Robert Heeter
ELEC 378
3/24/2023

Problem Set #8

1. $\sigma(z) = \frac{1}{1+e^{-z}}$

$\hat{c}(\underline{x}_i) = \sigma(\langle \underline{x}_i, \underline{w} \rangle + b)$ $\left\{ \begin{array}{l} \underline{w} \in \mathbb{R}^p \\ b \in \mathbb{R} \\ y_i \in \{0, 1\} \\ \underline{x}_i \in \mathbb{R}^p \end{array} \right\}$ minimize objective fxn: $\ell(\underline{w}, b) = \sum_{i=1}^{n} -y_i \log(\hat{c}(\underline{x}_i)) - (1-y_i)\log(1-\hat{c}(\underline{x}_i))$

a. $\hat{y}_i = \begin{cases} 0, & \hat{c}(\underline{x}_i) \geq 1/2 \\ 1, & \hat{c}(x_i) < 1/2 \end{cases}$ → substitute in $\hat{c}(\underline{x}_i)$

$= \begin{cases} 0, & \sigma(\langle \underline{x}_i, \underline{w} \rangle + b) \geq \frac{1}{2} \\ 1, & \sigma(\langle \underline{x}_i, \underline{w} \rangle + b) < \frac{1}{2} \end{cases}$

$= \begin{cases} 0, & \frac{1}{1+e^{-(\langle \underline{x}_i, \underline{w} \rangle + b)}} \geq \frac{1}{2} \quad Ⓐ \\ 1, & \frac{1}{1+e^{-(\langle \underline{x}_i, \underline{w} \rangle + b)}} < \frac{1}{2} \quad Ⓑ \end{cases}$

For Ⓐ to be true: $\langle \underline{x}_i, \underline{w} \rangle + b \geq 0$

$\therefore \langle \underline{x}_i, \underline{w} \rangle \geq -b$

For Ⓑ to be true: $\langle \underline{x}_i, \underline{w} \rangle + b < 0$

$\therefore \langle \underline{x}_i, \underline{w} \rangle < -b$

Therefore, $\hat{y}_i$ can be rewritten as:

$$\boxed{\hat{y}_i = \begin{cases} 0, & \langle \underline{x}_i, \underline{w} \rangle \geq -b \quad Ⓐ \\ 1, & \langle \underline{x}_i, \underline{w} \rangle < -b \quad Ⓑ \end{cases}}$$ → "threshold linear regression"

b. $\frac{d}{dz}\sigma(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{d}{dz}\left((1+e^{-z})^{-1}\right)$

$= -(1+e^{-z})^{-2}(-e^{-z})$

$= \frac{e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{(1+e^{-z})-1}{1+e^{-z}}\right)$

$= \underbrace{\left(\frac{1}{1+e^{-z}}\right)}_{=\sigma(z)}\left(1 - \underbrace{\frac{1}{1+e^{-z}}}_{=\sigma(z)}\right) = \sigma(z)(1-\sigma(z))$

$$\boxed{\frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z))}$$

(1). c. $\nabla_{\underline{w}}\, \hat{c}(\underline{x}_i) = \hat{c}(\underline{x}_i)(1-\hat{c}(\underline{x}_i))\underline{x}_i$?

$\leftarrow$ gradient $\nabla_{\underline{w}}$ is derivative of $\hat{c}$ with respect to $\underline{w}$

$$\nabla_{\underline{w}}\, \hat{c}(\underline{x}_i) = \frac{d}{dw}\hat{c}(\underline{x}_i) = \frac{d}{dw}\left(\sigma(\langle \underline{x}_i, \underline{w}\rangle + b)\right)$$

From part B: $\frac{d}{dx}\sigma(x)' = \sigma(x)(1-\sigma(x))$

chain rule

$$\therefore \frac{d}{dw}\left(\sigma(\overbrace{\langle \underline{x}_i, \underline{w}\rangle + b}^{x})\right) = \underbrace{\sigma(\langle \underline{x}_i, \underline{w}\rangle + b)}_{=\hat{c}(\underline{x}_i)}\left(1-\underbrace{\sigma(\langle \underline{x}_i, \underline{w}\rangle + b)}_{=\hat{c}(\underline{x}_i)}\right)\underbrace{\frac{d}{dw}(\langle \underline{x}_i, \underline{w}\rangle + b)}_{=\underline{x}_i}$$

$$\therefore \boxed{\nabla_{\underline{w}}\, \hat{c}(\underline{x}_i) = \hat{c}(\underline{x}_i)(1-\hat{c}(\underline{x}_i))\underline{x}_i}$$

d. $\nabla_b\, \hat{c}(\underline{x}_i) = \hat{c}(\underline{x}_i)(1-\hat{c}(\underline{x}_i))$?

$\leftarrow$ gradient ∇_b is derivative of $\hat{c}$ with respect to b

$$\nabla_b\, \hat{c}(\underline{x}_i) = \frac{d}{db}\hat{c}(\underline{x}_i) = \frac{d}{db}\left(\sigma(\langle \underline{x}_i, \underline{w}\rangle + b)\right)$$

From part B: $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$

chain rule

$$\therefore \frac{d}{db}\left(\sigma(\overbrace{\langle \underline{x}_i, \underline{w}\rangle + b}^{x})\right) = \underbrace{\sigma(\langle \underline{x}_i, \underline{w}\rangle + b)}_{\hat{c}(\underline{x}_i)}\left(1-\underbrace{\sigma(\langle \underline{x}_i, \underline{w}\rangle + b)}_{\hat{c}(\underline{x}_i)}\right)\underbrace{\frac{d}{db}(\langle \underline{x}_i, \underline{w}\rangle + b)}_{=1}$$

$$\therefore \boxed{\nabla_b\, \hat{c}(\underline{x}_i) = \hat{c}(\underline{x}_i)(1-\hat{c}(\underline{x}_i))}$$

e. $L(\underline{w}, b) = \sum_{i=1}^{n} -y_i \log(\hat{c}(\underline{x}_i)) - (1-y_i)\log(1-\hat{c}(\underline{x}_i))$

$\frac{d}{dx}\log(x) = \frac{1}{x\ln(10)}$

$$\nabla_{\underline{w}} L(\underline{w}, b) = \sum_{i=1}^{n}\left[\frac{d}{d\underline{w}}\left(-y_i \log(\hat{c}(\underline{x}_i))\right) - \frac{d}{dw}\left((1-y_i)\log(1-\hat{c}(\underline{x}_i))\right)\right]$$

chain rule $\rightarrow$

$$= \sum_{i=1}^{n}\left[\frac{-y_i \frac{d}{dw}(\hat{c}(\underline{x}_i))}{\hat{c}(\underline{x}_i)\ln(10)} - \frac{(1-y_i)\frac{d}{dw}(1-\hat{c}(\underline{x}_i))}{(1-\hat{c}(\underline{x}_i))\ln(10)}\right] \leftarrow \text{chain rule}$$

$$= \sum_{i=1}^{n}\left[\frac{-y_i \cancel{\hat{c}(\underline{x}_i)}(1-\hat{c}(\underline{x}_i))\underline{x}_i}{\cancel{\hat{c}(\underline{x}_i)}\ln(10)} - \frac{(1-y_i)(-1)\hat{c}(\underline{x}_i)\cancel{(1-\hat{c}(\underline{x}_i))}\underline{x}_i}{\cancel{(1-\hat{c}(\underline{x}_i))}\ln(10)}\right]$$

$$\boxed{\nabla_w L(\underline{w}, b) = \sum_{i=1}^{n}\left[\frac{-y_i(1-\hat{c}(\underline{x}_i))\underline{x}_i + (1-y_i)\hat{c}(\underline{x}_i)\underline{x}_i}{\ln(10)}\right]} \rightarrow$$

$$\underline{w}^{t+1} = \underline{w}^t + \mu^t \nabla_w L(\underline{w}, b)$$

(1). f. $\nabla_b L(\underline{w}, b) = \sum_{i=1}^{n} \left[\frac{d}{db}\left(-y_i \log(\hat{c}(\underline{x}_i))\right) - \frac{d}{db}\left((1-y_i)\log(1-\hat{c}(\underline{x}_i))\right) \right]$

chain rule $= \sum_{i=1}^{n} \left[\frac{-y_i \frac{d}{db}(\hat{c}(\underline{x}_i))}{\hat{c}(\underline{x}_i)\ln(10)} - \frac{(1-y_i)\frac{d}{db}(1-\hat{c}(\underline{x}_i))}{(1-\hat{c}(\underline{x}_i))\ln(10)} \right]$ chain rule

$= \sum_{i=1}^{n} \left[\frac{-y_i \cancel{\hat{c}(\underline{x}_i)}(1-\hat{c}(\underline{x}_i))}{\cancel{\hat{c}(\underline{x}_i)}\ln(10)} - \frac{(1-y_i)(-1)\hat{c}(\underline{x}_i)\cancel{(1-\hat{c}(\underline{x}_i))}}{\cancel{(1-\hat{c}(\underline{x}_i))}\ln(10)} \right]$

$$\boxed{\nabla_b L(w, b) = \sum_{i=1}^{n} \left[\frac{-y_i(1-\hat{c}(\underline{x}_i)) + (1-y_i)\hat{c}(\underline{x}_i)}{\ln(10)} \right]}$$

$b^{t+1} = b^t + \mu^t \nabla_b L(\underline{w}, b)$

2. See code and output below. The model performs well — only 2 of the training data digits were mislabeled by the logistic regression. This equates to an error rate of $\approx 3\%$.

3. a. See code and output below. The RMS (root mean square) error was used as a metric for the accuracy of each method. Both visually and with the RMS error the wiener filter via least squares is more accurate than via the least mean square algorithm.

b. The LMS algorithm to estimate the wiener filter via stochastic gradient descent forms an adaptive filter (aka. the filter changes with time, x_t, y_t) which is valuable for adapting to existing data and for better predicting future data. For stocks, this would be useful for cases where one wants to predict future stock price data with the adaptive filter (i.e. using prior stock prices). Least squares to compute the wiener filter is a fixed filter since it only uses data from a fixed time rather than continually / iteratively updating.

Stochastic gradient descent also allows for greater computational efficiency with large datasets compared to performing calculations with the whole dataset in normal gradient descent or least squares. For example, if an invester wants to use a Wiener filter for many stocks over a long time frame (i.e. a very large data set), it may be difficult or impossible to compute the Wiener filter directly with least squares.

ELEC378-HW8

March 24, 2023

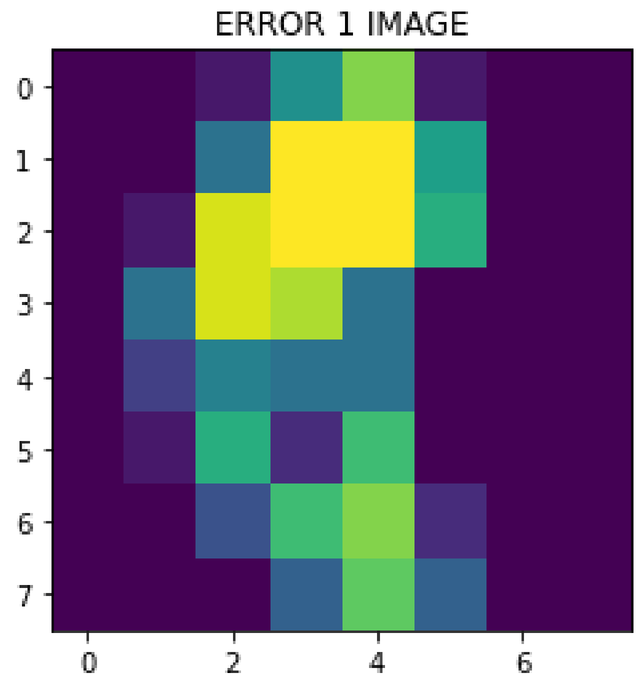
[1]:
```python
# ROBERT HEETER
# ELEC 378 Machine Learning
# 24 March 2023

# PROBLEM SET 8
```

[2]:
```python
# PROBLEM 2

import numpy as np
from matplotlib import pyplot as plt

from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression

# load digits
digits = load_digits()

# select 8's and 9's; save first 288 for training and remaining for testing
all_X = digits['data']
all_y = digits['target']

idx_89 = np.where((all_y==9) | (all_y==8))[0]

X = all_X[idx_89,:]
y = all_y[idx_89]

X_train = X[:288]
X_test = X[288:]

y_train = y[:288]
y_test = y[288:]

# logistic regression
log = LogisticRegression(penalty='l2', tol=0.0001, fit_intercept=True,␣
↪max_iter=200).fit(X_train, y_train)
y_approx = log.predict(X_test)
y_approx_probabilities = log.predict_proba(X_test)
```

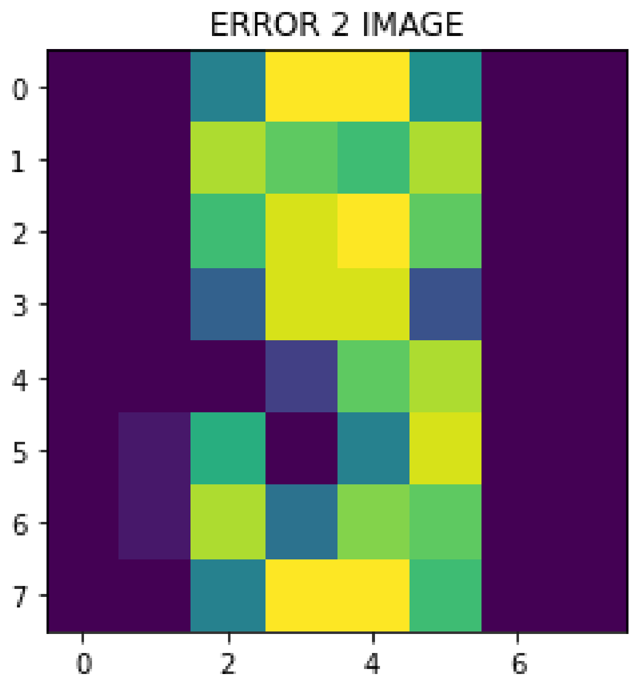
```python
# find errors and display
idx_errors = np.where((y_test-y_approx) != 0)[0]

print(f'y_test digit labels:\n{y_test}')
print(f'\ny_approx digit labels from logistic regression:\n{y_approx}')

print(f'\nerror indices:\n{idx_errors}')

error_rate = len(idx_errors)/len(y_test)
print(f'\nerror rate from test data:\n{100*error_rate}%')

print(f'\nERROR 1:\nactual label (y_test) = {y_test[idx_errors[0]]}')
print(f'predicted label from logistic regression (y_approx) = {y_approx[idx_errors[0]]}')
print(f'model probability of 8 = {y_approx_probabilities[idx_errors[0]][0]}')
print(f'model probability of 9 = {y_approx_probabilities[idx_errors[0]][1]}')

# error 1 image
index = idx_errors[0]
plt.imshow(X_test[index].reshape((8,8)))
plt.title('ERROR 1 IMAGE')
plt.show()

print(f'\nERROR 2:\nactual label (y_test) = {y_test[idx_errors[1]]}')
print(f'predicted label from logistic regression (y_approx) = {y_approx[idx_errors[1]]}')
print(f'model probability of 8 = {y_approx_probabilities[idx_errors[1]][0]}')
print(f'model probability of 9 = {y_approx_probabilities[idx_errors[1]][1]}')

# error 2 image
index = idx_errors[1]
plt.imshow(X_test[index].reshape((8,8)))
plt.title('ERROR 2 IMAGE')
plt.show()
```

```
y_test digit labels:
[8 9 8 8 9 9 8 9 8 9 8 9 8 8 9 8 9 8 8 9 9 8 9 9 9 8 9 8 8 9 9 8 9 8 9 9 9
 8 9 8 8 9 8 9 8 9 9 9 8 9 8 8 9 9 8 9 8 9 8 9 8 8 9 8 9 8]

y_approx digit labels from logistic regression:
[8 9 8 8 9 9 8 9 8 9 8 9 8 8 9 8 9 8 9 9 9 8 9 9 9 8 9 8 8 9 9 8 9 8 9 8 9
 8 9 8 8 9 8 9 8 9 9 9 8 9 8 8 9 9 8 9 8 9 8 9 8 8 9 8 9 8]

error indices:
[18 35]
```

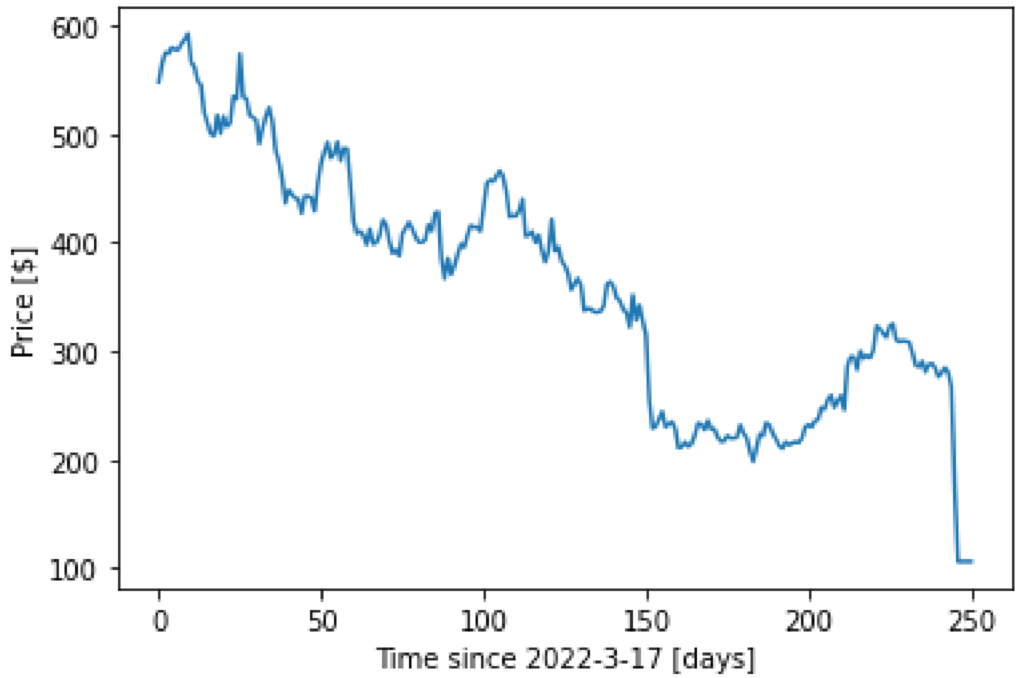
```
error rate from test data:
3.0303030303030303%

ERROR 1:
actual label (y_test) = 8
predicted label from logistic regression (y_approx) = 9
model probability of 8 = 0.25486973286795456
model probability of 9 = 0.7451302671320454
```




```
ERROR 2:
actual label (y_test) = 9
predicted label from logistic regression (y_approx) = 8
model probability of 8 = 0.9989519838330652
model probability of 9 = 0.001048016166934838
```



[3]:
```
# PROBLEM 3

import numpy as np
import matplotlib.pyplot as plt

import yfinance as yf
from scipy.linalg import lstsq

data = yf.Ticker('SIVB')
stock = data.history(period='1d', start='2022-3-17', end='2023-3-17')
price = stock['Open'].values

plt.plot(price)
plt.xlabel('Time since 2022-3-17 [days]')
plt.ylabel('Price [$]')
plt.show()
```



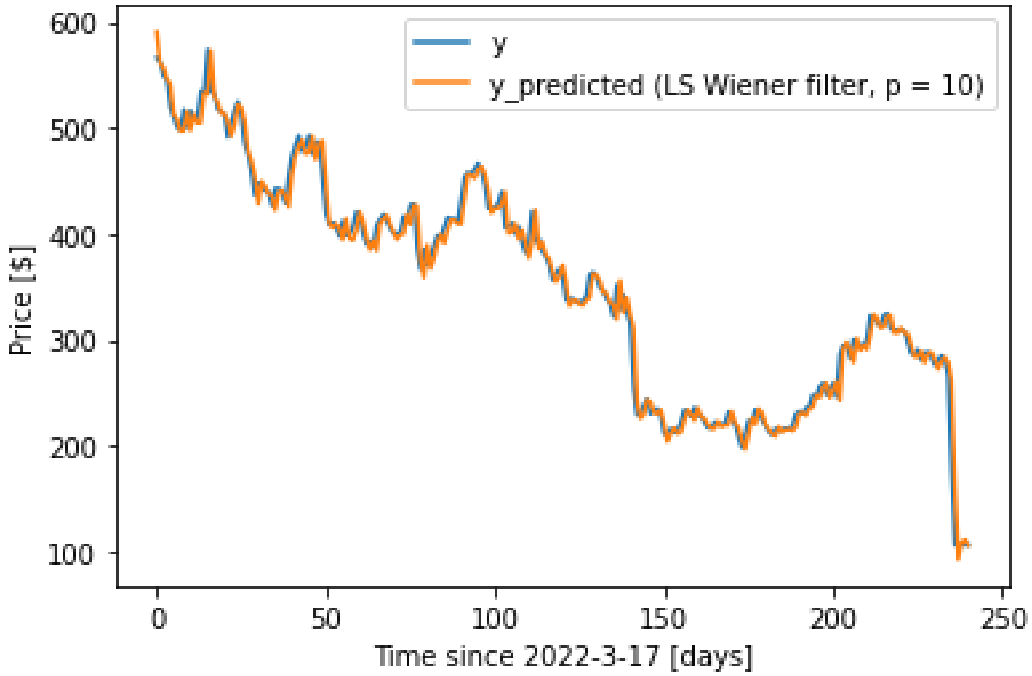
[4]:

```
# constructing the Toeplitz data matrix (from LMS slides page 7)
p = 10
toeplitz_indices = np.arange(p-1,-1,-1)[None,:] + np.arange(len(price)-p)[:
 ↪,None]
prediction_indices = np.arange(p,len(price))

X = price[toeplitz_indices]
y = price[prediction_indices]
```

[5]:

```
# predict y from X using the Wiener filter found via least squares

wiener_filter_LS = np.linalg.inv(X.T @ X) @ X.T @ y
y_predicted_LS = X @ wiener_filter_LS

error_LS = y - y_predicted_LS
RMSE_error_LS = np.average(np.square(error_LS))**(1/2)
print(f'Wiener filter via LS RMS error = ${np.round(RMSE_error_LS,2)}')

plt.plot(y)
plt.plot(y_predicted_LS)
plt.legend(['y',f'y_predicted (LS Wiener filter, p = {p})'])
plt.xlabel('Time since 2022-3-17 [days]')
plt.ylabel('Price [$]')
plt.show()
```

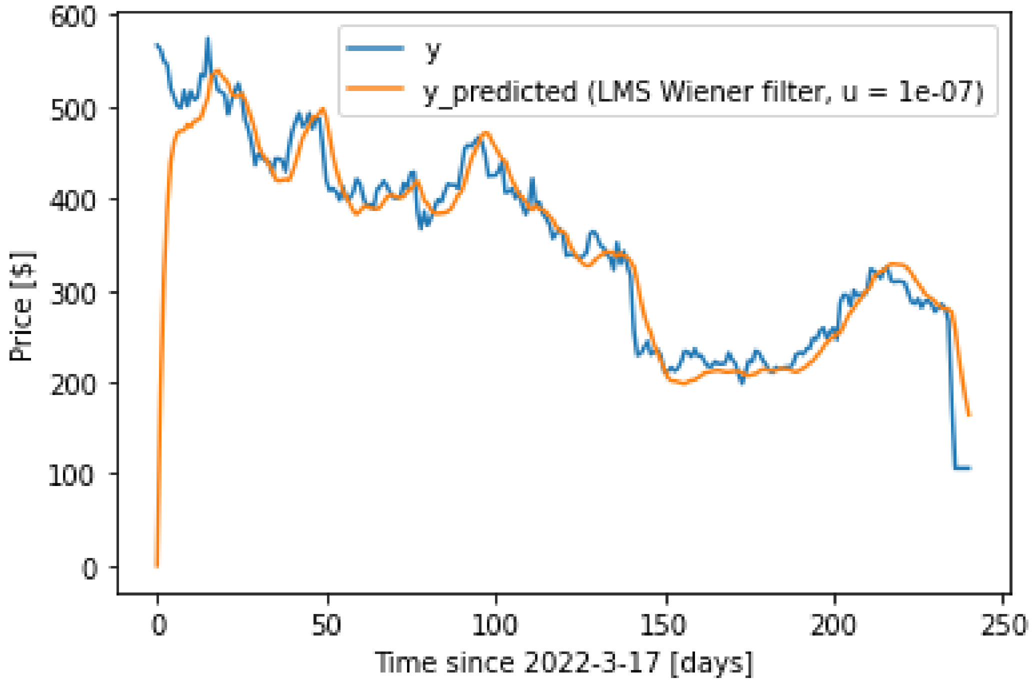
```
Wiener filter via LS RMS error = $14.44
```



[6]:
```
# predict y from X using the Wiener filter estimated adaptively via LMS

# initialize the weights and output
wiener_filter_LMS = np.zeros(p)
y_predicted_LMS = np.zeros(y.shape)
u = 0.0000001

for i in range(toeplitz_indices.shape[0]):
    # predict the output given the current estimate of the wiener filter
    y_predicted_LMS[i] = np.dot(X[i],wiener_filter_LMS)

    # use the prediction error to update the wiener filter estimate
    wiener_filter_LMS = wiener_filter_LMS + u*X[i]*(y[i] - y_predicted_LMS[i])

error_LMS = y - y_predicted_LMS
RMSE_error_LMS = np.average(np.square(error_LMS))**(1/2)
print(f'Wiener filter via LMS RMS error = ${np.round(RMSE_error_LMS,2)}')

plt.plot(y)
plt.plot(y_predicted_LMS)
plt.legend(['y',f'y_predicted (LMS Wiener filter, u = {u})'])
plt.xlabel('Time since 2022-3-17 [days]')
plt.ylabel('Price [$]')
```

```
plt.show()
```

```
Wiener filter via LMS RMS error = $54.85
```



[ ]:

```

```