

The background features several abstract, colorful shapes: a large purple arrow-like curve on the left, a green curved shape at the top, and a light blue circular outline on the right. Scattered throughout are small yellow triangles pointing in various directions.

Principles of Artificial Intelligence(305450)

Lecture 7: Introduction to Learning

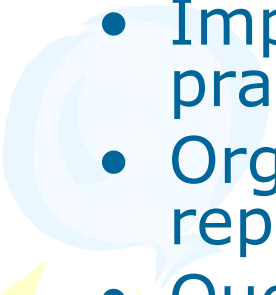
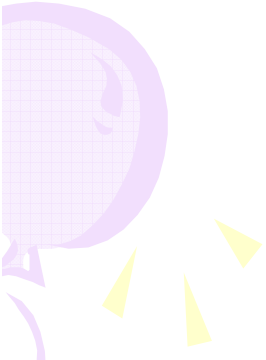


Learning

- It is often hard to articulate the knowledge we need to build AI systems
- Often, we don't even know it!
- Frequently, we can arrange to build systems that learn it themselves



What is Learning?

- Memorizing something
 - Learning facts through observation and exploration
 - Improving motor and/or cognitive skills through practice
 - Organizing new knowledge into general, effective representations
 - Quote from Herb Simon
 - “Learning denotes changes in the system that are adaptive in the sense that they enable the system to do task or tasks drawn from the same population more efficiently and more effectively the next time”
- 
- 

Induction

Piece of bread 1 was nourishing when I ate it.
Piece of bread 2 was nourishing when I ate it.

...

Piece of bread 100 was nourishing when I ate it.

Therefore, all pieces of bread will be nourishing
if I eat them.




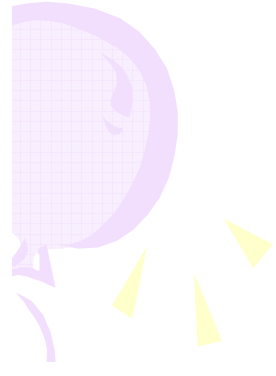
Why is Induction Okay?

It has been argued that we have reason to know the future will resemble the past, because what was the future has constantly become the past, and has always been found to resemble the past, so that we really have experience of the future, namely of times which were formerly future, which we may call past futures. But such an argument really begs the very question at issue. We have experience of past futures, but not of future futures, and the question is: Will future futures resemble past futures?

- If asked why we believe the sun will rise tomorrow, we shall naturally answer, 'Because it has always risen everyday'. We have a form of believe that it will rise in the future, because it has risen in the past.



Kinds of Learning

- **Supervised Learning:** given a set of example input/output pairs, find a rule that does a good job of predicting the output associated with a new input
 - **Clustering:** given a set of examples, but no labeling of them, group the examples into natural clusters
 - **Reinforcement:** an agent interacting with the world makes observations, takes actions and is rewarded or punished; it should learn to choose actions in such a way as to obtain a lot of reward
- 
- 



Learning a Function

- Given a set of examples of input/output pairs, find a function that does a good job of expressing the relationship
 - Pronunciation: function from letters to sounds
 - Throw a ball: function from target locations to joint torques
 - Read handwritten characters: function from collections of image pixels to letters
 - Diagnose diseases: function from lab test results to disease categories



Aspects of Function Learning

- memory

- averaging

- generalization

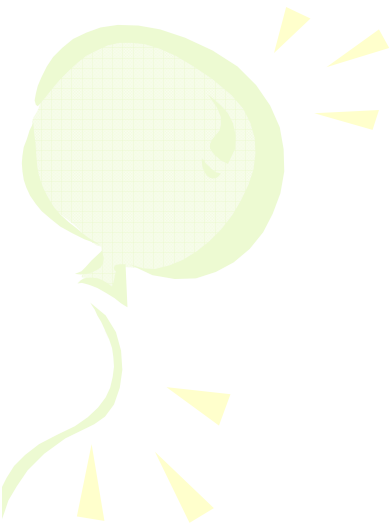




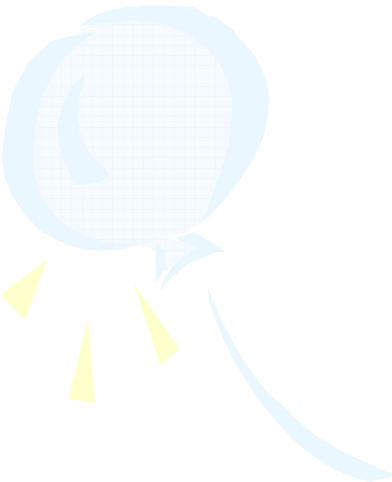
Example problem

- When to drive the car? Depends on:
 - temperature
 - expected precipitation
 - day of the week
 - whether she needs to shop on the way home
 - what she's wearing

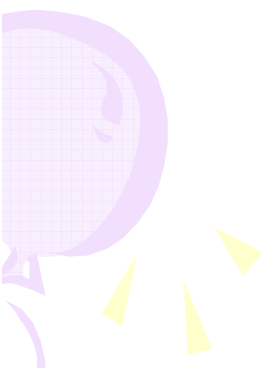
Memory



temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
19	snow	mon	yes	casual	drive
65	none	tues	no	casual	walk



temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
19	snow	mon	yes	casual	drive
65	none	tues	no	casual	walk
19	snow	mon	yes	casual	



temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
19	snow	mon	yes	casual	drive
65	none	tues	no	casual	walk
19	snow	mon	yes	casual	drive



Averaging

- Dealing with noise in the data

temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	drive
80	none	sat	no	casual	drive
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk

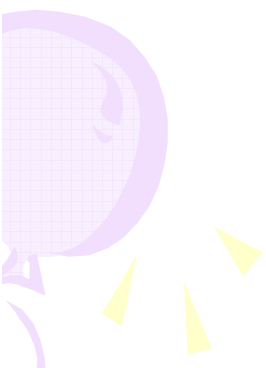


Averaging

- Dealing with noise in the data



temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	drive
80	none	sat	no	casual	drive
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	



A decorative graphic on the left side of the slide features three balloons in green, blue, and purple, each with yellow triangular rays emanating from it. The balloons are connected by thin, curved lines.

Averaging

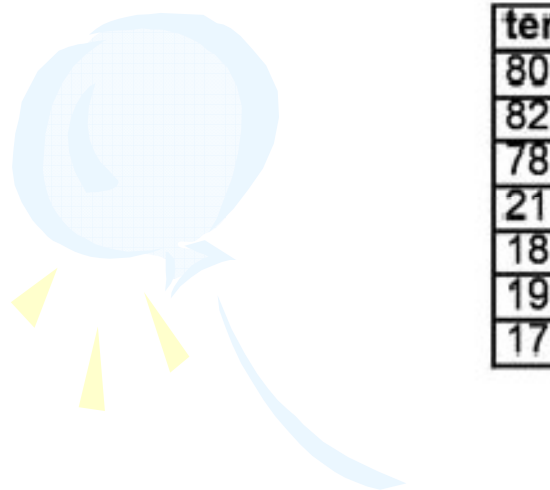
- Dealing with noise in the data

temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	drive
80	none	sat	no	casual	drive
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk
80	none	sat	no	casual	walk



Sensor noise

- Dealing with noise in the data



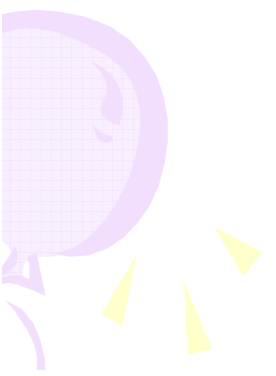
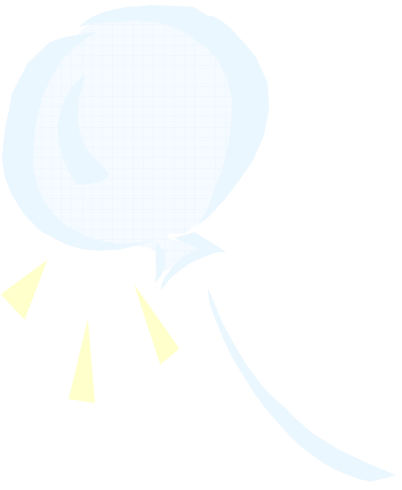
temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
82	none	sat	no	casual	walk
78	none	sat	no	casual	walk
21	none	sat	no	casual	drive
18	none	sat	no	casual	drive
19	none	sat	no	formal	drive
17	none	sat	no	casual	drive





Sensor noise

- Dealing with noise in the data



temp	precip	day	shop	clothes	
80	none	sat	no	casual	walk
82	none	sat	no	casual	walk
78	none	sat	no	casual	walk
21	none	sat	no	casual	drive
18	none	sat	no	casual	drive
19	none	sat	no	formal	drive
17	none	sat	no	casual	drive
20	none	sat	no	casual	drive



Generalization

- Dealing with previously unseen cases

temp	precip	day	shop	clothes	
71	none	fri	yes	formal	drive
36	none	sun	yes	casual	walk
62	rain	weds	no	casual	walk
93	none	mon	no	casual	drive
55	none	sat	no	formal	drive
80	none	sat	no	casual	walk
19	snow	mon	yes	casual	drive
65	none	tues	no	casual	walk

