AI HW4

tags: `AI` `HW`

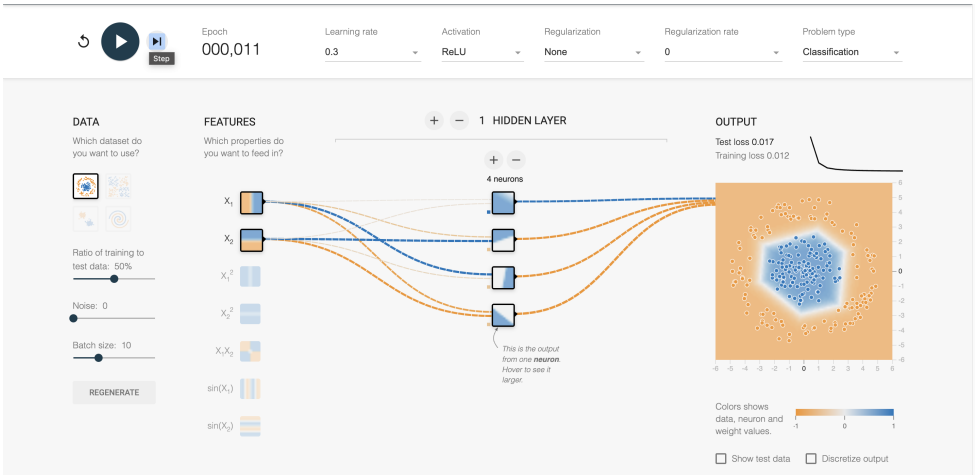
41047025S 王重鈞

hackmd link to protect TA eyes: **AI HW4** (https://hackmd.io/@4y-lwOz-TsG_bfIr6Wx73A/BJDAma7H3)

(A)

Using X1 X2

Feature: X1, X2
Hidden layer: 4 neurons
Test loss: 0.017
Epoch: 11
Learning rate: 0.3
Activation: ReLU



