

Lecture #7: Linear Models - Linear Regression and Logistic Regression

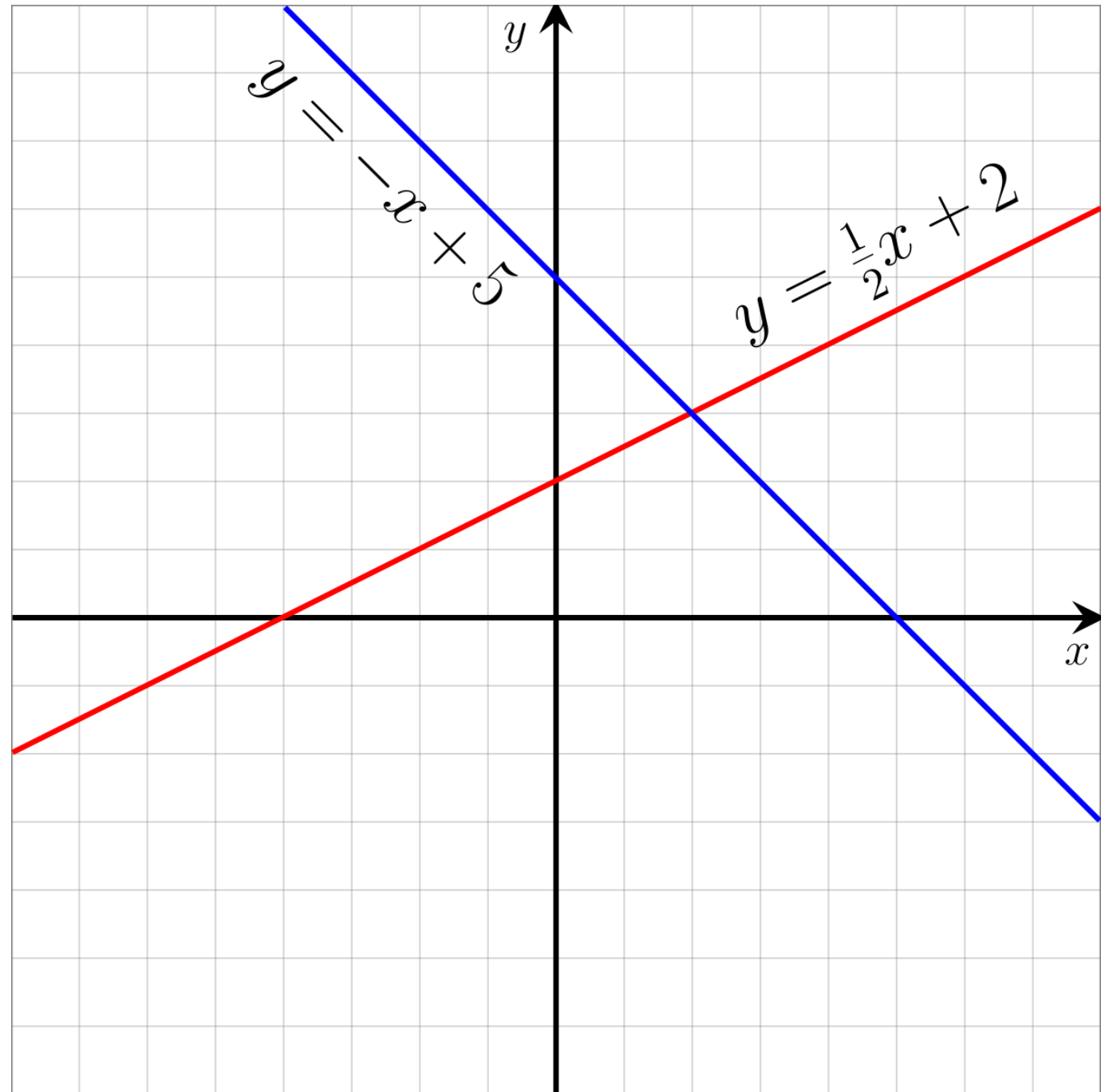
COMP3608 - Intelligent Systems
Inzamam Rahaman

Machine Learning

- Last week we discussed the construction of machine learning pipelines
- And ML concepts that apply across many models
- This week, we will look at an actual family of machine learning models - linear models - that are used for supervised learning.
 - Output is dependent on linear function of input
- Concepts explored are important for neural networks
- Linear models have a formal basis in probability
 - Not all machine learning models have such a formal basis
 - Will look at both geometric and probabilistic interpretations

