Regularization

Ivan Corneillet

Data Scientist

GENERAL ASSEMBLY

Learning Objectives

After this lesson, you should be able to:

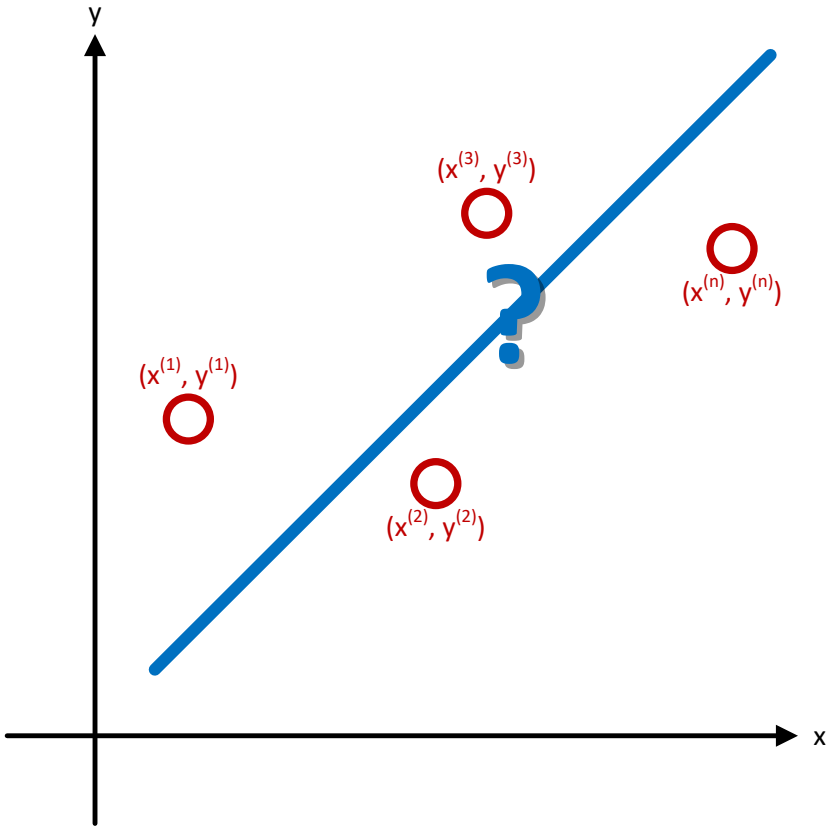
- Understand the closed-form solution of the regression coefficients for linear regression models
- Use ordinary least squares, loss functions, and gradient descent to also derive estimations for the coefficients
- Understand the regularization bias-variance trade-off

Here's what's happening today:

- How to fit a linear regression model on a dataset?
  - Closed-form solution for $\hat{\beta}$
  - Ordinary Least Squares (OLS) and Loss Functions
- Gradient descent
- Regularization

How to fit a linear regression model on a dataset?

How do we estimate $\hat{\beta}$?



Notations

$$y^{(i)} = \sum_{j=0}^{k} \beta_j \cdot x_j^{(i)} + \varepsilon^{(i)} \; \forall i, 1 \leq i \leq n$$

$$y = X \cdot \beta + \varepsilon$$

$$y = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix};\; X = \begin{pmatrix} | & | & & | \\ x_0 & x_1 & \cdots & x_k \\ | & | & & | \end{pmatrix};\; \beta = \begin{pmatrix} | \\ \beta \\ | \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix};\; \varepsilon = \begin{pmatrix} \varepsilon^{(1)} \\ \vdots \\ \varepsilon^{(n)} \end{pmatrix}$$

$$x_0 = \begin{pmatrix} x_0^{(1)} = 1 \\ \vdots \\ x_0^{(n)} = 1 \end{pmatrix};\; x_j = \begin{pmatrix} x_j^{(1)} \\ \vdots \\ x_j^{(n)} \end{pmatrix}$$

Matrix Multiplication ($X \cdot \beta$)

(row i/column j of $X \cdot \beta$ is the dot product of row i of X and column 1 of β)

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}$$

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_k^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \cdots & x_k^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_k^{(n)} \end{pmatrix} \begin{pmatrix} \beta_0 \cdot 1 + \beta_1 \cdot x_1^{(1)} + \cdots + \beta_k \cdot x_k^{(1)} \\ \vdots \\ \beta_0 \cdot 1 + \beta_1 \cdot x_1^{(i)} + \cdots + \beta_k \cdot x_k^{(i)} \\ \vdots \\ \beta_0 \cdot 1 + \beta_1 \cdot x_1^{(n)} + \cdots + \beta_k \cdot x_k^{(n)} \end{pmatrix} = X \cdot \beta$$

$$y = X \cdot \beta + \varepsilon$$

$$\underbrace{\begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \beta_0 + \beta_1 \cdot x_1^{(1)} + \cdots + \beta_k \cdot x_k^{(1)} \\ \vdots \\ \beta_0 + \beta_1 \cdot x_1^{(n)} + \cdots + \beta_k \cdot x_k^{(n)} \end{pmatrix}}_{X \cdot \beta} + \underbrace{\begin{pmatrix} \varepsilon^{(1)} \\ \vdots \\ \varepsilon^{(n)} \end{pmatrix}}_{\varepsilon}$$

How to fit a linear regression model on a dataset?

Closed-form solution for $\hat{\beta}$

Closed-form solution for $\hat{\beta}$ – Take 1

$$y_{train} = X_{train} \cdot \hat{\beta}$$

- We would like to left multiply both sides by X^{-1} and get:

$$X_{train}^{-1} \cdot y_{train} = X_{train}^{-1} \cdot \left(X_{train} \cdot \hat{\beta}\right) = \left(X_{train}^{-1} \cdot X_{train}\right) \cdot \hat{\beta} = \hat{\beta}$$

- However, X_{train} is usually not invertible (it would need to be a square matrix in the first place which would mean having as many features as samples; not good, right?)

$$\hat{\beta} = X_{train}^{-1} \cdot y_{train}$$

Closed-form solution for $\hat{\beta}$ – Take 2

- Let's start over and this time, we left multiply both sides by X_{train}^T:

$$X_{train}^T \cdot y_{train} = X_{train}^T \cdot \left(X_{train} \cdot \hat{\beta}\right) = \left(X_{train}^T \cdot X_{train}\right) \cdot \hat{\beta}$$

- $X^T \cdot X$ is a symmetric matrix and is usually invertible; if not we can slightly reformulate the problem to make it invertible

$$\left(X_{train}^T \cdot X_{train}\right)_{i,j} = \sum_{l=0}^{k} x_l^{(i)} \cdot x_l^{(j)} = \left(X_{train}^T \cdot X_{train}\right)_{j,i}$$

- We can now multiply both sides by $(X_{train}^T \cdot X_{train})^{-1}$:

$$\left(X_{train}^T \cdot X_{train}\right)^{-1} \cdot X^T \cdot y_{train} = \left(X_{train}^T \cdot X_{train}\right)^{-1} \cdot \left(\left(X_{train}^T \cdot X_{train}\right) \cdot \hat{\beta}\right)$$

$$= \left(\left(\left(X_{train}^T \cdot X_{train}\right)^{-1}\right) \cdot \left(\left(X_{train}^T \cdot X_{train}\right)\right)\right) \cdot \hat{\beta} = \hat{\beta}$$

Closed-form solution for $\hat{\beta}$ – Take 2 (cont.)

$$\hat{\beta} = \left(X_{train}^{T} \cdot X_{train}\right)^{-1} \cdot X_{train}^{T} \cdot y_{train}$$

$$\hat{y} = X_{predict} \cdot \hat{\beta}$$

Closed-form solution for $\hat{\beta}$ – Take 2 (cont.)

- Was the matrix X_{train}^{T} the only matrix possible we could use for the left-multiply operation?
  - No. But it will because clear in the next section

Activity | Closed-form solution for $\hat{\beta} = (\hat{\beta}_0)$

EXERCISE

DIRECTIONS (10 minutes)

1. Using the matrix closed form solution for $\hat{\beta}$, calculate the following special case when only the intercept $\hat{\beta}_0$ is estimated (i.e., no feature from the samples are included)

$$X_{train} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix};\ \hat{\beta} = (\hat{\beta}_0);\ y_{train} = \begin{pmatrix} y_{train}^{(1)} \\ \vdots \\ y_{train}^{(n)} \end{pmatrix}$$

2. When finished, share your answers with your table

DELIVERABLE

Answers to the above questions

Activity | Closed-form solution for $\hat{\beta} = (\hat{\beta}_0)$ (cont.)

EXERCISE

- Let's calculate $X_{train}^T \cdot X_{train}$:

$$X_{train} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$X_{train}^T = (1 \quad \cdots \quad 1) \quad (n) = X_{train}^T \cdot X_{train}$$

- So $(X_{train}^T \cdot X_{train})^{-1} = \left(\frac{1}{n}\right)$, and:

$$X_{train}^T = (1 \quad \cdots \quad 1) \qquad y_{train} = \begin{pmatrix} y_{train}^{(1)} \\ \vdots \\ y_{train}^{(n)} \end{pmatrix}$$

$$(X_{train}^T \cdot X_{train})^{-1} = \left(\frac{1}{n}\right) \qquad \left(\frac{1}{n} \quad \cdots \quad \frac{1}{n}\right) \qquad \left(\frac{y_{train}^{(1)} + \cdots + y_{train}^{(n)}}{n} = \bar{y}_{train}\right) = \hat{\beta} = (\hat{\beta}_0)$$

- Therefore

$$\boldsymbol{\hat{\beta}_0 = \bar{y}_{train}}$$

Closed-form solution for $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$

$$X_{train} = \begin{pmatrix} 1 & x_{train}^{(1)} \\ \vdots & \vdots \\ 1 & x_{train}^{(n)} \end{pmatrix};\ \hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}\ ;\ y_{train} = \begin{pmatrix} y_{train}^{(1)} \\ \vdots \\ y_{train}^{(n)} \end{pmatrix}$$

Closed-form solution for $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$: $X_{train}^T \cdot X_{train}$

$$X_{train} = \begin{pmatrix} 1 & x_{train}^{(1)} \\ \vdots & \vdots \\ 1 & x_{train}^{(n)} \end{pmatrix}$$

$$X_{train}^T = \begin{pmatrix} 1 & \cdots & 1 \\ x_{train}^{(1)} & \cdots & x_{train}^{(n)} \end{pmatrix} \quad \begin{pmatrix} n & s_x \\ s_x & s_{x^2} \end{pmatrix} = X_{train}^T \cdot X_{train}$$

$$\text{with } s_x = \sum_{i=1}^{n} x_{train}^{(i)} \text{ and } s_{x^2} = \sum_{i=1}^{n} \left(x_{train}^{(i)} \right)^2$$

Closed-form solution for $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$: $\left(X_{train}^T \cdot X_{train}\right)^{-1}$

$$= \begin{pmatrix} n & s_x \\ s_x & s_{x^2} \end{pmatrix}$$

$$\left(X_{train}^T \cdot X_{train}\right)^{-1} = \begin{pmatrix} u & v \\ w & z \end{pmatrix} \quad \begin{pmatrix} n \cdot u + s_x \cdot v = 1\ (1) & s_x \cdot u + s_{x^2} \cdot v = 0\ (2) \\ n \cdot w + s_x \cdot z = 0\ (3) & s_x \cdot w + s_{x^2} \cdot z = 1\ (4) \end{pmatrix}$$

$$s_{x^2} \times (1) - s_x \times (2) \Rightarrow \left(n \cdot s_{x^2} - (s_x)^2\right) \cdot u = s_{x^2} \Rightarrow u = \frac{s_{x^2}}{n \cdot s_{x^2} - (s_x)^2}$$

$$n \times (2) - s_x \times (1) \Rightarrow \left(n \cdot s_{x^2} - (s_x)^2\right) \cdot v = -s_x \Rightarrow v = \frac{-s_x}{n \cdot s_{x^2} - (s_x)^2}$$

$$s_{x^2} \times (3) - s_x \times (4) \Rightarrow \left(n \cdot s_{x^2} - (s_x)^2\right) \cdot w = -s_x \Rightarrow u = \frac{-s_x}{n \cdot s_{x^2} - (s_x)^2}$$

$$n \times (4) - s_x \times (3) \Rightarrow \left(s_{x^2} - (s_x)^2\right) \cdot z = n \Rightarrow u = \frac{n}{n \cdot s_{x^2} - (s_x)^2}$$

$$\left(X_{train}^T \cdot X_{train}\right)^{-1} = \frac{1}{n \cdot s_{x^2} - (s_x)^2} \cdot \begin{pmatrix} s_{x^2} & -s_x \\ -s_x & n \end{pmatrix}$$

