Tutorial 8

- Neural machine translation
- Attention
- Dropout Layer

Tomer Golany – tomer.golany@gmail.com
Attention - Introduction
A recent trend in Deep Learning are Attention Mechanisms. ( 2016 )

Attention Mechanisms in Neural Networks are (very) loosely based on the visual attention mechanism found in humans.

Human visual attention models essentially come down to being able to focus on a certain region of an image with “high resolution” while perceiving the surrounding image in “low resolution”

And then adjusting the focal point over time.
What problem does Attention Solve

- To understand what attention can do for us, we will use Neural Machine Translation (NMT) as an example.

- In NMT, we map the meaning of a sentence into a fixed-length vector representation and then generate a translation based on that vector.

- Most NMT systems work by **encoding** the source sentence (e.g. a French sentence) into a vector using a Recurrent Neural Network,

- And then **decoding** an English sentence based on that vector, also using a RNN.
Neural machine translation
Neural machine translation:

See architecture on board
An encoder reads the input sentence \( x = (x_1, x_2, ..., x(T_x)) \) (sequence of vectors) into a vector \( c \) (the context).

The most common approach is to use an RNN.
The decoder is trained to predict the next word $y_t$ given the context vector $c$ and all the previously predicted words $\{y_1, \cdots, y_{t-1}\}$.

Defines a probability over the translation $y$ by decomposing the joint probability into the ordered conditionals:

$$p(y) = \prod_{t=1}^{T} p(y_t | \{y_1, \cdots, y_{t-1}\}, c)$$

Each conditional probability is modeled as:

$$p(y_t | \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$$

where $g$ is a nonlinear, potentially multi-layered function that outputs the probability of $y_t$ and $s_t$ is the hidden state of the RNN.
Neural machine translation:

- We can see that the decoder is supposed to generate a translation solely based on the last hidden state from the encoder.

- This vector must encode everything we need to know about the source sentence. \( h(t) \) must fully capture its meaning.

- In more technical terms, that vector is a **sentence embedding**.

- In fact, if you plot the embeddings of different sentences in a low dimensional space using PCA or t-SNE for dimensionality reduction, you can see that semantically similar phrases end up close to each other.
Neural machine translation:

- Still, it seems somewhat unreasonable to assume that we can encode all information about a potentially very long sentence into a single vector and then have the decoder produce a good translation based on only that.

- Let’s say your source sentence is 50 words long.

- The first word of the English translation is probably highly correlated with the first word of the source sentence.

- But that means decoder has to consider information from 50 steps ago, and that information needs to be somehow encoded in the vector.
Different approaches to solve the problem of long range dependencies:

- In theory, architectures like LSTM should be able to deal with this, but in practice long-range dependencies are still problematic.

- It turn out that reversing the source sequence (feeding it backwards into the encoder) produces significantly better results because it shortens the path from the decoder to the relevant parts of the encoder.

- Feeding a sequence twice also seems to help a network to better memorize things.

- Attention Mechanisms!
Attention mechanism
Attention mechanism:

- With an attention mechanism we no longer try encode the full source sentence into a fixed-length vector.

- We allow the decoder to “attend” to different parts of the source sentence at each step of the output generation.

- Importantly, we let the model learn what to attend to based on the input sentence and what it has produced so far.

- The decoder would probably choose to attend to things sequentially. Attending to the first word when producing the first English word, and so on.
Unlike the existing encoder–decoder approach, here the probability is conditioned on a distinct context vector $c_i$ for each target word $y_i$. The context vector $c_i$ depends on a sequence of annotations $(h_1, \ldots, h_{T_i})$ to which an encoder maps the input sentence. Each annotation $h_i$ contains information about the whole input sequence with a strong focus on the parts surrounding the $i$-th word of the input sequence. The context vector $c_i$ depends on a sequence of annotations $(h_1, \ldots, h_{T_i})$ to which an encoder maps the input sentence. Each annotation $h_i$ contains information about the whole input sequence with a strong focus on the parts surrounding the $i$-th word of the input sequence. The graphical illustration of the proposed model trying to generate the $i$-th target word $y_i$ given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$
\sum_{k=1}^{n} \exp (e_{ik})
$$

The alignment model which scores how well the inputs around position $j$ and the output at position $i$ match.

$$
e_{ij} = a(s_{i-1}, h_j)
$$

Just a single-layer multilayer perceptron

$$
a(s_{i-1}, h_j) = v_{\alpha}^T \tanh (W_{\alpha}s_{i-1} + U_{\alpha}h_j)
$$

does not depend on $i$, so pre-compute it in advance.
Attention mechanism:

- The above illustration uses a bidirectional recurrent network.

- The important part is that each decoder output word $Y_t$ now depends on a \textit{weighted combination of all the input states}, not just the last state.

- The $a$’s are weights that define in how much of each input state should be considered for each output.

- So, if $a_{3,2}$ is a large number, this would mean that the decoder pays a lot of attention to the second state in the source sentence while producing the third word of the target sentence.
Dropout layer
What is Dropout in Neural Networks?

The term “dropout” refers to dropping out units (both hidden and visible) in a neural network.

Dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random.

At each training stage, individual neurons are either dropped out of the net with probability $1-p$ or kept with probability $p$.

A reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.
Why do we need Dropout?

- To prevent Overfitting 😊

- A fully connected layer occupies most of the parameters.

- And hence, neurons develop co-dependency amongst each other during training which reduces the individual power of each neuron leading to over-fitting of training data.
Dropout is an approach to regularization in neural networks which helps reducing interdependent learning amongst the neurons.

Training Phase:

For each hidden layer, for each training sample, for each iteration:

– ignore (zero out) a random fraction, $p$, of nodes (and corresponding activations).
Dropout layer – Technical Details

(a) Standard Neural Net

(b) After applying dropout.
Testing Phase:
- Use all activations and all neurons.

Some Observations:
1. Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

2. Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.

3. With H hidden units, each of which can be dropped, we have $2^H$ possible models. In testing phase, the entire network is considered and each activation is reduced by a factor $p$. 
We build a deep network in Keras and tried to validate it on the CIFAR-10 dataset.

The deep network is built with three convolution layers of size 64, 128 and 256 followed by two fully connected layers of size 512 and an output layer layer of size 10.

Took ReLU as the activation function for hidden layers and sigmoid for the output layer.

Used the standard categorical cross-entropy loss.

Finally, used dropout in all layers and increase the fraction of dropout from 0.0 (no dropout at all) to 0.9 with a step size of 0.1 and ran each of those to 20 epochs.
 Dropout layer – Experiments

- Results:

![Accuracy vs Dropout Distribution]

Accuracy vs Dropout Distribution
Conclusions:

– with increasing the dropout, there is some increase in validation accuracy and decrease in loss initially before the trend starts to go down.

– There could be two reasons for the trend to go down if dropout fraction is 0.2:

1. 0.2 is actual minima for the this dataset, network and the set parameters used

2. More epochs are needed to train the networks.