Neural Architectures for Image, Language, and Speech Processing (Cont.)

Karl Stratos

June 27, 2018

Overview

- Feedforward Networks
  - Need for Specialized Architectures
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
  - Long Short-Term Memory Networks (LSTMs)
  - Example: Bidirectional LSTM Network for POS Tagging
- Encoder-Decoder Models
  - Example: RNN-Based Seq2Seq
  - Bonus: Connectionist Temporal Classification (CTC)

General Idea

Much of machine learning: given **some complicated structure** x, predict **some complicated structure** y

General Idea

Much of machine learning: given **some complicated structure** x, predict **some complicated structure** y

Machine translation:

$$x = \texttt{Le programme a été mis en application}$$

$$y = \texttt{And the programme has been implemented}$$

General Idea

Much of machine learning: given **some complicated structure** x, predict **some complicated structure** y

Machine translation:

$$x = \texttt{Le programme a été mis en application}$$
$$y = \texttt{And the programme has been implemented}$$

Encoder-decoder models are **conditional** models that handle this wide class of problems in two steps:

1. **Encode** the given input x using some architecture.
2. **Decode** y, typically in a sequential manner using an RNN.

Overview

- Feedforward Networks
  - Need for Specialized Architectures
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
  - Long Short-Term Memory Networks (LSTMs)
  - Example: Bidirectional LSTM Network for POS Tagging
- Encoder-Decoder Models
  - Example: RNN-Based Seq2Seq
  - Bonus: Connectionist Temporal Classification (CTC)

Basic Seq2Seq Framework

Model parameters

- Vector $e_x \in \mathbb{R}^d$ for every $x \in V^{\text{src}}$
- Vector $e_y \in \mathbb{R}^d$ for every $y \in V^{\text{trg}} \cup \{*\}$
- Encoder RNN $\psi : \mathbb{R}^d \times \mathbb{R}^{d'} \to \mathbb{R}^{d'}$ for V^{src}
- Decoder RNN $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \to \mathbb{R}^{d'}$ for V^{trg}
- Feedforward $f : \mathbb{R}^{d'} \to \mathbb{R}^{|V^{\text{trg}}|} + 1$

Basic Seq2Seq Framework

Model parameters

- Vector $e_x \in \mathbb{R}^d$ for every $x \in V^{\text{src}}$
- Vector $e_y \in \mathbb{R}^d$ for every $y \in V^{\text{trg}} \cup \{*\}$
- Encoder RNN $\psi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ for V^{src}
- Decoder RNN $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ for V^{trg}
- Feedforward $f : \mathbb{R}^{d'} \rightarrow \mathbb{R}^{|V^{\text{trg}}| + 1}$

Basic idea

1. Transform $x_1 \ldots x_m \in V^{\text{src}}$ with ψ into some representation ξ.
2. Build a sequence model ϕ over V^{trg} conditioning on ξ.

Encoder

For $i = 1 \ldots m$,

$$h_i^{\psi} = \psi\left(e_{x_i}, h_{i-1}^{\psi}\right)$$

Encoder

For $i = 1 \dots m$,

$$h_i^{\psi} = \psi\left(e_{x_i}, h_{i-1}^{\psi}\right)$$

$$h_m^{\psi} = \psi\left(e_{x_m}, \psi\left(e_{x_{m-1}}, \psi\left(e_{x_{m-2}}, \cdots \psi\left(e_{x_1}, h_0^{\psi}\right) \cdots\right)\right)\right)$$

Decoder

Initialize $h_0^\phi = h_m^\psi$ and $y_0 = *$.

Decoder

Initialize $h_0^{\phi} = h_m^{\psi}$ and $y_0 = *$.

For $i = 1, 2, \ldots$, the decoder defines a probability distribution over $V^{\text{trg}} \cup \{\texttt{STOP}\}$ as ($\oplus$ denotes vector concatenation)

$$h_i^{\phi} = \phi\left(e_{y_{i-1}} \oplus h_m^{\psi},\ h_{i-1}^{\phi}\right)$$

$$p_{\Theta}(y|x_1 \ldots x_m,\ y_0 \ldots y_{i-1}) = \mathsf{softmax}_y(f(h_i^{\phi}))$$

Decoder

Initialize $h_0^{\phi} = h_m^{\psi}$ and $y_0 = *$.

For $i = 1, 2, \ldots$, the decoder defines a probability distribution over $V^{\text{trg}} \cup \{\texttt{STOP}\}$ as ($\oplus$ denotes vector concatenation)

$$h_i^{\phi} = \phi\left(e_{y_{i-1}} \oplus h_m^{\psi},\ h_{i-1}^{\phi}\right)$$

$$p_{\Theta}(y|x_1 \ldots x_m,\ y_0 \ldots y_{i-1}) = \mathsf{softmax}_y(f(h_i^{\phi}))$$

Probability of translation $y_1 \ldots y_n$ given $x_1 \ldots x_m$:

$$p_{\Theta}(y_1 \ldots y_n|x_1 \ldots x_m) = \prod_{i=1}^{n} p_{\Theta}(y_i|x_1 \ldots x_m,\ y_0 \ldots y_{i-1}) \times p_{\Theta}(\texttt{STOP}|x_1 \ldots x_m,\ y_0 \ldots y_n)$$

Training

Given parallel text of N sentence-translation pairs $(x^{(1)}, y^{(1)}) \ldots (x^{(N)}, y^{(N)})$, find parameters Θ^* that maximize the log likelihood of the data:

$$\Theta^* \approx \arg\min_{\Theta} \underbrace{- \sum_{i=1}^{N} \log p_{\Theta}(y^{(i)} | x^{(i)})}_{\text{loss}}$$

Greedy Translation

Given sentence $x_1 \ldots x_m \in V^{\textsf{src}}$,

1. Encode the sentence: for $i = 1 \ldots m$,

$$h_i^{\psi} = \psi\left(e_{x_i}, h_{i-1}^{\psi}\right)$$

Greedy Translation

Given sentence $x_1 \dots x_m \in V^{\text{src}}$,

1. Encode the sentence: for $i = 1 \dots m$,

$$h_i^\psi = \psi\left(e_{x_i}, h_{i-1}^\psi\right)$$

2. Initialize $h^\phi \leftarrow h_m^\psi$ and $S \leftarrow [*]$.

Greedy Translation

Given sentence $x_1 \dots x_m \in V^{\text{src}}$,

1. Encode the sentence: for $i = 1 \dots m$,

$$h_i^{\psi} = \psi\left(e_{x_i}, h_{i-1}^{\psi}\right)$$

2. Initialize $h^{\phi} \leftarrow h_m^{\psi}$ and $S \leftarrow [*]$.
3. Keep repeating

$$h^{\phi} \leftarrow \phi(e_{S[-1]} \oplus h_m^{\psi}, h^{\phi})$$
$$y \leftarrow \arg\max_{y \in V^{\text{trg}} \cup \{\texttt{STOP}\}} \text{softmax}_y\left(f(h^{\phi})\right)$$
$$S \leftarrow S + [y]$$

until $y = \texttt{STOP}$.

Decoder with Attention

- Instead of using 1 fixed vector to encode all $x_1 \ldots x_m$, **decoder decides which words to pay attention to**.

Decoder with Attention

- Instead of using 1 fixed vector to encode all $x_1 \dots x_m$, **decoder decides which words to pay attention to**.

- For $i = 1, 2, \dots$,

$$h_i^{\phi} = \phi\left(e_{y_{i-1}} \oplus \left(\sum_{j=1}^{m} \alpha_{i,j} h_j^{\psi}\right),\ h_{i-1}^{\phi}\right)$$

$$p_{\Theta}(y | x_1 \dots x_m,\ y_0 \dots y_{i-1}) = \mathsf{softmax}_y(f(h_i^{\phi}))$$

Attention Weights

$$\sum_{j=1}^{m} \alpha_{i,j} h_j^{\psi}$$

- $\alpha_{i,j}$: Importance of x_j for predicting i-th translation

Attention Weights

$$\sum_{j=1}^{m} \alpha_{i,j} h_j^{\psi}$$

- $\alpha_{i,j}$: Importance of x_j for predicting i-th translation
- Various options

$$\beta_{i,j} = u^\top \tanh\left(W h_{i-1}^{\phi} + V h_j^{\psi}\right)$$

$$\beta_{i,j} = \left(h_{i-1}^{\phi}\right)^\top h_j^{\psi}$$

$$\beta_{i,j} = \left(h_{i-1}^{\phi}\right)^\top B h_j^{\psi}$$

Typically take softmax to make them probabilities:

$$(\alpha_{i,1} \dots \alpha_{i,m}) = \text{softmax}\,(\beta_{i,1} \dots \beta_{i,m})$$

Overview

- Feedforward Networks
  - Need for Specialized Architectures
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
  - Long Short-Term Memory Networks (LSTMs)
  - Example: Bidirectional LSTM Network for POS Tagging
- Encoder-Decoder Models
  - Example: RNN-Based Seq2Seq
  - Bonus: Connectionist Temporal Classification (CTC)

CTC in Speech

- CTC is an approach to handle the following setting.
  - Training time: given a pair of sequences $(\boldsymbol{x}, \boldsymbol{y})$ where the length of $\boldsymbol{y}$ is shorter than $\boldsymbol{x}$.
  - Test time: must map any input sequence $\boldsymbol{x}$ to a corresponding sequence $\boldsymbol{y}$.
- CTC treats this problem as a latent-variable model in which there is an intermediate sequence $\boldsymbol{z}$ with the same length as $\boldsymbol{x}$ from which $\boldsymbol{y}$ can be retrieved.
- Has been a dominant approach in speech recognition.
  - Alternatively, can we just use seq2seq for this problem?

Input-Latent-Output Example

$$\boldsymbol{x} = x_1\ x_2\ x_3\ x_4\ x_5\ x_6 \qquad x_t \in \mathbb{R}^d$$

$$\boldsymbol{z} = \text{t e e l } \epsilon \text{ l} \qquad z_t \in \mathcal{C} \cup \{\epsilon\}$$

$$\boldsymbol{y} = \text{t e l l} \qquad y_i \in \mathcal{C}$$

Other possible $\boldsymbol{z}$ sequences

$$\boldsymbol{z} = \epsilon \text{ t e l } \epsilon \text{ l}$$

$$\boldsymbol{z} = \text{t e l } \epsilon\ \epsilon \text{ l}$$

$$\boldsymbol{z} = \text{t e l } \epsilon \text{ l } \epsilon$$

$$\vdots$$

CTC Model

- Encode $\boldsymbol{x} = x_1 \ldots x_T$ into vectors $h_1 \ldots h_T \in \mathbb{R}^{|\mathcal{C}|+1}$ (e.g., by CNN/RNN/CNN+RNN)
- The model defines the probability of $z_t \in \mathcal{C} \cup \{\epsilon\}$ independently of other z_l (conditioning on $\boldsymbol{x}$) as

$$p(z_t = z|\boldsymbol{x}) = \mathsf{softmax}_z(h_t)$$

CTC Training

$|\boldsymbol{x}| = T, |\boldsymbol{y}| = N$

- $p(\boldsymbol{y}|\boldsymbol{x})$ given by marginalizing over all $\boldsymbol{z}$ valid for $(\boldsymbol{x}, \boldsymbol{y})$

CTC Training

$|\boldsymbol{x}| = T, |\boldsymbol{y}| = N$

- $p(\boldsymbol{y}|\boldsymbol{x})$ given by marginalizing over all $\boldsymbol{z}$ valid for $(\boldsymbol{x}, \boldsymbol{y})$
- Given by $\pi(N, T)$ where

$$\pi(i, t) = \text{prob that we have consumed } y_1 \dots y_i \text{ from } x_1 \dots x_t$$

CTC Training

$|\boldsymbol{x}| = T, |\boldsymbol{y}| = N$

- $p(\boldsymbol{y}|\boldsymbol{x})$ given by marginalizing over all $\boldsymbol{z}$ valid for $(\boldsymbol{x}, \boldsymbol{y})$
- Given by $\pi(N, T)$ where

$$\pi(i, t) = \text{prob that we have consumed } y_1 \dots y_i \text{ from } x_1 \dots x_t$$

- Dynamic programming

$$\begin{aligned}
\pi(0,0) &= 1 & \\
\pi(i,0) &= 0 & \forall i \geq 1 \\
\pi(0,t) &= \prod_{k=1}^{t} p(z_t = \epsilon|\boldsymbol{x}) & \forall t \geq 1
\end{aligned}$$

CTC Training

$|\boldsymbol{x}| = T, |\boldsymbol{y}| = N$

- $p(\boldsymbol{y}|\boldsymbol{x})$ given by marginalizing over all $\boldsymbol{z}$ valid for $(\boldsymbol{x}, \boldsymbol{y})$
- Given by $\pi(N, T)$ where

$$\pi(i,t) = \text{prob that we have consumed } y_1 \dots y_i \text{ from } x_1 \dots x_t$$

- Dynamic programming

$$\begin{aligned}
\pi(0,0) &= 1 \\
\pi(i,0) &= 0 && \forall i \geq 1 \\
\pi(0,t) &= \prod_{k=1}^{t} p(z_t = \epsilon|\boldsymbol{x}) && \forall t \geq 1
\end{aligned}$$

$$\pi(i,t) = \pi(i,t-1)p(z_t = \epsilon|\boldsymbol{x}) + \pi(i-1,t-1)p(z_t = y_i|\boldsymbol{x})$$

CTC Training

$|\boldsymbol{x}| = T, |\boldsymbol{y}| = N$

- $p(\boldsymbol{y}|\boldsymbol{x})$ given by marginalizing over all $\boldsymbol{z}$ valid for $(\boldsymbol{x}, \boldsymbol{y})$
- Given by $\pi(N, T)$ where

$$\pi(i, t) = \text{prob that we have consumed } y_1 \dots y_i \text{ from } x_1 \dots x_t$$

- Dynamic programming

$$\pi(0,0) = 1$$

$$\pi(i,0) = 0 \qquad \forall i \geq 1$$

$$\pi(0,t) = \prod_{k=1}^{t} p(z_t = \epsilon|\boldsymbol{x}) \qquad \forall t \geq 1$$

$$\pi(i,t) = \pi(i, t-1)p(z_t = \epsilon|\boldsymbol{x}) + \pi(i-1, t-1)p(z_t = y_i|\boldsymbol{x})$$

- Minimize loss $-\log \pi(N, T)$.

CTC Test Time

- Given $\boldsymbol{x}$, we can predict $z_1 \ldots z_T \in \mathcal{C} \cup \{\epsilon\}$ using the model $p(z_t|\boldsymbol{x})$ either greedily or by beam search.

References

- CTC tutorial: `https://distill.pub/2017/ctc/`