数据挖掘导论

Introduction to Data Mining

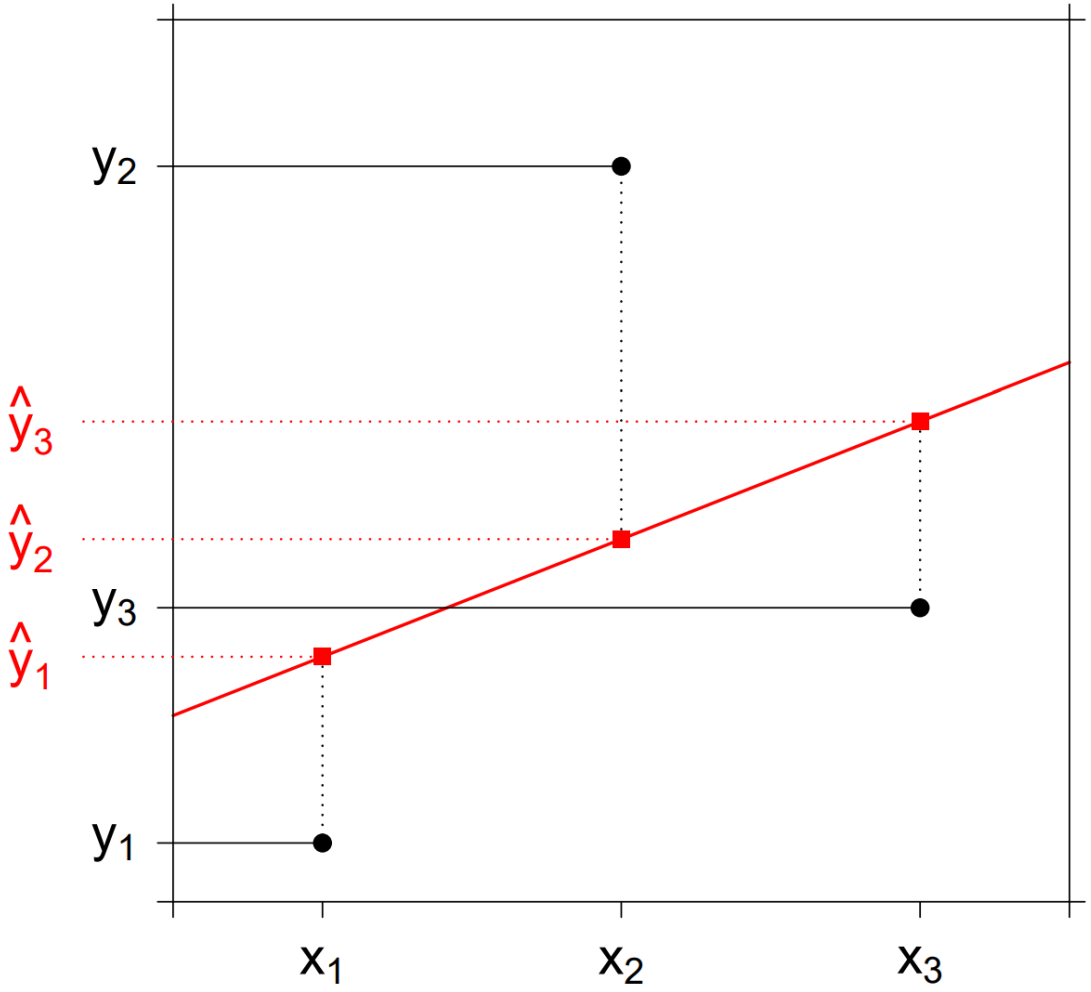
Advanced Classification

数据智能实验室
DATA INTELLIGENCE LABORATORY
DI-LAB
浙江大学
Zhejiang University

Agenda

- Linear Regression
- Logistic Regression
- Model Ensemble
- Semi-Supervised Classification

Linear Regression



- observed points:

$$(x_i, y_i)$$

- points on the line:

$$(x_i, \hat{y}_i)$$

Image source: http://hs-stat.com/courses/BUS274/slides/bus274_ch15_v2015-04-26.pdf

The Method of Least Square

- Define $\hat{y}_i = a + bx_i$ and $e_i = y_i - \hat{y}_i$.
- a, b such that the sum of squared distances is minimized:

$$Q(a,b) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

- The line $y = a + bx$ with these parameters a and b is called the regression line of Y with respect to X.

Image source: http://hs-stat.com/courses/BUS274/slides/bus274_ch15_v2015-04-26.pdf

The Method of Least Square

Computing the regression line.

Minimizing Q leads to the following equations for the slope b and the intercept a:

$$b = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

$$a = \bar{y} - b\bar{x}.$$

Image source: http://hs-stat.com/courses/BUS274/slides/bus274_ch15_v2015-04-26.pdf

The Method of Least Square

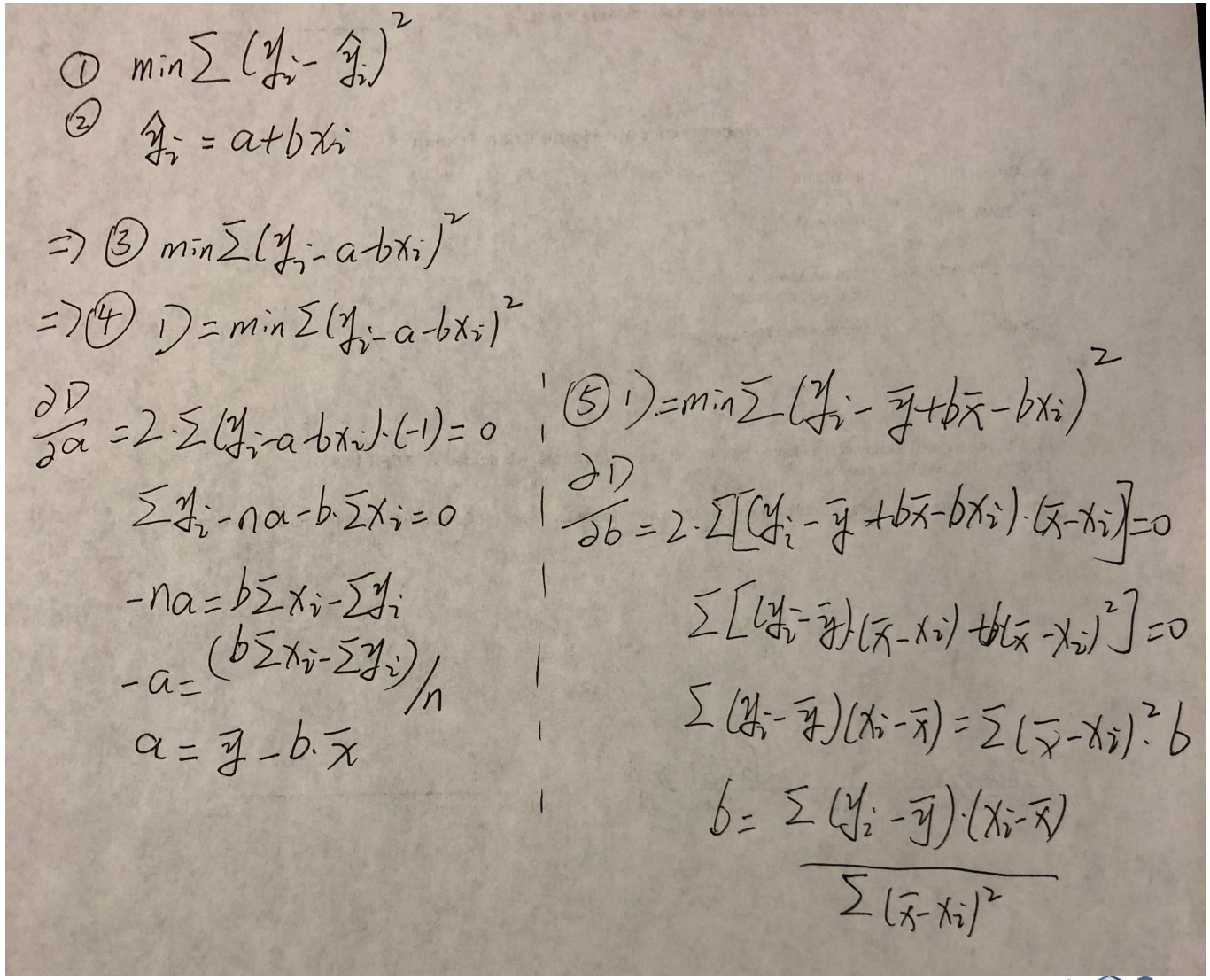


Image source: https://blog.csdn.net/YtdxYHZ/article/details/51703445

The Method of Least Square

i	x_i	y_i	x_i^2	y_i^2	$x_i y_i$	$\hat{y}_i$
1	5	15	25	225	75	13.9
2	10	8	100	64	80	11.3
3	15	12	225	144	180	8.7
4	20	5	400	25	100	6.1
$\sum$	50	40	750	458	435	40

