Linear Regression, Regression Trees, and Gradient Descent

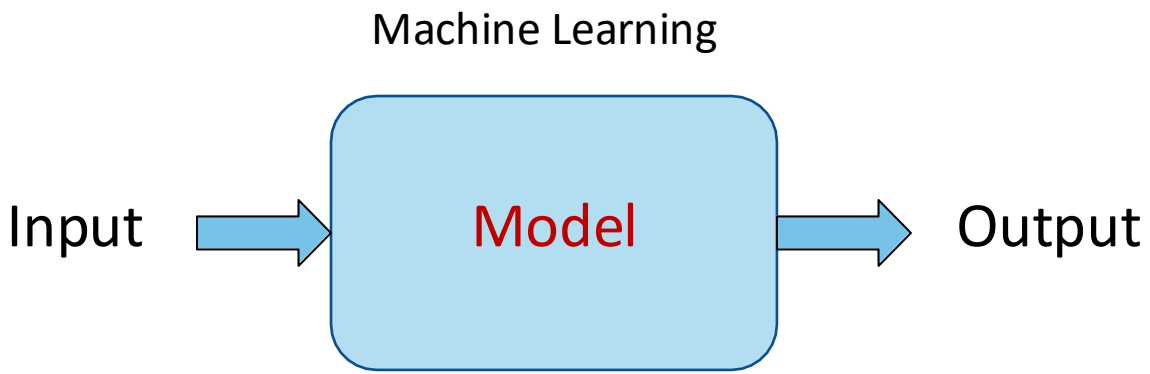
Mehdi Ataei

Recap

- So far, we have covered the **Input** and **Output** to a machine learning model
- We discussed that the input can be of many forms: text, images, etc.
- We discussed that the output depends one the **goal** of the ML model for example:
  - The output may be a **probability** in the case of classification tasks.
  - The output could be a **numerical value** when dealing with regression.
  - The output might **mirror** the input data, especially in generative models like language models.

Machine Learning Model

- Now we want to focus on how we could find the ML model mapping the input to the output
- Remember that goal of ML is to find the model that **best maps** the input to output
- How can we find this mapping?



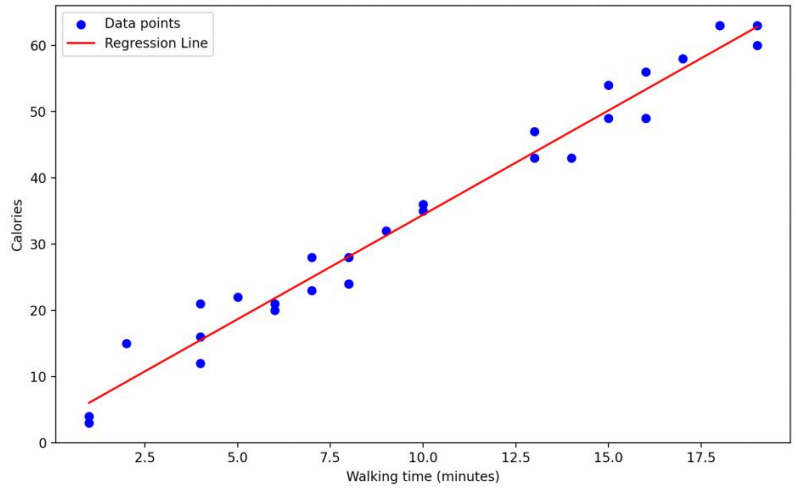
Linear Regression

- Let's start with the most basic ml model: Linear regression
- A linear regression predicts continuous values using a line equation:

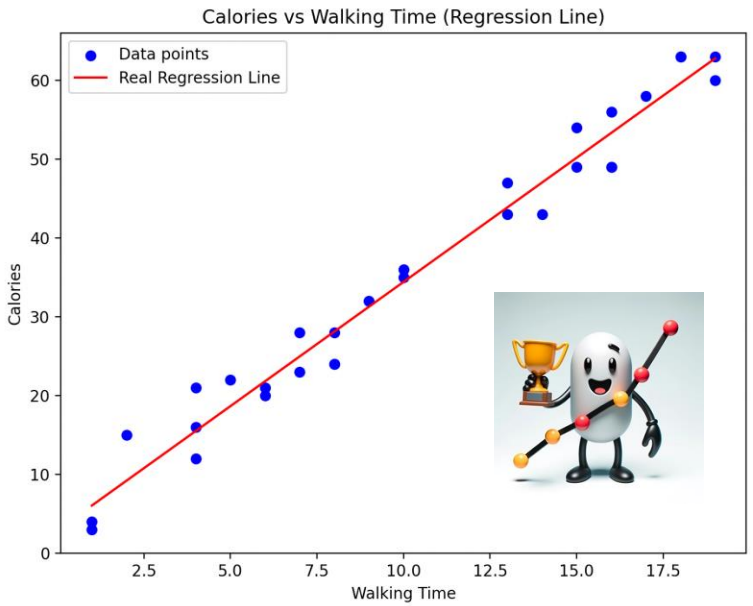
$$\hat{y} = a \cdot x + b$$

Slope (weight)

bias (intercept)



Which line is better?



