UM NOVO MÉTODO KERNEL PARA A ANÁLISE DISCRIMINANTE DE SEQUÊNCIAS BIOLÓGICAS

RAUL FONSECA NETO
Departamento de Ciência da Computação – UFJF
raulfonsecaneto@ig.com.br
VICTOR S. DE A. MENESES
Programa de Pós-Graduação em Engenharia de Sistemas e Computação – COPPE/UFRJ
victor_stroele@yahoo.com.br

Resumo - Este trabalho tem como objetivo explicitar a dimensão do problema de avaliação da similaridade entre seqüências biológicas e apresentar uma proposta apropriada para a solução do mesmo desenvolvendo e implementando um algoritmo baseado em um método *kernel* aplicado a solução de problemas de classificação em Bioinformática. O método proposto se baseia na exploração da estrutura sintática das seqüências representadas na sua forma primária como uma cadeia de caracteres. A nova metodologia foi utilizada para o desenvolvimento de um classificador capaz de predizer a classe de pertinência de uma seqüência no sentido de apresentar ou não uma localização de *splice junctions* podendo ser do tipo *intron/exon* ou *exon/intron* . Para tanto, foi utilizado um classificador SVM, também conhecido como máquina de vetores suporte, que se utiliza da matriz *kernel* gerada pelo método aplicado a um problema de multi-classificação.

Palavras-Chave – Máquinas de Vetores Suportes, Métodos *Kernel*, Similaridade de Sequências Biológicas, Tabela de Sufixos, Função *Kernel Fuzzy*

1. Introdução

O surgimento das máquinas de vetores suportes, Cortes e Vapnik [1995], e, posteriormente, o desenvolvimento de classificadores *kernel*, Smola e Scholkopf [2001], acarretaram um grande avanço, e, até mesmo, uma mudança de perspectiva, no campo da aprendizagem de máquinas, no contexto, sobretudo, do aprendizado supervisionado. Dentre as principais aplicações se destaca a área de Bioinformática, também conhecida como Biologia Molecular Computacional. Recentemente, foi publicado uma coletânea de trabalhos específicos abordando o desenvolvimento e a aplicação de métodos *kernel* em problemas relacionados a Biologia Computacional, Scholkopf, Tsuda e Vert [2004] .

Atualmente, um grande esforço vem sendo realizado no sentido de organizar e analisar os dados gerados pelo projeto Genoma. Um dos interesses principais é a identificação da estrutura dos genes através das seqüências obtidas. Para tal, busca-se por sinais particulares associados com a expressão gênica. Para o estudo entre eucariontes e procariontes é interessante a análise de ocorrência ou não de *splice junctions* nas seqüências biológicas. Isto se deve ao fato dos eucariontes apresentar esta característica e os procariontes não.

O problema em questão trata do desenvolvimento de um classificador SVM que seja capaz de distinguir corretamente três tipos de seqüências. As seqüências podem ser divididas em dois grandes grupos: as que representam *splice junctions* e as que não representam. Além disso, as seqüências que representam esta característica podem, também, ser subdivididas: as que apresentam *splice junctions* do tipo *intron/exon* e as que apresentam do tipo *exon/intron.* Desta forma, é gerado um problema de multiclassificação relacionado a existência de três classes distintas.

2. Métodos: SVMs e *Kernels*

2.1 Máquinas de Vetores Suportes

Máquinas de Vetores Suportes (SVMs) são classificadores binários de máxima margem introduzidos, inicialmente, por Boser, Guyon e Vapnik em 1992. Esta técnica busca separar o conjunto de treinamento por um hiperplano que maximize a distância entre elementos pertencentes a classes opostas. Além disso, classificadores construídos por SVMs buscam uma performance de generalização melhor em problemas de classificação quando comparados com classificadores perceptrons.

Para problemas linearmente separáveis, a margem, γ, é definida pela distância mínima de um ponto para o hiperplano. Existem dois tipos diferentes de margens, a margem funcional dada por: $g_f = \min_x \langle w, x \rangle + b$,e a margem geométrica, definida como a margem funcional dividida pela norma L_2 do vetor *w*. Assim, esta margem será $g_g = g_f / \|w\|$.

