

Gradiente descendente

Eduardo Vargas Ferreira

Teorema: Seja $\mathbf{X} \in M_{n \times p}(\mathbb{R})$, com $n > p$ e $\text{posto}(\mathbf{X}) = p$. Definimos $J : \mathbb{R}^p \rightarrow \mathbb{R}$ da seguinte forma:

$$J(\beta) = \langle \mathbf{X}\beta - \mathbf{y}, \mathbf{X}\beta - \mathbf{y} \rangle ; \beta \in \mathbb{R}^p.$$

Então, o **Problema de Minimização**: encontrar $\hat{\beta} \in \mathbb{R}^p$ tal que

$$J(\hat{\beta}) = \min \{ J(\beta) ; \beta \in \mathbb{R}^p \}$$

é equivalente ao **Sistema Normal**

$$\mathbf{X}^t \mathbf{X} \beta = \mathbf{X}^t \mathbf{y}.$$

que resulta em

$$\hat{\beta} = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}.$$

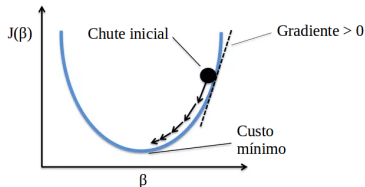
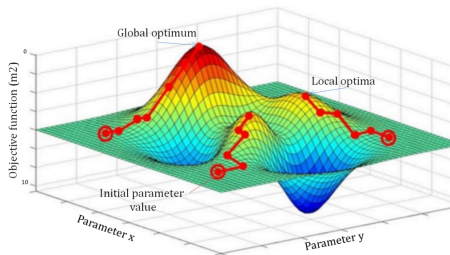
Método do Gradiente Descendente

- O **Gradiente Descendente** (GD) é um método para encontrar o mínimo de uma função de forma iterativa:

Algoritmo: Escolha um chute inicial, $\beta^{(0)} \in \mathbb{R}^p$, repita:

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla J(\beta^{(k)}), \quad k = 0, 1, \dots$$

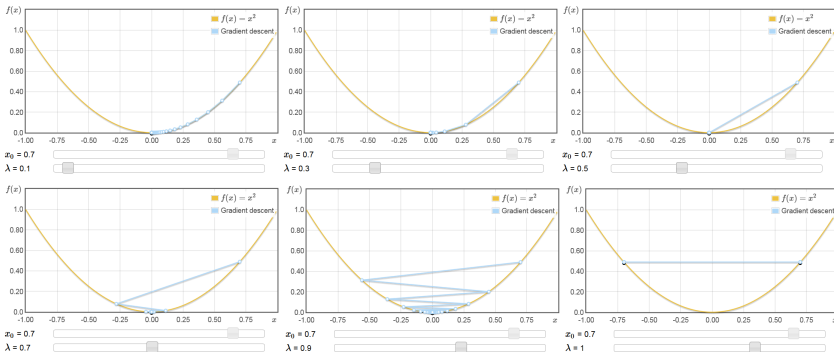
pare quando atingir convergência.



Taxa de aprendizagem α



- Taxa de aprendizagem controla o tamanho do passo em cada iteração;
- Selecionar o valor correto é crítico:
 - ★ Se tomarmos α pequeno, o método fica lento;
 - ★ Se α muito grande, o método diverge.

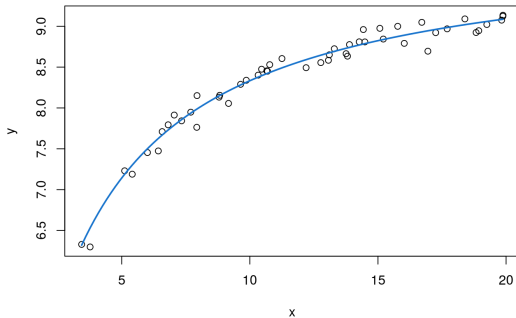


Exemplo simulado 1



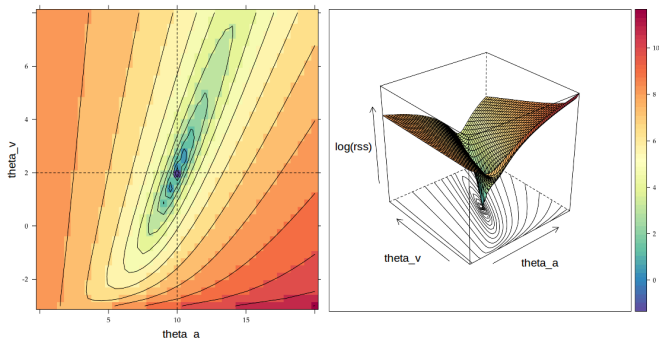
- Considere os dados simulados a partir do modelo:

$$f(x_i, \theta) = \frac{\theta_a x_i}{\theta_v + x_i}, \quad \text{com } (\theta_a, \theta_v)' = (10, 2).$$



- O nosso interesse será encontrar os valores de θ , que minimize a função:

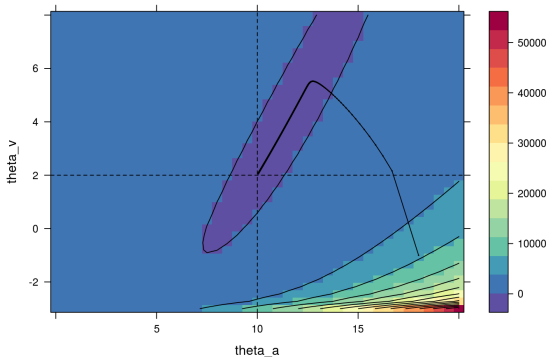
$$RSS = \sum_{i=1}^n \left(y_i - \frac{\theta_a x_i}{\theta_v + x_i} \right)^2 .$$



Exemplo simulado 1



- Utilizamos os critérios de parada: $|\theta^{(k+1)} - \theta^{(k)}| < 0.0001$ ou 10000 iterações. E a taxa de aprendizado, α , de 0,001.



- O algoritmo convergiu para os pontos $(\theta_a, \theta_v) = (10,03; 2,05)$, na iteração 2412.

- ✓ Ideia simples e cada iteração é barata;
- ✓ Garantia de convergência para o mínimo local;
- ✓ Com vários algoritmos de segunda ordem para acelerar sua convergência;
- ✓ Muito rápido para matrizes bem condicionadas e problemas fortemente convexos;
- ✗ Frequentemente é lento, pois problemas interessantes não são fortemente convexos ou bem condicionados;
- ✗ Não lida com funções não diferenciáveis (dica: use o método Subgradiente);
- ✗ Utiliza todos os dados de treinamento para estimar os parâmetros. Assim, para grandes bancos de dados, torna-se lento.

Gradiente Descendente Estocástico

- Considere o par (x_i, y_i) amostrado do treinamento. A atualização dos parâmetros é dada por

Algoritmo: Escolha um chute inicial, $\beta^{(0)} \in \mathbb{R}^{p+1}$, repita:

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla J(\beta^{(k)}; x_i, y_i), \quad k = 0, 1, \dots$$

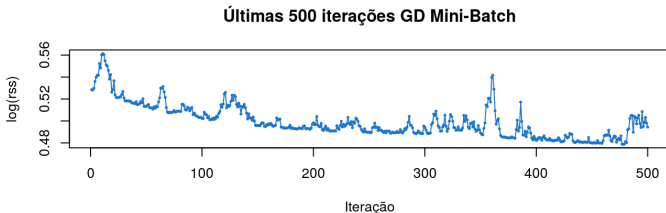
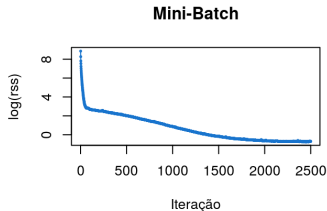
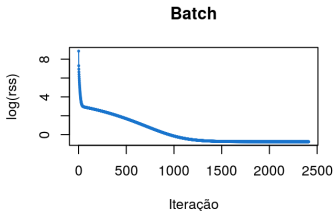
pare quando atingir convergência.

- No GDE a taxa de aprendizagem, α , é, tipicamente, menor do que o GD (*batch*). Isso ocorre, pois temos uma maior variância nas atualizações;
- Métodos mais sofisticados incluem o uso de **Backtracking line search** ou **Exact line search**.

Exemplo simulado 1 (continuação)



- Como era de se esperar, o método Mini-Batch utiliza mais iterações para atingir a convergência.



- ✓ Convergência mais rápida, especialmente com grandes bancos de dados ou dados redundantes, p. ex.:
 - Imagine que temos dados de treino de tamanho 100.000;
 - Mas na verdade são 100 cópias de 1000;
 - Ou seja, padrões parecidos, com mesmo efeito;
 - Batch será, pelo menos, 100 vezes mais devagar.
- ✓ A trajetória estocástica permite escapar de um mínimo local;
- ✗ Prova da convergência é probabilística;
- ✗ Muitos métodos de segunda ordem não funcionam.