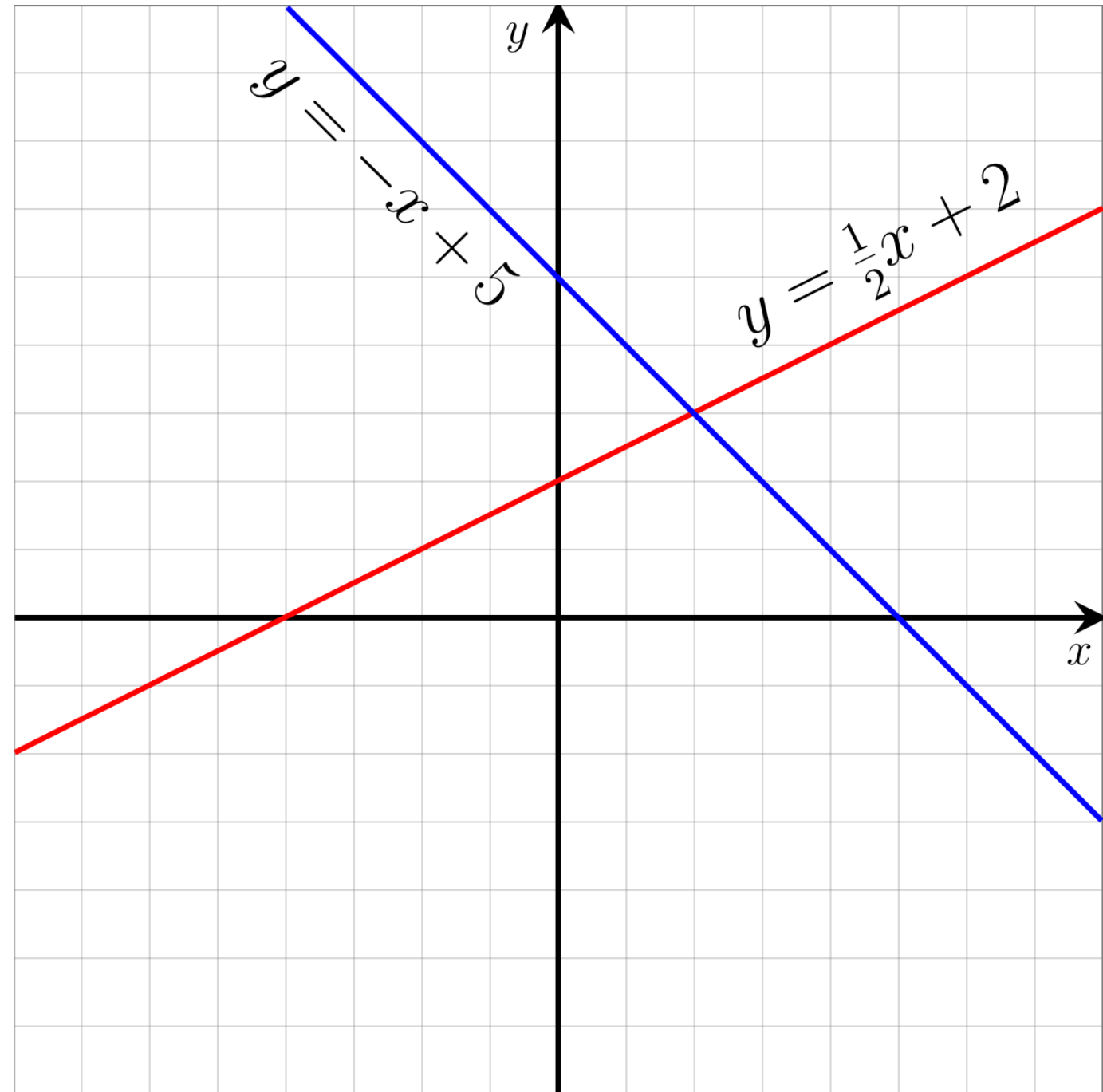
Linear Models

- Linear means line
- So think about $y = w^T x + c$ and its generalisations where
 - $x \in \mathbb{R}^n$
 - $w \in \mathbb{R}^n$
 - $y \in \mathbb{R}$
 - $c \in \mathbb{R}$
- Will be using dot product based formulation of line



Linear Models

- Linear means line
- So think about $y = mx + c$ and its generalisations
- Will be using dot product based formulation of line
- Suppose that $\hat{x} = x \oplus 1$ and $w = m \oplus c$, can express $y = mx + c$ as $y = w^T \hat{x}$
- Here \oplus is the concatenation operator, so $\begin{bmatrix} 2 \\ 3 \end{bmatrix} \oplus [1] = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$



Linear Regression

- Used for regression problems
- Recall that in regression, we have independent/input variables drawn from $X = \mathbb{R}^m$ and dependent variable $Y = \mathbb{R}$
 - m - number of input features
 - n - number of data points used for training
- Want to learn function from X to Y such that function captures general relationship well
- This function is our model

Linear Regression - geometric interpretation

- Will look at geometric interpretation first
- We assume that the relationship between input variables and output variable is linear or captured by linear function
- In other words, $y_i = w^T x_i + c$

Linear Regression - geometric interpretation

- Will look at geometric interpretation first
- We assume that the relationship between input variables and output variable is linear or captured by linear function

- In other words, $y_i = w^T x_i + c$

weights

that apply to every example

intercept for every
example

Linear Regression

- Essentially, linear regression is about finding the best-fit line for our data
- How can we formalise a notion of finding the best-fit line?