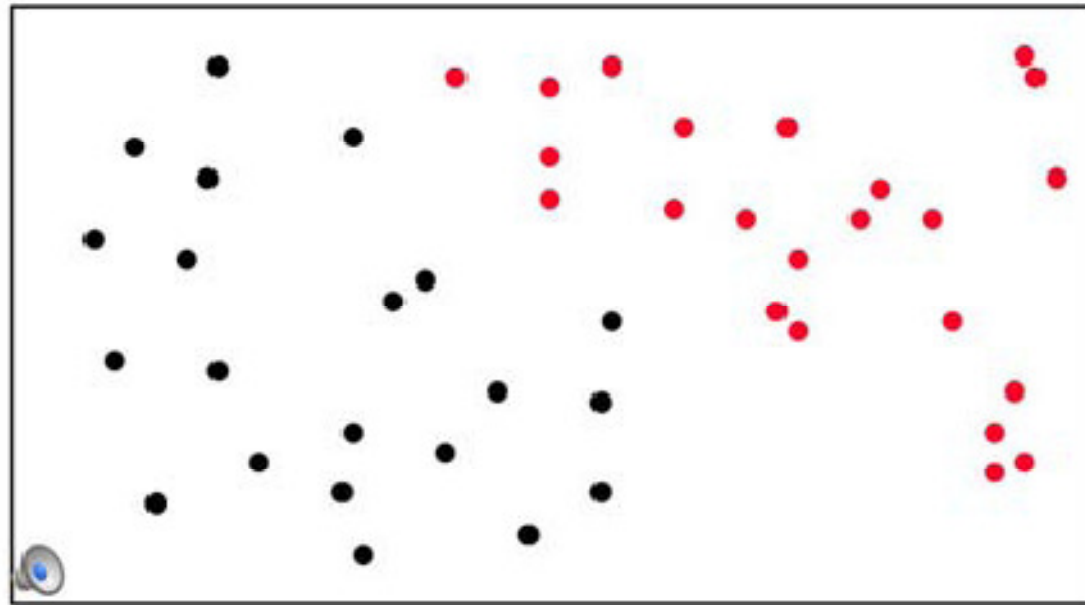


Generalization

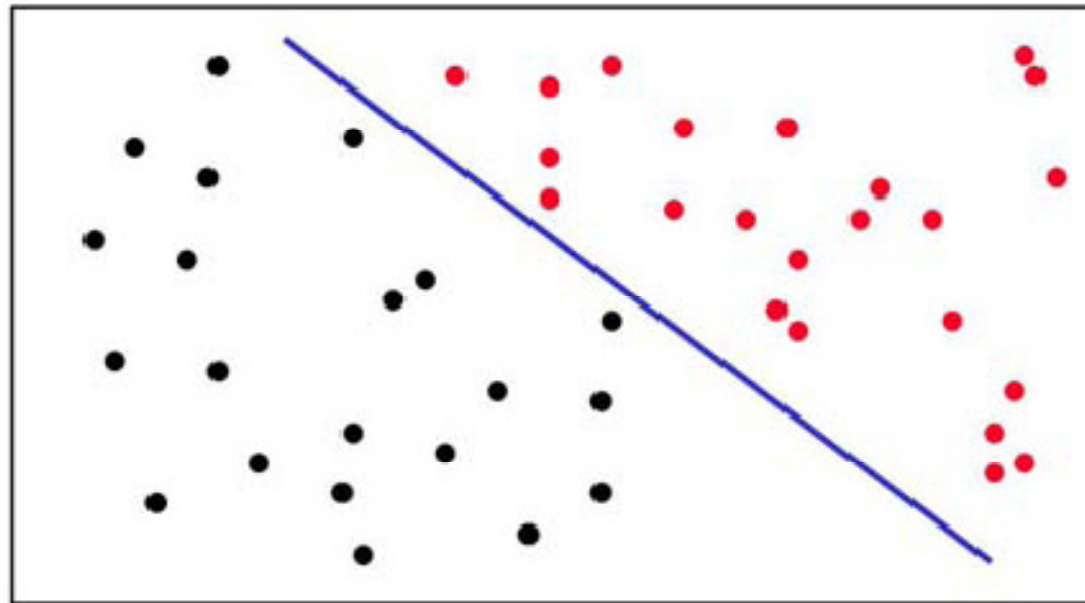
- Dealing with previously unseen cases

temp	precip	day	shop	clothes	
71	none	fri	yes	formal	drive
36	none	sun	yes	casual	walk
62	rain	weds	no	casual	walk
93	none	mon	no	casual	drive
55	none	sat	no	formal	drive
80	none	sat	no	casual	walk
19	snow	mon	yes	casual	drive
65	none	tues	no	casual	walk
58	rain	mon	no	casual	

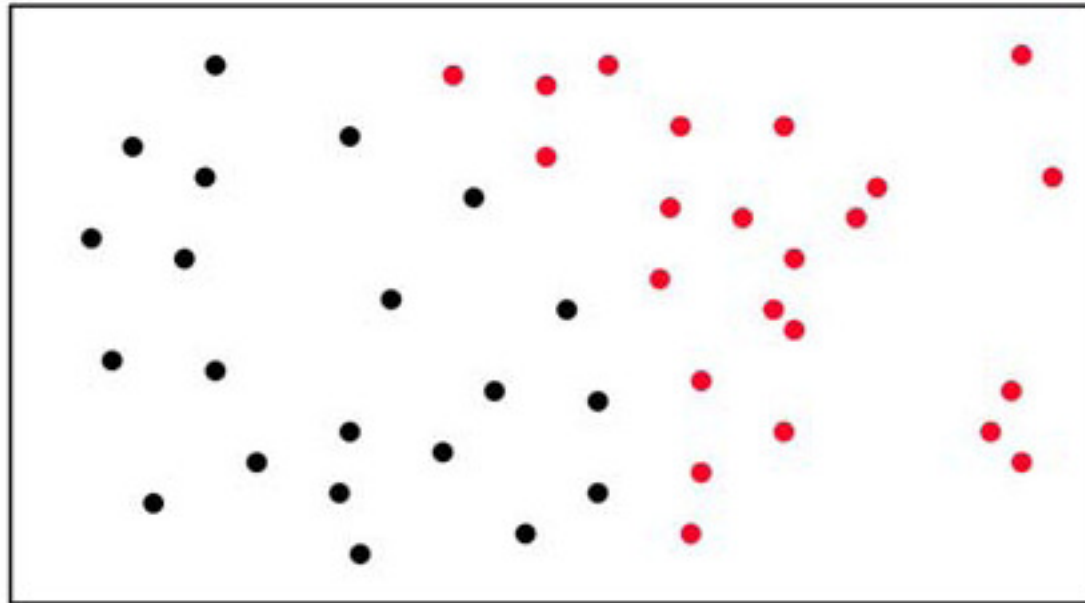
The Red and the Black



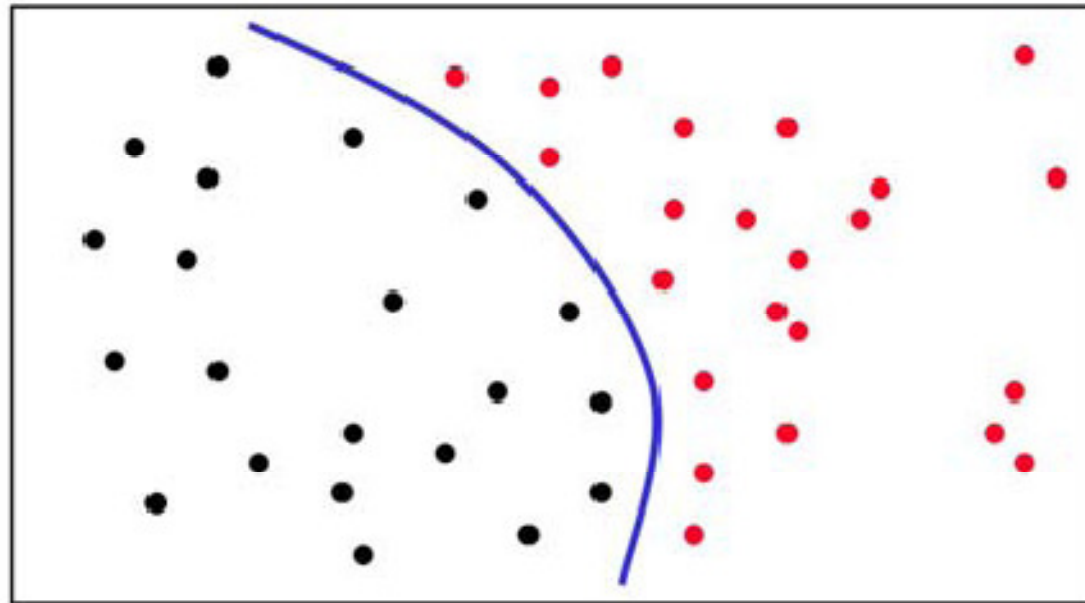
What's the right hypothesis



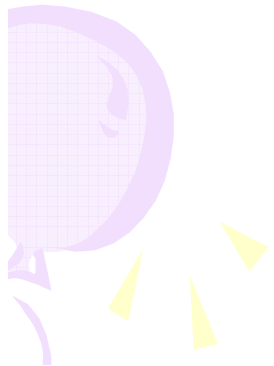
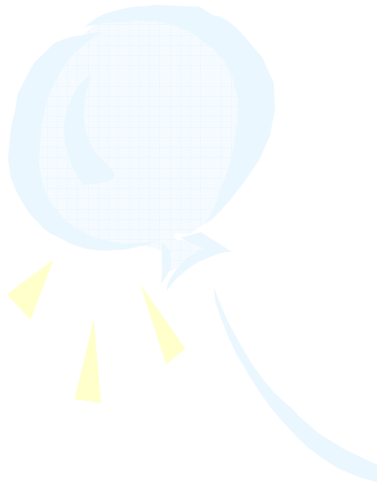
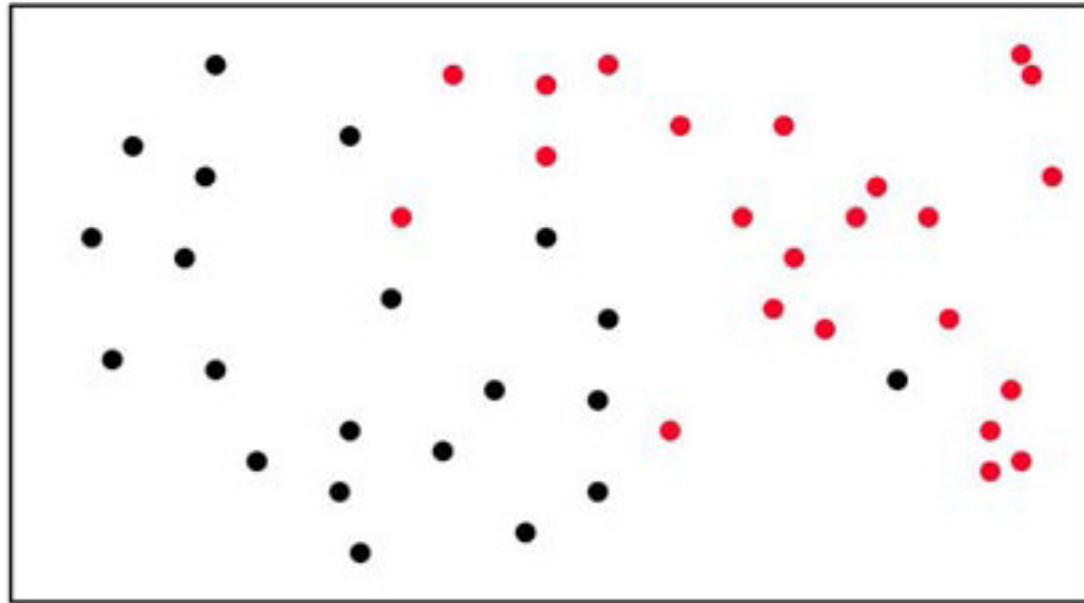
Now what's the right hypothesis