Using X1^2 X2^2

Feature: X1^2, X2^2

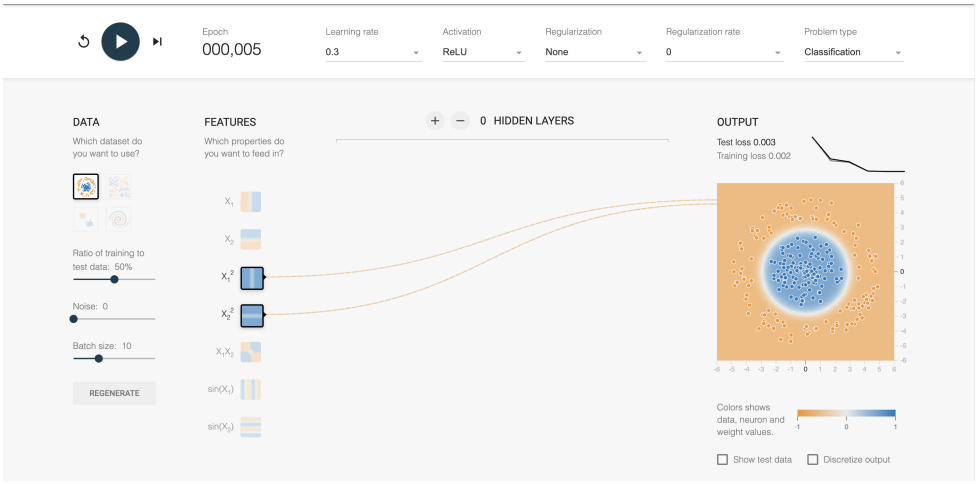
Hidden layer: None

Test loss: 0.003

Epoch: 5

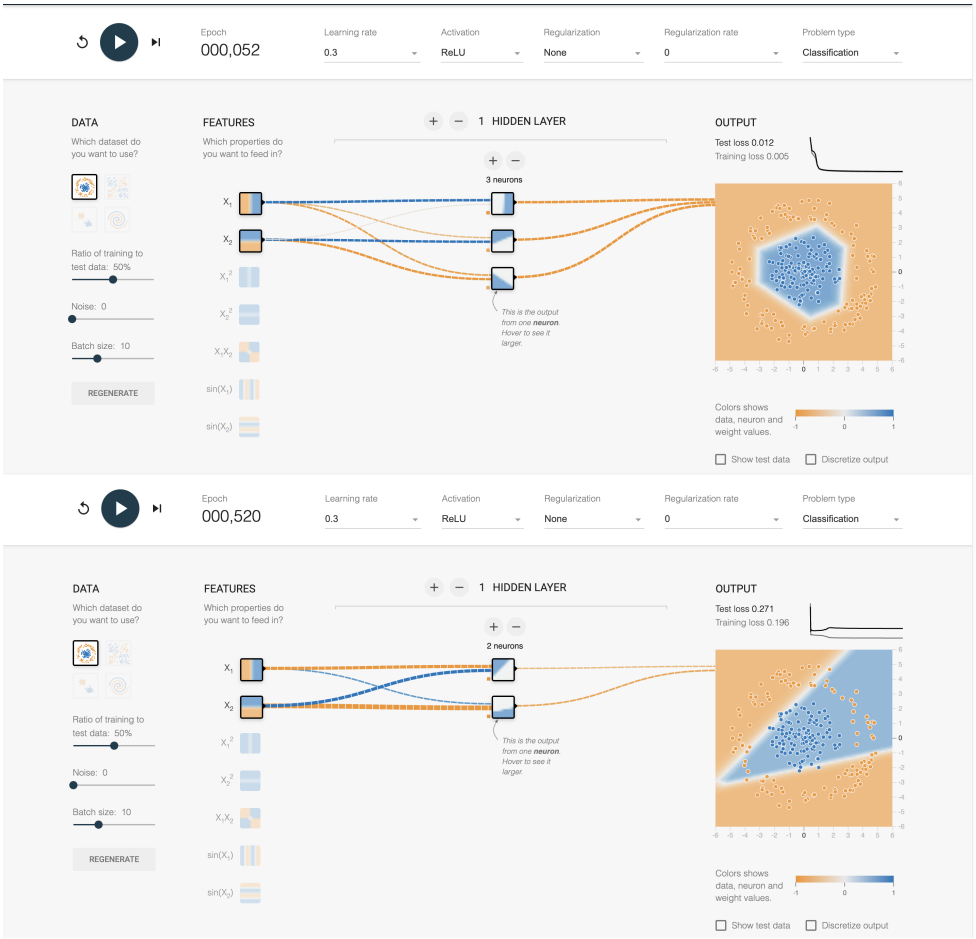
Learning rate: 0.3

Activation: ReLU



Why

I think test2 is better because X1^2 and X2^2 feature can extract center feature easiler. By X1^2 and X2^2, it can get center datt by up-down and left-right. However, X1 and X2 are linear, it would need four line to surround center data. To prove it, I use **3** neurons and **2** neurons in test1.



For 3 neurons, it still can extract center data, but in 2 neurons, two line can't surround an area.

Conclusion

So if want to classify this data, the network need to have ability to get a surround area.

(B)