Closed-form solution for $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$ (cont.)

$$X_{train}^T = \begin{pmatrix} \cdots & 1 & \cdots \\ \cdots & x_{train}^{(j)} & \cdots \end{pmatrix} \qquad y_{train} = \begin{pmatrix} \vdots \\ y_{train}^{(j)} \\ \vdots \end{pmatrix}$$

$$\left(X_{train}^T \cdot X_{train}\right)^{-1} = \frac{1}{n \cdot s_{x^2} - (s_x)^2} \cdot \begin{pmatrix} s_{x^2} & -s_x \\ -s_x & n \end{pmatrix}$$

$$\frac{1}{n \cdot s_{x^2} - (s_x)^2} \cdot \begin{pmatrix} \cdots & s_{x^2} - s_x \cdot x_{train}^{(j)} & \cdots \\ \cdots & n \cdot x_{train}^{(j)} - s_x & \cdots \end{pmatrix}$$

$$\frac{1}{n \cdot s_{x^2} - (s_x)^2} \cdot \begin{pmatrix} s_{x^2} \cdot s_y - s_x \cdot s_{xy} \\ n \cdot s_{xy} - s_x \cdot s_y \end{pmatrix} = \hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$$

var and cov

$$var(x) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \left(x^{(i)} - \underbrace{\bar{x}}_{s_x/n} \right)^2 = \frac{s_{x^2} - \frac{(s_x)^2}{n}}{n-1}$$

$$cov(x,y) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \left(x^{(i)} - \underbrace{\bar{x}}_{s_x/n} \right) \cdot \left(y^{(i)} - \underbrace{\bar{y}}_{s_y/n} \right) = \frac{s_{xy} - \frac{s_x \cdot s_y}{n}}{n-1}$$

Closed–form solution for $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$ (cont.)

- Therefore,

$$\hat{\beta}_1 = \frac{n \cdot s_{xy} - s_x \cdot s_y}{n \cdot s_{x^2} - (s_x)^2} = \frac{s_{xy} - \frac{s_x \cdot s_y}{n}}{s_{x^2} - \frac{(s_x)^2}{n}} = \frac{cov(x,y)}{var(x)}$$

- And

$$\hat{\beta}_0 = \frac{s_{x^2} \cdot s_y - s_x \cdot s_{xy}}{n \cdot s_{x^2} - (s_x)^2} = \frac{\frac{s_y}{n} \cdot s_{x^2} - \frac{s_x}{n} \cdot s_{xy}}{s_{x^2} - \frac{(s_x)^2}{n}} = \frac{\overbrace{\frac{s_y}{n}}^{\bar{y}} \cdot \left(s_{x^2} - \frac{(s_x)^2}{n} \right) - \overbrace{\frac{s_x}{n}}^{\bar{x}} \cdot \overbrace{\left(s_{xy} - \frac{s_x \cdot s_y}{n} \right)}^{cov(x,y)}}{s_{x^2} - \frac{(s_x)^2}{n}} = \bar{y} - \bar{x}\hat{\beta}_1$$

Closed-form solution for $\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}$ (cont.)

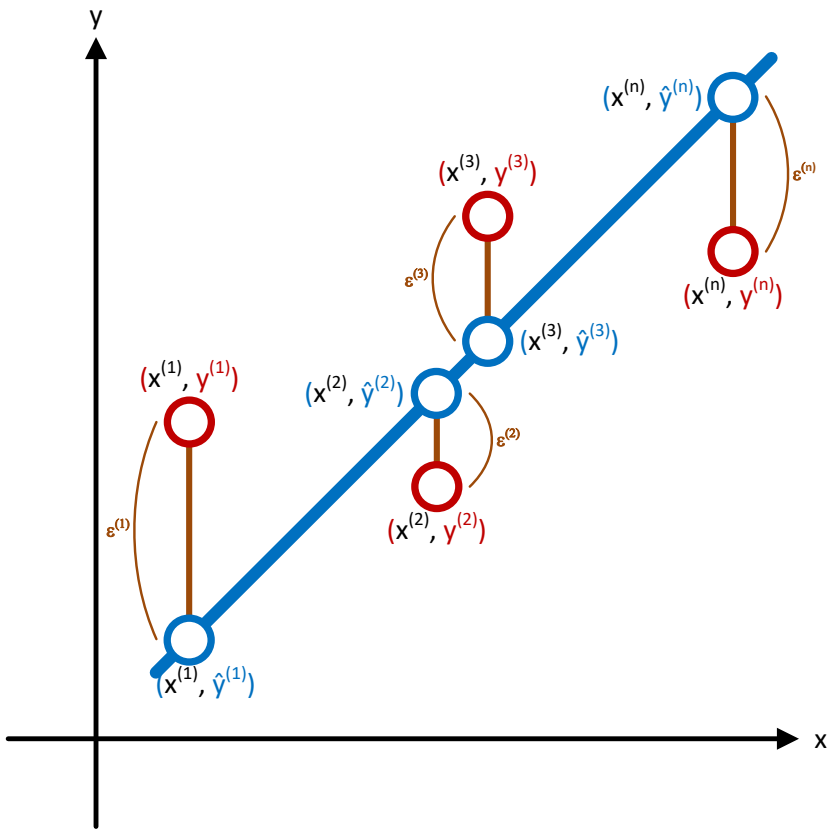
$$\hat{\beta}_1 = \frac{cov(x_{train}, y_{train})}{var(x_{train})}$$

$$\hat{\beta}_0 = \bar{y}_{train} - \bar{x}_{train} \cdot \hat{\beta}_1$$

How to fit a linear regression model on a dataset?

Ordinary Least Squares (OLS) and Loss Functions

We can also estimate $\hat{\beta}_0$ and $\hat{\beta}_1$ with Ordinary Least Squares



- Hypothesis

$$y = \beta_0 + \beta_1 \cdot x$$

- Parameters

$$\beta_0, \beta_1$$

- Goal

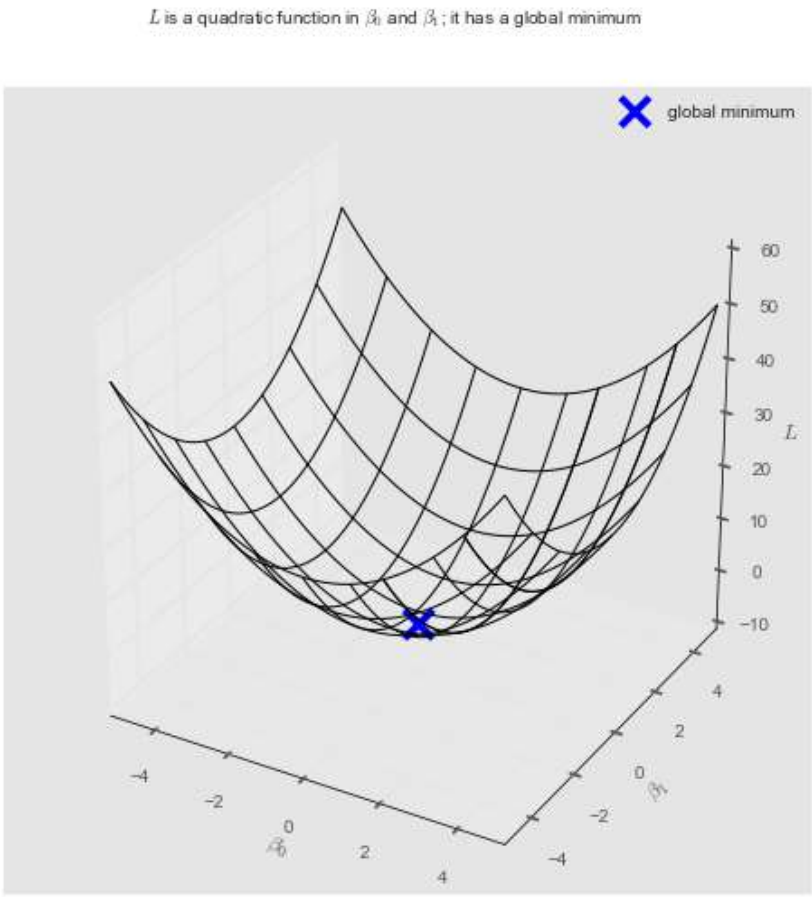
$$\min_{\beta_0,\beta_1} \underbrace{\sum_{i=1}^{n} \left(y^{(i)} - y\left(x^{(i)}\right)\right)^2}_{L(\beta_0,\beta_1)}$$

(i.e., minimizing the least square errors)

$L\left(y^{(i)} - y\left(x^{(i)}\right)\right)$ is a quadratic function in β_0 and β_1 in the form

$$A\beta_0^2 + B\beta_0\beta_1 + C\beta_1^2 + D\beta_0 + E\beta_1 + F$$

(A, B, C, D, E, and F constant)



Activity | Loss function $L(\beta_0)$ and its global minimum $\hat{\beta}_0$

EXERCISE

DIRECTIONS (10 minutes)

Consider the following linear regression:

- Hypothesis: $y = \beta_0$
- Parameters: β_0, β_1
- Goal : $\min_{\beta_0} \underbrace{\sum_{i=1}^{n} \left(y^{(i)} - y\left(x^{(i)}\right)\right)^2}_{L(\beta_0)}$

1. Calculate the loss function $L(\beta_0)$
2. For which value(s) of β_0 does $L(\beta_0)$ reaches a global minimum?
3. When finished, share your answers with your table

DELIVERABLE

Answers to the above questions

Activity | Loss function $L(\beta_0)$ and its global minimum $\hat{\beta}_0$ (cont.)

EXERCISE

- Let's calculate $L(\beta_0)$:

$$L(\beta_0) = \sum_{i=1}^{n} \left(y^{(i)} - \underbrace{y(x^{(i)})}_{\beta_0} \right)^2 = s_{y^2} - 2 \cdot \beta_0 \cdot s_y + n \cdot \beta_0^2 = n \cdot \left(\beta_0 - \underbrace{\frac{s_y}{n}}_{\bar{y}} \right)^2 - n \cdot \left(\frac{s_y}{n} \right)^2 + s_{y^2}$$

$$= n \cdot (\beta_0 - \bar{y})^2 + \underbrace{s_{y^2} - \frac{(s_y)^2}{n}}_{constant}$$

- $L(\beta_0)$ reaches a global minimal when $\beta_0 - \bar{y} = 0$, therefore

$$\boldsymbol{\hat{\beta}_0 = \bar{y}_{train}}$$

Loss function $L(\beta_0, \beta_1)$ and its global minimum $\left(\hat{\beta}_0, \hat{\beta}_1\right)$

$$L(\beta_0, \beta_1) = \sum_{i=1}^{n} \left(y^{(i)} - y\left(x^{(i)}\right)\right)^2 = \sum_{i=1}^{n} \left(\beta_0 + \beta_1 \cdot x^{(i)} - y^{(i)}\right)^2$$

Partial derivatives $\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_0}$ and $\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_1}$

$$\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_0}$$

$$= 2\sum_{i=1}^{n}\left(\beta_0 + \beta_1 \cdot x^{(i)} - y^{(i)}\right)$$

$$= 2\left(n \cdot \beta_0 + s_x \cdot \beta_1 - s_y\right)$$

$$= 2n \cdot (\beta_0 + \bar{x} \cdot \beta_1 - \bar{y})$$

$$\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_1}$$

$$= 2\sum_{i=1}^{n} x^{(i)}\left(\beta_0 + \beta_1 \cdot x^{(i)} - y^{(i)}\right)$$

$$= 2\left(s_x \cdot \beta_0 + s_{x^2} \cdot \beta_1 - s_{xy}\right)$$

The global minimum $\left(\hat{\beta}_0, \hat{\beta}_1\right)$ is at $\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_0} = 0$ and $\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_1} = 0$

$$\frac{\partial L(\beta_0, \beta_1)}{\partial \beta_0} = 0 \Rightarrow n \cdot \beta_0 + s_x \cdot \beta_1 = s_y \ (1)$$

$$\frac{\partial L(\beta_0, \beta_1)}{\partial \beta_1} = 0 \Rightarrow s_x \cdot \beta_0 + s_{x^2} \cdot \beta_1 = s_{xy} (2)$$

