

Neural Networks

Saumya Bhatnagar

March 21, 2020

Table of contents I

General

ANN

RNN

CNN

Optimization

computer vision

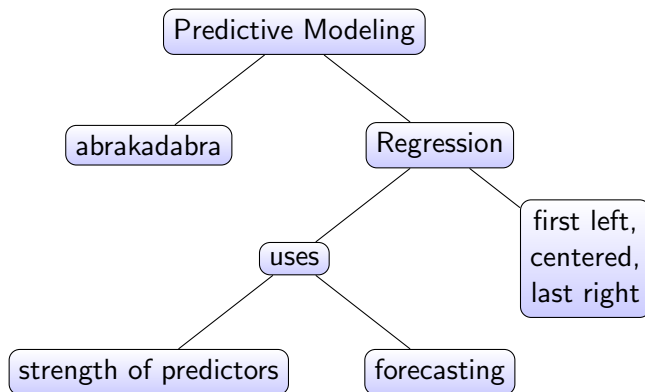
ALGORITHMS

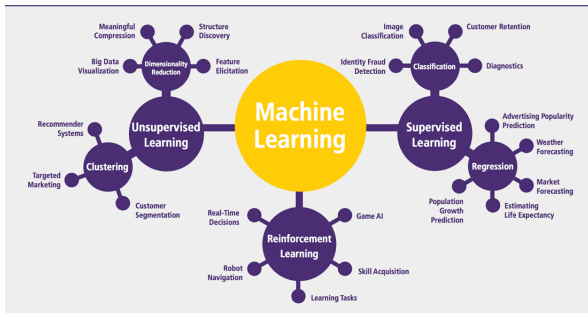
Other topics

Graph Theory

Algorithms & Data Structures

RBM - Restrictive Boltzmann Machine





	Categorical values	Continuous values
Supervised learning	Two-class/Binary classification <ul style="list-style-type: none"> Decision Forest Decision Tree Naive Bayes/Bayes Classifier (Deep) Neural Networks Support Vector Machines Multi-class classification <ul style="list-style-type: none"> Decision Forest Logistic Regression (Deep) Neural Networks 	Regression <ul style="list-style-type: none"> Bayesian Linear Regression Decision Forest Regression Decision Tree Regression Linear Regression Neural Network Regression Ordinary Least Squares Regressor
Unsupervised learning	Association Rule Learning <ul style="list-style-type: none"> Apriori algorithm Frequent Pattern Growth Classification <ul style="list-style-type: none"> Autoencoders Restricted Boltzmann Machines 	Clustering <ul style="list-style-type: none"> Density-based Clustering Hierarchical Clustering Partitional Clustering (incl. K-means) Dimensionality reduction <ul style="list-style-type: none"> Principal Component Analysis (PCA)

	Categorical values
Supervised learning	Two-class/Binary classification <ul style="list-style-type: none"> Decision Forest Decision Tree Naive Bayes/Bayes (Deep) Neural Netw Support Vector Ma Multi-class classification <ul style="list-style-type: none"> Decision Forest Logistic Regression (Deep) Neural Netw
Unsupervised learning	Association Rule Learning <ul style="list-style-type: none"> Apriori algorithm Frequent Pattern Gr Classification <ul style="list-style-type: none"> Autoencoders Restricted Boltzma



Why to Study Neural Computation

- ▶ to study how brain works
- ▶ to understand the styles of parallel computations inspired by neurons and their adaptive connections
- ▶ to solve practical problems by using novel learning algorithms inspired by brain

TIMIT BENCHMARK FOR SOUND

content...

Linear Neurons I

Binary Threshold Neurons

Output, $z = b + \sum x_i w_i$

Linear Neurons

Output, $y = b + \sum x_i w_i$

= activities on input line

times weight

$x_i = i^{th}$ input

$w_i = i^{th}$ weight

$b = bias$

$$y = \begin{cases} 1, & z \geq 0 \\ 0, & otherwise \end{cases} \quad (1)$$

OR Output, $z = \sum x_i w_i$

$$y = \begin{cases} 1, & z \geq \theta \\ 0, & otherwise \end{cases} \quad (2)$$

$\theta = \text{threshold} = -b$

Linear Neurons II

Sigmoid Neurons

Output, $z = b +$

$$\sum x_i w_i$$

derivatives makes

learning easy

use logistic function, y

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

Stochastic Binary

Neurons: o/p of
logistic as probability
of spike in a short time
window

Rectified Linear Neurons

aka Linear Threshold Neurons

Output, $z = b + \sum x_i w_i$

= non-linear function \leftarrow linear weighted
sum of inputs

$$y = \begin{cases} z, & z \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Apply stochastic methods in ReLU

o/p = rate of producing spikes

(deterministic)

use rate to get the actual times at which
spikes are produced \Rightarrow Random process

Linear Neurons III

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (s.k. a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [21]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [23]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_{\sigma_e}(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016
(<http://sebastianraschka.com/>)

