

Boosting

Victor Freguglia Souza

RA: 137784

and

Leonardo Uchoa Pedreira

RA: 156231

25 de Novembro de 2018

Resumo

The text of your abstract. 200 or fewer words.

Keywords: Quando for a versão final., Tirar esse campo na .tex;

1 Introdução

Boosting é o nome dado a um tipo de algoritmo que, assim como outros métodos em Machine Learning como Bagging e Florestas Aleatórias, busca combinar um grande número de preditores com baixo poder de predição (isto é, um pouco mais eficientes do que a escolha ao acaso) para compor um bom preditor. O conceito é fundamentado nas ideias apresentadas em Kearns (1988) e Schapire (1990).

Diferentemente dos outros métodos citados, onde os preditores fracos que serão combinados são criados de maneira independente (e aleatória devido ao processo de bootstrap), no método de Boosting os preditores fracos são criados de maneira a melhorar o desempenho em regiões com altas taxas de erro. Isto é, se pensarmos que cada preditor tem um “voto” na decisão final, o método nos fornece um comitê em que aqueles que tem grande convicção têm mais poder na decisão. Estes de grande convicção, sabem muito sobre uma parte do espaço amostral (não sei se amostral é a melhor palavra).

O algoritmo AdaBoost (de *Adaptive Boosting*), apresentado pela primeira vez em Freund & Schapire (1997), é um dos exemplos mais clássicos de algoritmo de boosting para o problema de classificação binária. Nele, a cada passo m , um novo classificador G_m é ajustado com base em uma versão ponderada do conjunto de dados original, na qual o peso de cada observação depende do desempenho do classificador anterior: pontos classificados de maneira errada recebem peso maior, e assim, têm uma chance maior de serem corrigidos pelos classificadores ajustados na próxima iteração. O Algoritmo 1 apresenta a descrição completa do AdaBoost. Note que os classificadores G_m a serem usados não precisam ser de nenhum tipo específico e, portanto, se comporta como um parâmetro a ser escolhido por um processo de regulação (tuning) do problema. Apesar disso, o mais comum, pela grande flexibilidade, é a árvore de classificação e regressão (CART).

Embora o AdaBoost seja um bom ponto de início para entender o conceito de Boosting, ele foi projetado para resolver problemas de classificação binária, e por isso, não apresenta resultados tão bons quando diretamente adaptado a problemas de classificação múltipla e regressão, então diferentes algoritmos são necessários para esses casos. Hastie et al. (2009) propõe uma modificação do AdaBoost para o caso de classificação múltipla, por exemplo.

A principal diferença entre os diferentes algoritmos de Boosting propostos está na ma-

início

$(y_i, x_i), i = 1, \dots, N; y_i \in \{-1, 1\}; x_i \in \mathbb{R}^p;$

Inicie todos pesos $w_i = 1/N;$

para $m = 1, \dots, M$ **faça**

1. Ajuste um classificador $G_m(x);$

2. Calcular

$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i};$$

3. Calcular $\alpha_m = \log\left(\frac{1-\text{err}_m}{\text{err}_m}\right);$

4. Atualizar $w_i \leftarrow w_i \exp(\alpha_m I(y_i \neq G_m(x_i))), i = 1, \dots, N;$

fim

Defina o classificador final como

$$G(x) = \text{sign}\left(\sum_{m=1}^M \alpha_m I(G_m(x) = k)\right).$$

fim

Algoritmo 1: Algoritmo AdaBoost apresentado em Friedman et al. (2001). Aqui, as classes do problema de classificação são representadas pelos valores -1 e 1.

neira com que os a ponderação dos dados é feita em cada passo. Nesse trabalho, por apresentar uma forma mais geral, será considerado o algoritmo Gradient Boosting, que pode lidar com todos os tipos de problemas de predição com a devida escolha da função perda a ser minimizada. A teoria que o fundamenta e sua descrição detalhada são apresentadas na Seção 2. O método é então aplicado ao conjunto de dados MNIST, descrito na Seção 3 e os resultados são mostrados na Seção 4.

