Deep Learning

AlejandroMllo

This document serves as a very brief summary of the topics covered in each chapter of the book Deep Learning [1].

Disclaimer: This document is completely extracted from [1], the author does not attribute any ownership over the material.

9 Convolutional Networks

- CNNs are a specialized kind of NN for processing data that has a known grid-like topology. They are simply NN that use convolution (a specialized kind of liner operation) in place of general matrix multiplication in at least one of their layers.
- In its more general form, convolution is an operation on two functions of a real valued argument. The first argument (the first function) to the convolution is often referred to as the input, and the second argument (the second function) as the kernel. The output is sometimes referred to as the feature map. A convolution over functions x and w is denoted with an asterisk: $s(t) = (x * w)(t)$.
- In ML applications, the input is usually a multidimensional array of data, and the kernel is usually a multidimensional array of parameters that are adapted by the learning algorithm. These multidimensional arrays are known as tensors.
- Convolutions can be used over more than one axis at a time. Also they are commutative. Ex: if using a two-dimensional image I as input, then is probable that a two-dimensional kernel K is wanted: $S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$.
- Many NN libraries implement a related function call the cross-correlation, which is the same as convolution but without flipping the kernel: $S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$.
- Convolution enables sparse interactions, parameter sharing and equivariant representations. They also provide a means for working with inputs of variable size.
- Receptive field: the output units in a layer that are input to a unit in other layer.
- Sparse interactions/connectivity/weights: every output unit does not interact with every input unit.
- Parameter sharing: using the same parameter for more than one function in a model. It means that the network has tied weights.
- Equivariance: a function is equivariant if, in the case the input changes, the output changes in the same way. The function f is equivariant to g if: $f(g(x)) = g(f(x))$.
- Stages of a convolutional network's layers:
  1. The layer performs several convolutions in parallel to produce a set of linear activations.
  2. (Detector stage) Each linear activation is run through a nonlinear activation function.
  3. A pooling function is used to modify the output of the layer further.
- A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. It helps to make the representation approximately invariant to small translations of the input. Its use can be viewed as adding an infinitely strong prior that the function the layer learns must be invariant to small translations.
- Prior probability distribution: PD over the parameters of a model that encodes the beliefs about what models are reasonable, before any data is seen. A weak prior is a prior distribution with high entropy. A strong prior has very low entropy. An infinitely strong prior places zero probability on some parameters and says that these parameter values are completely forbidden, regardless of how much support the data give to those values.
- It is possible to think of the use of convolution as introducing an infinitely strong prior PD over the parameters of a layer.
- Convolution and pooling can cause underfitting.

- Convolution in the context of NN actually means an operation that consists of many applications of convolution in parallel. The objective is that each layer of the network extracts many kinds of features, at many locations.
- Because convolutional networks usually use multichannel convolution, the linear operations they are based on are commutative only if each operation has the same number of output channels as input channels.
- Assume we have a 4-D kernel tensor $\mathbf{K}$ with element $K_{i,j,k,l}$ giving the connection strength between a unit in channel i of the output and a unit in channel j of the input, with an offset of k rows and l columns between the output unit and the input unit. Assume our input consists of observed data $\mathbf{V}$ with element $V_{i,j,k}$ giving the value of the input unit within channel i at row j and column k. Assume our output consists of $\mathbf{Z}$ with the same format as $\mathbf{V}$. If $\mathbf{Z}$ is produced by convolving $\mathbf{K}$ across $\mathbf{V}$ without flipping $\mathbf{K}$, then
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n},$$
where the summation over l, m and n is over all values for which the tensor indexing operations inside the summation are valid.
- If we want to sample only every s (stride) pixels in each direction in the output, the we can define a downsampled convolution function c such that
$$Z_{i,j,k} = c(\boldsymbol{K}, \boldsymbol{V}, s)_{i,j,k} = \sum_{l,m,n} [V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}].$$
- One essential feature of any convolutional network implementation is the ability to implicitly zero pad the input $\mathbf{V}$ to make it wider. Without this feature, the width of the representation shrinks by one pixel less than the kernel width at each layer.
- Unshared convolution: in some cases convolutions are not used, and instead locally connected layers are used. In this case, the adjacency matrix in the graph of the MLP is the same, but every connection has its own weight, specified by a 6-D tensor $\mathbf{W}$. The indices into $\mathbf{W}$ are respectively: i, the output channel; j, the output row; k, the output column; l, the input channel; m, the row offset within the input; and n, the column offset withing the input. The linear part of a locally connected layer is then given by
$$Z_{i,j,k} = \sum_{l,m,n} [V_{l,j+m-1,k+m-1} w_{i,j,k,l,m,n}]$$
This is useful when is known that each feature should be a function of a small part of space, but there is no reason to think that the same feature should occur across all of space.
- Tiled convolution: rather than learning a separate set of weights at every spatial location, we learn a set of kernels that we rotate through as we move through space.
- To perform learning in a CNN, it must be possible to compute the gradient w.r.t. the kernel, given the gradient w.r.t. the outputs.
- Convolution, backprop from output to weights, and backprop from output to inputs are the operations sufficient to compute all the gradients needed to train any depth of feedforward convolutional network, as well as to train convolutional networks with reconstruction functions based on the transpose of convolution.
- CNNs can be used to output a high-dimensional structured object. Typically this object is just a tensor.
- The data used with a CNN usually consists of several channels, each channel being the observation of a different quantity at some point in space or time.
- When the inputs are of different size (only because they contain varying amounts of observations of the same kind of things) convolution is straightforward to apply; the kernel is simply applied a different number of times depending on the size of the input, and the output of the convolution operation scales accordingly.
- Convolution is equivalent to converting both the input and the kernel to the frequency domain using a Fourier transform, performing point-wise multiplication of the two signals, and converting back to the time domain using an inverse Fourier transform.
- When a d-dimensional kernel can be expressed as the outer product of d vectors, one vector per dimension, the kernel is called separable. When the kernel is separable, naive convolution is equivalent to compose d one-dimensional convolutions with each of these vectors. The composed approach is significantly faster.
- The most expensive part of CNN training is learning the features. When performing supervised training with gradient descent, every gradient step requires a complete run of forward propagation and backward propagation through the entire network.

- To obtain convolution kernels without supervised learning, simply initialize them randomly, design them by hand or learn them with an unsupervised criterion.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.