Finding the Best Line

- How can we find the best line in a linear regression problem?
- What “best” means here (best pair of a and b)
- We need a **metric to measure goodness**

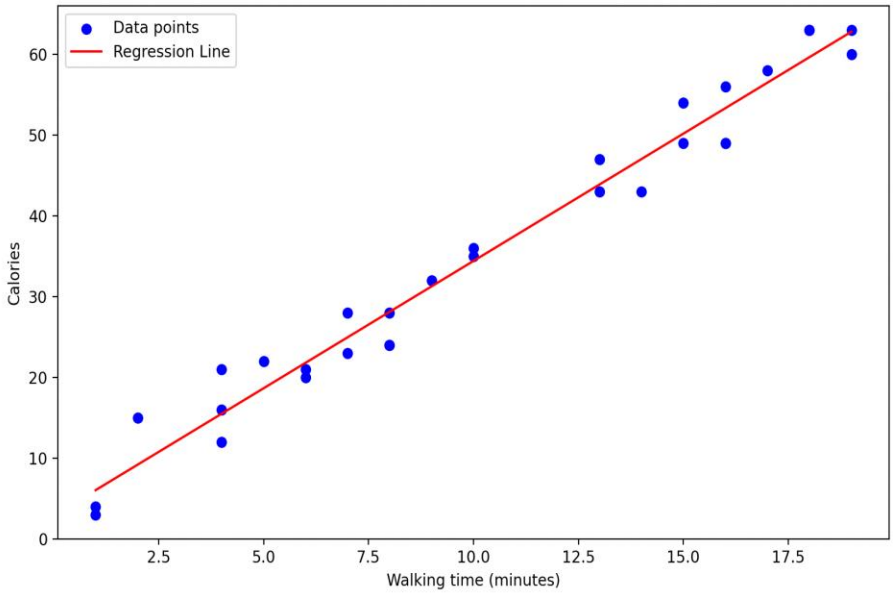
Loss (Error)

- Loss (or Error) is a way for us to measure (and compare) different models

- Mean squared error (MSE):

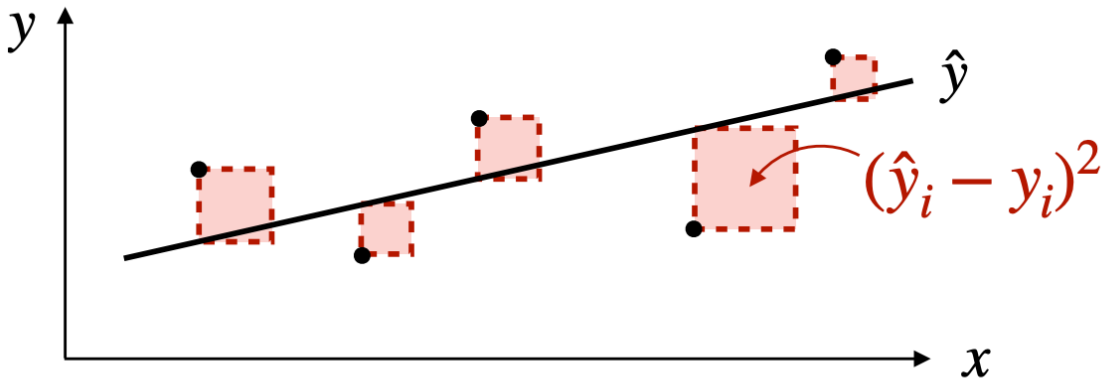
$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

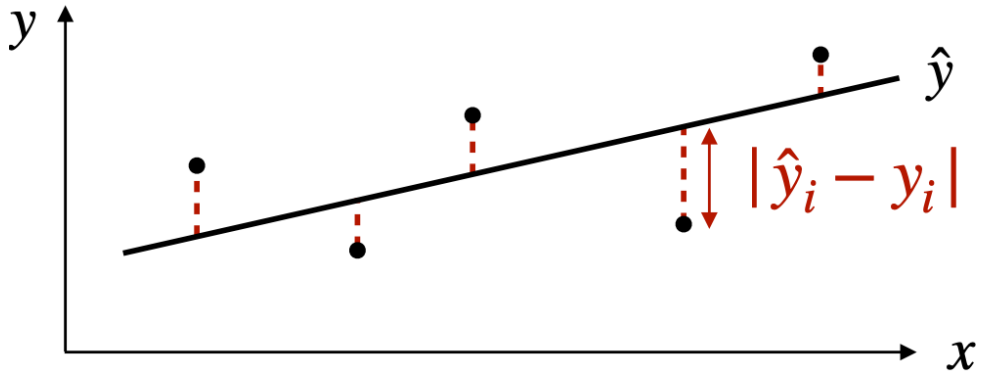
- y_i: target value for data point i (e.g., calories)
- $\hat{y}_i$: model prediction
- n: number of data points



