Relatório Final de Aprendizado de Máquina Sobre o Banco de Dados MNIST

Carlos F. C. Mello

Instituto Metrópole Digital - Universidade Federal do Rio Grande do Norte(UFRN)
Natal - Rio Grande do Norte - Brasil

carvalheirafc@gmail.com

Abstract. *This paper explores the usage and performance of different supervised learning algorithms, ensemble classifiers methods and homogeneous stacking on MNIST Handwritten digits database. Also the impact of usage of generated subsets of the original database.*

Resumo. *Esse artigo explora o uso e performance de diferentes algoritmos de aprendizado supervisionados, comitês classificadores e técnicas de empilhamentos de classificadores homogêneas sobre o conjunto de dados MNIST Handwritten digits. Também explora o impacto na performance de cada classificador usando diferentes versões deste dataset.*

1 Introdução

Nesse artigo, irei explorar o dataset MNIST, composto por imagens de dígitos escritos a mão e tentar classificá-los usando diferentes algoritmos de aprendizagem supervisionadas em diferentes técnicas de comitês classificadores e com diferentes configurações de treinamento.

A ideia desse trabalho é verificar o comportamento e performance de algoritmos supervisionados e o impacto de uso de comitês classificadores usando diferentes algoritmos como estratégias de classificação.

2 Base de Dados

A base de dados utilizada no projeto é a MNIST Handwritten Digits, no caso específico deste trabalho foi utilizado apenas do original o conjunto de treino, contendo 60000 amostras de imagens de dígitos escritos a mão variando de 0 a 9,

em imagens monocromáticas de 28x28 representadas em matrizes do mesmo tamanho com os seus elementos variando de 0(preto total) a 255(branco total).

A partir da base de treino do MNIST foram então criadas 3 novos conjuntos, onde dois deles tem como o objetivo a diminuição da quantidade de amostras do conjunto, sendo assim uma base com 30000 amostras e um outro conjunto contendo ainda menos casos 12000 casos. A terceira e última base foi criada tendo em mente criar um problema linearmente separável. Então sobre o conjunto inicial foram tiradas apenas os casos onde as amostras eram dos dígitos 0 e 1, fazendo assim com que o problema agora tivesse apenas duas classes finais linearmente separáveis e por consequência de casos dos valores no conjunto de dados original um valor também reduzido em número de casos, 12665 amostras.

Então ao final todas as bases de dados foram divididas seguindo a mesma metodologia de conjunto de treino e teste (hold-out) nas respectivas configurações:
90/10, 80/20, 70/30.

Ao final então tinha-se as seguintes configurações:

- MNIST Treino Original 90/10, 80/20, 70/30
- MNIST Duas Classes 90/10, 80/20, 70/30
- MNIST Metade das amostras 90/10, 80/20, 70/30
- MNIST 20% das amostras 90/10, 80/20, 70/30

3 Metodologia dos Experimentos

3.1 Configuração Iniciais

Todos os experimentos foram feitos na plataforma online do Google Colab, usando a versão gratuita, e com o runtime padrão da plataforma, sem uso de aceleração de hardware por meio de GPU ou TPU.
Configurações do runtime:

- Processador: Intel Xeon 2.2GHz
- Memória: 13341992 kB
- Idle cut-off: 90 minutos
- Maximum Runtime: 12 horas

3.2 Configurações dos classificadores

Seguindo as recomendações propostas pelos orientadores, foram criados então três versões de cada modelos de aprendizado de máquina supervisionados, os modelos escolhidos foram: Arvore de decisão, K-Neighbors, Gaussian Naive Bayes e por fim uma Multilayer-Perceptron.

As seguintes configurações para cada um dos modelos foram adotadas:

- Árvore de decisão, variação do ccp-alpha: default(0.0), 0.1, 0.5
- K-Neighbors, variação do número de vizinhos: default(5), 10, 25
- Gaussian N-Bayes, variação do var-smoothing: default(1e-9), 1e-20, 1e-1
- MLP, variação das camadas escondidas: default(100;), (100,100), (24, 64)

3.3 Configurações Classificadores

Seguindo os resultados obtidos no trabalho de avaliação dos modelos supervisionados, foram então escolhidos para estratégias de classificação os mesmos modelos supervisionados(Árvores de decisão, K-Neighbors, Gaussian N-Bayes e MLP) e usados os parâmetros que obtiveram os melhores resultados.

