Assignment one

Machine Learning

Sigve Skaugvoll, MIT, H2018

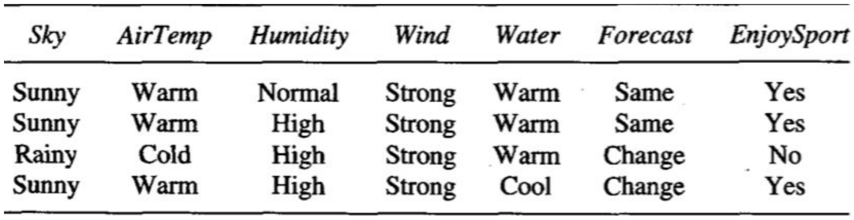
Theory [1.5 point]

1. What is concept learning, explain with an example

“Inferring a boolean-valued function from training examples of its input and output”
In other words; Automatically inferring the general definition of some concept, given examples labeled as members or nonmembers of the concept.

Thus concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation, where the goal is to find the hypothesis that best fits the training examples.

An example:
Given a dataset EnjoySport with entities containing 6 features (with 3,2,2,2,2,2 different possible values) and one class. We want to teach the model to find out what combination of feature values (hypothesis) represent the different classes (yes / no), so that the model can predict the class of unseen entities.



Sky	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

So what hypothesis should we use to predict the class EnjoySport?
If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a Positive. We can see that all entities that have AirTemp = Warm gives Yes. We can also see that Sky = Sunny gives Yes, and that Rainy and cold gives no. Thus we can create a hypothesis that classifies unknown entities to yes if (Sunny, Warm, ?, Strong, ?, ?) - ? represents that the feature can have any value is acceptable for this attribute, meaning that the model has learned that hypothesis where all possible values for that attribute is allowed / gives a class of yes.

2. What is function approximation and why do we need them?

Target function is a function which learns to choose among the legal “moves” an model can use. This function accepts as input some state representation and produces as output some action / move / the next state.

In many problems it’s difficult to learn an algorithm which always finds the optimal action Everytime. Thus we need function approximation because we often acquire learning algorithms to acquire only some approximation to the target function, and for this reason the process of learning the target function is often called function approximation.

It is a function that is learned by the algorithm.

3. What is inductive bias in the context of machine learning, and why is it so important? Decision tree learning and the candidate elimination algorithm are two different learning algorithms. What can you say about the inductive bias for each of them?

What is the policy by which ID3 generalizes from observed training examples to classify unseen instances?

Inductive bias is a term used to refer to a policy which describes how the algorithm generalizes from observed training examples to classify unseen instances. All it does is add additional assumptions, so that something can be provable from something else. (i.e if z follows follows deductively from y, this means that z is provable from y). The Additional assumptions (inductive bias) makes it sufficient to justify its inductive inferences as deductive inferences. The inductive bias is important to allow for further inferences - learning beyond training examples.

Def:

Consider a concept learning algorithm ***L*** for the set of instances X. Let c be an arbitrary concept defined over X, and let ***D,*** = ***((x,c(x))}***be an arbitrary set of training examples of c. Let ***L(xi,D,)*** denote the classification assigned to the instance ***xi*** by ***L*** after training on the data ***D,.*** The **inductive bias** of ***L*** is any minimal set of assertions ***B*** such that for any target concept c and corresponding training examples ***Dc***

(Vxi **E** ***X)[(B*****A** ***Dc A xi)*** k ***L(xi, D,)]***

a. Decision tree

It chooses the first acceptable tree it encounters in its simple-to-complex, hill- climbing search through the space of possible trees. Roughly speaking, then, the ID3 search strategy (a) selects in favor of shorter trees over longer ones, and (b) selects trees that place the attributes with highest information gain closest to the root.

Giving us two different approximations of the IB:

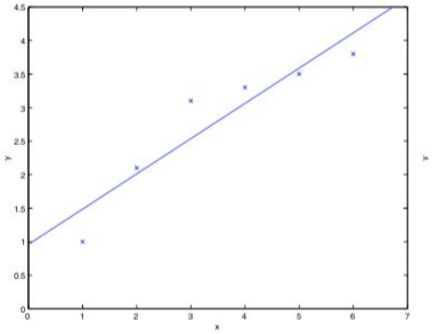
a. **Approximate inductive bias of ID3:** Shorter trees are preferred over larger trees.
b. **A closer approximation to the inductive bias of ID3:** Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

b. Candidate elimination algorithm

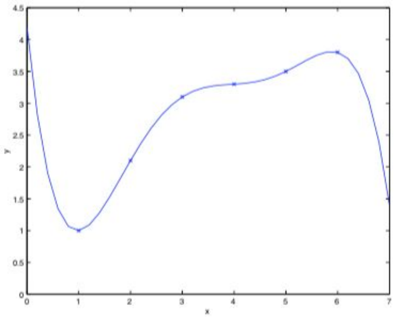
It is simply the assumption ***c*** ∈ H. Given this assumption, each inductive inference performed by the CANDIDATE-ELIMINATION algorithm can be justified deductively.

“The target concept c is contained in the given hypothesis space H”

4. What is overfitting, and how does it differ from underfitting? Briefly explain what a validation set is. How can cross-validation be used to mitigate overfitting?



Underfitting - in which the data clearly shows structure not captured by the model.



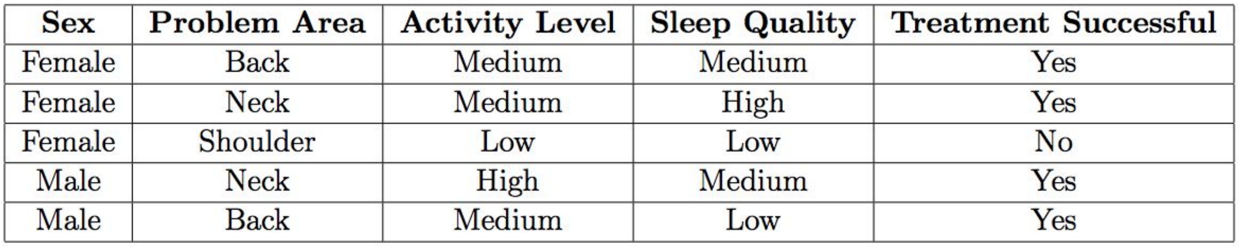
Overfitting - Clearly shows capturing a way to structure the data, which is way to specific, it will not work very well for untrained entities.

Overfitting decreases generalization accuracy over unseen examples/

Validation set is used to evaluate the accuracy of the trained/learned over subsequent data and, in particular, to evaluate the impact of pruning. The motivation is; Even though the learner may be misled by random errors and coincidental regularities with in the training set, the validation set is unlikely to exhibit the same random fluctuations. Therefor the validation set can be expected to provide a safety check against overfitting the spurious characteristics of the training set. It is important that the validation set be large enough to itself provide a statistically significant sample of the instances. One common heuristic is to withhold one-third of the available examples for the validation set.

Cross-validation and mitigating overfitting: One of the most successful methods for overcoming the overfitting. problem is to simply provide a set of validation data to the algorithm in addition to the training data. The algorithm monitors the error with respect to this validation set, while using the training set to drive the gradient descent search. In essence, this allows the Algorithm itself to plot the two curves shown in Figure 4.9. How many weight-tuning iterations should the algorithm perform? Clearly, it should use the number of iterations that produces the lowest error over the validation set, since this is the best indicator of network performance over unseen examples

5. Apply candidate elimination (CE)



Sex	Problem Area	Activity Level	Sleep Quality	Treatment Successful
Female	Back	Medium	Medium	Yes
Female	Neck	Medium	High	Yes
Female	Shoulder	Low	Low	No
Male	Neck	High	Medium	Yes
Male	Back	Medium	Low	Yes

Attribute : possible values
Sex : 2
Problem Area : 3
Activity Level : 3
Sleep Quality: 3
Treatment Successful: 2

4 * 5 * 5 * 5 = 500 hypothesis (including ø and ?)

Describe version space, specific and general boundary.
Version space: The **version space,** denoted $VS_{H,D}$ *with* respect to hypothesis space **H** and training examples D, is the subset of hypotheses from **H** consistent with the training examples in D.

The candidate-elimination algorithm represents the set of all hypotheses consistent (fulfilled) with the observed training examples

The algorithm starts with computing the version space containing all hypotheses from H that are consistent with an observed sequence of training examples. It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain the most general hypothesis in H, and initializing the S boundary set to contain the most specific

G = {<?,?,?,?>}
S = {<ø,ø,ø,ø>}

D1 = <Female, Back, Medium, Medium> (+)

- Nothing to remove from G
- Remove <ø,ø,ø,ø> from S
- Add <F,B,M,M> to S

G = {<?,?,?,?>}
S = {<F,B,M,M>}

D2 = <F, N, M, Heigh> (+)

- Nothing to remove from G
- Compare s = <F,B,M,M> and d = <F,N,M,H> → S does not satisfy the new positive instance d.
    - Remove s = <F,B,M,M> from S
- Add most specific generalization of s
    - <F,?,M,?>

G = {<?,?,?,?>}
S = {<F,?,M,?>}

D3 = <F,S,L,L> (-)

- S will classify d3 ass negative (correct classify), thus, no need to remove from S.
- G is to generall and will classify d3 as positive, thus we need to make it more specific.
    - Remove from G, g = {<?,?,?,?>}
- Find more specific g's
    - To make the G more specific, we take all possible combinations of values not in d3(example) and still accepted by S.
      All values are <(F,M), (B,N,S), (H,M,L) (H,M,L)>
        - This gives that we can not use
            - F, since it s in d3
            - S (Prob. Area), since it is in d3

- L (Act.lvl), since its in d3
- L (sleep.quality), since its in d3
- H (act.lvl), since not allowed by S
-

- g = <?,B,?,? >
- g = <?,N,?,?>
- g = <?,?,M,?>
- g = <?,?,?,M>
- g= <?,?,?,H>
- G = {<?,B,?,? >, <?,N,?,?>, <?,?,M,?>, <?,?,?,M>, <?,?,?,H>}

G = {<?,B,?,? >, <?,N,?,?>, <?,?,M,?>, <?,?,?,M>, <?,?,?,H>}
S = {<F,?,M,?>}

D4 = <M,N,H,M> (+)

- Remove from G, any that does not satisfy d4
    - g = <?,B,?,?> does not match,
    - g = <?,?,M,?> does not match,
    - g = <?,?,?,H>, does not match,
    - G = {<?,N,?,?>, <?,?,?,M>}
- Remove from S all hypothesis that does not accept d4
    - s = <F,?,M,?> does not match
    - S = {}
- Add to S all minimum generalisations of s
    - <?,?,?,?>
    - S = {<?,?,?,?>}
- Remove from S any hyp that is more general than another
    - Nothing to remove

NOTE: Now we see that S is more general than G, this is not allowed!

G = {<?,B,?,? >, <?,N,?,?>, <?,?,M,?>, <?,?,?,M>}
S = {<?,?,?,?>}

D5 = <M,B,M,L> (+)

- Remove from G any hyp not consistent with d5
    - <?,N,?,?> not consistent
    - <?,?,?,M> not consistent
    - G = {<?,B,?,? >, <?,?,M,?>}
- For each hyp s in S not consistent with d5, remove s. But S is as general as can be. Thus nothing to remove.
- Add all min. Gen of s, But s cannot be made more general.

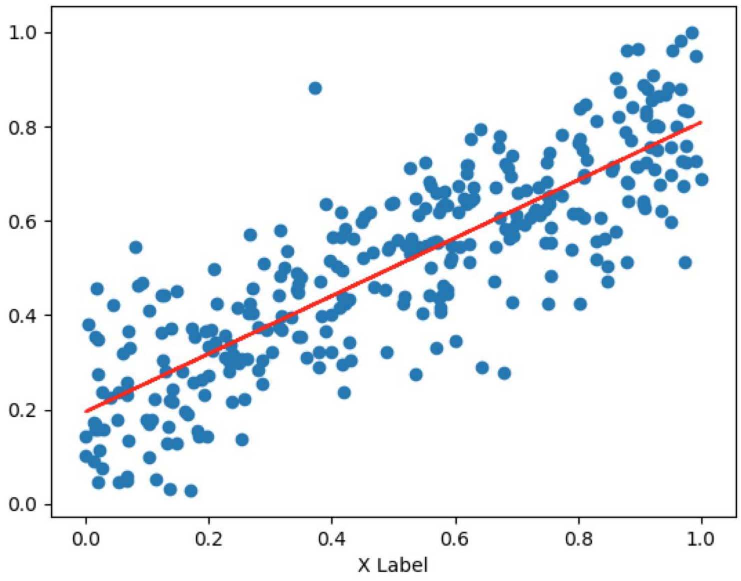
- Remove from S any hypt that is more general than another hyp in s: nothing to remove.

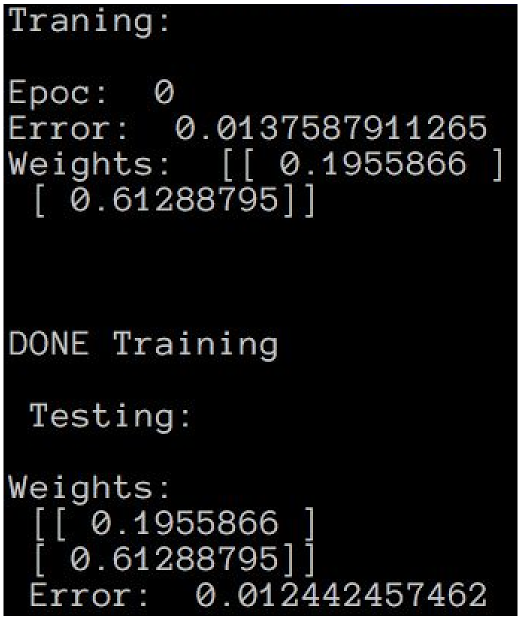
G = {<?,B,?,? >, <?,?,M,?>}
S = {<?,?,?,?>}

S becomes more general than G in iteration 4 and G becomes and empty set in iteration 5, which points to error in dataset, which is one of the “criterias”: CA requires noise-free data.

Linear regression:

1. Se code “Linear_regression.py
2.

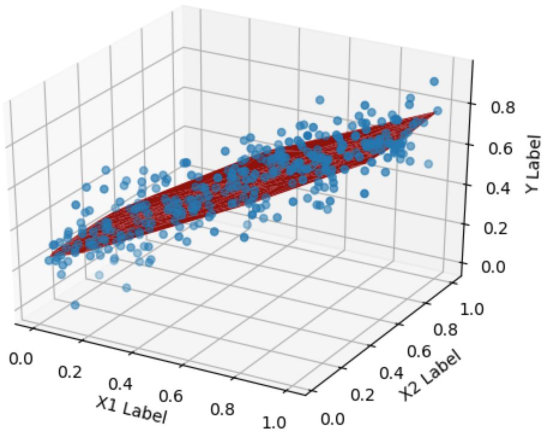


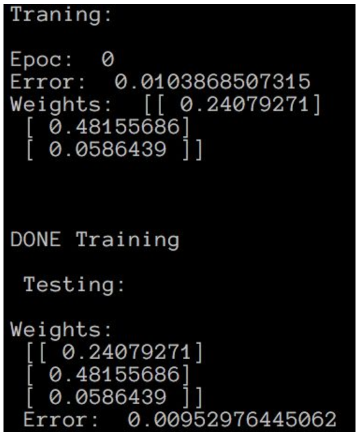


```
Traning:

Epoc:  0
Error:  0.0137587911265
Weights:  [[ 0.1955866 ]
 [ 0.61288795]]


DONE Training

 Testing:

Weights:
 [[ 0.1955866 ]
 [ 0.61288795]]
 Error:  0.012442457462
```

The model is generalizing well, based on the error.

3.





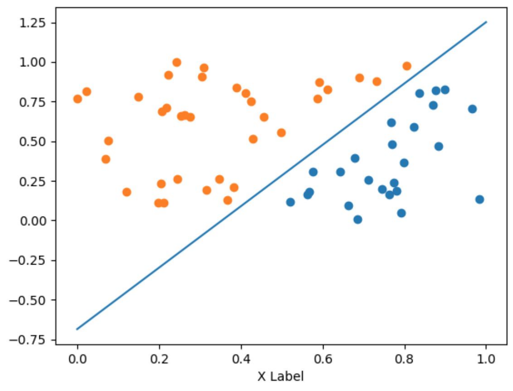
```
Traning:

Epoc:  0
Error:  0.0103868507315
Weights:  [[ 0.24079271]
 [ 0.48155686]
 [ 0.0586439 ]]



DONE Training

 Testing:

Weights:
 [[ 0.24079271]
 [ 0.48155686]
 [ 0.0586439 ]]
 Error:  0.00952976445062
```

The model is generalizing well, based on the error.

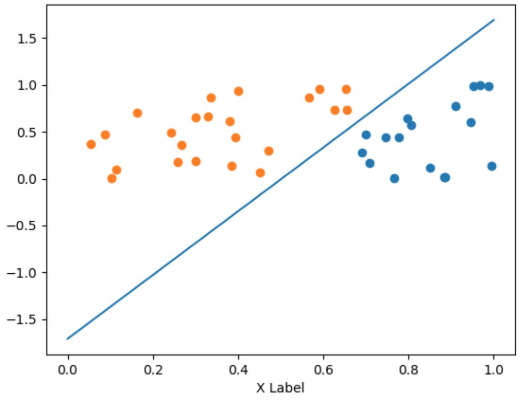
Logistic regression:

1.

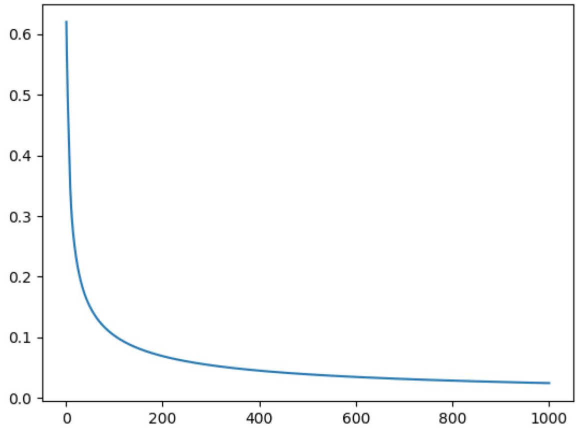
When plotting the data, one can easily see that the data is linearly separable.

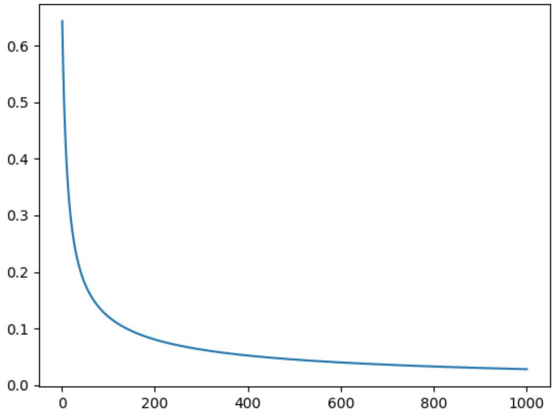


Training



Testing

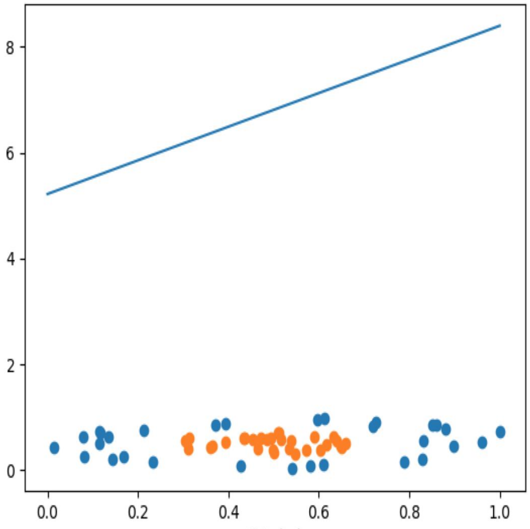


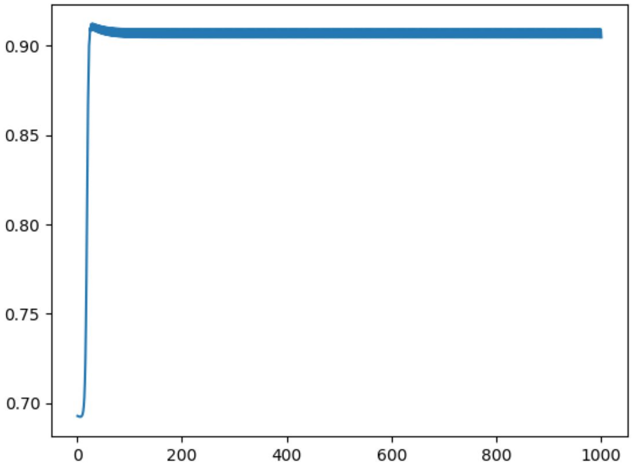


Cross entropy: Training: Testing

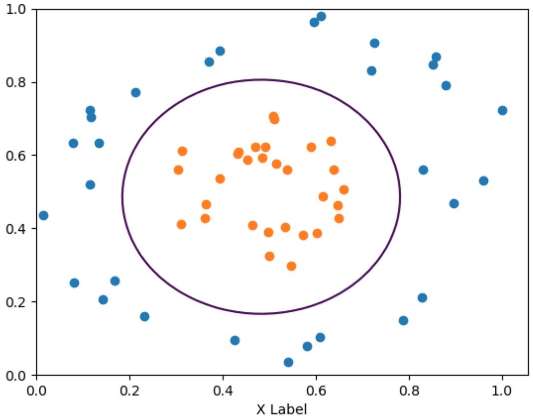
Initial weights : 0, 0, 0
Learning rate: 0.1

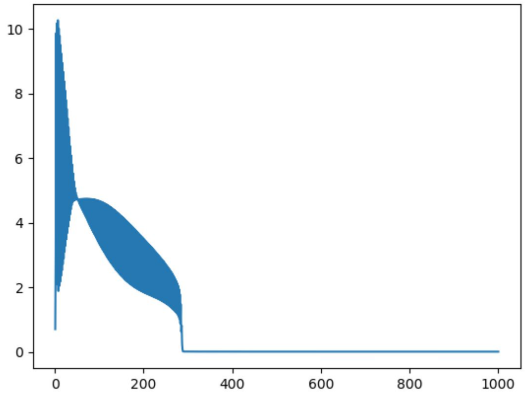
2.
The data is not linearly separable, because there is no clear “line” that can be drawn to differentiate the points based on the two points. Thus, to classify this one can use a circle to draw around / classify the points.

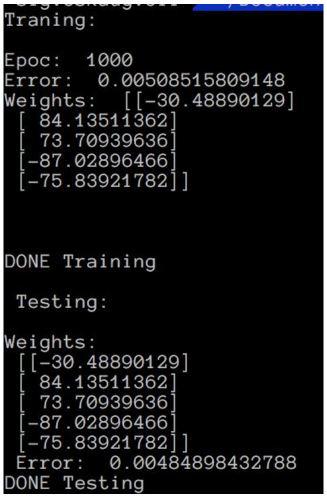




Training set

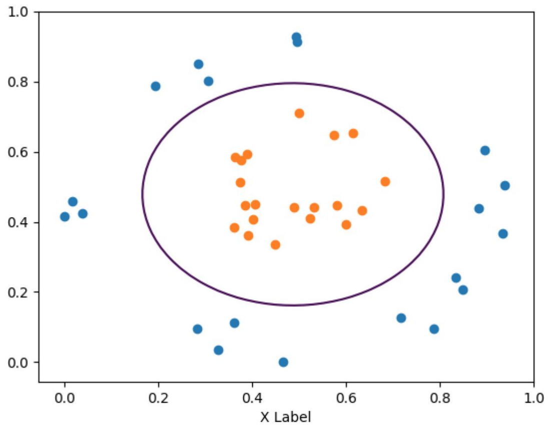


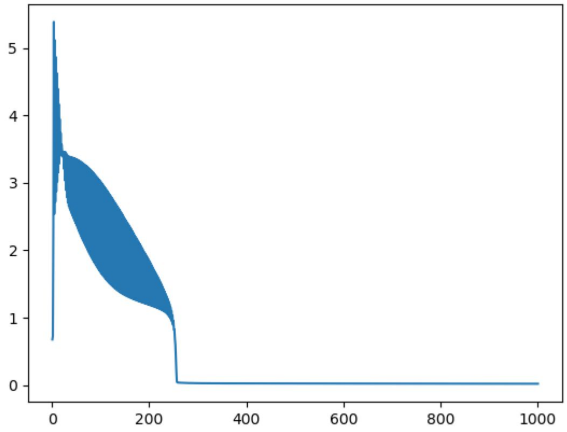




```
Traning:

Epoc:  1000
Error:  0.00508515809148
Weights:  [[-30.48890129]
 [ 84.13511362]
 [ 73.70939636]
 [-87.02896466]
 [-75.83921782]]



DONE Training

 Testing:

Weights:
 [[-30.48890129]
 [ 84.13511362]
 [ 73.70939636]
 [-87.02896466]
 [-75.83921782]]
 Error:  0.00484898432788
DONE Testing
```

Testing set:


X Label


5
4
3
2
1
0
0
200
400
600
800
1000