Gradient Descent = not efficient !

Sometimes learning rate too small ⇒ slow
Sometimes learning rate too large ⇒ diverges and doesn't reach minimum

There is an algorithm called the ADAM algorithm

↓

Adaptive Movement Estimation

Adjusts 'α' (the learning rate) accordingly

how?

doesn't use 1 α

Uses different α for each parameter ($\vec{w}$ and b)

For eg. if our algorithm has 10 w's.

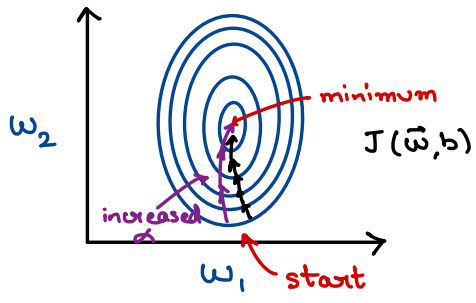
$$w_1 = w_1 - \alpha_1 \frac{\partial J(\vec{w}, b)}{\partial w_1}$$

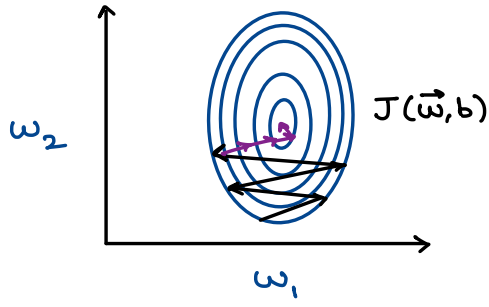
.
.

$$w_{10} = w_{10} - \alpha_{10} \frac{\partial J(\vec{w}, b)}{\partial w_{10}}$$

$$b = b - \alpha_{11} \frac{\partial J(\vec{w}, b)}{\partial b}$$

Intuition of ADAM algorithm





If w_j (or b_j) moves in the same direction then increase α_j.

If w_j (or b_j) keeps oscillating, then reduce α_j

IN CODE

model

```
model = Sequential ([
        Dense (units=25, activation="relu"),
        Dense (units=15, activation="relu"),
        Dense (units=1, activation="linear")
        ])
```

compile

new line

```
model.compile (optimizers = tf.keras.optimizers.Adam (learning_
        rate = 1e-3),
loss = tf.keras.losses SparseCategoricalCrossentropy (from_
        logits=True))
```

$\alpha = 10^{-3} = 0.001$ (just needs a default value)

fit

```
model.fit(x, y, epochs = 100)
```