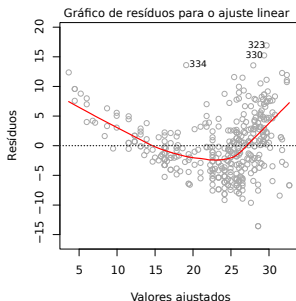
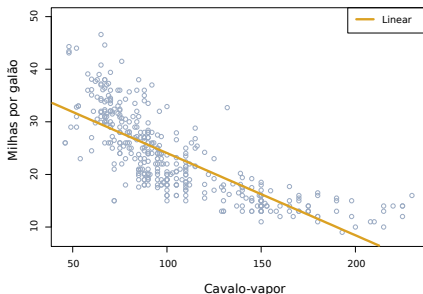
Gradiente boosting

Lembrando do método boosting



- No exemplo abaixo, estamos avaliando a relação entre consumo de combustível e a potência do automóvel.

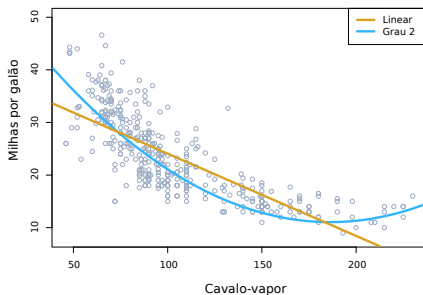
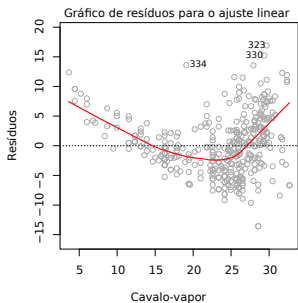
$$\begin{aligned}mpg &= \beta_0 + \beta_1 \times (\text{cavalo vapor}) + \text{Resíduo} \\ \underbrace{\sum_{i=1}^n (y_i - \bar{y})^2}_{SQT} &= \underbrace{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}_{SQR} + \underbrace{\sum_{i=1}^n (y_i - \hat{y}_i)^2}_{SQE}\end{aligned}$$



Lembrando do método boosting



$$\text{Resíduo} = \beta_2 \times (\text{cavalo vapor})^2 + \text{Resíduo 2}$$



$$mpg = \underbrace{\beta_0 + \beta_1 \times (\text{cavalo vapor})}_{\text{Parte1}} + \underbrace{\beta_2 \times (\text{cavalo vapor})^2}_{\text{Parte2}} + \underbrace{\text{Resíduo 2}}_{\text{Atualizado}}$$

- Considere o seguinte procedimento

$$Y = h(x) + \text{Resíduo} \quad (1)$$

- Se o **Resíduo** não for um ruído branco (mas algo correlacionado com Y)

$$\text{Resíduo} = g(x) + \text{Resíduo 2} \quad (2)$$

- Combinando (1) e (2)

$$Y = h(x) + g(x) + \text{Resíduo 2}$$

- Pode-se dizer que $h(x)$ foi atualizada com uma parte do **Resíduo**, ou seja

$$h(x)^{(2)} = h(x)^{(1)} + g(x)$$

- Queremos minimizar

$$J[y_i, h(\mathbf{x})] = \frac{1}{2n} \sum_{i=1}^n [y_i - h(\mathbf{x}_i)]^2$$

- Derivando com relação a $h(\mathbf{x}_i)$ temos

$$\frac{\partial J[y_i, h(\mathbf{x})]}{\partial h(\mathbf{x}_i)} = h(\mathbf{x}_i) - y_i$$

- Podemos interpretar os resíduos como o negativo do gradiente

$$\text{Resíduos} = y_i - h(\mathbf{x}_i) = -\frac{\partial J[y_i, h(\mathbf{x})]}{\partial h(\mathbf{x}_i)}$$

Resíduo \Leftrightarrow **Negativo do gradiente**
Atualizar $h(\mathbf{x}_i)$ com o resíduo \Leftrightarrow **Atualizar $h(\mathbf{x}_i)$ com o negativo do gradiente**