Learning rate 0.03 and tanh

Feature: X1, X2

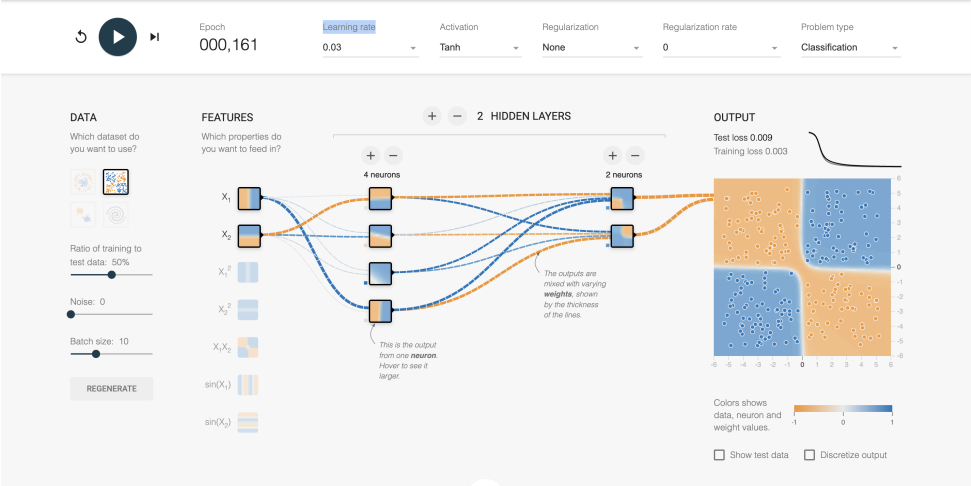
Hidden layer: 4 neurons, 2 neurons

Test loss: 0.009

Epoch: 161

Learning rate: 0.03

Activation: Tanh



Using 3, 2 hidden layer

Feature: X1, X2

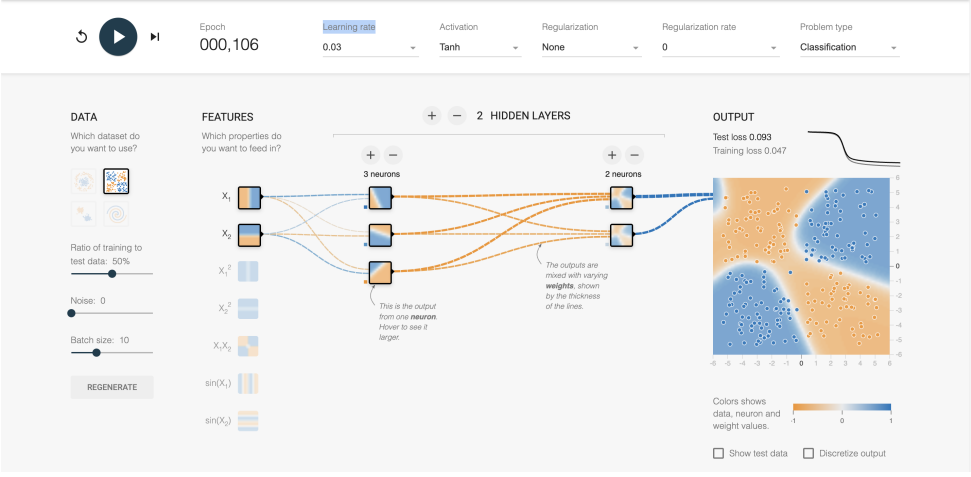
Hidden layer: 3 neurons, 2 neurons

Test loss: 0.093

Epoch: 106

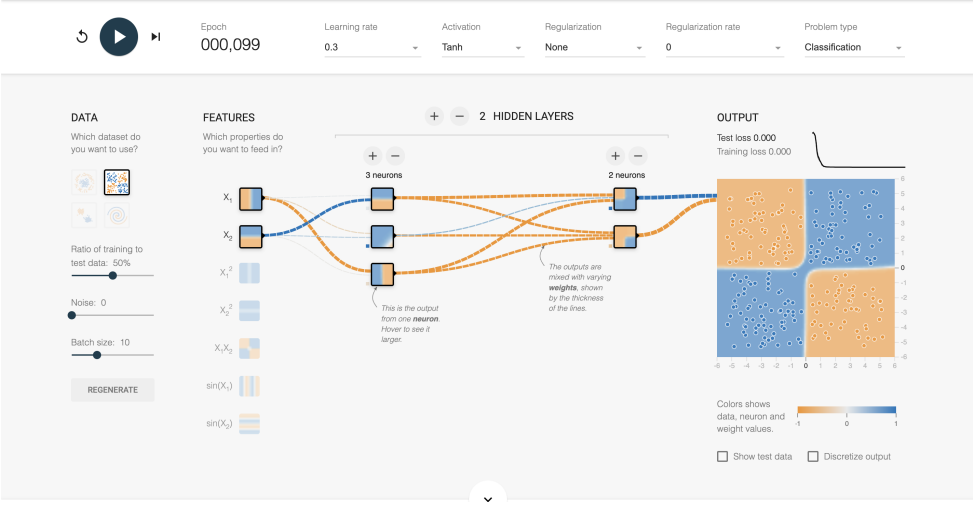
Learning rate: 0.3

Activation: Tanh

