EE 4146 Data Engineering and Learning Systems

Lecture 12: Non-linear Perceptron and Classifier ensemble

Semester A, 2021-2022

Schedules

Week	Date	Topics
1	Sep. 1	Introduction
2	Sep. 8	Data exploration
3	Sep. 15	Feature reduction and selection (HW1 out)
4	Sep. 22	Mid-Autumn Festival
5	Sep. 29	Clustering I: Kmeans based models (HW1 due in this weekend)
6	Oct. 6	Clustering II: Hierarchical/density based/fuzzing clustering
7	Oct. 13	Midterm (no tutorials this week)
8	Oct. 20	Adverse Weather
9	Oct. 27	Linear classifiers
10	Nov. 3	Classification based on decision tree (Tutorial on project) (HW2 out)
11	Nov. 10	Bayes based classifier (Tutorial on codes) (HW2 due in this weekend)
12	Nov. 17	Non-linear Perceptron and Classifier ensemble
13	Nov. 24	Deep learning based models (Quiz)
14		Summary: based on the poll, we will do off-line video recording for the summary and will upload the video before Dec. 1st.

Course content

- Course grades
  - 2 Homework assignments (10%) (105 submission & 108 submission)
  - 1 Project report (10%) (Deadline Dec. 8)
  - 1 Midterm (20%) (109 submission)
  - 1 Quiz (10%) (Nov. 24)
  - Final exam (50%)

- Remarks:
  - By university regulation, to pass the course, students are required to achieve at least 30% in course work and 30% in the examination.
  - Attend at least two course work, including homework assignments and quiz.

Quiz 2

- Zoom Quiz
  - We will have the second quiz on Nov. 24 from 4:00 PM-5:00 PM. You will have 15 mins to scan your results and uploaded them through the assignment. It will have 4 calculation & understanding-related questions, covering all lecture notes (more emphasis on the notes after the midterm). Please join in this quiz through Canvas ZOOM.
  - Please take photos of your hand-written results together with your Cityu id, combine these calculation results in one file, and upload the file through the assignment. As mentioned in the class, you can use the matlab to calculate the final results, but you should write the detailed steps for each question. The one with only final results will not get the full marks.
  - No code-related questions.
  - Open-book and open-notes.
  - One-camera with your front face is acceptable
  - No late submission is allowed since you have 15mins to upload your results. If you could not upload the results, please send me emails of your results.
- Lecture at 5:30 PM- 6:50 PM, Nov. 24

Outline

- Non-linear perceptron
- Classifier ensemble

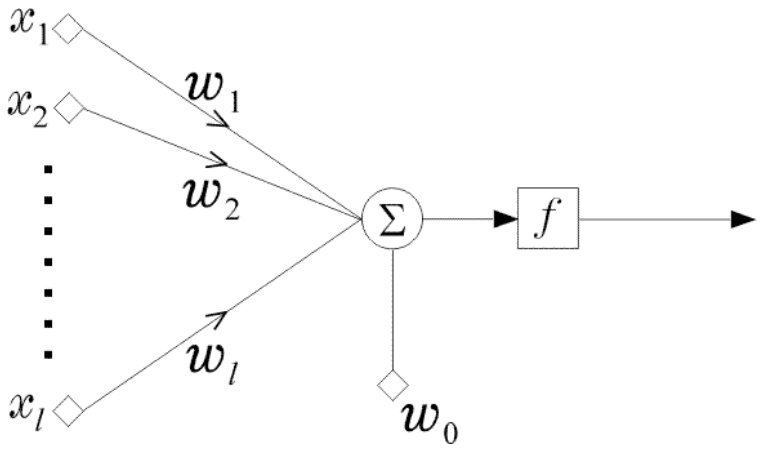
- Review Decision tree

Review: The Perceptron

$$y(\mathbf{x}) = f\left(\mathbf{w}^t\mathbf{x} + w_0\right)$$

$$y(\mathbf{x}) \geq 0 \rightarrow \mathbf{x} \text{ assigned to } C_1$$
$$y(\mathbf{x}) < 0 \rightarrow \mathbf{x} \text{ assigned to } C_2$$

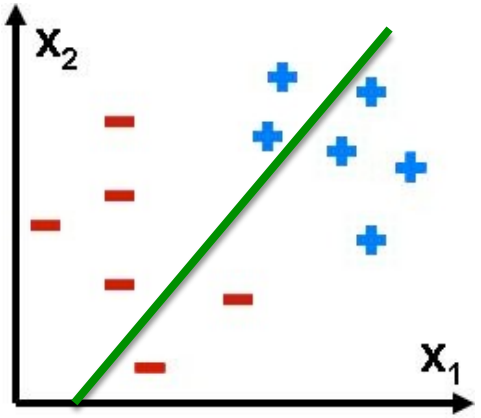
- A classifier based upon this simple generalized linear model is called a (single layer) perceptron.
- It can also be identified with an abstracted model of a neuron.



Review: The Perceptron Algorithm

- The perceptron algorithm was invented by Frank Rosenblatt (1962).
- The algorithm is iterative.
- The strategy is to start with a random guess at the weights **w**, and to then iteratively change the weights to move the hyperplane in a direction that lowers the classification error.

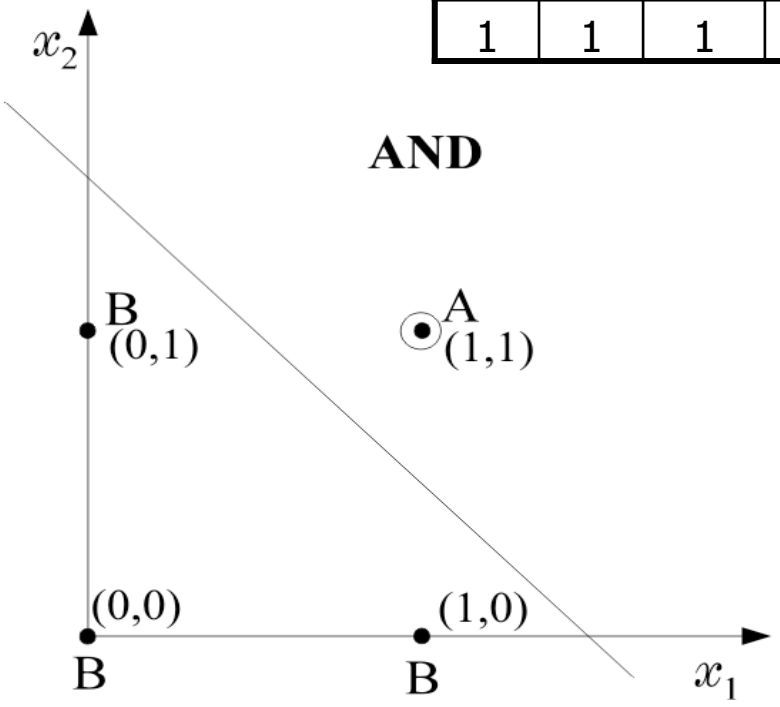
Frank Rosenblatt (1928 – 1971)



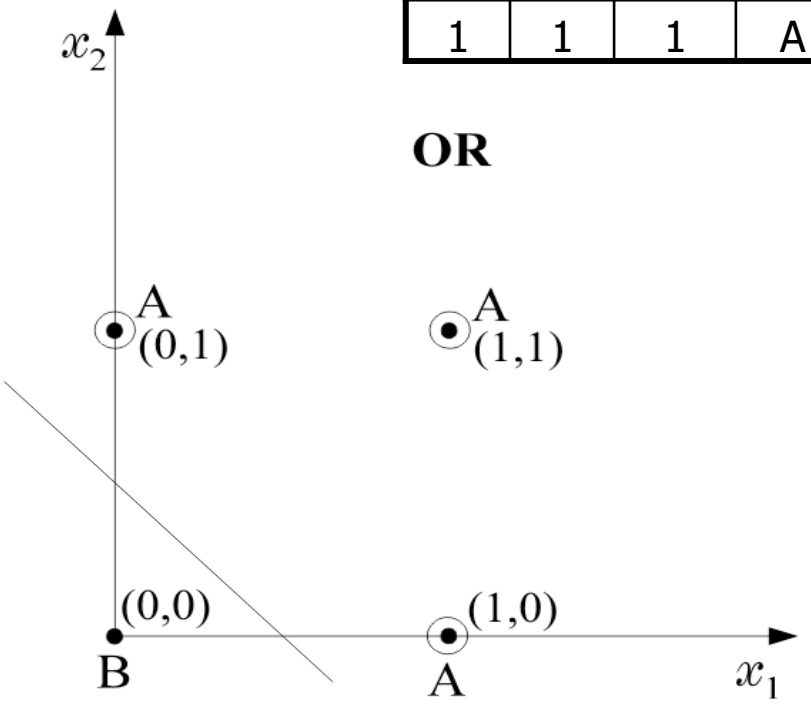
Implementing Logical Relations

- AND and OR operations are linearly separable problems

x_1	x_2	AND	Class
0	0	0	B
0	1	0	B
1	0	0	B
1	1	1	A



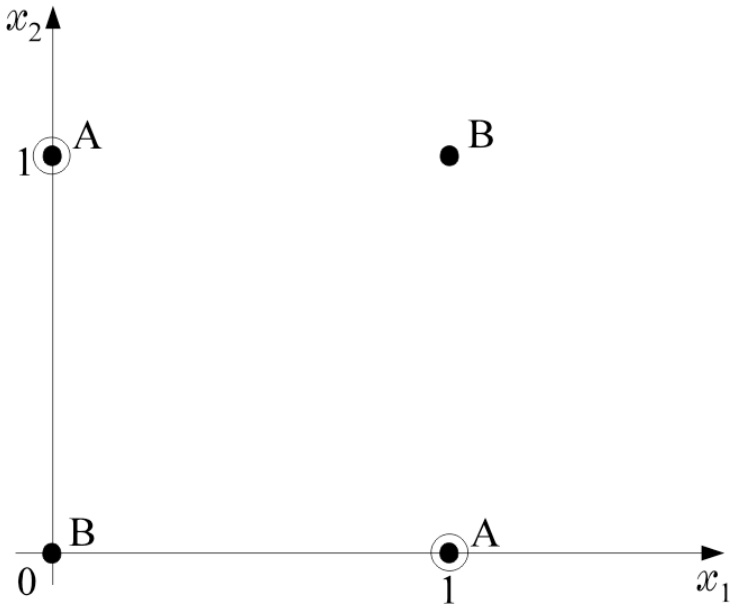
x_1	x_2	OR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	1	A



The XOR Problem

- XOR is not linearly separable

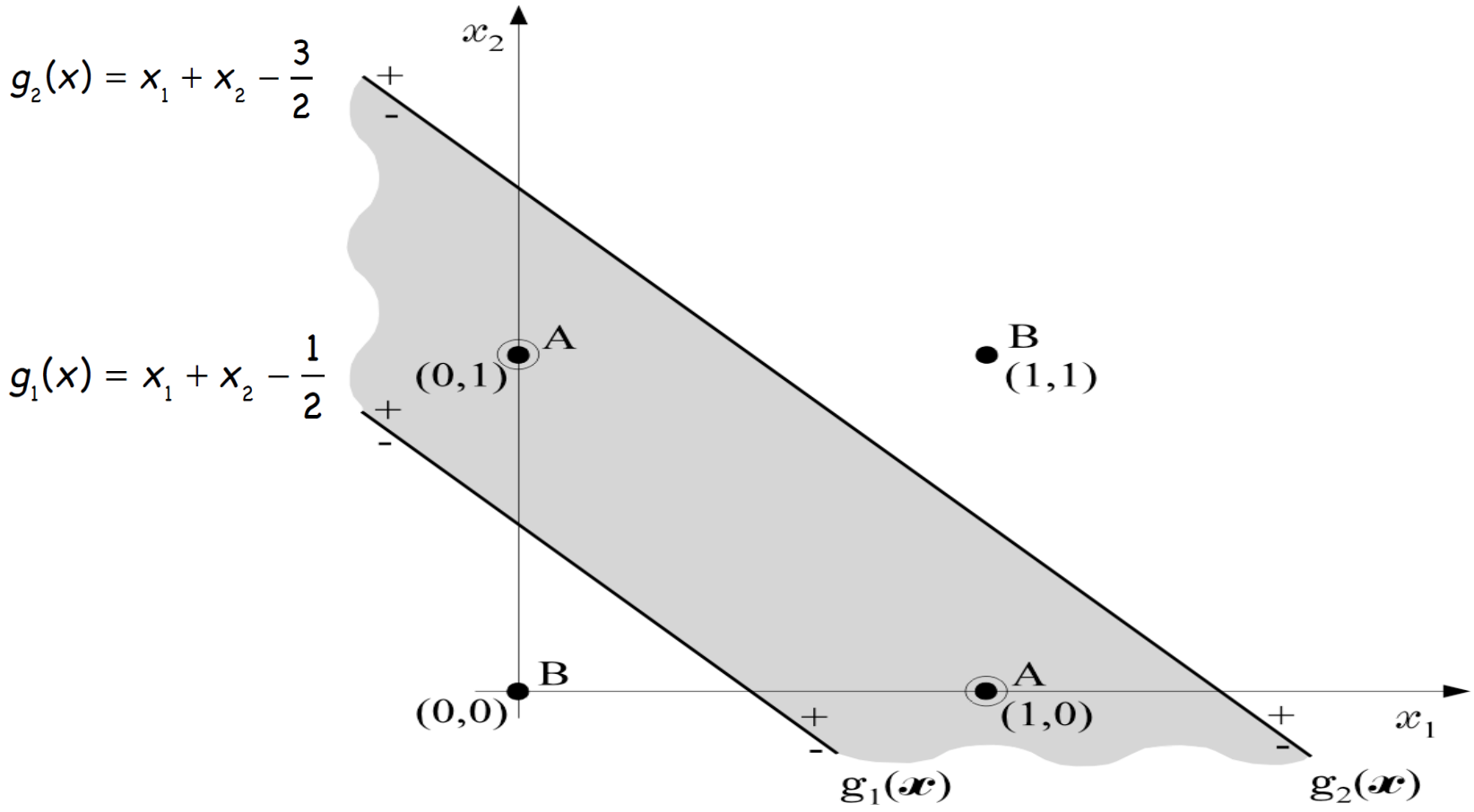
x_1	x_2	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B



- How can we use linear classifiers to solve this problem

Combining two linear classifiers

- Idea: use a logical combination of two linear classifiers.