- $n \cdot (2) - s_x \cdot (1) \Rightarrow$

$$\left(n \cdot s_{x^2} - (s_x)^2\right)\beta_1 = n \cdot s_{xy} - s_x \cdot s_y$$

$$\hat{\beta}_1 = \frac{s_{xy} - \frac{s_x \cdot s_y}{n}}{s_{x^2} - \frac{(s_x)^2}{n}} = \frac{cov(x,y)}{var(x)}$$

- $(1) \Rightarrow$

$$\hat{\beta}_0 = \bar{y} - \bar{x}\hat{\beta}_1$$

The global minimum is at $\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_0} = 0$ and $\frac{\partial L(\beta_0,\beta_1)}{\partial \beta_1} = 0$ (cont.)

$$\hat{\beta}_1 = \frac{cov(x_{train}, y_{train})}{var(x_{train})}$$

$$\hat{\beta}_0 = \bar{y}_{train} - \bar{x}_{train} \cdot \hat{\beta}_1$$

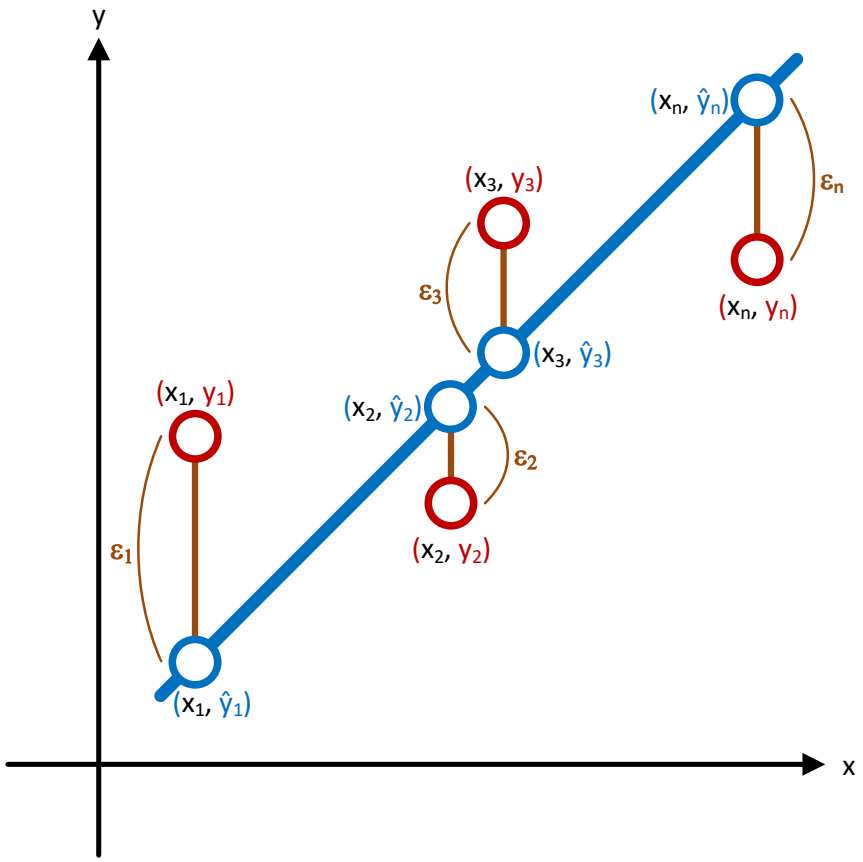
How to fit a linear regression model on a dataset?

Ordinary Least Squares and the closed-form solution for $\hat{\beta}$

Ordinary Least Squares and the closed-form for $\hat{\beta}$

- Modeling just an intercept ($y = \beta_0$) or an intercept and a slope ($y = \beta_0 + \beta_1 \cdot x$) yield the same results for both the closed-form solution for $\hat{\beta}$ and the Ordinary Least Squares
  - In fact, minimizing $L(\beta) = (y - X \cdot \beta)^T \cdot (y - X \cdot \beta)$ in the general case yields our previous closed-form solution for $\hat{\beta} = \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot y$

We can estimate $\hat{\beta}$ with Ordinary Least Squares (cont.)



- Hypothesis

$$y = X \cdot \beta$$

- Parameters

$$\beta$$

- Goal

$$\min_{\beta} (y - X \cdot \beta)^T \cdot (y - X \cdot \beta)$$

- Assuming X has full column rank, β has a closed-form solution

$$\hat{\beta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

$$L(\beta) = (X \cdot \beta - y)^T \cdot (X \cdot \beta - y)$$

$$L(\beta) = \underbrace{(X \cdot \beta - y)^T}_{\substack{(X\cdot\beta)^T - y^T \\ \beta^T \cdot X^T - y^T}} \cdot (X \cdot \beta - y)$$

$$= \beta^T \cdot X^T \cdot X \cdot \beta - \underbrace{\beta^T \cdot X^T \cdot y}_{y^T \cdot X \cdot \beta} - y^T \cdot X \cdot \beta + y^T \cdot y$$

$$= \beta^T \cdot \left(X^T \cdot X\right) \cdot \beta - 2\left(y^T \cdot X\right) \cdot \beta + y^T \cdot y$$

The global minimum is at $\frac{\partial L(\beta)}{\partial \beta} = 0$

$$\frac{\partial}{\partial \beta}\left(\beta^T \cdot \left(X^T \cdot X\right) \cdot \beta\right) = 2\beta^T \cdot \left(X^T \cdot X\right)$$

($\frac{\partial}{\partial y}\left(x^T \cdot A \cdot x\right) = 2x^T \cdot A \cdot \frac{\partial x}{\partial y}$ if A is a symmetric matrix)

$$\frac{\partial}{\partial \beta}\left(\left(y^T \cdot X\right) \cdot \beta\right) = y^T \cdot X$$

($\frac{\partial}{\partial y}\left(A \cdot x\right) = A \cdot \frac{\partial x}{\partial y}$)

$$\frac{\partial L(\beta)}{\partial \beta} = 2\left(\beta^T \cdot \left(X^T \cdot X\right) - y^T \cdot X\right)$$

$$\frac{\partial L(\beta)}{\partial \beta} = 0 \Rightarrow \beta^T \cdot \left(X^T \cdot X\right) = y^T \cdot X$$

$$\left(X^T \cdot X\right) \cdot \beta = X^T \cdot y$$

(transpose)

$$\boldsymbol{\hat{\beta} = \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot y}$$

Ordinary Least Squares and Maximum Likelihood Estimation

- $Y^{(i)} = X^{(i)} \cdot \beta + E^{(i)}$
- If, $Y^{(i)}$ random variables realizing $y^{(i)}$,
  - $X^{(i)}$ are given number, not random variables,
  - $E^{(i)} \sim N(0, \sigma^2)$ and i.i.d
- Then
  - $E^{(i)} \sim N(X^{(i)} \cdot \beta, \sigma^2)$, i.i.d
- With a likelihood function of the form:

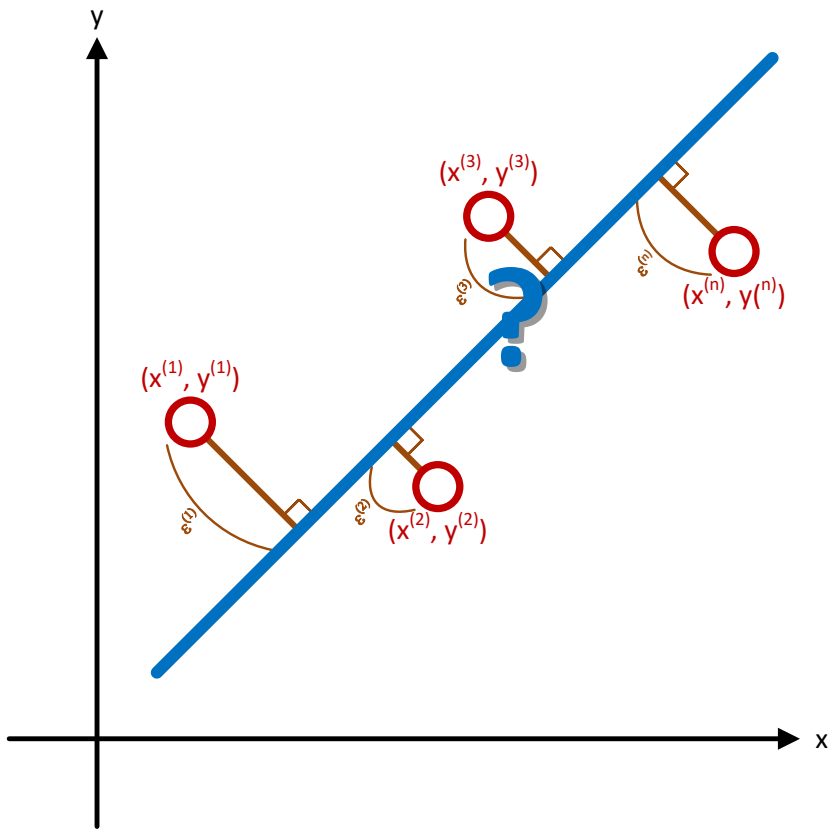
$$l_Y(y, \beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y^{(i)} - X^{(i)} \cdot \beta\right)^2}{2\sigma^2}}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y^{(i)} - X^{(i)} \cdot \beta\right)^2} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{L(\beta)}{2\sigma^2}}$$

  - Maximizing the likelihood function is equivalent to minimizing the loss function

There are many way to fit a line…

- It can be shown that $\hat{y}$ is unbiased for y
  - I.e., $E[\hat{y}] = y$
- E.g., if $\hat{\beta} = (\hat{\beta}_0)$, then $y = \bar{y}$ is unbiased
  - In the SF housing dataset, if you don't anything about the houses (e.g., size, etc…) then the best estimation of the sale price you can give is the mean of the training set's sale price



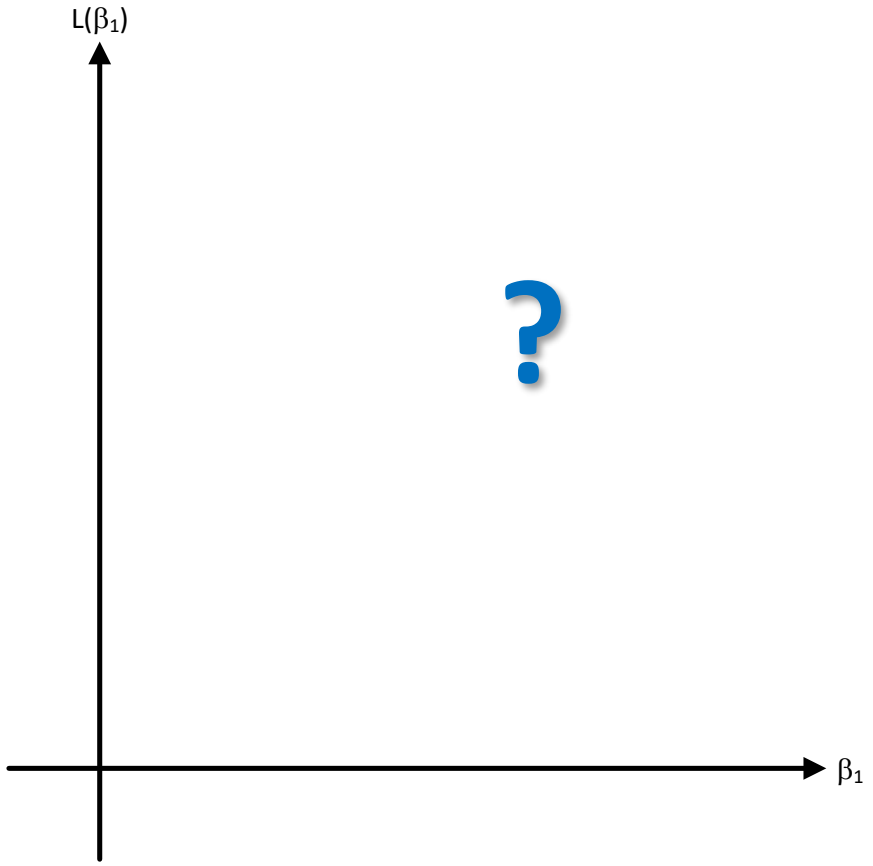
Gradient Descent

What's the shape of the loss function L and what's the optimal $\hat{\beta}_0$ and $\hat{\beta}_1$? For the time being, let's set $\hat{\beta}_0$ to .1551, draw L as a function of $\hat{\beta}_1$ only, and find the optimal $\hat{\beta}_1$