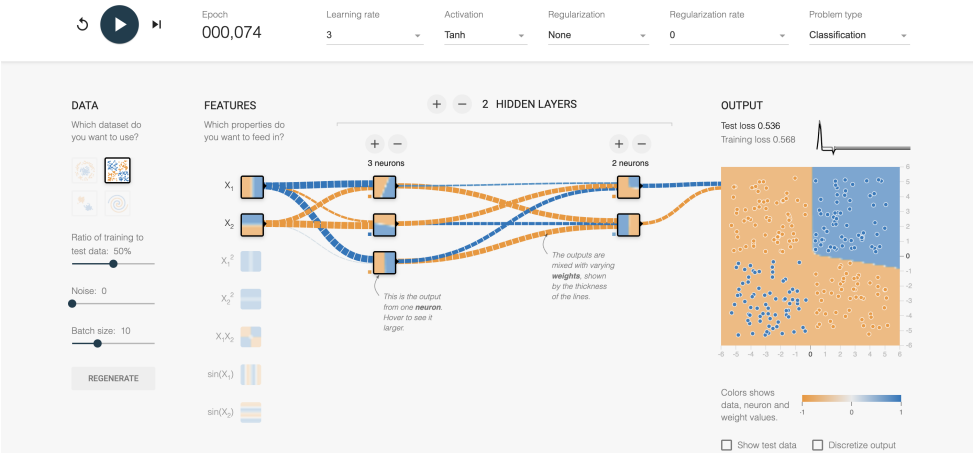


Differ learning rate

Learning rate: 0.03

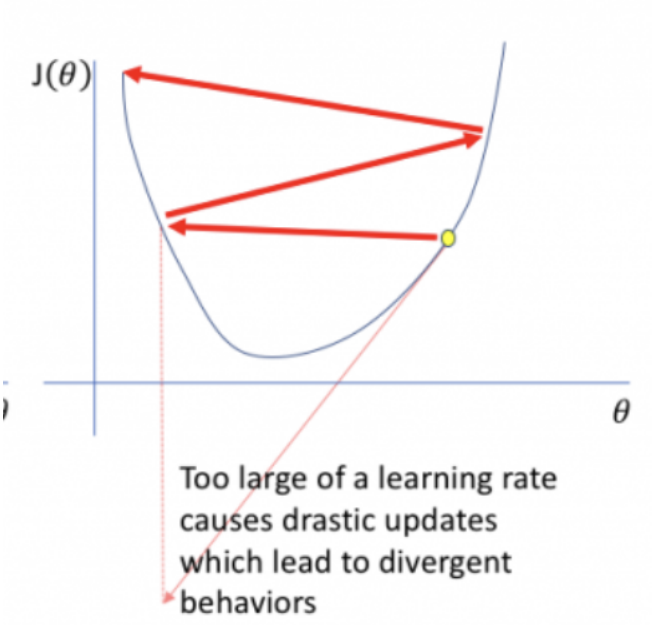


Learning rate: 3



Why

For Learning rate = 3, it step to large, so it would always miss best answer, like this.



Another Network

Feature: X1, X2

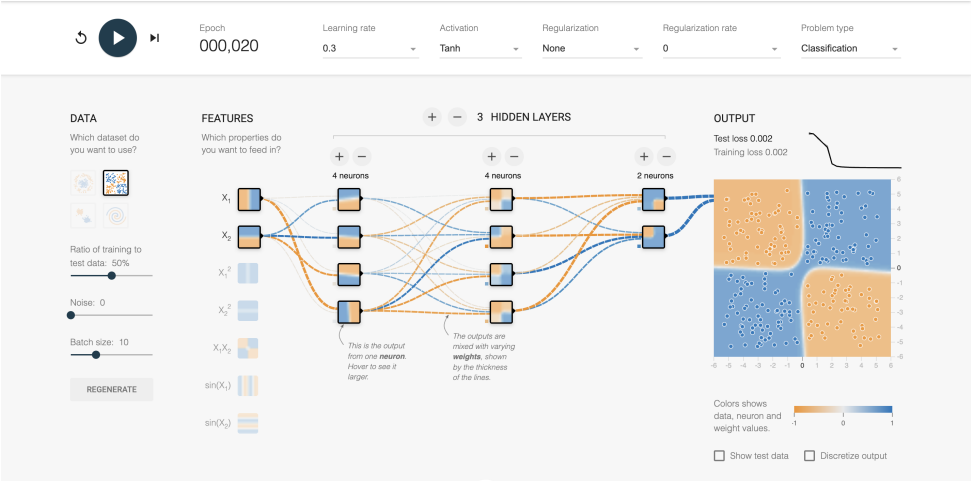
Hidden layer: 4 neurons, 4 neurons, 2 neurons

Test loss: 0.002

Epoch: 20

Learning rate: 0.3

Activation: Tanh



(C)

