L1 Regularization

L1 regularization effect on the neural network weight values is that it penalizes weight values that are close to 0 by making them equal to 0. Negative weight values also take a value of 0; so if a weight value is -2, under the effect of L1 regularization, it becomes 0.

The general intuition with L1 regularization is that, if a weight value is close to 0 or very small, then it’s negligible when it comes to the overall performance of the model, therefore making it 0 does not affect the model’s performance and can reduce the memory capacity of the model.

The L1 regularization penalty is computed as: loss = l1 * reduce_sum(abs(x))

- L1 penalizes the sum of the absolute values of the weights (|weight|)

برای پیدا کردن m_1l_reg و size_h مناسب؛ با مقادیر مختلف test accuracyرا محاسبه میکنیم.

hidden_size: 32 reg_l1_param: 10e-06 test accuracy: 0.8677999973297119

hidden_size: 32 reg_l1_param: 10e-05 test accuracy: 0.8646000027656555

hidden_size: 32 reg_l1_param: 0.001 test accuracy: 0.871399998664856

hidden_size: 32 reg_l1_param: 0.01 test accuracy: 0.8712000250816345

hidden_size: 32 reg_l1_param: 0.1 test accuracy: 0.8483999967575073

hidden_size: 32 reg_l1_param: 1.0 test accuracy: 0.10700000077486038

hidden_size: 100 reg_l1_param: 0.001 test accuracy: 0.8835999965667725 #BEST

hidden_size: 200 reg_l1_param: 0.001 test accuracy: 0.8903999924659729

hidden_size: 300 reg_l1_param: 0.001 test accuracy: 0.8971999883651733

hidden_size: 400 reg_l1_param: 0.001 test accuracy: 0.8956000208854675

hidden_size: 500 reg_l1_param: 0.001 test accuracy: 0.8953999876976013

hidden_size: 100 reg_l1_param: 0.0001 test accuracy: 0.8822000026702881

hidden_size: 100 reg_l1_param: 0.01 test accuracy: 0.8808000087738037

با توجه به اینکه افزایش size_h سرعت‌را به شدت کاهش میدهد و تاثیر ناچیزی بر accuracy دارد؛ m_l1_reg = 0.001 و size_h = 100 انتخاب میکنیم که accuracy قابل قبول 0.88 دارد.

Feature Selection

در ابتدا شبکه‌را با همه x_train آموزش میدهیم. وزن های لایه اول‌را بدست می آوریم. این وزن هارا به متد feature_selection میدهیم.

متد feature_selection: فیچرها و وزن های هر فیچررا دریافت میکند. برای هر فیچر مجموع قدر مطلق وزن های آن فیچررا محاسبه میکند.(تا وزن های منفی و مثبت همدیگررا خنثی نکنند) در نهایت 10 فیچر که بیشترین مجموع وزن‌را دارند انتخاب میکند.

از مجموعه x_train فیچرهای برگزیده‌را جدا میکنیم ← x_train_best

svmرا با x_train_best آموزش میدهیم. Accuracyرا محاسبه میکنیم. با توجه به:

score = accuracy – penalty

scoreرا بدست می آوریم و ذخیره میکنیم.

از مجموعه x_train فیچرهای برگزیده‌را حذف میکنیم ← x_train_new

حال مراحل‌را مجددا با x-train_new تکرار میکنیم تا 20 فیچر برتر بعدی‌را پیدا کنیم.

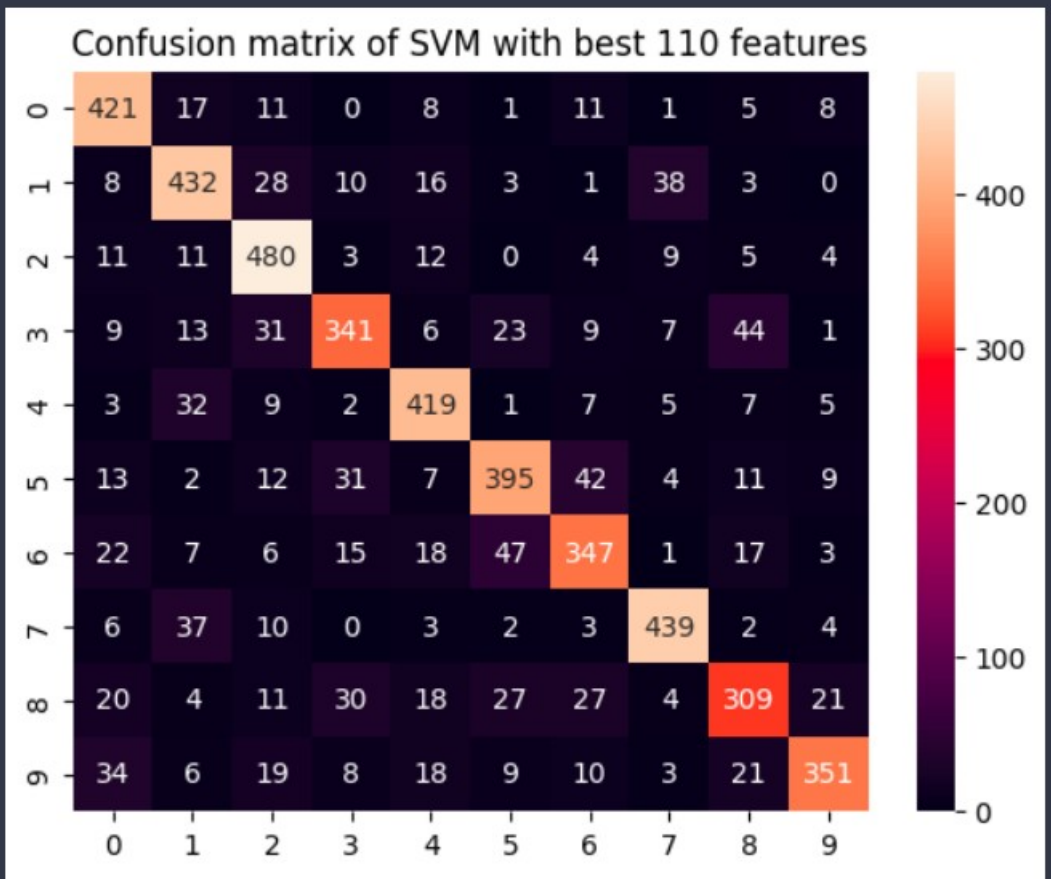
این فیچرهارا به x_train_best اضافه میکنیم و مجددا svmرا آموزش میدهیم.

این حلقه‌را 15 بار تکرار میکنیم. در نهایت 15 مقدار score داریم. که از 0.37 شروع میشود و روند صعودی دارد تا به مقدار 0.70 میرسد. و پس از آن کاهش می یابد.

10 features: accuracy = 0.3764, score = 0.3689
20 features: accuracy = 0.4466, score = 0.4316
30 features: accuracy = 0.5160, score = 0.4935
40 features: accuracy = 0.6042, score = 0.5742
50 features: accuracy = 0.6730, score = 0.6355
60 features: accuracy = 0.6906, score = 0.6456
70 features: accuracy = 0.7166, score = 0.6641
80 features: accuracy = 0.7264, score = 0.6664
90 features: accuracy = 0.7566, score = 0.6891
100 features: accuracy = 0.7702, score = 0.6952
110 features: accuracy = 0.7868, score = 0.7043
120 features: accuracy = 0.7892, score = 0.6992
130 features: accuracy = 0.7948, score = 0.6973
140 features: accuracy = 0.8022, score = 0.6972
150 features: accuracy = 0.8092, score = 0.6967

بنابرین بهترین score برابر 0.70 است که با 110 فیچر بدست می آید.

```
accuracy = 0.7868
```



برای بهبود score میتوانیم از متد feature_selection پیچیده تری استفاده کنیم.