Coping with the learning dilemma: cramming with ReLU for real-number inputs

國立政治大學 資訊管理學系
蔡瑞煌 特聘教授

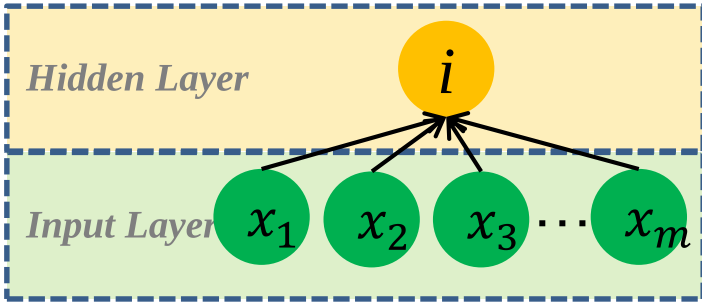
The AI application

- The AI expertise (developing a new AI system is like playing with Lego – lots of pre-built or self-built modules; transfer learning; learning algorithms: supervised, unsupervised, or reinforcement; ...)
- The math expertise / the computer expertise (HW, SW, cloud, edge computing, ...)
- The application domain know-how
- The performance evaluation

Where we are now...

The AI application →
ideas/concepts →
(pre-existed & extra) modules →
new learning algorithm →
codes →
intelligent system for the AI application

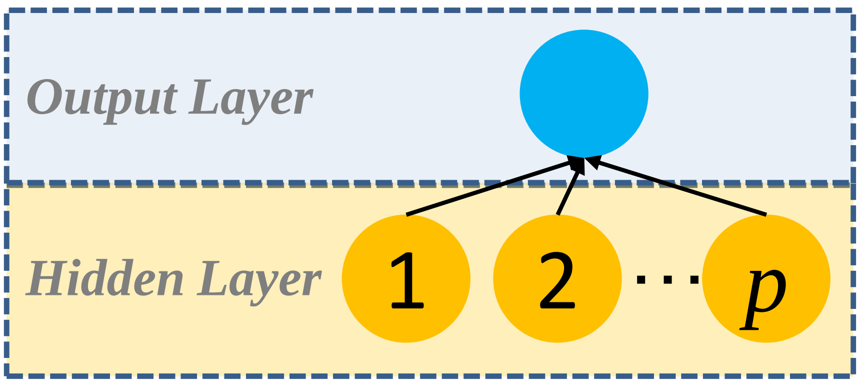
The forward operation SLFN with one output node



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;

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

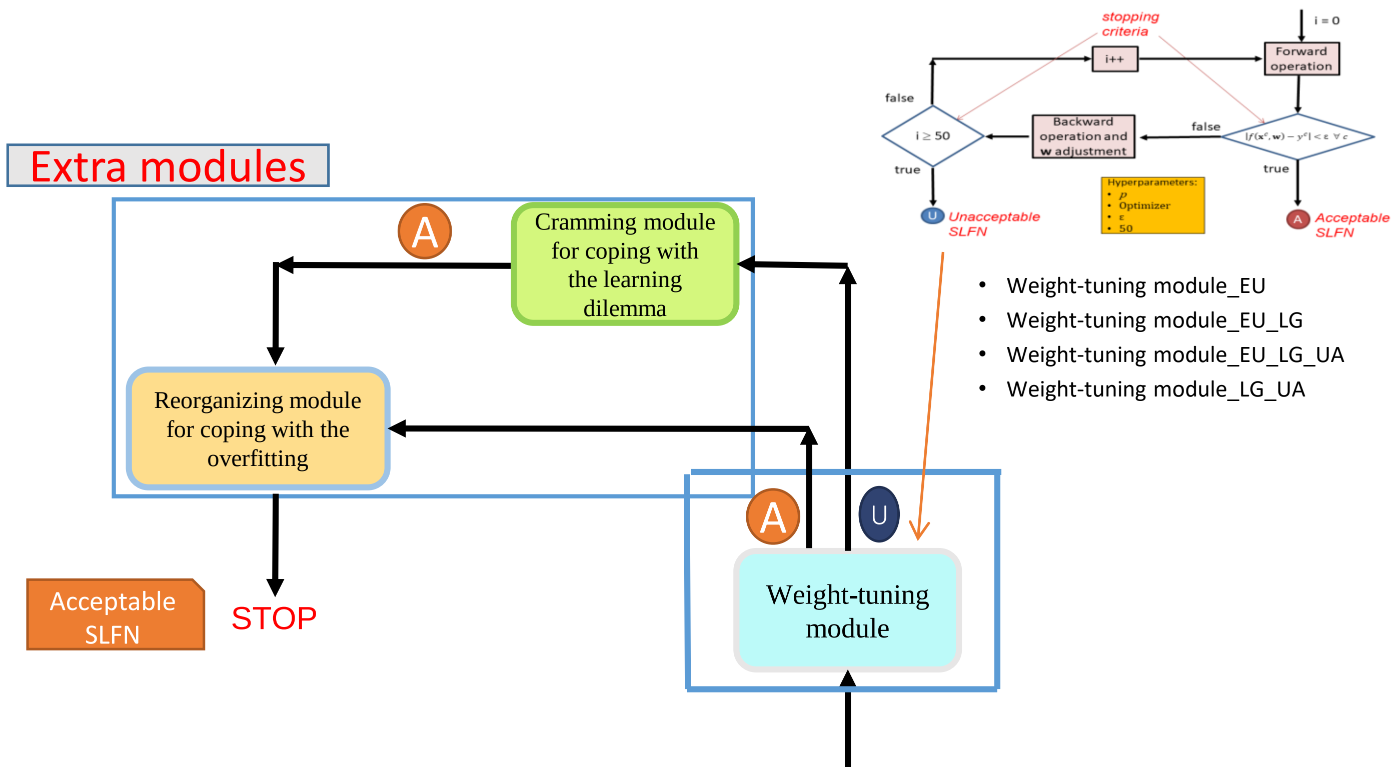
Depend on the application!

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

Should specify (binary or real numbers)

Where we are now...

Cope with the overfitting and learning dilemma

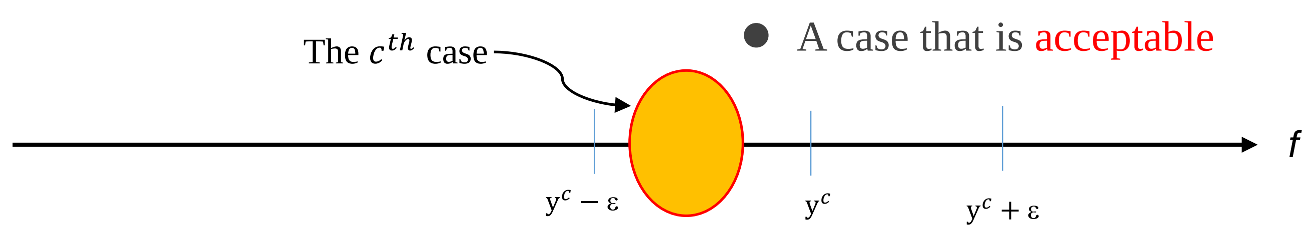


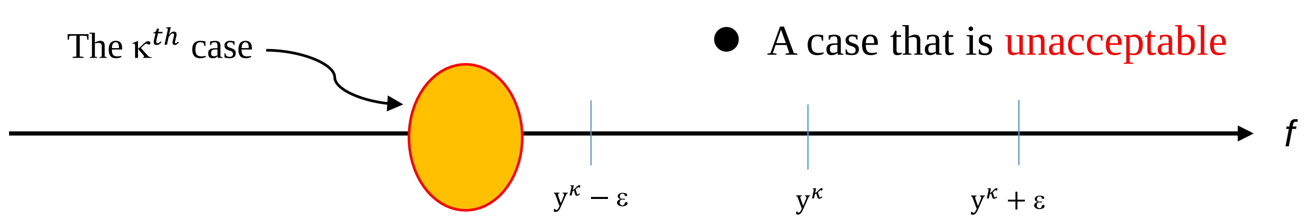
- Weight-tuning module_EU
- Weight-tuning module_EU_LG
- Weight-tuning module_EU_LG_UA
- Weight-tuning module_LG_UA

The unacceptable case

- The learning goal focuses on each individual case, not the loss function.
- The acceptability of each case is related with the learning goal.

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





Cramming (education)

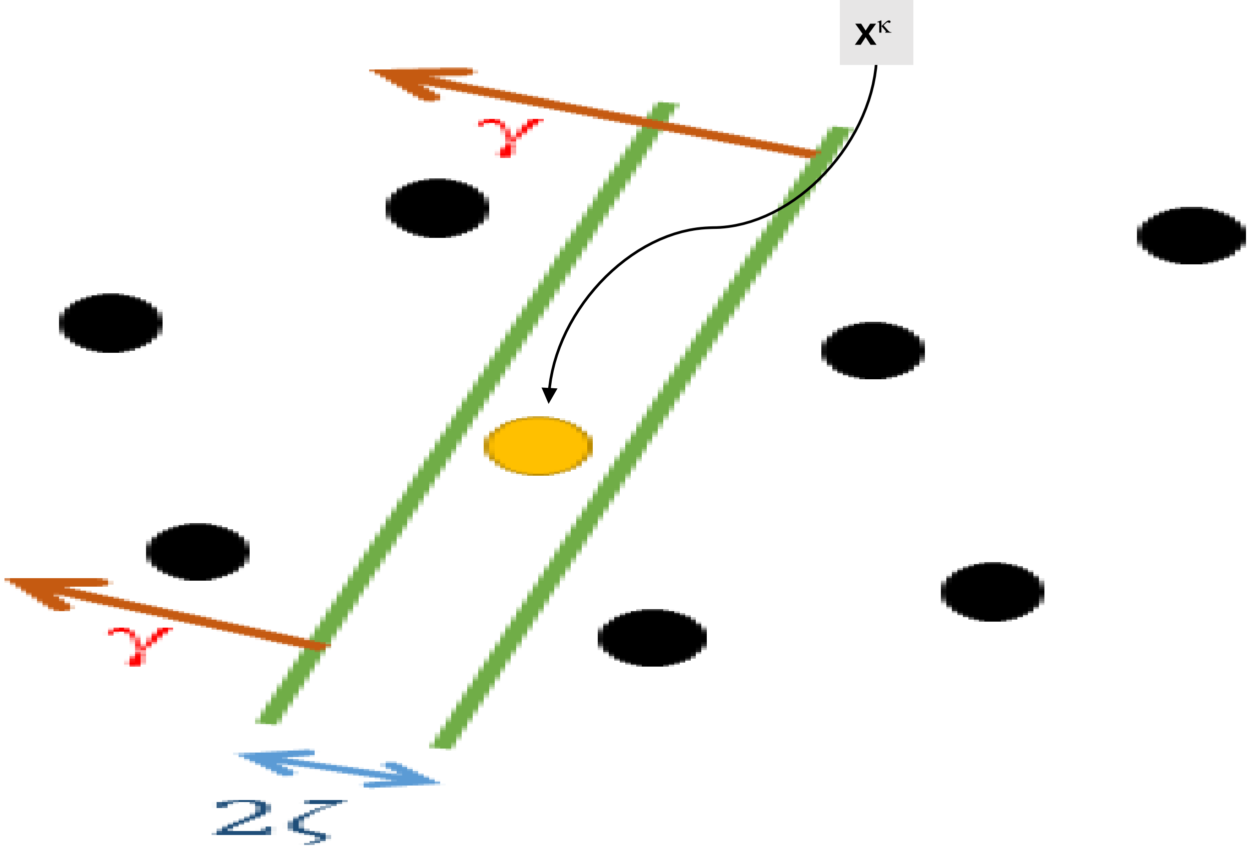
(Cramming (education) - Wikipedia)

- ✓ In education, **cramming** (also known as **mugging** or **swotting**, from *swot*, akin to "sweat", meaning "to study with determination") is the practice of working intensively to absorb large volumes of information in short amounts of time.
- ✓ It is often done by students in preparation for upcoming exams, especially just before they are due.
- ✓ Usually the student's priority is to obtain shallow recall suited to a superficial examination protocol, rather than to internalize the deep structure of the subject matter.
- ✓ Cramming is often discouraged by educators because the hurried coverage of material tends to result in poor long-term retention of material.

The cramming mechanism

Idea/concept:

- Assumption: Only the output of κ^{th} case is unacceptable regarding the learning goal, while outputs of other cases are acceptable regarding the learning goal.
- Method: With recruiting extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.



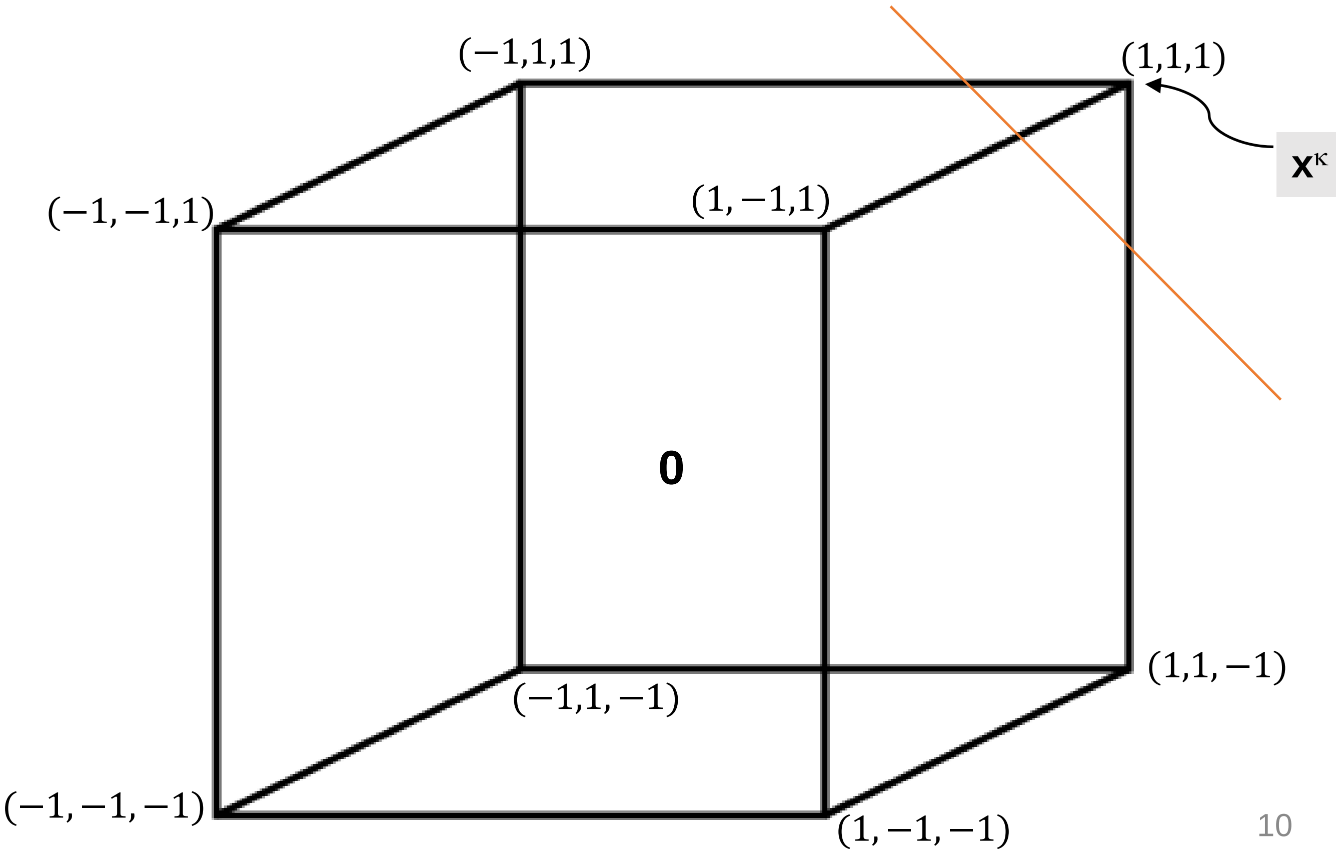
Where we are now...

The isolation module with ReLU in the $\{-1,1\}^m$ space

Idea/concept:

- Assumption: Only the output of κ^{th} case is unacceptable regarding the learning goal, while outputs of other cases are acceptable regarding the learning goal.
- Method: With recruiting an extra hidden node, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.

The $\{-1,1\}^3$ space



Where we are now...

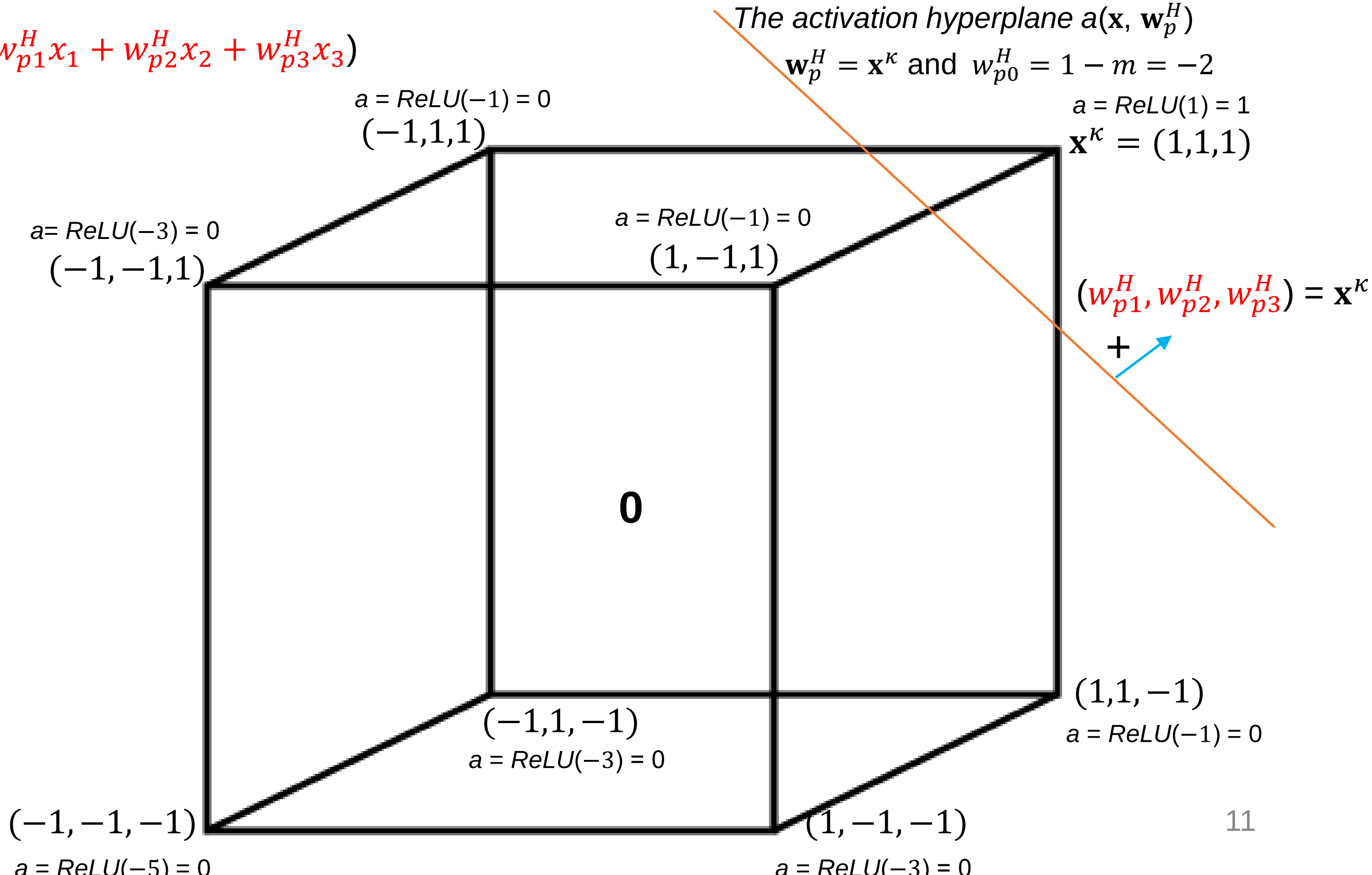
The isolation module with ReLU in the $\{-1,1\}^m$ space

Let $p + 1 \rightarrow p$, add the new p^{th} hidden node with $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa\ \forall\ j$) and $w_{p0}^H = 1 - m$

Because of $\mathbf{x} \in \{-1, 1\}^m$, this isolation module renders only $a(\mathbf{x}^\kappa, \mathbf{w}_p^H)$, which equals $ReLU(1)$, be 1 and the other $a(\mathbf{x}^c, \mathbf{w}_p^H)$s, which equal $ReLU(-1)$, $ReLU(-3)$, $ReLU(-5)$, …, be 0.

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

The $\{-1,1\}^3$ space



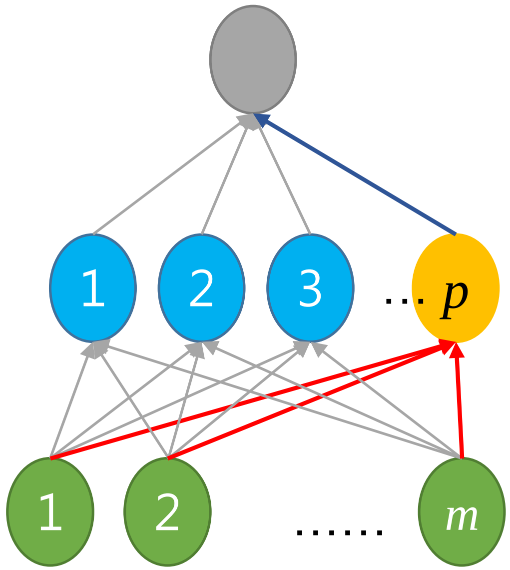
A rule-based mechanism

The cramming module_ReLU_BI_SO_LGT1_SU

classification problems with LGT1

Let $p+1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way to make LGT1 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\, c \in \mathbf{I}\}$ true:

1) $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa\ \forall\, j$) and $w_{p0}^H = 1 - m$
2) $w_p^o = y^\kappa - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^\kappa$



$$w_p^o = y^\kappa - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^\kappa$$

$$w_{p0}^H = 1 - m$$

$$w_{pj}^H = x_j^\kappa\ \forall\, j$$

The cramming module_ReLU_BI_SO_LGT3_SU

classification problems with LGT3

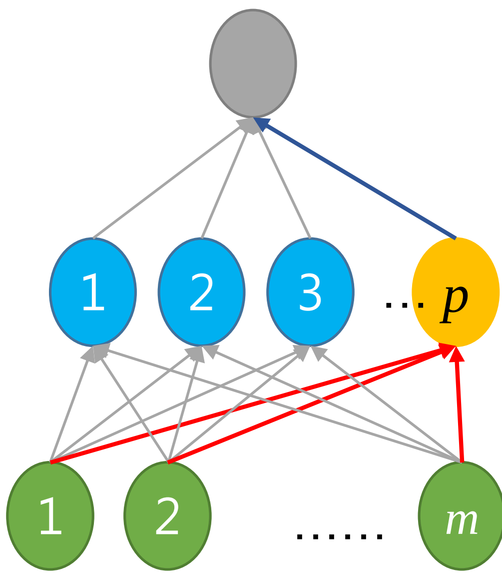
Let $p+1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way to make LSC regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\ c \in \mathbf{I}\}$ true:

1) $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa\ \forall\ j$) and $w_{p0}^H = 1 - m$

2) $$w_p^o = \begin{cases} \max\limits_{u \in \mathbf{I}_2 - \{\kappa\}} \sum_{i=1}^{p-1} w_i^o a_i^u - \sum_{i=1}^{p-1} w_i^o a_i^\kappa & if\ \kappa \in \mathbf{I}_2 \\ \min\limits_{v \in \mathbf{I}_1 - \{\kappa\}} \sum_{i=1}^{p-1} w_i^o a_i^v - \sum_{i=1}^{p-1} w_i^o a_i^\kappa & if\ \kappa \in \mathbf{I}_1 \end{cases}$$

$$\beta' = w_0^o + \max_{u \in \mathbf{I}_2 - \{\kappa\}} \sum_{i=1}^{p-1} w_i^o a_i^u$$

$$\alpha' = w_0^o + \min_{v \in \mathbf{I}_1 - \{\kappa\}} \sum_{i=1}^{p-1} w_i^o a_i^v$$



$$w_p^o = \begin{cases} \max\limits_{u \in \mathbf{I}_2 - \{\kappa\}} \sum_{i=1}^{p-1} w_i^o a_i^u - \sum_{i=1}^{p-1} w_i^o a_i^\kappa & if\ \kappa \in \mathbf{I}_2 \\ \min\limits_{v \in \mathbf{I}_1 - \{\kappa\}} \sum_{i=1}^{p-1} w_i^o a_i^v - \sum_{i=1}^{p-1} w_i^o a_i^\kappa & if\ \kappa \in \mathbf{I}_1 \end{cases}$$

$$w_{p0}^H = 1 - m$$

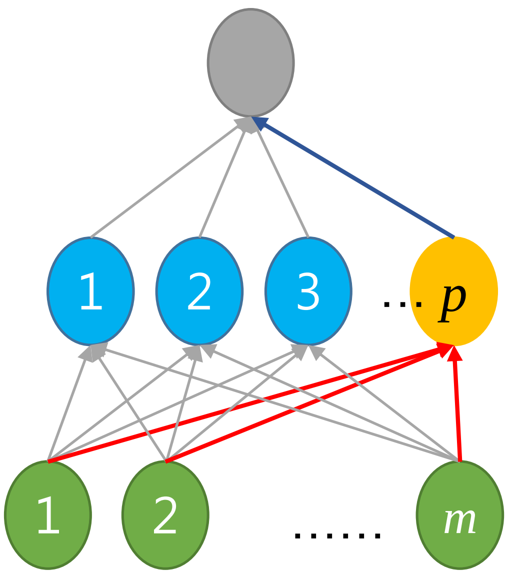
$$w_{pj}^H = x_j^\kappa\ \forall\ j$$

The cramming module_ReLU_BI_SO_RE_SU

regression problems

Let $p+1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way to make $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2 \; \forall \; c \in \mathbf{I}$ true:

1) $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa \; \forall \; j$) and $w_{p0}^H = 1 - m$
2) $w_p^o = y^\kappa - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^\kappa$



$$w_p^o = y^\kappa - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^\kappa$$

$$w_{p0}^H = 1 - m$$

$$w_{pj}^H = x_j^\kappa \; \forall \; j$$

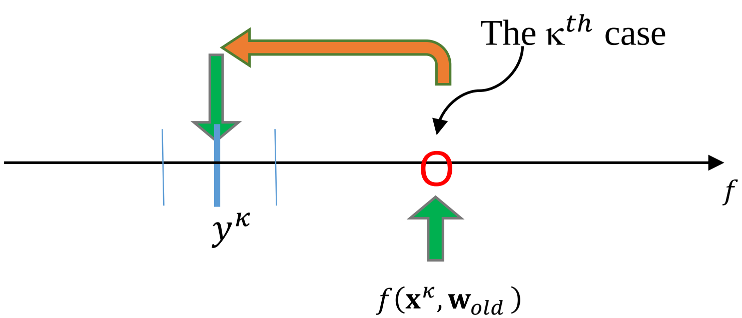
Note that the cramming module_ReLU_BI_SO_RE_SU is the same as the cramming module_ReLU_BI_SO_LGT1_SU.

The cramming module for multiple unacceptable cases

The cramming module – the case of binary-number inputs, real-number desired output & ReLU

regression problems

- $\mathbf{x} \in \{-1, 1\}^m$; $f \in$ R; $y^c \in$ R.
- Assume the current SLFN makes $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2$ $\forall$ $c \in$ **I**-$\{\kappa_1, \kappa_2, \ldots\}$ true, but $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2$ $\forall$ $c \in$ **I** is false.
- Method: For each unacceptable case ($\mathbf{x}^\kappa$, y^κ), with recruiting an extra hidden node, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.



The cramming module – the case of binary-number inputs, real-number desired output & ReLU

regression problems

For each unacceptable case ($\mathbf{x}^{\kappa}$, y^{κ}):

Let $p + 1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way:

1) $\mathbf{w}_p^H = \mathbf{x}^{\kappa}$ (i.e., $w_{pj}^H = x_j^{\kappa}\ \forall j$) and $w_{p0}^H = 1 - m$

Because of $\mathbf{x} \in \{-1, 1\}^m$, the isolation module renders only $a(\mathbf{x}^{\kappa}, \mathbf{w}_p^H)$, which equals $ReLU(1)$, be 1 and the other $a(\mathbf{x}^c, \mathbf{w}_p^H)$s, which equal $ReLU(-1)$, $ReLU(-3)$, $ReLU(-5)$, …, be 0.

The cramming module – the case of binary-number inputs, real-number desired output & ReLU

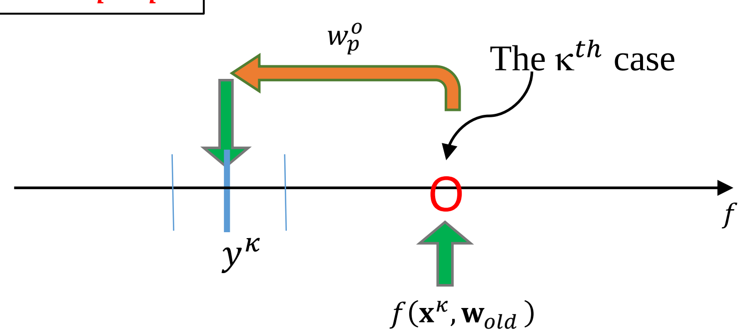
regression problems

For each unacceptable case $(\mathbf{x}^{\kappa}, y^{\kappa})$:

Let $p+1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way:

1) $\mathbf{w}_p^H = \mathbf{x}^{\kappa}$ (i.e., $w_{pj}^H = x_j^{\kappa}\ \forall\ j$) and $w_{p0}^H = 1 - m$
2) $w_p^o = y^{\kappa} - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^{\kappa}$

$$f(\mathbf{x}^{\kappa}, \mathbf{w}_{old}) = w_0^o + \sum_{i=1}^{p-1} w_i^o a_i^{\kappa}$$
$$f(\mathbf{x}^{c}, \mathbf{w}_{new}) = f(\mathbf{x}^{c}, \mathbf{w}_{old}) + w_p^o a_p^c$$

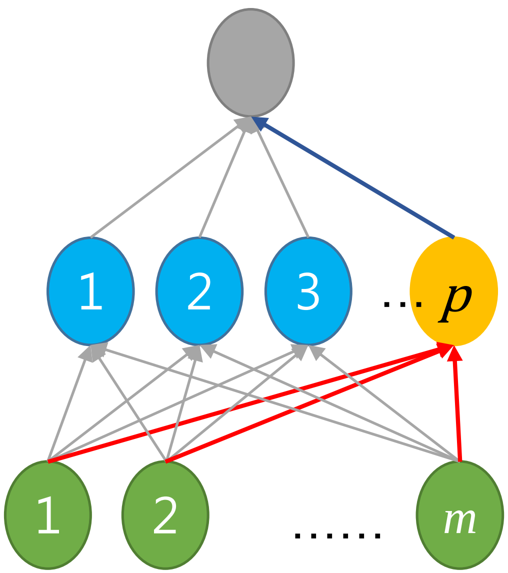


The cramming module_ReLU_BI_SO_RE_MU

For each unacceptable case $(\mathbf{x}^{\kappa}, y^{\kappa})$:

Let $p + 1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way:

1) $\mathbf{w}_p^H = \mathbf{x}^{\kappa}$ (i.e., $w_{pj}^H = x_j^{\kappa} \; \forall \; j$) and $w_{p0}^H = 1 - m$
2) $w_p^o = y^{\kappa} - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^{\kappa}$



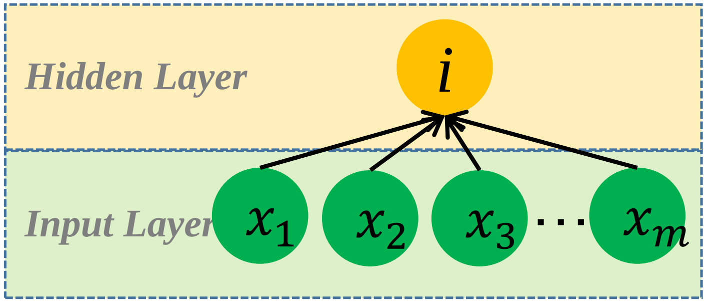
$$w_p^o = y^{\kappa} - w_0^o - \sum_{i=1}^{p-1} w_i^o a_i^{\kappa}$$

$$w_{p0}^H = 1 - m$$

$$w_{pj}^H = x_j^{\kappa} \; \forall \; j$$

The cramming module for SLFN with multiple output nodes

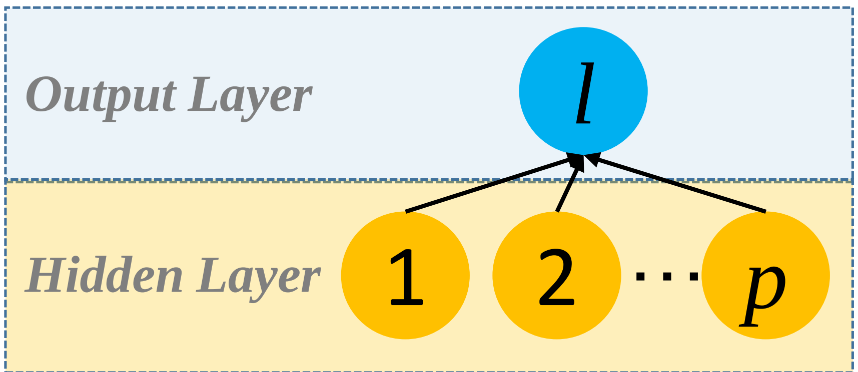
The forward operation SLFN with multiple output nodes



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:

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

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

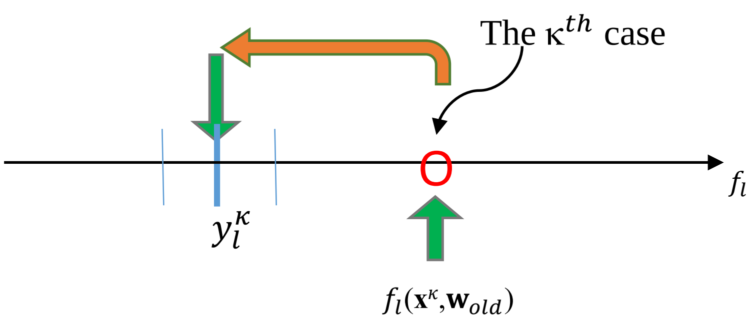
$E_N(\mathbf{w}) \equiv \frac{1}{N}\sum_{c\in\mathbf{I}}\sum_{l=1}^{q}(f_l(\mathbf{x}^c,\mathbf{w}) - y_l^c)^2$: the loss function;

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

The cramming module – the case of binary-number inputs, real-number desired outputs & ReLU

regression problems

- $\mathbf{x} \in \{-1, 1\}^m$; $f \in R^q$; $y^c \in R^q$.
- Assume the current SLFN makes $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}\ \forall\ l$ true, but $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}\ \forall\ l$ is false.
- Method: Regarding every l^{th} output node, in which $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false, with recruiting an extra hidden node, the κ^{th} input is isolated from all other inputs so that its (wrong) output can be changed into the right value while outputs of other inputs are still the same.



The cramming module – the case of binary-number inputs, real-number desired outputs & ReLU

regression problems

Regarding every l^{th} output node, in which $(e_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}$-$\{\kappa\}$ is true, but $(e_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false:

Let $p + 1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way:

1) $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa\ \forall\ j$) and $w_{p0}^H = 1 - m$
2) $w_{lp}^o = y_l^\kappa - w_{l0}^o - \sum_{i=1}^{p-1} w_{li}^o a_i^\kappa$ and $w_{kp}^o = 0\ \forall\ k \neq l$

$e_l^c \equiv f_l(\mathbf{x}^c, \mathbf{w}_{old}) - y_l^c$

Because of $\mathbf{x} \in \{-1, 1\}^m$, this isolation module renders only $a(\mathbf{x}^\kappa, \mathbf{w}_p^H)$, which equals *ReLU*(1), be 1 and the other $a(\mathbf{x}^c, \mathbf{w}_p^H)$s, which equal *ReLU*(-1), *ReLU*(-3), *ReLU*(-5), …, be 0.

The cramming module – the case of binary-number inputs, real-number desired outputs & ReLU

regression problems

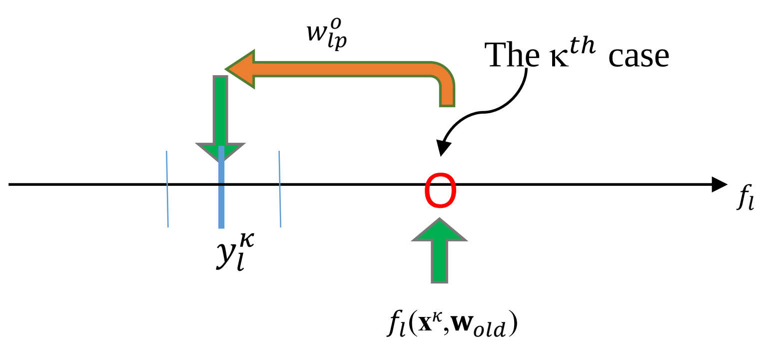
Regarding every l^{th} output node, in which $(e_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(e_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false:

Let $p + 1 \rightarrow p$, add the new p^{th} hidden node to the existing SLFN, and then assign $\mathbf{w}_p^H$, w_{p0}^H and w_p^o in the following way:

1) $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa\ \forall\ j$) and $w_{p0}^H = 1 - m$
2) $w_{lp}^o = y_l^\kappa - w_{l0}^o - \sum_{i=1}^{p-1} w_{li}^o a_i^\kappa$ and $w_{kp}^o = 0\ \forall\ k \neq l$

$$f_l(\mathbf{x}^\kappa, \mathbf{w}_{old}) = w_0^o + \sum_{i=1}^{p-1} w_i^o a_i^\kappa$$
$$f_l(\mathbf{x}^c, \mathbf{w}_{new}) = f_l(\mathbf{x}^c, \mathbf{w}_{old}) + w_{lp}^o a_p^c$$

$$e_l^c \equiv f_l(\mathbf{x}^c, \mathbf{w}_{old}) - y_l^c$$



A rule-based mechanism

The cramming module_ReLU_BI_MO_RE_SU

regression problems

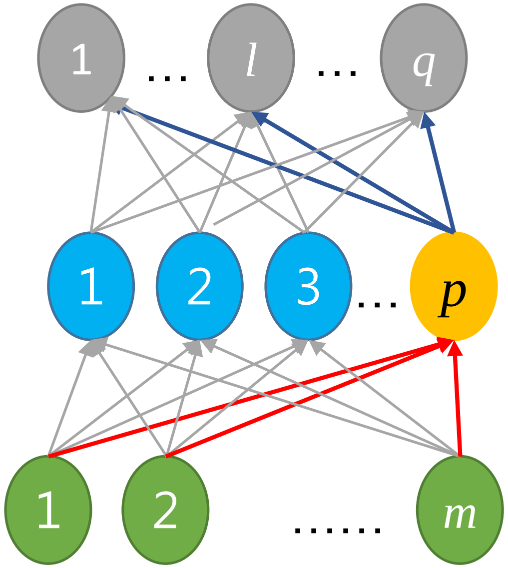
Regarding every l^{th} output node, in which $(e_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(e_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false:

1. Let $p + 1 \rightarrow p$ and add the new p^{th} hidden node to the existing SLFN.
2. Assign $\mathbf{w}_p^H$, w_{p0}^H, w_{lp}^o and $w_{kp}^o\ \forall\ k \neq l$ in the following way:

   1) $\mathbf{w}_p^H = \mathbf{x}^\kappa$ (i.e., $w_{pj}^H = x_j^\kappa\ \forall\ j$) and $w_{p0}^H = 1 - m$

   2) $w_{lp}^o = y_l^\kappa - w_{l0}^o - \sum_{i=1}^{p-1} w_{li}^o a_i^\kappa$ and $w_{kp}^o = 0\ \forall\ k \neq l$

$e_l^c \equiv f_l(\mathbf{x}^c, \mathbf{w}_{old}) - y_l^c$



$w_{lp}^o = y_l^\kappa - w_{l0}^o - \sum_{i=1}^{p-1} w_{li}^o a_i^\kappa$ & $w_{kp}^o = 0\ \forall\ k \neq l$

$w_{p0}^H = 1 - m$

$w_{pj}^H = x_j^\kappa\ \forall\ j$

The cramming module for multiple output nodes and multiple unacceptable cases

The cramming module_ReLU_BI_MO_RE_MU

The cramming modules

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

✓The cramming module_ReLU_BI_SO_LGT1_SU

✓The cramming module_ReLU_BI_SO_LGT3_SU

✓The cramming module_ReLU_BI_SO_RE_SU

✓The cramming module_ReLU_BI_SO_RE_MU

✓The cramming module_ReLU_BI_SO_LGT1_MU

✓The cramming module_ReLU_BI_SO_LGT3_MU

✓The cramming module_ReLU_BI_MO_RE_SU

✓The cramming module_ReLU_BI_MO_LGT1_SU

✓The cramming module_ReLU_BI_MO_LGT3_SU

✓The cramming module_ReLU_BI_MO_RE_MU

✓The cramming module_ReLU_BI_MO_LGT1_MU

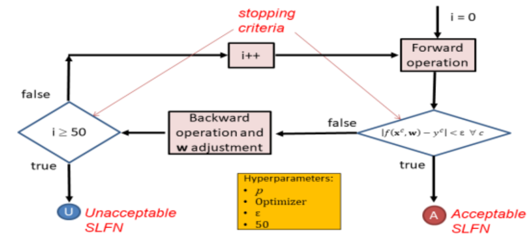
✓The cramming module_ReLU_BI_MO_LGT3_MU

✓Your creative idea

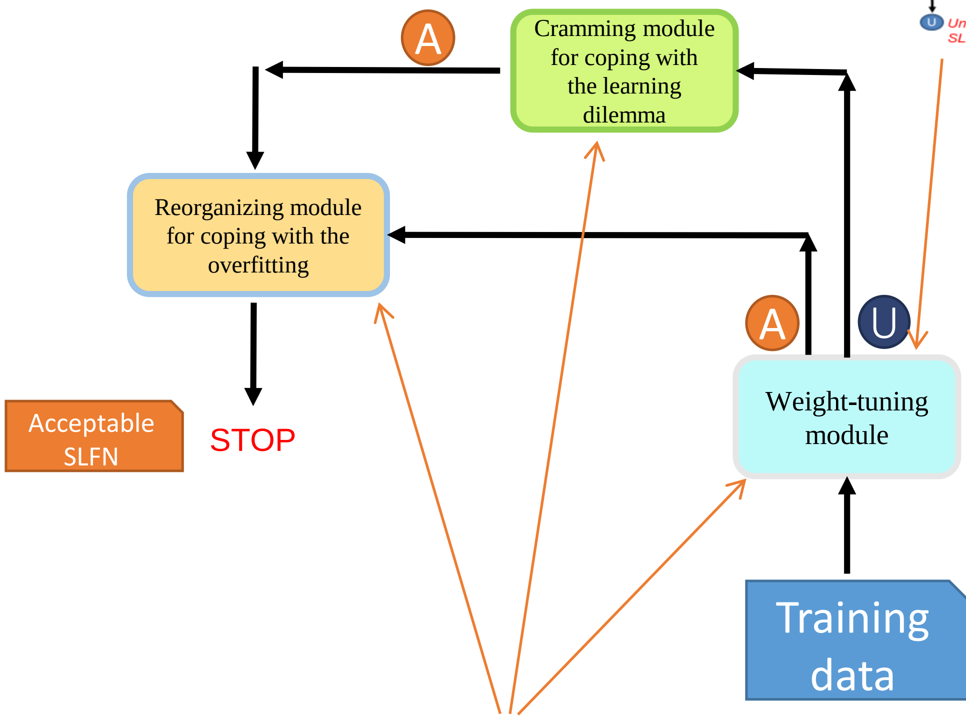
You may derive the

red parts by yourself.

The weight-tuning, cramming and reorganizing modules



- Weight-tuning module_EU
- Weight-tuning module_EU_LG
- Weight-tuning module_EU_LG_UA
- Weight-tuning module_LG_UA



Learning is the process of acquiring new, or modifying existing, knowledge, behaviors, skills, values, or preferences. -- Richard Gross, Psychology: The Science of Mind and Behaviour 6E, Hachette UK, ISBN 978-1-4441-6436-7

Representation and development

(Algorithm - Wikipedia)

- Algorithms can be expressed in many kinds of notation, including natural languages, pseudocode, flowcharts, drakon-charts, programming languages or control tables (processed by interpreters).
  - ✓ Natural language expressions of algorithms tend to be verbose and ambiguous, and are rarely used for complex or technical algorithms.
  - ✓ Pseudocode, flowcharts, drakon-charts and control tables are structured ways to express algorithms that avoid many of the ambiguities common in the statements based on natural language.
  - ✓ Programming languages are primarily intended for expressing algorithms in a form that can be executed by a computer, but are also often used as a way to define or document algorithms.
- Typical steps in the development of algorithms:
  - ✓ Problem definition
  - ✓ Development of a model
  - ✓ Specification of the algorithm
  - ✓ Designing an algorithm
  - ✓ Checking the correctness of the algorithm
  - ✓ Analysis of algorithm
  - ✓ Implementation of algorithm
  - ✓ Program testing
  - ✓ Documentation preparation

Program testing for the new learning mechanism

- Not merely double check the correctness of codes
- New learning mechanism → new learning process ← Double check the learning process (ALWAYS!!!)
- Simple checks:
  1) whether the evolution of $L(\mathbf{w})$ values is reasonable?
  2) whether the tuning of $\mathbf{w}$ is reasonable?
  3) whether the evolution of $f(\mathbf{x}^c, \mathbf{w})$ values all c is reasonable?
- Complicated checks:

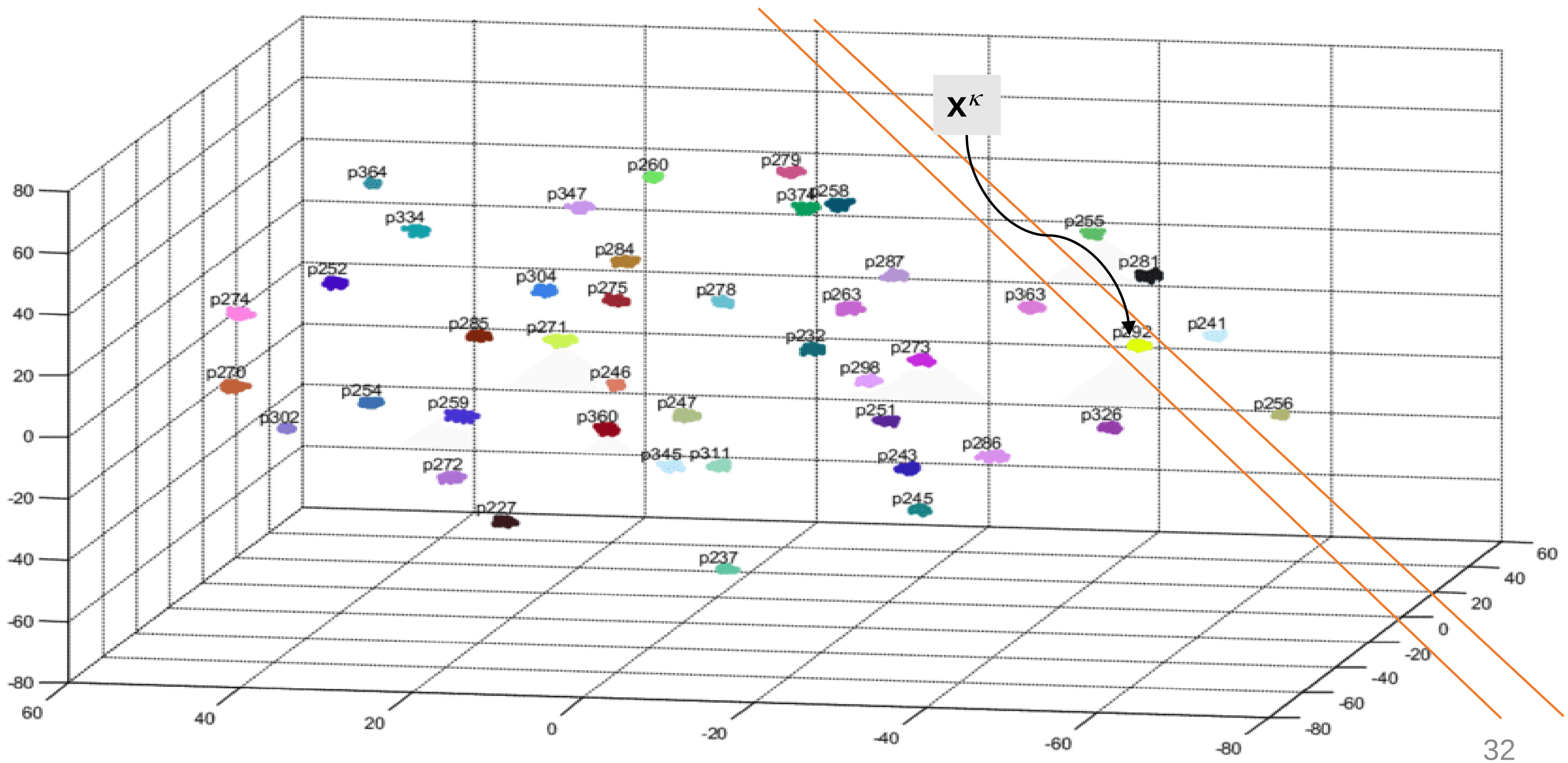
  Not the performance, which is related with the inferencing, but whether the learning process is reasonable?

Real-number inputs

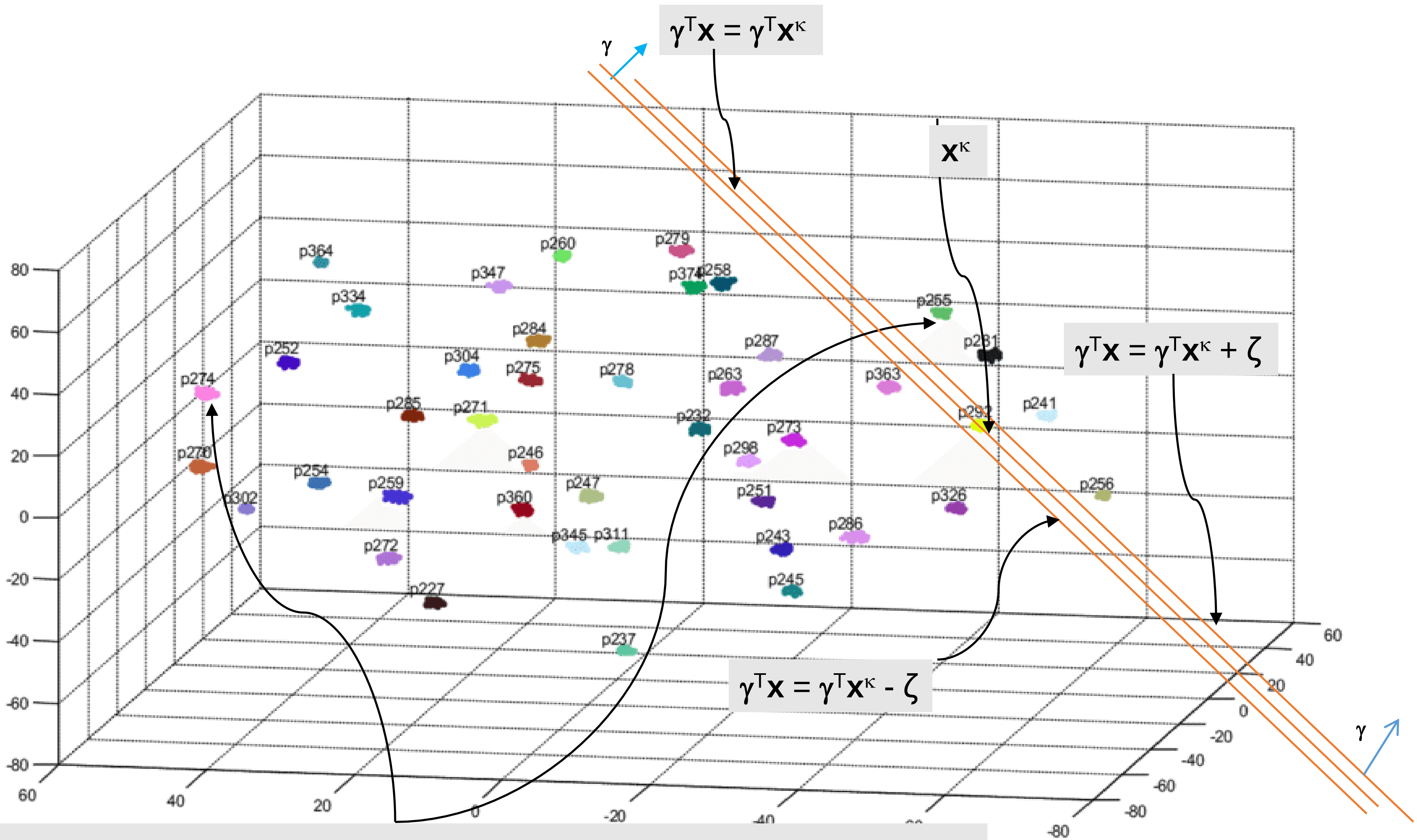
The isolation module with ReLU in the R^m space

Idea/concept:

- Assumption: Only the output of κ^{th} case is unacceptable regarding the learning goal, while outputs of other cases are acceptable regarding the learning goal.
- Method: With recruiting extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.

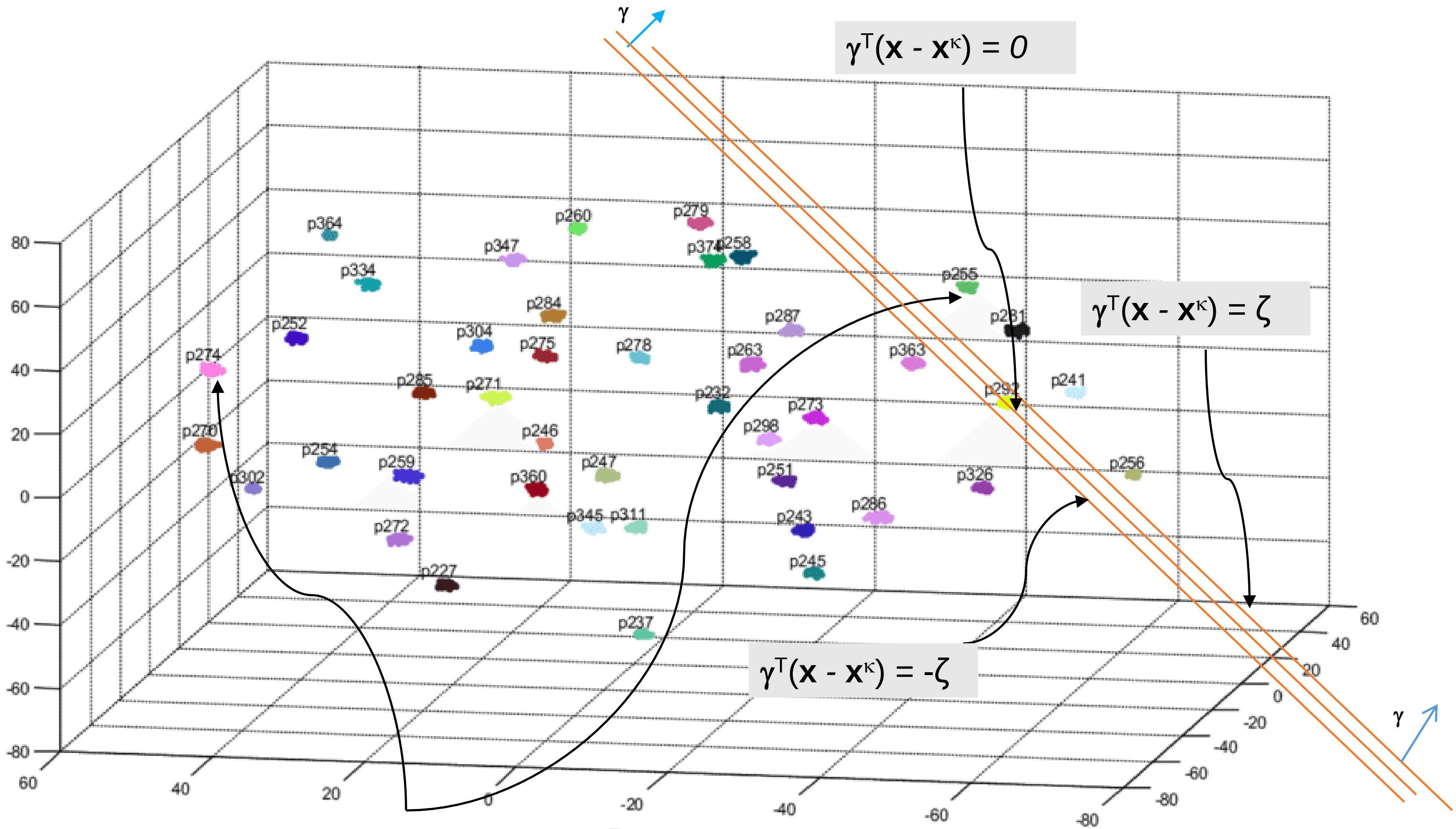


The isolation module with ReLU in the R^m space



$\mathbf{x}^c$ whose $\gamma^T\mathbf{x}$ values are either greater than $\gamma^T\mathbf{x}^\kappa + \zeta$ or less than $\gamma^T\mathbf{x}^\kappa - \zeta$

The isolation module with ReLU in the R^m space



for $\mathbf{x}^c \neq \mathbf{x}^\kappa\ \forall\ c \in \mathbf{I}-\{\kappa\}$: the corresponding $\boldsymbol{\gamma}^T(\mathbf{x}^c - \mathbf{x}^\kappa)$ value is either greater than ζ or less than $-\zeta$.

That is, $(\zeta+\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$.

The contribution of these three extra hidden nodes to the output value

$\Delta f(\mathbf{x}^c) = w_p^o$*[ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)+\zeta$) - 2ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)$) + ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)-\zeta$)]

for $\mathbf{x}^c \neq \mathbf{x}^\kappa\ \forall\ c \in \mathbf{I}-\{\kappa\}$: the corresponding $\gamma^T(\mathbf{x}^c - \mathbf{x}^\kappa)$ value is either greater than ζ or less than $-\zeta$

$\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)$	ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)+\zeta$)	-2ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)$)	ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)-\zeta$)	Δf
2ζ	3ζ	-4ζ	ζ	0
ζ	2ζ	-2ζ	0	0
0.7ζ	1.7ζ	-1.4ζ	0	$0.3\zeta w_p^o$
0.5ζ	1.5ζ	-1.0ζ	0	$0.5\zeta w_p^o$
0.3ζ	1.3ζ	-0.6ζ	0	$0.7\zeta w_p^o$
0	ζ	0	0	ζw_p^o
-0.3ζ	0.7ζ	0	0	$0.7\zeta w_p^o$
-0.5ζ	0.5ζ	0	0	$0.5\zeta w_p^o$
-0.7ζ	0.3ζ	0	0	$0.3\zeta w_p^o$
$-\zeta$	0	0	0	0
-2ζ	0	0	0	0

$\Delta f(\mathbf{x}^\kappa)$

$\Delta f(\mathbf{x}^c) = 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$

The isolation module_Math

Use the isolation module_Math to create an *m*-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ and then pick up a small number ζ such that $(\zeta+\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.

The isolation module_Math

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

Step 2: Let $\mathbf{C}_k \equiv \{c$: $c \in \mathbf{I}-\{\kappa\}$ AND $x_j^c = x_j^\kappa \ \forall \ j = 1, \ldots, k\}$. Considering β_k as the unknown and β_j, $j = 1, \ldots, k-1$, as previously determined, set β_k = the smallest integer that is greater than or equal to 1 and $\sum_{j=1}^{k} \beta_j \left(x_j^c - x_j^\kappa\right) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}-\mathbf{C}_k$.

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

Step 4: Set $\gamma_j = \dfrac{\beta_j}{\sqrt{\sum_{j=1}^{m} {\beta_j}^2}} \ \forall \ j = 1, \ldots, m$.

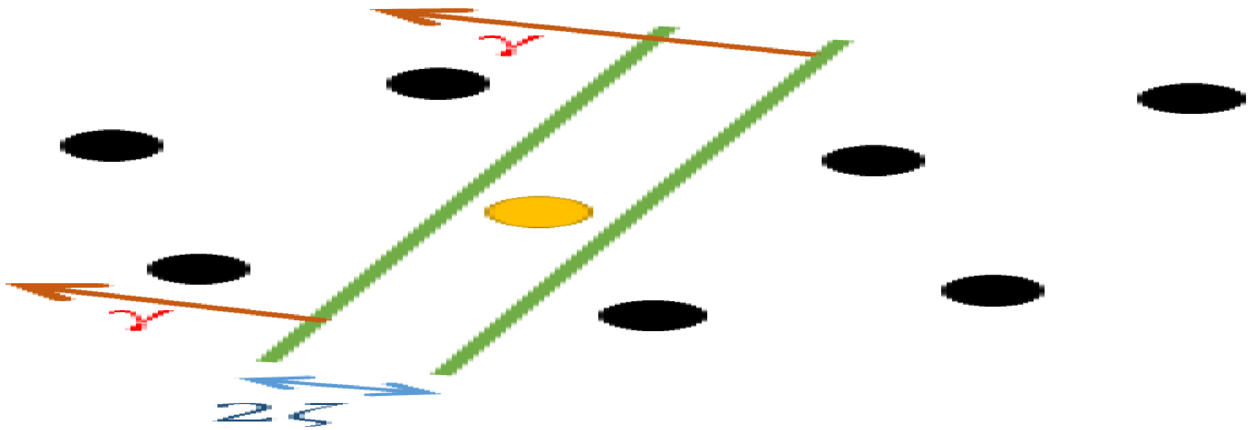
Step 5: Pick up a small number ζ such that $(\zeta+\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ and STOP.

Assume $\mathbf{x}^i \neq \mathbf{x}^j$ when $i \neq j$.

The isolation module_Math

Use the isolation module_Math to create an *m*-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \;\forall\; c \in \mathbf{I}-\{\kappa\}$ and then pick up a small number ζ such that $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \;\forall\; c \in \mathbf{I}-\{\kappa\}$.

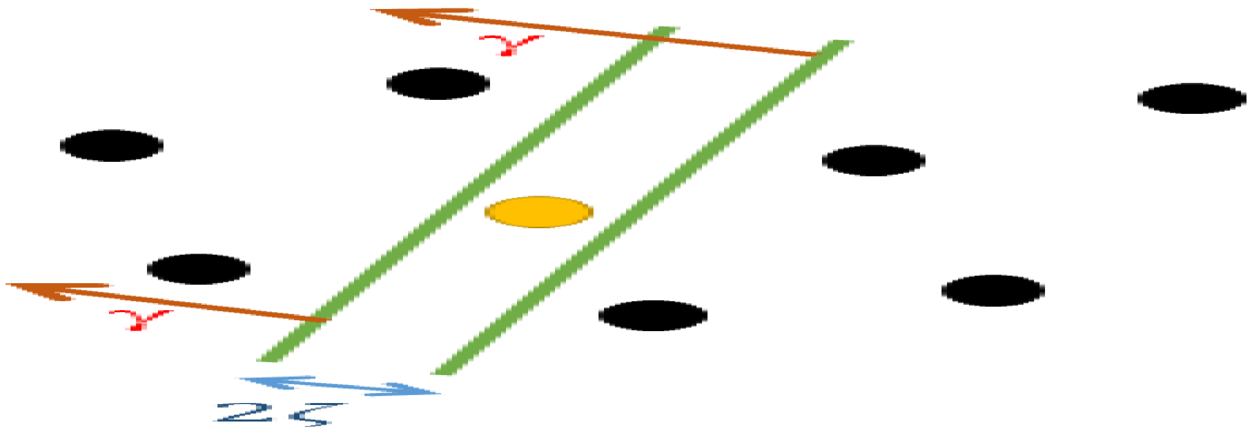
- The statement of $\gamma^T(\mathbf{x}^\kappa - \mathbf{x}^c) \neq 0$ requires that the vector γ cannot lie in the m-1 dimensional hyperplane that contains the origin **0** and the vectors which any $\mathbf{x}^\kappa - \mathbf{x}^c$ vector is perpendicular to.
- For $\mathbf{x}^c \neq \mathbf{x}^\kappa \;\forall\; c \in \mathbf{I}-\{\kappa\}$: the corresponding $\gamma^T(\mathbf{x}^c - \mathbf{x}^\kappa)$ value is either greater than ζ or less than $-\zeta$. That is, $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \;\forall\; c \in \mathbf{I}-\{\kappa\}$.
- Given N-1 vectors $\mathbf{x}^c$ $c \in \mathbf{I}-\{\kappa\}$ and $\mathbf{x}^\kappa$, the isolation module_Math is just one of many ways of creating a γ of length one such that $\gamma^T(\mathbf{x}^\kappa - \mathbf{x}^c) \neq 0 \;\forall\; c \in \mathbf{I}-\{\kappa\}$ and then pick up a small number ζ such that $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \;\forall\; c \in \mathbf{I}-\{\kappa\}$.



The isolation module_R1

Use the isolation module_R1 to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ and then a small number ζ such that $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.

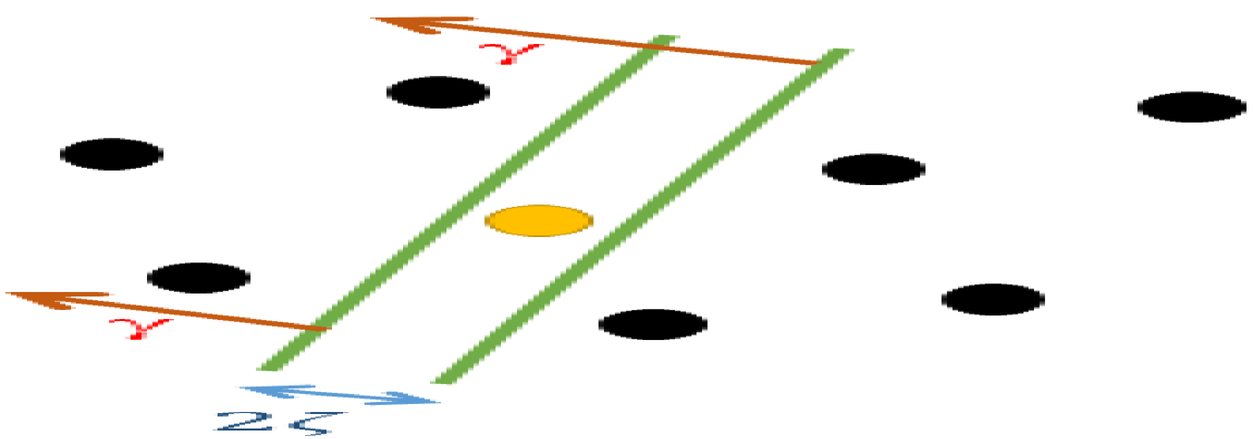
You can use the isolation module_R1 (i.e., the random number generation method) to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ and then a small number ζ such that $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.



The isolation module_R2

Use the isolation module_R2 to pick up a tiny number ζ and then use the random-number generation method to create an *m*-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$.

Or, you can use the isolation_R2 to pick up a tiny number ζ and then create an *m*-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$.



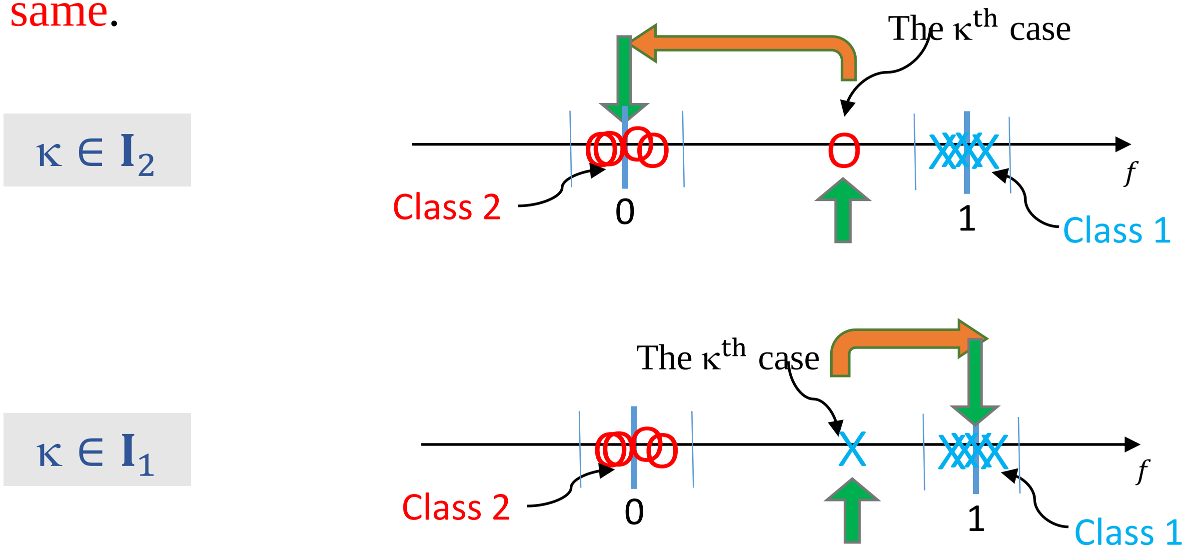
Real-number inputs

Classification applications

The cramming module – the case of real-number inputs, binary-number desired output & ReLU

classification problems with LGT1

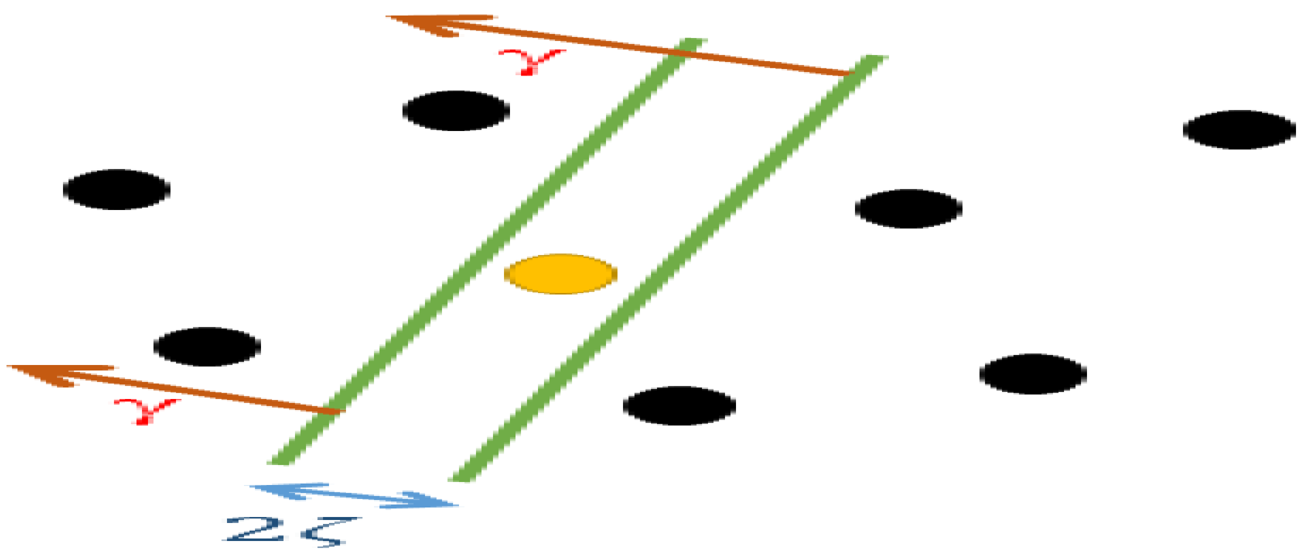
- $\mathbf{x} \in \mathrm{R}^m$; $f \in \mathrm{R}$; $y^c = 1$ if $c \in \mathbf{I}_1$; $y^c = 0$ if $c \in \mathbf{I}_2$.
- Assume the current SLFN makes LGT1 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}\}$ true, but LGT1 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\ c \in \mathbf{I}\}$ is false.
- The cramming module wants to recruit some extra hidden nodes to make LGT1 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\ c \in \mathbf{I}\}$ true.
- Method: With recruiting three extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.



The cramming module – the case of real-number inputs, binary-number desired output & ReLU

classification problems with LGT1

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.



The cramming module – the case of real-number inputs, binary-number desired output & ReLU

classification problems with LGT1

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$.

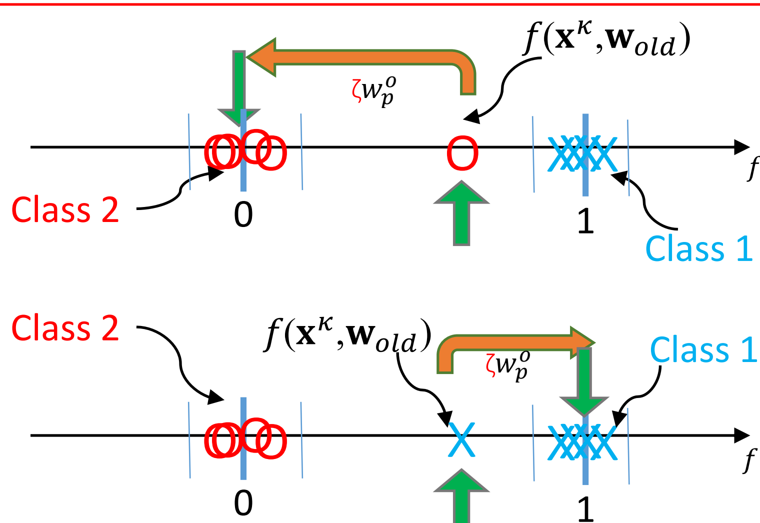
Step 2: Let $p+3 \rightarrow p$, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way to make the LGT1 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\ c \in \mathbf{I}\}$ true:

- $\mathbf{w}^H_{p-2} = \gamma$, $w^H_{p-2,0} = \zeta-\gamma^T\mathbf{x}^\kappa$, $w^o_{p-2} = \frac{y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i}{\zeta}$
- $\mathbf{w}^H_{p-1} = \gamma$, $w^H_{p-1,0} = -\gamma^T\mathbf{x}^\kappa$, $w^o_{p-1} = \frac{-2(y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i)}{\zeta}$
- $\mathbf{w}^H_{p} = \gamma$, $w^H_{p,0} = -\zeta-\gamma^T\mathbf{x}^\kappa$, $w^o_{p} = \frac{y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i}{\zeta}$

The magnitudes of w^o_p, w^o_{p-1} and w^o_{p-2} become larger when ζ is smaller. Therefore, the isolation module_R2 is recommended.

