The Validation Experiment

國立政治大學 資訊管理學系

蔡瑞煌 特聘教授

The initializing_1_ReLU_LR

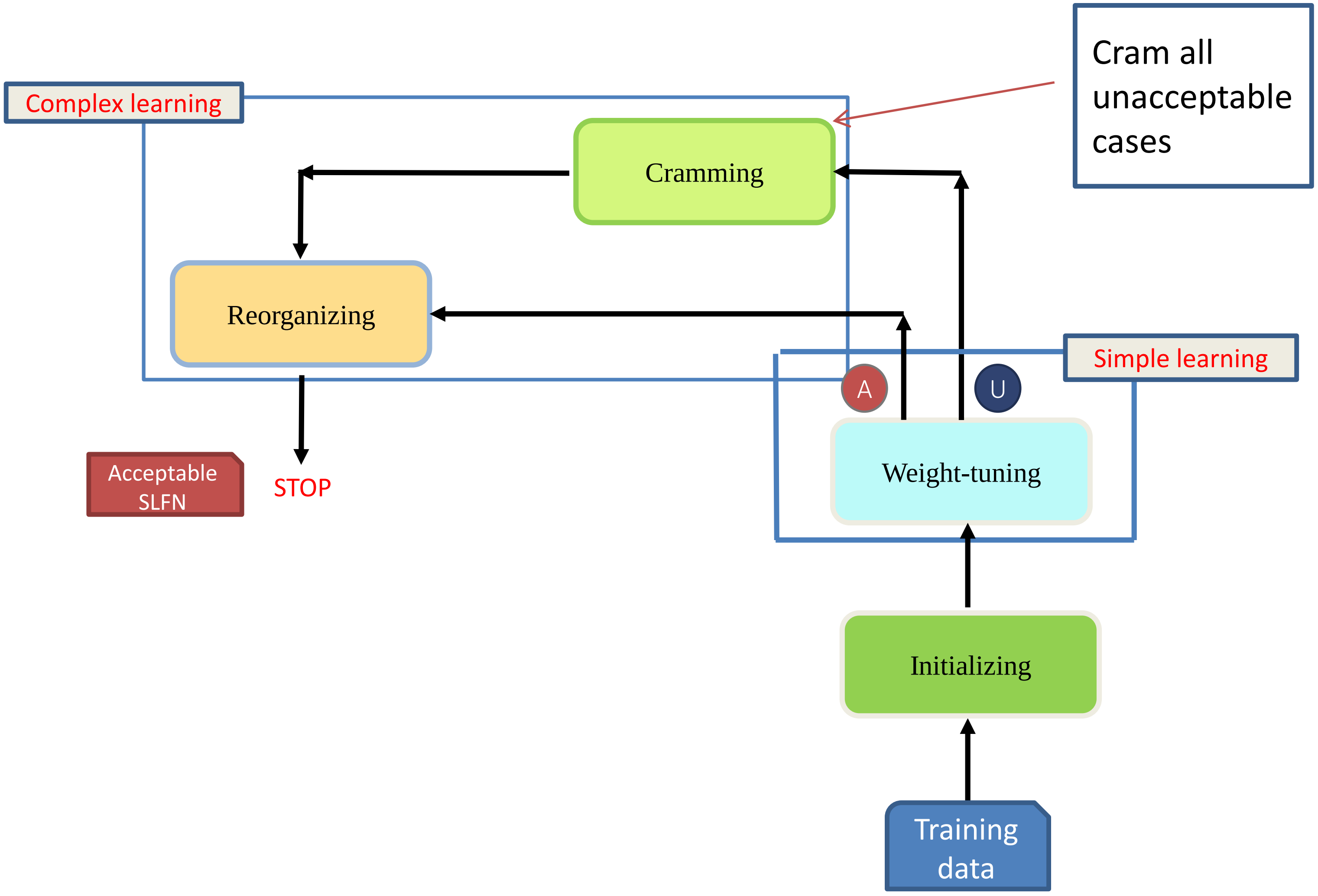
Step 1: Apply the linear regression method to the data set $\{(\mathbf{x}^c, y^c - \min_{u \in \mathbf{I}} y^u): c \in \mathbf{I}\}$ to obtain a set of $m + 1$ weights $\{w_j: j = 0, 1, \ldots, m\}$.

Step 2: Set up the SLFN with one hidden node whose w_{1j}^H equals w_j $\forall$ $j = 1, \ldots, m$, w_{10}^H equals w_0, w_1^o equals 1 and w_0^o equals $\min_{u \in \mathbf{I}} y^u$.

The initializing_*p*_ReLU_WT

The first new learning mechanism

(in flowchart)



The obtaining_LTS

Step 1: Sort all training data $\{(\mathbf{x}^c, y^c)\ c \in \mathbf{I}(N)\}$ by their squared residuals in ascending order as $(e^{[1]})^2 \le (e^{[2]})^2 \le \ldots \le (e^{[N]})^2$.

Step 2: $n = \arg\max_c((e^{[c]})^2 \le \varepsilon^2)$.

The learning goal

Step 3: Obtain the n training data $\{(\mathbf{x}^c, y^c)\}$ that are the ones with the smallest n squared residuals among current N squared residuals.

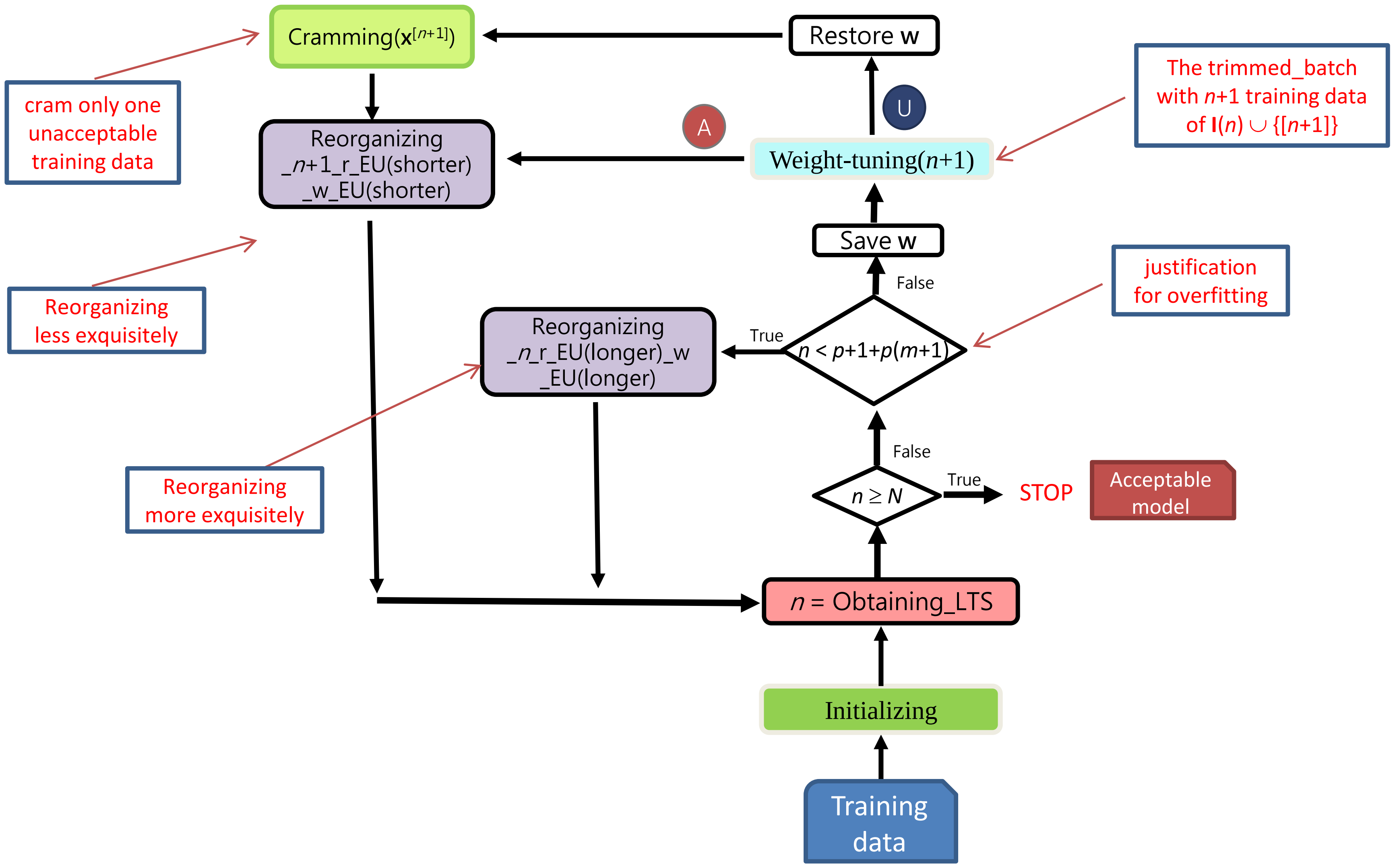
Step 4: Let $\mathbf{I}(n)$ be the set of indices of these data.

$e^c \equiv f(\mathbf{x}^c, \mathbf{w}) - y^c$

Where we are now...

The second new learning mechanism

(in flowchart)



Cramming($\mathbf{x}^{[n+1]}$)
Restore w
cram only one unacceptable training data
Reorganizing _n+1_r_EU(shorter) _w_EU(shorter)
A
U
Weight-tuning(n+1)
The trimmed_batch with n+1 training data of $\mathbf{I}(n) \cup \{[n+1]\}$
Save w
False
Reorganizing less exquisitely
justification for overfitting
Reorganizing _n_r_EU(longer)_w _EU(longer)
True
$n < p+1+p(m+1)$
False
Reorganizing more exquisitely
True
$n \geq N$
STOP
Acceptable model
n = Obtaining_LTS
Initializing
Training data

The selecting_LTS

At the nth stage,

Step 1: Sort all training data $\{(\mathbf{x}^c, y^c)\ c \in \mathbf{I}(N)\}$ by their squared residuals in ascending order as $(e^{[1]})^2 \leq (e^{[2]})^2 \leq \ldots \leq (e^{[N]})^2$.

Step 2: Select the n training data $\{(\mathbf{x}^c, y^c)\}$ that are the ones with the smallest n squared residuals among current N squared residuals.

Step 3: Let $\mathbf{I}(n)$ be the set of indices of these data.

$e^c \equiv f(\mathbf{x}^c, \mathbf{w}) - y^c$

The third new learning mechanism

(in nature language)

- Select the training data one by one according to the LTS principle.

Small errors first, larger errors later

- When a new training data is presented, check first to see whether it is acceptable.
  - ✓ If acceptable, reorganize the network to more concisely integrate all learnt knowledge.
  - ✓ If unacceptable, weight-tune the network to learn it.
    - ➢ If it can be learned, then reorganize the network to more concisely integrate all learnt knowledge.
    - ➢ If it cannot be learned, then cram it. Later, reorganize the network to more concisely integrate all learnt knowledge.