$$b = \frac{4 \cdot 435 - 50 \cdot 40}{4 \cdot 750 - 50^2} = -0.52$$

$$a = \frac{40}{4} - (-0.52) \cdot \frac{50}{4} = 16.5$$

The regression line is: $y = 16.5 - 0.52x$

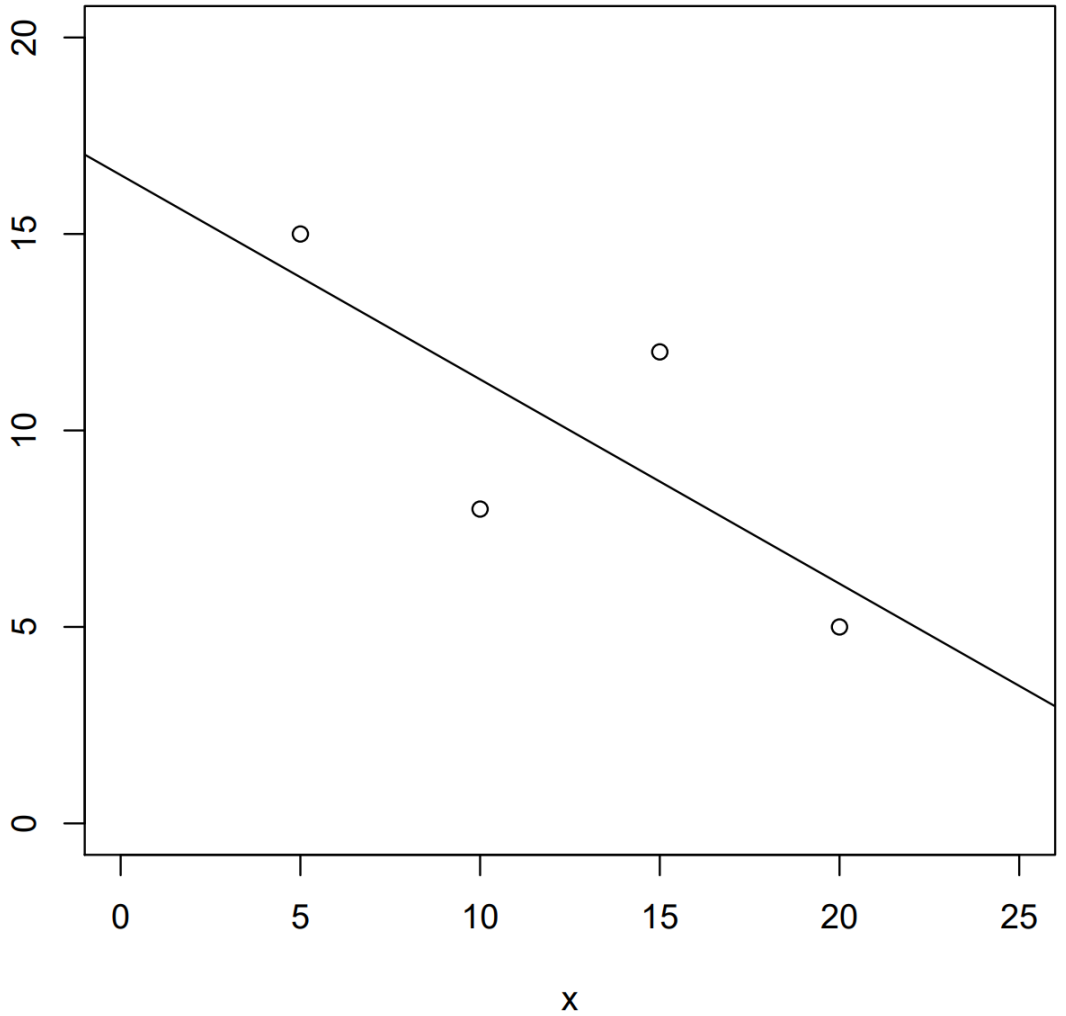
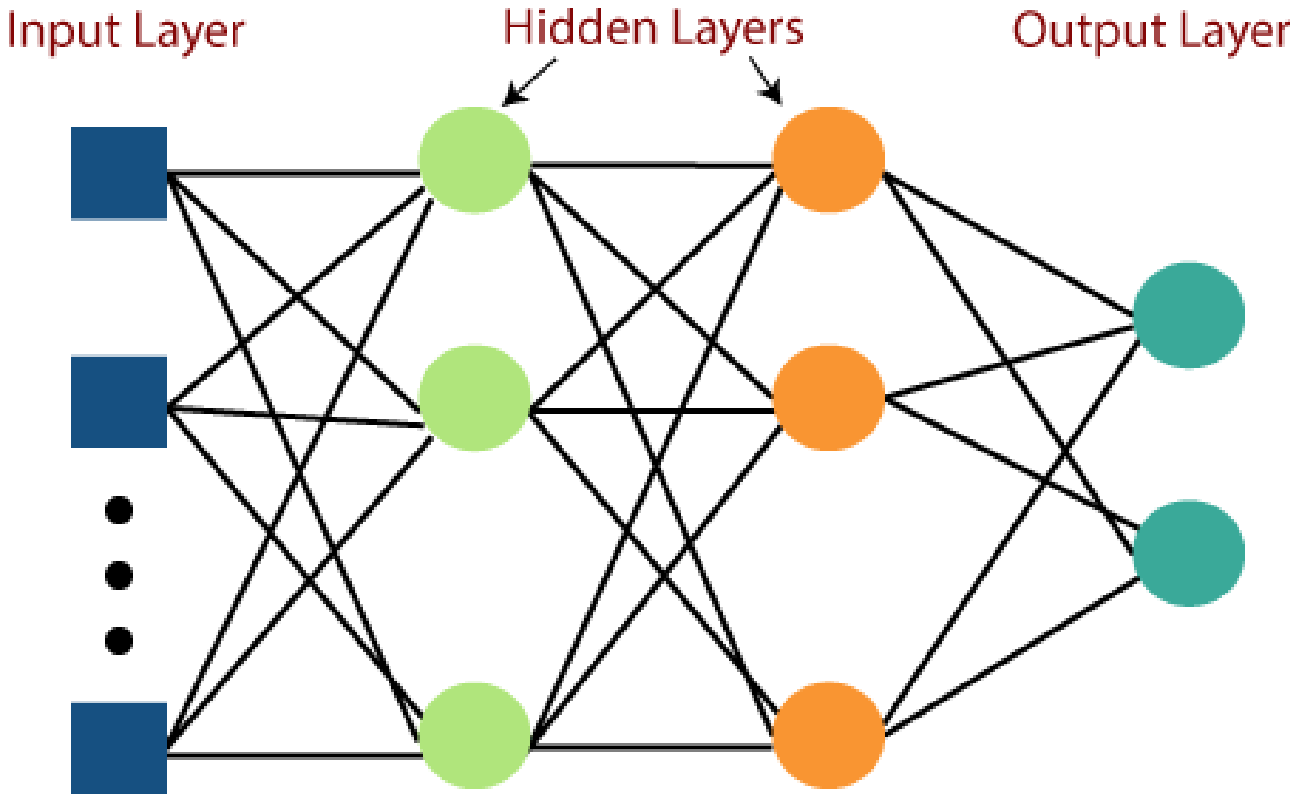


Image source: http://hs-stat.com/courses/BUS274/slides/bus274_ch15_v2015-04-26.pdf

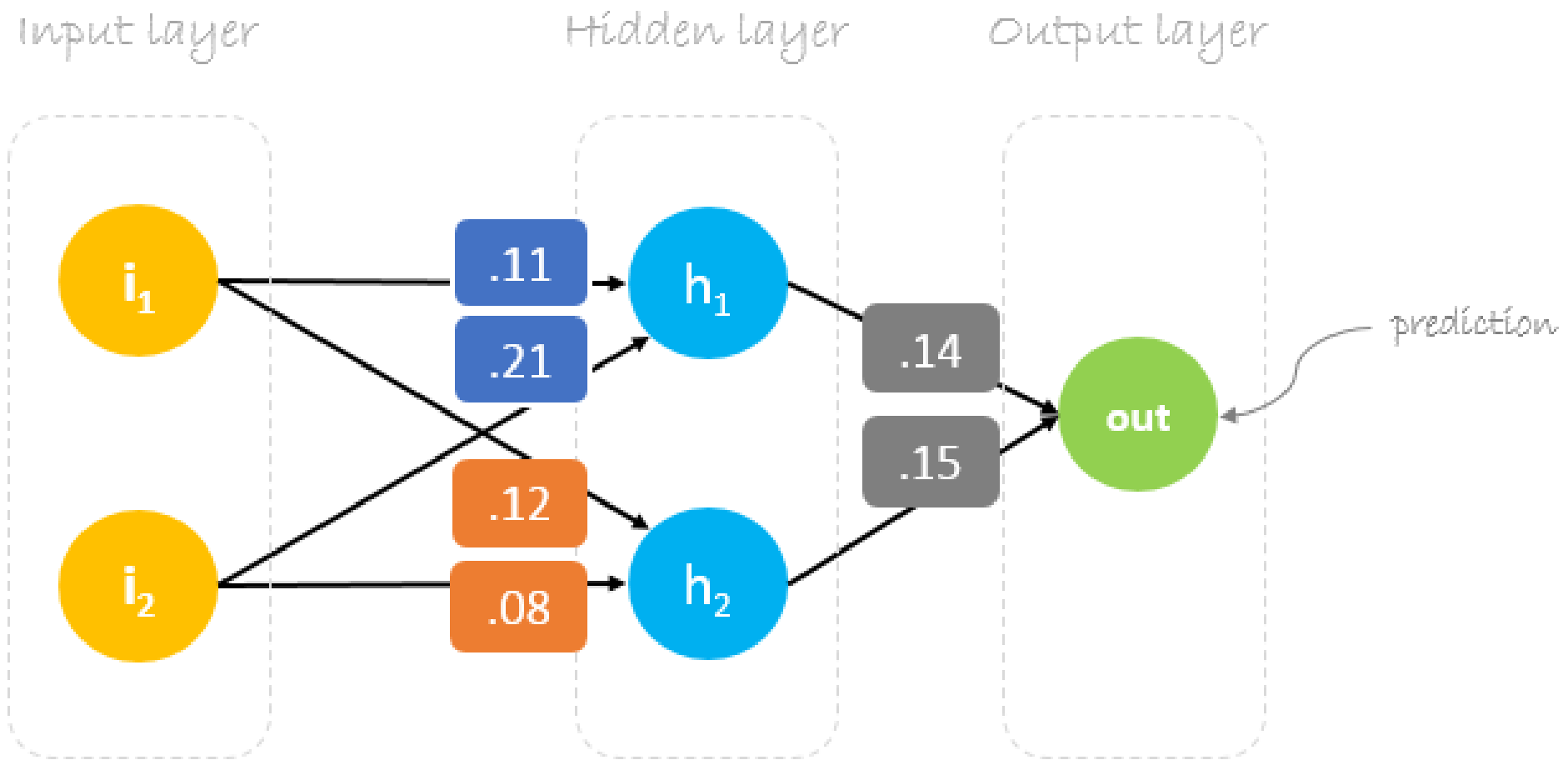
Agenda

- Linear Regression
- Logistic Regression
- Model Ensemble
- Semi-Supervised Classification

