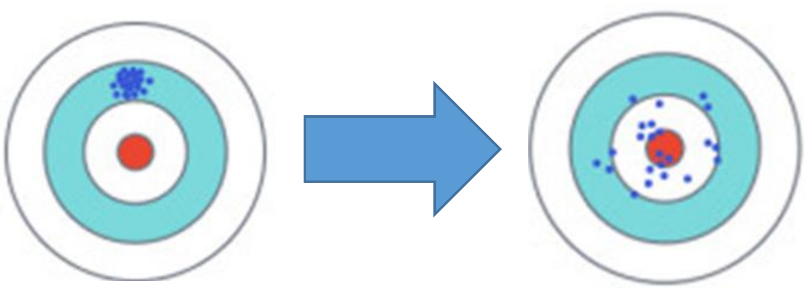
Ensemble

Ensemble: Bagging

Review: Bias v.s. Variance

Large Bias

Small Variance



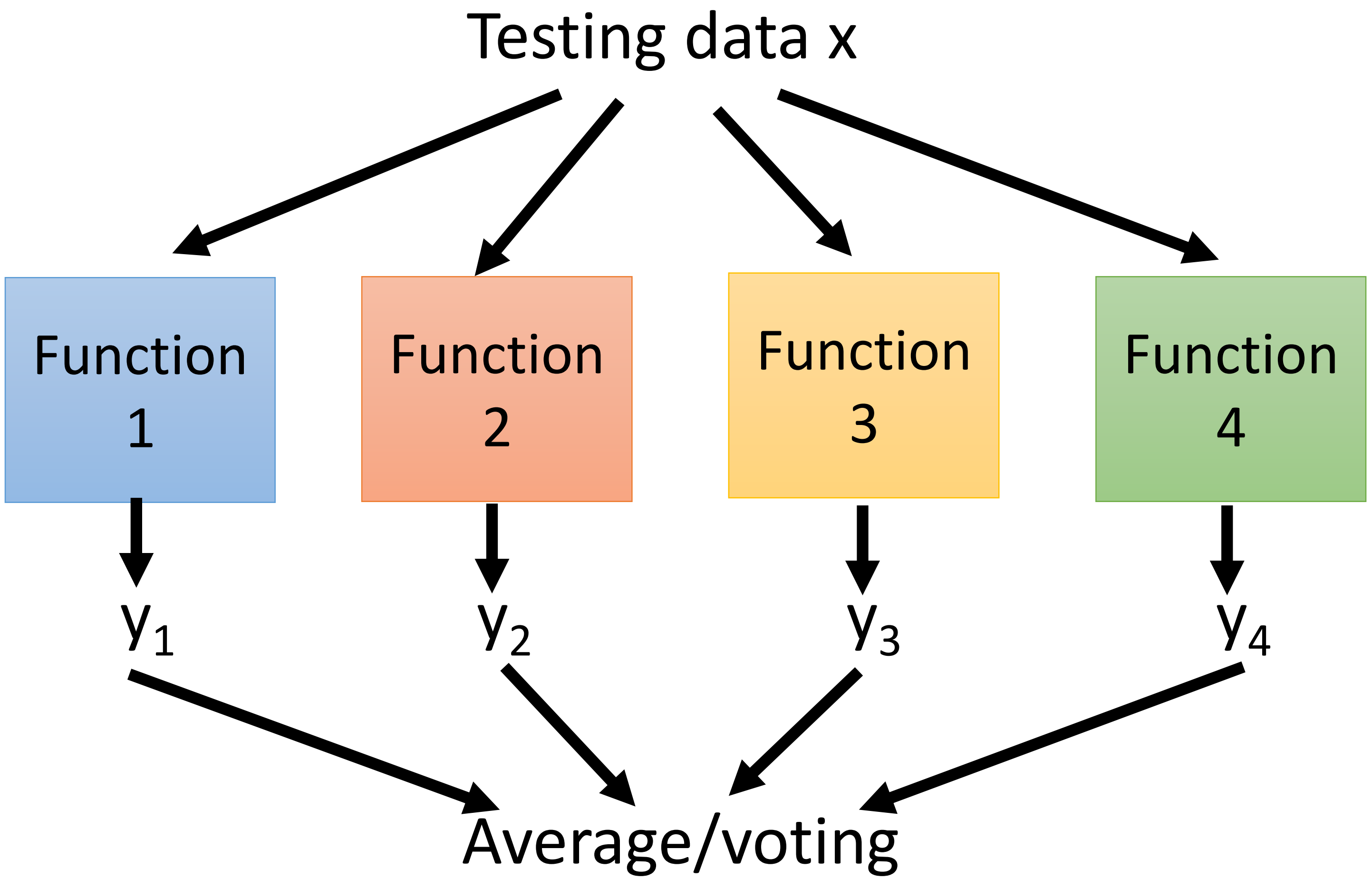
Small Bias

Large Variance

Bagging

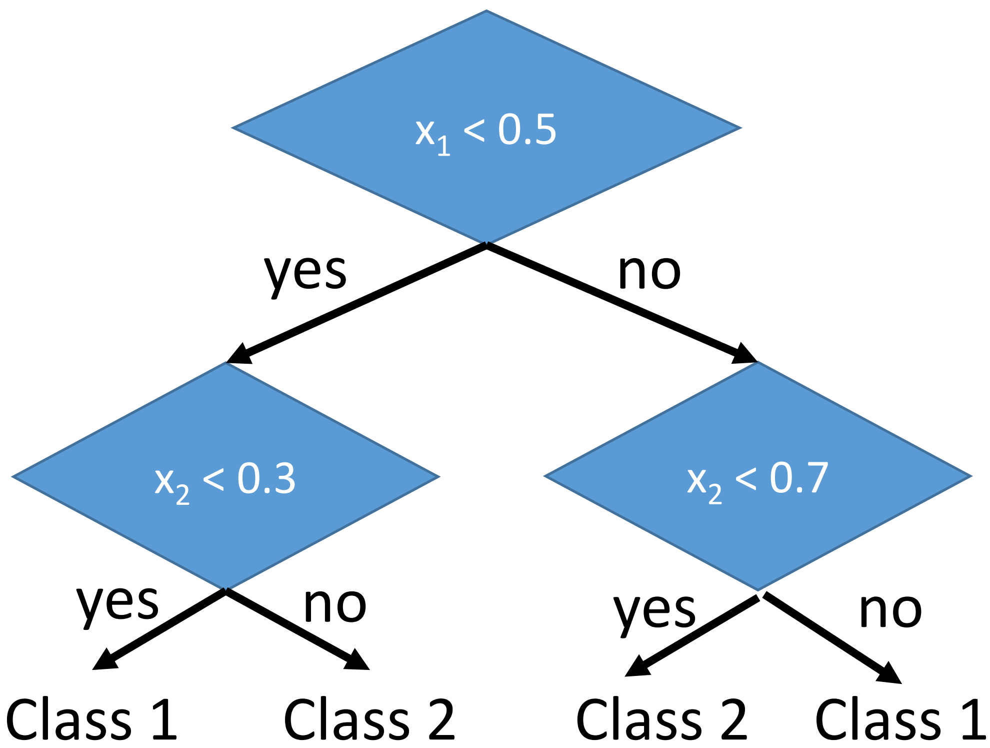
This approach would be helpful when your model is complex, easy to overfit.

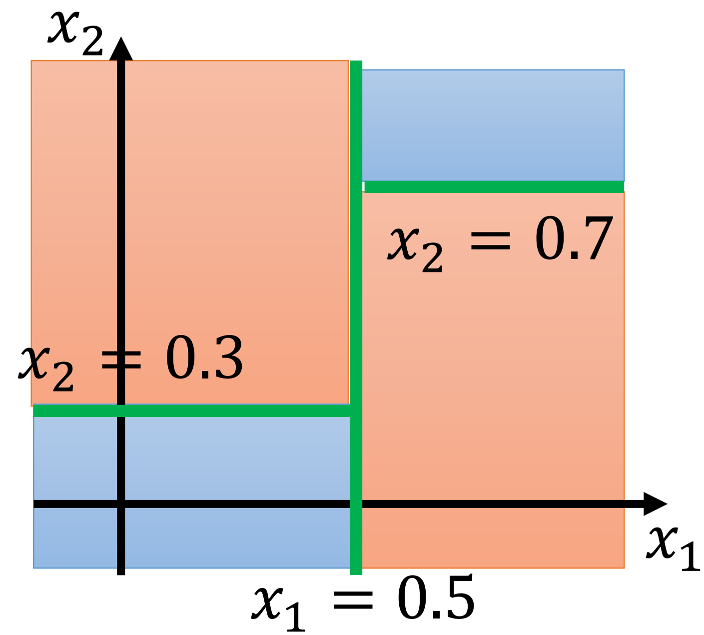
e.g. decision tree



Decision Tree

Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$





Can have more complex questions

The questions in training .....

number of branches,
Branching criteria,
termination criteria,
base hypothesis

Random Forest

train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging
  - Using RF = f_2+f_4 to test x^1
  - Using RF = f_2+f_3 to test x^2
  - Using RF = f_1+f_4 to test x^3
  - Using RF = f_1+f_3 to test x^4

Out-of-bag (OOB) error

Good error estimation of testing set

Ensemble: Boosting

Improving Weak Classifiers

Boosting

Training data:

$\{(x^1, \hat{y}^1), \cdots, (x^n, \hat{y}^n), \cdots, (x^N, \hat{y}^N)\}$

$\hat{y} = \pm 1$ (binary classification)

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - …… Finally, combining all the classifiers
- The classifiers are learned sequentially.

