COMP4901K/Math4824B
Machine Learning for Natural Language Processing

Lecture 13: Neural Language Models

Instructor: Yangqiu Song

Slides credits: Piotr Mirowski, Noah Smith, Dan Roth, and Chris Manning

Recap: what is a statistical LM?

- A model specifying probability distribution over word sequences
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001
- It can be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

Probabilistic Language Models

- Probability of a sequence of words:

$$P(w_1 \ \dots \ w_N)$$

- Chain rule of probability:

$$P(w_1 \, w_2 \ \dots wN) = P(w_1)p(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_N|w_1, w_2, \dots, w_{N-1})$$

- (n-1)$^{\text{th}}$ order Markov assumption

$$P(w_i|w_1, w_2, \dots, w_{i-1}) = P(w_i|w_{i-1}, \dots, w_{i-n+1})$$

Learning probabilistic language models

- Learn joint likelihood of training sentences under (n-1)th order Markov assumption using n-grams

$$P(w_1\, w_2\ \dots wN) = \prod_{i=1}^{N} P(w_i | w_{i-1}, \dots, w_{i-n+1}) = \prod_{i=1}^{N} P(x_i | \boldsymbol{h}_i)$$

where w_i=x_i(word token w_i, word type x_i in vocabulary)

$\boldsymbol{h}_i = w_{i-1}, \dots, w_{i-n+1}$ is word history

- Maximize the log-likelihood: $\prod_{i=1}^{N} P(x_i | \boldsymbol{h}_i)$
  - Now, given the above reformulation, we will change the notations again (to derive neural language models)!

Featurized Language Models: Re-parameteriazation

- Maximize the log-likelihood: $\prod_{i=1}^{N} P(x_i|\boldsymbol{h}_i)$
- Assuming a parametric model **w** (note here **w** is the parameter vector similar use as perceptron)

$$\prod_{i=1}^{N} P(x_i|\boldsymbol{h}_i) \equiv \prod_{i=1}^{N} P_{\boldsymbol{w}}(x_i|\boldsymbol{h}_i)$$

- Consider $\boldsymbol{h}_i$ as features instead of just a sequences of historical words
  - Modeling with log-linear models
  - Moving from generative models to discriminative models

Log-linear Models

$$P(x_i|\boldsymbol{h}_i, \boldsymbol{w}_i) = \frac{\exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i)}{\sum_{x_i} \exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i)}$$

- Linear score $\boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i)$
- Nonnegative exponential: $\exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i)$
- Normalizer $\sum_{x_i} \exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i) \equiv Z_{\boldsymbol{w}}(\boldsymbol{h}_i)$
- Log-linear comes from the fact that

$$\log P(x_i|\boldsymbol{h}_i, \boldsymbol{w}_i) = \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i) - \log Z_{\boldsymbol{w}}(\boldsymbol{h}_i)$$

$\log Z_{\boldsymbol{w}}(\boldsymbol{h}_i)$ is a constant in x_i

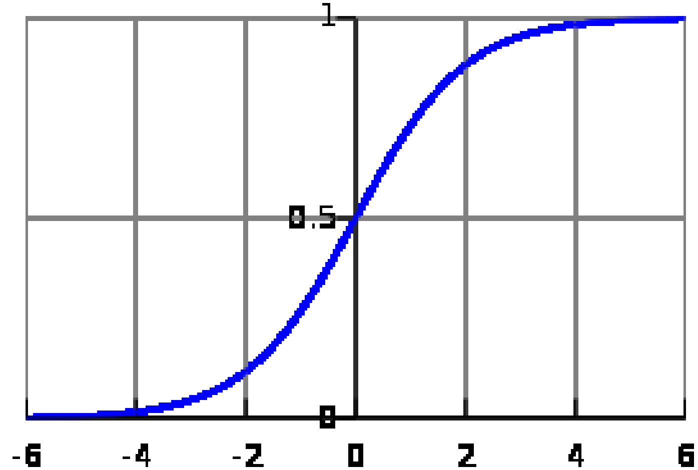
- This is an instance of the family of generalized linear models

