Impactos da Base de Aprendizagem em Diferentes Classificadores Lineares

Thiago Prado de Campos[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
ACF Centro Politécnico – Jd. Das Américas – CEP 81531-980 – Curitiba – PR

contato@thiagotpc.com

Resumo. *Este é um relatório de atividade de laboratório da disciplina de Aprendizagem de Máquina (2020/Período Especial).*

1. Classificadores

Existem diversos algoritmos para classificação de dados. Neste laboratório aborda-se os algoritmos de classificação lineares: *K Nearest Neighbor (KNN), Naive Bayes, Linear Discriminant Analisys (LDA), Logistic Regression e Perceptron*.

O *KNN* consiste em classificar um elemento com base nos k exemplos mais próximos no espaço de recursos.

O *Naive Bayes* é um classificador probabilístico baseado no Teorema de Bayes com suposição de independência das características.

O *LDA* tenta encontrar um plano por meio de transformação linear que sirva de fronteira para maximizar distância entre-classes e minimizar distância intra-classes.

O algoritmo de Regressão Logística (*Logistic Regression*) usa uma função logística para retornar uma classificação associada a um valor de probabilidade.

E o *Perceptron* é uma rede neural de um neurônio cujo somatório dos pesos associados às características e bias pode decidir a pertença ou não à uma classe.

2. Atividade Proposta

Dado um conjunto de amostras para treinamento (composto por 20 mil elementos) e um conjunto de testes para classificação (composto por aproximadamente 58 mil elementos), propõe-se experimentar o treinamento com quantidades diferentes de amostras (1000, 2000, ..., 20000) para cada um dos cinco classificadores citados acima e testar todos os elementos do conjunto de testes. Cada amostra e elemento do conjunto de testes é um vetor de características de tamanho 132.

O objetivo é comparar o desempenho e os resultados para cada arranjo de classificador e quantidade de amostras utilizadas.

3. Estratégia

Para alcançar o objetivo do laboratório foi criado um script na linguagem Python para carregar as amostras de treinamento e os dados para testes. Por meio de um laço de repetição (i) para cada 1000*i amostras até o total de amostras disponíveis realizou-se todo o processo de treinamento e testes para cada um dos cinco tipos de classificadores

investigados. 20 conjuntos de amostras combinando treinamento e testes com 5 classificadores, totalizam 100 execuções.

Estrutura Lógica da Estratégia

```
Para i de 1000 a 20000 passo 1000:
   Para cada classificador em {kNN, Naive Bayes, LDA, Regressão Linear, Peceptron}:
      Treinar e Classificar(amostras[0,i*1000])
Gravar Resultados
```

Para realizar o treino e classificação foi adotada a implementação padrão dos cinco algoritmos pela biblioteca Scikit Learn (https://scikit-learn.org/).

A saída do script retorna a matriz de confusão para cada teste e uma tabela contendo informações sobre todas as execuções, em arquivos do tipo CSV que depois foram analisados em *software* de Planilha Eletrônica.

Toda a implementação está disponível em repositório do GitHub: https://github.com/thiagotpc/ml-laboratorio-02, incluindo as saídas geradas pela execução.

4. Resultados Obtidos

4.1. Para 1.000 amostras

Com o tamanho mínimo de amostras para treino testada, o LDA foi o algoritmo de melhor resultado, em acurácia e F1-Score. Também foi o mais rápido para treinar. Os algoritmos Perceptron e de Regressão Logística tiveram velocidade semelhante. Já Os testes foram muito mais rápidos no algoritmo de Naive Bayes. KNN destacou-se pelo elevado tempo de treinamento.

Tabela 1 - Resultados para 1000 amostras de treinamento

Classificador	Tempo de Treinamento	Tempo de Testes	Acurácia	F1Score
LDA	0,0480	0,0780	0,7873	0,7873
KNN	17,0552	0,0310	0,7758	0,7820
Perceptron	0,0490	0,0430	0,7469	0,7561
Regressão Logística	0,0510	0,1229	0,7157	0,7291
Naive Bayes	1,3423	0,0070	0,6961	0,6744

4.2. Para 20.000 amostras

Com o tamanho máximo de amostras para treino, os melhores resultados foram obtidos pelo KNN, tanto em acurácia quanto em F1-Score. Porém, continuou com o pior tempo para treino. Os melhores tempo de treino continuaram com o LDA, Perceptron e Regressão Logística, enquanto o melhor tempo de testes ficou com o Naive Bayes.