Calories=3.1524×(Walking time)+2.8972

MSE vs MAE





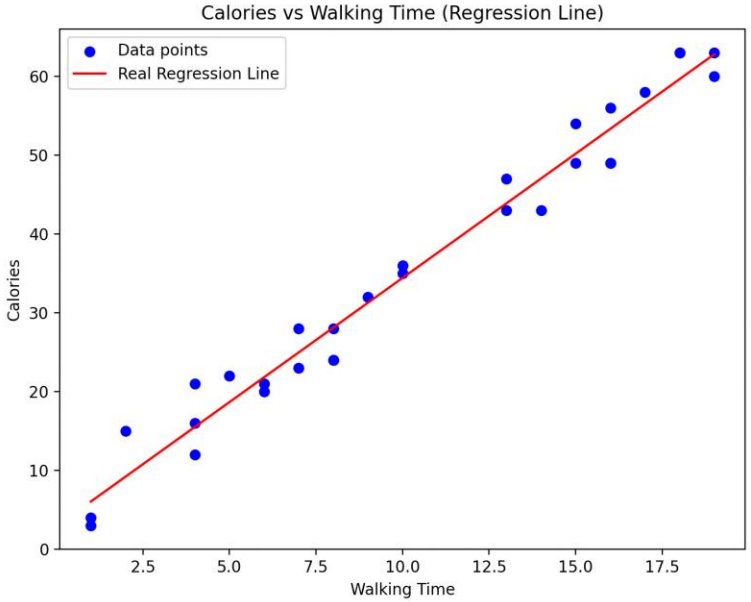
source

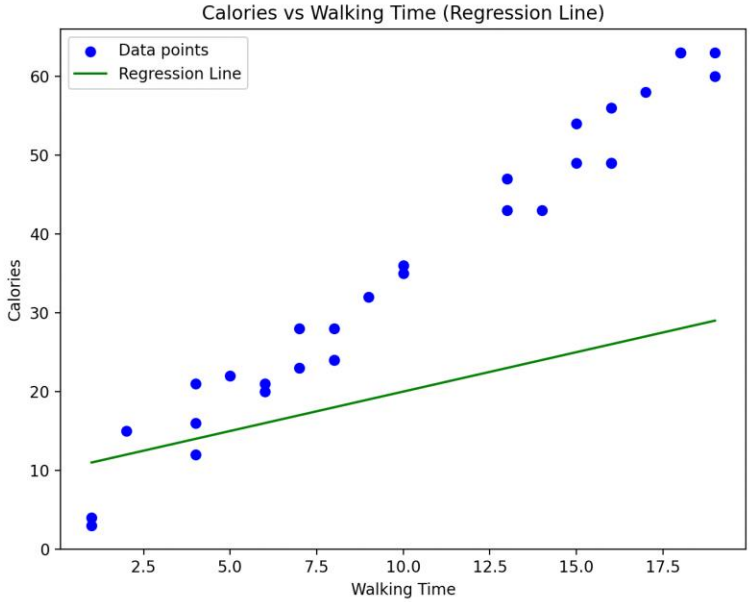
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

MSE (Mean Squared Error) is more affected by outliers than MAE (Mean Absolute Error) because it squares each error, causing larger errors to have a significantly greater impact on the overall loss.

MSE





$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = 9.18$$

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = 382.92$$

Loss Metric Constitute Our Objective

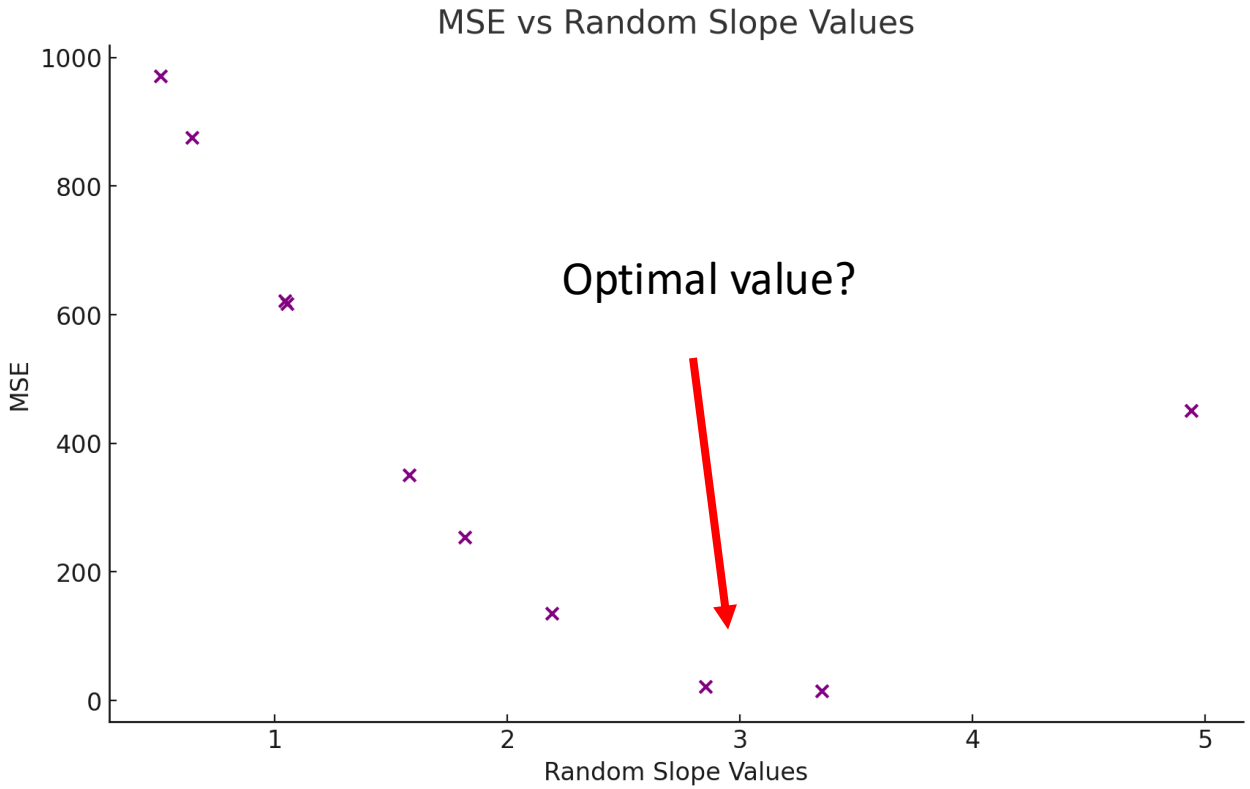
- The Loss or Error Metric is our central objective for optimizing the model's performance
- Minimizing this metric is key to aligning the model more closely with **desired** outcomes
- The choice of an appropriate loss function, such as Mean Squared Error for regression is crucial for the model's effectiveness

