neural networks in ocaml

colin shaw
07.14.2016

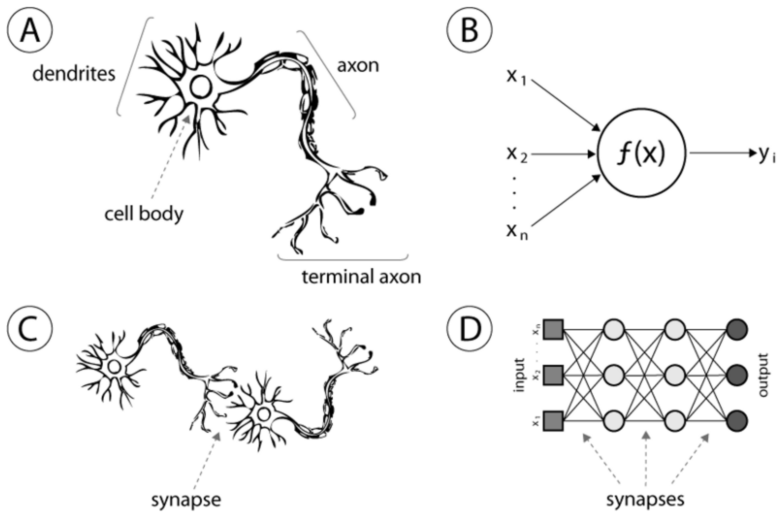
1. **introduction**
    a. welcome to computer science club
    b. thank you to revunit for sponsoring
    c. what are we talking about
        i. neural networks
            1. motivation
            2. history
            3. types
            4. theory
            5. implementation
        ii. blas / lapack
        iii. ocaml
            1. applicative / functional programming
            2. imperative features
            3. interface with other languages
            4. modular design

2. **motivations**
    a. problem solving
        i. problems we can solve explicitly
            1. closed form solutions
        ii. problems we cannot solve explicitly
            1. do not understand the problem
            2. do not understand how to optimize specific solution
            3. desire more organic solution
        iii. learning algorithms solve problems without us knowing how
    b. biological imperative
        i. solving problems without understanding explicit solutions
        ii. modeling neurons since we know they work

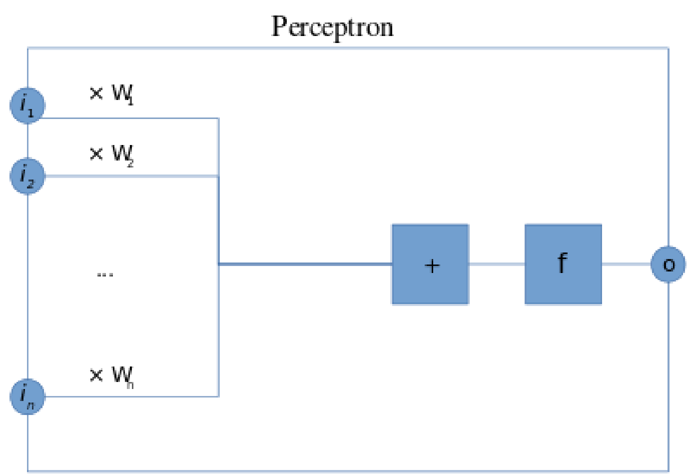
3. **alternatives**
    a. depends on type of problem
    b. some examples
        i. regression
        ii. analysis of variance
        iii. k-means and similar clustering
        iv. principal component analysis
        v. support vector machine
        vi. self organizing maps

4. **brief history**
    a. 1800s
        i. linear regression (legendre, gauss)
    b. 1900s
        i. gradient descent (hadamard)
    c. 1940s
        i. early architecture (mcculloch, pitts)
    d. 1950s
        i. simple supervised learning (perceptron; rosenblatt)
        ii. visual cortex experiments (hubel, wiesel)
    e. 1960s
        i. unsupervised learning
        ii. gradient descent backpropagation
        iii. multiple layer perceptron (deep learning)
        iv. grossberg, ivakhnenko
    f. 1970s
        i. convolution, subsampling (fukushima)
    g. 1980s
        i. convolutional backpropagation (rumelhart)
        ii. hopfield network, bolzmann machines
    h. 1990s
        i. convergence optimization
        ii. neural networks start winning pattern recognition contests
    i. 2000s
        i. gpus become prevalent in non-graphical computation
    j. 2010s
        i. mnist record broken by neural network
        ii. gpu-based neural network surpasses human vision recognition
        iii. neural network beats world class go player
        iv. neural network drives a car