Tabela 2 - Resultados para 20000 amostras de treinamento

Classificador	Tempo de Treinamento	Tempo de Testes	Acurácia	F1Score
KNN	182,7883	1,3632	0,9388	0,9385
LDA	0,0500	0,8935	0,9279	0,9279

Perceptron	0,0550	0,7316	0,9263	0,9270
Regressão Logística	0,0490	3,5959	0,9118	0,9118
Naive Bayes	1,2273	0,0660	0,8900	0,8891

4.3. Melhores desempenhos de Treinamento

Considerando as 100 execuções, os melhores tempos de treinamento ficaram com os algoritmos de LDA (13 de 20), Perceptron (9 de 20) e Regressão Logística (8 de 20) nos mais variados tamanhos de amostras, reforçando o descoberto nas seções acima.

Tabela 3 - 30 melhores tempos de treinamento

Classificador	Amostras	Tempo de Treinamento
perceptron	18000	0,0460
perceptron	17000	0,0470
perceptron	15000	0,0470
perceptron	12000	0,0470
perceptron	3000	0,0470
regressao_logistica	19000	0,0470
regressao_logistica	15000	0,0470
regressao_logistica	6000	0,0470
regressao_logistica	5000	0,0470
regressao_logistica	4000	0,0470
regressao_logistica	3000	0,0470
lda	10000	0,0480
lda	8000	0,0480
lda	5000	0,0480
lda	2000	0,0480
lda	1000	0,0480
perceptron	13000	0,0480
perceptron	2000	0,0480
perceptron	10000	0,0480
regressao_logistica	14000	0,0480
regressao_logistica	10000	0,0480
lda	11000	0,0490
lda	12000	0,0490
lda	13000	0,0490
lda	9000	0,0490
lda	14000	0,0490
lda	15000	0,0490
lda	7000	0,0490
lda	3000	0,0490
perceptron	1000	0,0490

4.4 Piores desempenho de Treinamento

O KNN teve todos os piores tempos de treinamento, aumentando conforme incremento nas amostras. O Naive Bayes completou a lista dos 30 piores tempo de treinamento com valores de amostras entre 1000-10000 e entre 10001-20000 alternando. Ou seja, o tempo não aumenta necessariamente apenas em função do tamanho da amostra.

Tabela 4 - 30 piores tempo de treinamento

Classificador	Amostras	Tempo de Treinamento
knn	17000	195,9883
knn	18000	194,3837
knn	20000	182,7883
knn	19000	174,4421
knn	16000	173,1757
knn	12000	146,9863
knn	14000	142,6824
knn	15000	142,3570
knn	13000	130,4503
knn	11000	116,3067
knn	10000	107,9432
knn	9000	95,8523
knn	8000	94,1488
knn	7000	76,9545
knn	6000	71,4895
knn	5000	59,8364
knn	4000	53,7262
knn	3000	38,8738
knn	2000	29,8039
knn	1000	17,0552
naive_bayes	1000	1,3423
naive_bayes	4000	1,2553
naive_bayes	3000	1,2553
naive_bayes	19000	1,2483
naive_bayes	8000	1,2473
naive_bayes	5000	1,2463
naive_bayes	9000	1,2423
naive_bayes	10000	1,2413
naive_bayes	16000	1,2343
naive_bayes	17000	1,2303

4.5. Melhores desempenho em Testes

O Naive Bayes teve todas as configurações de experimento figurando entre os 30 melhores tempo de testes, ocupando a maioria dos primeiros lugares. O KNN, LDA e o Perceptron, com número baixo de amostras (1000 a 3000) também entraram na lista. A Regressão Logística apareceu apenas na configuração de 1000 amostras.