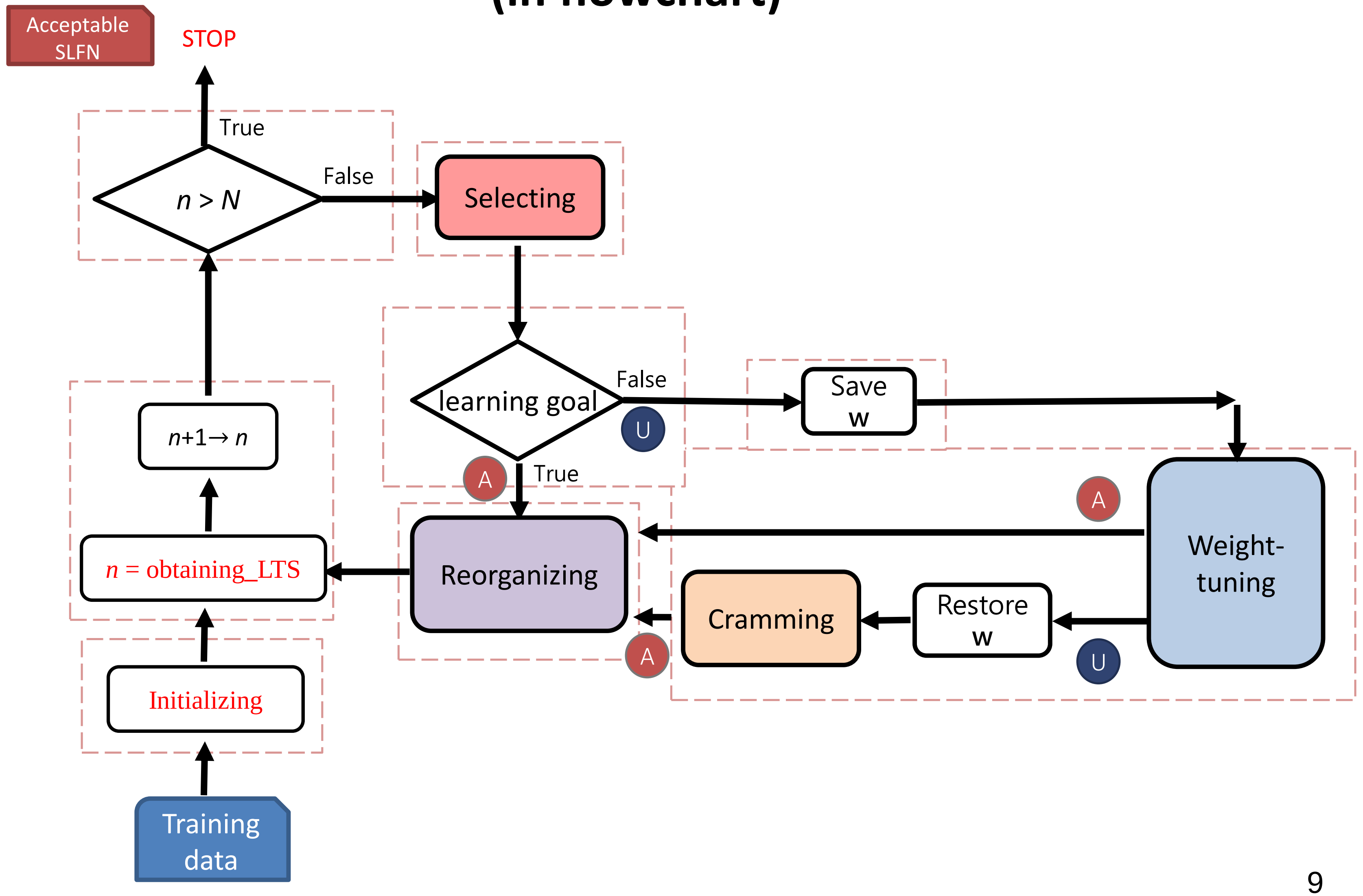
The third new learning mechanism

(in pseudocode)

```
Step 1: Initialize an SLFN with one hidden node.                                  Initializing
Step 2: $n$ = obtaining_LTS and then $n+1 \rightarrow n$.                          Obtaining
Step 3: If $n > N$, STOP.                 The stopping criterion of the learning mechanism
Step 4: Run Selecting_LTS($n$). Let $\mathbf{I}(n)$ be the set of indices of the picked data.   Selecting
Step 5: If the learning goal regarding $\{f(\mathbf{x}^c, \mathbf{w}), \forall c \in \mathbf{I}(n)\}$ is satisfied, go to Step 8;
        otherwise, there is one and only one $\kappa \in \mathbf{I}(n)$ that causes the contradiction
        and $\kappa = [n]$.                                                        Check the learning goal
Step 6: Save $\mathbf{w}$.
Step 7: Weight tune the current SLFN. At the end,                                  Weight-tuning
      (1) if the obtained SLFN is acceptable, go to Step 8;
      (2) otherwise, restore $\mathbf{w}$ and then cram $(\mathbf{x}^{[n]}, y^{[n]})$ to obtain a new   Cramming
          acceptable SLFN.
Step 8: Reorganize the SLFN to regularize the weights and then identify and
        remove the potentially irrelevant hidden node.                             Reorganizing
Step 9: Go to Step 2.                                                              Loop
```

The third new learning mechanism

(in flowchart)



Homework #4

1. Pick up one of the following data files (Copper_forecasting_data.csv (real number inputs) or SPECT_data.txt (binary number inputs)) to decide your AI application problem.
2. Pick up one of the learning mechanisms stated in page 3, page 5 or page 9.
3. Based upon the picked-up, write down the details of your own learning mechanism with ppt.
4. Based upon the details of your own learning mechanism, make the coding of your own learning mechanism with PyTorch or TensorFlow.

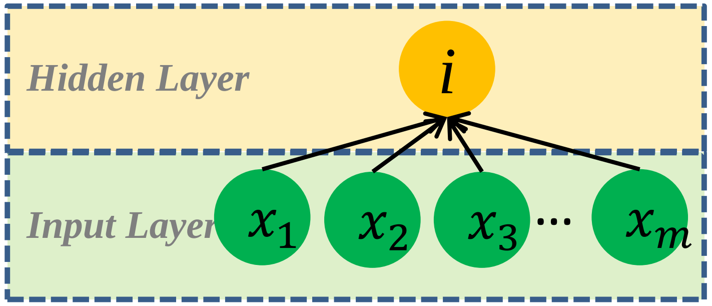
- Note that the learning goals used in all modules (i.e., the initializing module, the obtaining module, the selecting module, the weight-tuning module, the cramming module, the regularizing module, and the reorganizing module) should be consistent.

Present your learning mechanism

1. The SLFN model with notations
2. The learning goal
3. The learning mechanism
4. The detailed arrangement of each module
   - the initializing
   - the obtaining
   - the selecting
   - the weight-tuning
   - the cramming
   - the reorganizing
   - …

Where we are now...

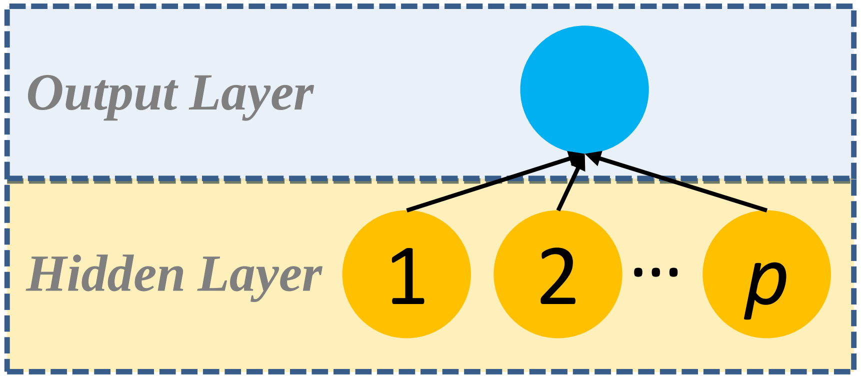
The SLFN with one output node



The hidden layer:

$$a_i^c \equiv ReLU\left(w_{i0}^H + \sum_{j=1}^{m} w_{ij}^H x_j^c\right)$$

$$\mathbf{a} \equiv ReLU(\mathbf{W}^H\mathbf{x} + \mathbf{w}_0^H)$$



The output layer:

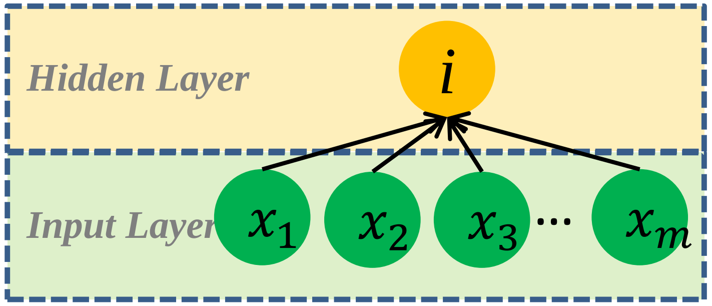
$$f(\mathbf{x}^c,\mathbf{w}) \equiv w_0^o + \sum_{i=1}^{p} w_i^o a_i^c$$

$$\boldsymbol{f}(\mathbf{x}^c, \mathbf{w}) \equiv \mathbf{W}^o\mathbf{a} + \mathbf{w}_0^o$$

$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}(f(\mathbf{x}^c,\mathbf{w}) - y^c)^2$: the loss function;

$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}(f(\mathbf{x}^c,\mathbf{w}) - y^c)^2 + \lambda(\sum_{i=0}^{p}(w_i^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2)$: the loss function with the regularization term.

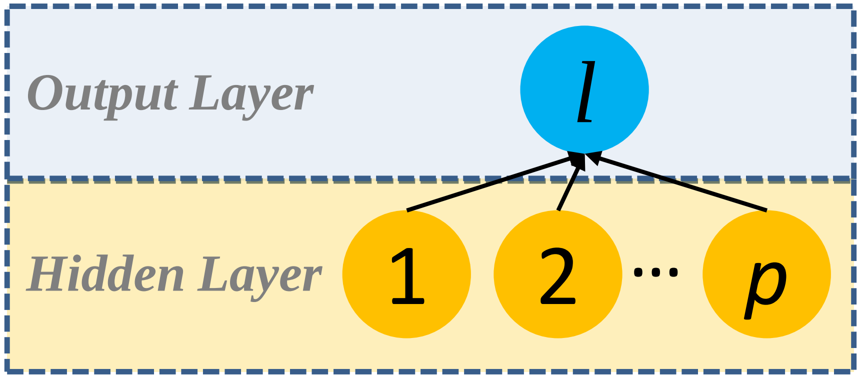
The SLFN with multiple output nodes



The hidden layer:

$$a_i^c \equiv ReLU\left(w_{i0}^H + \sum_{j=1}^{m} w_{ij}^H\, x_j^c\right)$$

$$\mathbf{a} \equiv ReLU(\mathbf{W}^H\mathbf{x} + \mathbf{w}_0^H)$$



The output layer:

$$f_l(\mathbf{x}^c,\mathbf{w}) \equiv w_{l0}^o + \sum_{i=1}^{p} w_{li}^o a_i^c$$

$$\boldsymbol{f}(\mathbf{x}^c, \mathbf{w}) \equiv \mathbf{W}^o\mathbf{a} + \mathbf{w}_0^o$$

$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}\sum_{l=1}^{q}(f_l(\mathbf{x}^c,\mathbf{w}) - y_l^c)^2$: the loss function;

$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}\sum_{l=1}^{q}(f_l(\mathbf{x}^c,\mathbf{w}) - y_l^c)^2 + \lambda(\sum_{l=1}^{q}\sum_{i=0}^{p}(w_{li}^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2)$: the loss function with the regularization term.

Where we are now...

Two-layer nets / SLFN with one output node

The notations and indexes

- $ReLU(x) \equiv max(0, x)$;
- N: the number of data;
- m: the number of input nodes;
- $\mathbf{x}^c \equiv (x_1^c, x_2^c, \dots, x_m^c)^{\mathrm{T}}$: the c^{th} input;
- p: the number of adopted hidden nodes; p is adaptable within the training phase;
- $w_{i,0}^{\mathrm{H}}$: the bias value of i^{th} hidden node;
- $w_{i,j}^{\mathrm{H}}$: the weight between the j^{th} input node and the i^{th} hidden node, $j = 1, 2, \dots, m$;
- $\mathbf{w}_i^{\mathrm{H}} \equiv (w_{i,1}^{\mathrm{H}}, w_{i,2}^{\mathrm{H}}, \dots, w_{i,m}^{\mathrm{H}})^{\mathrm{T}}, i=1, 2, \dots, p$;
- $\mathbf{w}^H \equiv (\mathbf{w}_1^H, \mathbf{w}_2^H, \dots, \mathbf{w}_p^H)^{\mathrm{T}}$;
- $\mathbf{w}_0^H \equiv (w_{1,0}^H, w_{2,0}^H, \dots, w_{p,0}^H)^{\mathrm{T}}$;
- w_0^o: the bias value of output node;
- w_i^o: the weight between the i^{th} hidden node and the output node;
- $\mathbf{w}^o \equiv \left(w_1^o, w_2^o, \dots, w_p^o\right)^{\mathrm{T}}$;
- $\mathbf{w} \equiv \{\mathbf{w}^H, \mathbf{w}_0^H, \mathbf{w}^o, w_0^o\}$;
- a_i^c: the activation value of i^{th} hidden node corresponding to $\mathbf{x}^c$;
- $\mathbf{a}^c \equiv \left(a_1^c, a_2^c, \dots, a_p^c\right)^{\mathrm{T}}$;
- $f(\mathbf{x}^c, \mathbf{w}) \in \mathrm{R}$: the output value of SLFN corresponding to $\mathbf{x}^c$;
- y^c: the desired output value corresponding to $\mathbf{x}^c$;
- $e^c \equiv f(\mathbf{x}^c, \mathbf{w}) - y^c$.

Should specify (binary or real numbers)

The adaptive SLFN, if you adopt the new learning mechanism

Depend on the application!

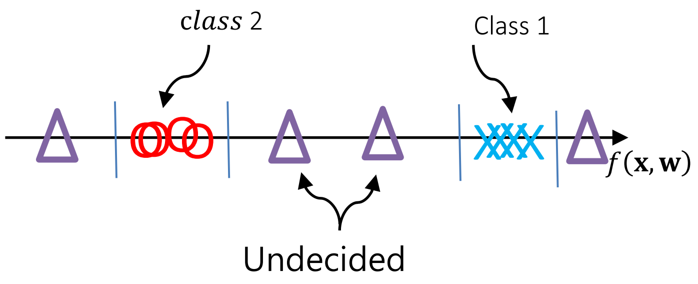
Different stopping criteria result in different length of training time and different model.

Should specify (binary or real numbers)

Where we are now...

