BIANCA ALVES DE ALMEIDA

PREDIÇÃO DE SÍTIOS DE INÍCIO DE TRADUÇÃO EM SEQÜÊNCIAS DE RNAm UTILIZANDO REDES NEURAIS ARTIFICIAIS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS

MINAS GERAIS - BRASIL

2008

BIANCA ALVES DE ALMEIDA

PREDIÇÃO DE SÍTIOS DE INÍCIO DE TRADUÇÃO EM SEQÜÊNCIAS DE RNAm UTILIZANDO REDES NEURAIS ARTIFICIAIS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:

Bioinformática

Orientador:

Prof. Dr. Thiago de Souza Rodrigues

LAVRAS

MINAS GERAIS – BRASIL

2008

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Almeida, Bianca Alves de

Predição de Sítios de Início de Tradução em seqüências de RNAm utilizando Redes Neurais Artificiais/Bianca Alves de Almeida – Minas Gerais, 2008.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Bioinformática. 2. Sítios de Início de Tradução. 3. Redes Neurais Artificiais. I. ALMEIDA, B. A. II. Universidade Federal de Lavras. III. Título.

BIANCA ALVES DE ALMEIDA

PREDIÇÃO DE SÍTIOS DE INÍCIO DE TRADUÇÃO EM SEQÜÊNCIAS DE RNAm UTILIZANDO REDES NEURAIS ARTIFICIAIS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 18 de junho de 2008

_____
Prof. Dr. Wilian Soares Lacerda

_____
Prof. Dr. Joaquim Quinteiro Uchôa

_____
Prof. Dr. Thiago de Souza Rodrigues
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2008

Àqueles que sempre acreditaram em mim: meus pais.

Agradecimentos

Agradeço, primeiramente, a Deus pela proteção e sustento.

À minha mãe pelo amor incondicional e pela amizade.

Ao meu pai pelo apoio e cuidado.

E a ambos por terem se sacrificado tanto para que eu aqui chegasse.

Ao meu irmão Diego por ter me ensinado tantas coisas importantes.

Ao meu avô tão querido pelo imenso carinho.

A todos de minha família por participarem da minha vida de maneira tão especial.

Aos meus amigos inesquecíveis, Rafa, Emi, Camilinha, Dadá, Sheila, Vê, Luci...
Sentirei saudades eternas!

Obrigada por alegrarem meus dias.

À SWFactory pela oportunidade de aprender tantas coisas importantes.

Ao meu orientador Thiago pela paciência e atenção.

A todos que aqui não nomeei, mas que cruzaram meu caminho durante minha graduação. Há um propósito maior em tudo, mesmo que, a priori, não o reconheçamos.

Há tempo para tudo nesta vida, e este é o de agradecer.

Obrigada a todos!

PREDIÇÃO DE SÍTIOS DE INÍCIO DE TRADUÇÃO EM SEQÜÊNCIAS DE RNAm UTILIZANDO REDES NEURAIS ARTIFICIAIS

RESUMO

O acelerado desenvolvimento das pesquisas na área da biologia molecular, tem disponibilizado um grande volume de informações nas bases de dados biológicas, tornando inviável a análise deste sem algum suporte tecnológico. Um importante tópico da biologia molecular é saber como identificar o Sítio de Início de Tradução (SIT) dada uma seqüência de nucleotídeos. No presente trabalho foi implementada uma rede neural artificial visando predizer o início da tradução numa dada seqüência de RNAm.

Palavras-Chave: Bioinformática, Sítio de Início de Tradução (SIT), Redes Neurais Artificial.

PREDICTION OF TRANSLATION INICIATION SITE IN mRNA SEQUENCES USING ARTIFICIAL NEURAL NETWORK

ABSTRACT

The quick development of research in the molecular biology, make available a large amount that information in the biologics databases, rendering impracticable the analisys without technological support. An important topic of the molecular biology is how to identify translation iniciation site let a nucleotide sequence. This present work has been implemented a artificial neural network for to prediction sites starting translation in a mRNA sequence.

Key-Words: Bioinformatic , Translation Iniciation Site, Neural Networks.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

DNA - Ácido desoxirribonucléico

MCP - *McCulloch e Pitts Perceptron*

MLP- *Multilayer Perceptron*

MSE - *Mean Squared Error*

NCBI - *National Center for Biotechnology Information*

ORF - *Open Reading Frame*

RNAm - Ácido Ribonucléico Mensageiro

SIT - Sítio de Início de Tradução

SVM - *Support Vector Machines*

URT - *Untranslated Regions*

1. INTRODUÇÃO

1.1. Motivação

Logo que a descoberta do DNA (ácido desoxirribonucléico) foi determinada, no início dos anos 50, ficou claro que a informação hereditária nas células estava codificada na seqüência de nucleotídeos do DNA (ALBERTS et al., 1999). Desde então houve uma grande mobilização do meio científico que procurava mapear e interpretar o DNA a fim de buscar solução para cura de doenças ou simplesmente entender melhor os processos fisiológicos dos seres vivos.

Grandes projetos como o Projeto Genoma Humano, do *International Human Genome Sequence Consortium*[1], que buscavam fazer o seqüenciamento do DNA geraram grandes quantidades de dados genéticos e bioquímicos. Tais dados, estão relacionados entre si através de uma origem comum: as células dos organismos vivos. Para entender esta relação, a Bioinformática usa o poder computacional para estruturar, catalogar e organizar estas informações de forma compreensiva.

Os métodos computacionais, que já têm mostrado sua utilidade em áreas como a busca de genes e a predição da função e estrutura das proteínas, são decisivos e imprescindíveis para mostrar novos rumos para a biologia molecular (FUCHS, 2002).

Um dos problemas em estudo na comunidade científica é a predição de Sítios de Início de Tradução (SITs) em seqüências codificadoras. É sabido que num RNAm (RNA mensageiro) a tradução para proteínas se inicia num trio de nucleotídeos ATG. Determinar em qual ATG a tradução começa não é uma tarefa trivial. Para tanto, métodos computacionais são aplicados, a fim de obter uma solução aproximada para este problema.

Este trabalho visa utilizar uma rede neural artifical para predizer sítios de início de tradução num conjunto de moléculas de RNAm.

[1] http://www.genome.gov

1.2. Objetivos

O objetivo deste trabalho é desenvolver uma rede neural capaz de predizer SITs em seqüências de RNAm. Os objetivos específicos deste trabalho são:

- Adquirir conhecimento relativo ao problema com enfoque biológico;
- Abordar temas relativos à redes neurais e *data mining*;
- Fazer pesquisas em banco de dados biológicos;
- Implementar e treina uma rede capaz de resolver o problema proposto por este trabalho;
- Analisar os resultados obtidos.

1.3. Estrutura do Trabalho

O capítulo 2 apresenta a teoria que dá sustentamento ao trabalho, apresentando os conceitos e teorias necessárias para a compreensão do mesmo.

O capítulo 3 apresenta a metodologia, mostrando quais técnicas foram utilizadas para resolução do problema objeto deste trabalho.

O capítulo 4 apresenta os resultados obtidos.

O capítulo 5 apresenta a conclusão do trabalho.

O capítulo 6 apresenta o referencial bibliográfico utilizado neste trabalho.

2.REFERENCIAL TEÓRICO

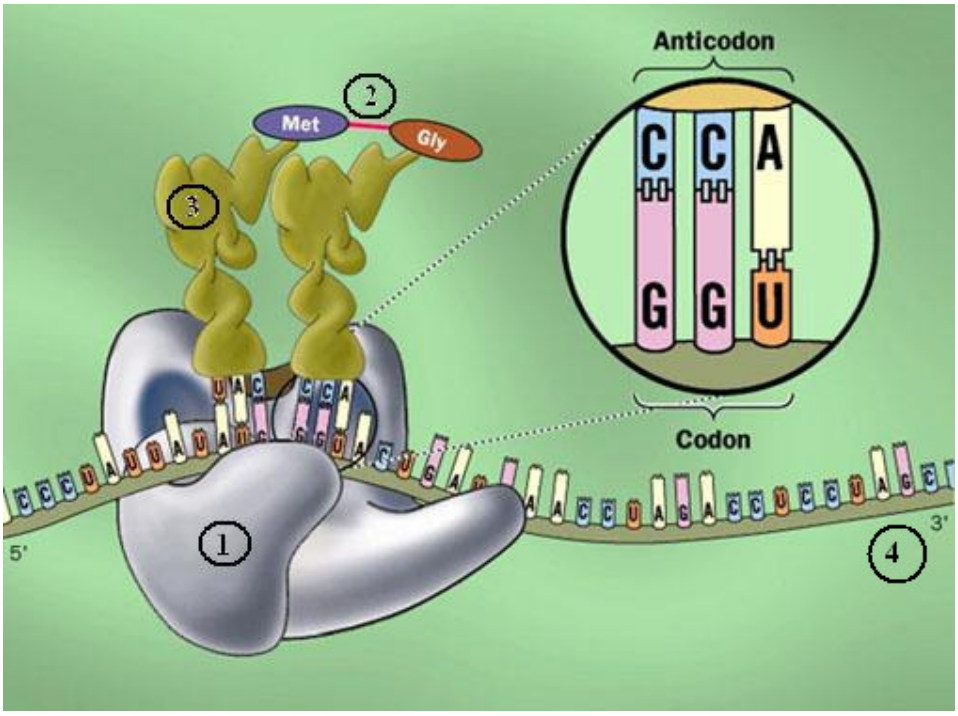
Neste capítulo é relacionada a teoria que dá sustentação ao trabalho, o conjunto de conhecimentos amplamente referenciados que propiciaram o entendimento do problema em estudo.

Qualquer espécie de pesquisa, em qualquer área, supõe e exige uma pesquisa bibliográfica prévia, quer para o levantamento do *estado da arte* do tema, quer para a fundamentação teórica ou ainda para justificar os limites e contribuições da própria pesquisa (CERVO & BERVIAN, 2002).

2.1. A síntese protéica

As proteínas podem executar praticamente todas as funções celulares. Essa multiplicidade de funções desempenhadas por proteínas resulta do número enorme de diferentes formas tridimensionais que estas podem assumir: a função é determinada pela estrutura (ALBERTS et al., 1999). Essas moléculas são as mais complexas que constituem as células. Cada trio de nucleotídeos do RNAm gera um aminoácido, que combinado a outros aminoácidos, formam as proteínas.

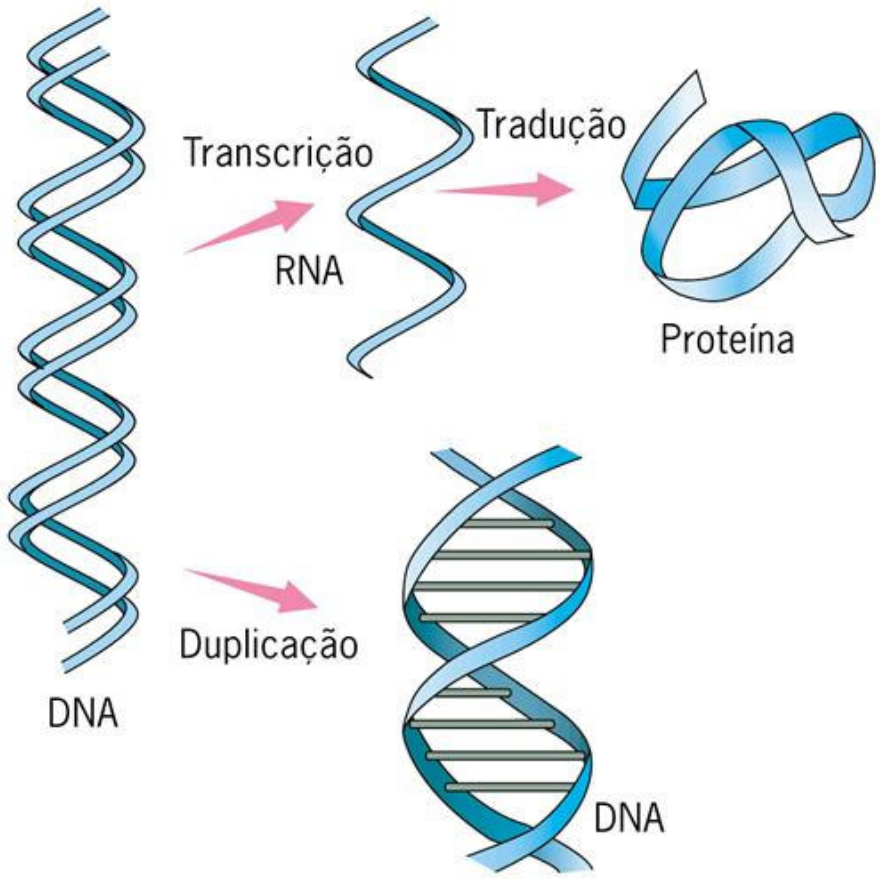
A Figura 2.1 mostra o momento em que a proteína é fabricada. O ribossomo(1) se acopla à fita de RNAm (4), então o RNAt (RNA transportador) (3) transporta o aminoácido (que estava livre no citoplasma) correspondente ao códon da fita de RNAm, une aos aminoácidos já gerados formando as proteínas (2).



