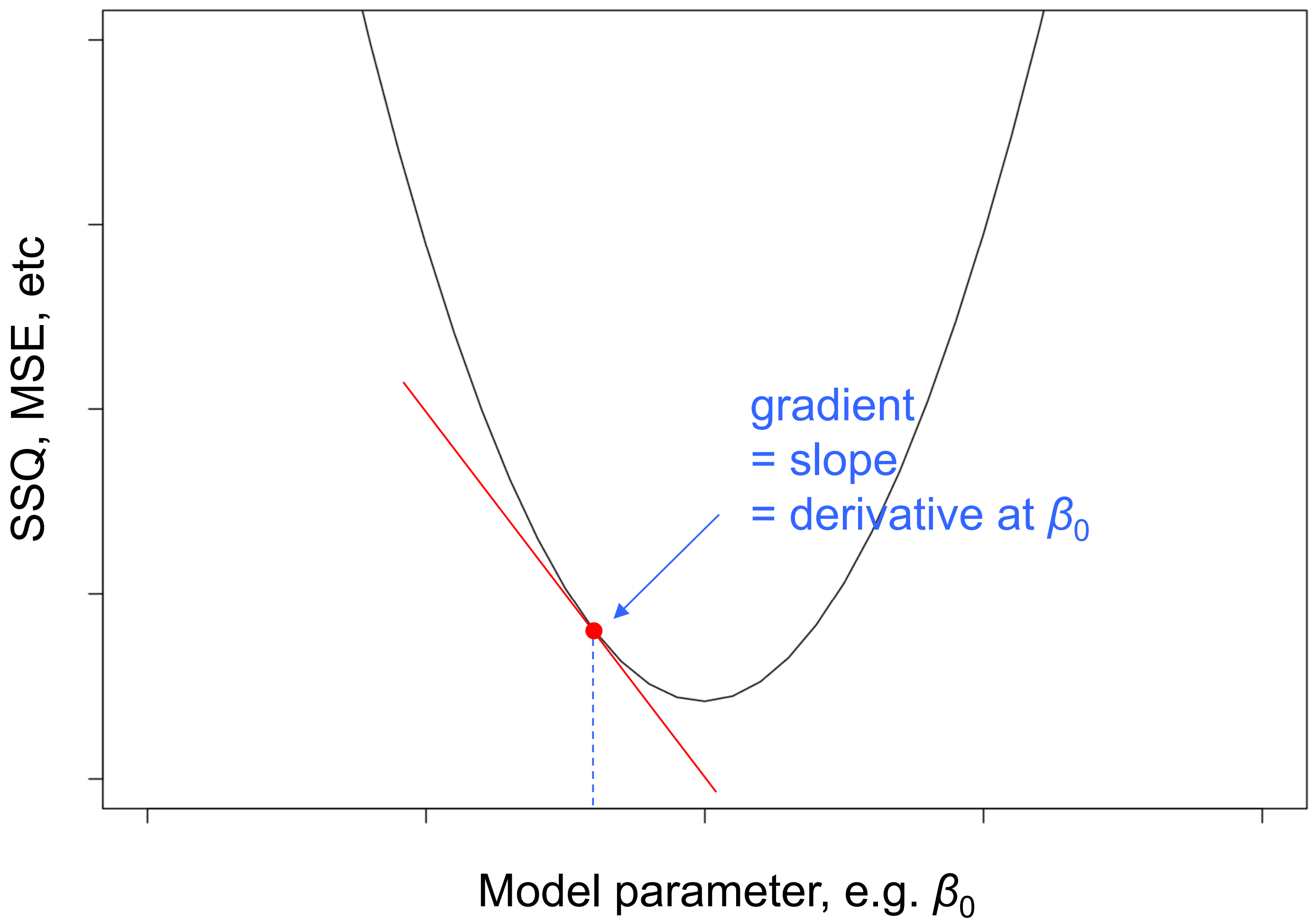
Today

- Ensemble methods
  - Bagging
  - Random forest
  - Boosting

- But first
  - basic gradient descent
  - boosting is a variant

Gradient descent



Finding the gradient

- It turns out that the gradient for a linear model is a function of the residuals
- See math

