EE2211 Introduction to Machine Learning

T14 & T22, Chua Dingjuan elechuad@nus.edu.sg
Materials @ tiny.cc/ee2211tut

Tutorial 8

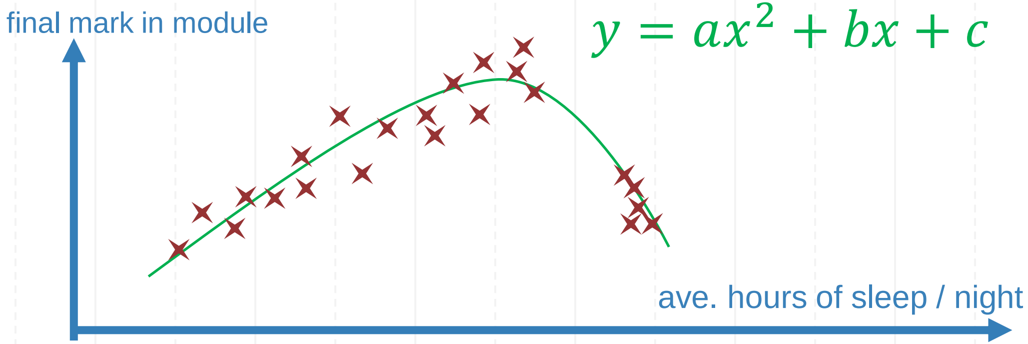
Gradient Descent

Discussion of Solutions 8

Q1,2,3,4

Polynomial Regression… Always?

	1 … d	*y*	
	ave. hrs of sleep / night	final mark in module	
x_1	5	23	y_1
x_2	4	45	y_2
x_3	3	89	y_3
x_4	8	46	y_4
x_5	9	90	y_5
…	8.5	80	…
x_m	13	30	y_m



$$y = ax^2 + bx + c$$

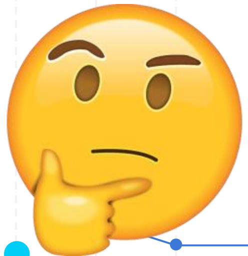
What if it's not a linear nor polynomial model?

Sinusoidal

exp

ln / lg / logn

relu

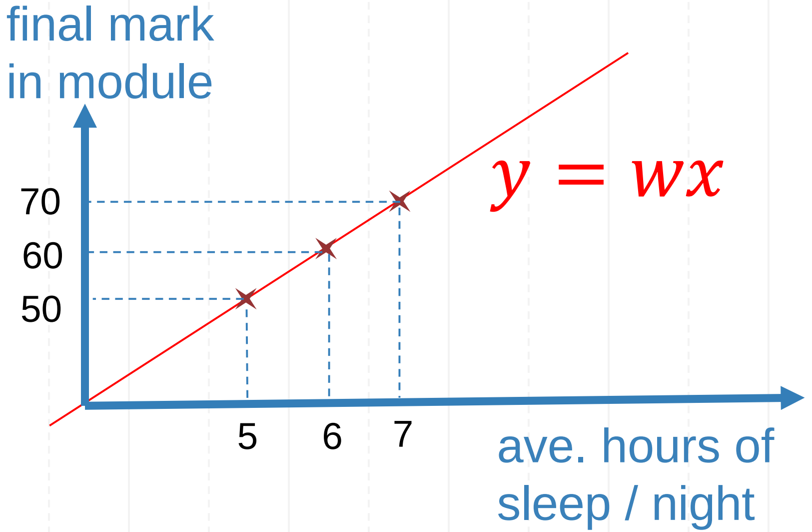


$$\boldsymbol{y} = \boldsymbol{P}\boldsymbol{w}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} \Rightarrow \begin{bmatrix} 23 \\ 45 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 4 & 16 \\ 1 & \dots & \dots \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix}$$

Linear & Polynomial Regression

	1 … d	y	
	ave. hrs of sleep / night	final mark in module	
x_1	5	50	y_1
x_2	6	60	y_2
x_3	7	70	y_3



$$\boldsymbol{f_w(x) = y = Pw}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} [w]$$

