Raiffeisen AI Course

Introduction to AI

Decoding the Black Box

Introduction on
How Machine Learning Models Make Decisions

Content

Gradient descent

Tree based algorithms

Neural Networks

Sentiment analysis model

Text autocomplete model

Gradient descent - black-box tool

The most popular learning method in Machine Learning

Used in many algorithms from classical ML algorithms to deep neural networks

Many people use it but do not understand it

Gradient descent - a way to minimize an objective function

Derivata de gradul I

Derivata unei funcții f(x) reprezintă rata de variație a funcției în raport cu variabila sa.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$(X^y)' = yX^{y-1}$

$(X^2)' = 2X$

$(E^2)' = 2E * E'$

Speed - Acceleration

T - Time (measured in seconds)
V - speed (km/h * 10)

Interval 0 - 3s:
1.5s - acceleration
1.5s - break

$V(T) = 4T(3 - T)$

Exemplu: *v(t) = 4t(3 - t)*

$$v\ (t) = 12t - 4t^2$$

$$v\ (t) = -4t^2 + 12t$$

$$v'(t) = -8t + 12$$

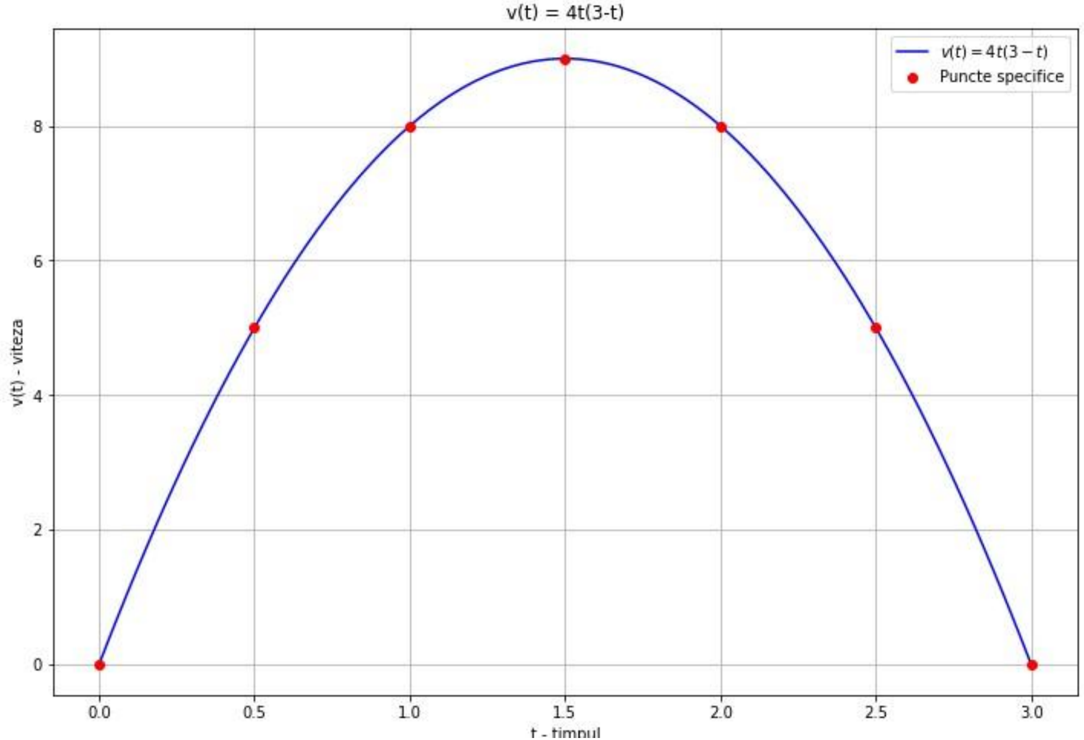
$$v'(0.5) = 8$$

$$v'(1) = 4$$

$$v'(1.5) = 0$$

$$v'(2) = -4$$

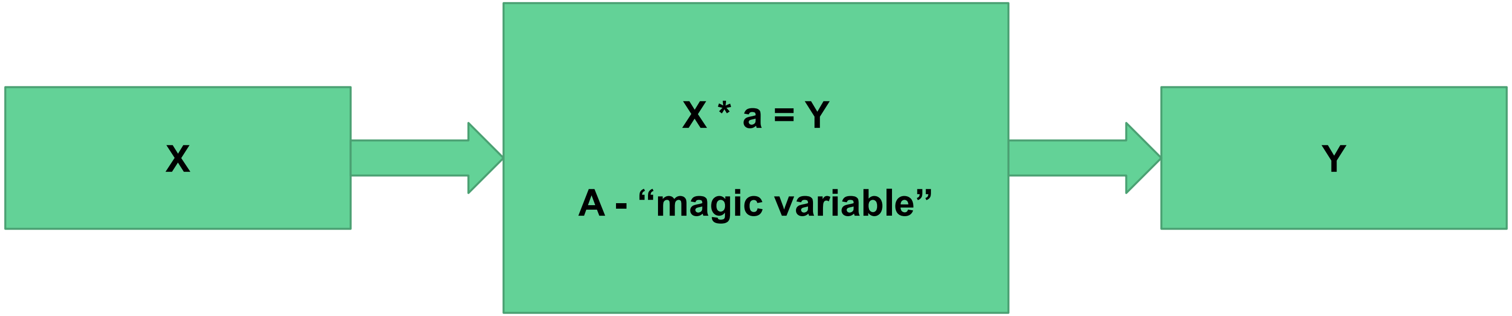
$$v'(2.5) = -8$$



Gradient descent weights

Gradient descent models use weights

X - feature
Y - target
A - weight



Learning step

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_{\text{pred}} - y_i)^2$$

(Ypred - Y)**2

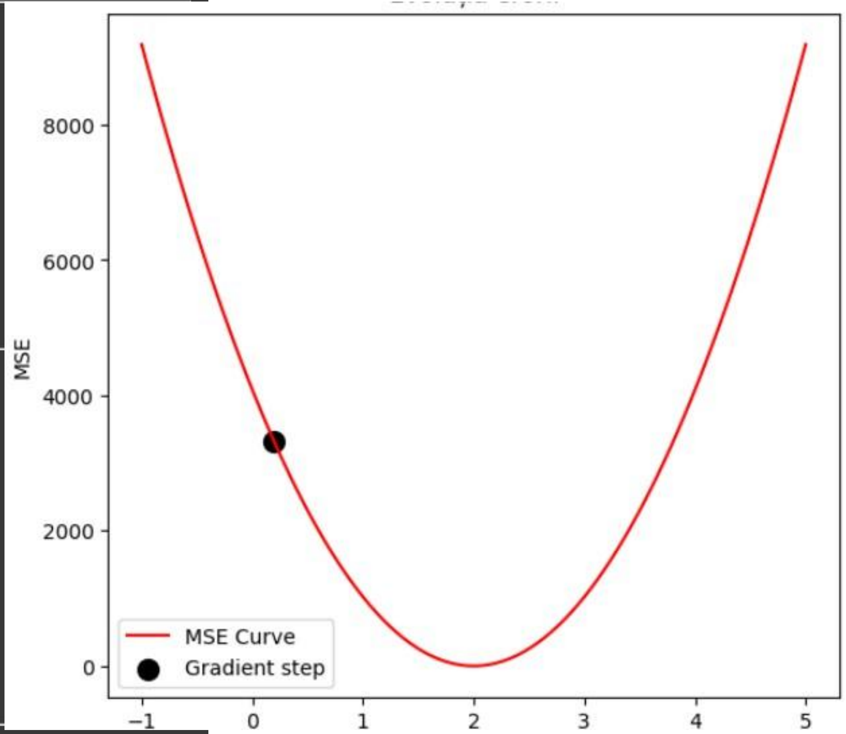
$$\text{ypred} = \text{model prediction}$$
$$\text{ypred} = \text{a * x}$$
$$\text{y} = \text{true value of y}$$

$$\text{error} = (ypred - y)^2$$
$$\text{error} = (a * x - y)^2$$
$$\text{a - random selected weight we want to improve}$$
$$\text{error ---> 0}$$