Combining two linear classifiers

- Let $f(x)$ be the unit step activation function:

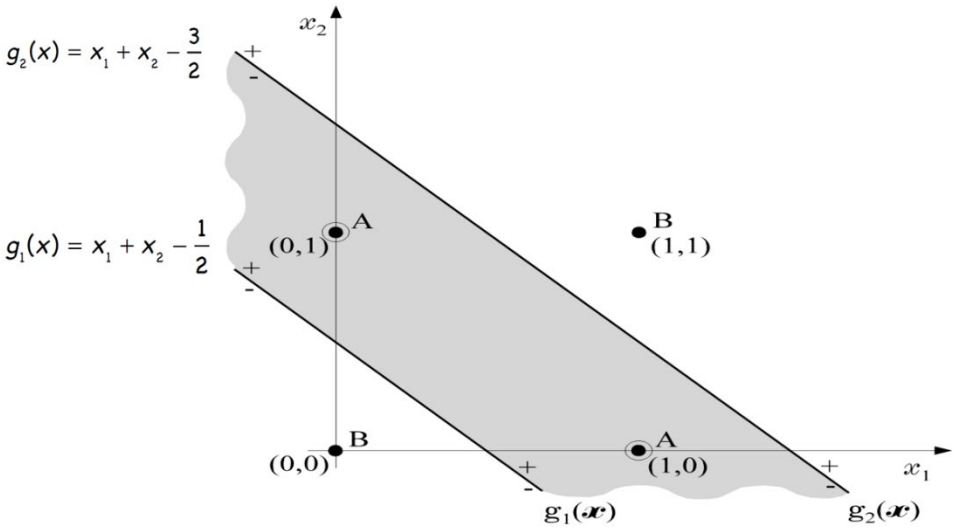
$$f(x) = 0, \quad x < 0$$
$$f(x) = 1, \quad x \geq 0$$

- Observe that the classification problem is then solved by

$$f\left(y_1 - y_2 - \frac{1}{2}\right)$$

where

$$y_1 = f\left(g_1(x)\right) \text{ and } y_2 = f\left(g_2(x)\right)$$



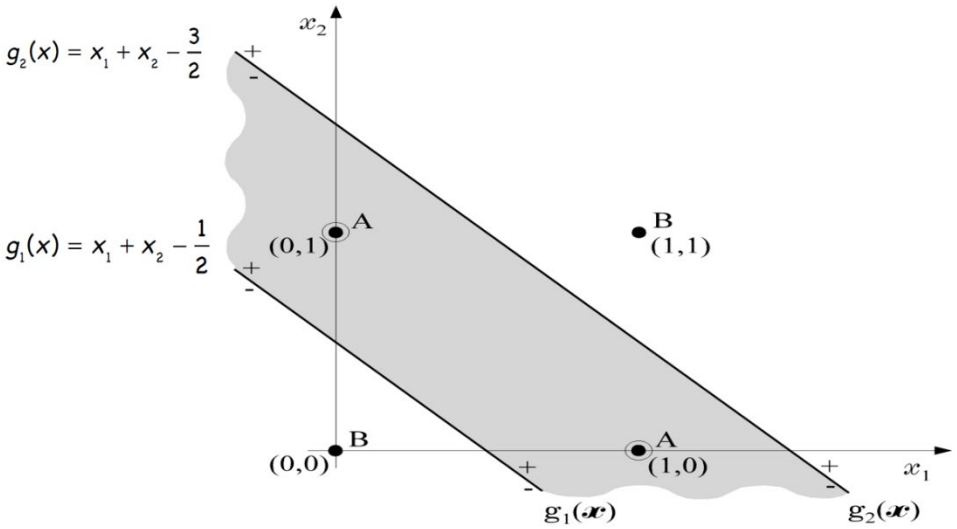
Combining two linear classifiers

- This calculation can be implemented sequentially:
  - Compute y1 and y2 from x1 and x2.
  - Compute the decision from y1 and y2.
- Each layer in the sequence consists of one or more linear classifications.
- This is therefore a two-layer perceptron.

$$f\left(y_1 - y_2 - \frac{1}{2}\right)$$

where

$$y_1 = f\left(g_1(x)\right) \text{ and } y_2 = f\left(g_2(x)\right)$$



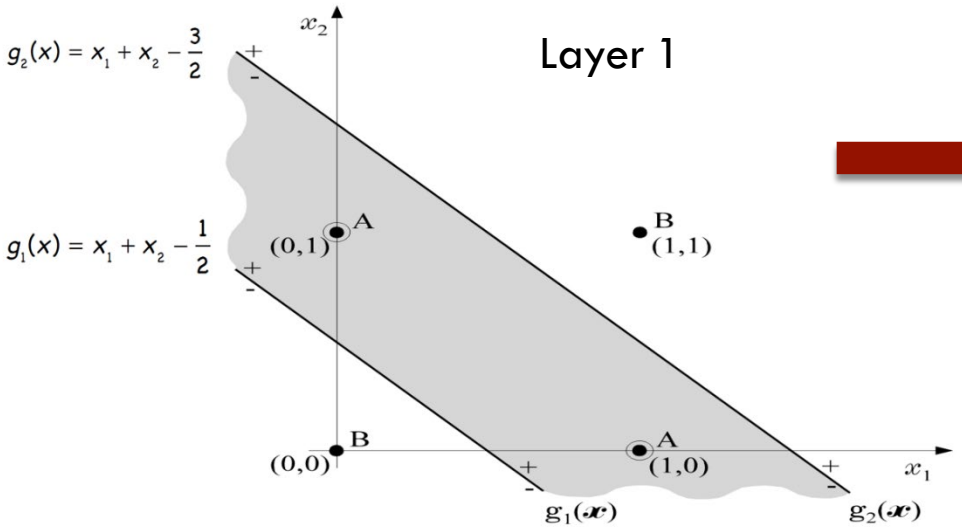
The Two-Layer Perceptron

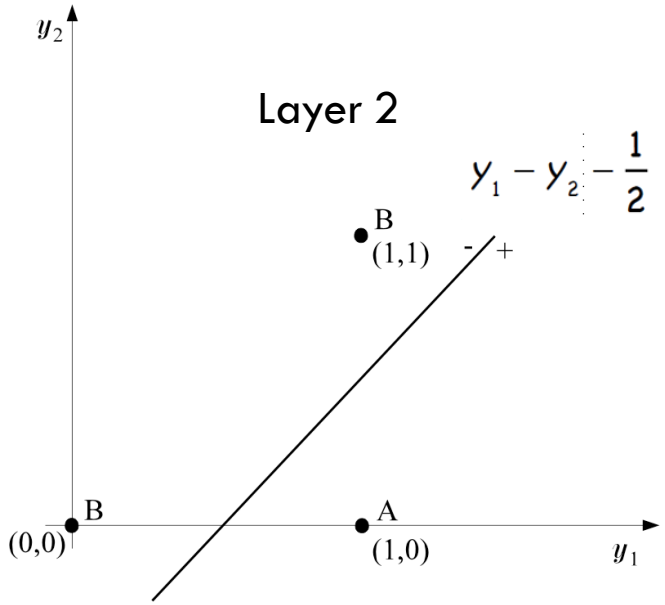
Layer 1				Layer 2
x_1	x_2	y_1	y_2	
0	0	0(-)	0(-)	B(0)
0	1	1(+)	0(-)	A(1)
1	0	1(+)	0(-)	A(1)
1	1	1(+)	1(+)	B(0)

$$f\left(y_1 - y_2 - \frac{1}{2}\right)$$

where

$$y_1 = f\left(g_1(x)\right) \text{ and } y_2 = f\left(g_2(x)\right)$$

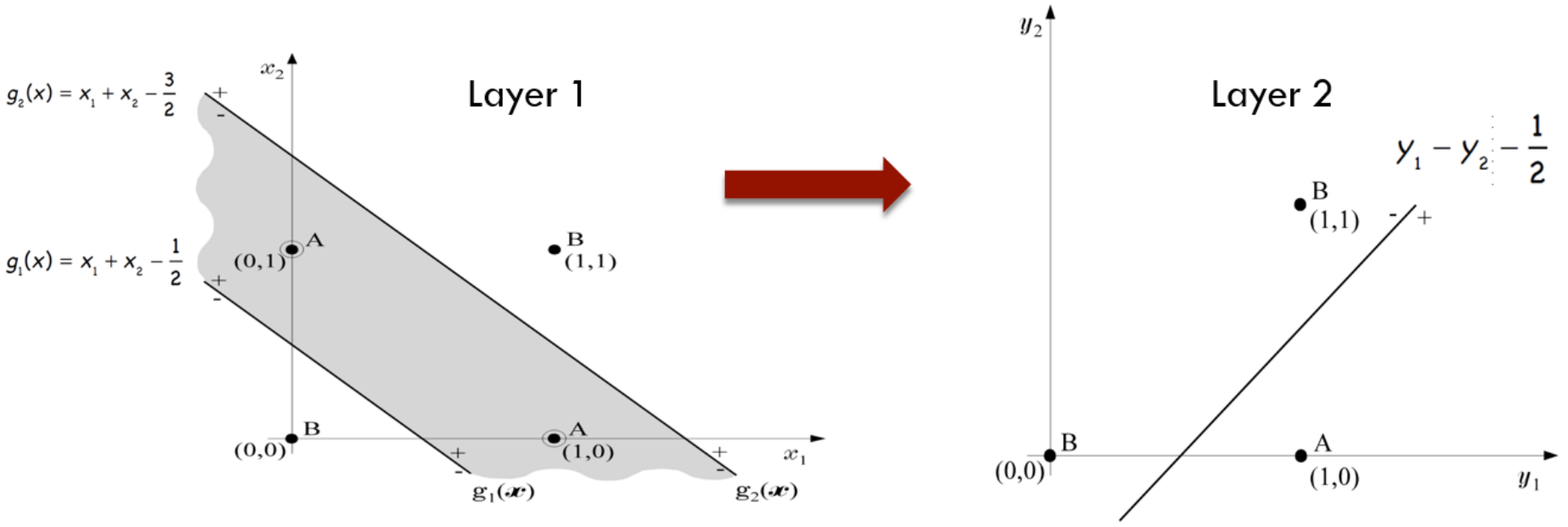




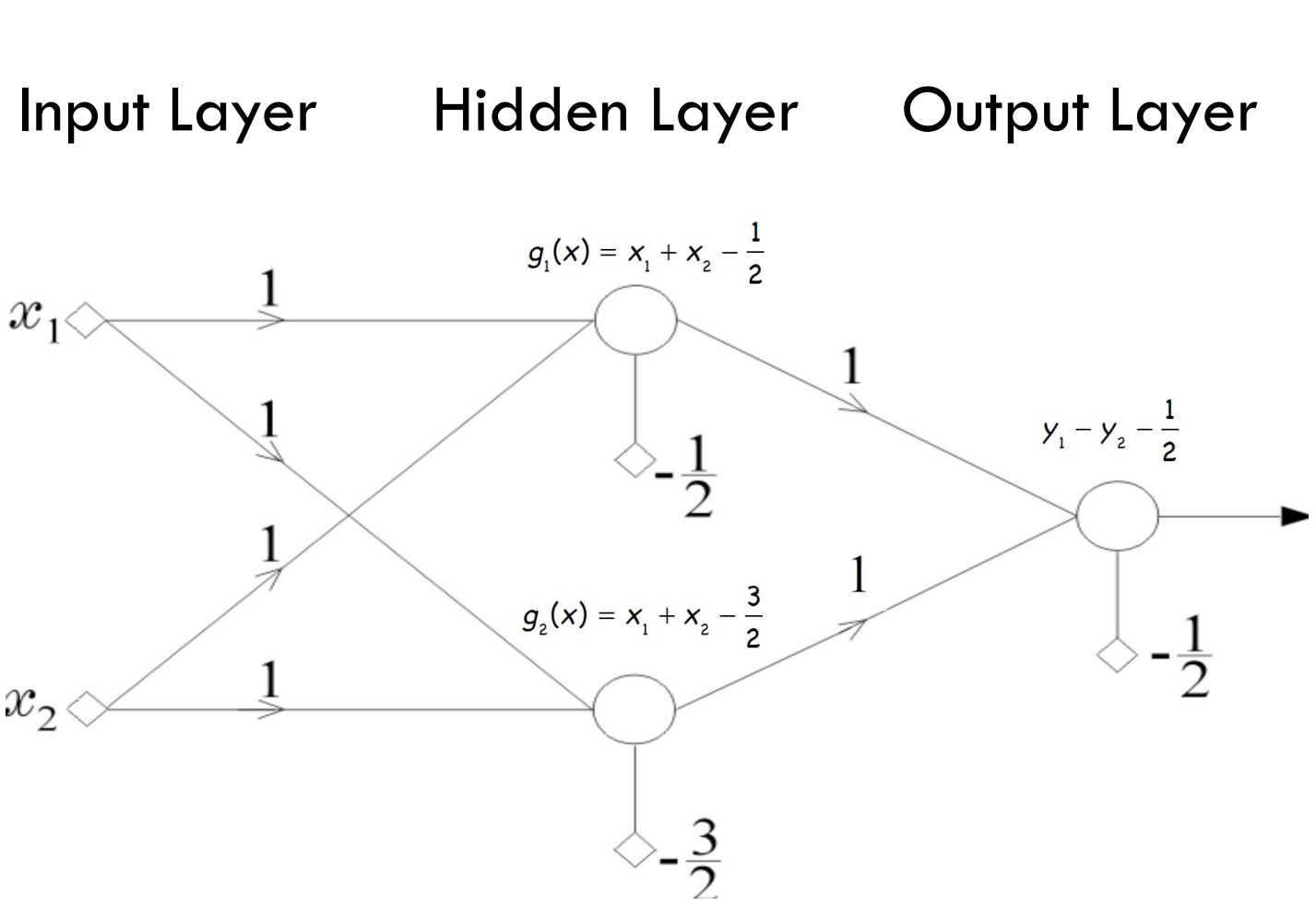
The Two-Layer Perceptron

- The first layer performs a nonlinear mapping that makes the data linearly separable.