Entretanto, para se obter o hiperplano de máxima margem, que classifique corretamente todos os dados, $\{(x_i,y_i),\ i=1,...,m\}$, é necessário resolver o seguinte problema de otimização:

$$\begin{aligned} &\max_{w,b} \left(\min_x \ y_i \left(\langle w, x_i \rangle + b \right) / \|w\| \right) \\ &\textit{sujeito a} \\ &y_i \left(\langle w, x_i \rangle + b \right) > 0, \ \textit{para}\ i = \{1, ..., m\} \end{aligned} \qquad (1)$$

Definindo: $g_g||\boldsymbol{w}|| = 1$ como sendo o valor da margem funcional mínima, Vapnik derivou a seguinte formulação alternativa:

$$\begin{aligned} &\min \|w\| \\ &\textit{sujeito a} \\ &y_i\left(\langle w, x_i\rangle + b\right) \geq 1. \end{aligned} \tag{2}$$

Para facilitar a solução deste problema, é conveniente relaxar as restrições de desigualdade introduzindo um conjunto de multiplicadores Lagrangeanos não negativos, ou variáveis duais, a_i, onde $i = \{1,...,m\}$. Assumindo a forma quadrática da função objetiva e incorporando a penalização das restrições, será obtida uma função Lagrangeana descrita pela seguinte equação:

$$L(w,b,a) = \tfrac{1}{2} w^t w - \sum_i a_i\, y_i\left(\langle w, x_i\rangle + b\right) + \sum_i a_i. \tag{3}$$

A função Lagrangeana precisa ser minimizada em relação aos parâmetros w e b, e maximizada em relação ao parâmetro a, sujeito a $a_i \geq 0$, $i = \{1,...,m\}$. Esta solução pode ser obtida através da maximização de uma função dual, onde os parâmetros w e b são substituídos. Esta formulação do problema é chamada forma dual de Wolfe. A mesma pode ser conseguida substituindo-se os parâmetros w e b em função da solução do sistema relativo às derivadas parciais da função Lagrangeana em relação à w e b.

Desta maneira, é construída a formulação final do problema SVM escrito somente como uma função do vetor de variáveis duais a::

$$\begin{aligned} &\max L(a) = \sum_i a_i - \tfrac{1}{2}\sum_i \sum_j a_i a_j y_i y_j \langle x_i, x_j\rangle \\ &\textit{sujeito a} \\ &\sum_i a_i y_i = 0 \\ &a_i \geq 0, i = \{1,...,m\}. \end{aligned} \tag{4}$$

Solucionando este problema e encontrando os valores a^*, será obtido o seguinte vetor normal:

$$w^* = \sum_i a_i^* y_i x_i. \tag{5}$$

Para se obter a função discriminante final é necessário computar o parâmetro bias, b. Este é facilmente calculado utilizando a condição de "complementaridade" de Karush-Kuhn-Tucker:

$$a_i\left(y_i\left(\langle w, x_i\rangle + b\right) - 1\right) = 0. \tag{6}$$

É interessante observar que esta condição é somente verificada para valores positivos de a_i. Estes multiplicadores são associados aos pontos que definem a posição do hiperplano, sendo assim chamados de vetores suportes. Desta forma, se o parâmetro *bias* é corretamente computado, tem-se que: para $a_i > 0$ implica em $y_i(\langle w,x_i\rangle + b) = 1$, tornando a equação de complementaridade satisfeita.

2.2 Método *Kernel*

Para a solução de um problema não linearmente separável é necessária a introdução de funções *kernel*. Dado uma função de mapeamento, Φ, definida na forma: $\Phi : \Re^d \rightarrow F$,sendo F o espaço de características, é possível criar uma representação dos pontos em um espaço de mais alta dimensão no qual o problema de classidicação se torna linearmente separável. Na maioria dos casos, não é preciso conhecer o mapeamento, Φ, explicitamente, desde que se utilize uma função *kernel*, definida como $k : \Re^d \times \Re^d \rightarrow \Re$, que corresponde ao produto interno dos vetores neste espaço de mais alta dimensão. Ou seja, $k(x,y) = \langle \Phi(x), \Phi(y)\rangle$. Como exemplo, tem-se o desenvolvimento de um classificador que utiliza uma base de funções polinomiais. Neste sentido, considerando o grau do polinômio igual a 2, relativo a uma expansão quadrática dos vetores do espaço de entrada, tem-se, para um conjunto X definido em $\Re^2$, a seguinte função *kernel:*

