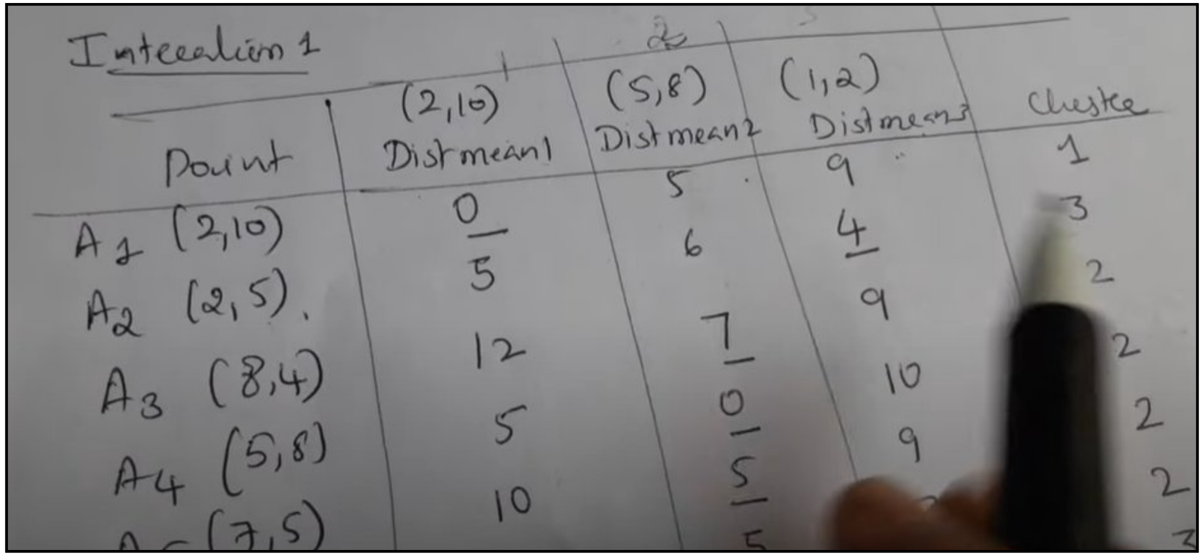
Performing Differentiation More on Regression

What to do when the number of features is 3 or more for Linear Regression?

Logistic Regression

What can we do after learning NumPy?

- The arithmetic calculations which we usually do by hand, can be done via code using NumPy.
- Some intelligent tasks like – grouping and averaging all the points belonging to the same cluster can be easily done using NumPy.



But, can we use NumPy to perform differentiation?

- We can perform differentiation by hand.
- $L = w^2$
- Determine $\frac{dL}{dw}$ by hand.
- What will be the value of $\frac{dL}{dw}$ for w=4?
- Now the question is, can this be done using NumPy?

Introducing TensorFlow

- Want to build Machine Learning/ Deep Learning model from scratch?
- TensorFlow can be your solution (for the differentiation based models.)
- Now, let's use TensorFlow to perform some differentiations.

TensorFlow and Differentiation

Have you done this by hand yet?

- $L = w^2$
- Determine $\frac{dL}{dw}$ by hand.
- What will be the value of $\frac{dL}{dw}$ for w=4?

TensorFlow and Differentiation

Have you done this by hand yet?

- $L = w^2$
- Determine $\frac{dL}{dw}$ by hand.
- What will be the value of $\frac{dL}{dw}$ for w=4?

```python
import tensorflow as tf

w = tf.Variable(4.0)


with tf.GradientTape() as tape:

    L = w**2


grad = tape.gradient(L, w)

print(f"The value of dL/dw when w=4 is: {grad.numpy()}")
```

```
The value of dL/dw when w=4 is: 8.0
```

Another One

- $z = 3 + 10x$
- $a = \frac{1}{1+e^{-z}}$
- $\frac{da}{dx} = ?$
- What is the value of $\frac{da}{dx}$ for x=2?

