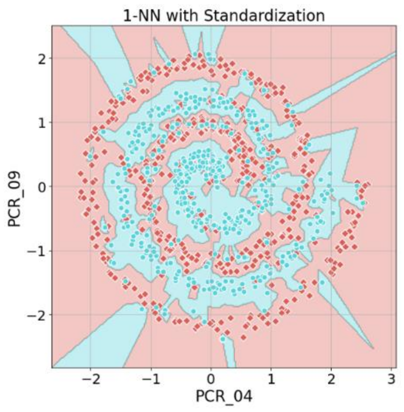
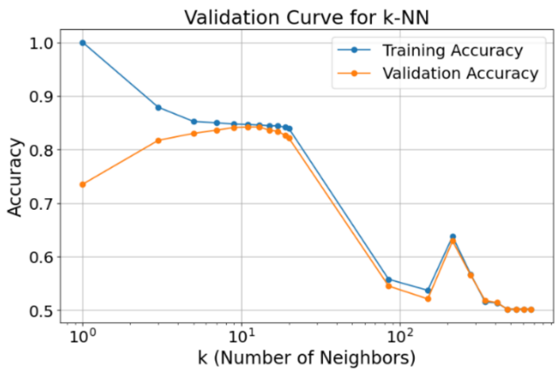
Part 1

A1:



A2:



Best k: **11**

Average Training Accuracy: **0.8466**

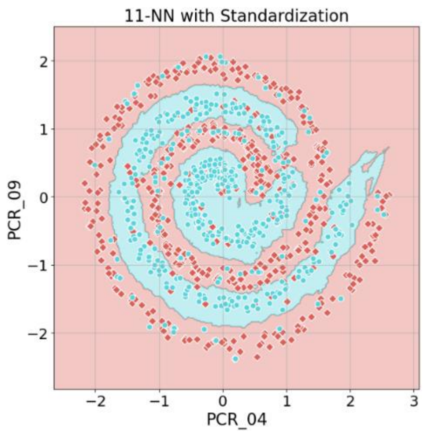
Average Validation Accuracy: **0.8420**

Values that cause underfitting: **85, 150, 215, 280, 345, 410, 475, 540, 605, 670**

Values that cause overfitting: **1, 3**

Underfitting occurs when the model is too simple to capture the structure of the data – resulting in poor performance on the training data and on the testing data.

On the other hand, **overfitting** occurs when the model fits the training data too well, catching outliers and noise that doesn't generalize the data well enough.



A3: the Test Accuracy is **0.5**

A4:

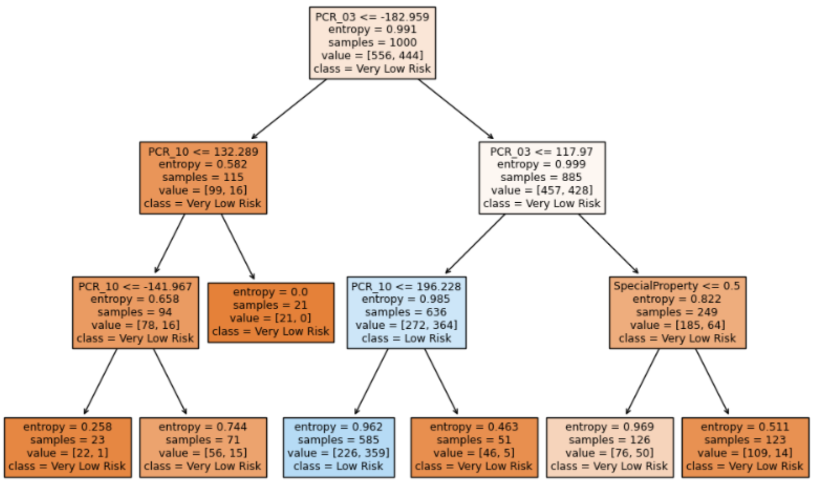
the 1-NN boundary is highly dependant on 1 neigber as suggested and thus is "jittery" and sensitive to noise and outliers in the training data, in comparison, the 11-NN boundary is way smoother and more generalized as it takes into account more of the surronding which makes it less likely to follow noise and outliers as the bigger the surronding concidered, the more it overlaps with the "next-door" surronding concidered from the training data. despite this being said, the smoothness did not transfer into better results as can be observed in its poorer performance (0.5 Accuracy compared to 1-NN's 0.72 Accuracy)

