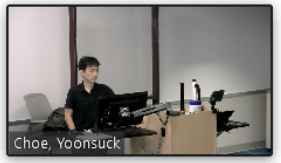
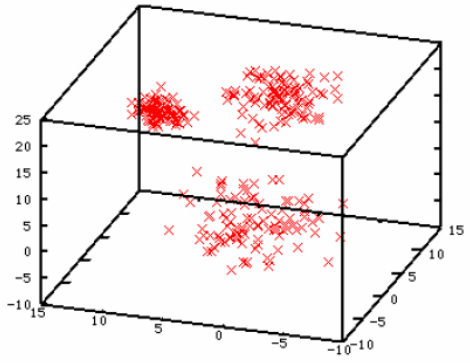


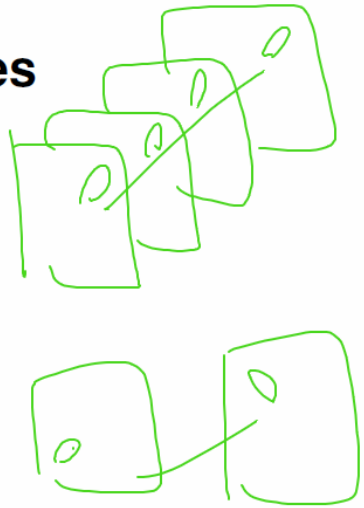
Unsupervised Learning

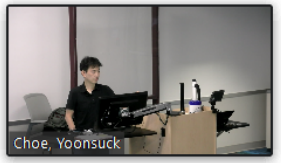


Clustering, feature extraction, blind source separation, dimensionality reduction, etc.

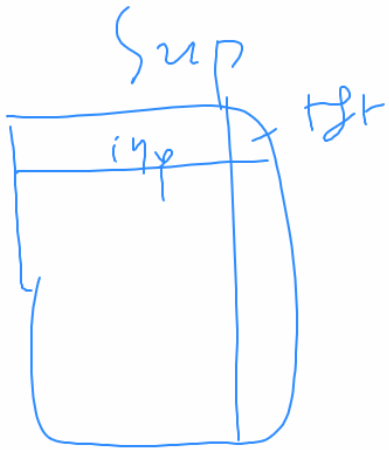
- Principal Component Analysis (PCA)
- Self-Organizing Maps (SOM)
- Independent Component Analysis (ICA)
- Multi-Dimensional Scaling (MDS)
- ISOMAP, Locally Linear Embedding (LLE)
- t-distr. Stochastic N[illegible]

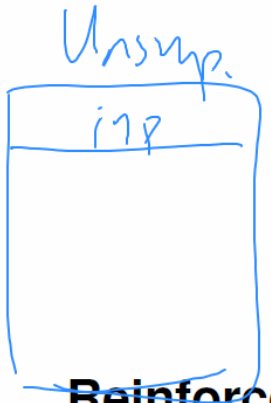
Unsupervised Learning Issues





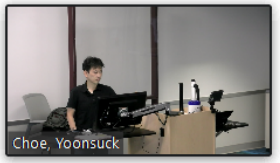
- Discovering structure.
- Discovering features.
- Removing redundancy.
- How many clusters?
- What distance measures to use?





Reinforcement Learning

Reinforcement Learning

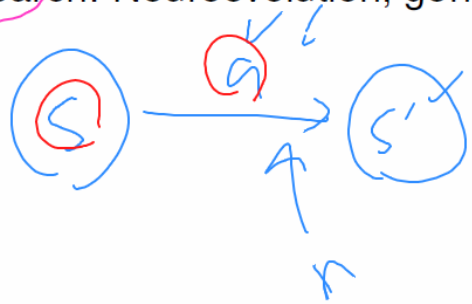


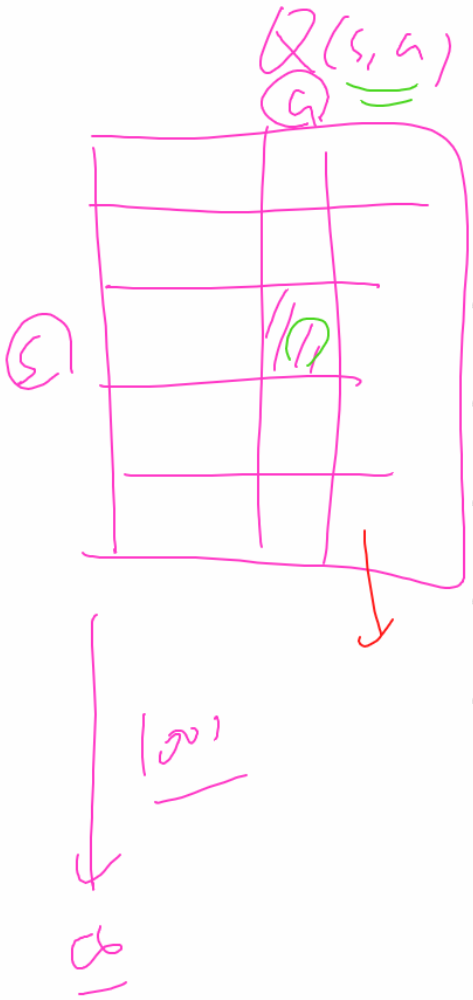
Policy function

$\pi : S \rightarrow a$

- Very different from supervised and unsupervised learning.
- Multi agent control, robot control, game playing, scheduling, etc.
- Techniques:
  - Value function-based: Q-learning, Temporal difference (TD) learning
  - Direct policy search: Neuroevolution, genetic algorithms.

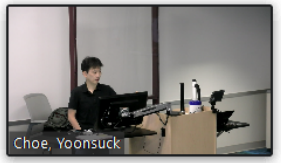
$V(s)$ $Q(s,a)$

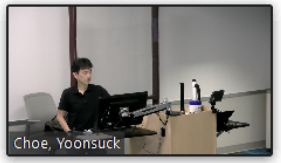




Reinforcement Learning Issues

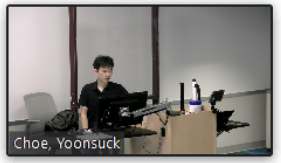
- Discrete states and actions was a norm.
- Scalability an issue.
- Certain assumptions: state-action pair visited infinitely often.
- Online learning, safety, transfer, imitation, etc.
- Deep reinforcement learning disrupted a lot of the traditional assumptions.





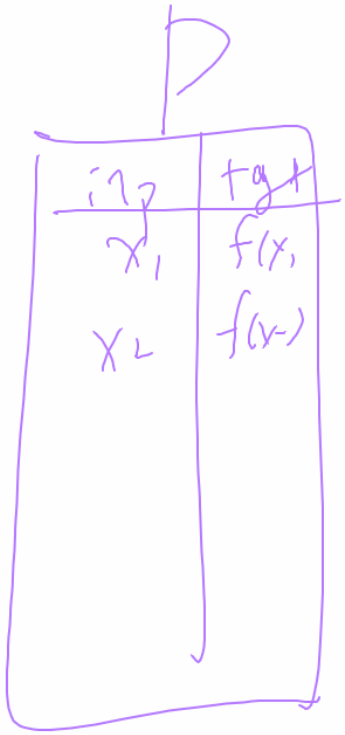
Summary (ML Intro)

- Machine learning is a rapidly developing field with great promise:
  - Big data
  - Deep neural networks
  - Fast computing: GPGPU, cloud, etc.
- Three types of ML:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Need to look beyond ML:
  - ML good at solving problems, but not posing problems (Choe and Mann 2012).



