Deep Learning Chapter 6. Deep Feedforward Networks.

Deep feedforward Networks : information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.

When it's extended to include feedback connections, it's recurrent neural network

The MSE loss function :

$$J(\theta) = \frac{1}{m} \sum_{x \in X} (f^*(x) - f(x;\theta))^2$$

1. The main architectural considerations are to choose the depth of the network and the width of each layer.

   The ideal network architecture for a task must be found via experimentation guided by monitoring the validation set error.

Chapter 7. Regularization for deep learning.

1. L^2 parameter Regularization.

$$\Omega(\theta) = \frac{1}{2}\|w\|^2.$$

2. Sparse Representations.

3. Bagging (bootstrap aggregating) is a technique for reducing generalization error by combining several models.

4. Dropout: provides a computationally inexpensive but powerful method of regularizing a broad family of methods.

5. Stochastic gradient descent (SGD) = incremental gradient descent.

6. The most popular optimization algorithms actively in use include:
   i). SGD.
   ii). SGD with momentum.
   iii). RMSprop
   iv). RMSProp with momentum.
   v). AdaDelta.
   vi). Adam.

7. Limited Memory BFGS (or L-BFGS): the memory costs of the BFGS algorithm can be significantly decreased by avoiding storing the complete inverse Hessian approximation M.

8. It is more important to choose a model family that is easy to optimize than to use a powerful optimization algorithm.

Chapter 9. Convolutional Neural Network.

1. There are two arguments in CNN, one is input, the other is kernel. The output is referred to as the feature map.

   The input is a multidimensional array of data, and the kernel is usually a dimensional array of parameters.

   For example, use a two-dimensional image I as input, also, use a two-dimensional kernel K, then:

   $$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n)$$

   Convolution is commutative, so:

   $$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) \cdot K(m,n)$$

   Actually, the kernel is much smaller than the input data.

2. i) Tradition neural Network: $\theta^T \cdot X \rightarrow$ each input unit interacts with each output unit.

   Sparse interaction in CNN: by making the kernel smaller then the input.

   ii). Tradition neural network: each element of the weight matrix is used exactly once when computing the output of a layer.

   Parameter sharing in CNN: each member of the kernel is used at every position of the output.

   Sparse connectivity and parameter sharing can dramatically improve the efficiency of a linear function for detecting edges in an image.

   iii). Equivariance: $f(g(x)) = g(f(x))$, then g and f are equivariant.

   If the input changes, the output changes in the same way, if we move an event later in time in the input, the exact same representation of it will appear in the output, just later in time.

   Convolution is not naturally equivariant to other transformations, such as changes in the scale or rotation of an image.

3. Pooling.

A typical layer of a convolutional network consists of three stages.

i). First stage. the layer performs several convolutions in parallel to produce
convolution stage. a set of linear activactions.

ii). Detector stage, each linear activation runs through a nonlinear activation function, i.e.. the rectified linear activation function.

iii) Pooling stage: use a pooling function 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.

In all cases, pooling helps to make the representation become approximately invariant to small translations of the input.

Invariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is.

4. Convolutional Networks can be used to output a high-dimensional, structured object, rather than just predicting a class label for a classification task or a real value for a regression task.

5. The most expensive part of CNN is learning the features.
Three basic strategies for obtaining convolution kernels without supervised training:

i). Simply initialize them randomly.

ii). Design them by hand.

iii). Learn the kernels with an unsupervised criterion: { K-means

CS519. Convolutional Neural Network.

- Filter size, and input/output size:

  filter: $m \times m$, input $n \times n \rightarrow$ output $(n-m+1) \times (n-m+1)$

Each connection is a convolution followed by rectified linear unit (ReLu).
Zero padding the inputs so that the output is $N \times N$.

Pooling:

i) Localized max-pooling helps achieving some location invariance.

ii). Filtering out irrelevant background information.

i.e.: $x_{out} = \max(x_{11}, x_{12}, x_{21}, x_{22})$.

The VGG (Visual Geometry Group) Network.

Why 224×224?

The magic number $224 = 2^5 \times 7$, so that there is always a center-surround pattern in any layer.

Another potential candidate is $384 = 2^7 \times 3$.

However, more layers + higher dimensions $\rightarrow$ more difficult to train, and more machines to tune parameters.

Backpropagation. for CNN.

Le Net: CNN are invented by Yann LeCun, on handwritten digits classification

Strides: another way to reduce image size is by strides, set stride $= n$, then convolution on every n pixels.

Le Net

- Max Pooling: a form of non-linear down sampling, max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value.

  Advantages: i) By eliminating non-maximal values, it reduces computation for upper layers.

  ii). It provides a form of translation invariance, and it's a smart way of reducing the dimensionality of intermediate representations.

Max-pooling is done in Theano by way of theano.tensor.signal.pool.pool_2d.