UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

GUSTAVO MARTINS WAMSER

IDENTIFICAÇÃO DE PROMOTORES EM SEQUÊNCIAS DE DNA DE BACTÉRIAS *Escherichia coli* ATRAVÉS DE MÁQUINAS DE VETORES DE SUPORTE

CAXIAS DO SUL

2020

GUSTAVO MARTINS WAMSER

IDENTIFICAÇÃO DE PROMOTORES EM SEQUÊNCIAS DE DNA DE BACTÉRIAS *Escherichia coli* ATRAVÉS DE MÁQUINAS DE VETORES DE SUPORTE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dr. André Luis Martinotto

CAXIAS DO SUL

2020

GUSTAVO MARTINS WAMSER

IDENTIFICAÇÃO DE PROMOTORES EM SEQUÊNCIAS DE DNA DE BACTÉRIAS *Escherichia coli* ATRAVÉS DE MÁQUINAS DE VETORES DE SUPORTE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 03/07/2020

BANCA EXAMINADORA

Prof. Dr. André Luis Martinotto
Universidade de Caxias do Sul - UCS

Prof. Me. Gustavo Sganzerla Martinez
Universidade de Caxias do Sul - UCS

Profa. Dra. Scheila de Avila e Silva
Universidade de Caxias do Sul - UCS

RESUMO

A região promotora, localizada anteriormente à região codificadora dos genes, é essencial para o processo de transcrição presente nas células. Dada a sua importância, a identificação dessas regiões em sequências de DNA é de grande interesse para a comunidade científica. Este trabalho teve como objetivo o desenvolvimento de uma solução para a identificação de regiões promotoras em trechos de DNA de bactérias *Escherichia coli*. A classificação foi realizada através do método de Máquinas de Vetores de Suporte, fazendo uso da biblioteca LibSVM. Para os treinamentos e validações foram utilizadas sequências de DNA obtidas da base RegulonDB, além de versões embaralhadas dessas mesmas sequências. Os testes foram realizados com diferentes fatores sigma, obtendo-se uma acurácia de 75.6% para o σ24, 71.2% para o σ28, 71.2% para o σ32, 68.4% para o σ38, 63.9% para o σ54 e 72.2% para o σ70.

Palavras-chave: DNA, região promotora, promotor, genética, Máquinas de Vetores de Suporte, SVM, LibSVM, RegulonDB

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DNA	*Ácido Desoxirribonucleico*
KNN	*K Nearest Neighbours*
RBF	*Radial Basis Function*
RNA	*Ácido Ribonucleico*
RNAm	*RNA Mensageiro*
RNAr	*RNA Ribossomal*
RNAt	*RNA Transportador*
RNAP	*RNA Polimerase*
RN	*Rede Neural Artificial*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

A célula é a menor parte viva de qualquer organismo vivo. O material genético de uma célula é responsável pela síntese de biomoléculas, reprodução celular e transmissão de características, apresentando-se na forma de duas moléculas principais: Ácido Desoxirribonucleico (DNA) e Ácido Ribonucleico (RNA). O DNA possui trechos conhecidos como genes, responsáveis pela síntese de diferentes biomoléculas dentro da célula. Uma região específica do gene, conhecida como região promotora, é essencial para que o gene possa ser expressado, ou seja, para que aquela biomolécula seja sintetizada corretamente (CARVALHO; RECCO-PIMENTEL, 2013) (SANDERS; BOWMAN, 2014).

Alterações na região promotora podem impedir o correto funcionamento de um gene, fazendo com que biomoléculas importantes deixem de ser produzidas, ou características celulares importantes deixem de ser expressadas. Logo, a região promotora possui papel regulador na expressão gênica nos organismos vivos (CARVALHO; RECCO-PIMENTEL, 2013). De fato, diversos estudos observaram a relação entre alterações em promotores e várias doenças, incluindo Talassemia Intermédia (KULOZIK *et al.*, 1991), Síndrome de Rubinstein-Taybi (PETRIF *et al.*, 1995), alergias (HOBBS *et al.*, 1998), asma (BURCHARD *et al.*, 1999) e diabetes (IONESCU-TIRGOVISTE *et al.*, 2015). A identificação de promotores em sequências de DNA é, portanto, de grande interesse para a comunidade científica, uma vez que pode auxiliar no estudo de expressão gênica e doenças relacionadas.

Diversos métodos biotecnológicos podem ser usados para a identificação de promotores em sequências de DNA, como por exemplo o *Full Length cDNA Sequencing*, o *Expressed Sequence Tags (ESTs)* e o *Cap Analysis of Gene Expression (CAGE)* (JACOBI, 2014). Porém, esses métodos não são adequados para serem usados em larga escala. Para realizar essa identificação em tempo menor e com um custo mais baixo, métodos computacionais podem ser aplicados, sendo que estes geralmente são baseados no uso de técnicas de aprendizado de máquina. Dentre os métodos computacionais disponíveis, os mais utilizados são Redes Neurais Artificiais, K Vizinhos Mais Próximos e Máquinas de Vetores de Suporte (FACELI *et al.*, 2011).

Considerando este contexto, neste trabalho foi desenvolvida uma implementação para a identificação de regiões promotoras em sequências de DNA de bactérias *Escherichia coli*, utilizando Máquinas de Vetores de Suporte. Foram utilizadas sequências de DNA provenientes da base RegulonDB (versão 10.6.3) (SANTOS-ZAVALETA *et al.*, 2019), além de versões embaralhadas dessas mesmas sequências. Os treinamentos e validações foram realizados utilizando o método de validação cruzada (KOHAVI, 1995) (DUDA; HART; STORK, 2000), e ao final foi feita uma análise sobre a acurácia, especificidade, sensibilidade e precisão da implementação desenvolvida.

1.1 OBJETIVOS

O principal objetivo deste trabalho consistiu no desenvolvimento de uma aplicação para a identificação de regiões promotoras em sequências de DNA, utilizando Máquinas de Vetores de Suporte. Para tanto, os seguintes objetivos específicos foram realizados:

1. Desenvolvimento de uma implementação para a identificação de regiões promotoras em sequências de DNA, fazendo uso de Máquinas de Vetores de Suporte;
2. Treinamento e validação da implementação desenvolvida;
3. Análise da acurácia, especificidade, sensibilidade e precisão da implementação desenvolvida.

1.2 ESTRUTURA DO TRABALHO

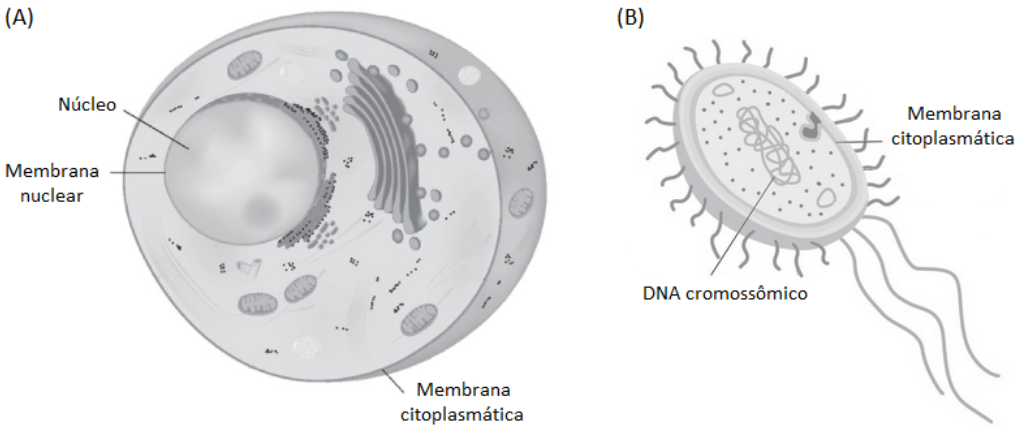
O presente trabalho está organizado da seguinte forma:

- No Capítulo 2 é apresentada a estrutura básica das células e do material genético. Além disso, são apresentados os principais processos realizados durante a síntese de biomoléculas, destacando-se o papel da região promotora presente nos genes.
- No Capítulo 3 são apresentados os métodos computacionais mais utilizados para a identificação de regiões promotoras em sequências de DNA.
- No Capítulo 4 são apresentados os principais conceitos referentes a Máquinas de Vetores de Suporte.
- No Capítulo 5 é apresentada a implementação desenvolvida neste trabalho e os resultados obtidos.
- Por fim, no Capítulo 6 são apresentadas as considerações finais e sugestões de trabalhos futuros.

2 SÍNTESE GENÉTICA BACTERIANA

A célula é a menor parte viva de qualquer organismo. Alguns organismos são unicelulares, como é o caso de bactérias e protozoários, e outros são pluricelulares, como é o caso de mamíferos, peixes, insetos, entre outros. Existem dois tipos de células: as eucariotas e as procariotas. As células eucariotas (Figura 1A) possuem uma membrana separando seu material genético do citoplasma, formando assim um ou mais núcleos celulares. As células procariotas (Figura 1B) não apresentam tal membrana, sendo que seu material genético flutua livremente no citoplasma. O citoplasma é a porção líquida da célula, onde ocorrem reações químicas e a síntese de proteínas, lipídios e outros compostos químicos importantes para a célula (PIMENTA; LIMA, 2015).

Figura 1 – (A) Célula eucariota. (B) Célula procariota.

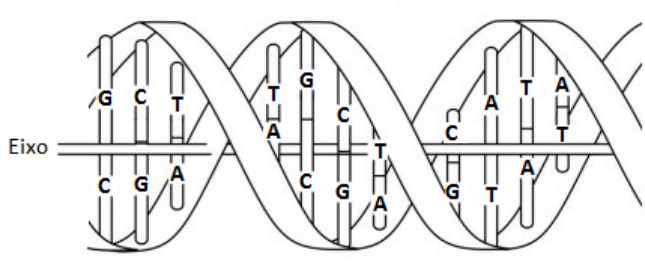


Fonte: Almeida & Pires (2014)

O material genético presente nas células é utilizado tanto no processo de produção de biomoléculas como no processo de reprodução celular e transmissão de características. Este material apresenta-se na forma de duas moléculas principais: o Ácido Desoxirribonucleico (DNA) e o Ácido Ribonucleico (RNA) (CARVALHO; RECCO-PIMENTEL, 2013).

O DNA é um polímero em formato de hélice-dupla, formado por monômeros dos nucleotídeos Adenina (A), Citosina (C), Guanina (G) e Timina (T). Em cada uma das duas fitas do DNA encontram-se sequências desses nucleotídeos, sendo que há um pareamento de bases entre as fitas, sempre entre uma base Adenina e uma Timina, ou entre uma base Citosina e uma Guanina (CARVALHO; RECCO-PIMENTEL, 2013). Na Figura 2 pode ser observada a estrutura de fita-dupla do DNA, onde as duas fitas giram em torno de um eixo imaginário para formar uma hélice. Na Figura 2 também pode ser observado o pareamento de bases nucleotídicas entre as fitas.

Figura 2 – Estrutura do DNA.



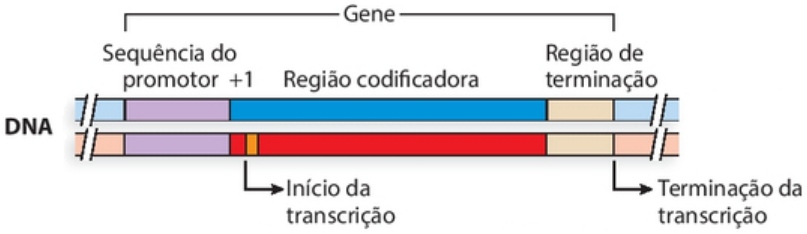
Fonte: Bordoni *et al.* (2011)

O RNA possui uma estrutura similar ao DNA, mas se diferencia pelo fato de possuir apenas uma fita e não possuir a base Timina (T), sendo que essa é substituída por uma base chamada Uracila (U). Existem três famílias principais de RNA, presentes tanto nas células procariotas como eucariotas: o RNA transportador (RNAt), responsável pela identificação, ligação e transporte de aminoácidos presentes no citoplasma para a síntese de biomoléculas; o RNA mensageiro (RNAm), responsável por carregar a informação necessária para a síntese de uma biomolécula; e o RNA ribossomal (RNAr), que junto com proteínas específicas, forma uma organela conhecida como ribossomo (CARVALHO; RECCO-PIMENTEL, 2013).

Em bactérias, dois processos importantes utilizam o DNA e o RNA: transcrição e tradução. O processo de transcrição tem a função de sintetizar os diferentes tipos de RNA a partir de uma fita-molde do DNA. O processo de tradução tem a função de sintetizar biomoléculas, fazendo uso dos RNAs construídos pelo processo de transcrição. (CARVALHO; RECCO-PIMENTEL, 2013) (SANDERS; BOWMAN, 2014).

O DNA possui trechos conhecidos como genes, que por sua vez são subdivididos em regiões com papéis específicos (Figura 3). A região codificadora contém a informação necessária para a síntese de RNA, e seu nucleotídeo inicial é denominado +1. A região promotora antecede a região codificadora, sendo responsável por controlar o acesso de uma enzima chamada RNA polimerase (RNAP), dando início ao processo de transcrição. A região de terminação sucede a região codificadora, e é responsável por finalizar o processo de transcrição (SANDERS; BOWMAN, 2014).

Figura 3 – Regiões de um gene.

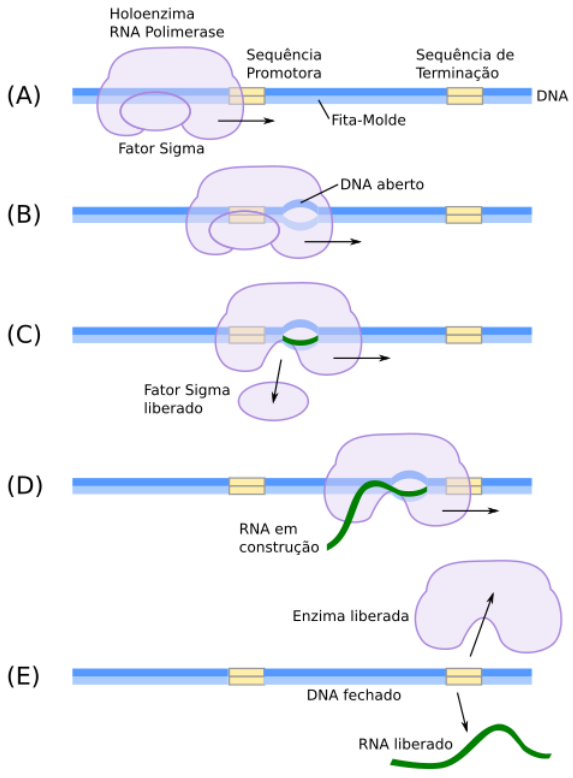


Fonte: Sanders & Bowman (2014)

Como mencionado anteriormente, em bactérias, o processo de transcrição tem a função de sintetizar os diferentes tipos de RNA a partir de uma fita-molde do DNA. Uma enzima específica, chamada RNA polimerase, também conhecida pela sigla RNAP, é responsável por esse processo. A RNAP é composta por uma enzima principal, que é ativada ao ligar-se com um polipeptídeo denominado fator sigma. Em sua forma ativa, recebe o nome de holoenzima RNA polimerase (SANDERS; BOWMAN, 2014).

