Multilayer Perceptrons

CHEN Si

1 Introduction

- For short, **MLPs**
- Also called **Deep Feedforward Networks** or **Feedforward Neural Networks**

2 Motivation

1. Weaknesses of linear models
    - Linearity is a strong assumption
        - Because outputs are always monotonic with inputs
    - We want to learn a representation of our data (the first $L-1$ layers), take into account the relevant interactions among our features, on top of which a linear model (the final layer) would be suitable
2. How to obtain the features $\phi(\boldsymbol{x})$ of $\boldsymbol{x}$
    - Kernel trick, such as infinite-dimensional RBF Kernel
        - Enough capacity to fit the training set, but poor generalization
    - Manually engineer ϕ
        - Require lots of effort for each separate task
3. From neuroscience
    - The idea of using many layers of vector-valued representations is drawn from neuroscience
4. From mathematical and engineering disciplines
    - Function approximation machines

3 Goal

1. Deal with the weaknesses of linear models
   - With deep neural networks, we used observational data to jointly learn both a representation via hidden layers and a linear predictor that acts upon that representation
   - To capture complex interactions among our inputs via their hidden neurons
   - To use a model that learns a different feature space in which a linear model is able to represent the solution
2. Learn the features $\phi(\boldsymbol{x}; \boldsymbol{\theta})$
   - We have a model $y = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$
3. **Universal Approximators**
   - Designed to achieve statistical generalization (rather than to model our brain)
   - Even with a single-hidden-layer network, given enough nodes (possibly absurdly many), and the right set of weights, we can model any function
   - Approximate many functions much more compactly by using **deeper** (vs. **wider**) networks
   - Learning that function is actually the hard part

4 Model

4.1 Terminology

- Layer: A function in the chain structure
- Hidden layers: The layers for which the training data does not show the desired output

4.2 Assumptions

- For each example, $\boldsymbol{x}$ is accompanied by a label $y \approx f^*(\boldsymbol{x})$

5 Advantages

- Being highly generic: broad family $\phi(\boldsymbol{x}; \boldsymbol{\theta})$
- Generalization: only need to find the right general function family rather than finding precisely the right function

6 Disadvantages

- Give up on the convexity of the training problem