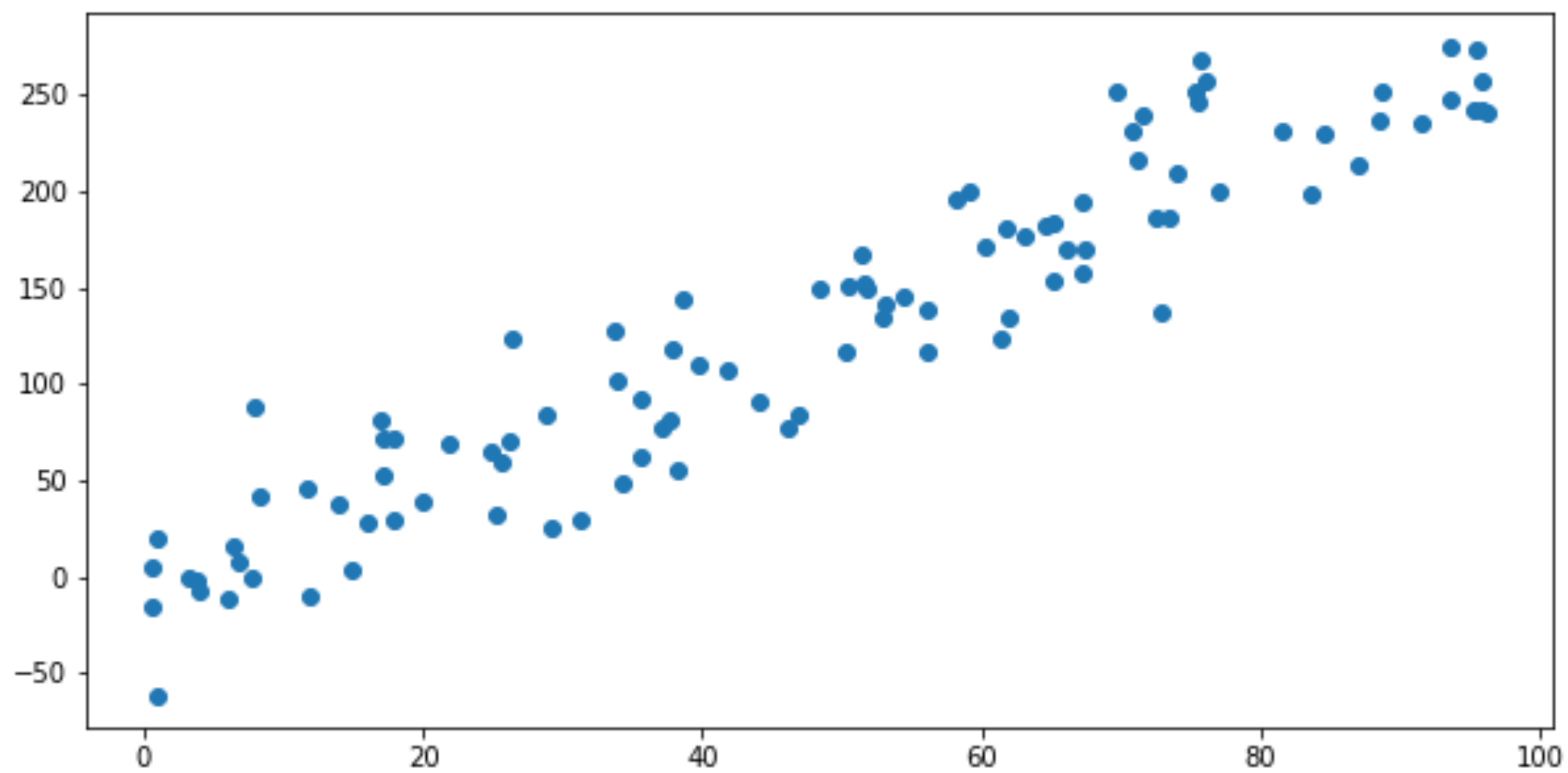
Linear Regression

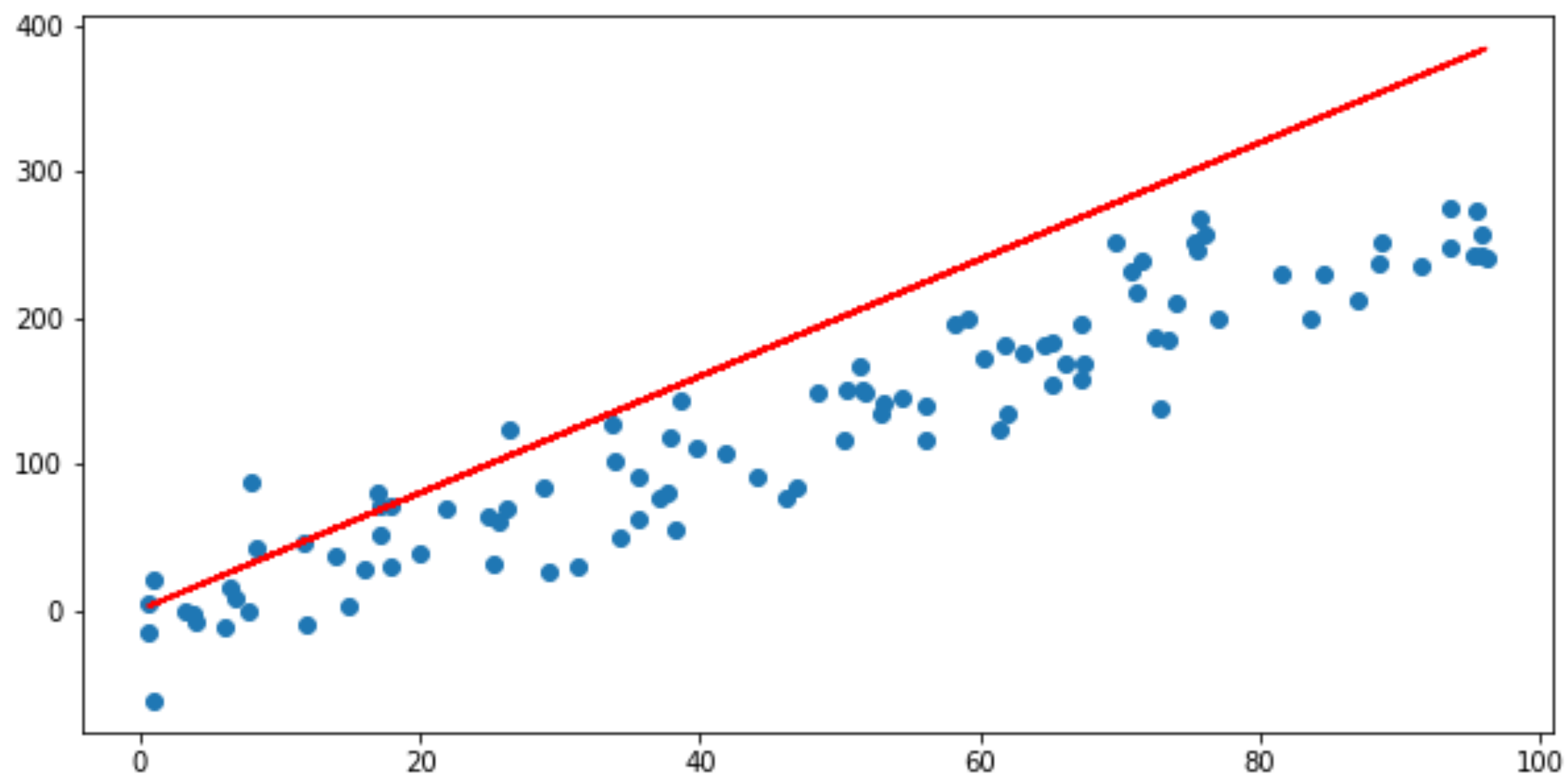
- Essentially, linear regression is about finding the **best**-fit line for our data
- How can we formalise a notion of **finding** the **best**-fit line?

