Universidade Federal do Paraná

Laboratório de Estatística e Geoinformação - LEG

Gradiente descendente (batch, stochastic e boosting)

Prof.: Eduardo Vargas Ferreira

Solução de quadrados mínimos

- Vamos lembrar do problema:

$$y_i = \boldsymbol{\beta}^t \mathbf{x_i} + \varepsilon_i, \quad \text{com } \boldsymbol{\varepsilon} \sim N(0, \sigma^2).$$

- Isso implica em

$$P(y_i|x_i;\boldsymbol{\beta}) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y_i - \boldsymbol{\beta}^t \mathbf{x}_i)^2}{2\sigma^2}\right).$$

- A notação $P(y_i|x_i;\boldsymbol{\beta})$ indica que essa é a distribuição de y_i dado $\mathbf{x}_i$ e parametrizado por $\boldsymbol{\beta}$ (não é condicionado em $\boldsymbol{\beta}$, pois ele não é v.a.);

- Calculando a log-verossimilhança temos

$$\begin{aligned} \ell(\boldsymbol{\beta}) &= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y_i - \boldsymbol{\beta}^t \mathbf{x}_i)^2}{2\sigma^2}\right) \\ &= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^t \mathbf{x}_i)^2. \end{aligned}$$

- Assim, maximizar a log-verossimilhança é equivalente a minimizar $J(\boldsymbol{\beta})$.

Solução de quadrados mínimos

- Seja $\mathbf{X} \in \mathrm{M}_{n\times p}(\mathrm{R})$, com $n > p$ e $posto(\mathbf{X}) = p$. Dado $\mathbf{y} \in \mathrm{R}^n$, definimos o seguinte problema de minimização:

$$\left\| X\hat{\boldsymbol{\beta}} - \mathbf{y} \right\|_2^2 = min\left\{ \|X\boldsymbol{\beta} - \mathbf{y}\|_2 \ ; \ \boldsymbol{\beta} \in \mathrm{R}^{p+1} \right\}$$

- Dizemos que o elemento $\hat{\beta}$ é uma solução de quadrados mínimos;

Teorema: Seja $\mathbf{X} \in \mathrm{M}_{n\times p}(\mathrm{R})$, com $n > p$ e $posto(\mathbf{X}) = p$. Definimos o funcional $J : \mathrm{R}^{p+1} \to \mathrm{R}$ da seguinte forma:

$$J(\boldsymbol{\beta}) = \langle \mathbf{X}\boldsymbol{\beta} - \mathbf{y}, \mathbf{X}\boldsymbol{\beta} - \mathbf{y} \rangle \ ; \ \boldsymbol{\beta} \in \mathrm{R}^{p+1}.$$

Então, o **Problema de Minimização**: encontrar $\hat{\beta} \in \mathrm{R}^{p+1}$ tal que

$$J(\hat{\boldsymbol{\beta}}) = min\left\{ J(\boldsymbol{\beta}) \ ; \ \boldsymbol{\beta} \in \mathrm{R}^{p+1} \right\}$$

é equivalente ao **Sistema Normal**

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

Demonstração

- O Gradiente do funcional J no ponto $\boldsymbol{\beta} \in \mathrm{R}^{p+1}$ é definido por:

$$\begin{aligned}
\nabla J(\boldsymbol{\beta}) &= \nabla \left(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\right)^t \left(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\right) \\
&= \nabla \left(\boldsymbol{\beta}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^t\mathbf{X}^t\mathbf{X}\mathbf{y} - \mathbf{y}^t\mathbf{X}\boldsymbol{\beta} + \mathbf{y}^t\mathbf{y}\right) \\
&= \nabla tr \left(\boldsymbol{\beta}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^t\mathbf{X}^t\mathbf{y} - \mathbf{y}^t\mathbf{X}\boldsymbol{\beta} + \mathbf{y}^t\mathbf{y}\right) \\
&= \nabla \left(tr\boldsymbol{\beta}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} - 2tr\mathbf{y}^t\mathbf{X}\boldsymbol{\beta}\right) \\
&= \nabla \left(\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} + \mathbf{X}^t\mathbf{X}\boldsymbol{\beta} - 2\mathbf{X}^t\mathbf{y}\right) \\
&= 2\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} - 2\mathbf{X}^t\mathbf{y}
\end{aligned}$$

