CS365: Deep Learning

Recurrent Neural Network

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
`arijit@iitp.ac.in`

Introduction

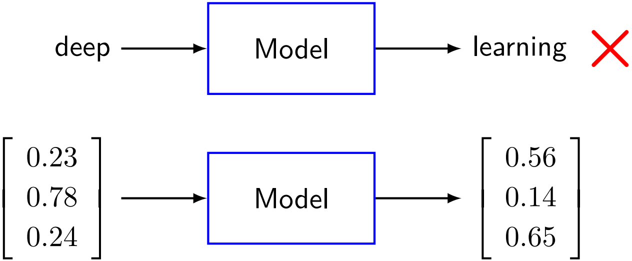
- Recurrent neural networks are used for processing sequential data in general
  - Convolution neural network is specialized for image
- Capable of processing variable length input
- Track long term dependencies
- Need to maintain information about ordering
- Shares parameters across different part of the model
- Examples:
  - Example: "I went to IIT in 2017" or "In 2017, I went to IIT"
  - Example: "I grew up in Bengal. .... I can speak fluent ____"
  - Example: The food was good, not bad at all -vs- The food was bad, not good at all
  - For traditional machine learning models require to learn rules for different positions

Modeling of Natural Language

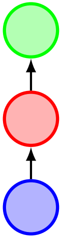
- Language are the most common sequence models
- Natural language model requires a probability distribution over string
- Terminology
  - Bag of Words - a representation of text that describes the occurrence of words within a document
  - $TF(x, d)$ – we count how prevalent each term x is in a single document d
    - Words are commonly normalized to lowercase and stemmed by removing their suffixes; common stopwords (such as a, an, the, etc.) are removed
  - $IDF(x) = 1 + \log\left(\frac{\text{total number of documents}}{\text{number of documents containing } x}\right)$
  - $TFIDF(x, d) = Tf(x, d) \times IDF(x)$ – used for measuring similarity between a query and a document
  - N-grams – A sequence of n adjacent words is called an n-gram
    - A bag of words is the 1-gram or unigram model

RNN: Predict next word

- Word cannot be fed directly
- Words need to be represented as numbers
- Encoding scheme:
  - 1-hot encoding: Only one element will be 1, eg, $[0, 0, 1, 0, \ldots]$. Vector size $=$ Vocabulary size
  - Word embedding: Only a set of numbers will be used, eg., $[0.87, 0.93, 0.14, \ldots]$. Vector size is $\ll$ Vocabulary size

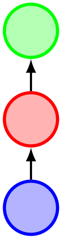


Types of applications



Feed-forward network

Types of applications



Feed-forward network

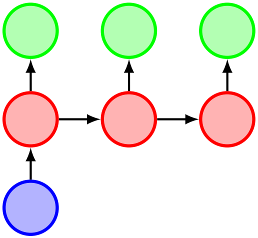
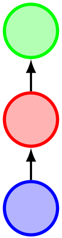


Image captioning

Types of applications



Feed-forward network

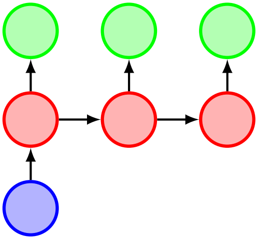
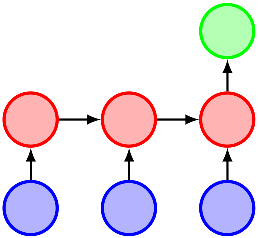
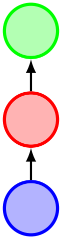


Image captioning



Sentiment analysis

Types of applications



Feed-forward network

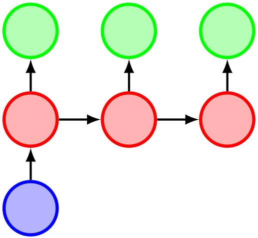
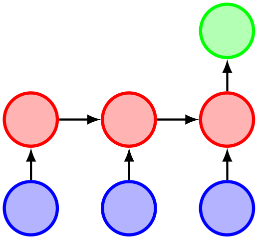
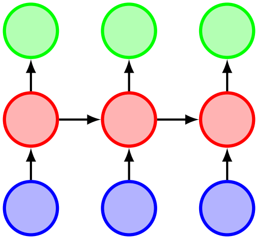


Image captioning

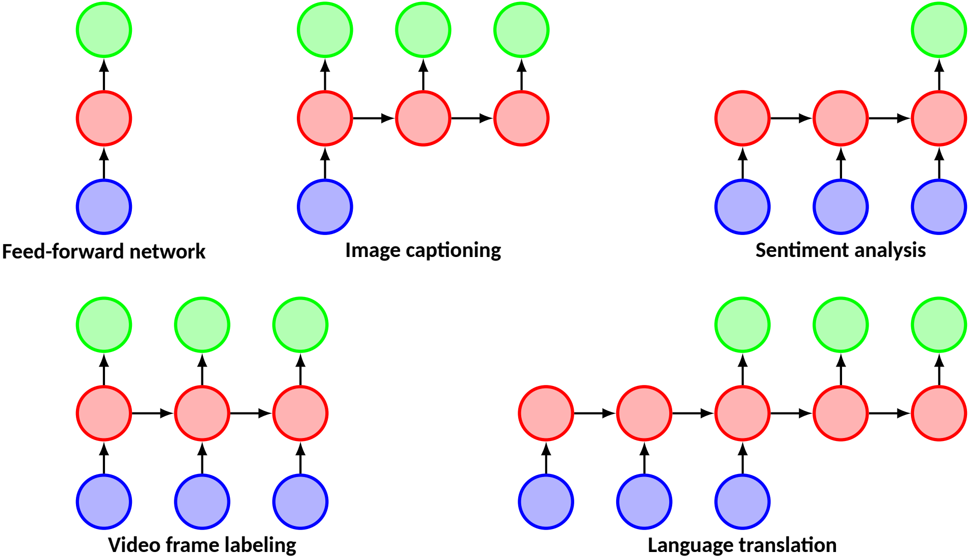


Sentiment analysis



Video frame labeling

Types of applications



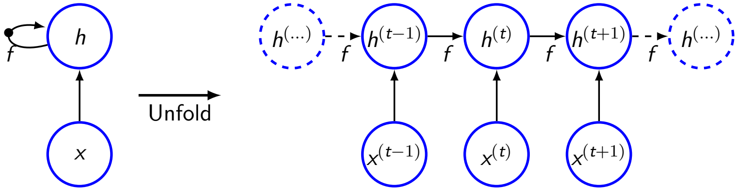
Computational graph

- Formal way to represent the computation
- Unfolding the graph results in sharing of parameters
- Consider a system $\mathrm{s}^{(t)} = f(\mathrm{s}^{(t-1)}, \boldsymbol{\theta})$ where $\mathrm{s}^{(t)}$ denotes the state of the system
  - It is recurrent
  - For finite number of steps, it can be unfolded
  - Example: $s^{(3)} = f(s^{(2)}, \theta) = f(f(s^{(1)}, \theta), \theta)$

$$s^{(\dots)} \xrightarrow{f} s^{(t-1)} \xrightarrow{f} s^{(t)} \xrightarrow{f} s^{(t+1)} \xrightarrow{f} s^{(\dots)}$$

System with inputs

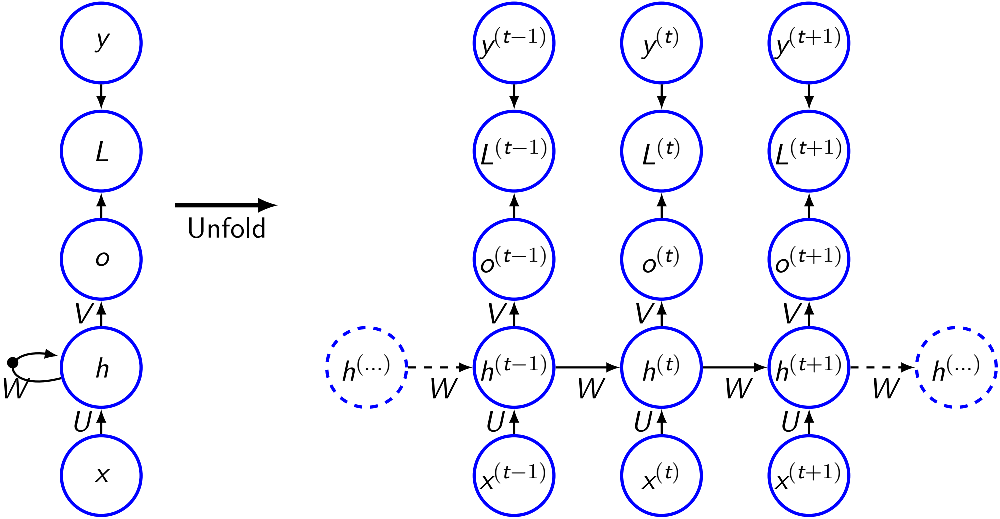
- A system will be represented as $s^{(t)} = f(s^{(t-1)}, x^{(t)}, \boldsymbol{\theta})$
  - A state contains information of whole past sequence
- Usually state is indicated as hidden units such that $h^{(t)} = f(h^{(t-1)}, x^{(t)}, \boldsymbol{\theta})$
- While predicting, network learn $h^{(t)}$ as a kind of lossy summary of past sequence upto t
  - $h^{(t)}$ depends on $(x^{(t)}, x^{(t-1)}, \ldots, x^{(1)})$



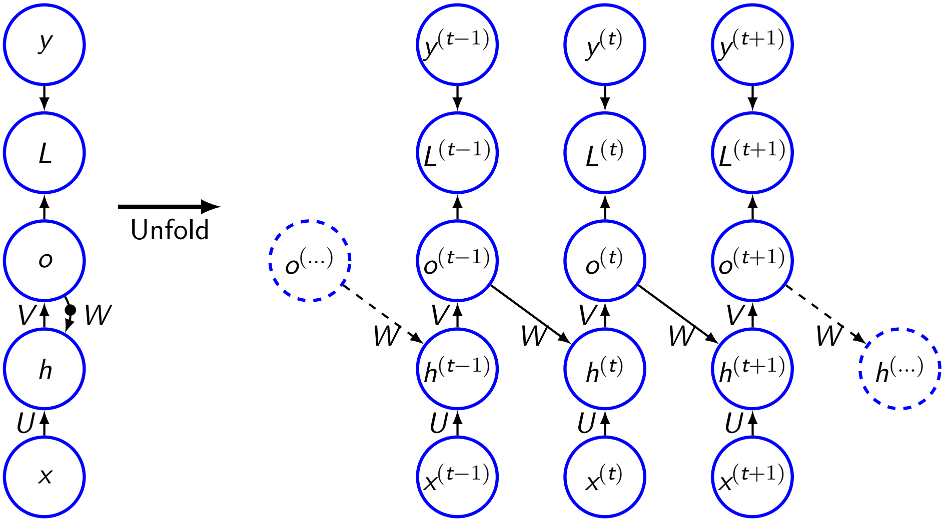
System with inputs (contd.)

- Unfolded recursion after t steps will be $\mathsf{h}^{(t)} = \mathsf{g}^{(t)}(\mathsf{x}^{(t)}, \mathsf{x}^{(t-1)}, \ldots, \mathsf{x}^{(1)}) = f(\mathsf{h}^{(t-1)}, \mathsf{x}^{(t)}, \boldsymbol{\theta})$
- Unfolding process has some advantages
  - Regardless of sequence length, learned model has same input size
  - Uses the same transition function f with the same parameters at every time steps
- Can be trained with fewer examples
- Recurrent graph is succinct
- Unfolded graph illustrates the information flow

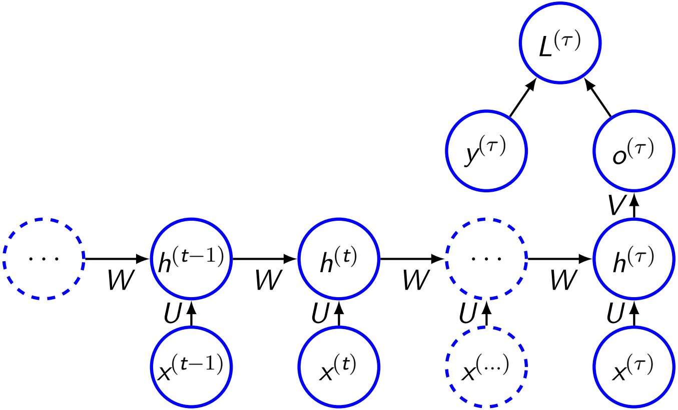
Recurrent connection in hidden units



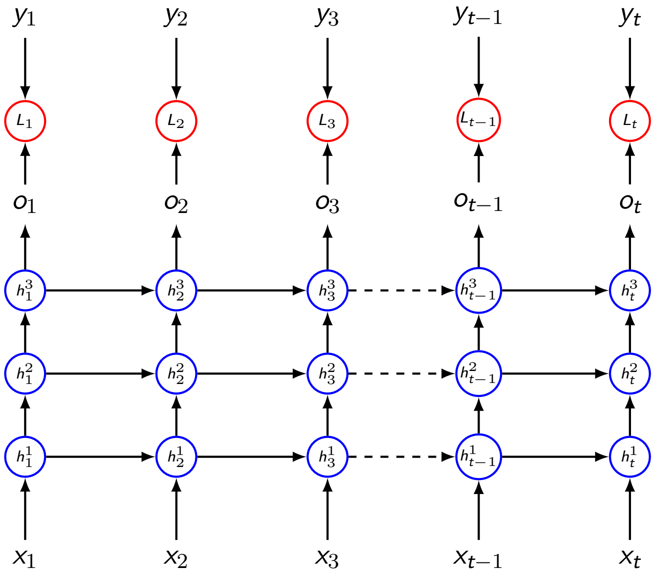
Output to hidden unit connection



Sequence processing

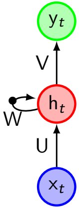


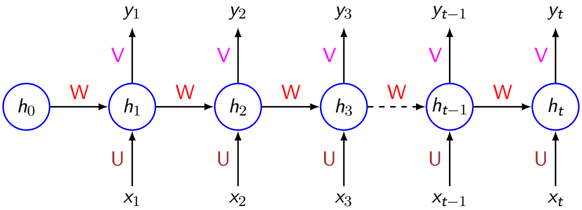
Stacked/Deep RNN



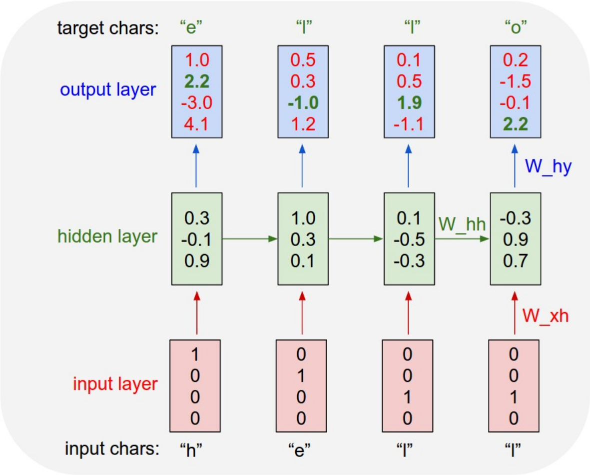
Recurrent neural network

- Function computable by a Turing machine can be computed by such recurrent network of finite size
- $\tanh$ is usually chosen as activation function for hidden units
- Output can be considered as discrete, so y gives unnormalized log probabilities
- Forward propagation begins with initial state h_0
- So we have,
  - $\mathsf{a}^{(t)} = \mathsf{b} + \mathsf{W}\mathsf{h}^{(t-1)} + \mathsf{U}\mathsf{x}^{(t)}$
  - $\mathsf{h}^{(t)} = \tanh(\mathsf{a}^{(t)})$
  - $\mathsf{y}^{(t)} = \mathsf{c} + \mathsf{V}\mathsf{h}^{(t)}$
  - $\hat{\mathsf{y}}^{(t)} = \mathrm{softmax}(\mathsf{y}^{(t)})$
- Input and output have the same length





Example: char-rnn



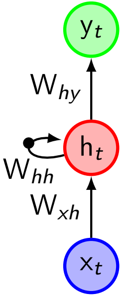


Image source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN: William Shakespeare

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

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.
```

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

RNN: Maths / Latex-1

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

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

RNN: Maths / Latex-2

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

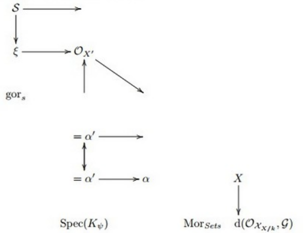
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

1. $\mathcal{F}$ is an algebraic space over S.
2. If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad -1(\mathcal{O}_{X_{étale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

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

RNN: Linux Kernel

```c
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

```c
/*
 *  Copyright (c) 2006-2010, Intel Mobile Communications.  All rights reserved.
 *
 *   This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 *        This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *
 *  GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *    along with this program; if not, write to the Free Software Foundation,
 *  Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>
```


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

RNN: Visualization

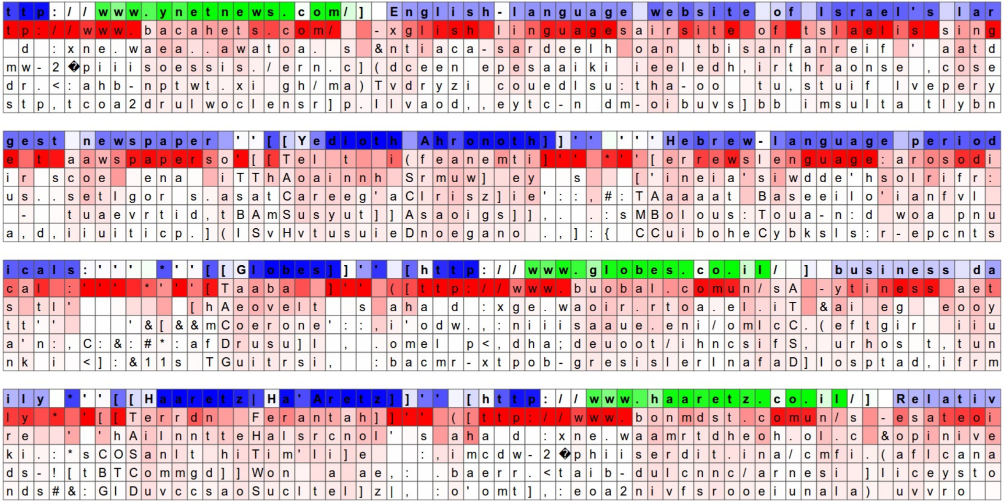
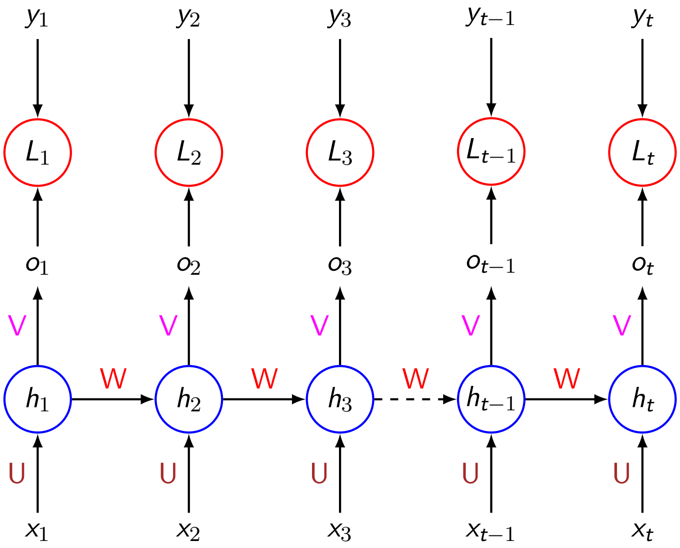


