ENGR 421

Homework 02: Discrimination by Regression

Deadline: October 21, 2019, 11:59 PM

Sitare Arslantürk, 57677

All the following formulas are taken from “Introduction to Machine Learning” by Ethem Alpaydın from chapter 10.8. In this assignment, a multiclass classification is implemented via discrimination by regression. We will be dealing with a probabilistic model described as

$$\boldsymbol{r^t} = \boldsymbol{y^t} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} \sim N_k(0, \sigma^2 \boldsymbol{I_k})$. Assuming a linear model for each class,

$$y_i^t = \text{sigmoid}(\boldsymbol{w}_i^T \boldsymbol{x}^t + w_{i0}) = \frac{1}{1 + \exp[-(\boldsymbol{w}_i^T \boldsymbol{x}^t + w_{i0})]}$$

Assuming $\boldsymbol{r}|\boldsymbol{x} \sim N(y, \sigma^2)$ the sample likelihood in regression is,

$$l(\{\boldsymbol{w}_i, w_{i0}\}_i | \mathcal{X}) = \prod_t \frac{1}{(2\pi)^{K/2} |\Sigma|^{1/2}} \exp\left[-\frac{\|\boldsymbol{r}^t - \boldsymbol{y}^t\|^2}{2\sigma^2}\right]$$

Thus, the error function becomes

$$E(\{\boldsymbol{w}_i, w_{i0}\}_i | \mathcal{X}) = \frac{1}{2} \sum_t \|\boldsymbol{r}^t - \boldsymbol{y}^t\|^2 = \frac{1}{2} \sum_t \sum_i (r_i^t - y_i^t)^2$$

The update functions for I = 1…K are

$$\begin{aligned} \Delta \boldsymbol{w}_i &= \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \boldsymbol{x}^t \\ \Delta w_{i0} &= \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \end{aligned}$$

The given formulas are applied in the algorithm step by step following the following recipe:

1- W, w0 and eta are initialized

2- The gradients of ΔW and $\Delta w0$ are calculated

3- W and w0 are updated using ΔW and $\Delta w0$

4- Go to step 2, if there is any change in the parameters until the max_iteration is reached, STOP

The confusion matrix is printed with the help of sklearn metrics library and to plot the graph, I used matplotlib. The following are the pseudocodes of the fitting and prediction functions of the Regression class.

```
def fit(X, Y):
    error = []
    for i in max_iteration:
        pY = sigmoid(w_i^t x^t + w_i0)
        dW += eta * ∑(Y − pY) * pY * (1 − pY) * x^t
```

```
        dw0 += eta * ∑(Y − pY) * pY * (1 − pY)
        err = ∑(|pY − Y|²)/2
        error.append(err)

def predict(X):
    return argmax{sigmoid(w_i^t x^t + w_i0)}
```