$$\begin{aligned} &\Phi : \Re^2(X) \rightarrow \Re^3(F), \\ &(x_1, x_2) \rightarrow \Phi(x_1, x_2) = \left(x_1^2, \sqrt{2.x_1.x_2}, x_2^2\right) \\ &k(x,y) = \langle \Phi(x), \Phi(y)\rangle = \left(x_1^2, \sqrt{2.x_1.x_2}, x_2^2\right).\left(y_1^2, \sqrt{2.y_1.y_2}, y_2^2\right)^T \\ &\quad = \left((x_1, x_2).(y_1, y_2)^T\right)^2 = \left(\langle x, y\rangle\right)^2 \end{aligned} \tag{7}$$

Considerando a definição de w em função do vetor de multiplicadores a, $w = \sum_i \alpha_i y_i \Phi(x_i)$, a função discriminante pode ser reescrita na forma:

$$f(x_i) = \sum_j a_j y_j y_i\, K\left(x_i, x_j\right) + b. \tag{8}$$

Entretanto, para que uma função real k seja caracterizada como um *kernel:*

$k : \hat{A}^d \times \hat{A}^d \rightarrow \hat{A}$, tal que $\Phi : \hat{A}^d \rightarrow F$ e $k(x,y) = k(y,x) = \langle \Phi(x), \Phi(y)\rangle$, $x,y \in \hat{A}^d$,

é necessário satisfazer algumas condições. A primeira condição a ser estabelecida é a simetria, derivada da condição de simetria do produto interno.

Também, deve-se verificar a condição de existência de um *kernel*. Esta condição foi estabelecida por Mercer [1909], que determinou as condições de necessidade e suficiência para que uma função simétrica $k(x,y)$ seja um *kernel*. Esta condição está associada ao fato de que a função deva ser semi-definida positiva, ou seja, para um conjunto de vetores $x_1, x_2, ..., x_l$, e para um conjunto de escalares $\lambda_1, \lambda_2, .., \lambda_l$, a função k deve satisfazer:

$$\sum_i \sum_j \lambda_i.\lambda_j.k(x_i,x_j) \geq 0. \tag{9}$$

Para melhorar a eficiência de certas aplicações, todos os valores de uma função *kernel*, relativos a cada par de pontos do conjunto de treinamento, podem ser computados uma única vez e armazenados em uma matriz *kernel*.

2.3 SVM e Multiclassificação

Para a utilização de classificadores SVM em problemas de multiclassificação, deve-se considerar o treinamento de um conjunto de hiperplanos separadores formando uma função de decisão mais complexa. Este processo pode ser desenvolvido de duas formas diferentes, que apresentam, entretanto, resultados equivalentes.

A primeira consiste no método de treinamento denominado "um contra todos", ou seja, para cada classe do problema, é treinado um classificador considerando como positivos os exemplos da classe e negativos os demais. Tem-se, portanto, a geração de *n* hiperplanos separadores, sendo *n* o número de classes existentes.

Neste caso, a função de decisão rotula como classe de um novo exemplo àquela associada ao hiperplano que esteja mais afastado do vetor, no sentido de garantir uma maior generalização. Considerando $g(x)$ a função que computa a distância de um ponto ao hiperplano associado e $c(x)$ a função que rotula a classe, tem-se $c(x) = Arg\ Max_{\ i}\ \{ g_i(x)\}$.

Para se testar a pertinência de um novo exemplo são computadas *n* distâncias, realizando-se *n-1* comparações.

A segunda alternativa consiste em se treinar uma classe contra a outra, considerando somente os exemplos de treinamento associados às respectivas classes. Neste caso, serão realizados $C_{n,2}$ ou exatamente $n.(n-1)/2$ treinamentos, com a geração de um número quadrático de hiperplanos separadores na forma $g_{i,j}(x)$ *para* $i \neq j$. A função de decisão envolve a formação de uma árvore binária de decisão contendo n vértices terminais relativos as n classes e $1 + 2 + ... + n-1 = n.(n-1)/2$ vértices internos, correspondentes a confrontação de todos os pares de hiperplanos obtidos. Para testar a pertinência de um novo exemplo é necessária a realização de *n-1* comparações, cada uma envolvendo a avaliação de uma função de decisão do tipo $\varphi(g_{i,j}(x))$, que estabelece um caminho da raiz da árvore até um vértice terminal associado a classe vencedora.

