UNIVERSITÄT DES SAARLANDES

Saarland University, Department of Computer Science
Neural Network Assignment 7

Deborah Dormah Kanubala (7025906) , Irem Begüm Gündüz (7026821), Anh Tuan Tran (7015463)

January 4, 2023

Exercise 7.1 - Universal Approximation Theorem

Exercise 7.1.a

Investigation of approximating Multilayer Neural Networks has been an area of interest to many researchers and a subsequent amount of work has gone into this. The main contribution of Kurt Hornik's paper is to provide proof that a feedforward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function to any desired degree of accuracy, provided that the activation function used by the network is bounded and constant. Also, Hornik shows that a standard multilayer feedforward network with an activation function (which is continuous, bounded, and nonconstant) can approximate any continuous function provided there are sufficiently many hidden units available. As such feedforward neural networks with a single hidden layer are capable of representing a wide range of complex tasks.

Exercise 7.1.b

In this paper, G. Cybenko proves that a feedforward neural network with a single internal layer and an arbitrary continuous sigmoidal function can approximate continuous functions to any desired degree of accuracy providing that no constraints are placed on the number of nodes or the size of the weights. The result in this paper is weaker because it only applies to feedforward networks with a single hidden layer and a continuous sigmoidal nonlinearity activation function.

Exercise 7.1.c

Deep networks can learn more abstract and more in-depth representations of the data which can be used for downstream machine learning tasks. The useful learned data representation can also be useful for training models with better generalizations to unseen data. Also, deeper networks increase the complexity or capacity of the model which can be useful when training a model with data with high variability.

Exercise 7.2 - L1 and L2 Regularization

Exercise 7.2.a

Considering a simple model such as $y = W^T X + b$. The weights specify how the output y interacts with the inputs x but the bias b only controls the output y. This, therefore, means that one will require fewer data samples to be able to fit b than the weights. This means that we do not induce too much variance by leaving the biases unregularized.

Exercise 7.2.b

L_1 regularization would be preferable when we want to perform feature selection. It is useful in situations where the number of features is large and not all of them are relevant to the prediction task. As L_1 tends to shrink coefficients to 0 and results to produce a more sparse model. However, this makes the L_1 regularization technique more prone to overfitting.
On the other hand, L_2 adds a penalty term that is proportional to the square of the parameters resulting in a model with small non-zero parameters. L_2 will be preferable if we want to prevent overfitting in our model. Also, L_2 is more easily differentiable hence making this regularization technique easier to optimize. In summary, deciding on which regularization technique to use will depend on the ML task and what goal we want to achieve.

Exercise 7.2.c

Yes, the elastic net is a regularization method that uses both L_1 and L_2. Elastic net regularization will be beneficial when one wants to balance the benefits of the two regularization methods (L_1 and L_2). While L_1 is useful in performing feature selection and L_2 is useful in preventing overfitting, combining these two will help achieve the benefit of these two. The objective function of elastic net regularization is given by:

$$\text{Objective function} = lossfunction + \lambda\left(\alpha \sum \mid W \mid + (1-\alpha) \sum W^2\right)$$

NB: Where α is the mixing parameter that controls the balance between L_1 and L_2 .

Exercise 7.2.d

yes, this is true. The objective of regularization is to constrain the model's weight by adding a penalty term to the objective function which we are interested in optimizing during training time. The penalty term will serve to minimize the model's complexity, reducing overfitting and improving regularization.

Exercise 7.2.e

The L_1 and L_2 regularization for a single feature is given as:

$$L_1 = \frac{1}{2m} \sum (y_1 - \hat{y_1})^2 + \lambda \mid\mid w_1 \mid\mid$$

$$L_2 = \frac{1}{2m} \sum (y_1 - \hat{y_1})^2 + \lambda w_1^2$$

Where $\hat{y_1} = w_1 x_1 + b$ L_1 regularization adds a penalty proportional to the absolute value of the weights, while L_2 regularization adds a penalty proportional to the square of the weights.

In the case of the L_2 regularization, the penalty term λw_1^2 is quadratic in terms of the weight w, which means that the penalty increases as the weight increases. As a result, the optimization process will try to minimize the weights while still minimizing the error between the predicted and actual values. Therefore, the weights are unlikely to be set to zero, as this would significantly increase the error.

On the other hand, in the case of L_1 regularized least squares, the penalty term is linear in weights rather than quadratic like in the L_2. As a result, the optimization process will try to minimize the weights by setting some of them to zero, rather than just reducing their values.

Exercise 7.3 - Regularization Methods

Exercise 7.3.a

$$\begin{aligned} w_{i+1} &= w_i - \eta \nabla_w \left(J(w) + \frac{\lambda}{2} w^t \cdot w \right) \\ &= w_i - \eta \nabla_w J(w) - \eta \nabla_w \frac{\lambda}{2} \left(w^t \cdot \omega \right) \\ &= w_i - \eta \nabla_w J(w) - \eta \nabla_\omega \lambda \cdot \|\omega\| \end{aligned}$$

which is equivalent to L_2 regularization.

$$\left(1 - \eta \nabla_\omega \lambda\right) \omega_i - \eta \nabla_\omega J(\omega)$$

Exercise 7.3.b

$$\begin{aligned} \hat{J}(\boldsymbol{\theta}) = \; & J\left(\boldsymbol{w}^*\right) + \tfrac{1}{2} \left(\boldsymbol{w} - \boldsymbol{w}^*\right)^\top \boldsymbol{H} \left(\boldsymbol{w} - \boldsymbol{w}^*\right) \\ & \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H} \left(\boldsymbol{w} - \boldsymbol{w}^*\right) \\ & \boldsymbol{w}^{(\tau)} = \boldsymbol{w}^{(\tau-1)} - \epsilon \nabla_{\boldsymbol{w}} \hat{J} \left(\boldsymbol{w}^{(\tau-1)}\right) \\ & \quad = \boldsymbol{w}^{(\tau-1)} - \epsilon \boldsymbol{H} \left(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*\right) \\ & \boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* = \left(\boldsymbol{I} - \epsilon \boldsymbol{H}\right) \left(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*\right) \end{aligned}$$

H matrix can be written as:

$$\begin{aligned} \boldsymbol{H} &= \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \\ \boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* &= \left(\boldsymbol{I} - \epsilon \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top\right) \left(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*\right) \\ \boldsymbol{Q}^\top \left(\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^*\right) &= (\boldsymbol{I} - \epsilon \boldsymbol{\Lambda}) \boldsymbol{Q}^\top \left(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*\right) \end{aligned}$$

The weights after τ iterations are, assuming $w(0) = 0$:

$$\boldsymbol{Q}^\top \boldsymbol{w}^{(\tau)} = \left[\boldsymbol{I} - (\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})^\top\right] \boldsymbol{Q}^\top \boldsymbol{w}^*$$

The optimal weights obtained from L_2 regularization:

$$\begin{aligned} \boldsymbol{Q}^\top \tilde{\boldsymbol{w}} &= (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^* \\ \boldsymbol{Q}^\top \tilde{\boldsymbol{w}} &= \left[\boldsymbol{I} - (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \alpha\right] \boldsymbol{Q}^\top \boldsymbol{w}^* \end{aligned}$$

If we compare the weights after τ iterations on early stopping with weights obtained from L_2 regularization we'll get:

$$(\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})^\tau = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \alpha,$$

If we plug in

$$\tau \approx \frac{1}{\epsilon \alpha}$$

early stopping is equivalent to L_2 regularization.

Exercise 7.3.c

To perform label smoothing on a one-hot target vector, we add a small amount of noise ϵ to the vector and divide this by K the number of classes. The ϵ determines the level of smoothing, and if we set ϵ to 0 we get the one hot encoding and if we set its value to ϵ we get a uniform distribution. As such to modify the one hot target vector, the ϵ should be set to be between 0 and 1. For example, consider a one-hot target vector $[0, 1, 0, 0]$with $\epsilon = 0.1$ and $K = 4$. The modified target vector with label smoothing would be $[\frac{\epsilon}{k}, \frac{\epsilon}{k}, \frac{\epsilon}{k}, \frac{\epsilon}{k}, \frac{\epsilon}{k}] = [0.025, 0.9, 0.025, 0.025]$

Exercise 7.4 - Dropout

Exercise 7.4.a

Dropout refers to dropping arbitrary neurons during the training process. It prevents overfitting of the model because it forces the network to learn redundant representation. Some features may co-adapt to each other, by dropping them in an arbitrary order, dropout prevents co-adaptation. Thus, the model becomes more robust to changes in neurons, as it does not depend on the activation of certain sets of neurons.

Dropout won't be useful when the network has a small number of parameters, to begin with. In that case, the activation of the neurons will be highly correlated to each other, therefore, dropping out the neurons won't be effective. In this case, batch normalization will be an alternative to drop out.

Exercise 7.4.b

Bagging and dropout are similar as both methods average the model. Dropout is used to prevent overfitting in large training models, as it drops random neurons during the training process, which makes the model less sensitive to the weights of certain neurons. Bagging, on the other hand, is used to increase the model accuracy by training the model using sets of different subsets of training data, which reduces the variance therefore the noise in the training dataset.

Exercise 7.4.c

Depending on the dataset, one can choose to apply l2 regularization and dropout together. However, l1 regularization already performs a feature selection thus, using it along with dropout may lead to longer training time and still overfitting. The figure below is taken from Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15, no. 1 (2014): 1929-1958.

Method	**Test Classification error %**
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	**1.05**

Table 9: Comparison of different regularization methods on MNIST.

According to the paper, when L_2 regularization is used along with dropout, the classification error in the MNSIT dataset was lower when only L_2 regularization or $L_2 + L_1$ is applied together on the model.

Exercise 7.4.d

In regular dropout, the test sets activations need to be scaled by dropout rate p because there is no dropout applied on test sets. In inverted dropout, inverse scaling is applied during the training therefore, the test set no longer needs to be processed. It decreases the inference time compared to regular dropout.