Topics:
• Recurrent Neural Networks
• Long-Short Term Memory (LSTMs)
• **Assignment 3 out**
  • Due date **extended** to **March 18th 11:59pm EST**.

• **Projects**
  • Released assignments; please **reach out** to your groups to discuss team formation
  • FB forum is being set up; right now post questions on piazza and I will relay
  • Project proposal due **March 22nd**
Friday

- Guest Lecture: Arjun Majumdar
  - Transformers, BERT, ViLBERT

https://arjunmajum.github.io/
Sequences in Input or Output?

• It’s a spectrum…

<table>
<thead>
<tr>
<th>Input: No sequence</th>
<th>Output: No sequence</th>
<th>Example: “standard” classification / regression problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>one to one</td>
<td>one to many</td>
<td>Im2Caption</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input: Sequence</th>
<th>Output: No sequence</th>
<th>Example: sentence classification, multiple-choice question answering</th>
</tr>
</thead>
<tbody>
<tr>
<td>many to one</td>
<td>many to many</td>
<td>machine translation, video classification, video captioning, open-ended question answering</td>
</tr>
</tbody>
</table>
Recurrent Neural Network
Recurrent Neural Network

usually want to predict a vector at some time steps
(Vanilla) Recurrent Neural Network
The state consists of a single “hidden” vector $h$:

$$y_t = W_{hy} h_t + b_y$$

$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$
Recurrent Neural Network

We can process a sequence of vectors $\mathbf{x}$ by applying a recurrence formula at every time step:

$$
\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)
$$

- New state
- Old state
- Some function with parameters $W$
- Input vector at some time step
Recurrent Neural Network

We can process a sequence of vectors $\mathbf{x}$ by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.
RNN: Computational Graph: Many to Many

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
RNN: Computational Graph: Many to One
RNN: Computational Graph: One to Many

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Sequence to Sequence: Many-to-one + one-to-many

**Many to one:** Encode input sequence in a single vector
Sequence to Sequence: Many-to-one + one-to-many

**Many to one:** Encode input sequence in a single vector

**One to many:** Produce output sequence from single input vector

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Distributed Representations Toy Example

• Can we interpret each dimension?
Power of distributed representations!

Local: \( \bullet \bullet \bigcirc \bullet = VR + HR + HE = ? \)

Distributed: \( \bullet \bullet \bigcirc \bullet = V + H + E \approx \bigcirc \)
Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence: “hello”
Training Time: MLE / “Teacher Forcing”

Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model.
Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model.
Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Let's do Monday.
Monday works for me.
Either day works for me.

Reply  Reply all  Forward
Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient.
**Truncated Backpropagation through time**

Run forward and backward through chunks of the sequence instead of whole sequence
Truncated Backpropagation through time

Loss

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
**Truncated** Backpropagation through time
min-char-rnn.py gist: 112 lines of Python

https://gist.github.com/karpathy/d4dee56687f829f086
THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decrease,
His tender heir might bear his memory:
But thou, contracted to one self with me,
For my fair love do sing thy own praise:
Thou art thyself, so sweetly doth thy soul spose
That art the rosy rap of many crowns,
And sweet perfection of all fairest things.
In every walk of life, thy fair proportions
A mark of strange integrity and grace,
And in thy other to all men's eyes a grace.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gratefully
Will be a ruin word of small worth held:
Then being asked, where all thy beauty lies,
Where is the treasure of thy hour decayd:
To say, within this close and cramm'd eye,
Were an all eating, and all-dulling glass.
How much more praise deserve's thy beauty's use,
If thou could'st answer this fair child of mine:
Shall sum my count, and make my old excuse;
Proving his beauty by succession there:
This verse is he new made when thou art old,
And set thy blood warm when thou art cold.
at first:

- Train more
- "Tmont thithey" fomesscerliund
  Keushey. Thom here
  sheulke, ammenith ol sivh I lalterhend Bleipile shuwy fil on aseterlome
  coaniogennc Phe lism thond hon at. MeIDimorotion in ther thize."
  - Train more
- Aftair fall unsuch that the hall for Prince Velzonski's that me of
  her hearily, and behs to so arwage fiving were to it beloge, pavu say falling misfort
  how, and Gognition is so overetical and ofter.
  - Train more
- "Why do what that day," replied Natasha, and wishing to himself the fact the
  princess, Princess Mary was easier, fed in had oftened him.
  Pierre aking his soul came to the packs and drove up his father-in-law women.
PANDARUS:
Alas, I think he shall be come approached and the day
When little strain 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 abs, 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.
The Stacks Project: open source algebraic geometry textbook

![Stacks Project Website Screenshot]

- The stacks project is licensed under the [GNU Free Documentation License](http://www.gnu.org/licenses/gpl-3.0.txt)
For $\bigoplus_{i=1}^{n}$ where $L_{n} = 0$, hence we can find a closed subset $\mathcal{H}$ in $H$ and any sets $F$ 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

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

and the comparision in the fibre product covering we have to prove the lemma generated by $\bigsqcup Z \times_U U \to V$. Consider the maps $M$ along the set of points $\text{Sch}/\text{Spec}$ and $U \to U$ is the fiber category of $S$ in $U$ in Section 77 and the fact that any $U$ affine, see Morphisms, Lemma 77. Hence we obtain a scheme $S$ and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_S U_i$$

which has a nonzero morphism we may assume that $f_i$ is of finite presentation over $S$. We claim that $\mathcal{O}_{X,i}$ is a scheme where $x, x', x'' \in S$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}_{X,x''}$ is separated. By Algebra, Lemma 77 we can define a map of complexes $\text{GL}_{s}(x'/S')$ and we win. □

To prove study we see that $F_{U}$ is a covering of $X'$, and $T_{i}$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $F_{i}$ exists and let $F_{i}$ be a presheaf of $\mathcal{O}_{X}$-modules on $C$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\tilde{M} = \mathcal{F} \otimes_{\text{Spec}(S)} \mathcal{O}_{X} = \mathcal{O}_{X}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)^{pp} \otimes_{\text{Prof}} (\text{Sch}/S)^{pp}$$

and

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

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 77. It may replace $S$ by $X_{\text{space,étale}}$ which gives an open subspace of $X$ and $T$ equal to $S_{\text{étale}}$, see Descent, Lemma 77. Namely, by Lemma 77 we see that $R$ is geometrically regular over $S$.

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

Suppose $X = \lim \{X_i\}$ by the formal open covering $X$ and a single map $\text{Proj}_{X}(A) = \text{Spec}(B)$ over $U$ compatible with the complex

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

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{U}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition 77 (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 $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

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

The following lemma surjective restcomposes of this implies that $\mathcal{F}_{\text{spec}} = \mathcal{F}_{\text{spec}} = \mathcal{F}_{\text{spec}}$.

**Lemma 0.2.** Let $X$ be a locally Noetherian scheme over $S$, $E = \mathcal{F}_{X/S}$. Set $I = \mathcal{F}_{X/S}$. Since $\mathcal{T}_{i} \subset \mathcal{T}_{i}$. Since $\mathcal{T}_{i}$ and $\mathcal{T}_{i}$ are nonzero over $\mathcal{T}$ is a subset of $\mathcal{T}_{i} \circ \mathcal{T}_{i}$ works.

**Lemma 0.3.** In Situation 77. Hence we may assume $q' = 0$.

Proof. We will use the property we see that $p$ is the next functor (77). On the other hand, by Lemma 77 we see that

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

where $K$ is an $F$-algebra where $\delta_{+1}$ is a scheme over $S$. □
Proof. Omitted.

**Lemma 0.1.** Let $C$ be a set of the construction.

Let $C$ be a gerber covering. Let $F$ be a quasi-coherent sheaves of $O$-modules. We have to show that
\[ O_{O_X} = O_X(L) \]

Proof. This is an algebraic space with the composition of sheaves $F$ on $X_{	ext{smooth}}$ we have
\[ O_X(F) = \{ \text{morph}_1 \times_{O_X} (G,F) \} \]
where $G$ defines an isomorphism $F \to F$ of $O$-modules.

**Lemma 0.2.** This is an integer $Z$ 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 $U \subset X$ 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
\[ b : X \to Y' \to Y \to Y'' \times_X Y \to X. \]
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 $F$ be a quasi-coherent sheaf of $O_X$-modules. The following are equivalent

1. $F$ is an algebraic space over $S$.
2. $X$ is an affine open covering.

Consider a common structure on $X$ and $X$ the functor $O_X(U)$ which is locally of finite type.

This since $F \in F$ and $x \in G$ the diagram

\[ \begin{array}{ccc} S & \to & X \\ \downarrow & & \downarrow \\ \xi & \to & O_X \\ \downarrow & & \downarrow \\ \text{Spec}(K) & \to & \text{Mod}_{f, h} \\ & & d(O_{X_{K_{(x)}} \times G}) \end{array} \]

is a limit. Then $G$ is a finite type and assume $S$ is a flat and $F$ and $G$ is a finite type $\text{Spec}(K)$. This is of finite type diagrams, and

- the composition of $G$ is a regular sequence,
- $O_X$ is a sheaf of rings.

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

Proof. This is clear that $G$ is a finite presentation, see Lemmas ??.

A reduced above we conclude that $U$ is an open covering of $C$. The functor $F$ is a "field"
\[ O_{X,h} \to F, \text{if } (O_{X_{h_{(x)}}} \to O_X) \text{ is an isomorphism of covering of } O_X. \]

If $F$ is the unique element of $F$ such that $X$ is an isomorphism.

The property $F$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $O_X$-algebra with $F$ are open of finite type over $S$.

If $F$ is a scheme theoretic image points.

If $F$ is a finite direct sum $O_{X_h}$ is a closed immersion, see Lemma ?? This is a sequence of $F$ is a similar morphism.
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << i))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &offset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
/*
 * 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>
```c
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setsid.h>
#include <asm/pgproto.h>

#define REG_PG  vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)  (func)

#define SWAP_ALLOCATE(nr)  (*)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %esp, %0, %3" : : "r" (0));  \  
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \  pC>[1]);

static void
os_prefix(unsigned long sys)
{
    ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
               (unsigned long)-1->lr_full; low;
    }
```
Searching for interpretable cells

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Searching for interpretable cells

```
/* Unpack a filter field's string representation from user-space */
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.
    */
```
Searching for interpretable cells

"You mean to imply that I have nothing to eat out of... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Searching for interpretable cells

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action—the one Kutuzov and the general mass of the army demanded—namely, simply to follow the enemy up. The French crown fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all—carried on by vis inertiae—pressed forward into boats and into the ice-covered water and did not surrender.

line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Searching for interpretable cells

```c
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask, siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!((current->notifier)(current->notifier_data))) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission
Searching for interpretable cells

Cell that turns on inside comments and quotes:

```c
/* Duplicate LSM field information. The lsm_rule is opaque, so */
/* re-initialized. */
static inline int audit_duplicate_lsm_field(struct audit_field *df,
    struct audit_field *sf)
{
    int ret = 0;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(lsm_str))
        return -ENOMEM;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void **)&df->lsm_rule);
    /* keep currently invalid fields around in case they */
    /* become valid after a policy reload. */
    if (ret == -EINVAL)
        pr_warn("Audit rule for LSM '%s' is invalid",
            df->lsm_str);
    ret = 0;
}
return ret;
```

quote/comment cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission
Searching for interpretable cells

code depth cell
Multilayer RNNs

\[ h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_t^{l-1} \end{pmatrix} \]

\[ h \in \mathbb{R}^n \quad W^l \in [n \times 2n] \]
Vanilla RNN Gradient Flow

\[ h_t = \text{tanh}(W_{hh}h_{t-1} + W_{hx}x_t) \]
\[ = \text{tanh} \left( (W_{hh} \quad W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]
\[ = \text{tanh} \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]
Vanilla RNN Gradient Flow

Backpropagation from $h_t$ to $h_{t-1}$ multiplies by $W$ (actually $W_{hh}$)

\[
h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)
= \tanh \left( \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
= \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\]

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Bengio et al. “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

Bengio et al., “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Vanilla RNN Gradient Flow

Computing gradient of \( h_0 \) involves many factors of \( W \) (and repeated \( \tanh \))

- **Largest singular value > 1:** Exploding gradients
- **Largest singular value < 1:** Vanishing gradients

Gradient clipping: Scale gradient if its norm is too big

\[
\text{grad norm} = \text{np.sum}(\text{grad} \times \text{grad})
\]
\[
\text{if grad norm} > \text{threshold}:
\]
\[
\text{grad} *= (\text{threshold} / \text{grad norm})
\]

Bengio et al., “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

Change RNN architecture

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Long Short Term Memory (LSTM)

Vanilla RNN

\[ h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]

LSTM

\[
\begin{pmatrix}
    i \\
    f \\
    o \\
    g
\end{pmatrix} =
\begin{pmatrix}
    \sigma \\
    \sigma \\
    \sigma \\
    \tanh
\end{pmatrix} W \begin{pmatrix}
    h_{t-1} \\
    x_t
\end{pmatrix}
\]

\[ c_t = f \odot c_{t-1} + i \odot g \]

\[ h_t = o \odot \tanh(c_t) \]

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997
Meet LSTMs

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs Intuition: Memory

- **Cell State / Memory**
LSTMs Intuition: Forget Gate

- Should we continue to remember this “bit” of information or not?

\[ f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \]
LSTMs Intuition: Input Gate

- Should we update this “bit” of information or not?
  - If so, with what?

\[
i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)
\]
\[
\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\]
LSTMs Intuition: Memory Update

• Forget that + memorize this

\[ C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \]
LSTMs Intuition: Output Gate

- Should we output this “bit” of information to “deeper” layers?

\[
o_t = \sigma (W_o [h_{t-1}, x_t] + b_o) \\
h_t = o_t \cdot \tanh (C_t)
\]
LSTMs Intuition: Additive Updates

Backpropagation from $c_t$ to $c_{t-1}$ only elementwise multiplication by $f$, no matrix multiply by $W$
LSTMs Intuition: Additive Updates

Uninterrupted gradient flow!
LSTMs Intuition: Additive Updates

Uninterrupted gradient flow!

Similar to ResNet!
LSTMs

- A pretty sophisticated cell

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTM Variants: Gated Recurrent Units

• Changes:
  – No explicit memory; memory = hidden output
  – $Z = \text{memorize new and forget old}$

$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh (W \cdot [r_t \cdot h_{t-1}, x_t])$$
$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$$