

# BOOSTING

Gabriel Jonas

Natalia Sá

Roberto Moure

- 
- O que é boosting
  - Loss Function
  - Weak Learning
  - Modelo Aditivo
  - Boosting
    - AdaBoost
    - Gradient Boosting
    - KGBost
  - Considerações
  - Experimentação

# Boosting

---

- O Boosting, assim como o Bagging, é um *framework* que é utilizado para implementar *Ensembles*;
- Ele também pode ser definido como um meta-algoritmo;
- *Ensemble* é um conjunto de preditores que separados tem pouco poder de predição, quando combinados tem alto poder de predição.

# Boosting

---

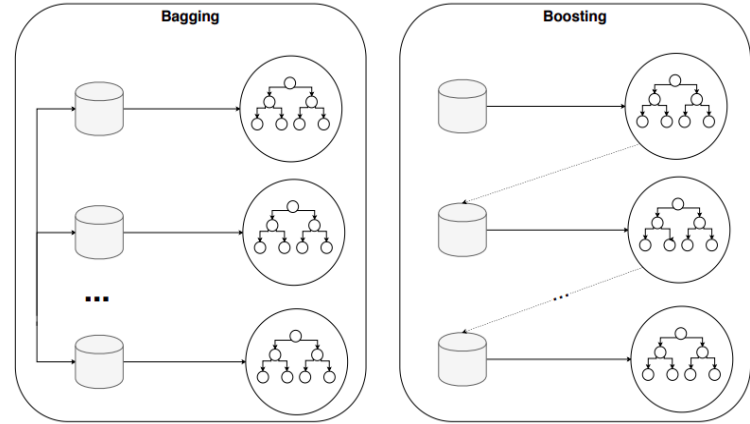
- O boosting é baseado na hipótese de que se um preditor focar nos exemplos errados de um preditor anterior, ele terá sucesso no aprendizado.

*"The idea is to use the weak learning method several times to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified. ... Note, however, it is not obvious at all how this can be done"<sup>1</sup>*

<sup>1</sup> Arney, C., 2014. Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World. *Mathematics and Computer Education*, 48(1), p.126.

# Boosting

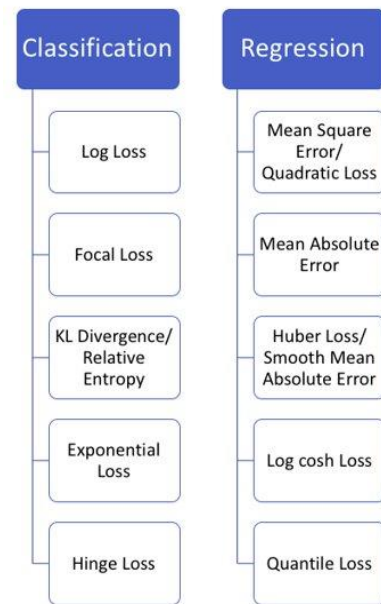
- Uma *loss function* a ser otimizada.
- Um *weak learner* para realizar previsões.
- Dessa forma, os preditores são adicionados de forma sequencial para minimizar o erro do preditor anterior.



# Loss function

---

- Classificação: usada para predição de classes discretas;
- Regressão: usada para predição de valores contínuos;



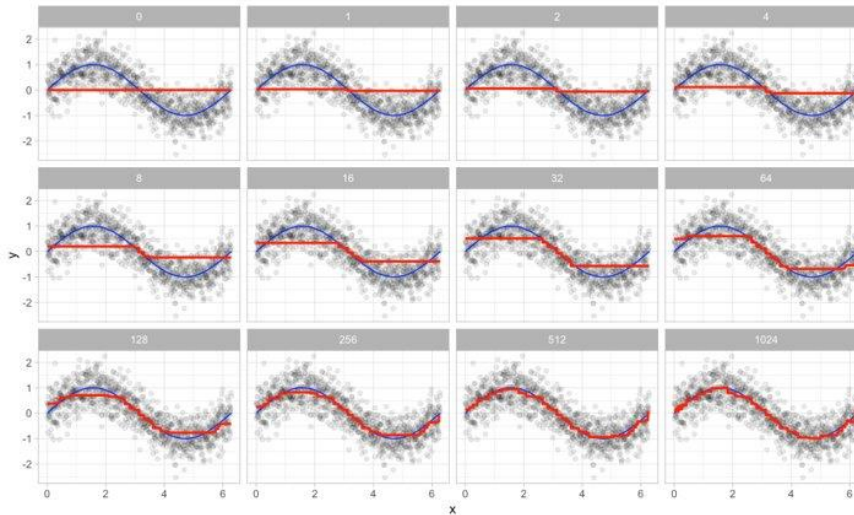
# Weak Learner

---

- São classificadores simples, leves, que não tem tanto poder preditivo de forma individual;
- A complexidade de implementação e computacional são baixas.
- Alguns exemplos de preditores que podem ser utilizados:
  - Árvores de decisão
  - Perceptron

# Modelo aditivo

- Os preditores são adicionados de forma sequencial.
- O preditor  $p_t$  será treinado utilizando o erro e amostras classificadas de forma errada do preditor  $p_{t-1}$





# Boosting

---

- Se o preditor sempre foca no erro do anterior, como evitar o *overfitting*?
- Qual o valor de erro o próximo preditor irá prever?
  - O valor absoluto do erro?
  - Gradiente negativo?
- Os preditores são treinados de forma sequencial, isso não pode levar a problemas de desempenho?
- Isso é resolvido apenas quando implementamos alguma técnica de ML que utiliza *boosting*.

# Boosting

---

- AdaBoost
- Gradient Boosting Machine (GBM)
- eXtreme Gradient Boosting (XGBoost)
- LightGBM
- CatBoost
- *OLBoosting*
- *OzaBoosting*
- ...

# Boosting

---

- **AdaBoost**
- **Gradient Boosting Machine**
- **eXtreme Gradient Boosting (XGBoost)**
- **LightGBM**
- **CatBoost**
- *OLBoosting*
- *OzaBoosting*
- ...

# AdaBoost

---

- Utiliza árvores de decisão como preditores
  - Cada árvore tem no máximo profundidade 1 (*Decision Stumps*)
  - As folhas da arvores são binárias.
- Cada amostra tem um peso no treinamento
  - A cada preditor novo adicionado, o peso das amostras que foram classificadas de forma errada são elevados.
- Cada preditor também tem um peso associado.

# AdaBoost

---

1. Inicializa o peso das amostras de forma igual
2. Seleciona uma amostra do conjunto de treinamento e constrói um modelo utilizando o preditor base
3. Avalia esse preditor no conjunto de testes e calcula a métrica de erro, baseado no pesos das amostras.
4. Calcula o peso desse preditor.
5. Atualiza o peso das amostras.

# AdaBoost

---

- Para fazer a predição de novas amostras é utilizada a seguinte equação:

$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)$$

- Onde,  $\alpha_t$  é o peso do preditor e  $h_t$  é o preditor base.

# Gradient Boosting

---

- É uma evolução do Boosting
- Usualmente utiliza árvores como preditor base
- Utiliza o Gradiente Descendente para construir os modelos subsequentes
  - Dessa forma, a função de perda deve ser diferenciável.
- A cada novo preditor adicionado ao *ensemble*, ele tenta prever o erro residual do anterior
  - O erro residual é baseado no gradiente negativo do preditor anterior.
- Algumas implementações de Gradient Boosting utilizam algumas técnicas de regularização e penalização para evitar *overfitting*
  - Número mínimo de amostras em uma folha.
  - Penalização por complexidade nas árvores.

# Gradient Boosting

---

1. Constrói um preditor base nos dados de treinamento
2. Avalia o erro residual desse preditor utilizando a função de perda
3. Constrói o próximo preditor utilizando o  $y$  como o erro residual anterior.



# XGBoost

---

- O XGBoost é uma biblioteca que implementa o GBM, foi implementada por Tianqi Chen.
- Implementa 3 tipos de Gradient Boosting
  - Gradient Boosting
  - Stochastic Gradient Boosting
  - Regularized Gradient Boosting
- Utiliza de diversas *features* para melhorar o desempenho preditivo e computacional
- É um dos algoritmos com mais vitórias em competições de ML

# XGBoost

---

- É possível paralelizar o treinamento
- Utilizar computação distribuída
- Computação *Out-of-core*
- Otimização de Cache

# XGBoost

---


- O algoritmo é *Sparse Aware*
  - Consegue lidar com dados faltantes.
- Utiliza uma estrutura de bloco nas árvores
  - Permitindo assim a paralelização na construção das árvores
- Treinamento contínuo

# Considerações

---

- Ao utilizar o Boosting podemos atingir um poder preditivo muito alto com preditores simples.
- Muito sensível a implementação e hiper-parâmetros.
  - O desempenho computacional e preditivo do boosting está muito correlacionado com os hiper-parâmetros utilizados, técnicas de regularização, etc.

# EXPERIMENTAÇÃO



Obrigado pela  
atenção