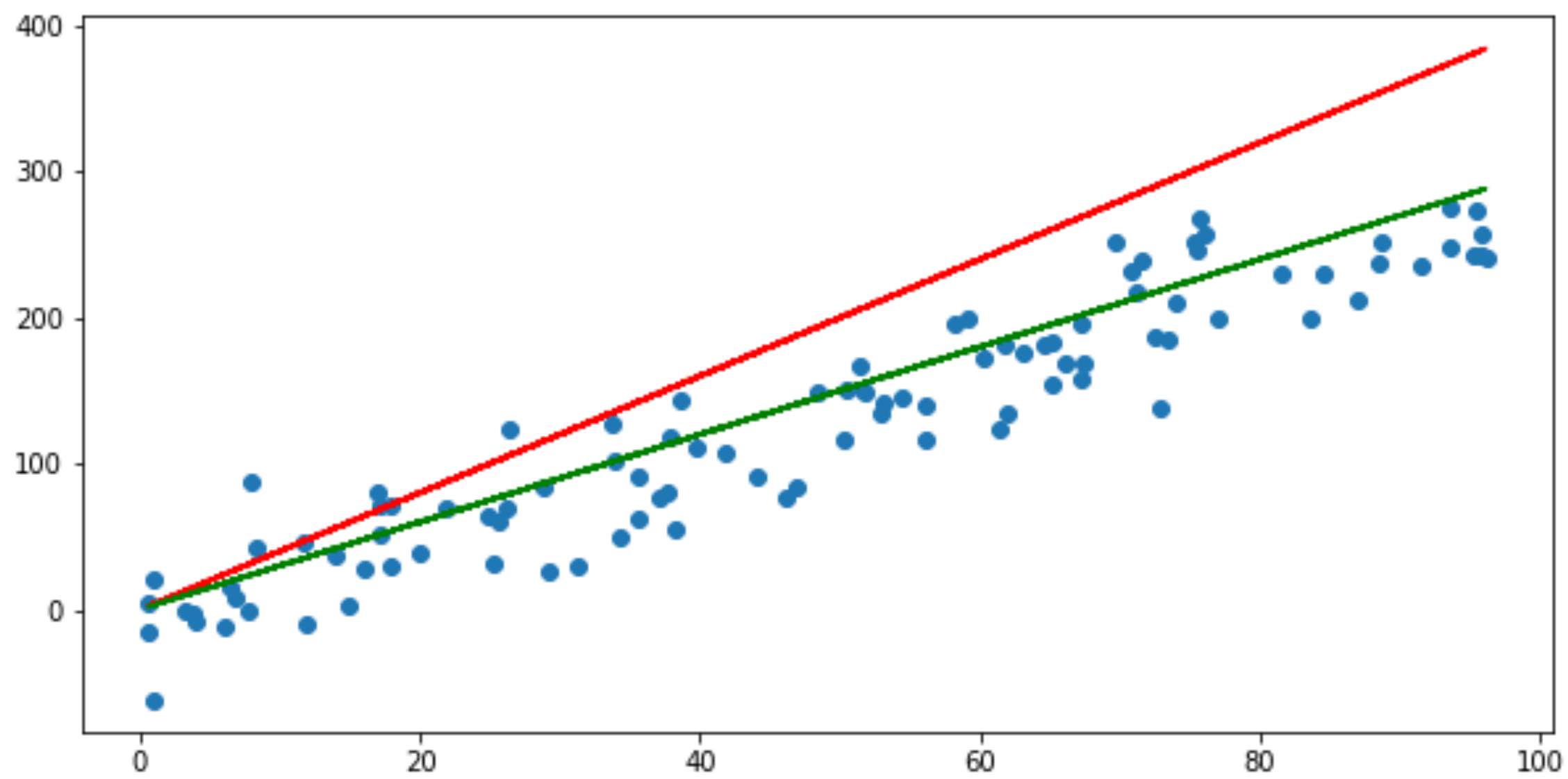
Linear Regression

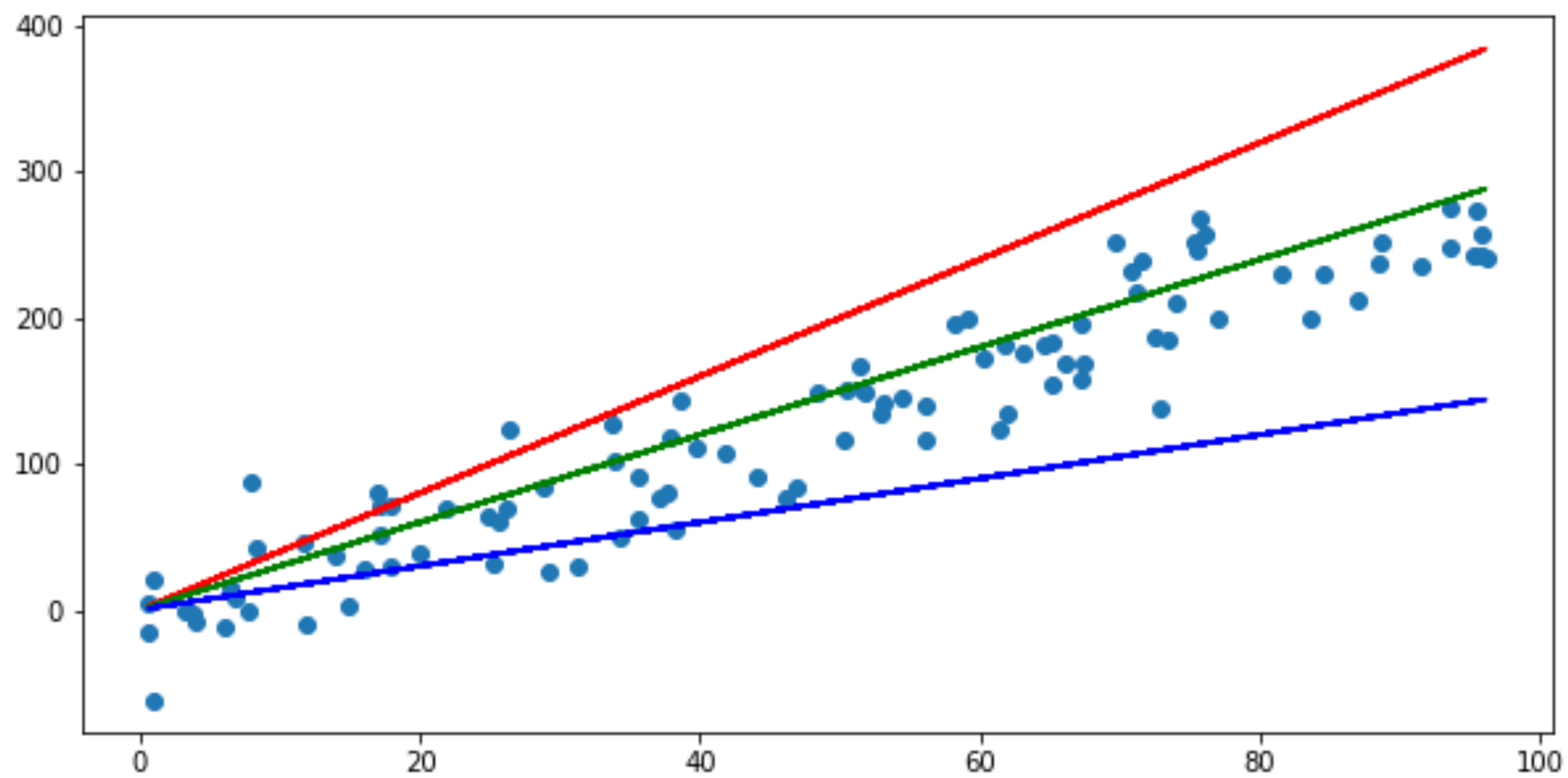
- Essentially, linear regression is about finding the **best**-fit line for our data
- How can we formalise a notion of **finding** the **best**-fit line?

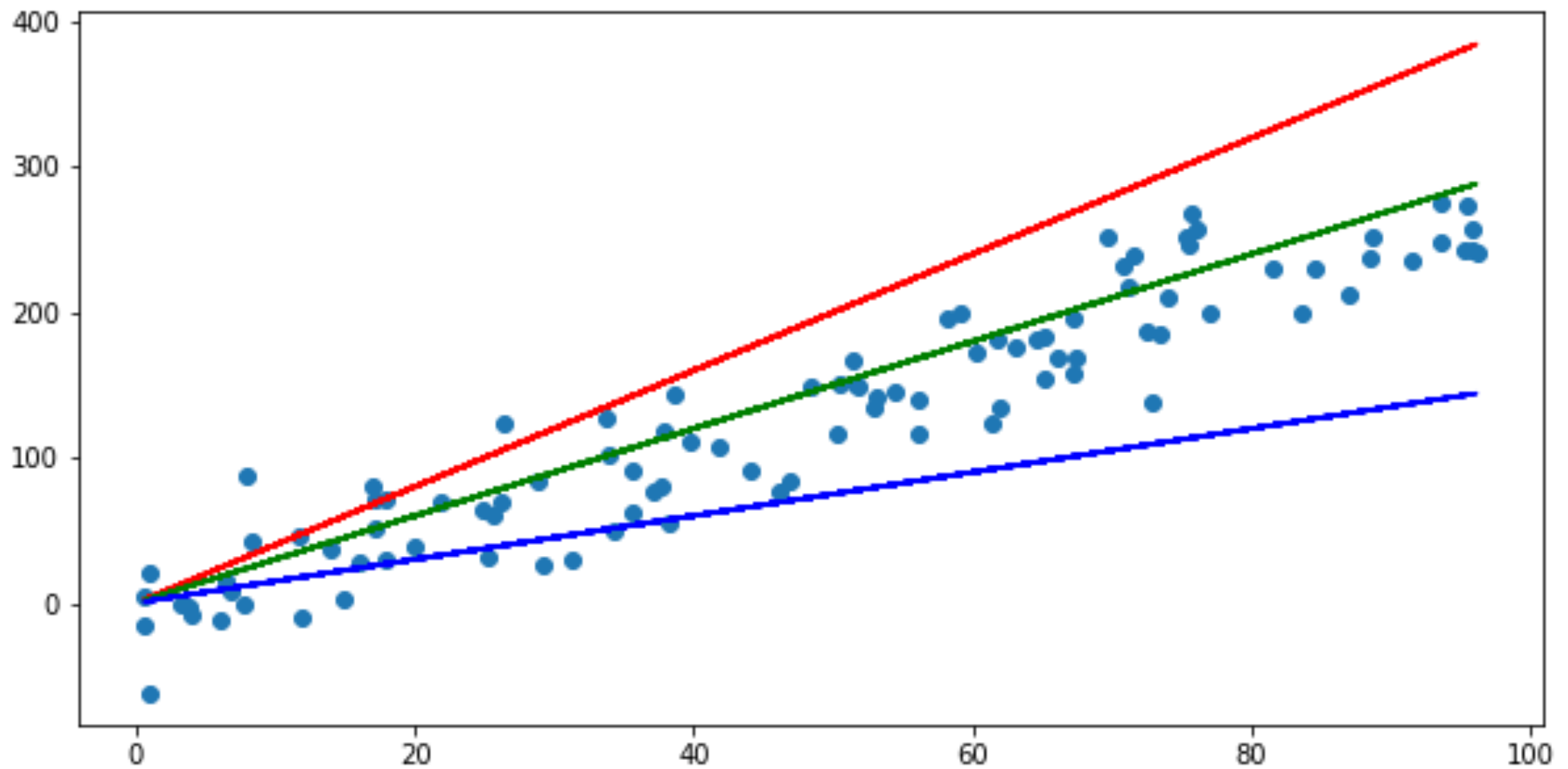
Optimisation!



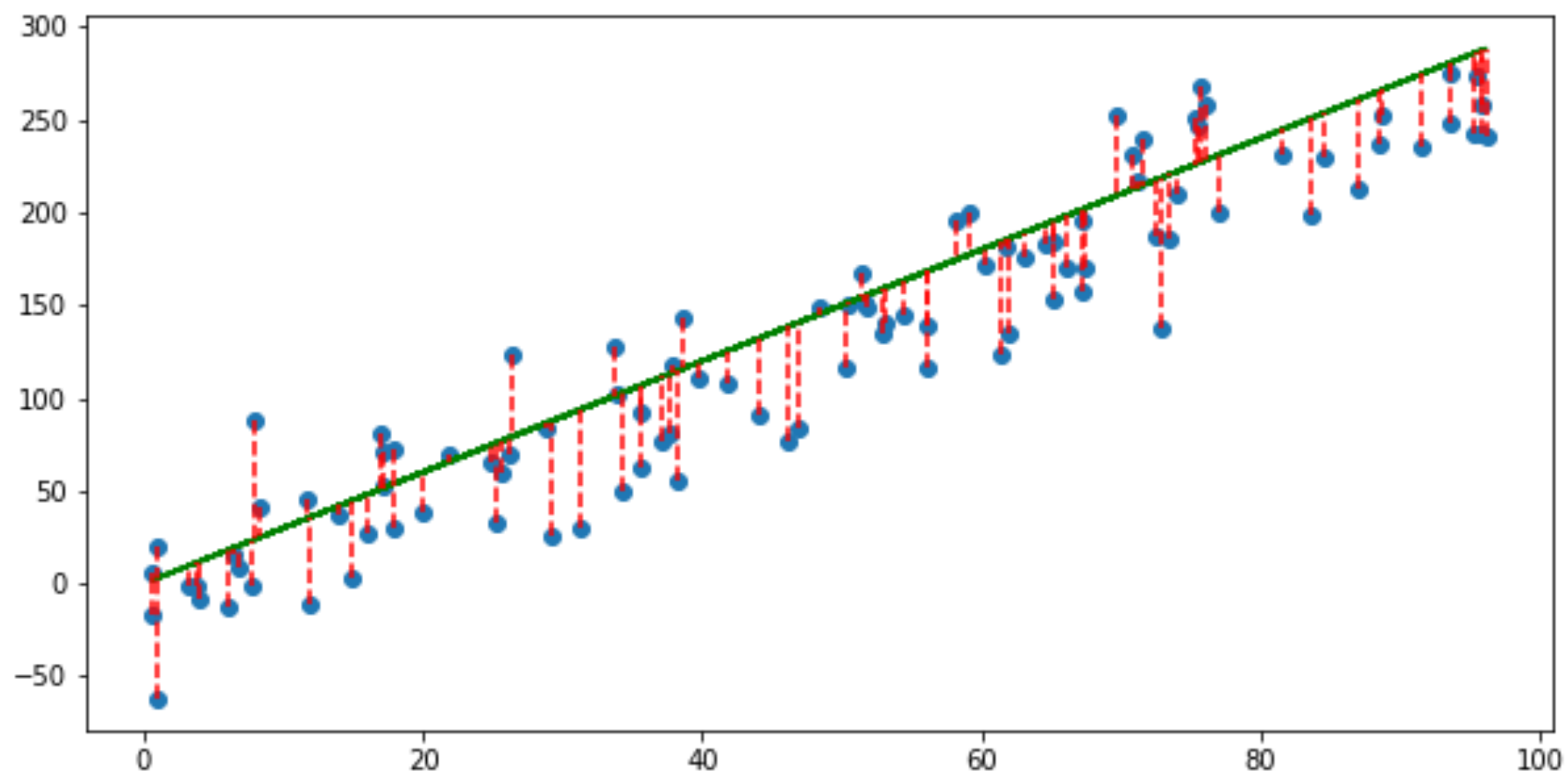


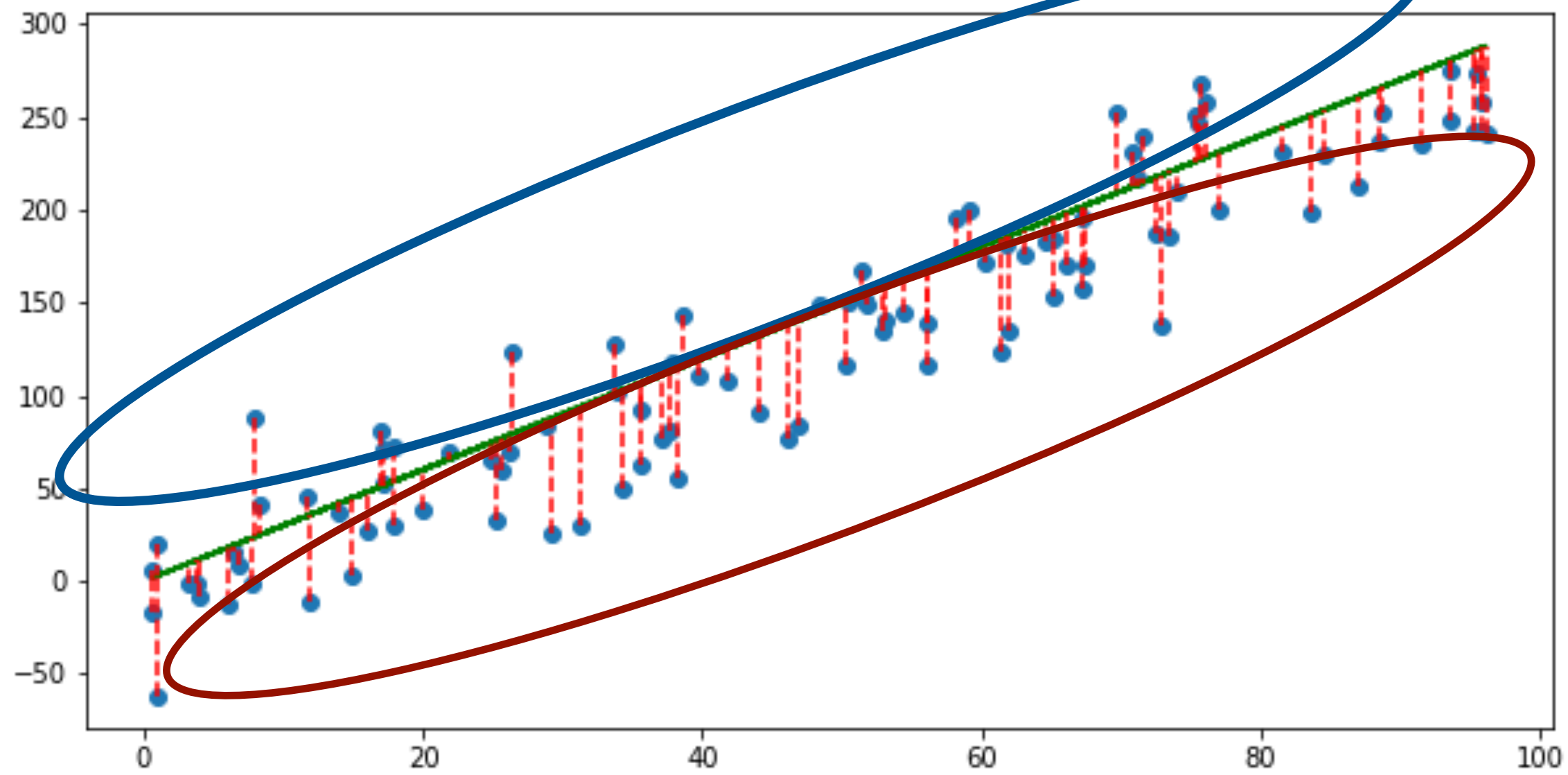


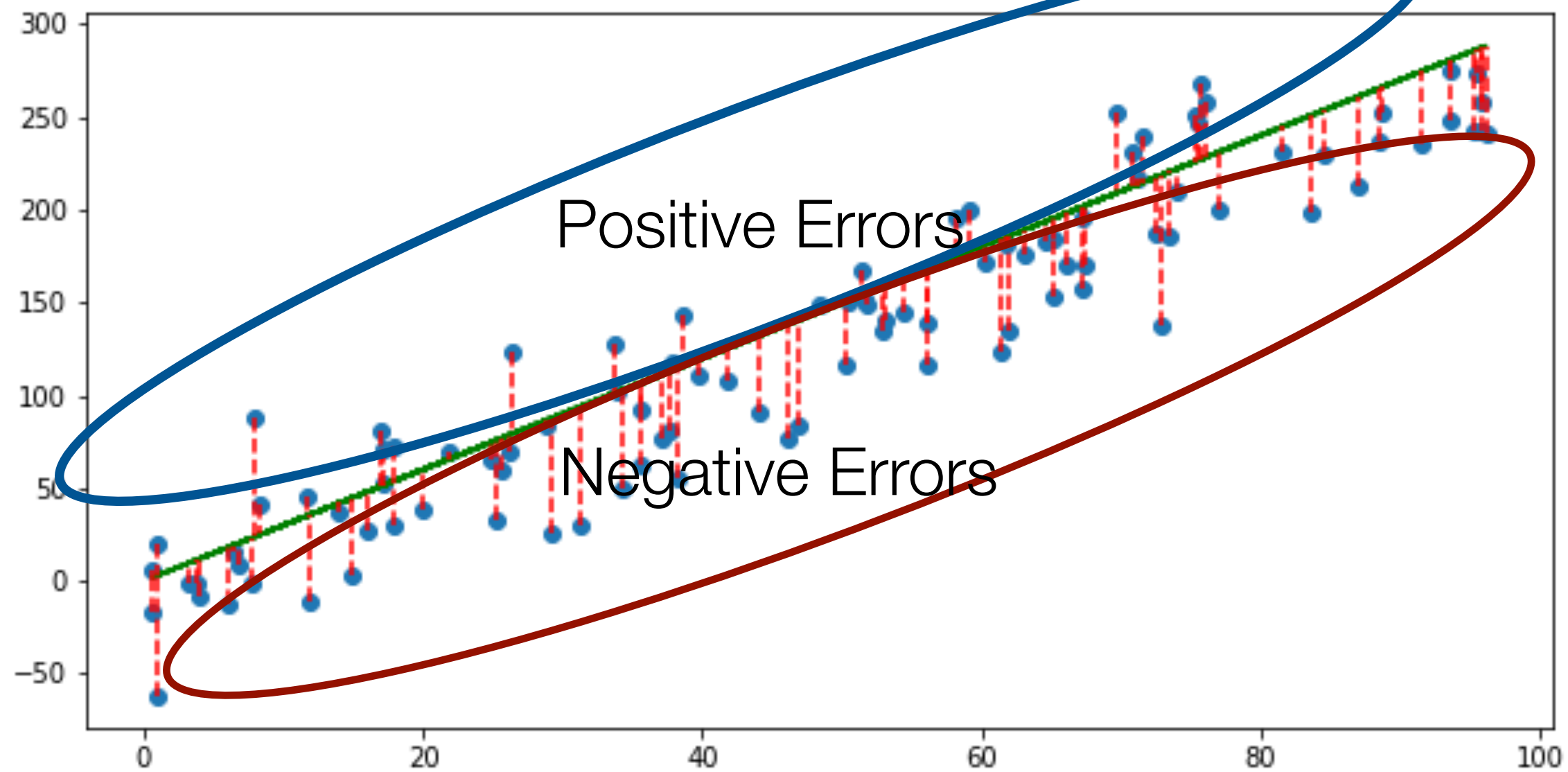




How can formalise which line is the best?







Measuring Line Quality

- Can't simply add up errors
 - Errors will cancel each other out
 - Also, we want to be “fair” to all data points
- Need way to measure line quality that circumvents these problems

Mean Squared Error - MSE

- Mean Squared Error avoids these problems
 - By squaring the error, we ensure that we get positive contributions to sum - errors would not cancel each other out
 - By taking the mean, we are “fair” to other data points!
- Suppose that $y_i = w^T x_i + c$, then the MSE for that w, c is

$$\frac{1}{n} \sum_{i=1}^n (y - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y - (w^T x_i + c))^2$$

Linear Regression As an Optimisation Problem

- Hence, the problem of finding a best fit line can be framed as an optimisation problem of the form

