Neural networks

Victor Kitov

Yandex School of Data Analysis

Table of Contents

History

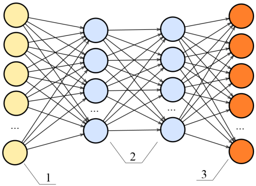
- Neural networks originally appeared as an attempt to model human brain

- Human brain consists of multiple interconnected neuron cells
    - cerebral cortex (the largest part) is estimated to contain 15–33 billion neurons
    - communication is performed by sending electrical and electro-chemical signals
    - signals are transmitted through axons - long thin parts of neurons.

Table of Contents

Definition

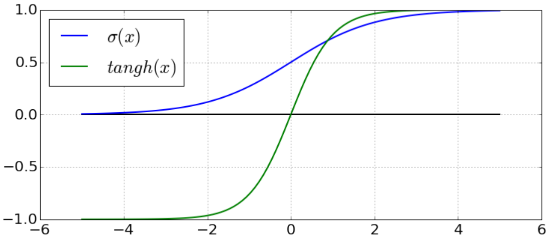
- linear / logistic regression - simplest case
- acyclic directed graph
- verticals called neurons
- edges correspond to certain weighs



- Structure of neural network:
  - 1-input layer
  - 2-hidden layers
  - 3-output layer

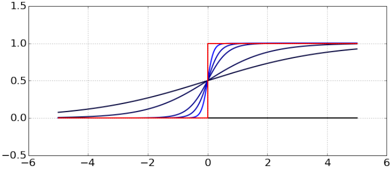
Definition

- Each neuron j is associated a non-linear transformation φ.
- For multilayer perceptron class neural networks φ belongs to a class of activation functions.
- Most common activation functions:
  - sigmoidal: $\sigma(x) = \frac{1}{1+e^{-x}}$
    - 1-layer neural network with sigmoidal activation is equivalent to logistic regression
  - hyperbolic tangent: $\text{tangh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

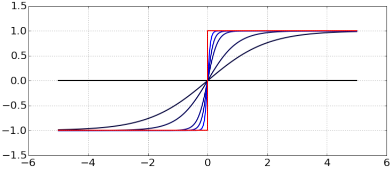


Activation functions

Activation functions are smooth approximations of step functions:



$\sigma(ax)$ limits to 0/1-step function as $a \to \infty$



tangh(ax) limits to -1/1-step function as $a \to \infty$

Definition details

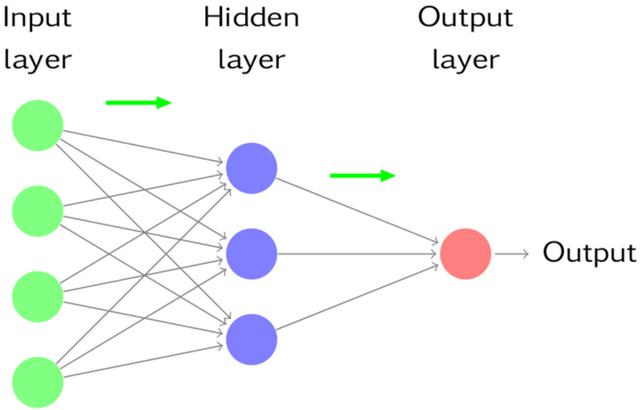
- Label each neuron with integer i.
- Denote: I_i - input to neuron i, O_i - output of neuron i
- Output of neuron i: $O_i = A(I_i)$, where A is activation function.
- Input to neuron i: $I_i = \sum_{k \in inc(i)} w_{ki} O_k + w_{k0}$,
    - w_{k0} is the bias term
    - $inc(i)$ is a set of neurons with outgoing edges to neuron i.
    - further we will assume that at each layer there is a vertex with constant output $O_{const} \equiv 1$, so we can simplify notation

$$I_i = \sum_{k \in inc(i)} w_{ki} O_k$$

Table of Contents

Output generation

- Forward propagation is a process of successive calculations of neuron outputs for given features.



Output generation

- Output layer transformations
  - regression: $\varphi(I) = I$
  - classification:
    - 2 classes: sigmoid, indicating target class probability
$$\varphi(I) = \frac{1}{1 + e^{-I}}$$
    - multiple classes: softmax, indicating probabilities of each class:
$$\varphi(I_i) = \frac{e^{O_i}}{\sum_{k \in OL} e^{O_k}}, \; i \in OL$$
where OL denotes neuron indices at output layer.

Number of layers selection

- Number of layers usually denotes all layers except input layer (hidden layers+output layer)
- We will consider only continuous activation functions.
- Classification:
    - single layer network selects arbitrary half-spaces
    - 2-layer network selects arbitrary convex polyhedron (by intersection of 1-layer outputs)
        - therefore it can approximate arbitrary convex sets
    - 3-layer network selects (by union of 2-layer outputs) arbitrary finite sets of polyhedra
        - therefore it can approximate almost all sets with well defined volume (Borel measurable)

Number of layers selection

- Regression
  - single layer can approximate arbitrary linear function
    - 2-layer network can model indicator function of arbitrary polyhedron
    - 3-layer network can uniformly approximate arbitrary continuous function (as sum of indicators of various polyhedra)

Sufficient amount of layers

Any continuous function on a compact space can be uniformly approximated by 2-layer neural network with linear output and wide range of activation functions (excluding polynomial).

- In practice often it is more convenient to use more layers with fewer amount of neurons
  - model becomes more interpretable and tunable

Table of Contents

Network optimization: regression

- Single output:

$$\frac{1}{N}\sum_{n=1}^{N}(\widehat{y}_n(x_n) - y_n)^2 \to \min_{w}$$

Network optimization: regression

- Single output:

$$\frac{1}{N}\sum_{n=1}^{N}(\widehat{y}_n(x_n)-y_n)^2 \to \min_w$$

- K outputs

$$\frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}(\widehat{y}_{nk}(x_n)-y_{nk})^2 \to \min_w$$

Network optimization: classification

- Two classes ($y \in \{0, 1\}$, $p = P(y = 1)$):

$$\prod_{n=1}^{N} p(y_n = 1|x_n)^{y_n} [1 - p(y_n = 1|x_n)]^{1-y_n} \to \max_w$$

Network optimization: classification

- Two classes ($y \in \{0, 1\}$, $p = P(y = 1)$):

$$\prod_{n=1}^{N} p(y_n = 1|x_n)^{y_n}[1 - p(y_n = 1|x_n)]^{1-y_n} \to \max_w$$

- C classes ($y_{nc} = \mathbb{I}\{y_n = c\}$):

$$\prod_{n=1}^{N}\prod_{c=1}^{C} p(y_n = c|x_n)^{y_{nc}} \to \max_w$$

Network optimization: classification

- Two classes ($y \in \{0, 1\}$, $p = P(y = 1)$):

$$\prod_{n=1}^{N} p(y_n = 1|x_n)^{y_n}[1 - p(y_n = 1|x_n)]^{1-y_n} \to \max_{w}$$

- C classes ($y_{nc} = \mathbb{I}\{y_n = c\}$):

$$\prod_{n=1}^{N}\prod_{c=1}^{C} p(y_n = c|x_n)^{y_{nc}} \to \max_{w}$$

- In practice log-likelihood is maximized.

Neural network optimization

- Let W denote the total dimensionality of weights space
- Let $E(\widehat{y}, y)$ denote the loss function of output
- We may optimize neural network using gradient descent:

while (stop criteria not met):
    $w^{k+1} = w^k - \eta\nabla E(w^k)$

- Standardization of features makes gradient descend converge faster
- Other optimization methods are more efficient (conjugate gradients)

Neural network optimization

- Direct $\nabla E(w)$ calculation, using

$$\frac{\partial E}{\partial w_i} = \frac{E(w+\varepsilon_i) - E(w)}{\varepsilon} + O(\varepsilon)$$

or better

$$\frac{\partial E}{\partial w_i} = \frac{E(w+\varepsilon_i) - E(w-\varepsilon_i)}{\varepsilon} + O(\varepsilon^2)$$

has complexity $O(W^2)$ [W forward propagations to evaluate W derivatives]

Backpropagation algorithm needs only $O(W)$ to evaluate all derivatives.

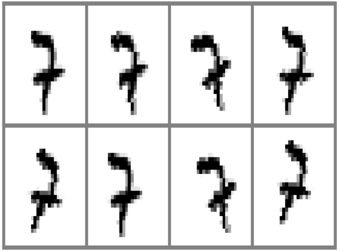
Multiple local optima problem

- Instability with respect to:
    - different starting parameter values
    - different subsamples
    - different feature selections
- Solutions
    - select best optimum from local optima
    - average predictions for different local optima

Table of Contents

Invariances

- It may happen that solution should not depend on certain kinds of transformations in the input space.
- Example: character recognition task
    - translation invariance
    - scale invariance
    - invariance to small rotations
    - invariance to small uniform noise



Invariances

- Approaches to build an invariant model:
    - augment training objects with their transformed copies according to given invariances
        - amount of possible transformations grows exponentially with the number of invariances
    - add regularization term to the target cost function, which penalizes changes in output after invariant transformations
        - see tangent propagation
    - extract features that are invariant to transformations
    - build the invariance properties into the structure of neural network
        - see convolutional neural networks

Augmentation of training samples

1. generate a random set of invariant transformations
2. apply these transformations to training objects
3. obtain new training objects

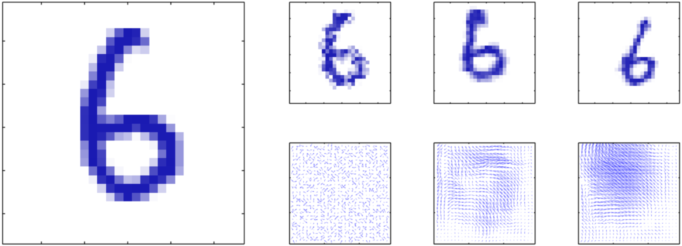
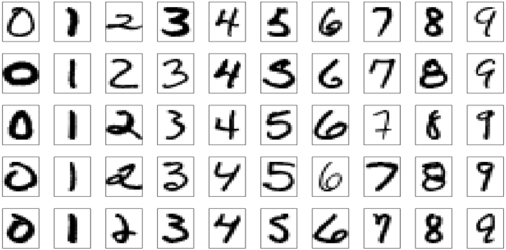


Table of Contents

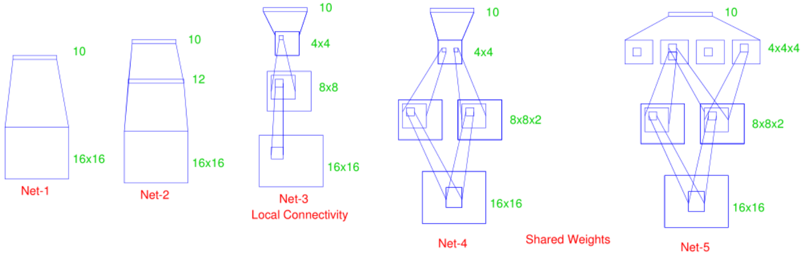
Case study (due to Hastie et al. The Elements of Statistical Learning)

ZIP code recognition task

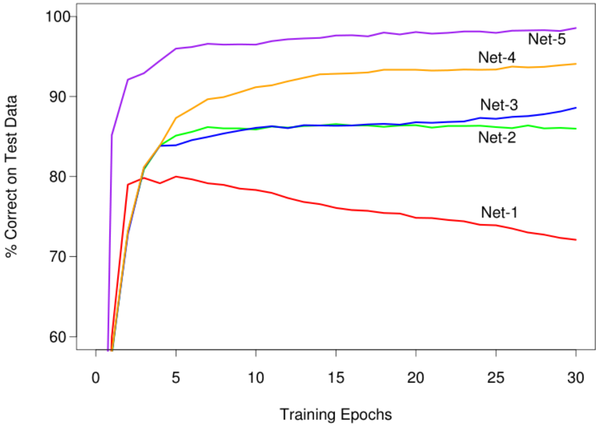


Neural network structures

Net1: no hidden layer
Net2: 1 hidden layer, 12 hidden units fully connected
Net3: 2 hidden layers, locally connected
Net4: 2 hidden layers, locally connected with weight sharing
Net5: 2 hidden layers, locally connected, 2 levels of weight sharing



Results


100
90
80
70
60
% Correct on Test Data
0
5
10
15
20
25
30
Training Epochs
Net-5
Net-4
Net-3
Net-2
Net-1

Conclusion

- Deep learning
- **Advantages of neural networks:**
  - can model accurately complex non-linear relationships
  - easily parallelizable
- **Disadvantages of neural networks:**
  - hardly interpretable ("black-box" algorithm)
  - optimization requires skill
    - too many parameters
    - may converge slowly
    - may converge to inefficient local minimum far from global one