Lazy Learners

- The classification algorithms presented before are **eager learners**
  - Construct a model before receiving new tuples to classify
  - Learned models are ready and eager to classify previously unseen tuples
- **Lazy learners**
  - The learner waits till the last minute before doing any model construction
  - In order to classify a given test tuple
    - Store training tuples
    - Wait for test tuples
    - Perform generalization based on similarity between test and the stored training tuples

K NN

Lazy vs Eager

Eager Learners	Lazy Learners
• Do lot of work on training data	• Do less work on training data
• Do less work when test tuples are presented	• Do more work when test tuples are presented

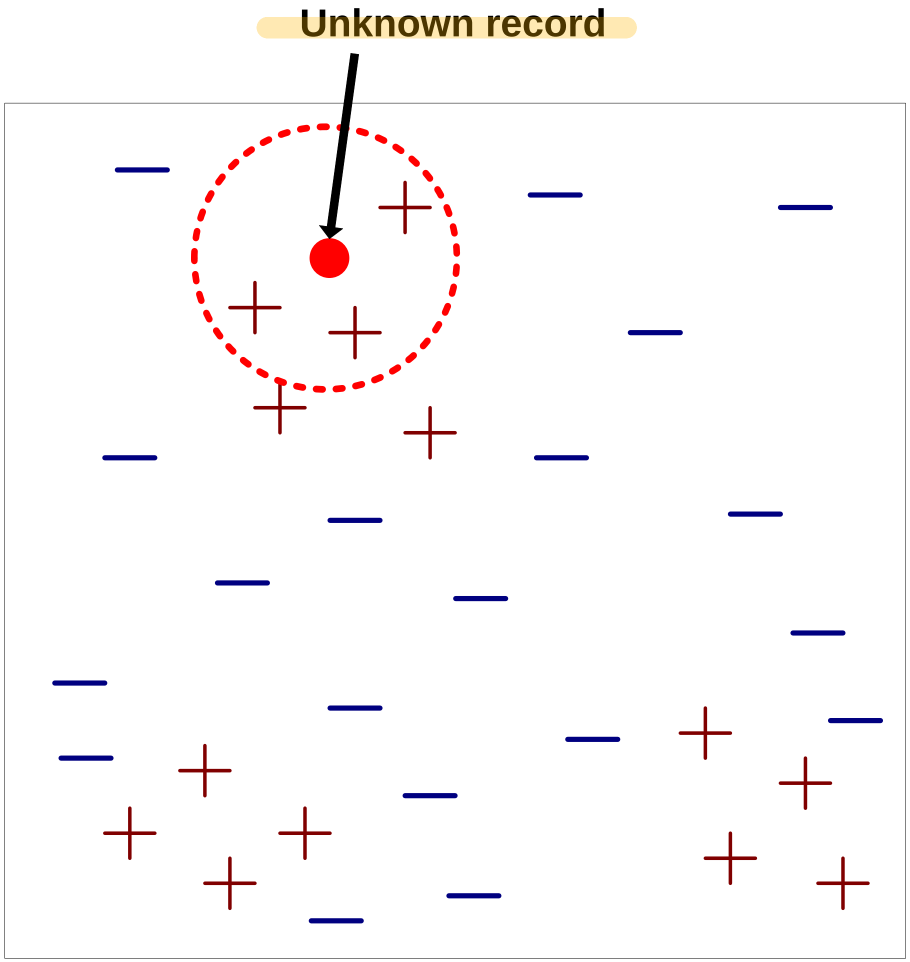
Basic k-Nearest Neighbor Classification

- Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Define a distance metric between points in input space *$D(x_1,x_i)$*
  - E.g., Eucledian distance, Weighted Eucledian, Mahalanobis distance, TFIDF, etc.

- Training method:
  - Save the training examples
- At prediction time:
  - Find the ***k*** training examples *$(x_1,y_1),\ldots(x_k,y_k)$* that are **closest** to the test example *x* given the distance *$D(x_1,x_i)$*
  - Predict the most frequent class among those *y_i*'s.

→ Majority Rule

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Imp

K-Nearest Neighbor Model

- **Classification:**

Multinomial Classification

$$\hat{y} = \text{most common class in set } \{y_1, \ldots, y_K\}$$

$c = \{c_1, c_2, \cdots\}$

- **Regression:**

$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} y_k$$

(K nearest avg value)