Supervised vs Unsupervised vs Reinforcement Learning

Model-class: Predicted output, $\hat{y} = f(x; W)$

W = Numerical parameters

input vector

Regression, minimize $\frac{1}{2}(y - \hat{y})^2$

Reinforcement Learning:

o/p is action or sequence of actions

supervision by rewards

max expected sum of future rewards

use discount factor for delayed rewards, so won't have to look too far into future

why not RL?: in long sequence of actions rewards are delayed so it's hard to know where we went wrong
a scalar reward doesn't supply much info

Feedforward NN

information comes from input unit, flows in one direction towards hidden layer and gives the output

if more than one hidden layer \Rightarrow **Deep NN**

compute transformations:

activities of neurons in each layer are a non-linear func of the activities in layer below

Perceptron convergence procedure

training binary o/p neurons as classifiers

add an extra component with value 1 to each i/p vector. the "bias" wt on this component is minus the threshold. now forget the threshold.

pick training cases using any policy that ensures that every training case will keep getting picked

- ▶ if o/p is correct, do nothing
- ▶ if incorrectly o/ps to 0, add i/p vector to wt vector
- ▶ if incorrectly o/ps to 1, subtract i/p vector from wt vector

Padding and stride

Padding is used to preserve the boundary information , since without padding they are only traversed once.//

example

content...

Autoencoders

content...

Sequential Data

o/p dependent on prev computations saved in memory

Feedforward(touches node once) vs **RNN**(loops: present in recent past)

why not feedforward?: can't predict the next word in a sentence if I use feed-forward

Backpropagation of error and gradient descent

Error will return via **Backpropagation** adjusting weights
sentence of 5 words, 5 layered NN

- ▶ ~ very deep nets with hidden layer per time slice
- ▶ get input at every time slice
- ▶ use same weights at every time slice

RNN with multiple hidden layers are special case where some hidden → hidden connections are missing

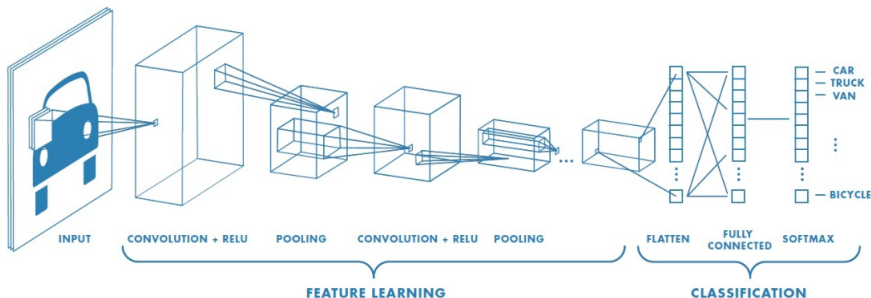
content...

convnets I

A convolution operation is basically computing a dot product between their weights and a small region they are connected (currently overlapping) to in the input volume. This will change the dimensions depending on the filter size used and number of filters used.

Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

convnets II



a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer

spatial size of the output volume

for calculating how many neurons "fit", use

$$(W-F+2P)/S + 1$$

is a function of the:

input volume size (W),

the receptive field size of the Conv Layer neurons (F),

the stride with which they are applied (S),

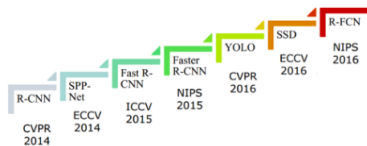
and the amount of zero padding used (P) on the border

For example for a 7×7 input and a 3×3 filter with stride 1 and pad 0 we would get a 5×5 output. With stride 2 we would get a 3×3 output.

Non-Linear Optimization

ALGORITHMS

types



General

ooooo
ooooooooo
oo
oo
oo

Optimization

o

computer vision

o●oooo

Other topics

oooo
o

ALGORITHMS

types

content...

Saumya Bhatnagar

Neural Networks

types

content...

General

○○○○○
○○○○○○○
○○
○○

Optimization

○

computer vision

○○○●○○

Other topics

○○○○
○

ALGORITHMS

types

content...

Saumya Bhatnagar

Neural Networks

General

ooooo
oooooooo
oo
oo
oo

Optimization

o

computer vision

oooo●o

Other topics

oooo
o

ALGORITHMS

types

content...

Saumya Bhatnagar

Neural Networks

General

ooooo
oooooooo
oo
oo
oo

Optimization

o

computer vision

ooooo●

Other topics

oooo
o

ALGORITHMS

types

content...

Saumya Bhatnagar

Neural Networks

OOPS

Overloading vs overriding

Abstract vs Interface

content...

Principle

ACID: Atomic ... Isolate ...

Dependency Inversion Principle: High level module shouldn't depend on low level module

Liskov Principle

Interface segregation

CURD (Create, Read, Update, Delete)

Non

content...

Thank You!