INFONET Seminar Application Group 2014/08/14

Neural Networks and Their Applications

Chris M. Bishop

Review of Science instruments, 1993

Presenter Pavel Ni

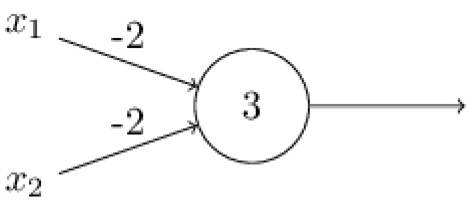
Gwangju Institute of Science and Technology

Contents

Introduction to Deep Learning

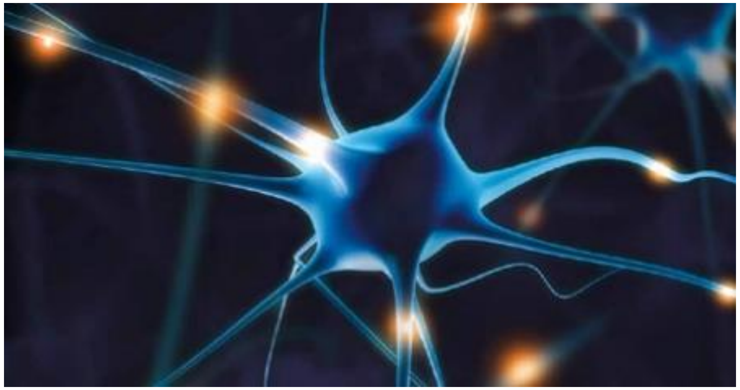
What is an Artificial Neural Network?

- Computational model inspired by a biological central nervous system(brain)
- Purpose of ANN – use its capabilities to perform a machine learning and pattern recognition tasks like humans.
- ANN consist of connected neurons
- Perceptron is a type of an artificial neuron



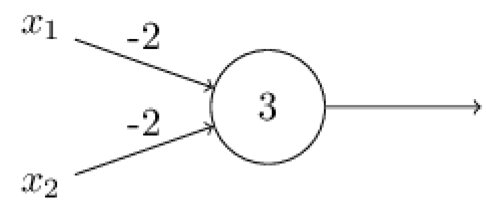
Pic. 1. Schematics of a perceptron


Pic. 2. Human brain and active central nervous system


Pic. 3. Active neuron

Perceptron and Sigmoid neuron (1)

- A perceptron takes several binary *inputs* x_1, x_2 and produces a single binary *output*
- *Weights* – real numbers expressing importance of respective inputs to the output
- *Bias* – threshold value

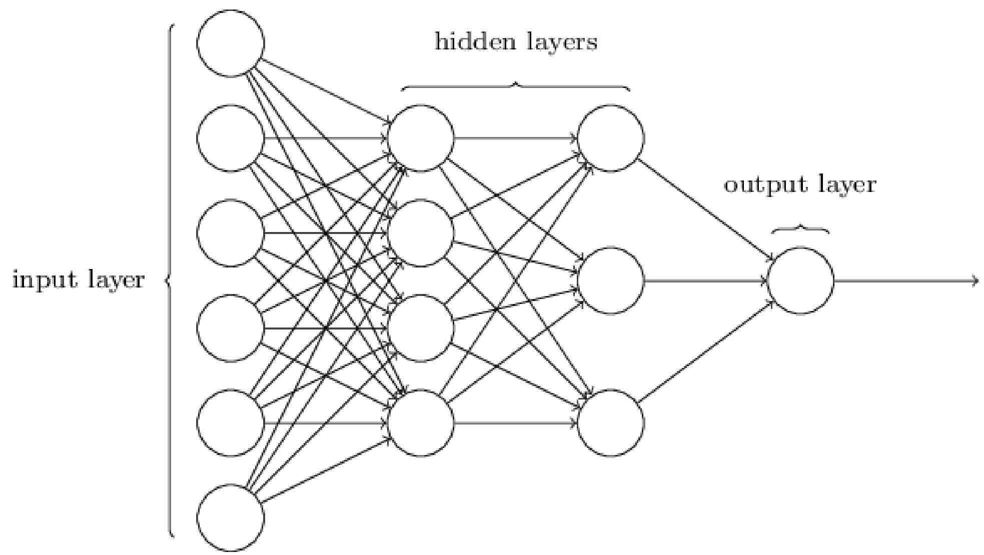


Pic. 4. Schematics of a perceptron

$$\text{output} = \begin{cases} 0 \text{ if } \sum_j w_j x_j \le \text{ threshold} \\ 1 \text{ if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

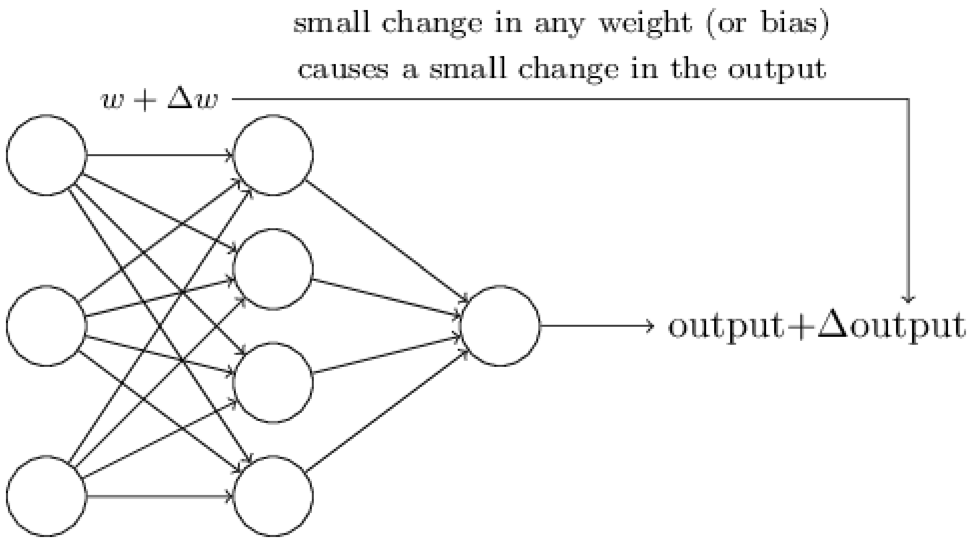
$$w \cdot x \equiv \sum_j w_j x_j$$

where w and x are vectors whose components are the weights and input



Pic. 5. Architecture of neural networks with 2 hidden layers

Perceptron and Sigmoid neuron (2)



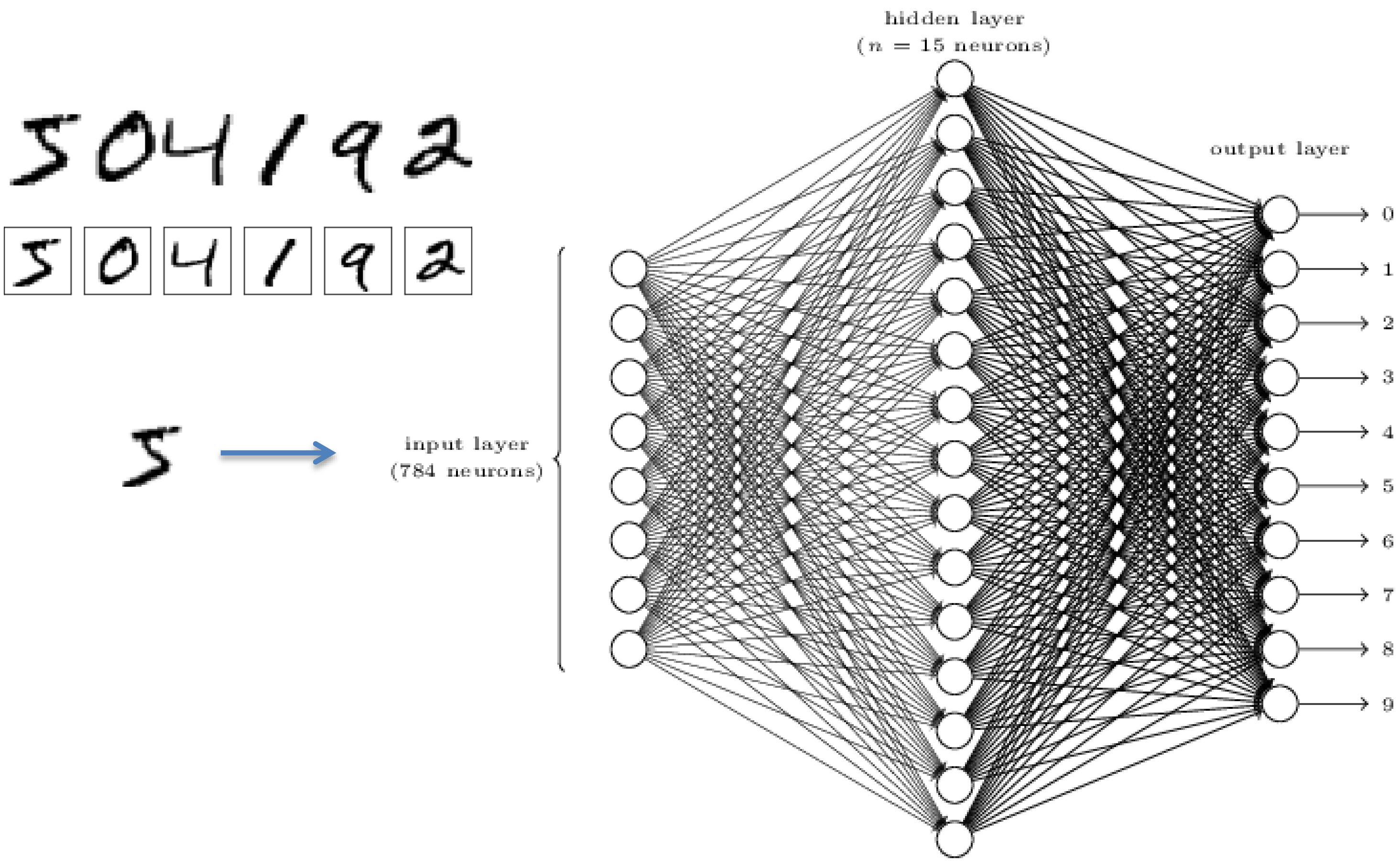
Pic. 6. Schematics of a perceptron

Sigmoid neurons are similar to perceptron's, but modified so that small changes in their weights and bias cause only a small change in their output.

$$\sigma(z) \equiv \sigma(w \cdot x + b) \equiv \frac{1}{1+e^{-z}} \equiv \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

sigmoid function

Example: hand written digit recognition



Pic. 7. hand written digit recognition

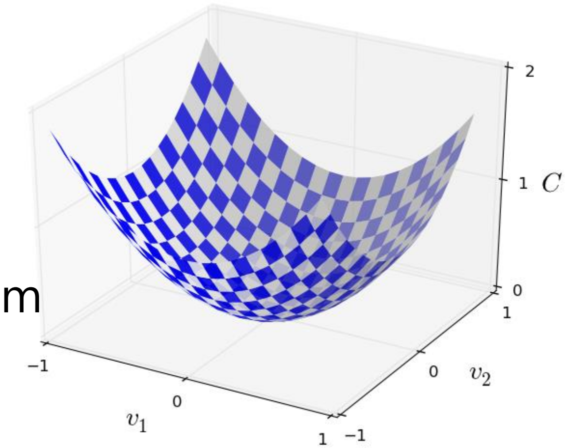
- We denote training input x as a $28 \times 28 = 784$ dimensional vector (gray scale pixels).
- Corresponding desired output by $y = y(x)$

Learning with gradient descent (1)

$$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x)-a\|^2$$

- where w denotes all weights in the network, b all the biases, n is the total number of training inputs. a is the vector of outputs from the network when x is input
- C is a cost function used to quantify how well algorithm can find weights and biases (i.e. *mean squared error*)
- Training a neural network means to find set of weights and biases which makes the quadratic cost function as small as possible

Learning with gradient descent (2)



- Lets assume that C is a function of two variables

v_1 and v_2

- Our goal to find where C achieves its global minimum

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2$$

$$\Delta v \equiv (\Delta v_1, \Delta v_2)^T, \qquad \nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T.$$

$$\Delta C \approx \nabla C \cdot \Delta v \longrightarrow \Delta v = -\eta \nabla C \longrightarrow \Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2$$

$$v \to v' = v - \eta \nabla C$$

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}, \qquad b_l \to b'_l - \eta \frac{\partial C}{\partial w_l}$$

Now we can replace vector of changes with weights and biases

Backpropagation algorithm (1)

- Backpropagation lets us to compute the partial derivatives ∂Cx/∂w and ∂Cx/∂b for a single training example.
- it shows how changing the weights and biases in a network changes the cost function (overall behavior of the network).

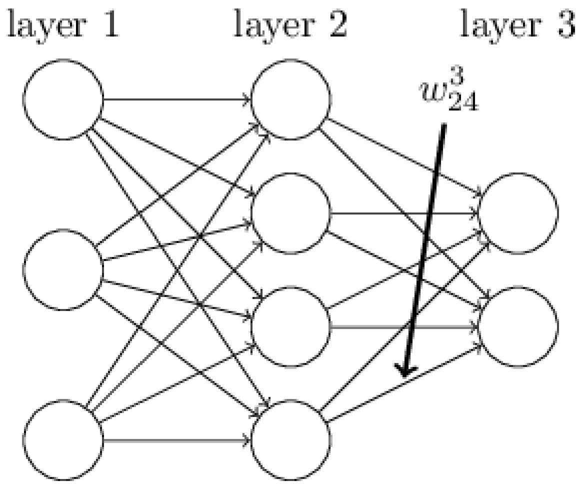
$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \qquad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \qquad \text{(BP3)}$$

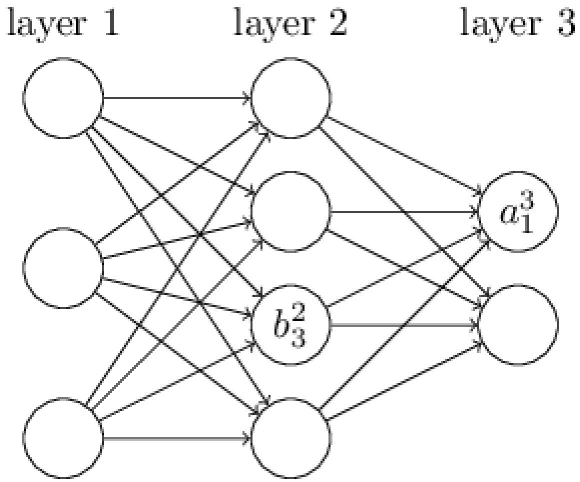
$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \qquad \text{(BP4)}$$

Backpropagation algorithm (2)



Pic. 8. Schematics of a perceptron

Notations:
w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer



Pic. 9. Schematics of a perceptron

Notations:
we use b^l_j for the bias of the j^{th} neuron in the l^{th} layer.
And we use a^l_j for the activation of the j^{th} neuron in the l^{th} layer.

Backpropagation algorithm (3)

The activation a_j^l of the j^{th} neuron in the l^{th} layer is related to the activation in the $(l-1)^{th}$ layer by the equation:

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right),$$

Sum of the all neurons in the $(l-1)^{th}$ layer

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

In the matrix form

$$z^l \equiv w^l a^{l-1} + b^l$$

weighted input

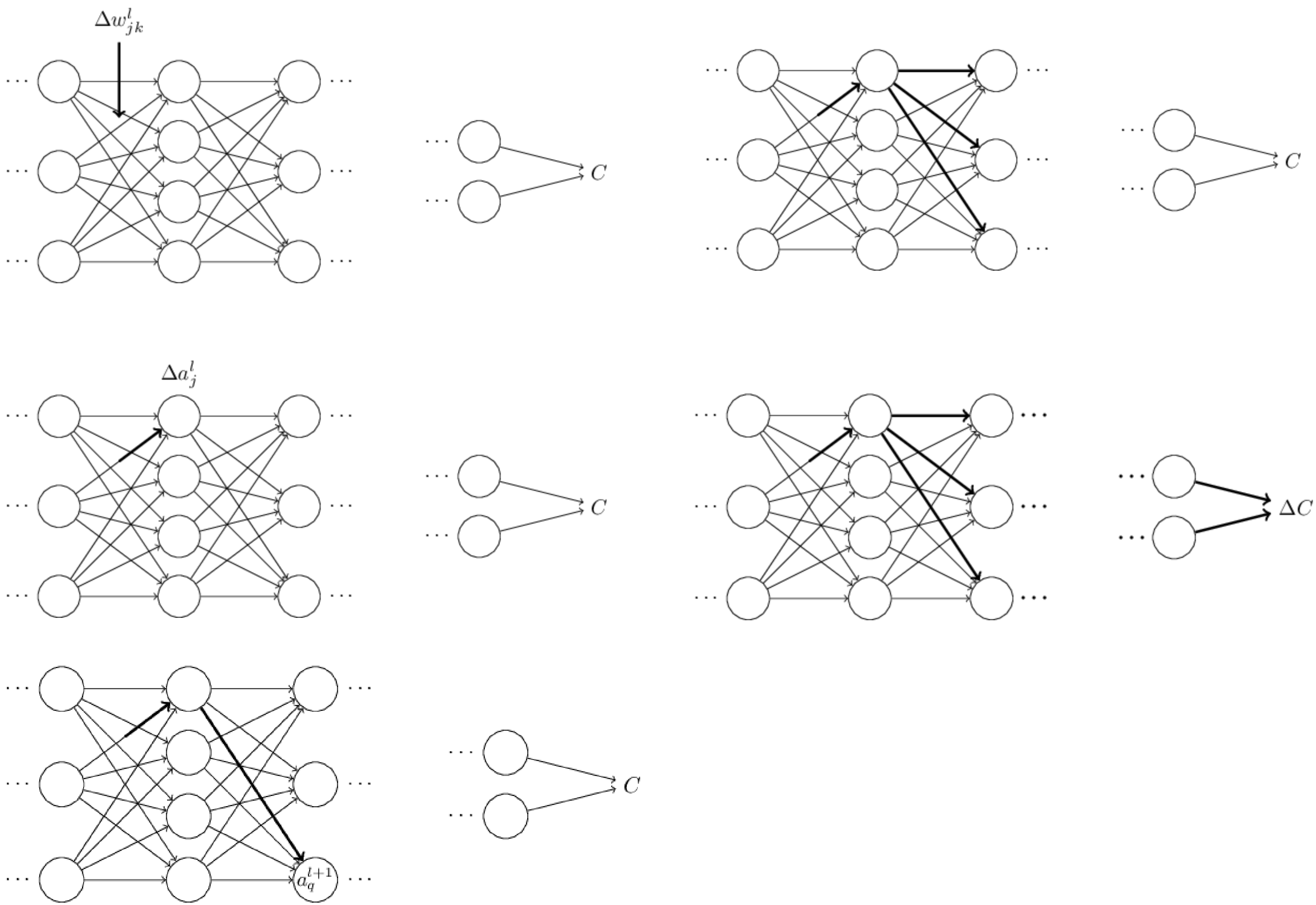
$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

- Error in the j^{th} neuron in the l^{th} layer

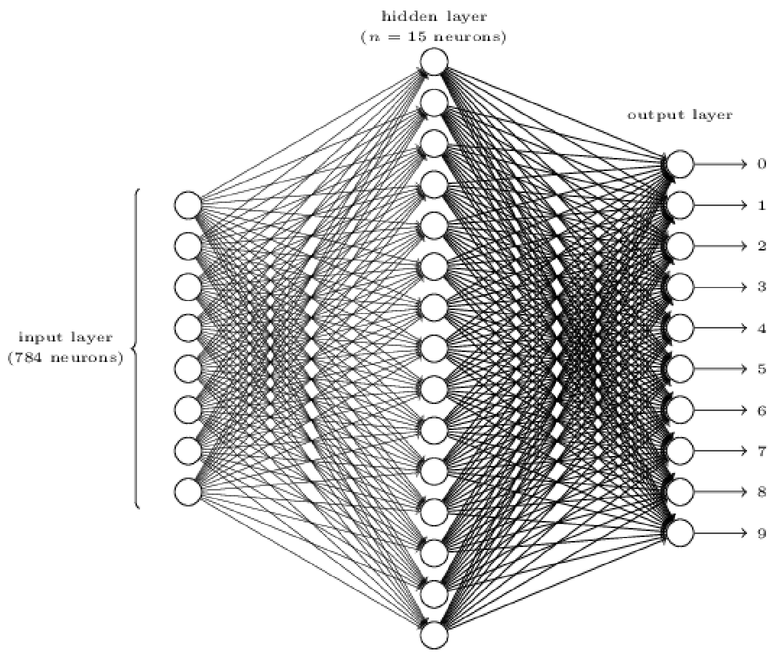
Backpropagation algorithm (4)

1. **Input** x**:** Set the corresponding activation a^1 for the input layer.
2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
3. **Output error** δ^L **:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.
4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

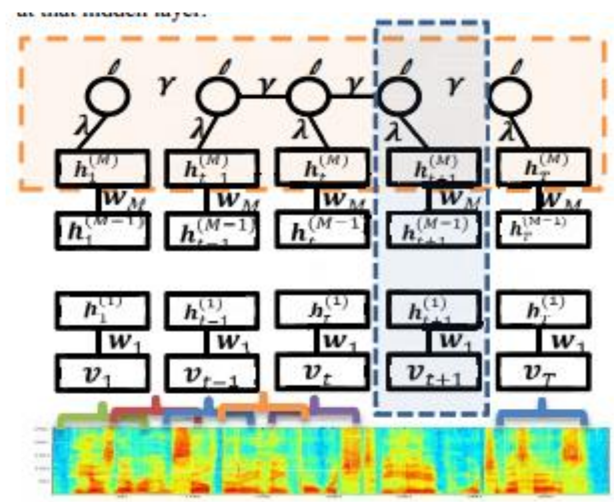
Backpropagation algorithm (5)



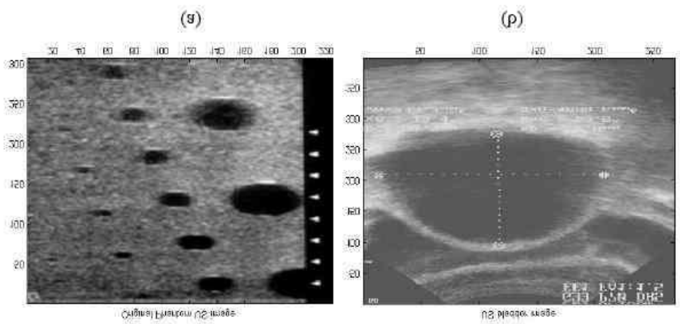
Applications



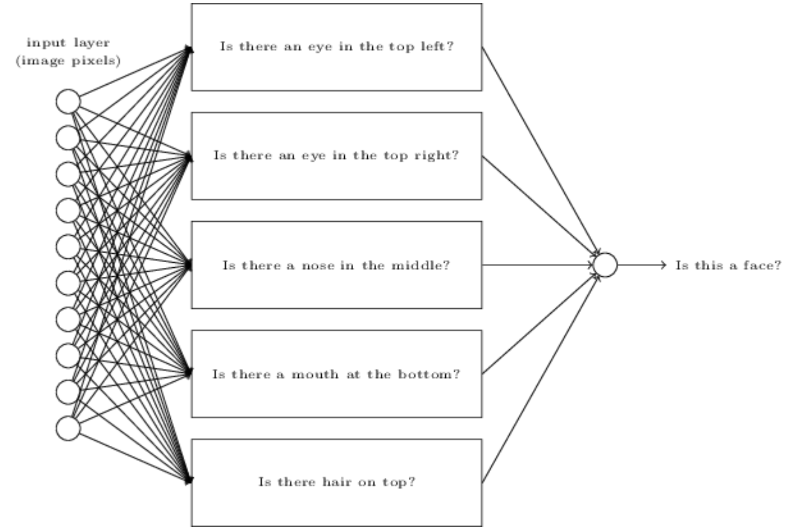
Pic. 10. Digit recognition using NN



Pic. 11. Speech recognition using NN



Pic. 12. ultrasound shape recognition



Pic. 13. Face recognition using NN

Q&A

Thank you