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1)$ $u^1 = \cancel{1}$ 0.4

$(x^2, \hat{y}^2, u^2)$ $u^2 = \cancel{1}$ 2.1

$(x^3, \hat{y}^3, u^3)$ $u^3 = \cancel{1}$ 0.7

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$$\Downarrow$$

$$L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)$$

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

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

Changing the example weights from u_1^n to u_2^n such that

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

The performance of f_1 for new weights would be random.

Training $f_2(x)$ based on the new weights u_2^n

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

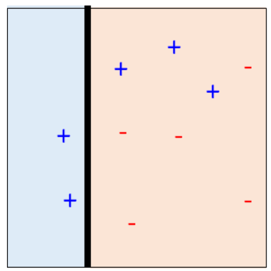
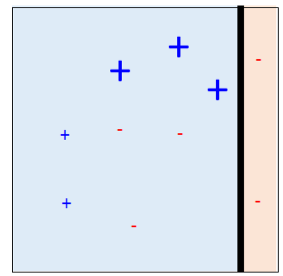
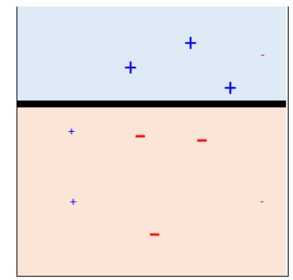
$$\begin{cases} \text{If } x^n \text{ misclassified by } f_1 \ (f_1(x^n) \neq \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \quad \boxed{\text{increase}} \\ \text{If } x^n \text{ correctly classified by } f_1 \ (f_1(x^n) = \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \quad \boxed{\text{decrease}} \end{cases}$$

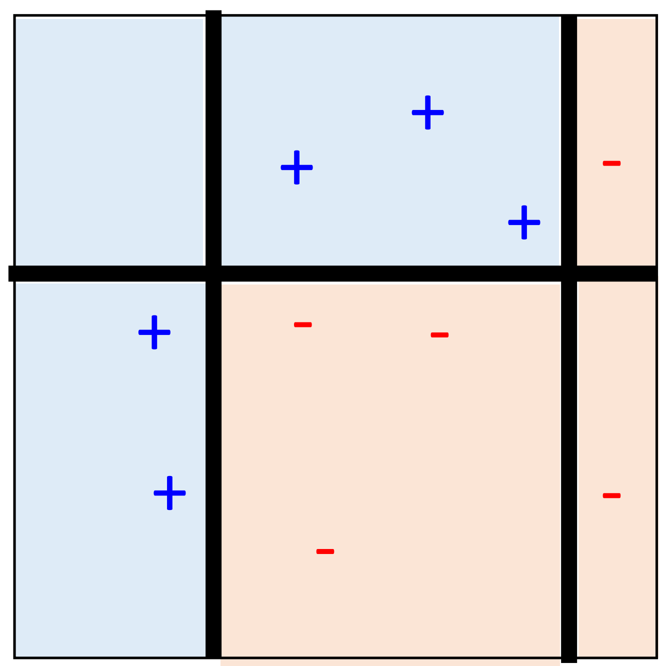
f_2 will be learned based on example weights u_2^n

$$d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} > 1$$

Toy Example

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

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



General Formulation of Boosting

- Initial function $g_0(x) = 0$
- For t = 1 to T:
  - Find a function $f_t(x)$ and α_t to $\boxed{\text{improve}}$ $g_{t-1}(x)$
    - $g_{t-1}(x) = \sum_{i=1}^{t-1} \alpha_i f_i(x)$
  - $g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$
- Output: $H(x) = sign\big(g_T(x)\big)$

What is the learning target of $g(x)$?

Minimize $L(g) = \sum_n l\big(\hat{y}^n, g(x^n)\big) = \sum_n exp(-\hat{y}^n g(x^n))$

Gradient Boosting

- Find g(x), minimize $L(g) = \sum_n exp(-\hat{y}^n g(x^n))$
  - If we already have g$(x) = g_{t-1}(x)$, how to update g(x)?

- Find g(x), minimize $L(g) = \sum_n exp(-\hat{y}^n g(x^n))$

$$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$$

α_t is something like learning rate

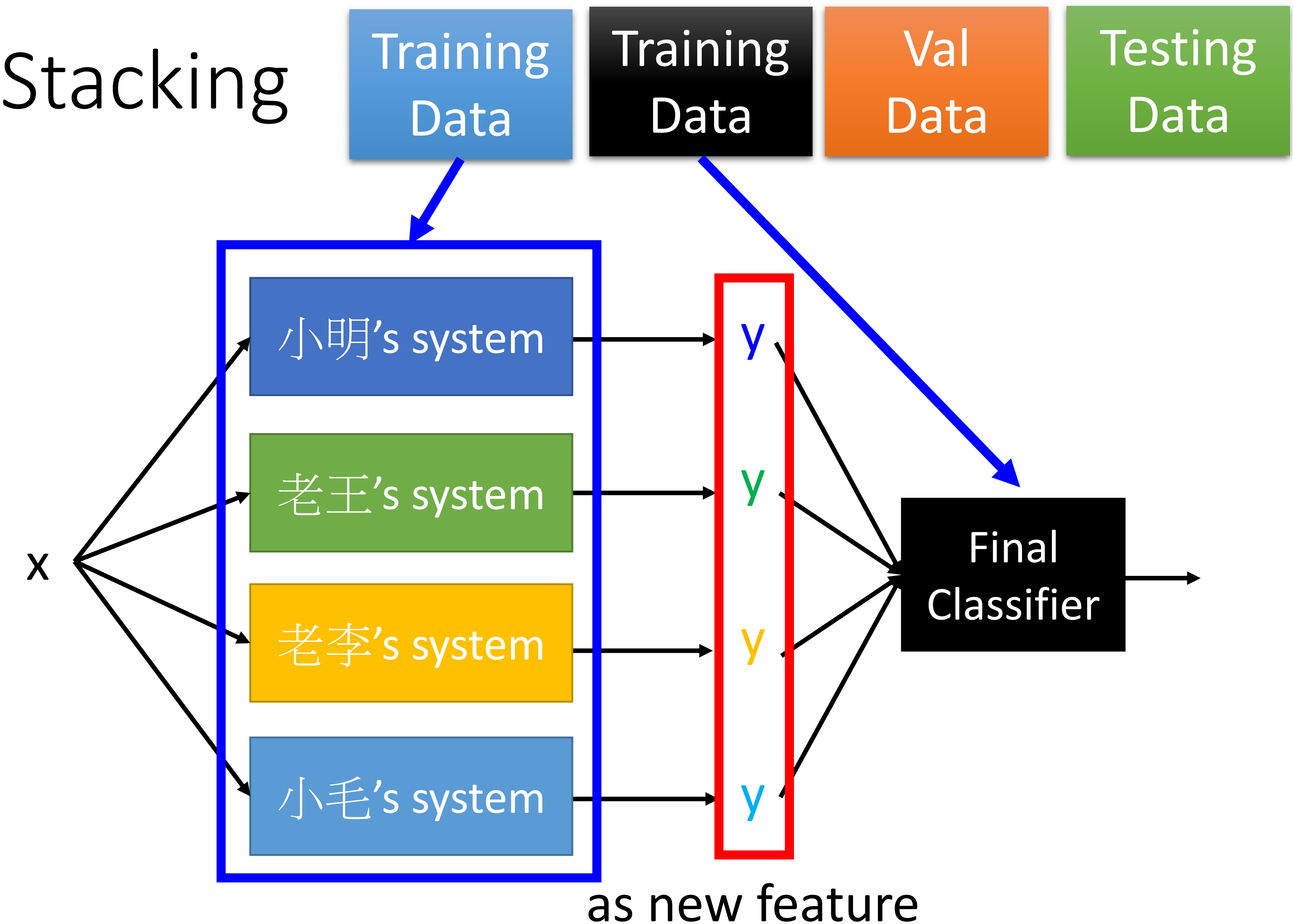
Find α_t minimzing $L(g_{t+1})$

$$L(g) = \sum_n exp\left(-\hat{y}^n\left(g_{t-1}(x) + \alpha_t f_t(x)\right)\right)$$

Find α_t such that $\frac{\partial L(g)}{\partial \alpha_t} = 0$ $\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$

Ensemble: Stacking

Stacking



Training Data
Training Data
Val Data
Testing Data
小明’s system
老王’s system
老李’s system
小毛’s system
x
y
y
y
y
Final Classifier
as new feature