Recurrent Neural Networks (RNN)

```
KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Some slides from Richard Socher, Geoffrey Hinton, Andrej Karpathy

Motivating Example: Language Models

- Want to assign probability to a sentence
  - “Dafjdkf adkjfhalj fadlag dfah” – zero probability
  - “Furiously sleep ideas green colorless” – very low probability
  - “Colorless green ideas sleep furiously” – slightly higher
  - “The quick brown fox jumped over the lazy dog” – even higher

https://en.wikipedia.org/wiki/Colorless_green_ideas_sleep_furiously

Application for Language Models

- Applications
  - OCR gives several hypotheses, need to choose the most probable one
  - Choose a plausible translation from English to French
  - Complete the sentence “A core objective of a learner is to generalize from its […]”
- In every case, a language model can be used to evaluate all the possible hypotheses, and select the one with the highest probability

Sentence Completion

- Suppose a language model M can compute
$$P_M(w_1, w_2, \dots, w_k)$$
- For an incomplete sentence $w_1\ w_2\ w_3 \dots w_{k-1}$, find $argmax_{w_k} P(w_1, w_2, w_3, \dots, w_k)$ to complete the sentence
- Now, fix w_k, and find $argmax_{w_{k+1}} P(w_1, w_2, w_3, \dots, w_k, w_{k+1})$

Probabilistic Sentence Generation

- $P(w_k|w_1, w_2, \ldots w_{k-1}) = \frac{P(w_1 \ldots w_{k-1} w_k)}{P(w_1 \ldots w_{k-1})} \propto P(w_1 \ \ldots \ w_{k-1}\, w_k)$
- Choose word $w^{(j)}$ according to

$$\frac{\exp(\alpha\, \hat{P}(w_1 w_2 \ldots w_{k-1}\, w^{(j)}))}{\sum_j \ \exp(\alpha \hat{P}(w_1 w_2 \ldots w_{k-1}\, w^{(j)}))}$$

- (Question: Higher α => ?)
- Generally, take $\hat{P}$ to be the input to the softmax that produces the probability in the RNN (better), or simply the probability of the sentence

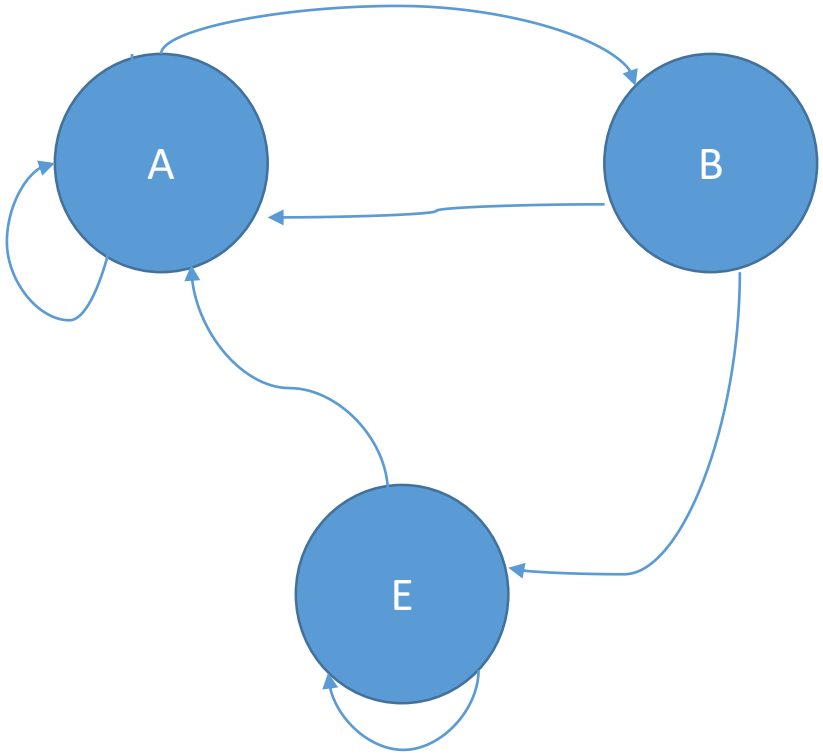
Generating “Shakespeare” character-by-character with RNN