K-Nearest Neighbor Model: Weighted by Distance

- **Classification:**

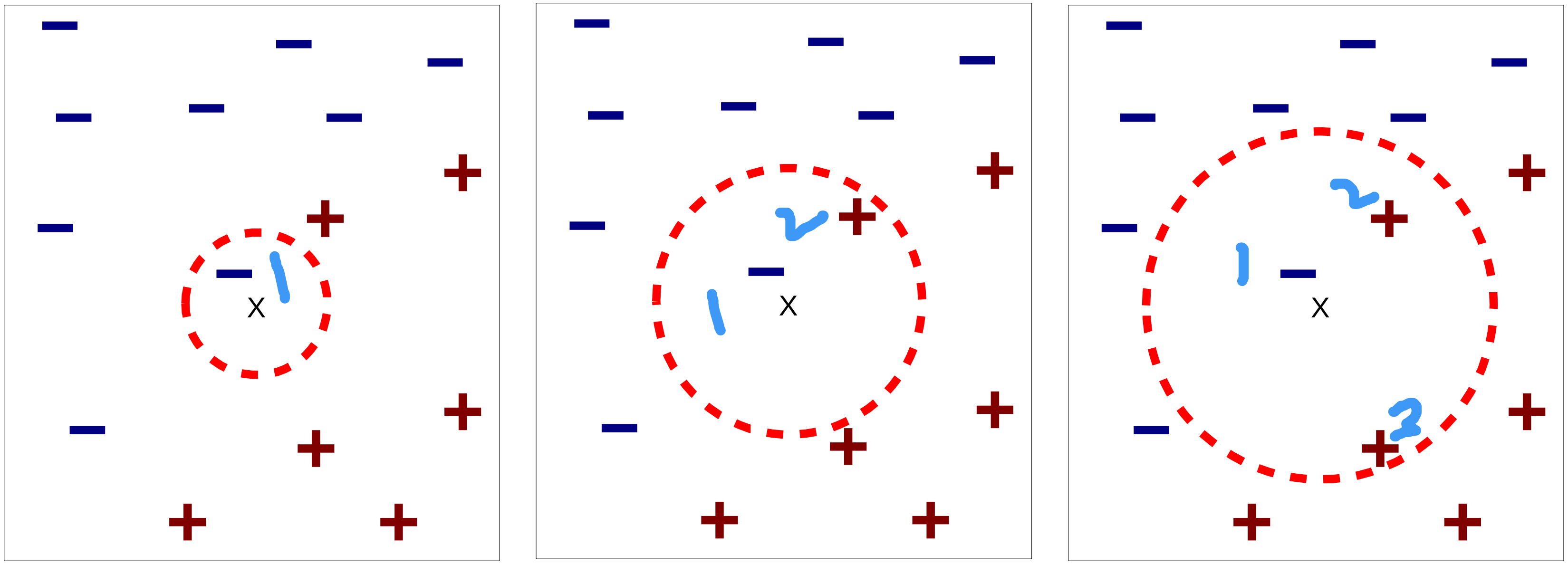
$$\hat{y} = \text{ most common class in wieghted set}$$

$$\{D(\mathbf{x},\mathbf{x}_1)y_1,\ldots,D(\mathbf{x},\mathbf{x}_K)y_K\}$$

- **Regression:**

$$\hat{y} = \frac{\sum_{k=1}^{K} D(x,x_k)y_k}{\sum_{k=1}^{K} D(x,x_k)}$$

Definition of Nearest Neighbor



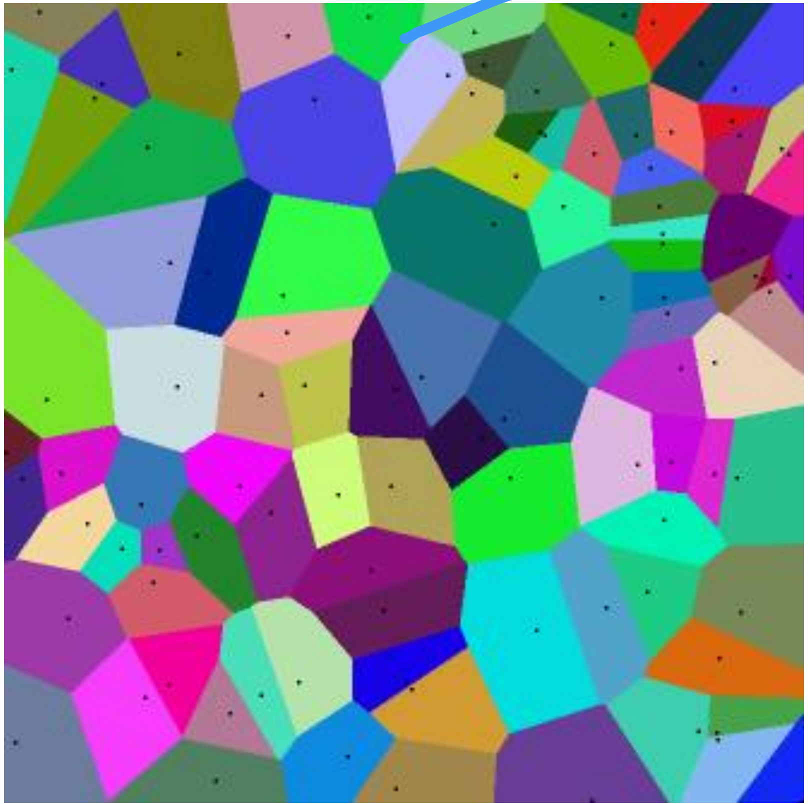
(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

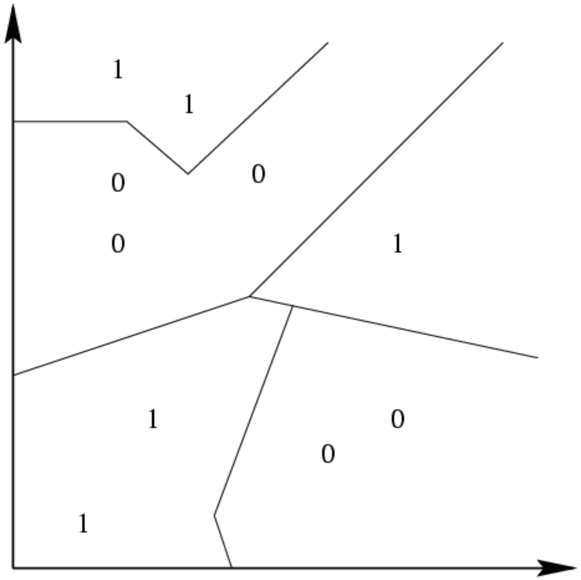
K-nearest neighbors of a record x are data points that have the k smallest distance to x