Hidden Layer

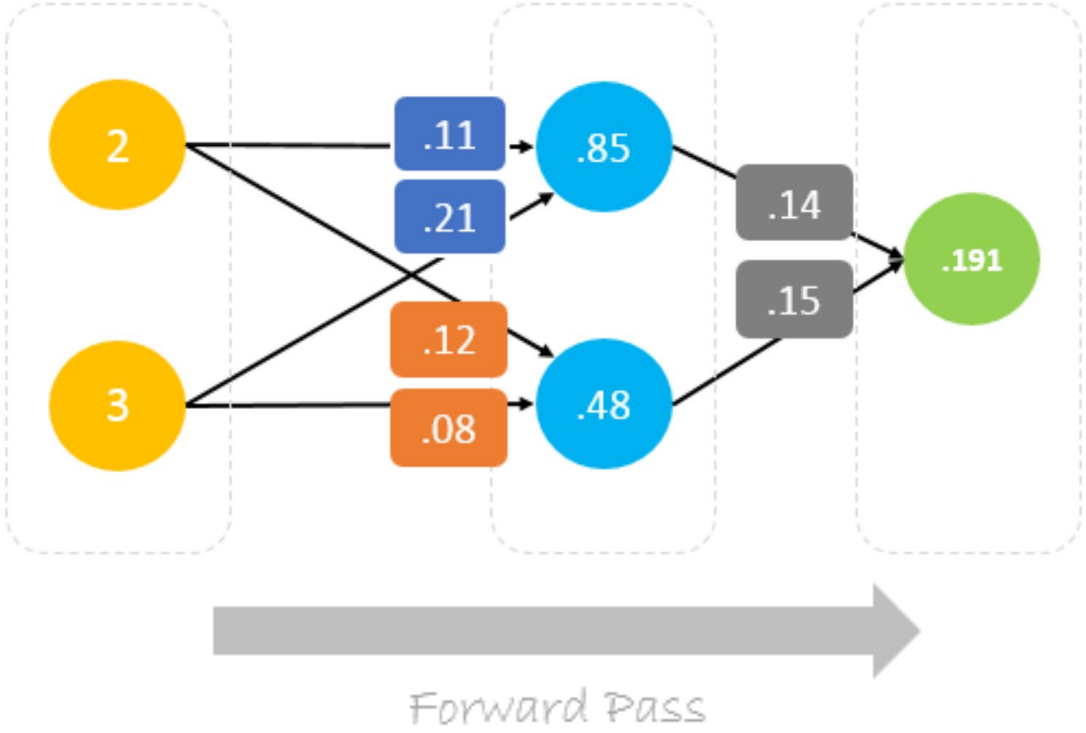


Hidden Layer



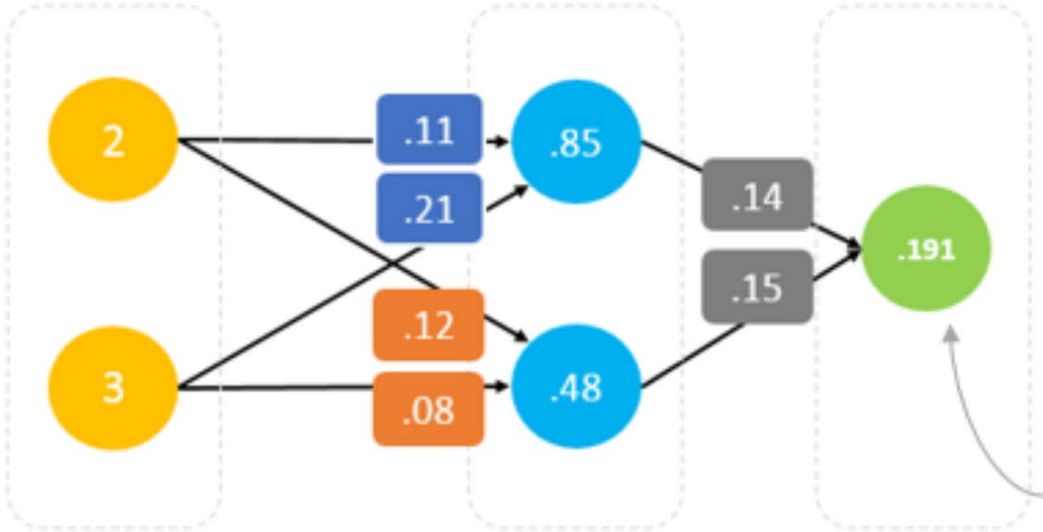
Example from: https://hmkcode.com/ai/backpropagation-step-by-step/

FeedForward Process



$$[2 \quad 3] \cdot \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} = [0.85 \quad 0.48] \cdot \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} = [0.191]$$

Error/Loss Calculation



Input

Actual output

prediction

actual

Error = 0 , if prediction = actual

$$\text{Error} = \frac{1}{2}(\text{prediction} - \text{actual})^2$$

Error is always positive because of the square

$\frac{1}{2}$ is added to ease the calculation of the derivative

$$\text{Error} = \frac{1}{2}(0.191 - 1.0)^2 = 0.327$$

浙江大學 Zhejiang University

Error/Loss Calculation

- How to update the weights value so that the error is reduced?

$$\text{prediction} = \text{out}$$

$$\text{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$$h_1 = i_1 w_1 + i_2 w_2$$

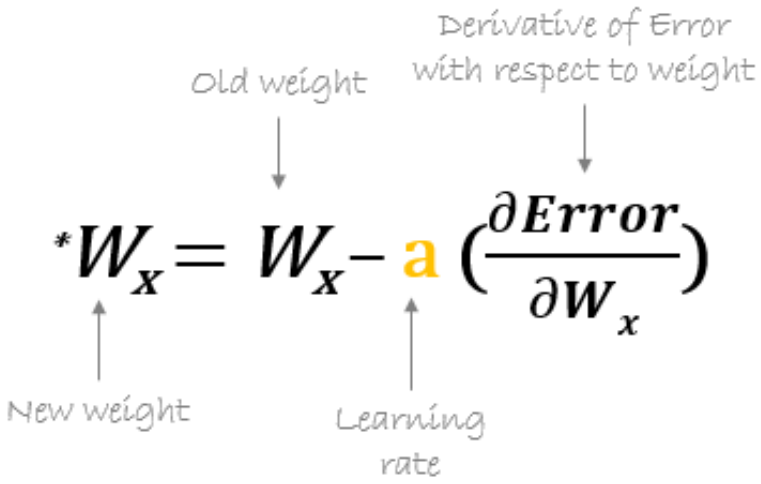
$$h_2 = i_1 w_3 + i_2 w_4$$

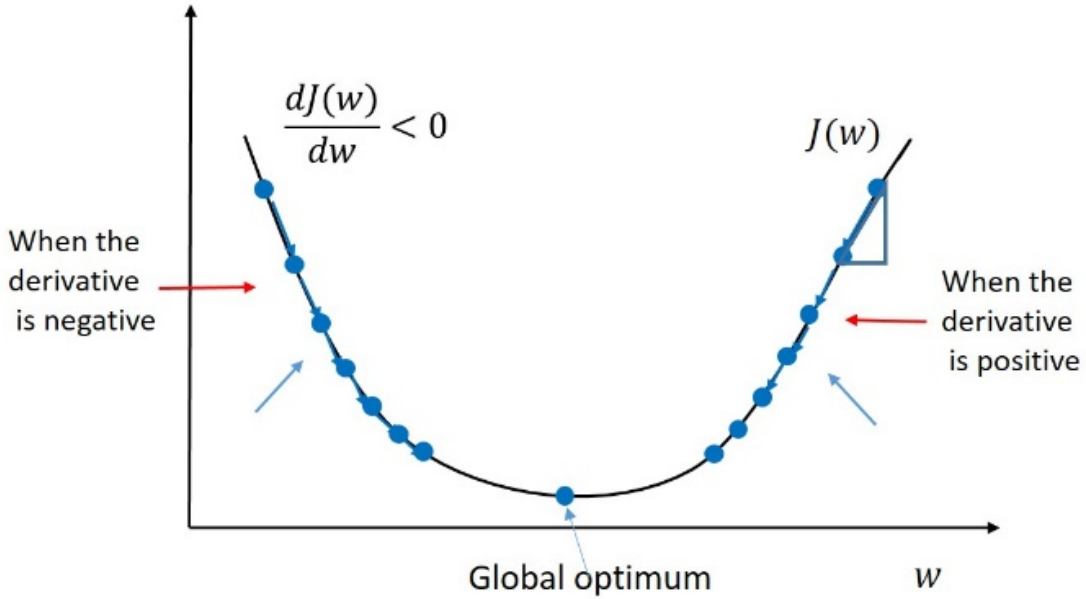
$$\text{prediction} = (i_1 w_1 + i_2 w_2)\, w_5 + (i_1 w_3 + i_2 w_4)\, w_6$$

to change ***prediction*** value,
we need to change ***weights***

Back Propagation

- **Gradient descent is an iterative optimization algorithm for finding the minimum of a function**
  - ✓ In our case, we want to minimize the error function.
  - ✓ To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.