- Dizemos que $\hat{\boldsymbol{\beta}}$ é um ponto crítico do funcional J se, e somente se,

$$\nabla J(\boldsymbol{\beta})(v) = 2\left\langle \mathbf{X}^t\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}^t\mathbf{y}, \mathbf{v} \right\rangle = 0 \text{ , para todo } \mathbf{v} \in \mathrm{R}^{p+1}$$

- $\nabla J(\boldsymbol{\beta})(v)$ é derivada direcional de J no ponto $\hat{\boldsymbol{\beta}}$ na direção de $\mathbf{v} \in \mathrm{R}^{p+1}$;

- Portanto, o único ponto crítico do funcional J é caracterizado como:

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

Método do gradiente descendente (GD)

- O **Gradiente descendente** (GD) é um método para encontrar o mínimo de uma função de forma iterativa;
- Cada passo pode ser visto como o problema de minimização

$$\boldsymbol{\beta}^{(\mathbf{k+1})} = \underset{\boldsymbol{\beta}}{\textit{argmin}}\ J(\boldsymbol{\beta}^{(k)}) + \nabla J(\boldsymbol{\beta}^{(k)})^t \left(\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}\right) + \frac{1}{2\alpha} \left\|\boldsymbol{\beta} - \boldsymbol{\beta}^{(\mathbf{k})}\right\|^2$$

- Note que é uma aproximação quadrática trocando $\nabla^2 J(\boldsymbol{\beta}^{(k)})$ por $\frac{1}{2\alpha} I$.
- Derivando com relação a $\boldsymbol{\beta}$ temos

$$0 = \nabla J(\boldsymbol{\beta}^{(k)}) + \frac{1}{\alpha}(\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}) \Rightarrow \boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \nabla J(\boldsymbol{\beta}^{(k)})$$

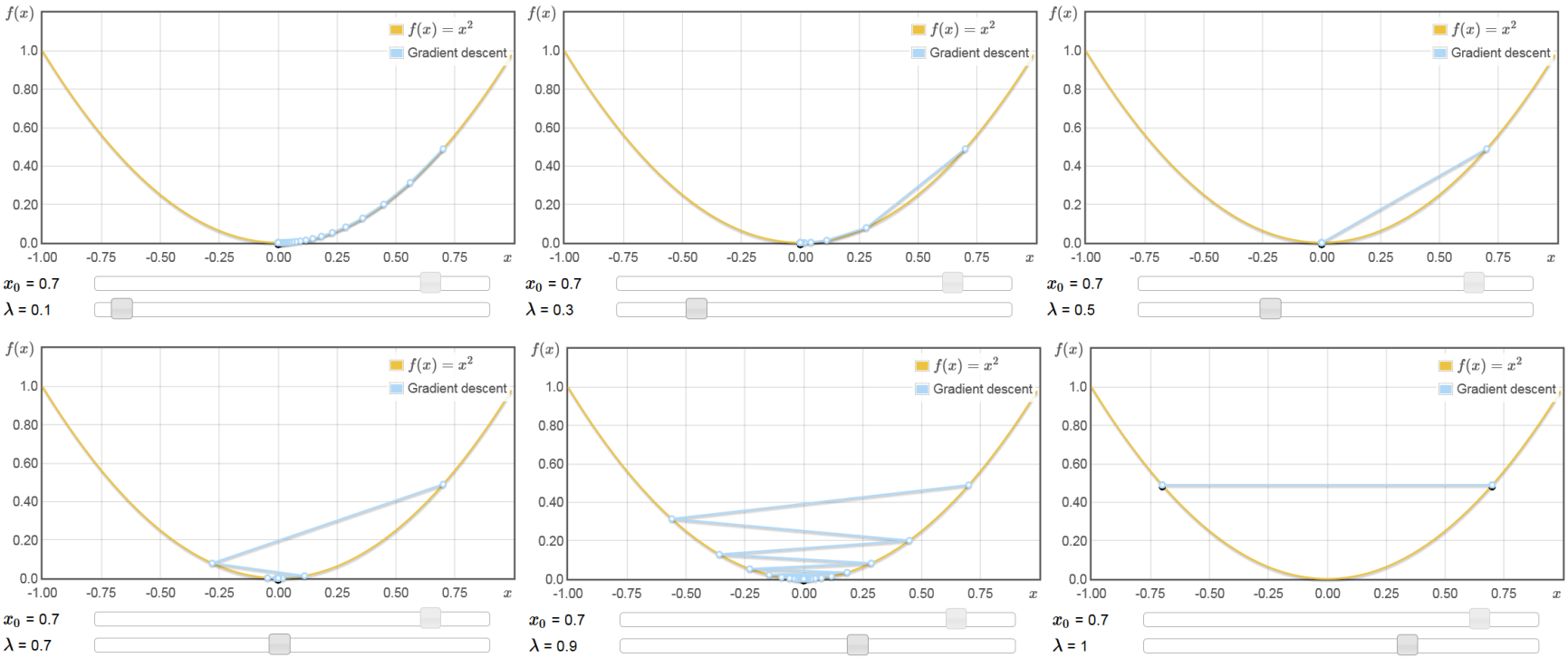
Algoritmo: Escolha um chute inicial, $\boldsymbol{\beta}^{(0)} \in \mathrm{R}^{p+1}$, repita:

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

pare quando atingir convergência.

Taxa de aprendizagem α

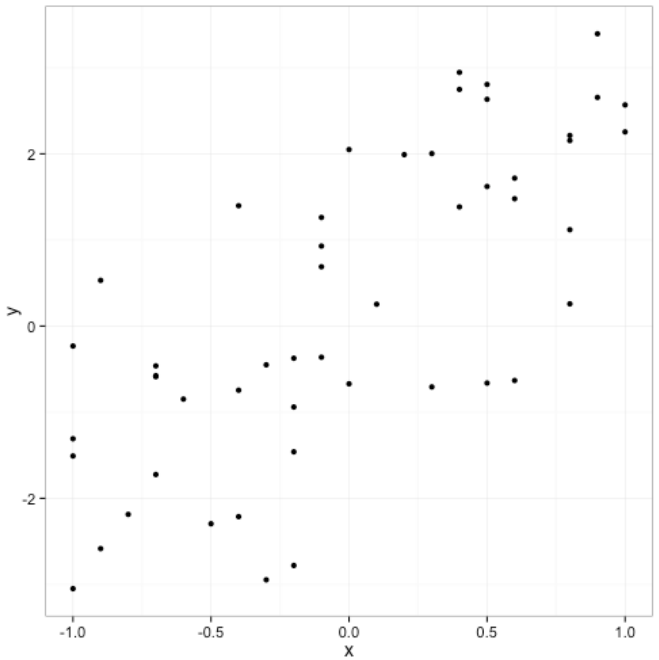
- Taxa de aprendizagem controla o tamanho do passo em cada iteração;
- Selecionar o valor correto é crítico
  - ⋆ Se tomarmos α pequeno, o método fica lento;
  - ⋆ Se α muito grande, o método diverge.



Exemplo 1

- Vamos começar com um exemplo simulado de regressão linear simples
- O algoritmo não será apresentado por completo, sua implementação ficará como exercício (depois compare os resultados).

```
set.seed(12345)
x <- sample(seq(from = -1, to = 1, by = 0.1), size = 50, replace = TRUE)
y <- 2 * x + rnorm(50)
```

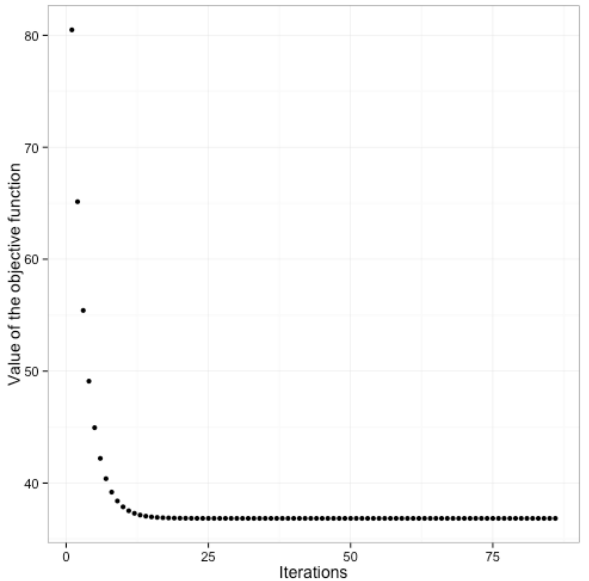


Exemplo 1

```
X <- as.matrix(x)
 y <- as.vector(y)
 f <- function(X, y, b) {
     (1/2) * norm(y - X %*% b, "F")^2
     }
 grad_f <- function(X, y, b) {
     t(X) %*% (X %*% b - y)
     }
 simple_ex <- graddesc(f, grad_f, X, y, 0.01)

plot_loss(simple_ex)
```

```
## Minimum function value:
##  36.85
##
## Intercept:
##  0.28
##
## Coefficient(s):
##  2.123
```



Exemplo 2

- O segundo exemplo vamos utilizar o `moviebudgets` dataset. O banco contém as notas de 5183 filmes, orçamento aproximado, ano etc.;

```
data(moviebudgets)
head(moviebudgets)
```

```
##                                 title year length    budget rating votes
## 1                             Titanic 1997    194 200000000    6.9 90195
## 2                          Spider-Man 2 2004    127 200000000    7.9 40256
## 3                                 Troy 2004    162 185000000    7.1 33979
## 4                           Waterworld 1995    176 175000000    5.4 19325
## 5 Terminator 3: Rise of the Machines 2003    109 175000000    6.9 32111
## 6                       Wild Wild West 1999    107 170000000    4.0 19078
##     r1   r2   r3   r4   r5   r6   r7   r8   r9  r10  mpaa Action Animation
## 1 14.5  4.5  4.5  4.5  4.5  4.5 14.5 14.5 14.5 24.5 PG-13      0         0
## 2  4.5  4.5  4.5  4.5  4.5  4.5 14.5 24.5 24.5 24.5 PG-13      1         0
## 3  4.5  4.5  4.5  4.5  4.5 14.5 14.5 14.5 14.5 14.5     R      1         0
## 4  4.5  4.5  4.5 14.5 14.5 14.5 14.5 14.5  4.5  4.5 PG-13      1         0
## 5  4.5  4.5  4.5  4.5  4.5 14.5 24.5 24.5  4.5 14.5     R      1         0
## 6 14.5 14.5 14.5 14.5 14.5 14.5  4.5  4.5  4.5  4.5 PG-13      1         0
##   Comedy Drama Documentary Romance Short
## 1      0     1           0       1     0
## 2      0     0           0       0     0
## 3      0     1           0       1     0
## 4      0     1           0       0     0
## 5      0     0           0       0     0
## 6      1     0           0       0     0
```

Exemplo 2

```
f <- function(X, y, b) {
      (1/2) * norm(y - X %*% b, "F")^2
      }
 grad_f <- function(X, y, b) {
      t(X) %*% (X %*% b - y)
      }

 X <- as.matrix(moviebudgets$budget)
 y <- as.vector(moviebudgets$rating)
 movies1 <- graddesc(f, grad_f, X, y, 1e-04, 5000)
```

- Função `graddesc`

```
## Minimum function value:
##  6174
##
## Intercept:
##  6.149
##
## Coefficient(s):
##  -8.533e-10
```

- Função `lm`

```
##
## Call:
## lm(formula = y ~ X)
##
## Coefficients:
## (Intercept)            X
##    6.15e+00    -8.53e-10
```

Prós e contras do GD

✓ 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;

χ Frequentemente é lento, pois problemas interessantes não são fortemente convexos ou bem condicionados;

χ Não lida com funções não diferenciáveis (dica: use o método Subgradiente).

χ Utiliza todos os dados de treinamento para estimar os parâmetros. Assim, para grandes bancos de dados torna-se lento;

- Diante deste último aspecto, por que não em cada iteração selecionar um valor na amostra e com sua informação executar um passo?

Gradiente descendente estocástico (GDE)

Gradiente descendente estocástico (GDE)

- Como vimos, no gradiente descendente utilizamos a **amostra completa** para atualizar os parâmetros (é um processo determinístico);
- Assim, se o tamanho da amostra de treino for grande (na verdade MUITO grande!) o gradiente descendente levará muito tempo em cada passo;
- A diferença no **Gradiente descendente estocástico** (GDE) está na utilização de somente **uma observação** em cada iteração.
- Então, cada passo é realizado com uma v.a. de um processo estocástico;
- Em redes neurais, por exemplo, o custo para se fazer **backpropagation** com os dados completos é muito alto. Portanto, abordagens estocásticas como esta torna o método mais atrativo.

