Introdução à Bioinformática

Trabalho Prático 01
Alinhamento Sequencial de Proteínas

Aluno: Gustavo Campos Ferreira Guimarães – 2009052247

Introdução

O presente trabalho foi desenvolvido durante a disciplina de Introdução à Bioinformática, ministrada pela professora Sabrina Silveira. O objetivo principal, como interpretado pelo aluno, seria de analisar duas sequências proteicas providas pela professora e, utilizando-se das técnicas de alinhamento sequencial aprendidas em sala de aula, julgar quais as mutações mais danosas à função original da proteína ocorridas entre as duas versões.

Dada à natureza do problema fortemente orientada ao casamento de padrões de caracteres e manipulações de listas e *strings*, a linguagem escolhida para implementação foi Python, em sua versão 2.7.5, na qual o aluno já se considerava previamente proficiente.

Decisões de Implementação

Frente à inexperiência no campo da bioinformática e restrições de tempo, diversas abstrações foram assumidas de forma a simplificar o problema do alinhamento proteico. Além destas, outras abstrações sabidamente limitadoras foram adotadas ao se realizar o julgamento de mérito das mutações ocorridas. Cada uma dessas decisões será comentada abaixo:

1. Algoritmo Utilizado

O trabalho implementa o algoritmo de alinhamento conhecido como Needleman-Wunsch, que realiza o alinhamento global das sequências. Esse algoritmo se mostrou mais simples de ser implementado do que outros pesquisados, encaixando-se bem ao caráter pedagógico e ao prazo proposto.

Por não se preocupar com alinhamentos locais, é sabido que esse algoritmo não necessariamente produzirá os melhores alinhamentos, mas julga-se que os resultados são aceitáveis para a proposta do trabalho.

2. Política de Gapping

O algoritmo de Needleman-Wunsch utiliza penalidade linear para situações de gapping. A implementação dessa penalidade mostra-se bastante simples, já que utiliza-se do caráter de programação dinâmica do algoritmo: basta inicializar as primeiras linha/coluna da matriz de distâncias com as penalidades desejadas e o algoritmo trata de propagar essas penalidades pela matriz.

Essa política linear é sabidamente limitadora, uma vez que assume que todos os gaps produzidos são igualmente danosos à função da proteína, o que sabe-se em geral não ser universalmente verdade. Mais uma vez, assume-se essa limitação dado o caráter pedagógico do trabalho desenvolvido.

3. Escolha do Gap Inicial

A fim de melhorar marginalmente a qualidade da penalização por gapping empregada, foi pensada uma regra de decisão automática e bastante simplória para a escolha da penalidade de Gap: o alinhamento inicial entre a proteína selvagem e a mutante é realizado utilizando-se uma série de penalidades de gap diferentes, e a cada caso é contada a quantidade de mutações ocorridas na sequência final. Assume-se, arbitrariamente, que o melhor alinhamento é aquele com a menor quantidade de mutações, e o gap utilizado para obter este resultado será empregado no restante da análise.

Após diversos testes, notou-se que a penalidade de gap somente alterava os resultados do alinhamento num raio muito pequeno, de modo que na implementação final a regra de negócios está aplicada para o raio de penalidade de gap que vai de -5 a 5, apenas.

4. Matriz de Similaridade

A matriz de similaridade (erroneamente chamada no código de matriz de substituição) define qual o custo aproximado de edição para cada uma das mutações. A matriz escolhida para essa implementação foi a PAM (Point Accepted Matrix).

Ao pesquisar na Internet por versões precalculadas dessa matriz, a melhor ferramenta encontrada[1] permite a criação de PAM's com valor de 1 a 512. Na impossibilidade de escolher um valor adequado à implementação, frente a inexperiência no assunto, foi requisitada uma PAM com o valor padrão, de 250. Essa matriz foi salva no diretório fasta_dados com o nome de pam250.txt e é ativamente utilizada no software.

5. Avaliação de Mutações

De posse do alinhamento realizado entre a proteína selvagem e sua versão mutante, torna-se necessário julgar as mutações ocorridas quanto à sua influência na função original da proteína. Diversas estratégias foram consideradas para este fim, em especial a avaliação de propriedades similares dos resíduos, como hidrofilia, carga elétrica e tamanho. Por restrições de tempo, porém, foi acatada apenas uma das sugestões da professora: a comparação com outras proteínas da mesma família da selvagem.

De posse dos dados providos de sequência de toda a família, decidiu-se por avaliar as mutações que ocorrem naturalmente nessa família e considerar como mais danosas as mutações ocorridas no par proposto que são mais raras na família.

[1] http://www.bioinformatics.nl/tools/pam.html

Esse processo mostrou-se insatisfatório pois, dada a implementação proposta neste trabalho, é impossível garantir que a mutação ocorrida entre o par original é a mesma que está sendo avaliada nos alinhamentos com a família.

6. Alinhamento de Família

Uma das razões para a não confiabilidade do método adotado é o alinhamento inadequado das proteínas da família apresentada com a original 2YPI. Entende-se que o melhor método de alinhamento para este caso seria executar um alinhamento múltiplo, cujo algoritmo é consideravelmente mais complexo que o Needleman-Wunsch utilizado, mas por restrições de tempo evitou-se implementar a melhor solução.