2 Metodologia

Em problemas de regressão, escolher o melhor preditor significa estimar uma função f^* que minimiza o risco, definido como o valor esperado da função de perda $L(\cdot, \cdot)$, ou seja,

$$f^* = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{Y|x} [L(Y, f(x)) | x]. \quad (1)$$

Note que o risco para um preditor específico f é desconhecido, uma vez que não assumimos nenhuma distribuição, o que impossibilita o cálculo do valor esperado da função perda para esse preditor. Uma boa estratégia para criação da função de predição $f \in \mathcal{F}$ consiste em escolher preditores de forma a minimizar o risco empírico.

$$f^* = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^N L(y_i, f(x_i)) \quad (2)$$

sujeito à que f seja árvore

Uma forma de resolver o problema 2 é via utilização de algoritmos numéricos. Gradiente descendente (citar watkins), por exemplo, é uma estratégia habitualmente utilizada, que levaria ao algoritmo

$$f^{(i+1)} = f^{(i)} + \sum_{i=1}^N \nabla_f L(y_i, f(x_i)) \quad (3)$$

Entretanto (Friedman (2001)) cita que simplesmente utilizar isto teria efeitos catastróficos no modelo preditivo. Primeiro porque só são levados em consideração as observações que temos (i.e, não existe nenhuma intenção de extrapolar poder preditivo, como por exemplo a separação entre teste e treinamento ou validação cruzada faria), segundo que não é levado em consideração à relação entre as covariáveis. Para contornar este problema, o autor sugere utilizar árvores de decisão de uma forma muito inteligente.

Desta forma Boosting busca resolver o problema de otimização 1 sujeito à restrição de que f^* sejam árvores de decisão (i.e, \mathcal{F} é o conjunto de todas as possíveis árvores). Formalmente, uma árvore de decisão pode ser escrita como

2.1 Gradient Boosting

- Citar o xgboost, caret.

início

$(y_i, x_i), i = 1, \dots, N;$

Inicie com o preditor constante $H_0 = \arg \min_c \sum_{i=1}^N L(y_i, c);$

para $m = 1, \dots, M$ **faça**

1. Calcular

$$r_{im} = - \left[\frac{\partial L(y_i, G(x_i))}{\partial} \right]_{G=H_{m-1}} ;$$

2. Ajustar um novo preditor fraco h_m ao conjunto de dados $(r_{im}, x_i);$

3. ;

4. ;

fim

Defina o classificador final como

$$G(x) = H_M(x).$$

fim

Algoritmo 2: Algoritmo Gradient Boosting apresentado em Friedman (2001).

2.2 PCA Whitening

Talvez isso não seja necessário pois habitualmente as arvores nao precisam desse tipo de tratamento. Svm e redes habitualmente precisam pq usam demais combinacoes lineares.

Mas essa secao pode ser util pra fazer comparacao com os outros metodos em que isso serve e melhorar a secao de discussao.

3 Dados

Conjunto de dados MNIST.

- Mostrar alguns números

4 Aplicação

- Ajuste

- Comparar com Random Forest, nnet, pca-whitened

5 Discussão

- Não sei o que entra em discussão/conclusão?

Proposta:

Discutir (pq em aplicação não há discussão crítica) os resultados da comparação do gbm com os métodos anteriores em questão de velocidade, precisão, flexibilidade. Considerar as variantes xgboost, lightgbm

Discutir efeito do pca no gbm e em outros métodos que precisem disso, como redes e svm.

Usar outros aspectos considerados no arquivo “kuhn_cheatsheet” pra comparar?

6 Conclusão

Fazer por último e ver o que aprendemos do trabalho.

Referências

- Freund, Y. & Schapire, R. E. (1997), ‘A decision-theoretic generalization of on-line learning and an application to boosting’, *Journal of computer and system sciences* **55**(1), 119–139.
- Friedman, J. H. (2001), ‘Greedy function approximation: a gradient boosting machine’, *Annals of statistics* pp. 1189–1232.
- Friedman, J., Hastie, T. & Tibshirani, R. (2001), *The elements of statistical learning*, Vol. 1, Springer series in statistics New York, NY, USA:.
- Hastie, T., Rosset, S., Zhu, J. & Zou, H. (2009), ‘Multi-class adaboost’, *Statistics and its Interface* **2**(3), 349–360.
- Kearns, M. (1988), ‘Thoughts on hypothesis boosting’, *Unpublished manuscript* **45**, 105.
- Schapire, R. E. (1990), ‘The strenght of weak learnability’, *Machine Learning* **5**, 197–227.