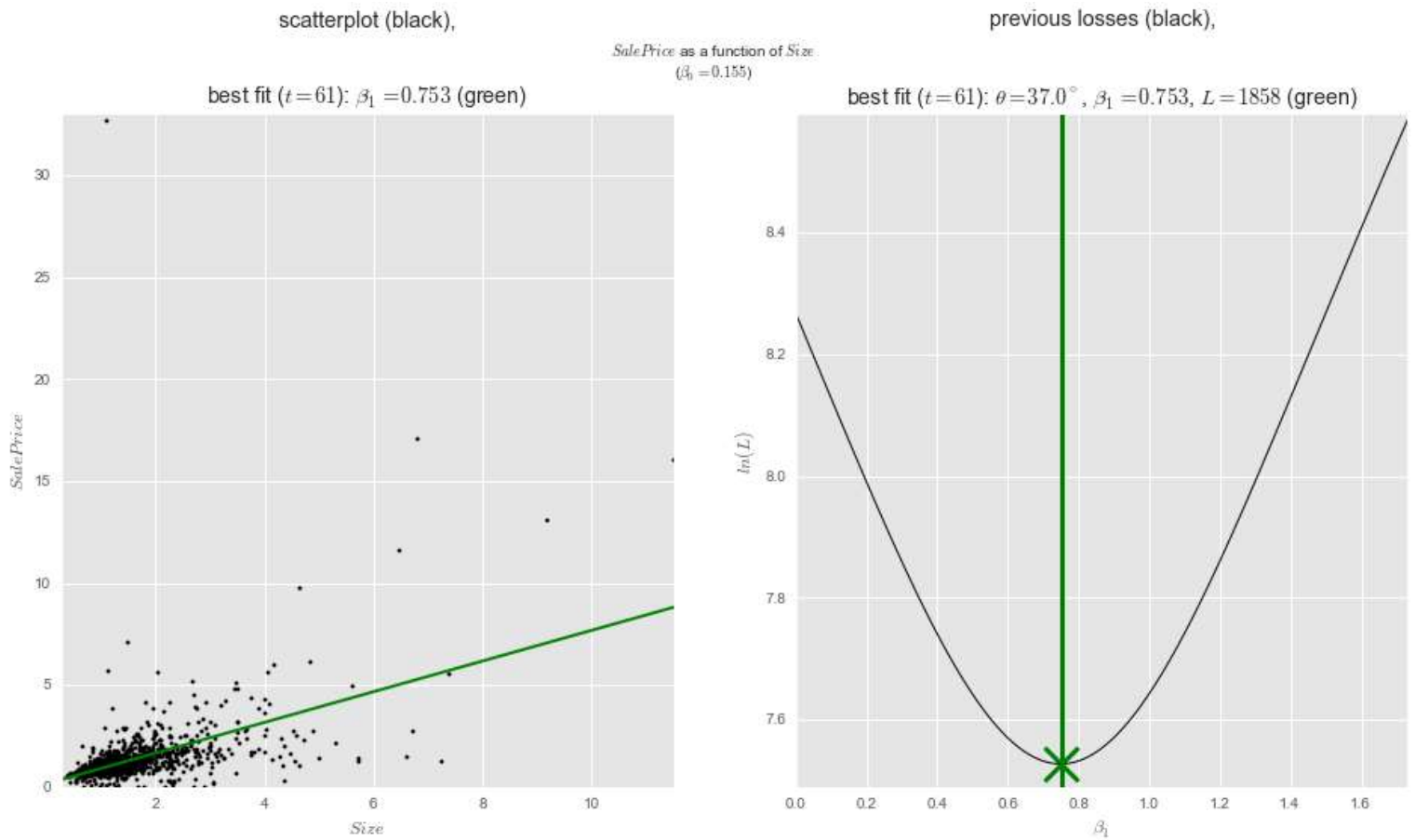
Dep. Variable:	SalePrice	**R-squared:**	0.236
Model:	OLS	**Adj. R-squared:**	0.235
Method:	Least Squares	**F-statistic:**	297.4
Date:		**Prob (F-statistic):**	2.67e-58
Time:		**Log-Likelihood:**	-1687.9
No. Observations:	967	**AIC:**	3380.
Df Residuals:	965	**BIC:**	3390.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
Intercept	0.1551	0.084	1.842	0.066	-0.010 0.320
Size	0.7497	0.043	17.246	0.000	0.664 0.835

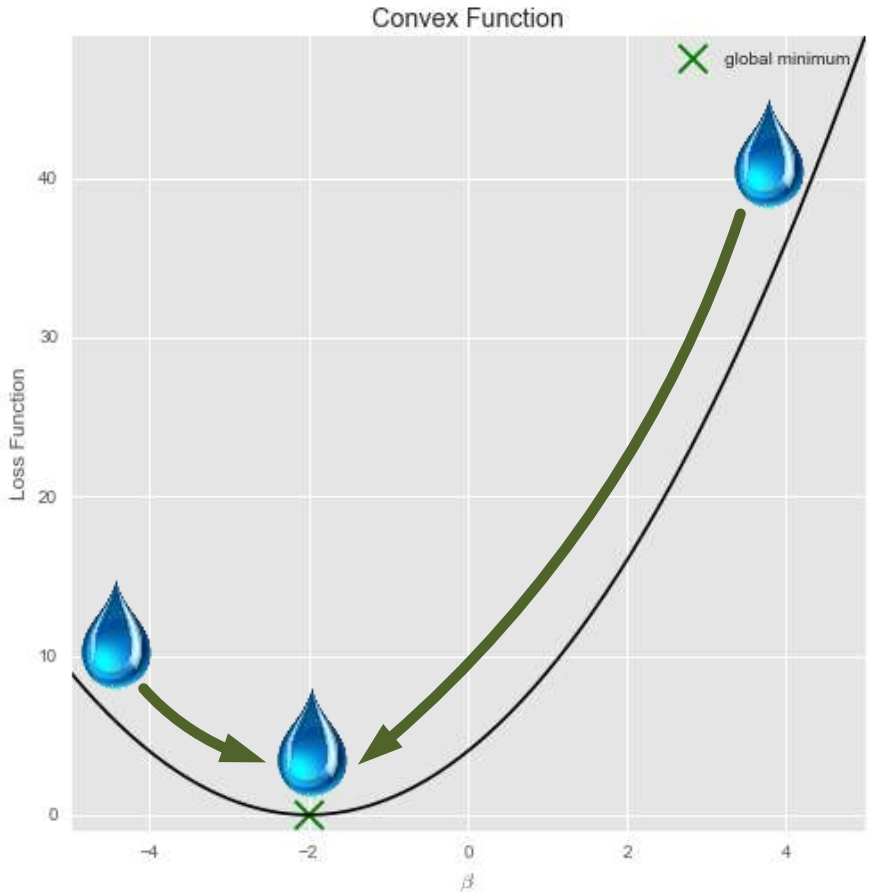
Omnibus:	1842.865	**Durbin-Watson:**	1.704
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3398350.943
Skew:	13.502	**Prob(JB):**	0.00
Kurtosis:	292.162	**Cond. No.**	4.40



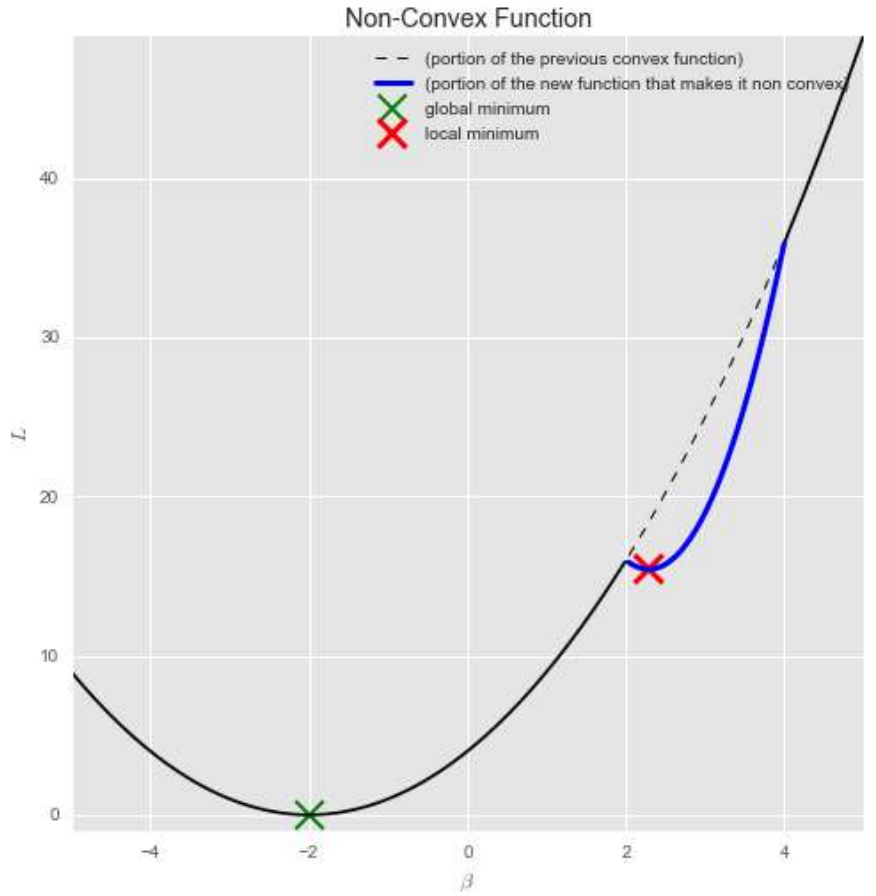
Demo | On the left, the best fitted line with $\hat{\beta}_1 = .753$. On the right $L(\beta_1)$



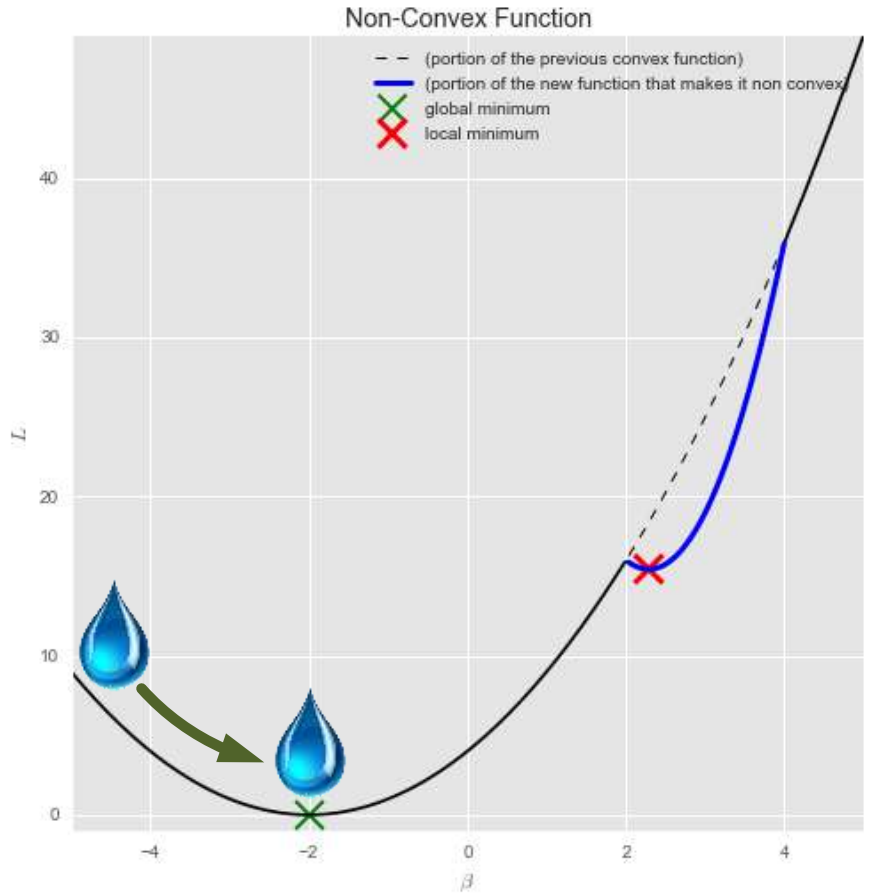
$L(\beta_1)$ is a convex function: It reaches a minimum value only once (global minimum) and following the gravity (greatest slope), a drop of water placed anywhere along the curve would descent towards that minimum