Using X1 and less nueron

ANS: YES

Feature: X1

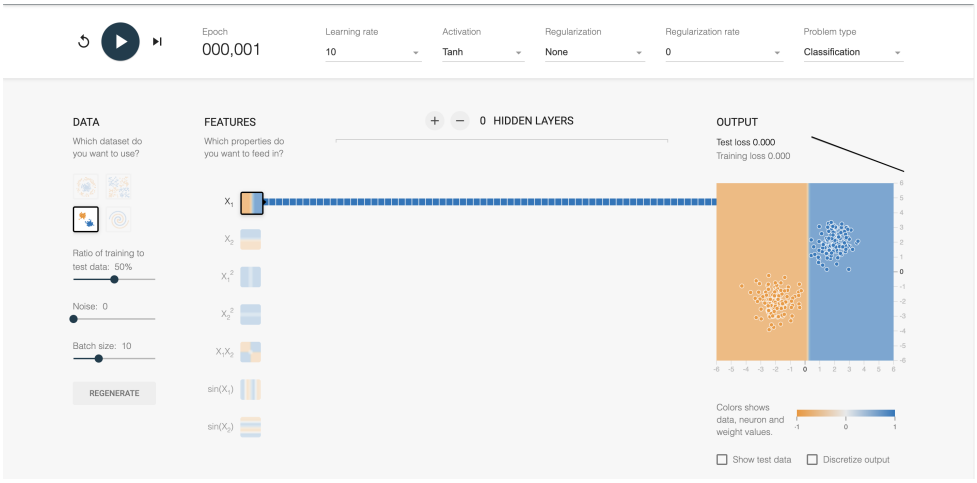
Hidden layer: None

Test loss: 0.000

Epoch: 20

Learning rate: 10

Activation: Tanh



Using X1^2 X2^2

ANS: NO

I try about 20 network, each test loss hardly decrase.

I guess the X1^2 X2^2 are symmetry feature, so it's difficult to train better network.

(D)