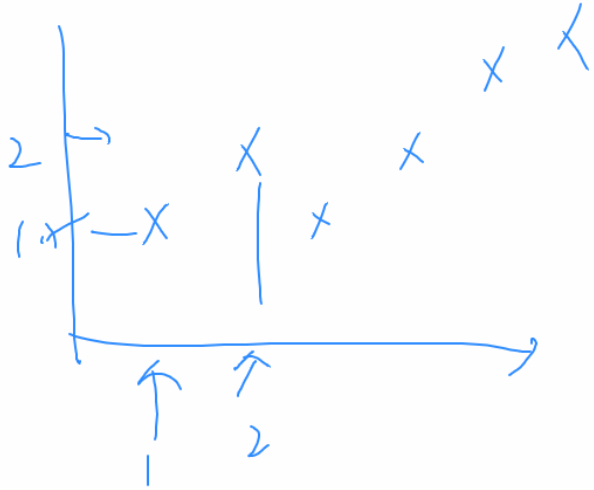
Summary (ML Intro)

- Machine learning is a rapidly developing field with great promise:
  - Big data
  - Deep neural networks
  - Fast computing: GPGPU, cloud, etc.
- Three types of ML:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Need to look beyond ML:
  - ML good at solving problems, but not posing problems (Choe and Mann 2012).

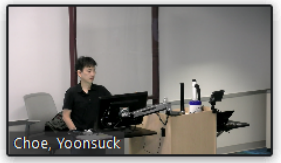


Inductive Learning

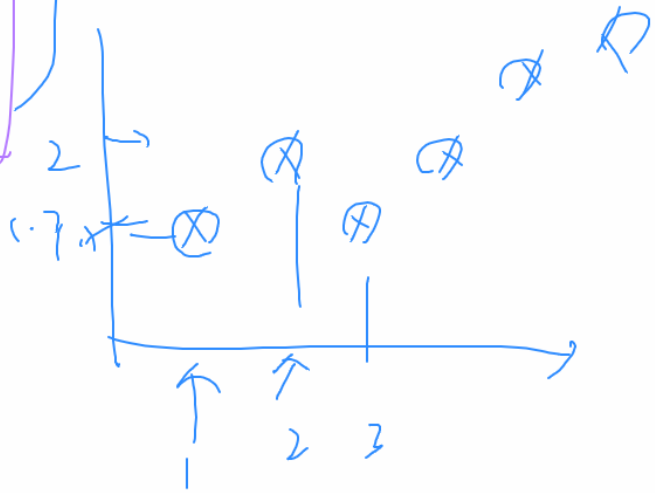
- Given **example** pairs $(x, f(x))$, return a function h that approximates the function f:
  - **pure inductive inference,** or **induction**.
- The function h is called a **hypothesis**.

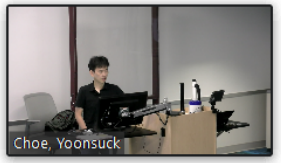


Inductive Learning



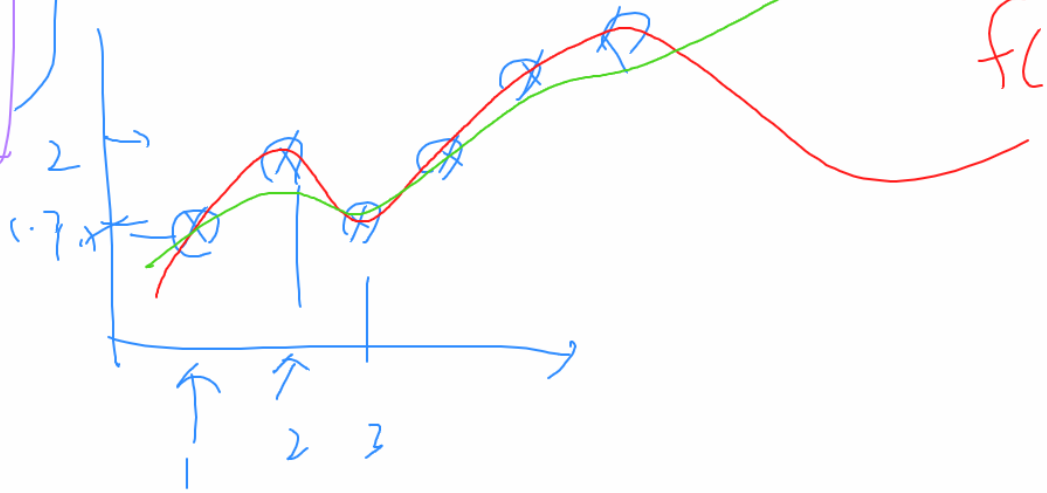
- Given **example** pairs $(x, f(x))$, return a function h that approximates the function f:
  - **pure inductive inference,** or **induction**.
- The function h is called a **hypothesis**.





Inductive Learning

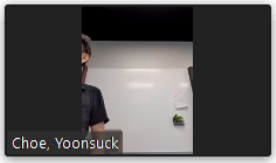
- Given **example** pairs $(x, f(x))$, return a function h that approximates the function f:
  - **pure inductive inference,** or **induction.**
- The function h is called a **hypothesis**.



h = b

h(x) : hypothesis : model

f(x) : unknown.



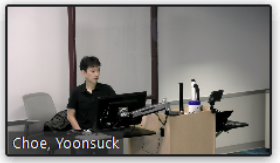
Supervised Learning

- Regression: approximating $y = f(x)$
- Classification: face recognition, hand-written character recognition, credit risk assessment, etc.
- Techniques:
  - Neural networks
  - Decision tree learning
  - Support vector machines
  - Radial basis functions
  - Naive Bayes learning
  - k-nearest neighbor

→ ML 421

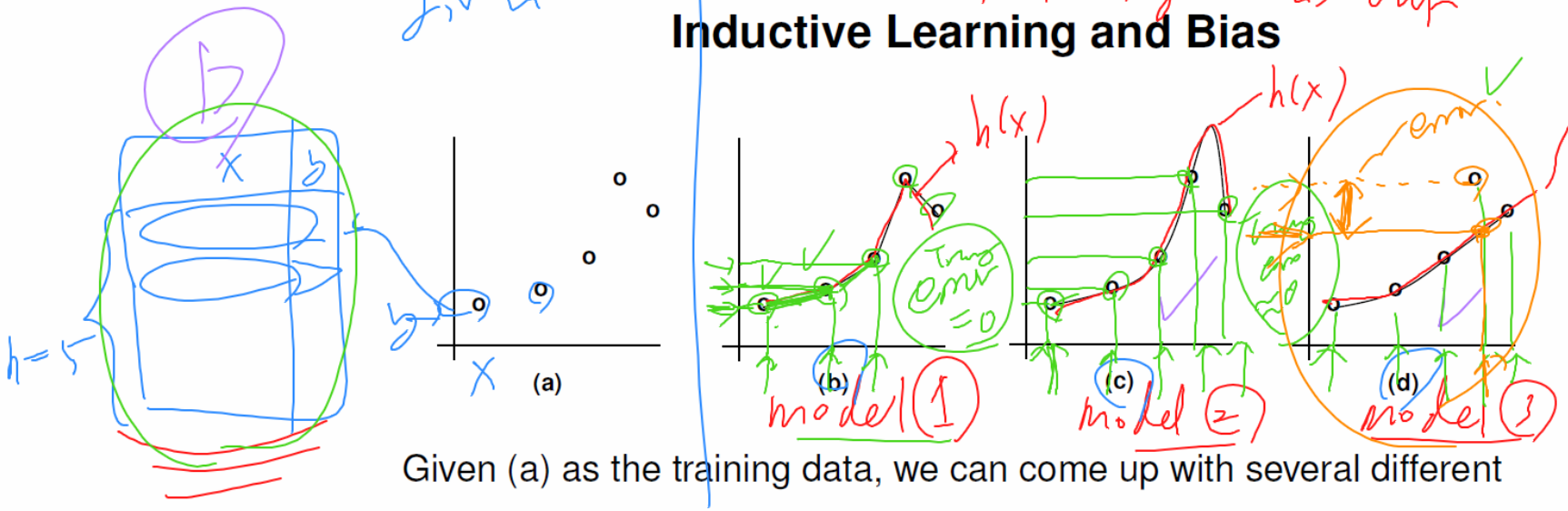
these other machine learning.