No processo de transcrição (Figura 4), a holoenzima RNA polimerase se liga ao DNA para realizar a leitura do mesmo (A). A holoenzima abre a fita-dupla do DNA, e usa uma delas como fita-molde para a construção do RNA (B). A holoenzima desliza sobre a fita até encontrar uma sequência promotora, causando a separação do fator sigma e iniciando a construção do RNA (C). A enzima prossegue com a construção do RNA, até encontrar uma sequência de terminação (D). Nesse ponto, a enzima se desliga do DNA, liberando o RNA e fazendo com que o DNA se feche novamente (E) (SANDERS; BOWMAN, 2014).

Figura 4 – Processo de transcrição.

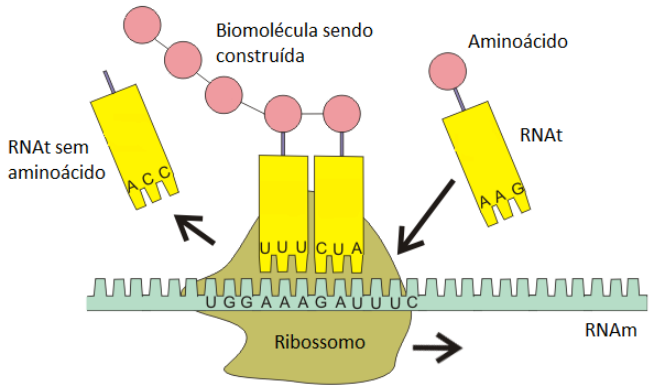


Fonte: Sanders & Bowman (2014)

Em bactérias, diferentes tipos de fatores sigma causam mudanças de conformação na RNAP, permitindo que ela se ligue a diferentes sequências promotoras (SANDERS; BOWMAN, 2014). Os principais fatores sigma conhecidos são $\sigma 24$, $\sigma 28$, $\sigma 32$, $\sigma 38$, $\sigma 54$ e $\sigma 70$, sendo que estes são rotulados a partir de seu peso molecular. Cada fator sigma possui um papel geral e faz o reconhecimento de uma sequência promotora específica (SILVA, 2011).

O processo de tradução (Figura 5), como mencionado anteriormente, é responsável pela síntese de biomoléculas dentro da célula, garantindo o correto funcionamento de diversos processos presentes em células, tecidos e órgãos. Nesse processo o ribossomo se liga ao RNAm para efetuar a leitura, enquanto diversos RNAts transportam aminoácidos para a construção de uma cadeia. O RNAt é ejetado após o aminoácido fazer sua ligação, e o ciclo continua até que a cadeia esteja completamente construída (CARVALHO; RECCO-PIMENTEL, 2013).

Figura 5 – Processo de tradução.



Fonte: Bonora *et al.* (2016)

Alterações na região promotora podem impedir que a RNAP se ligue ao DNA, afetando o processo de transcrição e fazendo com que a produção de um determinado RNA não aconteça. Com a produção do RNA afetada, biomoléculas importantes podem deixar de ser produzidas, ou características celulares podem deixar de ser expressadas. Logo, o promotor possui papel regulador na expressão gênica nos organismos vivos (CARVALHO; RECCO-PIMENTEL, 2013). De fato, diversos estudos observaram a relação entre alterações em promotores e várias doenças, incluindo Talassemia Intermédia (KULOZIK *et al.*, 1991), Síndrome de Rubinstein-Taybi (PETRIF *et al.*, 1995), alergias (HOBBS *et al.*, 1998), asma (BURCHARD *et al.*, 1999) e diabetes (IONESCU-TIRGOVISTE *et al.*, 2015). A identificação de promotores em sequências de DNA é, portanto, de grande interesse para a comunidade científica, pois isso pode ajudar no estudo, diagnóstico e tratamento de doenças ocasionadas por alterações em regiões promotoras.

3 IDENTIFICAÇÃO DE REGIÕES PROMOTORAS

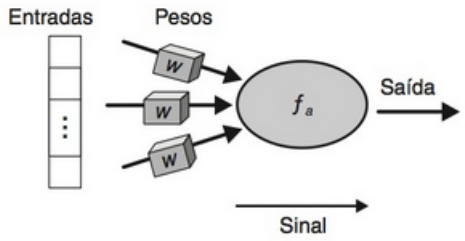
Dada a importância dos promotores no processo de transcrição, a identificação dessas regiões em sequências de DNA é de grande interesse para a comunidade científica. Diversos métodos biotecnológicos podem ser usados para este fim, como por exemplo o *Full Length cDNA Sequencing*, o *Expressed Sequence Tags (ESTs)* e o *Cap Analysis of Gene Expression (CAGE)* (JACOBI, 2014). Porém, esses métodos geralmente não são adequados para serem usados em larga escala, devido principalmente ao tempo e ao custo desses métodos. Para realizar essa identificação em tempo menor e com um custo mais baixo, podem ser utilizados métodos computacionais. Esses frequentemente são baseados em algoritmos de aprendizado de máquina, sendo que os métodos mais utilizados são Redes Neurais Artificiais, K Vizinhos Mais Próximos e Máquinas de Vetores de Suporte (FACELI *et al.*, 2011).

3.1 REDES NEURAIS ARTIFICIAIS

O cérebro humano possui a capacidade de processar informações dos mais variados tipos e de realizar diversas tarefas de forma simultânea. Desta forma, é natural o interesse científico em simular o seu funcionamento. Com essa motivação foram criadas as Redes Neurais Artificiais (RNs), que são redes de neurônios artificiais simulados computacionalmente, com capacidade de aprendizado, e que podem ser utilizadas na resolução de problemas complexos (FACELI *et al.*, 2011).

Os neurônios artificiais (Figura 6) são a unidade básica de uma RN. Cada neurônio possui um ou mais canais de entrada e um ou mais canais de saída. Os valores recebidos como entrada são ponderados através de pesos e funções matemáticas, e o resultado desses cálculos serve como o valor de saída do neurônio (FACELI *et al.*, 2011).

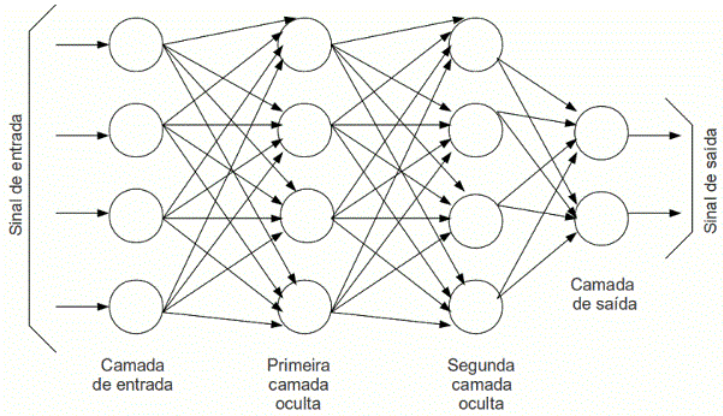
Figura 6 – Estrutura de um neurônio artificial.



Fonte: Faceli *et al.* (2011)

Por si só, um neurônio é capaz de realizar apenas tarefas simples de classificação. O poder das RNs é obtido pela conexão de vários neurônios entre si, normalmente dispostos em camadas. Cada camada de neurônios se comunica com a próxima através de seus canais de entrada e saída, sendo que a saída de um neurônio funciona como a entrada de outro. Diferentes padrões de conexões, camadas e algoritmos de aprendizado podem ser usados, gerando arquiteturas com propósitos e capacidades variados (LUGER, 2013). A Figura 7 apresenta um exemplo de RN em camadas, composta por uma camada de entrada, duas camadas ocultas e uma camada de saída. As camadas ocultas são assim denominadas pois não são diretamente acessíveis ou visíveis de fora da rede (LUGER, 2013).

Figura 7 – Estrutura de uma RN em camadas.



Fonte: Volpi (2015)

Uma RN pode ”aprender” a separar classes usando como base dados previamente classificados, em uma técnica conhecida como aprendizado supervisionado. Nela, os dados de treinamento são utilizados para ajustar os pesos dos canais de entrada e saída dos neurônios, com base em erros de classificação. Após a fase de treinamento, quando os erros de classificação chegaram a níveis aceitáveis, a rede pode ser usada para a classificação de dados que não foram utilizados na fase de treinamento (FACELI *et al.*, 2011) (LUGER, 2013).

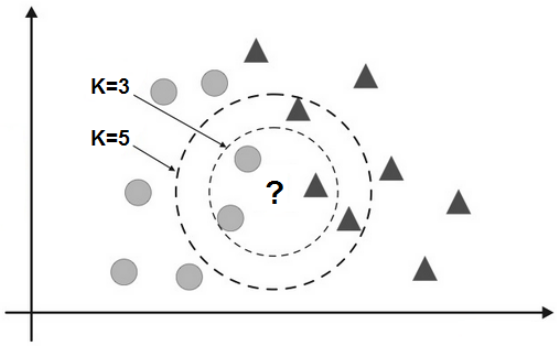
3.2 K VIZINHOS MAIS PRÓXIMOS

O algoritmo K Vizinhos Mais Próximos (do inglês K *Nearest Neighbours*, ou KNN) está entre os mais utilizados na área de reconhecimento de padrões, tendo em vista que sua especificação é simples e intuitiva. Este é um algoritmo baseado em distâncias, e parte da suposição de que dados com características similares tendem a estar espacialmente próximos. Dessa forma, um dado pode ser classificado com base em exemplos previamente classificados (FACELI *et al.*, 2011) (WEBB; COPSEY, 2011).

O KNN é considerado um algoritmo *lazy* (do inglês, "preguiçoso"), por não inferir um modelo de classificação para os dados. A sua fase de treinamento consiste apenas em carregar amostras previamente classificadas em memória. Na fase de classificação é calculada a distância entre o dado a ser classificado e todas as amostras do modelo. A classe que possuir mais incidência entre as K amostras mais próximas será atribuída àquele dado (FACELI *et al.*, 2011).

Por exemplo, a Figura 8 apresenta amostras pertencentes a duas classes distintas (círculos e triângulos) e um dado a ser classificado (interrogação). Utilizando-se o valor $K = 3$, seriam consideradas as três amostras mais próximas ao dado, e este seria atribuído à classe "círculo". Utilizando-se o valor $K = 5$, seriam consideradas as cinco amostras mais próximas ao dado, e este seria atribuído à classe "triângulo". A definição do valor de K não é trivial e possui grande importância, visto que valores diferentes de K podem resultar em classificações diferentes (FACELI *et al.*, 2011).

Figura 8 – Classificação através do algoritmo KNN.



Fonte: Faceli *et al.* (2011)

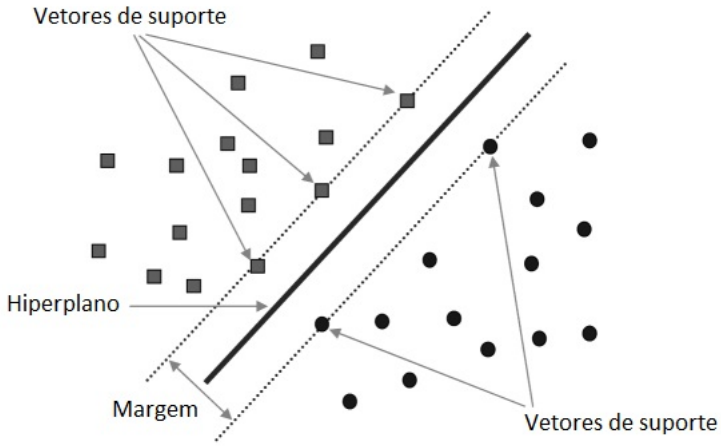
3.3 MÁQUINAS DE VETORES DE SUPORTE

A aplicação de Máquinas de Vetores de Suporte (do inglês *Support Vector Machines*, ou SVMs) vem recebendo grande atenção da comunidade de aprendizado de máquina (FACELI *et al.*, 2011). De fato, os resultados obtidos pela aplicação de SVMs na área de bioinformática são comparáveis aos obtidos com Redes Neurais Artificiais (MCQUISTEN; PEEK, 2009) (SHARMA *et al.*, 2011).

As SVMs ”aprendem” a separar dados em classes através da técnica de aprendizado supervisionado. Através dessa técnica, os dados de treinamento são mapeados para um espaço de características de dimensão mais alta, onde é construído um hiperplano que maximize a margem de separação das classes. Após a obtenção do hiperplano, esse pode ser usado para classificação de dados, observando-se apenas em qual lado do hiperplano encontra-se um determinado dado. Ou seja, dados de uma classe estarão em um lado do hiperplano, enquanto os dados de outra classe estarão no outro lado (WEBB; COPSEY, 2011).

A Figura 9 mostra um exemplo de SVM sendo usada para classificar um grupo de dados em duas classes distintas (quadrados e círculos). O hiperplano obtido pelo algoritmo (linha escura) apresenta a maior margem de separação dos dados (linhas pontilhadas). Os dados ao longo das linhas pontilhadas são os vetores de suporte, e são os dados mais importantes do grupo, pois servem como base para a criação do hiperplano (FACELI *et al.*, 2011).

Figura 9 – Máquina de Vetores de Suporte.



Fonte: Chen *et al.* (2009)

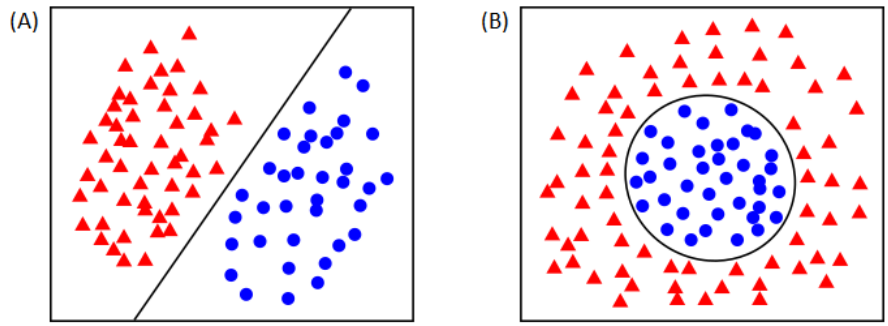
4 MÁQUINAS DE VETORES DE SUPORTE

As Máquinas de Vetores de Suporte são algoritmos de classificação inicialmente formulados por Vladimir Vapnik, em 1963. Em 1971, juntamente com Alexey Chervonenkis, Vapnik continuou a aperfeiçoar essa teoria, que em 1995 ficou conhecida como Teoria do Aprendizado Estatístico (KOWALCZYK, 2017).

As SVMs foram desenvolvidas para classificação binária, ou seja, a separação de um conjunto de dados em duas classes distintas. Por exemplo, tendo-se um conjunto de dados X, deseja-se atribuir a cada dado $\vec{x_i}$ um rótulo y_i, sendo que amostras da primeira classe possuirão o rótulo $y_i = 1$ e amostras da segunda classe possuirão o rótulo $y_i = -1$. Para tanto, deve-se construir algum tipo de fronteira de decisão que separe os dados, buscando minimizar os erros de classificação. Esses erros ocorrem quando dados pertencentes à primeira classe são incorretamente rotulados como sendo da segunda classe, ou vice-versa (FACELI *et al.*, 2011) (KOWALCZYK, 2017).

As SVMs tradicionais, conhecidas como SVMs lineares, permitem apenas a classificação de dados linearmente separáveis, enquanto que as SVMs não lineares possibilitam a separação de dados não linearmente separáveis. Os dados são linearmente separáveis se, em duas dimensões, estes podem ser separados por uma reta; em três dimensões, podem ser separados por um plano; e em quatro ou mais dimensões, podem ser separados por um hiperplano (FACELI *et al.*, 2011). Na Figura 10 são apresentados exemplos de dados em duas dimensões. Na Figura 10 (A) os dados são linearmente separáveis, pois as duas classes podem ser separadas por uma reta. Na Figura 10 (B) os dados não são linearmente separáveis, pois as duas classes não podem ser separadas por uma reta (KOWALCZYK., 2014).

Figura 10 – Separabilidade linear de dados.



Fonte: Kowalczyk. (2014)

4.1 HIPERPLANO

As SVMs buscam uma fronteira de decisão que possibilite a separação de duas classes. Como deseja-se construir um método capaz de trabalhar com dados em qualquer número de dimensões, um hiperplano pode ser utilizado como fronteira de decisão (FACELI *et al.*, 2011).

No hiperplano H_0 representado pela Equação 4.1, $\vec{w} \cdot \vec{x}$ é o produto escalar dos vetores $\vec{w}$ e $\vec{x}$, onde $\vec{w}$ é um vetor perpendicular ao hiperplano, $\vec{x}$ é um vetor de dados, e b é um valor escalar. Sendo d a distância entre o hiperplano e a origem, b obedece a relação representada pela Equação 4.2, onde $\|\vec{w}\|$ é a norma[1] do vetor $\vec{w}$ (FACELI *et al.*, 2011).

$$H_0 : \vec{w} \cdot \vec{x} + b = 0 \tag{4.1}$$

$$d = \frac{b}{\|\vec{w}\|} \tag{4.2}$$

Partindo-se da Equação 4.1, é possível chegar a uma fronteira de decisão em qualquer número de dimensões. Por exemplo, é possível obter a equação da reta representada pela Equação 4.3, que pode ser utilizada para separar um conjunto de dados em duas dimensões (KOWALCZYK, 2017).

$$y = ax + c \tag{4.3}$$

Em duas dimensões, pode-se considerar que $\vec{w}$ e $\vec{x}$ são vetores de dois elementos, onde $\vec{w} = (w_0, w_1)$ e $\vec{x} = (x, y)$. Substituindo-se $\vec{w}$ e $\vec{x}$ na Equação 4.1 e isolando a variável y, tem-se a Equação 4.4. Considerando as variáveis $a = -\frac{w_0}{w_1}$ e $c = -\frac{b}{w_1}$, é possível substituir as frações da Equação 4.4 para obter a equação da reta (Equação 4.3) (KOWALCZYK, 2017).

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + b = 0 \quad \Rightarrow \quad w_0 x + w_1 y + b = 0 \quad \Rightarrow \quad y = -\frac{w_0}{w_1} x - \frac{b}{w_1} \tag{4.4}$$

Os mesmos passos podem ser realizados em qualquer dimensão, bastando alterar o tamanho dos vetores $\vec{w}$ e $\vec{x}$. Por exemplo, para se obter um plano de separação para três dimensões, podem ser utilizados os vetores $\vec{w} = (w_0, w_1, w_2)$ e $\vec{x} = (x, y, z)$. Como um hiperplano permite a separação de dados em qualquer dimensão, as SVMs implementam hiperplanos como fronteiras de decisão (KOWALCZYK, 2017).

[1] Raiz quadrada do produto escalar do vetor com relação a ele mesmo

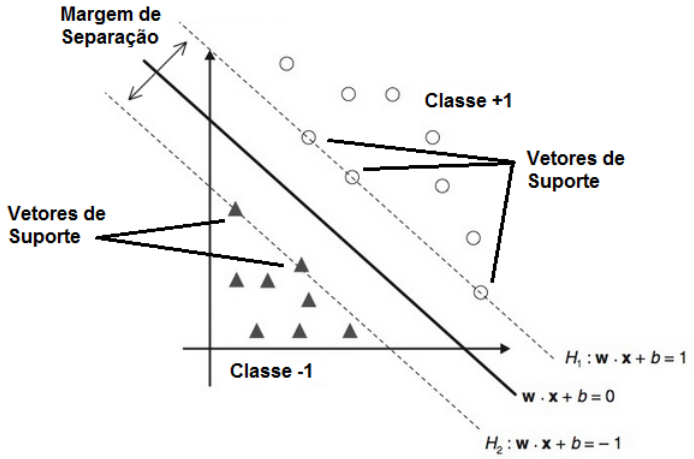
4.2 MARGEM DE SEPARAÇÃO

A partir da Equação 4.1, é possível obter infinitos hiperplanos paralelos a H_0, bastando atribuir diferentes valores ao lado direito da igualdade. Assim, é possível definir dois hiperplanos, H_1 e H_2, que encontram-se representados na Equação 4.5 (FACELI *et al.*, 2011).

$$\begin{aligned} H_1 &: \vec{w} \cdot \vec{x} + b = 1 \\ H_2 &: \vec{w} \cdot \vec{x} + b = -1 \end{aligned} \tag{4.5}$$

Como pode ser observado na Figura 11, H_1 e H_2 são equidistantes a H_0, e também são capazes de separar as duas classes. Os dados posicionados exatamente sobre H_1 e H_2 são denominados vetores de suporte e a distância entre esses dois hiperplanos é denominada margem de separação (FACELI *et al.*, 2011).

Figura 11 – Margem de separação.



Fonte: Faceli *et al.* (2011)

4.2.1 Restrições para a Definição dos Hiperplanos

A definição dos hiperplanos H_1 e H_2 deve assegurar que não existam pontos dentro da margem de separação, de forma a minimizar os erros de classificação. Para isso, a restrição representada na Equação 4.6 deve ser respeitada para todos os pontos pertencentes à classe $y = 1$. Da mesma forma, a restrição representada na Equação 4.7 deve ser respeitada para todos os pontos pertencentes à classe $y = -1$ (FACELI *et al.*, 2011).

$$\vec{w} \cdot \vec{x} + b \geq 1 \tag{4.6}$$

$$\vec{w} \cdot \vec{x} + b \leq -1 \tag{4.7}$$

A partir das Equações 4.6 e 4.7, é possível obter uma única equação que inclua ambas as restrições. Para isso, basta multiplicar ambas as equações pela variável y utilizada como rótulo das classes, obtendo-se as Equações 4.8 e 4.9, respectivamente (FACELI *et al.*, 2011).

$$y(\vec{w} \cdot \vec{x} + b) \geq y(1) \tag{4.8}$$

$$y(\vec{w} \cdot \vec{x} + b) \geq y(-1) \tag{4.9}$$

Sabe-se que todos os dados que obedecem à restrição da Equação 4.8 pertencem à classe $y = 1$. Da mesma forma, sabe-se que todos os dados que obedecem à restrição da Equação 4.9 pertencem à classe $y = -1$. Assim, é possível substituir a variável y pelo seu respectivo valor em ambas as equações, obtendo-se as Equações 4.10 e 4.11 (FACELI *et al.*, 2011).

$$y(\vec{w} \cdot \vec{x} + b) \geq (1)(1) \quad \Rightarrow \quad y(\vec{w} \cdot \vec{x} + b) \geq 1 \tag{4.10}$$

$$y(\vec{w} \cdot \vec{x} + b) \geq (-1)(-1) \quad \Rightarrow \quad y(\vec{w} \cdot \vec{x} + b) \geq 1 \tag{4.11}$$

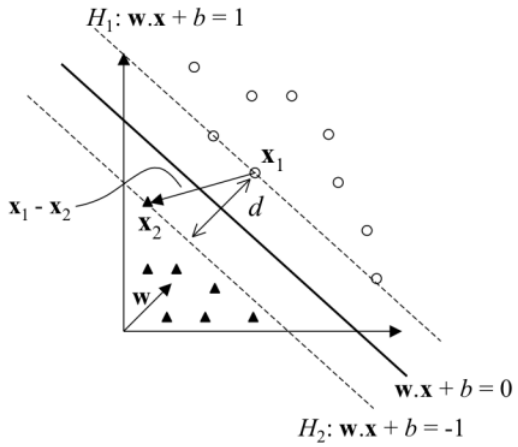
Como as Equações 4.10 e 4.11 são iguais, pode-se assumir que a Equação 4.12 pode ser usada como restrição para ambas as classes (FACELI *et al.*, 2011).

$$y(\vec{w} \cdot \vec{x} + b) - 1 \geq 0 \tag{4.12}$$

4.2.2 Cálculo da Margem de Separação

Um dos objetivos do método SVM é encontrar a maior margem de separação possível entre as classes. Para isso, faz-se necessário calcular o tamanho dessa margem. Isso é equivalente a calcular a distância entre os hiperplanos H_1 e H_2, ou ainda, a calcular a distância d entre dois pontos, um em cada hiperplano, conforme a Figura 12 (LORENA; CARVALHO, 2007).

Figura 12 – Distância entre os hiperplanos H_1 e H_2.



Fonte: Lorena & Carvalho (2007)

Partindo da Equação 4.5, é possível isolar o vetor $\vec{x}$ para obter a Equação 4.13 (LORENA; CARVALHO, 2007).

$$\begin{aligned} H_1 : \vec{x_1} &= \frac{1-b}{\vec{w}} \\ H_2 : \vec{x_2} &= \frac{-1-b}{\vec{w}} \end{aligned} \tag{4.13}$$

A distância euclideana entre dois vetores pode ser calculada pela norma da diferença entre esses vetores, conforme a Equação 4.14 (SANTOS; FERREIRA, 2009).

$$d = \|\vec{x_1} - \vec{x_2}\| \tag{4.14}$$

Substituindo-se os valores de $\vec{x_1}$ e $\vec{x_2}$ na Equação 4.14, obtém-se o tamanho da margem de separação (Equação 4.15) (LORENA; CARVALHO, 2007).

$$d = \left\| \frac{1-b}{\vec{w}} - \frac{-1-b}{\vec{w}} \right\| \quad \Rightarrow \quad d = \left\| \frac{2}{\vec{w}} \right\| \quad \Rightarrow \quad d = \frac{2}{\|\vec{w}\|} \tag{4.15}$$

Percebe-se que quanto maior o valor de $\|\vec{w}\|$, menor será o resultado da divisão, e menor será a margem de separação. Logo, a maximização da margem de separação pode ser obtida pela minimização de $\|\vec{w}\|$ (LORENA; CARVALHO, 2007).

4.3 MAXIMIZAÇÃO DA MARGEM DE SEPARAÇÃO

A definição dos hiperplanos H_1 e H_2 deve obedecer às restrições da Equação 4.12, e a maximização da margem de separação pode ser obtida pela minimização de $\|\vec{w}\|$. A combinação desses dois conceitos resulta no problema de otimização apresentado na Equação 4.16 (LORENA; CARVALHO, 2007).

$$\begin{gathered} \text{Minimizar: } \|\vec{w}\| \\ \text{Com as restrições: } y(\vec{w} \cdot \vec{x} + b) - 1 \geq 0 \end{gathered} \tag{4.16}$$

Uma vez que a solução deste problema de otimização envolverá a derivação de funções, é conveniente expressar $\|\vec{w}\|$ em sua forma integral (Equação 4.17). Uma vez que a derivada de $\frac{1}{2}\|\vec{w}\|^2$ é igual a $\|\vec{w}\|$, tem-se que a lógica inicial do problema é mantida (LORENA; CARVALHO, 2007).

$$\int \|\vec{w}\| \, dw = \frac{1}{2}\|\vec{w}\|^2 \tag{4.17}$$

Dessa forma, é obtido o problema de otimização da Equação 4.18 (LORENA; CARVALHO, 2007).

$$\begin{gathered} \text{Minimizar: } \frac{1}{2}\|\vec{w}\|^2 \\ \text{Com as restrições: } y(\vec{w} \cdot \vec{x} + b) - 1 \geq 0 \end{gathered} \tag{4.18}$$

Para resolver esse problema, é possível utilizar o método de Multiplicadores de Lagrange, desenvolvido pelo matemático Joseph-Louis Lagrange, representado na Equação 4.19. Esse método é usado para encontrar um máximo ou mínimo local de uma função $f(\vec{x})$, sujeita às restrições $g(\vec{x_i})$. As variáveis α_i são denominadas Multiplicadores de Lagrange e representam a influência de cada restrição no problema como um todo (KOWALCZYK, 2017).

$$\mathcal{L}(\vec{x}, \vec{\alpha}) = f(\vec{x}) - \sum_{i=1}^{n} \alpha_i g(\vec{x_i}) \tag{4.19}$$

Substituindo as funções $f(\vec{x})$ e $g(\vec{x})$ pelas funções do problema de otimização (Equação 4.18), obtém-se a Equação 4.20, denominada Forma Primal (KOWALCZYK, 2017).