$$\text{error} = (a \cdot x - y)^2$$
$$\frac{d}{da}\text{error} = 2 \cdot (a \cdot x - y) \cdot \frac{d}{da}(a \cdot x - y)$$
$$\frac{d}{da}(a \cdot x - y) = x$$
$$\frac{d}{da}\text{error} = 2 \cdot (a \cdot x - y) \cdot x$$

$$\text{gradient} = 2 \cdot (a \cdot x - y) \cdot x$$

$$a = a - gradient * learningrate$$



Linear regression

$$f(x) = w_0 + w_1 x$$

Compute the Cost Function

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

Compute the Gradient Descent Update Rules

$$w_0 := w_0 - \eta \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)$$

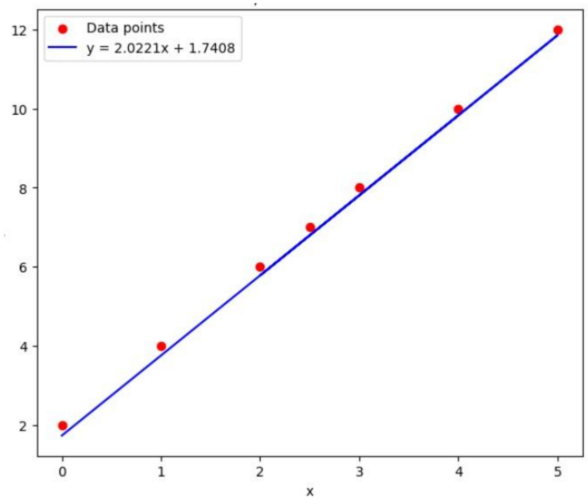
$$w_1 := w_1 - \eta \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i) x_i$$

Repeat Until Convergence

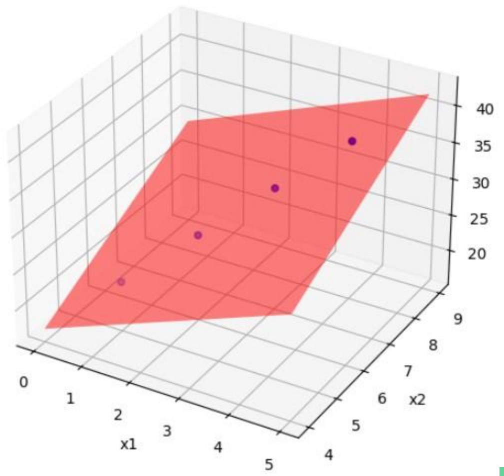
$$w_j := w_j - \eta \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i) x_i$$

Predict Output

$$y_{\text{pred}} = f(x) = w_0 + w_1 x$$



3D Linear Regression: y = ax1 + bx2 + beta



Q1. Gradient descent variants

Which method is better?

- SGD - one point per step
- Batch GD - all points per step

Variant	**Description**	**Time**	**Memory**
Stochastic GD	One point	?	?
Batch GD	All points	?	?

Gradient descent variants

Variant	Description	Time	Memory
Stochastic GD	One point	Slow	Low
Batch GD	All points	Fast	High

Gradient descent variants

Variant	Description	Time	Memory
Stochastic GD	One point	Slow	Low
Batch GD	All points	Fast	High

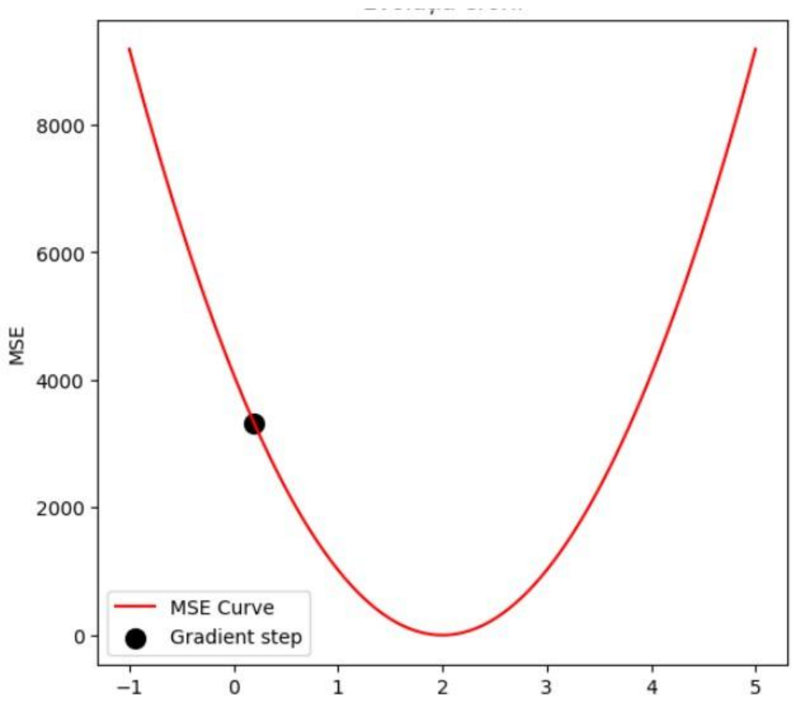
Variant	Description	Time	Memory
Stochastic GD	One point	Slow	Low
Batch GD	All points	Fast	High
Mini-Batch GD	Batches	Medium	Medium

Q2. Starting point

If we start the gradient from 2 random points both will go in the same point?