Tabela 5 - 30 melhores tempos de testes

Classificador	Amostras	Tempo de Testes
naive_bayes	1000	0,0070
naive_bayes	2000	0,0080
naive_bayes	3000	0,0100
naive_bayes	4000	0,0120
naive_bayes	5000	0,0140
naive_bayes	6000	0,0160
naive_bayes	7000	0,0180
naive_bayes	9000	0,0240
naive_bayes	8000	0,0260
knn	1000	0,0310
naive_bayes	10000	0,0350
naive_bayes	11000	0,0370
naive_bayes	12000	0,0400
perceptron	1000	0,0430
naive_bayes	13000	0,0440
naive_bayes	14000	0,0450
naive_bayes	15000	0,0540
naive_bayes	16000	0,0540
perceptron	2000	0,0590
naive_bayes	18000	0,0590
naive_bayes	17000	0,0650
naive_bayes	20000	0,0660
naive_bayes	19000	0,0660
lda	1000	0,0780
knn	2000	0,0859
perceptron	3000	0,0860
lda	2000	0,0939
lda	3000	0,1169
regressao_logistica	1000	0,1229
knn	3000	0,1299

4.6. Piores desempenhos em Testes

O pior tempo de teste coube a Regressão Logística, com todas as configurações >= 10000 amostras. Ela ocupou 16 das 30 primeiras posições. Também não teve bom tempo o KNN e o LDA, para configurações de muitas amostras. O Perceptron figurou em tempo ruim de testes quando usou as 20 mil amostras para treino.

Tabela 6 - 30 piores tempos de testes

Classificador	Amostras	Tempo de Testes
regressao_logistica	20000	3,5959
regressao_logistica	19000	3,5530
regressao_logistica	18000	3,1952
regressao_logistica	17000	3,0153
regressao_logistica	16000	2,8624
regressao_logistica	15000	2,7394
regressao_logistica	13000	2,6305
regressao_logistica	14000	2,6295
regressao_logistica	12000	2,3187
regressao_logistica	11000	2,1478
regressao_logistica	10000	1,9739
knn	18000	1,7920
regressao_logistica	9000	1,7170
regressao_logistica	8000	1,4911
knn	20000	1,3632
knn	19000	1,3432
regressao_logistica	7000	1,2603
knn	17000	1,2223
knn	16000	1,1204
lda	20000	0,8935
lda	19000	0,8415
lda	18000	0,8355
regressao_logistica	6000	0,8265
knn	12000	0,8265
knn	15000	0,7656
lda	17000	0,7546
perceptron	20000	0,7316
knn	14000	0,7316
lda	16000	0,7036
regressao_logistica	5000	0,6656

4.7. Melhores Resultados em Acurácia

O KNN mostrou possuir melhor acurácia na maioria das configurações, mas, principalmente com bastante amostras. O LDA também obteve boa acurácia para grande número de amostras de treinamento. Já o Perceptron obteve bons resultados de acurácia mas variando o número de amostras. A melhor acurácia obtida por ele, por exemplo, foi com 11 mil amostras.

Tabela 7 - 30 mehores resultados em acurácia

Classificador	Amostras	Acuracia
knn	20000	0,9388
knn	19000	0,9385
perceptron	11000	0,9384
knn	18000	0,9376
knn	16000	0,9373
knn	17000	0,9371
knn	14000	0,9360
knn	13000	0,9360
knn	12000	0,9358
knn	15000	0,9356
perceptron	17000	0,9352
knn	10000	0,9351
knn	11000	0,9347
perceptron	18000	0,9341
knn	9000	0,9329
knn	8000	0,9296
perceptron	13000	0,9283
lda	20000	0,9279
lda	10000	0,9279
lda	19000	0,9265
perceptron	20000	0,9263
lda	11000	0,9261
knn	7000	0,9251
lda	18000	0,9251
lda	12000	0,9246
lda	13000	0,9243
lda	9000	0,9238
lda	17000	0,9232
perceptron	19000	0,9224
lda	14000	0,9221

4.8. Melhores Resultados em F1-Score

O resultado em relação ao F1-Score foi muito próximo aos obtidos em acurácia. Novamente o KNN dominou a lista dos 30 melhores resultados, seguido por LDA e Perceptron, na mesma proporção encontrada com relação a acurácia.

Tabela 8 - 30 melhores resultados em F1-Score

Classificador	Amostras	F1Score
knn	20000	0,9385
perceptron	11000	0,9384
knn	19000	0,9382
knn	18000	0,9373
knn	16000	0,9371
knn	17000	0,9369
knn	13000	0,9358
knn	14000	0,9357
knn	12000	0,9355
knn	15000	0,9353
perceptron	17000	0,9353
knn	10000	0,9348
knn	11000	0,9344
perceptron	18000	0,9339
knn	9000	0,9326
knn	8000	0,9293
perceptron	13000	0,9282
lda	20000	0,9279
lda	10000	0,9279
perceptron	20000	0,9270
lda	19000	0,9264
lda	11000	0,9261
lda	18000	0,9251
knn	7000	0,9248
lda	12000	0,9245
lda	13000	0,9242
lda	9000	0,9237
lda	17000	0,9231
perceptron	19000	0,9223
lda	14000	0,9220

4.9. Convergência da Acurácia em Função de Amostras

Ao plotar a evolução da acurácia em função da quantidade de amostras utilizadas para treino, percebe-se que, a partir de 9 mil ou 10 mil amostras atinge-se uma estabilização em todos os algoritmos, exceto no Perceptron. No caso do Perceptron, provavelmente em função do fenômeno de esquecimento.

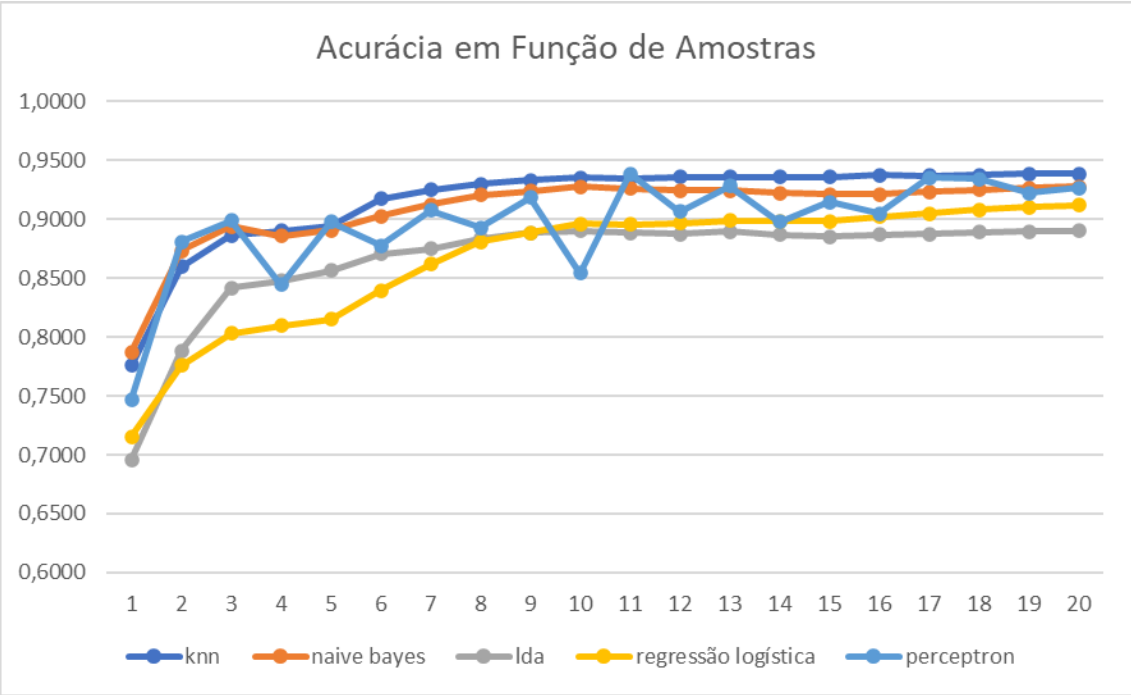

Figura 1 - Acurácia em Função de Amostras Utilizadas

4.10. Convergência do F1-Score em Função de Amostras

O mesmo comportamento se observa com relação ao F1-Score. A partir de 9 ou 10 mil amostras, há estabilização nos resultados. Mas aqui, há de se destacar uma melhora da Regressão Logística no intervalo de 15 a 20 mil amostras.

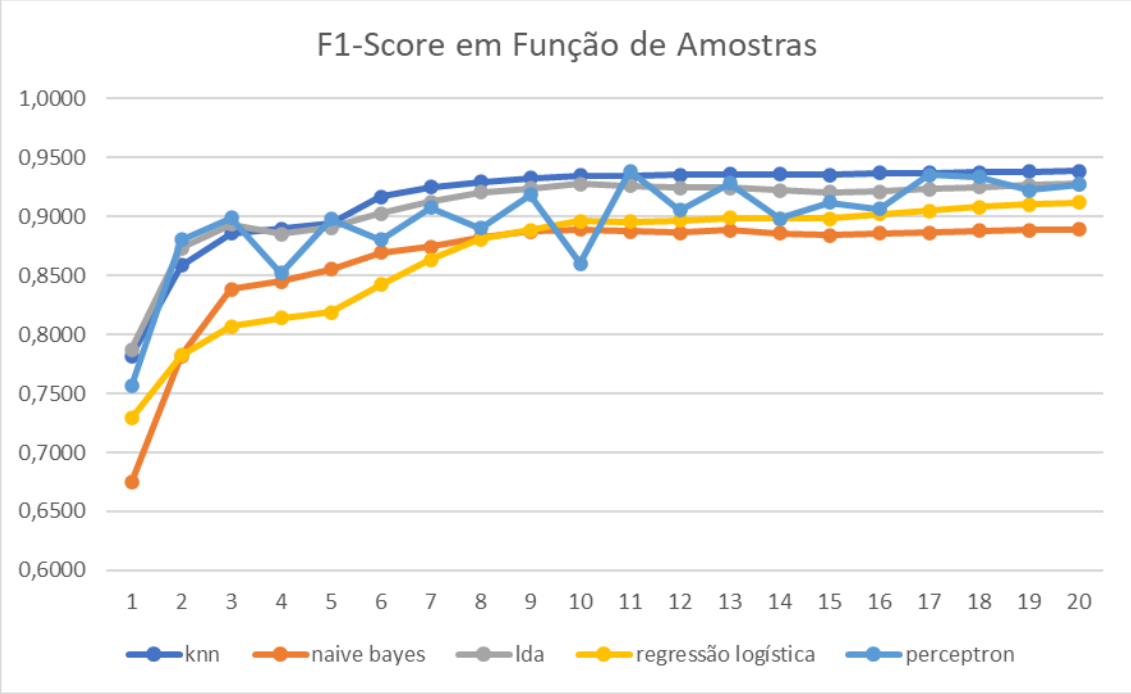

Figura 2 - F1-Score em Função de Amostras Utilizadas

4.11. Análise das Matrizes de Confusão

Observando as matrizes de confusão de cada classificador – Tabelas 10, 11, 12, 13 e 14 - quando do uso de todo conjunto de amostras para treinamento, destacamos os erros maiores ou igual a 294 ocorrências (0,5% de 58.646) na Tabela 9.

Com este critério, o principal erro do KNN é classificar a classe 3 como 5. O Naive Bayes erra classificando 2 como 1, 3 como 2 ou 7 e; 7 como 1. O LDA possui erros mais espalhados, sendo os maiores, 3 por 5 e 9 por 8. A Regressão Logística erra classificando 3 por 5 e; o Perceptron erra principalmente 0 com 8 e 3 com 8.

Podemos concluir que a classe que gera mais erros é a 3, sendo mais frequentemente confundida com 5, mas também podendo ser erroneamente classificada como 2, 7 ou 8. A 8, aliais, é a escolha errada mais comum também para as classes 0 e 9, mas só em alguns algoritmos (LDA e Perceptron, respectivamente).

Tabela 9 - Erros mais comuns das Matrizes de Confusão

Algoritmo	0	1	2	3	4	5	6	7	8	9
KNN				5						
NB				2 ou 7				1		
LDA				5						8
RL				5						
Perceptron	8			8						

Tabela 10 - Matriz de Confusão para KNN

	0	1	2	3	4	5	6	7	8	9
0	5472	3	1	15	6	2	26	2	32	1
1	0	6105	175	119	56	6	35	66	34	59
2	12	11	5607	165	3	1	16	51	20	2
3	4	1	25	5646	2	51	1	53	20	16
4	12	11	13	3	5305	9	132	24	11	202
5	9	3	9	489	4	4842	41	16	83	43
6	31	10	4	2	3	44	5724	0	40	0
7	1	25	41	119	54	1	0	5773	7	76
8	36	24	42	114	32	38	50	27	5165	167
9	16	9	17	107	78	9	9	131	34	5403

Tabela 11 - Matriz de Confusão para Naive Bayes

	0	1	2	3	4	5	6	7	8	9
0	5220	1	11	32	2	1	41	0	251	1
1	1	5184	583	238	86	22	85	340	80	36
2	9	24	5289	447	4	1	8	52	53	1
3	2	1	212	5390	1	33	0	127	31	22
4	14	2	44	12	5273	0	32	44	90	211
5	9	6	29	103	31	4958	46	2	169	186
6	78	7	89	8	15	90	5286	0	285	0
7	1	47	175	426	21	1	1	5323	60	42
8	175	5	53	182	23	7	38	13	5112	87
9	25	5	62	151	221	4	0	55	184	5106

Tabela 12 - Matriz de Confusão para LDA

	0	1	2	3	4	5	6	7	8	9
0	5358	10	11	15	19	0	47	17	80	3
1	0	6027	222	85	9	22	38	199	31	22
2	22	41	5605	12	1	0	4	175	27	1
3	1	12	29	5470	1	19	1	247	23	16
4	20	71	42	0	5208	0	86	5	29	261
5	9	11	6	314	4	5015	50	24	67	39
6	77	49	37	15	56	36	5460	0	125	3
7	0	58	47	6	58	1	0	5882	22	23
8	80	59	38	5	51	29	54	57	4961	361
9	34	31	9	91	69	7	16	98	29	5429

Tabela 13 - Matriz de Confusão para Regressão Logística

	0	1	2	3	4	5	6	7	8	9
0	5380	5	17	12	15	4	69	6	51	1
1	1	5595	116	269	197	74	179	74	80	70
2	22	18	5585	89	12	1	33	82	45	1
3	4	3	37	5598	16	39	1	73	20	28
4	35	8	30	1	5315	2	104	41	9	177
5	6	11	24	498	78	4728	49	22	73	50
6	88	26	0	1	20	94	5517	0	112	0
7	0	41	40	121	167	2	0	5598	17	111
8	83	43	47	59	84	46	54	58	4998	223
9	55	22	8	143	251	0	4	151	18	5161

Tabela 14 - Matriz de Confusão para Perceptron

	0	1	2	3	4	5	6	7	8	9
0	5532	1	0	6	0	1	18	1	1	0
1	14	6114	46	217	14	176	27	43	2	2
2	88	32	5548	137	2	0	16	62	3	0
3	5	3	12	5698	0	60	1	28	2	10
4	116	13	46	17	5172	7	108	39	5	199
5	21	5	4	129	3	5318	40	1	6	12
6	129	8	5	4	5	57	5648	0	2	0
7	2	42	51	157	31	4	0	5796	1	13
8	329	39	45	457	35	225	185	20	4211	149
9	89	36	26	115	106	25	3	83	3	5327

4.12. Algoritmos Complementares

Com base no entendimento das matrizes de confusão, pode ser que combinando algoritmos diferentes obtenha-se melhores resultados. Por exemplo, Regressão Logística e Perceptron ou KNN e Naive Bayes possuem erros diferentes de forma que, um arranjo entre os classificadores pode minimizar ou "corrigir" os principais erros.

5. Considerações Finais

Observa-se, ao final, que para o caso aqui tratado, a quantidade de amostras usadas no treinamento de fato impacta no resultado final, seja em termos de acurácia quanto de desempenho Vimos que a partir de 9 ou 10 mil exemplos, atinge-se um certo platô de otimização da classificação para a maioria dos algoritmos.

Outro aprendizado é com relação ao tempo de aprendizado e versus a acurácia. O KNN, por exemplo, tem pior tempo de treinamento, mas o melhor resultado de classificação. Já LDA e Perceptron parecem equilibrar melhor estes dois fatores, apresentando baixo tempo de treino e bom resultado de classificação.

No algoritmo de Regressão Logística, um *warning (ConvergenceWarning)* foi emitido quando se atingiu o limite de interações máximas padrão (max_iter = 100). Na sequência foi executada a bateria de testes novamente, desta vez aumentando o número máximo de interações para 200. Não houve *warning*, mas não modificou significativamente os resultados.

O parâmetro tempo de execução (tanto de treinamento quanto de testes) deve ser interpretado com cautela pois pode ser influenciado por outros processos em andamento ou lançados pelo sistema operacional durante a execução do script Python que, naturalmente, concorrerão com uso de CPU e memória. Portanto, algumas diferenças em escala de milissegundos ou mesmo poucos segundos podem ocorrer a cada execução e, portanto, não devem ser necessariamente enfatizadas.