BIS 557 Final Project

Weight Regularization in CNN with Sign Language MNIST data

Shiyao Ying

December 2020

1 Introduction

A convolutional neural network(CNN) is a class of deep learning models that typically takes input images and differentiate the images. It is a regularized version of multi-level Perceptrons which fully connects all the neurons. CNN reduces the images into a simpler form for processing and it usually consists of an input layer, an output layer and hidden layers. The hidden layers generally includes convolutional layers which convolve the features, the pooling layers that reduce the spatial sizes of the features and fully connected layers which perform classification tasks[4].

Overfitting occurs when the bias is minimized by the model parameters but the variance would be very large. Ridge and Lasso regression are common techniques used in reducing overfitting in regression models. The linear combination of residual sum of squares and l_2 norm is minimized in Ridge regression and it is useful to reduce the problem of multicolinearity. In Lasso (least absolute shrinkage and selection operator), the l_1 norm is used instead and it improves prediction accuracy and interpretability of models. The elastic net method includes both ridge and lasso regression by having

$$\hat{\beta} = \text{argmin}_{\beta}(||y - \mathbf{X}\beta||^2 + \lambda_2||\beta||^2 + \lambda_1||\beta||_1)$$

In deep learning models, overfitting occurs when a model fits the training dataset with all details and noises. For the regression equation $y = WX + b$ in the neural network, x is the input into a layer, W refers to the weight matrix and b refers to the bias term. In keras, there are three types of regularizers: kernel, bias and activity, that could be applied to Dense, Conv1D, Conv2D and Conv3D layers. A kernel regularizer applies penalty on the layer's kernel which is W. A bias regularizer applies penalty on the layer's bias and an activity regularizer penalizes the output. The regularizers could be l_1, l_2, and l_1l_2 constraints and in this study, we would use these three types of weight decay on three layer weight regularizers on sign language MNIST dataset to see the effects of regularizers.

2 Methods and Data

2.1 data

Sign Language MNIST data[5] is a drop-in replacement of the original MNIST image. Sign language letters are as shown in Figure 1. Different gestures represent the letters and building deep learning models on such dataset could help hard of hearing people by recognizing the sign language gestures.

The data has 26 labels for 26 alphabetic letters and 28*28 pixels on 0-255 grayscale values for data points. As shown in Figure 2, rescaled images in gray scale also contains great amount of information. The training data has 27455 entries and the test data consists of 7172 entries. The dataset is in csv format with a column of label and 784 columns of pixel values. The training and test data are converted to rdata format and included in the bis557 package. They are converted to tensor with shape (number of images)*28*28*1 and the labels are converted to categorical values before fitting the models.

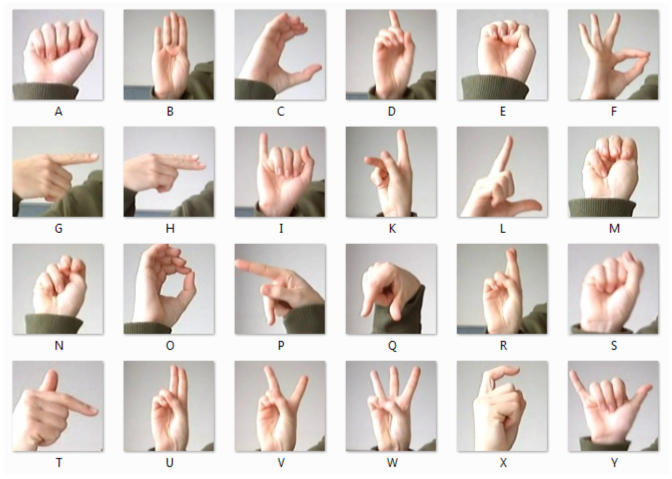

Figure 1: American Sign Language Letters[5]

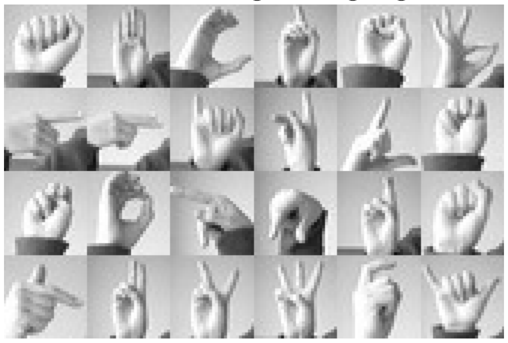

Figure 2: American Sign Language Letters in grayscale and rescaled to 28*28[5]

2.2 Methods

After running several simple CNN models, the general model without weight penalization is selected. As shown in Fig. 3, with 5 epoches, the model converges and has a validation accuracy very close to 1. The input goes through a 3*3 conv2D layer with 32 filters and then goes to a max pooling (2*2) layer.Then it goes the same sets of conv2D and max pooling layers and flatten out and performs classification with fully connected layers of 512 neurons in reLU and 26 (number of classes) in softmax. We use a batch size of 32 and 5 epoches for the model. An internal cross-validation of 20 percent data is used and the loss is computed with *loss categorical crossentropy* and optimizer of *adam*. The model could be further improved by increasing the epoches and adding more layers.

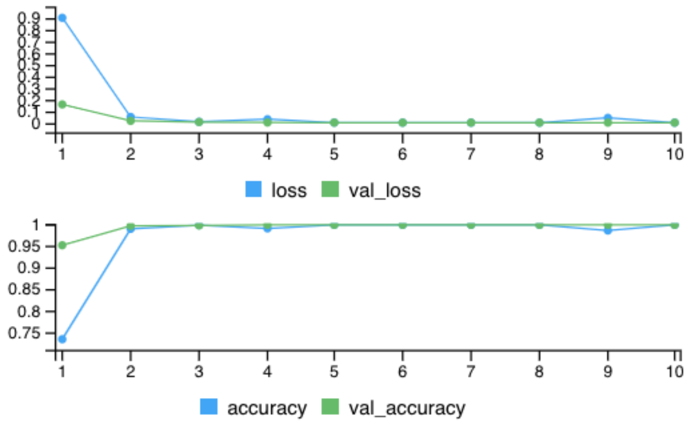

Figure 3: Model fitting process with 10 epoches[5]

To compare l_1,l_2 and the elastic net, a neural network is fitted as above and the accuracy on test is computed. For each weight decay methods, we fit the $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$ *and* 10^{-1} on kernel, bias and activity regularizers and compute their classification accuracy based on test data by tables and plots.

3 Results

For three regularizers in l1 and l2, there were 5 models built with different penalization values and the accuracy and loss were evaluated on predicting the test data labels. The values were recorded and plotted to compare with each other. The code for building models and plotting were included as rmarkdown file.

According figure 4, the models with l_1 bias regularizers constantly had a good prediction accuracy and sometimes it exceeded the accuracy without weight penalization. The l_1 activity regularizers performed decent work with λ values of $10^{-5}, 10^{-4}$ and 10^{-3} and then it came to a sudden decrease on the prediction accuracy to almost zero. The result for l_1 kernel regularizers was about the same, but it did not provide any greater prediction accuracy than the original model. The prediction accuracy from model with penalization of 10^{-3} started to decreases. Correspondingly, the loss for model with l_1 bias regularizers kept constant about 0. The loss for model with l_1 activity and kernel regularizers started to increase from about 10^{-4} and 10^{-3} to about 8 and 3.5.

As shown in Fig.5, the models with l_2 bias regularizers performed prediction well and had about the same prediction accuracy as the original model. The model with l_2 kernel regularizers generally did a decent job on classification on the test data and it decreased

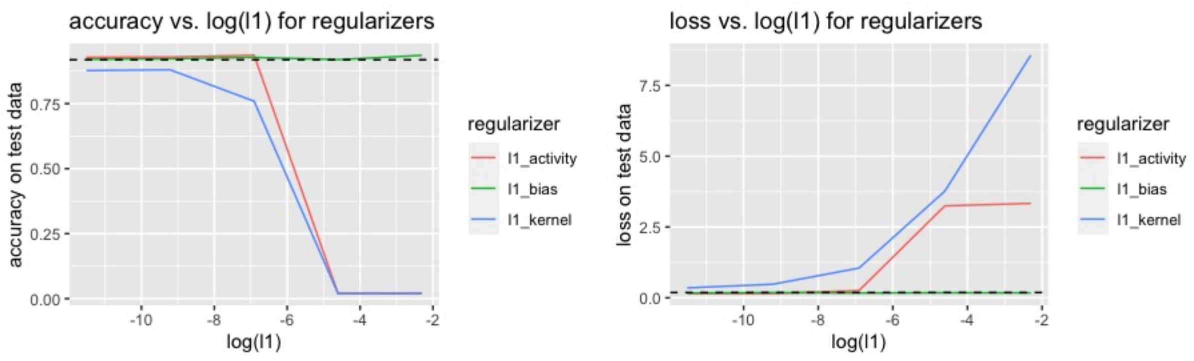

Figure 4: Accuracy and loss on predicting labels on the test data for regularizers with l_1 weight penalization

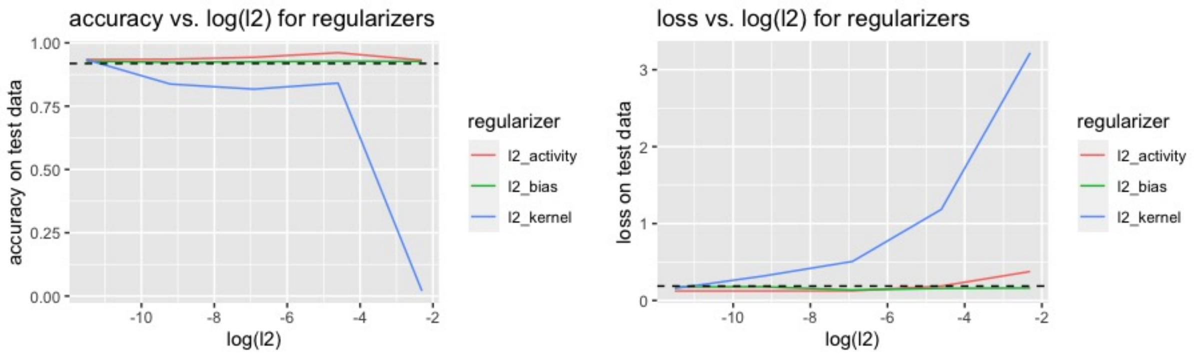

Figure 5: Accuracy and loss on predicting labels on the test data for regularizers with l_2 weight penalization

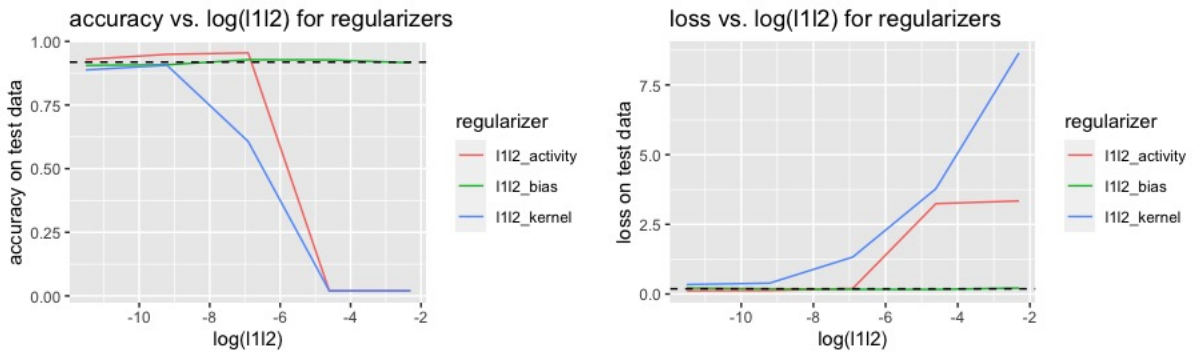

Figure 6: Accuracy and loss on predicting labels on the test data for regularizers with elastic net weight penalization

to about 0 with $\lambda = 0.1$. The overall prediction accuracy on the model with l_2 activity regularizers beat the original model. The plot with loss showed the same trend. Models with l_2 bias regularizers and activity regularizers had about the same loss as the original model while the models with l_2 kernel regularizers increased with $\log(\lambda)$.

The models with same values for λ_1 and λ_2 for elastic net weight penalization showed the additive effects on accuracy and loss (Fig.6). They generally followed the same trends as the models with l_1 regularizers but had slight differences as l_2 also affected the model.

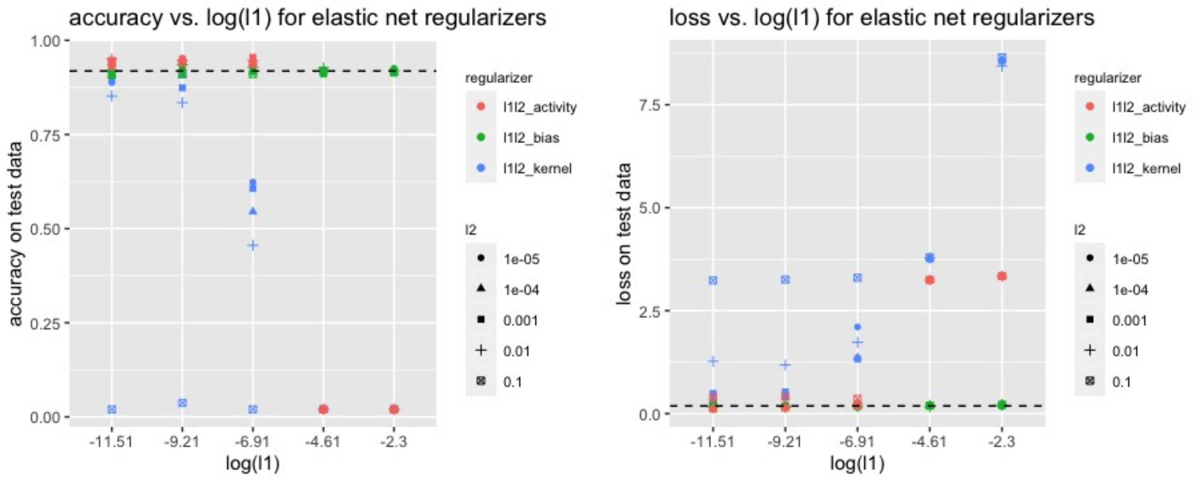

Figure 7: Accuracy and loss on predicting labels on test data for regularizers with elastic net weight penalization with multiple l_2 as factors

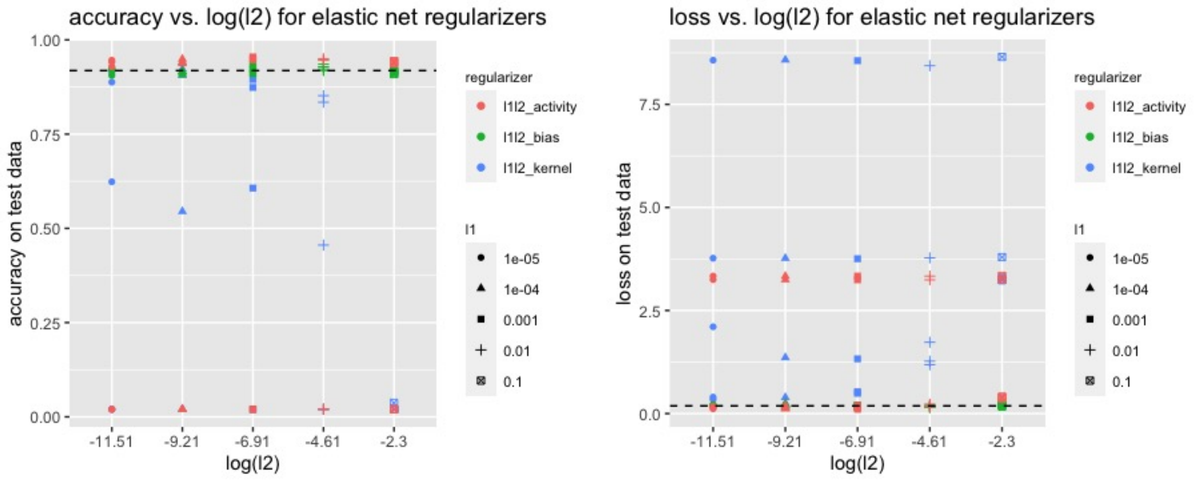

Figure 8: Accuracy and loss on prediction labels on test data for regularizers with elastic net weight penalization with multiple l_1 as factors

The scatter plots, as in figure 7, showed the general trend of accuracy and loss varying with λ_1 and λ_2 in the weight penalization. In addition to the plots shown above with same parameters for both l_1 and l_2, these plots showed the combinations of different values. The $l_1 l_2$ kernel regularizer with $l_1 = 10^{-5}$ and $l_2 = 0.1$ had a very low prediction accuracy. With $l_1 = 10^{-5}, 10^{-4}$ and 10^{-3}, all models with 5 values in l_2 activity regularizers performed better prediction than the original model. We could also see that the loss introduced by adding l_2 values in kernel regularizers have smaller effect than increasing l_1 values. Bias regularizer introduced little variation on the accuracy and loss and they were stable for all the tested values.

Table 1. Accuracy with l_1l_2 kernel regularizer

l_2 values or l_1 values	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-5}	0.8879	0.9228	0.8950	0.8521	0.0201
10^{-4}	0.9131	0.9062	0.8740	0.8346	0.0372
10^{-3}	0.6235	0.5449	0.6069	0.4557	0.0201
10^{-2}	0.0201	0.0201	0.0201	0.0201	0.0201
10^{-1}	0.0201	0.0201	0.0201	0.0201	0.0201

Table 2. Accuracy with l_1l_2 bias regularizer

l_2 values or l_1 values	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-5}	0.9056	0.9402	0.9205	0.9204	0.9091
10^{-4}	0.9228	0.9080	0.9349	0.9354	0.9136
10^{-3}	0.9193	0.9162	0.9282	0.9282	0.9105
10^{-2}	0.9225	0.9148	0.9105	0.9274	0.9184
10^{-1}	0.9260	0.9154	0.9134	0.9186	0.9161

Table 3. Accuracy with l_1l_2 activity regularizer

l_2 values or l_1 values	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-5}	0.9290	0.9373	0.9472	0.9502	0.9441
10^{-4}	0.9462	0.9492	0.9506	0.9459	0.9409
10^{-3}	0.9412	0.9417	0.9548	0.9470	0.9375
10^{-2}	0.0201	0.0201	0.0201	0.0201	0.0201
10^{-1}	0.0201	0.0201	0.0201	0.0201	0.0201

The three tables showed the exact accuracy values for three l_1l_2 regularizers. We could see that the models with kernel regularizer are sensitive to large weight penalization in both l_1 and l_2. Models with bias regularizers did not vary much and there were improvement on the accuracy. Models with activity regularizers improved with small l_1 and l_2 values and worked poorly with large l_1 values.