5. **types of neural network**
    a. the inspiration

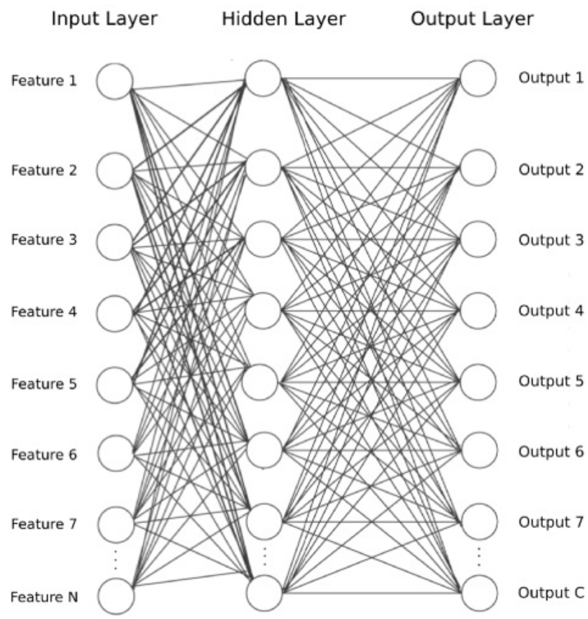


    b. perceptron
        i. 1957 by rosenblatt as mark i perceptron (physical machine)
        ii. sparked hype about artificial intelligence
        iii. can only solve linearly separable problems
        iv. cannot represent xor

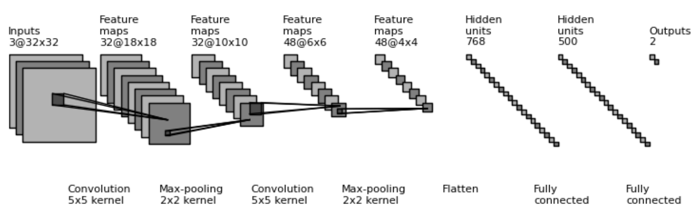


$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

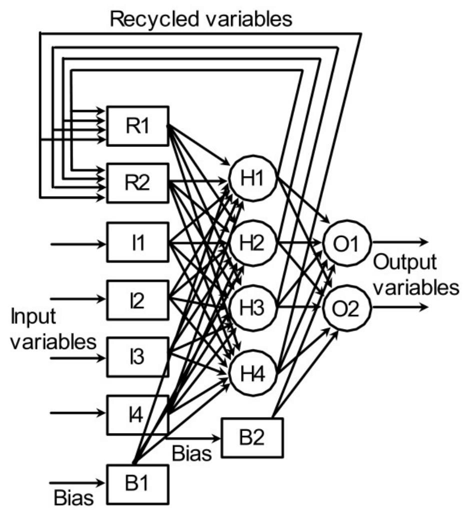
c. multiple layer perceptron
   i. one or more hidden layers
   ii. can represent xor



d. convolutional
   i. term derived generally from convolution operations on images
   ii. preliminary feature detectors
   iii. typically followed by mlp hidden layers



e. recurrent
    i. can learn based on training order
    ii. must abide shannon sampling theorem
    iii. scaling can be difficult
    iv. possible implementation
        1. inter-layer recurrence
        2. delay line recurrence



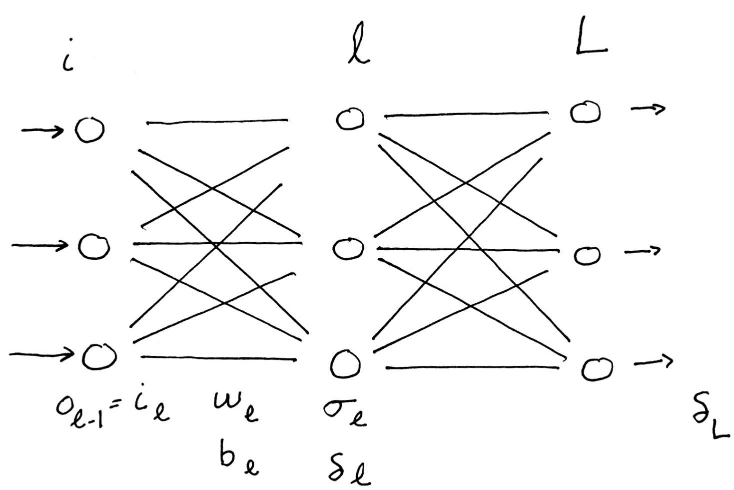
6. how the multiple layer perceptron works

a. simple regression example
    i. $y = m x + b$
    ii. compute output error for given input
        1. $\Delta y = y_{new} - y_{old}$
    iii. compute derivatives
        1. $\frac{dy}{dm} = x$
        2. $\frac{dy}{db} = 1$
    iv. update a and b
        1. let η be a learning constant
        2. $\Delta m = \eta \cdot \frac{dy}{dm} \cdot \Delta y = \eta \cdot x \cdot \Delta y$
        3. $\Delta b = \eta \cdot \frac{dy}{db} \cdot \Delta y = \eta \cdot \Delta y$

