ANLY601 Assignment1

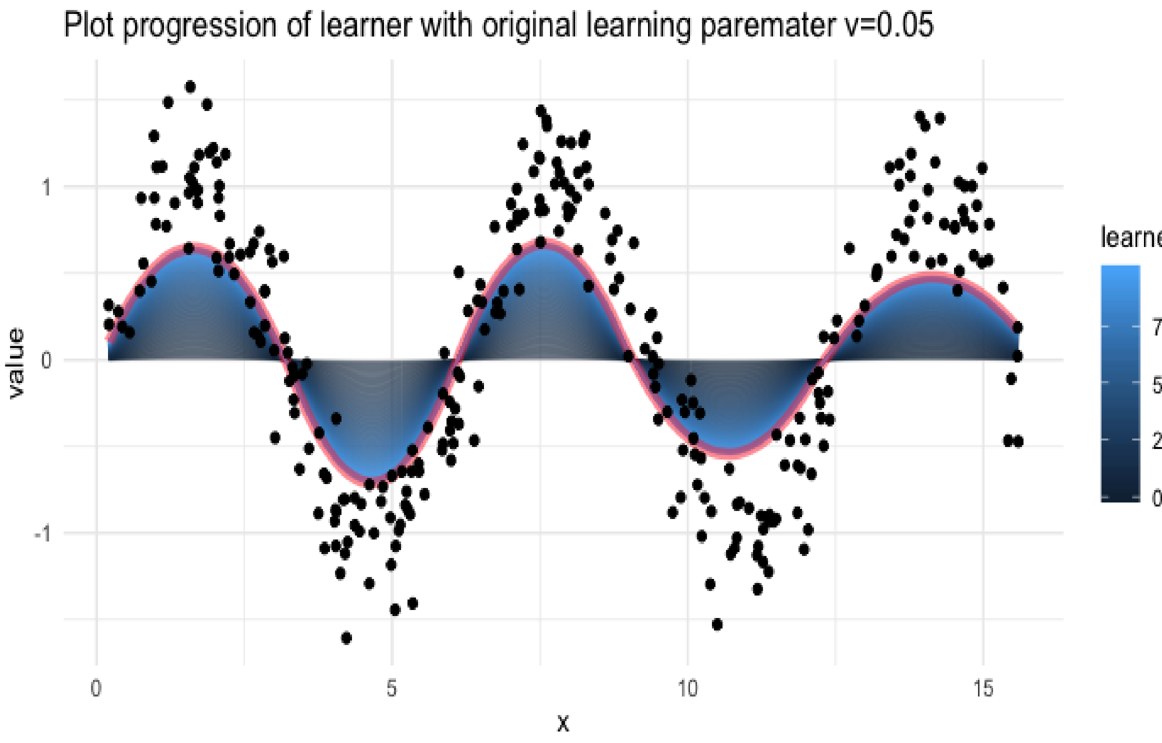
Shaoyu Feng (sf865)
Collobrator: Mengtong Zhang (mz500)
Collobrator: Yunjia Zeng (yz682)

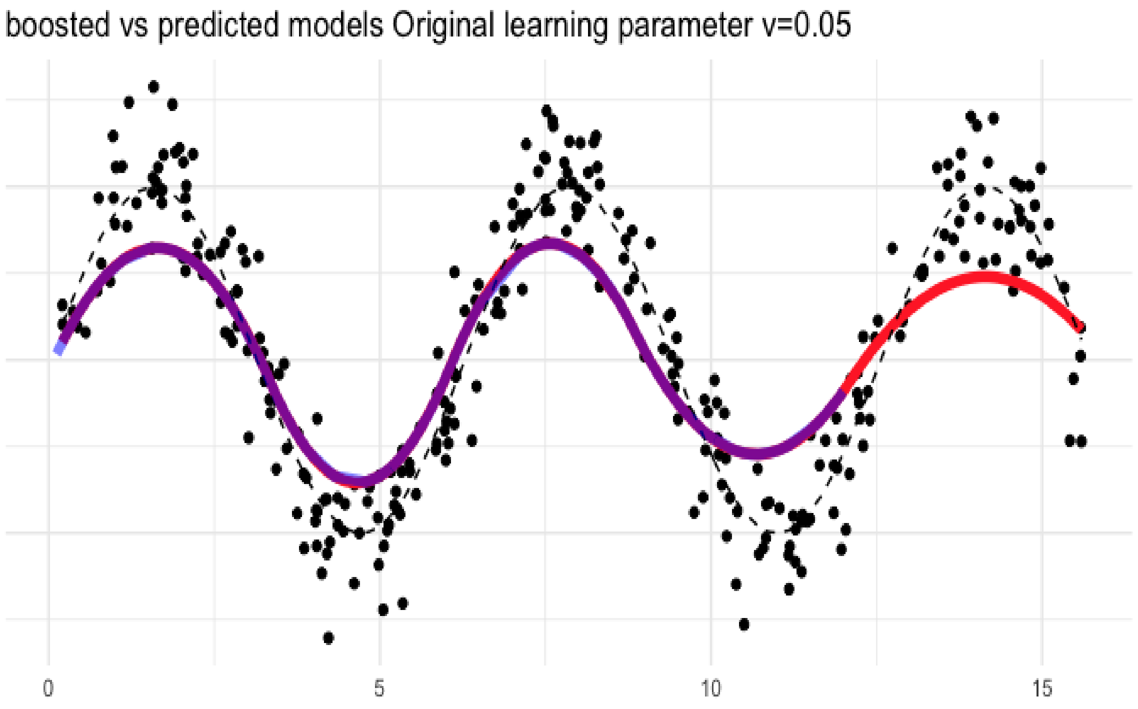
April 2020

1 Question1 Boosting

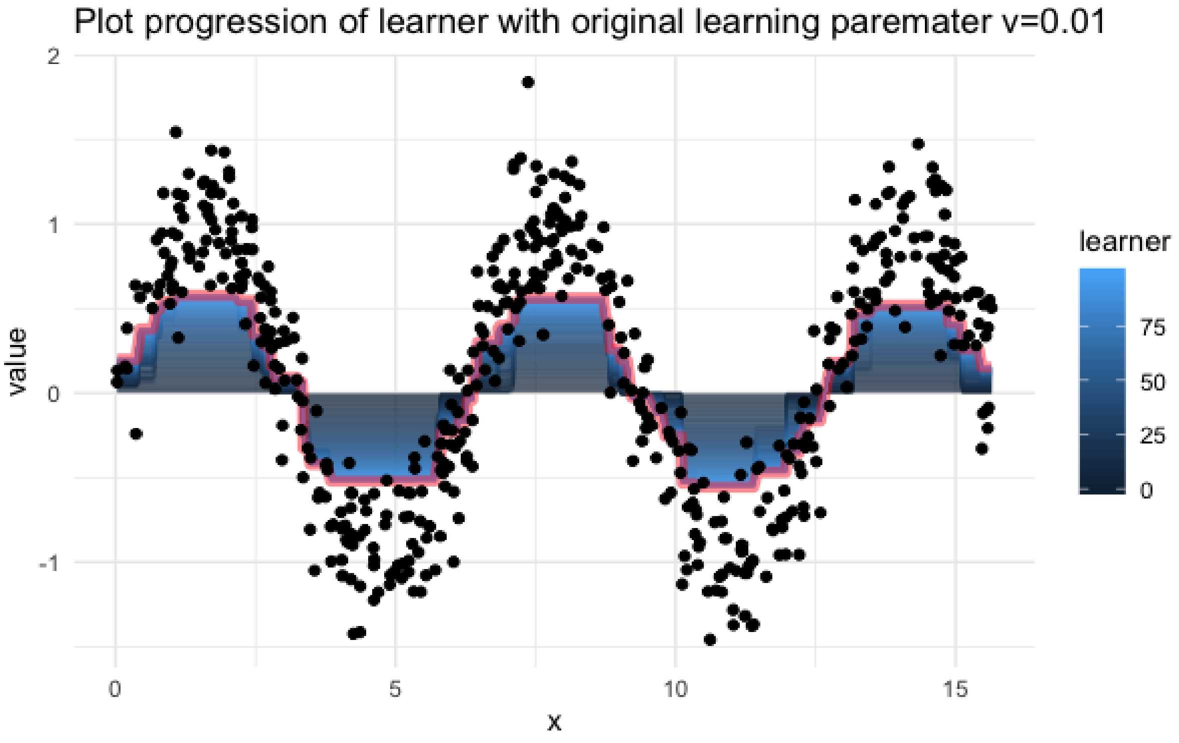
1. **Part1**

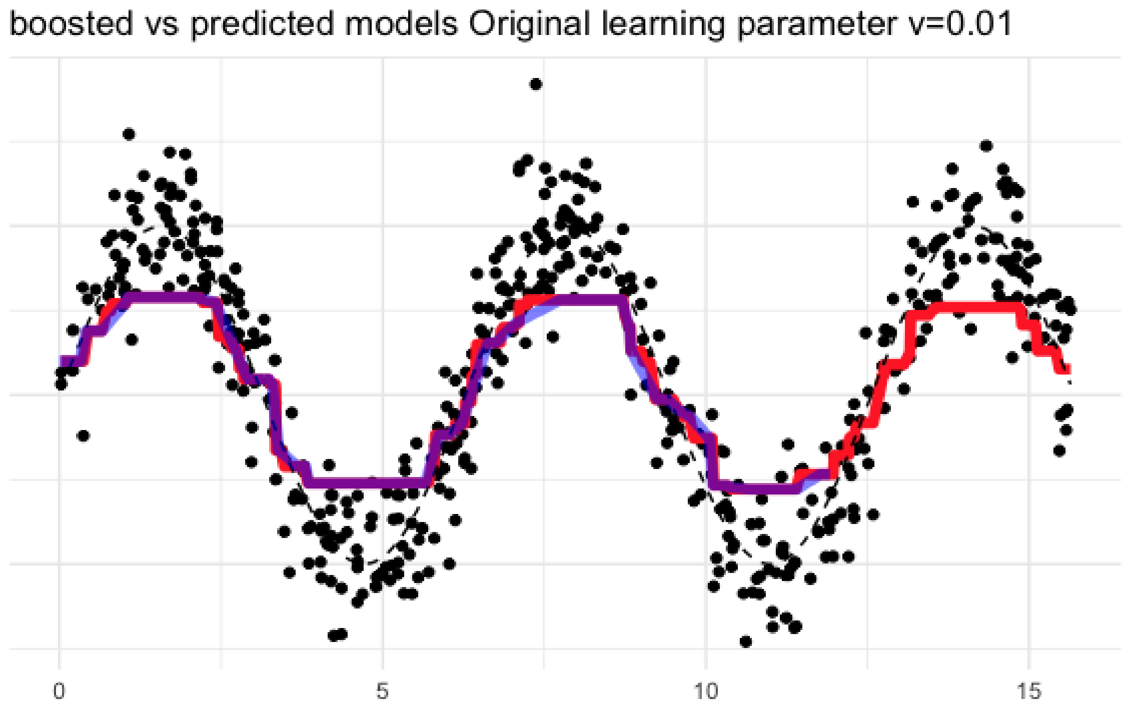
   (a) Q0



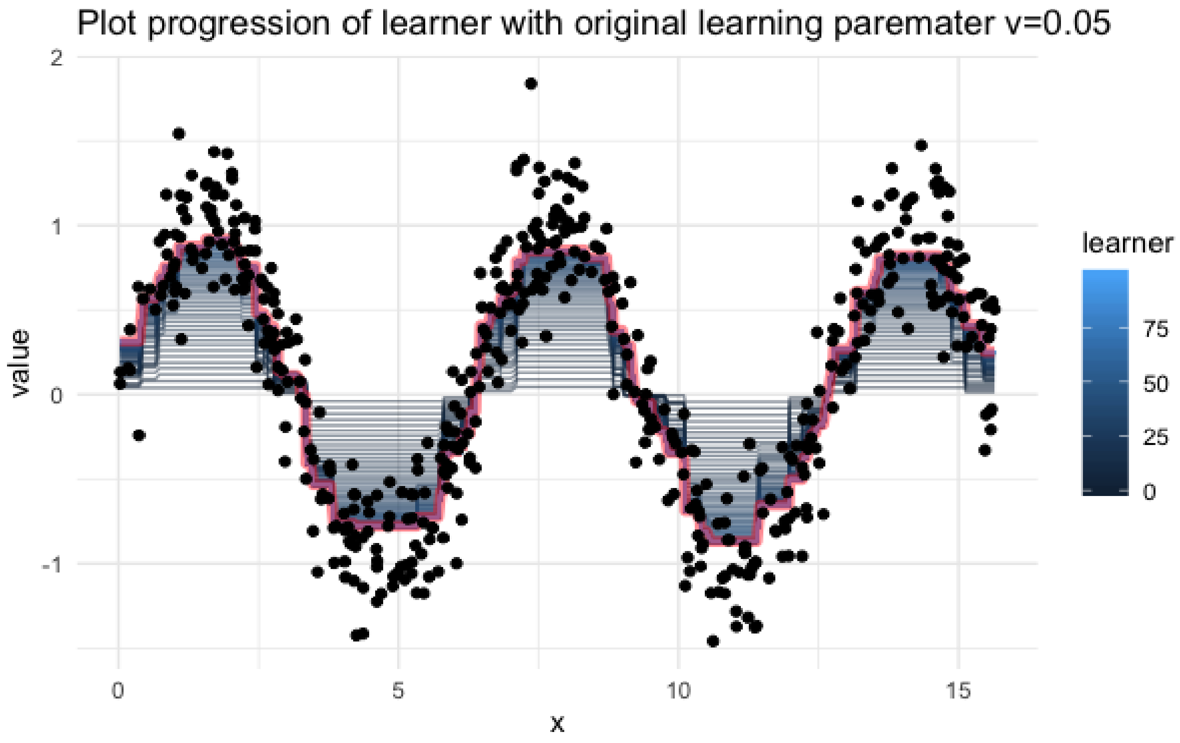

boosted vs predicted models Original learning parameter v=0.05
1
0
-1
0
5
10
15

(b) Q1

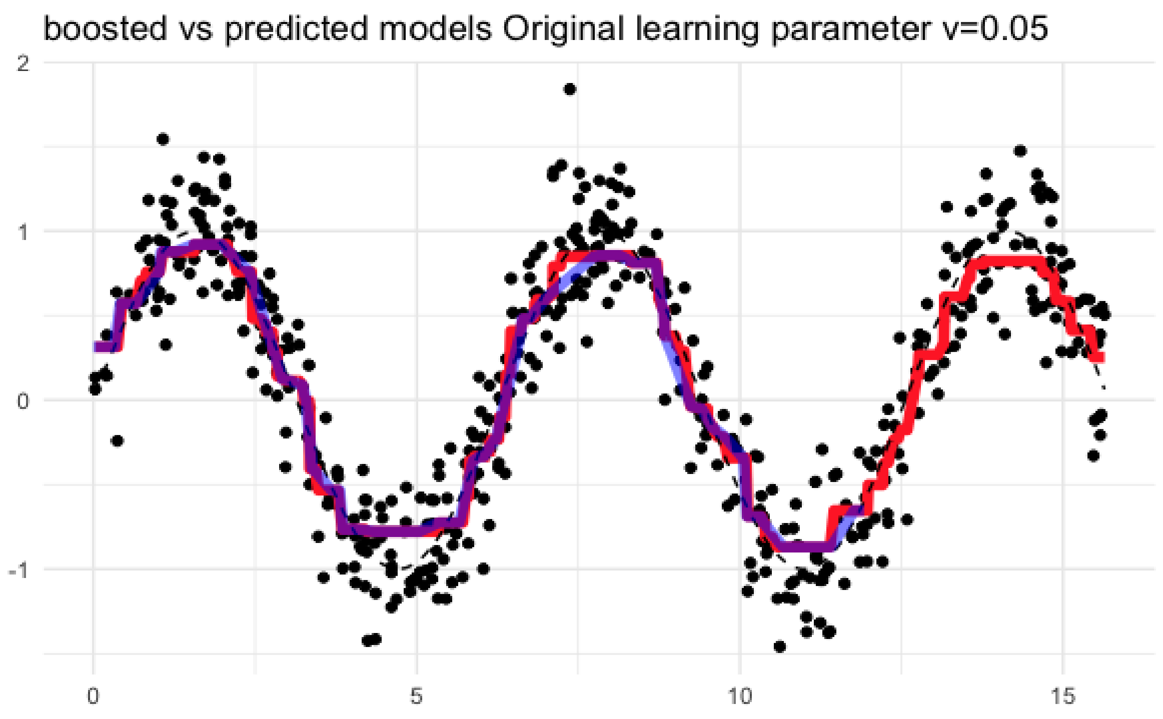

Plot progression of learner with original learning paremater v=0.01
2
1
0
-1
value
0
5
10
15
x
learner
75
50
25
0