Using X1 X2

Feature: X1, X2

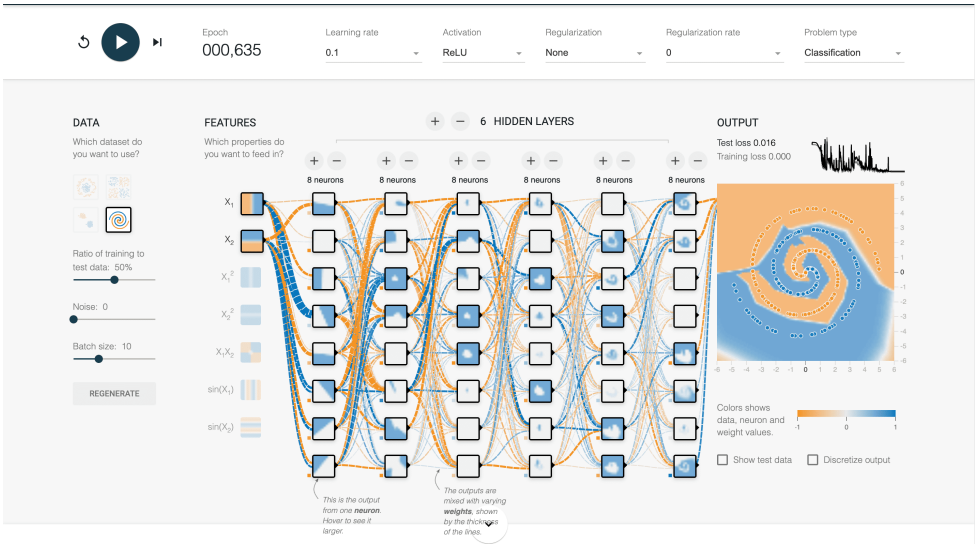
Hidden layer: 8 neurons * 8

Test loss: 0.0016

Epoch: 635

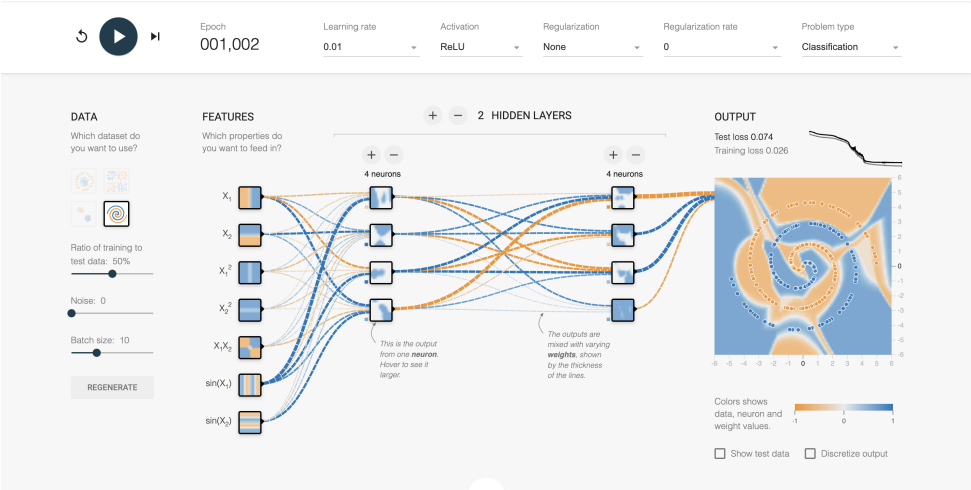
Learning rate: 10

Activation: ReLU



Using 7 feature

Feature: X1, X2, X1^2, X2^2, X1X2, sin(X1), sin(X2)
Hidden layer: 4 neurons, 4 neurons
Test loss: 0.074
Epoch: 1002
Learning rate: 0.01
Activation: ReLU



(E) (F)(G)(H)

I think reoreder is better to understand

Noise

Noise is means that some data would be tagged wrong. Such as a dog picture and it be tagged cat. This noise would cause overfitting and other situation.

Overfitting

Overfitting means the network is in order to fit train data due to noise data or some extreme data, lead to the network performence bad in test data.

We want to prevent overfitting, so we can use regulation.

Regulation

The purpose of regulation is in order to prevent overfitting. When updating the error for an parameter, we may use MSE to update it, such as following:

$$J(\Theta) = [y(\Theta) - y] ^ 2$$

> J(Θ): error
> Θ：parameter in network
> $y(\Theta)$：parameter through network
> y:Answer

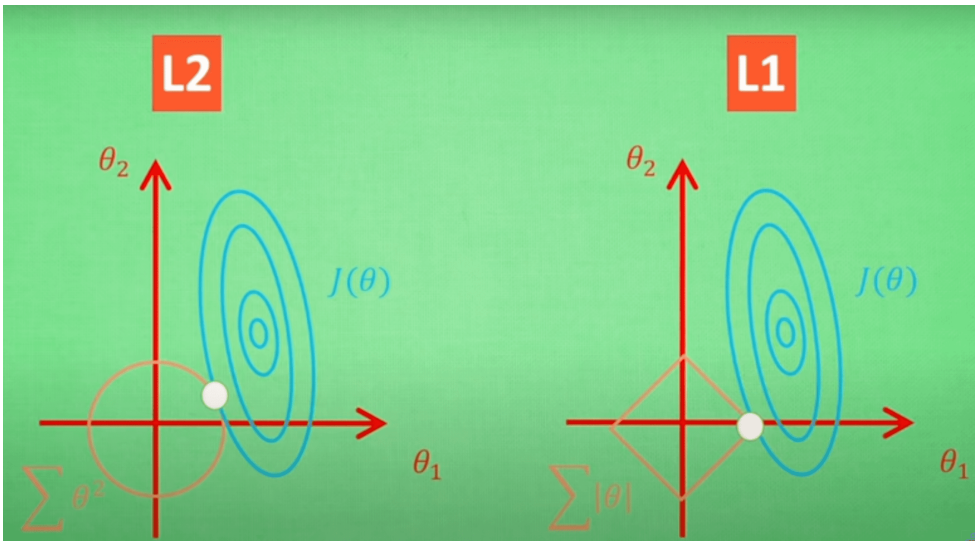
If $J(\Theta)$ is large, means there would be an noise data lead to large J(Θ). We don't want this noise data affect our parameter change drastically. So we use regulation, following is L1 regulation:

$$J(\Theta) = [y(\Theta) - y] ^ 2 + (|\Theta1| + |\Theta2| + |\Theta3|...)$$

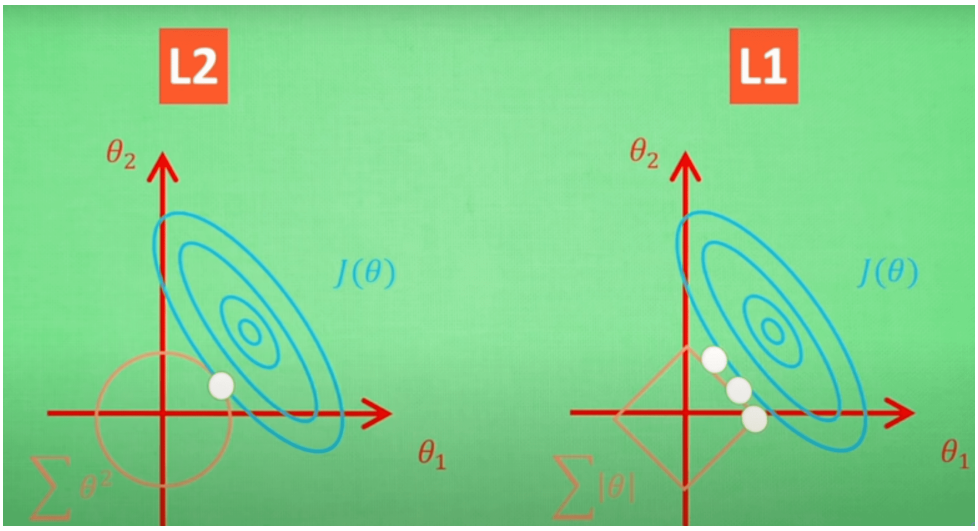
By doing so, if current data is good data(it can improve network to good direction). $[y(\Theta) - y] ^ 2$ and $(|\Theta1| + |\Theta2| + |\Theta3|...)$, so the error will fix correctly. However, it's a noise data, $[y(\Theta) - y] ^ 2$ will be large, but $(|\Theta1| + |\Theta2| + |\Theta3|...)$ is small compared to good data. Make $J(\Theta)$ wouldn't change drastically.

Different of L1 and L2

L1: $J(\Theta) = [y(\Theta) - y] ^ 2 + (|\Theta1| + |\Theta2| + |\Theta3...)$
L2: $J(\Theta) = [y(\Theta) - y] ^ 2 + (\Theta1^2 + \Theta2^2 + \Theta3^2...)$
What's different? We can to two picture to explain (Here's picutre reference from 莫烦Python):
Suppose we have two parameter(Θ1, Θ2) want to update. The white dot is the least loss. We can find that L1 ignore the Θ2. So, if we want to train the network which need extract some important feature, we can use L1 regulation. Otherwise, we can use L2 regulation.



However, when some situation, the least loss in L1 would change drastically:



We can find, the least loss would change drastically. So, compared to L2, L1 is unstable relatively.

Regualtion rate

In regualtion, we add a parameter to control the regualtion affection. Take L1 for instance:

L1: \(J(Θ) = [y(Θ) - y] ^ 2 + λ(|Θ1| + |Θ2| + |Θ3...)\)

λ: regualtion rate

This λ is regualtion rate, we can adjust regualtion rate to control the regualtion affection.

(I)

Batch size is the number of data be feeding into network once. If there's no batch, we feeding all data into network and backpropagation once, the accuracy increasing, but the memory would explosion. If there's no batch, we feeding one data and backpropagation once, first the time will explosion,

second the noise and extreme data would make network explosion.
So we divide all data into some batch, after train all batch once, we call it’s epoch.

Reference

Noise:
https://medium.com/機器學習基石系列/機器學習基石系列-5-noise和error-52351ac29dce
(https://medium.com/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E5%9F%BA%E7%9F%B3%E7%B3%BB%E5%88%97/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E5%9F%BA%E7%9F%B3%E7%B3%BB%E5%88%97-5-noise%E5%92%8Cerror-52351ac29dce)
Overfitting:
https://allen108108.github.io/blog/2019/10/22/L1 , L2 Regularization 到底正則化了什麼 _ /
(https://allen108108.github.io/blog/2019/10/22/L1%20,%20L2%20Regularization%20%E5%88%B0%E5%BA%95%E6%AD%A3%E5%89%87%E5%8C%96%E4%BA%86%E4%BB%80%E9%BA%BC%20_/)
Regulation:
https://www.youtube.com/watch?v=TmzzQoO8mr4
(https://www.youtube.com/watch?v=TmzzQoO8mr4)
Batch:
https://www.cupoy.com/qa/club/ai_tw/0000016D6BA22D97000000016375706F795F72656C656173654B5741535354434C5542/0000017D085DB5E3000000026375706F795F72656C656173655155455354
(https://www.cupoy.com/qa/club/ai_tw/0000016D6BA22D97000000016375706F795F72656C656173654B5741535354434C5542/0000017D085DB5E3000000026375706F795F72656C656173655155455354)