Back Propagation

$$^{*}W_6 = W_6 - a\left(\frac{\partial Error}{\partial W_6}\right)$$

$$\frac{\partial \text{ Error}}{\partial W_6} = \frac{\partial \text{ Error}}{\partial \text{ prediction}} * \frac{\partial \text{ prediction}}{\partial W_6}$$

Chain rule

$$Error = \frac{1}{2}(\text{ prediction } - \text{ actual })^2$$

$$\text{prediction } = (i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6$$

$$\frac{\partial \text{ Error}}{\partial W_6} = \frac{\partial(\frac{1}{2}(\text{ predictoin} - \text{actual })^2)}{\partial \text{ prediciton}} * \frac{\partial((i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6)}{\partial W_6}$$

$$\frac{\partial Error}{\partial W_6} = 2 * \frac{1}{2}(\textbf{ predictoin } - \textbf{ actual}) \frac{\partial(\text{ predictoin} - \text{actual })}{\partial \text{ prediciton}} * (i_1 w_3 + i_2 w_4)$$

$h_2 = i_1 w_3 + i_2 w_4$

$$\frac{\partial Error}{\partial W_6} = (\textbf{ predictoin } - \textbf{ actual })^{*}(h_2)$$

$$\frac{\partial \text{ Error}}{\partial W_6} = \Delta h_2$$

$\Delta =$ prediction $-$ actual ← **delta**

Back Propagation

updated weights

$$^{*}w_6 = w_6 - a\,(h_2 \,.\, \Delta)$$
$$^{*}w_5 = w_5 - a\,(h_1 \,.\, \Delta)$$
$$^{*}w_4 = w_4 - a\,(i_2 \,.\, \Delta w_6)$$
$$^{*}w_3 = w_3 - a\,(i_1 \,.\, \Delta w_6)$$
$$^{*}w_2 = w_2 - a\,(i_2 \,.\, \Delta w_5)$$
$$^{*}w_1 = w_1 - a\,(i_1 \,.\, \Delta w_5)$$

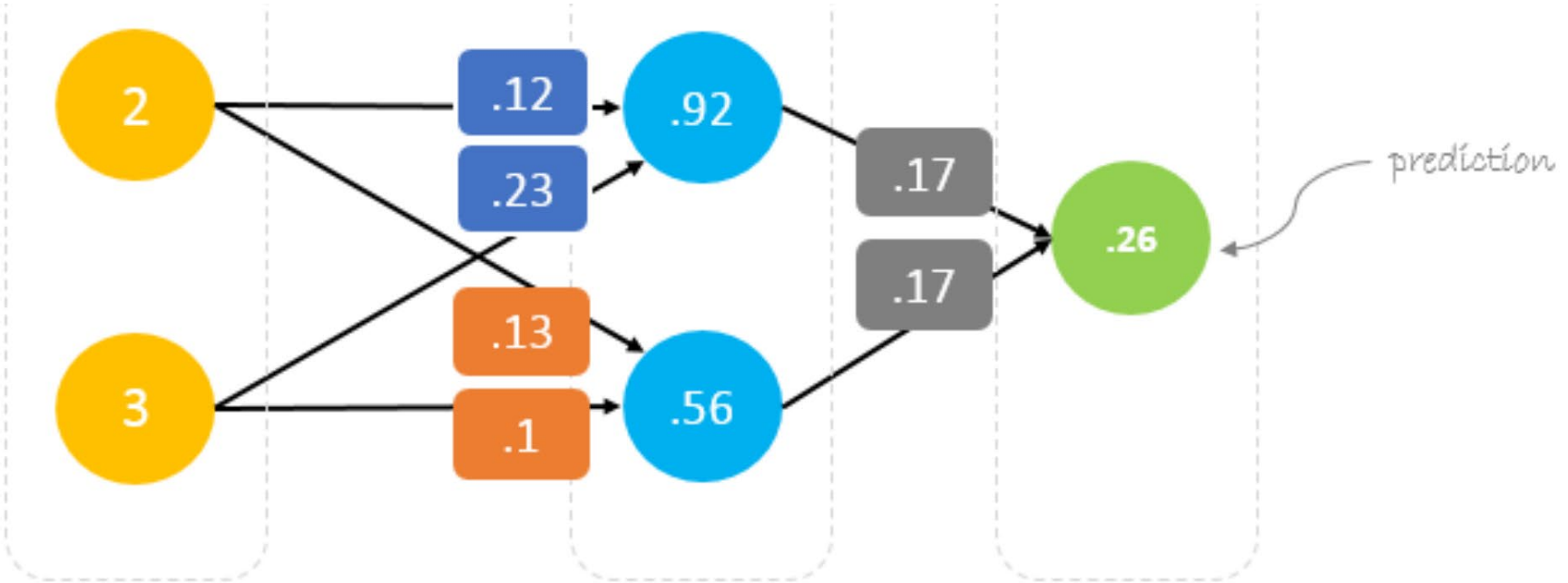
$\Delta = 0.191 - 1 = -0.809$ ← **Delta** = prediction - actual

$a = 0.05$ ← **Learning rate,** we smartly guess this number

$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809)\begin{bmatrix} 0.85 \\ 0.48 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - \begin{bmatrix} -0.034 \\ -0.019 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - 0.05(-0.809)\begin{bmatrix} 2 \\ 3 \end{bmatrix}.[0.14 \quad 0.15] = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - \begin{bmatrix} -0.011 & -0.012 \\ -0.017 & -0.018 \end{bmatrix} = \begin{bmatrix} .12 & .13 \\ .23 & .10 \end{bmatrix}$$

FeedForward Process



Forward Pass

Matrix multiplication

$$[2 \quad 3] \cdot \begin{bmatrix} 0.12 & 0.13 \\ 0.23 & 0.10 \end{bmatrix} = [0.92 \quad 0.56] \cdot \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix} = [0.26]$$

Details

2 x .12 + 3 x .23 = .85

2 x .13 + 3 x .10 = .48

.92 x .17 + .56 x .17 = .26

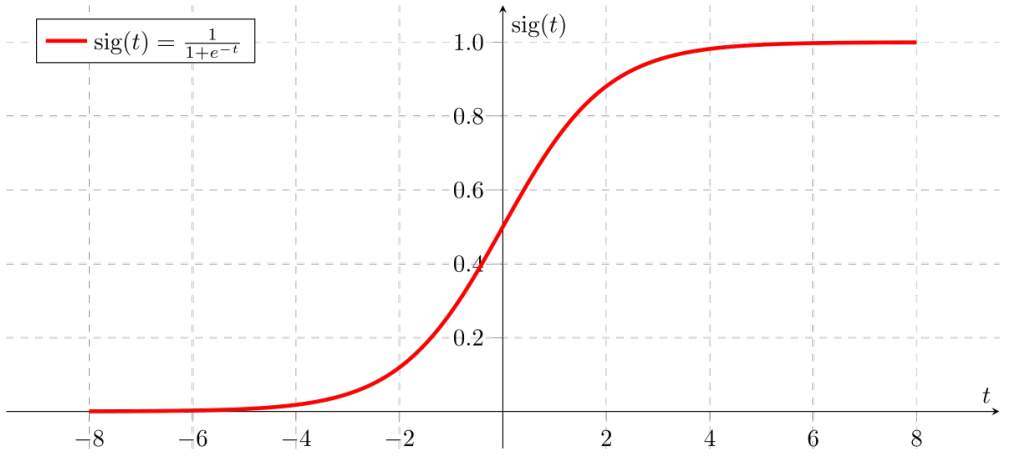
Logistic Regression

- **Name is kind of misleading**
  - ✓ It is in fact used for classification, not regression
  - ✓ A multi-dimensional feature space (features can be categorical or continuous)
  - ✓ Outcome is the probability to a class
- **Idea of the Model**
  - ✓ For a point **x** in the feature space, project it onto β to convert it into a real number z

$$z = \alpha + \boldsymbol{\beta} \cdot \mathbf{x} = \alpha + \beta_1 x_1 + \cdots + \beta_d x_d$$

  - ✓ Map z to the range 0 and 1 using the logistic function

$$p = 1/(1+e^{-z})$$



- **It can viewed as a one-layer neural network**

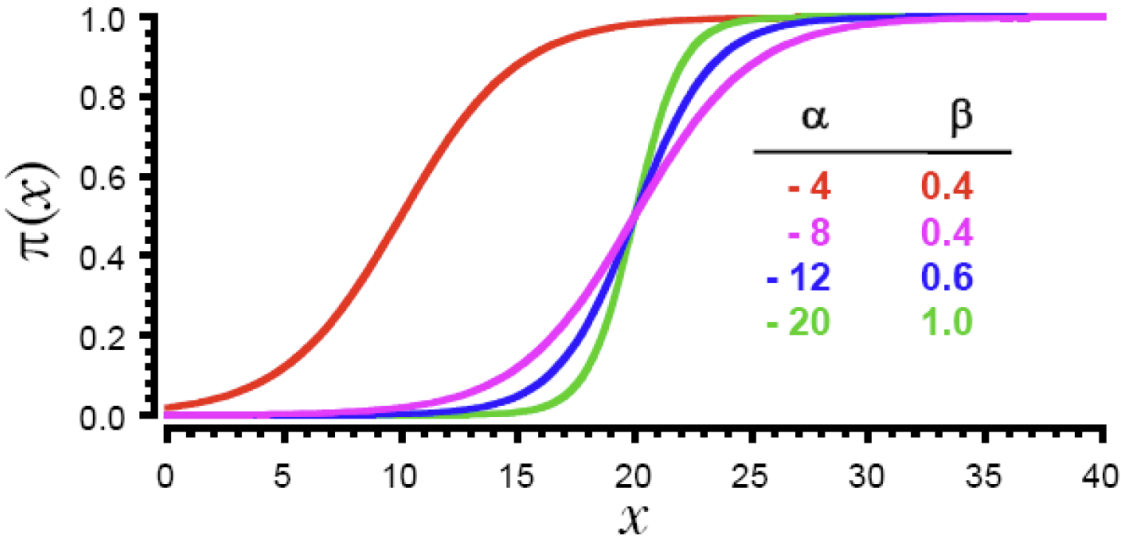
Reference: http://courses.washington.edu/css490/2012.Winter/lecture_slides/05b_logistic_regression.pdf

Logistic Regression

Parameters control shape and location of sigmoid curve

- ✓ α controls location of midpoint
- ✓ β controls slope of rise

$$\pi(x) = \exp(\alpha + \beta x) / (1 + \exp(\alpha + \beta x))$$



Model Training

- ✓ Usually done by numerical approximation of maximum likelihood
- ✓ On really large datasets, may use stochastic gradient descent

Agenda

- ☐ Linear Regression
- ☐ Logistic Regression
- ☐ Model Ensemble
- ☐ Semi-Supervised Classification

Ensemble: Bagging

- Decision trees are highly **interpretable** and **fast to train**
- However, in order to capture a complex decision boundary (or to approximate a complex function), we need to use a **large tree**, which is **prone to overfitting**

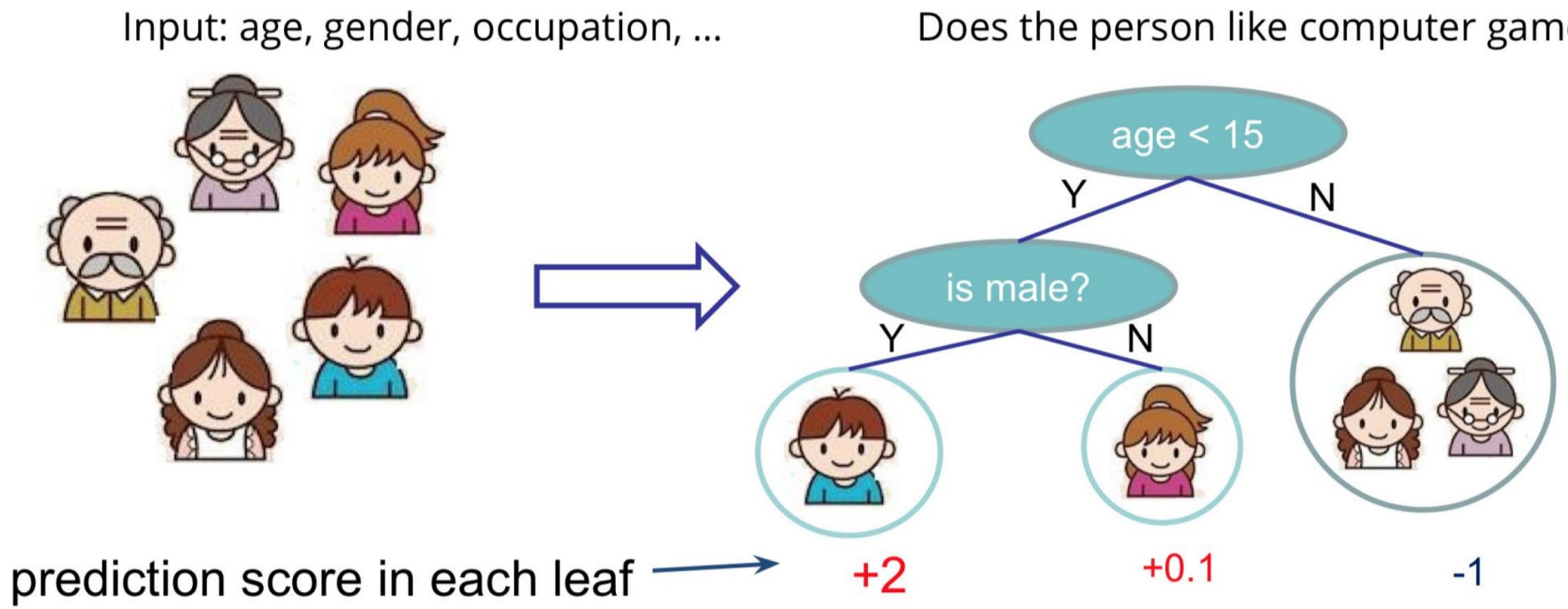
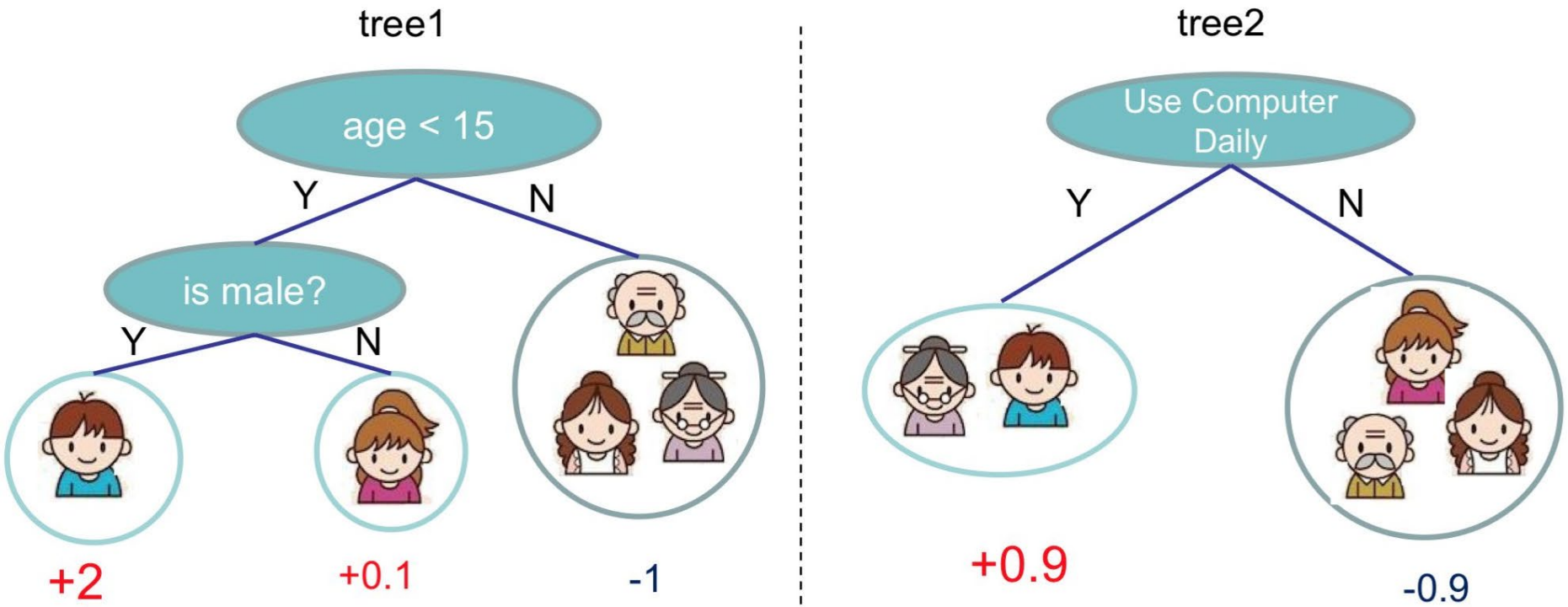


Image from: https://www.cs.toronto.edu/~rgrosse/courses/csc2515_2019/tutorials/tut4/tut4_boost.pdf

Random Forest

- Average multiple decision trees



f() = 2 + 0.9= 2.9 f()= -1 - 0.9= -1.9

Random Forest

Construction

Image source: https://easyai.tech/ai-definition/random-forest/

Ensemble: Boosting

- **Random Forest is a bagging approach that works on strong classifiers**
- **Work on weak classifiers with the following guarantee:**
  - ✓ If your ML model can produce error rate smaller than 50% on the training data, you can obtain 0% error rate classifier after boosting
- **Framework of Boosting: The classifiers are learned sequentially**
  - ✓ Obtain the first classifier F1(x)
  - ✓ Find another function F2(x) to help F1(x)
    - If F2(x) is similar to F1(x), it will not help much
    - How to make F2(x) complementary with F1(x)?
  - Find another function F3(x) to help F2(x)
  - …
  - Finally, combining all the classifiers

Slide reference: Hungyi-Lee

AdaBoost

- **Idea: training F2(x) on the new training set that fails F1(x)**
  - ✓ Pick the samples that F1(x) predicts wrong results

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \varepsilon_1 < 0.5$$

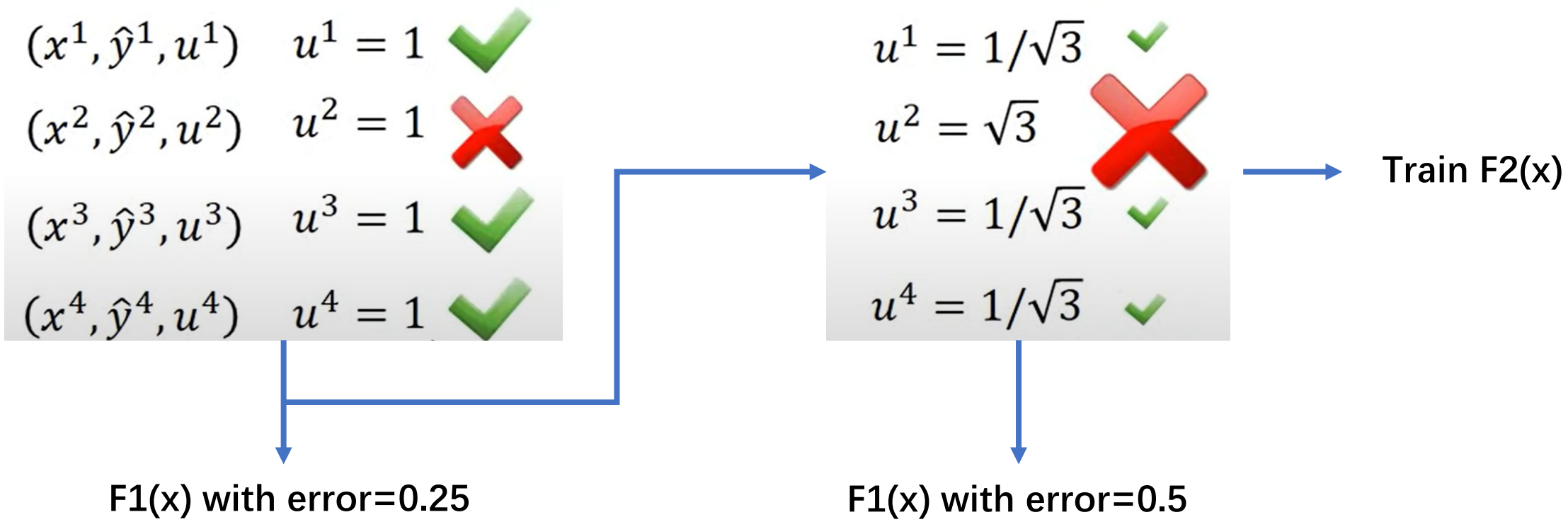
  - ✓ Update the weights of these samples

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

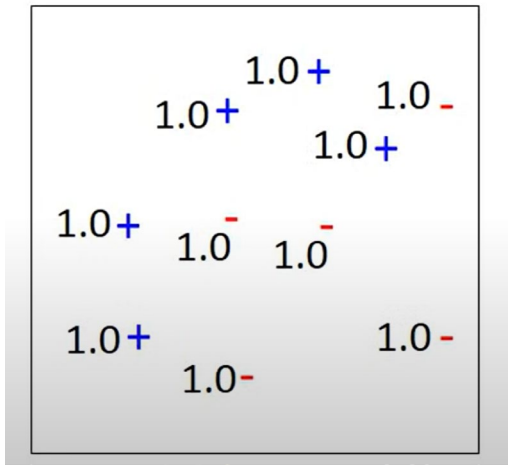
$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

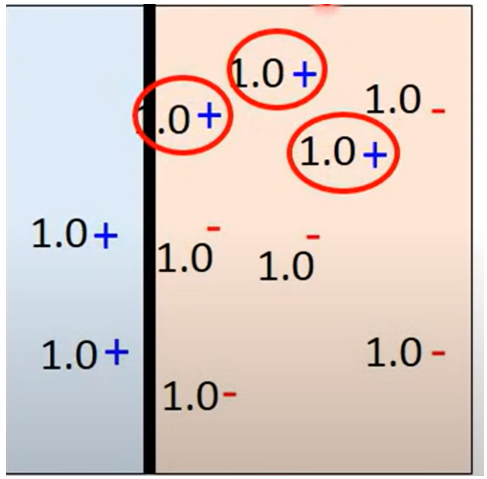
$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

Example for Re-weighting Training Data

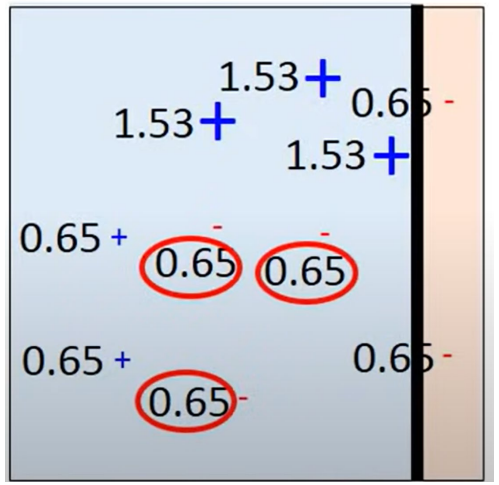


Toy Example for AdaBoost

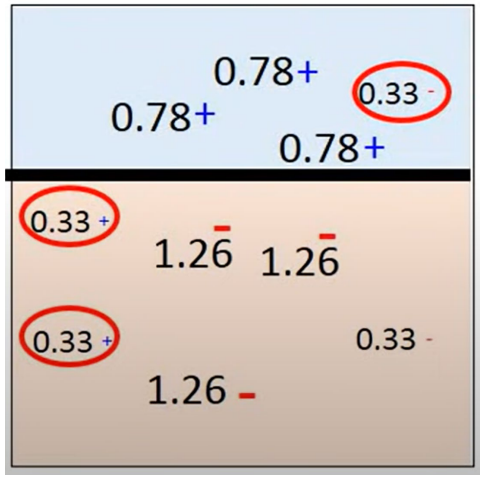




F1(x): Error Rate=0.3



F2(X): Error Rate=0.21

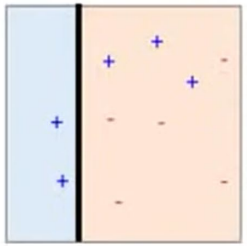
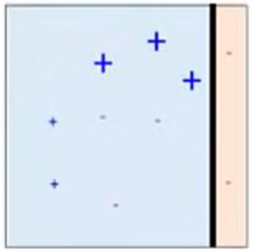
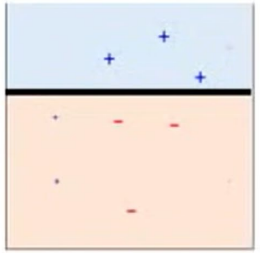


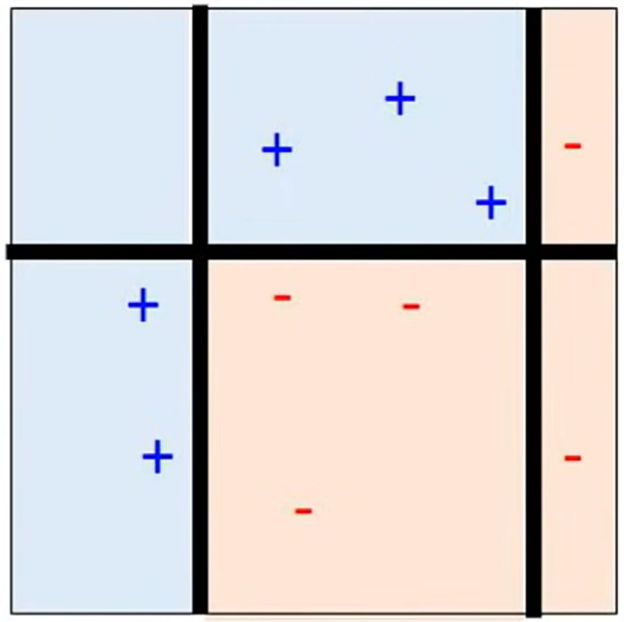
F3(X): Error Rate=0.13

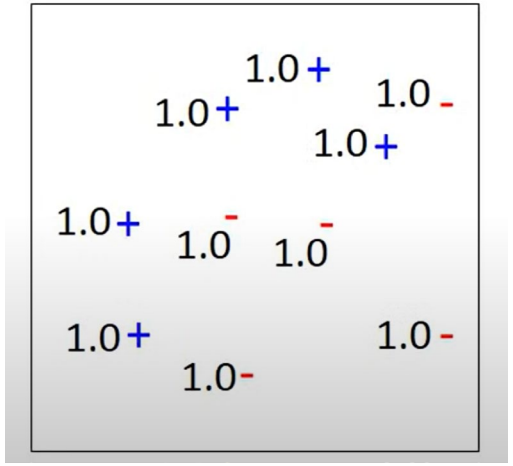
Toy Example for AdaBoost

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$$

$sign($ 0.42  + 0.66  + 0.95  $)$



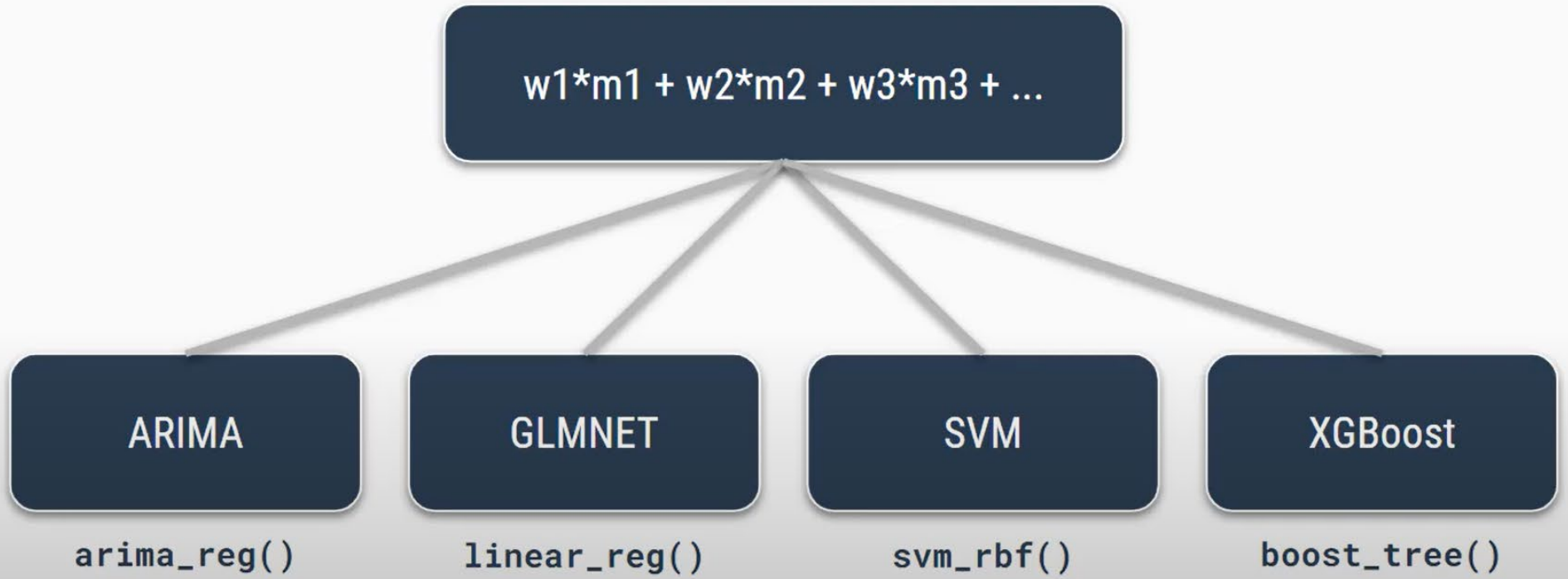


Ensemble: Stacking

Level 2:
Weighted Stack

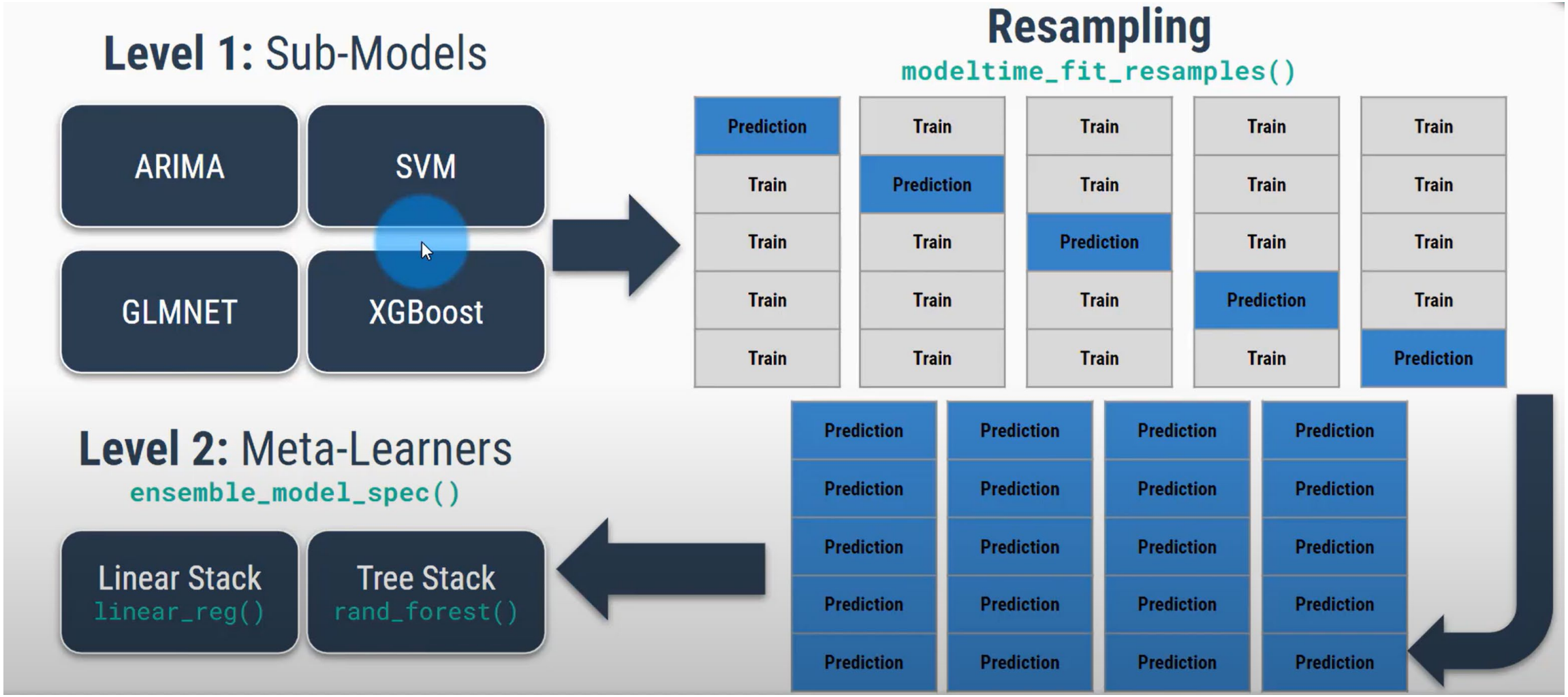
```
ensemble_weighted()
ensemble_average()
```

Level 1:
Sub-Models



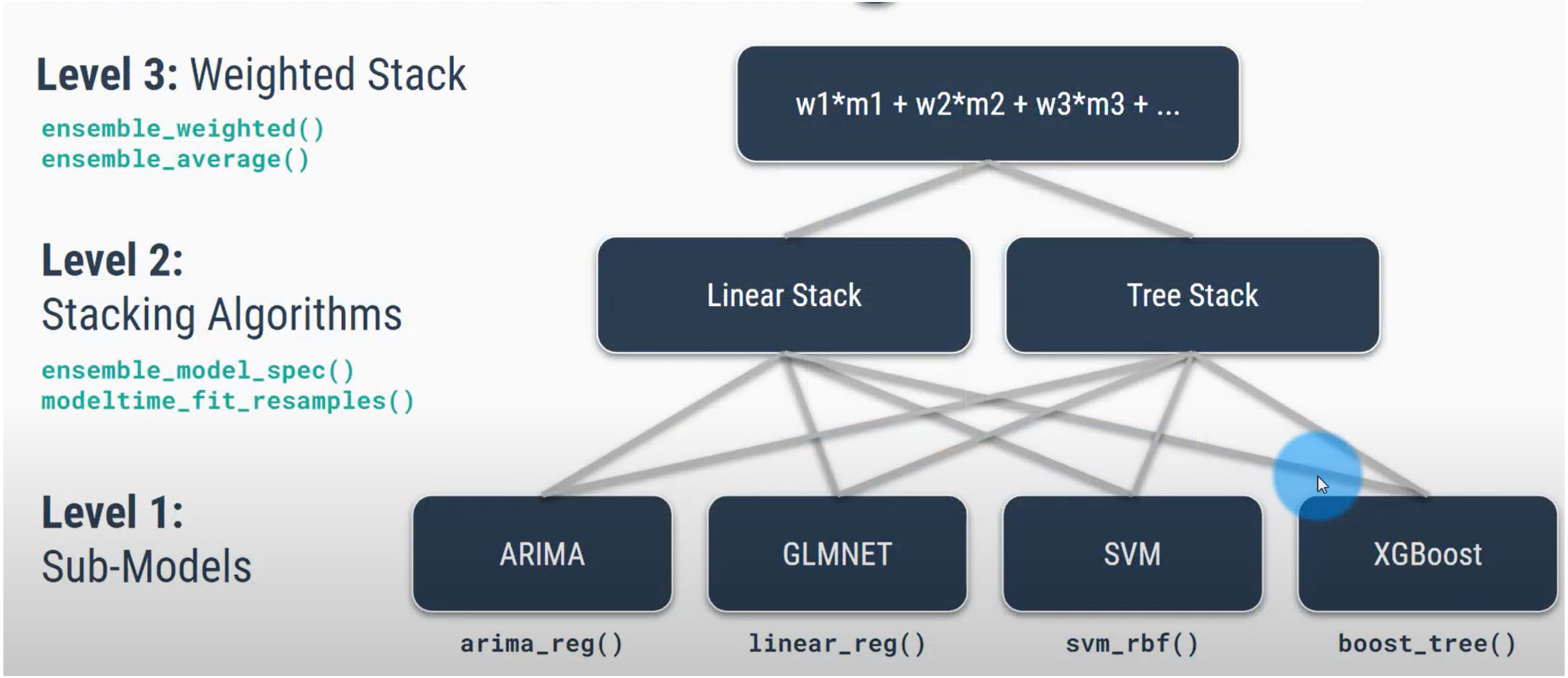
参考：https://www.youtube.com/watch?v=6YdYMvclOV0

Ensemble: Stacking



参考：https://www.youtube.com/watch?v=6YdYMvclOV0

Ensemble: Stacking



参考：https://www.youtube.com/watch?v=6YdYMvclOV0

Last Year's Project

Number	Model Setting	Score
0	ridge+lasso+elasticnet auto wieght	0.15927
1	gbr+lightgbm+xgboost auto wieght	0.12102
2	7_model(without svm) auto wieght	0.12630
3	7_model(without svm) modified wieght	0.12188
4	8_model auto wieght	0.12348
5	ridge+gbr+lightgbm+xgboost auto wieght	0.12062
6	ridge+lasso+gbr+lightgbm+xgboost auto wieght	0.12695
7	ridge+gbr+lightgbm+xgboost+stack_gen auto wieght	0.12035
8	ridge+gbr+lightgbm+xgboost+stack_gen modified wieght	**0.11983**

Agenda

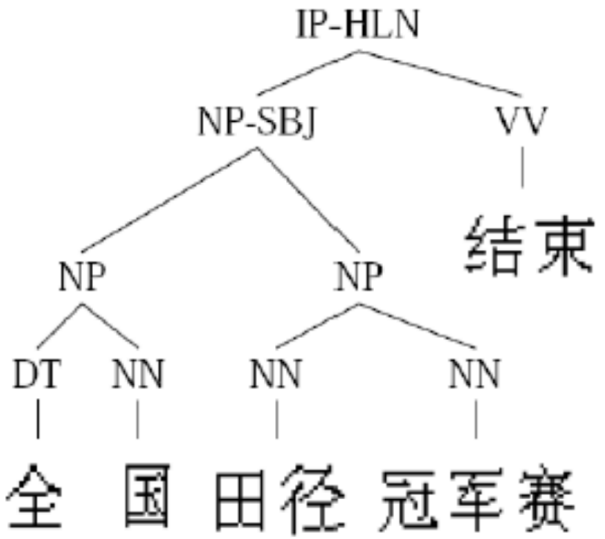
- ☐ Linear Regression
- ☐ Logistic Regression
- ☐ Model Ensemble
- ☐ Semi-Supervised Classification

Why Semi-Supervised Classification

- Label data can be expensive to collect
- Unlabeled data are cheap and abundant
- Semi-supervised learning exploits both labeled and unlabeled data

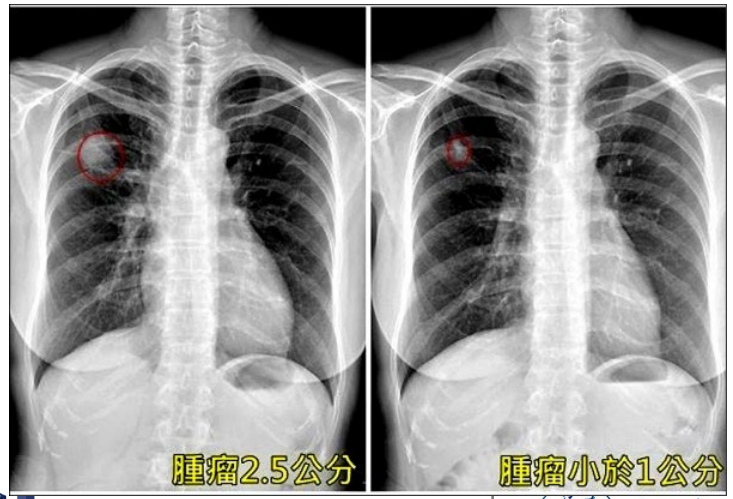
Task: natural language parsing

- Penn Chinese Treebank
- 2 years for 4000 sentences



Task: speech analysis

- Switchboard dataset
- telephone conversation transcription
- 400 hours annotation time for each hour of speech



Self-Training

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
2. Repeat:
3. Train f from L using supervised learning.
4. Apply f to the unlabeled instances in U.
5. Remove a subset S from U; add $\{(\mathbf{x}, f(\mathbf{x}))|\mathbf{x} \in S\}$ to L.

Self-training is a *wrapper* method

- the choice of learner for f in step 3 is left completely open
- good for many real world tasks like natural language processing
- but mistake by f can reinforce itself

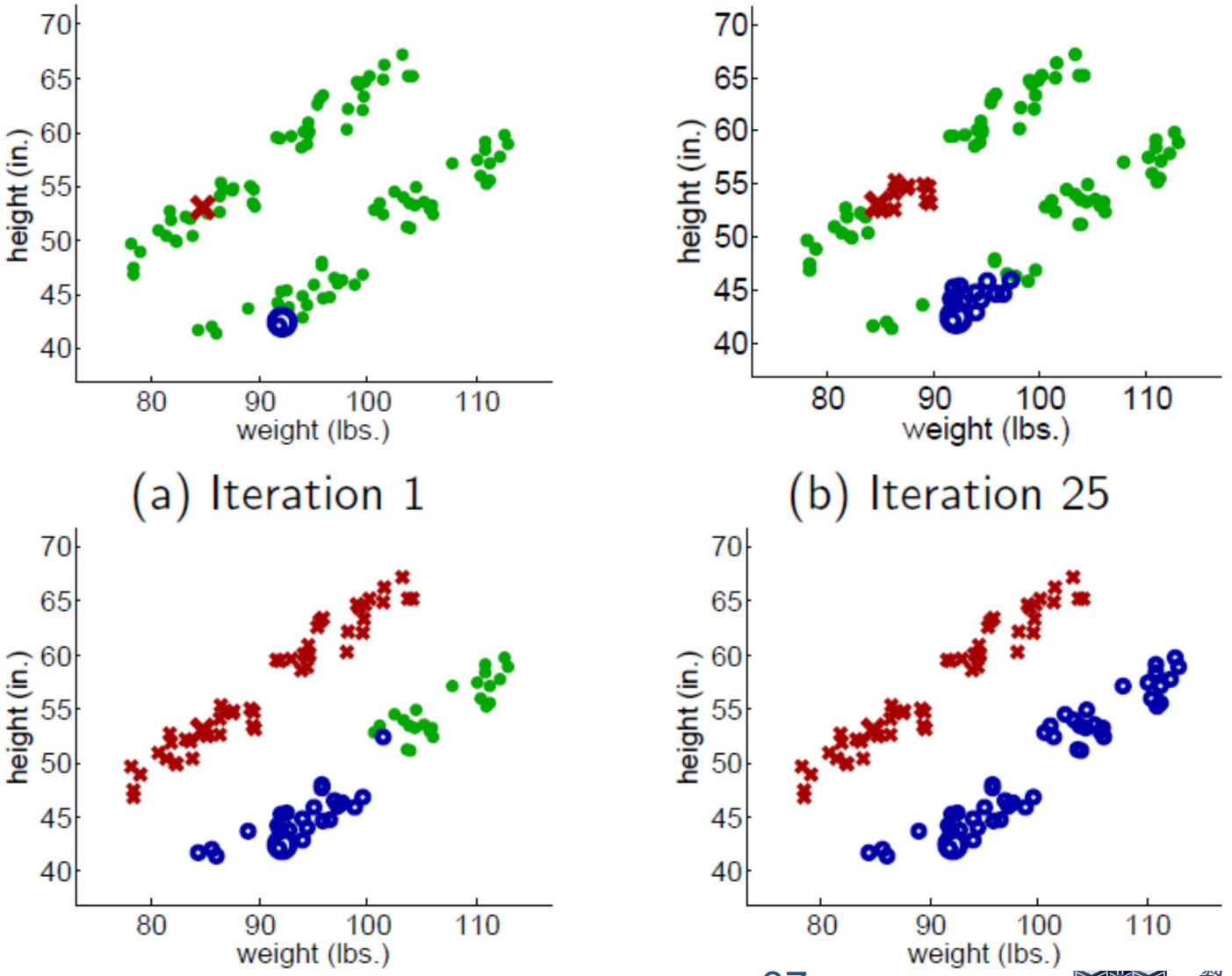
Self-Training Example: Propagating 1-NN

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, distance function $d()$.

1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
2. Repeat until U is empty:
3. Select $\mathbf{x} = \text{argmin}_{\mathbf{x} \in U} \min_{\mathbf{x}' \in L} d(\mathbf{x}, \mathbf{x}')$.
4. Set $f(\mathbf{x})$ to the label of $\mathbf{x}$'s nearest instance in L. Break ties randomly.
5. Remove $\mathbf{x}$ from U; add $(\mathbf{x}, f(\mathbf{x}))$ to L.

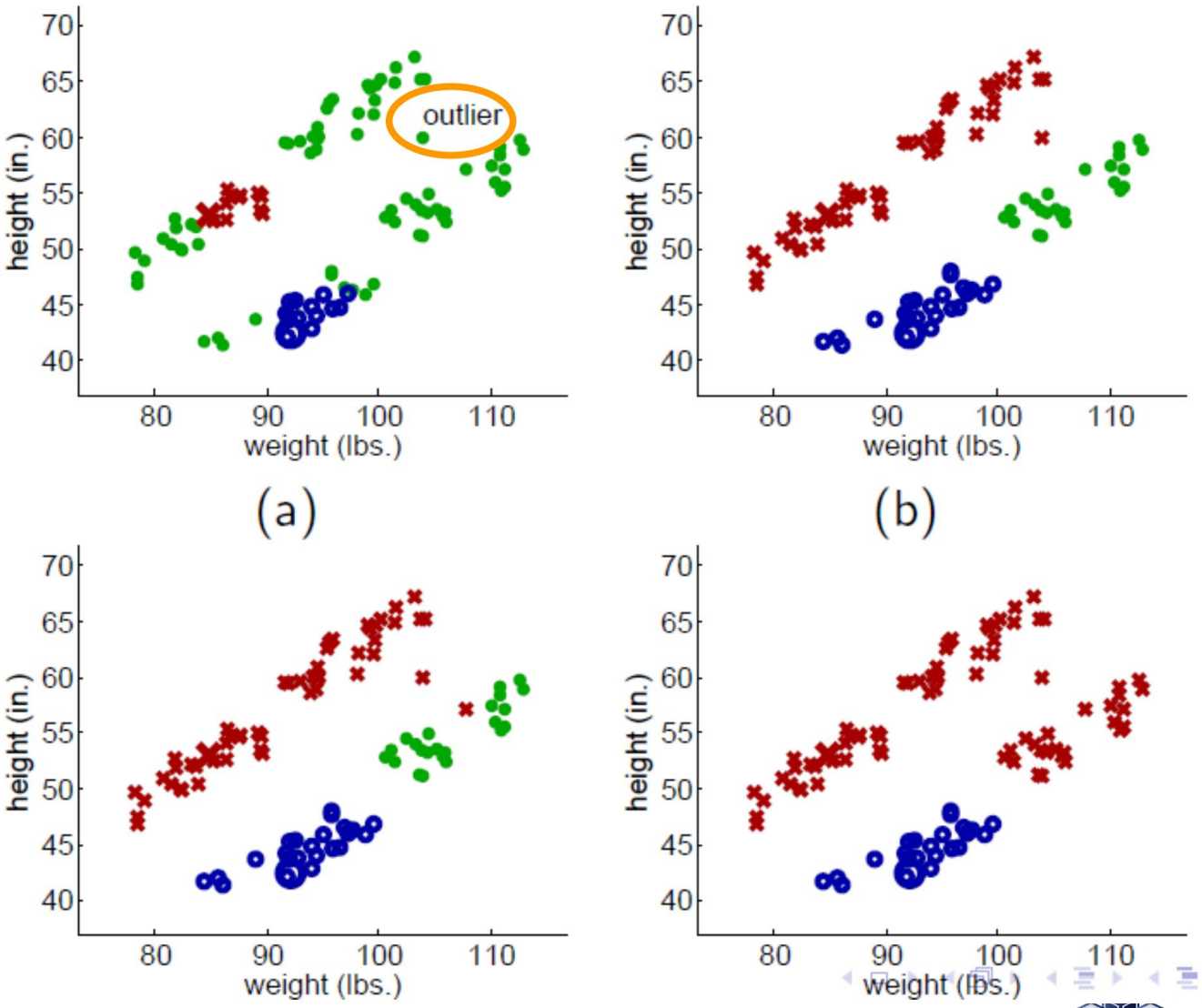
Example from: https://www.cs.cmu.edu/~10701/slides/17_SSL.pdf

Self-Training



Example from: https://www.cs.cmu.edu/~10701/slides/17_SSL.pdf

Sensitive to Outlier



Example from: https://www.cs.cmu.edu/~10701/slides/17_SSL.pdf