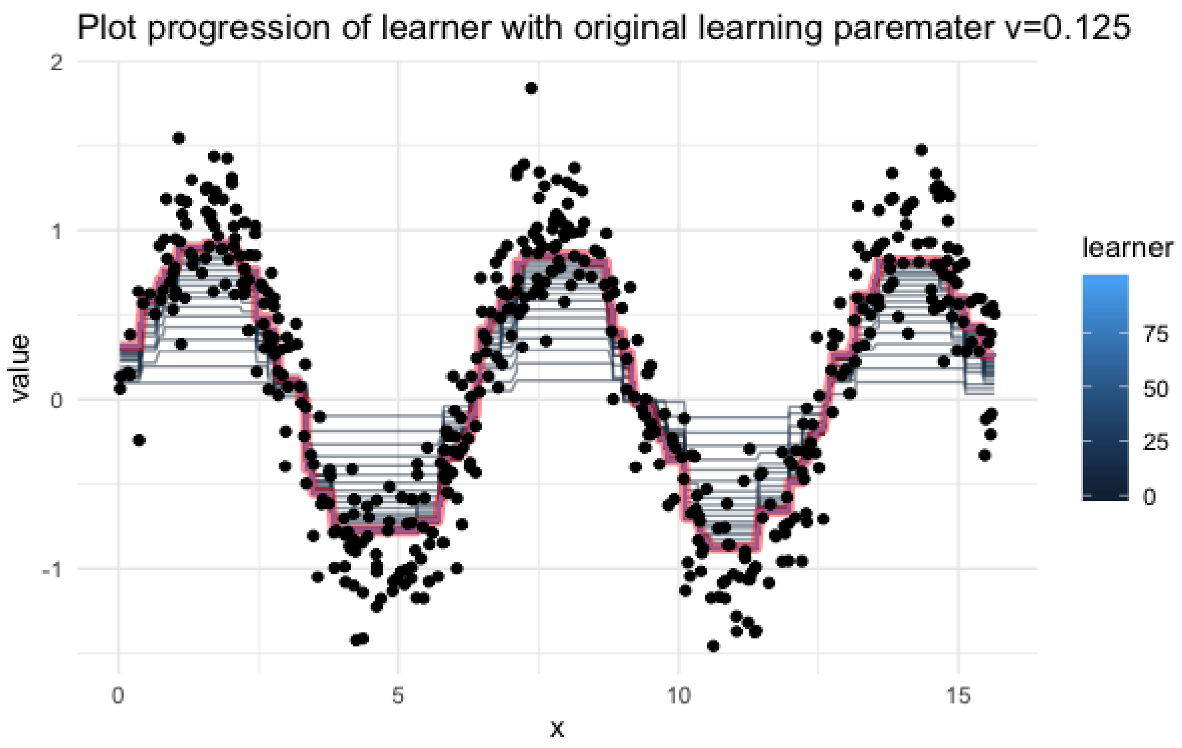


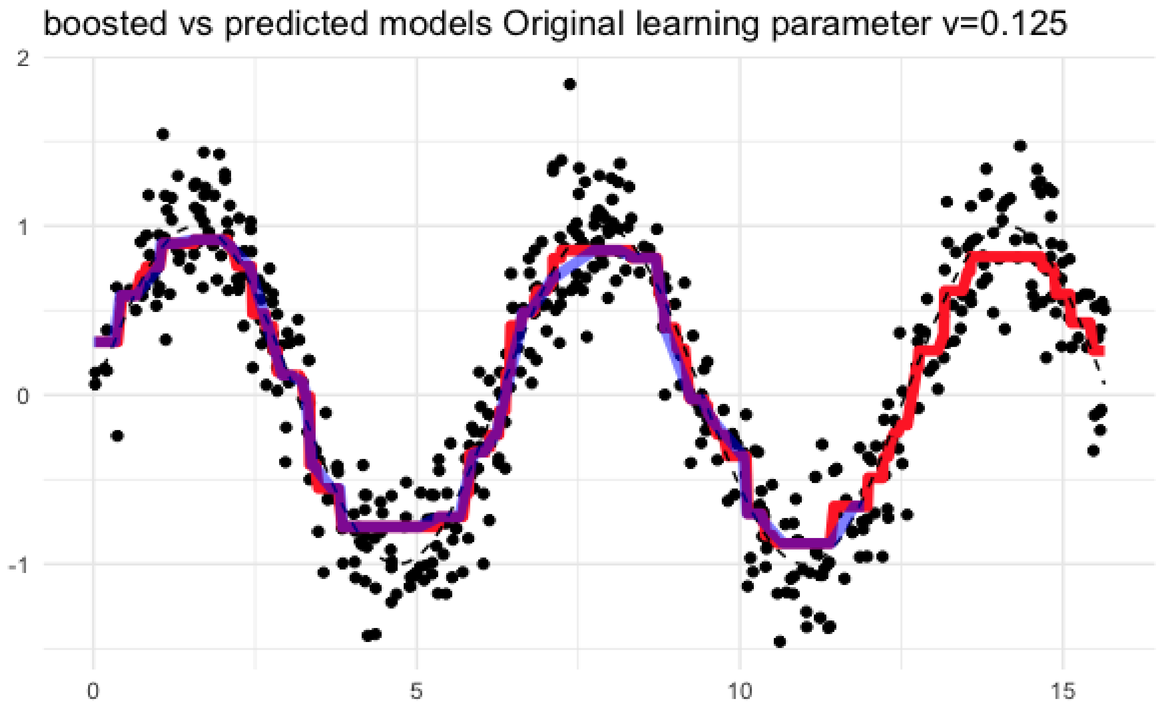
boosted vs predicted models Original learning parameter v=0.01
2
1
0
-1
0
5
10
15



Plot progression of learner with original learning paremater v=0.05
value
x
2
1
0
-1
0
5
10
15
learner
75
50
25
0

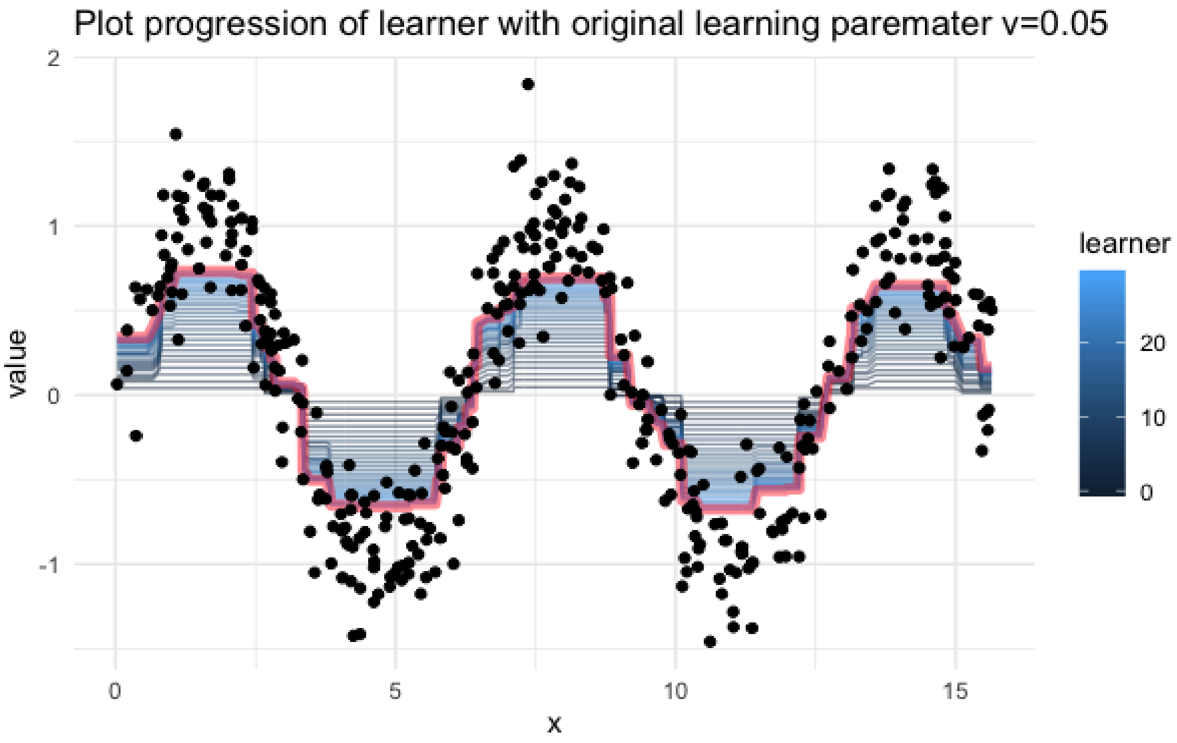

boosted vs predicted models Original learning parameter v=0.05
2
1
0
-1
0
5
10
15


Plot progression of learner with original learning paremater v=0.125
value
x
learner
75
50
25
0
2
1
0
-1
0
5
10
15



As we observed, when the learning parameter is small, the overall model seems to be under-fitted, and the model progress slower by each learner than a relatively large learning parameter(v=0.125)

(c) Q2
Part A
The idea is to to check validation data RMSE during training stage, if we see the change in validation RMSE is small(the threshold is adjustable), we can early stop the training process.
Please refer to **tree-boosting-with-early-stop.R** script for the implementation.
Part B
If we set the validation RMSE change to be 0.005, the training stops with 30 trees in my case.



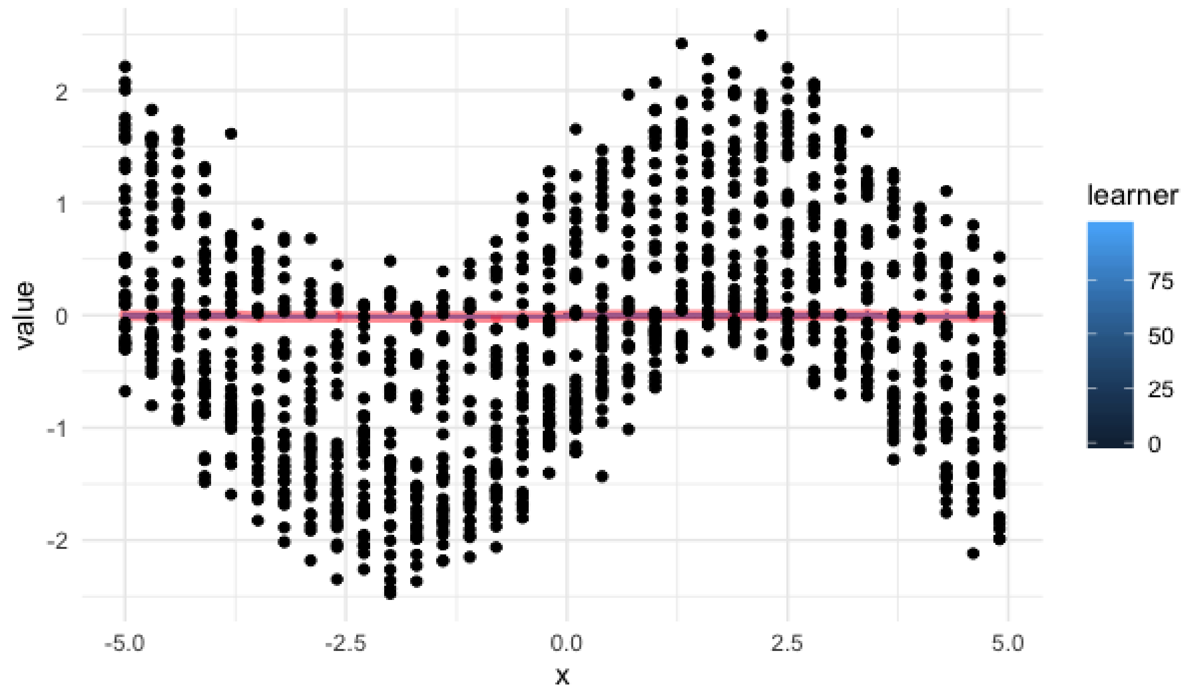
Part C
Test Data RMSE is .3549077 in my training.

(d) Q3
Please refer to **tree-boosting-with-grid-search.R** for implementation. The best set of parameter is when minsplit=5, maxdepth=5 and cp=0.05, in such case, RMSE is 0.312280434490507