Resíduo \Leftrightarrow Negativo do gradiente

Atualizar $h(\mathbf{x}_i)$ com o resíduo \Leftrightarrow Atualizar $h(\mathbf{x}_i)$ com o negativo do gradiente

$$h(\mathbf{x}_i)^{(k+1)} = h(\mathbf{x}_i)^{(k)} + \text{Resíduo}$$

$$h(\mathbf{x}_i)^{(k+1)} = h(\mathbf{x}_i)^{(k)} - \frac{\partial J(y_i, h(\mathbf{x}))}{\partial h(\mathbf{x}_i)}$$

Algoritmo: Escolha um chute inicial, $h(\mathbf{x}_i)^{(0)}$, e:

* Calcule $-\frac{\partial J(y_i, h(\mathbf{x}))^{(k)}}{\partial h(\mathbf{x}_i)^{(k)}}$;

* Ajuste um modelo de regressão $g(\mathbf{x}_i)^{(k)}$ baseado no negativo do gradiente, e faça:

$$h(\mathbf{x}_i)^{(k+1)} = h(\mathbf{x}_i)^{(k)} + \rho g(\mathbf{x}_i)^{(k)}, \quad k = 0, 1, \dots$$

pare quando atingir convergência.

- **Soma dos desvios absolutos (SDA)**

$$J[y_i, h(\mathbf{x})] = \frac{1}{n} \sum_{i=1}^n |y_i - h(\mathbf{x}_i)|$$

$$-\frac{\partial J[y_i, h(\mathbf{x})]}{\partial h(\mathbf{x}_i)} = \text{sign}[y_i - h(\mathbf{x}_i)] = \begin{cases} 1, & \text{se } |y_i - h(\mathbf{x}_i)| < 0, \\ -1, & \text{se } |y_i - h(\mathbf{x}_i)| > 0 \end{cases}$$

- **Huber-M cost**

$$J[y_i, h(\mathbf{x})] = \frac{1}{n} \sum_{i=1}^n \begin{cases} \frac{1}{2} [y_i - h(\mathbf{x}_i)]^2, & \text{para } |y - h(\mathbf{x}_i)| \leq \delta, \\ \delta |y_i - h(\mathbf{x}_i)| - \frac{1}{2} \delta^2, & \text{caso contrário.} \end{cases}$$

$$-\frac{\partial J[y_i, h(\mathbf{x})]}{\partial h(\mathbf{x}_i)} = \begin{cases} y_i - h(\mathbf{x}_i), & \text{se } |y_i - h(\mathbf{x}_i)| \leq \delta, \\ \delta \text{sign}[y_i - h(\mathbf{x}_i)], & \text{caso contrário.} \end{cases}$$

- Para mais detalhes, veja [Greedy Function Approximation: A Gradient Boosting Machine](#)

XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting)



Modelos de Regressão



Extreme Gradient Boosting



- XGBoost é o algoritmo mais popular de ML atualmente. Desde a sua criação, em 2014, tornou-se o “*true love*” dos competidores do Kaggle;

- **booster[default=gbtree]**

- ★ Determina o tipo de booster (gbtree, gblinear ou dart).

- **nrounds[default=100]**

- ★ Controla o número de iterações. Para classificação, é similar ao número de árvores.

- **eta[default=0.3][range: (0,1)]**

- ★ Controla a taxa de aprendizado. Tipicamente, utilizamos valores entre 0,01 e 0,3.

- **max_depth[default=6][range: (0,Inf)]**

- ★ Controla o tamanho de cada árvore. Geralmente, utilizamos árvores menores, para evitar o superajuste.

- **eval_metric [no default, depends on objective selected]**
 - ★ Métrica utilizada para avaliar o modelo.
- **subsample[default=1][range: (0,1)]**
 - ★ Controla o tamanho da amostra em cada árvore. Tipicamente, são valores entre 0,5 e 0,8.
- **colsample_bytree[default=1][range: (0,1)]**
 - ★ Controla o número de variáveis apresentada à árvore. Tipicamente, são valores entre 0,5 e 0,9.
- **lambda[default=0]**
 - ★ Controla o *penalty* ℓ_2 nos pesos (equivalente ao Ridge). Utilizado para evitar superajuste.

- James, G., Witten, D., Hastie, T. e Tibshirani, An Introduction to Statistical Learning, 2013;
- Hastie, T., Tibshirani, R. e Friedman, J., The Elements of Statistical Learning, 2009;
- Lantz, B., Machine Learning with R, Packt Publishing, 2013;
- Tan, Steinbach, and Kumar, Introduction to Data Mining, Addison-Wesley, 2005;
- Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani