Instituto de Matemática e Estatística
Universidade de São Paulo

Projeto PageRank

Relatório

Gabriel Milani - NUSP: 9913641
Guilherme Ventura - NUSP: 11340293
Milton Leal - NUSP: 8973974
Richard Sousa - NUSP: 11810898

MAP 2110 - Modelagem Matemática
Docente: Saulo Rabello Maciel de Barros

Sumário

1) Introdução

O presente relatório discorre sobre o processo de desenvolvimento de algoritmos de classificação. Ele é parte de um projeto da disciplina de Modelagem Matemática do 1º semestre do bacharelado de Matemática Aplicada Computacional do Instituto de Matemática e Estatística da Universidade de São Paulo (IME/USP).

Como problema, foi pedido que criássemos um programa para ranquear páginas. O critério de classificação seria o grau de importância de cada página em relação a uma possível busca de palavra-chave realizada em uma rede fictícia que simulasse a internet.

O programa foi desenvolvido em linguagem de programação de alto nível Python, ao longo de 30 dias, e estima-se que tenha consumido em torno de 120 horas de trabalho, incluindo desde a leitura dos papers científicos que tratam sobre o assunto até os testes finais.

O cerne do projeto compreende a realização de duas tarefas distintas (Tarefa 1 e Tarefa 2) utilizando dois métodos diferentes para encontrar a solução:

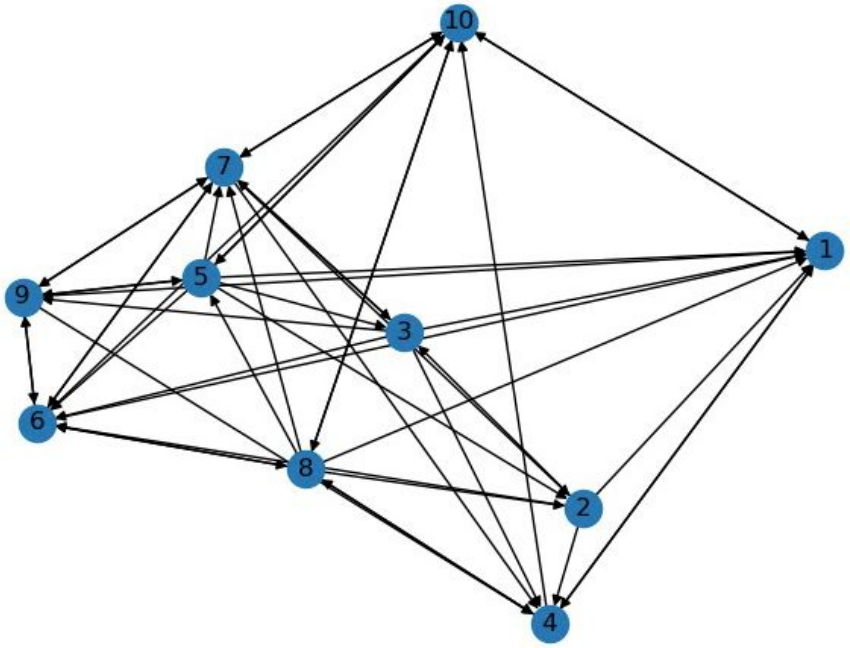
- escalonamento de matrizes utilizando eliminação de Gauss; e

- método iterativo de multiplicação de matriz por vetores, denominado Power Method, criado pelos fundadores do Google, Larry Page e Sergey Brin.

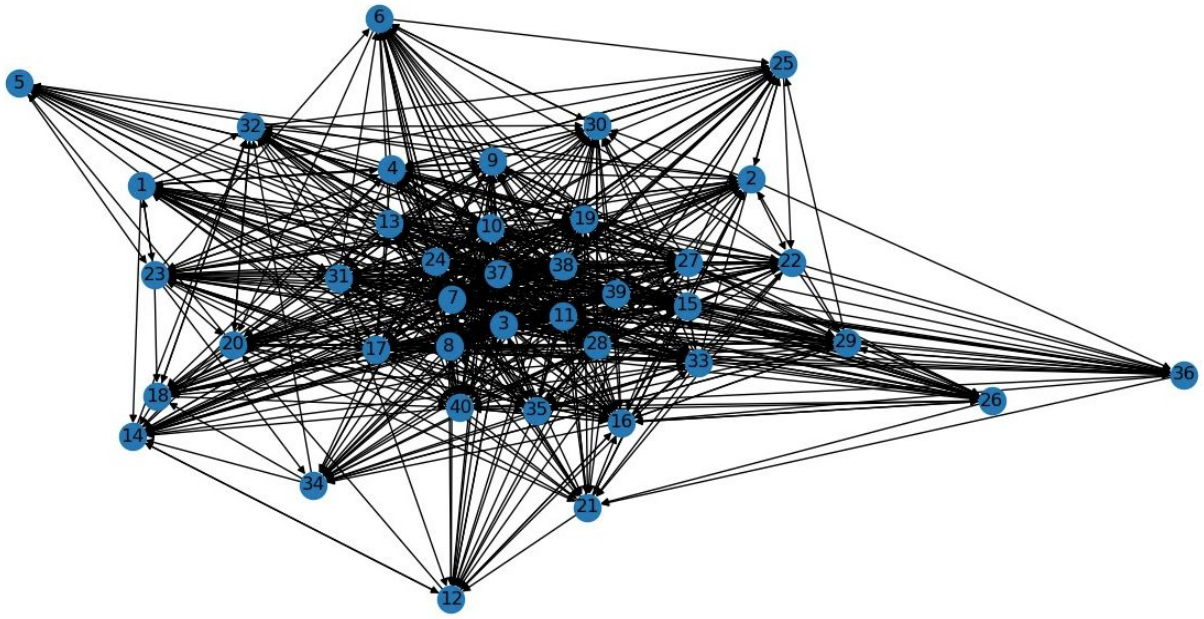
Tarefa 1

Realiza o ranking de importância de páginas de uma rede qualquer que representa um grafo orientado, cujos vértices são páginas da internet e cujas arestas são ligações orientadas entre elas.

Exemplo de rede com 10 páginas:



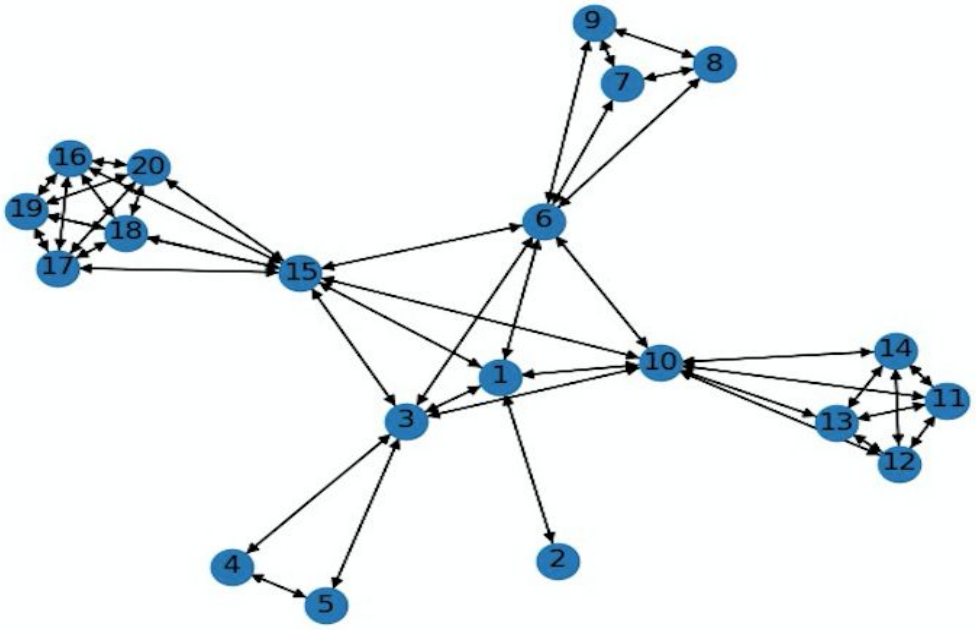
Exemplo de rede com 40 páginas:



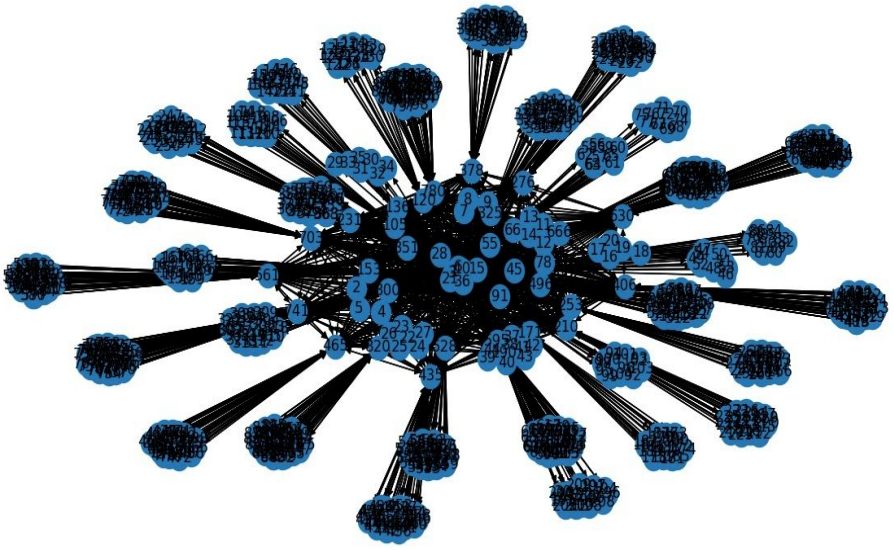
Tarefa 2

Realiza o ranking de importância das páginas de uma rede do tipo cacique-tribo, cujos caciques representam páginas da internet que possuem um subconjunto de páginas (índios) conectado a ela e também ligados entre si, mas que não estão conectados a outros grupos de páginas representados por outros caciques.

Exemplo de rede do tipo cacique-tribo com 5 caciques e respectivos índios:



Exemplo de rede do tipo cacique-tribo com 40 caciques e respectivos índios:



2) Estrutura do programa

2.1) Entrada dos dados

O usuário possui duas opções para entrar com os dados:

I. Gerar uma rede do tipo cacique-tribo a partir da escolha do número total de caciques; ou

II. Inserir arquivo .txt que tenha como conteúdo uma matriz de "zeros" e "uns", na qual "zeros" representam ausência de conexão entre as páginas e "uns" representam existência de conexão entre as páginas.

2.2) Representação matricial dos grafos

Caso o usuário opte pela opção de entrada de dados I, uma classe de funções realiza a construção do grafo que representa determinada rede.

Em seguida, as conexões entre as páginas são transplantadas para uma matriz primitiva que contém apenas "zeros" e "uns". Este tipo de matriz é a mesma que o usuário deverá inserir caso opte pela opção II da entrada de dados.

2.3) Atribuição de pesos

Depois disso, com base na fórmula de atribuição de pesos de importância para as páginas, explicitada no PDF do projeto, modifica-se a matriz primitiva, que passa a ser uma matriz cujas entradas não nulas representam os pesos das páginas. Denominamos esta nova matriz como "matriz de ligação".

2.4) Modificação da matriz de ligação

Em seguida, modifica-se a matriz de ligação utilizando o parâmetro alpha (padrão: 0.15) com o objetivo de substituir as entradas nulas da matriz e transformá-la em uma matriz irredutível.

Neste momento, a depender do método de resolução para obtenção do vetor solução, realizam-se os seguintes procedimentos:

2.5) Método de escalonamento

O próximo passo envolve a subtração da matriz irredutível pela matriz identidade. (isso apenas no caso do escalonamento).

Executa-se o algoritmo de eliminação de Gauss com pivotamento.

Por fim, executa-se o algoritmo que encontra a solução e normaliza-se o vetor solução.

2.6) Método iterativo

A partir da matriz irredutível (modificada com o parâmetro alpha), determina-se a constante C que será utilizada para o cálculo do erro que controla o número de iterações do programa.

Criam-se os vetores V, L, C que representam as entradas não nulas e suas devidas posições na matriz primitiva do item 2.2.

Executa-se o algoritmo que encontra o vetor solução utilizando o erro como fator que controla o número de iterações realizado pelo programa. Por fim, normaliza-se a solução.

2.7) Saída do programa

Como opções de saída, oferece-se ao usuário a possibilidade de escolher a quantidade de páginas no ranking que se deseja visualizar ou a geração de um arquivo com o ranking das páginas.

Imprimimos também a diferença numérica entre os pesos das páginas da rede em relação aos dois métodos de resolução. Além disso, mostramos o tempo de execução de cada método.

3) Testando diferentes alphas

Após realizarmos testes com diferentes valores para o parâmetro alpha que modifica a matriz de ligação, observamos que esse parâmetro atua como uma espécie de controle da taxa de convergência do vetor solução.

Quanto maior o alpha escolhido, mais rápida é a convergência. Além disso, quanto maior o alpha, maior é a influência das conexões entre as páginas na determinação dos pesos e do ranking entre elas.

Por outro lado, quanto menor o alpha escolhido, mais devagar é a convergência e o vetor solução torna-se mais sensível à conexão existente entre as páginas.

Conclui-se, portanto, que valores ligeiramente diferentes para o alpha podem gerar rankings bastante diferentes. Por isso, a determinação do valor de alpha é um exercício de equilíbrio.

4) Desafios e soluções

4.1) Complexidade do projeto

Para nós, ingressantes no curso do BMAC, este projeto foi bastante desafiador. Principalmente para quem começou a ter contato com programação este ano, o nível de dificuldade desse projeto foi bem maior que os exercícios-programas que realizamos na disciplina de Introdução à Computação. Para contornar isso, pesquisamos muito em fóruns de programação e aprendemos muitas coisas novas no Python.

Além do desafio de programação, os conceitos matemáticos em si associados à álgebra linear eram novos para nós. Como solução, dedicamos bastante tempo para leitura da bibliografia indicada no paper em inglês do projeto, além de consultar livros de álgebra linear e sites na internet.

4.2) Entrada de dados

Para a entrada de dados, inicialmente, estudamos uma biblioteca do Python chamada NetworkX, que gera grafos direcionados.

Na primeira implementação do programa, nós utilizamos essa biblioteca para gerar a conexão entre as páginas dado um número inserido pelo usuário para o número de caciques.

Depois, nós substituímos a biblioteca pela construção manual das conexões entre as páginas. Pudemos comparar a nossa implementação manual com a implementação da biblioteca para garantir que as redes estavam sendo geradas corretamente.

Por fim, incluímos a opção do usuário inserir um arquivo .txt com uma matriz primitiva. Realizamos o teste com a matriz de 8 páginas dada no PDF do projeto e vários outros testes com matrizes de diferentes tamanhos.

4.3) Pivotamento

Inicialmente, havíamos entendido que o pivotamento ocorreria durante o processo de escalonamento. Porém, percebemos, durante os testes, que a matriz resultante do escalonamento apresentava erros na diagonal principal.

Como solução, realizamos o pivoteamento completo da matriz para posteriormente realizar o escalonamento.

4.4) Problema da constante C e da matriz modificada

Devido a problemas inerentes à linguagem Python, tivemos dificuldade em operar as matrizes, pois o Python tende a modificar o conteúdo das listas (matrizes) quando estas eram copiadas para fazer outras operações. Possivelmente, o Python não estava fazendo uma cópia, mas sim fazendo apenas uma referência à matriz em questão. A solução para este problema foi alterar a ordem do código no programa com o objetivo de evitar que o Python subscrevesse a matriz.

4.5) Interpretação do método iterativo

Inicialmente, desenvolvemos o código do método iterativo sem utilizar operações que envolviam o parâmetro alpha. Apesar de o vetor solução se aproximar do resultado do método de escalonamento, percebemos que poderíamos aperfeiçoar o resultado incluindo operações envolvendo o parâmetro alpha. Dessa forma, modificamos o vetor solução do método iterativo e obtivemos melhores resultados.

5) Resultados

Vamos agora apresentar alguns resultados do programa.

5.1) Rede cacique-tribo

Para 20 caciques, observamos uma diferença da ordem de 1e-7 nos pesos entre os dois métodos de resolução. Vale notar que a ordem das páginas nos rankings, em ambos métodos, difere apenas entre os índios de uma mesma tribo, devido a problemas de arredondamento numérico computacional. Além disso, obtivemos os seguintes resultados que mostram a superioridade em eficiência do método iterativo em relação ao escalonamento:

```
Número de iterações do Método Iterativo: 35
Tempo de execução do Método de Escalonamento (em segundos): 1.3742663860321045
Tempo de execução do Método Iterativo (em segundos): 0.08181118965148926
```

Para 40 caciques, a diferença de performance se torna ainda maior:

```
Número de iterações do Método Iterativo: 37
Tempo de execução do Método de Escalonamento (em segundos): 87.24118638038635
Tempo de execução do Método Iterativo (em segundos): 1.2112243175506592
```

Percebemos ainda que o número de iterações no método iterativo fica entre 35 e 37 devido à natureza do tipo de arquitetura da rede cacique-tribo. Com esse tipo de arquitetura, o programa converge com menos iterações do que por exemplo quando se tem uma rede mais aleatória.

5.2) Arquivo inserido pelo usuário

Realizamos o teste com a rede de 8 páginas sugerida no PDF do projeto introduzindo-a por meio de um arquivo .txt (rede_8_paginas.txt).

Como resultado, obtivemos pesos oriundos dos dois métodos com diferença na segunda casa decimal. Além disso, observamos que o número de iterações do método iterativo foi de 61, devido às poucas ligações existente na rede de exemplo.

Além disso, devido ao tamanho relativamente pequeno da rede, o algoritmo de escalonamento obteve performance mais eficiente:

```
Número de iterações do Método Iterativo: 61
Tempo de execução do Método de Escalonamento (em segundos): 0.0
Tempo de execução do Método Iterativo (em segundos): 0.0029969215393066406
```

Para 32 páginas, temos:

```
Número de iterações do Método Iterativo: 89
Tempo de execução do Método de Escalonamento (em segundos): 0.012987613677978516
Tempo de execução do Método Iterativo (em segundos): 0.1084282398223877
```

Para 64 páginas, temos:

```
Número de iterações do Método Iterativo: 90
Tempo de execução do Método de Escalonamento (em segundos): 0.16192412376403809
Tempo de execução do Método Iterativo (em segundos): 0.4102823734283447
```

Notamos que quanto maior a quantidade de páginas, menor é a diferença de tempo de execução entre o método de escalonamento e iterativo.

6) O que aprendemos?

No decorrer do projeto, fomos expostos a diferentes conceitos de álgebra linear, como eliminação de Gauss, pivotamento, normalização de vetores, auto-valores, auto-vetores, auto-espaço, raio espectral, matrizes primitivas, matrizes irredutíveis, matrizes estocásticas, cadeias de Markov e o teorema de Perron-Frobenius.

Ainda que não estudados com a devida profundidade necessária, foi possível entender basicamente do que se trata cada um desses conceitos e suas relações com o presente projeto, especialmente no que tange à teoria que sustenta o Power Method.

No caso do método de resolução por escalonamento, reforçamos aquilo que já havíamos aprendido tanto no ensino médio quanto no início do curso de Modelagem Matemática que envolve os conceitos de eliminação de Gauss.

No que tange ao método iterativo, compreendemos que o ranking de páginas é obtido a partir de um autovetor associado ao maior autovalor real único de uma

matriz cujas entradas são todas positivas e cujas colunas são compostas por vetores cuja soma é 1, resultado que está baseado no teorema de Perron-Frobenius.

No desenvolvimento do projeto, fomos expostos ao fato de que a estrutura de hyperlinks da World Wide Web pode ser modelada como uma cadeia de Markov, cujas entradas descrevem a probabilidade, em um trajeto aleatório, de ir de uma página para a outra.

Também aprendemos que essa teoria somente pode ser aplicada à Web caso ela não tenha o que podemos chamar de páginas mortas (dangling pages), que são páginas que não contêm hyperlinks, ou seja, que não apontam para nenhuma outra página. A solução nesse caso é atribuir para os zeros o valor correspondente a 1/n, onde n é o tamanho da matriz quadrada.

Ou seja, ela só pode ser aplicada se tivermos uma matriz irredutível, cujo grafo correspondente mostra que cada página (nó) pode ser alcançada a partir de qualquer outra página.

Aprendemos ainda que uma cadeia de Markov irredutível com uma matriz de transição primitiva é chamada de cadeia aperiódica e que a irredutibilidade da cadeia garante a existência do vetor solução que buscamos para realizar o ranking de páginas.

Também aprendemos que para se obter a matriz irredutível que garantirá a existência do vetor solução buscado é necessário forçar uma modificação com base no parâmetro alpha.

De maneira geral, a realização deste projeto nos trouxe ricas experiências tanto na área de programação, quanto na matemática.