Special Case: Logistic Regression

- Consider the case where $x \in \{+1,-1\}$

$$P(x = 1|\boldsymbol{h}, \boldsymbol{w})$$
$$= \frac{\exp \boldsymbol{w}^T \phi(1, \boldsymbol{h})}{\exp \boldsymbol{w}^T \phi(1, \boldsymbol{h}) + \exp \boldsymbol{w}^T \phi(-1, \boldsymbol{h})}$$
$$= \frac{1}{1 + \exp[\boldsymbol{w}^T \phi(-1, \boldsymbol{h}) - \exp \boldsymbol{w}^T \phi(1, \boldsymbol{h})}$$
$$= \sigma(\boldsymbol{w}^T \phi(-1, \boldsymbol{h}) - \exp \boldsymbol{w}^T \phi(1, \boldsymbol{h}))$$
$$= \sigma(x \boldsymbol{w}^T f(\boldsymbol{h}))$$

  - where $\sigma(t) = \frac{1}{1+\exp(-t)}$



log-linear models are often called multinomial logistic regression (softmax function)

Special Case: N-gram Language Model

$$P(x_i|\boldsymbol{h}_i, \boldsymbol{w}_i) = \frac{\exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i)}{\sum_{x_i} \exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i)}$$

- Consider an n-gram language model
  - $\boldsymbol{h}_i = w_{i-1}, \dots, w_{i-n+1}$ as n-1 historical words
  - $\phi(x_i, \boldsymbol{h}_i) = \log c(x_i, \boldsymbol{h}_i)$
  - $\boldsymbol{w}_i = \mathbf{1}$ (all one vector for all x_i)
  - $\sum_{x_i} \exp \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h}_i) = \sum_{x_i} \exp \log c(x_i, \boldsymbol{h}_i) = \sum_{x_i} c(x_i, \boldsymbol{h}_i) = c(\boldsymbol{h}_i)$
- What features are there used in $\boldsymbol{\phi(x, h)}$ more than traditional n-gram language models?

What features in $\phi(x, \boldsymbol{h})$

	Saturday
	Sunday
	Monday
I visited Central last ______	*month*
	...
	pizza

- Traditional n-gram features: last ^ Saturday
- "Gappy" n-gram features: Central ^ Saturday
- Spelling features: first character is capitalized
- Class features: whether it is a member of class 132
- Gazetteer features: whether it is listed as a geographic place name, date/time, person name, organization name, etc.

What features in $\phi(x, \boldsymbol{h})$

- You can define any features you want!
  - Too many features, and your model will overt
    - "Feature selection" methods, e.g., ignoring features with very low counts, can help
  - Too few (good) features, and your model will not learn

Softmax Function

$$\langle x_1, x_2, \ldots, x_k \rangle \mapsto \left\langle \frac{e^{x_1}}{\sum_{j=1}^{k} e^{x_j}}, \frac{e^{x_2}}{\sum_{j=1}^{k} e^{x_j}}, \ldots, \frac{e^{x_k}}{\sum_{j=1}^{k} e^{x_j}} \right\rangle$$

- $x_i = \boldsymbol{w}_i{}^T \phi(x_i, \boldsymbol{h})$

```
>>> x = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
>>> softmax = lambda x:np.exp(x)/np.sum(np.exp(x))
>>> softmax(x) array([0.02364054, 0.06426166, 0.1746813 , 0.474833 ,
0.02364054, 0.06426166, 0.1746813 ])
```

