Problem Set 04: Linear Regression and Classifier with Regularization

Johannes C. B. Dietschreit, Sascha Mausenberger

Due: 04.11.2024

The problems below are meant to be solved using Python functions and libraries. You can do so by either writing short scripts or using Jupyter notebooks. **Report your results in the Quiz on Moodle** to obtain a grade.

Introduction

In this PSet you will primarily work on data processing/learning (best use `numpy`) and visualization (best use `matplotlib` or `seaborn`).
Code, which will be of help / you should base your code on for the exercises below, is:

```python
import numpy as np

def classify(x: np.ndarray | list[float] | float,
             w: np.ndarray | list[float],
             b: float=0) -> int | np.ndarray]:
    return np.sign(np.dot(x, w) + b)


def train_perceptron_SGD(Xs: np.ndarray,
                         ys: np.ndarray,
                         T:int = 1000,
                         seed:int = 42) -> tuple[np.ndarray, float]:
    np.random.seed(seed)
    n_samples, dim = Xs.shape
    w = np.zeros(dim)
    b = 0.0
    for t in range(T):
        rand_idx = np.random.randint(0, n_samples)
        x, y = Xs[rand_idx], ys[rand_idx]
        if y*classify(x, w, b) <= 0:
            w += y*x
            b += y

    return w, b
```

The Problems

1 Linear Classification with Regularization

For this part of the problem set you will need to load the arrays `X.npy` and `y.npy`.

1.1 Load the data set. Visualize the data for yourself to answer the question "Is the data linearly separable?" (0.5 P)

1.2 Split the data (Xs and ys) using the function `train_test_split` from sklearn with the usual setting `test_size=0.2` and `random_state=42`. Implement a `train_Pegasos_SGD` in analogy to the provided `train_perceptron_SGD`. Train 2 models, one with `lam=0.0` and the other with `lam=1.0`. Both models use `T=1000` and `seed=42`.

Report the slope of the decision boundary (i.e., if one were to draw the line $\mathbf{w} \cdot \mathbf{x} + b = 0$ in 2D as $y = mx + y_0$ we are asking for the m) rounded to 3 digits after the decimal point for both models. (each 0.5 P)

1.3 Using the same train-test splits from before. Train 100 models for the perceptron and the Pegasos algorithm each. Use `T=1000` for both models and for the Pegasos model also `lam=1.0`. Use the random seed 0 to 99 (`= range(100)`).

Plot the test data with the 100 decision boundaries, make 2 separate (sub)plots one for each model type. (1.5 P)

1.4 Again we use the same train-test split from the beginning of the exercise. Now we will only use the pegasos algorith. Always use `T=1000` and `seed=42`. Train a model for the following values of regularization strength `np.linspace(0, 10, 100)`. Calculate the score (1 - missclassification error) for each of these model. Plot the score against the regularization strength. (1.0 P)

2 Linear Regression

For this part of the problem set you need to load the arrays `X_regression.npy` and `y_regression.npy`.

2.1 Visualize the data in a plot. (0.5 P)

2.2 Use again `train_test_split(X, y, test_size=0.2, random_state=42)` to split the data into train and test sets.

Use the closed from solution (matrix inversion) to compute the slope and offset that best fits the data. Round your results to 3 digits after the decimal point. (each 0.5 P)

2.3 Compute the mean sum of squared residuals (mean squared error, MSE) and the mean of the absolute of the residuals (mean absolute error) using the estimated parameters and the test data. If you have not been able to perform the closed form solution use $w_1 = 2.718$ and $b = 0.693$ instead. (each 0.5 P)

2.4 Make a histogram of the residuals (difference between prediction and true label). To have enough data, use the full data set not the split parts. How are they distributed? (0.5 P)

2.5 Using the SGD template of the Perceptron as a template, implement linear regression stochastic gradient descent as outlined in the script (do NOT include any regularization!). For the learning rate use $\eta_t = 1/(t+1)$, so that $\eta_0 = 1$ for the first step. As in the case for the linear classifier, train 100 models using the random number seeds $[0, 99]$, each for 100 steps, i.e., `T=100`. Plot the trained linear models in the same plot with the closed form solution. (1.0 P)

2.6 Compute the residuals for those 100 models. Compute the MSE, for each model. Visualize them as a histogram draw in the closed form solution as a vertical line for comparison. (1.0 P)

2.7 Rewrite you stochastic Linear Regression algorithm such that it saves the intermediate values for slope and offset. Train a single model for 100 steps `T=100`. This should give you 101 values for the parameters, where element 0 is the initial values and number 100 the parameters after the last training step.

Compute the mean sqaured error for the test and training setfor each of the 101 tuples of slope and offset. Plot the two error curves (training and test) against the epoch number. (1.0 P)