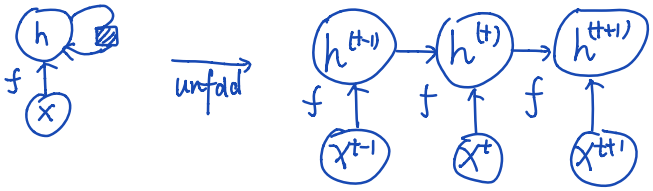
Unfold Computational Graphs

- Two representations of Recurrent Neural Networks



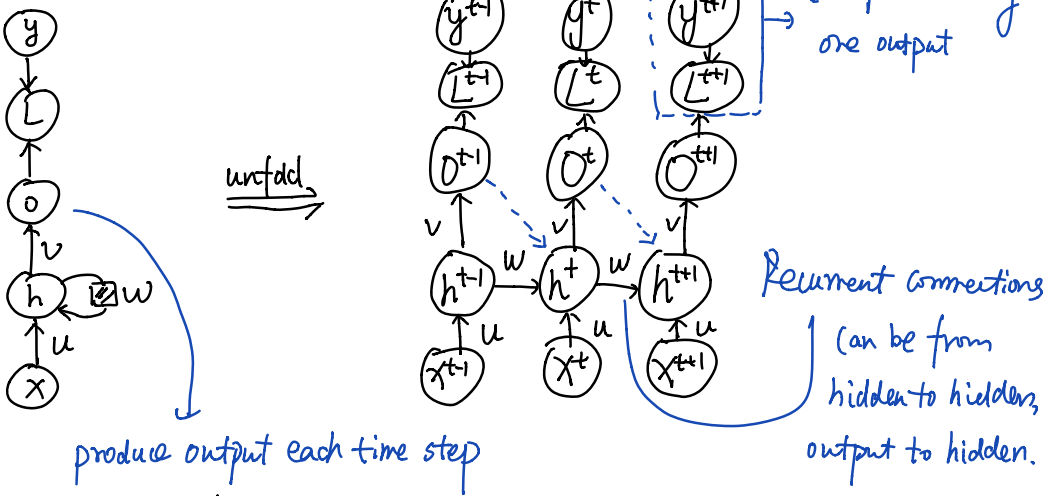
$$h^{(t)} = f(h^{(t-1)}, x^t, \theta)$$

The unfolding representation can clearly show the operations and shared weights, also can represent fixed length input and output.

- Here we can see that the operation at time step t, actually involves information from $1, \dots, t-1$, directly and indirectly.

Recurrent Neural Networks

Design patterns



- Running forward and computing the loss

Forward Operation:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$
$$h^{(t)} = \tanh(a^{(t)})$$
$$o^{(t)} = c + Vh^{(t)}$$
$$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$$

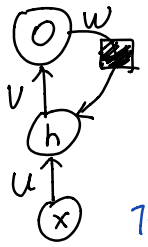
$$L(\{x^{(1)}, \dots, x^{(t)}\}, \{y^{(1)}, \dots, y^{(t)}\}) = \sum_t L^{(t)} = -\sum_t \log P_{\text{model}}(y^{(t)} \mid x^{(1)}, \dots, x^{(t)})$$

Negative log likelihood

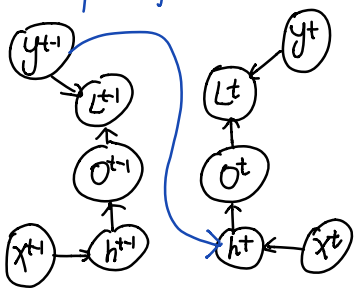
- Teacher Forcing and Networks with Output Recurrence.

Focus on training and analysis a specific RNN model structure.

The output of time step t-1 is one of the input of $h^{(t)}$



Inferencing



(The ground truth $y^{(t-1)}$ in training set is used for training)

Training

- Limits of teacher forcing
  - Low ability of representation (h^t need to remember all the past)
  - Low perform with training and testing difference is big.

- Computing Gradient of an RNN

computing the gradient of hidden units:

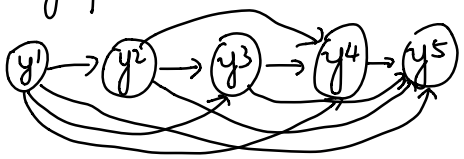
$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^T (\nabla_{h^{(t+1)}} L) + \left(\frac{\partial O^{(t)}}{\partial h^{(t)}}\right)^T \nabla_{O^{(t)}} L$$

pass to $h^{(t+1)}$, influencing $L^{(t+1)}$ — self to $L^{(t)}$

other gradients on W, V, U, b, c, can be computed based on this.