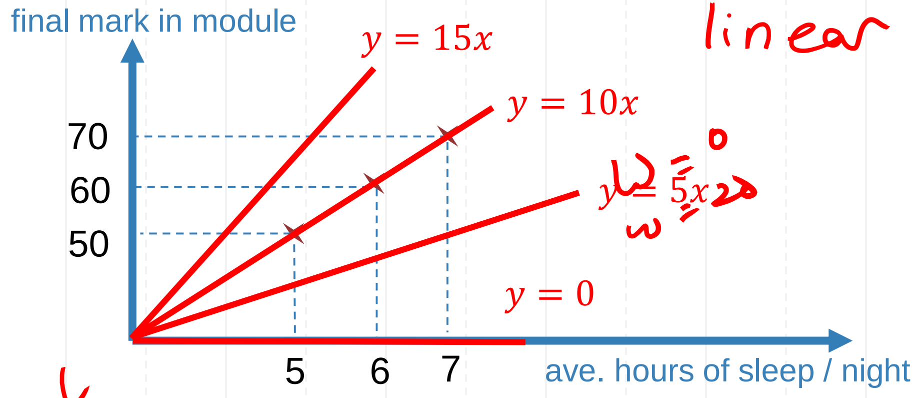
$$\begin{bmatrix} 50 \\ 60 \\ 70 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix} [10]$$

$$\boldsymbol{w = P^{-1}y}$$

Gradient Descent

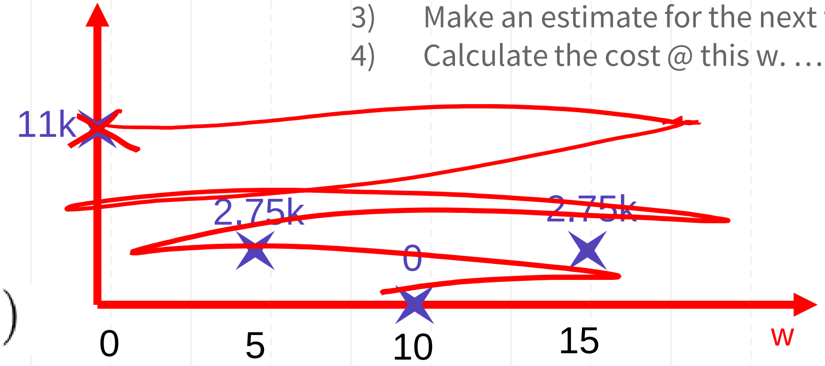
	1 … d	y	
	ave. hrs of sleep / night	final mark in module	
x_1	5	50	y_1
x_2	6	60	y_2
x_3	7	70	y_3

$y = wx$ ➔ $w = ?$



Cost Function, C(w)
$= \sum (f(x_i,w)-y_i)^2$

1) Make an initial guess for w, assume w=0.
2) Calculate the cost @ this w.
3) Make an estimate for the next w, eg. w=5.
4) Calculate the cost @ this w. … repeat…



- Minimum cost occurs at w = 10 ➔ ☺
- Learning Rate η = ?
- Making use of the gradient of cost function
  ➔ MINIMUM cost!

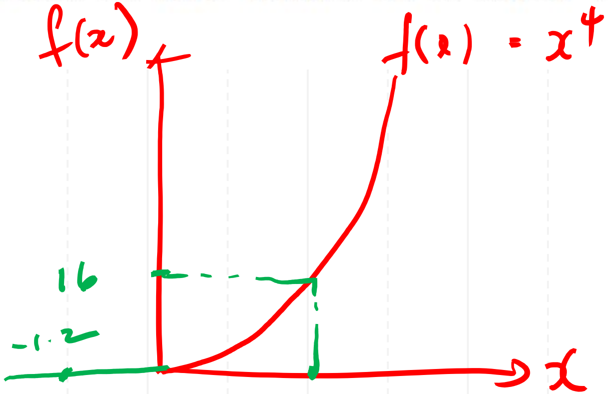
$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \eta \nabla_{\mathbf{w}} C(\mathbf{w}_k)$$

- Pros and Cons????

time ↑ local minima

Question1

Suppose we are minimizing $f(x) = x^4$ with respect to x. We initialize x to be 2. We perform gradient descent with learning rate 0.1. What is the value of x after the first iteration?