Part 2

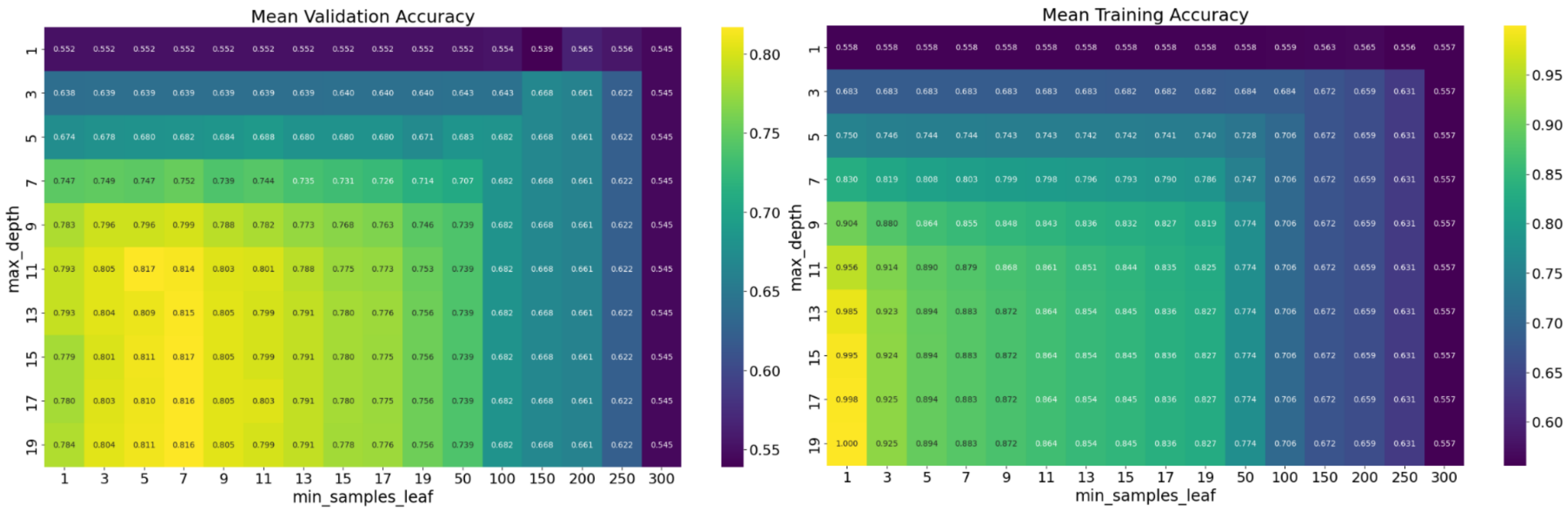
A5:

Training Accuracy: 0.69

'Risk' Decision Tree using PCR_03, PCR_10 and SpecialProperty



A6:



c. Optimal: (max_depth,min_samples_leaf) = (11,5)

d+f. Underfitting: (max_depth,min_samples_leaf) = (1,1), because both validation and training accuracies are low

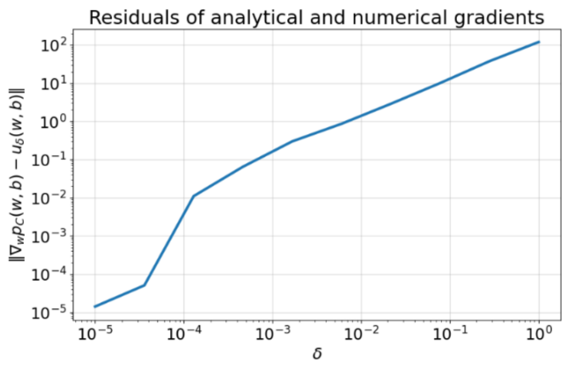
e+f. Overfitting: (max_depth,min_samples_leaf) = (19,1) because it has the highest training\validation difference (0.216 diff)

A7: the number hyperparameter combinations in my grid is: 10*16 = 160, if I wished to add a third hyperparameter the number of combinations would be 160*(amount_of_values) which significantly increases the computational cost and time required for a grid search, which in result would increase the model complexity