As seguintes configurações para cada um dos modelos foram adotadas:

- Árvore de decisão, *ccp-alpha*: default(0.0).
- K-Neighbors, número de vizinhos: default(5).
- Gaussian N-Bayes, *var_smoothing*: default(1e-1).
- MLP, camadas escondidas: default(100;), função de ativação: relu, e *solver*: adam.

Além dessas configurações, foi adotada também como meio de comparação entre os modelos um contador de tempo em segundos na fase de treino de cada modelo.

Para o treino dos classificadores foram usadas apenas duas versões da base de dados original, foram usadas então as versões MNIST 20% e MNIST Duas Classes.

3.4 Comitês Classificadores

Foram usadas dois tipos de classificadores o BaggingClassifer e AdaBoostClassifier da biblioteca ensemble da Sklearn. Variando o número de classificadores entre 10, 15 e 20. Para os casos de abordagem homogêneas foi usada a StackingClassifier, também da mesma biblioteca, também variando o número de classificadores entre 10, 15, 20 e usando as melhores estratégias de classificação obtidas no experimento de Boosting e Bagging, no caso as Árvores de Decisão e a MLP. O método de stacking heterogêneo não foi abordado neste trabalho.

3.5 Resultados

Após a secção de treino todos as versões dos classificadores estudados nesse artigo foram submetidos a um fase de teste. Onde cada classificador foi exposto ao conjunto de testes e gerou um vetor contendo as classes preditas pelo modelo treinado. Sendo pode-se fazer uma comparação entre os valores preditos com os valores das classes do conjunto de treino. A métrica usada foi a acurácia, sendo a soma dos valores verdadeiros positivos com os verdadeiros negativos dividida por o somatório geral dos valores.

3.6 Teste de Friedman

Para o teste de Friedman foram adotadas as seguintes hipóteses. H_0, Hipótese nula, onde não há diferenças significantes entres os valores obtidos em cada classificador. H_1, Hipótese alternativa, onde considera-se que há diferenças significativas nos resultados dos classificadores. E por fim foi adotado o valor um alpha de 0.5.

Os seguintes testes foram feitos, comparação interna entre os classificadores, comparando então as diferentes configurações de cada classificador. Comparação externa dos classificadores, comparando os resultados então entre as melhores versões de cada classificador. Bagging e Boosting, comparação entre número de estimadores e classificador usado e por fim na técnica de empilhamento de classificadores Stacking a comparação entre os diferentes número de estimadores usando como classificadores a MLP e a Decision Tree.

3.7 Teste de Wilcoxon

Para o teste de Wilcoxon foram adotadas também as seguintes hipóteses. H_0, Hipótese nula, onde não há diferenças significantes entres os valores obtidos em cada classificador. H_1, Hipótese alternativa, onde considera-se que há diferenças significativas nos resultados dos classificadores. E por fim foi adotado o valor um alpha de 0.5.

Os seguintes testes foram feitos, comparação interna entre os classificadores. Testando então dois a dois cada configuração de cada classificador. Na comparação externa entre os classificadores também foi adotada a mesma estratégia de comparação em pares dos resultados obtidos. Para Bagging e Boosting a estratégia adotada das comparações foram também resultados pareados tanto dos números de estimadores quanto dos classificadores usados. E por fim na técnica de empilhamento de classificadores Stacking foram comparados em pares a quantidade de número de estimadores.

3.8 Problemas na etapa de treino

Durante o experimento alguns das estratégias de classificação tiveram alguns problemas com o comitê de Boosting. No caso os algoritmos foram o da MLP e o K-NN.

Durante o experimento o algoritmo KNN, apresentou instabilidades em algumas das versões da base de dados, apresentando resultado consistentes apenas para a versão MNIST Duas Classes e MNIST 20%.