SOLUTION

- $f(x) = x^4$, gradient $f'(x)=4x^3$
- Initialize x=2 ➔ $f(2) = 2^4 = 16$, $f'(2) = 4*(2^3) = 32$
- Using learning rate of 0.1, after first iteration of gradient descent : x = 2

$$x = \overset{2}{\cancel{f(2)}} - \lambda^* f'(2) = 2 - 0.1(32) = -1.2$$

$f(x) = x^4$

$f'(x) = 4x^3 = +ve$

Question2

Please consider the csv file (government-expenditure-on-education.csv), which depicts the government's educational expenditure over the years. We would like to predict expenditure as a function of year. To do this, fit an exponential model $f(\mathbf{x}, \mathbf{w}) = \exp(-\mathbf{x}^T\mathbf{w})$ with squared error loss to estimate $\mathbf{w}$ based on the csv file and gradient descent. In other words, $C(\mathbf{w}) = \sum_{i=1}^{m}(f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$.

Note that even though year is one dimensional, we should add the bias term, so $\mathbf{x} = [1 \ \text{year}]^T$. Furthermore, optimizing the exponential function is tricky (because a small change in $\mathbf{w}$ can lead to large change in f). Therefore for the purpose of optimization, divide the "year" variable by the largest year (2018) and divide the "expenditure" by the largest expenditure, so that the resulting normalized year and normalized expenditure variables are between 0 and 1. Use a learning rate of 0.03 and run gradient descent for 2000000 iterations.

norm.

(a) Plot the cost function C(w) as a function of the number of iterations.

(b) Use the fitted parameters to plot the predicted educational expenditure from year 1981 to year 2023.

(c) Repeat (a) using a learning rate of 0.1 and learning rate of 0.001. What do you observe relative to (a)?

differentiating the sum wrt w
⇒ gradient $\frac{d}{dw}$

sum of squared errors between actual y (y_i) and predicted y ($f(x_i, w)$)

$$\nabla_{\mathbf{w}} \underbrace{C(\mathbf{w})}_{\text{cost}} = \nabla_{\mathbf{w}} \sum_{i=1}^{m} (\underbrace{f(\mathbf{x}_i, \mathbf{w})}_{\text{learning model}} - y_i)^2$$

$$\frac{d}{dx}[g(x)]^n = n[g(x)]^{n-1} g'(x)$$

$$= \sum_{i=1}^{m} \nabla_{\mathbf{w}} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

$$= \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \nabla_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}) \quad \text{chain rule}$$

$$f(x_i, w) = e^{-x_i^T w}$$

$$= \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \nabla_{\mathbf{w}} \exp(-\mathbf{x}_i^T \mathbf{w})$$

$$\frac{d}{dw} e^{-x_i^T w}$$

$$= - \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \exp(-\mathbf{x}_i^T \mathbf{w}) \nabla_{\mathbf{w}} (\mathbf{x}_i^T \mathbf{w})$$

$$= \underbrace{\frac{d}{dw}(-x_i^T w)}_{\text{given in tutorial}} . e^{-x_i^T w}$$

$$= -x_i$$

$$= - \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) \exp(-\mathbf{x}_i^T \mathbf{w}) \mathbf{x}_i$$

$$= - \sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) f(\mathbf{x}_i, \mathbf{w}) \mathbf{x}_i$$

Please consider the csv file (government-expenditure-on-education.csv), which depicts the government's educational expenditure over the years. We would like to predict expenditure as a function of year. To do this, fit an exponential model $f(\mathbf{x}, \mathbf{w}) = \exp(-\mathbf{x}^T\mathbf{w})$ with squared error loss to estimate $\mathbf{w}$ based on the csv file and gradient descent. In other words, $C(\mathbf{w}) = \sum_{i=1}^{m}(f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$.

Note that even though year is one dimensional, we should add the bias term, so $\mathbf{x} = [1 \ \text{year}]^T$. Furthermore, optimizing the exponential function is tricky (because a small change in $\mathbf{w}$ can lead to large change in f). Therefore for the purpose of optimization, divide the "year" variable by the largest year (2018) and divide the "expenditure" by the largest expenditure, so that the resulting normalized year and normalized expenditure variables are between 0 and 1. Use a learning rate of 0.03 and run gradient descent for 2000000 iterations.

Question2

(a) Plot the cost function C(w) as a function of the number of iterations.

$$\nabla_{\mathbf{w}} C(\mathbf{w})$$

$$= -\sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) f(\mathbf{x}_i, \mathbf{w}) \mathbf{x}_i$$

1980 - 2022

STEPS : :

- Import file using pandas and extract year and expenditure.
- Normalize. Year (norm) → x training data, Expenditure (norm) → y training data.
- Create X matrix with bias

Gradient Descent

- Set w to initial value.
  - Using gradient descent method and specific learning rate, find next w.
  - For each w, find cost.
- Repeat iteratively for 2e6 iterations.

Question2 (a)

(a) Plot the cost function C(w) as a function of the number of iterations.

(a) SOLUTION

```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)
y = expenditure/max_expenditure
X = np.ones([len(y), 2])
X[:, 1] = years/max_year

# Gradient descent
learning_rate = 0.03
w = np.zeros(2)
pred_y, cost, gradient = exp_cost_gradient(X, w, y)
num_iters = 2000000;
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 % 200000 == 0):
        print('Iter', i, ': cost =', cost)

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

Gradient descent formula

$$= -\sum_{i=1}^{m} 2(f(\mathbf{x}_i, \mathbf{w}) - y_i) f(\mathbf{x}_i, \mathbf{w}) \mathbf{x}_i$$

```python
def exp_cost_gradient(X, w, y):

    # Compute prediction, cost and gradient based on mean square error loss
    pred_y = np.exp(-X @ w)
    cost   = np.sum((pred_y - y)*(pred_y - y))
    gradient = -2 * (pred_y - y) * pred_y @ X

    return pred_y, cost, gradient
```

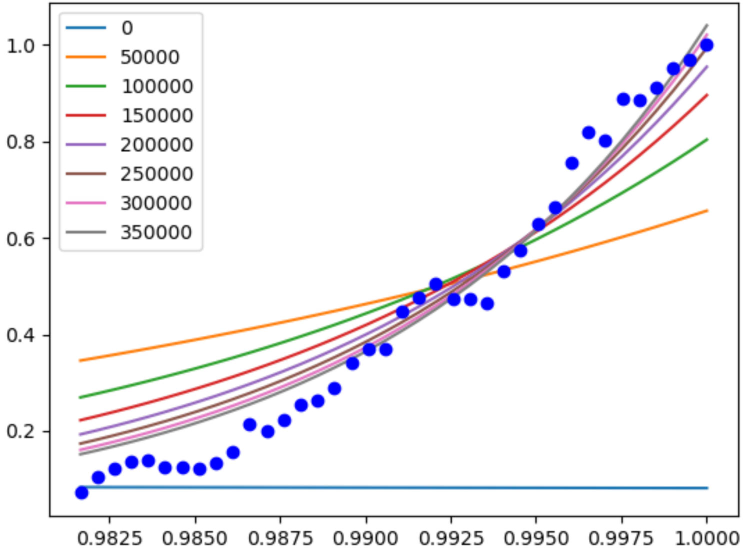
```python
# Importing data from .csv using pandas read_csv function
df = pd.read_csv("government-expenditure-on-education.csv")
expenditure = df['total_expenditure_on_education'].to_numpy()
years = df['year'].to_numpy()

# Nnormalization of x and y variables
x = years/max(years)
y = expenditure/max(expenditure)
X = np.column_stack((np.ones((len(x))),x))

# Implementation of Gradient Descent
num_iter = 2000000

# Initialization
w = np.zeros(2)
learning_rate = 0.03

#calculates the pred y value current w value across all X points
pred_y = np.exp(-X @ w)

#creating a zeroes matrix to store the cost values
cost = np.zeros(num_iter)

# Running gradient descent for 2000000 iterations
for i in range(0, num_iter):

    #gradient at this current w
    gradient = -2 * (pred_y - y) * pred_y @ X

    #** Finding NEW w **#
    w = w - learning_rate * gradient

    pred_y = np.exp(-X @ w)

    current_cost = np.sum((pred_y - y)*(pred_y - y))
    cost[i] = current_cost
```

Question2 (a)

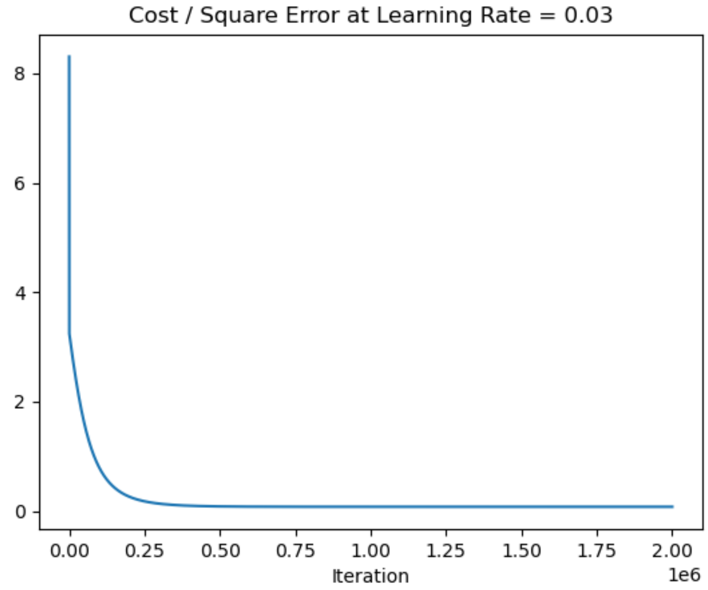
(a) Using learning rate of 0.03, run gradient descent for 2e6 iterations. Plot the cost function C(w) as a function of the number of iterations.

(a) SOLUTION

Notice how the exponential curve slowly approaches our training data



Notice how cost approaches zero

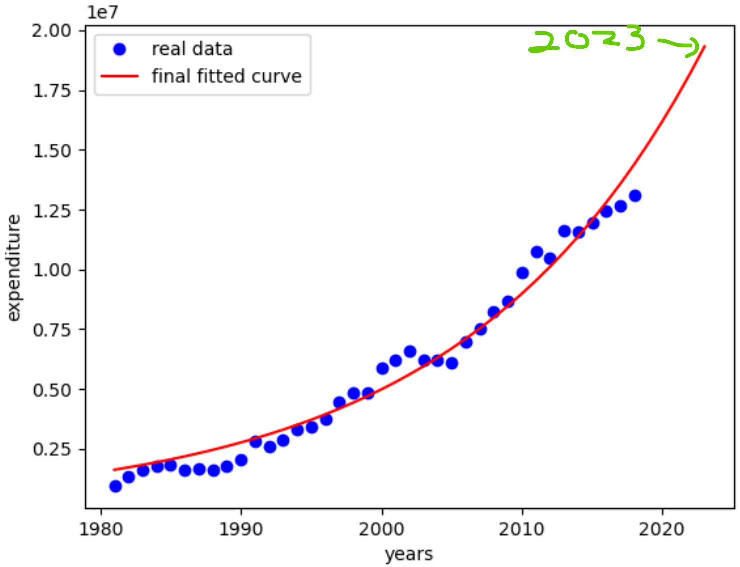


Question2 (b)

(b) Use the fitted parameters to plot the predicted educational expenditure from year 1981 to year 2023.

(b) SOLUTION

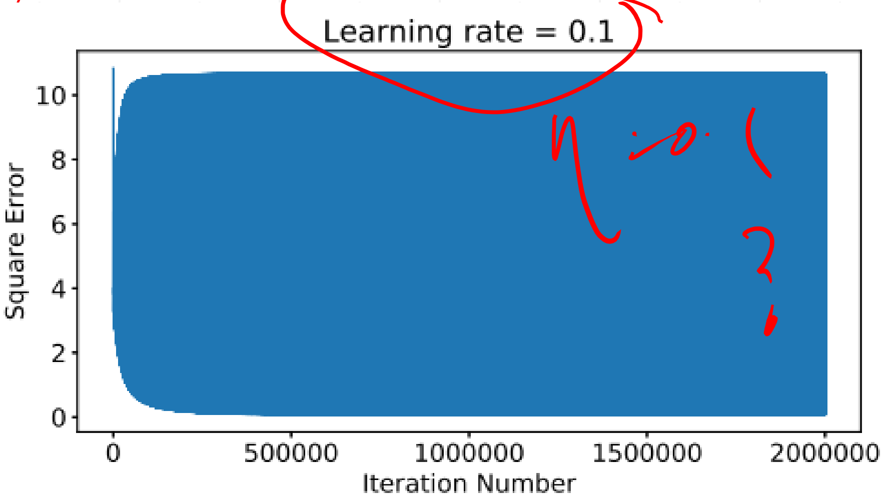
```
#1b - Extrapolate to 2024
#Generating a new list of years, from 1981 to 2024
year_new = np.arange(min(years), 2024)

# Normalizing and creating X matrix to find the predicted y values
x_new = year_new / max(years)
X_new = np.column_stack((np.ones((len(x_new))),x_new))
pred_y_new = np.exp(-X_new @ w)


#Plotting De-Normalized Training Data & Final exponential curve
plt.plot(years,expenditure,'bo',label='real data')
plt.plot(year_new,pred_y_new*max(expenditure),'r',label='final fitted curve')
plt.xlabel('years')
plt.ylabel('expenditure')
plt.legend()
plt.show()
```

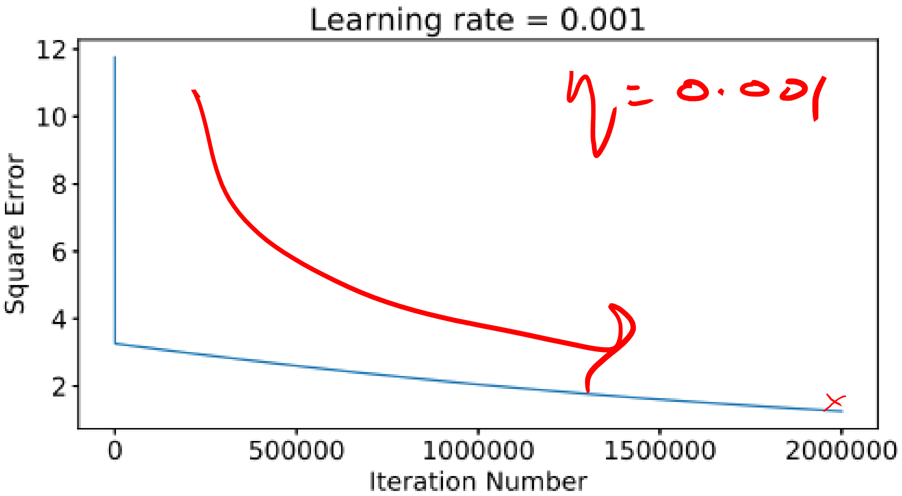


Question2(c)

(c) Repeat (a) using a learning rate of 0.1 and learning rate of 0.001. What do you observe relative to (a)?

(c) SOLUTION





- η=0.1 is too big ➔ cost function fluctuate a lot without convergence (does not decrease monotonically with increasing iterations). Final cost function value is much worse that (a).
- η=0.001 is too small ➔ cost function decreases monotonically with increasing iterations, but has not converged even after 2000000 iterations. Final cost function value is much worse that (a).

Question3

Given the linear learning model $f(\mathbf{x}, \mathbf{w}) = \mathbf{x}^T \mathbf{w}$, where $\mathbf{x} \in \mathbb{R}^d$. Consider the loss function $L(f(\mathbf{x}_i, \mathbf{w}), y_i) = (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4$, where i indexes the i-th training sample. The final cost function is $C(\mathbf{w}) = \sum_{i=1}^{m} L(f(\mathbf{x}_i, \mathbf{w}), y_i)$, where m is the total number of training samples. Derive the gradient of the cost function with respect to $\mathbf{w}$.

SOLUTION

$$\begin{aligned}
\nabla_{\mathbf{w}} C(\mathbf{w}) &= \nabla_{\mathbf{w}} \sum_{i=1}^{m} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4 \\
&= \sum_{i=1}^{m} \nabla_{\mathbf{w}} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4 \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \nabla_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}) \qquad \text{chain rule} \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \nabla_{\mathbf{w}} (\mathbf{x}_i^T \mathbf{w}) \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \mathbf{x}_i
\end{aligned}$$

Question4

let this be a

Repeat Question 3 using $f(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w})$, where $\sigma(a) = \frac{1}{1+\exp(-\beta a)}$

SOLUTION

$f(x_i, w) = \sigma(x_i^T w)$

$\nabla_w f(x_i, w) = \nabla_w \underbrace{\sigma(x_i^T w)}_{y}$

$\frac{\partial y}{\partial w} = \frac{\partial y}{\partial a} \cdot \frac{\partial a}{\partial w}$

$\frac{\partial}{\partial a}\sigma(a)$ $\quad \frac{\partial}{\partial w} x_i^T w = x$

$$\nabla_{\mathbf{w}} C(\mathbf{w}) = \nabla_{\mathbf{w}} \sum_{i=1}^{m} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4$$

$$= \sum_{i=1}^{m} \nabla_{\mathbf{w}} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4$$

$$= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \nabla_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}) \quad \text{chain rule}$$

$$= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \nabla_{\mathbf{w}} \boxed{\sigma(\mathbf{x}_i^T\mathbf{w})}$$

$$= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \frac{\partial \sigma(a)}{\partial a} \nabla_{\mathbf{w}} (\mathbf{x}_i^T\mathbf{w}) \quad \text{chain rule}$$

$$= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \frac{\partial \sigma(a)}{\partial a} \mathbf{x}_i$$

So we just have to evaluate $\frac{\partial \sigma(a)}{\partial a}$ and plug it into the above equation. Note that $\frac{\partial \sigma(a)}{\partial a}$ is evaluated at $a = \mathbf{x}_i^T\mathbf{w}$, so

$$\frac{\partial \sigma(a)}{\partial a} = \frac{\partial}{\partial a}\left(\frac{1}{1+\exp(-\beta a)}\right) \tag{20}$$

$$= -\frac{1}{(1+e^{-\beta a})^2}\frac{\partial(1+e^{-\beta a})}{\partial a} \tag{21}$$

$$= \frac{\beta}{(1+e^{-\beta a})^2}e^{-\beta a} \tag{22}$$

$$= \frac{\beta}{(1+e^{-\beta a})^2}(1+e^{-\beta a} - 1) \tag{23}$$

$$= \beta\left(\frac{1}{1+e^{-\beta a}} - \frac{1}{(1+e^{-\beta a})^2}\right) \tag{24}$$

$$= \beta\left(\sigma(a) - \sigma^2(a)\right) \tag{25}$$

$$= \beta\sigma(a)(1-\sigma(a)) \tag{26}$$

$$= \beta\sigma(\mathbf{x}_i^T\mathbf{w})(1-\sigma(\mathbf{x}_i^T\mathbf{w})) \tag{27}$$

$[1+\exp(-\beta a)]^{-1}$

$-1[\quad]^{-2}$

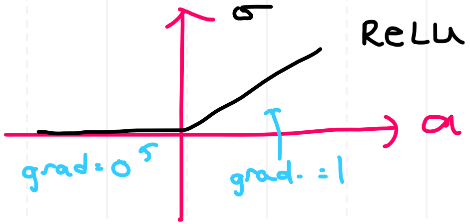
$-\beta e^{-\beta a}$

fore,

$$\nabla_{\mathbf{w}} C(\mathbf{w}) = \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \beta\sigma(\mathbf{x}_i^T\mathbf{w})(1-\sigma(\mathbf{x}_i^T\mathbf{w}))\mathbf{x}_i \tag{28}$$

Question5

Repeat Question 3 using $\underbrace{f(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w})}_{a = x^T w}$, where $\sigma(a) = \max(0, a)$



SOLUTION

$$
\begin{aligned}
\nabla_{\mathbf{w}} C(\mathbf{w}) &= \nabla_{\mathbf{w}} \sum_{i=1}^{m} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4 \\
&= \sum_{i=1}^{m} \nabla_{\mathbf{w}} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^4 \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \nabla_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}) \qquad \text{chai} \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \nabla_{\mathbf{w}} \sigma(\mathbf{x}_i^T\mathbf{w}) \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \frac{\partial \sigma(a)}{\partial a} \nabla_{\mathbf{w}} (\mathbf{x}_i^T\mathbf{w}) \\
&= \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \frac{\partial \sigma(a)}{\partial a} \mathbf{x}_i
\end{aligned}
$$

So we just have to evaluate $\frac{\partial \sigma(a)}{\partial a}$ and plug it into the above equation. Note that $\frac{\partial \sigma(a)}{\partial a}$ is evaluated at $a = \mathbf{x}_i^T\mathbf{w}$. When $a < 0$, $\sigma(a) = 0$, so $\frac{\partial \sigma(a)}{\partial a} = 0$. When $a > 0$, $\sigma(a) = a$, so $\frac{\partial \sigma(a)}{\partial a} = 1$. Let us define $\delta(\mathbf{x}_i^T\mathbf{w} > 0) = \begin{cases} 1 & \text{if } \mathbf{x}_i^T\mathbf{w} > 0 \\ 0 & \text{if } \mathbf{x}_i^T\mathbf{w} < 0 \end{cases}$, so we get

$$
\frac{\partial \sigma(a)}{\partial a} = \delta(\mathbf{x}_i^T\mathbf{w} > 0) \tag{35}
$$

Therefore, we get

$$
\nabla_{\mathbf{w}} C(\mathbf{w}) = \sum_{i=1}^{m} 4(f(\mathbf{x}_i, \mathbf{w}) - y_i)^3 \mathbf{x}_i \delta(\mathbf{x}_i^T\mathbf{w} > 0), \tag{36}
$$