Homework # 4

|  |  |
| --- | --- |
| 姓名 | 學號 |
| 操之晴 | R13922A04 |

where

since . Then Hessian , where D is

Thus,



From the SGD update rule for MLR, we know that the gradient of the error function

with respect to each weight vector is as follows:

For the correct class :

For the other classes :

Let represent the k-th element of u, corresponding to the partial derivative for class k.

For :

For :

Thus, the vector u in is:

where the only positive term is at position , representing the correct class, while all other terms are negative.

Thus,

For Softmax function,

Since K=2, we can simplify as

When ,

When ,

So,

Take the partial derivatives of the above expression with respect to and separately, we get:

or

So,

Also from , we can get

In this case, we can reduce to because is zero (we fit exactly two points).

Thus,

一張含有 文字, 筆跡, 字型, 數字 的圖片

自動產生的描述



一張含有 文字, 螢幕擷取畫面, 行, 字型 的圖片

自動產生的描述 一張含有 文字, 螢幕擷取畫面, 軟體, 作業系統 的圖片

自動產生的描述

Figure 10a. average of Ein, Eout in Linear & SGD Figure 10b. snapshot of my code

(blue curve) drops quickly at the start, which is typical for SGD. It finds a decent solution fast in the early iterations, but then the convergence slows down. As the iterations increases, the model has settled into a stable error level on the training data. This usually means the parameters are in a good range.

(orange curve) drops initially but then rises slightly. This means some variation in the model's generalization ability on the test set, possibly due to SGD’s randomness, making it a bit of overfitting.

The red and blue dashed lines, which show the average , linear regression has much lower error than SGD. This is because linear methods use all data, making it easier to find a globally optimal solution.

*For problems 10 to 12, the full code has been made available on* [*GitHub*](https://github.com/7ching/HTML/tree/main/HW4)*.*

*Feel free to check it out if required or interested!*



一張含有 文字, 圖表, 螢幕擷取畫面, 繪圖 的圖片

自動產生的描述 一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

Figure 11a. histogram of Ein Gain Figure 11b. snapshot of my code



Figure 11c. the average of Ein Gain

Most Ein gain values are clustered around low values (0 to 20), showing that the polynomial transform didn't greatly improve error in most experiments. This might mean the original data's linear features were already sufficient, or the polynomial transform didn’t add much useful information in these cases.

As Ein gain gets larger (40 or 60 and above), the frequency decreases, meaning polynomial transform only significantly reduced training error in a few cases. This might be because, in certain random samples, the polynomial transform better captured nonlinear features.

In summary, the results show that the polynomial transform has limited error improvement in most cases, but it can make a big difference in a few cases. This suggests that the transform is especially effective with data that has certain nonlinear features but helps less with more linear data.

一張含有 文字, 螢幕擷取畫面, 圖表, 陳列 的圖片

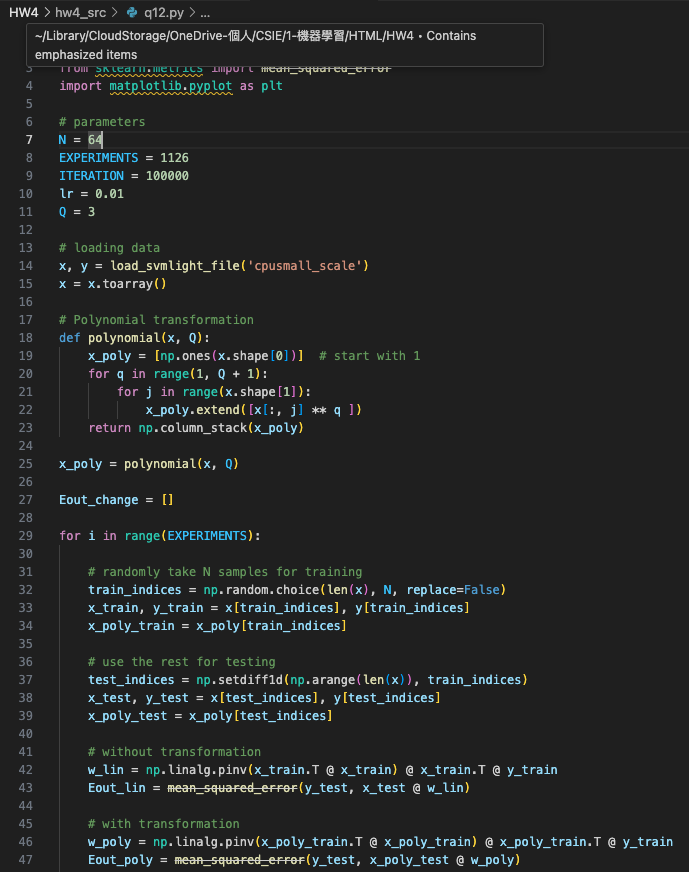
自動產生的描述

Figure 12a. histogram of Eout Change Figure 12b. snapshot of my code



Figure 12c. the average of Eout Change

Most of the Eout change values are close to zero, meaning that the polynomial transform had minimal impact on the test error. This suggests that in most cases, the polynomial transform didn’t significantly improve the model’s generalization on the test set and behaved similarly to the linear model.

A few negative values (below -0.4) show that, in certain experiments, the polynomial transform actually reduced the test error. This shows that in specific cases, the transform captured some nonlinear features, leading to better generalization.

In summary, the results show that the polynomial transform has minimal effect on test error, with significant improvements in only a few cases. This means that the transform’s effect on the test set is inconsistent, performing well on specific nonlinear data but offering little benefit in most cases.