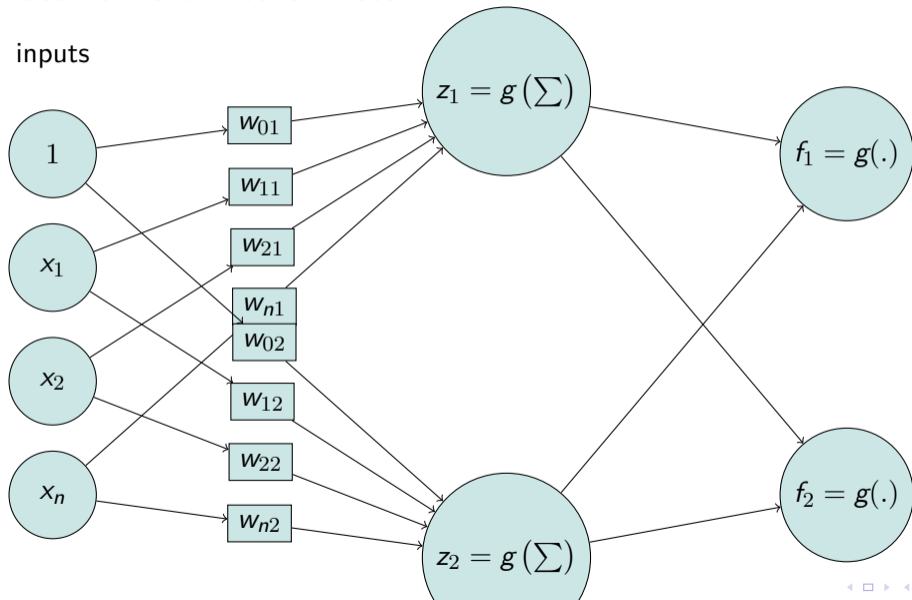


# Lecture 20 contd: Neural Network Training using Backpropagation, Convolutional And Recurrent Neural Networks

Instructor: Prof. Ganesh Ramakrishnan

# Feed-forward Neural Nets

inputs



# Training a Neural Network

STEP 0: Pick a network architecture

- Number of input units: Dimension of features  $\phi(\mathbf{x}^{(i)})$ .
- Number of output units: Number of classes.
- Reasonable default: 1 hidden layer, or if  $>1$  hidden layer, have same number of hidden units in every layer.
- Number of hidden units in each layer a constant factor (3 or 4) of dimension of  $x$ .
- We will use
  - ▶ the smooth sigmoidal function  $g(s) = \frac{1}{1+e^{-s}}$ : **We have now learnt how to train a single node sigmoidal (LR) neural network**
  - ▶ instead of the non-smooth step function  $g(s) = 1$  if  $s \in [\theta, \infty)$  and  $g(s) = 0$  otherwise.

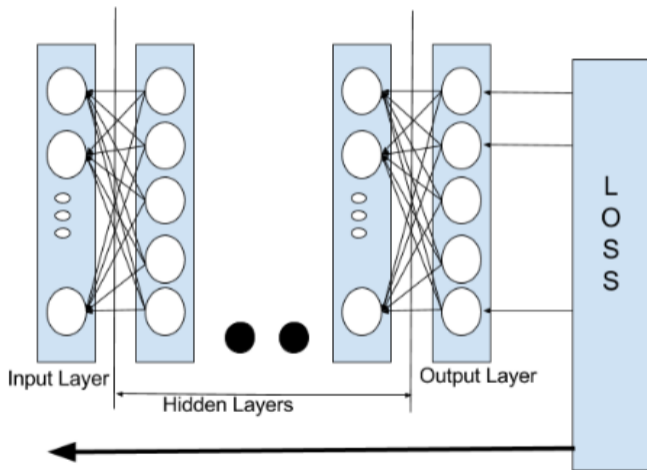
# High Level Overview of Backpropagation Algorithm for Training NN

- 1 Randomly initialize weights  $w_{ij}^l$  for  $l = 1, \dots, L$ ,  $i = 1, \dots, s_l$ ,  $j = 1, \dots, s_{l+1}$ .
- 2 Implement **forward propagation** to get  $f_w(\mathbf{x})$  for any  $x \in \mathcal{D}$ .
- 3 Execute **backpropagation**
  - 1 by computing partial derivatives  $\frac{\partial}{\partial w_{ij}^{(l)}} E(w)$  for  $l = 1, \dots, L$ ,  $i = 1, \dots, s_l$ ,  $j = 1, \dots, s_{l+1}$ .
  - 2 and using gradient descent to try to minimize (non-convex)  $E(w)$  as a function of parameters  $w$ .

$$w_{ij}^l = w_{ij}^l - \eta \frac{\partial}{\partial w_{ij}^{(l)}} E(w)$$

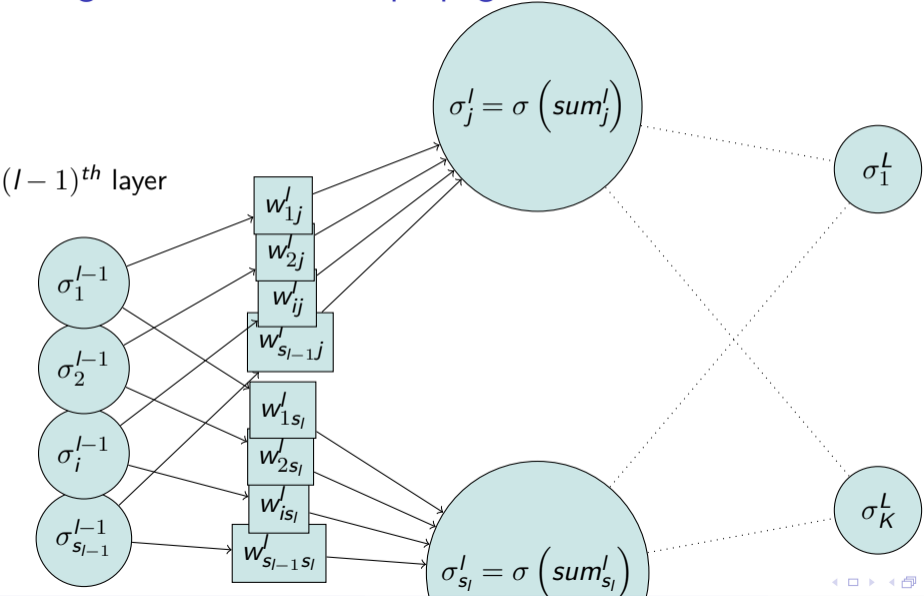
- 4 Verify that the cost function  $E(w)$  has indeed reduced, else resort to some random perturbation of weights  $w$ .

# The Backpropagation Algorithm



Iteratively update the weights along the direction where Loss decreases.

# Setting Notation for Backpropagation



## Gradient Computation

- The Neural Network objective to be minimized:

$$E(\mathbf{w}) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log \left( \sigma_k^L(\mathbf{x}^{(i)}) \right) + (1 - y_k^{(i)}) \log \left( 1 - \sigma_k^L(\mathbf{x}^{(i)}) \right) \right] + \frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{s_{l-1}} \sum_{j=1}^{s_l} (w_{ij}^l)^2 \quad (1)$$

# Gradient Computation

- The Neural Network objective to be minimized:

$$E(\mathbf{w}) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log \left( \sigma_k^L(\mathbf{x}^{(i)}) \right) + (1 - y_k^{(i)}) \log \left( 1 - \sigma_k^L(\mathbf{x}^{(i)}) \right) \right] + \frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{s_{l-1}} \sum_{j=1}^{s_l} (w_{ij}^l)^2 \quad (1)$$

- $sum_j^l = \sum_{k=1}^{s_{l-1}} w_{kj}^l \sigma_k^{l-1}$  and  $\sigma_i^l = \frac{1}{1 + e^{-sum_i^l}}$
- $\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial \sigma_j^l} \frac{\partial \sigma_j^l}{\partial sum_j^l} \frac{\partial sum_j^l}{\partial w_{ij}^l} + \frac{\lambda}{2m} w_{ij}^l$
- $\frac{\partial \sigma_j^l}{\partial sum_j^l} = \left( \frac{1}{1 + e^{-sum_j^l}} \right) \left( 1 - \frac{1}{1 + e^{-sum_j^l}} \right) = \sigma_j^l (1 - \sigma_j^l)$
- $\frac{\partial sum_j^l}{\partial w_{ij}^l} = \frac{\partial}{\partial w_{ij}^l} \left( \sum_{k=1}^{s_{l-1}} w_{kj}^l \sigma_k^{l-1} \right) = \sigma_i^{l-1}$



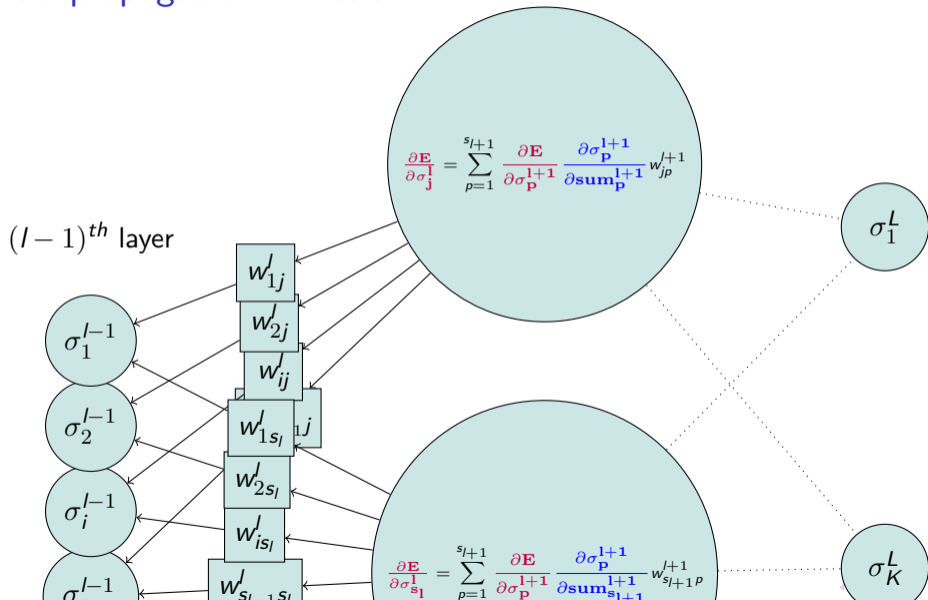
- For a single example  $(\mathbf{x}, y)$ :

$$\begin{aligned}
 & - \left[ \sum_{k=1}^K y_k \log(\sigma_k^L(\mathbf{x})) + (1 - y_k) \log(1 - \sigma_k^L(\mathbf{x})) \right] \\
 & + \frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{s_{l-1}} \sum_{j=1}^{s_l} (w_{ij}^l)^2
 \end{aligned} \tag{2}$$

- $\frac{\partial \mathbf{E}}{\partial \sigma_j^l} = \sum_{p=1}^{s_{l+1}} \frac{\partial E}{\partial \text{sum}_p^{l+1}} \frac{\partial \text{sum}_p^{l+1}}{\partial \sigma_j^l} = \sum_{p=1}^{s_{l+1}} \frac{\partial \mathbf{E}}{\partial \sigma_p^{l+1}} \frac{\partial \sigma_p^{l+1}}{\partial \text{sum}_p^{l+1}} w_{jp}^{l+1}$  since  $\frac{\partial \text{sum}_p^{l+1}}{\partial \sigma_j^l} = w_{jp}^{l+1}$

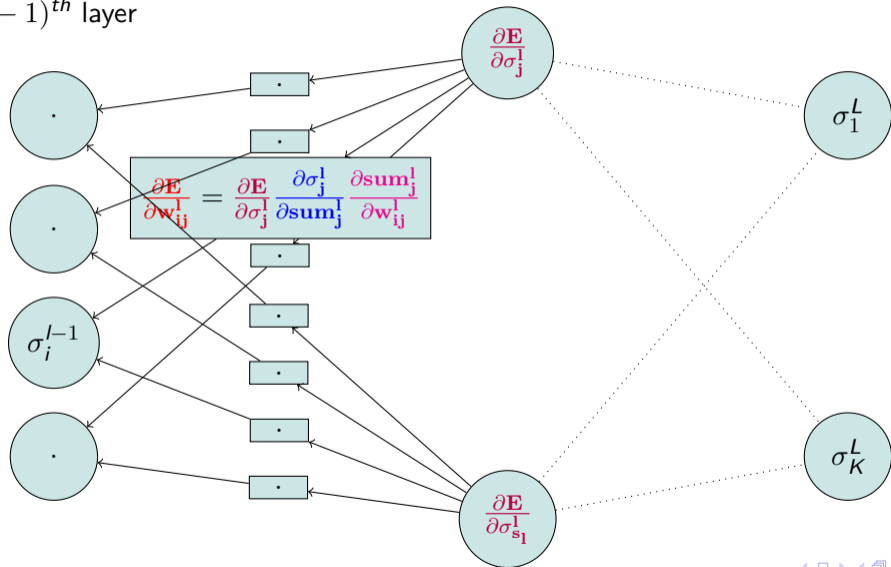
- $\frac{\partial \mathbf{E}}{\partial \sigma_j^L} = -\frac{y_j}{\sigma_j^L} - \frac{1-y_j}{1-\sigma_j^L}$

# Backpropagation in Action



# Backpropagation in Action

$(l-1)^{th}$  layer

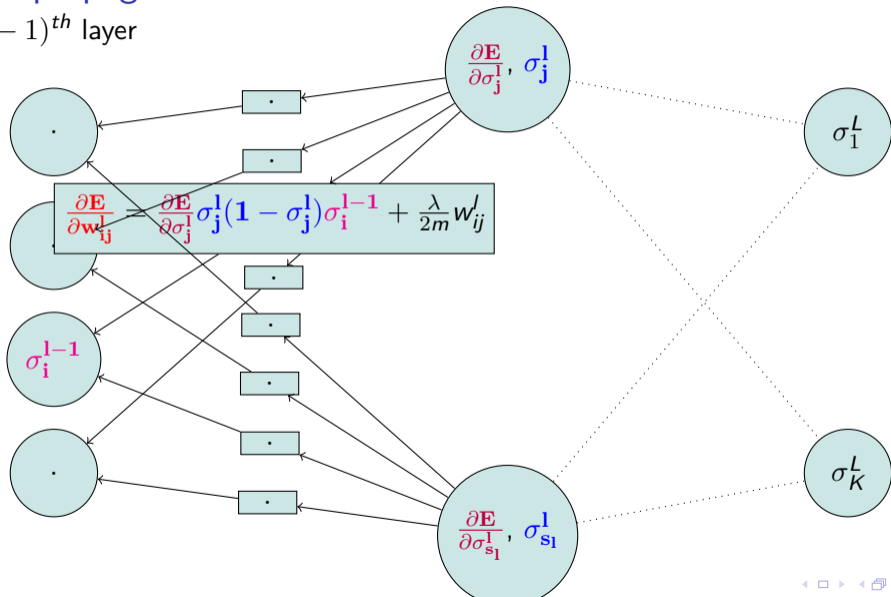


## Recall and Substitute

- $sum_j^l = \sum_{k=1}^{s_{l-1}} w_{kj}^l \sigma_k^{l-1}$  and  $\sigma_i^l = \frac{1}{1+e^{-sum_i^l}}$
- $\frac{\partial \mathbf{E}}{\partial w_{ij}^1} = \frac{\partial \mathbf{E}}{\partial \sigma_j^1} \frac{\partial \sigma_j^1}{\partial sum_j^1} \frac{\partial sum_j^1}{\partial w_{ij}^1} + \frac{\lambda}{2m} w_{ij}^1$
- $\frac{\partial \sigma_j^1}{\partial sum_j^1} = \sigma_j^1 (1 - \sigma_j^1)$
- $\frac{\partial sum_j^1}{\partial w_{ij}^1} = \sigma_i^{1-1}$
- $\frac{\partial \mathbf{E}}{\partial \sigma_j^1} = \sum_{p=1}^{s_{l+1}} \frac{\partial \mathbf{E}}{\partial \sigma_j^{l+1}} \frac{\partial \sigma_j^{l+1}}{\partial sum_p^{l+1}} w_{jp}^{l+1}$
- $\frac{\partial \mathbf{E}}{\partial \sigma_j^L} = -\frac{y_j}{\sigma_j^L} - \frac{1-y_j}{1-\sigma_j^L}$

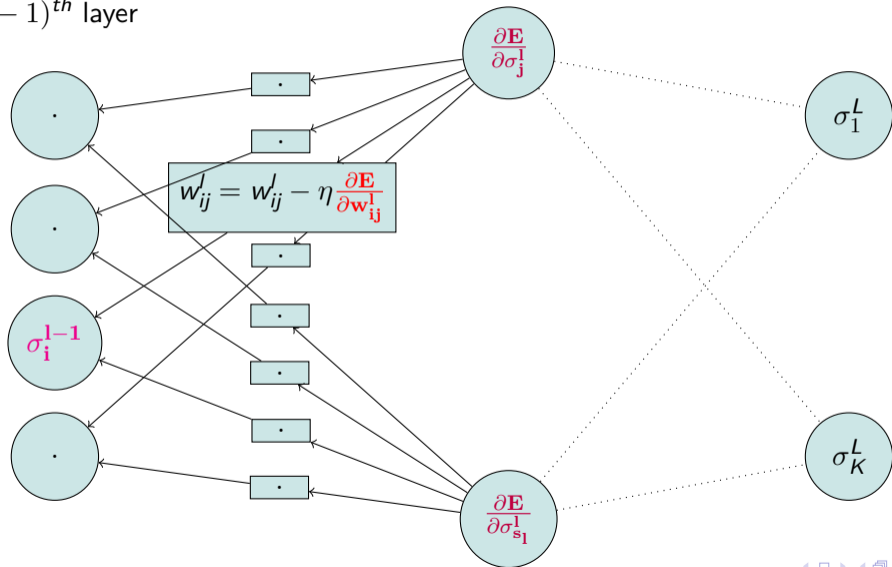
# Backpropagation in Action

$(l-1)^{th}$  layer



# Backpropagation in Action

$(l-1)^{th}$  layer



# The Backpropagation Algorithm for Training NN

- 1 Randomly initialize weights  $w_{ij}^l$  for  $l = 1, \dots, L$ ,  $i = 1, \dots, s_l$ ,  $j = 1, \dots, s_{l+1}$ .
- 2 Implement **forward propagation** to get  $f_w(\mathbf{x})$  for every  $\mathbf{x} \in \mathcal{D}$ .
- 3 Execute **backpropagation** on any misclassified  $\mathbf{x} \in \mathcal{D}$  by performing gradient descent to minimize (non-convex)  $E(\mathbf{w})$  as a function of parameters  $\mathbf{w}$ .
- 4  $\frac{\partial E}{\partial \sigma_j^L} = -\frac{y_j}{\sigma_j^L} - \frac{1-y_j}{1-\sigma_j^L}$  for  $j = 1$  to  $s_L$ .
- 5 For  $l = L - 1$  down to 2:
  - 1  $\frac{\partial E}{\partial \sigma_j^l} = \sum_{p=1}^{s_{l+1}} \frac{\partial E}{\partial \sigma_j^{l+1}} \sigma_j^{l+1} (1 - \sigma_j^{l+1}) w_{jp}^{l+1}$
  - 2  $\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial \sigma_j^l} \sigma_j^l (1 - \sigma_j^l) \sigma_i^{l-1} + \frac{\lambda}{2m} w_{ij}^l$
  - 3  $w_{ij}^l = w_{ij}^l - \eta \frac{\partial E}{\partial w_{ij}^l}$
- 6 Keep picking misclassified examples until the cost function  $E(\mathbf{w})$  shows significant reduction; else resort to some random perturbation of weights  $\mathbf{w}$  and restart a couple of times.

# Challenges with Neural Network Training



# What Changed with Neural Networks?

- Origin: Computational Model of Threshold Logic from Warren McCulloch and Walter Pitts (1943)
- Big Leap: For ImageNet Challenge, AlexNet achieved 85 % accuracy (NIPS 2012). Prev best was 75 % (CVPR 2011).
- Present best is 96.5 % MSRA (arXiv 2015). Comparable to human level accuracy.
- Challenges involved were varied background, same object with different colors (e.g. cats), varied sizes and postures of same objects, varied illuminated conditions.
- Tasks like OCR, Speech recognition are now possible without segmenting the word image/signal into character images/signals.

## LeNet(1989 and 1998) v/s AlexNet(2012)

	LeeNet 1989	LeeNet 1998	AlexNet 2012
Task	Digit Recognition	Digit Recognition	Object Recognition
# Classes	10	10	1k
Image Size	16 X 16	28 X 28	256 X 256 X 3
# Examples	7k	60k	1.2 M
units	1256	8084	658 k
parameters	9760	60k	60 M
connections	65k	344k	652M
Summation	Sigmoid	Sigmoid	ReLU
GPU/ Non-GPU	Non-GPU based.	Non-GPU based.	GPU based.
Operations	11 billion	412 billions	200 quadrillions

## Reasons for Big Leap

- Why LeeNet was not as successful as AlexNet, though the algorithm was same?
- Right algorithm at wrong time.
- Modern features.
- Advancement in Machine learning.
- Realistic data collection in huge amount due to: regular competitions, evaluation metrics or challenging problem statements.
- Advances in Computational Resources: GPUs, industrial scale clusters.
- Evolution of tasks: Classification of 10 objects to 100 objects to “structure of classes”.

# Convolutional Neural Network

- Variation of multi layer feedforward neural network designed to use minimal preprocessing with wide application in image recognition and natural language processing
- Traditional multilayer perceptron(MLP) models do not take into account spatial structure of data and suffer from curse of dimensionality
- Convolution Neural network has smaller number of parameters due to **local connections** and **weight sharing**