O algoritmo de MLP teve um problema de fitting dos dados no comitê de classificador. Especificamente com o conjunto de dados MNIST 20%. No caso do KNN tive problemas com a implementação do AdaBoostClassifier do Sklearn, pois o mesmo não suporta por padrão o KNN como estimador por padrão.

4 Melhorias

Reformular totalmente a base de dados mudando de alguma forma a sua representação passando de um formato (*n_samples*, 28, 28, 1) para um formato vetorial, usado talvez uma função de flatten. Uma melhoria significativa também seria ao invés de usar uma RNN (Recurrent Neural Network) ao invés de uma MLP, nesse último caso não haveria a necessidade de pré-processar a mudança proposta anteriormente, visto que haveria já este processo nas camadas de convolução da rede

Para gerar os valores dos treinos que não foram possíveis, seria interessante verificar uma forma de aplicar o KNN no AdaBoostClassifier da sklearn, assim como foi feito no caso da MLP(originalmente também não é suportada no AdaBoostClassifier).

5 Resultados

Os valores abaixo de cada configuração de cada classificador está sempre expondo a acurácia e o tempo de treino.

5.1 Classificadores Individuais

5.1.1 Gaussian Naive Bayes

- *var_smothing* 1e-9: 67.74 e 0.35
- *var_smothing* 1e-20: 61.10 e 0.35
- *var_smothing* 1e-1: 85.54 e 0.34

5.1.2 Decision Tree

- *ccp_alpha* 0.0: 88.35 e 6.69
- *ccp_alpha* 0.1: 33.03 e 6.77
- *ccp_alpha* 0.5: 21.71 e 6.75

5.1.3 Knn

- *n_neighbors* 5: 97.95 e 21.39
- *n_neighbors* 10: 97.64 e 1.04
- *n_neighbors* 25: 96.80 e 1.03

5.1.4 MLP

- *hidden_layers* (100): 96.29 e 51.17
- *hidden_layers* (100,100): 95.47 e 52.30
- *hidden_layers* (24, 64): 94.21 e 79.06

5.2 Bagging

5.2.1 Gaussian Naive Bayes

Configuração do classificador com o *var_smothing* igual a 1e-9

- 10 estimadores: 79.33 e 3.12
- 15 estimadores: 79.33 e 4.64
- 20 estimadores: 79.33 e 6.24

5.2.2 Decision Tree

Configuração do classificador com o *ccp_alpha* igual a 0.0

- 10 estimadores: 95.86 e 10.15
- 15 estimadores: 96.25 e 15.35
- 20 estimadores: 96.45 e 20.73

5.2.3 Knn

Configuração do classificador com o *n_neighbors* igual a 5

- 10 estimadores: 97.96 e 447.33
- 15 estimadores: 97.91 e 671.53
- 20 estimadores: 97.95 e 894.085

5.2.4 MLP

Configuração do classificador com o *hidden_layers* igual a (100)

- 10 estimadores: 97.73 e 101.24
- 15 estimadores: 97.80 e 152.13
- 20 estimadores: 97.88 e 204.72

5.2.5 Melhor Configuração Bagging

- Decision Tree(*ccp_alpha*=0.0) e 10 estimadores.

5.3 Boosting

5.3.1 Gaussian Naive Bayes

Configuração do classificador com o *var_smothing* igual a 1e-9

- 10 estimadores: 74.23 e 6.28
- 15 estimadores: 69.87 e 9.48
- 20 estimadores: 65.56 e 12.58

5.3.2 Decision Tree

Configuração do classificador com o *ccp_alpha* igual a 0.0

- 10 estimadores: 91.28 e 1.91
- 15 estimadores: 91.28 e 1.90
- 20 estimadores: 91.28 e 1.92

5.3.3 Knn

Configuração do classificador com o *n_neighbors* igual a 5

- 10 estimadores: sem resultados
- 15 estimadores: sem resultados
- 20 estimadores: sem resultados

5.3.4 MLP

Configuração do classificador com o *hidden_layers* igual a (100)

- 10 estimadores: 46.73 e 69.62
- 15 estimadores: 46.73 e 105.93
- 20 estimadores: 46.73 e 135.96

5.3.5 Melhor Configuração Boosting

- Decision Tree(*ccp_alpha*=0.0) e 10 estimadores.