$\kappa \in \mathbf{I}_2$

$\kappa \in \mathbf{I}_1$



- $f(\mathbf{x}^\kappa, \mathbf{w}_{old}) = w^o_0 + \sum_{i=1}^{p-3} w^o_i a^\kappa_i$
- $f(\mathbf{x}^c, \mathbf{w}_{new}) = f(\mathbf{x}^c, \mathbf{w}_{old}) + w^o_p$*[ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)+\zeta$) - 2ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)$) + ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)-\zeta$)]
- $\Delta f(\mathbf{x}^\kappa) = \zeta w^o_p$ and $\Delta f(\mathbf{x}^c) = 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$

Only the output of κth input is changed into the right value, while outputs of other inputs are still the same.

A rule-based mechanism

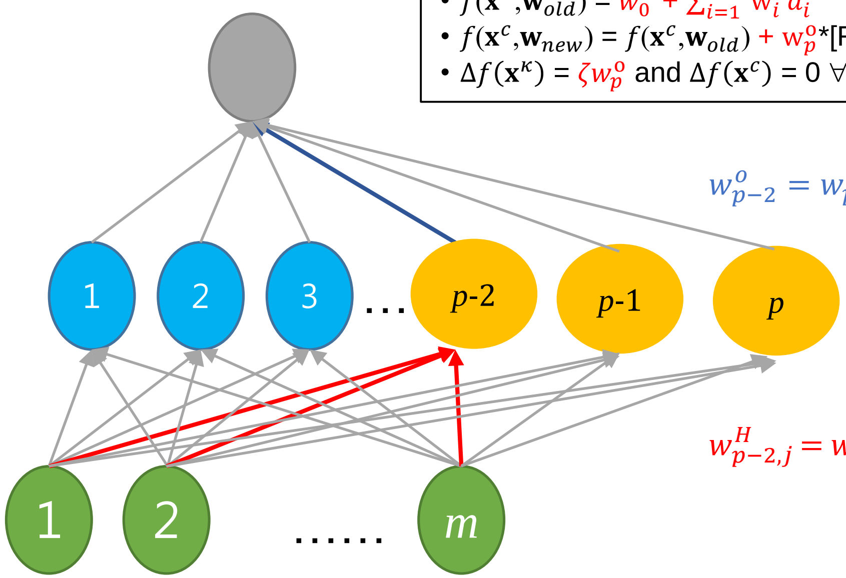
The cramming module_ReLU_RI_SO_LGT1_SU

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall\ c \in \mathrm{I}-\{\kappa\}$ AND $(\zeta+\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall\ c \in \mathrm{I}-\{\kappa\}$.

Step 2: Let p+3 → p, add three new hidden nodes p-2[th], p-1[th] and p[th] to the existing SLFN, and then assign their associated weights in the following way to make to make the LGT1 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\ c \in \mathbf{I}\}$ true:

- $\mathbf{w}_{p-2}^H = \boldsymbol{\gamma}$, $w_{p-2,0}^H = \zeta-\gamma^T\mathbf{x}^\kappa$, $w_{p-2}^o = \frac{y^\kappa - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^\kappa}{\zeta}$
- $\mathbf{w}_{p-1}^H = \boldsymbol{\gamma}$, $w_{p-1,0}^H = -\gamma^T\mathbf{x}^\kappa$, $w_{p-1}^o = \frac{-2(y^\kappa - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^\kappa)}{\zeta}$
- $\mathbf{w}_{p}^H = \boldsymbol{\gamma}$, $w_{p,0}^H = -\zeta-\gamma^T\mathbf{x}^\kappa$, $w_{p}^o = \frac{y^\kappa - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^\kappa}{\zeta}$

- $f(\mathbf{x}^\kappa, \mathbf{w}_{old}) = w_0^o + \sum_{i=1}^{p-3} w_i^o a_i^\kappa$
- $f(\mathbf{x}^c, \mathbf{w}_{new}) = f(\mathbf{x}^c, \mathbf{w}_{old}) + w_p^o$*[ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)+\zeta$) - 2ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)$) + ReLU($\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)-\zeta$)]
- $\Delta f(\mathbf{x}^\kappa) = \zeta w_p^o$ and $\Delta f(\mathbf{x}^c) = 0\ \forall\ c \in \mathrm{I}-\{\kappa\}$



$w_{p-2}^o = w_p^o = \frac{y^\kappa - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^\kappa}{\zeta}$; $w_{p-1}^o = \frac{-2(y^\kappa - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^\kappa)}{\zeta}$

$w_{p-2,0}^H = \zeta-\gamma^T\mathbf{x}^\kappa$; $w_{p-1,0}^H = -\gamma^T\mathbf{x}^\kappa$; $w_{p,0}^H = -\zeta-\gamma^T\mathbf{x}^\kappa$

$w_{p-2,j}^H = w_{p-1,j}^H = w_{p,j}^H = \gamma_j \quad \forall\ j$

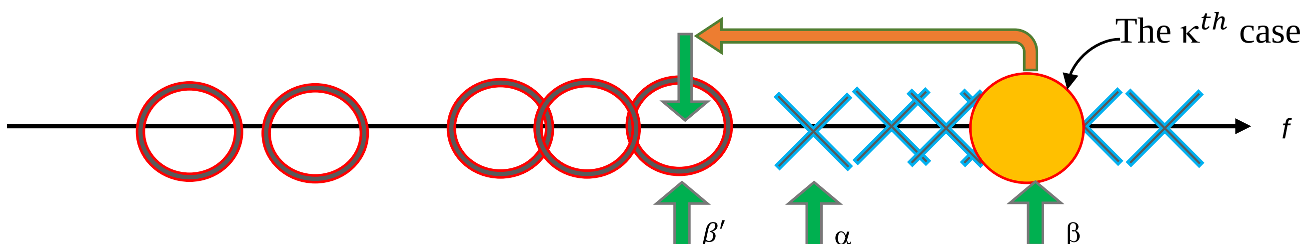
The cramming module – the case of real-number inputs, binary-number desired output & ReLU

classification problems with LGT3

- Assume $\mathbf{x} \in \mathrm{R}^m$; $f \in \mathrm{R}$; $y^c = 1$ if $c \in \mathbf{I}_1$; $y^c = 0$ if $c \in \mathbf{I}_2$
- Assume the current SLFN makes LGT3 (i.e., LSC) regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\, c \in \mathbf{I}\text{-}\{\kappa\}\}$ true, but LGT3 regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\, c \in \mathbf{I}\}$ is false.
- Regarding the case of real-number inputs, binary-number desired output and ReLU, the cramming module recruits some extra hidden nodes to make LSC regarding $\{f(\mathbf{x}^c, \mathbf{w})\ \forall\, c \in \mathbf{I}\}$ true.
- Method: With recruiting three extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.

$\kappa \in \mathbf{I}_2$

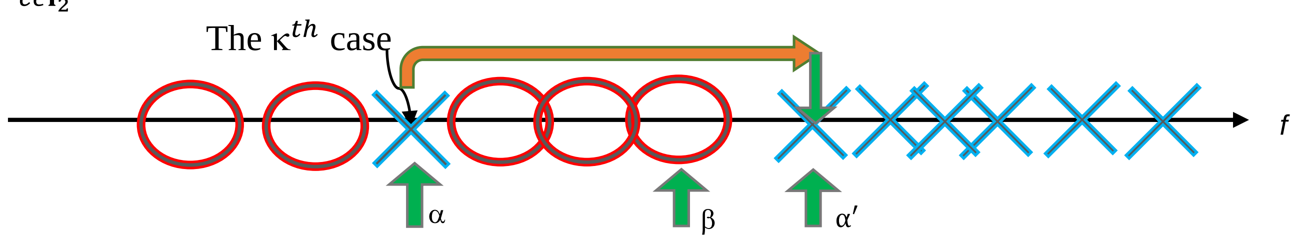
$$\beta' = \max_{c \in \mathbf{I}_2 - \{\kappa\}} f(\mathbf{x}^c, \mathbf{w}_{old})$$



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

$\kappa \in \mathbf{I}_1$

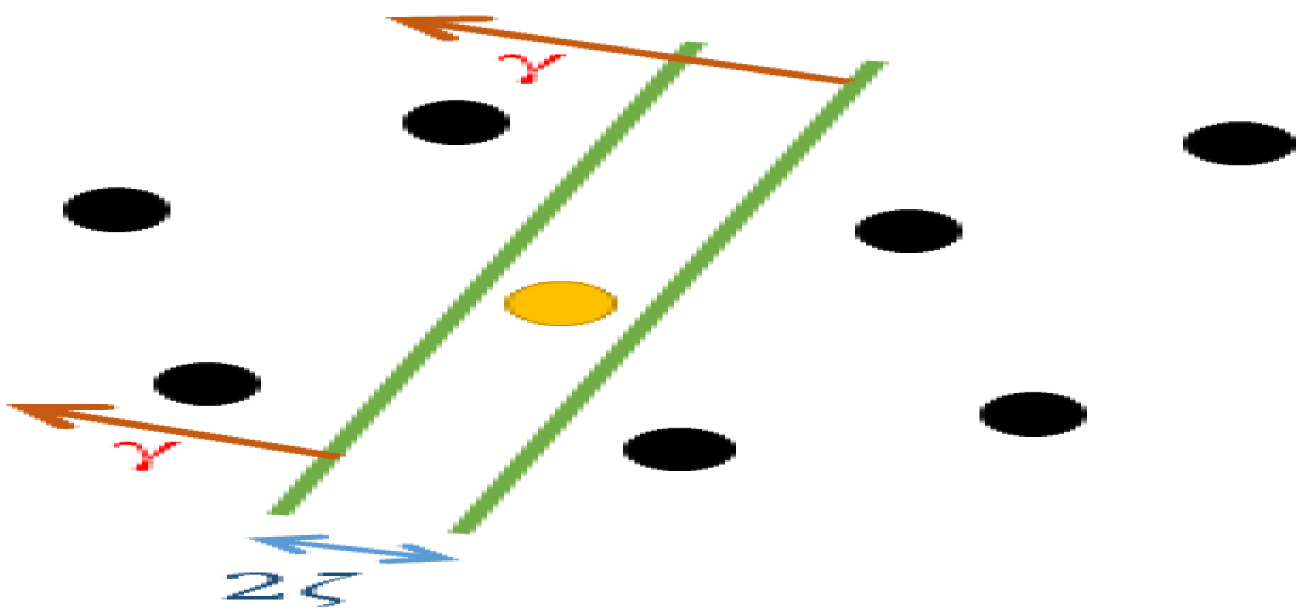
$$\alpha' = \min_{c \in \mathbf{I}_1 - \{\kappa\}} f(\mathbf{x}^c, \mathbf{w}_{old})$$



The cramming module – the case of real-number inputs, binary-number desired output & ReLU

classification problems with LGT3

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.



The cramming module – the case of real-number inputs, binary-number desired output & ReLU

classification problems with LGT3

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}) \neq 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$.

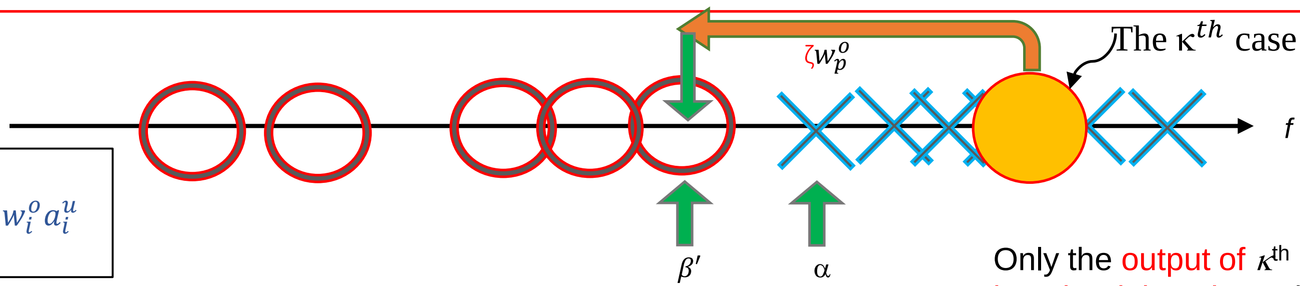
Step 2: Let p+3 → p, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way to make the LSC regarding $\{f(\mathbf{x}^{c},\mathbf{w})\ \forall\ c \in \mathbf{I}\}$ true:

- $\mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \gamma$
- $w^{H}_{p-2,0} = \zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p-1,0} = -\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p,0} = -\zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$
- $$w^{o}_{p-2} = w^{o}_{p} = \begin{cases} \dfrac{\max_{u\in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^\kappa}{\zeta} & if\ \kappa \in \mathbf{I}_2 \\ \dfrac{\min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v - \sum_{i=1}^{p-3} w_i^o a_i^\kappa}{\zeta} & if\ \kappa \in \mathbf{I}_1 \end{cases};\ w^{o}_{p-1} = \begin{cases} \dfrac{-2(\max_{u\in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^\kappa)}{\zeta} & if\ \kappa \in \mathbf{I}_2 \\ \dfrac{-2(\min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v - \sum_{i=1}^{p-3} w_i^o a_i^\kappa)}{\zeta} & if\ \kappa \in \mathbf{I}_1 \end{cases}$$

- $f(\mathbf{x}^{\kappa},\mathbf{w}_{old}) = w_0^o + \sum_{i=1}^{p-3} w_i^o a_i^\kappa$
- $f(\mathbf{x}^{c},\mathbf{w}_{new}) = f(\mathbf{x}^{c},\mathbf{w}_{old}) + w_p^o*[\mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})+\zeta) - 2\mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) + \mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})-\zeta)]$
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w_p^o$ and $\Delta f(\mathbf{x}^{c}) = 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$

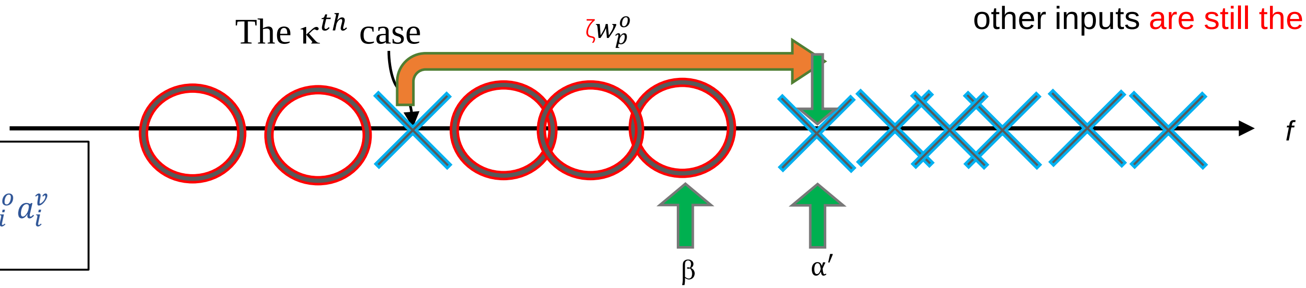
$\kappa \in \mathbf{I}_2$

$$\beta' = w_0^o + \max_{u\in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u$$



$\kappa \in \mathbf{I}_1$

$$\alpha' = w_0^o + \min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v$$



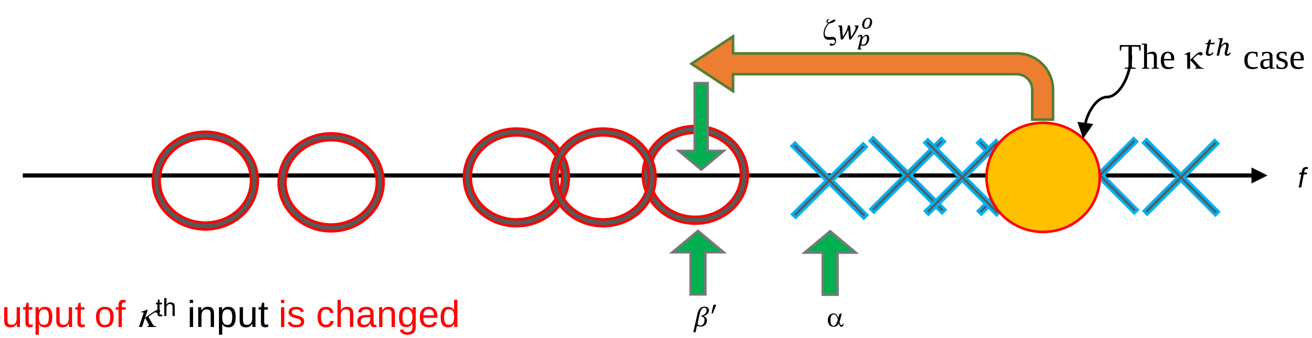
Only the output of κ^{th} input is changed into the right value, while outputs of other inputs are still the same.

Case of $\kappa \in \mathbf{I}_2$

- $\mathbf{w}_{p-2}^{H} = \boldsymbol{\gamma}$, $w_{p-2,0}^{H} = \zeta - \gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w_{p-2}^{o} = \dfrac{\max_{u \in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}}{\zeta}$
- $\mathbf{w}_{p-1}^{H} = \boldsymbol{\gamma}$, $w_{p-1,0}^{H} = -\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w_{p-1}^{o} = \dfrac{-2(\max_{u \in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa})}{\zeta}$
- $\mathbf{w}_{p}^{H} = \boldsymbol{\gamma}$, $w_{p0}^{H} = -\zeta - \gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w_{p}^{o} = \dfrac{\max_{u \in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}}{\zeta}$

- $f(\mathbf{x}^{\kappa}, \mathbf{w}_{old}) = w_0^o + \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}$
- $f(\mathbf{x}^{c}, \mathbf{w}_{new}) = f(\mathbf{x}^{c}, \mathbf{w}_{old}) + w_p^o * [\mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})+\zeta) - 2\mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) + \mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})-\zeta)]$
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w_p^o$ and $\Delta f(\mathbf{x}^{c}) = 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$

$\beta' = w_0^o + \max_{u \in \mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^u$



Only the output of κ^{th} input is changed into the right value, while outputs of other inputs are still the same.

Case of $\kappa \in \mathbf{I}_1$

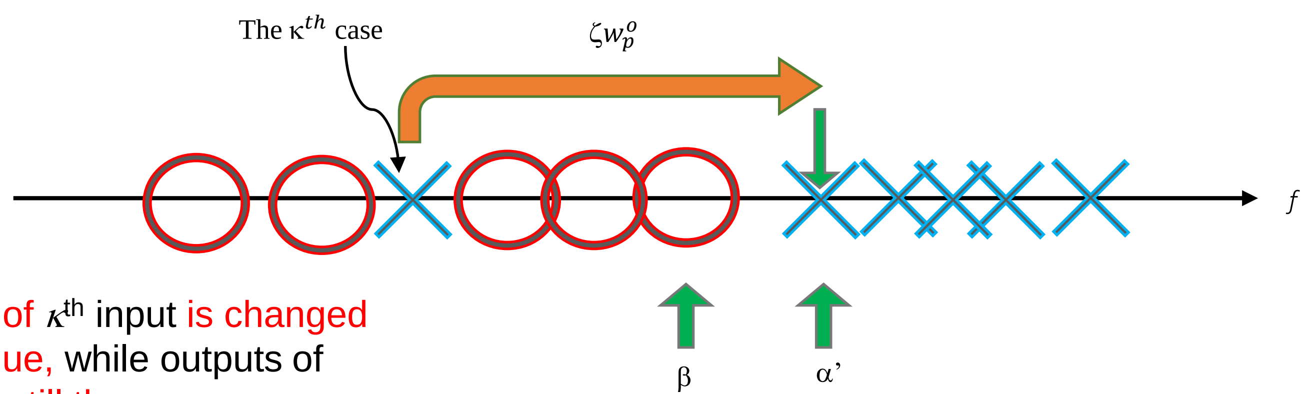
- $\mathbf{w}^H_{p-2} = \boldsymbol{\gamma}$, $w^H_{p-2,0} = \zeta - \gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^o_{p-2} = \dfrac{\min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}}{\zeta}$

- $\mathbf{w}^H_{p-1} = \boldsymbol{\gamma}$, $w^H_{p-1,0} = -\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^o_{p-1} = \dfrac{-2\left(\min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}\right)}{\zeta}$

- $\mathbf{w}^H_{p} = \boldsymbol{\gamma}$, $w^H_{p0} = -\zeta - \gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^o_{p} = \dfrac{\min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}}{\zeta}$

- $f(\mathbf{x}^{\kappa}, \mathbf{w}_{old}) = w_0^o + \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}$
- $f(\mathbf{x}^{c}, \mathbf{w}_{new}) = f(\mathbf{x}^{c}, \mathbf{w}_{old}) + w_p^o * [\mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^{\kappa}) + \zeta) - 2\mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^{\kappa})) + \mathrm{ReLU}(\gamma^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^{\kappa}) - \zeta)]$
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w_p^o$ and $\Delta f(\mathbf{x}^{c}) = 0 \ \forall\ c \in \mathbf{I} - \{\kappa\}$

$\alpha' = w_0^o + \min_{v\in \mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w_i^o a_i^v$



Only the output of κ^{th} input is changed into the right value, while outputs of other inputs are still the same.

A rule-based mechanism

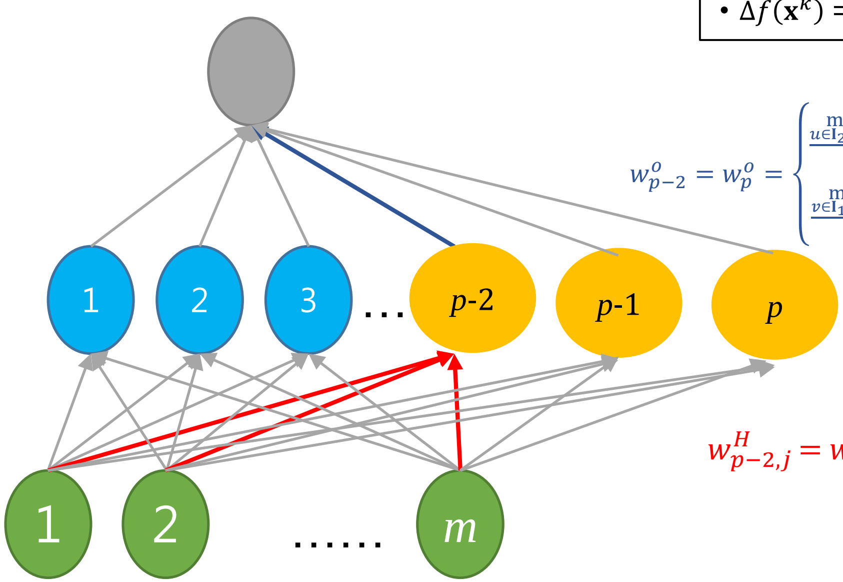
The cramming module_ReLU_RI_SO_LGT3_SU

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^{T}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}) \neq 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^{T}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\gamma^{T}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$.

Step 2: Let p+3 → p, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way to make the LSC regarding $\{f(\mathbf{x}^{c},\mathbf{w})\ \forall\ c \in \mathbf{I}\}$ true:

- $\mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \gamma$
- $w^{H}_{p-2,0} = \zeta-\gamma^{T}\mathbf{x}^{\kappa}$, $w^{H}_{p-1,0} = -\gamma^{T}\mathbf{x}^{\kappa}$, $w^{H}_{p,0} = -\zeta-\gamma^{T}\mathbf{x}^{\kappa}$
- $w^{o}_{p-2} = w^{o}_{p} = \begin{cases} \dfrac{\max_{u\in\mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{u}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta} & if\ \kappa \in \mathbf{I}_2 \\ \dfrac{\min_{v\in\mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{v}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta} & if\ \kappa \in \mathbf{I}_1 \end{cases}$; $w^{o}_{p-1} = \begin{cases} \dfrac{-2(\max_{u\in\mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{u}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta} & if\ \kappa \in \mathbf{I}_2 \\ \dfrac{-2(\min_{v\in\mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{v}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta} & if\ \kappa \in \mathbf{I}_1 \end{cases}$

- $f(\mathbf{x}^{\kappa},\mathbf{w}_{old}) = w^{o}_{0} + \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}$
- $f(\mathbf{x}^{c},\mathbf{w}_{new}) = f(\mathbf{x}^{c},\mathbf{w}_{old}) + w^{o}_{p}$*[ReLU$(\gamma^{T}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})+\zeta)$ - 2ReLU$(\gamma^{T}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))$ + ReLU$(\gamma^{T}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})-\zeta)$]
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w^{o}_{p}$ and $\Delta f(\mathbf{x}^{c}) = 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$



$$w^{o}_{p-2} = w^{o}_{p} = \begin{cases} \dfrac{\max_{u\in\mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{u}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta} & if\ \kappa \in \mathbf{I}_2 \\ \dfrac{\min_{v\in\mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{v}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta} & if\ \kappa \in \mathbf{I}_1 \end{cases}; w^{o}_{p-1} = \begin{cases} \dfrac{-2(\max_{u\in\mathbf{I}_2-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{u}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta} & if\ \kappa \in \mathbf{I}_2 \\ \dfrac{-2(\min_{v\in\mathbf{I}_1-\{\kappa\}} \sum_{i=1}^{p-3} w^{o}_{i} a^{v}_{i} - \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta} & if\ \kappa \in \mathbf{I}_1 \end{cases}$$

$$w^{H}_{p-2,0} = \zeta-\gamma^{T}\mathbf{x}^{\kappa};\ w^{H}_{p-1,0} = -\gamma^{T}\mathbf{x}^{\kappa};\ w^{H}_{p,0} = -\zeta-\gamma^{T}\mathbf{x}^{\kappa}$$

$$w^{H}_{p-2,j} = w^{H}_{p-1,j} = w^{H}_{p,j} = \gamma_j\quad \forall\ j$$

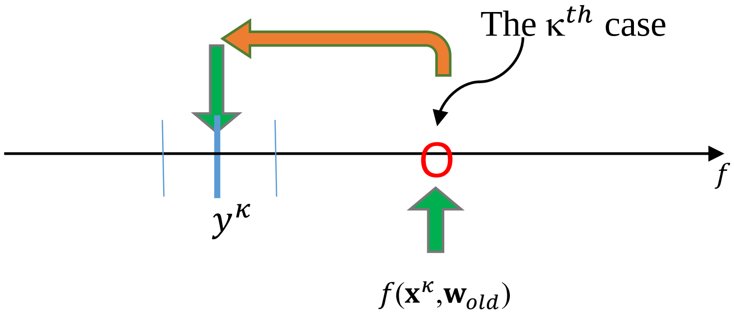
Real-number inputs

Regression applications

The cramming module – the case of real-number inputs, real-number desired output & ReLU

regression problems

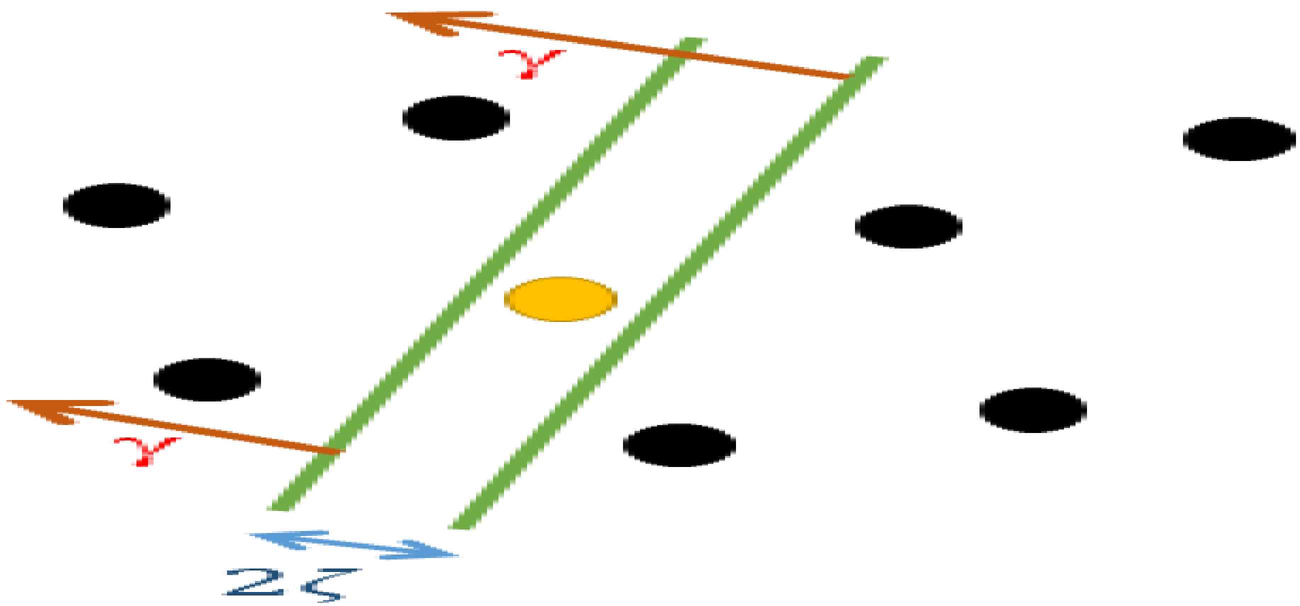
- Assume $\mathbf{x} \in \mathrm{R}^m$; $f \in \mathrm{R}$; $y^c \in \mathrm{R}$
- Assume the current SLFN makes $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ true, but $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false.
- Regarding the case of real-number inputs, real-number desired output and ReLU, the cramming module recruits some hidden nodes to make $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}$ true.
- Method: With recruiting three extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.



The cramming module – the case of real-number inputs, real-number desired output & ReLU

regression problems

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.



The cramming module – the case of real-number inputs, real-number desired output & ReLU

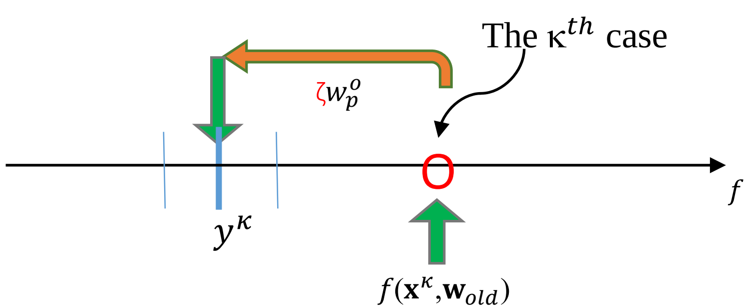
regression problems

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}) \neq 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$.

Step 2: Let $p+3 \rightarrow p$, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way to make $(f(\mathbf{x}^{c},\mathbf{w})-y^{c})^{2} \leq \varepsilon^{2} \ \forall\ c \in \mathbf{I}$ true :

- $\mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \boldsymbol{\gamma}$
- $w^{H}_{p-2,0} = \zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p-1,0} = -\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p,0} = -\zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$
- $w^{o}_{p-2} = w^{o}_{p} = \frac{y^{\kappa}-w^{o}_{0}-\sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta}$; $w^{o}_{p-1} = \frac{-2(y^{\kappa}-w^{o}_{0}-\sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta}$

- $f(\mathbf{x}^{\kappa},\mathbf{w}_{old}) = w^{o}_{0} + \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}$
- $f(\mathbf{x}^{c},\mathbf{w}_{new}) = f(\mathbf{x}^{c},\mathbf{w}_{old}) + w^{o}_{p}*[\mathrm{ReLU}(\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})+\zeta) - 2\mathrm{ReLU}(\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) + \mathrm{ReLU}(\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})-\zeta)]$
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w^{o}_{p}$ and $\Delta f(\mathbf{x}^{c}) = 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$



Only the output of κ^{th} input is changed into the right value, while outputs of other inputs are still the same.

A rule-based mechanism

The cramming module_ReLU_RI_SO_RE_SU

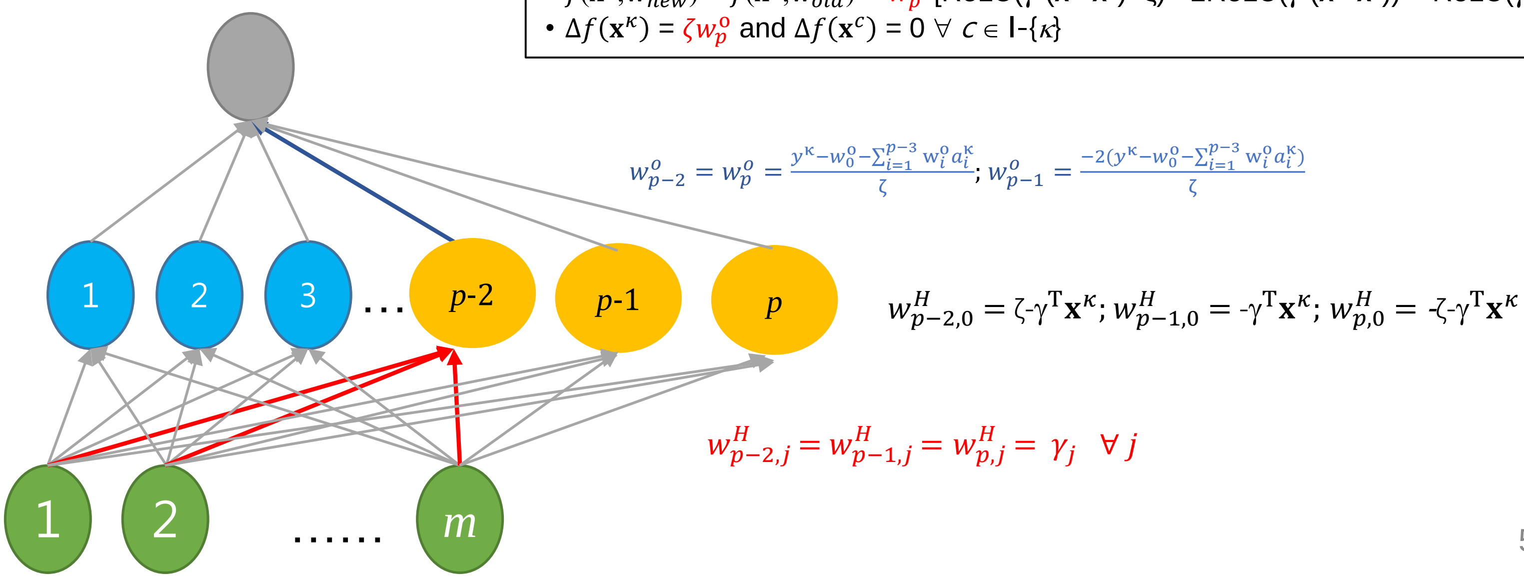
Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.

Step 2: Let $p+3 \rightarrow p$, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way to make $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \leq \varepsilon^2 \ \forall \ c \in \mathbf{I}$ true:

- $\mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \boldsymbol{\gamma}$
- $w^{H}_{p-2,0} = \zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p-1,0} = -\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p,0} = -\zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$
- $w^{o}_{p-2} = w^{o}_{p} = \frac{y^{\kappa}-w^{o}_{0}-\sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta}$; $w^{o}_{p-1} = \frac{-2(y^{\kappa}-w^{o}_{0}-\sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta}$

Note that the cramming module_ReLU_RI_SO_RE_SU is the same as the cramming module_ReLU_RI_SO_LGT1_SU.

- $f(\mathbf{x}^{\kappa},\mathbf{w}_{old}) = w^{o}_{0} + \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}$
- $f(\mathbf{x}^{c},\mathbf{w}_{new}) = f(\mathbf{x}^{c},\mathbf{w}_{old}) + w^{o}_{p}$*[ReLU($\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})+\zeta$) - 2ReLU($\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})$) + ReLU($\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})-\zeta$)]
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w^{o}_{p}$ and $\Delta f(\mathbf{x}^{c}) = 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$

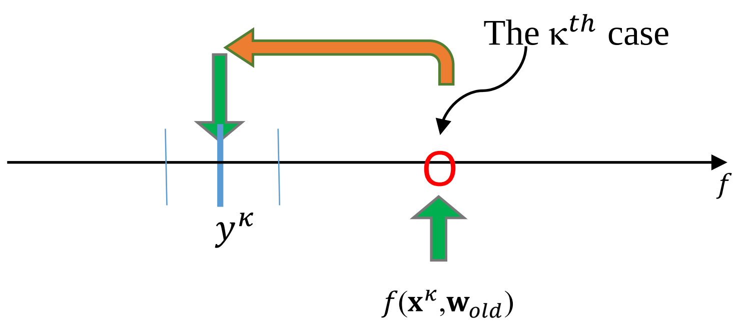


The cramming module for multiple unacceptable cases

The cramming module – the case of real-number inputs, real-number desired output & ReLU

regression problems

- Assume $\mathbf{x} \in \mathrm{R}^m$; $f \in \mathrm{R}$; $y^c \in \mathrm{R}$
- Assume the current SLFN makes $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \le \varepsilon^2 \;\forall\, c \in \mathbf{I}$-$\{\kappa_1, \kappa_2, \ldots\}$ true, but $(f(\mathbf{x}^c, \mathbf{w}) - y^c)^2 \le \varepsilon^2 \;\forall\, c \in \mathbf{I}$ is false.
- Method: For each unacceptable case $(\mathbf{x}^\kappa, y^\kappa)$, with recruiting three extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its output can be changed into the right value while outputs of other inputs are still the same.

