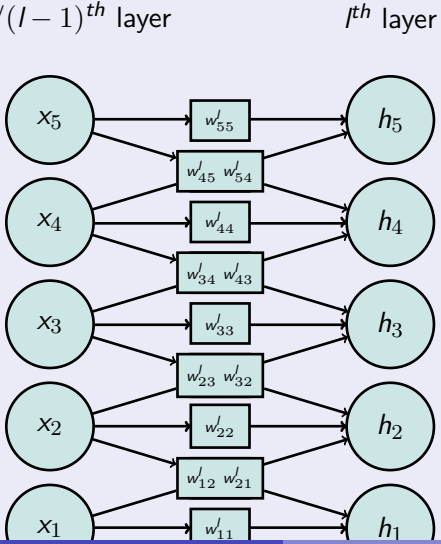
Lecture 22 contd: Convolutional And Recurrent Neural Networks

Instructor: Prof. Ganesh Ramakrishnan

Recap: The Lego Blocks in Modern Deep Learning

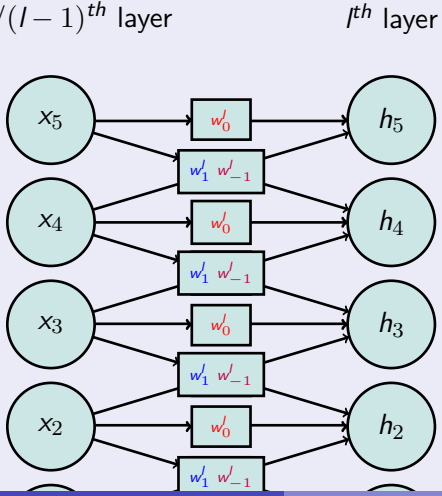
1. **Depth/Feature Map**
2. **Patches/Kernels (provide for spatial interpolations) - Filter**
3. **Strides (enable downsampling)**
4. **Padding (shrinking across layers)**
5. Pooling
6. Embeddings
7. Memory cell and Backpropagation through time

Convolution: Sparse Interactions through Kernels (for Single Feature Map)



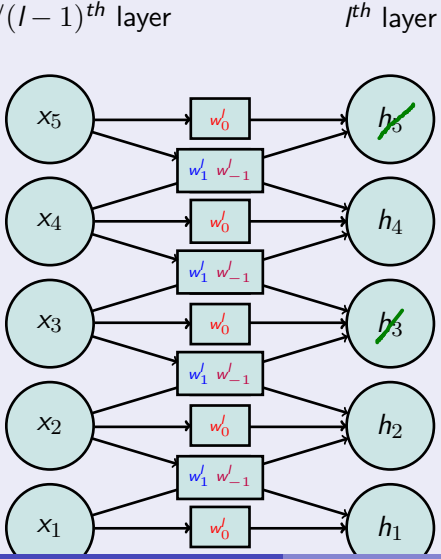
- $h_i = \sum_m x_m w_{mi} K(i-m)$
- On LHS, $K(i-m) = 1$ *iff* $|m-i| \leq 1$
- For 2-D inputs (such as images): $h_{ij} = \sum_m \sum_n x_{mn} w_{ij,mn} K(i-m, j-n)$
- Intuition: Neighboring signals x_m (or pixels x_{mn}) more relevant than one's further away, reduces prediction time
- Can be viewed as multiplication with a Toeplitz[a] matrix K
- Further, K is often sparse (eg: $K(i-m) = 1$ *iff* $|m-i| \leq \theta$)

Convolution: Shared parameters and Patches (for Single Feature Map)



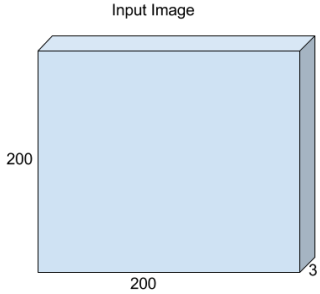
- $h_i = \sum_m x_m w_{i-m} K(i-m)$
- On LHS, $K(i-m) = 1$ *iff* $|m-i| \leq 1$
- For 2-D inputs (such as images): $h_{ij} = \sum_m \sum_n x_{mn} w_{i-m,j-n} K(i-m, j-n)$
- **Intuition:** Neighboring signals x_m (or pixels x_{mn}) affect in similar way irrespective of location (*i.e.*, value of m or n)
- **More Intuition:** Corresponds to moving **patches around the image**
- Further reduces *storage* requirement;

