Relatório EP1
Machine Learning
MAP3121

Gustavo Carvas 10335962

Rafael Sobral 10337193

21 de maio de 2019

1 Descrição

Em nosso trabalho escrevendo a solução para o problema no enunciado do Exercício-Programa, tivemos como primeiro e principal objetivo obter as soluções desejadas pelo enunciado, e o segundo objetivo foi, dentro das bibliotecas permitidas, otimizá-lo, para não passarmos dias rodando o Exercício-Programa.
Nosso código segue a seguinte lógica de programação:Inicialmente o usuário tem uma escolha, realizar os exercícios a),b),c) ou d) do enunciado ou realizar o treinamento. Caso selecione realizar o treinamento, após pedir o número de arquivos que quer usar para treinar, para testar, e para montar a matriz W_d para o usuário, temos um laço que conta de zero a nove para montar as matrizes W_d de cada um desses dígitos d. Nesse loop, o arquivo de treino é aberto, lido, e com ele é montada a matriz A, já normalizada. Iniciamos W como uma matriz randômica e logo após, entramos no laço principal do treino.
Este laço só se encerra quando é atingido o valor máximo de iterações permitido ou o módulo da diferença entre o erro anterior e o atual, conforme sugerido pelo enunciado, seja inferior a 10^{-5}. Dentro desse laço, realiza-se o algoritmo de normalizar a matriz W, escaloná-la usando RotGivens junto da matriz A original, e usar isso pra calcular H. Depois, transformamos H em uma matriz positiva transpomos ela, usando-a como matriz de coeficientes para resolver o problema $H^tW^t = A^t$. Para isso, escalonamos a H transposta junto da A original transposta, calculamos W^t com isso, e depois transpomos W de volta. O erro é calculado sempre com a matriz A original e as matrizes W e H encontradas naquela iteração, e tirando-se a raiz quadrada da soma de todas as diferenças entre os elementos de A e de WH ao quadrado.
Uma vez que o programa sai do laço, seja por ter atingido o número de iterações máximas ou encontrado um erro inferior a 10^{-5}, a matriz W encontrada é a W_d e é armazenada em um dicionário com todas as W_d. Após o fim desse laço, é realizado o teste, com o arquivo das imagens de teste. Para isso, resolve-se o problema $W_dH = A$, onde A é a matriz com os dígitos de teste, escalonando W_d com A e depois usando essas matrizes para encontrar H, e calculamos o erro dessa solução para cada coluna de A. Isso é feito para todas as W_d encontradas e o dígito que tiver o menor erro para determinada coluna é armazenado como o dígito correspondente àquela coluna.
Por fim, é feita somente a comparação entre os resultados encontrados e os gabaritos presentes em outro arquivo fornecido. É calculada a porcentagem de acertos de cada um dos dígitos, com relação ao total desse dígito presentes no gabarito, e a porcentagem total de acertos.

1.1 Funções

Nesta seção está a explicação de cada uma das funções implementadas

1.1.1 EncontraWeA(M1,K2)

Recebe uma matriz M1 (na verdade o W ou H^t) e K2 (na verdade a matriz original A, que representa a imagem no treinamento e nos testes) e então usa a M1 para identificar as rotações que devem ser feitas achando 'c' e 's' e então chama a RotGivensComA para realizar as rotações efetivamente.

1.1.2 RotGivensComA(i,j,c,s,W,A)

Realiza as rotações de Givens definidas pelos parâmetros 'c' e 's' sobre as matrizes W e A, porém é mais otimizada em relação à função recomendada no enunciado, ela realiza operações em linhas inteiras de cada uma das matrizes usando a biblioteca numpy.

1.1.3 CalculaH(M1,M2)

Utilizando o W e A rotacionados na 'EncontraWeA', resolvemos o sistema múltiplo para encontrar o H.

1.1.4 montaA(arq,ndig_treino)

Essa função é responsável por ler o arquivo e montar a matriz A que usaremos, seja no treino ou no teste.