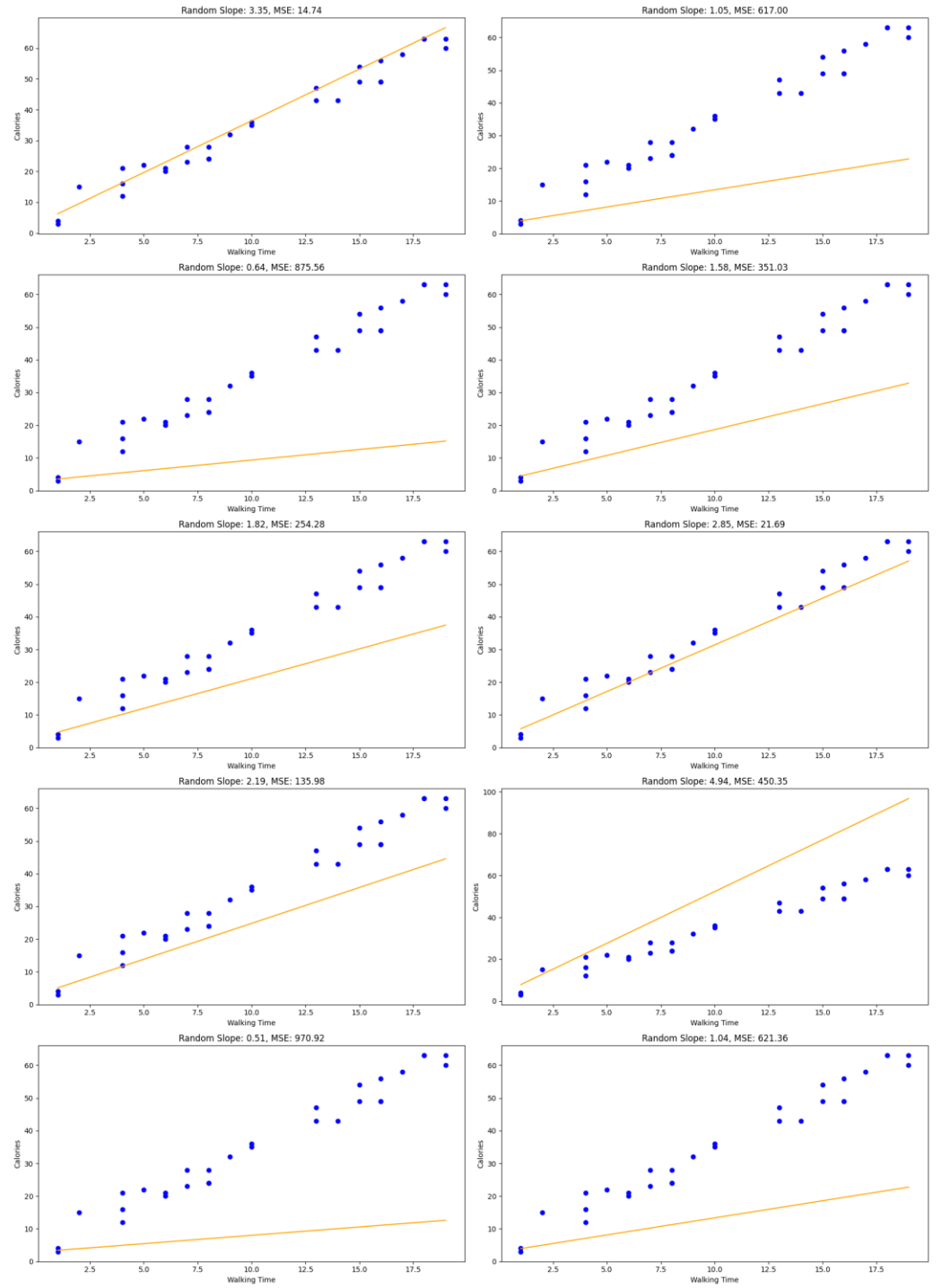
Finding the Best a and b

- Now that we understand how to gauge our model's performance using a loss term, the next step is to identify the optimal set of parameters that minimizes this loss
- Specifically, in the context of linear regression, this translates to determining the ideal values for the coefficients a and b
- Ideal a and b: The pair that **minimizes** the loss!

Brute Force

- Ok, so our first option is to try every single pair of a and b values





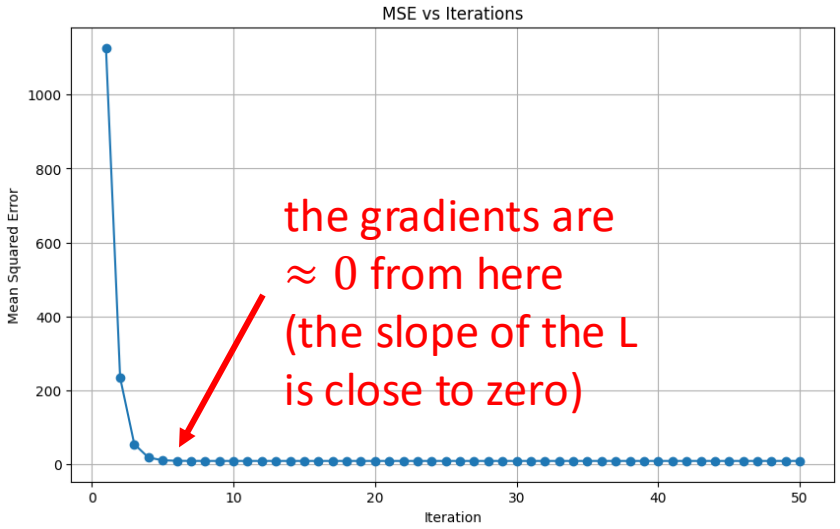
Brute Force: Bad Idea

- Why brute force is a bad idea in practice?
- Brute force methods often involve exploring all possible solutions to a problem, which can be extremely time-consuming and computationally expensive
- As the size of the dataset or the complexity of the problem increases, the time and resources required can grow exponentially, making it impractical for real-world applications
- In many cases, the computational cost becomes prohibitively high, making it impossible to arrive at a solution within a reasonable time frame

Gradient Descent

- Can we use the information from the loss function (L) to **guide** the adjustments to our model parameters (a and b)?
- What if we compute $\frac{\partial L}{\partial a}$ and $\frac{\partial L}{\partial b}$?
- These derivatives gives us the **direction** of change needed for each parameter!
- E.g., If $\frac{\partial L}{\partial a} > 0$: it means we need to **decrease $\boldsymbol{a}$** to reduce L
- E.g., If $\frac{\partial L}{\partial b} < 0$: it means we need to **increase $\boldsymbol{b}$** to reduce L
- The "**descent"** means: we want move in the **opposite** direction of the gradient!
- By **iteratively** updating these parameters in the direction of steepest descent, we can minimize the loss function and optimize the model's performance
- Note that this is only true for **small** amount of changes in a and b!

Linear Regression Derivatives



$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

$$\frac{\partial L}{\partial a} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - (ax_i + b))$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - (ax_i + b))$$

$$a_{new} = a_{old} - \epsilon\frac{\partial L}{\partial a}$$

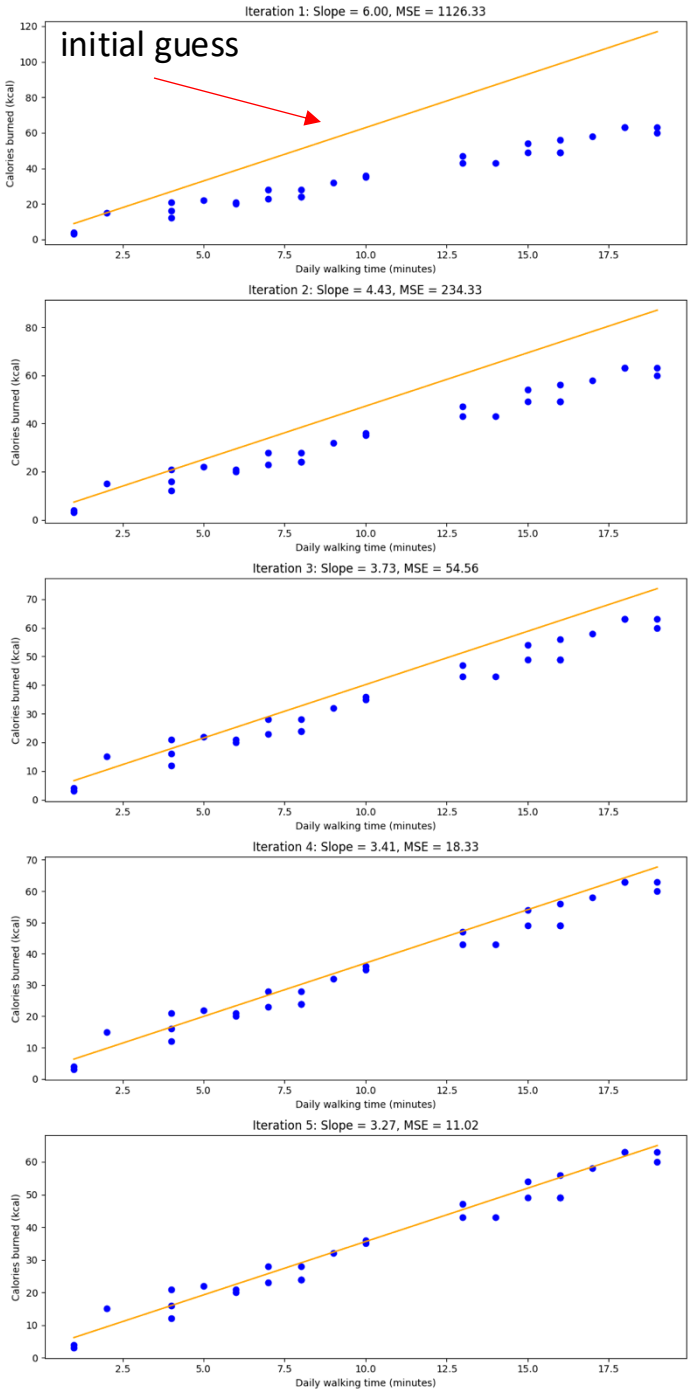
$$b_{new} = b_{old} - \epsilon\frac{\partial L}{\partial b}$$

iterative

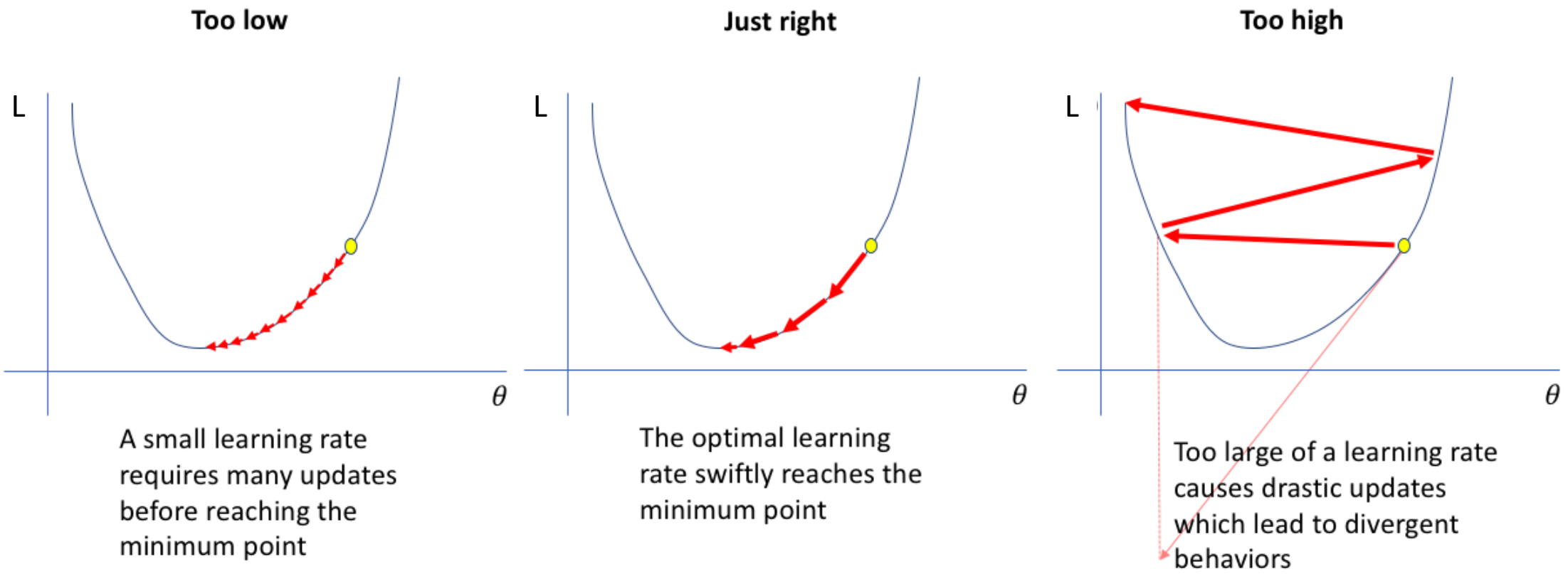
learning rate

- Start from an initial guess for a and b
- Iterate until the gradients are very small!
  - $a_{new} \approx a_{old}, b_{new} \approx b_{old}$

negative since we are moving in the opposite of the gradient!



Learning rate



https://www.jeremyjordan.me/nn-learning-rate/

NOTE! Analytical Solution

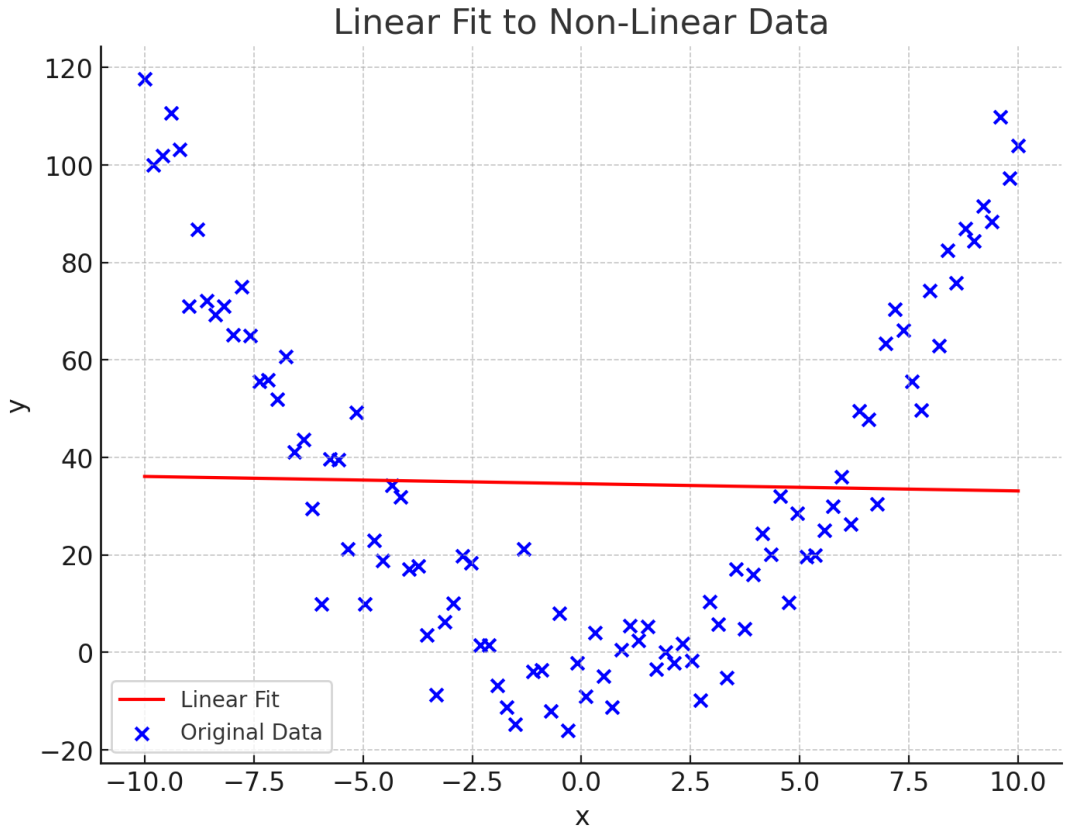
- We use linear regression to exemplify the gradient descent approach
- Unlike many other models where numerical optimization (e.g., Gradient Descent) is required, for linear regression, an exact analytical solution exists:

$$a = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sum(x_i - \overline{x})^2}$$

$$b = \overline{y} - a\overline{x}$$

Non-linearity

- The linear model is not always suitable!