Inductive Learning



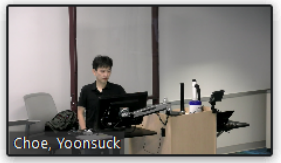
- Given **example** pairs $(x, f(x))$, return a function h that approximates the function f:
  - **pure inductive inference,** or **induction**.
- The function h is called a **hypothesis**.

Inductive Learning and Bias

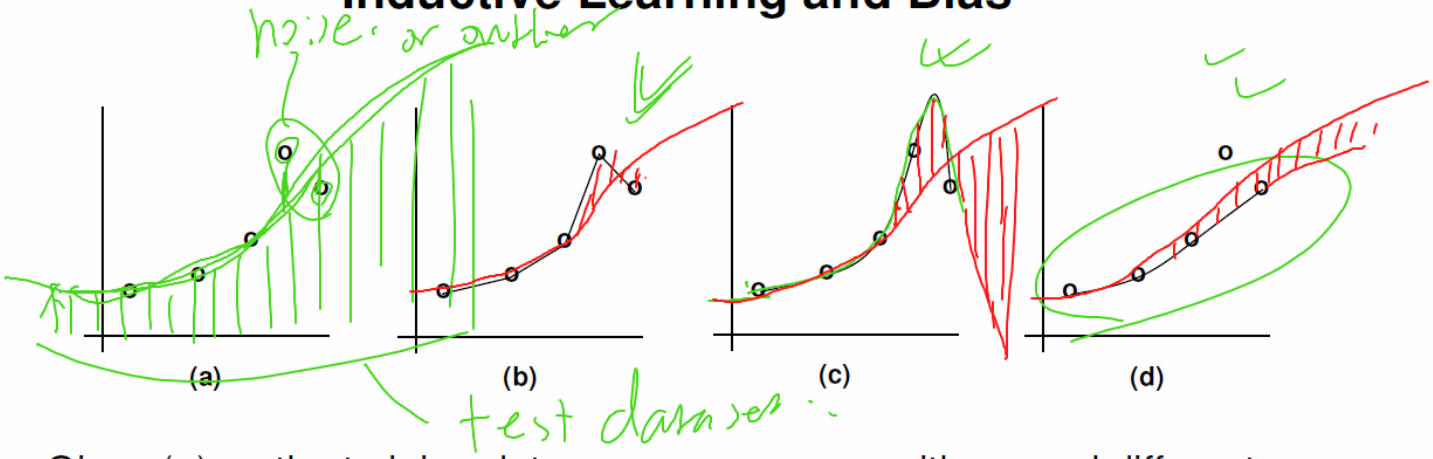


Given (a) as the training data, we can come up with several different hypotheses: (b) to (d)

- selection of one hypothesis over another is called an **inductive bias**.
  - exact match to training data
  - prefer imprecise but smooth approximation
  - etc.

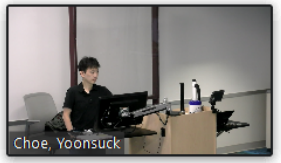


Inductive Learning and Bias

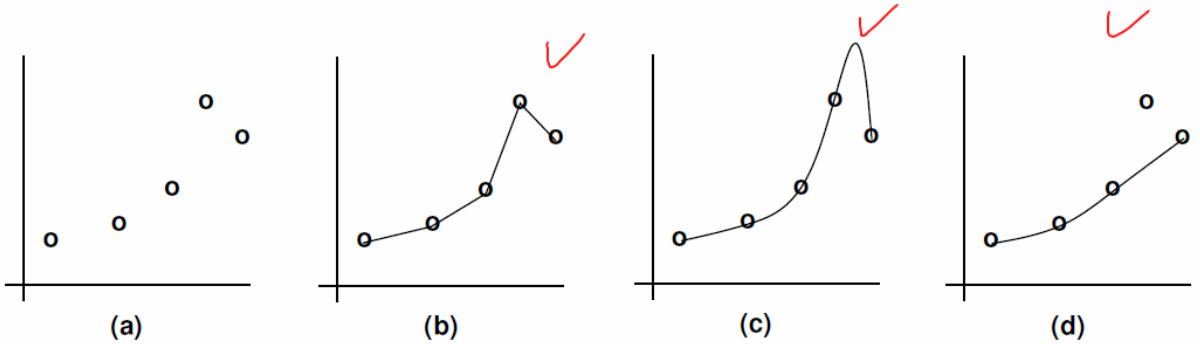


Given (a) as the training data, we can come up with several different hypotheses: (b) to (d)

- selection of one hypothesis over another is called an **inductive bias**.
  - exact match to training data
  - prefer imprecise but smooth approximation
  - etc.

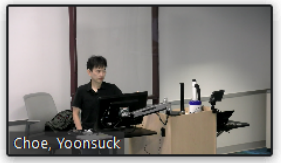


Inductive Learning and Bias



Given (a) as the training data, we can come up with several different hypotheses: (b) to (d)

- selection of one hypothesis over another is called an **inductive bias**.
  - exact match to training data
  - prefer imprecise but smooth approximation
  - etc.

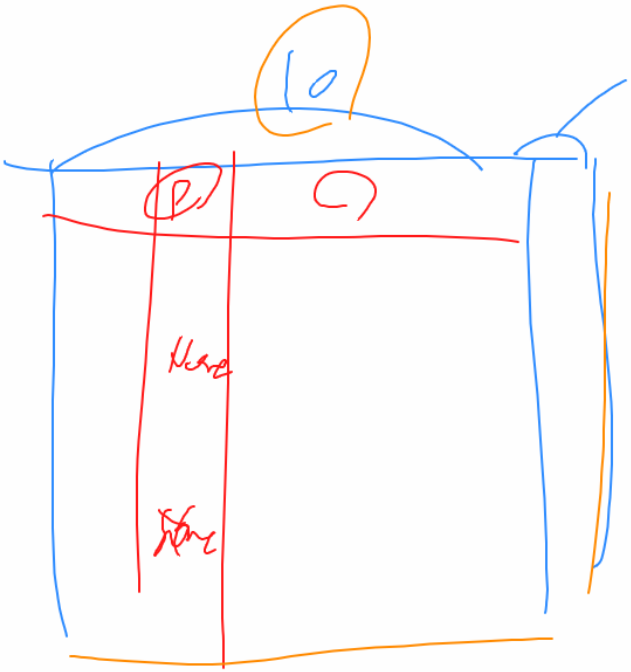


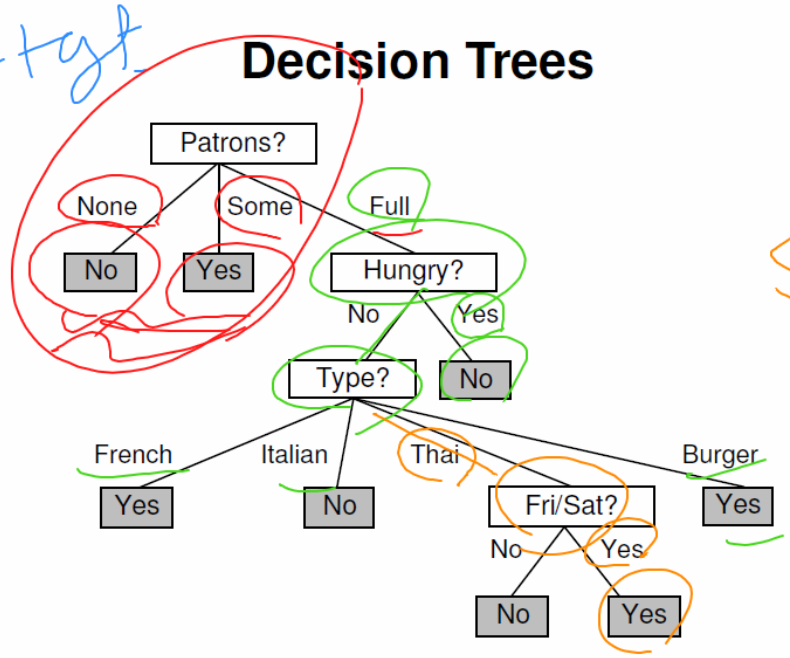
Constructing Decision Trees from Examples

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

- Given a set of examples (**training set**), both **positive** and **negative**, the task is to construct a decision tree that describes a concise decision path.
- Using the resulting decision tree, we want to **classify** new instances of examples (either as **yes** or **no**).

