Deep Learning

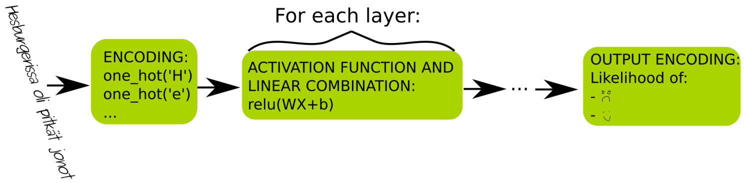
Data encoding / representation

Tero Keski-Valkama

2016-09-22

A quick recap

- Deep learning refers to deep neural networks. 1990s networks were shallow, and hence relatively useless.
- 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. Training set is used to tune parameters, test set is used to tune hyperparameters, and validation set is used to check that the model does not overfit the test set. Boottrapping can be used to validate the data sectioning.
- Underfitting = high bias, overfitting = high variance



Training Neural Networks

- In supervised training, the neural networks requires input and target output. The system finds a layered non-linear function from input to output.
- In unsupervised training, the network is only given input, and it learns a structure that captures the input statististical distributions and correlations.
- Unsupervised training uses energy-based methods which are better able to capture deep associations than a simple backpropagation, hence it is used for pre-training.
- A good platform to use for neural network experimentation is Google's TensorFlow, based on Python.
- Data representation is critical in neural networks, in the input representation, in the output representation (and by extension in defining the loss function), and also in the internal neuron representation.

Tips & Tricks

Data Representation

- Input representation
  - Boolean values, Continuous values
  - One-hot encoding for encoding exclusive choice
  - Encoding time
- Output representation
  - Boolean values, Continuous values
  - One-hot encoding for encoding exclusive choice
  - Mixture distributions
- Internal representation
  - Abstraction levels
  - Bottlenecks and compression

Input Representation

-

Output Representation / Categorical and Class Outputs

- Categorical outputs refer to a situation where the output can take one out of a limited set of values. These are used for example in classification tasks where the input signal is classified to represent one of predetermined classes. For example “cat” vs. “dog”, or “walking” vs. “driving”.
- For categorical outputs, one-hot softmax encoding is typically used. In generation, the outputs represent probabilities for picking a specific category.
- Softmax normalizes the sum of the output values to be one, and scales all values to be positive, i.e. forms a proper probability distribution.
- The loss is calculated using cross entropy. In Tensorflow, you skip the softmax layer in the loss function and use the unnormalized outputs (interpreted as log-probabilities) directly with the tf.nn.softmax_cross_entropy_with_logits function.
- Sometimes the outputs are not mutually exclusive. For example, a photo might contain both a cat and a person. In these cases you normalize each output between 0 and 1 using a sigmoid function. In Tensorflow, you would use tf.nn.sigmoid_cross_entropy_with_logits function to calculate the loss from raw, unscaled outputs.

Output Representation / Mixture Density Outputs

- For continuous values, it makes sense to use mixture density distributions, that is, using a weighted mixture of several distributions (e.g. 3 x normal distributions) with parameters (for normal distribution: means and variances) estimated by the network.
- The relative weights are softmaxed to represent a discrete probability distribution, much like categorical outputs above.
- The distribution parameters are normalized to an appropriate range if necessary, for example variance of the normal distribution is typically transformed through exp() function to limit it to be always non-negative. This also has a rationale from statistical likelihoods.
- Generating values from mixture density distributions is done by first picking the distribution index from the set of distributions using the relative weight probabilities.
- Then a value is picked from the selected distribution using sampling from that kind of distribution with those parameters.
- Mixture density distributions work especially well for continuous physical signals where the system has several different "decisions" to make from moment to moment. This corresponds to degrees of freedom in some sense.
- It is not always clear how many distributions you should use. Analogous to K-means clustering.

Output Representation / Audio

- In audio signals, it has been noted that instead of using mixture density distributions it is better to use one-hot softmax classes for different signal values at a certain time.
- This corresponds to a discrete distribution, and it works better because it is not clear for audio signals how many mixtures one should use.
- To keep the output tractable, the signal levels are squashed from full 16 bit to for example 255 different values using μ law encoding. [1]

[1] https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Internal Representation

-

What did we learn?

- To make neural networks learn effectively, the data must be represented in a suitable fashion.