Convolution: Strides and Padding (for Single Feature Map)



- Consider only h_i's where i is a multiple of s.
- **Intuition:** Stride of s corresponds to moving the patch by s steps at a time
- **More Intuition:** Stride of s corresponds to downsampling by s
- What to do at the ends/corners: Ans: **Pad** with either 0's (**same padding**) or let the next layer have fewer nodes (**valid padding**)
- Reduces *storage* requirement as well as prediction time

The Convolutional Filter



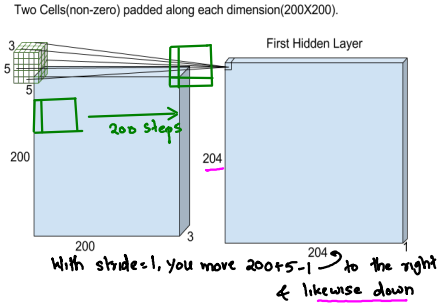
(Patch ⇒ shared wts)

Patch of size 5×5×3

RGB

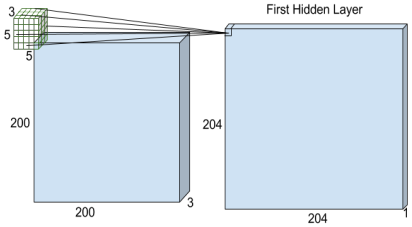
Note: There should be atleast 1 pixel overlap between the patch & image

The Convolutional Filter

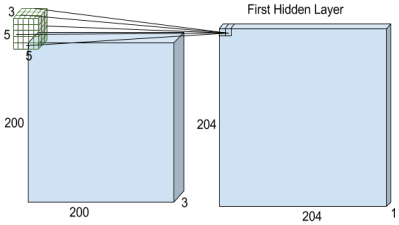


The Convolutional Filter

Two Cells(non-zero) padded along each dimension(200X200).



Filter shifted right by the stride of 1.



The Convolutional Filter

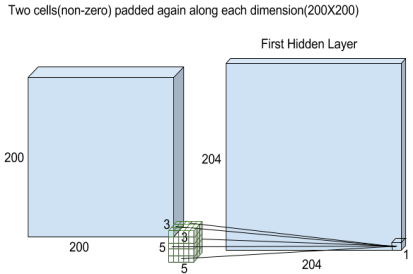


Image Example MLP Vs CNN

Input Image Size: 200 X 200 X 3
MLP: Hidden Layer has 40k neurons, so it has 4800000 parameters.
CNN: Hidden layer has 20 feature-maps each of size 5 X 5 X 3 with stride $= 1$, *i.e.* maximum overlapping of convolution windows.
Question: How many parameters?
Answer: Just 1500 $= 20 * (5 \times 5 \times 3)$
Question: How many neurons (location specific)?

Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let s be stride length
Answer:
Output size $= \left(\frac{M+P}{s} - 1\right) \times \left(\frac{N+Q}{s} - 1\right)$.
To discard the (corner) case when kernel has no overlap with image
$20 \times ((200+5)/\textit{stride}) - 1) \times ((200+5)/\textit{stride}) - 1)$
$= 832320$ (around 830 thousand which can increase with max-pooling).

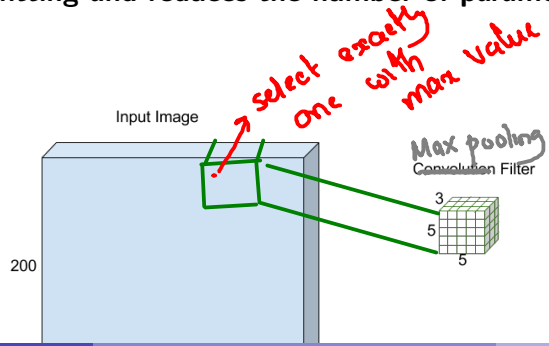
The Lego Blocks in Modern Deep Learning

1. Depth/Feature Map
2. Patches/Kernels (provide for spatial interpolations) - **Filter**
3. Strides (enable downsampling)
4. Padding (shrinking across layers)
5. **Pooling** (More downsampling) - **Filter**
6. **RNN and LSTM** (Backpropagation through time and Memory cell) (??)
7. Embeddings (After discussing unsupervised learning)

The Max Pooling Filter

Max pooling is a (special purpose) downsampling filter/kernel that selects the maximum value from its patch.