3. *Kernel* baseado em Tabela de Sufixos

A questão principal que se levanta na análise de seqüências biológicas por algoritmos de aprendizado está relacionada à forma de representação e de codificação dos dados de entrada. Por tratar-se de dados estruturados que não possuem, inicialmente, uma representação vetorial definida, como representá-los de forma adequada para que todas as informações associadas aos mesmos sejam extraídas pelo processo de treinamento? Os métodos tradicionais, como as redes neurais artificiais e algoritmos genéticos sugerem uma representação codificada dos dados, geralmente na forma binária, para a construção dos vetores de representação.

Na análise de seqüências biológicas tem-se a representação clássica ou na forma canônica dos nucleotídeos {A,C,G,T} pelos respectivos vetores binários (1,0,0,0), (0,1,0,0), (0,0,1,0) e (0,0,0,1). Esta representação possibilita a simples comparação posicional entre duas seqüências de mesmo tamanho através da utilização do produto interno entre os vetores codificados. O valor do produto interno representará de forma obvia o número de repetições de nucleotídeos nas duas seqüências analisadas. Entretanto, apesar de ser possível expandir esta forma de representação com a aplicação de funções *kernel* polinomiais e gaussianas sobre os vetores, ela representa dois inconvenientes principais que inviabilizam a sua utilização em quase todos os problemas de análise de seqüências biológicas: as seqüências têm que ter o mesmo tamanho e as seqüências tem que estar alinhadas por um processo de alinhamento múltiplo de complexidade $O(n^3)$

Neste sentido, como na maioria das vezes os fragmentos de seqüências biológicas não estão alinhados e muito menos são de mesmo tamanho, pode-se optar por uma forma de representação que ao invés de se preocupar com a representação vetorial ou com a codificação dos dados de entrada, ou mesmo com a sua projeção no espaço de características, se preocupa com a geração de uma matriz *kernel* cujos valores devem traduzir de forma adequada o grau de similaridade dos dados. Esta nova forma de representação dos dados, a qual possibilita o treinamento eficiente de algoritmos de aprendizado sem mesmo conhecer a disposição espacial dos vetores no espaço de entrada, conforme apresentado na figura 1, é a base do desenvolvimento de novos classificadores que utilizam a teoria dos métodos *kernel*, cujos resultados podem ser estendidos, também, para outros tipos de análises.

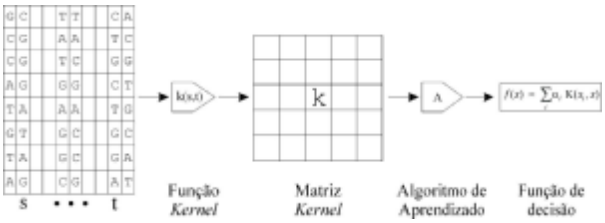

Figura 1: Representação das etapas de construção de um classificador

3.1 Operadores e Vetores de Características

Suponha que se deseja medir a similaridade entre duas seqüências de nucleotídeos, analisando as subseqüências repetidas de tamanhos dois. Para tanto, deve-se construir os vetores de características, $\Phi_k(x)$ para $k = 2$, considerando o alfabeto:

$$\sum^2 = \left\{ \begin{matrix} AA, AC, AG, AT, CA, CC, CG, CT, \\ GA, GC, GG, GT, TA, TC, TG, TT \end{matrix} \right\} \quad (10)$$

Portanto, para *s* = GCATATTCGACTCCAG e *t* = TGATATCCGACTCGAG, tem-se os seguintes vetores de características binários:

$$\begin{aligned} \Phi_2(s) &= [0,1,1,1|1,1,1,1|1,1,0,0|1,1,0,1] \\ \Phi_2(t) &= [0,1,1,1|0,1,1,1|1,0,0,0|1,1,1,0]. \end{aligned} \quad (11)$$

Após a construção dos vetores de características é possível avaliar a função *kernel* $k_2(s,t)$ através do operador binário. Este valor representa o número exato de ocorrências de subseqüências iguais de tamanho dois entre as duas seqüências. No entanto, não é considerado a ocorrência de repetições, fazendo com que as subseqüências repetidas sejam consideradas uma única vez somente.