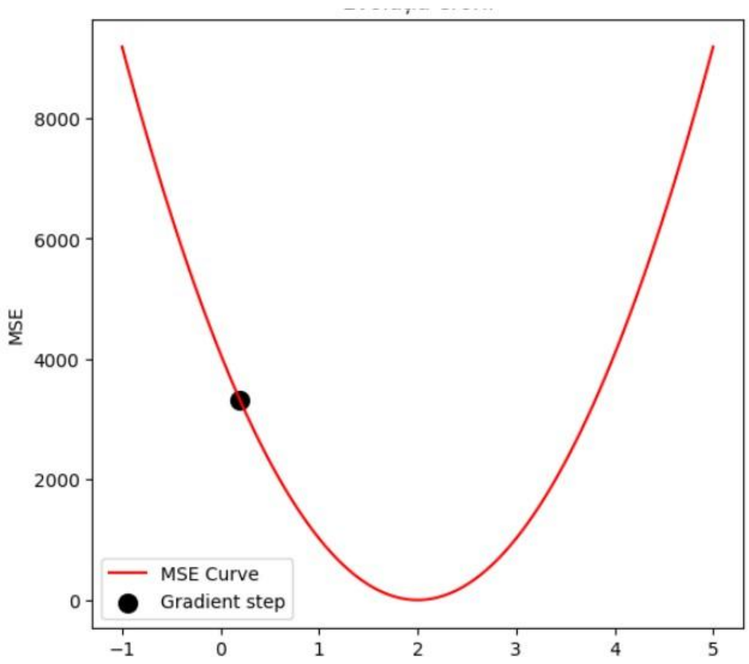
Q2. Starting point

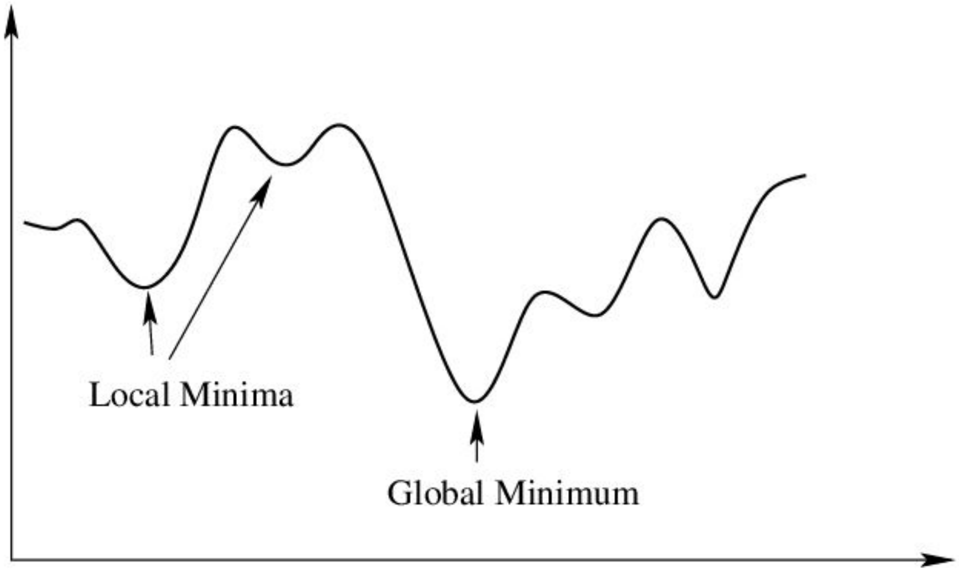
If we start the gradient from 2 random points both will go in the same point?



