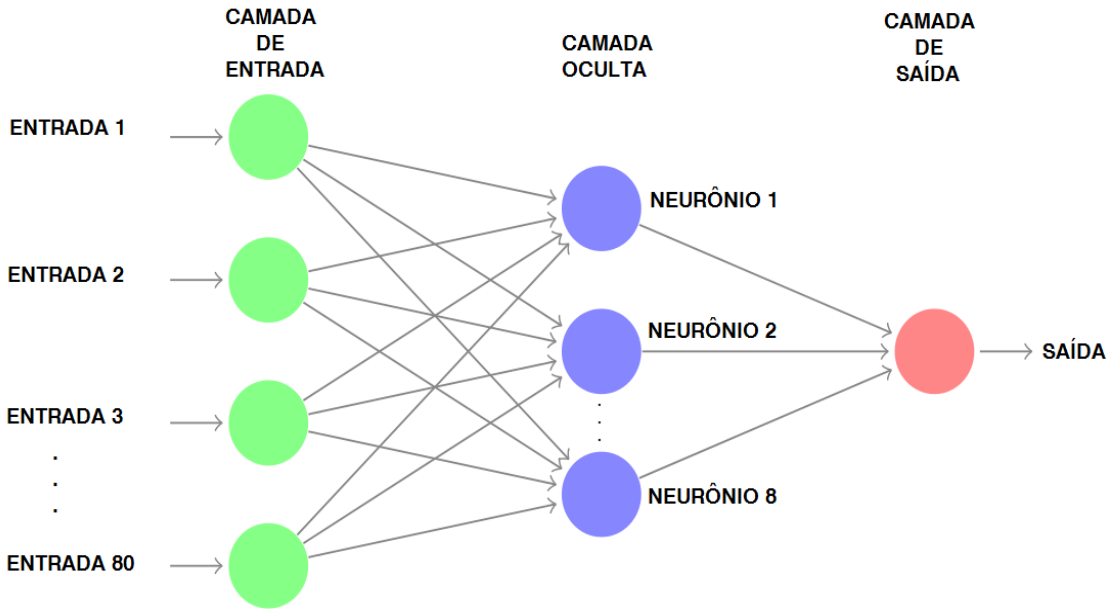


Figura 3.9: Arquitetura MLP com duas camadas ocultas.

Segundo Haykin (1998), a topologia MLP supervisionado é uma das arquiteturas mais utilizadas recentemente em pesquisas de redes neurais. Isto

ocorre devido ao seu alto potencial para ajustes funcionais, devido às suas capacidades de aproximação (HORNIK *et al*., 1989).

Não existe na literatura uma definição em relação a melhor combinação quanto ao número de camadas e de neurônios por camada pois é muito dependente do problema em questão. Sendo assim, normalmente, constrói-se diversos modelos com características distintas. Aquele que atingir o melhor desempenho nos treinamentos é definido como melhor arquitetura para a aplicação desejada. No entanto, deve se levar em consideração que em redes de múltiplas camadas, as funções implementadas tornam-se cada vez mais complexas à medida que um sinal atravessa suas camadas.

Como forma de validar os dados utilizados na rede de maneira estatística, pode-se utilizar a técnica estatística de validação cruzada (*k-fold cross validation*). O objetivo dela é o particionamento do conjunto de dados em k subconjuntos mutualmente exclusivos. A partir desta divisão, utiliza-se alguns destes subconjuntos para o treinamento da rede e o restante dos dados (teste) são empregados na validação da rede.

3.2.2 Máquinas de Vetor de Suporte (SVM)

Trata-se de uma técnica de aprendizagem de máquina supervisiona, contendo apenas uma camada escondida. Ela é baseada no principio de minimização do risco estrutural (*Structural Risk Minimization* - SRM), proporcionando assim alta capacidade de generalização e baixa necessidade de configuração. No entanto, o maior problema encontrado na utilização da técnica SVM é o alto custo computacional imposto pela mesma já que o custo aumenta quadraticamente conforme se aumenta o número de vetores de treinamento. É possível reduzir este custo através da diminuição do conjunto de treinamento através da pré-seleção de vetores. Além disso, diversas técnicas tem sido desenvolvidas para tentar reduzir este custo, mas a mais utilizada atualmente é a seleção aleatória dos vetores de suporte (XUE, YANG e CHEN, 2009).

Boser *et al*. (1992) fizeram a primeira descrição formal das Máquinas de Vetores Suporte. Elas têm se tornado uma alternativa eficiente às redes neurais (subseção 4.1.1) e vêm sendo aplicadas em problemas de diversas áreas de

conhecimento: (1) mensagens de correio eletrônico indesejadas (DRUCKER *et al*., 1999), (2) análise de expressão gênica (BROWN *et al*., 2000; VALENTINI, 2002), (3) categorização de textos (JOACHIMS, 1998), (4) reconhecimento de sinais olfativos (DISTANTE *et al*., 2003), (5) reconhecimento de faces (JEFFREY e SHAOGANG, 2002; WANG *et al*., 2002; PANG *et al*., 2003), (6) segmentação de imagens (KOTROPOULOS e PITAS, 2003).

Basicamente, SVMs trabalham com funções não-lineares que realizam um mapeamento do espaço de entrada em um espaço de dimensão elevada, conhecido como "espaço de características". Neste espaço, o hiperplano de separação entre as classes permite a classificação das mesmas. Podem ser utilizadas funções diversas neste processo: (A) lineares, (B) polinomiais, (C) gaussianas (RBF - *Radial Basis Functions*) e (D) sigmóides.

A classificação em uma SVM é realizada considerando o erro de treinamento e a margem de separação entre as classes. A margem de separação pode ser controlada pelo especialista e está relacionada com o erro de classificação permitido no hiperplano de separação.

Considere um conjunto de treinamento $\{(x_i, y_i)\}_{i=1}^{p}$ contendo duas classes de padrões representados por (*+1*) e (*-1*), sendo *x* o vetor de entrada, *y* a saída desejada para o padrão de entrada *x* e *p* o número de padrões de entrada. Um vetor pode ser considerado corretamente classificado quando a Equação 3.12 for satisfeita, onde $\varphi_i(\cdot)$ representa o mapeamento do vetor de entrada x_j em um espaço de dimensão elevada, denominado de "espaço de característica" por Cover (1965). As componentes do vetor de peso são dadas por *w*, o termo de polarização por *b* e a função de mapeamento por φ (LORENA *et al*., 2007).

$$y_i\left(\sum_{i=1}^{n} w_i \varphi_i(x_j) + b\right) \geq 1 \; para \; j = 1,2,\ldots,p \qquad (3.12)$$

Durante o treinamento, é feita uma seleção dos vetores considerados mais importantes no processo de definição dos limites de cada classe. Esses vetores definem o hiperplano com os menores erros de classificação e com a máxima margem de separação (Figura 3.10), e pode ser representado em sua forma vetorial na Equação 3.13.

$$w^T \varphi(x) + b = 0 \qquad (3.13)$$

Onde $\varphi(x) = [\varphi_1(x)\varphi_2(x)...\varphi_p(x)]^T$ e $w = [w_1 w_2 ... w_p]^T$, sendo x^T a transposta do vetor *x*.

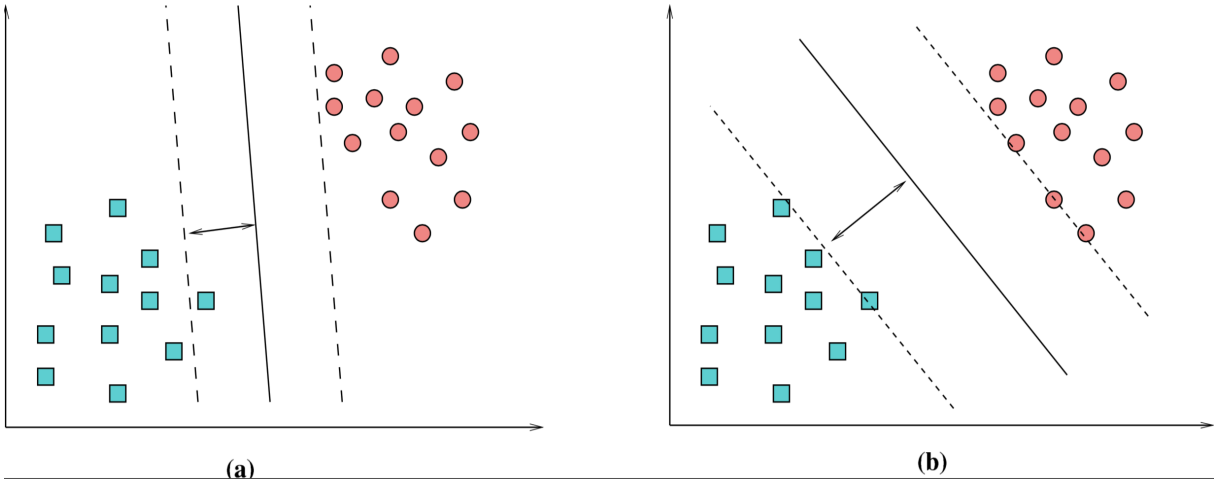


Figura 3.10: SVMs e a Máxima Margem de Separação: (a) Uma possível superfície de separação e (b) o hiperplano com máxima margem de separação.

* Os quadrados são os exemplos positivos e os círculos são os exemplos negativos. Adaptado de Lorena et al. (2007).

A classificação de um vetor como de suporte é realizada com base no valor de cada multiplicador de Lagrange obtido após a otimização dos vetores. Multiplicadores de Lagrange nulos indicam vetores corretamente classificados, mas que estão situados fora da região entre as margens e são descartados. Já os vetores situados sobre a margem estão associados a multiplicadores de Lagrange na faixa $0 < \alpha_i < C$. Finalmente, vetores situados fora da margem são indicados por multiplicadores de Lagrange igual a *C*. Este dois últimos grupos de vetores são conhecidos como vetores de suporte e irão definir o hiperplano de separação entre as classes. Estas três situações são descritas na Figura 3.11 (LORENA *et al*., 2007).

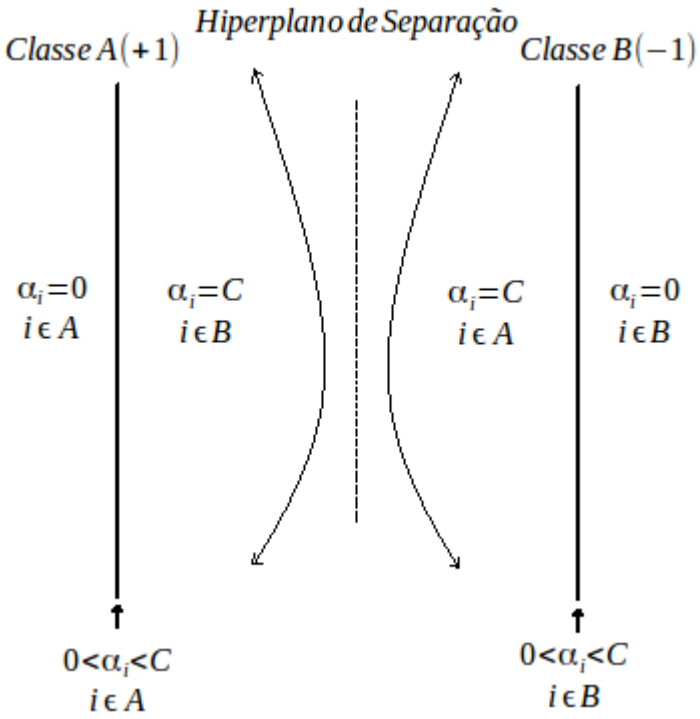


Figura 3.11: Interpretação Geométrica dos Vetores de Suporte.

* Adaptado de Luenberger (1986).

As condições de Karush-Kuhn-Tucker (KKT) (LUENBERGER, 1986) para o problema de classificação permitem definir os vetores da seguinte forma:

1. Vetor comum: $\alpha_i=0, y_i f(x_i)>1$ (situado no lado correto);
2. vetor de suporte: $0<\alpha_i<C, y_i f(x_i)=1$ (situado sobre a margem do lado correto);
3. vetor de suporte: $\alpha_i=C, y_i f(x_i)<1$ (situado no lado errado).

Na Figura 3.12, é apresentada a região factível para um caso 3D. A região factível é formada pela interseção entre as restrições de igualdade e desigualdade, gerando assim um hipercubo de lado *C* e um hiperplano. Durante a otimização, devem ser gerados valores para os multiplicadores de Lagrange dentro da região

factível e a interseção entre as restrições de igualdade e desigualdade deve ser válida.

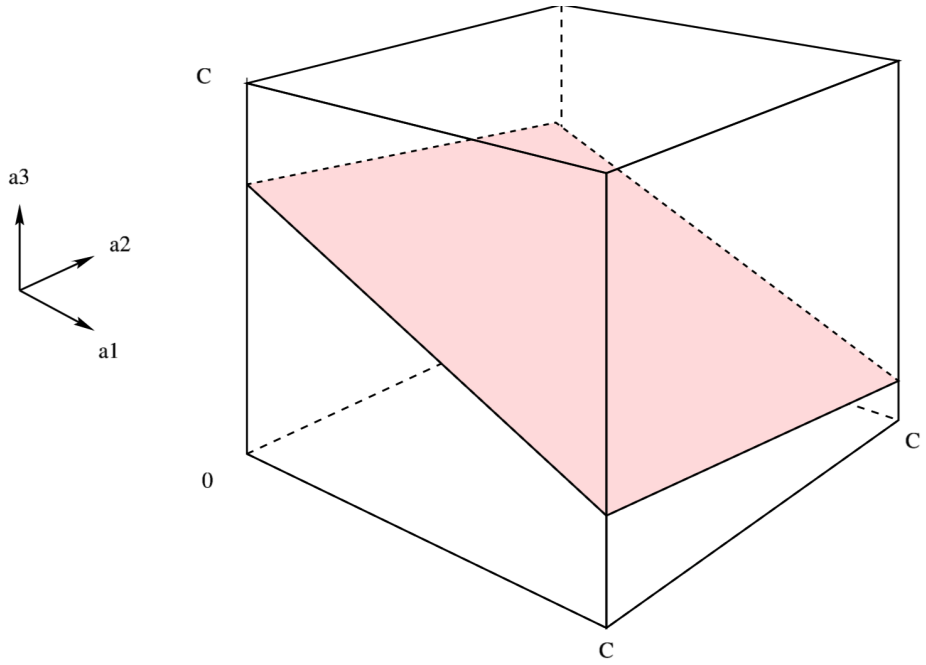


Figura 3.12: Região Factível para um Caso 3D SVM.

* Adaptado de Campbell (2002).

Boser *et al*. (1992) propuseram o primeiro algoritmo de treinamento para máquinas de vetor de suporte. Basicamente, ele maximiza a margem de separação entre vetores de treinamento e um hiperplano de separação, sendo esses vetores chamados de “vetores de suporte". No entanto, o desempenho deste método só é satisfatório quando se utiliza padrões de treinamento linearmente separáveis. SVMs baseadas neste algoritmo são conhecidas como SVMs com margens rígidas (CORTES e VAPNIK, 1995).

O processo de treinamento definido por Cortes e Vapnik (1995) foi adaptado em diversos trabalhos através do uso de “variáveis de folga" (VAPNIK, 1998; CRISTIANINI e SHAWE-TAYLOR, 2000; CAMPBELL, 2002), criando assim as SVMs com “margens flexíveis". Desta forma, foi possível obter hiperplanos de separação com margem máxima e menores erros de classificação, mesmo com padrões linearmente não-separáveis. Os principais métodos de treinamento para SVM são apresentados na sequência.

O treinamento de uma máquina de vetor de suporte consiste basicamente em solucionar um problema quadrático (*Quadratic Problem* - QP) com restrições lineares que depende dos vetores de treinamento, alguns parâmetros da função utilizada e um limite para os multiplicadores de Lagrange. A partir do treinamento, é possível distinguir quais vetores são essenciais na definição do hiperplano de separação entre as classes. Estes são chamados de vetores de suporte.

Segundo Almeida (2002), os métodos de treinamento para SVM podem ser divididos nas seguintes categorias: (1) Clássicos, (2) Geométricos, (3) Iterativos e (4) Conjuntos Ativos. Estes métodos são descritos no decorrer desta seção e uma comparação entre os quatro tipos de solução é apresentada na Tabela 3.2.

Tabela 3.2: Comparação entre as Categorias de Métodos de Treinamento SVM.

Categoria	Estratégia	Limitação	Exemplo
Clássico	Matriz de *Kernel* e QP	Alto custo de armazenamento	LSSVM
Geométrico	Algoritmo NPA	Classes bem definidas e unimodais	CSVM
Iterativo	Otimização iterativa de Lagrange	Treinamento para mínimos locais	SOR
Conjuntos ativos	Divisão em problemas menores	Treinamento para mínimos locais	SMO

Os métodos clássicos baseiam-se na construção da matriz de *kernel* (maiores informações sobre este tipo de matriz em Almeida (2002)) e no emprego de rotinas de programação quadráticas. Este tipo de método é eficiente em conjuntos de treinamento com poucas centenas de vetores. Conforme a matriz aumenta, o custo de armazenamento aumenta proporcionalmente, o que pode tornar a solução inviável. Segundo Suykens e Vandewalle (1999), é possível contornar este problema reduzindo o mesmo à um problema de programação linear (*Least Square* SVM} - LSSVM). No entanto, com esta solução ainda é necessário gerar um grande número de vetores de suporte.

Keerthi *et al*. (1999) reformularam o problema de treinamento de vetores de suporte através do algoritmo *Nearest Point Algorithm* (NPA), gerando assim os métodos conhecidos como geométricos. No mesmo contexto, encontra-se na literatura o trabalho desenvolvido por Zhang (1999), chamado *Central Support Vector Machine* (CSVM). Eles aproveitam os centros das aglomerações de dados para reescrever as fórmulas de treinamento. Apesar de tornar a solução menos sensível a ruídos, este trabalho tem a limitação de só poder ser utilizado em casos onde as classes são bem definidas e unimodais.

Já os métodos iterativos visam otimizar iterativamente um multiplicador de Lagrange por vez, utilizando informação do gradiente a partir de uma estimativa inicial. Devido à convexidade do problema, a função objetivo é monotonicamente aumentada a cada iteração, até que o seu máximo seja alcançado. SOR (*Successive Over Relation*) é um exemplo deste tipo de método e através deste é possível obter uma regra para atualização dos multiplicadores de Lagrange (MANGASARIAN e MUSICANT, 1999).

Por fim, existe uma estratégia de otimização para o treinamento que visa a divisão do problema original em problemas menores (OSUNA *et al*., 1997). Cada sub-problema é otimizado separadamente. Sendo assim, estes métodos são baseados em conjuntos ativos. Os subproblemas são determinados através de heurísticas deve-se manter as condições KKT. Como exemplo, podemos citar o método *Sequential Minimal Optimization* (SMO) (PLATT, 1998; KEERTHI *et al*., 1999). Trata-se de uma solução que resolve a questão do custo de armazenamento da matriz de *kernel* devido ao fator quadrático já que se possui uma solução analítica para o problema. Segundo Almeida (2002), o SMO é rápido e capaz de resolver problema da ordem de dezenas de milhares de vetores de treinamento. No entanto, tantos os métodos iterativos quanto os baseados em conjuntos ativos apresentam seguinte desvantagem: o método pode direcionar o treinamento para mínimos locais uma vez que padrões aprendidos são “esquecidos" de forma gradual caso não sejam mais apresentados.

3.3 RN E SVM APLICADAS NA PREDIÇÃO DE PROMOTORES BACTERIANOS

As primeiras aplicações de Redes Neurais Artificiais na predição de promotores, apresentados nos trabalhos de Demeler e Zhou (1991) e O'Neill (1991), obtiveram uma alta acurácia na predição, mas o número de falsos positivos foi igualmente alto.

Outra abordagem foi apresentada por Mahadevan e Ghosh (1994) através de uma combinação entre duas RNs para a identificação de promotores de *E. coli*. Todos os promotores deste trabalho tinham espaçamento entre 15 e 21 nucleotídeos entre os hexâmeros característicos. A primeira RN predizia os hexâmeros consensuais, enquanto a segunda foi designada para o reconhecimento da

sequência (65 nucleotídeos), sendo o espaço entre os hexâmeros variável. Uma vez usada a informação da sequência inteira, ocorreram dependências entre as bases em várias posições. Isto refletiu em um treinamento pobre e uma predição realizada por duas redes sem neurônios na camada oculta.

Pedersen e Engelbrecht (1995) predisseram o local de início da transcrição (TSS), a medida do conteúdo da informação e identificaram novos sinais característicos correlacionados com o local de início. Para isto, foram usados dois diferentes esquemas de codificação, com janelas de 51 e 65 nucleotídeos. Uma ideia interessante, neste trabalho, foi a medida do conteúdo de informação relativa dos dados de entrada, pelo uso da habilidade da RN para aprender corretamente, como avaliado pelo coeficiente de correlação do teste máximo.

Outra ferramenta baseada em RNs é o *Nureka Artificial Neural Systems* (NANS). Oppon (2000) executou um teste neste sistema a partir do conjunto composto de 31 sequências de 75 bases, sendo 5 regiões promotoras e 26 regiões codificantes de *Echerichia coli*. Com um limiar de corte de 6, o NANS acerta classificar uma sequência como promotora 60% das vezes e em afirmar que não é promotora em 50% das vezes. Uma provável explicação para este baixo desempenho, além do baixo número de amostras, é a especificidade do sistema, que foi desenvolvido especificamente para um organismo. Em 2005, Burden *et al*. melhoraram este sistema incorporando nele a informação sobre a distância entre o sítio de início de transcrição (TSS) e o sítio de início da tradução TLS (primeiro nucleotídeo do gene). Com um conjunto de dados de 771 promotores, eles melhoraram a predição em 60%, quando comparado ao trabalho de Oppon (2000), e reduziram o número de falsos positivos.

Cotik *et al*. (2005) criaram uma metodologia híbrida, denominada HPAM (*Hybrid Promoter Analysis Methodology*) que combina redes neurais, lógica *Fuzzy* e algoritmos genéticos. Seu funcionamento pode ser descritos pelos seguintes passos: (1) decomposição através da rede neural em módulos dos motivos das regiões de ligação referentes a sequências não específicas de DNA, (2) inferências de lógica *Fuzzy* para associação entre os módulos identificados pela rede e (3) utilização do método de reconhecimento de padrões que usa algoritmos genéticos chamado MOSS (*Multi-objective Scatter Search*) para encontrar os motivos mais representativos (BAJESTANI et al., 2009).

Na sequência, Santos *et al*. (2008) propuseram um método de aprendizado para um classificador *Bayesiano* para reconhecimento de promotores procarióticos. Para isto, implementaram em Java e Matlab um modelo de classificador tendo como base o método *Naive Bayes* para identificação de promotores reconhecidos pelo fator Sigma70. A bactéria utilizada foi a *Echerichia coli* e o método apresentou acurácia de 90%, mas a quantidade de sequências utilizadas foi 99.

Bland *et al*. (2010) utilizaram redes neurais artificiais que usavam perfis de dados SIDD (*Stress-Induced Duplex Destabilization*). Ou seja, redes que levam em consideração propriedades estruturais para a predição de uma região promotora. Segundo os autores, houve um ganho significativo no desempenho da predição quando utilizado SIDD juntamente com redes neurais. Eles utilizaram o genoma de *E. coli* e locais de inícios de transcrição do banco de dados Regulon, totalizando 1648 promotores. A melhor acurácia obtida ficou na faixa de 70% a 80%.

De Avila e Silva et al. (2011) desenvolveram um *software* chamado BacPP (*Bacterial Promoter Predicition*) cujo objetivo é a utilização de Redes Neurais Artificiais na predição, caracterização e reconhecimento de promotores de bactérias Gram-negativas a partir de um determinado sigma. Ele está disponível em versão Web (implementada em linguagem PHP e Python) e versão *desktop* (implementada em linguagem Python e linguagem R). Para utilizar este sistema, o usuário deve inserir a sequência de nucleotídeos ou enviar um arquivo no formato FASTA. Após, ele deve selecionar fatores σ e escolher o formato de saída (monitor ou arquivo). Os resultados obtidos com o BacPP foram satisfatórios e comparáveis com a literatura. Através da análise da melhor arquitetura para cada sigma, eles obtiveram acurácia média de 71.67%, especificidade de 71.08% e sensibilidade de 72.98%, com baixa variação entre os fatores sigma (σ70 obteve o melhor desempenho, 77% de acurácia). Através de suas simulações, foi possível perceber que o aumento do número de neurônios na camada oculta não necessariamente melhora o desempenho da rede neural.

Meng *et al*. (2013) desenvolveram uma biblioteca para melhorar a eficiência do treinamento da rede neural através da análise quantitativa dos possíveis pontos de mutação e de sequências conservadas. Eles utilizaram a ferramenta *Neural Network Toolbox* disponibilizado no Matlab. Eles configuraram a rede com três camadas, sendo uma oculta. Foram utilizados 896 neurônios na camada de entrada

e 1 neurônio na camada de saída. No entanto, sua validação foi feita apenas com 100 sequências, o que não permite avaliar de maneira satisfatória o seu desempenho devido a pequena amostragem.

Por fim, os trabalhos mais recentes encontrados sobre predição de promotores através do uso de redes neurais artificiais foram realizados por Umarov e Solovyev (2017) e por Kumar e Bansal (2017).

Umarov e Solovyev implementaram um *software* em linguagem de programação Python (biblioteca Keras) chamado CNNProm que usa rede neural convolucional e obteve alta acurácia (*E. coli* 84% e *B. Subtilis* 86%), porém apenas utilizaram σ70 totalizando 746 sequências promotoras com 81 nucleotídeos cada. Seus dados foram coletados do banco de dados DBTBS (SIERRO et al., 2008). Este tipo de rede se baseia na organização do *cortex* visual dos animais para usar como padrão de conectividade entre os neurônios, permitindo a redução de conexões desnecessárias e o compartilhamento dos pesos das arestas.

Kumar e Bansal focaram seus esforços na análise de características estruturais (baixa estabilidade, baixa flexibilidade e alta curvatura) de promotores de *E. coli*. Eles afirmam que regiões promotoras associadas com alta expressão genética tem estruturas de baixa estabilidade, mais rígidas e maior curvatura se comparados com outros genes. Eles utilizaram sequências de 1001 nucleotídeos divididos em 5-*Fold*, selecionando 80% dos dados para treinamento.

A maioria dos trabalhos, que usam a tecnologia SVM para predição de promotores, utiliza como objeto de pesquisa bactérias Gram-negativas (KIRYU *et al*., 2005; POLAT e GUNES, 2007; GORDON *et al*., 2006; GORDON *et al*. 2007; DAMASEVICIUS, 2010) ou dados genéricos (ZANATY, 2012). As abordagens mais similares ao presente trabalho foram feitos por Monteiro *et al*. (2005), De Jong *et al* (2012) e Meysman *et al*. (2014). O último trabalho encontrado, Siwo *et al*. (2016) não discrimina o tipo de dado utilizado em sua pesquisa.

KIRYU *et al*. (2005) utilizaram matriz ponderada e SVM de regressão e o banco de dados micro arranjo KEGG, porém obtiveram apenas 96 perfis genéticos de *E. coli*. Já Polat e Gunes (2007) criaram uma solução chamada FS_LSSVM (*Feature Selection Least Square Support Vector Machine*) que usa uma combinação de seleção de recurso e LSSVM. Eles obtiveram 80% de acurácia, mas usaram apenas 53 exemplos de promotores de *E. coli*.

Ademais, Gordon *et al*. (2007) também estudaram *E. coli* utilizando SVM com uma função de alinhamento de núcleo (*SAK - Sequence Alignment Kernel*), resultando em uma eficiência de 70% para uma amostra de 100 promotores. Eles trabalharam com dois conjuntos de dados: (i) promotores e regiões codificantes e (ii) promotores e regiões intergênicas, mas também usaram uma pequena quantidade de dados para validar sua solução.

Já Damaševičius (2010) analisou a estrutura de sequências reguladores de DNA através de inferência Gramatical e SVM (algoritmo MEME), tentando atenuar a falta de informação estrutural na abordagem SVM. Mas ele não usou como objeto de estudo nenhuma bactéria, apenas drosófilas, vertebrados e plantas monocotiledôneas.

Zanaty (2012) propôs uma nova função *kernel* chamada *Gaussian Radial Basis Function* (GRPF) para melhorar o desempenho da classificação se comparado com *Multilayer Perception*. Foram combinadas funções polinomiais com funções gaussianas, procurando melhorar a eficiência na classificação de dados de alta dimensão. Como base de dados, foi usado DNA porém não identificou qual parte do DNA. Na sequência, Gusmäo and Souto (2014) fizeram um estudo empírico sobre a seleção de regiões não-promotoras de *E. coli* em seu treinamento (validação cruzada 10-*Fold*) e afirmaram que o desempenho de um classificador pode ser prejudicada caso a seleção não seja feita corretamente. Eles usaram três técnicas de aprendizado de máquina: árvores de decisão, Naive Bayes e SVM. Eles utilizaram 812 sequências promotoras obtidas no banco de dados RegulonDB (GAMA-CASTRO et al., 2008), todas com 80 nucleotídeos cada. Tendo em vista a alta quantidade de dados disponível de *E. coli,* eles usaram um baixo número de amostras. Para os exemplos negativos, eles coletaram sequências de regiões codificantes, regiões não-codificantes e segmentos aleatórios do genoma.

Monteiro *et al*. (2005) usaram várias técnicas de aprendizado de máquina para a predição de promotores em *B. subtilis* e *E. coli*. No entanto, usaram um *software* chamado WEKA (HALL et al., 2009) e não desenvolveram sua própria solução. Além disso, foi usada uma quantidade baixa de dados para validação, apenas 53 promotores de *E. coli* e 112 promotores de *B. subtilis,* e obtiveram acurácia de 76%. De Jong *et al*. (2012) desenvolveram um *webserver* que permite a predição de promotores em *E. coli* e em duas bactérias Gram-positivas: *Lactococcus lactis* e *B.*

subtilis. Apesar de ser uma base de dados interessante e diversificada, eles não apresentam resultados.

Meysman *et al*. (2014) também estudaram bactérias Gram-positivas (*Bacillus anthracis*), mas sua atenção estava voltada para perfis estruturais de regiões promotoras e não em predição promotora. Eles afirmam que as características estruturais de *E. coli* não representam as demais bactérias e que as regiões promotoras são menos estáveis e mais rígidas que o resto do genoma, mas isto é menos visível em Gram-positivas.

Por fim, Siwo *et al*. (2016) participaram de um desafio proposto pela DREAM (*Dialogue on Reverse Engineering Assessment and Methods*) cujo objetivo era predizer a atividade de promotores fornecidos, os quais 53 sequências de 100 pb eram utilizados para teste e 90 para treino (pequeno número de amostras). Eles utilizaram três módulos do *software* WEKA: SVM usando SMO (*Sequential Minimal Optimization*), regressão linear e árvores de regressão. Sua avaliação foi realizada através do coeficiente de correlação Pearson e computada em código R, obtendo um coeficiente de 0.73 (sendo que 1 significa correlação perfeita). Foi usada validação cruzada estatística de 5-*fold* e 10-*fold*. Eles não utilizaram nenhuma informação referente a fatores de transcrição em seu modelo e usaram o *software* WEKA para fazer a predição através das técnicas SVM, regressão linear e árvores de regressão.

A Tabela 3.3 apresenta uma comparação entre os trabalhos descritos anteriormente que utilizam Redes Neurais Artificiais como abordagem computacional para o problema de predição de regiões promotoras. Já a Tabela 3.4 apresenta um comparativo entre os trabalhos que solucionam o problema de predição de promotores através de Máquinas de Vetor de Suporte. Como pode ser observado em ambas tabelas, a maioria dos trabalhos visa tratar da predição de promotores em bactérias Gram-negativas, tendo em vista a maior quantidade de dados disponíveis.

Tabela 3.3: Comparação entre as soluções que utilizam a técnica de Redes Neurais Artificiais.

Ferramenta	Bactéria	Características	Desempenho	Autores
NeuralWare	*E. coli*	- arquitetura simplificada - número de neurônios (1 a 10, 24) - *software Neuralware II Professional* (Demeler e Zhou) - linguagem FORTRAN em um SUN 386 (O'Neill) - linguagem C em um sistema UNIX (Mahadevan e Ghosh)	- número de amostras: 80, 39 e 126	Demeler e Zhou (1991); O'Neill (1991); Mahadevan e Ghosh (1994).
KullbackDist	*E. coli*	- coeficiente de correlação do teste máximo - neurônios na camada oculta (2 a 3) - linguagem C em um sistema UNIX	- número de amostras: 167	Pedersen e Engelbrecht (1995).
NANS	*E. coli; B. subtilis; Mycobacterium tuberculosis*	- análise de distribuição de frequência - *Hidden Markov Model* - *software Nureka Artificial Neural Systems*	- acurácia (50 a 60%) - número de amostras: 10 a 50 - sequências não normalizadas (40 a 75pb) - não foi feita análise de falsos negativos	Oppon (2000).
NNPP2.2	*E. coli*	- distância entre o local de início da transcrição e o local de início da tradução	- número de amostras: 272 - acurácia (64%)	Burden *et al*. (2005).
Naive Bayes	*E. coli*	- classificador *Bayesiano (método de Naive Bayes)* - *softwares* de alinhamento *WSONSENSUS e PATSER* - linguagem *Java e software* GNU Octave	- número de amostras: 99	Santos *et al*. (2008).
SIDD	*E. coli*	- perfis de dados SIDD (dados estruturais dos promotores)	- acurácia (70 a 80%) - número de amostras: 1648	Bland et al. (2010).
BacPP	*E. coli*	- análise de diferentes fatores de transcrição - validação cruzada *(k-cross validation)* - análise de desempenho através de 3 métricas: sensibilidade, especificidade e acurácia. - linguagens R e Python	- acurácia (s24, 86.9%; s28, 92.8%; s32, 91.5%; s38, 89.3%, s54, 97.0%; e s70, 83.6%). No geral, 76%. - número de amostras (s24, 69; s28; 21; s32, 71; s38, 99; s54, 38; s70, 740)	De Avila e Silva et al. (2011).
Neural Network Toolbox	*E. coli*	- análise de pontos de mutação - análise de sequências conservadas - coeficiente de correlação - neurônios na camada oculta (19) - linguagem Matlab em um Microsoft Windows 7 64-bit	- número de amostras: 100 - alto tempo de treinamento devido ao número de neurônios na camada oculta.	Meng *et al*. (2013).
CNNProm	*E. coli; B. subtilis*	- página web (*webserver*) - não utiliza nenhuma informação sobre características de promotores específicos	- acurácia (*E. coli* 84% e *B. subtilis* 86%) - apenas σ70 e σA	Umarov e Solovyev (2017).
TSS RN	*E. coli*	- análise de características estruturais (baixa estabilidade, baixa flexibilidade e alta curvatura) de promotores - 6 diferentes transcriptogramas procarióticos	- o perfil de curvatura obtido apresenta maior curva em regiões promotoras	Kumar e Bansal (2017).

Tabela 3.4: Comparação entre as soluções que utilizam a técnica de Máquinas de Vetor de Suporte.

Ferramenta	Bactéria	Características	Desempenho	Autores
SAK	*E. coli*	- função kernel de alinhamento de sequências - local de início da transcrição - matriz de posição peso	- acurácia (70%) - relação próxima entre sensibilidade e especificidade - número de amostras: 100	Gordon *et al*. (2003); Gordon *et al*. (2006).
KEGG	*E. coli*	- matriz ponderada e SVM de regressão - banco de dados KEGG - grau de conservação da sequencia e expressão do gene	- número de amostras: 96	Kiryu *et al*. (2005).
WEKA	*E. coli; B. subtilis*	- árvores de decisão, SVM, RN, Naive Bayes - *software* WEKA	- acurácia (76%) - número de amostras (57 para *E. coli* e 112 para *B. subtilis*)	Monteiro *et al*. (2005).
FS_LSSVM	*E. coli*	- *combinação de seleção de recurso e LSSVM*	- acurácia (80%) - número de amostras: 53	Polat e Gunes (2007).
MEME	Nenhuma	- analise estrutural das sequências - inferência Gramatical e SVM	- número de amostras: 6500 entre drosófilas e vertebrados.	Damaševičius (2010).
GRPF	Nenhuma	- função kernel que combina funções polinomiais com funções gaussianas para melhorar o desempenho da classificação - comparação entre RN e SVM - linguagem Matlab	- o tempo de treinamento obtido pela RN foi dez vezes mais longo que o tempo da SVM	Zanaty (2012).
PePPER	*E. coli; B. subtilis; Lactococcus lactis*	- página web (*webserver*) com uma série de ferramentas e base de dados.	- não foi realizado um teste de desempenho.	De Jong *et al*. (2012).
Z-curve	*E. coli*	- seleção de regiões não-promotoras durante o treinamento - árvores de decisão, SVM, Naive Bayes, e método Z-curve. - linguagem Matlab	- desempenho de um classificador pode ser prejudicada caso a seleção não seja feita corretamente - número de amostras (812 reconhecidos em σ19 , σ24 , σ28 , σ32 , σ38 , σ54 e σ70)	Gusmão and Souto (2014).
TSS SVM	*E. coli, Bacillus anthracis, etc.*	- perfis estruturais das regiões promotoras - local de início da transcrição - linguagem Matlab	- análise estrutural de estabilidade, curvatura, maleabilidade, etc. - as características estruturais de E. coli não representam as demais bactérias.	Meysman *et al*. (2014).
SVM SMO	Não revelado no desafio	- não utilizou nenhuma informação sobre fatores de transcrição - local de início da transcrição - *software* WEKA (SVM SMO, regressão linear e árvores de regressão) e linguagem R	- acurácia (fator de correlação de *Pearson* de 0.73, sendo 1 uma correlação perfeita)	Siwo *et al*. (2016)

4 MATERIAL E MÉTODOS

A metodologia geral empregada no presente trabalho é apresentada de forma esquematizada na Figura 4.1. Ela é composta de três passos: (4.1) coleta de dados e pré-processamento de dados; (4.2) simulação da Rede Neural através do BacPP+ e (4.3) simulação da Máquina de Vetor de Suporte através do BacSVM+. Todas as simulações foram realizadas em um *MacBook Pro* com processador 2,9 GHz Intel Core i5 e 8 GB de memória. Para a validação dos dados de forma estatística em ambas as soluções, optou-se pela metodologia de validação cruzada (*k-fold cross-validation*). Rodriguez *et al*. (2010) fizeram uma análise comparativa da estimativa de erro a partir da variação do valor de k (2, 5, 10 e n) para a validação cruzada no WEKA. Eles indicaram o valor 5 ou 10, mas devido a quantidade de dados disponíveis, o valor de k foi delimitado em 5. Os dados obtidos e os *softwares* desenvolvidos podem ser acessados a partir dos *links* a seguir:

- http://bacpp.bioinfoucs.com/rafael/Sigmas.zip
- http://bacpp.bioinfoucs.com/rafael/Passo1Dados.zip
- http://bacpp.bioinfoucs.com/rafael/Passo2BacPP+.zip
- http://bacpp.bioinfoucs.com/rafael/Passo3BacSVM+.zip

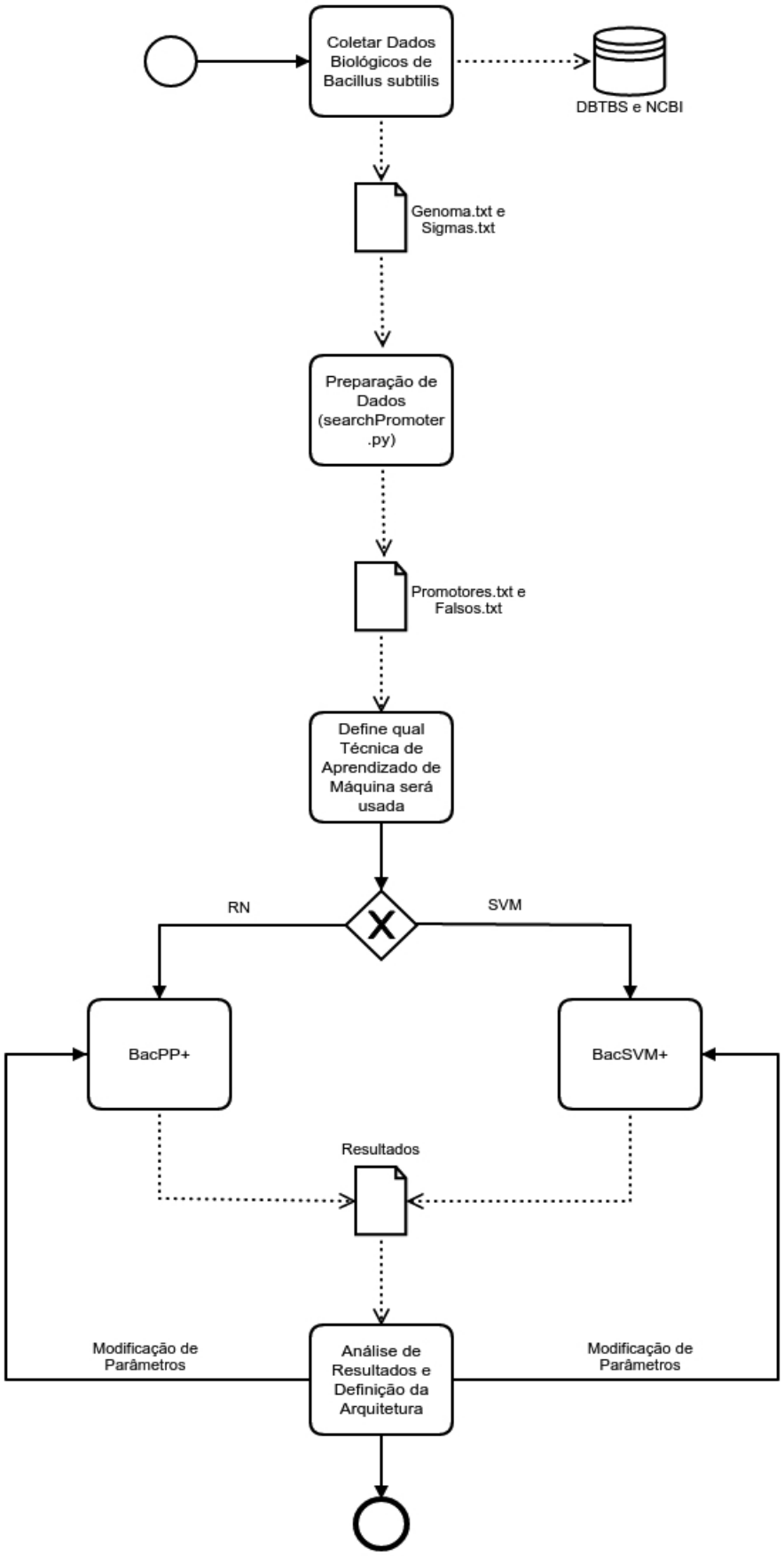


Figura 4.1: Esquema da metodologia empregada no trabalho para o reconhecimento de promotores através do uso de RN e SVM.

4.1 COLETA E PRÉ-PROCESSAMENTO DE DADOS

O primeiro desafio deste projeto foi a criação de uma base dados com as informações gênicas da bactéria *B. subtilis*. As regiões promotoras, as regiões intergênicas e os dados relacionados às características dos promotores foram obtidos de bancos de dados biológicos e de artigos científicos (HELMANN, 1995; MONTEIRO, 2005). As bases de dados públicas utilizadas foram:

- DBTBS (*Database of Transcriptional Regulation in Bacillus subtilis*): banco de dados público de genomas procarióticos (SIERRO *et al*., 2008). Aqui se encontram dados de genomas completos, de regiões específicas (genes, promotores, regiões intergênicas, homologias), além de possuir informações sobre os artigos relacionados a cada descoberta genética. Maiores informações podem ser encontradas no endereço da *Internet*: http://dbtbs.hgc.jp/.
- NCBI (*National Center for Biotechnology Information*): abriga uma série de bancos de dados públicos de biotecnologia e biomedicina. Desta, podem-se extrair sequências de genes, proteínas, genomas completos, dados de homologia e expressão gênica, além de possuir informações sobre os artigos relacionados a cada descoberta genética. Endereço da *Internet*: http://www.ncbi.nlm.nih.gov.

Inicialmente, foi obtido o arquivo FASTA (gi|225184640|emb|AL009126.3|:1-4215606 *Bacillus subtilis* str. 168 complete genome) contendo o genoma da bactéria *B. subtilis* a partir do NCBI. Foi selecionado esta linhagem por se tratar de um dos mais utilizados em pesquisas desta bactéria (ZEIGLER et al., 2008). A seguir, as sequências promotoras identificadas pelos fatores Sigma deste tipo de bactéria foram obtidos no banco de dados DBTBS (SIERRO, 2008), totalizando 15 fatores divididos em 2 domínios: (1) Sigma 54 (SigL) e (2) Sigma 70 (demais fatores Sigma). Este foi um processo bastante demorado devido a falta de dados de bactérias Gram-positivas disponíveis.

Os fatores *Sigma* da bactéria *B. subtilis* estão listados na Seção 5.1, e são apresentadas as seguintes informações: ORF (*Open Reading Frame*), descrição e número de *operons*. Destaca-se o fator Sigma SigA devido ao seu alto número de *operons* e potencial para a identificação dos promotores.

Além disso, no mesmo banco de dados foram coletadas informações relativas aos *operons* de cada fator *Sigma*. Por exemplo, no que diz respeito as evidências experimentais do *Sigma* SigL, acoABCL foi evidenciado por mapeamento de extremidades 5' do mRNA por extensão do primer para o gene acoA e por pesquisa homológica (ALI *et al*., 2001); levDEFG-sacC foi evidenciado por mapeamento de extremidades 5' do mRNA por extensão do primer para o gene levD (MARTIN *et al*., 1989), gene reportado e interrupção do fator de ligação do gene (DÉBARBOUILLE *et al*., 1991); ptb-bcd-buk-lpdV-bkdAABB foi evidenciado por *mapeamento de extremidades 5' do mRNA por extensão do primer* para o gene ptb (DÉBARBOUILLE *et al*., 1999); rocABC foi evidenciado por mapeamento de extremidades 5' do mRNA por extensão do *primer* para o gene rocA (CALOGERO *et al*., 1994); rocDEF foi evidenciado por *mapeamento de extremidades 5' do mRNA por extensão do primer* para o gene rocD (GARDAN *et al*., 1995); e rocG foi evidenciado por *mapeamento de extremidades 5' do mRNA por extensão do primer* para o gene rocG (BELITSKY e SONENSHEIN, 1999).

Todos os fatores obtidos foram separados e convertidos individualmente em arquivos texto. As Tabela 4.1 apresenta exemplos de sigmas SigA. O formato do arquivo são os dados separados por tabulações e representam respectivamente as informações: (1) Operon, (2) Gene Regulado, (3) Posição Absoluta, (4) Localização e (5) Sequência de Ligação. Isto se torna necessário para facilitar a entrada de dados dos programas criados para o pré-processamento de dados. Um exemplo do *Motif* do Sigma A pode ser visto na Figura 4.2.

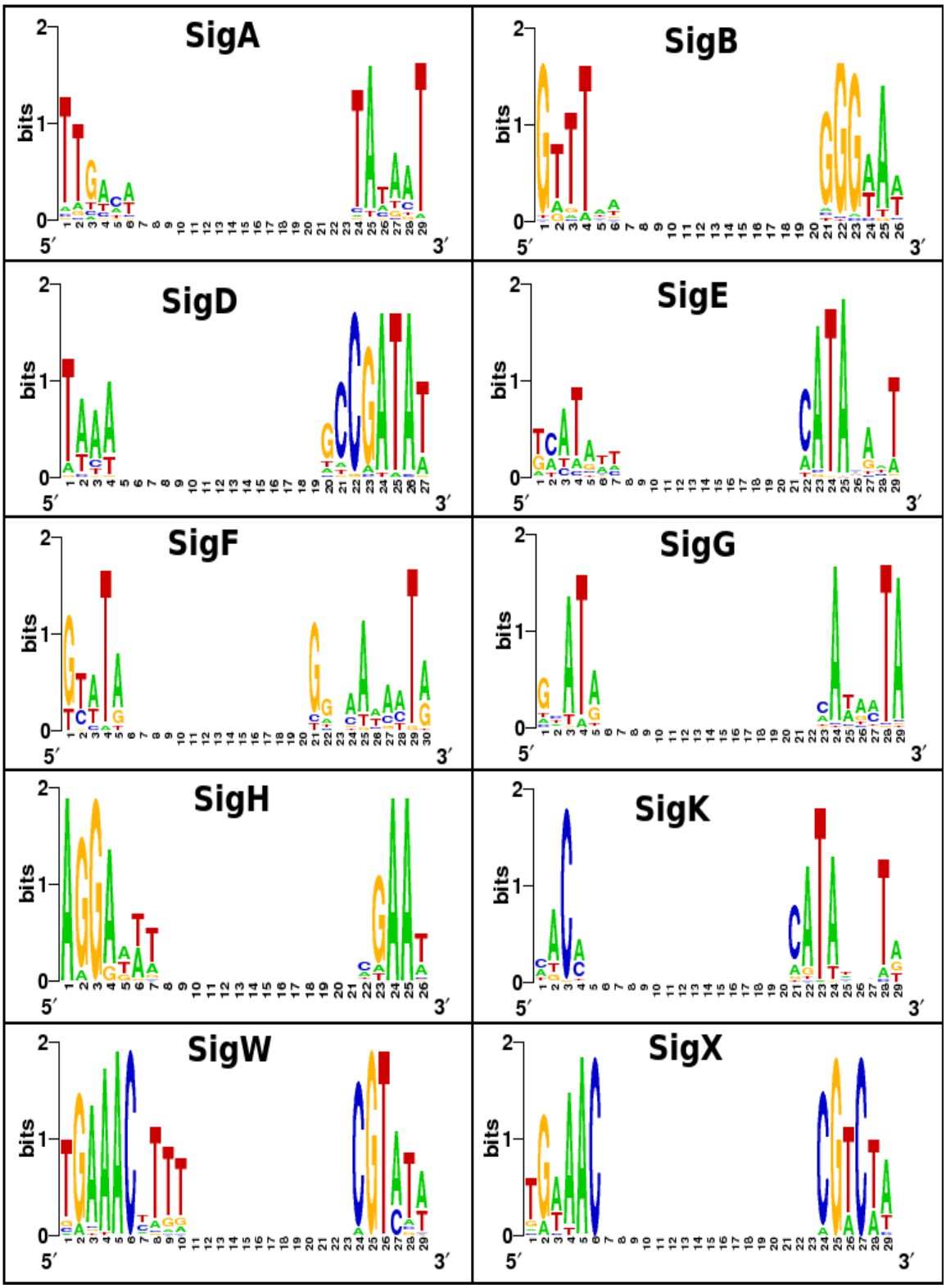


Figura 4.2: Motif dos fatores sigma do domínio sigma70 e sigma54.

* Adaptado de Crooks et al. (2004).

Tabela 4.1: Exemplos de Sigma SigA.

Operon	Gene	Posição Absoluta	Localização	Sequencia de ligação
abnA	abnA	2950133..2950187	-48:+7	CTATTTTTTTGTCTGTACAAATTACAGCAT AGTGACTACAATAAAGGGGATACCG
abrB	abrB	45214..45269	-48:+8	TCTTACAATCAATAGTAAACAAAATGATTG ACGATTATTGGAAACCTTGTTATGCT
abrB	abrB	45200..45257	-50:+8	TAGTAAACAAAATGATTGACGATTATTGGA AACCTTGTTATGCTATGAAGGTAAGGAT
ackA	ackA	3016385..3016434	-44:+6	CAAGTTGACTTGAAAAGCCGACATGACA ATGTTTAAATGGAAAAGTCAGA
acuABC	acuA	3040002..3040068	-49:+18	AAAATATAAACCATGTTGAAAACGCTTTAT AATTTGGTATTCTTAAAGAAGGCATGTATT TTTGATA
adaAB	adaA	203559..203625	-49:+18	ACATTCTAACCACCTTATTTTTTTCTATTTA TGAGGTTATAGTGTAGTTATCAAGAATGCT AAACGG
addBA	addB	1136244..1136311	-48:+20	ACCAATAGATCAGATTGGTCATTTTCGTC AACATTCGATAAAATATAGAGAGATAAAAA GAGAGGGGT
ahpCF	ahpC	4118845..4118891	-41:+6	TATGGCTTGACAAAAAATATATATTAATTAA TAATTCATATATAATT
aprX	aprX	1862727..1862775	-37:+12	ACTTACCAAACCTATCATCTGTTCATAGAG TATACAGAGAACCTTATAA

O arquivo FASTA e os fatores sigma obtidos foram utilizados como entradas no programa escrito em linguagem Python (ROSSUM, 1995) chamado SearchPromoter.py (vide Capítulo 5.1). Este programa foi desenvolvido para procurar por regiões promotoras em genomas completos a partir dos fatores Sigma. O programa busca os promotores no arquivo FASTA utilizando a posição absoluta e, caso não seja encontrado, procura-se pela sequência. Este processo foi realizado para todos os fatores Sigma obtidos. Os promotores encontrados através da saída da execução do programa SearchPromoter.py constituem os exemplos positivos do treinamento. Um exemplo de arquivo obtido por este processo pode ser visto na Figura 4.3.

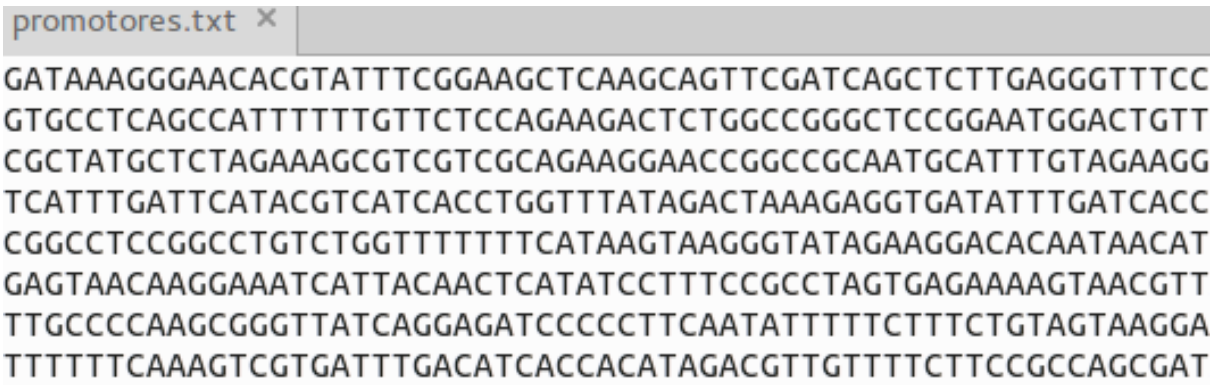


Figura 4.3: Exemplo de Arquivo contendo os Exemplos Positivos de Promotores.

Já os exemplos negativos são sequências geradas pelo programa GenerateNonPromoter.py (vide Apêndice 1). Os exemplos negativos foram gerados através da pesquisa no arquivo FASTA por sequências de nucleotídeos que não são promotores (fazem parte dos exemplos positivos).

Por fim, foram criados programas auxiliares que podem ser utilizados para procurar nos promotores encontrados pelo programa searchPromoter.py por uma sequência de nucleotídeos específica e para verificar se os promotores encontrados tem o mesmo tamanho.

Através do procedimento de pré-processamento de dados foram obtidos 767 promotores para a Bactéria *B. subtilis*, sendo a maioria encontrados pelo fator Sigma SigA (46,07% do total). Os demais sigmas são: SigL (0,77%), SigB (8,62%), SigD (3,86%), SigE (10,68%), SigF (3,86%), SigG (7,85%), SigH (3,08%), SigI (0,90%), SigW (4,37%), SigX (1,93%), SigY (0,12%) e YlaC (0,12%). Os promotores reconhecidos por estes sigmas (767 promotores) são utilizados como entrada nos *softwares* BacPP+ e BacSVM+, descritos respectivamente nas seções 4.2 e 4.1.

Para a realização dos treinamentos de redes neurais que empregam a informação dos nucleotídeos, as sequências foram representadas por uma codificação ortogonal de quatro dígitos binários dados por: A=0100, T=1000, C=0001 e G=0010 (DEMELER e ZHOU, 1991). Um extrato do arquivo contendo os promotores processados pode ser observado na Figura 4.4. Já para os dados serem aceitos pelas máquinas de vetor de suporte, foi utilizado o padrão da biblioteca LibSVM (CHANG e LIN, 2011) (vide Figura 4.5).

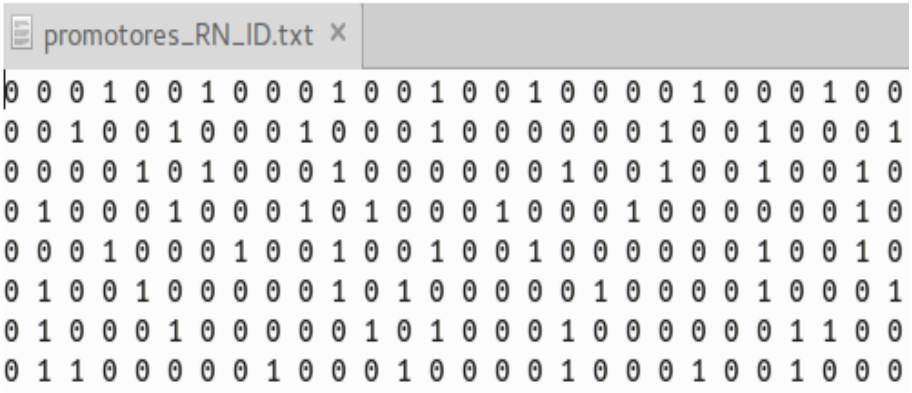


Figura 4.4: Exemplo de Arquivo contendo os Exemplos Positivos de Promotores após processamento para ser utilizados em RN.

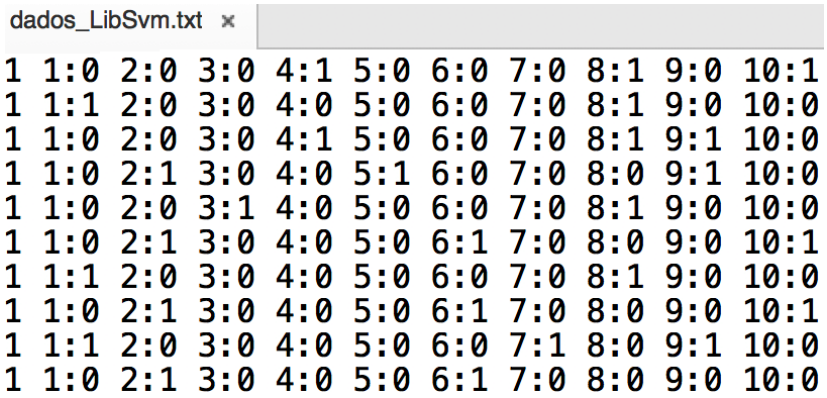


Figura 4.5: Exemplo de Arquivo contendo os Exemplos Positivos de Promotores após processamento para ser utilizados em SVM.

4.2 BACPP+

O BacPP+ foi implementado em linguagem Python e linguagem R para fazer a predição de promotores da bactéria *B. Subtilis* através de uma rede neural artificial. Como pode ser visto na Figura 4.6, o processamento do *software* é dividido em duas fases: (1) preparação de dados (descrito na seção 4.1) e (2) treinamento da rede neural.

O primeiro passo para a simulação da rede neural foi a escolha da arquitetura que melhor caracteriza as regiões promotoras. Ela foi determinada através de simulações de todas as arquiteturas possíveis em um intervalo finito. Foi escolhida a arquitetura que apresentou o menor valor de erro médio quadrático (RMS - *Root Mean Square*) durante o processo de treinamento e teste.

Foram utilizados os seguintes parâmetros nas simulações: sequência de nucleotídeos, o número de neurônios na camada de entrada (obtido pela quantidade de nucleotídeos multiplicada pela quantidade de dígitos da codificação ortogonal) e a quantidade de neurônios na camada oculta (varia entre 1 e 8). Como saída da rede neural, foi usado apenas um neurônio já que o objetivo da rede é binário.

As 32000 simulações para determinação da arquitetura que melhor classifica as sequências foram realizadas no ambiente R. Segundo Jadid e Fairbairn (1996), o algoritmo *Back-propagation* é o mais popular para aprendizado de regras em redes neurais artificiais. Sendo assim, ele foi escolhido para a classificação das sequências no presente trabalho.

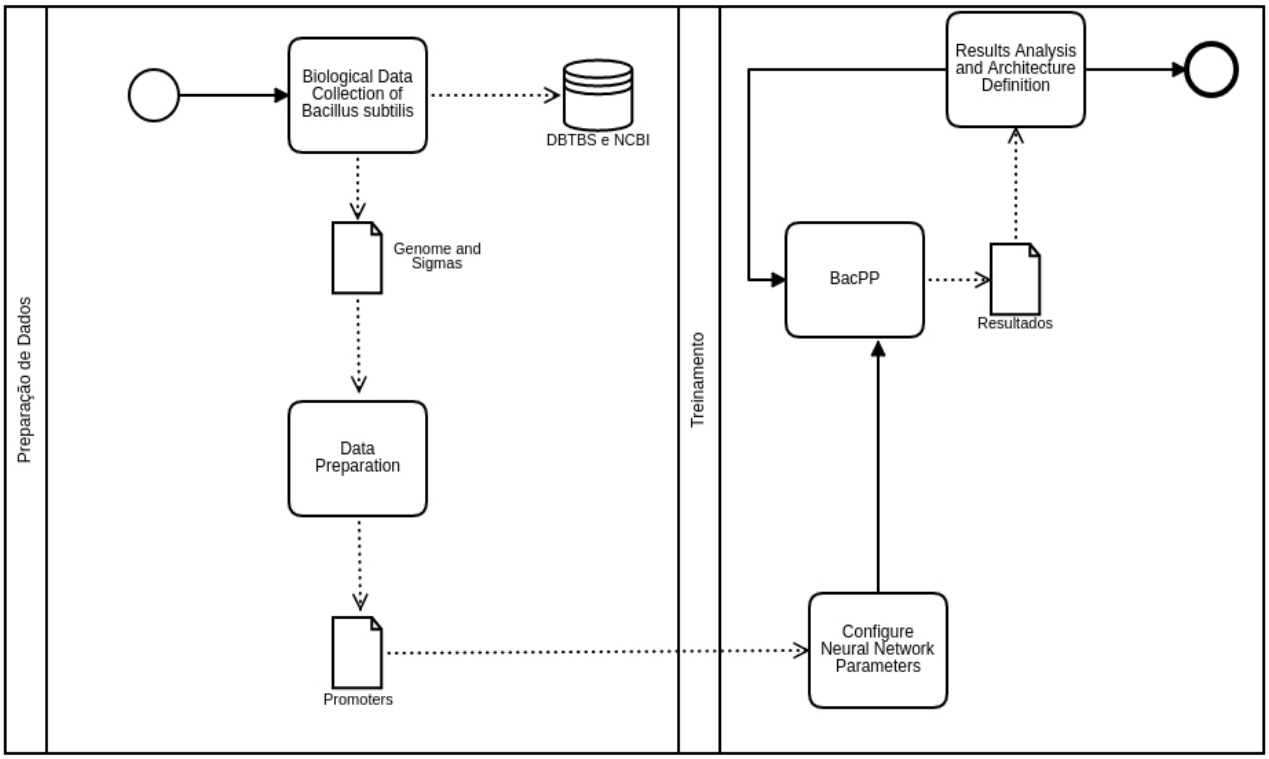


Figura 4.6: Funcionamento do *software* BacPP+.

4.3 BACSVM+

Para a simulação SVM, foi desenvolvido um *software* nas linguagens de programação Java e Python que utiliza internamente a biblioteca LibSVM (CHANG e LIN, 2011) para fazer a predição de promotores da bactéria *B. subtilis*. O *software* se chama BacSVM+.

O funcionamento do BacSVM+ está apresentado na Figura 4.7 e é dividido em três fases: (1) preparação dos dados (aba Data), (2) treinamento SVM (aba LibSVM) e (3) predição de Promotores (aba LibSVM). Baseado nos promotores obtidos durante a primeira fase, é possível definir os parâmetros disponibilizados pela biblioteca LibSVM e simular a classificação dos promotores. Desta forma, a eficiência de todos os modelos definidos pode ser verificada.

O programa permite definir uma serie de parâmetros, como pode ser observado no Capítulo 5.1 (HSU *et al*., 2003). Entre eles, destaca-se a importância dos parâmetros *gamma* (G) e custo (C). O parâmetro C indica em quanto os vetores de suporte são penalizados por erros de predição. Já o parâmetro G permite a configuração correta do *kernel*. Sendo assim, foi criada uma facilidade no BacSVM+ para permitir a pesquisa extensiva entre valores possíveis para estes dois parâmetros, melhorando assim a funcionalidade do *software*.

Foram feitas todas as combinações possíveis entre os tipos disponíveis (C-SVC, NU-SVC, ONE-CLASS, EPSILON-SVR e NU-SVR), os *kernels* disponíveis (LINEAR, POLY, RBF, SIGMOID e PRECOMPUTED) e os parâmetros custo (valor inicial 0.00390625, valor final 65536 e fator multiplicativo de 16) e *gamma* (valor inicial 1.52587890625E-5, valor final 256 e fator multiplicativo de 16), totalizando 17150 simulações. Os valores iniciais e finais foram definidos através de testes de força bruta. Os demais parâmetros foram escolhidos conforme seu respectivo valor padrão da biblioteca LibSVM.

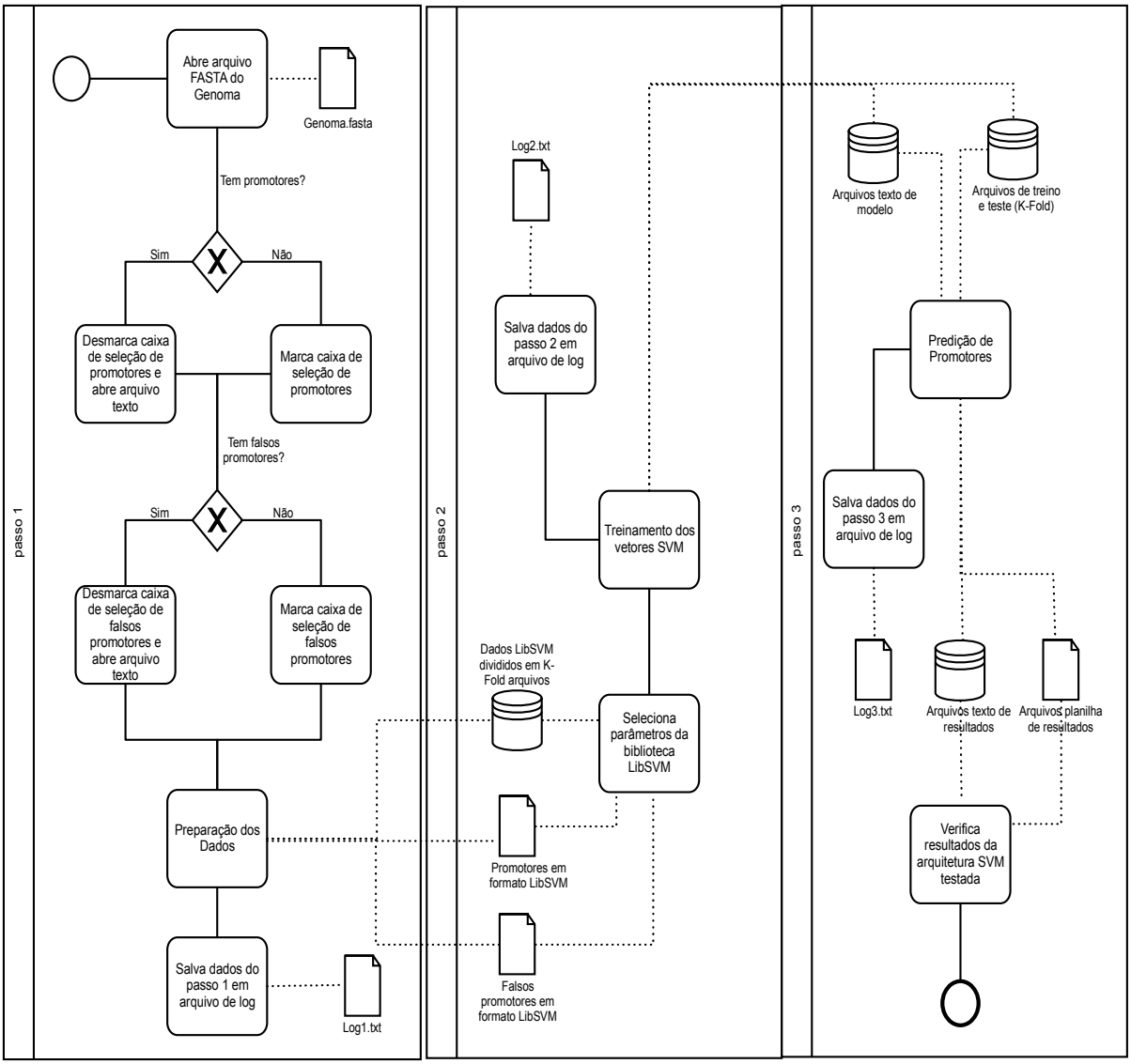


Figura 4.7: Funcionamento do *software* BacSVM+.

5 RESULTADOS

Os resultados obtidos estão organizados na forma de artigos científicos (capítulos I e II) e em um capítulo de discussão, sendo os dois capítulos de artigo a seguir:

- Capítulo I - *Bacillus subtilis promoter sequences used in Gram-positive Promoter Prediction with Support Vector Machine.*
- Capítulo II – *Gram-positive Bacteria Promoter Prediction with Artificial Neural Network.*

5.1 CAPÍTULO I - *BACILLUS SUBTILIS* PROMOTER SEQUENCES USED IN GRAM-POSITIVE BACTERIA PROMOTER PREDICTION WITH SUPPORT VECTOR MACHINE.

Este capítulo apresenta o artigo "*Bacillus subtilis promoter sequences used in Gram-positive Promoter Prediction with Support Vector Machine*" que foi aceito para publicação na revista científica *Data in Brief*. Esta revista apenas publica artigos que descrevem dados científicos que são disponibilizados através de um repositório ou no próprio artigo. O seu foco é na descrição dos dados e não no processo em si. Desta forma, dados que normalmente são colocados em segundo plano podem ser reutilizados por outros pesquisadores. No artigo submetido, são descritos os dados dos promotores obtidos para a bactéria *Bacillus subtilis* e sua validação através do *software BacSVM+* que foi desenvolvido utilizando a biblioteca LibSVM.

Title: *Bacillus subtilis* promoter sequences data set for Promoter Prediction of Gram-positive

Authors: Rafael Vieira Coelho[1], Scheila de Avila e Silva[2], Sergio Echeverrigaray[2], Ana Paula Longaray Delamare[2]

Affiliations: [1]Rio Grande do Sul Federal Institute of Education, Science and Technology (IFRS) – Farroupilha Campus. Farroupilha – RS – Brazil. Tel: +55 54 3260-2400.
[2]Biotechnology Institute – University of Caxias do Sul (UCS). Caxias do Sul – RS – Brazil. Tel: +55 54 3218-2100.

Contact email: rafael.coelho@farroupilha.ifrs.edu.br, {sasilva6,selaguna,apldelam}@ucs.br

Abstract

This paper presents the prediction of *B. subtilis* promoters by a Support Vector Machine system. In literature, there is a lack of information on Gram-positive promoter sequences compared with Gram-negative bacteria. Promoters sequences identification is essential for gene expression. Initially we collected *B. subtilis* genome sequence in NCBI, and promoters recognized by its sigma factors in DBTBS database. It was gathered promoters of 15 factors divided in 2 domains, corresponding to sigma 54 and sigma 70 of Gram-negative bacteria. Based on these data we developed a script in python language to search for promoters in *B. subtilis* genome. After processing the data, we obtained 767 promoter sequences of *B. subtilis*, most recognized by Sigma SigA. To validate the data found, we developed a software called BacSVM+ that receives as input the promoters and returns the best combination of parameters in LibSVM library to predict promoter regions of the bacteria used in the simulation. All data gathered and BacSVM+ software is available for download at http://bacpp.bioinfoucs.com/rafael/Sigmas.zip.

Keywords: promoter sequences, *Bacillus subtilis*, SVM.

Conflict of interest: none.

Specifications Table

Subject area	biology
More specific subject area	promoter sequences
Type of data	text file
How data was acquired	script developed in python language
Data format	raw
Experimental factors	not applicable
Experimental features	It were collected the genome and promoters recognized by *B. subtilis* sigma factors. *The* data (767 promoter sequences) obtained were validated in a software called BacSVM+ which simulates the prediction of promoters in *B. subtilis* bacteria.
Data source location	not applicable
Data accessibility	http://bacpp.bioinfoucs.com/rafael/Sigmas.zip
Related research article	S.A. Silva, S. Echeverrigaray, G. Gerhardt, BacPP: Bacterial promoter prediction—A tool for accurate sigma-factor specific assignment in enterobacteria, Journal of theoretical biology 287 (2011): 92-99.

Value of the Data

- The data obtained can be used in further studies on gene regulation expression. The regulation of gene expression is essential on bacterial metabolic adaptation to environmental changes, allowing bacterial survival and multiplication.
- Most related papers on bacterial promoters are restricted to Gram-negative bacteria, particularly *E. coli*. The promoters of *B. subtilis described* in this paper allows further research in this area.
- There are few data of Gram-positive bacteria promoters in literature. The process described here can be tested by researchers to validate promoters in other bacteria of these type.

Data

The coding region transcription starts when the RNA polymerase (RNAp) enzyme recognizes the promoter region. Promoter regions are conserved DNA sequences that signal and direct the transcription of the adjacent gene or group of genes. Promoters are considered as key factors as they are the initial step of gene expression and part of the transcriptional regulation [13]. For this to occur, the Sigma factor (protein factor component of RNA polymerase) must be present into the enzyme. The sigma factor determines the specificity of the RNA polymerase of a promoter sequence. After RNA polymerase attachment, the sigma factor is release and gene transcription begins generating a RNA molecule [11].

A typical bacterial promoter is located approximately 70bp downstream from the start point of gene transcription. The comparative analysis of several sigma 70 promoters (Gram-negative bacteria) allowed to identify two consensus sequences: (A) one localized at -10bp (5'-TATAAT-3 ') from the transcription start point; and (B) another located at -35bp (5'-TTGAC-3 '). These conserved regions define the affinity of the RNA polymerase complex in

the promoter, and the accuracy of gene expression. The aim of this paper was to study the promoter regions of *B. subtilis* bacteria and make available a promoter data set. Since this bacteria is considered the model agent in laboratory researches due to its easy genetic manipulation [10]. The data obtained consists in 767 promoters separated in fasta files, each one representing a promoter sequence of *B. subtilis* with length of 80 nucleotides.

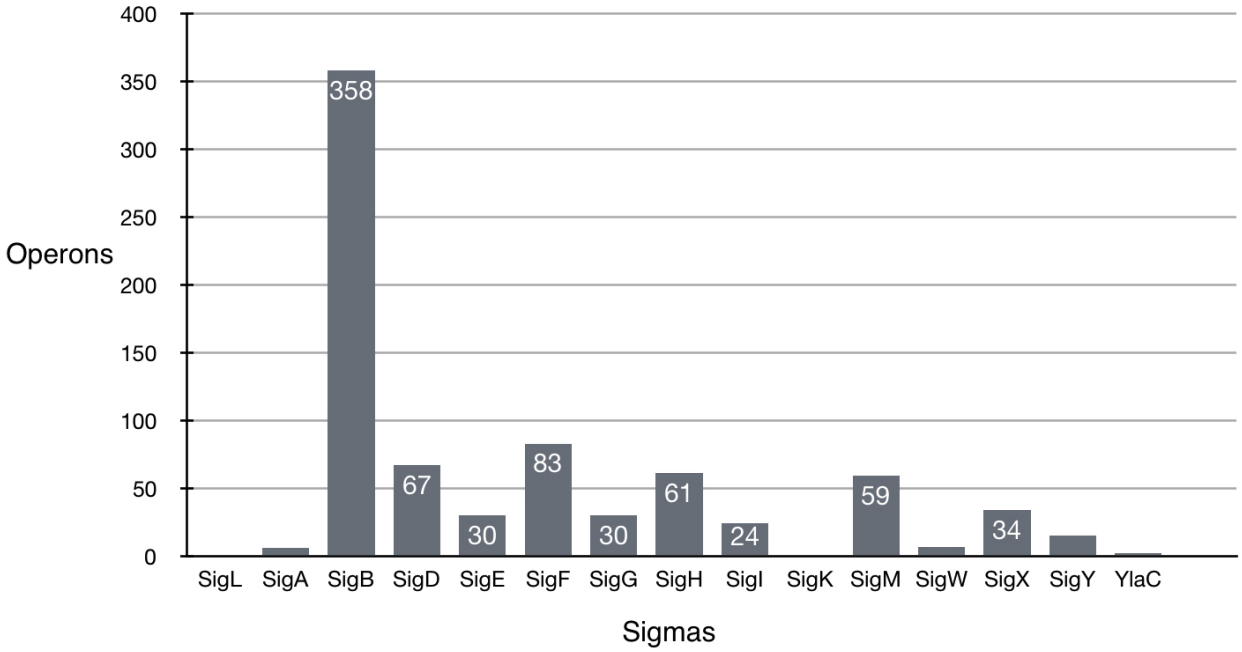
Experimental Design, Materials, and Methods

Initially, it were collected the fasta file containing the genome of *B. subtilis* in NCBI (National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov) and the promoters recognized by its Sigmas in DBTBS (Database of Transcriptional Regulation in *B. subtilis*) database [14], totaling 15 factors divided in 2 domains: Sigma 54 (SigL) and Sigma 70 (SigA and others). They are presented in Table 1 with the following informations: ORF (Open Reading Frame), description and *operons*. SigA stands out due to its high number of operons and promoters identified (46.07%). Fig. 1 shows the proportion of each Sigma operons.

Table 1: Sigma Factors of *B. subtilis* [14].

Domain	ORF	Description	Operons
sigma54	SigL	RNA polymerase Sigma-54 factor (Sigma L)	6
sigma70	SigA	RNA polymerase major Sigma-43 factor (Sigma A). Essential gene.	358
	SigB	RNA Polymerase Sigma-37 Factor (Sigma-B). General stress factor sigma.	67
	SigD	RNA polymerase Sigma-28 factor (Sigma D). Autolytic enzymes; defect in flagellar synthesis.	30
	SigE	RNA polymerase sporulation-specific sigma-29 factor. Processed by SpoIIGA after Tyr-27.	83
	SigF	Synthesized shortly after the onset of sporulation but do not become active until after polar division.	30
	SigG	Control of transcription in the forespore at late stages of sporulation.	61
	SigH	RNA polymerase Sigma-30 Factor. Non-essential sigma factor involved in expression of vegetative and early stationary-phase genes.	24
	SigI	Temperature-sensitive growth in a null mutant; transcription induced by heat shock in rich medium but not in minimal medium; reduced amount of GsiB protein in a sigI mutant under heat shock conditions.	1
	SigK	Formed by a site-specific recombination event which joins the previously separated spoIVCB and spoIIIC genes into a single cistron.	59
	SigM	Essential for growth and survival in high concentrations of salt; expression maximal during exponential growth and increased in high concentrations of salt; activity negatively regulated by YhdL and YhdK.	7
	SigW	ECF-type sigma factor that mediates the transcriptional response to cell wall stress.	34
	SigX	RNA polymerase SigX.	15
	SigY	RNA polymerase ECF(extracytoplasmic function)-type sigma factor	2
	YlaC	RNA polymerase ECF(extracytoplasmic function)-type sigma factor	1

Fig. 1. Number of operons per Sigma factor of *B. subtilis*. X-axes show the Sigmas. Y-axes present the number of Operons.



The data obtained in DBTBS database had the following informations: (1) Operon; (2) Regulated Gene; (3) Absolute Position; (4) Location; and (5) Link Sequence. Due to space restrictions, we present in Table 2 only the data obtained for *Sigma* SigL operons. It is described the operon with gene transcription, its transcription start location, genome position (absolute position), binding sequence (red characters are the exact sequence and black characters are the start sequence) and experimental evidence (scientific work that prove the data).

Table 2: Operons list of SigL sigma [14].

Operon	**Gene**	**Absolute Location**	**Position**	**Biding Sequence**	**Experimental Evidence**
levDEFG-sacC	levD	2762858..2762897	-35:+5	ACTGTGTTGGCACGATCCTTGCATTATATATGGAT**G**TACA	Martin I, et al. [10]; Débarbouille M, et al. [7]
rocABC	rocA	3880546..3880585	-35:+5	AAGAAAATGGCATGATTCTTGCATTTTTTATTCATA**T**G**C**GA	Calogero S, et al. [5]
rocDEF	rocD	4145549..4145588	-35:+5	CTTGATTTGGCACAGAACTTGCATTTATATAAAGG**G**AAAG	Gardan R, et al. [9]
rocG	rocG	3882094..3882135	-34:+8	CAAAAGCTGGTACGGATCTTGCATGATGATAAGGGTGAATCC	Belitsky et al. [2]
ptb-bcd-buk-lpdV-bkdAABB	ptb	2504685..2504726	-34:+8	TAAGAGCTGGCATGGAACTTGCATAATAAAAGGC**G****G**AGTCGA	Débarbouille M, et al. [8]
AcoABCL	AcoA	878940..878979	-34:+6	AAAAGACTGGCACACTTCTTGCATTTATAATGGTGAACCC	Ali NO, et al. [1]

Concerning the experimental evidence of Sigma SigL, acoABCL was evidenced by extremities mapping 5' of mRNA by primer extension for acoA gene and by homology analysis [1]. levDEFG-sacC was evidenced by extremities mapping 5' of mRNA by primer extension for gene levD [10], the use of a reporter gene, and the disruption of the gene binding factor [7]. Finally, the evidences of ptb-bcd-buk-lpdV-bkdAABB, rocABC, rocDEF and rocG came from extremities mapping 5' of mRNA by primer extension for gene ptb [8], for gene rocA [5], for gene rocD [9] and for gene rocG [2], respectively.

The FASTA file and the promoters obtained were used as input in the program written in Python [15] called searchPromoter.ph (source code in Appendix A). This program was developed to look for promoter regions in complete genomes. The program search the promoters in the FASTA file using the absolute position and, if it is not found, searches for the sequence. This process was performed for all data obtained. After processing the data with the script, we obtained 767 promoter regions for *B. subtilis*, mostly related to Sigma SigA. All data obtained is available for download at http://bacpp.bioinfoucs.com/rafael/Sigmas.zip. Fig. 2 shows an example of how the promoter sequence of acuABC Operon from Sigma SigA is selected from *B. subtilis* genome.

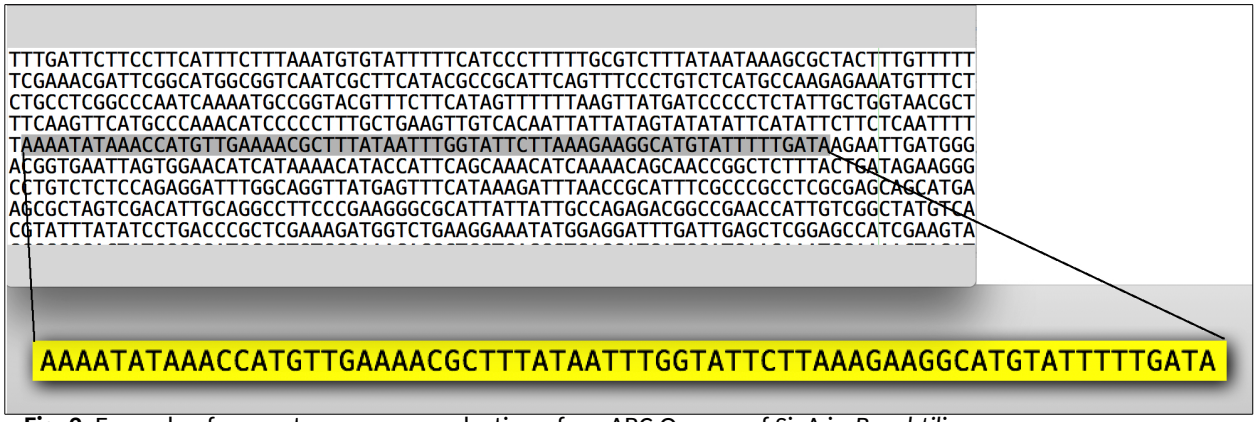


Fig. 2. Example of promoter sequence selection of acuABC Operon of SigA in *B. subtilis* genome.

To validate the data found, was developed a software called BacSVM+ that uses LibSVM library [6] to implement Support Vector Machines [3] for promoter prediction. It receive as input the promoters and returns the best combination of parameters of LibSVM library to predict promoter regions of the bacteria used in the simulation. Its operation is based on the search for the best combination of LibSVM parameters to maximize prediction accuracy. For this, three steps must be followed during its execution: (A) data preparation; (B) support vectors training; and (C) promoter prediction.

The lack of a friendly database could make this first step demanding for users. In this context, the major innovation of BacSVM+ is its data preparation step. If the user does not have the promoters, the program search (with the python script described earlier) in the whole genome for promoters of the respective bacteria. Based on the promoters gathered during the first step, it is possible to define LibSVM parameters and simulate promoter classification.

LibSVM library allows setting a wide range of parameters, see Table 3. Among them, the most important are cost (C) and gamma (G) parameters, where C indicates how much the support vectors are penalized when the prediction is wrong. In other words, when points are placed outside the range of correct classification in the hyperplane. On the other hand, G parameter is a way to configure the kernel. In the case of a Gaussian function, this parameter controls the standard deviation function. BacSVM+ allows an extensive search of C and G parameters by setting a range of possible values.

Table 3: Configuration Parameters of BacSVM+.

Name	Description
gamm	set gamma in kernel function (default is
cost	only in C-SVC, epsilon-SVR, and nu-SVR (default is 1)
svm	C_SVC (default), NU_SVC, ONE_CLASS,
kernel	set type of kernel function
coef0	set coefficient zero in kernel function (default 0)
degre	set degree in kernel function (default 3)
nu	only in nu-SVC, one-class SVM, and nu-SVR (default
cache	cache memory size in MB (default 100)
epsilo	tolerance of termination criterion (default 0.001)
shrinki	whether to use the shrinking heuristics
proba	whether to train a SVC or SVR model for probability
weight	set the parameter C of class i to weight*C, for C-SVC

Finally, in the last step, the user can predict promoter regions and the results can be exported to text or spreadsheet. The architectures performance were evaluated by their accuracy (A), specificity (S) and sensitivity (SN) values, as the following formulas [18].

$$A = (TP + TN) / (TN + TP + FN + FP) \quad (1)$$

$$S = TN / (TN + FP) \quad (2)$$

$$SN = TP / (TP + FN) \quad (3)$$

where:

TP = promoter sequences classified as promoters (true positives); TN = promoter sequences classified as non-promoters (true negatives); FP = promoter sequences not classified as promoter (false positives); FN = promoter sequences classified as non-promoter (false negatives).

All possible combinations between algorithms (C-SVC, NU-SVC, ONE-CLASS, EPSILON-SVR and NU-SVR) and kernels (LINEAR, POLY, RBF, SIGMOID and PRECOMPUTED) available were made. Cost parameter was set between 0.00390625 and 65536, with multiplicative factor of 16. And Gamma parameter was set between 1.52587890625E-5 and 256, with multiplicative factor of 16. The initial and final values were defined through brute-force tests. The other parameters were chosen according to default values of the LibSVM library.

The results obtained during simulations with 767 promoters of B. subtilis are consistent with related works found in literature, thus validating the data gathered. The best combination found is NU-SVC and C-SVC algorithms with RBF kernel, obtaining 93.20% and 95.63% prediction accuracy. BacSVM+ main innovation is in the feature of promoter searching during data preparation step, allowing the user to use the software even if he does not have the promoters and non-promoters examples for simulation. The results can be seen in Table 4.

Table 4: SVM Results.

Type	Kernel	C	G	A (%)	S (%)	SN (%)
C-SVC	SIGMOID	0.062	1.53E-5	82.04	94.17	69.90
C-SVC	SIGMOID	1.0	0.0039	85.44	86.41	84.47
C-SVC	SIGMOID	16.0	2.44E-4	87.86	88.35	87.38
C-SVC	RBF	1.0	2.44E-4	82.04	94.17	69.90
C-SVC	RBF	0.062	0.0039	86.41	98.06	74.76
C-SVC	RBF	16.0	2.44E-4	91.26	92.23	90.29
C-SVC	LINEAR	0.0039	1.52E-5	87.86	88.35	87.38
NU-SVC	SIGMOID	16.0	0.062	57.28	54.37	60.19
NU-SVC	SIGMOID	1.0	0.0039	93.20	94.17	92.23
NU-SVC	RBF	256.0	2.44E-4	95.63	96.12	95.15

* Cost (C), Gamma (G), Accuracy (A), Specificity (S) and Sensibility (SN).

Related works that predicts *B. subtilis* promoter regions with support vector machines were found in literature. Monteiro et al. (2005) [12] did not develop their own software. They used the WEKA software, unlike BacsVM+ which is implemented in Python and Java languages. In contrast to 767 promoters used to validate BacsVM+, 112 promoters of *B. subtilis* were used. The accuracy obtained was lower than BacsVM+, 76%. PePPER was developed as a webserver (it does not require installation and can be accessed over the Internet), but they did not showed results [4]. Finally, TSS SVM [11] analyzes the structural profiles of promoter regions and did not focus specifically on the problem of promoter prediction. They state that the promoter regions are less stable and more rigid than the rest of the genome, but that this is less visible in Gram-positive bacteria like *B. subtilis*.

Acknowledgments

This work was supported by grants from the National Council for Scientific and Technological Development (CNPq). The authors wish to thank University of Caxias do Sul and Federal Institute of Education, Science and Technology for their support in these research.

References

[1] N.O. Ali, J. Bignon, G. Rapoport, M. Debarbouille, Regulation of the acetoin catabolic pathway is controlled by sigma L in Bacillus subtilis, Journal of bacteriology 183.8 (2001): 2497-2504.

[2] B.R. Belitsky, A.L. Sonenshein, An enhancer element located downstream of the major glutamate dehydrogenase gene of Bacillus subtilis, Proceedings of the National Academy of Sciences 96.18 (1999): 10290-10295.

[3] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, Proceedings of the fifth annual workshop on Computational learning theory. ACM, 1992.

[4] A. DE JONG, H. PIETERSMA, M. CORDES, O. KUIPERS, K. JAN. PePPER: a webserver for prediction of prokaryote promoter elements and regulons. BMC Genomics, v.13, n.1, p.1–10, 2012.

[5] S. Calogero, R. Gardan, P. Glaser, J. Schweizer, G. Rapoport, M. Debarbouille, RocR, a novel regulatory protein controlling arginine utilization in Bacillus subtilis, belongs to the

NtrC/NifA family of transcriptional activators, Journal of bacteriology 176.5 (1994): 1234-1241.

[6] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM transactions on intelligent systems and technology (TIST) 2.3 (2011): 27.

[7] M. Débarbouille, I. Martin-Verstraete, A. Klier, G. Rapoport. The transcriptional regulator LevR of Bacillus subtilis has domains homologous to both sigma 54-and phosphotransferase system-dependent regulators. Proceedings of the National Academy of Sciences 88.6 (1991): 2212-2216.

[8] M. Débarbouille, M. Débarbouille, R. Gardan, M. Arnaud, G. Rapoport, Role of BkdR, a Transcriptional Activator of the SigL-Dependent Isoleucine and Valine Degradation Pathway inBacillus subtilis, Journal of bacteriology 181.7 (1999): 2059-2066.

[9] R. Gardan, G. Rapoport, M. Débarbouillé, Expression of therocDEFOperon Involved in Arginine Catabolism inBacillus subtilis, Journal of molecular biology 249.5 (1995): 843-856.

[10] I. Martin, A.M.K. Débarbouille, G. Rapoport, Induction and metabolite regulation of levanase synthesis in Bacillus subtilis, Journal of bacteriology 171.4 (1989): 1885-1892.

[11] P. Meysman, J. Collado-vides, E. Morett, R. Viola, K. Engelen, K. Laukens. Structural Properties of Prokaryotic Promoter Regions Correlate with Functional Features. PloS one, v.9, n.2, p.e88717, 2014.

[12] M. Monteiro, M. De Souto, L. Gonçalves, L. Agnez-lima. Machine Learning Techniques for Predicting Bacillus subtilis Promoters. In: BSB. 2005. p.77–84.

[13] G.G. Perron, L. Whyte, P.J. Turnbaugh, J. Goordial, W.P. Hanage, G. Dantas, M.M. Desai, Functional characterization of bacteria isolated from ancient arctic soil exposes diverse resistance mechanisms to modern antibiotics, PLoS One 10.3 (2015): e0069533.

[14] L. Reys, Dogma Central da Biologia Molecular e Introdução à Bioinformática. Av. L2 Sul Quadra 603 Conjunto C. CEP 70200-630. Brasília-DF: W Educacional Editora e Cursos Ltda (2011).

[15] G. Rossum, Extending and embedding the python interpreter, Report CS-R9527: (1995).

[16] E.F., Ruff, M.T. Record, I. Artsimovitch, Initial events in bacterial transcription initiation, Biomolecules 5.2 (2015): 1035-1062.

[17] N. Sierro, Y. Makita, M. De Hoon, K. Nakai, DBTBS: a database of transcriptional regulation in Bacillus subtilis containing upstream intergenic conservation information, Nucleic acids research 36.suppl_1 (2007): D93-D96.

[18] S.A. Silva, S. Echeverrigaray, G. Gerhardt, BacPP: Bacterial promoter prediction—A tool for accurate sigma-factor specific assignment in enterobacteria, Journal of theoretical biology 287 (2011): 92-99.

Appendix A. searchPromoter.py source code.

```
def openFile(fileName, permission, outputValue):
   try:
      file = open(fileName, permission)
      return file
   except IOError as erro:
      print ('Error:' + str(erro))
def recordInFile(fileName, text, exitOutputValue):
   file = openFile(os.curdir + fileName, 'w', exitOutputValue)
   file.write(text)
   file.flush()
   file.close()

def getLinesFromFile(title, fileName, splitFile):
   print(title, fileName)
   file = openFile(os.curdir + fileName, 'r', -1)
   if (splitFile == 1):
      lines = file.read().split('\n')
   else:
      lines = file.read()
   file.close()
   return lines

def listFilesFromFolder(folderName, text):
   files = os.listdir(os.curdir + folderName)
   numberFiles = len(files)
   return files

def recordPromoterInFile(sigmaFormated, sigma, folderName, name):
   if ('Promotores' not in os.listdir(os.curdir)):
      os.mkdir(os.curdir + "/Promoters/")
   if (folderName not in os.listdir(os.curdir + "/Promoters/")):
      os.mkdir(os.curdir + "/Promoters/" + folderName)
   path = os.path.expanduser(os.curdir + "/Promoters/" + folderName)
   try:
      if (len(sigma) == 80):
         filePromoter = open(path + '/' + name + '.promoter.fasta', 'w')
         print(sigma + ' - ' + str(len(sigma)))
         filePromoter.write(sigma)
         filePromoter.flush()
         filePromoter.close()
         return sigma
   except IOError:
      print('Error: output file promoter ' + name + '.txt not created.')

def savePromoter(lines, sigma, folderName, index, name, isAbsolute):
   posBefore = int((sigma.split('\t')[3]).split(':')[0])
   posAfter = int((sigma.split('\t')[3]).split(':')[1])
   sigmaFormated = '>' + re.sub("\t", "|", sigma)
 return recordPromoter(lines, posBefore, posAfter, sigmaFormated,
 folderName, index, name, isAbsolute)

def recordPromoterNone(lines, posBefore, sigmaSize, posAfter,
   sigmaFormated, folderName, name, i):
```

```
    before = lines[(i - posBefore):(i + 1)]
    midle = lines[(i + 1):(i + 1 + sigmaSize)]
    after = lines[(i + 1 + sigmaSize):(i + 1 + sigmaSize + posAfter)]
    if (before.__contains__('\n')):
        before = before.split('\n')[0] + before.split('\n')[1]
    if (midle.__contains__('\n')):
        midle = midle.split('\n')[0] + midle.split('\n')[1]
    if (after.__contains__('\n')):
        after = after.split('\n')[0] + after.split('\n')[1]
    sigma = before + midle +  after
    if (len(sigma) == 79):
        after = lines[(i + 1 + sigmaSize):(i + 2 + sigmaSize + posAfter)]
    sigma = before + midle +  after
    if (sigma.__contains__('\n')):
        sigma = sigma.split('\n')[0] + sigma.split('\n')[1]
    return recordPromoterInFile(sigmaFormated, sigma, folderName, name)

def savePromoterNone(lines, sigmas, folderName, name, i):
    sigmaSize = len(sigmas.split('\t')[4])
    position = ((80 - sigmaSize) / 2)
    posBefore = int(round(position, 0))
    posAfter = int(position)
    sigmaFormated = '>' + re.sub("\t", "|", sigmas)
    total = posBefore + sigmaSize + posAfter
    return recordPromoterNone(lines, posBefore, sigmaSize,
    posAfter, sigmaFormated, folderName, name, i)

def recordPromoter(lines, posBefore, posAfter, sigmaFormated,
    folderName, i, name, isAbsolute):
    if (isAbsolute == 1):
        sigmaSize = posAfter - posBefore
    else:
        sigmaSize = len(sigmaFormated.split("|")[4])
    posAfter =  19 - posAfter
    posBefore  =  61 + posBefore
    if (posAfter < 0):
        posBefore += posAfter
    if (posBefore < 0):
        posAfter += posBefore + 1
    if (posBefore >= 0):
        before = lines[(i - posBefore):(i + 1)]
        midle = lines[(i + 1):(i + 1 + sigmaSize)]
        after = lines[(i + 1 + sigmaSize):(i + 1 + sigmaSize + posAfter)]
        if (midle.__contains__('\n')):
            midle = midle.split('\n')[0] + midle.split('\n')[1]
        if (before.__contains__('\n')):
            before = before.split('\n')[0] + before.split('\n')[1]
        if (after.__contains__('\n')):
            after = after.split('\n')[0] + after.split('\n')[1]
        sigma = before + midle + after
    else:
        sigma = lines[i:(i + 81)]
    if (sigma.__contains__('\n')):
      sigma = sigma.split('\n')[0] + sigma.split('\n')[1]
    return recordPromoterInFile(sigmaFormated, sigma, folderName, name)

sigmaFiles = listFilesFromFolder('/Sigmas/', 'Number of Sigma Files: ')
```

```
genomeFiles = listFilesFromFolder('/Genomes/', 'Number of Genome Files: ')
for genomeFile in genomeFiles:
   genomLines = getLinesFromFile('Genome ', "/Genomes/" + genomeFile, 0)
   print('_'*100)
   number_genome = 1
   number_promoters = 1
   finalArchive = ''"
   for sigmaFile in sigmaFiles:
      sigmas = getLinesFromFile('File: ', '/Sigmas/' + sigmaFile, 1)
      print('Number of Sigmas: ', len(sigmas))
      index = 0
      while (index < len(sigmas)):
            try:
               sigma = sigmas[index].split('\t')[4]
               if (genomLines.find(sigma) != -1):
                  i = genomLines.index(sigma)
                  try:
                     sigmaOk = savePromoter(genomLines, sigmas[index], 'GENOME_'
                     + str(number_genome), i, str(number_promoters), 0)
                     if (sigmaOk != ''):
                        print(str(number_promoters) + ' - ' + sigmas[index] + ' (SIG)')
                        number_promoters += 1
                        finalArchive += sigmaOk + '\n'
                  except ValueError:
                     positions = sigmas[index].split('\t')[2]
                     if (positions != 'ND'):
                        sigmaOk = savePromoterNone(genomLines, sigmas[index], 'GENOME_'
                        + str(number_genome), str(number_promoters), i)
                        if (sigmaOk != ''):
                           print(str(number_promoters) + ' - ' + sigmas[index] + ' (None)')
                           number_promoters += 1
                           finalArchive += sigmaOk + '\n'
               else:
                  positions = sigmas[index].split('\t')[2]
                  if (positions != 'ND'):
                     sigmaFormated = '>' + re.sub("\t", "|", sigmas[index])
                     try:
                        posAbs1 = int((sigmas[index].split('\t')[2]).split('..')[0])
                       sigmaOk = savePromoter(genomLines, sigmas[index], 'GENOME_'
                       + str(number_genome), posAbs1, str(number_promoters), 1)
                        if (sigmaOk != ''):
                           print(str(number_promoters) + ' - ' + sigmaFormated + ' (ABS)')
                           number_promoters += 1
                           finalArchive += sigmaOk + '\n'
                     except ValueError:
                           sigmaOk = savePromoterNone(genomLines, sigmas[index], 'GENOME_'
                           + str(number_genome), str(number_promoters), i)
                     if (sigmaOk != ''):
                           print(str(number_promoters) + ' - ' + sigmaFormated + ' (None)')
                           number_promoters += 1
                           finalArchive += sigmaOk + '\n'
                  else:
                     print('ERROR: ' + sigmas[index])
                     ads=1
            except IndexError:
               print(sigmas[index])
            index+=1
```

5.2 CAPÍTULO II - GRAM-POSITIVE BACTERIA PROMOTER PREDICTION WITH ARTIFICIAL NEURAL NETWORK

Este capítulo apresenta o artigo “Gram-positive Bacteria Promoter Prediction with Artificial Neural Network” que será submetido para publicação na revista científica Applied Microbiology and Biotechnology. Esta revista concentra-se em células procarióticas, células eucarióticas, enzimas, proteínas, genética aplicada e biotecnologia molecular, genômica, proteômica, fisiologia microbiana, celular aplicada, biotecnologia ambiental, etc. O periódico recebe artigos completos e revisões de novos produtos, processos e tecnologias emergentes. Neste artigo, são descritos os resultados obtidos com as simulações de redes neurais para a predição de promotores da bacteria *B. subtilis.*

Gram-positive Bacteria Promoter Prediction with Artificial Neural Network

Rafael Vieira Coelho **Scheila de Avila e Silva**
Sergio Echeverrigaray **Ana Paula Longaray Delamare**

Abstract

The transcription consists in reading the information contained in the DNA to generate the corresponding RNA messenger. To start these process for a gene, the RNA polymerase enzyme recognizes a promoter region, regulating gene expression. The literature proposes several computational methods for the prediction and recognition of promoter sequences, but these problem has not been satisfactorily addressed and the most of the works focuses on Gram-negative bacteria promoters due to the amount of data available. Therefore, the promoter identification in prokaryotes remains a challenge in the bioinformatics field. This paper uses the same methodology of the BacPP application but for Gram-positives bacteria. As input of the artificial neural network, it was used 767 *Bacillus subtilis* promoter regions, each sequence with 80 length. Simulations were performed to find the architecture that makes the best prediction. The results are consistent and higher than those found in the literature, obtaining 98.57% and 97.69% prediction efficiency with a multilayer perceptron neural network (MLP) with 5 and 7 neurons in the hidden layer and 1 neuron in the output layer. To statistically validate the results, it was used the k-cross validation technique with 5-fold.

Keywords Promoter Prediction *Bacillus subtilis* Neural Network

1 Introduction

Bacteria are single-celled organisms that can be separated into two large groups based on the structure of their cell walls (Gram differential staining technique). Therefore, two bacterial phylogenetic groups with important genetic and metabolic differences are distinguished, including their promoter sequences: (A) Gram-negative bacteria; (B) Gram-positive bacteria[1].

Regardless of the bacteria type, it must express the information contained in its coding regions. This process involves the production of RNA molecules transcribed from the DNA template. The transcription is the first step to gene expression whose purpose is to decode

the information of a particular gene into protein. To initiate the transcription of a coding region, the RNA polymerase (RNAp) enzyme must recognize the promoter region.

The promoters are specific sections of DNA sequences recognized by specific proteins, unlike other parts of the DNA that are transcribed and translated. A typical bacterial promoter is located approximately 70bp before the gene transcription start point. Through the comparative analysis of several bacterial promoters, two consensus sequences were defined: (A) one located at $10bp^{1}5^{0}$ T AT AAT 3^{0o} from the transcription start point; and (B) one located at $35bp^{1}5^{0}$ TTG AC 3^{0o}. These regions define the affinity of the RNA polymerase by the promoter[2].

The promoters determines the gene, the moment and the quantity that will be expressed by the cell, influencing the regulation of genes[1]. The study of gene regulation remains a relevant subject in the scientific community. In order for a gene to be transcribed, specific proteins called sigma factors must locate a sequence in the DNA called promoter region. After these process, the Sigma factor must be released for the transcript to continue[3]. The identification of promoter regions is a challenge of the post-genomic because they can act as inhibitors or activators of genes.

However, there are some trammels that hamper the development of this research area. The promoter sequence is short and not completely conserved, thus making it difficult to locate and can generate a high number of false positives. In addition, molecular techniques for the identification of promoters are expensive, thus indicating that in silico approaches are more attractive for these problem.

Several computational methods that aim to predict promoter regions were found in literature: (1) probabilistic analysis; (2) pattern recognition; and (3) machine learning. Among these, the solutions that were most successful are machine learning solutions, such as artificial neural networks (ANN) and support vector machines (SVM). Artificial neural networks were originally developed with the purpose of modeling the information processing and learning of the brain[4].

According to Rebouças(2011)[5], model structures based on neural networks are generalizations of linear models that are appropriate for the identification of non linear systems. They consist of layers of processing units (artificial neurons) with connections (associated with weights) between them. These are distributed parallel structures, consisting of simple processing units which compute certain mathematical functions from incoming inputs.

The training of a ANN aims at assigning values to a set of weights (randomly initialized), applying the input data to the network and verifying how it responds to certain sets of weights. If the performance is unsatisfactory, the weights must be modified by the

algorithm of the architecture and the procedure must be restarted. This procedure is repeated until the weights provide the network with the ability to represent the problem in hand, usually represented by the range of a pre-specified shutdown criterion [6]. Once trained, the weights are fixed and the network can be used as a model, estimating outputs from an input data set.

In the majority of computational solutions found, only Gram-negative bacteria were used due to the amount of data available. Based on this gap, this paper aims to predict and recognize promoters in intergenic regions of Gram-positive *Bacillus subtilis* bacteria through the application of an artificial neural network.

2 Material and Methods

2.1 Input Data

According to Masayuki et al. (1991)[7], Gram-positive and Gram-negative bacterial promoters have similarities. They analyzed the similarities between *Bacillus subtilis* (Gram-positive) and *Escherichia coli* (Gram-negative) bacteria. However, they also claim that some promoters of *Escherichia coli* (for example, the promoter of *LacUV5* gene) can not be transcribed by the RNA polymerase of *Bacillus subtilis*. In addition, biochemical studies demonstrate that *Bacillus subtilis* is less tolerant in the deviation of the consensus 12pb[8]. Promoters transcribed by the factors ^{A} or ^{70} have the following similarities:

1. the sequences are conserved in the hexamers 35 and 10;
2. the distance between the hexamers;
3. position of transcription start site.

There is no consolidated database with the genetic information of Gram-positive bacteria. Promoter regions, intergenic regions and data related to promoter characteristics were obtained from biological databases. The public databases used were DBTBS (Database of Transcriptional Regulation in *Bacillus subtilis*) and NCBI (National Center for Biotechnology Information).

Initially, the FASTA file containing the bacterial genome of *Bacillus subtilis* was obtained from NCBI (gi|225184640|emb|AL009126.3|: 1 4215606 *Bacillus subtilis* str. 168 complete genome). Sigma factors of this type of bacteria were obtained in the DBTBS database, totaling 15 factors divided into 2 domains: Sigma 54 (SigL); (2) Sigma 70 (other Sigma factors). This was a rather lengthy process due to the lack of available Gram-positive bacteria data. The factors gathered are as described in Table 1.

Table 1 Sigma factors of *Bacillus subtilis*.

Domain	ORF	Description	Operons
Sigma54	SigL	RNA polymerase sigma-54 factor (sigma-L)	6
Sigma70	SigA	RNA polymerase sigma factor RpoD	386
	SigB	RNA polymerase sigma factor SigB	67
	SigD	RNA polymerase sigma factor SigD	30
	SigE	sporulation sigma factor SigE	85
	SigF	sporulation sigma factor SigF	30
	SigG	sporulation sigma factor SigG	62
	SigH	RNA polymerase factor sigma-70	26
	SigI	putative RNA polymerase sigma factor SigI	1
	SigK	late mother cell-specific gene expression (stage IV sporulation)	60
	SigM	RNA polymerase sigma factor SigM	7
	SigW	RNA polymerase sigma factor SigW	34
	SigX	RNA polymerase sigma factor SigX	15
	SigY	RNA polymerase sigma factor SigY	2
	YlaC	RNA polymerase ECF-type sigma factor	1

All factors obtained were in the worksheet format and separated by Sigma factor. All were converted individually into text files. The file format is the data separated by tabs and represents the information respectively: (1) Operon; (2) Regulated Gene; (3) Absolute Position; (4) Location; and (5) Link Sequence. This is necessary to facilitate data entry of programs created for data preprocessing.

The FASTA file and the sigma factors were used as inputs in a script written in Python language (ROSSUM, 1995). This program was developed to search for promoter regions in complete genomes from Sigma factors. It searches the promoters in the FASTA file using the absolute position and, if it is not found, searches for the sequence. This process was performed for all the Sigma factors gathered, thus forming the set of positive examples for the training. The set of negative examples were created using random sequences that are not promoter sequences.

To use the information of the nucleotides, the sequences were represented by an orthogonal coding of four binary digits given by: A = 0100, T = 1000, C = 0001 and G = 0010 [9]. Through the data preprocessing procedure 767 promoters were obtained for the *Bacillus subtilis* bacterium, most of which were found through the SigA factor.

2.2 Neural Network Simulation

This paper uses the ANN methodology of BacPP software as can be seen in Figure 1. It is divided into two steps: (1) data preparation (described in section 2.1); and (2) neural network training [10].

The training of neural networks was performed through an algorithm that is based on the rule of learning by error correction. It is called backpropagation algorithm (backpropagation). This is the most widespread algorithm for multilayer perceptron (MLP) [11] supervised network. It has high potential for functional adjustments due to its approximation capabilities [12].

The propagation of the inputs sends the functional signal through the network until the generation of an output, keeping the synapse weights fixed. In Back-propagation, the error is compared to a target producing an error signal. This signal propagates from the output to input, and the weights are adjusted in a way to minimize the error.

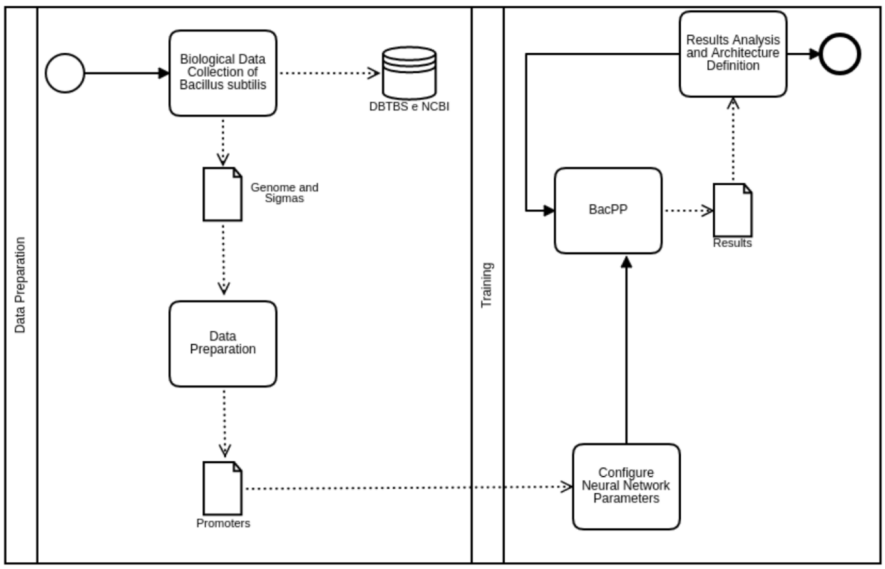


Fig. 1 ANN Methodology based on BacPP software.

At each step t of training, the error signal of each output neuron j is calculated. For a ANN with i output neurons, the cost function is defined as the sum of the i error signals (Equation 1).

$$E(t) = \frac{1}{2} \times \sum_{j=1}^{i} e_j^2(t) \qquad (1)$$

Although it is not feasible to calculate the errors of the hidden neurons directly, they are also responsible for the total error calculated as output of the network. Thus, the error signal must be back propagated, making it possible to adjust the weights in the hidden layers. It is an estimate recursively determined in terms of the error signals of all neurons in the posterior layer.

Several models with different characteristics were constructed. The one that achieves the best performance in training is defined as the best architecture for the desired application. However, it should be borne in mind that in multi-layered net-works, the implemented functions become increasingly complex as a signal traverses its layers.

The first step in neural network simulation was the choice of the architecture that best characterizes the promoter regions. It was determined by simulations of all possible architectures in a finite interval. The architecture that presented the lowest root mean square error (RMS) was chosen during the training and testing process.

The following parameters were used in the simulations: nucleotide sequence, the number of neurons in the input layer (obtained by the number of nucleotides multiplied by the number of digits of the orthogonal coding) and the number of neurons in the hidden layer (ranges from 1 to 8). As output from the neural network, only one neuron was used since the purpose of the network is binary.

The simulations for determining the architecture that best classifies the sequences were performed in the R environment. The algorithm chosen for the classification of sequences was Back-propagation. In order to validate the data in a statistical way, we opted for the k-fold cross-validation methodology. It was selected 5-Fold due to the amount of data gathered and simulations results.

3 Results and Discussion

The first ANN applications in promoter prediction obtained a high accuracy, but the number of false positives was also high[9]. Another approach was presented by Mahadevan and Ghosh (1994)[13] through the combination of two ANNs for the identification of *Echerichia coli* promoters. All promoters in this work had spaced between 15 and 21 nucleotides between the characteristic hexamers. The first ANN predicted the consensus hexamers, while the second was designated for recognition of the entire sequence (65 nucleotides), with the space between the hexamers variable. Once the information of the sequence had been used, dependencies occurred between the bases at various positions. This reflected in poor training.

On the other hand, Pedersen and Engelbrecht (1995)[14] predicted the transcription start site (TSS), measuring the content of the information and identifying new characteristic signals correlated with the start site. For this, two different coding schemes were used, one with windows 1 to 51 nucleotides and the other with a win-dow of 65 nucleotides. An interesting idea in this work was to measure the relative information content of the input data by using ANN's ability to learn correctly as assessed by the correlation coefficient of the maximum test.

Oppon (2000)[15] did a test on this system from the set consisting of 31 sequences of 75 bases, 5 being promoter regions and 26 *E. coli* coding regions. With a threshold of 6, Neural Networks Promoter Prediction (NNPP) classify as promoter sequence with 60% and as non promoter with 50% efficiency. One probable explanation for this poor performance, in addition to the low number of samples, is the specificity of the system which was developed specifically for an organism. Burden et al.[16] improved this system by incorporating information of the distance between the TSS and the translation start site (TLS). With a data set of 771 promoters, they improved the prediction and reduced the number of false positives.

Cotik et al. (2005)[17] created a hybrid methodology, called Hybrid Promoter Analysis Methodology (HPAM) that combines neural networks, Fuzzy logic and genetic algorithms. Their operation may be described by the following steps: (1) decomposition through the neural network into modules of the motifs of the bind-ing regions concerning non-specific DNA sequences; (2) Fuzzy logic inferences for association between the modules identified by the network; and (3) use the pattern recognition method that uses genetic algorithms called Multi-objective Scatter Search (MOSS) to find the most representative motifs.

Still, Santos et al. (2008)[18] proposed a learning method for a Bayesian classifier for recognition of prokaryotic promoters. They implemented in Java and Matlab a classifier model based on the Naive Bayes method to identify promoters recognized by the Sigma70 factor. The bacterium used was *Echerichia coli* and the method showed efficiency of 90%, but the amount of sequences was low, only 99%. Therefore, the method needs a more rigid validation. On te other hand, Bland et al. (2010) [19] used ANN that used Stress-Induced Duplex Destabilization (SIDD) data profiles. That is, networks that take into account structural properties for the prediction of a promoter region.

In addition, De Avila e Silva et al. (2011) [10] developed a software called BacPP (Bacterial Promoter Prediction) whose objective is the use of artificial neural networks in the prediction, characterization and recognition of promoters of Gram-negative bacteria. The results obtained with BacPP were satisfactory and com-parable with the literature. Thus, we selected BacPP to expand their methodology, allowing Gram-positive bacteria to be used.

Meng et al. (2013)[20] developed a library to improve the efficiency of neural network training through the quantitative analysis of possible mutation points and conserved sequences. They used the Neural Network Toolbox provided in Matlab and configured the network with three layers, one being hidden. They used 896 neurons in the input layer and 1 neuron in the output layer. However, its validation was done only with 100 sequences, which impairs their performance evaluation due to small sampling.

Finally, a software called CNNProm was developed and obtained high accuracy (*Escherichia coli* 84% and *Bacillus subtilis* 86%), but only used 70 factors[21]. In addition, they did not use any information on specific promoter characteristics. Kumar and Bansal (2017)[22] focused their efforts on the analysis of structural characteristics (low stability, low flexibility and high curvature) of *Escherichia Coli* promoters. But their results indicate a probable failure in network learning (results close to 100%).

The performance of the neural network in this paper were evaluated through the values of accuracy (A - Equation 2), specificity (S - Equation 3) and sensitivity (SN - Equation 4), where T P, T N, FP and F N represents respectively, true positives, true negatives, false positives and false negatives.

$$A = \frac{TP + TN}{TN + TP + FN + FP} \quad (2)$$

$$S = \frac{TN}{TN + FP} \quad (3)$$

$$SN = \frac{TP}{TP + FN} \quad (4)$$

ANN simulations were carried out, totaling 32000 simulations with the 767 promoter regions collected. The results are consistent with those found in the literature. As a highlight, we obtained 98.57% and 97.69% efficiency in its prediction with ANN. The multilayer perceptron network was configured with 5 and 7 neurons in the hidden layer and 1 neuron in the output layer. After 7 neurons, the network showed signals of overtraining. The table 2 shows the best results obtained based on the efficiency of each configured network.

Moreover, the only works found that used *Bacillus subtilis* as study subject were NANS[15] and CNNProm[21]. NANS (Nureka Artificial Neural Systems) used frequency distribution analysis in a Hidden Markov Model. They used a low number of samples (10 to 50) of non-normalized sequences (40pb to 75pb), obtained low accuracy (50% to 60%) and did not analyzed false negatives in their system. In the other hand, CNNProm created a web-server that uses only 70 and does not use any information about characteristics of specific promoters. Its results showed high accuracy (*Escherichia coli* 84% e *Bacillus subtilis* 86%) but this paper had even better accuracy.

Table 2 Best Results obtained in ANN solution

Network	Period	Hidden	TP	FN	FP	TN	A	SN	S
1	50	1	36	2	2	20	92.76	94.76	89.38
1	75	2	36	2	4	19	91.44	95.81	84.07
1	50	3	37	1	1	21	97.03	97.90	95.57
1	25	5	37	1	1	21	96.71	98.42	93.80
1	40	6	37	1	4	19	92.76	97.90	84.07
1	25	7	37	1	1	22	97.69	97.90	97.34
2	55	1	73	3	3	42	94.73	95.54	93.36
2	55	2	72	4	4	41	93.42	94.50	91.59
2	35	3	75	1	1	44	98.35	98.95	97.34
2	25	4	74	2	0	45	97.86	96.85	99.55
2	25	5	75	1	2	43	98.02	98.95	96.46
2	40	6	75	1	4	41	96.38	98.95	92.03
3	55	1	111	4	3	64	96.05	96.50	95.28
3	75	2	81	32	4	64	80.04	71.37	94.69
3	45	3	113	1	2	65	98.02	98.77	96.75
3	50	5	113	1	1	66	98.57	98.95	97.93
3	35	6	113	2	5	62	96.16	98.25	92.62
4	50	1	119	3	4 3	87	84.62	77.74	96.23
4	40	3	147	5	3	87	96.71	96.46	97.12
4	55	4	121	3	10	90	86.75	79.18	99.55
4	40	5	151	2	2	88	98.10	98.56	97.34
4	40	6	151	2	4	86	97.86	98.95	96.01
5	35	1	156	35	6	107	86.38	81.46	94.69
5	40	3	184	7	6	107	95.65	96.23	94.69
5	55	4	157	3	40	113	88.61	82.09	99.64
5	35	5	188	3	4	109	97.82	98.42	96.81
5	30	6	184	7	9	104	94.86	96.43	92.21

4 Conclusions

Artificial neural networks and support vector machines allows the prediction of bacterial promoter regions in a computational manner. Previously works found in literature focused their efforts primarily on gram-negative bacteria, more specifically *Echerichia coli*. Differently, this paper study *Bacillus subtilis*. For this, a tool called BacPP + was implemented. It's main goal is the adjustment of neural network parameters for the best possible prediction of promoter regions in gram-positive bacteria. This software allows the classification of a certain sequence as a promoter and the probability of being recognized by the different sigma factors of this bacterium.

Initially, the FASTA file containing the genome in the NCBI and the promoters recognized by the transcription factors of *Bacillus subtilis* were obtained, totaling 15 factors divided into 2 domains: (1) Sigma 54 (SigL); (2) Sigma 70 (other transcription factors). A total of 767 promoters were obtained for *Bacillus subtilis*, most of which were found by the SigA transcription factor.

After data gathering and preprocessing, 32000 simulations were performed with the artificial neural network. The results obtained are consistent with those found in literature. The prediction with neural networks through a Multilayer Perceptron with 5 and 7 neurons in the hidden layer and 1 neuron in the output layer obtained an efficiency of 98.57 % and 97.69 %. With 8 neurons in the hidden layer, the network showed overtraining behavior. For statistical validation of the results, the 5-fold k-cross validation technique was used. Finally, comparing the results of the solution with those found in literature, they are satisfactory and competitive.

Finally, there are structural characteristics in promoters that can not be ruled out. They may aid in the differentiation of non-promoter regions. Therefore, it is indicated as a possible future work the use of structural information (e.g. stability, curvature and deformability) of promoter sequences to improve the performance of the developed solution.

Acknowledgements

We are grateful to Universidade de Caxias do Sul (UCS) and Capes Foundation for financial support.

References

1. E. Ruff, T. Record, I. Artsimovitch, Biomolecules **5**(2), 1035 (2015). DOI 10.3390/biom5021035. URL http://www.mdpi.com/2218-273X/5/2/1035

2. D. Nelson, M. Cox, *Princípios de Bioquímica de Lehninger* (Artmed, 2011)

3. L. Reys, J. Macedo, J.C.P. Damalio, *Dogma Central da Biologia Molecular e Introdução a Bioinformática* (W Educacional Editora e Cursos Ltda, Av. L2 Sul Quadra 603 Conjunto C. CEP 70200-630. Brasília-DF, 2011)

4. A. Braga, A. de Leon, F. de Carvalho, T. Bernarda, *Redes Neurais Artificiais - Teoria e Pratica* (LTC, 2007)

5. D. Reboucas, Utilização de redes neurais artificiais para detecção e diagnóstico de falhas. Master's thesis, Universidade Federal do Rio Grande do Norte (2011)

6. C. Wu, J. McLarty, *Neural Networks and Genome Informatics* (Elsevier Science Inc., New York, NY, USA, 2000)

7. Y. Masayuki, K. Motoki, M. Toshio, S. Reiko, H. Yuji, I. Tadayuki, Gene **99**(1), 109 (1991). DOI http: //dx.doi.org/10.1016/0378-1119(91)90041-9. URL http://www.sciencedirect.com/science/article/pii/0378111991900419

8. M. O'Neill, Nucleic Acids Research **19**(2), 313 (1991)

9. B. Demeler, G. Zhou, Nucleic Acids Research **19**(7), 1593 (1991)

10. Scheila de Avila e Silva, Sergio Echeverrigaray, Journal of Theoretical Biology **287**(0), 92 (2011). DOI http://dx.doi.org/10.1016/j.jtbi.2011.07.017. URL http://www.sciencedirect.com/science/article/pii/S0022519311003675

11. D. Rumelhart, G. Hinton, R. Williams, *Neurocomputing: Foundations of Research* (MIT Press, Cam-bridge, MA, USA, 1988). URL http://dl.acm.org/citation.cfm?id=65669.104451

12. K. Hornik, M. Stinchcombe, H. White, Neural Netw. **2**(5), 359 (1989). DOI 10.1016/0893-6080(89) 90020-8. URL http://dx.doi.org/10.1016/0893-6080(89)90020-8

13. I. Mahadevan, I. Ghosh, Nucleic Acids Res **22**(11), 2158 (1994). URL http://www.biomedsearch. com/nih/Analysis-Ecoli-promoter-structures-using/8029027.html

14. A. Pedersen, J. Engelbrecht, in *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, Cambridge, United Kingdom, July 16-19, 1995* (1995), pp. 292–299. URL http://www.aaai.org/Library/ISMB/1995/ismb95-035.php

15. E. Oppon, Synergistic use of promoter prediction algorithms: A choice for a small training dataset. Doutorado em ciencia da computacao, South African National ioinformatics Institute (SANBI) (2000)

16. S. Burden, Y. Lin, R. Zhang, Bioinformatics **21**(5), 601 (2005). URL http://dblp.uni-trier. de/db/journals/bioinformatics/bioinformatics21.html

17. V. Cotik, R. Zaliz, I. Zwir, Fuzzy Sets Syst. **152**(1), 83 (2005). DOI 10.1016/j.fss.2004.10.016. URL http://dx.doi.org/10.1016/j.fss.2004.10.016

18. I. dos Santos, R. Alves, C. Blaha, VIII ERMAC - 8 Encontro Regional de Matemática Aplicada e Computacional (2008)

19. C. Bland, A. Newsome, A. Markovets, BMC Bioinformatics **11**(6), 1 (2010). DOI 10.1186/1471-2105-11-S6-S17. URL http://dx.doi.org/10.1186/1471-2105-11-S6-S17

20. H. Meng, J. Wang, Z. Xiong, F. Xu, G. Zhao, Y. Wang, PLoS ONE **8**(4), 1 (2013). DOI 10.1371/journal.pone.0060288. URL http://dx.doi.org/10.1371%2Fjournal.pone.0060288

21. R.K. Umarov, V.V. Solovyev, PLOS ONE **12**(2), 1 (2017). DOI 10.1371/journal.pone.0171410. URL https://doi.org/10.1371/journal.pone.0171410

22. A. Kumar, M. Bansal, DNA Research **24**(1), 25 (2017). DOI 10.1093/dnares/dsw045. URL +http: //dx.doi.org/10.1093/dnares/dsw045

5.3 COMPARAÇÃO E DISCUSSÃO

Os trabalhos previamente descritos na literatura cujo objetivo é a predição de regiões promotoras em procariotos focaram seus esforços primordialmente em bactérias Gram-negativas, mais especificamente *E. coli.* Diferentemente destes, o presente trabalho teve como foco de estudo a bactéria Gram-positiva *B. subtilis* em duas ferramentas computacionais. Estes *softwares* permitem classificação de uma determinada sequência como promotora e a probabilidade de ser reconhecida pelos diferentes fatores Sigma desta bactéria. Foram utilizadas as metodologias computacionais de redes neurais artificiais e a de máquinas de vetor de suporte, gerando assim os *softwares* BacPP+ (5.2) e BacSVM+ (5.1). O objetivo de ambas é o ajuste de parâmetros para a melhor predição possível de regiões promotoras da bactéria *B. subitilis*.

As redes neurais artificiais, utilizadas no *software* BacPP+, são sistemas de aprendizado de máquina inspirados no funcionamento de redes neurais biológicas. Elas aprendem a partir dos exemplos de sequencias promotoras e apresentam capacidade de generalização do conjunto de treinamento. A vantagem desta solução é o fato delas puderem aprender a reconhecer padrões degenerados, imprecisos e incompletos como é o caso das regiões promotoras. Além disso, apresentam ótimo desempenho em grandes sequências genômicas. Como desvantagem, destaca-se a necessidade do alinhamento das sequências antes da alimentação da rede. Já nas máquinas de vetor de suporte, utilizadas no *software* BacSVM+, o objetivo é a definição de um hiperplano como superfície de decisão de tal forma que a margem de separação entre os promotores (exemplos positivos) e os falsos promotores (exemplos negativos) seja máxima. No entanto, este tipo de solução não incorpora conhecimento do domínio do problema, como por exemplo características estruturais dos promotores.

Para validar o BacPP+ e o BacSVM+, foi feita a coleta e pré-processamento dos dados genéticos da bactéria *B. subtilis,* obtendo 767 sequências promotoras. Este processo foi longo e dificultado devido a falta de um banco de dados que armazene os dados gênicos de regiões promotoras de *B. subtilis,* o que acontece

com *E. coli* através do RegulonDB. A partir dos dados obtidos, foram realizadas 32000 simulações com a rede neural artificial e 17150 com a máquina de vetor de suporte. Destacam-se como melhores resultados 98.57% e 97.69% de acurácia em sua predição com redes neurais (*Multilayer Perceptron* – MLP com 5 e 7 neurônios na camada oculta e 1 neurônio na camada de saída). Com 8 neurônios na camada oculta, a rede apresentou comportamento de *overtraining*. Já a máquina de vetor de suporte, foi possível obter 93.20% e 95.63% de acurácia em sua predição combinando os kernels SIGMOID e RBF com o algoritmo Nu-SVC. Ambos os resultados são condizentes com os encontrados na literatura.

As soluções encontradas na literatura que utilizam redes neurais artificiais para a predição de promotores em *B. subtilis* são: (1) NANS (OPPON, 2000) e (2) CNNProm (UMAROV E SOLOVYEV, 2017). O NANS acerta classificar uma sequência como promotora 60% das vezes e em afirmar que não é promotora em 50% das vezes. Diferentemente do NANS, o BacPP+ foi validado com um número maior de amostras, resultando assim em um melhor aprendizado da rede e em maior acurácia. Já o CNNProm utilizaram a biblioteca Keras (linguagem Python) e obtiveram acurácia de 86% para *B. subtilis.* No entanto, não utilizaram promotores reconhecidos pelo sigma SigL, resultando um número ligeiramente menor (746 sequencias promotoras) que o utilizado para validar o BacPP+ (767 sequencias promotoras). Seus dados também foram coletados do banco de dados DBTBS.

Os trabalhos encontrados na literatura que fazem a predição de regiões promotoras de *B. subtilis* a partir de uma solução que utiliza máquinas de vetor de suporte são: (1) WEKA (MONTEIRO et al., 2005), (2) PePPER (DE JONG et al., 2012) e (3) TSS SVM (MEYSMAN et al., 2014). Monteiro *et al*. (2005) não desenvolveram um software próprio. Eles utilizaram a ferramenta WEKA, diferentemente do BacSVM+ que é implementado em linguagem Python e linguagem Java. Além disso, foram usados apenas 112 promotores de *B. subtilis* (767 promotores foram utilizados para validar o BacSVM+). A acurácia obtida foi menor que a do BacSVM+, 76%. Já De Jong *et al*. (2012) desenvolveram o *webserver PePPER* (não requer instalação e pode ser acessado pela Internet), mas eles não apresentam resultados validados. Finalmente, o TSS SVM analisaram os perfis estruturais de regiões promotoras e não focaram suas atenções especificamente ao problema de predição promotora. Eles afirmam que as regiões

promotoras são menos estáveis e mais rígidas que o resto do genoma, mas isto é menos visível em Gram-positivas.

Sendo assim, é possível perceber que quando comparados os resultados obtidos pelas duas soluções com os trabalhos encontrados na literatura com o o mesmo objetivo, os resultados são satisfatórios e competitivos. Ambas as tecnologias de redes neurais artificiais e máquinas de vetor de suporte podem ser utilizadas para utilizadas com sucesso como solução do problema de predição de promotores da bactéria *B. subtilis*, embora as RN tenham obtido melhor desempenho.

Por fim, algumas arquiteturas de BacPP+ e BacSVM+ obtiveram resultados altíssimos (maiores dos que os apresentados como melhores resultados), mas não podem ser consideradas a melhor solução pois seus valores de sensibilidade e especificidade não foram similares. Quando estes diferem muito, o número de falsos positivos e falsos negativos podem ser altos. Além disso, ressalta-se que em relação ao parâmetro C, o kernel SIGMOID necessitou aumentar mais o custo de punição dos vetores de suporte quando falhavam em relação aos outros *kernels* (valor ótimo de 256). Já em relação ao parâmetro G, o desempenho das arquiteturas foram satisfatórios quando usados valores baixos para *Gamma* (e.g., 1.52587890625E − 5). Em ambas as soluções foram usados os mesmos dados de entrada e validação cruzada (*k-cross validation*) de 5-*fold*.

6 CONCLUSÕES

O presente trabalho teve como objetivo a predição de regiões promotoras da bactéria Gram-positiva *B. subtilis*. Os promotores obtidos de *B. subtilis* podem ser utilizados em estudos futuros sobre regulação gênica. A regulação da expressão do gene é essencial na adaptação metabólica à mudanças no ambiente, permitindo a multiplicação e sobrevivência bacteriana. Além disso, grande parte dos trabalhos sobre promotores bacterianos estão restritos a bactérias Gram-negativas, particularmente *E. coli*. Os promotores obtidos de *B. subtilis* possibilitam futuras pesquisas nesta área. Por fim, existem poucos dados de promotores de bactéria Gram-positiva na literatura. A metodologia implementada no presente trabalho pode ser utilizada por outros pesquisadores para validar sequências promotoras bacterianas Gram-positivas.

Foram utilizadas duas técnicas computacionais para solucionar este problema de predição: redes neurais artificiais (*software* BacPP+) e máquinas de vetor de suporte (*software* BacSVM+). Para validar ambas as soluções, foram obtidos o arquivo FASTA contendo o genoma no NCBI e os promotores reconhecidos pelos fatores de transcrição da bactéria *B. subtilis*, totalizando 15 fatores divididos em 2 domínios: (1) Sigma 54 (SigL) e (2) Sigma 70 (demais fatores de transcrição). Devido a dificuldade em obter dados sobre promotores de *B. subtilis*, a coleta de dados e pré-processamento dos mesmos é a maior contribuição do trabalho. Foram obtidos 767 promotores para a Bactéria *B. subtilis*, sendo a maioria encontrados através do fator de transcrição SigA (46.07%).

A partir dos dados obtidos, foram realizadas 32000 simulações com BacPP+ e 17150 com BacSVM+. Os melhores resultados são 98.57% e 97.69% de acurácia em sua predição com redes neurais (*Multilayer Perceptron* – MLP com 5 e 7 neurônios na camada oculta e 1 neurônio na camada de saída). Já o BacSVM+ obteve 93.20% e 95.63% de acurácia em sua predição combinando os kernels SIGMOID e RBF com o algoritmo Nu-SVC. Ambos os resultados são condizentes e competitivos se comparados com os encontrados na literatura. Além disso, o comportamento de ambas comprova a confiabilidade dos dados obtidos. Por fim, é

possível concluir que é viável a predição de sequências promotoras de *B. subtilis* através de ambas as técnicas computacionais de redes neurais artificiais e máquinas de vetor de suporte.

Como trabalhos futuros, indica-se o uso de informações estruturais (e.g. estabilidade, curvatura e deformabilidade) de sequências promotoras para melhorar o desempenho das soluções desenvolvidas. Elas podem auxiliar na diferenciação de regiões não-promotoras. Além disso, uma possível melhoria que pode ser implementada no *software* BacSVM+ é a possibilidade de restauração de estado de cada passo a partir dos arquivos de *log* gerados. Por fim, indica-se uma análise sobre o possível uso de redes neurais convolucionais e *ensemble learning* para melhorar a automatização no reconhecimento das regiões promotoras.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, M.B. **Treinamento de SVMs Utilizando Reamostragem Baseada Em Erro e Estratégias A Priori de Seleção de Amostras.** 326 p., 2002. Doutorado em Engenharia Elétrica — Universidade Federal de Minas Gerais.

ALI, N. O.; BIGNON, J.; RAPOPORT, G.; DEBARBOUILLE, M. Regulation of the Acetoin Catabolic Pathway Is Controlled by Sigma L in *Bacillus subtilis*. **Journal of Bacteriology**. v. 183, n. 8, p. 2497-2504, 2001.

BALDI, P.; BRUNAK, S. **Bioinformatics: The Machine Learning Approach**. 2ª Edição. Cambridge: MIT. 351 p., 2001.

BAJESTANI, M. A.; RABBANI, M.; RAHIMI-VAHED, A. R.; KHOSHKHOU, G. B. A Multi-objective Scatter Search for a Dynamic Cell Formation Problem. **Computers & Operations Research**, v. 36, n. 3, p. 777-794, 2009.

BELITSKY, B. R., SONENSHEIN, A. L. An Enhancer Element Located Downstream of the Major Glutamate Dehydrogenase Gene of *Bacillus subtilis*. **Proceedings of the National Academy of Sciences of the United States of America**. v. 96, n. 18, p. 10290-10295, 1999.

BERNARDA, T. **Redes Neurais Artificiais – Teoria e Prática.** 2ª Edição. 898 p., LTC, 2011.

BLAND, C.; NEWSOME, A.; MARKOVETS, A. Promoter Prediction in *E. coli* based on SIDD Profiles and Artificial Neural Networks. **BMC Bioinformatics**, v. 11, n. 6, p. 1–4, 2010.

BOSER, B.; GUYON, I.; VAPNIK, V. A Training Algorithm for Optimal Margin Classifiers. Fifth Annual Worwshop on Computacional Learning Theory, Pittsburgh. **Anais**. . . ACM. p.144–152, 1992.

BROWN, M.; GRUNDY, W.; LIN, D.; CRISTIANINI, N.; SUGNET, C.; FUREY, T.; ARES, M.; HAUSSLER, D. Knowledge-based Analysis of Microarray Gene Expression Data by Using Support Vector Machines. **Proceedings of The National Academy of Sciences of The United States of America**, v. 97, p. 262–267, 2000.

BURDEN, S.; LIN, Y.-X.; ZHANG, R. Improving Promoter Prediction for the NNPP2.2 Algorithm: A Case Study using *Escherichia coli* DNA Sequences. **Bioinformatics** 21: 601-607, 2005.

CALOGERO, S., GARDAN, R., GLASER, P., SCHWEIZER, J., RAPOPORT, G., DÉBARBOUILLE, M. RocR, A Novel Regulatory Protein Controlling Arginine Utilization in *Bacillus subtilis*, Belongs to the NtrC/NifA Family of Transcriptional Activators. **Journal of Bacteriology,** v. 176, n. 5, p. 1234-1241, 1994.

CAMPBELL, C. Kernel Methods: A Survey of Current Techniques. **Neurocomputing**, v. 48, p. 63–84, 2002.

CASEY, E., MAHONY, J., O'CONNEL-MOTHERWAY, M., BOTTACINI, F., CORNELISSEN, A., NEVE, H., VAN SINDEREN, D. Molecular Characterization of Three *Lactobacillus delbrueckii subsp. bulgaricus phages*. **Applied and environmental microbiology**, v. 80, n. 18, p. 5623-5635, 2014.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A Library for Support Vector Machines. **ACM Trans. Intell. Syst. Technol.**, v. 2, n. 3, p. 27, 2011.

CORTES, C.; VAPNIK, V. Support Vector Networks. **Machine Learning**, v. 20, p. 1–25, 1995.

COTIK, V.; ZALIZ, R.; ZWIR, I. A Hybrid Promoter Analysis Methodology for Prokaryotic Genomes. **Fuzzy Sets Syst**., v. 152, n. 1, p. 83–102, 2005.

COVER, T. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. **IEEE Transactions on Computers**, v. 14, p. 326–334, 1965.

CRISTIANINI, N.; SHAWE-TAYLOR, J. **An Introduction to Support Vector Machines and Other kernel-based Learning Methods**. Cambridge: Cambridge University Press, 204 p., 2000.

CROOKS, G. E.; HON G.; CHANDONIA, J. M.; BRENNER, S. E. WebLogo: A Sequence Logo Generator. Genome Research, v. 14, p. 1188-1190, 2004.

DAMASEVICIUS, R. Structural Analysis of Regulatory DNA Sequences Using Grammar Inference and Support Vector Machine. **Neurocomput.,** v. 73, n. 633, 2010.

DE AVILA E SILVA, S.; ECHEVERRIGARAY, S.; GERHARDT, G.J.L. BacPP: Bacterial Promoter Prediction a Tool for Accurate Sigma Factor Specific Assignment in *Enterobacteria*. **Journal of Theoretical Biology**, v. 287, n. 0, p. 92 – 99, 2011.

DE JONG, A.; PIETERSMA, H.; CORDES, M.; KUIPERS, O.; JAN, K. PePPER: A Webserver for Prediction of Prokaryote Promoter Elements and Regulons. **BMC Genomics**, v. 13, n. 1, p. 1–10, 2012.

DÉBARBOUILLE, M., MARTIN-VERSTRAETE, I., KLIER, A., RAPOPORT, G. The Transcriptional Regulator LevR of *Bacillus subtilis* has Domains Homologous to Both Sigma 54 and Phosphotransferase System-dependent Regulators. **Proceedings of the National Academy of Sciences,** v. 88, n. 6, p. 2212-2216, 1991.

DÉBARBOUILLE, M., DÉBARBOUILLE, R., GARDAN, R., ARNAUD, M., RAPOPORT, G. Role of BkdR, a Transcriptional Activator of the SigL-Dependent Isoleucine and Valine Degradation Pathway in *Bacillus subtilis*, **Journal of bacteriology,** v. 181, n. 7, p. 2059-2066, 1999.

DEMELER, B.; ZHOU, G. Neural Network Optimization for *E. coli* Promoter Prediction. **Nucleic Acids Research**, v. 19, n. 7, p.1593–1599, 1991.

DISTANTE, C.; ANCONA, N.; SICILIANO, P. Support Vector Machines for Olfactory Signal Recognition. **Sensors and Actuators**, v. 88, p. 30–39, 2003.

DRUCKER, H.; WU, D.; VAPNIK, V. Support Vector Machines for Spam Categorization. **IEEE Transactions on Neural Networks**, v.10, p.1048–1054, 1999.

GAMA-CASTRO, S.; JIMÉNEZ-JACINTO, V.; PERALTA-GIL, M.; SANTOS-ZAVALETA, A.; PENALOZA-SPINOLA, M. I.; CONTRERAS-MOREIRA, B.; BONAVIDES-MARTINEZ, C. RegulonDB (version 6.0): Gene Regulation Model of *Escherichia coli* K-12 Beyond Transcription, Active (Experimental) Annotated Promoters and Textpresso Navigation. **Nucleic acids research**, v. 36, p. 120-124, 2008.

GARDAN, R.; RAPOPORT, G.; DÉBARBOUILLE, M. Expression of the rocDEFOperon Involved in Arginine Catabolism in *Bacillus subtilis*. **Journal of Molecular Biology**, v. 249, n. 5, p. 843-856, 1995.

GORDON, L.; CHERVONENKIS, A. Y.; GAMMERMAN, A. J.; SHAHMURADOV, I. A.; SOLOVYEV, V. V. Sequence Alignment for Recognition of Promoter Regions. **Bioinformatics,** v. 19, p. 1964-1971, 2003.

GORDON , J. J.;TOWSEY, M.; HOGAN, J.; MATHEWS, S. A. TIMMS, P.; Improved Prediction of Bacterial Transcription Start Sites. **Bioinformatics,** v. 22, p. 142-148, 2006.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I.H. The WEKA data mining software: an update. **ACM SIGKDD explorations newsletter**, v. 11, n. 1, p. 10-18, 2009.

HAYKIN, S. **Neural networks: a comprehensive foundation**. 2. ed. New Jersey: Prentice-Hall, 1999. 842 p.

HELMANN, J. Compilation and Analysis of *Bacillus subtilis* sigma A-dependent Promoter Sequences: evidence for extended contact between rna polymerase and upstream promoter dna. **Nucleic Acids Research**, v.23, n.13, p.2351–2360, 1995.

HERTZ, G. Z.; STORMO, G. D. Identifying DNA and Protein Patterns with Statistically Significant Alignments of Multiple Sequences. **Bioinformatics,** v. 15, p. 563-577, 1999.

HOOK-BARNARD, I.; JOHNSON, X. B.; HINTON, D. M. *Escherichia coli* RNA Polymerase Recognition of a σ70 – Dependent Promoter Requering a -35 DNA Element and an Extended -10 TGn Motif. **Journal of Bacteriology,** v. 188, p. 8352-8359, 2006.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer Feedforward Networks Are Universal Approximators. **Neural Network**, v. 2, n. 5, p. 359–366, 1989.

HSU, C.; CHANG, C.; LIN, C. A Practical Guide to Support Vector Classification, p. 1-16, 2003.

JEFFREY, N.; SHAOGANG, G. Composite Support Vector Machines for Detection of Faces Across Views and Pose Estimation. **Image and Vision Computing**, v. 20, p. 359–368, 2002.

JADID, M. N.; FAIRBAIRN, D. R. Neural-network Applications in Predicting Moment-curvature Parameters from Experimental Data. **Engineering Applications of Artificial Intelligence**, v. 123, n. 4, p. 261-266, 1996.

JOACHIMS, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: European Conference on Machine Learning, Berlin. **Proceedings**. . . Springer, p. 137–142, 1998.

KANHERE, A.; BANSAL, M. A Novel Method for Prokaryotic Promoter Prediction Based on DNA Stability. **BMC Bioinformatics,** v. 6, n. 1, 2005.

KEERTHI, S.; SHEVADE, S.K.; BHATTACHARYYA, C.; KARUTURI, R.K. A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. **IEEE Transactions on Neural Networks**, v. 11, n. 1, p. 124-136, 2000.

KIRYU, H.; OSHIMA, T.; KIYOSHI, A. Extracting Relations Between Promoter Sequences and Their Strengths from Microarray Data. **Bioinformatics**, v. 21, p. 1062-1068, 2005.

KOTROPOULOS, C.; PITAS, I. Segmentation of Ultrasonic Images using Support Vector Machines. **Pattern Recognition Letters**, v. 24, p. 715–727, 2003.

KOZOBAY-AVRAHAM, L.; HOSID, S.; VOLKOVICH, Z.; BOLSHOY, A. Prokaryote Clustering based on DNA Curvature Distributions. **Discrete Applied Mathematics**, v. 11, p. 2378-2387**,** 2008.

KREBS, J. E.; GOLDSTEIN, E. S.; KILPATRICK, S. T. **Lewin's Genes XI**, Jones & Bartlett, Burlington, MA , 940 p., 2017.

KUMAR, A.; BANSAL, M. Unveiling DNA Structural Features of Promoters Associated with Various Types of TSSs in Prokaryotic Transcriptomes and Their Role in Gene Expression. **DNA Research**, v. 24, n. 1, p. 25-35, 2016.

LETUNIC, I.; BORK, P. Interactive Tree of Life (iTOL) v3: An Online Tool for the Display and Annotation of Phylogenetic and Other Trees. **Nucleic acids research**, v. 44, n. 1, p. 242-245, 2016.

LISSER, S.; MARGALIT, H. Compilation of E. coli mRNA Promoter Sequences. **Nucleic Acids Res**, v. 21, n. 7, p. 15–16, 1993.

LORENA, A.; LEON, A.; CARVALHO, F. Uma Introdução às Support Vector Machines. **RITA**, v. 14, n. 2, p. 43–67, 2007.

LUENBERGER, D. **Linear and Nonlinear Programming**. 2ª Edição. USA: Addison-Wesley, 492 p., 1986.

MADER, S. S.; WINDELSPECHT, M.; COGNATO, A. **Biology.** 10ª Edição. New York, NY: McGraw-Hill, 907 p., 2015.

MAHADEVAN, I.; GHOSH, I. Analysis of *E. coli* Promoter Structures using Neural Networks. **Neural networks,** v. 12, p. 717-725, 1999.

MANGASARIAN, O.; MUSICANT, D. **Data Discrimination via Nonlinear Generalized Support Vector Machines**. Complementarity: Applications, Algorithms and Extensions, p. 233-251, 1999.

MARTIN, I., DÉBARBOUILLE, M., KLIER, A. Rapoport G. Induction and Metabolite Regulation of Levanase Synthesis in *Bacillus subtilis*. **Journal of Bacteriology**. v. 171, n. 4, p. 1885-1892, 1989.

YAMADA, M.; KUBO, M.; MIYAKE, T.; SAKAGUCHI, R.; HIGO, Y.; IMANAKA, T. Promoter Sequence Analysis in *Bacillus* and *Escherichia*: Construction of Strong Promoters in *E. coli*. **Gene**, v. 99, n. 1, p. 109–114, 1991.

MENG, H.; WANG, J.; XIONG, Z.; XU, F.; ZHAO, G.; WANG, Y. Quantitative Design of Regulatory Elements Based on High-precision Strength Prediction Using Artificial Neural Network. **PLoS ONE**, v. 8, n. 4, p. 1–9, 2013.

MEYSMAN, P.; COLLADO-VIDES, J.; MORETT, E.; VIOLA, R.; ENGELEN, K.; LAUKENS, K. Structural Properties of Prokaryotic Promoter Regions Correlate with Functional Features. **PloS one**, v. 9, n. 2, 2014.

MONTEIRO, M.; DE SOUTO, M.; GONÇALVES, L.; AGNEZ-LIMA, L. Machine Learning Techniques for Predicting *Bacillus subtilis* Promoters. **Brazilian Symposium on Bioinformatics.** Springer, p. 77–84, 2005.

MOUNT, D. W. **Bioinformatics : Sequence and Genome Analysis.** 2ª Edição. New York: Cold Spring Harbor Laboratory Press, 564 p., 2013.

NELSON, D.; COX, M. **Princípios de Bioquímica de Lehninger**. 6ª Edição. Artmed, 1328 p., 2011.

O'NEILL, M. C. Training Back-propagation Neural Networks to Define and Detect DNA-Binding Sites. **Nucleic Acids Research,** v. 19, p. 313-318, 1991.

OPPON, E. **Synergistic Use of Promoter Prediction Algorithms: a choice for a small training dataset**. Doutorado em Ciência da Computação — South African National ioinformatics Institute (SANBI), 2000.

OSUNA, E.; FREUND, R.; GIROSI, F. An Improved Training Algorithm for Support Vector Machines. IEEE Workshop on Neural Networks for Signal Processing. p. 276-285, 1997.

PANG, S.; KIM, D.; BANG, S. Y. Membership Authentication in the Dynamic Group by Face Classification Using SVM Ensemble. **Pattern Recognition Letters**, v.24, p. 215–255, 2003.

PEDERSEN, A. G.; ENGELBRECHT, J. Investigations of *Escherichia coli* Promoter Sequences with Artificial Neural Networks: New Signals Discovered Upstream of the Transcriptional Start Point. **Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology** (ISMB-95), v. 3, p. 292-299, 1995.

PLATT, J. **Advances in Kernel Methods: Support Vector Machines**. Cambridge, MA: MIT Press, 373 p., 1998.

POLAT, K.; GÜNES, S. A novel approach to estimation of *E. coli* promoter gene sequences: Combining feature selection and least square support vector machine (FS_LSSVM). **Applied Mathematics and Computation,** v. 190, p. 1574–1582, 2007.

REBOUÇAS, D. **Utilização de Redes Neurais Artificiais para Detecção e Diagnóstico de Falhas**. Dissertação de Mestrado em Ciência da Computação — Universidade Federal do Rio Grande do Norte, 93 p., 2011.

REIS, A. **Reconhecimento e Predição de Promotores Procarióticos: Investigação de uma Metodologia *in silico* Baseada em HMMs**. Dissertação de Mestrado em Ciência da Computação — Universidade do Vale do Rio dos Sinos, 116 p., 2005.

REYS, L.; MACEDO, J.; DAMALIO, J. C. P. **Dogma Central da Biologia Molecular e Introdução à Bioinformática**. Av. L2 Sul Quadra 603 Conjunto C. CEP 70200-630. Brasília-DF: W Educacional Editora e Cursos Ltda, 2011.

RODRIGUEZ, J. D.; PEREZ, A.; LOZANO, J. A. Sensitivity Analysis of K-Fold Cross Validation in Prediction Error Estimation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 3, p. 569-575, 2010.

ROSSUM, G. V. **Extending and Embedding the Python Interpreter**. Report CS-R9527, 1995.

RUFF, E.; RECORD, T.; ARTSIMOVITCH, I. Initial Events in Bacterial Transcription Initiation. **Biomolecules**, v. 5, n. 2, p. 1035, 2015.

RUMELHART, D. E; HINTON, G. E.; WILLIAMS, R. J Learning Representations by Back-propagating Errors. **Nature,** v. 323, p. 533-536, 1986.

SANTOS, I.; ALVES, R.; BLAHA, C. Estudo para Identificação de Promotores Procarióticos Através de Classificação *Bayesiana*. **VIII ERMAC - 8º Encontro Regional de Matemática Aplicada e Computacional**, 2008.

SIERRO, N.; MAKITA Y.; DE HOON M.; NAKAI, K. DBTBS: A Database of Transcriptional Regulation in *Bacillus subtilis* Containing Upstream Intergenic Conservation Information. **Nucleic Acids Research**, v. 36, p. 93–96, 2008.

SIWO, G.; RIDER, A.; TAN, A.; PINAPATI, R.; EMRICH, S.; CHAWLA, N.; FERDIG, M. Prediction of Fine-tuned Promoter Activity from DNA Sequence. **F1000Research**, v. 5, 2016.

SONG, W.; MAISTE, P. J.; NAIMAN, D. Q.; WARD, M. J. Sigma 28 Promoter Prediction in Members of the Gammaproteobacteria. **FEMS Microbiology Letters,** v. 271, p. 222-229, 2007.

SUYKENS, J.; VANDEWALLE, J. Least Squares Support Vector Machine Classifiers. **Neural Processing Letters**, v. 9, p. 293–300, 1999.

UMAROV, R. K.; SOLOVYEV, V. V. Recognition of Prokaryotic and Eukaryotic Promoters using Convolutional Deep Learning Neural Networks. **PloS one**, v. 12, n. 2, 2017.

VALENTINI, G. Gene Expression Data Analysis of Human Lymphoma Using Support Vector Machines and Output Coding Ensembles. **Artificial Intelligence in Medicine**, v. 26, p. 281–304, 2002.

VAPNIK, V. **Statistical Learning Theory**. 1ª Edição. John Wiley & Sons, 768 p., 1998.

WANG, Y.; CHUA, C.-S.; HO, Y.-K. Facial Feature Detection and Face Recognition from 2D and 3D Images. **Pattern Recognition Letters**, v. 23, p. 1191–1202, 2002.

WANG, W.; SUN, M.. Phylogenetic Relationships Between *Bacillus* Species and Related Genera Inferred from 16s rDNA Sequences. **Brazilian Journal of Microbiology**, v. 40, n. 3, p. 505-521, 2009.

WU, C. H.; MCLARTY, J. W. **Neural Networks and Genome Informatics.** New York: Elsevier, 205 p., 2000.

XUE, H.; YANG, Q.; CHEN, S. **SVM: Support Vector Machines**. Chapman & Hall/CRC: London, UK, 2009.

ZANATY, E. Support Vector Machines (SVMs) versus Multilayer Perceptron (MLP) in Data Classification. **Egyptian Informatics Journal,** p. 177 – 183, 2012.

ZEIGLER, D. R.; PRÁGAI, Z.; RORDRIGUEZ, S.; CHEVREUX, B.; MUFFLER, A.; ALBERT, T.; PERKINS, J. B. The Origins of 168, W23, and Other *Bacillus subtilis* Legacy Strains. **Journal of Bacteriology**, v. *190, n.* 21, p. 6983 – 6995, 2008.

APÊNDICE 1 – CÓDIGO-FONTE

```
# DESCRIÇÃO: programa procura por regiões não  promotoras em genomas  completos de maneira aleatória
# ENTRADAS:
#     - arquivos texto na pasta /Promotores/ contendo as sequencias promotoras
#     - arquivos texto na pasta /Genomas/  contendo as sequencias completas de DNA das bacterias (ex: file1.txt)
# SAÍDAS:
#     - arquivos texto na pasta /Falsos/ contendo as sequencias aleatórias. Ex: _cuABC.fasta

import sys, os.path, os, time, re, random

inicio = time.time()
if ('Falsos' not in os.listdir(os.curdir)):
  os.mkdir(os.curdir + "/Falsos/")

pastasPromotores = listFilesFromFolder('/Promotores/', 'Genomas : ')
print('-'*100)
print('Falsos Promotores Encontrados: ')
print('-'*100)
cont = 1
for pasta in pastasPromotores:
   arquivosPromotores = listFilesFromFolder('/Promotores/' + pasta + '/', 'Numero Total de Promotores Encontrados: ')
   for arquivo in arquivosPromotores:
      linhasPromotores = getLinesFromFile('', "/Promotores/" + pasta +'/' + arquivo, 1, 1)
   listaFalsos = []
   x = 0
   for arquivo in arquivosPromotores:
      if (pasta not in os.listdir(os.curdir + "/Falsos/")):
         os.mkdir(os.curdir + "/Falsos/" + pasta)
      path = os.path.expanduser(os.curdir + "/Falsos/" + pasta)
      try:
         arquivoNaoPromotor = open(path + '/nonPromoter' + str(cont) + '.fasta', 'w')
         regiaoNaoPromotora = searchNonPromoter(listaFalsos, linhasPromotores, "/Genomas/file1.fasta")
         listaFalsos.append(regiaoNaoPromotora)
         print(str(cont) + ' ' + regiaoNaoPromotora)
         cont+=1
         arquivoNaoPromotor.write(regiaoNaoPromotora)
         arquivoNaoPromotor.write('\n')
         arquivoNaoPromotor.flush()
         arquivoNaoPromotor.close()
      except IOError:
         print('Erro: Impossível criar o arquivo de saida: não promoter ' + arquivo + '.fasta')
   x += 1
   file = open(os.curdir + "/Falsos/" + 'Falsos' + str(x) + '.txt', 'w')
   for naoProm in listaFalsos:
      file.write(naoProm + '\n')
      file.flush()
   file.close()
fim = time.time()
print(' Tempo Total: ', round(fim-inicio, 2), 's')
```