LINEAR REGRESSION (A BEST-FIT LINE PERSPECTIVE)

Inzamam Rahaman

REGRESSION

- Suppose that we have dataset $\mathscr{D} = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$ where $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$

- x_i are our independent variable and y_i is our dependent

- Regression is the problem of learning a mapping (model) $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x_i) \approx y_i$

REGRESSION

- Suppose that we have dataset $\mathscr{D} = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$ where $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$

- Regression is the problem of learning a mapping (model) $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x_i) \approx y_i$

mapping is a "generally good" mapping

REGRESSION

- Suppose that we have dataset $\mathscr{D} = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$ where $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$

- Regression is the problem of learning a mapping (model) $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x_i) \approx y_i$

mapping is a "generally good" mapping

What does it mean for a mapping to be "generally good"?

LINEAR REGRESSION

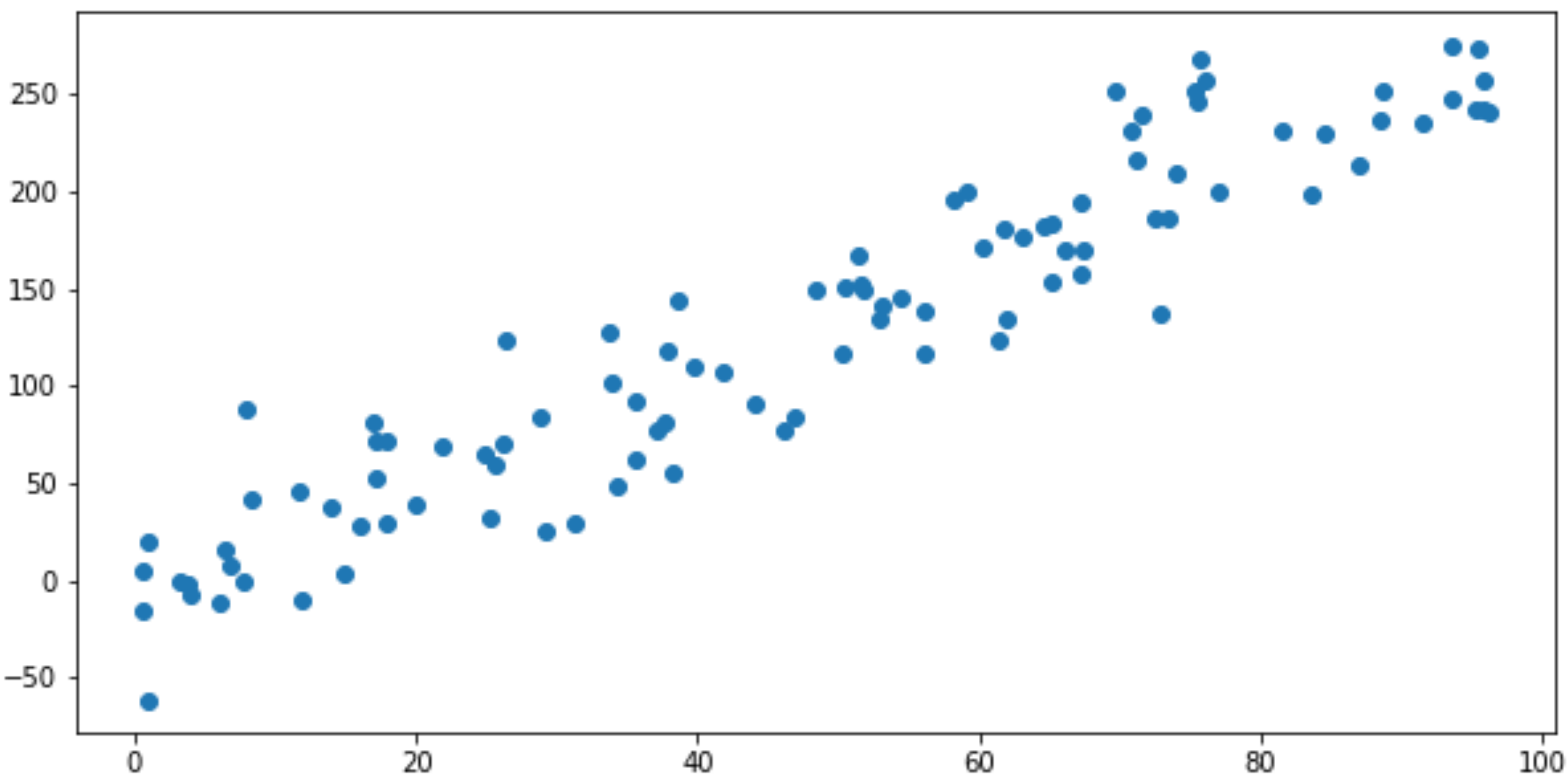
- Space of mappings is infinite!
- Need to devise a structure for our mappings
- Linear regression: restrict our search to a linear function of $\mathbb{R}^n$

