



deeplearning.ai

# Programming Frameworks

---

## TensorFlow

# Motivating problem

$$\begin{aligned} J(w) &= \boxed{w^2 - 10w + 25} \\ &\quad \swarrow \\ &\quad (w-5)^2 \\ &\quad w=5 \end{aligned}$$

$$\begin{aligned} J(w, b) \\ \uparrow \quad \uparrow \end{aligned}$$

# Code example

```
import numpy as np
import tensorflow as tf
```

```
coefficients = np.array([[1], [-20], [25]])
```

```
w = tf.Variable([0], dtype=tf.float32)
```

```
x = tf.placeholder(tf.float32, [3, 1])
```

```
cost = x[0][0]*w**2 + x[1][0]*w + x[2][0] # (w-5)**2
```

```
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

```
init = tf.global_variables_initializer()
```

```
session = tf.Session()
```

```
session.run(init)
```

```
print(session.run(w))
```

```
with tf.Session() as session:
```

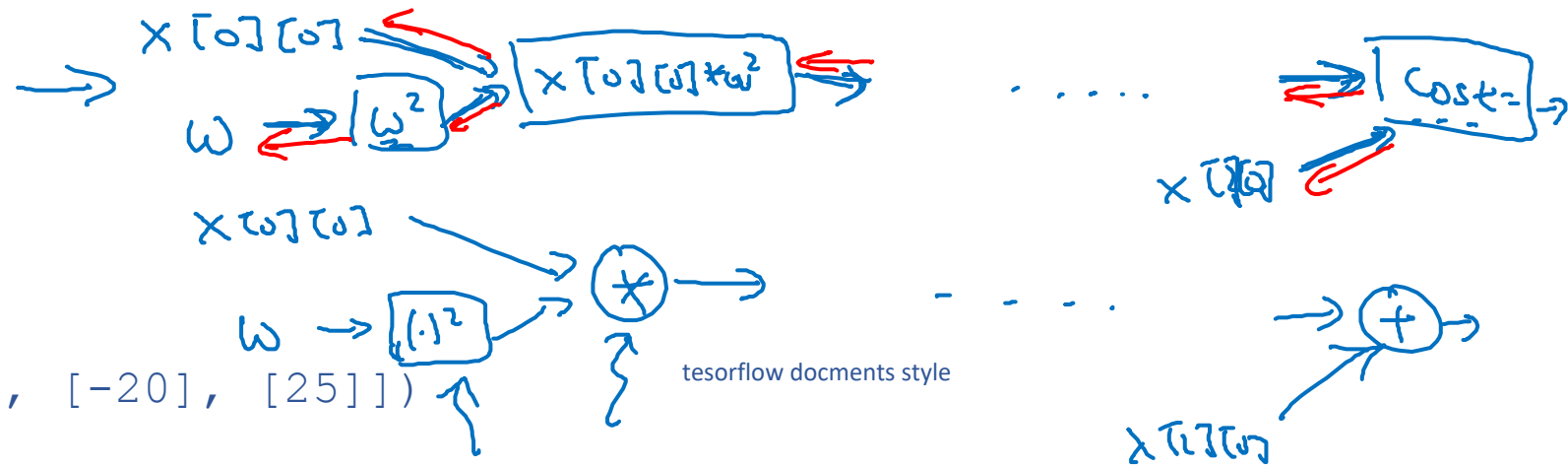
```
    session.run(init)
```

```
    print(session.run(w))
```

```
for i in range(1000):
```

```
    session.run(train, feed_dict={x:coefficients})
```

```
print(session.run(w))
```



And programming frameworks like TensorFlow have already built-in the necessary backward functions, which is why by using the built-in functions to compute the forward function, it can automatically do the backward functions as well to implement back propagation through even very complicated functions and compute derivatives for you.

So that's why you don't need to explicitly implement back prop.

And this is one of the things that makes the programming frameworks help you become really efficient.