# EE4400 Tut 1 Q1

$f(x, w) = x^T w$

$f = [1 \;\; x_1]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = w_0 + w_1 x_1$

? ?

$\frac{\partial f}{\partial w} = x$

$\underline{x}$   Target $\underline{y}$

(1980 / 2018)   \$ -- \$

1   0.2   0.3

1   ⋮   ⋮

1

$E = \frac{1}{2}(y - f)^2$

$$\frac{\partial E}{\partial \underline{w}} = \frac{\partial E}{\partial f} \cdot \frac{\partial f}{\partial \underline{w}}$$

$$= (y-f)(-1)x$$

$$= (f-y)x$$

$$\Delta w = -\frac{\partial E}{\partial w} = \underline{(y-f)x}$$

eta

note that x is a vector

**EE4400 - Tutorial 1, Question 1**

In [15]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [16]:

```python
def exp_cost_gradient(X, w, y):
    # Compute prediction, cost and gradient based on mean square error loss
    pred_y = X @ w
    cost   = np.sum((y - pred_y)*(y - pred_y))
    gradient = -(y - pred_y) @ X
#     print(gradient)
#     print(cost)
    return pred_y, cost, gradient
```

In [17]:

```python
# Load data
df = pd.read_csv('government-expenditure-on-education.csv')
expenditure = df['total_expenditure_on_education'].to_numpy()
years = df['year'].to_numpy()

# create normalized variables
max_expenditure = max(expenditure)
max_year = max(years)
min_year = min(years)

y = expenditure/max_expenditure
X = np.ones([len(y), 2])
X[:, 1] = (years-min_year)/(max_year-min_year)
#X[:, 1] = np.arange(0,1,0.1)

# Gradient descent
learning_rate = 0.01
w = np.array([0,0])
pred_y, cost, gradient = exp_cost_gradient(X, w, y)
num_iters = 200;
cost_vec = np.zeros(num_iters)
print('Initial Cost =', cost)
for i in range(0, num_iters):

    # update w
    w = w - learning_rate*gradient

    # compute updated cost and new gradient
    pred_y, cost, gradient = exp_cost_gradient(X, w, y)
    cost_vec[i] = cost

    if(i % 20 == 0):
        print('Iter', i, ': cost =', cost)

pred_y, cost, gradient = exp_cost_gradient(X, w, y)
print('Final Cost =', cost)
```

```
Initial Cost = 10.837516320395803
Iter 0 : cost = 4.3543130851728815
Iter 20 : cost = 0.7976577410102698
Iter 40 : cost = 0.37869717216854304
Iter 60 : cost = 0.234492466616192
Iter 80 : cost = 0.18485773091454086
Iter 100 : cost = 0.16777363522753555
Iter 120 : cost = 0.16189335156495857
Iter 140 : cost = 0.1598693792935628
Iter 160 : cost = 0.15917273539159435
Iter 180 : cost = 0.15893295309170388
Final Cost = 0.1588527935553728
```

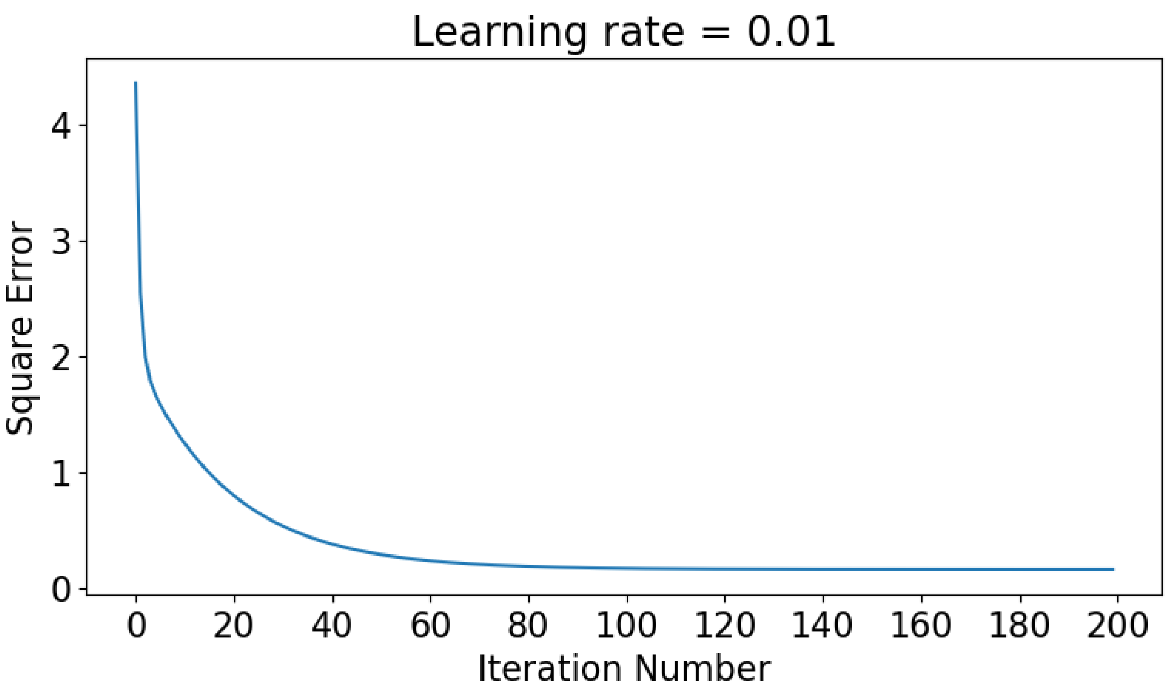
In [18]:

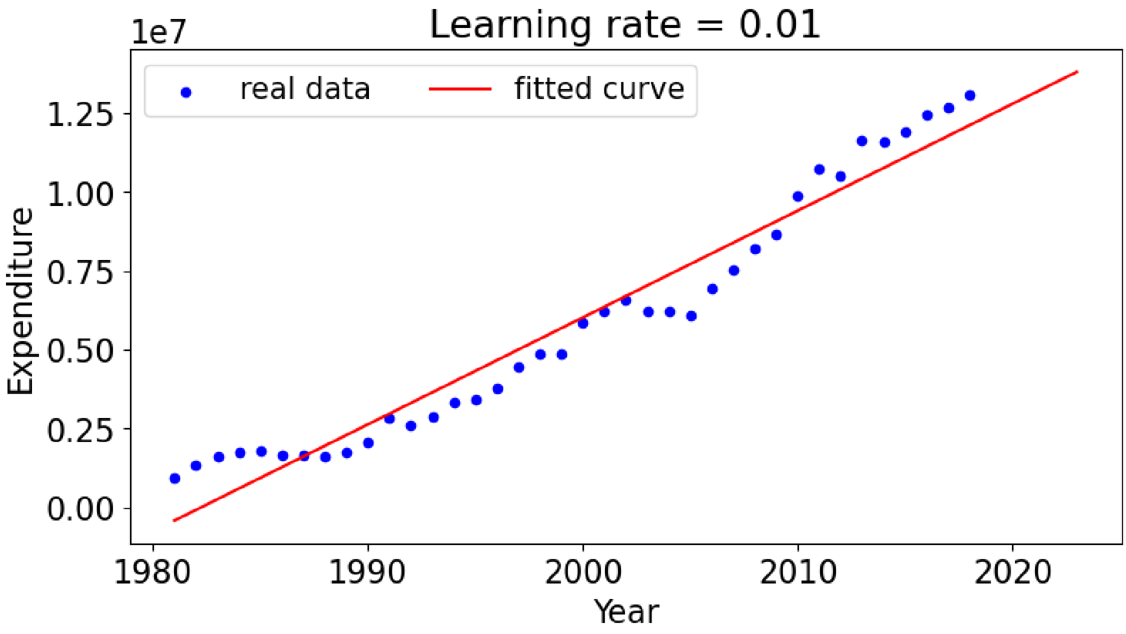
```
# Plot cost function values over iterations
plt.figure(0, figsize=[9,4.5])
plt.rcParams.update({'font.size': 16})
plt.plot(np.arange(0, num_iters, 1), cost_vec)
plt.xlabel('Iteration Number')
plt.ylabel('Square Error')
plt.xticks(np.arange(0, num_iters+1, 20))
plt.title('Learning rate = ' + str(learning_rate))
#plt.savefig('Figures/FigTut1Cost' + str(learning_rate) + '.eps')

# Extrapolate until year 2023
ext_years = np.arange(min_year,2024,1)
ext_X = np.ones([len(ext_years), 2])
ext_X[:, 1] = (ext_years-min_year)/(max_year-min_year)
pred_y = ext_X @ w # model-dependent

# Plot extrapolation
plt.figure(1, figsize=[9,4.5])
plt.rcParams.update({'font.size': 16})
plt.scatter(years, expenditure, s=20, marker='o', c='blue', label='real data')
plt.plot(ext_years, pred_y * max_expenditure, c='red', label='fitted curve')
plt.xlabel('Year')
plt.ylabel('Expenditure')
plt.title('Learning rate = ' + str(learning_rate))
plt.legend(loc='upper left',ncol=3, fontsize=15)
```

Out[18]: <matplotlib.legend.Legend at 0x1aa72f39130>




Learning rate = 0.01
1e7
real data
fitted curve
1.25
1.00
0.75
0.50
0.25
0.00
Expenditure
1980
1990
2000
2010
2020
Year

**NUS Engineering** EE4400

Tutorial 1 Q2: **Back Propagation**

(CK Tham, ECE NUS)