$$y_1 = f\left(g_1(x)\right) \text{ and } y_2 = f\left(g_2(x)\right)$$



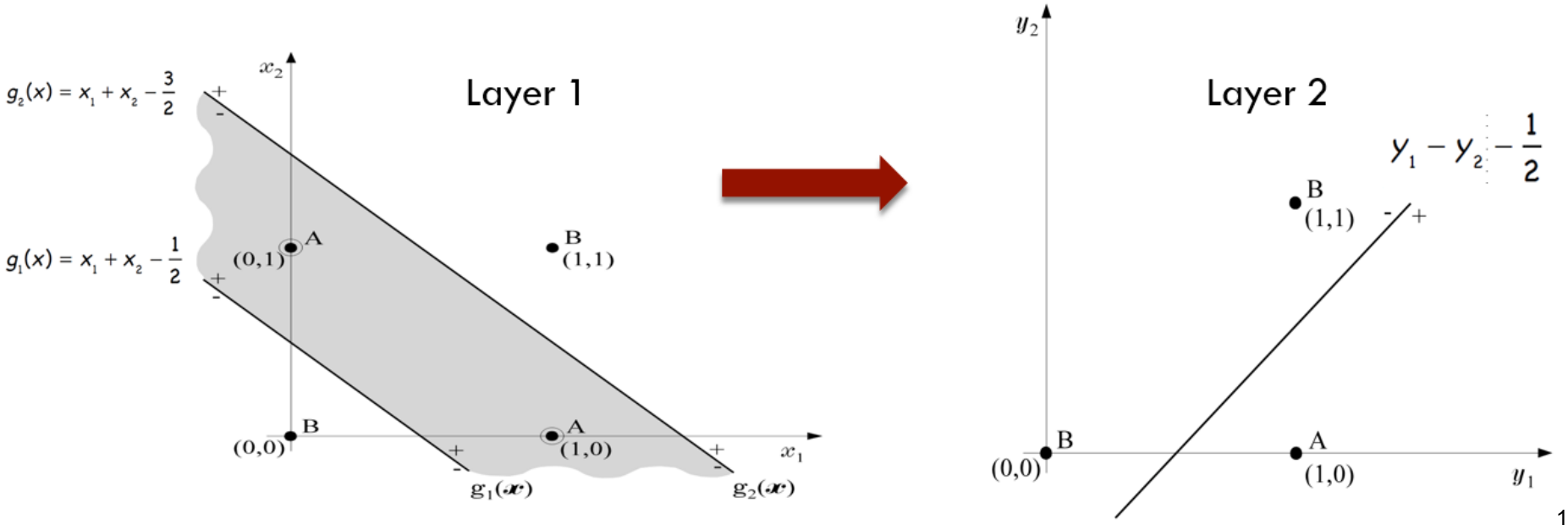
The Two-Layer Perceptron Architecture



The Two-Layer Perceptron

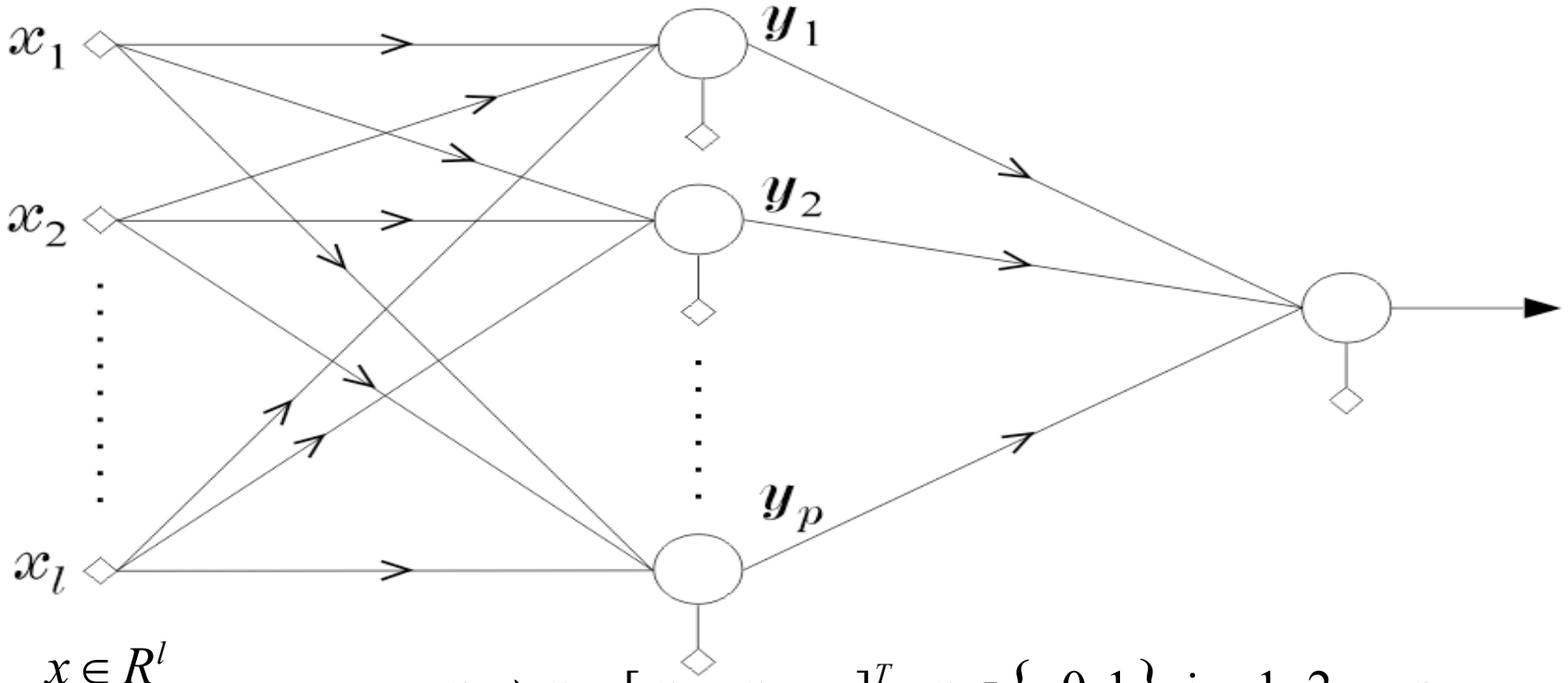
- Note that the hidden layer maps the plane onto the vertices of a unit square

$$y_1 = f\left(g_1(x)\right) \text{ and } y_2 = f\left(g_2(x)\right)$$



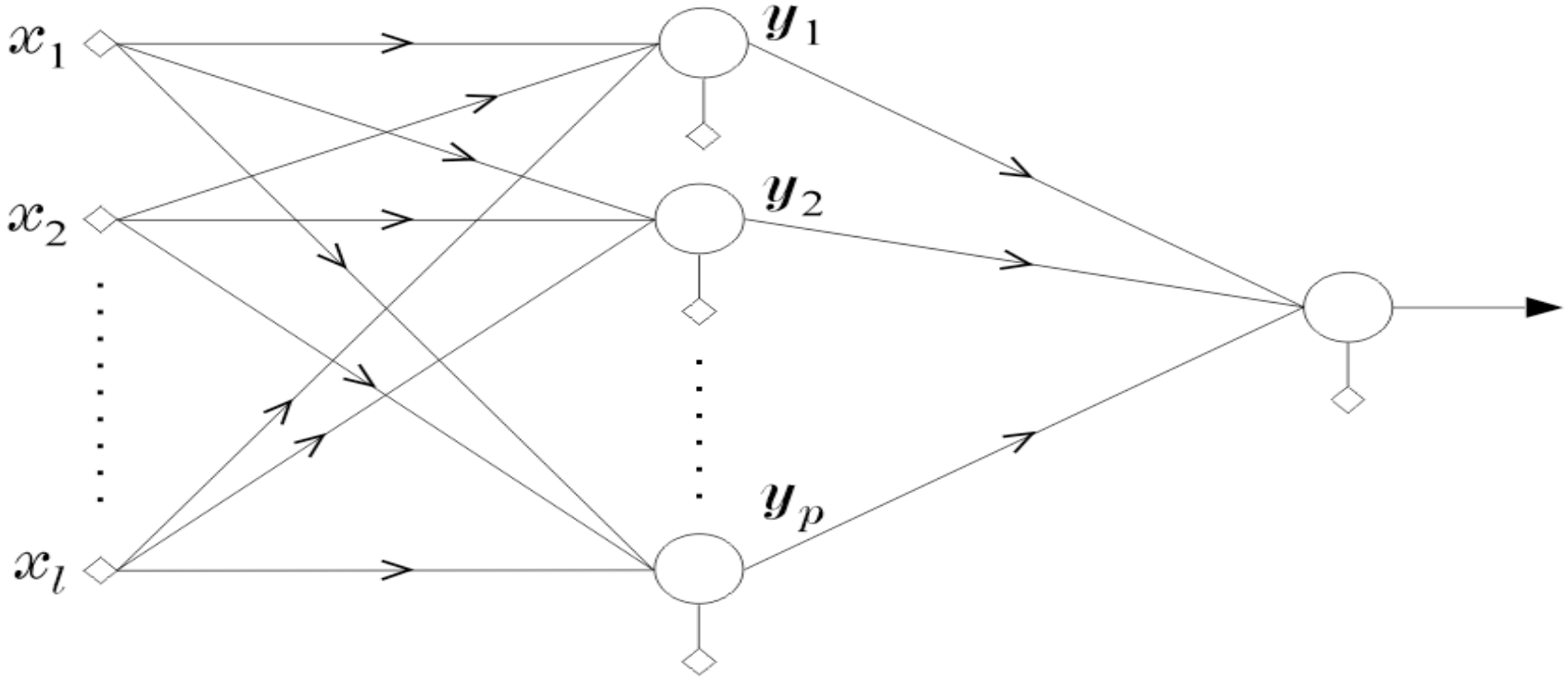
Higher Dimensions

- Each hidden unit realizes a hyperplane discriminant function.
- The output of each hidden unit is 0 or 1 depending upon the location of the input vector relative to the hyperplane.



Higher Dimensions

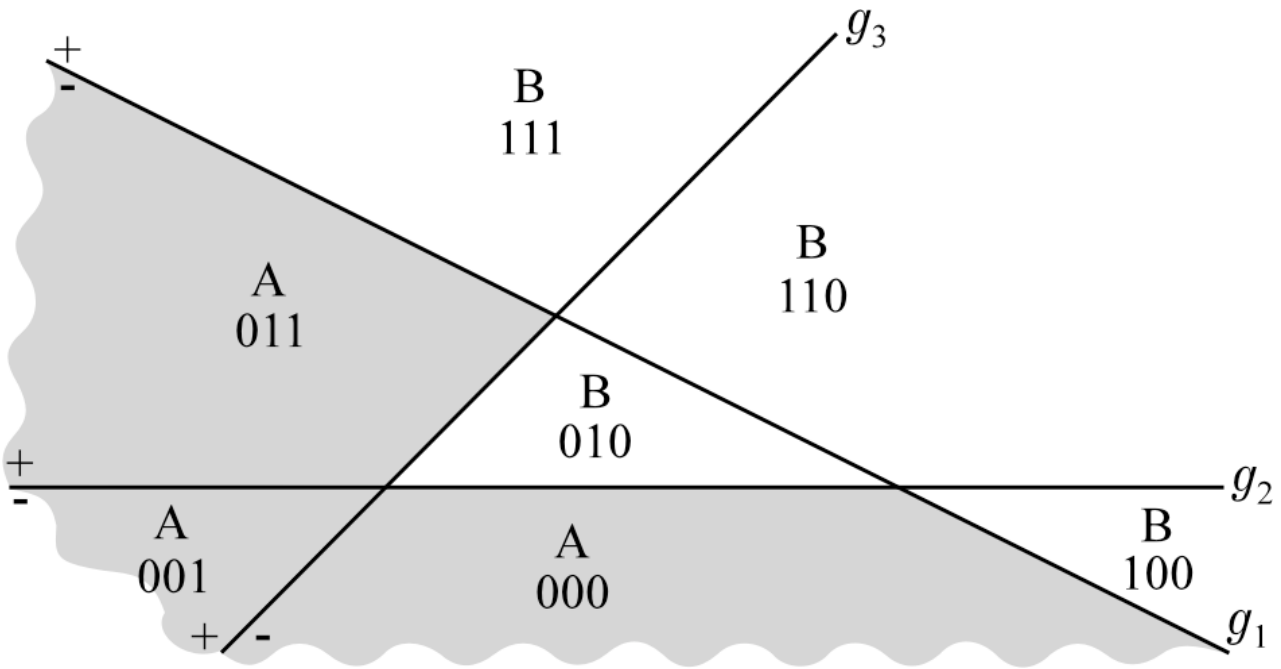
- Together, the hidden units map the input onto the vertices of a p-dimensional unit hypercube.