Neste caso, uma nova alternativa consiste, ao invés de se computar vetores binários, construir vetores de números inteiros, onde cada posição indicará o número de ocorrências daquela subseqüência na seqüência que está sendo analisada. Considerando as mesmas seqüências *s* e *t* definidas anteriormente, os novos vetores de características serão:

$$\begin{aligned}\Phi_2(s) &= [0,1,1,2\,|\,2,1,1,1\,|\,1,1,0,0\,|\,1,2,0,1]\\ \Phi_2(t) &= [0,1,1,2\,|\,0,1,2,1\,|\,3,0,0,0\,|\,1,2,1,0].\end{aligned} \quad (12)$$

A utilização de valores absolutos pode trazer algumas incorreções na análise de similaridade de seqüências de tamanhos diferentes. Neste sentido, pode-se optar pelo uso de um vetor de probabilidades onde a ocorrência de cada subseqüência fica representada pela respectiva freqüência relativa em relação ao número total de ocorrências. Assim, para o exemplo descrito será obtido, para *s*, o seguinte vetor de probabilidades:

$$\Phi_2(s) = \begin{bmatrix} 0,1/15,1/15,2/15\,|\,2/15,1/15,1/15,1/15\,| \\ 1/15,1/15,0,0\,|\,1/15,2/15,0,1/15 \end{bmatrix}. \quad (13)$$

Com esses novos vetores é possível utilizar tanto o operador produto, quanto o operador *min*, para o cálculo da função *kernel*. No primeiro caso, aplicando-se o operador produto, pode-se afirmar que os valores produzidos pela função *kernel* estão associados a um somatório de produto de probabilidades individuais. Caso os eventos sejam independentes este valor traduz o somatório das probabilidades conjuntas de ocorrência de todas as subseqüências no par de seqüências que está sendo analisado. Este tipo de cálculo pode gerar valores de similaridade elevados e não normalizados, não representando, muitas vezes, de forma adequada, o grau de similaridade entre as seqüências.

Alternativamente, pode ser utilizado o operador *min* equivalente à operação de interseção entre os vetores para o cálculo da função *kernel*. Este operador parece ser mais adequado que os outros, visto que, ao considerar, somente, a repetição conjunta de subseqüências, computa de forma mais realista a medida de similaridade entre as seqüências. A utilização do operador *min* sobre os vetores de probabilidades não possui um significado estatístico claro. Entretanto, caso este operador seja aplicado sobre vetores de possibilidades, cujos valores estão relacionados a valores de funções de pertinência, pode ser construído um significado nebuloso dentro da teoria de conjuntos *fuzzy*. Neste caso, computa-se a interseção de valores *fuzzy*, representados pelas respectivas funções de pertinência, como o mínimo dos valores existentes.

Considerando que os valores dos vetores de probabilidades e possibilidades pertencem ao intervalo fechado [0,1], foram representados na figura 2 o conjunto imagem relativo à aplicação dos operadores em uma função de dois parâmetros com domínio [0..1] x [0..1].

Entretanto, o uso explícito de vetores de características não é muito recomendável, pois o seu tamanho cresce exponencialmente em relação ao tamanho do alfabeto. Por exemplo, para k = 5 e considerando o alfabeto de nucleotídeos, o vetor de características teria 4^5 componentes, tornando o seu cálculo extremamente complexo. Neste sentido, buscando solucionar este problema, optou-se pelo desenvolvimento de um método *kernel* que não necessita representar de forma explícita os vetores característicos.

a) b)

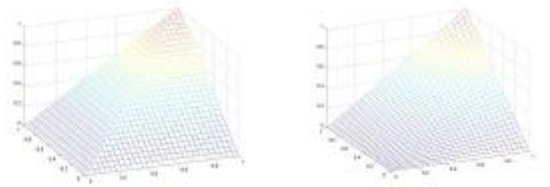

Figura 2: Representação gráfica dos operadores *min* (a) *produto* (b)

3.2 Tabela de Sufixos

Para evitar problemas com a explosão do espaço de características, surgiu a idéia de formar um vetor de tamanho fixo *n*, também denominado $\Phi_k(x)$, composto por todas as subseqüências de tamanho *k*.

