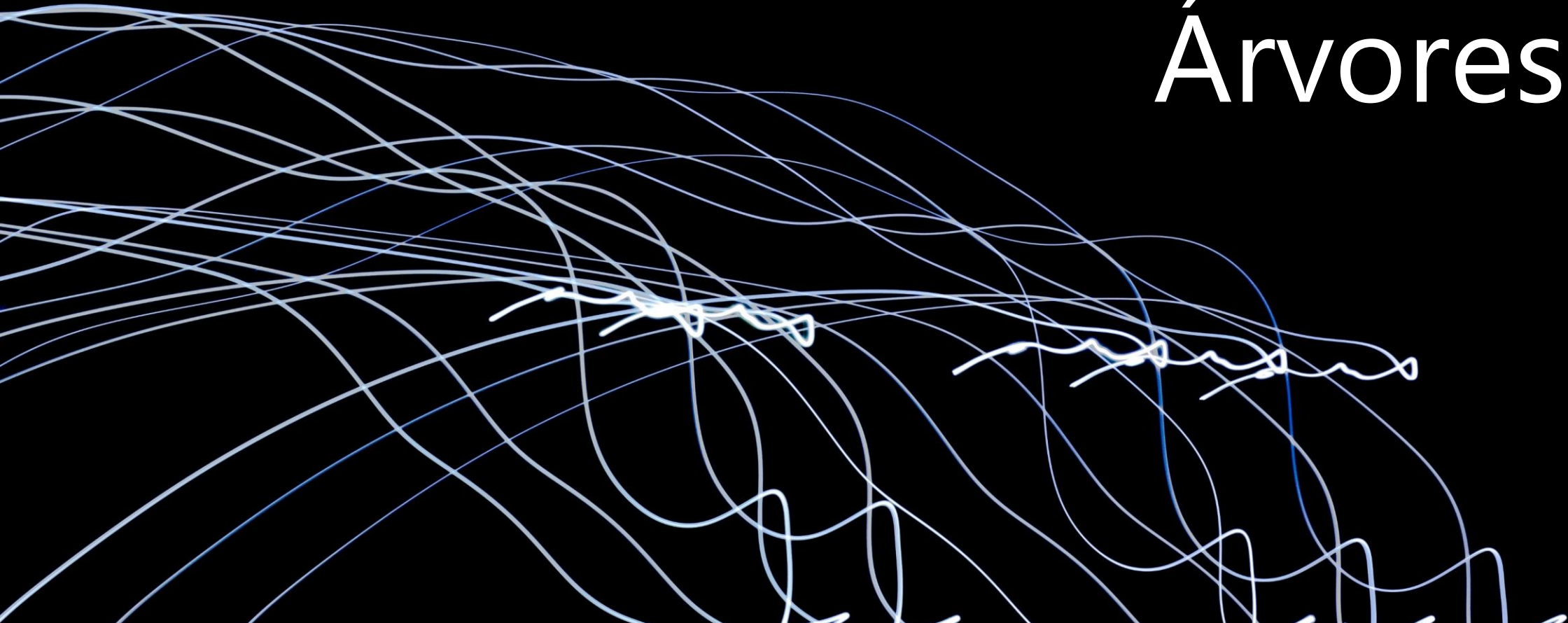


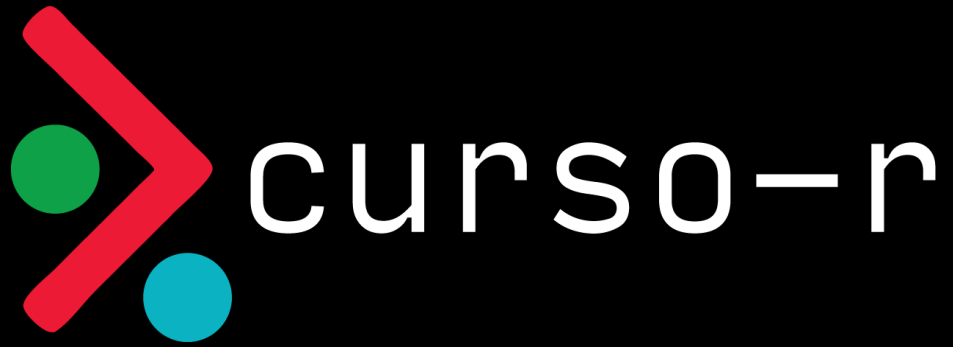
Modelos de Árvores





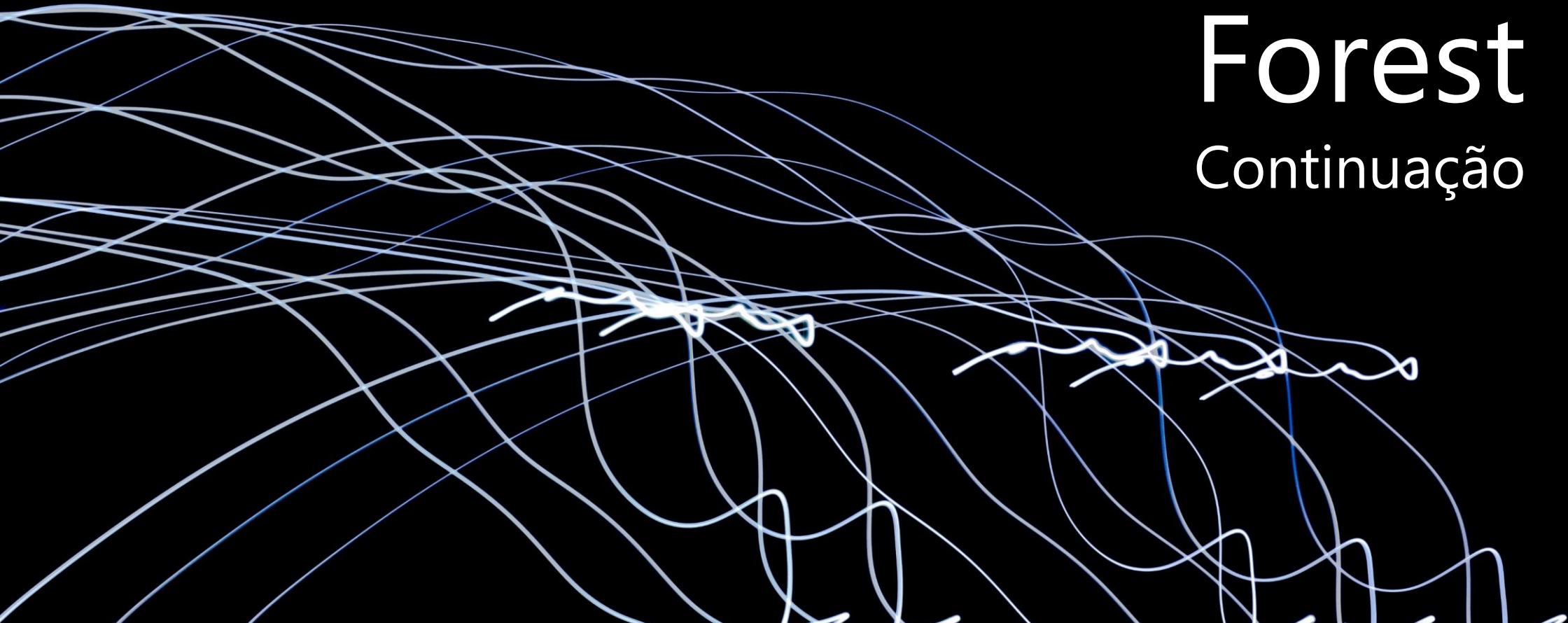
Agenda

- Aplicação de Random Forest no R
- Gráfico de Dependência Parcial (PDP)
- Introdução ao Boosting
- AdaBoost – breve apresentação
- Gradient Boosting
- XGBoost
- Titanic





Random Forest

Continuação



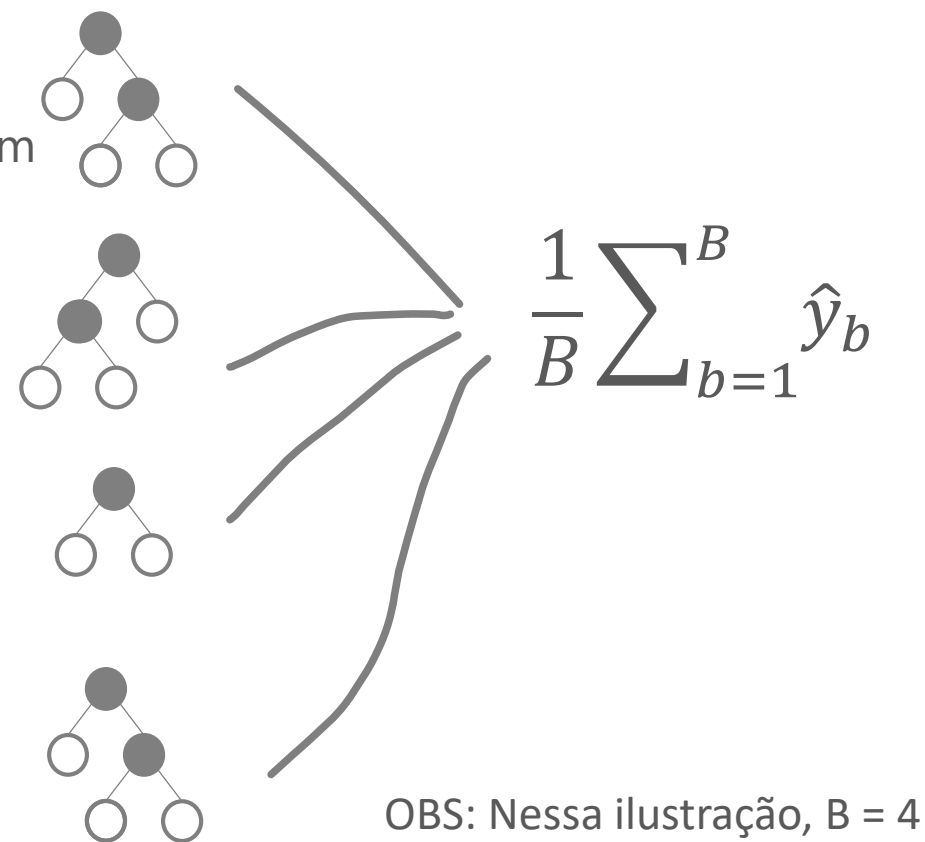


Random Forest

- Random Forest é a combinação de “palpites” de um monte de árvores de decisão.
- É um algoritmo de uma classe especial de ENSEMBLE: BAGGING.
- ENSEMBLE: mistura de 2 ou mais modelos.  ESL p 605
- BAGGING: Bootstrap AGGregation  ESL p 282
- Diferencial para os BAGGINs tradicionais: Sorteia as colunas também

