

Pima Indians Diabetes Database-Predict the onset of diabetes based on diagnostic measures

深度學習：基礎及應用

106753020 資科碩三 李俊毅

108753205 資科碩一 蔡宗諺

108753208 資科碩一 葉冠宏



INTRODUCTION

- **Datasets is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.**
- **The objective of the dataset is to diagnostically predict whether or not a patient has diabetes.**
- **All patients here are females at least 21 years old of Pima Indian heritage.**



INTRODUCTION

- Datasets(Real number input, Binary output)

Variables	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction	Diabetes pedigree function
Age	Age (years)



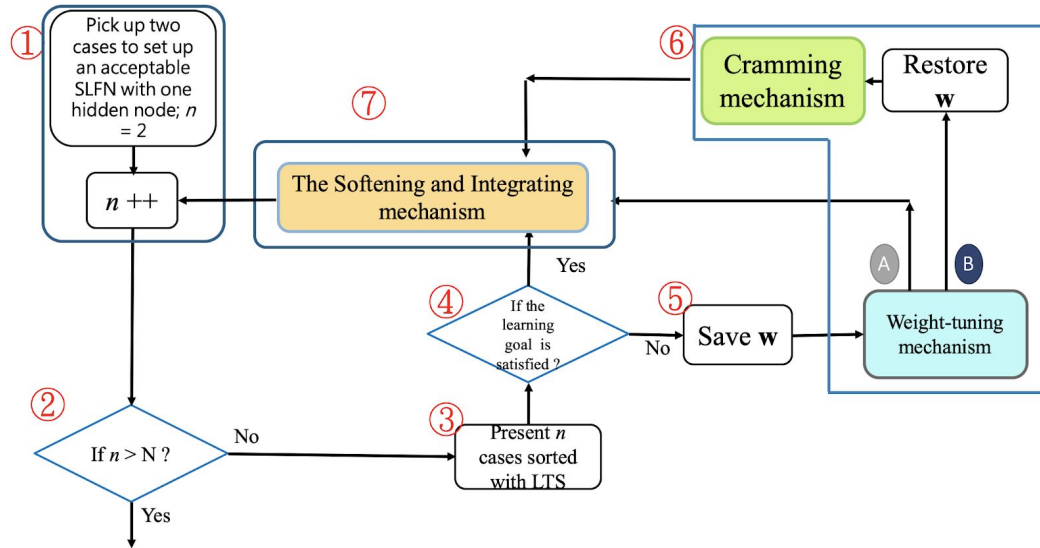
RELATED WORK

- Classification ability of single hidden layer feedforward neural networks.
- Sales forecasting using extreme learning machine with applications in fashion retailing.
- Single-layer learning revisited: a stepwise procedure for building and training a neural network.



METHODS

Single Hidden Layer Feedforward Neural Network(SLFN)





METHODS

- Data Preprocessing

1. Use MinMaxScaler to preprocess the data.
2. Train only two data set to get 100% prediction rate and initial weight.

$$X_{\text{new}} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$



METHODS

Literature Review

Least Trimmed Squares (LTS)

(Tsaih & Cheng, 2009; Tsaih, et al., 2018)

- Pick up the **first** n reference observations $\{(x^c, y^c)\}$ which are **sorted** by **all** N reference observations' **squared residuals** in **ascending** order.

$$(e^{(1)})^2 \leq (e^{(2)})^2 \leq \dots \leq (e^{(n)})^2, \text{ in which } (e^{(c)})^2 = (f(x^c, w) - y^c)^2$$

