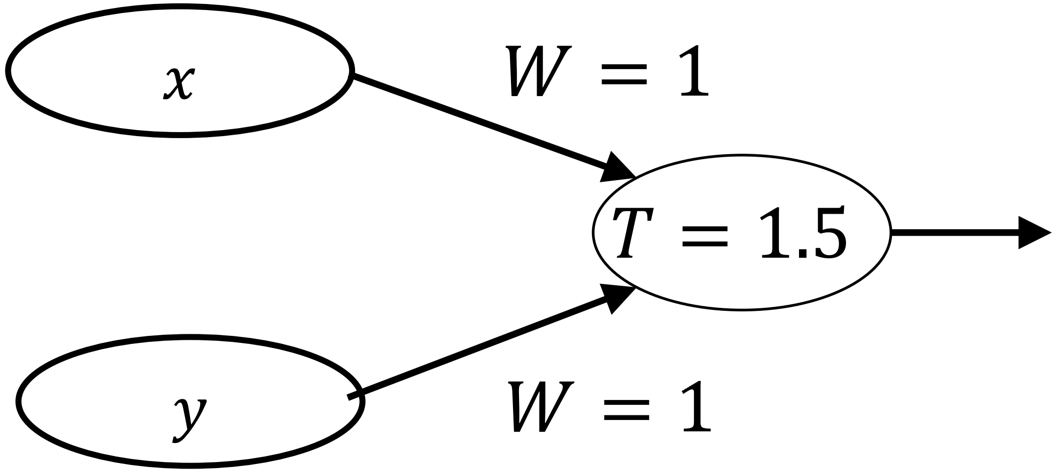
Basics of Neural Networks and Convolutional Neural Networks

Chetan Arora

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

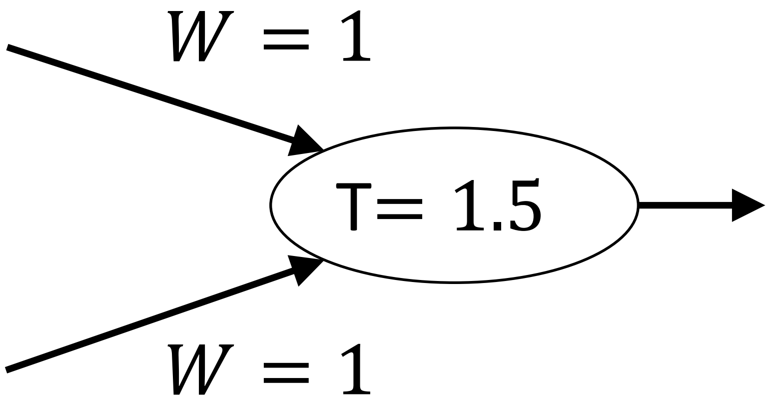
Perceptron



$$\text{output} = \begin{cases} 1 & \text{if } \displaystyle\sum_{i=0}^{n} W_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

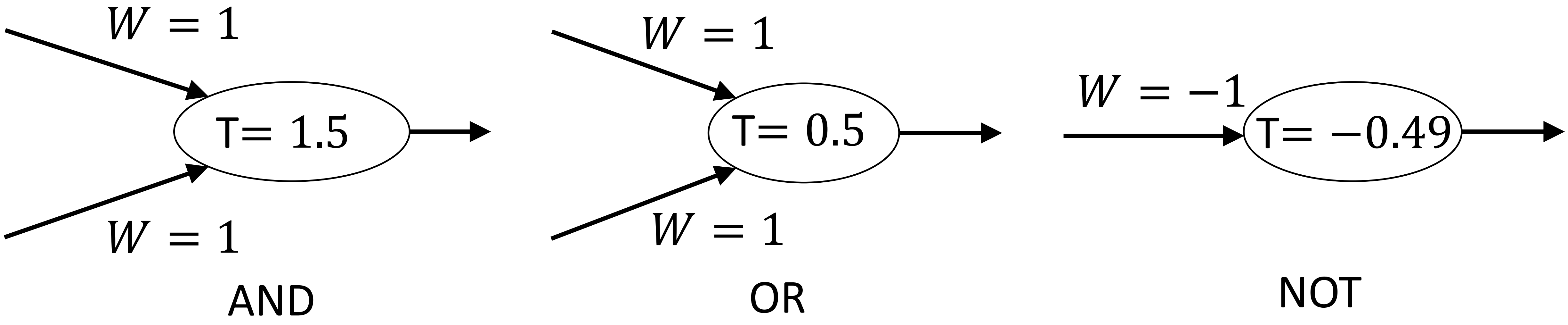
Perceptron



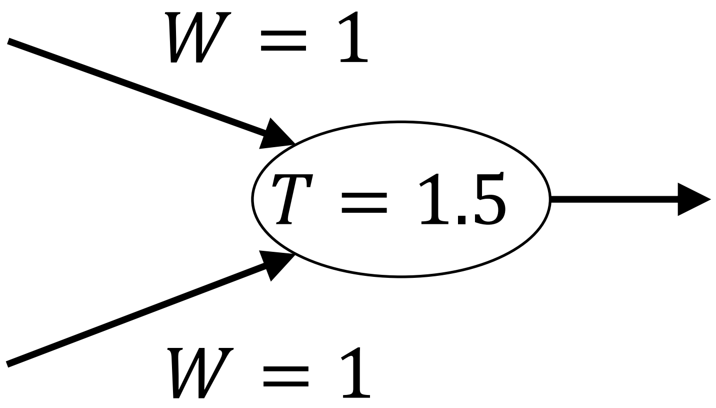
AND

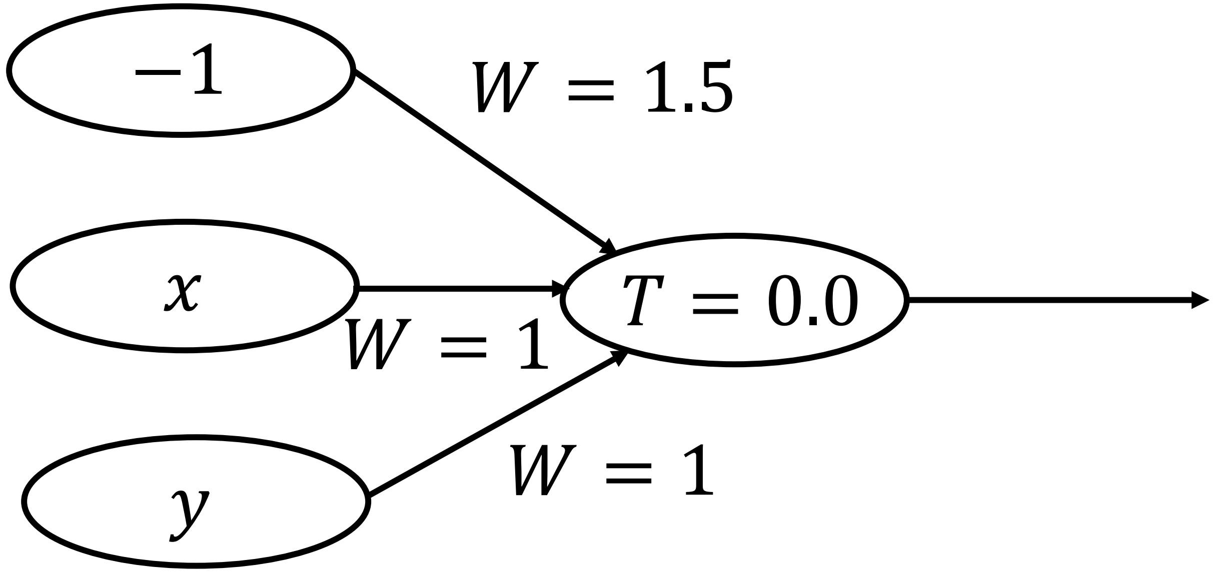
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Perceptron



Perceptron: Linear Threshold Unit



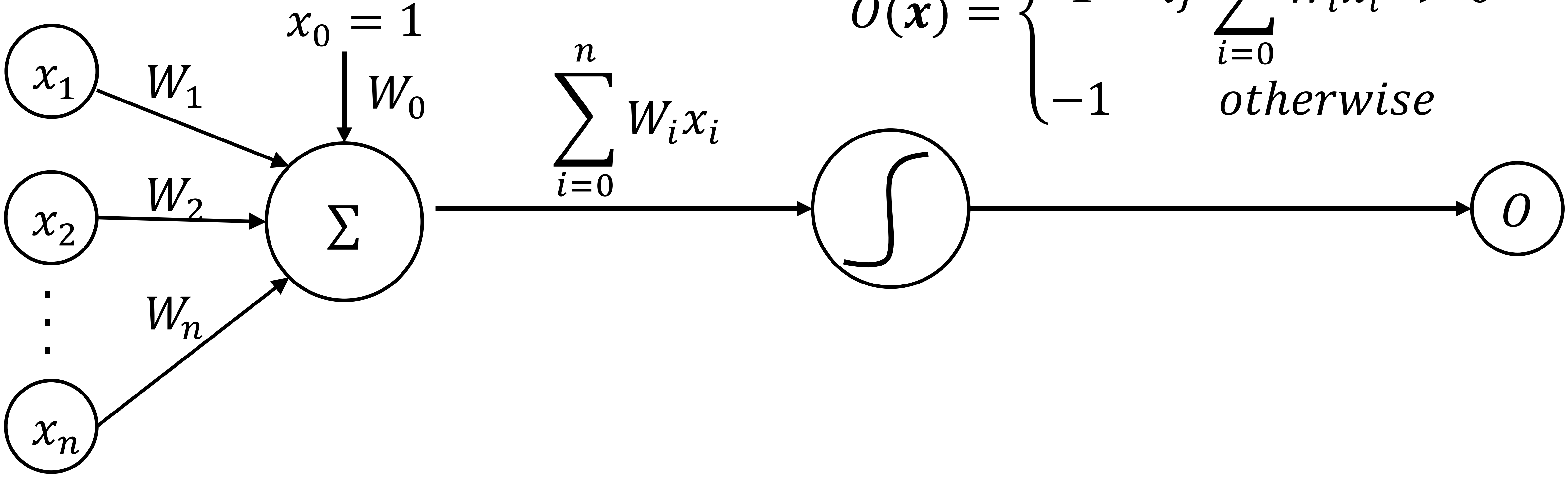


$$output = \begin{cases} 1 & if \sum_{i=0}^{n} W_i x_i \;\; > 0 \\ -1 & otherwise \end{cases}$$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Perceptron

