IE 678 Deep Learning

02 – Feedforward Neural Networks

Part 5: Multi-Layer Perceptrons

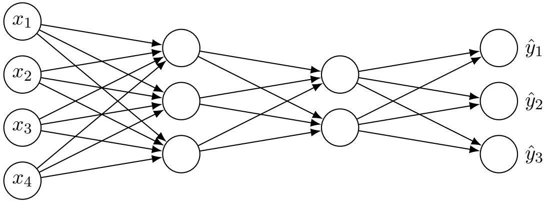
Prof. Dr. Rainer Gemulla

Universität Mannheim

Version: 2024-1

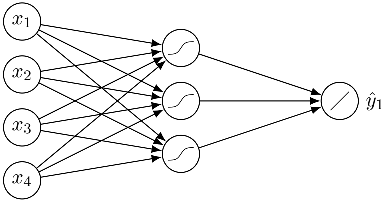
Multi-layer FNNs

- So far: mostly FNNs without hidden layers
  - These networks are limited in what they can do
  - Linear regression, logistic regression, ...
  - We can improve performance by engineering better features
- In neural networks, hidden layers generally necessary
  - Recall: can interpret hidden layers as features for the next layer
  - By including hidden layers, we aim to let the network “do” the feature engineering
  - If there is at least one hidden layer, the network is called a **multi-layer FNN** (also confusingly: **multi-layer perceptron**, MLP)
  - If more than one (or more than some value $L > 1$), called **deep**



How powerful are multi-layer FNNs? (1)

- Example *template architecture* of multi-layer FNNs
  - Hidden layers are sigmoidal (e.g., logistic neuron)
  - Output layer is linear (for regression) or sigmoidal (for classification)
  - Fully connected, number/sizes of hidden layers are hyperparameters
  - More confusion: this template architecture often also called **MLP**
- How powerful are such networks?
- Consider a basic **wide MLP** with
  - D inputs, 1 linear output
  - One fully-connected hidden layer with Z sigmoidal neurons
- **Universal approximation theorem**: This wide MLP can approximate any continuous function on $[0, 1]^D$ given sufficiently (but finitely) many hidden neurons [Cybenko, 1989]



How powerful are multi-layer FNNs? (2)

- Similar universality results exist for **deep MLPs**
  - Any continuous function on $[0,1]^D$ can be approximated arbitrarily well given sufficiently (but finitely) many hidden layers, each with at most $D+1$ units in each hidden layer [Hanin and Selke, 2017]
- Universal approximation means that “any” function can be represented
  - Either via sufficient width or sufficient depth
- But that doesn’t mean that we can learn that function
  - Training methods may fail to find good parameterization
  - Overfitting may occur
  - Number of required units can be exponential in the input dimensionality
  - ...

Wide or deep?

- Deep models tend to show better generalization performance (but architecture matters)
  - Encode belief that function to be learned involves a composition of several simpler functions
  - Interpretation: hidden layer output = intermediate values in a multi-step computation
- But modern architectures are be quite wide, too
  - E.g., recall T5-Base transformer models: maximum hidden layer width of encoder is 3072 $\times$ input size (in number of tokens)

Example

Task: transcribe multi-digit numbers from photographs

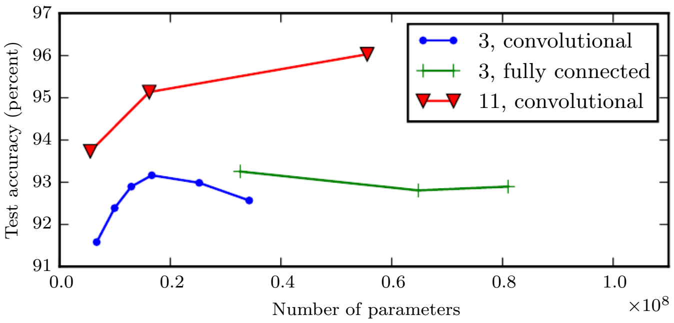
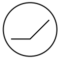
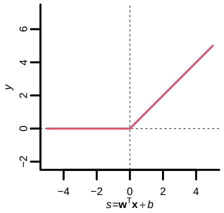


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million.

Goodfellow et al., 2016

Rectified linear neuron (ReLU)

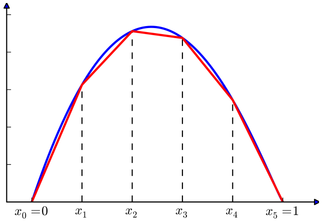
- Also called **linear threshold neuron** or **rectifier**
- $\phi(s) = \max\{0, s\} = \begin{cases} s & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases}$
- Notation: 



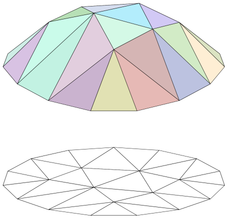
- Note: again, a non-linear transfer function
- **Common non-linearity for intermediate layers in deep NNs**

Rectifier networks

- **Rectifier network** = MLP with only rectifier units in hidden + output layers
- Function computed by rectifier network is **piecewise linear**
  - Approximates a function by decomposing it into **linear regions**
  - Roughly: the more linear regions, the more flexible / expressive



1D example



2D example

Expressivity of rectifier networks

Montúfar et al. (2014) have shown:

- Consider rectifier networks of form
  - D = input dimensionality (assumed constant)
  - H = total number of hidden units
  - L = total number of hidden layers, each of width $Z \geq D$
- Number of linear regions **at most** 2^H
  - $\rightarrow$ No more than exponentially many linear regions possible
- Number of attainable linear regions **at least** $\Omega((Z/D)^{(L-1)D} Z^D)$
  - Attainable = maximum over all possible parameterizations
  - Polynomial in Z (width)
  - Exponential in L (depth)
  - $\rightarrow$ Exponentially many linear regions indeed possible
- Note: result also affects classification
  - E.g., for binary classification, use a binary threshold neuron or logistic neuron as output (rest rectifiers)
  - Observe: Within each linear region, we obtain a linear classifier
  - Each linear region mapped to either one class only (values all positive or all negative) or to two classes separated by a hyperplane