Boosting

Victor B. R. Jorge

28 de Junho de 2017

1 Introdução

Em aprendizado de máquina, Boosting consiste no processo de agrupar vários classificadores fracos de forma a formar um modelo robusto. Boosting faz parte das técnicas chamadas de *ensemble*, palavra que pode ser traduzida como "conjunto de itens visto como um todo". Basicamente, no Boosting vários classificadores bastante simples e, portanto, com baixa variância e alto viés, são agrupados em um sistema de "voto ponderado" de maneira a reduzir o viés do modelo final. Algoritmos baseados em Boosting são, hoje, estado da arte em termos de classificação de dados estruturados.

Este trabalho tem como objetivo implementar um algoritmo clássico de Boosting, conhecido como AdaBoost (Adaptive Boosting). O AdaBoost foi um dos primeiros algoritmos de Boosting conhecidos. Além disso, o desempenho do algorimto será avaliado em um ambiente de classificação binária com dados categóricos. A base de dados tic-tac-toe, que apresenta dados de partidas do jogo da velha, será utilizada.

2 Descrição e implementação

2.1 Base de Dados

A base de dados tic-tac-toe codifica o conjunto completo de possíveis configurações de tabuleiro no final dos jogos da velha, assumindo-se que o jogador que utiliza o símbolo "x" tenha jogado primeiro. O conceito de classificação é "vitória de x", ou seja, verdadeiro quando "x" tem uma das 8 formas possíveis de ganhar o jogo e falso caso contrário.

Portanto, cada coluna da base de dados representa uma posição do tabuleiro que pode estar marcada com "x", "o" ou "b" que representa uma posição vazia. Sendo assim, são 9 atributos com 3 possíveis valores.

2.2 Classificadores fracos

Foram utilizadas *decision stumps* como nosso classificadores fracos. *Decision stumps* nada mais são que árvores de decisão de profundidade um. Portanto, consistem de classificadores extremamente simples, o que é ideal para o processo de Boosting.

Teremos, no total, 54 decision stumps possíveis. Duas para cada um dos 3 atributos para cada uma das 9 posições do tabuleiro.

2.3 AdaBoost

No AdaBoost um conjunto de pesos é associado ao conjunto de treinamento. Portanto, cada entrada do treino possui um peso associado a ela. Para reduzir o viés de classificação, a cada iteração do algoritmo o conjunto de pesos é atualizado de maneira a tentar aumentar os pesos daqueles entradas que foram classificadas de maneira equivocada pelo classificador utilizado na iteração anterior. A ideia por trás dessa abordagem é fazer com que o próximo

classificador se preocupe em acertar as entradas que o classificador anterior foi incapaz de classificar corretamente.

A implementação do AdaBoost proposta neste trabalho é bastante simples. Primeiramente, os pesos do conjunto de treinamento é inicializado de maneira a todas as instâncias teremos o mesmo peso e $\sum w_i = 1$. Portanto, para cada entrada $w = \frac{1}{m}$, onde m é a quantidade de exemplos do conjunto de treinamento.

Para cada iteração do algoritmo, testamos os 54 stumps no conjunto de treinamento e com os pesos atuais. O stump que apresentar o menor erro de classificação ϵ_t é selecionado. Como tratamos de atributos categóricos e classificação binária, o erro de cada stump é a soma dos pesos das instâncias classificadas equivocadamente. O stump selecionado será nosso classificador fraco para aquela iteração. Precisamos, agora, calcular o peso do "voto" daquele classificador no modelo final. Esse peso, conhecido como α no AdaBoost, pode ser calculado pela seguinte fórmula: $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$. Calculado o alfa, adicionamos, então, o classificador ao nosso modelo.

Precisamos, agora, atualizar os pesos do nosso conjunto de treinamento de modo a priorizar as instâncias classificadas de maneira equivocada pelo nosso classificador. No AdaBoost, os pesos são atualizados pela seguinte fórmula: $w_{t+1} = w_t * e^{-\alpha_t y h_t(x)}$, onde y é o vetor que representa as classes corretas para as instâncias do treino e $h_t(x)$ é a previsão feita pelo stump selecionado. O novo conjunto de pesos é então normalizado para respeitar a regra $\sum w_i = 1$ e seguimos para a próxima iteração do algoritmo.

No final do nosso algoritmo, teremos n *decision stumps* e cada uma terá um α associado. A modelo final terá o seguinte formato: $h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + ... + \alpha_n h_n(x)$. Ainda não falamos nada sobre o nosso número de iterações. Isso ocorreu justamente porque n é o único parâmetro do nosso algoritmo. Ele define quantos classificadores fracos nosso modelo possui e deve ser obtido empiricamente.

