Machine Learning

Neural Networks, part 2; and the beginning

Jeff Abrahamson

octobre 2020

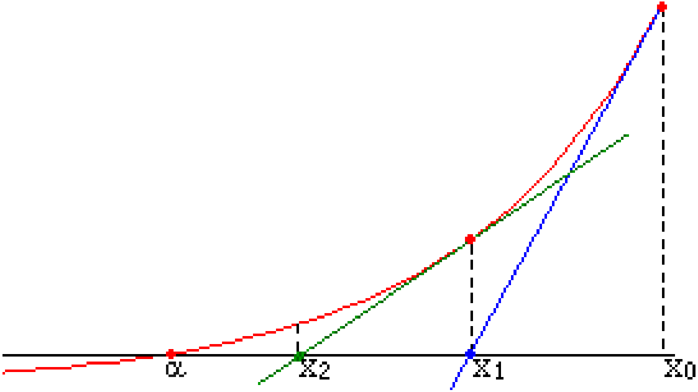
Backpropagation

Backpropagation

Assign small random weights
while *not converged* **do**
    Feed forward to find values at each node
    Backpropagate to correct weights:
    Algorithm `algo()`
        Compute partial derivatives
        Use partial derivatives to compute gradient
        Use gradient to update weights (classic gradient descent)
end

Backpropagation

Newton's Method



Newton's Method

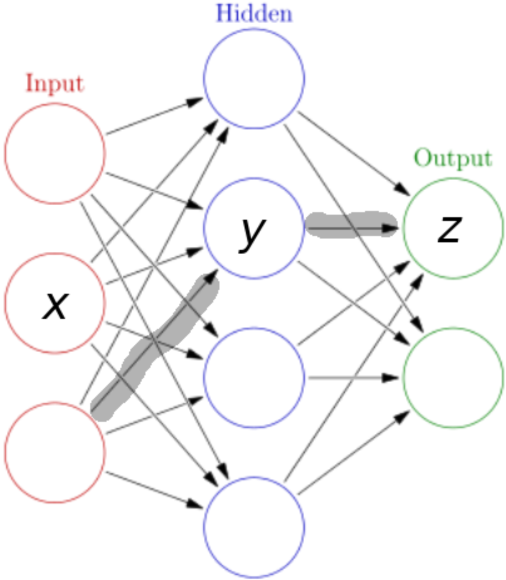
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{f(x_n)}{\frac{\mathrm{d}f}{\mathrm{d}x}(x_n)}$$

Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{f(x_n)}{\frac{\mathrm{d}f}{\mathrm{d}x}\left(x_n\right)}$$

or

$$y = \frac{\mathrm{d}f}{\mathrm{d}x}\left(x_n\right)(x - x_n) + f(x_n)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

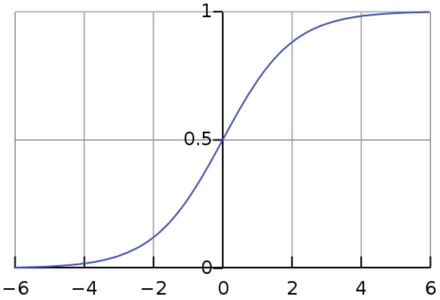
Perceptron

Vocabulary

Activation function

How the neuron decides when to fire.

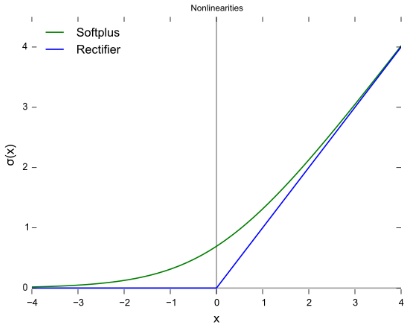
Vocabulary



Sigmoid

By Qef (talk) - Created from scratch with gnuplot, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=4310325