LINEAR REGRESSION

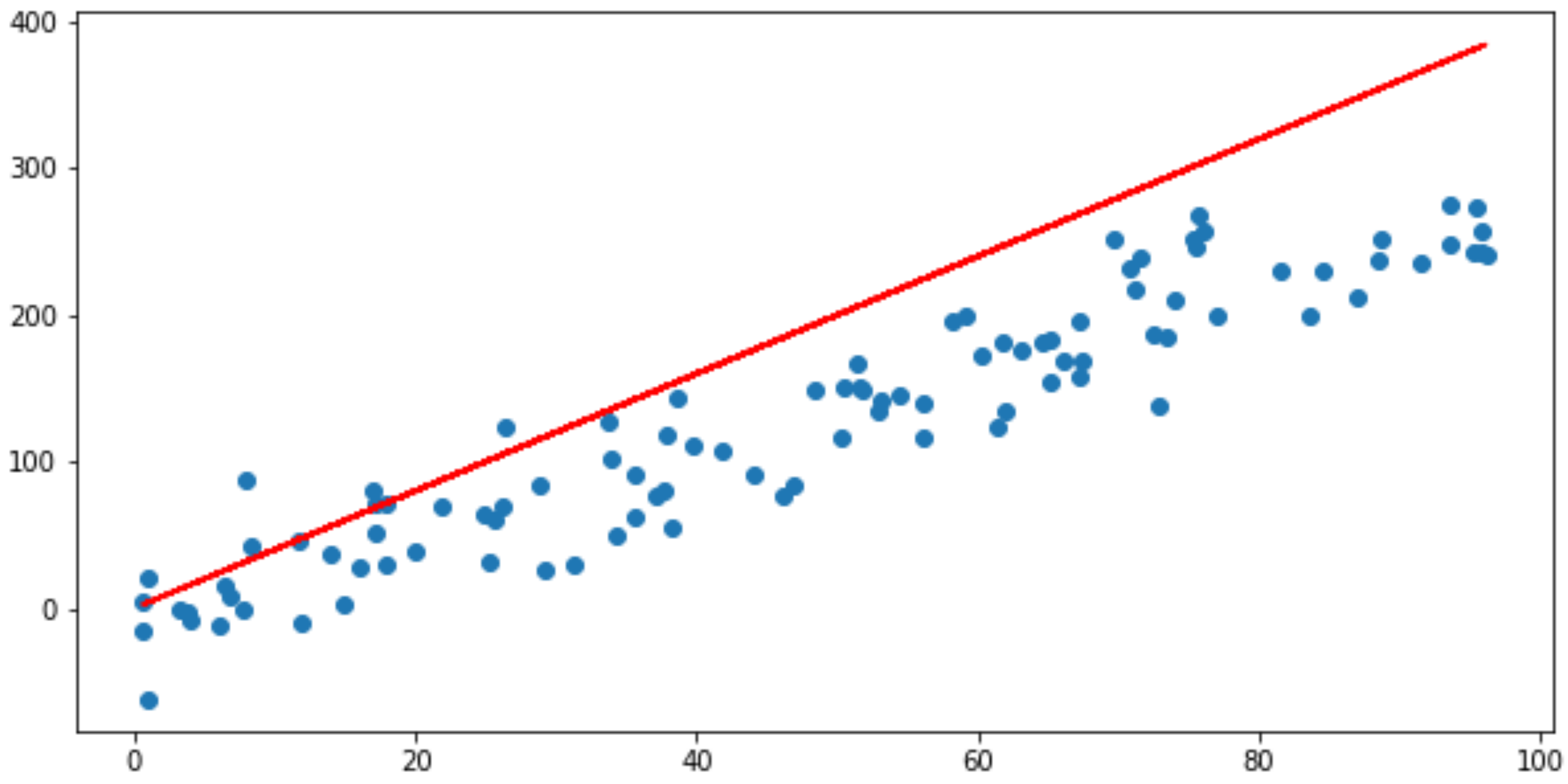
- Linear functions are well understood and malleable
- Assume straight line relationship between input and output
    - think $y = mx + c$
- Applies limitations on model (LR is high-bias low-variance machine learning model family)

LINEAR REGRESSION

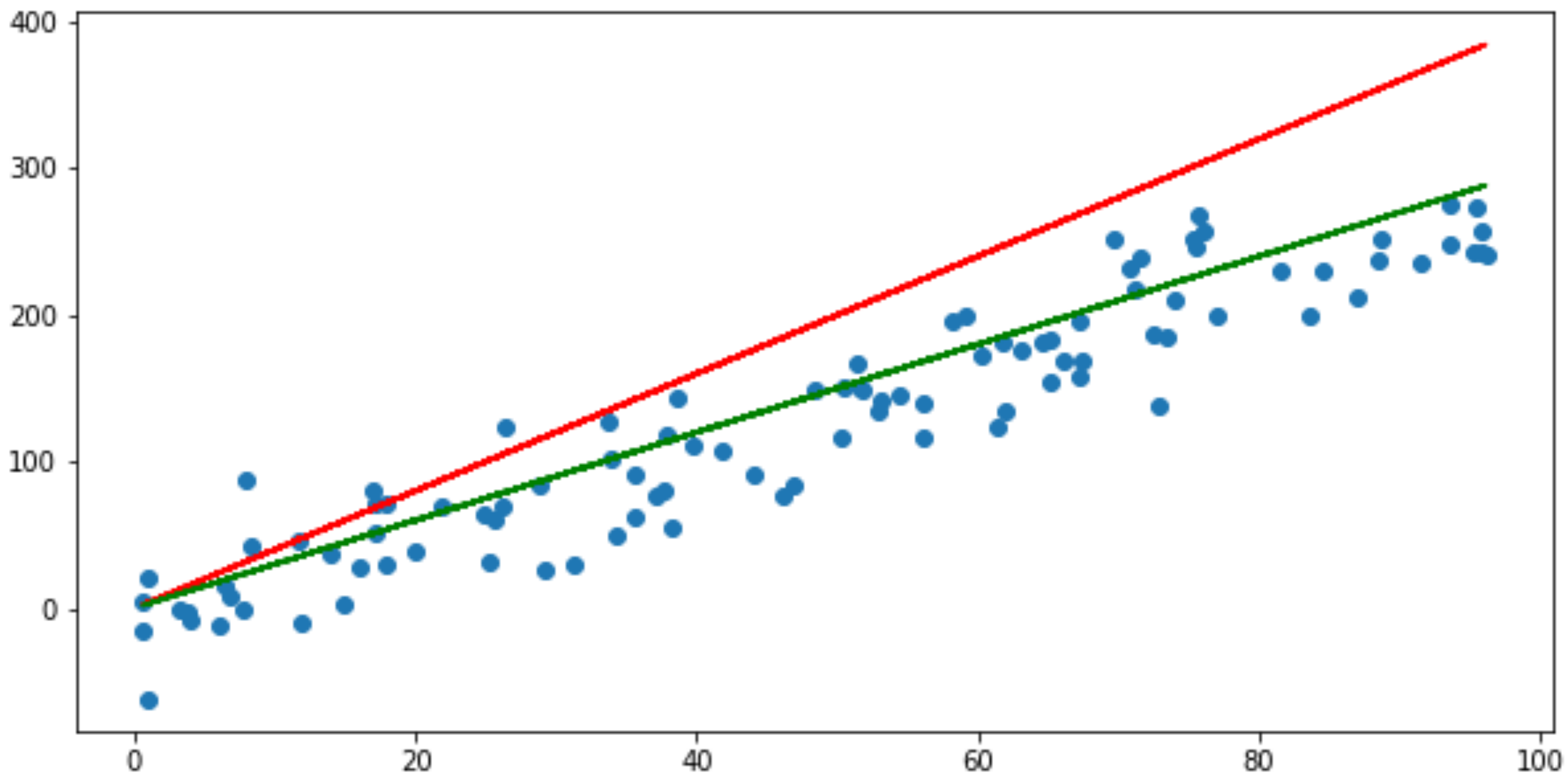
- Because linear, maths scales well to many dimensions
- Will focus on the case of 2 dimensions and assume $c = 0$
  - But maths is trivially extensible



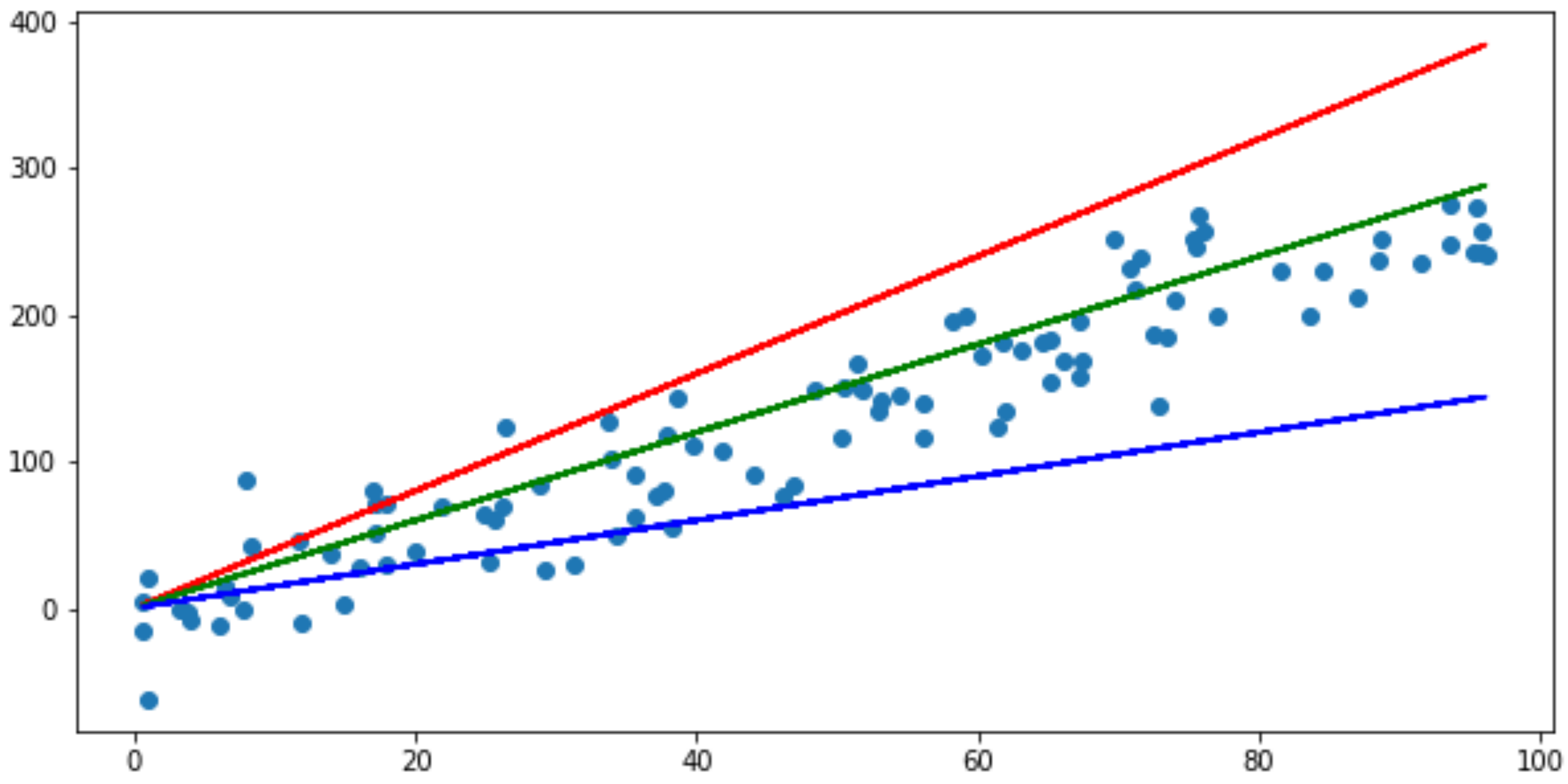
250
200
150
100
50
0
−50
0
20
40
60
80
100



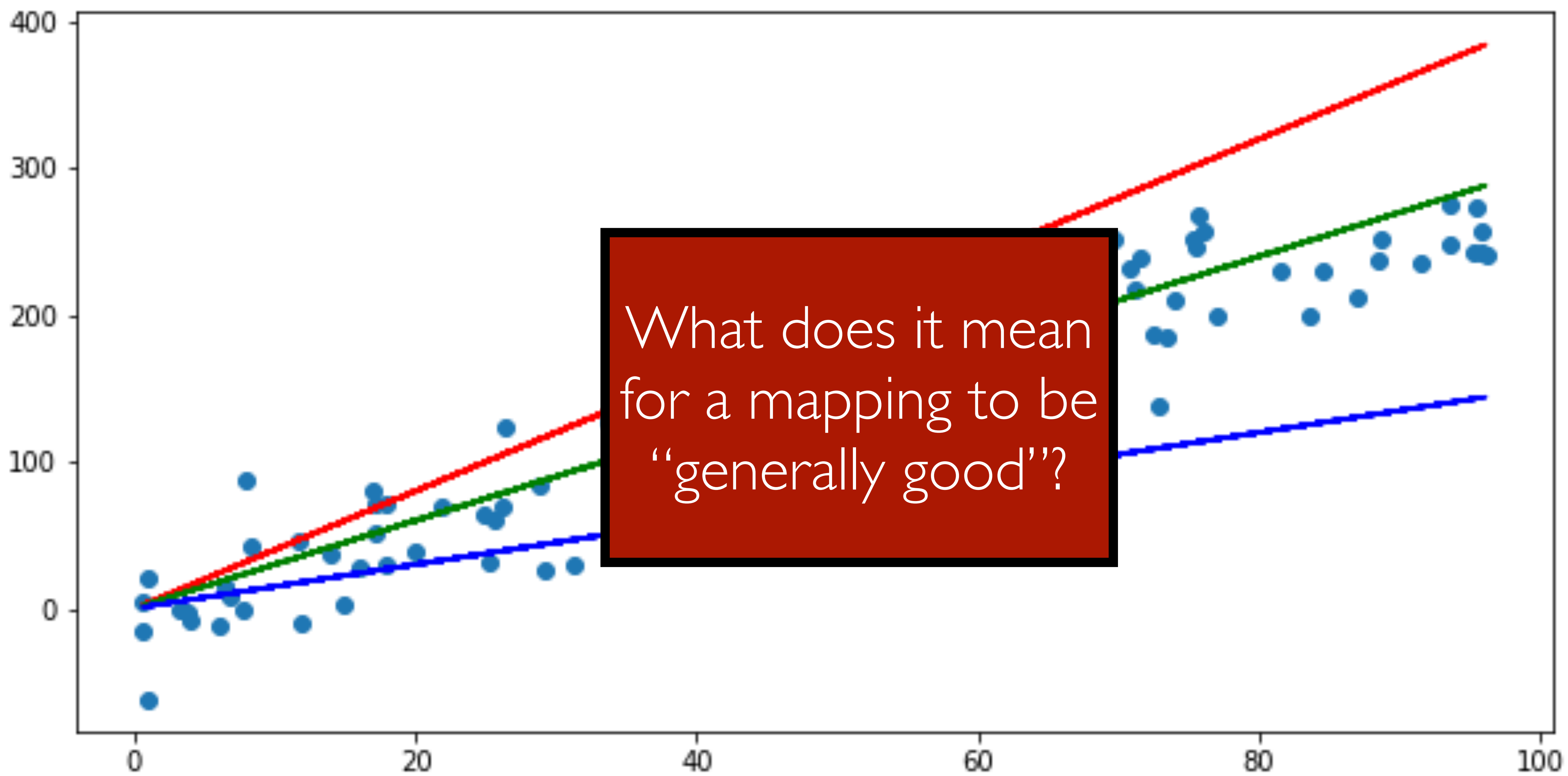
400
300
200
100
0
0
20
40
60
80
100



400
300
200
100
0
0
20
40
60
80
100



400
300
200
100
0
0
20
40
60
80
100


400
300
200
100
0
0
20
40
60
80
100
What does it mean for a mapping to be "generally good"?

“All models are wrong, but some models are useful”

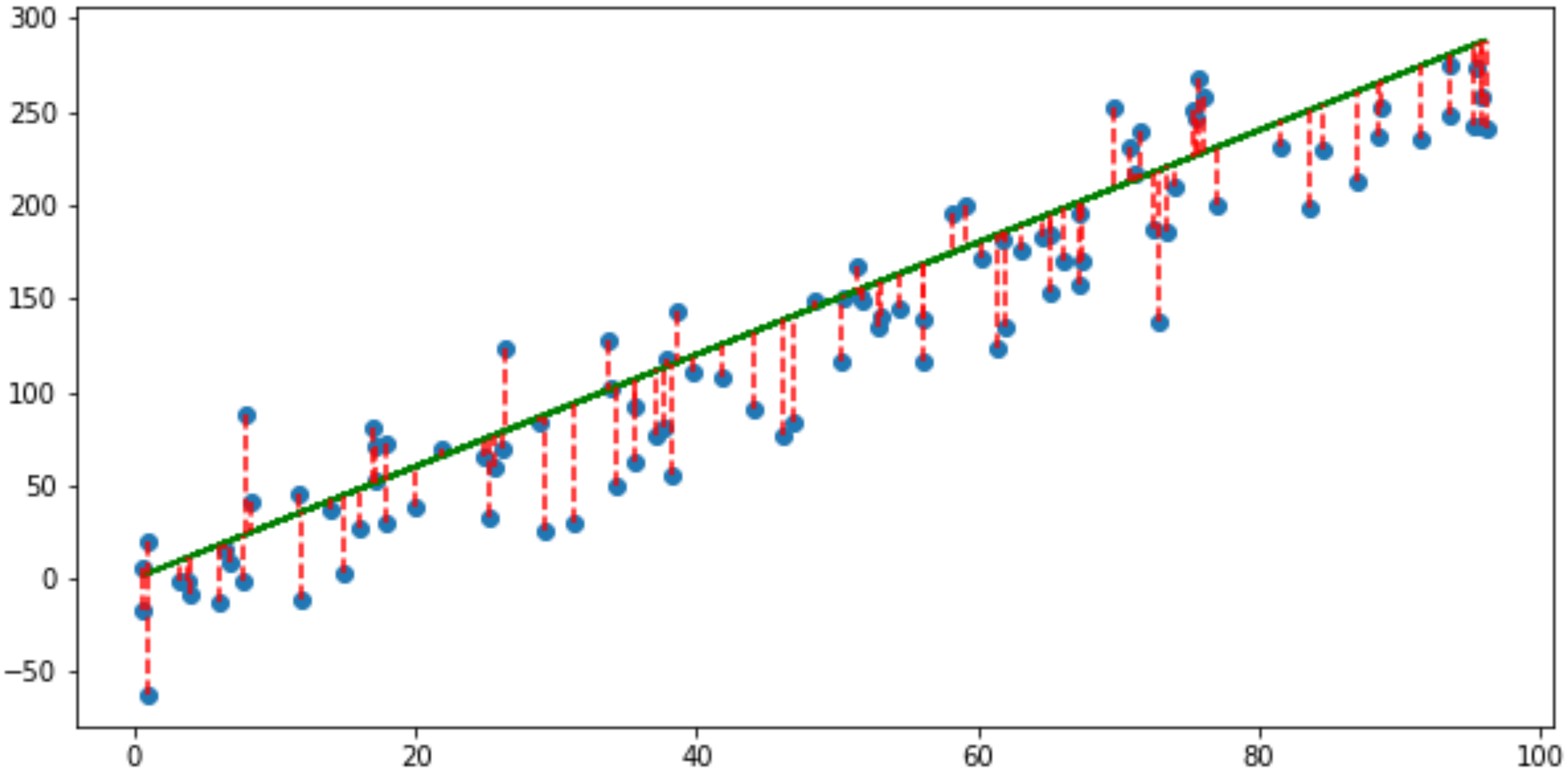
–George Box

OPTIMISATION AND MACHINE LEARNING

- Machine Learning uses optimisation heavily
- Restrict model space. Make parameters of the model space the inputs to a minimisation problem
    - Want to minimise how "bad" the model performs on **seen** data
    - See Empirical Risk Minimisation for further details

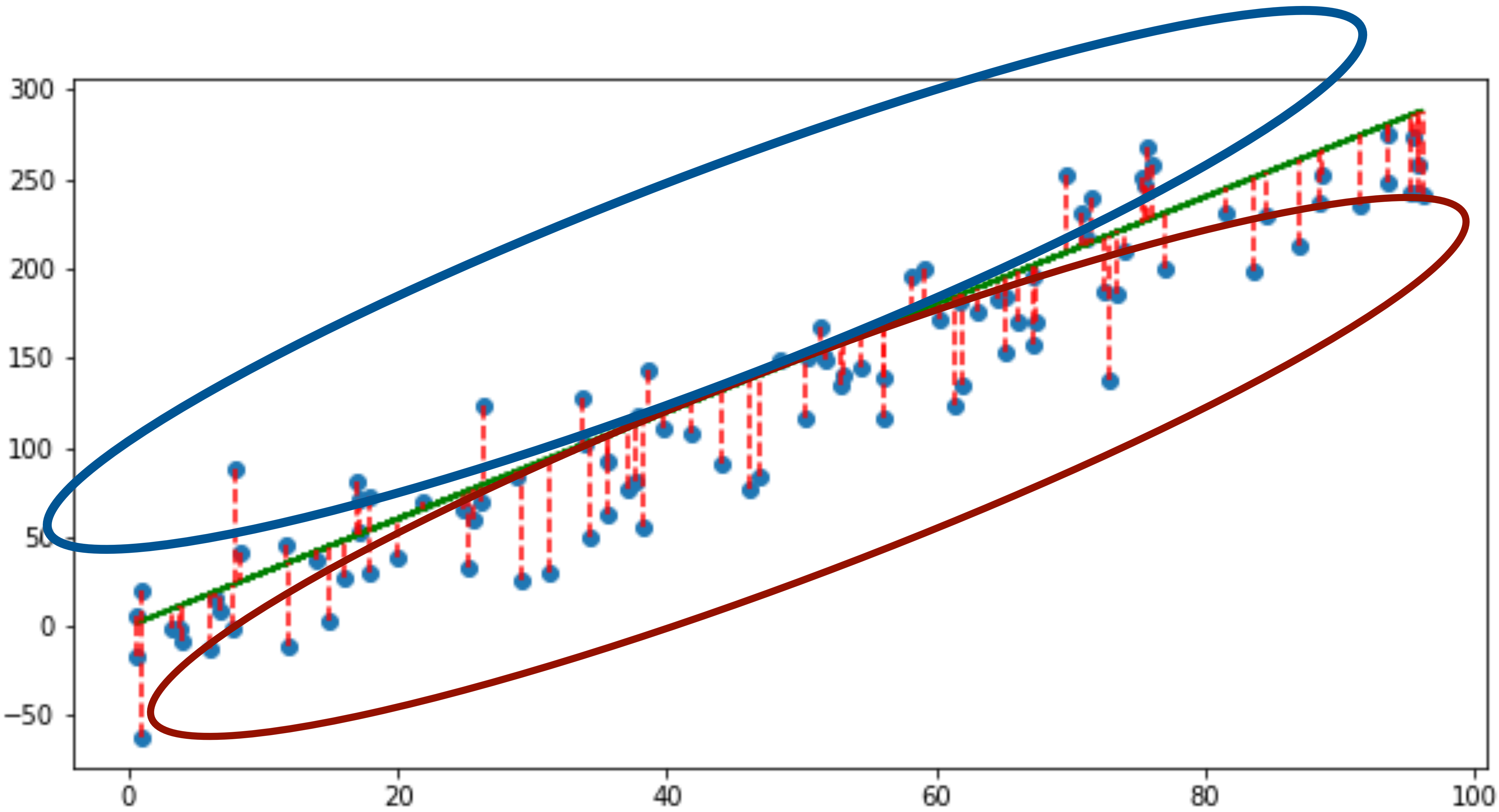
MSE LOSS

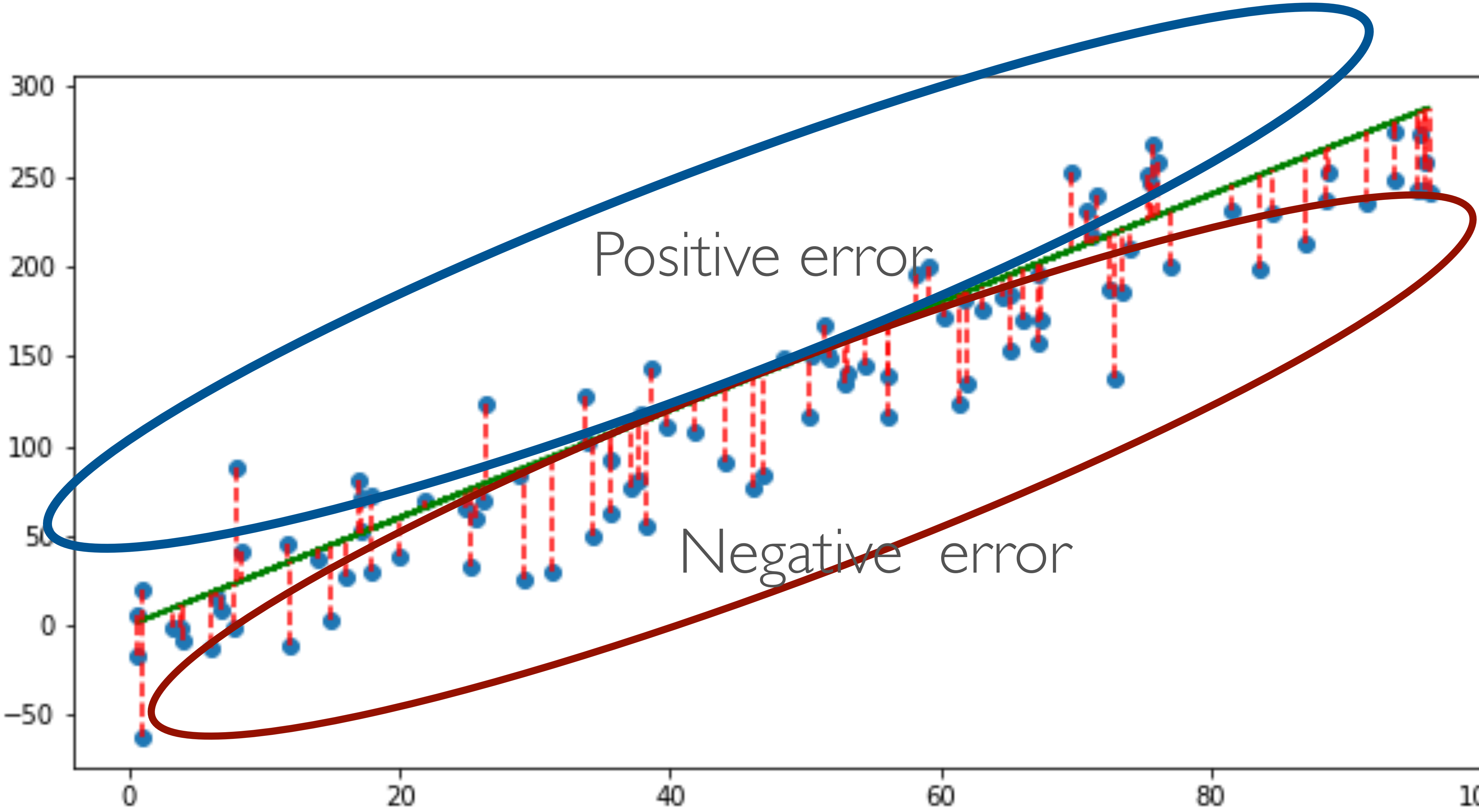
- Need to define a loss function to frame the problem of linear regression as a minimisation problem
- Idea: use difference between predicted output and actual output


300
250
200
150
100
50
0
−50
0
20
40
60
80
100

MSE LOSS

- Let $f_w(x_i) = \hat{y}_i = wx_i$
- Error for example i is $y_i - \hat{y}_i$
- Generally good can be taken to mean a low average error
- $\frac{1}{n}\sum_{i=1}^{n} y_i - \hat{y}_i$?


300
250
200
150
100
50
0
−50
0
20
40
60
80
100



Positive error
Negative error
300
250
200
150
100
50
0
−50
0
20
40
60
80
100

MSE LOSS

- Summing might raw errors might cause errors to cancel out, thereby "drowning" out signal
  - This is undesirable
- Need to add only positive values. Two possibilities:
  - Take the absolute difference (robust regression): $|y_i - \hat{y}_i|$
  - Take the square difference (linear regression): $(y_i - \hat{y}_i)^2$

LINEAR REGRESSION

- For model f_w, where w is our parameter, our loss can be derived as follows:

$$\mathscr{L}(D, f_w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - wx_i)^2$$

Need to solve

$$\underset{w \in \mathbb{R}}{\text{minimize}}\ \mathscr{L}(D, f_w)$$

HOW TO SOLVE

- Gradient Descent!
- Recall that $f_3 = f_1 + f_2 \implies \dfrac{df_3}{dx} = \dfrac{df_1}{dx} + \dfrac{df_2}{dx}$
- Also recall the chain rule of differentiation

Let $\mathcal{L}_i(D, f_w) = (y - wx_i)^2$

Hence $\dfrac{d\mathcal{L}_i(D, f_w)}{dw} = -2x_i(y - wx_i)$

Hence $\dfrac{d\mathcal{L}(D, f_w)}{dw} = \dfrac{1}{n}\sum_{i=1}^{n} -2x_i(y - wx_i)$

Let $\mathscr{L}_i(D, f_w) = (y - wx_i)^2$

Hence $\dfrac{d\mathscr{L}_i(D, f_w)}{dw} = 2x_i(y - wx_i)$

Hence $\dfrac{d\mathscr{L}(D, f_w)}{dw} = \dfrac{-2}{n} \sum_{i=1}^{n} x_i(y - wx_i)$

Our gradient in gradient descent

WHAT ABOUT THE INTERCEPT

- Can use partial derivative to find update rule for intercept

$$\text{Hence } \frac{d\mathcal{L}(D, f_w)}{dc} = 1$$

IMPROVEMENTS

- In practice, we don't use vanilla gradient descent
- Use a variations such as stochastic gradient descent and friends (AdaGrad, RMSProp, etc…)
- In SGD, we don't calculate loss over all data points per iteration, only a sample
  - SGD is faster in practice, but may not find as a good a point as GD

REGULARIZATION

- To prevent overfitting, we use regularization
- Penalize "size" of weights using l_1 or l_2 norm