Voronoi Diagram

Decision Boundary

Decision surface formed by the training examples





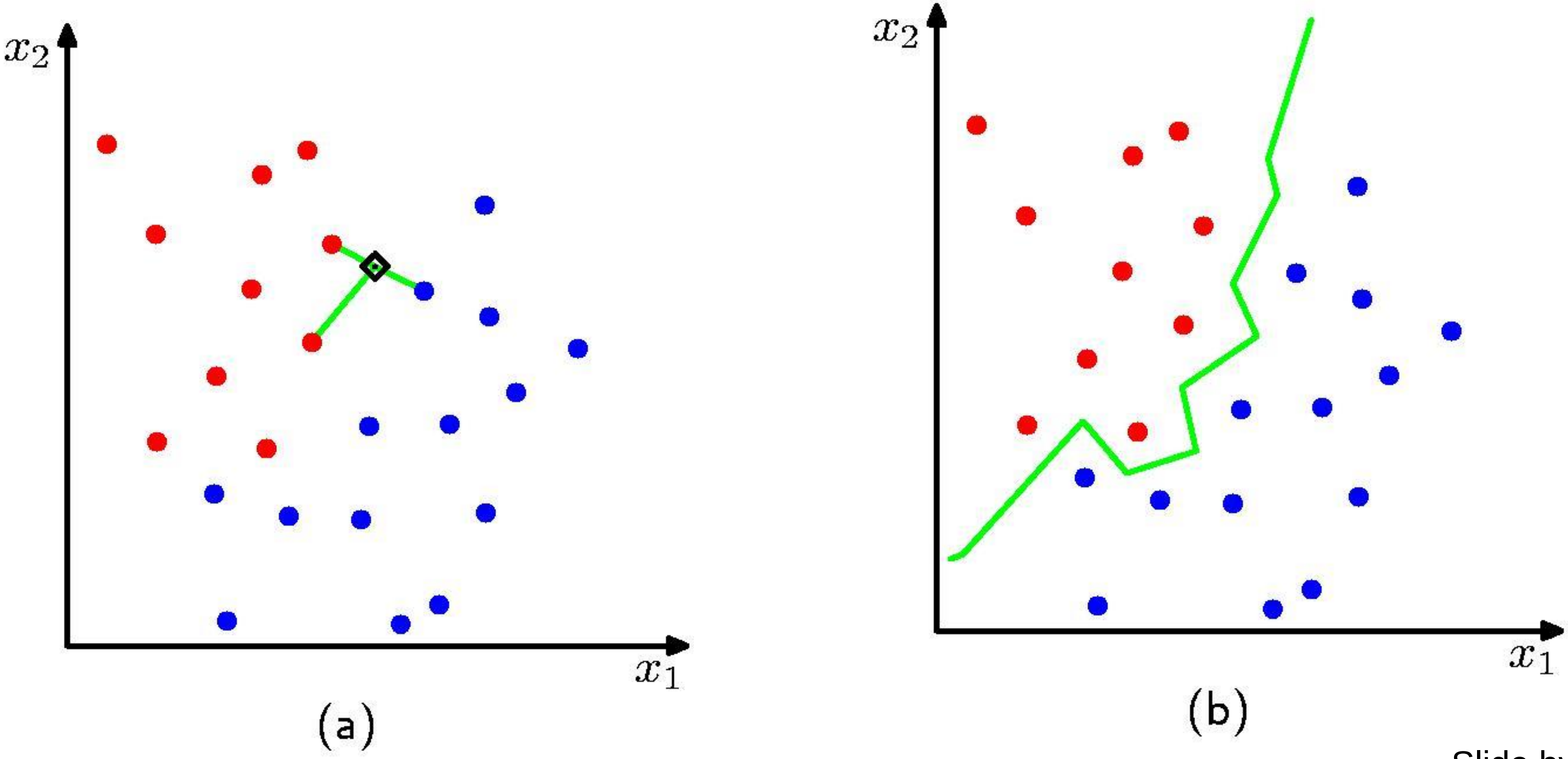
- Each line segment is equidistance between points in opposite classes.
- The more points, the more complex the boundaries.

multinomial Classifier.

~~The~~ decision boundary implemented by 3NN

Imp

The boundary is always the perpendicular bisector of the line between two points (Voronoi tessellation)



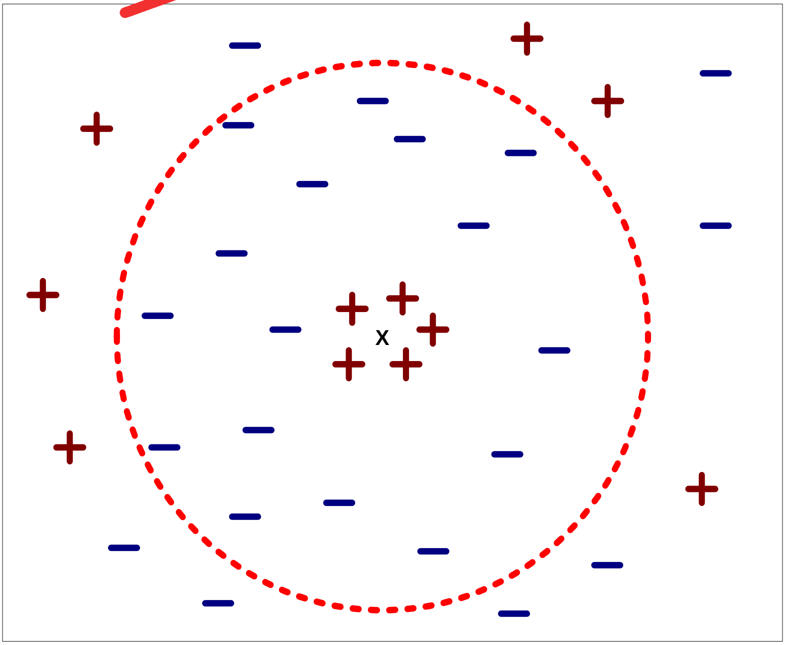
(a)

(b)

Slide by Hinton

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Determining the value of k

- In typical applications k is in units or tens rather than in hundreds or thousands
- Higher values of k provide smoothing that reduces the risk of overfitting due to noise in the training data
- Value of k can be chosen based on error rate measures
- We should also avoid over-smoothing by choosing k=n, where n is the total number of tuples in the training data set

underfitting.