A8: Test Accuracy using (max_depth = 11,min_samples_leaf = 5): **0.8520**

Part 3

A9: δ that is too large will make the numeric derivative be further away from the real derivative therefore the norms between the analytical gradient and the numeric gradient will only increase



A10:

C is extremely large therefore the clsasifier will try it's best to missclassify as little as possible(somewhat similar to hardSVM).

looking at the data plot we can see that the data inseperable linearly which is why we won't be able to get high accuracy(around 53% at best). Another thing is that we use hinge-loss which penalizes missclassifications by their distance from the margin(which also affects the accuracy of the classifier).

lr(Learning Rate = step size) is extremely small and as a result it takes a while to find the best accuracy(in our case ~3500 steps)

Train Accuracy:

- initial increase – as expected because the model learns and becomes better at predicting.
- Fluctuations - might be caused by overfitting in a case where the model learned the noise of the training data

Train Loss:

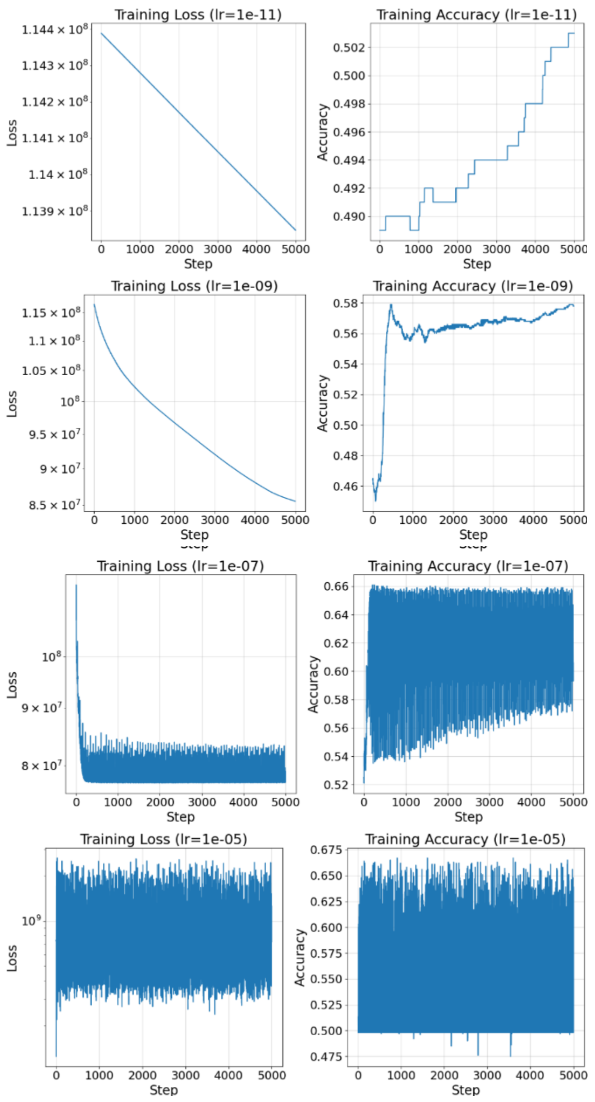
- Initial decrease - the loss starts at a very high values and initially it decreases rapidly – as expected because the model is quickly learning and reducing errors on the training data.
- Plateauing – it gets to a point where further training doesn't significantly reduce the training error

A11:

I didn't include the 1e-03 plot here but it is similar to the 1e-05 plot;

The learning rate I'd choose is **1e-09** because it seems to have the best Training Loss & Accuracy plots as they are not as jittery as the 1e-05&1e-07 plots(too big of a step resulting in the model not being able to converge to a minimum as good as the models with the smaller step)

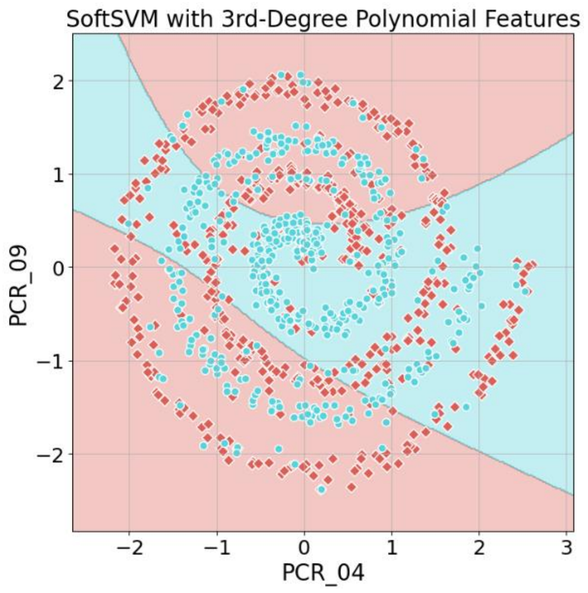
Also, the training accuracy of the 1e-09 is better than of the 1e-11 because it takes way less steps to get to the maximum accuracy



A12:

Training Accuracy: 0.59

Test Accuracy: 0.57



Part 4

A13:

a) $\boldsymbol{h(x) = sign(\sum_{i=1}^{m} a_i y_i K(x, x_i)) = \ sign(\sum_{i=1}^{m} y_i e^{-\gamma\|x_i - x\|_2}) =}$

$\boldsymbol{sign\left(\sum_{i\in\{j\in[m]:y_j=1\}} y_i e^{-\gamma\|x_i-x\|_2} + \sum_{i\in\{j\in[m]:y_j=-1\}} y_i e^{-\gamma\|x_i-x\|_2}\right) =}$

b) $\boldsymbol{= sign\left(\sum_{i\in\{j\in[m]:y_j=1\}} e^{-\gamma\|x_i-x\|_2} - \sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma\|x_i-x\|_2}\right) =}$

$\boldsymbol{sign\left(\sum_{i\in\{j\in[m]:y_j=1\}} K(x, x_i) - \sum_{i\in\{j\in[m]:y_j=-1\}} K(x, x_i)\right)}$

c) $\boldsymbol{\sum_{i\in\{j\in[m]:y_j=1\}} K(x, x_i) = \ \sum_{i\in\{j\in[m]:y_j=1\}} e^{-\gamma\|x_i-x\|_2} >}$

$\boldsymbol{\sum_{i\in\{j\in[m]:y_j=1\}} e^{-\gamma(\delta-1)}}$ (given γ is positive) $>$

$\boldsymbol{e^{-\gamma(\delta-1)}}$ $for\ e^{-x}\ is\ positive\ for\ all\ x$

d) $\boldsymbol{\sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma\|x_i-x\|_2} < \ \sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma 3\delta} <}$

$\boldsymbol{\sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma(2\delta+1)} < \sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma(2\delta-1)} = \ \frac{m}{2} e^{-\gamma(2\delta-1)}}$

e) $\boldsymbol{\sum_{i\in\{j\in[m]:y_j=1\}} e^{-ln(\frac{m}{2})\|x_i-x\|_2} - \ \sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\ln\left(\frac{m}{2}\right)\|x_i-x\|_2} >}$

$\boldsymbol{e^{-\ln\left(\frac{m}{2}\right)(\delta-1)} - \frac{m}{2} e^{-\ln\left(\frac{m}{2}\right)(2\delta-1)} \ = \ e^{-\ln\left(\frac{m}{2}\right)\delta + \ln\left(\frac{m}{2}\right)} - \frac{m}{2} e^{-\ln\left(\frac{m}{2}\right)(2\delta-1)} =}$

$\boldsymbol{\frac{m}{2}(e^{-\ln\left(\frac{m}{2}\right)\delta} - \ e^{-\ln\left(\frac{m}{2}\right)(2\delta-1)}) = \frac{m}{2}(e^{\ln\left(\frac{m}{2}^{-\delta}\right)} - e^{\ln\left(\frac{m}{2}^{-(2\delta-1)}\right)}) =}$

$\boldsymbol{\frac{m}{2}\left(\frac{m}{2}^{-\delta} - \frac{m}{2}^{-(2\delta-1)}\right) = \left(\frac{2}{m}^{\delta-1} - \frac{2}{m}^{2(\delta-1)}\right) = \ x - \ x^2}$

what is left is to show this is always positive which means 0 < x < 1

the left side is obviose as $\frac{2}{m}^{\delta-1} > \boldsymbol{0}$ as $\frac{2}{m}^{\square}$ is positive. The right hand side is proven as following: $\log\left(\frac{2}{m}^{\delta-1}\right) < \log(1)$

