EE698R: Advanced Topics in Machine Learning

Recurrent Neural Networks

Lecture Notes

Ananyae Bhartari - 190129
Devansh Parmar - 190273

1 Sequential data

1.1 What is sequential data?

When the points in the data set are dependent on the other points in data set then the data is termed as sequential.
Examples of sequential data:-

- Video data
- Audio or speech signals
- Stock prices

1.2 Methods to deal with Sequential Data

The various approaches we have seen so far are:-

- Template matching
- HMM
- CNN

1.3 Template matching

1.3.1 Pros

- Very quick to implement
- No training needed

1.3.2 Cons

- No knowledge of optimal templates
- Not robust, hence variation in data leads to problems
- Computationally inefficient
- No training needed

1.4 HMM

1.4.1 Pros

- Easy to train and computationally manageable
- Compared to template matching more robust to variation in data
- Can amalgamate easily with powerful models for $p(x_n|z_n)$

1.4.2 Cons

- Self transition probability decays exponentially

$$\mathrm{P}(z_n = k|z_{n-1} = k)^M$$

- First order Markov assumption

1.5 CNN

1.5.1 Pros

- Shared parameter (hence easy to train)

- Can be implemented faster using parallelisation

1.5.2 Cons

- Finite context length hence cant handle long term dependencies

- Not robust, hence variation in data leads to problems

- Computationally inefficient

2 Recurrent Neural Networks

A Recurrent Neural Network differs from a regular Neural Network in the sense that it has a temporal delay present in the system. In the regular NN, if we feed input $x^{(t)}$ at time instant t, then the output $y^{(t)}$ depends only on $x^{(t)}$, and not other value of x at some different time instant. This is because the hidden layer $h^{(t)}$ depends only on the input $x^{(t)}$. [Fig. 1] Whereas in an RNN, the hidden layer $h^{(t)}$ not only depends upon the input $x^{(t)}$, but also the last hidden layer $h^{(t-1)}$. [Fig. 2] This creates a slightly changed update equation, which is given below.

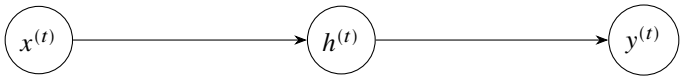

Figure 1: Neural Network

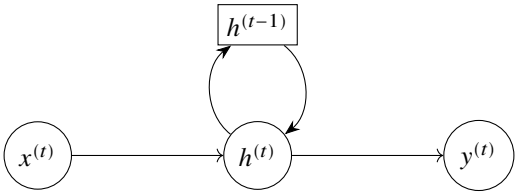

Figure 2: Recurrent Neural Network

2.1 Difference in Update equation

2.1.1 NN

$$h^{(t)} = f(Ux^{(t)} + b) \tag{1}$$

$$y^{(t)} = softmax(Vh^{(t)} + c) \tag{2}$$

Here,

$h^{(t)}$ is hidden layer

U and V are weights, while b and c are biases

$x^{(t)}$ is input, and $y^{(t)}$ is output

2.1.2 RNN

$$h^{(t)} = f(Ux^{(t)} + Wh^{(t-1)} + b) \tag{3}$$

$$y^{(t)} = softmax(Vh^{(t)} + c) \tag{4}$$

Here,

$h^{(t)}$ is hidden layer

U, V and W are weights, while b and c are biases

$x^{(t)}$ is input, and $y^{(t)}$ is output

2.2 Graph Unfolding

To evaluate $h^{(t)}$ we need the value of previous $h^{(t-1)}$,and hence $h^{(t-2)}$, and so on.

$$h^{(t)} = f(h^{(t-1)}, \theta) \tag{5}$$

Where θ is a parameter. Now when we do it recursively (Say where $t = 3$) :-

$$h^{(3)} = f(h^{(2)}; \theta) \tag{6}$$

$$h^{(3)} = f(f(h^{(1)}; \theta); \theta) \tag{7}$$

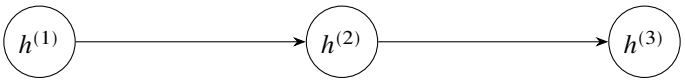

Figure 3: Graph Unfolding

2.3 RNN: Information Representation

$$h^{(t)} = f'(x^{(t)}, x^{(t-1)}, ..., x^{(1)}; \theta)$$

But this is a lossy representation as we're mapping sequences of $x^{(t)}, x^{(t-1)}, ..., x^{(1)}$ to a fixed $h^{(t)}$. The training will itself select what to keep, and the rest would be discarded.

2.3.1 Forward pass: Inference

Given $x^{(1:t)}$ and parameter θ, $y^{(t)}$ can be calculated. Now, this can not be parallelized and will take O(t) time.

2.4 Parameter Training

The parameters are trained through Back Propagation Through Time (BPTT).

$$h_{i_1}^{(t)} = \sigma(\sum_{i_o} u_{i_i i_2} h_{i_1}^{(t)} + c_{i_2}) \tag{8}$$

$$y_{i_2}^{(t)} = softmax(\sum_{i_1} V_{i_o i_1} x_{i_o}^{(t)} + \sum_{j_1} w_{j_1 i_1} h^{(t-1)} + b_{i_1}) \tag{9}$$

Updating V and C are straight forward, but to update U, W, b we need $h^{(t-1)}$. We can unfold further to $h^{(t-M)}$ where M is an integer.