Written Review : Convolutional LSTM Networks for Subcellular Localization of Proteins

(Charles Bournhonesque, David Golub, Charles Lu, Olivier Moindrot)

https://arxiv.org/pdf/1503.01919.pdf

Introduction

General goal: Analyse biological sequential data
Precise objective: Given the protein sequence, predict the **subcellular localization** of Proteins
Technique: LSTMs. They also try to improve results by adding **convolutional filters** and experimenting with **attention.**
Visualization:

- new ways to *visualize* the filters of convolutions
- use attention mechanism to visualize where an LSTM focuses
- goal: extract *biological knowledge*

Why: Usually SVMs or feed-forward NN used, but no real way of handling varying length sequences. Here come LSTMs.

Results: High accuracy prediction (0.902), better than state-of-the-art. Only use protein sequence and still better results than other models using other additional features. Can be trained with small (6k sequences) datasets

Model

- regular LSTM (R-LSTM)
  - 1D convolution with different filter sizes
    - 1, 3, 5, 9, 15 and 21
    - 10 filters for each size
  - bi-directional LSTM
  - output from the last hidden states of forward and backward
- Attention-LSTM (A-LSTM)
  - 1D convolution with different filter sizes
    - 1, 3, 5, 9, 15 and 21
    - 10 filters for each size
  - bi-directional LSTM

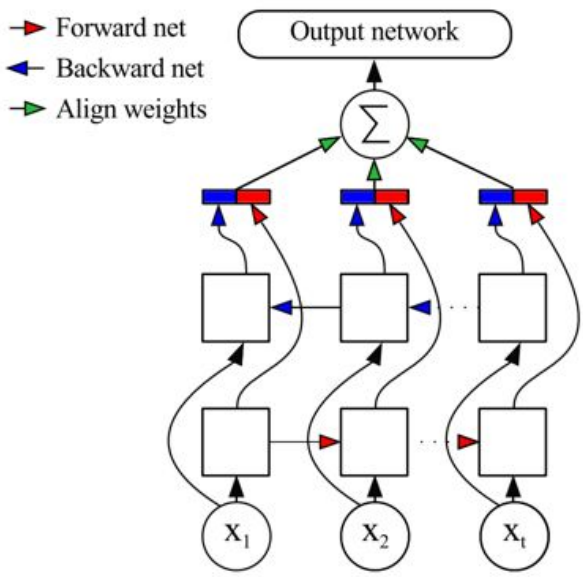

Figure 3. A-LSTM network. Each state of the hidden units, h_t are weighted and summed before the output network calculates the predictions.

- For each hidden state of the forward and backward,we compute a weight

$$a_t = \tanh(h_t W_a) v_a^{\mathrm{T}}$$

  - W_a is an attention hidden weight matrix
  - v_a is an attention output vector
  - Both are learnable parameters of the model
  - they used v_a of size 400
- we get a context vector from the weighted sum of hidden vectors (which are the concatenation from the forward and backward hidden states)
- we apply a fully connected layer and a softmax to get a final result from this context vector

What is the task / dataset?

- From the sequence of the protein, predict the sub-cellular location of the protein (cytoplasm, chloroplasts (only with plants), ER, extracellular space, lysosomes (only with animals), mitochondria, Golgi apparatus, peroxisomes, plasma membrane, and vacuoles (only with plants and fungi)
- input: protein sequence, truncated to size 1000
  - if the sequence is too long, remove the middle (because the terminal regions have more information (sorting signals))
  - if too short, pad middle with 0
  - one-hot encoding
  - 80 features per amino-acid
  - i.e input of size (batch_size, seq_length=1000, input_length=80)
- dataset: http://abi.inf.uni-tuebingen.de/Services/MultiLoc/multiloc_information
  - 5959 proteins
  - 11 output classes

Visualization

- For convolutional weights: use a PSSM (position specific scoring matrix) Lfilter*Lenc matrix
  - because the convolution was 1D, over the sequence
  - height of a column: position importance
  - height of a letter: amino acid importance
- For attention: plot the context vector (weighted sum of hidden vectors)
- For regular LSTM: plot the last hidden state
- t-SNE of the hidden representations
  - the clusters from regions close together in the cell are close together here

Training

- On Theano
- WIth gradient descent (ADAM) with learning rate 2e-4
- dropout of 50% on every layer
- 100 epochs
- 80% data for training, 20% for testing

Results

- Regular LSTM performs better than Attention LSTM
- Even better for an ensemble of R-LSTM (0.902 accuracy)

Comments

- with that little data (6000), difficult to have deeper models (too much overfit)
    - they need to apply 50% dropout to regularize
- the t-SNE doesn't bring a lot of info…
    - Obviously it will cluster the proteins with the same sub-cellular localization together, because we optimize on this…
    - it would have been informative to see that t-SNE cluster well proteins on a different task, while only trained for sub-cellular localization
- They mention that they use LSTMs because the sequences are of varying length, but they truncate everything to size 1000 and pad in the middle
- They could try using GRUs cells instead of LSTMs, which have fewer parameters and might work better on a small dataset
- Attention doesn't seem very useful considering that it always looks at the start/end of the sequence
    - this is also due to the fact that some sequence are padded with 0 in the middle so attention will never focus on the middle for them
    - it translates into worse performance for A-LSTM