$\underline{x} \in R^l$ $\qquad$ $\underline{x} \to \underline{y} = [y_1, y_2, \ldots y_p]^T, y_i \in \{0,1\}\ i = 1, 2, \ldots p$

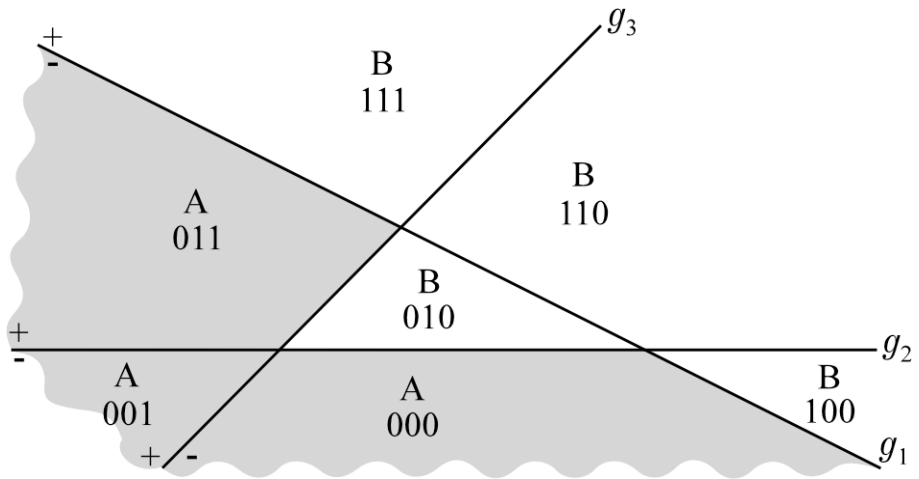
Two-Layer Perceptron

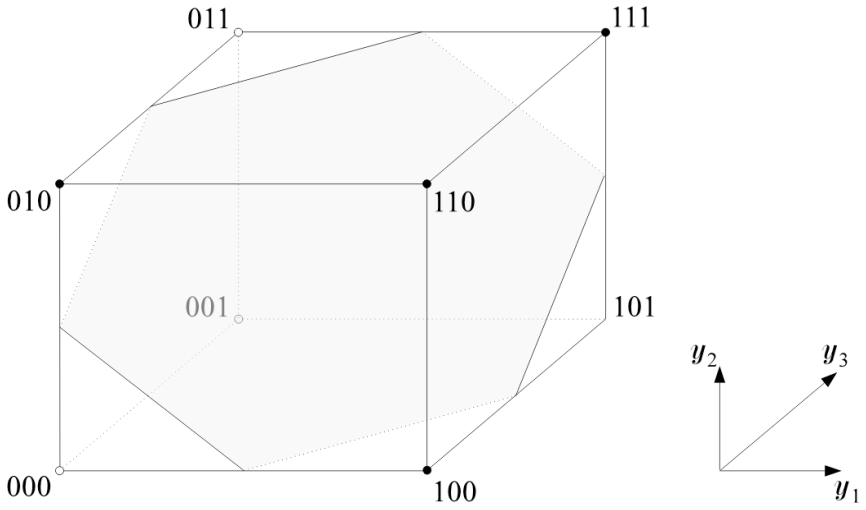
- These p hyperplanes partition the *l*-dimensional input space into polyhedral regions
- Each region corresponds to a different vertex of the *p*-dimensional hypercube represented by the outputs of the hidden layer.



Two-Layer Perceptron

- In this example, the vertex (0, 0, 1) corresponds to the region of the input space where:
  - $g_1(x) < 0$
  - $g_2(x) < 0$
  - $g_3(x) > 0$



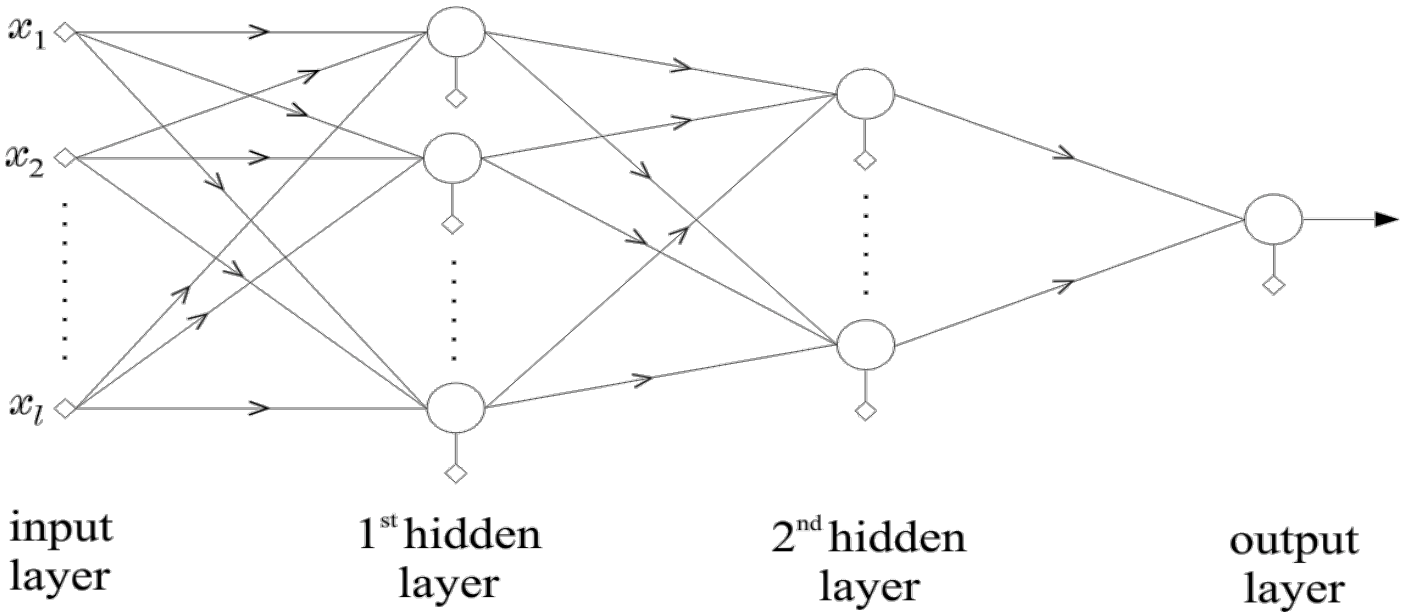


Limitations of a Two-Layer Perceptron

- The output neuron realizes a hyperplane in the transformed space that partitions the p vertices into two sets.
- Thus, the two layer perceptron has the capability to classify vectors into classes that consist of unions of polyhedral regions.
- But NOT ANY union. It depends on the relative position of the corresponding vertices.

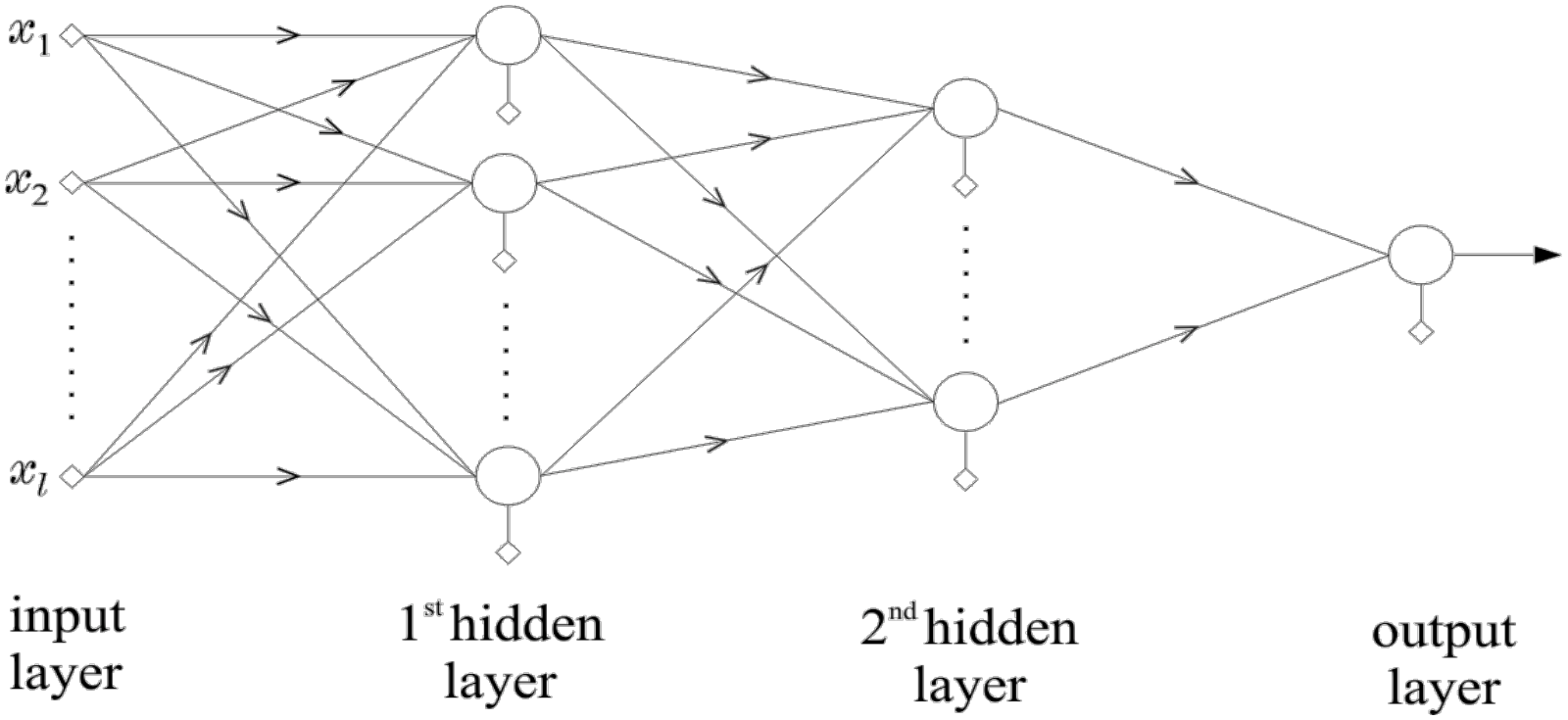
The Three-Layer Perceptron

- Suppose that Class A consists of the union of K polyhedral in the input space.
- Use K neurons in the 2nd hidden layer.
- Train each to classify one region as positive, the rest negative.
- Now use an output neuron that implements the OR function.



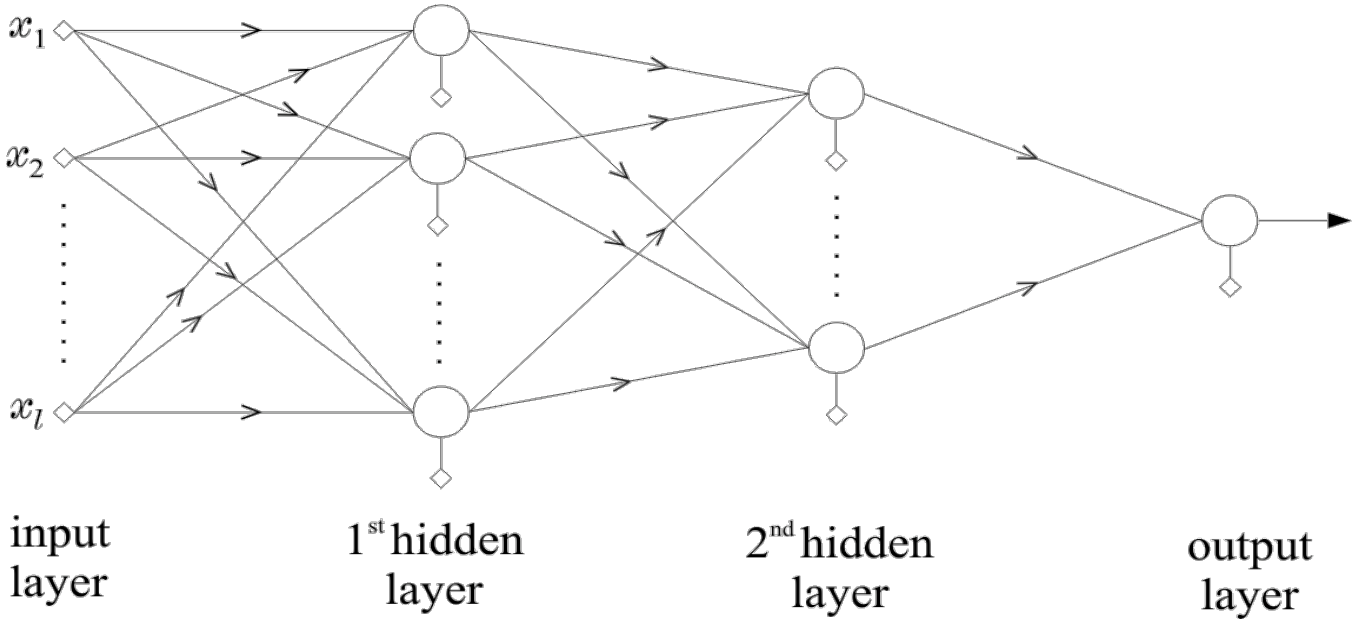
The Three-Layer Perceptron

- Thus the three-layer perceptron can separate classes resulting from any union of polyhedral regions in the input space.



The Three-Layer Perceptron

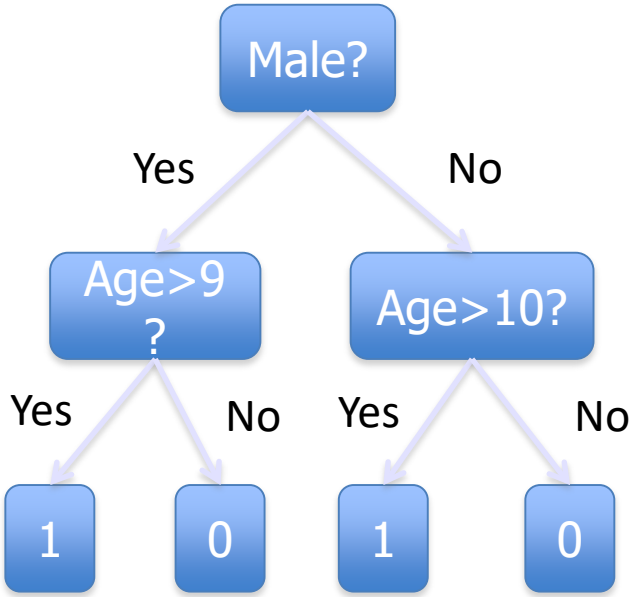
- The first layer of the network forms the hyperplanes in the input space.
- The second layer of the network forms the polyhedral regions of the input space
- The third layer forms the appropriate unions of these regions and maps each to the appropriate class.



Outline

- Non-linear perceptron
- Classifier ensemble

Why ensemble



Person	Age	Male?	Height > 55”
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	8	0	0

$$x = \begin{bmatrix} age \\ 1_{[gender=male]} \end{bmatrix} \qquad y = \begin{cases} 1 & height > 55" \\ 0 & height \leq 55" \end{cases}$$

Why ensemble

- Performance
  - None of the classifiers is perfect
  - Complementary
  - Examples which are not correctly classified by one classifier may be correctly classified by the other classifiers
- Potential Improvements?
  - Utilize the complementary property

Why ensemble

- Idea
  - Combine the classifiers to improve the performance
- Ensembles of Classifiers
  - Combine the classification results from different classifiers to produce the final output
  - Unweighted voting
  - Weighted voting

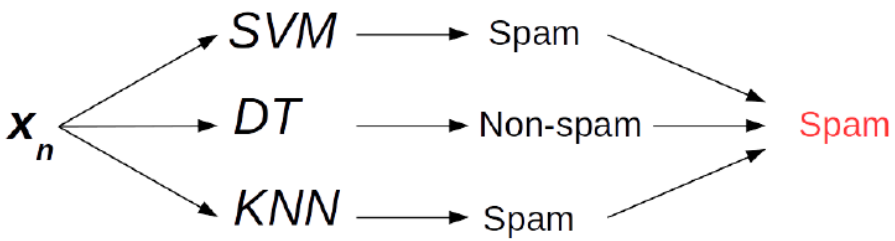
Ensemble Methods

- Construct a set of base classifiers learned from the training data

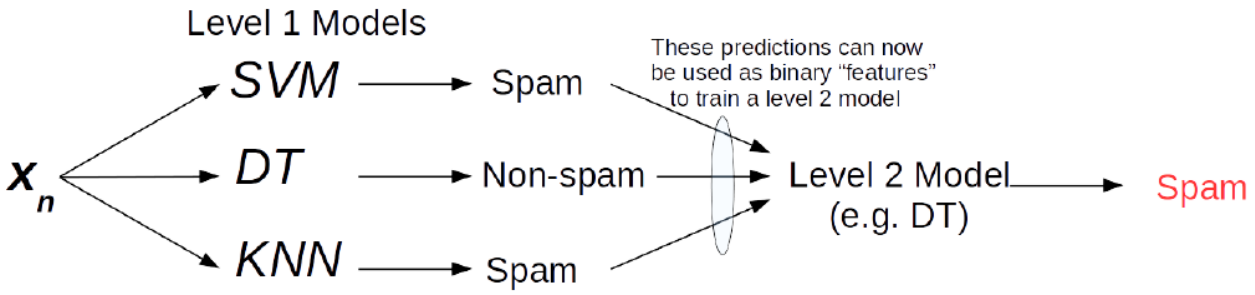
- Predict class label of test records by combining the predictions made by multiple classifiers (e.g., by taking majority vote)

Some Simple Ensembles

- Voting or Averaging of predictions of multiple pre-trained models

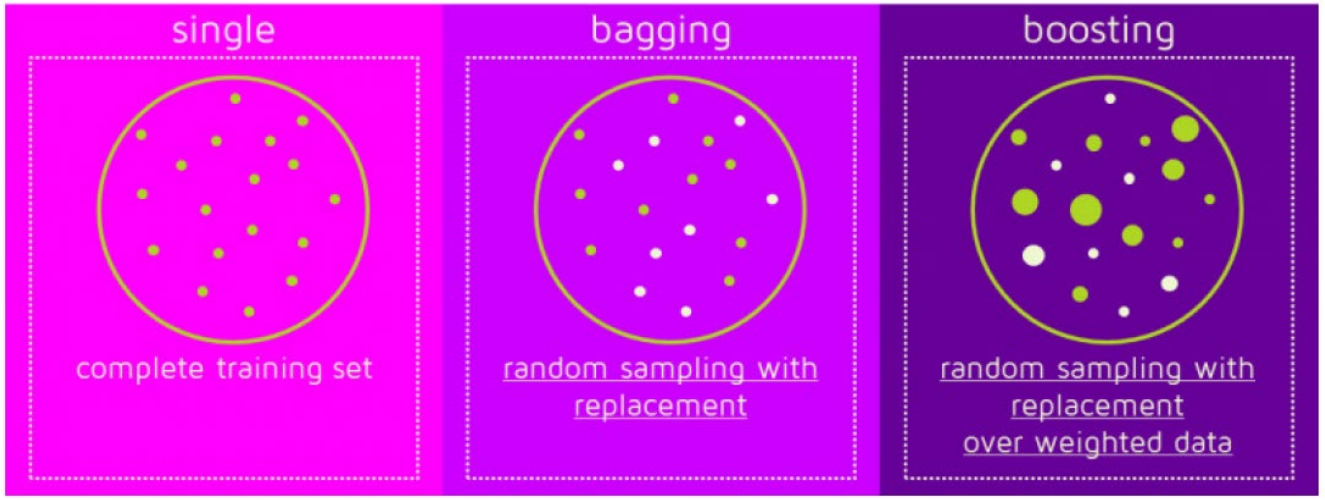


- Stacking: Use predictions of multiple models as features to train a new model and use the new model to make predictions on test data



Ensembles: Another Approach

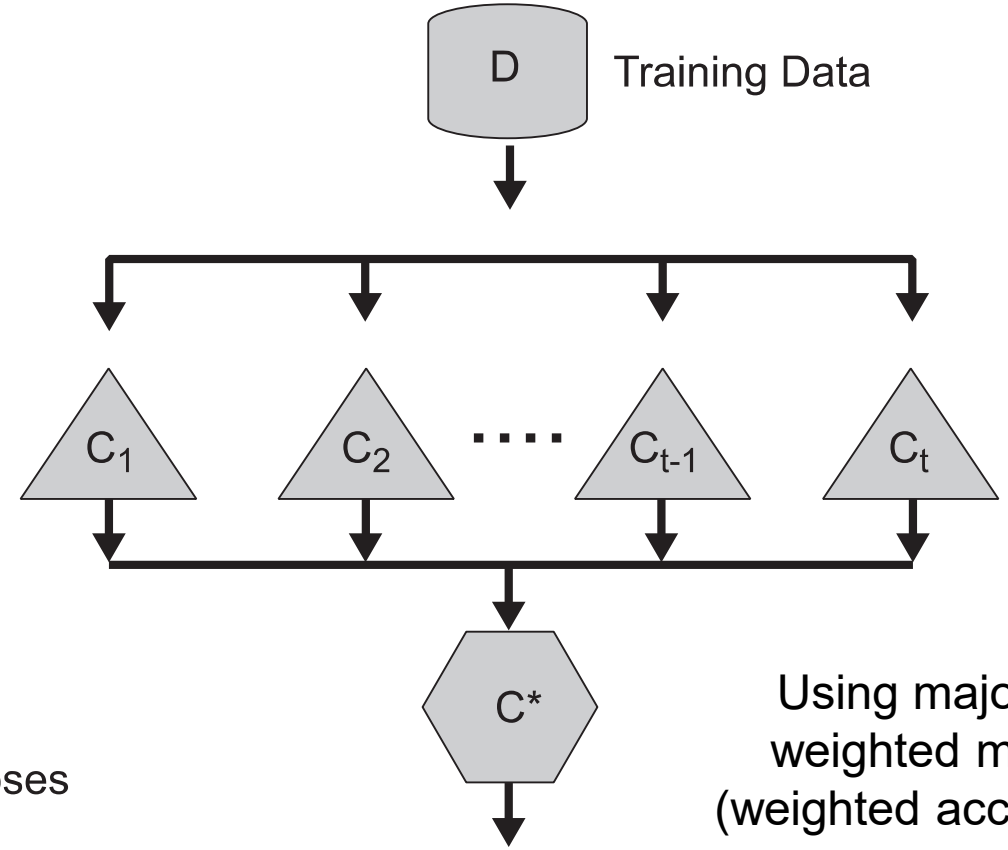
- Instead of training different models on same data, train same model multiple times on different data sets, and combine these different models
- We can use some simple/weak model as the base model
- How do we get multiple training data sets (in practice, we only have one data set at training time)?



Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
  - All base classifiers are independent of each other
  - All base classifiers perform better than random guessing (error rate < 0.5 for binary classification)
- Ensemble Methods work best with **unstable base classifiers**
  - Classifiers that are sensitive to minor perturbations in training set, due to *high model complexity*
  - Examples: Unpruned decision trees, ANNs, …

General Approach of Ensemble Learning



Constructing Ensemble Classifiers

- By manipulating training set
  - Example: bagging, boosting, random forests
- By manipulating input features
  - Example: random forests
- By manipulating class labels
  - Example: error-correcting output coding
- By manipulating learning algorithm
  - Example: injecting randomness in the initial weights of ANN

Bagging (Bootstrap AGGregatING)

- Bagging stands for Bootstrap Aggregation
- Takes original data set D with N training examples
- Creates M copies $\{\tilde{D}_m\}_{m=1}^{M}$
  - Each $\tilde{D}_m$ is generated from D by sampling with replacement
  - Each data set $\tilde{D}_m$ has the same number of examples as in data set D
  - These data sets are reasonably different from each other (since only about 63% of the original examples appear in any of these data sets)
- Train models $h_1, \ldots, h_M$ using $\tilde{D}_1, \ldots, \tilde{D}_M$, respectively
- Use an averaged model $h = \frac{1}{M} \sum_{m=1}^{M} h_m$ as the final model
- Useful for models with high variance and noisy data

Bagging (Bootstrap AGGregatING)

- Bootstrap sampling: sampling with replacement

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
- Each sample has probability $(1 – 1/n)^n$ of not being selected as test data
- Training data = $1- (1 – 1/n)^n$ of the original data

Bagging (Bootstrap AGGregatING)

- This method is also called the 0.632 bootstrap
  - A particular training data has a probability of 1-1/n of not being picked
  - Thus its probability of ending up in the test data (not selected) is

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} = 0.368$$

  - This means the training data will contain approximately 63.2% of the instances

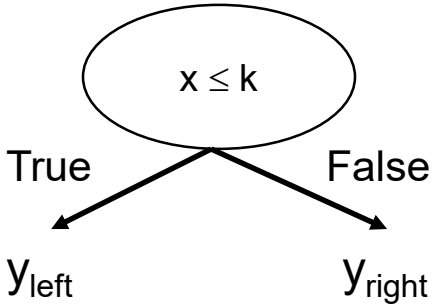
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump (decision tree of size 1)
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

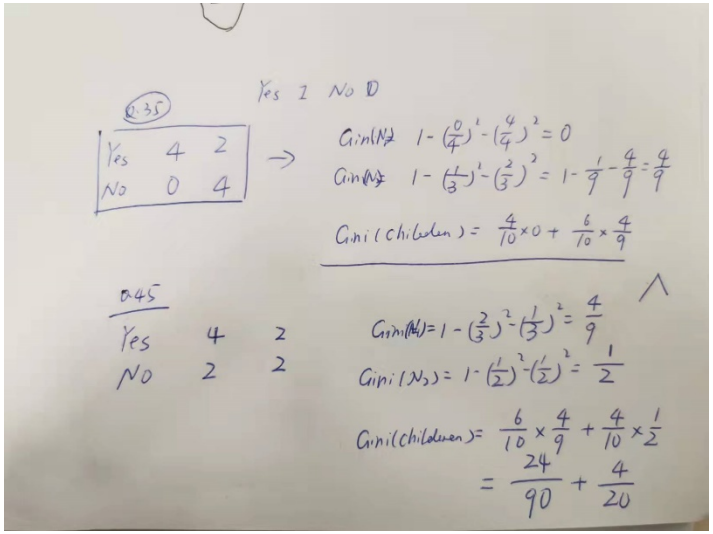
x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

- Sort the attribute on values
- Linearly scan these values, each time updating the count matrix and computing gini index
- Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income										
Sorted Values → 60	60	70	75	85	90	95	100	120	125	220
Split Positions → 55	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes				0 3							
No				3 4							
Gini				0.343							

Gini(N1) = 1 – (0/3)² – (3/3)² = 0
Gini(N2) = 1 – (3/7)² – (4/7)² = 0.4898

Gini(Children) = 3/10*0+ 7/10 * 0.4898
= 0.343



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 → y = 1
x > 0.35 → y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 → y = 1
x > 0.7 → y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 → y = 1
x > 0.35 → y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 → y = 1
x > 0.3 → y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 → y = 1
x > 0.35 → y = -1

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Trained Decision Stumps:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Use majority vote (sign of sum of predictions) to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

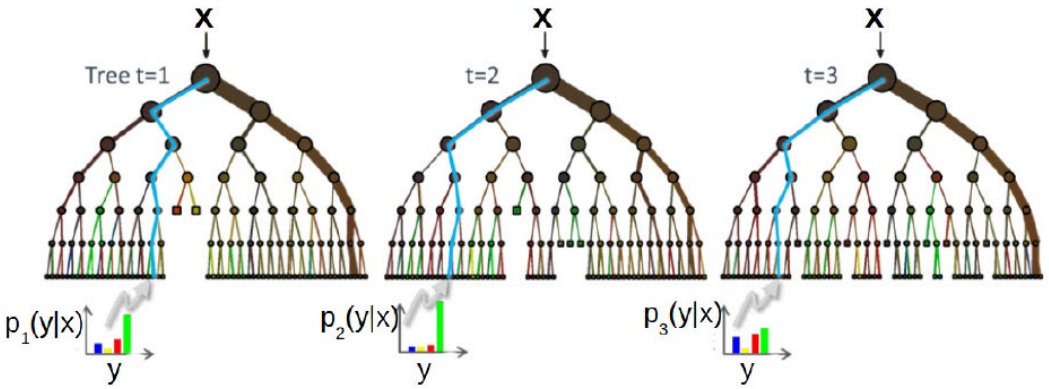
- Bagging can also increase the complexity (representation capacity) of simple classifiers such as decision stumps

Random Forests

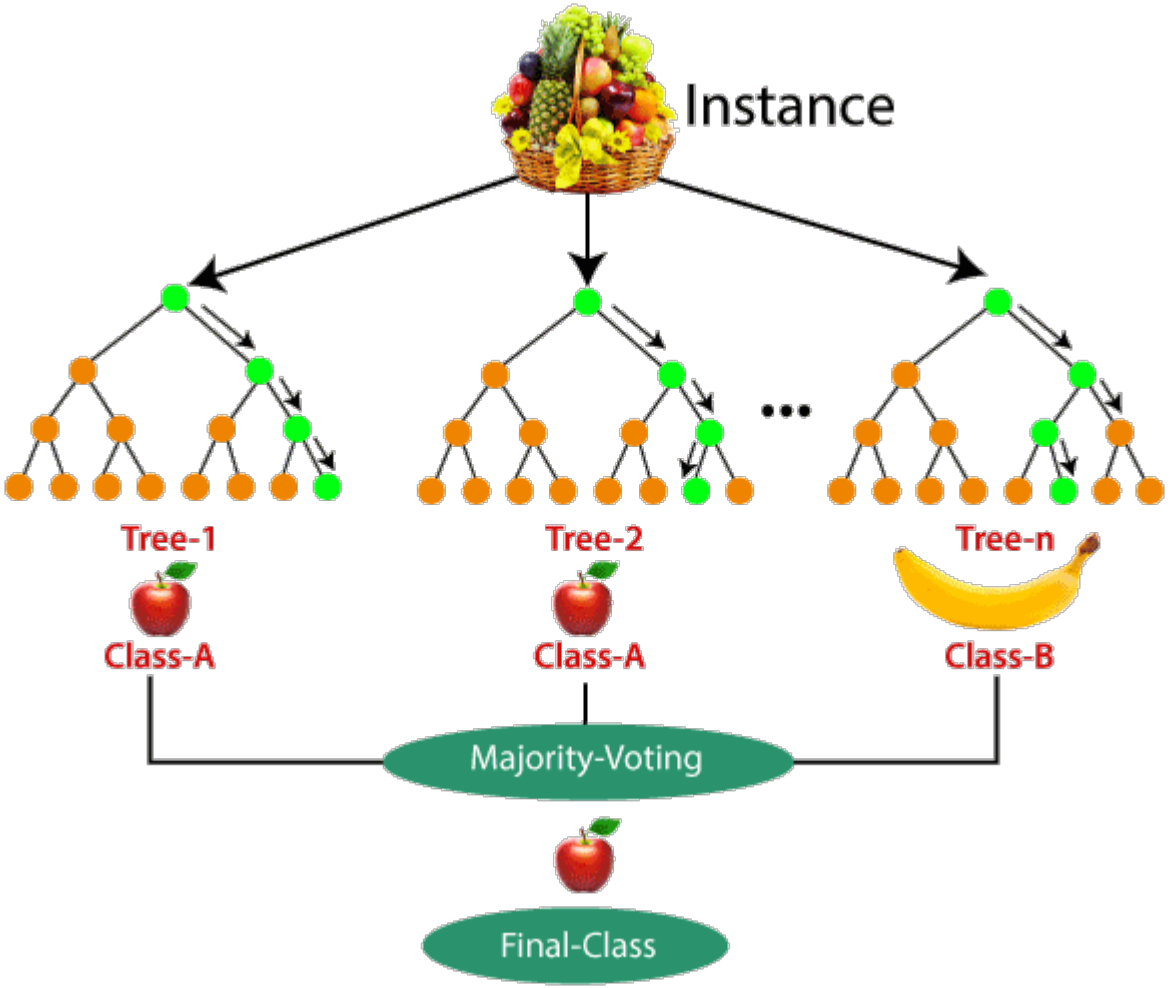
- Construct an ensemble of decision trees by manipulating training set as well as features
  - Use bootstrap sample to train every decision tree (similar to Bagging)
  - Use the following tree induction algorithm:
    - At every internal node of decision tree, randomly sample p attributes for selecting split criterion
    - Repeat this procedure until all leaves are pure (unpruned tree)

Random Forests

- An ensemble of decision tree (DT) classifiers
- Uses bagging on features (each DT will use a random set of features)
  - Given a total of D features, each DT uses sqrt (D) randomly chosen features
  - Randomly chosen features make the different trees uncorrelated
- All DTs usually have the same depth
- Each DT will split the training data dierently at the leaves
- Prediction for a test example votes on/averages predictions from all the DTs



Random Forests



Boosting

- The basic idea
  - Take a weak learning algorithm
  - Only requirement: Should be slightly better than random
  - Turn it into an awesome one by making it focus on difficult cases
- Most boosting algorithms follow these steps:
  - 1 Train a weak model on some training data
  - 2 Compute the error of the model on each training example
  - 3 Give higher importance to examples on which the model made mistakes
  - 4 Re-train the model using importance weighted training examples
  - 5 Go back to step 2

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 (for being selected for training)
  - Unlike bagging, weights may change at the end of each boosting round

Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

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

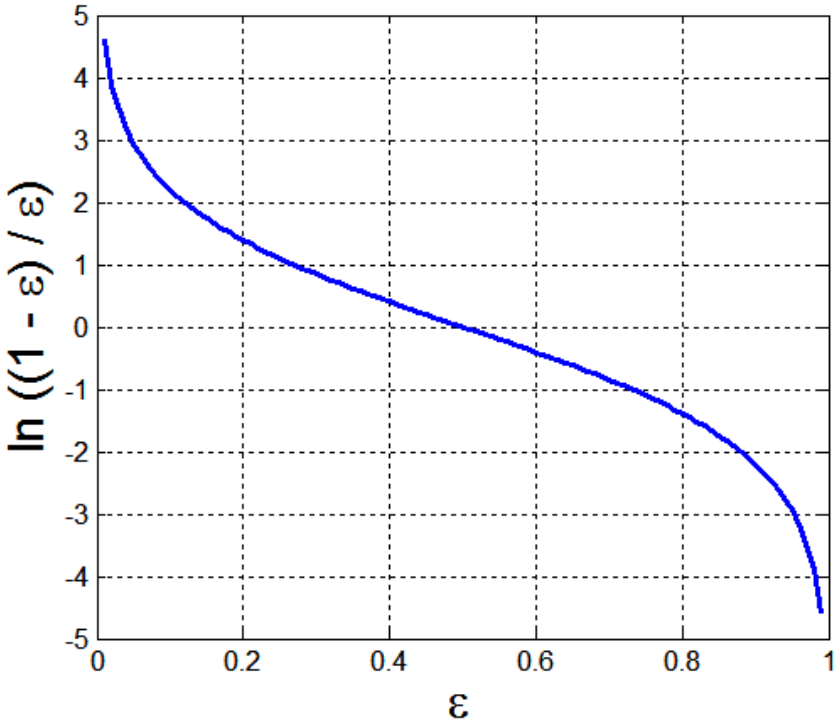
- Base classifiers: C_1, C_2, …, C_T

- Error rate of a base classifier:

$$\epsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j^{(i)}\, \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

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



AdaBoost Algorithm 1

- Weight update:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ e^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

Where Z_i is the normalization factor

- 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:

$$C^*(x) = \arg\max_y \sum_{i=1}^{T} \alpha_i \delta(C_i(x) = y)$$

Boosting

- Given: Training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $y_n \in \{-1, +1\}$, $\forall n$
- Initialize weight of each example $(\boldsymbol{x}_n, y_n)$: $D_1(n) = 1/N$, $\forall n$
- For round $t = 1 : T$
  - Learn a weak $h_t(\boldsymbol{x}) \rightarrow \{-1, +1\}$ using training data weighted as per D_t
  - Compute the weighted fraction of errors of h_t on this training data

$$\epsilon_t = \sum_{n=1}^{N} D_t(n) \mathbb{1}[h_t(\boldsymbol{x}_n) \neq y_n]$$

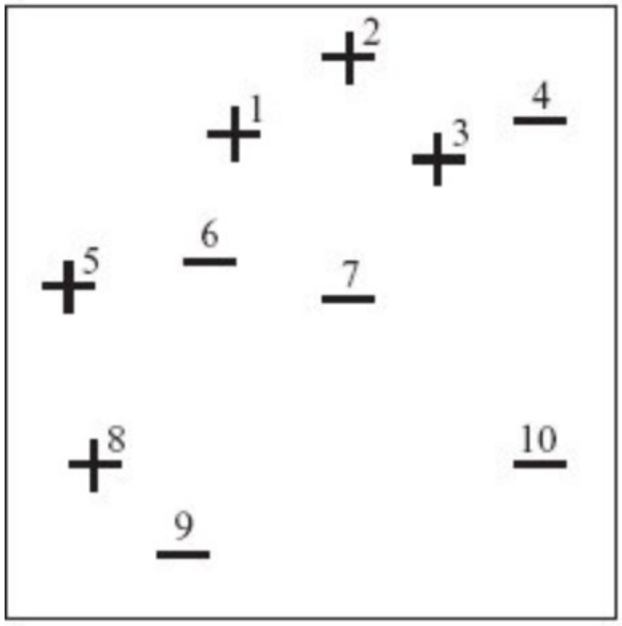
  - Set "importance" of h_t: $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$(gets larger as ϵ_t gets smaller)
  - Update the weight of each example

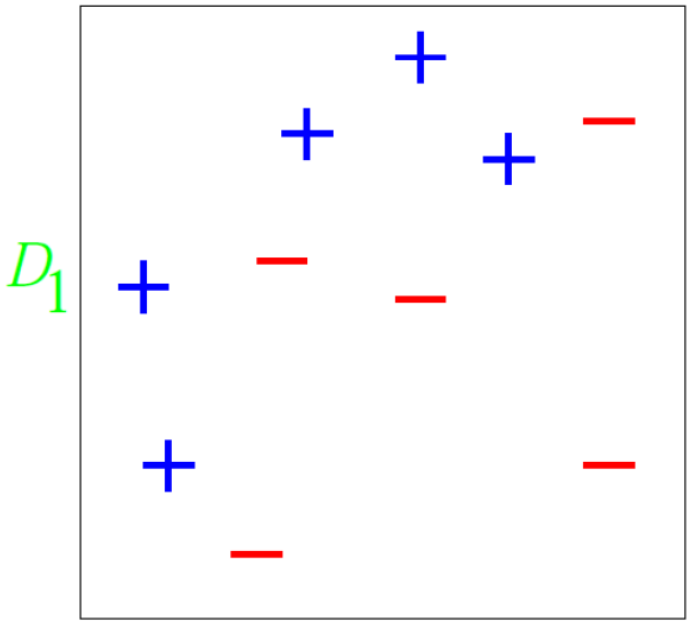
$$\begin{aligned} D_{t+1}(n) &\propto \begin{cases} D_t(n) \times \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) = y_n \quad \text{(correct prediction: decrease weight)} \\ D_t(n) \times \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) \neq y_n \quad \text{(incorrect prediction: increase weight)} \end{cases} \\ &= D_t(n) \exp(-\alpha_t y_n h_t(\boldsymbol{x}_n)) \end{aligned}$$

  - Normalize D_{t+1} so that it sums to 1: $D_{t+1}(n) = \frac{D_{t+1}(n)}{\sum_{m=1}^{N} D_{t+1}(m)}$
- Output the "boosted" final hypothesis $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

AdaBoost Example 1

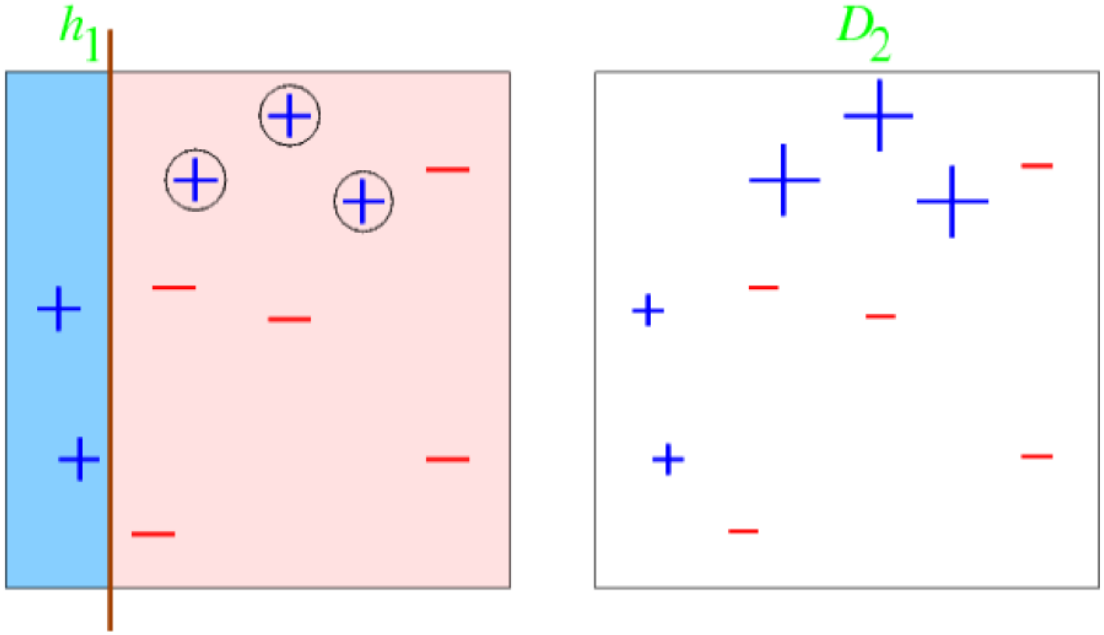
- Consider binary classification with 10 training examples
- Initial weight distribution D1 is uniform (each point has equal weight = 1/10)





AdaBoost Example 1

- After round 1



Error rate of h_1: $\epsilon_1 = 0.3$; weight of h_1: $\alpha_1 = \frac{1}{2}\ln((1-\epsilon_1)/\epsilon_1) = 0.42$

Each misclassified point upweighted (weight multiplied by $\exp(\alpha_2)$) exp(0.42)=1.522

Each correctly classified point downweighted (weight multiplied by $\exp(-\alpha_2)$) exp(-0.42)=0.6570

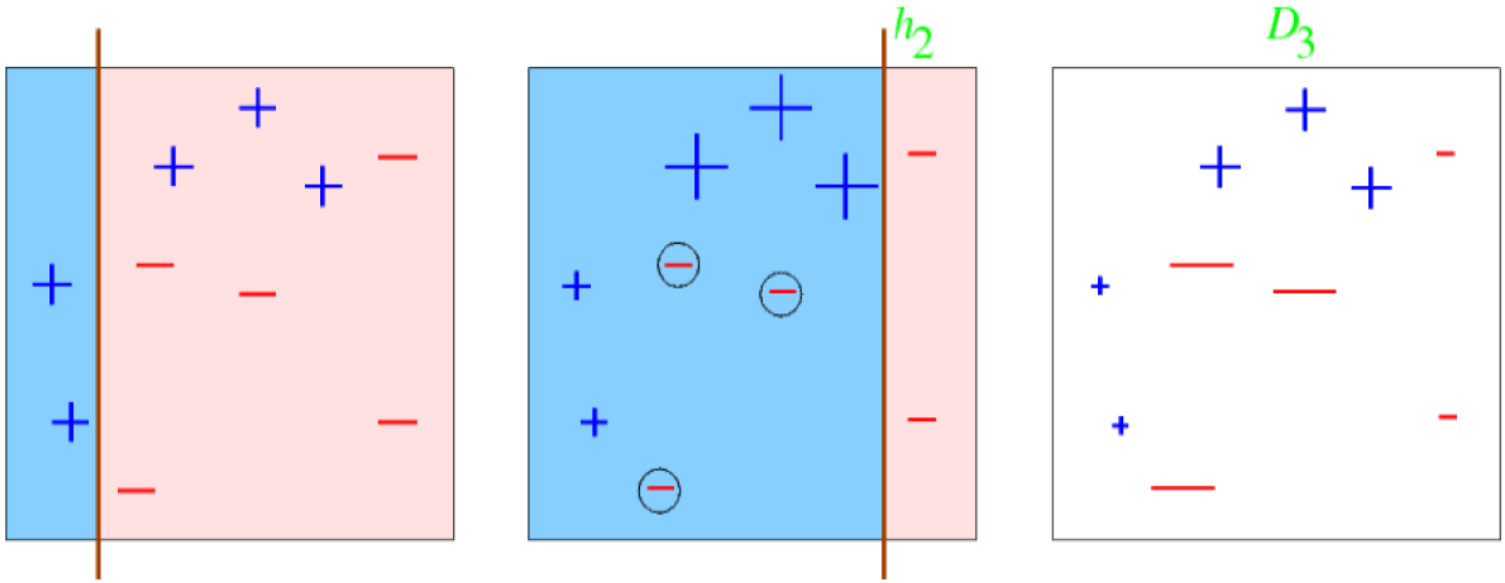
Normalization: 3 samples with 0.1522 7 samples with 0.06570

Then the weights will be changed to 0.1522/(0.1522*3+0.0657*7)=0.1661~0.17;

0.0657 /(0.1522*3+0.0657*7)=0.0717

AdaBoost Example 1

- After round 2



Error rate of h_2: $\epsilon_2 = 0.21$; weight of h_2: $\alpha_2 = \frac{1}{2}\ln((1-\epsilon_2)/\epsilon_2) = 0.65$

Each misclassified point upweighted (weight multiplied by $\exp(\alpha_2)$) exp(0.65)=1.9155

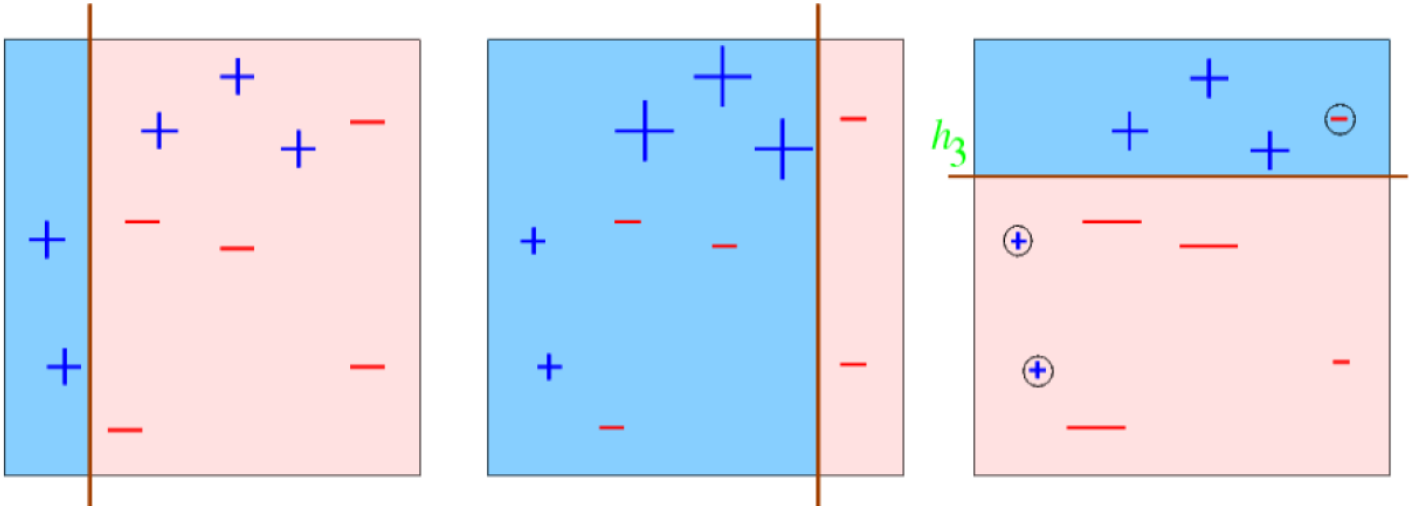
Each correctly classified point downweighted (weight multiplied by $\exp(-\alpha_2)$) exp(-0.65)=0.5220

$D_2(i)$	0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$\epsilon_2 \approx 0.21, \alpha_2 \approx 0.65$
$e^{-\alpha_2 y_i h_2(x_i)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
$D_2(i)\, e^{-\alpha_2 y_i h_2(x_i)}$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$

Used for normalization

AdaBoost Example 1

- After round 3



Error rate of h_3: $\epsilon_3 = 0.14$; weight of h_3: $\alpha_3 = \frac{1}{2}\ln((1-\epsilon_3)/\epsilon_3) = 0.92$

Suppose we decide to stop after round 3 exp(0.92)=2.51

Our ensemble now consists of 3 classifiers: h_1, h_2, h_3 exp(-0.92)=0.40

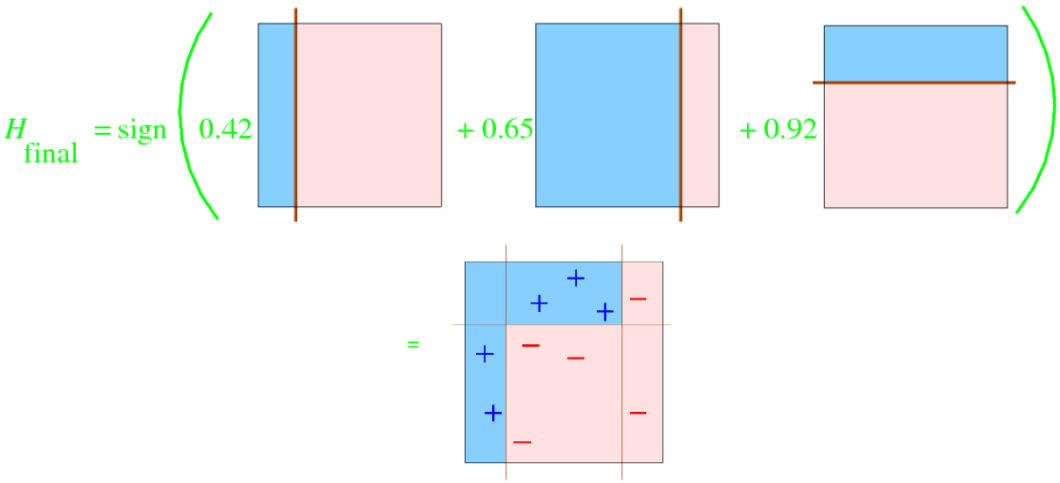
$D_3(i)$	0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05	$\epsilon_3 \approx 0.14, \alpha_3 \approx 0.92$
$e^{-\alpha_3 y_i h_3(x_i)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
$D_3(i)\ e^{-\alpha_3 y_i h_3(x_i)}$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

AdaBoost Example 1

	1	2	3	4	5	6	7	8	9	10	
$D_1(i)$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	$\epsilon_1 = 0.30, \alpha_1 \approx 0.42$
$e^{-\alpha_1 y_i h_1(x_i)}$	1.53	1.53	1.53	0.65	0.65	0.65	0.65	0.65	0.65	0.65	
$D_1(i)\ e^{-\alpha_1 y_i h_1(x_i)}$	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$Z_1 \approx 0.92$
$D_2(i)$	0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$\epsilon_2 \approx 0.21, \alpha_2 \approx 0.65$
$e^{-\alpha_2 y_i h_2(x_i)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
$D_2(i)\ e^{-\alpha_2 y_i h_2(x_i)}$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$
$D_3(i)$	0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05	$\epsilon_3 \approx 0.14, \alpha_3 \approx 0.92$
$e^{-\alpha_3 y_i h_3(x_i)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
$D_3(i)\ e^{-\alpha_3 y_i h_3(x_i)}$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

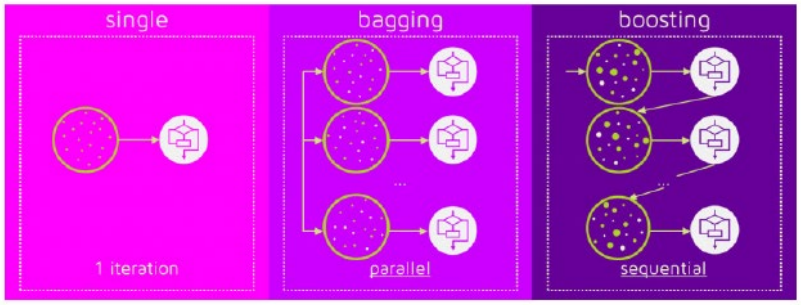
AdaBoost Example 1

- Final Classier
  - Final classier is a weighted linear combination of all the classifiers
  - Classifier h_i gets a weight α_i
- Multiple weak, linear classiers combined to give a strong, nonlinear classier



Bagging vs Boosting

- No clear winner; usually depends on the data
- Bagging is computationally more effcient than boosting (note that bagging can train the M models in parallel, boosting can't)



- Both reduce variance (and overfitting) by combining different models
  - The resulting model has higher stability as compared to the individual ones
- Bagging usually can't reduce the bias, boosting can (note that in boosting, the training error steadily decreases)
- Bagging usually performs better than boosting if we don't have a high bias and only want to reduce variance (i.e., if we are overfitting)

Outline

- Non-linear perceptron
- Classifier ensemble

- Review Decision tree

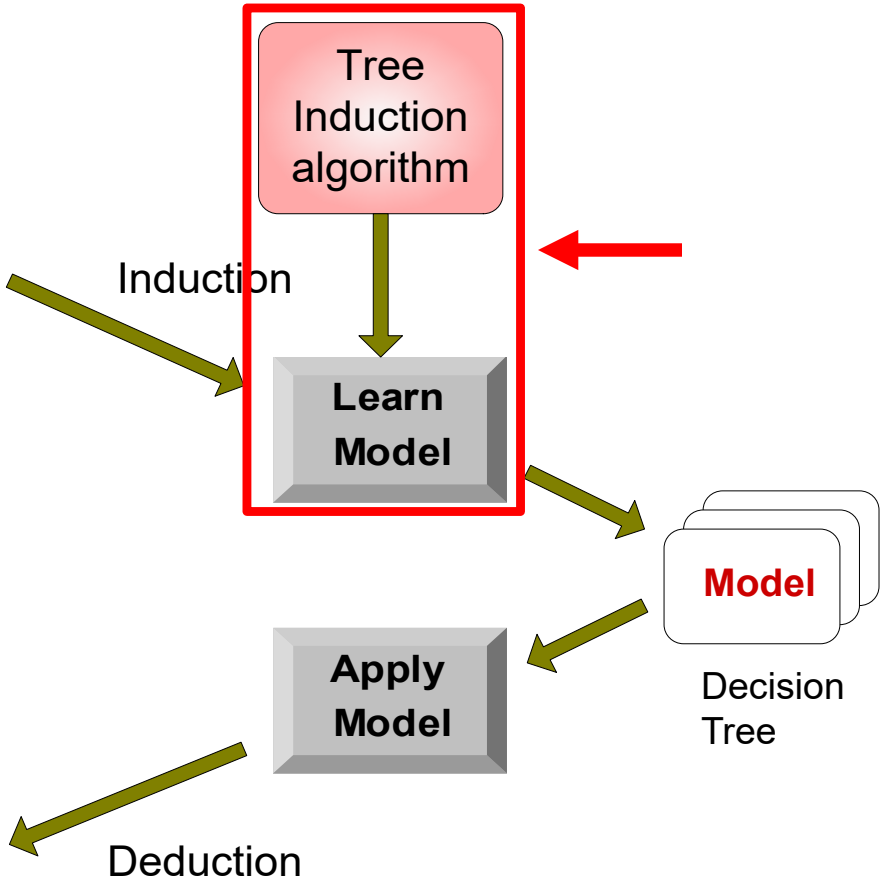
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

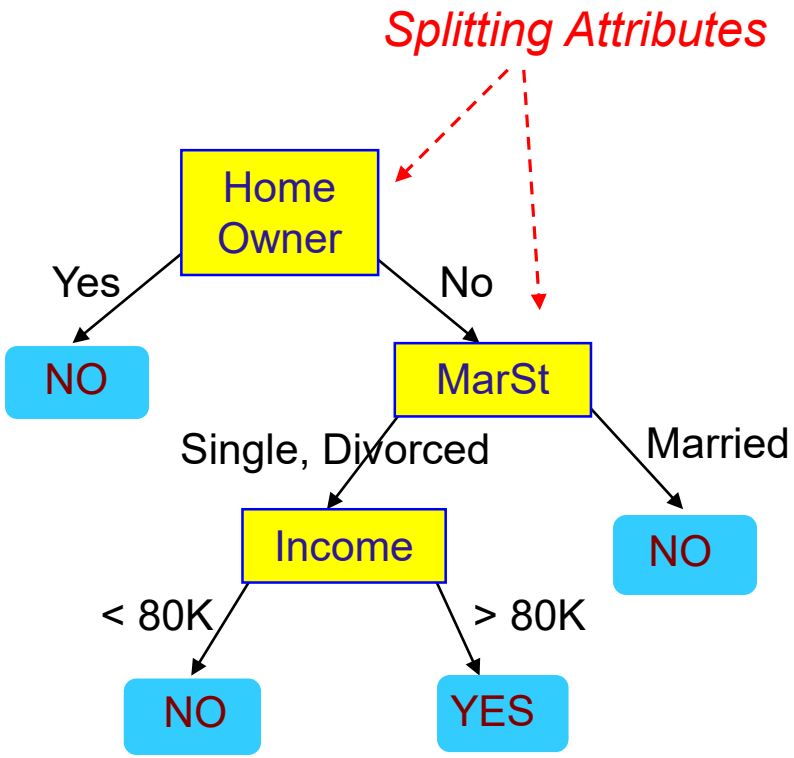
Test Set



Example of a Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

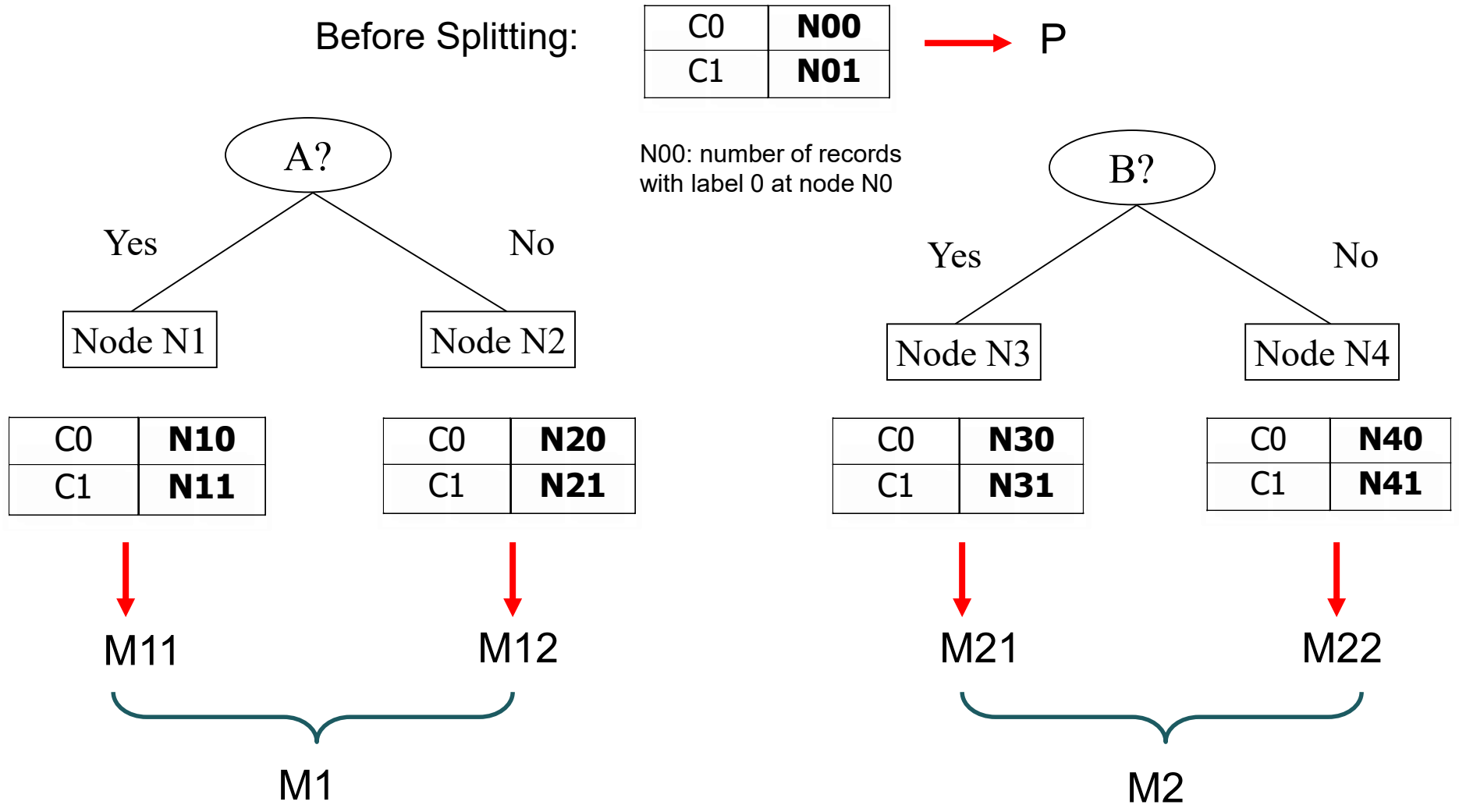
Finding the Best Split

- Compute impurity measure (P) before splitting
- Compute impurity measure (M) after splitting
  - Compute impurity measure of each child node
  - M is the weighted impurity of children
- Choose the attribute test condition that produces the highest gain

$$\text{Gain} = P - M$$

or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split



Measures of Node Impurity

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

t is a node

- Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

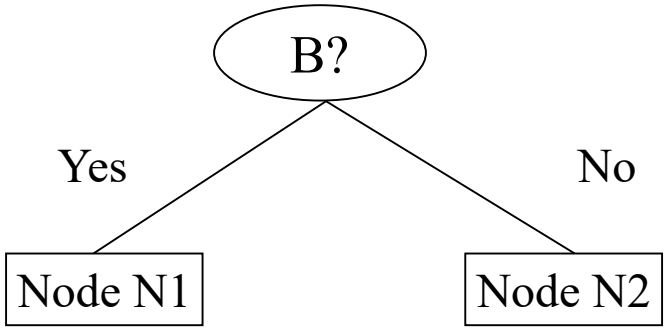
- Misclassification error

Entropy quantifies uncertainty

$$Error(t) = 1 - \max_i P(i \mid t)$$

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



	Parent
C1	**7**
C2	**5**
Gini = 0.486	

Gini(P)
$= 1 - (5/12)^2 - (7/12)^2$
$= 0.586$

	N1	N2
C1	**5**	**2**
C2	**1**	**4**
Gini=0.361		

Gini(N1)
$= 1 - (5/6)^2 - (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 - (2/6)^2 - (4/6)^2$
$= 0.444$

Weighted Gini of N1 N2
$= 6/12 * 0.278 +$
$6/12 * 0.444$
$= 0.361$

Gain = 0.486 – 0.361 = 0.125

Decision Tree example 1

- We will use the dataset below to learn a decision tree which predicts if people pass machine learning (Yes or No), based on their previous GPA (High, Medium, or Low) and whether or not they studied. ($\log_2 3 \approx 1.6$)

GPA	Studied	Passed
L	F	F
L	T	T
M	F	F
M	T	T
H	F	T
H	T	T

1. What is the entropy H(Passed)?
2. What is the entropy H(Passed | GPA)?
3. What is the entropy H(Passed | Studied)?
4. What is the root for the decision tree.

Decision Tree example 1

- We will use the dataset below to learn a decision tree which predicts if people pass machine learning (Yes or No), based on their previous GPA (High, Medium, or Low) and whether or not they studied. ($\log_2 3 \approx 1.6$)

GPA	Studied	Passed
L	F	F
L	T	T
M	F	F
M	T	T
H	F	T
H	T	T

1.What is the entropy H(Passed)?

$$H(Passed) = -(\frac{2}{6}\log_2\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6})$$
$$H(Passed) = -(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3})$$
$$H(Passed) = \boxed{\log_2 3 - \frac{2}{3} \approx 0.92}$$

2.What is the entropy H(Passed | GPA)?

$$H(Passed|GPA) = -\frac{1}{3}(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) - \frac{1}{3}(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) - \frac{1}{3}(1\log_2 1)$$
$$H(Passed|GPA) = \frac{1}{3}(1) + \frac{1}{3}(1) + \frac{1}{3}(0)$$
$$H(Passed|GPA) = \boxed{\frac{2}{3} \approx 0.66}$$

Decision Tree example 1

- We will use the dataset below to learn a decision tree which predicts if people pass machine learning (Yes or No), based on their previous GPA (High, Medium, or Low) and whether or not they studied. ($\log_2 3 \approx 1.6$)

GPA	Studied	Passed
L	F	F
L	T	T
M	F	F
M	T	T
H	F	T
H	T	T

3. What is the entropy H(Passed | Studied)??

$$H(Passed|Studied) = -\frac{1}{2}(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}) - \frac{1}{2}(1\log_2 1)$$

$$H(Passed|Studied) = \frac{1}{2}(\log_2 3 - \frac{2}{3}$$

$$H(Passed|Studied) = \boxed{\frac{1}{2}\log_2 3 - \frac{1}{3} \approx 0.46}$$

4.What is the root for the decision tree

Studied

Decision Tree example 2

- Considering the training examples shown in Table 1 for a binary classification problem.

(a) What is the entropy of this collection of training examples with respect to the positive and negative classes?

(b) For a3, which is a continuous attribute, compute the information gain for every possible binary split.

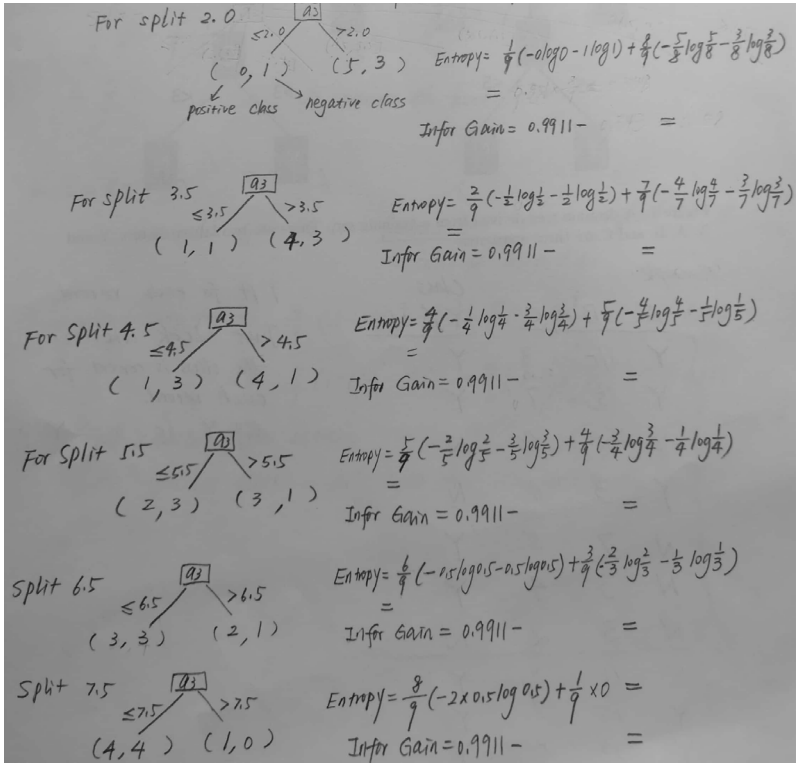
(c) What is the best split (between a1 and a2) according to the Gini index.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	4.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	6.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	3.0	−

Decision Tree example 2

(a) 5 negative classes and 4 positive classes. Entropy=-4/9 $\log_2(4/9)$ – 5/9 $\log_2(5/9)$ =0.9911.

(b)



a3	label	Split point	Entropy	Infor Gain
1.0	-	2.0	0.8484	0.1427
3.0	+	3.5	0.9885	0.0025
4.0	-	4.5	0.7616	0.2295
4.0	-			
5.0	+	5.5	0.9000	0.0911
6.0	+	6.5	0.9728	0.01833
7.0	+	7.5	0.8888	0.1022
7.0	-			
8.0	+			

Parent's Entropy = 0.9911 (from (a))

(c) Follow the definition of Gini index.
Weighted sum for a1: $4/9*(1-(1/4)^2-(3/4)^2)+5/9*(1-(4/5)^2-(1/5)^2)=0.344$
Weighted sum for a2: $5/9*(1-(3/5)^2-(2/5)^2)+4/9*(1-(1/2)^2-(1/2)^2))=0.4889$.
Choose the smaller one, so a1 is better.