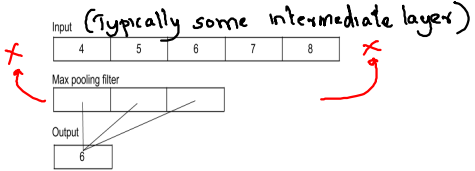
- **It is a sample-based discretization process.**
- **Objective is dimensionality reduction through down-sampling of input representation (eg: image),** some kind of "existential" quantification
- **Allows for translation invariance to the internal representation.** $\exists x$ s.t. ...
- **Helps avoid overfitting and reduces the number of parameters to learn.**



Therefore max pooling is generally applied only after a few regular convolution layers

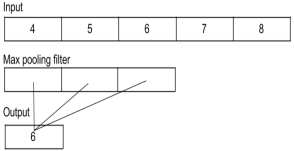
Max pooling (with downsampling) for a Single Feature Map

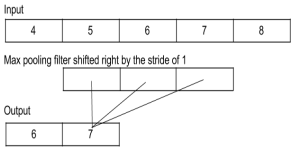
1-d example



Max pooling (with downsampling) for a Single Feature Map

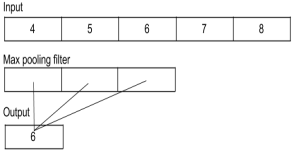
1-d example

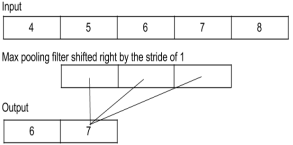


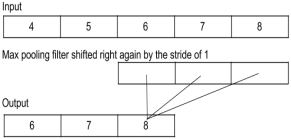


Max pooling (with downsampling) for a Single Feature Map

1-d example



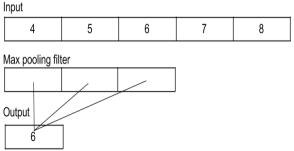


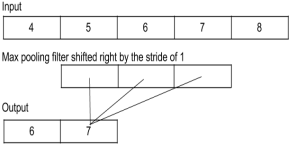


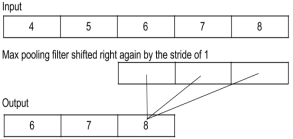
Note: i/p size = 5 (M)
max pooling filter = 3 (P)
o/p size = 3 $\left(\frac{M-P}{S}+1\right)$

Max pooling (with downsampling) for a Single Feature Map

1-d example





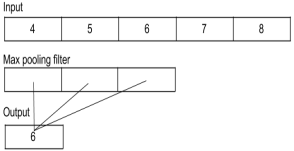


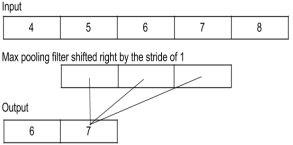
What will be the output if input and max pooling filter remains same but stride changes to 2?

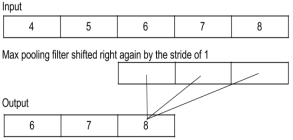
$\frac{M-P}{S}+1 = 2$ & 6 8 will be o/p

Max pooling (with downsampling) for a Single Feature Map

1-d example







What will be the output if input and max pooling filter remains same but stride changes to 2?
[6,8]

Max pooling in 2-D for a Single Feature Map

- Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let s be stride length
- Max pooling takes every $M \times N \times 3$ patch from the input and set the output to the maximum value in that patch
- Output size = $\left(\frac{M-P}{s}+1\right)\left(\frac{N-Q}{s}+1\right)$

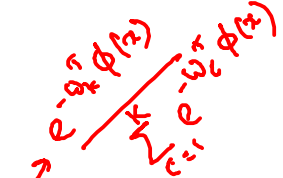
Max pooling in 2-D for a Single Feature Map

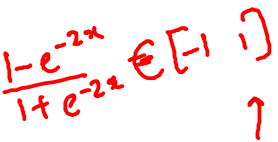
- Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let s be stride length
- Max pooling takes every $M \times N \times 3$ patch from the input and set the output to the maximum value in that patch
- Output size $= \left(\frac{M-P}{s} + 1\right) \times \left(\frac{N-Q}{s} + 1\right)$. For Eg:
  - Input: A 3D image of size with $M = N = 5$, $P = Q = 3$ and with (default) stride of 1.
  - Output size will be $3 \times 3 \times 1$

Tutorial 8, Problem 5

ConvNetJS (`http://cs.stanford.edu/people/karpathy/convnetjs/`) is a Javascript library for training Deep Learning models (Neural Networks) entirely in your browser. Try different choices of network configurations which include the choice of the stack of convolution, pooling, activation units, number of parallel networks, position of fully connected layers and so on. You can also save some network snapshots as JSON objects. What does the network visualization of the different layers reveal?
Also try out the demo at `http://places.csail.mit.edu/demo.html` to understand the heat maps and their correlations with the structure of the neural network.