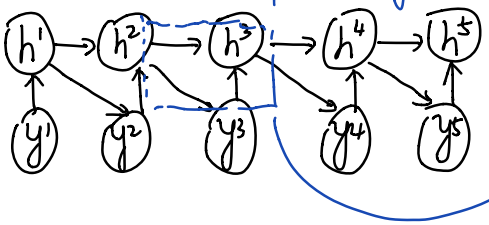
- Recurrent Networks as Directed Graphical Models.

A directed graphical model with no recurrent hidden units:



Too many parameters and dependencies

↓ After adding recurrent hidden units



This part of weights are shared

- Determining the output length
  - Definition a stop symbol
  - Each output associated with a binary variable indicating end or not.
  - Output an integer "t" of the length of sequence.
- Modeling Sequence Conditioned on Contexts with RNNs
  - Fixed length non-sequential input, output a sequence describing input, we input with an encoding / embedding style to the hidden units and recurrent use output and hidden units to next output / hidden timestep.
  - Sequential fixed length input and output, each hidden unit receives an input x, and output a y, y can also be as input to next-step hidden units for more representative information (not teacher forcing, cause we have hidden to hidden connection)

Bidirectional RNN

Adding backward connection from $g^{t+1} \rightarrow g^t$, namely considering another information flow from x^t to x^1 parrelly, each output is affected by both past and future.

Sequence to Sequence

- Encoder: An RNN inputing sequential x^t, and output a vector representation C, noticeable vector
- Decoder: As image caption, decoding the embedding vector C into to sequential output y.
- C: often called context, attention mechanism to solve the drawbacks of fixed length context set up.

Deep Recurrent Networks

W, U, V (connection between hidden, output and input) are best designed as Neural Networks

Recursive Neural Networks

Tree-structured, recursive operation and finally reaching solely one output.

The challenge of Long-Term Dependencies

$h^{(t)} = W h^{(t-1)}$

↳ $h^{(t)} = (W^t)^T h^{(0)} = Q^T \Lambda^t Q h^{(0)}$ where the exploding / vanishing occurs.

Echo State Networks

- Spectral Radius:

  Jacobian J, $\lambda = \max(\lambda_i)$, where λ is eigen-values of Jacobian.

  when going backward or forward, the value of spectral radius of J, reveals the variation rate of updating steps.

- Echo state

  Control spectral radius in a reasonable value when going back/forward.

Strategies for multiple time scales

- Adding skip connections / Removing connections then adding skip connections
- Leaky units controlling how much information will preserve from the past.

$$\mu^{(t)} \leftarrow \alpha \mu^{(t-1)} + (1-\alpha) v^{(t)}$$

LSTM and other Gated RNN (GRU)

One principle: Adding gate between input to hidden, hidden to hidden and hidden to output.

LSTM:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{ij}^f x_j + \sum_j W_{ij}^f h_j^{(t-1)}\right)$$ (Input / Output gate follow the same logic consisting x and h)

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{ij} x_j + \sum_j W_{ij} h_j^{(t-1)}\right)$$

GRU:

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + \left(1 - u_i^{(t-1)}\right) \sigma\left(b_i + \sum_j U_{ij} x_j^{(t)} + \sum_j W_{ij} r_j^{(t)} h_j^{(t-1)}\right)$$

u: update gate; r: reset gate

Optimizing for gradient exploding / Cliff error surface

second-order method (Newton method) can eliminate, but replacable by other models with lower cost.

- Clipping gradients

  if $\|g\| > v$, then $g \leftarrow \frac{gv}{\|g\|}$

- Regularizing to Encourage Information Flow

  Adding regularizing term $\Omega = \sum_t \left(\frac{\|\nabla_{h^{(t)}} L \frac{\partial h^{(t)}}{\partial h^{(t-1)}}\|}{\|\nabla_{h^{(t)}} L\|} - 1 \right)^2$

Explicit Memory

A task networks to control co-efficient of different sub-modules (memory cells).

Recurrent model based