low bias

high variance

6

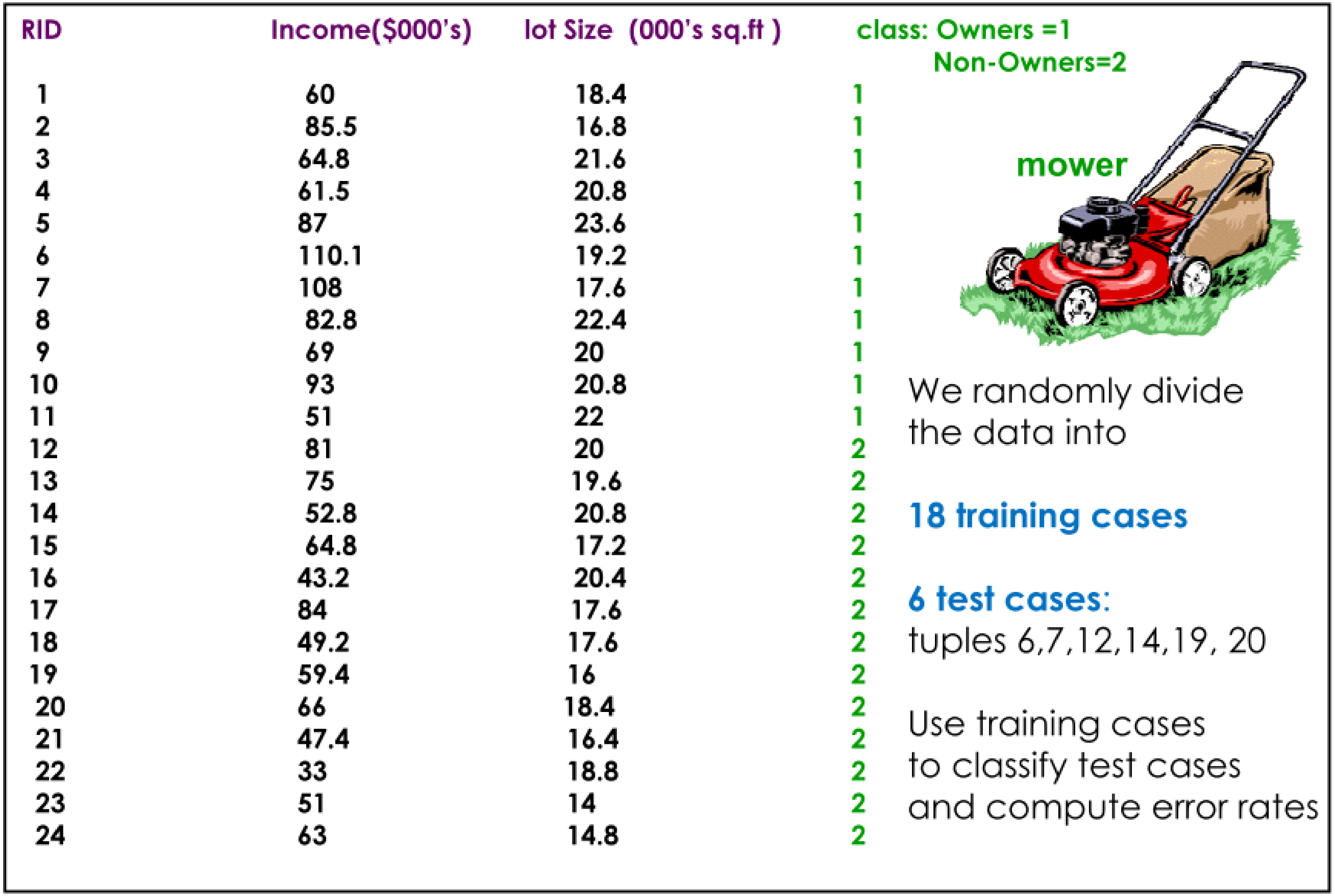
Determining the value of k

- Given training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Use N fold cross validation
  - Search over K = (1,2,3,…,*Kmax*). Choose search size *Kmax* based on compute constraints
  - Calculated the average error for each K:
    - Calculate predicted class $\hat{y}_i$ for each training point

      $$(\mathbf{x}_i, y_i), \quad i = 1, \ldots, N$$

      (using all other points to build the model)
    - Average over all training examples
- Pick K to minimize the cross validation error

Example



RID	Income($000's)	lot Size (000's sq.ft)	class: Owners =1
			Non-Owners=2

1	60	18.4	1
2	85.5	16.8	1
3	64.8	21.6	1
4	61.5	20.8	1
5	87	23.6	1
6	110.1	19.2	1
7	108	17.6	1
8	82.8	22.4	1
9	69	20	1
10	93	20.8	1
11	51	22	1
12	81	20	2
13	75	19.6	2
14	52.8	20.8	2
15	64.8	17.2	2
16	43.2	20.4	2
17	84	17.6	2
18	49.2	17.6	2
19	59.4	16	2
20	66	18.4	2
21	47.4	16.4	2
22	33	18.8	2
23	51	14	2
24	63	14.8	2

mower

We randomly divide the data into

18 training cases

6 test cases:
tuples 6,7,12,14,19, 20

Use training cases
to classify test cases
and compute error rates

Choosing k

- If we choose **k=1** we will classify in a way that is very sensitive to the local characteristics of our data

- If we choose a **large value of k** we average over a large number of data points and average out the variability due to the noise associated with data points

- If we choose **k=18** we would simply predict the most frequent class in the data set in all cases
  - → Very stable but completely ignores the information in the independent variables

k	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**18**
Misclassification error %	33	33	33	33	33	17	17	50