```
KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

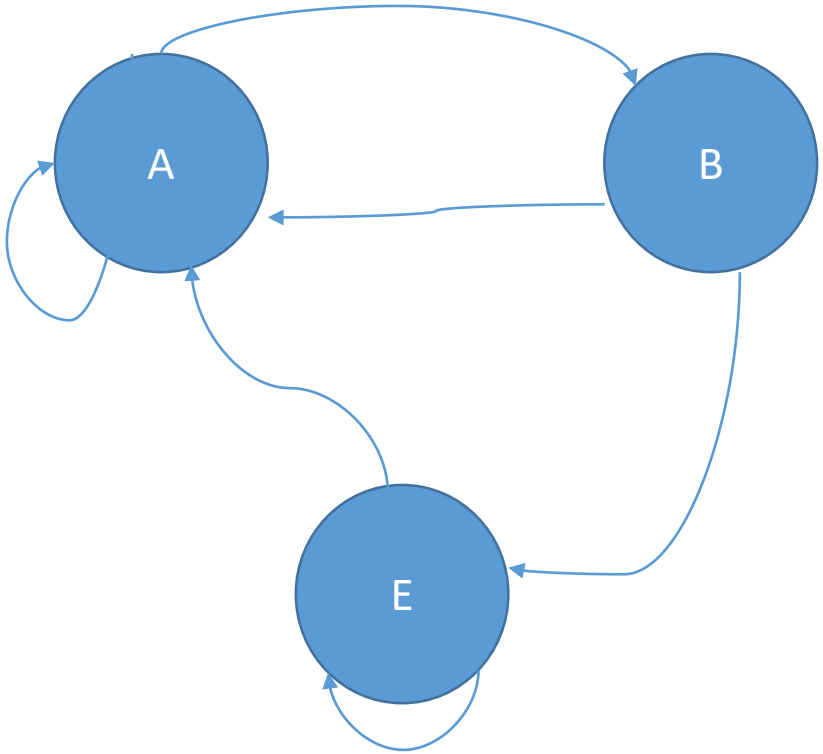
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

(Here, the w’s are *characters*, not words)

