Lesson 9

- Regularization
- Normalization
- Attention

Regularization

Overfitting

Neural Networks are prone to overfitting.

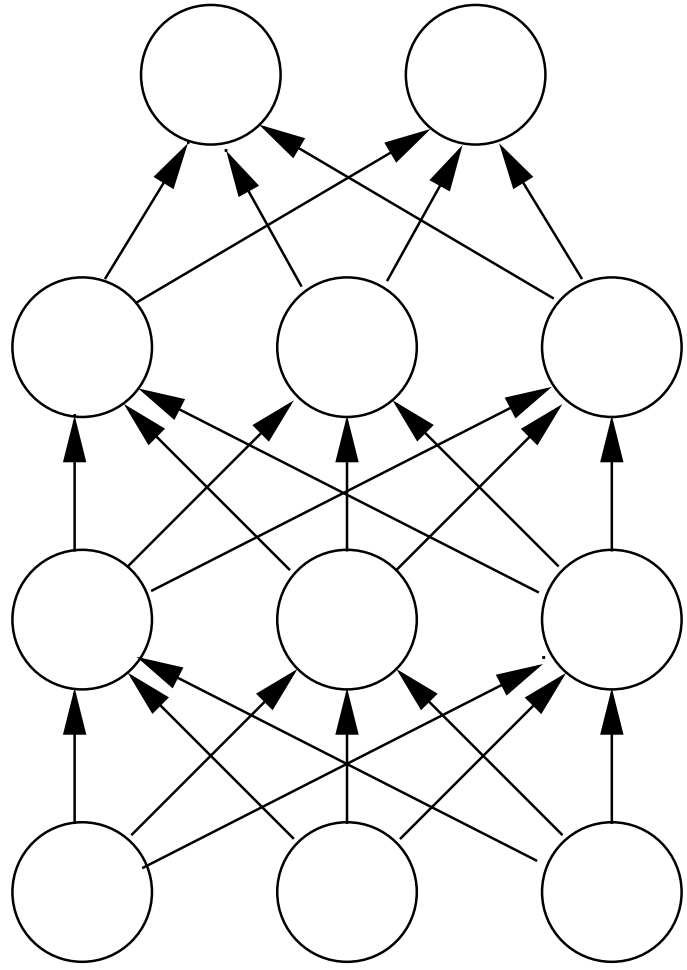
Main techniques to contrast overfitting:

- Collect more data
- Constrain the model expressiveness
- Early stopping
- Regularization, e.g. Weight-decay
- Model averaging
- Data augmentation
- **Dropout**

Dropout

Idea: "cripple" the neural network stocastically removing hidden units

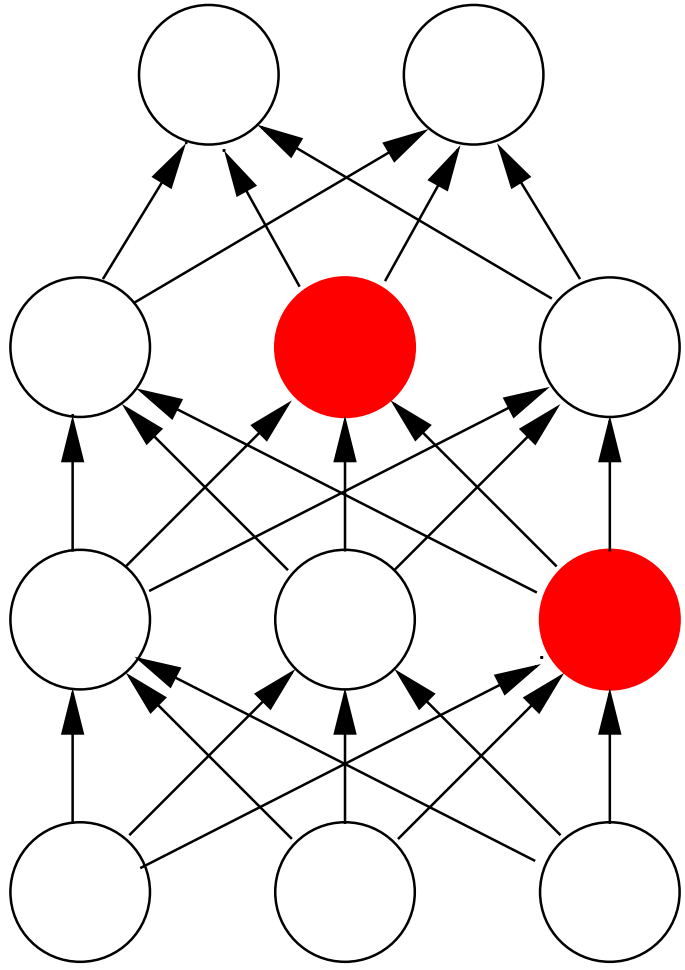
- during training, at each iteration hidden units are disabled with probability p (e.g. 0.2)
- hidden units cannot co-adapts with other units
- similar to train many networks and averaging between them



Dropout

Idea: "cripple" the neural network stocastically removing hidden units

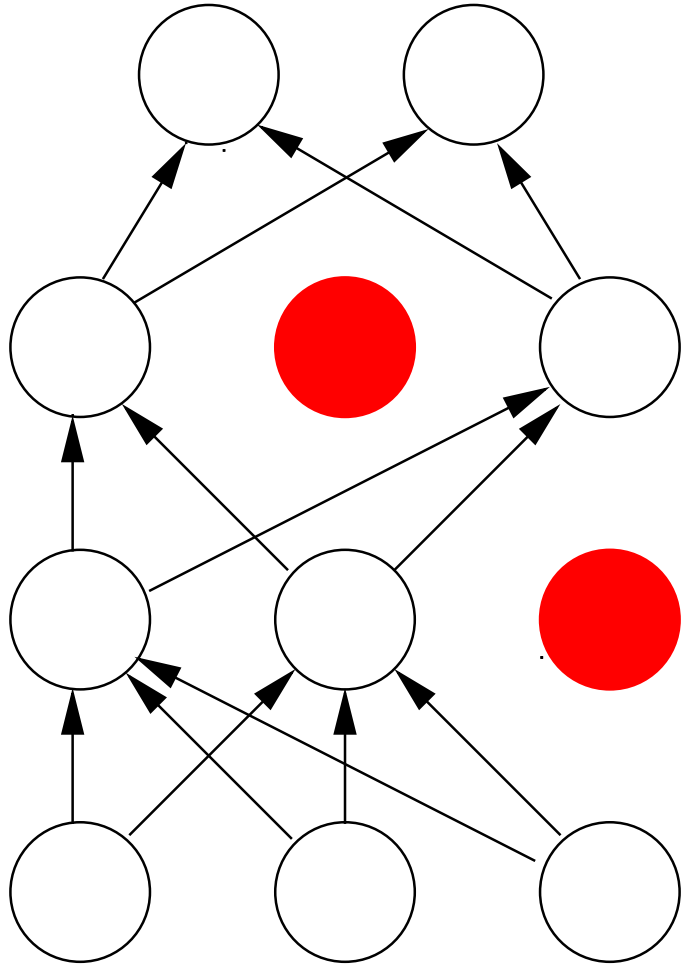
- during training, at each iteration hidden units are disabled with probability p (e.g. 0.5)
- hidden units cannot co-adapts with other units
- similar to train many networks and averaging between them



Dropout

Idea: "cripple" the neural network stocastically removing hidden units

- during training, at each iteration hidden units are disabled with probability 1-p (e.g. 0.5)
- hidden units cannot co-adapts with other units
- similar to train many networks and averaging between them



Geometric averaging

At each stage of training, only the crippled network is trained by means of backpropagation. Then, the omitted units are reinserted and the process repeated (hence weights are shared among the crippled networks).

To compensate the disable unit, the weight of the other unist is multiplied by $1/p$.

We are essentially averaging among a huge number ofdifferent crippled networks.

Sharing weights means that every model is strongly regularized by all the other models.

Normalization layers

Expected benefit

- have a more stable and possibly faster training
- increase the independence between layers

Main problem: normalization operates on the whole training set, while during training, we only have access to a batch of data.

Batch Normalization

Batch normalization operates on single layers, per channel base.

At each training iteration, the input is normalized according to **batch (moving) statistics**, subtracting the mean μ^B and dividing by the standard deviation σ^B.

Then, an opposite transformation is applied based on **learned** parameters γ (scale) and *beta* (center).

$$BN(x) = \gamma \cdot \frac{x - \mu^B}{\sigma^B} + \beta$$

Batch statistics are **moving** in the sense that we keep running values and slowly update them based on the current minibatch.

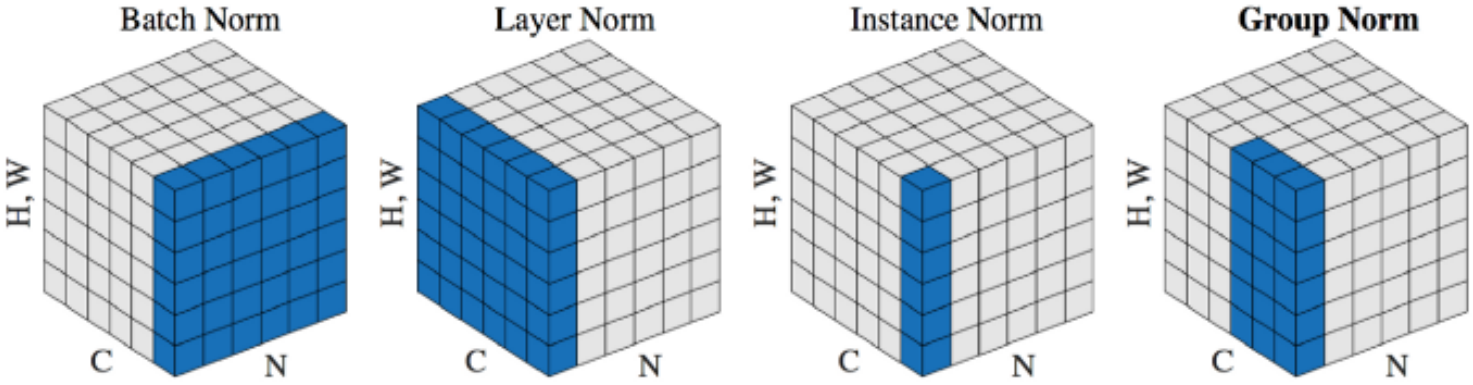
Batch Normalization at prediction time

Batch normalization behaves differently in training mode and prediction mode.

Typically, after training, we use the entire dataset to compute stable estimates of the variable statistics and use them at prediction time.

Once training is concluded, statistics (over a given training set) do not change any more.

Other forms of normalization



Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes.

The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

See Group Normalization for a discussion.

Attention

Attention

Attention

Attention is the ability to focus on different parts of the input, according to the requirements of the problem being solved.

It is an **essential** component of any intelligent behaviour, with potential application to a wide range of domains.

A differential mechanism

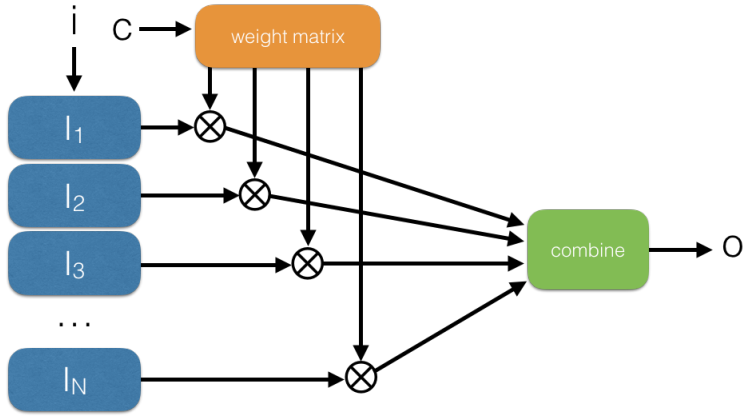
From the point of view of Neural Networks, we would expect the attention mechanism to be **differentiable**, so that we can learn where to focus by standard **backpropagation techniques**.

The current approach (not necessarily the best one) is to **focus everywhere**, just to **different extents**.

Attention as gating maps

Attention is typically implemented by means of gating maps, associating different weights to different inputs.

The gating maps are dynamically generated by some neural net, allowing to focus on different part on the input at (e.g.) different times.



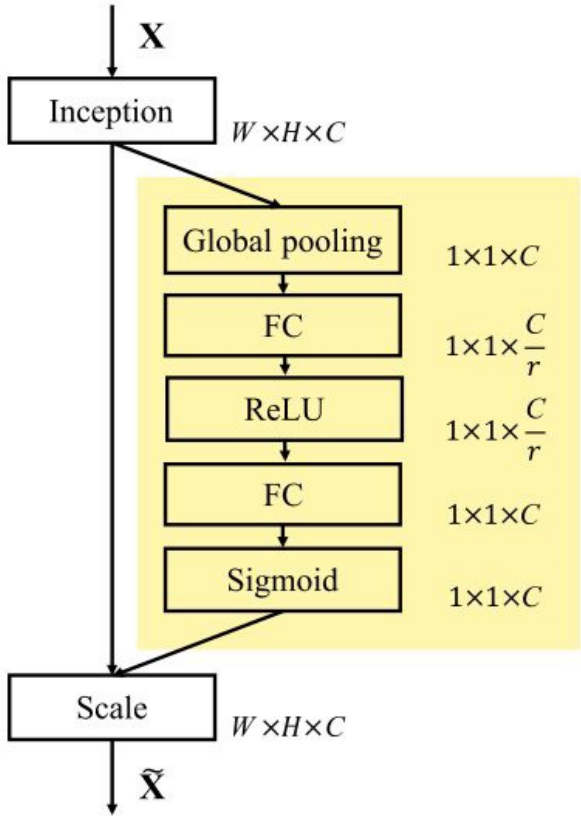
Picture from The fall of RNN / LSTM by E.Colurciello

The forget map, input map and output map in LSTMs are examples of attention mechanisms.

An example: squeeze and excitation

SE layers are a building component of Squeeze and Excitation Networks

SE layers implement a form of self attention, allowing to focus on particular channels in a dinamical way, according to the input under consideration.



SE-Inception Module

Key-Value activation

The most typical attention layer is based on the **key-value** paradigm, implementing a sort of associative memory, or dictionary.

We access this memory with **queries** to be matched with keys.

The resulting **scores** generate a boolean map that is used to weight values.

Attention: the key-value approach

For each key k_i compute the **scores** a_i as

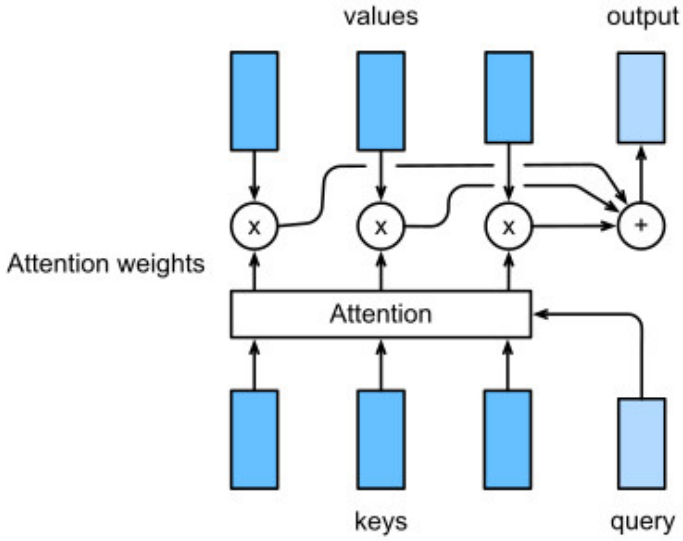
$$a_i = \alpha(q, k_i)$$

obtain **attention weights** (the gating map) via softmax:

$$\vec{b} = softmax(\vec{a})$$

retrun a weighted sum of the values:

$$o = \sum_{i=1}^{n} b_i v_i$$



In many applications, values are also used as keys: **self-attention**.

Typical score functions

Different score functions lead to different attention layers.

Two commonly used approaches are:

- Dot product.

$$\alpha(q, k) = q \cdot k / \sqrt{(d)}$$

  The query and the key must have the same dimension d

- MLP: α is computed by a neural network (usually composed by a single layer):

$$\alpha(k, q) = tanh(W_k \vec{k} + W_q \vec{q})$$

See Attention layer for the Keras implementation of this layer.

An application to translation

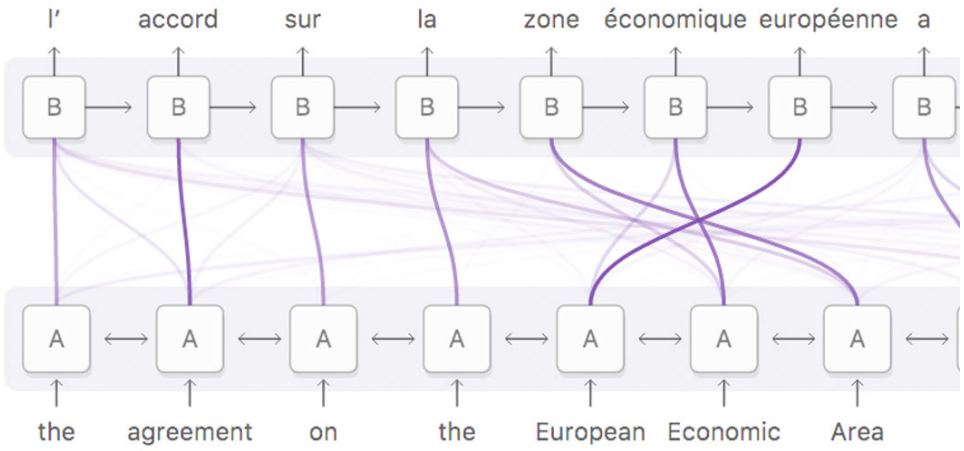
Neural Machine Translation by jointly learning to align and to translate, D.Bahdanau, K.Cho, Y.Bengio (2015)

alignment identify which parts of the input sequence are relevant to each word in the output

translation is the process of using the relevant information to select the appropriate output.

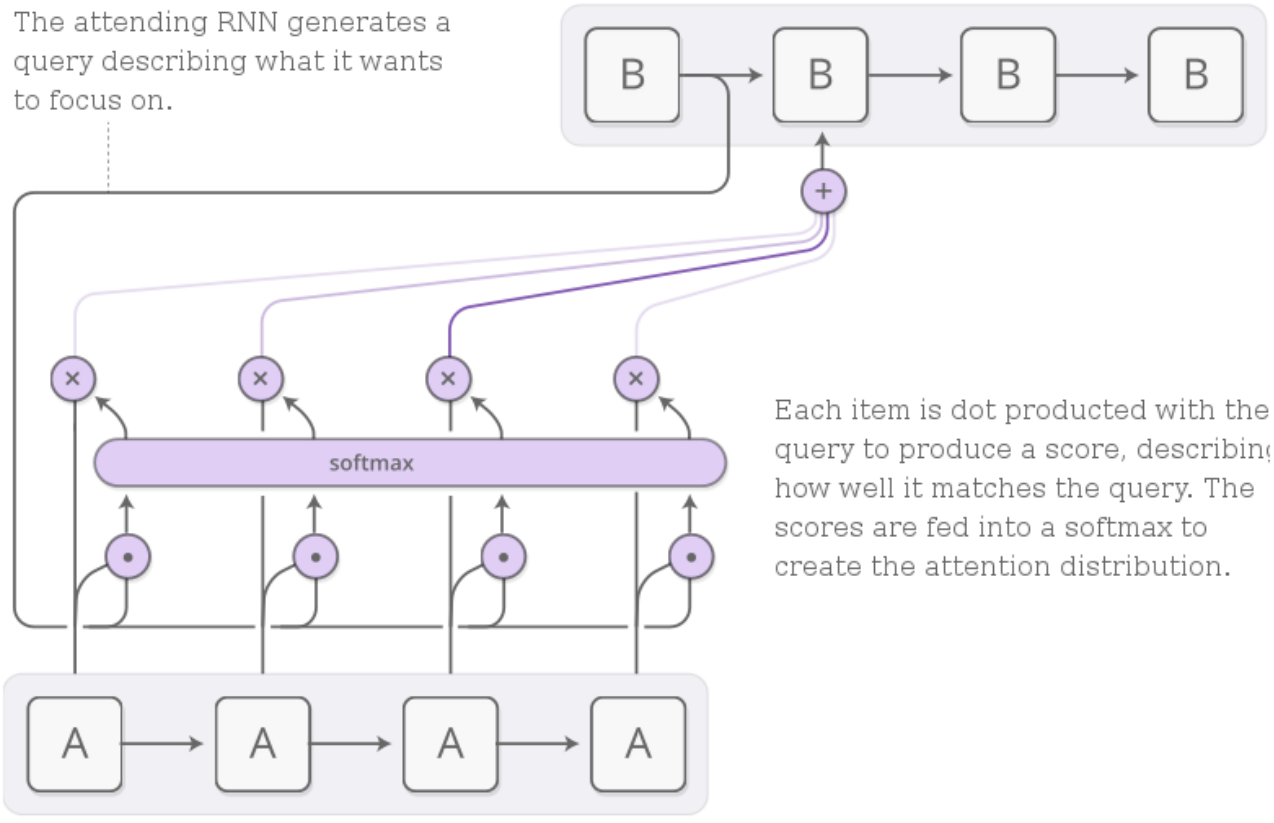
Attention to alignement

Alignement is a form of attention!



Picture from Attention and Augmented Recurrent Neural Networks, by C.Olah and S.Carter.

Producing attention maps



Picture from Attention and Augmented Recurrent Neural Networks.

Producing attention maps

"The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed- length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly".

Attention and Augmented Recurrent Neural Networks.

The attention layer

In Keras,

- Inputs are a list with 2 or 3 elements:
  1. A **query** tensor of shape (batch_size, Tq, dim_q).
  2. A **value** tensor of shape (batch_size, Tv, dim_v).
  3. A optional **key** tensor of shape (batch_size, Tv, dim_q). The key must have the same dimension of the query (for dot product attention). If the key tensor is not supplied, value will be used as a key (in that case, dim_q=dim_v)
- Output. A tensor of shape (batch_size, Tq, dim_v).