ME 491 Lecture 21

Ch.6 Neural Networks and Deep Learning

Why neural network is so successful?
(1) continued growth of computational power
(2) exceptionally large labeled data sets

General optimization construction of neural networks

$$\underset{A_j}{\operatorname{argmin}}\left(f_M(A_M, \ldots, f_2(A_2, f_1(A_1, x))\ldots) + \lambda g(A_j)\right) \quad (6.1)$$

Solved using stochastic gradient descent and backpropagation algorithm.

A_k: weights connecting the neural network from kth to the $(k+1)$th layer
$g(A_j)$: regularizer

(Massively under-determined system)

Notation in (6.1) is motivated from solving linear systems $Ax = b$ through regression, highlighted in the first few sections.

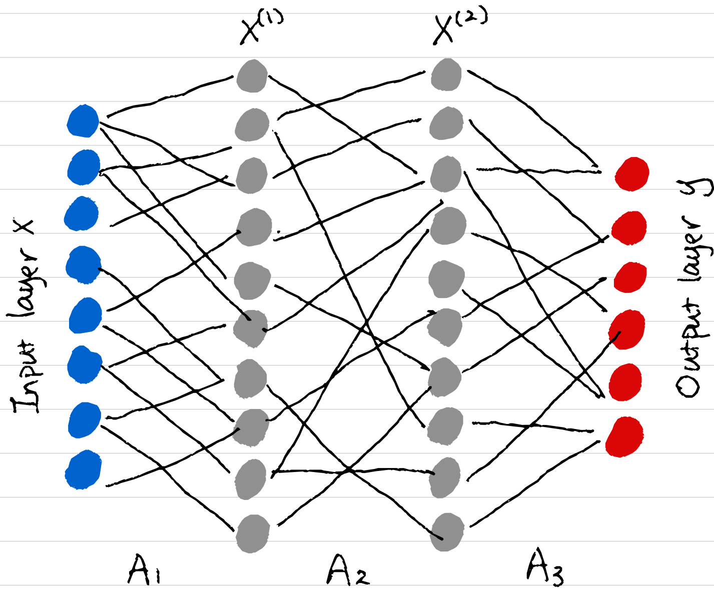
Broader framework: mapping input data X to output data Y using a model $f(\cdot)$

Represent (6.1) in deep learning model:

$$\underset{\theta}{\operatorname{argmin}}\, f_\theta(x)$$

θ: neural network weights
$f(\cdot)$: characterize the network (number of layers, structure, regularizers)

6.1 Neural Networks: Single-Layer Networks



(NN architecture to map an input layer x to output layer y.
Middle (hidden) layers are denoted $x^{(j)}$ where j determines the ordering.
Matrices A_j contain the coefficients that map each variable from one layer to the next.).

For classification tasks, we want to train NN to accurately map the data x_j to the correct label y_j.

* Design questions regarding NNs.
- How many layers?
- Dimension of the layers?
- Output layer design?
- What kind of mapping?

Let's first consider mapping for Fig. 6.1.

$x^{(k)}$: layers between input and output
k : layer number.

For a linear mapping between layers, we have

$$x^{(1)} = A_1 x$$

$$x^{(2)} = A_2 x^{(1)}$$

$$y = A_3 x^{(2)}$$

combine these equations, we have

$$y = A_3 A_2 A_1 x$$

The basic architecture can scale up to M layers:

$$y = A_M A_{M-1} \cdots A_2 A_1 x$$

This is a highly under-determined system that requires some constraints to have a unique solution. One constraint is that the M matrice mappings must be distinct.

We can write a similar construct for nonlinear mappings

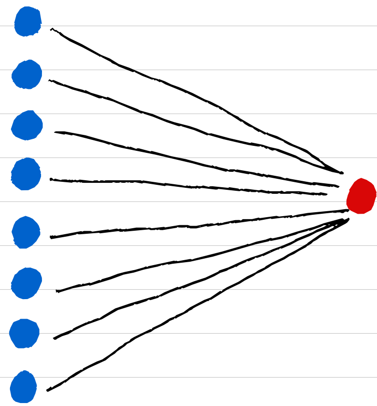
$$y = f_M(A_M, \ldots, f_2(A_2, f_1(A_1, x))\ldots)$$

Note: the nonlinear function $f_j(\cdot)$ can be the same or different for different layers.

* A single-Layer Network

Let's try to build a single-layer network to classify between cats and dogs to understand how NNs work.

Input layer x



Perception $y \in \{\pm 1\}$

+1 dog

-1 cat

A linear mapping between the input image space and output layer can be constructed for training data by solving

$$A = YX^{+}$$

This problem becomes a least-squares regression for the matrix A mapping the images to label space.

To build a single layer NN to classify dog and cat:

* output: $\{\text{dog}, \text{cat}\} = \{+1, -1\}$
* input: images $x_j \in \mathbb{R}^n$

$$AX = Y \Rightarrow [a_1, a_2, \dots, a_n] \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_p \\ | & | & & | \end{bmatrix} = [+1, +1 \dots, -1, -1]$$

We want to determine the mapping A

$$A = YX^{+}$$