Algoritmo:

- 1) Sorteie B conjuntos de observações da base D
- 2) Para cada conjunto b de B, sorteie m variáveis de D
- 3) Para cada uma das B sub-bases geradas por (b, m) construa uma árvore de decisão
- 4) Para previsão final, agregue as previsões individuais de cada uma das B árvores.





Hiperparâmetros e Overfitting

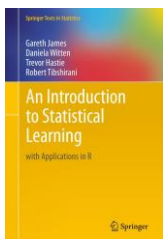
Hiperparâmetros do pacote `randomForest` do R

ntree – Número de árvores (amostras bootstrap) para treinar. Não afeta muito o overfitting.

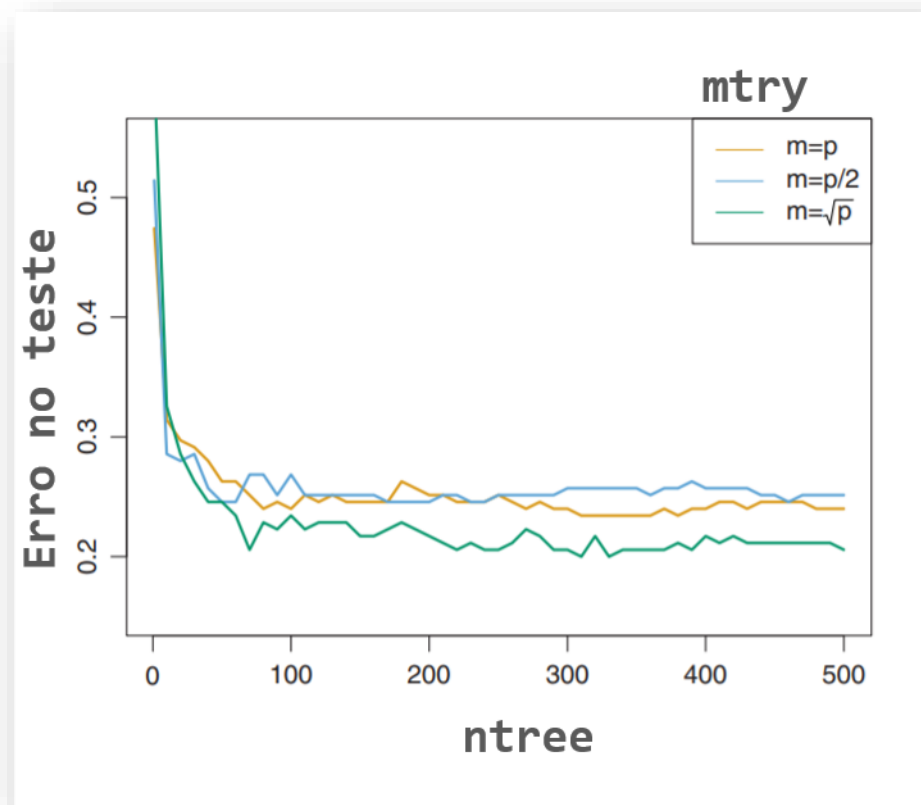
mtry – Quantidade de variáveis (colunas) sorteadas por árvore. Tem que testar via cross-validation, pois é afetado pela razão entre #variáveis boas e #ruído.

nodesize – Análogo ao **minsplit** do `rpart`. Quantidade mínima de observações no nó para poder dividir.

OBS: random forest não usa CP. Ele permite que as árvores cresçam indeterminadamente, condicionadas apenas pelo **nodesize**.



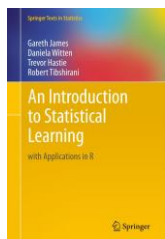
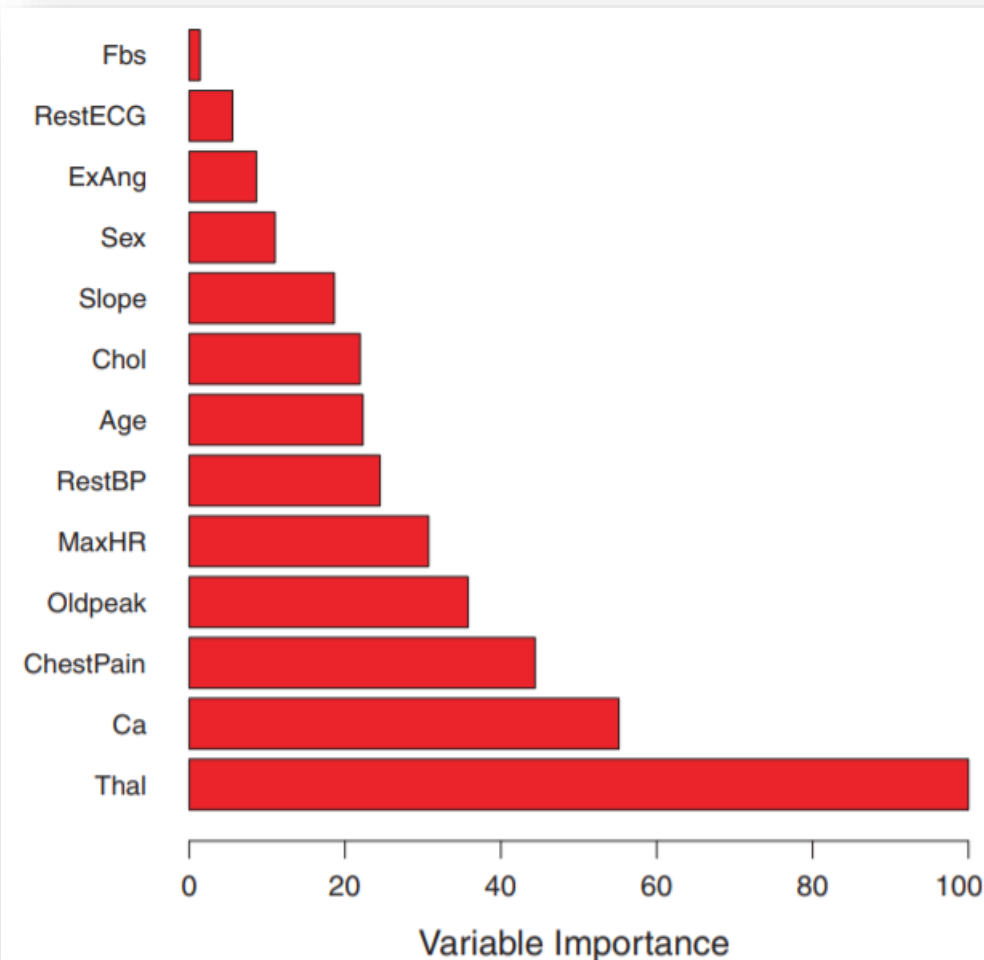
ISL
p 322





Importância das Variáveis (*variable importance*)

- A importância de uma certa variável X é calculada (na maioria das vezes) pela média dos ganhos de informação das quebras feitas por aquela variável.
- O gráfico ao lado mostra uma escala de 0 a 100. É a maneira como se costuma apresentar a importância da variável uma vez que a média do ganho de informação é difícil de interpretar.
- No R: `varImp(modelo)`



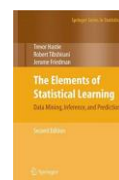
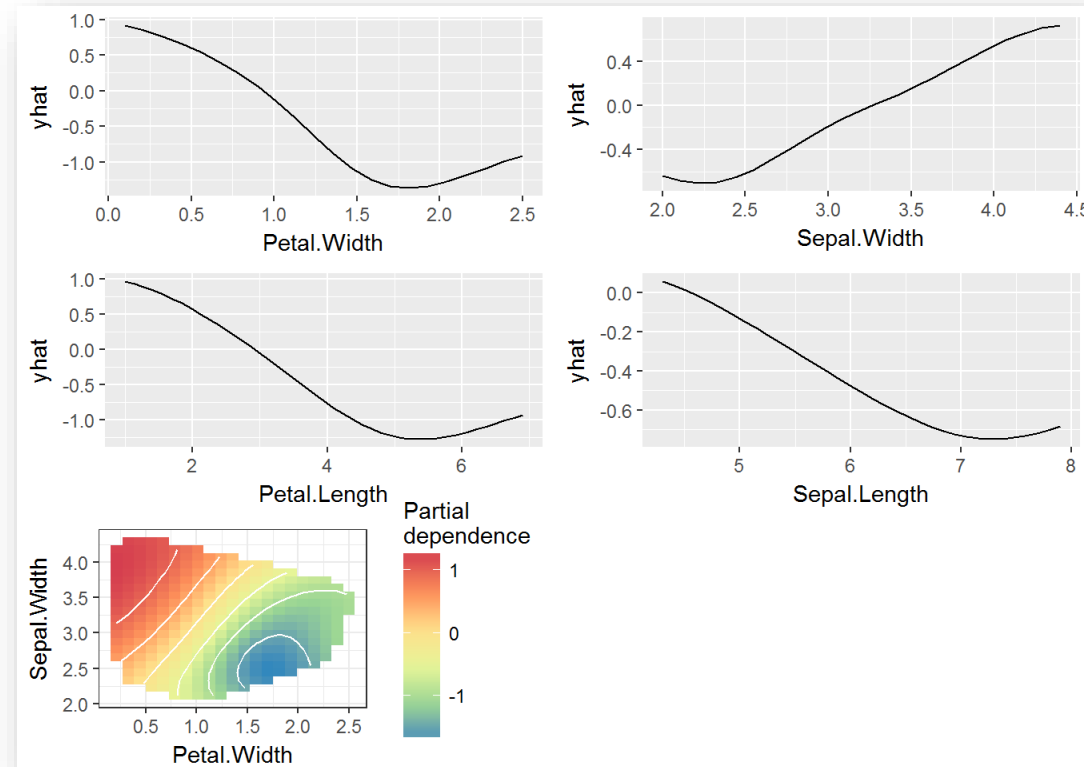
ISL
p 319



Gráfico de Dependência Parcial (*partial dependence plot*)

pdp – Partial Dependence Plot: Serve para mostrar o efeito (marginal) de uma variável explicativa na estimativa do modelo.

- É possível fazer o efeito conjunto de duas ou mais variáveis.
- **O motivo da sua existência:** Random Forest, Boosting, Redes Neurais, SVM e tantos outros modelos são difíceis de serem interpretados diretamente pelos seus parâmetros.
- **Receita:** para cada observação da sua base, crie N linhas a mais alterando os valores de uma variável enquanto mantém as demais características fixas. Então, calcule as respectivas estimativas.
- No R: pacotes **pdp** ou **plotmo**

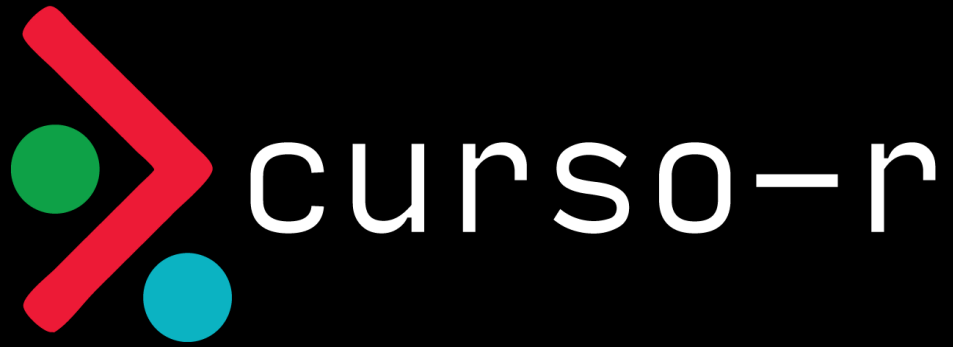


ESL
p 369

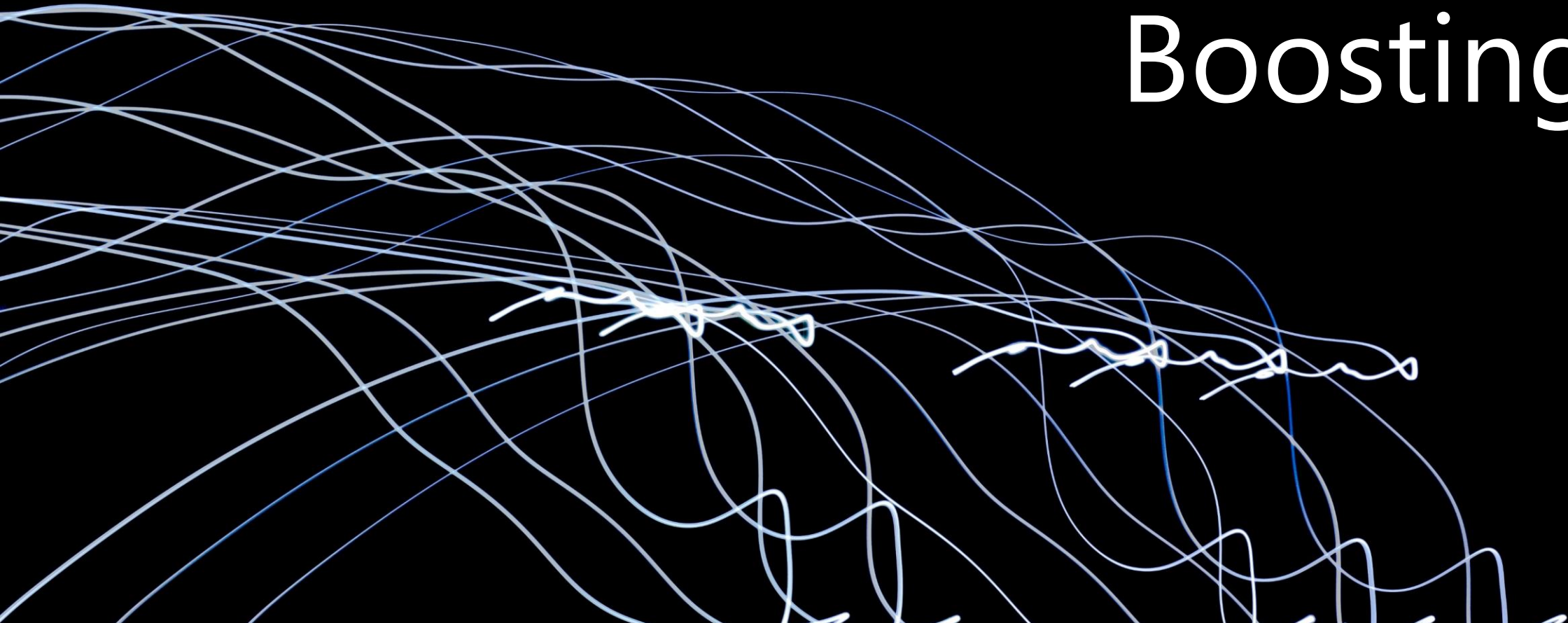


```
library(randomForest)
```

Ao R...



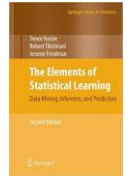
Gradient Boosting





Boosting

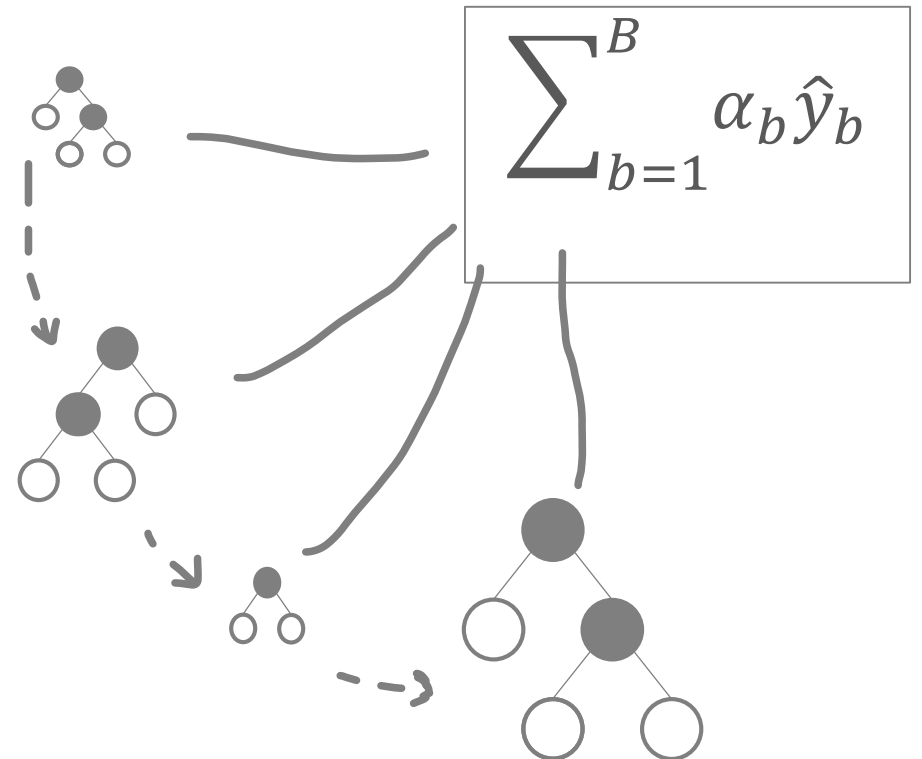
- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



ESL
p 341

Modelo proposto: Expansão de bases de funções

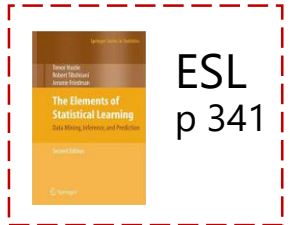
$$f(x) = \sum_{m=1}^M \beta_m b(x_i; \gamma_m)$$





Boosting

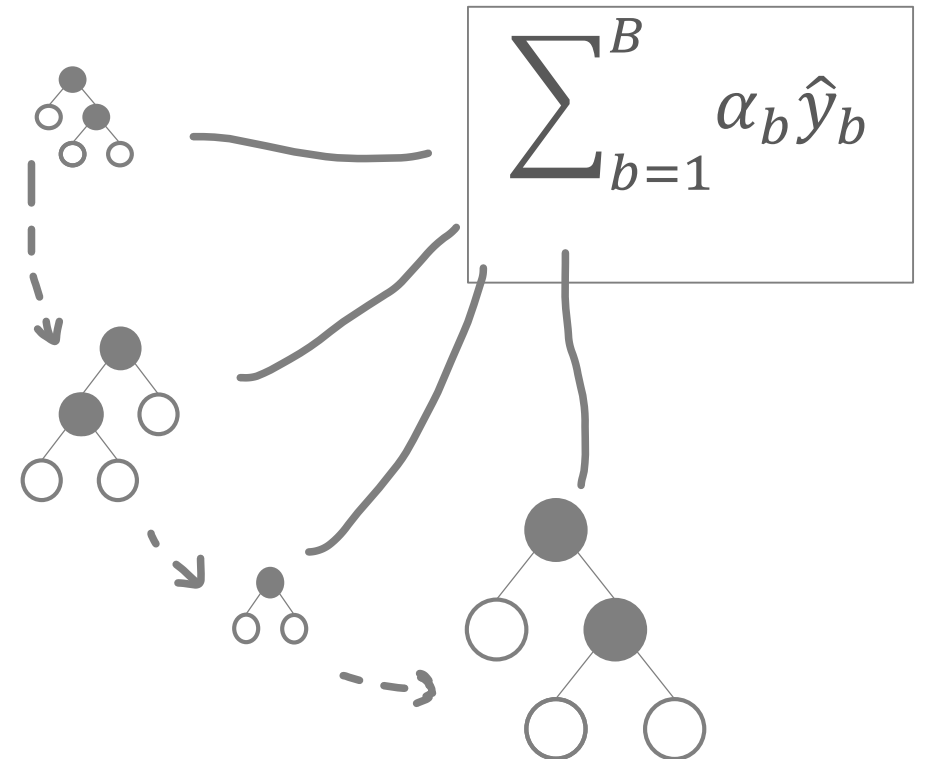
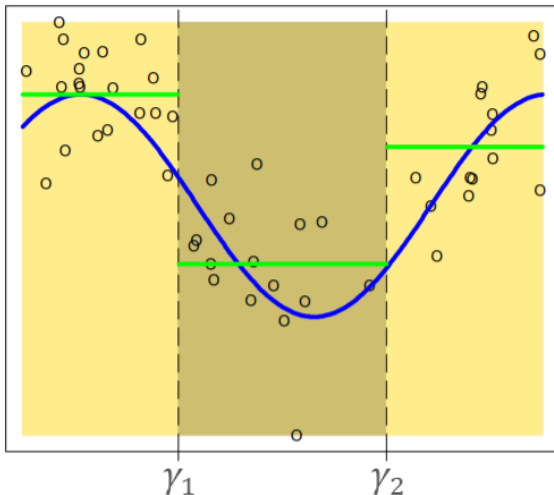
- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



Modelo proposto: Expansão de bases de funções

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$

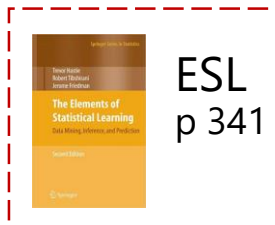
EXEMPLO: $b(x, \gamma_1) = I(x < \gamma_1)$, $b(x, \gamma_2) = I(x < \gamma_2)$, $b(x, \gamma_3) = I(x < \gamma_3)$





Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



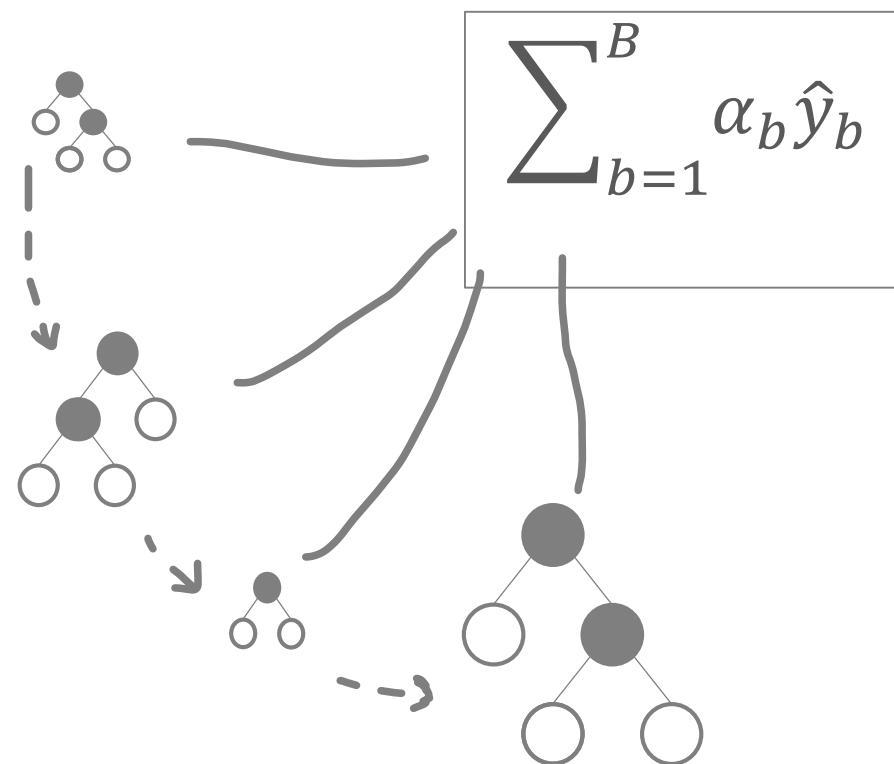
Modelo proposto: Expansão de bases de funções

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$

EXEMPLO: $b(x, \gamma_1) = I(x < \gamma_1)$, $b(x, \gamma_2) = I(x < \gamma_2)$, $b(x, \gamma_3) = I(x < \gamma_3)$

PROBLEMAS:

- No exemplo γ_1, γ_2 e γ_3 foram fixados. Gostaríamos de encontrar conjuntamente (β_m, γ_m) que melhor se ajustasse, mas é virtualmente impossível na maioria das vezes.





Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.

Forward Stage-wise Modeling (coração do Boosting):

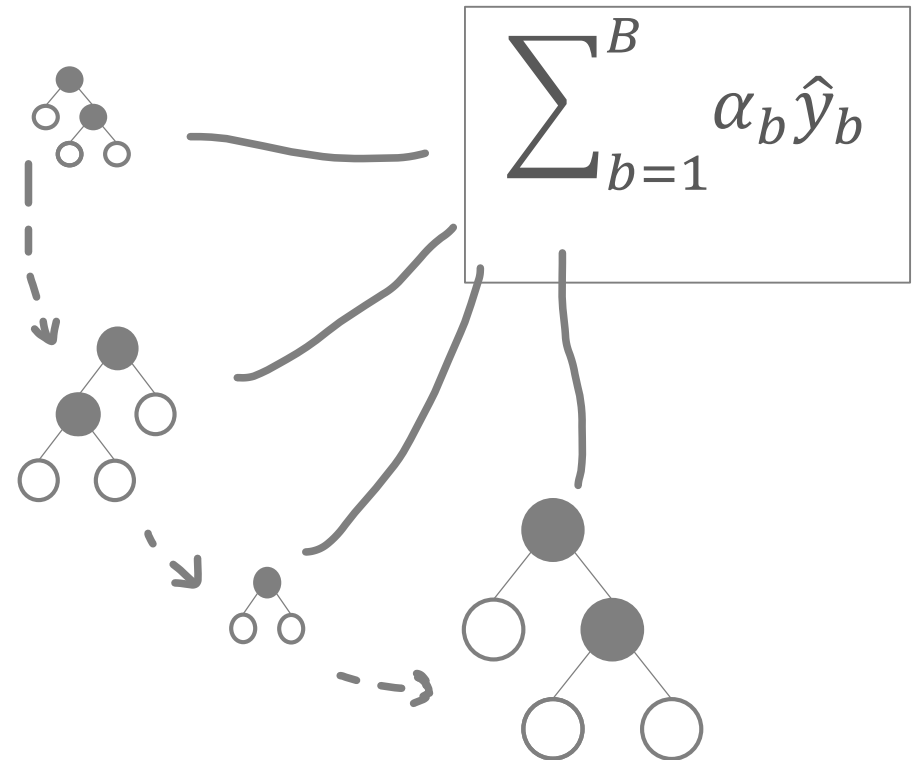
1) Initialize $f_0(x) = 0$

2) Para $m = 1$ a M :

a) Calcule:

$$(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

b) Atribua $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$





Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.

Forward Stage-wise Modeling (coração do Boosting):

1) Inicialize $f_0(x) = 0$

2) Para $m = 1$ a M :

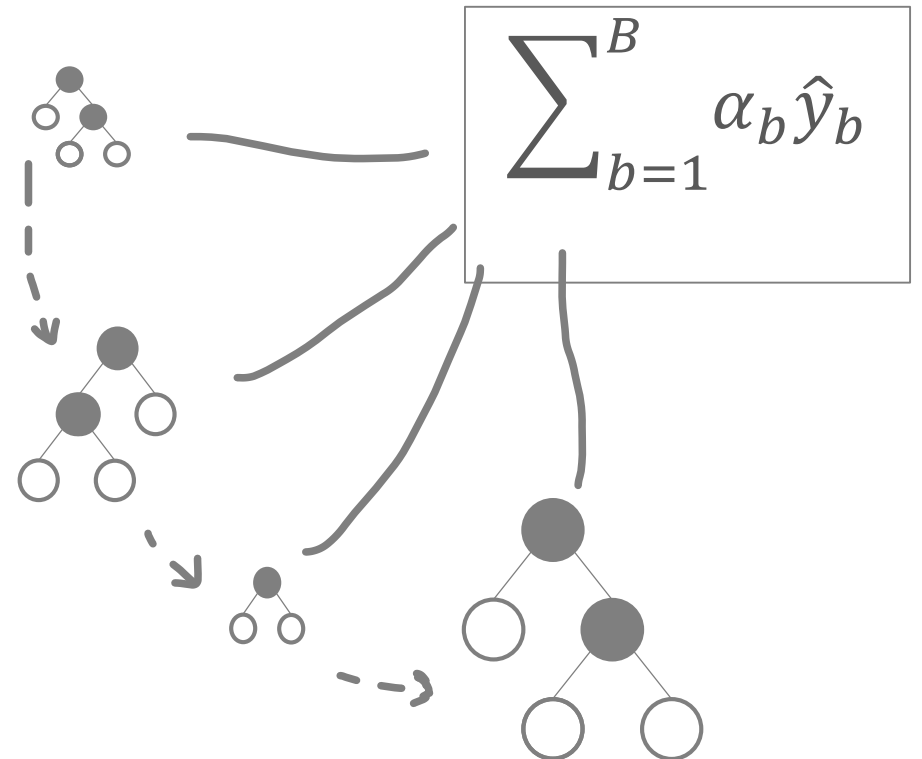
a) Calcule:

$$(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

b) Atribua $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

EXEMPLO DE LOSS FUNCTION: $L(y_i, f(x)) = (y_i - f(x))^2$

$$\begin{aligned} L(y_i, f(x)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2 \\ &= (r_i - \beta b(x_i; \gamma))^2 \end{aligned}$$





Boosting

- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.

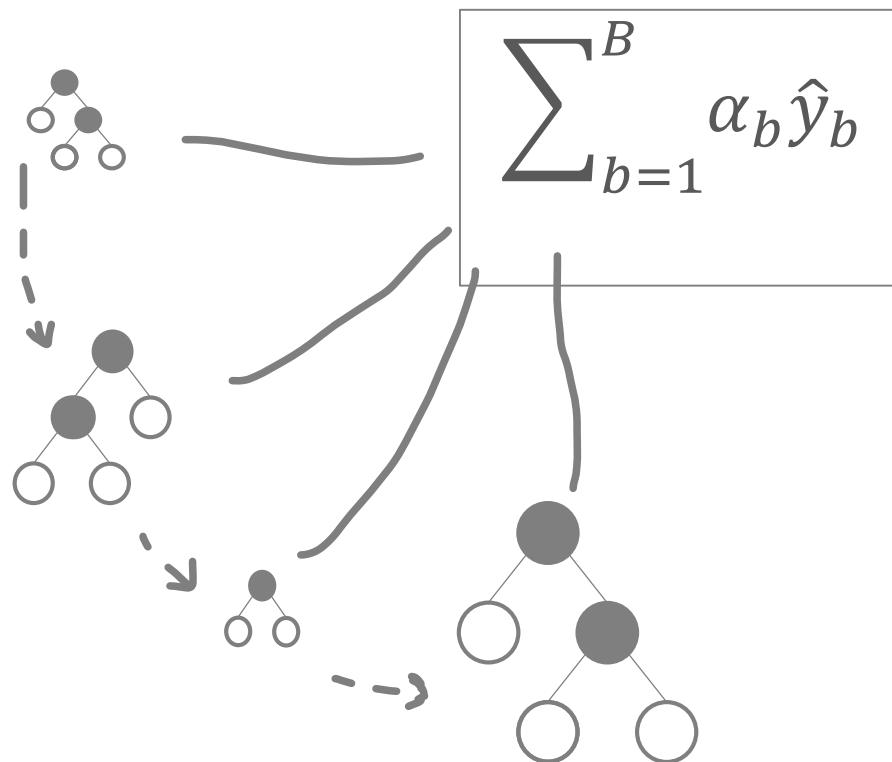
AdaBoost (versão de classificação binária):

- 1) LOSS utilizada: $L(y, f(x)) = \exp(-y * f(x))$. Em que $Y \in \{-1, 1\}$.
- 2) As funções $b(x, \gamma)$ agora serão árvores $T_m(x)$ que irão retornar ou 1 ou -1.
- 3) Temos que minimizar, então (para ser adicionado em cada passo m):

$$(\beta_m, T_m) = \arg \min_{\beta, T} \sum_{i=1}^N \exp(-y_i (f_{m-1}(x_i) + \beta T(x_i)))$$

- 4) Solução: $T_m = \arg \min_T \sum_{i=1}^N w_i^{(m)} I(y_i \neq T(x_i))$ $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$

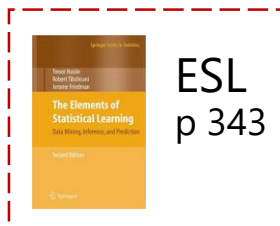
$$\beta_m = \frac{1}{2} \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right)$$





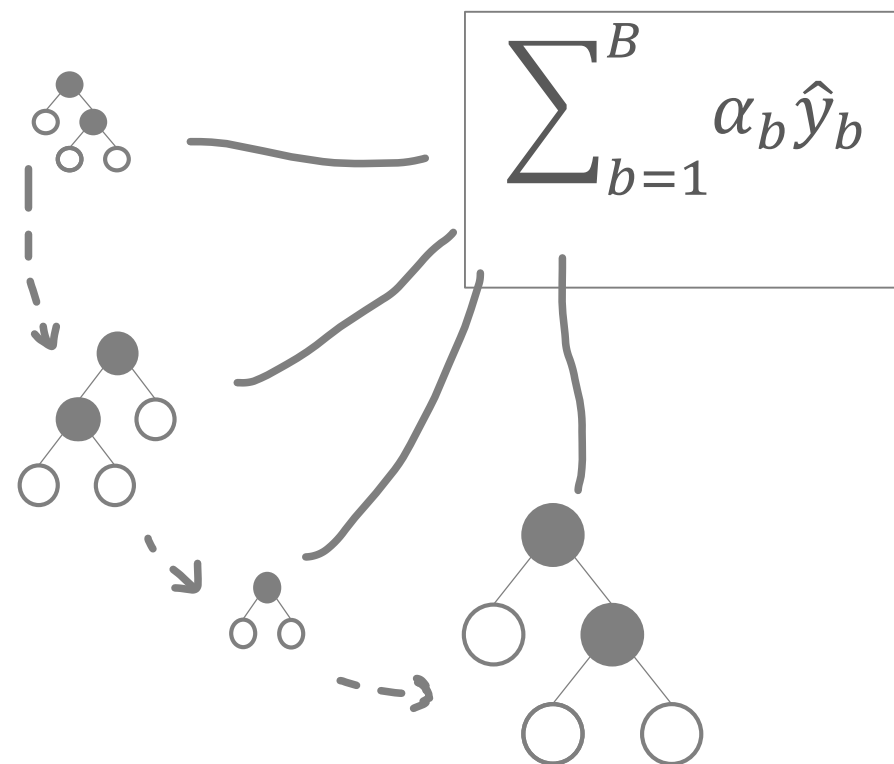
Boosting

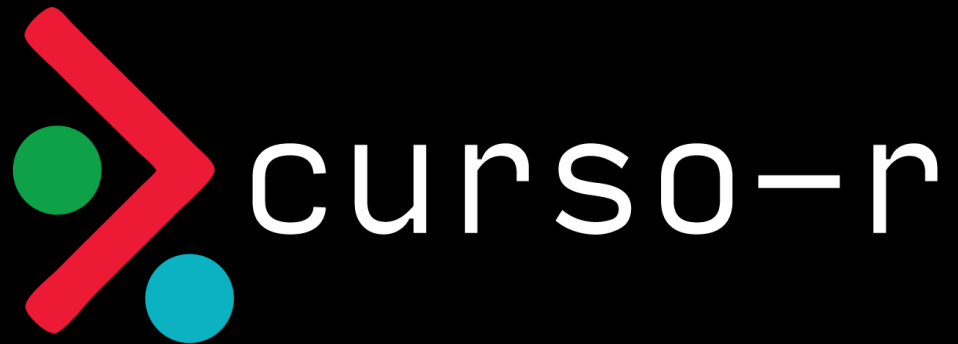
- Boosting também é a combinação de “palpites” de um monte de árvores de decisão.
- Porém, não existe amostras bootstrap dentro do algoritmo.
- As árvores são construídas **sequencialmente**. Cada árvore é construída usando informação da árvore passada.



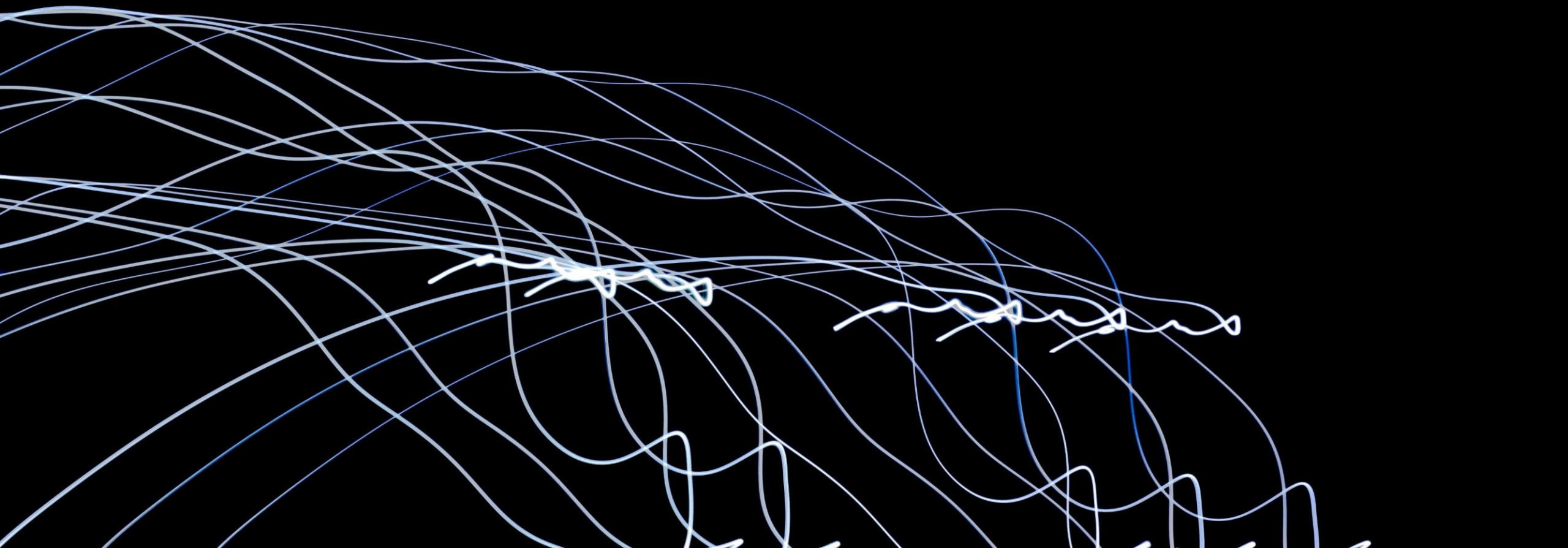
Algoritmo AdaBoost (versão de classificação binária):

- 1) Codifique $Y \in \{-1, 1\}$.
- 2) Inicializa-se pesos $w_i = \frac{1}{N}, i = 1, 2, \dots, N$
- 3) Para $m = 1$ a M :
 - a) Ajuste uma árvore $T_m(x)$ usando os pesos w_i .
 - b) Calcule o erro de m : $err_m = \frac{\sum_{i=1}^N w_i I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i}$
 - c) Compute $\alpha_m = \log\left(\frac{1-err_m}{err_m}\right)$
 - d) Atualize os pesos: $w_i \leftarrow w_i \cdot \exp\left(\alpha_m \cdot I(y_i \neq T_m(x_i))\right), i = 1, \dots, N$
- 4) Previsão: $sign[\sum_{m=1}^M \alpha_m T(x)]$





XGBoost





XGBoost

- XGBoost é uma implementação eficiente do Gradient Boost.
- Ele também é uma sofisticação. O XGBoost traz de volta um monte de hiperparâmetros de regularização.
- Top 2 ou Top 3 no Ranking de Algoritmos que mais ganharam Kaggle.
- No R: `library(xgboost)`

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^n [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T\end{aligned}$$

Hiperparâmtros do pacote `xgboost` do R

nrounds – Número de árvores.

max_depth – Profundidade máxima da árvore.

eta – Tamanho do passo em busca do mínimo da função de custo. Quanto menor, mais devagar. Aconselha-se aumentar o número de árvores junto!

gamma – Parâmetro regularizador. Análogo ao CP do `rpart`.

colsample_bytree – Qtd de variáveis sorteadas por árvore.

subsample – Quantidade de observações por árvore.

min_child_weight – Observações mínimas nas folhas.

lambda - Regularizador L2. Controla o tamanho dos parâmetros das folhas como se fosse o RIDGE.