# Assignment 3: Introduction to Big Data

*Yotam Lifschytz 209579077; Pan Eyal 208722058*

1. Added to ZIP file.

Chart, line chart

Description automatically generated

* 1. As we can see from the plot, the algorithm was able to differentiate between 0 and 1 quite good already from 1000 sample points with error.

The error on 3 and 5 is bigger, (We can see that it is noisier then the received error graph on 0 and 1.

A screenshot of a computer

Description automatically generated with low confidenceLogo

Description automatically generated with low confidence

Every pixel in each heatmaps represents respectively the probability it will lit if the label was one, and if the label was 0. Therefore, the images show the ‘average’ of all the ones and zeros images. We can notice that the 1 image has high ‘heat’ on the middle, but on the edge, it gets smudged as the 1-digit might skew, 0-digit image is more stable as it is more symmetric.

* 1. We received the following result:

percent of the 0\_1 set that their label changed from -1 to 1: 0.0

percent of the 0\_1 set that their label changed from 1 to -1: 0.0

percent of the 3\_5 set that their label changed from -1 to 1: 0.008937960042060988

percent of the 3\_5 set that their label changed from 1 to -1: 0.0

We can explain those results as:

Therefore, the only difference of will occur if the sign will flip. Because only the parameter changes, it means that will become larger or smaller then .

parameter value as calculated by the code resulted around .

Thus, . Now,

This amount can be neglect compared to:

As in each sum we will go roughly around half of the features.

The more ‘certain’ the algorithm is, the larger of these sums absolute value get.

Therefore, we can see that for the 0-1 set (which achieved very low error) no changes in labels had occurred, while on the less accurate 3-5 results, only a minor change () had occurred.

1. The hypothesis class of homogenous linear predictors is defined as:

Where we have:

We will use the “vector” version of as seen in class, where we assume an ordered where .

(BTW this is all going to be indexed horrendously since we were told in the forum to not use a “layered” version of a NN unless we show equivalence to the “vector” version we saw in class, so sorry in advance).

* 1. We define a neural network with a sized input layer where the input is . We have no “hidden layers”, just an output layer of size 1. The output is, by definition:

Let us define a vector of dimension where:

Thus:

We denote the hypothesis class this NN represents as .

So, as seen above:

* 1. Say we have , and 1 hidden layer of size . Thus:

Let us denote :

Thus:

Let us denote the hypothesis class this NN represents as . First, we shall show that :

Let there be some Let’s say we have a NN with the same architecture as above. If we set one of the output weights (those that appear in the term above) for some index as , and the others as , we have:

Thus, if we set the chosen weights as then we have:

Now we must prove :

Let us set all the output weights uniformly as .

We now set ’s s.t . We change indexes

, and we now have:

This is a function that sums the signs of the coordinates of , and outputs iff the result is , i.e., iff there are more positive coordinates in . Otherwise, it returns . Let’s assume by contradiction that there exists a s.t iff there are more positive coordinates then negative, otherwise . We have:

For we have:

And this is true also for .

For where we have (more positive coordinates):

is arbitrary, so we must have this for all

In contradiction with the fact that .

So, there cannot be a s.t. acts as stated. Contradiction.

So:

1. We assume we are restricted to using test functions of the form
   1. First, creating a tree of the form:

-1

1

-1

1

1

0

1

0

1

0

We can easily see that we return iff , otherwise we return . So, we have a tree of depth that perfectly classifies the sample.

Can we do the same for a tree of smaller depth (depth of )?

Obviously not. If for some :

0

1

-1

1

Obviously, this will trivially not work – all coordinates are independent, thus is totally irrelevant, and also, we can have for example and have this test on our 1 node, with or and no way of knowing if without conducting a test on (this is the definition of independence, basically). So, we will have a completely random answer in the above example. Thus, we conclude that depth does not work and we need a depth of at least .

* 1. First, we will observe that if every possible example appears the same number of times, we can w.l.o.g look at the case of the sample where all examples appear once. In this case as half of the examples hold and the other half does not.

Let’s see how the greedy algorithm will work using the

function,

where

that has been defined in class:

Iteration 1:

Now, for each :

And therefore, for each ,

Let assume the returned

Iteration 2: (In the left node where )

And therefore, for each ,

Let assume the returned

Iteration 3: (In the right node where )

And therefore, for each ,

Let assume the returned

We received a tree with depth of 2 that each of its leaves has an error of

Therefore, the error of the decision tree that is received from this run, achieved an error of

1. Our objective function defined as: . Let us decompose it to linear and quadratic parts:

does not depend on and therefore can be neglected on optimization.

Quadratic programming: s.t.

Let us define . And now,

Linear part:

For : Let

And for : Let

So:

In this way:

We will need the following restrictions to preserve the absolute value:

and

Therefore:

For : Let

For : Let

So:

And

Quadratic part:

Let us define:

For : Let

For : Let

, and now, if we will look on the matrix block we will receive:

And all other elements for will be zeros.

So:

* 1. Because there is a unique solution for , when rows are the given examples from training sample , we can deduce that is invertible. If it wasn’t, then multiple solutions could have been found.

We know that invertible matrices are full ranked.

We also know that for any matrix .

Therefore,

* 1. Let is the optimal solution to the linear regression problem on ,

The set of optimal solutions to the linear regression problem on is:

Proof:

Because every , for every :

Let us mark the last equation as

Let there be

From :

Now, because is the optimal solution to the linear regression problem on , for every

And now again from :

Overall, we received that:  for every and every . Therefore: is the optimal solution.

1. (All relevant python code will be attached in the zip file)
   1. For

We will build:

The python code will calculate ,

and then summarize the smallest eigenvalues that returned from function.

The distortion that received from the algorithm is: 4.151633772242231

* 1. We will use the largest k eigenvalues and correspondence eigen vectors that we found in (a.) of matrix and stack them on top of each other in descending order from largest eigen vector to smallest. Then we will normalize them (each row of the matrix) to receive .

The result that received from the algorithm is:

* 1. The restored matrix will be calculated by the algorithm in the following way:

The restored that received from the algorithm is:

Meaning that:

After using the function: we will receive that the distortion is: 4.1516337722422545. It is similar to the answer we got in (a.), that is because: , as we have seen in class.

* 1. Single linkage clustering does not satisfy the scale invariance axiom.

Proof:

We will show it with a counter example:

For , and , the single linkage algorithm will choose to merge cluster and cluster {1} as and therefore

But, for , where , we will receive that the new distance between cluster {0} and cluster {1} is: and therefore, no merging will occur and .

Because the invariance axiom does not hold.

* 1. Single linkage clustering does satisfy the richness axiom.

Proof:

Let be a valid partition of .

We will define

Firstly, we will show that holds the symmetric, positive and triangle inequality axioms.

Let there be :

Symmetric:

* If , then
* If andare both in cluster , then
* Else, andare not in the same cluster,  
  and then

Positive:

* , and therefore:

Triangle inequality:

Let there be

* If , then , as is positive.
* If andare both in cluster , then:
  + w.l.o.g, if :

* + If then

* + If then

* Else, andare not in the same cluster, then:
  + w.l.o.g, if :

* + w.l.o.g, if and are in the same cluster,

* + If all are not in the same cluster,

Now, we will show that the algorithm output will be .

As we seen in class, for graph , where and  
 the returned clusters from Single-linkage method will be the connected components of .

Therefore, we need to show that for every : and that for every where .

Let there be , thus, by the definition of as declared above, . Therefore and will be classified together.

Let there be where thus, by the definition of as declared above, . Therefore and will not be classified together.