→ We would choose k=11 (or possibly 13) in this case

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
- Adv: No training time
- Disadv:
  - Testing time can be long, classifying unknown records are relatively expensive
  - Curse of Dimensionality : Can be easily fooled in high dimensional spaces
    - Dimensionality reduction techniques are often used

Ensemble Methods

1 - One of the eager methods => builds model over the training set

[illegible]

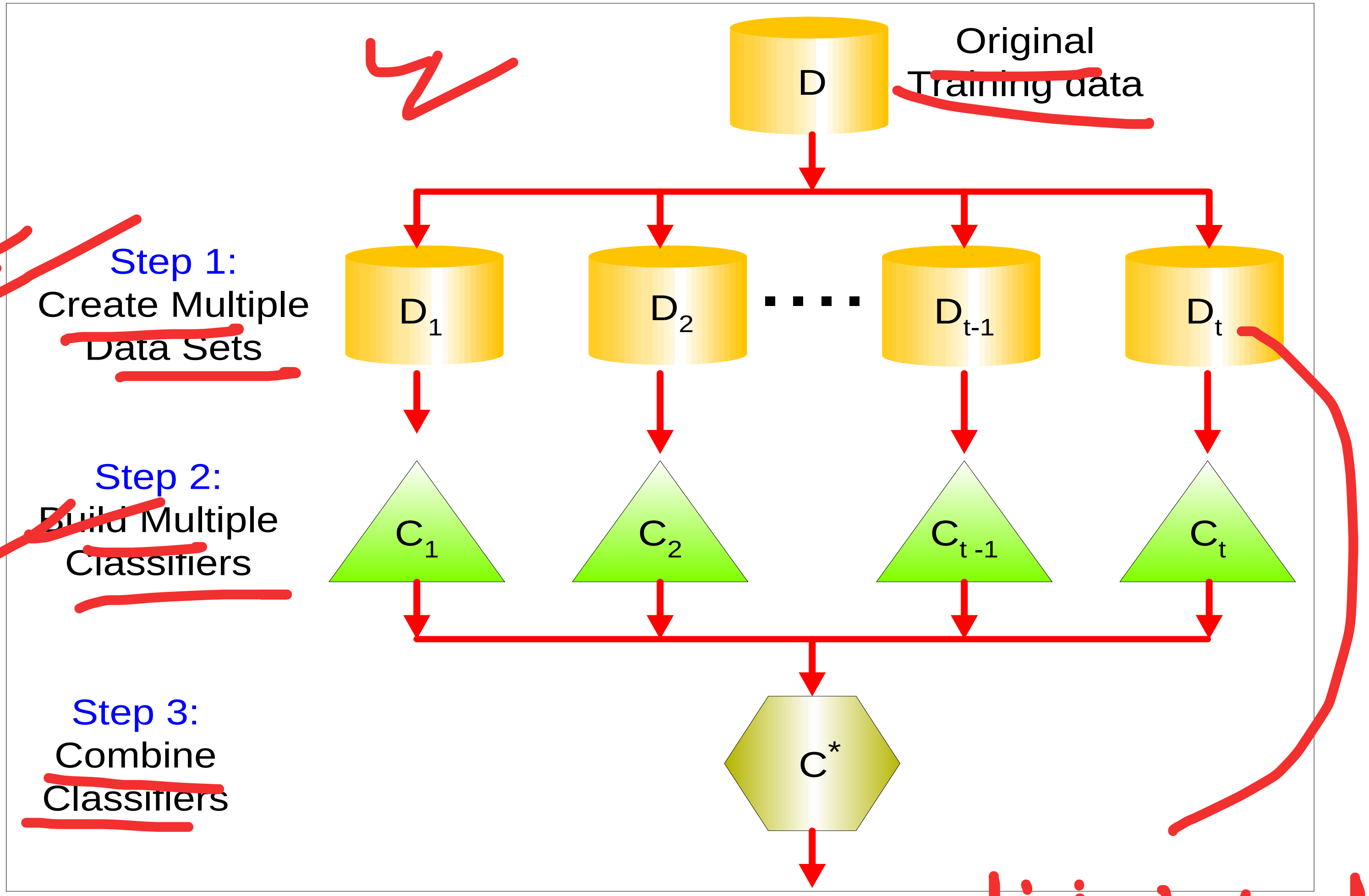
- Construct a set of classifiers from the training data

2 [illegible]

3 - Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

[illegible] majority rule

General Idea



Step 1:
Create Multiple Data Sets

Step 2:
Build Multiple Classifiers

Step 3:
Combine Classifiers

disjoint dataset

Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - 1 Bagging
  - 2 Boosting
  - 3 Random Forests

Bagging: Bootstrap AGGregatING

1 2

- Bootstrap: data resampling
  - Generate multiple training sets
    - Resample the original training data
    - With replacement
  - Data sets have different “specious” patterns
- Sampling with replacement
  - Each sample has probability $(1 – 1/n)^n$ of being selected

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
  - Specious patterns will not correlate
- Underlying true pattern will be common to many
- Combine the classifiers: Label new test examples by a majority vote among classifiers

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round
- The final classifier is the weighted combination of the weak classifiers.