Another One

- $z = 3 + 10x$
- $a = \frac{1}{1+e^{-z}}$
- $\frac{da}{dx} = ?$
- What is the value of $\frac{da}{dx}$ for x=5?

```python
import tensorflow as tf


x = tf.Variable([5.0], dtype=tf.float32)

with tf.GradientTape(persistent=True) as tape:
    z = 3 + 10 * x
    a = 1 / (1 + tf.exp(-z))

da_by_dx = tape.gradient(a, x)

result= da_by_dx.numpy()
print("Gradient da/dx:", da_by_dx.numpy())
```

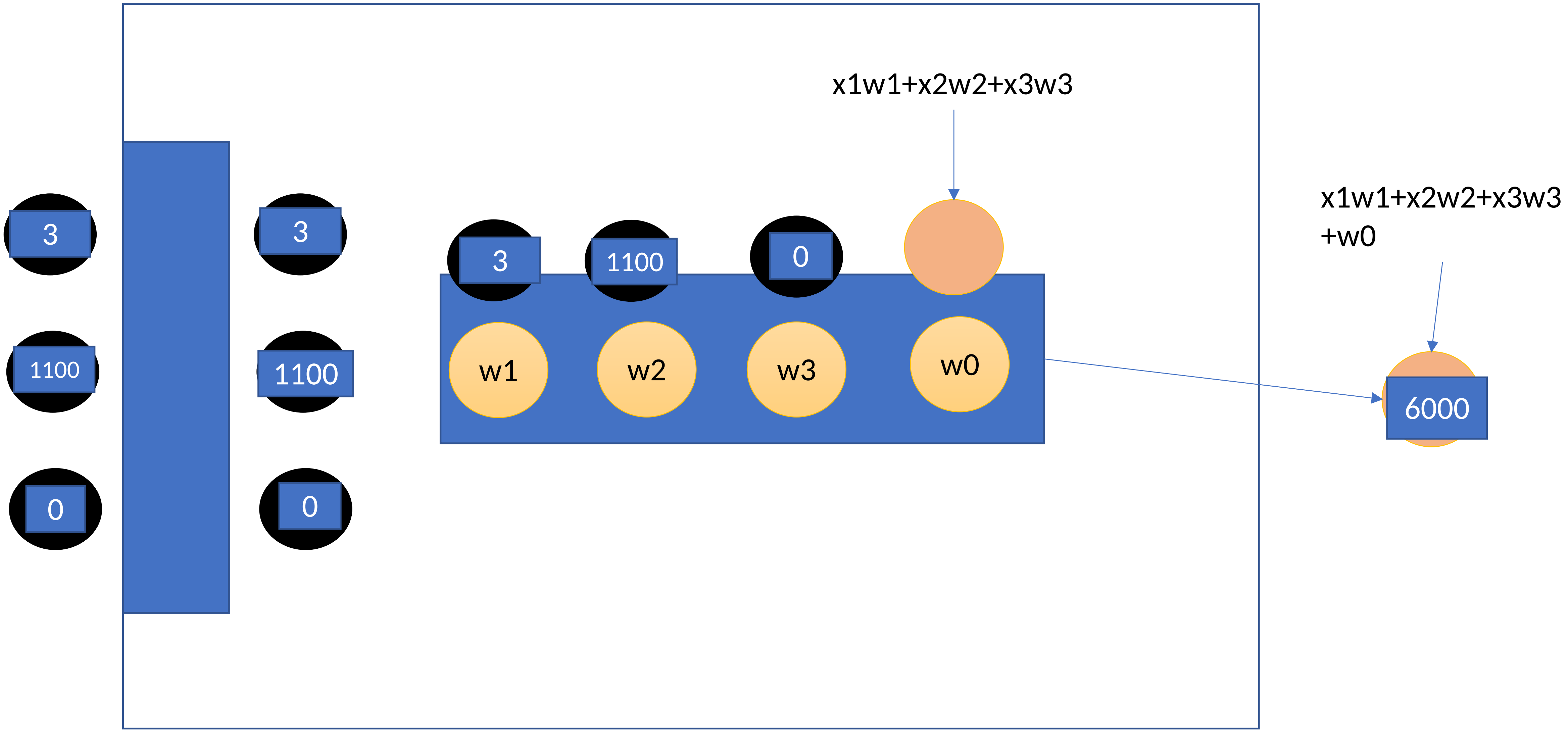
```
Gradient da/dx: [9.60268e-23]
```

Linear Regression Again

- Previously we could perform linear regression with up to 2 features.
- But now we are going to use 3 or more.

SN	No of Rooms	Area	In DOHS?	Rent
1	3	1100	0(No)	6,000
2	5	1300	0	8,000
3	2	1200	1	7,500
4	4	2200	1	20,000
…	…	…	…	…

Visual Representation of Linear Regression



Ok so how do we find the w's?

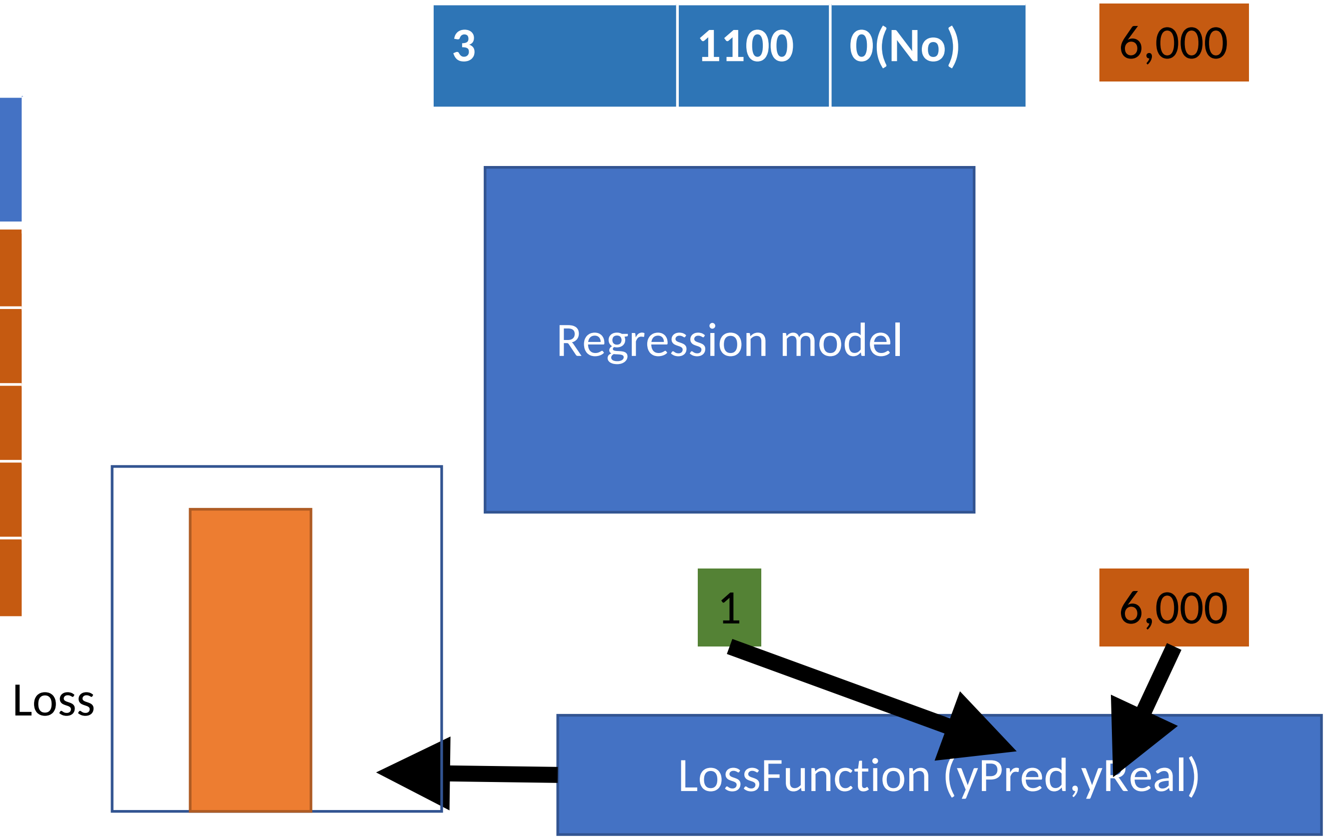
- Yes! By using differentiation.
- But differentiate w.r.t. what?

The Loss Function

The Loss Function

- It is the measure of how much the predicted output varies from the actual output