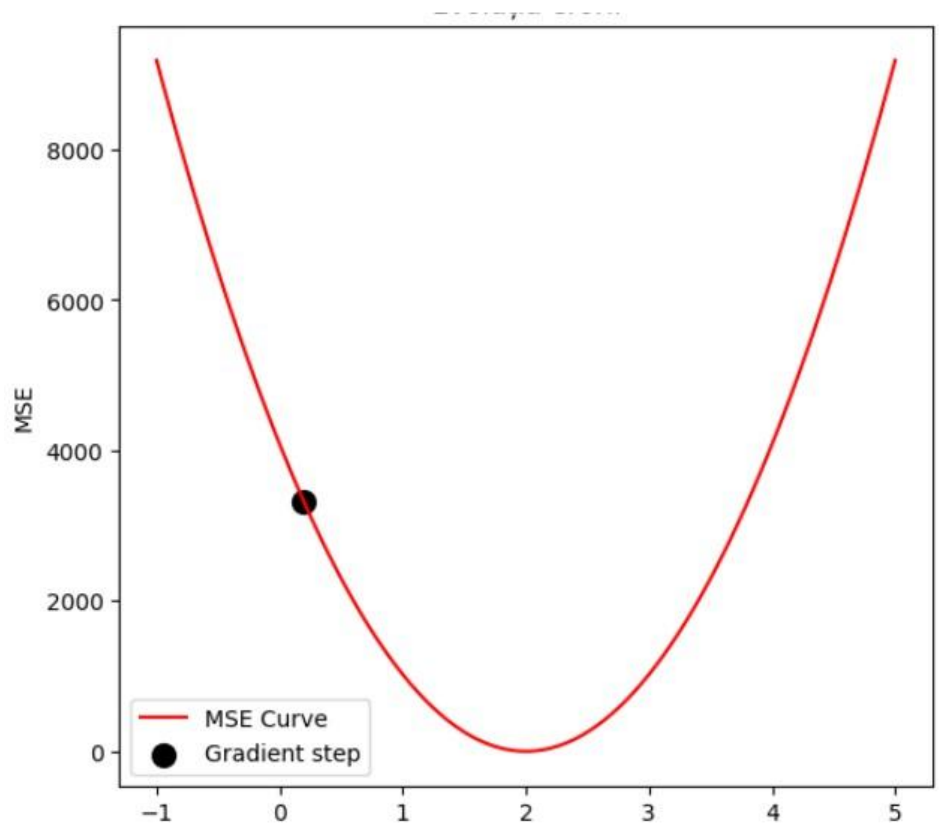
Q2. Starting point

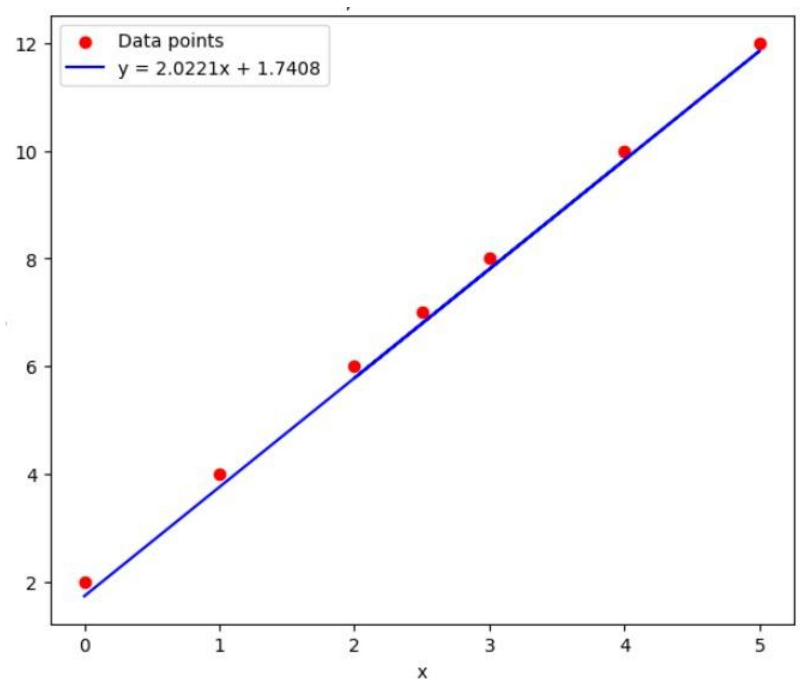
If we start the gradient from 2 random points both will go in the same point?





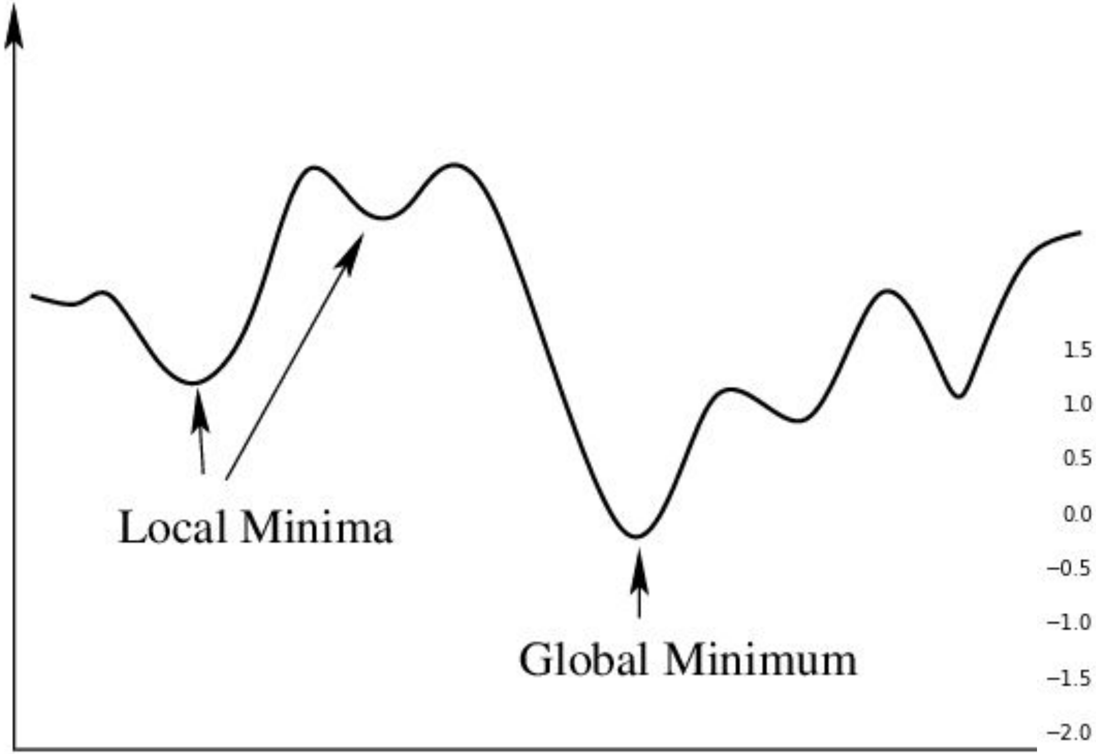
Q3. Will GD conduct to ZERO error?