Boosting

misclassified increase

around weight

Wg

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost (Freund and Schapire, 1996)

- Initialize distribution over the training set $D_1(i) = 1/m$

no of rows

- For $t = 1, \ldots, T$:

  1. Train *Weak Learner* using distribution D_t.
  2. Choose a weight (or confidence value) $\alpha_t \in \mathbf{R}$.
  3. Update the distribution over the training set:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} \tag{2}$$

Where Z_t is a normalization factor chosen so that D_{t+1} will be a distribution

$\alpha_1 + \alpha_2 \ldots = 1$

- Final vote $H(x)$ is a weighted sum:

$$H(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{3}$$

+1

-1

Example: AdaBoost

ε = Errors

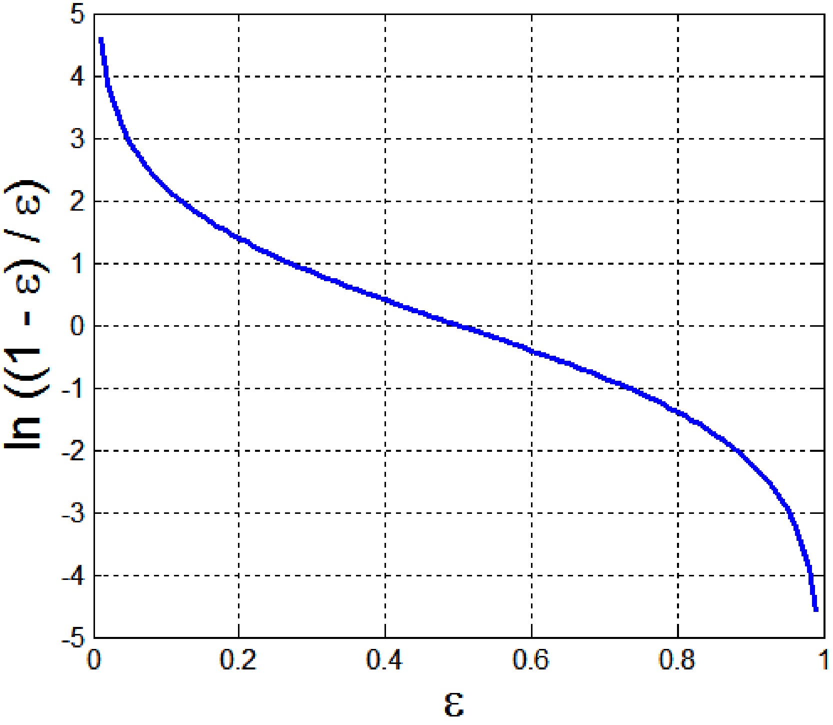
- Base classifiers (weak learners): C$_1$, C$_2$, …, C$_T$ $h_t : X \rightarrow \{-1,+1\}$
- Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

$$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Example: AdaBoost

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

$$\text{where } Z_j \text{ is the normalization factor}$$

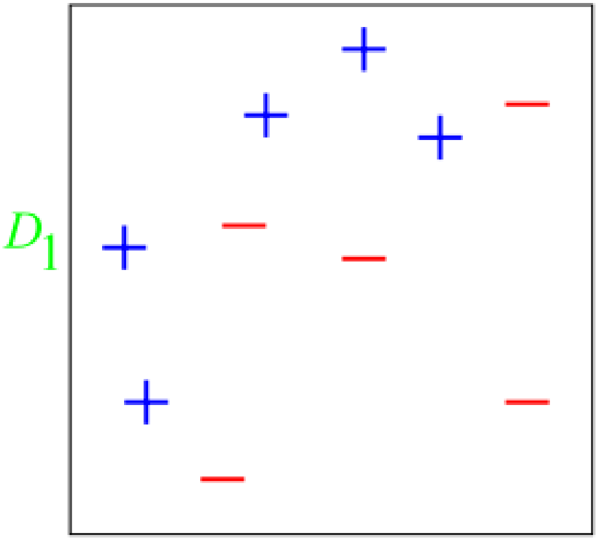
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \exp(-\alpha_t\, y_i\, h_t(x_i))$$

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

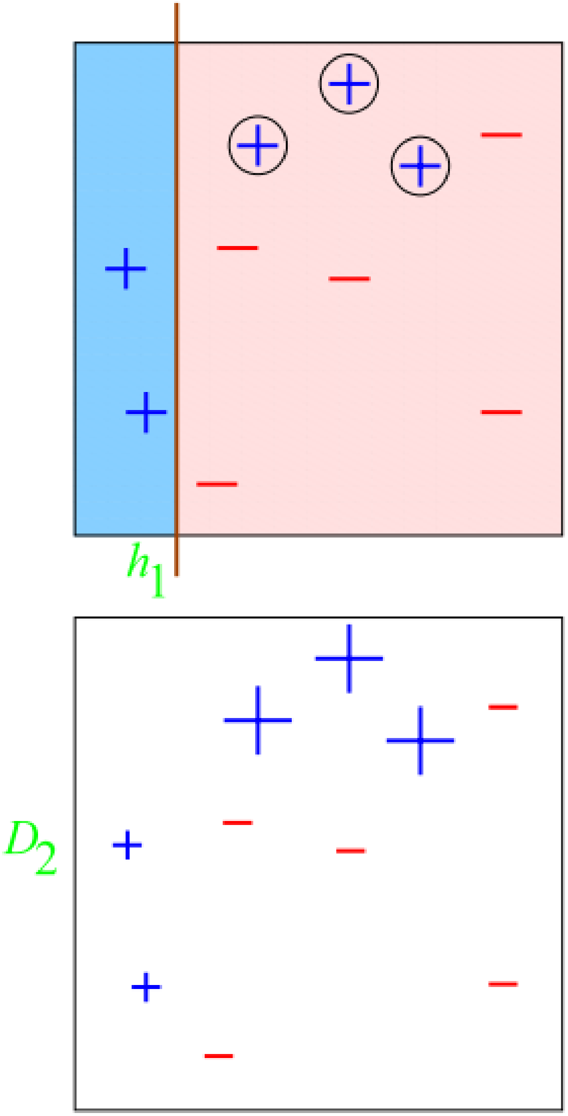
$$H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$$

2D Example



2D Example

Round 1

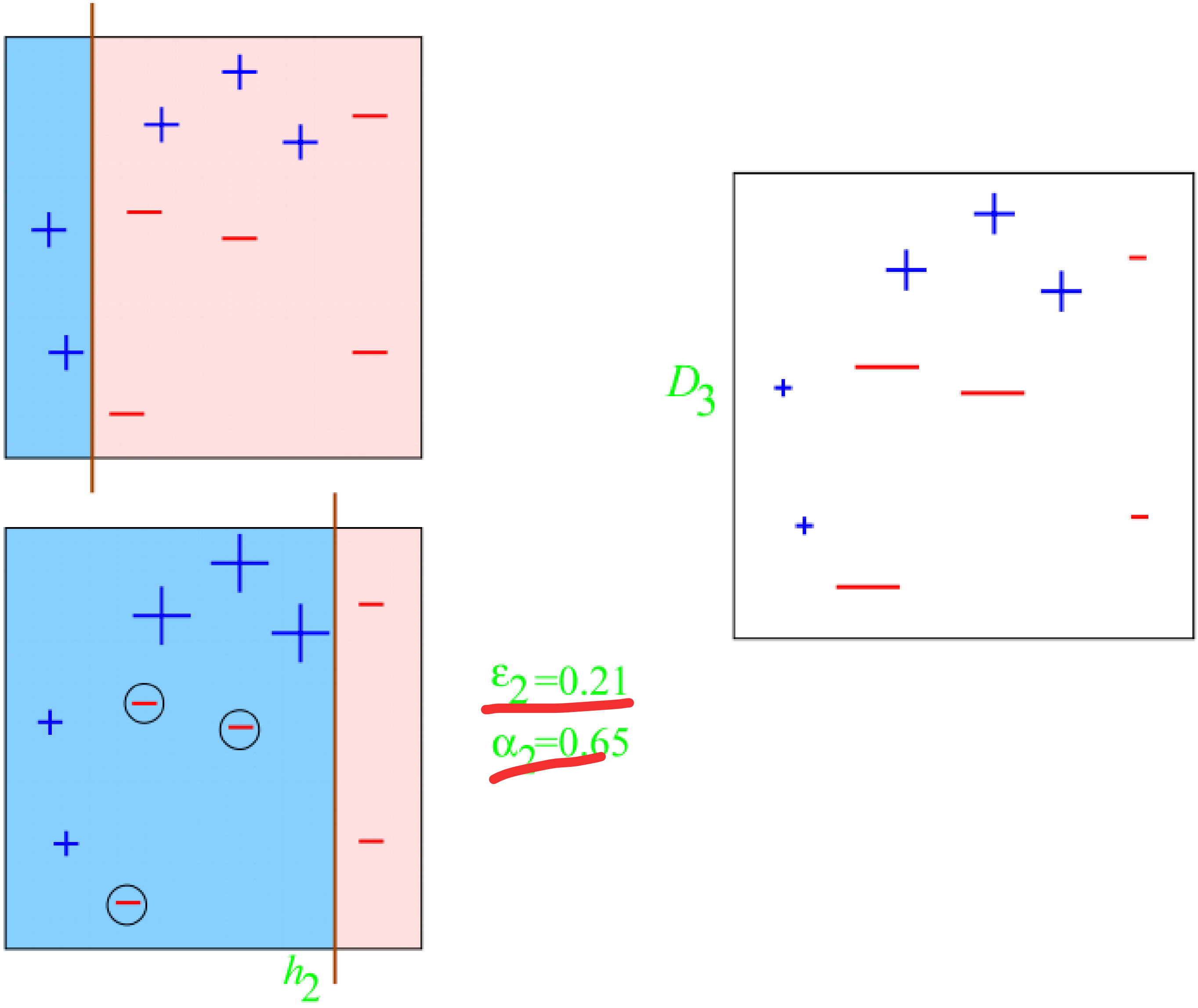


$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

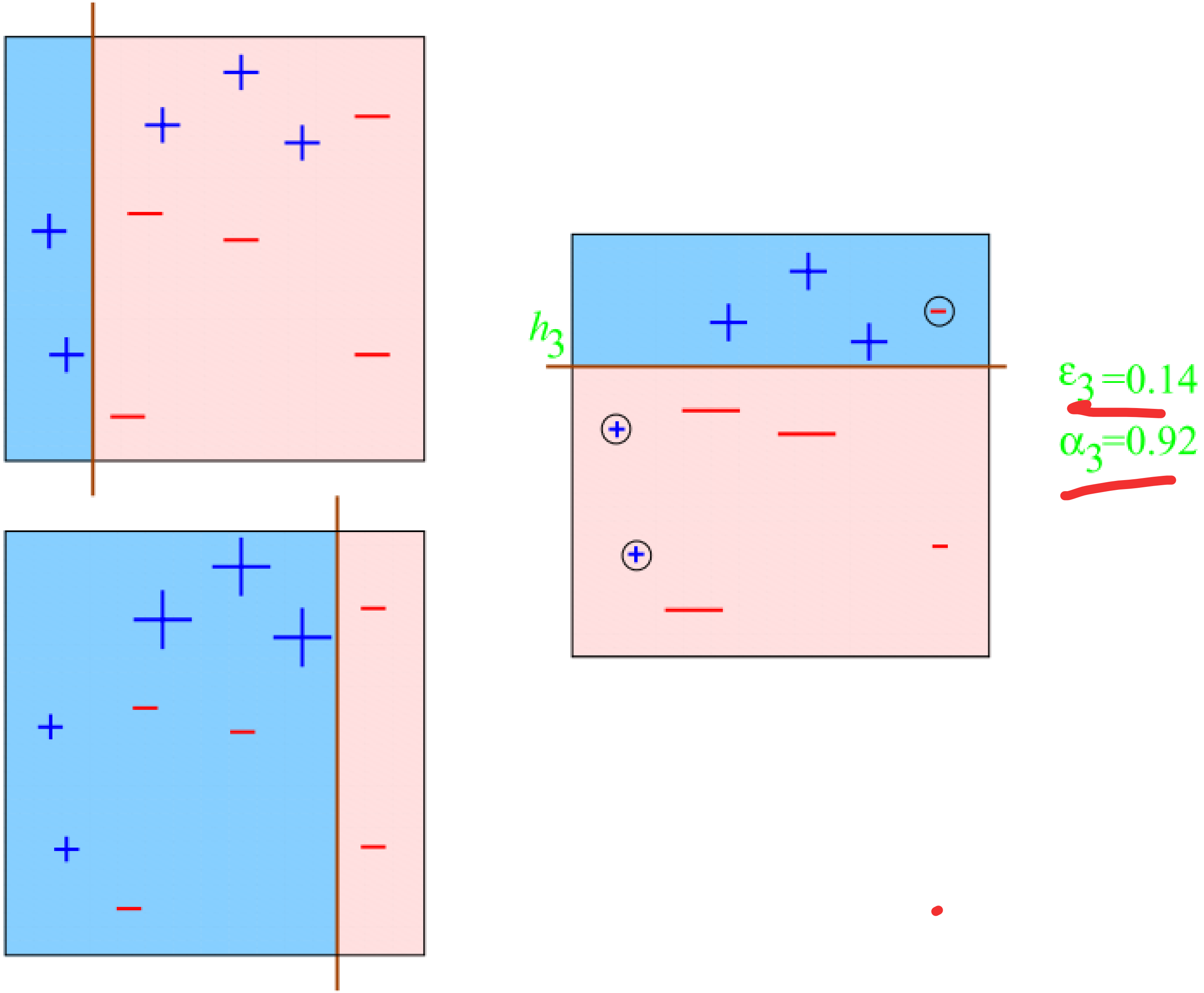
2D Example

Round 2

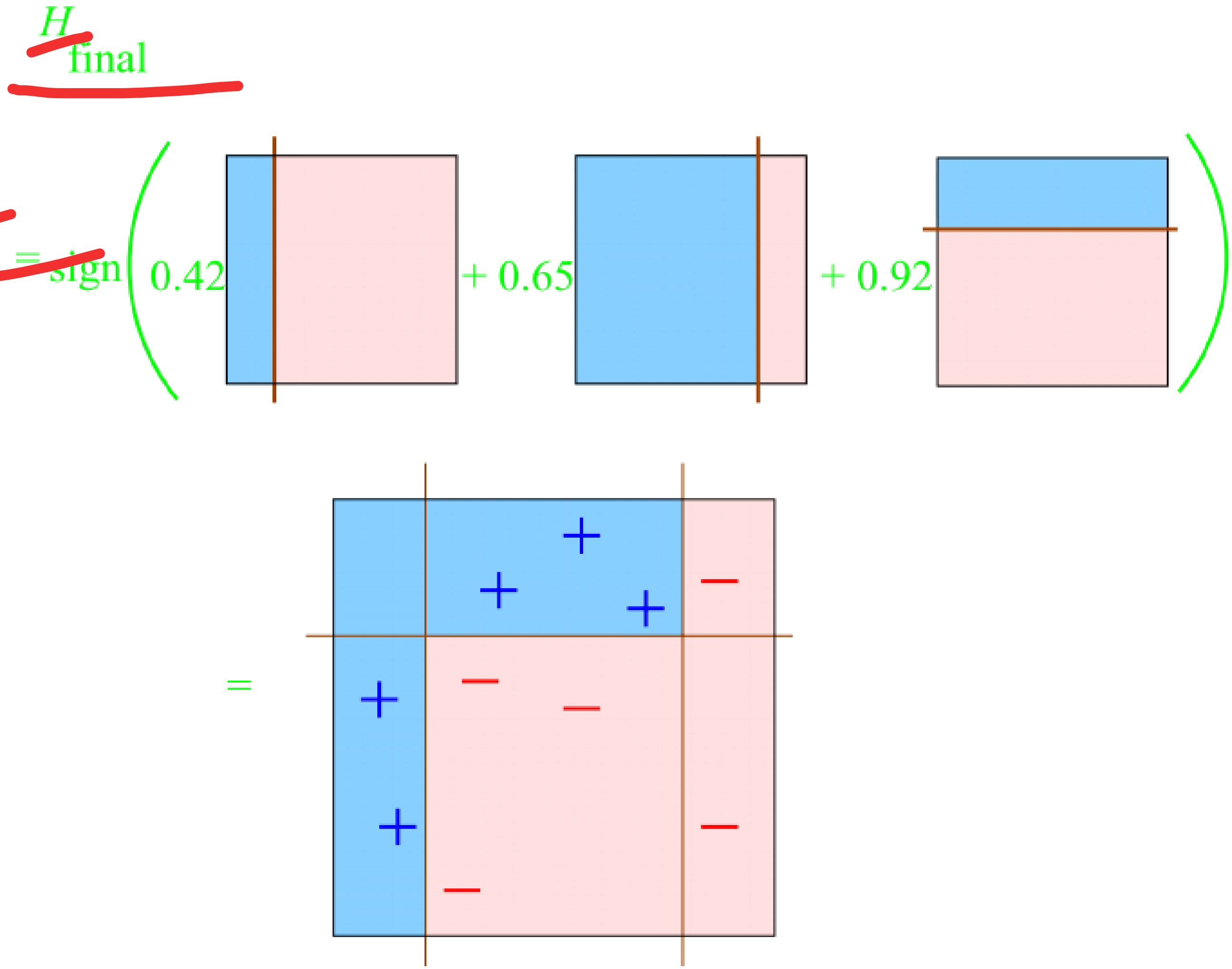


2D Example

Round 3



2D Example – Final hypothesis



- See demo at: www.research.att.com/˜yoav/adaboost