

Cancer Detection Using Machine Learning

11.02.2020

Nishchay Trivedi (Student ID: 1106924)

Shiv Hansoti (Student ID: 1105615)

Motivation

9.56 Million People in the world die due to cancer. It is the second most deathly factor. Many people lose their dear ones due to it. So this is just a try to solve the problem of medical science using advanced computer science research techniques and algorithms.

Overview

In this project, we have taken the best available dataset from different resources and tried to use different machine learning and deep learning techniques for the detection of cancer and optimizing the result achieved by machine learning models.

Project Files (Python)

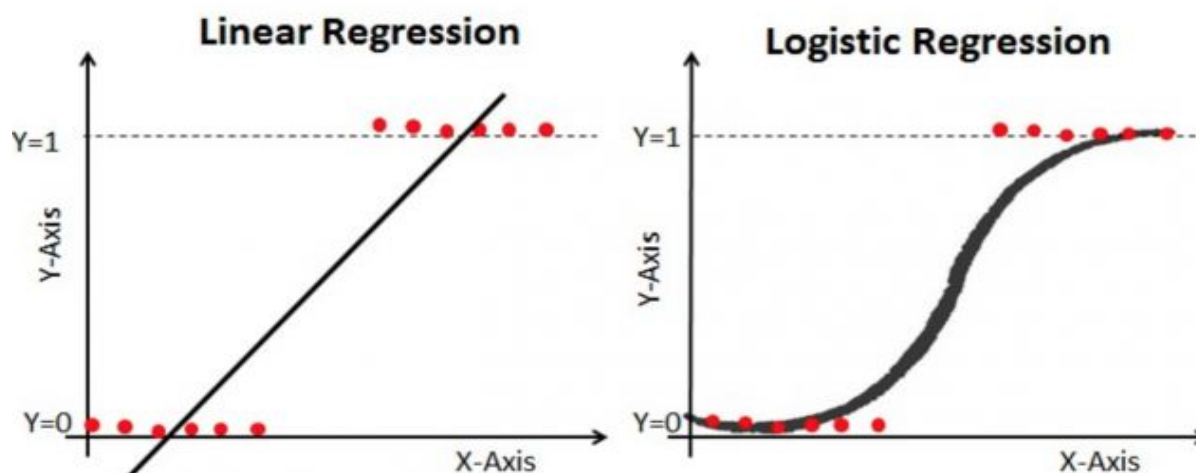
GitHub:- https://github.com/shiv96/Cancer_detection

Dataset

1. The breast cancer dataset imported from scikit-learn which contains 569 samples including 30 real, positive features (including cancer mass attributes like mean radius, mean texture, mean perimeter, etcetera). Of the samples, 212 are labeled malignant and 357 are labeled benign.
2. The original dataset consisted of 162 whole mount slide images of Breast Cancer (BCa) specimens scanned at 40x. From that, 277,524 patches of size 50 x 50 were extracted (198,738 IDC negative and 78,786 IDC positive). Each patch's file name is of the format: uxXyYclassC.png(example 10253idx5x1351y1101class0.png) , Where u is the patient ID (10253idx5), X is the x-coordinate of where this patch was cropped from, Y is the y-coordinate of where this patch was cropped from, and C indicates the class where 0 is non-IDC and 1 is IDC.

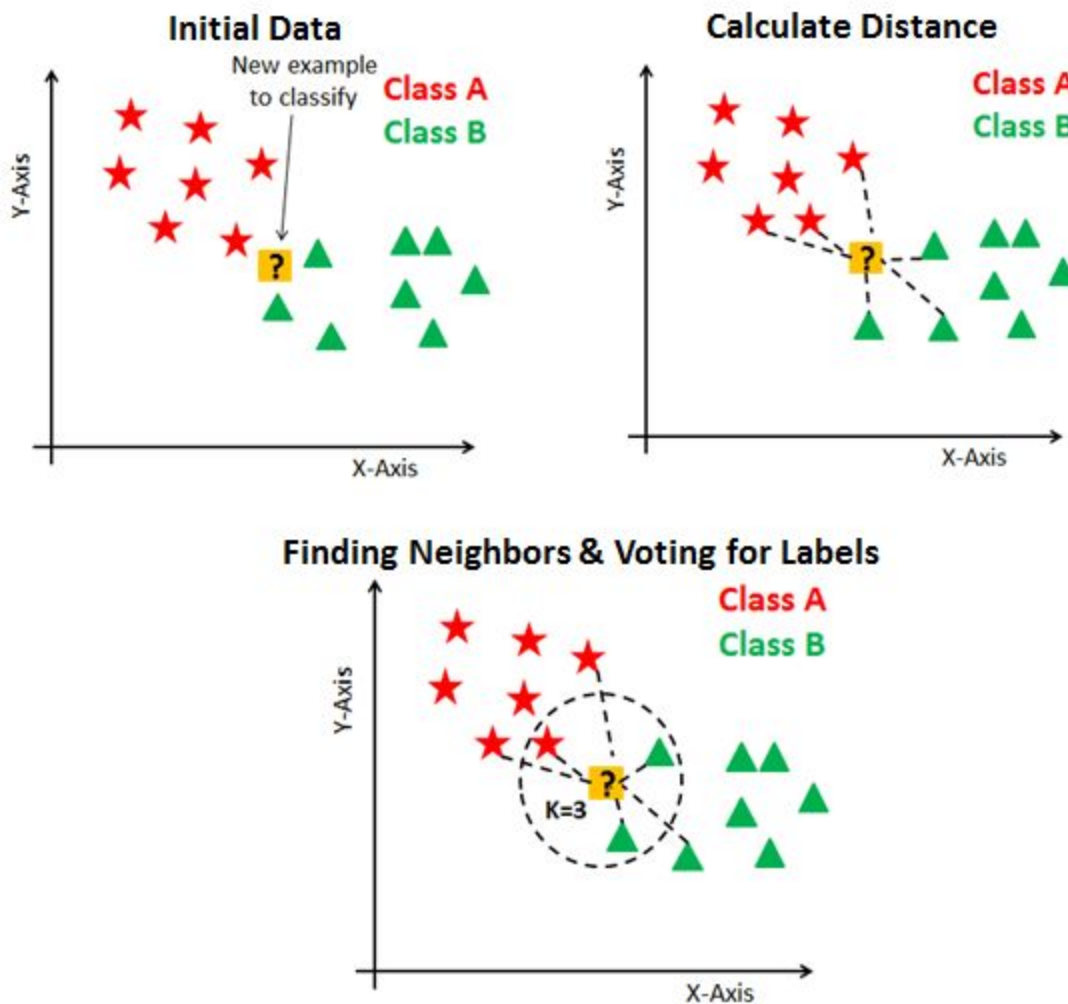
Algorithm-1 (Dataset 1)

The first approach is logistic regression for classification. Logistic regression is a statistical model that uses a logistic function to model a binary dependent variable, even though several more complex extensions do exist. Then, in regression analysis, logistic regression also known as logit regression is done to estimate the logistic model parameters, which is a form of binary regression. Moreover, a binary logistic model always has a dependent variable with two possible values, such as pass/fail, which are represented by an indicator variable. The two values here are labeled 0 and 1. Now in the logistic model, the log-odds(i.e., the logarithm of the odds) for the value which is labeled as 1 is a linear combination of one or more independent variables known as predictors. The independent variables can also be a binary variable or a continuous variable that can be any value. So, the corresponding probability of the value initially labeled as 1 can vary between 0 (value as 0) and 1 (value as 1). The function which converts the log-odds into probability here is the logistic function. The unit for measurement for the log-odds scale is called the logit, from logistic units, and hence the alternative names are used. Analogous models with a different sigmoid function in place of the logistic function can be used, like the probit model. The logistic model's defining characteristic is that when increasing one of the independent variables multiplicatively scales than the odds of the given outcome at a constant rate, with each independent variable having its parameter, for a binary dependent variable, this generalizes the odds ratio.



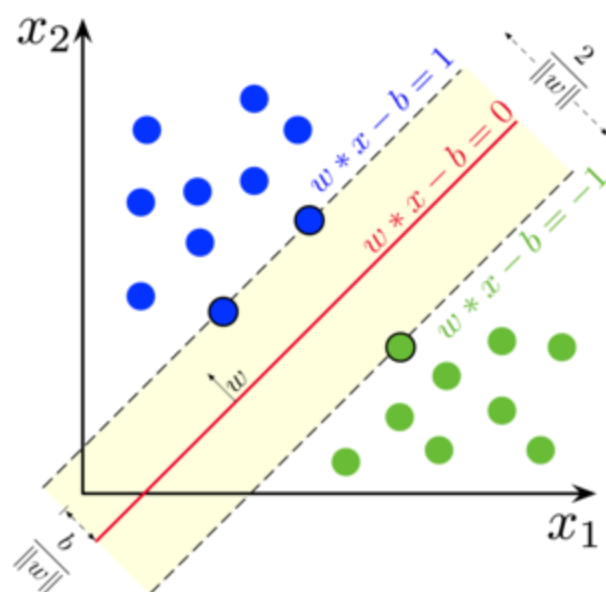
Algorithm-2 (Dataset 1)

The second approach is K-nearest neighbor for classification. In the k-NN classification, the result obtained is a binary categorized class. An item here is classified by a majority vote of its neighbors, assigning the object to the most common class of its nearest k neighbors where k is a positive integer. If $k = 1$, the object is strictly allocated to the closest single neighbor's class.



Algorithm-3 (Dataset 1)

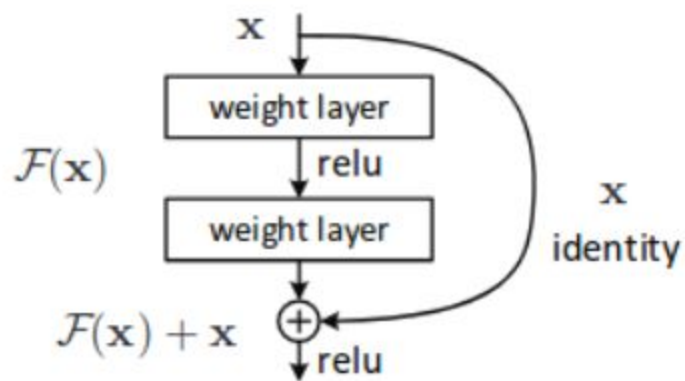
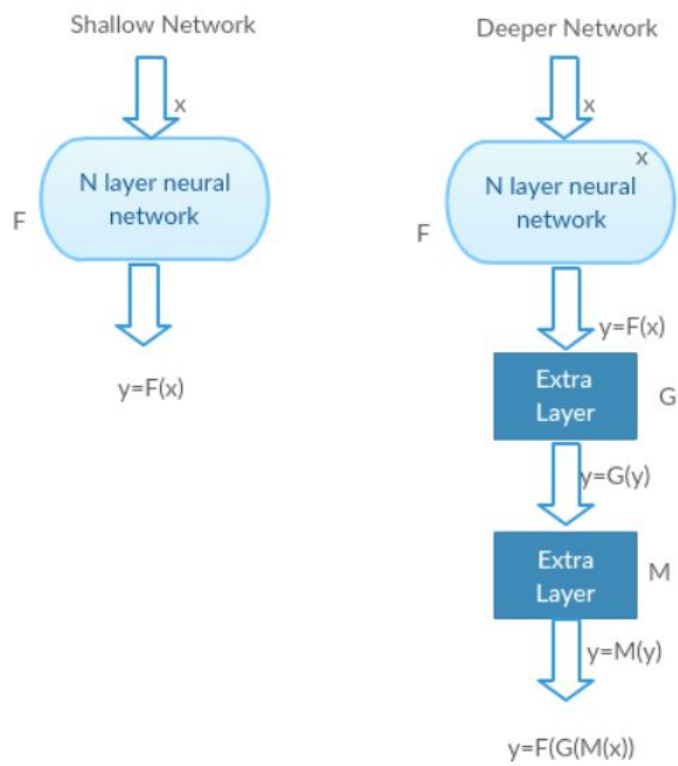
The third method of classification is the Support vector machine. The SVM model represents the samples as space points, represented in such a way that the different group examples are separated by a simple distance that is as wide as possible. Instead, new instances are drawn into the same space and expected to belong to a division depending on the side of the distance they fall through. In addition to performing linear classification, SVMs can effectively perform a nonlinear classification using what is called the kernel trick, projecting their inputs into high-dimensional feature spaces implicitly.




Algorithm-4 (Dataset 2)

The first approach used for the DCNN is ResNet V-50. A residual neural network (ResNet) is an Artificial Neural Network (ANN) of a type that builds on the constructs which are known from pyramidal cells present in the cerebral cortex. The residual neural networks do this task by utilizing skip connections, or shortcuts to jump over some of the layers. Moreover, typical ResNet models are implemented with a double- or triple- layer skips that do contain nonlinearities (ReLU) and use batch normalization in between. An additional weight matrix may also be used in order to learn to skip weights. These models are known as HighwayNets. Now, the models with several of the parallel skips are referred to as the densenets. A non-residual network can be described as a plain network when thinking in the context of residual neural networks.

When deeper networks start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly.





Instead of learning the direct mapping of $x \rightarrow y$ with a the function $H(x)$ (A few stacked non-linear layers) let us define the residual function using $F(x) = H(x) - x$, which can be reframed into $H(x) = F(x) + x$, where $F(x)$ and x represents the stacked non-linear layers and the identity function(input=output) respectively.

If the identity mapping is optimal, We can quickly push the residuals to zero ($F(x) = 0$) than to fit an identity mapping (x , input=output) by a stack of non-linear layers. In simple language, it is straightforward to come up with a solution like $F(x) = 0$ rather than $F(x) = x$ using a stack of non-linear CNN layers as a function (Think about it). So, this function $F(x)$ is what the authors called Residual function.



Modal Summary (ResNet V50)

Sequential			
=====			
Layer (type)	Output Shape	Param #	Trainable
=====			
Conv2d	[64, 24, 24]	9,408	True
<hr/>			
BatchNorm2d	[64, 24, 24]	128	True
<hr/>			
ReLU	[64, 24, 24]	0	False
<hr/>			
MaxPool2d	[64, 12, 12]	0	False
<hr/>			
Conv2d	[64, 12, 12]	4,096	True
<hr/>			
BatchNorm2d	[64, 12, 12]	128	True
<hr/>			



Conv2d	[64, 12, 12]	36,864	True
--------	--------------	--------	------

BatchNorm2d	[64, 12, 12]	128	True
-------------	--------------	-----	------

Conv2d	[256, 12, 12]	16,384	True
--------	---------------	--------	------

BatchNorm2d	[256, 12, 12]	512	True
-------------	---------------	-----	------

ReLU	[256, 12, 12]	0	False
------	---------------	---	-------

Conv2d	[256, 12, 12]	16,384	True
--------	---------------	--------	------

BatchNorm2d	[256, 12, 12]	512	True
-------------	---------------	-----	------

Conv2d	[64, 12, 12]	16,384	True
--------	--------------	--------	------

BatchNorm2d	[64, 12, 12]	128	True
-------------	--------------	-----	------

Conv2d	[64, 12, 12]	36,864	True
--------	--------------	--------	------

BatchNorm2d	[64, 12, 12]	128	True
-------------	--------------	-----	------

Conv2d	[256, 12, 12]	16,384	True
--------	---------------	--------	------

BatchNorm2d	[256, 12, 12]	512	True
-------------	---------------	-----	------

ReLU	[256, 12, 12]	0	False
------	---------------	---	-------

Conv2d	[64, 12, 12]	16,384	True
--------	--------------	--------	------

BatchNorm2d	[64, 12, 12]	128	True
-------------	--------------	-----	------

Conv2d	[64, 12, 12]	36,864	True
--------	--------------	--------	------

BatchNorm2d	[64, 12, 12]	128	True
-------------	--------------	-----	------

Conv2d	[256, 12, 12]	16,384	True
--------	---------------	--------	------

BatchNorm2d	[256, 12, 12]	512	True
-------------	---------------	-----	------

ReLU	[256, 12, 12]	0	False
------	---------------	---	-------

Conv2d	[128, 12, 12]	32,768	True
--------	---------------	--------	------

BatchNorm2d	[128, 12, 12]	256	True
-------------	---------------	-----	------

Conv2d	[128, 6, 6]	147,456	True
--------	-------------	---------	------

BatchNorm2d	[128, 6, 6]	256	True
-------------	-------------	-----	------

Conv2d	[512, 6, 6]	65,536	True
--------	-------------	--------	------

BatchNorm2d	[512, 6, 6]	1,024	True
-------------	-------------	-------	------

ReLU	[512, 6, 6]	0	False
------	-------------	---	-------

Conv2d	[512, 6, 6]	131,072	True
--------	-------------	---------	------



BatchNorm2d	[512, 6, 6]	1,024	True
-------------	-------------	-------	------

Conv2d	[128, 6, 6]	65,536	True
--------	-------------	--------	------

BatchNorm2d	[128, 6, 6]	256	True
-------------	-------------	-----	------

Conv2d	[128, 6, 6]	147,456	True
--------	-------------	---------	------

BatchNorm2d	[128, 6, 6]	256	True
-------------	-------------	-----	------

Conv2d	[512, 6, 6]	65,536	True
--------	-------------	--------	------

BatchNorm2d	[512, 6, 6]	1,024	True
-------------	-------------	-------	------

ReLU	[512, 6, 6]	0	False
------	-------------	---	-------

Conv2d	[128, 6, 6]	65,536	True
--------	-------------	--------	------

BatchNorm2d	[128, 6, 6]	256	True
-------------	-------------	-----	------

Conv2d	[128, 6, 6]	147,456	True
--------	-------------	---------	------


BatchNorm2d	[128, 6, 6]	256	True
-------------	-------------	-----	------

Conv2d	[512, 6, 6]	65,536	True
--------	-------------	--------	------

BatchNorm2d	[512, 6, 6]	1,024	True
-------------	-------------	-------	------

ReLU	[512, 6, 6]	0	False
------	-------------	---	-------

Conv2d	[128, 6, 6]	65,536	True
BatchNorm2d	[128, 6, 6]	256	True
Conv2d	[128, 6, 6]	147,456	True
BatchNorm2d	[128, 6, 6]	256	True
Conv2d	[512, 6, 6]	65,536	True
BatchNorm2d	[512, 6, 6]	1,024	True
ReLU	[512, 6, 6]	0	False
Conv2d	[256, 6, 6]	131,072	True
BatchNorm2d	[256, 6, 6]	512	True
Conv2d	[256, 3, 3]	589,824	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[1024, 3, 3]	262,144	True
BatchNorm2d	[1024, 3, 3]	2,048	True
ReLU	[1024, 3, 3]	0	False



Conv2d	[1024, 3, 3]	524,288	True
--------	--------------	---------	------

BatchNorm2d	[1024, 3, 3]	2,048	True
-------------	--------------	-------	------

Conv2d	[256, 3, 3]	262,144	True
--------	-------------	---------	------

BatchNorm2d	[256, 3, 3]	512	True
-------------	-------------	-----	------

Conv2d	[256, 3, 3]	589,824	True
--------	-------------	---------	------

BatchNorm2d	[256, 3, 3]	512	True
-------------	-------------	-----	------

Conv2d	[1024, 3, 3]	262,144	True
--------	--------------	---------	------

BatchNorm2d	[1024, 3, 3]	2,048	True
-------------	--------------	-------	------

ReLU	[1024, 3, 3]	0	False
------	--------------	---	-------

Conv2d	[256, 3, 3]	262,144	True
--------	-------------	---------	------

BatchNorm2d	[256, 3, 3]	512	True
-------------	-------------	-----	------


Conv2d	[256, 3, 3]	589,824	True
--------	-------------	---------	------

BatchNorm2d	[256, 3, 3]	512	True
-------------	-------------	-----	------

Conv2d	[1024, 3, 3]	262,144	True
--------	--------------	---------	------

BatchNorm2d	[1024, 3, 3]	2,048	True
-------------	--------------	-------	------

ReLU	[1024, 3, 3]	0	False
Conv2d	[256, 3, 3]	262,144	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[256, 3, 3]	589,824	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[1024, 3, 3]	262,144	True
BatchNorm2d	[1024, 3, 3]	2,048	True
ReLU	[1024, 3, 3]	0	False
Conv2d	[256, 3, 3]	262,144	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[256, 3, 3]	589,824	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[1024, 3, 3]	262,144	True
BatchNorm2d	[1024, 3, 3]	2,048	True



ReLU	[1024, 3, 3]	0	False
Conv2d	[256, 3, 3]	262,144	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[256, 3, 3]	589,824	True
BatchNorm2d	[256, 3, 3]	512	True
Conv2d	[1024, 3, 3]	262,144	True
BatchNorm2d	[1024, 3, 3]	2,048	True
ReLU	[1024, 3, 3]	0	False
Conv2d	[512, 3, 3]	524,288	True
BatchNorm2d	[512, 3, 3]	1,024	True
Conv2d	[512, 2, 2]	2,359,296	True
BatchNorm2d	[512, 2, 2]	1,024	True
Conv2d	[2048, 2, 2]	1,048,576	True
BatchNorm2d	[2048, 2, 2]	4,096	True
ReLU	[2048, 2, 2]	0	False

Conv2d	[2048, 2, 2]	2,097,152	True
--------	--------------	-----------	------

BatchNorm2d	[2048, 2, 2]	4,096	True
-------------	--------------	-------	------

Conv2d	[512, 2, 2]	1,048,576	True
--------	-------------	-----------	------

BatchNorm2d	[512, 2, 2]	1,024	True
-------------	-------------	-------	------

Conv2d	[512, 2, 2]	2,359,296	True
--------	-------------	-----------	------

BatchNorm2d	[512, 2, 2]	1,024	True
-------------	-------------	-------	------

Conv2d	[2048, 2, 2]	1,048,576	True
--------	--------------	-----------	------

BatchNorm2d	[2048, 2, 2]	4,096	True
-------------	--------------	-------	------

ReLU	[2048, 2, 2]	0	False
------	--------------	---	-------

Conv2d	[512, 2, 2]	1,048,576	True
--------	-------------	-----------	------

BatchNorm2d	[512, 2, 2]	1,024	True
-------------	-------------	-------	------

Conv2d	[512, 2, 2]	2,359,296	True
--------	-------------	-----------	------

BatchNorm2d	[512, 2, 2]	1,024	True
-------------	-------------	-------	------

Conv2d	[2048, 2, 2]	1,048,576	True
--------	--------------	-----------	------

BatchNorm2d	[2048, 2, 2]	4,096	True
-------------	--------------	-------	------

ReLU	[2048, 2, 2]	0	False
------	--------------	---	-------

AdaptiveAvgPool2d	[2048, 1, 1]	0	False
-------------------	--------------	---	-------

AdaptiveMaxPool2d	[2048, 1, 1]	0	False
-------------------	--------------	---	-------

Flatten	[4096]	0	False
---------	--------	---	-------

BatchNorm1d	[4096]	8,192	True
-------------	--------	-------	------

Dropout	[4096]	0	False
---------	--------	---	-------

Linear	[512]	2,097,664	True
--------	-------	-----------	------

ReLU	[512]	0	False
------	-------	---	-------

BatchNorm1d	[512]	1,024	True
-------------	-------	-------	------

Dropout	[512]	0	False
---------	-------	---	-------

Linear	[2]	1,026	True
--------	-----	-------	------

Total params: 25,615,938

Total trainable params: 25,615,938

Total non-trainable params: 0

Optimized with 'torch.optim.adam.Adam', betas=(0.9, 0.99)

Using true weight decay as discussed in
<https://www.fast.ai/2018/07/02/adam-weight-decay/>

Loss function : CrossEntropyLoss

=====

Callbacks functions applied

MixedPrecision

Results

I. Dataset 1

ML Algorithm	Cancer Type	Dataset Type	Accuracy
Logistic Regression	Breast Cancer	Numeric	79%
KNN	Breast Cancer	Numeric	91%
SVM	Breast Cancer	Numeric	88%

II. Dataset 2

DCNN	Cancer Type	Dataset Type	Accuracy
ResNet V50	Breast Cancer	Image	84%