Image source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

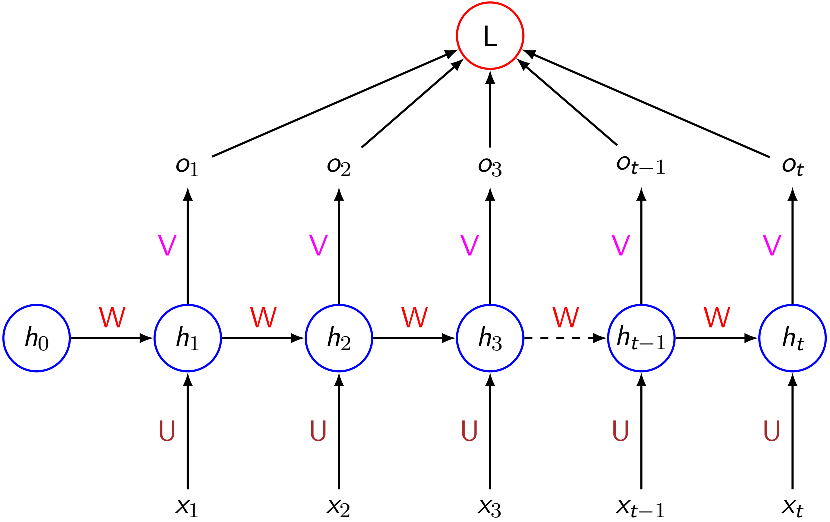
RNN: Pros and Cons

- Pros:
  - Can process any length input
  - Computation for step t can use information from distant past (theoretically)
  - Model size does not increase for longer input
  - Same weight parameters are shared across the time steps
- Cons:
  - Computation is usually slow
  - Difficult to access information to distant past

Loss per time point



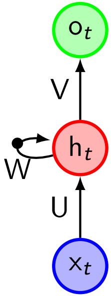
Total loss



L
o_1
o_2
o_3
o_{t-1}
o_t
V
V
V
V
V
h_0
W
h_1
W
h_2
W
h_3
W
h_{t-1}
W
h_t
U
U
U
U
U
x_1
x_2
x_3
x_{t-1}
x_t

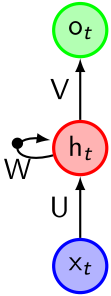
Gradient computation in RNN

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients



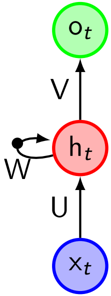
Gradient computation in RNN

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients
- Loss function
  - $l_t = \frac{1}{2} \sum_{k=1}^{\text{out}} (o_k - y_k)^2,$



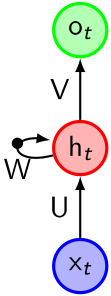
Gradient computation in RNN

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients
- Loss function
  - $l_t = \frac{1}{2}\sum_{k=1}^{\text{out}}(o_k - y_k)^2, \ L = \frac{1}{2}\sum_{t=1}^{\tau}\sum_{k=1}^{\text{out}}(o_{tk} - y_{tk})^2$



Gradient computation in RNN

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients
- Loss function
  - $l_t = \frac{1}{2}\sum_{k=1}^{\text{out}}(o_k - y_k)^2, \; L = \frac{1}{2}\sum_{t=1}^{\tau}\sum_{k=1}^{\text{out}}(o_{tk} - y_{tk})^2$
  - $L = -\sum_{t=1}^{\tau}\sum_{k=1}^{\text{out}}\left[y_{tk}\ln o_{tk} + (1 - y_{tk})\ln(1 - o_{tk})\right]$
- Truncated backpropagation through time (BPTT)



Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

- Assume that the loss is computed by unfolding the system for τ units of time

$$L(x_1, \ldots, x_\tau, y_1, \ldots, y_\tau, w_h, w_o) = \frac{1}{\tau} \sum_{t=1}^{\tau} l(y_t, o_t)$$

Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

- Assume that the loss is computed by unfolding the system for τ units of time

$$L(x_1, \ldots, x_\tau, y_1, \ldots, y_\tau, w_h, w_o) = \frac{1}{\tau} \sum_{t=1}^{\tau} l(y_t, o_t)$$

- Compute the derivative with respect to w_h using chain rule

$$\frac{\partial L}{\partial w_h}$$

Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

- Assume that the loss is computed by unfolding the system for τ units of time

$$L(x_1, \ldots, x_\tau, y_1, \ldots, y_\tau, w_h, w_o) = \frac{1}{\tau} \sum_{t=1}^{\tau} l(y_t, o_t)$$

- Compute the derivative with respect to w_h using chain rule

$$\frac{\partial L}{\partial w_h} = \frac{1}{\tau} \sum_{t=1}^{\tau} \frac{\partial l(y_t, o_t)}{\partial w_h}$$

Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

- Assume that the loss is computed by unfolding the system for τ units of time

$$L(x_1, \ldots, x_\tau, y_1, \ldots, y_\tau, w_h, w_o) = \frac{1}{\tau}\sum_{t=1}^{\tau} l(y_t, o_t)$$

- Compute the derivative with respect to w_h using chain rule

$$\frac{\partial L}{\partial w_h} = \frac{1}{\tau}\sum_{t=1}^{\tau}\frac{\partial l(y_t, o_t)}{\partial w_h} = \frac{1}{\tau}\sum_{t=1}^{\tau}\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_o)}{\partial h_t}\frac{\partial h_t}{\partial w_h}$$

Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

- Assume that the loss is computed by unfolding the system for τ units of time

$$L(x_1, \ldots, x_\tau, y_1, \ldots, y_\tau, w_h, w_o) = \frac{1}{\tau}\sum_{t=1}^{\tau} l(y_t, o_t)$$

- Compute the derivative with respect to w_h using chain rule

$$\frac{\partial L}{\partial w_h} = \frac{1}{\tau}\sum_{t=1}^{\tau}\frac{\partial l(y_t, o_t)}{\partial w_h} = \frac{1}{\tau}\sum_{t=1}^{\tau}\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_o)}{\partial h_t}\frac{\partial h_t}{\partial w_h}$$

- Computation of third factor in above, $\frac{\partial h_t}{\partial w_h}$, is tricky. It needs to be computed recurrently

$$\frac{\partial h_t}{\partial w_h}$$

Gradients in RNN-1

- Consider a simple situation with following dynamics:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h) \\ o_t &= g(h_t, w_o) \end{aligned}$$

- Assume that the loss is computed by unfolding the system for τ units of time

$$L(x_1, \ldots, x_\tau, y_1, \ldots, y_\tau, w_h, w_o) = \frac{1}{\tau} \sum_{t=1}^{\tau} l(y_t, o_t)$$

- Compute the derivative with respect to w_h using chain rule

$$\frac{\partial L}{\partial w_h} = \frac{1}{\tau} \sum_{t=1}^{\tau} \frac{\partial l(y_t, o_t)}{\partial w_h} = \frac{1}{\tau} \sum_{t=1}^{\tau} \frac{\partial l(y_t, o_t)}{\partial o_t} \frac{\partial g(h_t, w_o)}{\partial h_t} \frac{\partial h_t}{\partial w_h}$$

- Computation of third factor in above, $\frac{\partial h_t}{\partial w_h}$, is tricky. It needs to be computed recurrently

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial w_h}$$

Gradients in RNN-2

- Above equation is similar to $a_0 = 0$, and $a_t = b_t + c_t a_{t-1}$ for $t = 1, 2, \ldots$

Gradients in RNN-2

- Above equation is similar to $a_0 = 0$, and $a_t = b_t + c_t a_{t-1}$ for $t = 1, 2, \ldots$
- Then for $t \geq 1$, a_t can be expressed in the following form

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i$$

Gradients in RNN-2

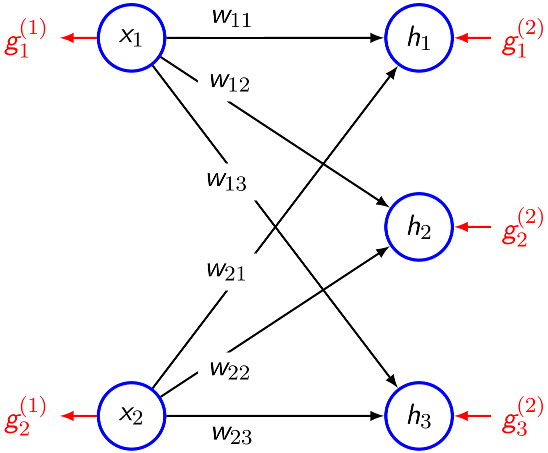
- Above equation is similar to $a_0 = 0$, and $a_t = b_t + c_t a_{t-1}$ for $t = 1, 2, \ldots$
- Then for $t \geq 1$, a_t can be expressed in the following form

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i$$

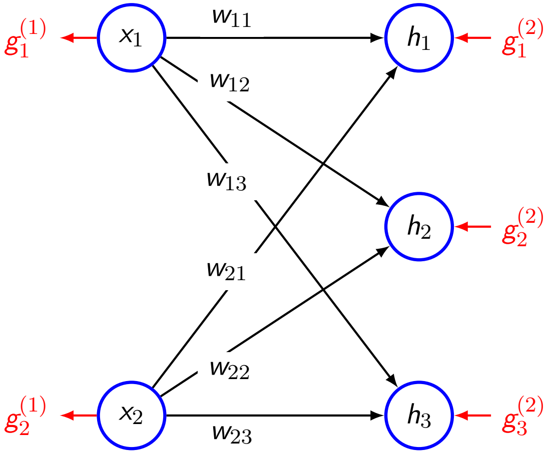
- Hence, by substituting a_t, b_t, c_t appropriately, we get

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_h)}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}, w_h)}{\partial w_h}$$

Gradient in matrix form



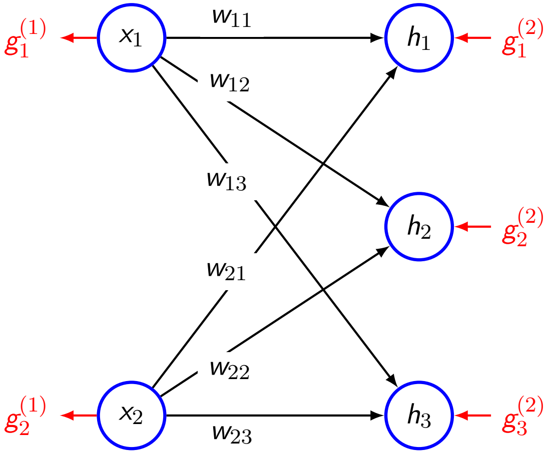
Gradient in matrix form



$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\mathrm{h} = \mathrm{W}^T \mathrm{x}$$

Gradient in matrix form



$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\mathrm{h} = \mathrm{W}^T \mathrm{x}$$

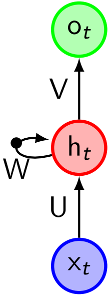
$$\begin{bmatrix} g_1^{(1)} \\ g_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} g_1^{(2)} \\ g_2^{(2)} \\ g_3^{(2)} \end{bmatrix}$$

$$\mathrm{g}^{(1)} = \mathrm{W}\mathrm{g}^{(2)}$$

Backpropagation through time-1

- Consider RNN excluding bias, $x_t \in \mathbb{R}^d$, $h_t \in \mathbb{R}^h$, $o_t \in \mathbb{R}^q$, $W \in \mathbb{R}^{h\times h}$, $U \in \mathbb{R}^{h\times d}$, $V \in \mathbb{R}^{q\times h}$

$$
\begin{aligned}
h_t &= Wh_{t-1} + Ux_t \\
o_t &= Vh_t
\end{aligned}
$$



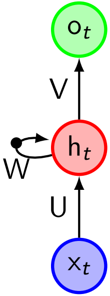
Backpropagation through time-1

- Consider RNN excluding bias, $\mathrm{x}_t \in \mathbb{R}^d$, $\mathrm{h}_t \in \mathbb{R}^h$, $\mathrm{o}_t \in \mathbb{R}^q$, $\mathrm{W} \in \mathbb{R}^{h\times h}$, $\mathrm{U} \in \mathbb{R}^{h\times d}$, $\mathrm{V} \in \mathbb{R}^{q\times h}$

$$\begin{aligned} \mathrm{h}_t &= \mathrm{W}\mathrm{h}_{t-1} + \mathrm{U}\mathrm{x}_t \\ \mathrm{o}_t &= \mathrm{V}\mathrm{h}_t \end{aligned}$$

- Loss function over a period of τ time units can be computed as

$$L = \frac{1}{\tau}\sum_{t=1}^{\tau} l(y_t, o_t)$$



Backpropagation through time-1

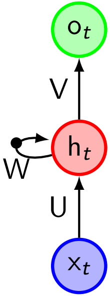
- Consider RNN excluding bias, $\mathsf{x}_t \in \mathbb{R}^d$, $\mathsf{h}_t \in \mathbb{R}^h$, $\mathsf{o}_t \in \mathbb{R}^q$, $\mathsf{W} \in \mathbb{R}^{h\times h}$, $\mathsf{U} \in \mathbb{R}^{h\times d}$, $\mathsf{V} \in \mathbb{R}^{q\times h}$

$$\begin{aligned} \mathsf{h}_t &= \mathsf{W}\mathsf{h}_{t-1} + \mathsf{U}\mathsf{x}_t \\ \mathsf{o}_t &= \mathsf{V}\mathsf{h}_t \end{aligned}$$

- Loss function over a period of τ time units can be computed as

$$L = \frac{1}{\tau}\sum_{t=1}^{\tau} l(y_t, o_t)$$

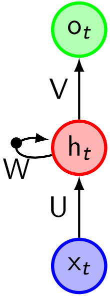
- We need to compute $\dfrac{\partial L}{\partial \mathsf{W}}, \dfrac{\partial L}{\partial \mathsf{U}}, \dfrac{\partial L}{\partial \mathsf{V}}$



Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$



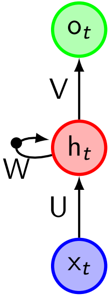
Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$

- Calculate the gradient of loss wrt V in output layer

$$\frac{\partial L}{\partial \mathrm{V}}$$



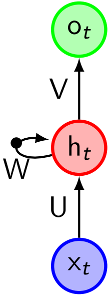
Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$

- Calculate the gradient of loss wrt V in output layer

$$\frac{\partial L}{\partial \mathrm{V}} = \sum_{t=1}^{\tau} \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial V} \right) \right] =$$



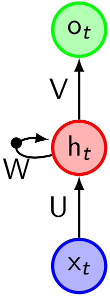
Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$

- Calculate the gradient of loss wrt V in output layer

$$\frac{\partial L}{\partial \mathrm{V}} = \sum_{t=1}^{\tau} \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial V} \right) \right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathrm{o}_t} \mathrm{h}_t^T$$



Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

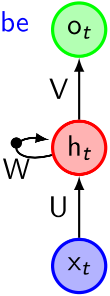
$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$

- Calculate the gradient of loss wrt V in output layer

$$\frac{\partial L}{\partial \mathrm{V}} = \sum_{t=1}^{\tau} \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial V} \right) \right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathrm{o}_t} \mathrm{h}_t^T$$

- At the final time step τ, L depends on h_τ only via o_τ. Therefore, the gradient will be

$$\frac{\partial L}{\partial \mathrm{h}_\tau}$$



Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

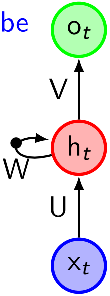
$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$

- Calculate the gradient of loss wrt V in output layer

$$\frac{\partial L}{\partial \mathrm{V}} = \sum_{t=1}^{\tau} \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial V} \right) \right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathrm{o}_t} \mathrm{h}_t^T$$

- At the final time step τ, L depends on h_τ only via o_τ. Therefore, the gradient will be

$$\frac{\partial L}{\partial \mathrm{h}_\tau} = \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_\tau}, \frac{\partial \mathrm{o}_\tau}{\partial \mathrm{h}_\tau} \right) \right] =$$



Backpropagation through time-2

- Differentiating the loss with respect to model output at any time step t is

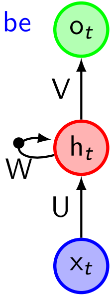
$$\frac{\partial L}{\partial \mathrm{o}_t} = \frac{\partial l(\mathrm{o}_t, y_t)}{\partial \mathrm{o}_t \cdot \tau} \in \mathbb{R}^q$$

- Calculate the gradient of loss wrt V in output layer

$$\frac{\partial L}{\partial \mathrm{V}} = \sum_{t=1}^{\tau} \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial V} \right) \right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathrm{o}_t} \mathrm{h}_t^T$$

- At the final time step τ, L depends on h_τ only via o_τ. Therefore, the gradient will be

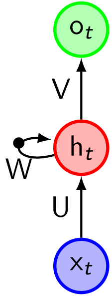
$$\frac{\partial L}{\partial \mathrm{h}_\tau} = \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_\tau}, \frac{\partial \mathrm{o}_\tau}{\partial \mathrm{h}_\tau} \right) \right] = \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_\tau}$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

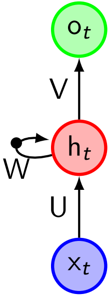
$$\frac{\partial L}{\partial \mathrm{h}_t}$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

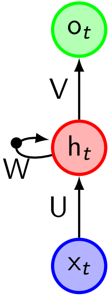
$$\frac{\partial L}{\partial \mathrm{h}_t} = \left[\prod \left(\frac{\partial L}{\partial \mathrm{h}_{t+1}}, \frac{\partial \mathrm{h}_{t+1}}{\partial \mathrm{h}_t} \right)\right] + \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial \mathrm{h}_t} \right)\right] =$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

$$\frac{\partial L}{\partial \mathrm{h}_t} = \left[\prod \left(\frac{\partial L}{\partial \mathrm{h}_{t+1}}, \frac{\partial \mathrm{h}_{t+1}}{\partial \mathrm{h}_t}\right)\right] + \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial \mathrm{h}_t}\right)\right] = \mathrm{W}^T \frac{\partial L}{\partial \mathrm{h}_{t+1}} + \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_t}$$



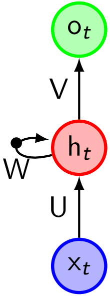
Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

$$\frac{\partial L}{\partial \mathrm{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathrm{h}_{t+1}}, \frac{\partial \mathrm{h}_{t+1}}{\partial \mathrm{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial \mathrm{h}_t}\right)\right] = \mathrm{W}^T \frac{\partial L}{\partial \mathrm{h}_{t+1}} + \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_t}$$

- Expanding the recurrence computation for any time step $1 \le t \le \tau$, we get

$$\frac{\partial L}{\partial \mathrm{h}_t}$$



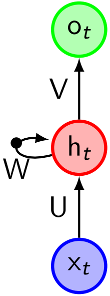
Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

$$\frac{\partial L}{\partial \mathrm{h}_t} = \left[\prod \left(\frac{\partial L}{\partial \mathrm{h}_{t+1}}, \frac{\partial \mathrm{h}_{t+1}}{\partial \mathrm{h}_t}\right)\right] + \left[\prod \left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial \mathrm{h}_t}\right)\right] = \mathrm{W}^T \frac{\partial L}{\partial \mathrm{h}_{t+1}} + \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_t}$$

- Expanding the recurrence computation for any time step $1 \leq t \leq \tau$, we get

$$\frac{\partial L}{\partial \mathrm{h}_t} = \sum_{i=t}^{\tau} (\mathrm{W}^T)^{\tau-i} \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_{\tau+t-i}}$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

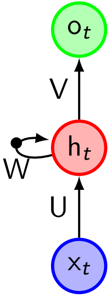
$$\frac{\partial L}{\partial \mathsf{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_{t+1}}, \frac{\partial \mathsf{h}_{t+1}}{\partial \mathsf{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathsf{o}_t}, \frac{\partial \mathsf{o}_t}{\partial \mathsf{h}_t}\right)\right] = \mathsf{W}^T \frac{\partial L}{\partial \mathsf{h}_{t+1}} + \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_t}$$

- Expanding the recurrence computation for any time step $1 \leq t \leq \tau$, we get

$$\frac{\partial L}{\partial \mathsf{h}_t} = \sum_{i=t}^{\tau} (\mathsf{W}^T)^{\tau-i} \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_{\tau+t-i}}$$

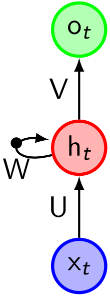
- Computing gradient wrt U

$$\frac{\partial L}{\partial \mathsf{U}}$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule
$$\frac{\partial L}{\partial \mathrm{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathrm{h}_{t+1}}, \frac{\partial \mathrm{h}_{t+1}}{\partial \mathrm{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial \mathrm{h}_t}\right)\right] = \mathrm{W}^T \frac{\partial L}{\partial \mathrm{h}_{t+1}} + \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_t}$$
- Expanding the recurrence computation for any time step $1 \leq t \leq \tau$, we get
$$\frac{\partial L}{\partial \mathrm{h}_t} = \sum_{i=t}^{\tau} (\mathrm{W}^T)^{\tau - i} \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_{\tau+t-i}}$$
- Computing gradient wrt U
$$\frac{\partial L}{\partial \mathrm{U}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathrm{h}_t}, \frac{\partial \mathrm{h}_t}{\partial \mathrm{U}}\right)\right]$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

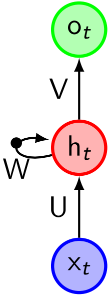
$$\frac{\partial L}{\partial \mathrm{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathrm{h}_{t+1}}, \frac{\partial \mathrm{h}_{t+1}}{\partial \mathrm{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathrm{o}_t}, \frac{\partial \mathrm{o}_t}{\partial \mathrm{h}_t}\right)\right] = \mathrm{W}^T \frac{\partial L}{\partial \mathrm{h}_{t+1}} + \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_t}$$

- Expanding the recurrence computation for any time step $1 \leq t \leq \tau$, we get

$$\frac{\partial L}{\partial \mathrm{h}_t} = \sum_{i=t}^{\tau} (\mathrm{W}^T)^{\tau-i} \mathrm{V}^T \frac{\partial L}{\partial \mathrm{o}_{\tau+t-i}}$$

- Computing gradient wrt U

$$\frac{\partial L}{\partial \mathrm{U}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathrm{h}_t}, \frac{\partial \mathrm{h}_t}{\partial \mathrm{U}}\right)\right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathrm{h}_t} \mathrm{x}_t^T$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

$$\frac{\partial L}{\partial \mathsf{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_{t+1}}, \frac{\partial \mathsf{h}_{t+1}}{\partial \mathsf{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathsf{o}_t}, \frac{\partial \mathsf{o}_t}{\partial \mathsf{h}_t}\right)\right] = \mathsf{W}^T \frac{\partial L}{\partial \mathsf{h}_{t+1}} + \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_t}$$

- Expanding the recurrence computation for any time step $1 \leq t \leq \tau$, we get

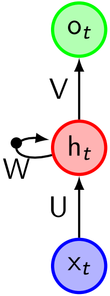
$$\frac{\partial L}{\partial \mathsf{h}_t} = \sum_{i=t}^{\tau} (\mathsf{W}^T)^{\tau - i} \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_{\tau+t-i}}$$

- Computing gradient wrt U

$$\frac{\partial L}{\partial \mathsf{U}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_t}, \frac{\partial \mathsf{h}_t}{\partial \mathsf{U}}\right)\right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathsf{h}_t} \mathsf{x}_t^T$$

- Computing gradient wrt W

$$\frac{\partial L}{\partial \mathsf{W}}$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

$$\frac{\partial L}{\partial \mathsf{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_{t+1}}, \frac{\partial \mathsf{h}_{t+1}}{\partial \mathsf{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathsf{o}_t}, \frac{\partial \mathsf{o}_t}{\partial \mathsf{h}_t}\right)\right] = \mathsf{W}^T \frac{\partial L}{\partial \mathsf{h}_{t+1}} + \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_t}$$

- Expanding the recurrence computation for any time step $1 \le t \le \tau$, we get

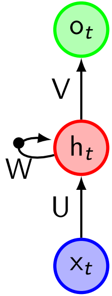
$$\frac{\partial L}{\partial \mathsf{h}_t} = \sum_{i=t}^{\tau} (\mathsf{W}^T)^{\tau-i} \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_{\tau+t-i}}$$

- Computing gradient wrt U

$$\frac{\partial L}{\partial \mathsf{U}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_t}, \frac{\partial \mathsf{h}_t}{\partial \mathsf{U}}\right)\right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathsf{h}_t} \mathsf{x}_t^T$$

- Computing gradient wrt W

$$\frac{\partial L}{\partial \mathsf{W}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_t}, \frac{\partial \mathsf{h}_t}{\partial \mathsf{W}}\right)\right]$$



Backpropagation through time-3

- For $t < \tau$, L depends on h_t via h_{t+1} and o_t. Hence, using chain rule

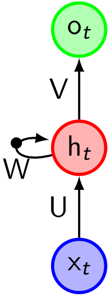
$$\frac{\partial L}{\partial \mathsf{h}_t} = \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_{t+1}}, \frac{\partial \mathsf{h}_{t+1}}{\partial \mathsf{h}_t}\right)\right] + \left[\prod\left(\frac{\partial L}{\partial \mathsf{o}_t}, \frac{\partial \mathsf{o}_t}{\partial \mathsf{h}_t}\right)\right] = \mathsf{W}^T \frac{\partial L}{\partial \mathsf{h}_{t+1}} + \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_t}$$

- Expanding the recurrence computation for any time step $1 \leq t \leq \tau$, we get

$$\frac{\partial L}{\partial \mathsf{h}_t} = \sum_{i=t}^{\tau} (\mathsf{W}^T)^{\tau-i} \mathsf{V}^T \frac{\partial L}{\partial \mathsf{o}_{\tau+t-i}}$$

- Computing gradient wrt U

$$\frac{\partial L}{\partial \mathsf{U}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_t}, \frac{\partial \mathsf{h}_t}{\partial \mathsf{U}}\right)\right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathsf{h}_t} \mathsf{x}_t^T$$



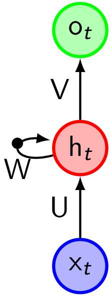
- Computing gradient wrt W

$$\frac{\partial L}{\partial \mathsf{W}} = \sum_{t=1}^{\tau} \left[\prod\left(\frac{\partial L}{\partial \mathsf{h}_t}, \frac{\partial \mathsf{h}_t}{\partial \mathsf{W}}\right)\right] = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathsf{h}_t} \mathsf{h}_{t-1}^T$$

Garident issues

- Gradient

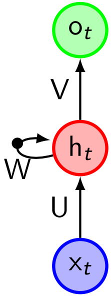
$$\frac{\partial L}{\partial W}$$



Garident issues

- Gradient

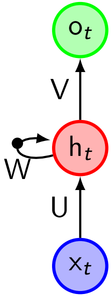
$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial W}$$



Garident issues

- Gradient

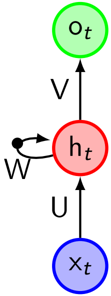
$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial W} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial y_t}$$



Garident issues

- Gradient

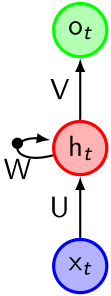
$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial W} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial y_t} \frac{\partial y_t}{\partial h_t}$$



Garident issues

- Gradient

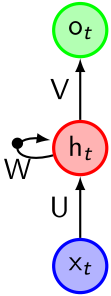
$$\frac{\partial L}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial \mathrm{y}_t} \frac{\partial \mathrm{y}_t}{\partial \mathrm{h}_t} \frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k}$$



Garident issues

- Gradient

$$\frac{\partial L}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial \mathrm{y}_t} \frac{\partial \mathrm{y}_t}{\partial \mathrm{h}_t} \frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k} \frac{\partial \mathrm{h}_k}{\partial \mathrm{W}}$$



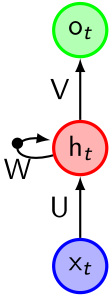
Garident issues

- Gradient

$$\frac{\partial L}{\partial \mathsf{W}} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial \mathsf{W}} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial \mathsf{y}_t} \frac{\partial \mathsf{y}_t}{\partial \mathsf{h}_t} \frac{\partial \mathsf{h}_t}{\partial \mathsf{h}_k} \frac{\partial \mathsf{h}_k}{\partial \mathsf{W}}$$

- Now we have,

$$\frac{\partial \mathsf{h}_t}{\partial \mathsf{h}_k}$$



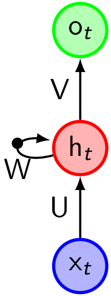
Garident issues

- Gradient

$$\frac{\partial L}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial \mathrm{y}_t} \frac{\partial \mathrm{y}_t}{\partial \mathrm{h}_t} \frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k} \frac{\partial \mathrm{h}_k}{\partial \mathrm{W}}$$

- Now we have,

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathrm{h}_i}{\partial \mathrm{h}_{i-1}}$$



Garident issues

- Gradient

$$\frac{\partial L}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \frac{\partial l_t}{\partial \mathrm{W}} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial l_t}{\partial \mathrm{y}_t} \frac{\partial \mathrm{y}_t}{\partial \mathrm{h}_t} \frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k} \frac{\partial \mathrm{h}_k}{\partial \mathrm{W}}$$

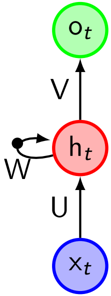
- Now we have,

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathrm{h}_i}{\partial \mathrm{h}_{i-1}} = \prod_{i=k+1}^{t} \mathrm{W}^T \mathrm{diag}[\phi'(\mathrm{h}_{i-1})]$$

- Issues in gradient

$$\left\| \frac{\partial \mathrm{h}_i}{\partial \mathrm{h}_{i-1}} \right\| \leq \|\mathrm{W}^T\| \, \|\mathrm{diag}[\phi'(\mathrm{h}_{i-1})]\| \leq \lambda_{\mathrm{W}} \lambda_{\phi}$$

$$\left\| \frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_k} \right\| \leq (\lambda_{\mathrm{W}} \lambda_{\phi})^{t-k}$$



Gradient issues in RNN

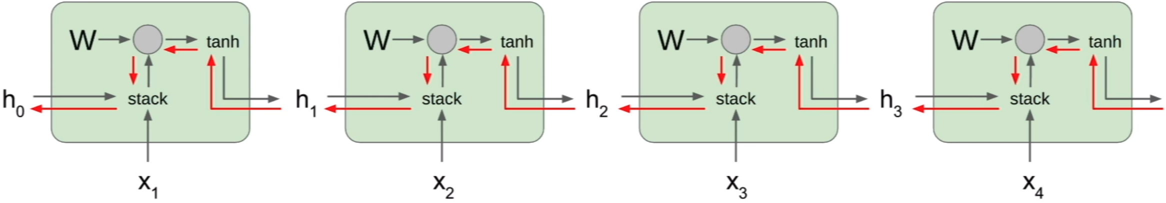
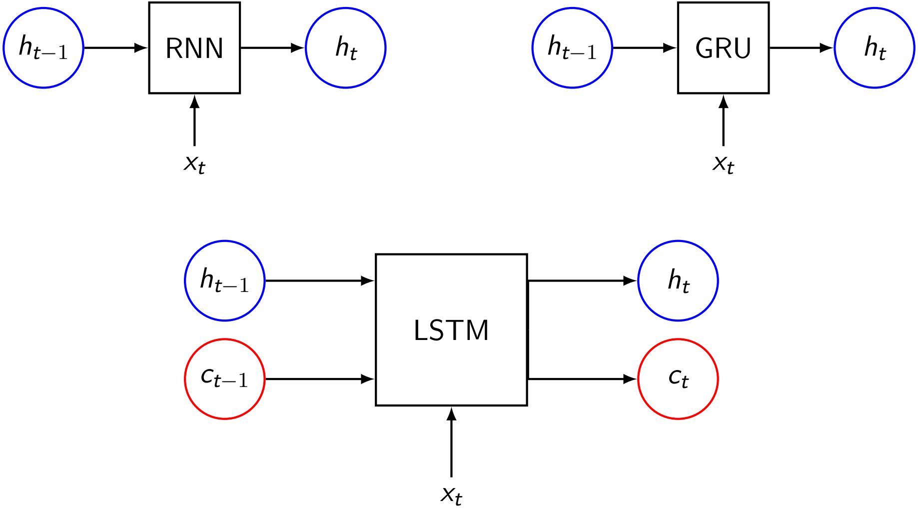


Image source: Internet

Issues with Vanilla RNN

- Hard to retain the information in hidden state with successive matrix multiplications
  - Hidden states of recurrent networks are inherently short-term
  - No mechanism exist for fine grained control of what information to retain from hidden state
- The LSTM / GRU use analog gates to control the flow of information

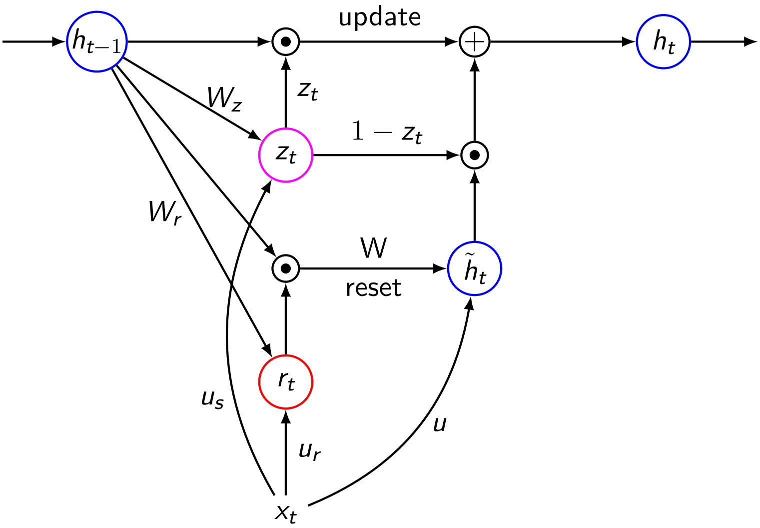
RNN variants



Gated Recurrent Unit

- An improved version of RNN
- It uses the notion of gating in propagating information
- Similar to Long Short-Term Memory (LSTM)
- Uses less number of parameters compared to LSTM

GRU: Architecture



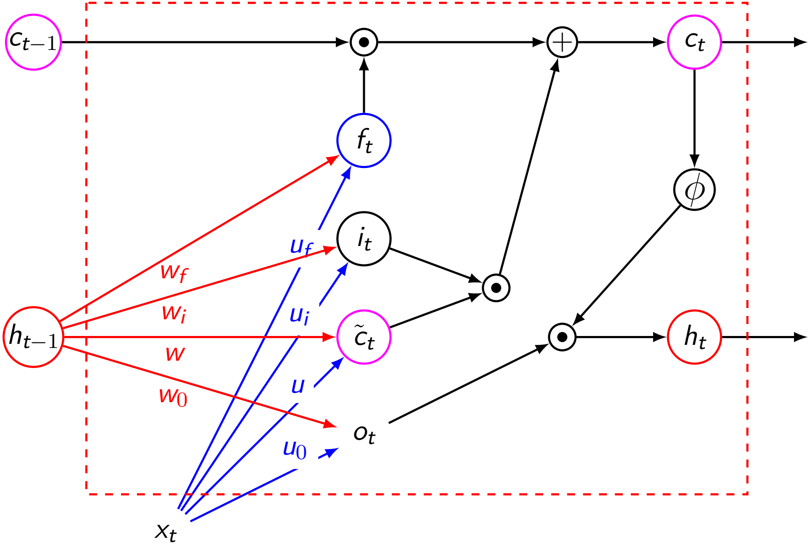
GRU: Functionality

- Update gate: $z_t = \sigma(w_z h_{t-1} + u_z x_t)$
- Reset gate: $r_t = \sigma(w_r h_{t-1} + u_r x_t)$
- Candidate gate: $\tilde{h}_t = \phi(w(r_t \odot h_{t-1}) + u x_t)$
- Output gate: $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$
- Analogy: x_t - weather today, h_{t-1} - clothes wore yesterday, $\tilde{h}_t$ - candidate clothes for today, h_t - actual clothes wear today.
- Update and reset gates determine to what extent we take into account these factors - Ignore weather completely, Forget what we wore.

Long-term vs Short-term Memory

- A vanilla RNN carries forward a hidded state across the time layers
- An LSTM carries forward both a hidden state h_t and a cell state c_t
  - The hidden state is like short-term memory
  - The cell state is like a long-term memory
  - Gates are used to control updates from layer to layer
  - Leaking between short-term and long-term meory allowed

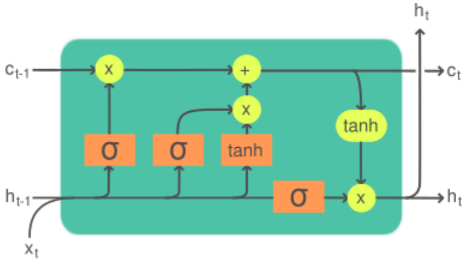
LSTM

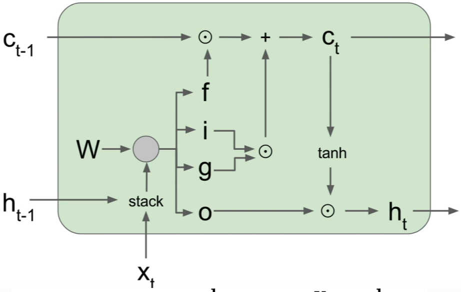


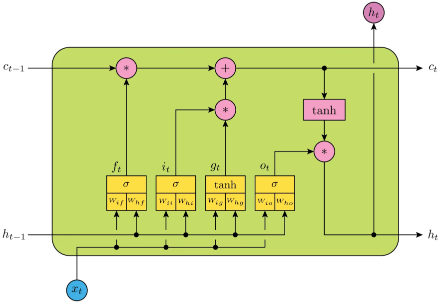
LSTM: Functionality

- Forget gate: $f_t = \sigma(w_f h_{t-1} + u_f x_t + b_f)$
- Input gate: $i_t = \sigma(w_i h_{t-1} + u_i x_t + b_i)$
- Output gate: $o_t = \sigma(w_o h_{t-1} + u_o x_t + b_o)$
- Candidate memory: $\tilde{c}_t = \phi(w h_{t-1} + u x_t + b_c)$
- Memory cell $c_t = f_t c_{t-1} + i_t \tilde{c}_t$
- Output gated memory: $h_t = o_t \phi(c_t)$

LSTM Representations







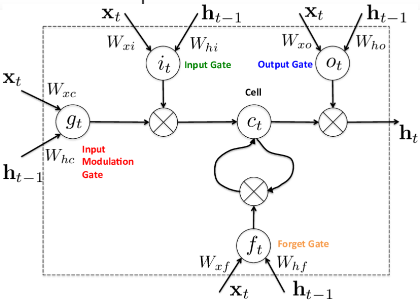
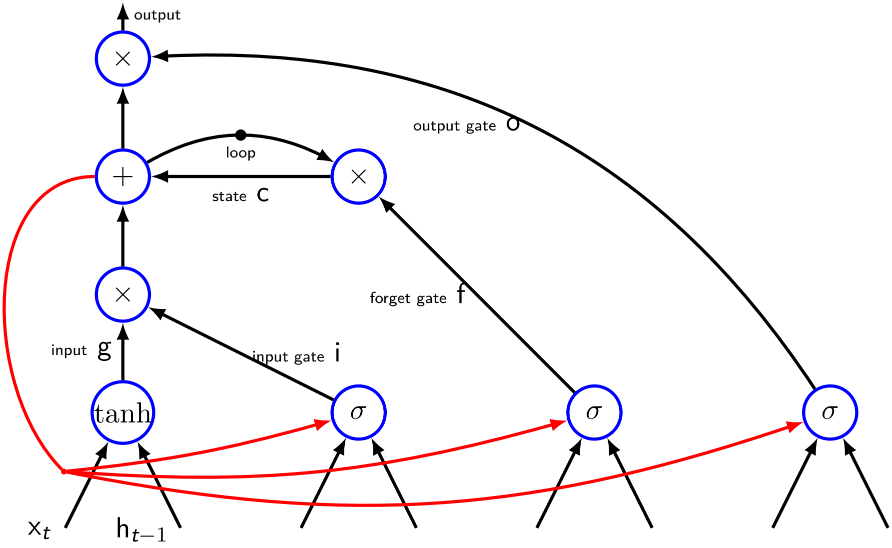


Image source: Internet

LSTM

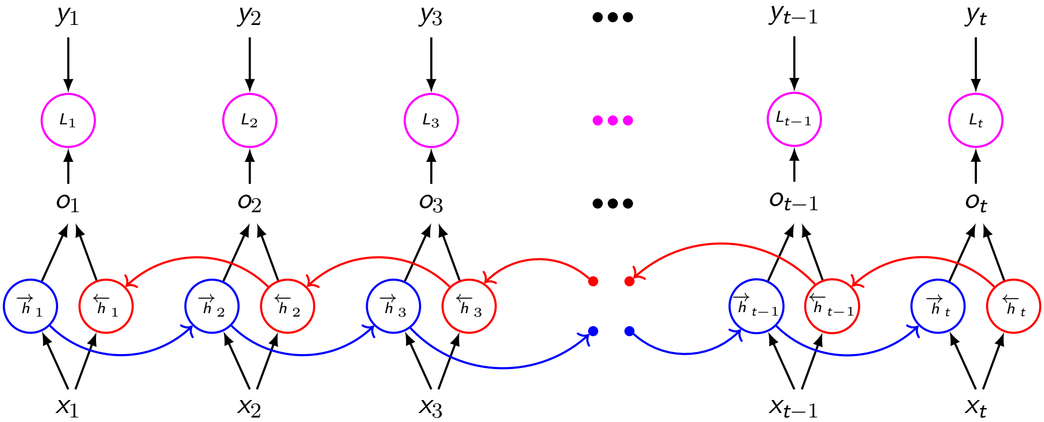

output
×
output gate o
loop
+
×
state c
×
forget gate f
input g
input gate i
tanh
σ
σ
σ
x_t
h_{t-1}

Bidirectional RNN

- 1. I am ____. 2. I am ____ hungry. 3. I am ____ hungry, and I can eat a full tandoori!
- Possible tokens: First - happy, Second - not / very, Third - 'not' is incompatible

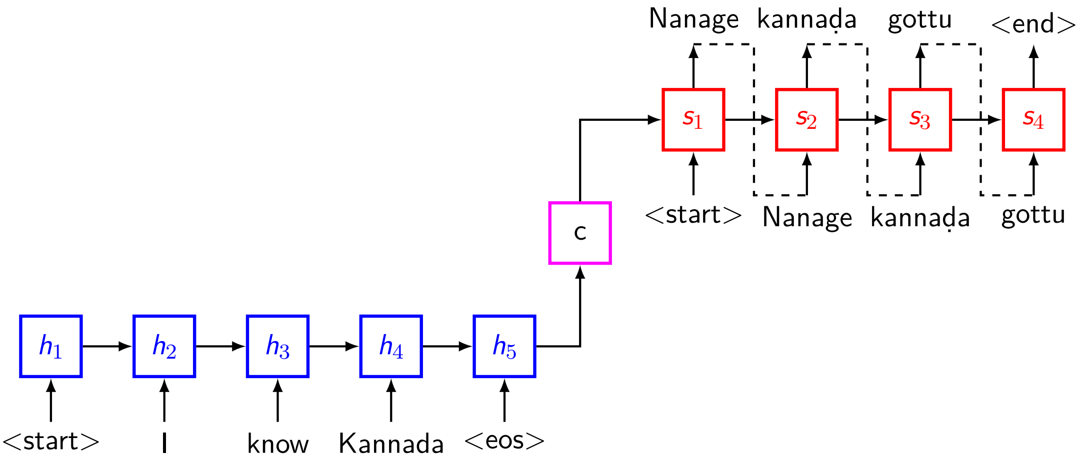
Bidirectional RNN

- 1. I am ____. 2. I am ____ hungry. 3. I am ____ hungry, and I can eat a full tandoori!
- Possible tokens: First - happy, Second - not / very, Third - 'not' is incompatible



Machine Translation: Encoder-Decoder

Machine Translation: Encoder-Decoder



Attention with RNN

- $\alpha_t = NN(s_{t-1}, h_t)$
- Softmax is used for weightage
- Context $= \sum_t \alpha_t h_t$

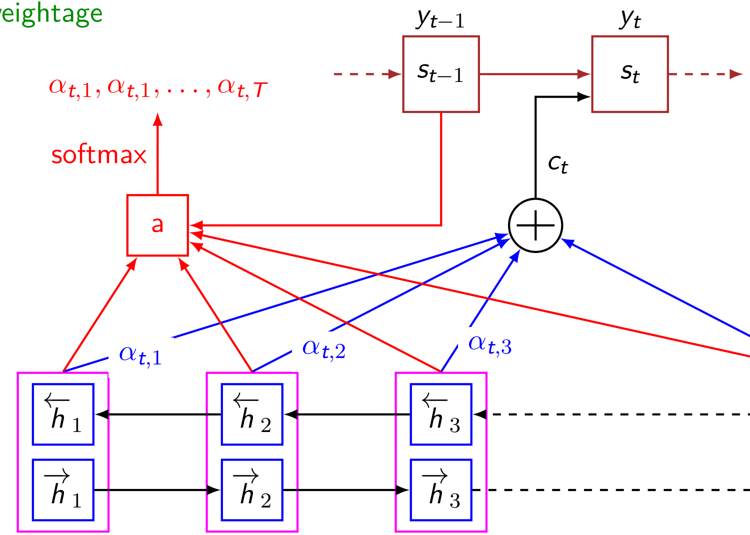


Image captioning

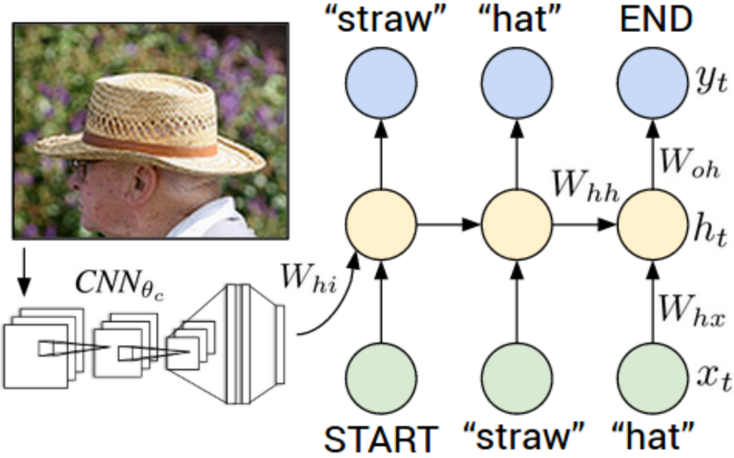


Image source: Deep Visual-Semantic Alignments for Generating Image Descriptions

Image captioning - success story

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

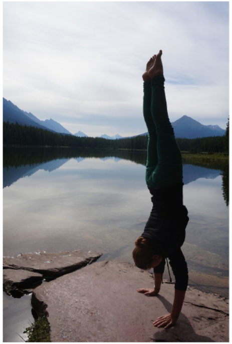
A man riding a dirt bike on a dirt track

Image source: Li et. al. CS231 slides

Image captioning - failure story

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Image source: Li et. al. CS231 slides

Visual Question Answering - 1

Vehicles and Transportation

Q: What sort of vehicle uses this item?
A: firetruck

Brands, Companies and Products

Q: When was the soft drink company shown first created?
A: 1898

Objects, Material and Clothing

Q: What is the material used to make the vessels in this picture?
A: copper

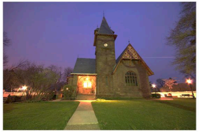
Sports and Recreation

Q: What is the sports position of the man in the orange shirt?
A: goalie

Cooking and Food

Q: What is the name of the object used to eat this food?
A: chopsticks

Geography, History, Language and Culture



Q: What days might I most commonly go to this building?
A: Sunday

People and Everyday Life

Q: Is this photo from the 50's or the 90's?
A: 50's

Plants and Animals

Q: What phylum does this animal belong to?
A: chordate, chordata

Science and Technology

Q: How many chromosomes do these creatures have?
A: 23

Weather and Climate

Q: What is the warmest outdoor temperature at which this kind of weather can happen?
A: 32 degrees

Image source: Internet

Visual Question Answering - 2

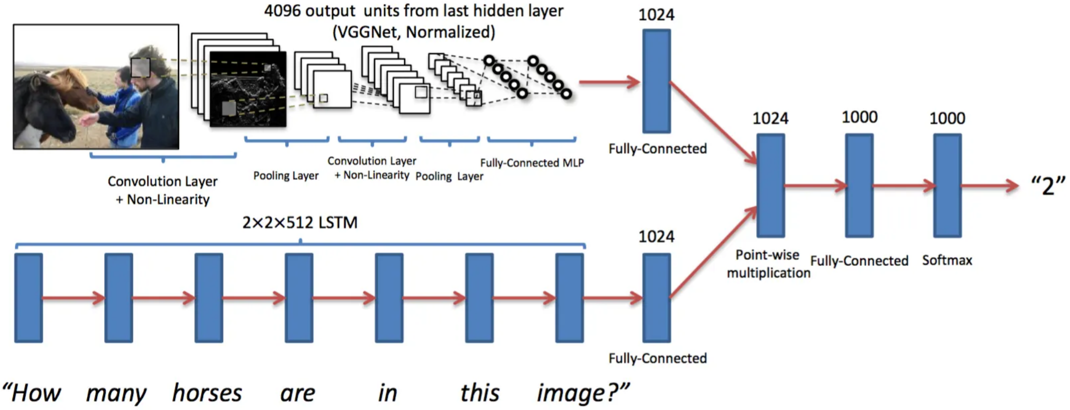


Image source: Internet

RNN: Summary

- RNN is good to model sequence relation
- It models sequence using recurrence relation
- It can handle variable length input
- RNN needs to be trained backpropagation through time

Word Embedding

- Computer only understands numbers
- Words need to be converted into numbers
  - 1-hot encoding - no relation among similar words
  - Embedding - similar words have close relation
- Neural networks can be used to learn word embedding
- Consider the following situation: We have n documents and a vocabulary of size d
  - It can be represented as a document-word matrix of size $n \times d$
  - Let it be factorized as $D \approx UV$, where $U = n \times k$ and $V = k \times d$
    - Rows of U contains embedding of documents
    - Columns of V contains embedding of words

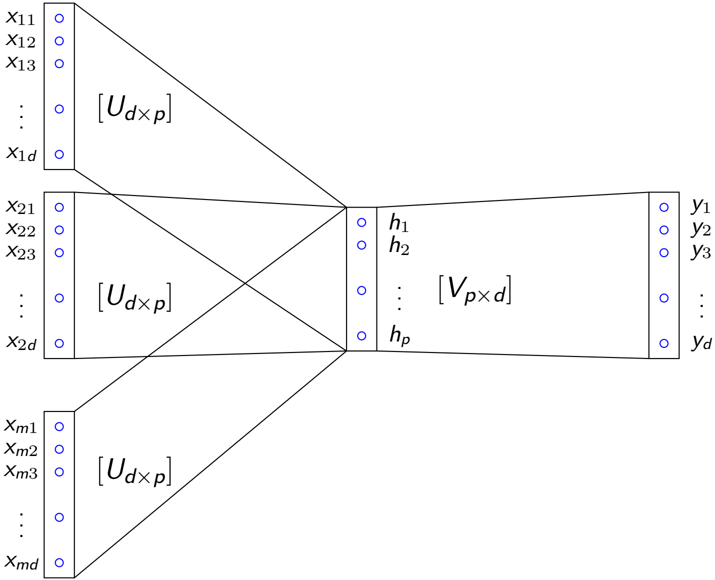
Word2Vec

- Predicting target word from a given context
  - It tries to predict ith word in a sentence using a window of width t around the word
    - $w_{i-t} \dots w_{i-1} w_{i+1} \dots w_{i+t}$ are used to predict w_i
    - This model is known as continuous bag of words (CBOW) model
- Predicting context from target word
  - It tries to predict a context given a single word
    - Predict $w_{i-t} \dots w_{i-1} w_{i+1} \dots w_{i+t}$ from the given w_i
    - This is known as skipgram model

CBOW Model

- Training inputs are all context-word pairs
  - Context is input, word - outcome. Supervised learning
  - Context length $m = 2t$ (eg. $w_1, \dots, w_m$), outcome is w
  - w may be viewed as categorical variable with d possible values, d is the size of vocabulary
  - Target is to compute $p(w|w_1 \dots w_m)$ and maximize the product of these probabilities over all training examples

CBOW Model: Architecture



Architecture details

- Input: $m \times d$, one-hot encoding $(x_{ij} \in \{0, 1\})$ for each m. i - context position, j - word identifier
- Hidden layer - p nodes
- Output - d nodes
- $\bar{u}_j = (u_{j1}, \ldots, u_{jp})$ - p dimensional embedding of the j th word over entire corpus
- $\bar{h} = (h_1, \ldots, h_p)$ - embedding of specific instatiation of an input context
    - $h_q = \sum_{i=1}^{m} \left[\sum_{j=1}^{d} u_{jq} x_{ij} \right] \qquad \forall q = \{1, \ldots, p\}$
    - In vectored form $\bar{h} = \sum_{i=1}^{m} \sum_{j=1}^{d} \bar{u}_j x_{ij}$
    - One hot encoding are aggregated - ordering of words within the window size m does not affect the output

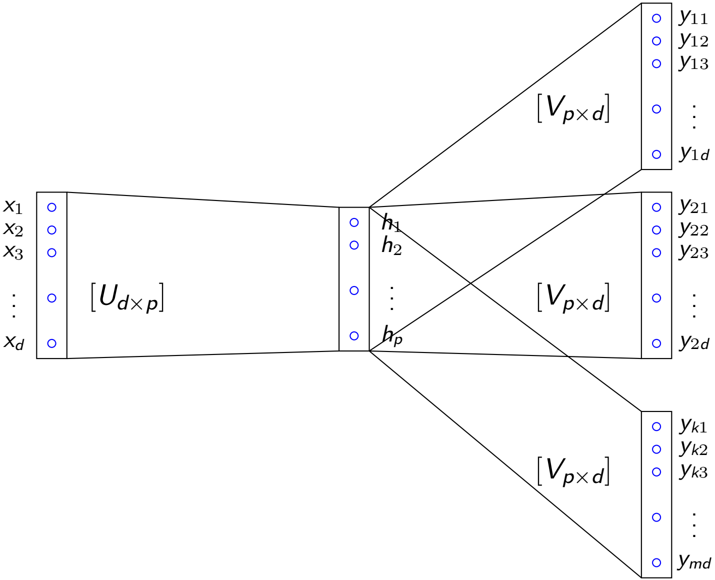
Architecture details (contd)

- Output $y_j = 1$ if the target word w is the j th word, 0 otherwise
- Softmax computes the probability $p(w|w_1 \ldots w_m)$ of the one-hot encoded ground truth outputs y_j as follows: $\hat{y}_j = p(y_j = 1|w_1 \ldots w_m) = \dfrac{\exp(\sum_{q=1}^{p} h_q v_{qj})}{\sum_{k=1}^{d} \exp(\sum_{q=1}^{p} h_q v_{qk})}$

Skipgram

- Reverse model of CBOW
- Traget word is used to predict m context words
    - One input, m output
    - w is input, $w_1, \ldots, w_m$ - output
    - Goal is to estimate $p(w_1, \ldots, w_m | w)$
    - Input is one-hot encoding
    - Output is also one-hot encoding

Skipgram model: Architecture



Skipgram model

- Input - $x_1, \ldots, x_d$ - binary inputs
- Output - $m \times d$, $y_{ij} \in \{0, 1\}$
- Final output $\hat{y}_{ij} = p(y_{ij} = 1|w)$, probabilities $\hat{y}_{ij}$ in the output layer for fixed i and varying j sum to 1
- Hidden layer contains p units, $h_1, \ldots, h_p$
- Each x_j is connected to all p nodes, matrix U has size $d \times p$
- The p hidden nodes are connected to each of m groups of d output nodes with the same set of shared weights, matrix V has size $p \times d$

Skipgram model

- The output of hidden layer can be computed as $h_q = \sum_{j=1}^{d} u_{jq} x_j, \quad \forall q$
  - If the input word w is the r th word, then one can simply copy u_{rq} to the q th node
  - Eventually r th row $(\bar{u}_r)$ of U is copied to the hidden layer
- Output is determined by V
  - Output $\hat{y}_{ij}$ is the probability that the word in the i th context position takes on the j th word
  - Since V is shared, the neural network predicts the same multinomial distribution for each context word
  - Therefore we have $\hat{y}_{ij} = p(y_{ij}|w) = \dfrac{\exp(\sum_{q=1}^{p} h_q v_{qj})}{\sum_{k=1}^{d} \exp(\sum_{q=1}^{p} h_q v_{qk})}, \quad \forall i$
  - Denominator is independent of context position