2.1 - Um trio de nucleotídeos se une e forma um aminoácido
Fonte: http://www.unb.br/ib/cel/disciplinas/biomol1/traducao/

Uma proteína é um polímero de aminoácidos ligados de modo a formar uma longa cadeia que então enovela-se em uma estrutura tridimensional característica de cada tipo de proteína. Normalmente são encontrados 20 tipos de aminoácidos nas proteínas. As proteínas são os principais constituintes das células e determinam não somente sua estrutura, mas também suas funções.

A informação genética presente no DNA determina a seqüência de aminoácidos das proteínas. O DNA é transcrito em RNA e do RNA é tirada a informação necessária para a geração da proteína. Este é um princípio fundamental e tem sido denominado como *dogma central da biologia molecular.* O dogma central da biologia molecular se divide em duas partes (*vide* Figura2.2*)*: a primeira delas é a síntese protéica, onde a proteína é fabricada de acordo com a informação expressa no DNA, e a segunda é a replicação onde são geradas cópias do DNA.



2.2 - Dogma Central da Biologia Molecular
Fonte: http://biologiacesaresezar.editorasaraiva.com.br/navitacontent_/userFiles/File/

A transcrição e a tradução são os meios pelos quais as células interpretam e expressam suas instruções genéticas – seus *genes*. A primeira etapa da síntese protéica é a *transcrição,* onde a parte requerida do DNA é copiada para uma seqüência de nucleotídeos de RNA. A transcrição produz RNA complementar a uma das fitas do DNA, que atua como molde para a síntese de RNA. Esse RNA produto da transcrição é chamado de *transcrito primário*.

O transcrito primário passa por um conjunto de modificações antes de ser traduzido, a este processamento dá-se o nome de *pós-transcricional*. O processamento pós-transcricional consiste nestes três processos:

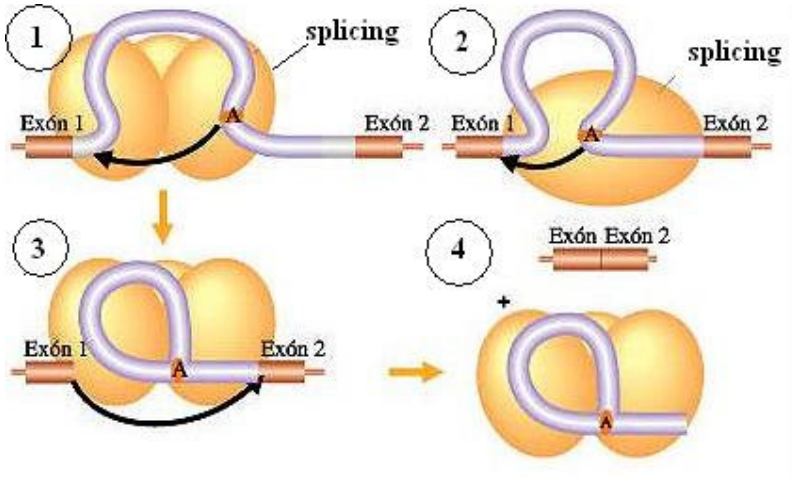
- Inclusão da cauda Poli-A;
- Inclusão do *Cap*;

- *Splicing.*

A cauda Poli-A tem a função de atuar como acentuadora da tradução, proteger o mRNA da degeneração por outras substâncias e aumentar a estabilidade da molécula.

O *Cap* do RNA é uma modificação que ocorre na sua extremidade 5' (Figura 2.1). Ele confere a essa molécula uma maior estabilidade, pois a protege da degradação. Além disso o *Cap 5'* aumenta a chance desse RNA ser capturado pelos sistemas eucarióticos de tradução, levando a uma maior produção de proteínas.

Neste RNA primário há duas regiões possíveis e alternantes: os íntrons e os éxons. Os éxons são fragmentos codificadores e, portanto, relevantes. Já os íntrons são eliminados enzimaticamente, por não serem regiões codificadoras. Este processamento é denomidado *splicing de RNA* e seu produto final é o RNA mensageiro(RNAm). De maneira simplificada pode-se dizer que o *splicing* faz uma "limpeza" no RNA primário, gerando uma fita contínua codificadora – o RNAm. Este processo é ilustrado pela Figura 2.3.



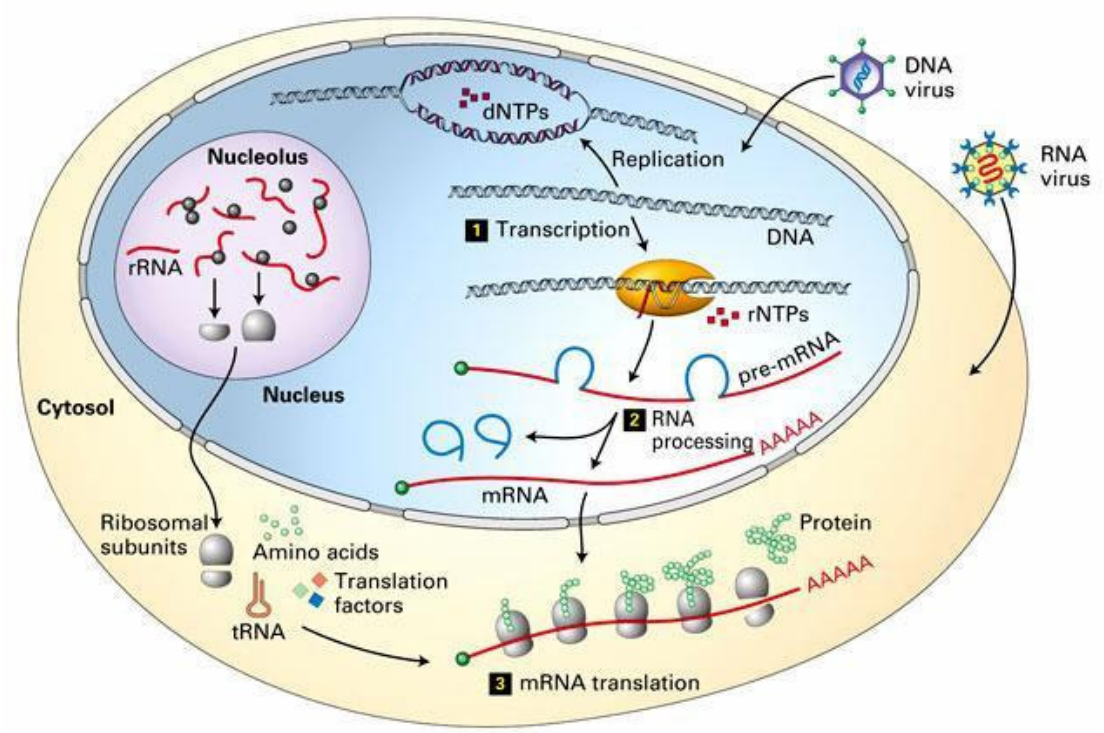
2.3 - Os íntros são eliminados, restando apenas regiões codificadoras

Segundo Zien et al. (2000) nem toda a seqüência de nucleotídeos carregando a informação genética de um indivíduo codifica proteínas. Sabe-se que no RNAm todos os nucleotídeos são codificadores de aminoácidos. No entanto, apenas um segmento contínuo do RNAm é traduzido em aminoácidos. Ou seja, mesmo o RNAm podendo ser traduzido em sua totalidade, apenas um trecho deste RNAm é traduzido em aminoácidos, A este segmento que será traduzido dá-se o nome de *Open Reading Frame (ORF)* enquanto as

demais partes denomina-se *UnTranslated Regions (URT).* A Figura 2.4 mostra uma ORF numa seqüência de RNAm

2.4 - *Open Reading Frame* numa seqüência de RNAm

A partir do RNAm, ocorre o processo de tradução, onde uma proteína será sintetizada pela ação do ribossomo. A Figura 2.5 ilustra a síntese protéica com todos os seus processos, sendo mostrada a fase de transcrição, o processamento do RNA, e a fase de tradução do RNAm em proteína.



2.5 - Visão geral da síntese protéica

O problema abordado neste trabalho é a determinação da posição da ORF em uma molécula de RNAm.

2.2. Sítios de Início de Tradução

Cada trinca de nucleotídeos, componentes do RNAm, resulta em um aminoácido, componente básico de uma proteína. Essas trincas são chamadas de códons. Em eucariotos, o modelo de escaneamento supõe que os ribosomos se ligam primeiro à região 5' (*vide* Figura 2.4) do RNAm e percorre em direção à região 3' (*vide* Figura 2.4) até encontrar o primeiro ATG da seqüência (KOZAK, 1999). Desta forma, começa-se a tradução dos códons para os aminoácidos.

Numa seqüência de RNAm há trios de nucleotídeos com funções especiais. Sabe-se que o início da tradução sempre se dá num códon ATG (*start codon*) e que o fim sempre ocorre quando um dos *stop codons* (TAA, TAG, TGA) é encontrado no processo de escaneamento. No entanto a tradução, em eucariotos, pode não se iniciar no primeiro ATG da seqüência (PEDERSEN & NIELSEN, 1997). Para Alberts (1999), a depender da posição de início da síntese na fita de RNAm, o trio de nucleotídeos selecionado para a síntese poderá variar, variando também os aminoácidos que serão gerados.

Essa falta de padrão no início da tradução faz da predição de SIT uma tarefa complexa, por isso métodos computacionais de busca de um conjunto de características devem ser utilizados a fim de realizar esta detecção.

2.3. Sistemas de Aprendizado

Um sistema de aprendizado caracteriza-se por ter a capacidade de aprender a partir de eventos anteriores. Segundo Weiss (1991) citado por Batista (1997), um sistema de aprendizado é um programa de computador que toma decisões baseado em experiências acumuladas contidas em casos resolvidos com sucesso. Um sistema de aprendizado tem por objetivo extrair conhecimento de um conjunto de dados conhecidos e aplicar a novos dados.

No presente trabalho serão abordados dois tipos de sistemas de aprendizado: aprendizado supervisionado (Rede Neural Artificial) e não-supervisionado (K-Médias). No aprendizado supervisionado, dado um conjunto de dados cujas classes são conhecidas, procura-se encontrar regras capazes de classificar dados com classes desconhecidas. O aprendizado supervisionado tenta induzir a classe dos dados a partir dos dados de entrada, com base nos dados com classes conhecidas.

No aprendizado não-supervisionado, é fornecido um conjunto de dados o qual não se conhece sua classe, e a partir desses dados busca-se estabelecer a existência de classes ou *clusters* dentro do conjunto. A este procedimento dá-se o nome de *clustering*. Cole (1998) define *clustering* como um procedimento exploratório que busca uma estrutura "natural" dentro de um conjunto de dados. Os dados são agrupados segundo suas similaridades ou diferenças, sem nenhuma suposição sobre a estrutura dos dados. O objetivo do aprendizado não-supervisionado é estabelecer as regularidades do conjunto de treinamento.

2.4. Método K-Médias

Muitas vezes ao lidar com grandes conjuntos de dados se faz necessário subdividí-los e classificá-los, de alguma maneira. Os métodos de clusterização fragmentam esses conjuntos a fim de obter subgrupos (*cluster*) menores que possam ser analisados de maneira mais detalhada. O critério para classificação dos elementos pode ser a similaridade ou dissimilaridade entre os elementos.

Métodos de agrupamento de dados são de grande importância para a mineração de dados (*data mining*), porque permitem detectar padrões implícitos nos dados armazenados.

O algoritmo K-Médias classifica as informações de acordo com os próprios dados, através de análises e comparações entre seus valores numéricos. Desta maneira, o algoritmo faz uma classificação automática sem depender de supervisão humana, ou seja, sem pré-classificação existente. Por conta desta característica, o K-Médias é classificado como um algoritmo de mineração de dados (*data mining*) não-supervisionado.

Na maioria das vezes usam-se dados contínuos como entrada para este algoritmo, mas nada impede que dados discretos também sejam utilizados, desde que sejam mapeados para valores numéricos correspondentes. É o que acontece quando se usa o K-Médias pra agrupar seqüências de nucleotídeos.

O algoritmo escolhe aleatoriamente *K* centróides, por isso é classificado como um algoritmo não-determinístico. O número de conglomerados é definido a priori, isto é, o usuário fornece o número *k* de classes a serem agrupadas. O algoritmo analisa todos os

dados e os classifica, selecionando uma classe (*cluster*) e dizendo quais elementos pertencem a esta classe.

Geralmente, para calcular a que distância uma ocorrência está da outra, é utilizada a distância Euclidiana .

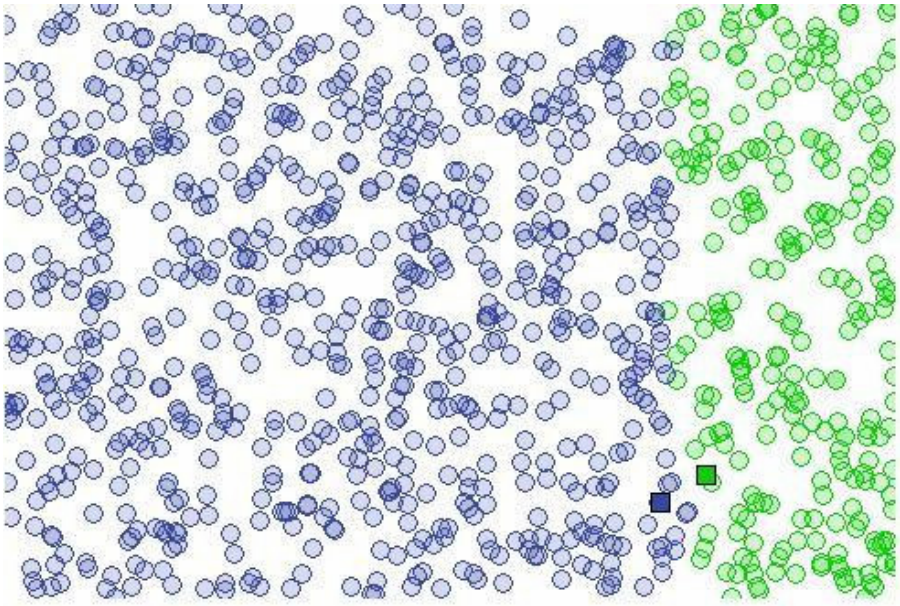
Após o cálculo das distâncias, o algoritmo calcula o centróide para cada uma das classes. Conforme as iterações do algoritmo, o valor de cada centróide é refinado pela média das distâncias de cada ocorrência pertencente a este centróide. Com isso, o algoritmo gera novos *k* centróides. O resultado é apenas a pertinência final de cada padrão aos aglomerados.

A condição de parada pode ser determinada pelo número de iterações desejadas, ou até o momento em que as coordenadas dos centros de agrupamentos não se alterarem durante mais de uma iteração.

Para ilustrar melhor a execução do algoritmo, segue os passos para sua execução:

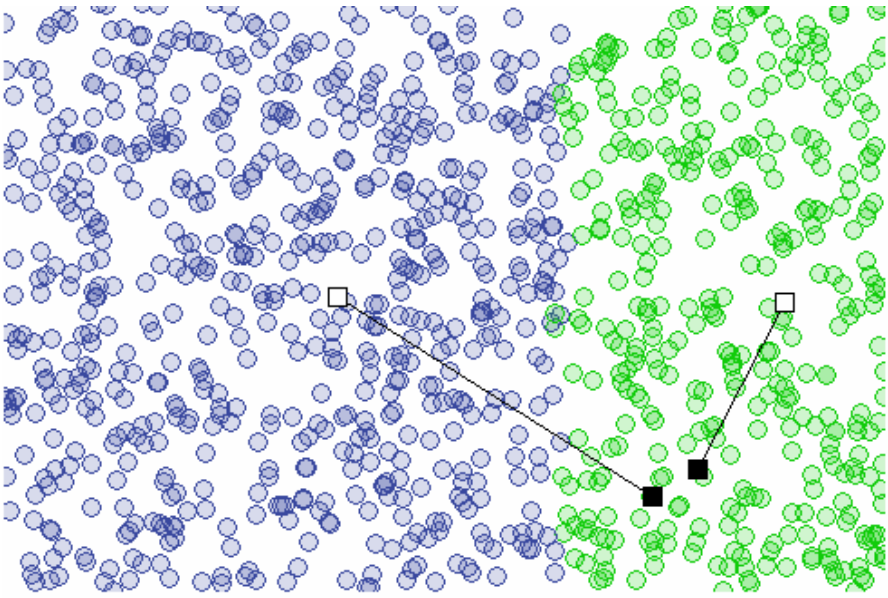
1) Definir o número *k* de aglomerados.
2) Escolher valores iniciais aleatórios para os centróides $\mu_{i.}$ (Centro de um conjunto de pontos x_j, pertencentes ao grupo $S_{i\ (i=1...k)}$).
3) Cada ponto x_j, é associado ao grupo S_i cujo centróide μ_i seja o mais próximo de x_j.
4) Os centróides de cada grupo S_i são recalculados com base nos pontos associados a eles.
5) Repete-se os passos 3 e 4 até a convergência (os centróides não mudam mais de lugar).

A Figura 2.6 apresenta um exemplo de conjunto de pontos quaisquer num plano cartesiano 2D, onde o K-Médias foi aplicado para k=2, ou seja, dois centros de agrupamento, escolhidos aleatoriamente.



2.6 - Pontos do conjunto a ser agrupado e dois centróides escolhidos aleatoriamente (*k*=2).

Cada ponto processado será alocado para o centróide do qual estiver mais próximo. Para calcular a distância dos pontos aos centróides, geralmente, usa-se a distância Euclidiana. Após alocados os pontos, os centróides são recalculados. (Figura 2.7)



2.7 - Primeira iteração do algoritmo, depois de iniciar o agrupamento, novos centróides são escolhidos.

O K-Médias encerra a execução quando os centróides convergirem, não mudando mais de lugar.

2.5. Redes Neurais Artificiais

Lippmann (1997) citado por Filho (2003) afirma que redes neurais artificiais são sistemas inspirados nos neurônios biológicos e na estrutura massivamente paralela do cérebro, com capacidade de adquirir, armazenar e utilizar conhecimento experimental.

Segundo Braga et al(2007) e Haykin(2001), redes neurais artificiais são sistemas paralelos distribuídos compostos por unidades de processamento simples(neurônios artificiais) que calculam determinadas funções matemáticas (normalmente não-lineares).

Uma rede neural artificial assemelha-se ao cérebro em dois aspectos:

1. O conhecimento é adquirido pela rede através de um processo de aprendizagem.

2. As forças das conexões entre neurônios, conhecidas por pesos sinápticos, é utilizada para armazenar conhecimento.

As redes neurais artificiais possuem a capacidade de aprender a partir de exemplos e de fazer interpolações e extrapolações do que aprenderam (BRAGA et al.,2007). A principal virtude de uma rede neural é sua capacidade de generalização.

Para Haykin (1999), "uma rede generaliza bem quando o mapeamento entrada-saída computado pela rede for correto (ou aproximadamente correto) para dados de teste não utilizados para criação ou treinamento da rede".

Existem vários tipos de redes neurais, dentre eles se destacam:

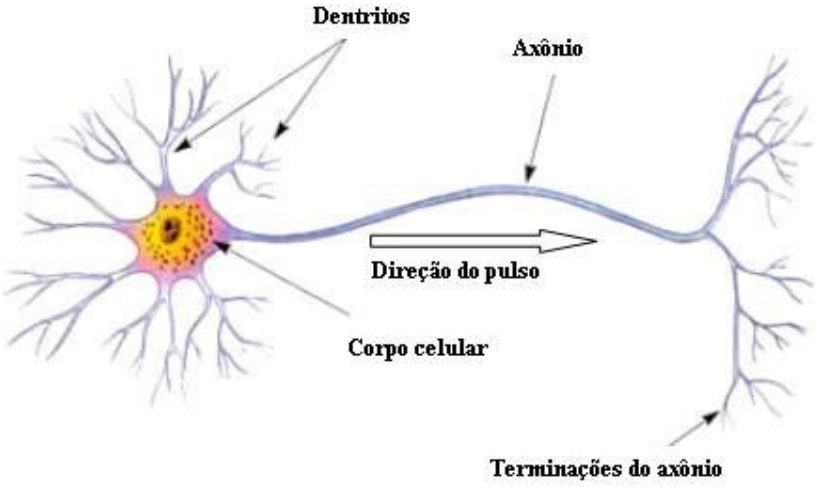
- Redes *feedforward* de camada única: Esta é a forma mais simples de uma rede em camadas, tem-se a camada de entrada de nós fontes (por onde são passados os dados de entrada), que se projeta sobre uma camada de saída (de neurônios). O termo "camada única" é usado apenas para designar a camada de saída, assim a camada não é contada. A denominação *feedforward* deve-se ao fato de os dados fluírem num sentido único: da camada de entrada para a camada de saída.

- Redes *feedforward* com múltiplas camadas: Neste tipo de rede os dados também fluem num único sentido. No entanto, podem existir uma ou

mais camadas ocultas de neurônios em camadas intermediárias. Desta forma a rede torna-se capaz de extrair estatística de ordem elevadas, melhorando assim o ajuste dos pesos.

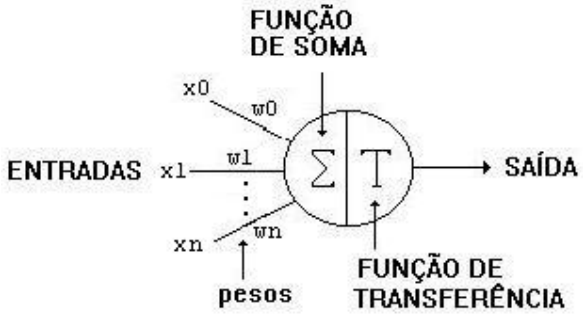
Uma rede neural é formada por vários neurônios artificiais, que simulam o funcionamento dos neurônios biológicos. Um neurônio biológico é tipicamente mais lento que as portas lógicas utilizadas pelos computadores convencionais. Enquanto os eventos em um circuito de silício acontecem na ordem de nanossegundos (10^{-9}s), no cérebro humano os eventos são da ordem de milissegundos (10^{-3}s). No entanto essa taxa de operação lenta é compensada pelo grande número de neurônios, com conexões maciças entre si (HAYKIN,2001).

Um neurônio biológico é composto por três estruturas básicas: dendritos, axônio e corpo celular. Os dendritos são especializados em receber estímulos e repassar ao corpo celular; o corpo celular recebe estímulos dos dendritos, processa-os e repassa para o axônio, que gera e conduz potencial de ação ou impulsos (*spikes*) aos neurônios vizinhos, conectados a ele por meio das *sinapses.* A Figura 2.8 mostra um neurônio biológico de maneira simplificada.



2.8 - Estrutura de um neurônio biológico

O modelo de neurônio artificial proposto por McCulloch e Pitts em 1943, denominado modelo MCP, consistiu numa tentativa de emular o que se sabia a respeito da estrutura e do funcionamento do neurônio biológico na época. A Figura 2.9 mostra o esquema do neurônio proposto.



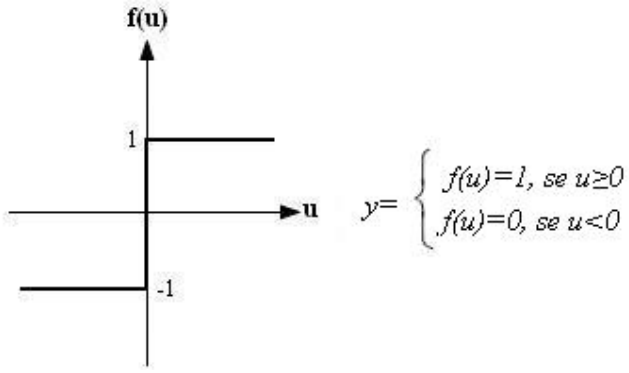
2.9 - Neurônio artificial proposto por McCulloch e Pitts em 1943

As entradas simulam os dendritos, e recebem valores de neurônios anteriores; o corpo celular é representado pela função de soma e pela função de transferência ou função de ativação; a saída simula o axônio e é ligada na entrada de outros neurônios.

A função de soma é definida pelo somatório da Equação 2.1.

$$u = \sum_{i=1}^{n} x_i w_i \quad (2.1)$$

A função de ativação é responsável por gerar a saída *y* do neurônio a partir da aplicação da função de soma sobre os valores do vetor de peso $w=(w_1,w_2,w_3,...,w_n)$ e de entrada $x=(x_1,x_2,x_3,...,x_n)$. Várias funções de ativação podem ser usadas para diferentes propósitos. A função de ativação do MCP é a função degrau, aplicada ao valor da função soma (u). A função degrau é definida conforme a Figura 2.10.



2.10- Definição da função de limiar

Um único neurônio possui capacidade limitada, sendo capaz de traçar apenas um hiperplano, para o caso *n*-dimensional, como superfície de decisão. Para superar esta

limitação e poder lidar com problemas mais complexos são criadas redes com vários neurônios.

O neurônio possui ainda uma entrada de valor constante, denominada *bias*. O *bias* tem o papel de aumentar ou diminuir a influência do valor das entradas. Geralmente usa-se a notação x_0 para descrevê-lo.

2.5.1. Perceptron

O trabalho original de McCulloch e Pitts em 1943 conseguiu representar um neurônio biológico e sua capacidade computacional através de várias topologias de rede com capacidade de execução de funções booleanas (BRAGA, 2007).

Em 1958 Frank Rosenblatt propôs o conceito de aprendizado em redes neurais artificiais, até então ausente nesta técnica. O modelo proposto por ele era composto por uma estrutura que continha neurônios do tipo MCP e uma regra de aprendizagem, que tratava de ajustar os parâmetros livres dos neurônios.

Rosenblatt provou que, se os vetores usados para treinar o perceptron são retirados de duas classes linearmente separáveis, então o algoritmo converge e posiciona a superfície de decisão na forma de um hiperplano entre as duas classes (HAYKIN,1999). O perceptron é a forma mais simples de separação para classificação de padrões ditos linearmente separáveis.

Este tipo de neurônio, da mesma forma que o MCP, recebe um vetor de atributos $x=(x_1,x_2,x_3,...,x_n)$, de *n* dimensões, sendo que a função de limiar *u* é a soma ponderada das entradas, conforme descreve a Equação 2.2.

$$u(x) = w_o + \sum_{i=1}^{n} w_i x_i \quad (2.2)$$

O modelo do perceptron se baseia na idéia de que, tem-se disponível um conjunto de dados de treinamento {x(i),d(i)} e um conjunto de dados de teste {x(i),d(i)}, sendo d(i)($\in${0,1}) o valor da saída desejada de y(x(i)) se o vetor de pesos *w* for escolhido corretamente. O treinamento do *perceptron* ajusta os pesos de maneira que consiga identificar padrões nos dados do conjunto de treinamento e aplicá-los ao conjunto de dados de teste. Pelos exemplos de treinamento um algoritmo de aprendizagem pode ser aplicado

iterativamente para estimar o valor correto de *w*. De acordo com o Teorema da Convergência de Rosenblatt, a atualização dos pesos pela Equação 2.3 leva sempre a uma solução, caso as classes em questão sejam linearmente separáveis (BRAGA,2007).

$$w(t+1) = w(t) + \eta * (d-y) * x(t) \quad (2.3)$$

Onde :

- *w* é o vetor de pesos;
- *t* é o número da iteração;
- η é a taxa de aprendizado;
- *d* é a saída desejada,
- *y* é a saída obtida da rede;
- *x* é o vetor de entradas;

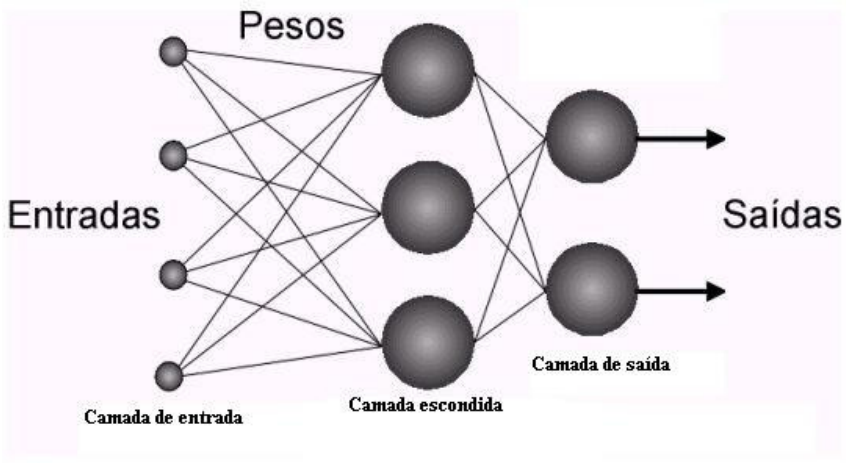
Na Equação 2.3, η representa a taxa de aprendizado. Esta taxa determina quanto os pesos deverão variar a cada ajuste. Os ajustes só serão feitos caso haja diferença entre a saída obtida e a saída desejada.

O uso do *perceptron* para resolução de alguns problemas se faz inviável por não apresentar convergência para conjuntos de dados não linearmente separáveis. Esta restrição induz à utilização de redes neurais de múltiplas camadas, para aumentar o poder de processamento e ampliar a aplicabilidade das redes neurais aos problemas reais.

2.5.2. Multilayer Perceptron

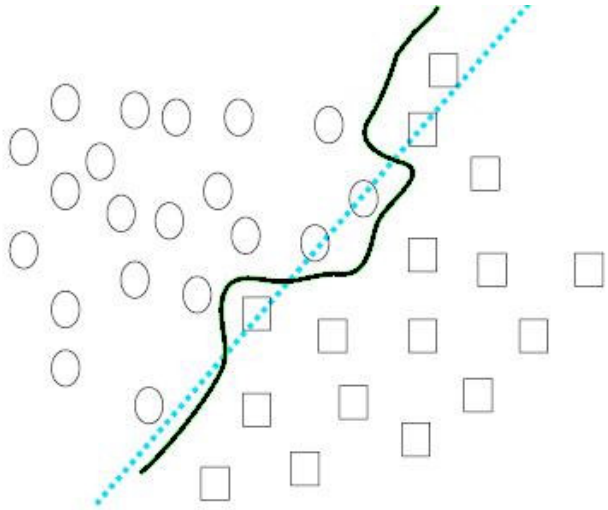
As redes MLP, como são chamadas as redes *Multilayer Perceptron* são as mais utilizadas comercialmente, por representarem bem a não linearidade dos problemas reais.

Segundo Haykin (1999) uma rede MLP consiste num conjunto de unidades sensoriais que constituem a camada de entrada, uma ou mais camadas ocultas de nós computacionais e uma camada de saída conforme Figura 2.11.



2.11- Estrutura de uma rede *Multilayer perceptron*

A Figura 2.12 ilustra de maneira simplificada a diferença no mapeamento de dados não lineares. Os dados desse conjunto fictício são divididos em duas classes: bolinha e quadrado. Como alguns exemplares da amostra estão misturados, uma reta não seria capaz de separar essas classes. Enquanto um único *perceptron* só é capaz de traçar um hiperplano que não separa corretamente os dados (linha pontilhada da Figura 2.12), a rede MLP (linha contínua da Figura 2.12) consegue mapear, de maneira mais precisa, as classes dos dados de entrada.



2.12 - Superfície de separação para um conjunto de dados não linearmente separável.

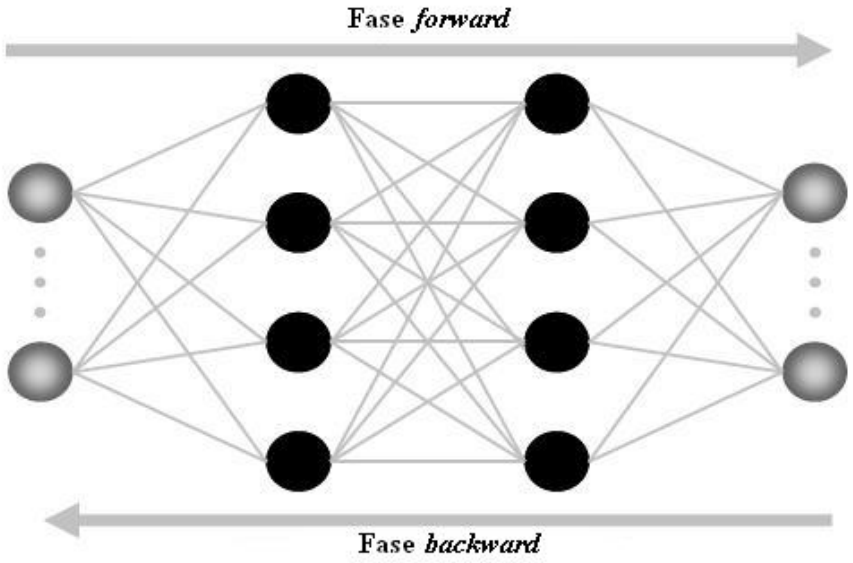
Uma MLP providencia um mapeamento não-linear entre entradas e saídas. É provado que com um número suficiente de neurônios escondidos, uma MLP com

aproximadamente duas camadas escondidas é capaz de aproximar um mapeamento complexo dentro de um intervalo de tempo finito e exeqüível (KRÖSE & VAN DER SMAGT,1996 apud OLIVEIRA,2007).

2.5.3. Treinamento

De acordo com Haykin (1999) o treinamento da rede neural é o processo pelo qual os parâmetros livres são adaptados por meio da estimulação do ambiente na qual a rede está inserida. Na sua forma básica o treinamento da rede neural consiste num processo iterativo, onde os pesos das entradas vão sendo ajustados e melhorados gradativamente.

Treinar uma rede consiste ajustar os pesos de suas entrada tendo como base as entradas do conjunto de treinamento e suas saídas desejadas. O algoritmo mais comumente utilizado em redes MLP é o *backpropagation*. Este algoritmo opera em duas fases e em casa fase a rede é percorrida num sentindo. A fase *forward* trata de enviar os resultados do processamento para a saída da rede e a fase *backward* compara a saída obtida com a saída desejada para atualizar os pesos das sinapses. A Figura 2.13 mostra o fluxo de execução do *backpropagation*.



2.13 - Fluxo de execução do *Backpropagation*

O algoritmo *backpropagation* é uma generalização da Regra Delta. A Regra Delta mede a distância entre a saída obtida e a saída desejada, então são feitos ajustes de modo a reduzir essa distância. A Equação de atualização dos pesos no *backpropagation* é dada pela Equação 2.4.

$$W_{ji}(t+1) = w_{ji}(t) + \eta * \delta_j(t) * x_i(t) \quad (2.4)$$

Onde:

- $w_{ji}(t)$ é o peso da conexão entre os neurônios *i* e *j* na iteração *t*;

- η é taxa de aprendizagem;

- δ_j é o erro do neurônio na saída do neurônio *j*.

- $x_i(t)$ representa o vetor dos valores de entrada na iteração *t* do neurônio *i*;

O erro δ_j é calculado de maneira diferente para neurônios da camada escondida e da camada de saída. Para os neurônios da camada de saída o erro é dado pela Equação 2.5, onde $f'(net_j)$ é a derivada da função de ativação utilizada no neurônio *j*.

$$\delta_j=(d_j - y_j)\, f'(net_j) \quad (2.5)$$

Para os neurônios da camada escondida, o erro é dado pela Equação 2.6.

$$\delta_j= f'(net_j) \sum_l \alpha_l w_{lj} \quad (2.6)$$

Cada combinação de pesos e limiares corresponde a um ponto na superfície. Considerando que a altura de ponto é diretamente proporcional ao erro associado a este ponto, a solução está nos pontos mais baixos da superficie.

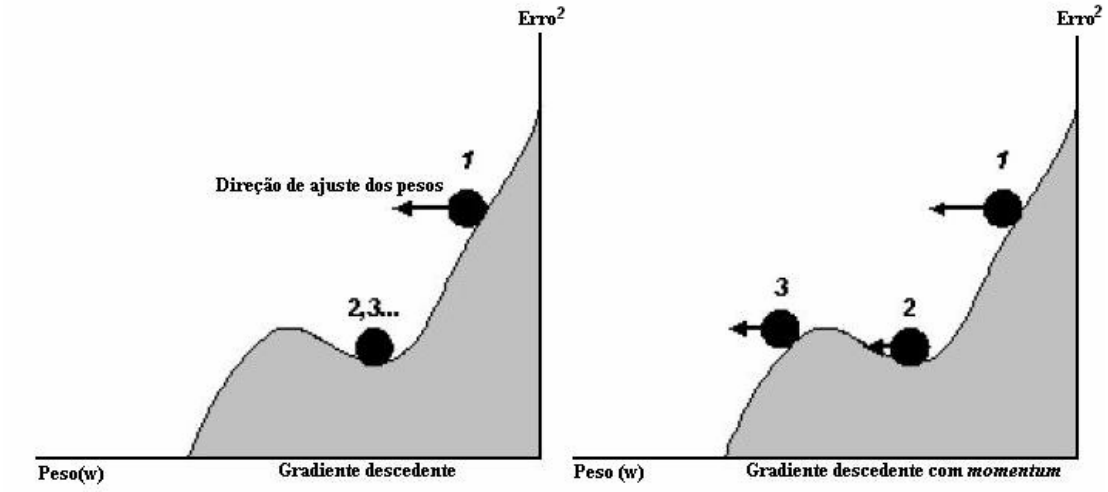
O algoritmo *backpropagation* procura minimizar o erro obtido pela rede ajustando pesos e limiares para que eles correspondam a um ponto na superfície da solução. Para isto ele utiliza um método de gradiente descendente. O gradiente descendente de uma função está na direção e sentido onde a função varia mais rapidamente, assim a rede caminha na direção de maior decréscimo do erro.

A função de ativação da rede precisa ser não linear e diferenciável, visto que o gradiente precisa ser calculado, direcionando o ajuste dos pesos, que são atualizados proporcionalmente à taxa de aprendizado.

Na superfície de erro, em regiões de baixo gradiente ou que possuem mínimos locais, a convergência tende a ser mais difícil (BRAGA,2007). Dentre as técnicas para acelerar a convergência estão a utilização de uma taxa de aprendizado decrescente, que diminui a intensidade dos ajustes nos pesos, e a adição do termo *momentum* na Equação de ajuste dos pesos, (*vide* Equação 2.7).

$$w_{ji}(t+1) = w_{ji}(t) + \eta * \delta_j(t) * x_i(t) + \beta[w_{ij}(t) - w_{ij}(t-1)] \quad (2.7)$$

Onde β é o momentum. A adição do momentum à equação de ajuste dos pesos aumenta a velocidade de convergência em regiões de descida da superfície do erro e também evita que o algoritmo fique estagnado em mínimos locais, sendo que em algum outro lugar do espaço de pesos existe um mínimo global. Evidentemente não é desejado que o treinamento termine num mínimo local. (*vide* Figura 2.14).



2.14 - Comparação da evolução do erro no *backpropagation* com *momentum*

Normalmente o momentum é ajustado entre 0,5 e 0,9. Há ainda casos em que o *momentum* é ajustado durante o treinamento, sendo muito pequeno no início e aumentado quando o erro estiver estabilizado.

O algoritmo *backpropagation* é muito lento para várias aplicações, mesmo em problemas simples, os dados de treinamento precisam ser apresentados inúmeras vezes à rede. Para sanar esta deficiência, foram criadas algumas variações do *backpropagation.* Dentre elas existe um grupo denominado de *backpropagation com gradiente conjugado.*

O algoritmo básico *backpropagation* ajusta os pesos na direção na qual o erro diminui mais rapidamente. No entanto, isso não garante que convergirá mais rápido. Na maioria dos algoritmos de aprendizado, a taxa de aprendizado é fixa. Na maioria dos métodos de gradiente conjugado, a taxa de aprendizado varia a cada iteração. Nestes métodos uma pesquisa é feita ao longo da direção do conjugado da função, a fim de determinar a taxa de aprendizado. Estes algoritmos apresentam, geralmente, convergência mais rápida que os algoritmos que usam o negativo do gradiente para atualizar os pesos.

2.5.4. Conclusão

As redes neurais são capazes de, a partir de dados previamente conhecidos, extrair conhecimento para aplicar sobre dados não conhecidos. No caso deste trabalho, foi aplicada a técnica de redes neurais sobre seqüências RNAm para predição de SITs.

3. METODOLOGIA

O desenvolvimento do presente trabalho se dividiu nas seguintes etapas:

- Estudo do problema, com enfoque na biologia envolvida;
- Seleção dos dados;
- Aplicação do K-Médias;
- Implementação da rede neural;
- Análise dos resultados.

3.1. Conjunto de dados

Para compor este trabalho, foram selecionadas todas as seqüências de RNAm do *Mus musculus* (camundongo) do banco de dados RefSeq[2] do NCBI[3]. Optou-se pelo *Mus musculus* por já haver estudos com essa espécie na literatura, em Nobre(2007). O NCBI é um órgão mantido pelo governo americano que disponibiliza, para acesso público, dados de pesquisas na área de biotecnologia. No NCBI são disponibilizados genoma e proteoma de inúmeras espécies. Alguns bancos de dados no NCBI, como o GenBank[4], permitem, além do acesso, que qualquer pessoa poste dados na sua base de dados. No entanto, neste trabalho, utilizou-se seqüências do banco RefSeq, o qual possui as seguintes características:

- Não redundância;
- Dados são atualizados e representam as informações mais recentes;
- Dados são validados e têm formatos padronizados;
- Dados curados pela equipe do NCBI.

[2]www.ncbi.nlm.nih.gov/refseq/

[3]www.ncbi.nlm.nih.gov

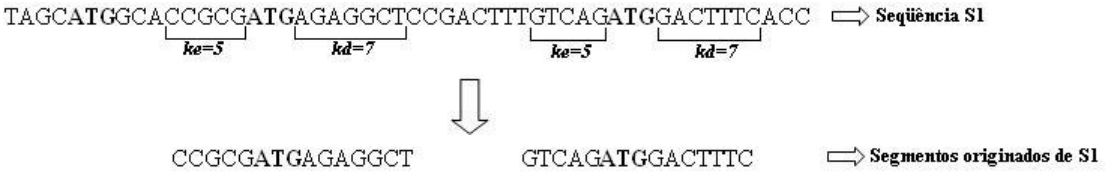
[4]www.ncbi.nlm.nih.gov/Genbank/

Dentre estas características, a mais importante para este trabalho é o fato de o RefSeq possuir dados curados. O termo "curado" indica que os dados foram verificados, inspecionados, e isto garante a confiabilidade que este trabalho exige.

3.2. Formatação dos dados

Antes de utilizar os dados, o arquivo originado do RefSeq foi formatado por um programa implementado em Java, que fez todas as manipulações dos dados necessárias para que o formato do arquivo se adequasse ao formato exigido nas entradas do Weka (.arff) e do Matlab (.txt e outros).

A análise dos SITs foi feita analisando-se cada ATG encontrado. No total foram encontrados 16.844 ATGs. As seqüências de RNAm têm tamanho muito variado, a menor seqüência possui 522 nucleotídeos, enquanto a maior possui 13.939. Cada seqüência foi divida em segmentos, cada segmento representava um ATG analisado. Para fazer a análise dos segmentos, dividiu-se os segmentos obtidos em 2 grupos: segmentos positivos e segmentos negativos. Os positivos eram em segmentos com ATG que inicia a tradução, e negativos os que possuem ATG que não inicia. Para que pudesse ser extraído o conhecimento dos padrões das seqüências, além do ATG, alvo da análise, foram selecionadas faixas de nucleotídeos antes e depois deste ATG. Assim considera-se *ke* sendo o número de nucleotídeos à esquerda do ATG e *kd* o número de nucleotídeos à direita do ATG. Há casos em que um segmento pode possuir mais de um ATG, no entanto o ATG analisado foi sempre o que estava entre a faixa *ke* e *kd.* A Figura 3.1 ilustra o processo de obtenção dos segmentos para análise, neste caso tem-se *ke*=5 e *kd*=7.



3.1 - Exemplo de corte de uma seqüência

Somente os ATGs que possuíam quantidade de nucleotídeos à esquerda maior ou igual à *ke* e que possuíam quantidade de nucleotídeos à direita maior ou igual à *kd* foram usados, os demais foram descartados. A Figura 3.1 mostra um ATG que não gerou um

segmento, o primeiro ATG da seqüência possuía à esquerda apenas quatro nucleotídeos, tamanho insuficiente, visto que neste caso o *ke*=5.

Após a obtenção dos segmentos, estes foram codificados. Cada nucleotídeo foi representado por 4 dígitos binários : A=0001, T=0010, C=1000 e G=0100. A partir dessas manipulações foram gerados os arquivos de entrada para o Weka (.arrf) e para o Matlab[5] (.txt).

Foram aplicados dois métodos na classificação dos segmentos, a saber:

- K-Médias, com *k=2*, utilizando a implementação da Weka;
- Rede *feedforward* com unidades ocultas, utilizando a *toolbox* de redes neurais do Matlab 6.0.

Para a avaliação dos métodos a métrica utilizada foi o número de segmentos classificados corretamente. A aplicação do K-Médias mostrou indícios de que a predição de SITs não se trata de um problema linearmente separável, uma vez que o resultados obtidos não foram satisfatórios, apresentando um erro muito alto na classificação dos segmentoss. É importante ressaltar que não foi feita nenhuma prova matemática para provar que os dados não eram linearmente separáveis, é apenas um indício. Como o K-Médias não apresentou bons resultados para a resolução deste problema, outras técnicas mais sofisticadas foram ser aplicadas. Neste trabalho optou-se por aplicar as técnicas de redes neurais artificiais. Os detalhes dos resultados serão apresentados no capítulo seguinte.

3.3. K-Médias

Para aplicação do K-Médias foi a utilizado o algoritmo já implementado na Weka[6]. A Weka (*Waikato Environment for Knowledge Analysis*) é uma coleção de ferramentas *open source* de *data minig,* desenvolvida pela Universidade de Waikato, na Nova Zelândia. A Weka contém ferramentas para pré-processamento de dados, classificação, regressão, *clustering*, criação de regras de associação e visualização. Pode-se utilizar a Weka como

[5] http://www.mathworks.com/

[6] http://www.cs.waikato.ac.nz/ml/weka/

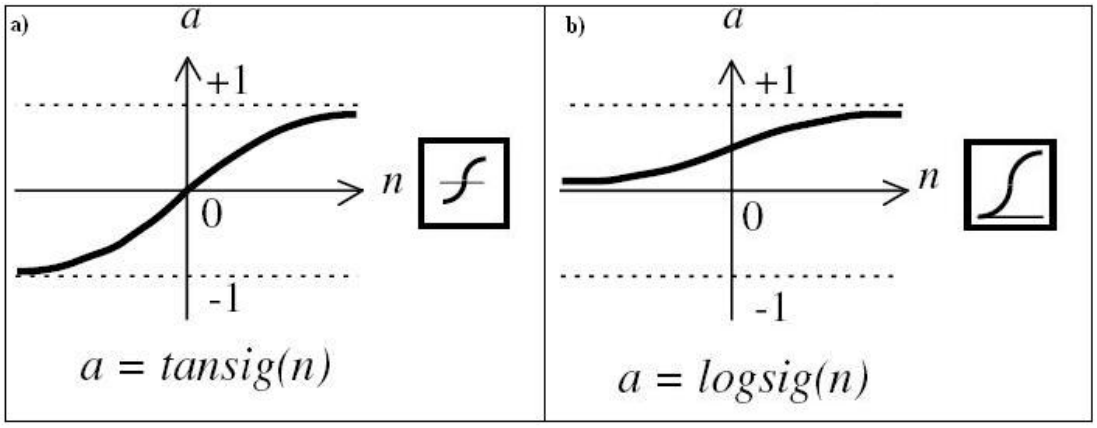
uma biblioteca java ou então utilizar a interface gráfica disponível. Neste trabalho utilizou-se a interface gráfica.

Para avaliar a aplicabilidade do K-Médias ao problema em estudo, foram testadas as entradas com várias configurações no tamanho da faixa em torno do ATG analisado. Com essa variação tornou-se possível constatar uma baixa taxa de acerto nas classificações. Os resultados da aplicação deste métodos estão detalhados no próximo capítulo.

3.4. Implementação da Rede Neural

Para a implementação da rede neural utilizada neste trabalho utilizou-se a *toolbox* de redes neurais do Matlab 6.0.0.88 release 12.

Inicialmente foi criada uma rede neural simples, do tipo *feedforward*, com 10 neurônios na camada oculta e apenas um neurônio na camada de saída, pois apenas duas saídas são passiveis de serem avaliadas – sim (ATG é início de tradução) e não (ATG não é início de tradução). As funções de ativação para esta primeira configuração eram *tansig* (tangente sigmoidal) para a camada escondida e *logsig* (função logística) para a camada de saída (*vide* Figura 3.2).



3.2 - Funções de ativação das camadas de neurônios

Após criação a rede, o passo seguinte foi obter as entradas para treinamento e validação da rede. Fez-se necessária então a escolha do tamanho dos segmentos a serem tomados como entrada. Dependendo do tamanho da janela de nucleotídeos escolhida era gerada uma quantidade de segmentos diferentes, sempre utilizando 90% para treinamento e 10% para teste da rede. A entrada da rede era sempre um vetor de 0s e 1s. Por exemplo, o

códon CCG seria codificado como 1000 1000 0100, o vetor de entrada teria então 12 dimensões.

Para Pedersen & Nielsen (1997), o tamanho da faixa de nucleotídeos ideal para predição de SITs é de 203, sendo 100 bases antes do ATG e 100 bases depois, onde obtiveram 85% de acerto.

Nobre (2007) utilizou SVM para predição de SITs e obteve acurácia da ordem de 98,8%. Neste trabalho 12 nucleotídeos à esquerda e 12 à direita forneceram um resultado tão bom quanto 99 nucleotídeos em torno do ATG.

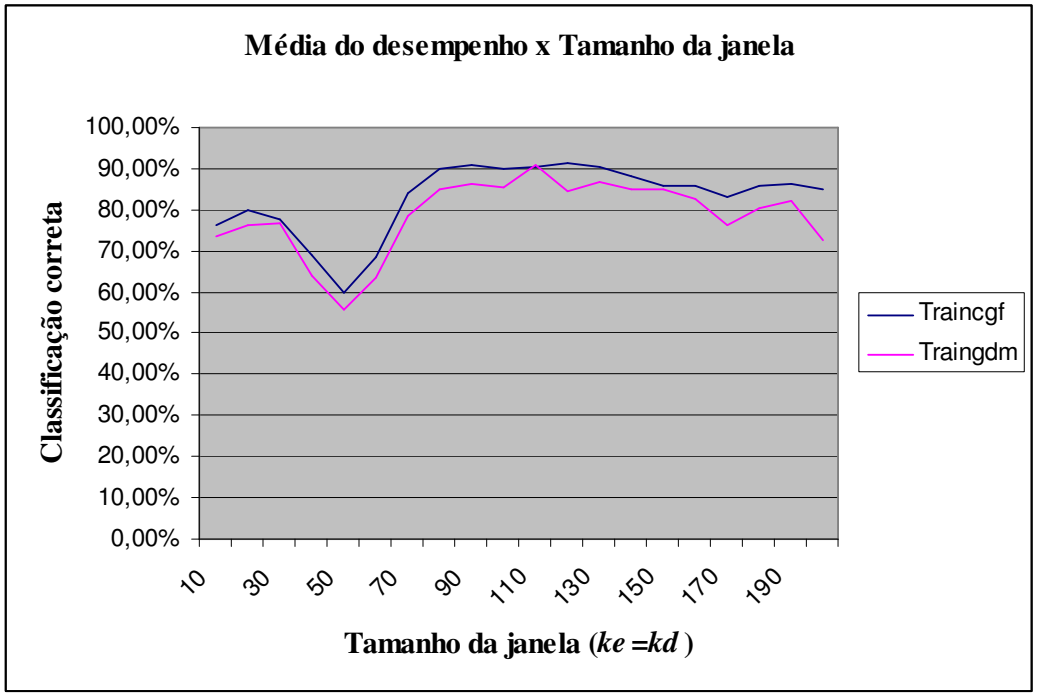
Para verificar qual era a conFiguração mais adequada para a rede neural e para os parâmetros de treinamento, fixou-se o tamanho dos segmentos em 100 nucleotídeos à esquerda e 100 à direita, como citado em Pedersen & Nielsen (1997). O conjunto de segmentos foi dividido de maneira que, 90% fossem usados para treinamento da rede e 10% para validação.

O processo de treinamento da rede foi empírico. Os ajustes na rede foram feitos baseando-se nos resultados obtidos com a rede já treinada. Vários algoritmos de treinamento foram utilizados. Os parâmetros ajustados foram a taxa de aprendizado, o número de épocas e erro desejado. Para a conFiguração da rede foram testadas redes que variavam de 10 à 35 neurônios na camada escondida.

Ao fim dos ajustes, observou-se que os melhores resultados eram alcançados utilizando-se 20 neurônios na camada escondida, 400 épocas, taxa de aprendizado de 0.9 e taxa de erro desejado = 0.0001. Em relação aos algoritmos de treinamento, os que obtiveram melhor desempenho foram os métodos do gradiente descendente com *momentum* (*traingdm)* e gradiente conjugado (*traincgf*).

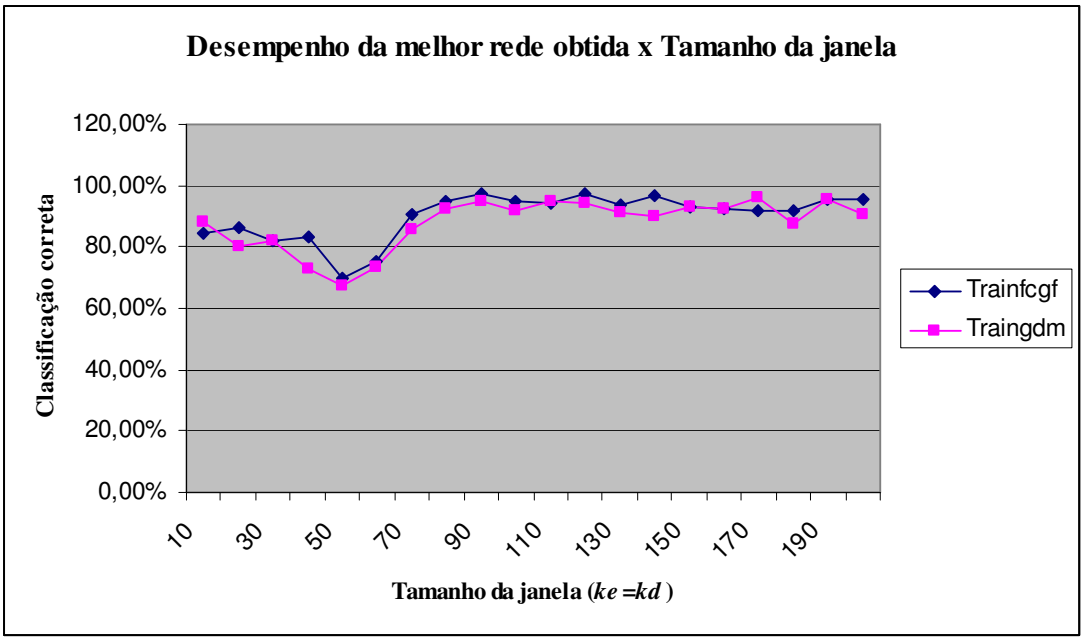
Para que o desempenho da rede fosse melhorado, optou-se por analisar de maneira mais detalhada redes treinadas com os algoritmos que se mostraram mais eficazes nos testes iniciais- *traincgf* e *traingdm*. Foram criados conjuntos de treinamento e teste com janela de 10,20,30 até 200 nucleotídeos à esquerda e à direita. Para cada tamanho de janela foram criadas 10 redes com o algoritmo de treinamento *traincgf* e 10 com o *traingdm.* Cada rede foi treinada e validada junto ao conjunto de teste. O gráfico da Figura 3.3 mostra

a média do desempenho das 10 redes com cada algoritmo para os tamanhos de janela selecionados.



3.3 - Gráfico comparativo do desempenho dos algoritmos *traincgf* e *traingdm*, em 10 redes criadas com cada algoritmo, para diferentes tamanhos de janela.

O gráfico 3.3 mostra a evolução das redes em relação ao tamanho da janela de nucleotídeos dos segmentos de entrada. Em praticamente todos os casos as redes que foram treinadas com o algoritmo *traincgf* tiveram um desempenho melhor que as redes que usavam o *traingdm*. Foi avaliada também, dentre as 10 redes criadas para cada algoritmo e tamanho de janela, qual obteve o melhor desempenho. O gráfico da Figura 3.4 mostra novamente uma vantagem do *traincgf* em relação ao *traingdm*.