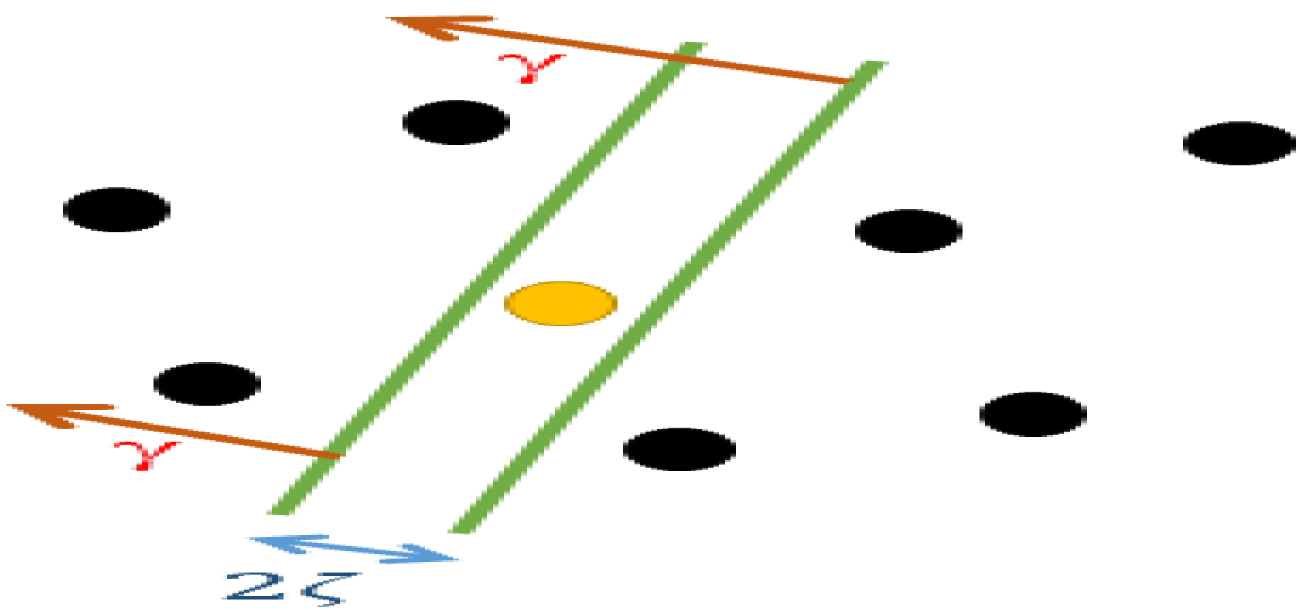


The cramming module – the case of real-number inputs, real-number desired output & ReLU

regression problems

For each unacceptable case ($\mathbf{x}^\kappa$, y^κ):

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0 \ \forall \ c \in \mathbf{I}-\{\kappa\}$.



The cramming module – the case of real-number inputs, real-number desired output & ReLU

regression problems

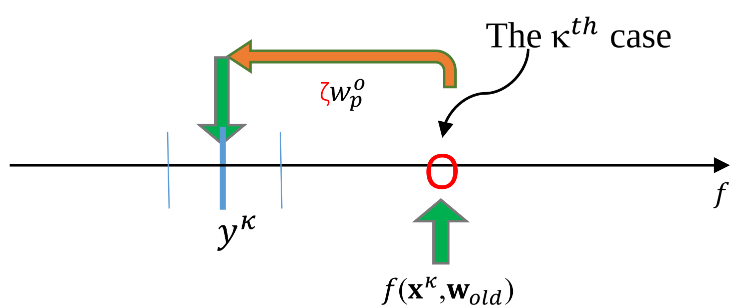
For each unacceptable case ($\mathbf{x}^{\kappa}$, y^{κ}):

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}) \neq 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa}))*(\zeta-\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})) < 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$.

Step 2: Let $p+3 \rightarrow p$, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way:

- $\mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \mathbf{w}^{H}_{p-2} = \boldsymbol{\gamma}$
- $w^{H}_{p-2,0} = \zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p-1,0} = -\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$, $w^{H}_{p,0} = -\zeta-\gamma^{\mathrm{T}}\mathbf{x}^{\kappa}$
- $w^{o}_{p-2} = w^{o}_{p} = \frac{y^{\kappa}-w^{o}_{0}-\sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}}{\zeta}$; $w^{o}_{p-1} = \frac{-2(y^{\kappa}-w^{o}_{0}-\sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i})}{\zeta}$

- $f(\mathbf{x}^{\kappa},\mathbf{w}_{old}) = w^{o}_{0} + \sum_{i=1}^{p-3} w^{o}_{i} a^{\kappa}_{i}$
- $f(\mathbf{x}^{c},\mathbf{w}_{new}) = f(\mathbf{x}^{c},\mathbf{w}_{old}) + w^{o}_{p}$*[ReLU($\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})+\zeta$) - 2ReLU($\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})$) + ReLU($\gamma^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{\kappa})-\zeta$)]
- $\Delta f(\mathbf{x}^{\kappa}) = \zeta w^{o}_{p}$ and $\Delta f(\mathbf{x}^{c}) = 0 \ \forall\ c \in \mathbf{I}-\{\kappa\}$



Only the output of κth input is changed into the right value, while outputs of other inputs are still the same.

The cramming module_ReLU_RI_SO_RE_MU

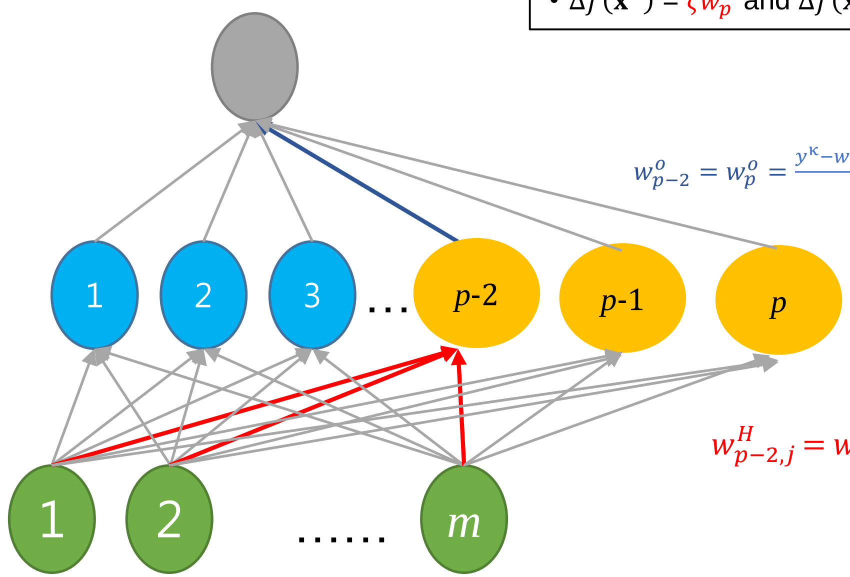
For each unacceptable case ($\mathbf{x}^\kappa$, y^κ):

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$ AND $(\zeta+\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$.

Step 2: Let $p+3 \rightarrow p$, add three new hidden nodes p-2th, p-1th and pth to the existing SLFN, and then assign their associated weights in the following way:

- $\mathbf{w}^H_{p-2} = \mathbf{w}^H_{p-2} = \mathbf{w}^H_{p-2} = \boldsymbol{\gamma}$
- $w^H_{p-2,0} = \zeta-\gamma^T\mathbf{x}^\kappa$, $w^H_{p-1,0} = -\gamma^T\mathbf{x}^\kappa$, $w^H_{p,0} = -\zeta-\gamma^T\mathbf{x}^\kappa$
- $w^o_{p-2} = w^o_p = \frac{y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i}{\zeta}$; $w^o_{p-1} = \frac{-2(y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i)}{\zeta}$

- $f(\mathbf{x}^\kappa,\mathbf{w}_{old}) = w^o_0 + \sum_{i=1}^{p-3} w^o_i a^\kappa_i$
- $f(\mathbf{x}^c,\mathbf{w}_{new}) = f(\mathbf{x}^c,\mathbf{w}_{old}) + w^o_p*[\mathrm{ReLU}(\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)+\zeta) - 2\mathrm{ReLU}(\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)) + \mathrm{ReLU}(\gamma^T(\mathbf{x}^c-\mathbf{x}^\kappa)-\zeta)]$
- $\Delta f(\mathbf{x}^\kappa) = \zeta w^o_p$ and $\Delta f(\mathbf{x}^c) = 0\ \forall\ c \in \mathbf{I}-\{\kappa\}$



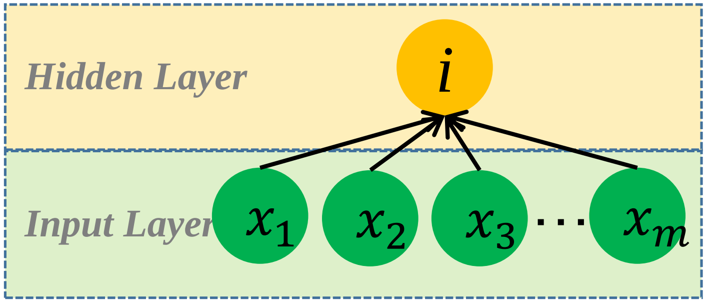
$w^o_{p-2} = w^o_p = \frac{y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i}{\zeta}$; $w^o_{p-1} = \frac{-2(y^\kappa - w^o_0 - \sum_{i=1}^{p-3} w^o_i a^\kappa_i)}{\zeta}$

$w^H_{p-2,0} = \zeta-\gamma^T\mathbf{x}^\kappa$; $w^H_{p-1,0} = -\gamma^T\mathbf{x}^\kappa$; $w^H_{p,0} = -\zeta-\gamma^T\mathbf{x}^\kappa$

$w^H_{p-2,j} = w^H_{p-1,j} = w^H_{p,j} = \gamma_j\ \ \forall\ j$

The cramming module for multiple output nodes

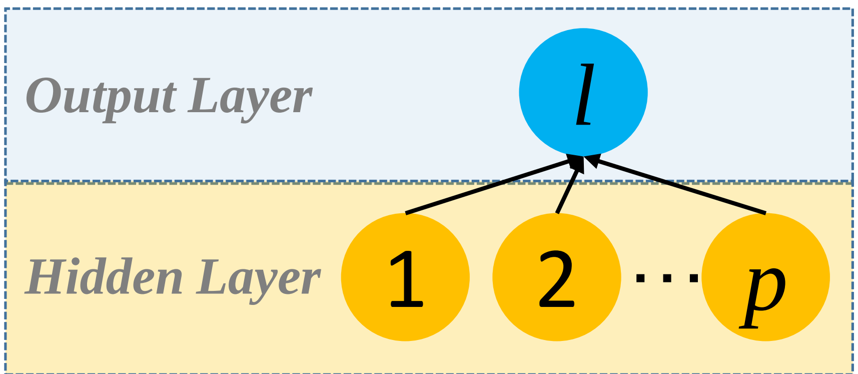
The forward operation SLFN with multiple output nodes



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:

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

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

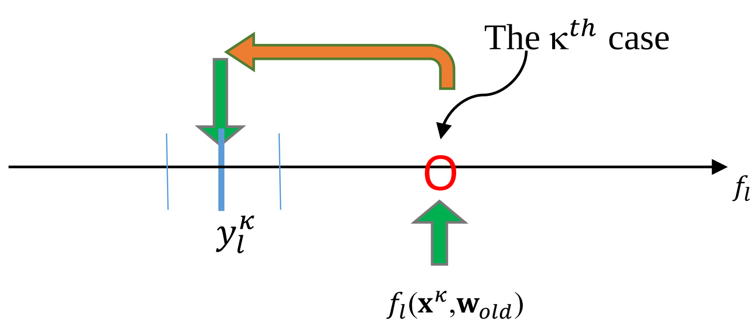
$E_N(\mathbf{w}) \equiv \frac{1}{N} \sum_{c \in \mathbf{I}} \sum_{l=1}^{q} (f_l(\mathbf{x}^c,\mathbf{w}) - y_l^c)^2$: the loss function;

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

The cramming module – the case of real-number inputs, real-number desired outputs & ReLU

regression problems

- Assume $\mathbf{x} \in \mathrm{R}^m$; $f \in \mathrm{R}^q$; $y^c \in \mathrm{R}^q$
- Assume the current SLFN makes $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}\ \forall\ l$ true, but $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}\ \forall\ l$ is false.
- Method: Regarding every l^{th} output node, in which $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(f_l(\mathbf{x}^c,\mathbf{w})-y_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false, with recruiting three extra hidden nodes, the κ^{th} input is isolated from all other inputs so that its (wrong) output can be changed into the right value while outputs of other inputs are still the same.



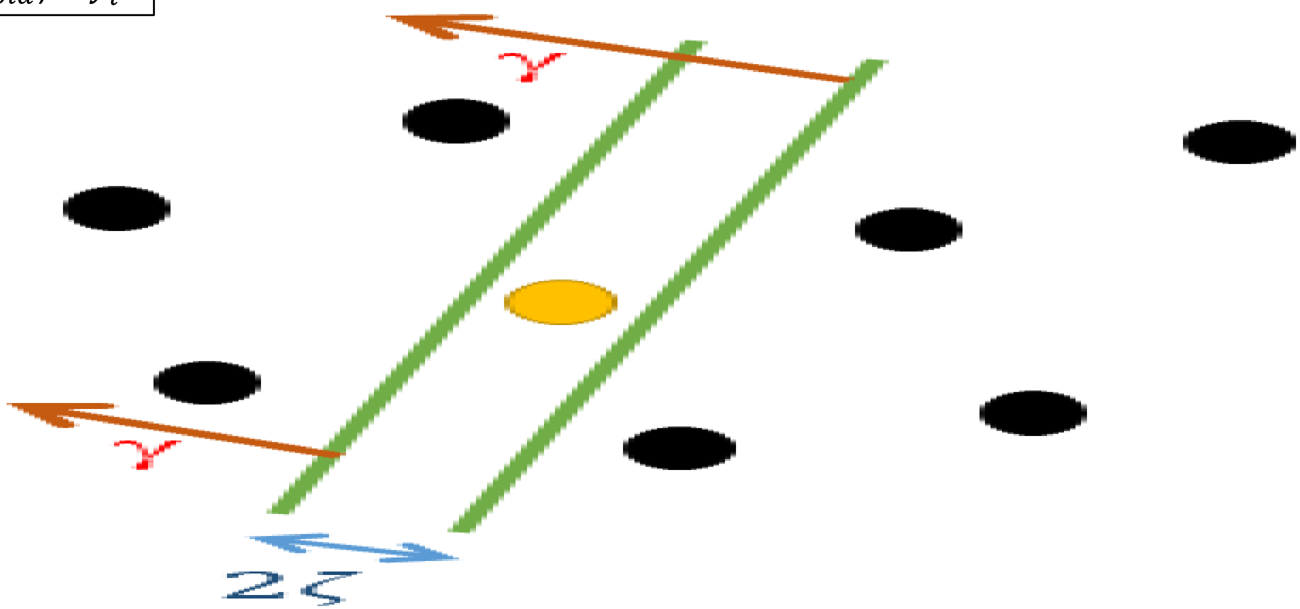
The cramming module – the case of real-number inputs, real-number desired outputs & ReLU

regression problems

Regarding every l^{th} output node, in which $(e_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(e_l^c)^2 \leq \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false:

Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa) \neq 0\ \forall\ c \in \mathrm{I}\text{-}\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa))*(\zeta\text{-}\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa)) < 0\ \forall\ c \in \mathrm{I}\text{-}\{\kappa\}$.

$e_l^c \equiv f_l(\mathbf{x}^c, \mathbf{w}_{old}) - y_l^c$



The cramming module – the case of real-number inputs, real-number desired outputs & ReLU

regression problems

Regarding every l^{th} output node, in which $(e_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(e_l^c)^2 \le \varepsilon^2\ \forall\ c \in \mathbf{I}$ is false:

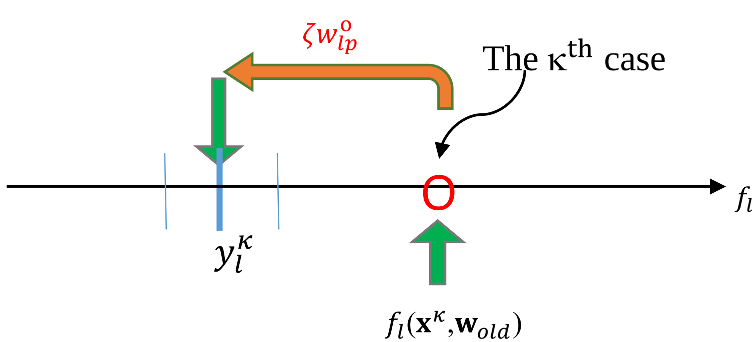
Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa) \neq 0\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$ AND $(\zeta+\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa))*(\zeta-\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)) < 0\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$.

Step 2: Let $p+3 \rightarrow p$, add three new hidden nodes p-2th, p-1th and p^{th} to the existing SLFN, and then assign their associated weights in the following way:

- $\mathbf{w}^H_{p-2} = \mathbf{w}^H_{p-2} = \mathbf{w}^H_{p-2} = \boldsymbol{\gamma}$
- $w^H_{p-2,0} = \zeta\text{-}\gamma^T\mathbf{x}^\kappa$, $w^H_{p-1,0} = \text{-}\gamma^T\mathbf{x}^\kappa$, $w^H_{p,0} = \text{-}\zeta\text{-}\gamma^T\mathbf{x}^\kappa$
- $w^o_{l,p-2} = w^o_{lp} = \frac{y_l^\kappa - w^o_{l0} - \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li}}{\zeta}$; $w^o_{l,p-1} = \frac{-2(y_l^\kappa - w^o_{l0} - \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li})}{\zeta}$; and $w^o_{kp} = 0\ \forall\ k \neq l$

$e_l^c \equiv f_l(\mathbf{x}^c, \mathbf{w}_{old})$ - y_l^c

- $f_l(\mathbf{x}^\kappa, \mathbf{w}_{old}) = w^o_{l0} + \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li}$
- $f_l(\mathbf{x}^c, \mathbf{w}_{new}) = f_l(\mathbf{x}^c, \mathbf{w}_{old}) + w^o_{lp}$*[ReLU($\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)+\zeta$) - 2ReLU($\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)$) + ReLU($\boldsymbol{\gamma}^T(\mathbf{x}^c-\mathbf{x}^\kappa)-\zeta$)]
- $\Delta f(\mathbf{x}^\kappa) = \zeta w^o_{lp}$ and $\Delta f(\mathbf{x}^c) = 0\ \forall\ c \in \mathbf{I}\text{-}\{\kappa\}$



Only the output of κ^{th} input is changed into the right value, while outputs of other inputs are still the same.

A rule-based mechanism

The cramming module_ReLU_RI_MO_RE_SU

Regarding every l^{th} output node, in which $(e_l^c)^2 \le \varepsilon^2\ \forall\, c \in \mathbf{I}\text{-}\{\kappa\}$ is true, but $(e_l^c)^2 \le \varepsilon^2\ \forall\, c \in \mathbf{I}$ is false:

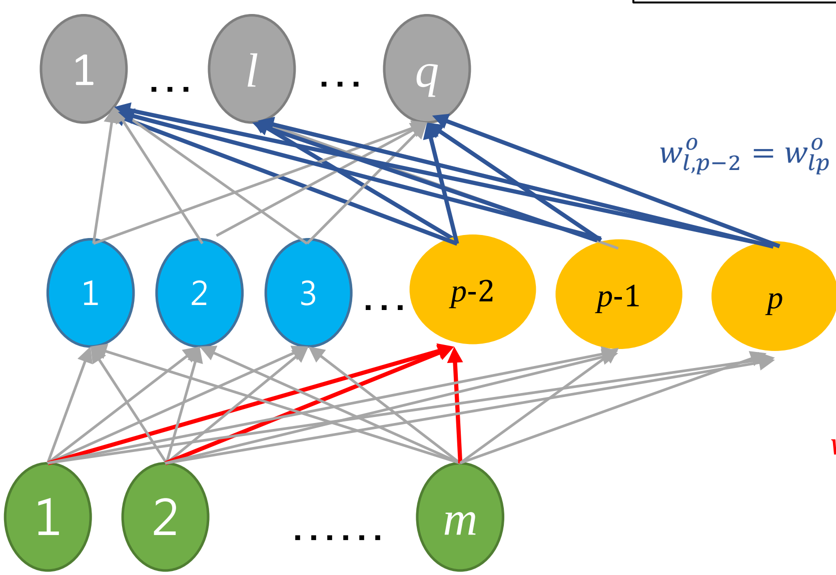
Step 1: Pick up a tiny number ζ and then use the random-number generation method to create an m-vector γ of length one such that $\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa) \neq 0\ \forall\, c \in \mathbf{I}\text{-}\{\kappa\}$ AND $(\zeta+\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa))*(\zeta\text{-}\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa)) < 0\ \forall\, c \in \mathbf{I}\text{-}\{\kappa\}$.

Step 2: Let p+3 → p, add three new hidden nodes p-2$^{\text{th}}$, p-1$^{\text{th}}$ and p^{th} to the existing SLFN, and then assign their associated weights in the following way:

- $\mathbf{w}^H_{p-2} = \mathbf{w}^H_{p-2} = \mathbf{w}^H_{p-2} = \gamma$
- $w^H_{p-2,0} = \zeta\text{-}\gamma^T\mathbf{x}^\kappa$, $w^H_{p-1,0} = \text{-}\gamma^T\mathbf{x}^\kappa$, $w^H_{p,0} = \text{-}\zeta\text{-}\gamma^T\mathbf{x}^\kappa$
- $w^o_{l,p-2} = w^o_{lp} = \frac{y_l^\kappa - w^o_{l0} - \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li}}{\zeta}$; $w^o_{l,p-1} = \frac{-2(y_l^\kappa - w^o_{l0} - \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li})}{\zeta}$; and $w^o_{kp} = 0\ \forall\, k \neq l$

$e_l^c \equiv f_l(\mathbf{x}^c, \mathbf{w}_{old}) - y_l^c$

- $f_l(\mathbf{x}^\kappa, \mathbf{w}_{old}) = w^o_{l0} + \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li}$
- $f_l(\mathbf{x}^c, \mathbf{w}_{new}) = f_l(\mathbf{x}^c, \mathbf{w}_{old}) + w^o_{lp}$*[ReLU($\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa)+\zeta$) - 2ReLU($\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa)$) + ReLU($\gamma^T(\mathbf{x}^c\text{-}\mathbf{x}^\kappa)-\zeta$)]
- $\Delta f(\mathbf{x}^\kappa) = \zeta w^o_{lp}$ and $\Delta f(\mathbf{x}^c) = 0\ \forall\, c \in \mathbf{I}\text{-}\{\kappa\}$



$w^o_{l,p-2} = w^o_{lp} = \frac{y_l^\kappa - w^o_{l0} - \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li}}{\zeta}$; $w^o_{l,p-1} = \frac{-2(y_l^\kappa - w^o_{l0} - \sum_{i=1}^{p-3} w^o_{li} a^\kappa_{li})}{\zeta}$; and $w^o_{kp} = 0\ \forall\, k \neq l$

$w^H_{p-2,0} = \zeta\text{-}\gamma^T\mathbf{x}^\kappa$; $w^H_{p-1,0} = \text{-}\gamma^T\mathbf{x}^\kappa$; $w^H_{p,0} = \text{-}\zeta\text{-}\gamma^T\mathbf{x}^\kappa$

$w^H_{p-2,j} = w^H_{p-1,j} = w^H_{p,j} = \gamma_j\quad \forall\, j$

The cramming module for multiple output nodes and multiple unacceptable cases

The cramming module_ReLU_RI_MO_RE_MU

The cramming modules

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

- ✓The cramming module_ReLU_RI_SO_LGT1_SU
- ✓The cramming module_ReLU_RI_SO_LGT3_SU
- ✓The cramming module_ReLU_RI_SO_RE_SU
- ✓The cramming module_ReLU_RI_SO_RE_MU
- ✓The cramming module_ReLU_RI_SO_LGT1_MU
- ✓The cramming module_ReLU_RI_SO_LGT3_MU
- ✓The cramming module_ReLU_RI_MO_RE_SU
- ✓The cramming module_ReLU_RI_MO_LGT1_SU
- ✓The cramming module_ReLU_RI_MO_LGT3_SU
- ✓The cramming module_ReLU_RI_MO_RE_MU
- ✓The cramming module_ReLU_RI_MO_LGT1_MU
- ✓The cramming module_ReLU_RI_MO_LGT3_MU
- ✓Your creative idea

You may derive the red parts by yourself.

Algorithm development

(Algorithm - Wikipedia)

- Typical steps in the development of algorithms:
  - ✓ Problem definition
  - ✓ Development of a model
  - ✓ Specification of the algorithm
  - ✓ Designing an algorithm
  - ✓ Checking the correctness of the algorithm
  - ✓ Analysis of algorithm
  - ✓ Implementation of algorithm
  - ✓ Program testing
  - ✓ Documentation preparation

The new learning mechanism is designed to have a good performance (i.e., the effectiveness and the efficiency) in the inferencing phase of the AI application.

Checking the correctness of the new mechanism

- Cannot validate the new learning mechanism through the **mathematical proof**.
- To validate the new learning mechanism, you need to set up an AI application experiment with the real 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.

Program testing of the new mechanism

- To validate the new learning mechanism, you need to code it.
- What should I do if the code cannot be run, the learning process is weird, or the performance is unsatisfied? ← Debug! Debug! And Debug!

First: Debug each module/block through printing out some information associated with the module/block.

Second: Debug the consistency amongst several consecutive modules/blocks through printing out some data flow between these consecutive modules/blocks.

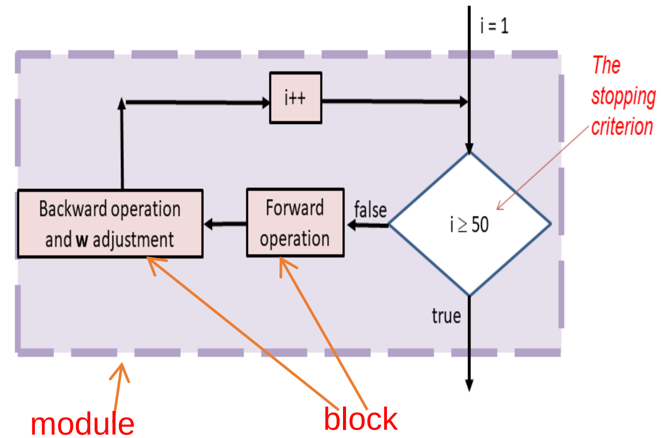
Third: Debug the logic of the whole mechanism through printing out some performance results.

TensorFlow: Loss

```
N, D, H = 64, 1000, 100

x = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
y = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
w1 = tf.Variable(tf.random.uniform((D, H)))  # weights
w2 = tf.Variable(tf.random.uniform((H, D)))  # weights

optimizer = tf.optimizers.SGD(1e-6)

for t in range(50):
  with tf.GradientTape() as tape:
    h = tf.maximum(tf.matmul(x, w1), 0)
    y_pred = tf.matmul(h, w2)
    diff = y_pred - y
    loss = tf.losses.MeanSquaredError()(y_pred, y)
  gradients = tape.gradient(loss, [w1, w2])
  optimizer.apply_gradients(zip(gradients, [w1, w2]))
```

Use predefined loss functions

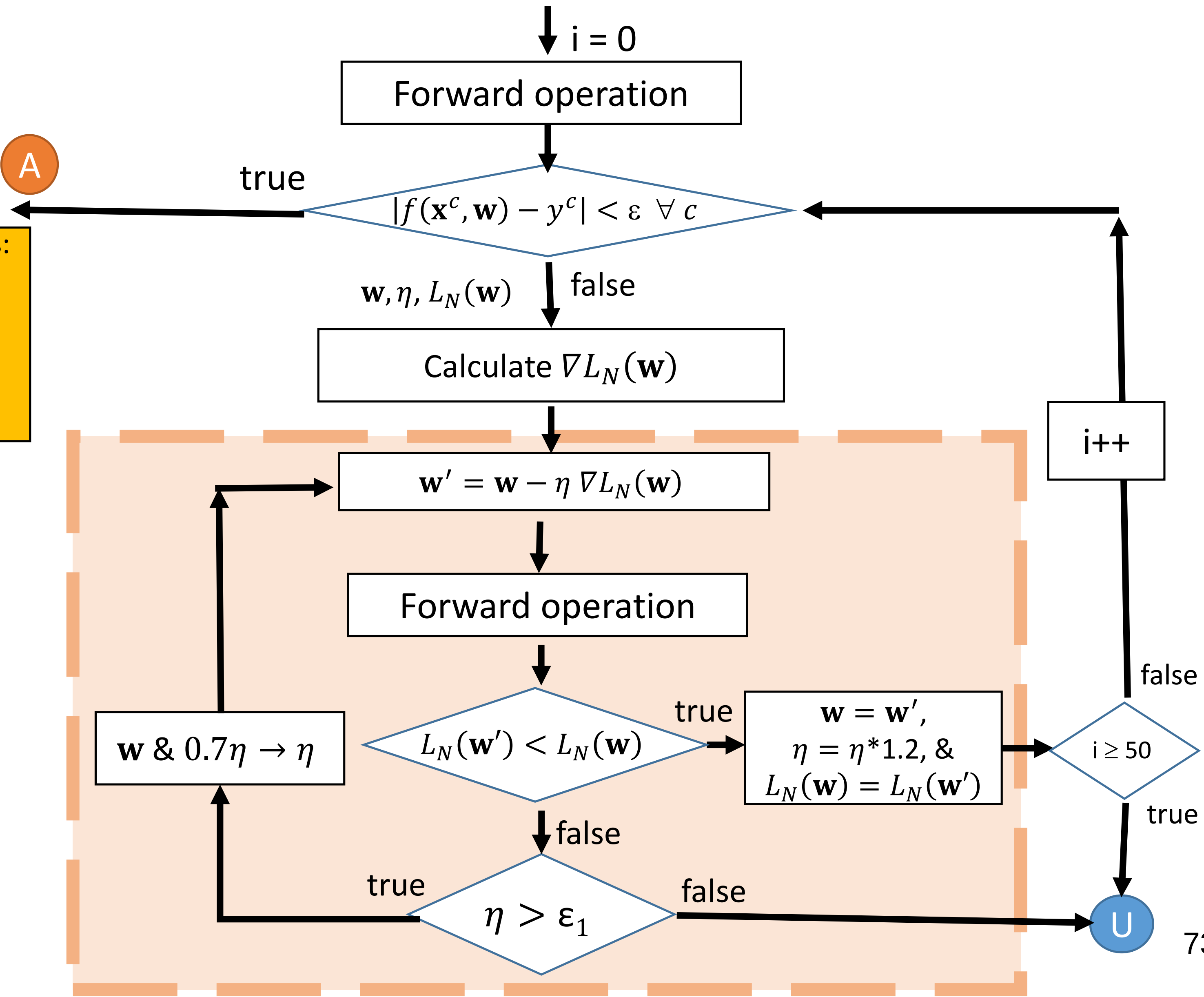
The flowchart form of algorithm



This is the program/code, not good for the algorithm.

Where we are now...

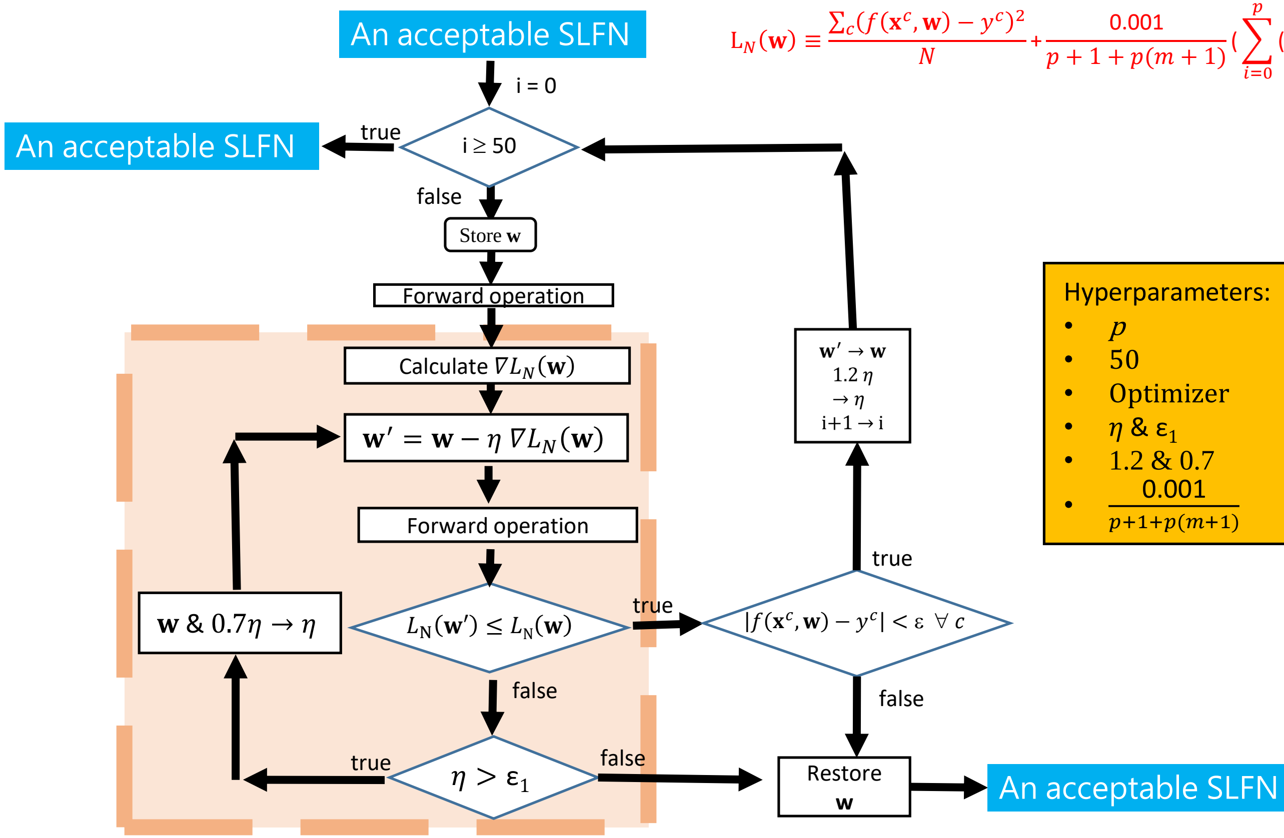
The weight-tuning module_EU_LG_UA



Hyperparameters:
- p
- Optimizer
- ε & ε_1
- 1.2 & 0.7
- 50

The regularizing module_EU_LG_UA

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



Hyperparameters:

- p
- 50
- Optimizer
- η & ε_1
- 1.2 & 0.7
- $\frac{0.001}{p+1+p(m+1)}$

Where we are now...

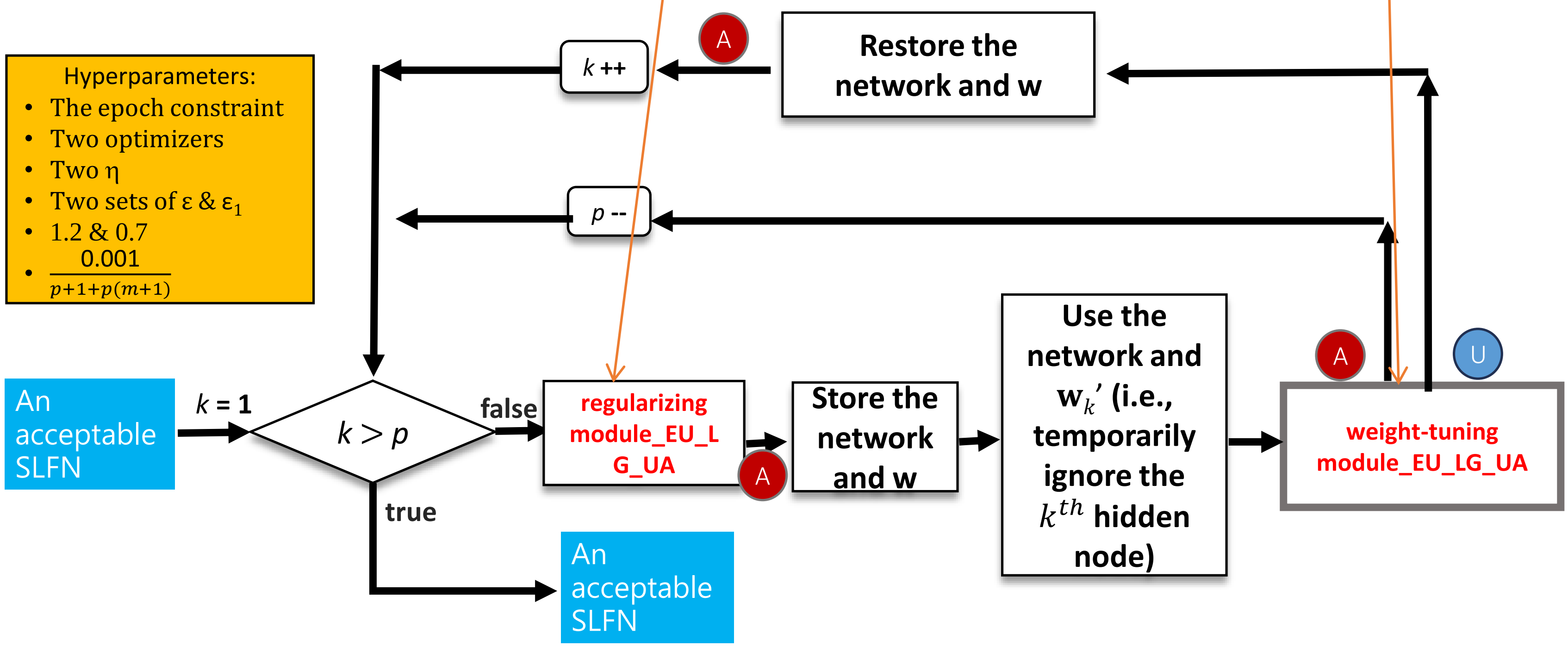
The reorganizing module_ALL_r_EU_LG_UA_w_EU_LG_UA

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

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

Another for the pruning purpose

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



Fix the mechanism with debugging

1. Debug each module/block through printing out some information associated with the module/block. ← Fix the bugs of each module/block or replace the module/block.
2. Debug the consistency amongst several consecutive modules/blocks through printing out some data flow between these consecutive modules/blocks. ← Fix the inconsistency via fine-tuning or replacing some modules/blocks.
3. Debug the logic of the whole mechanism through printing out some results. ← Fix the logic via fine-tuning some modules/blocks or replacing them. This is the core of validating the new mechanism.