Generating Strings with Finite State Machines



Generating Strings with Finite State Machines

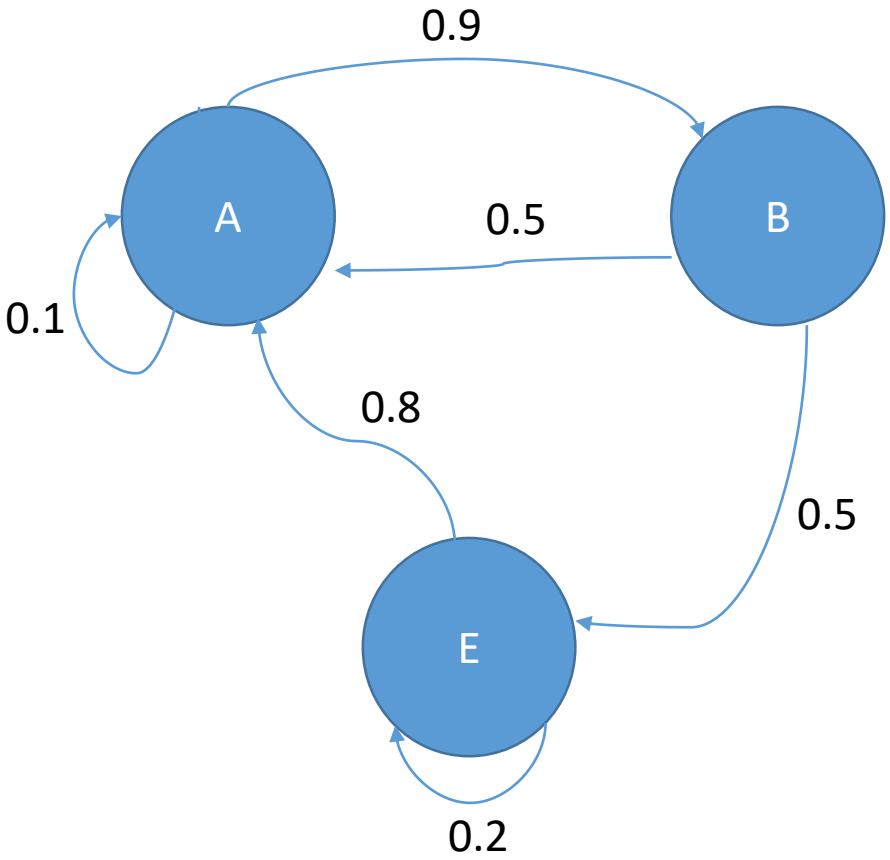


ABAAEEAAEABEEEEEAAABEEAAB – Not allowed

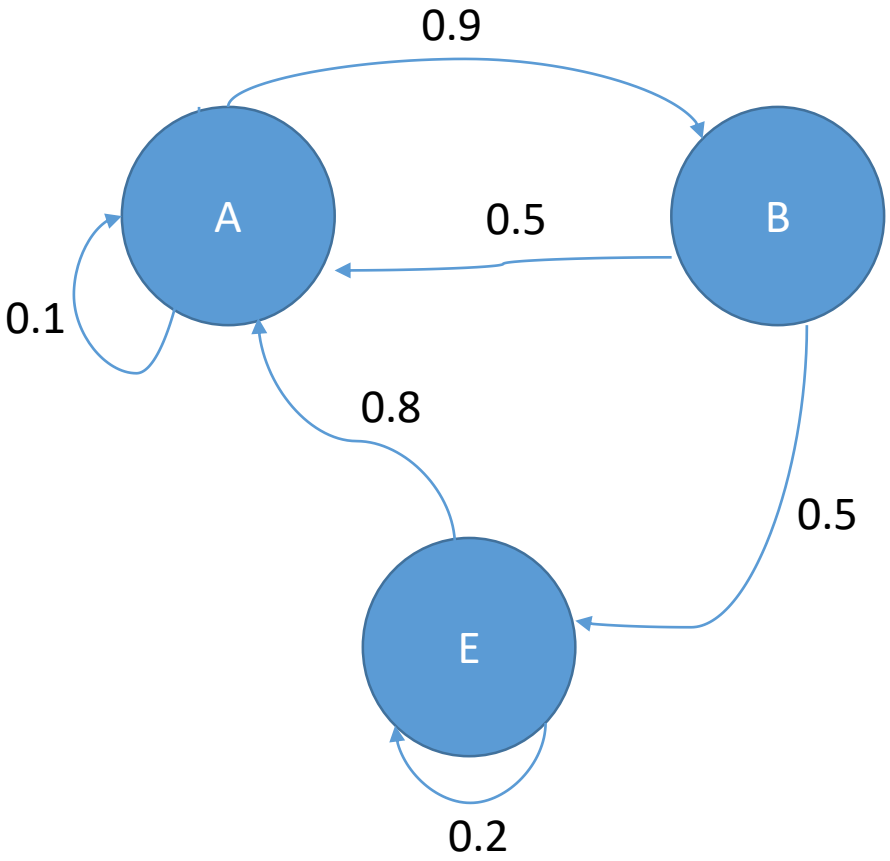
AABEA – Allowed

ABBA – Not allowed

Markov Models: Probabilistic Finite State Machines



Markov Models: Probabilistic Finite State Machines



$$P(w_k = A | w_{k-1} = E) = 0.8$$

$$P(w_k = B | w_{k-1} = B) = 0$$