4 Discussion

When we perform gradient descent with l_2 regularization (ridge), we are penalizing the gradient weight by a factor of $2\lambda_1$. It leads to a reduction of large components in the weights. For l_1 regularization (LASSO), we penalize the weight by sign of it and cause the weight matrix to have zeroes. We will get a sparse weight matrix with large λ_1 values. Generally, the kernel regularizer is used when there is no prior assumption on the model and it would not introduce error if we have a large neural network. For small neural network, penalizing weights can lead to poor results. A bias regularizer is used to have a bias (intercept) close to zero. Using an activity regularizer is to reduce overfitting problem by having a smaller output.

On the sign language MNIST dataset, using a large l_1 regularization, which introduced sparsity to the model, led to a very poor prediction on the kernel and activity regularizers. The model in this study was not large enough to tolerate the sparsity assumption in this case. A small penalization did not affect the performance of the weight much. However, we did get some improvement from l_2 regularization. $\lambda_2 = 0.01$ brought the largest accuracy and $\lambda_2 = 10^{-5}$ brought the smallest loss according to the figures. The network was less sensitive to the change in the penalization values, so a great value up to 0.01 in the kernel regularizer still had a relatively good result. Penalization on bias and output did not affect the model much in this case. The elastic net, $l_1 l_2$ regularizer showed the same interpretation as above. The kernel regularizers were sensitive to both large λ_1 and λ_2 values. The activity regularizer was more sensitive than the bias regularizer and a large λ_1 constraint could lead to a poor result.

The project was proposed to run on CNN of VGG 16 or Resnet 18 which was not needed for this sign language MNIST dataset since a simple network could classify sign language letters. The model was also chosen for time reason since there were total of 105 models run with 5 epoches. Each epoch took about 20 seconds to run. The model could be improved with more epoches and having more layers. We could also add drop out layer and tested the effect of it in CNN. The simple convolutional neural network in this study did not tolerate the sparsity on the weight penalization on $\mathbf{W}$. The effect of the kernel regularizer could be tested on more complicated and larger architectures. The common structures in CNN [3] includes classic ones: LeNet-5, AlexNet, VGG 16, and modern ones: Inception, ResNet, ResNeXt, DenseNet. Further study could be done on larger dataset such as dog breed classification[1], bird species[2] classification with these neural networks.

5 Conclusion

With convolutional neural network fitted on the Sign Language MNIST data, the effect of l_1, l_2 and $l_1 l_2$ types of weight decay on kernel, bias and activity regularizers were compared. The kernel regularizer was the most sensitive to large λ values in this relatively simple neural network. The sparsity brought by l_1 penalization made the prediction accuracy poor and the effect could be further explored by having a larger network. l_2 penalization made some improvement on the prediction accuracy on the test data. The elastic net regularization could be better with improved l_1 regularization on other models.

6 Note

The code for building the models and making plots could be found in model.rmd and plot.r in the same folder. The data has been converted to rdata in the package bis557. The csv file is also uploaded.

References

[1] B. Y. Aditya Khosla, Nityananda Jayadevaprakash and L. Fei-Fei. Novel dataset for fine-grained image categorization. first workshop on fine-grained visual categorization (fgvc), ieee conference on computer vision and pattern recognition (cvpr), 2011. URL `http://vision.stanford.edu/aditya86/ImageNetDogs/`.

[2] Gerry. 240 bird species - 33448 train, 1200 test, 1200 validation images 224x224x3 jpg format. URL `https://www.kaggle.com/gpiosenka/100-bird-species`.

[3] J. JORDAN. Common architectures in convolutional neural networks. URL `https://www.jeremyjordan.me/convnet-architectures/`.

[4] S. Saha. A comprehensive guide to convolutional neural networks — the eli5 way. URL `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`.

[5] tecperson. Sign language mnist - drop-in replacement for mnist for hand gesture recognition tasks. URL `https://www.kaggle.com/datamunge/sign-language-mnist`.