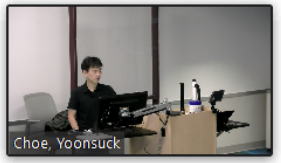
So if you can just check 3 attributes instead of 10 to figure out this answer, What's your less than that video



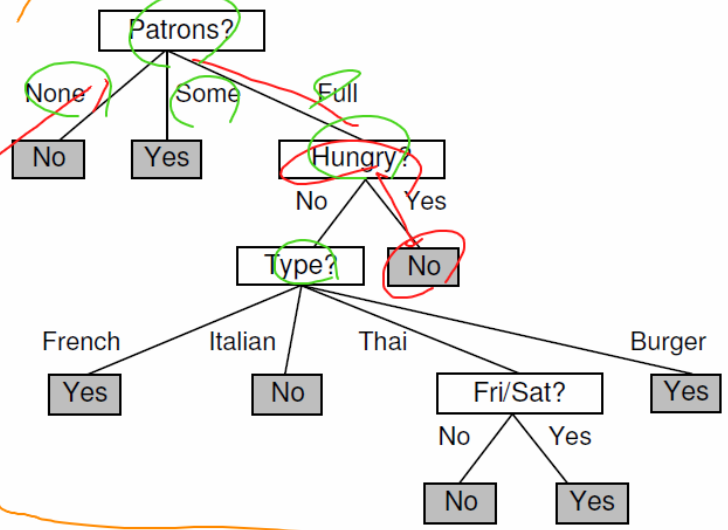


- learn to approximate **discrete-valued** target functions.
- step-by-step decision making (disjunction of conjunctions)
- applications: medical diagnosis, assess credit risk of loan applicants, etc.

So this is a very, very economical way of representing your entire knowledge that you gathered in the data set

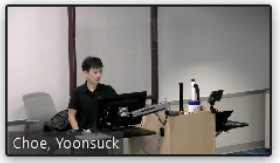


Decision Trees

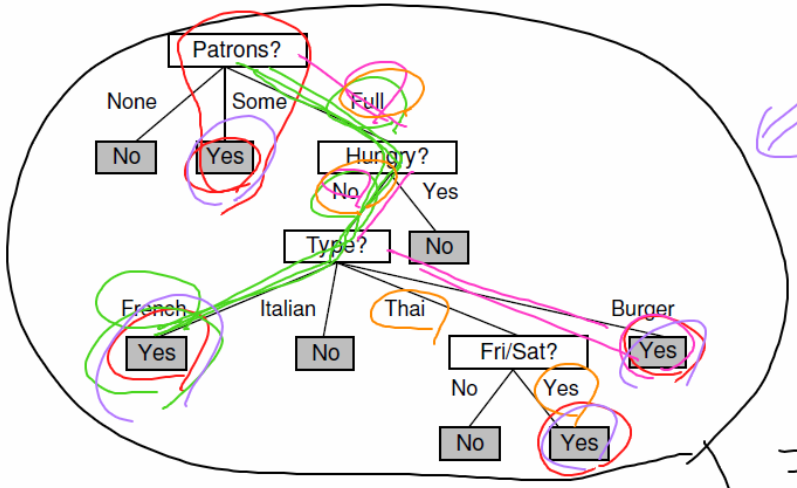


- learn to approximate **discrete-valued** target functions.
- step-by-step decision making (disjunction of conjunctions)
- applications: medical diagnosis, assess credit risk of loan applicants, etc.

Each of these attributes might be different, depending on the domain and the possible values might also



Decision Trees: What They Represent

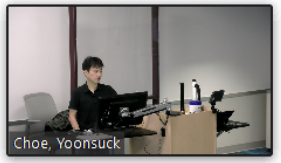


Wait or not (**Yes**/**No**)? The decision tree above corresponds to:

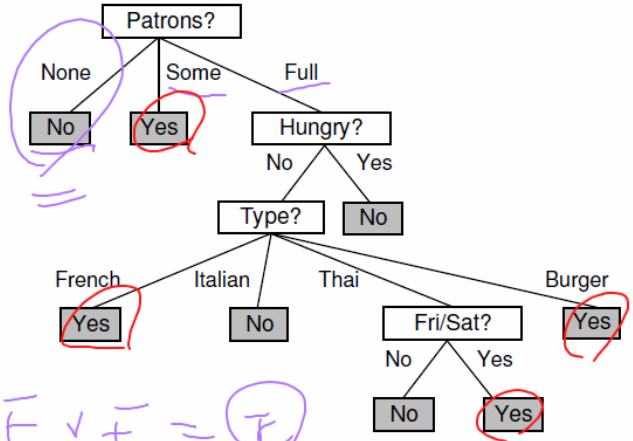
$$(Patrons = Some)$$
$$\vee (Patrons = Full \wedge Hungry = No \wedge Type = French)$$
$$\vee (Patrons = Full \wedge Hungry = No \wedge Type = Thai \wedge Fri/Sat = Yes)$$
$$\vee (Patrons = Full \wedge Hungry = No \wedge Type = Burger)$$

Decision trees represent disjunction of conjunctions.

And then, finally, you link them up with this function either this or that or that, or that, and that's your that's your logical expression that fully describes this



Decision Trees: What They Represent

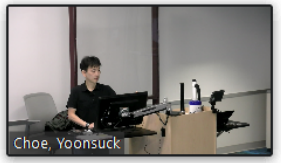


Wait or not (**Yes**/**No**)? The decision tree above corresponds to:

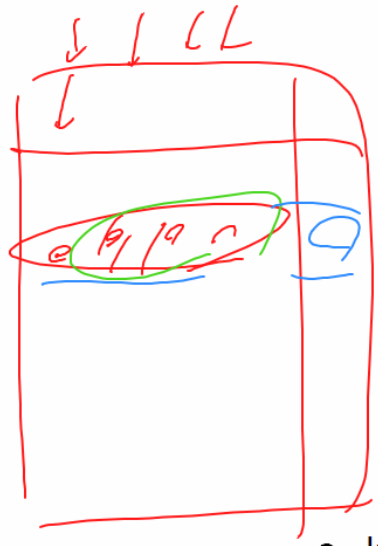
$$
\begin{aligned}
&(Patrons = Some) \\
\vee &(Patrons = Full \wedge Hungry = No \wedge Type = French) \\
\vee &(Patrons = Full \wedge Hungry = No \wedge Type = Thai \wedge Fri/Sat = Yes) \\
\vee &(Patrons = Full \wedge Hungry = No \wedge Type = Burger)
\end{aligned}
$$

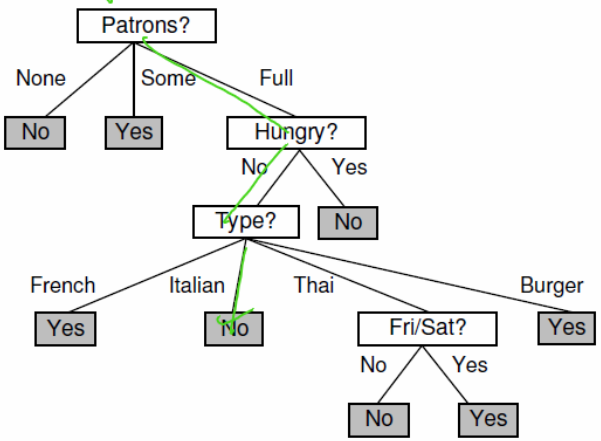
Decision trees represent disjunction of conjunctions.