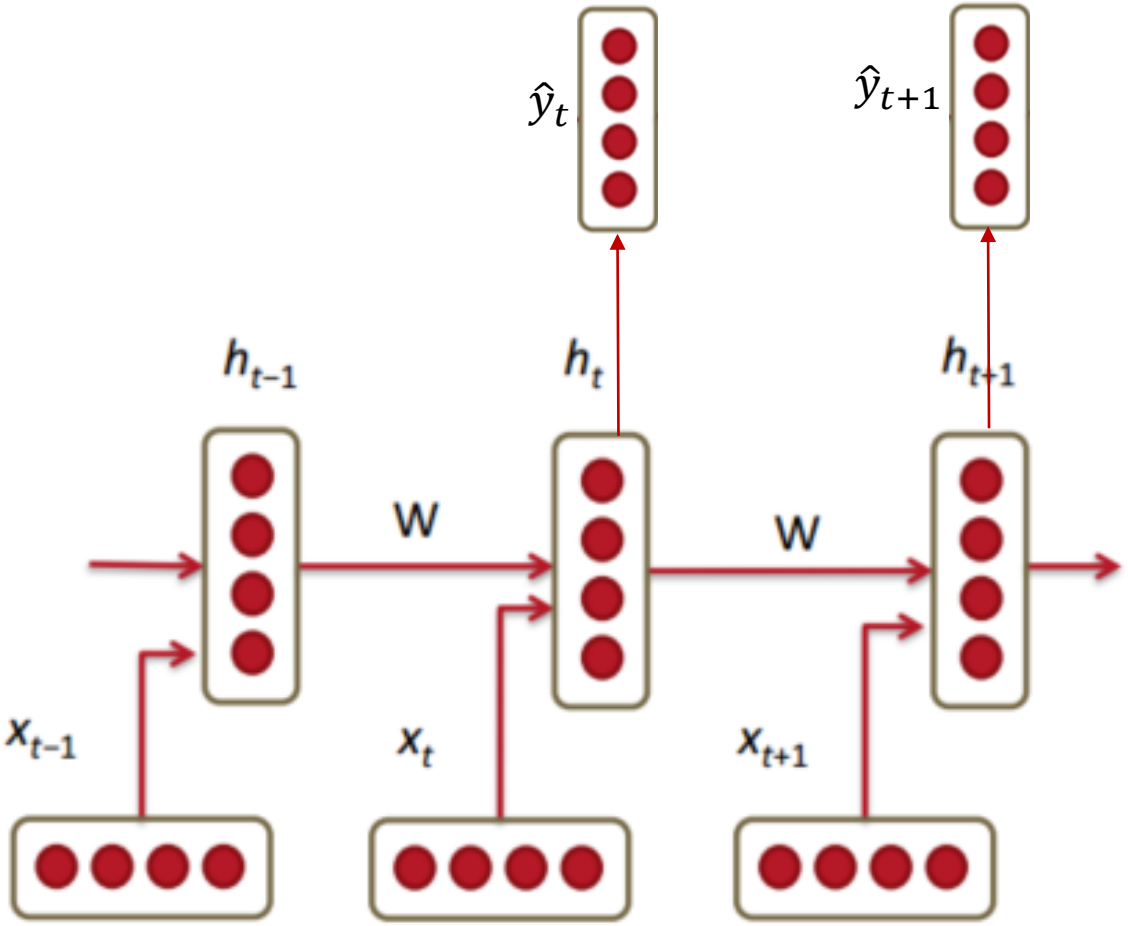
A More Complicated FSM

- Suppose our alphabet is {“a”, “b”, “c”, “e”, “i”}
- Encode the rule:

  “i” before “e,” except after “c”

  (think “receive,” “believe,” “deceit”…)

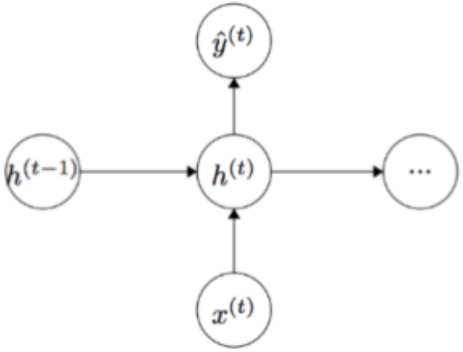
Recurrent Neural Networks

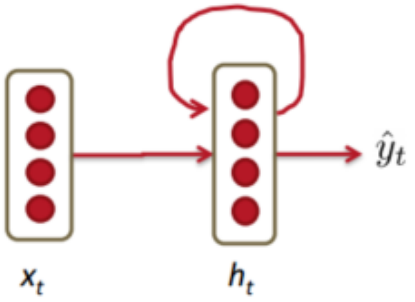


x_t-the t[th] character of the string ("the character at time t")
h_t-the t[th] character of the string ("the character at time t")