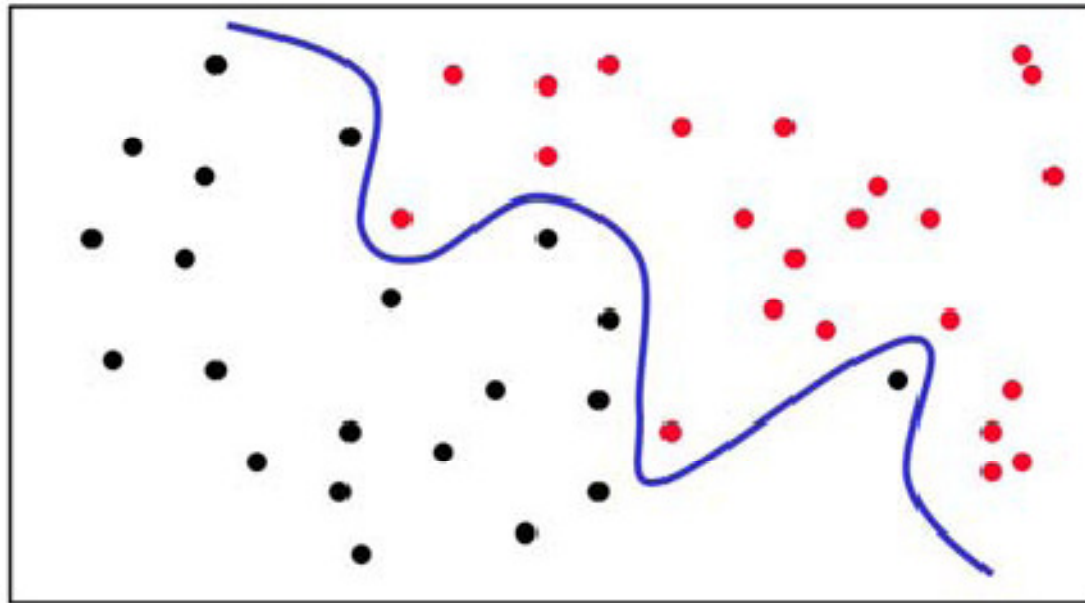
Now what's the right hypothesis



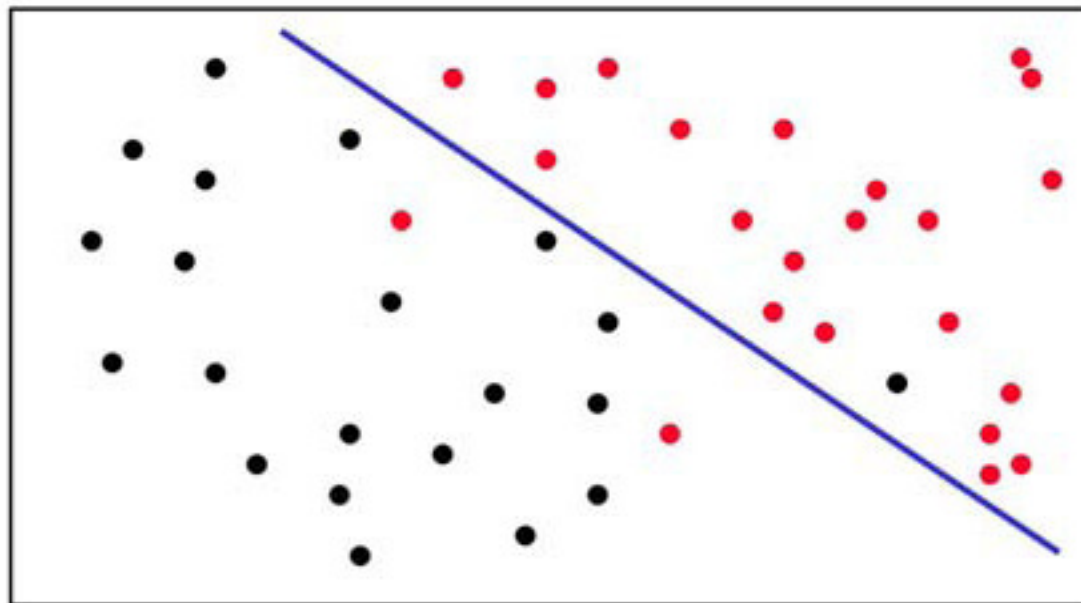
How about now?



How about now? Answer 1



How about now? Answer 2



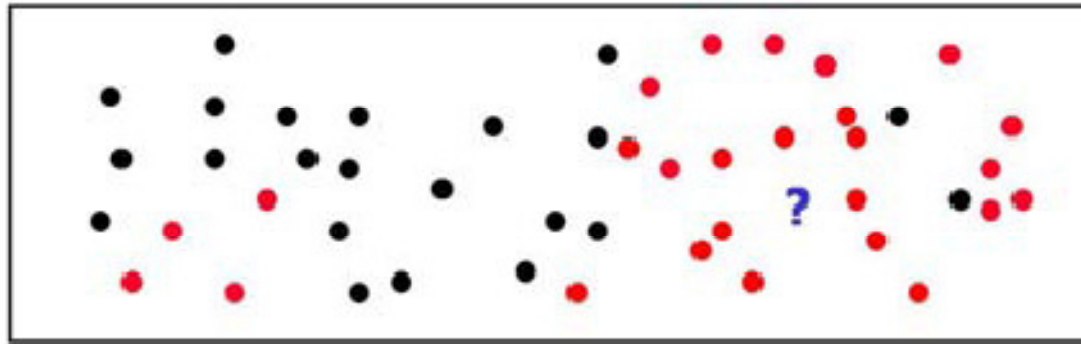


Variety of Learning Methods

- Learning methods differ in terms of:
 - the form of the hypothesis
 - the way the computer finds a hypothesis given the data

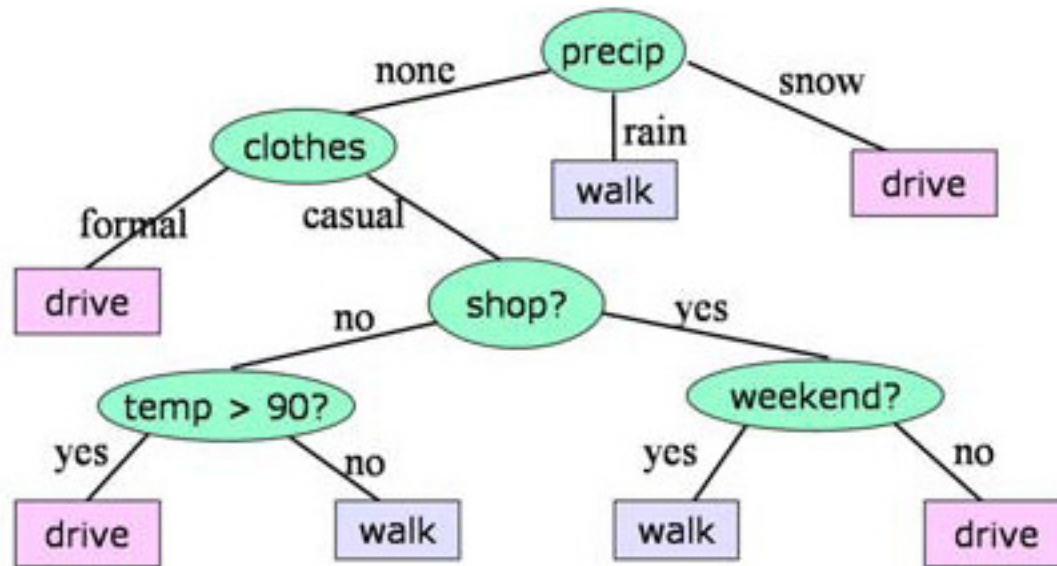
Nearest Neighbor

- Remember all your data
- When someone asks a question,
 - Find the nearest old data point,
 - Return the answer associated with it