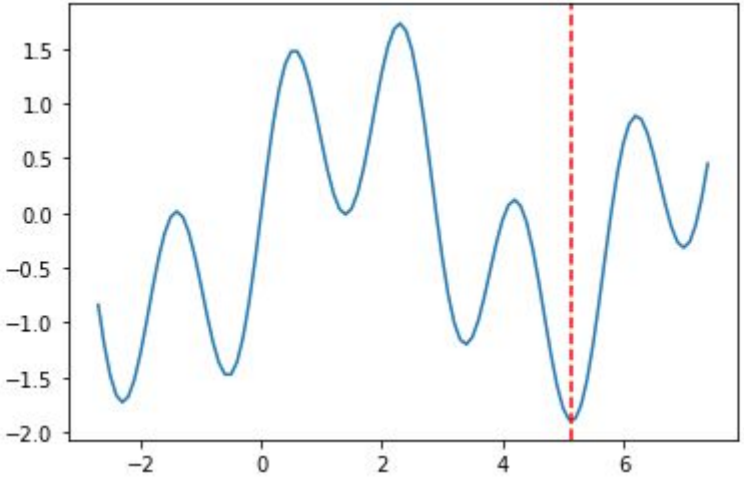




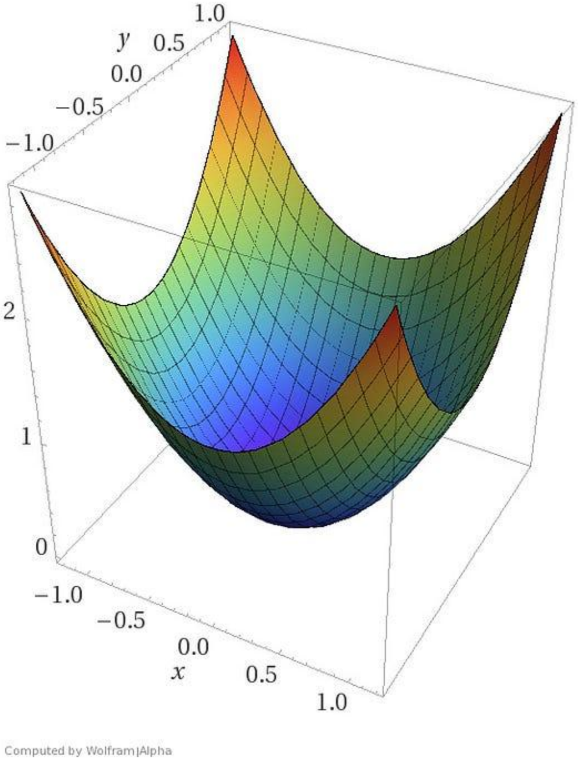
Q4. We always need global minimum?

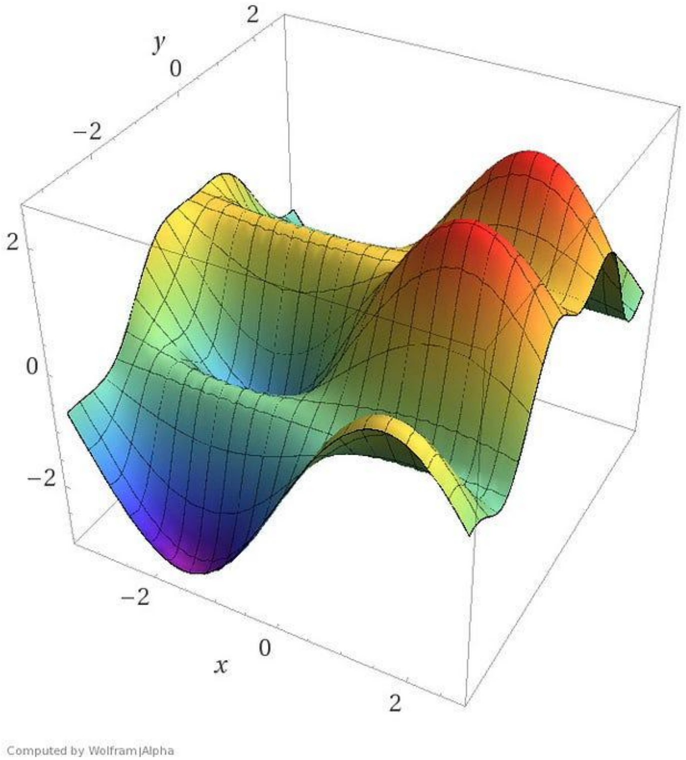
Q4. We always need global minimum?





Local and global minim



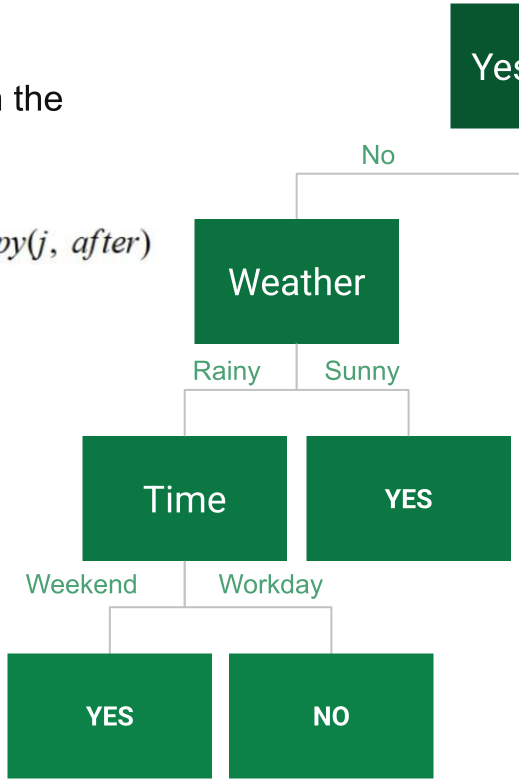


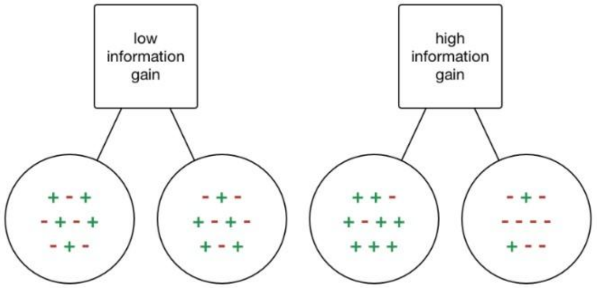
Decision Tree

Entropy is a measure of the randomness in the information being processed

$$Information\ Gain = Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Yesterday	Weather	Time	Decision
No	Sunny	Weekend	**YES**
No	Sunny	Workday	**YES**
Yes	Rainy	Weekend	**NO**
No	Rainy	Workday	**NO**
Yes	Sunny	Workday	**NO**
Yes	Sunny	Weekend	**NO**
No	Rainy	Weekend	**YES**





The **total entropy** of the target variable ("Decision") before any splitting is calculated as follows:

We have:

- **Yes**: 3 (F1, F2, F7)
- **No**: 4 (F3, F4, F5, F6)

Total instances = 7

The formula for entropy is:

$$H(S) = -p_1 \log_2(p_1) - p_2 \log_2(p_2)$$

Where:

- $p_1 = \frac{3}{7}$ (Yes)
- $p_2 = \frac{4}{7}$ (No)

$$H(S) = -\left(\frac{3}{7}\log_2\left(\frac{3}{7}\right)\right) - \left(\frac{4}{7}\log_2\left(\frac{4}{7}\right)\right)$$

$$H(S) = -(0.4286 \log_2(0.4286)) - (0.5714 \log_2(0.5714))$$

$$H(S) \approx 0.985$$

So, the total entropy of the target variable "Decision" is approximately **0.985**.

Yes (F3, F5, F6):

- **Yes**: 0
- **No**: 3 (F3, F5, F6)

$$H(\text{Yesterday} = \text{Yes}) = -\left(\frac{0}{3}\log_2\left(\frac{0}{3}\right)\right) - \left(\frac{3}{3}\log_2\left(\frac{3}{3}\right)\right) = 0$$

No (F1, F2, F4, F7):

- **Yes**: 3 (F1, F2, F7)
- **No**: 1 (F4)

$$H(\text{Yesterday} = \text{No}) = -\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right)\right) - \left(\frac{1}{4}\log_2\left(\frac{1}{4}\right)\right)$$

$$H(\text{Yesterday} = \text{No}) = 0.811$$

Weighted Entropy After Split by "Yesterday"

Now, calculate the **weighted entropy** for "Yesterday":

$$H(\text{Yesterday}) = \frac{3}{7} \times H(\text{Yesterday} = \text{Yes}) + \frac{4}{7} \times H(\text{Yesterday} = \text{No})$$

$$H(\text{Yesterday}) = \frac{3}{7} \times 0 + \frac{4}{7} \times 0.811$$

$$H(\text{Yesterday}) \approx 0.463$$

Sunny (F1, F2, F5, F6):

- **Yes:** 2 (F1, F2)
- **No:** 2 (F5, F6)

$$H(\text{Sunny}) = -\left(\frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) - \left(\frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 1$$

Rainy (F3, F4, F7):

- **Yes:** 1 (F7)
- **No:** 2 (F3, F4)

$$H(\text{Rainy}) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) - \left(\frac{2}{3}\log_2\left(\frac{2}{3}\right)\right)$$

$$H(\text{Rainy}) \approx 0.918$$

Weighted Entropy After Split by "Weather"

Now, calculate the **weighted entropy** for "Weather":

$$H(\text{Weather}) = \frac{4}{7} \times H(\text{Sunny}) + \frac{3}{7} \times H(\text{Rainy})$$

$$H(\text{Weather}) = \frac{4}{7} \times 1 + \frac{3}{7} \times 0.918$$

$$H(\text{Weather}) \approx 0.957$$

3.1: **Information Gain for "Yesterday"**

$$\text{Information Gain (Yesterday)} = H(S) - H(\text{Yesterday})$$

$$\text{Information Gain (Yesterday)} = 0.985 - 0.463$$

$$\text{Information Gain (Yesterday)} \approx 0.522$$

3.2: **Information Gain for "Weather"**

$$\text{Information Gain (Weather)} = H(S) - H(\text{Weather})$$

$$\text{Information Gain (Weather)} = 0.985 - 0.957$$

$$\text{Information Gain (Weather)} \approx 0.028$$

3.3: **Information Gain for "Time"**

$$\text{Information Gain (Time)} = H(S) - H(\text{Time})$$

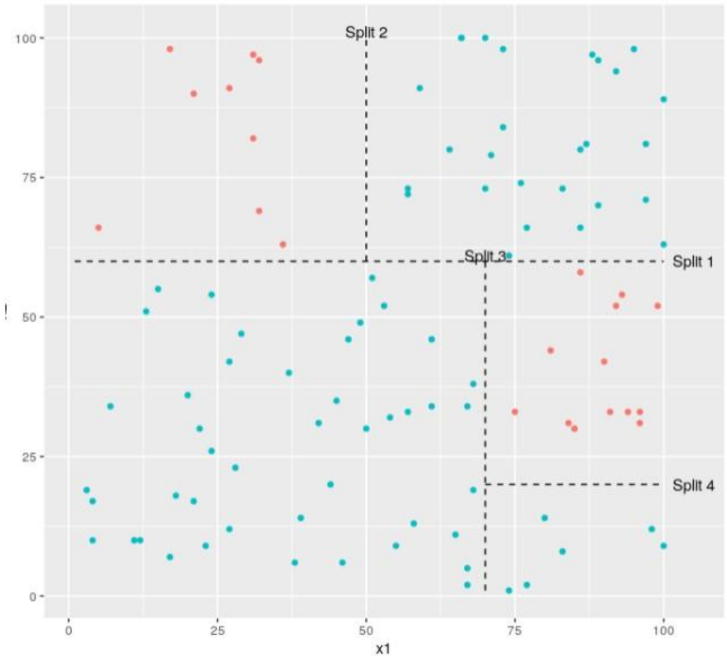
$$\text{Information Gain (Time)} = 0.985 - 0.957$$

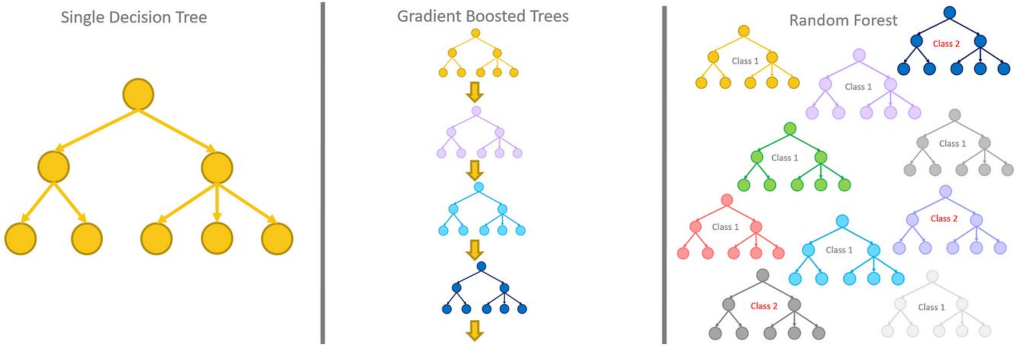
$$\text{Information Gain (Time)} \approx 0.028$$

Bagging and boosting

Underfitting **Overfitting**

Model complexity





Bagging - reduces complexity (reduce overfitting)
Boosting - increase complexity (reduce underfitting)

Q1

Which model is more complex and learn more:

- One decision tree with all features and samples
- Two decision trees with features and samples splitted

Q2

Can we use string features for categorical variables is a decision tree?

Q3

How high correlated features may affect a decision tree?

Q4

How a decision tree works with outliers

Text features

Models works only with numeric features

How we transform text features in numeric features

- Label Encoding
- One Hot Encoder
- TF-IDF vectorizer

TF-IDF Formula:

The **TF-IDF** of a term t in a document d is computed as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Where:

1. **TF (Term Frequency)** is a measure of how frequently a term t appears in a document d. It's typically calculated as:

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

2. **IDF (Inverse Document Frequency)** measures the importance of the term t in the entire corpus (set of documents). It is calculated as:

$$\text{IDF}(t) = \log\left(\frac{N}{\text{df}(t)}\right)$$

- **Document 1 (D1):** "apple orange apple"
- **Document 2 (D2):** "apple banana fruit"
- **Document 3 (D3):** "banana fruit apple"

1. **Document 1 (D1)** has 3 terms ("apple", "orange", "apple"), and the term "apple" appears 2 times.

$$\mathrm{TF}(apple, D1) = \frac{2}{3} = 0.6667$$

2. **Document 2 (D2)** has 3 terms ("apple", "banana", "fruit"), and the term "apple" appears 1 time.

$$\mathrm{TF}(apple, D2) = \frac{1}{3} = 0.3333$$

3. **Document 3 (D3)** has 3 terms ("banana", "fruit", "apple"), and the term "apple" appears 1 time.

$$\mathrm{TF}(apple, D3) = \frac{1}{3} = 0.3333$$

$$\mathrm{IDF}(apple) = \log\left(\frac{N}{\mathrm{df}(apple)}\right) = \log\left(\frac{3}{3}\right) = \log(1) = 0$$

1. **For Document 1 (D1):**

$$\text{TF-IDF}(apple, D1) = \mathrm{TF}(apple, D1) \times \mathrm{IDF}(apple) = 0.6667 \times 0 = 0$$

2. **For Document 2 (D2):**

$$\text{TF-IDF}(apple, D2) = \mathrm{TF}(apple, D2) \times \mathrm{IDF}(apple) = 0.3333 \times 0 = 0$$

3. **For Document 3 (D3):**

$$\text{TF-IDF}(apple, D3) = \mathrm{TF}(apple, D3) \times \mathrm{IDF}(apple) = 0.3333 \times 0 = 0$$

- **Document 1 (D1):** "apple orange apple" → TF(banana, D1) = 0 (doesn't appear).
- **Document 2 (D2):** "apple banana fruit" → TF(banana, D2) = 1/3.
- **Document 3 (D3):** "banana fruit apple" → TF(banana, D3) = 1/3.

$$\text{IDF}(banana) = \log\left(\frac{3}{2}\right) \approx \log(1.5) \approx 0.1761$$

1. **For Document 1 (D1):**

$$\text{TF-IDF}(banana, D1) = \text{TF}(banana, D1) \times \text{IDF}(banana) = 0 \times 0.1761 = 0$$

2. **For Document 2 (D2):**

$$\text{TF-IDF}(banana, D2) = \text{TF}(banana, D2) \times \text{IDF}(banana) = \frac{1}{3} \times 0.1761 \approx 0.0587$$

3. **For Document 3 (D3):**

$$\text{TF-IDF}(banana, D3) = \text{TF}(banana, D3) \times \text{IDF}(banana) = \frac{1}{3} \times 0.1761 \approx 0.0587$$

Output example

	w1	w2	w3	w4	w5	w6
D1	0	0.157	0.053	0	0.021	0.251
D2	0.027	0	0	0.058	0.027	0
D3	0	0.058	0.053	0	0.027	0.072
D4	0.002	0	0.157	0.027	0	0.157

Questions

Q1. How is TF-IDF in terms of natural language understanding?

Q2. How is TF-IDF in terms of memory?

Neural Networks

$$a_1 = (w_{1,1}x_1) + (w_{2,1}x_2)$$
$$= (0.2 * 0.35) + (0.2 * 0.7)$$
$$= 0.21$$

$$y_j = F(a_j) = \frac{1}{1+e^{-a_1}}$$

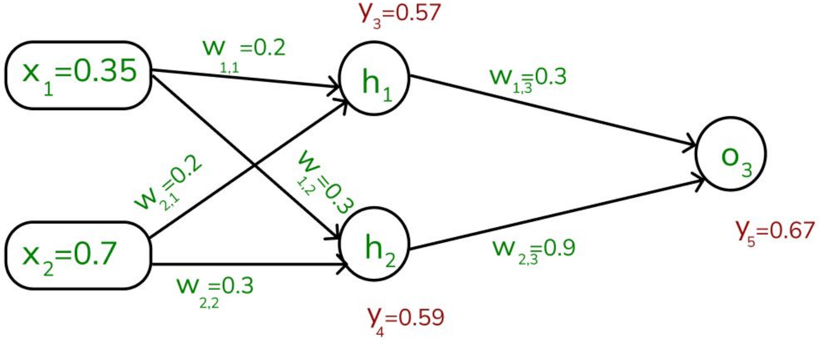
$$y_3 = F(0.21) = \frac{1}{1+e^{-0.21}}$$

$$y_3 = 0.56$$

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

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

$$\Delta w_{ij} = \eta \cdot \delta_j \cdot O_i$$



For O3:

$$\delta_5 = y_5(1-y_5)(y_{target} - y_5)$$
$$= 0.67(1-0.67)(-0.17) = -0.0376$$

For h1:

$$\delta_3 = y_3(1-y_3)(w_{1,3} \times \delta_5)$$
$$= 0.56(1-0.56)(0.3 \times -0.0376) = -0.0027$$

For h2:

$$\delta_4 = y_4(1-y_4)(w_{2,3} \times \delta_5)$$
$$= 0.59(1-0.59)(0.9 \times -0.0376) = -0.0819$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_i}$$

$$\Delta w_{2,3} = 1 \times (-0.0376) \times 0.59 = -0.022184$$

$$E = \frac{1}{2}(y_{\text{target}} - y_j)^2$$

$$w_{2,3}(\text{new}) = -0.22184 + 0.9 = 0.67816$$

$$\frac{\partial E}{\partial y_j} = -(y_{\text{target}} - y_j)$$

$$\Delta w_{1,1} = 1 \times (-0.0027) \times 0.35 = 0.000945$$

$$y_j = \frac{1}{1+e^{-net_j}}$$

$$w_{1,1}(\text{new}) = 0.000945 + 0.2 = 0.200945$$

$$\frac{\partial y_j}{\partial net_j} = y_j(1-y_j)$$

Questions

What is the role of activation functions in hidden layers?