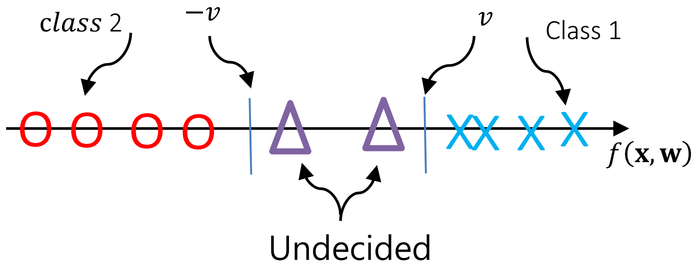
The learning goals for the SLFN with each output node whose output values are real numbers for the two-class classification application

$y^c = 1\ \forall\ c \in \mathbf{I}_1;\ y^c = 0\ \forall\ c \in \mathbf{I}_2$ X : $f(\mathbf{x}^c, \mathbf{w}),\ \forall\ c \in \mathbf{I}_1$ O : $f(\mathbf{x}^c, \mathbf{w}),\ \forall\ c \in \mathbf{I}_2$



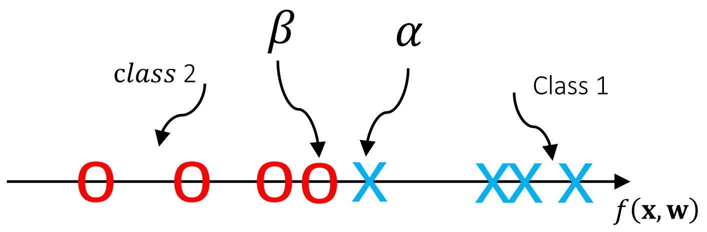
learning goal type 1
(also inferencing mechanism):
$|f(\mathbf{x}^c, \mathbf{w}) - y^c| \le \varepsilon\ \forall\ c \in \mathbf{I}_1;$
$|f(\mathbf{x}^c, \mathbf{w}) + y^c| \le \varepsilon\ \forall\ c \in \mathbf{I}_2$

ε Is a hyperparameter regarding the learning!



learning goal type 2
(also inferencing mechanism):
$f(\mathbf{x}^c, \mathbf{w}) \ge \nu\ \forall\ c \in \mathbf{I}_1;$
$f(\mathbf{x}^c, \mathbf{w}) \le -\nu\ \forall\ c \in \mathbf{I}_2$

ν Is a hyperparameter regarding the learning and the inferencing!

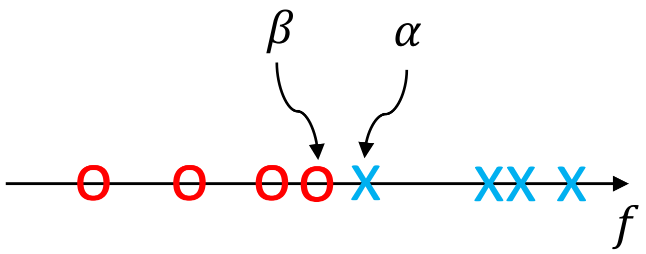


learning goal type 3: LSC
inferencing mechanism:
$f(\mathbf{x}^c, \mathbf{w}) \ge \nu\ \forall\ c \in \mathbf{I}_1;$
$f(\mathbf{x}^c, \mathbf{w}) \le -\nu\ \forall\ c \in \mathbf{I}_2$

ν Is a hyperparameter regarding the learning!

Where we are now…

The learning goals for the SLFN with each output node whose output values are real numbers for the two-class classification application



$$\alpha \equiv \min_{c\epsilon\mathbf{I}_1} f(\mathbf{x}^c, \mathbf{w})\,;\ \beta \equiv \max_{c\epsilon\mathbf{I}_2} f(\mathbf{x}^c, \mathbf{w})$$

learning goal type 3: LSC

When LSC $(\alpha > \beta)$ is true, the inferencing mechanism

$$f(\mathbf{x}^c, \mathbf{w}) \geq v\ \ \forall\, c \in \mathbf{I}_1 \text{ and } f(\mathbf{x}^c, \mathbf{w}) \leq -v\ \forall\, c \in \mathbf{I}_2$$

can be set by directly adjusting $\mathbf{w}^o$ according to the following formula:

$$\frac{2v}{\alpha-\beta} w_i^o \rightarrow w_i^o\ \ \forall\, i,$$

The weight vector between the hidden layer and the output node

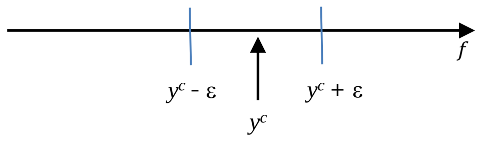
then $v - \min_{c\in\mathbf{I}_1} \sum_{i=1}^{p} w_i^o a_i^c \rightarrow w_0^o$

The threshold of the output node

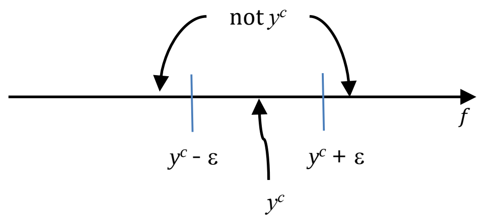
The regression applications

The learning goal

$$|f(\mathbf{x}^c, \mathbf{w}) - y^c| \leq \varepsilon \; \forall \; c \in \mathbf{I}$$



The inferencing mechanism

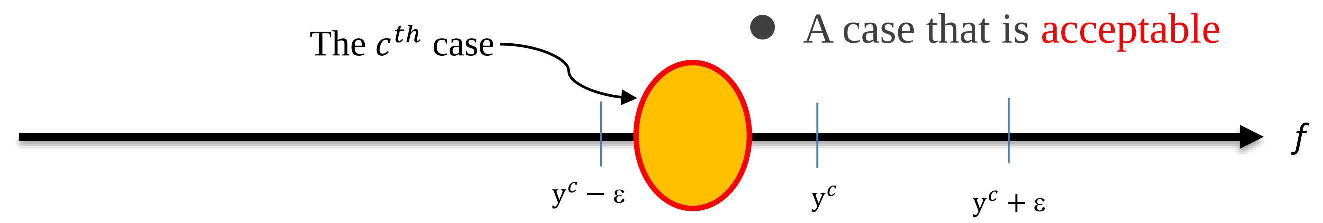


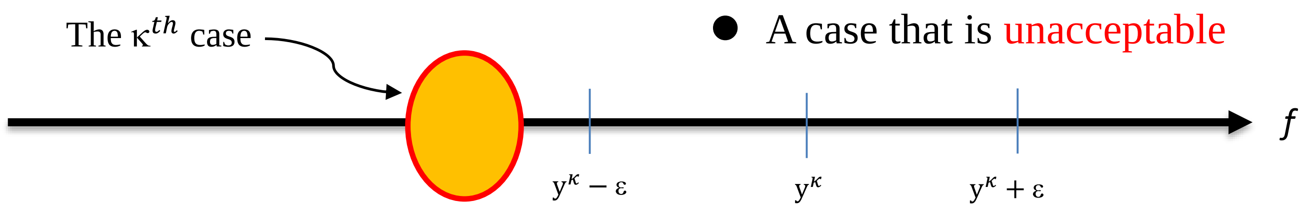
This learning goal and the associated inferencing mechanism is similar to LGT1:
$|f(\mathbf{x}^c, \mathbf{w}) - 1| \leq \varepsilon \; \forall \; c \in \mathbf{I}_1$ and $|f(\mathbf{x}^c, \mathbf{w})| \leq \varepsilon \; \forall \; c \in \mathbf{I}_2$

The unacceptable case

The acceptability of each case is related with the learning goal.

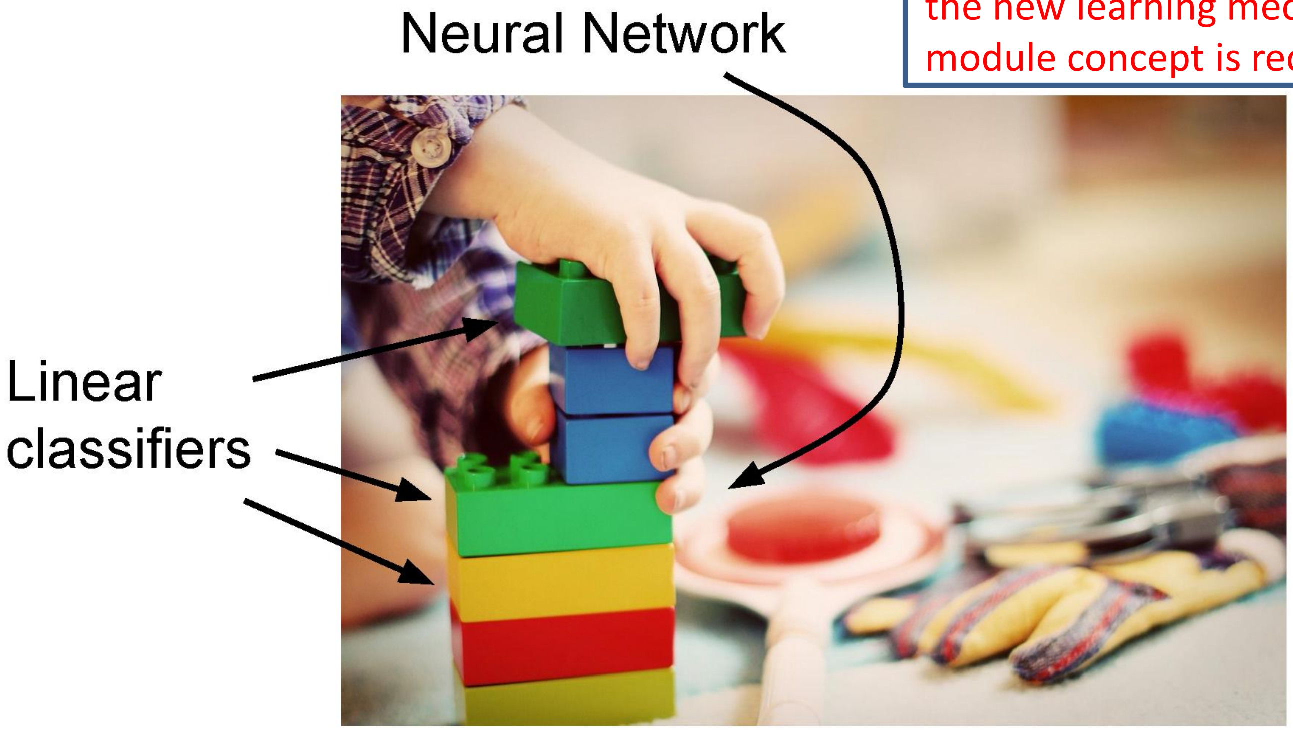
For example, the learning goal is $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2 \ \forall \ c \in \mathbf{I}$





Developing a new learning algorithm is like playing with Lego – lots of (pre-built or self-built) modules

For the AI application, some AI framework (e.g., PyTorch or TensorFlow) is used to implement the new learning mechanism. Thus, the module concept is recommended.



This image is CC0 1.0 public domain

The module list

- ✓ Weight-tuning
- ✓ Regularizing
- ✓ Reorganizing

Optimization mechanisms: much harder to be proved by mathematical proofs, but much easier to be approved by CS code.

- ✓ Cramming
- ✓ Initializing
- ✓ Obtaining
- ✓ Selecting
- ✓ …

Rule-based mechanisms: much easier to be proved by mathematical proofs, but the code approval is still required.

The weight-tuning module

- The weight-tuning module helps tune up the weights to decrease the loss function value to obtain an acceptable SLFN.

✓ the weight-tuning_EU

The simplest and the learning time length is expected

✓ the weight-tuning_EU_LG

Shorter learning time length than the weight-tuning_EU

✓ the weight-tuning_EU_LG_UA

The learning time length may be longer than the weight-tuning_EU_LG

✓ the weight-tuning_LG_UA

The learning time length is not an issue

✓ Your creative idea

The regularizing module

- After obtaining an acceptable SLFN, the regularizing module helps further regularize weights of the acceptable SLFN while keeping the learning goal satisfied.
- A well-regularized SLFN has a less-overfitting tendency.

✓the regularizing_LG_UA

The regularizing time length may be much longer

✓the regularizing_EU_LG_UA

The regularizing time length is expected

✓the regularizing_EU

The simplest and the regularizing time length is expected

✓the regularizing_DO

✓the regularizing_BN

✓Your creative idea

The reorganizing module

The reorganizing module helps regularize weights of an acceptable SLFN and then identify and prune some potentially irrelevant hidden nodes.

