Effects of Regularization on Equus Classification

Miguel Corona

April 2020

1 Introduction

Overfitting is the phenomena in which a model fails to generalize the data and only performs well on the training set. The model's capacity or memorization ability is tied to its number of trainable parameters. A compromise in complexity must be met in the model's hyperparameters to ensure that the model is capable of learning and not memorizing data. Complex models, in terms of trainable parameters, can yield detrimental results on unseen samples unless constraints are applied. Regularization is the practice of mitigating overfitting by placing constraints on the model. The Equus model was updated to include regularization to observe any changes in its performance.

2 Regularization

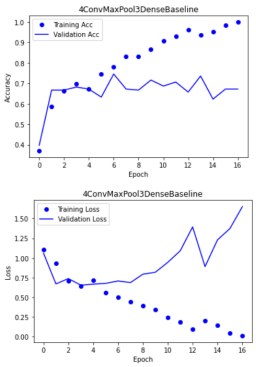
Regularization can prevent complex models from overfitting the training data. Applying a regularization technique on the parameters allows for models with the same hyperparameters to yield different training parameters that may result in different outputs. L1 and L2 regularization apply a cost dependent on the training parameters and aim to set the parameters to values that yield less volatile outputs. The training parameters can be perturbed by adjustments to the architecture by adding artificial noise to a layer's output via dropout. Batch normalization is a change to a layer's input that can influence the training parameters by adjusting the distribution of the data prior to a layer consuming data.