Vocabulary

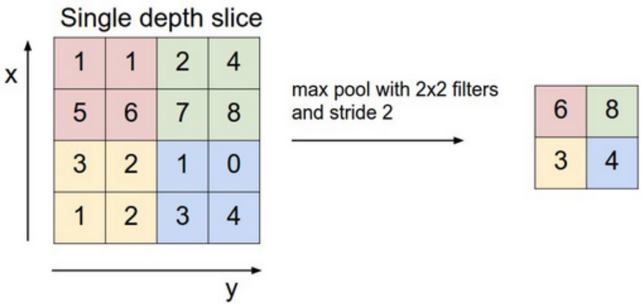


Wikimedia Commons

ReLU and Softmax

Rectified Linear Unit

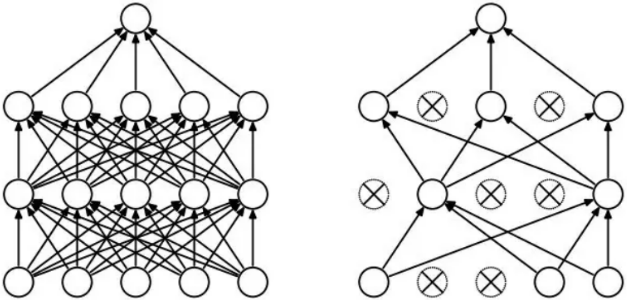
Vocabulary



Pooling

max pooling
downsampling

Vocabulary



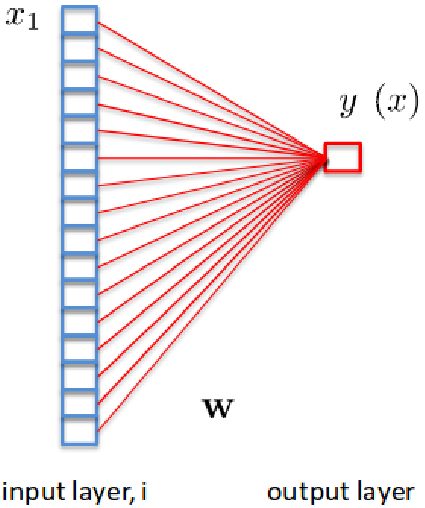
Dropout

Multilayer Perceptron (MLP)

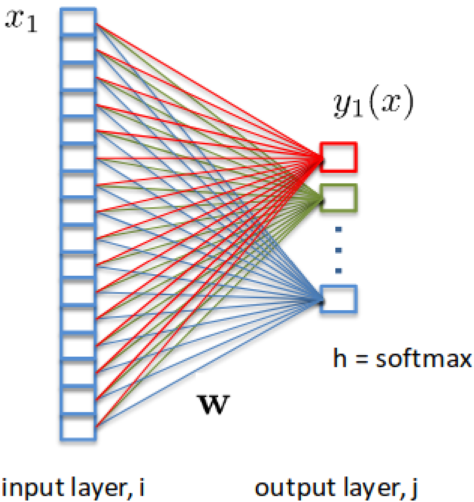
MLP

1. Multiple layers
2. First layer is linear
3. Later layers use non-linear activation functions (typically sigmoid)
4. Feedforward
5. Can find non-linear separators

