Deep Learning - Foundations and Concepts

Chapter 6. Deep Neural Networks

nonlineark@github

February 19, 2025

Outline

The curse of dimensionality

In spaces of higher dimensionality, the number of combinations of values must be considered could be huge. This effect is known as combinatorial explosion:

- A polynomial regression of order M for a single input variable needs $M + 1$ parameters. If there are D input variables, the number of parameters needed will be $\binom{M+D}{M}$.
- The histogram based classification for 1-dimensional input needs N buckets. If the input is D-dimensional, the number of buckets needed will be N^D.

For a machine learning model, this usually means that the amount of data needed to generlize accurately grows exponentially.

High-dimensional spaces

High-dimensional spaces can defeat one's geometrical intuitions:

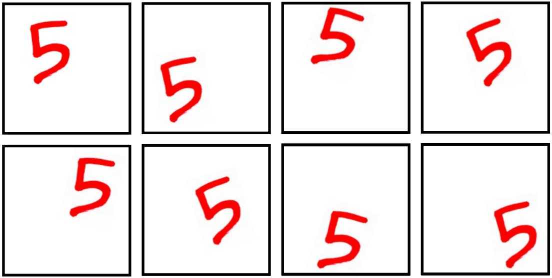
- In spaces of high dimensionality, most of the volume of a hypersphere is concentrated in a thin shell near the surface.
- In spaces of high dimensionality, the probability mass of the Gaussian is concentrated in a thin shell at a specific radius (a soap bubble).

Data manifolds

Although data may be in high-dimensional spaces, real data will generally be confined to a region of the data space having lower effective dimensionality. Effectively, neural networks learn a set of basis functions that are adpated to data manifolds.

Data manifolds

Figure: Images of a handwritten digit that lives on a nonlinear three-dimensional manifold



Data manifolds

Figure: Natural images vs. randomly generated images

Data-dependent basis functions

- Simple basis functions that are chosen independently of the problem being solved can run into significant limitations.
- Using expert knowledge to hand-craft the basis functions was superseded by data-driven approaches in which basis functions are learned from the training data.
- Methods such as radial basis functions and support vector machines have been superseded by deep neural networks, which are much better at exploiting very large data sets efficiently.

Parameter matrices

Consider a basic neural network model having two layers of learnable parameters:

$$a_m^{(1)} = \sum_{d=1}^{D} w_{md}^{(1)} x_d + w_{m0}^{(1)}$$

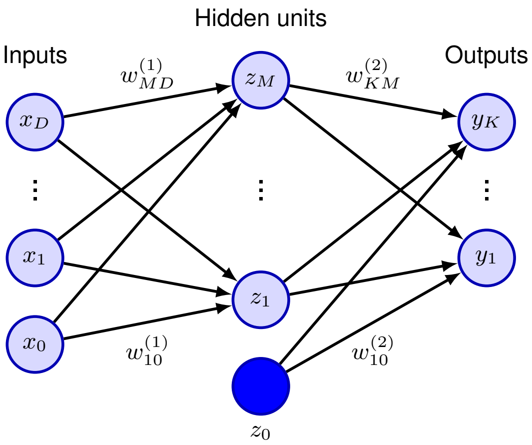
$$z_m^{(1)} = h(a_m^{(1)})$$

$$a_k^{(2)} = \sum_{m=1}^{M} w_{km}^{(2)} z_m^{(1)} + w_{k0}^{(2)}$$

where h is a differentiable, nonlinear activation function.

Parameter matrices

Figure: Network diagram for a two-layer neural network



Parameter matrices

The bias parameters can be absorbed into the set of weight parameters, so the two-layer neural network can be represented as:

$$y_k(x;w) = f(\sum_{m=0}^{M} w_{km}^{(2)} h(\sum_{d=0}^{D} w_{md}^{(1)} x_d))$$
$$y(x;w) = f(W^{(2)} h(W^{(1)} x))$$

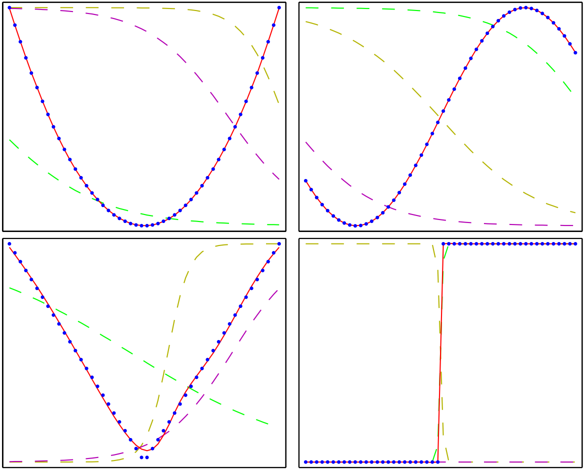
where f and h are activation functions evaluated on each vector element separately.

Universal approximation

- For a wide range of activation functions, two-layer feed-forward networks can approximate any function defined over a continuous subset of $\mathbb{R}^D$ to arbitrary accuracy.
- However, in a practical application, there can be huge benefits in considering networks having many more than two layers that can learn hierarchical internal representations.

Universal approximation

Figure: Two-layer neural networks are universal approximators

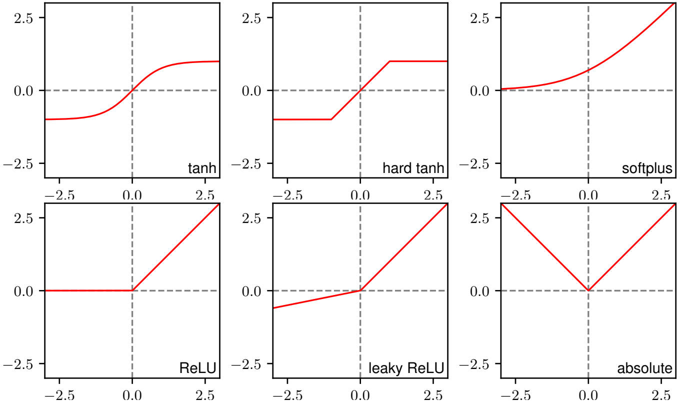


Hidden unit activation functions

- Activation functions for the output units are determined by the kind of distribution being modelled.
- For the hidden units, the only requirement is that they need to be differentiable.
- Obviously, the identity function, sometimes used as the activation function for output units, is not a good option for hidden units.

Hidden unit activation functions

Figure: A variety of nonlinear activation functions



Weight-space symmetries

Consider a two-layer network with M hidden units having $\tanh$ activation functions and full connectivity in both layers:

- Changing the sign of all the weights and the bias feeding into a particular hidden unit can be compensated by changing the sign of all the weights leading out of that hidden unit:
    - 2^M equivalent weight vectors.
- Interchange a particular hidden unit with a different hidden unit:
    - $M!$ equivalent weight vectors.