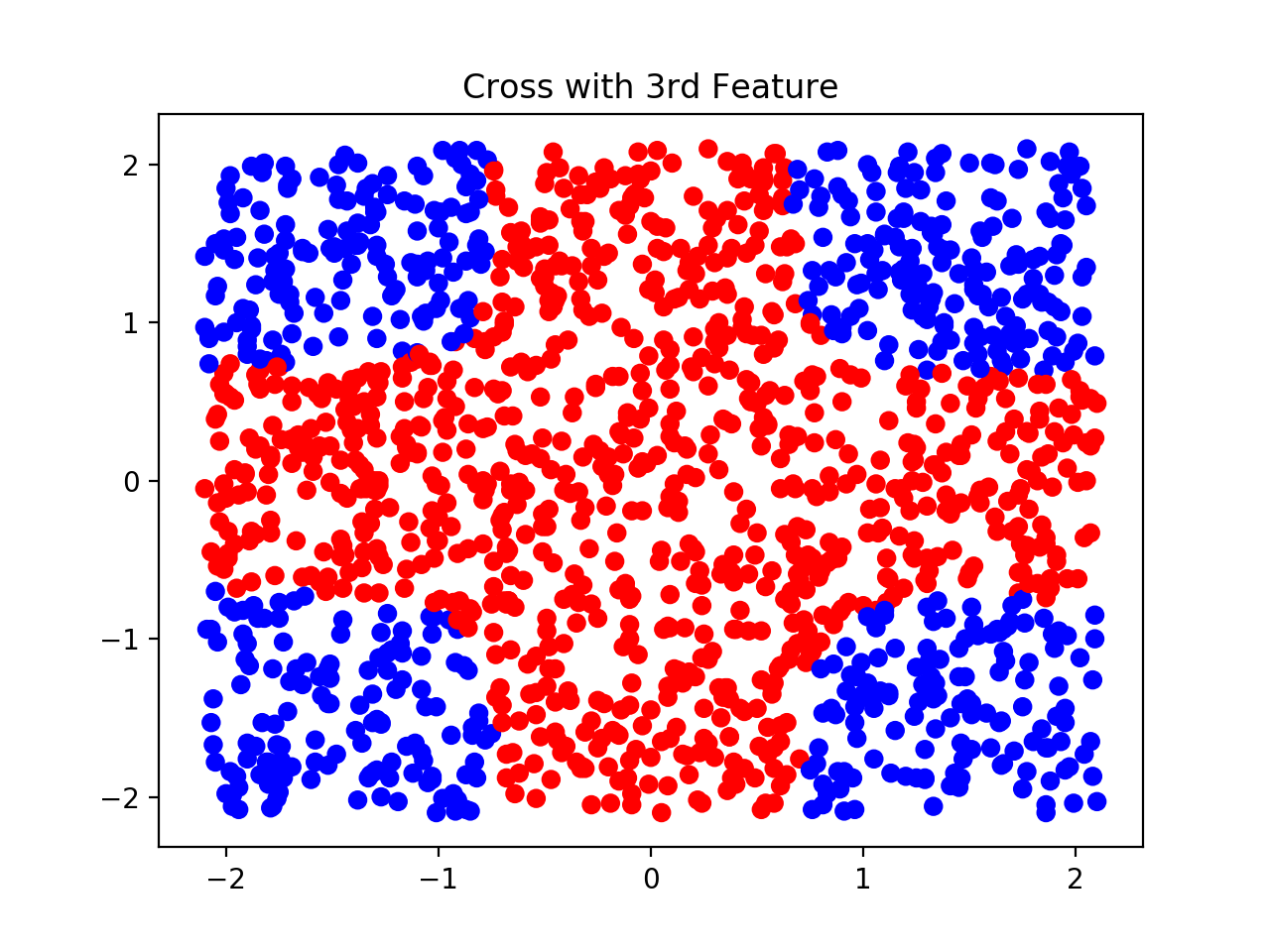
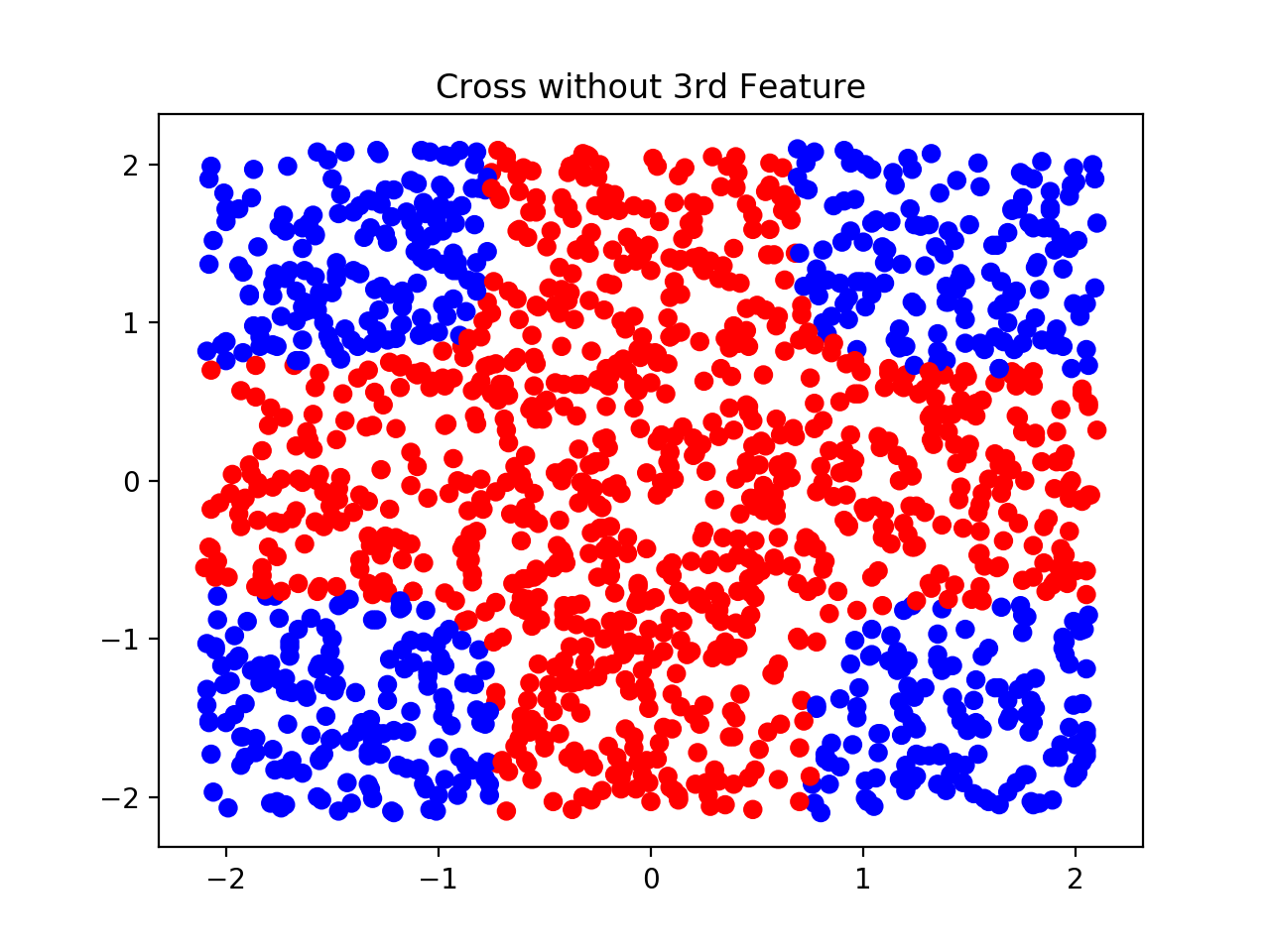
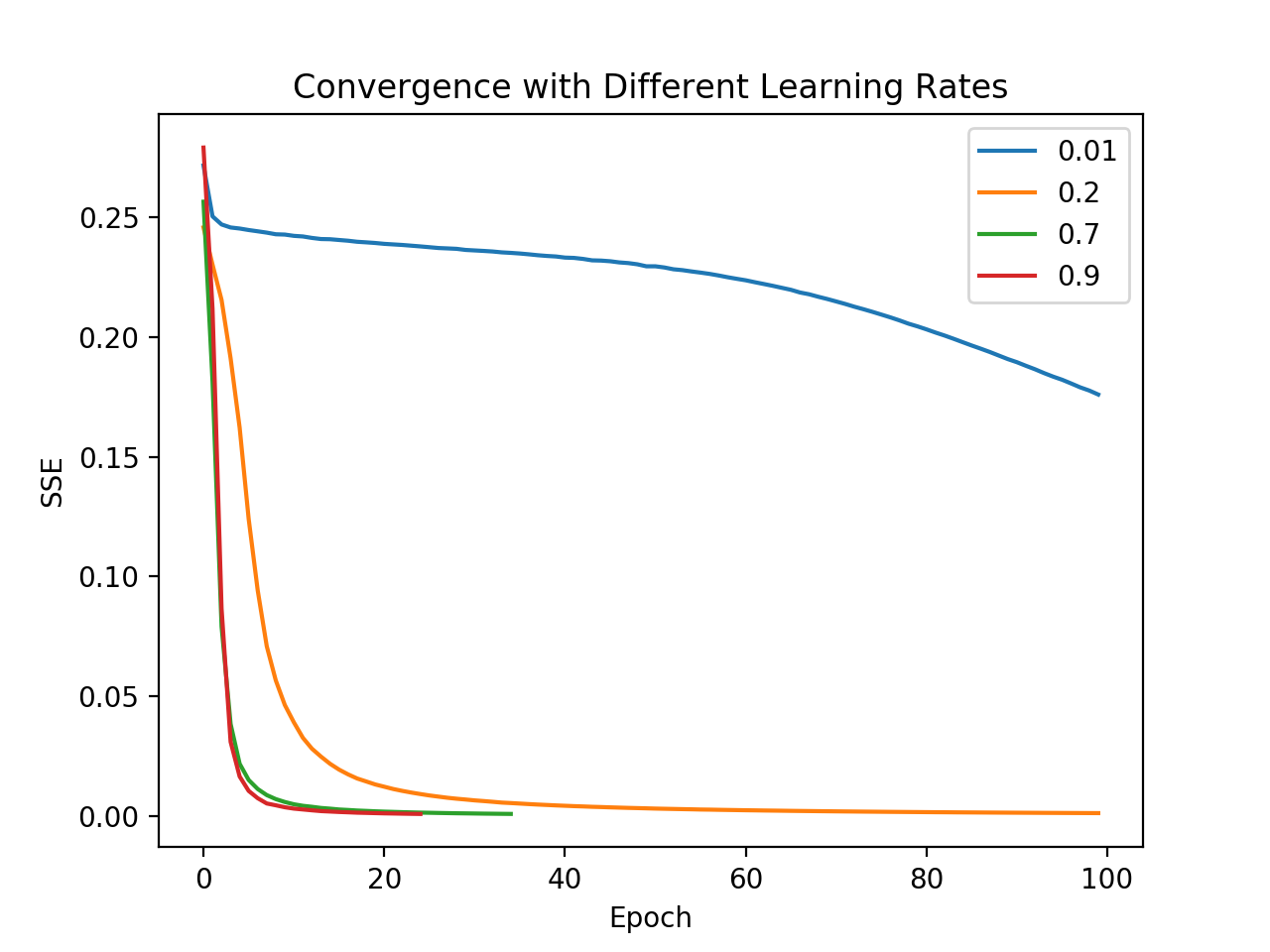
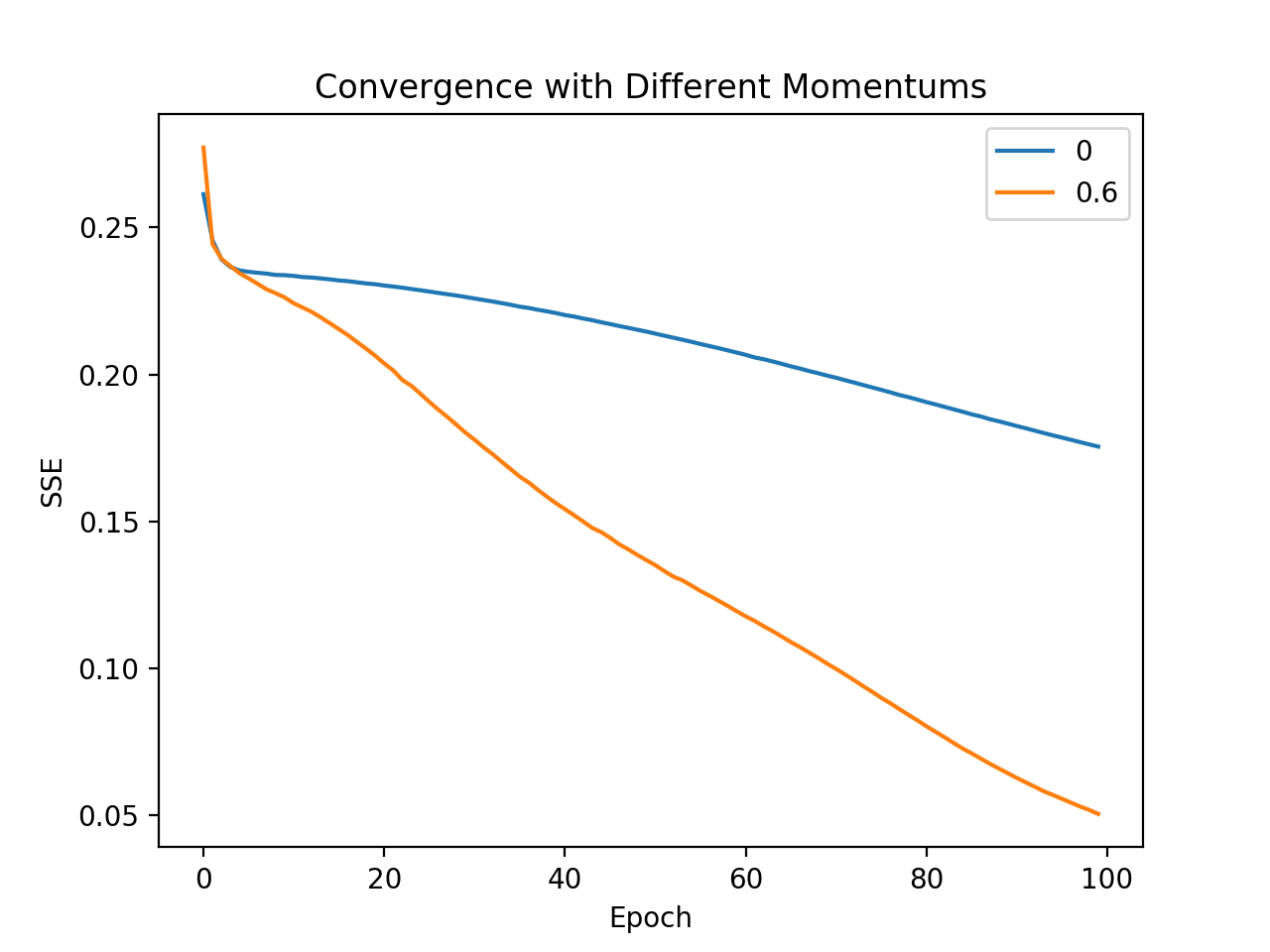
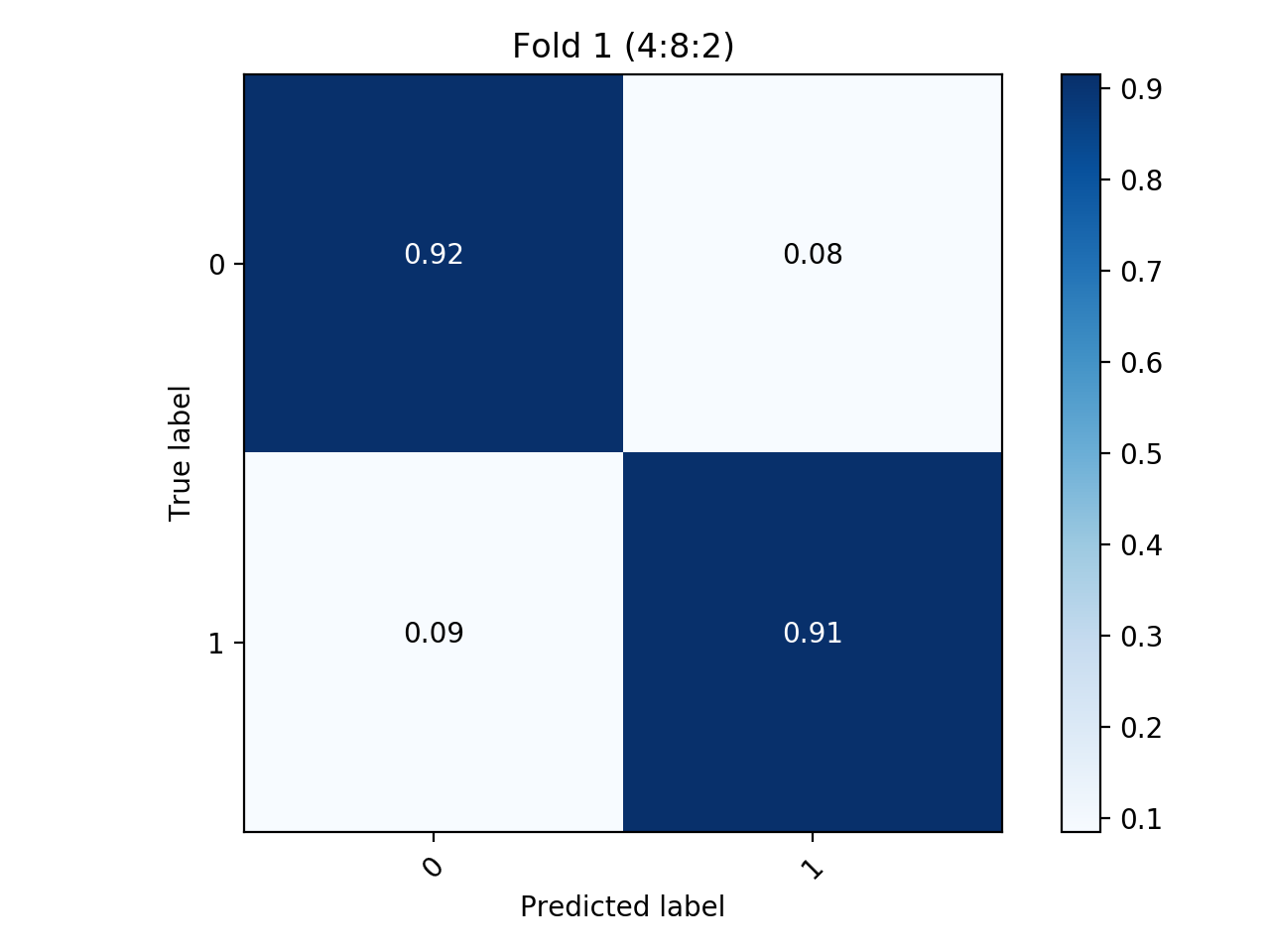
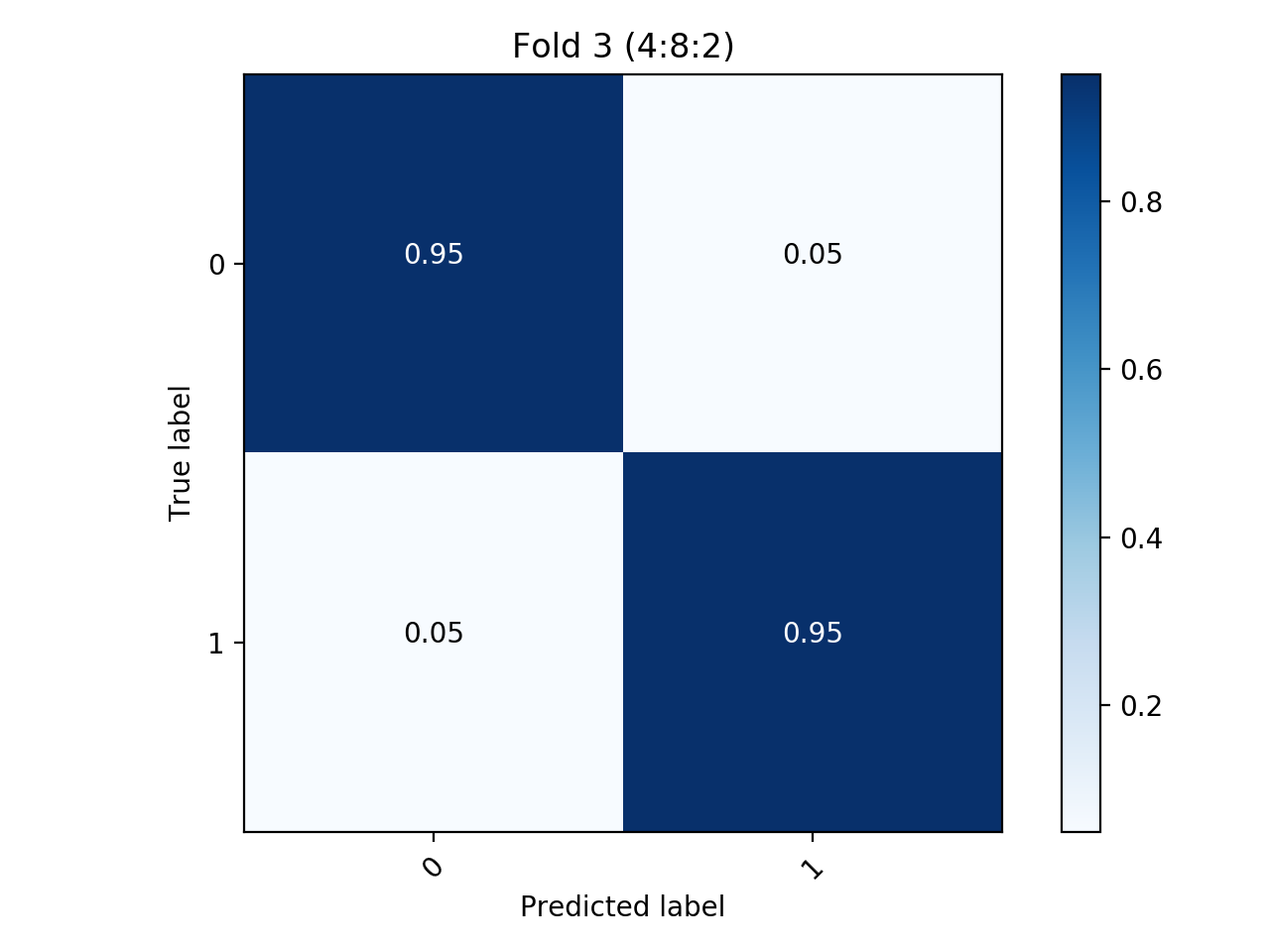
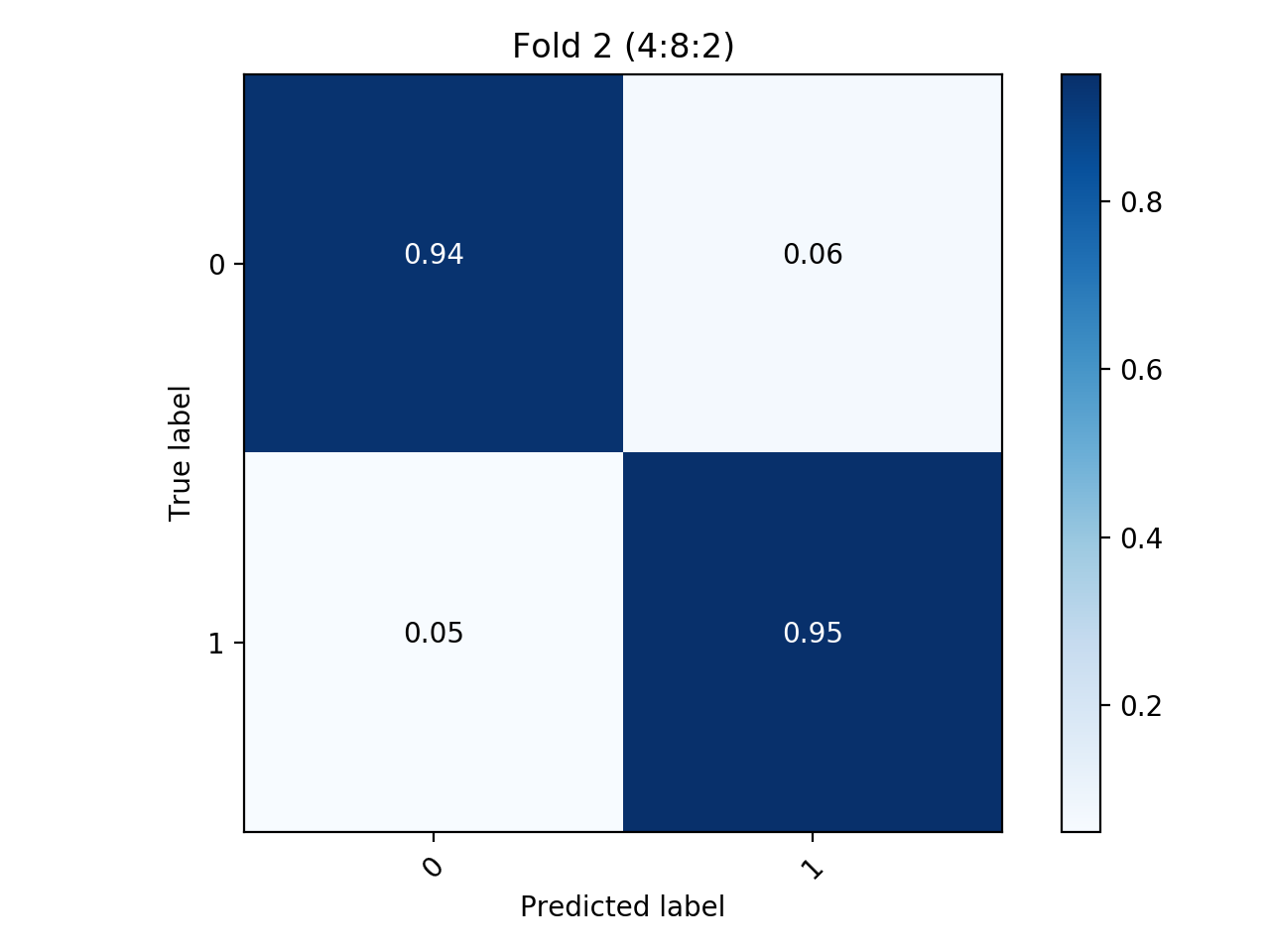
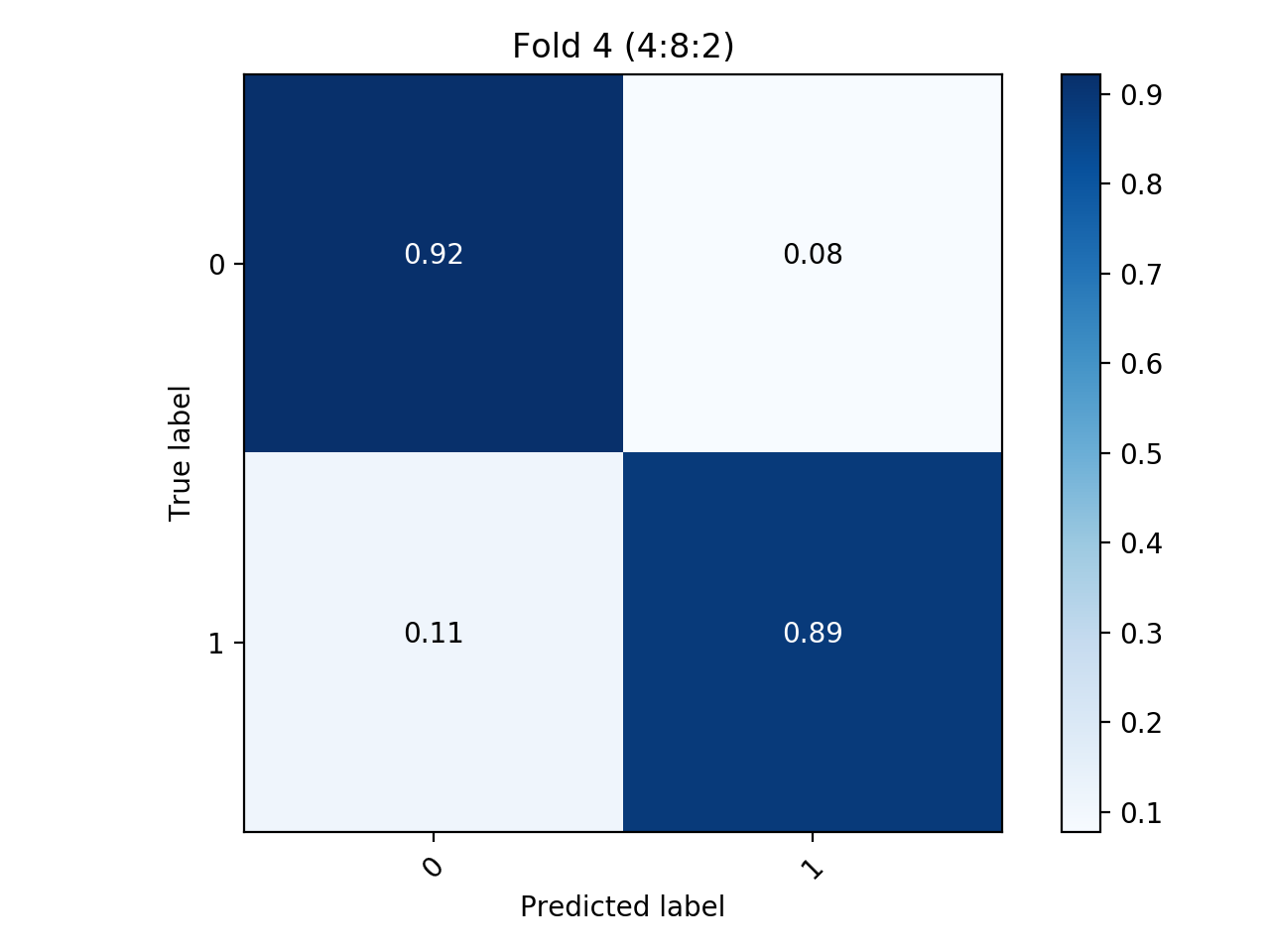
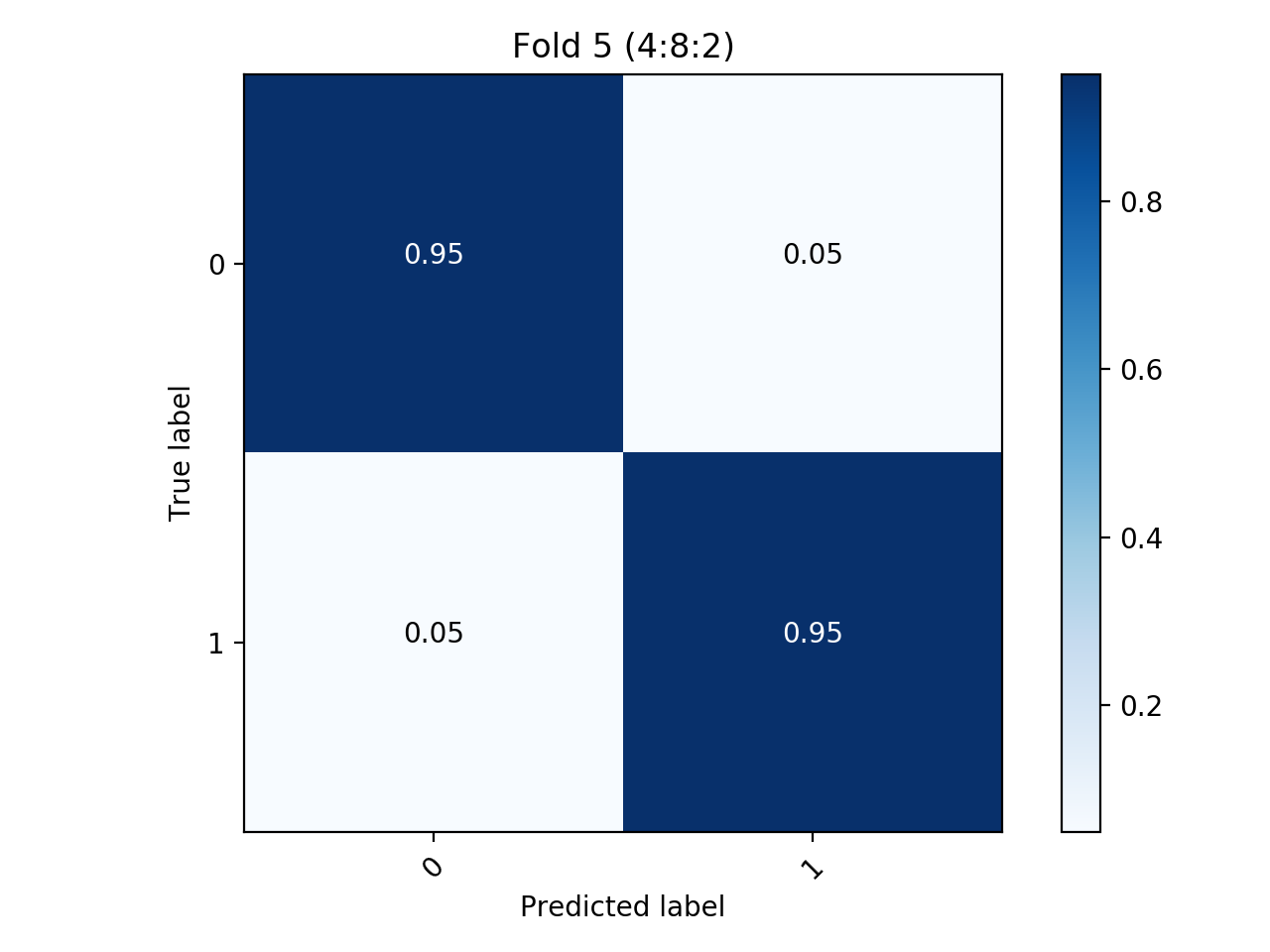
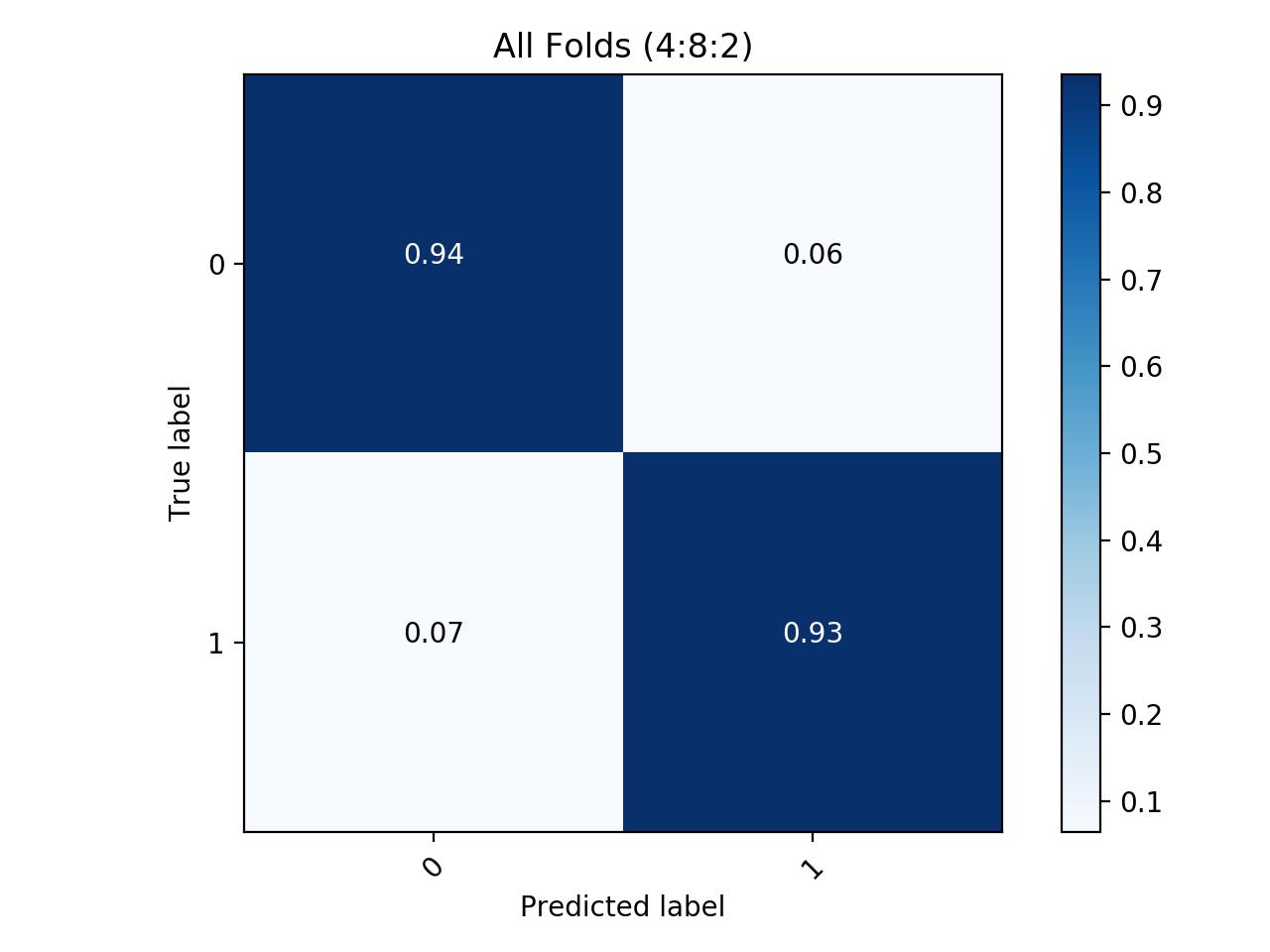
Lab 1 Part B

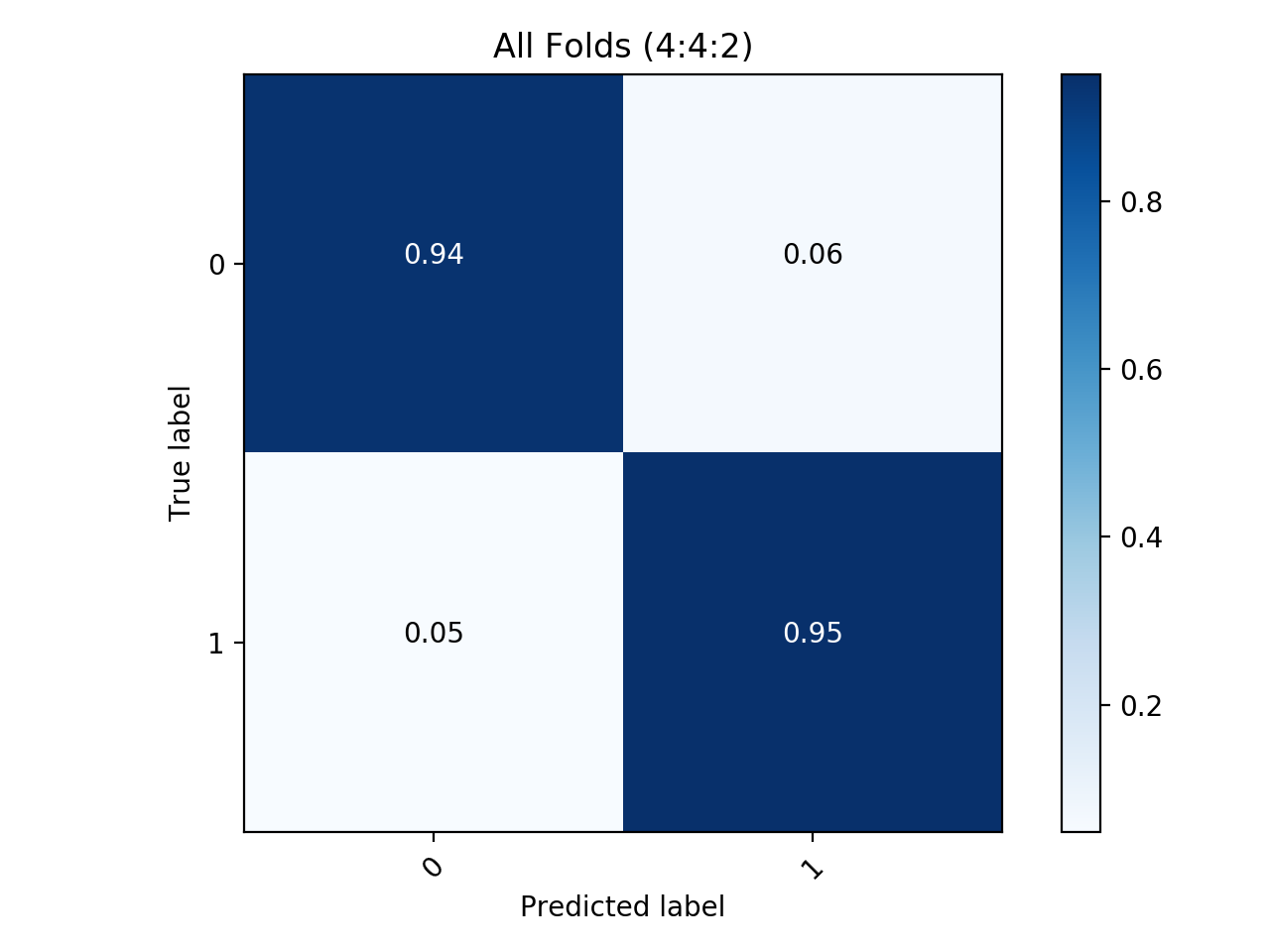
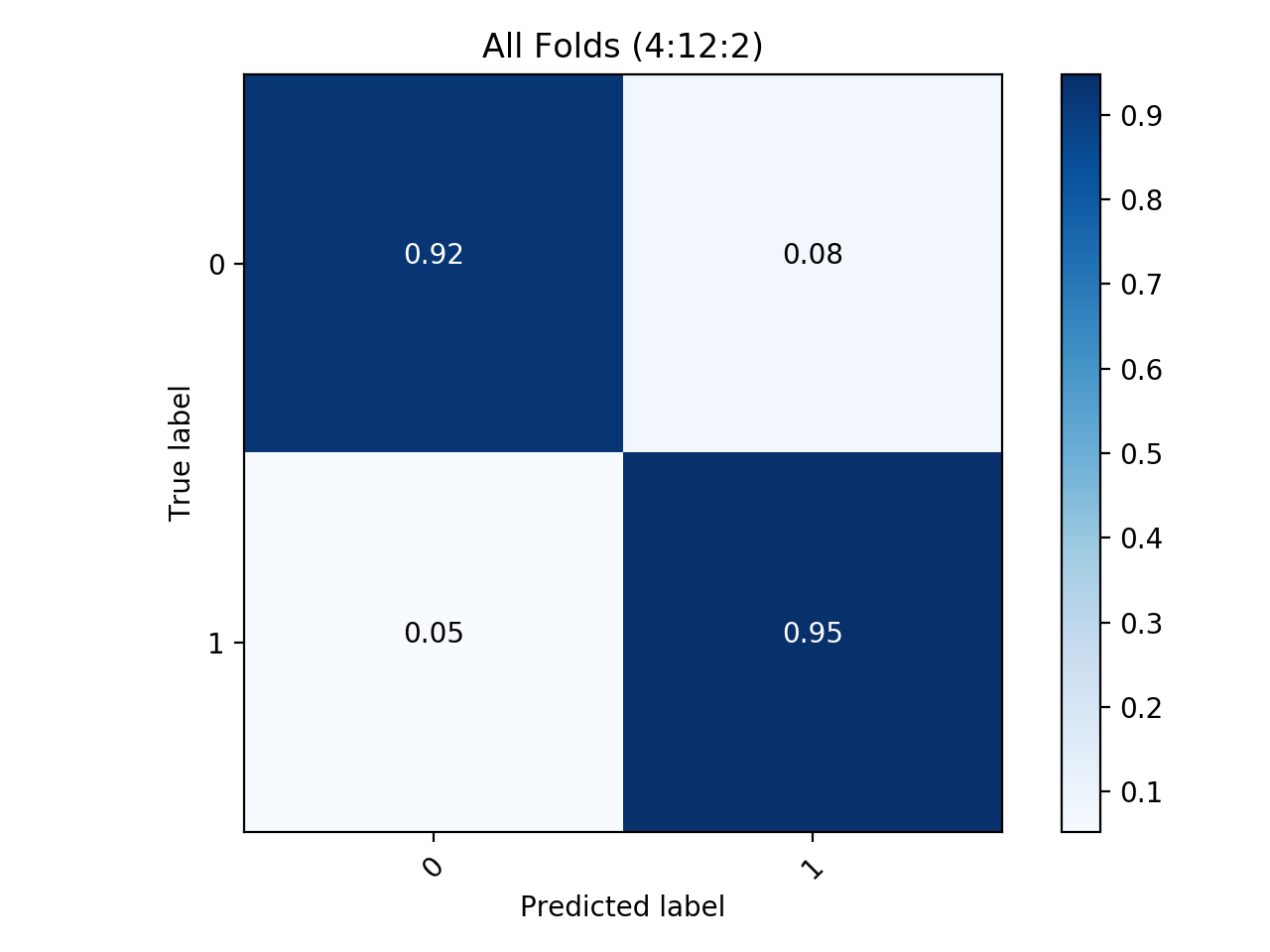
1. Training until convergence shows that the network is in fact working. To check how well it did, we have to check for generalization. To