Computational MR imaging

Laboratory 9: Machine learning in MRI and neural network architecture design

Nan Lan

1. Diffusion Data Classification:

i) In the data, different regions of brain are labeled. Classes are

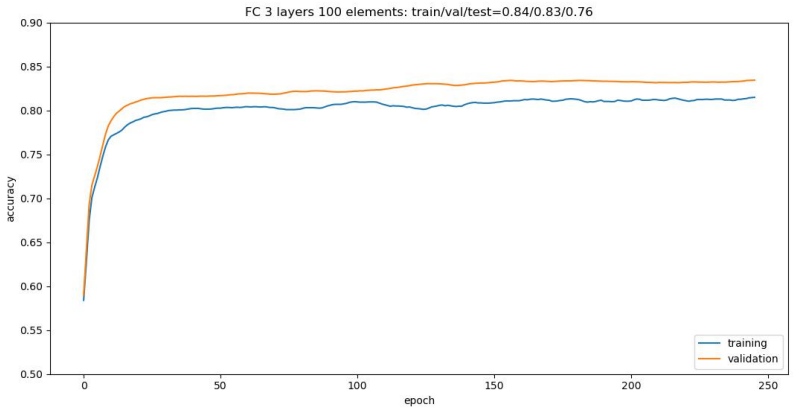
#1: left thalamus 左丘脑

#2: left genu of the corpus callosum 胼胝体左膝

#3: left subcortical white matter of inferior frontal gyrus 额下回左侧皮质下白质

ii) The evaluation and accuracy of the model within 250 epoch are as followed:

```
Evaluation results train data: 0.84
Evaluation results validation data: 0.83
Evaluation results test data: 0.76
```



iii) Dropping different parameter

Overfitting means that the model complexity is higher than the actual problem. The model performs well on the training set but poorly on the test set.

To avoid overfitting, we need to add regularization in the network, for example ‘’Drop Out’’. Concretely, we set activations to 0 randomly, with probability 1 − p.

The figure below shows the neural network model after applying dropout.

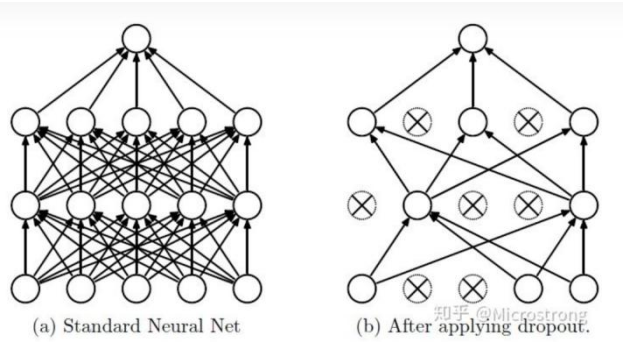


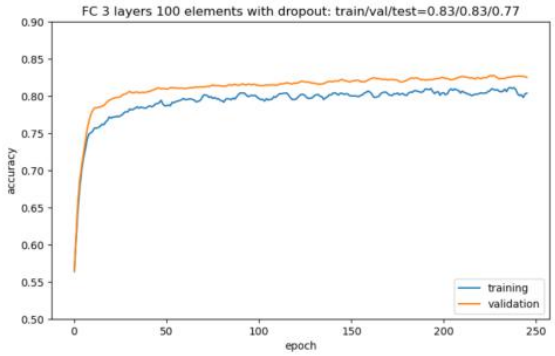
图1：使用Dropout的神经网络模型

The entire dropout process is equivalent to averaging many different neural networks. Different networks produce different overfitting, and some "reverse" fittings cancel each other to reduce overfitting as a whole and improve the robustness.

Here is the structure of model with dropout.

```
dropout_p = 0.7
is_Dropout = True
if is_Dropout:
    model = torch.nn.Sequential(
        nn.Linear(nFeatures, nElements, bias=True),
        nn.ReLU(),
        nn.Linear(nElements, nElements, bias=True),
        nn.ReLU(),
        nn.Linear(nElements, nElements, bias=True),
        nn.ReLU(),
        nn.Dropout(dropout_p),
        nn.Linear(nElements, nElements, bias=True),
        nn.ReLU(),
        nn.Linear(nElements, nClasses, bias=True),
    )
```

The image below is the evaluation and accuracy of the model with dropout.




```
Evaluation results train data: 0.83
Evaluation results validation data: 0.83
Evaluation results test data: 0.77
```

While the accuracy in test data of model without dropout is 0.76, the accuracy in test data of model with dropout is 0.77. The model's generalization ability improves a bit after applying the dropout layer.

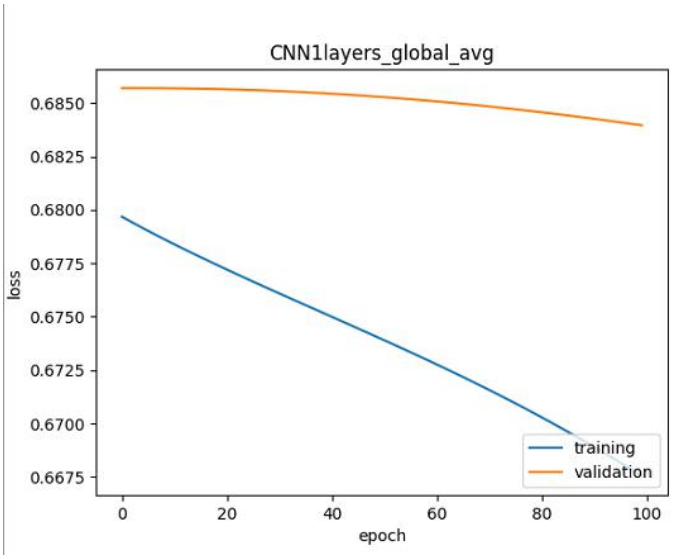
2. Image Quality Classification

The hyperparameter setting of network is: learning rate 1e-5; batch size 8; kernel size: 5×5

1) The first CNN network has 1 convolution layers, following with a fully connected layer.

```
def __init__(self):
    super(Model, self).__init__()
    self.conv1 = nn.Conv2d(1,16,(5,5),padding=(1,1))
    self.batchnorm1 = nn.BatchNorm2d(16)
    self.ReLU = nn.ReLU()
    self.pool = nn.MaxPool2d(2,2)
    self.out = nn.Linear(16,2)
    self.LogSoftmax = nn.LogSoftmax(dim=1)
```

The result below is the loss in 100 epoch.



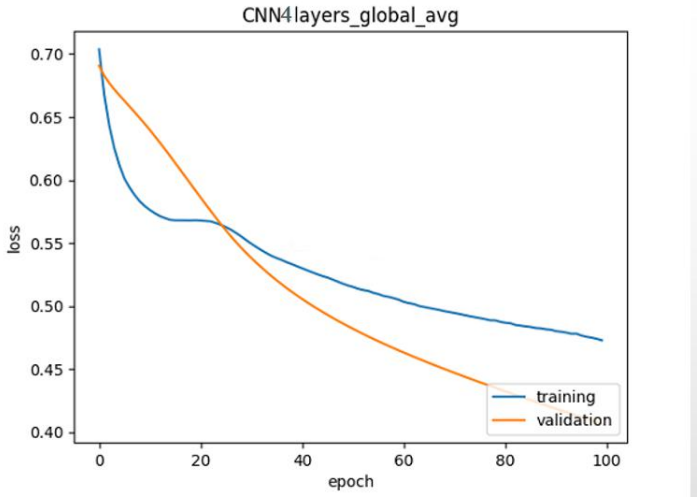
Here is the final loss and accuracy.

```
Training Time (in minutes) = 0.6904472192128499

Evaluation results train data: loss 0.67 acc 0.54
Evaluation results test data: loss 0.68 acc 0.56
```

2) The second CNN network has 4 convolution layers, following with a fully connected layer.

```
self.conv1 = nn.Conv2d(1, 16, kernel_size, padding=pad_size)
self.batchnorm1 = nn.BatchNorm2d(16)
self.ReLU = nn.ReLU()
self.pool = nn.MaxPool2d((2, 2))

self.conv2 = nn.Conv2d(16, 32, kernel_size, padding=pad_size)
self.batchnorm2 = nn.BatchNorm2d(32)
self.ReLU = nn.ReLU()
self.pool = nn.MaxPool2d((2, 2))

self.conv3 = nn.Conv2d(32, 64, kernel_size, padding=pad_size)
self.batchnorm3 = nn.BatchNorm2d(64)
self.ReLU = nn.ReLU()
self.pool = nn.MaxPool2d((2, 2))

self.conv4 = nn.Conv2d(64, 128, kernel_size, padding=pad_size)
self.batchnorm4 = nn.BatchNorm2d(128)
self.ReLU = nn.ReLU()
self.pool = nn.MaxPool2d((2, 2))

self.global_avg_pool = nn.AdaptiveAvgPool2d(1)
self.flatten = nn.modules.flatten.Flatten()

self.FC2 = nn.Linear(128, 2)
self.LogSoftmax = nn.LogSoftmax(dim=1)
```

The result below is the loss in 100 epoch.



Here is the final loss and accuracy.

```
Training Time (in minutes) = 3.1696311990420023

Evaluation results train data: loss 0.47 acc 0.83
Evaluation results test data: loss 0.41 acc 0.52
```

The second CNN network structure is more complex than the first one. Although some methods to against overfitting like implementing batch normalization, decresing the learning rate are used, overfitting still occur. In second CNN network structure, the loss is decreasing along with epoch. However, the accuracy of testing data is low(0.52), even though the accuracy of training data is high(0.83). It means the generalization ability of the model is quite weak.
The conclusion is: the more complex the model is, the higher possibility of overfitting the model has.

PS: Some knowledge about underfitting and overfitting:
Overfitting: the model performs well on the training set, but poorly on the test set and new data.
Underfitting: Due to insufficient learning ability, the model cannot learn the "general laws" in the data set, resulting in weak generalization ability. Model performs baddly both on the training set and the testing set

The reason for overfitting:
Too few samples
Too many parameters, the model complexity is too high;
The sample noise interference is too large, so that the machine regards part of the noise as a feature and disturbs the preset classification rules;

The reason for Underfitting: model is too simple

How to deal with overfitting:
Regularization (L1 and L2)
Data augmentation, that is, adding training data samples
Dropout
Early stopping

How to deal with underfitting:
Reduce the regularization parameter
Adjust the capacity of the model
Add new features