A fim de tentar contornar em parte esse problema, uma estratégia de *scoring* das mutações foi pensada. Para criar um ranking das mutações ocorrentes na família, cada uma das proteínas de família é alinhada com a proteína original 2YPI utilizando a mesma penalidade de gap que foi obtida no alinhamento original. Para cada mutação encontrada em cada alinhamento, um ponto é computado numa matriz unidimensional que mapeia mutações nas posições da proteína original.

A cada passo, essa matriz é atualizada com incrementos em todas as mutações encontradas no alinhamento da 2YPI com cada proteína da família.

Ao fim do processo, obtém-se uma matriz de score que contém em cada posição, um inteiro correspondente a quantas vezes o resíduo situado originalmente nessa posição foi trocado em toda a família.

7. Análise de Mutações

De posse da matriz de scores obtida acima, obtém-se a média aritmética das ocorrências de mutações na família provida. Outras medidas estatísticas, como média harmônica, foram testadas, mas os resultados não se mostraram particularmente melhores em nenhum dos demais casos.

A partir da média aritmética das ocorrências, percorre-se o alinhamento original entre 2YPI e dTIM. A cada mutação encontrada, verifica-se a média de vezes em que essa mutação ocorre na família, caso ela ocorra menos vezes do que a média, é considerada uma mutação potencialmente comprometedora, sendo assim adicionada na lista de mutações escolhidas

Detalhes de Implementação

O software consiste de apenas 5 módulos Python, comentados abaixo

guprotaligner.py

O módulo principal é, essencialmente, o que se espera de um módulo "main" em um software qualquer. Este módulo apenas faz a interpretação dos argumentos de chamada e executa os passos da estratégia definida sequencialmente. Em especial:

- Lê do arquivo a proteína selvagem
- Lê do arquivo a proteína mutante
- Lê do arquivo a matriz de similaridade
- Executa alinhamentos sequenciais entre as proteínas e obtém o melhor gap
- Lê do arquivo os dados das proteínas de família
- Cria a matriz de score das mutações na família
- Obtém a lista de mutações potencialmente danosas
- Escreve os dados obtidos em arquivo

protein.py

Este módulo contém uma única classe Protein que é abstração adotada para uma proteína em forma de estrutura de dados. Trata-se de uma classe bastante elementar que somente lê de um arquivo pré-formatado os dados referentes à proteína e armazena-os em variáveis internas.

submat.py

Mais uma classe elementar, cujo único objetivo é representar digitalmente a matriz de similaridades. Uma interpretação incorreta no início do desenvolvimento do trabalho resultou no nome dessa classe e das variáveis relacionadas no código remeterem a Matriz de Substituição, mas sua função foi corretamente implementada como de uma Matriz de Similaridade.

Como essa classe é capaz de ler de um arquivo PAM os dados da matriz, é bastante trivial substituir a PAM250 adotada por outra PAM com o índice que se deseje.

needwun.py

Esse módulo contém a class NeedWunAlignment que é responsável por realizar o alinhamento utilizando o algoritmo de Needleman-Wunsch bem como compartilhar os dados desse alinhamento entre os diversos contextos em que são solicitados através do software.

Os dois estágios do algoritmo de Needleman-Wunsch são implementados através dos métodos "_compute_matrix" e "_traceback", invocados pelo método "_align".

Essa classe possui duas formas de uso. Quando invocado o método "_default_alignment", assume-se que uma penalidade de gap arbitrária foi fornecida, e executa-se um alinhamento único usando essa penalidade.

Se, por outro lado, é invocado "_find_best_alignment", a classe irá executar uma série de alinhamentos utilizando penalidades diferentes e escolherá aquele com o menor número de mutações computadas como o alinhamento canônico, e o gap utilizado nesse alinhamento como definitivo.

family.py

O último módulo implementa a classe Family, responsável por executar as funções que abrangem toda a família de proteínas. Além de ler de um arquivo a lista com todas as proteínas de uma família, essa classe possui dois métodos importantes.

O primeiro, "build_score" é responsável por fazer a série de alinhamentos em par entre a proteína selvagem e cada uma das proteínas de família, incrementando a matriz de score a cada iteração. O segundo, "score_alignment", compara a sequência provida (no caso a sequência final da proteína mutante após o alinhamento) com a sequência base (alinhamento final da proteína selvagem) e identifica as mutações potencialmente danosas, de acordo com a estratégia previamente delineada.

Resultados Obtidos

O software retorna, finalmente, uma lista de mutações consideradas potencialmente danosas. Para a implementação que segue anexa a esta documentação, a saída obtida foi a seguinte:

Residuo Antigo	Novo Residuo	Posicao	Score
N	K	26	16
N	I	78	14
S	H	100	14
A	T	176	22
N	K	214	28
V	L	242	23

É interessante notar que essa tabela é consideravelmente menor do que a que foi obtida nos estágios de desenvolvimento do trabalho, quando foi utilizado apenas um subconjunto com cerca de 20 proteínas da família. Este pode ser um indício de que as proteínas dessa família são particularmente tolerantes a mutações de sua sequência, já que o conjunto total possui tantas mutações a ponto de muito poucas serem identificadas como potencialmente danosas.

A proposta original do trabalho incluía uma segunda pergunta, além de "Quais as mutações capazes de impactar na função da proteína?". Essa implementação, porém, não é adequada para se responder a pergunta "O que pode ser feito para restaurar essa funcionalidade?", uma vez que as propriedades dos resíduos trocados não foram avaliadas no score das mutações.