3 Resultados e Análises

Foram realizados, basicamente, dois experimentos. No primeiro, foi calculado o erro médio a cada iteração do algoritmo, durante o processo de treinamento. Além disso, foi calculado o erro médio dos stumps selecionados a cada iteração. Esse experimento tem por objetivo observar a convergência do erro de treinamento do modelo, além de analisar o processo de seleção dos stumps. Para todos os experimentos foi utilizado um protocolo de validação cruzada com 5 conjuntos. Portanto, os valores representam a média obtida após a validação cruzada. Os resultados do primeiro experimento pode ser observado no gráfico abaixo:

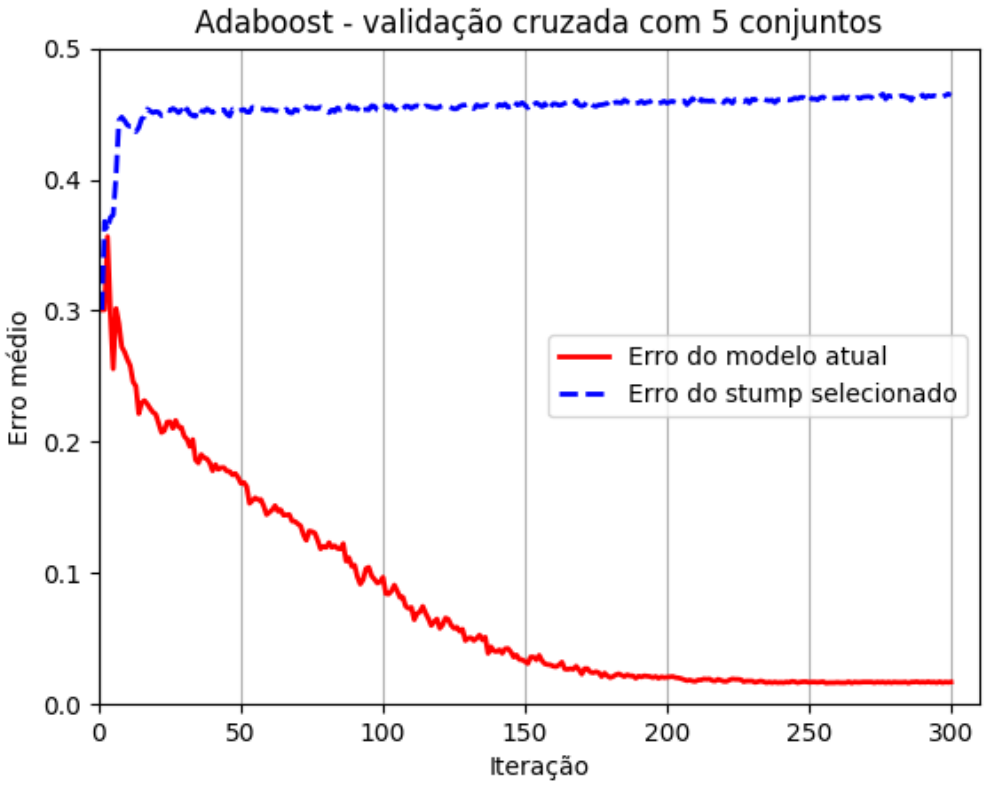

Figure 1: Erro médio a cada iteração do algoritmo durante o processo de treinamento

Podemos observar claramente, pela linha azul, a atuação do AdaBoost. Nosso conjunto de dados possui um *decision stump* capaz de prever a classe correta com apenas 30% de erro, em média. Caso as entradas classificadas equivocadamente não tivessem seu peso aumentado, selecionaríamos sempre o mesmo classificador fraco e nosso modelo ficaria enviesado. Porém, com o passar das iterações, a acurácia desse modelo fraco em específico é prejudicada pela atualização dos pesos e outros modelos são selecionados. Como podemos observar, a maioria dos classificadores fracos possuem acurácia um pouco melhor que 50%, o que é ideal para o processo de Boosting.

Além disso, podemos observar que a medida que mais classificadores fracos são adicionados ao nosso modelo, menor seu erro de treinamento. Essa relação começa a estagnar a partir da 200ª iteração. Isso é uma característica intrínseca aos dados, portanto difere de um cojunto de dados para outro. Neste trabalho, é possível notar que 200 iterações são o suficiente para se minimizar o erro de treinamento.

