Machine Learning - Documentação do TP2

Lucas C. Lima[1]

[1]Universidade Federal de Minas Gerais (UFMG)

1cs.chv@gmail.com[1]

1. Introdução

O aprendizado de máquina é uma das áreas da ciência da computação que mais evoluiu nos últimos anos. Hoje, diversas técnicas de aprendizado de máquina vem sendo utilizados em diversas tarefas diárias como classificação de documentos, resultados de pesquisa na web, reconhecimento de imagens e reconhecimento de voz, entre outros. Uma das técnicas mais poderosas em aprendizado de máquina é o Boosting. Boosting consiste em utilizar diversos modelos simples, combinando-os de forma a produzir um modelo forte. Este trabalho consiste em implementar um classificador conhecido como Adaboost, utilizando como algoritmo de aprendizado as Decision Strumps.

2. Principais decisões

- A rede foi implementada utilizando a linguagem Python na versão 2.7.12.
- Foram criadas três classes Dataset, Adaboost, DecisionStumps que possuem os métodos e necessários para a execução do algoritmo
- Foi implementado o Adaboost como classificador.
- O algoritmo de aprendizado utilizado foi o Decision Stumps.
- Para realização dos experimentos foi utilizado validação cruzada com 5 folds *Cross Validation*.
- Foram utilizadas as bibliotecas Pandas, scikit-learn e numpy para auxiliar na implementação e avaliação.
- Para a avaliação, foram calculados erros simples a cada iteração.
- A função para atualização dos pesos utilizada foi $w_i^{t+1} = \frac{w_i^t}{z} \times \epsilon^{-\alpha \times h^t(X) \times y(X)}$
- A função α utilizada para atualização dos pesos foi $\alpha = 0.5 \times log(\frac{1-\epsilon^t}{\epsilon^t})$.
- Foi necessário a criação de dois métodos para classe Adaboost, os métodos *fit()* e *predict()*, desta forma foi possível utilizar o método de validação de score por *cross_val_score()* da biblioteca scikit-learn.
- Como se tratava de dados categóricos foi necessário a realização de OneHotEncoder, onde cada coluna corresponde a cada valor possível de uma feature.
- Como se trata de um problema de classificação binária, os labels das classes foram transformados de $negative, positive$ para $-1, 1$ respectivamente.

3. Descrição do classificador implementado

Nesta seção serão apresentados detalhes sobre a arquitetura da classificador, implementação do algoritmo de aprendizado decision stumps e como foram realizados os cálculos dos erros.

3.1. Data

O classificador a ser implementada deverá ser capaz de classificar jogos do tic-tac-toe utilizando como treinamento o tic-tac-toe_dataset. Tic-tac-toe dataset consiste em 957 jogos com seus respectivos resultados, onde cada linha é um jogo e cada coluna representa uma posição preenchida, ou seja, temos um vetor de 9 posições representando cada jogo e a última coluna representando o label positivo ou negativo.

3.2. Adaboost

Em nosso projeto utilizaremos as técnicas aprendidas durante as aulas. No processo de Adaboost [Freund and Schapire 1996] cada classificador tem um α associado a este que é a sua importância, tendo assim como resultado final da classificação a combinação dos resultados dos diversos classificadores

$$H^*(X) = \alpha^1 \times h^1(X) + \alpha^2 \times h^2(X) + \alpha^3 \times h^3(X) + ...$$

Para realizarmos a implementação do algoritmo de classificação Adaboost(adaptive boosting) devemos decidir três etapas principais:

1. Como atualizar os pesos?
2. Como calcular o α?
3. Como escolher o weak classifier?

Para a atualização dos pesos utilizamos

$$w_i^{t+1} = \frac{w_i^t}{z} \times \epsilon^{-\alpha \times h^t(X) \times y(X)}$$

onde z é um fator de normalização, que permite que os pesos continuem sendo uma distribuição. Vale ressaltar que w^t esta associado ao α^t e que este será calculado da seguinte maneira:

$$\alpha = 0.5 \times log(\frac{1 - \epsilon^t}{\epsilon^t})$$

onde ϵ^t é o erro empírico na iteração t. O cálculo do erro é feito através do algoritmo de aprendizado Decision Stumps, que consiste em uma decision tree de profundidade 1.

3.3. Weak learners - Decision Stumps

O algoritmo de weak learner utilizado foi o Decision stump que é um algoritmo de árvore de decisão com apenas um nó. Ele faz a sua predição baseado em apenas um atributo. O número de cortes realizados nesse algoritmo portanto é de $2 \times Numof Features + 2$, pois a cada corte na base (transformada em hotenconded) teremos dois novos cortes respectivo ao valor do atributo e ao inverso dele. Podemos realizar cortes também dos quais assumimos que todos valores preditos são positivos ou todos negativos, resultando assim em um total de 56 cortes. Este algoritmo tem como principal função retornar o corte que resulta no menor erro de predição.

3.4. Detalhes de implementação

O algoritmo implementado se divide em 3 classes:

1. Dataset: Realiza o tratamento do tic-tac-toe_dataset retornando uma matriz $X_{[958,27]}$, onde cada linha representa um jogo e cada coluna uma feature e um vetor $Y_{[1,958]}$ de labels dos valores esperados.

2. AdaBoost: Realiza o cálculo dos pesos de cada classificador e salva cada uma das funções aprendidas pelo algoritmo de decision stump de cada classificador H_x. Possui uma função que realiza a predição dos valores de acordo com as funções aprendidas. Atualiza os pesos de acordo com as funções aprendidas e predições de erros.
3. DecisionStumps: Realiza os cortes e identifica e retorna o corte com o menor erro de predição. Possui também uma função de predição, que quando chamada retorna o vetor de erros de predição.

4. Análise experimental

Com o intuito de avaliar a performance do classificador, será utilizado uma validação cruzada utilizando 5 folds. A avaliação consiste em variar o número de weak learners(decision stumps) utilizado a cada iteração. Medimos então a média dos scores retornados pela validação de cada fold alterando o número de decision stumps. Pode-se observar na figura abaixo que a medida que aumentamos o número de weak learners a acurácia aumenta e o erro diminui. No entanto, a partir de aproximadamente 1100 wearklearners não há ganho significativo na acurácia 97,3%, ou seja, o modelo converge. Seria necessário então buscar uma estratégia mais complexa, ou testar para alguns outros diferentes weak learners.

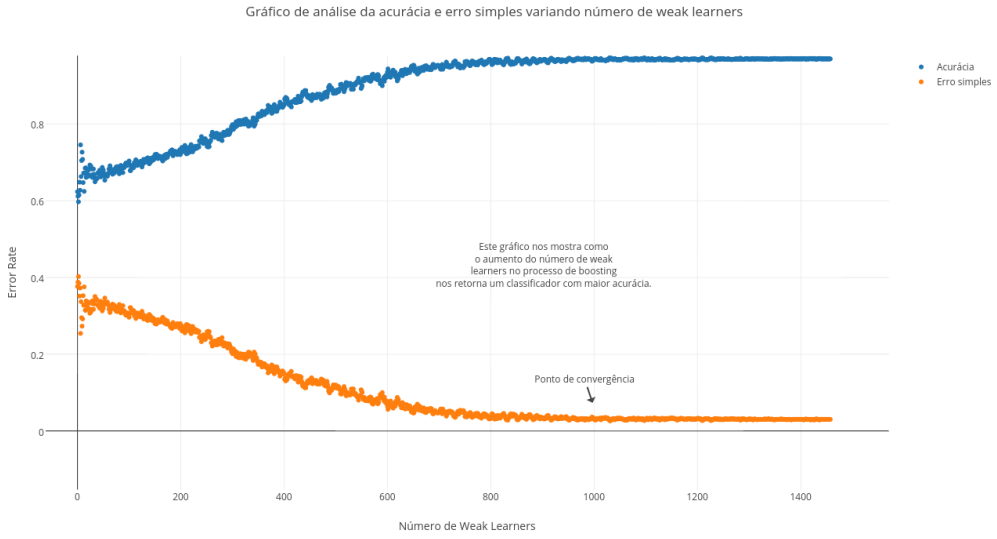

Figura 1. Comparação da qualidade do classificador variando o número de weak learners

5. Considerações finais

Com este trabalho foi possível observar como a implementação da estratégia de boosting retorna uma acurácia alta para classificação. Verificamos que a partir de aproximadamente 400 iterações o modelo já possui uma acurácia muito alta o que é relativamente bom. Por fim, podemos concluir que este trabalho foi de imenso aprendizado colocando a frente não apenas o estudo mas a implementação de estratégias de Machine Learning.

Referências

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

Winston, P. (2010). Learning: Boosting. `https://www.youtube.com/watch?v=UHBmv7qCey4&t=2771s`. [Online; accessed 16-June-2017].