CS 4803 / 7643: Deep Learning

Topics:
- Recurrent Neural Networks (RNNs)
  - (Truncated) BackProp Through Time (BPTT)
  - LSTMs

Dhruv Batra
Georgia Tech
Administrativia

• HW3 Reminder
  – Due: 10/07 11:59pm
  – Theory: Convolutions, Representation Capacity, Double Descent
  – Implementation: Saliency methods (e.g. Grad-CAM) in Python and PyTorch/Captum

• Project Teams
  – [https://gtvault-my.sharepoint.com/:x:/g/personal/dbatra8_gatech_edu/EY4_65XOzWtOkXSSz2WgpoUBY8ux2gY9PsRzR6KngIlFEQ?e=4tnKWI](https://gtvault-my.sharepoint.com/:x:/g/personal/dbatra8_gatech_edu/EY4_65XOzWtOkXSSz2WgpoUBY8ux2gY9PsRzR6KngIlFEQ?e=4tnKWI)
  – Project Title
  – 1-3 sentence project summary TL;DR
  – Team member names
Administrativia

- Guest Lecture: Arjun Majumdar
  - Next class (10/8)
  - Transformers, BERT, ViLBERT

https://arjunmajum.github.io/
Recap from last time
New Topic: RNNs

one to one

one to many

many to one

many to many
New Words

• Recurrent Neural Networks (RNNs)

• Recursive Neural Networks
  – General family; think graphs instead of chains

• Types:
  – “Vanilla” RNNs (Elman Networks)
  – Long Short Term Memory (LSTMs)
  – Gated Recurrent Units (GRUs)
  – ...

• Algorithms
  – BackProp Through Time (BPTT)
  – BackProp Through Structure (BPTS)
What’s wrong with MLPs?

• Problem 1: Can’t model sequences
  – Fixed-sized Inputs & Outputs
  – No temporal structure

• Problem 2: Pure feed-forward processing
  – No “memory”, no feedback

(C) Dhruv Batra

Image Credit: Alex Graves, book
Why model sequences?
Sequences are everywhere…

\[ a_1=2 \quad a_2=0 \quad a_3=1 \quad a_4=3 \quad a_5=4 \quad a_6=2 \quad a_7=5 \]

\[ x = \text{bringen} \quad \text{sie} \quad \text{bitte} \quad \text{das} \quad \text{auto} \quad \text{zurück} \ . \]

\[ y = \text{please return the car} \ . \]
Sequences in Input or Output?

- It’s a spectrum…

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No sequence</td>
<td>No sequence</td>
<td>“standard” classification / regression problems</td>
</tr>
<tr>
<td>No sequence</td>
<td>Sequence</td>
<td>Im2Caption</td>
</tr>
<tr>
<td>Sequence</td>
<td>No sequence</td>
<td>sentence classification, multiple-choice question answering</td>
</tr>
<tr>
<td>Sequence</td>
<td>Sequence</td>
<td>machine translation, video classification, video captioning, open-ended question answering</td>
</tr>
</tbody>
</table>

Image Credit: Andrej Karpathy

(C) Dhruv Batra
2 Key Ideas

• Parameter Sharing
  – in computation graphs = adding gradients
Computational Graph
2 Key Ideas

• Parameter Sharing
  – in computation graphs = adding gradients

• “Unrolling”
  – in computation graphs with parameter sharing
How do we model sequences?

- No input

\[ s_t = f_\theta(s_{t-1}) \]
How do we model sequences?

- With inputs

\[ s_t = f_\theta(s_{t-1}, x_t) \]
2 Key Ideas

• Parameter Sharing
  – in computation graphs = adding gradients

• “Unrolling”
  – in computation graphs with parameter sharing

• Parameter sharing + Unrolling
  – Allows modeling arbitrary sequence lengths!
  – Keeps numbers of parameters in check
Recurrent Neural Network
Recurrent Neural Network

RNN

usually want to predict a vector at some time steps

x

y

h

"cell"

Recceent unit of computation

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Recurrent Neural Network

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

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

\text{new state} \quad \text{old state} \quad \text{input vector at some time step} \\
\text{some function with parameters } W

$$y = f_{w_2}(h_t)$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Recurrent Neural Network

We can process a sequence of vectors \( x \) 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.
(Vanilla) Recurrent Neural Network

The state consists of a single “hidden” vector $h$:

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

$$y_t = \operatorname{softmax}(W_{hy}h_t + b_y)$$

Sometimes called a “Vanilla RNN” or an “Elman RNN” after Prof. Jeffrey Elman
RNN: Computational Graph
RNN: Computational Graph