No segundo experimento, foi utilizada a validação cruzada em 5 conjuntos para se avaliar a performance do modelo para diferentes n (número de iterações ou quantidade de modelos fracos). Esse experimento tem como objetivo avaliar a acurácia do classificador para diferentes tamanhos de n, já que o erro de treinamento não é uma boa medida para avaliar a robustez do modelo. Lembrando que os valores refletem a média entre as execuções realizadas na validação cruzada. Os resultados podem ser observados no gráfico abaixo:

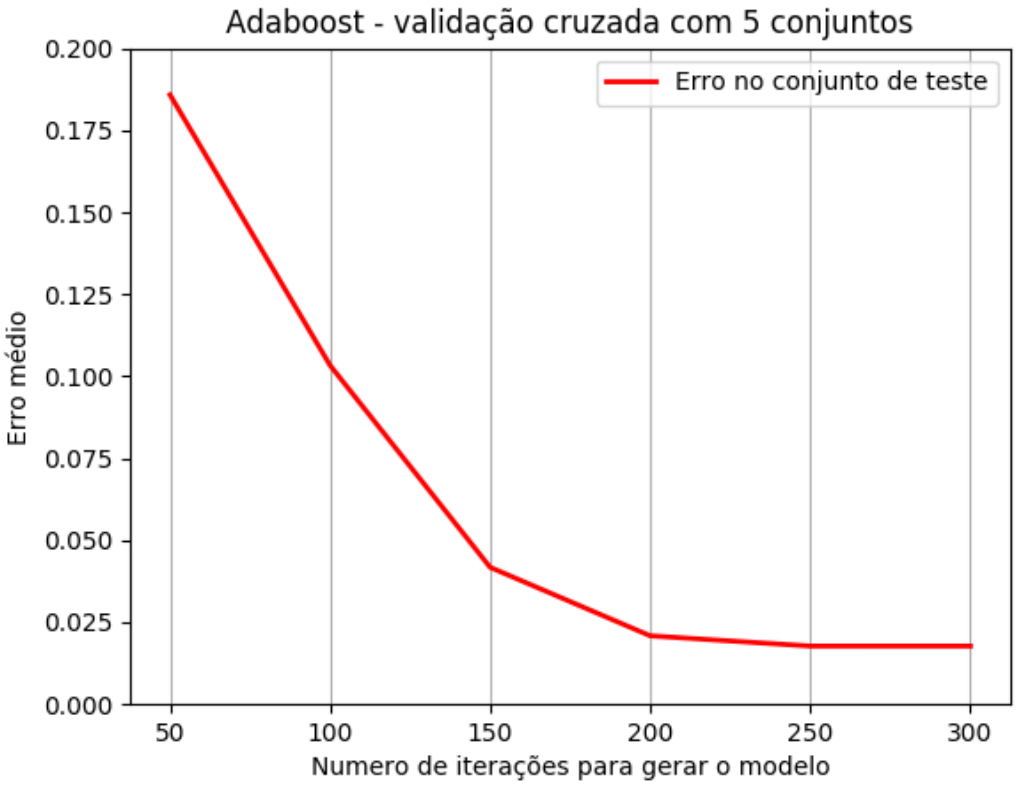

Figure 2: Erro médio para quantidade de iterações

O gráfico acima mostra a convergência do erro empírico. Como podemos observar, 50 iterações já são suficientes para obtermos um modelo com um erro abaixo dos 20%. Além disso, o erro cai quase que quadraticamente e começa a convergir a partir da 250ª iteração, quando atingimos o valor de 2,5% de erro. Um classificador com 97,5% de acurácia é um classificador bastante robusto e isso mostra o poder do Boosting e mais especificamente do AdaBoost. Como sabemos, Boosting é estado da arte em termos de classificação de dados estruturados.

4 Conclusão

Boosting é um meta-algoritmo de aprendizado de máquina que busca unir vários modelos fracos em um modelo robusto. É estado da arte para a classificação de dados estruturados e possui diversas implementações. AdaBoost é uma das pioneiras e mais eficientes implementações dessa técnica. Além de ser bastante robusta é computacionalmente muito mais barata que a abordagem por redes neuronais, por exemplo.

No processo de Boosting é muito importante definir quais serão seus modelos fracos e qual será a técnica utilizada para garantir que os modelos fracos não se sobreponham. Além disso, a quantidade de modelos utilizadas para se formar seu modelo final é de extrema importância para a convergência do erro e depende diretamente do conjunto de dados a ser classificado.

Por fim, este trabalho foi bastante prazeroso de ser implementado porque podemos gerar, na prática, modelos que são realmente robustos e conseguem classificar os dados com alta acurácia. Diferentemente do último trabalho, onde as redes necessitam de treinamento exaustivo, o AdaBoost pode ser treinado em pouco tempo e consegue bons resultados.