5.4 Stacking

Stacking Homogênio, com dois classificadores, Decision Tree(*ccp_alpha*=0.0) e MLP(*hidden_layers*=(100))

- 10 estimadores: 96.65 e 64.44
- 15 estimadores: 96.21 e 66.25
- 20 estimadores: 96.43 e 69.51

5.5 Analise estatísticas dos resultados

5.5.1 Teste de Friedman

5.5.1.2 Resultado Classificadores Individuais

- Gaussian Naive Bayes. Comparação dos diferentes *var_smoothing*: *pvalue* igual a 0.011
- Decision Tree. Comparação entre diferentes *ccp_alpha*: *pvalue* menor que 0.000
- Knn. Comparação entre números de vizinhos: 0.640
- MLP. Comparação entre diferentes estratégias de camadas escondidas: 0.006

5.5.1.3 Resultado Classificadores Externa

- *pvalue* menor igual a 0.000

5.5.1.4 Bagging

Resultado entre os Classificadores

- *pvalue* igual a 0.029

Resultado entre o número de estimadores

- *pvalue* igual a 0.174

5.5.1.5 Boosting

Resultado entre os Classificadores

- *pvalue* igual a 0.050

Resultado entre o número de estimadores

- *pvalue* igual a 0.368

5.5.1.6 Stacking

Comparação entre os diferentes números de estimadores

- *pvalue* igual a 0.289

5.5.2.1Teste Wilcoxon

5.5.2.2 Resultado Classificadores Individuais

Gaussian Naive Bayes

- *var_smothing* 1e-9, 1e-20 igual a 0.002
- *var_smothing* 1e-9, 1e-01 igual a 0.011
- *var_smothing* 1e-20, 1e-01 igual a 0.008

Decision Tree

- *ccp_alpha* 0.0, 0.1 igual a 0.002
- *ccp_alpha* 0.0, 0.5 igual a 0.001
- *ccp_alpha* 0.1, 0.5 igual a 0.043

Knn

- *n_neighbors* 5, 10 igual a 0.008
- *n_neighbors* 5, 25 igual a 0.005
- *n_neighbors* 10, 25, igual a 0.005

MLP

- *hidden_layers* (100), (100, 100) igual a 0.196
- *hidden_layers* (100), (24, 64) igual a 0.003
- *hidden_layers* (100, 100), (24, 64) igual a 0.011

5.5.2.3 Resultado Classificadores Externa

- MLP(*hidden_layers*=100), Knn(*n_neighbors*=5) igual a 0.003
- MLP(*hidden_layers*=100), Gaussian NB(*var_smoothing*=1e-1) igual a 0.008
- MLP(*hidden_layers*=100), Decision Tree(*cpp_alpha*=0.0) igual a 0.002
- Knn(*n_neighbors*=5), Gaussian NB(*var_smoothing*=1e-1) igual a 0.001
- Knn(*n_neighbors*=5), Decision Tree(*cpp_alpha*=0.0) igual a 0.002
- Decision Tree(*cpp_alpha*=0.0), Gaussian NB(*var_smoothing*=1e-1) igual a 0.245

5.5.2.4 Bagging

Resultado Classificadores

- Decision Tree(*cpp_alpha*=0.0), MLP(*hidden_layers*=100) igual a 0.109
- Decision Tree(*cpp_alpha*=0.0), Knn(*n_neighbors*=5) igual a 0.109
- Decision Tree(*cpp_alpha*=0.0), Gaussian NB(*var_smoothing*=1e-1) igual a 0.109
- MLP(*hidden_layers*=100), Knn(*n_neighbors*=5) igual a 0.109
- MLP(*hidden_layers*=100), Gaussian NB(*var_smoothing*=1e-1) igual a 0.109
- Knn(*n_neighbors*=5), Gaussian NB(*var_smoothing*=1e-1) igual a 0.109