- ✓ The reorganizing_ALL_r_EU_LG_UA_w_EU_LG_UA that regularizes weights of an acceptable SLFN while keeping the learning goal satisfied as well as examines all hidden nodes one by one to see any of them is potentially irrelevant. Remove potentially irrelevant hidden nodes identified within the process.
- ✓ The reorganizing_R3_r_EU_LG_UA_w_EU_LG_UA that regularizes weights of an acceptable SLFN while keeping the learning goal satisfied as well as randomly picks up 3 hidden nodes and examines whether they are potentially irrelevant. Remove potentially irrelevant hidden nodes identified within the process.
- ✓ The reorganizing_PCA_r_EU_LG_UA_w_EU_LG_UA that regularizes weights of an acceptable SLFN while keeping the learning goal satisfied as well as uses PCA to pick up a hidden node and examines whether it is potentially irrelevant. If yes, remove it and then repeat the process; otherwise, stop the process.
- ✓ The reorganizing_MAW_r_EU_LG_UA_w_EU_LG_UA that regularizes weights of an acceptable SLFN while keeping the learning goal satisfied as well as uses $k = \arg\min_i |w_i^o|$ to pick up a hidden node and examines whether it is potentially irrelevant. If yes, remove it and then repeat the process; otherwise, stop the process.
- ✓ The reorganizing_ETP_r_EU_LG_UA_w_EU_LG_UA that regularizes weights of an acceptable SLFN while keeping the learning goal satisfied as well as calculates the entropy of each hidden node and then, based on the obtained entropy, picks up a hidden node and examines whether it is potentially irrelevant. If yes, remove it and then repeat the process; otherwise, stop the process.
- ✓ Your creative idea

Where we are now...

A rule-based mechanism

The cramming module for SLFN with single output node

The cramming module helps add extra hidden nodes with proper weights to the existing SLFN to make the learning goal satisfied immediately.

- ✓The cramming_ReLU_BI_SO_LGT1_SU; The cramming_ReLU_RI_SO_LGT1_SU
- ✓The cramming_ReLU_BI_SO_LGT3_SU; The cramming_ReLU_RI_SO_LGT3_SU
- ✓The cramming_ReLU_BI_SO_RE_SU; The cramming_ReLU_RI_SO_RE_SU
- ✓The cramming_ReLU_BI_SO_RE_MU; The cramming_ReLU_RI_SO_RE_MU
- ✓The cramming_ReLU_BI_SO_LGT1_MU; The cramming_ReLU_RI_SO_LGT1_MU
- ✓The cramming_ReLU_BI_SO_LGT3_MU; The cramming_ReLU_RI_SO_LGT3_MU
- ✓Your creative idea

You may derive the red parts by yourself.

Where we are now…

A rule-based mechanism

The cramming module for SLFN with multiple output nodes

The cramming module helps add extra hidden nodes with proper weights to the existing SLFN to make the learning goal satisfied immediately.

- ✓The cramming_ReLU_BI_MO_RE_SU; The cramming_ReLU_RI_MO_RE_SU
- ✓The cramming_ReLU_BI_MO_LGT1_SU; The cramming_ReLU_RI_MO_LGT1_SU
- ✓The cramming_ReLU_BI_MO_LGT3_SU; The cramming_ReLU_RI_MO_LGT3_SU
- ✓The cramming_ReLU_BI_MO_RE_MU; The cramming_ReLU_RI_MO_RE_MU
- ✓The cramming_ReLU_BI_MO_LGT1_MU; The cramming_ReLU_RI_MO_LGT1_MU
- ✓The cramming_ReLU_BI_MO_LGT3_MU; The cramming_ReLU_RI_MO_LGT3_MU
- ✓Your creative idea

You may derive the red parts by yourself.

The initializing module

The initializing module helps initialize an SLFN with few hidden nodes and proper weights to make several (at least two) training data acceptable.

✓The initializing_1_ReLU_LR

✓The initializing_*p*_ReLU_WT   ($p > 1$)

✓Your creative idea

Validate the new mechanism

- Cannot validate the new learning mechanism through the **mathematical proof**.
- To validate the new learning mechanism, you need to make it and then to set up an AI application experiment with the real-world data, the proposed learning mechanism, and the computation capability.
- Check whether the corresponding learning process does display the proposed ideas/concepts. This is an AI fundamental study issue regarding the learning mechanism.
- Check whether the proposed learning mechanism does lead to good performances in the AI application. This is an AI application study issue regarding the AI system.

Present your AI application

1. 描述核心問題：AI 應用研究要處理之問題是什麼？
2. 描述自變數**x**和因變數*y*。
3. 描述實際使用的資料。
4. 描述解決核心問題所採用的新型學習演算法。
5. 描述新型學習演算法的電腦實作環境。
6. 描述實驗目的：如何驗證此新型學習演算法的優點。
7. 描述新型學習演算法之對照組。
8. 描述實驗結果。
9. 結論與討論

Objectives of the experiment

The experiment results should provide evidences for examining whether

(1) the corresponding learning process does display the proposed ideas/concepts as well as the LTS module, the cramming module, and the reorganizing module can help cope with the encountered undesired attractors and alleviate the overfitting tendency.

(2) the proposed learning algorithm does lead to good application performance (in terms of effectiveness and efficiency) – the proposed learning algorithm can have better accuracy than other tools in the literature and the total amount of training time is acceptable.

Objectives of the experiment

- From the learning mechanism perspective, a good experiment design can lead to better learning insights for the AI professionals.
- From the AI application perspective, a good experiment design can lead to better insights for the domain professionals.

The AI Application Problem

- Single Proton Emission Computed Tomography (SPECT) heart diagnosis data set (Kurgan, et al., 2001; UCI Machine Learning)
- 267 instances (55 normal samples and 212 abnormal samples)
- 23 attributes
  (x: 22 (binary) attributes,
  y: 1 (binary) attribute)
- y: Binary (normal: 0, abnormal: 1)
- Randomly generate 20 sets of training and testing data.

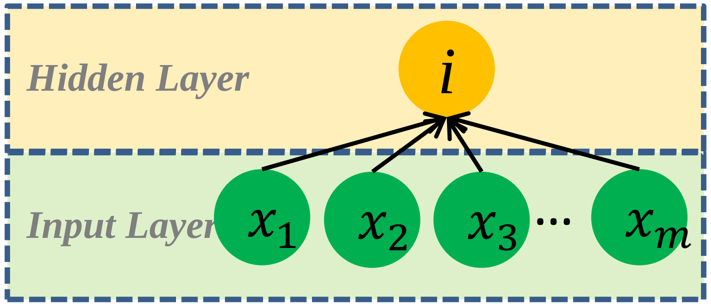
Training data set	Amount of Normal samples	40
	Amount of abnormal samples	40
Testing data set	Amount of normalsamples	15
	Amount of abnormal samples	172
Total amount of samples		267

The AI Application Problem (2nd version)

- Single Proton Emission Computed Tomography (SPECT) heart diagnosis data set (Kurgan, et al., 2001; UCI Machine Learning)
- 267 instances (55 normal samples and 212 abnormal samples)
- 23 attributes
  (x: 22 (binary) attributes,
  y: 1 (binary) attribute)
- y: Binary (normal: 0, abnormal: 1)
- The SPECT dataset has a total of 267 instances, and the ratio of normal to abnormal instances is approximately 1:4.
- After cleaning the data, there are a total of 250 instances, with 52 normal instances and 198 abnormal instances.
- Randomly generate 20 sets of training and testing data.

Training data set	Amount of normal samples	21
	Amount of abnormal samples	59
Testing data set	Amount of normal samples	31
	Amount of abnormal samples	139
Total amount of samples		250

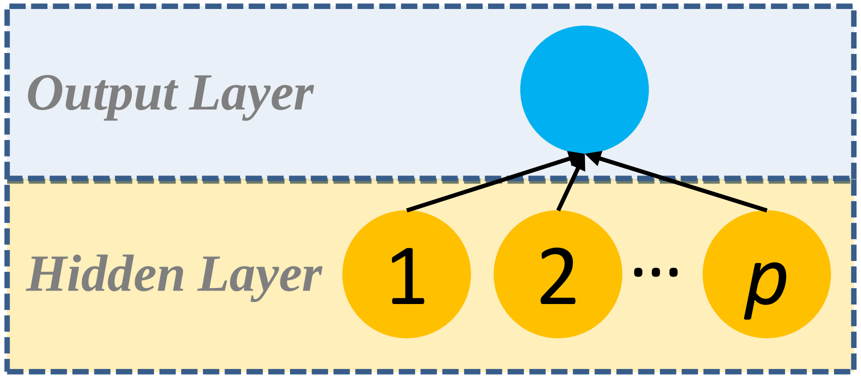
The SLFN with one output node



The hidden layer:

$$a_i^c \equiv ReLU\left(w_{i0}^H + \sum_{j=1}^{m} w_{ij}^H x_j^c\right)$$

$$\mathbf{a} \equiv ReLU(\mathbf{W}^H\mathbf{x} + \mathbf{w}_0^H)$$



The output layer:

$$f(\mathbf{x}^c,\mathbf{w}) \equiv w_0^o + \sum_{i=1}^{p} w_i^o a_i^c$$

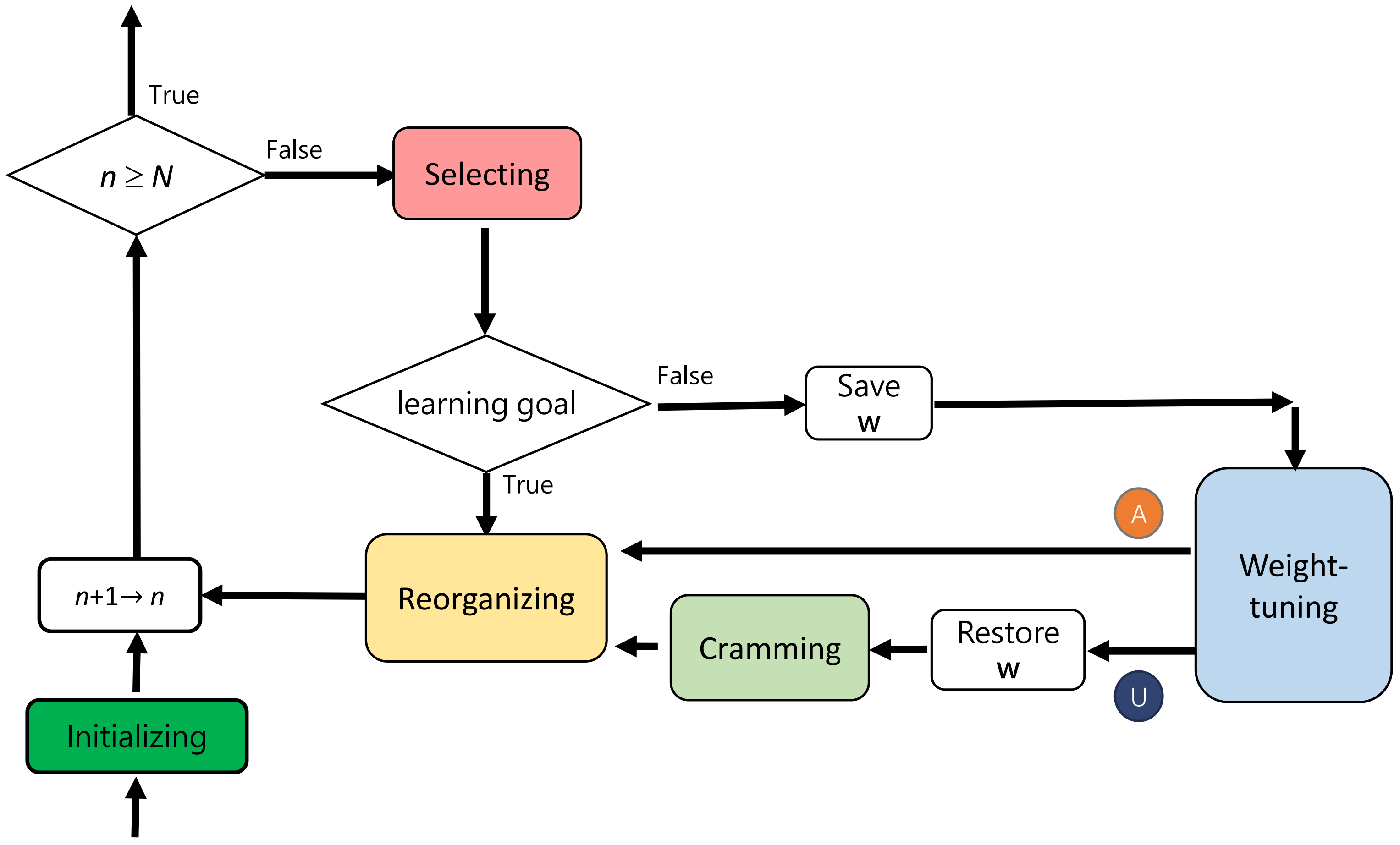
$$\boldsymbol{f}(\mathbf{x}^c, \mathbf{w}) \equiv \mathbf{W}^o\mathbf{a} + \mathbf{w}_0^o$$

$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}(f(\mathbf{x}^c,\mathbf{w}) - y^c)^2$: the loss function;