$$SSQ = \sum_{i}^{n} (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

$$r_i = y_i - \beta_0 - \beta_1 x_i$$

$$= \sum_{i}^{n} r_i^2$$

$$= \sum_{1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$= \sum_{1}^{n} y_i^2 - 2y_i\beta_0 - 2y_i\beta_1 x_i + \beta_0^2 + 2\beta_0\beta_1 x_i + \beta_1^2 x_i^2$$

$$\frac{\partial SSQ}{\partial \beta_1} = \sum_{1}^{n} (-2y_i x_i + 2\beta_0 x_i + 2\beta_1 x_i^2)$$

$$= \sum_{1}^{n} -2x_i (y_i - \beta_0 - \beta_1 x_i)$$

$$= -2 \sum_{i}^{n} r_i x_i$$

$$\frac{\partial SSQ}{\partial \beta_0} = \sum_{1}^{n} -2y_i + 2\beta_0 + 2\beta_1 x_i$$

$$= -2 \sum r_i$$

Gradient descent

Gradient descent training algorithm for a linear model

set lambda (learning rate)
make initial guess for β_0, β_1
for many iterations
    find gradient at β_0, β_1
    step down: $\beta = \beta$ - lambda * gradient(β)
print β_0, β_1

Gradient boosting

Gradient boosting algorithm (intuitive version)

set lambda (learning rate)

fit a model, m, to the data
keep a fraction of the model, λm
calculate the left over variation, $\hat{r}$

repeat until no more systematic variation in $\hat{r}$
    fit a model, m, to $\hat{r}$
    keep a fraction of the model, λm
    add to previous fraction
    calculate the left over variation, $\hat{r}$

Gradient boosting

Algorithm

load y, x, x_{new}

guess parameters:

set $\hat{f}(x_{\text{new}})$ = 0

set $r \leftarrow y$ (residuals equal to the data)

for m in 1 to n_iterations

    train model on r and x

    predict residuals, $\hat{r}_m(x)$, from trained model

    update residuals: $r \leftarrow r - \lambda\hat{r}_m(x)$

    predict y increment, $\hat{f}_m(x_{\text{new}})$, from trained model

    update prediction: $\hat{f}(x_{\text{new}}) \leftarrow \hat{f}(x_{\text{new}}) + \lambda\hat{f}_m(x_{\text{new}})$

return $\hat{f}(x_{\text{new}})$

Can be any model

Gradient descent

Gradient descent

predict residuals, $\hat{r}_m(x)$, from trained model
update residuals: $r \leftarrow r - \lambda \hat{r}_m(x)$

