Key Techniques Enabling Deep Learning

Ira A. Fulton Schools of Engineering
Arizona State University

Objectives

Objective

Explain how, in principle, learning is achieved in a deep network

Objective

Explain key techniques that enable efficient learning in deep networks

Overview

- Back-propagation algorithm
- Design of activation functions
- Regularization for improving performance

* Technological advancement in computing hardware is certainly another enabling factor but our discussion will focus on basic, algorithmic techniques.

Back Propagation (BP) Algorithm

- Simple Perceptron algorithm illustrates a path to learning by iterative optimization
  - Updating weights based on network errors under current weights, and optimal weights are obtained when errors become 0 (or small enough)

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \bigtriangledown J(\mathbf{W})$$

W is the parameter of the network; J is the objective function.

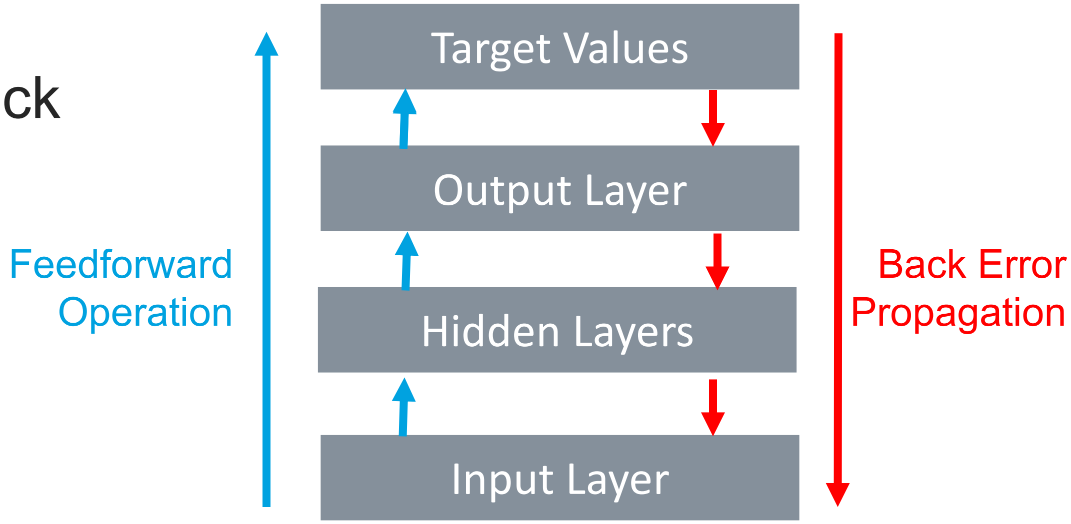
- Gradient descent is a general approach to iterative optimization
  - Define a loss function J
  - Iteratively update the weights W according to the gradient of J with respect to W.

Back Propagation (BP) Algorithm (cont'd)

- Generalizes/Implements the idea for multi-layer networks
  - Gradient descent for updating weights in optimizing a loss function
  - Propagating gradients back through layers
    - hidden layer weights are linked to loss gradient at output layer

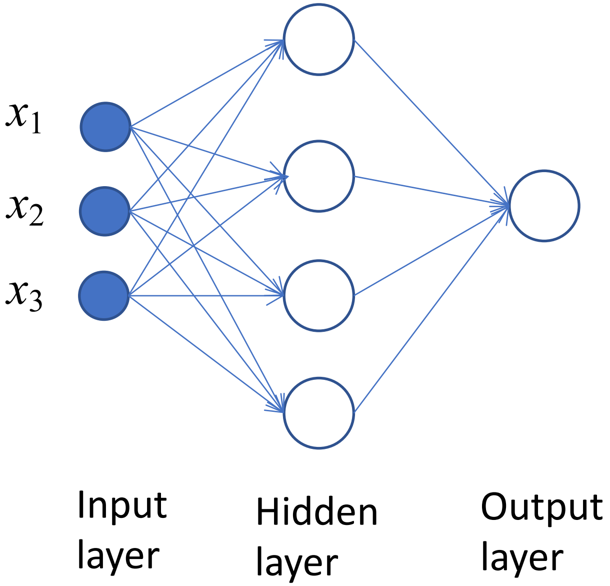
$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla J(\mathbf{W})$$

W is the parameter of the network; J is the objective function.



Illustrating the BP Algorithm 1/6

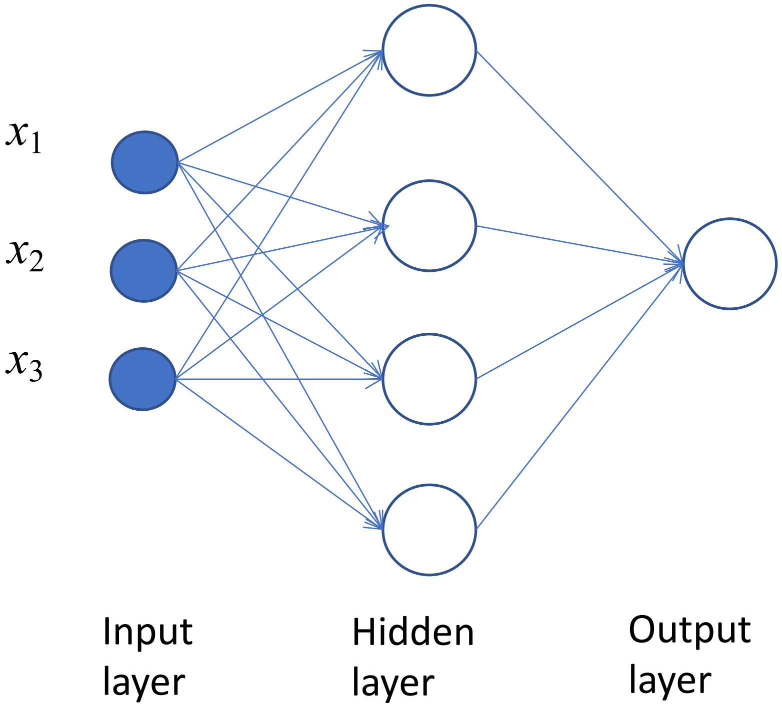
- Let's consider a simple neural network with a single hidden layer. (We will only outline the key steps.)
  - Let's write the net input and activation for a hidden node:



  - Let's write the net input and activation for the hidden layer:

Illustrating the BP Algorithm 2/6

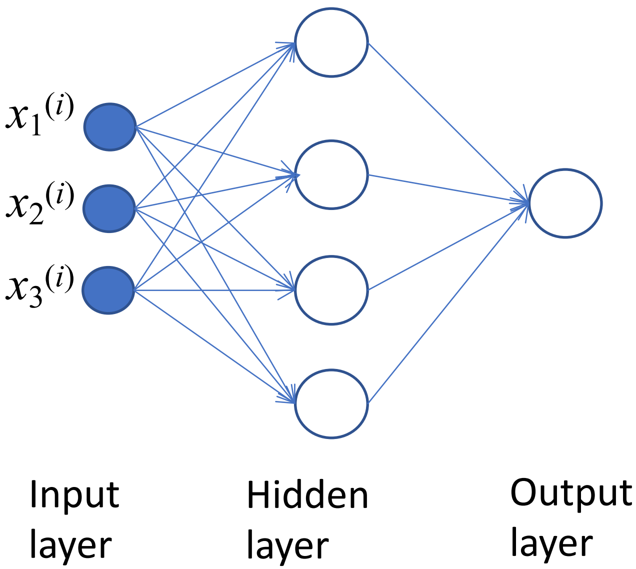
Using matrix/vector notations, for the hidden layer:



Similarly, for the output layer → Homework.

Illustrating the BP Algorithm 3/6

Now consider m samples as input.



Output layer is similarly done.

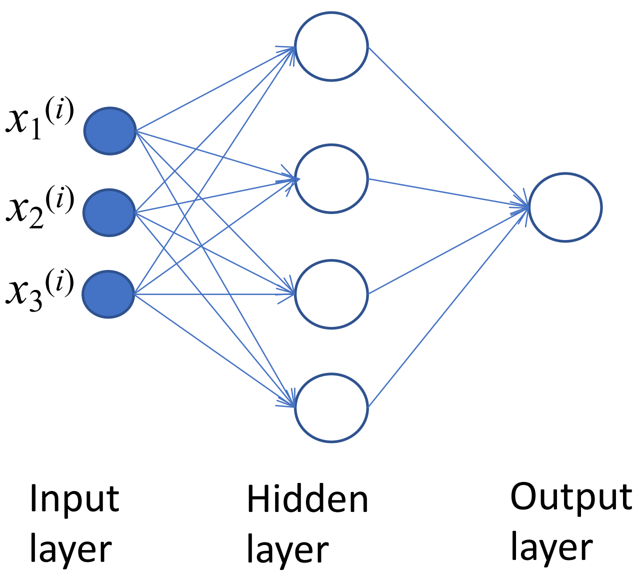
Illustrating the BP Algorithm 4/6

- Overall we have this flow of *feedforward* processing (note the notation change for simplicity: subscripts are for layers):

$$A_0 = X \xrightarrow[3\text{xm}]{} \boxed{Z_1 = W_1 A_0 + b_1} \xrightarrow[4\text{xm}]{Z_1} \boxed{A_1 = f(Z_1)} \xrightarrow[4\text{xm}]{A_1} \boxed{Z_2 = W_2 A_1 + b_2} \xrightarrow[1\text{xm}]{Z_2} \boxed{A_2 = f(Z_2)} \xrightarrow[1\text{xm}]{A_2} \boxed{L(A_2, Y)}$$

W_1: 4x3
b_1: 4x1

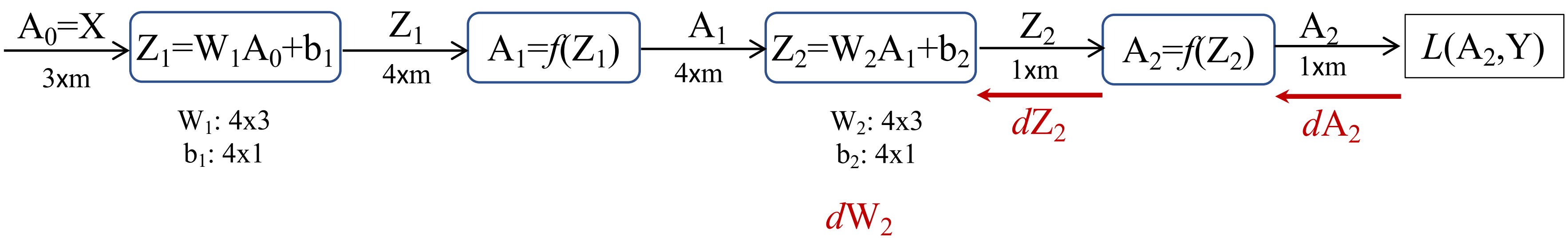
W_2: 4x3
b_2: 4x1



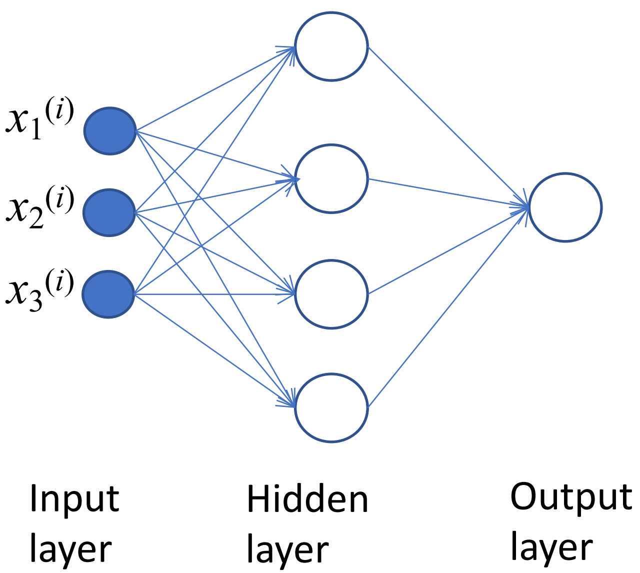
Consider $\quad dW_2 \triangleq \dfrac{\partial L}{\partial W_2}$

Illustrating the BP Algorithm 5/6

Back-propagation

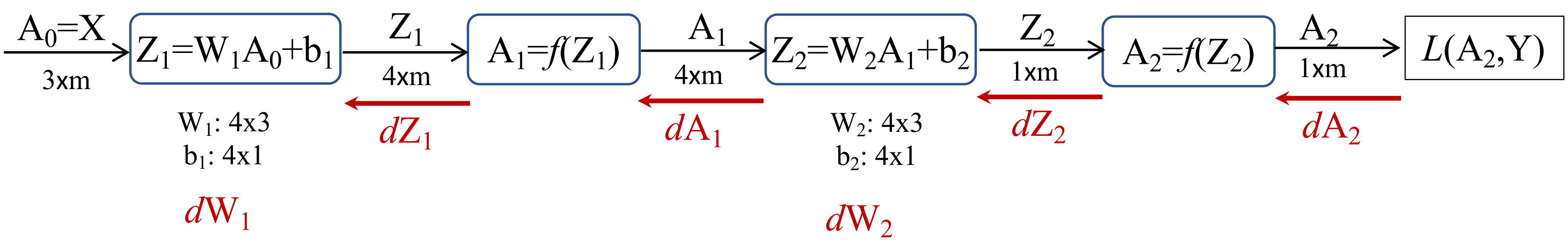


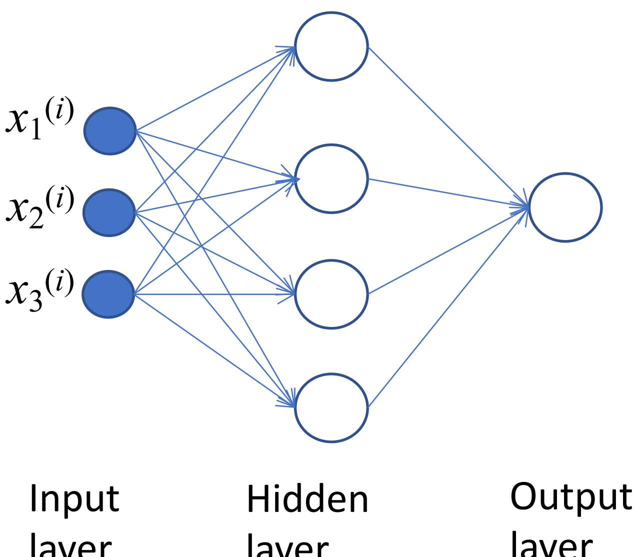
$$\text{Consider} \quad \mathrm{dW}_1 \triangleq \frac{\partial L}{\partial \mathrm{W}_1}$$

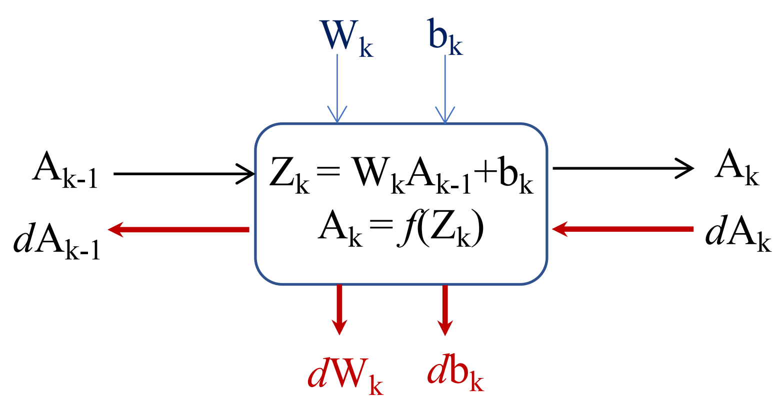


Illustrating the BP Algorithm 6/6

A modular view of the layers







BP Algorithm Recap

- The feedforward process: ultimately produce $A^{[K]}$ that leads to the prediction for Y.

- The backpropagation process:
  - First compute the loss
  - Then compute the gradients via back-propagation through layers
  - Key: use the chain rule of differentiation

- Essential to deep networks

- Suffers from several practical limitations
  - gradient exploding
  - gradient vanishing
  - etc.

- Many techniques were instrumental to enabling learning with BP algorithm for deep neural networks

Activation Functions: Importance

- Provides non-linearity
- Functional unit of input-output mapping
- Its form impacts on gradients in BP algorithm

Activation Functions: Choices

Older Types

- Thresholding
- Logistic function
- tanh

Newer Types

- Rectifier $f(x) = \mathbf{max}(0, x)$ and its variants
- Rectified Linear Unit (ReLU)

ReLU and Some Variants

$$a_{\mathrm{ReLU}}(x) = \max(0, x)$$

$$a_{\mathrm{s}}(x) = \log(1 + e^{x})$$

$$a_{\mathrm{n}}(x) = \max(0, x + \varphi),$$
with $\varphi \sim \mathcal{N}\big(0, \sigma(x)\big)$,

$$a_{\mathrm{L}}(x) = \begin{cases} x, & if\ x > 0 \\ \delta x, & otherwise \end{cases}$$
with δ a small positive number

x

The Importance of Regularization

- The parameter space is huge, if there is no constraint in search for a solution, the algorithm may converge to poor solutions.

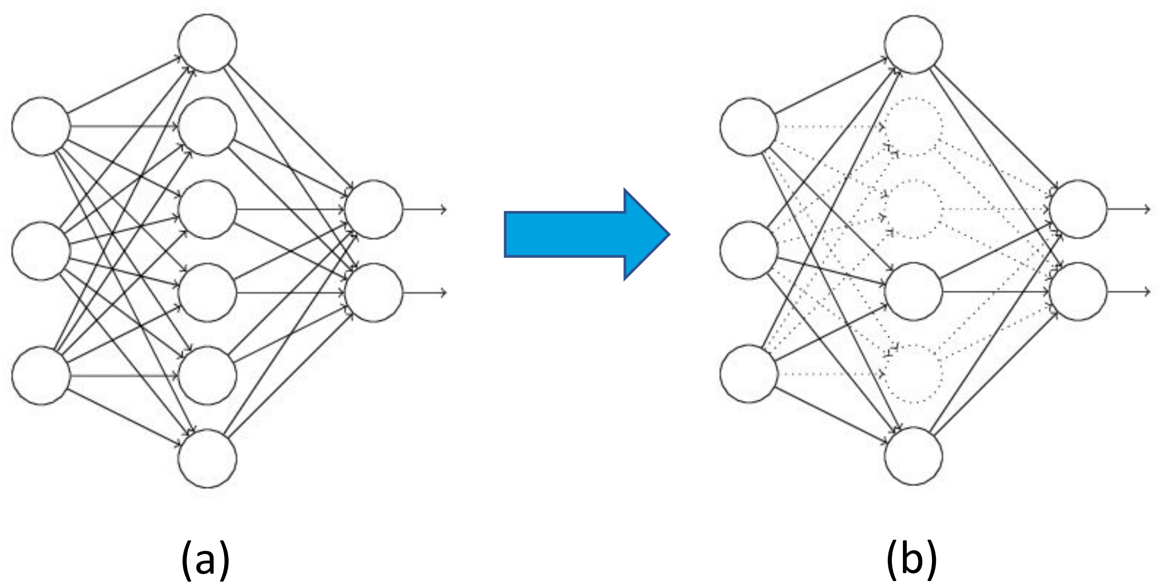
- Overfitting is a typical problem
  - Converging to local minimum good only for the training data

Some Ideas for Regularization

- Favoring a network with small weights
  - achieved by adding a term of L2-norm of the weights to original loss function
- Preventing neurons from “co-adaptation” ➔ Drop-out
- Making the network less sensitive to initialization/learning rate etc.
  ➔ Batch normalization
- Such regularization techniques have been found to be not only helpful but sometimes critical to learning in deep networks

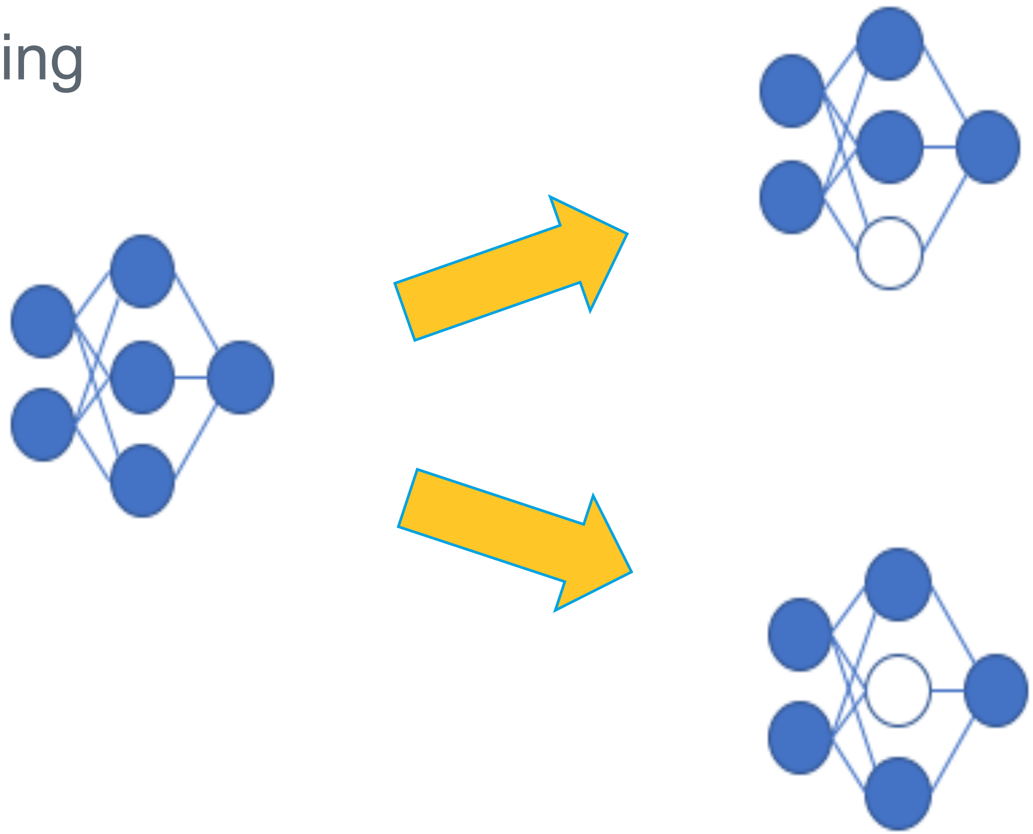
Drop-out

1. Obtain (b) by randomly deactivate some hidden nodes in (a)
2. For input x, calculate output y by using the activated nodes ONLY
3. Use BP to update weights (which connect to the activated nodes) of network
4. Activate all nodes
5. Go back to first step



Why Drop-out?

- Reducing co-adaptation of neuron
- Model averaging



Batch Normalization (BN)

- Inputs to network layers are of varying distributions, the so-called internal covariate shift [Ioffe and Szegedy, 2015]
  - Careful parameter initialization and low learning rate are required

- BN was developed to solve this problem by normalizing layer inputs of a batch

The Simple Math of BN

- For a mini-batch with size = m, first calculate

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

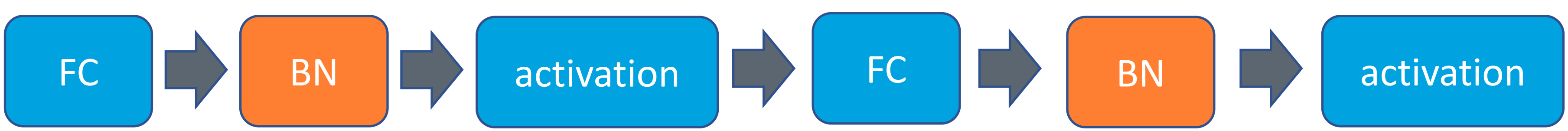
- Up to this point, $\widehat{x}$ has mean = 0 and standard deviation = 1

How is BN Used in Learning?

- Define two parameters β and γ so that the output of the BN layer can be calculated as:

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

- Parameters β and γ can be learned by minimizing the lost function via gradient descent
- Usually used right before the activation functions



Other Regularization Techniques

- Weight sharing
- Training data conditioning
- Sparsity constraints
- Ensemble methods (committee of networks)

→ Some of these will be discussed in later examples of networks.