b. cost function
    i. a measure of output error
    ii. in our regression example: $\Delta y = y_{new} - y_{old}$
        1. the gradient of a cost function (∇C)
        2. cost function itself is $\frac{1}{2}(y_{new} - y_{old})^2$
    iii. in general can be any analytic function
    iv. to be able to perform mini-batch optimizations
        1. must satisfy $C = \frac{1}{n}\sum_{x=1}^{n} C_x$

c. activation function
    i. purpose
        1. facilitating better convergence and problem conditioning
        2. avoid stalling for zero-valued weights
        3. avoid excessive change for large weights and errors
    ii. common examples
        1. hidden layer (zero-centered, bounded)
            a. tanh
            b. softsign
        2. output layer (non-negative, bounded)
            a. rectified linear (relu)
            b. softplus

c. putting it all together
    i. Network diagram



    ii. compute forward pass
        1. $o_l = \sigma_l (w_l \cdot i_l + b_l)$
        2. i_0 is the input (feature)
        3. o_L is the final output

iii. compute final layer error
    1. compute gradient of cost function
        a. example: quadratic cost function
        b. $\nabla C = o_{new} - o_{old}$
    2. compute output error
        a. $\delta_L = \nabla C \odot \frac{d\sigma_L}{dx}(o_L)$
iv. recursively compute prior layer errors
    1. $\delta_{l-1} = [\,(w_l)^T \cdot \delta_l\,] \odot \frac{d\sigma_{l-1}}{dx}(o_{l-1})$
v. update weight
    1. $\Delta w_l = \frac{\eta}{m}\delta_l (o_{l-1})^T$
    2. η learning constant, m layer inputs
vi. update bias
    1. $\Delta b_l = \frac{\eta}{m}\delta_l$
    2. η learning constant, m layer inputs
vii. repeat until satisfied with training

d. training issues
    i. convergence
        1. cost function minimization
        2. problem solves general test cases sufficiently
    ii. overfitting
        1. too much training fits the training set
        2. does not make for a general purpose solution

e. mini-batches
    i. compute multiple forward stages
    ii. cost function must satisfy summation condition
    iii. computational complexity
        1. eliminates most of the backpropagation steps
        2. eliminates most of the weight and bias updates
        3. problem still remains in same class, just faster

7. Implementation

a. intro to ocaml
    i. applicative / functional programming
        1. lists
        2. first class functions
        3. higher order functions
    ii. imperative features
        1. refs
        2. arrays

b. intro to blas / lapack
    i. blas 1
        1. vector - scalar
        2. O(n) operations for O(n) data
        3. memory bandwidth limited
    ii. blas 2
        1. matrix - vector
        2. $O(n^2)$ operations for $O(n^2)$ data
        3. memory bandwidth limited
    iii. blas 3
        1. matrix - matrix
        2. $O(n^3)$ operations for $O(n^2)$ data
        3. most able to be optimized
    iv. compare with lists in heap

c. lacaml
    i. brief overview of organization
    ii. what we are using it for

d. source
    i. math.ml
        1. functional
            a. lists
            b. heap
        2. imperative
            a. arrays
            b. fixed memory block
        3. performance and elegance tradeoffs
        4. binop versus full blas / lapack routines
    ii. types.ml
    iii. activation.ml
    iv. cost.ml
    v. layer.ml
    vi. network.ml
        1. three data structures
            a. layer: $[\ell_1; \ell_2; \ell_3; \dots ; \ell_L]$
            b. forward: $[i; o_1; o_2; \dots ; o_{L-1}]$
            c. backward: $[\delta_1; \delta_2; \delta_3; \dots ; \delta_L]$
    vii. utils.ml
    viii. time.ml

e. examples (including data)
    i. iris.ml
    ii. letters.ml
    iii. mnist.ml

f. observations
    i. issues training letters example
    ii. mini-batch optimization and speed enhancements
        1. can treat mini-batch as matrix input instead of vector
        2. blas 3 optimizations matter
        3. significantly increased performance, same computational complexity

8. the future

a. technique
    i. half-precision floating point
    ii. convolutional, recurrent networks
b. gpus
    i. baseline
        1. raspberry pi 2 b (used in demo)
        2. 98 mflops (single precision, one core)
    ii. modern gpu
        1. nvidia tesla p100
        2. 18.7 tflops half-precision
c. example libraries
    i. accelerator
        1. cuda
        2. opencl
    ii. machine learning
        1. tensor flow
        2. caffe
        3. theano