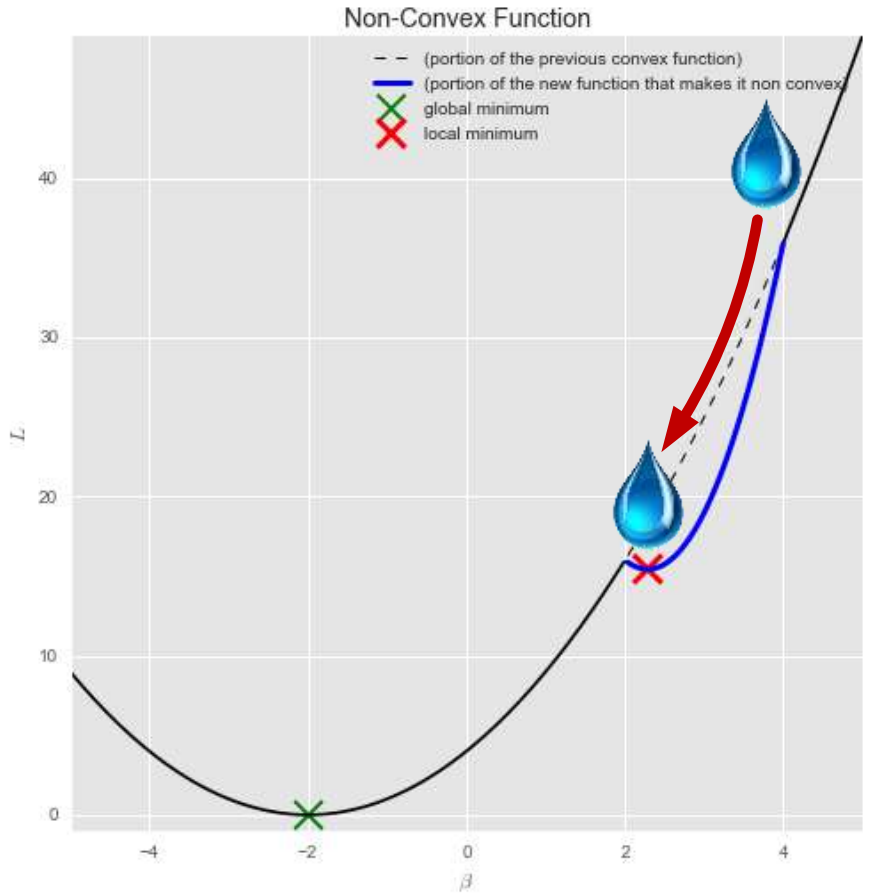


In contrast, here's an example of non-convex function: The function has a global minimum but also another (local) minimum



The drops of water can go down to the global minimum or get stuck in a local minimum





Reaching the global minimum following the greatest slope is the idea behind gradient descent (assuming a convex function)

- Goal

$$\min_{\beta_1} L(\beta_1)$$

(i.e., minimizing the least squares)

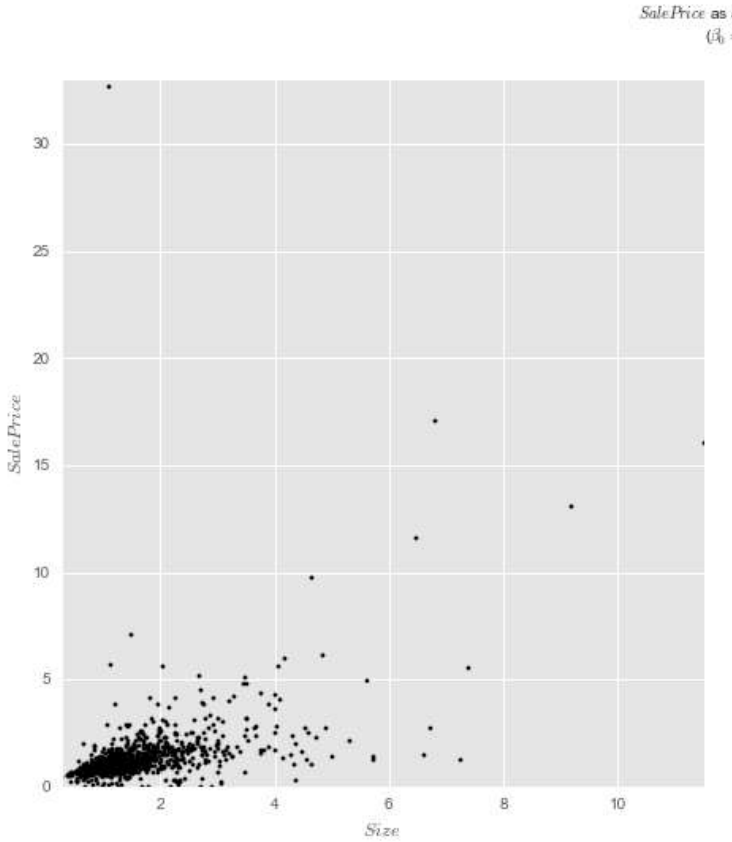
- Gradient Descent Algorithm
  - Start with some β_1, e.g., $\beta_1 = 0$
  - Repeat until convergence

$$\beta_1 := \beta_1 - \alpha \underbrace{\frac{\partial}{\partial \beta_1} L(\beta_1)}_{\frac{1}{m}\sum_{i=1}^{m} x_i \cdot \underbrace{(\beta_0 + \beta_1 \cdot x_i - y_i)}_{-\varepsilon_i}} = \beta_1 + \frac{\alpha}{m} \sum_{i=1}^{m} \varepsilon_i \cdot x_i$$

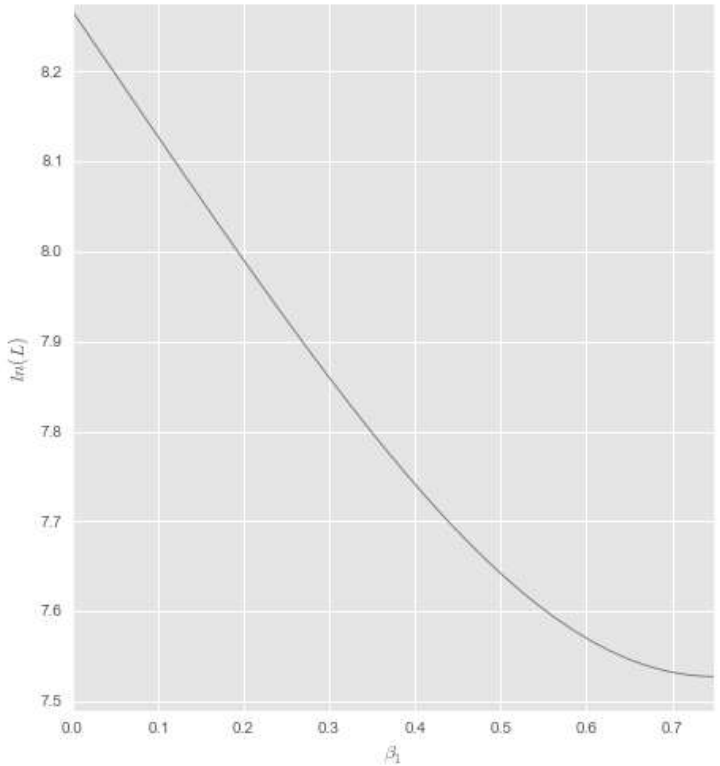
```
def y_hat(beta_0_hat, beta_1_hat, x):
    return beta_0_hat + beta_1_hat * x

def L(y_hat):
    return sum((train_y - y_hat) ** 2)

beta_1_hat = 0

for _ in range(n):
    beta_1_hat += alpha *\
        ((train_y -\
            y_hat(beta_0_hat, beta_1_hat,
                train_x)) * train_x).mean()
```

Demo | On the left, the scatterplot of our data and the fitted lines at different angles θ ($\beta_1 = \tan(\theta)$). On the right $L(\beta_1)$. Note: we show the shape of the loss function but the gradient descent algorithm doesn't know it…

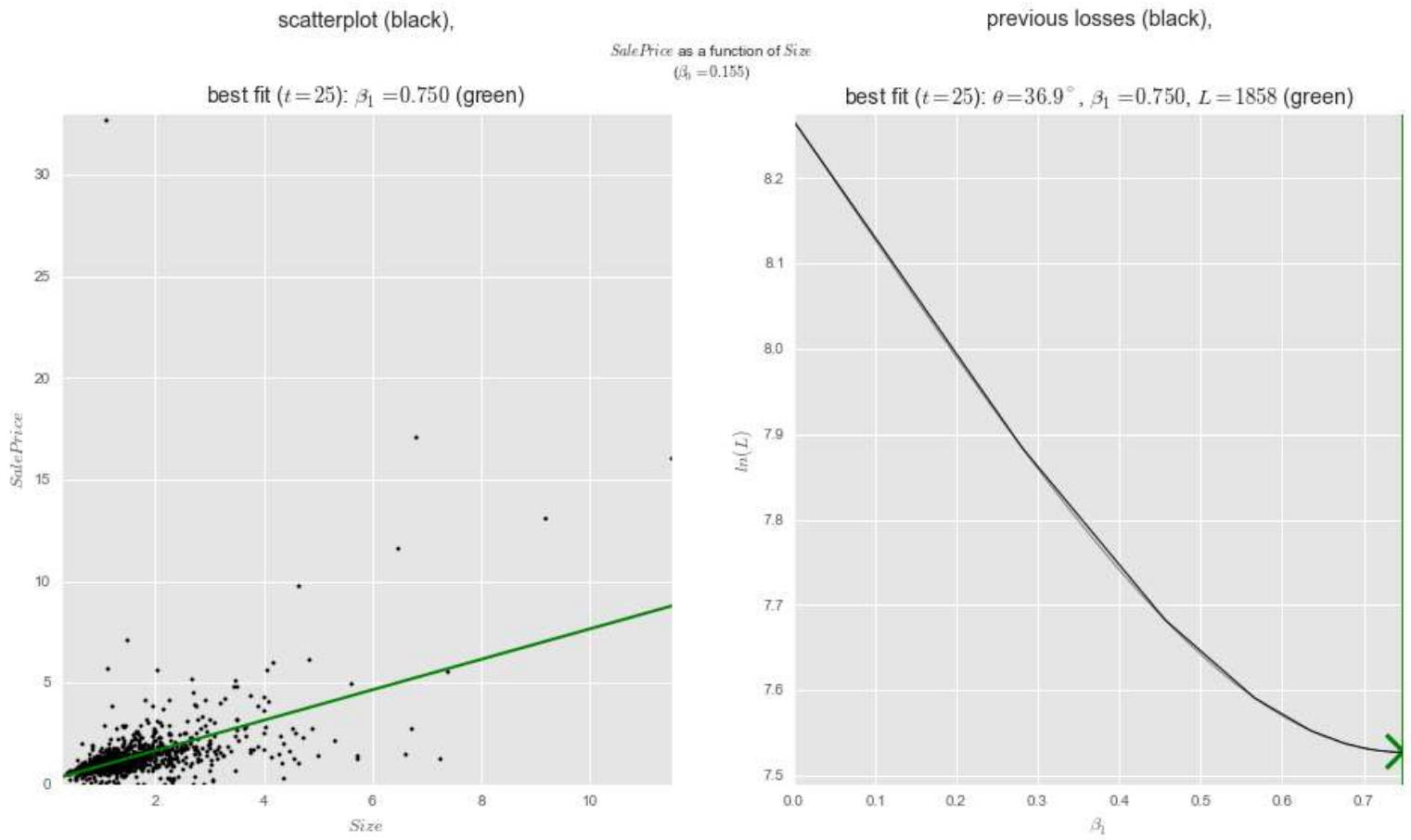
scatterplot (black)



L (loss function)

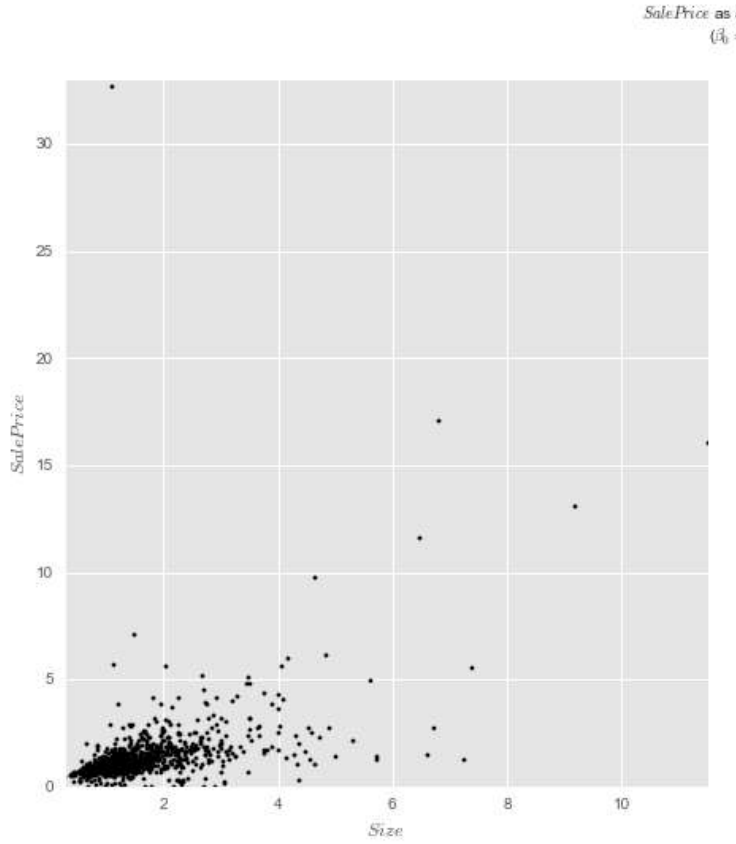


Demo | On the left, the best fitted line with $\hat{\beta}_1 = .750$. On the right the points of $L(\beta_1)$ following the gradient descent

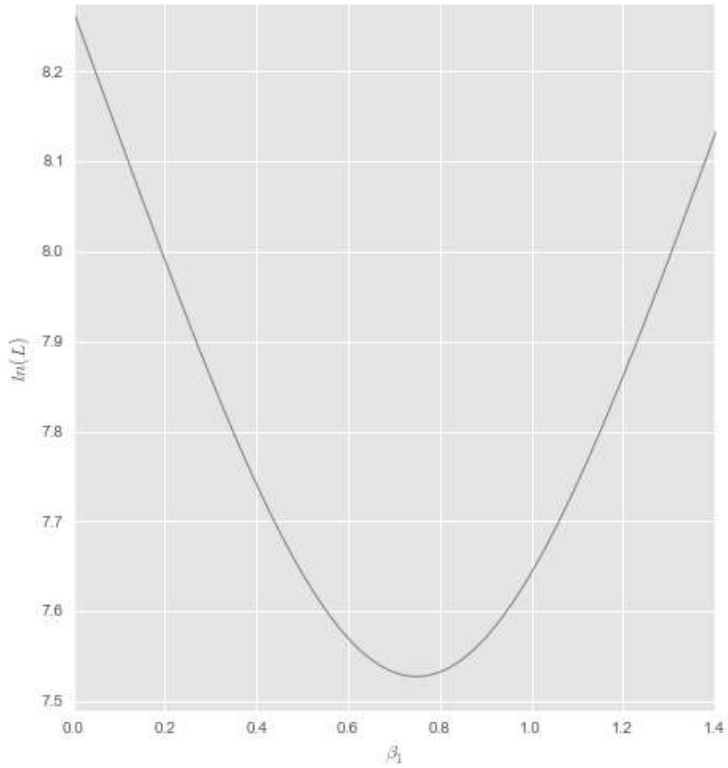


Demo | α is the learning rate. In the previous example it was set to .25. The idea is to only incorporate a fraction of the learning. But what happens if α is too high? Let's try $\alpha = .5$…

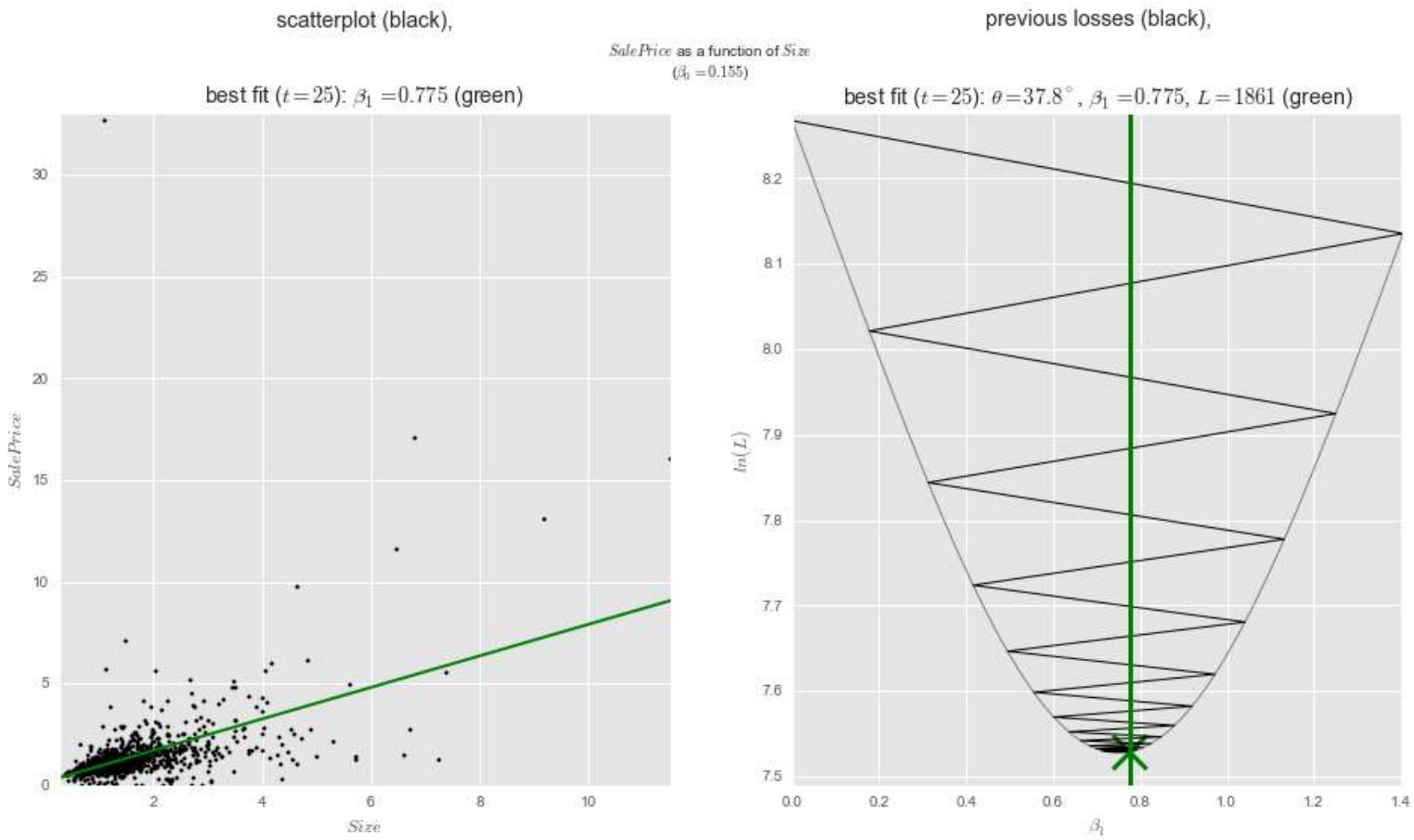
scatterplot (black)



L (loss function)

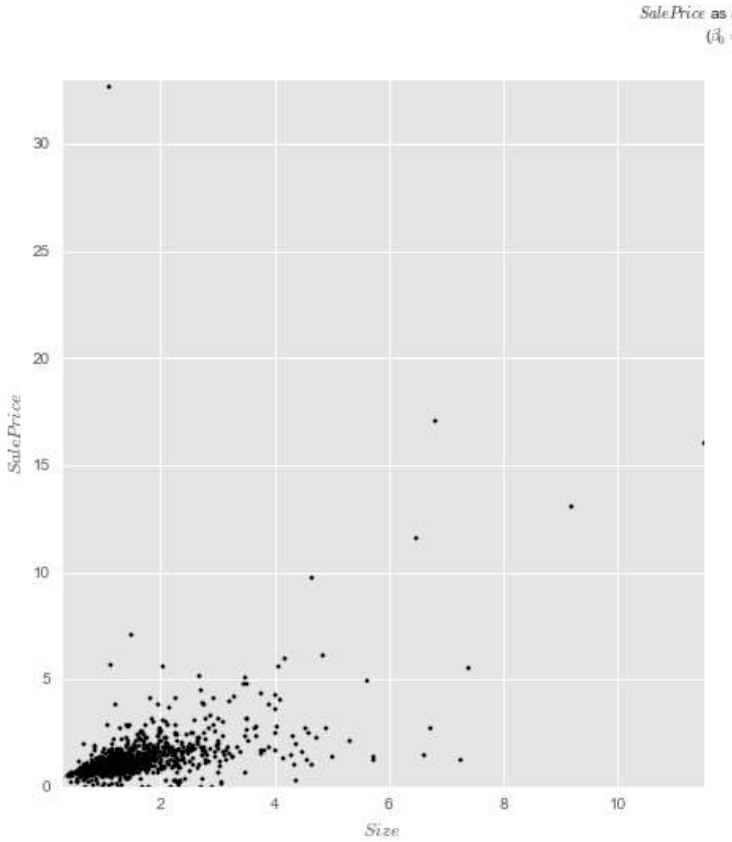


Demo | $t = 25$

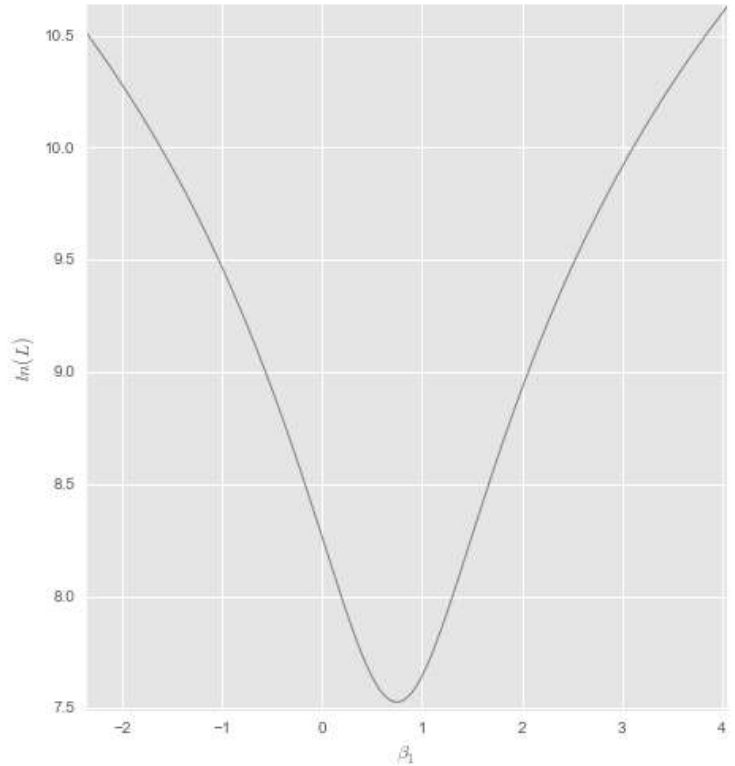

scatterplot (black),
$SalePrice$ as a function of $Size$
($\beta_0 = 0.155$)
best fit ($t = 25$): $\beta_1 = 0.775$ (green)
$SalePrice$
$Size$
previous losses (black),
best fit ($t = 25$): $\theta = 37.8^\circ$, $\beta_1 = 0.775$, $L = 1861$ (green)
$ln(L)$
β_1

Demo | Again with $\alpha = .55$…

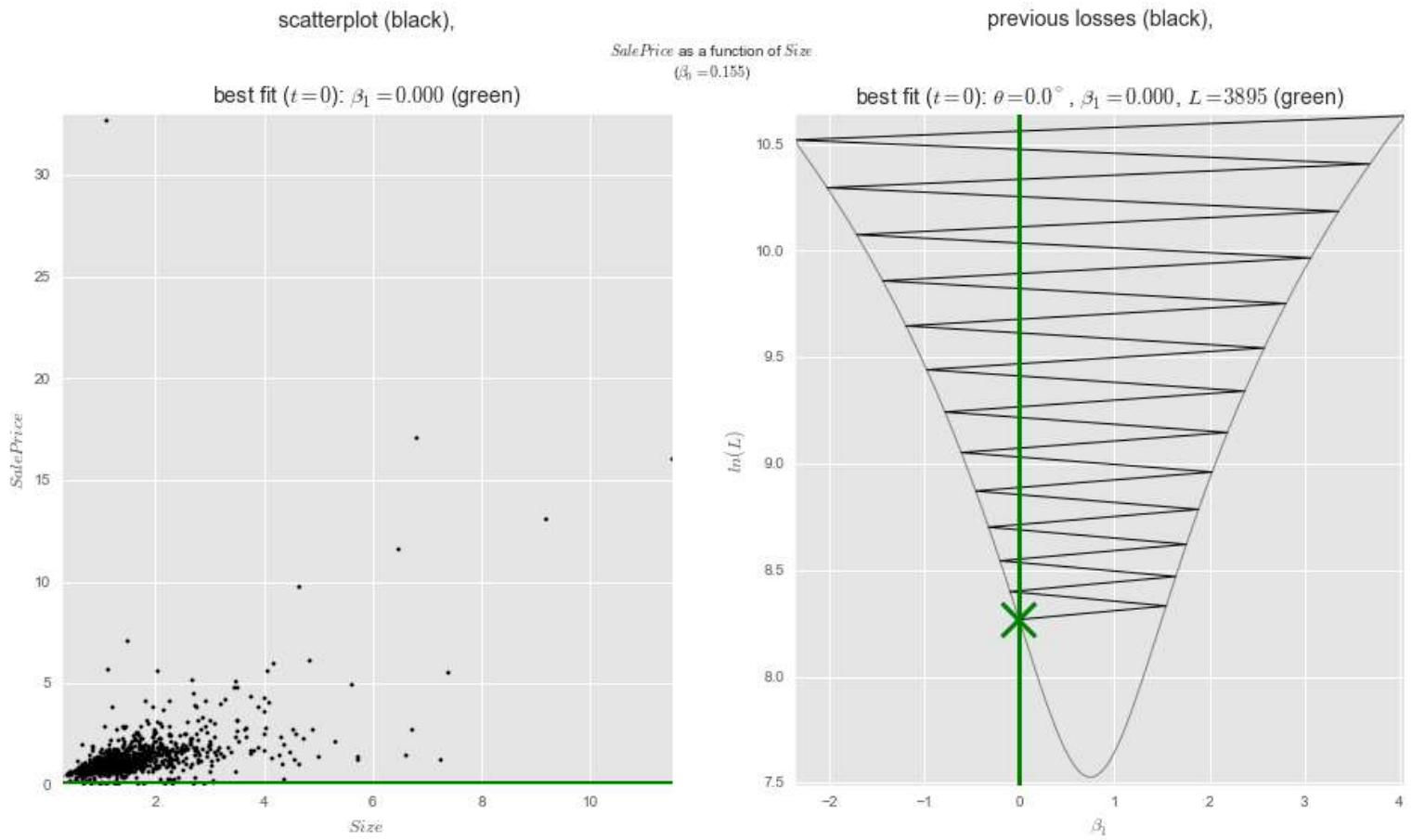
scatterplot (black)



L (loss function)



Demo | $t = 25$


scatterplot (black),
$SalePrice$ as a function of $Size$
($\beta_1 = 0.155$)
best fit ($t = 0$): $\beta_1 = 0.000$ (green)
$SalePrice$
$Size$
previous losses (black),
best fit ($t = 0$): $\theta = 0.0^\circ$, $\beta_1 = 0.000$, $L = 3895$ (green)
$ln(L)$
β_1

We can generalize the gradient function for more than one variable as long as the function to optimize is convex

- Goal

$$\min_{\beta_0,\beta_1} L(\beta_0,\beta_1)$$

(i.e., minimizing the least squares)

- Gradient Descent Algorithm
  - Start with some β_0 and β_1
  - Repeat until convergence

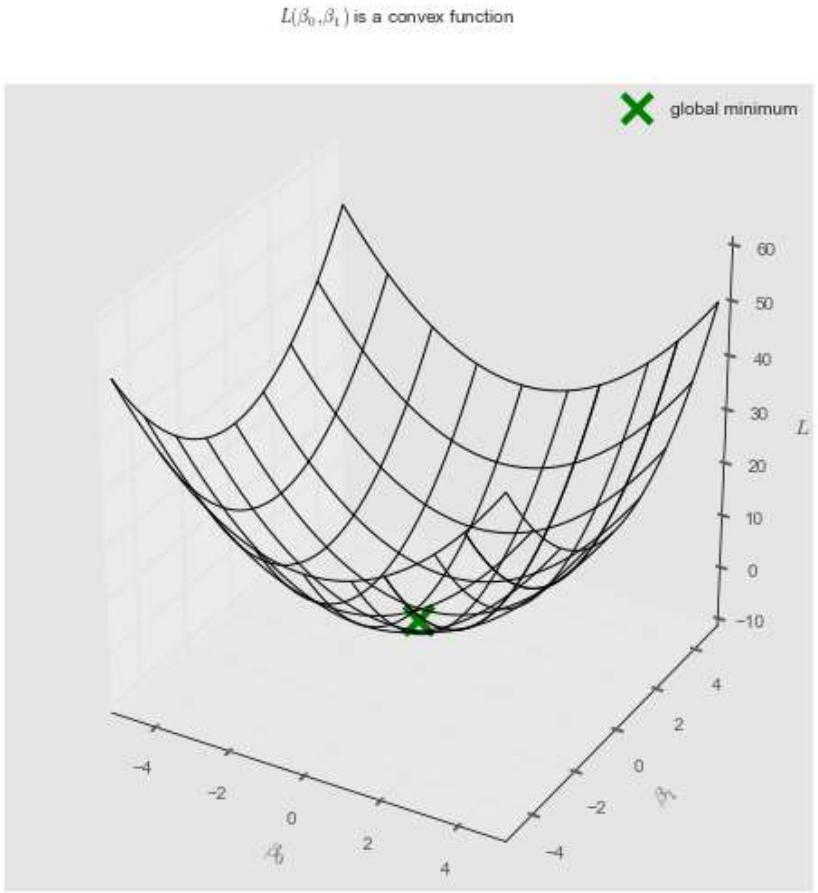
$$\beta_0 := \beta_0 - \alpha \underbrace{\frac{\partial}{\partial\beta_0} L(\beta_0,\beta_1)}_{\frac{1}{m}\sum_{i=1}^{m} 1\cdot\underbrace{(\beta_0+\beta_1\cdot x_i-y_i)}_{-\varepsilon_i}} = \beta_0 + \frac{\alpha}{m}\sum_{i=1}^{m}\varepsilon_i$$

$$\beta_1 := \beta_1 - \alpha \underbrace{\frac{\partial}{\partial\beta_1} L(\beta_0,\beta_1)}_{\frac{1}{m}\sum_{i=1}^{m} x_i\cdot\underbrace{(\beta_0+\beta_1\cdot x_i-y_i)}_{-\varepsilon_i}} = \beta_1 + \frac{\alpha}{m}\sum_{i=1}^{m}\varepsilon_i\cdot x_i$$

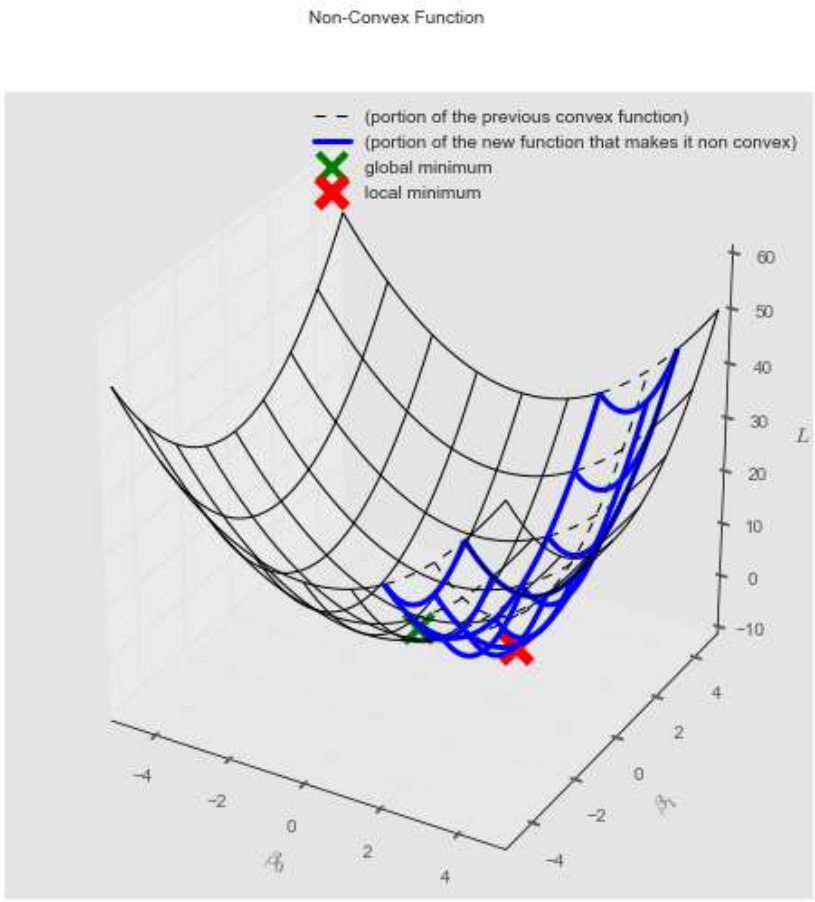
Here's some code to estimate $\hat{\beta}_0$ and $\hat{\beta}_1$. Note that the δs need to be all computed together, i.e., before updating $\hat{\beta}_0$ or $\hat{\beta}_1$

```
beta_0_hat = 0
beta_1_hat = 0

for _ in range(n):
    beta_0_hat_delta = alpha *\
        (y - y_hat(beta_0_hat, beta_1_hat, x)).mean()
    beta_1_hat_delta = alpha *\
        ((y - y_hat(beta_0_hat, beta_1_hat, x)) * x).mean()

    beta_0_hat += beta_0_hat_delta
    beta_1_hat += beta_1_hat_delta
```

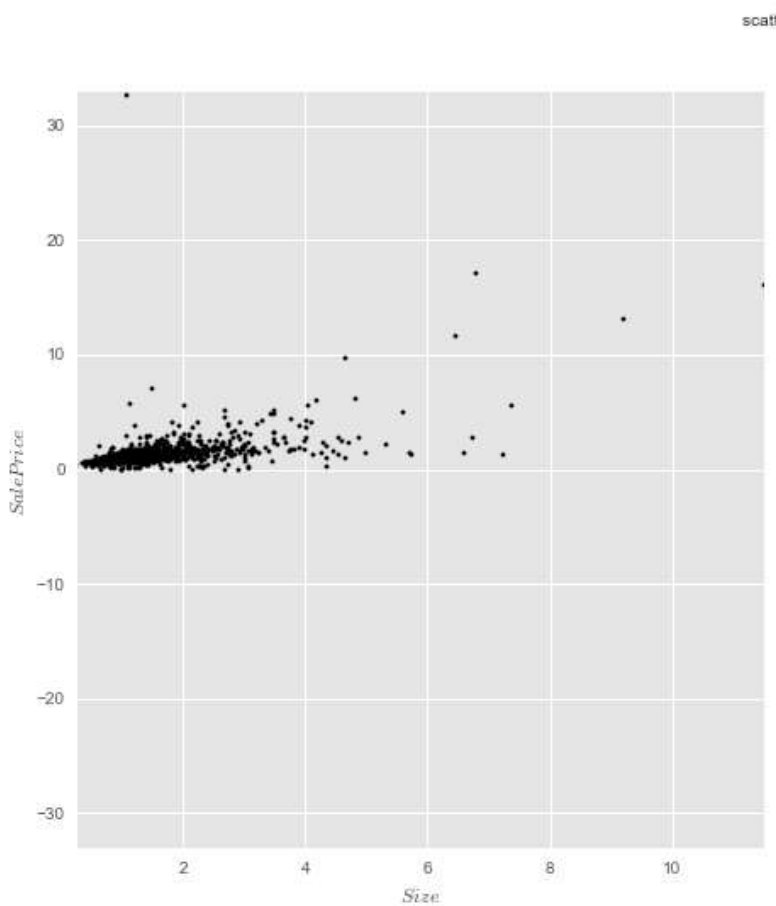
$L(\beta_0, \beta_1)$ is a convex function so we can use gradient descent to estimate $\hat{\beta}_0$ and $\hat{\beta}_1$

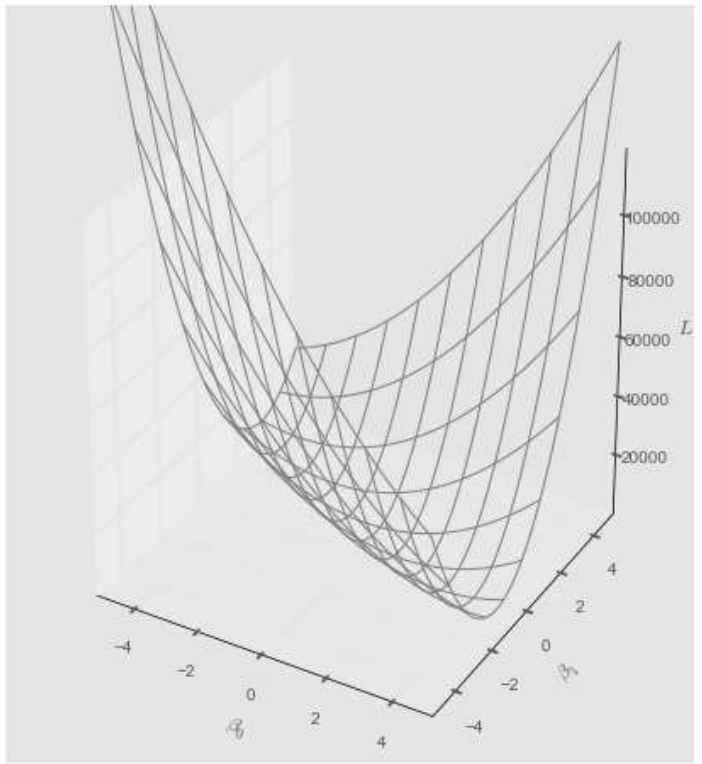


On the other hand, here's a function that is not convex

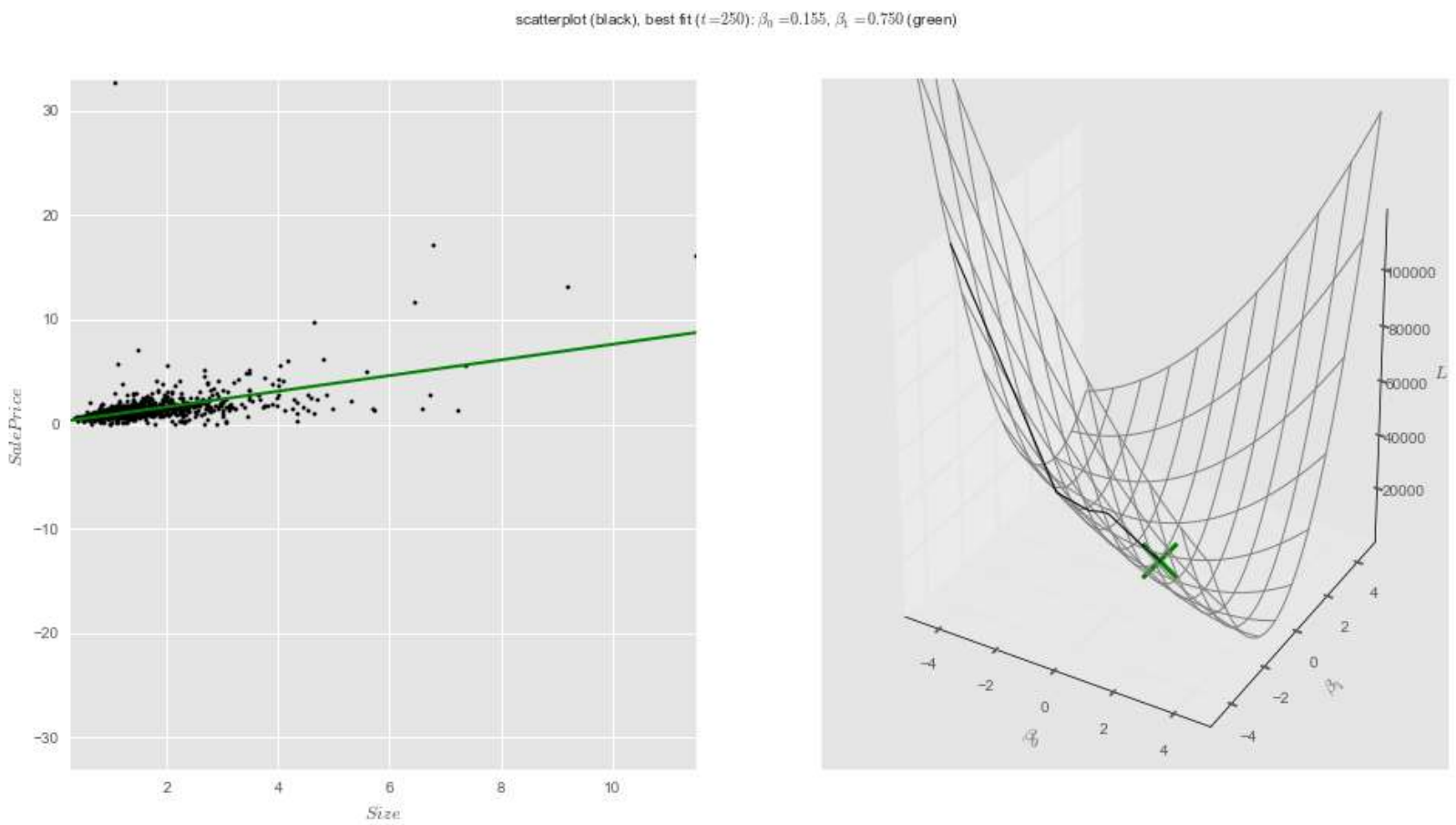


Demo | Our familiar graphs but with $L(\beta_0, \beta_1)$ on the right instead of $L(\beta_1)$



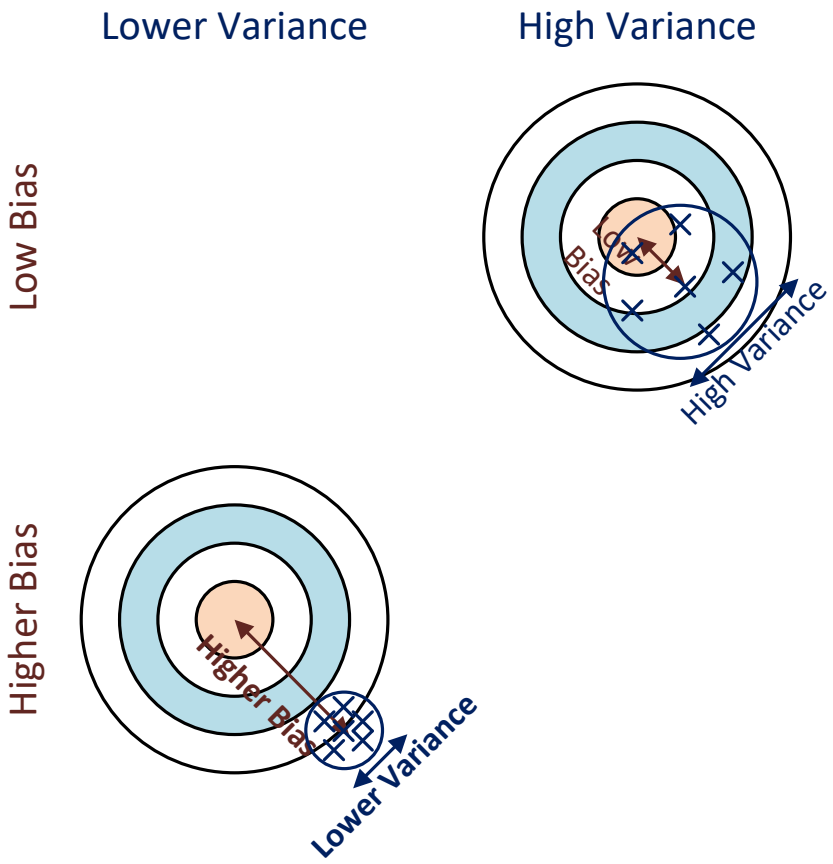


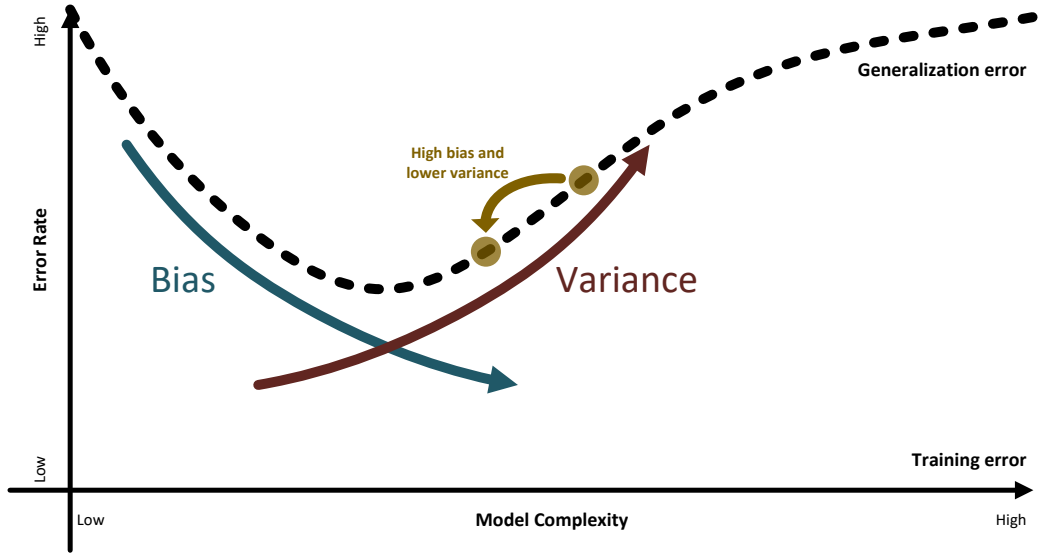
Demo | On the left, the best fitted line with $\hat{\beta}_0 = .155$ and $\hat{\beta}_1 = .750$. On the right the points of $L(\beta_0, \beta_1)$ following the gradient descent



Regularization

OLS yields unbiased estimators at the cost of high variance. Can we trade some (higher) bias for lower variance and get ahead on the bias-variance trade-of?





Revisiting complexity

- E.g., as a function of the size of the coefficients
  - $\|\beta\|_p = \left(\sum_{j=0}^{k} |\beta_j|^p\right)^{1/p}$ (Lp-norm)
  - $\|\beta\|_1 = \sum_{j=0}^{k} |\beta_j|$ (L1-norm)
  - $\|\beta\|_2 = \left(\sum_{j=0}^{k} |\beta_j|^2\right)^{1/2}$ (L2-norm)

Regularization prevents overfitting by explicitly controlling model complexity

- These definitions of complexity lead to the following regularization techniques
  - $min\left(\underbrace{\|y - x \cdot \beta\|^2}_{OLS\ term} + \underbrace{\lambda\|\beta\|_1}_{regularization\ term}\right)$ (Lasso regularization using the L1 norm)
  - $min\left(\|y - x \cdot \beta\|^2 + \lambda\|\beta\|_2^2\right)$ (Ridge regularization using the L2 norm)
    - (note that in the loss function the term β_0 isn't regularized and is in fact excluded from the norm here)
- This formulation reflects the fact that there is a cost associated with regularization that we want to minimize

A word about loss functions

- Loss functions are a powerful tool to optimize the fit of machine learning algorithms
- Loss functions are not limited to linear regression- and regularization-based models. E.g. training a logistic regression algorithm (while also leveraging linear regression) is also modeled and fitted with loss functions