Tutorial 8, Problem 6





Discuss the advantages and disadvantages of different activation functions: tanh, sigmoid, ReLU, softmax. Explain and illustrate when you would choose one activation function in lieu of another in a Neural Network. You can also include any experiences from Problem 5 in your answer.

Alex-net [NIPS 2012]

For max pooling, stride > 1 amounts to assumptions abt image such as location of objects etc

Stack of two types of **parallel** networks

- First 5 convolution layers
  - First convolution layer takes input of size $224 \times 224 \times 3$, 48 $(\times 2)$ features each with filter/kernel of size $11 \times 11 \times 3$ with stride of 4
    - ⋆ Thus, $((224 + 11)/4 - 1) \times ((224 + 11)/4 - 1) = 57 \times 57$.
  - Max-pooling $(3 \times 3 \times 1$ with stride of $1)$ in the end reduces size to 55×55 for each filter
- Fully connected last 3 layers

Typically max pooling is restricted to stride = 1

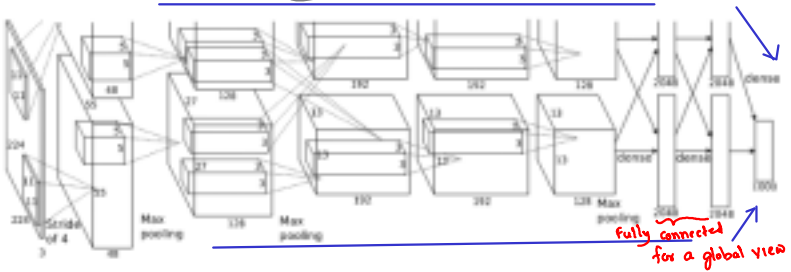
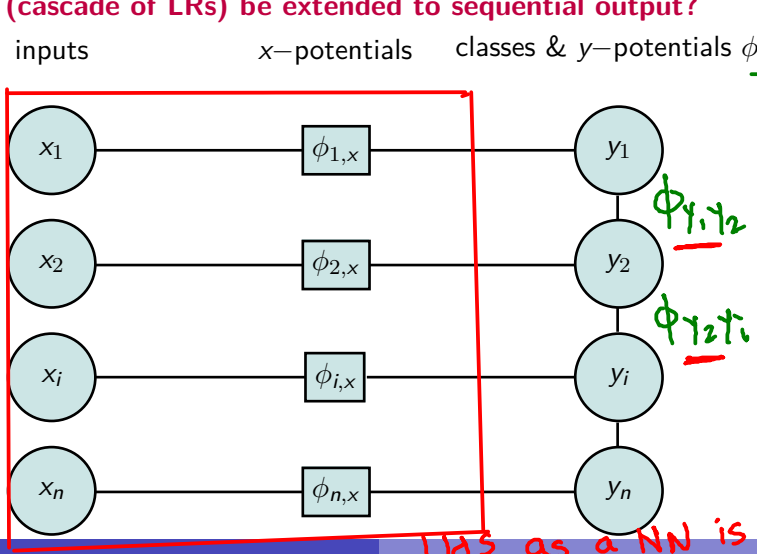


Image reference: "Imagenet Classification with Deep Convolution Neural Networks", NIPS 2012.

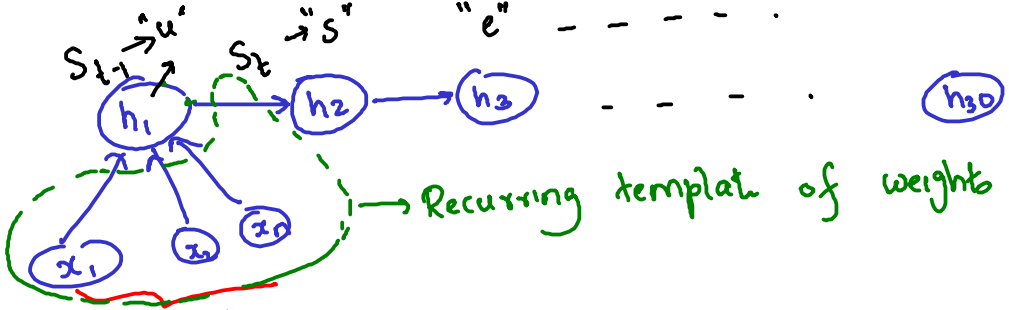
Recap: Linear Conditional Random Fields (CRF)

Just as CRF was an extension of Logistic Regression (LR) can Neural Networks (cascade of LRs) be extended to sequential output?



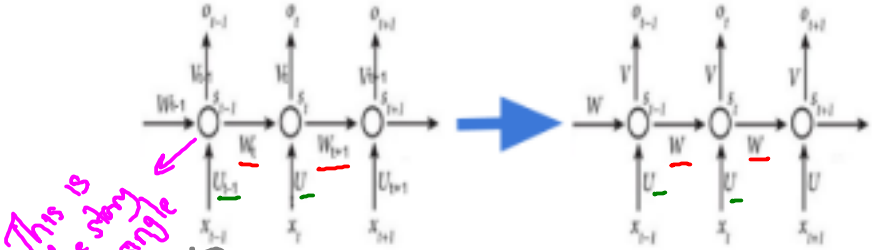
Recurrent Neural Network (RNN) Intuition

- Recall: In CNN we used the trick of common parameters for many neurons
- RNN intuition 1: We want a neuron's output at time t to depend on its state s at time $t-1$
- RNN intuition 2: Share parameters across time steps
- Recurrent $\Rightarrow$ Performing the same task for every element of sequence.



Recurrent Neural Network (RNN) Intuition

- Recall: In CNN we used the trick of common parameters for many neurons
- RNN intuition 1: We want a neuron's output at time t to depend on its state s at time $t-1$
- RNN intuition 2: Share parameters across time steps
- Recurrent $\Rightarrow$ Performing the same task for every element of sequence.

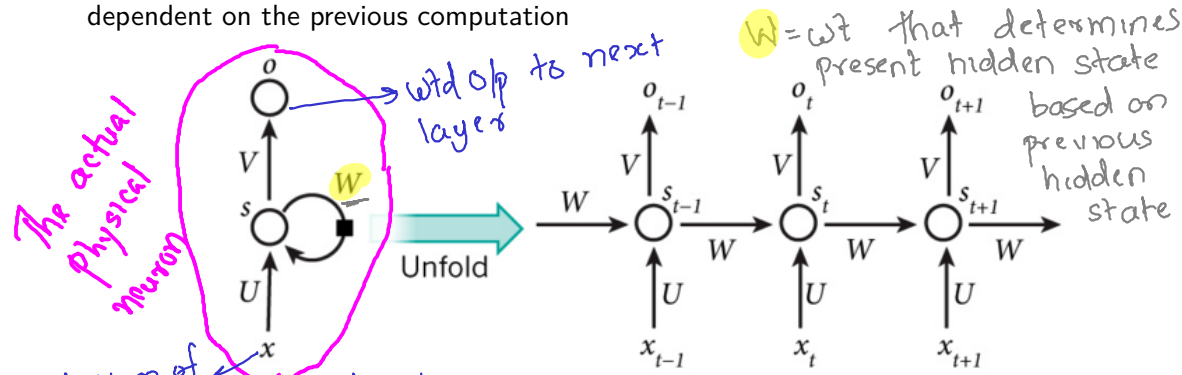


Tutorial 8: Problem 7

Try the text generation RNN (Recurrent Neural Network) demo at `http://www.cs.toronto.edu/~ilya/rnn.html`. State any interesting observations. How would you improve the performance of the RNN?

RNN: Compact representation

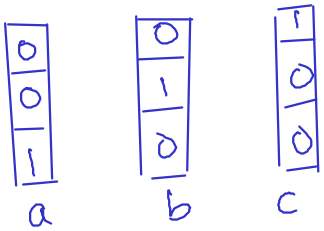
- Generalization of Neural networks to Sequential tasks such as *language modeling*, *word prediction*, *etc.*.
- Perform the same *task* for every element of the sequence, with the output being dependent on the previous computation



A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

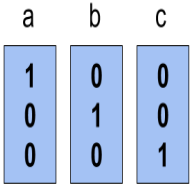
RNN: One Hot Encoding for Language Model

- With 3 characters in vocabulary, a,b and c, what would be the best encoding to inform each character occurrence to the network?
- One Hot Encoding: Give a unique key k to each character in alpha-numeric order, and encode each character with a vector of vocabulary size, with a 1 for the k^{th} element, and 0 for all other elements.

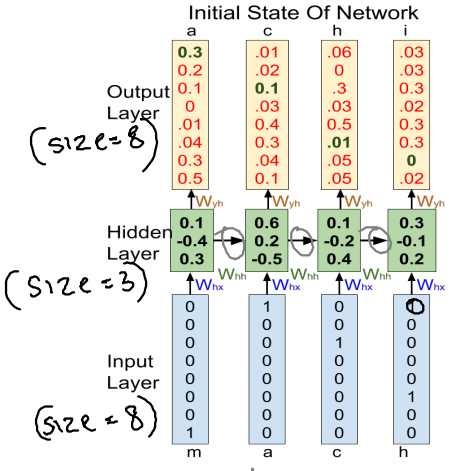


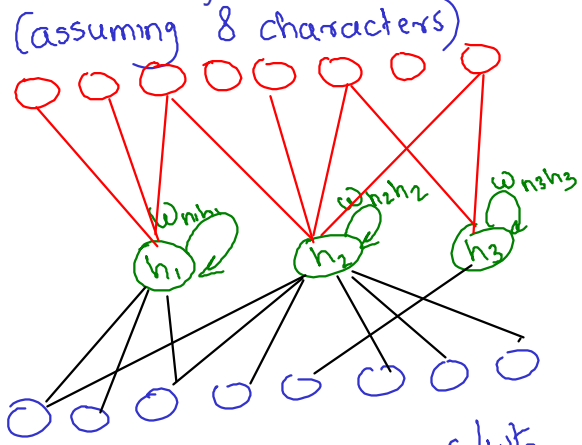
RNN: One Hot Encoding for Language Model

- With 3 characters in vocabulary, *a*,*b* and *c*, what would be the best encoding to inform each character occurrence to the network?
- One Hot Encoding: Give a unique key *k* to each character in alpha-numeric order, and encode each character with a vector of vocabulary size, with a 1 for the k^{th} element, and 0 for all other elements.

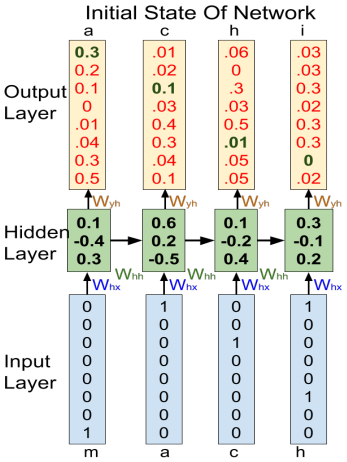


RNN: Language Model Example with one hidden layer of 3 neurons





RNN: Language Model Example with one hidden layer of 3 neurons



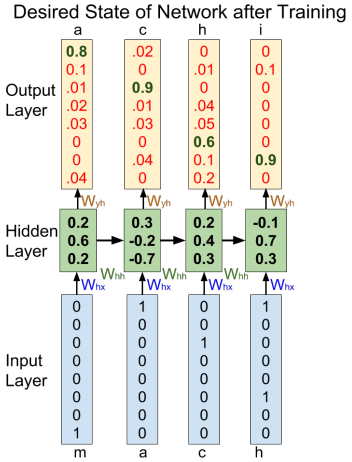


Figure: Unfolded RNN for 4 time units

RNN: Equations

wtd sum of activations coming into this neuron at time t

- $h_t = g(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$
  - Activation function g could be sigmoid σ or its extension to multiclass called softmax[1], tanh $\left(\frac{1-e^{-2x}}{1+e^{-2x}}\right)$ which is simply a scaled[2] and shifted version of the sigmoid function
  - A network may have combination of different activation functions[3]
- $y_t = W_{yh}\ h_t$ $\left(\text{or} \quad y_t = \sum_i w_{y_h i}\ h_t^i\right)$
- The new (present) hidden state depends upon the previous hidden state(s) and the present input.
- The present output depends upon present hidden state (and in turn upon previous hidden states).

[1] Tutorial 7

[2] $tanh(x) = 2\sigma(2x) - 1$

[3] http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/

Back Propagation Through Time: BPTT Illustration

- $h_1 = g(W_{hh}h_0 + W_{hx}x_0 + b_h)$, initialize h_0 and x_0 as zero vectors.

Back Propagation Through Time: BPTT Illustration

- $h_1 = g(W_{hh}h_0 + W_{hx}x_0 + b_h)$, initialize h_0 and x_0 as zero vectors.
- At $t = 2$ we present x_2 as 'a' at input and desire y_2 as 'c' at output in one hot encoded form as shown previously
- $h_2 = g(W_{hh}h_1 + W_{hx}x_1 + b_h)$

Back Propagation Through Time: BPTT Illustration

- $h_1 = g(W_{hh}h_0 + W_{hx}x_0 + b_h)$, initialize h_0 and x_0 as zero vectors.
- At $t = 2$ we present x_2 as 'a' at input and desire y_2 as 'c' at output in one hot encoded form as shown previously
- $h_2 = g(W_{hh}h_1 + W_{hx}x_1 + b_h)$
- At $t = 3$, $x_3 =$ 'c', y_3 we desire is 'h'.
- $y_3 = W_{yh}\ \sigma(W_{hh}h_2 + W_{hx}x_2 + b_h)$

Back Propagation Through Time: BPTT Illustration

- $h_1 = g(W_{hh}h_0 + W_{hx}x_0 + b_h)$, initialize h_0 and x_0 as zero vectors.
- At $t = 2$ we present x_2 as 'a' at input and desire y_2 as 'c' at output in one hot encoded form as shown previously
- $h_2 = g(W_{hh}h_1 + W_{hx}x_1 + b_h)$
- At $t = 3$, $x_3 =$ 'c', y_3 we desire is 'h'.
- $y_3 = W_{yh}\ \sigma(W_{hh}h_2 + W_{hx}x_2 + b_h)$
- Put h_1 and h_2 in the last equation and then tune weights (through back propagation) to get the appropriate y_3 first corresponding to vectors x_3, x_2 and x_1.
- Similarly use h_1 in equation for y_2 and tune weights to get the appropriate y_2 corresponding to vectors x_2 and x_1.
- Then tune for y_1.

RNN Parameters

- In previous example, we used the sequence length of 4, i.e. no. of time steps to unroll the RNN.
- We used the batch size of 1, i.e. the number of input vectors presented at single time step to the RNN.
- One hot encoding is the best suited encoding for such tasks while training the neural networks.
- We can vary these parameters according to the task at hand.

RNN Limitations

- We want to train our networks with long term dependencies.
- In RNN, the influence of the given input decays exponentially as it cycles around the network recurrent connections. The limitation of learning small context of RNN is called "vanishing gradient".
- Gradient vanishes especially when we use sigmoid function and several gradient values v with $|v| < 1$, get multiplied during BPTT to give a zero.
- Instead, if we used an alternative function that gives value > 1 as output, we will face the problem of 'exploding gradient'.

$V_1 V_2 V_3 \cdots \rightarrow 0$

LSTM

Solution: Memory that captures selected history short long

Vanishing Gradient Problem

The sensitivity(derivative) of network w.r.t input(@t $= 1$) decays exponentially with time, as shown in the unfolded (for 7 time steps) RNN below. Darker the shade, higher is the sensitivity w.r.t to x_1.

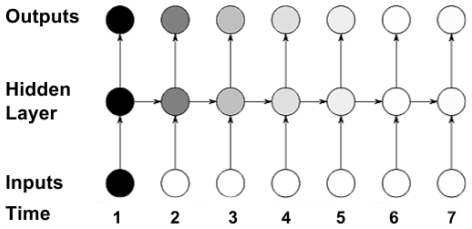


Image reference: Alex Graves 2012.

Long Short-Term Memory (LSTM) Intuition

- Learn when to propagate gradients and when not, depending upon the sequences.
- Use the memory cells to store information and reveal it whenever needed.
- I live in **India**.... I visit **Mumbai** regularly.
- For example: Remember the context "India", as it is generally related to many other things like language, region etc. and forget it when the words like "Hindi", "Mumbai" or End of Line/Paragraph appear or get predicted.