Why do we need them?

A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”, 2012.
Recurrent Neural Networks (RNNs)

- RNNs are very powerful because:
  - Distributed hidden state that allows them to store a lot of information about the past efficiently.
  - Non-linear dynamics that allows them to update their hidden state in complicated ways.

- With enough neurons and time, RNNs can compute anything that can be computed by your computer.

- More formally, RNNs are Turing complete.

Adapted from Hinton
Unfolding

Feed-forward networks

Recurrent networks

Previously on CHIC 783!
Previously on CENG 783:

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Reminder: Backpropagation with weight constraints

It is easy to modify the backprop algorithm to incorporate linear constraints between the weights.

- We compute the gradients as usual, and then modify the gradients so that they satisfy the constraints.
  - So if the weights started off satisfying the constraints, they will continue to satisfy them.

To constrain: \( w_1 = w_2 \)

We need: \( \Delta w_1 = \Delta w_2 \)

Compute: \( \frac{\partial E}{\partial w_1} \) and \( \frac{\partial E}{\partial w_2} \)

Use \( \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \) for \( w_1 \) and \( w_2 \)
Backpropagation through time

- We can think of the recurrent net as a layered, feed-forward net with shared weights and then train the feed-forward net with weight constraints.

- We can also think of this training algorithm in the time domain:
  - The forward pass builds up a stack of the activities of all the units at each time step.
  - The backward pass peels activities off the stack to compute the error derivatives at each time step.
  - After the backward pass we add together the derivatives at all the different times for each weight.
The problem of exploding or vanishing gradients

- What happens to the magnitude of the gradients as we backpropagate through many layers?
  - If the weights are small, the gradients shrink exponentially.
  - If the weights are big the gradients grow exponentially.

- Typical feed-forward neural nets can cope with these exponential effects because they only have a few hidden layers.

- In an RNN trained on long sequences (e.g. 100 time steps) the gradients can easily explode or vanish.
  - We can avoid this by initializing the weights very carefully.

- Even with good initial weights, it's very hard to detect that the current target output depends on an input from many time-steps ago.
  - So RNNs have difficulty dealing with long-range dependencies.
Exploding and vanishing gradients problem

• Solution 1: Gradient clipping for exploding gradients:

  **Algorithm 1** Pseudo-code for norm clipping
  
  $g = \frac{\partial E}{\partial \theta}$
  
  if $\|g\| \geq$ threshold
  
  $\hat{g} = \frac{\text{threshold}}{\|g\|} g$
  
  end if

  • For vanishing gradients: Regularization term that penalizes changes in the magnitudes of back-propagated gradients

  $\Omega = \sum_k \Omega_k = \sum_k \left( \left\| \frac{\partial E}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial x_k} \right\| - 1 \right)^2$

---

*Figure 6. We plot the error surface of a single hidden unit recurrent network, highlighting the existence of high curvature walls. The solid lines depict standard trajectories that gradient descent might follow. Using dashed arrow the diagram shows what would happen if the gradients is rescaled to a fixed size when its norm is above a threshold.*
Exploding and vanishing gradients problem

- Solution 2:
  - Use methods like LSTM
Meet LSTMs

- How about we explicitly encode memory?
LSTM in detail

- We first compute an activation vector, $a$:
  \[ a = W_x x_t + W_h h_{t-1} + b \]
- Split this into four vectors of the same size:
  \[ a_i, a_f, a_o, a_g \leftarrow a \]
- We then compute the values of the gates:
  \[ i = \sigma(a_i) \quad f = \sigma(a_f) \quad o = \sigma(a_o) \quad g = \tanh(a_g) \]
  where $\sigma$ is the sigmoid.
- The next cell state $c_t$ and the hidden state $h_t$:
  \[ c_t = f \odot c_{t-1} + i \odot g \quad h_t = o \odot \tanh(c_t) \]
  where $\odot$ is the element-wise product of vectors.

Eqs: Karpathy
Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.
Example: Character-level Text Modeling
A simple scenario

- Alphabet: h, e, l, o
- Text to train to predict: “hello”
Sample predictions (when trained on the works of *Shakespeare*):

- 3-level RNN with 512 hidden nodes in each layer

---

PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attaing'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.
Sample predictions (when trained on Wikipedia):

- Using LSTM

Naturalism and decision for the majority of Arab countries' capital was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzhou's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Imminences]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servacious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963879.htm Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Sample predictions (when trained on Latex documents):

- Using multi-layer LSTM

For $\bigoplus_{n=1}^{m}$, where $L_{\omega} = 0$, hence we can find a closed subset $H$ in $H$ and any sets $T$ 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 compatibility in the fibre product covering we have to prove the lemma generated by $\prod_{T \to U} U \to V$. Consider the maps $M$ along the set of points $\mathcal{S}_{T \to U}$ 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 $\mathcal{S}(G)$ such that $\text{Spec}(R) \to S$ is smooth or an

$U = \bigcup_{i=1}^{m} U_i \times_S U_i$

which has a nonzero morphism we may assume that $f_1$ is of finite presentation over $S$. We claim that $O_{X,x}$ is a scheme where $x,x',x'' \in S$ such that $O_{X,x} \to O_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $GL_s^B(x/S')$ and we win. $\square$

To prove we see that $F|_U$ is a covering of $X$. Let $T_i$ be an object of $S_{X/S}$ for $i > 0$ and $F_i$ 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

$\overline{M} = \mathcal{T} = \mathcal{T} \otimes_{\text{Spec}(R)} O_{S,s} - \Gamma_j(F)$

is a unique morphism of algebraic stacks. Note that

Arrows $= (\text{Sch}/S)^{pp}_{S_{X/S}}(S_{X/S})$

and

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

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. $\square$

The result for prove any open covering follows from the less of Example ???. It may replace $S$ by $X_{spaces,\text{etale}}$ 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 (9) 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

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

When in this case of to show that $\mathcal{O} \to \mathcal{O}_{S/X}$ 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 $S = \text{Spec}(R)$ 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 $U \cap U = \coprod_{i=1}^{n} U_i$ be the scheme $X$ over $S$ at the schemes $X_i \to X$ and $U = \lim_i X_i$.

The following lemma surjective rest decomposes of this implies that $F_{x_0} = F_{x_0} = F_{x_1 \ldots 0}$.

Lemma 0.2. Let $X$ be a locally Noetherian scheme over $S$, $E = \mathcal{F}_{X/S}$. Set $\mathcal{L} = \mathcal{L}_1 \subset \mathcal{L}_2$. Since $T^0 \subset T^0$ are nonzero over $t \leq p$ is a subset of $\mathcal{F}_{A_1 \ldots 0}$ 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 $S$. $\square$

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
He was elected President during the Revolutionary War and forgave Opus Paul at Rome. The regime of his crew of England, is now Arab women's icons in and the demons that use something between the characters' sisters in lower coil trains were always operated on the line of the ephemeral street, respectively, the graphic or other facility for deformation of a given proportion of large segments at RTUS). The B every chord was a "strongly cold internal palette pour even the white blade."
Some completions produced by the model

- Sheila thrunge\textsuperscript{s} (most frequent)
- People thrunge (most frequent next character is space)
- Shiela, Thrungelini del Rey (first try)
- The meaning of life is literary recognition. (6\textsuperscript{th} try)

- The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. (one of the first 10 tries for a model trained for longer).
RNNs for predicting the next word

- Tomas Mikolov and his collaborators have trained quite large RNNs on quite large training sets using BPTT.
  - They do better than feed-forward neural nets.
  - They do better than the best other models.
  - They do even better when averaged with other models.

- RNNs require much less training data to reach the same level of performance as other models.

- RNNs improve faster than other methods as the dataset gets bigger.
  - This is going to make them very hard to beat.
Word-level RNN for news title generation

https://github.com/larspars/word-rnn
Today

- Finalize RNNs
  - Language modeling with words
  - Word embedding
  - Example Application: Image Captioning
  - Example Application: Language Translation

- NOTES:
  - Final Exam date: 3 January, 17:40 – BMB1
  - Lecture on 31st of December (Monday)
  - Project deadline (w/o Incomplete period): 26th of January
  - Project deadline (w Incomplete period): 2nd of February
  - ICLR 2019 Reproducibility challenge
Word Embedding (word2vec)

Fig: http://www.languagejones.com/blog-1/2015/11/1/word-embedding
Why do we embed words?

- 1-of-n encoding is not suitable to learn from
  - It is sparse
  - Similar words have different representations
  - Compare the pixel-based representation of images: Similar images/objects have similar pixels

- Embedding words in a map allows
  - Encoding them with fixed-length vectors
  - “Similar” words having similar representations
  - Allows complex reasoning between words:
    - king - man + woman = queen

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>NEAREST TOKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris - France + Italy</td>
<td>Rome</td>
</tr>
<tr>
<td>bigger - big + cold</td>
<td>colder</td>
</tr>
<tr>
<td>sushi - Japan + Germany</td>
<td>bratwurst</td>
</tr>
<tr>
<td>Cu - copper + gold</td>
<td>Au</td>
</tr>
<tr>
<td>Windows - Microsoft + Google</td>
<td>Android</td>
</tr>
<tr>
<td>Montréal Canadiens - Montreal + Toronto</td>
<td>Toronto Maple Leafs</td>
</tr>
</tbody>
</table>

More examples

http://deeplearning4j.org/word2vec
More examples

Male-Female

Verb tense

Country-Capital
More examples

• Geopolitics: *Iraq* - *Violence* = *Jordan*
• Distinction: *Human* - *Animal* = *Ethics*
• *President* - *Power* = *Prime Minister*
• *Library* - *Books* = *Hall*

http://deeplearning4j.org/word2vec
Using word embeddings

- E.g., for language modeling

- Given “I am eating …”, a language model can predict what can come next.
  - This both requires syntax and semantics (context)

- Before deep learning, n-gram (2-gram, 3-gram) models were state of the art.