O processo de geração destes vetores é realizado em tempo *O(n)*, já que cada seqüência é avaliada por um processo de janelas deslizantes, passando por toda a seqüência uma única vez. Utilizando a seqüência *s*, descrita anteriormente, o novo vetor característico de subsequências de tamanho fixo 2 seria:

$$\Phi_2(s) = \begin{bmatrix} GC, CA, AT, TA, AT, TT, TC, CG, \\ GA, AC, CT, CC, CA, AG, Ge \end{bmatrix}. \quad (14)$$

Entretanto, observa-se que, com o uso de uma tabela, é possível obter uma única representação para tamanhos variáveis de subseqüências.

$T_8(s)$	A	T	G	C	A	T	A	G
	T	G	C	A	T	A	G	A
	G	C	A	T	A	G	A	C
	C	A	T	A	G	A	C	ε
	A	T	A	G	A	C	ε	
	T	A	G	A	C	ε		
	A	G	A	C	ε			
	G	A	C	ε				
	A	C	ε					
	C	ε						

Figura 3: Tabela de sufixos da seqüência *s* para subseqüências de tamanho $k = 8$

Considere as sequências *s*= ATGCATAGAC e *t* = TTCGATAACG, e a utilização de subseqüências de tamanho $k=8$. Os vetores correspondentes poderão ser armazenados na tabela de sufixos $T_k(x)$ que será formada por todas as ocorrências das subseqüências de tamanho menor ou igual a *k* na seqüência *x*. Na figura 3, mostra-se uma tabela que representa as subsequências de tamanho menor ou igual a 8 da sequência s = ATGCATAGAC. A existência do símbolo terminal e indica que a subseqüência não pode ser formada com comprimento maior ou igual a *k*.

Utilizando este tipo de representação para os vetores de características haveria um elevado custo para o armazenamento das subseqüências de ordem $O(n^2)$. Entretanto, cada uma destas tabelas de sufixos pode ser representada de forma linear por um vetor de inteiros de forma que as subseqüências passam a serem acessadas de forma indireta pelos respectivos índices. É importante ressaltar que esta forma de representação é mais eficiente que a representação baseada em uma árvore de sufixos (Ukkonen [1995]). Após a construção das tabelas a próxima etapa na avaliação da função *kernel* consiste de um processo de *match* relacionado à aplicação do operador *min*. Este processo busca comparar cada posição entre os vetores de forma a contar quantas subseqüências se repetem. Para que este processo tenha complexidade linear é necessário executar previamente uma ordenação das tabelas. A ordenação dos vetores pode ser feita em tempo *O(k.n)* considerando um processo de partição recursivo em função do alfabeto de símbolos finito, $\sum$. Vale ressaltar que a complexidade do processo de ordenação é independente do tamanho ou da cardinalidade do alfabeto, dependendo, somente, do tamanho das seqüências (n) e das subseqüências (k). No exemplo citado tem-se a seguinte representação vetorial para a sequência *s* relativa a tabela $T_8(s)$ na sua forma ordenada: $\Phi_8(s) = [9,7,5,1,4,10,8,3,6,2]$.

Ao final do processo de *match* é retornado o número de subseqüências idênticas, sendo este número utilizado para o cálculo da função *kernel*. Este algoritmo possui como entrada um conjunto de seqüências biológicas das quais deseja-se obter uma medida de similaridade. O algoritmo computa a similaridade aplicando a função *kernel* em todos os pares de seqüências, formando uma matriz de similaridades, denominada *matriz kernel*. Como a comparação entre as seqüências é independente da ordem entre as mesmas, esta matriz torna-se simétrica. Assim, pode-se armazenar esta matriz como uma matriz triangular de diagonal unitária. Um importante problema relacionado ao desenvolvimento de métodos *kernel* baseados em medida de similaridade está na garantia da propriedade de semi-definidade positiva da função *kernel*. Entretanto, caso este problema ocorra, pode-se, segundo Shawe-Taylor e Cristianini [2004], aumentar os valores dos elementos da diagonal principal como forma de se eliminar os autovalores negativos. Este procedimento corresponde à adoção de um classificador SVM de margem flexível no sentido de minimizar a norma L_2 do vetor relacionado ao grau de violação das restrições de classificação.