do this, we have to create a new set of points and run them through the model. In this case, for the first two input features we sample [-2.1, 2.1] giving us a point in the square [-2.1, 2.1] x [-2.1, 2.1]. The third coordinate is a random number close to zero. After training our model with all three features from the training data, we run our new set of generated points through a forward pass and classify them. The result of that looks like this.

Overall, the model does very well. We get a well-defined region for each class. The question now is whether or not the third input feature is helping us classify the data. What happens if we remove it from the training set and the set that we just classified? The result of only using two input features looks like this.

The plots look almost identical. This tells us that the third feature does not add any meaningful information into the model. Furthermore, not only does it not help, it could actually hurt the model. By providing more degrees of freedom to the model for no reason, it is highly likely that these extra degrees of freedom will lead to overfitting. In general, the model should be as simple as possible while still getting good results. Therefore, the third input feature does not help in classification and should be left out.

1. Another factor that affects the model is learning rate. By adjusting the learning rate, one can alter the rate at which a model converges. By increasing the learning rate, the model will converge faster, but there is a risk of “overshooting” the minimum that is trying to be found and the weights will actually never converge. With a lower learning rate, the model will have a lower risk of non-convergence, but will take a much longer time to converge. To see this, the model was trained with a 0.01, 0.2, 0.7, and 0.9 learning rate and the results are below.
2. Aside from learning rate, momentum is another meta-parameter that can affect the rate at which a model converges. Momentum simply rewards a model for moving in the correct direction. If a weight has been consistently decreasing, momentum encourages the weight to continue to decrease, thus helping it converge faster. A good demonstration of this is to look at how momentum can speed up the training of the model with a low learning rate. With a learning rate of 0.01, the model trains very slowly, as can be seen in section 2. However, by adding momentum, the training can converge must faster.
3. Now the model is modified to perform 5-fold cross-validation on a new data set that contains four input features in one of two classes. The structure of the network is 4:N:2, where N is the number of hidden nodes in the network. First, the model was trained using 5-fold cross validation where N=8. The results are below.

In addition to N=8, the cross-validation was also performed for the cases N=4 and N=12. The results of these two cases are shown below. Note that these are simply the aggregated confusion matrices, the same type as the bottom-left figure on the previous page.



All tests were carried out using a learning rate of 0.2, momentum of 0.1, and were trained for 500 epochs for each fold. Overall, the accuracy was highest when N=4. However, the training errors were the lowest when N=12. These two facts suggest that some overfitting may have occurred. When given more degrees of freedom, the model “memorized” training data which would explain lower training errors yet lower accuracy on withheld data. Perhaps an improvement on this could be training for fewer epochs while increasing the learning rate slightly. After performing these changes, the training errors increased across all N that were tested, and N=4 still had the highest accuracy on withheld data throughout the cross-validation process.

In conclusion, throughout this lab we saw the effects of adjusting parameters such as learning rate, momentum, and number of nodes in hidden layers. In addition, we looked at how cross validation can be used to assess the accuracy and generalizability of a model. The effects of these parameter modifications behaved similarly to what I was expecting. The biggest surprise to me was the effect of adjusting the number of hidden nodes. Intuition would tell you that the more hidden nodes, the higher the accuracy. However, performing the cross-validation tests shows that adding more nodes does not actually increase accuracy on data that was withheld from the training process.