NYCU Pattern Recognition, Homework 1

312551810, Alexandre PAULY

Part. 1, Coding (60%):

(10%) Linear Regression Model - Closed-form Solution

1. (10%) Show the weights and intercepts of your linear model.

```
2024-04-02 11:57:36.312 | INFO     |   main  :main:218 - LR_CF.weights = array([ 2.19749444,  0.81356777, -1.54225466, -0.27388105]), LR_CF.intercept = -0.8906
```

(40%) Linear Regression Model - Gradient Descent Solution

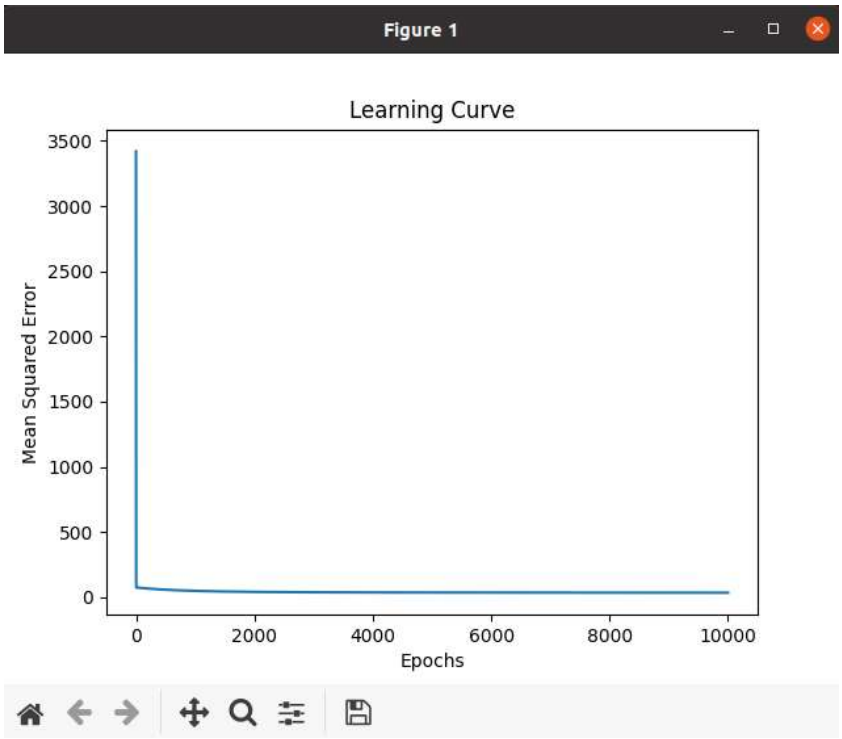
2. (0%) Show the learning rate and epoch (and batch size if you implement mini-batch gradient descent) you choose.

```
losses = LR_GD.fit(train_x, train_y, learning_rate=1e-4, epochs=10000)
```

3. (10%) Show the weights and intercepts of your linear model.

```
2024-04-02 11:57:43.397 | INFO     |   main  :main:224 - LR_GD.weights = array([ 1.90497403,  0.80941907, -1.15106371, -0.24725716]), LR_GD.intercept = -0.9304
```

4. (10%) Plot the learning curve. (x-axis=epoch, y-axis=training loss)

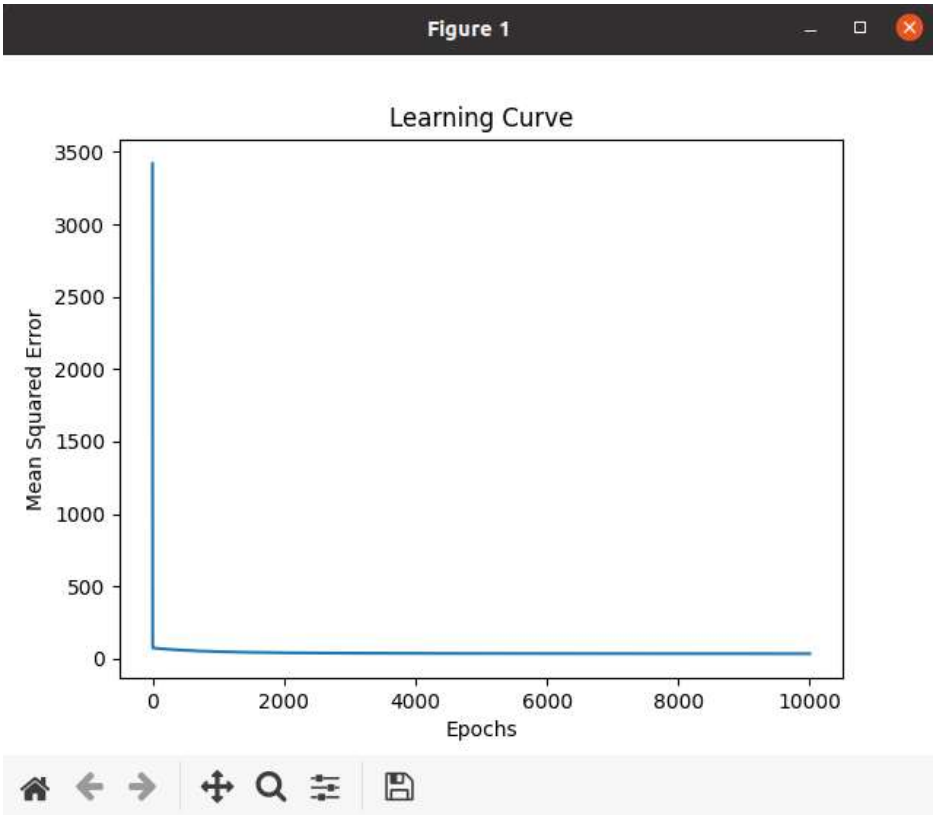


5. (20%) Show your error rate between your closed-form solution and the gradient descent solution.

```
2024-04-02 11:57:43.411 | INFO     | __main__:main:234 - Prediction difference: 2165.4079
2024-04-02 11:57:43.412 | INFO     | __main__:main:239 - mse_cf=33.1460, mse_gd=33.1303. Difference: 0.047%
```

6. (Bonus 5 points) Implement the L1 regularization into the gradient descent method and show the weights, intercept, and learning curve.

```
2024-04-02 12:18:23.741 | INFO     | __main__:main:217 - LR_CF.weights = array([ 2.19749444,  0.81356777, -1.54225466, -0.27388105]), LR_CF.intercept = -0.8906
2024-04-02 12:18:34.336 | INFO     | __main__:main:223 - LR_GD.weights = array([ 1.90342933,  0.80929747, -1.14913245, -0.24634589]), LR_GD.intercept = -0.9307
2024-04-02 12:18:34.340 | INFO     | __main__:main:233 - Prediction difference: 2172.0104
2024-04-02 12:18:34.340 | INFO     | __main__:main:238 - mse_cf=33.1460, mse_gd=33.1385. Difference: 0.023%
```



(10%) Code Check and Verification

7. (10%) Lint the code and show the PyTest results.

```
============================================================ 2 passed, 1 warning in 12.24s ============================================================
● cytech@student-laptop:~/Cours/ING2/S2/Pattern_recognition/Homeworks/HW1/release$ pytest -s
================================================================= test session starts =================================================================
platform linux -- Python 3.8.10, pytest-8.1.1, pluggy-1.4.0
rootdir: /home/cytech/Cours/ING2/S2/Pattern_recognition/Homeworks/HW1/release
collected 2 items

test_main.py 2024-04-02 13:06:48.297 | INFO     | test_main:test_regression_cf:27 - model.weights=array([[3.]]), model.intercept=array([4.])
.2024-04-02 13:06:57.456 | INFO     | test_main:test_regression_gd:39 - model.weights=array([[2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.9991151
4,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514,
        2.99911514, 2.99911514, 2.99911514, 2.99911514, 2.99911514]]), model.intercept=3.99635374863424
.

================================================================== warnings summary ==================================================================
../../../../../../../.local/lib/python3.8/site-packages/jupyter_client/connect.py:22
  /home/cytech/.local/lib/python3.8/site-packages/jupyter_client/connect.py:22: DeprecationWarning: Jupyter is migrating its paths to use standard plat
formdirs
  given by the platformdirs library.  To remove this warning and
  see the appropriate new directories, set the environment variable
  `JUPYTER_PLATFORM_DIRS=1` and then run `jupyter --paths`.
  The use of platformdirs will be the default in `jupyter_core` v6
    from jupyter_core.paths import jupyter_data_dir, jupyter_runtime_dir, secure_write

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
============================================================ 2 passed, 1 warning in 9.58s ============================================================
```

The test_main.py file has been modified to display this information:

```
================================================================= test session starts =================================================================
platform linux -- Python 3.8.10, pytest-8.1.1, pluggy-1.4.0
rootdir: /home/cytech/Cours/ING2/S2/Pattern_recognition/Homeworks/HW1/release
collected 2 items

test_main.py 2024-04-02 13:18:37.147 | INFO     | test_main:test_regression_cf:28 - model.weights=array([[3.]]), model.intercept=array([4.])
.2024-04-02 13:18:44.446 | INFO     | test_main:test_regression_gd:42 - EPOCH 0, loss=30622.06060606061
2024-04-02 13:18:44.447 | INFO     | test_main:test_regression_gd:42 - EPOCH 10000, loss=2.1678106331676865
2024-04-02 13:18:44.448 | INFO     | test_main:test_regression_gd:42 - EPOCH 20000, loss=0.29565425557308317
2024-04-02 13:18:44.449 | INFO     | test_main:test_regression_gd:42 - EPOCH 30000, loss=0.04231077925172794
2024-04-02 13:18:44.450 | INFO     | test_main:test_regression_gd:42 - EPOCH 40000, loss=0.00802789681587633
2024-04-02 13:18:44.451 | INFO     | test_main:test_regression_gd:42 - EPOCH 50000, loss=0.003388677187345874
2024-04-02 13:18:44.451 | INFO     | test_main:test_regression_gd:42 - EPOCH 60000, loss=0.0027608898709784013
2024-04-02 13:18:44.463 | INFO     | test_main:test_regression_gd:44 - model.weights=2.9991, model.intercept=3.99635374863424
.

================================================================== warnings summary ==================================================================
../../../../../../../.local/lib/python3.8/site-packages/jupyter_client/connect.py:22
  /home/cytech/.local/lib/python3.8/site-packages/jupyter_client/connect.py:22: DeprecationWarning: Jupyter is migrating its paths to use standard plat
formdirs
  given by the platformdirs library.  To remove this warning and
  see the appropriate new directories, set the environment variable
  `JUPYTER_PLATFORM_DIRS=1` and then run `jupyter --paths`.
  The use of platformdirs will be the default in `jupyter_core` v6
    from jupyter_core.paths import jupyter_data_dir, jupyter_runtime_dir, secure_write

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
============================================================ 2 passed, 1 warning in 7.70s ============================================================
```

Part. 2, Questions (40%):

1. (10%) Please describe the Vanishing Gradient Problem in detail, and provide **at least two solutions** to overcome this problem.

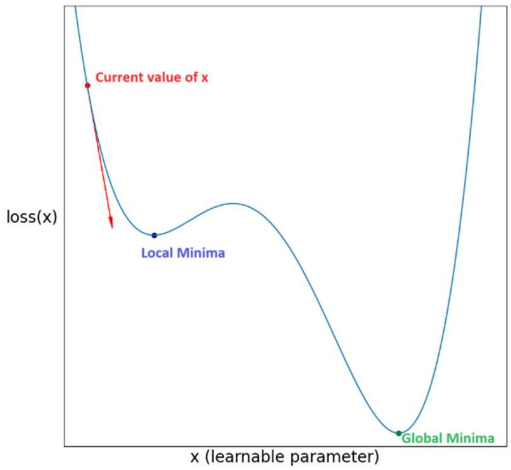
The Vanishing Gradient Problem take place when training recurrent neural networks with gradient-based learning methods and backpropagation. For each training iterations, weights are updated with the partial derivative of the error function with respect to the current weight. When the sequence length increases, the gradient magnitude decrease (or grow uncontrollably), slowing the training process. In some case, the training can be stopped.

An example of a problem comes from in the traditional activation functions like the hyperbolic tangent function which have gradient in the range [-1,1]. During backpropagation, gradients are chain-multiplied through the layers, which can lead to an exponential decrease in gradient with network depth. This means that the initial layers entrain very slowly, if at all.

To solve this problem, we can use the following two solutions:

1. **Use alternative activation functions:** By replacing traditional activation functions with functions whose derivatives can take on larger values, the problem of the vanishing gradient can be avoided. For example, the ReLU (Rectified Linear Unit) function has a derivative of 1 for positive values, which limits the problem of the vanishing gradient.
2. **Weight initialization techniques:** Appropriate initialization of weights can help alleviate the problem of the missing gradient. Techniques such as Xavier initialization or He initialization are used to adjust weights so that the gradient does not become too small during backpropagation. This maintains significant gradients across the network layers and improves learning efficiency.

2. (15%) Gradient descent often suffers from the issue of getting stuck at local minima (refer to the figure provided). Please provide **at least two methods** to overcome this problem and discuss how these methods work.



Gradient descent often encounters the problem of getting stuck in local minima during loss function optimization. This can lead to suboptimal solutions or even completely prevent reaching the desired global minimum.

Here are two commonly used methods to overcome this problem:

1. **Using stochastic gradient descent methods:** Instead of updating the model weights based on the gradient computed over the entire training data at each iteration, stochastic gradient descent (SGD) performs weight updates based on random mini-batches of data. This approach allows for more effective exploration of the parameter space and can help avoid getting stuck in local minima. Additionally, variants such as mini-batch gradient descent can combine the benefits of SGD and batch gradient descent, thereby improving convergence and optimization stability.
2. **Using more advanced optimization algorithms:** More sophisticated optimization algorithms have been developed to overcome the limitations of traditional gradient descent. For example, the Adam algorithm (Adaptive Moment Estimation) adaptively adjusts learning rates for each model

parameter based on estimates of the first and second moments of gradients. This enables Adam to dynamically adapt to different characteristics of the loss landscape and more effectively navigate around local minima. Other popular algorithms include RMSProp (Root Mean Square Propagation) and AdaGrad (Adaptive Gradient Algorithm).

By employing these methods, it is possible to enhance the ability of machine learning models to navigate the parameter space and find higher-quality solutions during loss function optimization.

3. (15%) What are the basic assumptions of Linear regression between the features and the target? How can techniques help Linear Regression extend beyond these assumptions? Please at least answer one technique.

Linear regression model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$$

where,

- Y is the target
- X_i, i = 1,…,n is the features matrix
- β_0 is the intercept and $\beta_1,\beta_2,\ldots,\beta_p$ are the weights
- n is the size of the data
- □ the random part (residual) under the following assumptions:
  - Null expectation (centering): $E(\varepsilon_i) = 0$, i = 1,…,n
  - Homoscedasticity: $Var(\varepsilon_i) = \sigma^2$, pour tout i = 1,…,n
  - Non-correlation (exogeneity): $Cov(\varepsilon_i, \varepsilon_j) = 0$ pour $i \neq j$, i = 1,…,n et j = 1,…,n
  - $\varepsilon_i = y_i - \hat{y}_i$, ε_i -> $N(0, \sigma^2)$
- n > p + 1 et rang(X) = p + 1 (number of observations > number of variables and no relationship between them X_i)
- Linearity: The relationship between features and target is assumed to be linear. This means that the change in the target is proportional to the change in the characteristics, with constant coefficients.

As regularized regression techniques that can help overcome overfitting problems and manage models with many potentially redundant or correlated features, we can consider Ridge and Lasso regression:

1. **Ridge:** The principle is to shrink the ranges of values that the estimated parameters β_hat_{ij} can take, referred to as "shrinkage". The variables must be centered and scaled to avoid variables with high variance from having too much influence. The variable Y must be centered to remove the regression constant. This method uses an L2 penalty. Ridge is very suitable when independent variables are strongly correlated.

2. **Lasso:** The principle is the same as Ridge regression, except that regularization uses the L1 norm. Compared to Ridge regression, Lasso has the advantage of serving as a variable selection device by nullifying certain coefficients βj, and the variables associated with βj = 0 are excluded from the predictive model. Additionally, the variables must be centered and scaled to avoid variables with high variance from having too much influence, as well as Y to remove the regression constant. Overall, Lasso will allow for variable selection to make the model simpler.