Neural Networks Homework 8

Sayeh Jarollahi (7073520, saja00006@stud.uni-saarland.de)
Mahsa Amani (7064006, maam00002@stud.uni-saarland.de)

December 15, 2024

Exercise 8.1

Proof. a) The Universal Approximation Theorem states that a feedforward neural network with at least one hidden layer and a squashing activation function can approximate any Borel measurable function between finite-dimensional spaces. This approximation can achieve any desired accuracy, given enough hidden units and a linear output layer.
b) According to the universal approximation theorem we need a feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function. If we only use linear layers, these layers can be rewrite as one linear function and there is squashing in the process.
c)
i) The first reason is training difficulty. It mentions in the book that for a good performance we may need and exponential number of hidden layers which requires high degree of freedom. Also, the other reason is overfitting to data. The network may fail to learn and generalize correctlyand the training algorithm might choose the wrong function as a result of overfitting
ii) Reduction in number of units is the first reason. using deeper models can reduce the number of units required to represent the desired function. This suggests that deeper networks are often more efficient in representing complex functions with fewer units compared to very large shallow networks. Improved Generalization is another reason. The textbook mentions that using deeper models can reduce the amount of generalization error. So we can see that deeper architectures can improve generalization and help avoid overfitting.

□

Exercise 8.2

Proof. a) Regularization aims to prevent overfitting, which occurs when there are too many neurons predicting the output. By applying regularization, we essentially exclude some neurons. In contrast, bias is not a connection that transmits output from the previous layer; rather, it connects to a single virtual neuron with a constant output of one. Since it is constant, bias does not result from computation and is independent of the input.
b) Model predictions can be calculated using:

$$Xw^* = X(X^\top X + \lambda I_d)^{-1}X^\top y$$

For $X^\top X$ and considering U is a orthogonal matrix, we have:

$$X^\top X = (U\Sigma V^\top)^\top (U\Sigma V^\top) = V\Sigma^\top U^\top U \Sigma V^\top = V\Sigma^\top \Sigma V^\top$$

where $\Sigma^\top\Sigma$ is a diagonal matrix with entries σ_i^2, that are eigenvalues of $X^\top X$.
We start by substituting the SVD of X into the prediction formula using the above equation:

$$Xw^* = X(X^\top X + \lambda I_d)^{-1} X^\top y = X(V\Sigma^\top\Sigma V^\top + \lambda I_d)^{-1} X^\top y = XV(\Sigma^\top\Sigma + \lambda I_d)^{-1} V^\top X^\top y$$

$$Xw^* = U\Sigma V^\top V(\Sigma^\top\Sigma + \lambda I_d)^{-1} V^\top V \Sigma^\top U^\top y$$

As V is an orthogonal matrix, thus $V^\top V = I_d$, and we have:

$$Xw^* = U\Sigma(\Sigma^\top\Sigma + \lambda I_d)^{-1}\Sigma^\top U^\top y$$

We know that $\Sigma^\top\Sigma$ is diagonal matrix with entries σ_i^2, so:

$$Xw^* = U\Sigma \begin{bmatrix} \frac{1}{\sigma_1^2+\lambda} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2^2+\lambda} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma_r^2+\lambda} \end{bmatrix} \Sigma^\top U^\top y$$

$$Xw^* = U \begin{bmatrix} \frac{\sigma_1^2}{\sigma_1^2+\lambda} & 0 & \dots & 0 \\ 0 & \frac{\sigma_2^2}{\sigma_2^2+\lambda} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \frac{\sigma_r^2}{\sigma_r^2+\lambda} \end{bmatrix} U^\top y$$

Finally, we can write:

$$Xw^* = \sum_{i=1}^{r} \mathbf{u}_i \frac{\sigma_i^2}{\sigma_i^2 + \lambda} \mathbf{u}_i^\top y$$

c) The gradient of $\tilde{J}(w)$ is:

$$\nabla_w \tilde{J}(w) = \nabla_w J(w) + \nabla_w \left(\frac{\lambda}{2} w^\top w \right) = \nabla_w J(w) + \lambda w$$

We substitute $\nabla_w \tilde{J}(w)$ into weight update rule:

$$w_{i+1} = w_i - \eta \left(\nabla_w J(w_i) + \lambda w_i \right) = w_i - \eta \nabla_w J(w_i) - \eta\lambda w_i = w_i(1 - \eta\lambda) - \eta \nabla_w J(w_i)$$

The term $w_i(1-\eta\lambda)$, at each step, scales the weights down by a factor of $(1-\eta\lambda)$, which we call **weight decay**. This helps prevent the weights from growing too large and improves generalization.

d) To compare early stopping with L^2 regularization, we examine a simple setting where the only parameters are linear weights ($\theta = \mathbf{w}$). We can model the cost function J with a quadratic approximation in the neighborhood of the empirically optimal value of the weights $\mathbf{w}^*$:

$$\hat{J}(\theta) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\top \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where $\mathbf{H}$ is the Hessian matrix of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$. Given the assumption that $\mathbf{w}^*$ is a minimum of $J(\mathbf{w})$, we know that $\mathbf{H}$ is positive semidefinite.

Under a local Taylor series approximation, the gradient is given by:

$$\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

For simplicity, we set the initial parameter vector to the origin, $\mathbf{w}^{(0)} = \mathbf{0}$. We have:

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \epsilon \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}^{(\tau-1)}) = \mathbf{w}^{(\tau-1)} - \epsilon \mathbf{H}(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$
$$\mathbf{w}^{(\tau)} - \mathbf{w}^* = (\mathbf{I} - \epsilon \mathbf{H})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$

Using the eigendecomposition of $\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, where $\mathbf{\Lambda}$ is a diagonal matrix and $\mathbf{Q}$ is an orthonormal basis of eigenvectors, we can rewrite the above equation:

$$\mathbf{w}^{(\tau)} - \mathbf{w}^* = (\mathbf{I} - \epsilon \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top)(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$
$$\mathbf{Q}^\top(\mathbf{w}^{(\tau)} - \mathbf{w}^*) = (\mathbf{I} - \epsilon \mathbf{\Lambda})\mathbf{Q}^\top(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$

Assuming that $\mathbf{w}^{(0)} = \mathbf{0}$ and that ϵ is chosen to be small enough to guarantee $|1 - \epsilon\lambda_i| < 1$, the parameter trajectory during training after τ parameter updates is as follows:

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

Now, using the L^2 regularization expression $\tilde{\mathbf{w}} = \mathbf{Q}(\mathbf{\Lambda} + \alpha \mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^\top \mathbf{w}^*$, we have:

$$\mathbf{Q}^\top \tilde{\mathbf{w}} = (\mathbf{\Lambda} + \alpha \mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^\top \mathbf{w}^* = [\mathbf{I} - (\mathbf{\Lambda} + \alpha \mathbf{I})^{-1}\alpha]\mathbf{Q}^\top \mathbf{w}^*$$

Comparing the last 2 equations, we see that if the hyperparameters ϵ, α, and τ are chosen such that:

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

then L^2 regularization and early stopping can be seen as equivalent (at least under the quadratic approximation of the objective function).

$\square$