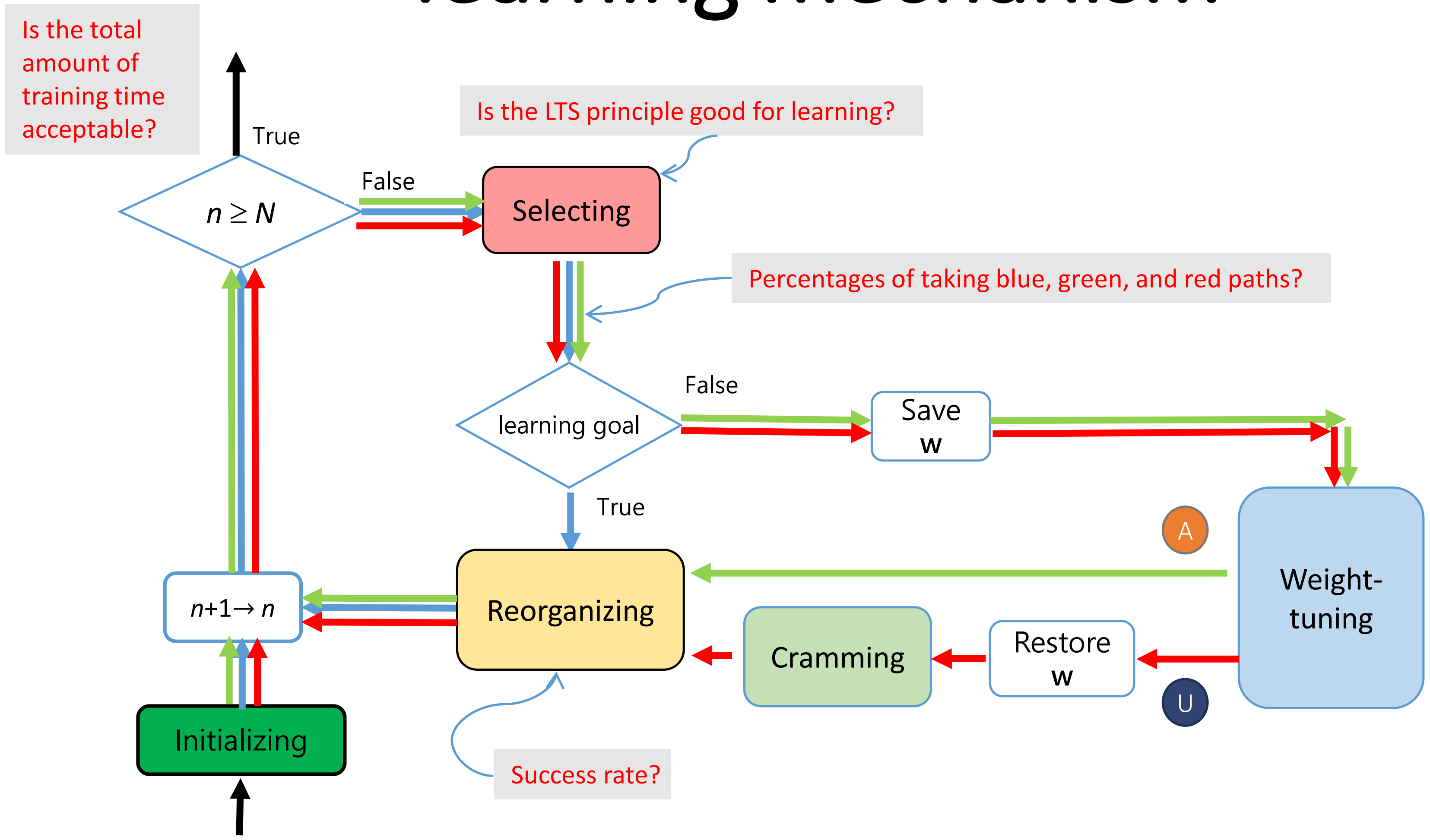
$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}(f(\mathbf{x}^c,\mathbf{w}) - y^c)^2 + \lambda(\sum_{i=0}^{p}(w_i^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2)$: the loss function with the regularization term.

The proposed learning mechanism

(in flowchart)



Validate the process flow of the new learning mechanism


Is the total amount of training time acceptable?
True
Is the LTS principle good for learning?
False
$n \geq N$
Selecting
Percentages of taking blue, green, and red paths?
False
learning goal
Save w
True
A
Weight-tuning
Reorganizing
n+1→ n
Cramming
Restore w
U
Initializing
Success rate?

Four versions

For the validation purpose, there are four versions of the proposed learning mechanism (i.e., four different module arrangements).

Version	The selection module	The reorganizing module
CSI-100	PO_n^N	Reorganizing(100)
CSI-LTS-0	LTS_n^N	Reorganizing(0)
CSI-LTS-100	LTS_n^N	Reorganizing(100)
CSI-LTS-500	LTS_n^N	Reorganizing(500)

LTS_n^N: the module that follows the least trimmed squares principle to pick up *n* training data from *N* training data.

PO_n^N: the module that follows the pre-order principle to pick up first *n* training data from *N* training data.

Reorganizing(100): the module that helps further regularize weights one hundred epochs as well as identify and remove the potentially irrelevant hidden node.

Reorganizing(500): the module that helps further regularize weights five hundred epochs as well as identify and remove the potentially irrelevant hidden node.

Reorganizing(0): the module that helps merely identify and remove the potentially irrelevant hidden node.

Where we are now...

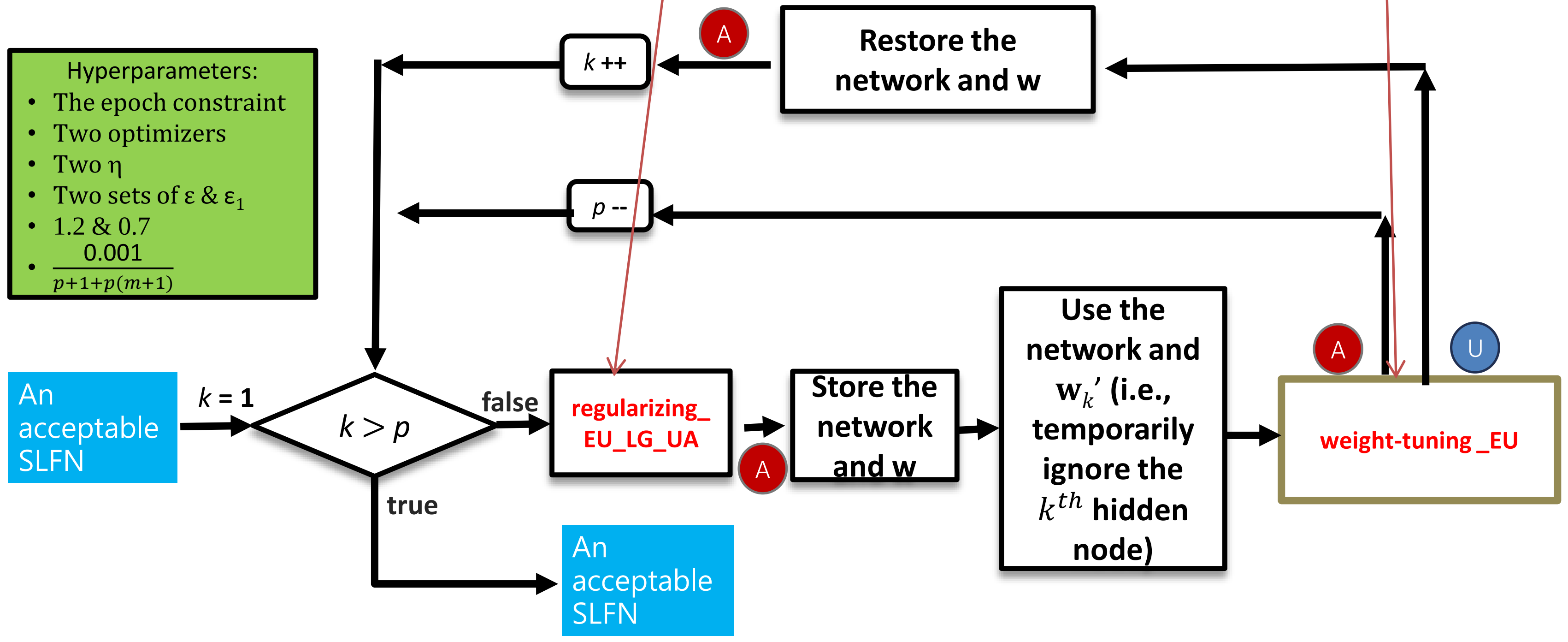
An optimization mechanism

The reorganizing_ALL_r_EU_LG_UA_w_EU_LG_UA

$$L_N(\mathbf{w}) \equiv \frac{\sum_c (f(\mathbf{x}^c,\mathbf{w}) - y^c)^2}{N} + \frac{0.001}{p+1+p(m+1)}\left(\sum_{i=0}^{p}(w_i^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2\right)$$

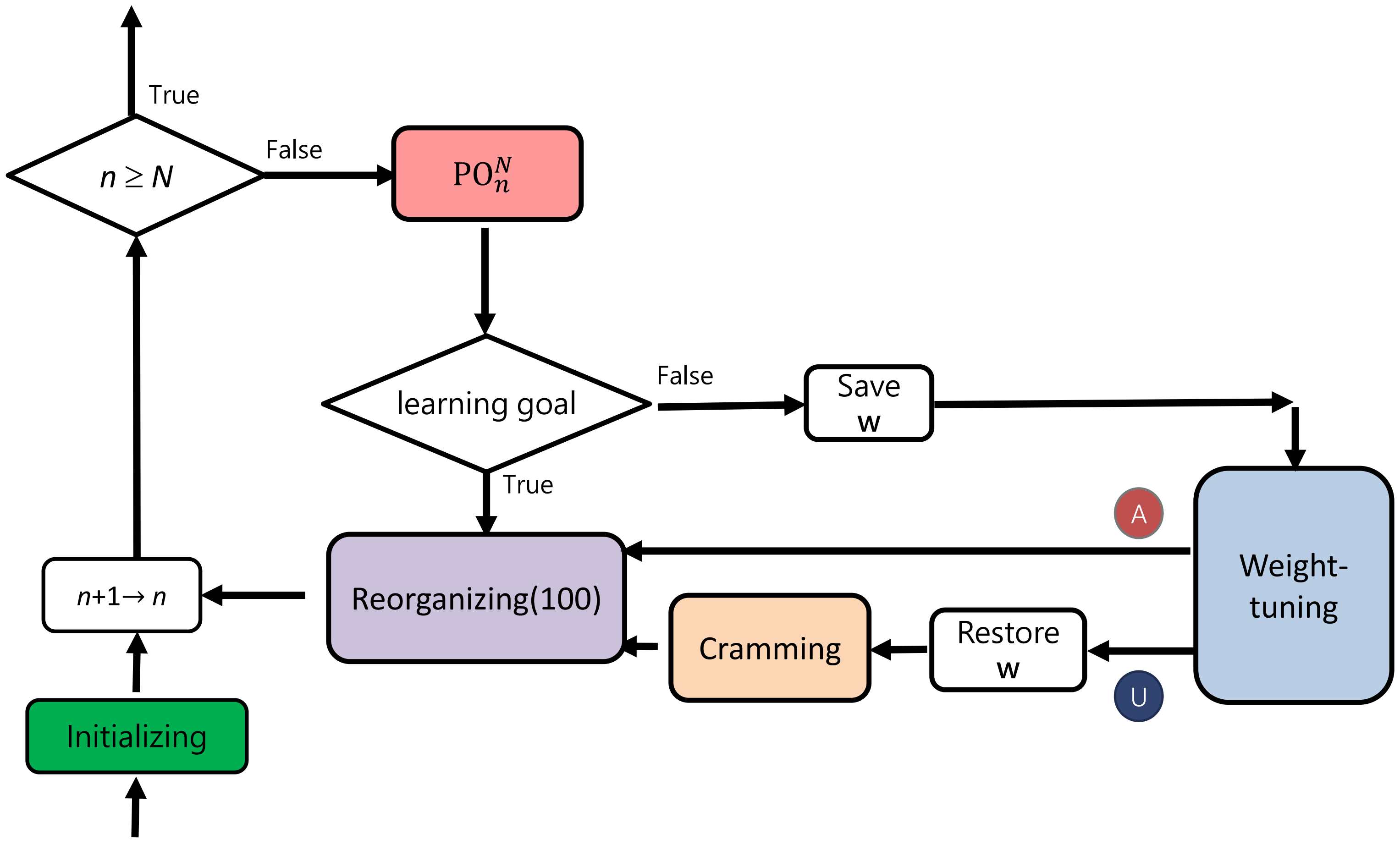
$$L_N(\mathbf{w}) \equiv \frac{\sum_c (f(\mathbf{x}^c,\mathbf{w}) - y^c)^2}{N}$$

Note that there are two optimizers: One for the regularizing purpose Another for the pruning purpose