Parameter Estimation

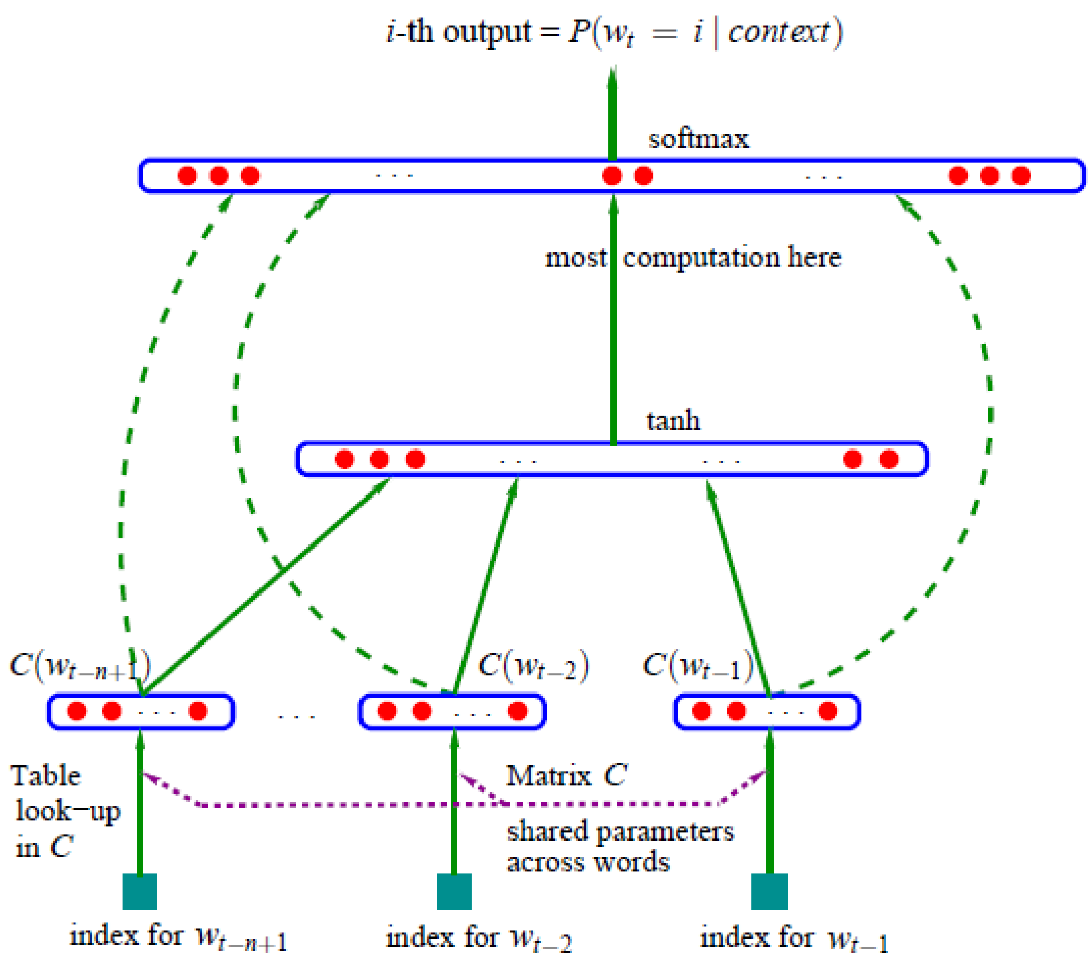
- Gradient descent!
  - no closed form as traditional n-gram language models

- Further Reading
  - Berger et al. (1996).A Maximum Entropy Approach to Natural Language Processing.
  - Collins (2011). Course notes for COMS w4705: Log-linear models, MEMMs, and CRFs, 2011.
    - http://www.cs.columbia.edu/~mcollins/crf.pdf
  - Smith (2004). Log-linear models, 2004.
    - https://homes.cs.washington.edu/~nasmith/papers/smith.tut04.pdf

Extension: Neural Language Models

- Feedforward Neural Network Language Model
  Bengio et al. (2003)
  - A generalization of featurized language model
  - Word embeddings can be learnt!

Feedforward Neural Network Language Model
Bengio et al. (2003)



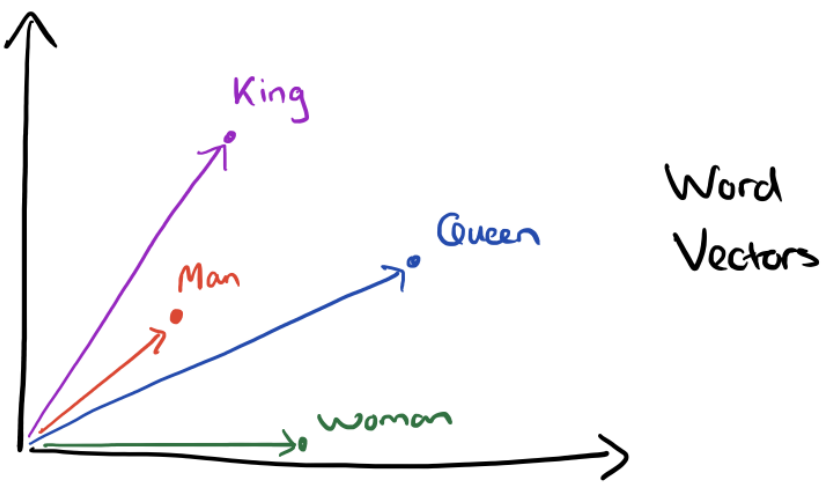
Results

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

Table 1: Comparative results on the Brown corpus. The deleted interpolation trigram has a test perplexity that is 33% above that of the neural network with the lowest validation perplexity. The difference is 24% in the case of the best n-gram (a class-based model with 500 word classes). n : order of the model. c : number of word classes in class-based n-grams. h : number of hidden units. m : number of word features for MLPs, number of classes for class-based n-grams. *direct*: whether there are direct connections from word features to outputs. *mix*: whether the output probabilities of the neural network are mixed with the output of the trigram (with a weight of 0.5 on each). The last three columns give perplexity on the training, validation and test sets.

Important Idea: Words as Vectors

- The idea of "embedding" words much older than neural language models.
- Deerwester et al. (1990) explored dimensionality reduction techniques for information retrieval-style querying of text collections
- We will come back to this topic



Indexing by latent semantic analysis (1990) by Scott Deerwester , Susan T. Dumais , George W. Furnas , Thomas K. Landauer , Richard Harshman

Parameter Estimation

- Good news:
  - The whole computation is differentiable with respect to parameters, so gradient-based methods are available
  - Lots more details in Bengio et al. (2003)
- Bad news for neural language models (in 2003):
  - Calculating log-likelihood and its gradient is very expensive (5 epochs took 3 weeks on 40 CPUs, in Bengio et al. (2003))

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. J. Mach. Learn. Res. 3 (March 2003), 1137-1155.

Observations about Neural Language Models (So Far)

- There's no knowledge built in that the most recent word should generally be more informative than earlier ones
  - This has to be learned

- In addition to choosing (n-1)gram historical words, also have to choose dimensionalities like d and H

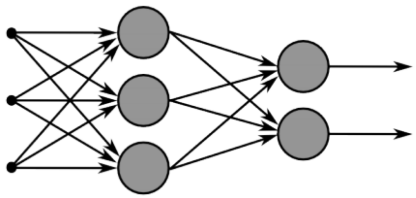
- Parameters of these models are hard to interpret
  - Individual word embeddings can be clustered and dimensions can be analyzed (e.g., Tsvetkov et al. (2015))

- Still, impressive perplexity gains got people's interest

Recurrent Neural Networks

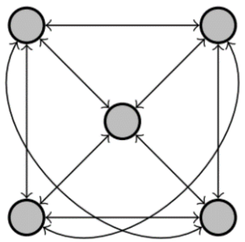
- General ideas
  - Can we use more historical words?
  - Can parameters be shared?

Recurrent Neural Networks



- Multi-layer feed-forward NN: DAG
  - Just computes a fixed sequence of

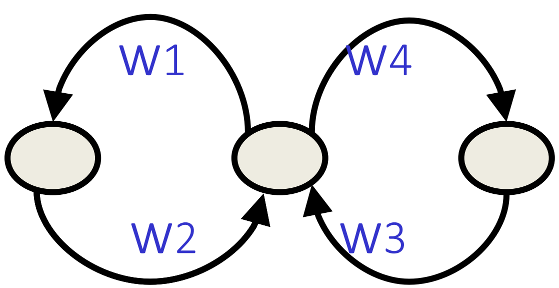
  non-linear learned transformations to convert an input patter into an output pattern

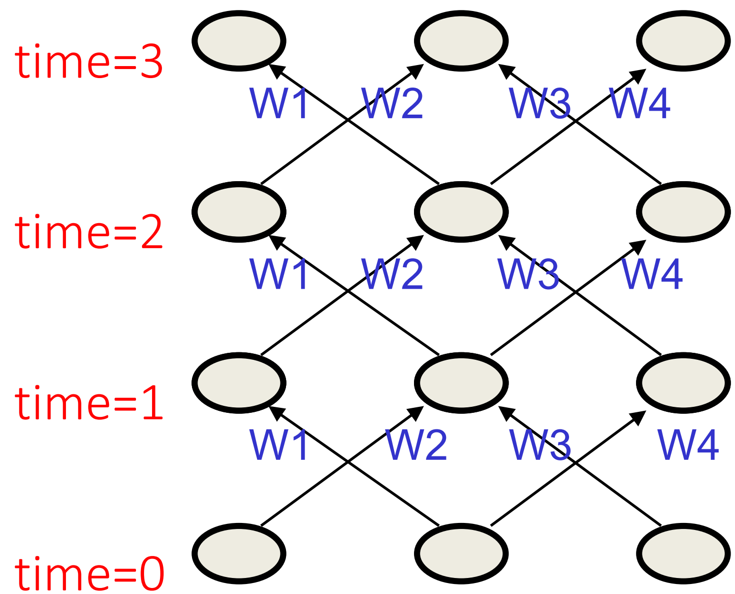


- Recurrent Neural Network: Digraph
  - Has cycles.
  - Cycle can act as a memory;
  - The hidden state of a recurrent net can carry along information about a "potentially" unbounded number of previous inputs.
  - They can model sequential data in a much more natural way.

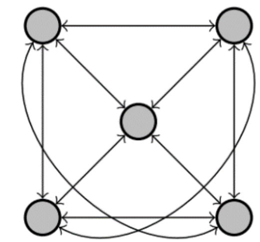
Equivalence between RNN and Feed-forward NN

- 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.





Recurrent Neural Networks



- Training a general RNN's can be hard
  - Here we will focus on a **special family of RNN's**
- Prediction on chain-like input:
  - Language model

$X, h =$	This	is	a	sample	sentence	.
$Y =$	is	a	sample	sentence	.	<EOS>

  - POS tagging words of a sentence

$X =$	This	is	a	sample	sentence	.
$Y =$	DT	VBZ	DT	NN	NN	.

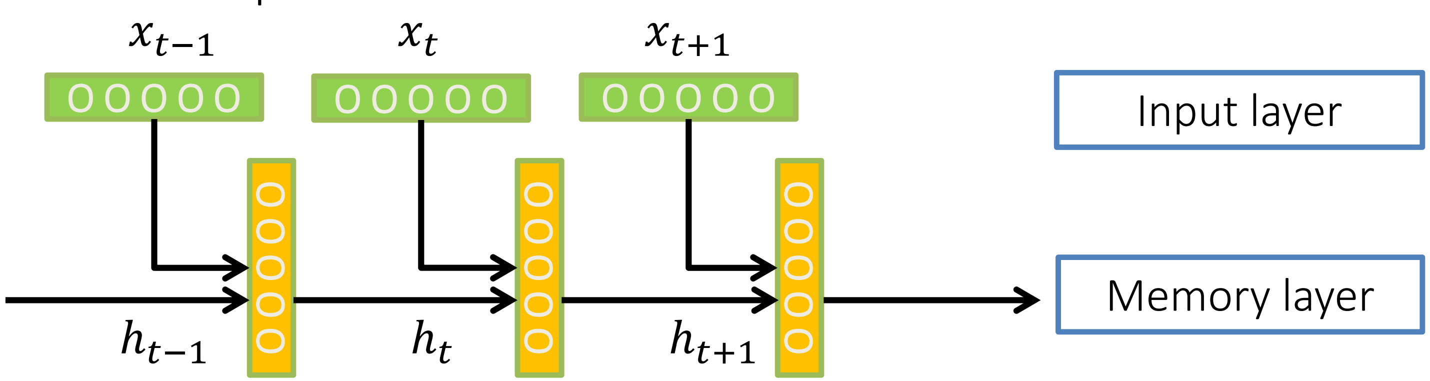
  - Sentiment classification

$X =$	This	is	a	sample	sentence	.

$Y =$ Positive/Negative

Recurrent Neural Networks

- A chain RNN:
  - Has a chain-like structure
  - Each input is replaced with its vector representation x_t
  - Hidden (memory) unit h_t contain information about previous inputs and previous hidden units h_{t-1}, h_{t-2}, etc
    - Computed from the past memory and current word. It summarizes the sentence up to that time.

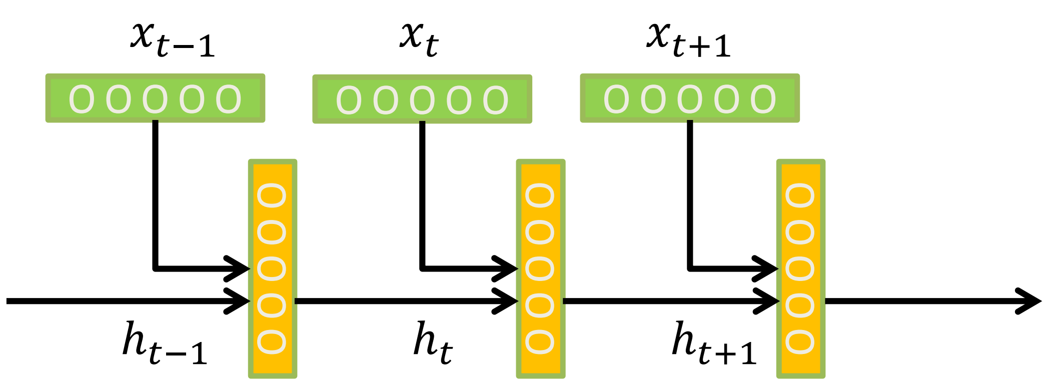


Recurrent Neural Networks

- A popular way of formalizing it:

$$h_t = f(W_h h_{t-1} + W_i x_t)$$

  - Where f is a nonlinear, differentiable (why?) function.
- Outputs?
  - Many options; depending on problem and computational resource



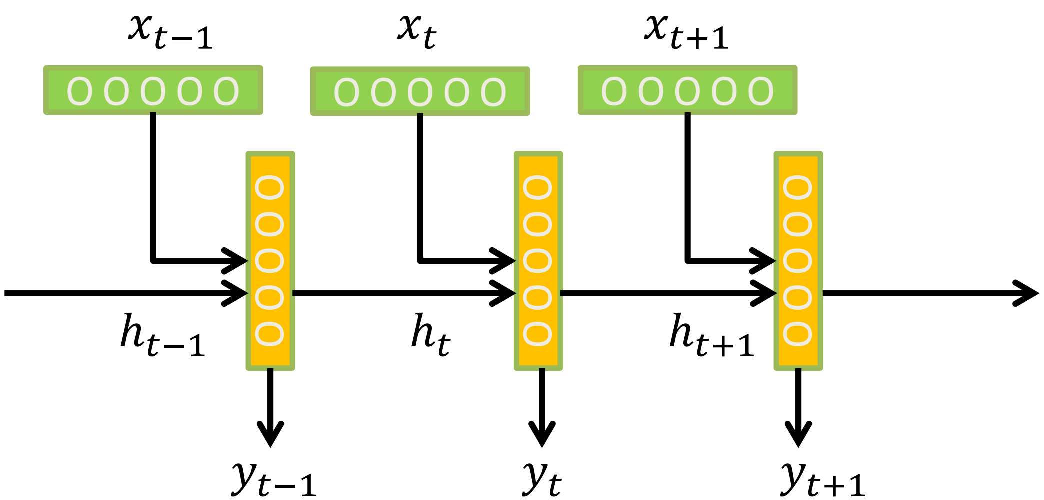
Recurrent Neural Networks

- Prediction for x_t, with h_t
- Prediction for x_t, with $h_t, \dots, h_{t-\tau}$
- Prediction for the whole chain

$$y_t = \mathrm{softmax}(W_o h_t)$$

$$y_t = \mathrm{softmax}\left(\sum_{i=0}^{\tau} \alpha^i W_o^{t-i} h_{t-i}\right)$$

$$y_T = \mathrm{softmax}(W_o h_T)$$



Training RNNs

- How to train such model?
  - Generalize the same ideas from back-propagation
- Total output error: $E(\vec{y}, \vec{t}) = \sum_{t=1}^{T} E_t(y_t, t_t)$

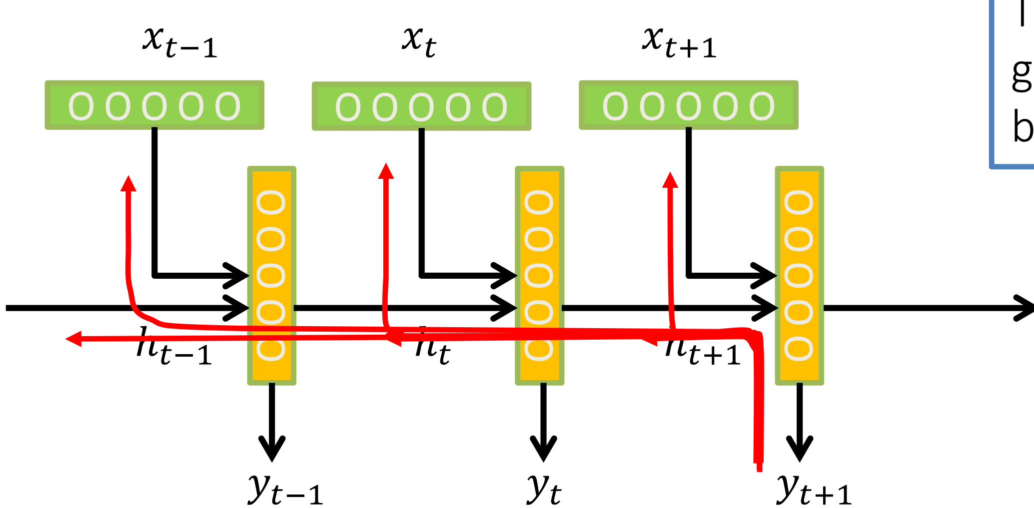
Reminder:
$y_t = \text{softmax}(W_o h_t)$
$h_t = f(W_h h_{t-1} + W_i x_t)$

Parameters?
W_o, W_i, W_h + vectors for input

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_{t-k}}} \frac{\partial h_{t-k}}{\partial W}$$

This sometimes is called "Backpropagation Through Time", since the gradients are propagated back through time.



Backpropagation for RNN

Recurrent Neural Network

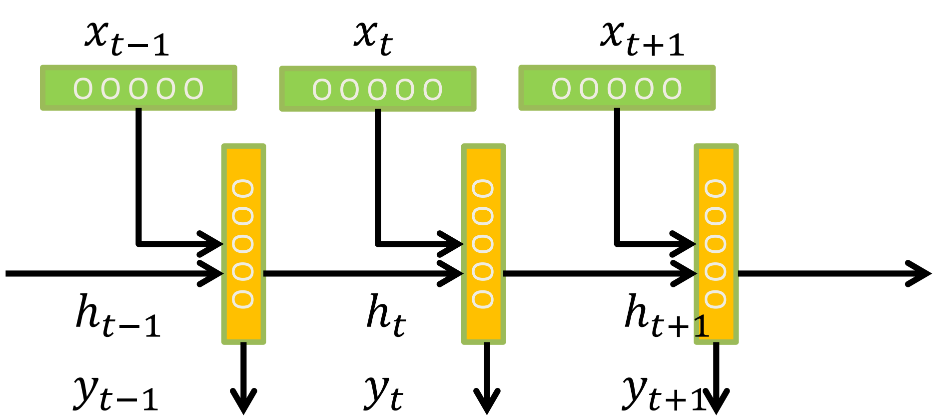
$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_{t-k}}} \frac{\partial h_{t-k}}{\partial W}$$

Reminder:
$y_t = \text{softmax}(W_o h_t)$
$h_t = f(W_h h_{t-1} + W_i x_t)$

$$\frac{\partial h_t}{\partial h_{t-1}} = W_h \text{diag}[f'(W_h h_{t-1} + W_i x_t)]$$

$$\text{diag}[a_1, \dots, a_n] = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_n \end{bmatrix}$$

$$\frac{\partial h_t}{\partial h_{t-k}} = \prod_{j=t-k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=t-k+1}^{t} W_h \text{diag}[f'(W_h h_{t-1} + W_i x_t)]$$



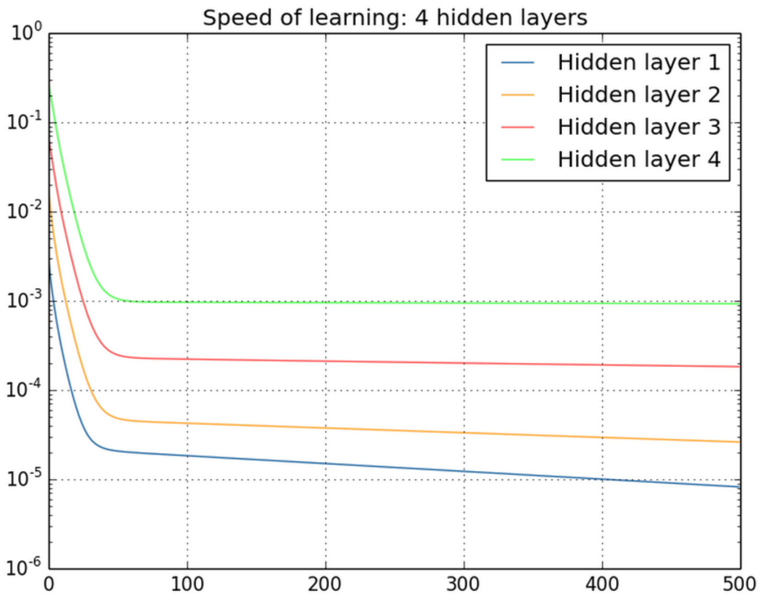
Vanishing/exploding gradients

$$\frac{\partial h_t}{\partial h_{t-k}} = \prod_{j=t-k+1}^{t} W_h \text{diag}[f'(W_h h_{t-1} + W_i x_t)]$$

$$||\frac{\partial h_t}{\partial h_k}|| \leq \prod_{j=t-k+1}^{t} \|W_h\| \|\text{diag}[f'(W_h h_{t-1} + W_i x_t)]\| \leq \prod_{j=t-k+1}^{t} \alpha\beta = (\alpha\beta)^k$$

Gradient can become very **small or very large quickly**, and the locality assumption of gradient descent breaks down (Vanishing gradient) [Bengio et al 1994]

- Vanishing gradients are quite prevalent and a serious issue.
- A real example
  - Training a feed-forward network
  - y-axis: sum of the gradient norms
  - Earlier layers have exponentially smaller sum of gradient norms
  - This will make training earlier layers much slower.



Vanishing/exploding gradients

- In an RNN trained on long sequences (*e.g.* 100 time steps) the gradients can easily explode or vanish.
  - So RNNs have difficulty dealing with long-range dependencies.
- Many methods proposed for reduce the effect of vanishing gradients; although it is still a problem
  - Introduce shorter path between long connections
  - Abandon stochastic gradient descent in favor of a much more sophisticated Hessian-Free (HF) optimization
  - Add fancier modules that are robust to handling long memory; e.g. Long Short Term Memory (LSTM)
    - http://colah.github.io/posts/2015-08-Understanding-LSTMs/
    - A very good explanatiom of LSTM
- One trick to handle the exploding-gradients:
  - Clip gradients with bigger sizes:

Define $g = \frac{\partial E}{\partial W}$

If $\|g\| \geq threshold$ then

$$g \leftarrow \frac{threshold}{\|g\|} g$$

Perplexity Results

- KN5 = Count-based language model with Kneser-Ney smoothing & 5-grams

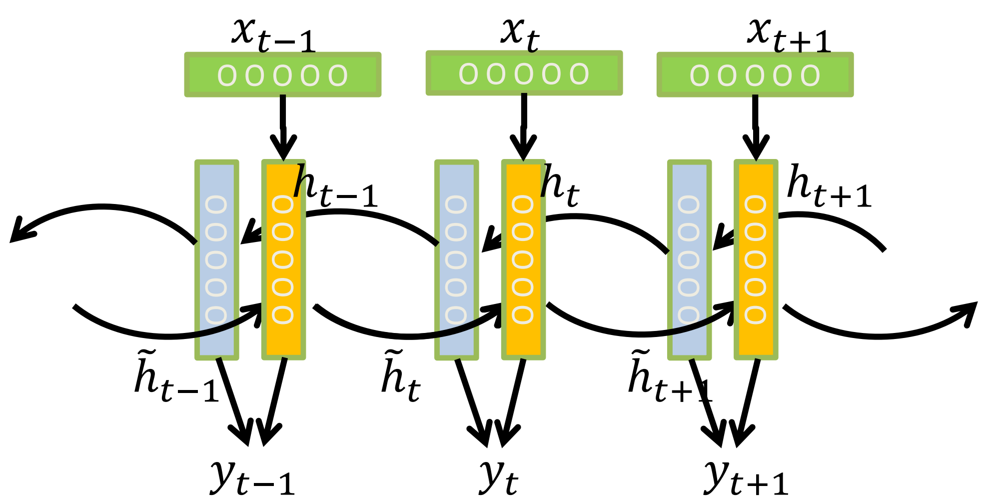
Table 2. *Comparison of different neural network architectures on Penn Corpus (1M words) and Switchboard (4M words).*

	Penn Corpus		Switchboard	
Model	NN	NN+KN	NN	NN+KN
KN5 (baseline)	-	141	-	92.9
feedforward NN	141	118	85.1	77.5
RNN trained by BP	137	113	81.3	75.4
RNN trained by BPTT	123	106	77.5	72.5

- Table from paper *Extensions of recurrent neural network language model* by Mikolov et al 2011

Bi-directional RNN

- One of the issues with RNN:
- Hidden variables capture only one side context
- A bi-directional structure



$$h_t = f(W_h h_{t-1} + W_i x_t)$$

$$\tilde{h}_t = f(\widetilde{W}_h \tilde{h}_{t+1} + \widetilde{W}_i x_t)$$

$$y_t = \mathrm{softmax}\left(W_o h_t + \widetilde{W}_o \tilde{h}_t\right)$$

Stack of bi-directional networks

- Use the same idea and make your model further complicated: