Machine Learning Exercise Sheet 4

Linear Regression

Group_369

Fan Xue – fan98.xue@tum.de

Xing Zhou – xing.zhou@tum.de

Jianzhe Liu – jianzhe.liu@tum.de

November 17, 2021

Problem 4

The equation can be found by following steps:

first we change $\sum_{i=1}^{N} t_i$ in to a diagonal matrix $\boldsymbol{T}$. In this matrix $\boldsymbol{T}$, from t_1 to t_N are on the diagonal.

In this way, the given equation are transformed to:

$$E_{weighted}(w) = \frac{1}{2}(\boldsymbol{\Phi w} - \boldsymbol{y})^T \boldsymbol{T}(\boldsymbol{\Phi w} - \boldsymbol{y})$$

then, similar to the optimal solution for OLS:

$$w_{OLS} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$$

we have:

$$w_{weighted} = (\boldsymbol{\Phi}^T\boldsymbol{T}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{T}\boldsymbol{y}$$ ✓

1). To explain this, let's change the β^{-1} in $y_i \sim \mathcal{N}(\boldsymbol{w}^T\phi(x_i), \beta^{-1})$ to ${t_i}^{-1}$.

$$E_{ML} = -\log p = \frac{1}{2}\sum_{i=1}^{N} t_i \left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) - y_i\right]^2 - \frac{1}{2}\sum_{i=1}^{N}\log t_i + \text{const.}$$

this shows that:

$$\nabla E_{ML}(w) = \nabla E_{weighted}(w)$$

2). When this t_i is integer, it can be regarded as t_i times copies of the point $(\boldsymbol{x_i}, y_i)$

Problem 5

To show this, we need to first write down the OLS form:

$$\frac{1}{2}(\boldsymbol{\Phi w} - \boldsymbol{y})^T(\boldsymbol{\Phi w} - \boldsymbol{y})$$

then with the given augmented design matrix and target vector we have:

$$\begin{aligned}\frac{1}{2}(\hat{\boldsymbol{\Phi}}\boldsymbol{w} - \hat{\boldsymbol{y}})^T(\hat{\boldsymbol{\Phi}}\boldsymbol{w} - \hat{\boldsymbol{y}}) &= \frac{1}{2}(\boldsymbol{\Phi w} - \boldsymbol{y})^T(\boldsymbol{\Phi w} - \boldsymbol{y}) + \frac{1}{2}\left(\sqrt{\lambda}\boldsymbol{I_M w}\right)^T\left(\sqrt{\lambda}\boldsymbol{I_M w}\right) \\ &= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\Phi_i w} - y_i)^2 + \frac{\lambda}{2}||\boldsymbol{w}||_2^2\end{aligned}$$

✓

It can be seen that this simplified result is exact equal to the ridge regression loss function.

Problem 6

a). The code won't do what it suppose to do, because it use X and y in the load data to train parameter w, but then again use the same w to compute the $y_p redicted$, this will lead to over-fitting, which means that the best degree will always be the largest degree, cos higher polynomial is always more suitable for this data-set. ✓

b). The solution to this problem can be, split the data-set into two parts, training-set and validation-set. With this parameter x in training-set, we should then choose the lowest error in validation-set to be our target. ✓

Problem 7

We have

$$p(\boldsymbol{w}, \beta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w}, \beta)$$

So

$$\begin{aligned}\log p(\boldsymbol{w}, \beta, \mid \mathcal{D}) &= \log \prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} e^{-\frac{\beta}{2}(\boldsymbol{\Phi w} - \boldsymbol{y})^T(\boldsymbol{\Phi w} - \boldsymbol{y})} \\ &+ \log\left(\frac{1}{\sqrt{(2\pi)^M |\beta^{-1}\boldsymbol{S_0}|}} e^{-\frac{\beta}{2}(\boldsymbol{w} - \boldsymbol{m_0})^T \boldsymbol{S_0}^{-1}(\boldsymbol{w} - \boldsymbol{m_0})} \cdot \frac{b_0^{a_0}\beta^{a_0 - 1}e^{-b_0\beta}}{\Gamma(a_0)}\right) \\ &= \frac{M}{2}\log\beta - \frac{\beta}{2}\boldsymbol{w}^T\left(\boldsymbol{S_0^{-1}} + \boldsymbol{\Phi^T\Phi}\right)\boldsymbol{w} + \beta\left(\boldsymbol{m_0}^T\boldsymbol{S_0}^{-1} + \boldsymbol{y}^T\boldsymbol{\Phi}\right)\boldsymbol{S_N S_N^{-1}} \\ &+ \left(\frac{N}{2} + a_0 - 1\right)\log\beta - \left(b_0 + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} + \frac{1}{2}\boldsymbol{m_0}^T\boldsymbol{S}^{-1}\boldsymbol{m_0}\right)\beta + \text{const.}\end{aligned}$$

We can expand the $p(\boldsymbol{w}, \beta \mid \mathcal{D})$

$$\begin{aligned}\log p(\boldsymbol{w}, \beta \mid \mathcal{D}) &= \frac{M}{2}\log\beta - \frac{\beta}{2}\boldsymbol{w}^{-1}\boldsymbol{S_N}^{-1}\boldsymbol{w} + \beta\boldsymbol{m_N}^T\boldsymbol{S_N}^{-1}\boldsymbol{w} + (a_N - 1)\log\beta \\ &- \beta\left(\frac{1}{2}\boldsymbol{m_N}^T\boldsymbol{S_N}^{-1}\boldsymbol{m_N} - b_N\right) + \text{const.}\end{aligned}$$

Comparing the two expressions, we can find that

$$\begin{aligned}
\boldsymbol{m_N} &= \left(\left(\boldsymbol{m_0}^T \boldsymbol{S_0}^{-1} + \boldsymbol{y}^T \boldsymbol{\Phi}\right) \boldsymbol{S_N}\right)^T \\
\boldsymbol{S_N} &= \left(\boldsymbol{S_0}^{-1} + \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \\
a_N &= \frac{N}{2} + a_0 \\
b_N &= b_0 + \frac{1}{2}\left(\boldsymbol{m_0}^T \boldsymbol{S_0}^{-1} \boldsymbol{m_0} - \boldsymbol{m_N}^T \boldsymbol{S_N}^{-1} \boldsymbol{m_N} + \boldsymbol{y}^T \boldsymbol{y}\right)
\end{aligned}$$ ✓

Problem 8

$$\begin{aligned}
E_{ridge}(\boldsymbol{w}) &= \frac{1}{2}\sum_{i=1}^{N}\left(\boldsymbol{w}^T \boldsymbol{\phi} \boldsymbol{x_i} - y_i\right)^2 + \frac{\lambda}{2}\boldsymbol{w}^T \boldsymbol{w} \\
&= \frac{1}{2}(\boldsymbol{\Phi w} - \boldsymbol{y})^T (\boldsymbol{\Phi w} - y) + \frac{\lambda}{2}\boldsymbol{w}^T w
\end{aligned}$$

The gradient of $E_{ridge}(\boldsymbol{w})$ is

$$\begin{aligned}
\nabla_{\boldsymbol{w}} E_{ridge}(\boldsymbol{w}) &= \boldsymbol{\Phi}^T \boldsymbol{\Phi w} - \boldsymbol{\Phi}^T \boldsymbol{y} + \lambda \boldsymbol{w} \\
&= \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right) \boldsymbol{w} - \boldsymbol{\Phi}^T \boldsymbol{y}
\end{aligned}$$

Let the gradient be zero, we get

$$\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right) \boldsymbol{w} - \boldsymbol{\Phi}^T \boldsymbol{y} \overset{!}{=} 0$$

Therefore

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$ ✓

If $N < M$, the matrix $\boldsymbol{\Phi}^T \boldsymbol{\Phi} \in \mathbb{R}^{M \times M}$ is not invertible. The equation $\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right) \boldsymbol{w} - \boldsymbol{\Phi}^T \boldsymbol{y} = 0$ does not have only solution.
With the regulation, the normal equation is changed to $\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right) \boldsymbol{w} - \boldsymbol{\Phi}^T \boldsymbol{y} = 0$, and the problem is fixed.

Problem 9

a) We want to find the same prediction, which means

$$\hat{y}_i = \boldsymbol{w}^{*T} \boldsymbol{x_i} = \boldsymbol{w}_{new}^T x_{i,new} = a \boldsymbol{w}_{new}^T \boldsymbol{x_i}$$

Hence,

$$\boldsymbol{w_{new}} = \frac{1}{a}\boldsymbol{w}^*$$ ✓

b) According to the result of Problem 8, the solution for $\boldsymbol{w}^*$ on the original dataset $\boldsymbol{X}$ is

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

We want to find the new regulation factor λ_{new} for $\boldsymbol{X_{new}}$, it is to find the λ_{new}, such that $\boldsymbol{w^*_{new}} = \frac{1}{a}\boldsymbol{w^*}$.

We have

$$\begin{aligned}\boldsymbol{w_{new}} &= \left(\boldsymbol{X_{new}}^T\boldsymbol{X_{new}} + \lambda_{new}\boldsymbol{I}\right)^{-1}\boldsymbol{X_{new}}^T\boldsymbol{y} \\ &= \left(a^2\boldsymbol{X}^T\boldsymbol{X} + \lambda_{new}\boldsymbol{I}\right)^{-1} a\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

We can observe that if we let $\lambda_{new} = a^2\lambda$, we will have ✓

$$\begin{aligned}\boldsymbol{w_{new}} &= a\left(a^2\boldsymbol{X}^T\boldsymbol{X} + a^2\lambda\boldsymbol{I}\right)^{-1} a\boldsymbol{X}^T\boldsymbol{y} \\ &= \frac{1}{a}\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= \frac{1}{a}\boldsymbol{w}^*\end{aligned}$$

which satisfies the condition.

exercise_04_linear_regression

November 17, 2021

1 Programming Task: Linear Regression

```
[ ]: import numpy as np

     from sklearn.datasets import load_boston
     from sklearn.model_selection import train_test_split
```

1.1 Your task

This notebook provides a code skeleton for performing linear regression. Your task is to complete the functions where required. You are only allowed to use built-in Python functions, as well as any `numpy` functions. No other libraries / imports are allowed.

In the beginning of every function there is docstring which specifies the input and and expected output. Write your code in a way that adheres to it. You may only use plain python and anything that we imported for you above such as numpy functions (i.e. no scikit-learn classifiers).

1.2 Exporting the results to PDF

Once you complete the assignments, export the entire notebook as PDF and attach it to your homework solutions. The best way of doing that is 1. Run all the cells of the notebook (`Kernel -> Restart & Run All`) 2. Export/download the notebook as PDF (`File -> Download as -> PDF via LaTeX (.pdf)`) 3. Concatenate your solutions for other tasks with the output of Step 2. On Linux you can simply use `pdfunite`, there are similar tools for other platforms too. You can only upload a single PDF file to Moodle.

Make sure you are using `nbconvert` **Version 5.5 or later** by running `jupyter nbconvert --version`. Older versions clip lines that exceed page width, which makes your code harder to grade.

1.3 Load and preprocess the data

In this assignment we will work with the Boston Housing Dataset. The data consists of 506 samples. Each sample represents a district in the city of Boston and has 13 features, such as crime rate or taxation level. The regression target is the median house price in the given district (in $1000's).

More details can be found here: http://lib.stat.cmu.edu/datasets/boston

```
[ ]: X , y = load_boston(return_X_y=True)
```

```python
# Add a vector of ones to the data matrix to absorb the bias term
# (Recall slide #7 from the lecture)
X = np.hstack([np.ones([X.shape[0], 1]), X])
# From now on, D refers to the number of features in the AUGMENTED dataset (i.e.
↪ including the dummy '1' feature for the absorbed bias term)

# Split into train and test
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

```
C:\ProgramData\Miniconda3\envs\ML\lib\site-
packages\sklearn\utils\deprecation.py:87: FutureWarning: Function load_boston is
deprecated; `load_boston` is deprecated in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

1.4 Task 1: Fit standard linear regression

```
[ ]: def fit_least_squares(X, y):
         """Fit ordinary least squares model to the data.

         Parameters
         ----------
         X : array, shape [N, D]
             (Augmented) feature matrix.
         y : array, shape [N]
             Regression targets.

         Returns
         -------
         w : array, shape [D]
             Optimal regression coefficients (w[0] is the bias term).

         """
         ### YOUR CODE HERE ###

         w = np.linalg.inv(X.T @ X) @ X.T @ y

         return w
```

✓

1.5 Task 2: Fit ridge regression

```
[ ]: def fit_ridge(X, y, reg_strength):
         """Fit ridge regression model to the data.

         Parameters
         ----------
         X : array, shape [N, D]
             (Augmented) feature matrix.
         y : array, shape [N]
             Regression targets.
         reg_strength : float
             L2 regularization strength (denoted by lambda in the lecture)

         Returns
         -------
         w : array, shape [D]
             Optimal regression coefficients (w[0] is the bias term).

         """
         ### YOUR CODE HERE ###

         w = np.linalg.inv(X.T @ X + np.identity(~~1~~)) @ X.T @ y
```

✗

```
    return w
```

1.6 Task 3: Generate predictions for new data

```
[ ]: def predict_linear_model(X, w):
         """Generate predictions for the given samples.

         Parameters
         ----------
         X : array, shape [N, D]
             (Augmented) feature matrix.
         w : array, shape [D]
             Regression coefficients.

         Returns
         -------
         y_pred : array, shape [N]
             Predicted regression targets for the input data.

         """
         ### YOUR CODE HERE ###

         y_pred = X @ w          ✓

         return y_pred
```

1.7 Task 4: Mean squared error

```
[ ]: def mean_squared_error(y_true, y_pred):
         """Compute mean squared error between true and predicted regression targets.

         Reference: `https://en.wikipedia.org/wiki/Mean_squared_error`

         Parameters
         ----------
         y_true : array
             True regression targets.
         y_pred : array
             Predicted regression targets.

         Returns
         -------
         mse : float
             Mean squared error.

         """
```

```
    ### YOUR CODE HERE ###

    mse = ((y_true - y_pred)**2).mean()

    return mse
```

✓

1.8 Compare the two models

The reference implementation produces * MSE for Least squares ≈ **23.96** * MSE for Ridge regression ≈ **21.03**

You results might be slightly (i.e. ±1%) different from the reference soultion due to numerical reasons.

[ ]:

```
# Load the data
np.random.seed(1234)
X , y = load_boston(return_X_y=True)
X = np.hstack([np.ones([X.shape[0], 1]), X])
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)

# Ordinary least squares regression
w_ls = fit_least_squares(X_train, y_train)
y_pred_ls = predict_linear_model(X_test, w_ls)
mse_ls = mean_squared_error(y_test, y_pred_ls)
print('MSE for Least squares = {0}'.format(mse_ls))

# Ridge regression
reg_strength = 1
w_ridge = fit_ridge(X_train, y_train, reg_strength)
y_pred_ridge = predict_linear_model(X_test, w_ridge)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
print('MSE for Ridge regression = {0}'.format(mse_ridge))
```

```
MSE for Least squares = 23.964571384949412
MSE for Ridge regression = 22.254437477617838
```

✗