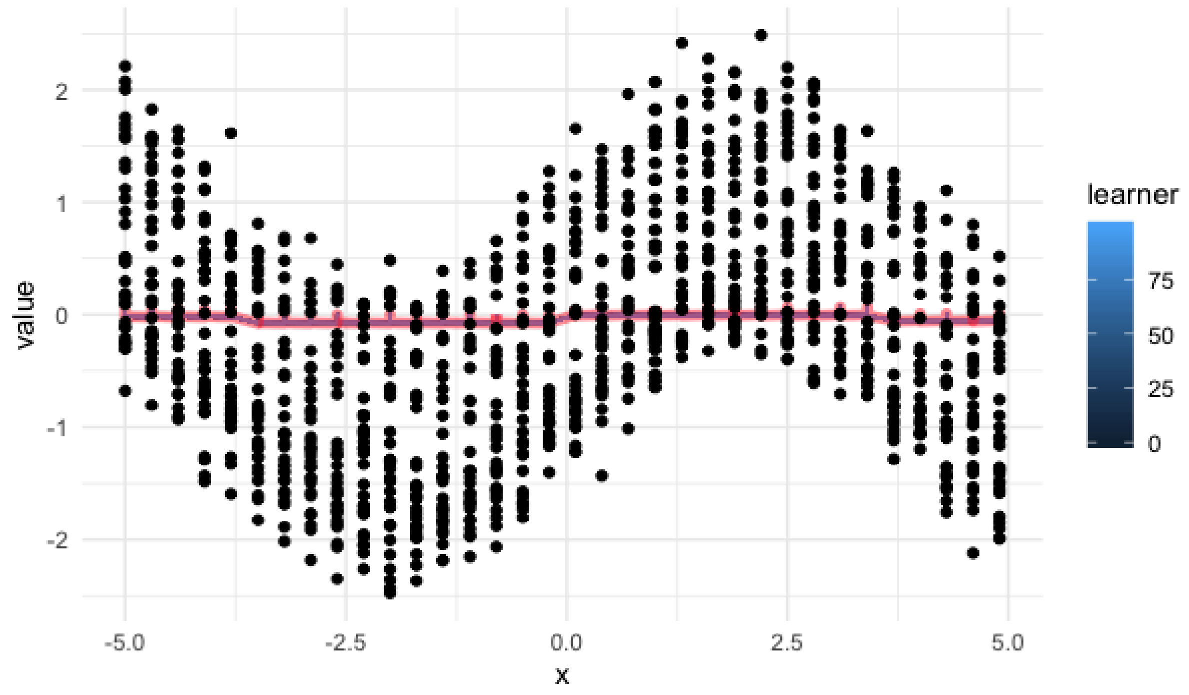
2. **Part2: Multiple-Dimension Case**

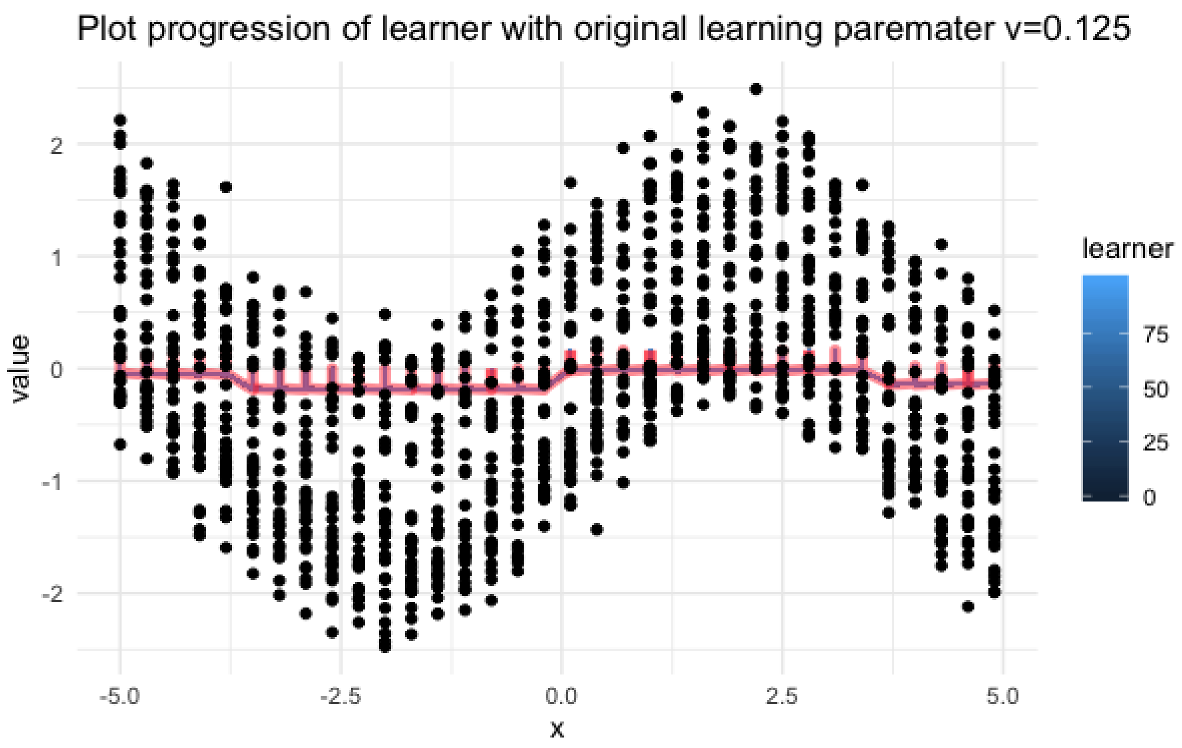
(a) Q1

Plot progression of learner with original learning paremater v=0.01



Plot progression of learner with original learning paremater v=0.05



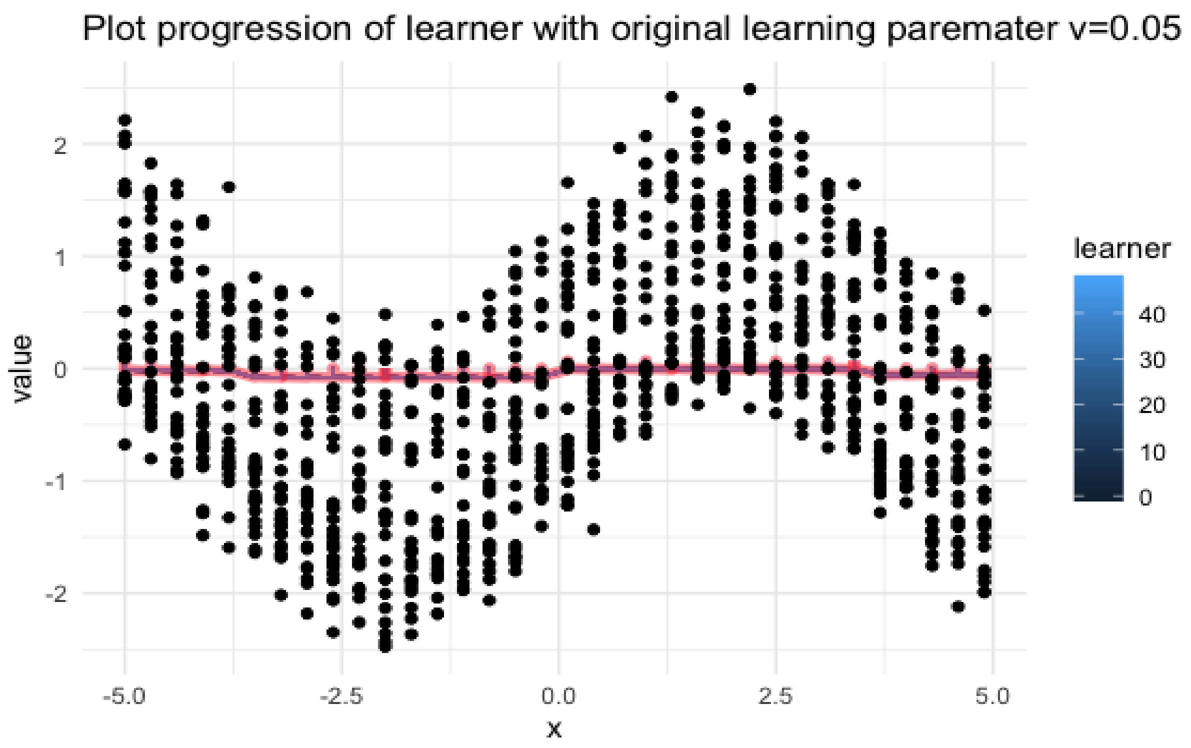


As we observed, the model seems to be under-fitted with current model.

(b) Q2

Part A

Please refer to **tree-boosting-multiple-dimension-early-stop.R** script for the implementation.



Part B
48 trees are included.

Part C
RMSE for test dataset is 0.09576191.

(c) Q3
Please refer to **tree-boosting-multiple-dimension-grid-search.R** script for the implementation.

Please refer to **tree-boosting-with-grid-search.R** for implementation. The best set of parameter is when minsplit=2, maxdepth=5 and cp=0.01, in such case, RMSE is 0.101362044548129

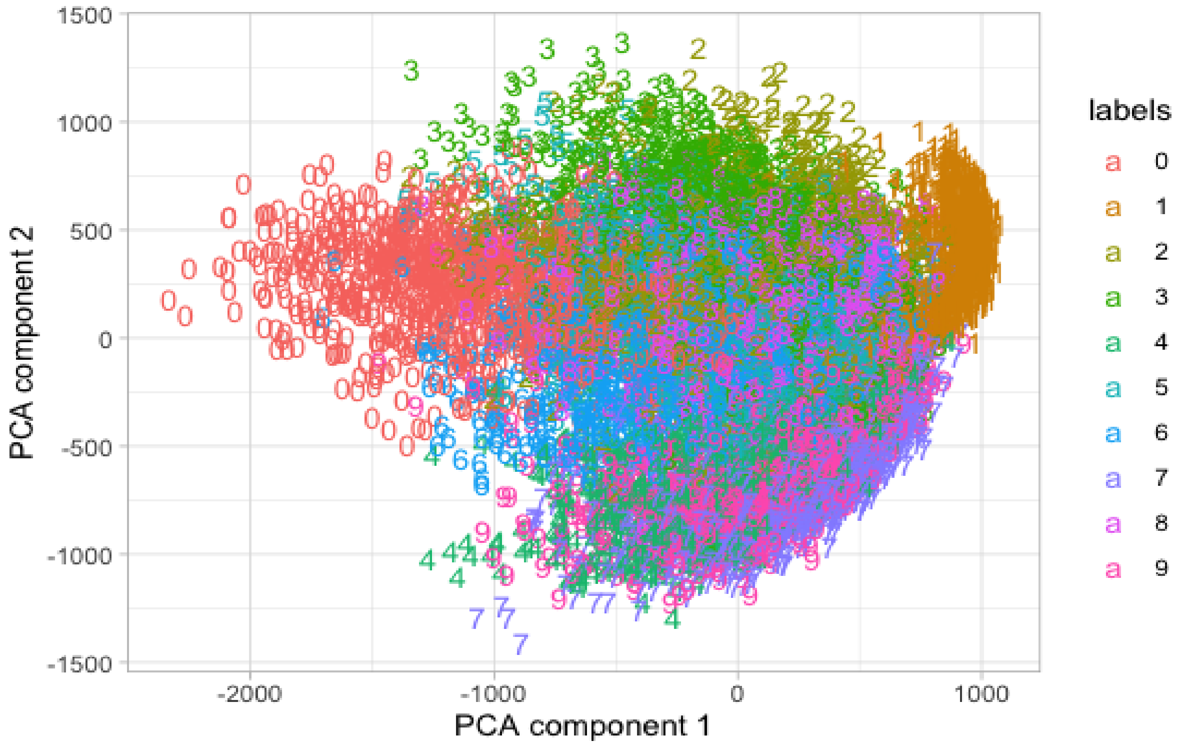
2 Question 2 (TSNE)

Part 1
(a) Yes, it matters. It indicates that the two points are close in their original dimension space.
(b) It is an estimation of the amount of close neighbors each point has and can be user to balance attention between local and global aspects of your data.
(c) If the number of steps small, t-SNE plot may have strange "pinched" shapes because it stops early and ends before convergence state. Optimal number of steps are dependent on data sample you have.
(d) read topological information off t-SNE plot typically requires views at multiple perplexities (multiple plots). One of the simplest topological properties is containment. t-SNE with low perplexity value greatly exaggerates the size of the smaller group of points. Another is trefoil knot, low perplexity values failed to

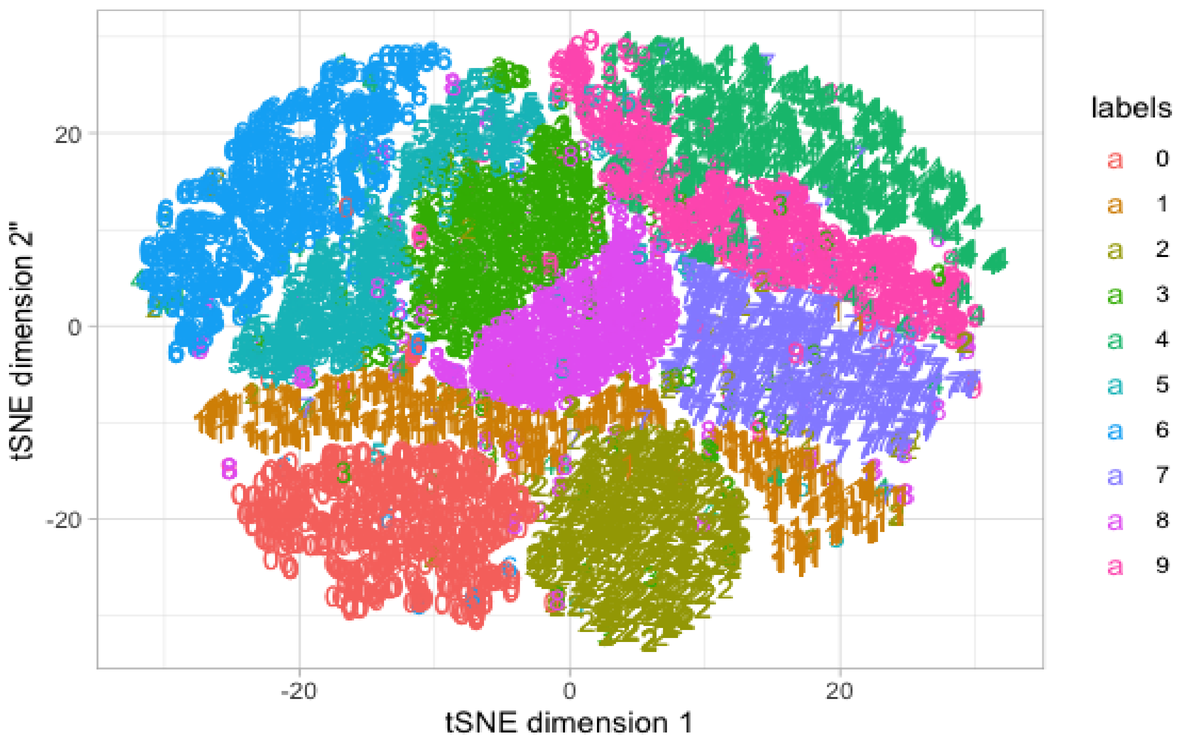
show global connectivity.

Part 2

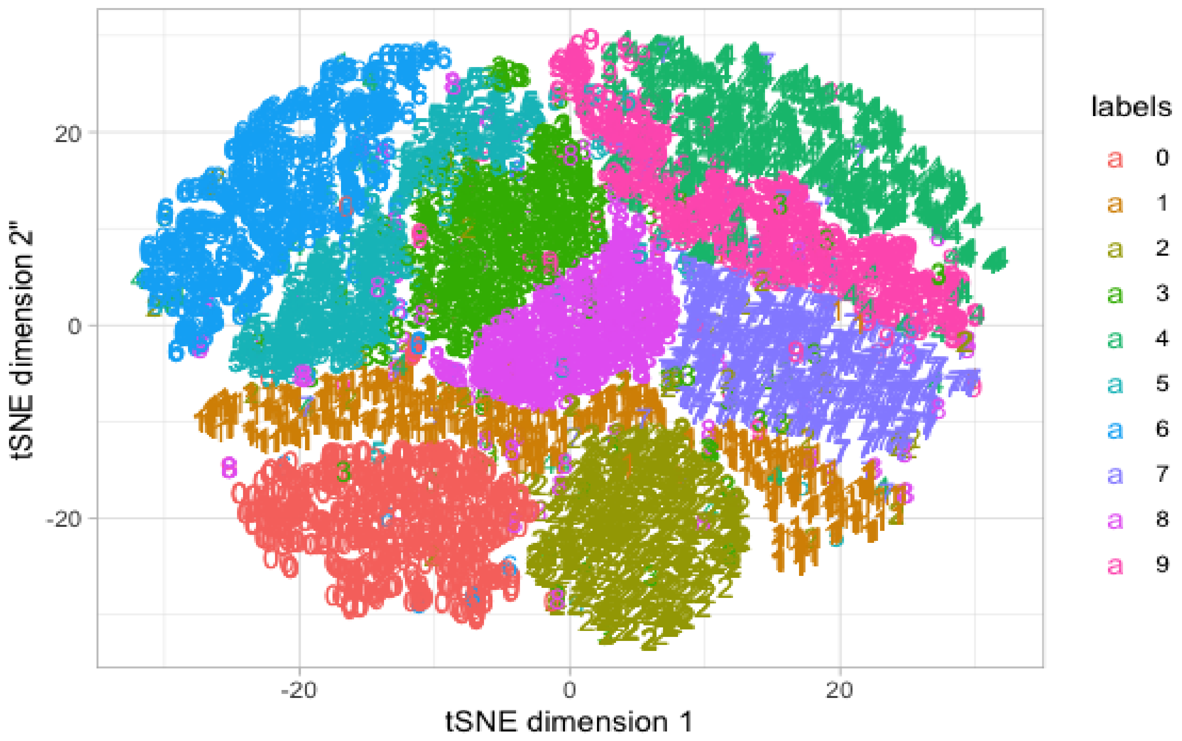
(a)

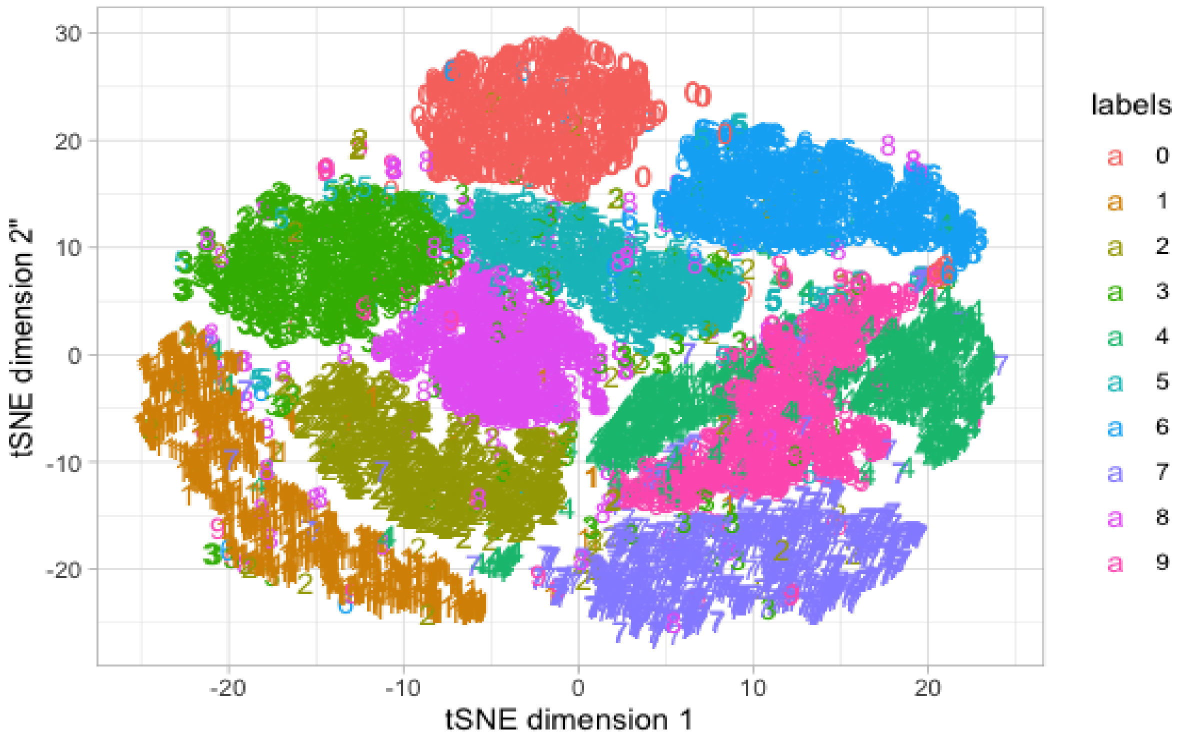


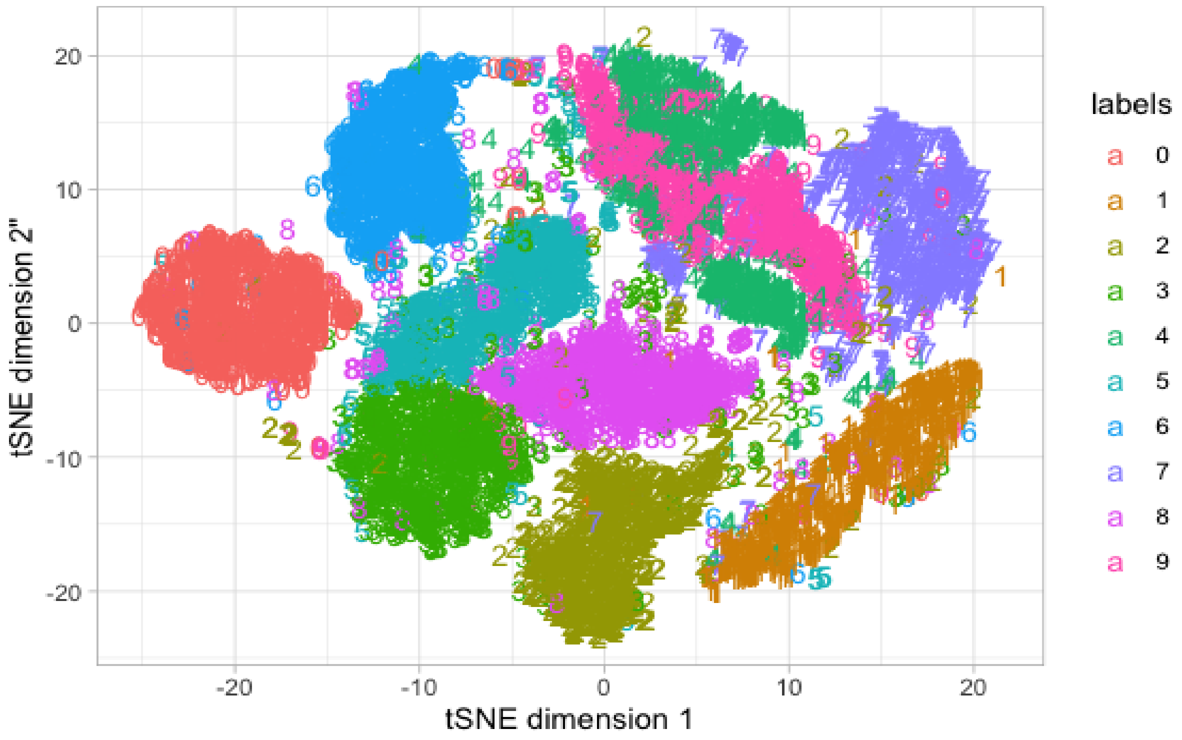
(b)

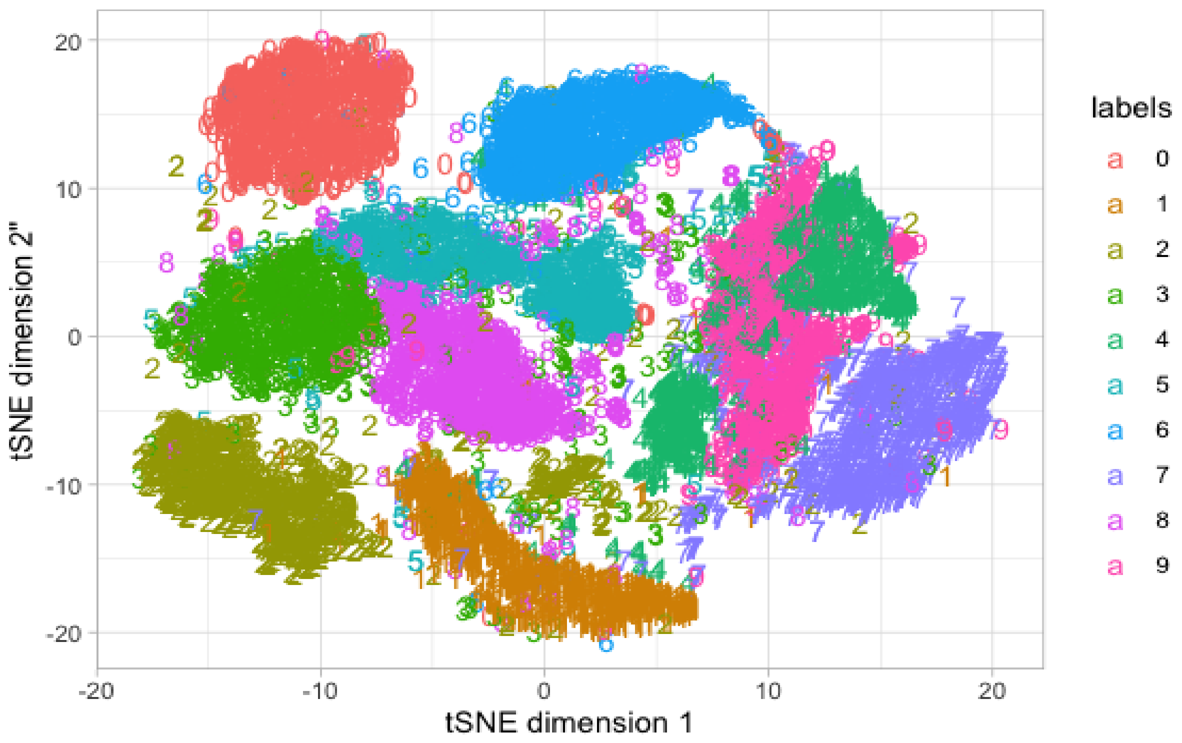


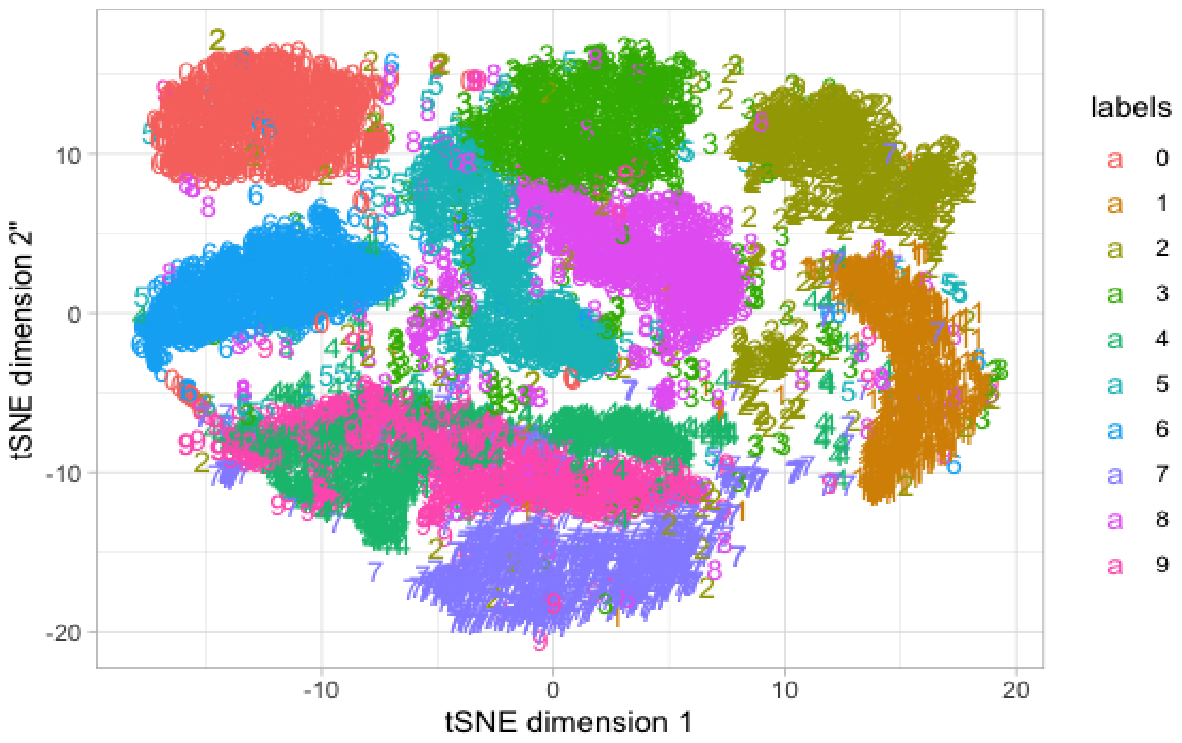
(c)
The following charts are shown when perlexity is 5, 20,60,100,125,160.
As we can see, the larger the perlexity, the more diverse each group is apart from each other.

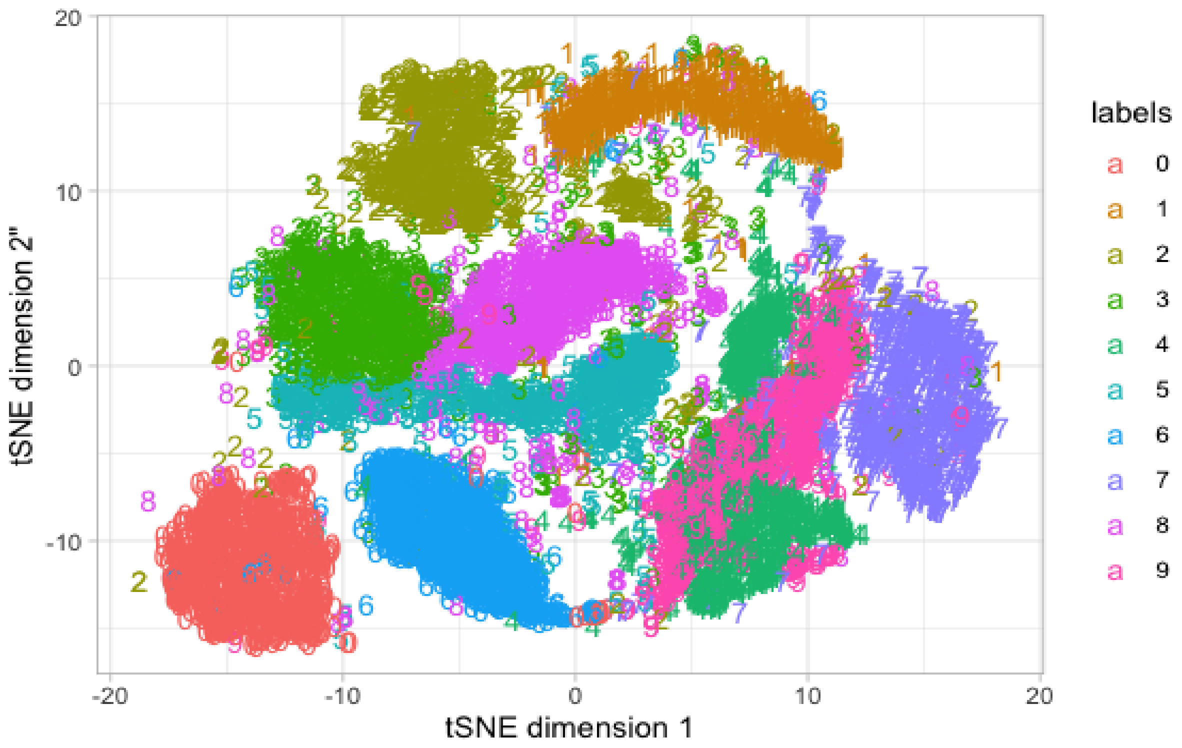



tSNE dimension 2"
tSNE dimension 1
30
20
10
0
-10
-20
-20
-10
0
10
20
labels
a 0
a 1
a 2
a 3
a 4
a 5
a 6
a 7
a 8
a 9

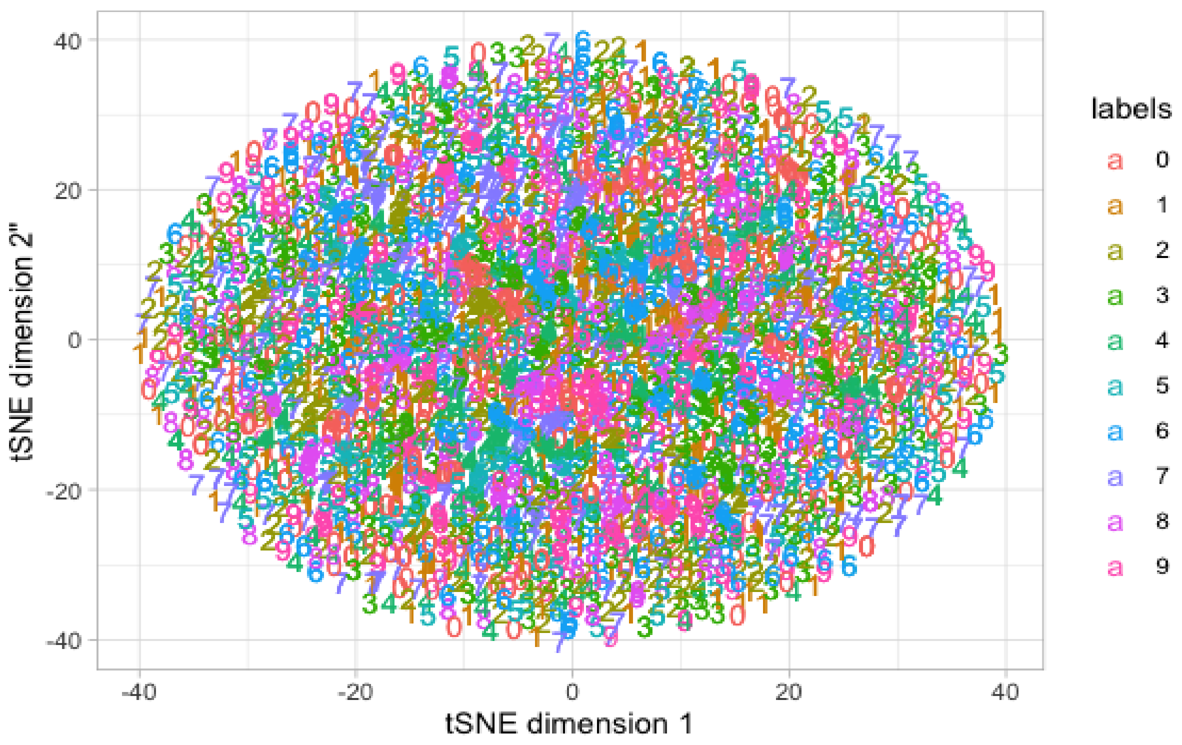

tSNE dimension 2"
tSNE dimension 1
20
10
0
-10
-20
-20
-10
0
10
20
labels
a 0
a 1
a 2
a 3
a 4
a 5
a 6
a 7
a 8
a 9


labels
a 0
a 1
a 2
a 3
a 4
a 5
a 6
a 7
a 8
a 9
tSNE dimension 1
tSNE dimension 2"
20
10
0
-10
-20
-20
-10
0
10
20


labels
a 0
a 1
a 2
a 3
a 4
a 5
a 6
a 7
a 8
a 9
tSNE dimension 1
tSNE dimension 2"
10
0
-10
-20
-10
0
10
20

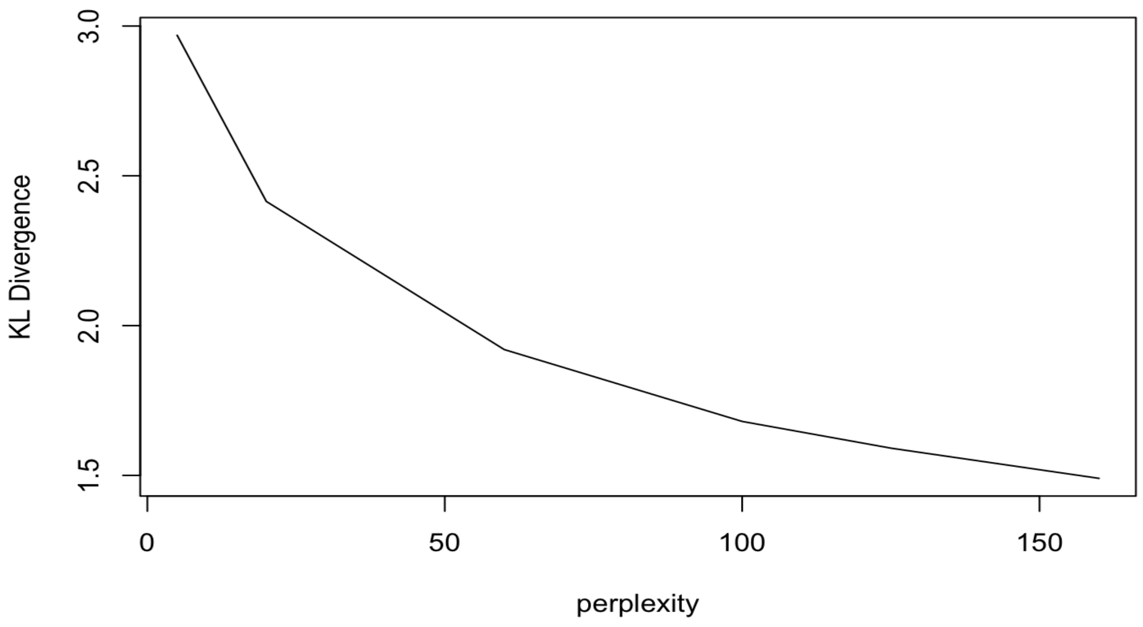


(d)