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

Re-use the same weight matrix at every time-step

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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Plan for Today

- Recurrent Neural Networks (RNNs)
  - Example Problem: (Character-level) Language modeling
  - Learning: (Truncated) BackProp Through Time (BPTT)
  - Visualizing RNNs
  - Example: Image Captioning
  - Inference: Beam Search
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)
Language Modeling

• Given a dataset, build an accurate model:

\[
P(y_1, y_2, \ldots y_T | \theta)
\]

\[
P(y_t | y_1, \ldots y_{t-1})
\]
Example:
Character-level Language Model

Vocabulary: \([h,e,l,o]\) = \(\checkmark\)

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

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

\[ h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h) \]
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

- Local vs Distributed

(a)

no pattern

\[ \cdot \cdot \cdot \cdot \cdot \]

\[ \cdot \cdot \cdot \cdot \cdot \]

\[ \cdot \cdot \cdot \cdot \cdot \]

\[ \cdot \cdot \cdot \cdot \cdot \]

\[ \cdot \cdot \cdot \cdot \cdot \]
Distributed Representations Toy Example

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

Local: ● ● ○ ● = VR + HR + HE = ?

Distributed: ● ● ○ ● = V + H + E ≈ ○
Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Training Time: \[ \text{MLE} / \text{“Teacher Forcing”} \]

Example:
Character-level
Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Let's do Monday.

Monday works for me.

Either day works for me.
Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Truncated Backpropagation through time

Run forward and backward through chunks of the sequence instead of whole sequence

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
**Truncated** Backpropagation through time

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.
Truncated Backpropagation through time
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 decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud distant that content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.
at first:

"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 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.
The Stacks Project: open source algebraic geometry textbook

http://stacks.math.columbia.edu/
The stacks project is licensed under the GNU Free Documentation License

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
For $\bigoplus_{n=1,\ldots,m}$ where $L_n = 0$, hence we can find a closed subset $H$ in $X$ and any set $S_{\text{empty}}$ on $X$, $U$ is a closed immersion of $S$ in $X$, then $U \to T$ is a separated algebraic space.

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

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

and the comparably in the fibre product covering we have to prove the lemma generated by $\prod Z \times U \to V$. 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 $\text{Spec}(R) \to S$ 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 $O_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $O_{X,x'} \to O_{X,x''}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $GL_{S}(x', x'')$ and we win.

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

$$\tilde{M} = \mathcal{T} \otimes_{\text{Spec}(k)} O_{S,x} - i^{-1} X$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}, (\text{Sch}/S)_{fppf}$$

and

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

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 ???. It may replace $S$ by $X_{\text{spaces, étale}}$ which gives an open subspace of $X$ and $T$ equal to $S_{\text{zar}}$, 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 $\text{Proj}_{X}(A) = \text{Spec}(B)$ over $U$) compatible with the complex

$$\text{Set}(A) = \Gamma(X, O_{X,x})$$

When in this case of to show that $Q \to C_{X/X}$ is stable under the following result in the second conditions of (1), (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 $S = \text{Spec}(R)$ 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 $U \cap U = \prod_{i=1,\ldots,n} U_i$ 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 $F_{x_0} = F_{x_0} = F_{x_{0,\ldots,0}}$.

Lemma 0.2. Let $X$ be a locally Noetherian scheme over $S$, $E = F_{X/S}$. Set $I = \mathcal{O}_1 \subset \mathcal{I}_n$. Since $T^n \subset T^n$ are nonzero over $i_0 \leq p$ is a subset of $J_{n,0} \circ \delta_{2}$ works.

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

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

$$D(O_X) = O_X(D)$$

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

\[ \square \]
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

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

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

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X}(\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ 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 $U \subset X$ 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

$$b: X \rightarrow Y' \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X$$

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 $X = \text{Spec}(R)$ 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"

$$\mathcal{O}_{X,S} \rightarrow \mathcal{F}_S \rightarrow -1(\mathcal{O}_{X_{\text{etale}}}) \rightarrow \mathcal{O}_{X,S}(\mathcal{O}_{X,S})$$

is an isomorphism of covering of $\mathcal{O}_X$. 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 $\mathcal{O}_{X,S}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ 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 & 0x00000000ffffff8) & 0x000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x2000000);
        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>
#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>

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

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

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \n    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

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

```c
/* 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

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 crowd 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

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
Searching for interpretable cells

Cell that turns on inside comments and quotes:

```
/* 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;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* 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\n",
            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

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

code depth 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
Plan for Today

• Recurrent Neural Networks (RNNs)
  – Example Problem: (Character-level) Language modeling
  – Learning: (Truncated) BackProp Through Time (BPTT)
  – Visualizing RNNs
  – Example: Image Captioning
  – Inference: Beam Search
  – Multilayer RNNs
  – Problems with gradients in “vanilla” RNNs
  – LSTMs (and other RNN variants)
Multilayer RNNs

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

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

\[ h_t = \tanh(W_{hh} h_{t-1} + W_{hx} x_t) \]

\[ = \tanh \left( (W_{hh} \quad W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]

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

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

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

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

$$= \tanh \left( (W_{hh} \ W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

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

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```
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

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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Meet LSTMs

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

• Cell State / Memory

(C) Dhruv Batra

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

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

• Forget that + memorize this

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

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

\[
o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)\\
h_t = o_t \ast \tanh(C_t)
\]
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!

(C) Dhruv Batra  Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs

- A pretty sophisticated cell
LSTM Variants #1: Peephole Connections

- Let gates see the cell state / memory

$$f_t = \sigma \left( W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f \right)$$
$$i_t = \sigma \left( W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i \right)$$
$$o_t = \sigma \left( W_o \cdot [C_t, h_{t-1}, x_t] + b_o \right)$$
LSTM Variants #2: Coupled Gates

- Only memorize new if forgetting old

\[ C_t = f_t \cdot C_{t-1} + (1 - f_t) \cdot \tilde{C}_t \]
LSTM Variants #3: Gated Recurrent Units

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

\[
\begin{align*}
z_t &= \sigma \left(W_z \cdot [h_{t-1}, x_t]\right) \\
r_t &= \sigma \left(W_r \cdot [h_{t-1}, x_t]\right) \\
\tilde{h}_t &= \tanh \left(W \cdot [r_t \cdot h_{t-1}, x_t]\right) \\
h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t
\end{align*}
\]
Other RNN Variants

[An Empirical Exploration of Recurrent Network Architectures, Jozefowicz et al., 2015]

\[
\text{MUT1:} \\
\begin{align*}
z &= \text{sigm}(W_{xz}x_t + b_z) \\
r &= \text{sigm}(W_{xr}x_t + W_{ht}h_t + b_r) \\
h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z \\
&\quad + h_t \odot (1 - z)
\end{align*}
\]

\[
\text{MUT2:} \\
\begin{align*}
z &= \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z) \\
r &= \text{sigm}(x_t + W_{hr}h_t + b_r) \\
h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\
&\quad + h_t \odot (1 - z)
\end{align*}
\]

\[
\text{MUT3:} \\
\begin{align*}
z &= \text{sigm}(W_{xz}x_t + W_{hz} \tanh(h_t) + b_z) \\
r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\
h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\
&\quad + h_t \odot (1 - z)
\end{align*}
\]
Plan for Today

- Recurrent Neural Networks (RNNs)
  - Example Problem: (Character-level) Language modeling
  - Learning: (Truncated) BackProp Through Time (BPTT)
  - Visualizing RNNs
  - Example: Image Captioning
  - Inference: Beam Search
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)
Neural Image Captioning

Image Embedding (VGGNet)

- Convolution Layer + Non-Linearity
- Pooling Layer
- Convolution Layer + Non-Linearity
- Pooling Layer
- Fully-Connected MLP

4096-dim
Neural Image Captioning

Image Embedding (VGGNet)

Convolution Layer + Non-Linearity

Pooling Layer

Convolution Layer + Non-Linearity

Pooling Layer

Fully-Connected MLP

4096-dim
Neural Image Captioning

<start>  Two  people  and  two  horses.
Neural Image Captioning

<start> Two people and two horses.
Sequence Model Factor Graph

$$P(y_t \mid y_1, \ldots, y_{t-1})$$
Beam Search Demo

- http://dbs.cloudcv.org/captioning&mode=interactive
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 Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Captioning: Failure Cases

A woman is holding a cat in her hand

A man in a baseball uniform throwing a ball

A bird is perched on a tree branch

A person holding a computer mouse on a desk

A woman standing on a beach holding a surfboard

Captions generated using neuraltalk2
All images are CC0 Public domain:
- fur coat
- handstand
- spider web
- baseball

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Typical VQA Models

Image Embedding (VGGNet)

Convolution Layer + Non-Linearity
Pooling Layer
Convolution Layer + Non-Linearity
Pooling Layer
Fully-Connected MLP

4096-dim

Neural Network Softmax over top K answers

Question Embedding (LSTM)

“How many horses are in this image?”
Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don’t work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed.