COMP 472: Artificial Intelligence
Machine Learning
Naive Bayes Classification video #2

- Russell & Norvig: Sections 12.2 to 12.6

Today

1. Introduction to ML
2. Naïve Bayes Classification

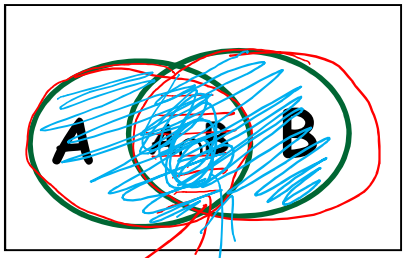
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Motivation

- How do we represent and reason when there is uncertainly in the necessary knowledge?
  - It *might* rain tonight
  - If you have red spots on your face, you *might* have the measles
  - This e-mail is *most likely* spam
  - I can't read this character, but it *looks* like a "B"
  - These 2 pictures are *very likely* of the same person
  - …
- One way, is to use probability theory

Remember…

- P is a probability function:
  - $0 \leq P(A) \leq 1$
  - $P(A) = 0 \Rightarrow$ the event A will never take place
  - $P(A) = 1 \Rightarrow$ the event A must take place
  - $\sum_i P(A_i) = 1 \Rightarrow$ one of the outcomes A_i will take place
  - $P(A) + P(\sim A) = 1$



- Joint probability
  - intersection $A_1 \cap \ldots \cap A_n$ is an event that takes place if all the events $A_1, \ldots, A_n$ take place
  - denoted $P(A \cap B)$ or $P(A, B)$

- Sum Rule
  - union $A_1 \cup \ldots \cup A_n$ is an event that takes place if at least one of the events $A_1, \ldots, A_n$ takes place
  - denoted $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

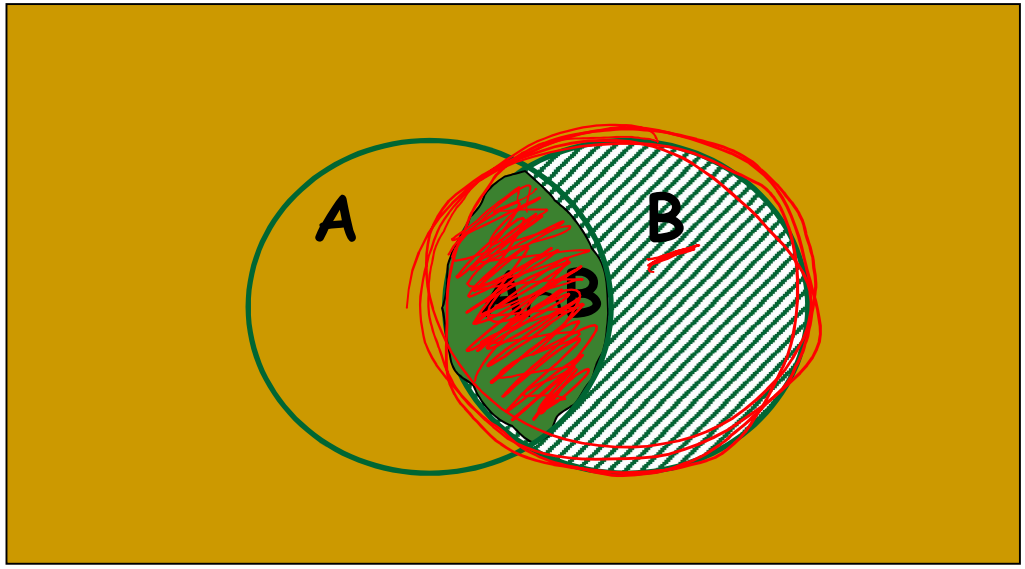
Conditional Probability

- Prior (or unconditional) probability
  - Probability of an event before any evidence is obtained
  - P(A) = 0.1 *P(rain today) = 0.1*
  - i.e. Your belief about A given that you have no evidence

- Posterior (or conditional) probability
  - Probability of an event given that you know that B is true (B = some evidence)
  - P(A | B) = 0.8 *P(rain today | cloudy) = 0.8*
  - i.e. Your belief about A given that you know B

Conditional Probability (con't)

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A,B)}{P(B)}$$



Chain Rule

- With 2 events, the probability that A and B occur is:

$$P(A, B) = P(A \mid B) \times P(B)$$

- With 3 events, the probability that A, B and C occur is:
  - The probability that A occurs
  - Times, the probability that B occurs, assuming that A occurred
  - Times, the probability that C occurs, assuming that A and B have occurred

- With n events, we can generalize to the Chain rule:

$$P(A_1, A_2, A_3, A_4, \ldots, A_n)$$
$$= P(\cap A_i)$$
$$= P(A_1) \times P(A_2 \mid A_1) \times P(A_3 \mid A_1, A_2) \times \ldots \times P(A_n \mid A_1, A_2, A_3, \ldots, A_{n-1})$$

So what?

- we can do probabilistic inference
  - i.e. infer new knowledge from observed evidence

Example 1

- Joint probability distribution:

P(Toothache ∩Cavity)

evidence

hypothesis

	Toothache	~Toothache	
Cavity	0.04	0.06	p(cavity) = 0.1
~Cavity	0.01	0.89	p(~cavity) = 0.9
	P(toothache) = 0.05	P(~toothache) = 0.9	Σ = 1

$$P(H \mid E) = \frac{P(H \cap E)}{P(E)}$$

$$P(\text{cavity} \mid \text{toothache}) = \frac{P(\text{cavity} \cap \text{toothache})}{P(\text{toothache})} = \frac{0.04}{0.04 + 0.01} = 0.8$$

Getting the Probabilities

- in most applications, you just count from a set of observations

$$P(A) = \frac{\text{count_of_A}}{\text{count_of_all_events}}$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{\text{count_of_A_and_B_together}}{\text{count_of_all_B}}$$

Combining Evidence

- Assume now 2 pieces of evidence:

- Suppose, we know that
  - P(Cavity | Toothache) = 0.12
  - P(Cavity | Young) = 0.18

- A patient complains about Toothache and is Young…
  - what is P(Cavity | Toothache $\cap$ Young) ?

Combining Evidence

	Toothache		~Toothache	
	Young	~ Young	Young	~ Young
Cavity	0.108	0.012	0.072	0.008
~Cavity	0.016	0.064	0.144	0.576

evidence #1

evidence #2

P(Toothache ∩Cavity ∩Young)

- But how do we get the data ?
- In reality, we may have dozens, hundreds of variables
- We cannot have a table with the probability of all possible combinations of variables
  - Ex. with 16 binary variables, we would need 2^{16} entries

Independent Events

- In real life:
  - some variables are independent…
    - eg: living in Montreal & tossing a coin
      - P(Montreal, head) = P(Montreal) * P(head)
    - eg: probability of tossing 2 heads in a row
      - P(head, head) = 1/2 * 1/2 = 1/4
  - some variables are not independent…
    - eg: living in Montreal & wearing boots
      - P(Montreal, boots) ≠ P(Montreal) * P(boots)

Independent Events

- Two events A and B are independent:
  - if the occurrence of one of them does not influence the occurrence of the other
  - i.e. A is independent of B if $P(A) = P(A|B)$

- If A and B are independent, then:
  - $P(A,B) = P(A|B) \times P(B)$ (by chain rule) // see previous slide 7
    $= P(A) \times P(B)$ (by independence)

- To make things work in real applications, we often assume that events are independent
  - $P(A,B) = P(A) \times P(B)$

Conditional Independent Events

- Two events A and B are conditionally independent given C:
  - Given that C is true, then any evidence about B cannot change our belief about A
  - $P(A, B \mid C) = P(A \mid C) \times P(B \mid C)$.

when C is True

Bayes' Theorem

- given: $P(A|B) = \frac{P(A,B)}{P(B)}$ so $P(A,B) = P(A|B) \times P(B)$

  $P(B|A) = \frac{P(A,B)}{P(A)}$ so $P(A,B) = P(B|A) \times P(A)$

- then: $P(A|B) \times P(B) = P(B|A) \times P(A)$

- and: $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$

So?

- We typically want to know: P(Hypothesis | Evidence)
  - P(Disease | Symptoms)… P(meningitis | red spots)
  - P(Cause | Side Effect)… P(misaligned brakes| squeaky wheels)
- But P(Hypothesis| Evidence) is hard to gather
  - *ex: out of all people who have red spots… how many have meningitis?*
- However P(Evidence | Hypothesis) is easier to gather
  - *ex: out of all people who have the meningitis … how many have red spots?*
- So

$$P(\text{Hypothesis} \mid \text{Evidence}) = \frac{P(\text{Evidence} \mid \text{Hypothesis}) \times P(\text{Hypothesis})}{P(\text{Evidence})}$$

Example 2

Assume we only have 1 hypothesis

Assume:

- P(spots=yes | meningitis=yes) = 0.4 — P(B|A)
- P(meningitis=yes) = 0.00003 — P(A)
- P(spots=yes) = 0.05 — P(B)

$$P(\underbrace{\text{meningitis} = \text{yes}}_{A} \mid \underbrace{\text{spots} = \text{yes}}_{B})$$

$$= \frac{P(\underbrace{\text{spots} = \text{yes}}_{B} \mid \underbrace{\text{meningitis} = \text{yes}}_{A}) \times P(\underbrace{\text{meningitis} = \text{yes}}_{A})}{P(\underbrace{\text{spots} = \text{yes}}_{B})}$$

$$= \frac{0.4 \times 0.00003}{0.05} = \boxed{0.00024}$$

P(A)

→ If you have spots… you are more likely to have meningitis than if we don't know about you having spots

Example 3

- Predict the weather tomorow based on tonight's sunset...

evidence
feature

- Assume we have 3 hypothesis...
  - H_1: *weather will be nice* $P(H_1) = 0.2$
  - H_2: *weather will be bad* $P(H_2) = 0.5$
  - H_3: *weather will be mixed* $P(H_3) = 0.3$
- And 1 piece of evidence with 3 possible values
  - E_1: today, there's a *beautiful sunset*
  - E_2: today, there's a *average sunset*
  - E_3: today, there's *no sunset*

$\boxed{P(E_2 | H_1)}$

$P(E_x \mid H_i)$	E_1	E_2	E_3
H_1	0.7	0.2	0.1
H_2	0.3	0.3	0.4
H_3	0.4	0.4	0.2

Example 3

- Observation: average sunset (E_2)
- Question: how will be the weather tomorrow?
  - $P(H_i \mid E_2)$?
  - predict the weather that maximizes the probability
  - select H_i such that $P(H_i \mid E_2)$ is the greatest

$$P(H_i \mid E_2) = \frac{P(H_i) \times P(E_2 \mid H_i)}{P(E_2)}$$

$$P(E_2) = P(H_1) \times P(E_2 \mid H_1) + P(H_2) \times P(E_2 \mid H_2) + P(H_3) \times P(E_2 \mid H_3)$$

$$= .2 \times .2 + .5 \times .3 + .3 \times .4 = .04 + .15 + .12 = 0.31$$

Example 3

$$P(H_1 \mid E_2) = \frac{P(H_1) \times P(E_2 \mid H_1)}{P(E_2)} = \frac{.2 \times .2}{.31} = .129$$

$$P(H_2 \mid E_2) = \frac{P(H_2) \times P(E_2 \mid H_2)}{P(E_2)} = \frac{.5 \times .3}{.31} = .484$$

$$P(H_3 \mid E_2) = \frac{P(H_3) \times P(E_2 \mid H_3)}{P(E_2)} = \frac{.3 \times .4}{.31} = .387$$

H_2 will still have the highest score

the argument

$\Rightarrow$ H_2 is the most likely hypothesis, given the evidence

$P(H_2 \mid E_2)$ is the highest

Tomorrow the weather will be bad

H_2

$$H_{NB} = \underset{H_i}{\operatorname{argmax}} \frac{P(H_i) \times P(E \mid H_i)}{P(E)}$$

H_2

select H_i that maximize the function

Bayes' Reasoning

- Out of n hypothesis…
  - we want to find the most probable H_i given the evidence E
- So we choose the H_i with the largest $P(H_i|E)$

$$H_{NB} = \underset{H_i}{\operatorname{argmax}}\, P(H_i \mid E) = \underset{H_i}{\operatorname{argmax}} \frac{P(H_i) \times P(E \mid H_i)}{P(E)}$$

- But… P(E)
  - is the same for all possible H_i (and is hard to gather anyways)
  - so we can drop it
- So Bayesian reasoning:

prior

$$H_{NB} = \underset{H_i}{\operatorname{argmax}} \frac{P(H_i) \times P(E \mid H_i)}{P(E)} = \underset{H_i}{\operatorname{argmax}}\, P(H_i) \times P(E \mid H_i)$$

Representing the Evidence

- The evidence is typically represented by many attributes/features

  100's 1000's

  - beautiful sunset? clouds? temperature? summer?, …
- so often represented as a feature/attribute vector

  =

	evidence					hypothesis
	sunset a_1	clouds a_2	temp a_3.	summer a_4		weather tomorrow
e1	beautiful	no	high	yes		*Nice*

- e1 = <sunset:beautiful, clouds:no, temp:high, summer:yes>

  features values

Combining Evidence

toothache	young	cavity
yes	yes	?

$$P(\text{Cavity} = \text{yes} \mid \text{Toothache} = \text{yes} \cap \text{Young} = \text{yes}) = ?$$

with Bayes Rule :

$$= \frac{P(\text{Toothache} = \text{yes} \cap \text{Young} = \text{yes} \mid \text{Cavity} = \text{yes}) \times P(\text{Cavity} = \text{yes})}{P(\text{Toothache} = \text{yes} \cap \text{Young} = \text{yes})}$$

with independence assumption :

$$= \frac{P(\text{Toothache} = \text{yes} \cap \text{Young} = \text{yes} \mid \text{Cavity} = \text{yes}) \times P(\text{Cavity} = \text{yes})}{P(\text{Toothache} = \text{yes}) \times P(\text{Young} = \text{yes})}$$

with conditional independence assumption :

$$= \frac{P(\text{Toothache} = \text{yes} \mid \text{Cavity} = \text{yes}) \times P(\text{Young} = \text{yes} \mid \text{Cavity} = \text{yes}) \times P(\text{Cavity} = \text{yes})}{P(\text{Toothache} = \text{yes}) \times P(\text{Young} = \text{yes})}$$

Now we have decomposed the joint probability distribution into much smaller pieces…

Combining Evidence

toothache	young	cavity
yes	yes	yes? or no?

But since we only care about ranking the hypothesis…

$$P(\text{Cavity} = \text{yes} \mid \text{Toothache} = \text{yes} \cap \text{Young} = \text{yes})$$

$$>$$

$$P(\text{Cavity} = \text{no} \mid \text{Toothache} = \text{yes} \cap \text{Young} = \text{yes})$$

$$\frac{P(\text{Cavity} = \text{yes}) \times P(\text{Toothache} = \text{yes} \mid \text{Cavity} = \text{yes}) \times P(\text{Young} = \text{yes} \mid \text{Cavity} = \text{yes})}{P(\text{Toothache} = \text{yes}) \times P(\text{Young} = \text{yes})}$$

$$>$$

$$\frac{P(\text{Cavity} = \text{no}) \times P(\text{Toothache} = \text{yes} \mid \text{Cavity} = \text{no}) \times P(\text{Young} = \text{yes} \mid \text{Cavity} = \text{no})}{P(\text{Toothache} = \text{yes}) \times P(\text{Young} = \text{yes})}$$

$$P(\text{Cavity} = \text{yes}) \times P(\text{Toothache} = \text{yes} \mid \text{Cavity} = \text{yes}) \times P(\text{Young} = \text{yes} \mid \text{Cavity} = \text{yes})$$

$$>$$

$$P(\text{Cavity} = \text{no}) \times P(\text{Toothache} = \text{yes} \mid \text{Cavity} = \text{no}) \times P(\text{Young} = \text{yes} \mid \text{Cavity} = \text{no})$$

$$H_{NB} = \underset{H_i}{\operatorname{argmax}} \frac{P(H_i) \times P(E \mid H_i)}{P(E)} = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times P(E \mid H_i) = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times P(< a_1, a_2, a_3, \ldots, a_n > \mid H_i) = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times \prod_{j=1}^{n} P(a_j \mid H_i)$$

Example 4

many pieces of **evidence**

features / attributes

2 hypothesis
2 classes

14 days

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

H_1 = no tennis

H_2 = play tennis

$p(H_2) = 9/14$

$p(H_1) = \frac{5}{14}$

Example 4

- Goal: Given a new instance $X=\langle a_1,\ldots, a_n\rangle$, classify as Yes/No

H_2 H_1

$$H_{NB} = \underset{H_i}{\operatorname{argmax}} \frac{P(H_i) \times P(E|H_i)}{P(E)} = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times P(E|H_i) = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times P(\langle a_1, a_2, a_3, \ldots, a_n \rangle | H_i) = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times \prod_{j=1}^{n} P(a_j | H_i)$$

- Naïve Bayes: Assumes that the attributes/features are conditionally independent given the hypothesis

Example 4

- Goal: Given a new instance $X=\langle a_1,\ldots, a_n\rangle$, classify as Yes/No

$$H_{NB} = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times \prod_{j=1}^{n} P(a_j \mid H_i)$$

1. 1st estimate the probabilities from the training examples:

   a) For each hypothesis H_i estimate $P(H_i)$

   b) For each attribute value a_j of each instance (evidence) estimate $P(a_j \mid H_i)$

Example 4

1. TRAIN:
   - compute the probabilities from the training set

$$P(\text{PlayTennis} = \text{yes}) = 9/14 = 0.64$$
$$P(\text{PlayTennis} = \text{no}) = 5/14 = 0.36$$

prior probabilities $P(H_i)$

$$P(\text{Out} = \text{sunny} \mid \text{PlayTennis} = \text{yes}) = 2/9 = 0.22$$
$$P(\text{Out} = \text{sunny} \mid \text{PlayTennis} = \text{no}) = 3/5 = 0.60$$
$$P(\text{Out} = \text{rain} \mid \text{PlayTennis} = \text{yes}) = 3/9 = 0.33$$
$$P(\text{Out} = \text{rain} \mid \text{PlayTennis} = \text{no}) = 2/5 = 0.4$$
$$\ldots$$
$$P(\text{Wind} = \text{strong} \mid \text{PlayTennis} = \text{yes}) = 3/9 = 0.33$$
$$P(\text{Wind} = \text{strong} \mid \text{PlayTennis} = \text{no}) = 3/5 = 0.60$$

conditional probabilities

$P(a_j \mid H_i)$

Example 4

2. TEST:

classify the new case: X=(Outlook: Sunny, Temp: Cool, Hum: High, Wind: Strong)

prior

conditional.

$$H_{NB} = \underset{H_i \in [yes,no]}{\operatorname{argmax}} P(H_i) \times P(X \mid H_i)$$

$$= \underset{H_i \in [yes,no]}{\operatorname{argmax}} P(H_i) \times \prod_j P(a_j \mid H_i)$$

$$= \underset{H_i \in [yes,no]}{\operatorname{argmax}} P(H_i) \times P(Outlook = sunny \mid H_i) \times P(Temp = cool \mid H_i)$$

$$\times P(Humidity = high \mid H_i) \times P(Wind = strong \mid H_i)$$

should have been computed in the previous slide

score H_2

1) P(PlayTennis = yes) = 9/14

x P(Outlook = sunny | PlayTennis = yes) x P(Temp = cool | PlayTennis = yes) x P(Hum = high | PlayTennis = yes) x P(Wind = strong | PlayTennis = yes) = 3/9

= 0.0053

score H_1

2) P(PlayTennis = no) = 5/14

x P(Outlook = sunny | PlayTennis = no) x P(Temp = cool | PlayTennis = no) x P(Hum = high | PlayTennis = no) x P(Wind = strong | PlayTennis = no) = 3/5

= 0.0206

2/9

3/5

$\Rightarrow$ answer : PlayTennis(X) = no

Application of Bayesian Reasoning

- Categorization: P(Category | Features of Object)
  - Diagnostic systems: P(Disease | Symptoms)
  - Text classification: P(sports_news | text)
  - Character recognition: P(character | bitmap)
  - Speech recognition: P(words | acoustic signal)
  - Image processing: P(face_person | image features)
  - Spam filter: P(spam_message | words in e-mail)
  - …

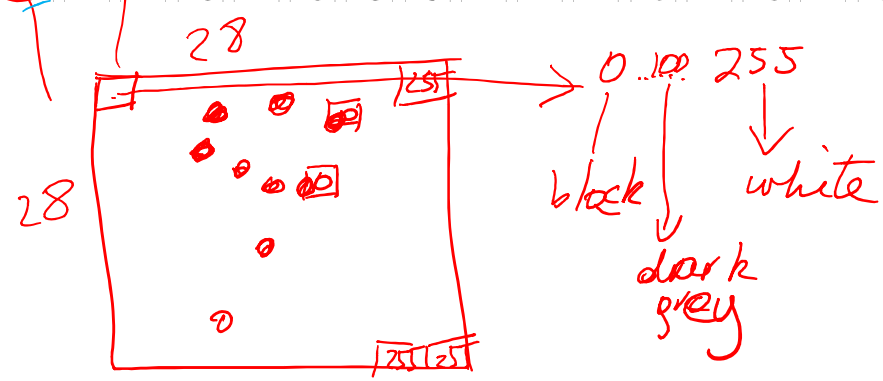
Digit Recognition

$H_0 = "0"$
$H_1 = "1"$
10 hyp.
H_9

$\langle 1,1:255,\ 1,2:100,\ \ldots\ldots\ 28,28:255 \rangle$

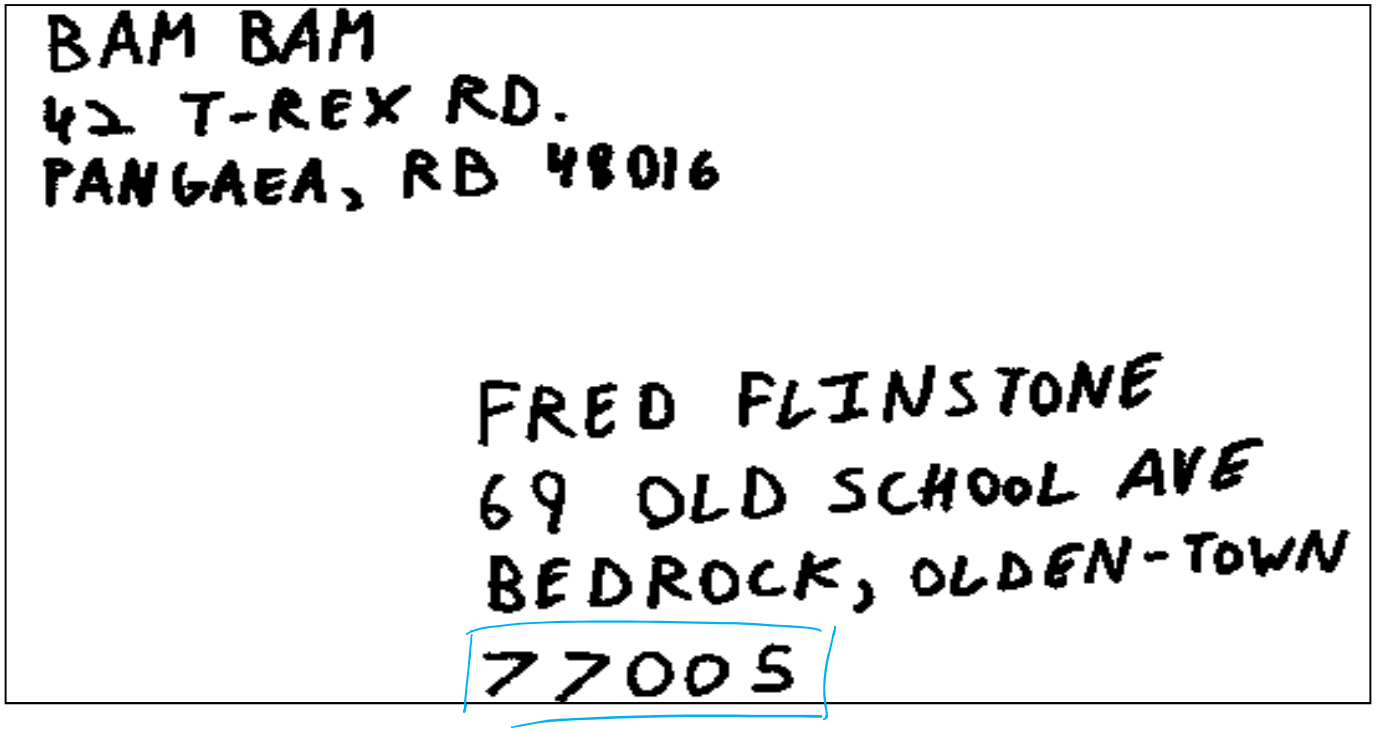
784 features

- MNIST dataset
- data set contains handwritten digits from the American Census Bureau employees and American high school students
- 28 x 28 grayscale images
- training set: 60,000 examples
- test set: 10,000 examples.
- Features: each pixel is used as a feature so:
  - there are 28x28 = 784 features
  - each feature = 256 greyscale value
- Task: classify new digits into one of the 10 classes



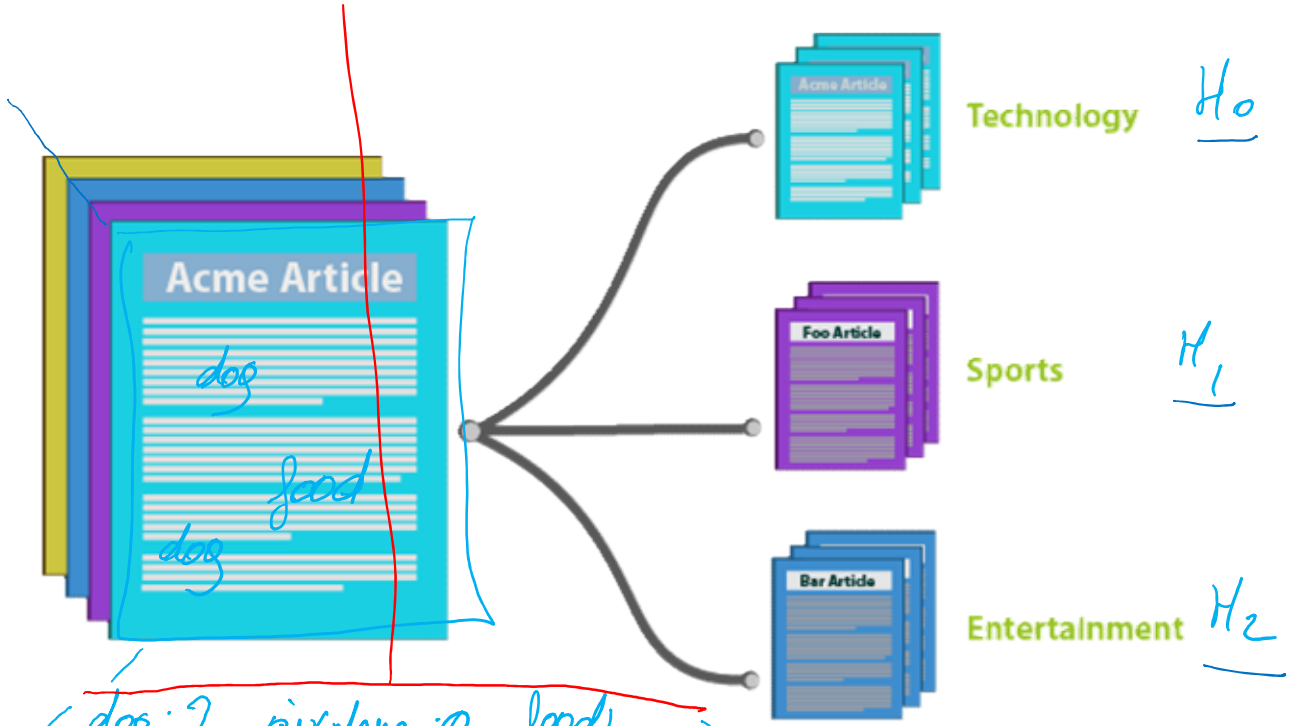
https://en.wikipedia.org/wiki/MNIST_database

Postal Code Recognition



Text Classification

10 000 features (i.e. 1 feature for each word in the dictionary)



text < dog : 2, airplane : 0, food 1, ... >

frequency of the word in the document

features : actual words in English

https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

Comments on Naïve Bayes Classification

- A simple probabilistic classifier based on Bayes' theorem
  - with strong (naive) independence assumption
  - i.e. the features/attributes are conditionally independent given the classes
    - eg: assumes that the word *ambulance* is conditionally independent of the word *accident* given the class SPORTS

$$P(\langle E \rangle \mid H_i) = \prod P(e_i \mid H_i)$$

- BUT:
  - fast, simple
  - gives confidence in its class predictions (i.e., the scores)
  - surprisingly very effective on real-world tasks
  - basis of many spam filters

score(H_1) = 0.5
score(H_2) = 0.00
score(H_3) = 0.00001

Today

1. Introduction to ML ✓
2. Naïve Bayes Classification ✓ video #2
   a. Application to Spam Filtering video #3
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

COMP 472: Artificial Intelligence
Machine Learning part 2
Naive Bayes Classification
Application to Spam Filtering video #3

- Russell & Norvig: Sections 12.2 to 12.6

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering

3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Recall

$$H_{NB} = \underset{H_i}{\operatorname{argmax}} \frac{P(H_i) \times P(E|H_i)}{\cancel{P(E)}} = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times P(E|H_i) = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times P(<a_1, a_2, a_3, \ldots, a_n>|H_i) = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times \prod_{j=1}^{n} P(a_j|H_i)$$

// prior // conditionals

$$\boxed{H_{NB} = \underset{H_i}{\operatorname{argmax}}\ P(H_i) \times \prod_{j=1}^{n} P(a_j|H_i)}$$

// attributes
// features
// piece of evidence

hypothesis
class
category
bucket

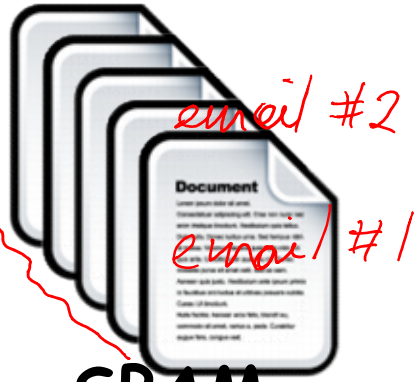
Application of Naive Bayes Classification: Spam Filtering

- Task: classify e-mails (documents) into a pre-defined class
  - ex: spam / ham
  - ex: sports, recreation, politics, war, economy,…

ex : product review

positive neutral negetiono

- Given
  - training set of documents already classified into the correct category



SPAM

HAM

e-mail Representation

- each e-mail is represented by a vector of feature/value:
  - feature = actual words in the e-mail
  - value = number of times that word appears in the e-mail

X

10000 words ... 100 000 words → f(x)

email #1 <airplane=0, banana=1, cat=5, duck=4, ..., zoo=0, class=SPAM>

email #2 <airplane=2, banana=0, cat=0, duck=8, ..., zoo=3, class=SPAM>

40 spam

assume 100 emails in training set

...

email <airplane=1, banana=1, cat=5, duck=8, ..., zoo=3, class=HAM>

<airplane=1, banana=3, cat=5, duck=0, ..., zoo=6, class=HAM>

60 ham

Strictly speaking, what this is called a Multinomial Naïve Bayes classifier, because we use the frequency of words, as opposed to just using binary values for the presence/absence of words.

Naïve Bayes Algorithm

// 1. training

for all classes c_i // ex. ham or spam

for all words w_j in the vocabulary

compute $P(w_j \mid c_i) = \dfrac{\text{count}(w_j, c_i)}{\sum_j \text{count}(w_j, c_i)}$

// conditionals.

for all classes c_i

compute $P(c_i) = \dfrac{\text{count(documents in } c_i)}{\text{count(all documents)}}$

// $P(\text{Hi})$ $P(\text{ham}) = 0.6$ $P(\text{spam}) = 0.4$

// 2. testing a new document D

for all classes c_i // ex. ham or spam

score(c_i) = $P(c_i)$

for all words w_j in the D

score(c_i) = score(c_i) x $P(w_j \mid c_i)$

choose c^* = with the greatest score(c_i)

HNB

	w_1	w_2	w_3	w_4	w_5	w_6
c1 : SPAM	$p(w_1\|c_1)$	$p(w_2\|c_1)$	$p(w_3\|c_1)$	$p(w_4\|c_1)$	$p(w_5\|c_1)$	$p(w_6\|c_1)$
c2 : HAM	$p(w_1\|c_2)$	$p(w_2\|c_2)$	$p(w_3\|c_2)$	$p(w_4\|c_2)$	$p(w_5\|c_2)$	$p(w_6\|c_2)$

Example 1

vocabulary = {best, book, cheap, sale, trip, meds}

- Dataset
  - c1: SPAM
    - doc1: "cheap meds for sale"
    - doc2: "click here for the best meds"
    - doc3: "book your trip"
  - c2: HAM
    - doc4: "cheap book sale, not meds"
    - doc5: "here is the book for you"

Training set

p(spam) = 3/5

p(best | spam) = 1/7

p(book | spam) = 1/7

p(meds | spam) = 2/7

/7

Σ 7/7

SPAM

p(ham) = 2/5

HAM

p(best | ham) = 0/5

Σ = 5/5

- Question:
  - doc6: "the cheap book"
  - should it be classified as HAM or SPAM?

Example 1

a word in an email is

Assume

vocabulary = {best, book, cheap, sale, trip, meds} |V| = 6 words only

If not in the vocabulary, ignore word

not realistic.

1. Training:

conditionals |V|

- P(best|SPAM) = 1/7 P(best|HAM) = 0/5
- P(book|SPAM) = 1/7 P(book|HAM) = 2/5
- P(cheap|SPAM) = 1/7 P(cheap|HAM) = 1/5
- P(sale|SPAM) = 1/7 P(sale|HAM) = 1/5
- P(trip|SPAM) = 1/7 P(trip|HAM) = 0/5
- P(meds|SPAM) = 2/7 P(meds|HAM) = 1/5

priors

- P(SPAM) = 3/5 P(HAM) = 2/5

2. Testing: "~~the~~ cheap book" trip

argmax

- Score(HAM)= P(HAM) x P(cheap|HAM) x P(book|HAM) x 0
- Score(SPAM)= P(SPAM) x P(cheap|SPAM) x P(book|SPAM) x 1/7

Be Careful: Smooth Probabilities

- normally: $P(w_i \mid c_j) = \dfrac{(\text{frequency of } w_i \text{ in } c_j)}{\text{total number of words in } c_j}$
- what if we have a $P(w_i|c_j) = 0$...?
  - ex. the word "dumbo" never appeared in the class SPAM?
  - then P("dumbo"| SPAM) = 0
- so if a text contains the word "dumbo", the class SPAM is completely ruled out !
- to solve this: we assume that every word always appears at least once (or a smaller value)
  - ex: add-1 smoothing:

original values

for smoothing purposes

$$P(w_i \mid c_j) = \frac{(\text{frequency of } w_i \text{ in } c_j) + 1}{\text{total number of words in } c_j + \text{size of vocabulary}}$$

Smoothing – add-1 smoothing

- Assume:
  - vocabulary V = {ball, heat, kitchen, referee, stove, the, ... }
  - |V| = 100
- Training set:

original data set

c1: COOKING	c2: SPORTS
doc_1: ... stove... kitchen... the... heat doc_2: ... kitchen... pasta... stove... ... doc_{100000}: ... stove...heat... ball...	doc_1: ... ball... heat... doc_2: ... the... referee... player... ... doc_{75000}: goal... injury ...
|V| ball heat kitchen + + + 100 extra words	100 extra words

new smoothed values for the conditional probs will be based on [ ] + [ ]

Be Careful: Use Logs

- if we really do the product of probabilities…
  - $\text{argmax}_{c_j} P(c_j) \prod P(w_i|c_j)$
  - we soon have numerical underflow…
  - ex: 0.01 x 0.02 x 0.05 x …
- so instead, we add the log of the probs
  - $\text{argmax}_{c_j} \log(P(c_j)) + \sum \log(P(w_i|c))$
  - ex: log(0.01) + log(0.02) + log(0.05) + … $= -3$ ✓ -4

so we keep the same ranking

– use the base (log) that you prefer

Example 2

- Training set:

c1: COOKING	c2: SPORTS
doc_1: … stove… kitchen… the… heat	doc_1: … ball… heat…
doc_2: … kitchen… pasta… stove…	doc_2: … the… referee… player…
…	…
doc_{100000}: … stove…heat… ball…	doc_{75000}: goal… injury …
Σ words = 500 000	Σ = 300 000 words

- Assume:
  - vocabulary V = {ball, heat, kitchen, referee, stove, the, … }
  - |V| = 100 // not realistic
  - 500,000 words in Cooking
  - 300,000 words in Sports
  - 100,000 docs in Cooking
  - 75,000 docs in Sports

Example 2

assume these numerators

- Training – Unsmoothed / Smoothed probs:

conditionals

|V| = 100 rows = 100 words

- P(ball|COOKING) = $\frac{10{,}000}{500{,}000}$ $\frac{??}{??}$
- P(heat|COOKING) = $\frac{255}{500{,}000}$ $\frac{??}{??}$
- P(kitchen|COOKING) = $\frac{2{,}600}{500{,}000}$ $\frac{??}{??}$
- P(referee|COOKING) = $\frac{0}{500{,}000}$ $\frac{??}{??}$
- P(stove|COOKING) = $\frac{3{,}600}{500{,}000}$ $\frac{??}{??}$
- P(the|COOKING) = $\frac{400{,}000}{500{,}000}$ $\frac{??}{??}$
- …

P(ball|SPORTS) = $\frac{10{,}000}{300{,}000}$ $\frac{??}{??}$

P(heat|SPORTS) = $\frac{1{,}8000}{300{,}000}$ $\frac{??}{??}$

P(kitchen|SPORTS) = $\frac{0}{300{,}000}$ $\frac{??}{??}$ $\frac{1}{300100}$

P(referee|SPORTS) = $\frac{1{,}500}{300{,}000}$ $\frac{??}{??}$

P(stove|SPORTS) = $\frac{4}{300{,}000}$ $\frac{??}{??}$ $\frac{5}{300100}$

P(the|SPORTS) = $\frac{19{,}000}{300{,}000}$ $\frac{??}{??}$

$\frac{10\,000+1}{500\,000+100}$

$\frac{0+1}{500\,000+100}$

priors

- P(COOKING) = $\frac{10{,}000}{175{,}000}$

P(SPORTS) = $\frac{75{,}000}{175{,}000}$

- Testing: "the referee hit the ~~blue bird~~"

argmax

- Score(COOKING)= $\log(\frac{100{,}000}{175{,}000})$ + log(P(the|COOKING)) + log(P(referee|COOKING)) + log(P(hit|COOKING)) + log(P(the|COOKING))
- Score(SPORTS)= $\log(\frac{75{,}000}{175{,}000})$ + log(P(the|SPORTS)) + log(P(referee|SPORTS)) + log(P(hit|SPORTS)) + log(P(the| SPORTS))

assume not in V

Today

1. Introduction to ML ✓
2. Naïve Bayes Classification ✓
   a. Application to Spam Filtering ✓ video #3
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees video #4
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

COMP 472: Artificial Intelligence
Machine Learning part #2
Decision Trees video #4

- Russell & Norvig: Sections 19.3

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees

4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Guess Who?

p1

p2

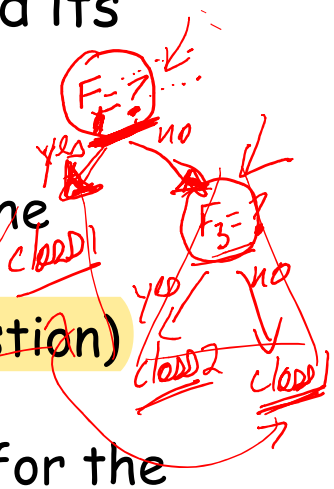
DT

Decision Trees

Ross Quinlan

- Simplest, but most successful form of learning algorithm
- Very well-know algorithm is ID3 (Quinlan, 1987) and its successor C4.5

3-48 . . .



1. Rank features based on how good they are to indicate the result
2. Put the most discriminating feature as a node (as a question) of a tree
3. Split the examples (training) so that those with different values for the chosen feature are in a different set
4. Repeat the same process with the nest most discriminating feature

http://www.rulequest.com/Personal/

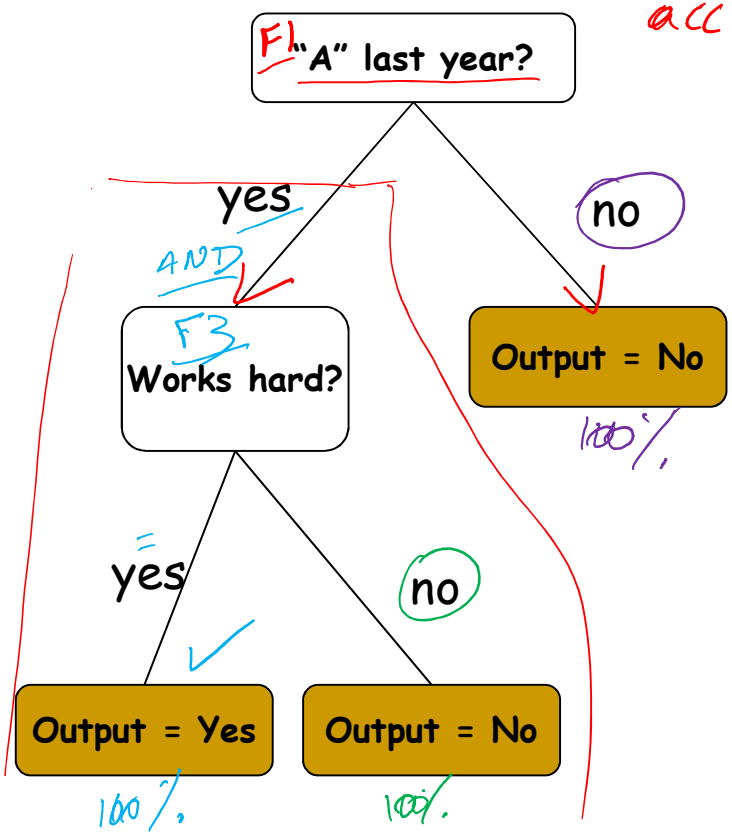
Example 1

Info on last year's students to determine if a student will get an 'A' this year

	Features (X)				Output f(X) class
Student	**'A' last year?** F1	**Black hair?** F2	**Works hard?** F3	**Drinks?** F4	**'A' this year?**
X1: Richard	Yes	Yes	No	Yes	**No**
X2: Alan	Yes	Yes	Yes	No	**Yes**
X3: Alison	No	No	Yes	No	**No**
X4: Jeff	No	Yes	No	Yes	**No**
X5: Gail	Yes	No	Yes	Yes	**Yes**
X6: Simon	No	Yes	Yes	Yes	**No**

Example 1

A random decision tree that fits the dataset with 100% accuracy



same data set as previous slide

	Features				Output f(X)
Student	'A' last year? F1	Black hair?	Works hard? F3	Drinks ?	**'A' this year?**
Richard	Yes	Yes	No	Yes	No
Alan	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	No
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No

- fits the training set with 100% accuracy
- but the features are chosen at random so there might be a shorter DT // better

Example 2: The Restaurant

- Goal: learn whether one should wait for a table
- Attributes

10 features

1. Alternate: another suitable restaurant nearby
2. Bar: comfortable bar for waiting
3. Fri/Sat: true on Fridays and Saturdays
4. Hungry: whether one is hungry
5. Patrons: how many people are present (none, some, full)
6. Price: price range ($, $$, $$$)
7. Raining: raining outside
8. Reservation: reservation made
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated wait by host (0-10 mins, 10-30, 30-60, >60)

Example 2: The Restaurant

- Training data:

10 features.

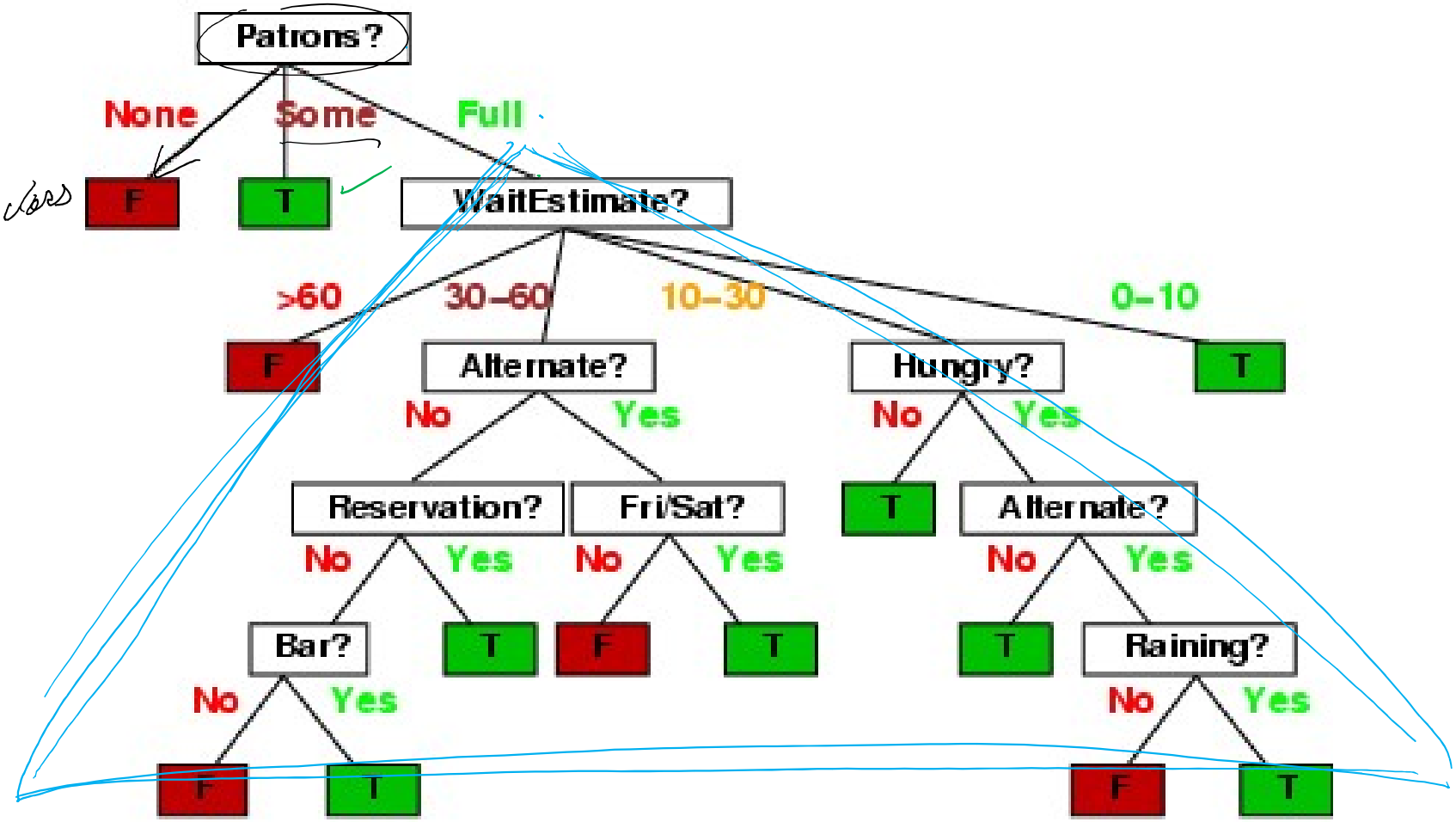
f(x) class

12 instances

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

source: Norvig (2003)

A First Decision Tree



- But is it the best decision tree we can build?

source: Norvig (2003)

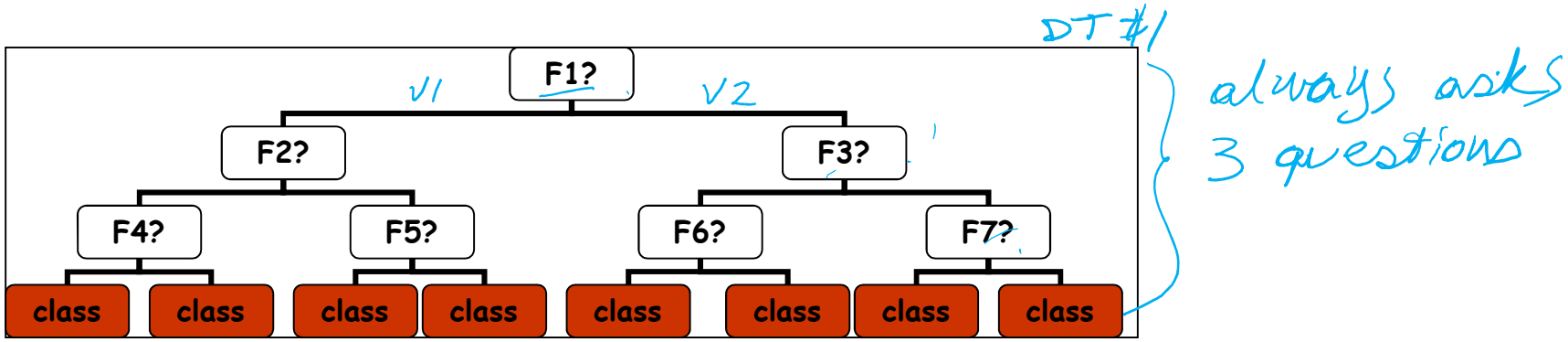
Ockham's Razor

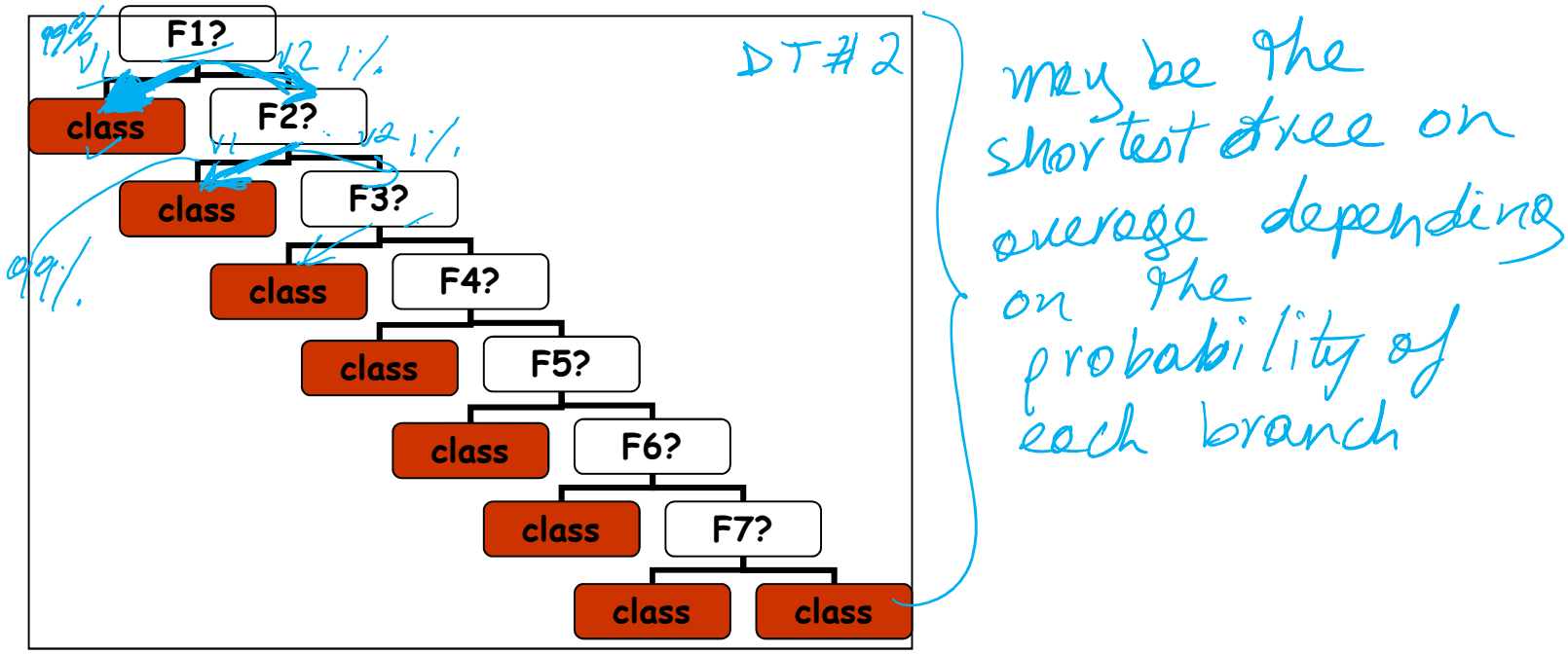
It is vain to do more than can be done with less… Entities should not be multiplied beyond necessity. [Ockham, 1324]

- In other words… always favor the simplest answer that correctly fits the training data
- i.e. the smallest tree on average
- This type of assumption is called inductive bias
  - inductive bias = making a choice beyond what the training instances contain

https://www.britannica.com/topic/Occams-razor

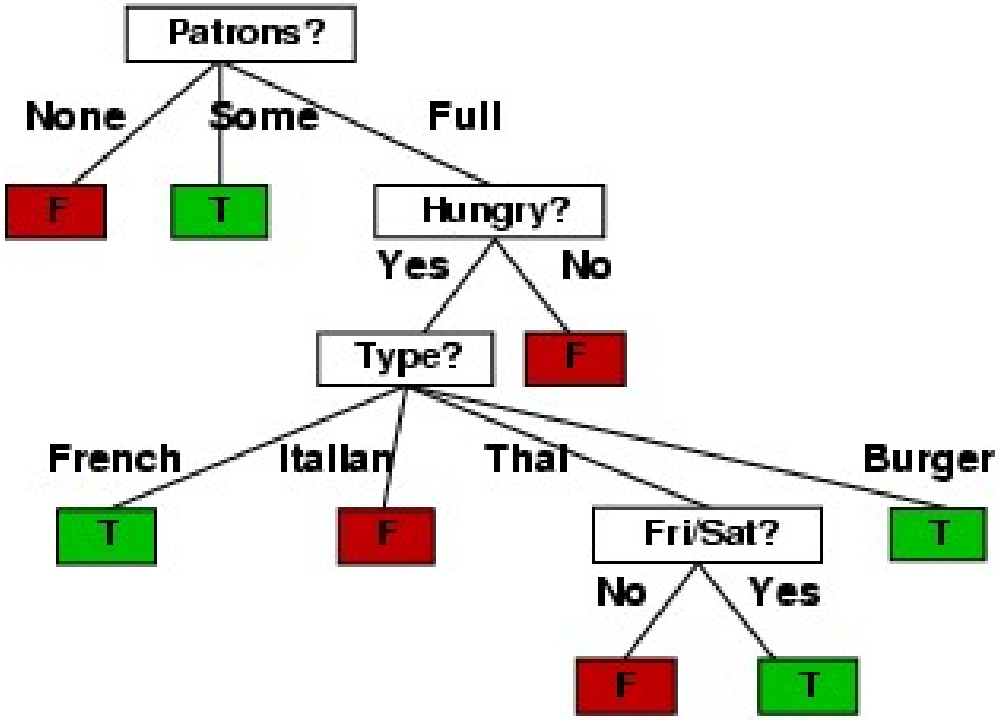
Which Tree is Best?





A Better Decision Tree

- 4 tests instead of 9
- 11 branches instead of 21



source: Norvig (2003)

Choosing the Next Feature

- The key problem is choosing which feature to split a given set of examples
- Different measures have been proposed
- ID3 uses Maximum Information-Gain
  - i.e. we choose the feature that has the largest information gain
  - we expect this feature to result in the smallest tree on average
  - based on information theory

Essential Information Theory

- Developed by Shannon in the 40s
- Shannon developed the notion of entropy (aka information content) of a random variable (RV)

- Entropy measures how "predictable" a RV is
  - If you already have a good idea about the answer (e.g. 90/10 split)
    → low entropy
  - If you have no idea about the answer (e.g. 50/50 split)
    → high entropy

Entropy

- Let X be a discrete RV with *i* possible outcomes x_i
- Entropy (or information content) of X

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

- measures:
  - the *amount of information* in a RV
  - *average uncertainty* of a RV
  - *average length of a message* needed to transmit an outcome x_i of that RV when encoded optimally over a binary channel
- measured in bits

x_i: 'e' ...

x_n n z ...

x

hello

base 2 always

Entropy of a Coin Toss

$$H(X) = -\sum_{x_i \in X} p(x_i)\log_2 p(x_i)$$

Entropy (or information content)

prob x_1 = "head"

prob x_2 = "tail"

$$H(\text{fair coin toss}) = -\sum_{x_i \in X} p(x_i)\log_2 p(x_i) = H\left(\frac{1}{2}, \frac{1}{2}\right)$$

$$= -\left(\underbrace{\frac{1}{2}\log_2\frac{1}{2}}_{\text{head}} + \underbrace{\frac{1}{2}\log_2\frac{1}{2}}_{\text{tail}}\right) = 1\,\text{bit}$$

entropy of a fair coin toss (the RV) with 2 possible outcomes, each with a probability of 1/2

a RV with only 2 outcomes x_1 and x_2 will have $1 \geq H(X) \geq 0$

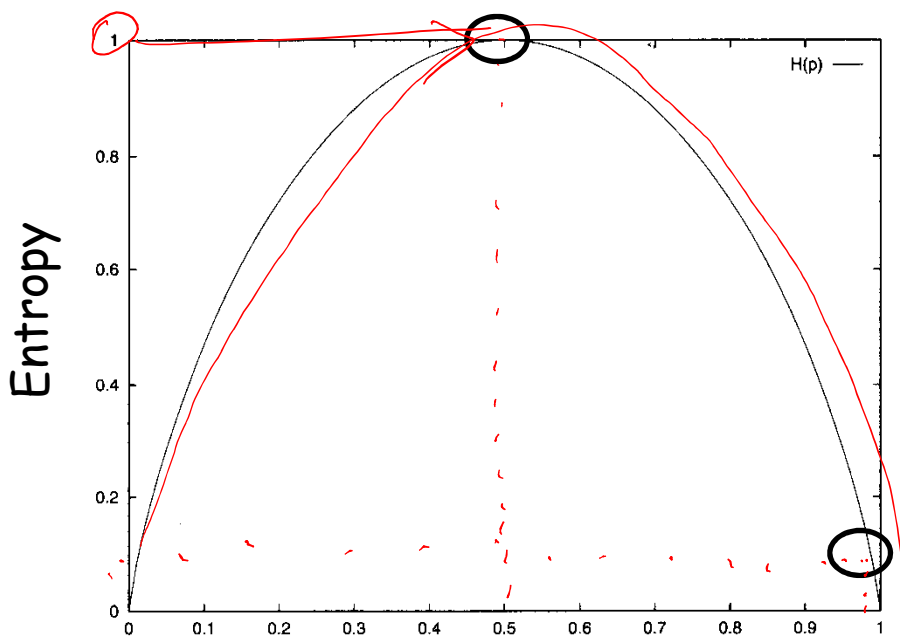
// sure thing

// total chaos

(unpredictable)

Example: The Coin Toss

- Fair coin: $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = -\left(\overbrace{\frac{1}{2}\log_2\frac{1}{2}}^{\text{head}} + \overbrace{\frac{1}{2}\log_2\frac{1}{2}}^{\text{tail}}\right) = 1\,\text{bit}$
- Rigged coin: $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = -\left(\underbrace{\frac{99}{100}\log_2\frac{99}{100}}_{\text{head}} + \underbrace{\frac{1}{100}\log_2\frac{1}{100}}\right) = 0.08\,\text{bits}$



fair coin -> high entropy

rigged coin -> low entropy

P(head)

So what?

for each attribute/feature I will compute how much knowing its values will reduce the entropy of the class

RV

- Training data:

10 features

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

x_1

x_2

entropy(Target Wait) = 1

source: Norvig (2003)

Information Gain

- information gain
  - measure the entropy reduction of a RV, once a piece of information is known
  - used to measure the "discriminating power" of an attribute A given a data set S

  - Let Values(A) = the set of values that attribute A can take
  - Let S_v = the set of examples in the data set which have value v for attribute A (for each value v from Values(A))

// see slide 15

information gain (or entropy reduction)

$$gain(S, A) = H(S) - H(S|A)$$

$$= H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \times H(S_v)$$

Some Intuition

$H(\text{output}) = 1$ // total chaos

3 features

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

- *Size* is the least discriminating attribute (i.e. smallest information gain)

- *Shape* and *color* are the most discriminating attributes (i.e. highest information gain)

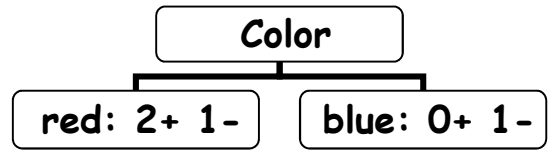
shape, color } more discriminating than size

A Small Example (1)

let's try F2 = color

Size	Color (F2)	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

Values(Color) = {red,blue}



$$\text{gain}(S,\text{Color}) = H(S) - \sum_{v \in \text{values(Color)}} \frac{|S_v|}{|S|} \times H(S_v)$$

entropy of Output

$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$ // total chaos, not knowing anything

for each v of Values(Color)

- $H(S|\text{Color}=\text{red}) = H\left(\frac{2}{3},\frac{1}{3}\right) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) = 0.918$
- $H(S|\text{Color}=\text{blue}) = H(1,0) = -\left(\frac{1}{1}\log_2\frac{1}{1}\right) = 0$ // sure thing

$$H(S|\text{Color}) = \frac{3}{4}(0.918) + \frac{1}{4}(0) = 0.6885$$ knowing the color

Red + Blue

$$\text{gain}(\text{Color}) = H(S) - H(S|\text{Color}) = 1 - 0.6885 = 0.3115$$

Note: by definition,

- $\text{Log}_2 0 = -\infty$
- $0\log_2 0$ is 0

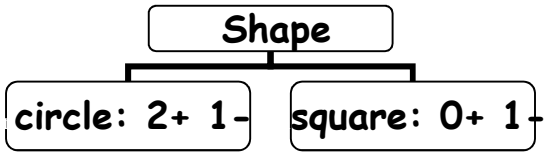
1 time out of 4 we have a blue

3 times out of 4 we have a red

A Small Example (2)

F3

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-



$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

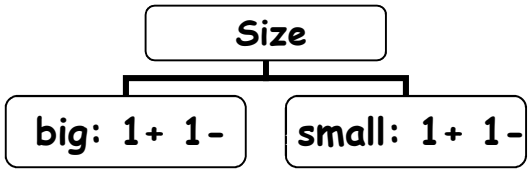
$$H(S|Shape) = \frac{3}{4}(0.918) + \frac{1}{4}(0) = 0.6885$$

$$gain(Shape) = H(S) - H(S|Shape) = 1 - 0.6885 = 0.3115$$

A Small Example (3)

F1

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-



$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

$$H(S|Size) = \frac{2}{4}(1) + \frac{2}{4}(1)$$

$$gain(Size) = H(S) - H(S|Size) = 1-1=0$$

H(output) without knowing any feature

H(output/Size) knowing the size

A Small Example (4)

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

$$\text{gain}(\text{Shape}) = 0.3115$$
$$\text{gain}(\text{Color}) = 0.3115$$
$$\text{gain}(\text{Size}) = 0$$

- So first separate according to either *color* or *shape* (root of the tree)

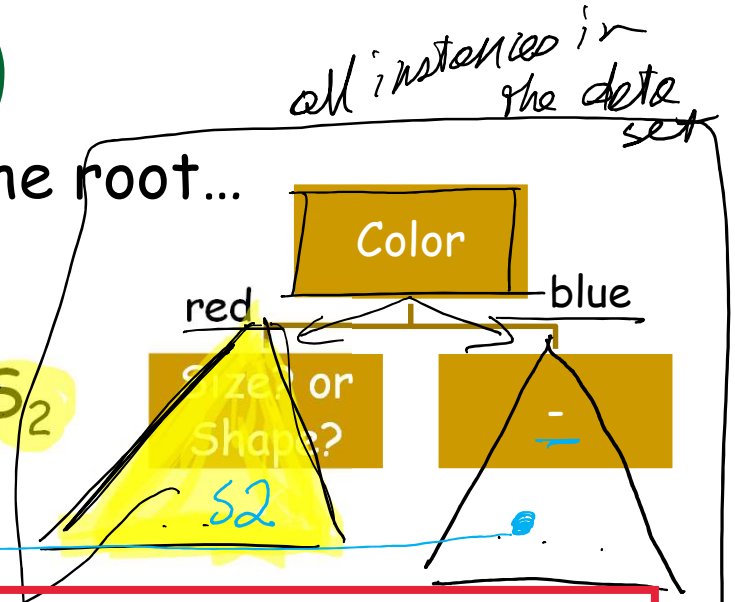
A Small Example (4)

- Let's assume we pick *Color* for the root…

Size	Color	Shape	Output
Big	Red	Circle	+
Small	Red	Circle	+
Small	Red	Square	-
Big	Blue	Circle	-

S2

S_2



$$H(S_2) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)$$

only use the instances with color = red

for each v of Values(Size)

$$H(S_2 \mid Size = big) = H\left(\frac{1}{1}, \frac{0}{1}\right) = 0$$

$$H(S_2 \mid Size = small) = H\left(\frac{1}{2}, \frac{1}{2}\right) = 1$$

$$H(S_2 \mid Size) = \frac{1}{3}(0) + \frac{2}{3}(1)$$

$$gain(Size) = H(S_2) - H(S_2 \mid Size)$$

only use the instances with color = blue

for each v of Values(Shape)

$$H(S_2 \mid Shape = circle) = H\left(\frac{2}{2}, \frac{0}{2}\right) = 0$$

$$H(S_2 \mid Shape = square) = H\left(\frac{0}{1}, \frac{1}{1}\right) = 0$$

$$H(S_2 \mid Shape)$$

$$gain(Shape) = H(S_2) - H(S_2 \mid Shape)$$

Highest gain is the root of the subtree

Back to the Restaurant

- Training data:

pick max info gain

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

source: Norvig (2003)

The Restaurant Example

$\forall$ features

$gain(alt) = \ldots$ $gain(bar) = \ldots$ $gain(fri) = \ldots$ $gain(hun) = \ldots$

$$gain(pat) = 1 - \left(\frac{2}{12} \times H\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} \times H\left(\frac{0}{4}, \frac{4}{4}\right) + \frac{6}{12} \times H\left(\frac{2}{6}, \frac{4}{6}\right)\right)$$

$$= 1 - \left(\frac{2}{12} \times -\left(\frac{0}{2}\log_2\frac{0}{2} + \frac{2}{2}\log_2\frac{2}{2}\right) + \frac{4}{12} \times -\left(\frac{0}{4}\log_2\frac{0}{4} + \frac{4}{4}\log_2\frac{4}{4}\right) + \ldots\right) \approx 0.541 \text{bits}$$

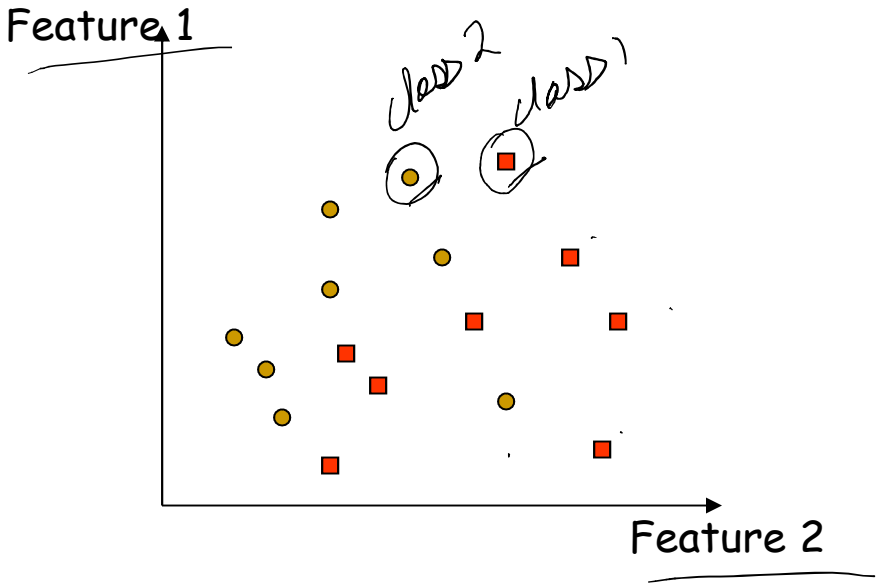
$gain(price) = \ldots$ $gain(rain) = \ldots$ $gain(res) = \ldots$

$$gain(type) = 1 - \left(\frac{2}{12} \times H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} \times H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} \times H\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} \times H\left(\frac{2}{4}, \frac{2}{4}\right)\right) = 0 \text{ bits}$$

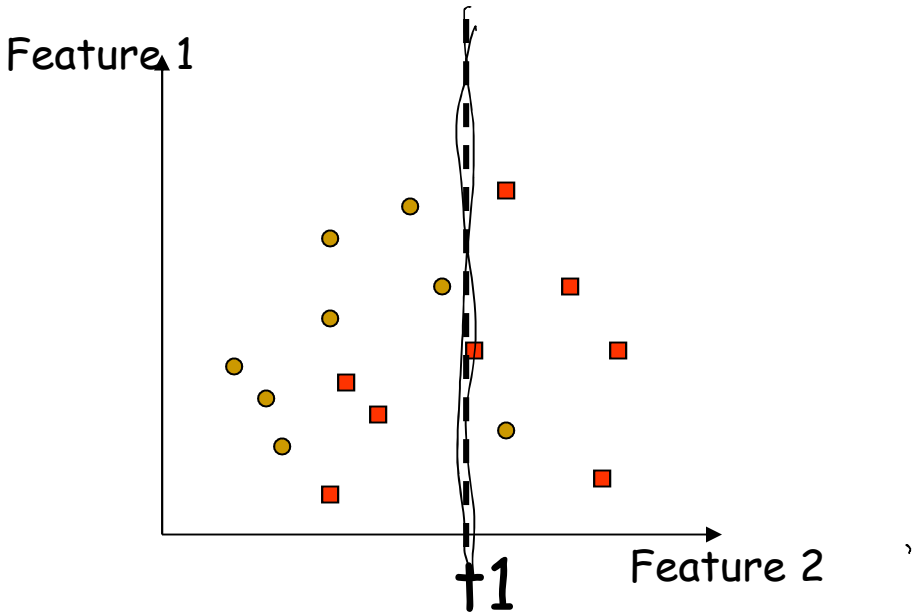
$gain(est) = \ldots$

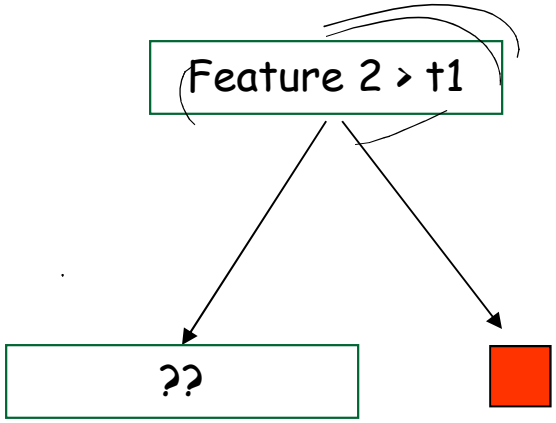
- Attribute pat (Patron) has the highest gain, so root of the tree should be attribute *Patrons*
- do recursively for subtrees

Decision Boundaries of Decision Trees

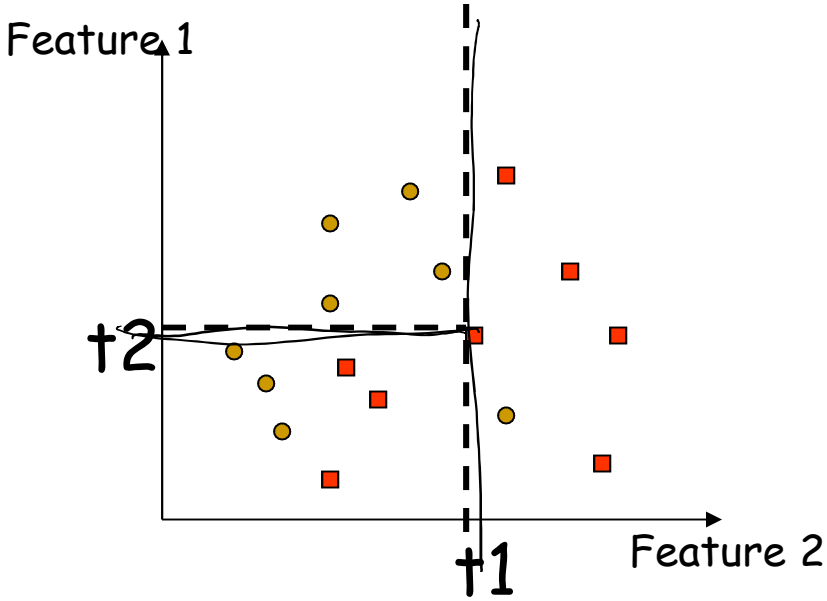


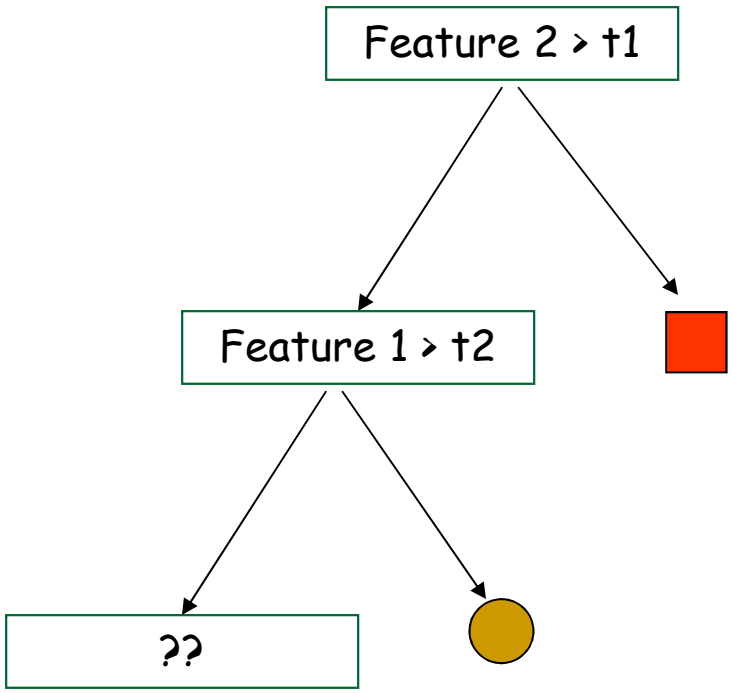
Decision Boundaries of Decision Trees



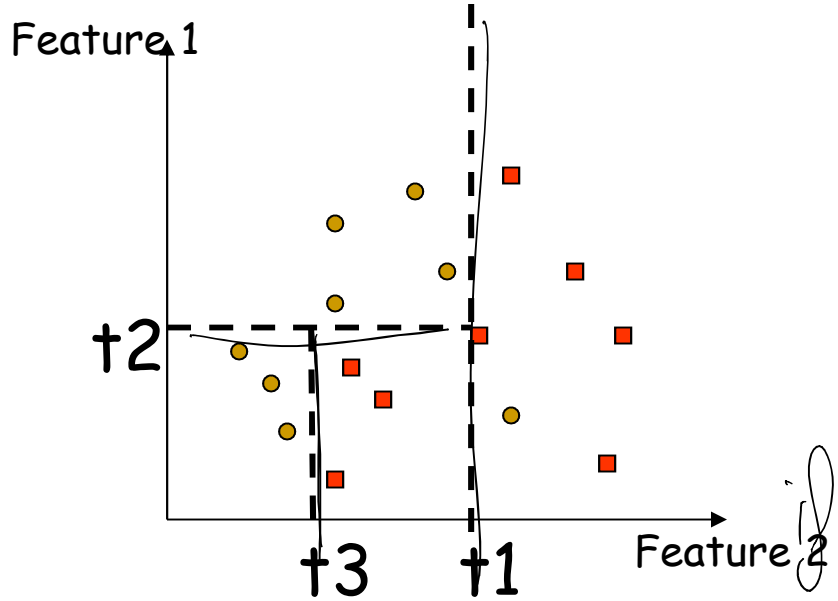


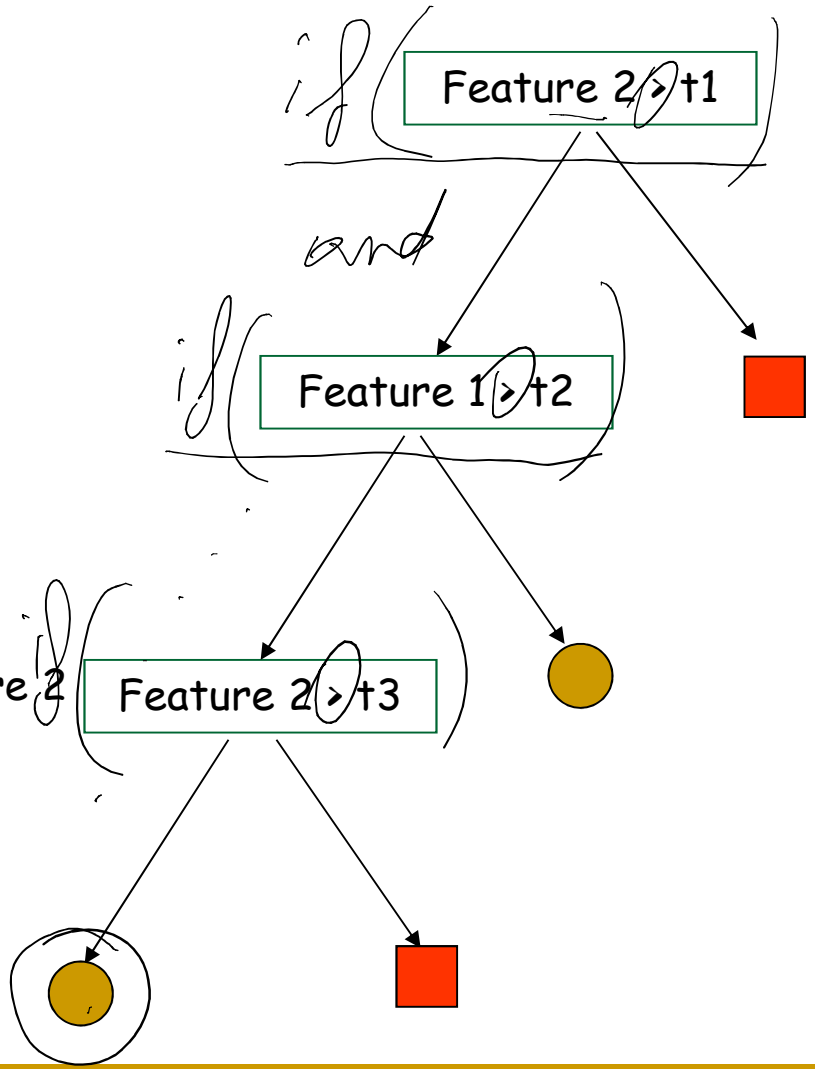
Decision Boundaries of Decision Trees





Decision Boundaries of Decision Trees





Applications of Decision Trees

- One of the most widely used learning methods in practice
  - Fast
  - Simple
  - Traceable (<-- very important!)

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees video #4
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. Evaluation
5. Unsupervised Learning
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

supervised learning

COMP472: Artificial Intelligence
Machine Learning
Evaluation video #5

- Russell & Norvig: Sections 19.4

Today

1. Introduction to ML
2. Naïve Bayes Classification Model #1
   a. Application to Spam Filtering
3. Decision Trees Model #2
4. Evaluation YOU ARE HERE!
5. Unsupervised Learning
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Data Sets

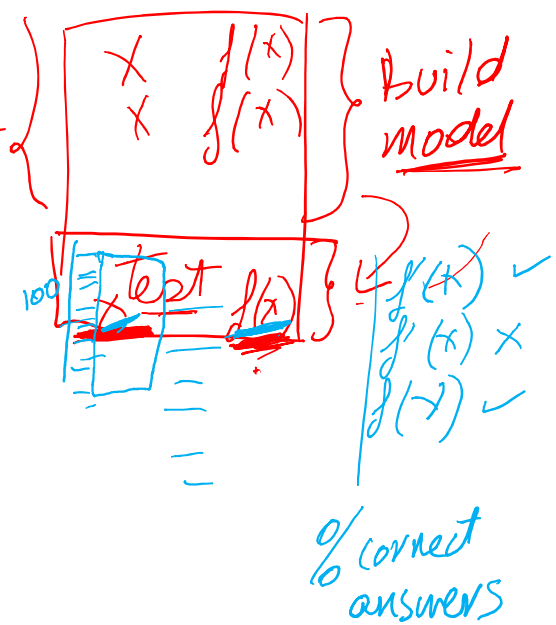
supervised learning

- How do you know if what you learned is correct?
- You run your classifier on a data set of unseen examples (that you did not use for training) for which you know the correct classification

- Split data set into 3 sub-sets
  1. Actual **training** set (~80%)
  2. **Validation** set (~20%)
  3. **Test** set

  ~80%

  ~20%



Standard Methodology

1. Collect a large set of examples (all with correct classifications)
2. Divide collection into training, validation and test set

Loop:

3. Apply learning algorithm to the training set to learn the parameters
4. Measure performance with the validation set, and adjust hyper-parameters* to improve performance
5. Measure performance with the test set

- DO NOT LOOK AT THE TEST SET until step 5.

Parameters:
basic values learned by the ML model. eg.

- for NB: prior & conditional probabilities (likelihood)
- for DTs: features to split
- for ANNs: weights

Hyper-parameters: parameters used to set up the ML model. eg.

- for NB: value of delta for smoothing, (ex. add-1, 0.5, 0.3)
- for DTs: pruning level
- for ANNs: nb of hidden layers, nb of nodes per layer…

Metrics

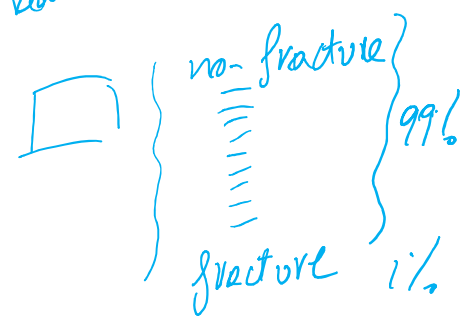
- 1 accuracy
  - % of instances of the test set the algorithm correctly classified
  - when all classes are equally important and represented

    $f(x)$ balanced dataset

- 2 Recall, Precision & F-measure
  - when one class is more important and the others

not balanced

80% dogh
10% dog
10% cat



Accuracy

- % of instances of the test set the algorithm correctly classifies
- when all classes are equally important and represented
- problem:
  - when one class (eg. sick) is more important and the others
  - eg. when data set is unbalanced

correct classification

prediction

	Target	system 1
X1	sick	ok ×
X2	sick	ok ×
X3	sick	ok ×
X4	sick	ok ×
X5	sick	ok ×
X6	ok	ok ✓
X7	ok	ok ✓
…	…	…
…	…	…
X500	ok	ok ✓
Accuracy		495/500 = 99% !

5 mistakes

Recall, Precision

- Recall: What proportion of the instances in the class of interest (eg. sick) are labelled correctly?
- Precision: What proportion of instances labeled with the class of interest (eg. sick) are actually correct?

		Correct class	
		instance should be in class C	instance should not be in class C
Model prediction	instance is put in class C	True Positive (TP)	False Positive (FP)
	instance is not put in class C	False Negative (FN)	True Negative (TN)

$$\text{Precision} = \frac{TP}{TP+FP}$$

TP: *nb of instances that are in class C and that the model identified as class C*

TP+FP: *nb of instances that the model labelled as class C*

$$\text{Recall} = \frac{TP}{TP+FN}$$

TP: *nb of instances that are in class C and that the model identified as class C*

TP+FN: *All instances that are in class C*

Example

correct classification

class of interest = sick

	Target	system 1	system 2	system 3
X1	sick	sick ✓	sick ✓	ok ×
X2	sick	ok ×	ok ×	sick ✓
X3	sick	ok ×	sick ✓	sick ✓
X4	sick	ok ×	sick ✓	sick ✓
X5	sick	ok ×	ok ×	sick ✓
X6	ok	ok ✓	ok ✓	sick ×
X7	ok	ok ✓	ok ✓	sick ×
..	ok	ok ✓	ok ✓	ok ✓
..	ok	ok ✓	ok ✓	ok ✓
X500	ok	ok ✓	ok ✓	ok ✓
Accuracy		496/500 = 99%	498/500 = 99.6%	497/500 = 99.4%
Precision		1/1 = 100%	3/3 = 100%	4/6 = 66.7%
Recall		1/5 = 20%	3/5 = 60%	4/5 = 80%

Which system is better?

Precision = O -> T

Recall = O/T
number of T = number of class of interest

A Single Measure

- cannot take mean of P&R
  - if R = 50% P = 50% M = 50%
  - if R = 100% P = 10% M = 55% (not fair)

1. take harmonic mean
   - which penalizes extreme values

$$HM = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

HM is high only when both P&R are high

if R = 50% and P = 50% HM = 50%

if R = 100% and P = 10% HM = 18.2%

2. if P and R should not have the same importance in the problem domain, take weighted harmonic mean

$$WHM = \frac{1}{\frac{1}{2}\frac{1}{R} + \frac{1}{2}\frac{1}{P}}$$ // if weight R = weight P = $\frac{1}{2}$

$\frac{1}{2} + \frac{1}{2} = 1$

$$WHM = \frac{1}{\frac{1}{a}\frac{1}{R} + \frac{1}{b}\frac{1}{P}}$$ // if weight R = $\frac{1}{a}$ weight P = $\frac{1}{b}$ and $\boxed{\frac{1}{a} + \frac{1}{b} = 1}$

Weighted Harmonic Mean of P&R

$$WHM = \frac{1}{\frac{1}{a}\frac{1}{R} + \frac{1}{b}\frac{1}{P}}$$ // if weight R = $\frac{1}{a}$ weight P = $\frac{1}{b}$ and $\frac{1}{a} + \frac{1}{b} = 1$

1. let $w_R = \frac{\delta}{\delta+1}$ $w_P = \frac{1}{\delta+1}$ // so that $w_R + w_P = \frac{\delta+1}{\delta+1} = 1$

$$WHM = \frac{1}{\left(\frac{\delta}{\delta+1}\right)\frac{1}{R} + \left(\frac{1}{\delta+1}\right)\frac{1}{P}} = \frac{\delta+1}{\delta\frac{1}{R} + 1\frac{1}{P}} = \frac{(\delta+1)PR}{\delta P + 1R}$$

2. let $\delta = \beta^2$

$$WHM = \boxed{\frac{(\beta^2+1)PR}{\beta^2 P + 1R}}$$ // called the F-measure

F-measure

- A weighted harmonic mean of precision and recall

$$F_\beta = \frac{(\beta^2+1)PR}{(\beta^2 P + R)}$$

- β represents the relative importance of recall to precision
  - when $\beta = 1$
    - F1 measure
    - precision & recall have same importance
  - when $\beta > 1$
    - recall is given more weigth
    - e.g. F_2 measure, recall is considered 2x more important than precision
  - when $\beta < 1$
    - precision is given more weigth
    - e.g. $F_{0.5}$ measure, precision is considered 2x more important than recall

$\beta = 2$

$\beta = 0.5$

Example

	Target	*system 1*	*system 2*	*system 3*
X1	sick	sick ✓	sick ✓	ok ×
X2	sick	ok ×	ok ×	sick ✓
X3	sick	ok ×	sick ✓	sick ✓
X4	sick	ok ×	sick ✓	sick ✓
X5	sick	ok ×	ok ×	sick ✓
X6	ok	ok ✓	ok ✓	sick ×
X7	ok	ok ✓	ok ✓	sick ×
..	ok	ok ✓	ok ✓	ok ✓
..	ok	ok ✓	ok ✓	ok ✓
X500	ok	ok ✓	ok ✓	ok ✓
Accuracy		496/500 = 99%	498/500 = 99.6%	498/500 = 99.6%
Precision		1/1 = 100%	3/3 = 100%	4/6 = 66.7%
Recall		1/5 = 20%	3/5 = 60%	4/5 = 80%
F1-measure 2PR/(P+R)		2*100*20/ (100+20) = 33%	75%	72.9%

$\beta = 1$ $F_1 = \frac{2PR}{P+R}$

P, R and F for Multiclass Classification

- previous P, R and F are ok when 1 particular class interests us (eg. sick)
- What if several classes interest us?
- then
  - compute *per-class* P, R, F
  - and to have a single measure for all classes: combine per-class F-measures via
    - macro F-measure, or
    - weighted-average F-measure

Per-class Precision & Per-class Recall

gold

		Correct Class			
		Cat	Dog	Fish	Total
Predicted Class	Cat	4	6	3	13
	Dog	1	2	0	3
	Fish	1	2	6	9
	Total	6	10	9	25

model prediction

images

- precision of class Cat: 4/(4+6+3) = 30.8%
- precision of class Dog: 2/(1+2+0) = 66.7%
- precision of class Fish: 6/(1+2+6) = 66.7%

- recall of class Cat: 4/(4+1+1) = 66.7%
- recall of class Dog: 2/(2+6+2) = 20%
- recall of class Fish: 6/(3+0+6) = 66.7%

Per-class F1-measure

Model #1

	Precision	Recall	F1
Cat	30.8%	66.7%	42.1%
Dog	66.7%	20.0%	30.8%
Fish	66.7%	66.7%	66.7%

F1 = 2PR/(P+R)

$F_{b=1}$

Model #2

F1

- F1 of class Cat: (2 x .308 x .667) / (.308 + .667) = 0.421
- F1 of class Dog: (2 x .667 x .200) / (.667 + .200) = 0.308
- F1 of class Fish: (2 x .667 x .667) / (.667 + .667) = 0.667

Macro and Weighted-Average Measures

macro precision,
macro recall,
macro F1

	Precision	Recall	F1
Cat	30.8%	66.7%	42.1%
Dog	66.7%	20.0%	30.8%
Fish	66.7%	66.7%	66.7%
average	(30.8+66.7+66.7) / 3 = 54.7%	(66.7 + 20.0 + 66.7) / 3 = 51.1%	(42.1+30.8+66.7) = 46.5%
weighted-average	(6x30.8 // 6 cat + 10x66.7 // 10 dog + 9x66.7) // 9 fish / (6+10+9) // 25 samples = 58.1%	(6x66.7 + 10x20.0 + 6x66.7) / 25 = 48.0%	(6x42.1 + 10x30.8 + 6x66.7) / 25 = 46.4%

weighted-averaged precision,
weighted-averaged recall,
weighted-averaged F1

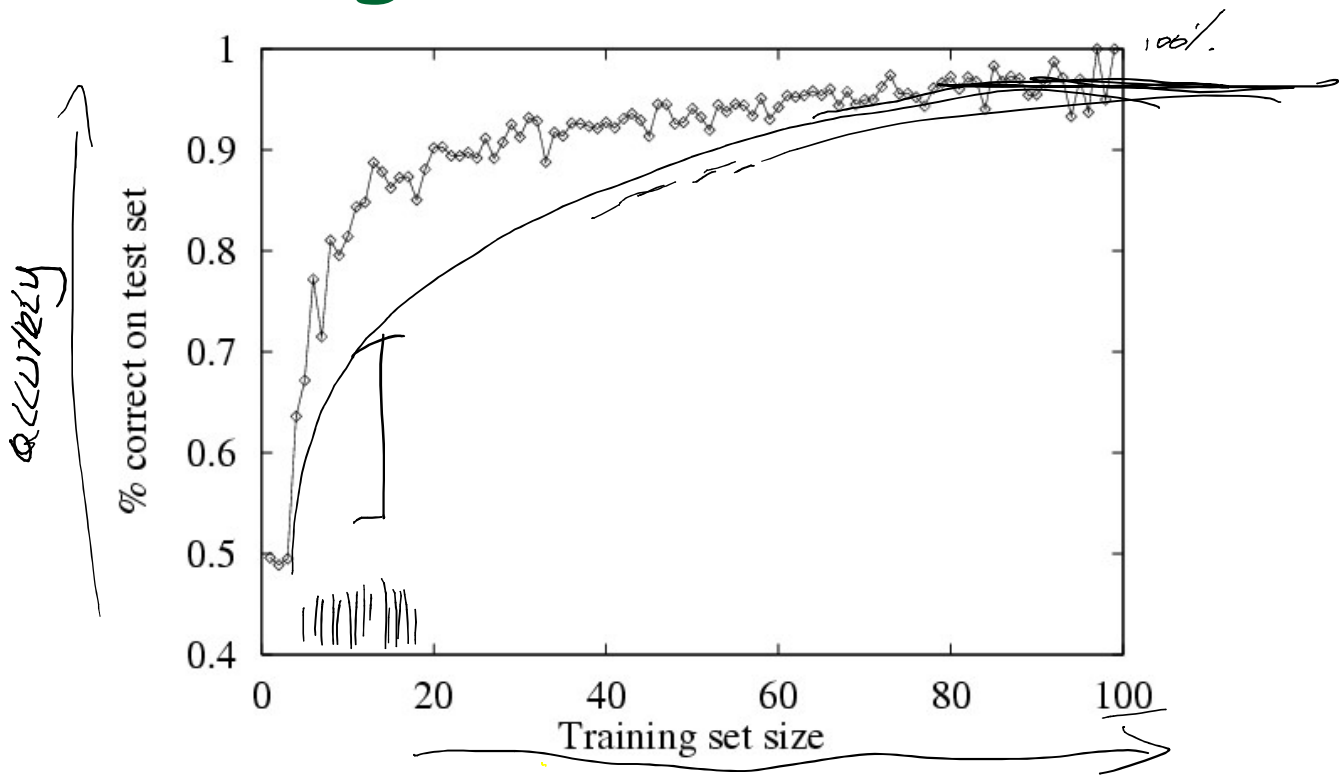
- to combine measures into a single one, we can:
  - take simple average
    - --> macro precision, macro recall, macro F1
  - take weighted average
    - ie. weight the average based on the nb of samples from each class
    - --> weighted averaged precision, weighted averaged recall, weighted averaged F1

Confusion Matrix

- to do an error analysis and find out where the model went wrong ?
- aka contingency table
- eg. 6 classes, 100 test instances

		Correct Class						
		C1	C2	C3	C4	C5	C6	Total
Predicted Class	C1	10✓	3×	0	0	3×	0	16
	C2	0	12✓	3×	4×	0	0	19
	C3	0	1×	9✓	2×	1×	2×	15
	C4	0	1×	3×	5✓	2×	0	11
	C5	0	0	3×	2×	10✓	3×	18
	C6	0	0	5×	0	5×	11✓	21
	Total	10	17	23	13	21	16	100

A Learning Curve



- Size of training set
  - the more, the better
  - but after a while, not much improvement…

source: Mitchell (1997)

Some Words on Training

- In all types of learning… watch out for:
  - Noisy input
  - Overfitting/underfitting the training data

Noisy Input

- In all types of learning… watch out for:
  - Noisy Input:
    - Two examples have the same feature-value pairs, but different outputs

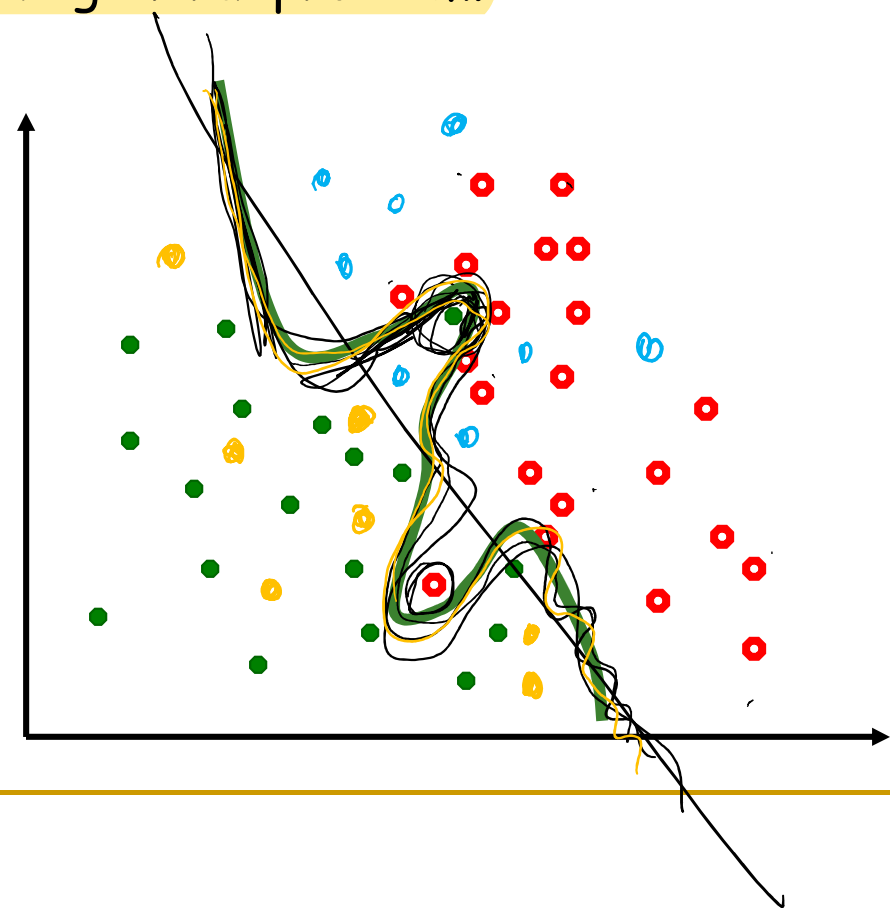
4th feature

Size	Color	N_F	Shape	Output
Big	Red	a	Circle	+
Big	Red	b	Circle square	+

    - Some values of features are incorrect or missing (ex. errors in the data acquisition)
    - Some relevant attributes are not taken into account in the data set

Overfitting

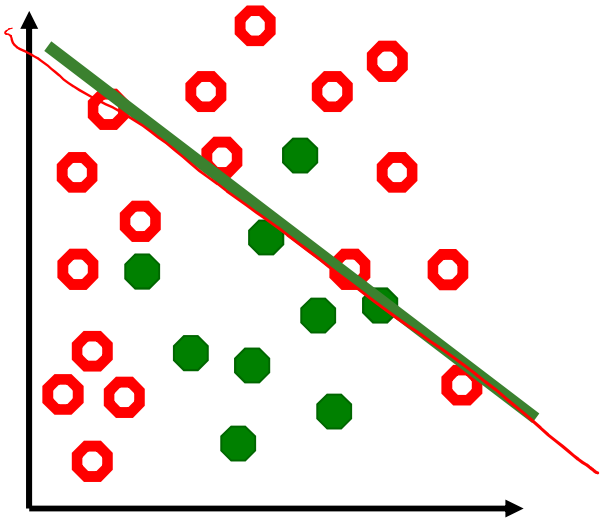
- If a large number of irrelevant features are there, we may find meaningless regularities in the data that are particular to the training data but irrelevant to the general problem.
- Complicated boundaries *overfit* the data
- they are too tuned to the particular training data at hand
- They do not *generalize* well to the new data
- Extreme case: "rote learning"



- Training error is low
- Testing error is high

Underfitting

- We can also underfit data, i.e. find a decision boundary that is too simple
- Model is not expressive enough (not enough features, or not enough capacity)
- eg. There is no way to fit a linear decision boundary so that the training examples are well separated



- Training error is high
- Testing error is high

Cross-validation

- K-fold cross-validation
  - run k experiments, each time you test on 1/k of the data, and train on the rest
  - than you average the results
- ex: 10-fold cross validation
  1. Collect a large set of examples (all with correct classifications)
  2. Divide collection into two disjoint sets: training (90%) and test (10% = 1/k)
  3. Apply learning algorithm to training set
  4. Measure performance with the test set
  5. Repeat steps 2-4, with the 10 different portions
  6. Average the results of the 10 experiments

entire DS

exp1:	train									test 1/k
exp2:	train								test 1/k	train
exp3:	train							test	train	
…	…									

exp 10

F-measure = 80 80
F-measure 8/60
F - 83 99
F - 82 50

stable unstable

average

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. Evaluation
5. Unsupervised Learning
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

COMP 472: Artificial Intelligence
Machine Learning
Unsupervised Learning video #6

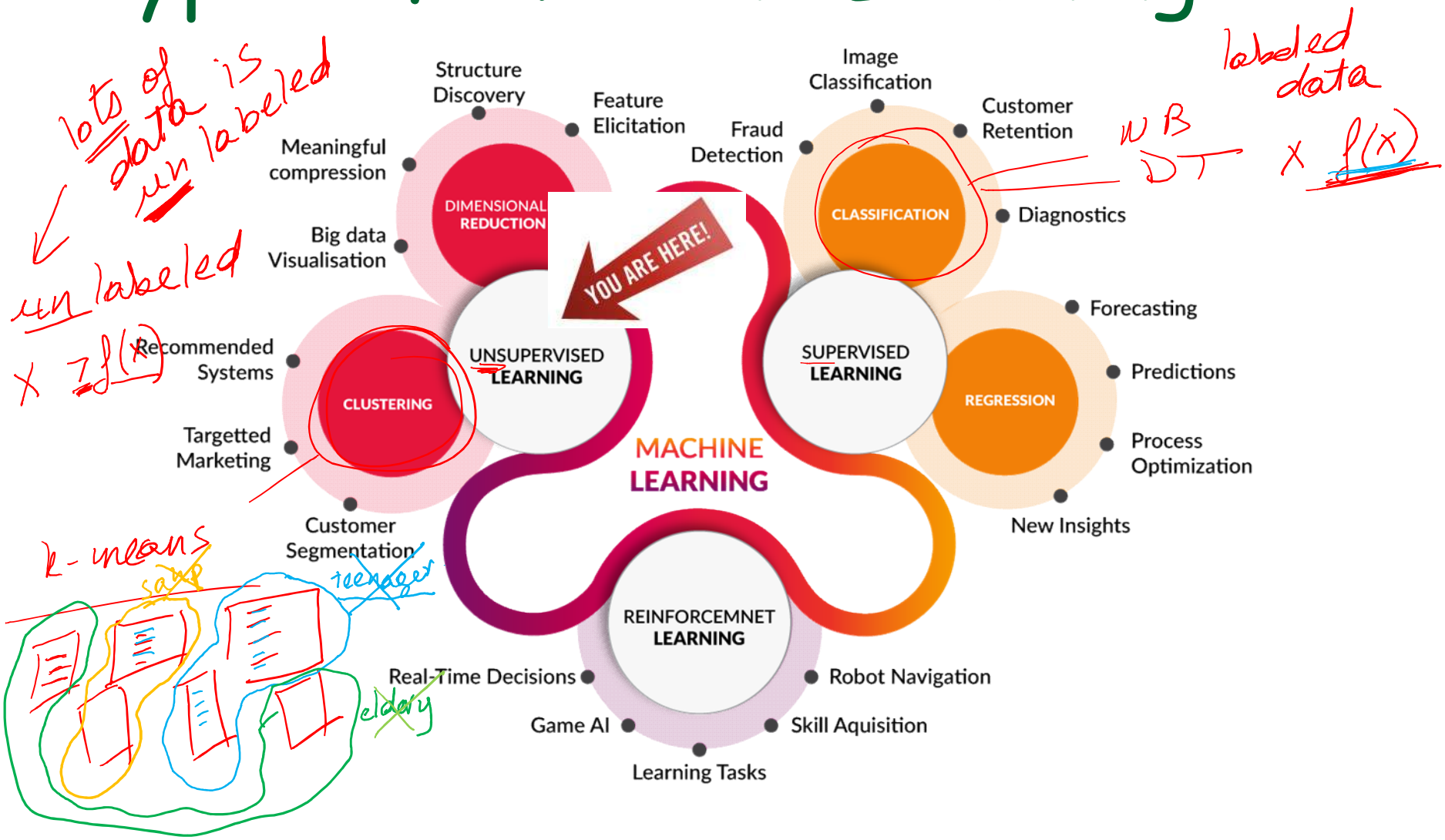
- Russell & Norvig: *not much really*

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. Evaluation
5. Unsupervised Learning
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

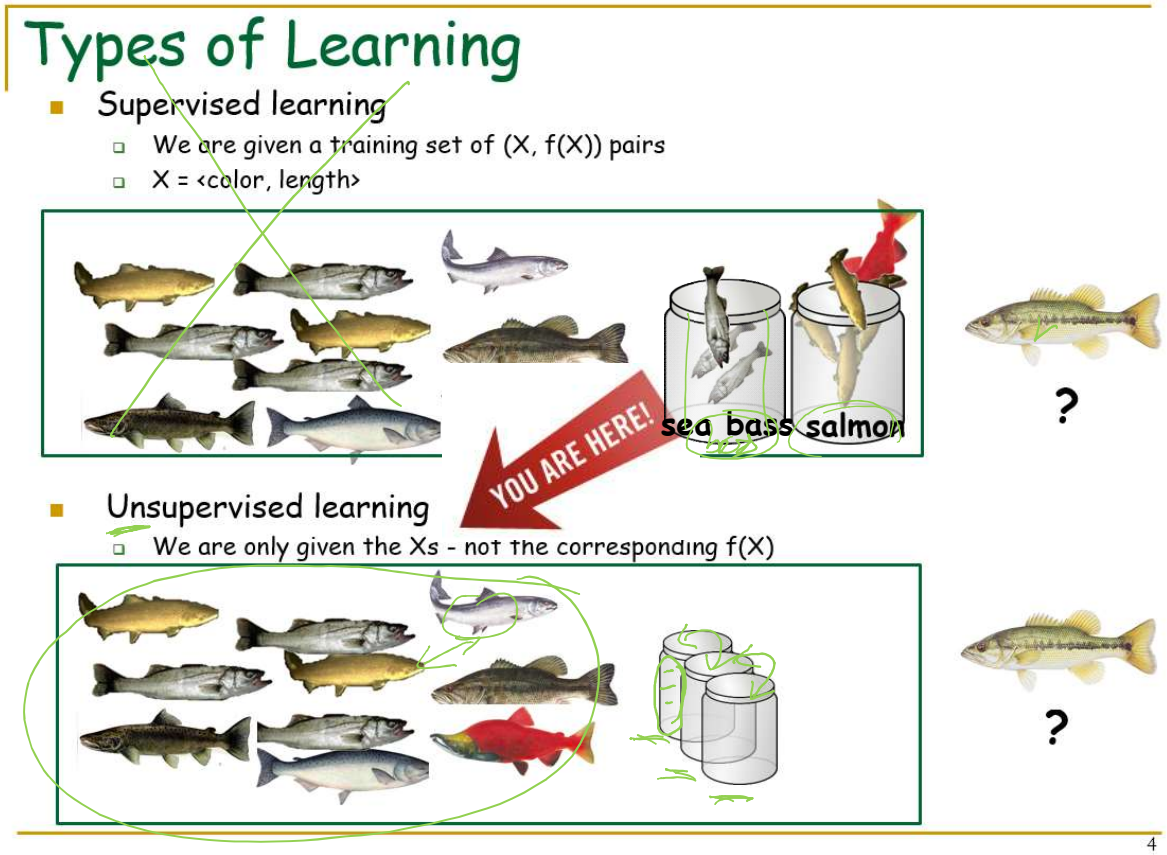
supervised

Types of Machine Learning



http://www.cognub.com/index.php/cognitive-platform/

Remember this slide?



Unsupervised Learning

- Learn without labeled examples
  - i.e. X is given, but not f(X)

small nose	big teeth	small eyes	moustache	f(X) = ?

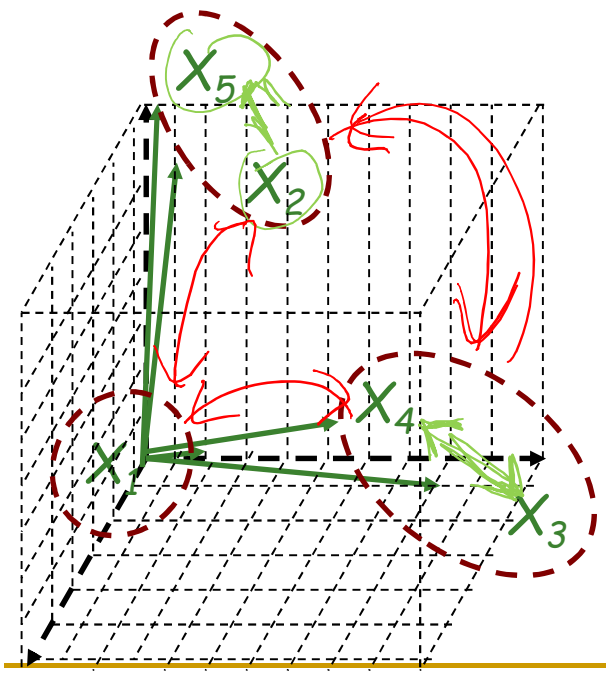
X

not given

- Without a f(X)
  - you can't really identify/label a test instance
  - but you can:
    - Cluster/group the features of the test data into a number of groups
    - Discriminate between these groups without actually labeling them

Clustering

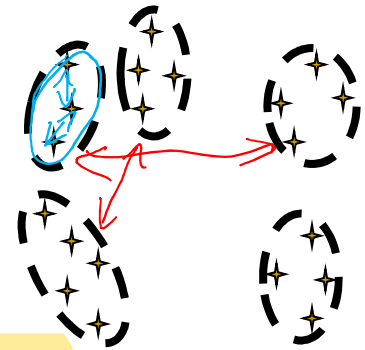
- Represent each instance as a vector <a_1, a_2, a_3,…, a_n>
- Each vector can be visually represented in a n dimensional space



	a_1	a_2	a_3	Output
X_1	1	0	0	?
X_2	1	6	0	?
X_3	8	0	1	?
X_4	6	1	0	?
X_5	1	7	1	?

k-means Clustering

1. Represent each instance as a point on a n dimensional space
2. Partition points into k regions such that:
   - distance between points within a region is minimized
   - distance between points across regions is maximized

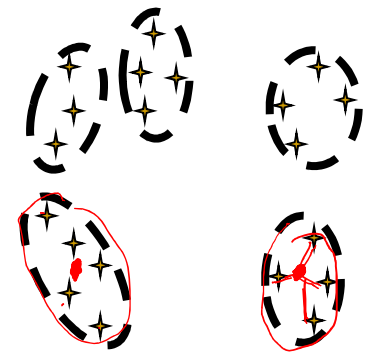


- Naturally works well with features with numerical values
  - where distance between points can be measured by the Euclidean distance
- Needs modifications for categorical values
  - which have no order
    - eg. "Honda", "Audi", "BMW", "Ferrari", "Nissan", "Lamborghini"
  - needs domain-specific distance measure

dist(Honda, Nissan) = 1
dist(Honda, Audi) = 3
dist(Ferrari, Lamborghini) = 1

editdist(Honday, Audi) > editdist(Honday, nissan)

k-means Clustering



- User selects how many clusters they want (the value of k)

1. Place k points into the space (eg. at random). These points represent initial ~~group~~ cluster centroïds.
2. Assign each data point x_n to the nearest centroïd.
3. When all data points have been assigned, recalculate the positions of the k centroïds as the average of the cluster
4. Repeat Steps 2 and 3 until none of the data instances change ~~group~~ cluster.

Euclidean Distance

- To find the nearest centroïd…
  - typical metric is the Euclidean distance
  - Euclidean distance between 2 pts:

$$p = (p_1, p_2, \ldots, p_n)$$
$$q = (q_1, q_2, \ldots, q_n)$$

dist(p,q)

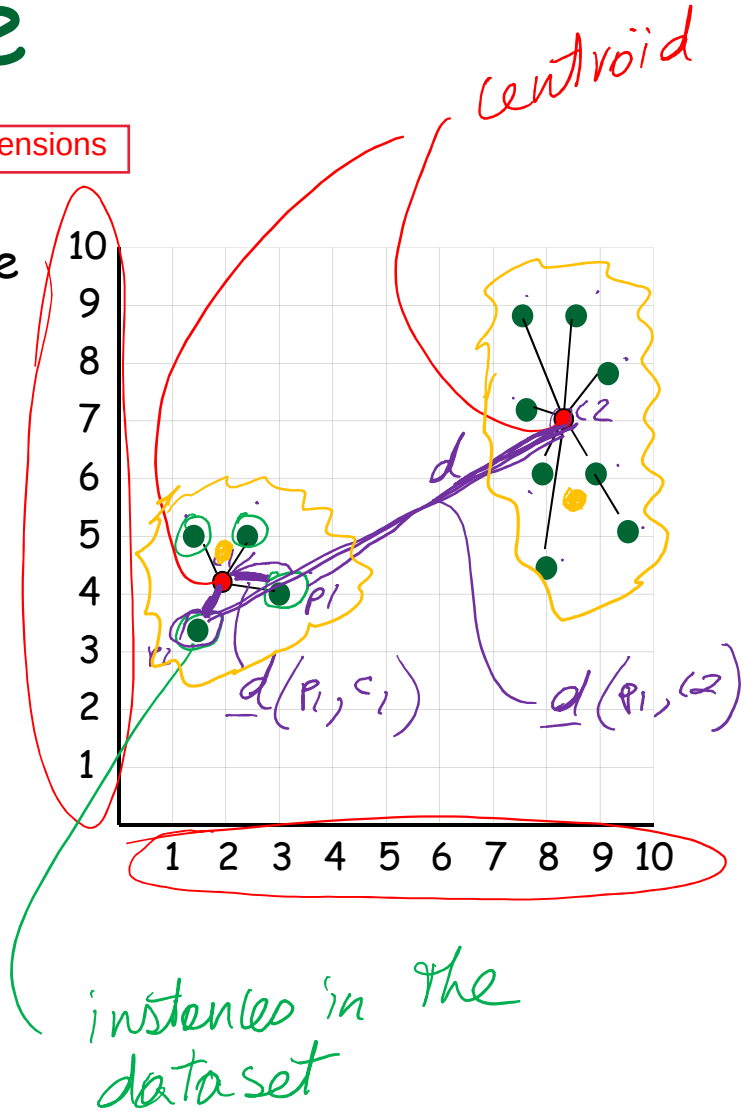
$$d = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

For all dimensions

- To compute the next generation of centroïds…
  - take mean of all points in the cluster in each dimension
  - mean of 2 points:

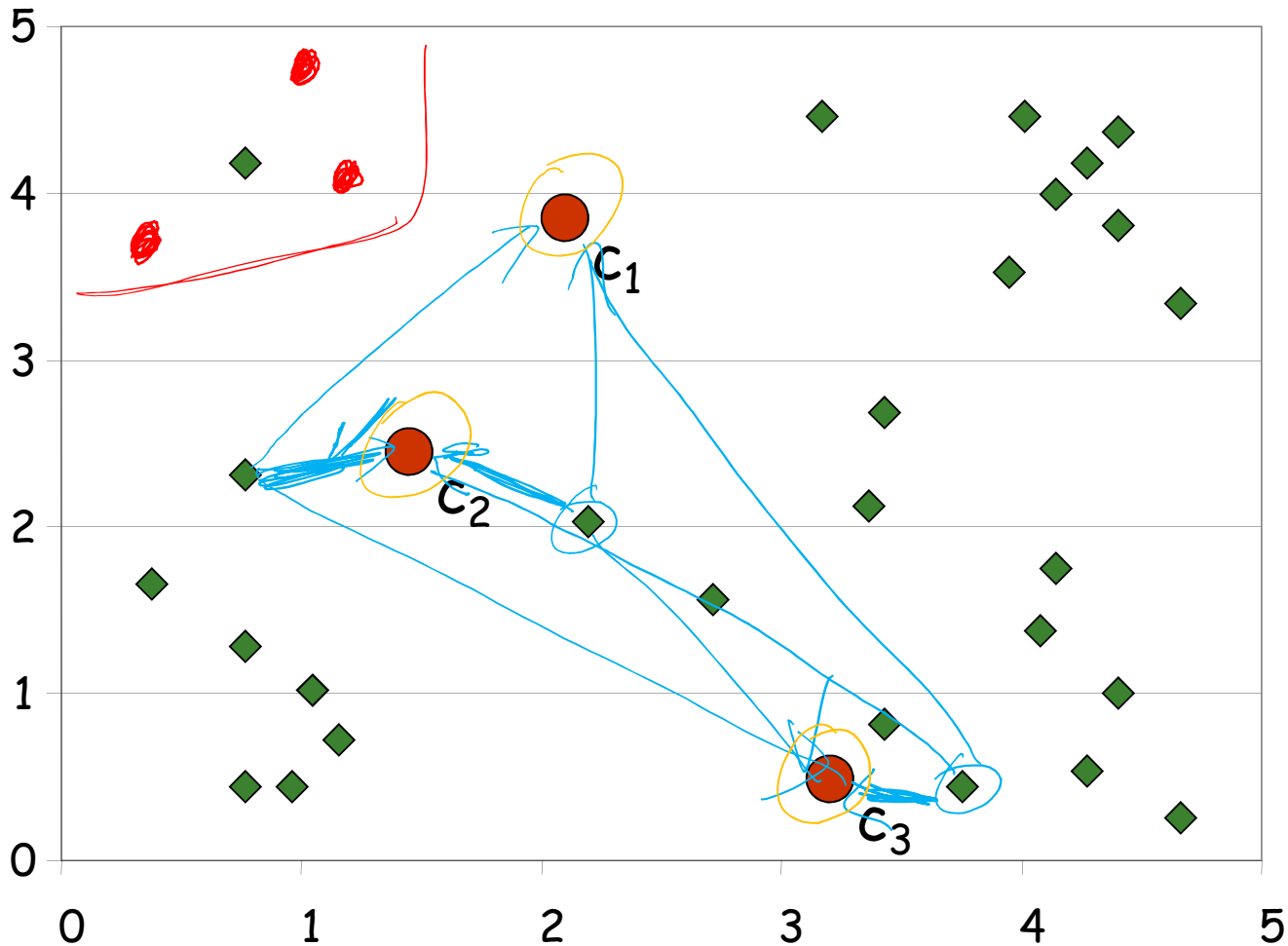
$$p = (p_1, p_2, \ldots, p_n)$$
$$q = (q_1, q_2, \ldots, q_n)$$

$$c = \left(\frac{p_1 + q_1}{2}, \frac{p_2 + q_2}{2}, \ldots, \frac{p_n + q_n}{2}\right)$$



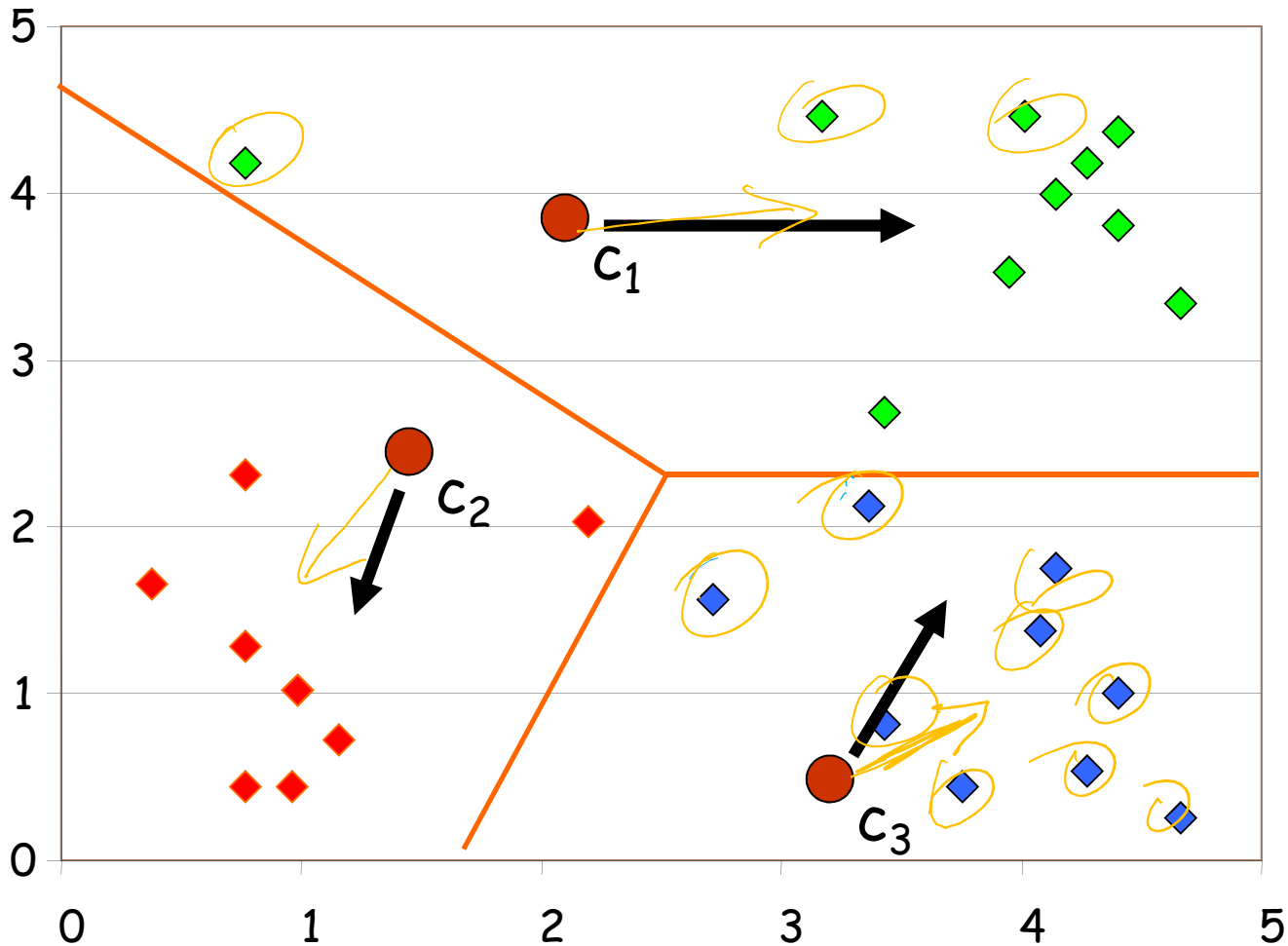
Example (in 2-D… i.e. 2 features)

initial 3 random centroïds



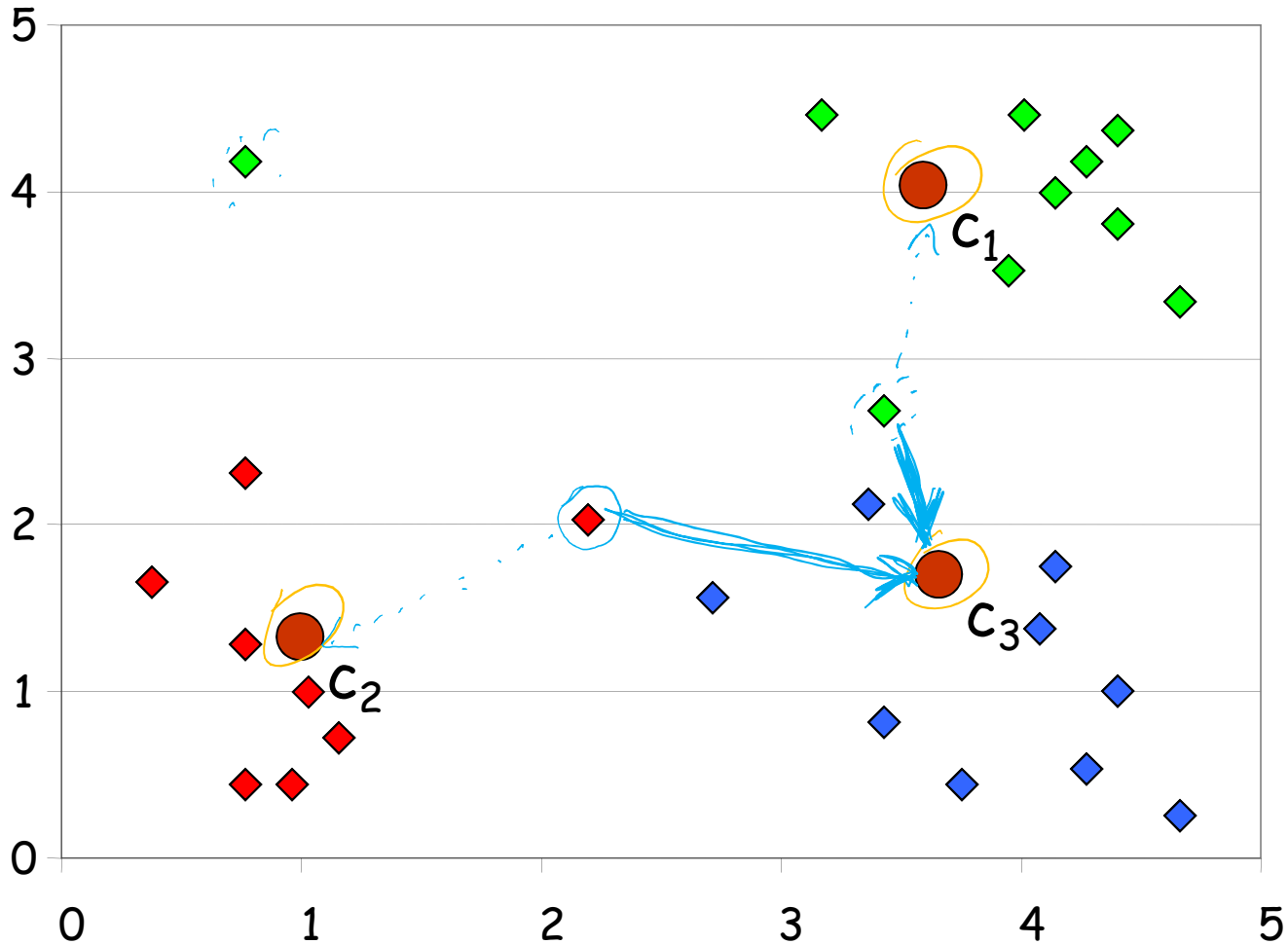
Example

partition data points to closest centroïd



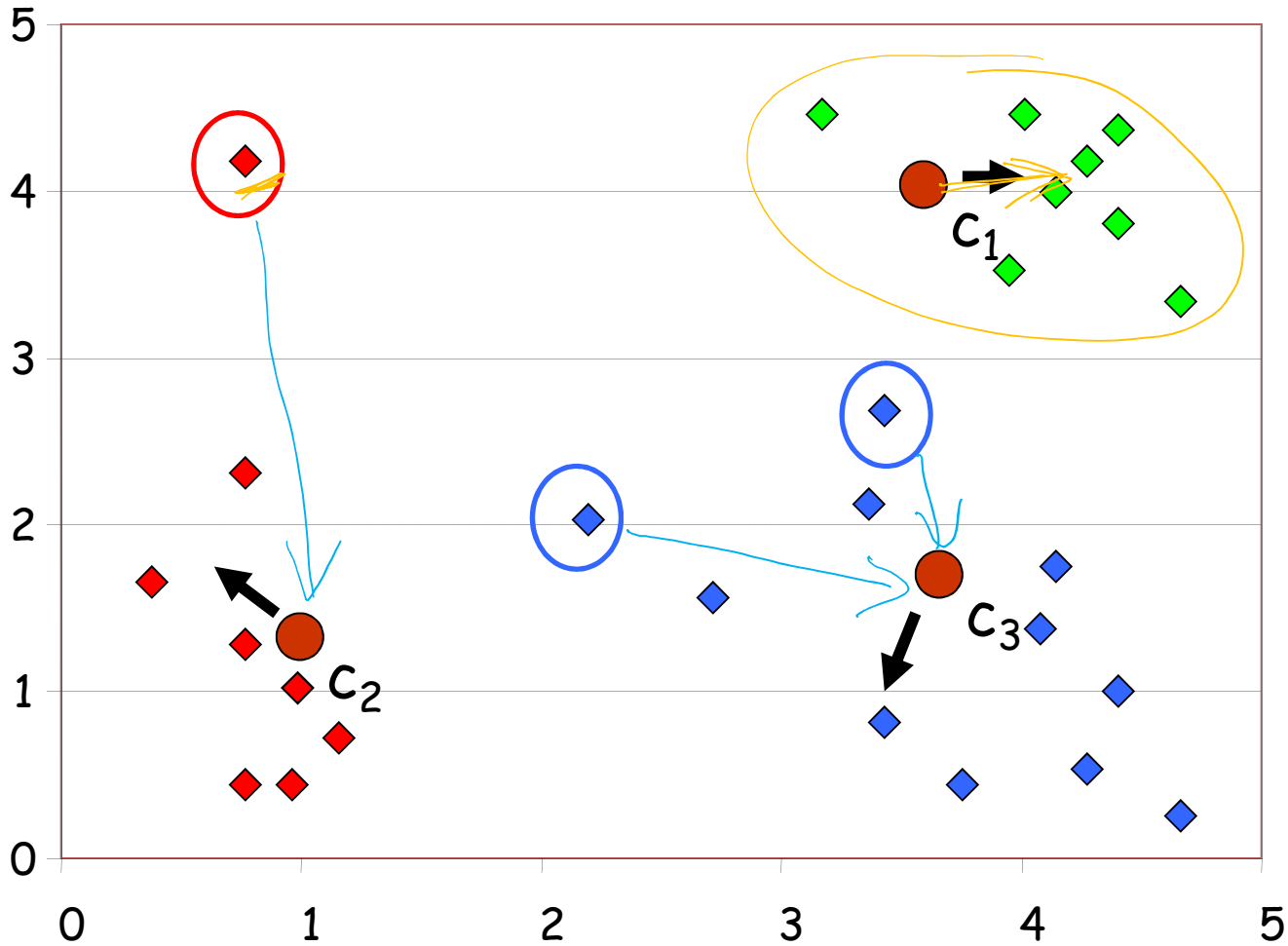
Example

re-compute new centroids

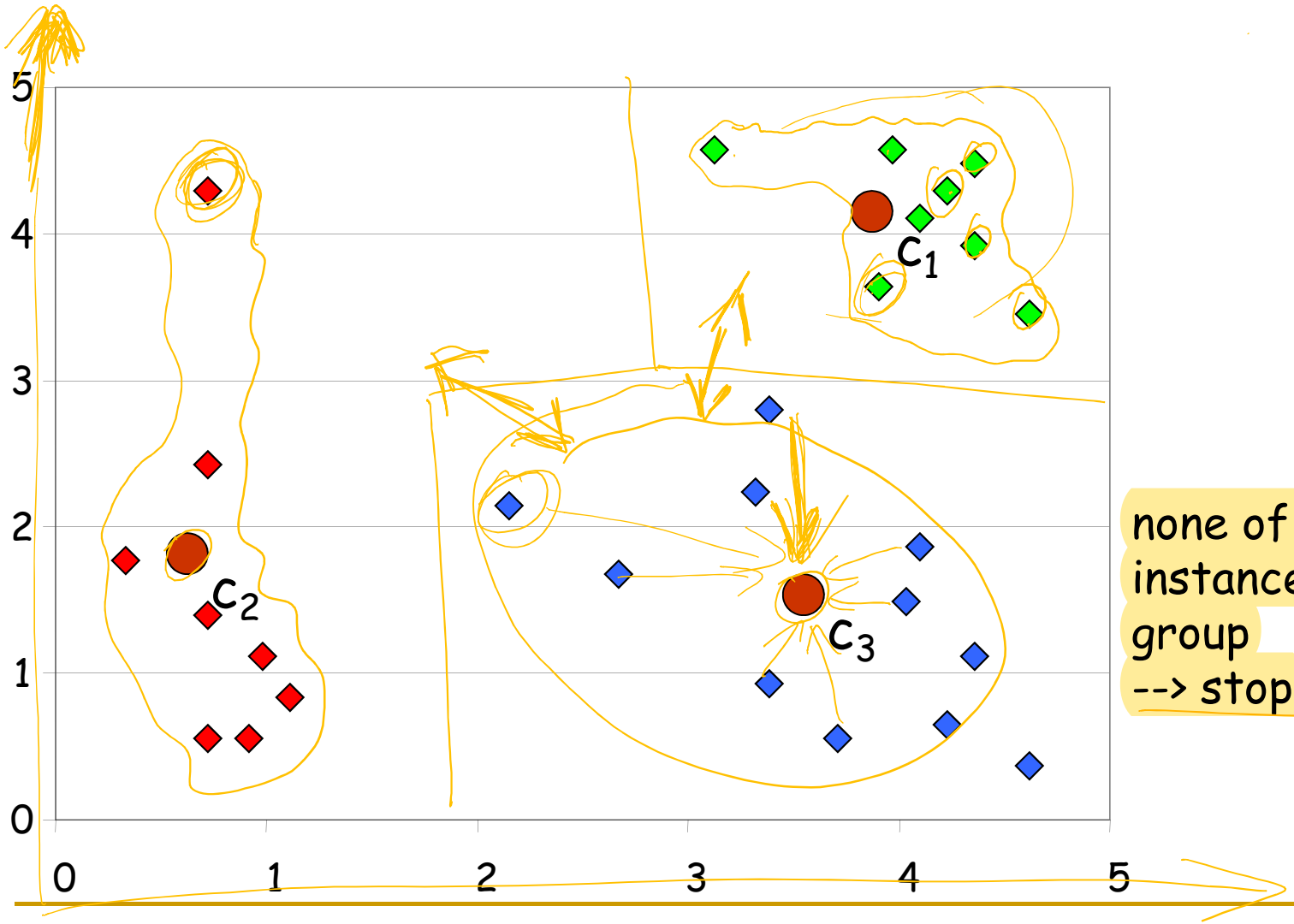


Example

re-assign data points to new closest centroïds



Example



none of the data instances change group
--> stop

Notes on k-means

- negatives:
  - does not guarantee to converge to the global optimum
  - very sensitive to initial choice of centroids
    - many find useless clusters…
  - user must set initial k
    - not easy to do…
- but converges very fast!
- many other clustering algorithms…

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks video #7
   a. Perceptrons
   b. Multi Layered Neural Networks

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

COMP 472 Artificial Intelligence
Machine Learning
Intro to Neural Networks & Perceptrons

Video #7

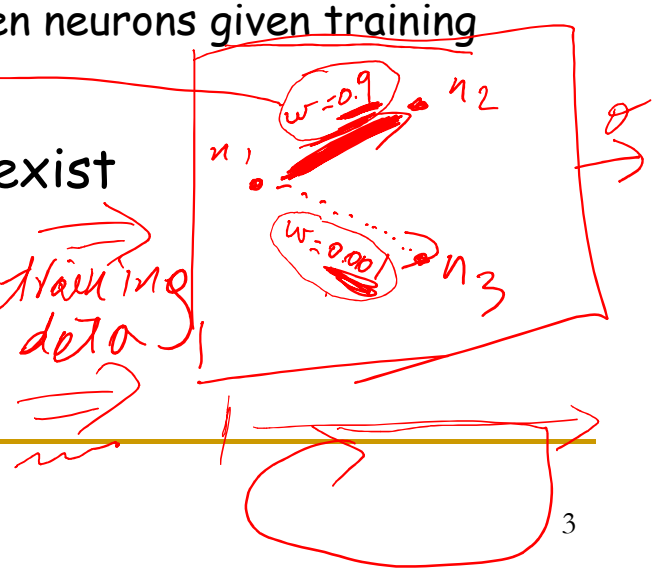
- Russell & Norvig: Sections 19.6, 21.1

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks YOU ARE HERE!
   a. Perceptrons
   b. Multi Layered Neural Networks

Neural Networks

- Learning approach inspired by biology
  - the neurons in the human brain
- Set of many simple processing units (called neurons) connected together
  - the behavior of each neuron is very simple
  - but a network of neurons can have sophisticated behavior and can be used for complex tasks
  - neurons are connected to each other to form a network
  - the strength of the connection between neurons are determined by weights
  - the network learns by learning the weights between neurons given training data
- Different types of network architectures exist
  - feed forward neural networks (FFNN)
  - recurrent neural networks (RNN)
  - convolutional neural networks (CNN)
  - ...

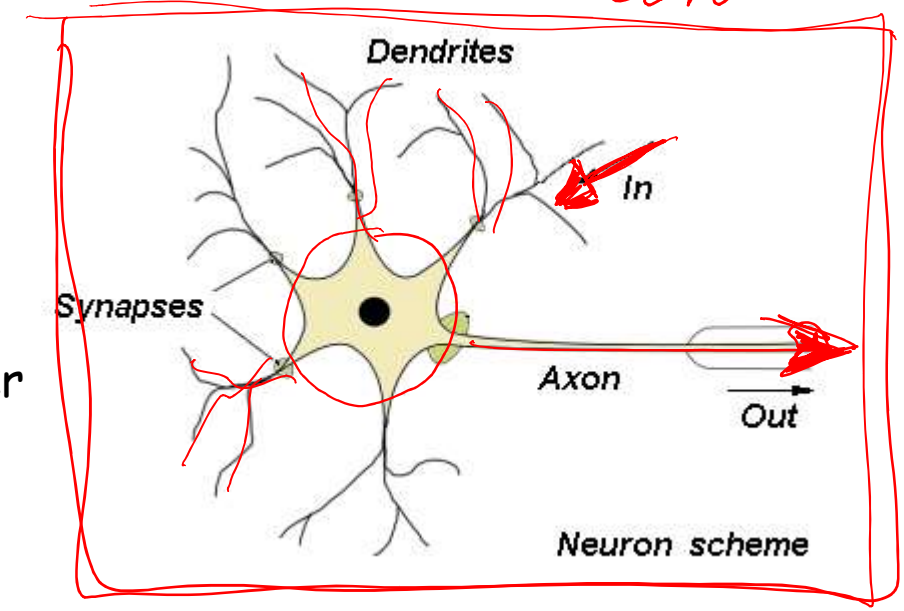


Biological Neurons

- Human brain =
  - 100 billion neurons
  - each neuron may be connected to 10,000 other neurons
  - passing signals to each other via 1,000 trillion **synapses**

- A neuron is made of:
  - Dendrites: filaments that provide input to the neuron
  - Axon: sends an output signal
  - Synapses: connection with other neurons – releases neurotransmitters to other neurons

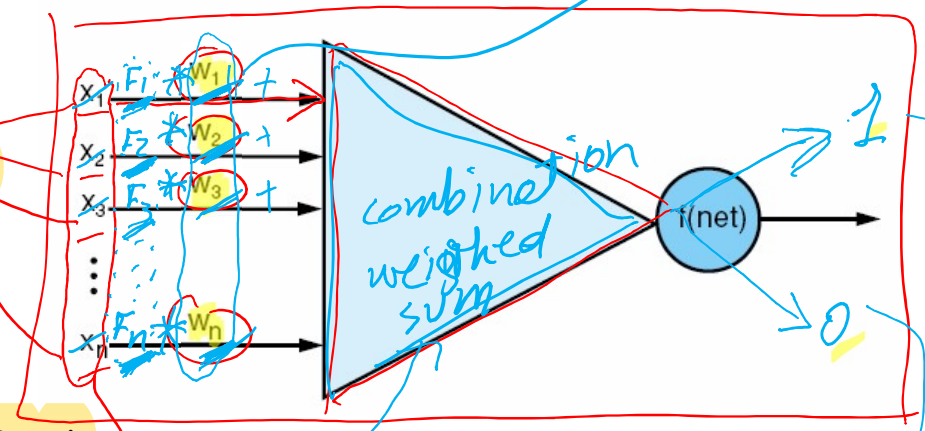
1 neuron



Source: http://www.human-memory.net/brain_neurons.html

A Perceptron = 1 single artificial neuron

will be learned



= n features

- A single computational neuron (no network yet…)
- Input:
  - input signals x_i
  - weights w_i for each feature x_i
    - represents the strength of the connection with the neighboring neurons

⚠ features x_n

- Output:
  - if sum of input weights >= some threshold, neuron fires (output=1)
  - otherwise output = 0
    - If $(w_1 x_1 + \ldots + w_n x_n) >= t$
    - Then output = 1
    - Else output = 0

threshold

- Learning :
  - use the training data to adjust the weights in the perceptron

source: Luger (2005)

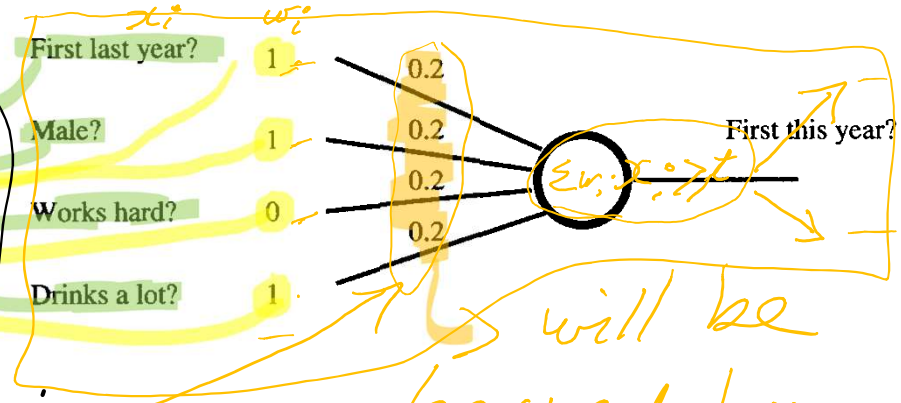
The Idea

4 features

2-class classification

Target class ≡ expected class

	Features (x_i)				Output
Student	First last year?	Male?	Works hard?	Drinks ?	First this year?
Richard	Yes / 1	Yes / 1	No / 0	Yes / 1	No / 0
Alan	Yes / 1	Yes / 1	Yes / 1	No / 0	Yes
...					



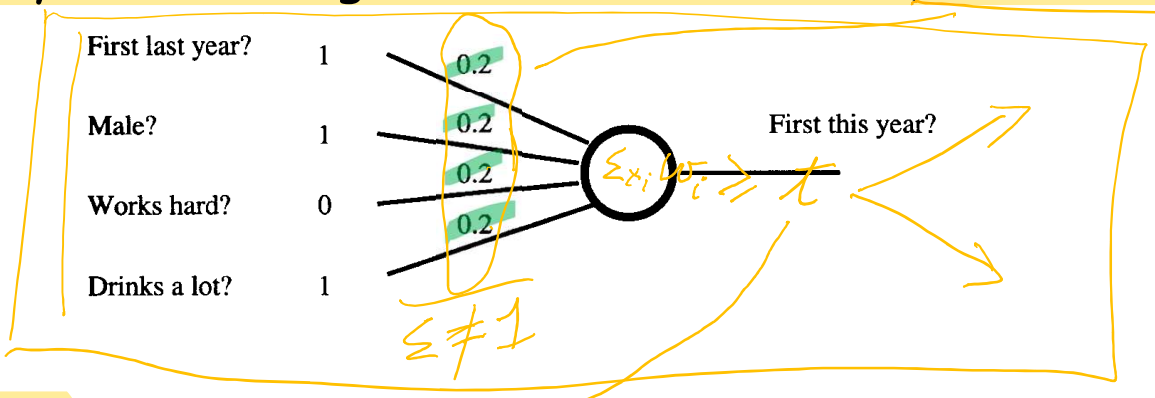
will be learned by the network using the training set

1. Step 1: Set weights to random values
2. Step 2: Feed perceptron with an input
3. Step 3: Compute the network outputs
4. Step 4: Adjust the weights
   1. if output correct → weights stay the same
   2. if output = 0 but it should be 1 →
      - increase weights on *active* connections (i.e. input x_i =1)
   3. if output = 1 but should be 0 →
      - decrease weights on *active* connections (i.e. input x_i =1)
5. Step 5: Repeat steps 2 to 4 a large number of times until the network converges to the right results for the given training examples

source: Cawsey (1998)

A Simple Example

- Turn feature values into numerical values
  - yes -> 1 no -> 0
  - e.g. if $x_1 = 1$, then student got an A last year
  - e.g. if $x_1 = 0$, then student did not get an A last year
- Initially, set all weights to random values (all 0.2 here)



- Assume:
  - threshold = 0.55 #1
  - constant learning rate = 0.05 #2

source: Cawsey (1998)

A Simple Example (2)

Target

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
Richard	yes/ 1	1	no/ 0	1	no **0** Target T
Alan	yes/ 1	1	1	no/ 0	yes **1**
Alison	0	0	1	0	**0**
Jeff	0	1	0	1	**0**
Gail	1	0	1	1	**1**
Simon	0	1	1	1	**0**

$\boxed{\sum_{i=1}^{4} x_i w_i} \geq t$

- Richard:
  - $(1 \times \underset{w_1}{0.2}) + (1 \times \underset{w_2}{0.2}) + (0 \times \underset{w_3}{0.2}) + (1 \times \underset{w_4}{0.2}) = 0.6 >= 0.55$ -->output is 1 — 0 output
  - non-active
  - ...but he did not get an A this year
  - So reduce weights of all active connections (with input 1) by 0.05. — learning rate
    Do not change the weight of the inactive connections.
  - So we get $w_1 = 0.15$, $w_2 = 0.15$, $w_3 = 0.2$, $w_4 = 0.15$

Bcs w(3,richard) = 0

A Simple Example (3)

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
~~Richard~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~
Alan	1	1	1	0	1 Target
Alison	0	0	1	0	0
Jeff	0	1	0	1	0
Gail	1	0	1	1	1
Simon	0	1	1	1	0

$w_1 w_2 w_3 w_4$

- Alan:
  - $(1 \times 0.15) + (1 \times 0.15) + (1 \times 0.2) + (0 \times 0.15) = 0.5 < 0.55$ → output is 0 Output
    w_1 w_2 w_3 w_4 $< t$ non-active
  - … but expected output is 1
  - So increase all weights of active connections by 0.05 constant learning rate
  - So we get $w_1 = 0.2$, $w_2 = 0.2$, $w_3 = 0.25$, $w_4 = 0.15$
- Alison…Jeff… Gail… Simon…

$w_1 w_2 w_3 w_4$ $w_1 w_2 w_3 w_4$

A Simple Example (4)

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
~~Richard~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~
~~Alan~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~
~~Alison~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~
~~Jeff~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~
~~Gail~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~
~~Simon~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~

- epoch 1: Richard, Alan, Alison, Jeff, Gail, Simon w_1, w_2, w_3, w_4
- epoch 2: Richard, Alan, Alison, Jeff, Gail, Simon
- ...
- epoch n... until all training data points are correctly classified
  - $w_1 = 0.25$ $w_2 = 0.1$ $w_3 = 0.2$ $w_4 = 0.1$

≡ 1 iteration over the training set

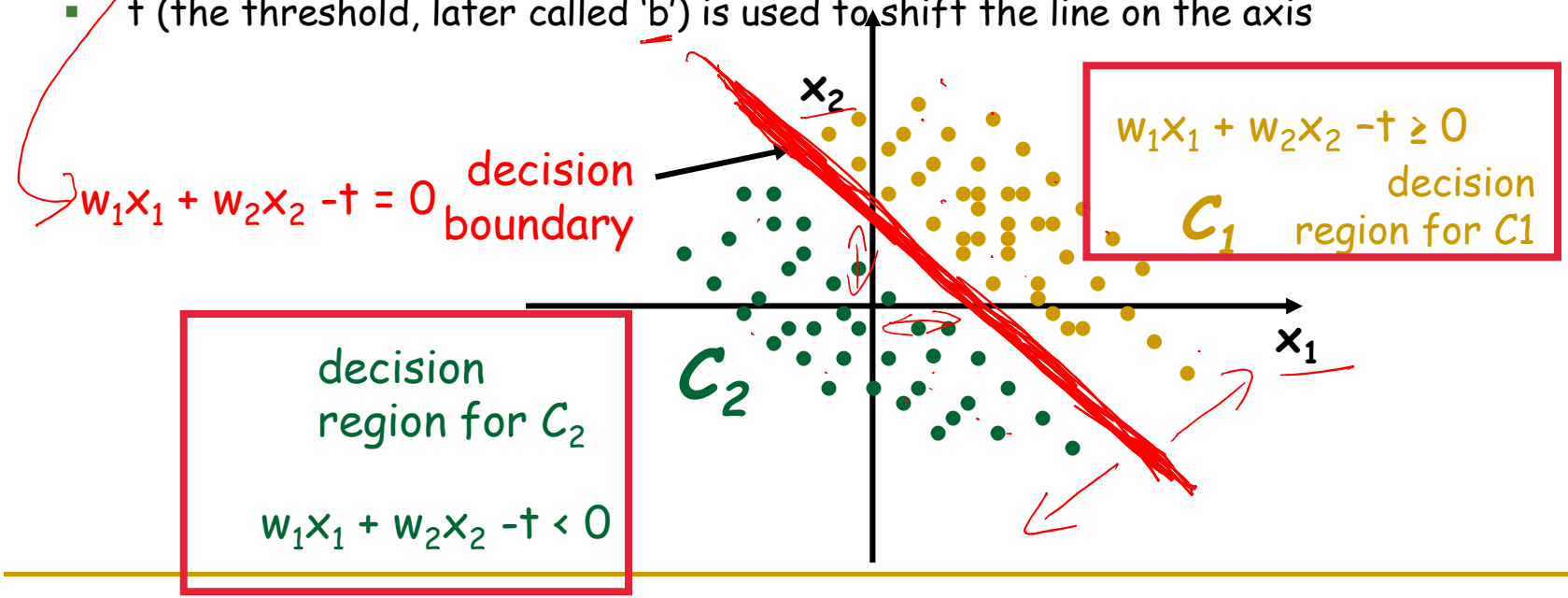
A Simple Example (5)

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
Richard	1	1	0	1	**0**
Alan	1	1	1	0	**1**
Alison	0	0	1	0	**0**
Jeff	0	1	0	1	**0**
Gail	1	0	1	1	**1**
Simon	0	1	1	1	**0**

- Let's check... (w_1= 0.2 w_2= 0.1 w_3= 0.25 w_4= 0.1)
  - Richard: (1×0.2) + (1×0.1) + (0×0.25) + (1×0.1) = 0.4 < 0.55 -> output is 0 ✓
  - Alan: (1×0.2) + (1×0.1) + (1×0.25) + (0×0.1) = 0.55 ≥ 0.55 -> output is 1 ✓
  - Alison: (0×0.2) + (0×0.1) + (1×0.25) + (0×0.1) = 0.25 < 0.55 -> output is 0 ✓
  - Jeff: (0×0.2) + (1×0.1) + (0×0.25) + (1×0.1) = 0.2 < 0.55 -> output is 0 ✓
  - Gail: (1×0.2) + (0×0.1) + (1×0.25) + (1×0.1) = 0.55 ≥ 0.55 -> output is 1 ✓
  - Simon: (0×0.2) + (1×0.1) + (1×0.25) + (1×0.1) = 0.45 < 0.55 -> output is 0 ✓

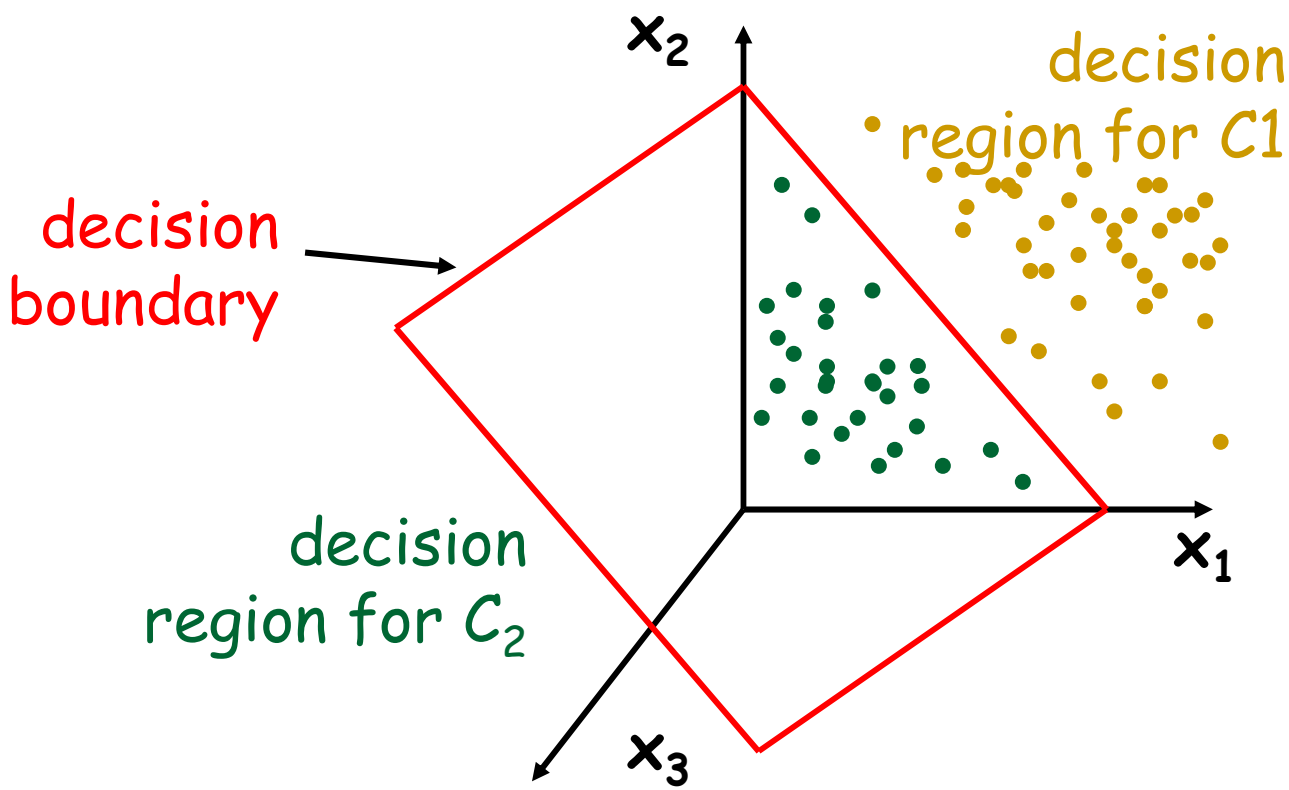
Decision Boundaries of Perceptrons

- So we have just learned the function:
  - If $(0.2x_1 + 0.1x_2 + 0.25x_3 + 0.1x_4 \geq 0.55)$ then 1 otherwise 0
  - If $(0.2x_1 + 0.1x_2 + 0.25x_3 + 0.1x_4 - 0.55 \geq 0)$ then 1 otherwise 0
- Assume we only had 2 features:
  - If $(w_1x_1 + w_2x_2 - t >= 0)$ then 1 otherwise 0
  - The learned function describes a line in the input space
  - This line is used to separate the two classes C1 and C2
  - t (the threshold, later called 'b') is used to shift the line on the axis



Decision Boundaries of Perceptrons

- More generally, with n features, the learned function describes a hyperplane in the input space.

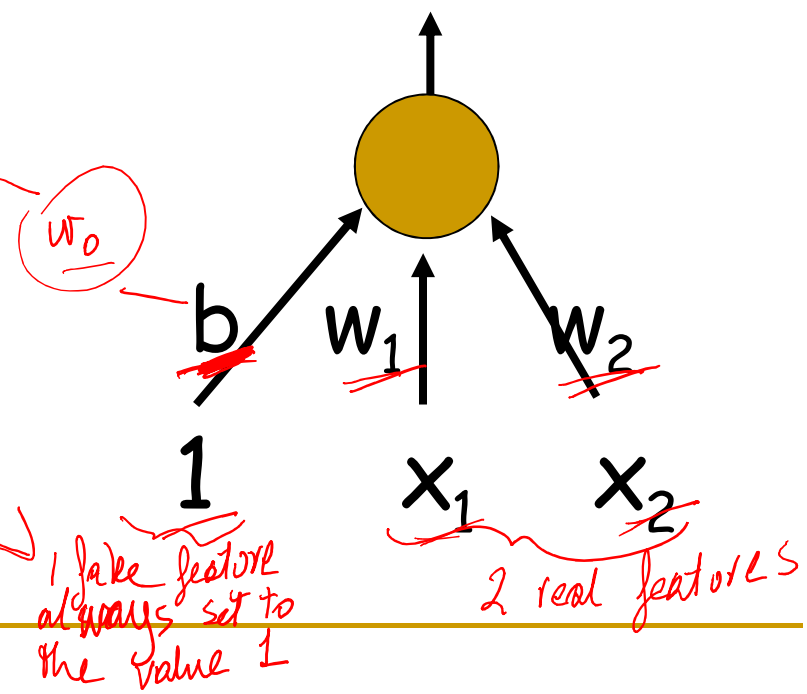


Adding a Bias #1

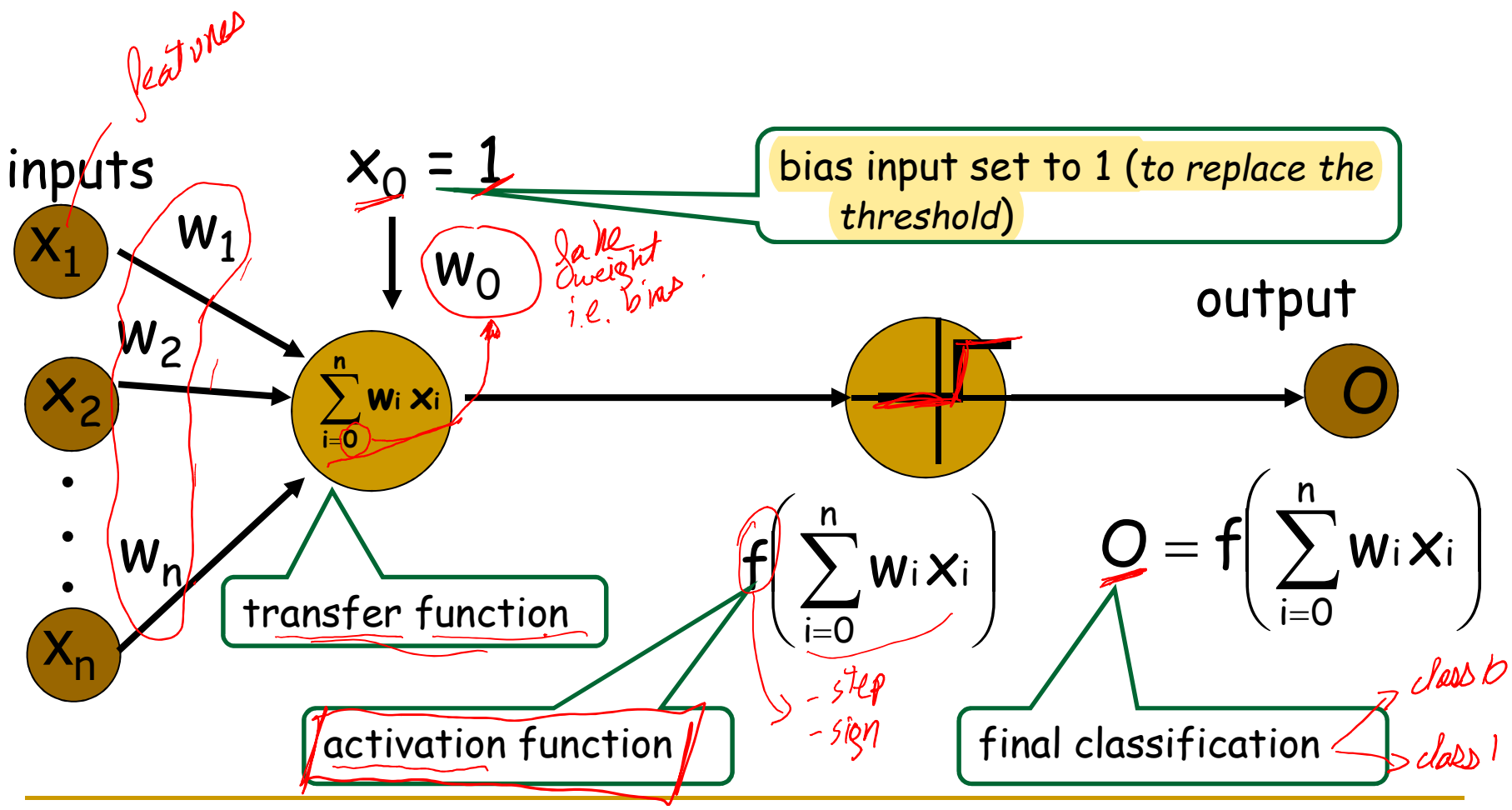
- We can avoid having to figure out the threshold by using a "bias"

$$b + \boxed{\sum_i x_i w_i}$$

- A bias is equivalent to a weight on an extra input feature that always has a value of 1.



Perceptron - More Generally



Common Activation Functions

threshold

- **step**

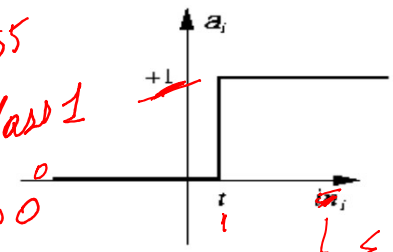
$$0 = \begin{cases} if\left(\sum_{i=1}^{n} w_i x_i\right) \geq t \rightarrow 1 \\ otherwise \rightarrow 0 \end{cases}$$

// class 1

// class 0

e.g. 0.55

// real features



$\sum w_i x_i$

(a) Step function

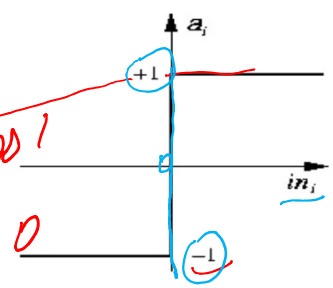
value is not important

- **sign**

$$0 = \begin{cases} if\left(\sum_{i=0}^{n} w_i x_i\right) \geq 0 \rightarrow 1 \\ otherwise \rightarrow -1 \end{cases}$$

// class 1

// class 0



(b) Sign function

merged together into

$$0 = \begin{cases} if\left(\sum_{i=0}^{n} w_i x_i\right) \geq 0) \rightarrow 1 \\ if\left(\sum_{i=0}^{n} w_i x_i\right) = 0) \rightarrow 0 \\ otherwise \rightarrow -1 \end{cases}$$

official sign function

3-classes

used in the perceptron

[Russell & Norvig, 1995]

Learning Rate #2

so far

1. Learning rate can be a constant value (as in the previous example)

$$\Delta w = \eta(T - O)$$

learning rate η = 0.05

Error = target output – actual output

target output

error

1 - 0 = 1

- So:
  - if T=zero and O=1 (i.e. a false positive) -> decrease w by η
  - if T=1 and O=zero (i.e. a false negative) -> increase w by η
  - if T=O (i.e. no error) -> don't change w

delta rule

2. Or, a fraction of the input feature x_i

$$\boxed{\Delta w_i = \eta(T - O)x_i}$$

value of input feature x_i

error

0.1

- So the update is proportional to the value of x
  - if T=zero and O=1 (i.e. a false positive) -> decrease w_i by ηx_i
  - if T=1 and O=zero (i.e. a false negative) -> increase w_i by ηx_i
  - if T=O (i.e. no error) -> don't change w_i
- This is called the delta rule or perceptron learning rule

Perceptron Convergence Theorem

- If a solution with zero error exists
  - i.e. the training data are linearly separable
- The delta rule will find a solution in finite time

Example of the Delta Rule

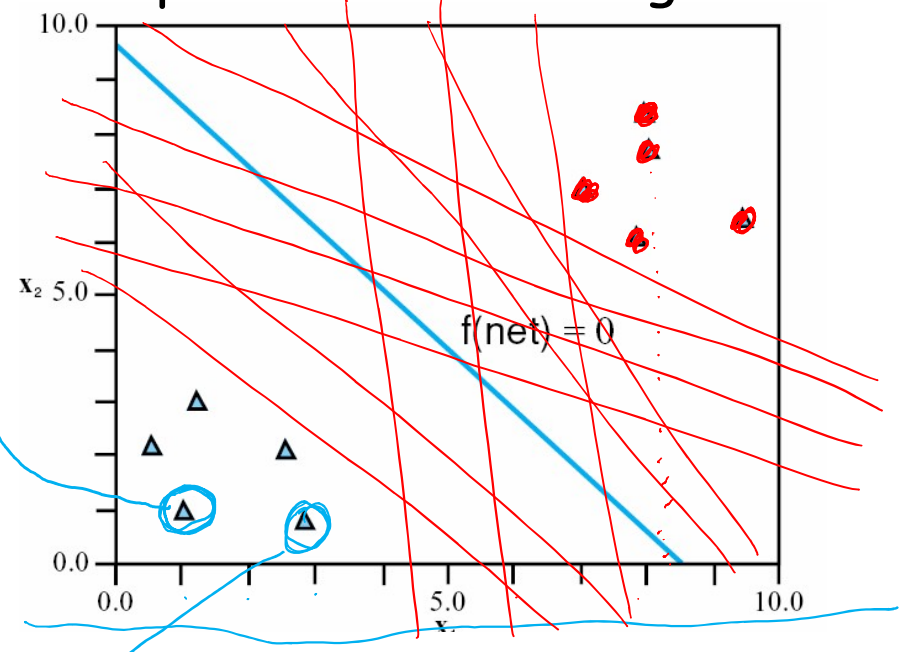
- training data:

x_1	x_2	x_3	Target ~~Output~~
1.0	1.0	1.0	1
9.4	6.4	1.0	−1
2.5	2.1	1.0	1
8.0	7.7	1.0	−1
0.5	2.2	1.0	1
7.9	8.4	1.0	−1
7.0	7.0	1.0	−1
2.8	0.8	1.0	1
1.2	3.0	1.0	1
7.8	6.1	1.0	−1

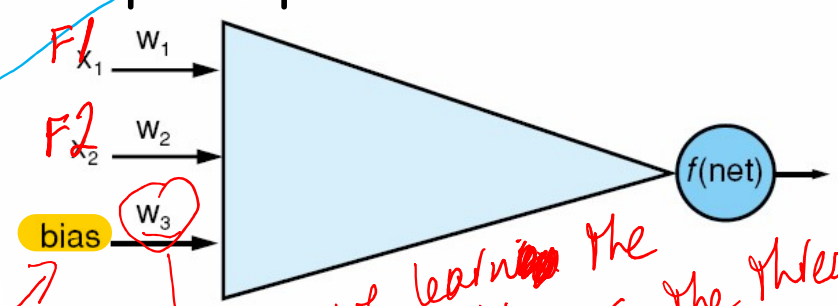
features

F1 F2 F3 bias

- plot of the training data:



- perceptron

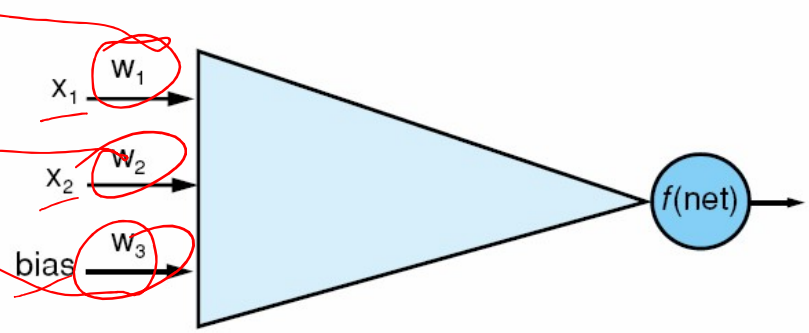


= we learning the bias = the threshold

source: Luger (2005)

Example of the Delta Rule

- assume random initialization
  - w1 = 0.75
  - w2 = 0.5
  - w3 = -0.6



- Assume:
  - sign function (threshold = 0)
  - learning rate η = 0.2 using the delta rule

source: Luger (2005)

Example of the Delta Rule

activation function (sign)

transfer function

$\sum_{i=0}^{n} x_i w_i$

- data #1: f(0.75x1 + 0.5x1 - 0.6x1) = f(0.65) -> 1 ✓
- data #2: f(0.75x9.4 + 0.5x6.4 - 0.6x1) = f(9.65) -> 1 ✗
  - --> error = (-1 - 1) = -2
    - --> w_1 = w_1 -2x0.2x9.4
    - = 0.75 - 3.76 = -3.01 new w_1
    - --> w_2 = w_2 -2x0.2x6.4 = -2.06 new w_2
    - --> w_3 = w_3 -2x0.2x1 = -1.00 new w_3
- data #3: f(-3.01x2.5 -2.06x2.1 -1x1) = f(-12.84) -> -1 ✗
  - --> error = (1 - -1) = 2
    - --> w_1 = -3.01 + 2x0.2x2.5 = -2.01
    - --> w_2 = -2.06 + 2x0.2x2.1 = -1.22
    - --> w_3 = -1.00 + 2x0.2x1 = -0.60
- repeat… over 500 iterations, we converge to:

  w_1 = -1.3 w_2 = -1.1 w_3 = 10.9

x_1	x_2	bias	~~Output~~ Target
1.0	1.0	1	1
9.4	6.4	1	−1
2.5	2.1	1	1
8.0	7.7	1	−1
0.5	2.2	1	1

error

learning rate

Delta formula

output

error T-O 1-1 =0 ⇒ weights do not change

error: T-O -1-1 = -2

T-O = 1-(-1) = 2

source: Luger (2005)

The Perceptron in 1958

1st perceptron

An IBM 704 – a 5-ton computer the size of a room – was fed a series of punch cards. After 50 trials, the computer taught itself to distinguish cards marked on the left from cards marked on the right.

Frank Rosenblatt

https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon

Remember this slide?

History of AI

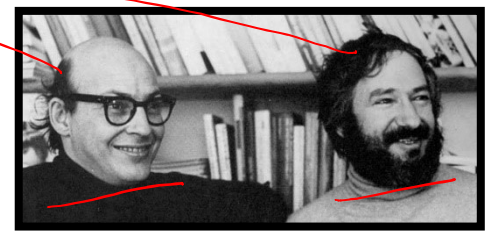
- Reality hits (late 60s - early 70s)
  - 1966: the ALPAC report kills work in machine translation (and NLP in general)
  - People realized that scaling up from micro-worlds (toy-worlds) to reality is not just a manner of faster machines and larger memories…
  - Minsky & Papert's paper on the limits of perceptrons (cannot learn just any function…) kills work in neural networks 1969
  - in 1971, the British government stops funding research in AI due to no significant results
  - it's the first major *AI Winter…*

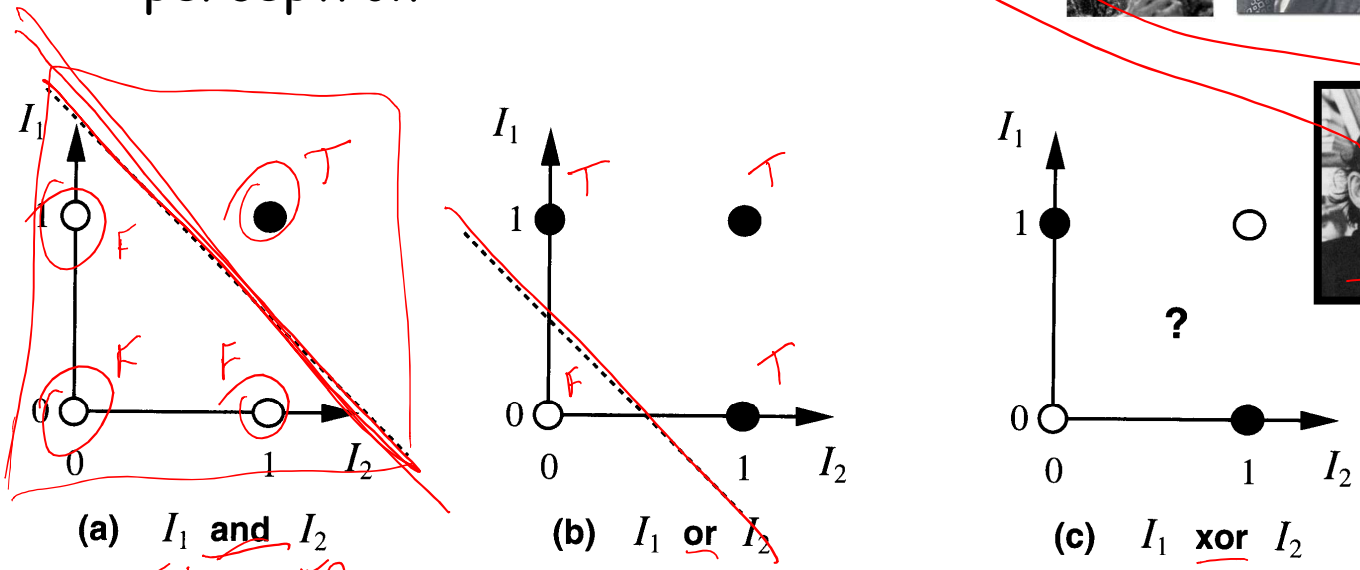
https://www.vectorstock.com/royalty-free-vector/freezing-snowman-vector-689086

32

Limits of the Perceptron

- In 1969, Minsky and Papert showed formally what functions could and could not be represented by perceptrons
- Only linearly separable functions can be represented by a perceptron

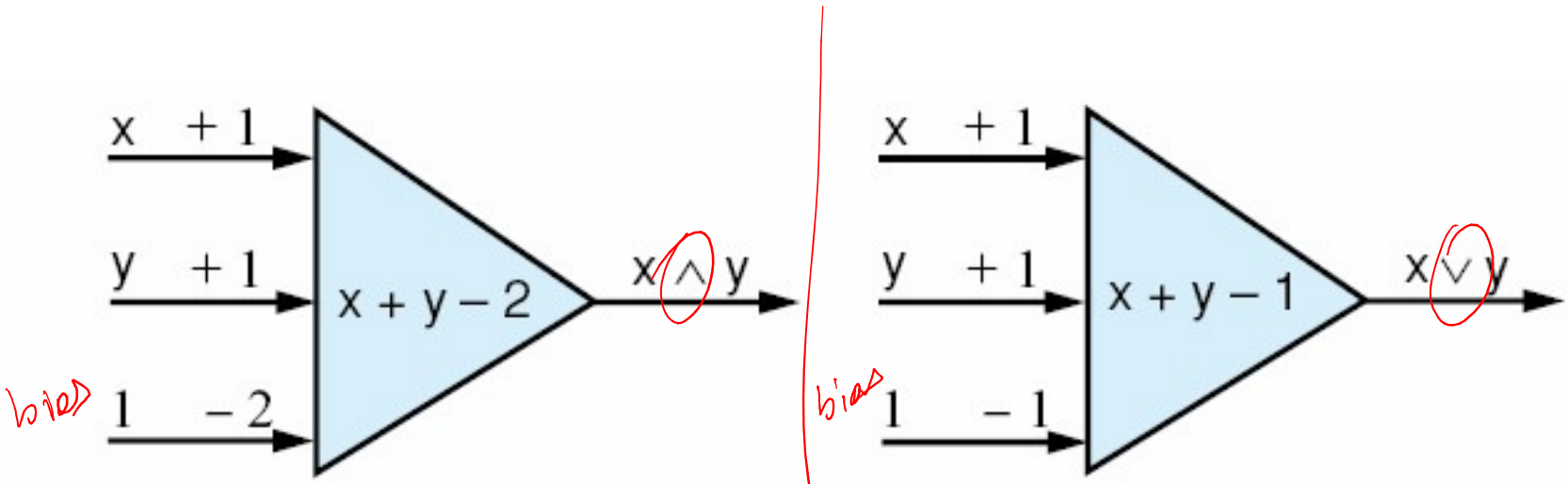




(a) I_1 **and** I_2 (b) I_1 **or** I_2 (c) I_1 **xor** I_2

source: Luger (2005)

AND and OR Perceptrons

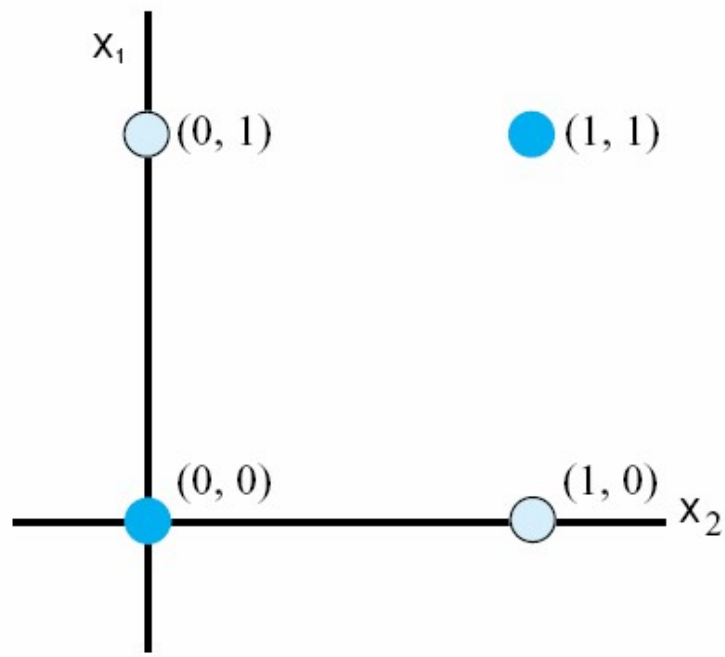


x	y	x + y − 2	Output
1	1	0	1
1	0	−1	−1
0	1	−1	−1
0	0	−2	−1

source: Luger (2005)

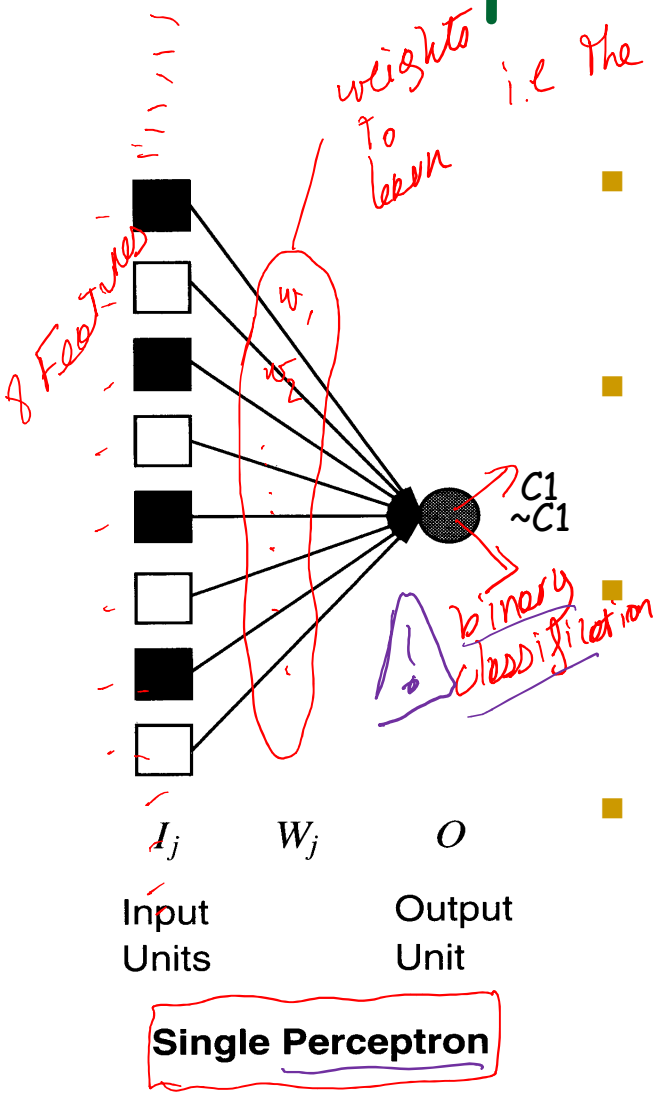
The XOR Function - Visually

- In a 2-dimentional space (2 features for the X)
- No straight line in two-dimensions can separate
  - (0, 1) and (1, 0) from
  - (0, 0) and (1, 1).



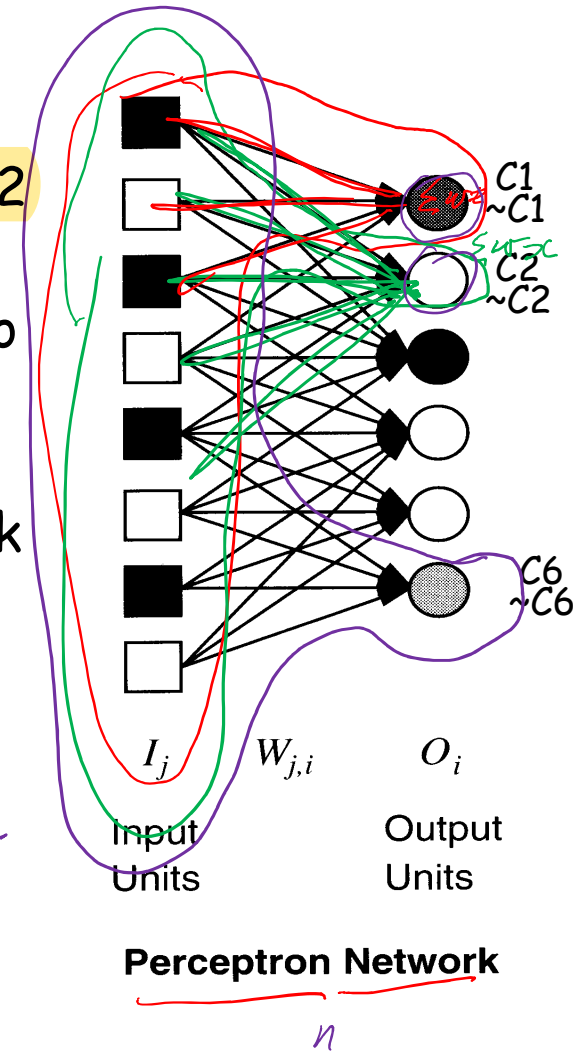
source: Luger (2005)

A Perceptron Network



Single Perceptron

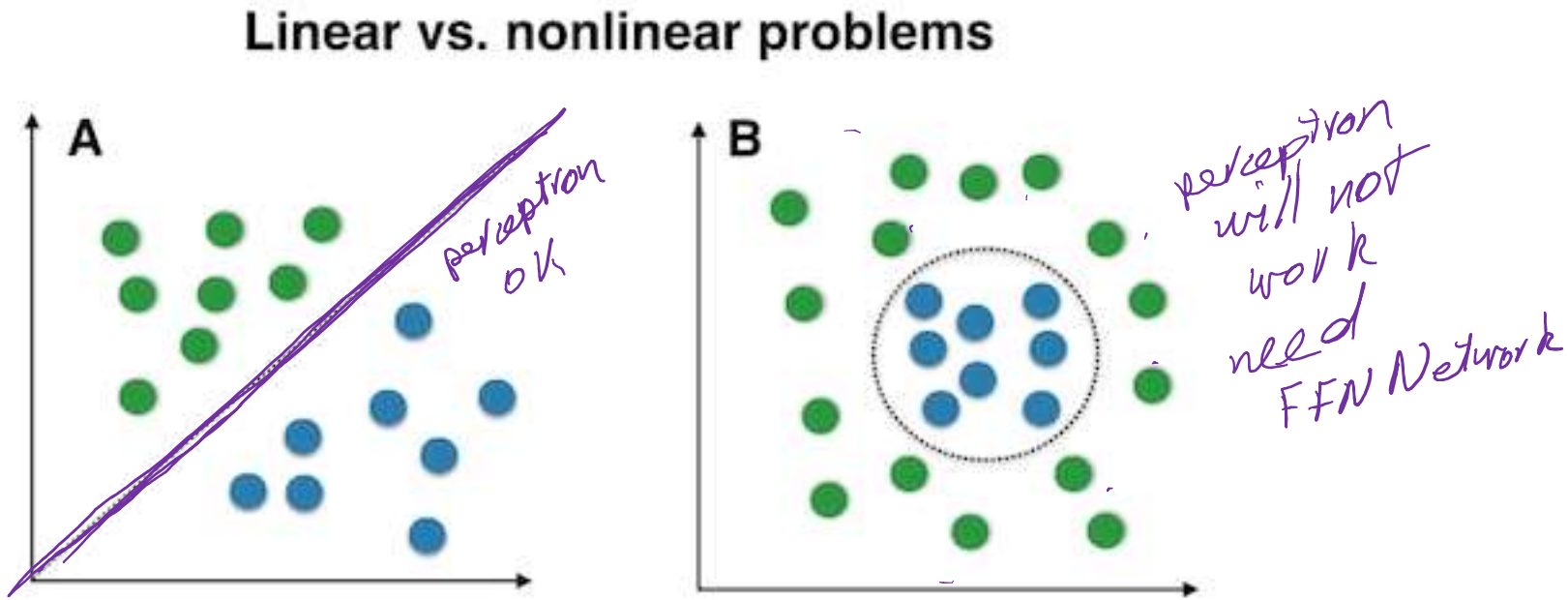
- A perceptron is a binary classifier (i.e. 2 classes)
- if the output needs to learn more than a binary decision
- we can have a network of perceptrons
- Eg: learning to recognize digit --> 10 possible outputs --> need a perceptron network



Perceptron Network

Non-Linearly Separable Functions

- Real-world problems cannot always be represented by linearly-separable functions…



- This caused a decrease in interest in neural networks in the 1970's
1980's

http://sebastianraschka.com/Articles/2014_kernel_pca.html

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks video #8.

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks