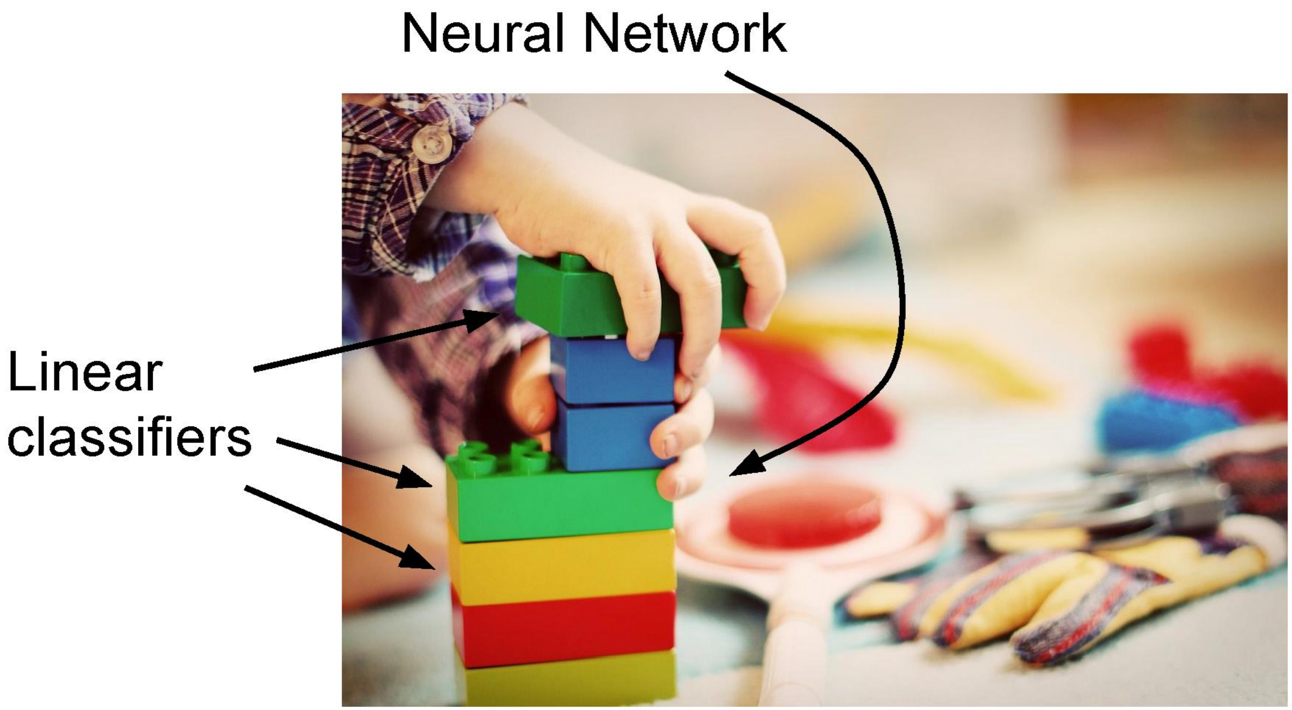
Learning goals and stopping criteria of learning mechanisms

國立政治大學 資訊管理學系
蔡瑞煌 特聘教授

Where we are now...

Developing a new AI system can be like playing with Lego – lots of modules



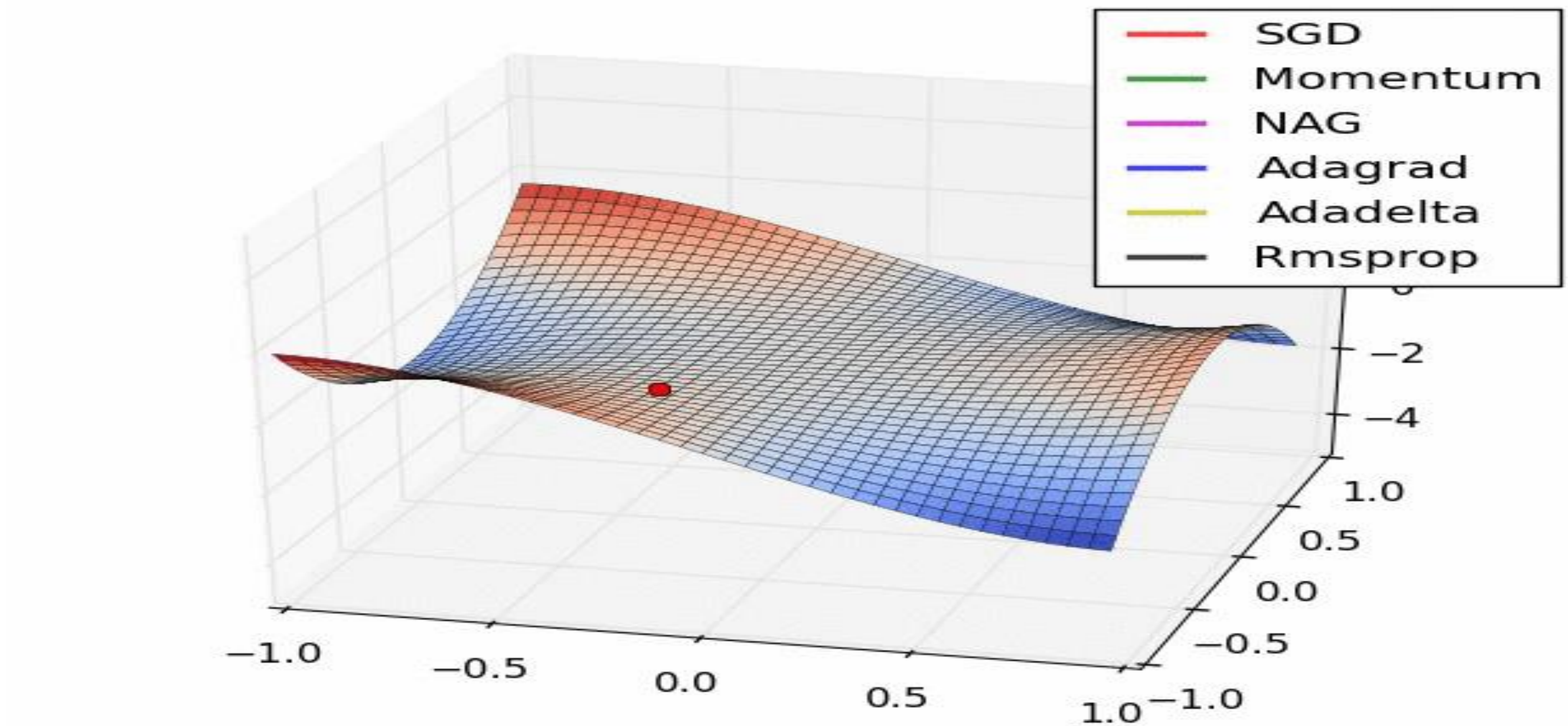
This image is CC0 1.0 public domain

Developing a new AI system is like playing with Lego – lots of tuning

Overview

1. **One time setup**
   activation functions, preprocessing, weight initialization, regularization, gradient checking
2. **Training dynamics**
   babysitting the learning process,
   parameter updates, hyperparameter optimization
3. **Evaluation**
   model ensembles, test-time augmentation, transfer learning

The learning process

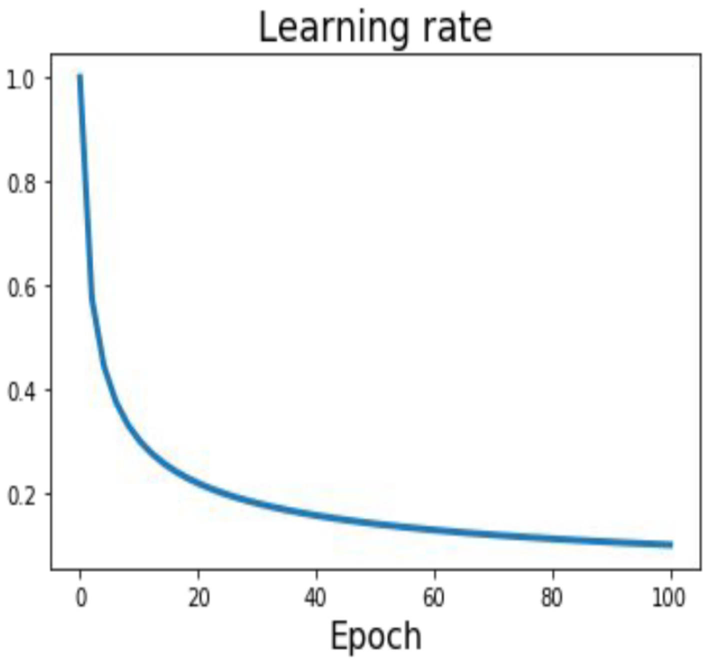


Reference:

1. https://en.wikipedia.org/wiki/Test_functions_for_optimization : Beale function
2. An overview of gradient descent optimization algorithms.pdf

Where we are now...

Learning Rate Decay



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos(t\pi/T)\right)$

Linear: $\alpha_t = \alpha_0(1 - t/T)$

Inverse sqrt: $\alpha_t = \alpha_0/\sqrt{t}$

α_0 : Initial learning rate
α_t : Learning rate at epoch t
T : Total number of epochs

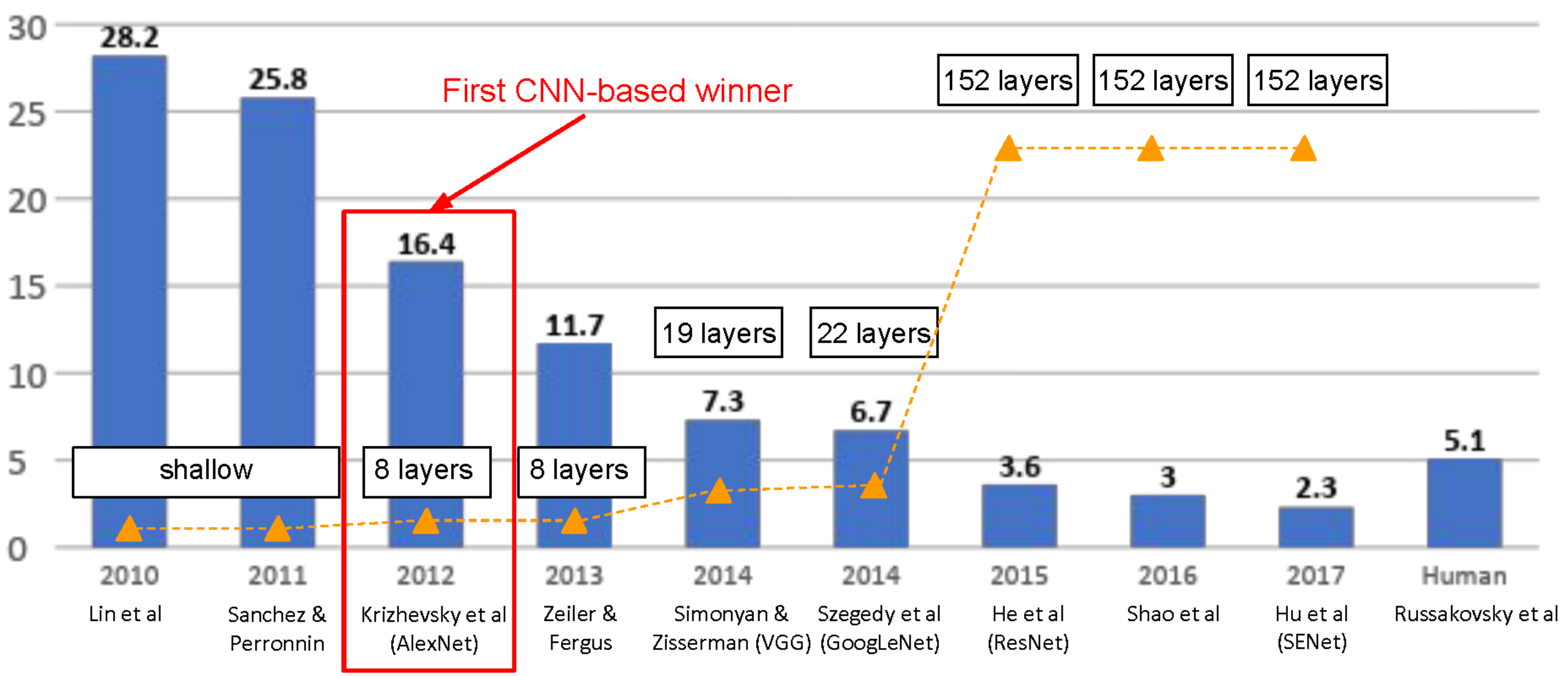
Vaswani et al, "Attention is all you need", NIPS 2017

AI applications

- Training phase: (training) data + AI model + algorithm & code + setting of network & hyperparameters → AI model/AI system
- Inferencing phase: performance is obtained from model((test) data)
- Goals of training are reasonable inferencing

CNN models

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Where we are now...

idea/concept of learning →
a learning algorithm →
codes →
an AI model/system

Where we are now…

Algorithm

(Algorithm - Wikipedia)

- In mathematics and computer science, an **algorithm** is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.
- Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, and other tasks.
- As an effective method, an algorithm can be expressed within a finite amount of space and time, and in a well-defined formal language for calculating a function.
- Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. ← Stopping criteria
- The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.

Where we are now...

Algorithm representation and development

(Algorithm - Wikipedia)

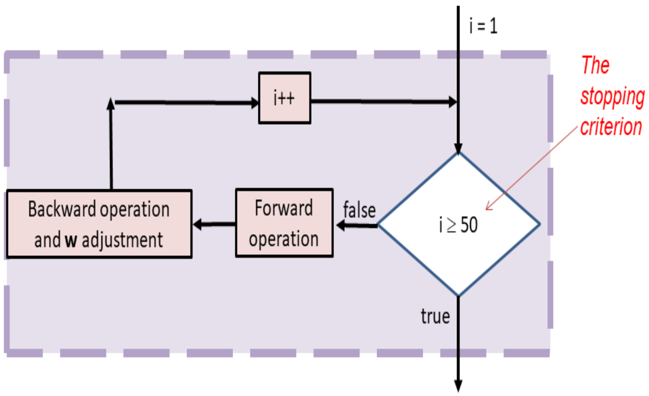
- Algorithms can be expressed in many kinds of notation, including natural languages, pseudocode, flowcharts, drakon-charts, programming languages or control tables (processed by interpreters).
  - ✓ Natural language expressions of algorithms tend to be verbose and ambiguous, and are rarely used for 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 ← The prediction problem
  - ✓ Development of a model ← A learning-based model
  - ✓ Specification of the algorithm ← The learning algorithm for 2-layer (or deep) neural networks
  - ✓ Designing an algorithm ← The gradient-descent-based learning algorithm
  - ✓ Checking the correctness of the algorithm ← The math proof of the proposed learning algorithm
  - ✓ Analysis of algorithm ← The amount of parameters, the (learning and inferencing) time scale, …
  - ✓ Implementation of algorithm ← The coding
  - ✓ Program testing
  - ✓ Documentation preparation

Where we are now...

TensorFlow: Loss

Use predefined loss functions

The flowchart form of algorithm




```
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]))
```

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

Where we are now...

The pseudocode form of Back Propagation learning algorithm

Step 0.1: Input all training data $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^N, y^N)\}$.

Step 0.2: Generate the initial values of $\mathbf{w}$.

Step 1: Execute the forward operation of SLFN regarding all training data.

Step 2: Based upon $f(\mathbf{x}^c, \mathbf{w})$ and y^c values, calculate the $L_N(\mathbf{w})$ value and store it.

Step 3: If $L_N(\mathbf{w})$ is less than the predetermined value (says, ε), then STOP.

Step 4: Calculate the gradient vector $\nabla_{\mathbf{w}} L_N(\mathbf{w})$ of SLFN.

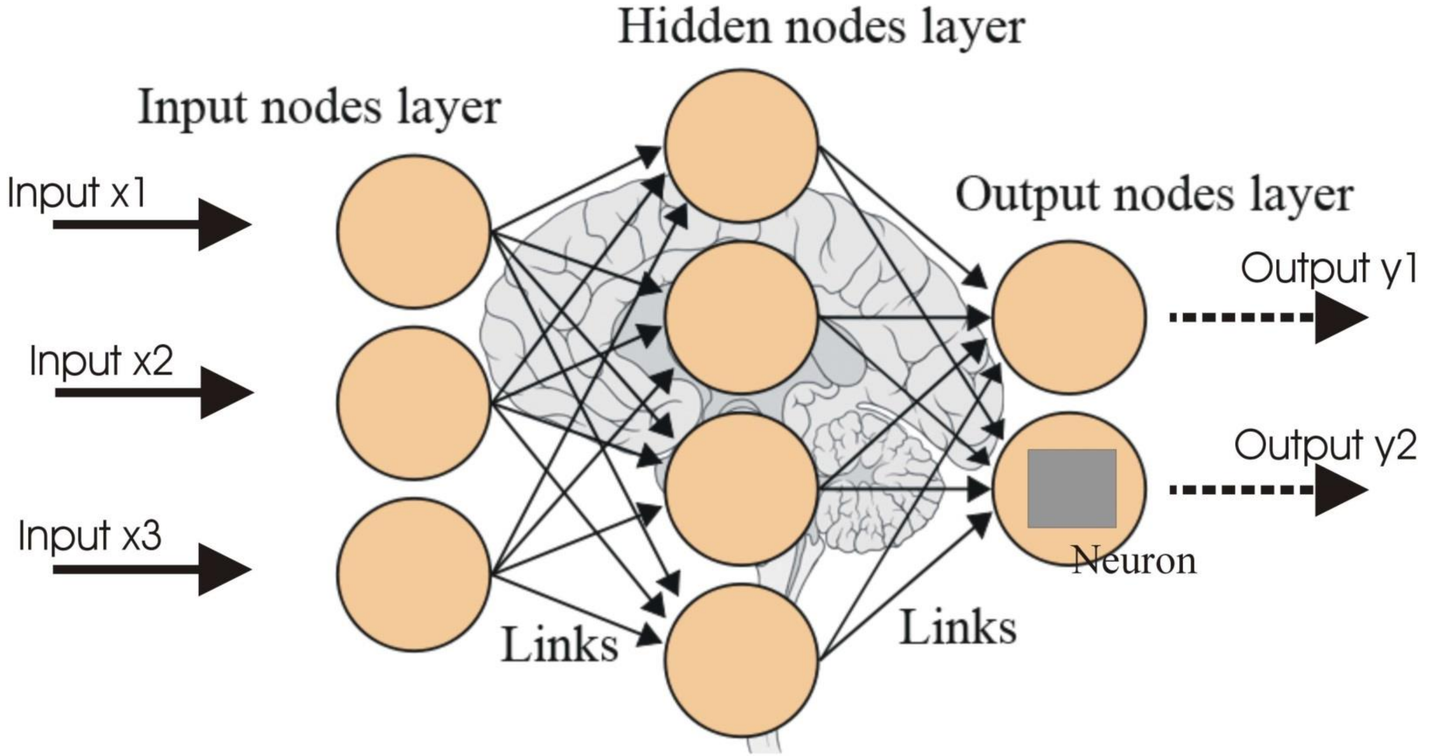
Step 5: With the gradient vector $\nabla_{\mathbf{w}} L_N(\mathbf{w})$ obtained in *Step* 4 and the learning rate η, update the values of $\mathbf{w}$ $(\text{i.e.}, \mathbf{w} \leftarrow \mathbf{w} - \eta * \nabla_{\mathbf{w}} L_N(\mathbf{w}))$.

Step 6: go to Step 1.

a loop

Where we are now...

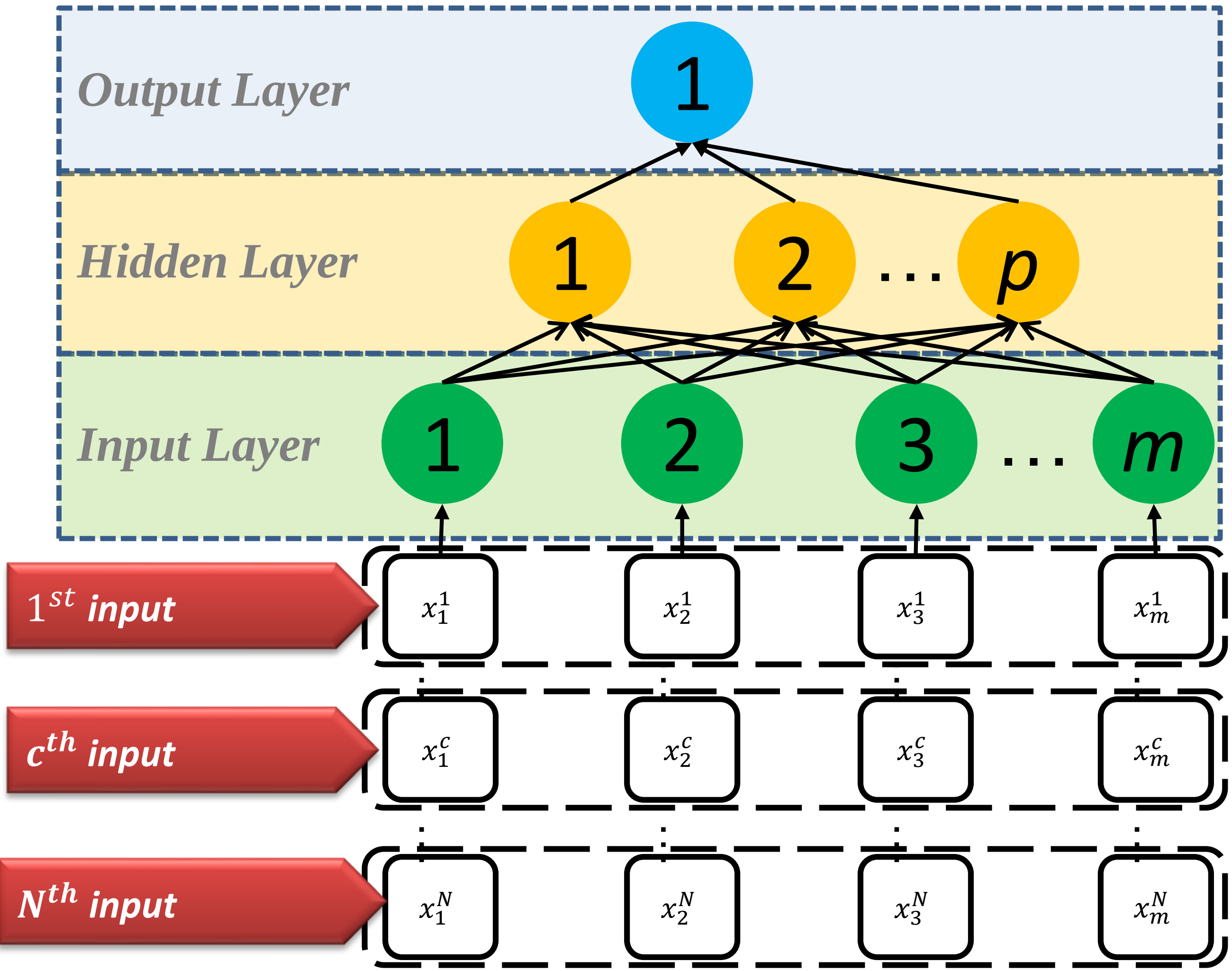
2-Layer Neural Networks; Single-hidden Layer Feed-forward Neural Networks (SLFN)



Where we are now...

The Network Structure of the SLFN with one output node

Fully-connected networks



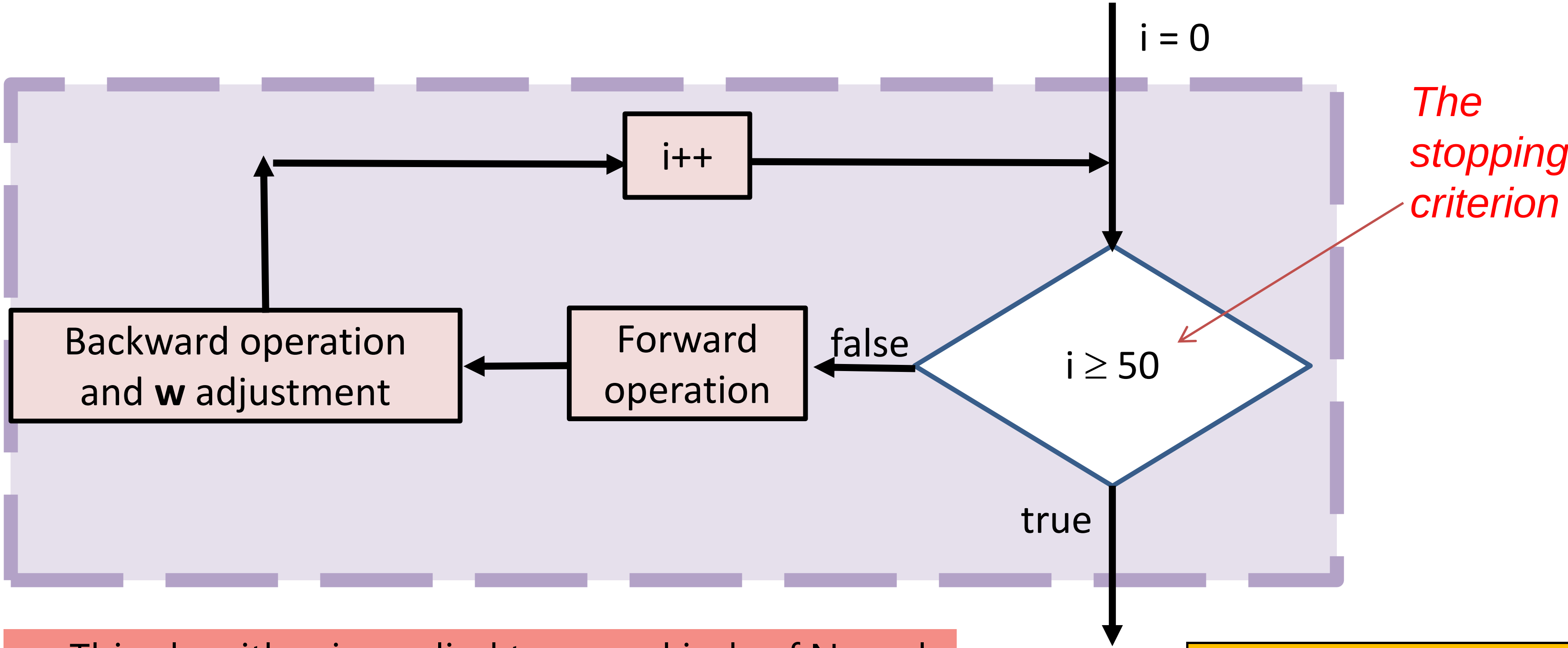
- ***1*** output node
- ***p*** hidden nodes
- ***m*** input nodes

Where we are now...

m	the number of input nodes，即單筆輸入資料中共有m個變數
p	the number of adopted hidden nodes, SLFN模型共有p個隱藏節點
w_i^o	第i個隱藏節點與輸出節點之間的激發值之權重，上標o表示該變數與輸出層相關
$\mathbf{w}^o \equiv (w_1^o, w_2^o, \ldots, w_p^o)^{\mathrm{T}}$	所有隱藏節點與輸出節點之間的激發值之權重的向量，$(\cdot)^{\mathrm{T}}$ 為矩陣$(\cdot)$的轉置矩陣
w_0^o	為輸出節點之閾值
w_{ij}^H	為第j個輸入節點與第i個隱藏節點之間的權重，上標H表示該變數與隱藏層相關
$\mathbf{w}_i^H \equiv (w_{i1}^H, w_{i2}^H, \ldots, w_{im}^H)^{\mathrm{T}}$	第i個隱藏節點與所有輸入節點即輸入層之間的權重之向量
$\mathbf{W}^H \equiv (\mathbf{w}_1^H, \mathbf{w}_2^H, \ldots, \mathbf{w}_p^H)^{\mathrm{T}}$	所有隱藏節點的權重的矩陣，即隱藏層與輸入層之間的權重的矩陣
w_{i0}^H	第i個隱藏節點之閾值
$\mathbf{w}_0^H \equiv (w_{1,0}^H, w_{2,0}^H, \ldots, w_{p0}^H)^{\mathrm{T}}$	所有隱藏節點的閾值之向量
$\mathbf{x}^c \equiv (x_1^c, x_2^c, \ldots, x_m^c)^{\mathrm{T}}$	the input vector of the c^{th} case
$\boldsymbol{a}^c \equiv \left(a_1^c, a_2^c, \ldots, a_p^c\right)^{\mathrm{T}}$	the hidden activation vector of the c^{th} case
y^c	the desired output value associated with $\mathbf{x}^c$

Where we are now...

The flowchart of learning algorithms for 2-layer neural networks in CS231n



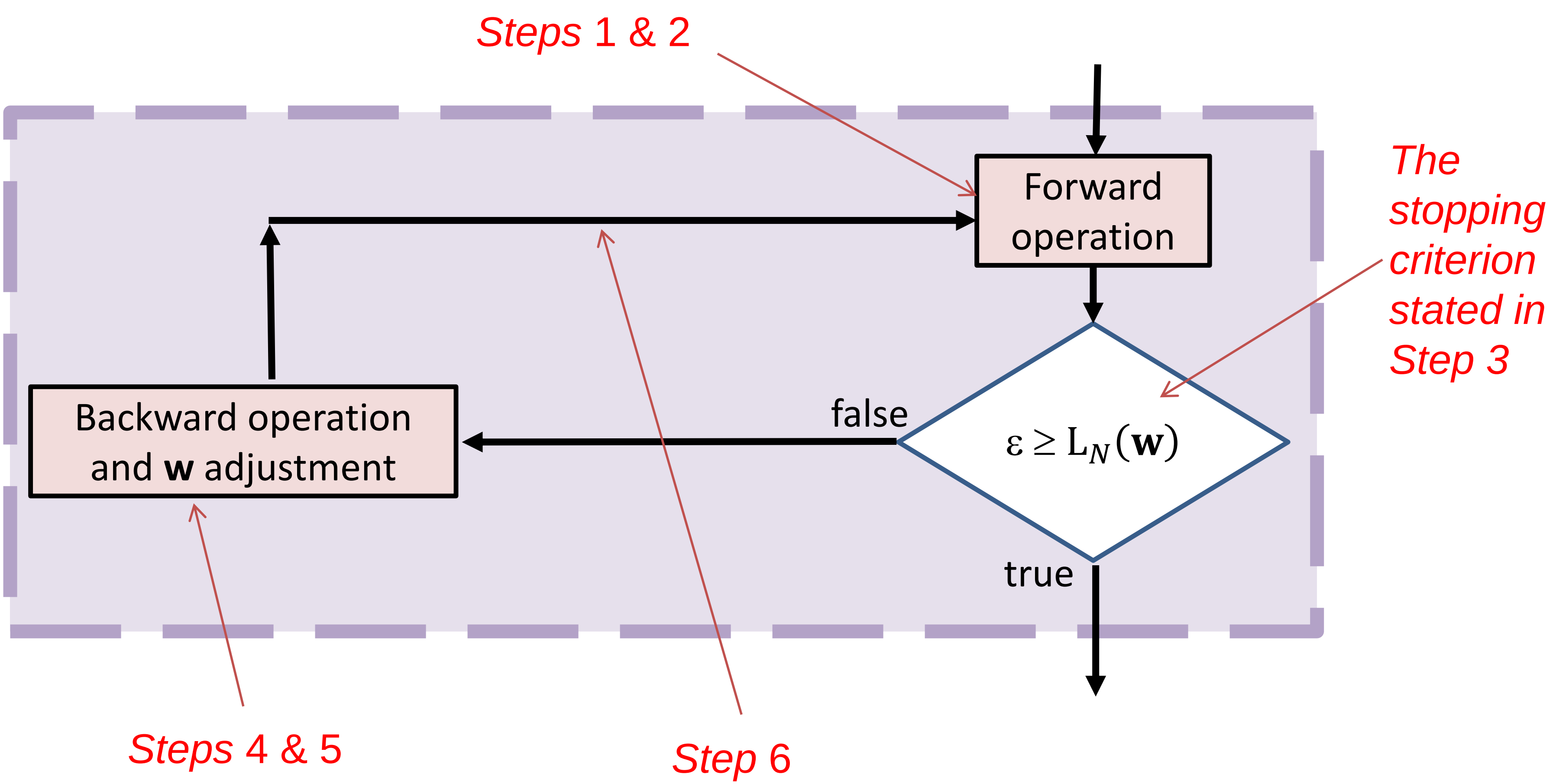
- This algorithm is applied to many kinds of Neural Networks, including 2-layer neural networks, CNN, RNN, reinforcement learning, GAN, BERT, and so on.
- The learning process stops when the stopping criterion is satisfied.

Hyperparameters:

- p
- Optimizer: SGD
- Epoch upper bound: 50
- Learning rate: 1e-6

Where we are now...

The flowchart of BP learning algorithm



Hyperparameters:

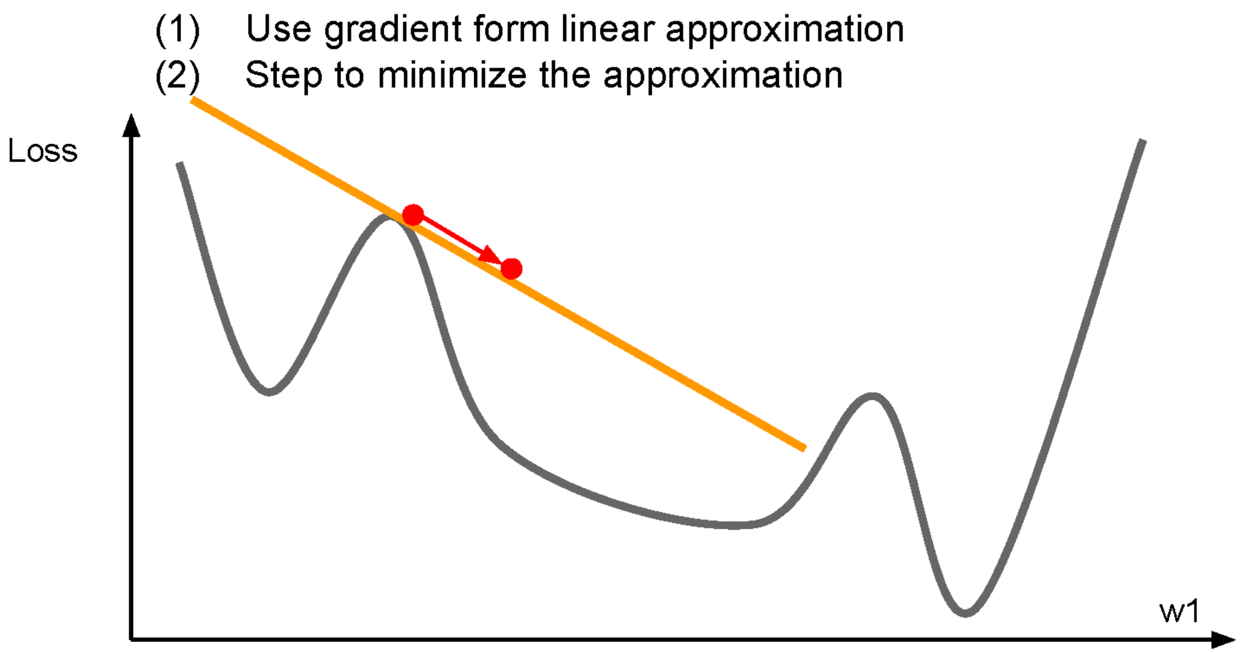
- p
- Optimizer: SGD
- Learning rate: 0.1
- ε

Q: Can this the learning process stop through satisfying the stopping criterion?
A: May not be! So, this stopping criterion is not good and thus this algorithm is not good.

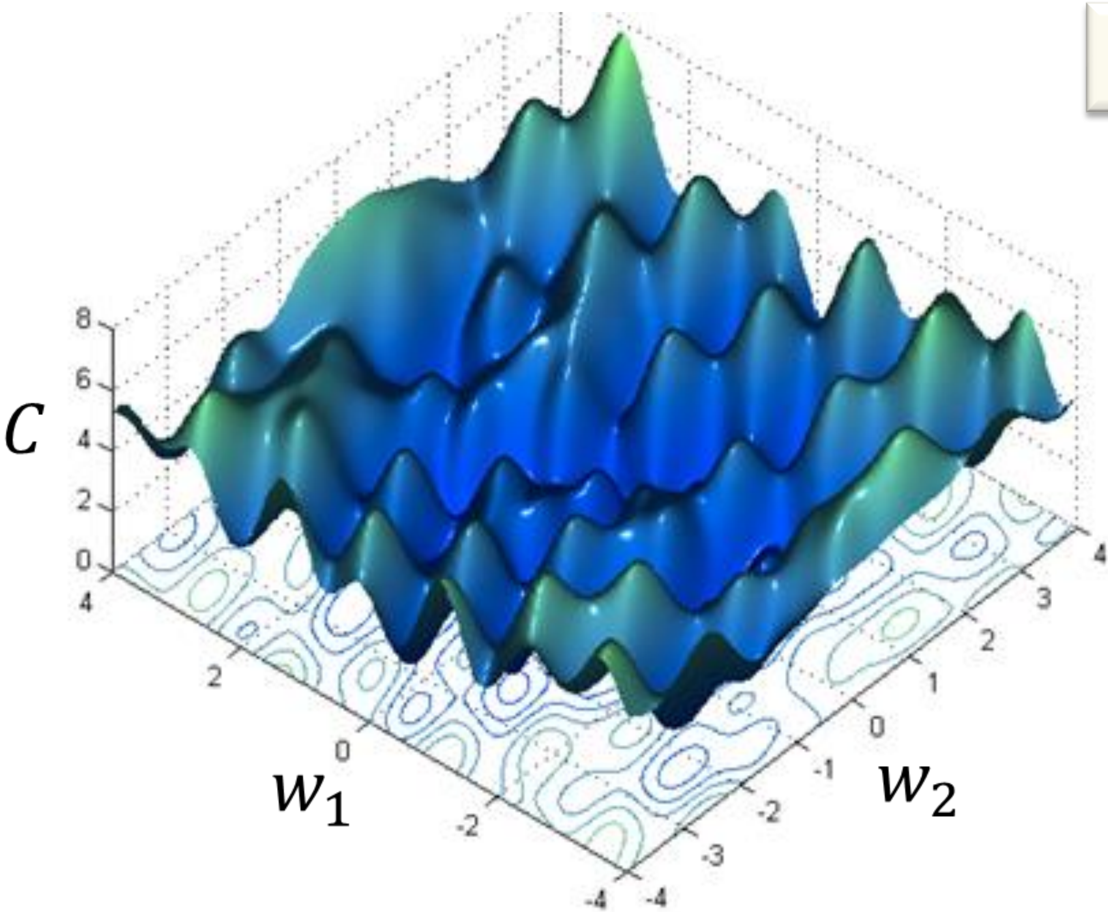
Where we are now...

The BP learning algorithm is gradient-descent-based

First-Order Optimization



A key issue of gradient-descent-based learning: it never guarantees a global minimum



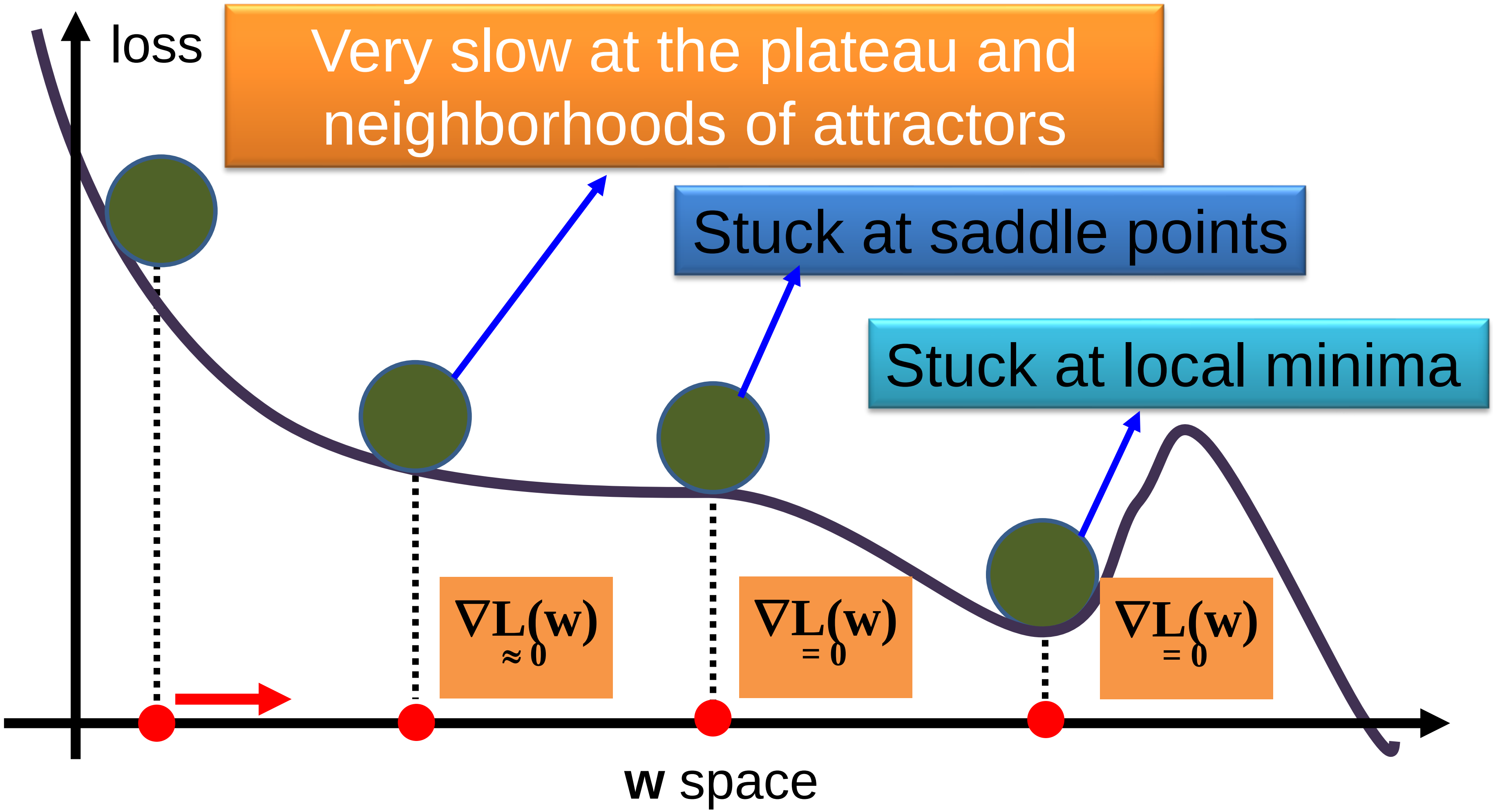
different initial weights

↓

different minimum

Where we are now...

Learning dilemma of gradient-descent-based learning



Goals of modeling/learning are reasonable inferencing

Rule-based

- $\mathbf{x} \rightarrow y$
- Modeling phase ← store all pairs of ($\mathbf{x}$, y); memorizing
- Inferencing phase ← put in any $\mathbf{x}$ to get its inferencing result.

Learning-based

- $y = f(\mathbf{x})$
- Modeling/learning phase ← tune weights according to pairs of ($\mathbf{x}$, y); learning
- Inferencing phase ← put in any $\mathbf{x}$ to get its inferencing result.

Goals of modeling/learning are reasonable inferencing

Rule-based

- $\mathbf{x} \rightarrow y$
- Modeling phase

✓ $\mathbf{x}^1 \rightarrow y^1$

✓ $\mathbf{x}^2 \rightarrow y^2$

✓ $\mathbf{x}^2 \rightarrow y^3$ and $y^3 \neq y^2$

- The inferencing result $f(\mathbf{x}^2)$? Since $y^3 \neq y^2$, the inferencing result of $\mathbf{x}^2$ is NaN

Learning-based

- $y = f(\mathbf{x})$
- Modeling/learning phase

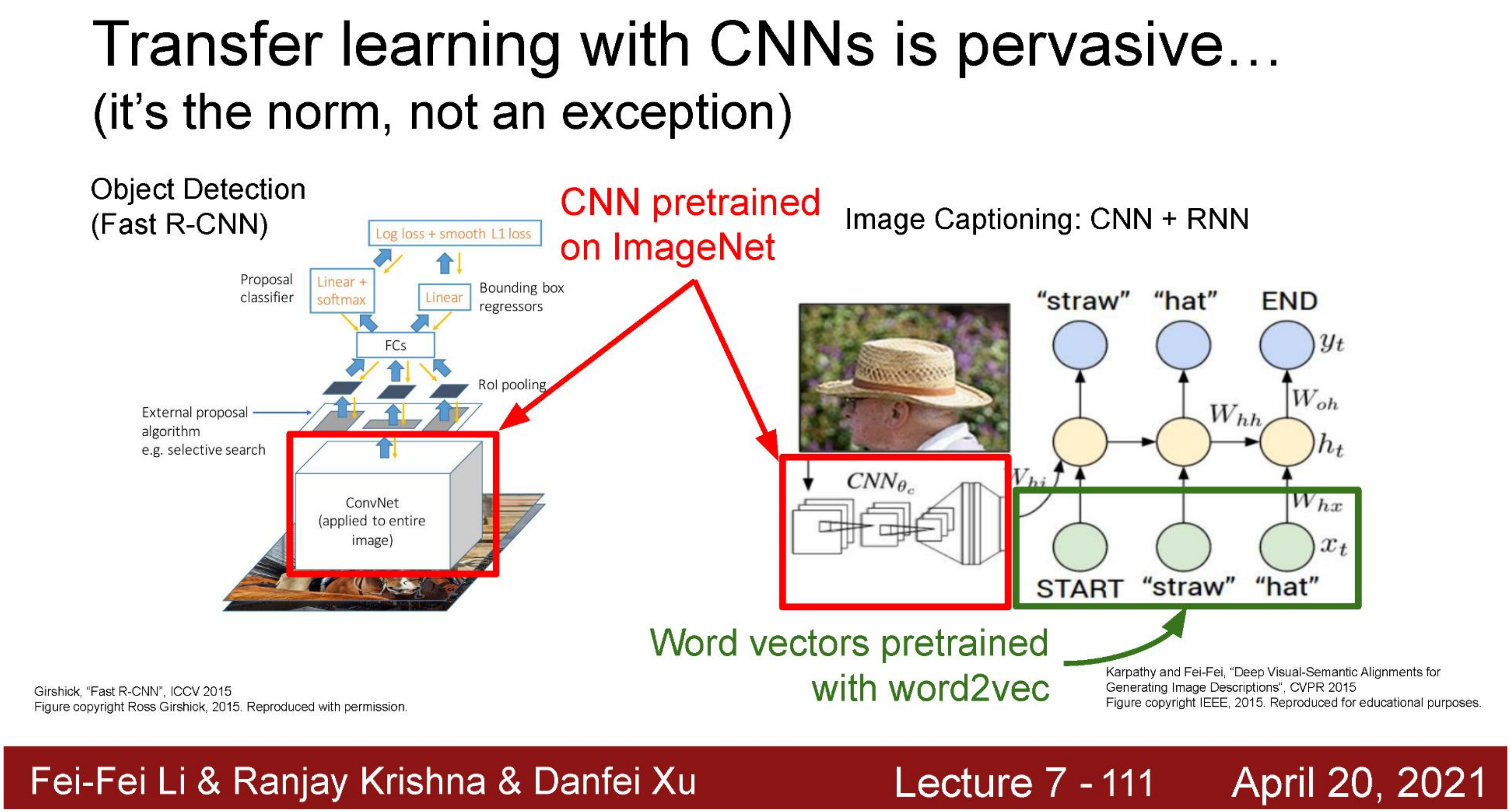
✓ $\mathbf{x}^1 \rightarrow y^1$

✓ $\mathbf{x}^2 \rightarrow y^2$

✓ $\mathbf{x}^2 \rightarrow y^3$ and $y^3 \neq y^2$

- If the learning goal is $L_N(\mathbf{w}) = 0$, then the *learnng cannot be done.*
- If the learning goal is $L_N(\mathbf{w}) > 0$ that allows $f(\mathbf{x}^2) = (y^3 + y^2)/2$, then the learning can be done and the inferencing result of $\mathbf{x}^2$ is $f(\mathbf{x}^2) = (y^3 + y^2)/2$

y label


Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)
Object Detection
(Fast R-CNN)
Log loss + smooth L1 loss
Proposal classifier
Linear + softmax
Linear
Bounding box regressors
FCs
RoI pooling
External proposal algorithm e.g. selective search
ConvNet (applied to entire image)
CNN pretrained on ImageNet
Image Captioning: CNN + RNN
"straw" "hat" END
y_t
W_{oh}
W_{hh}
h_t
CNN_{θ_c}
W_{hi}
W_{hx}
x_t
START "straw" "hat"
Word vectors pretrained with word2vec
Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.
Fei-Fei Li & Ranjay Krishna & Danfei Xu
Lecture 7 - 111
April 20, 2021

AI Applications
Design (**x** attributes & *y* label)

X:
- ✓ 年齡
- ✓ 性別
- ✓ 診斷
- ✓ 分期
- ✓ 診斷癌症期間
- ✓ 復發與否
- ✓ Kps (身體功能)
- ✓ gs1-gs22 (22項)症狀 (0:無; 1:有)

y: 疲倦，個案總計686位，過去一周平均疲倦程度 (f3) (0-10分，real-value variable) → 過去一周疲倦分組 (Gf3) (分成三組，binary variable) :
- ✓ 無: 46位
- ✓ 輕度:346 位
- ✓ 中至重度:294位

AI Applications Design (**x** attributes)

attributes (x)

- 年齡 ← binary or real number?
- 性別 ← binary
- 診斷 ← binary or real number?
- 分期 ← binary
- 診斷癌症期間 ← binary or real number?
- 復發與否 ← binary
- Kps (身體功能) ← binary
- gs1-gs22 (22項)症狀 (0:無; 1:有) ← binary

AI Applications Design (*y* label)

Output value: real number

疲倦
- ✓ 無: 46位
- ✓ 輕度:346 位
- ✓ 中至重度:294位

Learning phase:
y (i.e., target output):
- ✓ 無: 0
- ✓ 輕度: 5
- ✓ 中至重度: 10

Inferencing phase:
f (i.e., actual output):
- ✓ [-2.5, 2.5) → 無
- ✓ [2.5, 7.5) → 輕度
- ✓ [7.5, 12.5) → 中至重度
- ✓ $(-\infty, -2.5)$ OR $[12.5, \infty)$ → unknown

Output value: binary number

疲倦
- ✓ 無: 46位
- ✓ 輕度:346 位
- ✓ 中至重度:294位

Learning phase:
y (i.e., target output):
- ✓ 無: (0, 0)
- ✓ 輕度: (1, 0)
- ✓ 中至重度: (1, 1)

Inferencing phase:
f (i.e., actual output):
- ✓ (0, 0) → 無
- ✓ (1, 0) → 輕度
- ✓ (1, 1) → 中至重度
- ✓ (0, 1) → unknown

AI Applications Design (**x** attributes & y label)

- Data

x attributes	
x1	性別
x2	年齡
x3	國籍
x4	婚姻狀態
x5	直系親屬數
x6	最高學歷
x7	來台時長
x8	平均月收入
x9	剩餘居留時間

x attributes	
x10	借款時長
x11	借款金額
x12	用途
x13	工作性質
x14	工作地點
x15	雇主資訊
x16	薪資如期撥入
x17	薪資撥付方式
x18	薪資結匯方式

y label	
y (i.e., target output, real number)	信用評級 有5個等級

y	信用評級
1	E (最差)
2	D
3	C
4	B
5	A (最好)

AI Applications Design (y label)

- y (i.e., target output) $\in \{1, 2, 3, 4, 5\}$
- At the learning phase, let $\varepsilon = 0.2$. Then the learning goal is to make f (i.e., actual output) $\in \{[0.8, 1.2], [1.8, 2.2], [2.8, 3.2], [3.8, 4.2], [4.8, 5.2]\}$.
- At the inferencing phase, $y = 1$ if $f \in [0.5, 1.5)$; $y = 2$ if $f \in [1.5, 2.5)$; $y = 3$ if $f \in [2.5, 3.5)$; $y = 4$ if $f \in [3.5, 4.5)$; $y = 5$ if $f \in [4.5, 5.5)$
- y is unknown if $f < 0.5$ OR $f \geq 5.5$.

The learning goals
(also the stopping criteria for the learning)

The learning process should stop when

1. $\mathrm{L}_N(\mathbf{w}) = 0$
2. a tiny $\mathrm{L}_N(\mathbf{w})$ value
3. $|f(\mathbf{x}^c, \mathbf{w}) - y^c| < \varepsilon \ \forall\, c \ \ with\ \varepsilon\ being\ tiny$

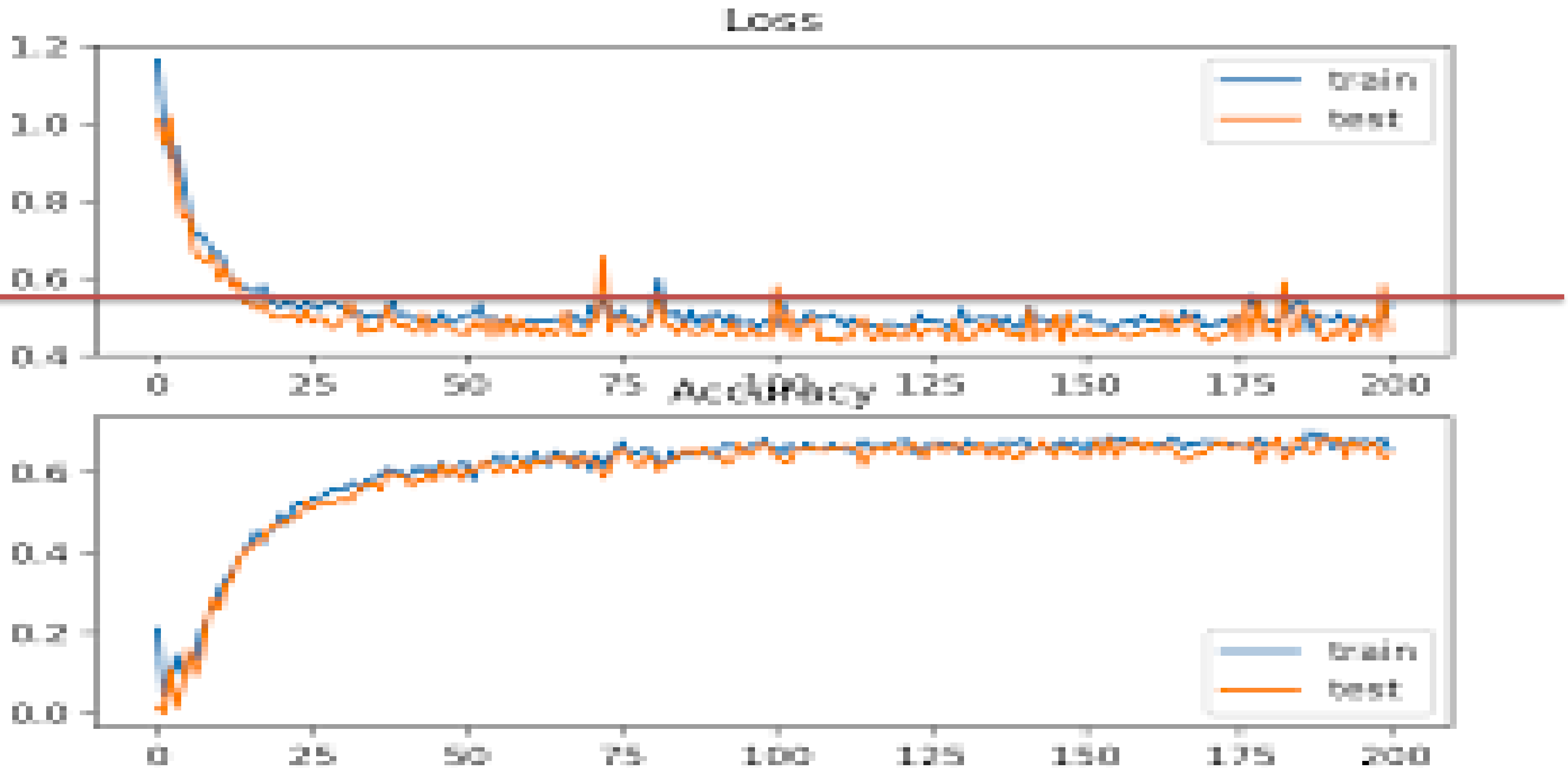
$\boxed{\mathrm{L}_N(\mathbf{w}) \equiv \frac{1}{N} \sum_{c=1}^{N} (f(\mathbf{x}^c, \mathbf{w}) - y^c)^2}$

The learning process should stop when

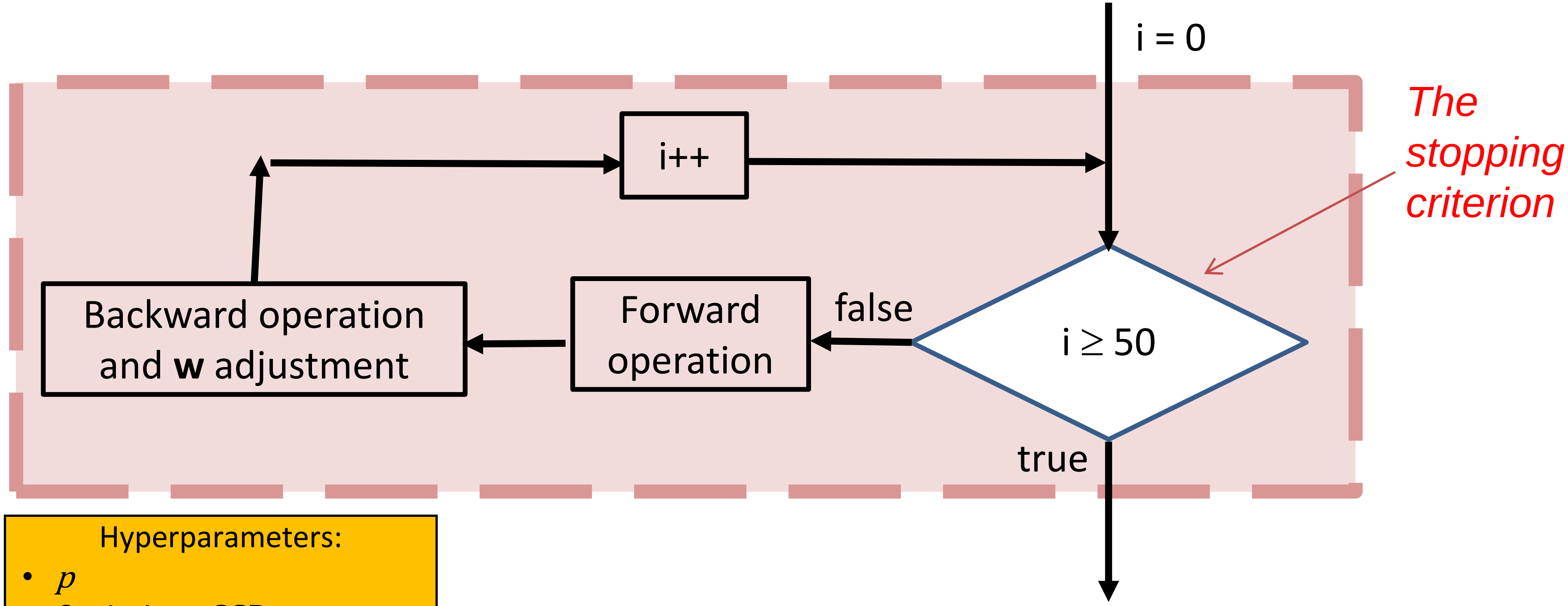
1. ~~$\mathrm{L}_N(\mathbf{w}) = 0$~~
2. a tiny $\mathrm{L}_N(\mathbf{w})$ value
3. $|f(\mathbf{x}^c, \mathbf{w}) - y^c| < \varepsilon \ \forall\, c \ \ with\ \varepsilon\ being\ tiny$

- Each reasonable learning goal can be used as a stopping criterion.
- Different stopping criterion results in different length of training time and different model.

Effect of Stopping criterion of $\varepsilon \geq L_N(\mathbf{w})$



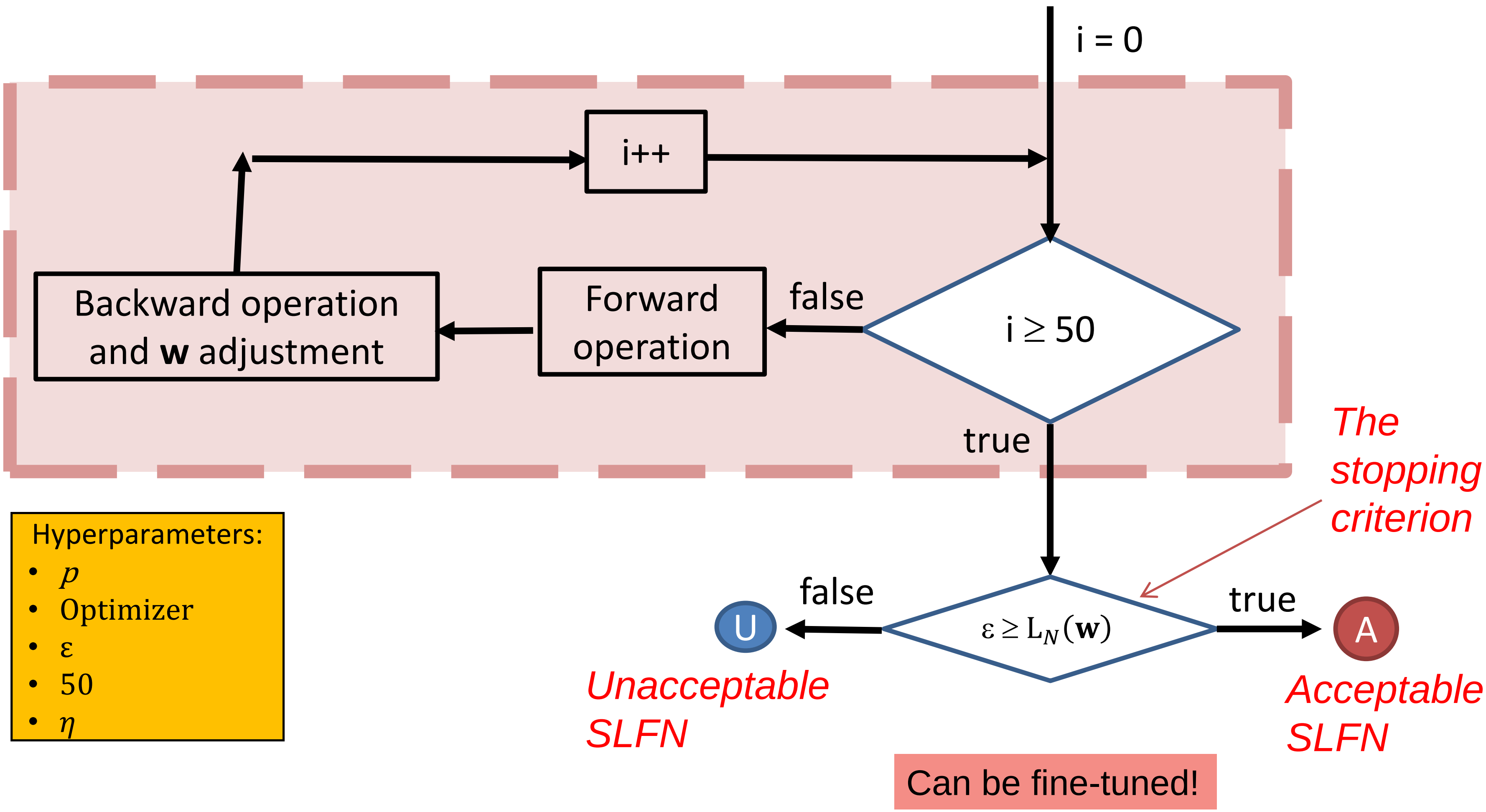
The flowchart of learning algorithms for 2-layer neural networks in CS231n



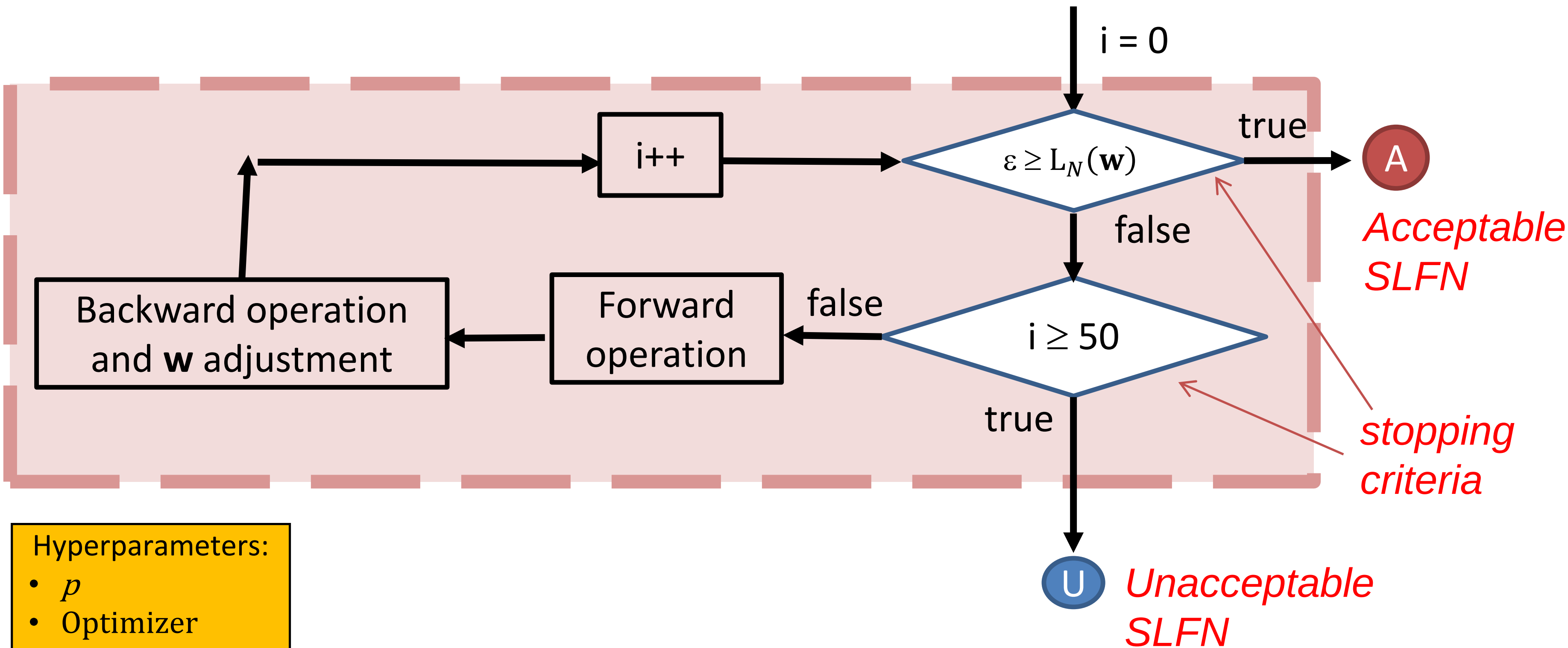
Hyperparameters:

- p
- Optimizer: SGD
- Epoch upper bound: 50
- Learning rate: 1e-6

The flowchart of learning algorithms for 2-layer neural networks in CS231n



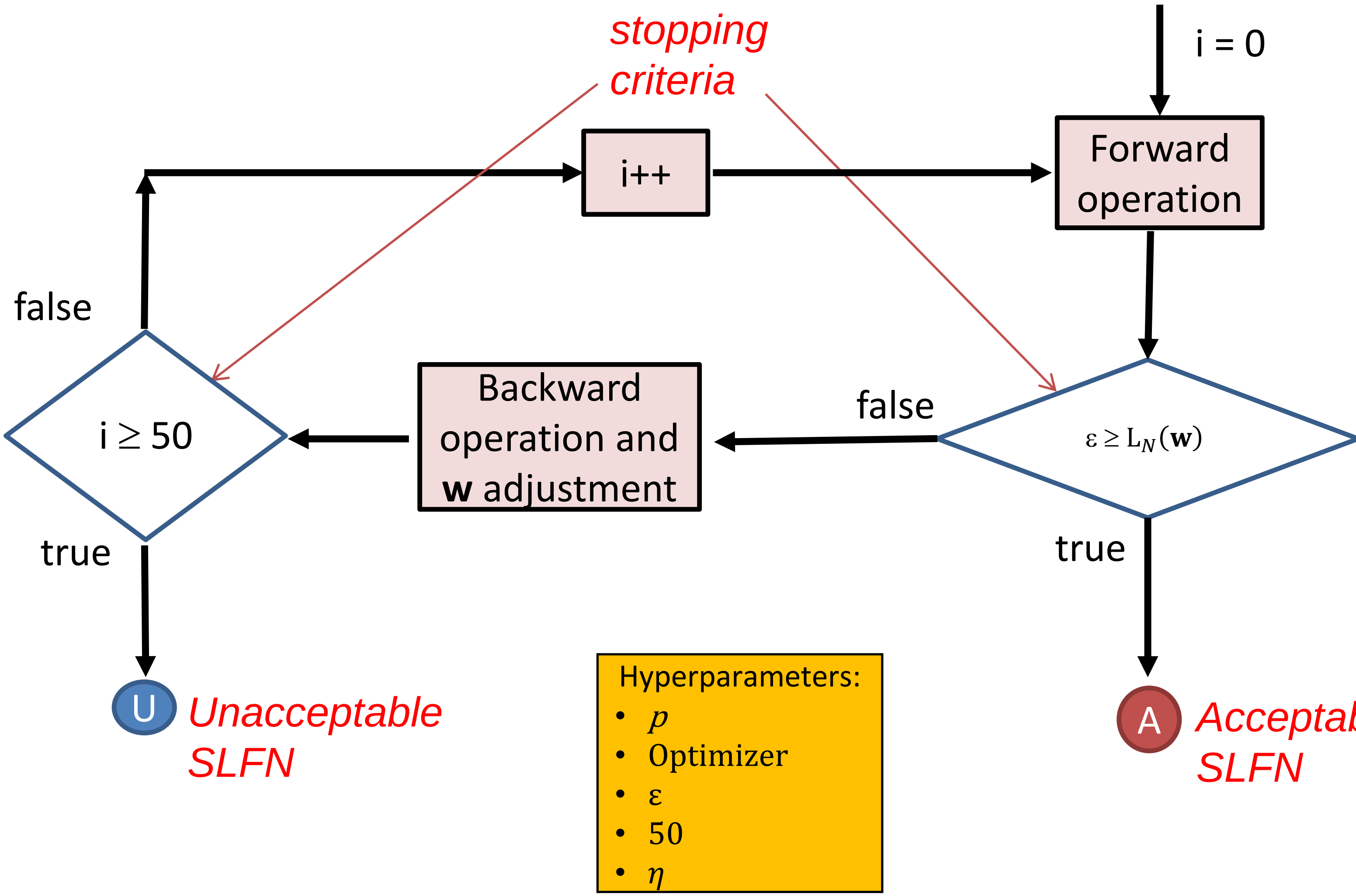
The flowchart of learning algorithms for 2-layer neural networks in CS231n



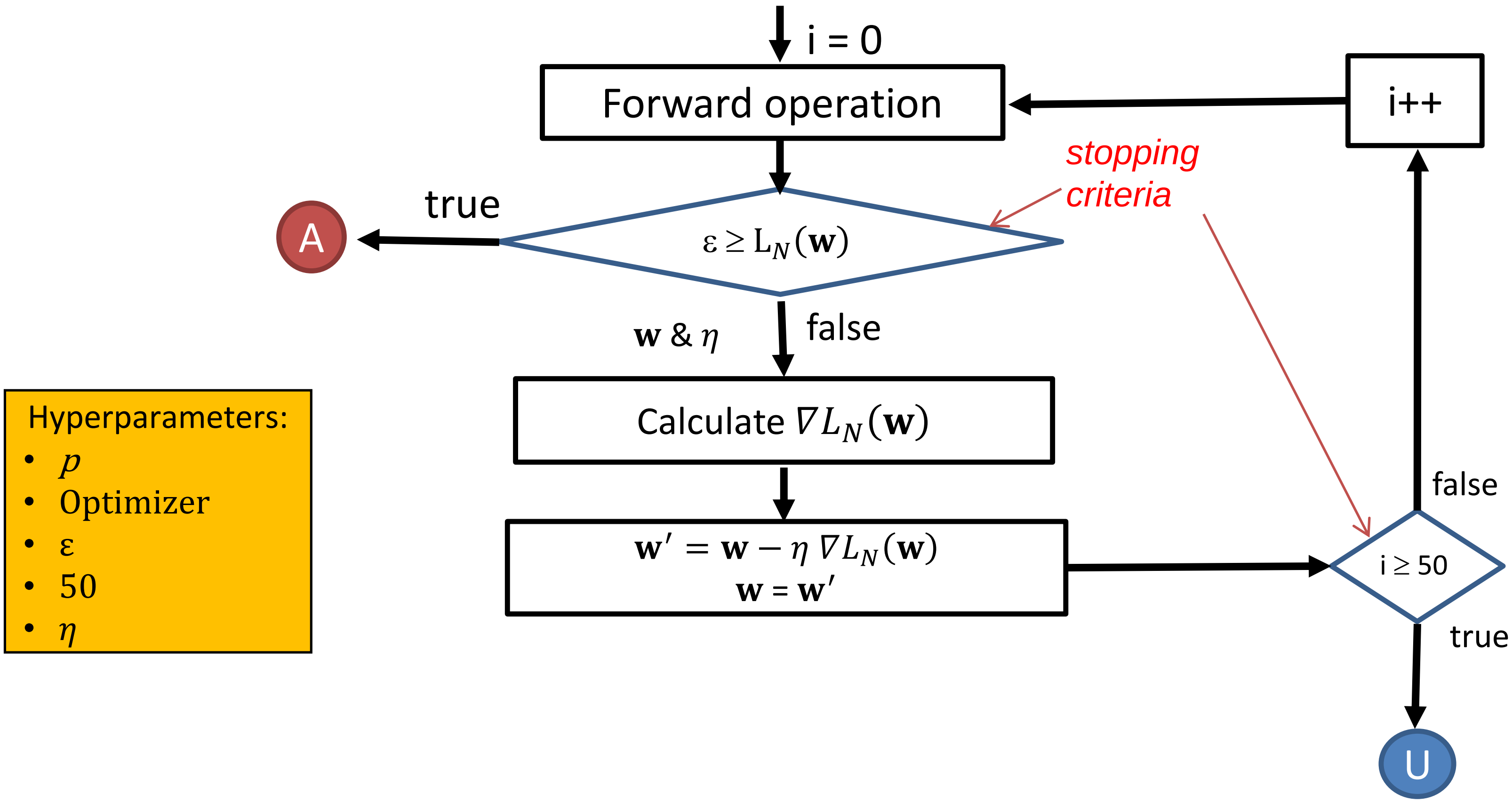
Hyperparameters:
- p
- Optimizer
- ε
- 50
- η

Can be fine-tuned!

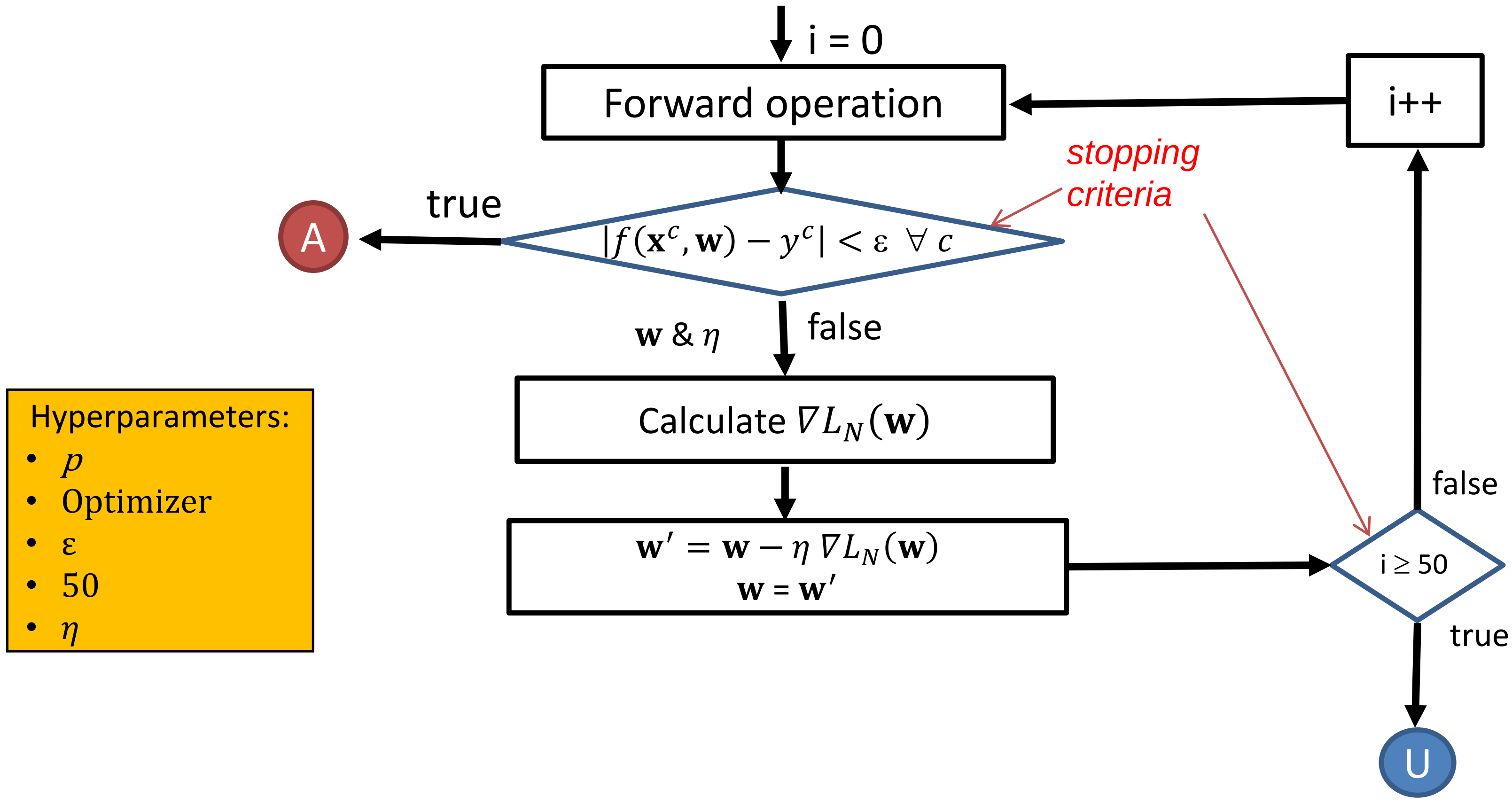
The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN

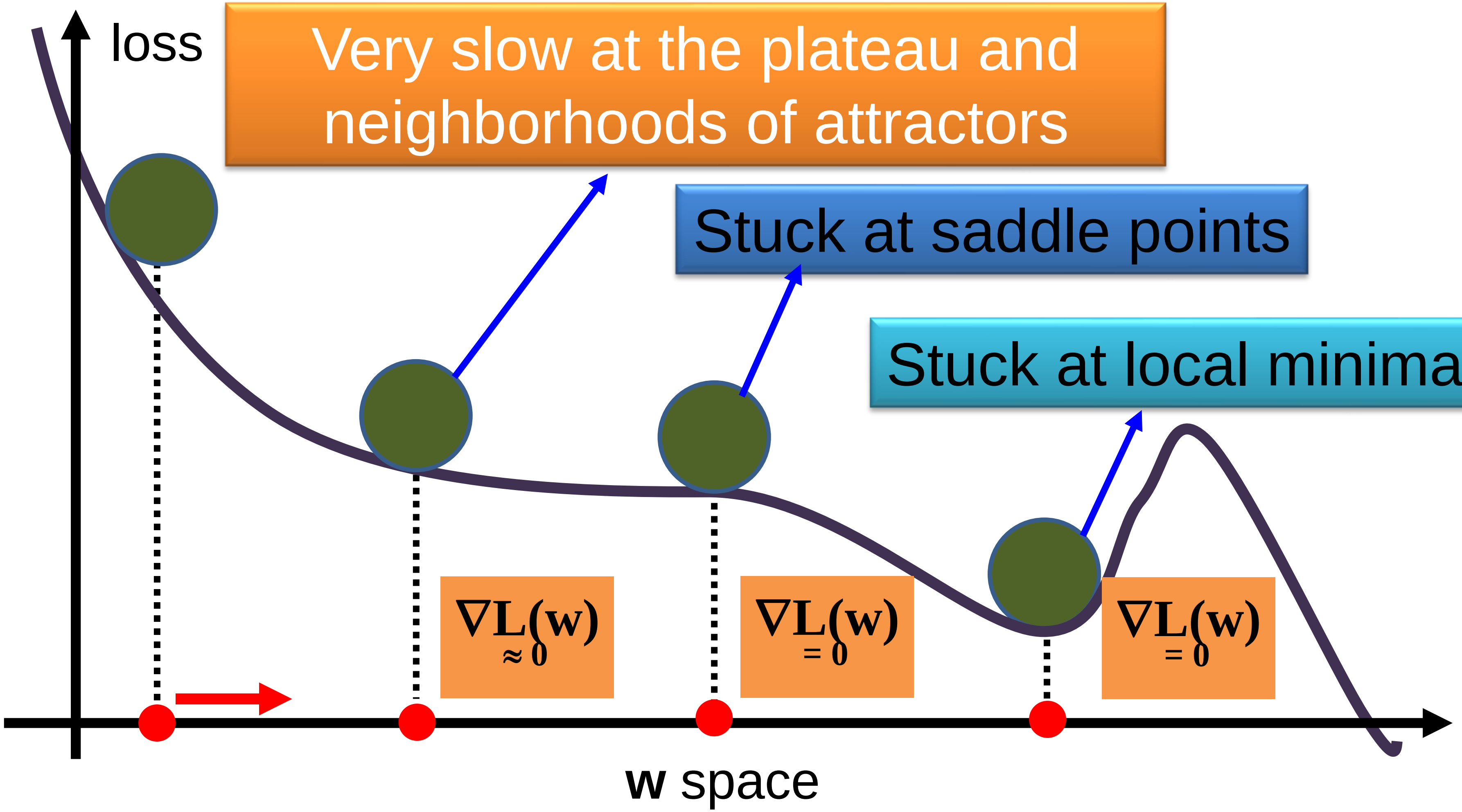


The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



Where we are now...

Learning dilemma of gradient-descent-based learning



Extra stopping criteria for the learning (not the learning goals)

1. The learning process should stop when $\|\nabla_{\mathbf{w}} \mathrm{L}_N(\mathbf{w})\| = 0$ but a tiny $\mathrm{L}_N(\mathbf{w})$ value cannot be accomplished.
2. The learning process should stop when $\|\nabla_{\mathbf{w}} \mathrm{L}_N(\mathbf{w})\|$ is tiny but a tiny $\mathrm{L}_N(\mathbf{w})$ value cannot be accomplished.
3. The learning process should stop when η (the learning rate) is tiny but a tiny $\mathrm{L}_N(\mathbf{w})$ value cannot be accomplished.

The neighborhood of undesired attractors, where $\|\nabla_{\mathbf{w}} \mathrm{L}_N(\mathbf{w})\| \approx 0$:

a) the local minimum, the saddle point, or the plateau.
b) the global minimum of the defective network architecture.

Extra stopping criteria for the learning (not the learning goals)

1. ~~The learning process should stop when $\|\nabla_{\mathbf{w}} \mathrm{L}_N(\mathbf{w})\| = 0$ but a tiny $\mathrm{L}_N(\mathbf{w})$ value cannot be accomplished.~~
2. The learning process should stop when $\|\nabla_{\mathbf{w}} \mathrm{L}_N(\mathbf{w})\|$ is tiny but a tiny $\mathrm{L}_N(\mathbf{w})$ value cannot be accomplished.
3. The learning process should stop when η (the learning rate) is tiny but a tiny $\mathrm{L}_N(\mathbf{w})$ value cannot be accomplished.

The neighborhood of undesired attractors, where $\|\nabla_{\mathbf{w}} \mathrm{L}_N(\mathbf{w})\| \approx 0$:

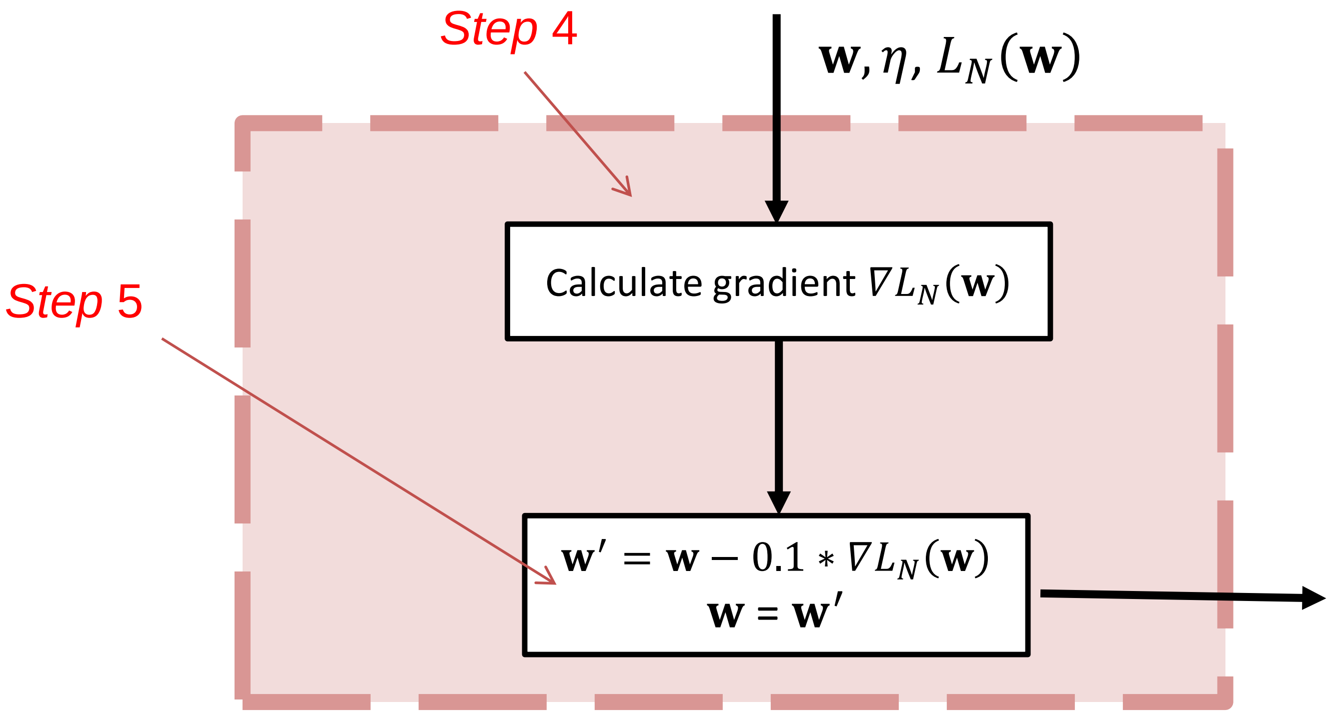
a) the local minimum, the saddle point, or the plateau.

b) the global minimum of the defective network architecture.

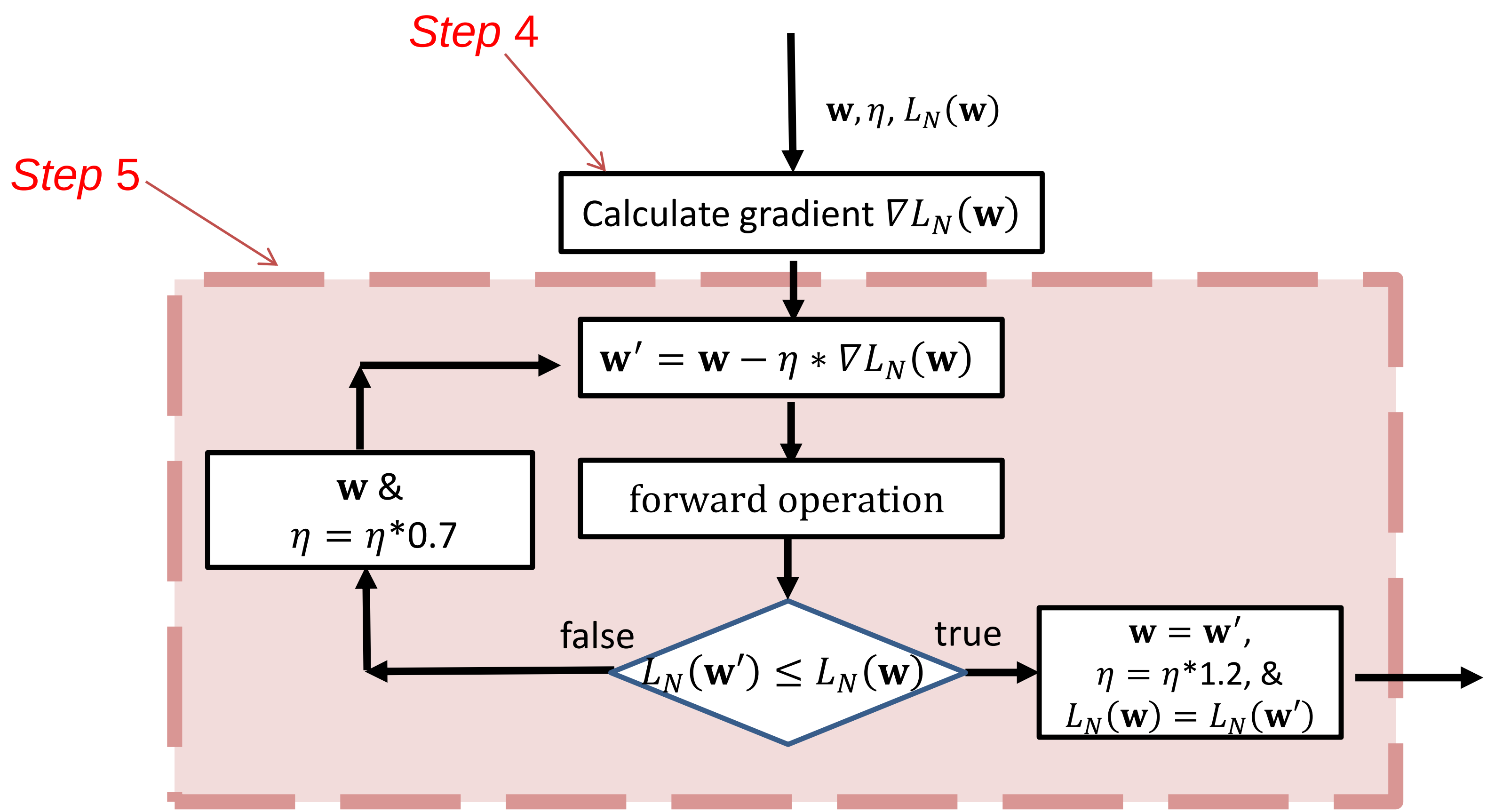
Detecting the neighborhood of undesired attractors, where η is tiny

The arrangement of adaptable learning rate

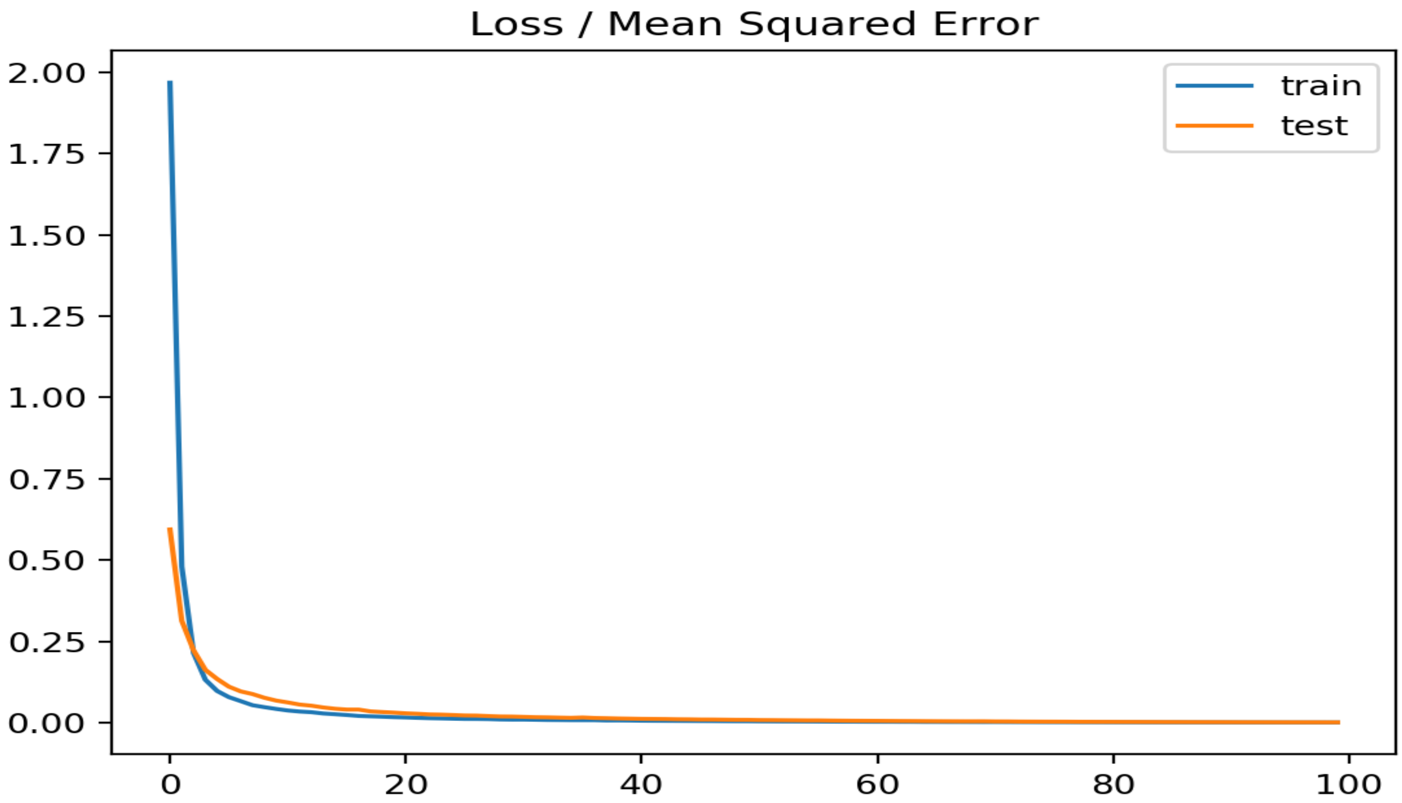
Module of backward operation and **w** adjustment



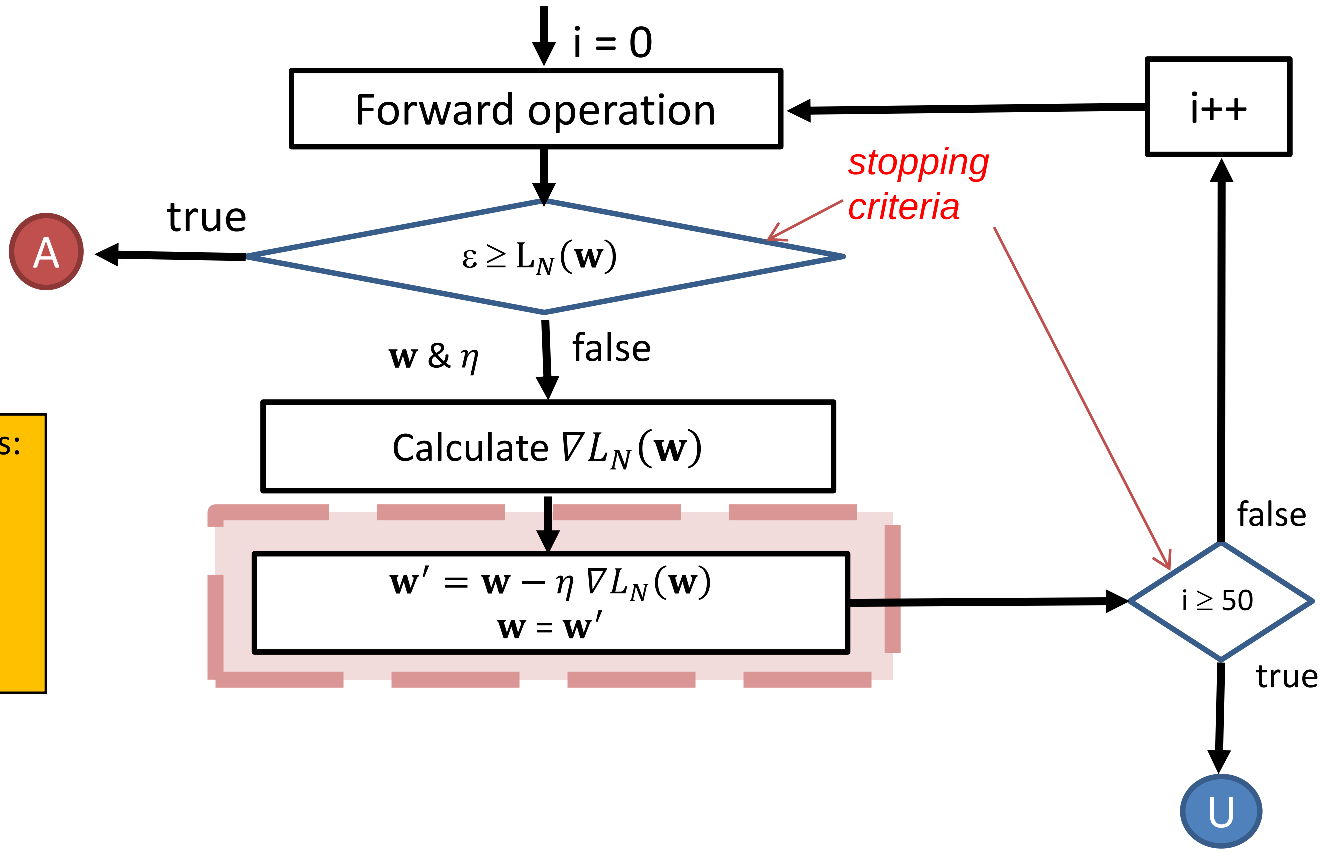
Module of backward operation and **w** adjustment with the adaptable learning rate arrangement guaranteeing the decrease of $L_N(\mathbf{w})$ in each epoch



The effort of adaptable learning rate



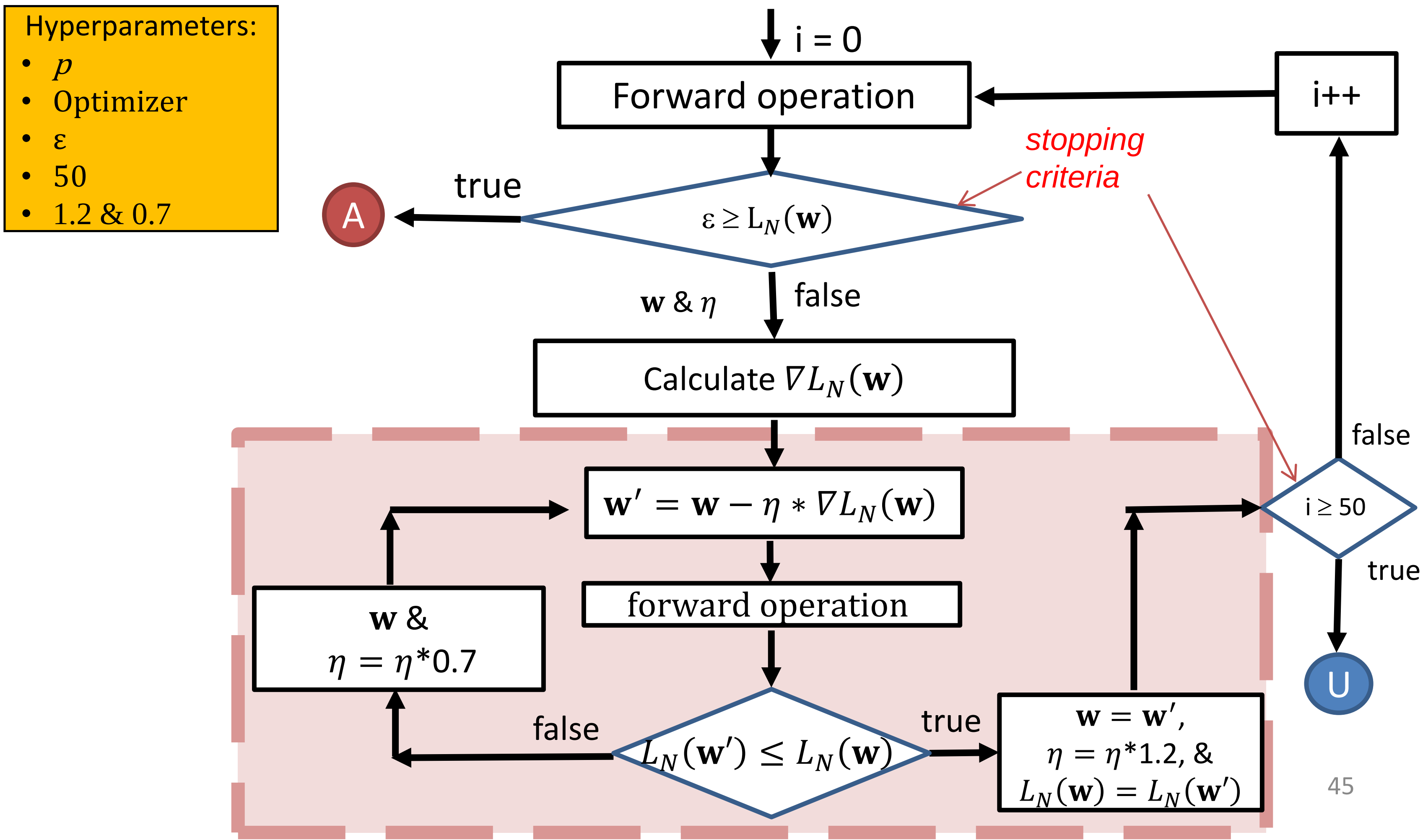
The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



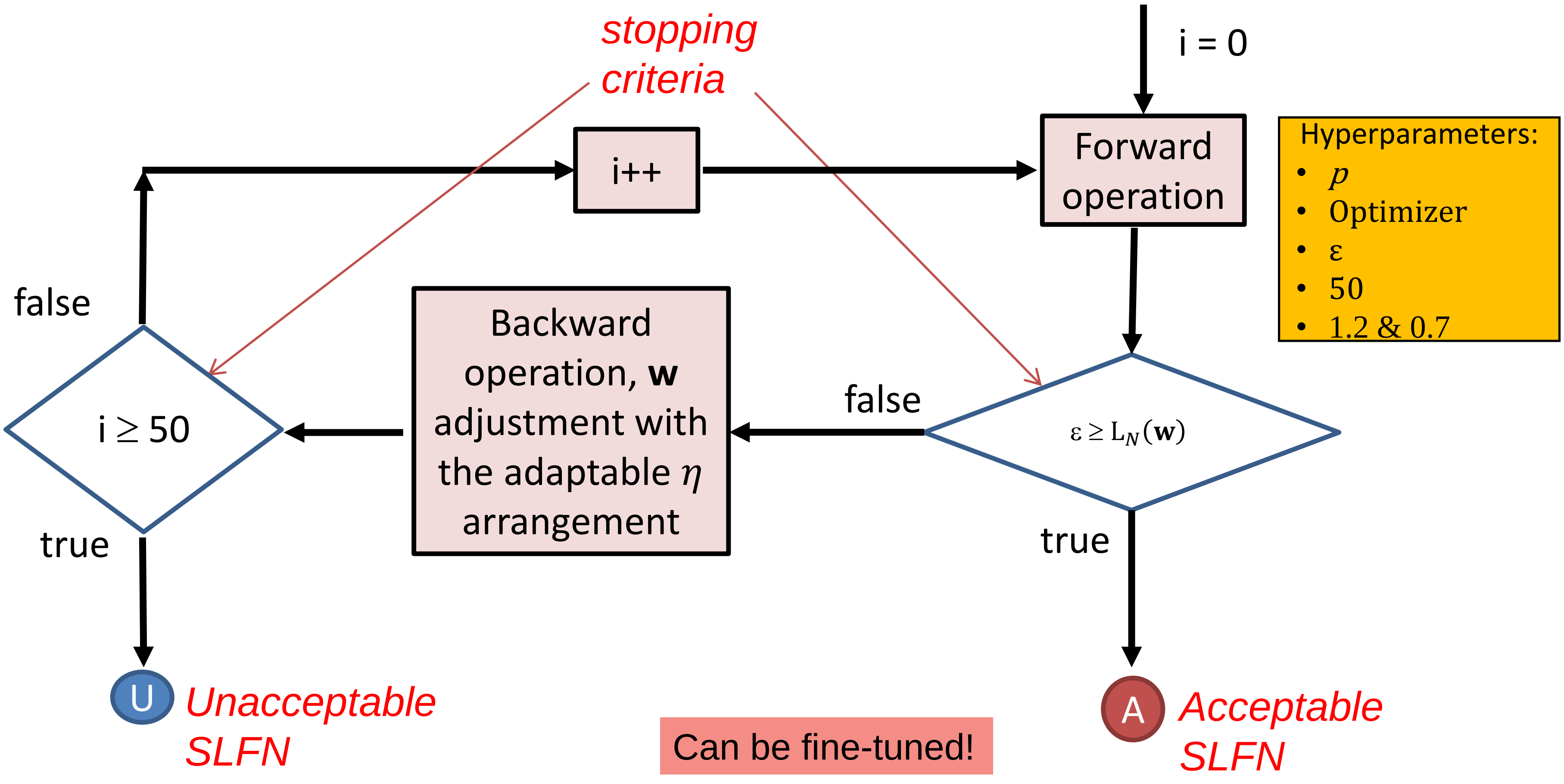
Hyperparameters:

- p
- Optimizer
- ε
- 50
- η

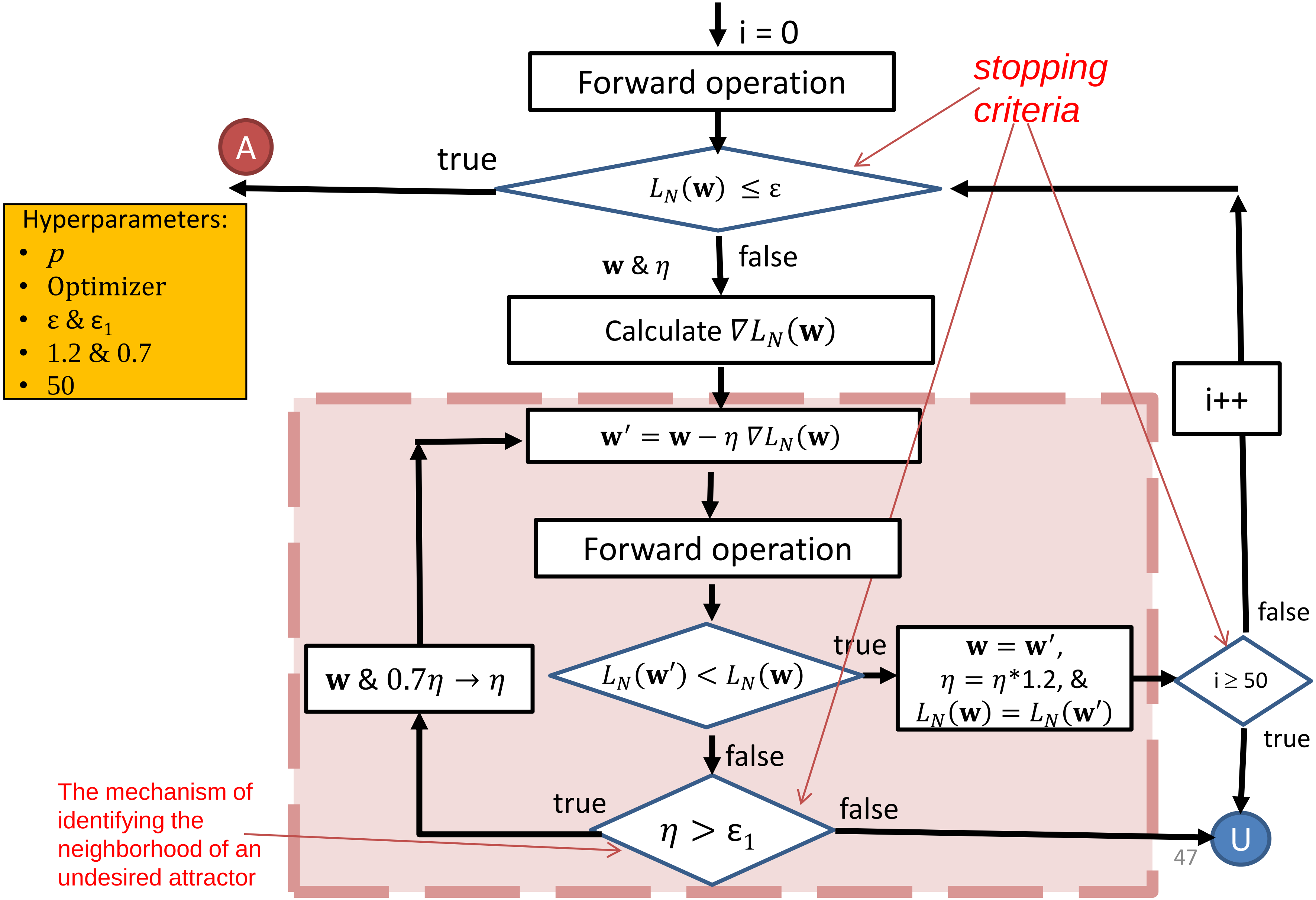
The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



The flowchart of learning algorithm including three stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



Stopping criteria (also the learning goals) for the learning for regression problems

The learning process should stop when

1. $\mathrm{L}_N(\mathbf{w}) = 0$
2. a tiny $\mathrm{L}_N(\mathbf{w})$ value
3. $|f(\mathbf{x}^c, \mathbf{w}) - y^c| < \varepsilon \ \forall c$ *with* ε *being tiny*

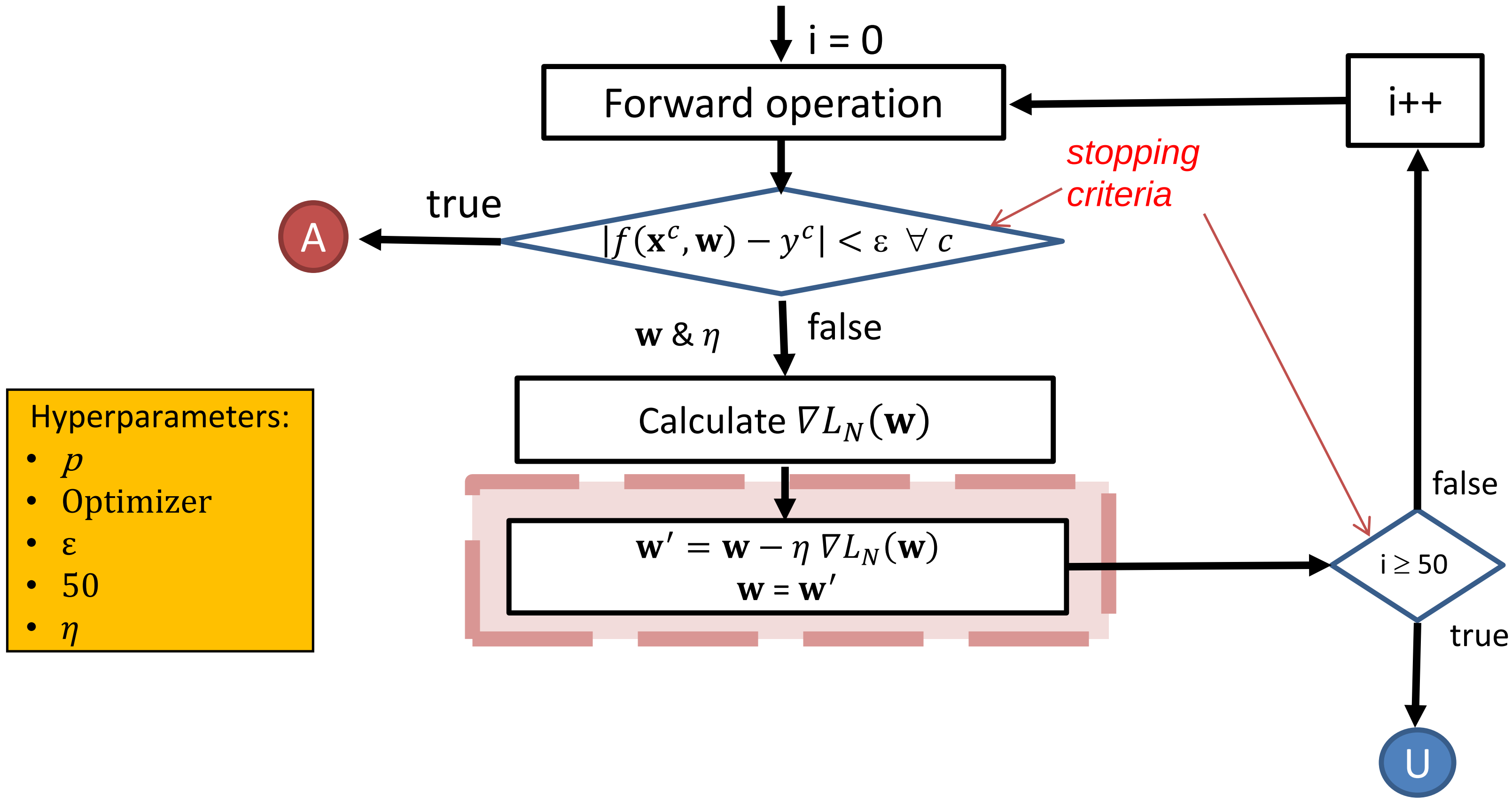
$$\mathrm{L}_N(\mathbf{w}) \equiv \frac{1}{N} \sum_{c=1}^{N} (f(\mathbf{x}^c, \mathbf{w}) - y^c)^2$$

The learning process should stop when

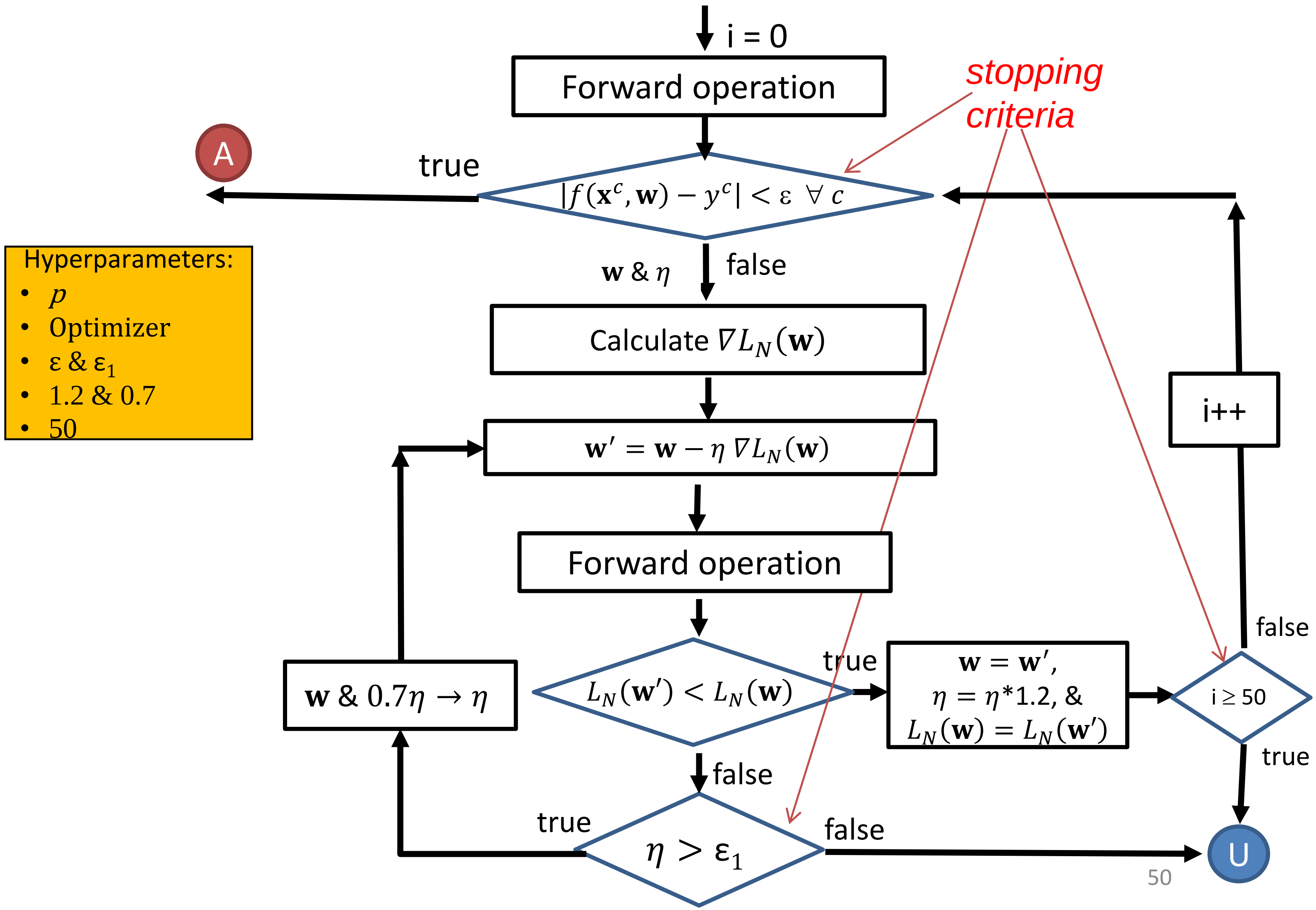
1. ~~$\mathrm{L}_N(\mathbf{w}) = 0$~~
2. a tiny $\mathrm{L}_N(\mathbf{w})$ value
3. $|f(\mathbf{x}^c, \mathbf{w}) - y^c| < \varepsilon \ \forall c$ *with* ε *being tiny*

- Each reasonable learning goal can be used as a stopping criterion.
- Different stopping criterion results in different length of training time and different model.

The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



The flowchart of learning algorithm including three stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



The extra stopping criteria (also the learning goals) for two-class classification problems

- Two-class classification problems with $\mathbf{I} \equiv \mathbf{I}_1 \cup \mathbf{I}_2$, where $\mathbf{I}_1$ and $\mathbf{I}_2$ are the sets of indices of given cases in classes 1 and 2. Furthermore, y^c is the target of the c^{th} case, with 1.0 and -1.0 being the targets of classes 1 and 2
- The learning process should stop when

1. $|f(\mathbf{x}^c, \mathbf{w}) - y^c| < \varepsilon \ \forall \ c$

2. $f(\mathbf{x}^c, \mathbf{w}) > \nu \ \forall \ c \in \mathbf{I}_1$ and $f(\mathbf{x}^c, \mathbf{w}) \leq -\nu \ \forall \ c \in \mathbf{I}_2$, with $1 > \nu > 0$

3. $\alpha \equiv \min_{c \epsilon \mathbf{I}_1} f(\mathbf{x}^c, \mathbf{w}) > \beta \equiv \max_{c \epsilon \mathbf{I}_2} f(\mathbf{x}^c, \mathbf{w})$
(Linearly seperating condition, LSC)

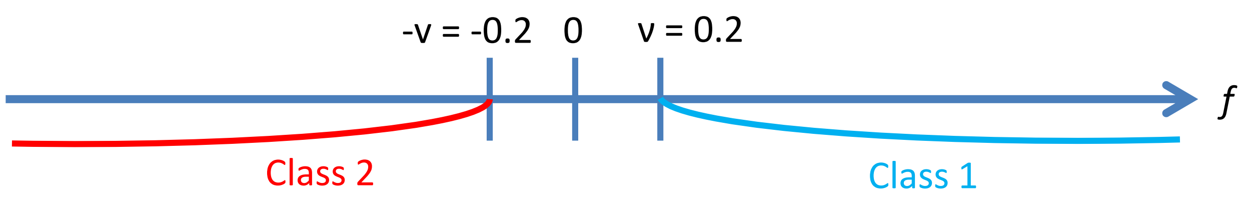
The extra stopping criteria (also the learning goals) for two-class classification problems

- The loss function

$$E(\mathbf{w}) \equiv \frac{1}{N}\sum_{c=1}^{N}(f(\mathbf{x}^c,\mathbf{w}) - y^c)^2$$

- The learning goal is to seek $\mathbf{w}$ where

$$f(\mathbf{x}^c,\mathbf{w}) \geq \nu \ \ \forall\, c \in \mathbf{I}_1 \text{ and } f(\mathbf{x}^c,\mathbf{w}) \leq -\nu \ \ \forall\, c \in \mathbf{I}_2 \text{ with } 1 > \nu > 0.$$



- An alternative learning goal is to seek $\mathbf{w}$ that satisfies the LSC regarding $\{f(\mathbf{x}^c,\mathbf{w}),\ \forall\, c \in \mathbf{I}\}$

Different stopping criterion results in different length of training time and different model.

Literature Review

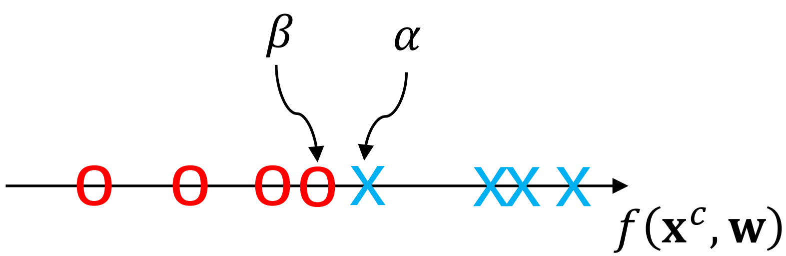
LSC

(Tsaih, 1993)

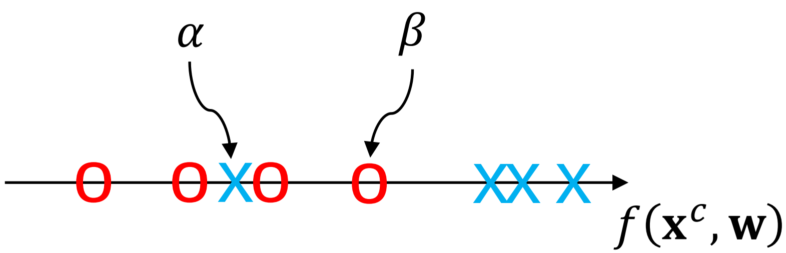
$y^c = 1 \ \forall\ c \in \mathbf{I}_1;\ y^c = -1 \ \forall\ c \in \mathbf{I}_2$

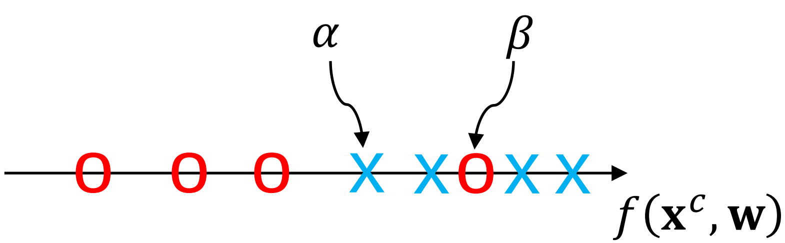
X : $f(\mathbf{x}^c, \mathbf{w}),\ \forall\ c \in \mathbf{I}_1$

O : $f(\mathbf{x}^c, \mathbf{w}),\ \forall\ c \in \mathbf{I}_2$



$\alpha > \beta$
LSC : True





$\alpha < \beta$
LSC: False

When LSC $(\alpha > \beta)$ is satisfied

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

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

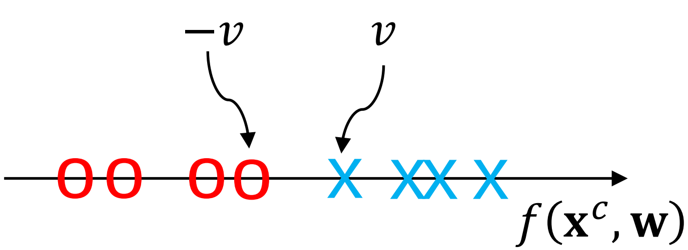
can be set by directly adjusting $\mathbf{w}^o$ according to the following formula:

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

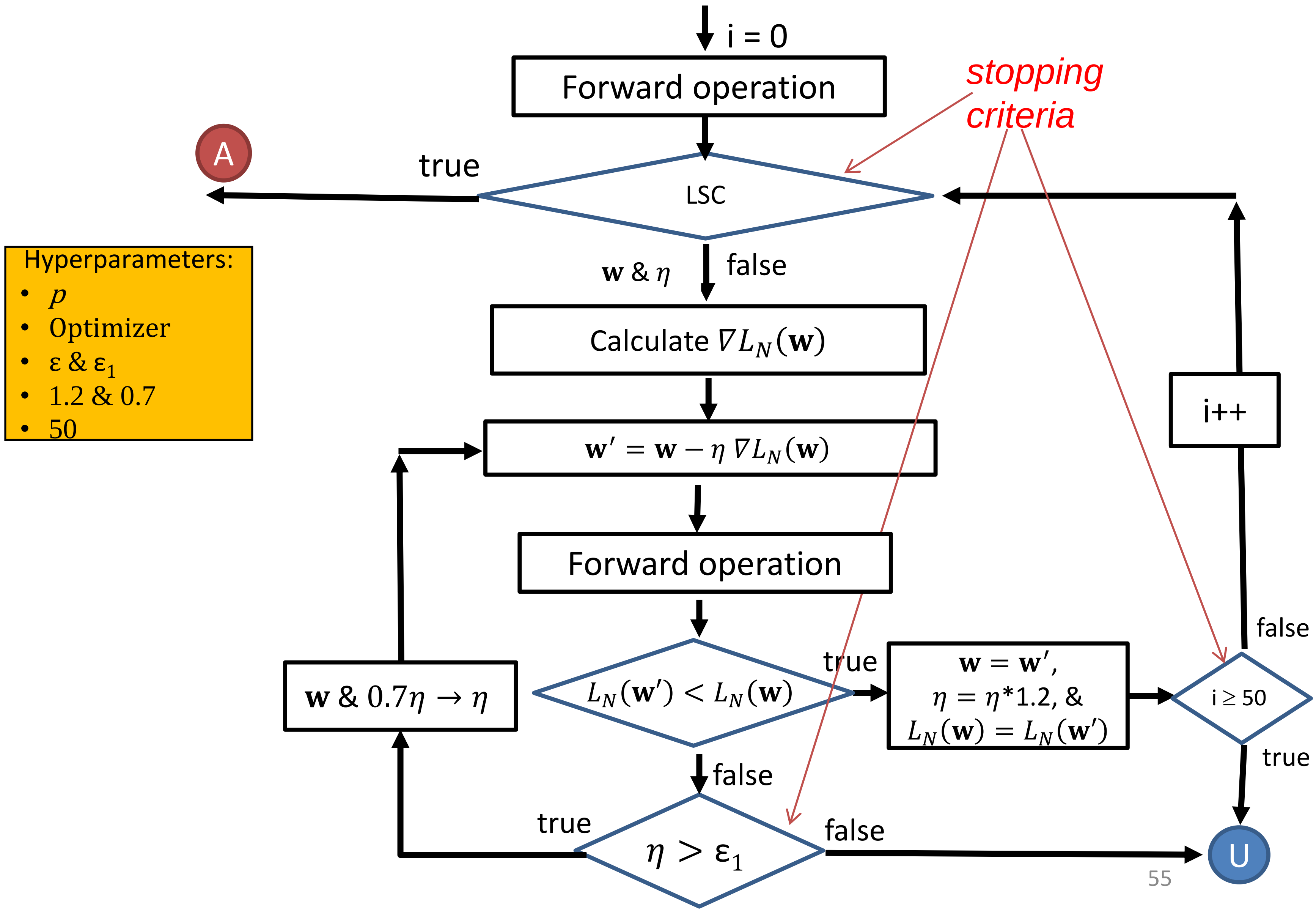
The weight vector between the hidden layer and the output node

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

The threshold of the output node



The flowchart of learning algorithm including three stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



The inferencing mechanism needs to match with the learning goal

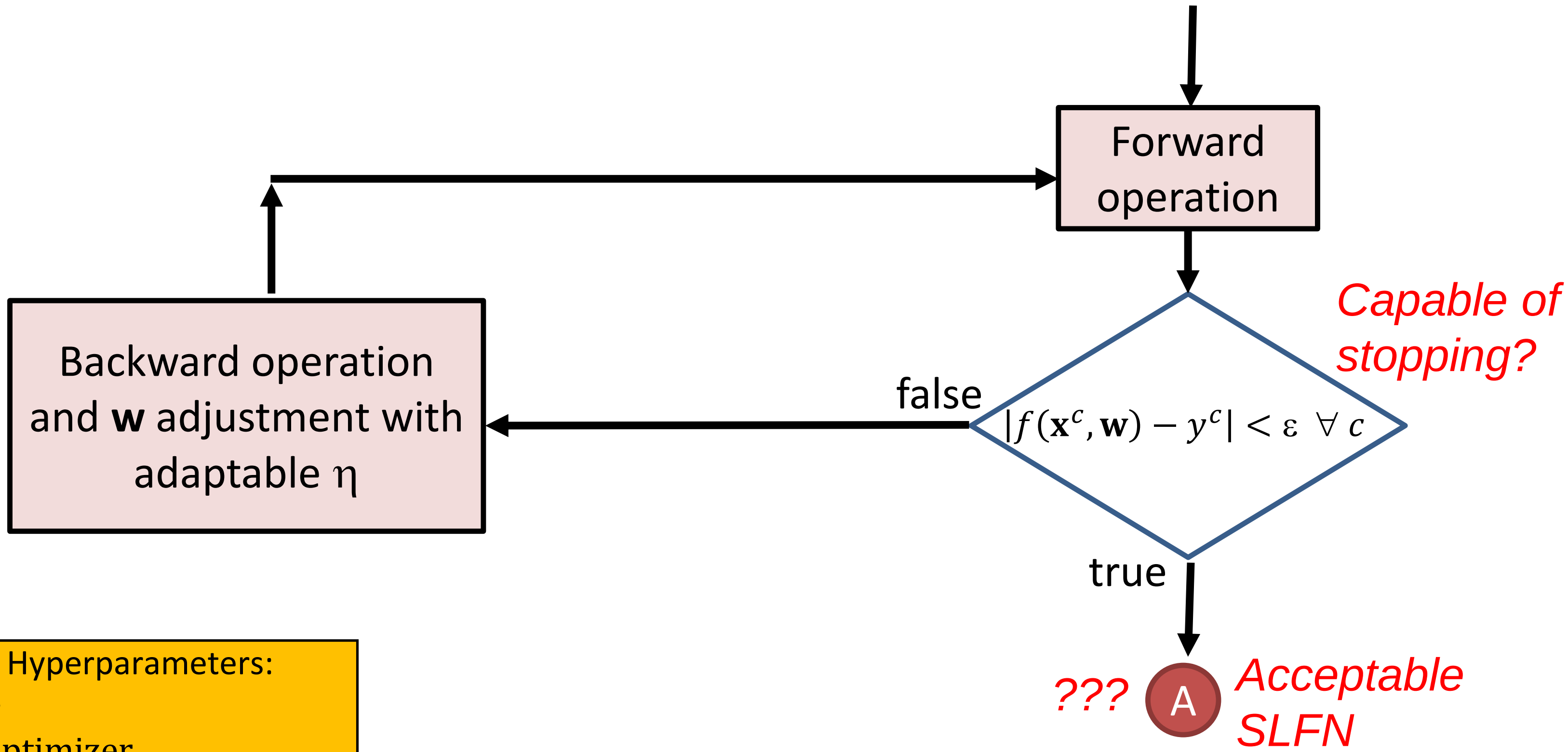
- Two-class Classification problems with $\mathbf{I} \equiv \mathbf{I}_1 \cup \mathbf{I}_2$, where $\mathbf{I}_1$ and $\mathbf{I}_2$ are the sets of indices of given cases in classes 1 and 2. Furthermore, y^c is the target of the c^{th} case, with 1.0 and -1.0 being the targets of classes 1 and 2
- The learning goal:

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

- The inferencing mechanism:

$f(\mathbf{x}^c, \mathbf{w}) > \nu \; \forall \; c \in \mathbf{I}_1$ and $f(\mathbf{x}^c, \mathbf{w}) \leq -\nu \; \forall \; c \in \mathbf{I}_2$, with $1 > \nu > 0$

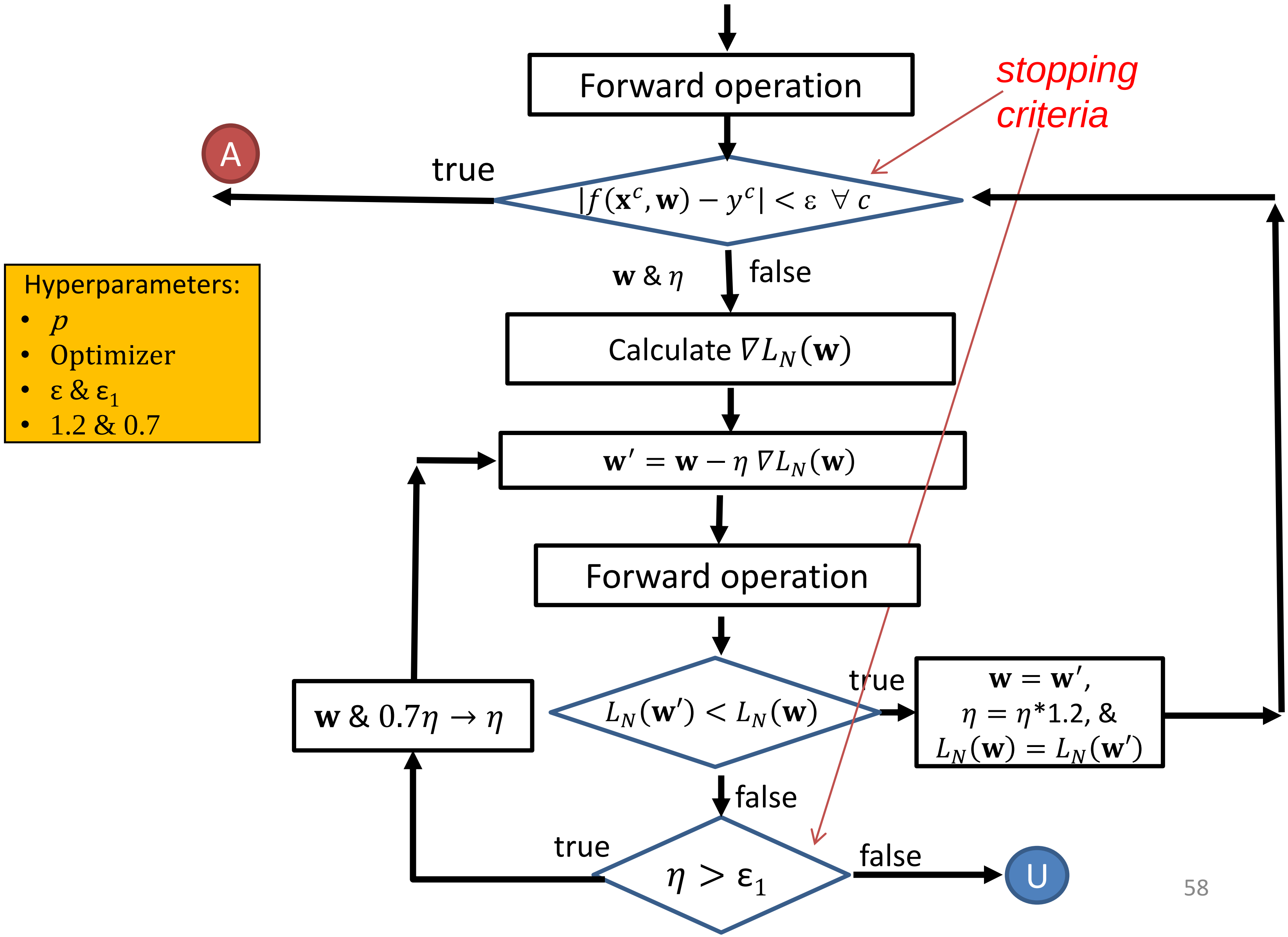
The flowchart of BP learning algorithm with the adaptable learning rate module



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

Q: Can this the learning process stop through satisfying the stopping criterion?
A: Maybe! Thus, this stopping criterion is not good and thus this algorithm is not good.

The flowchart of learning algorithm including two stopping criteria that indicate either an unacceptable SLFN or an acceptable SLFN



Homework #2

- Refer to page 49 to rewrite the code you have for HW #1 (refer to page 16).
- Refer to page 50 to rewrite the code you have for HW #1.
- Refer to page 58 to rewrite the code the code you have for HW #1.
- Once you have the code (regardless of which framework you choose above), you will apply the code to learn the train_all_0.csv dataset given in the LINE group.
- The training and test dataset is 80%/20%.
- The performance comparison benchmark is the code the code you have for HW #1.