3.3 Função *Kernel Fuzzy*

A avaliação de similaridade entre duas seqüências, tendo como base a ocorrência de subseqüências comuns, pode ser feita introduzindo uma função *kernel* normalizada que represente uma função de pertinência. Esta função determina uma forma de avaliação *fuzzy* mapeando valores no intervalo fechado [0..1]. Neste sentido, se as duas seqüências forem exatamente iguais, a função *kernel* assumirá o valor unitário. Do contrário, se as seqüências forem totalmente distintas, a função retornará o valor nulo.

A função de similaridade entre duas seqüências *s* e *t* pode ser obtida mesmo que estas seqüências não possuam o mesmo tamanho. Esta função de similaridade é definida baseando-se na ocorrência de subseqüências comuns de comprimento variando de 1 até *k*. Considerando *n* o tamanho da seqüência de menor comprimento, o cálculo da função *kernel* será dado por: $k(s,t)=\sum_{i=1}^{k} k_i(s,t).\, l^i$, onde l^i representa a função peso associada ao tamanho de cada subseqüência *i*.

Como se deseja obter valores de similaridades no intervalo [0..1], é preciso dividir a função *kernel* pelo valor máximo que ela possa atingir. O valor da similaridade máxima será atingido quando duas seqüências idênticas forem comparadas. Considerando, neste caso, $k_1(s,t) = n$, ..., $k_k(s,t) = n - (k-1)$, pode-se redefinir o valor da função *kernel* para:

$$k(s,t)=n.\frac{k_1(s,t)}{n}+n.\frac{k_2(s,t)}{n-1}+\ldots+n.\frac{k_k(s,t)}{n-(k-1)}, \qquad (15)$$

obtendo-se uma similaridade máxima entre as seqüências igual a $n.k$. Esta função *kernel* pode ser rescrita de forma simplificada como:

$$k(s,t)=\sum_{i=1}^{k} k_i(s,t).\, l^i \text{ e } l^i=\frac{n}{n-(i-1)},\ para\ i=1,\ldots,k. \qquad (16)$$

Como esta nova função possui valor máximo de similaridade igual a $n.k$, basta dividi-la por este mesmo valor para que o enquadramento destes valores esteja no intervalo [0..1]. Logo, a função *kernel fuzzy* para a avaliação de subseqüências variando de 1 até *k* pode ser escrita, finalmente, como:

$$m_{1,k}(s,t)=\frac{k(s,t)}{k.n}=\sum_{i=1}^{k} k_i(s,t).\, l^i \text{, e}$$
$$l^i=\frac{1}{(n-(i-1)).k}, \quad para\ i=1,\ldots,k. \qquad (17)$$

Assim como *k* é considerado um limite superior no tamanho das subseqüências, é possível, também, estabelecer um valor mínimo ou limite inferior no tamanho das subseqüências. Com este limite inferior, a função *kernel* torna-se mais geral, permitindo um

estudo mais preciso do grau de similaridade entre as seqüências. Representando este limite inferior por j, a função *kernel* será dada por:

$$m_{j,k}(s,t) = \sum_{i=j}^{k} k_i(s,t).\, l^i \text{, e}$$
$$l^i = \frac{1}{(n-(i-1)).(k-(j-1))}, \quad para\ i = j, ..., k. \tag{18}$$

Evidentemente, a definição do intervalo apropriado, ou seja, a definição dos limites inferior e superior, dependerão da natureza do problema envolvido.

4. Materiais: Conjunto de Dados e Matriz *Kernel*

4.1 Conjunto de Dados

O conjunto de dados consiste de seqüências de DNA com e sem a presença de *splice junctions*. São seqüências de primatas extraídas do *UCI benchmark*. A base de dados contém 3190 seqüências sendo distribuídas em três grupos. O primeiro grupo possui 767 (24%) seqüências do tipo *exon/intron* (EI) ou "*acceptors*", o segundo 768 (24%) seqüências do tipo *intron/exon* (IE) ou "*donors*" e o terceiro possui 1655 (52%) seqüências que não possuem nenhuma destas características. Todas as seqüências desta base de dados possuem tamanhos iguais. Cada uma das seqüências é formada por cerca de sessenta nucleotídeos ou pares de bases.