Fig: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/
Using word embeddings

Fig: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/
Using word embeddings

Fig: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/
word2vec

- “Similarity” to Sweden (cosine distance between their vector representations)

<table>
<thead>
<tr>
<th>Word</th>
<th>Cosine distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>norway</td>
<td>0.760124</td>
</tr>
<tr>
<td>denmark</td>
<td>0.715460</td>
</tr>
<tr>
<td>finland</td>
<td>0.620022</td>
</tr>
<tr>
<td>switzerland</td>
<td>0.588132</td>
</tr>
<tr>
<td>belgium</td>
<td>0.585835</td>
</tr>
<tr>
<td>netherlands</td>
<td>0.574631</td>
</tr>
<tr>
<td>iceland</td>
<td>0.562368</td>
</tr>
<tr>
<td>estonia</td>
<td>0.547621</td>
</tr>
<tr>
<td>slovenia</td>
<td>0.531408</td>
</tr>
</tbody>
</table>

http://deeplearning4j.org/word2vec
Two different ways to train

1. Using context to predict a target word (~ continuous bag-of-words)

2. Using word to predict a target context (skip-gram)
   - Produces more accurate results on large datasets
   - If the vector for a word cannot predict the context, the mapping to the vector space is adjusted
   - Since similar words should predict the same or similar contexts, their vector representations should end up being similar

http://deeplearning4j.org/word2vec
Two different ways to train

1. Using context to predict a target word (~ continuous bag-of-words)

https://www.tensorflow.org/versions/r0.7/tutorials/word2vec/index.html
Two different ways to train

2. Using word to predict a target context (skip-gram)
   • Produces more accurate results on large datasets
   
   • Given a sentence:
     
     the quick brown fox jumped over the lazy dog
   
   • For each word, take context to be
     
     (N-words to the left, N-words to the right)
   
   • If $N = 1$ (context, word):
     
     ([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...
Some details

• CBOW is called continuous BOW since the context is regarded as a BOW and it is continuous.

• In both approaches, the networks are composed of linear units

• The output units are usually normalized with the softmax

• According to Mikolov:
  • “Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.

  • CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words”
An extra issue: Sampling

- Greedy sampling: Take the most likely word at each step
  vs
- Beam search (or alternatives): Consider $k$ most likely words at each step, and expand search.
Example: Image Captioning

Fig: https://github.com/karpathy/neuraltalk2
Demo video

https://vimeo.com/146492001
Overview

Pre-trained CNN (e.g., on imagenet)

Pre-trained word embedding is also used

Image: Karpathy
Training

before:
\[ h_0 = \max(0, Wxh \times x_0) \]

now:
\[ h_0 = \max(0, Wxh \times x_0 + W_{ih} \times v) \]
<START>
sample!  <END> token => finish.
Example: Neural Machine Translation
Neural Machine Translation

- Model
Neural Machine Translation

- Model

Each box is an LSTM or GRU cell.

Sutskever et al. 2014

Haitham Elmarakeby
Neural Machine Translation

- Model: encoder

\[ f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .} ) \]

\[ e = (\text{Economic, growth, has, slowed, down, in, recent, years, .} ) \]

Cho: From Sequence Modeling to Translation

Haitham Elmarakeby
Neural Machine Translation

$e = \text{(Economic, growth, has, slowed, down, in, recent, years, .)}$
Neural Machine Translation

\[ f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .}) \]

- Model-decoder

\[ e = (\text{Economic, growth, has, slowed, down, in, recent, years, .}) \]
Decoder in more detail

Given

(i) the “summary” \( (c) \) of the input sequence,

(ii) the previous output / word \( (y(t-1)) \)

(iii) the previous state \( (h(t-1)) \)

the hidden state of the decoder is:

\[
h(t) = f(h(t-1), y(t-1), c)
\]

Then, we can find the most likely next word:

\[
P(y(t) \mid y(t-1), y(t-2), \ldots, c) = g(h(t), y(t-1), c)
\]

\( f, g \): activation functions of our choice. For \( g \), we need a function that maps to probabilities, e.g., softmax.
Encoder-decoder

- Jointly trained to maximize

\[
\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_{\theta}(y_n \mid x_n),
\]
Check the following tutorial

- http://smerity.com/articles/2016/google_nmt_arch.html
More on attention

Attention mechanism: A two-layer neural network.
Input: $z_i$ and $h_j$
Output: $e_j$, a scalar for the importance of word $j$.
The scores of words are normalized: $a_j = \text{softmax}(e_j)$
More on attention
NMT can be done at char-level too

This can be done with CNNs