Multivariable linear regression

$$H(x) = Wx + b$$

$$H(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} \left(H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)} \right)^2$$

⇩

$$H(x_1, \cdots x_n) = w_1 x_1 + w_2 x_2 + \cdots w_n x_n + b$$

⇩

"Matrix"로 표현하자.

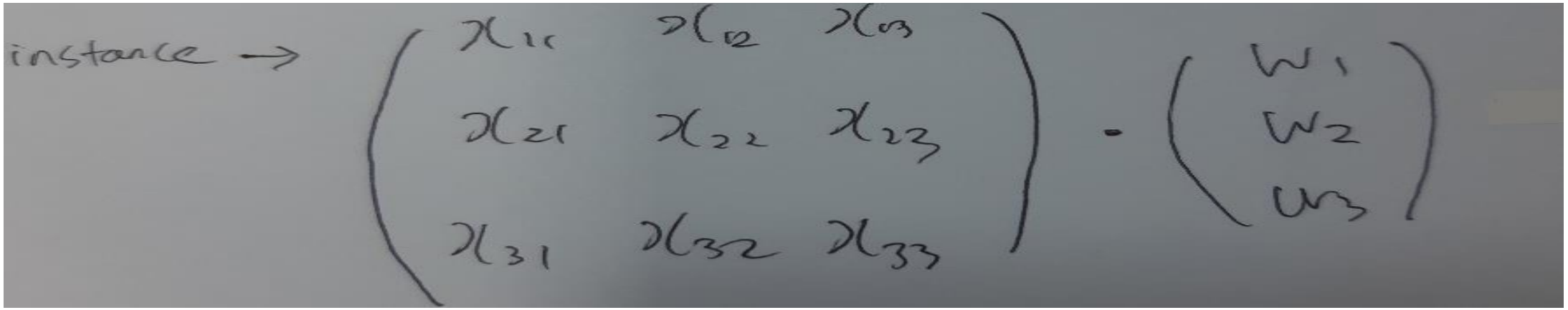
$$(x_1 \; x_2 \; x_3 \; \cdots x_n) \cdot \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \rightsquigarrow H(X) = XW$$

instance1

variable1

목표값

x1	x2	x3	Y
73	80	75	152
93	88	93	185
89	91	90	180



Multi-variable linear regression TF

```
x_data = [[73., 80., 75.],
          [93., 88., 93.],
          [89., 91., 90.],
          [96., 98., 100.],
          [73., 66., 70.]]
y_data = [[152.],
          [185.],
          [180.],
          [196.],
          [142.]]


# None = n과 같다고 생각하기
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
```

Shape 한번 더 확인

```
# Hypothesis
hypothesis = tf.matmul(X, W) + b

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)
```

```
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in range(2001):
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train], feed_dict={X: x_data, Y: y_data})
    if step % 10 == 0:
        print(step, "Cost: ", cost_val, "\nPrediction:\n", hy_val)
```

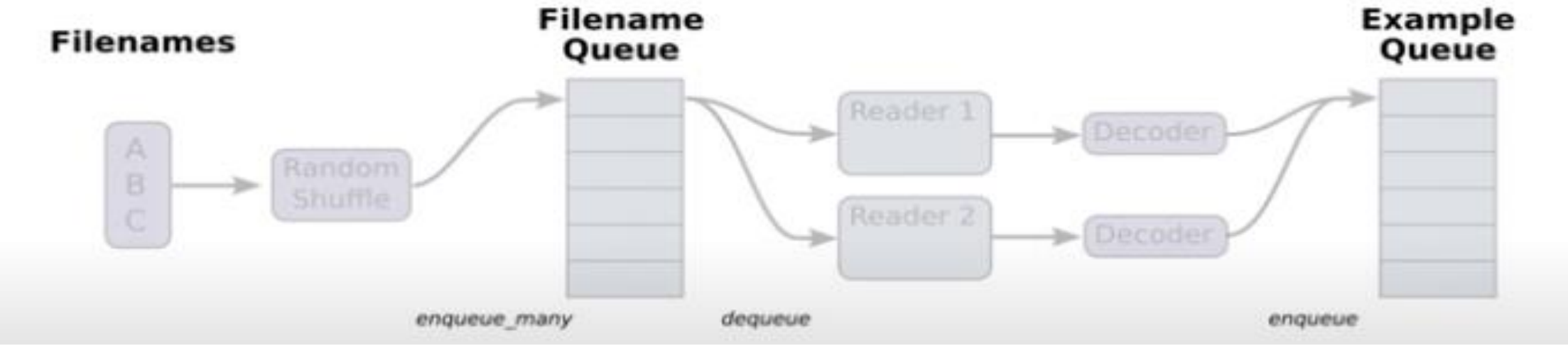
Loading Data from file

```
xy = np.loadtxt('data-01-test-score.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

X_data : 전체 n개의 instance 중에서 마지막 column 제외하고 가져온다.
Y_data : 전체 n개의 instance 중에서 마지막 column만 가져온다.

Queue Runners

- 파일의 크기가 클 때 메모리가 부족한 경우 발생
- 여러 개의 파일을 큐에 쌓고 Reader로 연결 후 Decoder을 거쳐 Queue에 쌓음



```
filename_queue = tf.train.string_input_producer(
    ['data-01-test-score.csv'], shuffle=False, name='filename_queue')
```

파일 불러오기

```
reader = tf.TextLineReader()
key, value = reader.read(filename_queue)
```

파일 읽기

```
# Default values, in case of empty columns. Also specifies the type of the
# decoded result.
record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)
```

읽어온 value값 parsing 및 type 지정

```
# collect batches of csv in
train_x_batch, train_y_batch =
    tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)
```

X Y

값들을 batch만큼 데이터 읽어 들임

```
for step in range(2001):
    x_batch, y_batch = sess.run([train_x_batch, train_y_batch])
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train], feed_dict={X: x_batch, Y: y_batch})
    if step % 10 == 0:
        print(step, "Cost: ", cost_val, "₩nPrediction:₩n", hy_val)

coord.request_stop()
coord.join(threads)
```

배치의 순서 무작위로 하고 싶다 : shuffle_batch 사용