4.2 Matriz *Kernel*

Para o treinamento do classificador SVM foram geradas quinze matrizes relacionadas, às três classes do problema e à variação de comprimento das subseqüências. Foram utilizados valores de intervalos correspondentes aos limites inferior e superior apresentados na tabela 1. Como exemplo, para *j = 1* e *k = 2*, representa-se a ocorrência simultânea de subseqüências de tamanho um e de tamanho dois.

5. Resultados e Discussão

Para a aplicação da metodologia foram feitas algumas variações nos tamanhos dos intervalos, para efeito de avaliação da similaridade, de forma a analisar os resultados do classificador. Tal estratégia se deve ao fato de que, a partir da realização de um alinhamento múltiplo para cada classe, foi constatado que o grupo de seqüências pertencente à primeira classe apresentava subseqüências conservadas de tamanho três, ou seja, esta classe apresentava três nucleotídeos quase idênticos em todas as seqüências em uma mesma região. Já a segunda classe, apresentava subseqüências conservadas de dois nucleotídeos. Finalmente, as seqüências da terceira classe não apresentavam conservação de subseqüências, ou seja, os nucleotídeos eram dispostos de forma aleatória. Neste sentido, para o cômputo da similaridade, foi escolhida uma região de 12 nucleotídeos dentro das seqüências contendo as sub-regiões conservadas. A fase de treinamento foi feita sempre entre duas classes. Assim, primeiramente, foi feito o treinamento entre as classes um e dois de forma a determinar uma função discriminante capaz de classificar corretamente os dados referentes às duas classes. Posteriormente, foram feitos os treinamentos entre as classes um e três e, depois, entre as classes dois e três.

Como resultados da fase de treinamento é obtido um valor real que indica o tamanho da margem do classificador SVM. Quanto maior o tamanho desta margem melhor será a classificação dos dados. A tabela abaixo ilustra o tamanho das margens obtidas a partir das variações do tamanho das subseqüências.

Limite Inferior	*Limite Superior*	*C1xC2*	*C1xC3*	*C2xC3*
1	1	0,055	0,077	0,0303
1	2	0,090	0,055	***0,0602***
1	3	0,097	0,074	0,0585
2	3	0,102	***0,108***	0,0464
3	3	***0,121***	0,102	0,0359

Tabela 1: Treinamento entre as classes

Como pode ser observado na tabela 1, o treinamento entre a primeira e a segunda classe apresentou um resultado superior quando a matriz *kernel* foi gerada sobre subseqüências de tamanhos três. Este resultado já era esperado em função da primeira classe ser representada por subseqüências conservadas de tamanhos três. Por outro lado, o treinamento entre a segunda e terceira classe apresentou um melhor resultado quando a matriz *kernel* foi gerada sobre subseqüências de tamanho um e dois. Este fato é explicado em função da classe dois apresentar subseqüências conservadas de dois nucleotídeos. Tais resultados comprovam, portanto, que a correta escolha do intervalo de avaliação é de fundamental importância para a geração de matrizes *kernel* que possibilitem uma melhor separação das classes com uma maior margem de separação resultando em classificadores com maior poder de generalização.

Referências Bibliográficas

Boser, B.E., Guyon, L.M. e Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proc. of the 5th Annual ACM workshop on COLT. ACM Press, 144-152, **1992**

Cortes, C. e V. Vapnik. Support Vector Networks. Machine Learning, 20: 273-297, **1995**

Mercer, J. Functions of positive and negative type and their connection with theory of integral equations. Philosophical Transactions of the Royal Society, London A 209, 415-446, **1909**.

Shawe-Taylor, John e Cristianini, Nello. Kernel Methods for Pattern Analysis. Cambridge University Press, **2004**

Scholkopf, Bernhard, Tsuda, Koji e Vert, Jean-Philippe edits. Kernel Methods in Computational Biology. MIT Press, 2004

Smola, A. e B. Scholkopf. Learning with Kernels. MIT Press, **2001**

Ukkonen, **E.** On-line construction of suffix trees. *Algorithmica*, **14**, 249-260, **1995**