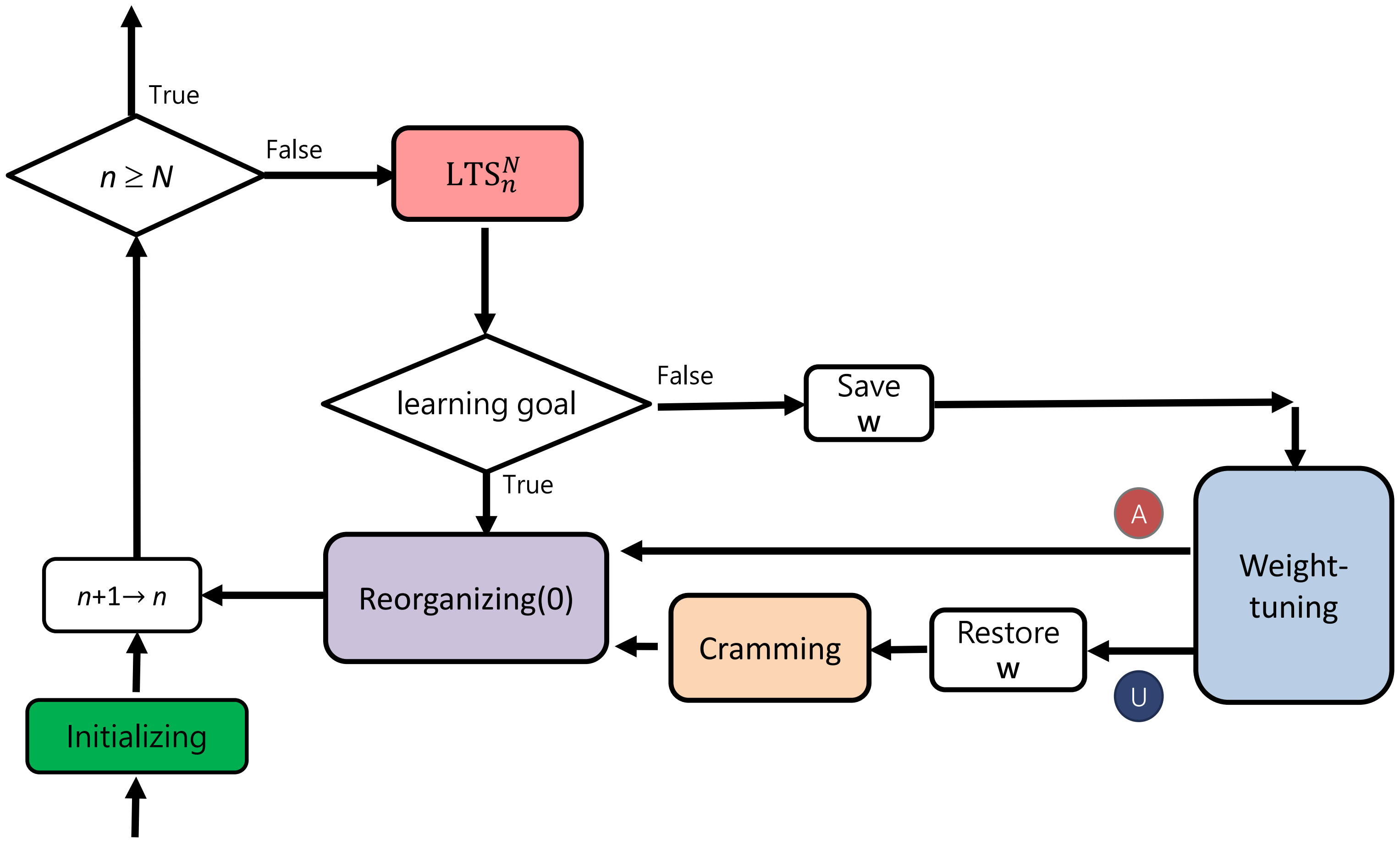

Hyperparameters:

- The epoch constraint
- Two optimizers
- Two η
- Two sets of ε & ε_1
- 1.2 & 0.7
- $\frac{0.001}{p+1+p(m+1)}$

