PROBLEM SET 3 Mounica Subramani

a) $f(B) = \|X_1 B - Y_1\|^2$, where $B \in \mathbb{R}^p$

Show that the minimizer of $f(B)$, denoted by $\hat{\beta}_1$, is equal to $(X_1^T X_1)^{-1} X_1^T Y_1$.

$$f(B) = \|X_1 B - Y_1\|^2$$

$$\boxed{Y' = X_1 B}$$

$$\Rightarrow \|Y' - Y_1\|^2$$

$$\Rightarrow (Y' - Y_1)^T (\hat{Y}^1 - Y_1)$$

$$\Rightarrow (X_1 B - Y_1)^T (X_2 B - Y_1)$$

diff. w.r.to B

$$\Rightarrow \frac{\partial}{\partial B}\left(B^T X_1^T X_1 B - B^T X_1^T Y_1 - Y_1^T X_2 B + Y_1^T Y_1\right)$$

diff. w.r.to x.

$$\frac{\partial}{\partial x} x^T x = 2x \quad \& \quad \frac{\partial}{\partial x} Ax = A^T.$$

$$\therefore \frac{\partial}{\partial B}(f(B)) = X_1^T X B - 2X^T Y_1$$

$$X^T X B - \cancel{2} X^T Y_1 = 0$$

$$X^T X B = X^T Y_1$$

$$\boxed{B = (X^T X)^{-1} X^T Y_1}$$

c) Assume η is infinitesimally small, w_t is accurately captured by,

$$\frac{dw_t}{dt} = -X_2^T (X_2 w_t - Y_2) \quad \text{—— (1)}$$

To solve:

w_t & loss curve $L(t) = \|w_t - \beta_2\|^2$.

Scanned with CamScanner

General solution for first - ODE is,

$$w_t = \exp(A_1 . t) . A_2 + A_3 . \quad \text{--- } (2)$$

Sub (2) in (1)

$$\frac{dw_t}{dt} = -X_2^T (X_2 (\exp(A_1 t) A_2 + A_3) - Y_2)$$

$$\Rightarrow -X_2^T (X_2 \exp(A_1 t) A_2 + X_2 A_3) - Y_2)$$

$$\Rightarrow -X_2^T X_2 \exp(A_1 t) A_2 - X_2^T X_2 A_3 + X_2^T Y_2$$

$$\boxed{A_1 = -X_2^T X_2} \quad \text{--- } (3)$$

$$X_2^T X_2 A_3 = X_2^T Y_2 . \quad \text{--- } (4)$$

$$A_3 = (X_2^T Y_2)(X_2^T X_2)^{-1} = \hat{\beta}_2 \quad \text{--- } (5)$$

$$\boxed{\therefore A_3 = \hat{\beta}_2}$$

also given that

initialization : $w_0 = \hat{\beta}_1$

So, $A_2 + A_3 = \hat{\beta}_1$

$A_2 = \hat{\beta}_1 - A_3$.

$$\boxed{A_2 = \hat{\beta}_1 - \hat{\beta}_2}$$

Substitute the values of A_1, A_2 and A_3 in eq (2)

$$w_t = \exp(-X_2^T X_2 t)(\hat{\beta}_1 - \hat{\beta}_2) + \hat{\beta}_2$$

Loss curve $L(t) = \|w_t - \hat{\beta}_2\|^2$.

$$\therefore L(t) = \|\exp(-X_2^T X_2 t)(\hat{\beta}_1 - \hat{\beta}_2) + \hat{\beta}_2 - \hat{\beta}_2\|^2$$

$\therefore$ The closed form of $L(t)$ depends only on $X_2, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_2$ and t.

Scanned with CamScanner

d) Assume, $X_2^T X_2 = I$

(i) use this assumption to rewrite $L(t)$ to a simpler form.

$$L(t) = \|(\exp(-It)(\hat{\beta}_1 - \hat{\beta}_2) + \hat{\beta}_2) - \beta_2\|^2$$

$$\Rightarrow \|e^{-t}(\hat{\beta}_1 - \hat{\beta}_2) + \hat{\beta}_2 - \beta_2\|^2$$

(ii) Write $\hat{\beta}_2 - \beta_2$ and $\hat{\beta}_1 - \beta_2$ in terms of $x_1, x_2, \varepsilon_1, \varepsilon_2$ using the fact that $\hat{\beta}_1, \hat{\beta}_2$ are the ordinary least square solutions to linear regression.

Given:

the labels for the data,

$$Y_1 = X_1\beta_1 + \varepsilon_1 \quad \text{——} \quad (1)$$

$$Y_2 = X_2\beta_2 + \varepsilon_2 \quad \text{——} \quad (2)$$

Assuming, $\beta_2 = \beta_1 + \delta R$. —— (3)

From a) we know, $\hat{\beta}_1 = (X_1^T X_1)^{-1} X_1^T Y_1$ —— (4)

From c) we know, $\hat{\beta}_2 = (X_2^T X_2)^{-1}(X_2^T Y_2)$ —— (5)

From (1), $\beta_1 = (Y_1 - \varepsilon_1) X_1^{-1}$

From (2), $\beta_2 = (Y_2 - \varepsilon_2) X_2^{-1}$

$$\hat{\beta}_1 - \beta_2 = \hat{\beta}_1 - (\beta_1 + \delta R)$$

$$\Rightarrow (X_1^T X_1)^{-1} X_1^T Y_1 - ((Y_1 - \varepsilon_1) X_1^{-1} + \delta R)$$

$$\Rightarrow (X_1^T X_1)^{-1} X_1^T Y_1 - Y_1 X_1^{-1} + \varepsilon_1 X_1^{-1} - \delta R.$$

$X_1^T X_1 = I$ (assuming)

$$\Rightarrow I X_1^T Y_1 - Y_1 X_1^{-1} + \varepsilon_1 X_1^{-1} - \delta R.$$

$$\boxed{\hat{\beta}_1 - \beta_2 \Rightarrow Y_1 (X_1^T - X_1^{-1}) + \varepsilon_1 X_1^{-1} - \delta R.}$$

$$\hat{\beta}_2 - \beta_2 = (X_2^T X_2)^{-1} X_2^T Y_2 - (Y_2 - \varepsilon_2) X_2^{-1}$$

$$\Rightarrow I\, X_2^T Y_2 - Y_2 X_2^{-1} + \varepsilon_2 X_2^{-1}$$

$$\boxed{\hat{\beta}_2 - \beta_2 \Rightarrow Y_2 (X_2^T - X_2^{-1}) + \varepsilon_2 X_2^{-1}}$$

Now, $\hat{\beta}_1 - \hat{\beta}_2$ is,

$$\Rightarrow \hat{\beta}_1 - \beta_2 - (\hat{\beta}_2 - \beta_2)$$

$$\Rightarrow Y_1 (X_1^T - X_1^{-1}) + \varepsilon_1 X_1^{-1} - \delta R - [Y_2 (X_2^T - X_2^{-1}) + \varepsilon_2 X_2^{-1}]$$

$$\hat{\beta}_1 - \hat{\beta}_2 = Y_1 (X_1^T - X_1^{-1}) + \varepsilon_1 X_1^{-1} - \delta R - Y_2 (X_2^T - X_2^{-1}) - \varepsilon_2 X_2^{-1}.$$

Loss $L(t) = \| e^{-t} . (\hat{\beta}_1 - \hat{\beta}_2) + \hat{\beta}_2 - \beta_2 \|^2$

$$L(t) = \| e^{t} . Y_1 (X_1^T - X_1^{-1}) + \varepsilon_1 X_1^{-1} - \delta R - Y_2 (X_2^T X_2^{-1}) - \varepsilon_2 X_2^{-1}$$
$$+ Y_2 (X_2^T X_2^{-1}) + \varepsilon_2 X_2^{-1} \|^2$$

iii) Using results, derive $E[L(t)]$, expectation of $L(t)$ over ε_1 and ε_2.

$$L(t) = \| e^{-t} (\hat{\beta}_1 - \hat{\beta}_2) + \hat{\beta}_2 - \beta_2 \|^2$$

$\hat{\beta}_1$ and $\hat{\beta}_2$ are OLS solutions to linear regression.

$$\therefore \hat{\beta}_1 = \beta_1 + \varepsilon_1$$
$$\hat{\beta}_2 = \beta_2 + \varepsilon_2. \qquad [\varepsilon_2 = \hat{\beta}_2 - \beta_2]$$

$$L(t) = \| e^{-t} (\hat{\beta}_1 - (\beta_2 + \varepsilon_2)) + \varepsilon_2 \|^2$$

$$L(t) = \| e^{-t} (\hat{\beta}_1 - (\beta_1 + \delta R + \varepsilon_2))) + \varepsilon_2 \|^2$$

$$\Rightarrow \| e^{-t} (\hat{\beta}_1 - \beta_1 - \delta R - \varepsilon_2) + \varepsilon_2 \|^2$$

$$\Rightarrow \| e^{-t} (\varepsilon_1 - \delta R - \varepsilon_2) + \varepsilon_2 \|^2$$

Integrate $L(t)$ over ε_1 and ε_2 for $E[L(t)]$.

$$E[L(t)] = \frac{e^{-2t} (\varepsilon_1 + \varepsilon_2 (e^{t} - 1) - \delta R)}{12 (e^{t} - 1)} + \varepsilon_1 c_1 + c_2$$

(iv) Optimal stopping time, $t^* = \arg\min_t E[L(t)]$.

Differentiate $E[L(t)]$ w.r.to t and set it to zero and solve for t.

PS3-transfer-learning-handout

December 17, 2020

1 CS7180 Problem Set 3: Transfer Learning for linear regression

[21]:
```
import numpy as np
from numpy.linalg import inv, norm
import matplotlib.pyplot as plt

np.random.seed(0)
```

The default parameter setting

[22]:
```
## parameters
p = 100
n_1 = 200
n_2 = 200

epochs = 500
lr = 0.001
```

1.1 Implement the transfer learning task

First, generate the parameters β_1, and $\beta_2 = \beta_1 + \delta \cdot \mathcal{N}(0, 1)$. Then the features X_1, X_2 from the standard normal distribution $\mathcal{N}(0, 1)$

Also generate the error terms $\varepsilon_1, \varepsilon_2$ from $\sigma_i \cdot \mathcal{N}(0, 1)$, and Y_1, Y_2 from $Y_i = X_i\beta_i + \varepsilon_i$

[31]:
```
def generate_params(p, n_1, n_2, sigma_1, sigma_2, delta):

    # GENERATE DATA HERE ####

    # task 1
    beta_1 = np.random.normal(0, 1, (p, 1))

    X_1 = np.random.normal(0, 1, (n_1, p))
    epsilon_1 = sigma_1 * np.random.normal(0, 1, (n_1, 1))
    Y_1 = X_1 @ beta_1 + epsilon_1

    # task 2
    beta_2 = beta_1 + delta * np.random.normal(0, 1, (p, 1))
```

```
    X_2 = np.random.normal(0, 1, (n_2, p))
    epsilon_2 = sigma_2 * np.random.normal(0, 1, (n_2, 1))
    Y_2 = X_2 @ beta_2 + epsilon_2

    ######################

    return (X_1, Y_1, X_2, Y_2, beta_2)
```

Implement gradient descent, using $w_0 = \hat{\beta}_1 = (X_1^\top X_1)^{-1} X_1^\top Y_1$ as initialization. Save the loss for each step, defined by $\|w_t - \beta_2\|^2$.

[45]:
```
def gradient_descent(params, epochs, lr):

    # unpack features, labels, parameters
    X_1, Y_1, X_2, Y_2, beta_2 = params
    list_dist = []

    # YOUR CODE HERE ####

    beta1 = np.matmul(np.linalg.inv(np.transpose(X_1).dot(X_1)), np.
 ↪transpose(X_1).dot(Y_1))

    for epoch in range(epochs):
        error2 = X_2.dot(beta1)-Y_2
        gd = np.dot(np.transpose(X_2), (X_2.dot(beta1) - Y_2))
        beta1 = beta1 - lr*gd
        list_dist.append(norm(beta1 - beta_2))

    return list_dist
```

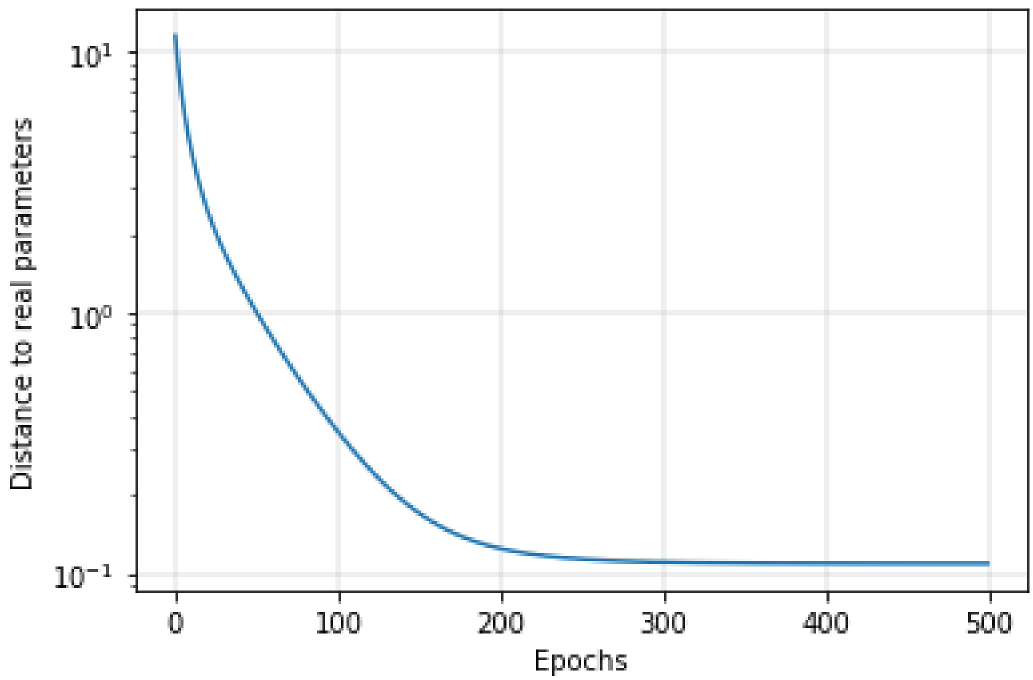
1.2 Plot $\|w_t - \beta_2\|^2$ versus t.

[41]:
```
def plot_dist(list_dist):
    # plot
    plt.xlabel('Epochs')
    plt.ylabel('Distance to real parameters')
    plt.grid(lw=0.4)
    plt.yscale('log')
    plt.plot(np.arange(len(list_dist)), list_dist)
    plt.show()
```

1.3 Try different sets of parameters. Observe the shapes for the loss curve

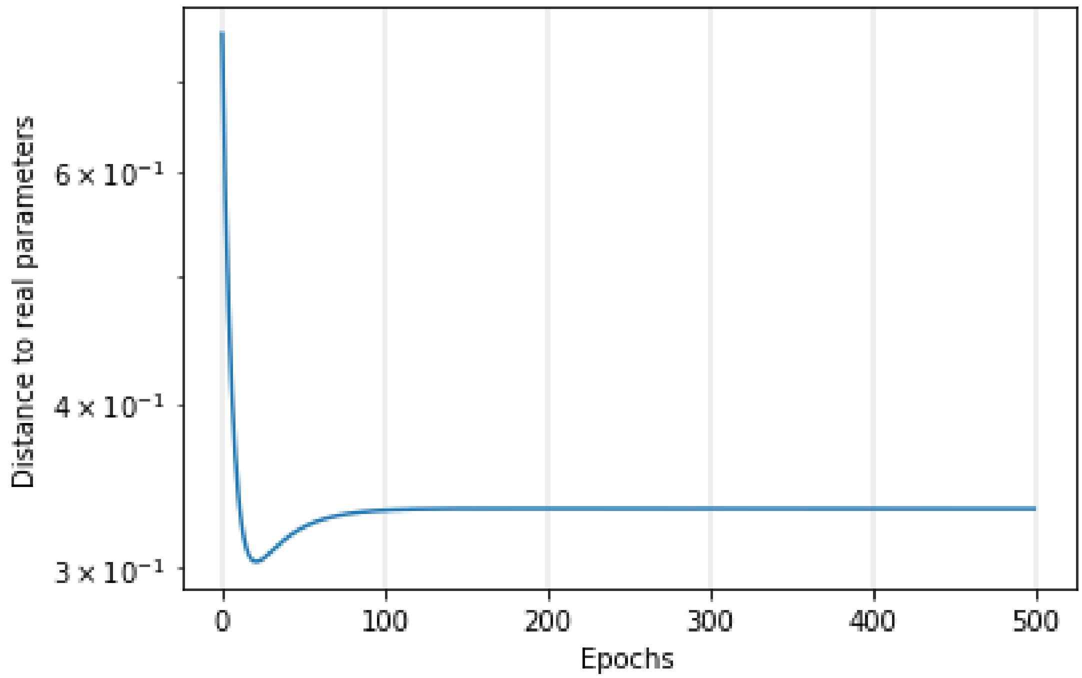
Refer to the problem description for suggested parameter settings

[46]:
```
params = generate_params(p=100, n_1=200, n_2=200, sigma_1=0.1, sigma_2=0.1,␣
↪delta=1.5)
dist = gradient_descent(params, epochs, lr)
plot_dist(dist)
```



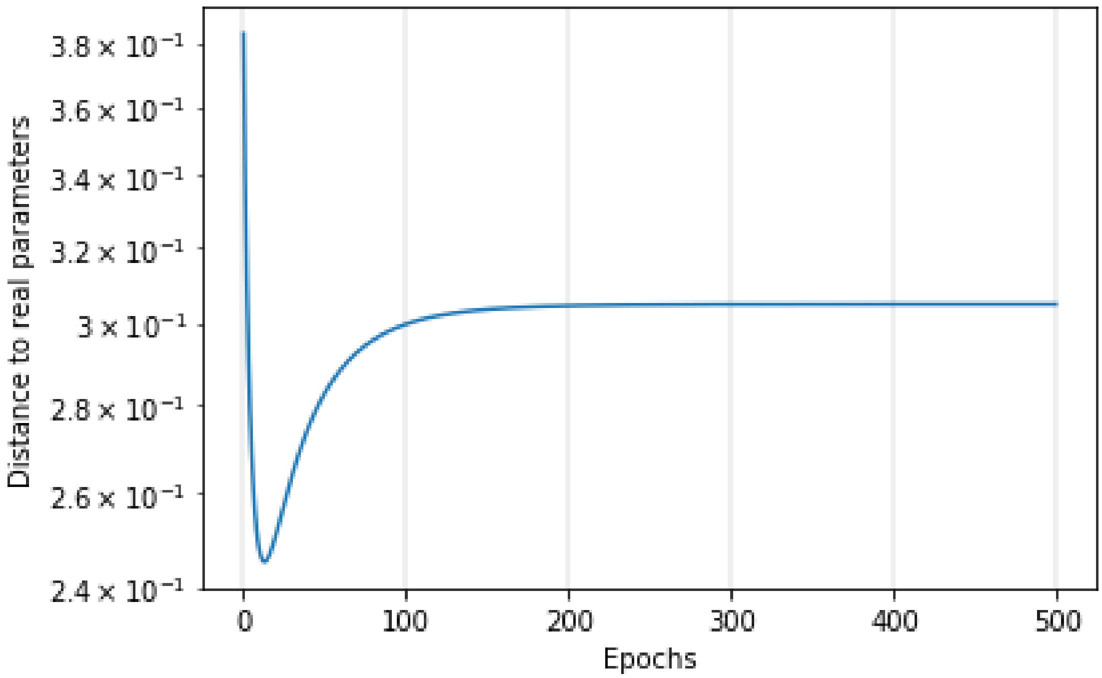
This graph shows that the loss is high for greater delta value, and it ceases after a long trainging.

[47]:
```
params = generate_params(p=100, n_1=200, n_2=200, sigma_1 = 0.1, sigma_2 = 0.3,␣
↪delta = 0.1)
dist = gradient_descent(params, epochs, lr)
plot_dist(dist)
```



This graph represents that the loss is considerably less for *delta = 0.1*, so the loss decreases with decrease in the hyper-parameter *"Delta"*. Also the model converges and finds minimum optimum after training for some time and not as late as previous graph.

[48]:
```
params = generate_params(p=100, n_1=200, n_2=200, sigma_1 = 0.1, sigma_2 = 0.3,␣
↪delta = 0.05)
dist = gradient_descent(params, epochs, lr)
plot_dist(dist)
```



Similarly, with further decrese in *delta*, it is observed that the loss can even go down but eventually comes back to the local minimum. So *delta = 0.05* can be the optimal value.