RNN for Language Modelling

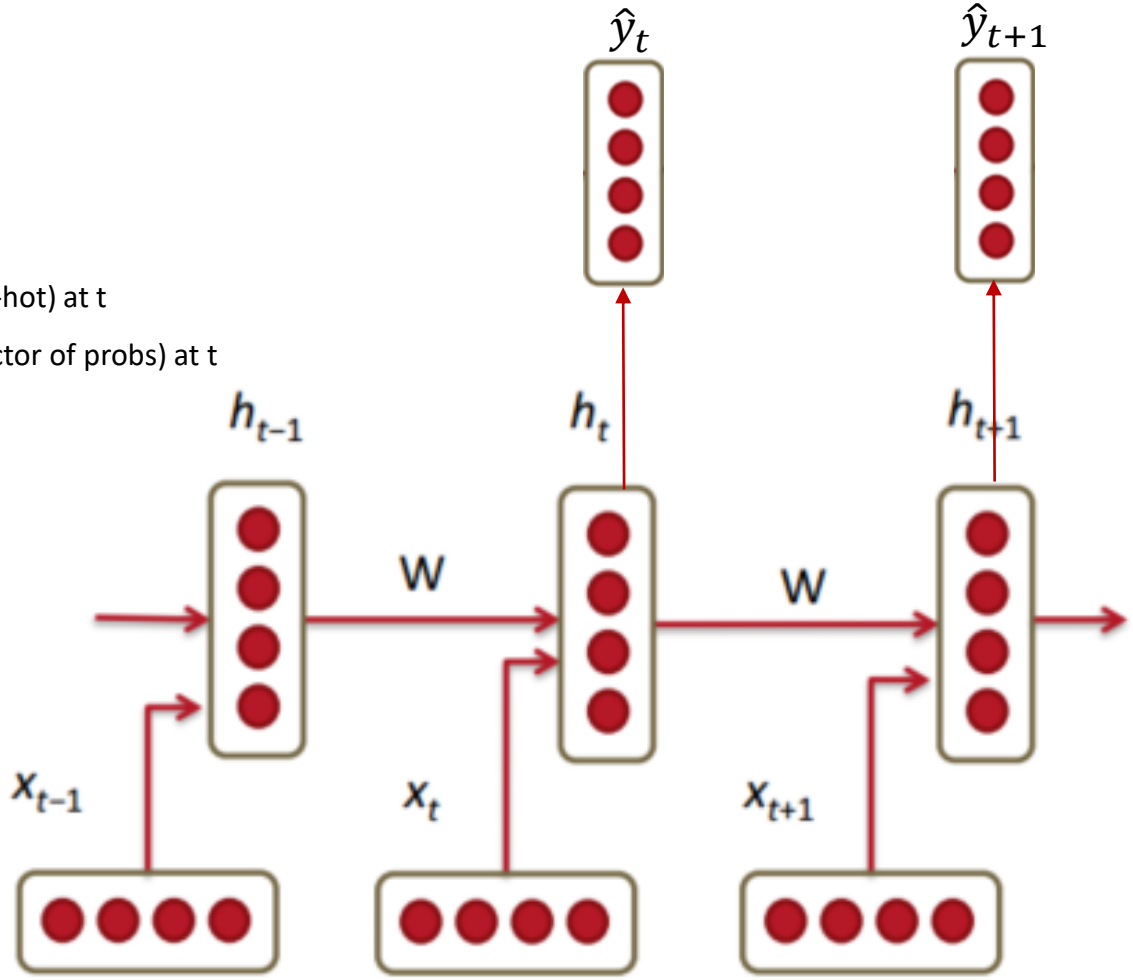
- Given a list of word vectors (e.g., one-hot encodings of words) $x_1, \dots, x_{t-1}, x_t, x_{t+1}, \dots, x_T$
  At a single time step:
  - $h_t = \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$
  - $\hat{y}_t = softmax(W^{(S)} h_t)$
  - $\hat{P}\left(x_{t+1} = v_j \middle| x_1, x_2, \dots, x_t\right) = \hat{y}_{t,j}$
- h is the *state* (e.g., the previous word vector could be part of h)
- x_t is the data
- $\hat{y}_t$ is the predicted output





x_t – the input (one-hot) at t

$\hat{y}_t$ - predictions (vector of probs) at t



$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right)$$
$$\hat{y}_t = softmax\left(W^{(S)}h_t\right)$$
$$\hat{P}\left(x_{t+1} = v_j \middle| x_1, x_2, \dots, x_t\right) = \hat{y}_{t,j}$$

Cost Function

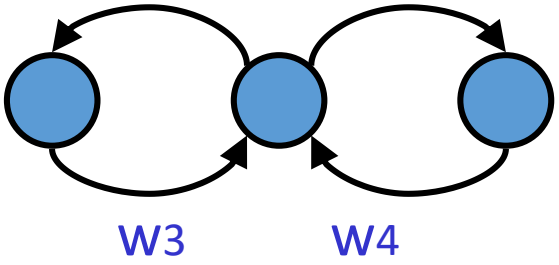
- Same as before: negative log-probability of the right answer:

$$J^{(t)} = -\sum_{j=1}^{V} y_{t,j} \log \hat{y}_{t,j}$$

$$J = \sum_{t} J^{(t)}$$

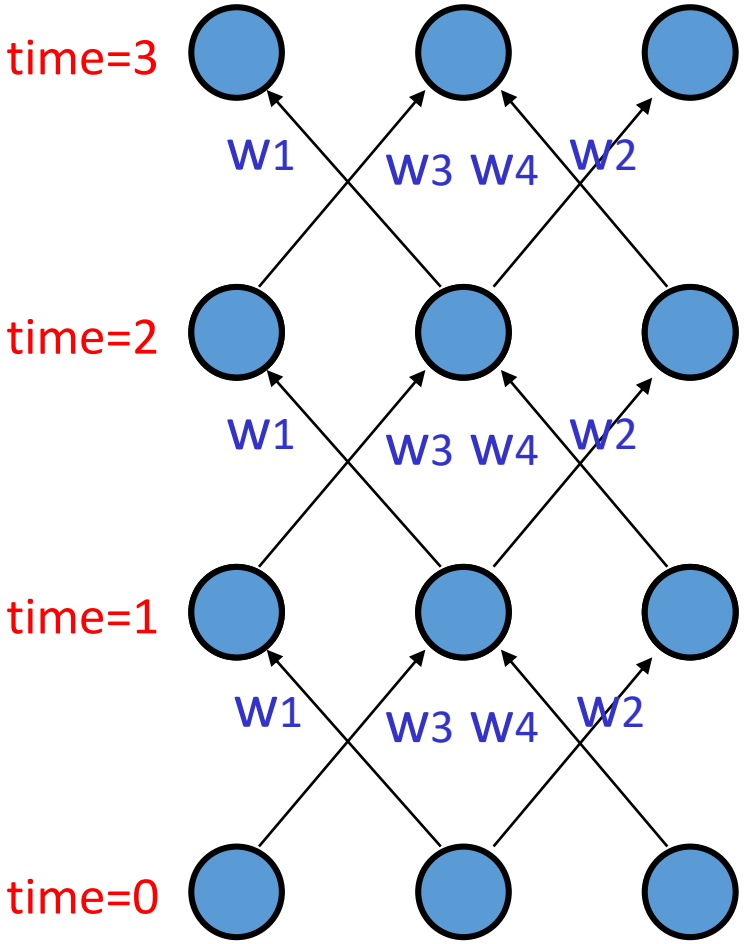
- $\hat{y}_{t,j} = 1$ iff $x_{t+1} = v_j$

RNN "=" feedforward net with shared weights

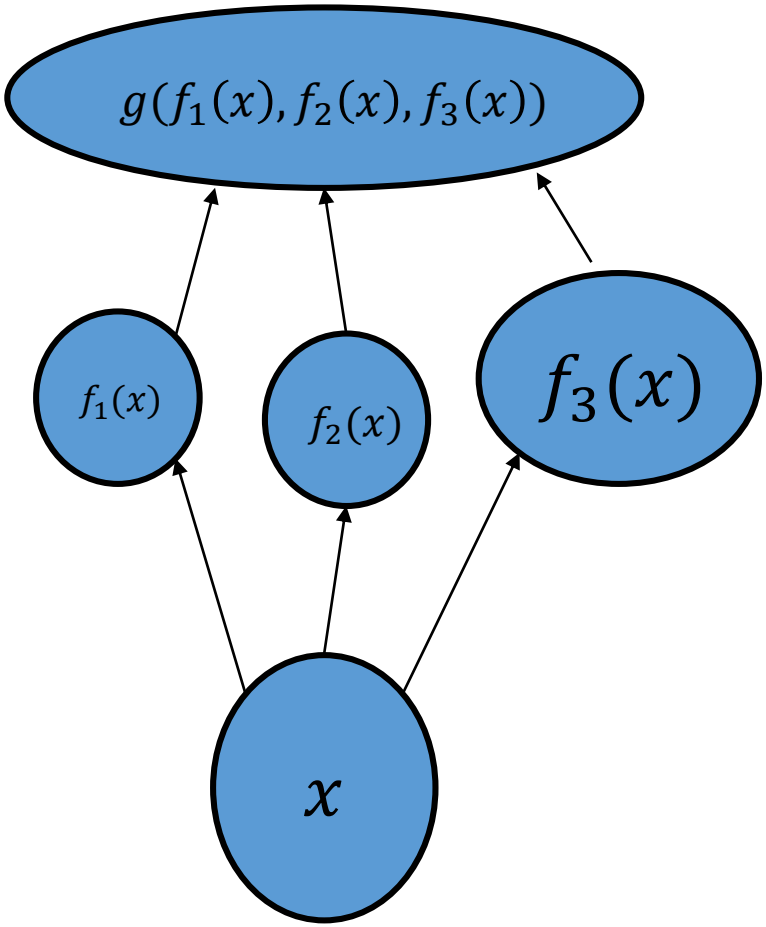


Assume that there is a time delay of 1 in using each connection.

The recurrent net is just a layered net that keeps reusing the same weights.



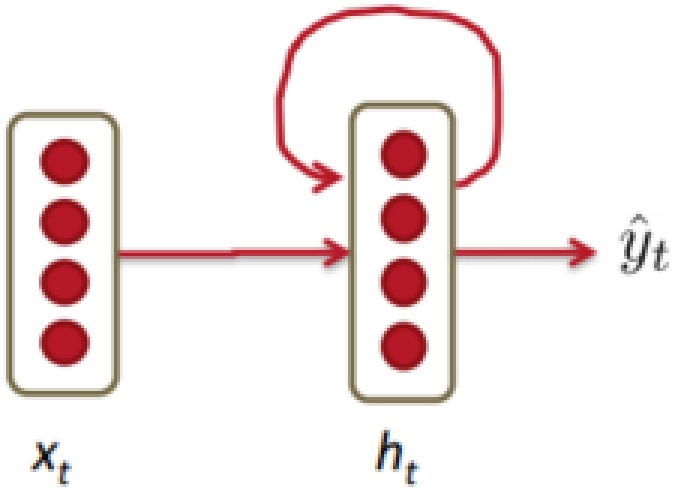
Reminder: Multivariate Chain Rule



$$\frac{\partial g}{\partial x} = \sum \frac{\partial g}{\partial f_i} \frac{\partial f_i}{\partial x}$$

Gradient

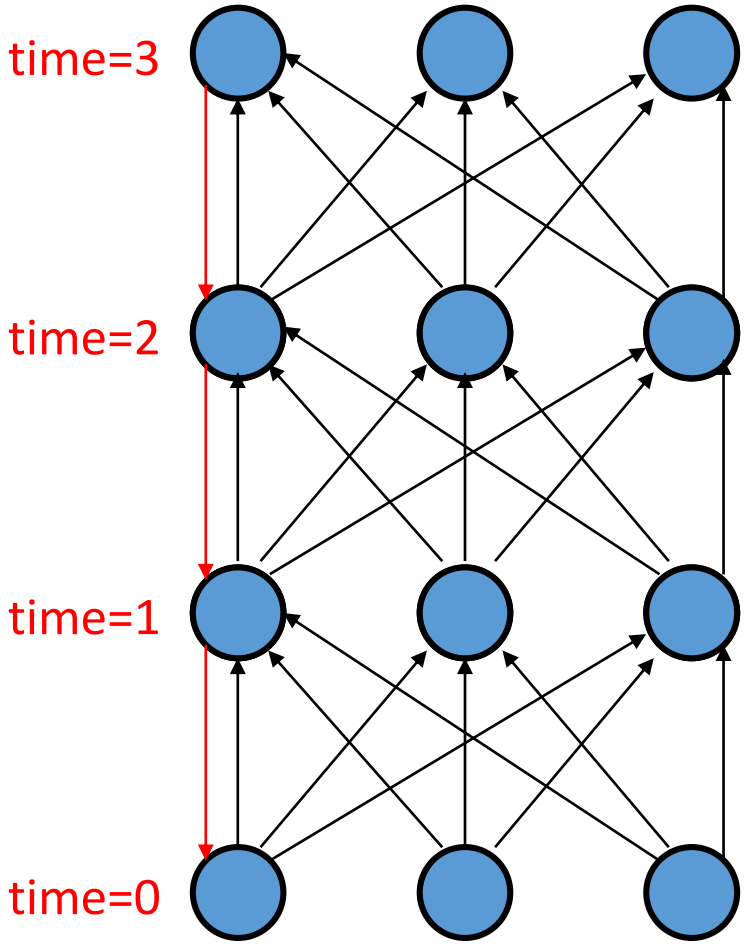
- $\frac{\partial J}{\partial W} = \sum_t \frac{\partial J^{(t)}}{\partial W}$
- $\frac{\partial J^{(t)}}{\partial W} = \sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial h_k} \frac{\partial h_k}{\partial W} = \sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$



RNN Gradient

- $\frac{\partial J}{\partial W} = \sum_t \frac{\partial J^{(t)}}{\partial W}$
- $\frac{\partial J^{(t)}}{\partial W} = \sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial h_k} \frac{\partial h_k}{\partial W} =$
  $\sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial \hat{y}_t} \frac{\hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$

$\hat{y}_t = h_t$

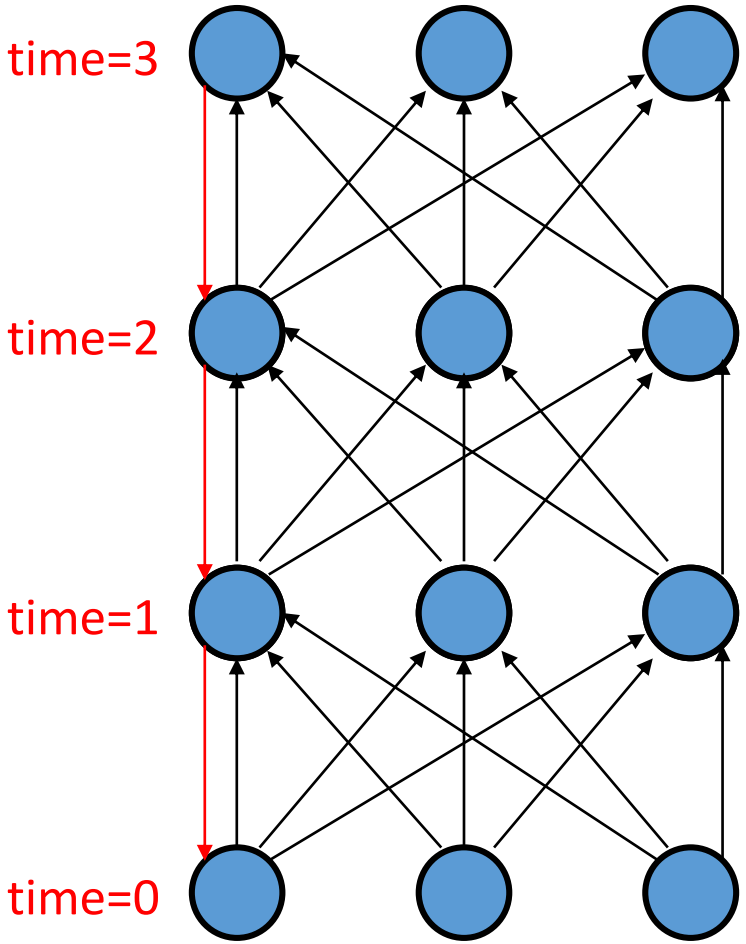


Vanishing and Exploding Gradient

- $\frac{\partial J^{(t)}}{\partial W} = \sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial h_k} \frac{\partial h_k}{\partial W} = \sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$
- $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$
- Assume 1-d h's
  - In fact, they are vectors
- Often, $\left|\frac{\partial h_j}{\partial h_{j-1}}\right| < 1$ for all j or $\left|\frac{\partial h_j}{\partial h_{j-1}}\right| > 1$ for all j
  - They are all related by the same weight to each other
- This means that $\frac{\partial h_t}{\partial h_k}$ is either very close to 0 (vanishing grad.) or very large (exploding grad.) for large $|t-k|$

Vanishing Gradient is a Problem

Problem: if the gradient vanishes, we can't figure out that the weight needs to be changed because of what's happening at time=0 to make the cost function at time=n smaller



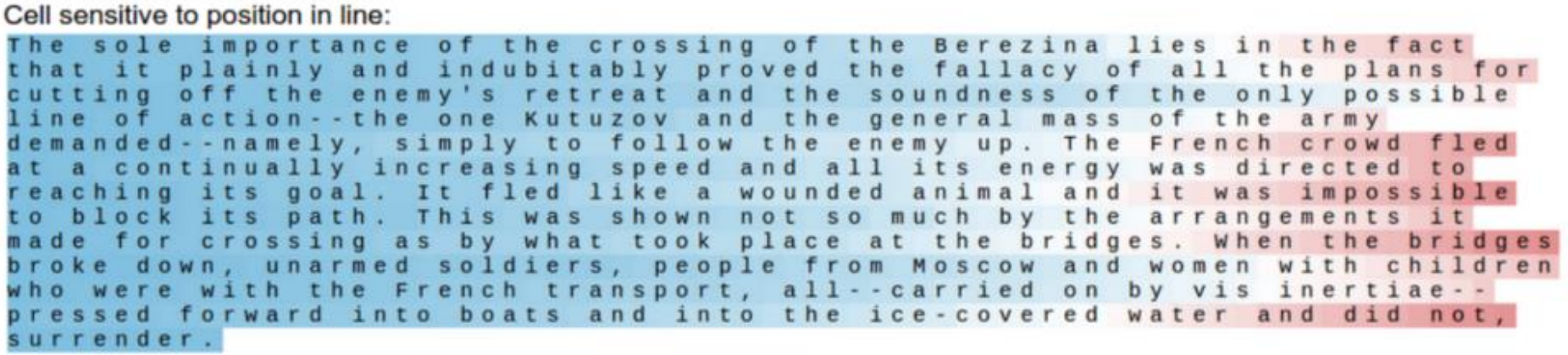
Exploding Gradient is a Problem

- Why?
- “Hacky” solution: clip the gradients

Vanishing Gradient in Language Models

- In the case of language modeling or question answering words from steps far away are not taken into consideration when training to predict the next word
- Example:
  Jane walked into the room. John walked in too. It was late in the day. Jane said hi to ____

Visualizing the hidden state*



Karpathy et al. “Visualizing and Understanding Recurrent Networks”
http://arxiv.org/abs/1506.02078

*The RNN there is somewhat more complicated than what we saw so far

Cell that turns on inside quotes:

Karpathy et al. “Visualizing and Understanding Recurrent Networks”
http://arxiv.org/abs/1506.02078

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

Karpathy et al. “Visualizing and Understanding Recurrent Networks”
http://arxiv.org/abs/1506.02078