- Detect the potential outliers in (Tsaih & Cheng, 2009; Tsaih, et al., 2018)
- Accelerate the learning process and mimic the human learning

```
File Edit Selection Find View Goto Tools Project Preferences Help
project.py x.py
818 K.set_value(model.layers[0].weights[1], r2)
819 K.set_value(model.layers[1].weights[0], r3)
820 K.set_value(model.layers[1].weights[1], r4)
821
822 x_50=np.array(Xtrain_nor[0:10,:])
823 y_50=np.array(Y_train[0:10,:])
824
825 model=LTS(x_50,y_50,model,ini_Minode,ini_Reg)
826 LTS(Xtrain_nor,Y_train,model,ini_Minode,ini_Reg)
827 model.save_weights('my_model_weights.h5')
828
829 x_test=np.array(Xtrain_nor[51,:]).reshape(-1,8)

C:\Users\chrysa21\Desktop\AI_project\project.py - Sublime Text (UNREGISTERED)
2
[0.00242325051209491, 0.18064471894673773]
[0]
[1]
LTS starts
3
[0.00242325051209491, 0.18064471894673773, 0.2022512639921139]
[0]
[1]
LTS starts
4
[0.00242325051209491, 0.006930679288164043, 0.18064471894673773, 0.2022512639921139]
[0, 0]
[1, 1]
LTS starts
```



METHODS

Literature Review

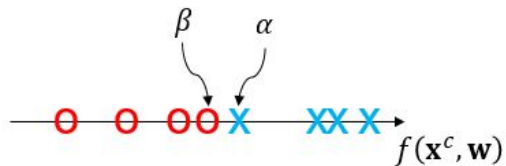
Condition \mathcal{L}

(Tsaih, 1993)

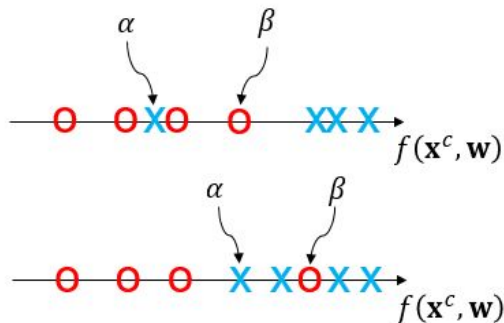
$y^c = 1$ all $c \in \mathcal{I}_1$; $y^c = -1$ all $c \in \mathcal{I}_2$

\times : $f(\mathbf{x}^c, \mathbf{w})$, all $c \in \mathcal{I}_1$

\circ : $f(\mathbf{x}^c, \mathbf{w})$, all $c \in \mathcal{I}_2$



$\alpha > \beta$
Condition \mathcal{L} : True

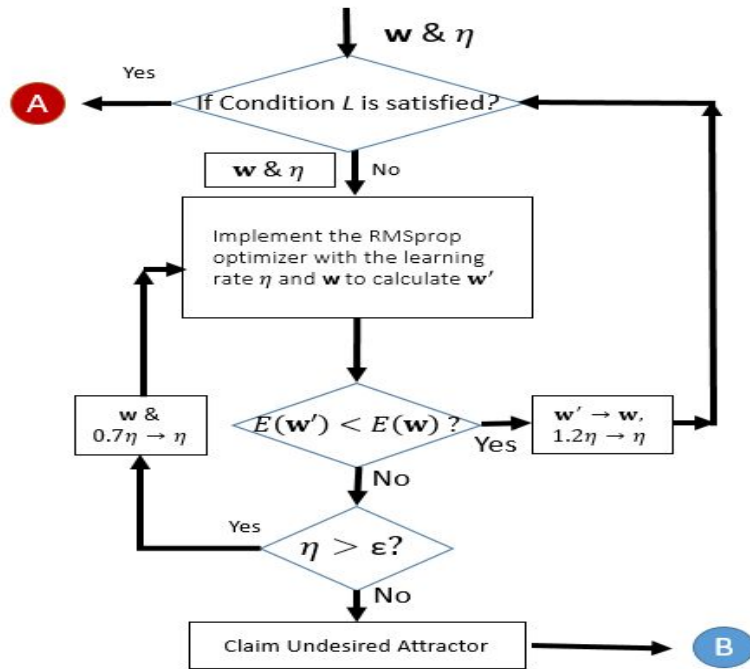


$\alpha < \beta$
Condition \mathcal{L} : False

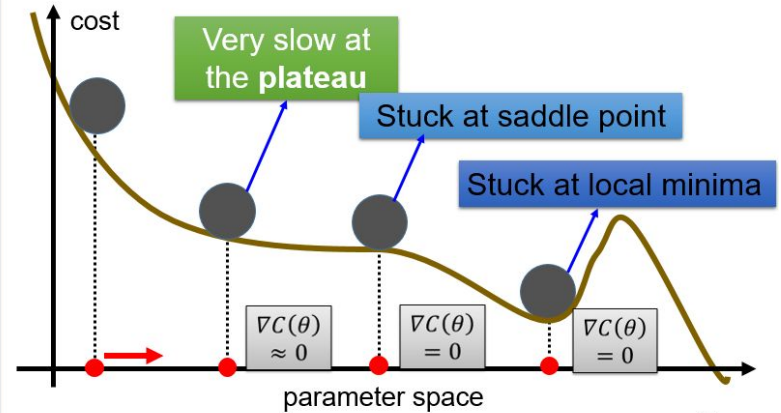


METHODS

Weight-tuning mechanism



You need to deal with undesired attractors. Not only for the learning purpose, but for the inferring.





METHODS

Weight-tuning mechanism

```
project.py
• 814 x_50=np.array(A11*data1*data1*(0.20,))
• 815 y_50=np.array(Y_train[0:50,:])
• 816
• 817 model=LTS(x_50,y_50,model,ini_Hinode,ini_Reg)

34
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
LTS starts
35
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
LTS starts
36
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
start tune
2
keep moving
3
keep moving
4
weight tune waits
5
keep moving
6
keep moving
7
keep moving

project.py
• 814 x_50=np.array(A11*data1*data1*(0.20,))
• 815 y_50=np.array(Y_train[0:50,:])
• 816
• 817 model=LTS(x_50,y_50,model,ini_Hinode,ini_Reg)

490
keep moving
491
keep moving
492
weight tune waits
493
keep moving
494
keep moving
495
weight tune waits
496
keep moving
497
weight tune waits
498
keep moving
499
keep moving
500
weight tune waits
end tune
weight before cram
```

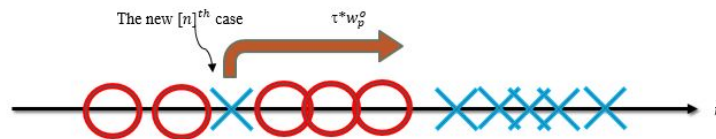
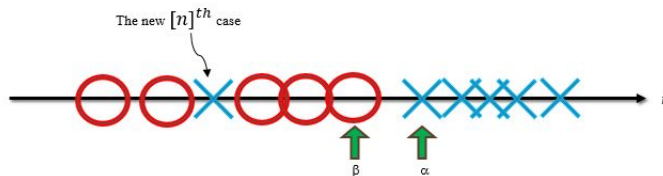


METHODS

Cramming mechanism

using the
LTS

When we encounter with a new case that **cannot be learnt well** with the current SLFN and the weight-tuning mechanism, an extra hidden node is used to **cram** this case.



- Via adding an extra hidden node, we would like to pull this new case into the right place with other cases **staying still at the same place** (for instance, in the above scenario).
- set up the τ value that renders $w_p^o \tau > \max_{u \in \mathbf{I}_2(n)} \sum_{i=1}^{p-1} w_i^o a_i^u > \sum_{i=1}^{p-1} w_i^o a_i^{[n]}$ be true.

A hyper-parameter



METHODS

■ Cramming mechanism

The image displays two side-by-side screenshots of a Sublime Text editor window, showing Python code and terminal output for a neural network training process.

Left Screenshot:

- Code:** Lines 523-535 show print statements for 'before cram', 'LTS sample', 'class0', 'class1', and 'weight before cram'. Line 532 shows a prediction: `#before_cram_pre=model.predict(LTS_X)`.
- Terminal Output:** Shows 'LTS starts' followed by a large array of zeros, then a large array of ones, and 'start tune'.

Right Screenshot:

- Code:** Identical to the left screenshot.
- Terminal Output:** Shows 'weight tune waits' (lines 23-26), 'undesired attractor', 'end tune', 'after cram', followed by a large array of zeros, then a large array of ones, and 'ini soft' and 'soft starts'.

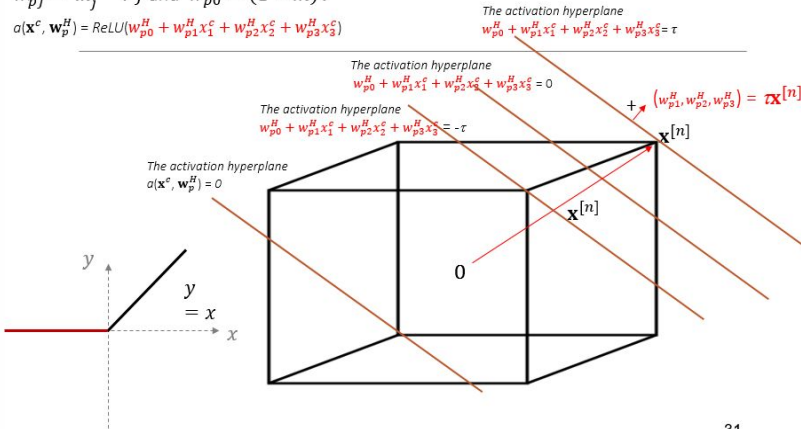


METHODS

The $\{-1,1\}^m$ space

$$w_{pj}^H = \tau x_j^{[n]} \quad \forall j \text{ and } w_{p0}^H = (1-m)\tau$$

$$a(\mathbf{x}^c, \mathbf{w}_p^H) = \text{ReLU}(w_{p0}^H + w_{p1}^H x_1^c + w_{p2}^H x_2^c + w_{p3}^H x_3^c)$$



31

```

C:\Users\chrysal2\2\Desktop\AI_project\project.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

project.py
• 811 r2=np.array(r2).reshape(1,)
• 812 r3=[[0.51014334]]
• 813 r4=[0.13128504]
• 814 r3=np.array(r3).reshape(1,1)
• 815 r4=np.array(r4).reshape(1,1)
816
817 K.set_value(model.layers[0].weights[0], r1)
818 K.set_value(model.layers[0].weights[1], r2)
819 K.set_value(model.layers[0].weights[2], r3)
820 K.set_value(model.layers[0].weights[3], r4)

24 weight tune waits
25 weight tune waits
26 undesired attractor
end tune
[[ [ 0.95872407 -0.0540977 -0.0550883 0.9163613 1.1043496 0.76023334
0.721476 0.6926064 0.6714882 0.6141457 0.42502826 0.57309955
0.5569941 0.5514321 0.44968647 0.45206773 0.460833 0.5152382
0.5129799 0.89717233]]
[[ [ 0.9587851 -0.05336528 -0.05506571 0.915873 1.1038613 0.7603554
0.721415 0.6929726 0.6714882 0.61413044 0.42502826 0.5728554
0.5569941 0.5514321 0.44968647 0.45206773 0.46091694 0.5152382
0.5129494 -0.08929281]]
after cram
[0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[Finished in 15.5s]

[0] 16 9.8231: missing whitespace after '[', [0] 16 9.8231: missing whitespace around operator, ASCII Line 821, Column 11
Tab Size: 4 Python
ENG 9:10 PM
INTL 7/1/2020
Type here to search

```



METHODS

The Cramming Mechanism

Step 1: Let ζ be a given small number. Find an m -vector γ of length one such that $\gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]}) \neq 0$ all $c \in I(n) - \{[n]\}$ AND $(\zeta + \gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})) * (\zeta - \gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})) < 0$ all $c \in I(n) - \{[n]\}$.

Step 2: Let $p+3 \rightarrow p$ and add three new hidden nodes $p-2^{\text{th}}$, $p-1^{\text{th}}$ and p^{th} to the existing SLFN with

- $w_{p-2,0}^H = \zeta - \gamma^T \mathbf{x}^{[n]}$, $w_{p-2}^H = \gamma$,
- $w_{p-1,0}^H = -\gamma^T \mathbf{x}^{[n]}$, $w_{p-1}^H = \gamma$,
- $w_{p0}^H = -\zeta - \gamma^T \mathbf{x}^{[n]}$, $w_p^H = \gamma$, and
- $w_{p-2}^o = -0.5w_{p-1}^o = w_p^o = 1.1(\max_{u \in I_2(n)} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^{[n]}) / \zeta$

Case of $y^{[n]} = 1.0$

$\tau = 1.1$

37

The Cramming Mechanism

Step 1: Let ζ be a given small number. Find an m -vector γ of length one such that $\gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]}) \neq 0$ all $c \in I(n) - \{[n]\}$ AND $(\zeta + \gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})) * (\zeta - \gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})) < 0$ all $c \in I(n) - \{[n]\}$.

Step 2: Let $p+3 \rightarrow p$ and add four new hidden nodes $p-2^{\text{th}}$, $p-1^{\text{th}}$ and p^{th} to the existing SLFN with

- $w_{p-2,0}^H = \zeta - \gamma^T \mathbf{x}^{[n]}$, $w_{p-2}^H = \gamma$,
- $w_{p-1,0}^H = -\gamma^T \mathbf{x}^{[n]}$, $w_{p-1}^H = \gamma$,
- $w_{p0}^H = -\zeta - \gamma^T \mathbf{x}^{[n]}$, $w_p^H = \gamma$, and
- $w_{p-2}^o = -0.5w_{p-1}^o = w_p^o = 1.1(\min_{u \in I_1(n)} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^{[n]}) / \zeta$

Case of $y^{[n]} = -1.0$

$\tau = 1.1$

38



METHODS

Contribution of three extra hidden nodes to the output

- \mathbf{x}^c : whose $\gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})$ is either greater than ζ or less than $-\zeta$
- $\Delta f = w_p^o * [\text{ReLU}(\zeta + \gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})) - 2\text{ReLU}(\gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})) + \text{ReLU}(-\zeta + \gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]}))]$

$\gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]})$	$(p-2)^{\text{th}}$	$-2(p-1)^{\text{th}}$	p^{th}	Δf
-2ζ	0	0	0	0
$-\zeta$	0	0	0	0
-0.7ζ	0.3ζ	0	0	$0.3\zeta w_p^o$
-0.5ζ	0.5ζ	0	0	$0.5\zeta w_p^o$
-0.3ζ	0.7ζ	0	0	$0.7\zeta w_p^o$
0	ζ	0	0	ζw_p^o
0.3ζ	1.3ζ	-0.6ζ	0	$0.7\zeta w_p^o$
0.5ζ	1.5ζ	-1.0ζ	0	$0.5\zeta w_p^o$
0.7ζ	1.7ζ	-1.4ζ	0	$0.3\zeta w_p^o$
ζ	2ζ	-2ζ	0	0
2ζ	3ζ	-4ζ	ζ	0

44

The algorithm for creating an m -vector γ of length one such that $\gamma^T(\mathbf{x}^c - \mathbf{x}^{[n]}) \neq 0$ all $c \in \mathbf{I}(n) - \{[n]\}$

- Assume $x^i \neq x^j$ when $i \neq j$.

Step 1: Set $\beta_1 = 1$ and let $k = 2$.

Step 2: Let $C_k \equiv \{c : c \in \mathbf{I}(n) - \{[n]\} \text{ AND } x_j^c = x_j^{[n]} \text{ all } j = 1, \dots, k\}$. Considering β_k as the unknown and $\beta_j, j = 1, \dots, k-1$, as previously determined, set $\beta_k =$ the smallest integer that is greater than or equal to 1 and $\sum_{j=1}^k \beta_j (x_j^c - x_j^{[n]}) \neq 0$ all $c \in \mathbf{I}(n) - \{[n]\} - C_k$.

Step 3: $k+1 \rightarrow k$. If $k \leq m$, go to Step 2;

Step 4: Set $\gamma_j = \frac{\beta_j}{\sqrt{\sum_{j=1}^m \beta_j^2}} \forall j = 1, \dots, m$ and STOP.



METHODS

```
C:\Users\chrysal21\Desktop\AI_project\project.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

project.py
• 814 A_20=np.zeros((A1.shape[0],1))
• 815 y_50=np.array(Y_train[0:50,:])
816
• 817 model=LTS(x_50,y_50,model,ini_Hinode,ini_Reg)

keep moving
end tune
weight before cram
[array([[0.6551119 ],
        [0.09545052],
        [0.02110529],
        [0.34002292],
        [0.99083555],
        [0.33055994],
        [0.82845074],
        [0.49274734]], dtype=float32), array([0.12430776], dtype=float32)]
weight after cram
[array([[ 0.6551119 ,  0.4984072 ,  0.4984072 ,  0.4984072 ],
        [ 0.09545052, -0.4628568 , -0.4628568 , -0.4628568 ],
        [ 0.02110529,  0.3450561 ,  0.3450561 ,  0.3450561 ],
        [ 0.34002292,  0.1678315 ,  0.1678315 ,  0.1678315 ],
        [ 0.99083555,  0.37587473,  0.37587473,  0.37587473],
        [ 0.33055994,  0.3385552 ,  0.3385552 ,  0.3385552 ],
        [ 0.82845074,  0.28320444,  0.28320444,  0.28320444],
        [ 0.49274734, -0.23241463, -0.23241463, -0.23241463]],
        dtype=float32), array([ 0.12430776, -0.09920378, -0.09930378, -0.09940378], dtype=float32)]
after cram
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

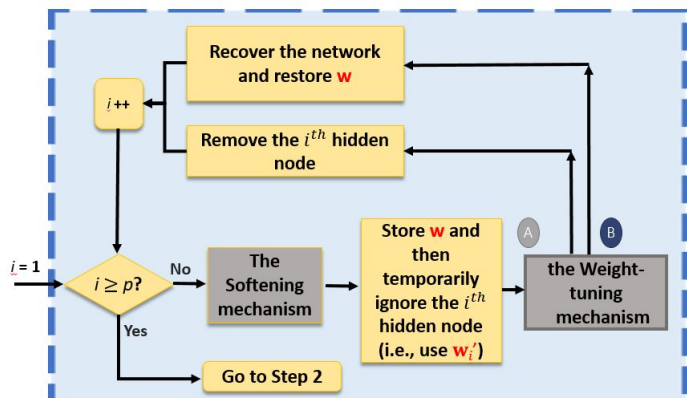
ASCL User ASCL Column 1
Tab Size: 4 Python
Type here to search
ENG 7:53 PM 7/1/2020
```



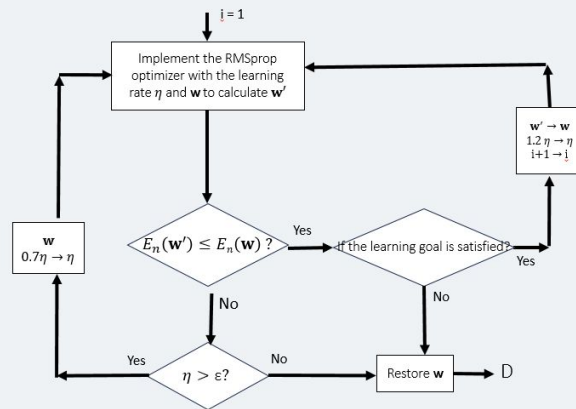

METHODS

■ The softening and integrating mechanism

The Softening and Integrating mechanism (I)
(sequentially check all hidden nodes)



The softening mechanism with the regularization term





File Edit Selection Find View Goto Tools Project Preferences Help

```

soft starts
[array([[ 0.6551119,  0.4984072,  0.4984072,  0.4984072 ],
        [ 0.09545052, -0.4628568, -0.4628568, -0.4628568 ],
        [ 0.02110529,  0.3450561,  0.3450561,  0.3450561 ],
        [ 0.34802292,  0.1678315,  0.1678315,  0.1678315 ],
        [ 0.99083555,  0.37587473,  0.37587473,  0.37587473 ],
        [ 0.33055994,  0.3385552,  0.3385552,  0.3385552 ],
        [ 0.82845074,  0.28320444,  0.28320444,  0.28320444 ],
        [ 0.49274734, -0.23241463, -0.23241463, -0.23241463]],
       dtype=float32), array([ 0.12430776, -0.09920378, -0.09930378, -0.09940378], dtype=float32)]
initial soft not satisfy
soft over
[array([[ 0.6551119,  0.4984072,  0.4984072,  0.4984072 ],
        [ 0.09545052, -0.4628568, -0.4628568, -0.4628568 ],
        [ 0.02110529,  0.3450561,  0.3450561,  0.3450561 ],
        [ 0.34802292,  0.1678315,  0.1678315,  0.1678315 ],
        [ 0.99083555,  0.37587473,  0.37587473,  0.37587473 ],
        [ 0.33055994,  0.3385552,  0.3385552,  0.3385552 ],
        [ 0.82845074,  0.28320444,  0.28320444,  0.28320444 ],
        [ 0.49274734, -0.23241463, -0.23241463, -0.23241463]],
       dtype=float32), array([ 0.12430776, -0.09920378, -0.09930378, -0.09940378], dtype=float32)]
ini soft end
[[0, 1, 2], (0, 1, 3), (0, 2, 3), (1, 2, 3)]
inte start

```

ASCE Use A24 Column 1

Type here to search

File Edit Selection Find View Goto Tools Project Preferences Help

```
[ 0.33855994, 0.3385552, 0.3385552, 0.3385552, 0.3385552, 0.07314805,
 0.07314805, 0.07314805, 0.20312351, 0.20312351, 0.20312351],
[ 0.82845074, 0.28320444, 0.28320444, 0.28320444, -0.2306406,
-0.2306406, -0.2306406, -0.57654506, -0.57654506, -0.57654506],
[ 0.49274734, -0.23241463, -0.23241463, -0.23241463, 0.12022953,
0.12022953, 0.12022953, -0.01939967, -0.01939967, -0.01939967]],
dtype=float32), array([ 0.12430776, -0.09920378, -0.09930378, -0.09940378,
-0.38851705, -0.38861707, 0.04691136, 0.04681136, 0.04671136],
dtype=float32))
ini soft end
[[ (0, 1, 2, 3, 4, 5, 6, 7, 8), (0, 1, 2, 3, 4, 5, 6, 7, 9), (0, 1, 2, 3, 4, 5, 6, 8, 9), (0, 1, 2, 3, 4, 5, 7, 8, 9), (0, 1, 2, 3, 4,
6, 7, 8, 9), (0, 1, 2, 3, 5, 6, 7, 8, 9), (0, 1, 2, 4, 5, 6, 7, 8, 9), (0, 1, 3, 4, 5, 6, 7, 8, 9), (0, 2, 3, 4, 5, 6, 7, 8, 9), (1,
2, 3, 4, 5, 6, 7, 8, 9)]
inte start
2
weight tune waits
C:\Users\chrystal21\Desktop\AI_project\project.py:68: RuntimeWarning: invalid value encountered in greater
if(self.predict[i][0]>0.5):
3
weight tune waits
4
weight tune waits
5
```

