Deep Learning

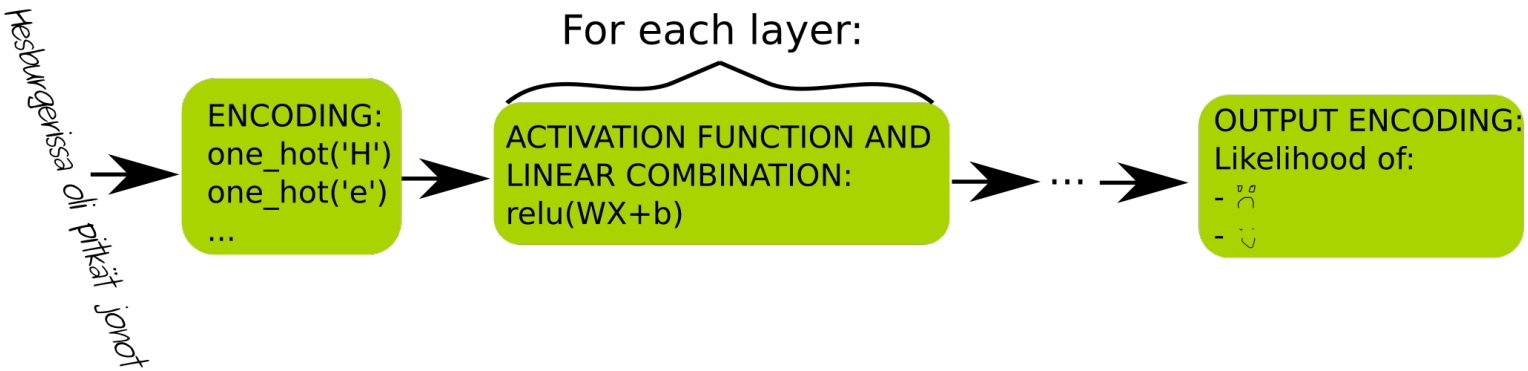
Data encoding / representation

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 a non-linear 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. Bootstrapping 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.
- Training a neural network leads to neural layers forming a mapping from input data vector space through feature filters to higher level non-linear feature spaces and ultimately to semantic spaces.

Data Representation

- Input representation
  - Normalization
  - Encoding time
  - Local conditioning
- Output representation
  - Boolean values, Continuous values
  - One-hot encoding for encoding exclusive choice
  - Mixture distributions
- Internal representation
  - Abstraction levels
  - Bottlenecks and compression
  - Bonus: Residual and skip connections

Input Representation / Normalization

- If a neural network represents a function $f(\mathbf{x}) = \mathbf{y}$, then input is a multidimensional vector $\mathbf{x}$.
- Deciding a representation for your input data is important:
- 0/1 flag-like attributes are typically encoded simply as number components on the vector.
- Categories, such as words or letters are encoded as one-hot vectors where all the other components are marked as 0, but the component for the category as 1. Words as one-hot vectors tend to create really large vectors, unknown words cannot be represented, and typographically similar words are not close to each other.
- Continuous values are discretized into sequences. If a value is a continuous value between low and high limits, then a simple normalization between $(0, 1)$ is often in order.
- If a value does not have limits, tanh or sigmoid normalization might be used to clamp the value between $(0, 1)$ or $(-1, 1)$.
- It is important to normalize the inputs so that they all have similar ranges. Otherwise the training concentrates on the large values only, disregarding the small ones. Also, the meaning of the signal should be somehow evident. If the signal is meaningfully continuous and the meaning comes from passing thresholds, then a continuous encoding is in order. But for example for waveforms and sounds it makes sense to use one-hot encoding.

- Fixed-rate sequences are easy for encoding. However, there are lots of data sources that are not fixed-rate.
- Delays and intervals are often important. Encoding delay as a single number works only if it is normalized between 0 and 1. However, this means that differences between short delays are much more significant than differences between long delays.
- Consider using "tick" events in variable-rate sequences. An LSTM network can more easily count ticks than associate to ad-hoc normalized delay components. (citation needed, "trust but verify")
- I have used such tick events in encoding Flexible Assembly System logs, and they work better than encoding delay as a scalar component.

- How to create networks that associate together low rate text with high rate audio signal?
- These are needed for voice synthesis and speech recognition systems.
- To condition the neural network with another signal requires the signals to be of the same rate. Consider for example lyrics and sound.
- Transposed convolution is often used to learn an upsampling for the lower rate signal. "Deconv" networks can learn even non-linear upsamplings. [1]
- Transposed convolution or fractional stride convolution is a kind of convolution that outputs larger signals than they are fed.
- These networks were originally called deconvolutional neural networks, but that is a misnomer, as deconvolution means an inverse operation to convolution, and this is not what transposed convolutions are.
- As the upsampling is learned, the network associates the lower rate signal to the higher rate signal matching the rates together. [2]
- A soft, learned window can also be used to match the input window progress for separate sequences. [3]

[1] A guide to convolution arithmetic for deep learning

[2] Conditional Image Generation with PixelCNN Decoders

[3] Generating Sequences With Recurrent Neural Networks

- 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.

- 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. [4]

[4] WaveNet: A Generative Model for Raw Audio

- Interpolating representations in feature and semantic spaces generate cognitive morphings.
- Interpolation in explicit feature space and transposed convolution: Learning to Generate Chairs with Convolutional Neural Networks
- Interpolation in implicit feature space: Conditional Image Generation with PixelCNN Decoders
- Internal representation for text allows arithmetic in the meaning space.
- Word2Vec: Learning 100-1000 dimensional embeddings for words based on their context with a 2-layer neural network predicting the current word based on context. For internal representation, interesting applications follow: Brother - Man + Woman = Sister.
- Understanding internal encoding and representation is crucial for interrogating neural networks. See also: MemNN [5]

[5] Yann Le Cunn: What's Wrong With Deep Learning?

Bonus: Residual and Skip Connections

- Training deep neural networks requires all sorts of tricks, some of which were handled in the previous lecture.
- Current state of the art convolutional networks are in fact trained with backpropagation only, but require additional tricks.
- In deep learning, naive backpropagation struggles in pushing back the loss through the layers, and rather uses the biases to make bets, driving the weights to zero. It causes the information to stop flowing from input to output.
- Residual connections are connections that skip one layer in an additive fashion. The input of a unit is simply added to the output also. Sometimes a rectifier activation is added after the summation. This propagates the loss gradients backwards better during training, and information forwards better during generation. [6]

[6] Deep Residual Learning for Image Recognition

What did we learn?

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