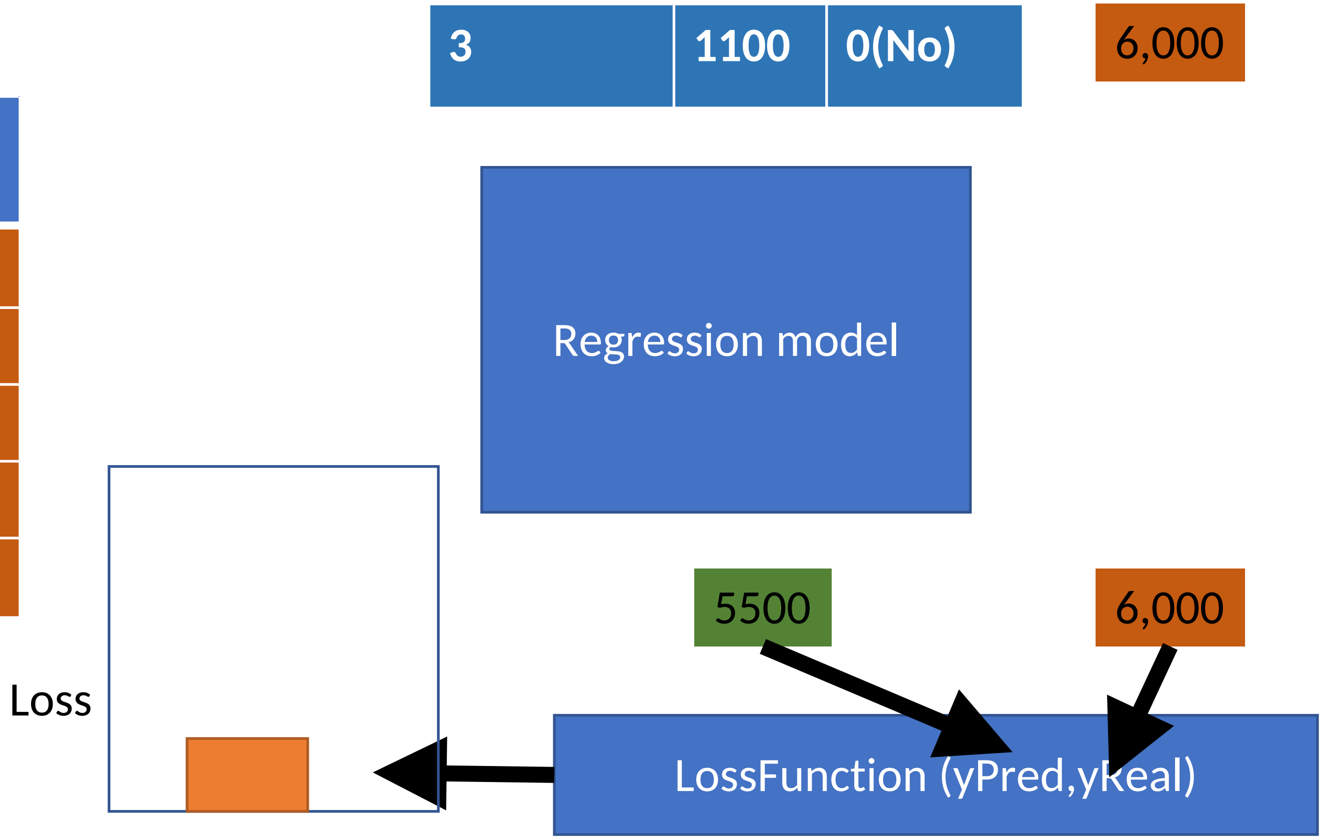
SN	No of Rooms	Area	In DOHS?	Rent
1	3	1100	0(No)	6,000
2	5	1300	0	8,000
3	2	1200	1	7,500
4	4	2200	1	20,000
…	…	…	…	…



The Loss Function

- It is the measure of how much the predicted output varies from the actual output

SN	No of Rooms	Area	In DOHS?	Rent
1	3	1100	0(No)	6,000
2	5	1300	0	8,000
3	2	1200	1	7,500
4	4	2200	1	20,000
…	…	…	…	…



The Algorithm – for linear regression

Get Dataset:

X and Y

Initialize Parameters:

- Set initial weights
- Define learning rate.
- Define number of epochs.

For i = 1 to epochs:

1. **Prepare a gradient tape**
2. **Make predictions:**
   - Calculate Ypred=X·W+b
   - Compute the loss L(Y,Ypred):
3. **Compute gradients**
   - Calculate dL/dw and dL/db
4. **Update the weights and bias:**
   - W = W - lr*dL/dW
   - b = b - lr*dL/db

Hill Climbing

```
function HILL-CLIMBING(problem) returns a state that is a local maximum

  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
      neighbor ← a highest-valued successor of current
      if neighbor.VALUE ≤ current.VALUE then return current.STATE
      current ← neighbor
```

- Let’s solve a problem using Hill Climbing

A better version of hill climbing – Simulated Annealing

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for t = 1 **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if T = 0 **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

Simulated Annealing

```
function SIMULATED_ANNEALING ()

  current ← MAKE-NODE(problem.INITIAL-STATE)
  INITIAL_TEMPERATURE ← 100
  MAX_ITERATIONS ← 1000

  for t = 1 to MAX_ITERATIONS do

    T ← INITIAL_TEMPERATURE / (1 + t)

    if T = 0 then
      return current

    next ← RANDOM_SUCCESSOR(current)

    ΔE ← next.Value - current.Value  # Calculate change in value (ΔE)

    if ΔE > 0 then # if the new state is better, simply take it
      current ← next
    else
      probability ← e^(ΔE / T)  # Calculate acceptance probability for worse state
      if RANDOM(0, 1) < probability then
        current ← next

  return current
```

Thank You