$\delta - 1 > 0\ since\ the\ base\ is\ smaller\ then\ 1 \log is\ negative.$
$always\ true.$

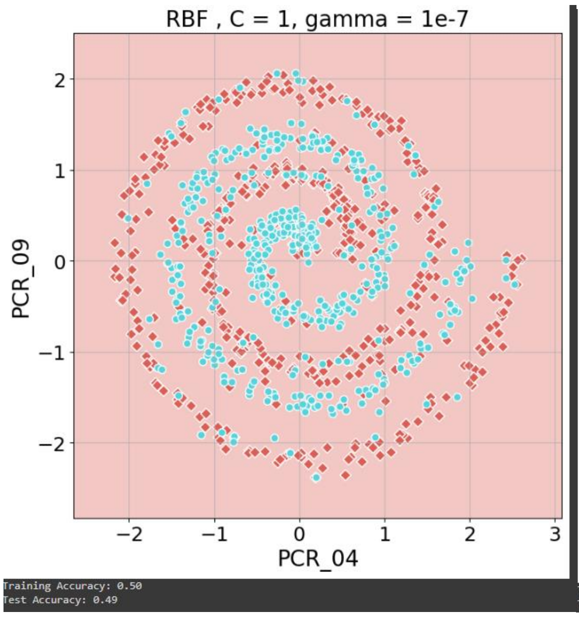
f) weve seen that $\boldsymbol{\sum_{i\in\{j\in[m]:y_j=1\}} K(x, x_i) >}$

$\boldsymbol{\sum_{i\in\{j\in[m]:y_j=1\}} e^{-\gamma(\delta-1)}}$ (given γ is positive) $> \boldsymbol{\sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma(2\delta-1)}} >$

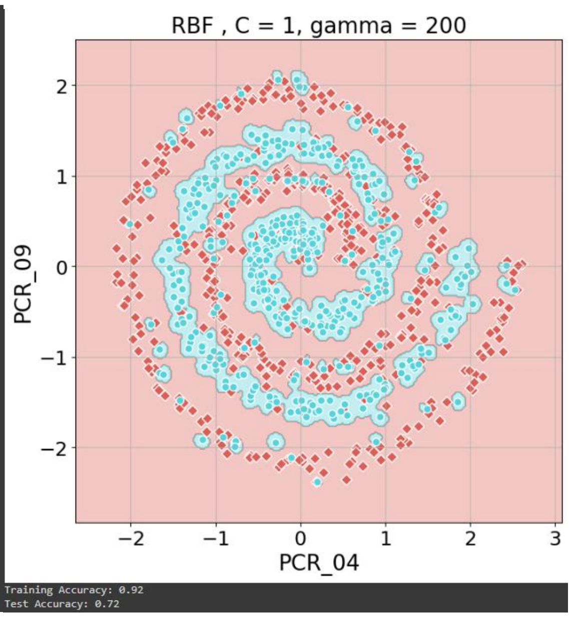
$\boldsymbol{\sum_{i\in\{j\in[m]:y_j=-1\}} e^{-\gamma\|x_i-x\|_2} = \ \sum_{i\in\{j\in[m]:y_j=-1\}} K(x, x_i)}$

so $sign(\boldsymbol{\sum_{i\in\{j\in[m]:y_j=1\}} K(x, x_i) - \ \sum_{i\in\{j\in[m]:y_j=-1\}} K(x, x_i)) = 1}$

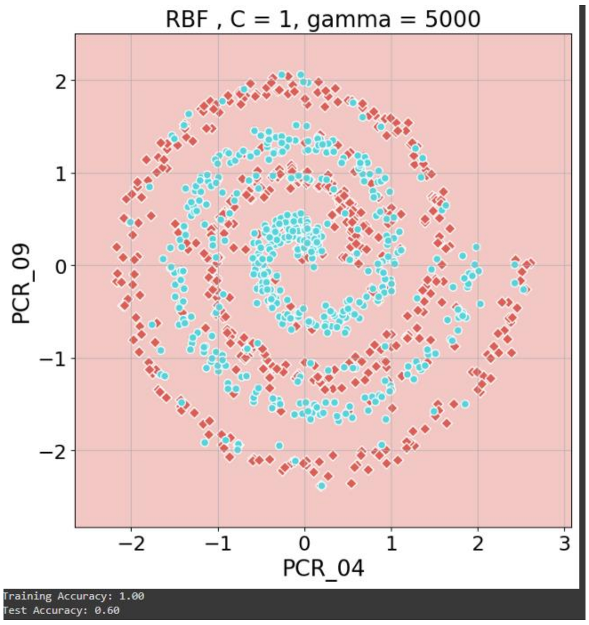
A14, A15, A16:



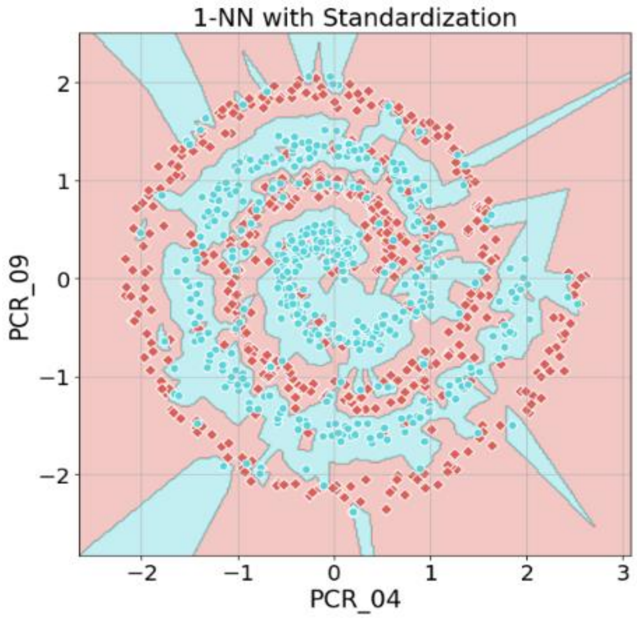
```
Training Accuracy: 0.50
Test Accuracy: 0.49
```



```
Training Accuracy: 0.92
Test Accuracy: 0.72
```



```
Training Accuracy: 1.00
Test Accuracy: 0.60
```



Looking at the figure below and at our results, it becomes even clearer that when sigma→ 0 , gamma → ∞, we should get a similar behavior (up to edge cases) to the 1-nearest-neighbor on the support vectors, although it seems different:
($a_i y_i$*exp{-gamma||x-x_i||} != 0) which means only x_i's very close to x are summed up (overfitting!) as the exp for other vectors can be considered epsilon in such a case, thus we get a majority vote on cases that are not “close” to x (in this case red won) explaining the difference from the 1NN case on the gamma = 5000. (as gamma >> ||X-x|| gamma can be considered -> inf). For gamma = very small we get the opposite effect as can be shown below where each vector has a larger support base and in the infinite case where gamma -> 0 we get exp(0) = 1, we result with sign{sum over i of $a_i y_i$} (everyone is in the support!) again a majority vote, only this time the nearest neighbor holds epsilon = 1/n weight on the decision, thus the whole h(x) will assign a label based on an agreed majority label for all x, i.e extremely underfitting which correlates with the accuracy result we received in both cases (which have a majority of red in them).

Figure (a)

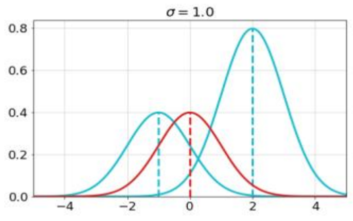
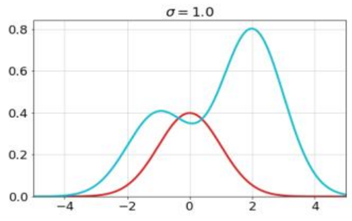
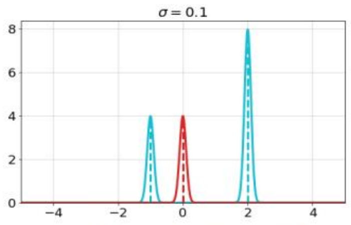


Figure (b)



Only very closely to $x = 0$, the model predicts $y = -1$.

Figure (c)



Each point affects only a small environment around it