3 Experiment

The objective of the experiment was to observe the effects of regularization on the Equus classification model. The Equus classification model with the best performance is given by the architecture:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 254, 254, 12)      336
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 127, 127, 12)      0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 125, 125, 12)      1308
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 62, 62, 12)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 60, 60, 32)        3488
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 28, 28, 32)        9248
_________________________________________________________________
flatten_2 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_6 (Dense)              (None, 24)                602136
_________________________________________________________________
dense_7 (Dense)              (None, 12)                300
_________________________________________________________________
dense_8 (Dense)              (None, 3)                 39
=================================================================
Total params: 616,855
Trainable params: 616,855
Non-trainable params: 0
```

The model was trained on the development set without data augmentation to yield the benchmark results:



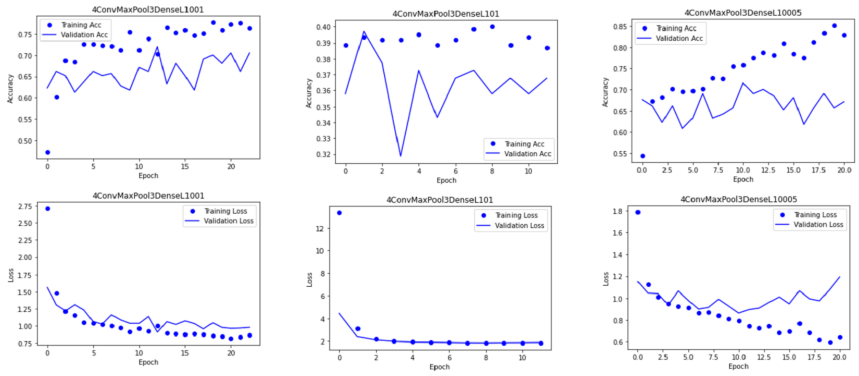
Dataset	Loss	Accuracy
Training	0.5786	0.7197
Validation	0.6281	0.7010
Test	0.6417	0.7043

The benchmark execution indicates that the model has statistical power on the data. The model's drawback is that it overfits the data early on in the training as depicted in the loss learning curves. The validation accuracy is consistent throughout the epochs even as the validation loss begins to diverge from the training loss.

3.1 L1 Regularization

Lasso regression, or L1 regularization, is the practice of placing a penalty proportional to the absolute value of each parameter onto the loss function. The loss contribution by the weights is controlled by the regularization parameter and this parameters was adjusted for each evaluation. The objective was to acquire a simpler model in terms of the magnitude of each layer's weights.

The following regularization parameters were identified and executed: 0.001, 0.01, 0.0005. A constant regularization parameter was set for each layer that utilized L1 regularization. The learning curves and corresponding accuracy and loss metrics are depicted such that the column learning curves correspond to the dataset rows e.g. first column corresponds to the first three rows:



Dataset	Reg Param	Loss	Accuracy
Training	0.001	0.8960	0.7295
Validation	0.001	0.9814	0.6912
Test	0.001	1.0212	0.6957
Training	0.01	1.8396	0.4197
Validation	0.01	1.8493	0.3676
Test	0.01	1.8568	0.3261
Training	0.0005	0.7356	0.8230
Validation	0.0005	0.8275	0.7255
Test	0.0005	0.8855	0.7130

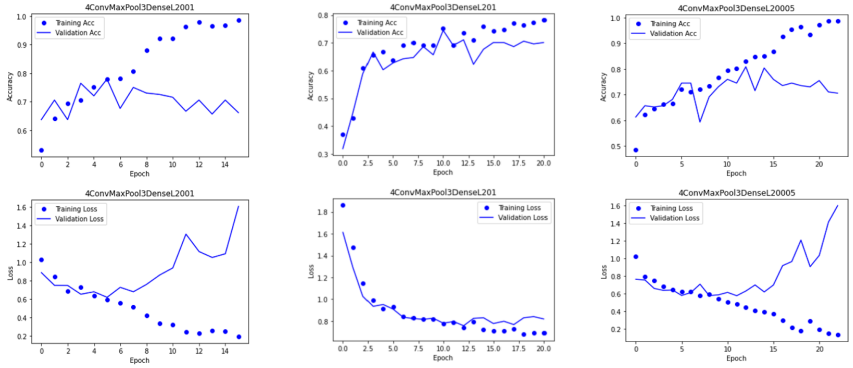
The results indicate that lasso regression can effectively prevent the validation loss from diverging from the training loss with a suitable regularization parameter. A regularization parameter of 0.01 resulted in the minimal divergence between the training and validation losses, but the model's performance was adversely affected based on the evaluations on the validation and test sets. The model no longer held the statistical power it had prior to regularization. The 0.0005 regularization parameter did not yield a learning curve with the validation loss aligned with the training loss, but the curves indicate that a bound was placed on the divergence. The loss was greater with the 0.0005 regularization parameter than in the benchmark, but the accuracy on the test set and validation set improved. The 0.001 regularization parameter yields the most promising learning curves as validation loss is bound and aligned with the training loss. Evaluations on the model with the 0.001 regularization parameter yield a greater loss on the datasets and the classification accuracy is similar to the those of the benchmark.

The L1 regularization did not outperform the benchmark model in terms of loss across the datasets even though minimal classification accuracy improvements were observed. The results indicate that L1 regularization may not be suitable due to the model's simplicity. Regularization parameters with values lower than 0.0005 may be worth further investigation prior to ruling out L1 regularization as these yield a low loss and improved classification accuracy.

3.2 L2 Regularization

Ridge regression, or L2 regularization, adds a cost proportional to the square of each trainable parameter onto the loss function. The loss contribution due to the square of each weight's magnitude is controlled via the regularization parameter.

The following regularization parameters were identified and executed: 0.001, 0.01, 0.0005. A constant regularization parameter was set for each layer that utilized L2 regularization. The learning curves and corresponding accuracy and loss metrics are depicted:



Dataset	Reg Param	Loss	Accuracy
Training	0.001	0.5159	0.8197
Validation	0.001	0.6519	0.7549
Test	0.001	0.6123	0.7652
Training	0.01	0.7043	0.7689
Validation	0.01	0.7766	0.7010
Test	0.01	0.8716	0.6652
Training	0.0005	0.4197	0.8525
Validation	0.0005	0.5930	0.7353
Test	0.0005	0.6147	0.7739