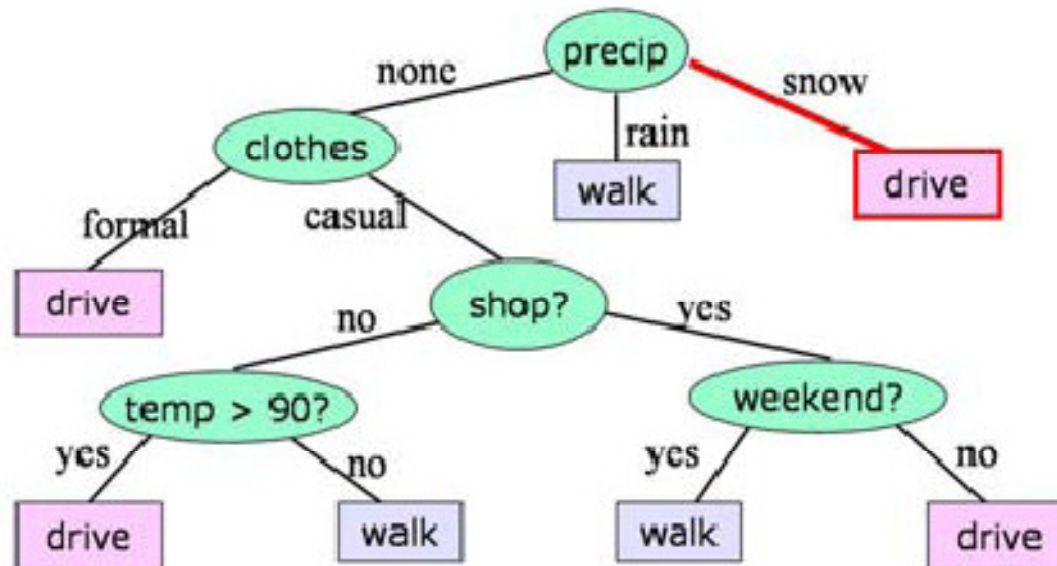
Decision Trees

- Use all the data to build a tree of questions with answer at leaves



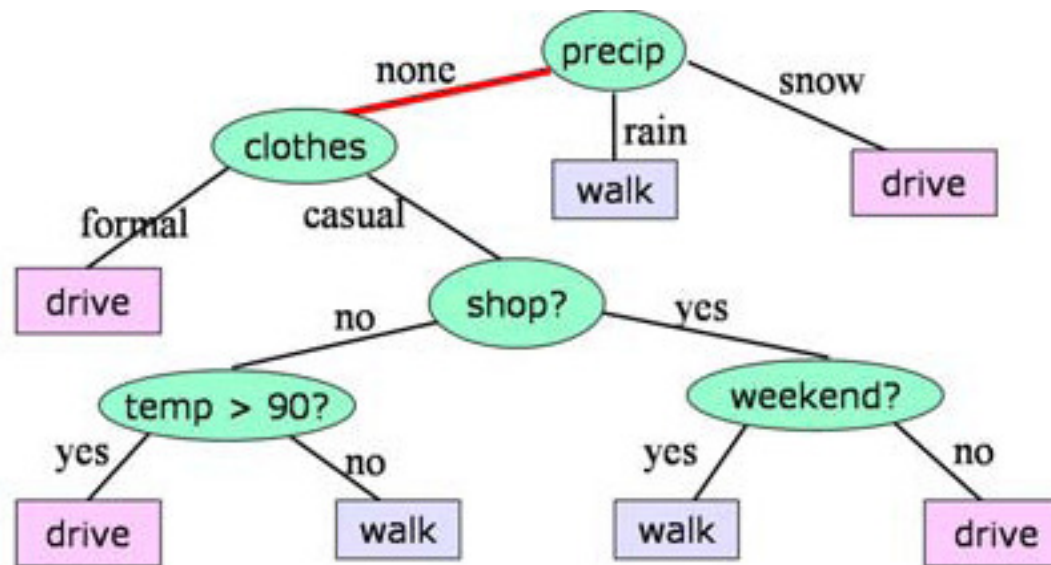
Decision Trees

- Use all the data to build a tree of questions with answer at leaves



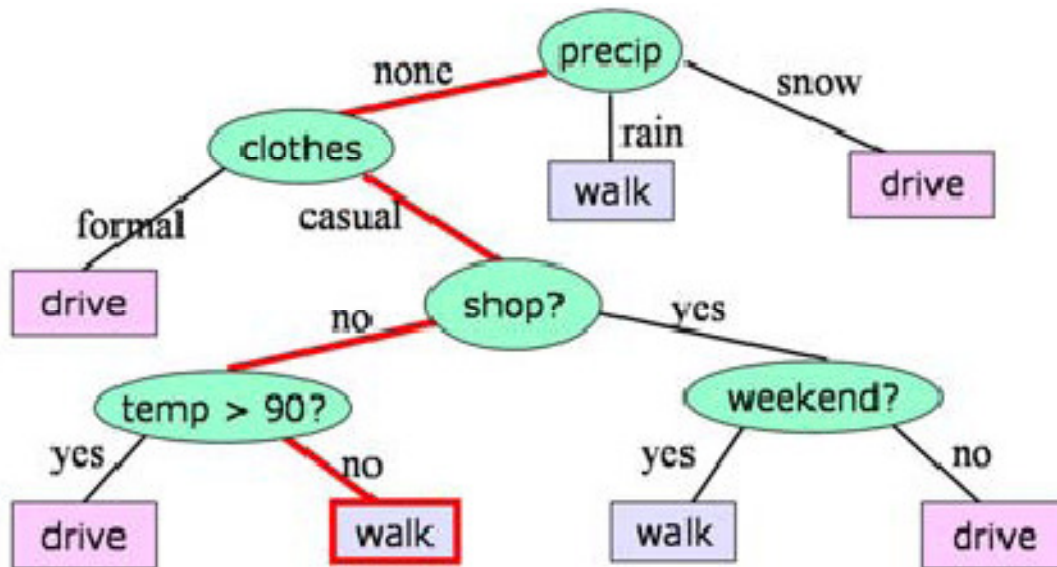
Decision Trees

- Use all the data to build a tree of questions with answer at leaves



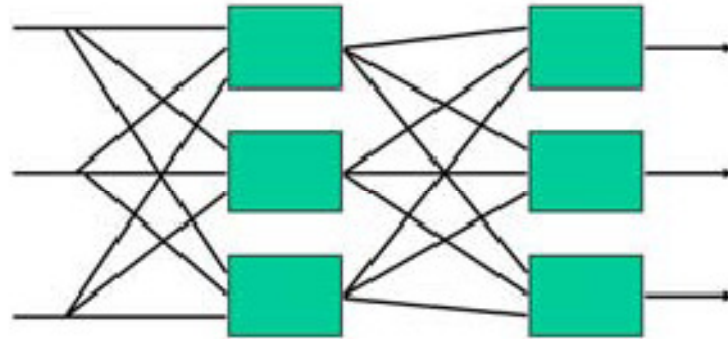
Decision Trees

- Use all the data to build a tree of questions with answer at leaves



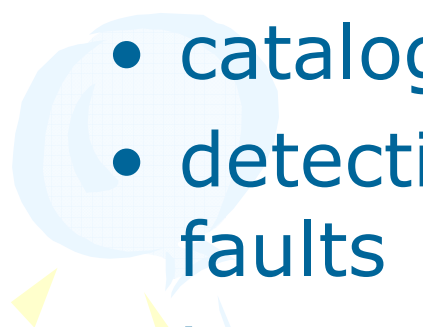

Neural Networks

- Represent hypotheses as combinations of simple computations
- Neurophysiologically plausible (sort of)





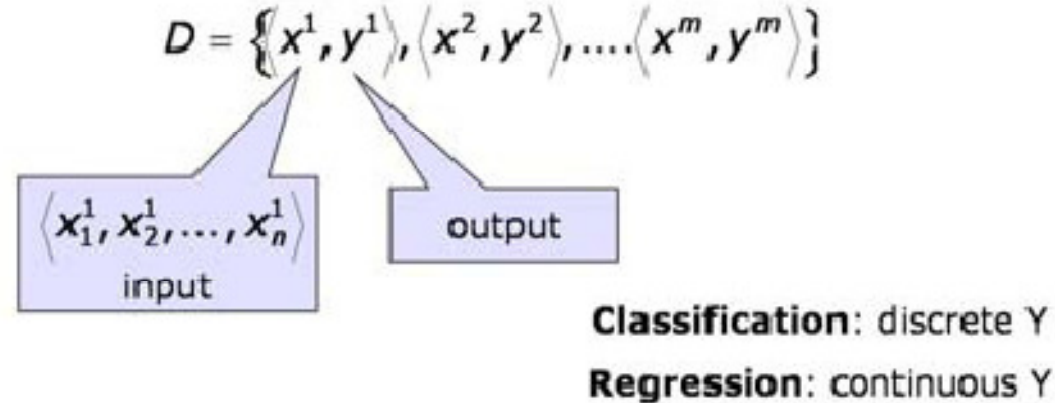
Machine Learning Successes

- assessing loan credit check
 - detecting credit fraud
 - cataloging astronomical images
 - detecting and diagnosing manufacturing faults
 - helping NBA coaches analyze performance
 - personalizing news and web searches
 - steering an autonomous car across the US
- 
- 



Supervised Learning

- Given data (training set)



- 
- Goal: find a hypothesis h in hypothesis class H that does a good job of mapping x to y



Best Hypothesis

- Hypothesis should:
 - do a good job of describing the data



– not too complex



Best Hypothesis

- Hypothesis should:

- do a good job of describing the data

- ideally: $h(x^i) = y^i$

- number of errors: $E(h, D)$

- not too complex

- measure: $C(h)$





Best Hypothesis

- Hypothesis should:

- do a good job of describing the data

- ideally: $h(x^i) = y^i$

- number of errors: $E(h, D)$

- not too complex

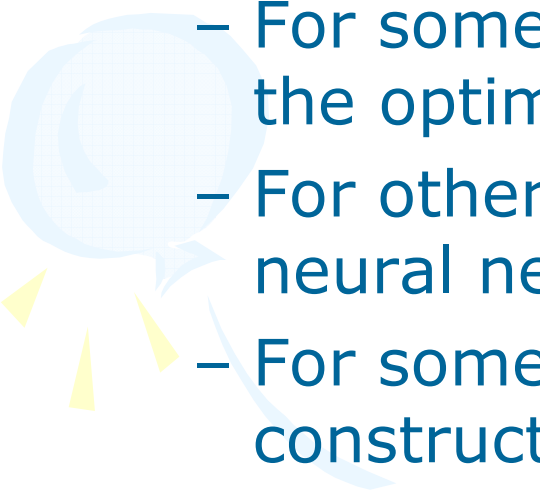
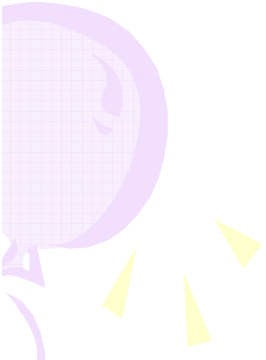
- measure: $C(h)$

