CSC321
Introduction to Neural Networks and Machine Learning

Lecture 1: What are neural networks?

Geoffrey Hinton

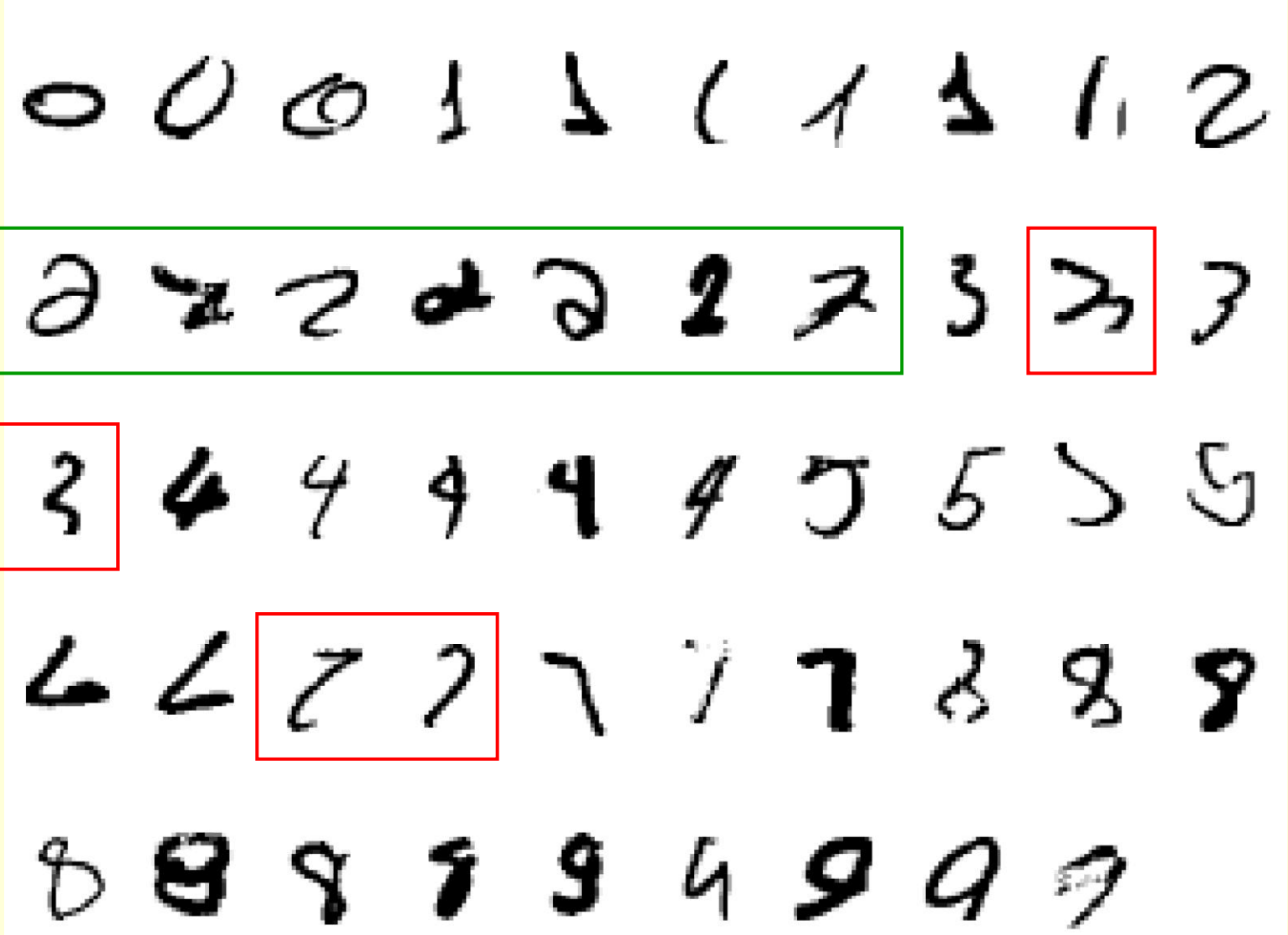
Lecture slides are available at

www.cs.toronto.edu/~hinton

What is Machine Learning?

- It is very hard to write programs that solve problems like recognizing a face.
  - We don't know what program to write because we don't know how its done.
  - Even if we had a good idea about how to do it, the program might be horrendously complicated.
- Instead of writing a program by hand, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
  - The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
  - If we do it right, the program works for new cases as well as the ones we trained it on.

It is very hard to say what makes a 2



Some examples of tasks that are best solved by using a learning algorithm

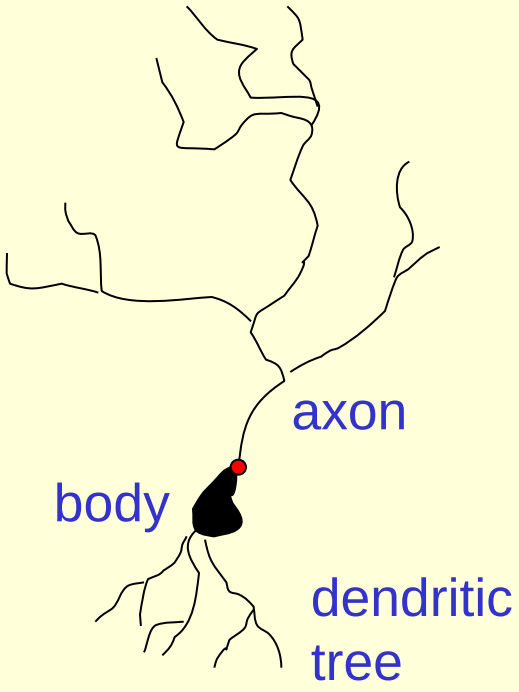
- Recognizing patterns:
  - Facial identities or facial expressions
  - Handwritten or spoken words
- Recognizing anomalies:
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant
- Prediction:
  - Future stock prices
  - Future currency exchange rates

The goals of neural computation

- To understand how the brain actually works
  - Its very big and very complicated and made of yukky stuff that dies when you poke it around
- To understand a new style of computation
  - Inspired by neurons and their adaptive connections
  - Very different style from sequential computation
    - should be good for things that brains are good at (e.g. vision)
    - Should be bad for things that brains are bad at (e.g. 23 x 71)
- To solve practical problems by developing novel learning algorithms
  - Learning algorithms can be very useful even if they have nothing to do with how the brain works

A typical cortical neuron

- Gross physical structure:
  - There is one axon that branches
  - There is a dendritic tree that collects input from other neurons
- Axons typically contact dendritic trees at synapses
  - A spike of activity in the axon causes charge to be injected into the post-synaptic neuron
- Spike generation:
  - There is an axon hillock that generates outgoing spikes whenever enough charge has flowed in at synapses to depolarize the cell membrane

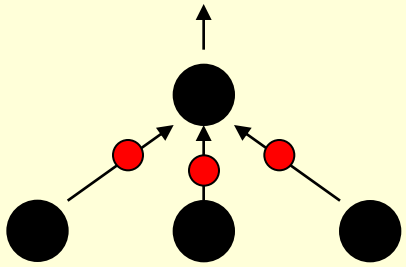


Synapses

- When a spike travels along an axon and arrives at a synapse it causes vesicles of transmitter chemical to be released
  - There are several kinds of transmitter
- The transmitter molecules diffuse across the synaptic cleft and bind to receptor molecules in the membrane of the post-synaptic neuron thus changing their shape.
  - This opens up holes that allow specific ions in or out.
- The effectiveness of the synapse can be changed
  - vary the number of vesicles of transmitter
  - vary the number of receptor molecules.
- Synapses are slow, but they have advantages over RAM
  - Very small
  - They adapt using locally available signals (but how?)

How the brain works

- Each neuron receives inputs from other neurons
  - Some neurons also connect to receptors
  - Cortical neurons use spikes to communicate
  - The timing of spikes is important
- The effect of each input line on the neuron is controlled by a synaptic weight
  - The weights can be positive or negative



- The synaptic weights adapt so that the whole network learns to perform useful computations
  - Recognizing objects, understanding language, making plans, controlling the body
- You have about 10^{11} neurons each with about 10^{3} weights
  - A huge number of weights can affect the computation in a very short time. Much better bandwidth than a computer.

Modularity and the brain

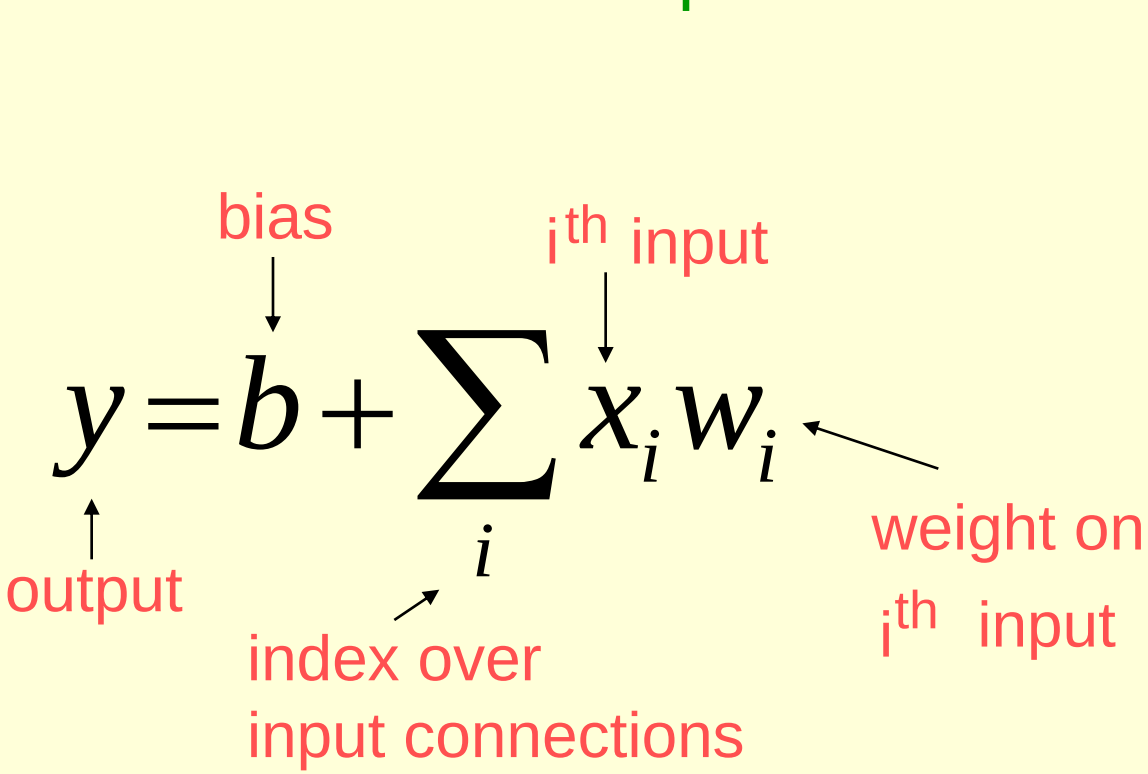
- Different bits of the cortex do different things.
  - Local damage to the brain has specific effects
  - Specific tasks increase the blood flow to specific regions.
- But cortex looks pretty much the same all over.
  - Early brain damage makes functions relocate
- Cortex is made of general purpose stuff that has the ability to turn into special purpose hardware in response to experience.
  - This gives rapid parallel computation plus flexibility.
  - Conventional computers get flexibility by having stored programs, but this requires very fast central processors that perform large computations sequentially.

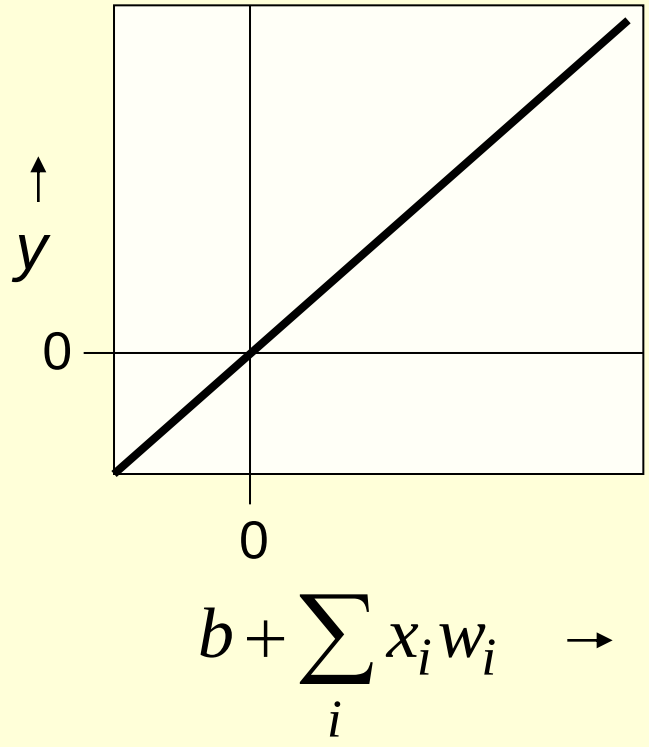
Idealized neurons

- To model things we have to idealize them (e.g. atoms)
  - Idealization removes complicated details that are not essential for understanding the main principles
  - Allows us to apply mathematics and to make analogies to other, familiar systems.
  - Once we understand the basic principles, its easy to add complexity to make the model more faithful
- It is often worth understanding models that are known to be wrong (but we mustn't forget that they are wrong!)
  - E.g. neurons that communicate real values rather than discrete spikes of activity.

Linear neurons

- These are simple but computationally limited
  - If we can make them learn we may get insight into more complicated neurons



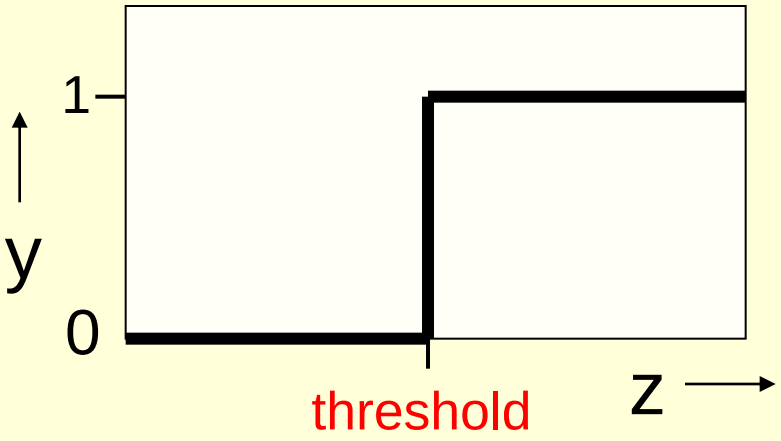


Binary threshold neurons

- McCulloch-Pitts (1943): influenced Von Neumann!
  - First compute a weighted sum of the inputs from other neurons
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
  - McCulloch & Pitts: each spike is like the truth value of a proposition and each neuron combines truth values to compute the truth value of another proposition.

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geq \theta \\ 0 \text{ otherwise} \end{cases}$$



Linear threshold neurons

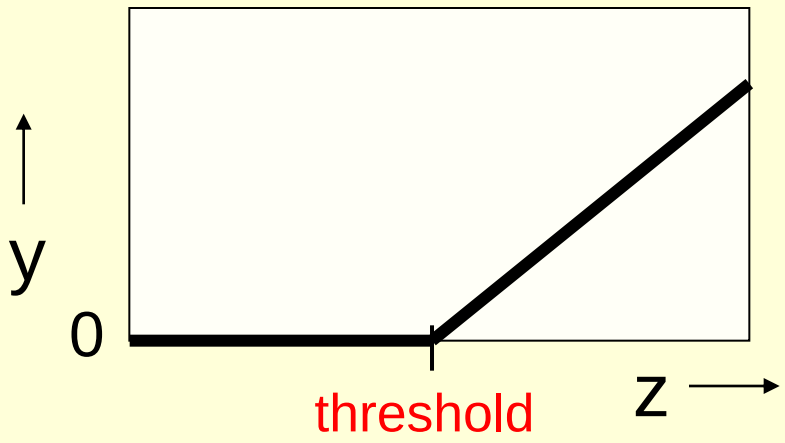
These have a confusing name.

They compute a linear weighted sum of their inputs

The output is a non-linear function of the total input

$$z_j = b_j + \sum_i x_i w_{ij}$$

$$y_j = \begin{cases} z_j \text{ if } z_j \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

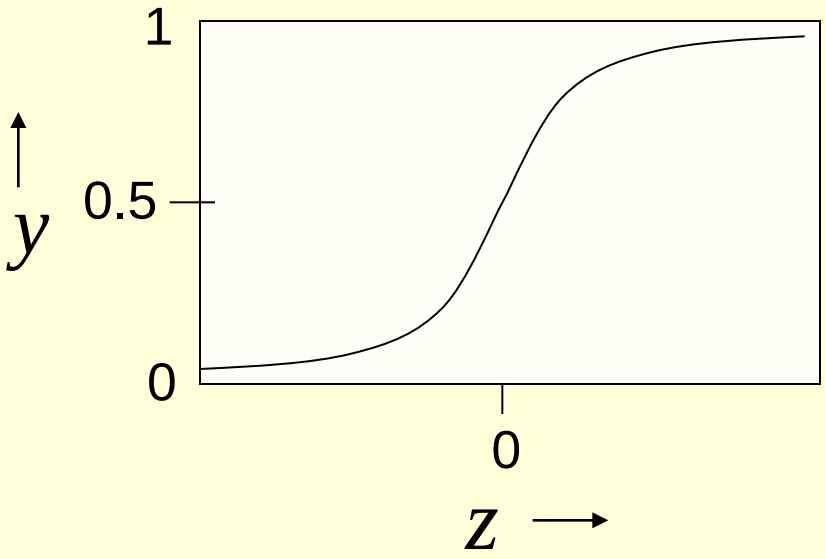


Sigmoid neurons

- These give a real-valued output that is a smooth and bounded function of their total input.
  - Typically they use the logistic function
  - They have nice derivatives which make learning easy (see lecture 3).
- If we treat y as a probability of producing a spike, we get stochastic binary neurons.

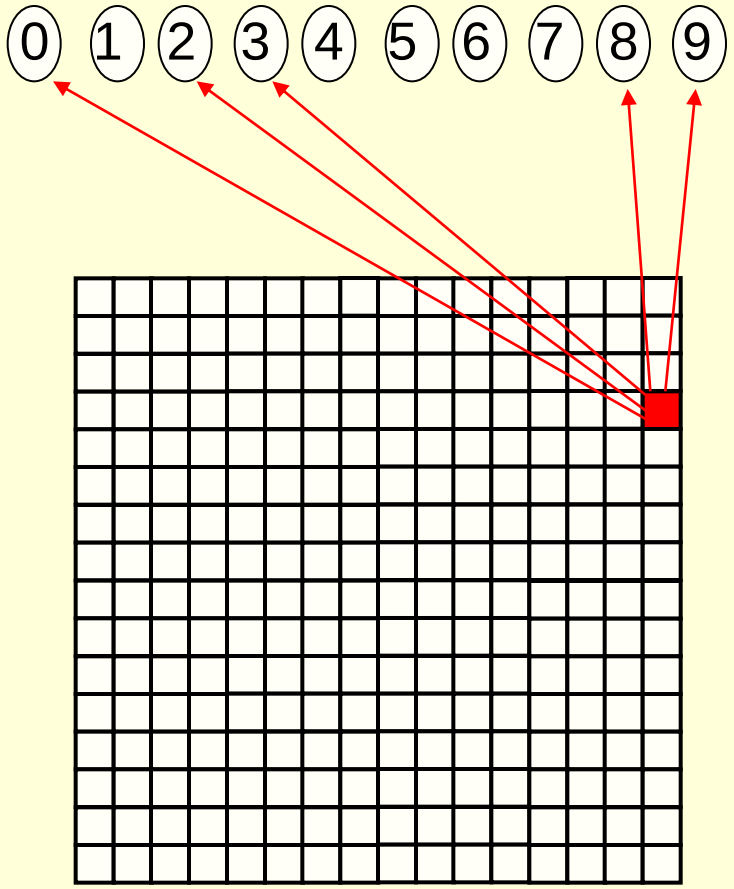
$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$

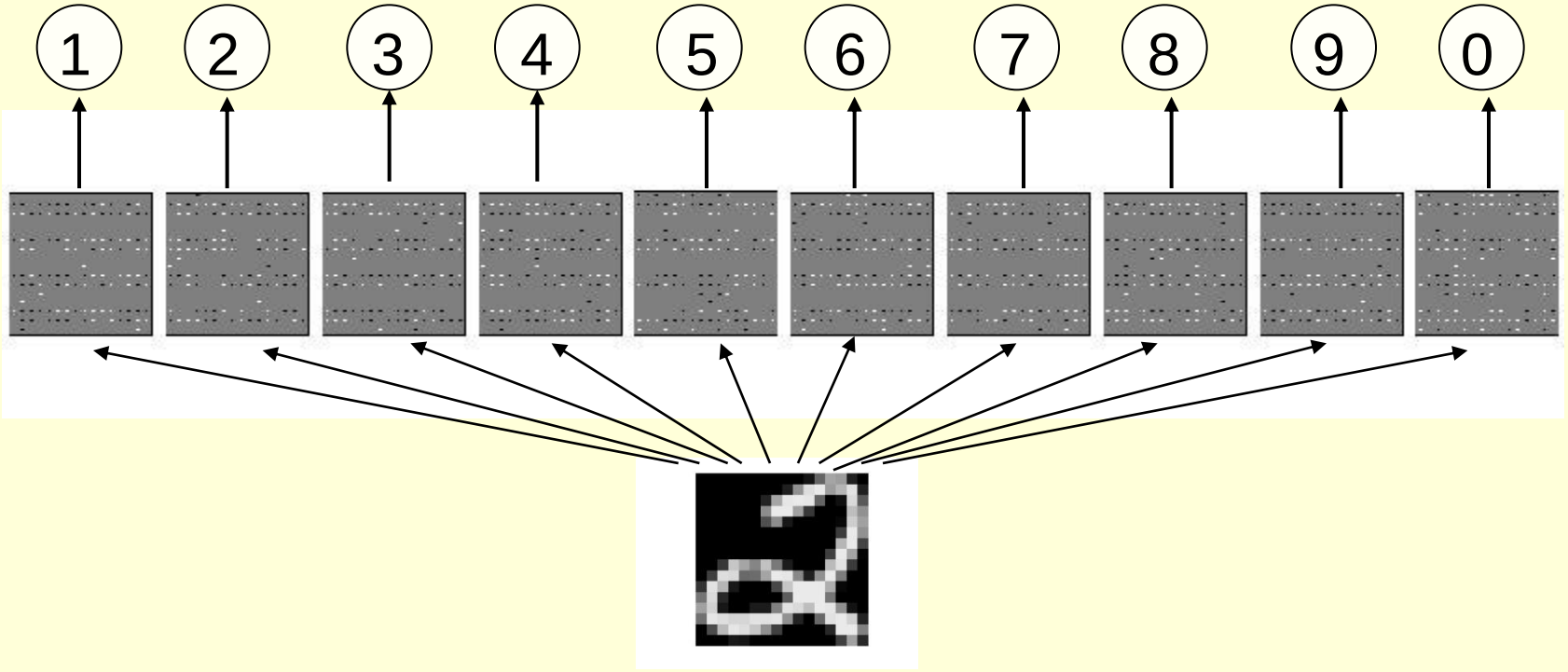


A very simple way to recognize handwritten shapes

- Consider a neural network with two layers of neurons.
  - neurons in the top layer represent known shapes.
  - neurons in the bottom layer represent pixel intensities.
- A pixel gets to vote if it has ink on it.
  - Each inked pixel can vote for several different shapes.
- The shape that gets the most votes wins.



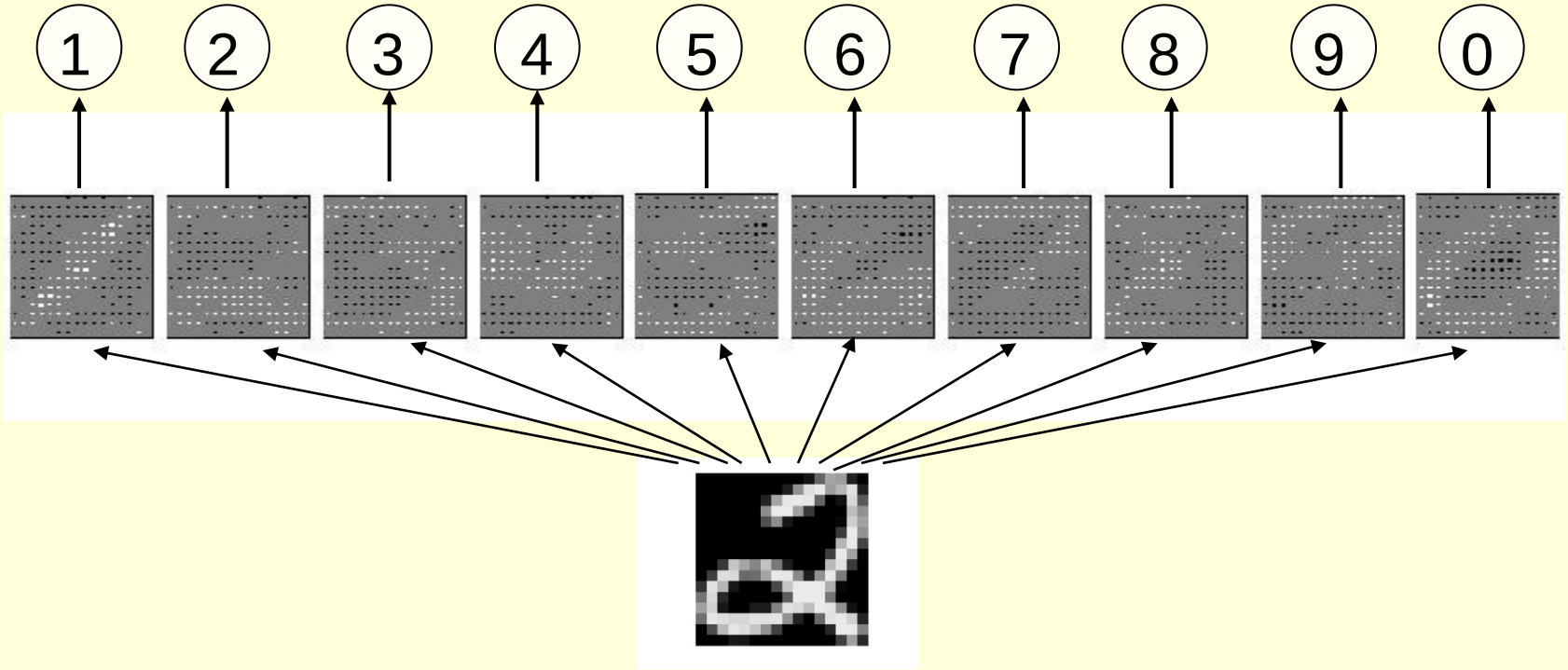
How to learn the weights



The image

Show the network an image and increment the weights from active pixels to the correct class.

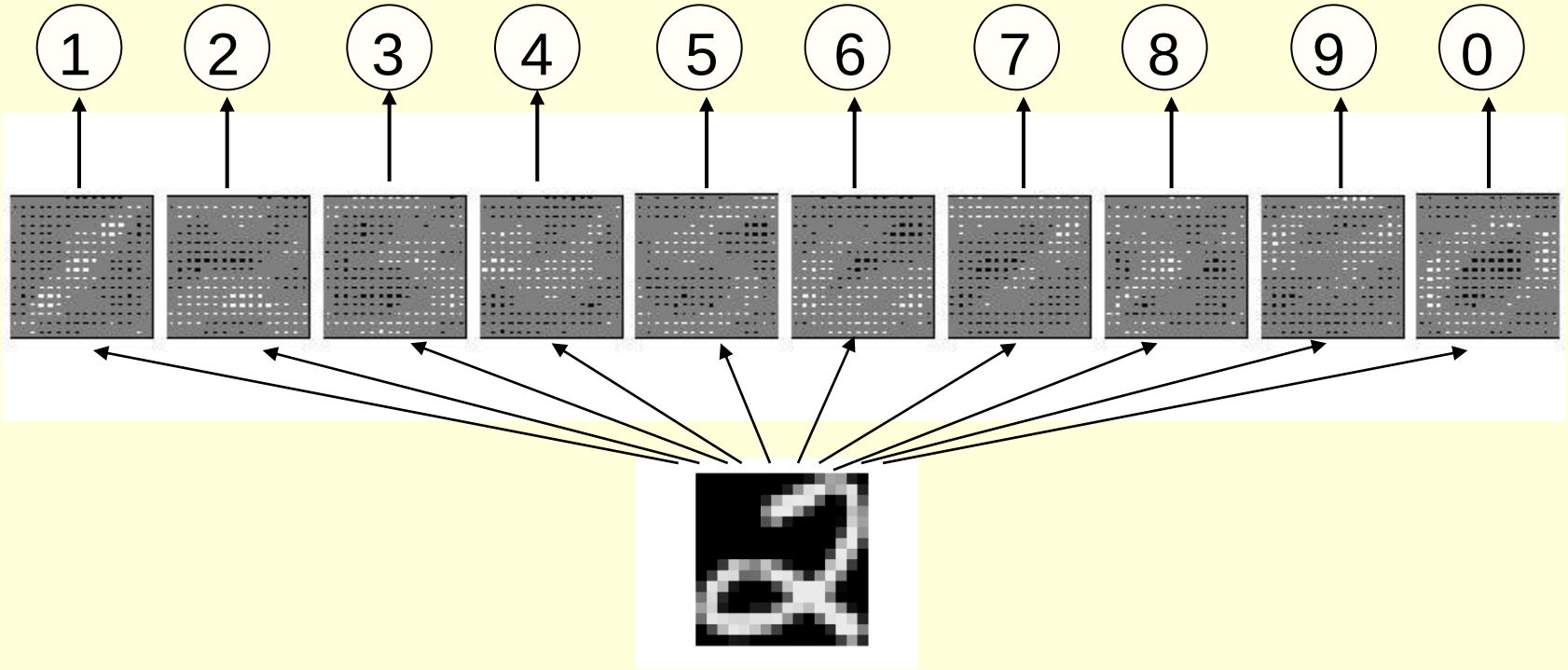
Then decrement the weights from active pixels to whatever class the network guesses.



The image

Show the network an image and increment the weights from active pixels to the correct class.

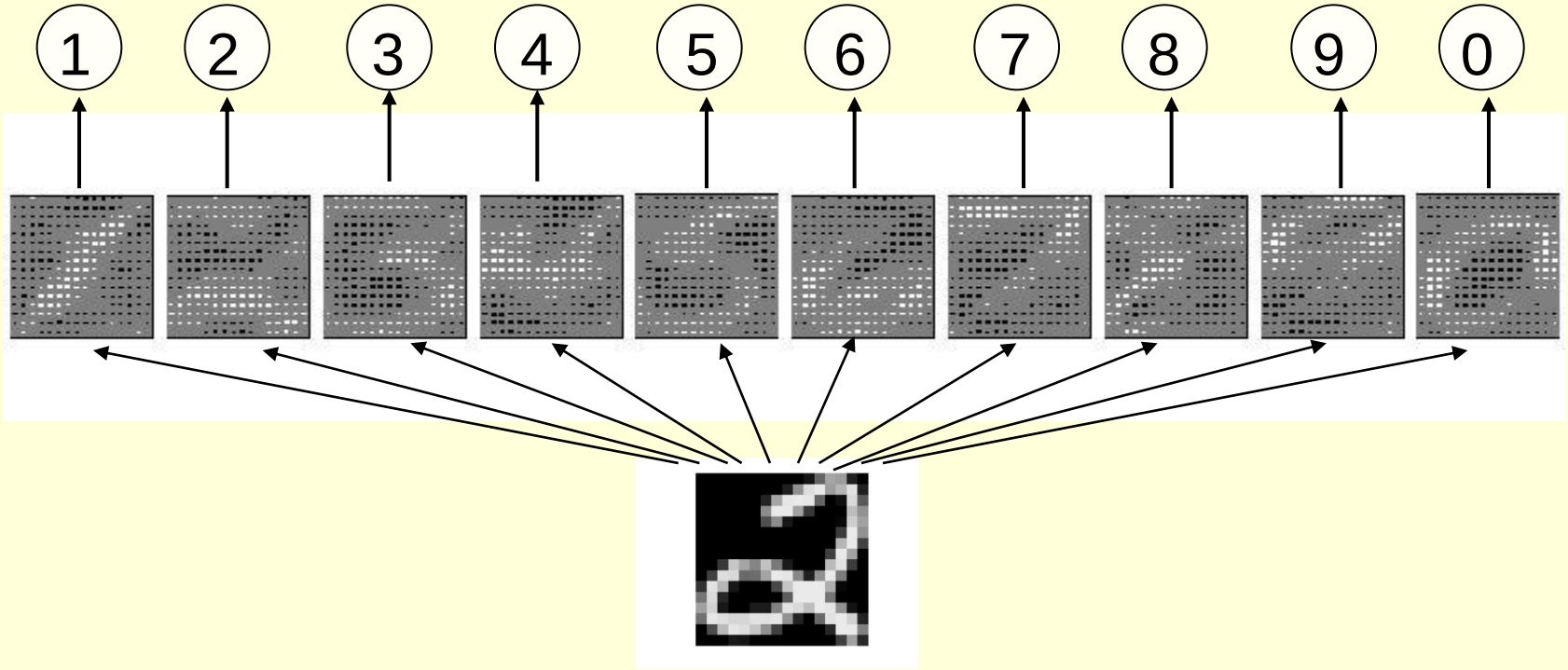
Then decrement the weights from active pixels to whatever class the network guesses.



The image

Show the network an image and increment the weights from active pixels to the correct class.

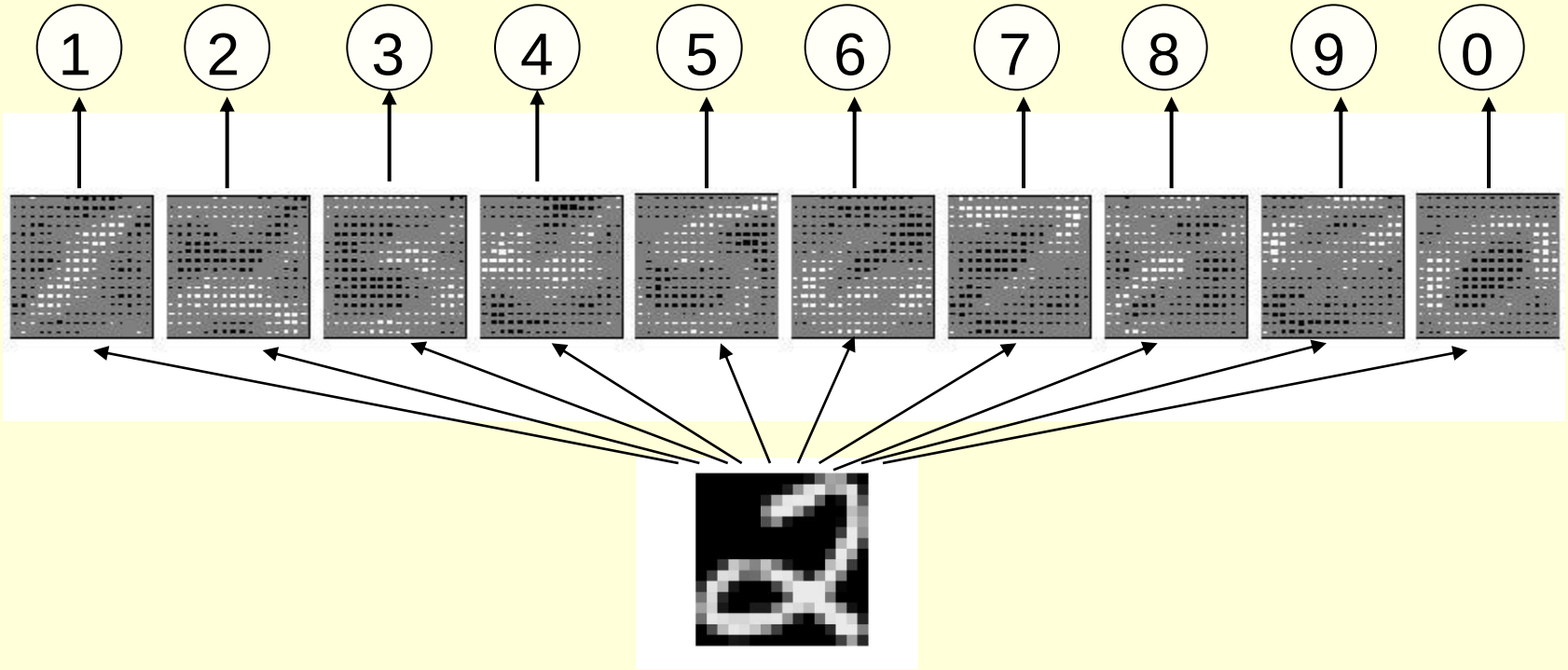
Then decrement the weights from active pixels to whatever class the network guesses.



The image

Show the network an image and increment the weights from active pixels to the correct class.

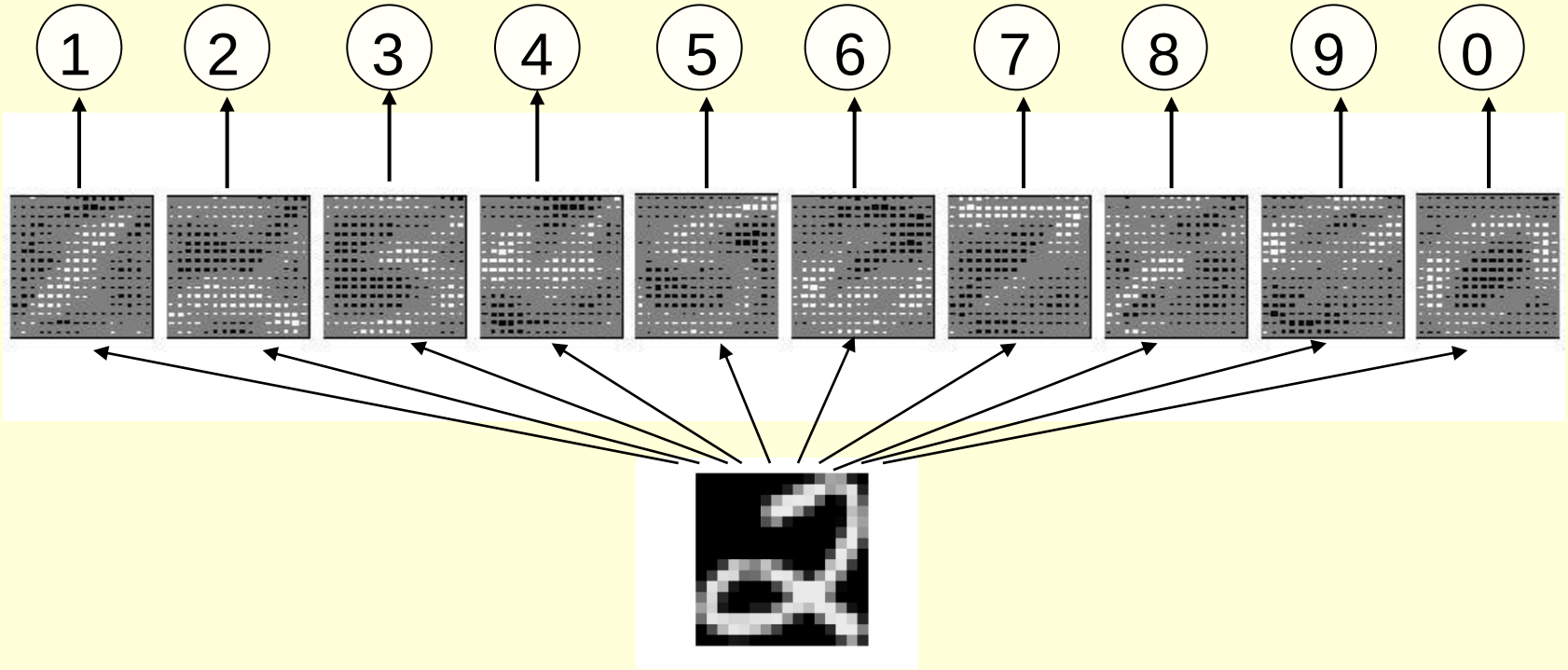
Then decrement the weights from active pixels to whatever class the network guesses.



The image

Show the network an image and increment the weights from active pixels to the correct class.

Then decrement the weights from active pixels to whatever class the network guesses.

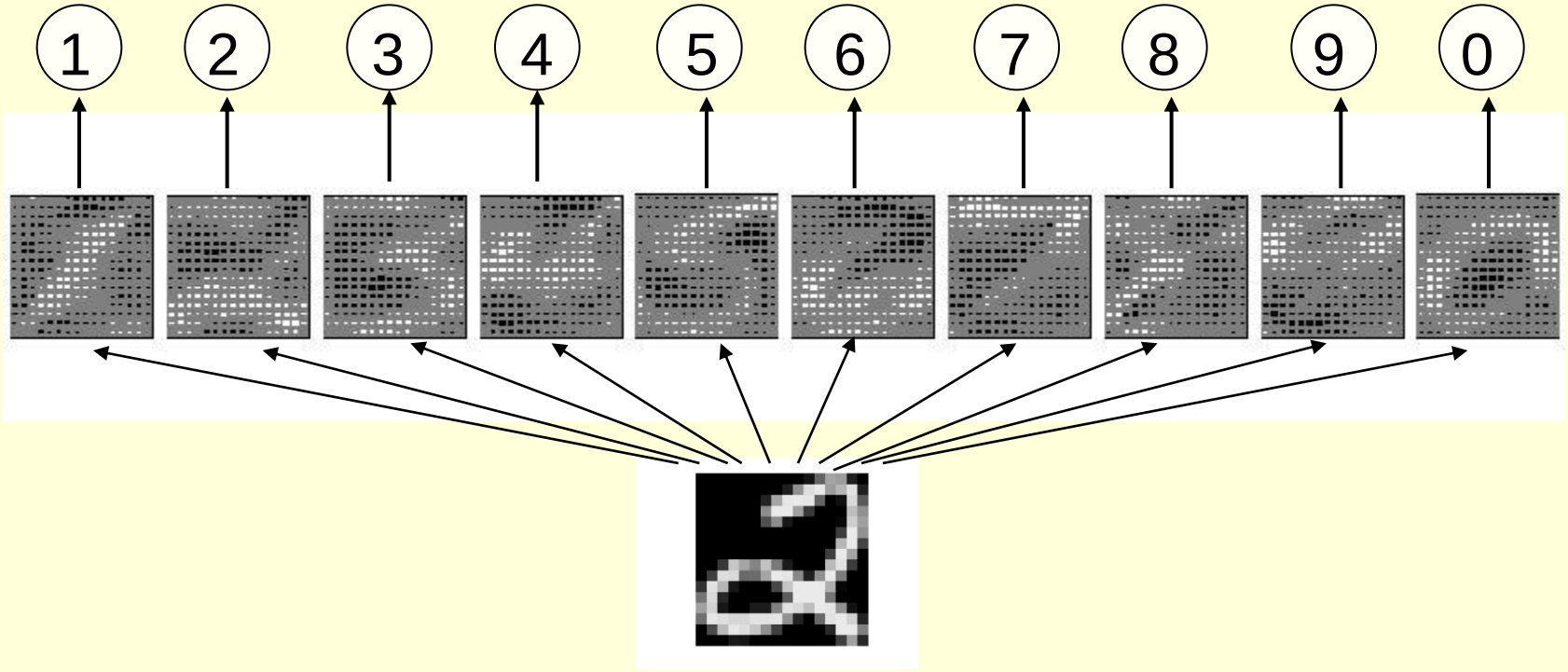


The image

Show the network an image and increment the weights from active pixels to the correct class.

Then decrement the weights from active pixels to whatever class the network guesses.

The learned weights



The image

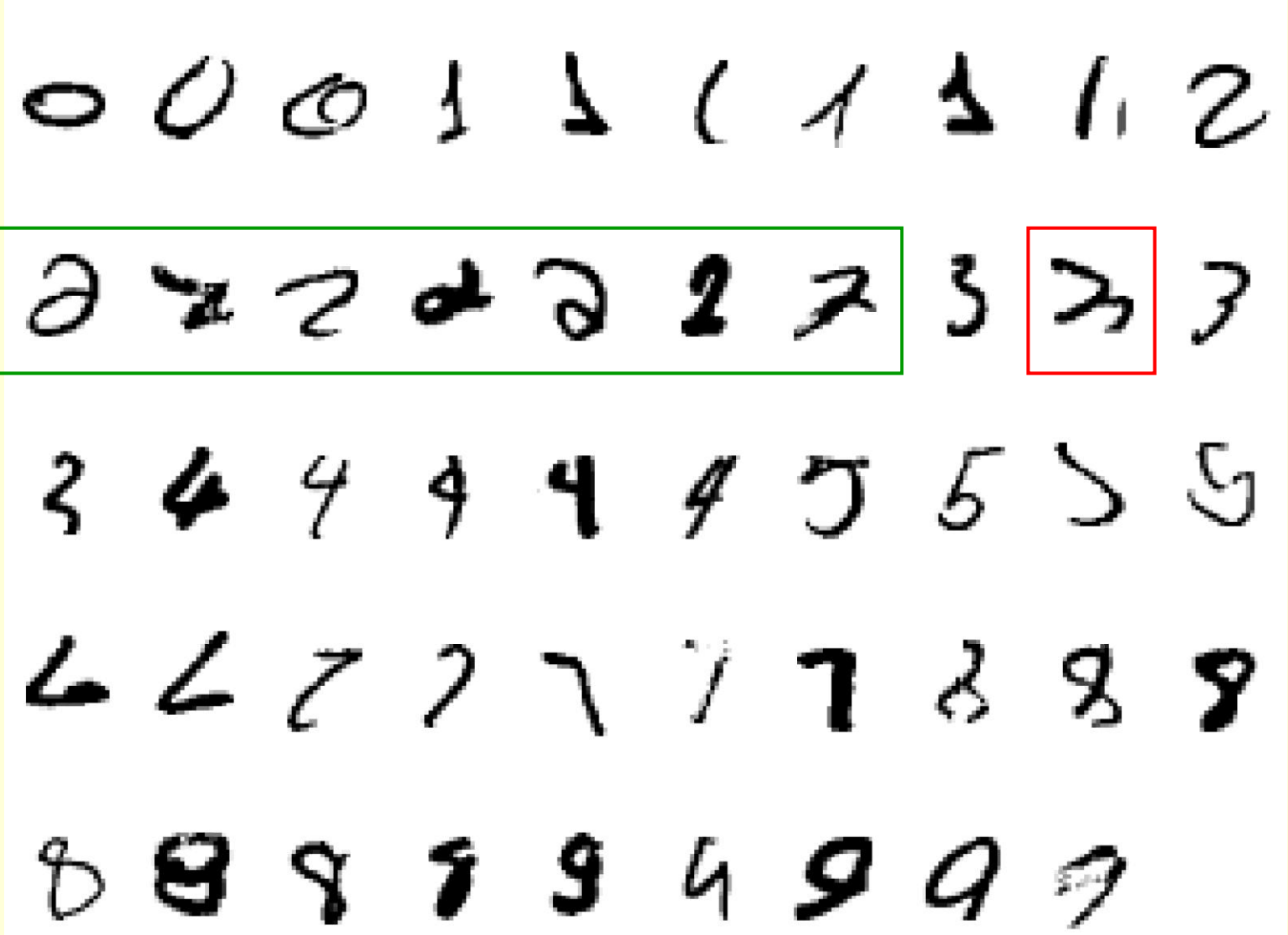
The precise details of the learning algorithm will be explained in future lectures.

Why the simple system does not work

- A two layer network with a single winner in the top layer is equivalent to having a rigid template for each shape.
  - The winner is the template that has the biggest overlap with the ink.

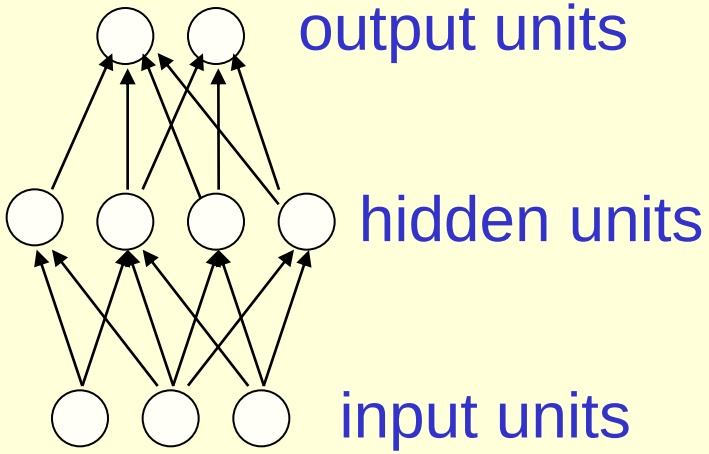
- The ways in which shapes vary are much too complicated to be captured by simple template matches of whole shapes.
  - To capture all the allowable variations of a shape we need to learn the features that it is composed of.

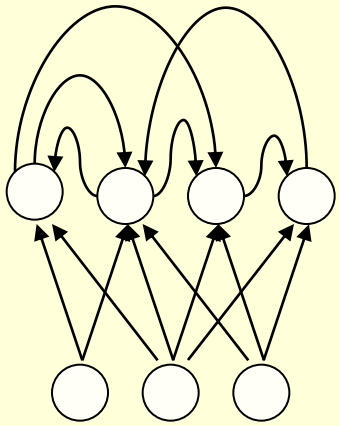
Examples of handwritten digits that need to be recognized correctly the first time they are seen



Types of connectivity

- Feedforward networks
  - These compute a series of transformations
  - Typically, the first layer is the input and the last layer is the output.
- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.





Types of learning task

- Supervised learning
  - Learn to predict output when given input vector
    - Who provides the correct answer?
- Reinforcement learning
  - Learn action to maximize payoff
    - Not much information in a payoff signal
    - Payoff is often delayed
- Unsupervised learning
  - Create an internal representation of the input e.g. form clusters; extract features
    - How do we know if a representation is good?