Recurrent Neural Networks

Chris Williams

School of Informatics, University of Edinburgh

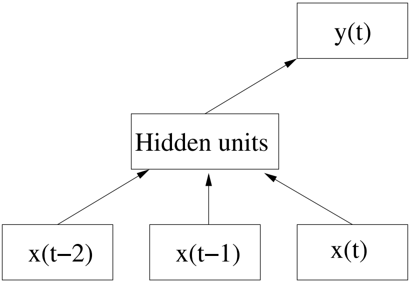
February 2019

- There was a wave of neural networks research from around 1985 to the early 1990s
- There is a lot of recent interest around “deep learning”, starting from around 2010
- For images: convolutional neural networks (CNNs)
- For sequences: recurrent neural networks

Time Series Tasks

- Prediction: predict $\mathbf{x}_t$ given $\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots$
- Labelling/annotation: predict y_t given $\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots$
- (Warning: change of notation relative to FSLDS slides)

A Simple Approach: Sliding Windows (CNN)

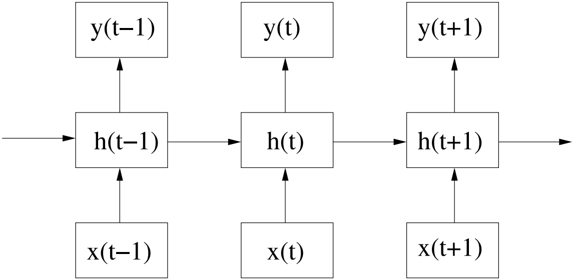


- *Finite* context determined by window width

$$h_i(t) = \sigma\left(\sum_{\tau=0}^{2}\sum_{j} w_{ij}^{\tau} x_j(t-\tau)\right)$$

where $\sigma(z) = 1/(1+e^{-z})$

Recurrent Neural Networks (RNNs)

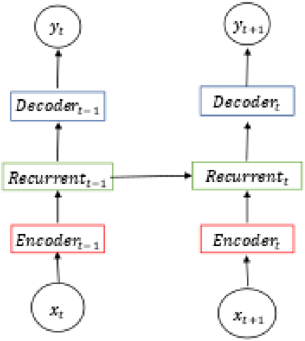


- RNN can potentially capture infinite context
- Trained by *backpropagation* (i.e., the chain rule)
- An RNN for a sequence of T inputs can be viewed as a deep T-layer network with shared weights
- Compare to SLDS structure and information flow (discriminative vs generative)

- Backpropagation involves taking the product of many gradients, which can lead to vanishing (component gradients < 1) or exploding (component gradients > 1) gradients
- This can prevent effective training
- Solution 1: modified optimization algorithms (e.g. RMSProp)
- Solution 2: modified hidden unit transfer functions, e.g. Long short term memory (LSTM, Hochreiter and Schmidhuber, 1997)
- LSTMs have self-recurrence for each hidden unit (long-term memory), and *gates* which are a function of their inputs

Example: Encoder-Recurrent-Decoder Networks

Fragkiadari et al, ICCV 2015



- Recurrent units use LSTMs
- Used for labelling video and motion capture data

Example: Predicting Patient State-of-Health

McCarthy and Williams (2016)

- Tackled the same tasks as for the DSLDS
- Recurrent network gave similar levels of performance to DSLDS

Comparing probabilistic graphical models and RNNs

- Probabilistic graphical models (PGMs) explicitly model uncertainty
- PGMs allow unsupervised learning (e.g. X-factor)
- PGMs may make rigid assumptions and require significant feature engineering
- RNNs allow representations to be learned automatically, but may not be so interpretable
- RNNs are "data hungry", PGMs can incorporate prior knowledge to mitigate this
- Johnson et al (NIPS 2016) argue that one can get the best of both worlds via latent graphical models and NN observation likelihoods