Minimize $E(h, D) + \alpha C(h)$

trade-off

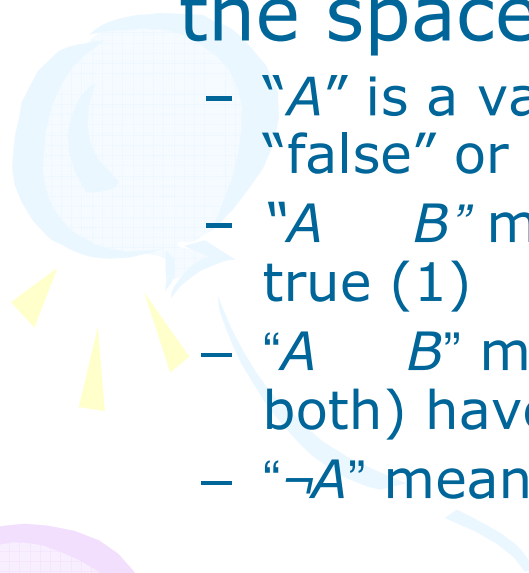
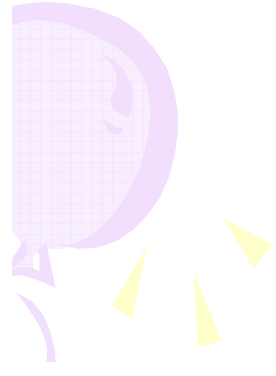


Learning as Search

- How can we find the hypothesis with the lowest value of $E(h, D) + \alpha C(h)$ Search!
 - For some hypothesis classes we can calculate the optimal h directly! (linear separators)
 - For others, do local search (gradient descent in neural networks)
 - For some structured hypothesis spaces, construct one greedily
- 
- 



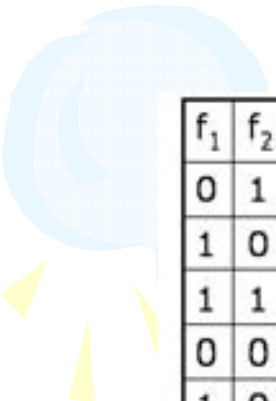
Propositional Logic

- Next sections use the language of propositional logic to specify hypothesis in the space of vectors of binary variables
 - “ A ” is a variable that can take on values “true” and “false” or 1 and 0
 - “ $A \square B$ ” means that variable A and B both have to be true (1)
 - “ $A \sqcup B$ ” means that either variable A or variable B (or both) have to be true (1)
 - “ $\neg A$ ” means that variable A has to be false (0)
- 
- 

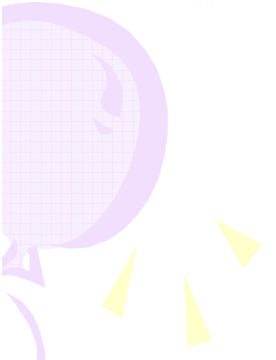


Learning Conjunctions

- Boolean features and output
- H = conjunctions of features



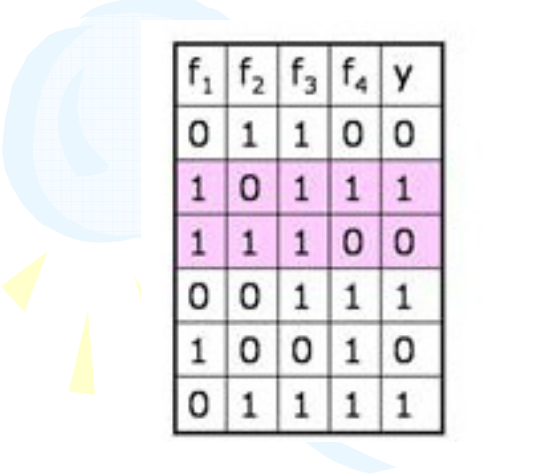
f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1





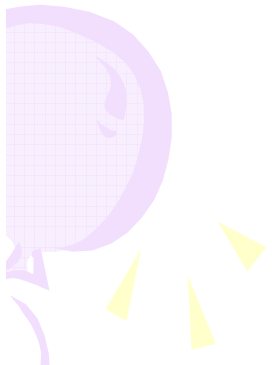
Learning Conjunctions

- Boolean features and output
- $H =$ conjunctions of features



f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

$$h = f_1 \wedge f_3$$



Learning Conjunctions

- Boolean features and output
- $H =$ conjunctions of features

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

$$h = f_1 \wedge f_3$$

$$E(h, D) = 3$$



Learning Conjunctions

- Boolean features and output
- H = conjunctions of features

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

$$h = f_1 \wedge f_3$$


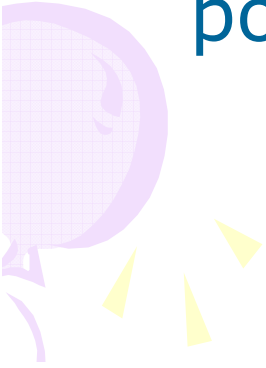
$$E(h, D) = 3$$

$$C(h) = 2$$

- Set alpha so we're looking for smallest h with 0

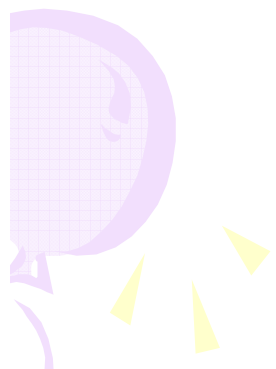
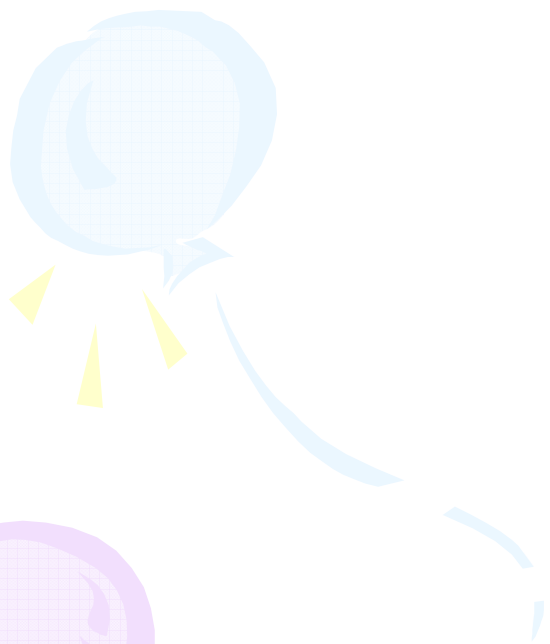
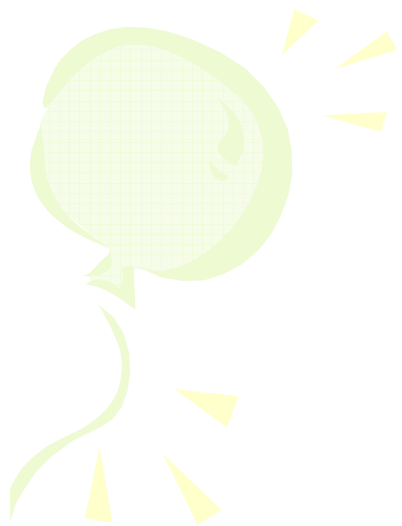


Algorithm

- Could search in hypothesis space using tools we've already studied
 - Instead, be greedy!
 - Start with $h = \text{True}$
 - All errors are on negative examples
 - On each step, add conjunct that out most new negatives (without excluding positives)
- 
- 

Pseudo-code

```
N = negative examples in D  
h = True  
Loop until N is empty
```





Pseudo-code

`N = negative examples in D`

`h = True`

`Loop until N is empty`

`For every feature j that does not have value 0 on
 any positive examples`

`$n_j :=$ number of examples in N
 for which $f_j = 0$`

`$j^* := j$ for which n_j is maximized`

`$h := h \wedge f_{j^*}$.`

`$N := N -$ examples in N for which $f_{j^*} = 0$`

`If no such feature found, fail`

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $N = \{x^1, x^3, x^5\}$, $h = \text{True}$

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $N = \{x^1, x^3, x^5\}$, $h = \text{True}$
- $n_3 = 1$, $n_4 = 2$

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $N = \{x^1, x^3, x^5\}$, $h = \text{True}$
- $n_3 = 1$, $n_4 = 2$
- $N = \{x^5\}$, $h = f_4$

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $N = \{x^1, x^3, x^5\}$, $h = \text{True}$
- $n_3 = 1$, $n_4 = 2$
- $N = \{x^5\}$, $h = f_4$
- $n_3 = 1$, $n_4 = 0$

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $N = \{x^1, x^3, x^5\}$, $h = \text{True}$
- $n_3 = 1$, $n_4 = 2$
- $N = \{x^5\}$, $h = f_4$
- $n_3 = 1$, $n_4 = 0$
- $N = \{\}$, $h = f_4 \wedge f_3$



A Harder Problem

- We made one negative into a positive

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1



A Harder Problem

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- We made one negative into a positive
- Only usable feature is f_3
- Can't add any more features to h
- We're stuck



A Harder Problem

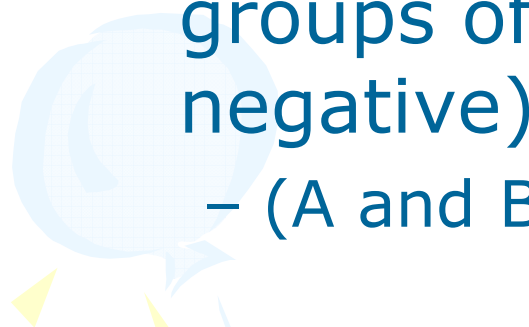
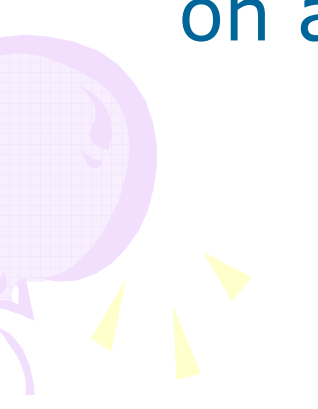
f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- We made one negative into a positive
- Only usable feature is f_3
- Can't add any more features to h
- We're stuck

- Best we can do when H is conjunctions
- Live with error or change H



Disjunctive Normal Form

- Like the opposite conjunctive normal form (but, for now, it's like finding multiple groups of positive examples within the negative)
 - (A and B and C) or (D and E and F)
 - Think of each disjunction as narrowing in on a subset of the positive examples
- 
- 