1.1.5 TreinamentoClassificacao()

É a função que realmente comanda o processo de Machine Learning proposto no enunciado do EP. Inicialmente, utilizando as entradas do usuário, usa as imagens de treino definidas para criar as matrizes W_d para cada um dos dígitos resolvendo iterativamente o problema $WH = A$ e então passa para a etapa teste. Nesta etapa, le-se o arquivo que contém as linhas das matrizes que definem as imagens teste, e então resolvemos o sistema $W_dH = A$ para cada um dos dígitos, onde A é a matriz que representa a imagem do dígito a ser identificado.
Então calculamos o erro para cada um dos dígitos e o que tiver menor erro é então definido como o dígito da imagem. Por ultimo comparamos nossas respostas com o gabarito das imagens teste e obtemos a nossa porcentagem de acerto para os testes com os parametros iniciais (numero de imagens teste e colunas em W).

1.1.6 Exercicio()

Realiza um dos exercícios definidos no enunciado do Exercício Programa, dependendo do input do usuário e imprime o resultado.

1.1.7 main()

Essa é a função que deve ser rodada para rodar o EP, é a que recebe os inputs iniciais do usuário para então direcioná-lo ou para a função de exercícios ou para a função de treino e testes.

2 Tarefas

2.1 Primeira Tarefa

Abaixo seguem as tarefas pedidas em a), b) c) e d). Ressaltamos que talvez seja interessante realizar os exercícios no próprio programa a fim de facilitar a visualização das matrizes e vetores:

a) $n = m = 64$, $W_{ii} = 2$, i = 1; n; $W_{ij} = 1$; se $|i - j| = 1$ e $W_{ij} = 0$; se |i - j| > 1; Use b(i) = 1; i = 1; n.

Usando esses parâmetros o x com arredondamento de três casas decimais para melhor representação foi esse:

x =[0.492, 0.015, 0.477, 0.031, 0.462, 0.046, 0.446, 0.062, 0.431, 0.077, 0.415, 0.092, 0.4 , 0.108, 0.385, 0.123, 0.369, 0.138, 0.354, 0.154, 0.338, 0.169, 0.323, 0.185, 0.308, 0.2 , 0.292, 0.215, 0.277, 0.231, 0.262, 0.246, 0.246, 0.262, 0.231, 0.277, 0.215, 0.292, 0.2 , 0.308, 0.185, 0.323, 0.169, 0.338, 0.154, 0.354, 0.138, 0.369, 0.123, 0.385, 0.108, 0.4 , 0.092, 0.415, 0.077, 0.431, 0.062, 0.446, 0.046, 0.462, 0.031, 0.477, 0.015, 0.492]

b) $n = 20; m = 17$, $W_{ij} = \frac{1}{i+j-1}$; se |i - j| ≤ 4 e $W_{ij} = 0$; se |i - j| > 4; Use $b(i) = i$; $i = 1$; n.

Usando esses parâmetros o x com arredondamento de três casas decimais para melhor representação foi esse:

x =[56.358, -45.875, -43.489, -48.577, -30.14, 89.812, 48.714, 59.239, 11.446, 109.163, -72.873, -54.363, -51.444, -25.015, 98.572, 218.659, 298.4]

c) $n = m = 64$, $W_{ii} = 2$, i = 1; n; $W_{ij} = 1$; se $|i - j| = 1$ e $W_{ij} = 0$; se |i - j| > 1; Defina m = 3, resolvendo 3 sistemas simultâneos, com $A_{i1} = 1$; $A_{i2} = i$; $A_{i3} = 2i - 1$; $i = 1; n$.

Usando esses parâmetros o H com arredondamento de três casas decimais para melhor representação foi esse:

H =
[[0.492, 0.0, -0.492], [0.015, 1.0, 1.98], [0.477, 0.0, -0.477],
[3.1, 2.0, 3.97], [0.462, 0.0, -0.462], [0.046, 3.0, 5.954],

