

Human Guide To Machine Learning



Python Mauritius UserGroup (pymug)

More info: mscc.mu/python-mauritius-usergroup-pymug/

Where Are We? 🏠

| Why | Where |
|----------------------|---|
| codes | github.com/pymug |
| share events | twitter.com/pymugdotcom |
| ping professionals | linkedin.com/company/pymug |
| all info | pymug.com |
| discuss | facebook.com/groups/318161658897893 |
| tell friends by like | facebook.com/pymug |

Support Us

Please support us by subscribing to our mailing list:

<https://mail.python.org/mailman3/lists/pymug.python.org/>

Abdur-Rahmaan Janhangeer
(Just a Python programmer)

twitter: [@osdotsystem](#)


github: github.com/abdur-rahmaanj

www.pythonmembers.club

Human Guide To Machine Learning

What is Machine Learning (ML)?

wikipaedia defines machine learning as:

the scientific study of algorithms and statistical models that computer systems use to progressively improve their performance  on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task

data -> train -> do tasks

Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning (no)

Supervised Learning

- Classification
- Regression
- Mixed Methods

Unsupervised Learning

- Clustering
- Association Analysis

Supervised Learning

supervised learning

Classification:

- logistic regression
- supervised clustering

supervised learning

Regression

- linear regression (single value)
- multivariate linear regression

supervised learning

Mixed Methods

- tree based
- random forest
- neural networks
- support vector machine

supervised learning: classification

logistic regression

classifying data into categories. instead of predicting continuous values, we predict discrete values.

continuous and discrete values

continuous values means if you have $y = mx + c$, you can have y for x : 34, x :34.1, x :34.12 etc, as much as you want. discrete values means the next value after a is b then c etc

to sum it, there are two ways to see counting. one is you say: 1 2 3 4 (discrete), and one you see infinity between 1 and 2 like 1.1, 1.11, 1.111111 etc (continuous)

logistic : the careful organization of a complicated activity so that it happens in a successful and effective way

regression means prediction

let us say we have a table of watermelon 🍉 and 🍏 apple weights.

let us put a 0 for watermelon and a 1 for apple

our data in kg/(0,1) :

| | | |
|------|--|---|
| 9.5 | | 0 |
| 10.0 | | 0 |
| 0.1 | | 1 |
| 9.4 | | 0 |
| 0.2 | | 1 |

we build a model according to our data. so, light weights (0.1 to 0.2) seem to be 1 and heavy (10 to 11) seems to be 0. that's our model.

now we throw in a weight, let us say 10.4, it will pass it in the model it will say 0.

supervised learning: classification

Supervised Clustering Methods (kNN)


well that is a kmeans method. let us say we have some data



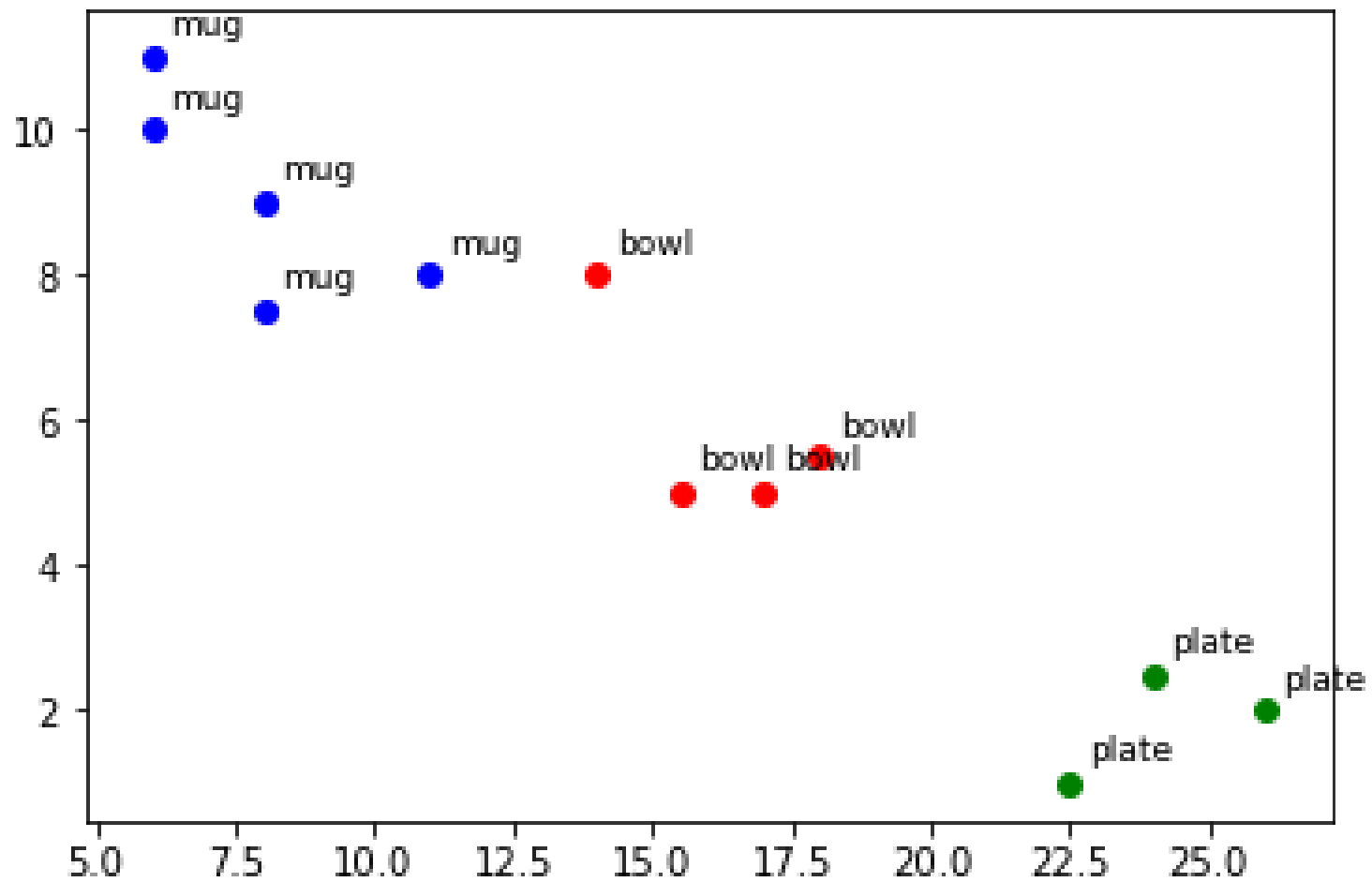
petal – plant type

| length | width | type |
|--------|-------|------|
| 2 | 0.5 | A |
| 4 | 1 | B |
| 1.5 | 0.3 | A |
| 6 | 1.7 | B |

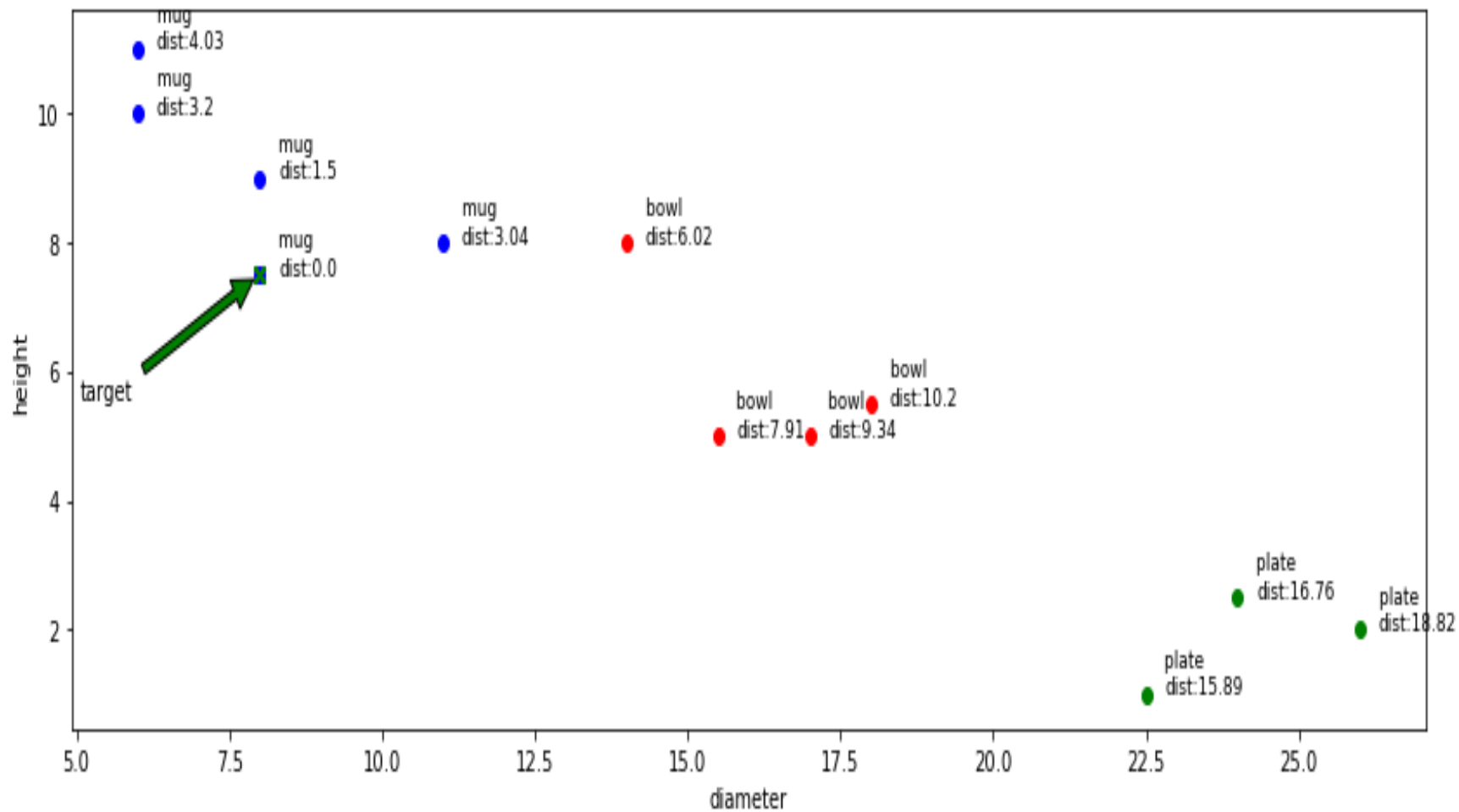
let us have a sample of petal length 5 and petal width 1.5 and we want to say if it's of type A or B. if we plot the graph, we can see two points (a cluster) at the bottom left corner and two points (another cluster) at the top right corner. 5,1.5 is nearer to the top right corner, hence type B

but how do we define near? near is by distance. we calculate distance  between each coordinate and if it is nearer to more type B, we say it is of type B. now we can add more data and they will thus be classified

example:



classifying



Supervised Learning: Regression

note: independent variables are also called *features*

regression simply means prediction. there are many types of regression methods:

- simple linear regression
- multivariate linear regression
- polynomial regression
- ridge regression
- lasso regression

Simple Linear Regression

Predicting for only two variables, a dependent and an independent one

let us say that we have many points on a graph of x,y and want to draw a line of best fit. we draw the line of best fit then we predict for further values of x and y. i.e. it calculates the values of m and c

in $y = m * x + c$

Multivariate Linear Regression

means predicting for one dependent variable and two or more independent variables

the formula is in the format

ind. var. means independent variable

```
stuff = m1 * ind. var. 1 + m2 * ind. var. 2 + ... + b
```

or

```
stuff = m1*feature1 + m2*feature2 + ... + b
```

b is the intercept

many real world examples better fit with multivariate regression.
like you might want to calculate fuel cost of trip based on many
factors such as state of engine, age etc etc

we train our data to get the m_1 m_2 m_3 etc then we can predict our
target value for such and such situations

Ridge regression

definition of ridge:

A ridge or a mountain ridge is a geological feature consisting of a chain of mountains or hills that form a continuous elevated crest for some distance.

just like the tops of mountains go up and down, similarly in the case where a graph's shape is like mountain tops, going up and down, ridge regression is used as a *regularisation* method to have a simple curve so as to ignore large coefficients and get better predictions

another name: tikhonov regularization

exercise:

1. dig into the maths of how
 - i. linear regression works (how the line of best fit is drawn)
 - ii. multivariate regression works
 - iii. polynomial regression works

Gradient Descent and Cost Function

Cost Function

cost function is also called mean squared error.

well mean means sum of elements / number of elements. here we take the sum of all squared errors

$$(\text{error1}^2 + \text{error2}^2 + \text{error3}^2)/3$$

/3 as there are 3 errors

we define error as the difference in y between your point and the y on the line you are trying to fit

-> y predicted – y on line

-> y predicted – m * x on line + c

Gradient Descent

wikipaedia defines gradient descent such:

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function.

well first, that has nothing specific to machine learning but concerns more maths

iterative means it repeats a process again and again

the minimum of a function is the lowest point of a u shape curve

in machine learning it means finding the line of best fit for a given training data

if you plot cost v/s m or cost v/s c you'll get a u shape graph

The Aim

when you plot the above graph, the minimum is the point where the error is the lowest (that's when you get the line of best fit). now, how exactly do you find it?

The How

well you plot either of the above graphs i.e. cost versus m or cost versus b and you find the minimum by checking the gradient at each point. at the minimum the gradient is 0 (gradient of straight line)

for the curve, you apply calculus to get the gradient function.

About Steps

well when you start, you need to check the gradient after an interval of c . when we see the gradient rising again we know we've found our minimum.

too small an interval of c (step) and your program might run too long

too big an interval of c and you miss your minimum

⚽ exercise:

1. google up stochastic gradient descent

supervised learning

Mixed Methods

- tree based
- random forest
- neural networks
- support vector machines

mixed methods are used for classification and regression.

🌱 tree based method

those trees used for both for classification and regression are called Classification And Regression Trees (CART) models

let us say that we want to predict whether an event will be good or bad, the event being having a good day at school

our data looks as follows

t. represents teacher

a means absent

p means present

mood stands for parents' mood

g good. b bad

hwork means homework

d done

nd not done

| Teacher | Mood | Hwork | Result |
|---------|------|----------|----------|
| absent | good | done | good day |
| present | bad | done | good day |
| absent | good | not done | good day |
| absent | bad | done | bad day |
| present | bad | not done | bad day |
| absent | good | not done | good day |
| present | bad | done | good day |
| absent | good | not done | good day |
| absent | good | done | good day |

let us say that today the student entered the school. he wants to know how his day will go, today he has

teacher : present

mood : good

hwork : not done

about splitting

the first step is to split the tree to get a high purity index (majority).
if we split by teacher's presence first, we get

```
absent
  good day 5 bad day 1
present
  good day 2 bad day 1
```

if we split by parents' mood we get

```
good
  good day 5 bad day 0
bad
  good day 2 bad day 2
```

if we split by homework done we get

done

good day 4 bad day 1

not done

good day 3 bad day 1

the highest index of purity was with parents' mood with good 5
and 0 bad day

we start with it

mood

— good

| | | | | | | |
|---|--|---|--|----|--|----|
| a | | g | | d | | gd |
| a | | g | | nd | | gd |
| a | | g | | nd | | gd |
| a | | g | | nd | | gd |
| a | | g | | d | | gd |

good day 5 bad day 0

— bad

| | | | | | | |
|---|--|---|--|----|--|----|
| p | | b | | d | | gd |
| a | | b | | d | | bd |
| p | | b | | nd | | bd |
| p | | b | | d | | gd |

good day 2 bad day 2

so bad mood must be split further as good mood had 100% purity with 5 good results, but we'll stop

our condition is

| | | |
|-----------|--|----------|
| teacher | | present |
| mood | | good |
| home work | | not done |

meaning today the mood was good, and when splitting most results for mood good returned good day

entropy and gain

entropy is just another word for expected value

in the past post, we decided what to use to split based on purity index. we can do the same thing mathematically (easier) by

P+ means probability of target (good day in our case)

P- means probability not target (bad day)

$\log_2(P_-)$ means $\log(P_-)$ base 2

$$H = -(P_+)\log_2(P_+) - (P_-)\log_2(P_-)$$

the above is the formula for the purity of subset. it measures how likely you get a positive element if you select randomly from a particular subset

let us take teacher's presence. we had

```
absent
  good day 5 bad day 1
present
  good day 2 bad day 1
```

$$H(\text{absent}) = -(5/6)\log_2(5/6) - (1/6)\log_2(1/6) \\ = 0.65$$

$$H(\text{present}) = -(2/3)\log_2(2/3) - (1/3)\log_2(1/3) \\ = 0.92$$

0 is extremely pure and 1 is extremely impure
so absent is purer than present

gain

gain is used to determine what to split first by finding the feature with the highest gain. 0 means irrelevant. 1 means very relevant

gain is defined by

$$\text{gain} = H(S) - \sum (SV/S) H(SV)$$

S -> total number of points in leaf

v is the possible values and SV -> subset for which we have those values

H is our entropy as above

taking for teacher's presence, our gain is

our S is 9 (records)

$$\text{gain} = H(\text{presence}) - \sum (SV/9 * H(SV))$$

$$\text{gain} = H(\text{presence}) - (6/9 * 0.65) - (3/9 * 0.92)$$

$$\begin{aligned} \text{our } H(\text{presence}) \text{ is } & -(7/9)\log_2(7/9) - (2/9)\log_2(2/9) \\ & = 0.76 \end{aligned}$$

(here we are checking for positive outcome for all records compared to positive outcome for a particular condition above)

$$\text{gain} = 0.76 - (6/9 * 0.65) - (3/9 * 0.92)$$

$$\text{gain} = 0.02$$

Random Forests Explained

overfitting and the problem with trees

trees classify by drawing square boxes around the data, which does not work well where many separations are needed. it overfits the data

pruning

pruning means to trim (a tree, shrub, or bush) by cutting away dead or overgrown branches or stems (or unwanted parts), especially to encourage growth.

for example, if you had a single data (a leaf) that is making your square a large one, so you just remove it so that the tree still maintains relevance. improves accuracy of the tree.

random forests

just like a forest is made up of trees, similarly random forest is a machine learning method made up of tree methods. random as it randomises the input of the trees.

the basic form of it acts on tally. a tree gives out an output, this one collects all the outputs of the trees and try to make up an answer. if 10% of trees said A and 90% said B, it will say B (majority). much like you ask questions to 10 people as to what book they think will be most popular this year or what team will win. just here as the inputs are randomised, each trees are not the same. it might be further tweaked to give more or less meanings to the answers in the tree.

use of random forests

used to identify customer type, whether people will like a product or predict behaviour.

supervised learning: mixed methods

support vector machines

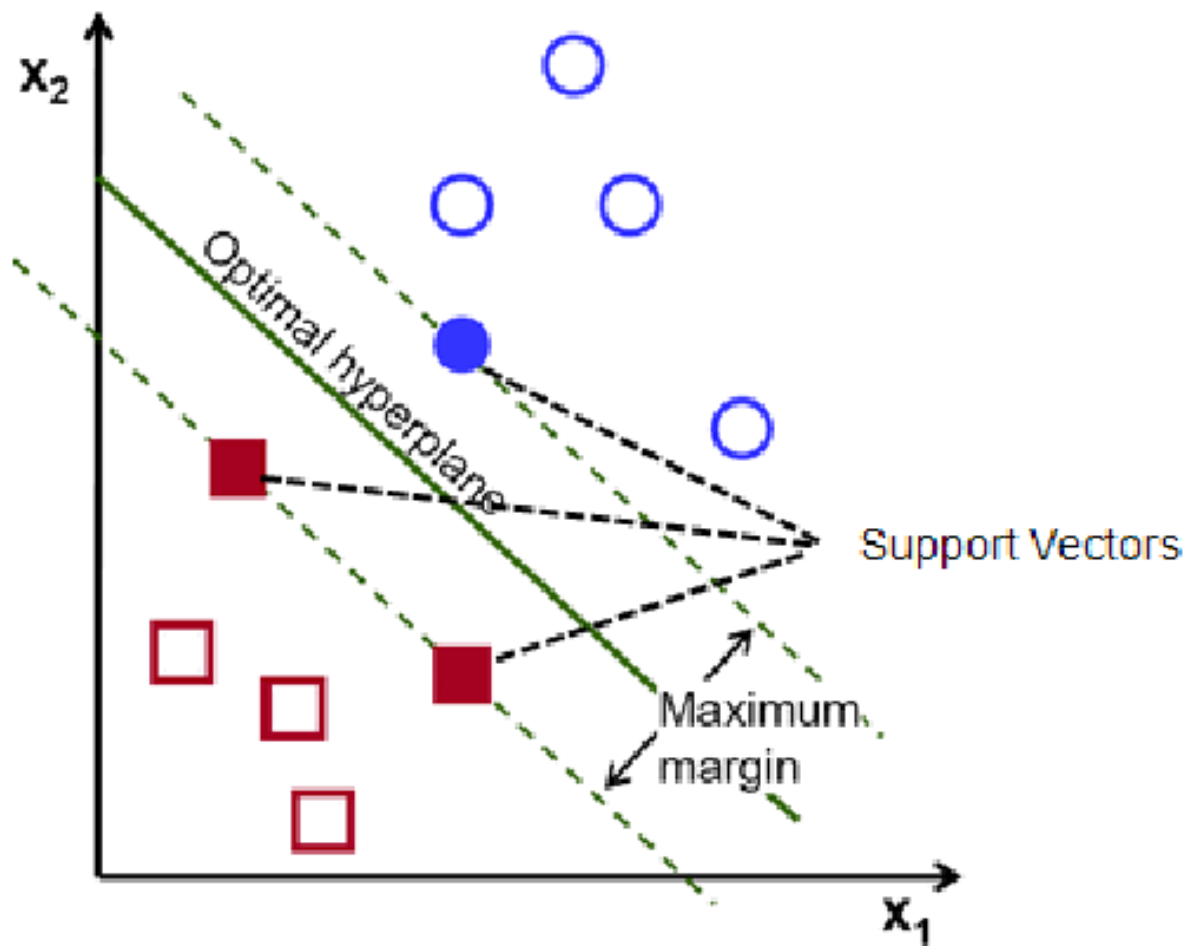
support vector machines (svm)

🔍 support vectors : read on to know

🔍 machine means model

support vector machines use hyperplanes to separate data. in a 2D plot, the line separating the data is called a separating hyperplane.

in 2D, a 1D hyperplane separates the data, in 1000D, a 999D hyperplane separates the data. so in N dimension, we need a hyperplane of $N-1$ dimension

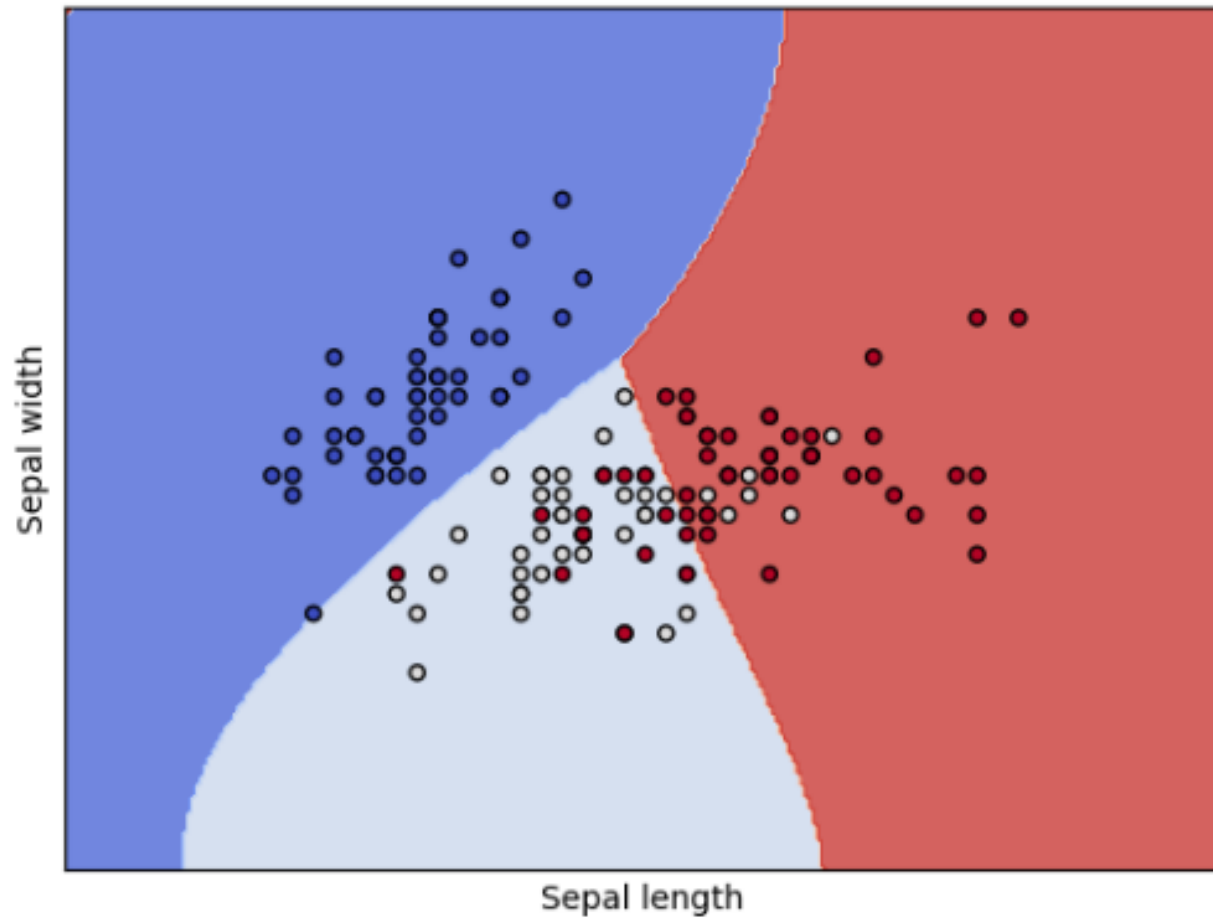


 on how to separate and how it shines

suppose we have some data scattered on the right and some on the left. we can do something like averaging distances (best fit) and drawing a line but, if we consider that the farthest a point is from our separating line, the more confident we are that we got it right (if many points close to the boundary, we would doubt as whether we got it right or not).

svm makes use of this idea. it divides the data in such a way as to maximise the distance between the line (hyperplane) and the data lying closest to the line. \sim it might not be a straight line.

however your data is separated (it needs not a straight line)



margin

the distance between the line and the closest coordinates is called margin. large margins make up for errors and limited data.

👉 🔍 support vectors are those points lying closest to the line (hyperplane) i.e. the closest coordinates

kernel trick

to separate complex shapes like a ring of data of type A surrounding data of type B appearing as a circle in the middle, SVM makes use of the kernel trick

the kernel trick transform inputs from a lower dimension (in 2D for example) to a higher dimension (like 3D) in a fashion that the data can be separated. mostly used in non-linear problems

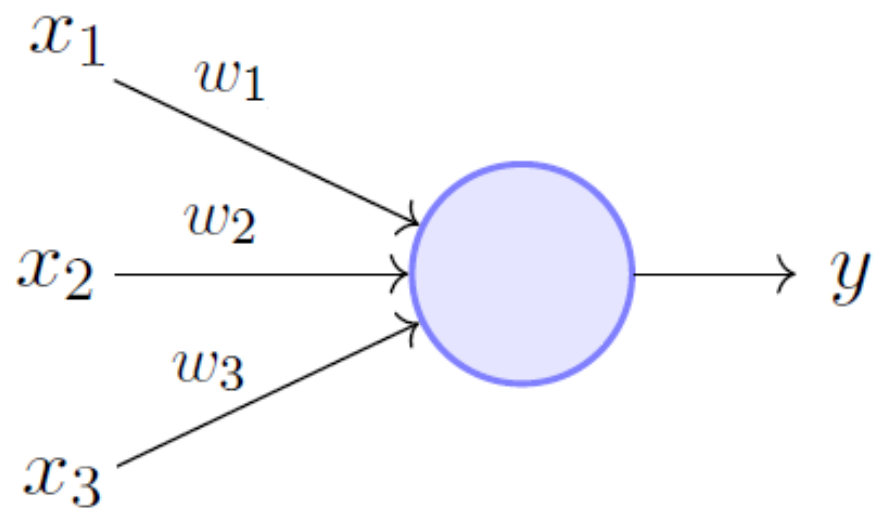
neural networks

neural networks require a bit of explanation. entire books can be written on the subject

the perceptron

the perceptron was the basis for neural networks. it consists of two inputs, a processor and an output.

the processor is just a function that decides what to output according to the inputs. it might check if $>$ a number or just check the sign or whatever



Perceptron Model (Minsky-Papert in 1969)

weight

the weight is just a value that we multiply the input by.

sum

we then sum the inputs * weight and pass them to the function.

so we can see input -> weight -> function

example

let us say our two inputs are

input1 : 5

input2 : 6

let us have some random weights

weight1 : 2

weight2 : 1

let us sum

$\text{input1} * \text{weight1} + \text{input2} * \text{weight2}$

$5 * 2 + 6 * 1$

16

let us say we configure our function as

```
def activation(sum):  
    if sum > 10:  
        return 0  
    else:  
        return 1
```

well, we just pass it to the activation function and it will return 1

 exercise:

1 find out the uses of neural networks (the fields)

hint: time series prediction, signal analysis ...

bias input

in our example above let us say that we have input1 0 and input2 0, $0 * \text{weight1} + 0 * \text{weight2}$ will always be zero, no matter what the weight. because of this we add a permanent input of 1 so that it becomes $0 * \text{weight1} + 0 * \text{weight2} + 1 * \text{weight3} == \text{weight3}$, in that case, the weight passes through the activation function

neural network

a neural network is a collection of perceptrons

training

whatever we wanted to do with a neural network, we must first adjust the weights as we gave it random weights at the beginning. first we test the perceptrons against inputs with known answer. we then compute the error (did it get the answer right or no). we then adjust weights according to the error. we repeat

error = desired output - guess output

| Desired | Guess | Error |
|---------|-------|-------|
| -1 | -1 | 0 |

etc

new weight = weight + Δ weight

Δ weight is calculated as the error multiplied by the input.

Δ weight = error * input

Therefore:

new weight = weight + error * input

learning constant

to decide at what rate we change our weight, we just use a value called learning constant. too large and we won't tune our weight correctly. too little and ... it takes a long time

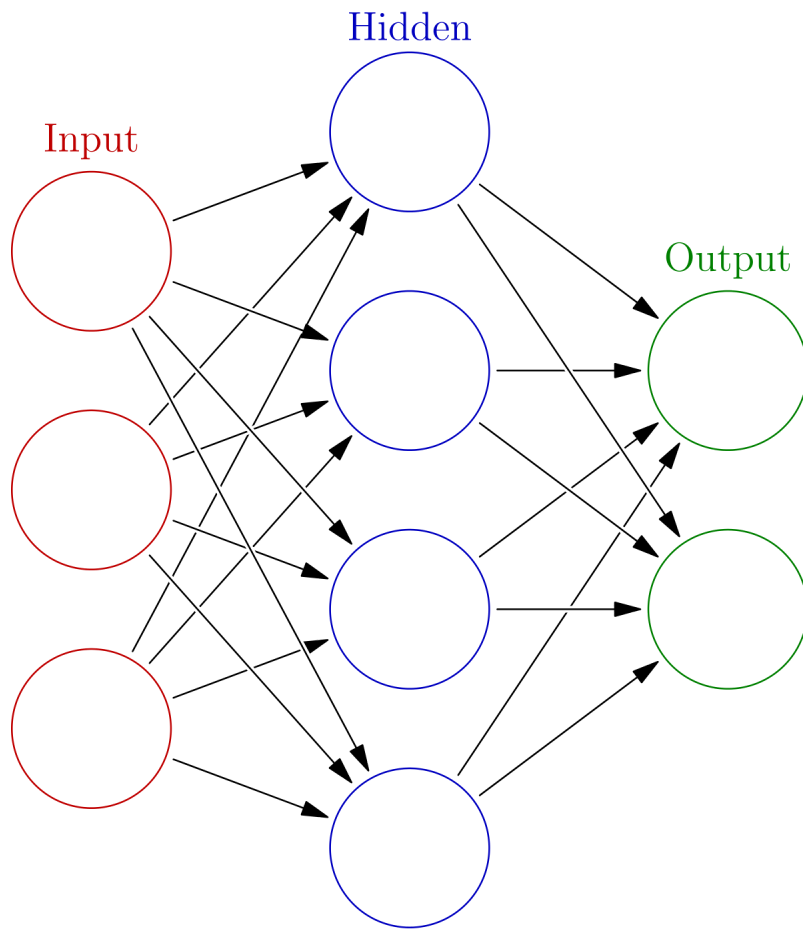
```
new weight = weight + error * input * learning constant
```

 exercise:

read implementations of neural networks from scratch in python

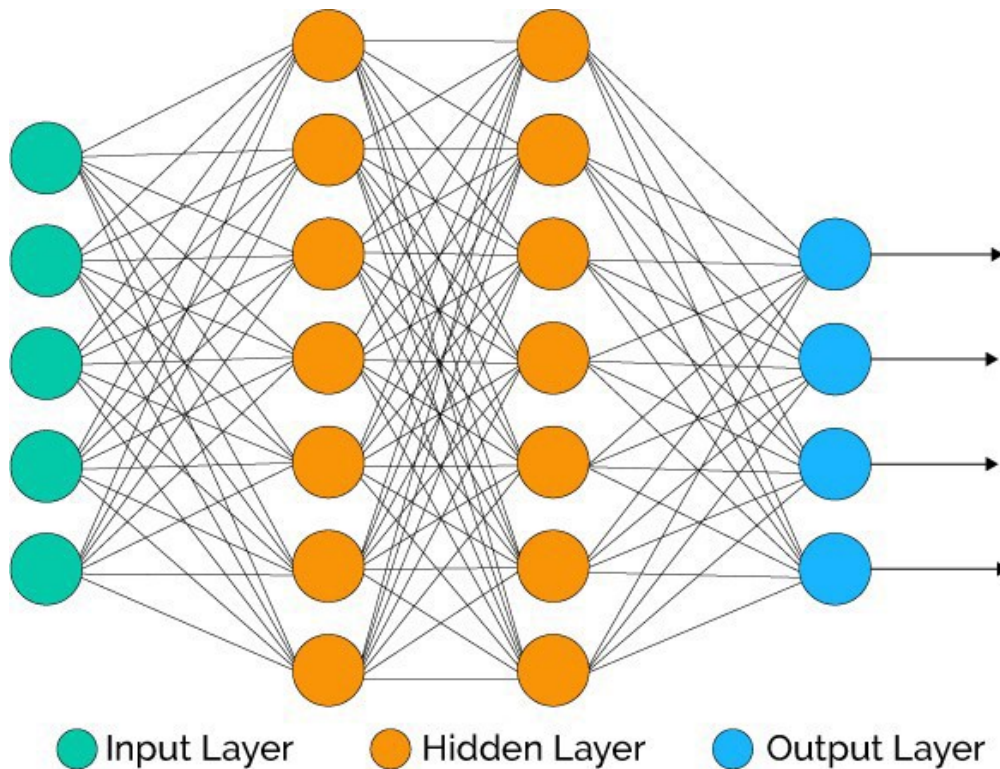
neural network

a neural network consists of many perceptrons



deep neural network

a neural network with more than one hidden layer



CNN -> see end

naive bayes


naive bayes classification

bayes theorem

bayes theorem states that

$$P(B|A) = (P(A|B) * P(B)) / P(A)$$

Probability of B given A = ...

 naive comes from the fact that features have been independently chosen from a distribution

conditional probability

that's it, checkout dependent events' probabilities. let us take a domino game. the probability taking out a piece containing one after having taken a piece containing a 6. in this example, the taking of one is affected at the very least by the fact that the tiles have been diminished by 1 ($7/28 \times 7/27$). similarly naive bayes allows us to guess events more precisely by adding known events in the game.

are you ill?

a famous use of the theorem is to find out whether you are ill given you have symptoms of a disease. the formula tells us that only looking if you have symptoms is not enough.

le us say that a new disease named scarius is out. now you check out the symptoms, brr you have those it seems ... but do you really have the disease?

$$P(\text{scarius given symptoms}) = (P(\text{symptoms given scarius}) * P(\text{scarius})) / P(\text{symptoms})$$

if symptoms is fever, $P(\text{symptoms})$ is $P(\text{fever})$ i.e. the probability that a person gets fever on any day.

$P(\text{symptoms given scarius})$ will be high since when you are ill you will get symptoms. let us say 0.96

$P(\text{scarius})$ ah let us say that disease is rare, 0.0001

$p(\text{fever})$ let us say 0.1

our P will be

$$(0.0001 \times 0.96) \div 0.1$$

$$= 0.00096$$

so, without taking in consideration those 3 probabilities, you'd make an erroneous guess

 exercise

see the theorem's derivation by using sets via a venn diagram

12 unsupervised learning

unsupervised learning is where your program has to find how the data relates to each other. there is no prior training

types of unsupervised learning

- ♠ clustering
- ♠ association

Clustering

clustering

the k means (the distance-based) algorithm is used to try to classify the data into clusters. it will find k clusters (k is the number the user provides). the centroid defines a cluster.

 centroid is a point at the center of the cluster

the how

first the centroids are randomly assigned. next each point in our dataset are assigned to a cluster. this is done by measuring nearness to the centroids. next the centroid locations are themselves updated by taking the mean value of points in the cluster

the SSE

how good was our clustering? by using the Sum of Squared Error. it is the sum of the difference between a point and the mean point in the cluster. a lower SSE means that points are closer to their centroids meaning you've got it right. we can split clusters with higher SSE into two clusters

 bisecting k means

one way to increase our clustering is to use the bisecting k means method. we choose the cluster with the largest SSE, split and repeat until you get the required number of clusters

 exercise

see an implementation in python

association analysis

finds associations between items, like what customers usually buy together in a store. if that is known, sales might be put on those items, or they could be placed together to ease the customers' life or ... why not rise the price on both?

 itemset: those items occurring together

 confidence

confidence measure how certain a rule. if clients always buy milk with eggs, we say that the confidence is 100%

$\text{confidence}(A, B) = P(B|A)$

i.e. $P(B)$ given A

* support

the support of an itemset is the percentage of the dataset that contains the itemset

let us say out of 5 transactions at a store, butter and bread was bought together 3 times. it's support is $3/5$

- with support and confidence, we can assess our association rule's success

* lift

lift is defined as

$$\text{support}(X \cup Y) / \text{support}(X) * \text{support}(Y)$$

if we are looking for milk, tea to sugar together in a dataset of 6 transactions,

$\text{support}(X \cup Y)$ means how many times over 6 they appear together, let us say it is 2

$\text{support}(X)$ means how many times milk, tea appear together in the dataset, let us say it is 4

$\text{support}(Y)$ means how many times sugar appear in the dataset, let us say 3 times

our support is

$$2/6 / 4/6 * 3/6$$

$$= 1$$

a lift of one means one has no effect on the other

if the lift is greater than one it means that one is dependent on the other (it matters)

a lift of less than one means that one adversely impacts the other

✳ a priori algorithm

this tells us that if an item set is frequent, then its subset must also be frequent. you specify a certain support level to go from as if you check each and every transaction, it is very slow!

* multilevel association rules

instead of pinpointing items, they can be associated higher up. like finding people who bought milk brand A and flour brand B together might be hard but finding that people bought milk and flour together might be easier

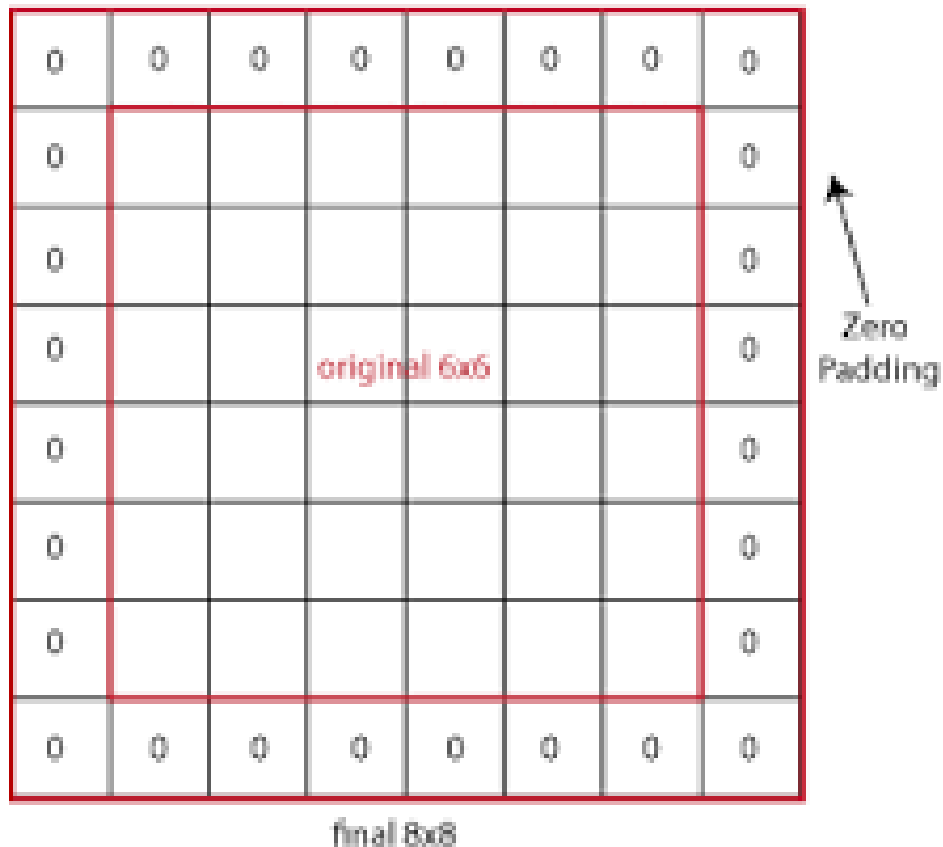
An Upgrade of the Neural Network

CNN (Convolutional Neural Network)

CNN image classifications takes an input image, process it and classify it under certain categories

padding

padding means to add some zeros around the data so as not to lose data



Convolution

| | | | | |
|---|-----------------|-----------------|-----------------|---|
| 1 | 1 _{x1} | 1 _{x0} | 0 _{x1} | 0 |
| 0 | 1 _{x0} | 1 _{x1} | 1 _{x0} | 0 |
| 0 | 0 _{x1} | 1 _{x0} | 1 _{x1} | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Image

| | | |
|---|---|--|
| 4 | 3 | |
| | | |
| | | |

Convolved
Feature

Lets have an image represented by

```
image 1 2 3 filter 1 0 output [ ][ ]
      4 3 2          0 1        [ ][ ]
      3 3 2
```

applying first

```
image 1 2 3 filter 1 0 output [ ][ ]
      4 3 2          0 1        [ ][ ]
      3 3 2
```

```
1 2 * 1 0 -> 1x1 + 2x0 + 4x0 + 3x1 = 4
4 3   0 1
```

```
output [4][ ]
        [ ][ ]
```

we shift to the right

$$\begin{array}{cc} 2 & 3 \\ 3 & 2 \end{array} * \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \rightarrow 2 \times 1 + 3 \times 0 + 3 \times 0 + 2 \times 1 = 4$$

output $\begin{bmatrix} 4 \\ \end{bmatrix} \begin{bmatrix} 4 \\ \end{bmatrix}$

then we go down

$$\begin{array}{cc} 4 & 3 \\ 3 & 3 \end{array} * \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \rightarrow 4 \times 1 + 3 \times 0 + 3 \times 0 + 3 \times 1 = 7$$

output $\begin{bmatrix} 4 \\ 7 \end{bmatrix} \begin{bmatrix} 4 \\ \end{bmatrix}$

etc

Real life convolution

| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 156 | 155 | 156 | 158 | 158 | ... |
| 0 | 153 | 154 | 157 | 159 | 159 | ... |
| 0 | 149 | 151 | 155 | 158 | 159 | ... |
| 0 | 146 | 146 | 149 | 153 | 158 | ... |
| 0 | 145 | 143 | 143 | 148 | 158 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #1 (Red)

| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 167 | 166 | 167 | 169 | 169 | ... |
| 0 | 164 | 165 | 168 | 170 | 170 | ... |
| 0 | 160 | 162 | 166 | 169 | 170 | ... |
| 0 | 156 | 156 | 159 | 163 | 168 | ... |
| 0 | 155 | 153 | 153 | 158 | 168 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #2 (Green)

| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 163 | 162 | 163 | 165 | 165 | ... |
| 0 | 160 | 161 | 164 | 166 | 166 | ... |
| 0 | 156 | 158 | 162 | 165 | 166 | ... |
| 0 | 155 | 155 | 158 | 162 | 167 | ... |
| 0 | 154 | 152 | 152 | 157 | 167 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #3 (Blue)

| | | |
|----|----|----|
| -1 | -1 | 1 |
| 0 | 1 | -1 |
| 0 | 1 | 1 |

Kernel Channel #1



161

| | | |
|---|----|----|
| 1 | 0 | 0 |
| 1 | -1 | -1 |
| 1 | 0 | -1 |

Kernel Channel #2



-9

| | | |
|---|----|---|
| 0 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | -1 | 1 |

Kernel Channel #3



659

+

+






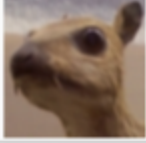

+ 1 = 812

↑
Bias = 1

Output

| | | | | |
|-----|-----|-----|-----|-----|
| -25 | 466 | 466 | 475 | ... |
| 295 | 787 | 798 | 812 | ... |
| | | | | ... |
| | | | | ... |
| ... | ... | ... | ... | ... |

application

| Operation | Filter | Convolved Image |
|----------------------------------|--|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |
| Sharpen | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| Box blur (normalized) | $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |
| Gaussian blur (approximation) | $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ |  |

Pooling

Let's say we have a convoluted feature

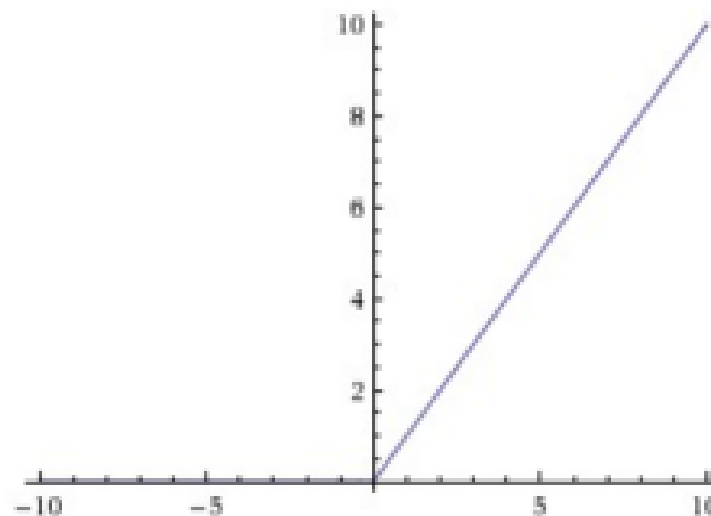
amax pooling will look out for the maximum

| | | | | | | | | | | | | |
|----|----|----|----|----|-------|----|--|----|----|----|----|----|
| 20 | 31 | 12 | 45 | | 20 | 31 | | 12 | 45 | | 89 | 45 |
| 12 | 89 | 34 | 32 | -> | 12 | 89 | | 34 | 3 | -> | 43 | 54 |
| 34 | 43 | 34 | 36 | | ----- | | | | | | | |
| 19 | 23 | 23 | 54 | | 34 | 43 | | 34 | 36 | | | |
| | | | | | 19 | 23 | | 23 | 54 | | | |

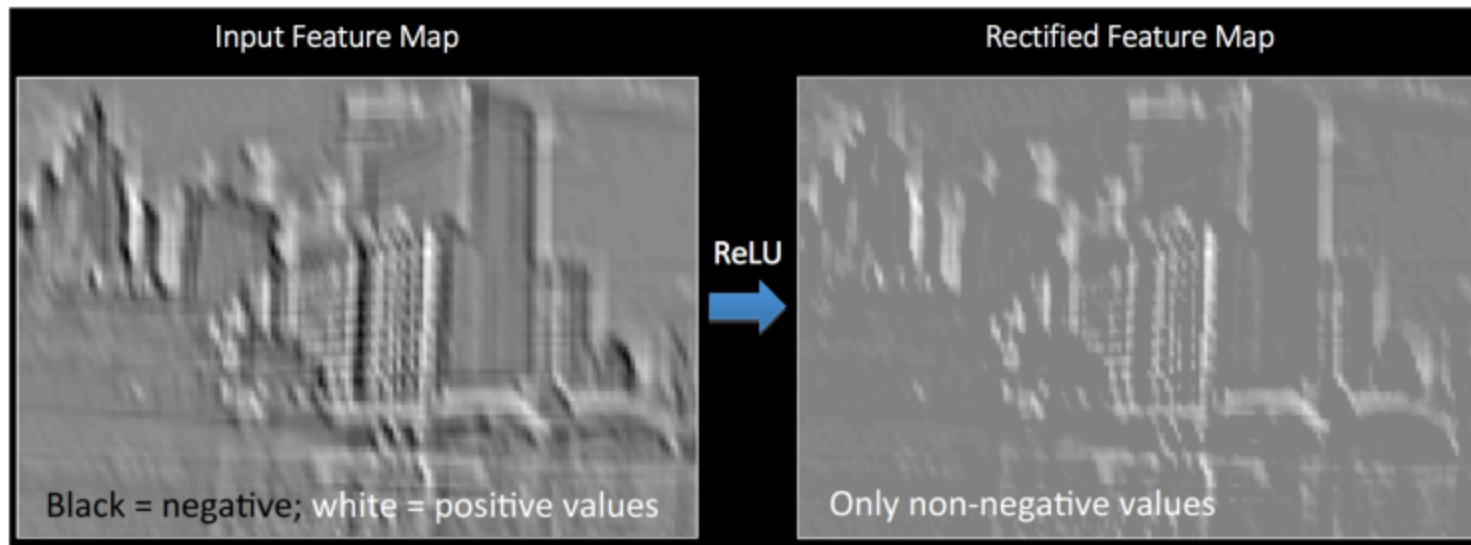
ReLU

ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

$$\text{Output} = \text{Max}(\text{zero}, \text{Input})$$



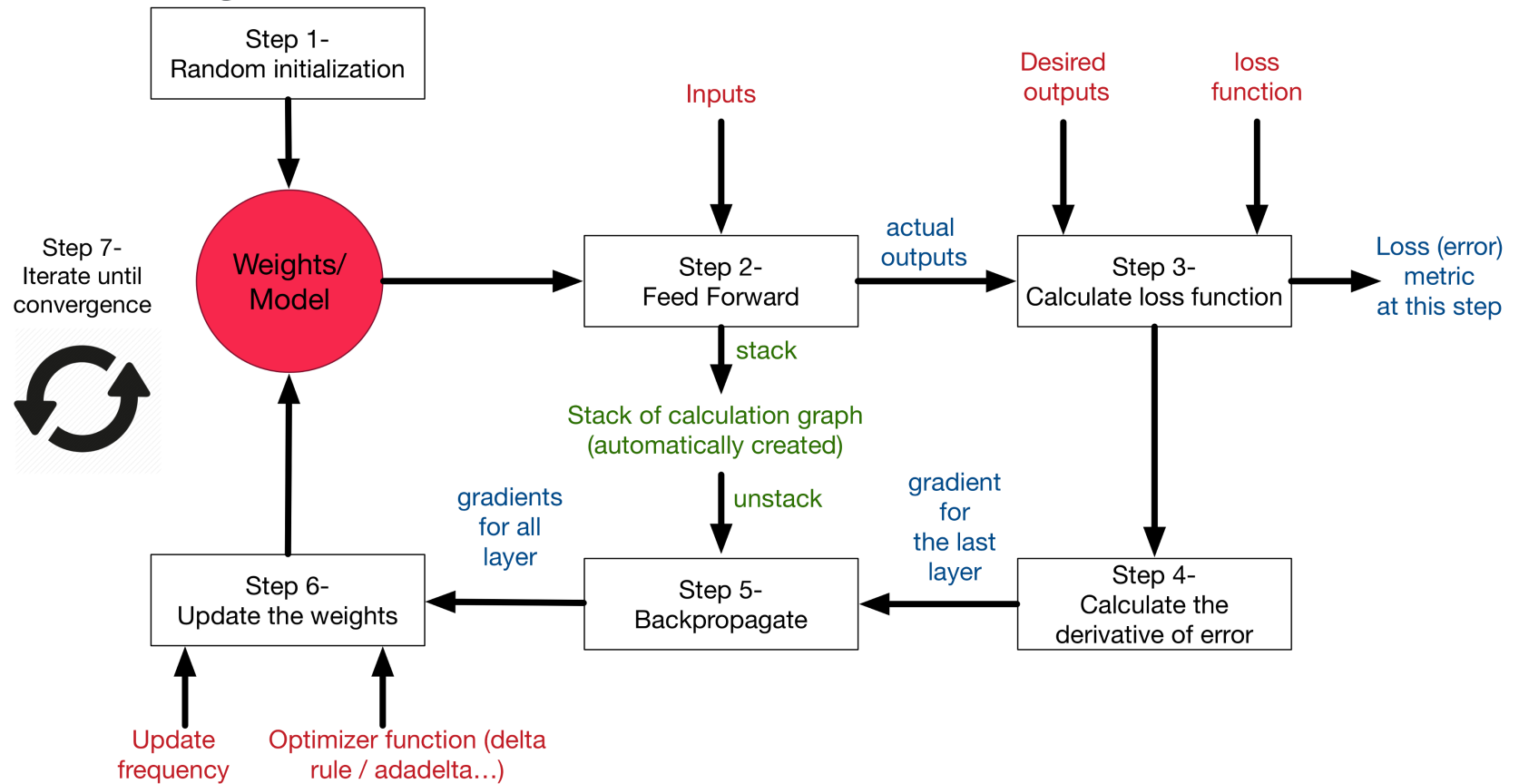
practical result



Classification — Fully Connected Layer (FC Layer)

This layer is the same as the neural network covered previously. basically takes an input volume (whatever the output is of the conv or ReLU or pool layer preceding it) and outputs an N dimensional vector where N is the number of classes that the program has to choose from. If we are classifying cats and dogs, N is two. Each number in this N dimensional vector represents the probability of a certain class. [0.6 0.4] may mean 60% chance of dog, 40% chance of cat.

How training is done



Full Overview