Disjunctive Normal Form

- Like the opposite conjunctive normal form (but, for now, it's like finding multiple groups of positive examples within the negative)
 - (A and B and C) or (D and E and F)
- Think of each disjunction as narrowing in on a subset of the positive examples

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

Disjunctive Normal Form

- Like the opposite conjunctive normal form (but, for now, it's like finding multiple groups of positive examples within the negative)
 - (A and B and C) or (D and E and F)
- Think of each disjunction as narrowing in on a subset of the positive examples

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

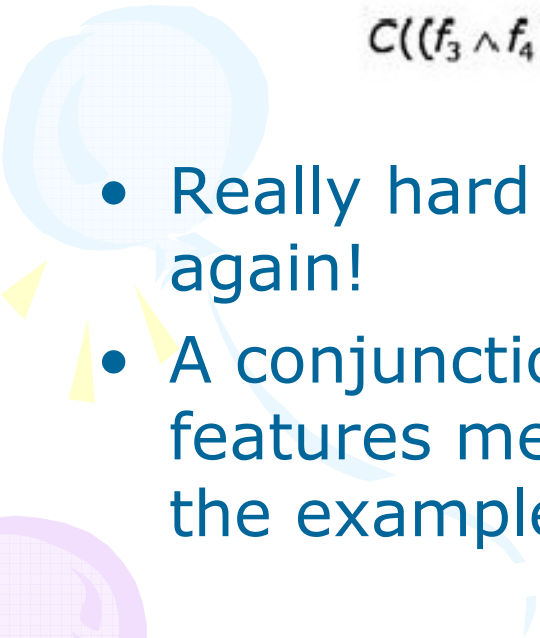
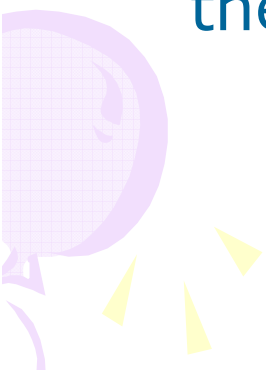
$$(f_3 \wedge f_4) \vee (f_1 \wedge f_3)$$



Learning DNF

- Let H be DNF expressions
- $C(h)$: number of mentions of features

$$C((f_3 \wedge f_4) \vee (f_1 \wedge f_2)) = 4$$

- 
- Really hard to search this space, so be greedy again!
 - A conjunction **covers** an example if all of the features mentioned in the conjunction are true in the example
- 



Algorithm

```
P = set of all positive examples
h = False
Loop until P is empty
  r = True
  N = set of all negative examples
  Loop until N is empty
    If all features are in r, fail
    Else, select a feature  $f_j$  to add to r
       $r = r \wedge f_j$ 
       $N = N - \text{examples in } n \text{ for which } f_j = 0$ 
  h := h  $\vee$  r
  Covered := examples in P covered by r
  If Covered is empty, fail
  Else P := P - Covered
end
```



Choosing a Feature

Heuristic: $v_j = \frac{n_j^+}{\max(n_j^-, 0.001)}$

n_j^+ = # not yet covered positive examples
covered by $r \wedge f_j$

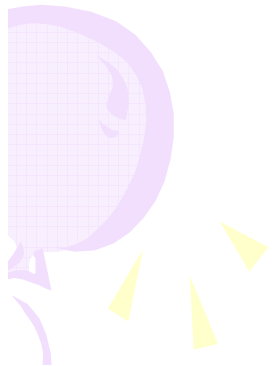
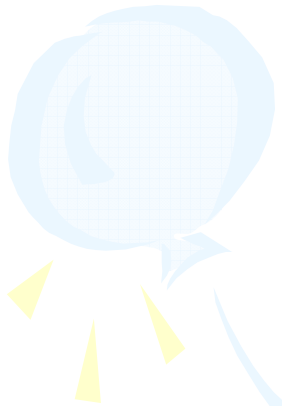
n_j^- = # not yet ruled out negative examples
covered by $r \wedge f_j$

Choose feature with largest value of V_j

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

• $h = \text{False}$, $P = \{x^2, x^3, x^4, x^6\}$



Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $h = \text{False}$, $P = \{x^2, x^3, x^4, x^6\}$
 - $r = \text{True}$, $N = \{x^1, x^5\}$
 - $v_1=2/1, v_2=2/1, v_3=4/1, v_4=3/1$
 - $r = f_3$, $N = \{x^1\}$
 - $v_1=2/0, v_2=2/1, v_4=3/0$
 - $r = f_3 \wedge f_4$, $N = \{\}$
- $h = f_3 \wedge f_4$, $P = \{x^3\}$

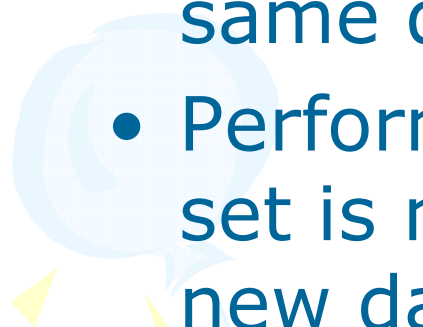

Simulation

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	1

- $h = \text{False}$, $P = \{x^2, x^3, x^4, x^6\}$
 - $r = \text{True}$, $N = \{x^1, x^5\}$
 - $v_1=2/1, v_2=2/1, v_3=4/1, v_4=3/1$
 - $r = f_3$, $N = \{x^1\}$
 - $v_1=2/0, v_2=2/1, v_4=3/0$
 - $r = f_3 \wedge f_4$, $N = \{\}$
- $h = f_3 \wedge f_4$, $P = \{x^3\}$
 - $r = \text{True}$, $N = \{x^1, x^5\}$
 - $v_1=1/1, v_2=1/1, v_3=1/1, v_4=0/1$
 - $r = f_1$, $N = \{x^1\}$
 - $v_2=1/0, v_3=1/0, v_4=0/1$
 - $r = f_1 \wedge f_2$, $N = \{\}$
- $h = (f_3 \wedge f_4) \vee (f_1 \wedge f_2)$, $P = \{\}$

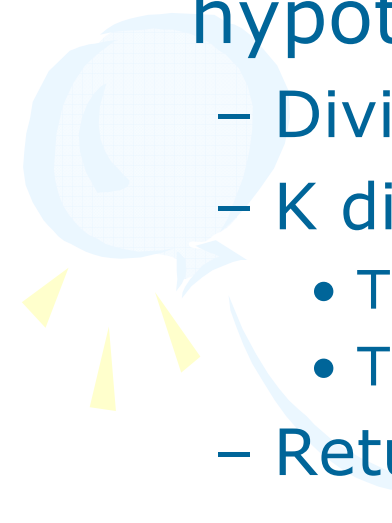
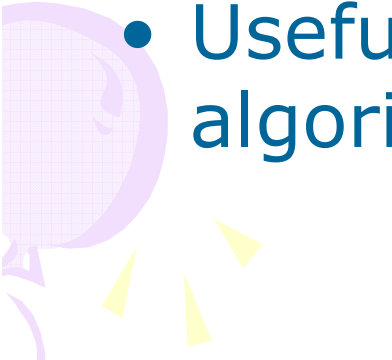


How well does it work?

- We'd like to know how well our h will perform on new data (drawn from the same distribution as the training data)
 - Performance of hypothesis on the training set is not indicative of its performance on new data
 - Save some data as a test set; performance on h is a reasonable estimate of performance on new data
- 
- 

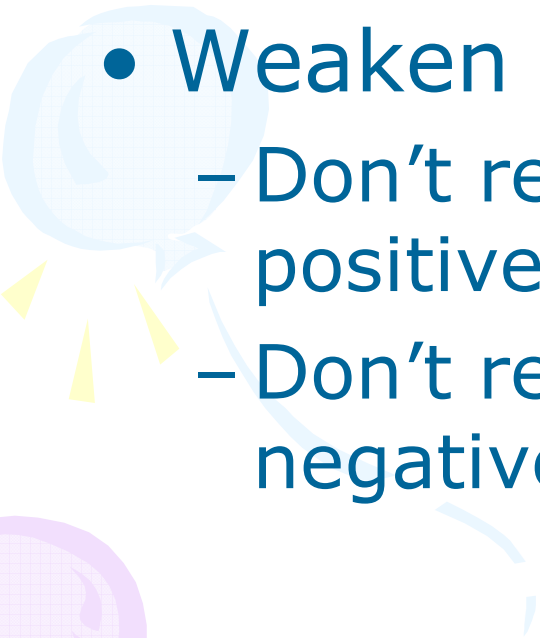
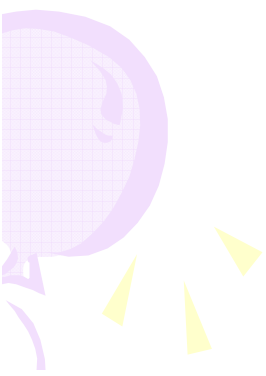


Cross Validation

- To evaluate performance of an algorithm as a whole (rather than a particular hypothesis):
 - Divide data into k subsets
 - K different times
 - Train on $k-1$ of the subsets
 - Test on the held-out subset
 - Return average test score over all k tests
 - Useful for deciding which class of algorithms to use on a particular data set
- 
- 



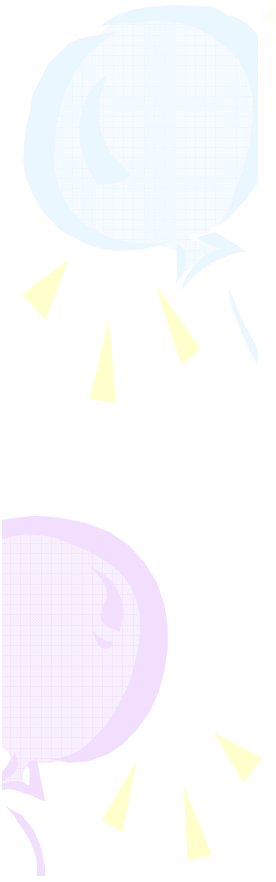
Noisy Data

- Have to accept non-zero on training data
 - Weaken DNF learning algorithm
 - Don't require the hypothesis to cover all positive examples
 - Don't require each rule to exclude all negative examples
- 
- 



Pseudo Code: Noisy DNF Learning

```
P := the set of positive examples
h := False
np := epsilon * number of examples in P
nn := epsilon * number of examples in N
Loop until P has fewer than np elements
  r = True
  N = the set of negative examples
  Repeat until N has fewer than nn elements
    Select a feature  $f_j$  to add to r
     $r := r \wedge f_j$ 
     $N := N - \text{examples in } N \text{ for which } f_j = 0$ 
  h := h  $\vee$  r
  P := P - elements in P covered by r
```



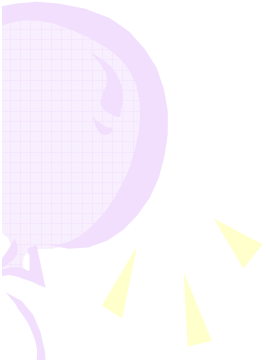
allow epsilon
percentage error



Pseudo Code: Noisy DNF Learning

```
P := the set of positive examples
h := False
np := epsilon * number of examples in P
nn := epsilon * number of examples in N
Loop until P has fewer than np elements or not progressing
  r = True
  N = the set of negative examples
  Repeat until N has fewer than nn elements or not progress
    Select a feature  $f_j$  to add to r
     $r := r \wedge f_j$ 
     $N := N - \text{examples in } N \text{ for which } f_j = 0$ 
  h := h  $\vee$  r
  P := P - elements in P covered by r
```

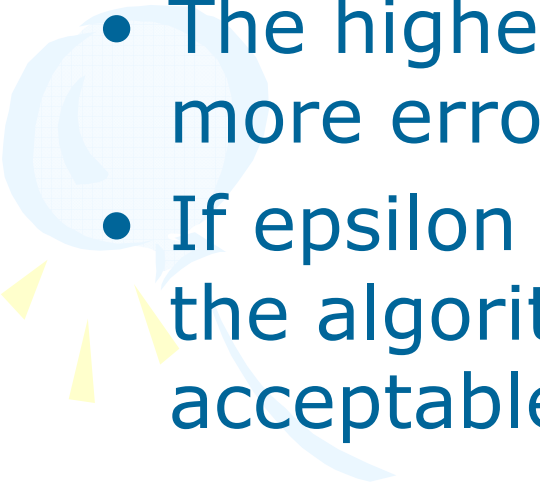
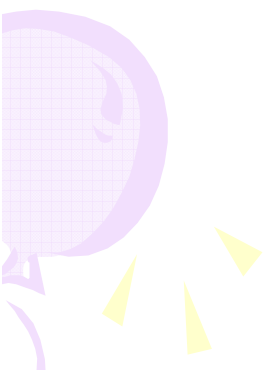
allow epsilon
percentage error



Handle failure to
progress

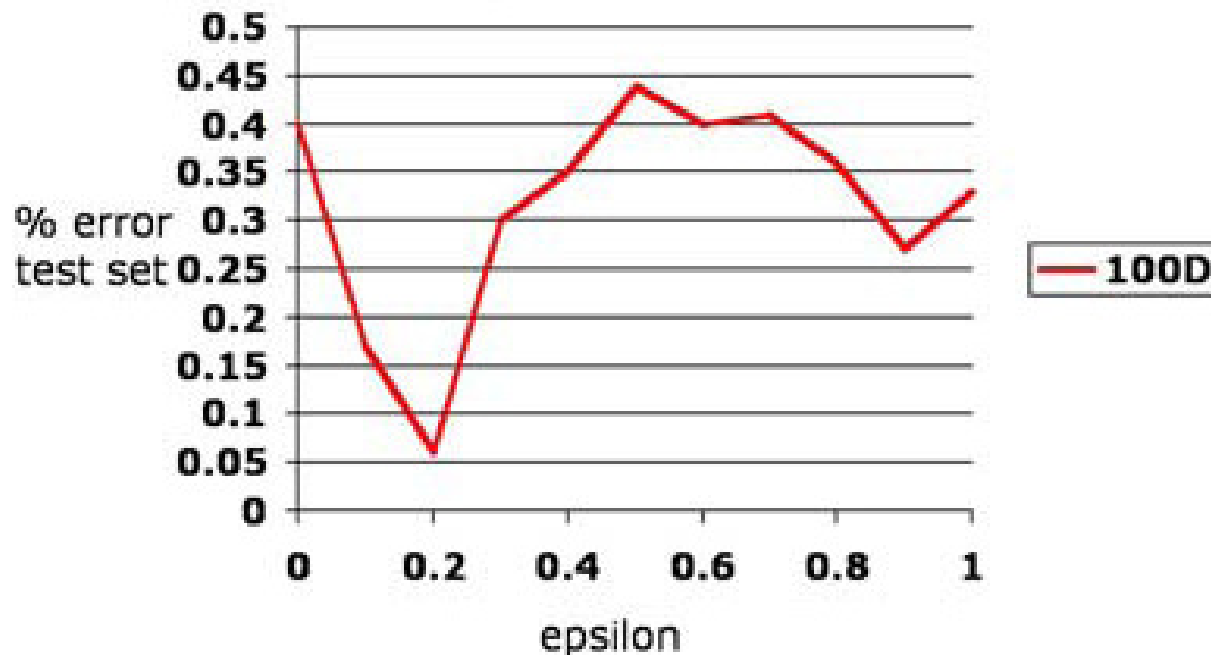


Epsilon is our Delta

- Parameter epsilon is the percentage error allowed
 - The higher the epsilon, the simpler and more error-prone the hypothesis
 - If epsilon is small and the data is noisy, the algorithm may fail to find an acceptable hypothesis
- 
- 

Overfitting curve

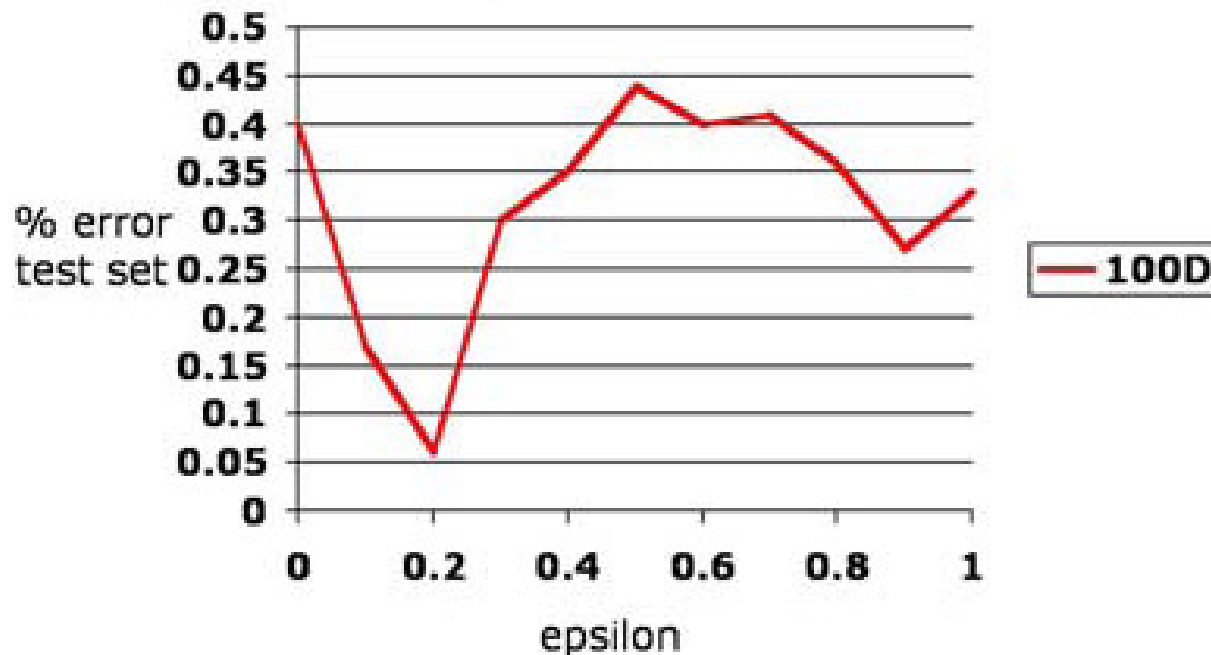
200 input dimensions; function = $f_{22}^{f_{55}} \vee f_{99}^{f_{34}}$



- In this example, the data has 10% noise, i.e. in 10 % of the cases, we expect the output set to be the opposite of the one specified by the target function
- x axis: epsilon ranging from 0 to 1
- y axis: percentage error on fresh data set

Overfitting curve

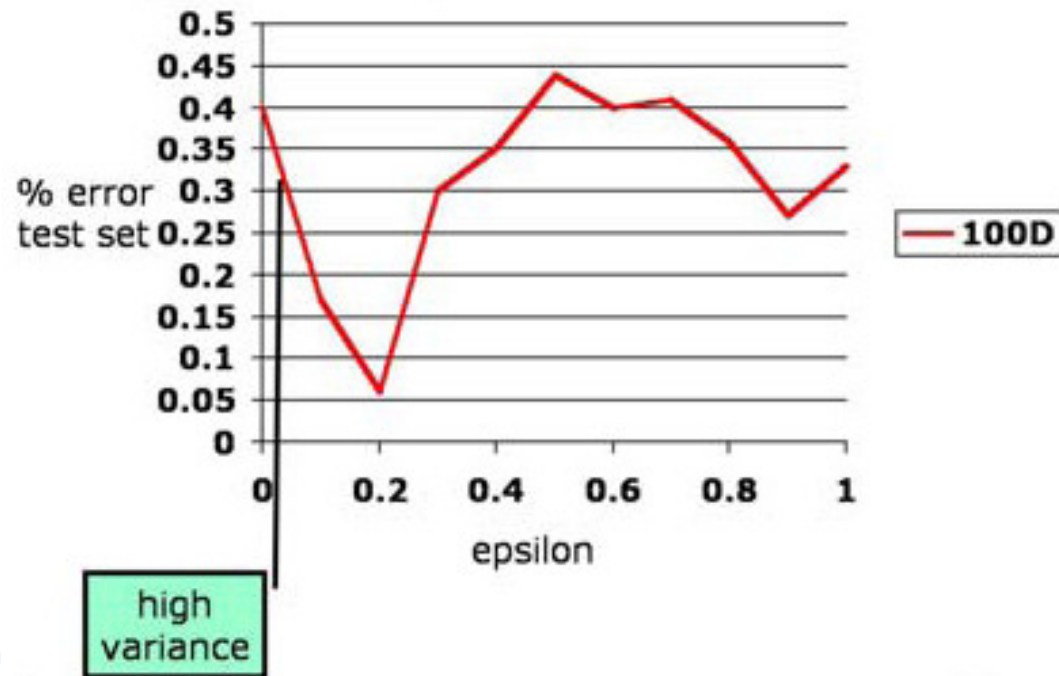
200 input dimensions; function = $f_{22}^{f_{55}} \vee f_{99}^{f_{34}}$



- If we set epsilon close to 0, we run into the problem of overfitting
- In overfitting, we try to reduce error of hypothesis in the training set
- But we may have spent a lot of effort modeling noise in the data, and h may perform poorly on the test set

Overfitting curve

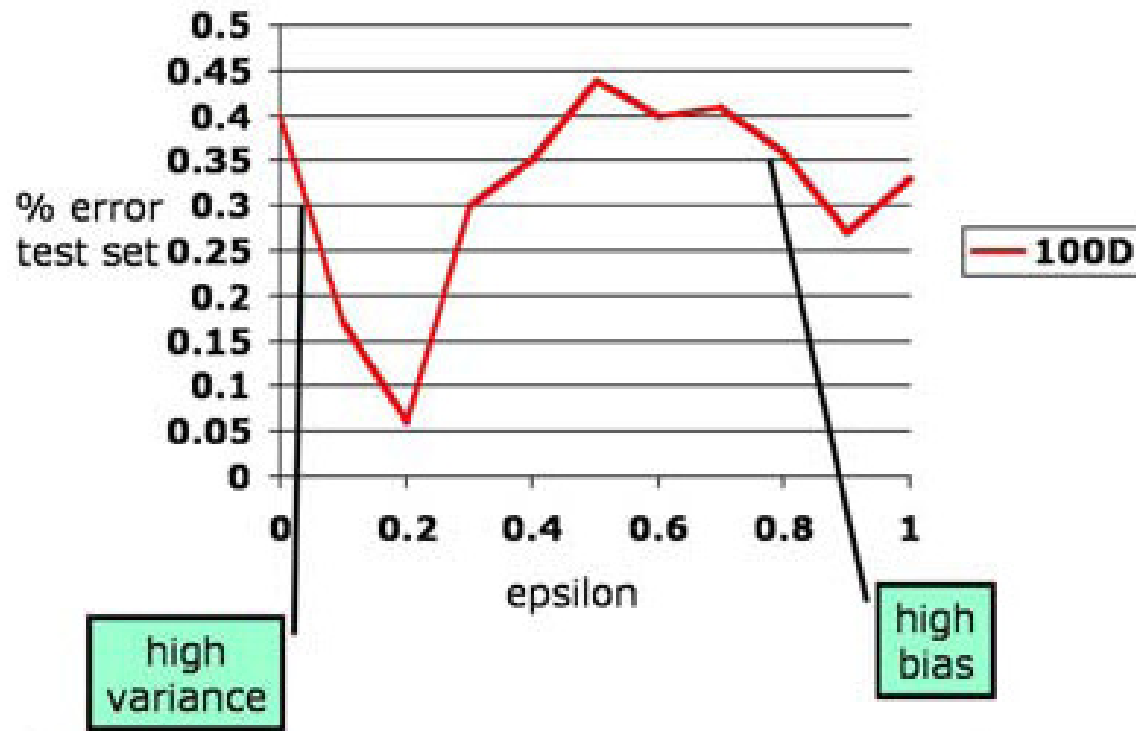
200 input dimensions; function = $f_{22}^{f_{55}} \vee f_{99}^{f_{34}}$



- In the overfitting cases, our algorithm will have high variance, i.e., when we use the algorithm to find h , from another noisy data set generated from the same target function, we will have different h
- If we turn epsilon up a bit, we generate simpler hypotheses that are not so susceptible to variations in the data set and can get better generalization performance

Overfitting curve

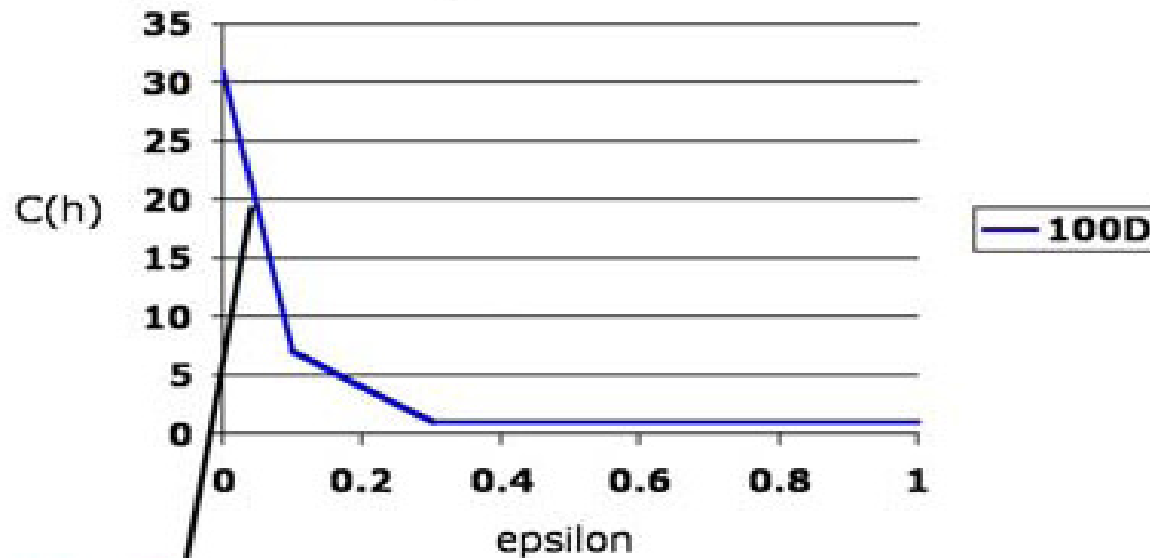
200 input dimensions; function = $f_{22}^{f_{55}} \vee f_{99}^{f_{34}}$



- if epsilon is too high, will keep us from building hypothesis with sufficient complexity
- we'd do poorly because we are unable to even represent the right answer

Hypothesis Complexity

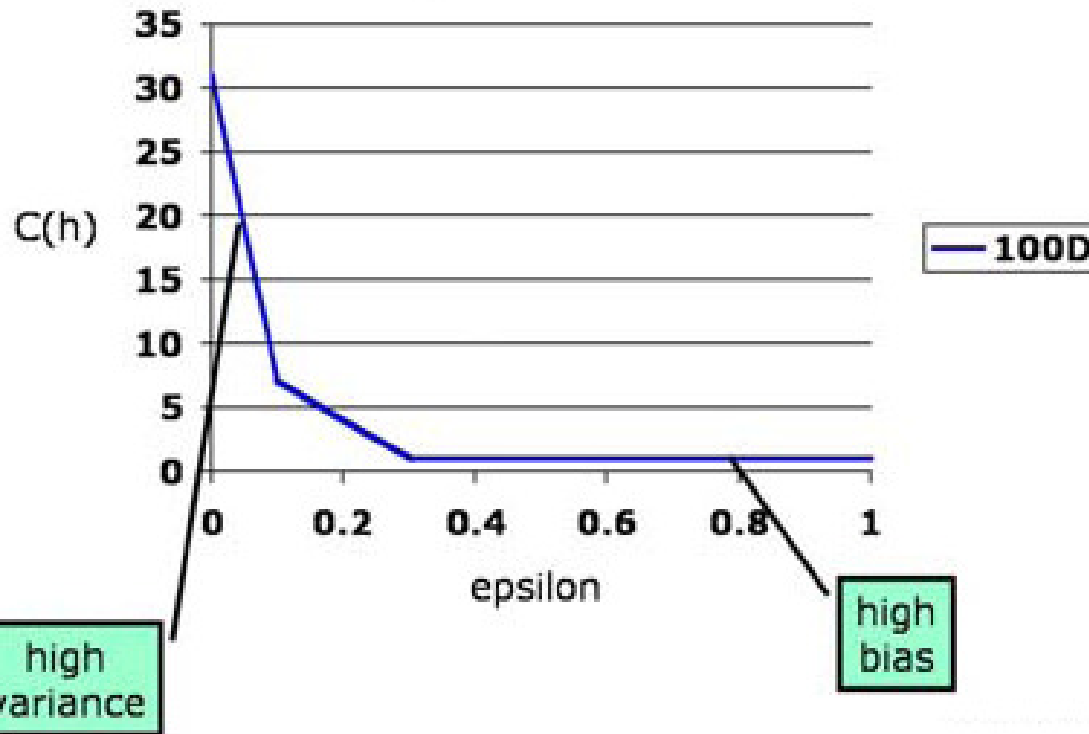
200 input dimensions; function = $f_{22}^{f_{55}} \vee f_{99}^{f_{34}}$



- when epsilon is 0, we are looking for a hypothesis with zero error on the training set
- we are able to find one but it's very complex (31 literals)
- this is clearly a high variance situation; i.e., that hypothesis is completely influenced by the particular training set and would change radically on a newly drawn training set (and therefore high error on test set as well)

Hypothesis Complexity

200 input dimensions; function = $f_{22}^{f_{55}} \vee f_{99}^{f_{34}}$



- When epsilon increase, we rapidly go down to a complexity of 1, which is incapable of representing the target hypothesis



Domains

- Congressional voting: given a congressional voting record (list of 1s and 0s), predict party
 - Spam filtering: encode every message as a vector of features, one per word; a feature is on if that word occurs in the message; predict whether or not the message is a spam
 - Marketing: predict whether a person will buy beer based on previous purchases; encode buying habits with a feature for all products, set 1 if previously purchased
- 