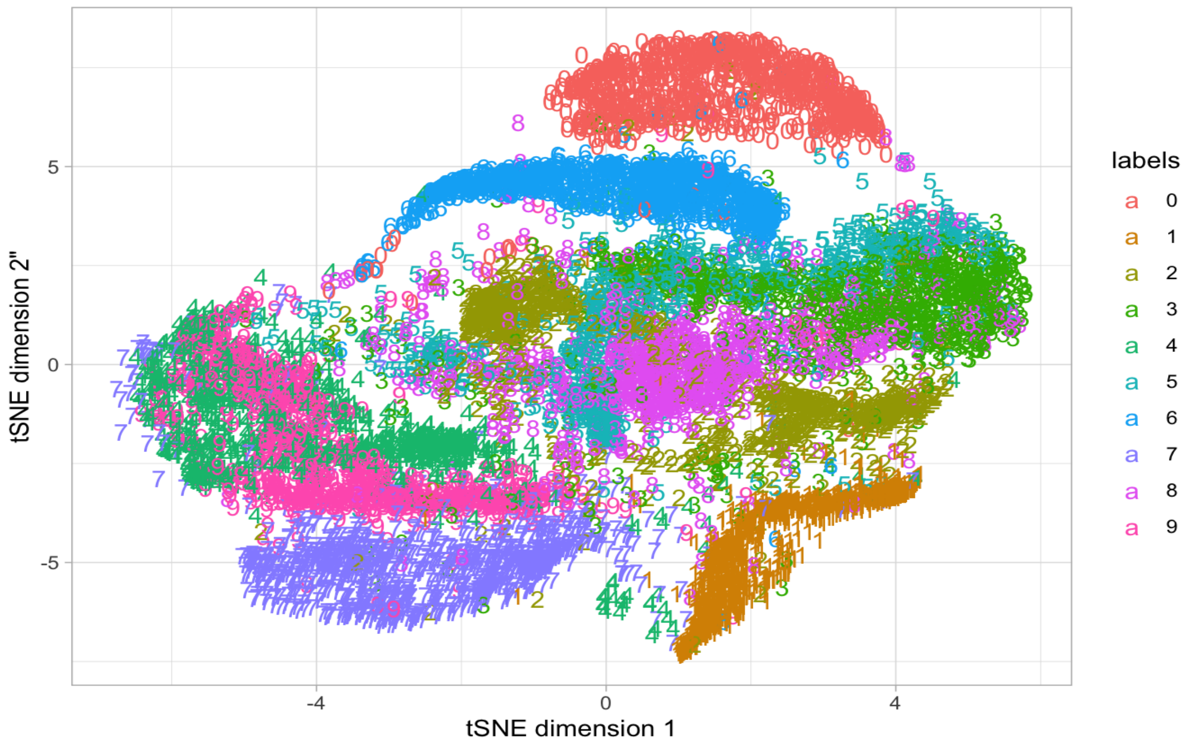


We can see that all the groups are mixed together, this means that we the perlexity is too small for the tSNE to converge. (e)

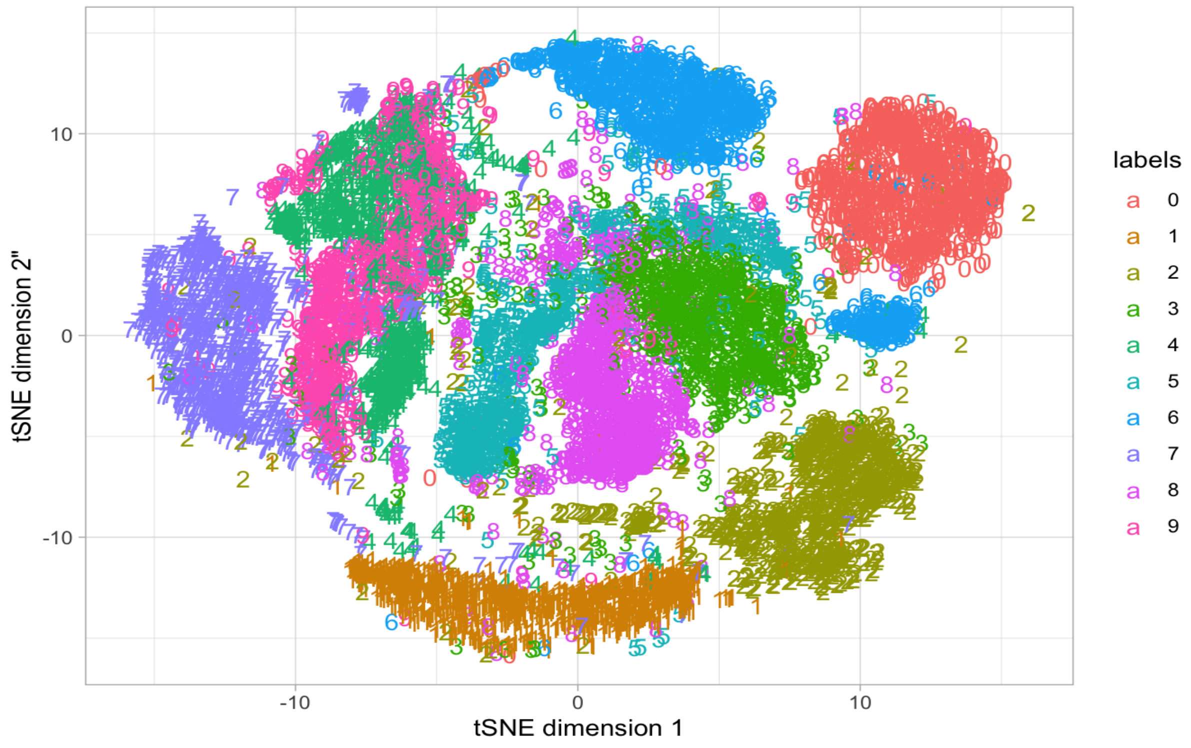
There will be clear decision boundaries between each group, they are totally seperable.
(f)

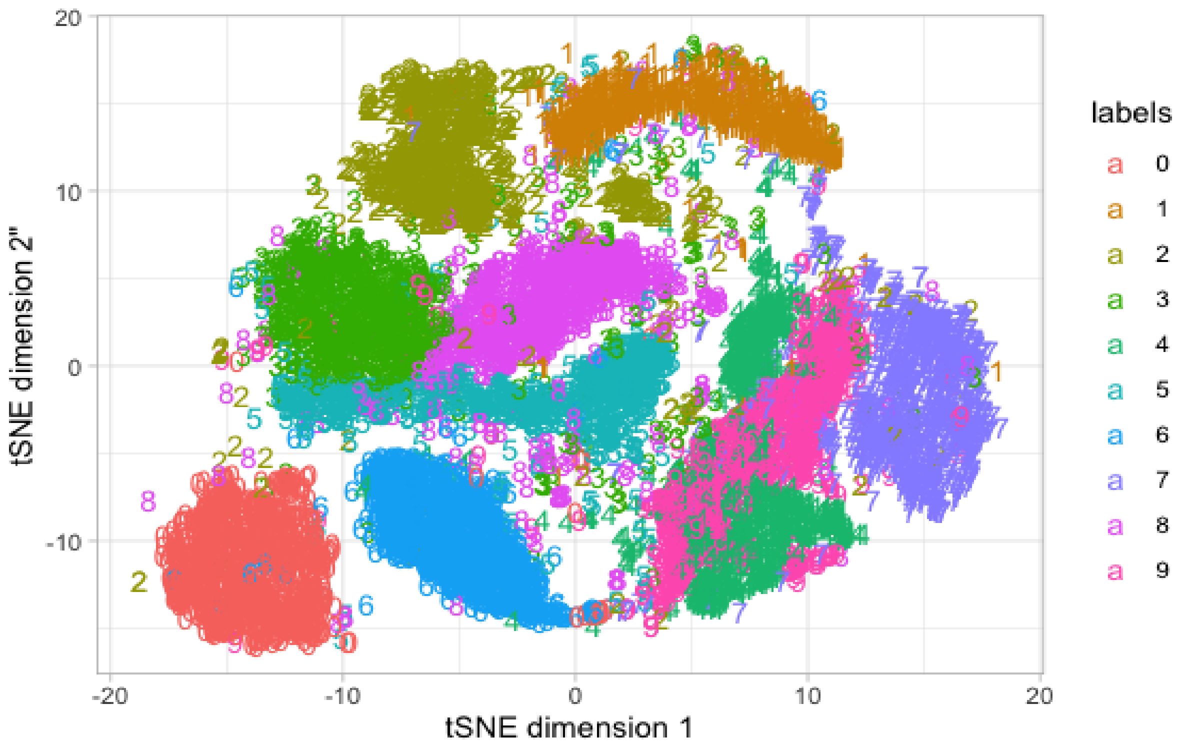


The optimal perplexity is 160.
(g)
LARGER learning rate will lead to more dispersed clustering result. Following are charts when eta=10, 100


tSNE dimension 2"
tSNE dimension 1
5
0
-5
-4
0
4
labels
a 0
a 1
a 2
a 3
a 4
a 5
a 6
a 7
a 8
a 9

and 200

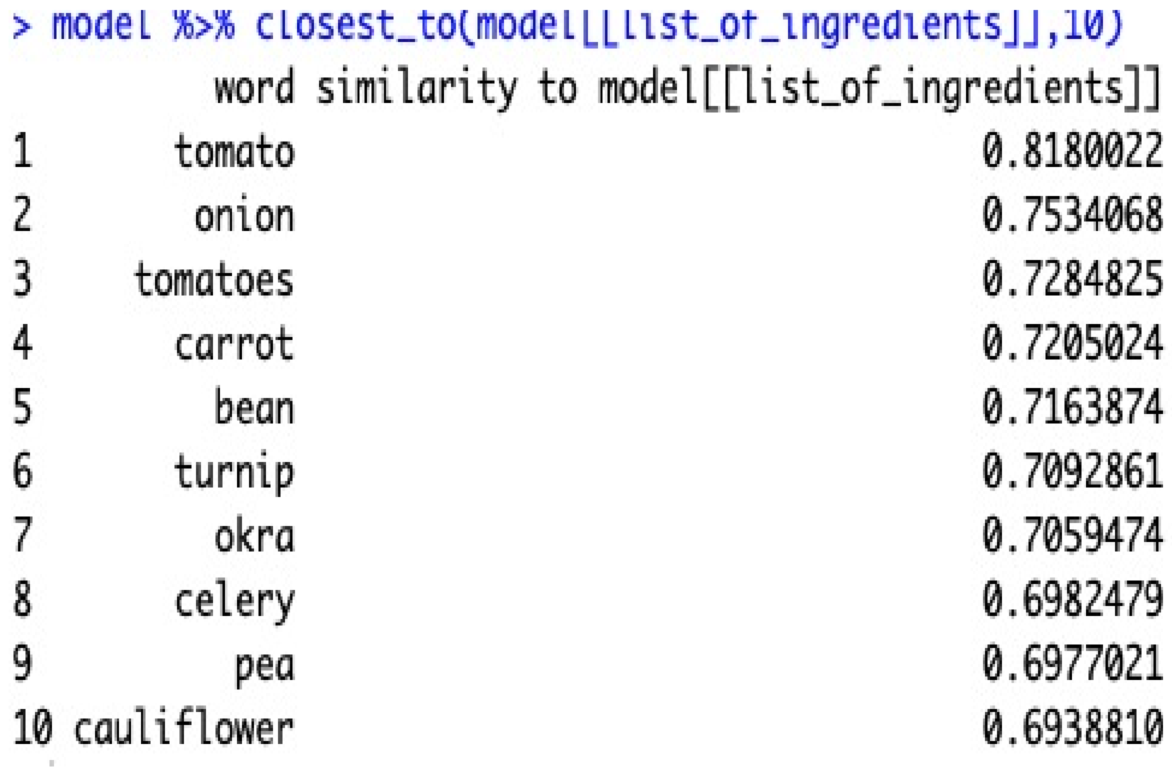

tSNE dimension 2"
tSNE dimension 1
10
0
-10
-10
0
10
labels
a 0
a 1
a 2
a 3
a 4
a 5
a 6
a 7
a 8
a 9



3 Question 3 (Word2Vec) Emeddings

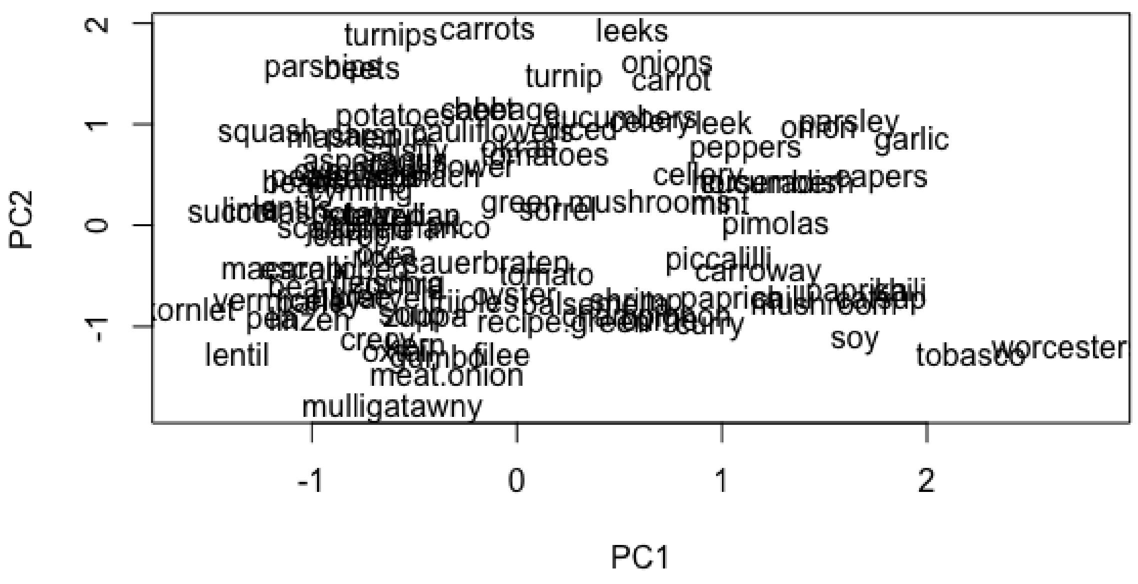
(1)
stop words removal, stemming or lemmatization and text normalization

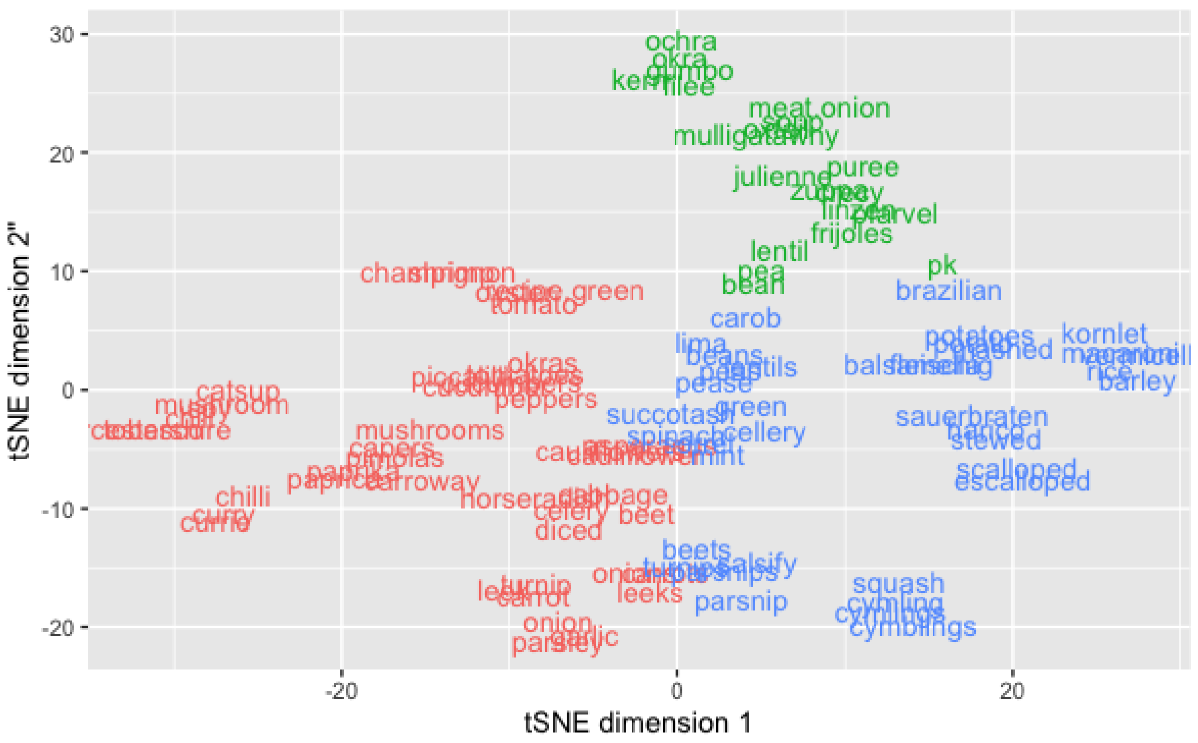
(2)
i chose BEAN, TOMATO and ONION as my ingredients.



```
> model %>% closest_to(model[[list_of_ingredients]],10)
         word similarity to model[[list_of_ingredients]]
1      tomato                                    0.8180022
2       onion                                    0.7534068
3    tomatoes                                    0.7284825
4      carrot                                    0.7205024
5        bean                                    0.7163874
6      turnip                                    0.7092861
7        okra                                    0.7059474
8      celery                                    0.6982479
9         pea                                    0.6977021
10 cauliflower                                   0.6938810
```

tomato, onion, tomatoes, carrot, bean are the top 5 similar ingredient.
(3)
t-SNE to see the relationships between set of words related with the three ingredients above.

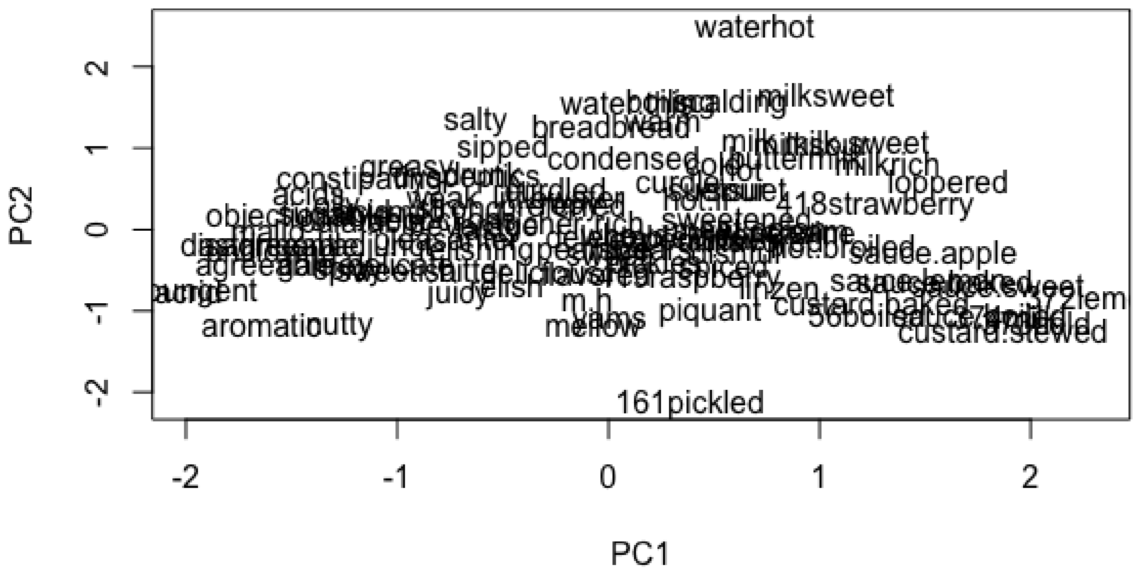


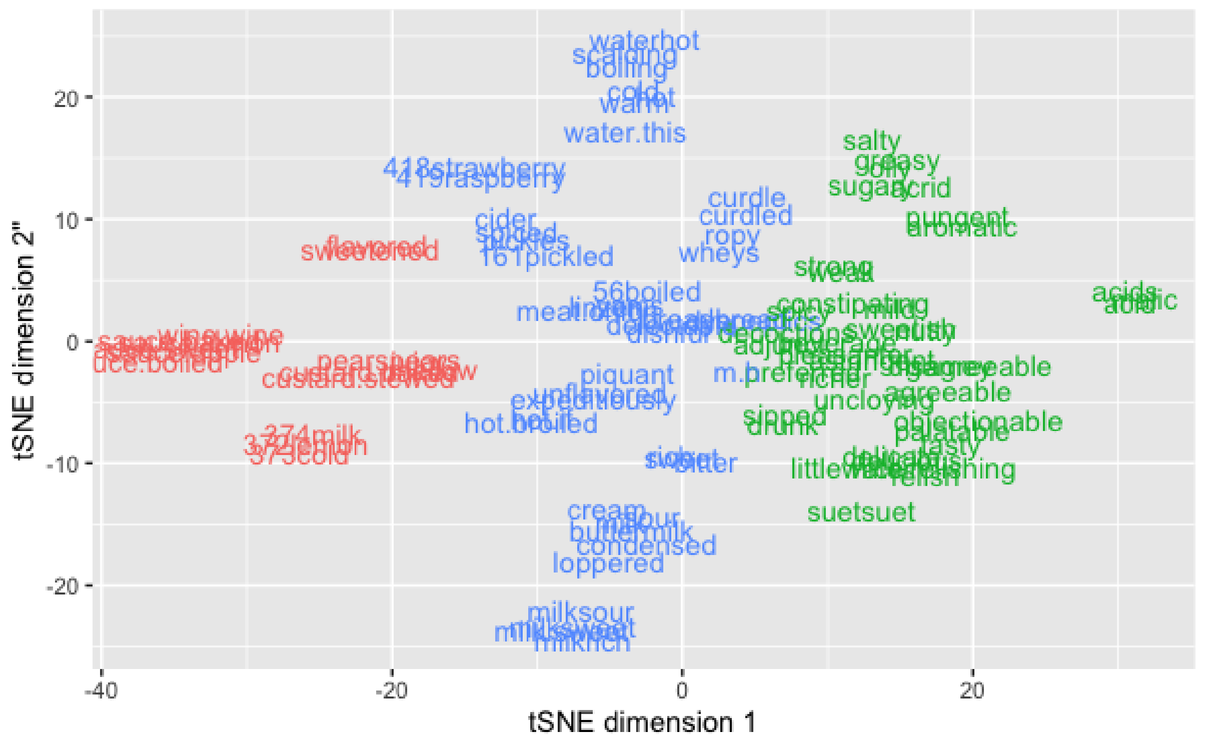


(4)

i use three tastes including hot, spicy and sour.

the results are reasonable in some way, however, the three cluster is not really as dispersed as previously.

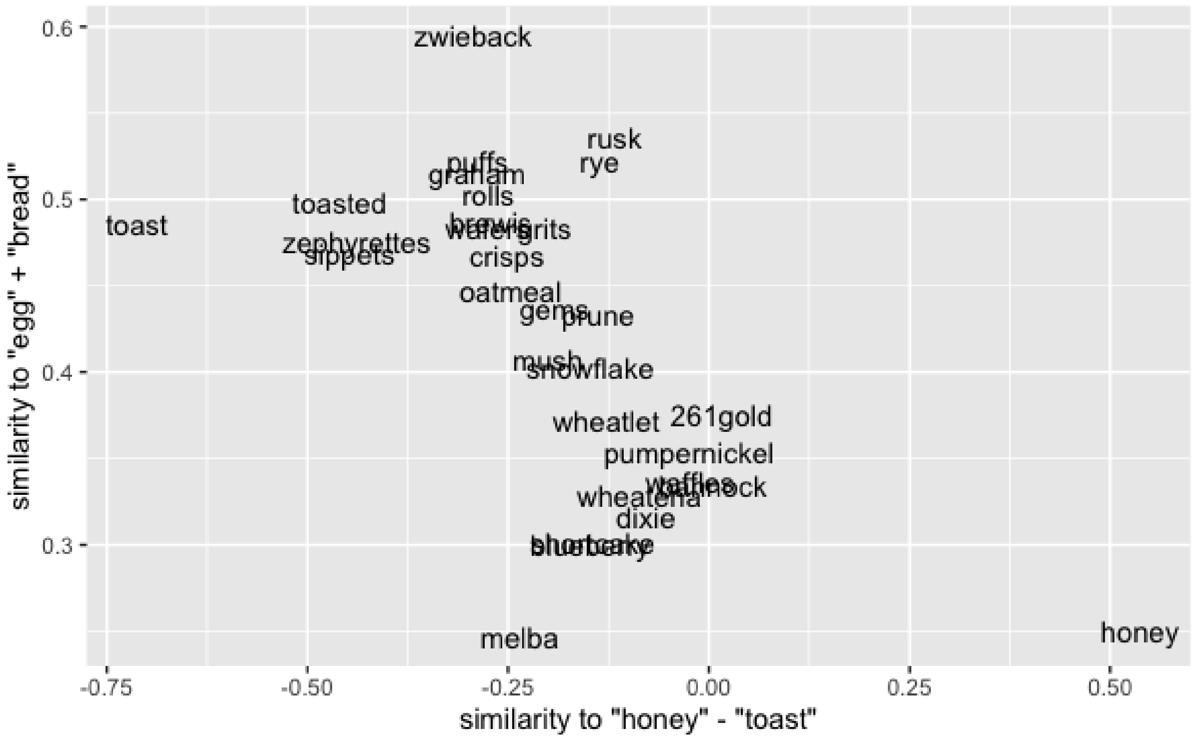




(5)

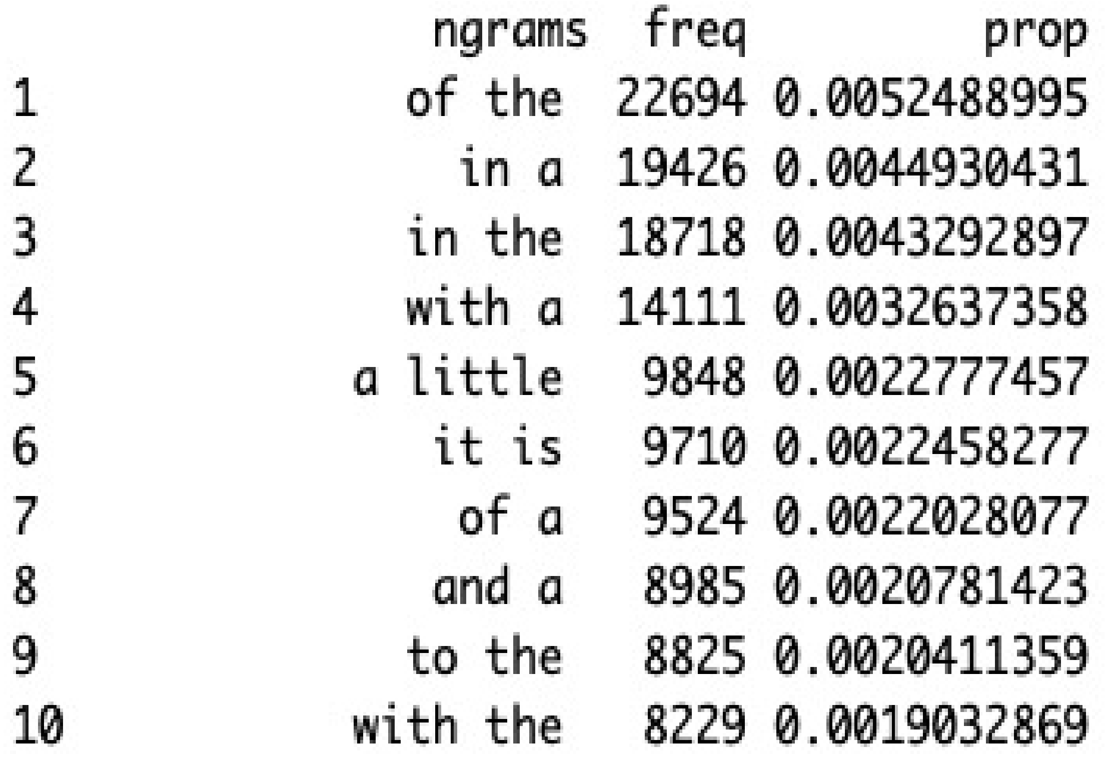
top$_e$$valuative_words = modelclosest_to(\ "honey" + "toast", n = 30)$

$goodness = modelclosest_to(\ ''honey'' - ''toast'', n = Inf)$
$taste = modelclosest_to(\ ''egg'' + ''bread'', n = Inf)$
$top_evaluative_wordsinner_join(goodness)inner_join(taste)ggplot()+geom_text(aes(x = `similarityto''honey'' - ''toast''`, y = `similarityto''egg'' + ''bread''`, label = word))$



(6)
we can see some interesting results like 'toast', 'oatmeal', as they are all common food for breakfast. But its hard to imagine what is the relationship between '261 gold' with above words.

(7)

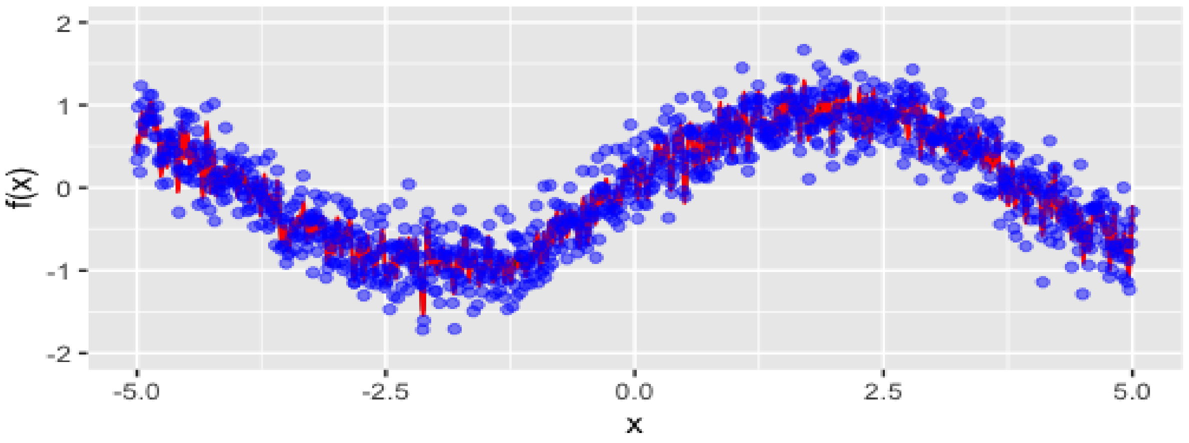


```
          ngrams  freq         prop
1         of the 22694 0.0052488995
2           in a 19426 0.0044930431
3         in the 18718 0.0043292897
4         with a 14111 0.0032637358
5       a little  9848 0.0022777457
6          it is  9710 0.0022458277
7           of a  9524 0.0022028077
8          and a  8985 0.0020781423
9         to the  8825 0.0020411359
10      with the  8229 0.0019032869
```

4 Question4 Gaussian Processes (Fitting a GP)

Refer to Question4.R for implementation.
Part 1
2. theta= 0.001 Negative Log Likelihood is 1872.595
theta= 0.01 Negative Log Likelihood is 5976.789
theta= 0.05 Negative Log Likelihood is 12251
theta= 0.1 Negative Log Likelihood is 21813.76
theta= 0.125 Negative Log Likelihood is 27880.98
It is optimal when theta is 0.001.

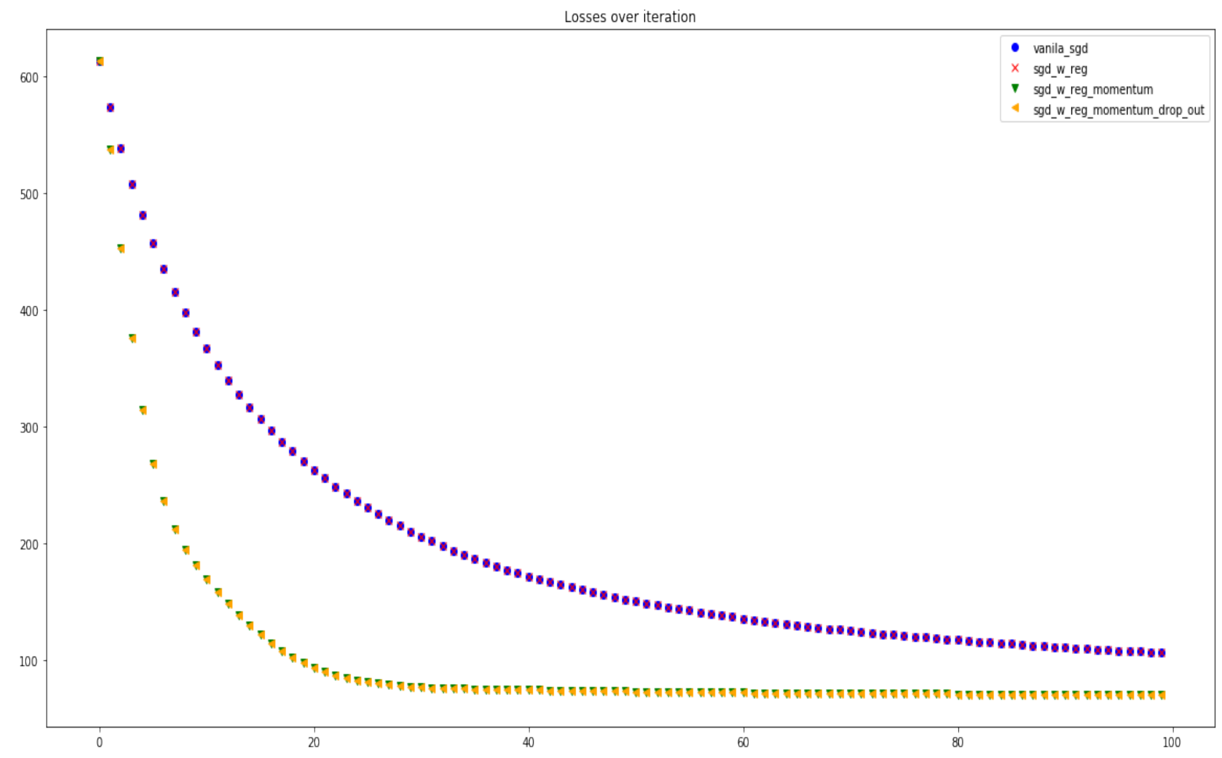
3.



Part 2

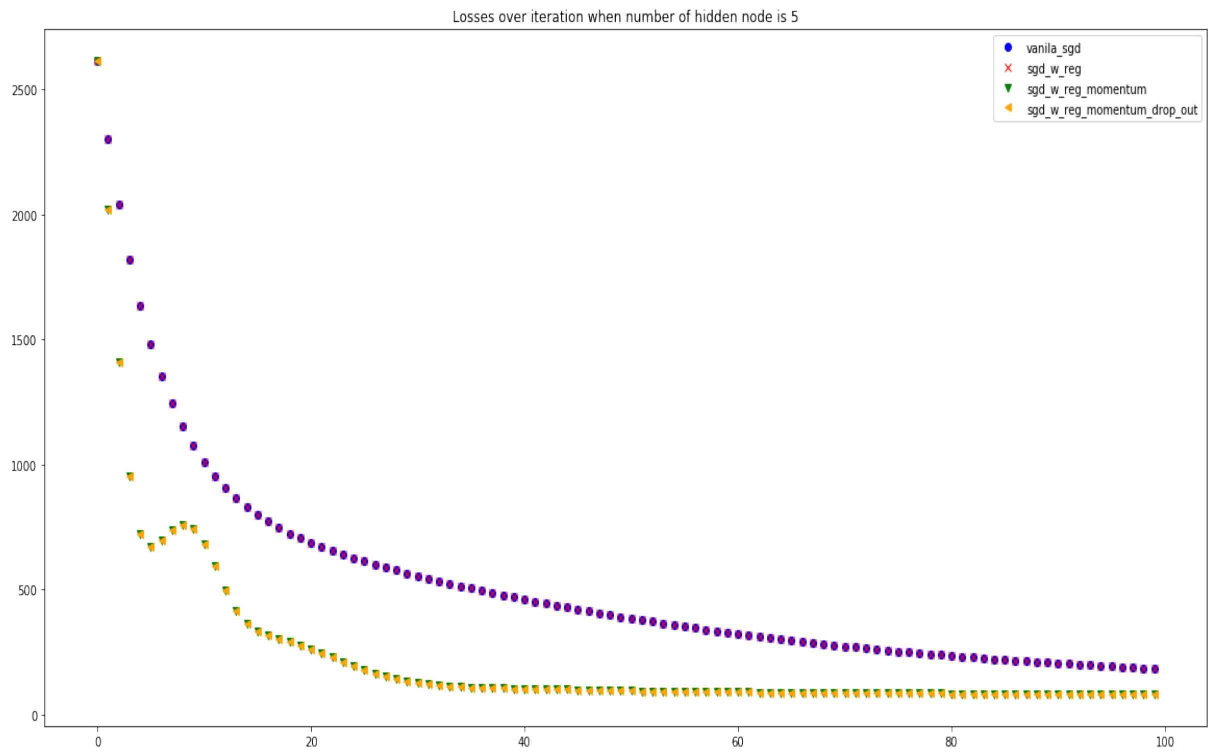
5 Question5 Building Neural Networks

Part 1: For a ReLu network
Part 1.A: Losses



(a) As we observe above, vanilla-sgd and sgd-w-reg have relative similar performance, while sgd-w-momentum and sgd-w-reg-momentum-drop-out have similar performance and they are both better than the previous two methods. And after 100 iterations, sgd-w-momentum and sgd-w-reg-momentum-drop-out achieved same loss.
(b) we can clearly see that momentum methods works better because Momentum helps accelerate gradients in the right direction during training process, thus with same number of iteration, we can see loss drops faster in momentum methods.
However, we can see both regularization and dropout does not affect the performance so much, this suggest that the model may be still under-fitted under 100 iterations.

Part 1.B: Losses when number of hidden node is 5



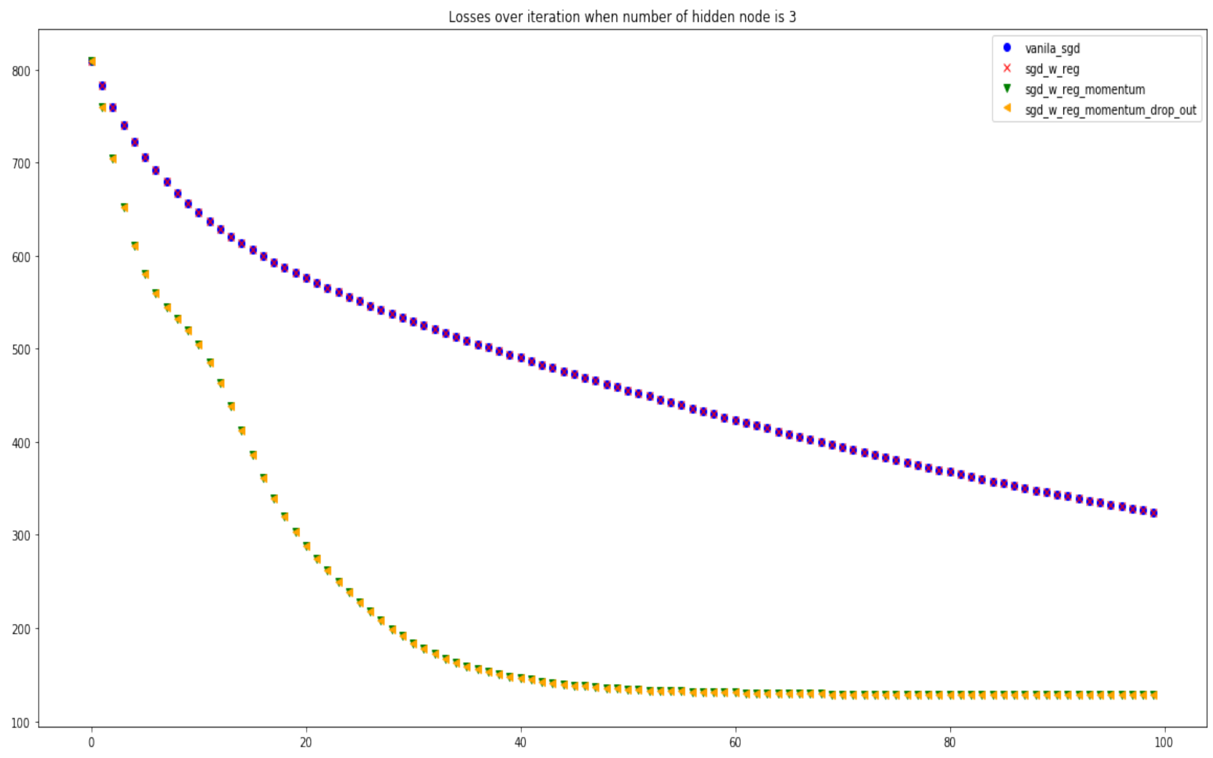
(a) And after 100 iterations, sgd-w-momentum and sgd-w-reg-momentum-drop-out achieved same loss. They are both better than the first two methods.
(b) The performance is not better than when number of hidden node is 3. This may suggests that the model starts to over-fit when number of hidden node is 5.

Part 1.C: Run-Time comparison

Model	**Time**
vanilla-sgd	1.2593
sgd-w-reg	1.2183
sgd-w-momentum	1.2227
sgd-w-reg-momentum-drop-out	1.2224

Part 2: For a Sigmoid network

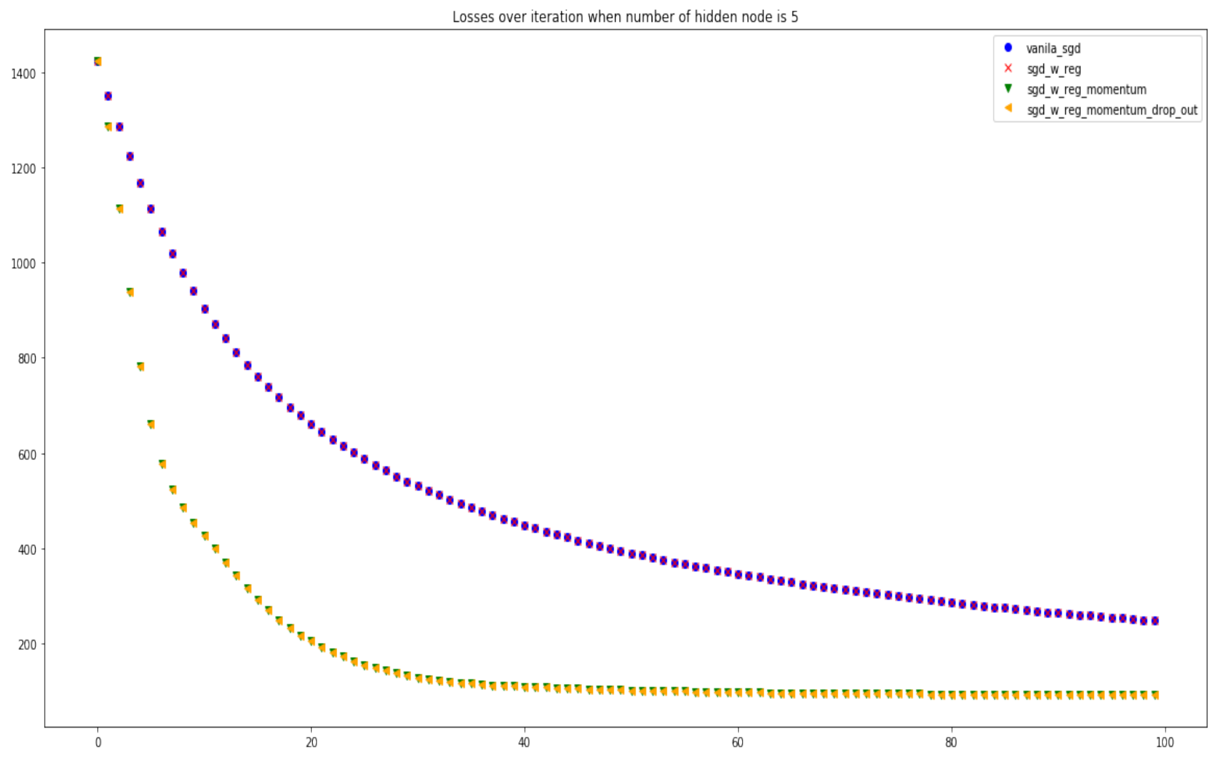
Part 2.A: Losses



As we observe above, vanilla-sgd and sgd-w-reg have relative similar performance, while sgd-w-momentum and sgd-w-reg-momentum-drop-out have similar performance and they are both better than the previous two methods. And after 100 iterations, sgd-w-momentum and sgd-w-reg-momentum-drop-out achieved same loss.

However, we can see both regularization and dropout does not affect the performance so much, this suggest that the model may be still under-fitted under 100 iterations.

Part 2.B: Losses when number of hidden node is 5



And after 100 iterations, sgd-w-momentum and sgd-w-reg-momentum-drop-out achieved same loss. They are both better than the first two methods.
This is better than when number of hidden node is 3, this means we need more complex models to learn better from the data.

Part 2.C: Run-Time comparison

Model	**Time**
vanilla-sgd	1.2548
sgd-w-reg	1.2751
sgd-w-momentum	1.2317
sgd-w-reg-momentum-drop-out	1.2653

Part 2.D: Compare the performance of a ReLu network to a sigmoid network
(1)

Model	**3-node-Relu Loss**	**3-node-Sigmoid Loss**
vanilla-sgd	106.058	324.282
sgd-w-reg	106.069	324.309
sgd-w-momentum	69.996	128.640
sgd-w-reg-momentum-drop-out	69.996	128.640

Relu is generally better.

(2)

Model	5-node-Relu Loss	5-node-Sigmoid Loss
vanilla-sgd	182.366	247.184
sgd-w-reg	182.380	247.210
sgd-w-momentum	79.418	91.377
sgd-w-reg-momentum-drop-out	79.418	91.377

Relu is generally better.
(3) run time comparison on default parameter

Model	3-node-Relu run-time	3-node-Sigmoid runtime
vanilla-sgd	1.2593	1.2548
sgd-w-reg	1.2183	1.2751
sgd-w-momentum	1.2227	1.2317
sgd-w-reg-momentum-drop-out	1.2224	1.2653

run-time is comparable.

6 Question 6 (Activation Functions)

For the code simulation, refer to the Activation Function Study Jupyter Notebook.
please be noted that those chart are generated from random data, and i picked up some of the most representative charts

part1

part1.a

For one layer with one node, we have $y = V * Activation(wx + b) + v_b$: it has following possible shapes for Relu and Sigmoid.

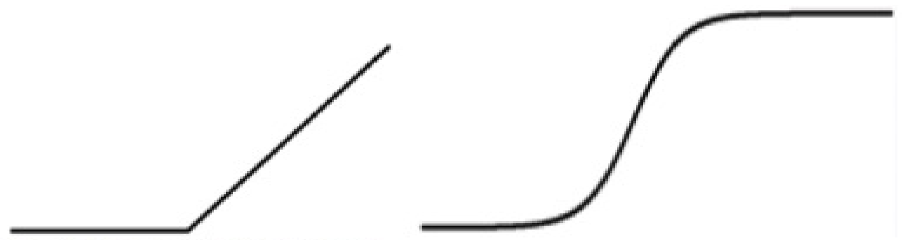


Figure 1 W Positive and V positive

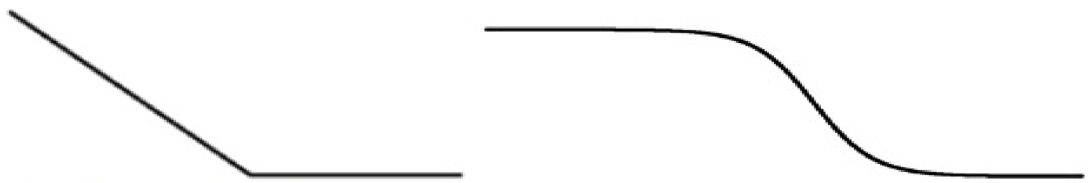


Figure 2 W negative and V positive

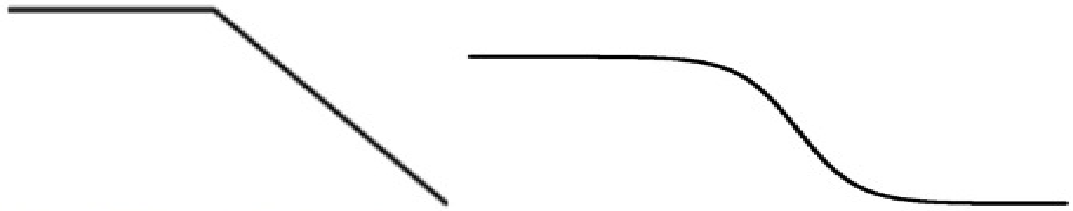


Figure 3 W Positive and V negative

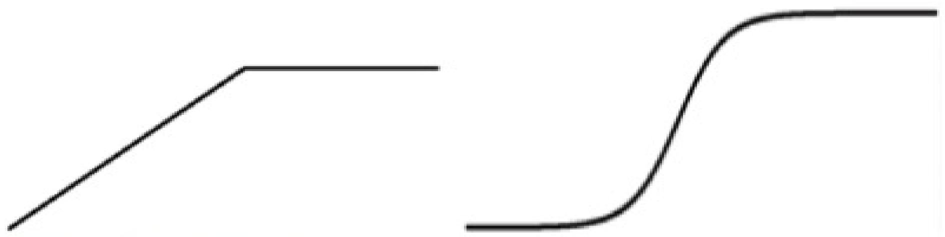
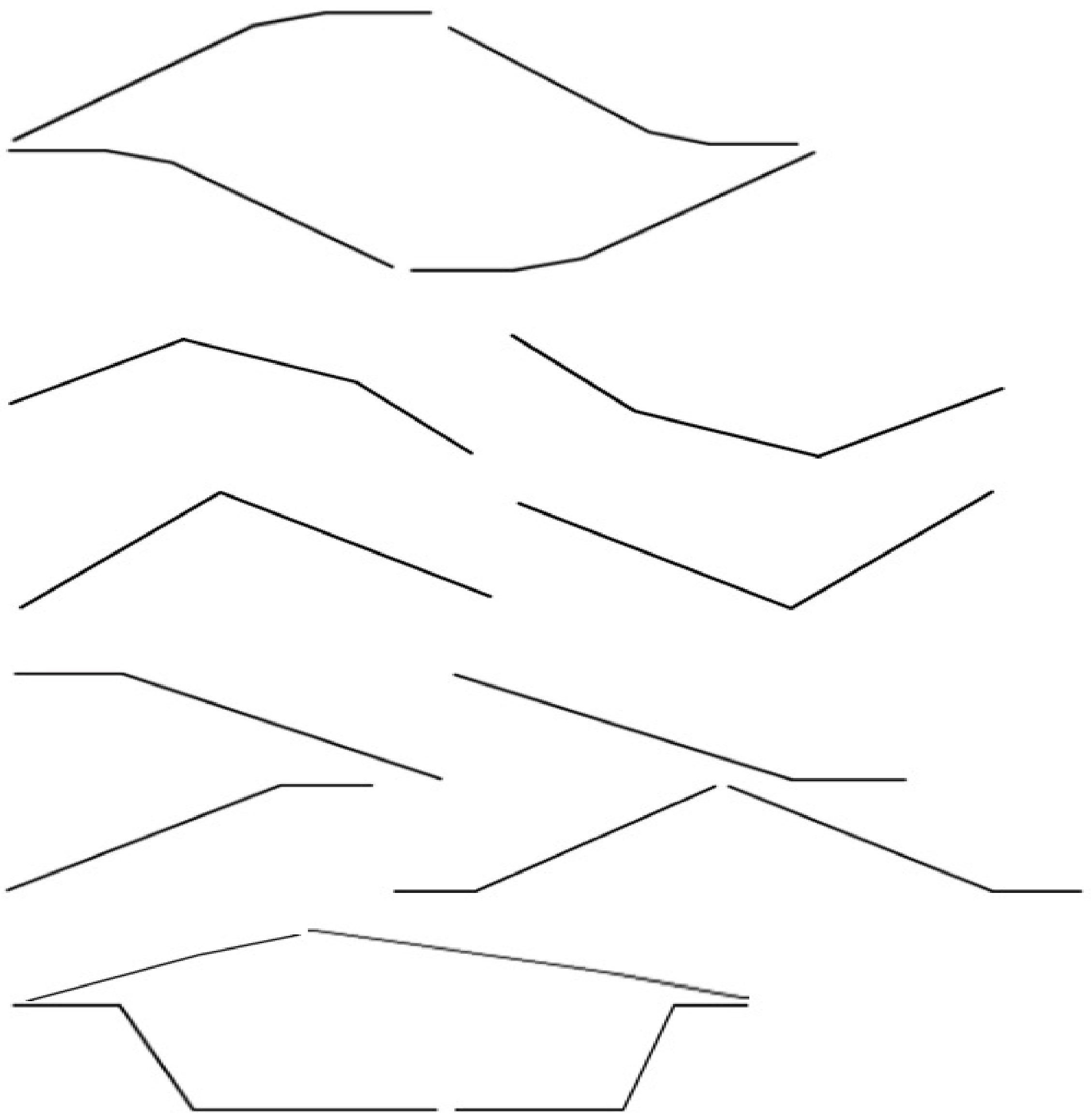
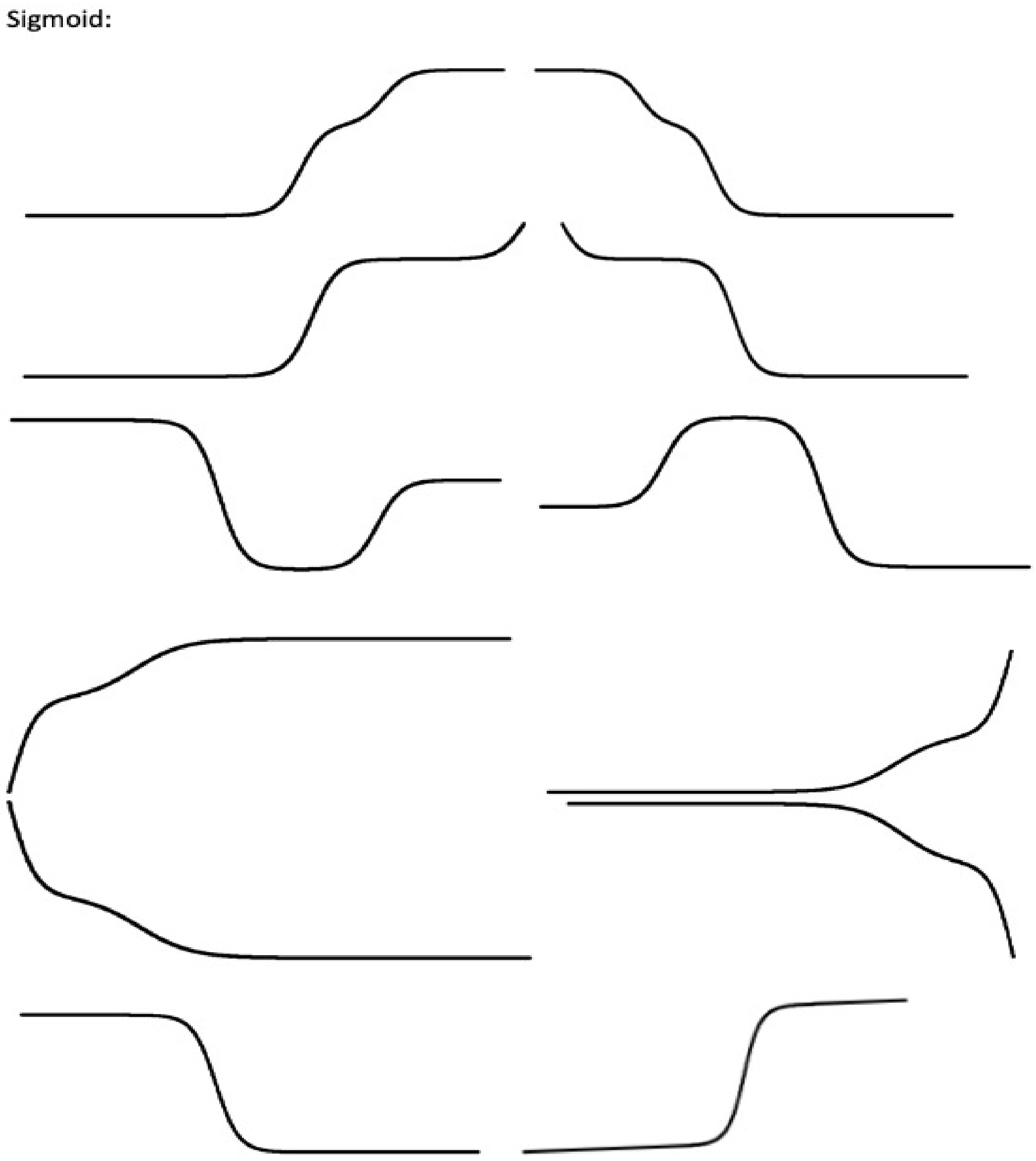


Figure 4 W and V negative

part1.b
For one layer with two node, we have $y = V_1 * Activation(w_1x + b_1) + V_2 * activation(w_2x + b_2) + v_b$: it has following possible shapes.
For ReLu:

Relu:





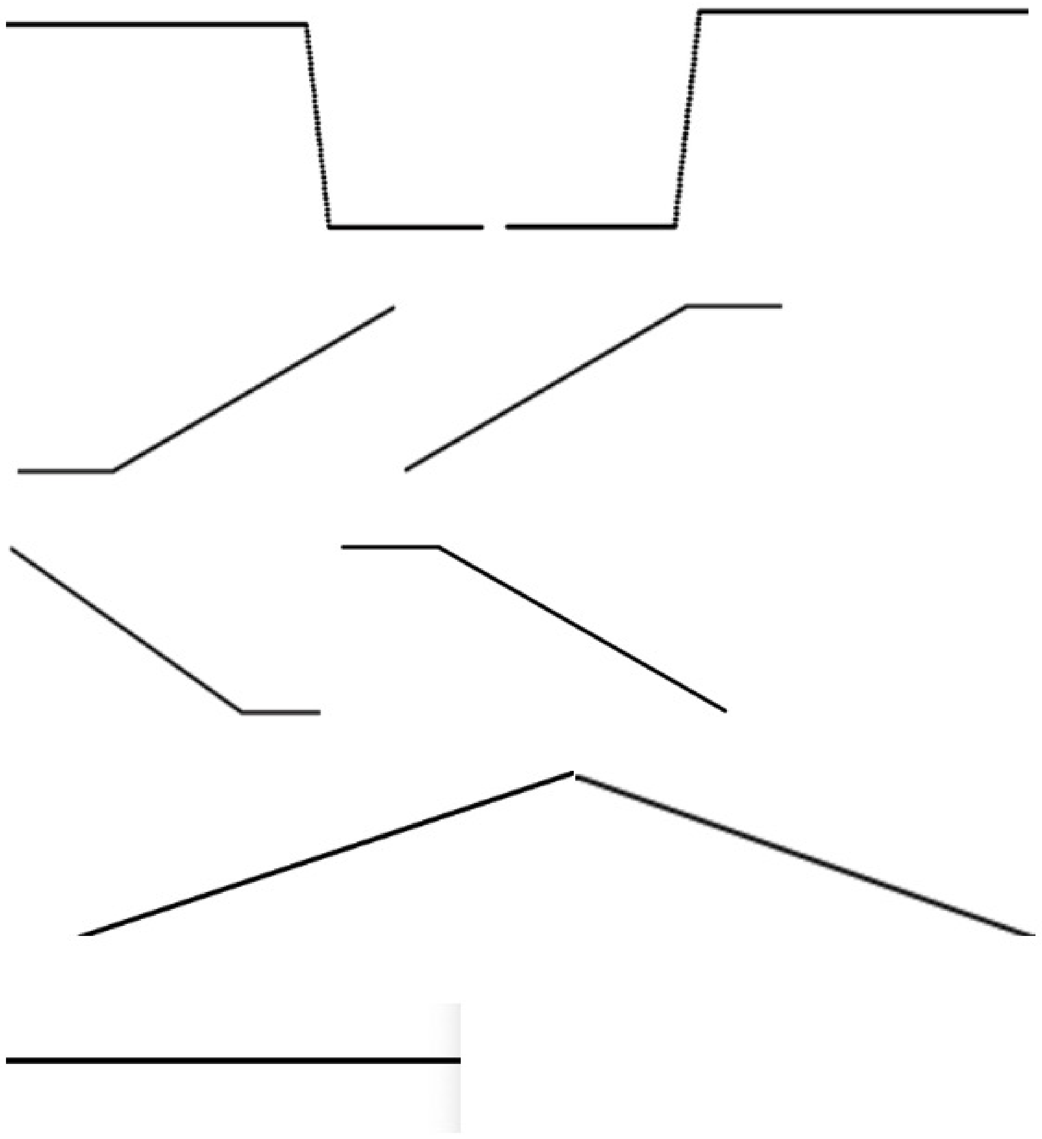
For Sigmoid:

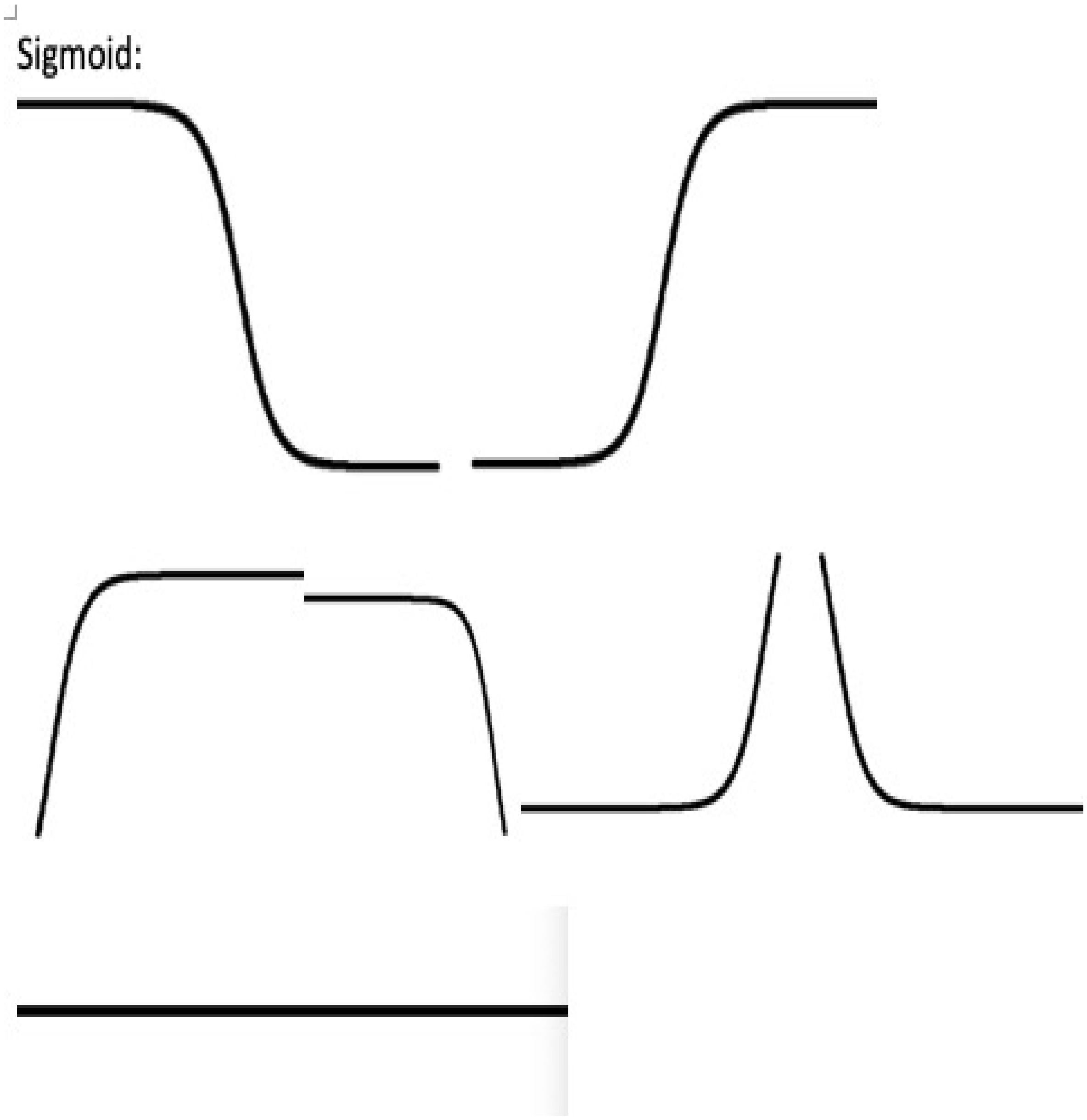
part2

part2.a

for two layers with one node, we have $y = v * Activation(w_2 * activation(w_1 * x + b_1) + b_2) + b$

Relu



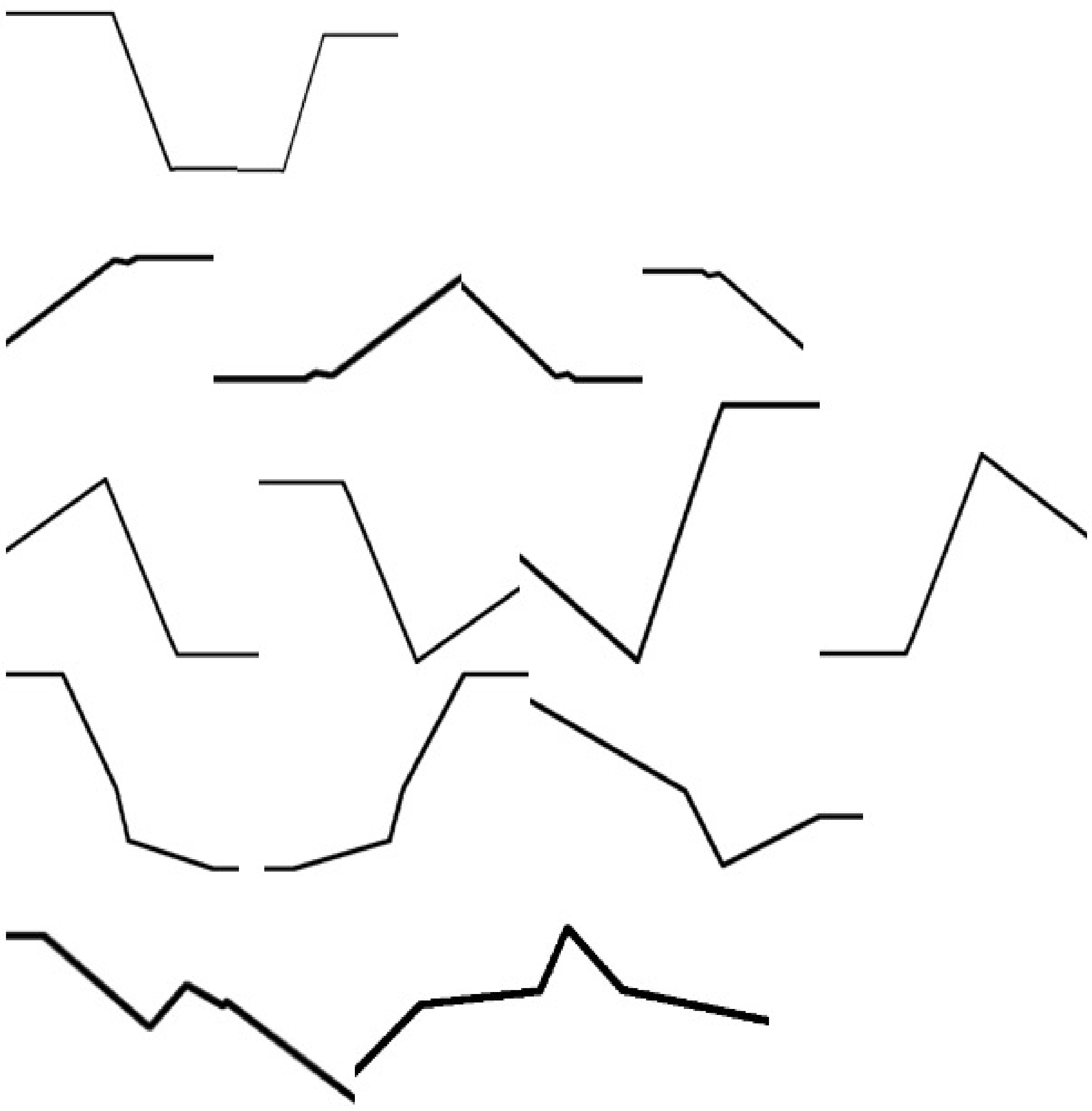


part2.b
For this case, we will have

$y = v1*acti(w11*acti(w1*x1+b1)+w21*acti(w2*x2+b2)+b21)+v2*acti(w12*acti(w1*x1+b1)+w22*acti(w2*x2+b2)+b22)+$

, we will have 8 weights terms and 5 bias term, there are a lot of possibilities. I have listed some of some for both Relu and Sigmoid.
Generally, it could simulate a lot of functions with few spikes/valleys. It is hard to list down all possibilities.

ReLu:



Sigmoid