Data Science Bootcamp

Hyperiondev

Stochastic Gradient Descent

WELCOME

Your Lecturer for This Session

Alfred Ndlovu

Lecture - Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- You can also submit questions here: hyperiondev.com/sbc4-ds-questions
- For all non-academic questions, please submit a query: hyperiondev.com/support
- Report a safeguarding incident: hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Lecture - Code Repo

Go to: github.com/HyperionDevBootcamps

Then click on the “**C4_DS_lecture_examples**” repository, do view or download the code.

- **POLL**

Objectives

1. Understand stochastic gradient descent
2. How it works in linear regression

Stochastic Gradient Descent?

- **Its also an optimization algorithm used in machine learning to find the minimum (or maximum) of a function.**

- **It is used to train models by minimizing the error between predicted and actual values.**
  - Predicted - what the model returns
  - Actual Values - what was meant to be returned

So what happens during training?

- The model takes the input data "x_train" and makes predictions.

- The predictions are compared with the corresponding true values in "y_train" to calculate the error or loss.

- The model adjusts its parameters using an optimization algorithm such as gradient descent to minimize the loss.

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Machine Learning Model(Linear Regression)

$$price = m * area + c$$

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Machine Learning Model(Linear Regression)

price = 1 * area + 1

predicted_price = 2601

Actual_price = 550000

error = (Actual_price - predicted_price)^2

Total_Error = error 1 + error 2.....+ error 5

Mean Squared Error = Total Error / 5

$m = m - \text{learning rate} * d(MSE)/dm$ $m = 1 - (-30) = 31$

$C = c - \text{learning rate} * d(MSE)/dc$ $C = 1 - (-3000) = 30001$

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Machine Learning Model(Linear Regression)

price = 31 * area + 30001

predicted_price = 110 601

Actual_price = 550000

error = (Actual_price - predicted_price)^2

BATCH GRADIENT DESCENT

We are basically calculating the gradient of the cost function with respect to each parameter by considering ALL the examples in the training dataset.

BATCH GRADIENT DESCENT

1. We go through all the **training dataset** calculate cumulative error

2. **And then adjust parameters**

What if we had 1 million training samples ???

A	B	C	D	E	F
louse	Price ($M)	House (K sf)	Lot (K sf)	Bedrooms	Bathrooms
louse 1	6	6.9	42.7	6	9
louse 2	5.8	8	36.6	6	8
louse 3	5.6	8	44	7	7
louse 4	3.5	3.8	18	4	4
louse 5	3.4	6.1	27.4	5	5
louse 6	3.4	4.3	22.2	5	4
louse 7	2.7	3.8	22	4	5
louse 8	2.6	5	29.3	4	5
louse 9	2.6	3.6	31.4	4	4
louse 10	2.3	3.1	22.2	4	5
louse 11	2.3	3.9	21.7	5	4
louse 12	2.3	3.2	24.4	4	4
louse 13	1.9	3.5	25.3	4	3
louse 14	1.9	3.4	24	5	4
louse 15	1.9	3.2	21.8	4	4
louse 16	1.6	3.3	6.6	4	4
louse 17	1.6	2.3	15.9	4	4
louse 18	1.5	2.3	21.2	3	3

1. To find cumulative error(epoch) for the first round we need to pass 1 million samples

Thats too much computation and your model will take forever to train

1. Randomly pick a single training sample

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Machine Learning Model(Linear Regression)

price = 1 * area + 1

predicted_price = 2601

Actual_price = 550000

error = (Actual_price - predicted_price)^2

2. Adjust your parameters soon after

- $m = m - \text{learning rate} * d(\text{error})/dm$ $m = 1 - (-30) = 31$
- $c = c - \text{learning rate} * d(\text{error})/dc$ $C = 1 - (-3000) = 30001$

STOCHASTIC GRADIENT DESCENT

We are basically calculating the gradient of the cost function with respect to each parameter by considering ONE example in the training dataset.

BUT ONE RANDOM SAMPLE PER epoch ?

mini - batch gradient descent

We are basically calculating the gradient of the cost function with respect to each parameter by considering a BATCH of examples in

training dataset.

Its like SGD but instead of randomly choosing a single training sample, you use a batch of randomly picked training samples

- You have 10 training samples ?
- Just use 4 samples to calculate the cumulative error
- Then adjust your parameters

MAIN DIFFERENCEs

Batch Gradient Descent	Stochastic Gradient Descent	Mini Batch Gradient Descent
1. Uses ALL training samples for one pass(for each epoch)	Uses ONE(randomly) picked samples for one pass and then updates parameters	Uses Batches
2. Good for small training datasets	Good for big data	Good for both

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any

Hyperiondev

Thank you for joining us