Deep Learning (IST, 2021-22)

Practical 3: Linear and Logistic Regression

André Martins, Andreas Wichert, Taisiya Glushkova, Luis Sá Couto

Question 1

Consider the following training data:

$$\boldsymbol{x}^{(1)} = [-2.0], \boldsymbol{x}^{(2)} = [-1.0], \boldsymbol{x}^{(3)} = [0.0], \boldsymbol{x}^{(4)} = [2.0]$$

$$y^{(1)} = 2.0, y^{(2)} = 3.0, y^{(3)} = 1.0, y^{(4)} = -1.0.$$

1. Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..
2. Predict the target value for $\boldsymbol{x}_{\text{query}} = [1]$.
3. Sketch the predicted hyperplane along which the linear regression predicts points will fall.
4. Compute the mean squared error produced by the linear regression.

Question 2

Consider the following training data:

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \boldsymbol{x}^{(2)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \boldsymbol{x}^{(4)} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$y^{(1)} = 1.4, y^{(2)} = 0.5, y^{(3)} = 2, y^{(4)} = 2.5$$

1. Find the closed form solution for a linear regression that minimizes the sum of squared errors on the training data..
2. Predict the target value for $\boldsymbol{x}_{\text{query}} = \begin{bmatrix} 2 & 3 \end{bmatrix}^\top$.
3. Sketch the predicted hyperplane along which the linear regression predicts points will fall.
4. Compute the mean squared error produced by the linear regression.

Question 3

Consider the following training data:

$$\boldsymbol{x}^{(1)} = [3], \quad \boldsymbol{x}^{(2)} = [4], \quad \boldsymbol{x}^{(3)} = [6], \quad \boldsymbol{x}^{(4)} = [10], \quad \boldsymbol{x}^{(5)} = [12]$$

$$y^{(1)} = 1.5, \quad y^{(2)} = 9.3, \quad y^{(3)} = 23.4, \quad y^{(4)} = 45.8, \quad y^{(5)} = 60.1$$

1. Adopt a logarithmic feature transformation $\phi(x_1) = \log(x_1)$ and find the closed form solution for this non-linear regression that minimizes the sum of squared errors on the training data.
2. Repeat the exercise above for a quadratic feature transformation $\phi(x_1) = x_1^2$.
3. Plot both regressions.
4. Which is a better fit, a) or b)?

Question 4

Consider the following training data:

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}^{(2)} = \begin{bmatrix} 0 \\ 0.25 \end{bmatrix}, \quad \boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}^{(4)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$y^{(1)} = 0, \quad y^{(2)} = 1, \quad y^{(3)} = 1, \quad y^{(4)} = 0$$

In this exercise, we will consider binary logistic regression:

$$p_{\boldsymbol{w}}(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})}$$

And we will use the cross-entropy loss function:

$$L(\boldsymbol{w}) = -\sum_{i=1}^{N} \log\left(p_{\boldsymbol{w}}\left(y^{(i)} \mid \boldsymbol{x}^{(i)}\right)\right) = -\sum_{i=1}^{N}\left(y^{(i)} \log \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right)\right)$$

1. Determine the gradient descent learning rule for this unit.
2. Compute the first gradient descent update assuming an initialization of all zeros. Assume a learning rate of 1.0.
3. Compute the first stochastic gradient descent update assuming an initialization of all zeros. Assume a learning rate of 1.0.

Question 5

Now it's time to try multi-class logistic regression on real data and see what happens.

1. Load the UCI handwritten digits dataset using `scikit-learn`:

```python
from sklearn.datasets import load_digits
data = load_digits()
```

This is a dataset containing 1797 8x8 input images of digits, each corresponding to one out of 10 output classes. You can print the dataset description and visualize some input examples with:

```
print(data.DESCR)

import matplotlib.pyplot as plt
plt.gray()
for i in range(10):
    plt.matshow(data.images[i])
plt.show()
```

Randomly split this data into training (80%) and test (20%) partitions. This can be done with:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

2. Run your implementation of the multi-class logistic regression algorithm on this dataset, using stochastic gradient descent with $\eta = 0.001$. Plot the loss and the training and test accuracy over the epochs.

3. Use `scikit-learn`'s implementation of multi-class logistic regression. This can be done with

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(fit_intercept=False, penalty='none')
clf.fit(X_train, y_train)
print(clf.score(X_train, y_train))
print(clf.score(X_test, y_test))
```

Compare the resulting accuracies.