CS 161 Intro. To Artificial Intelligence

Week 9, Discussion 1D

Today's Topics

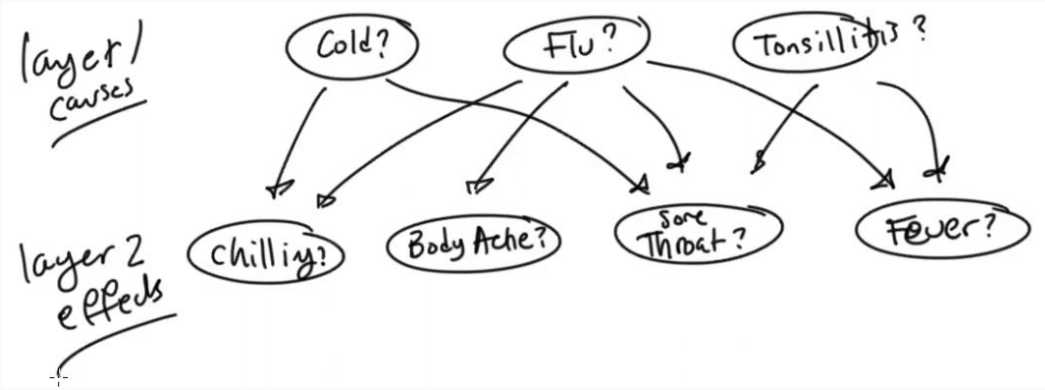
- **BN Learning and Inference**
  - Learning parameters
  - Learning BN structure
  - Model-oriented vs. Query-oriented learning
- Decision Tree and Random Forest
  - Entropy and Conditional Entropy
  - Classifier
  - BN Classifier
- Hint of HW9

BN Learning - Example

Example

The flu is an acute disease characterized by fever, body aches and pains, and can be associated with chilling and a sore throat. The cold is a bodily disorder popularly associated with chilling and can cause a sore throat. Tonsillitis is inflammation of the tonsils which leads to a sore throat and can be associated with fever.

- Ways to learn parameters (probabilities) and construct CPTs:
  - Problem statement
  - Subjective beliefs
  - **Learning from data**



BN Learning

CPTs can also be estimated from medical records of previous patients

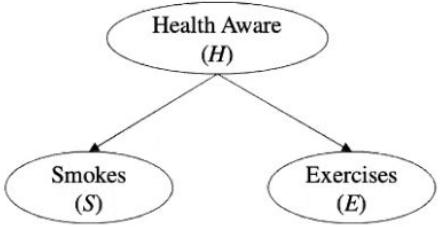
Case	*Cold?*	*Flu?*	*Tonsillitis?*	*Chilling?*	*Bodyache?*	*Sorethroat?*	*Fever?*
1	true	false	?	true	false	false	false
2	false	true	false	true	true	false	true
3	?	?	true	false	?	true	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Complete data (e.g. 2nd case): if **every** row is complete, dataset is complete efficient
- Incomplete data (e.g. 1st case): if **any** row is incomplete, dataset is incomplete
  - Use algorithm such as expectation maximization (EM) to find maximum likelihood parameters
- **BN structure + CPTs = BN**
  - May come out multiple BNs The BN with **max score** (computed by multiplying prob. assigned to all cases based on each BN) is better
  - It's called maximum likelihood principle

Learning Parameters

Goal: Estimate/Learn parameters in CPTs

- E.g. $\Theta_{s|h}$, $\Theta_{\neg s|h}$, $\Theta_{s|\neg h}$, $\Theta_{\neg s|\neg h}$ in the CPT of node S



Case 1: using **complete** data

- Construct empirical distribution table from data
- Compute parameters using conditional probability based on empirical distribution
  - E.g. $\Theta_{s|h} = \Pr(s|h) = \frac{\Pr(s \wedge h)}{\Pr(h)}$

Case	H	S	E
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T
9	T	F	T
10	F	F	T
11	T	F	T
12	T	T	T
13	T	F	T
14	T	T	T
15	T	F	T
16	T	F	T

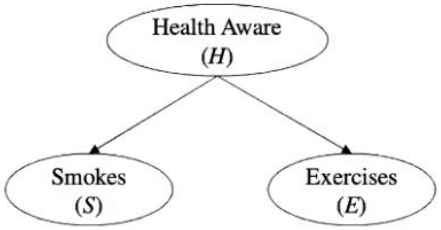
Empirical distribution

H	S	E	$\Pr_{\mathcal{D}}(.)$
T	T	T	2/16
T	T	F	0/16
T	F	T	9/16
T	F	F	1/16
F	T	T	0/16
F	T	F	1/16
F	F	T	2/16
F	F	F	1/16

Learning Parameters

Case 2: using **incomplete** data

- Apply **Expectation Maximization (EM)** algorithm
  1. Use estimate parameters to construct CPTs, BN and compute distribution. If for initialization, randomly choose parameters to get $\mathrm{CPT}_1 \rightarrow \mathrm{BN}_1 \rightarrow \Pr_1(\cdot)$
  2. Complete incomplete data cases/rows with all possible instances, and compute corresponding prob. using the current distribution
     - E.g. $\Pr(E = T | H = T, S = F)$
  3. Build empirical distribution based on prob. in step 2
  4. Estimate parameters using empirical distribution to get updated CPTs, and **iteratively** perform steps 1-4 until the likelihood/score of parameters converge



Case	H	S	E
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T
9	T	F	T
10	F	F	T
11	T	F	T
12	T	T	T
13	T	F	T
14	T	T	T
15	T	F	T
16	T	F	T

Empirical distribution

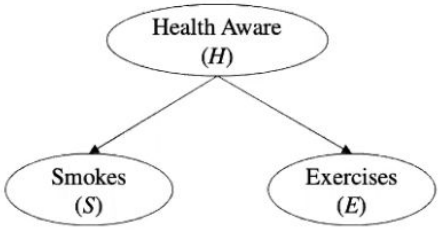
H	S	E	$\Pr_{\mathcal{D}}(.)$
T	T	T	2/16
T	T	F	0/16
T	F	T	9/16
T	F	F	1/16
F	T	T	0/16
F	T	F	1/16
F	F	T	2/16
F	F	F	1/16

Learning Parameters

Case 2: using **incomplete** data

- Apply **Expectation Maximization (EM)** algorithm
- It's guaranteed that the likelihood/score of parameters won't get worse through iterations
  - **Likelihood** is computed by multiplying probabilities of getting all cases
  - Likelihood will either be improved by iterations or converges to a number
- EM method is similar to local search algorithms in terms of initializing at random point and optimize through iterations
- Where EM converges to depends on where it started

 may have multiple maximum likelihood parameters if data is

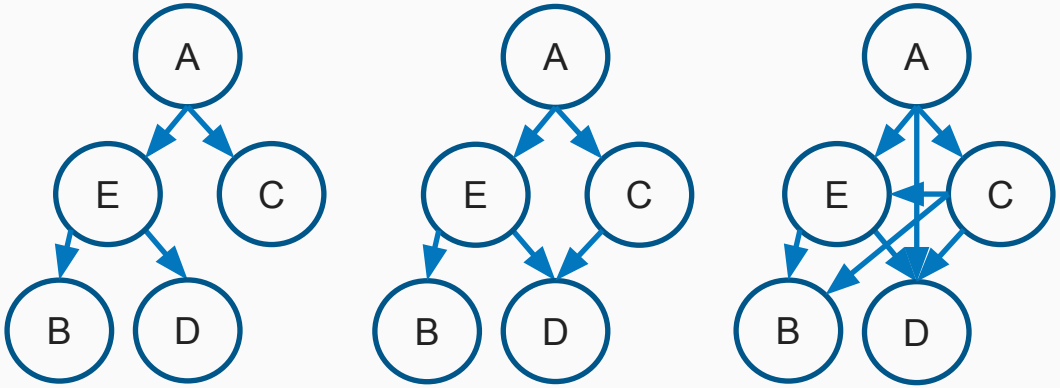


Case	H	S	E
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T
9	T	F	T
10	F	F	T
11	T	F	T
12	T	T	T
13	T	F	T
14	T	T	T
15	T	F	T
16	T	F	T

Empirical distribution

H	S	E	$\Pr_{\mathcal{D}}(.)$
T	T	T	2/16
T	T	F	0/16
T	F	T	9/16
T	F	F	1/16
F	T	T	0/16
F	T	F	1/16
F	F	T	2/16
F	F	F	1/16

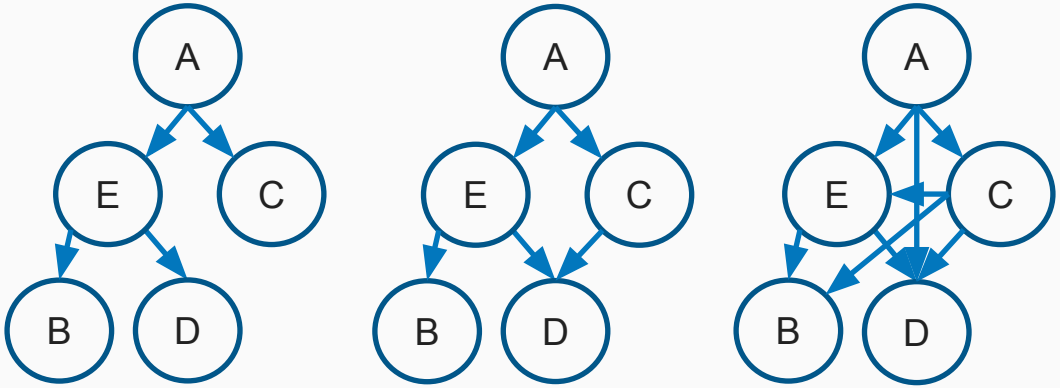
Learning BN Structure

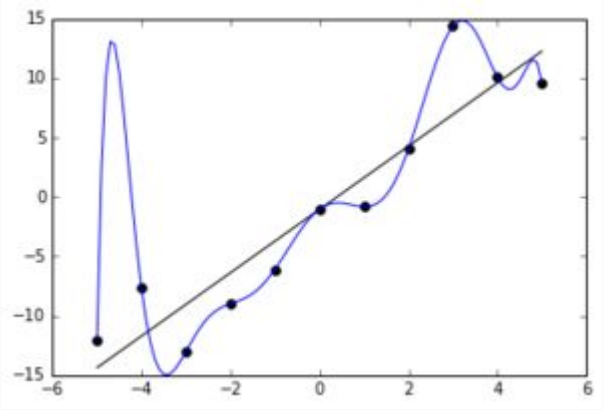


Goal: learn the BN structure given some scoring function

- Methods to find the structure with max. score:
  - Local search methods: add, remove, reverse an edge
    - Approximate method with no guarantee to find optimal structure, but efficient
  - Systematic search methods: A* search
    - Exact method to find optimal structure, but expensive

Learning BN Structure

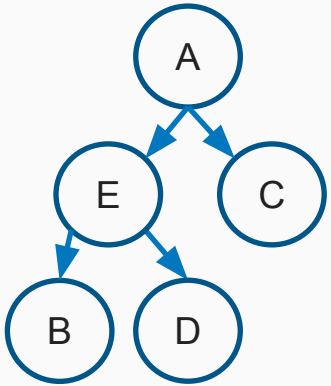




- Maximum likelihood is not enough due to **over-fitting problem**
  - Over-fitting: a statistical model is overfitting if it contains more parameters than can be justified by the data □ very likely fail to fit additional data
  - A penalty term that penalizing the # of parameters in BN is needed to calculate scores
  - A good way is using MDL score, which uses both likelihood and penalty terms

Model-oriented vs. Query-oriented Learning

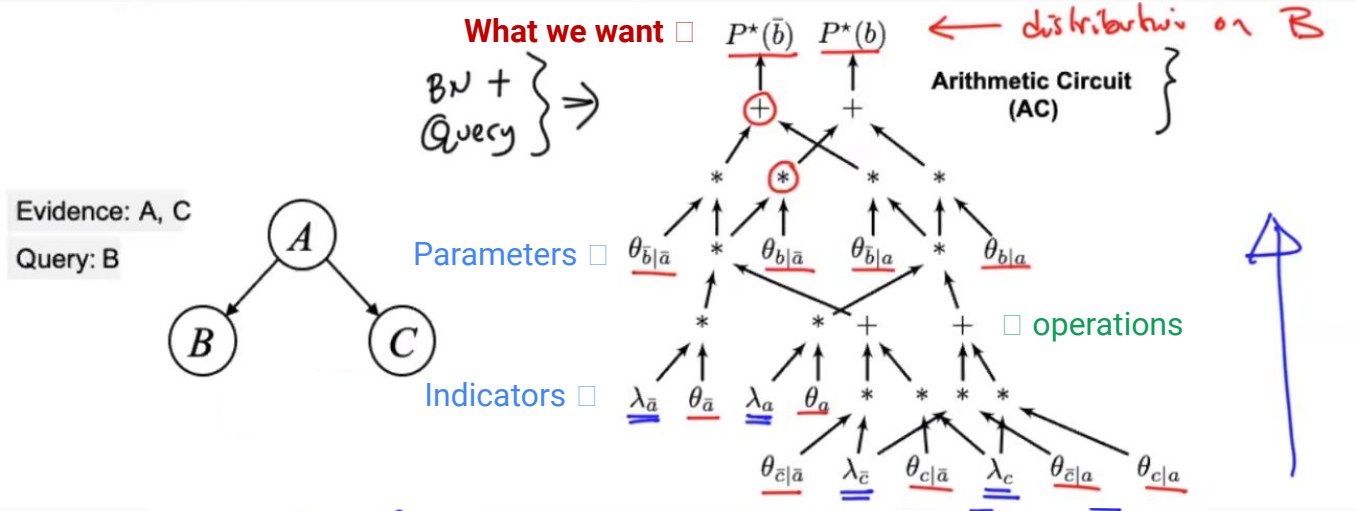
A.k.a. unsupervised vs. supervised learning, or un-labeled vs. labeled learning



Difference:

- Sometimes we want to learn the model to answer all kinds of queries
- Sometimes we only interest in one query, e.g. Pr(A | B, D, C), we need more focused learning optimize performance while answering the query
    - We need different optimization score

Arithmetic Circuit



A	λ_a	$\lambda_{\bar{a}}$
T	1	0
F	0	1
?	1	1

C

→ evidence (input)

A = T, C = F

$\lambda_a = 1$
$\lambda_{\bar{a}} = 0$
$\lambda_c = 0$
$\lambda_{\bar{c}} = 1$

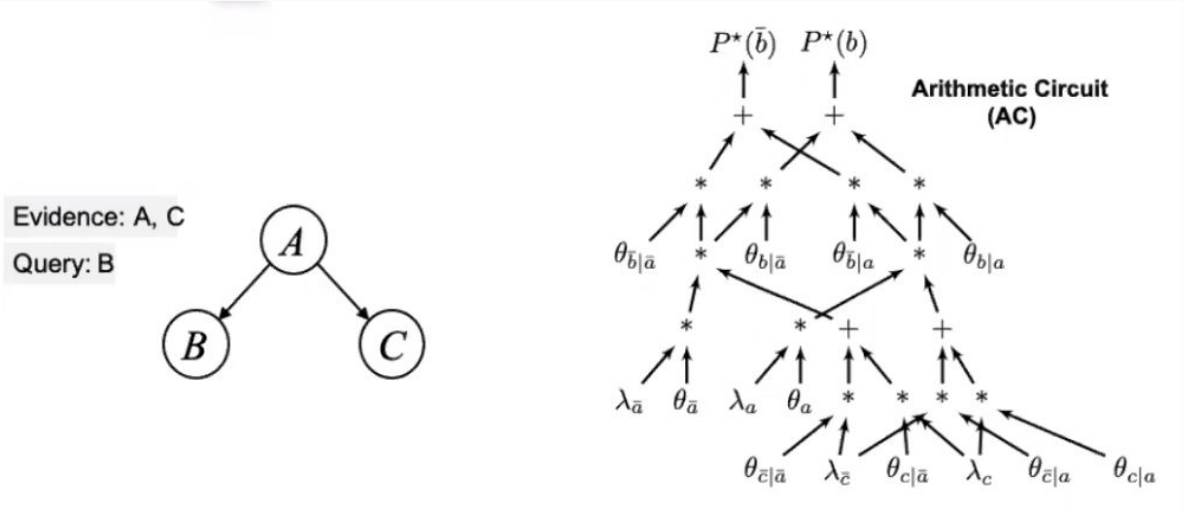
A = F

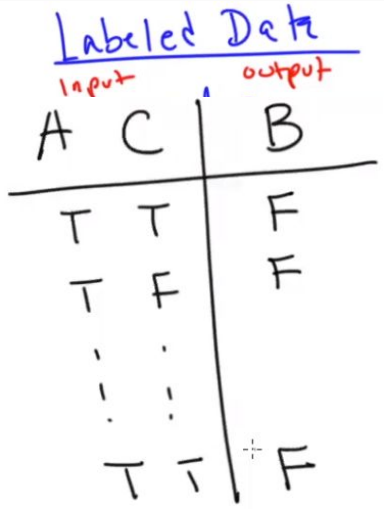
$\lambda_a = 0$
$\lambda_{\bar{a}} = 1$
$\lambda_c = 1$
$\lambda_{\bar{c}} = 1$

Supervised Learning in BN

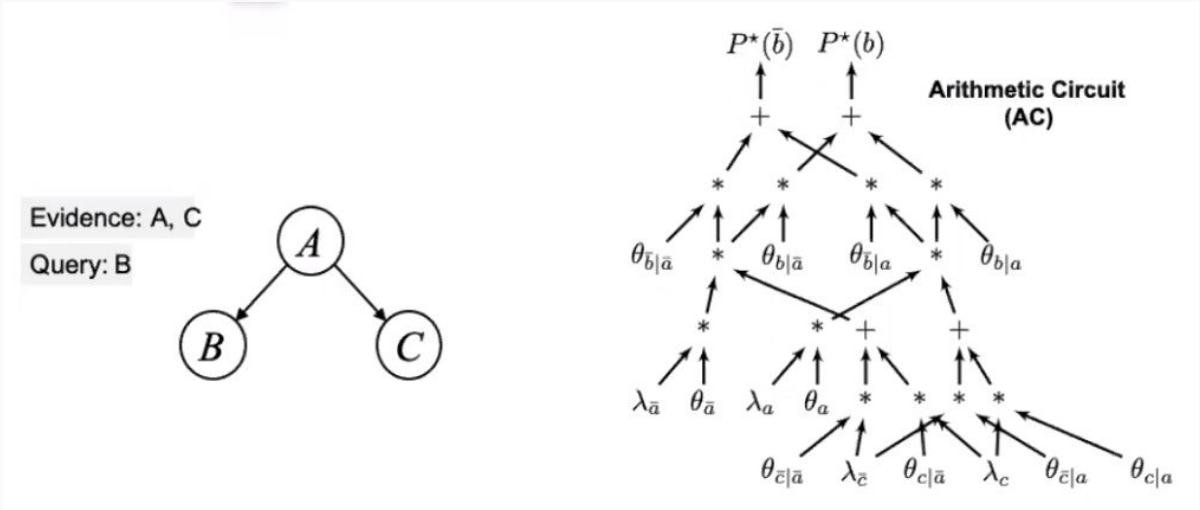
- Arithmetic circuit (AC) can be compiled from a BN in $O(n \cdot d^w)$ time
  - N: # of variables, d: max # of values of each variable, w: treewidth

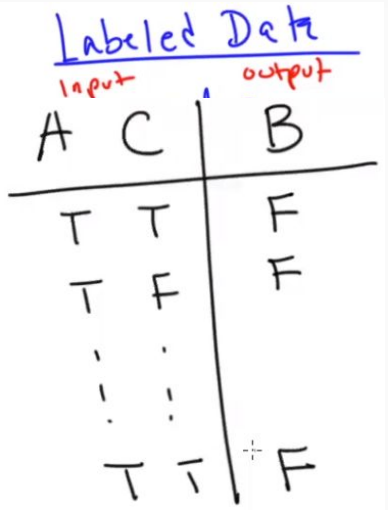
Q: If we don't know the parameters but has labeled data, what should we do?





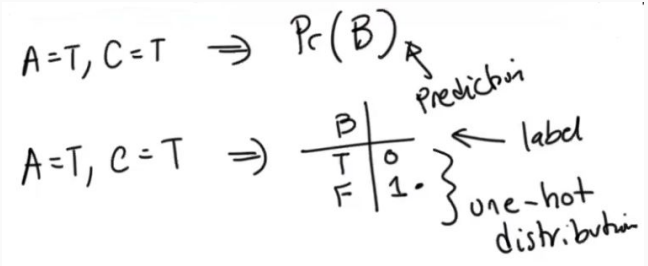
Supervised Learning in BN





Use **loss function** to optimize parameters:

- Minimize the mean cross entropy between <u>prediction in AC</u> and <u>one-hot distribution</u> obtained from labeled data
  - **Cross entropy**: measures how close two distributions are, it takes P(x) and Q(x) and returns a number



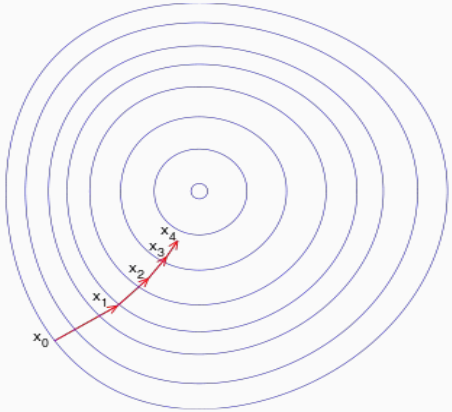
Supervised Learning in BN

Cross entropy (CE) calculation:

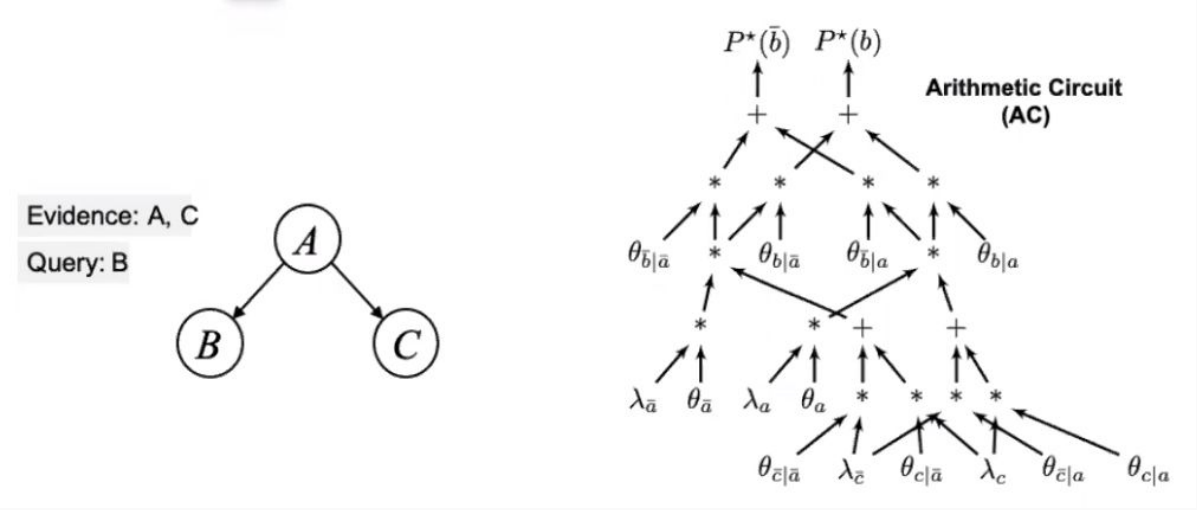
- Given two distributions: P(X) – prediction, Q(X) – label
- CE = $\sum_x Q(x) \log_2(P(x))$ → We try to minimize CE as a loss function using gradient descent

Gradient descent is often used to optimize loss function and find parameters together with neural network.

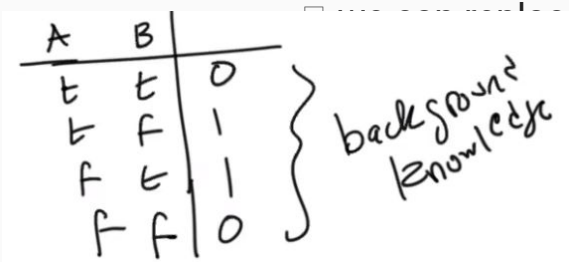
- It improves parameters by computing derivatives of loss function w.r.t those parameters, and move in the direction of the derivatives



Background Knowledge in BN



Assume we know “B = T iff A = F”, we then have background knowledge:



- e parameters in AC with numbers (no need to learn!)
- Background knowledge allows the model to achieve certain accuracy for prediction with less data
- Background knowledge makes the model more robust

Entropy

Entropy can quantify uncertainty:

- Entropy is non-negative, higher entropy means more uncertainty
- **ENT(X)** = $-\sum_x Pr(x)\, log_2(Pr(x))$

E.g.

	Earthquake	**Burglary**	**Alarm**
True	0.1	0.2	0.2442
False	0.9	0.8	0.7558
E	0.469	0.722	0.802

Conditional Entropy

We are interested in ENT(X) and we observe variable Y = y:

- **ENT(X | y)** = $-\sum_x Pr(x|y)\, log_2(Pr(x|y))$

If we want to know the entropy of X when we observe variable Y, but don't know the value of Y yet:

- **ENT(X | Y)** = $\sum_y Pr(y)\, ENT(x|y)$
- ENT(X | Y) ≤ ENT(X) → indicates that we never lose by observing a variable
- ENT(X | Y) = ENT(X) if X and Y are independent

Conditional Entropy - Example

	Burglary	Burglary | A=true	Burglary | A=false
True	0.2	0.741	0.025
False	0.8	0.259	0.975
E	0.722	0.825	0.169

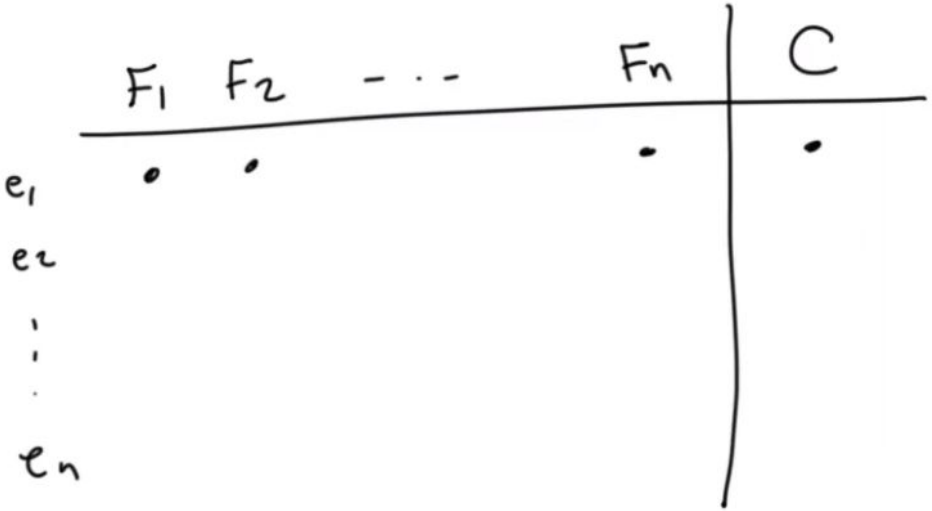
- Unlike ENT(X | Y), ENT(X | y) can be higher or lower than ENT(X).

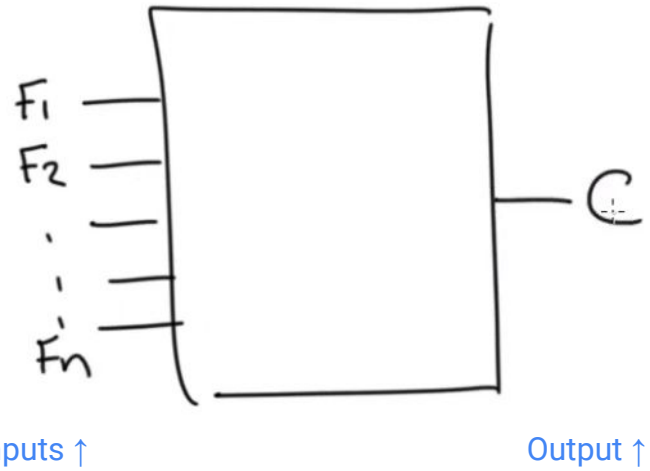
- ENT(Burglary | Alarm) = ENT(B | a) Pr(a) + ENT(B | $\neg$a) Pr($\neg$a)
  = 0.329 $\leq$ 0.722

Classifiers

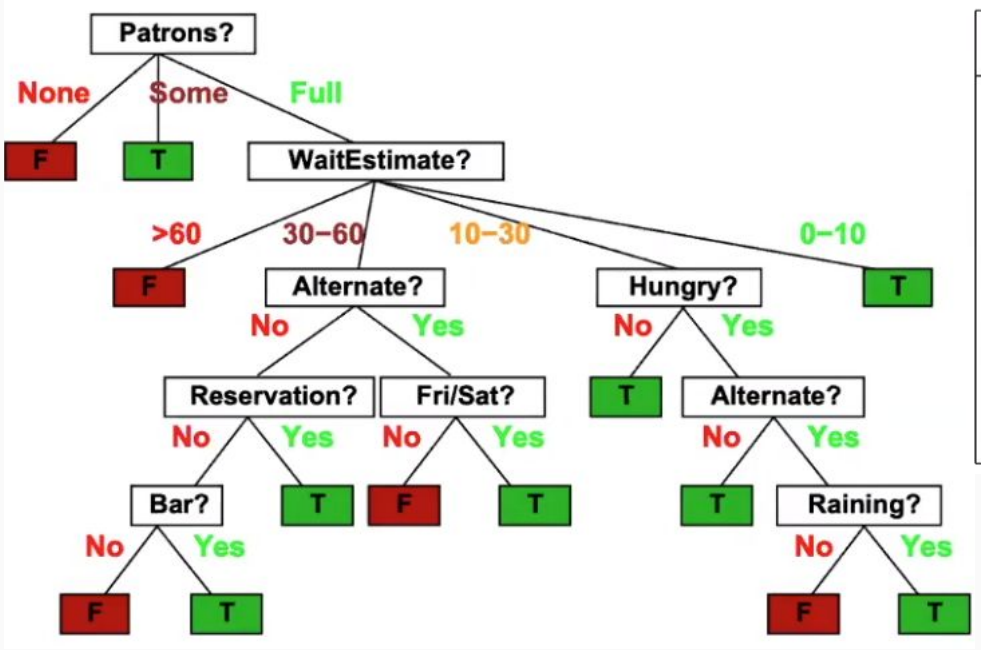
Supervised learning with **labeled** data

- Take features/attributes as input and output class labels
- A binary classifier has only two types of output





Decision Tree - Example



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
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

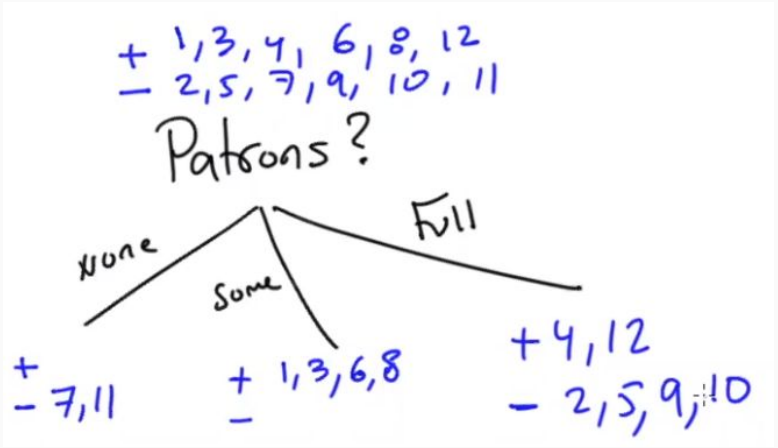
- This decision tree is interpretable
- Classification using decision tree is very simple (e.g. follow attributes to assign a missing label)

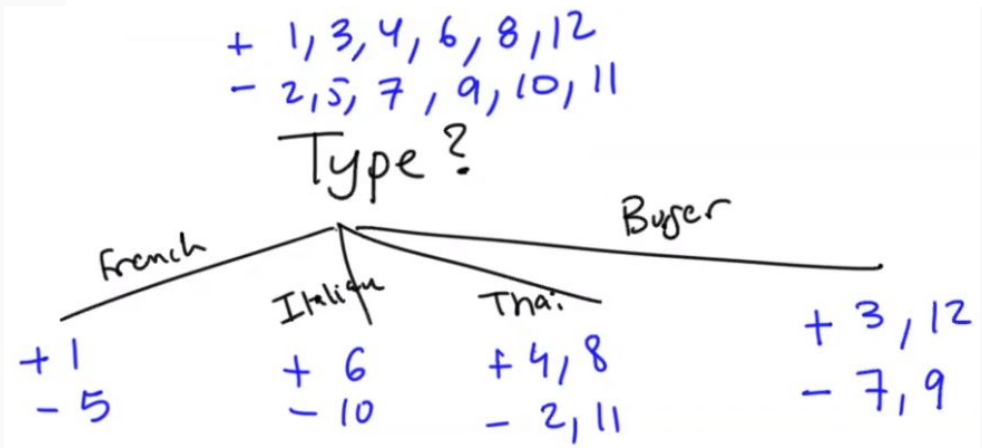
Build a Decision Tree

Key: decide which attribute to split on

- We want to split on the attribute with higher discriminating power
  - E.g. Patrons is better than Type as an attribute to split on

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
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



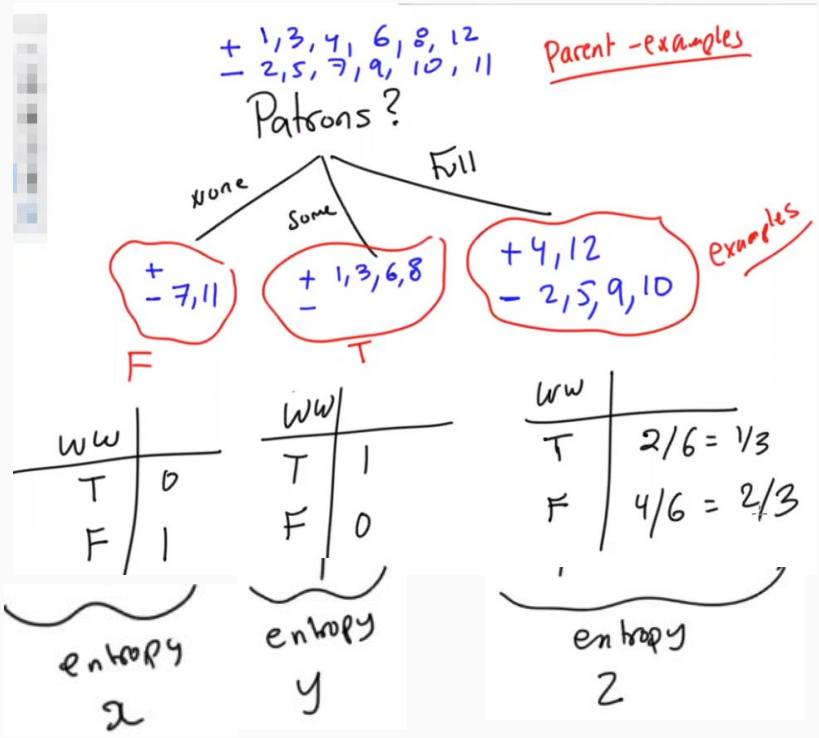


Build a Decision Tree

Steps:

- Consider each of split case as a distribution
- Use **entropy** to quantify the distributions (how certain we are about the class label) after the split
- Compute **conditional entropy** of the output label given current attribute → <u>score of attribute</u>
  - Recall ENT(X | Y) = $\sum_y Pr(y)\, ENT(x|y)$
  - E.g. ENT(WillWait | Patrons)

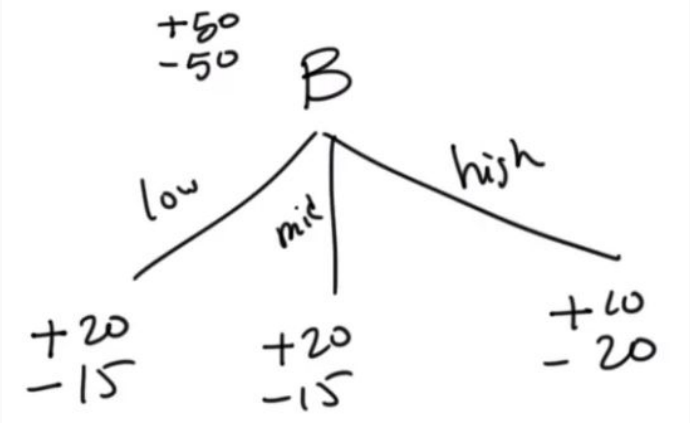
$$\frac{2}{12}x + \frac{4}{12}y + \frac{6}{12}z$$

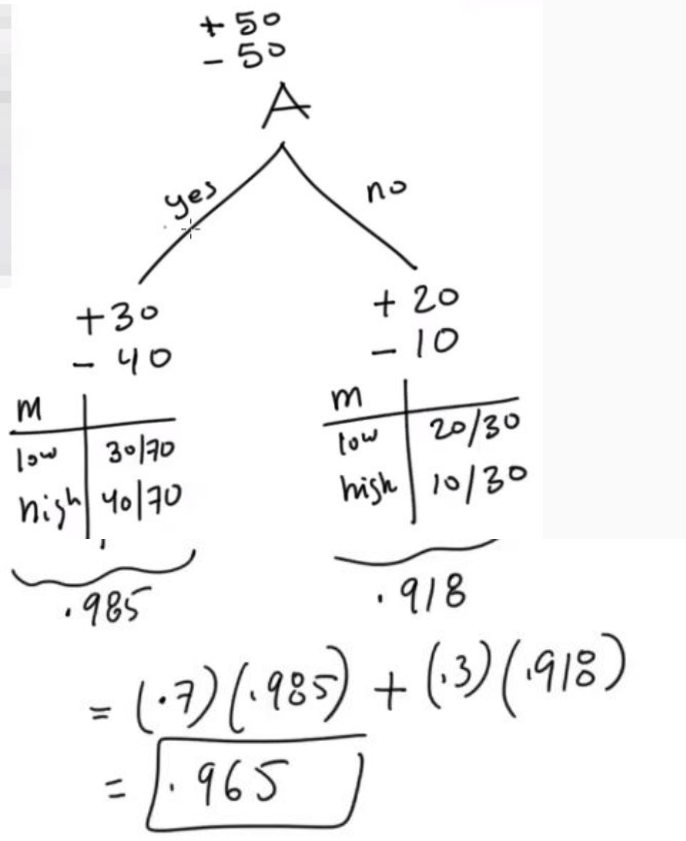


Build a Decision Tree – Example

Let's say we want to output labels of M = {low, high}

We have two attributes to choose – A and B





Algorithm

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) returns a tree

if *examples* is empty then return PLURALITY-VALUE(*parent_examples*)

else if all *examples* have the same classification then return the classification

else if *attributes* is empty then return PLURALITY-VALUE(*examples*)

else

$A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)

tree ← a new decision tree with root test A

for each value v_k of A do

exs ← $\{ e : e \in examples \text{ and } e.A = v_k \}$

subtree ← DECISION-TREE-LEARNING(*exs*, *attributes* − A, *examples*)

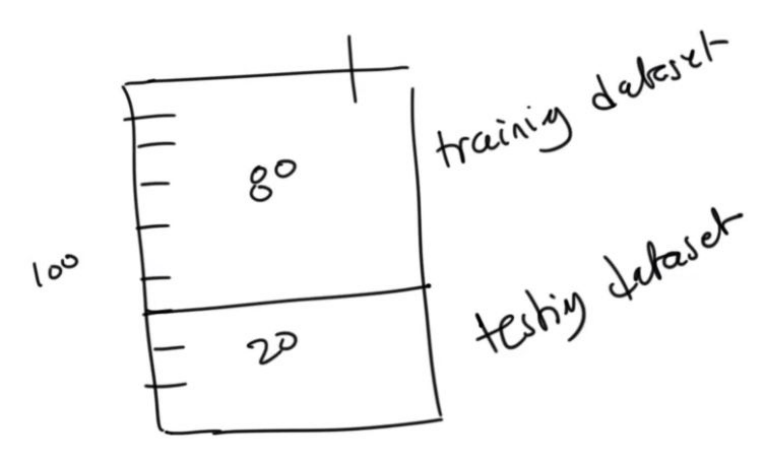
add a branch to *tree* with label ($A = v_k$) and subtree *subtree*

return *tree*

Cross Validation

Idea: split dataset to training and testing portions, build decision trees multiple times using different combinations of training and testing portions, take the average score

- Very useful to deal with over-fitting problem



Random Forests

Random Forests (RF) is an ensemble learning method that combine the results of multiple decision trees

- It does **majority voting** to do classification

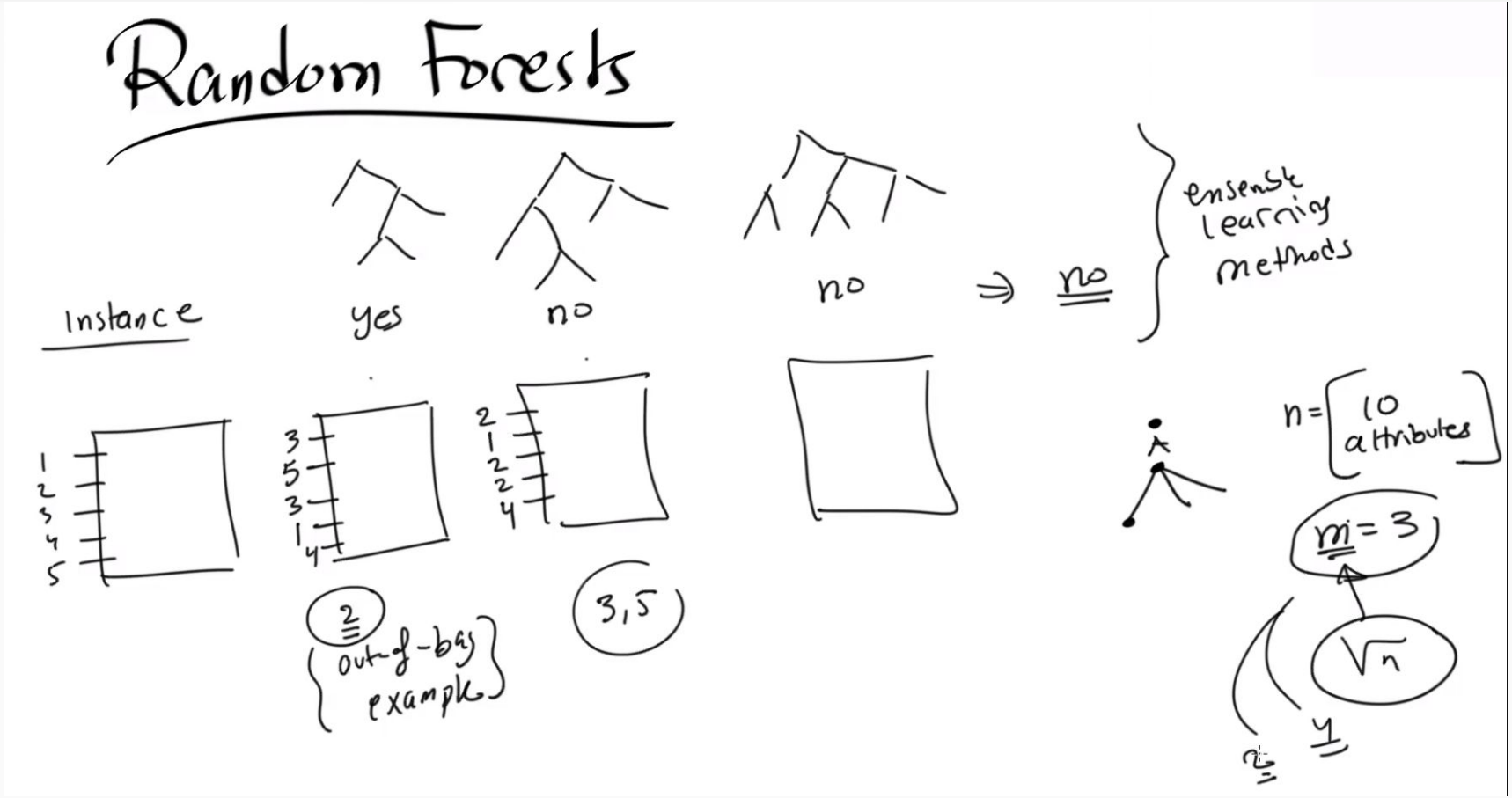
Techniques of construct individual decision trees in RF:

- **Bootstrapping dataset**: randomly sample (with replacement) from the dataset to construct each tree
  - Unselected examples are called out-of-bag examples
- Randomly choose a subset of attributes, then scoring and choose among them

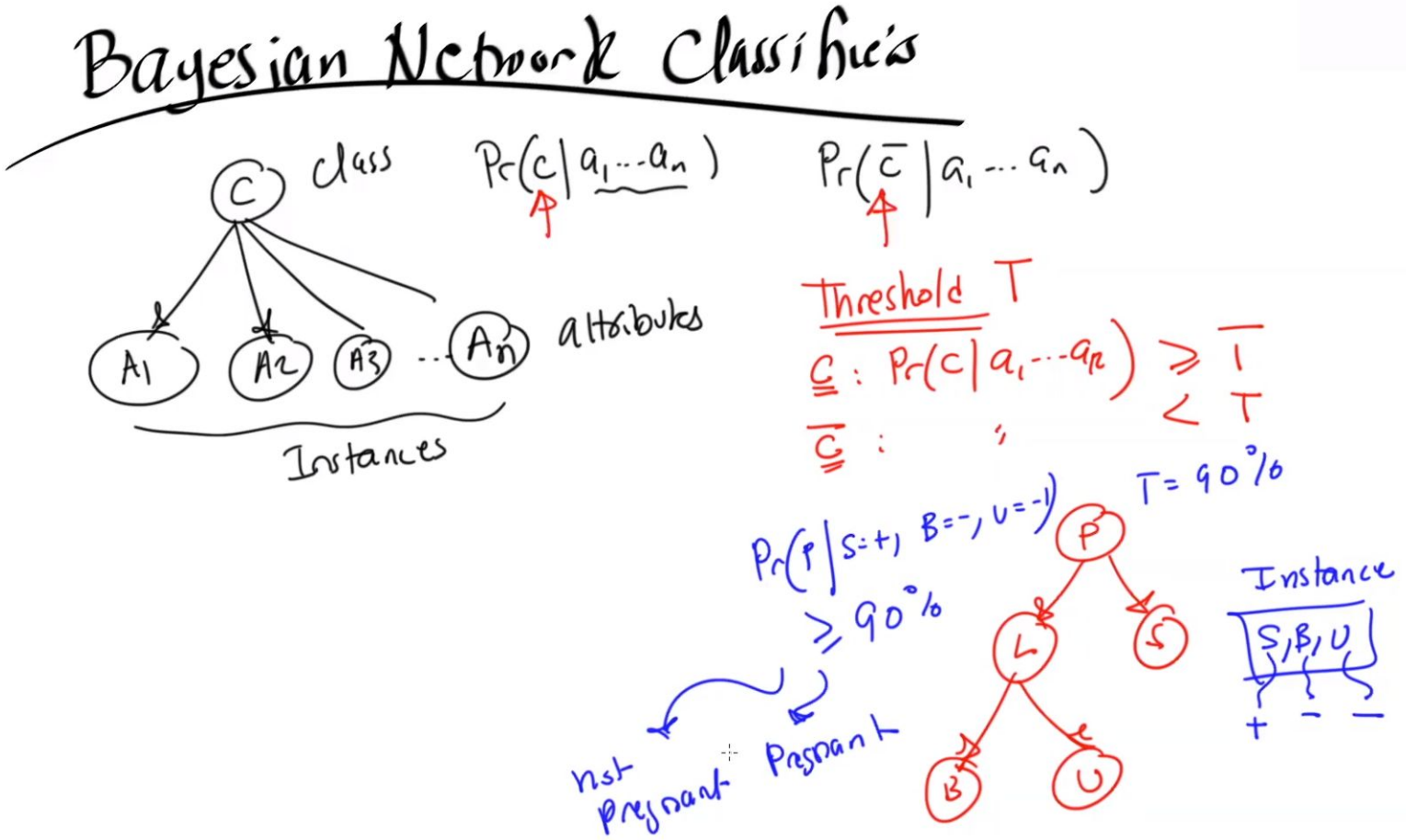
Evaluate the RF:

- Run the constructed RF on out-of-bag examples check how well it performs
- Change parameters (e.g. # of picked attributes) to improve performance

Random Forests – Example



BN Classifier



Hint of HW9

Q1:

- Build a decision tree based on a dataset
- Choose from attributes A, B and C to split based on their scoring
  - Calculate using conditional entropy, e.g. ENT(Class D | A)
  - Choose most discriminating attribute in each step

Example	Input Attributes			Class	#
	A	B	C	D	
x_1	T	T	T	Yes	1
x_2	T	T	F	Yes	6
x_3	T	F	T	No	3
x_4	T	F	F	No	1
x_5	F	T	T	Yes	1
x_6	F	T	F	No	6
x_7	F	F	T	Yes	2
x_8	F	F	F	No	3

Q2:

- Neural network will be explained in next week's lecture