Gradiente descendente estocástico (GDE)

- Considere o par $(\mathbf{x}_i, y_i)$ amostrado do treinamento. A atualização dos parâmetros é dada por

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

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha_k \nabla J(\boldsymbol{\beta}^{(k)}; \mathbf{x}_i, y_i), \ k = 0, 1, \ldots$$

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;

- Uma escolha de α que funciona bem na prática é uma taxa pequena o suficiente que dê uma convergência estável nas iterações iniciais;

- Métodos mais sofisticados incluem o uso de *Backtracking line search* ou *Exact line search*.

Prós e contras do GDE

✓ 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;

χ Prova da convergência é probabilística;

χ Muitos métodos de segunda ordem não funcionam;

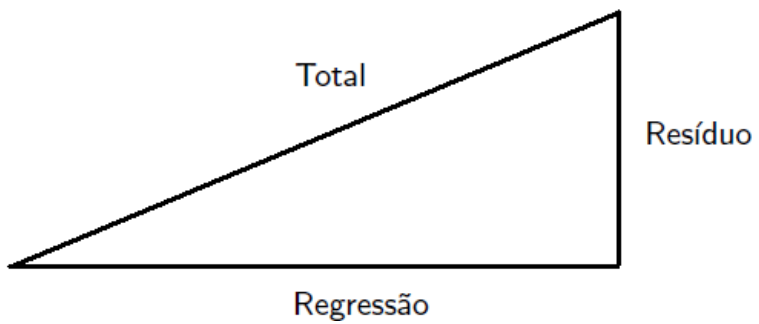
Gradiente boosting

Intuição

- Lembrando de regressão

$$\underbrace{\sum_{i=1}^{n}(y_i-\bar{y})^2}_{SQT}=\underbrace{\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2}_{SQR}+\underbrace{\sum_{i=1}^{n}(y_i-\hat{y})^2}_{SQE}.$$

- E, geometricamente, temos



- Isso quer dizer que toda variabilidade **não explicada** pela regressão ficará no resíduo (variáveis e funções delas!);

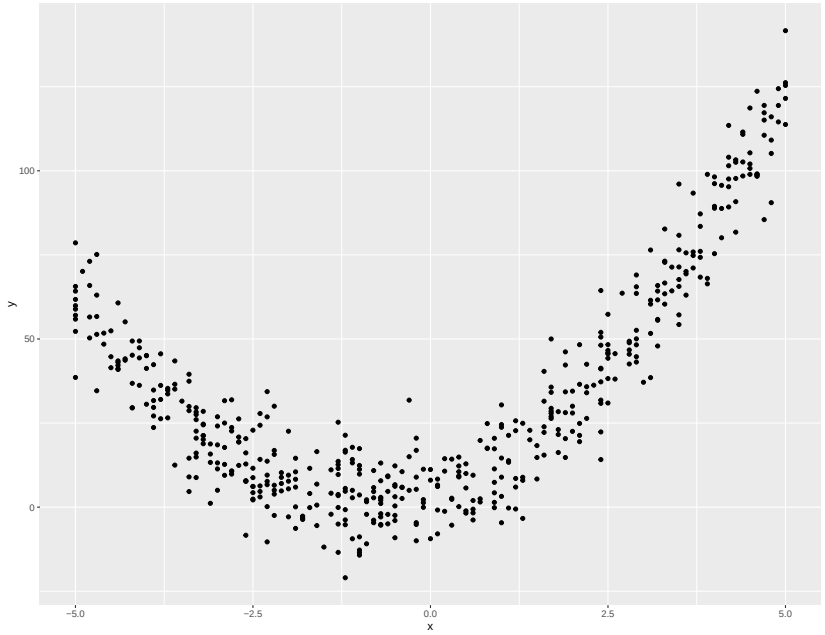
- Vejamos um exemplo.

Intuição

- Simular uma situação na qual a **verdadeira** relação entre X e Y é

$$y = 3,5x^2 + 6x + 5$$

```
x <- sample(seq(from = -5, to = 5, by = 0.1), size = 500, replace = TRUE)
y <- 3.5*x*x + 6*x + 5 + rnorm(500,0,10)
```



Intuição

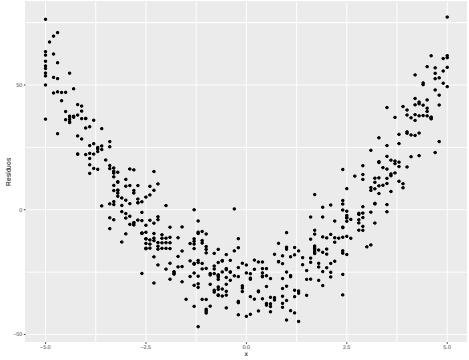
- Agora, vamos ajustar um modelo de regressão linear simples

```
ajuste = lm(y ~ x)
ajuste$coef
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  33.3499     1.2533   26.61   <2e-16 ***
## x             6.2182     0.4409   14.11   <2e-16 ***
```

- **Pergunta:** Se o termo quadrático não está no modelo, onde ele estará?
- **Resposta:** Nos resíduos.

```
e = ajuste$residuals
qplot(x,e,
      xlab="x",
      ylab="Valores ajustados")
```



Intuição

- Então, considere o seguinte procedimento

$$Y = h(x) + residuo \tag{1}$$

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

$$residuo = g(x) + residuo2 \tag{2}$$

- Combinando (1) e (2)

$$Y = h(x) + g(x) + residuo2$$

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

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

- Mas, como isto está relacionado com **Gradiente boosting**?

Gradiente boosting e resíduos

- 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

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

- Então, considerando perda quadrática, concluímos que

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

Gradiente boosting

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

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

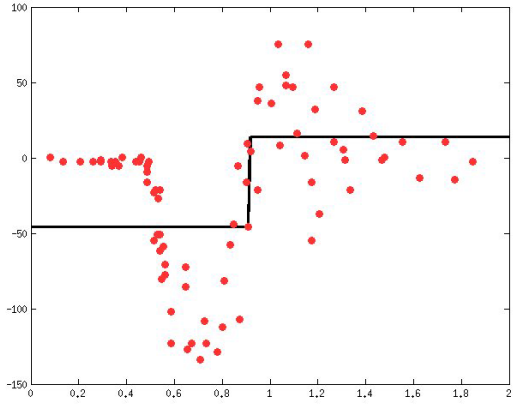
* Ajuste um modelo de regressão $g(\mathbf{x}_i)^{(k)}$ baseado no negativo do gradiente;

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

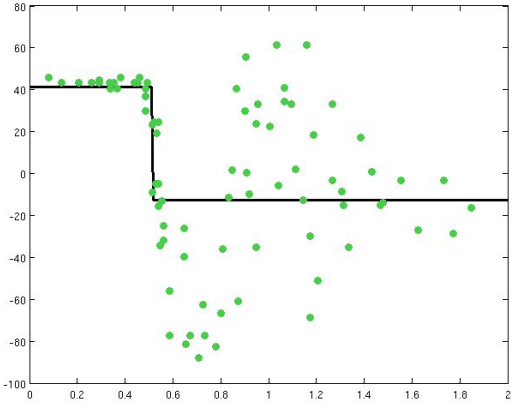
pare quando atingir convergência.

Exemplo:

Começando com um simples preditor



Aprimorando com os resíduos



Gradiente boosting

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

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

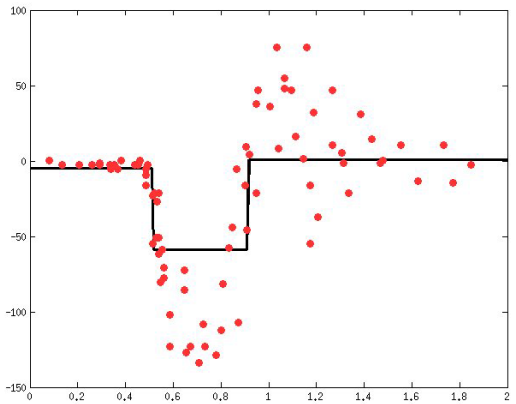
* Ajuste um modelo de regressão $g(\mathbf{x}_i)^{(k)}$ baseado no negativo do gradiente;

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

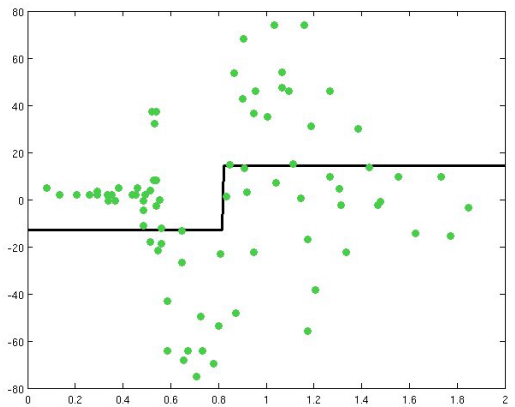
pare quando atingir convergência.

Exemplo:

Combinando, temos um melhor preditor

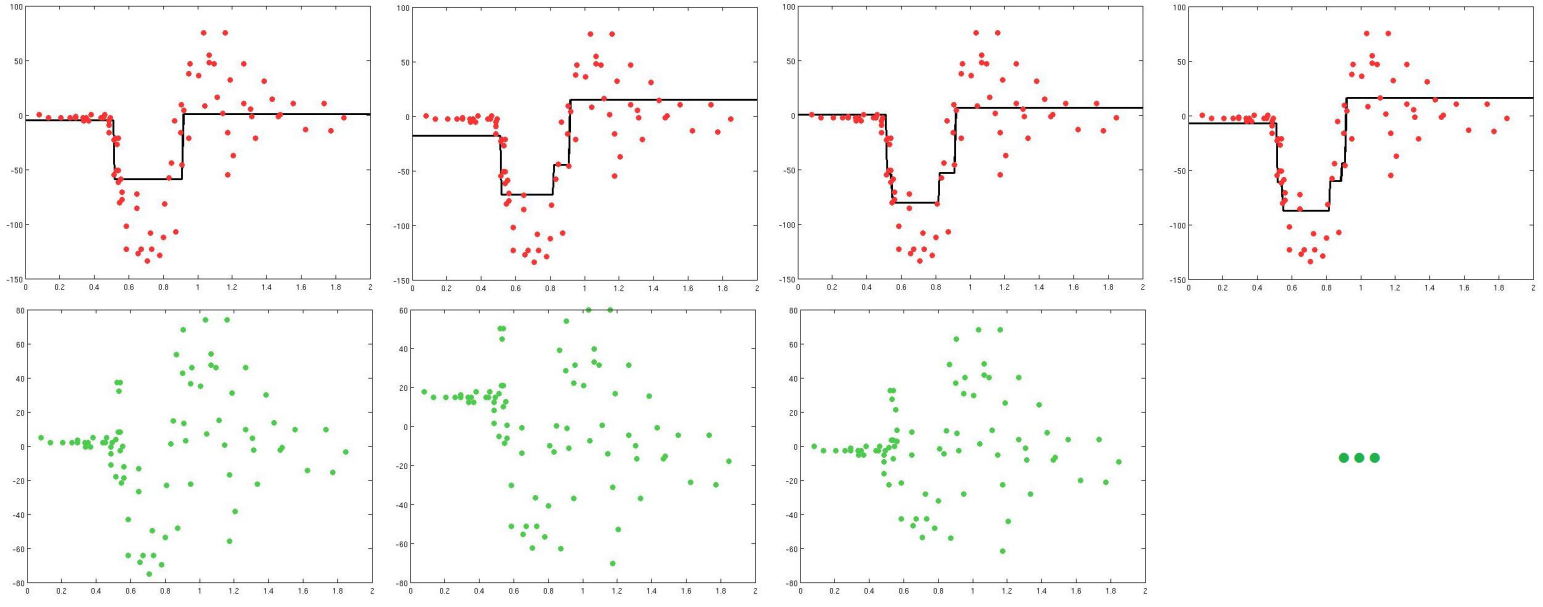


Novamente, aprimorando com os resíduos



Gradiente boosting

- O princípio básico é esse; propor um modelo e aprimorá-lo (ou "ensiná-lo") através da análise dos resíduos;



- Note que podemos considerar outras funções perda e derivar o algoritmo da mesma maneira.

Outras funções perda

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

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

  - ⋆ O negativo do gradiente fica

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

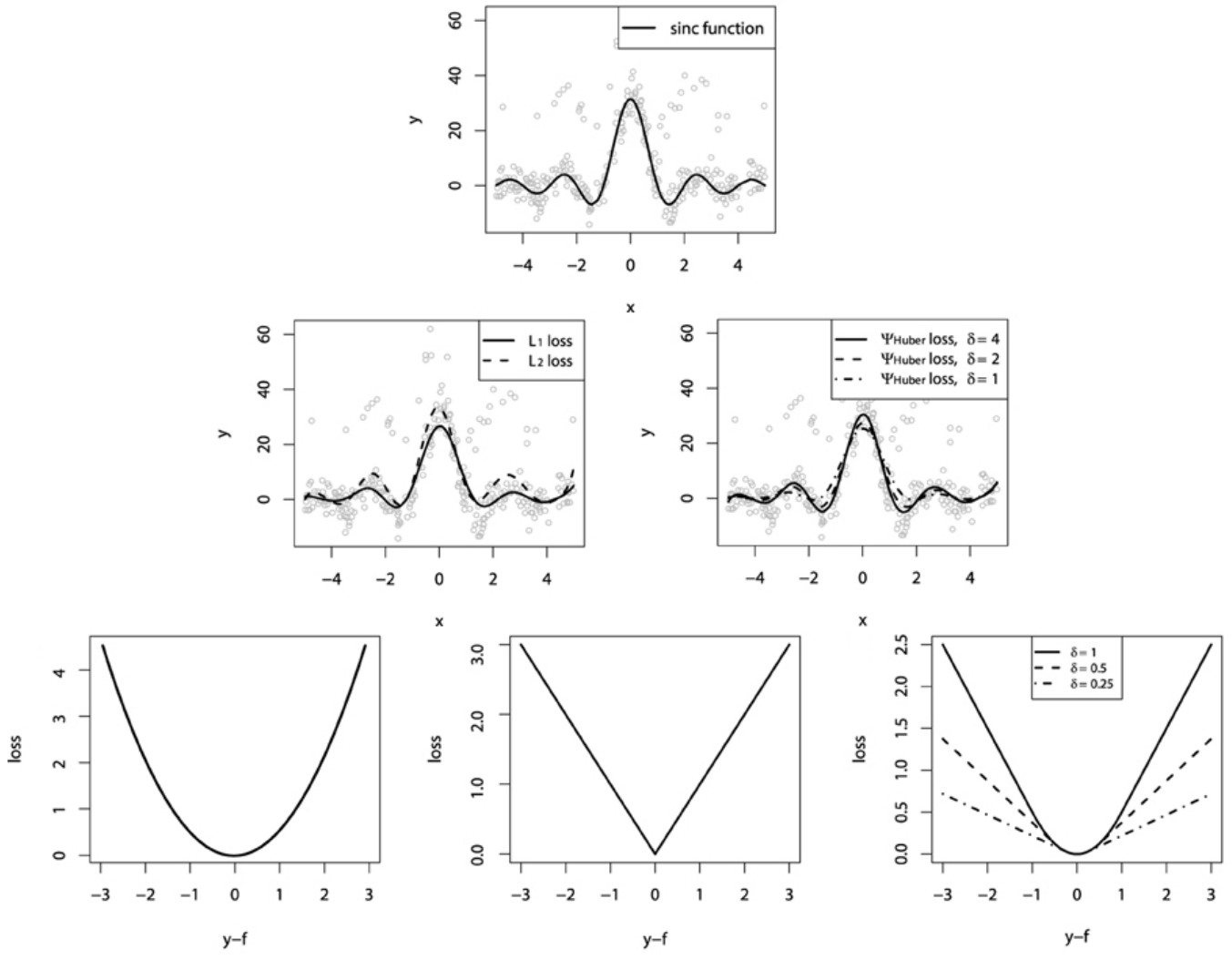
- **Huber-M cost**

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

  - ⋆ O negativo do gradiente fica

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

Outras funções perda



Conclusão

- O método introduz um novo modelo de regressão em cada iteração, a fim de compensar as deficiências do modelo existente;
- As deficiências são identificadas pelo negativo do gradiente;
- Para qualquer função perda podemos derivar o Gradiente boosting;
- Perda absoluta e Huber são mais robustos a outliers;
- Para detalhes de como escolher o valor de δ, veja
  - ▸ Greedy Function Approximation: A Gradient Boosting Machine