Machine Learning

Lab 3: Deep Learning

1 Overview: sample code in MATLAB

1.1 Single variable regression

The goal, here, is to train a scalar regression network $n(x) \approx \sin(x)$ given noisy training data (`MLPregression.m`):

```
% setting up two hidden layers with 10 nodes, each, all fully connected
layers = [ sequenceInputLayer( 1 )
    fullyConnectedLayer(10)
    tanhLayer
    fullyConnectedLayer(10)
    tanhLayer
    fullyConnectedLayer(1)
    regressionLayer
    ]

% training data
XTrain = 2*pi*rand(1,1e4); %  x, sampled uniformly from [0,2pi]
YTrain = sin(XTrain)+0.1*randn(size(XTrain)); % corresponding y w/ noise

% validation data
XV = 2*pi*rand(1,1e3);
YV = sin(XV);

% training options
options = trainingOptions('sgdm', ...
    'MaxEpochs',2000,...
    'InitialLearnRate',1e-5, ...
    'Verbose',true, ...
    'Plots','training-progress', ...
    'ValidationData', {XV,YV} );

% let MATLAB do the actual training
net = trainNetwork( XTrain, YTrain, layers, options );

% now use the trained network
x = 0:0.001:(2*pi); % testing x
y = net.predict(x); % network's prediction

figure;
plot( x, y ); hold on; % learned function
plot( x, sin(x) );     % true function
plot( XTrain(1:100:end), YTrain(1:100:end), 'ko'); % some data points
```

Notes:

- the input is scalar, represented as a sequence input node of length 1
- each layer of nodes is fully connected to the next; the parameter defines the number of nodes in the next layer
- we use tanh-activation for all nodes (differentiable version of sign-function)
- the output is scalar, represented as a single node
- regression output needs a regressionLayer at the end
- training data is noisy, while validation data is clean
- we visualize the learned regression model by plotting for nicely space x (lines 31–36)
- plotting 1% of the training data shows the noise that went into training

1.2 Classification network

The second example performs binary classification of synthetic swissroll data (includes 2 helper functions).

1) `swissroll.m` creates the synthetic data: (coordinates in `X`, labels in `Y`)

```
function [X,Y] = swissroll(N,sigma)
    t = 2*randn(1,N)+7.5;
    Y = sign(randn(1,N));
    X = [t.*cos(t+pi/2*Y); t.*sin(t+pi/2*Y) ] + sigma*randn(2,N);
end
```

2) `plotroll.m` plots the labeled data:

```
function plotroll(X,Y)
    scatter(X(1,Y>0), X(2,Y>0),'bo'); hold on;
    scatter(X(1,Y<0), X(2,Y<0),'rx'); hold on;
    set(gca,'XLim', [-15,15],'YLim', [-15,15] );
    daspect([1 1 1]);
end
```

The actual MLP code (`MLPclassification.m`):

```
sigma = 0.45;

[XTrain, YTrain] = swissroll(1e5,sigma);
[XV, YV] = swissroll(1e4,sigma);

plotroll(XTrain, YTrain); % show the training data

% setting up two hidden layers with 10 nodes, each, all fully connected
layers = [ sequenceInputLayer( 2 ) % 2-component input
    fullyConnectedLayer(50)
    tanhLayer
```

```
12      fullyConnectedLayer(30)
13      tanhLayer
14      fullyConnectedLayer(20)
15      tanhLayer
16      fullyConnectedLayer(2)      % there are two classes, so two of these nodes
17      softmaxLayer                %
18      classificationLayer         % these two are needed for classification output
19      ]
20
21  % training options
22  options = trainingOptions('sgdm', ...
23      'MaxEpochs',5000,...
24      'InitialLearnRate',1e-7, ...
25      'Momentum', 0.95,...
26      'Verbose',true, ...
27      'Plots','training-progress', ...
28      'ValidationData', {XV,categorical(YV)} );
29
30  % let MATLAB do the actual training
31  net = trainNetwork( XTrain, categorical(YTrain), layers, options );
32
33  % now use the trained network to paint entire feature space
34  [x1,x2]=meshgrid(linspace(-15,15,1000)); % testing x
35  XTest = [x1(:)'; x2(:)'];
36  y = net.classify(XTest); % network's prediction
37
38  YTest = zeros(1,size(XTest,2));  % convert categorical to numerical output
39  YTest(y == '1') = 1;
40  YTest(y == '-1') = -1;
41
42  figure; plotroll(XTest,YTest); % plot the classificatoin landscape
```

Notes:

- the input is 2D, represented as two input nodes
- the output is 2D, since MATLAB only knows 1-in-k coding
- classification output needs a softmaxLayer and a classificationLayer
- the training/validation labels need to be categorical
- we visualize the learned classification landscape by querying an entire grid of feature points (lines 34–36)
- recovering numbers from categorical labels is somewhat painful (lines 38–40)

2 Provided Resources

- All the sample functions, given above.
- `cbcl1.mat` and `news.mat` that was given for the SVM-lab.

3 Guide

1. First, play around:

   (a) Load and run the MLP regression code. You have to use MATLAB with the Deep Learning Toolbox and running version 2019b or newer—MATLAB in the cloud will work, but interaction with the toolbox window might be slow.

   (b) Study the plot. What does "loss" seem to represent? Is it going down monotonically? Is there a difference between training and validation performance? Explain what you observe!

   (c) Change some of the optimization parameters: What happens if you pick an initial learning rate that is 10x, 100x bigger/smaller? Can you find out what an "Epoch" is? What does `sgdm` stand for? Replace `sgdm` with a different choice (e.g., `adam` with much bigger initial learning rate, e.g., 0.1), and see what happens.

   (d) Remove a layer of nodes / Add another layer of nodes / Change the number of nodes in each layer (e.g., 5, 20, or 50). What happens?

   (e) Repeat the above for the MLP classification code.

2. Build/train a two-input–two-output regression network for Cartesian to polar conversion: $n(x,y) \approx \text{cart2pol}(x,y)$. Plot the learned radius and angle landscape to "see" how good the training works.

3. Pick CBCL1 or news data, and build a binary classifier. Put randomly selected portions of the data (10%, each) away for validation and testing (network trains on 80% data, validates on 10% while training, and when done, you run the network on the 10% left out for testing). What percent correct can you achieve on the training/validation/testing data? How does this compare against SVM (train/test SVM on the same data partitions used for the network).