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\vec{w} \cdot \vec{x_i} + b) - 1) \tag{4.20}$$

A Forma Primal pode ser resolvida analiticamente, mas apenas para um conjunto de dados pequeno. Para conjuntos de dados maiores, é necessário reescrever a Forma Primal utilizando o Princípio da Dualidade, obtendo-se a Forma Dual. Para isso, primeiramente é necessário derivar a Equação 4.20 em relação a $\vec{w}$, resultando na Equação 4.21 (KOWALCZYK, 2017).

$$\frac{\partial \mathcal{L}}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} \alpha_i y_i \vec{x_i} = 0 \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \tag{4.21}$$

Substituindo $\vec{w}$ na Equação 4.20, é possível obter a Equação 4.22 (KOWALCZYK, 2017).

$$\begin{aligned}
\mathcal{L}(b, \vec{\alpha}) &= \frac{1}{2}(\sum_{i=1}^{m} \alpha_i y_i \vec{x_i}) \cdot (\sum_{j=1}^{m} \alpha_j y_j \vec{x_j}) - \sum_{i=1}^{n} \alpha_i (y_i((\sum_{j=1}^{m} \alpha_j y_j \vec{x_j}) \cdot \vec{x_i} + b) - 1) \Rightarrow \\
\mathcal{L}(b, \vec{\alpha}) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) - \sum_{i=1}^{m} \alpha_i y_i ((\sum_{j=1}^{m} \alpha_j y_j \vec{x_j}) \cdot \vec{x_i} + b) + \sum_{i=1}^{m} \alpha_i \Rightarrow \\
\mathcal{L}(b, \vec{\alpha}) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) - b \sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \Rightarrow \\
\mathcal{L}(b, \vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) - b \sum_{i=1}^{m} \alpha_i y_i
\end{aligned} \tag{4.22}$$

Derivando-se a Equação 4.20 em relação a b, tem-se a Equação 4.23 (KOWALCZYK, 2017).

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{4.23}$$

Observa-se que a Equação 4.23 é igual ao último termo da Equação 4.22 e igual a zero. Assim, é possível remover esse último termo para obter a Forma Dual representada na Equação 4.24 (KOWALCZYK, 2017).

$$\mathcal{L}(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \tag{4.24}$$

A Forma Dual implica no problema de otimização da Equação 4.25. A grande vantagem da Forma Dual em relação à Forma Primal é que a resolução do problema depende apenas dos Multiplicadores de Lagrange (α_i), o que simplifica o problema (KOWALCZYK, 2017).

$$\begin{aligned} &\text{Maximizar: } \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \\ &\text{Com as restrições: } \begin{cases} \alpha_i \geq 0 \\ \sum_{i=1}^{m} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{4.25}$$

Solucionar o problema apresentado na Equação 4.25 envolve encontrar os valores ótimos para os Multiplicadores de Lagrange (α_i). Atualmente, o método mais utilizado para este propósito é o SMO (*Sequential Minimal Optimization*), desenvolvido pelo cientista da computação John Carlton Platt, em 1998 (PLATT, 1998). Este algoritmo otimiza dois dos multiplicadores por vez, sucessivamente, até que a solução ótima seja encontrada para todos (WEBB; COPSEY, 2011).

Após a solução ótima ser encontrada para todos os Multiplicadores de Lagrange, $\vec{w}$ pode ser encontrado a partir da Equação 4.21. O valor de b pode ser encontrado a partir das condições KKT (Karush-Kuhn-Tucker), desenvolvidas pelos matemáticos William Karush, Harold William Kuhn e Albert William Tucker entre os anos 1939 e 1951 (KARUSH, 1939) (KUHN; TUCKER, 1951). Mais especificamente, calcular a solução ótima de b exige que a condição da Equação 4.26 seja respeitada para todos os vetores de suporte (KOWALCZYK, 2017).

$$\alpha_i (y_i (\vec{w} \cdot \vec{x_i} + b) - 1) = 0 \tag{4.26}$$

Isolando-se b na Equação 4.26, obtém-se a Equação 4.27 (KOWALCZYK, 2017).

$$\begin{aligned} \alpha_i (y_i (\vec{w} \cdot \vec{x_i}) + y_i b - 1) &= 0 \\ \alpha_i y_i (\vec{w} \cdot \vec{x_i}) + \alpha_i y_i b - \alpha_i &= 0 \\ \alpha_i y_i b &= \alpha_i - \alpha_i y_i (\vec{w} \cdot \vec{x_i}) \\ b &= \frac{\alpha_i}{\alpha_i y_i} - \frac{\alpha_i y_i (\vec{w} \cdot \vec{x_i})}{\alpha_i y_i} \\ b &= \frac{1}{y_i} - \vec{w} \cdot \vec{x_i} \end{aligned} \tag{4.27}$$

A aplicação da Equação 4.27 para todos os vetores de suporte é equivalente a calcular a média do valor de b para cada vetor de suporte. Essa média pode ser calculada através da Equação 4.28, onde S é o número de vetores de suporte encontrados (KOWALCZYK, 2017).

$$b = \frac{1}{S} \sum_{i=1}^{S} \frac{1}{y_i} - \vec{w} \cdot \vec{x_i} \tag{4.28}$$

Após o cálculo de $\vec{w}$ e b, o hiperplano de separação ótimo está definido. A partir deste hiperplano ótimo, amostras $\vec{x_j}$ podem ser classificadas observando-se o sinal de y_j retornado pela Equação 4.29 (KOWALCZYK, 2017).

$$y_j = \vec{w} \cdot \vec{x_j} + b \tag{4.29}$$

É necessário que a Equação 4.29 retorne apenas rótulos válidos. Como se trata de uma classificação binária, apenas os valores $+1$ e -1 podem ser retornados. Para tanto, pode ser utilizada uma função sinal sgn, conforme a Equação 4.30 (KOWALCZYK, 2017).

$$y_j = sgn(\vec{w} \cdot \vec{x_j} + b) \tag{4.30}$$

Substituindo-se $\vec{w}$ pela Equação 4.21, a classificação passa a depender dos vetores de suporte $\vec{x_i}$ em vez do vetor $\vec{w}$, o que simplifica os cálculos a serem realizados. Dessa forma, obtém-se o classificador linear final representado na Equação 4.31 (KOWALCZYK, 2017).

$$y_j = sgn(\sum_{i=1}^{m} \alpha_i y_i (\vec{x_i} \cdot \vec{x_j}) + b) \tag{4.31}$$

4.4 MÁQUINAS DE VETORES DE SUPORTE NÃO LINEARES

O classificador apresentado na Equação 4.31 é eficaz somente para dados linearmente separáveis. Para classificar dados que não possuem essa característica, faz-se necessário a utilização de funções *kernel*. Essas funções são aplicadas aos dados de entrada, transformando o espaço de características de forma que um classificador linear possa ser aplicado. Partindo da Equação 4.31, uma função *kernel* $\mathcal{K}$ pode ser aplicada ao produto escalar $(\vec{x_i} \cdot \vec{x_j})$, dando origem à Equação 4.32 (FACELI *et al.*, 2011).

$$y_j = sgn(\sum_{i=1}^{m} \alpha_i y_i \mathcal{K}(\vec{x_i} \cdot \vec{x_j}) + b) \tag{4.32}$$

Os *kernels* mais utilizados são o linear, o polinomial, o RBF (do inglês *Radial Basis Function*, ou Função de Base Radial) e o sigmoidal (FACELI *et al.*, 2011).

4.4.1 Kernel Linear

A Equação 4.33 apresenta o *kernel* linear. A partir da substituição deste na Equação 4.32, reverte-se o classificador à sua forma linear (Equação 4.31), com funcionamento idêntico. Logo, SVMs não lineares podem ser utilizadas também para dados linearmente separáveis (KOWALCZYK, 2017).

$$\mathcal{K}(\vec{x_i} \cdot \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j}) \tag{4.33}$$

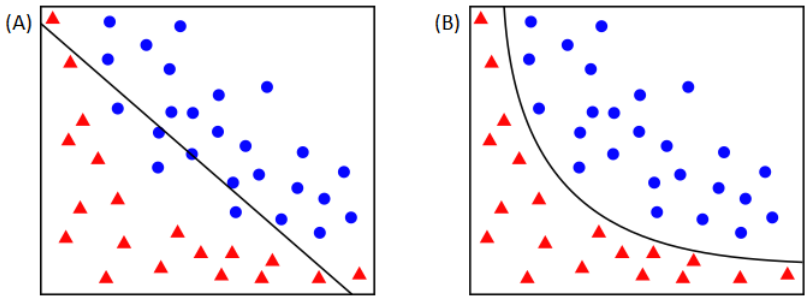
4.4.2 Kernel Polinomial

O *kernel* polinomial é apresentado na Equação 4.34, onde c e d são parâmetros livres. O parâmetro c é uma variável de controle, enquanto o parâmetro d representa o grau do polinômio a ser utilizado. Este *kernel* pode ser utilizado de forma linear aplicando-se os valores $c = 0$ e $d = 1$ (KOWALCZYK, 2017).

$$\mathcal{K}(\vec{x_i} \cdot \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + c)^d \tag{4.34}$$

A Figura 13 (A) apresenta um conjunto de dados não linearmente separáveis. Com a utilização de um *kernel* polinomial de grau 2, os dados podem ser separados conforme a Figura 13 (B) (KOWALCZYK, 2017).

Figura 13 – Kernel polinomial.



Fonte: Kowalczyk (2017)

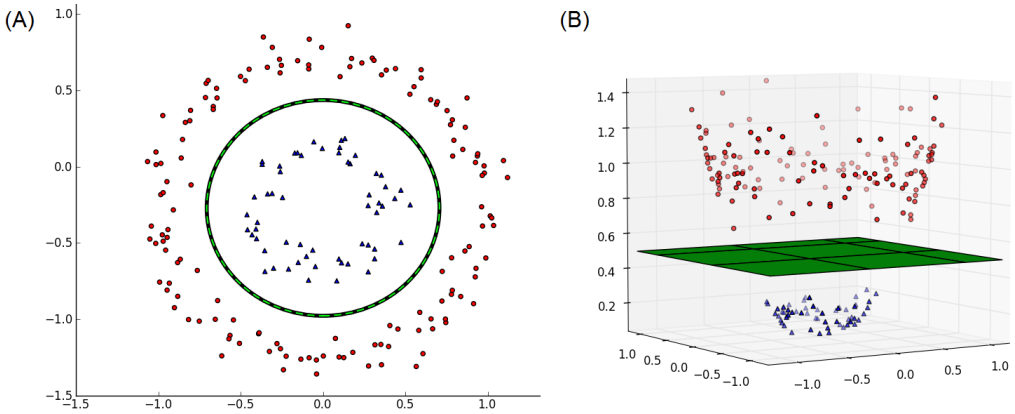
4.4.3 Kernel de Função de Base Radial

O *kernel Radial Basis Function* (Função de Base Radial, ou RBF), também conhecido como Gaussiano, mapeia as entradas para um espaço de características com maior dimensão. A Equação 4.34 apresenta esse *kernel*, onde γ é um parâmetro de controle (KOWALCZYK, 2017).

$$\mathcal{K}(\vec{x_i} \cdot \vec{x_j}) = exp(-\gamma \|\vec{x_i} \cdot \vec{x_j}\|^2) \tag{4.35}$$

O conjunto de dados apresentado na Figura 14 (A) não pode ser linearmente separado em duas dimensões. Com a projeção desses dados em um espaço de três dimensões, através do *kernel* RBF, um hiperplano pode ser utilizado para separar os dados, conforme a Figura 14 (B) (KOWALCZYK, 2017).

Figura 14 – Kernel RBF.



Fonte: Kim (2017)

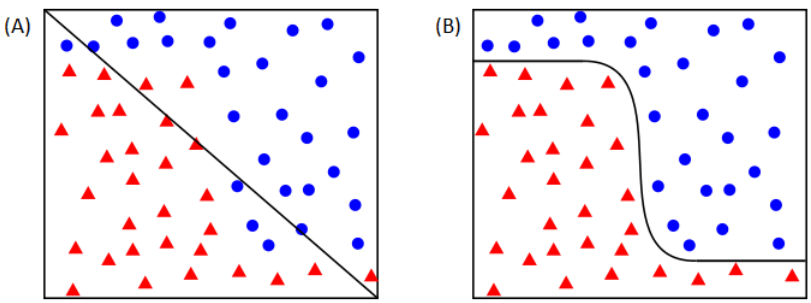
4.4.4 Kernel Sigmoidal

O *kernel* sigmoidal utiliza uma função homônima para transformar o espaço de características, conforme a Equação 4.34, onde c é um parâmetro de controle (FACELI *et al.*, 2011).

$$\mathcal{K}(\vec{x_i} \cdot \vec{x_j}) = tanh(\vec{x_i} \cdot \vec{x_j} + c) \tag{4.36}$$

A Figura 15 (A) apresenta dados não linearmente separáveis. Através de uma função sigmoide, esses dados podem ser separados conforme a Figura 15 (B). A principal diferença entre o *kernel* sigmoide e o polinomial é que o *kernel* sigmoide permite a separação de dados em linhas quase retas. Para atingir um efeito similar, seria necessário um *kernel* polinomial de grau elevado, o que tornaria os cálculos mais complexos (KOWALCZYK, 2017).

Figura 15 – Kernel sigmoidal.



Fonte: Kowalczyk (2017)

5 IMPLEMENTAÇÃO E RESULTADOS

Neste trabalho foi desenvolvida uma implementação para a identificação de regiões promotoras em sequências de DNA da bactéria *Escherichia coli*. Cada amostra é rotulada entre "possui região promotora" e "não possui região promotora", através do método SVM. Nas próximas seções é descrita a implementação desenvolvida, bem como os testes realizados e os resultados obtidos.

5.1 BIBLIOTECA SVM

Para este desenvolvimento, foram analisadas as principais bibliotecas disponíveis que contêm implementações do método SVM. Como requisitos principais para a escolha da biblioteca, utilizou-se:

- Estar disponível sob uma licença de código aberto, ou seja, deve ser gratuita e possibilitar sua livre alteração e distribuição;
- Ser multi-plataforma, ou seja, deve possibilitar seu uso em diferentes ambientes ou sistemas operacionais;
- Possuir suporte aos *kernels* Linear, Polinomial, RBF e Sigmoidal.

As bibliotecas analisadas foram: HeroSVM (versão 2.1), desenvolvida por Dong, Krzyzak & Suen (2002); LibSVM (versão 3.24), desenvolvida por Chang & Lin (2011); mySVM (versão 2.1.4), desenvolvida por Ruping (2000); e SVMlight (versão 6.02), desenvolvida por Joachims (1999). A Tabela 1 apresenta um resumo das características dessas bibliotecas.

Tabela 1 – Requisitos da linguagem de programação.

	HeroSVM	LibSVM	mySVM	SVMlight
Linguagens	C++	Java Python	C++	C
Multi-Plataforma	Não	Sim	Não	Não
Código Aberto	Não	Sim	Sim	Sim
Kernels	Linear Polinomial RBF	Linear Polinomial RBF Sigmoidal	Linear Polinomial RBF	Linear Polinomial RBF Sigmoidal

Fontes: Dong, Krzyzak & Suen (2002), Chang & Lin (2011), Ruping (2000), Joachims (1999)

Entre as bibliotecas analisadas, optou-se pela LibSVM, pois é a única multi-plataforma, disponibilizada em código aberto e que possui todos os *kernels* desejados (Tabela 1). Entre as implementações disponíveis, optou-se pela versão implementada na linguagem de programação Java, visto que essa linguagem fornece um melhor desempenho se comparada à linguagem Python (FOURMENT; GILLINGS, 2008).

Cada *kernel* disponível na biblioteca LibSVM possui um conjunto de parâmetros a ser ajustado, conforme a Tabela 2. O parâmetro C corresponde a uma penalização por amostras classificadas incorretamente; quanto maior o valor, maior a penalização e maior a precisão; seu valor padrão é $C = 1$. O parâmetro $Gamma$ é utilizado para a controlar o formato das curvas do *kernel*; quanto maior o valor, maior a largura da curva e menor a precisão; seu valor padrão é $Gamma = \frac{1}{n}$, onde n é o número de atributos de cada amostra. O parâmetro $Coef0$ é utilizado para normalizar os dados de entrada para um intervalo de -1 até 1; seu valor padrão é $Coef0 = 0$. O parâmetro $Degree$ define o grau do polinômio no *kernel* polinomial; seu valor padrão é $Degree = 3$ (CHANG; LIN, 2011).

Tabela 2 – Parâmetros disponíveis em cada *kernel* da biblioteca LibSVM.

Kernel	Parâmetros Disponíveis
Linear	C
Polinomial	C, $Gamma$, $Coef0$ e $Degree$
RBF	C, $Gamma$ e $Coef0$
Sigmoidal	C, $Gamma$ e $Coef0$

Fonte: Chang & Lin (2011)

5.2 BASE DE DADOS

Para treinamento e validação da implementação desenvolvida, foram utilizados arquivos disponíveis na base RegulonDB (versão 10.6.3). Essa base disponibiliza, entre outros dados, sequências de DNA contendo regiões promotoras de bactérias da espécie *Escherichia coli*. As amostras são divididas em arquivos conforme o fator sigma que se liga àquela região promotora (SANTOS-ZAVALETA *et al.*, 2019).

Um exemplo de arquivo da base RegulonDB pode ser visualizado na Figura 16, sendo que as seis primeiras colunas do arquivo estão representadas na Figura 16 (A), e as duas últimas colunas estão representadas na Figura 16 (B). As seis colunas da Figura 16 (A) contém, respectivamente, as seguintes informações: identificador único do promotor na base RegulonDB, nome do promotor, fita do DNA em que o promotor se encontra, posição da região codificadora no DNA, fator sigma que se liga àquele promotor e sequência de DNA do promotor. As duas colunas da Figura 16 (B) contém, respectivamente, as seguintes informações: evidência que suporta a existência de um promotor naquela sequência de DNA, e o grau de confiança nessa evidência (SANTOS-ZAVALETA *et al.*, 2019).

Figura 16 – Organização de arquivos da base RegulonDB (versão 10.6.3).

(A)

ECK125136390	dxrp4	forward	193338	Sigma28	acgacaaagtgaaagcactgttgaaagataaagagatcagcgaagacgacgatcgccgttCtcaggacgatgtacagaaac
ECK125137290	ecop7	forward	2303851	Sigma28	ttgtctgaatgttctttaagttattcataaagcaaattaataaatctgatgaatatgttaAccttcagcgacatcatcggt
ECK120009892	flgKp	forward	1138351	Sigma28	gaacattgataatctgttctgaataactcaagtccggcgggtcgctgccgataatactctGtaattgaaggcttataagga
ECK120009891	flgMp	reverse	1130165	Sigma28	gaatattcttataaacctgtaagctgttaaagattacccgtcccttgccgataaataagcAacacatgataaaagcgccct
ECK125137156	flhCp3	reverse	1977932	Sigma28	ttgacagccaccagacgattactcagttgacgcaagattcccgcgttgacgatctccagcAaattcataccggcatcatgc
ECK120010568	fliAp2	reverse	2001808	Sigma28	acccactaatcgtccgattaaaaaccctgcagaaacggataatcatgccgataactcataTaacgcagggctgtttatcgt
ECK120009887	fliCp	reverse	2003676	Sigma28	aaaaatggctgtttttgaaaaaaattctaaaggttgttttacgacagacgataacagggtTgacggcgattgagccgacgg
ECK120029598	fliDp1	forward	0	Sigma28	

(B)

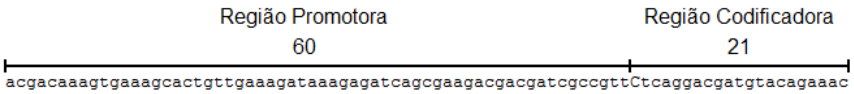
[ICWHO|W|Inferred computationally without human oversight]	Weak
[ICWHO|W|Inferred computationally without human oversight]	Weak
[HIPP|W|Human inference of promoter position]	Weak
[HIPP|W|Human inference of promoter position]	Weak
[ICWHO|W|Inferred computationally without human oversight]	Weak
[HIPP|W|Human inference of promoter position],[TIM|S|Transcription initiation mapping]	Strong
[HIPP|W|Human inference of promoter position],[FP|S|Footprinting]	Strong
[HIPP|W|Human inference of promoter position],[FP|S|Footprinting]	Strong

Fonte: Santos-Zavaleta *et al.* (2019)

Amostras que não contivessem uma sequência de DNA associada foram descartadas, como é o caso do promotor ECK120029598 apresentado na última linha da Figura 16 (A). Apesar de algumas amostras apresentarem múltiplos fatores sigma na coluna 5, foi considerado apenas o fator sigma principal do arquivo.

Na Figura 17, tem-se em destaque a sequência de DNA do promotor ECK125136390 apresentado na Figura 16 (A). Como pode ser observado, as sequências de DNA disponíveis possuem 81 nucleotídeos de largura, sendo que os primeiros 60 nucleotídeos fazem parte da região promotora, e os últimos 21 nucleotídeos fazem parte da região codificadora. O primeiro nucleotídeo da região codificadora encontra-se destacado em caixa-alta (SANTOS-ZAVALETA *et al.*, 2019).

Figura 17 – Organização de sequências de DNA da base RegulonDB (versão 10.6.3).



Fonte: Santos-Zavaleta *et al.* (2019)

Como amostras negativas, ou seja, que não possuem região promotora, foi utilizada a abordagem proposta por Gan, Guan & Zhou (2012), onde a sequência de DNA de cada amostra positiva é embaralhada de forma aleatória, gerando-se assim uma nova sequência não-promotora. Desta forma, são mantidas as proporções de nucleotídeos presentes em amostras reais (KANHERE; BANSAL, 2005). Neste trabalho, foi gerado um número de amostras negativas igual à quantidade presente em cada arquivo da base RegulonDB, de forma que cada conjunto de dados contivesse 50% de amostras positivas e 50% de amostras negativas. Caso alguma amostra gerada fosse idêntica a outra presente na base RegulonDB, essa amostra era descartada e uma nova era gerada. A Tabela 3 apresenta o número total de amostras presente em cada conjunto de dados, ou seja, o número de amostras positivas mais o número de amostras negativas.

Tabela 3 – Número total de amostras por conjunto de dados.

Fator Sigma do Conjunto de Dados	Número Total de Amostras
$\sigma 24$	1042
$\sigma 28$	288
$\sigma 32$	622
$\sigma 38$	478
$\sigma 54$	192
$\sigma 70$	3918

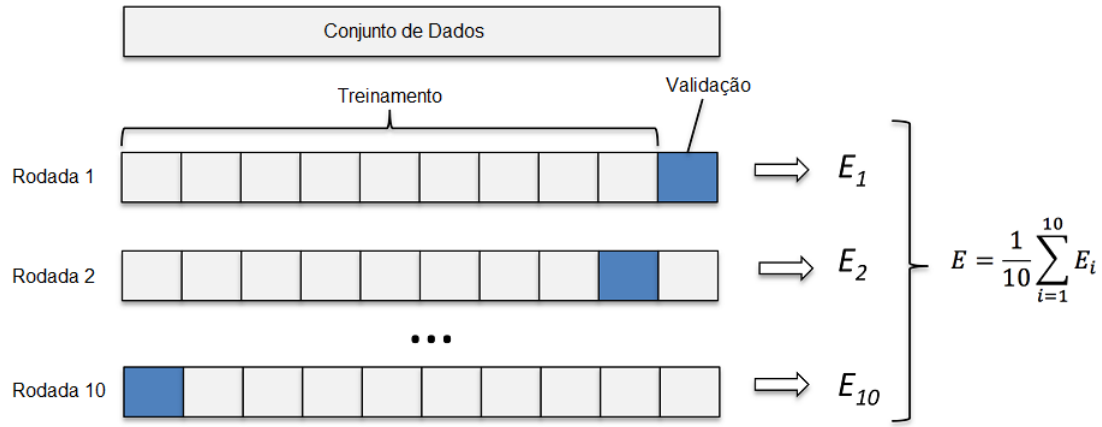
Fonte: Santos-Zavaleta *et al.* (2019)

5.3 TREINAMENTO E VALIDAÇÃO

Para a realização dos treinamentos e validações, foi utilizado o método de validação cruzada (*k-fold cross-validation*). Neste método, o conjunto de dados é dividido em k partições de igual tamanho. O classificador é treinado k vezes, sendo que a cada rodada, $k-1$ partições são utilizadas para treinamento e uma partição é utilizada exclusivamente para validação. A performance estimada é a média da performance de todas as rodadas (DUDA; HART; STORK, 2000) (KOHAVI, 1995).

A Figura 18 apresenta um exemplo de validação cruzada onde $k = 10$. O conjunto de dados é dividido em 10 partições, sendo que inicialmente as 9 primeiras partições são usadas para treinamento, e a última é usada para validação. Na rodada seguinte, a penúltima é usada para validação e as demais são usadas para treinamento. Esse processo é executado sucessivamente, sendo que cada uma das 10 rodadas utiliza uma partição diferente para validação e resulta em uma estimativa de performance E_i. A média das 10 estimativas é a performance estimada para aquele conjunto de dados (DUDA; HART; STORK, 2000).

Figura 18 – Método de Validação Cruzada.



Fonte: Norena (2018)

Cada execução resulta em uma contagem de acertos e erros de classificação. O resultado de cada execução pode ser organizado em uma matriz de confusão, conforme a Figura 19, onde os verdadeiros positivos (VP) correspondem a amostras positivas classificadas corretamente como positivas; verdadeiros negativos (VN) correspondem a amostras negativas classificadas corretamente como negativas; falsos positivos (FP) correspondem a amostras negativas classificadas erroneamente como positivas; por fim, falsos negativos (FN) correspondem a amostras positivas classificadas erroneamente como negativas (THEODORIDIS; KOUTROUMBAS, 2009).

Figura 19 – Matriz de Confusão.

Fonte: Souza (2009)

Os resultados apresentados na matriz de confusão podem ser utilizados para calcular a acurácia, especificidade, sensibilidade e precisão, conforme a Equações 5.1, 5.2, 5.3 e 5.4 respectivamente. A acurácia representa a porcentagem de amostras classificadas corretamente em relação ao total de amostras; a especificidade representa a porcentagem de amostras classificadas corretamente como negativas em relação ao total de amostras negativas; a sensibilidade representa a porcentagem de amostras classificadas corretamente como positivas em relação ao total de amostras positivas; a precisão representa a porcentagem de amostras classificadas corretamente como positivas em relação ao total de amostras classificadas como positivas (WEBB; COPSEY, 2011).

$$A = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

$$E = \frac{VN}{VN + FP} \tag{5.2}$$

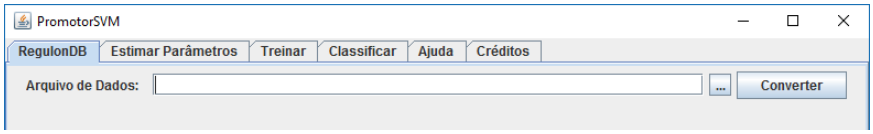
$$S = \frac{VP}{VP + FN} \tag{5.3}$$

$$P = \frac{VP}{VP + FP} \tag{5.4}$$

5.4 INTERFACE GRÁFICA

Para facilitar o uso da implementação, foi desenvolvida uma interface gráfica que permite ao usuário acessar suas funcionalidades. Como pode ser observado na Figura 20, a interface é composta por seis abas, sendo que cada aba contempla uma função específica. A primeira aba, intitulada ”RegulonDB”, permite ao usuário carregar um arquivo de texto proveniente da base de dados RegulonDB, e convertê-lo para um arquivo no formato CSV (*Comma-Separated-Values*).

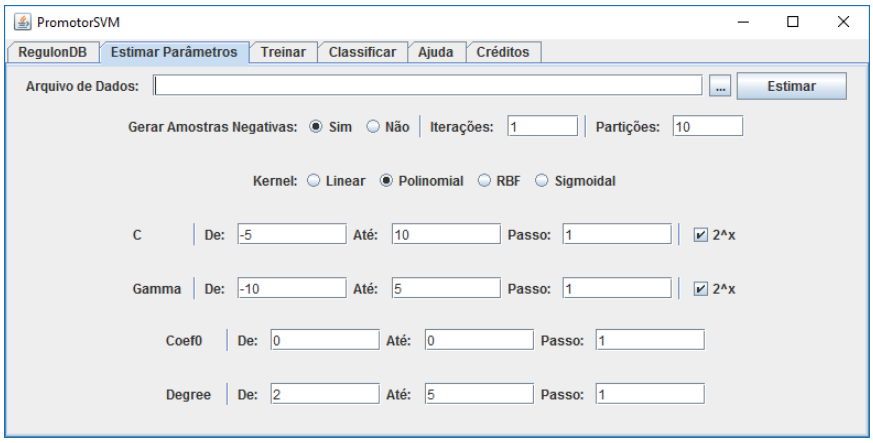
Figura 20 – Aba ”RegulonDB” da Interface Gráfica.



Fonte: O Autor

A segunda aba da interface é intitulada ”Estimar Parâmetros”, e pode ser observada na Figura 21. Esta opção apresenta como objetivo automatizar o processo de estimativa dos parâmetros, sendo que esse processo é realizado a partir de sucessivas execuções de treinamentos e validações.

Figura 21 – Aba ”Estimar Parâmetros” da Interface Gráfica.



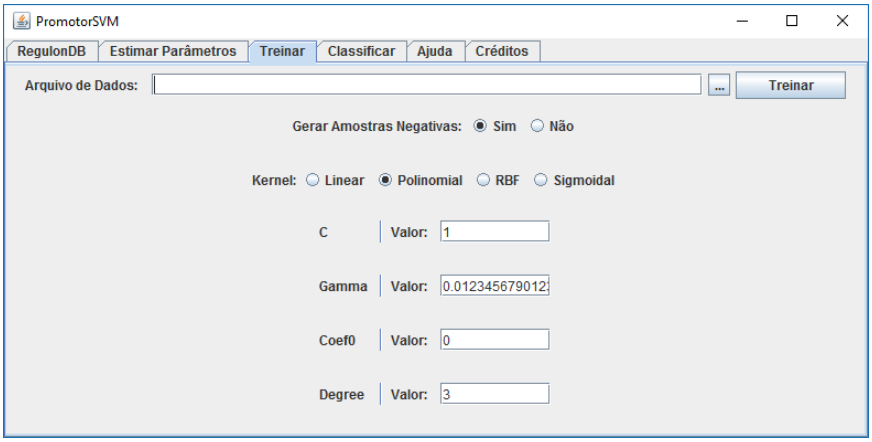
Fonte: O Autor

O campo ”Arquivo de Dados” é utilizado para carregar um arquivo no formato CSV, gerado previamente na aba ”RegulonDB”, através de outro software, ou de forma manual pelo usuário. É possível informar se o arquivo já possui amostras negativas ou se estas serão geradas em tempo de execução.

Além disso, a partir da interface é possível controlar o número de repetições da validação cruzada, o número de partições, o *kernel* a ser utilizado nos testes, e os parâmetros de controle desse *kernel*. Para cada parâmetro do *kernel*, é possível informar um valor inicial, um valor final, o incremento a ser utilizado, e se esse incremento é linear ou exponencial.

A terceira aba da interface é intitulada ”Treinar”, e pode ser observada na Figura 22. Esta aba permite ao usuário treinar a SVM com parâmetros específicos, e salvar o modelo resultante em um arquivo de texto. Os demais parâmetros possuem funcionamento idêntico aos da aba ”Estimar Parâmetros”.

Figura 22 – Aba ”Treinar” da Interface Gráfica.



Fonte: O Autor

A quarta aba da interface, intitulada ”Classificar”, pode ser observada na Figura 23. Esta aba permite ao usuário classificar amostras com base em um modelo previamente gerado na aba ”Treinar”. Após carregar o arquivo modelo, é possível carregar um arquivo no formato CSV, previamente gerado na aba ”RegulonDB” ou gerado de outra forma pelo usuário. As amostras presentes no arquivo CSV são classificadas conforme o modelo, e os resultados são salvos em um arquivo no formato CSV. A aba ”Ajuda” contém breves explicações sobre o funcionamento da implementação. Por fim, a aba ”Créditos” possui informações sobre a autoria da implementação.

Figura 23 – Aba ”Classificar” da Interface Gráfica.

Fonte: O Autor

5.5 TESTES E RESULTADOS

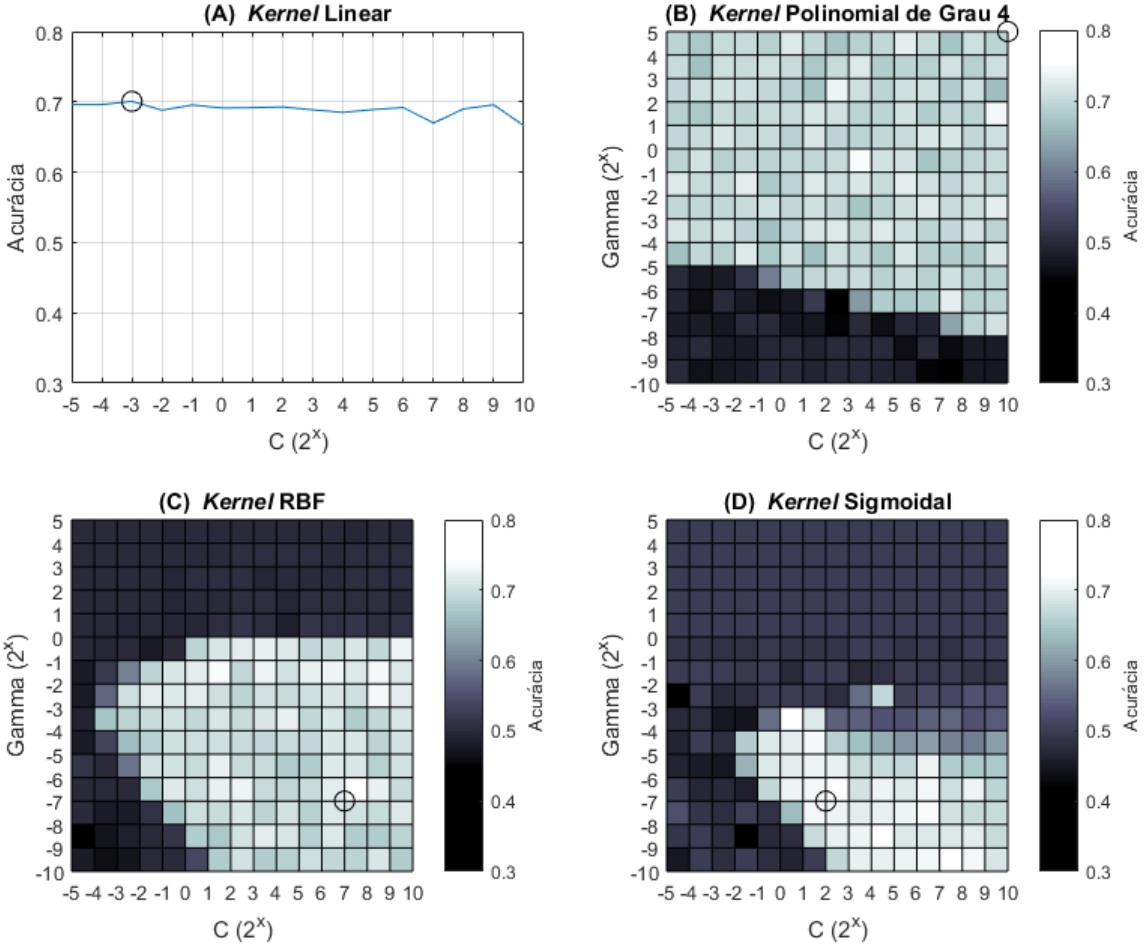
A validação cruzada de 10 partições foi executada para cada conjunto de dados, utilizando-se os *kernels* Linear, Polinomial, RBF e Sigmoidal, além de variações em seus parâmetros (KOHAVI, 1995) (DUDA; HART; STORK, 2000). O parâmetro C recebeu valores entre 2^{-5} e 2^{10}, com incremento de 1 unidade em seu expoente a cada teste. O parâmetro $Gamma$ recebeu valores entre 2^{-10} e 2^{5}, com incremento de 1 unidade em seu expoente a cada teste. O parâmetro $Degree$ recebeu valores entre 2 e 5, com incremento de 1 unidade a cada teste (HSU; CHANG; LIN, 2003).

Os *kernels* disponíveis na biblioteca LibSVM suportam como entrada apenas valores numéricos, portanto cada nucleotídeo foi substituído por um valor entre -1 e 1: $A = 0.25$, $C = 0.5$, $G = 0.75$ e $T = 1.0$. Esses valores foram escolhidos empiricamente através de testes preliminares. Como os valores de entrada estão dentro do intervalo de -1 até 1, o parâmetro $Coef0$ não precisou ser utilizado, recebendo valor zero em todos os testes.

5.5.1 Resultados Obtidos para o σ24

A Figura 24 apresenta os resultados obtidos para o σ24, organizados por *kernel*, sendo que um círculo identifica o melhor resultado obtido para aquele *kernel*. Entre todos os testes, o melhor resultado foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^{10}$, $Gamma = 2^5$ e $Degree = 4$, alcançando-se uma acurácia de 75.6%, uma especificidade de 78.8%, uma sensibilidade de 72.3% e uma precisão de 77.4%. Comparativamente, o trabalho desenvolvido por Silva (2011), no qual foram utilizadas Redes Neurais Artificiais, alcançou uma acurácia de 86.9%, uma especificidade de 95.6% e uma sensibilidade de 78.2%.

Figura 24 – Resultados obtidos para o σ24.

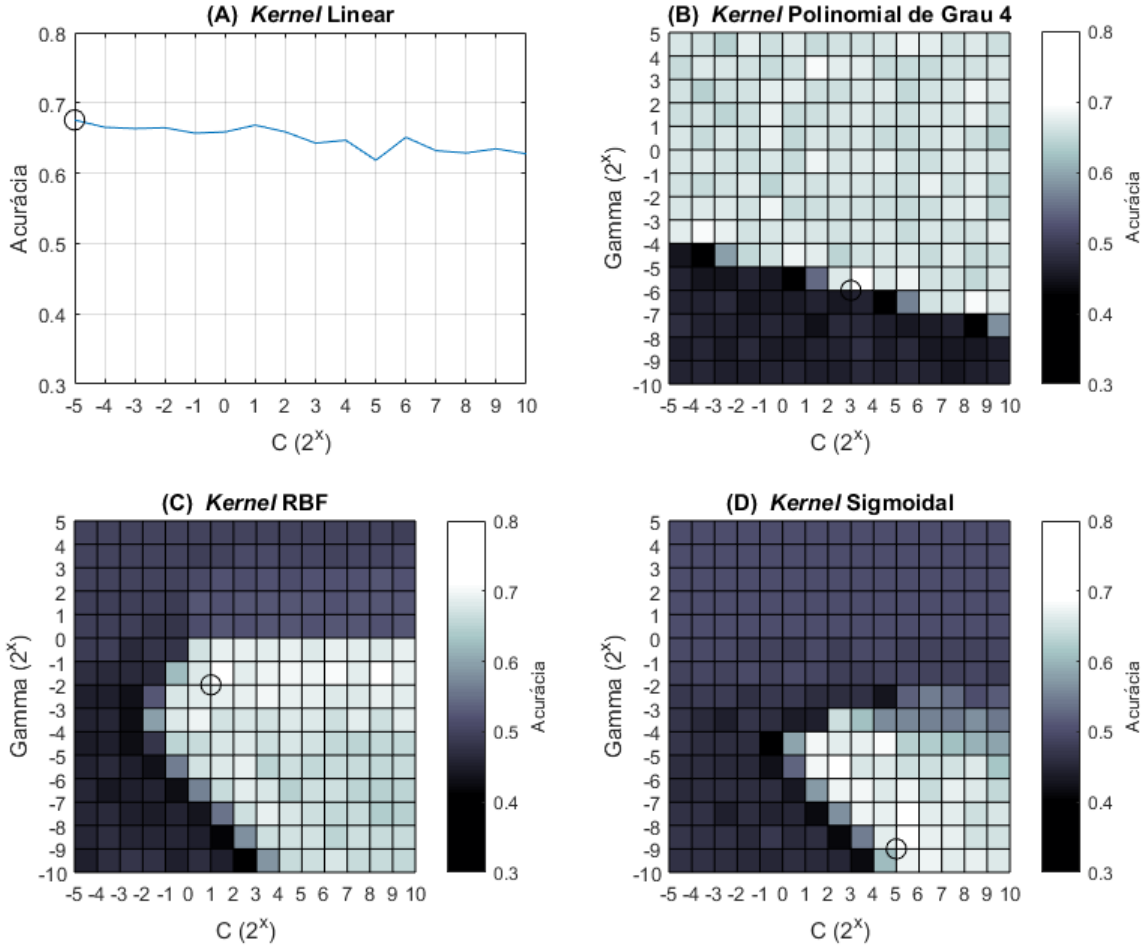


Fonte: O Autor

5.5.2 Resultados Obtidos para o $\sigma 28$

A Figura 25 apresenta os resultados obtidos para o $\sigma 28$, organizados por *kernel*, sendo que um círculo identifica o melhor resultado obtido para aquele *kernel*. Entre todos os testes, o melhor resultado foi obtido utilizando o *kernel* RBF com os parâmetros $C = 2^1$ e $Gamma = 2^{-2}$, alcançando-se uma acurácia de 71.2%, uma especificidade de 74.2%, uma sensibilidade de 68.2% e uma precisão de 72.5%. Comparativamente, o trabalho desenvolvido por Silva (2011), no qual foram utilizadas Redes Neurais Artificiais, alcançou uma acurácia de 92.8%, uma especificidade de 90.4% e uma sensibilidade de 95.2%.

Figura 25 – Resultados obtidos para o $\sigma 28$.

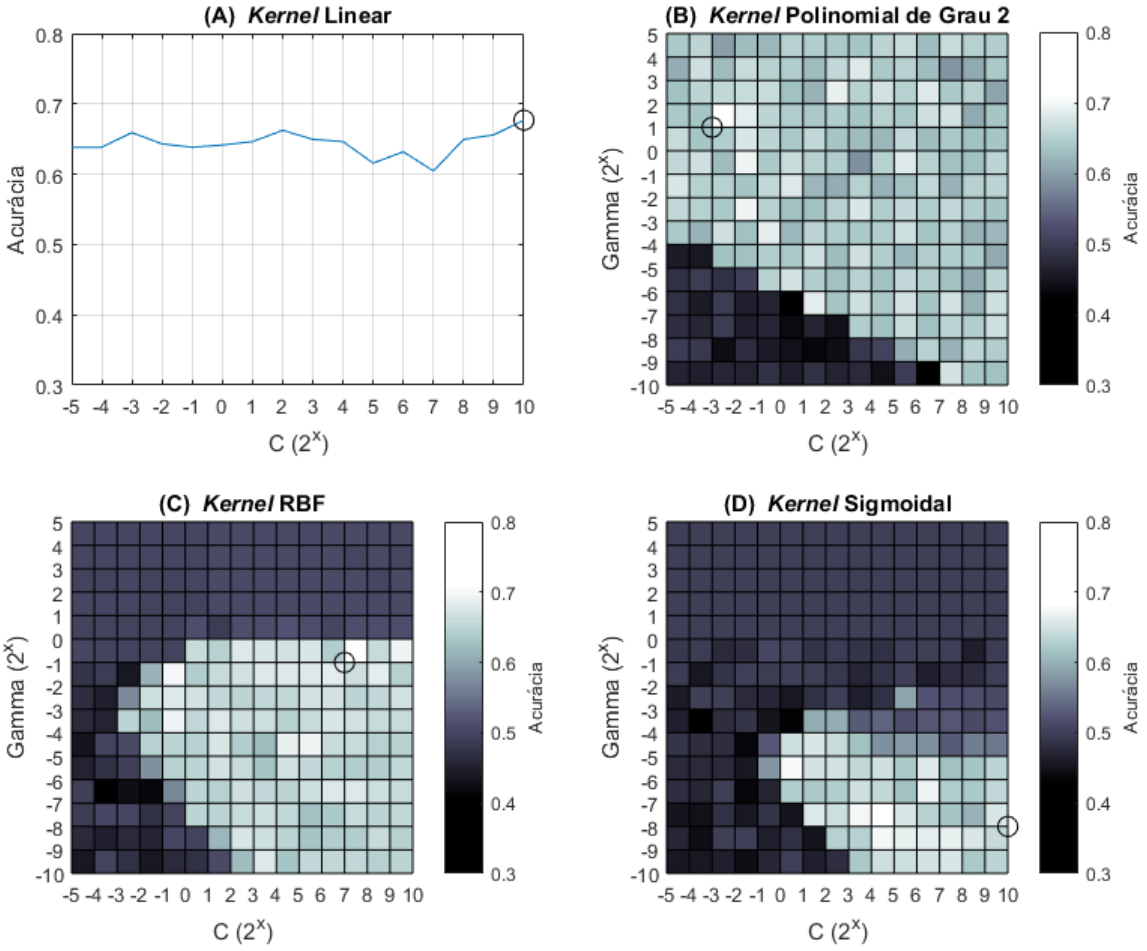


Fonte: O Autor

5.5.3 Resultados Obtidos para o $\sigma 32$

A Figura 26 apresenta os resultados obtidos para o $\sigma 32$, organizados por *kernel*, sendo que um círculo identifica o melhor resultado obtido para aquele *kernel*. Entre todos os testes, o melhor resultado foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^{-3}$, $Gamma = 2^{1}$ e $Degree = 2$, alcançando-se uma acurácia de 71.2%, uma especificidade de 74.9%, uma sensibilidade de 67.5% e uma precisão de 72.9%. Comparativamente, o trabalho desenvolvido por Silva (2011), no qual foram utilizadas Redes Neurais Artificiais, alcançou uma acurácia de 91.5%, uma especificidade de 92.9% e uma sensibilidade de 90.1%.

Figura 26 – Resultados obtidos para o $\sigma 32$.

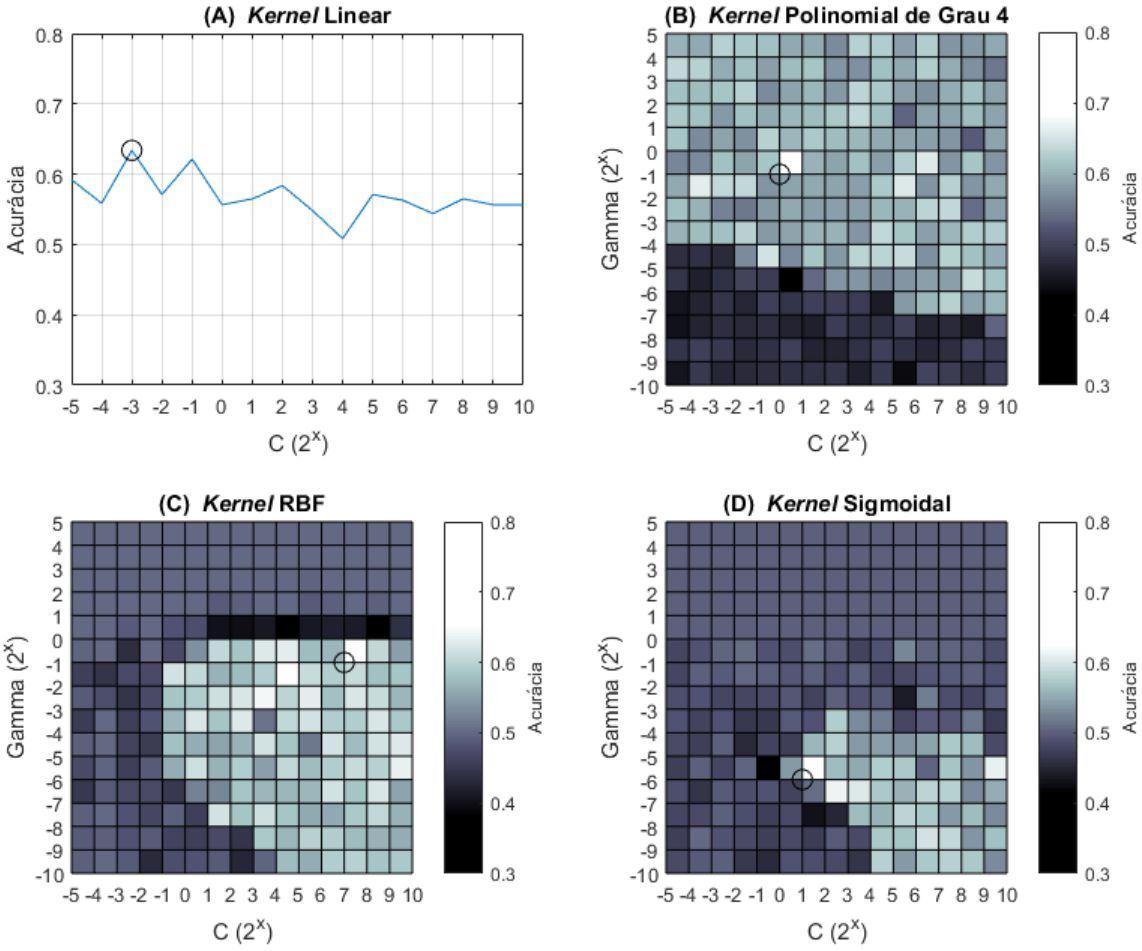


Fonte: O Autor

5.5.4 Resultados Obtidos para o $\sigma 38$

A Figura 27 apresenta os resultados obtidos para o $\sigma 38$, organizados por *kernel*, sendo que um círculo identifica o melhor resultado obtido para aquele *kernel*. Entre todos os testes, o melhor resultado foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^0$, $Gamma = 2^{-1}$ e $Degree = 4$, alcançando-se uma acurácia de 68.4%, uma especificidade de 75.3%, uma sensibilidade de 61.5% e uma precisão de 71.3%. Comparativamente, o trabalho desenvolvido por Silva (2011), no qual foram utilizadas Redes Neurais Artificiais, alcançou uma acurácia de 89.3%, uma especificidade de 83.0% e uma sensibilidade de 93.9%.

Figura 27 – Resultados obtidos para o $\sigma 38$.

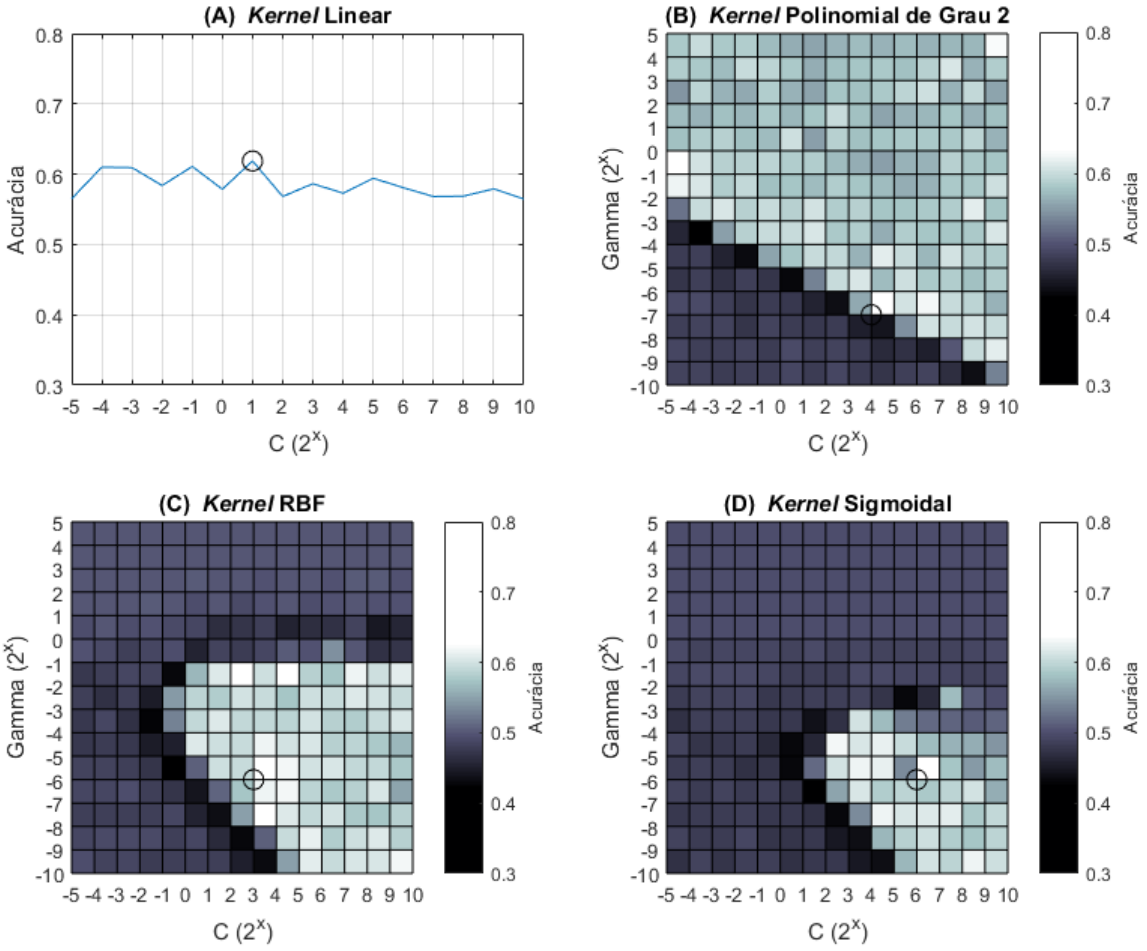


Fonte: O Autor

5.5.5 Resultados Obtidos para o $\sigma 54$

A Figura 28 apresenta os resultados obtidos para o $\sigma 54$, organizados por *kernel*, sendo que um círculo identifica o melhor resultado obtido para aquele *kernel*. Entre todos os testes, o melhor resultado foi obtido utilizando o *kernel* sigmoidal com os parâmetros $C = 2^6$ e $Gamma = 2^{-6}$, alcançando-se uma acurácia de 63.9%, uma especificidade de 55.8%, uma sensibilidade de 72.0% e uma precisão de 62.0%. Comparativamente, o trabalho desenvolvido por Silva (2011), no qual foram utilizadas Redes Neurais Artificiais, alcançou uma acurácia de 97.0%, uma especificidade de 100.0% e uma sensibilidade de 94.1%.

Figura 28 – Resultados obtidos para o $\sigma 54$.

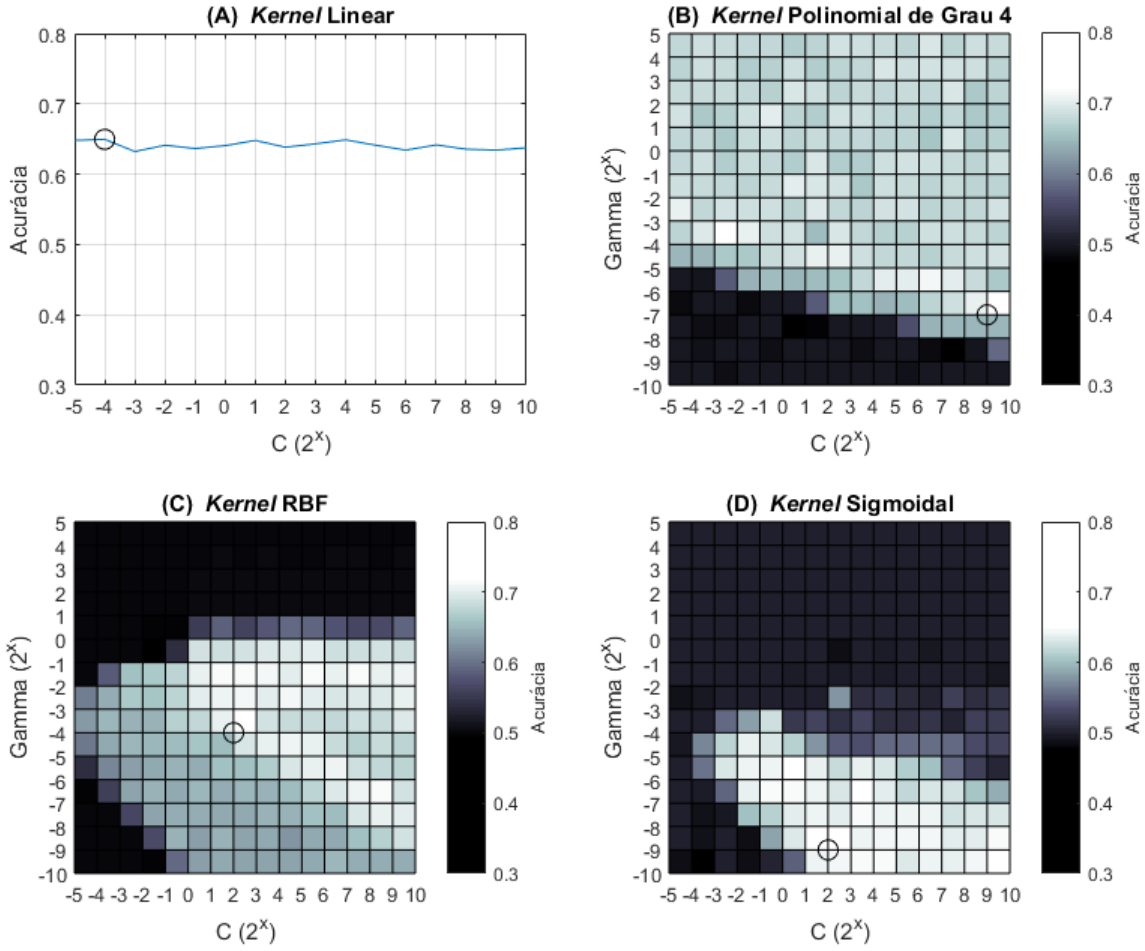


Fonte: O Autor

5.5.6 Resultados Obtidos para o σ70

A Figura 29 apresenta os resultados obtidos para o σ70, organizados por *kernel*, sendo que um círculo identifica o melhor resultado obtido para aquele *kernel*. Entre todos os testes, o melhor resultado foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^9$, $Gamma = 2^{-7}$ e $Degree = 4$, alcançando-se uma acurácia de 72.2%, uma especificidade de 70.5%, uma sensibilidade de 73.9% e uma precisão de 71.5%. Comparativamente, o trabalho desenvolvido por Silva (2011), no qual foram utilizadas Redes Neurais Artificiais, alcançou uma acurácia de 83.6%, uma especificidade de 85.4% e uma sensibilidade de 81.8%.

Figura 29 – Resultados obtidos para o σ70.



Fonte: O Autor

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvida uma implementação para a identificação de regiões promotoras em sequências de DNA, utilizando o método SVM. Após uma análise de diferentes bibliotecas SVM (DATA SCIENCE CENTRAL, 2015), optou-se pelo uso da biblioteca LibSVM por ser multi-plataforma, ser disponibilizada em código aberto, e implementar os *kernels* Linear, Polinomial, RBF e Sigmoidal (CHANG; LIN, 2011). A versão 3.24 da biblioteca LibSVM se encontra disponível nas linguagens de programação Java e Python, sendo que a linguagem Java foi escolhida por possuir um melhor desempenho quando comparada à linguagem de programação Python (FOURMENT; GILLINGS, 2008).

Para a realização dos treinamentos e testes, foram utilizadas como amostras positivas arquivos retirados da base RegulonDB (versão 10.6.3), divididos conforme o fator sigma que se liga àquela região promotora. Tais arquivos possuem amostras de DNA contendo regiões promotoras de bactérias *Escherichia coli*, cada uma com 81 nucleotídeos (SANTOS-ZAVALETA *et al.*, 2019).

Como amostras negativas, foi utilizada a abordagem proposta por Gan, Guan & Zhou (2012), na qual a sequência de DNA de cada amostra positiva é embaralhada de forma aleatória. Desta forma, são mantidas as proporções de nucleotídeos presentes em amostras reais (KANHERE; BANSAL, 2005). Para a realização dos testes foi gerado um número de amostras negativas igual à quantidade presente em cada arquivo da base RegulonDB, de forma que cada conjunto de dados contivesse 50% de amostras positivas e 50% de amostras negativas. Os treinamentos e validações utilizaram o método de validação cruzada, sendo que foram utilizadas 10 partições para cada conjunto de dados (KOHAVI, 1995) (DUDA; HART; STORK, 2000).

O melhor resultado para o σ24 foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^{10}$, $Gamma = 2^5$ e $Degree = 4$, alcançando-se uma acurácia de 75.6%. O melhor resultado para o σ28 foi obtido utilizando o *kernel* RBF com os parâmetros $C = 2^1$ e $Gamma = 2^{-2}$, obtendo-se uma acurácia de 71.2%. O melhor resultado para o σ32 foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^{-3}$, $Gamma = 2^1$ e $Degree = 2$, obtendo-se uma acurácia de 71.2%. O melhor resultado para o σ38 foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^0$, $Gamma = 2^{-1}$ e $Degree = 4$, obtendo-se uma acurácia de 68.4%. O melhor resultado para o σ54 foi obtido utilizando o *kernel* sigmoidal com os parâmetros $C = 2^6$ e $Gamma = 2^{-6}$, obtendo-se uma acurácia de 63.9%. Por fim, o melhor resultado para o σ70 foi obtido utilizando o *kernel* polinomial com os parâmetros $C = 2^9$, $Gamma = 2^{-7}$ e $Degree = 4$, obtendo-se uma acurácia de 72.2%.

Os resultados obtidos neste trabalho foram inferiores aos obtidos no trabalho desenvolvido por Silva (2011), que utilizou Redes Neurais Artificiais para classificação de regiões promotoras. Para os testes realizados em Silva (2011), como amostras negativas, foram utilizadas sequências de DNA geradas de forma aleatória, contendo uma proporção de nucleotídeos similar a amostras reais, além de áreas intergênicas reais, que não possuem região promotora.

A região promotora possui proporções de nucleotídeos diferentes de outras regiões do DNA (KANHERE; BANSAL, 2005); desta forma tem-se a possibilidade de que a diferença de proporções de nucleotídeos de uma região promotora e uma região intergênica seja maior que a diferença entre uma região promotora real e uma região promotora embaralhada. Sugere-se a realização de testes cruzados para verificar essa possibilidade, utilizando ambas as implementações e ambos os métodos de obtenção de amostras negativas.

6.1 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, propõe-se:

- Utilizar áreas intergênicas como amostras negativas, ou seja, sequências de DNA reais que comprovadamente não possuem região promotora, de forma similar ao trabalho realizado por Silva (2011);
- Utilizar o *kernel* String para classificação das amostras, atualmente indisponível na linguagem Java. A versão 3.24 da biblioteca LibSVM possui esse *kernel* em caráter experimental, implementado apenas na linguagem C (CHANG; LIN, 2011);
- Utilizar o método KNN para classificação das amostras;
- Desenvolver uma implementação capaz de classificar amostras quanto à presença de uma região de terminação. A versão 10.6.3 da base RegulonDB possui amostras desse tipo (SANTOS-ZAVALETA *et al.*, 2019).

REFERÊNCIAS

ALMEIDA, L. M. de; PIRES, C. E. de B. M. **Biologia celular**: Estrutura e organização molecular. São Paulo: Érica, 2014. ISBN 978-85-365-2080-3.

BONORA, T. *et al.* **O DNA e as Mutações**. 2016. <http://romeo.if.usp.br/~browngon/>. Acesso em: 11 agosto 2019.

BORDONI, E. *et al.* **A dupla hélice de DNA em detalhes**. 2011. <https://canalcederj.cecierj.edu.br/recurso/7211>. Acesso em: 11 agosto 2019.

BURCHARD, E. G. *et al.* Association between a sequence variant in the il-4 gene promoter and fev(1) in asthma. **ATS Journals**, n. 160 (3), 1999.

CARVALHO, H. F.; RECCO-PIMENTEL, S. M. **A célula**. 3. ed. Barueri, SP: Manole, 2013. ISBN 978-85-204-3578-6.

CHANG, C.-C.; LIN, C.-J. Libsvm: A library for support vector machines. **ACM Transactions on Intelligent Systems and Technology**, n. 2, p. 27:1–27:27, 2011.

CHEN, S.-T. *et al.* Comparative analysis of logistic regression, support vector machine and artificial neural network for the differential diagnosis of benign and malignant solid breast tumors by the use of three-dimensional power doppler imaging. **Korean Journal of Radiology**, n. 10 (5), p. 464–71, 2009.

DATA SCIENCE CENTRAL. **Collection Of SVM Libraries By Language**. 2015. <https://www.datasciencecentral.com/profiles/blogs/collection-of-svm-libraries-by-language>. Acesso em: 3 novembro 2019.

DONG, J. X.; KRZYZAK, A.; SUEN, C. Y. **A Fast SVM Training Algorithm**. 2002. <https://www.concordia.ca/research/cenparmi/resources/herosvm.html>. Acesso em: 3 novembro 2019.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern classification**. 2. ed. [S.l.]: Wiley-Interscience, 2000. ISBN 978-0471056690.

FACELI, K. *et al.* **Inteligência artificial**: uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011. ISBN ISBN: 978-85-216-1880-5.

FOURMENT, M.; GILLINGS, M. R. A comparison of common programming languages used in bioinformatics. **BMC Bioinformatics**, n. 9 (82), 2008.

GAN, Y.; GUAN, J.; ZHOU, S. A comparison study on feature selection of dna structural properties for promoter prediction. **BMC Bioinformatics**, v. 13, p. 4, 2012.

HOBBS, K. *et al.* Interleukin-10 and transforming growth factor-beta promoter polymorphisms in allergies and asthma. **ATS Journals**, n. 158 (6), 1998.

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. A practical guide to support vector classification. 2003.

IONESCU-TIRGOVISTE, C. *et al.* Structural properties of gene promoters highlight more than two phenotypes of diabetes. **PLoS ONE**, n. 10 (9): e0137950, 2015.

JACOBI, T. **Bioinformatic methods for eukaryotic RNA-Seq-based promoter identification**. Tese (Doutorado) — Faculdade de Engenharia da Universidade de Bielefeld, Alemanha, 2014.

JOACHIMS, T. **Making Large-Scale SVM Learning Practical**. 1999. <http://svmlight.joachims.org/>. Acesso em: 10 novembro 2019.

KANHERE, A.; BANSAL, M. Structural properties of promoters: similarities and differences between prokaryotes and eukaryotes. **Nucleic Acids Research**, n. 33 (10), p. 3165–3175, 2005.

KARUSH, W. **Minima of Functions of Several Variables with Inequalities as Side Conditions**. Chicago, IL, Estados Unidos: [s.n.], 1939.

KIM, E. **Everything You Wanted to Know about the Kernel Trick**: (but were too afraid to ask). 2017. <http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html>. Acesso em: 3 novembro 2019.

KOHAVI, R. A study of crossvalidation and bootstrap for accuracy estimation and model selection. **International Joint Conference on Articial Intelligence**, 1995.

KOWALCZYK., A. 2014. <https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/>. Acesso em: 13 outubro 2019.

KOWALCZYK, A. **Support Vector Machines Succinctly**. [S.l.]: Syncfusion, 2017.

KUHN, H. W.; TUCKER, A. W. Nonlinear programming. **Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability**, p. 481–492, 1951.

KULOZIK, A. E. *et al.* Thalassemia intermedia: Moderate reduction of beta globin gene transcriptional activity by a novel mutation of the proximal caccc promoter element. **Blood Journal**, n. 77, p. 2054–2058, 1991.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Uma introdução às support vector machines. **Revista de Informática Teórica Aplicada**, n. 14 (2), p. 43–67, 2007.

LUGER, G. F. **Inteligência artificial**. 6. ed. São Paulo: Pearson Education do Brasil, 2013. ISBN 978-85-8143-550-3.

MCQUISTEN, K. A.; PEEK, A. S. Comparing artificial neural networks, general linear models and support vector machines in building predictive models for small interfering rnas. **PLoS ONE**, n. 4(10):e7522, 2009.

NORENA, S. **Python Model Tuning Methods Using Cross Validation and Grid Search**. 2018. <https://medium.com/@sebastiannorena/some-model-tuning-methods-bfef3e6544f0>. Acesso em: 10 novembro 2019.

PETRIF, F. *et al.* Rubinstein-taybi syndrome caused by mutations in the transcriptional co-activator cbp. **Nature**, n. 376, p. 348–351, 1995.

PIMENTA, C. A. M.; LIMA, J. M. de. **Genética aplicada à biotecnologia**. São Paulo: Érica, 2015.

PLATT, J. C. **Sequential Minimal Optimization**: A fast algorithm for training support vector machines. [S.l.], 1998.

RUPING, S. **mySVM - Manual**. 2000. <https://www-ai.cs.tu-dortmund.de/SOFTWARE/MYSVM/index.html>. Acesso em: 10 novembro 2019.

SANDERS, M.; BOWMAN, J. **Análise genética**: uma abordagem integrada. São Paulo: Pearson Education do Brasil, 2014.

SANTOS, F. J. dos; FERREIRA, S. F. **Geometria Analítica**. Porto Algre: Bookman, 2009. ISBN ISBN: 978-85-7780-503-7.

SANTOS-ZAVALETA, A. *et al.* Regulondb v10.5: Tackling challenges to unify classic and high throughput knowledge of gene regulation in e. coli k-12. **Nucleic Acids Research**, n. 47 (D1), p. D212–D220, 2019.

SHARMA, A. *et al.* A comparative study of support vector machine, artificial neural network and bayesian classifier for mutagenicity prediction. **Interdisciplinary Sciences: Computational Life Sciences**, n. 3, 232, 2011.

SILVA, S. de Avila e. **Redes neurais artificiais aplicadas no reconhecimento de regiões promotoras em bactérias Gram-negativas**. Tese (Doutorado) — Universidade de Caxias do Sul, Caxias do Sul, 2011.

SOUZA, C. **Análise de Poder Discriminativo Através de Curvas ROC**. 2009. <http://crsouza.com/2009/07/13/analise-de-poder-discriminativo-atraves-de-curvas-roc/>. Acesso em: 10 novembro 2019.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern recognition**. 4. ed. Londres: Elsevier, 2009. ISBN 978-1-59749-272-0.

VOLPI, A. **Ferramentas de Python para Aprendizado de Máquina**. 2015. <https://alexandrevolpi.wordpress.com/2015/10/02/ferramentas-de-python-para-aprendizado-de-maquina/>. Acesso em: 11 agosto 2019.

WEBB, A. R.; COPSEY, K. D. **Statistical pattern recognition**. 3. ed. Reino Unido: Wiley, 2011.