These that have



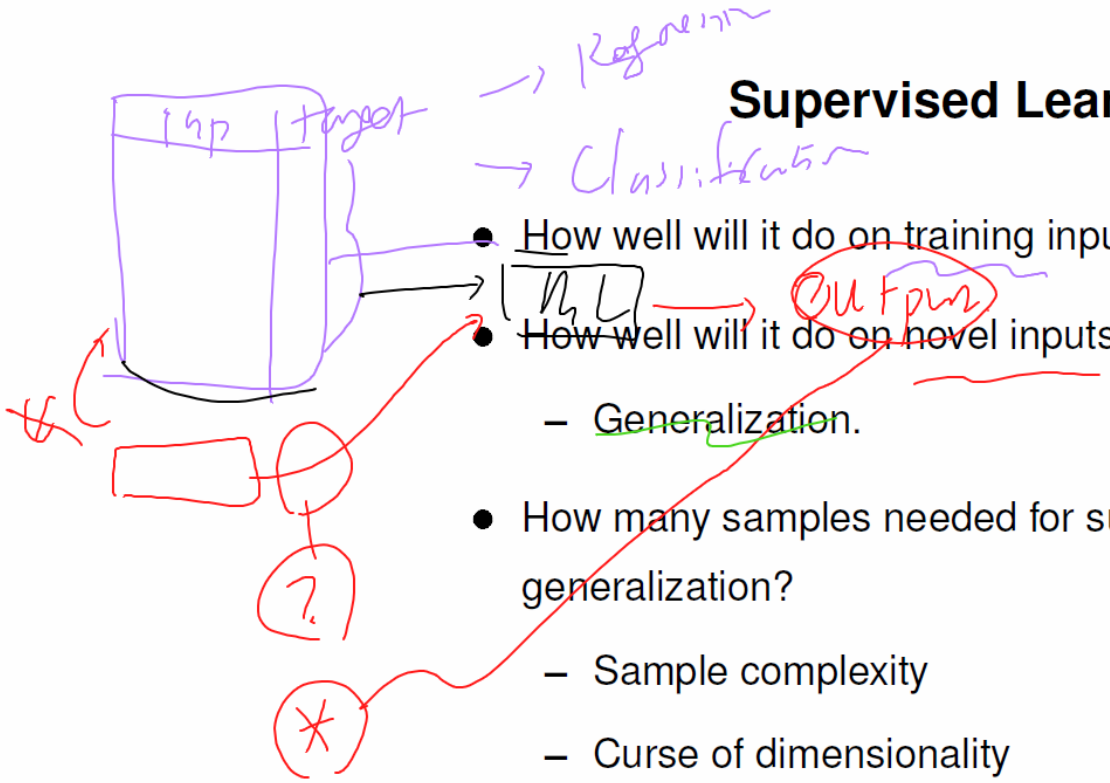
Decision Trees: What They Represent (cont'd)





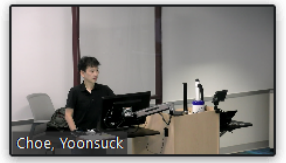
- In other words, for each instance (or example), there are attributes (Patrons, Hungry, etc.) and each instance have a full attribute value assignment.
- For a given instance, it is classified into different discrete classes by the decision tree.
- For training, many $(instance, class)$ pairs are used.

So instance and class pairs as the input

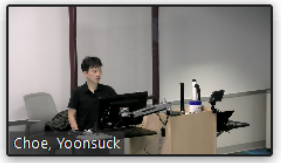


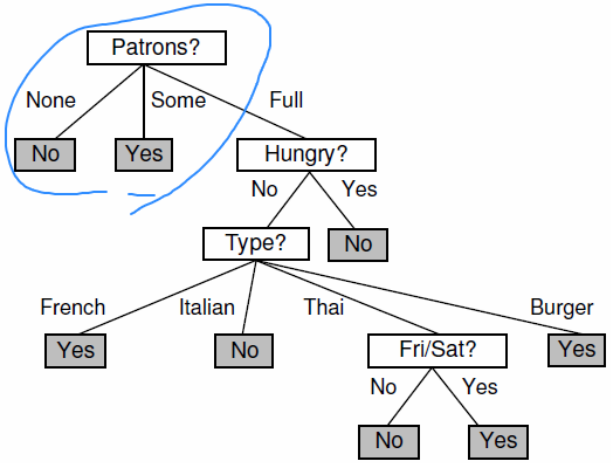
Supervised Learning Issues

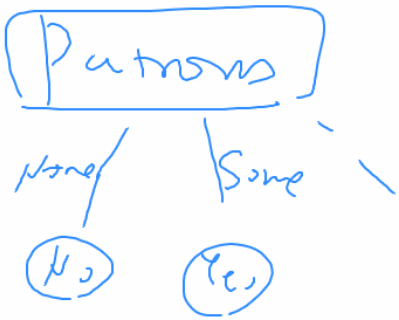
- How well will it do on training inputs?
- How well will it do on novel inputs?
  - Generalization.
- How many samples needed for sufficient performance and generalization?
  - Sample complexity
  - Curse of dimensionality
  - Computational learning theory
- Catastrophic forgetting (online learning hard).



Decision Trees: What They Represent (cont'd)



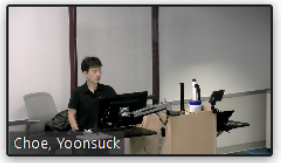


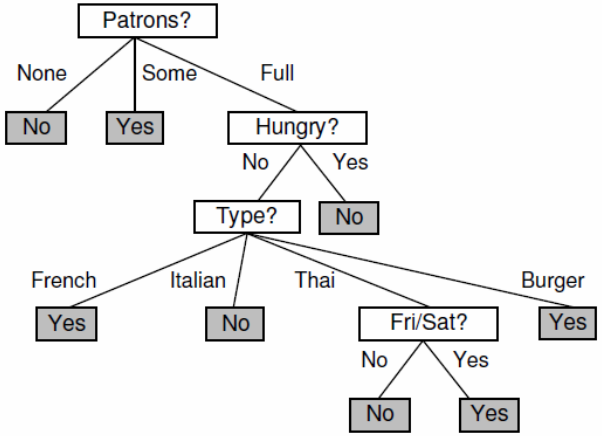


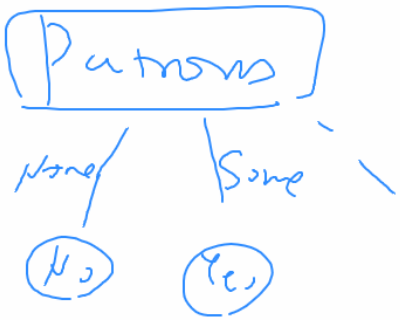
- In other words, for each instance (or example), there are attributes (Patrons, Hungry, etc.) and each instance have a full attribute value assignment.
- For a given instance, it is classified into different discrete classes by the decision tree.
- For training, many $(instance, class)$ pairs are used.

When the patrons were some, and so is yes so a and otherwise let's look at just

Decision Trees: What They Represent (cont'd)



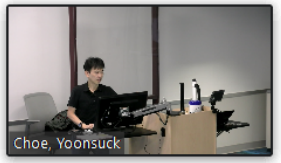




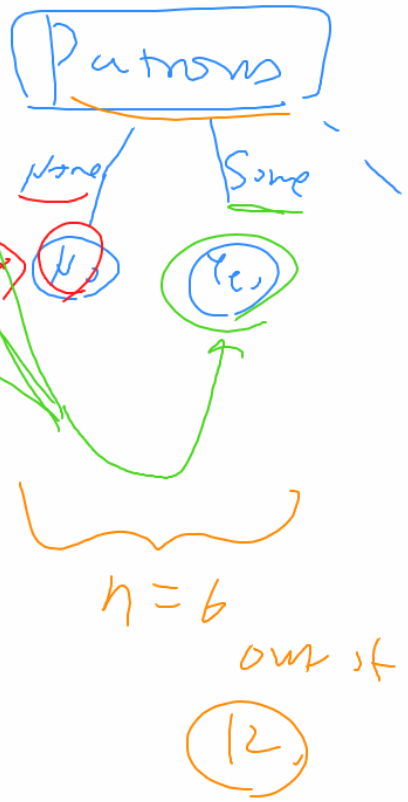
- In other words, for each instance (or example), there are attributes (Patrons, Hungry, etc.) and each instance have a full attribute value assignment.
- For a given instance, it is classified into different discrete classes by the decision tree.
- For training, many $(instance, class)$ pairs are used.

When the patrons were some, and so is yes, so a and otherwise. let's look at just this and see what happens in the data set

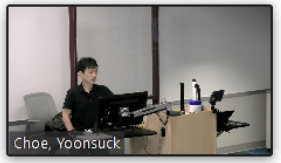
Constructing Decision Trees from Examples



Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*



- Given a set of examples (**training set**), both **positive** and **negative**, the task is to construct a decision tree that describes a concise decision path.
- Using the resulting decision tree, we want to **classify** new instances of examples (either as **yes** or **no**).



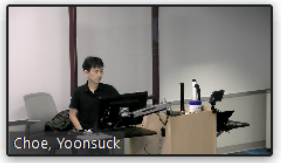
Constructing Decision Trees from Examples

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

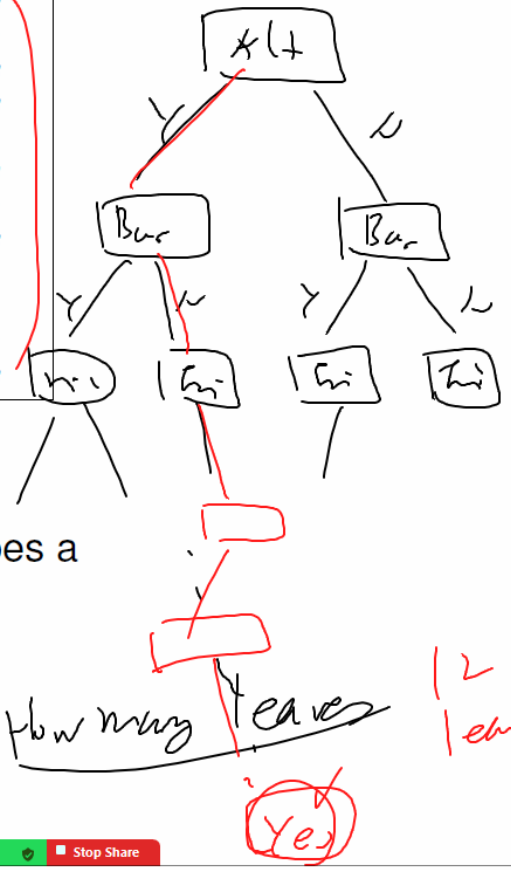
- Given a set of examples (**training set**), both **positive** and **negative**, the task is to construct a decision tree that describes a concise decision path.
- Using the resulting decision tree, we want to **classify** new instances of examples (either as **yes** or **no**).

time give you the answer right off the bat, even without going to any other attributes

Constructing Decision Trees from Examples



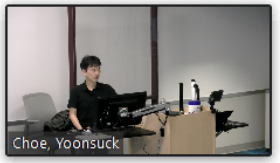
Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*



- Given a set of examples (**training set**), both **positive** and **negative**, the task is to construct a decision tree that describes a concise decision path.
- Using the resulting decision tree, we want to **classify** new instances of examples (either as **yes** or **no**).

So if anything new that come in where this was not part of this, then it would be kind of like

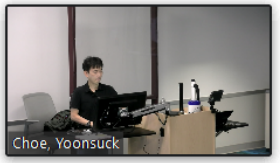
Constructing Decision Trees: Trivial Solution



- A trivial solution is to explicitly construct paths for each given example.
- The problem with this approach is that it is not able to deal with situations where, some attribute values are missing or new kinds of situations arise.
- Consider that some attributes may not count much toward the final classification.

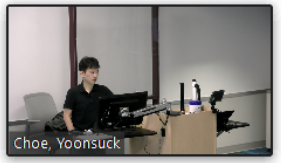
But also it's not able to deal with new values when they're when the data contains noise or missing values

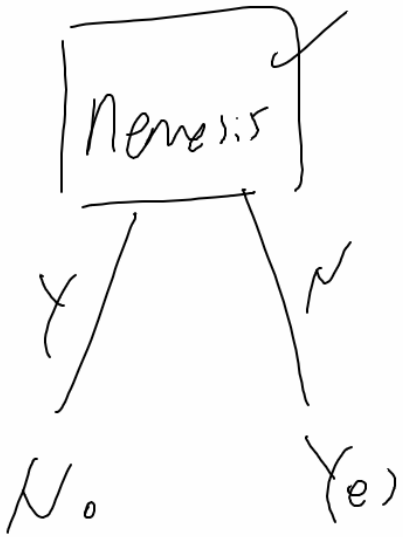
Finding a Concise Decision Tree



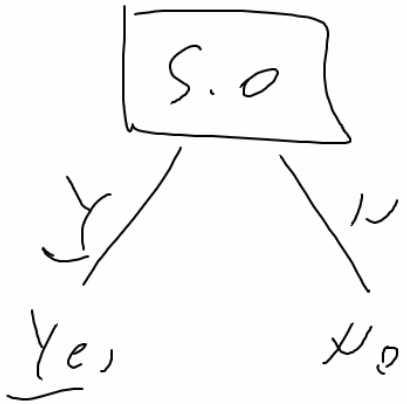
- Memorizing all cases may not be the best way.
- We want to extract a decision pattern that can describe a large number of cases in a **concise** way.
- Such an inductive bias is called **Ockham's razor**: *The most likely hypothesis is the simplest one that is consistent with all observations.*
- In terms of a decision tree, we want to make as few tests before reaching a decision, i.e. the depth of the tree should be shallow.

Finding a Concise Decision Tree (cont'd)



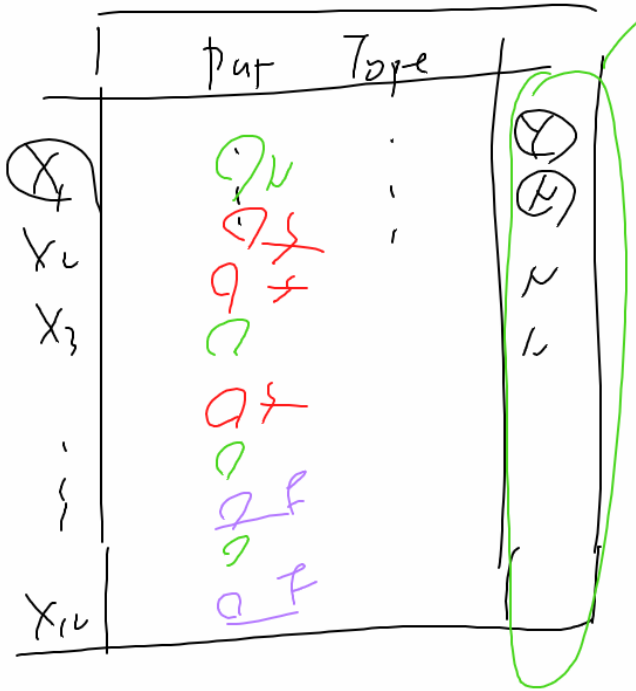


- Basic idea: pick up attributes that can clearly separate positive and negative cases.
- These attributes are more important than others: the final classification heavily depend on the value of these attributes.

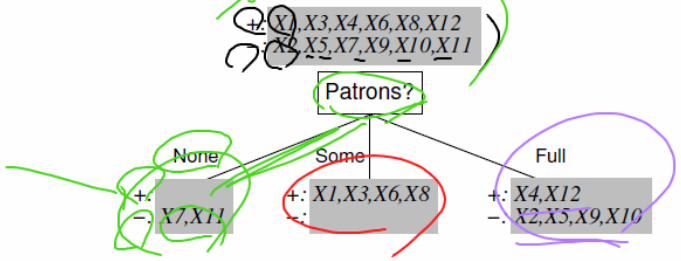


And so these in these are more important than all that attributes, because they can help you reduce the depths

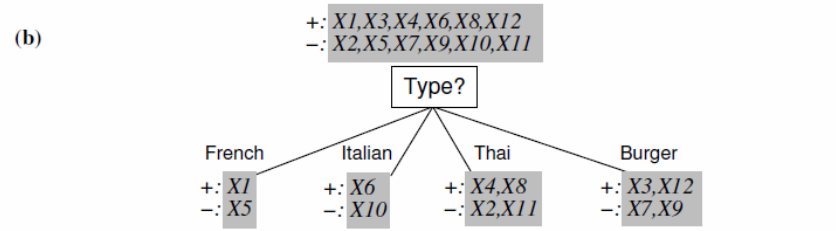
Finding a Concise Decision Tree (cont'd)



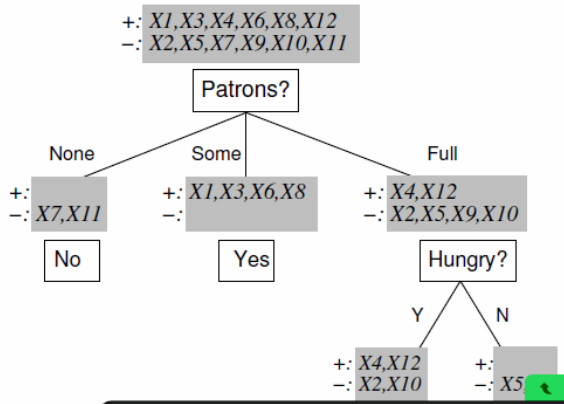
(a)

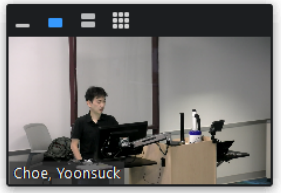


(b)



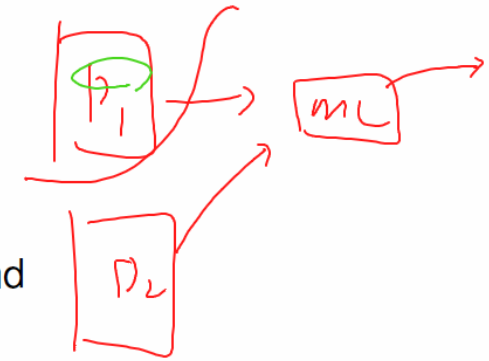
(c)



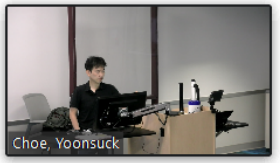


Supervised Learning Issues

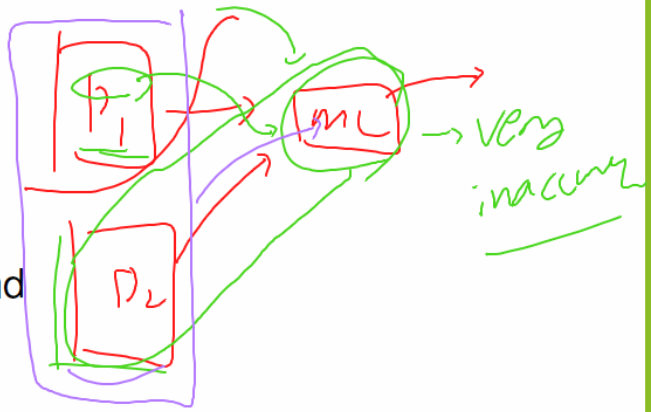
- How well will it do on training inputs?
- How well will it do on novel inputs?
  - Generalization.
- How many samples needed for sufficient performance and generalization?
  - Sample complexity
  - Curse of dimensionality
  - Computational learning theory
- Catastrophic forgetting (online learning hard).

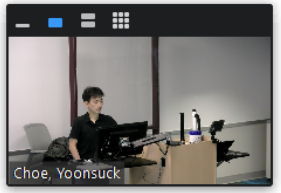


Supervised Learning Issues

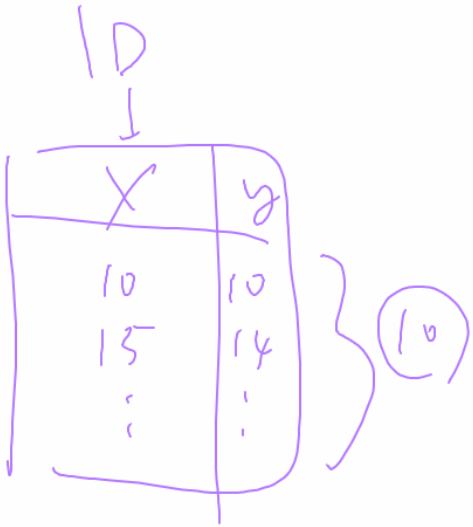


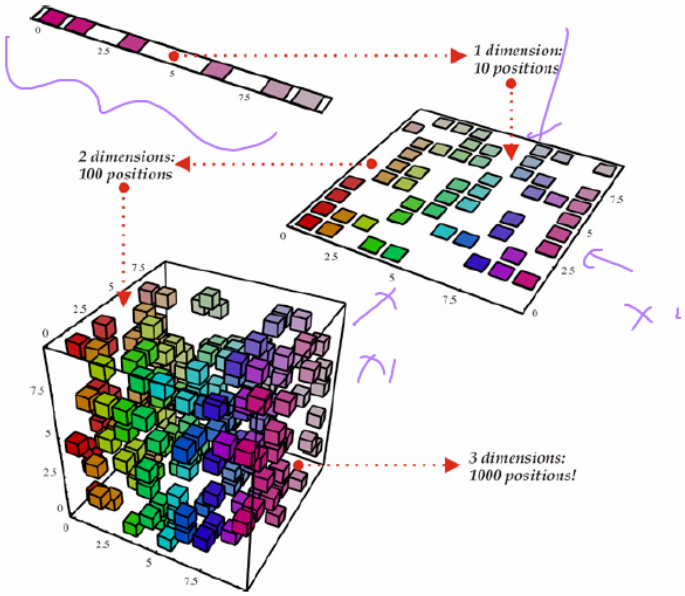
- How well will it do on training inputs?
- How well will it do on novel inputs?
  - Generalization.
- How many samples needed for sufficient performance and generalization?
  - Sample complexity
  - Curse of dimensionality
  - Computational learning theory
- Catastrophic forgetting (online learning hard).

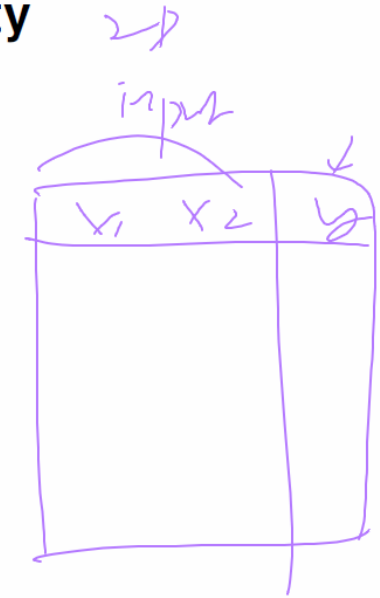




Addendum: Curse of Dimensionality



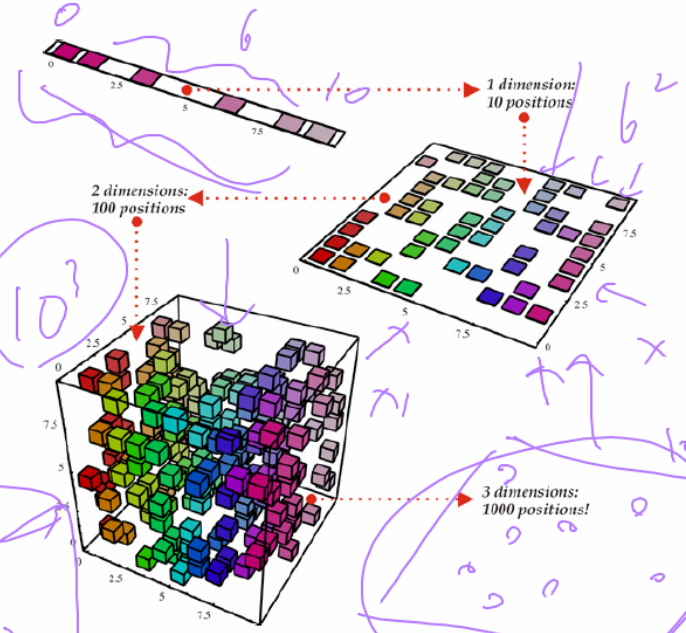




From: Yoshua Bengio's page

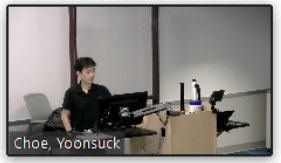
- Exponentially many points needed to achieve same density of training samples.

Addendum: Curse of Dimensionality

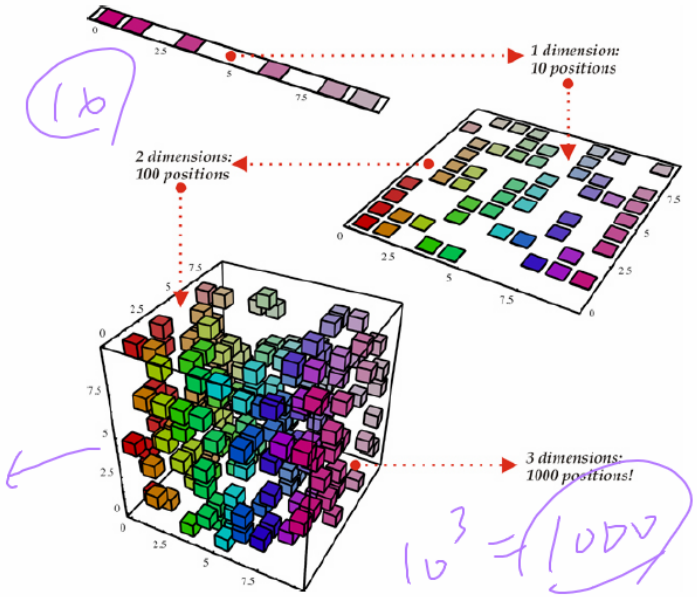


From: Yoshua Bengio's page

- Exponentially many points needed to achieve same density of training samples.



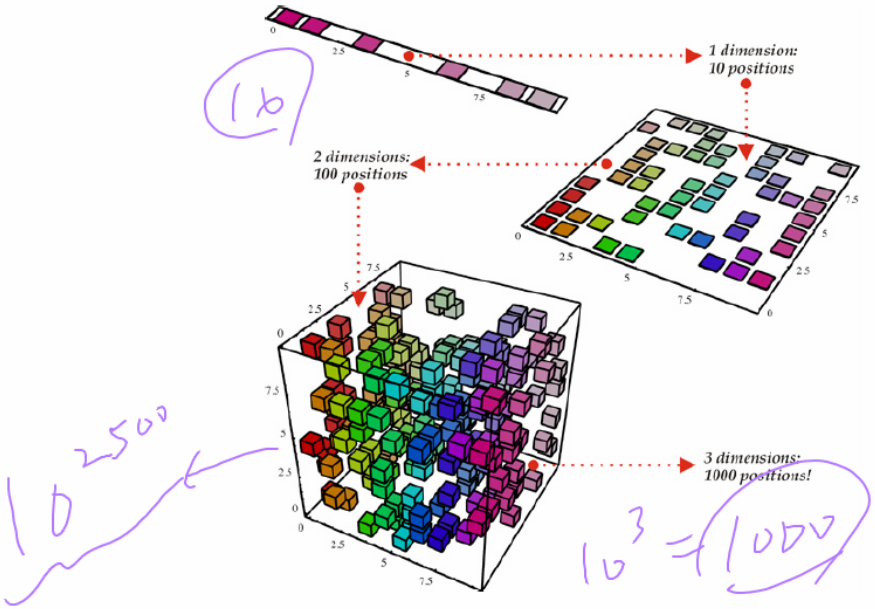
Addendum: Curse of Dimensionality



From: Yoshua Bengio's page

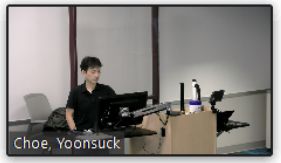
- Exponentially many points needed to achieve same density of training samples.

Addendum: Curse of Dimensionality

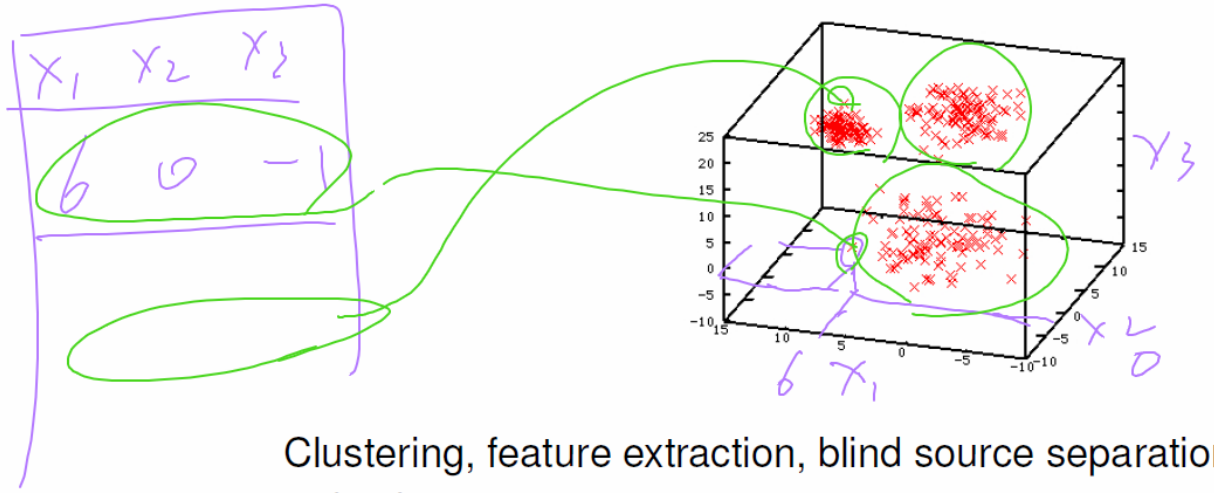


From: Yoshua Bengio's page

- Exponentially many points needed to achieve same density of training samples.



Unsupervised Learning



Clustering, feature extraction, blind source separation, dimensionality reduction, etc.

- Principal Component Analysis (PCA)
- Self-Organizing Maps (SOM)
- Independent Component Analysis (ICA)
- Multi-Dimensional Scaling (MDS)
- ISOMAP, Locally Linear Embedding (LLE)
- t-distr. Stochastic N...