Programming Exercise 6b: Evaluating a Learning Algorithm (Neural Network)

Data set

In this exercise the training data will be artificial, generated with the help of the `sklearn.datasets.make_blobs` function[1] :

```
from sklearn.datasets import make_blobs

classes = 6
m = 800
std = 0.4
centers = np.array([[-1, 0], [1, 0], [0, 1], [0, -1], [-2, 1], [-2, -1]])
X, y = make_blobs(n_samples=m, centers=centers, cluster_std=std,
  random_state=2, n_features=2)
```

We split the data set into training, cross-validation (CV) and test sets. In this example, we're increasing the percentage of cross-validation data points for emphasis:

```
X_train, X_, y_train, y_ = train_test_split(X, y, test_size=0.50,
  random_state=1)
X_cv, X_test, y_cv, y_test = train_test_split(X_, y_, test_size=0.20,
  random_state=1)
```

Figure 1.1 shows the resulting data set on the left. There are six clusters identified by color. Both training points (dots) and cross-validataion points (triangles) are shown. The interesting points are those that fall in ambiguous locations where either cluster might consider them members. On the right is an example of an 'ideal' model, or a model one might create knowing the source of the data. The lines represent 'equal distance' boundaries where the distance

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

between center points is equal. It's worth noting that this model would "misclassify" roughly 8% of the total data set.

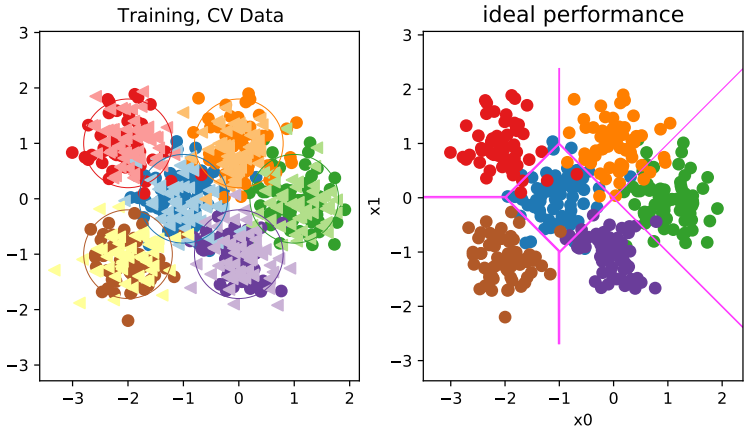

Figure 1.1: Data set and 'ideal model'

Model Complexity

You have to build two neural network models, a complex model and a simple model, and evaluate them to determine if they are likely to overfit or underfit the data set.

Complex model

Compose a three-layer neural network consisting of:

- Dense layer with 120 units, relu activation
- Dense layer with 40 units, relu activation
- Dense layer with 6 units and a linear activation

Train using:

- `torch.nn.CrossEntropyLoss` loss function,
- `torch.optim.Adam` optimizer with learning rate of 0.001, and
- 1000 epochs

This model should capture outliers of each category as shown in Figure 1.2, and, as a result, will miscategorize some of the cross-validation data with roughly 12% misclassified points.

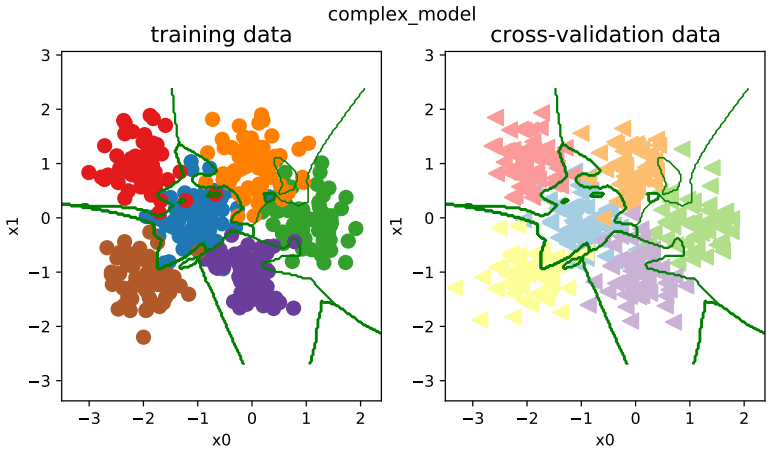

Figure 1.2: Complex model

Simple model

Now, try a simple model. Compose a two-layer model with:

- Dense layer with 6 units, relu activation
- Dense layer with 6 units and a linear activation.

Train using:

- `torch.nn.CrossEntropyLoss` loss function,
- `torch.optim.Adam` optimizer with learning rate of 0.01, and
- 1000 epochs

The simple model has a little higher classification error on training data ($\approx 6\%$) but does better on cross-validation data ($\approx 7.5\%$) than the more complex model as shown in Figure 1.3.

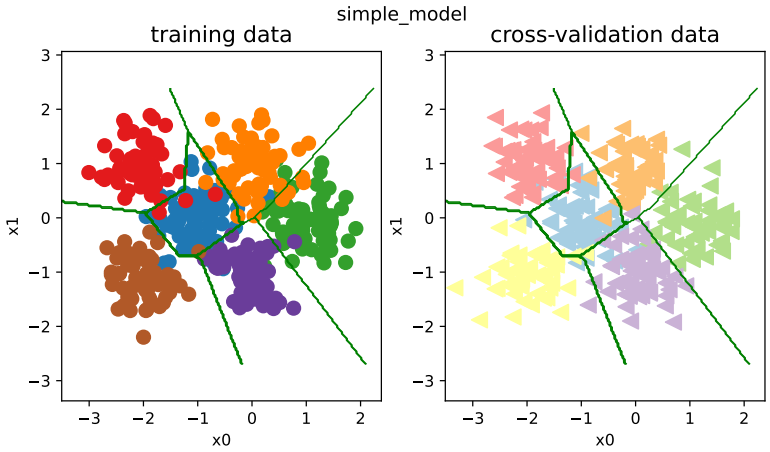

Figure 1.3: Simple model

Regularization

One can apply regularization to moderate the impact of a more complex model. Reconstruct your complex model, but this time include regularization. Compose a three-layer model consisting of:

- Dense layer with 120 units, relu activation
- Dense layer with 40 units, relu activation
- Dense layer with 6 units and a linear activation

Train using:

- `torch.nn.CrossEntropyLoss` loss function,
- `torch.optim.Adam` optimizer with learning rate of 0.001, L2 regularization with $\lambda = 0.01$ (`weight_decay` parameter) and
- 1000 epochs

The results should look very similar to the 'ideal' model, as shown in Figure 1.4, and provide similar results on training ($\approx 6.3\%$) and cross-validation data ($\approx 6.6\%$). The simple model is a bit better in the training set than the regularized model but it worse in the cross validation set.

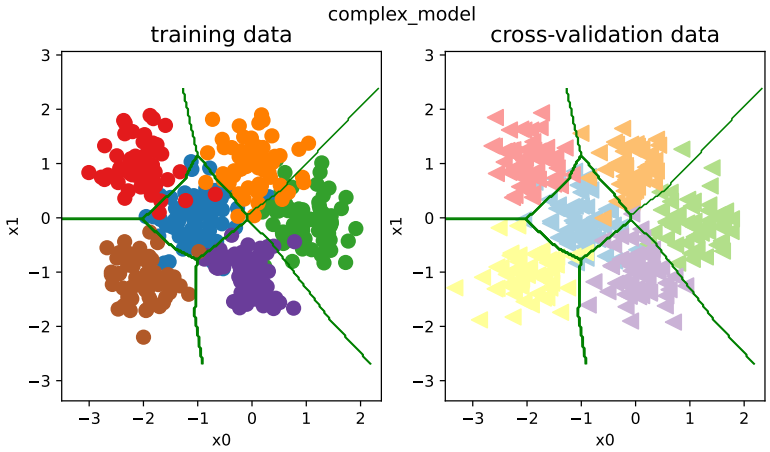

Figure 1.4: Regularized complex model

Iterate to find optimal regularization value

Try the following regularization values to find the optimal one: $[0.0, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3]$. For this data set and model, you should find out that $\lambda \approx 0.05$ seems to be a reasonable choice, as shown in Figure 1.5.

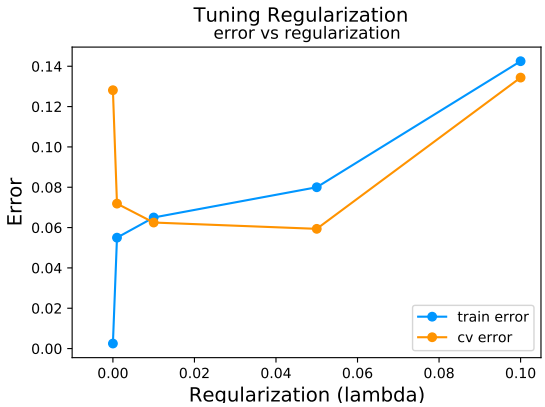

Figure 1.5: Optimal regularization value

Test

Finally, try your optimized models on the test set and compare them to 'ideal' performance. The test set is small and seems to have a number of outliers so classification error will be high. The

performance of the optimized models should be comparable to ideal performance, as shown in Figure 1.6.

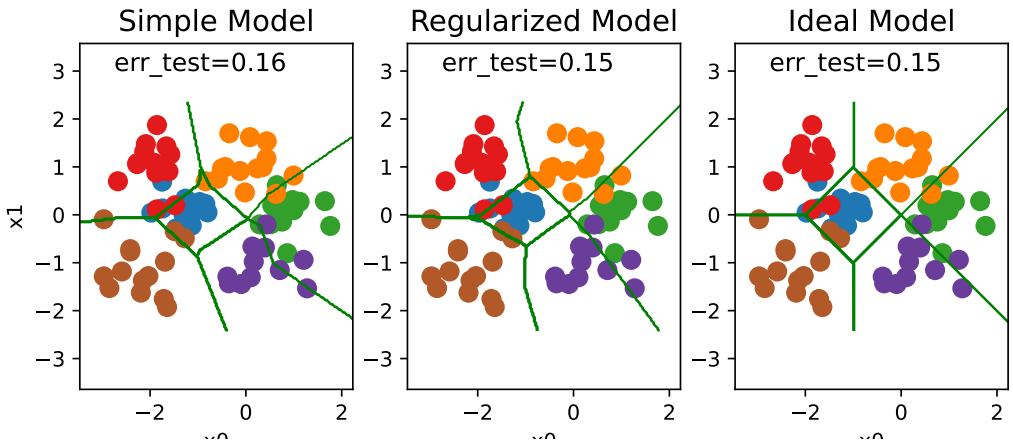

Figure 1.6: Results on the test set

Submission of the assignment

The assignment must be delivered using the submission mechanism of the virtual campus. A single file will be delivered in pdf format containing the memory of the practice, including the code developed and the comments and graphs that are considered most appropriate to explain the results obtained.