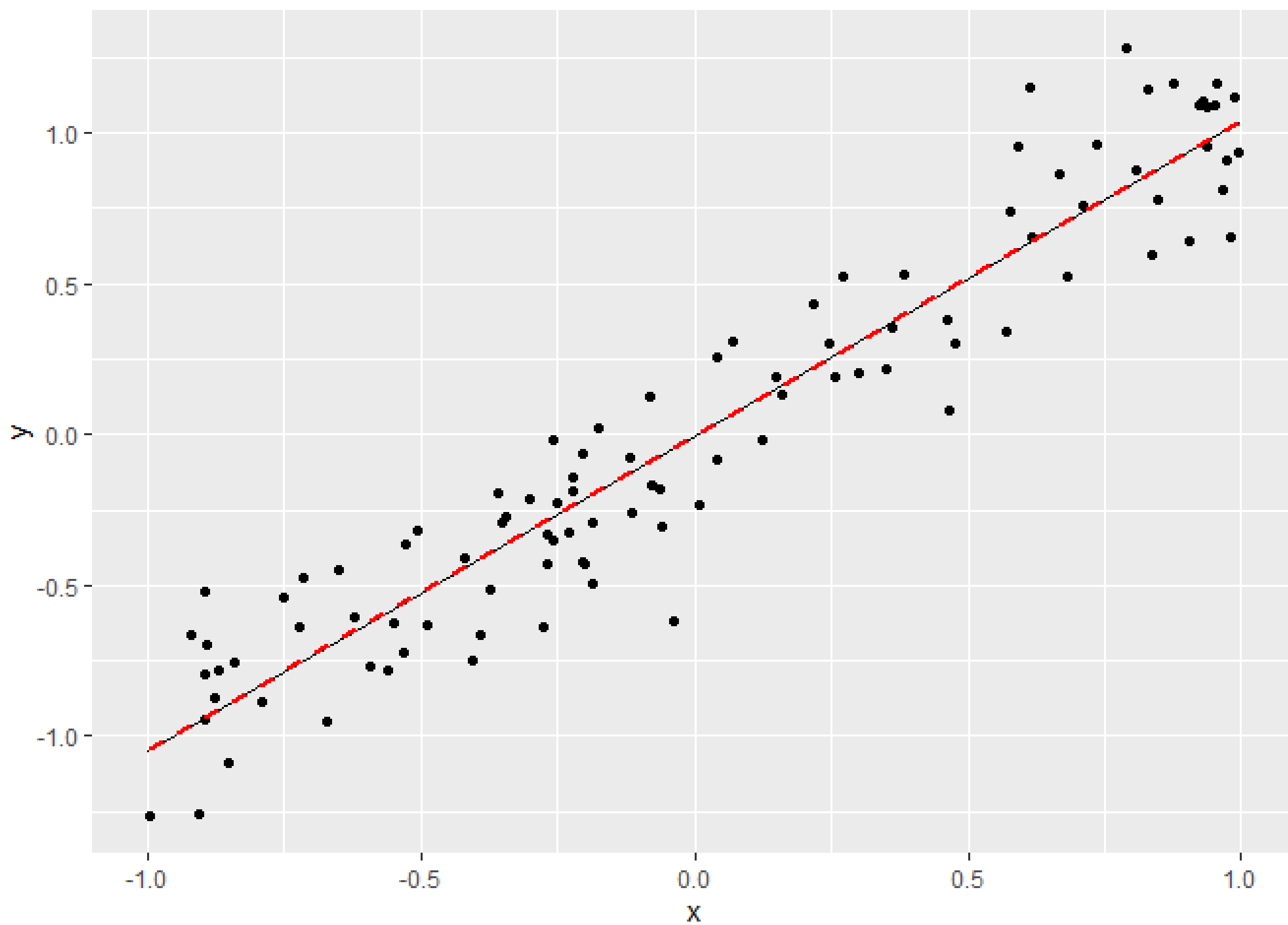
Loss function is MSE and we are descending its surface

Gradient at x is proportional to the predicted residual $\hat{r}_m(x)$

$\lambda \hat{r}_m(x)$ is the increment taken down the gradient

r gets closer to 0 at each step, so MSE goes down

Boosted linear model (black line) compared to linear regression (red dashed line). They are the same.



Boosted regression tree

Algorithm

load y, x, x_{new}

set parameters: tree_complexity, ntrees, λ

set $\hat{f}(x_{\text{new}})$ = 0

set $r \leftarrow y$ (residuals equal to the data)

for m in 1 to ntrees

    train tree model, m, on r and x

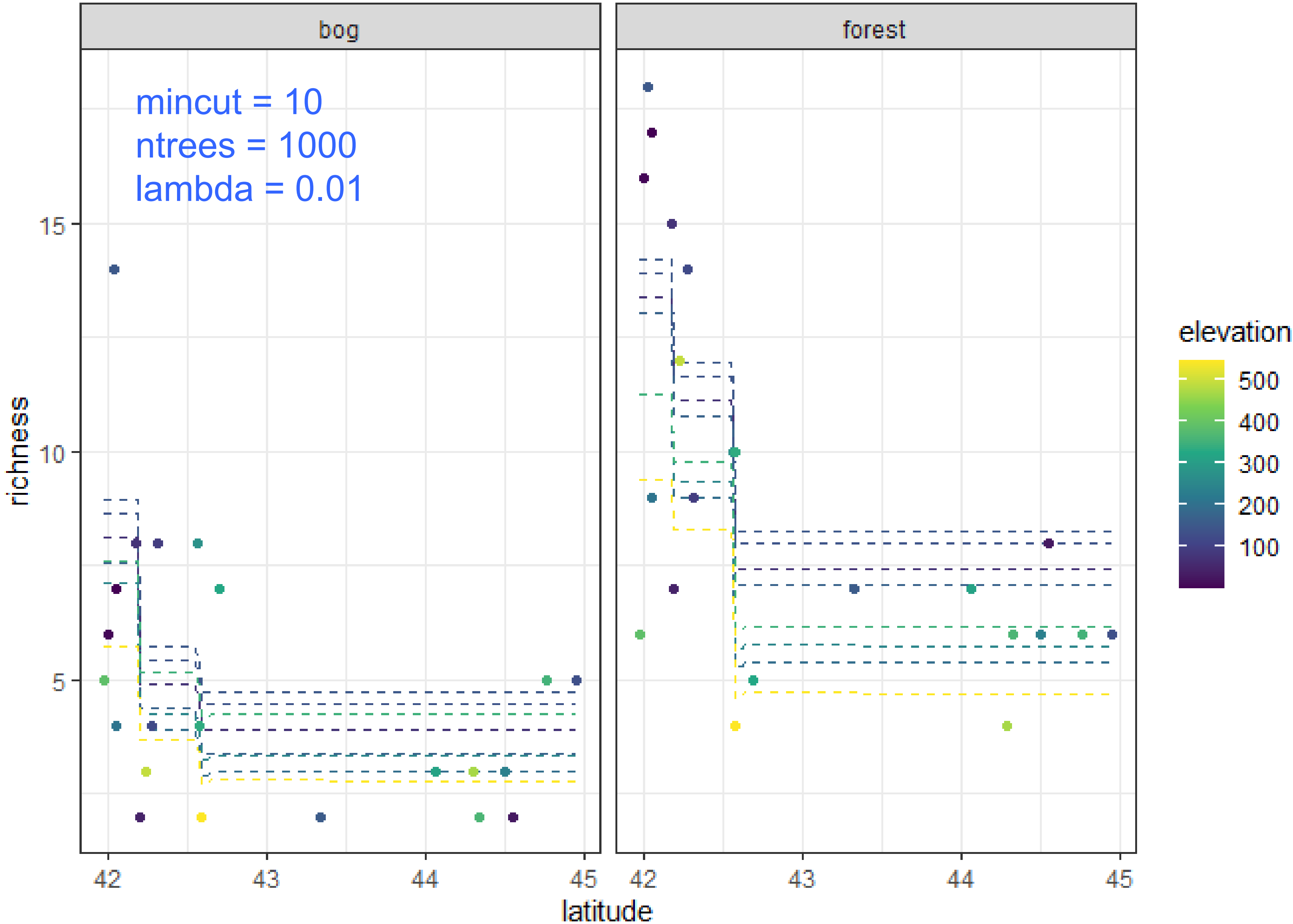
    predict residuals, $\hat{r}_m(x)$, from trained tree

    update residuals: $r \leftarrow r - \lambda \hat{r}_m(x)$

    predict y increment, $\hat{f}_m(x_{\text{new}})$, from trained tree

    update prediction: $\hat{f}(x_{\text{new}}) \leftarrow \hat{f}(x_{\text{new}}) + \lambda \hat{f}_m(x_{\text{new}})$

return $\hat{f}(x_{\text{new}})$


bog
forest
mincut = 10
ntrees = 1000
lambda = 0.01
richness
15
10
5
42
43
44
45
latitude
elevation
500
400
300
200
100

Gradient descent