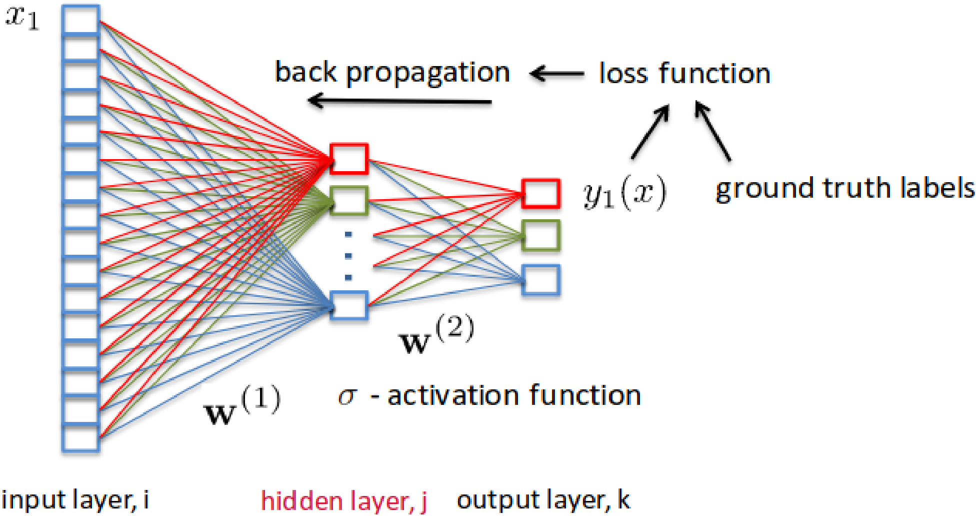
MLP



MLP



MLP


x_1
back propagation
loss function
$y_1(x)$
ground truth labels
$\mathbf{w}^{(2)}$
$\mathbf{w}^{(1)}$
σ - activation function
input layer, i
hidden layer, j
output layer, k

Example: MNIST

Example: simple classification tasks

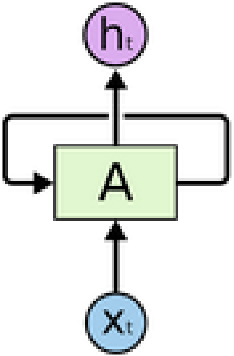
Example: time series

Recurrent Neural Networks

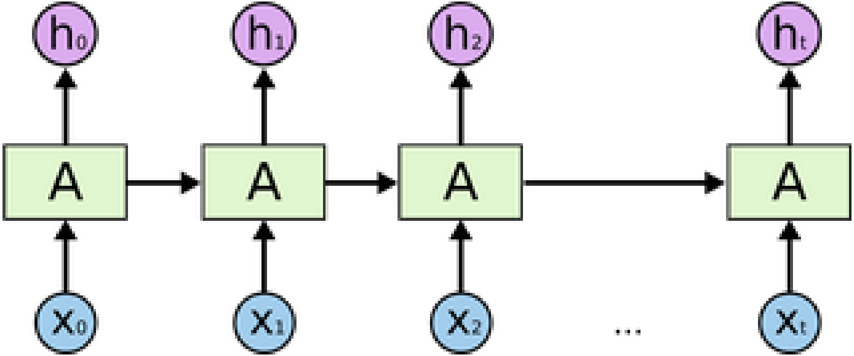
loops

loops

so also memory



https://colah.github.io/posts/2015-08-Understanding-LSTMs/



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

ConvNets

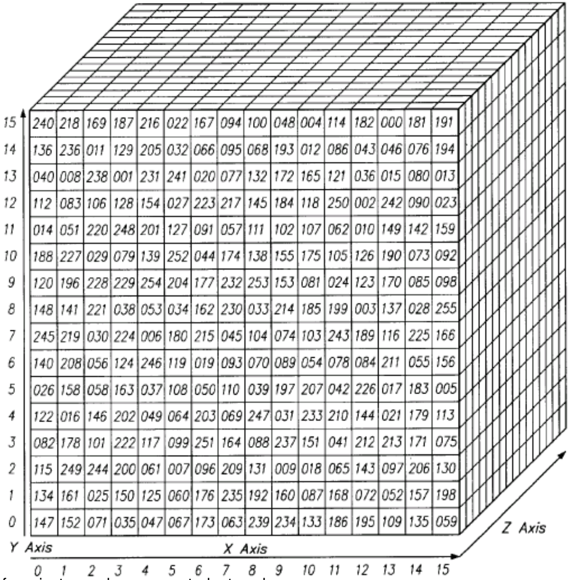
Tensors

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Tensors

$$\begin{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} & \begin{pmatrix} 3 \\ 4 \end{pmatrix} & \begin{pmatrix} 5 \\ 6 \end{pmatrix} \\ \begin{pmatrix} 7 \\ 8 \end{pmatrix} & \begin{pmatrix} 9 \\ 10 \end{pmatrix} & \begin{pmatrix} 11 \\ 12 \end{pmatrix} \end{pmatrix}$$

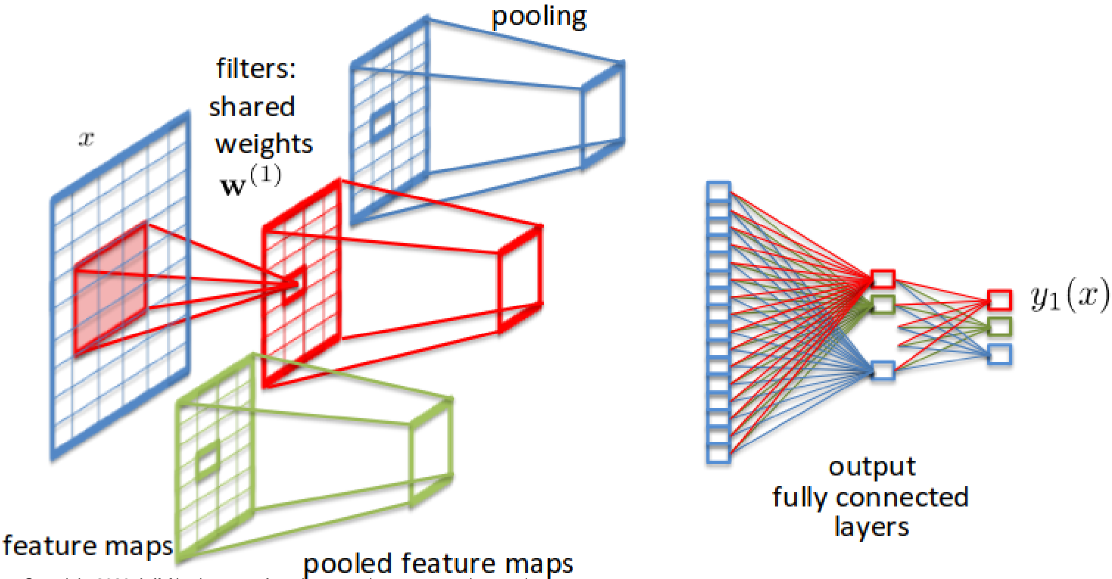
Tensors


15 240 218 169 187 216 022 167 094 100 048 004 114 182 000 181 191
14 136 236 011 129 205 032 066 095 068 193 012 086 043 046 076 194
13 040 008 238 001 231 241 020 077 132 172 165 121 036 015 080 013
12 112 083 106 128 154 027 223 217 145 184 118 250 002 242 090 023
11 014 051 220 248 201 127 091 057 111 102 107 062 010 149 142 159
10 188 227 029 079 139 252 044 174 138 155 175 105 126 190 073 092
9 120 196 228 229 254 204 177 232 253 153 081 024 123 170 085 098
8 148 141 221 038 053 034 162 230 033 214 185 199 003 137 028 255
7 245 219 030 224 006 180 215 045 104 074 103 243 189 116 225 166
6 140 208 056 124 246 119 019 093 070 089 054 078 084 211 055 156
5 026 158 058 163 037 108 050 110 039 197 207 042 226 017 183 005
4 122 016 146 202 049 064 203 069 247 031 233 210 144 021 179 113
3 082 178 101 222 117 099 251 164 088 237 151 041 212 213 171 075
2 115 249 244 200 061 007 096 209 131 009 018 065 143 097 206 130
1 134 161 025 150 125 060 176 235 192 160 087 168 072 052 157 198
0 147 152 071 035 047 067 173 063 239 234 133 186 195 109 135 059
Y Axis
X Axis
Z Axis
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

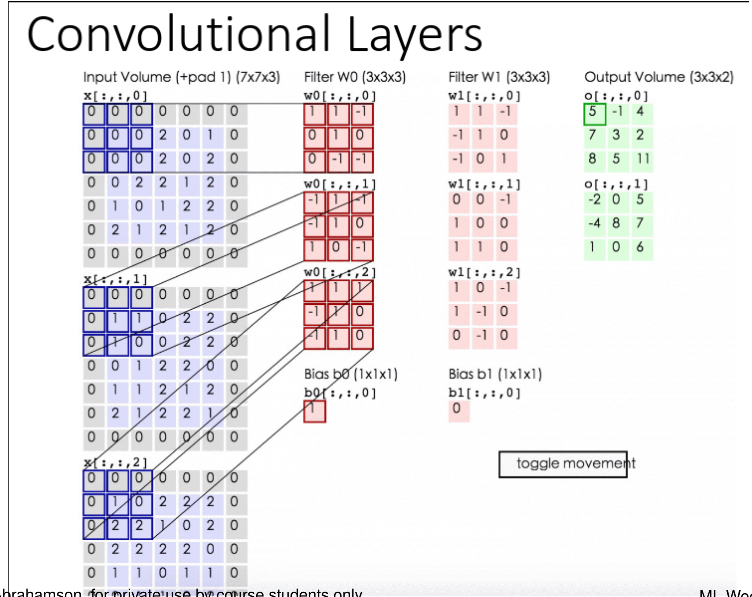
Convolutional Neural Networks

The neurons are convolutions

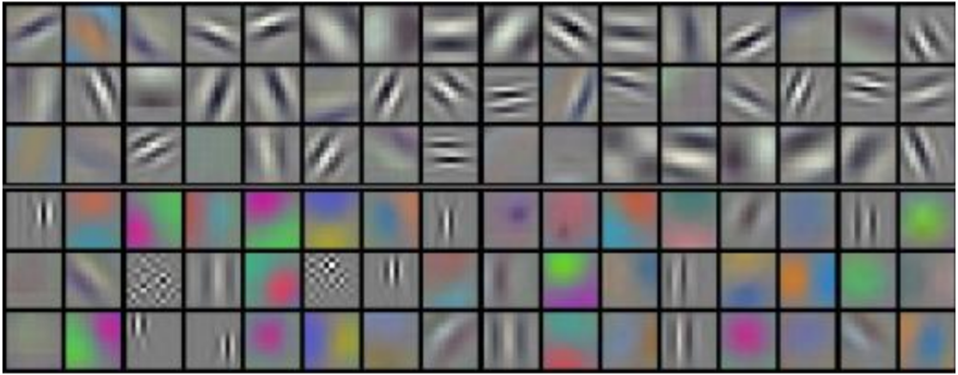
ConvNets


pooling
filters:
shared
weights
$\mathbf{w}^{(1)}$
x
$y_1(x)$
output
fully connected
layers
feature maps
pooled feature maps

ConvNets


Convolutional Layers
Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 2 0 1 0
0 0 0 2 0 2 0
0 0 2 2 1 2 0
0 1 0 1 2 2 0
0 2 1 2 1 2 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 1 0 2 2 0
0 1 0 0 2 2 0
0 0 1 2 2 0 0
0 1 1 2 1 2 0
0 2 1 2 2 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 1 0 2 2 2 0
0 2 2 1 0 2 0
0 2 2 2 2 0 0
0 1 1 0 1 1 0
Filter W0 (3x3x3)
w0[:,:,0]
1 1 -1
0 1 0
0 -1 -1
w0[:,:,1]
-1 1 -1
-1 1 0
1 0 -1
w0[:,:,2]
1 1 1
-1 1 0
-1 1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
1 1 -1
-1 1 0
-1 0 1
w1[:,:,1]
0 0 -1
1 0 0
1 1 0
w1[:,:,2]
1 0 -1
1 -1 0
0 -1 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
5 -1 4
7 3 2
8 5 11
o[:,:,1]
-2 0 5
-4 8 7
1 0 6
toggle movement

ConvNets



Example: images

Transfer Learning

- Pre-trained models
- Only retrain the classifier at the end

Example: PlacesVGG (2015)

PlacesVGG (Places2 2015)

`https://static.turi.com/models/.../places_vgg_16-1.0.tar.gz`

This model is trained with VGG-16 architechture, on the Places2 dataset. The Places2 dataset contains 8 million images of 400 different scene categories. More details about the dataset can be found at http://places2.csail.mit.edu/

https://turi.com/products/create/docs/graphlab.mxnet.pretrained_image_model.html

The Beginning

Perspectives

- Data science is iterative
- Start simple, get better

Perspectives

- Data science is iterative
- Start simple, get better

Sometimes no business case to do more.

Risks

- Nothing is guaranteed, but competitors are innovating and experimenting
- Examples from past projects can help, but often leads to “this is different”

Success

Think early about how to measure success.

Success

Think early about how to measure success.

The definition of success will evolve.


questions?