Multiple Linear Regression

- When we have more than 1 feature, we call it multiple linear regression
- $y_i = \alpha + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \varepsilon_i$
- Work likes simple linear regression
- Harder to visualize…

Regression Trees

Using decision trees when :

- The target is numerical
- The features are numerical

Dataset

	review_scores_rating	accommodates	bedrooms	beds	price
0	5.00	10	5.0	7.0	469.0
1	4.84	2	1.0	1.0	94.0
2	4.75	3	1.0	1.0	72.0
3	5.00	2	1.0	2.0	125.0
4	4.63	5	2.0	2.0	100.0
...	...	...	...	...	...
10690	2.00	4	1.0	2.0	165.0
10691	5.00	2	1.0	0.0	200.0
10692	5.00	1	1.0	1.0	47.0
10693	5.00	6	1.0	2.0	95.0
10694	5.00	4	1.0	2.0	194.0

10695 rows × 5 columns

Recap Decision Trees

- Measure entropy (without split)
- Go through each feature:
  - Measure the ”quality” of feature by comparing entropy before and after the split
- Select the best feature (highest information gain)
  - Split the dataset using this feature

Regression Trees

- In a Regression Tree, the prediction for a given input is determined by the mean (average) value of all the instances (data points) that fall within the same leaf node.
- This method hinges on the principle that instances grouped together in a leaf are similar to each other and thus can be represented by a common value, which is their mean.
- Let's compute the error before any split. Before any split is made in the tree, it's important to evaluate the baseline error. This error represents how well the current, unsplit model predicts the outcomes. To compute this, we use all the instances in the dataset to calculate the mean value:

$$\overline{price} = \frac{469+94+72+125+100}{5} = 172$$

- Measure MSE

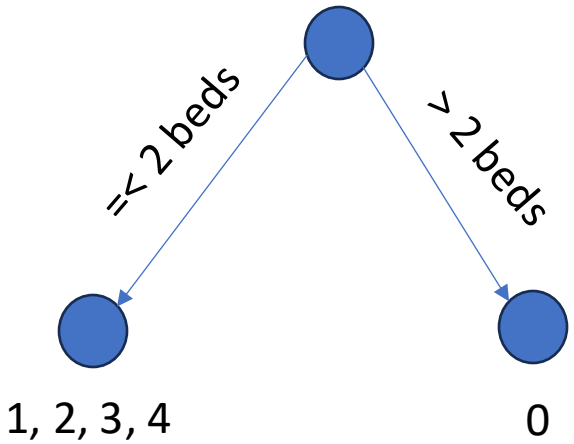
$$\text{MSE} = \frac{1}{5}((469-172)^2 + (94-172)^2 + (72-172)^2 + (125-172)^2 + (100-172)^2) = 12409.55$$

	review_scores_rating	accommodates	bedrooms	beds	price
0	5.00	10	5.0	7.0	469.0
1	4.84	2	1.0	1.0	94.0
2	4.75	3	1.0	1.0	72.0
3	5.00	2	1.0	2.0	125.0
4	4.63	5	2.0	2.0	100.0

Regression Trees

- Does the feature '> 2 beds' helps?
- Measure MSE

$$\text{MSE} = \frac{1}{5}((469-469)^2 + (94-97.75)^2 + (72-97.7)^2 + (125-97.7)^2 + (100-97.7)^2) \approx 285$$



	review_scores_rating	accommodates	bedrooms	beds	price
0	5.00	10	5.0	7.0	469.0
1	4.84	2	1.0	1.0	94.0
2	4.75	3	1.0	1.0	72.0
3	5.00	2	1.0	2.0	125.0
4	4.63	5	2.0	2.0	100.0

average: 469

average: 97.75

How do we go from ‘price’ to ‘> 2 beds’ ?

- Numerical example (Sorted rating):
  - [0.2, 0.5, 1.0, 2.5, 2.8, 3.2, 3.5, 3.5, 6]
- Various solutions:
  - Try every single unique value
  - Discretization: putting values into buckets so that there are a limited number of possible states [1]
    - Quantile-based (e.g., median) (qcut in pandas)
    - Equal-sized interval (cut in pandas)
    - Clustering (next weeks)
    - And more… (e.g., linear programming [2])

[1] https://docs.microsoft.com/en-us/analysis-services/data-mining/discretization-methods-data-mining?view=asallproducts-allversions

[2] http://gnpalencia.org/optbinning/