I. Formulas

1. General

$$g(x) = \frac{1}{1+e^{-x}}$$
$$g'(x) = g(x) * (1 - g(x))$$

2. Backward Pass

$$E = \frac{1}{2}\sum_k (d_k - y_k)^2$$
$$\frac{\partial E}{\partial y_k} = (y_k - d_k)$$

(a) Output Layer - node $k$

$$E = \frac{1}{2}\sum_k (d_k - y_k)^2 \text{ across all output nodes (for 1 training pattern)}$$
$$y_k = g(net_k) \text{ where } net_k = \sum_j w_{jk} y_j$$
$$\frac{\partial E}{\partial y_k} = (y_k - d_k)$$
$$\delta_k = \frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial net_k} = (y_k - d_k) \cdot g'(net_k)$$
$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{jk}} = \delta_k \cdot y_j$$
$$\Delta\, w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

(b) Hidden Layer - node $j$ (summation of $\delta_k$ over output nodes $k$)

$$\frac{\partial E}{\partial y_j} = \sum_k \left( \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial y_j} \right) = \sum_k (\delta_k \cdot w_{jk})$$
$$\delta_j = \frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} = \frac{\partial E}{\partial y_j} \cdot g'(net_j)$$
$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} = \delta_j \cdot y_i$$
$$\Delta\, w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

II. Application to Specific Scenario

1. Forward Pass

$$
\begin{aligned}
net_{h1} &= w_1 * i_1 + w_2 * i_2 + b_1 = 0.360 \\
y_{h1} &= g(net_{h1}) = 0.589 \\
net_{h2} &= w_3 * i_1 + w_4 * i_2 + b_2 = 0.881 \\
y_{h2} &= g(net_{h2}) = 0.707 \\
net_{o1} &= w_5 * y_{h1} + w_6 * y_{h2} + b_3 = 0.447 \\
y_{o1} &= g(net_{o1}) = 0.610 \\
net_{o2} &= w_7 * y_{h1} + w_8 * y_{h2} + b_4 = 0.408 \\
y_{o2} &= g(net_{o2}) = 0.601
\end{aligned}
$$

2. Backward Pass

(a) Error

$$
E_{total} = \frac{1}{2}\sum_k (d_k - y_k)^2 = 0.209
$$
$$
\text{where } d_k = target_{ok} \text{ and } y_k = y_{ok}
$$

(b) Output Layer

(i) Weight $w_5$ between node h1 and node o1

$$
\begin{aligned}
\frac{\partial E}{\partial y_{o1}} &= (y_{o1} - d_{o1}) = -0.340 \\
\delta_{o1} &= \frac{\partial E}{\partial net_{o1}} = \frac{\partial E}{\partial y_{o1}} \cdot \frac{\partial y_{o1}}{\partial net_{o1}} = (y_{o1} - d_{o1}) \cdot g'(net_{o1}) = -0.081 \\
\frac{\partial E}{\partial w_5} &= \frac{\partial E}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial w_5} = \delta_{o1} \cdot y_{h_1} = -0.048 \\
\Delta\, w_5 &= -\eta \frac{\partial E}{\partial w_5} = 0.005
\end{aligned}
$$

(ii) Weight $w_7$ between node h1 and node o2

$$
\begin{aligned}
\frac{\partial E}{\partial y_{o2}} &= (y_{o2} - d_{o2}) = 0.551 \\
\delta_{o2} &= \frac{\partial E}{\partial net_{o2}} = \frac{\partial E}{\partial y_{o2}} \cdot \frac{\partial y_{o2}}{\partial net_{o2}} = (y_{o2} - d_{o2}) \cdot g'(net_{o2}) = 0.132 \\
\frac{\partial E}{\partial w_7} &= \frac{\partial E}{\partial net_{o2}} \cdot \frac{\partial net_{o2}}{\partial w_7} = \delta_{o2} \cdot y_{h_1} = 0.078 \\
\Delta\, w_7 &= -\eta \frac{\partial E}{\partial w_7} = -0.008
\end{aligned}
$$

(c) Hidden Layer
For node h1, summation of $\delta_k$ over output nodes $k$, i.e. o1 and o2.
(i) Weight $w_1$ between input i1 and node h1

$$\frac{\partial E}{\partial y_{h1}} = \sum_k \left( \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial y_{h1}} \right) = \sum_k (\delta_k \cdot w_{jk}) = 0.019$$
$$\delta_{h1} = \frac{\partial E}{\partial net_{h1}} = \frac{\partial E}{\partial y_{h1}} \cdot \frac{\partial y_{h1}}{\partial net_{h1}} = \frac{\partial E}{\partial y_{h1}} \cdot g'(net_{h1}) = 0.005$$
$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial net_{h1}} \cdot \frac{\partial net_{h1}}{\partial w_1} = \delta_{h1} \cdot y_{i1} = 4.703 \times 10^{-4}$$
$$\Delta\ w_1 = -\eta \frac{\partial E}{\partial w_1} = -4.703 \times 10^{-5}$$