ASCE Use Only Column 1

Type here to search

Tab Size 4	Python
------------	--------

ENG 7:47 PM



Cramming mechanism and classification works very well in our training dataset.

```
C:\Users\sthal21\Documents\AI_project\AI_project - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

project.py x ltpy x
695 print('hidden node reduced more')
696 else:
697     if(first==0):
698         finalModel=model
699         tempnode=H1Node
700         print('still no')
701         print('ss start')
702         savesoft1=finalModel.layers[0].get_weights()
703         savesoft2=finalModel.layers[1].get_weights()
704         finalModel=sf.softening(X,Y,finalModel,ID,Regg,tempnode)
705         savesoft1=finalModel.layers[0].get_weights()
706         savesoft2=finalModel.layers[1].get_weights()
707         print('ss end')
708     else:
709         finalModel=model
710         tempnode=H1Node
711
712
713 integ=0
714 Loss_inte=0
715
716 #tf.keras.backend.clear_session()
```



EXPERIMENTAL RESULTS

We first train only on 50 dataset, and use the next 100 data as test dataset. Yet, the prediction rate is not good.

1. It may arise from overfitting due to overfitting from the cramming process. Even though we have softening and integrating mechanism to try to penalize high weight and reduce the amount of hidden nodes as many as possible, the mechanism doesn't kill any nodes in most of the time due to the quality of the data.

```
•523
•524
•525
•526
•527
•528
•529
•530
•531
•532
•533
•534
•535
•536

model.LTS(x_50,y_50,model,Ini_Minode,Ini_Reg)
PLTS(Xtrain nor,y_train,model,Ini_Minode,Ini_Reg)
model.save_weights('my_model_weights.h5')

x_test=np.array(Xtrain nor[51:,:]).reshape(-1,8)
y_test=np.array(Y_train[51:,:]).reshape(-1,1)
p_test=model.predict(x_test).reshape(-1,1)

p_test_trans=np.where(p_test>0.5,1,0)
p_test_trans=p_test_trans.reshape(-1,1)

diff=np.absolute((p_test_trans-v_test)).reshape(-1,1)

weight tune waits 0.46944385, 0.46944385, 0.46944385, -0.6601472, -0.6601472,
22 -0.6601472]], dtype=float32), array([ 0.12430776, 0.65718466, 0.65708464, 0.6569046, -0.28518912,
weight tune waits -0.2852891, -0.28538913, -0.20466235, -0.20476235, -0.20486236,
23 0.10009964, 0.09999964, 0.09989963, 0.21247786, 0.21237788,
weight tune waits 0.21227787, -0.00694886, -0.00704886, -0.00714886, 0.87791234,
24 0.8778123, 0.8777123, 0.15819001, 0.15809001, 0.15799001,
weight tune waits 0.44086856, 0.44076854, 0.44066855, 0.5557202, 0.55562025,
25 0.55552024, 0.3242295, 0.32412952, 0.3240295, 0.07114402,
weight tune waits 0.07104402, 0.07094403, -0.3785015, -0.37860152, -0.3787015,
26 -0.66598946, -0.6660895, -0.6661895, 0.16799164, 0.16789164,
undesired attractor 0.16779163], dtype=float32)]
still no
precision]
0.5656565656565657
[Finished in 582.4s]
ini soft end
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44), (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45), (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
```



REFERENCE

- S. Knerr, L. Personnaz and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network", *Neurocomputing*. Heidelberg : Springer Berlin Heidelberg, 1990.
- Guang-Bin Huang, Yan-Qiu Chen and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," in *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799-801, May 2000.
- Knerr, Stefan, Léon Personnaz, and Gérard Dreyfus. "Single-layer learning revisited: a stepwise procedure for building and training a neural network." *Neurocomputing*. Springer, Berlin, Heidelberg, 1990. 41-50.
- Sun, Zhan-Li, et al. "Sales forecasting using extreme learning machine with applications in fashion retailing." *Decision Support Systems* 46.1 (2008): 411-419.
- <https://food.ndtv.com/health/100-growth-in-diabetes-patients-in-india-in-the-last-15-years-1292926>
- <https://www.kaggle.com/uciml/pima-indians-diabetes-database>
- 國立政治大學 資訊管理學系 蔡瑞煌 特聘教授



THANKS!