Deep Learning

Learning High-degree Non-linear Models: Enabling factors, tips and tricks

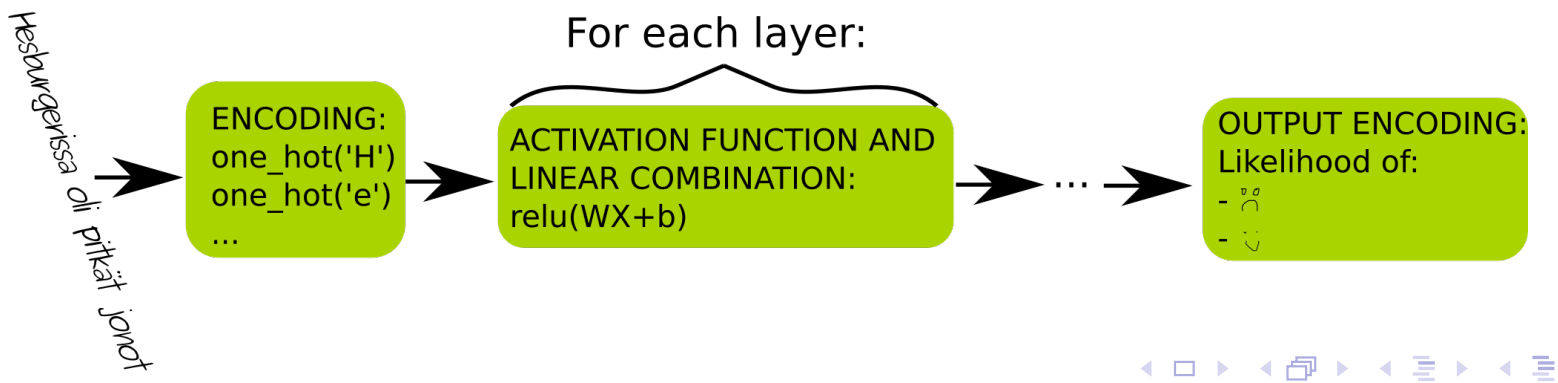
Tero Keski-Valkama

Cybercom

2016-08-24

A quick recap

- Deep learning refers to deep neural networks. It took about 10 years for neural network technology to surpass the issues they stalled with in the 1990s. In 00s, if an academic paper mentioned neural networks, it was less likely to be published than if not.
- 1990s networks were shallow, and hence relatively useless: It requires a bunch of non-trivial tricks to make deeper networks possible. This presentation will present the most important of these tricks.
- Neural networks are just complex non-linear models with lots of parameters. They are formed by layers of neurons, successive operations with an a linear combination of inputs from the previous layer and an activation function.
- In general, you always need a training set, a test set and a validation set. Typically you would divide your data into three parts, train the system with the training set, test if the system overfits with the test set, and finally for the final system you can validate that your metaparameters didn't just learn the test set using the validation set.
- Underfitting = high bias, overfitting = high variance



Training Neural Networks 1/2

- Networks can be trained in a supervised fashion, with target outputs(labels), using backpropagation of error. The output error is known, and it is propagated backwards, layer by layer, estimating an error for each weight parameter. The weights are then updated using this error multiplied by the learning factor ($<< 1$). Lots of learning steps makes the network learn the target function ($input \rightarrow target\_output$)
- Non-linearities (activation functions) between the layers make the model interesting, compared to other statistical models.
- When the layers are getting smaller than the previous layer, the information is being compressed (or filtered), and the layer activations represent a higher abstraction level, a higher semantic level information about the signal.

Training Neural Networks 2/2

- Unsupervised learning (no target output) can be done using backpropagation and autoassociativity ($input \rightarrow input$), (or prediction: $delayed\_input \rightarrow input$), or using Deep Belief Nets and Contrastive Divergence [1] (minimizes the energy for generating real data-related distributions while maximizing the energy for "confabulations") for pairs of subsequent layers one by one.
- Metaoptimizing the learning hyperparameters and neural network structure should always be performed.
- A good platform to use for neural network experimentation is Google's TensorFlow, based on Python.

Supervised	Unsupervised
Backpropagation	Contrastive Divergence
Non-energy-based Methods	Energy-based Methods

CYBERCOM GROUP

[1] http://deeplearning.cs.cmu.edu/notes/yuxiong_CD.pdf

Tips & Tricks

Problems

- Overlearning
- Getting stuck in local minima
- Exploding/diminishing gradients

Solutions

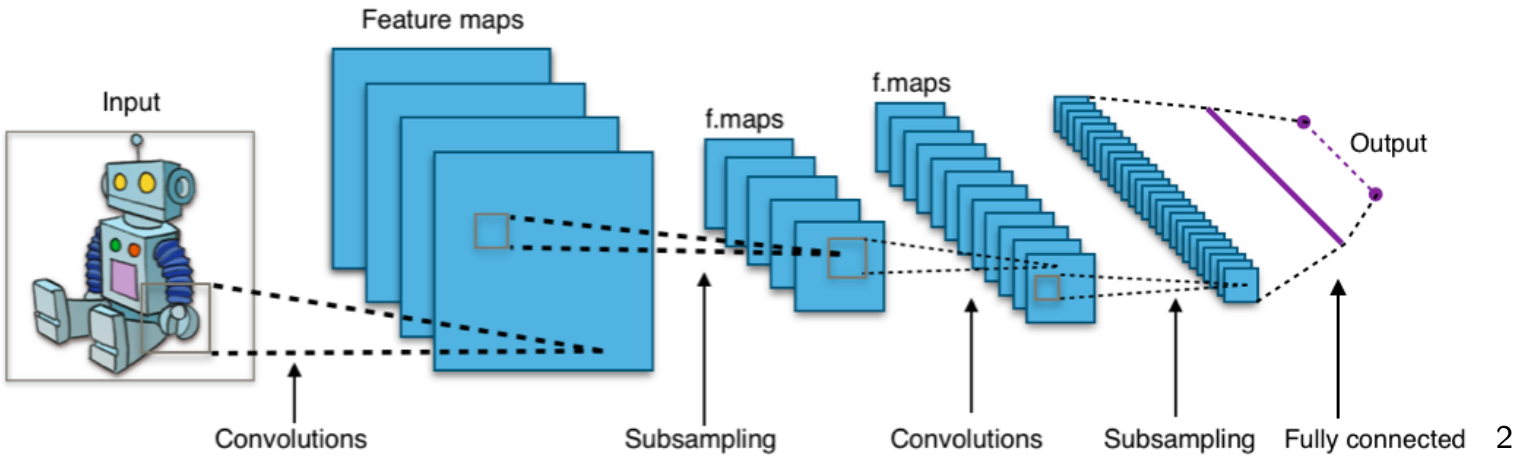
1. Weight-sharing – less parameters
2. Unsupervised pre-training – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping
6. Metaoptimization
7. Bonus: Reservoir computing

CYBERCOM GROUP

Regularization and Weight Sharing

- Reducing the number of parameters or degrees of freedom of the model is regularization. Regularization prevents overfitting.
- In Convolutional Neural Networks, the weights are identical for each window, and window outputs are max-pooled (sub-sampling) to the next layer. This exploits the translational symmetry of the domain, and is often used for images.
- Regularization can also be done by introducing a loss term for weights, L2 norm is often used.
- As always in multiobjective optimization, the factors and slopes of the loss function components are important.

$$RegularizedLoss = Loss + \alpha \cdot SquaredWeights^{\beta}$$



[2]Image from: Aphex34, Wikipedia