The results of L2 regularization are consistent with those of L1 regularization and indicate that an appropriate regularization parameter can prevent or minimize the divergence between the training and validation loss. The value

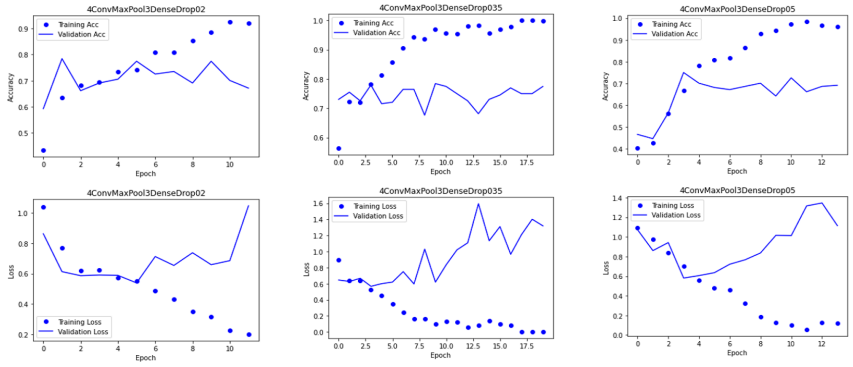
that constitutes as a suitable regularization parameter differs between lasso and ridge based on the executions. The learning curves for regularization parameter 0.01 converge the accuracy and loss of the validation set to those of the training set, but the loss and accuracy on the datasets are not better than those of the benchmark. Improvements in performance were observed with a regularization parameter of 0.001 and 0.0005 albeit with learning curves indicating that divergence occurred between the training and validation sets.

L2 regularization yielded performance improvements in loss and accuracy over the benchmark. The classification accuracy and the loss improved as the regularization parameter decreased indicating that the model may need larger weights to minimize the loss. Regularization parameters with values lower than 0.0005 may be worth further investigation.

3.3 Dropout

Dropout is a regularization technique in which noise is injected into the output of a layer by randomly setting tensor values to zero. The portion of tensor values set to zero are dependent on the dropout rate which traditionally ranges between the 0.2 and 0.5. We observe the behavior of the network with a Dropout layer introduced into the network.

The model was updated to include a Dropout layer prior to entry into the dense network. The Dropout layers were set to include dropout rate: 0.2, 0.35, and 0.5. The learning curves and corresponding accuracy and loss metrics are depicted:



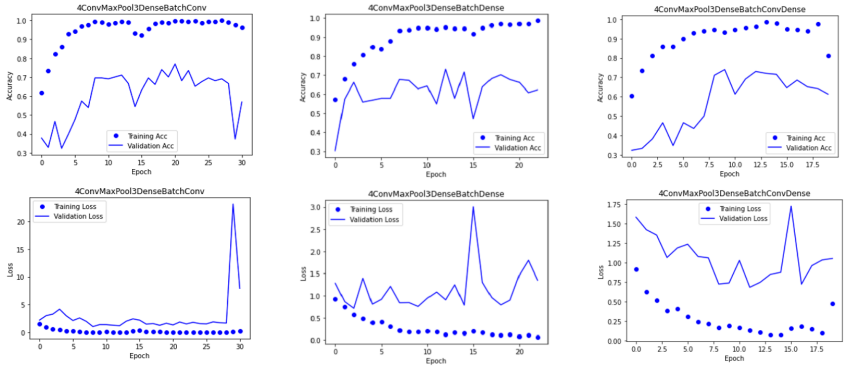
Dataset	Drop Rate	Loss	Accuracy
Training	0.2	0.4410	0.8049
Validation	0.2	0.5589	0.7745
Test	0.2	0.4784	0.7609
Training	0.35	0.4137	0.8443
Validation	0.35	0.5482	0.7843
Test	0.35	0.5321	0.7565
Training	0.5	0.5892	0.7443
Validation	0.5	0.6081	0.7353
Test	0.5	0.5975	0.6870

The results of dropout indicate that the model's performance benefits from the inclusion of Dropout layers. The loss and accuracy improvements were present across the training, validation, and test set for each dropout rate. The model benefited from the noise introduced in each epoch from the dropout, but the noise failed to mitigate the divergence in the learning curves. Deviations in the learning curves are not surprising due to the randomized nature of dropouts on each epoch. The dropout rate of 0.35, the midpoint of the range, appears to yield the greatest improvement though it is difficult confirm without additional executions.

3.4 Batch Normalization

Batch normalization is the practice of normalizing data prior to its consumption by another layer. The normalized inputs assist with gradient propagation and alleviates the vanishing gradient problem seen in very deep networks.

The model was modified to include BatchNormalization prior to the next layer consuming the output of the preceding layer. Three configurations were observed: batch normalization only applied after each convolution layer, batch normalization only applied after each dense layer, and batch normalization after each layer. The learning curves and corresponding accuracy and loss metrics are depicted:



Dataset	Layers	Loss	Accuracy
Training	Conv	0.1060	0.9607
Validation	Conv	1.2214	0.6716
Test	Conv	1.2890	0.6957
Training	Dense	0.5334	0.7672
Validation	Dense	0.7144	0.6716
Test	Dense	0.8349	0.5826
Training	Conv + Dense	0.0343	0.9967
Validation	Conv + Dense	0.7189	0.6618
Test	Conv + Dense	0.5232	0.7913

The results of the executions indicate that batch normalization may not be ideal for the Equus classification model and this may be due to the shallow nature of the model. Batch normalization is intended for deep networks to prevent a vanishing gradient and the model under observation has seven layers prior to the introduction of batch normalization. Any benefits to the metrics occurred in configurations in which batch normalization was applied in the dense network. Applying batch normalization after a convolution layer yielded a poor performance. The convolution layers do not appear to benefit from the batch normalization and this may be due to the presence of max-pooling or an incorrect usage of the max-pooling and batch normalization combination. The learning curves suggest that the deviation between the validation and training loss is mitigated with batch normalization only applied after the convolution layers. The inclusion of batch normalization into the dense layers causes fluctuations in the loss curves.

4 Conclusion

The experiments indicate that the Equus classification model can improve its performance with regularization. Not all regularization techniques may be applicable as it was observed that L1 regularization and batch normalization do not yield performance improvements. Minimal L2 regularization and dropout yield the most benefits compared to the benchmark although this conclusion is deceiving due to the static validation set. A more effective manner to observe the effects of the regularization techniques is to utilize cross-validation. The hyperparameters on the benchmark model were configured based on evaluations via hold out for validation. Cross-validation may reveal that the benchmark model may not be the ideal model and that a new candidate may benefit more from regularization.