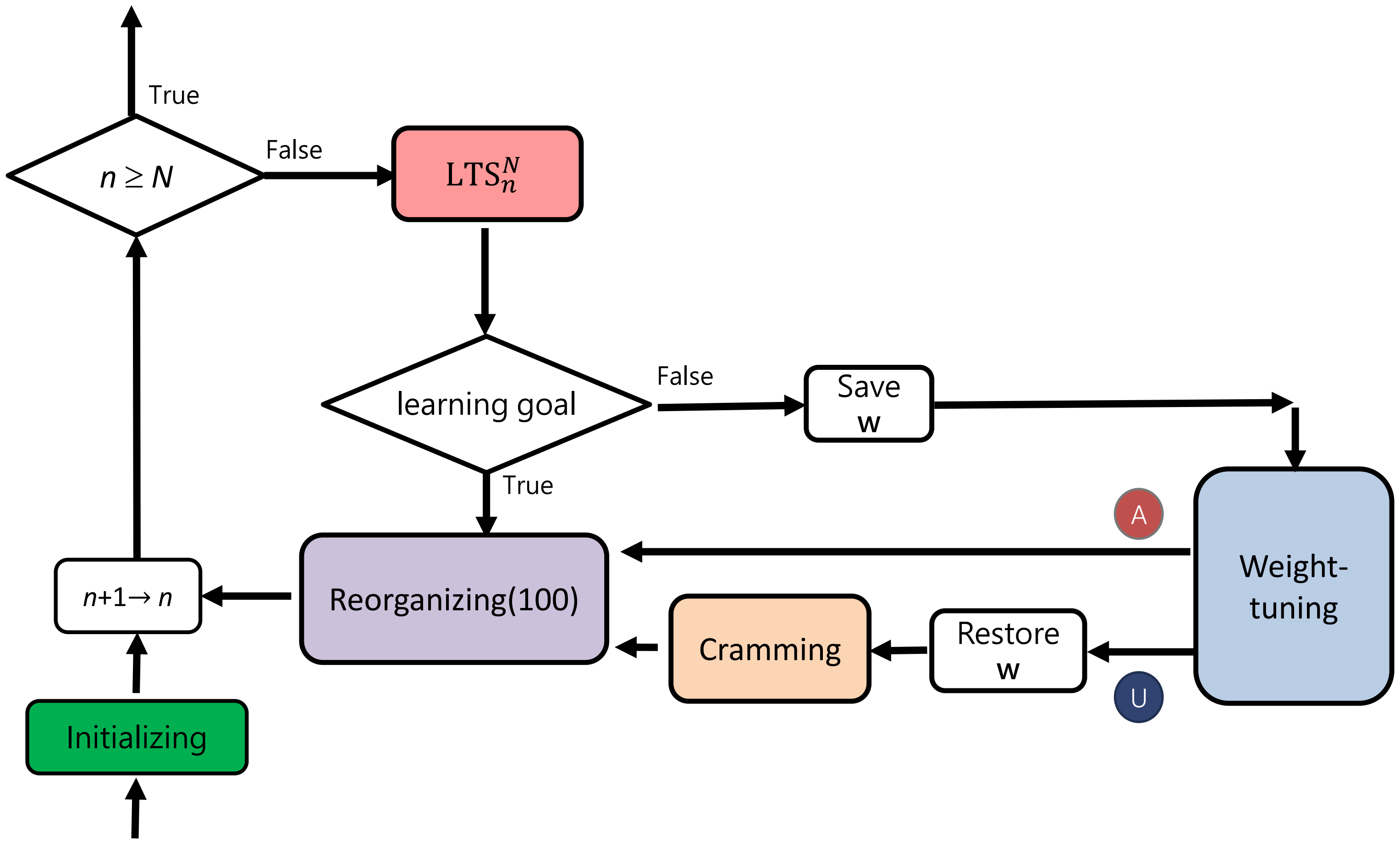
The proposed CSI-100



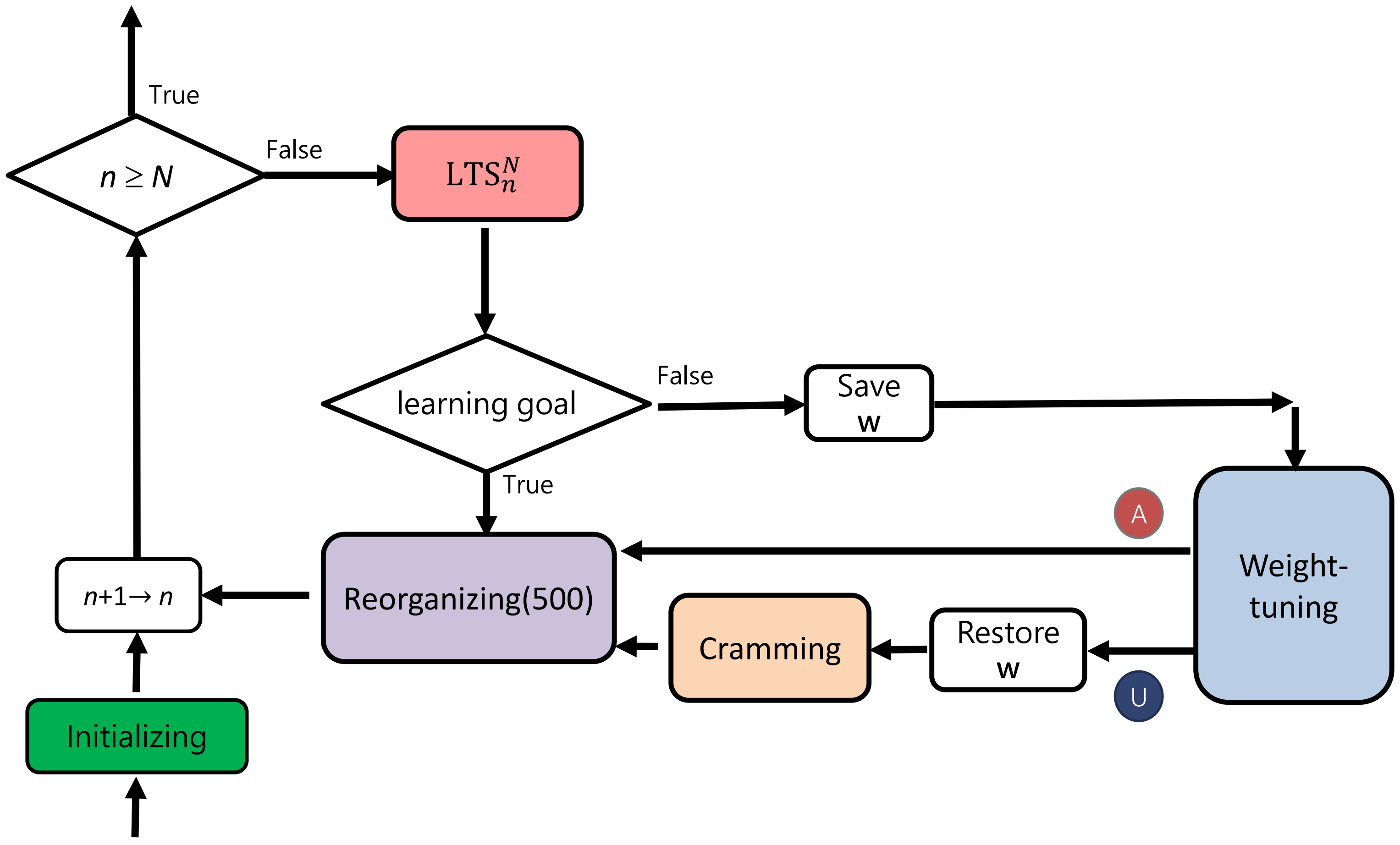
The proposed CSI-LTS-0



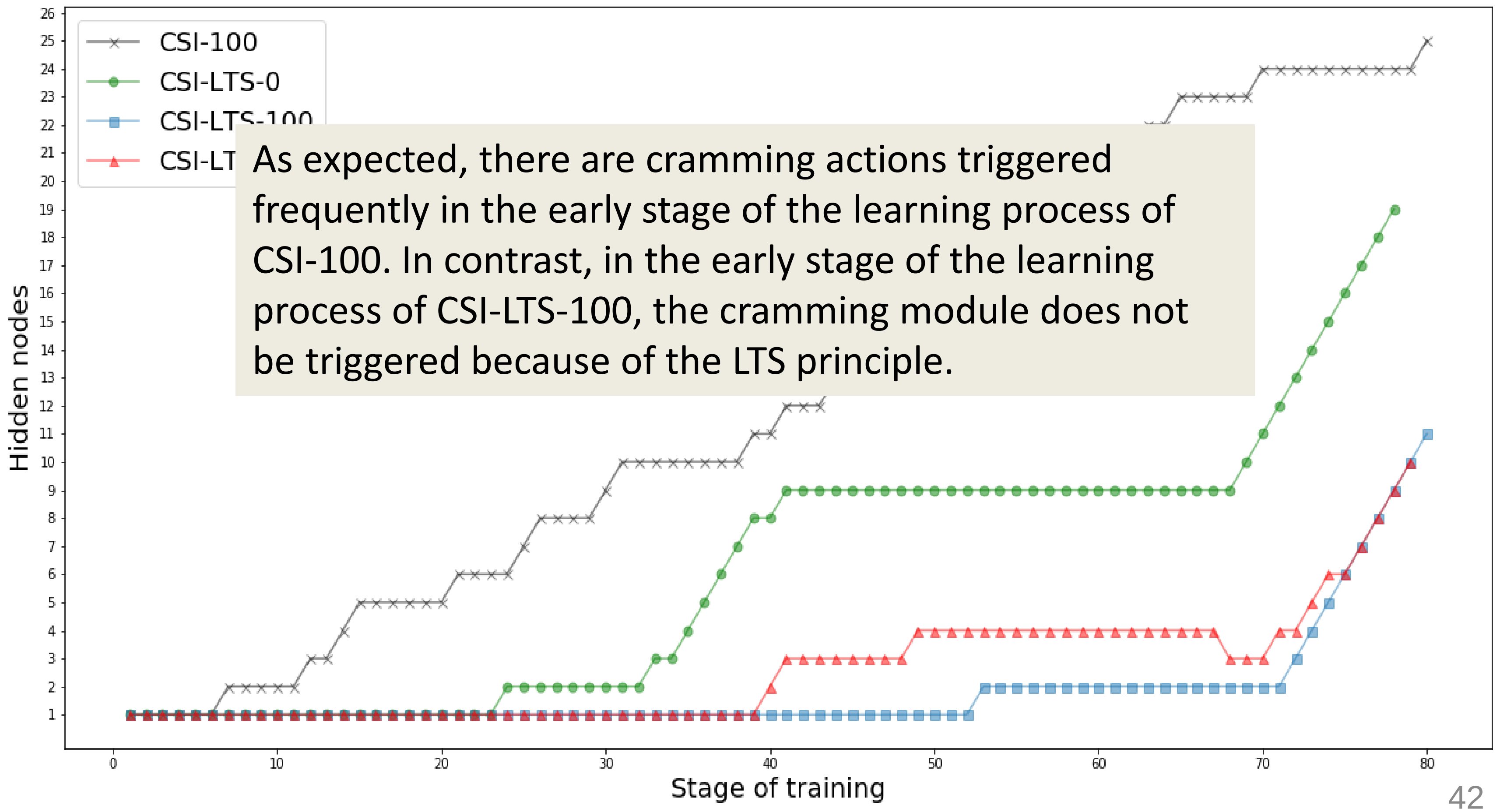
The proposed CSI-LTS-100



The proposed CSI-LTS-500



The evolution of total number of adopted hidden nodes in the learning process of the 1st training data set



As expected, there are cramming actions triggered frequently in the early stage of the learning process of CSI-100. In contrast, in the early stage of the learning process of CSI-LTS-100, the cramming module does not be triggered because of the LTS principle.

Total number of adopted hidden nodes

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	25	19	11	10
2	16			14
3	16			7
4	7			10
5	21			11
6	10			9
7	11	25	13	7
8				
9				
10				
11	13	3	7	11
12	15	12	9	11
13	17	23	10	15
14	7	22	12	12
15	16	9	11	10
16	8	21	9	13
17				
18				
19				
20				
Average	13.85	16.25	9.55	10.20
Standard deviation	5.9	8.1	2.06	2.6

- CSI--100: from 4 to 25.
- CSI-LTS-0: from 3 to 30.
- CSI-LTS-100: from 6 to 12.
- CSI-LTS-500: from 5 to 15.

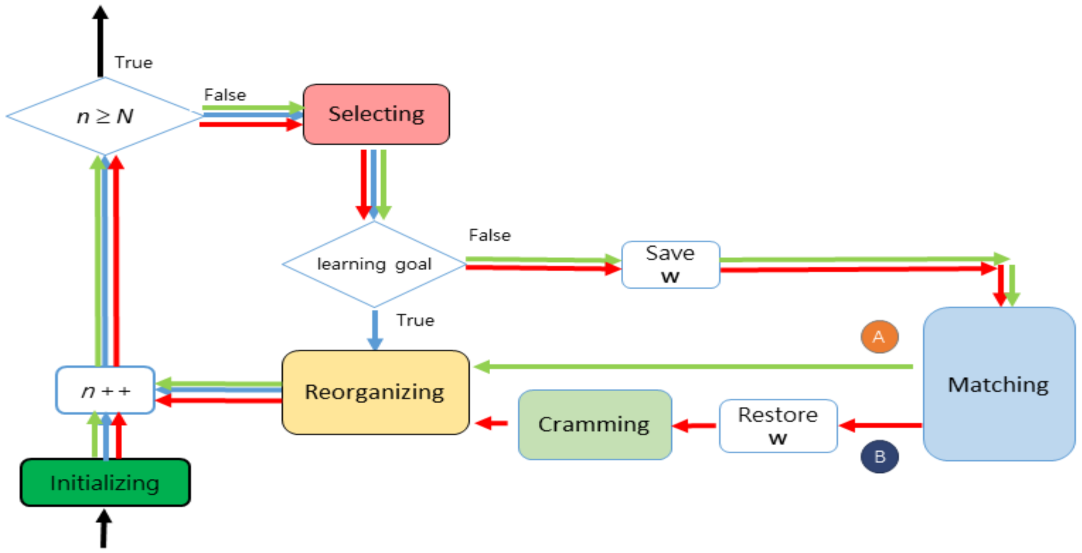
The adoption of LTS and the regularizing module helps reduce the total number of adopted hidden nodes, in average.

In terms of the total number of adopted hidden nodes, the average and standard deviation of CSI-LTS-100 are the smallest.

The occurrence percentages of blue, green and red paths

At every *n*th stage, the proposed mechanism follows one of the following three paths to get an acceptable SLFN:

- blue path
- green path
- red path



CSI-100

In average of 20 training datasets, there are approximately 67.75% of 80 learning processes that go through the blue path, 11.13% that go through the green path, and 21.13% that go through the red path.

Set No.	blue	green	red	
1	62.50%	2.50%	35.00%	
2	73.75%	3.75%	22.50%	
3	63.75%	15.00%	21.25%	
4	71.25%	18.75%	10.00%	
5	58.75%	11.25%	30.00%	
6	60.00%	25.00%	15.00%	
7	71.25%	12.50%	16.25%	
8	61.25%	6.25%	32.50%	
9	60.00%	2.50%	37.50%	
10	67.50%	26.25%	6.25%	
11	72.50%	7.50%	20.00%	
12	68.75%	8.75%	22.50%	
13	68.75%	3.75%	27.50%	
14	73.75%	15.00%	11.25%	
15	68.75%	5.00%	26.25%	
16	70.00%	16.25%	13.75%	
17	78.75%	10.00%	11.25%	
18	63.75%	20.00%	16.25%	
19	78.75%	7.50%	13.75%	
20	61.25%	5.00%	33.75%	
Average	67.75%	11.13%	21.13%	
Standard deviation	6.13%	7.28%	9.27%	

CSI-LTS-0

In average of 20 training datasets, there are approximately 63.31% of 80 learning processes that go through the blue path, 14.31% that go through the green path, and 22.38% that go through the red path.

Set No.	blue	green	red	
1	61.25%	13.75%	25.00%	
2	62.50%	30.00%	7.50%	
3	60.00%	23.75%	16.25%	
4	63.75%	5.00%	31.25%	
5	52.50%	12.50%	35.00%	
6	62.50%	7.50%	30.00%	
7	61.25%	5.00%	33.75%	
8	71.25%	10.00%	18.75%	
9	65.00%	21.25%	13.75%	
10	72.50%	17.50%	10.00%	
11	83.75%	11.25%	5.00%	
12	62.50%	21.25%	16.25%	
13	58.75%	11.25%	30.00%	
14	53.75%	15.00%	31.25%	
15	60.00%	27.50%	12.50%	
16	61.25%	11.25%	27.50%	
17	57.50%	8.75%	33.75%	
18	73.75%	13.75%	12.50%	
19	65.00%	16.25%	18.75%	
20	57.50%	3.75%	38.75%	
Average	63.31%	14.31%	22.38%	
Standard deviation	7.30%	7.37%	10.36%	

CSI-LTS-100

Set No.	blue	green	red
1	71.25%	11.25%	17.50%
2	73.75%	13.75%	12.50%
3	68.75%	15%	16.25%
4	72.5%	11.25%	16.25%
5	72.5%	10%	17.50%
6	68.75%	13.75%	17.50%
7	71.25%	7.5%	21.25%
8	70%	13.75%	16.25%
9	68.75%	17.5%	13.75%
10	75%	5%	20.00%
11	72.5%	13.75%	13.75%
12	77.5%	8.75%	13.75%
13	71.25%	11.25%	17.50%
14	65%	18.75%	16.25%
15	66.25%	16.25%	17.50%
16	68.75%	15%	16.25%
17	72.5%	16.25%	11.25%
18	63.75%	20%	16.25%
19	76.25%	7.5%	16.25%
20	71.25%	10%	18.75%
Average	70.88%	12.81%	16.31%
Standard deviation	3.51%	3.99%	2.42%

In average of 20 training datasets, there are approximately 70.88% of 80 learning processes that go through the blue path, 12.81% that go through the green path, and 16.31% that go through the red path.

CSI-LTS-500

Set No.	blue	green	red
1	72.50%	11.25%	16.25%
2	67.50%	11.25%	21.25%
3	67.50%	21.25%	11.25%
4	77.50%	5.00%	17.50%
5	75.00%	8.75%	16.25%
6	67.50%	17.50%	15.00%
7	67.50%	18.75%	13.75%
8	63.75%	18.75%	17.50%
9	68.75%	13.75%	17.50%
10	75.00%	11.25%	13.75%
11	50.00%	33.75%	16.25%
12	72.50%	12.50%	15.00%
13	75.00%	3.75%	21.25%
14	65.00%	18.75%	16.25%
15	67.50%	16.25%	16.25%
16	73.75%	5.00%	21.25%
17	65.00%	16.25%	18.75%
18	62.50%	27.50%	10.00%
19	58.75%	33.75%	7.50%
20	76.25%	7.50%	16.25%
Average	68.44%	15.63%	15.94%
Standard deviation	6.70%	8.63%	3.56%

In average of 20 training datasets, there are approximately 68.44% of 80 learning processes that go through the blue path, 15.63% that go through the green path, and 15.94% that go through the red path.

The occurrence percentages of blue paths

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	62.50%	61.25%	71.25%	72.50%
2	73.75%	[illegible]	[illegible]	67.50%
3	63.75%	[illegible]	[illegible]	67.50%
4	71.25%	[illegible]	[illegible]	77.50%
5	58.75%	[illegible]	[illegible]	75.00%
6	60.00%	[illegible]	[illegible]	67.50%
7	71.25%	61.25%	71.25%	67.50%
8	61.25%	71.25%	70.00%	63.75%
9	60.00%	65.00%	68.75%	68.75%
10	[illegible]	[illegible]	[illegible]	[illegible]
11	[illegible]	[illegible]	[illegible]	[illegible]
12	[illegible]	[illegible]	[illegible]	[illegible]
13	[illegible]	[illegible]	[illegible]	[illegible]
14	73.75%	53.75%	65.00%	65.00%
15	68.75%	60.00%	66.25%	67.50%
16	70.00%	61.25%	68.75%	73.75%
17	[illegible]	[illegible]	[illegible]	[illegible]
18	[illegible]	[illegible]	[illegible]	[illegible]
19	[illegible]	[illegible]	[illegible]	[illegible]
20	61.25%	57.50%	71.25%	76.25%
Average	**67.75%**	**63.31%**	**70.88%**	**68.44%**
Standard deviation	6.13%	7.30%	3.51%	6.70%

- CSI-100: from 58.75% to 78.75%.
- CSI-LTS-0: from 52.50% to 83.75%.
- CSI-LTS-100: from 63.75% to 76.25%.
- CSI-LTS-500: from 50.00% to 77.50%.

The average occurrence percentage of blue path over these four versions is 67.59% and the standard deviation is 7%. (???)

In terms of the occurrence percentage of blue path, the average of CSI-LTS-100 is the largest and the standard deviation of CSI-LTS-100 is the smallest.

The occurrence percentages of green path

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	2.50%	13.75%	11.25%	11.25%
2	3.75%			11.25%
3	15.00%			21.25%
4	18.75%			5.00%
5	11.25%			8.75%
6	25.00%			17.50%
7	12.50%	5.00%	7.50%	18.75%
8	6.25%	10.00%	13.75%	18.75%
9	2.50%	21.25%	17.50%	13.75%
10				
11				
12				
13				
14	15.00%	15.00%	18.75%	18.75%
15	5.00%	27.50%	16.25%	16.25%
16	16.25%	11.25%	15.00%	5.00%
17				%
18				%
19				%
20				%
Average	**11.13%**	**14.31%**	**12.81%**	**15.63%**
Standard deviation	7.28%	7.37%	3.99%	8.63%

- CSI-100: from 2.50% to 26.25%.
- CSI-LTS-0: from 3.75% to 30.00%.
- CSI-LTS-100: from 5.00% to 20.00%.
- CSI-LTS-500: from 3.75% to 33.75%.

The average occurrence percentage of green path over these four versions is 13.47% and the standard deviation is 7%.

In terms of the occurrence percentage of green path, the average of CSI-LTS-500 is the largest and the standard deviation of CSI-LTS-100 is the smallest.

The occurrence percentages of red path

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	35.00%	25.00%	17.50%	16.25%
2	22.50%	[illegible]	[illegible]	21.25%
3	21.25%	[illegible]	[illegible]	11.25%
4	10.00%	[illegible]	[illegible]	17.50%
5	30.00%	[illegible]	[illegible]	16.25%
6	15.00%	[illegible]	[illegible]	15.00%
7	16.25%	33.75%	21.25%	13.75%
8	32.50%	18.75%	16.25%	17.50%
9	37.50%	13.75%	13.75%	17.50%
10	[illegible]	[illegible]	[illegible]	[illegible]
11	[illegible]	[illegible]	[illegible]	[illegible]
12	[illegible]	[illegible]	[illegible]	[illegible]
13	[illegible]	[illegible]	[illegible]	[illegible]
14	11.25%	31.25%	16.25%	16.25%
15	26.25%	12.50%	17.50%	16.25%
16	13.75%	27.50%	16.25%	21.25%
17	[illegible]	[illegible]	[illegible]	[illegible]
18	[illegible]	[illegible]	[illegible]	[illegible]
19	[illegible]	[illegible]	[illegible]	[illegible]
20	[illegible]	[illegible]	[illegible]	[illegible]
Average	**21.13%**	**22.38%**	**16.31%**	**15.94%**
Standard deviation	9.27%	10.36%	2.42%	3.56%

- CSI-100: from 6.25% to 37.50%.
- CSI-LTS-0: from 7.50% to 38.75%.
- CSI-LTS-100: from 11.25% to 20.00%.
- CSI-LTS-500: from 7.50% to 21.25%.

The average occurrence percentage of red path over these four versions is 18.94% and the standard deviation is 8%.

In terms of the occurrence percentage of red path, the average of CSI-LTS-0 is the largest and the standard deviation of CSI-LTS-100 is the smallest.

Total number of cramming occurrences

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	26	18	12	11
2	16	[illegible]	[illegible]	15
3	15	[illegible]	[illegible]	7
4	6	[illegible]	[illegible]	12
5	22	[illegible]	[illegible]	11
6	10	[illegible]	[illegible]	10
7	11	25	15	9
8	24	13	11	12
9	28	9	9	12
10	3	6	14	9
11	14	2	9	11
12	[illegible]	[illegible]	[illegible]	[illegible]
13	[illegible]	[illegible]	[illegible]	[illegible]
14	[illegible]	[illegible]	[illegible]	[illegible]
15	19	8	12	11
16	9	20	11	15
17	[illegible]	[illegible]	[illegible]	[illegible]
18	[illegible]	[illegible]	[illegible]	[illegible]
19	[illegible]	[illegible]	[illegible]	[illegible]
20	[illegible]	[illegible]	[illegible]	[illegible]
Average	14.90	15.9	11.05	10.75
Standard deviation	7.4	8.3	1.93	2.8

- CSI-100: from 3 to 28.
- CSI-LTS-0: from 2 to 29.
- CSI-LTS-100: from 7 to 14.
- CSI-LTS-500: from 4 to 15.

The adoption of LTS and the regularizing module helps reduce the average total number of cramming occurrences.

In terms of the total number of cramming occurrences, the average of CSI-LTS-500 is the smallest, while the standard deviation of CSI-LTS-100 is the smallest.

Total number of hidden nodes pruned within the learning process

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	2	0	2	2
2	[illegible]	[illegible]	[illegible]	[illegible]
3	[illegible]	[illegible]	[illegible]	[illegible]
4	[illegible]	[illegible]	[illegible]	[illegible]
5	[illegible]	[illegible]	[illegible]	[illegible]
6	[illegible]	[illegible]	[illegible]	[illegible]
7	[illegible]	[illegible]	[illegible]	[illegible]
8	6	2	2	4
9	4	0	4	2
10	[illegible]	[illegible]	[illegible]	2
11	[illegible]	[illegible]	[illegible]	1
12	[illegible]	[illegible]	[illegible]	0
13	4	0	3	1
14	1	2	0	0
15	4	0	2	2
16	2	0	3	3
17	[illegible]	[illegible]	[illegible]	[illegible]
18	[illegible]	[illegible]	[illegible]	[illegible]
19	[illegible]	[illegible]	[illegible]	[illegible]
20	5	0	7	1
Average	2.05	0.65	2.50	1.55
Standard deviation	2.4	0.9	1.76	1.1

The percentage of the reorganizing module that does work

- CSI-100: fourteen out of twenty trials.
- CSI-LTS-0: eight out of twenty trials.
- CSI-LTS-100: seventeen out of twenty trials.
- CSI-LTS-500: sixteen out of twenty trials.

The average total number of hidden nodes pruned over these four versions is 1.69 and the standard deviation is 1.75.

In terms of the number of hidden nodes pruned, the average of CSI-LTS-100 is the largest, while the standard deviation of CSI-LTS-0 is the smallest.

The reorganizing effort

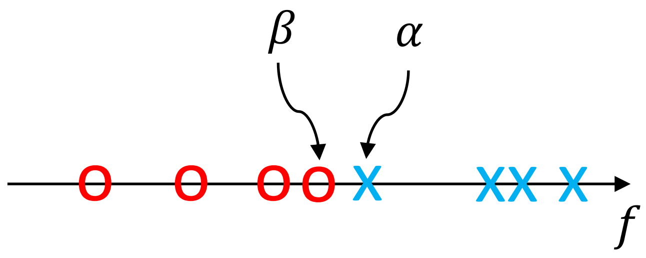
The experiment results show that more than one hidden nodes may be pruned within a reorganizing occurrence.

The best case over all training and testing samples

Generating diagnostic rules from cardiac SPECT data (Kurgan at al., 2001). This problem involved database containing cardiac SPECT heart images collected on 267 patients in stress and rest studies. CLIP3 algorithm was applied to generate diagnostic rules for overall diagnosis of the patient's study, by using information of partial, in the predefined regions of the heart muscle, diagnoses. This is a two-classes problem: first class describes normal patients (55 examples), and second patients with coronary artery disease (212 examples). Three diagnostic rules were generated. The rules accuracy was 84%.

This study also examines an issue in the medical domain: whether the proposed learning mechanism can have better accuracy of diagnoses than other methods in the current literature.

Inferencing mechanism



$$\alpha \equiv \min_{c\in \mathbf{I}_1} f(\mathbf{x}^c, \mathbf{w})\,;\ \beta \equiv \max_{c\in \mathbf{I}_2} f(\mathbf{x}^c, \mathbf{w})$$

learning goal type 3: LSC

When LSC ($\alpha > \beta$) is true, the inferencing mechanism

$$f(\mathbf{x}^c, \mathbf{w}) \geq v \ \ \forall\, c \in \mathbf{I}_1 \text{ and } f(\mathbf{x}^c, \mathbf{w}) \leq -v\ \forall\, c \in \mathbf{I}_2$$

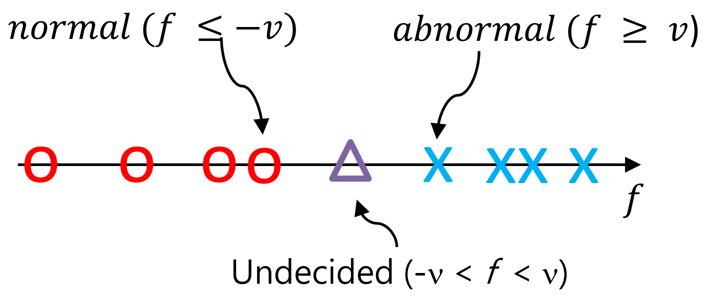
can be set by directly adjusting $\mathbf{w}^o$ according to the following formula:

$$\frac{2v}{\alpha-\beta} w_i^o \rightarrow w_i^o \ \ \forall\, i,$$

The weight vector between the hidden layer and the output node

then $v - \min_{c\in \mathbf{I}_1} \sum_{i=1}^{p} w_i^o a_i^c \rightarrow w_0^o$

The threshold of the output node



Actual / Predicted	Abnormal	Normal
Abnormal	TP	FP
Normal	FN	TN
Undecided	UP	UN

measurement

Predictive Accuracy	(TP+TN) / (TP+TN+FP+FN+UP+UN)
Type 1 error rate	FN / (TP+FN+UP)
Type 2 error rate	FP / (FP+TN+UN)
Sensitivity	TP / (TP+FN+UP)
Specificity	TN / (TF+FP+UN)
Undecided rate	(UP+UN) / (TP+TN+FN+FP+UP+UN)

The accuracy

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	0.417	0.545	0.604	0.626
2	0.358	0.674	0.572	0.524
3	0.342	0.615	0.663	0.722
9	0.246	0.615	0.717	0.690
10	0.711	[illegible]	[illegible]	0.535
11	0.417	0.674	0.733	0.786
12	0.428	0.455	0.711	0.578
13	0.299	0.471	0.743	0.690
14	0.706	0.588	0.701	0.663
15	0.444	0.684	0.556	0.588
16	0.717	0.561	0.663	0.690
17	0.674	0.578	0.684	0.551
18	0.380	0.722	0.652	0.631
19	[illegible]	[illegible]	[illegible]	0.711
20	0.310	0.374	0.711	0.684
Average	0.503	0.603	0.663	0.653
Standard deviation	0.2	0.1	0.07	0.1

Generating diagnostic rules from cardiac SPECT data (Kurgan at al., 2001). This problem involved database containing cardiac SPECT heart images collected on 267 patients in stress and rest studies. CLIP3 algorithm was applied to generate diagnostic rules for overall diagnosis of the patient's study, by using information of partial, in the predefined regions of the heart muscle, diagnoses. This is a two-classes problem: first class describes normal patients (55 examples), and second patients with coronary artery disease (212 examples). Three diagnostic rules were generated. The rules accuracy was 84%.

The best case over all training and testing samples

- [80 + (267-80)*66.3%]/267 = 76.4% ← in average
- [80 + (267-80)*78.1%]/267 = 84.6% ← the best case

Hyperparameter of the inferencing mechanism

- Above are the results regarding v = 0.9.
- Regarding v = 0, v = 0.8, v = 1.0, what the performance will be?

In ANN, this means more hidden nodes.

- In statistics, **overfitting** is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably."
- An **overfitted model** is a statistical model that contains more parameters than can be justified by the data -- Everitt B.S., Skrondal A. (2010), *Cambridge Dictionary of Statistics*, Cambridge University Press.

whether the proposed learning mechanism can help cope with the encountered undesired attractors and alleviate the overfitting tendency

Total number of adopted hidden nodes & the accuracy

Set No.	CSI-100		CSI-LTS-0		CSI-LTS-100		CSI-LTS-500	
1	25	0.417	19	0.545	11	0.604	10	0.626
2	16	0.358	5	0.674	7	0.572	14	0.524
9	25	0.246	10	0.615	6	0.717	11	0.690
10	4	0.711	5	0.620	11	0.781	8	0.535
11	13	0.417	3	0.674	7	0.733	11	0.786
12	15	0.428	12	0.455	9	0.711	11	0.578
17	8	0.674	23	0.578	8	0.684	13	0.551
18	12	0.380	9	0.722	12	0.652	7	0.631
19	10	0.754	14	0.695	8	0.679	5	0.711
20	17	0.316	30	0.374	7	0.711	11	0.684
Average	13.85	0.503	16.25	0.603	9.55	0.663	10.20	0.653
Standard deviation	5.9	0.2	8.1	0.1	2.06	0.07	2.6	0.1

It seems that the CSI-LTS-100 mechanism can help cope with the encountered undesired attractors and alleviate the overfitting tendency.

- An **overfitted model** is a statistical model that contains more parameters than can be justified by the data -- Everitt B.S., Skrondal A. (2010), *Cambridge Dictionary of Statistics*, Cambridge University Press.
- The CSI-LTS-100 version has a smallest average number of adopted hidden nodes and a best average accuracy.

It seems that CSI-LTS-100 is better than CSI-100, CSI-LTS-0 and CSI-LTS-500.

Total training Time (Sec.)

Set No.	CSI-100	CSI-LTS-0	CSI-LTS-100	CSI-LTS-500
1	1995	656	76	269
2				
3				
4				
5				
6	[illegible]	[illegible]	[illegible]	[illegible]
7	363	622	486	63
8	1459	78	98	67
9	1926	37	39	174
10	332	94	487	69
11	882	15	356	63
12	1255	576	38	423
13	1139	1612	315	214
14				
15				
16	1391	145	139	226
17	2290	839	26	130
18				
19				
20	[illegible]	[illegible]	[illegible]	[illegible]
Average	1250.95	604.55	201.10	149.90
Standard deviation	577.9	722.2	179.41	130.7

- CSI-100: from 332 seconds to 2290 seconds.
- CSI-LTS-0: from 15 seconds to 2702 seconds.
- CSI-LTS-100: from 26 seconds to 487 seconds.
- CSI-LTS-500: from 19 seconds to 452 seconds.

The adoption of LTS and “longer” regularizing module is good for speeding up the learning process.

In terms of the training time, the average and standard deviation of CSI-LTS-500 are the smallest.