Resultado entre número de estimadores

- *estimators*=10, *estimators*=15 igual a 0.465
- *estimators*=10, *estimators*=20 igual a 0.273

- *estimators*=15, *estimators*=20 igual a 0.068

5.5.2.5 Boosting

Resultado entre os Classificadores

- Decision Tree(*cpp_alpha*=0.0), MLP(*hidden_layers*=100) igual a 0.102
- Decision Tree(*cpp_alpha*=0.0), Gaussian NB(*var_smoothing*=1e-1) igual a 0.109
- MLP(*hidden_layers*=100), Gaussian NB(*var_smoothing*=1e-1) igual a 0.109

Resultado entre o número de estimadores

- *estimators*=10, *estimators*=15 igual a 0.655
- *estimators*=10, *estimators*=20 igual a 0.317
- *estimators*=15, *estimators*=20 igual a 0.180

5.5.2.6 Stacking

Resultado entre o número de estimadores

- *estimators*=10, *estimators*=15 igual a 0.091
- *estimators*=10, *estimators*=20 igual a 0.345
- *estimators*=15, *estimators*=20 igual a 0.080

5.6 Conclusões

Apenas duas configurações tiveram como resultado a Hipótese alternativa nos testes de Friedman e Wilcoxon, ou seja, o *pvalue* gerado foi um valor maior do que o alpha proposto.

Sendo as seguintes:

- Friedman, Knn, Comparação entre os números de vizinhos: p_value = 0.640
- Wilcoxon, Boosting, Comparação entre os números de estimadores: p_value=0.655

Seguindo a proposição então da hipótese alternativa, apenas esses dois classificadores possuem realmente uma diferença significativa nos valores dos resultados da acurácia.

Seguindo então a proposição da hipótese nula, todas as outras configurações das fases de treino não possuem uma diferença significativa entre os valores das acurácias.

Então seguindo os resultados obtidos nos testes de Wilcoxon e Friedman, posso concluir que o classificador com um tempo de execução menor para a fase de treino e com uma acurácia acima de 95% possa ser o fator determinante para a escolha do melhor classificador estudado.

Sendo assim podemos concluir que os seguintes classificadores tiveram uma performance melhor. Decision Tree com o alpha em 0.0, Bagging e Boosting com 10 comitês estimadores e Decision Tree com o alpha em 0.0 e Stacking Homogêneo com também 10 estimadores e os algoritmos de Decision Tree e MLP. Por fim ainda podemos concluir que o uso de comitês classificadores, Bagging, na maioria dos casos tem um impacto positivo no treinamento, tendo apenas como ponto negativo o aumento do tempo de execução. O mesmo podemos concluir para o uso de Stacking Homogêneo.

6 Referências

LeChun, Y., Cortes, C., Burges, C. J. C., (1998) “THE MNIST DATABASE of handwritten digits”, http://yann.lecun.com/exdb/mnist/, Dezembro, 2020.

Mello, C. F. C, (2020), “Código Fonte” https://github.com/carvalheirafc/mnist-deeplearning/blob/main/mnist_data_handler.ipynb, Dezembro, 2020.

Scikit-Learn Developers(2007-2020), “Ensemble SCIKIT-LEARN”, https://scikit-learn.org/stable/modules/ensemble.html, Dezembro, 2020.

Scikit-Learn Developers(2007-2020), “Naive Bayes SCIKIT-LEARN”, https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes, Dezembro, 2020.

Scikit-Learn Developers(2007-2020), “Decision Tree SCIKIT-LEARN”, https://scikit-learn.org/stable/modules/tree.html#tree, Dezembro, 2020.

Scikit-Learn Developers(2007-2020), “Nearest Neighbors SCIKIT-LEARN”, https://scikit-learn.org/stable/modules/neighbors.html#classification, Dezembro, 2020.

Scikit-Learn Developers(2007-2020), “Multi Layer Perceptron SCIKIT-LEARN”, https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron, Dezembro, 2020.

Scipy Developers, (2020) “Friedmanchisquare”, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.friedmanchisquare.html, Dezembro, 2020.

Scipy Developers, (2020) “Wilcoxon”, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html, Dezembro, 2020.