Neural networks in computational modeling

A brief overview

Georgii Zhulikov
24.11.2017

Deep belief networks

- Generative
- Unsupervised
- Flexible (can be upgraded to supervised)
- DBNs are based on simple networks - restricted Boltzmann machines
- Matlab implementation - “DeeBNet” toolbox (http://ceit.aut.ac.ir/~keyvanrad/DeeBNet%20Toolbox.html)
- RBMs are researched

DeeBNet

- Works on Windows and Linux (needs a fix for Mac, I'll send it in a letter)
- Pipeline:
  - Prepare data
  - Set up layer parameters
  - Train the network
  - [Train the network as a supervised one]
  - Check the results
- The toolbox provides functions for all parts of the pipeline
- Compatible with Matlab Neural Network toolbox
- Documentation can be better

Setting up DeeBNet data

```
data=DataClasses.DataStore();
data.valueType=ValueType.gaussian;
data.trainData = myTrainData;
data.testData = myTestData;
data.validationData = myValidationData;
```

Data will be used later for training

```
dbn.train(data);
```

For classification networks there are also fields for labes in DataStore class

Setting up DeeBNet layers

```
dbn=DBN('autoEncoder');


rbmParams=RbmParameters(300,ValueType.probability);
rbmParams.samplingMethodType=SamplingClasses.SamplingMethodType.PCD;
rbmParams.performanceMethod='reconstruction';
rbmParams.maxEpoch=100;
dbn.addRBM(rbmParams);
```

Then repeat for every layer.

There are additional parameters that help better define a network for its task, but they are not documented well, so the only way to learn what they do in practice is to experiment.

Training and testing DeeBNet network

```
dbn.train(data);

dbn.reconstructData(data.testData(n,:),1)
```

Additional steps in case of classifier:

```
classNumber=dbn.getOutput(data.testData,'bySampling');
errorBeforeBP=sum(classNumber~=data.testLabels)/length(classNumber)

dbn.backpropagation(data);

classNumber=dbn.getOutput(data.testData);
errorAfterBP=sum(classNumber~=data.testLabels)/length(classNumber)
```

Matlab autoencoders

- Unsupervised
- Flexible
- Generative
- Have great examples: https://www.mathworks.com/help/nnet/examples/training-a-deep-neural-network-for-digit-classification.html
- Have good documentation
- More black-box-like than DBNs in terms of computational modeling research

Using Matlab autoencoders

```
autoenc1 = trainAutoencoder(xTrainImages,hiddenSize, …
    'MaxEpochs',400, …
    %%% additional parameters %%%
    );
```

xTrainImages: a cell array or a matrix with training data
hiddenSize: number of hidden nodes of this layer

```
figure()
plotWeights(autoenc1);
```

```
xReconstructed = predict(autoenc,myImage);
imshow(xReconstructed{0});
```

A pure autoencoder is only good for reconstruction, but it can also be used to create deep networks

Stacking autoencoders

```
feat1 = encode(autoenc1,xTrainImages);
```

feat1 now represents training data

```
autoenc2 = trainAutoencoder(feat1,hiddenSize2, …
    'MaxEpochs',100, …);
```

```
feat2 = encode(autoenc2,feat1);
```

Now feat2 is just data that can be used as any other data for training networks. We can train a softmax layer to classify this set of features, but we need labels (tTrain) for that.

```
softnet = trainSoftmaxLayer(feat2,tTrain,'MaxEpochs',400);
```

Now we can stack these layers into a single network that will perform all the sequential encoding and classification automatically

```
deepnet = stack(autoenc1,autoenc2,softnet);
view(deepnet)
```

Stacking autoencoders

```
deepnet = stack(autoenc1,autoenc2,softnet);
```

Only one layer here is trained using supervised learning.

We can think of this as a “pre-trained” network. We can fine tune it by training the whole network on a set of data.

```
deepnet = train(deepnet,xTrain,tTrain);
```

This way the use of autoencoders helps initialize a deep classifier so training becomes fine tuning and the results can be expected to be better than with simple training.

Convolutional neural networks

- Classifiers
- Supervised
- Good for images: translation invariance, local connectivity
- Can be built in Matlab NN toolbox(needs to be assembled from layers)
- There are good examples https://www.mathworks.com/help/nnet/deep-learning-image-classification.html
- Documented
- (There are convolutional autoencoders that allow the “Good for images” benefits to be used for generative networks, but you have to use external toolboxes for this)

Setting up Matlab networks

Function **`trainNetwork`** can work with different data.

- A set of images (imageDatastore object)
- A 4-D array (width, height, channel, images)
- Other

```
myNet = trainNetwork(trainData,layers,options);
```

We need to define layers and options

Setting up layers for a convolutional network

```
layers = [...
    imageInputLayer([28 28 1])

    convolution2dLayer([4 3],12)
    reluLayer
    crossChannelNormalizationLayer(4)
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(5,16)
    reluLayer
    crossChannelNormalizationLayer(4)
    maxPooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(256)
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

Setting up options and using the network

```
options = trainingOptions('sgdm',...
    'MiniBatchSize',5,...
    'MaxEpochs',3,...
    'InitialLearnRate',0.0001);

myNet = trainNetwork(trainData,layers,options);

predictedLabels = classify(myNet,testData);
accuracy = sum(predictedLabels==testLabels)/numel(predictedLabels)
```

Thank you