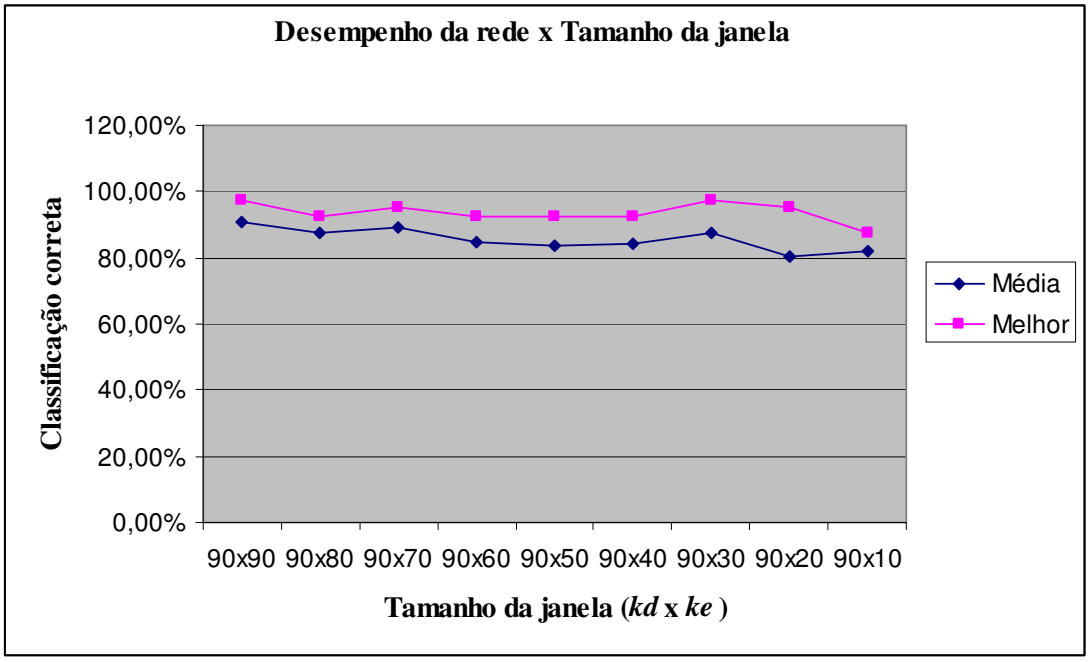


3.4 - Gráfico comparativo da melhor rede criada para cada tamanho de janela.

O melhor desempenho dentre todas as redes criadas foi o da rede que usava o *traincgf* e que foi testada para segmentos com janela de tamanho 90, que conseguiu classificar corretamente 97,50% dos segmentos do conjunto de teste. No decorrer dos testes, foi possível observar também que, o tempo de treinamento com o *traingdm* é maior do que com o *traincgf*.

Visando diminuir o tempo de treinamento da rede, foi feita uma diminuição da faixa de nucleotídeos à direita do ATG. A opção por diminuir a faixa depois do ATG e não a que o antecede seria mais passível de não afetar a classificação dos segmentos, por conta do escaneamento que ocorre durante a tradução no RNAm de eucariotos.

Foram criadas 10 redes para cada tamanho de janela. O tamanho da faixa depois do ATG foi diminuído gradativamente. O gráfico da Figura 3.5 mostra que com a janela de tamanho *ke*=90 e *kd*=30 , obteve-se o mesmo resultado (97,50% de classificação correta) do que com a janela *ke*=90 e *kd*=90, em relação ao melhor resultado obtido na criação das 10 redes.



3.5 - Gráfico comparativo do desempenho das redes para diferentes tamanhos de janela.

Com base nos resultado apresentados no gráfico da Figura 3.5, e com a diferença de desempenho mínima apresentada entre as faixas com *kd*=90 e *kd*=30, constata-se que a utilização de uma janela de *ke*=90 e *kd*=30 se mostra uma boa conFiguração para os segmentos, uma vez que apresenta bom desempenho nos testes e um custo computacional menor que os segmentos simétricos, de *ke*=90 e *kd*=90.

Com todos estes experimentos conclui-se que a melhor rede para ser empregada na resolução deste problema, dentre todas analisadas, seria uma rede do tipo *feedforward*, com 20 neurônios na camada oculta e algoritmo de treinamento *traincgf,* utilizando segmentos com uma faixa de 90 nucleotídeos à esquerda e 30 à direita.

4. RESULTADOS E DISCUSSÃO

Neste capítulo discute-se os resultados obtidos no trabalho. Primeiramente serão apresentados os resultados obtidos com o K-Médias, e logo após os resultados alcançados com a rede neural.

4.1. K-Médias

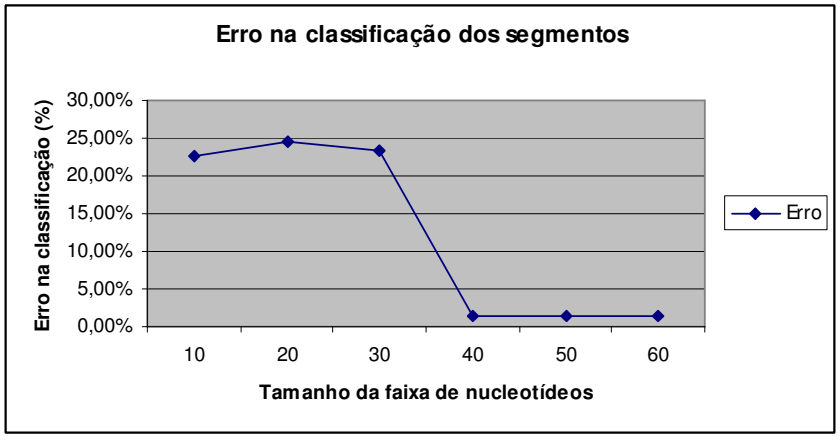
Inicialmente utilizou-se todos os segmentos obtidos na quebra das seqüências. A tabela 4.1 mostra os resultados obtidos.

Tabela 4.1 - Comparação entre a classificação real dos segmentos e a classificação do K-Médias aplicado a todos segmentos extraídos das seqüências

Faixa(*ke*/*kd*)		10/10	20/20	30/30	40/40	50/50	60/60
Desejado	Positivos	253	251	246	238	229	223
	Negativos	16522	16471	16403	16351	16282	16205
Obtido	Positivos	69	76	75	0	0	0
	Negativos	12920	12546	12692	16589	16611	16428

A tabela 4.1 mostra, no primeiro grupo, a quantidade real de segmentos positivos e negativos, e no segundo grupo, o número de segmentos positivos e negativos classificados corretamente pelo K-Médias.

Observa-se que a partir de 40 nucleotídeos à direita e à esquerda, todos os segmentos são agrupados num mesmo *cluster*. O gráfico da Figura 4.1 mostra a evolução do erro na classificação dos segmentos em relação ao aumento das faixas em torno do ATG.



4.1 - Erro na classificação dos segmentos utilizando o K-Médias com todos os segmentos gerados

Apesar de o erro evoluir assintoticamente, esse resultado não se mostra correto. A partir 40 nucleotídeos na faixa ao redor do ATG, o K-Médias agrupa todos os segmentos num só *cluster*, o *cluster* dos segmentos negativos. Como o número de segmentos positivos é menor que o número de segmentos negativos, então este valor baixo do erro (em torno de 1%), corresponde aos segmentos positivos que foram classificados como negativos erroneamente.

Este comportamento possivelmente deve-se ao fato dos dados de entrada não estarem balanceados em relação à quantidade de segmentos de cada grupo, uma vez que cada seqüência possui apenas um ATG que era início de tradução, naturalmente, o número de segmentos negativos era muito maior que o número de segmentos positivos.

Para se obter um conjunto de dados balanceado, tomou-se por base a quantidade de segmentos positivos, que era bem menor, e selecionou-se aleatoriamente no conjunto de segmentos de origem a mesma quantidade segmentos negativos. Dessa forma obte-ve um conjunto de dados balanceado, evitando que os métodos aplicados se comportassem de maneira tendenciosa, por conta do excesso de segmentos negativos.

Desta maneira, seria possível verificar se o K-Médias seria capaz ou não de resolver o problema da predição de SITs. A tabela 4.2 mostra o resultado obtido.

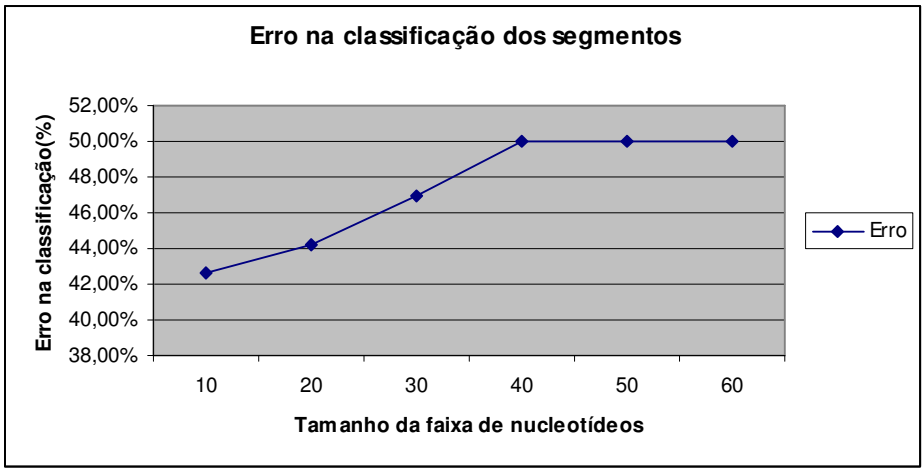
Tabela 4.2 - Comparação entre a classificação real dos segmentos e a classificação do K-Médias aplicado à quantidades iguais de segmentos positivos e negativos

Faixa(*ke*/*kd*)		10/10	20/20	30/30	40/40	50/50	60/60
Desejado	Positivos	253	251	246	238	229	223

	Negativos	253	251	246	238	229	223
Obtido	**Positivos**	69	76	75	238	229	223
	Negativos	221	204	186	0	0	0

A tabela 4.2 mostra, no primeiro grupo, a quantidade real de segmentos positivos e negativos, e no segundo grupo, o número de segmentos positivos e negativos classificados corretamente pelo K-Médias.

Observa-se que a partir de 40 nucleotídeos à direita e à esquerda, todos os segmentos são agrupados num mesmo *cluster*. O gráfico da Figura 4.2 mostra a evolução do erro em relação ao aumento das faixas.



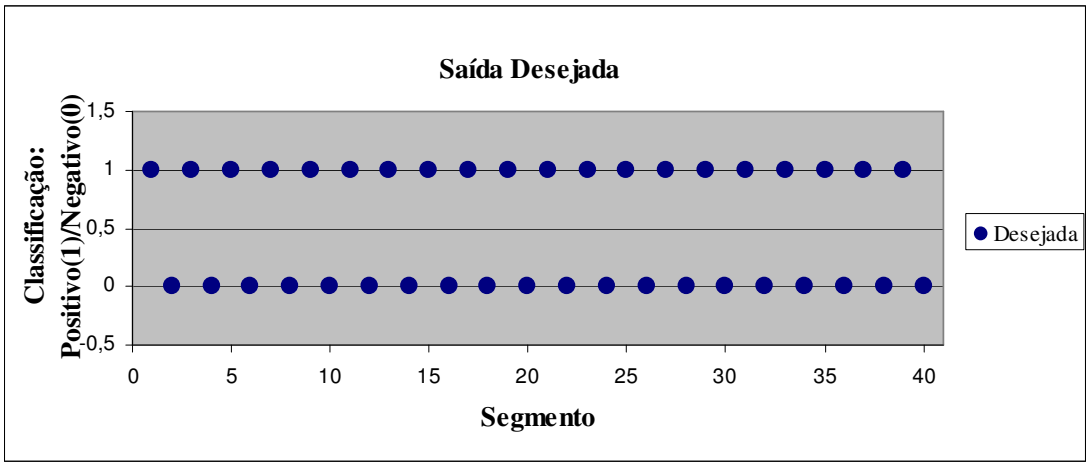
4.2 - Erro na classificação dos segmentos utilizando o K-Médias com quantidades iguais de segmentos positivos e negativos.

Conforme o gráfico 4.2 mostra, com quantidades iguais de segmentos positivos e negativos, o K-Médias possui uma alta taxa de erro. Este experimento é um indício, e não uma prova, de que o problema da predição de SITs não é um problema linearmente separável. Ou seja, os dados estão sobrepostos, de forma que não é possível separá-los por um hiperplano simples.

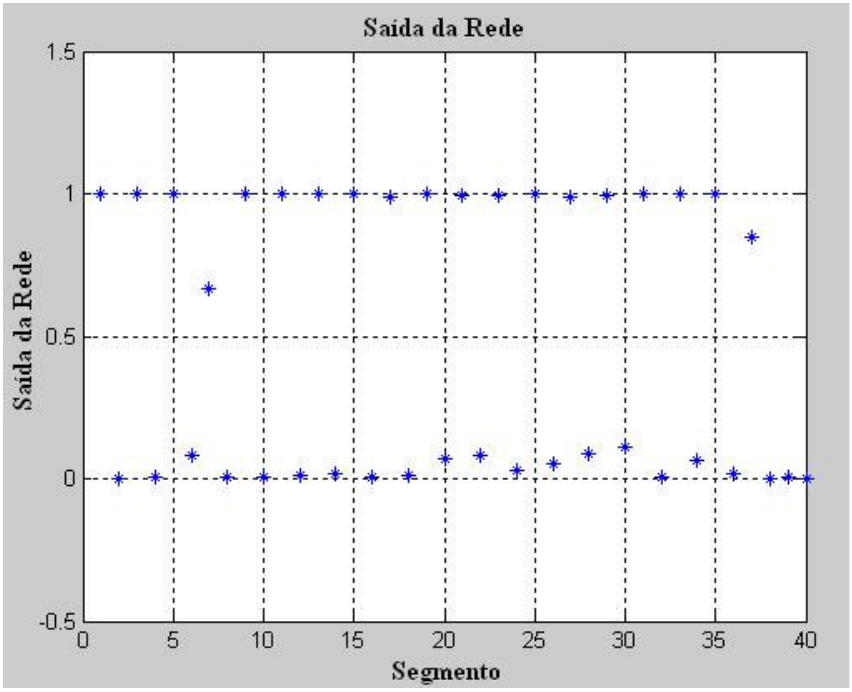
4.2. Redes Neurais

Após verificar a melhor configuração da rede e a melhor janela para o segmento ser analisado, a rede com esta configuração foi treinada. Para a janela de tamanho ke=90 e kd=30 foram gerados 396 segmentos, dos quais 356 foram usados para treinamento da rede

e 40 para testar a rede. Após o treinamento da rede, o conjunto de entrada de teste foi submetido à rede e sua saída foi comparada com o conjunto de saída de teste. A Figura 4.4 mostra a saída desejada, o conjunto de teste era composto por 40 segmentos.



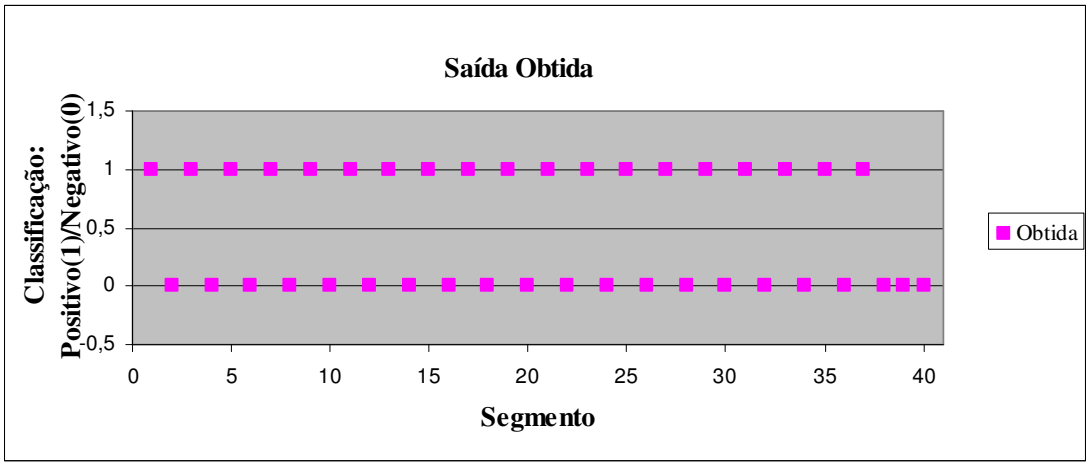
4.3 - Saída Desejada



4.4 - Saída da Rede

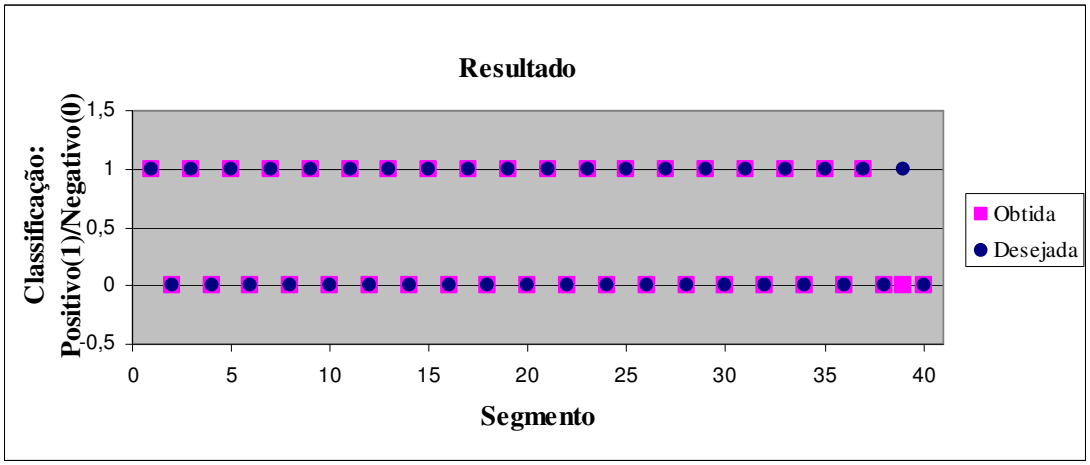
Sendo a função de ativação da camada de saída a *logsig*, o retorno seria um número no intervalo [0,1], como mostra a Figura 4.4. No entanto, para verificação do desempenho da rede, somente 0 (segmento negativo) e 1(segmento positivo) poderiam ser considerados. Para se adequar a isso, um arredondamento foi efetuado. Se a saída da rede fosse maior que

0.6, então a saída seria considerada como sendo 1; se a saída da rede fosse menor que 0.4, então a saída considerada seria 0; nos demais casos a saída seria considerada como sendo 0.5. Sempre que a saída para a classificação de um segmento fosse dada como 0.5, este segmento automaticamente seria contabilizado como sendo um segmento que foi classificado errado, isto porque as saídas válidas para a solução do problema são somente 0s e 1s.



4.5 - Saída da rede já arredondada.

A Figura 4.5 mostra dos valores da Figura 4.4 já arredondados, a saída obtida foi comparada com o resultado desejado. A métrica utilizada para avaliar o desempenho da rede neural foi contar a quantidade de segmentos do conjunto de teste que foram classificados corretamente.



4.6 - Resultado obtido com a rede neural.

Neste caso, observa-se pelo gráfico da Figura 4.6 que apenas um segmento foi classificado erroneamente, o que confere uma taxa de 97,50% de acerto para a rede neural.

5. CONCLUSÕES

Neste trabalho foi cria uma rede neural a fim de predizer SITs em seqüências de RNAm. Antes da implementação da rede neural foi feita a tentativa de resolução deste problema utilizando o método de aprendizado não-supervisionado K-Médias. Os resultados obtidos com este métodos mostraram indícios de que o problema envolvendo a predição de SITs não era linearmente separável, uma vez que a taxa de acerto na classificação dos segmentos foi muito baixa.

Várias redes neurais foram criadas, variando o número de neurônios, os algoritmos e seus parâmetros de treinamento e o tamanho da janela de nucleotídeos ao redor do ATG analisado. Ao fim das análises obteve-se, como melhor classificador, uma rede do tipo *feedforward*, com 20 neurônios na camada oculta, um neurônio na camada de saída, algoritmo de treinamento *traincgf*, e tamanho da janela sendo 90 nucleotídeos antes do ATG e 30 depois, onde alcançou-se uma taxa de acerto de 97,50% nos dados de teste.

Conclui-se assim que o problema de predição de SITs não se trata de um problema trivial, necessitando, portanto, da aplicação de técnicas mais elaboradas, como redes neurais artificiais, para se ter um resultado satisfatório.

Este trabalho deixa margem para futuros trabalhos. Propõe-se a utilização de uma codificação para as seqüências baseada em características físico-químicas dos nucleotídeos e também a utilização do método de treinamento multi-objetivo, que utiliza, além do erro quadrático como função de custo, a norma do vetor de pesos como segundo objetivo, proposto por Teixeira (2001).

6. REFERENCIAL BIBLIOGRÁFICO

ALBERTS, B., BRAY, D., JOHNSON, A., LEWIS, J., RAFF, M., ROBERTS, K., WALTER, P.. **Fundamentos da Biologia Celular – Uma introdução à biologia molecular da célula.** 3ª reimpressão, Porto Alegre: Editora Artmed, 1999.

BATISTA, G. E., **Um ambiente de avaliação de algoritmos de aprendizado de máquina utilizando exemplos**, dissertação de mestrado, Universidade de São Paulo, São Carlos, 1997.

BRAGA, A. P., CARVALHO, A. P., LUDERMIR, T. B., **Redes neurais artificiais: Teoria e aplicações**, LTC, 1–80 p., Rio de Janeiro, 2007.

CERVO, A. L., BERVIAN, P. A.. **Metodologia Científica**. 5ª ed. São Paulo: Pearson Prentice Hall, 2002.

COLE, M. R., **Clustering with Genetic Algorithms**, dissertação, University of Western, Austrália, 1998.

FILHO, A. M., **Um modelo para a implementação de consciência em robôs**, monografia de graduação, Universidade Federal de Santa Catarina, Florianópolis, 2003.

FUCHS, R.. **From sequence to biology: the impact on bioinformatics**., artigo, 2002.

HAYKIN, S., **Redes neurais: Princípios e prática., Bookman**, 21-222 p.Porto Alegre,1999.

KOZAK, M. **Initiation of translation in prokaryotes and eukaryotes**., artigo, 1999.

KRÖSE, B.; VAN DER SMAGT, P. **An introduction to neural networks**. Amsterdam,1996. 8ª edição.

LIPPMANN, R. P., **An introduction to computing with neurals nets**. IEEE ASSP, 1987.

NOBRE, C. N., ORTEGA, J. M., BRAGA, A. P., **High Efficiency on Prediction of Translation Initiation Site (TIS) of RefSeq Sequences**, artigo, 2007.

OLIVEIRA, J. R., **Uma proposta de utilização de redes neurais para reconhecimento de indivíduos através da fala**, monografia de graduação, Universidade Federal de Lavras, Lavras,2007.

PEDERSEN, A. e NIELSEN, H., **Neural network prediction of translation initiation sites in eukaryotes: perspectives for est and genome analysis.**,226–233 p., 1997.

STRINI, E. J., **Previsão da incidência de dengue por meio de redes neurais artificiais**, monografia de graduação, Universidade de São Paulo, Ribeirão Preto,1997.

TEIXEIRA, R.A., **Treinamento de redes neurais artificiais atravpes de otimização multi-objetivo: Uma nova abordagem para o equilíbrio entre a polarização e a variância**., tese de doutorado, Universidade Federal de Minas Gerais, Belo Horizonte, 2001.

WEISS, S. M., KULIKOWSKI, C. A. **Computer Systems That Learn. Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufmann Publishers**, CA, 1991.

ZIEN, A., RATSCH, G., MIKA, S., SCHOLKOPF, B., LEMMEN, C., SMOLA, A., LENGAUER, T., e MULLER, K. R, **Engineering support vector machine kernels that recognize translation initiation sites.**, artigo, 2000.