CMPUT 367

Roderick Lan

Contents

Lecture 16 - Mar 14; RNN Architectures

1 CNN Recap

basic idea = sliding window that captures some spatial invariance info
Pooling - reduce information to one value

2 Recurrent Neural Netowrk

Feedback loop in hidden layer
"recurrent" path from prev iteration hidden layer

2.1 Generation

pass input through NN to get relations, then generate.
For translation (and other similar applications), multi modal output (ie. multiple possible "answers") becomes an issue (doesn't know what it already outputed; frankenstein outputs)

> Seq2Seq solves this (passes prev output to side branches)

Encoder RNN → input to get relations
Decoder RNN → generate output

1. Self generated word during inference
2. Ground truth taken of the previous step

Use 1. for inference task (decode)
Use 2. initially for training task; but gradually move to 1.
Often times easier to follow prefix; this type of training task easier than inference, so we want to insert some of the difficulty from inference into training (get experience from generation, and learn how to recover from poor generation in training)

2.2 Vanilla RNN

$$h^{(t)} = f(W_h h^{(t-1)} + W_x x^{(t)} + b)$$

Backprop through time
CLT - var grows linearly w/ number of variables
Gradient vanishing or explosion

> BP - lin sys.
>
> FP - non line sys (potentially chaotic; small change amplified to large (unbounded))

Whats Wrong?

Exact gradient isn't what we want. (if we have such a large gradient, we can't make training stable)

3 LSTM and GRU

Long Short Term Memory

Keeps a cell c_t and hidden state h_t
Input gate - sigmoid function, tells us how much information we need to pick up
Forget gate - sigmoid func, tells us how much information we should drop/forget

Gated Recurrent Unit

Keep cell and hidden state

4 RNN Usage

Can also be bidirectional

5 Prior

CNN

spatial neighborhood

capture info via. sliding window

RNN

ordered info

sequential processing

6 Parse Tree

Onion/nested sentences
logically connects words