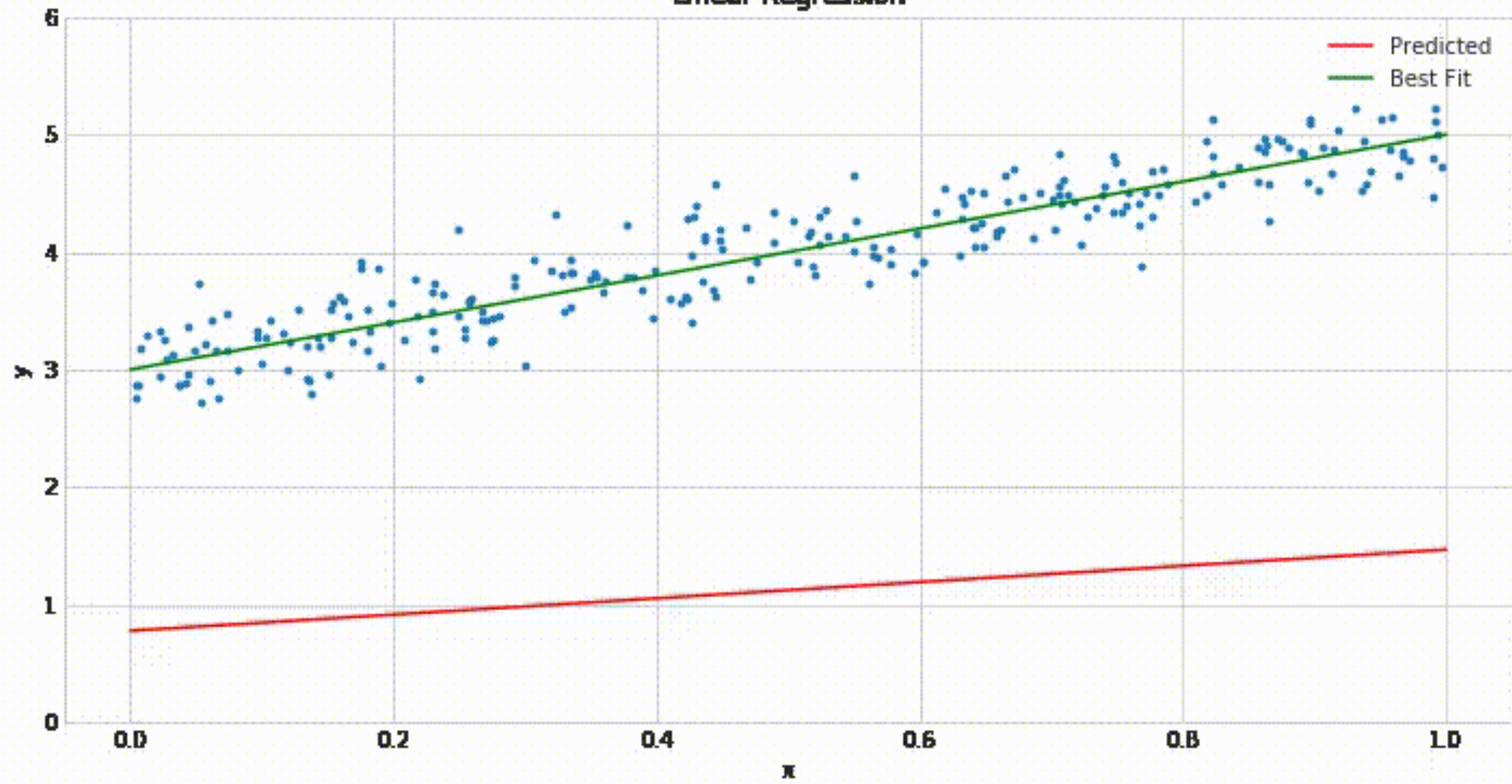
$$\frac{1}{n} \sum_{i=1}^n (y - (w^T x_i + c))^2$$

subject to

$$w \in \mathbb{R}^m, c \in \mathbb{R}$$

Can be solved using gradient descent!

Linear Regression



Linear Regression - Probabilistic Perspective

- Probabilistic perspective very important to understand neural networks
- Is a bit trick and makes assumptions of linear regression much more apparent
- MSE loss is not just meaningful from a geometric perspective - also meaningful from a probability perspective as well
- Will arrive at the same conclusion, but through a different avenue

Supervised Learning from a Probability Perspective

- From a probability perspective, our inputs (X) are random variables, and our outputs (Y) are random variables
- Our prediction for output should be what we expect Y to be if we know X
- In other words, our function is always $E[Y | X]$

Supervised Learning from a Probability Perspective

- To compute $E[Y | X]$, we need to know $P(Y | X)$
- How to find $P(Y | X)$?

Supervised Learning from a Probability Perspective

- To compute $E[Y | x_i]$, we need to know $P(Y | X = x_i)$ for all i
- How to find $P(Y | X)$?

Use MLE to fit $P(Y | X)$

Supervised Learning from a Probability Perspective

- We need to make some simplifying assumptions such as the form of $P(Y|X)$.
- Let's assume that $P(Y|X)$ follows a normal distribution
- However, recall that distributions are characterised by a set of parameters
- How can we find parameters for $P(Y|X)$?

$$\mathcal{N}(y; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{y-\mu}{2\sigma}\right)^2}$$

Supervised Learning from a Probability Perspective

- We let parameters of $P(Y|X)$ be a function of X .

Supervised Learning from a Probability Perspective

- We let parameters of $P(Y|X)$ be a function of X .
- In Linear Regression, we assume μ is a linear function of our inputs, and that σ is 1.
- Hence

$$P(y_i | x_i) = \mathcal{N}(y_i; \mu = w^T x_i + c, \sigma = 1)$$

Supervised Learning from a Probability Perspective

- We let parameters of $P(Y|X)$ be a function of X .
- In Linear Regression, we assume μ is a linear function of our inputs, and that σ is 1.
- Hence

$$P(y_i | x_i) = \mathcal{N}(y_i; \mu = w^T x_i + c, \sigma = 1)$$

Need to find w and c that maximises likelihood

$$\mathcal{L}(D; w, c) = \prod_{i=1}^n P(y_i | x_i)$$

$$\mathcal{L}(D; w, c) = \prod_{i=1}^n \mathcal{N}(y_i; \mu = w^T x_i + c, \sigma = 1)$$

$$\log \mathcal{L}(D; w, c) = \sum_{i=1}^n \log \mathcal{N}(y_i; \mu = w^T x_i + c, \sigma = 1)$$

$$\log \mathcal{L}(D; w, c) = \sum_{i=1}^n \left(\log 1 - \log \sigma - \log \sqrt{(2\pi)} - \frac{(y_i - \mu)^2}{2} \right)$$

$$\log \mathcal{L}(D; w, c) = \sum_{i=1}^n \left(\log 1 - \log \sigma - \log \sqrt{(2\pi)} - \frac{(y_i - (w^T x_i + c))^2}{2} \right)$$

$$\text{NLL}(D; w, c) = - \sum_{i=1}^n \left(\log 1 - \log \sigma - \log \sqrt{(2\pi)} - \frac{(y_i - (w^T x_i + c))^2}{2} \right)$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} \left(- \sum_{i=1}^n (\log 1 - \log \sigma - \log \sqrt{(2\pi)} - \frac{(y_i - (w^T x_i + c))^2}{2}) \right)$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} - \sum_{i=1}^n - \frac{(y_i - (w^T x_i + c))^2}{2}$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} - 1 \frac{-1}{2} \sum_{i=1}^n (y_i - (w^T x_i + c))^2$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} \sum_{i=1}^n (y_i - (w^T x_i + c))^2$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^n (y_i - (w^T x_i + c))^2$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} \left(- \sum_{i=1}^n (\log 1 - \log \sigma - \log \sqrt{(2\pi)} - \frac{(y_i - (w^T x_i + c))^2}{2}) \right)$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} - \sum_{i=1}^n - \frac{(y_i - (w^T x_i + c))^2}{2}$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} - 1 \frac{-1}{2} \sum_{i=1}^n (y_i - (w^T x_i + c))^2$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} \sum_{i=1}^n (y_i - (w^T x_i + c))^2$$

$$\min_{w \in \mathbb{R}^m, c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^n (y_i - (w^T x_i + c))^2$$

MSE Loss

Linear Regression - getting predictions

- So we fit the distribution
 - Now what?
- Recall that for a normal distribution, the expectation is simply μ
- Hence, once we fit the distribution, our output is simply $w^T x_i + c$
- Can use intermediary outputs in optimisation
- We have reconstructed linear regression from first principles!

Logistic Regression

- Despite its name logistic regression is actually used for classification rather than for regression per se
- We output the probability of a data point belonging to the different classes
- Two main variants:
 - Binary
 - Multi-class (NOT multi-label)
- Now that we have seen how MLE can be used to fit a machine learning model, we shall look at logistic regression primarily from a probabilistic perspective

Binary Logistic Regression

- Can model outcomes as yes (1) or no (0)
- These can be modelled as probability of true
- When we have two possible outcomes, what probability distribution do we use?

Binary Logistic Regression

- Can model outcomes as yes (1) or no (0)
- These can be modelled as probability of true
- When we have two possible outcomes, what probability distribution do we use?
 - We use the Bernoulli Distribution
 - Hence, we let $P(y_i | x_i) = \text{Bern}(y_i; p = f(x_i))$

The problem of the success parameter

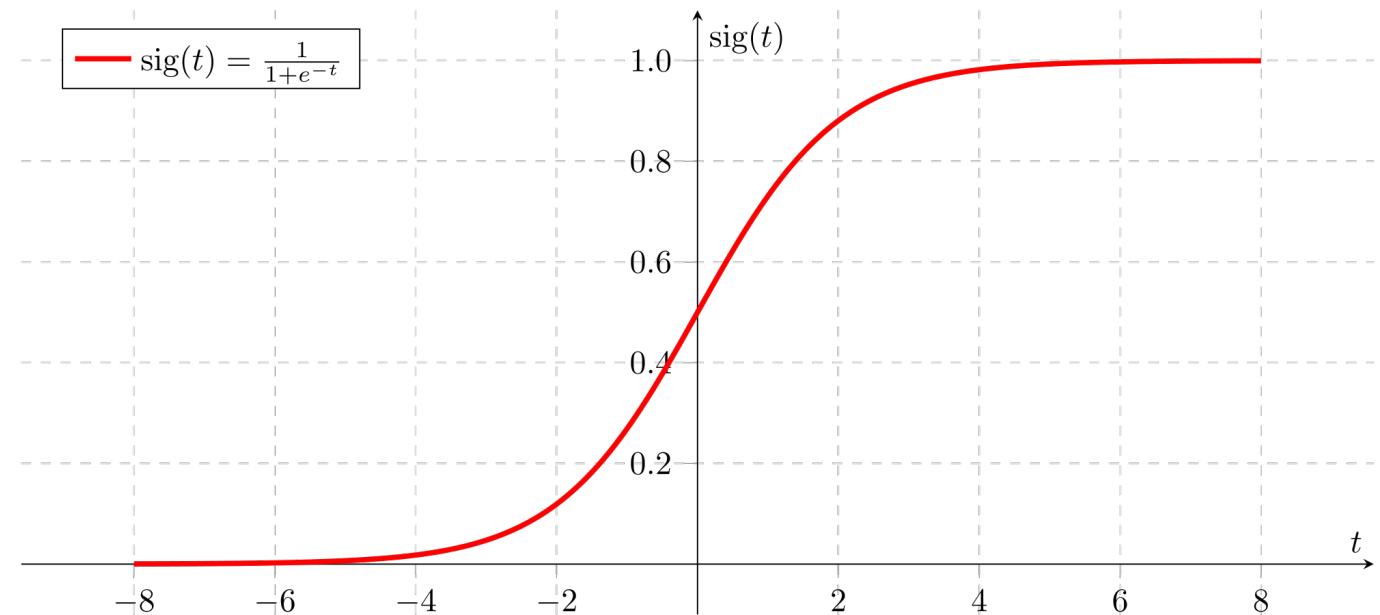
- The success parameter of a Bernoulli distribution is bounded within the range 0 to 1
- Hence, we need to ensure that $f(x_i)$ is a bounded between 0 and 1
- How can we achieve this?
 - We use the logistic sigmoid function
 - Usually denoted as σ

Logistic Sigmoid

- Logistic Sigmoid is special class of functions
- We use the standard logistic sigmoid
- And usually just say sigmoid function
- Squashes all input between 0 and 1 inclusive

- $$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- $$\frac{d\sigma}{dx} = \sigma(x) * (1 - \sigma(x))$$



Sigmoidal activation function usage

- We let
$$P(y_i | x_i) = \text{Bern}(y_i; p = \sigma(w^T x_i + c)) = p^{y_i} (1 - p)^{1 - y_i}$$
- Note that this is still considered a linear model
- As the input to the sigmoid function is a linear function of the problem input data
- Just like before, we use MLE to derive a loss function that we would use to get good values of w and c

$$\mathcal{L}(D; w, c) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}$$

$$\mathcal{L}(D; w, c) = \prod_{i=1}^n \sigma(w^T x_i + c)^{y_i} (1 - \sigma(w^T x_i + c))^{1-y_i}$$

$$\log \mathcal{L}(D; w, c) = \sum_{i=1}^n \left(y_i \log(\sigma(w^T x_i + c)) + (1 - y_i) \log(1 - \sigma(w^T x_i + c)) \right)$$

$$\text{NLL}(D; w, c) = - \sum_{i=1}^n \left(y_i \log(\sigma(w^T x_i + c)) + (1 - y_i) \log(1 - \sigma(w^T x_i + c)) \right)$$

Optimising NLL for Logistic Regression

- Just like before, we replace the summation with the mean for numerical stability
- So, our final loss function is

$$-\frac{1}{n} \sum_{i=1}^n \left(y_i \log(\sigma(w^T x_i + c)) + (1 - y_i)(1 - \sigma(w^T x_i + c)) \right)$$

Expectation of Logistic Regression

- $E[y_i | x_i] = \sigma(w^T x_i + c)$
- We output probability score!
- Can use this prediction directly in optimising model

Logistic Regression - Linear Separators

- The linear function in the sigmoid represents a hyperplane (generalisation of a line) that tries to separate the two classes
- Dot product tells us which side of the line a point is located
- Problems where the data is difficult to separate linearly cannot be adequately solved by linear regression :(
- That is where neural networks come in

Multiclass-Logistic Regression


- Multi-class LR differs from Binary LR in several important ways
- Multi-class LR uses a categorical distribution
- Instead of learning a weight vector, we learn a weight matrix
- If we have m features and d classes, our weights is a $m \times d$ matrix. In addition, instead of learning a single intercept, we learn d intercepts stored as a vector of length d

Multiclass-Logistic Regression

- Hence $W^T x_i + c \in \mathbb{R}^d$
- Each component represents the relative strength of the association towards each class
- Need to convert this to probabilities
- Conventional way is to use softmax function (which replaces sigmoid)

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^d e^{x_j}}$$

called softmax as the value with the highest score comes closest to 1


$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^d e^{x_j}}$$



i th component of
result

Generalised Linear Models

- Linear regression and logistic regression are members of a family of models called GLMs
- Other similar models exist:
 - Laplace Regression
 - Poisson Regression
 - Probit Regression
- Important building block towards developing neural networks

Regularisation

- Recall the problem of overfitting
- We can mitigate overfitting through the use of a regularisation term
- Core idea: larger magnitude weights imply overfitting
- ℓ_p regularisation term is $\lambda \|w\|_p$, where λ is the regularisation coefficient that controls the degree of influence regularisation has on the loss function
- Regularisation term added onto loss function to penalise the magnitude of weights
- Formal probabilistic basis: regularisation encodes information on prior distribution of parameters
 - ℓ_2 regularisation is probabilistically equivalent to Bayesian Linear Regression