[0.446, 0.0, -0.446], [0.062, 4.0, 7.938], [0.431, 0.0, -0.431],
[0.077, 5.0, 9.923], [0.415, 0.0, -0.415], [0.092, 6.0, 11.908],
[0.4, 0.0, -0.4], [0.108, 7.0, 13.892], [0.385, 0.0, -0.385],
[0.123, 8.0, 15.877], [0.369,0.0, -0.369], [0.138, 9.0, 17.862],
[0.354,0.0,-0.354],[0.154,10.0,19.846],[0.338, 0.0,-0.338],
[0.169,11.0,21.831], [0.323, 0.0, -0.323], [0.185, 12.0, 23.815],
[0.308, 0.0, -0.308], [0.2, 13.0, 25.8], [0.292, 0.0, -0.292],
[0.215, 14.0, 27.785], [0.277, 0.0, -0.277], [0.231, 15.0, 29.769],
[0.262, 0.0, -0.262], [0.246, 16.0, 31.754], [0.246, 0.0, -0.246],
[0.262, 17.0, 33.738], [0.231, 0.0, -0.231], [0.277, 18.0, 35.723],
[0.215, 0.0, -0.215], [0.292, 19.0, 37.708], [0.2, 0.0, -0.2],
[0.308, 20.0, 39.692], [0.185, 0.0, -0.185], [0.323, 21.0, 41.677],
[0.169, 0.0, -0.169], [0.338, 22.0, 43.662], [0.154, 0.0, -0.154],
[0.354, 23.0, 45.646], [0.138, 0.0, -0.138], [0.369, 24.0, 47.631],
[0.123, 0.0, -0.123], [0.385, 25.0, 49.615], [0.108, 0.0, -0.108],
[0.4, 26.0, 51.6], [0.092, 0.0, -0.092], [0.415, 27.0, 53.585],
[0.077, 0.0, -0.077], [0.431, 28.0,55.569], [0.062, 0.0, -0.062],
[0.446, 29.0, 57.554], [0.046, 0.0, -0.046], [0.462, 30.0, 59.538],
[0.031, 0.0, -0.031], [0.477, 31.0, 61.523], [0.015, 0.0, -0.015],
[0.492,32.0, 63.508]]

d) $n = 20; p = 17$, $W_{ij} = \frac{1}{i+j-1}$; se |i - j| ≤ 4 e $W_{ij} = 0$; se |i - j| > 4; Defina $m =$ 3, resolvendo 3 sistemas simultâneos, com $A_{i1} = 1; A_{i2} = i; Ai3 = 2i-1; i = 1; n$.

Usando esses parâmetros o H com arredondamento de três casas decimais para melhor representação foi esse:

$H =$
[[2.882, 56.358, 109.834], [-1.834, -45.875, -89.916], [-1.514, -43.489, -85.464], [-1.522, -48.577, -95.631], [-0.454, -30.14, -59.827], [5.857, 89.812, 173.768], [3.422, 48.714, 94.005], [3.657, 59.239, 114.822], [1.204, 11.446, 21.689], [6.125, 109.163, 212.2], [-2.48, -72.873, -143.266], [-1.478, -54.363, -107.248], [-1.204, -51.444, -101.685], [0.128, -25.015, -50.158], [6.502, 98.572, 190.642], [11.491, 218.659, 425.827], [14.581, 298.4, 582.22]]

2.2 Segunda Tarefa

Após implementar a fatoração por matrizes não negativas e inicializando o loop com

$$A = \begin{vmatrix} \frac{3}{10} & \frac{3}{5} & 0 \\ \frac{1}{2} & 0 & 1 \\ \frac{4}{10} & \frac{4}{5} & 0 \end{vmatrix}.$$

e com W com valores randômicos, alcançamos esses resultados:

$$W = \begin{vmatrix} 1.96 * 10^{-7} & 0.6 \\ 1 & 4.13 * 10^{-17} \\ 2.61 * 10^{-7} & 0.8 \end{vmatrix}.$$

$$H = \begin{vmatrix} 0.5 & 0 & 1 \\ 0.49 & 1 & 0 \end{vmatrix}.$$

Como é possível ver, as colunas na matriz W estão trocadas, porém isso não importa como na matriz H as linhas também estão trocadas, ou seja, a multiplicação das matrizes ainda tem o mesmo resultado do que teria caso não estivessem trocadas.
Alguns dos números aparecem elevados a potencias negativas de 10 bastante altas, fazendo com que basicamente sejam zero, por isso nosso resultado ainda está de acordo com a resposta definida no enunciado do exercício.

2.3 Tarefa Principal

Segue a etapa do EP que efetivamente realiza Machine Learning, em todos os testes foram feitos com 10000 imagens de teste (sem contar as usadas para treinamento)

2.3.1 100 imagens treino

a. Parâmetros: 100 imagens de treino, 5 colunas para W:

- Acertos totais: 8789 de 10000
- Tempo total do programa: 153.67070055007935 seg (aproximadamente 2min30s)
- Porcentagem total de acertos: 87.89%
- Porcentagem de acerto de cada dígito:
  (a) 1: 99.38325991189427
  (b) 2: 84.98062015503875
  (c) 3: 85.34653465346534
  (d) 4: 79.73523421588594
  (e) 5: 82.95964125560538
  (f) 6: 93.42379958246346
  (g) 7: 87.7431906614786
  (h) 8: 79.05544147843942
  (i) 9: 87.21506442021804

b. Parâmetros 100 imagens de treino, 10 colunas para W:

- Acertos totais: 8986 de 10000
- Tempo total do programa: 245.7024872303009 seg (aproximadamente 3 min 47 seg)

- Porcentagem total de acertos: 89.86%
- Porcentagem de acerto de cada dígito:
  (a) 0: 97.6530612244898,
  (b) 1: 99.55947136563876,
  (c) 2: 89.14728682170544,
  (d) 3: 85.44554455445544,
  (e) 4: 87.06720977596741,
  (f) 5: 85.76233183856502,
  (g) 6: 95.61586638830897,
  (h) 7: 90.75875486381322,
  (i) 8: 77.92607802874744,
  (j) 9: 87.21506442021804

c. Parâmetros 100 imagens de treino, 15 colunas para W:

- Acertos totais: 9053 de 10000
- Tempo total do programa: 355.0198006629944 seg (aproximadamente 5 min 55 seg)
- Porcentagem total de acertos: 90.53%
- Porcentagem de acerto de cada dígito:
  (a) 0: 97.55102040816327,
  (b) 1: 99.47136563876651,
  (c) 2: 89.82558139534885,
  (d) 3: 86.33663366336634,
  (e) 4: 86.9653767820774,
  (f) 5: 85.20179372197309,
  (g) 6: 95.19832985386222,
  (h) 7: 93.8715953307393,
  (i) 8: 81.51950718685832,
  (j) 9: 87.61149653121902

d. Erros absolutos giraram em torno de 100 até 150, o que é razoável, uma vez que para calcular esse problema, tinhamos 78400 elementos na matriz A. Pudemos perceber que conforme aumentamos o número de colunas da W, o tempo de execução do nosso código aumentou proporcionalmente, de forma quase linear para esses três primeiros testes. Nesses três testes, o aumento percentual de acerto em relação ao tamanho da W foi diminuindo quanto maior era a W, porém para os três valores de p sugeridos, o aumento foi relevante o suficiente para justificar seu aumento.

2.3.2 1000 imagens de treino

a. Parâmetros 1000 imagens de treino, 5 colunas para W:

- Acertos totais: 9089
- Tempo total do programa: 433.405473947525 seg (aproximadamente 7 min 13 seg)
- Porcentagem total de acertos: 90.89
- Porcentagem de acertos de cada dígito:
  (a) 0: 97.75510204081633,
  (b) 1: 99.38325991189427,
  (c) 2: 88.17829457364341,
  (d) 3: 90.89108910891089,
  (e) 4: 86.35437881873727,
  (f) 5: 88.34080717488789,
  (g) 6: 95.19832985386222,
  (h) 7: 89.29961089494164,
  (i) 8: 85.72895277207392,
  (j) 9: 86.62041625371654

b. Parâmetros 1000 imagens de treino, 10 colunas para W:

- Acertos totais: 9274
- Tempo total do programa: 858.9939441680908 seg (aproximadamente 14 minutos)
- Porcentagem total de acertos: 92.74%
- Porcentagem de acertos de cada dígito:
  (a) 0: 98.26530612244898
  (b) 1: 99.47136563876651
  (c) 2: 89.92248062015504
  (d) 3: 91.38613861386139
  (e) 4: 92.56619144602851
  (f) 5: 89.01345291479821
  (g) 6: 96.13778705636743
  (h) 7: 91.43968871595331
  (i) 8: 88.60369609856262
  (j) 9: 89.59365708622398

c. Parâmetros 1000 imagens de treino, 15 colunas para W:

- Acertos totais: 9323
- Tempo total do programa: 1109.2416377067566 seg (aproximadamente 18 min 29 seg)

- Porcentagem total de acertos: 93.23%
- Porcentagem de acertos de cada digito:
  (a) 0: 98.67346938775509
  (b) 1: 99.47136563876651
  (c) 2: 91.66666666666666
  (d) 3: 91.48514851485149
  (e) 4: 92.9735234215886
  (f) 5: 89.12556053811659
  (g) 6: 96.24217118997912,
  (h) 7: 92.89883268482491
  (i) 8: 87.782340862423
  (j) 9: 90.8820614469772

d. Erros absolutos estiveram em torno de 350-500, o que é razoável, uma vez que a matriz A tinha 784000 elementos. Observamos que o tempo de execução do nosso código continuou aumentando de forma quase linear com o número de colunas na W também para esses 3 testes. O aumento percentual de acerto já começou razoavelmente maior do que para 100, e a diferença que o número de colunas da W pareceu ter a mesma relevância que para 100 colunas. Acreditamos que nesse caso, o aumento também foi relevante o suficiente para justificar seu aumento e o maior tempo de execução do código.

2.3.3 4000 imagens de treino

a. Parâmetros 4000 imagens de treino, 5 colunas para W:

- Acertos totais: 9186
- Tempo total do programa: 1344.8773918151855 seg (aproximadamente 22 min 25 seg
- Porcentagem total de acertos: 91.86%
- Porcentagem de acerto de cada dígito:
  (a) 0: 97.6530612244898
  (b) 1: 99.29515418502203
  (c) 2: 89.43798449612403
  (d) 3: 92.97029702970298
  (e) 4: 88.4928716904277
  (f) 5: 88.2286995515695
  (g) 6: 96.65970772442589
  (h) 7: 89.59143968871595
  (i) 8: 88.80903490759754
  (j) 9: 86.42220019821606

b. Parâmetros 4000 imagens de treino, 10 colunas para W:

- Acertos totais: 9357
- Tempo total do programa: 2355.5983533859253 seg (aproximadamente 39 min 16 seg)
- Porcentagem total de acertos: 93.57%
- Porcentagem de acerto de cada dígito:
  (a) 0: 98.77551020408163
  (b) 1: 99.47136563876651
  (c) 2: 92.24806201550388
  (d) 3: 92.87128712871288
  (e) 4: 93.58452138492872
  (f) 5: 89.01345291479821
  (g) 6: 96.76409185803759
  (h) 7: 91.63424124513618
  (i) 8: 89.01437371663245
  (j) 9: 91.2784935579782

c. Parâmetros 4000 imagens de treino, 15 colunas para W:

- Acertos totais: 9398
- Tempo total do programa: 3523.546833753586s (um pouco menos que 1 hora)
- Porcentagem total de acertos: 93.98%
- Porcentagem de acerto de cada dígito:
  (a) 0: 98.67346938775509,
  (b) 1: 99.29515418502203,
  (c) 2: 91.18217054263566,
  (d) 3: 92.57425742574257,
  (e) 4: 93.78818737270875,
  (f) 5: 91.81614349775785,
  (g) 6: 97.18162839248434,
  (h) 7: 92.80155642023347,
  (i) 8: 89.11704312114989,
  (j) 9: 92.66600594648166

d. Erros absolutos estiveram em torno de 800-920, o que é razoável, uma vez que a matriz A tinha 3136000 elementos. Pudemos observar que, assim como nos testes anteriores, o tempo de execução permaneceu aumentando de forma aproximadamente linear para o número de colunas da W. Já entre diferentes tamanhos de amostra para treino e com a mesma quantidade de colunas da W, aumentar o tamanho das amostras não representa um

aumento tão próximo de linear quanto o aumento de colunas da W. Além disso, para esse número de colunas da W, entre o segundo e o terceiro teste o aumento percentual de acertos foi muito pequeno para justificar o tempo a mais que levamos executando o código.

3 Conclusões

Ao rodar o programa para esses três testes, pudemos perceber que, conforme aumentamos a base de dados para treino, temos como obter soluções mais precisas para o problema. Aumentar a quantidade de colunas da W_d também foi importante, melhorando razoavelmente a eficiência do nosso treinamento, entretanto é necessário tomar cuidado com essa parte. Rodamos testes com p muito maiores do que os sugeridos, com o mesmo ndig_treino e a eficiência diminuiu, podendo ter acontecido overfitting, fenômeno em que a nossa estimativa deixa de representar a realidade e passa a representar somente o banco de dados que usamos para treiná-la.
Resolvendo esse problema, pudemos perceber a importância de não se acomodar com a primeira solução que encontrar para um problema de programação, porque sempre pode haver um método mais rápido de resolvê-lo, com a ajuda de outras bibliotecas e coisas do tipo.