COMP219: what we **definitely** want to know

Dr. Xiaowei Huang

https://cgi.csc.liv.ac.uk/~xiaowei/

Part 1: Basic Knowledge

The supervised learning task

- problem setting
  - set of possible instances: *X*
  - unknown *target function:* $f : X \to Y$
  - set of *models* (a.k.a. *hypotheses*): $H = \{h \mid h : X \to Y\}$

- given *training set* of instances of unknown target function f

$$\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right) \dots \left(\mathbf{x}^{(m)}, y^{(m)}\right)$$

- Output
  - model $h \in H$ that best approximates target function

Unsupervised learning

- in unsupervised learning, we're given a set of instances, without y's

$$\mathbf{x}^{(1)},\mathbf{x}^{(2)} \ldots \mathbf{x}^{(m)}$$

goal: discover interesting regularities/structures/patterns that characterize the instances

- common unsupervised learning tasks
  - *clustering*
  - *anomaly detection*
  - *dimensionality reduction*

Clustering

- given
  - training set of instances $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ ... $\mathbf{x}^{(m)}$

- output
  - model $h \in H$ that divides the training set into clusters such that there is intra-cluster similarity and inter-cluster dissimilarity

Dimensionality reduction

- given
  - training set of instances $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ ... $\mathbf{x}^{(m)}$

- output
  - Model $h \in H$ that represents each x with a lower-dimension feature vector while still preserving key properties of the data

Marginal, joint, conditional probability

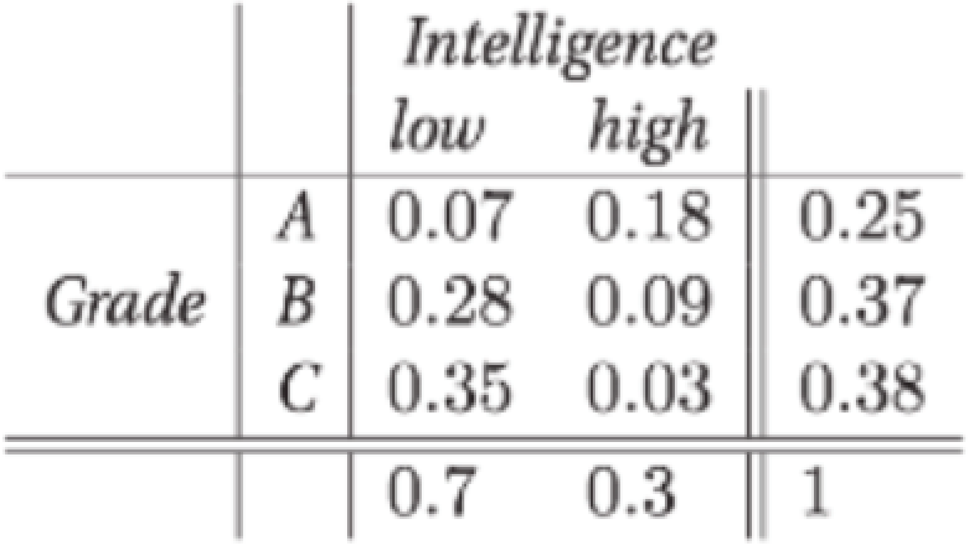
- **Marginal probability**: the probability of an event occurring (p(A)), it may be thought of as an unconditional probability. It is not conditioned on another event.
  - Example: the probability that a card drawn is red (p(red) = 0.5).
  - Another example: the probability that a card drawn is a 4 (p(four)=1/13).

- **Joint probability**: p(A and B). The probability of event A **and** event B occurring. It is the probability of the intersection of two or more events. The probability of the intersection of A and B may be written p(A ∩ B).
  - Example: the probability that a card is a four and red =p(four and red) = 2/52=1/26. (There are two red fours in a deck of 52, the 4 of hearts and the 4 of diamonds).

- **Conditional probability**: p(A|B) is the probability of event A occurring, given that event B occurs.
  - Example: given that you drew a red card, what's the probability that it's a four (p(four|red))=2/26=1/13. So out of the 26 red cards (given a red card), there are two fours so 2/26=1/13.

Conditional Probability

- *P(Intelligence|Grade=A)* describes the distribution over events describable by Intelligence given the knowledge that student's grade is *A*
    - It is not the same as the marginal distribution



		Intelligence		
		low	*high*	
	A	0.07	0.18	0.25
Grade	*B*	0.28	0.09	0.37
	C	0.35	0.03	0.38
		0.7	0.3	1

$P(Intelligence{=}high){=}0.3$

$P(Intelligence{=}high|Grade{=}A)$

$={0.18}/{0.25}$

$=0.72$

Chain Rules

chain rule (also called the **general product rule**[1][2]) permits the calculation of any member of the joint distribution of a set of random variables using only conditional probabilities.

$$\mathrm{P}(A_n, \ldots, A_1) = \mathrm{P}(A_n | A_{n-1}, \ldots, A_1) \cdot \mathrm{P}(A_{n-1}, \ldots, A_1)$$

$$\mathrm{P}(A_4, A_3, A_2, A_1) = \mathrm{P}(A_4 \mid A_3, A_2, A_1) \cdot \mathrm{P}(A_3 \mid A_2, A_1) \cdot \mathrm{P}(A_2 \mid A_1) \cdot \mathrm{P}(A_1)$$

Max vs. argmax

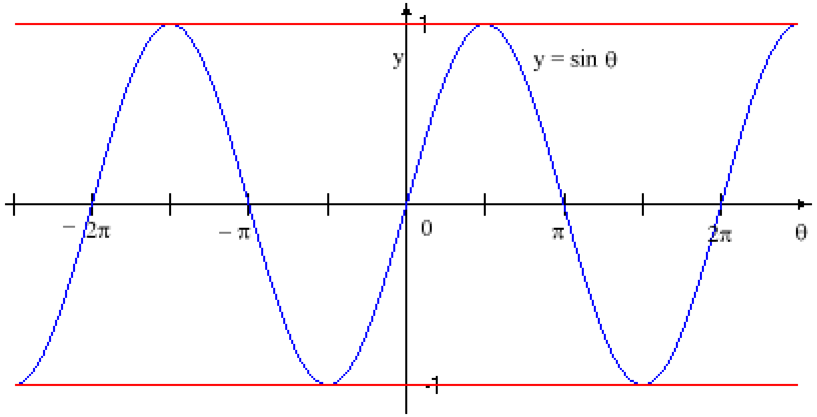
- Let x be in a range [a,b] and f be a function over [a,b], we have
  - max f(x) to represent the maximum value of f(x) as x varies through [a,b]
  - argmax f(x) to represent the value of x at which the maximum is attained

- max$_x$ sin(x)

  = 1
- argmax$_x$ sin(x)

  = {(0.5+2n)*pi | n is integer }

  = {…, -1.5pi, 0.5pi, 2.5pi, …}



Query Types

- Probability Queries
  - Given evidence (the values of a subset of random variables),
  - compute distribution of another subset of random variables
- MAP Queries
  - Maximum a posteriori probability
  - Also called MPE (*Most Probable Explanation*)
    - What is the most likely setting of a subset of random variables
  - Marginal MAP Queries
    - When some variables are known

MAP Queries (Most Probable Explanation)

- Finding a high probability assignment to some subset of variables
- Most likely assignment to all non-evidence variables *W* = V – *E*

$$MAP(W \mid e) = \arg\max_{w} P(w, e)$$

i.e., value of *w* for which *P(w,e)* is maximum

- Difference from probability query
  - Instead of a probability we get the most likely value for all remaining variables

Example of MAP Queries

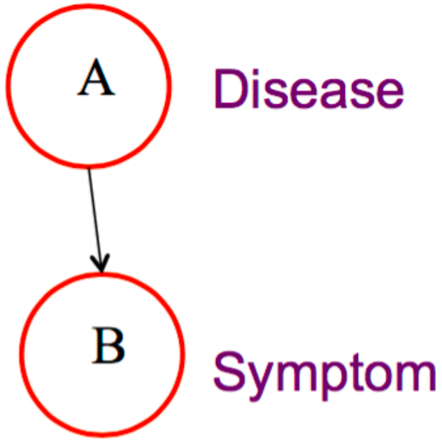
- Medical Diagnosis Problem
  - Diseases (*A*) cause Symptoms (*B*)
  - Two possible diseases: Mono and Flu
  - Two possible symptoms: Headache and Fever

- Q2: Most likely disease and symptom *P(A,B)*?

$$
\begin{aligned}
MAP(A,B) &= \arg\max\nolimits_{a,b} P(A,B) \\
&= \arg\max\nolimits_{a,b} P(B \mid A) P(A) \\
&= \arg\max\nolimits_{a,b} \{0.04, 0.36, 0.3, 0.3\} \\
&= a^0, b^1
\end{aligned}
$$

P(Diseases)

a^0	a^1
0,4	0.6



P(Symptom|Disease)

P(B|A)	b^0	b^1
a^0	0.1	0.9
a^1	0.5	0.5

L^P Norm

- Definition $||x||_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$
- L^2 Norm
  - Called Euclidean norm, written simply as $||x||$
  - Squared Euclidean norm is same as $x^T x$

$$||x||_2 = \sqrt{\sum_i |x_i|^2}$$

$$= \sqrt{x^T x}$$

L^P Norm

- Definition $||x||_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$
- L^1 Norm
  - also called Manhattan distance

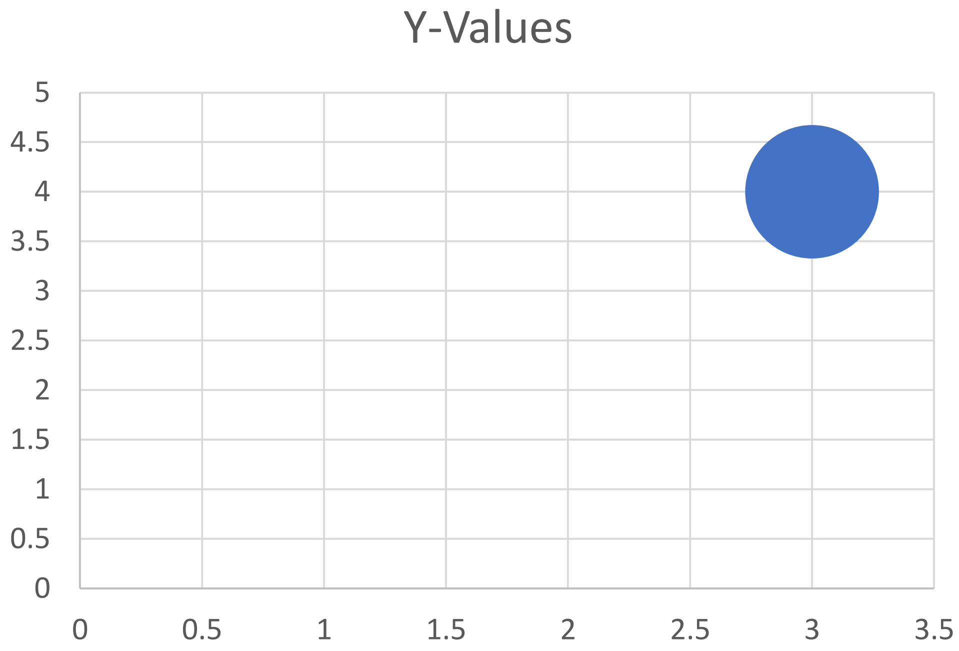
$$||x||_1 \quad = \quad \sum_i |x_i|$$

L^P Norm

- Definition $||x||_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$
- L^∞ Norm
  - also called max norm

$$||x||_\infty = \max_i |x_i|$$

Norms of two-dimensional Point



X = (3,4)

$||x||_1$ = 3+4 = 5

$$||x||_1 = \sum_i |x_i|$$

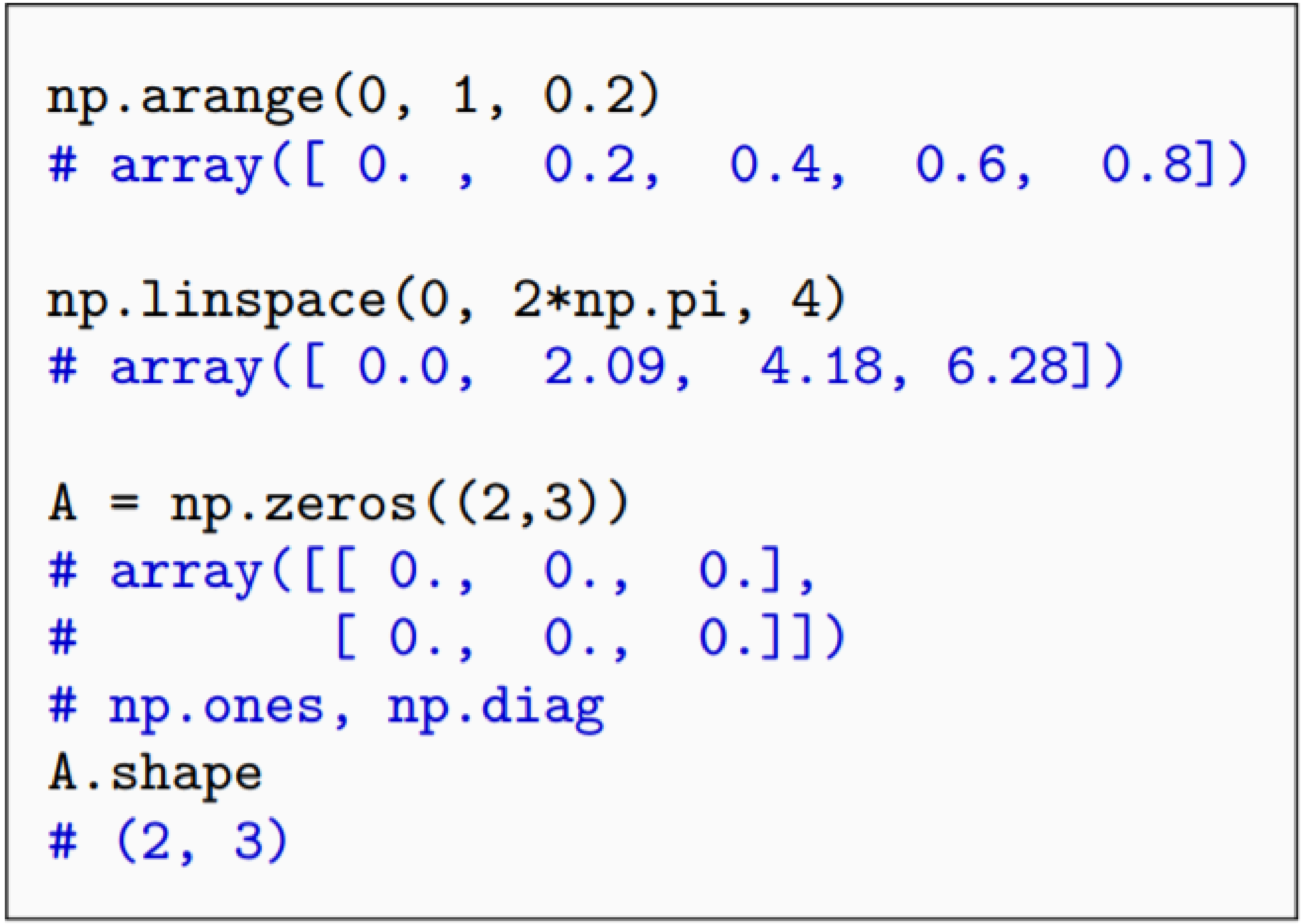
$||x||_2$ = $\sqrt{3^2 + 4^2} = 5$

$$||x||_2 = \sqrt{\sum_i |x_i|^2}$$

$||x||_{\infty}$ = $\max\{3, 4\} = 4$

$$||x||_{\infty} = \max_i |x_i|$$

Basics of Numpy



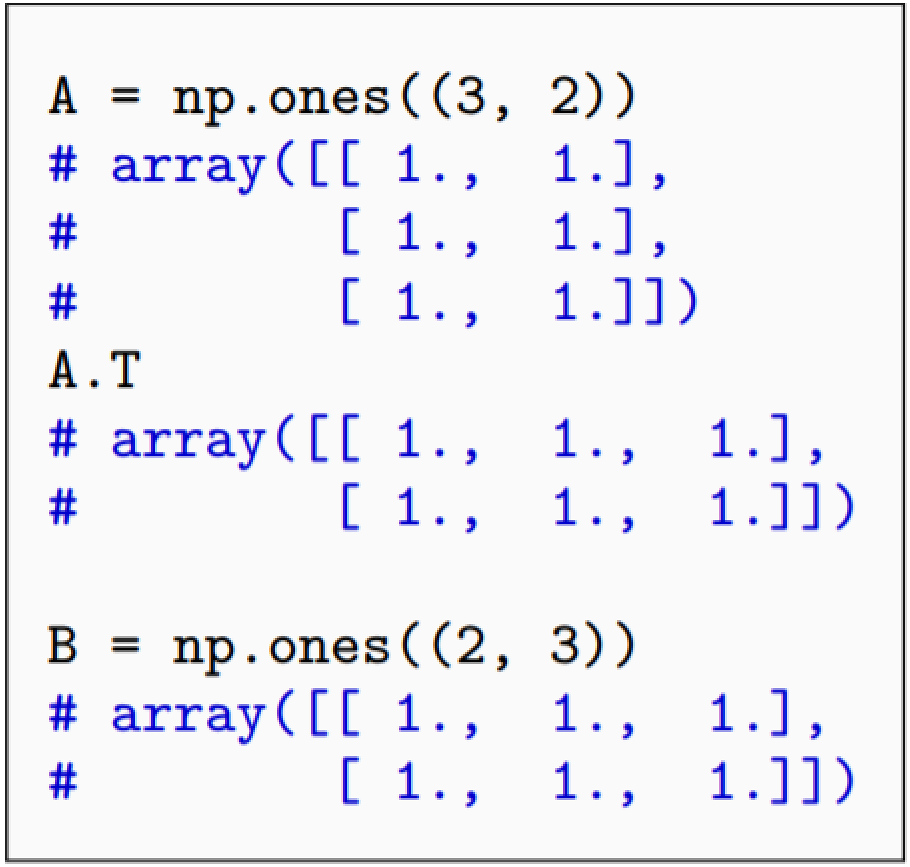
```
np.arange(0, 1, 0.2)
# array([ 0. ,  0.2,  0.4,  0.6,  0.8])

np.linspace(0, 2*np.pi, 4)
# array([ 0.0,  2.09,  4.18, 6.28])

A = np.zeros((2,3))
# array([[ 0.,  0.,  0.],
#         [ 0.,  0.,  0.]])
# np.ones, np.diag
A.shape
# (2, 3)
```

numpy.arange:
evenly spaced values within a given interval.

numpy.linspace:
evenly spaced numbers over a specified interval.

Matrix operations

- First, define some matrices:



```
A = np.ones((3, 2))
# array([[ 1.,  1.],
#         [ 1.,  1.],
#         [ 1.,  1.]])
A.T
# array([[ 1.,  1.,  1.],
#         [ 1.,  1.,  1.]])

B = np.ones((2, 3))
# array([[ 1.,  1.,  1.],
#         [ 1.,  1.,  1.]])
```

Part 2: Simple Learning Models

Decision tree exercise

- Suppose X_1 ... X_5 are Boolean features, and Y is also Boolean
- How would you represent the following with decision trees?

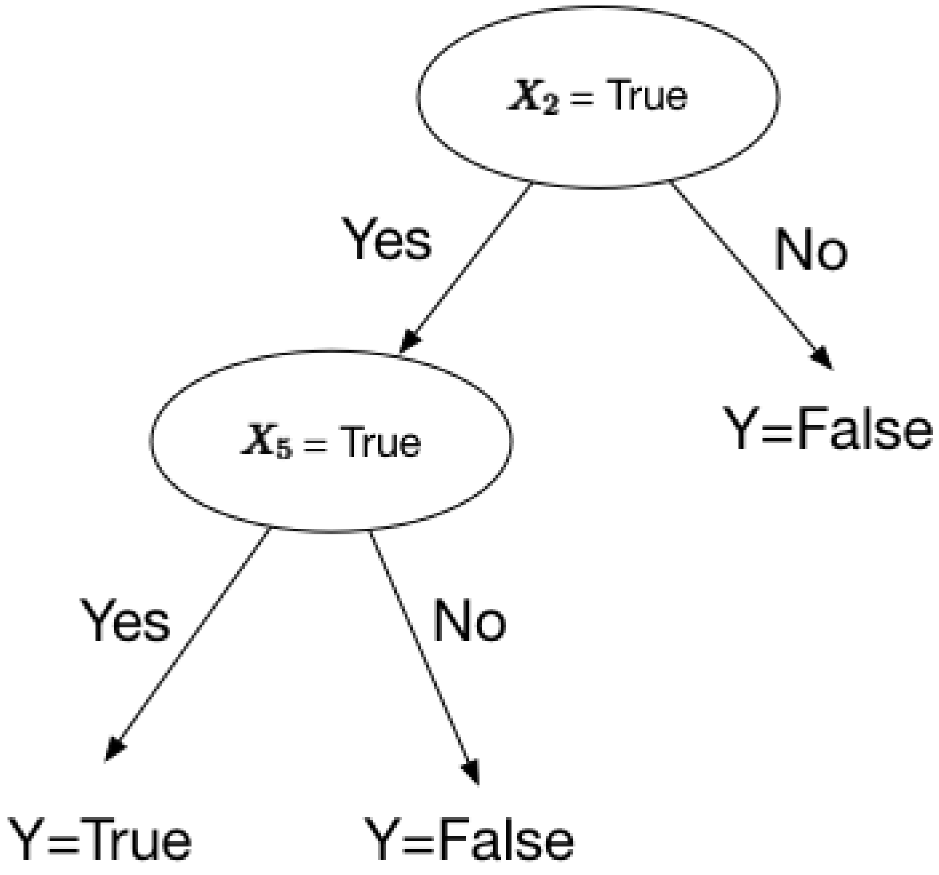
$$Y = X_2 X_5 \quad (\text{i.e.}, Y = X_2 \wedge X_5)$$

$$Y = X_2 \vee X_5$$

$$Y = X_2 X_5 \vee X_3 \neg X_1$$

Decision tree exercise

$$Y = X_2X_5$$

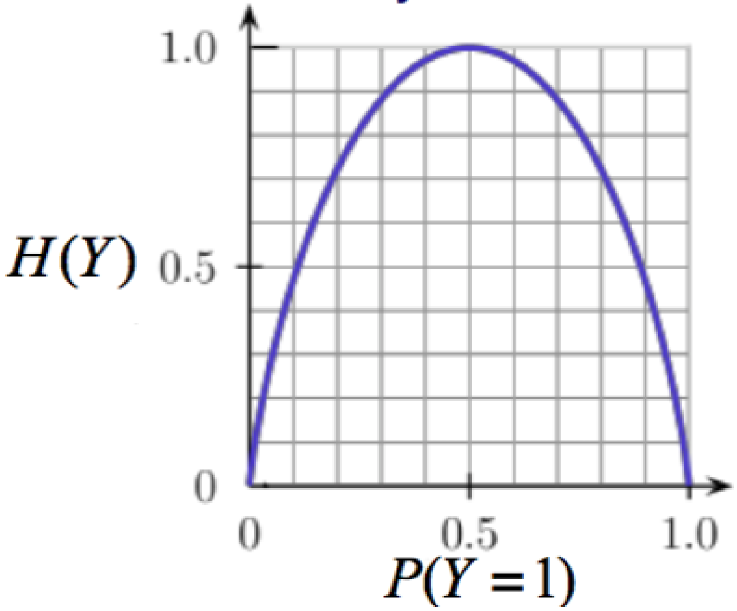


Entropy

- entropy is a measure of uncertainty associated with a random variable
- defined as the expected number of bits required to communicate the value of the variable

$$H(Y) = -\sum_{y \in \text{values}(Y)} P(y) \log_2 P(y)$$

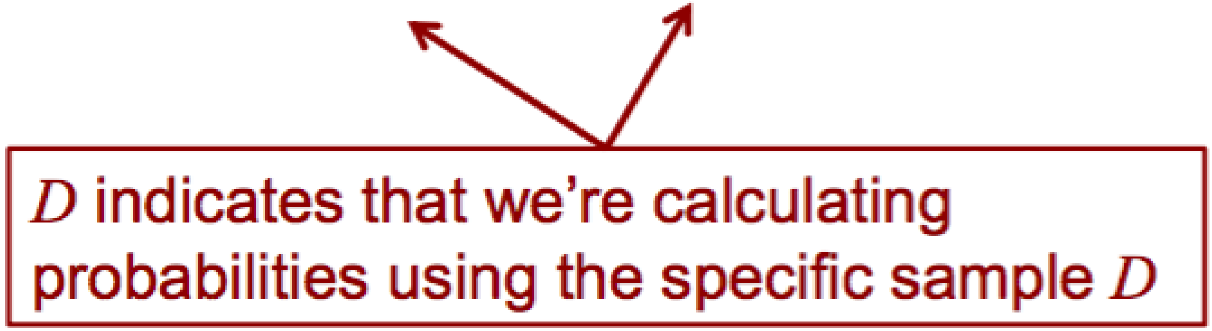
entropy function for binary variable



Information gain (a.k.a. mutual information)

- choosing splits in ID3: select the split S that most reduces the conditional entropy of Y for training set D

$$\text{InfoGain}(D,S) = H_D(Y) - H_D(Y \mid S)$$



D indicates that we're calculating probabilities using the specific sample *D*

How can we determine similarity/distance

- Example: X = (Height, Weight, RunningSpeed) Y = SoccerPlayer?
  - D: in the table
  - New instance: <185, 91, 13.0>
  - Suppose that Euclidean distance is used.
  - Is this person a soccer player?

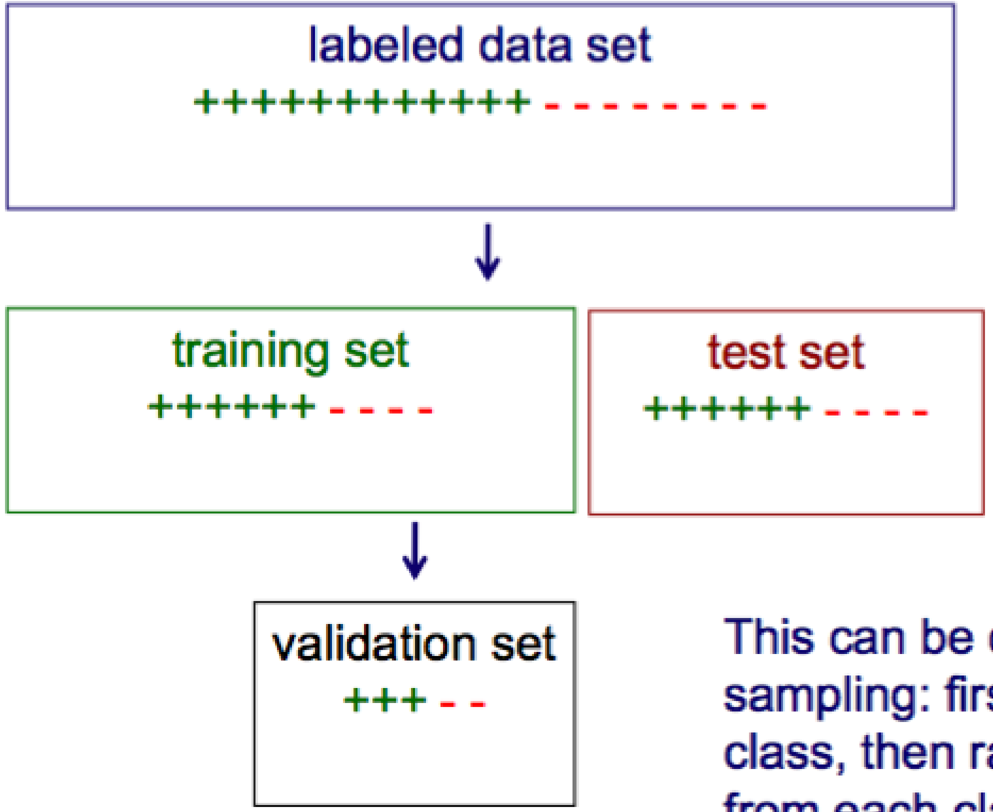
v1	v2	v3	y
182	87	11.3	No
189	92	12.3	Yes
178	79	10.6	Yes
183	90	12.7	No

New datum

185	**91**	**13.0**	

Stratified sampling

- When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set

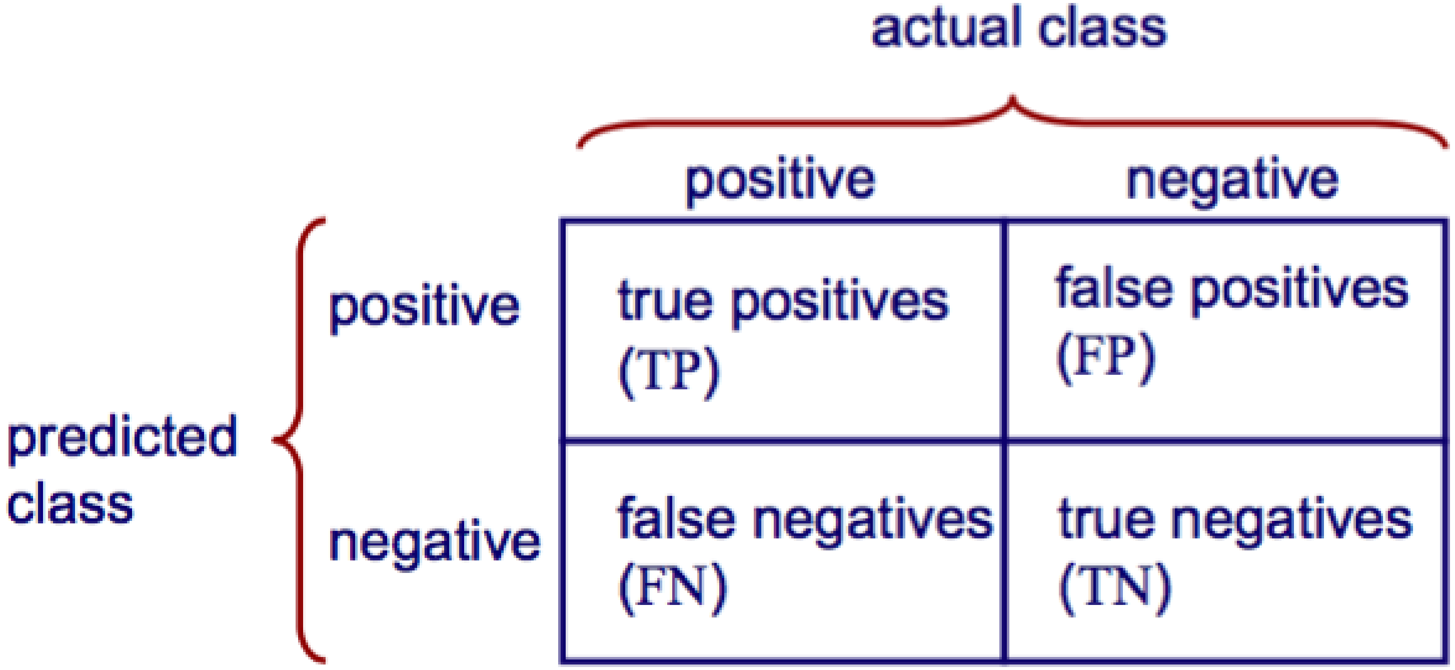


This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

Recall: a *validation set* (a.k.a. *tuning set*) is a subset of the training set that is held aside

Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset

Confusion matrix for 2-class problems



		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{error} = 1 - \text{accuracy} = \frac{FP + FN}{TP + FP + FN + TN}$$

k-nearest-neighbor classification

- classification task
  - **given**: an instance $x^{(q)}$ to classify
  - find the k training-set instances ($\mathbf{x}^{(1)}$, $y^{(1)}$)... ($x^{(k)}$, $y^{(k)}$) that are the most similar to $x^{(q)}$
  - return the class value

$$\hat{y} \leftarrow \underset{v \in \text{values}(Y)}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, y^{(i)}) \qquad \delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

  - (i.e. return the class that have the most instances)

k-nearest-neighbor *regression*

- learning stage
  - given a training set ($\mathbf{x}^{(1)}$, $y^{(1)}$) ... ($\mathbf{x}^{(m)}$, $y^{(m)}$), do nothing
    - (it's sometimes called a *lazy learner*)

- classification stage
    - **given**: an instance $x^{(q)}$ to classify
    - find the k training-set instances ($\mathbf{x}^{(1)}$, $y^{(1)}$)... ($\mathbf{x}^{(k)}$, $y^{(k)}$) that are most similar to $x^{(q)}$
    - return the value

$$\hat{y} \leftarrow \frac{1}{k}\sum_{i=1}^{k} y^{(i)}$$

Linear regression

Hypothesis Class H

- Given training data $\{(x^{(i)}, y^{(i)}) : 1 \leq i \leq m\}$ i.i.d. from distribution D
- Find $f_w(x) = w^T x$ that minimises

L^2 loss, or mean square error

$$\hat{L}(f_w) = \frac{1}{m} \sum_{i=1}^{m} (w^T x^{(i)} - y^{(i)})^2$$

- where
  - $w^T x^{(i)} - y^{(i)}$ represents the error of instance $x^{(i)}$
  - $\sum_{i=1}^{m} (w^T x^{(i)} - y^{(i)})^2$ represents the square error of all training instances

So, $\frac{1}{m} \sum_{i=1}^{m} (w^T x^{(i)} - y^{(i)})^2$ represents the mean square error of all training instances

Functions with multiple inputs

- Gradient is vector containing all of the partial derivatives denoted

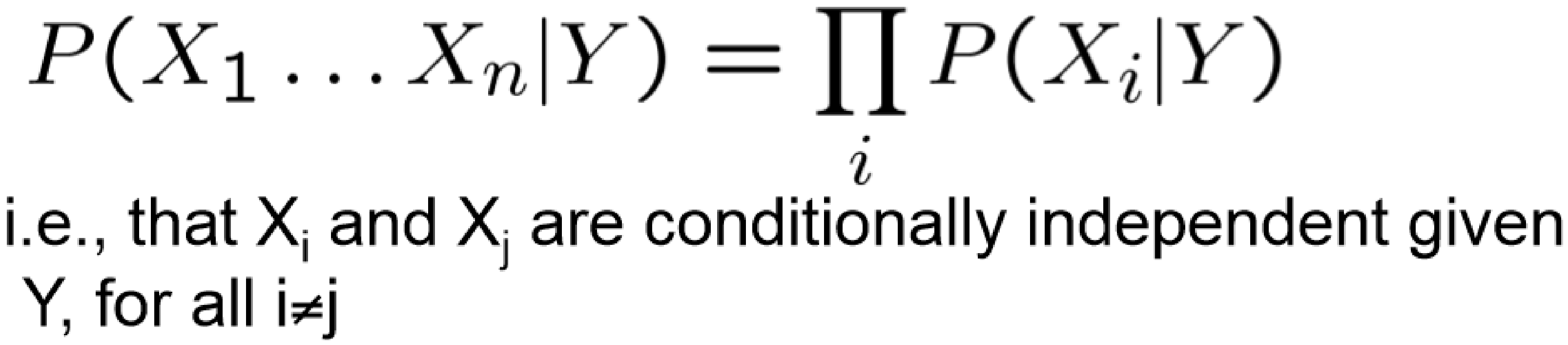
$$\nabla_x f(x) = (\frac{\partial}{\partial x_1} f(x), ..., \frac{\partial}{\partial x_n} f(x))$$

  - Element *i* of the gradient is the partial derivative of *f* wrt x_i
- Critical points are where every element of the gradient is equal to zero

$$\nabla_x f(x) = 0 \equiv \begin{cases} \frac{\partial}{\partial x_1} f(x) = 0 \\ ... \\ \frac{\partial}{\partial x_n} f(x) = 0 \end{cases}$$

Naïve Bayes

- Naïve Bayes assumes



$$P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)$$

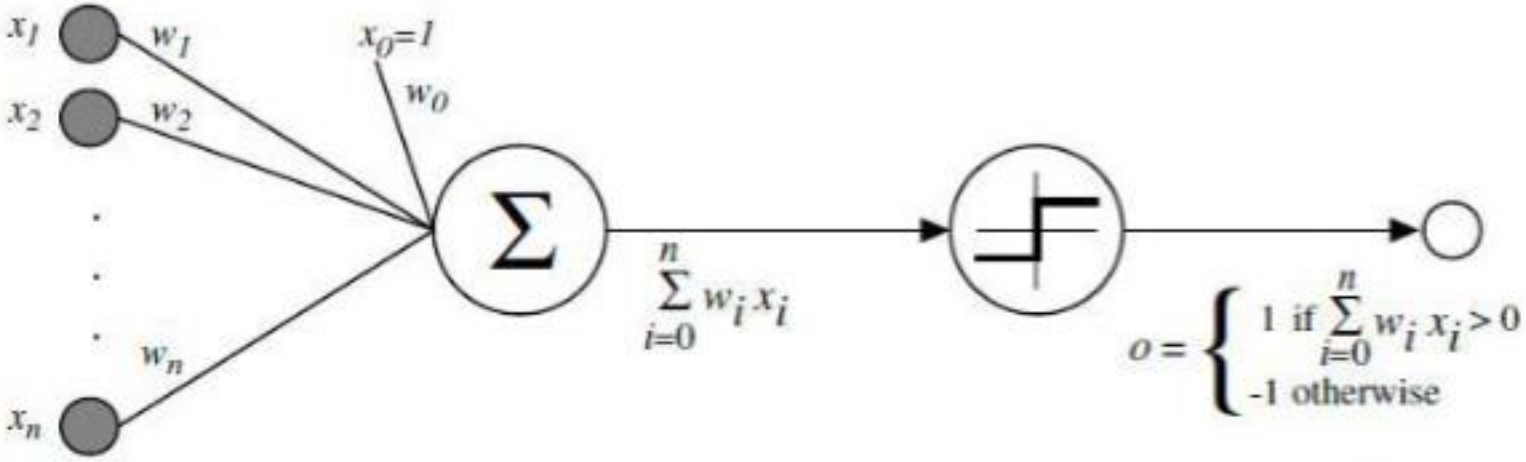
i.e., that X_i and X_j are conditionally independent given Y, for all $i \neq j$

Part 3: Deep Learning

Perceptrons

- Rosenblatt proposed a machine for binary classifications
- Main idea
  - One weight w_i per input x_i
  - Multiply weights with respective inputs and add bias x_0 =+1
  - If result larger than threshold return 1, otherwise 0

$w^T x + b$?



x_1
w_1
x_2
w_2
$x_0=1$
w_0
Σ
w_n
x_n
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

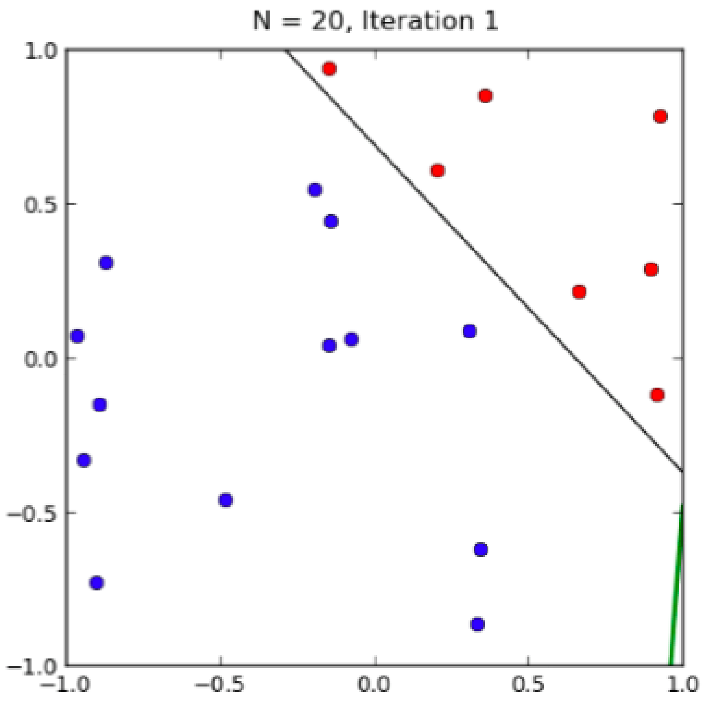
Training a perceptron

- Rosenblatt’s innovation was mainly the learning algorithm for perceptrons
- Learning algorithm
  - Initialize weights randomly
  - Take one sample x_i and predict y_i
  - For erroneous predictions update weights
    - If the output was $\hat{y}_i = 0$ and y_i = 1, increase weights
    - If the output was $\hat{y}_i = 1$ and y_i = 0, decrease weights

$$\Delta w_i = \eta(y - \hat{y})x_i \qquad w_i \leftarrow w_i + \Delta w_i$$

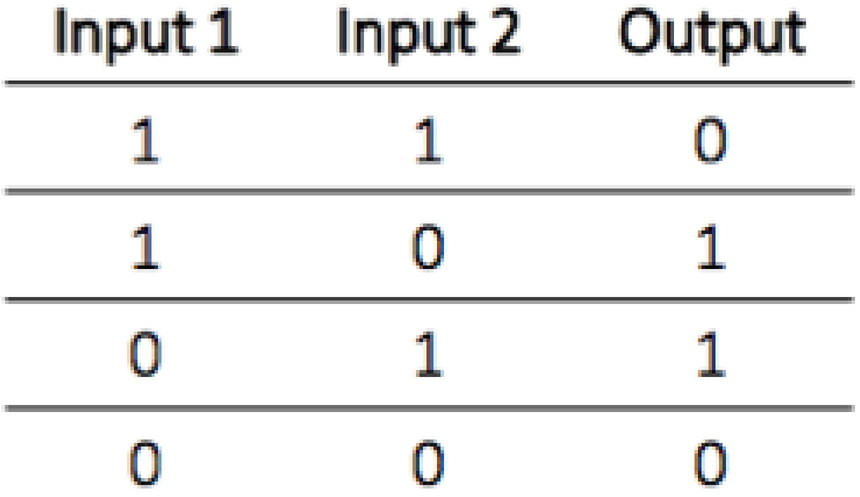
Repeat until no errors are made

η is learning rate; set to value << 1

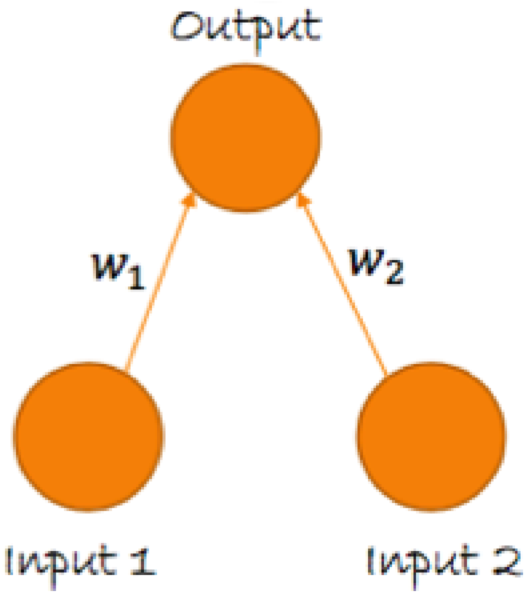


XOR & Multi-layer Perceptrons

- However, the exclusive or (XOR) cannot be solved by perceptrons
  - [Minsky and Papert, “Perceptrons”, 1969]



Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0

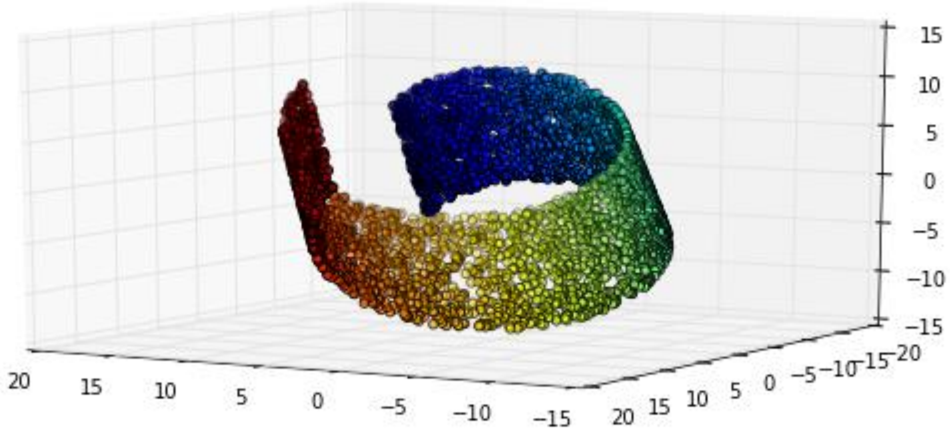


So, why now?

- Better hardware
- Bigger data
- Better regularization methods, such as dropout
- Better optimization methods, such as Adam, batch normalization

How to get good features?

- High-dimensional data (e.g. faces) lie in lower dimensional manifolds



Every point represents an input sample.

- This is so-called "swiss roll". The data points are in 3d, but they all lie on 2d manifold, so the dimensionality of the manifold is 2, while the dimensionality of the input space is 3.

How to get good features?

- High-dimensional data (e.g. faces) lie in lower dimensional manifolds
- Although the data points may consist of thousands of features, they may be described as a function of only a few underlying parameters.
  - That is, the data points are actually samples from a low-dimensional manifold that is embedded in a high-dimensional space.
- Goal: discover these lower dimensional manifolds
  - These manifolds are most probably highly non-linear

How to get good features?

- High-dimensional data (e.g. faces) lie in lower dimensional manifolds
  - Goal: discover these lower dimensional manifolds
  - These manifolds are most probably highly non-linear
- Hypothesis (1): Compute the coordinates of the input (e.g. a face image) to this non-linear manifold -> data become separable
- Hypothesis (2): Semantically similar things lie closer together than semantically dissimilar things

Chain rule

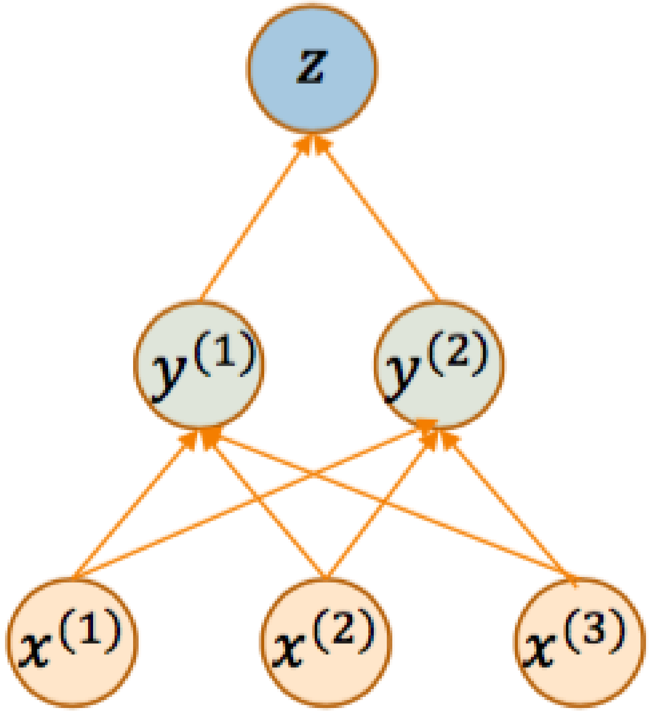
- Assume a nested function, $z = f(y)$ and y= g(x)
- Chain Rule for scalars x, y, z

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

- When $x \in$ R^m,$y \in$ R^n. $z \in$R

$$\frac{dz}{dx_i} = \sum_j \frac{dz}{dy_j}\frac{dy_j}{dx_i}$$

  - i.e., gradients from all possible paths



Chain rule

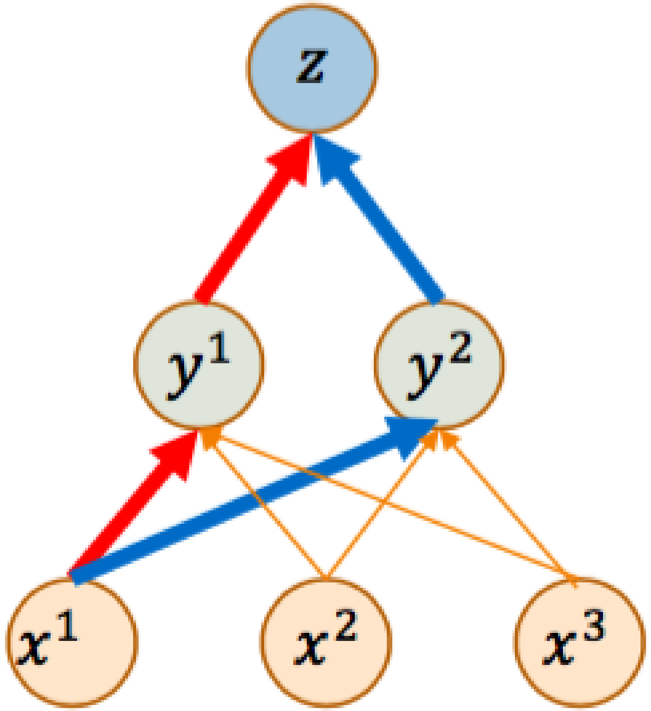
- Assume a nested function, $z = f(y)$ and y= g(x)
- Chain Rule for scalars x, y, z

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

- When $x \in$ R^m,$y \in$ R^n. $z \in$R

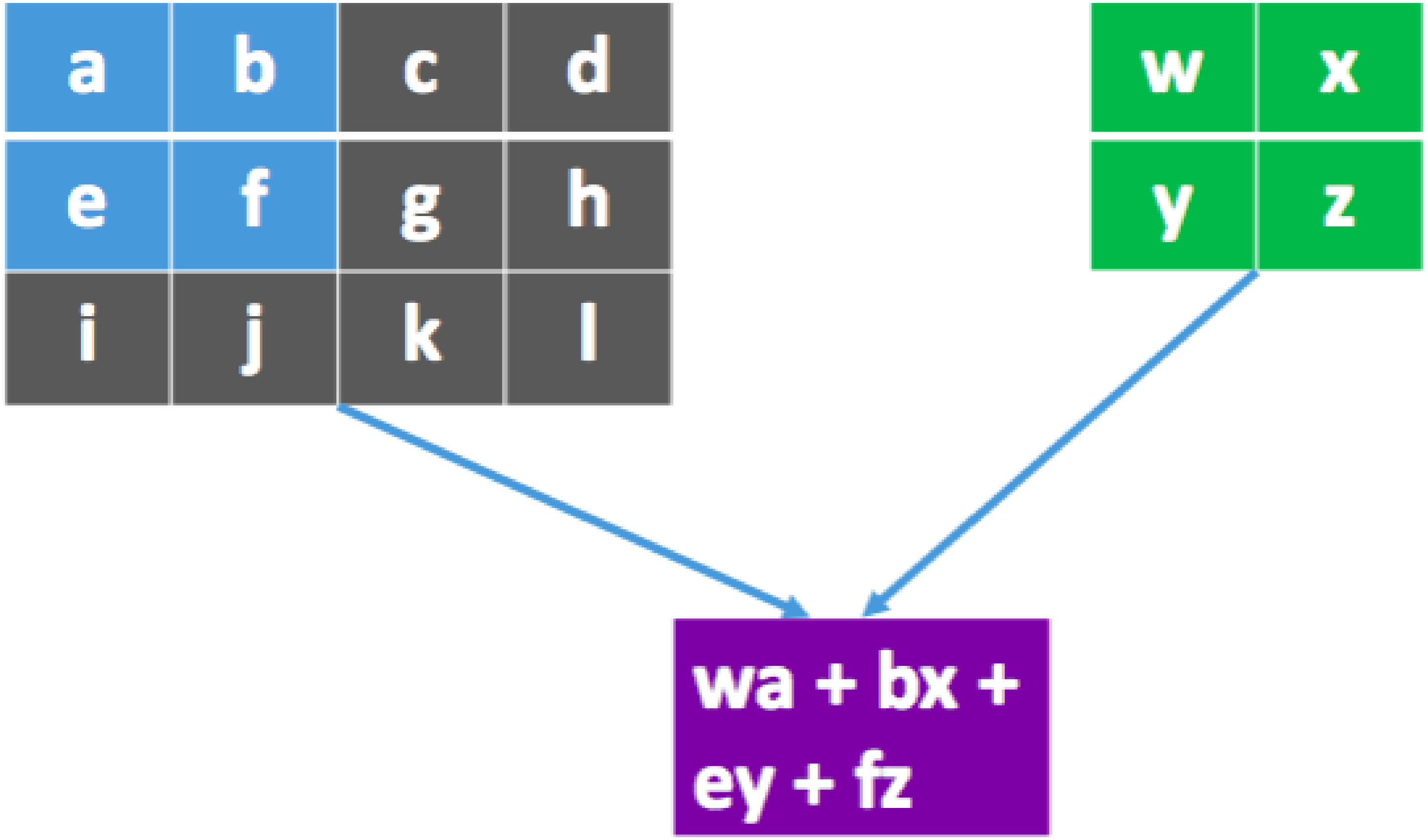
$$\frac{dz}{dx_i} = \sum_j \frac{dz}{dy_j}\frac{dy_j}{dx_i}$$

  - i.e., gradients from all possible paths



$$\frac{dz}{dx^1} = \frac{dz}{dy^1}\frac{dy^1}{dx^1} + \frac{dz}{dy^2}\frac{dy^2}{dx^1}$$

Illustration 2: two dimensional case

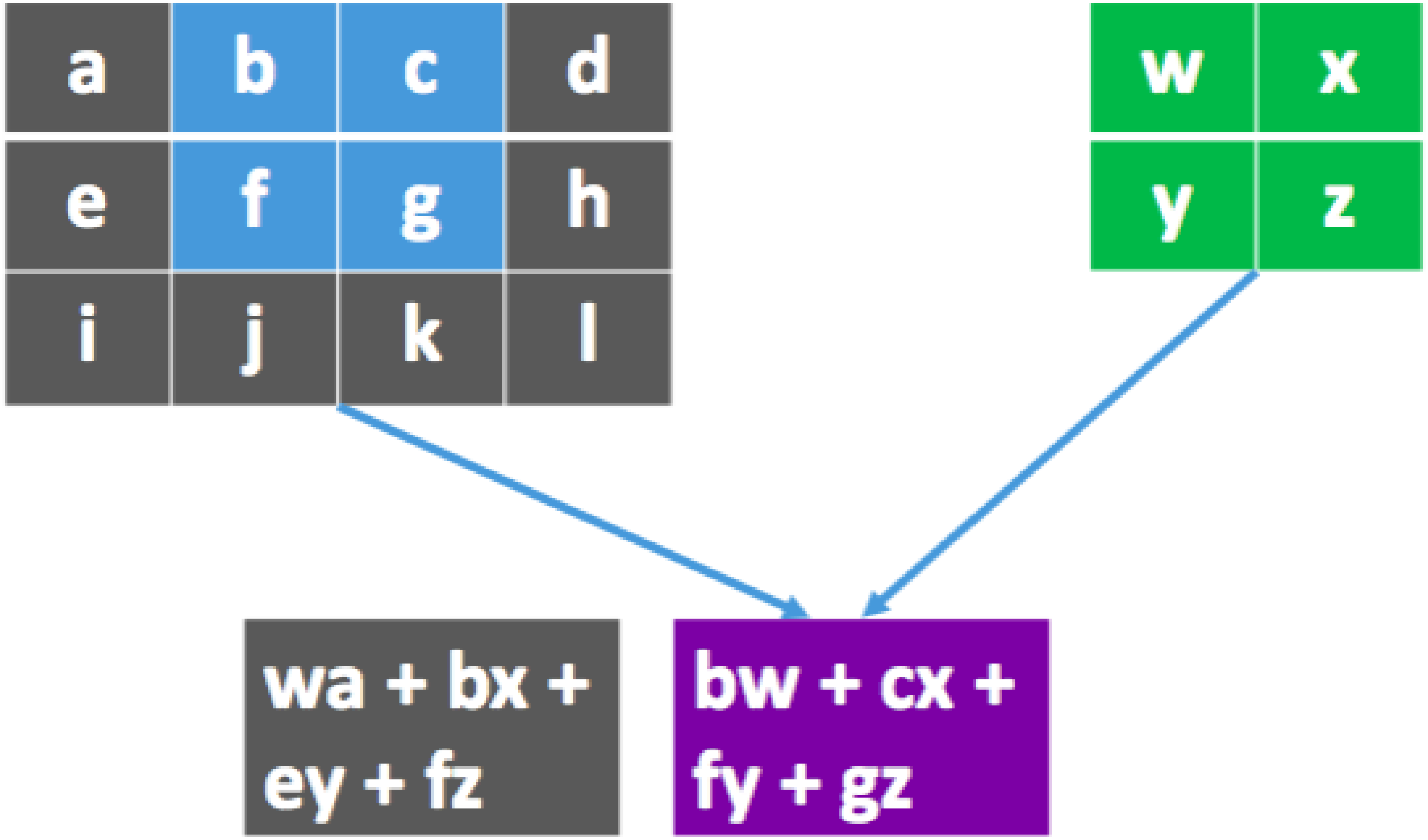


a	b	c	d
e	f	g	h
i	j	k	l

w	x
y	z

wa + bx + ey + fz

Illustration 2



a	b	c	d
e	f	g	h
i	j	k	l

w	x
y	z

wa + bx + ey + fz

bw + cx + fy + gz

Zero-Padding

filter

w	**x**
y	z

Input

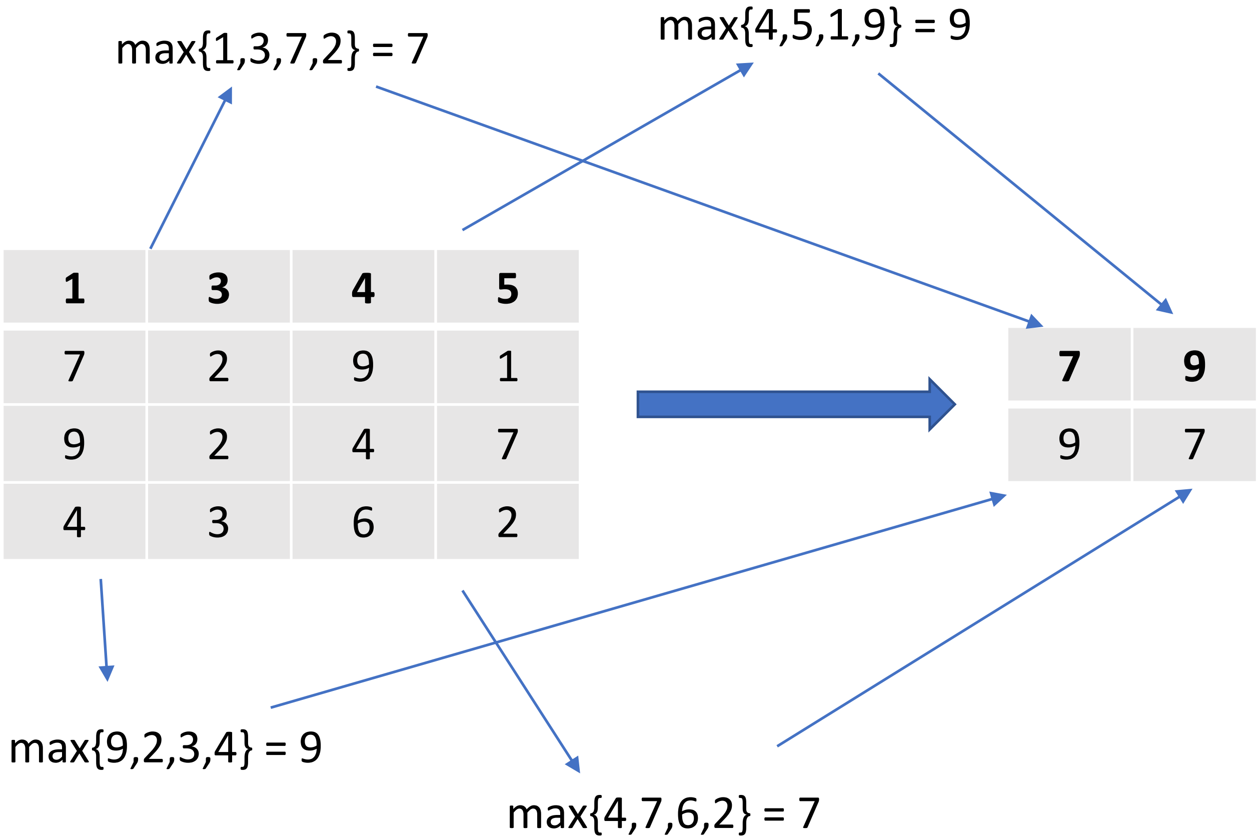
a	b	c	d
e	f	g	h
i	j	k	l

→

0	0	0	0	0	0
0	a	b	c	d	0
0	e	f	g	h	0
0	i	j	k	l	0
0	0	0	0	0	0

What’s the shape of the resulting matrix?

Max-pooling



1	**3**	**4**	**5**
7	2	9	1
9	2	4	7
4	3	6	2

7	**9**
9	7

max{1,3,7,2} = 7

max{4,5,1,9} = 9

max{9,2,3,4} = 9

max{4,7,6,2} = 7

Part 4: Probabilistic Graphical Models

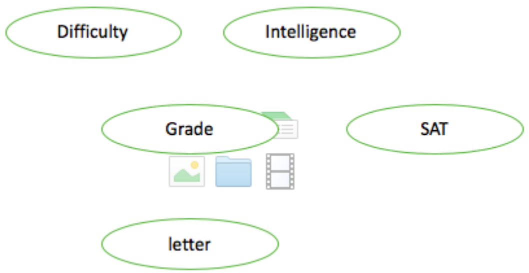
So What is a Graphical Model?

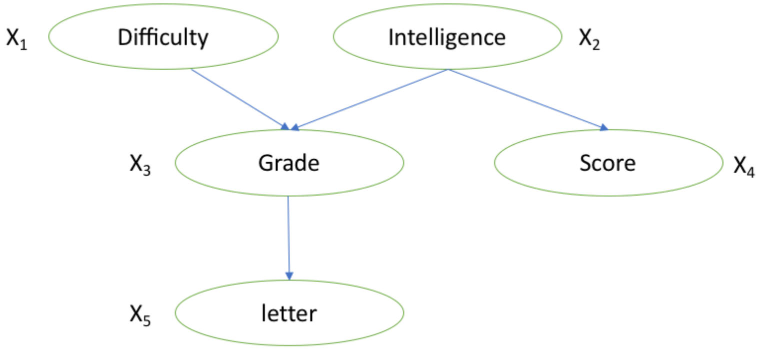
- In a nutshell,

GM = **Multivariate Statistics + Structure**

What is a Graphical Model?

- The informal blurb:
  - It is a smart way to write/specify/compose/design exponentially-large probability distributions without paying an exponential cost, and at the same time endow the distributions with ***structured semantics***





- A more formal description:
  - It refers to a family of distributions on a set of random variables that are compatible with all the probabilistic independence propositions encoded by a graph that connects these variables

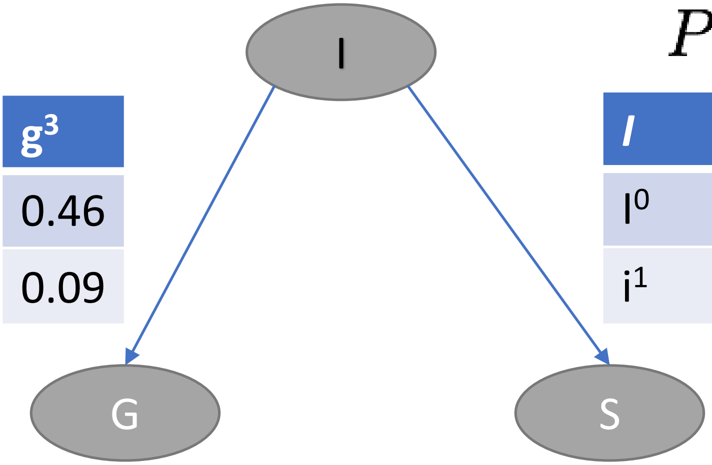
Naïve Bayes Model

- $Val(G) = \{g^1, g^2, g^3\}$ represents grades *A, B, C*

i^0	i^1
0.7	0.3

$P(G|I)$

I	g^1	g^2	g^3
i^0	0.2	0.34	0.46
i^1	0.74	0.17	0.09



$P(S|I)$

I	s^0	s^1
I^0	0.95	0.05
i^1	0.2	0.8

Recall: Naïve Bayes Model

- Score and Grade are independent given Intelligence (assumption)
  - Knowing Intelligence, Score gives no information about class grade
- Assertions
  - From probabilistic reasoning $P(I, S, G) = P(I)P(S, G \mid I)$
  - From assumption $P \models (S \perp G \mid I)$
- Combining, we have

$$P(S, G \mid I) = P(S \mid I)P(G \mid I)$$

$$P(I, S, G) = P(I)P(S \mid I)P(G \mid I)$$

Therefore, $P(i^1, s^1, g^2) = P(i^1)P(s^1 \mid i^1)P(g^2 \mid i^1)$
$= 0.3 * 0.8 * 0.17 = 0.0408$

Three binomials,
two 3-value multinomials:
7 params
More compact than joint distribution

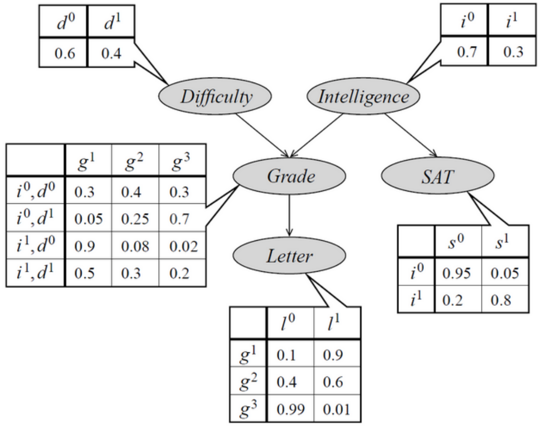
Local Independencies in a BN

- A BN G is a directed acyclic graph whose nodes represent random variables $X_i,..,X_n$.
- Let $Pa(X_i)$ denote parents of X_i in G
- Let $\mathit{Non\text{-}Desc}(X_i)$ denote variables in G that are not descendants of X_i
- Then G encodes the following set of *conditional independence* assumptions denoted *I*(G)
  - For each X_i: $(X_i \perp \mathit{Non\text{-}Desc}(X_i) \mid Pa(X_i))$
- Also known as *Local Markov Independencies*

Local Independencies

- Graph G with CPDs is equivalent to a set of independence assertions

$$P(D,I,G,S,L) = P(D)P(I)P(G \mid D,I)P(S \mid I)P(L \mid G)$$



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

- Local Conditional Independence Assertions (starting from leaf nodes):

$I(G) = \{(L \perp I,D,S \mid G),$ *L* is conditionally independent of all other nodes given parent G

$(S \perp D,G,L \mid I),$ *S* is conditionally independent of all other nodes given parent *I*

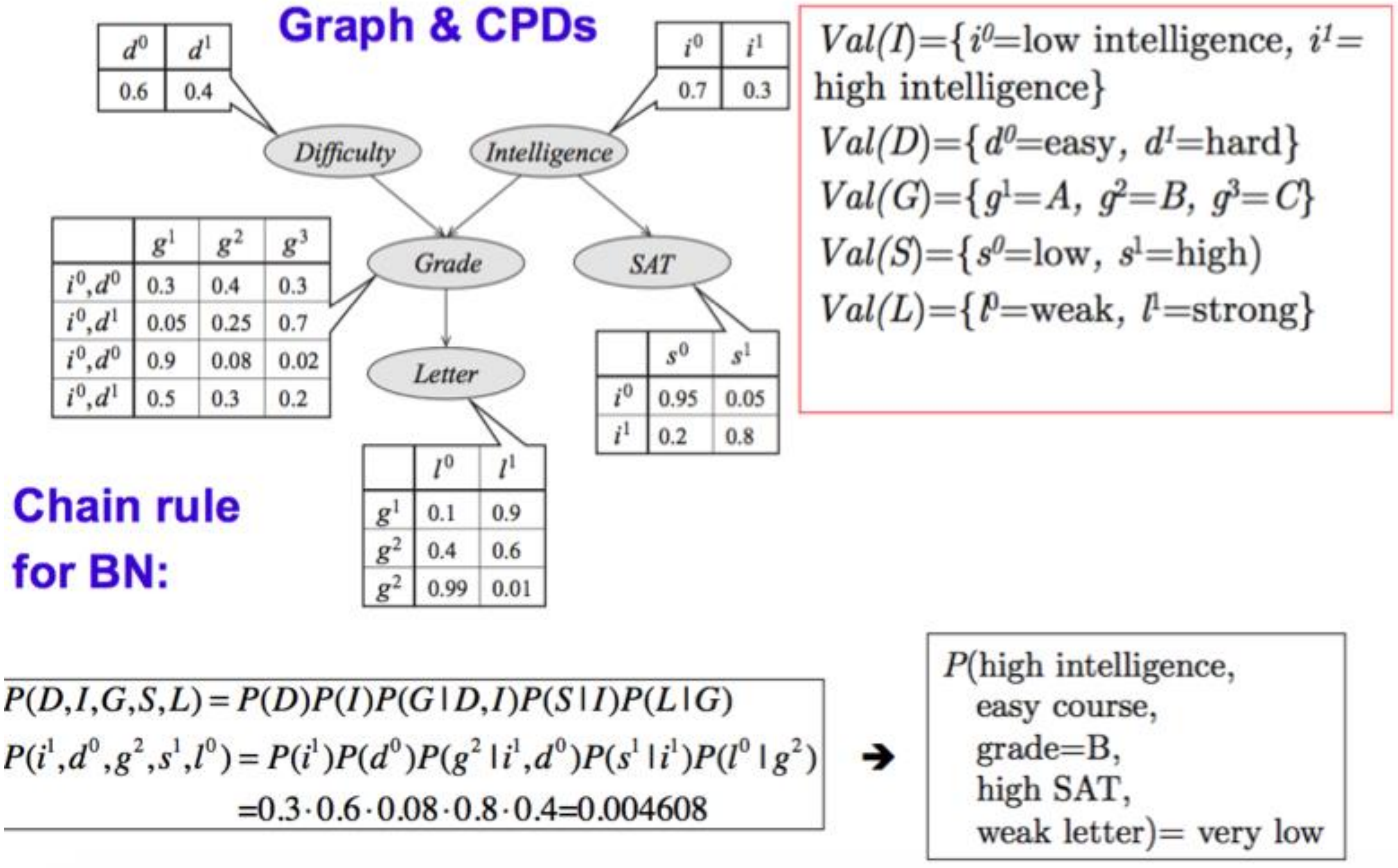
$(G \perp S \mid D,I),$ Even given parents, *G* is NOT independent of descendant *L*

$(\mathrm{I} \perp D \mid \phi),$ Nodes with no parents are marginally independent

$(D \perp I,S \mid \phi)\}$ *D* is independent of non-descendants *I* and *S*

- Parents of a variable shield it from probabilistic influence
  - Once value of parents known, no influence of ancestors
- Information about descendants can change beliefs about a node

Evaluating a Joint Probability



Graph & CPDs

d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^0,d^0	0.9	0.08	0.02
i^0,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^2	0.99	0.01

$Val(I)$={i^0=low intelligence, i^1= high intelligence}

$Val(D)$={d^0=easy, d^1=hard}

$Val(G)$={$g^1=A$, $g^2=B$, $g^3=C$}

$Val(S)$={s^0=low, s^1=high)

$Val(L)$={l^0=weak, l^1=strong}

Chain rule for BN:

$$P(D,I,G,S,L)=P(D)P(I)P(G\mid D,I)P(S\mid I)P(L\mid G)$$

$$P(i^1,d^0,g^2,s^1,l^0)=P(i^1)P(d^0)P(g^2\mid i^1,d^0)P(s^1\mid i^1)P(l^0\mid g^2)$$

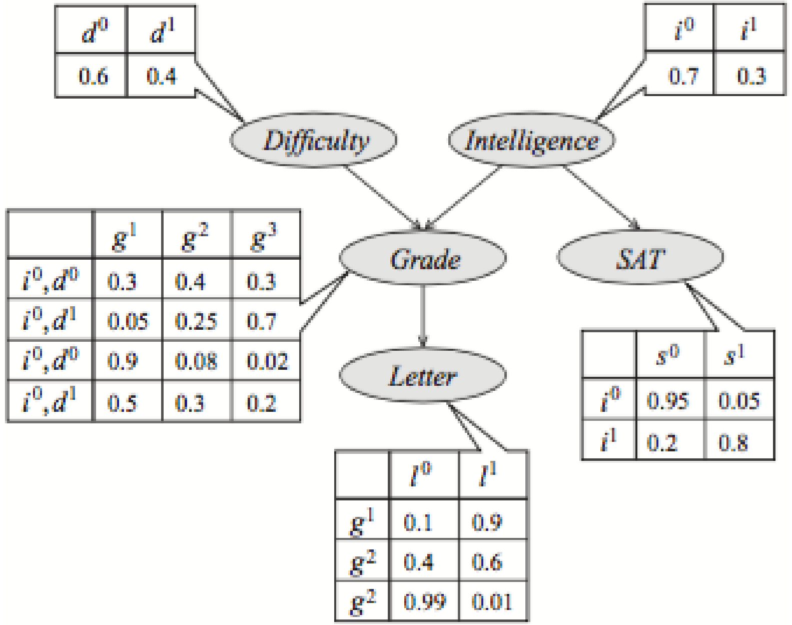
$$=0.3\cdot 0.6\cdot 0.08\cdot 0.8\cdot 0.4=0.004608$$

➔

P(high intelligence,
easy course,
grade=B,
high SAT,
weak letter)= very low

Reasoning Patterns

- **Reasoning about a student *George* using the model**



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^0, d^0	0.9	0.08	0.02
i^0, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

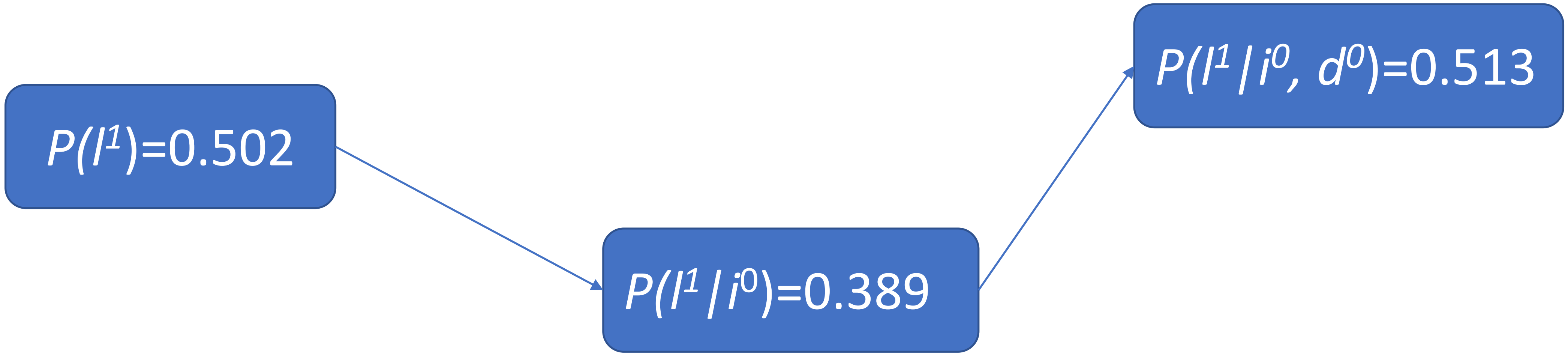
	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^2	0.99	0.01

George

- Causal Reasoning
  - *George* is interested in knowing as to how likely he is to get a strong *Letter* (based on *Intelligence*, *Difficulty*)?

- Evidential Reasoning
  - *Recruiter* is interested in knowing whether *George* is *Intelligent* (based on *Letter*, *SAT*)

Recruiter



$P(l^1)=0.502$

$P(l^1 \,|\, i^0)=0.389$

$P(l^1 \,|\, i^0, d^0)=0.513$

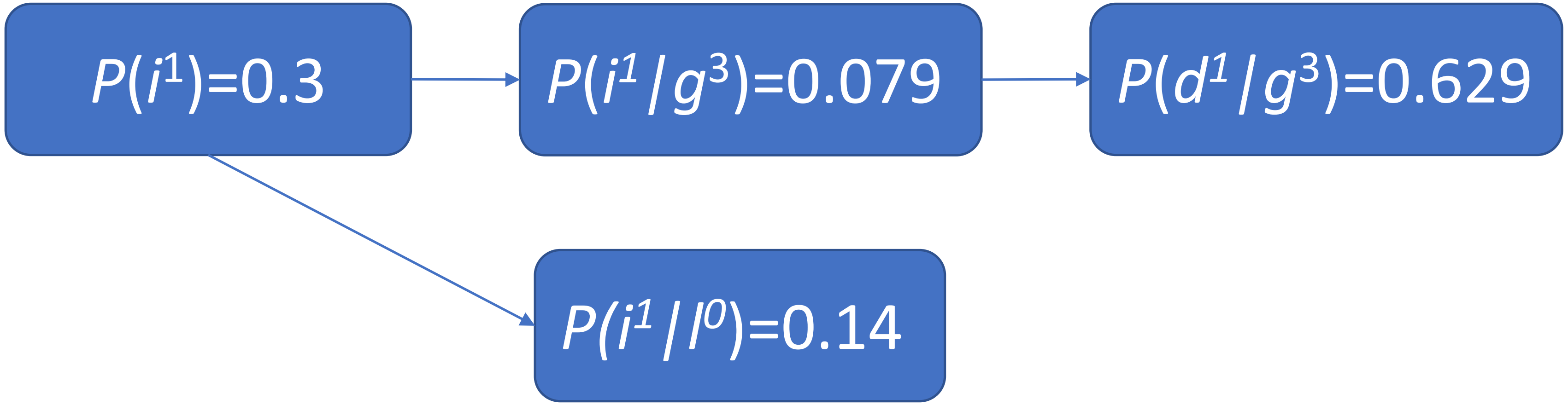
After knowing that the student is not as intelligent, we understand that the probability of getting a strong recommendation letter is lower.

→

After further knowing that the difficulty is low, the probability of getting a strong letter is higher.

low grade drastically decreases the probability of high intelligence

low grade justifies the difficulty



$P(i^1)=0.3$

$P(i^1 | g^3)=0.079$

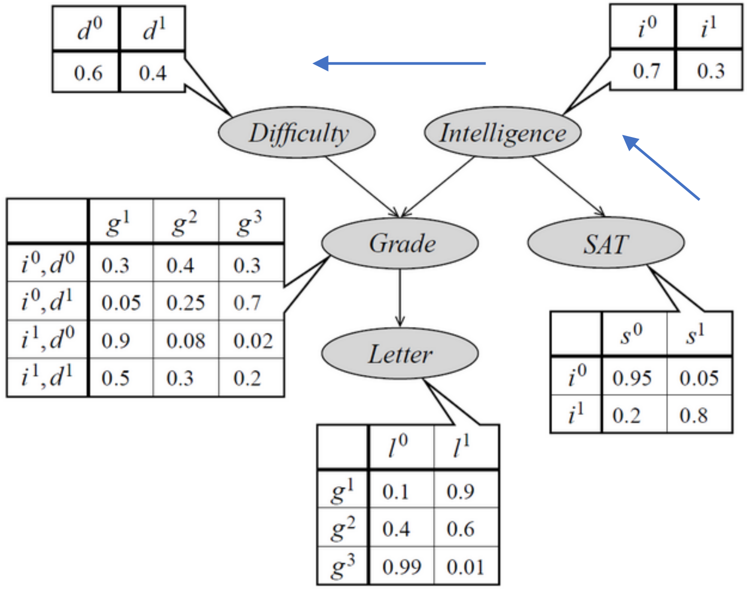
$P(d^1 | g^3)=0.629$

$P(i^1 | l^0)=0.14$

A weak letter drastically decreases the probability of high intelligence

Intercausal reasoning

- The previous example:
  - Information about Score gave us information about Intelligence which with Grade told us about difficulty of course
  - One causal factor for Grade, i.e., Intelligence, give us information about another (Difficulty)



d^0	d^1
0.6	0.4

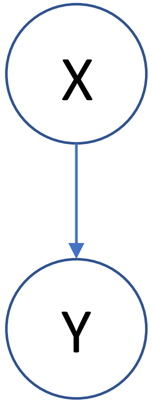
i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

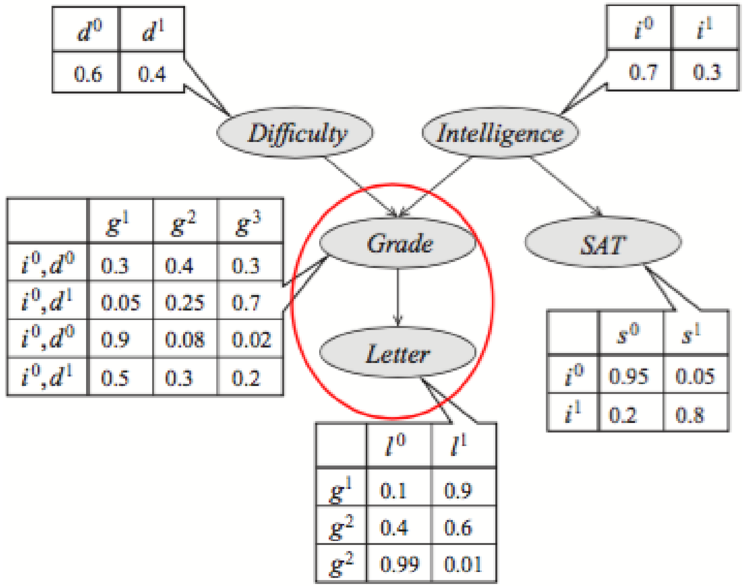
	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Direct Connection between *X* and *Y*



- *X* and *Y* are correlated regardless of any evidence about any other variables
  - E.g., Feature *Y* and character *X* are correlated
  - Grade *G* and Letter *L* are correlated
- If *X* and *Y* are directly connected we can get examples where they influence each other regardless of *Z*



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

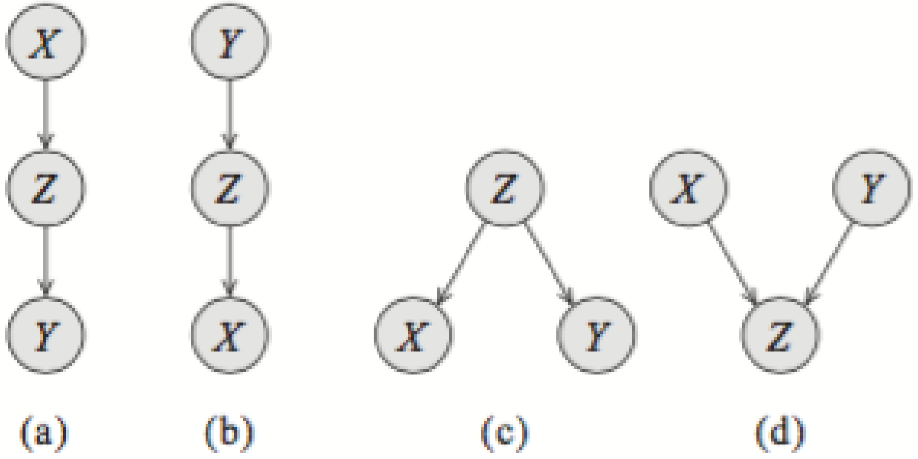
	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^0,d^0	0.9	0.08	0.02
i^0,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^2	0.99	0.01

Indirect Connection between *X* and *Y*

- Four cases where *X* and *Y* are connected via *Z*
- (a). Indirect causal effect
- (b). Indirect evidential effect
- (c). Common cause
- (d). Common effect



- We will see that first three cases are similar while fourth case (*V*-structure) is different

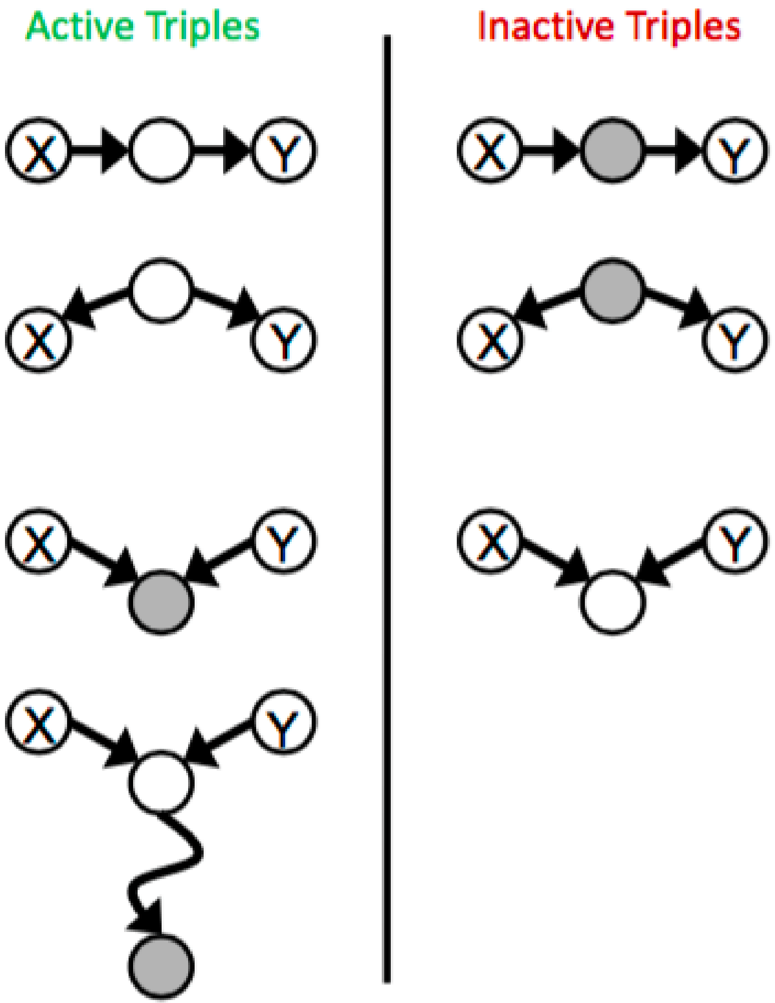
Summary of Indirect Connection

- Causal trail: *X->Z->Y*: active iff *Z* not observed
- Evidential Trail: *X<-Z<-Y*: active iff *Z* is not observed
- Common Cause: *X<-Z->Y*: active iff *Z* is not observed
- Common Effect: *X->Z<-Y*: active iff either *Z* or one of its descendants is observed

What is the general case?

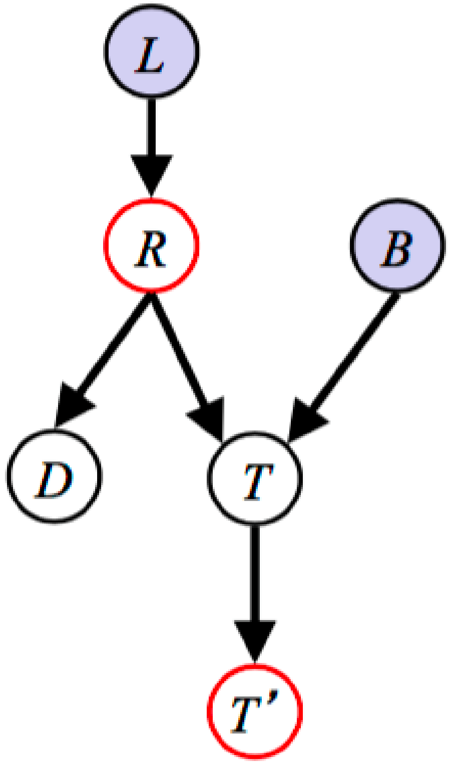
Active / Inactive Paths

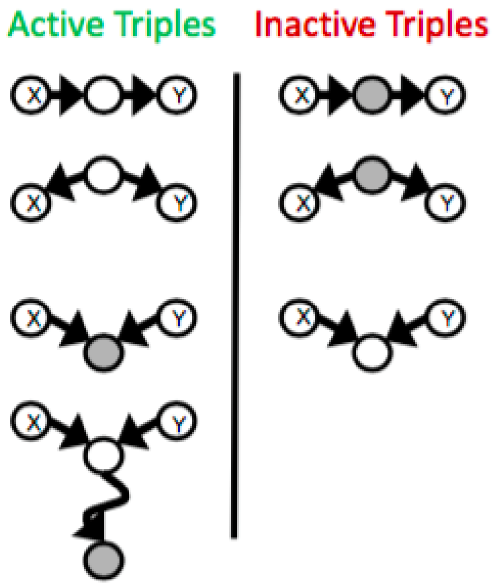
- Question: Are X and Y conditionally independent given evidence variables {Z}?
  - Yes,if X and Y “d-separated” by Z
  - Consider all (undirected) paths from X to Y
  - If no path is active -> independence!
- A ***path*** is active if every triple in path is active:
  - Causal chain A -> B -> C where B is unobserved (either direction)
  - Commoncause A <- B -> C where B is unobserved
  - Common effect (aka v-structure)
    A -> B <- C where B *or one of its descendants* is observed
- All it takes to block a path is a ***single*** inactive segment
  - (But ***all*** paths must be inactive)



Example

$L \perp\!\!\!\perp T' \mid T$	*Yes, Independent*
$L \perp\!\!\!\perp B$	*Yes, Independent*
$L \perp\!\!\!\perp B \mid T$	*No*
$L \perp\!\!\!\perp B \mid T'$	*No*
$L \perp\!\!\!\perp B \mid T, R$	*Yes, Independent*
$R \perp\!\!\!\perp T' \mid L, B$	*No*




Active Triples
Inactive Triples
X
Y