Word embeddings

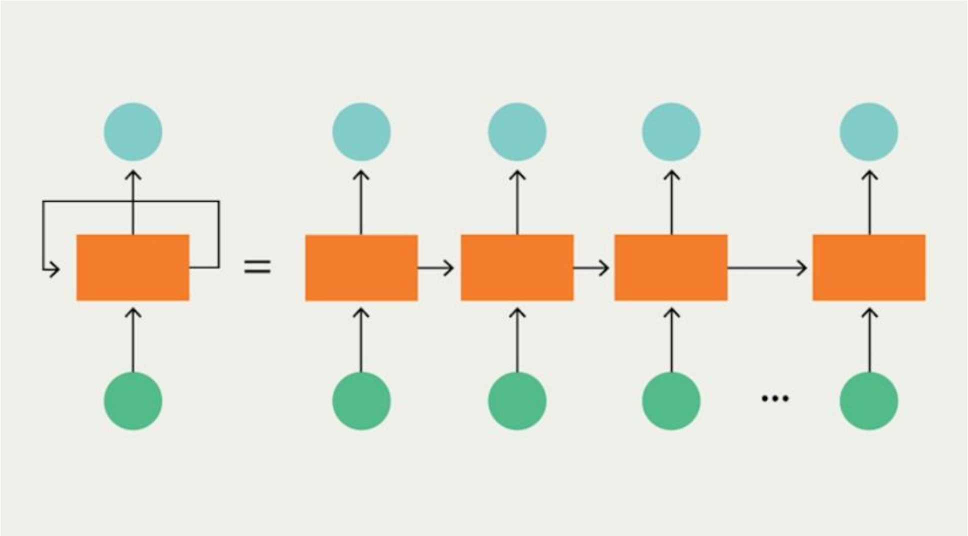
A word embedding is a learned representation of a word such that the words with same meaning have similar representation.

The geometric relationship between words should reflect the semantic relationship.

The vocabulary is predefined and learned over a large corpus of text.

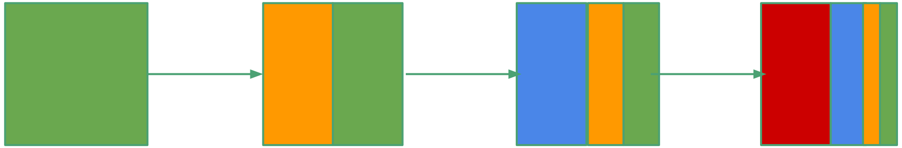
KING - MAN + WOMAN = QUEEN

Recurrent neural networks



[dog, cat, fox]

[[0.1, 0.2, 0.3, 0.4],
 [0.5, 0.6, 0.7, 0.8],
 [0.9, 0.1, 0,5, 0.7]]



Demo

Text autocomplete

- Word2Vec
- RNN - LSTM
- Evaluation

Transformers

Like the RNNs, transformers can handle distant information

RNNs struggle with long sequence data.

Transformers are not based on recurrent connections

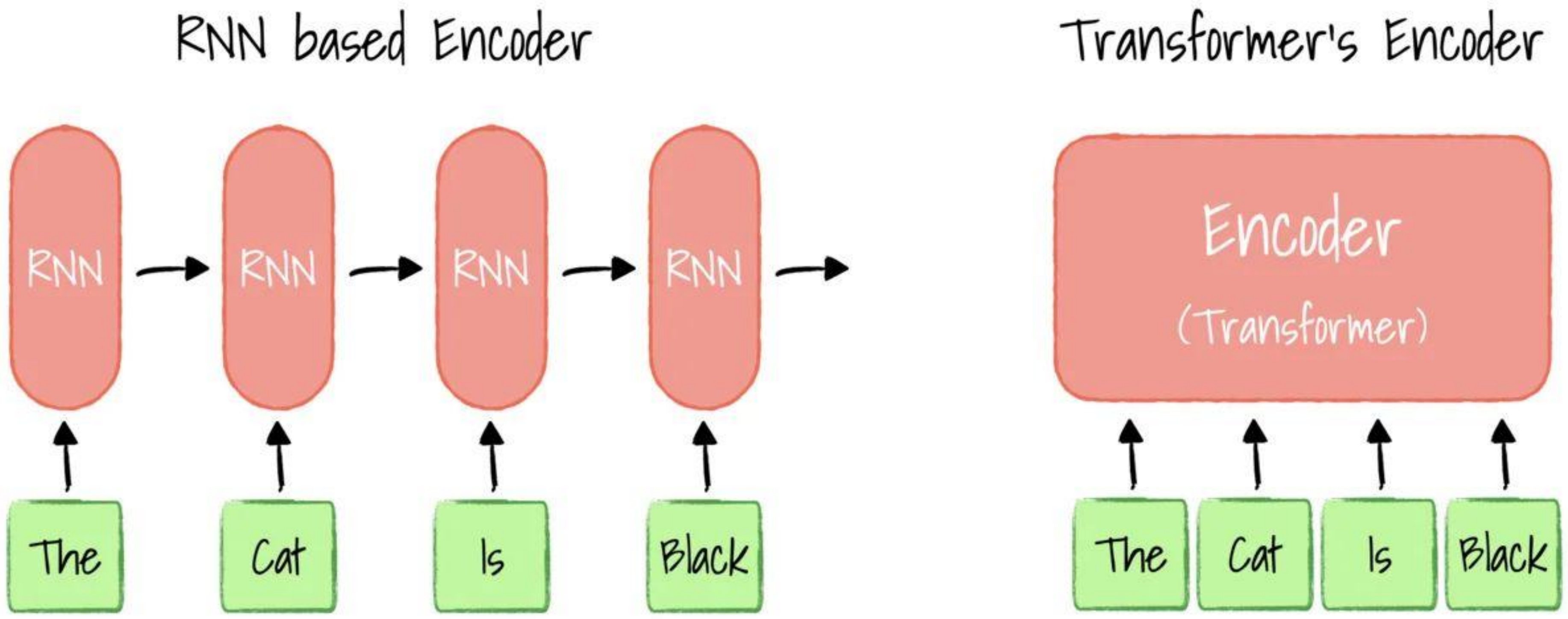
Transformers are more efficient to implement at scale

Transformers are made up of stacks of transformer blocks, each of which is a multilayer network made by combining:

- simple linear layers
- feedforward networks
- self-attention layers

Self-attention allows a network to directly extract and use information from arbitrarily large contexts without the need to pass it through intermediate recurrent connections as in RNNs

Transformers



Questions?