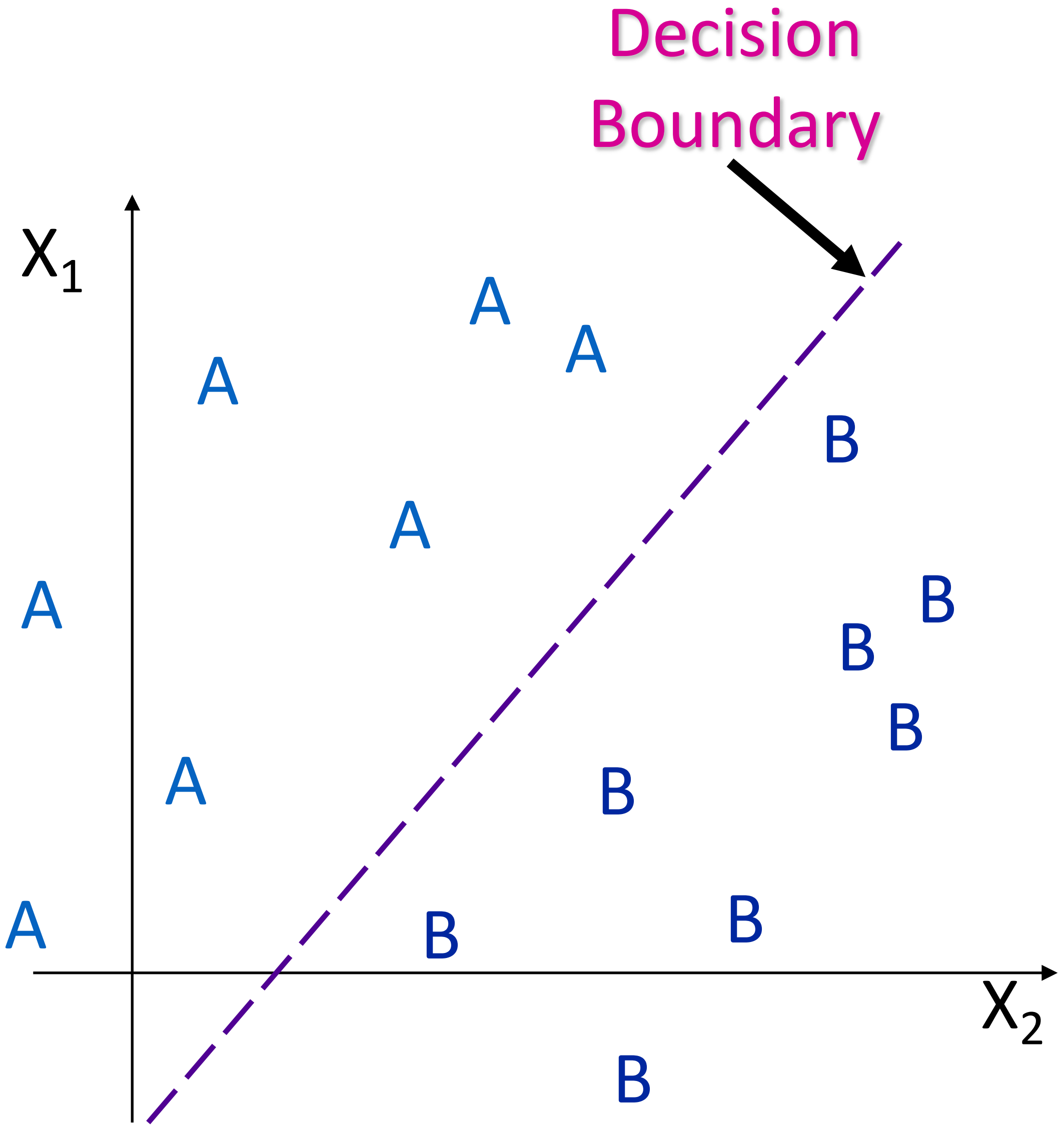
$$O(\boldsymbol{x}) = \begin{cases} 1 & if \sum_{i=0}^{n} W_i x_i > 0 \\ -1 & otherwise \end{cases}$$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Decision boundaries

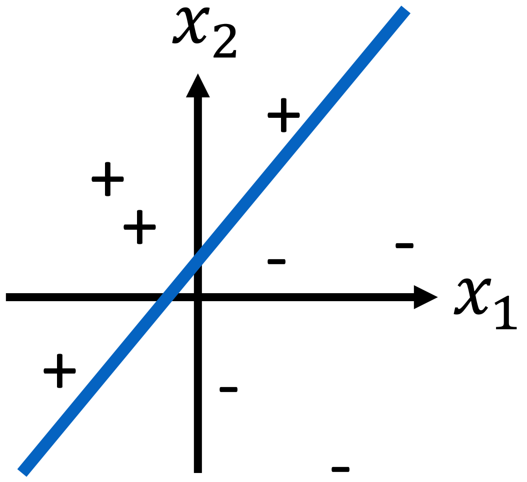
- In simple cases, divide feature space by drawing a hyperplane across it.
- Known as a decision boundary.
- Discriminant function: returns different values on opposite sides. (straight line)
- Problems which can be thus classified are linearly separable.



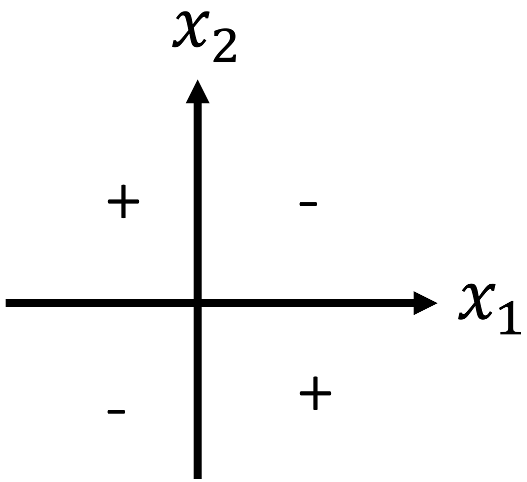
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Decision Surface of a Perceptron

- Perceptron is able to represent some useful functions
- $AND(x_1, x_2)$ choose weights $w_0 = -1.5, w_1 = 1, w_2 = 1$
- But functions that are not linearly separable (e.g. XOR) are not representable



Linearly separable



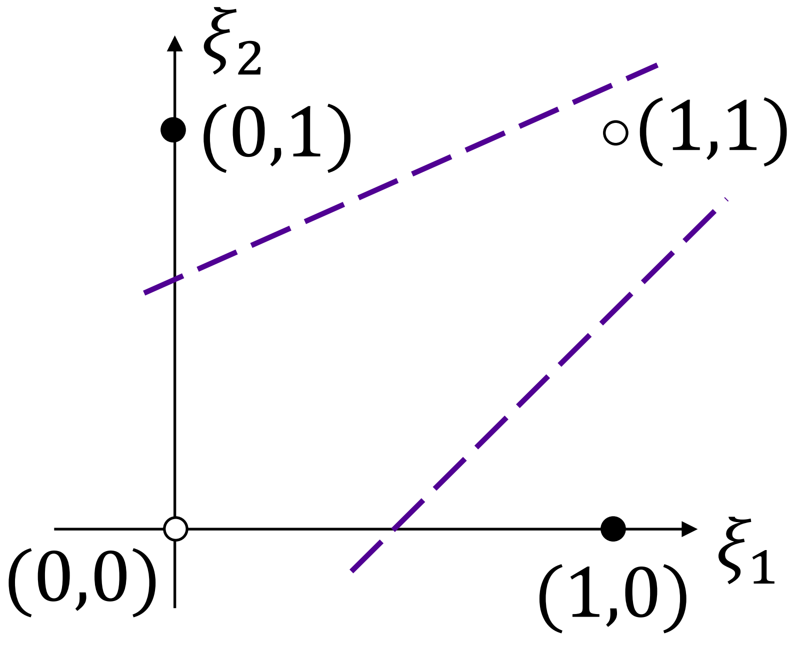
Non-Linearly separable

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Solution in 1980s: Multilayer Perceptrons

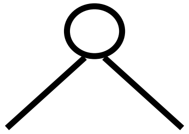
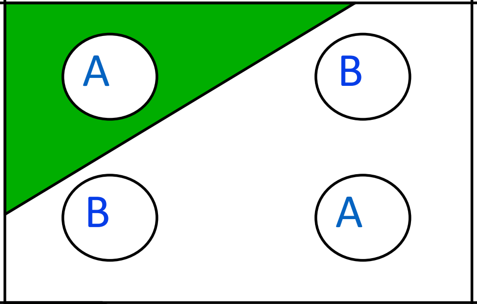
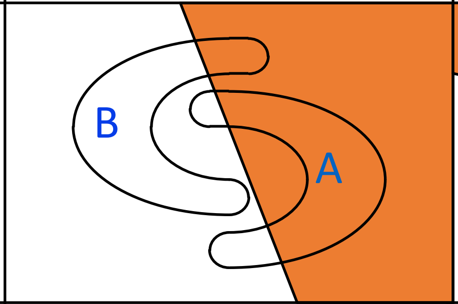
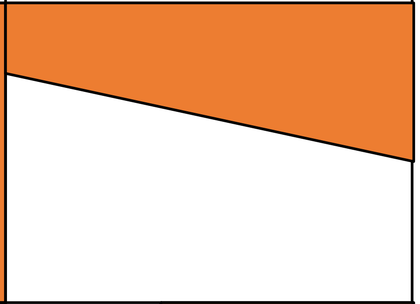
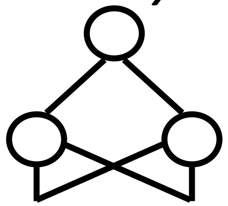
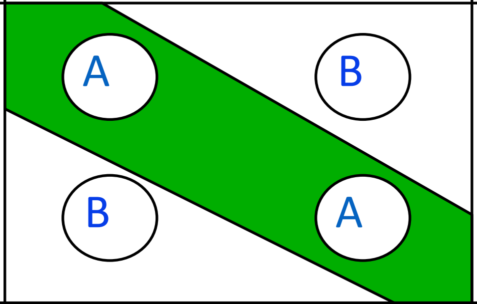
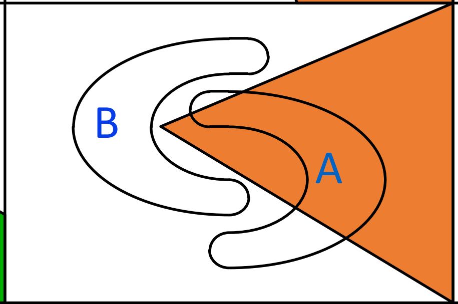
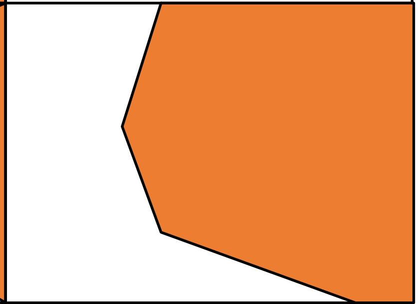
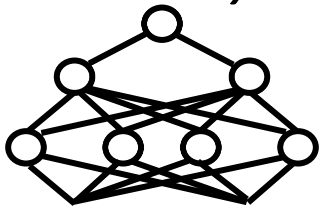
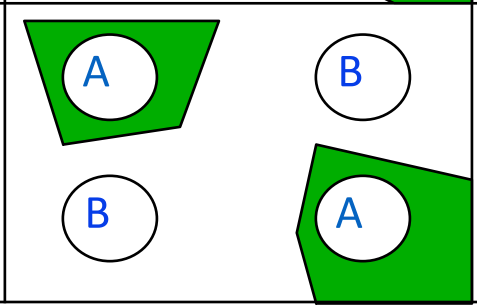
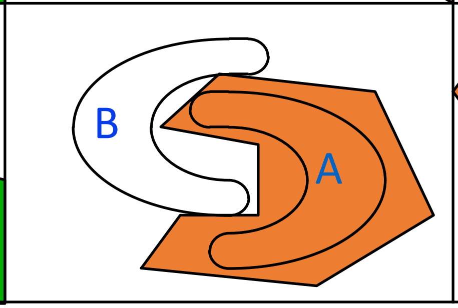
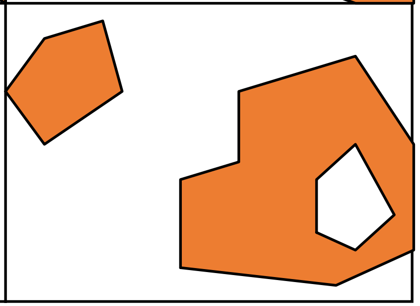
- Removes many limitations of single-layer networks
  - Can solve XOR

- Exercise: Draw a two-layer perceptron that computes the XOR function
  - 2 binary inputs ξ_1 and ξ_2
  - 1 binary output
  - One “hidden” layer
  - Find the appropriate weights and threshold



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

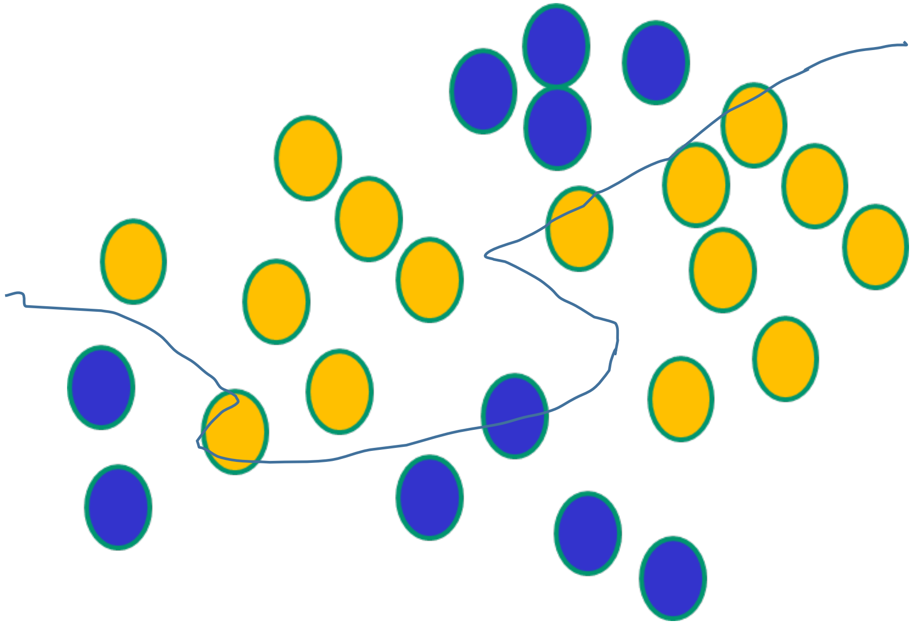
Different Non-Linearly Separable Problems

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

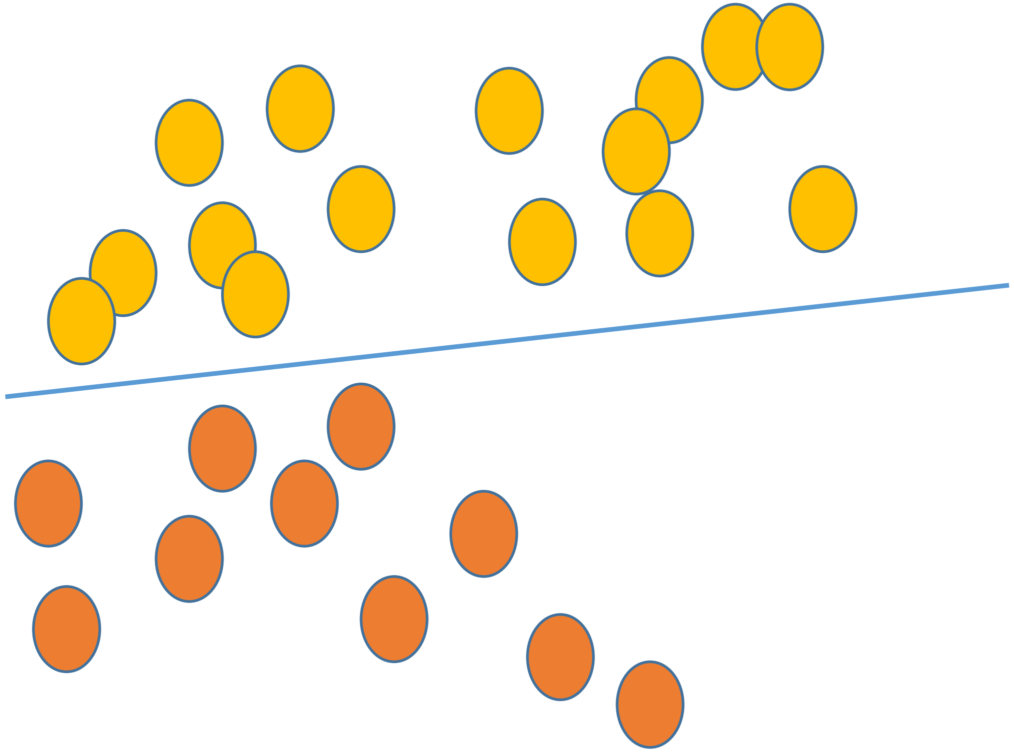
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Neural Networks: Two Perspectives

Use nonlinear $f(x)$ to draw complex boundaries, but keep the data unchanged



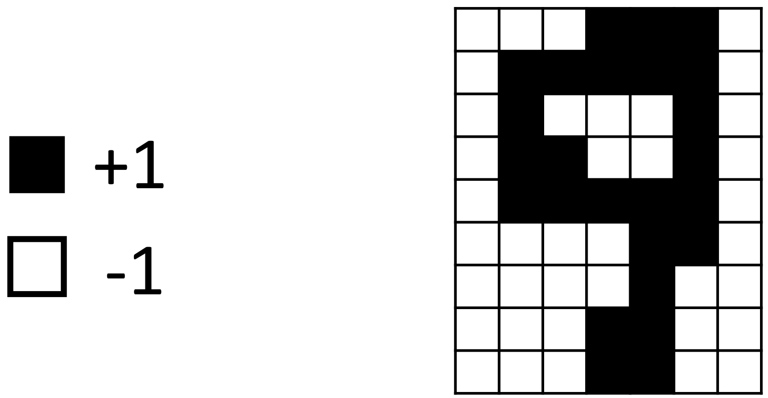
Find linear boundaries only, but transform the data first in a way that makes linear boundaries OK

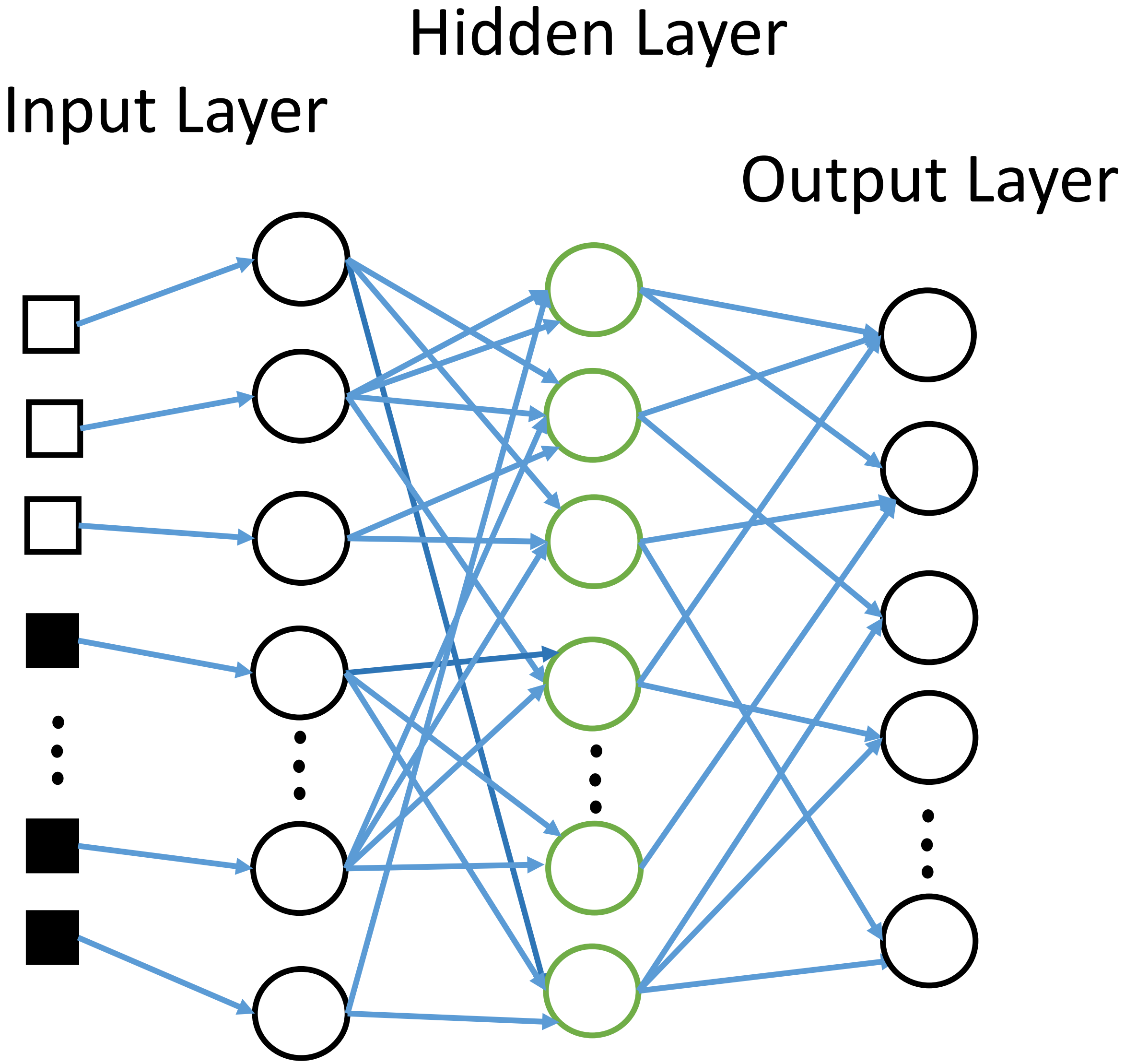


Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Neural Networks as Learning Representations

Handwritten images

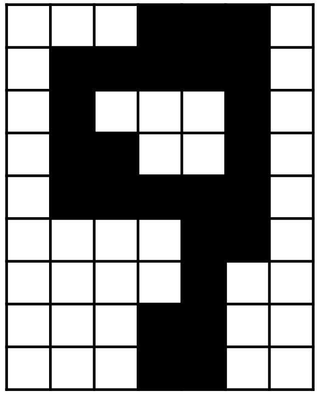


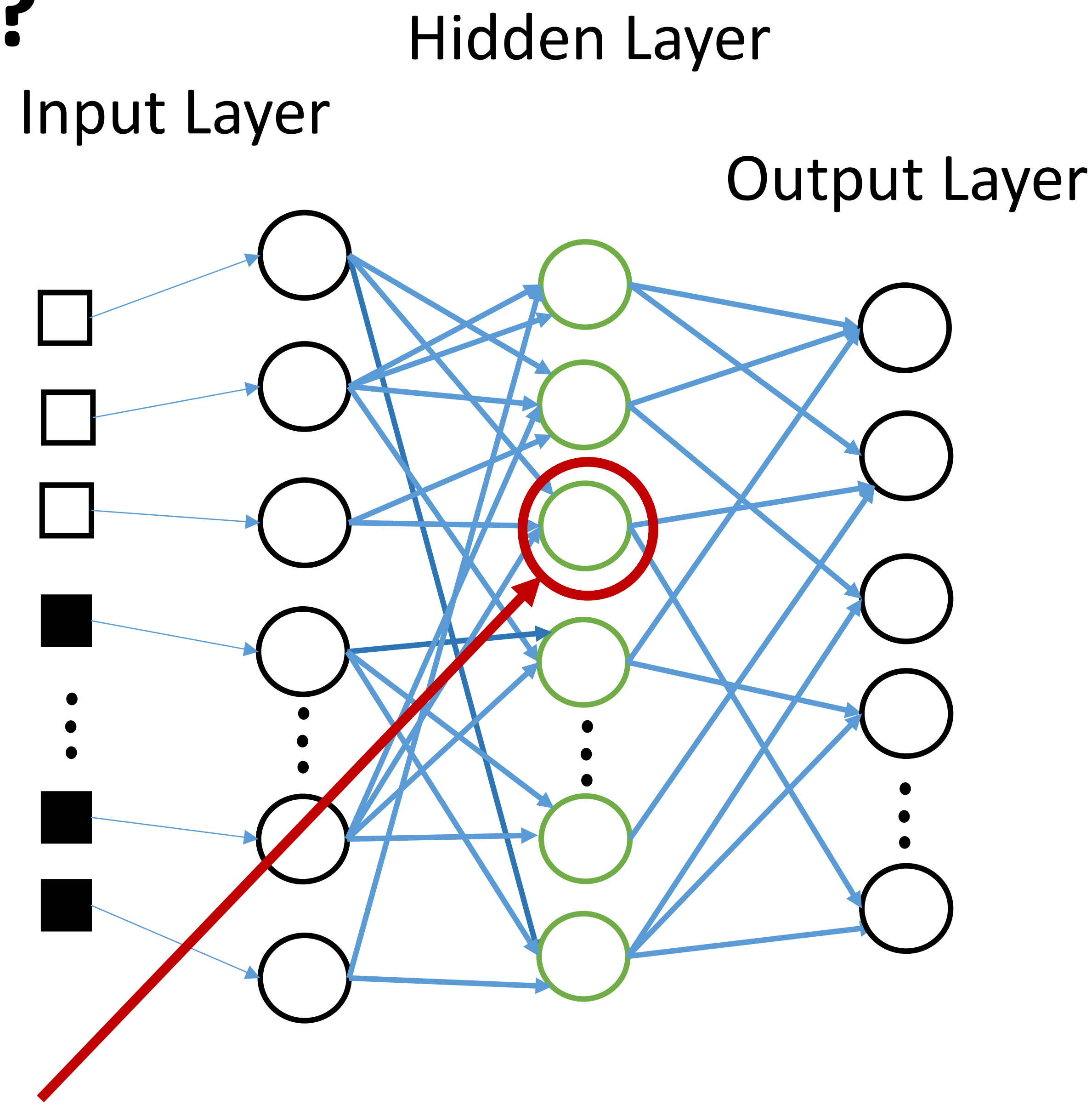


Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

What is this unit doing?

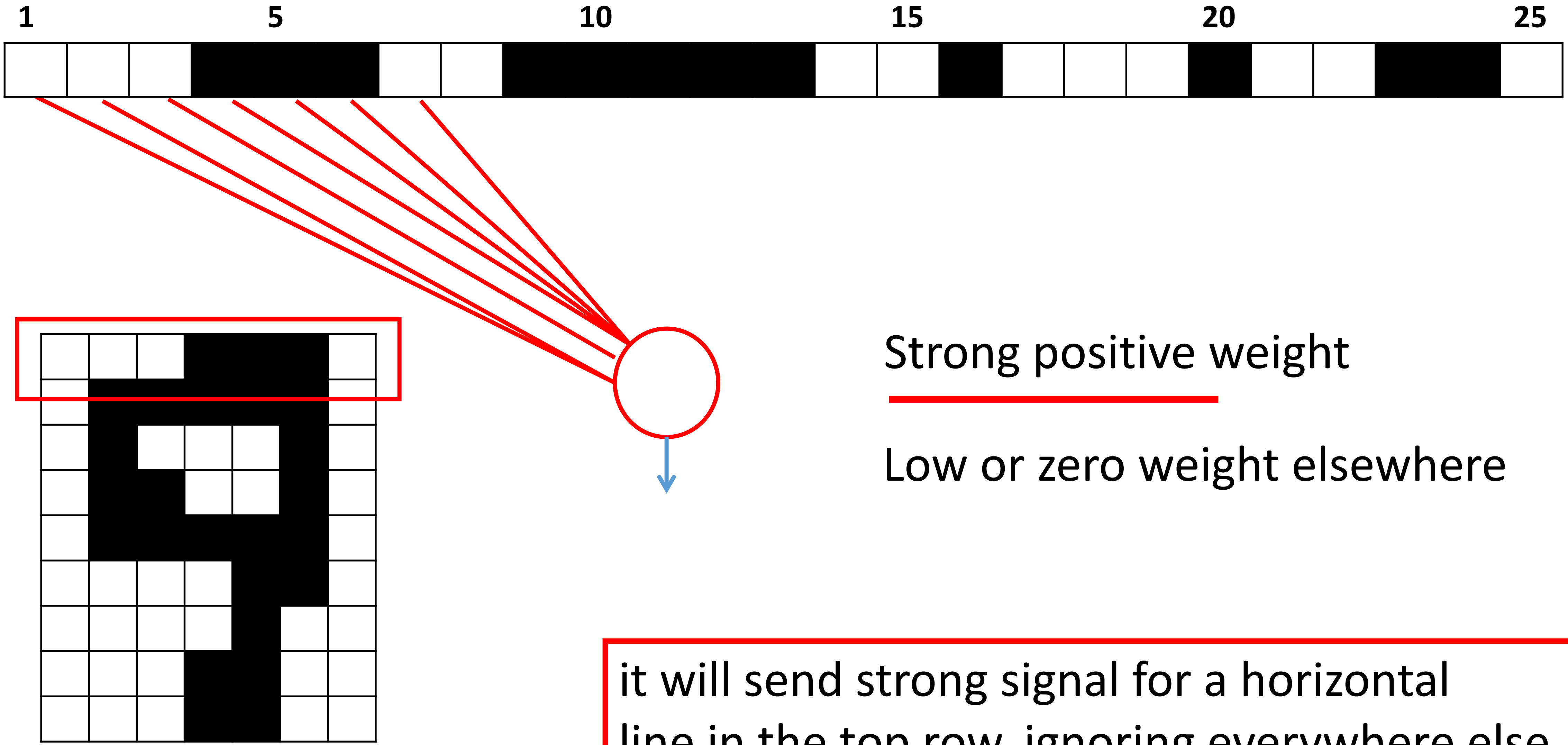
Handwritten images





Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

What does this unit detect?



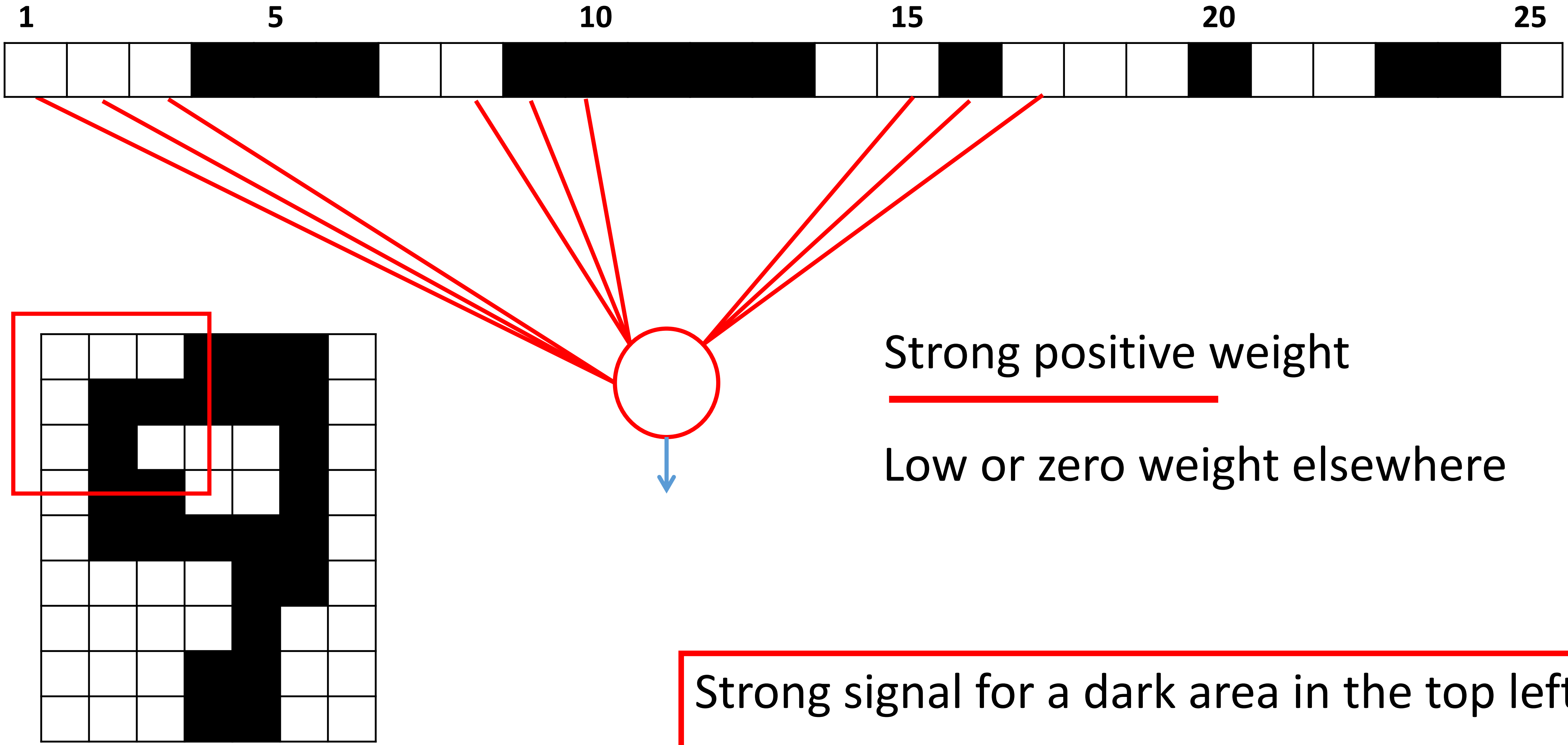
Strong positive weight

Low or zero weight elsewhere

it will send strong signal for a horizontal line in the top row, ignoring everywhere else

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

What does this unit detect?



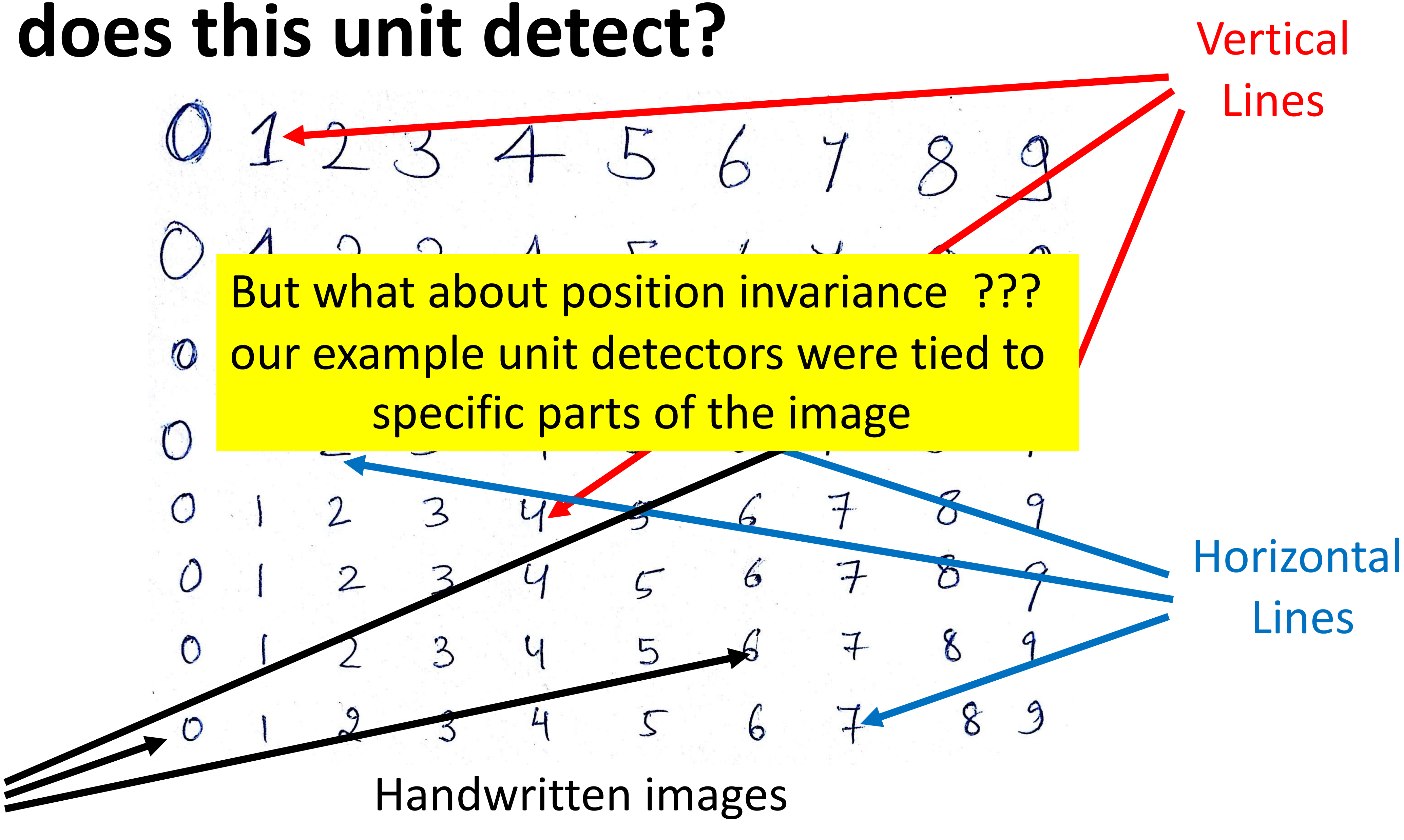
Strong positive weight

Low or zero weight elsewhere

Strong signal for a dark area in the top left corner

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

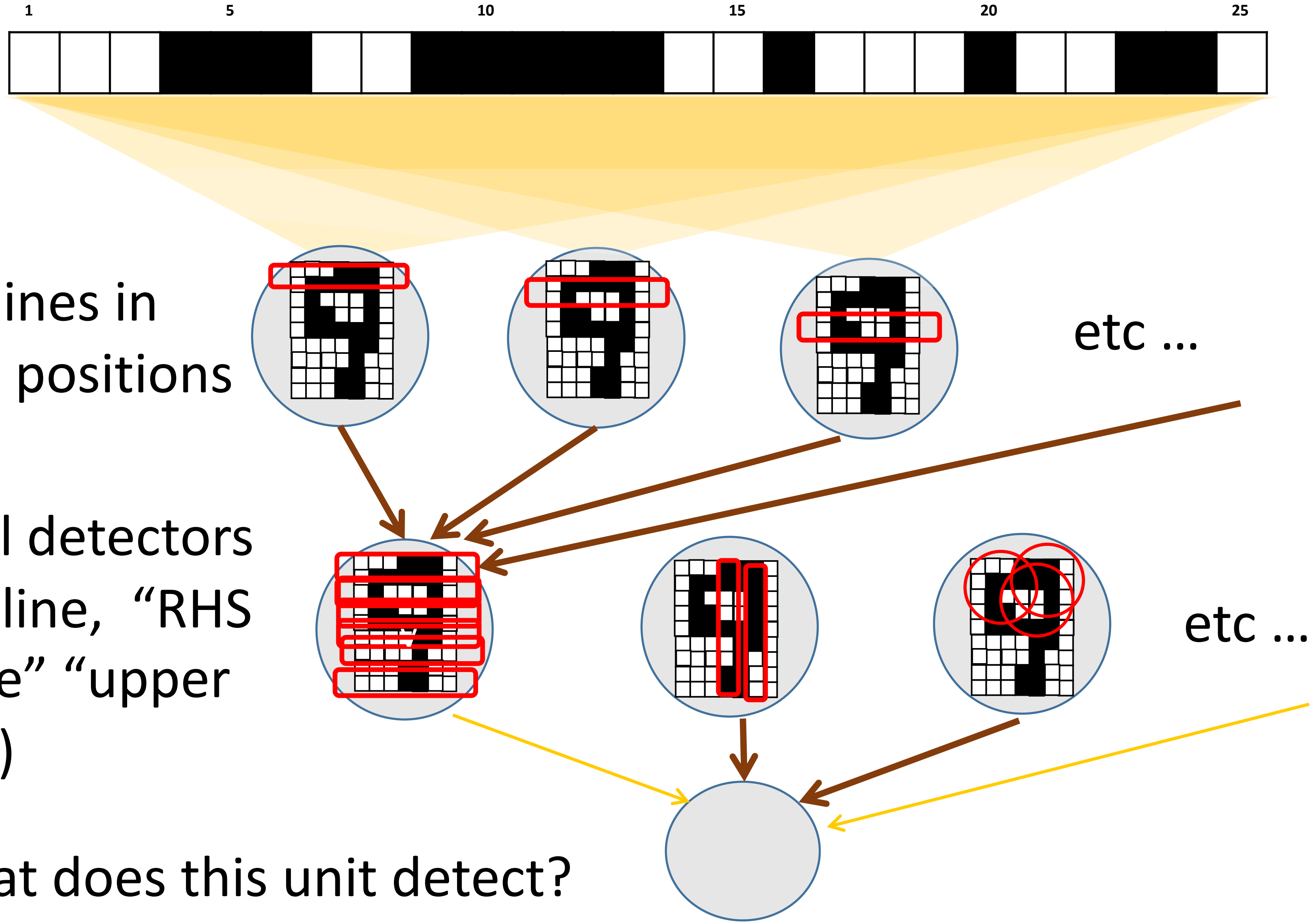
What does this unit detect?



But what about position invariance ???
our example unit detectors were tied to specific parts of the image

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Successive layers can learn higher representations



detect lines in specific positions

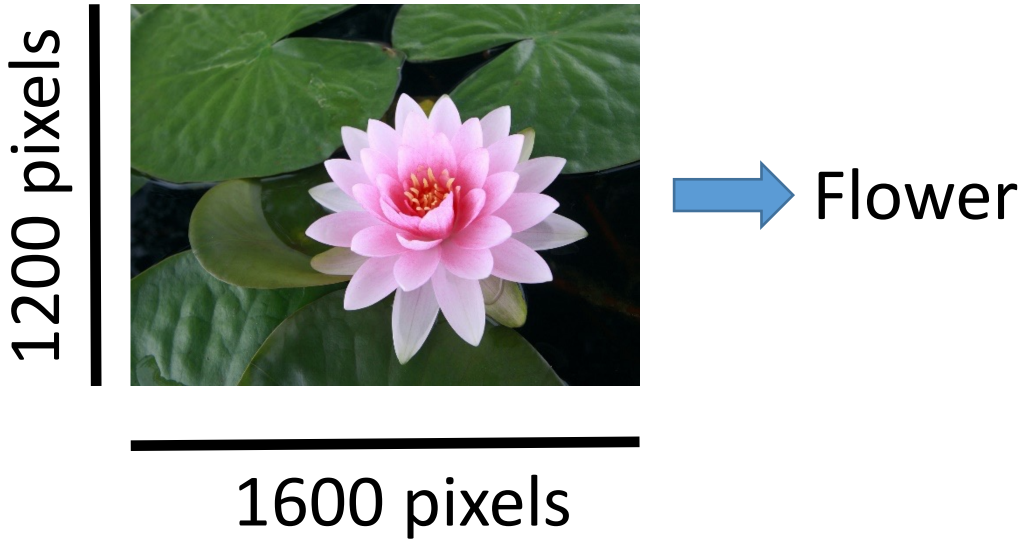
Higher level detectors (horizontal line, "RHS vertical lune" "upper loop", etc...)

What does this unit detect?

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Scaling-up

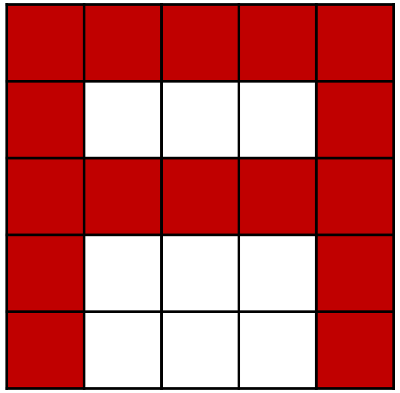
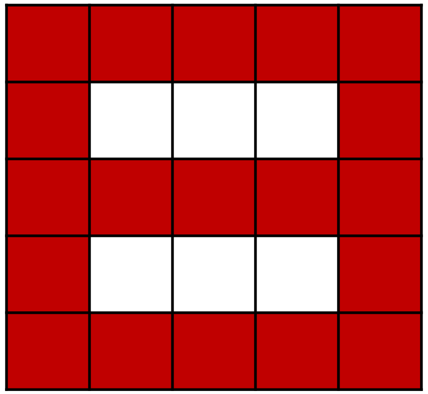
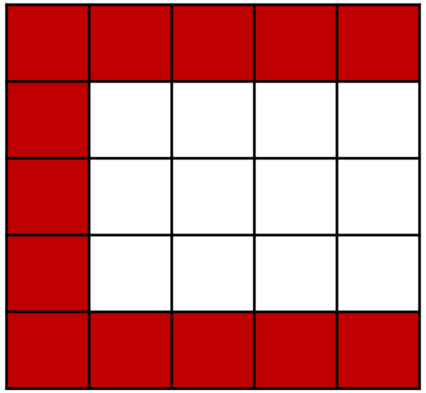
- Challenges of practical image classification
  - Must deal with very high-dimensional inputs
  - 1600×1200 pixels = 1.9m inputs, or 3×1.9m if RGB pixels
  - Can we exploit the 2D topology of pixels (or 3D for video data)?
  - Can we build invariance to certain variations we can expect
    - Translations, illumination, etc.



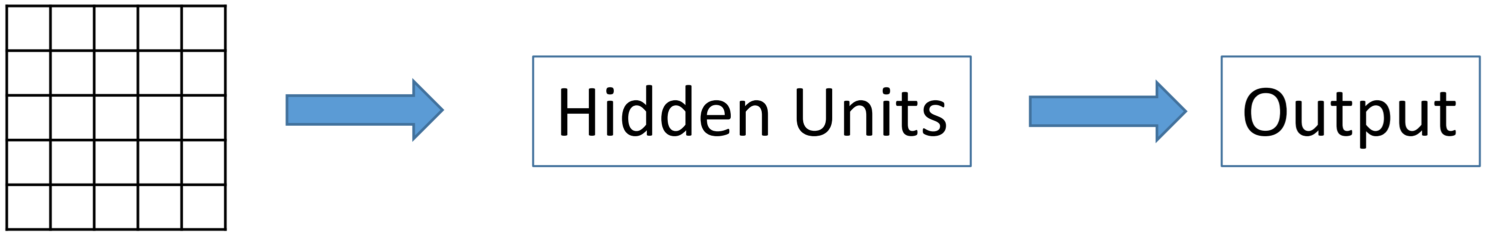
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Motivation: Recognizing an Image

- Input is 5x5 pixel array

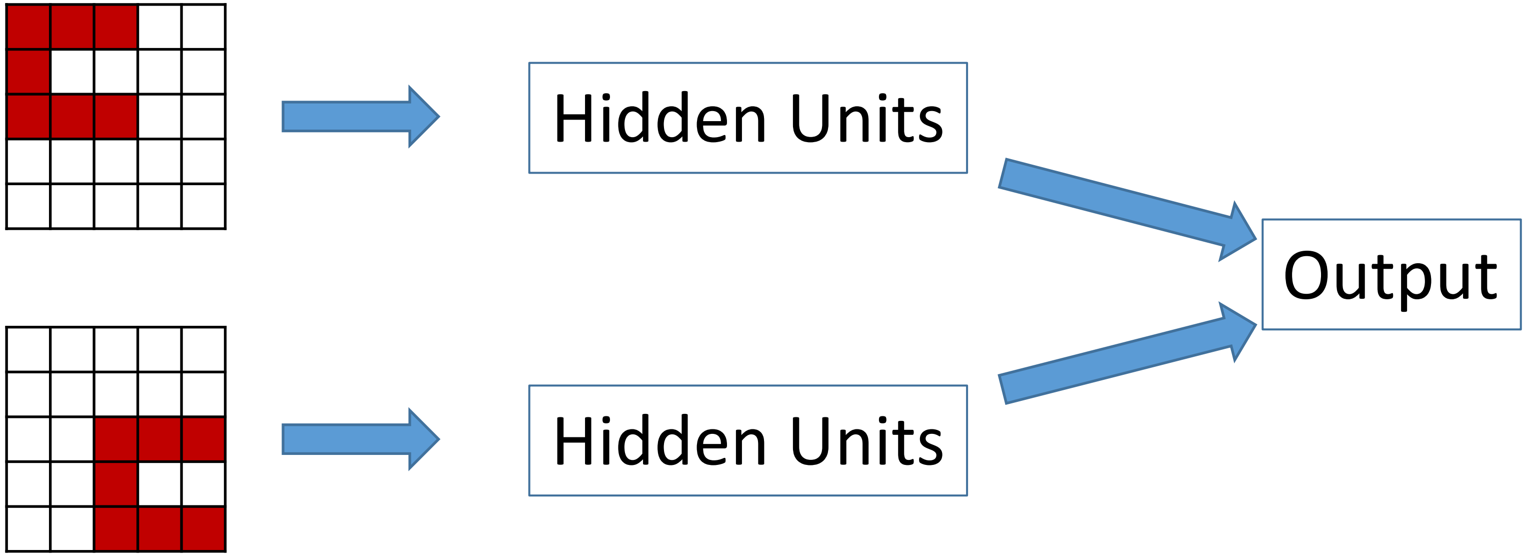
- Simple back propagation net



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Recognizing an Image with unknown location

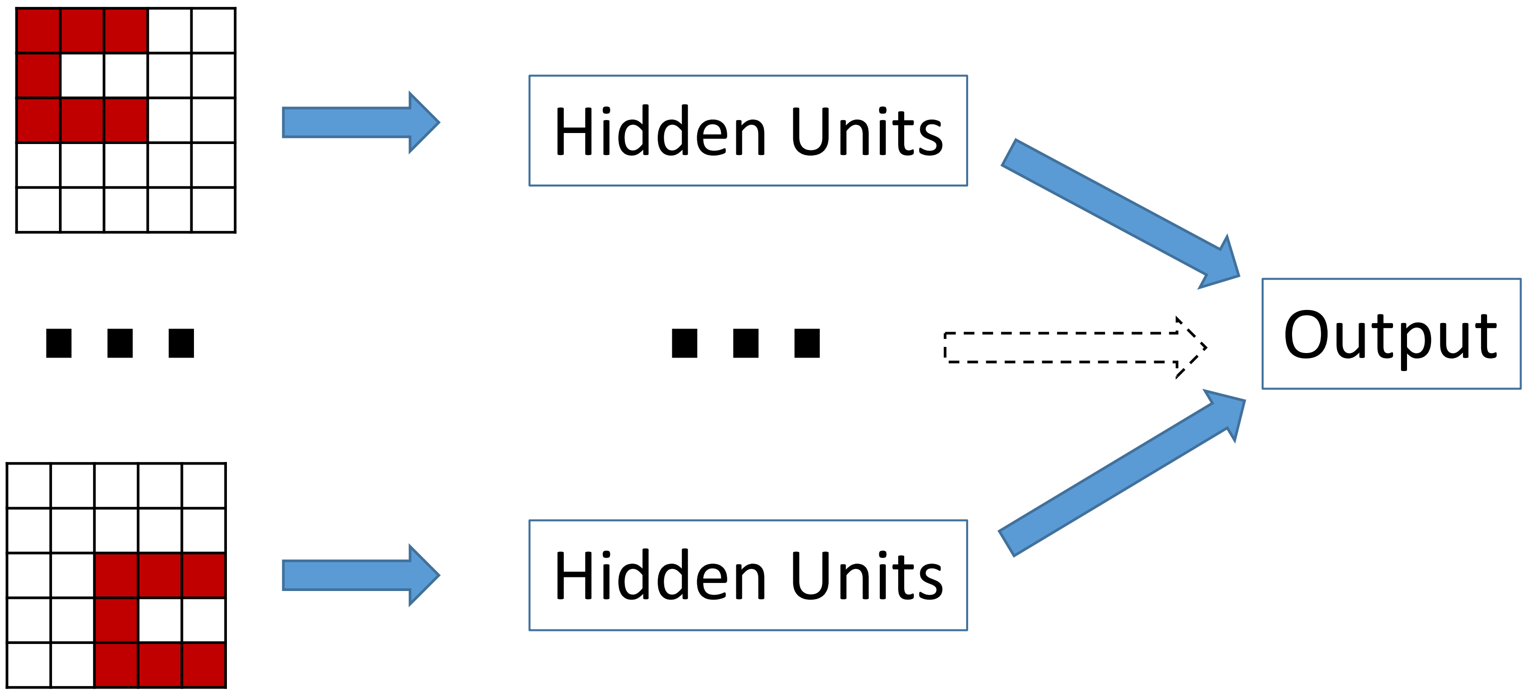
- Object can appear either in the top image or in the bottom image location
- Output indicates presence of object regardless of position
- What do we know about the weights?



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Recognizing an Image with any location

- Each possible location the object can appear in has its own set of hidden units
- Each set detects the same features except in a different location
- Locations can overlap



Key Ideas for a Usable ML Model for Images

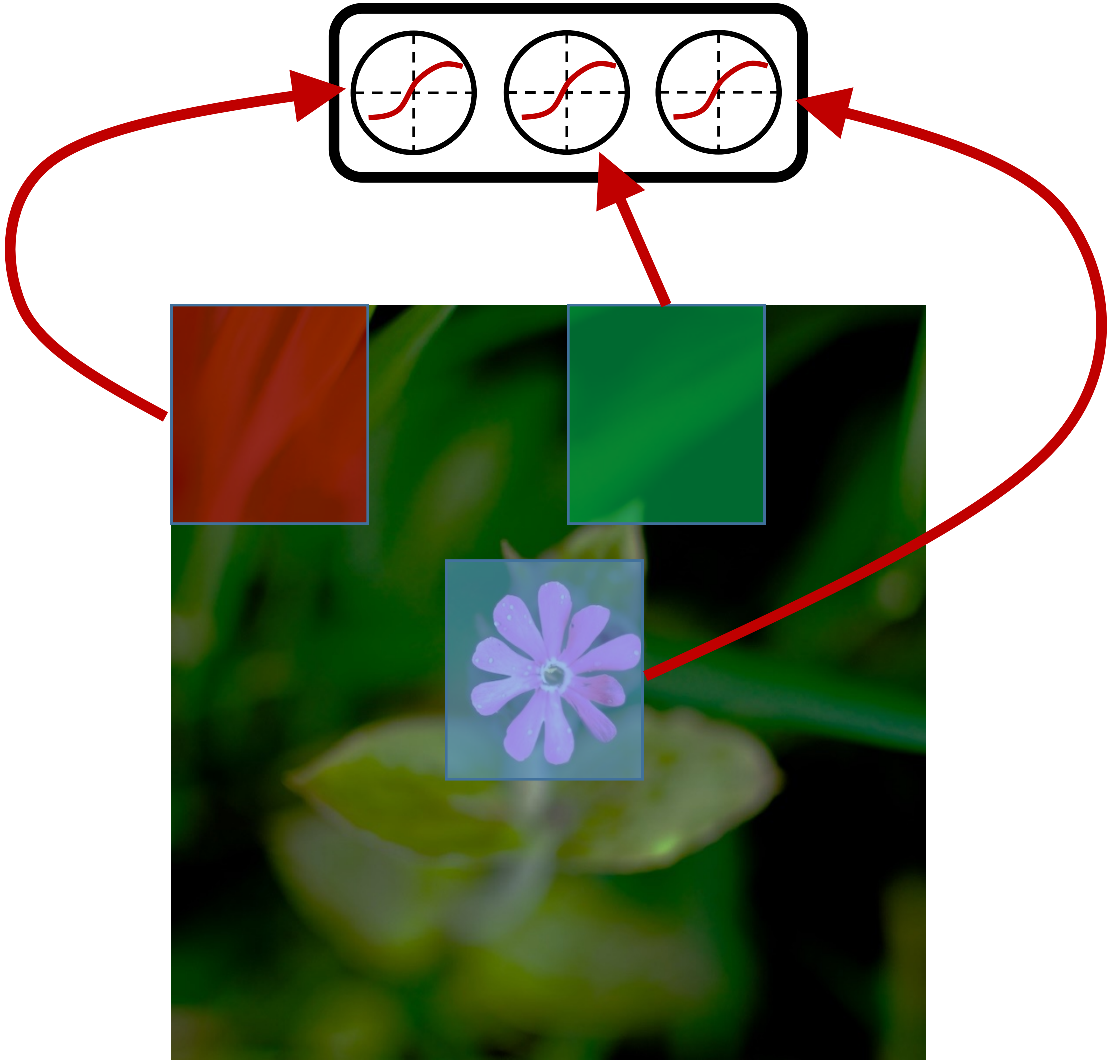
Images:

1. Local Structure
2. Reusable Composition
3. Hierarchical

Must exploit each of them

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

CNN: Local Connectivity



CNN: Local Connectivity

- Each hidden unit is connected only to a sub-region (patch) of the input image
- It is connected to all channels
  - 1 if greyscale image
  - 3 (R, G, B) for color image

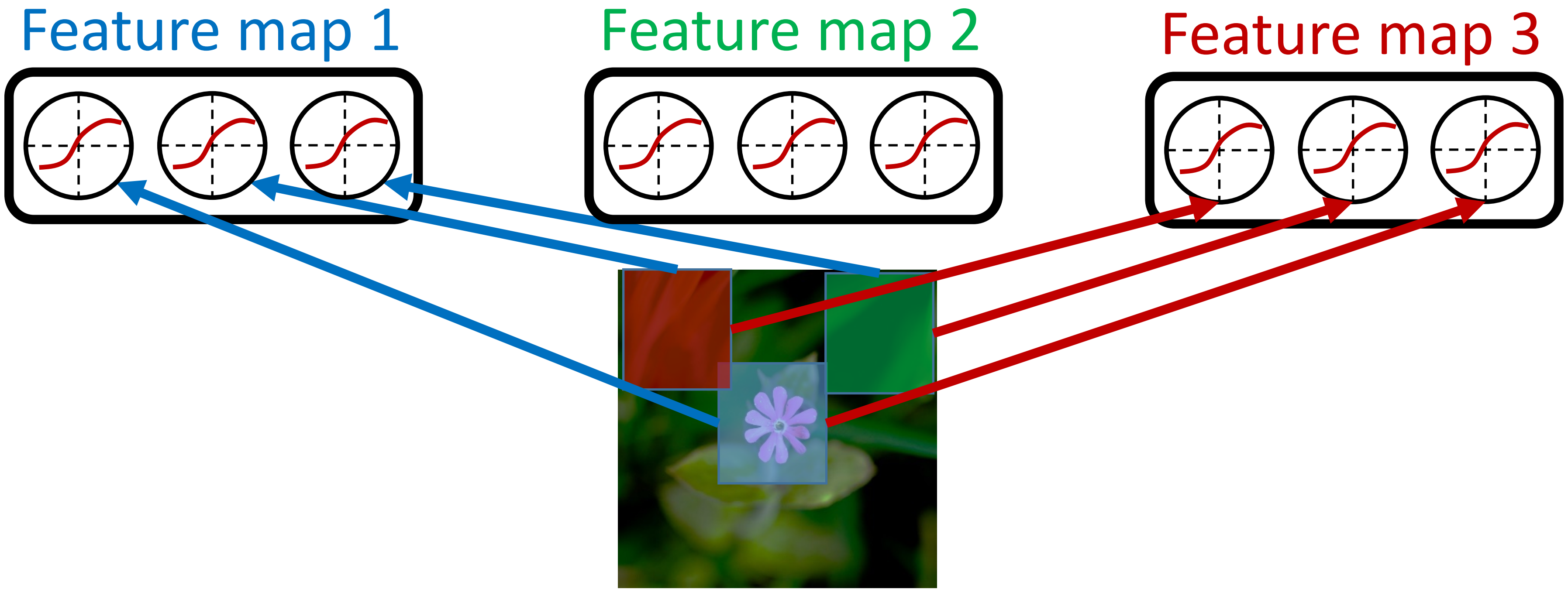
Solves the following problems:

- Fully connected hidden layer would have an unmanageable number of parameters
- Computing the linear activations of the hidden units would be very expensive

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

CNN: Parameter Sharing

- Units organized into the same “feature map/template” share parameters
- In this way all neurons detect the same feature/template at different positions in the input image
- Hidden units within a feature map cover different positions in the image



CNN: Parameter Sharing

How does it help?

- Reduces even more number of parameters (compared to local connectivity)
- Will extract the same features at every position
    - features are “equi-variant”

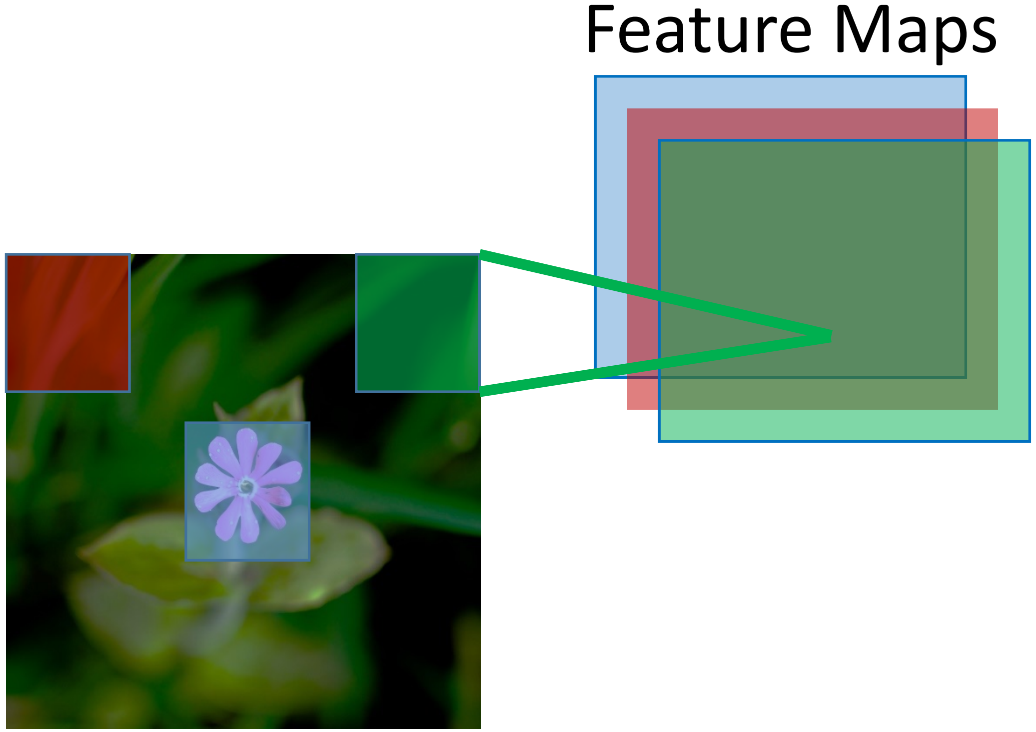
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

CNN: Parameter Sharing

- Each feature map forms a 2D grid of features
- Can be computed with a discrete convolution ($*$) of a kernel matrix k_{ij} which is the hidden weights matrix W_{ij} with its rows and columns flipped:

$$y_j = \tanh \sum_i k_{ij} x_i$$

- x_i is the i^{th} channel of input
- k_{ij} is the convolution kernel
- y_j is the hidden layer



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

CNN: Parameter Sharing

- **Stride** (typically denoted by **s** in CNNs) decides the spacing between overlapping convolutions

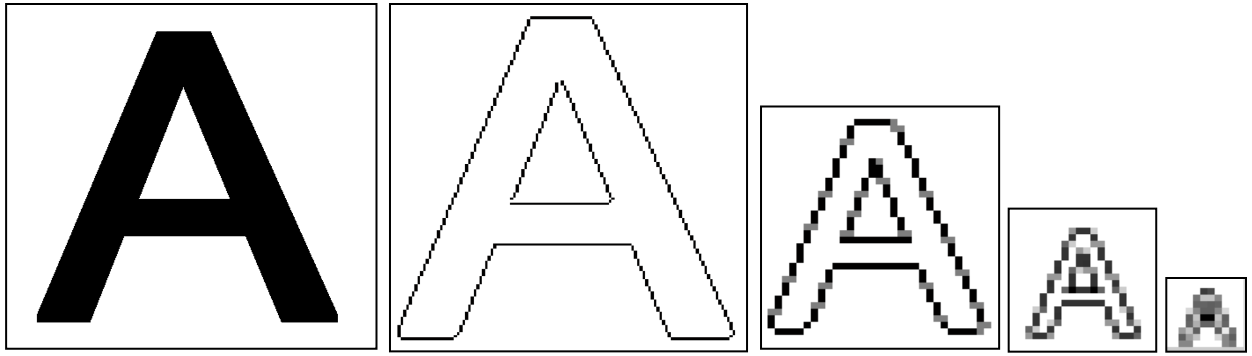
- Helps in reducing parameters further

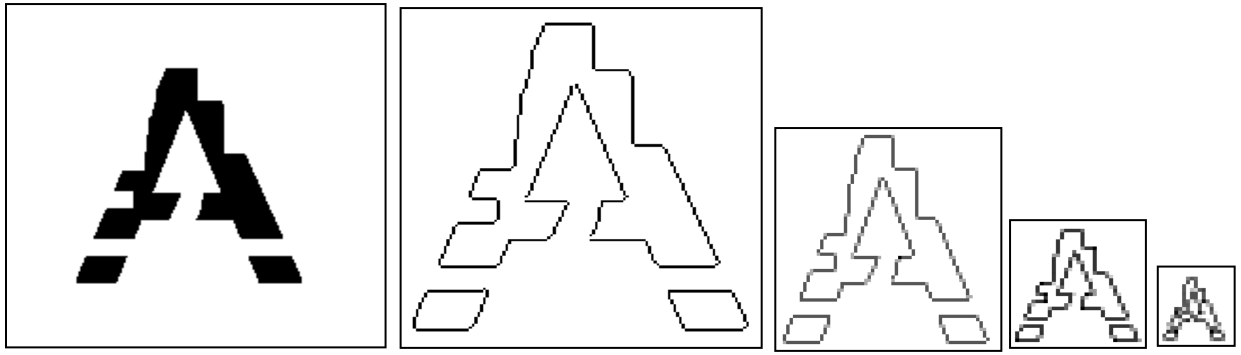
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Pooling or Subsampling Layer

- The pooling layers reduce the spatial resolution of each feature map. Helps to collate the features in a region

- By reducing the spatial resolution of the feature map, a certain degree of noise, shift and distortion invariance is also achieved.





Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Jargon

- Convolutional Neural Networks
  - also called CNNs, Conv Nets etc.
- Each hidden unit channel
  - also called **map, feature, feature type, dimension**
- Weights for each channel
  - also called **kernels**
- Input patch to a hidden unit at (x,y)
  - also called **receptive field**

Typical CNN

- Alternates convolutional and pooling layers

- Output layer is a regular, fully connected layer with softmax non-linearity
  - Output provides an estimate of the conditional probability of each class

- The network is trained by stochastic gradient descent
  - Backpropagation is used similarly as in a fully connected network
  - We have seen how to pass gradients through element-wise activation function
  - We also need to pass gradients through the convolution operation and the pooling operation

Backpropagation

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Backpropagation in General Cases

1. Decompose operations in layers of a neural network into function elements whose derivatives w.r.t. inputs are known by symbolic computation.

$$h_\theta(x) = \left(f^{(l_{max})} \circ \cdots \circ f^{(l)}_{\theta^{(l)}} \circ \cdots \circ f^{(2)}_{\theta^{(2)}} \circ f^{(1)}\right)(x)$$

where $f^{(1)} = x$ $\quad$ $f^{(l_{max})} = h_\theta$

and $\forall l: \dfrac{\partial f^{(l)}}{\partial f^{(l-1)}}$ is known

Backpropagation in General Cases

2. Backpropagate error signals corresponding to a differentiable cost function by numerical computation (Starting from cost function, plug in error signals backward).

$$\delta^{(l)} = \frac{\partial}{\partial f^{(l)}} J(\theta; x, y) = \frac{\partial J}{\partial f^{(l+1)}} \frac{\partial f^{(l+1)}}{\partial f^{(l)}} = \delta^{(l+1)} \frac{\partial f^{(l+1)}}{\partial f^{(l)}}$$

where $\frac{\partial J}{\partial f^{(l_{max})}}$ is known

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Backpropagation in General Cases

3. Use back-propagated error signals to compute gradients w.r.t. parameters only for the function elements with parameters where their derivatives w.r.t. parameters are known by symbolic computation.

$$\nabla_{\theta^{(l)}} J(\theta; x, y) = \frac{\partial}{\partial \theta^{(l)}} J(\theta; x, y) = \frac{\partial J}{\partial f^{(l)}} \frac{\partial f^{(l)}_{\theta^{(l)}}}{\partial \theta^{(l)}} = \delta^{(l)} \frac{\partial f^{(l)}_{\theta^{(l)}}}{\partial \theta^{(l)}}$$

where $\frac{\partial f^{(l)}_{\theta^{(l)}}}{\partial \theta^{(l)}}$ is known

Backpropagation in General Cases

4. Sum gradients over all example to get overall gradient.

$$\nabla_{\theta^{(l)}} J(\theta) = \sum_{i=1}^{m} \nabla_{\theta^{(l)}} J\left(\theta; x^{(i)}, y^{(i)}\right)$$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Convolution as Matrix Multiplication

$$\begin{bmatrix} 4 & 5 & 8 & 7 \\ 1 & 8 & 8 & 8 \\ 3 & 6 & 6 & 4 \\ 6 & 5 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 1 & 4 & 1 \\ 1 & 4 & 3 \\ 3 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 122 & 148 \\ 126 & 134 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 5 & 8 & 7 & 1 & 8 & 8 & 8 & 3 & 6 & 6 & 4 & 6 & 5 & 7 & 8 \end{bmatrix}^T$$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Convolution as Matrix Multiplication

4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

*

1	4	1
1	4	3
3	3	1

⇒

122	148
126	134

1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
	1	4	1	0	1	4	3	0	3	3	1	0			
				1	4	1	0	1	4	3	0	3	3	1	0
					1	4	1	0	1	4	3	0	3	3	1

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Convolution as Matrix Multiplication

1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
	1	4	1	0	1	4	3	0	3	3	1	0			
				1	4	1	0	1	4	3	0	3	3	1	0
					1	4	1	0	1	4	3	0	3	3	1

*

4
5
8
7
1
8
8
8
3
6
6
4
6
5
7
8

=

122
148
126
134

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Backpropagation in Convolution Layer

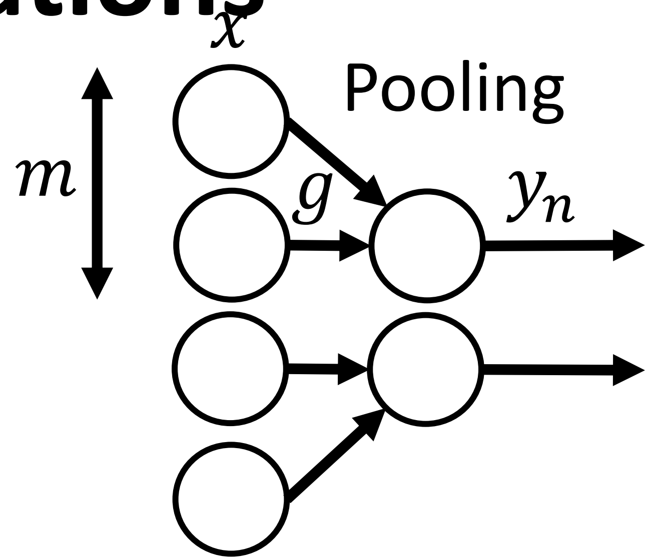
Forward propagation $x * w = y$

Backward propagation $\frac{\partial J}{\partial x} = flip(w) * \frac{\partial J}{\partial y}$

Gradient computation $x * \frac{\partial J}{\partial y} = \frac{\partial J}{\partial w}$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Pooling Layer: Subsampling Equations



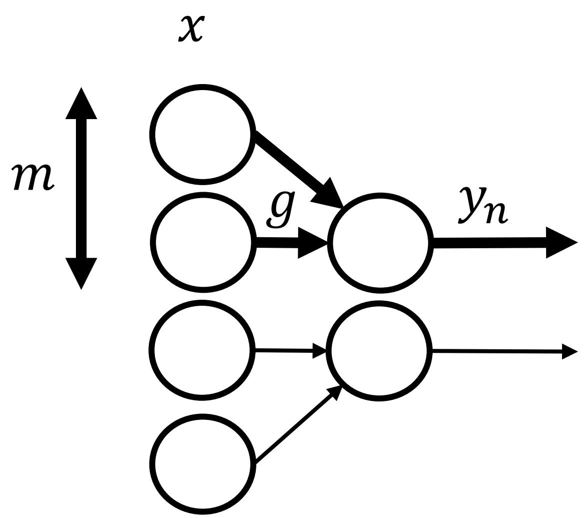
Mean Pooling $g(x) = \frac{\sum_{k=1}^{m} x_k}{m}$ $\frac{\partial g}{\partial x} = \frac{1}{m}$

Max Pooling $\mathrm{g(x)} = \max(x)$ $\frac{\partial g}{\partial x_i} = \begin{cases} 1 & if\ x_i = \max(x) \\ 0 & otherwise \end{cases}$

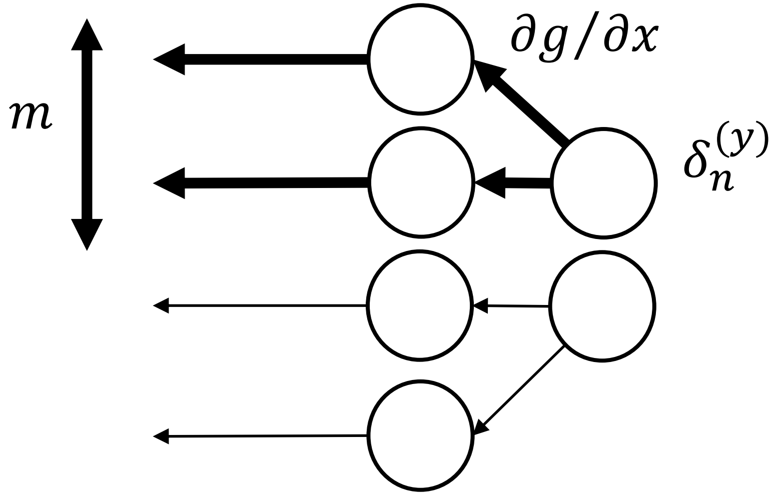
L^p Pooling $g(x) = \|x\|_p = \left(\sum_{k=1}^{m} |x_k|^P\right)^{\frac{1}{P}}$ $\frac{\partial g}{\partial x_i} = \left(\sum_{k=1}^{m} |x_k|^P\right)^{\frac{1}{P}-1} . |x_i|^{p-1}$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Backpropagation in the Pooling Layer



Forward propagation (subsampling)



Backward propagation (upsampling)