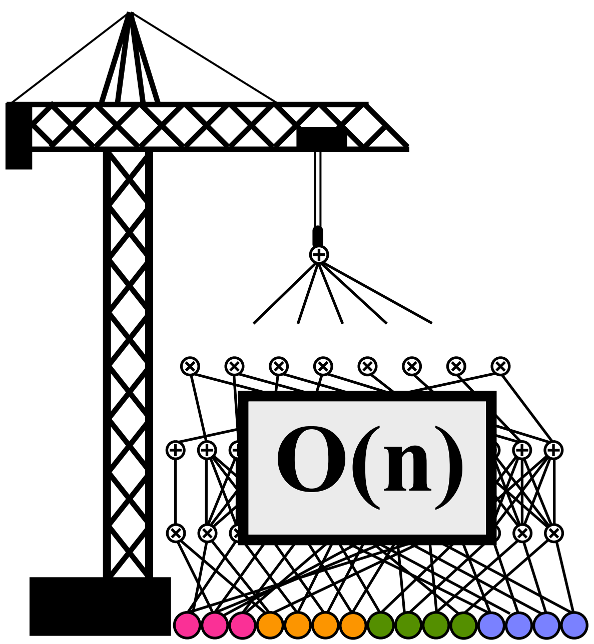
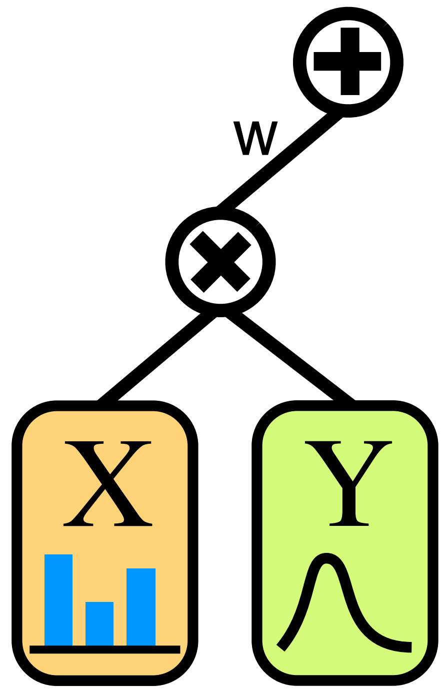
# Learning the Structure of Sum-Product Networks

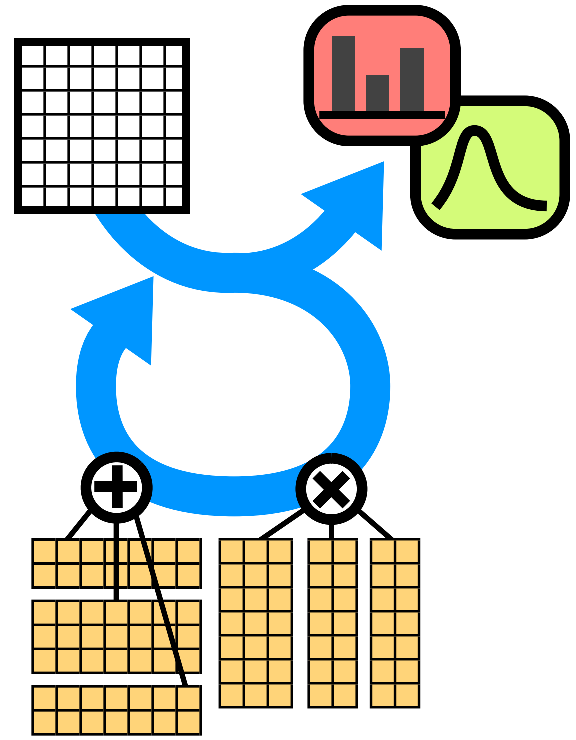
Robert Gens
Pedro Domingos



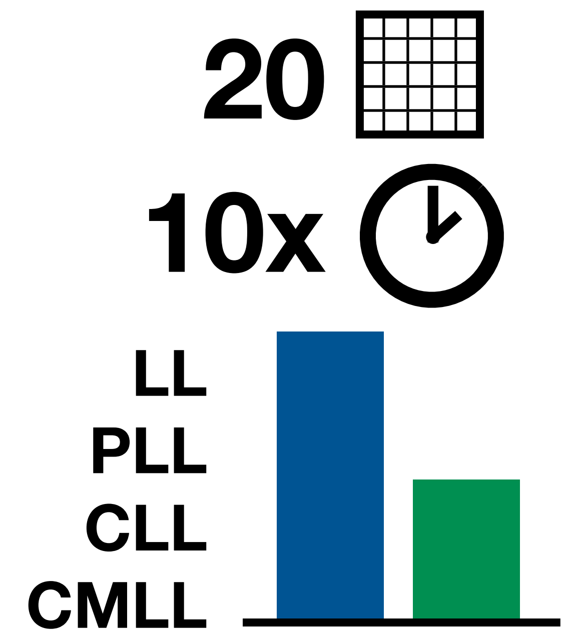

**Motivation**




**SPN Review**

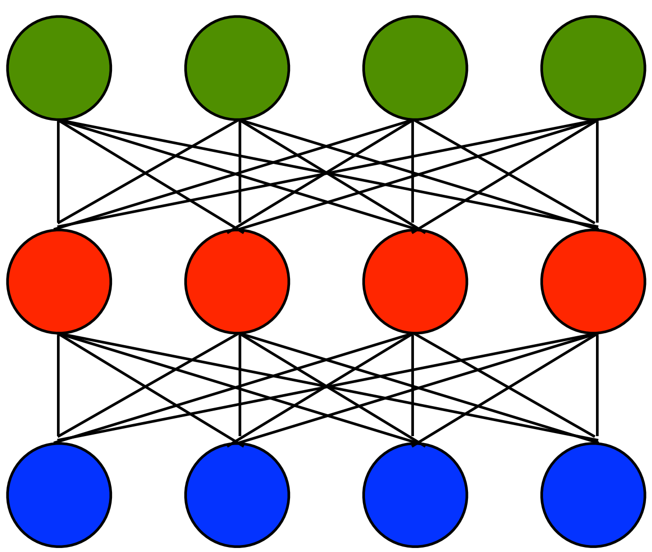


**Structure Learning**




**Experiments**

# Graphical Models

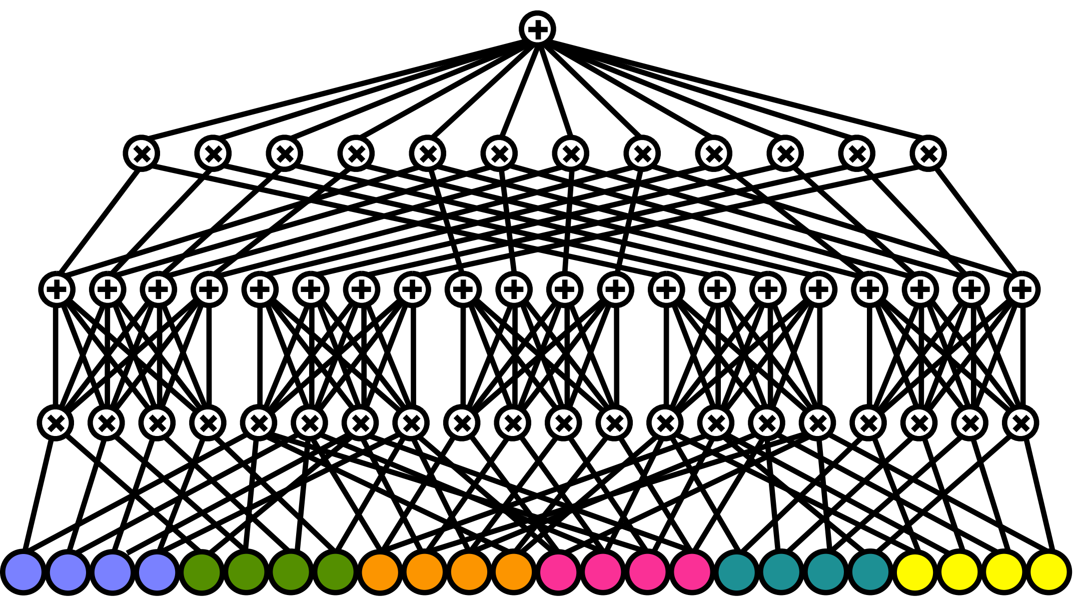


**Representation** Compact and expressive
Global independence

**Inference** <u>Exponential</u> in treewidth of graph

**Learning** Extremely difficult because:
- Learning requires inference
- Approximate inference is unreliable
- Hidden variables ➔ no global optimum

# Sum-Product Networks

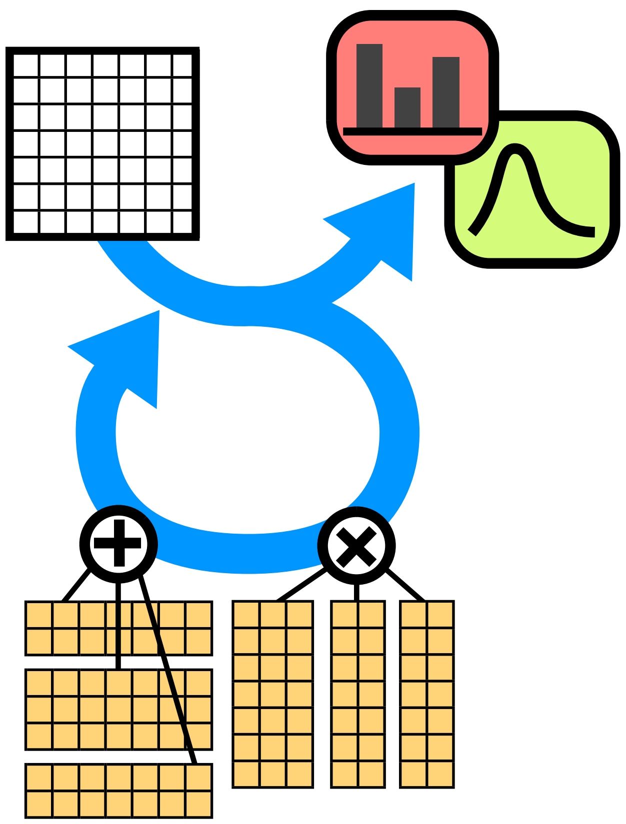


**Representation** Compact and expressive
Local independence

**Inference** Linear in number of edges

**Learning** Much easier because exact inference
Only weight learning to date

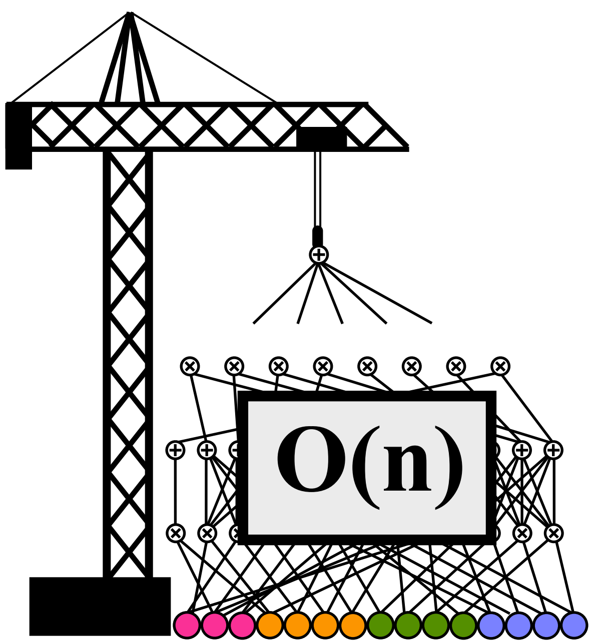
# This Paper:
# SPN Structure Learning



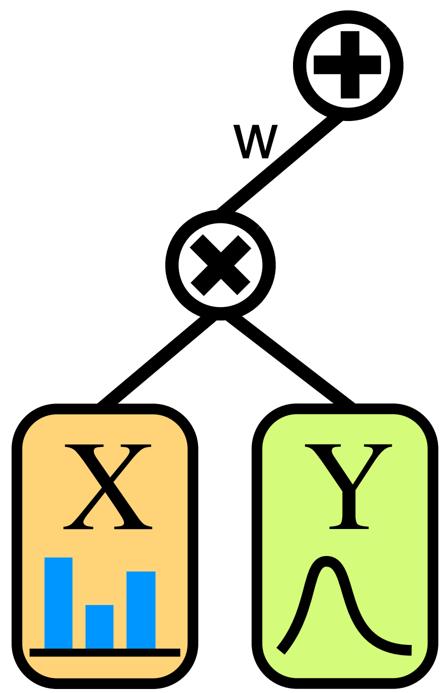
- **General-purpose SPN structure learning**
  - Discrete or continuous
  - Learns layers of hidden variables
  - Fully leverages context-specific independence

- **Simple and intuitive**

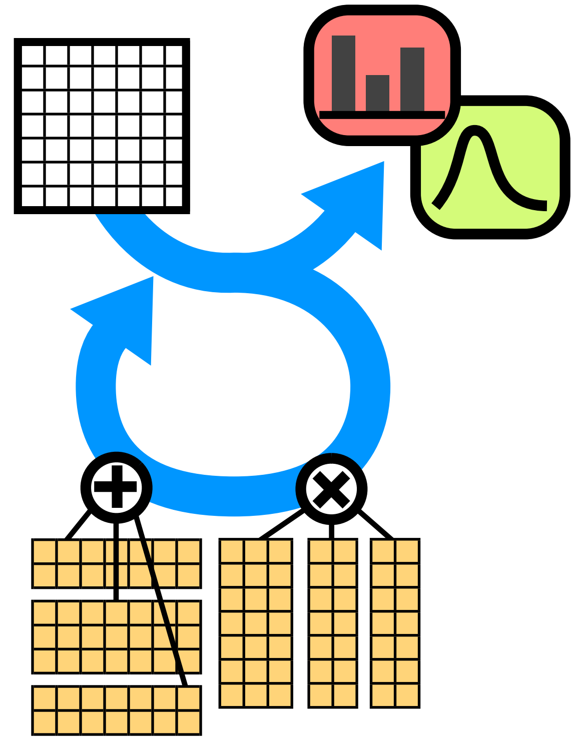
- **1-3 orders of magnitude faster, and more accurate at query time**



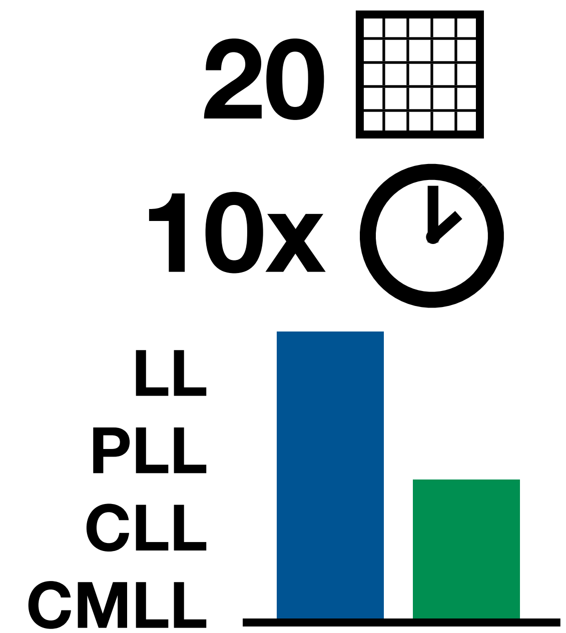

Motivation




SPN Review

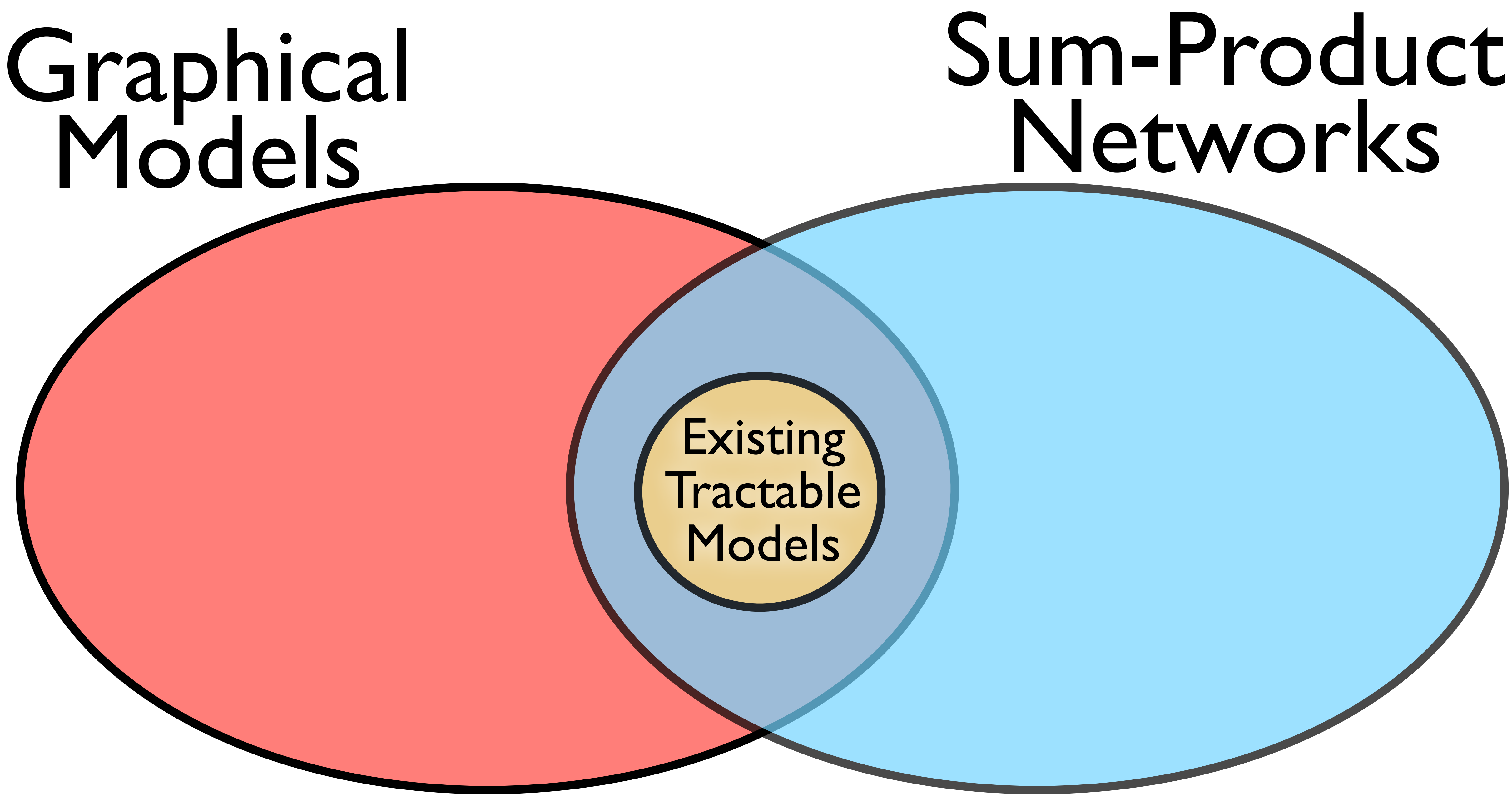


Structure Learning

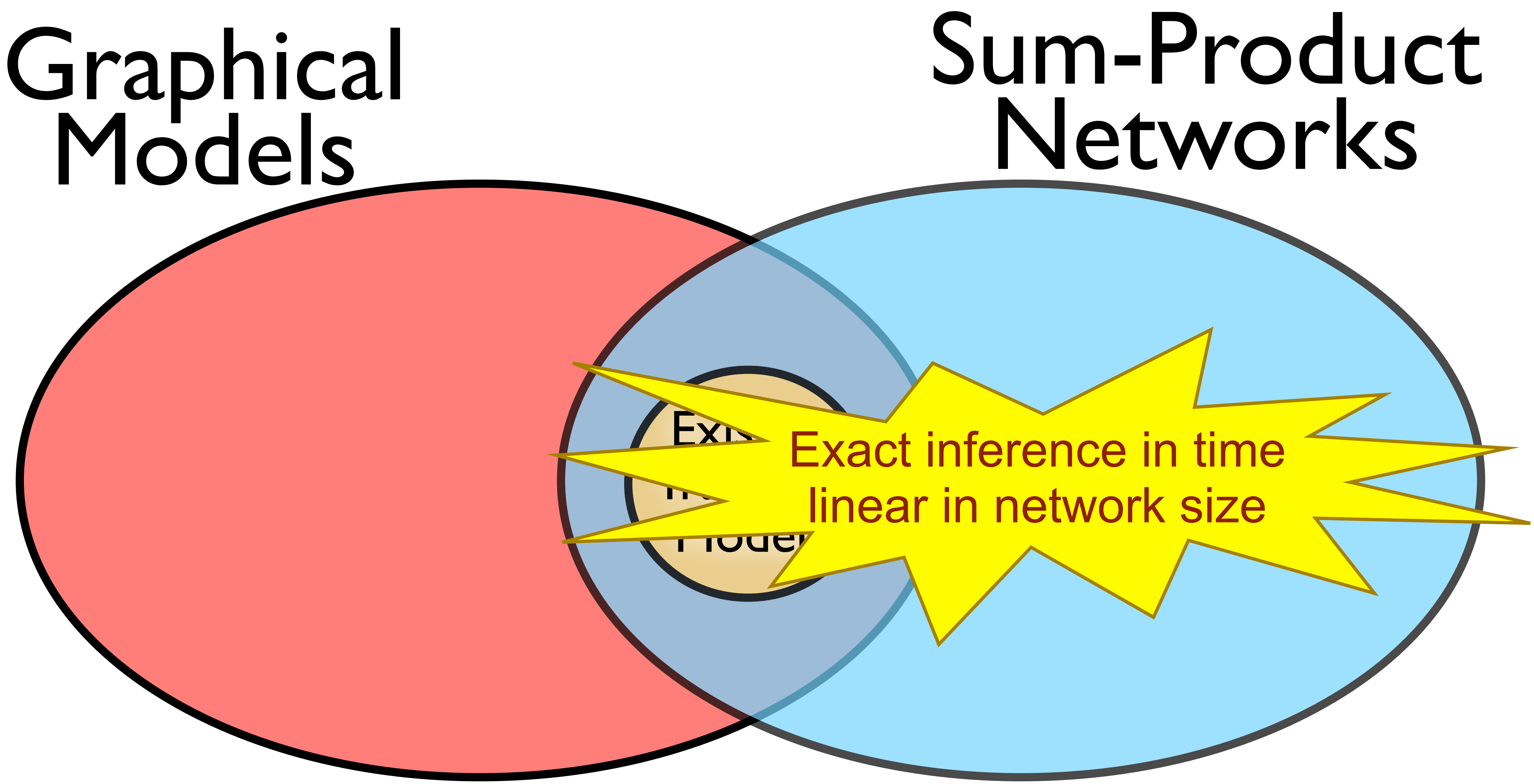



Experiments

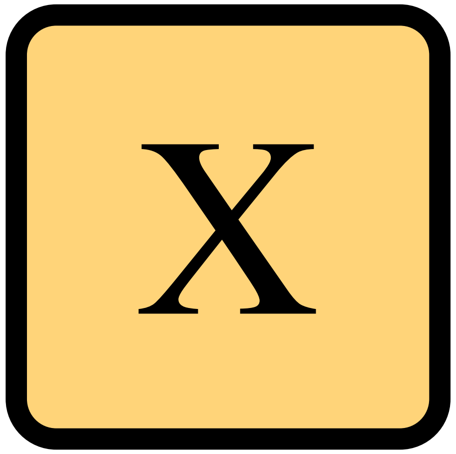
# Compactly Representable Probability Distributions



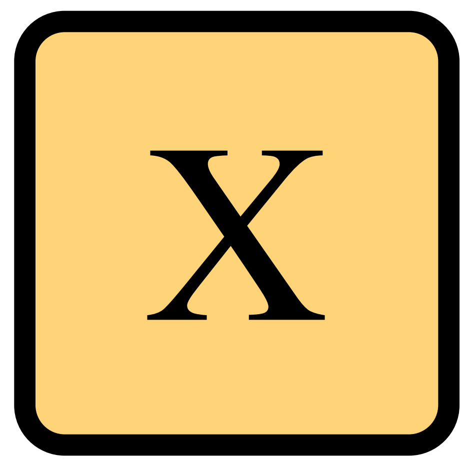
# Compactly Representable Probability Distributions

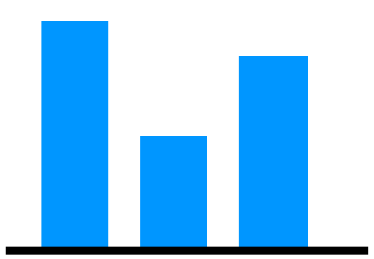


# A Univariate Distribution Is an SPN.

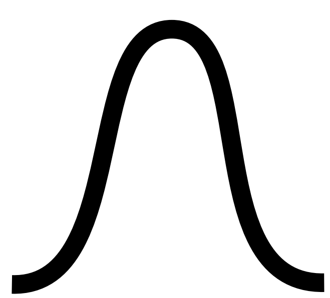


# A Univariate Distribution Is an SPN.

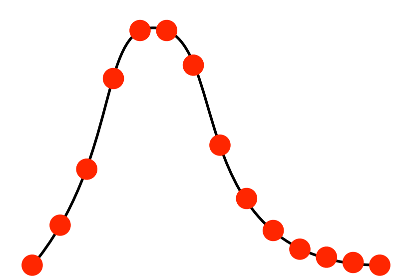


Multinomial



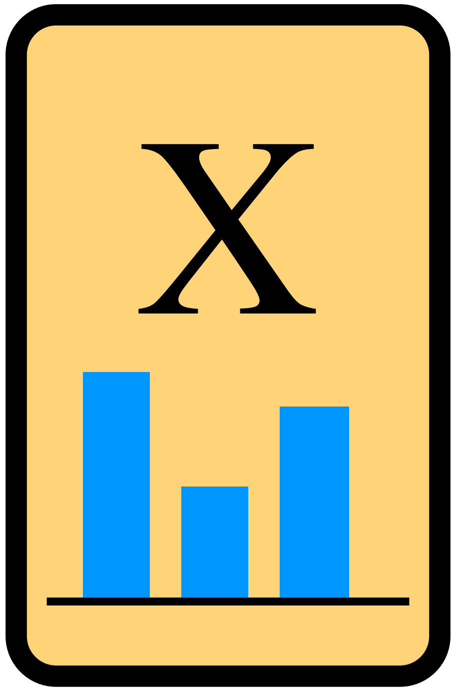
Gaussian

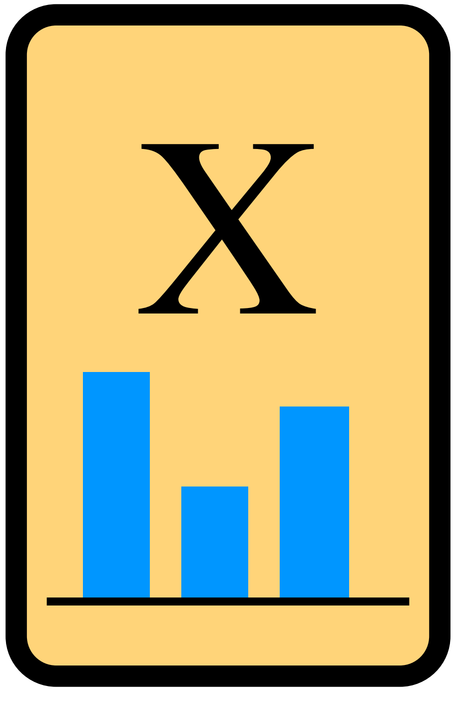


Poisson

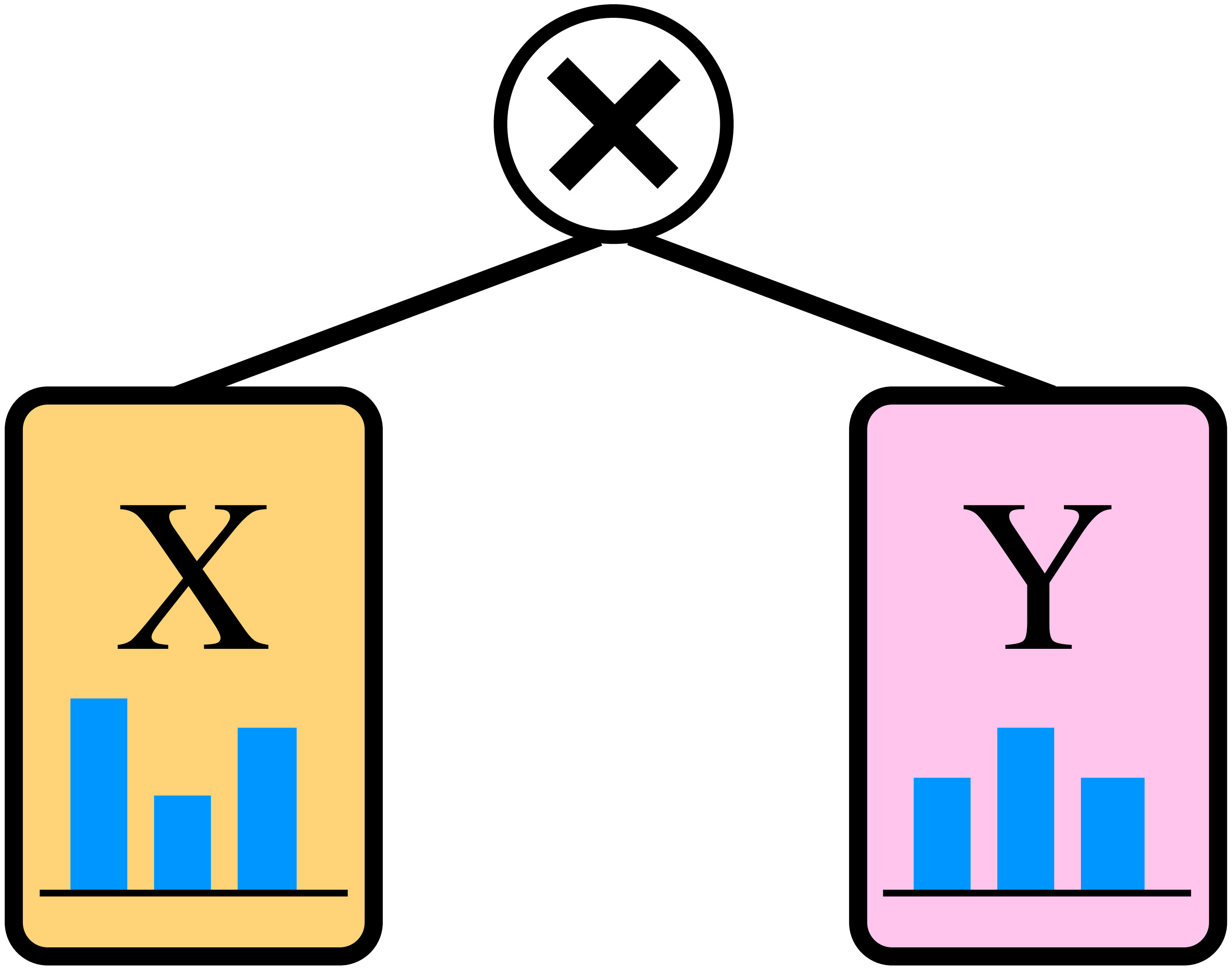
...

# A Univariate Distribution Is an SPN.

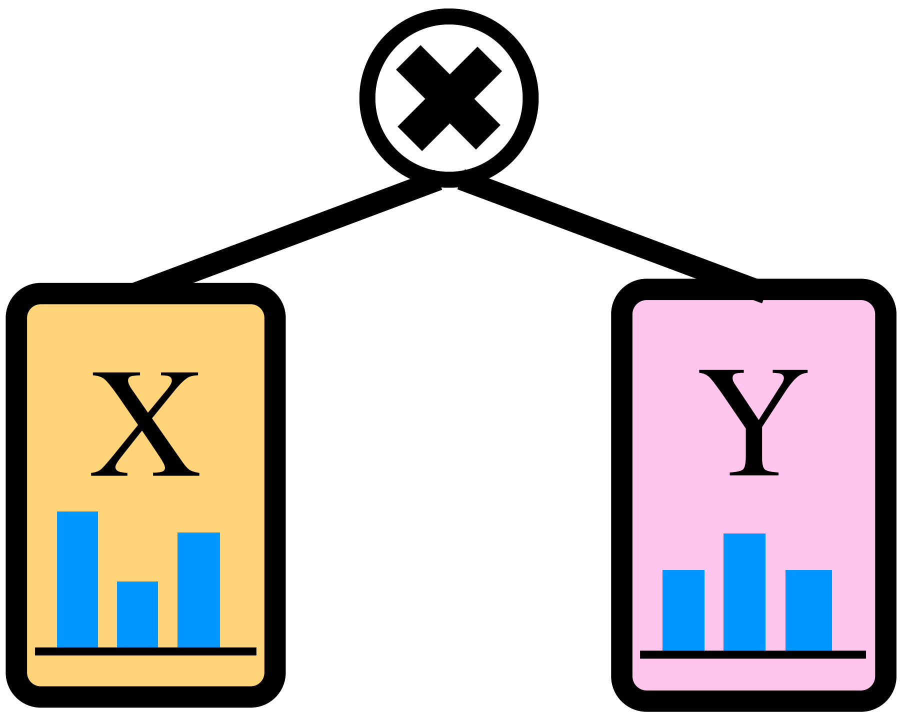


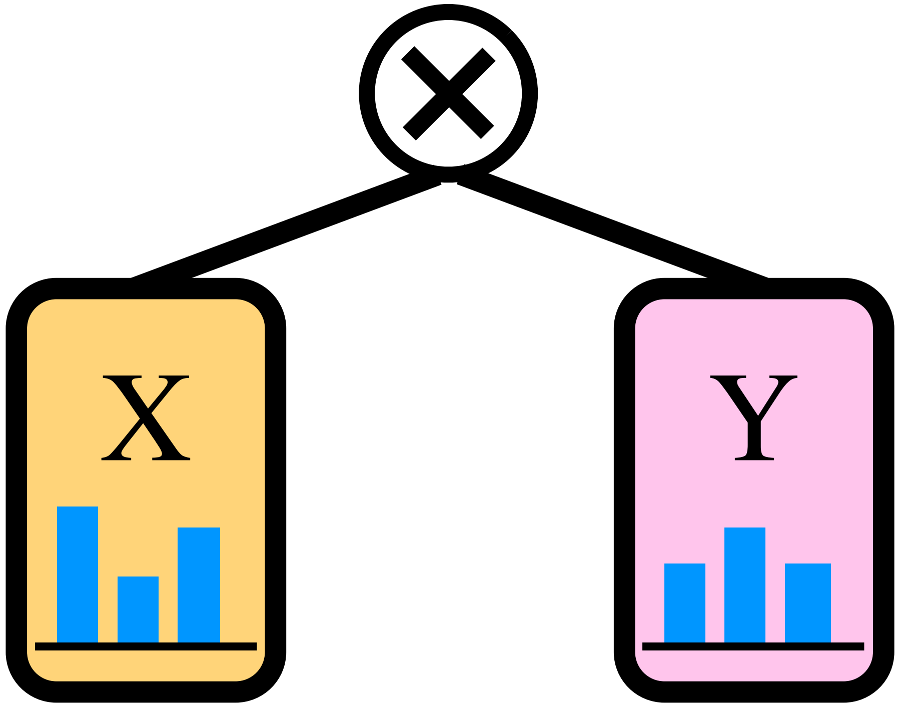

X

# A Product of SPNs over Disjoint Variables Is an SPN.

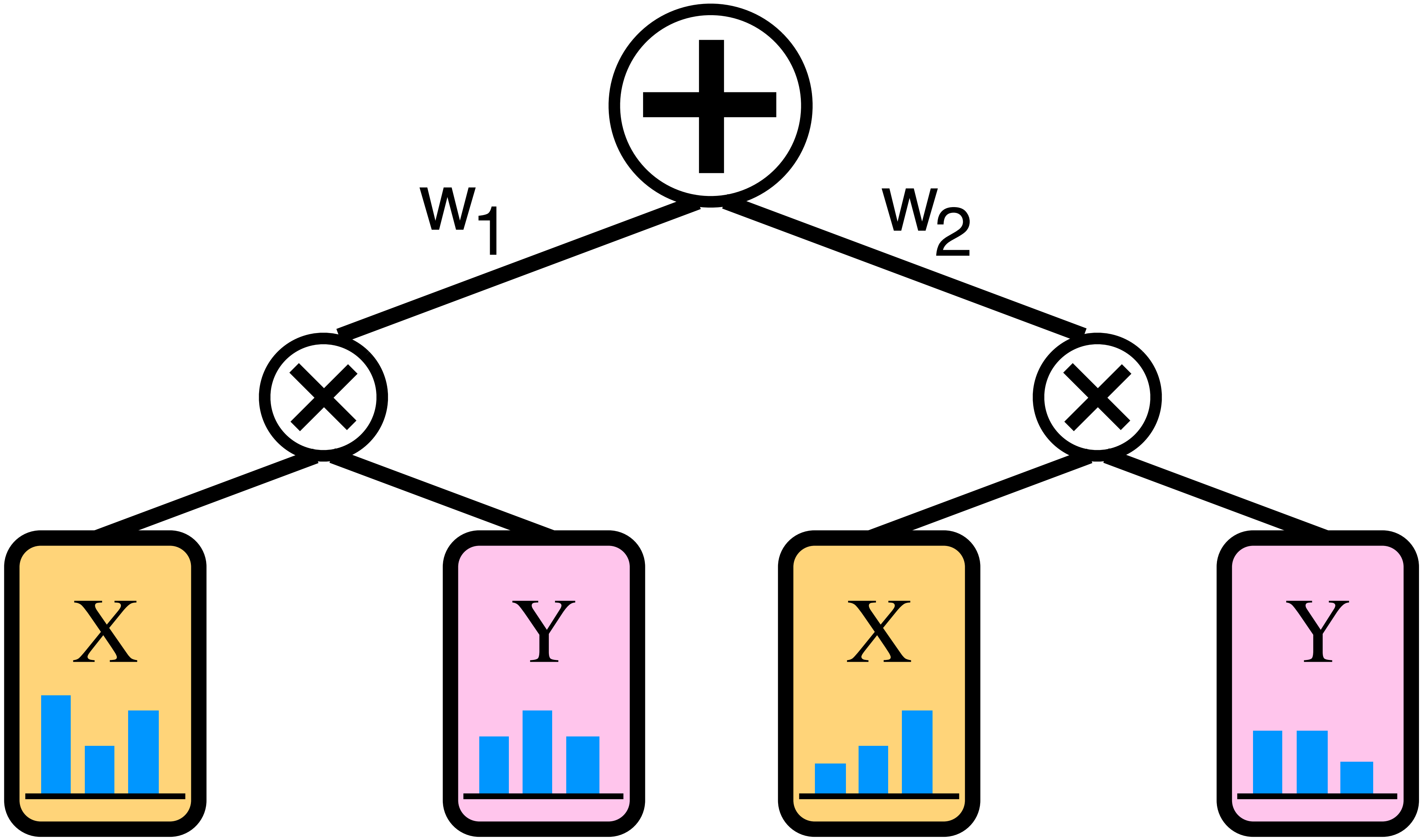


# A Product of SPNs over Disjoint Variables Is an SPN.

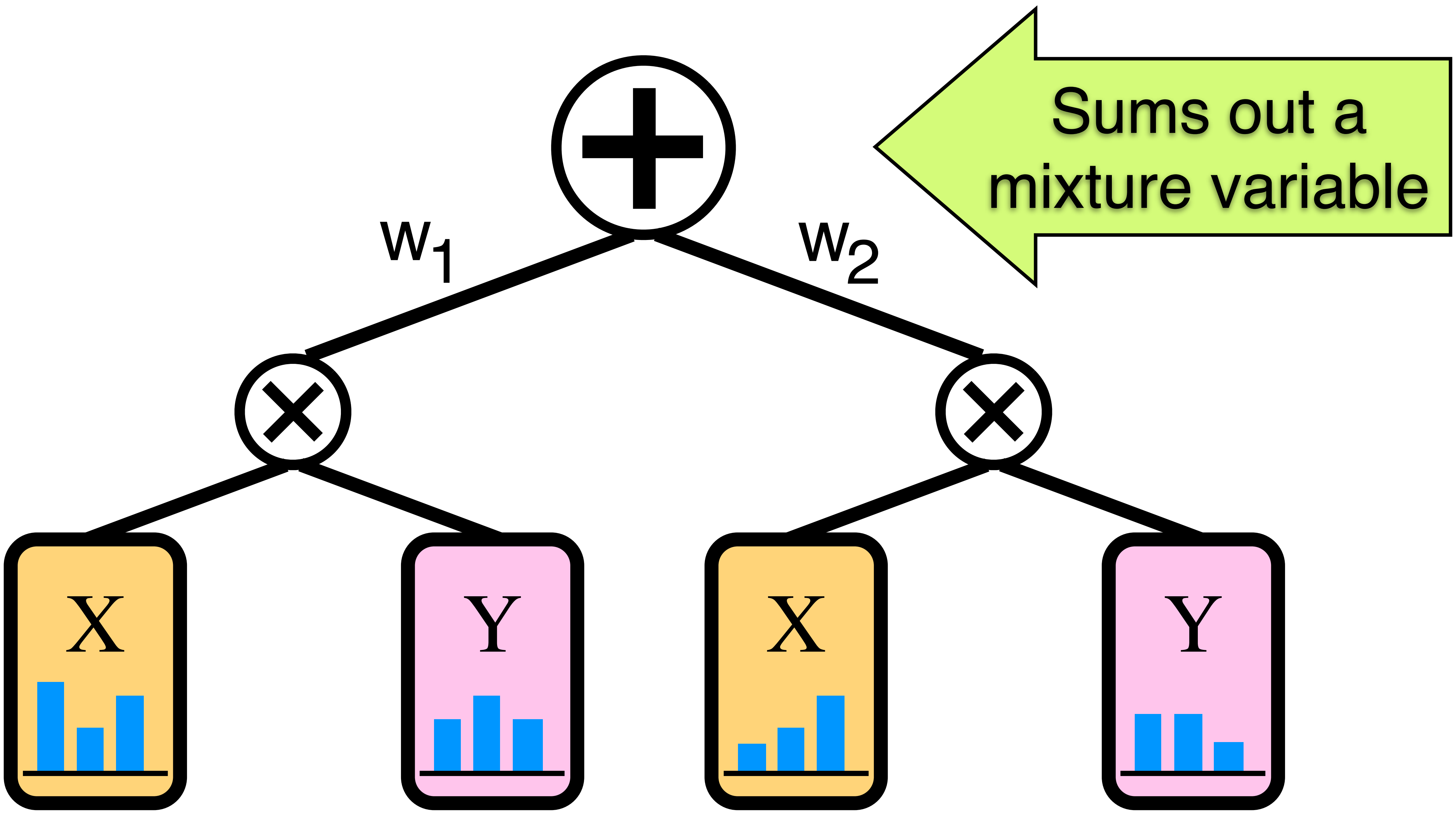


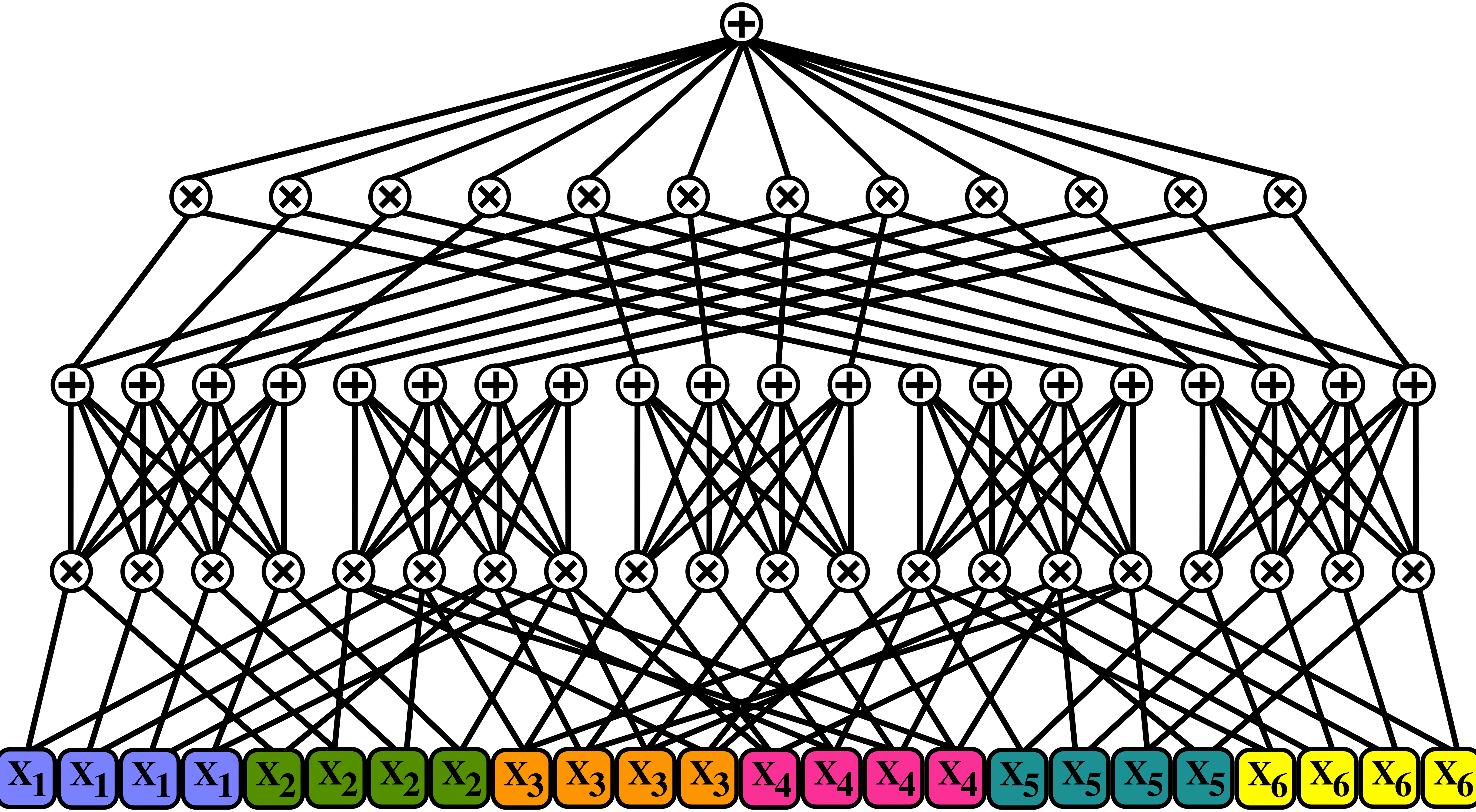

X
Y

# A Weighted Sum of SPNs over the Same Variables Is an SPN.



# A Weighted Sum of SPNs over the Same Variables Is an SPN.

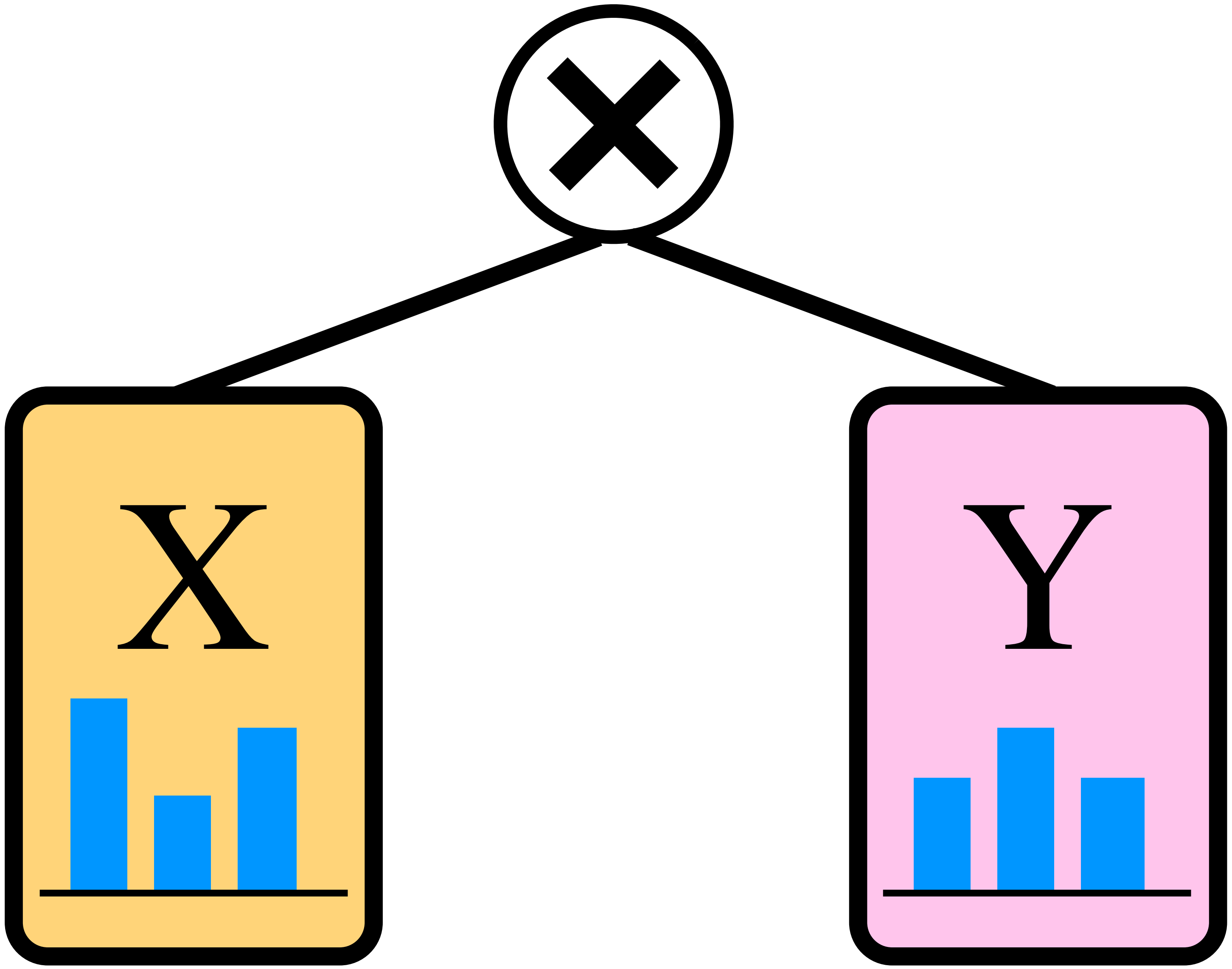




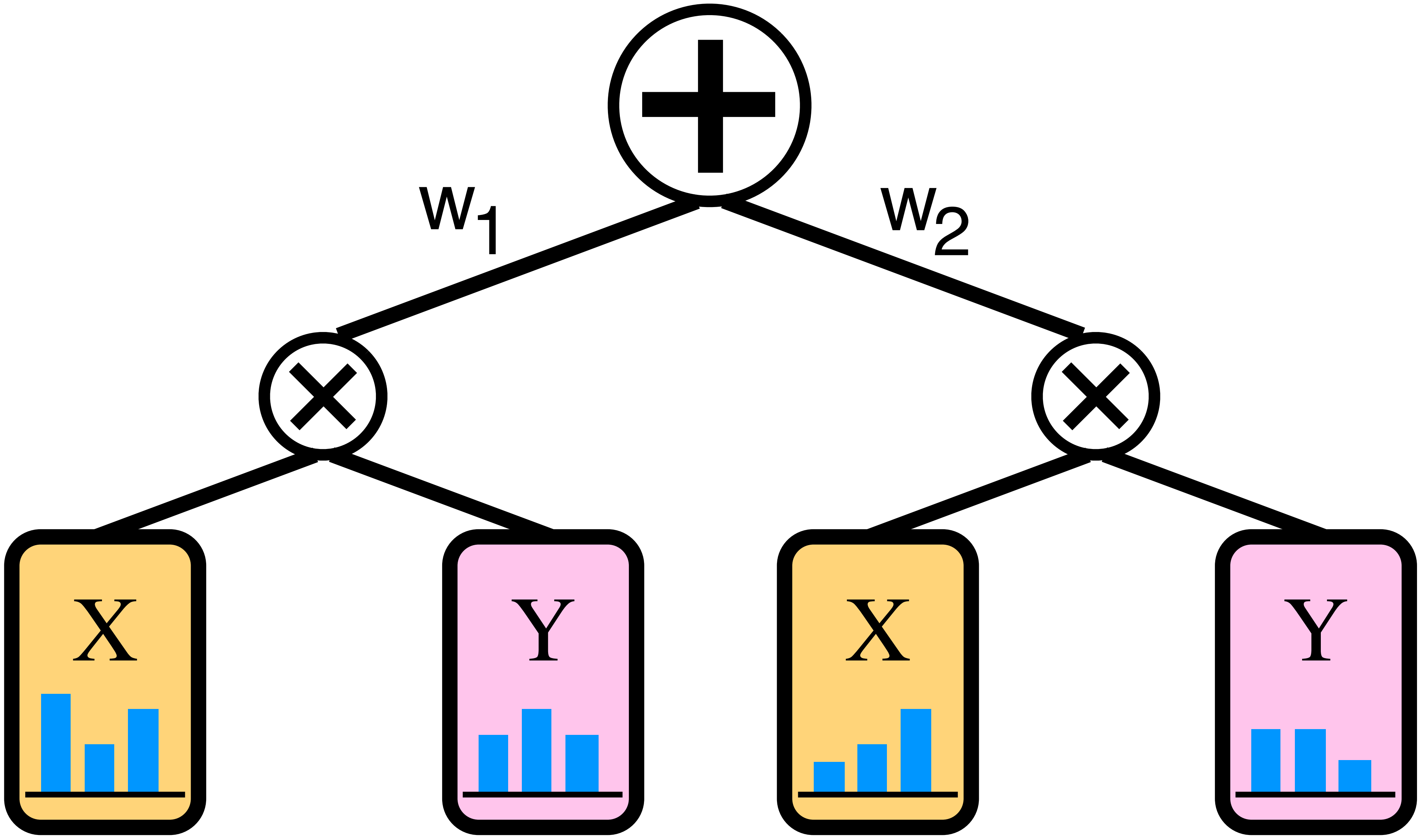
$X_1$ $X_1$ $X_1$ $X_1$ $X_2$ $X_2$ $X_2$ $X_2$ $X_3$ $X_3$ $X_3$ $X_3$ $X_4$ $X_4$ $X_4$ $X_4$ $X_5$ $X_5$ $X_5$ $X_5$ $X_6$ $X_6$ $X_6$ $X_6$

# All Marginals Are Computable in Linear Time

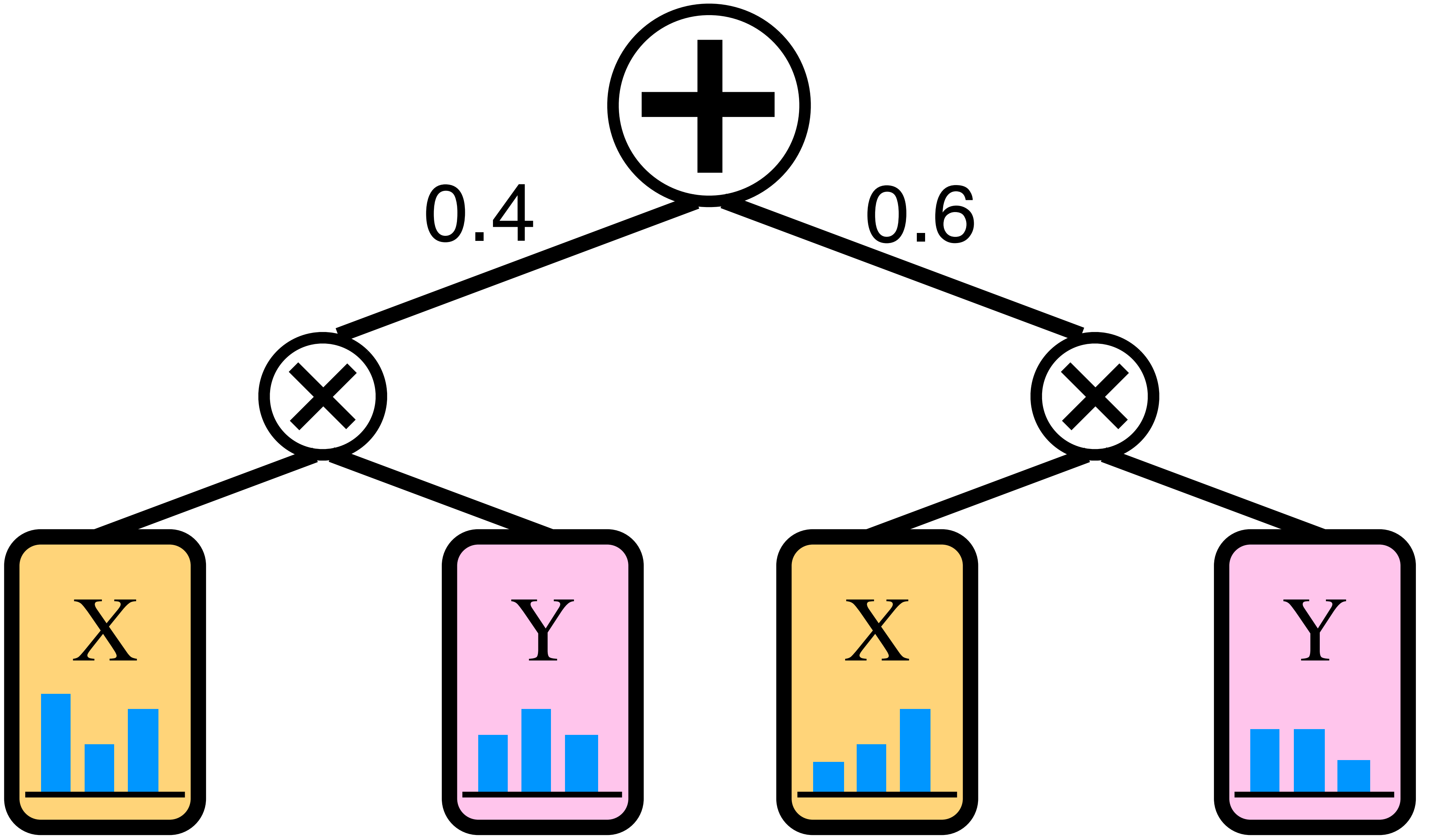
# All Marginals Are Computable in Linear Time



# All Marginals Are Computable in Linear Time

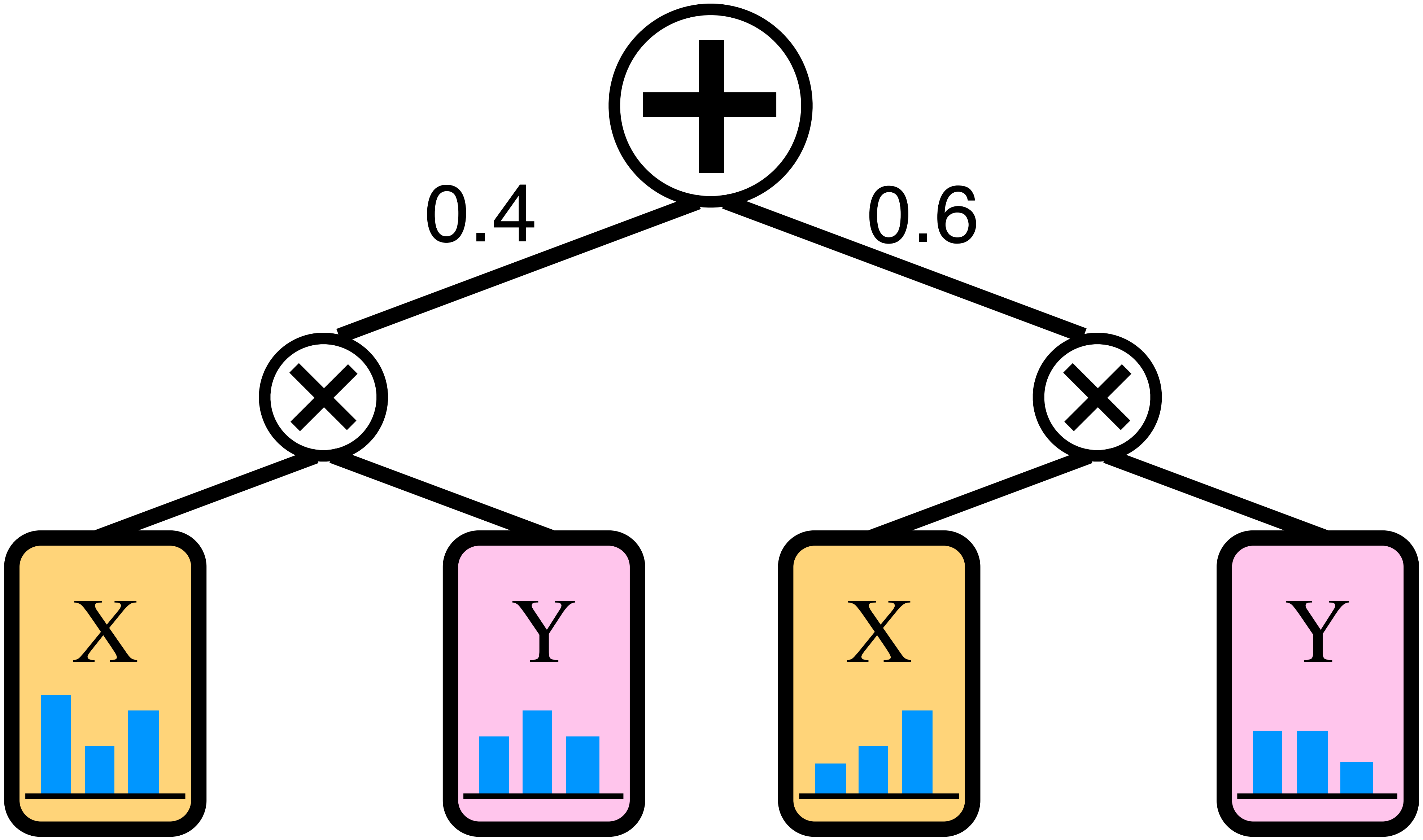


# All Marginals Are Computable in Linear Time



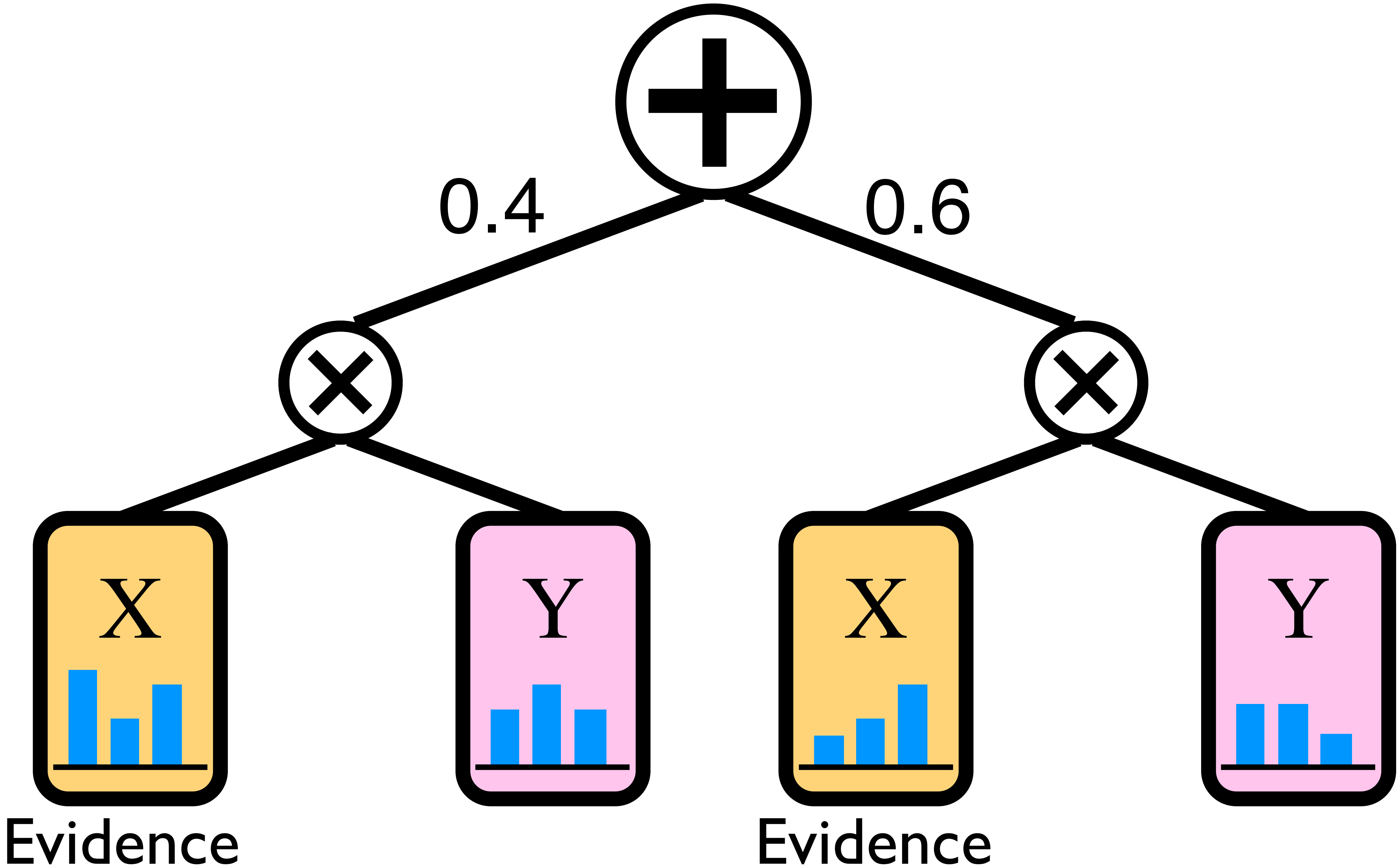
# All Marginals Are Computable in Linear Time

$P(X=0)$ ?



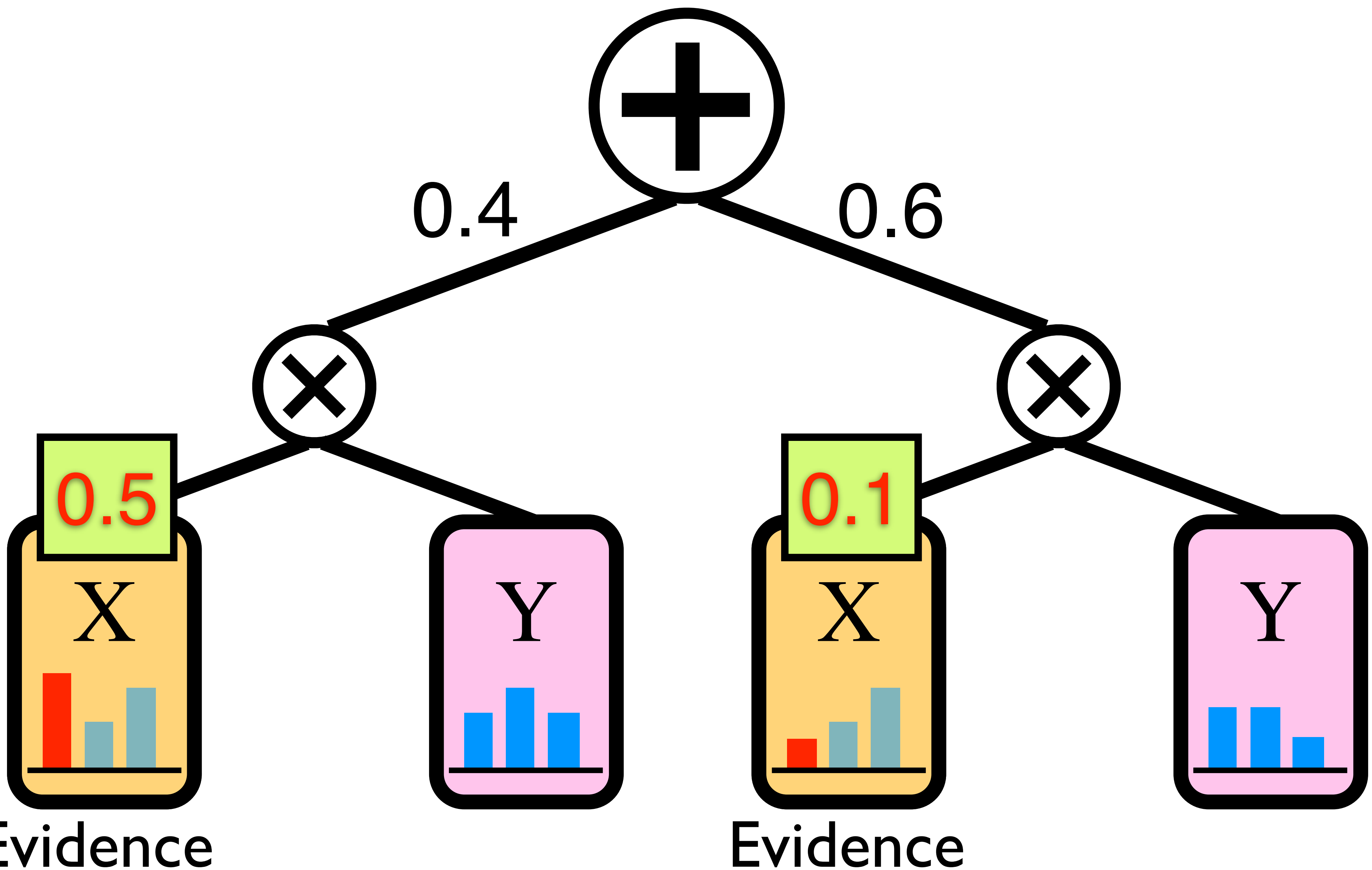
# All Marginals Are Computable in Linear Time

$P(X{=}0)$ ?



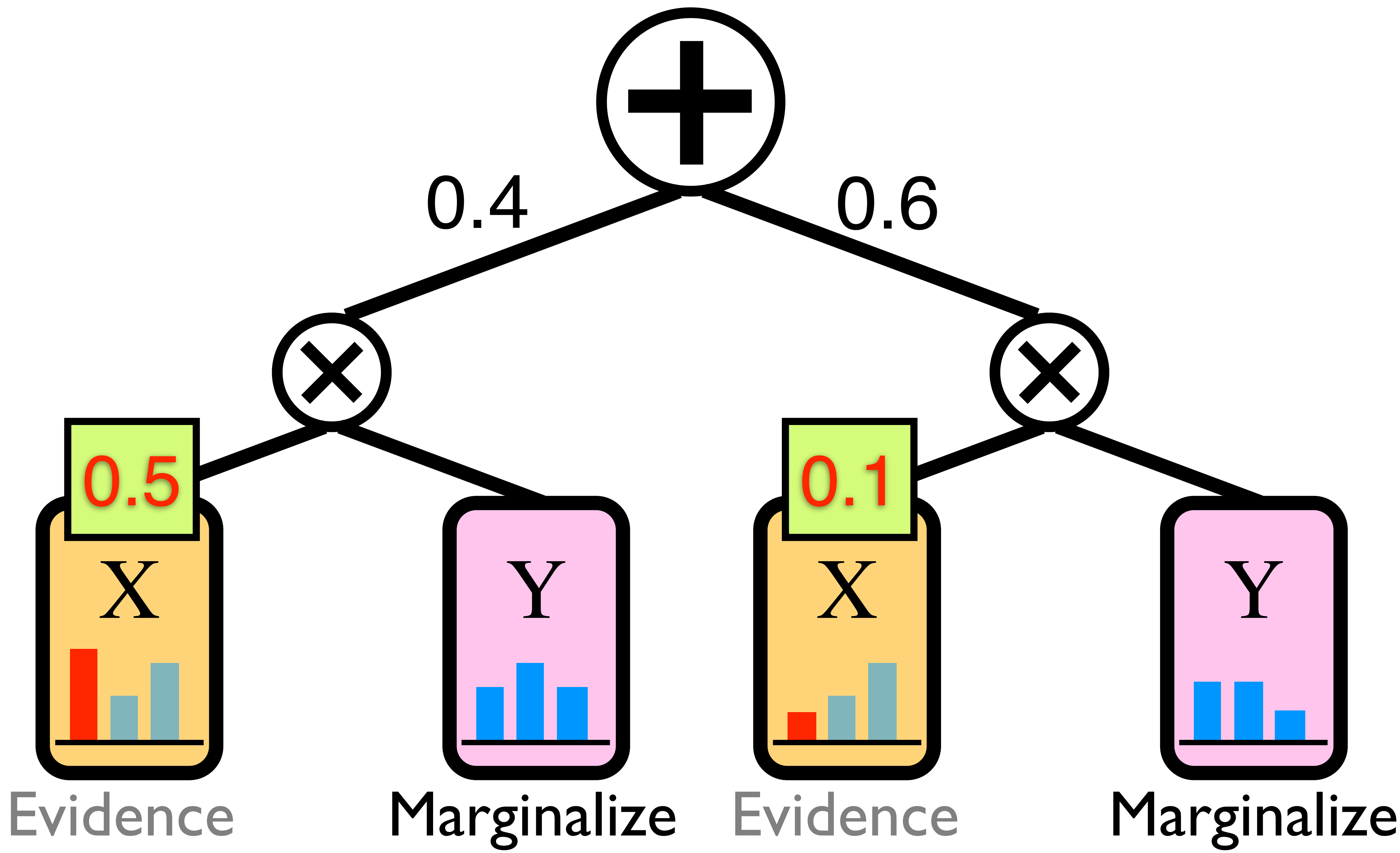
# All Marginals Are Computable in Linear Time

$P(X=0)$ ?



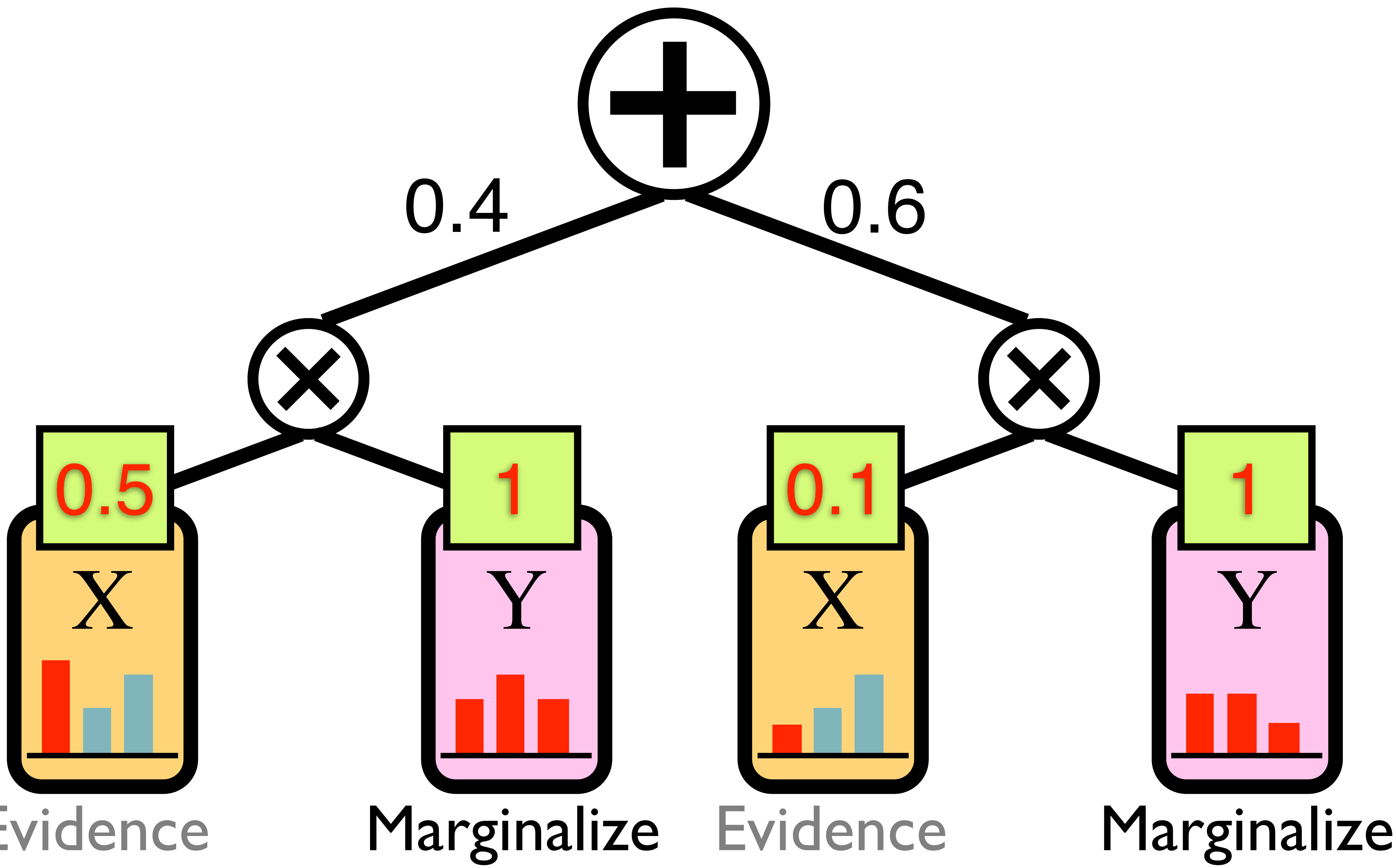
# All Marginals Are Computable in Linear Time

$P(X=0)$ ?



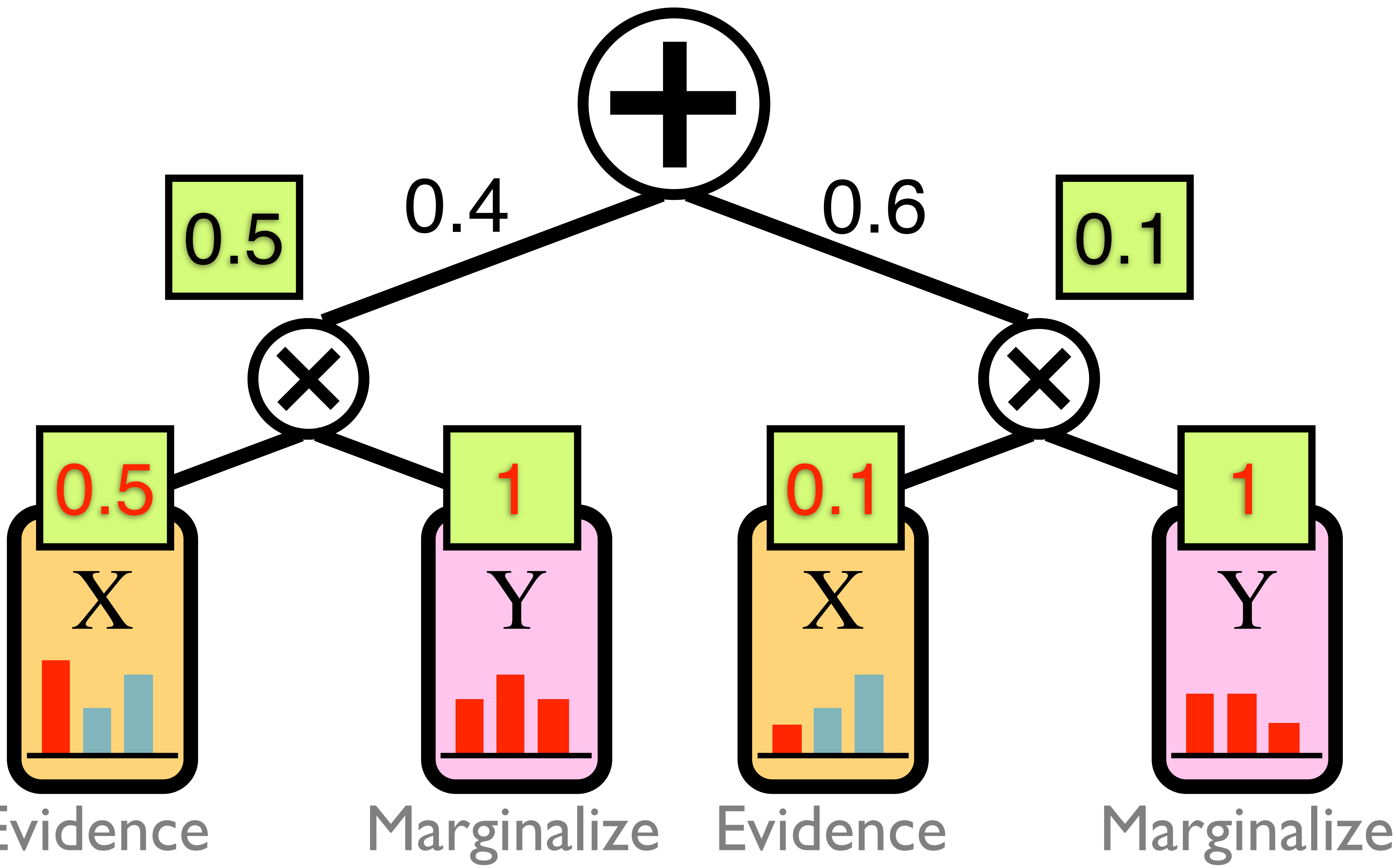
# All Marginals Are Computable in Linear Time

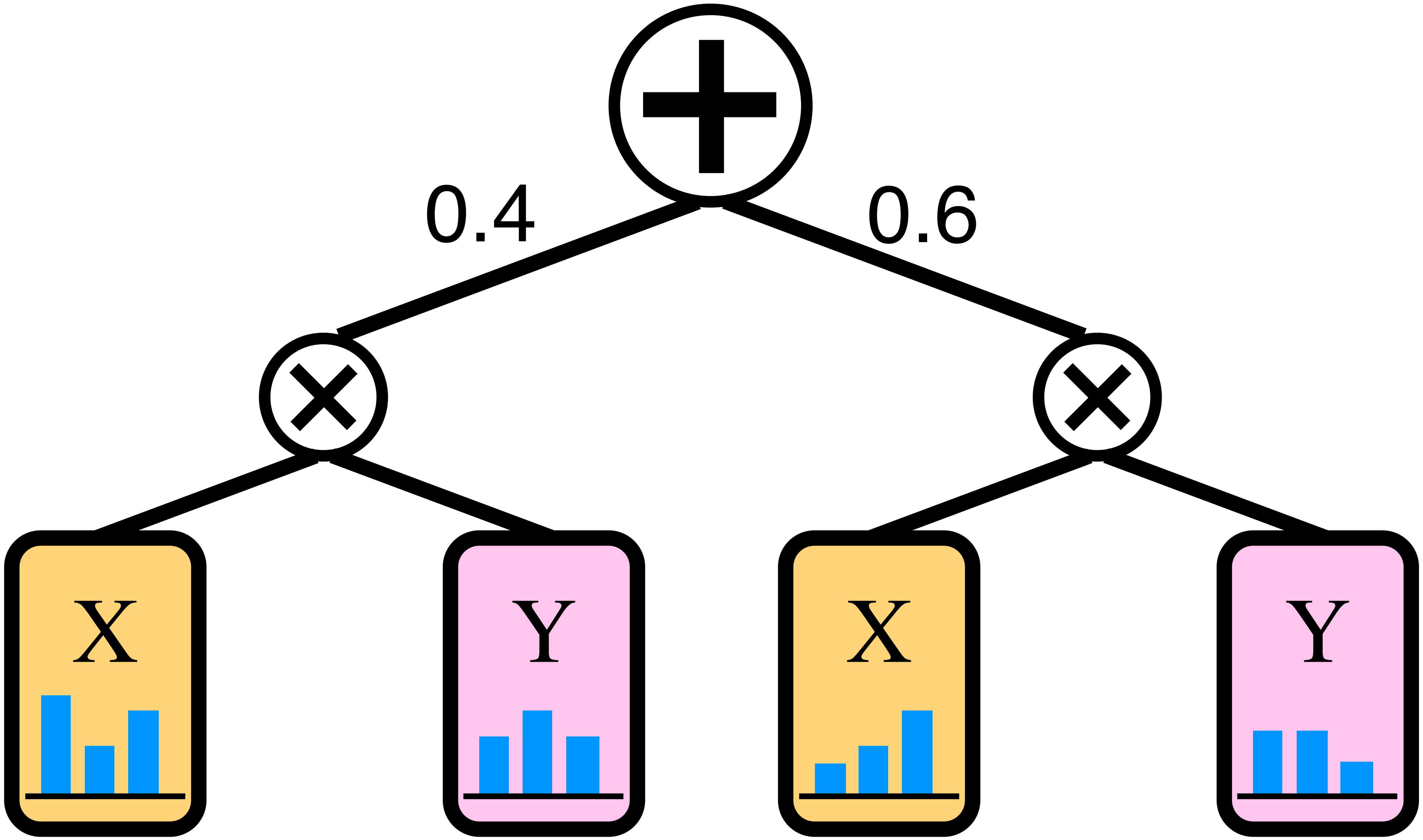
$P(X=0)$ ?



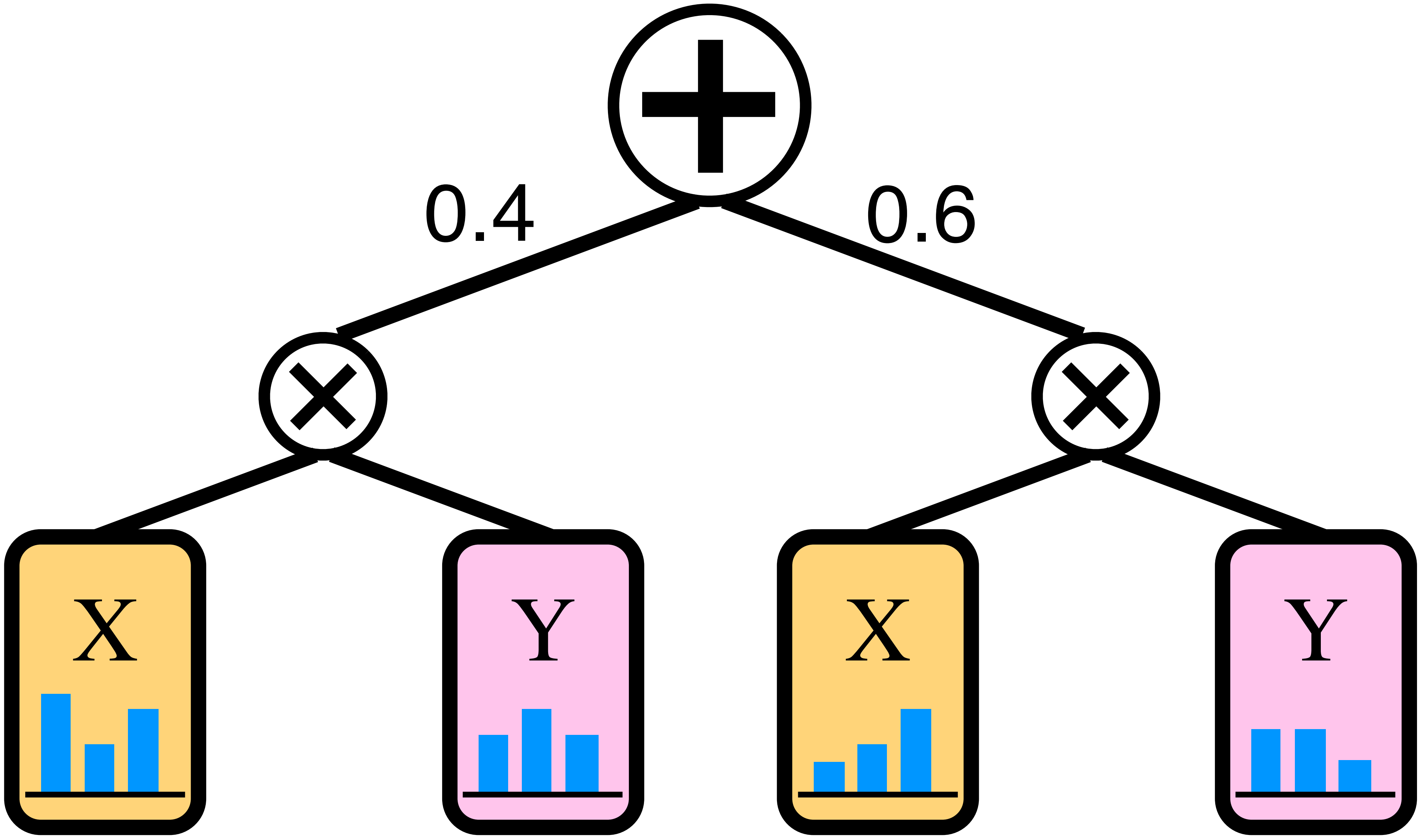
# All Marginals Are Computable in Linear Time

$P(X=0) =$ 0.26

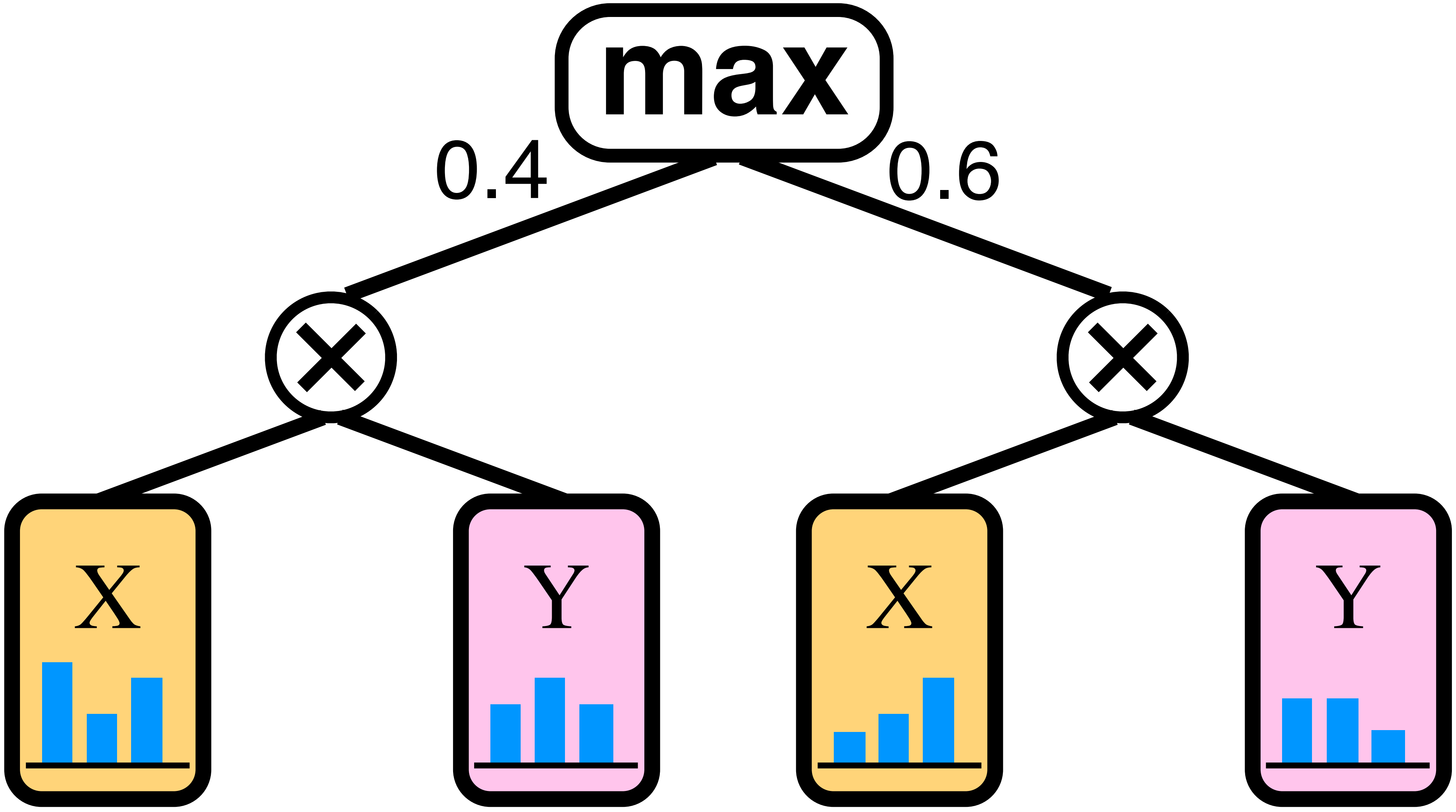



0.4
0.6
X
Y
X
Y

# All MAP States Are Computable in Linear Time

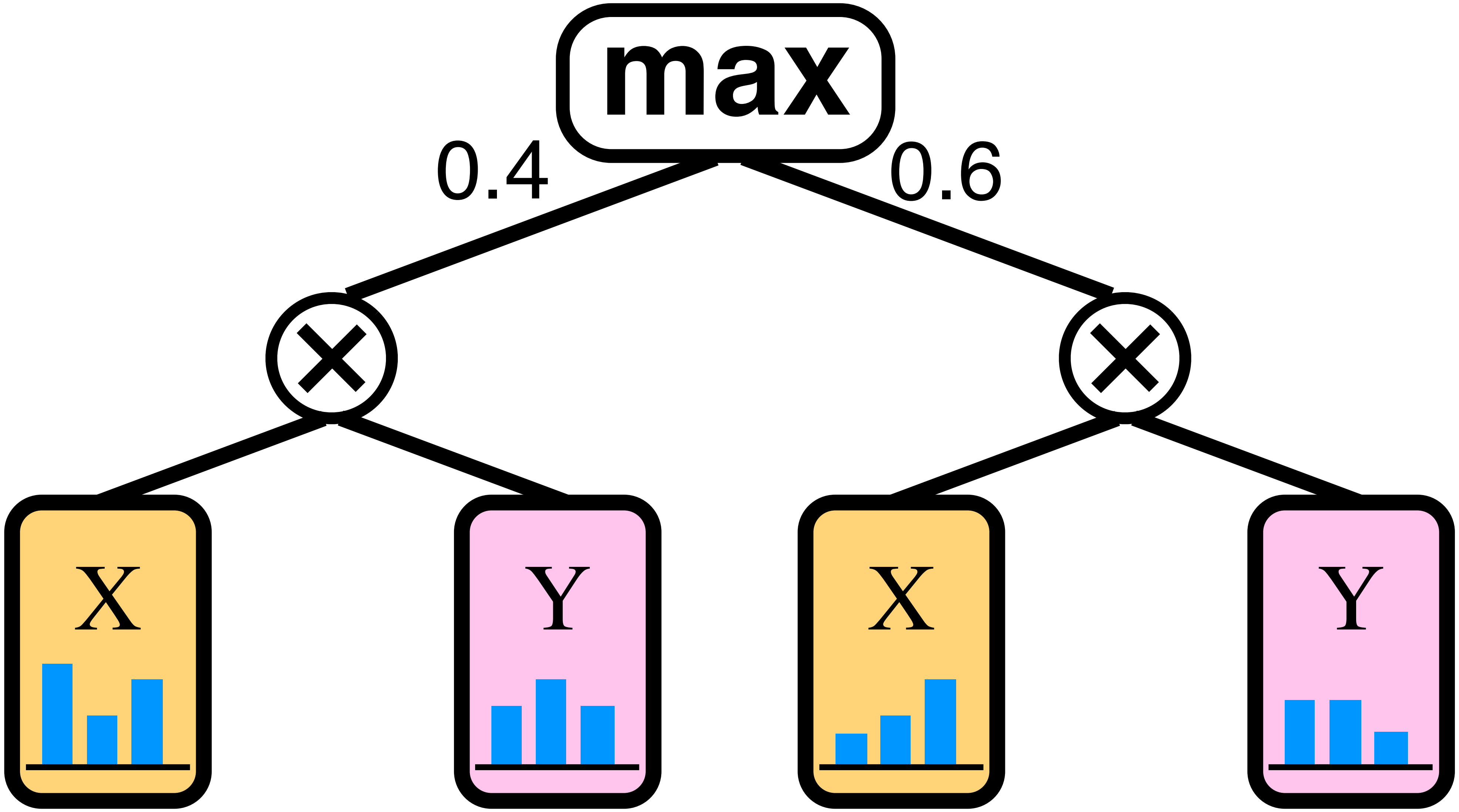


# All MAP States Are Computable in Linear Time



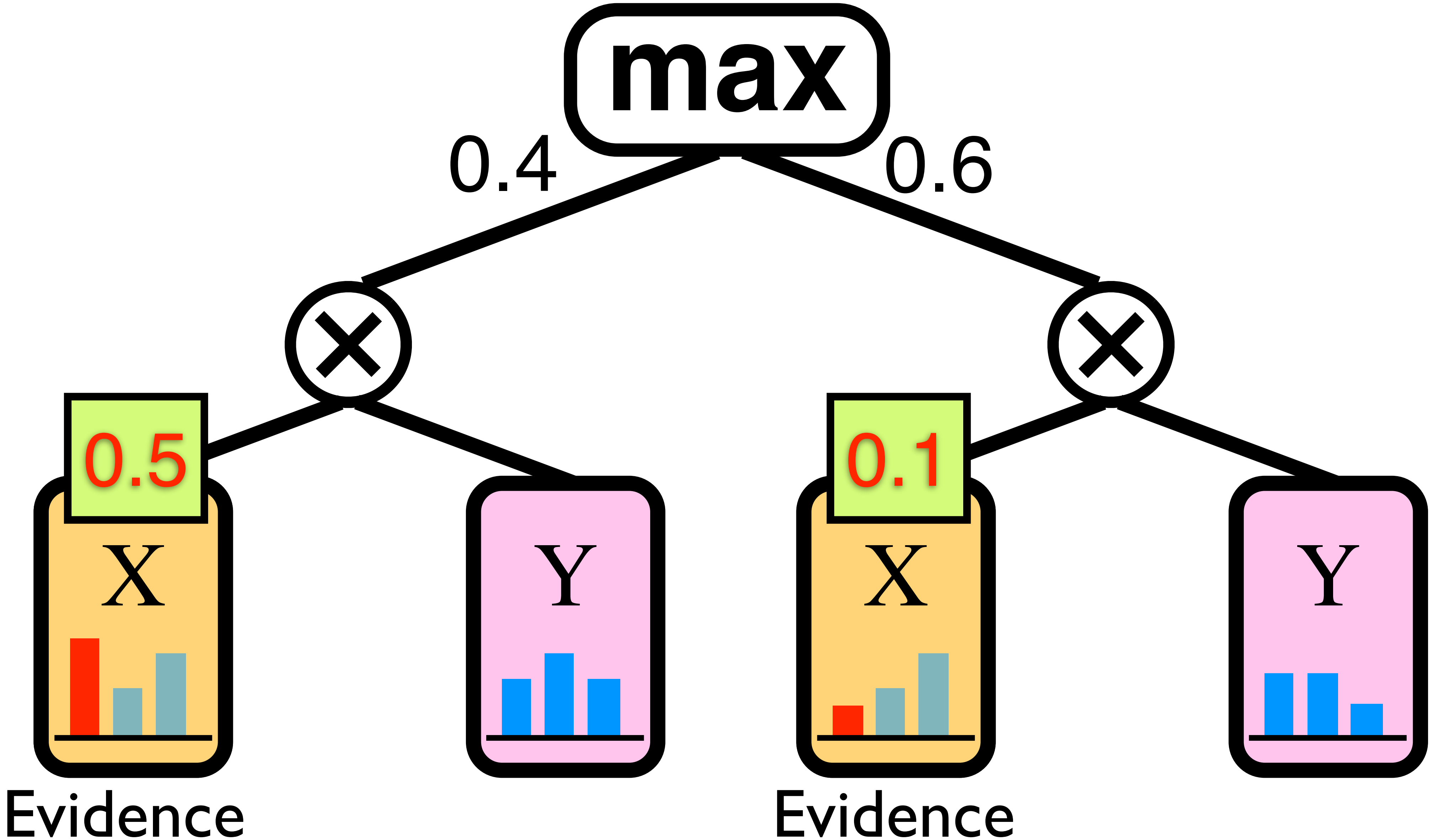
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h)$$ ?



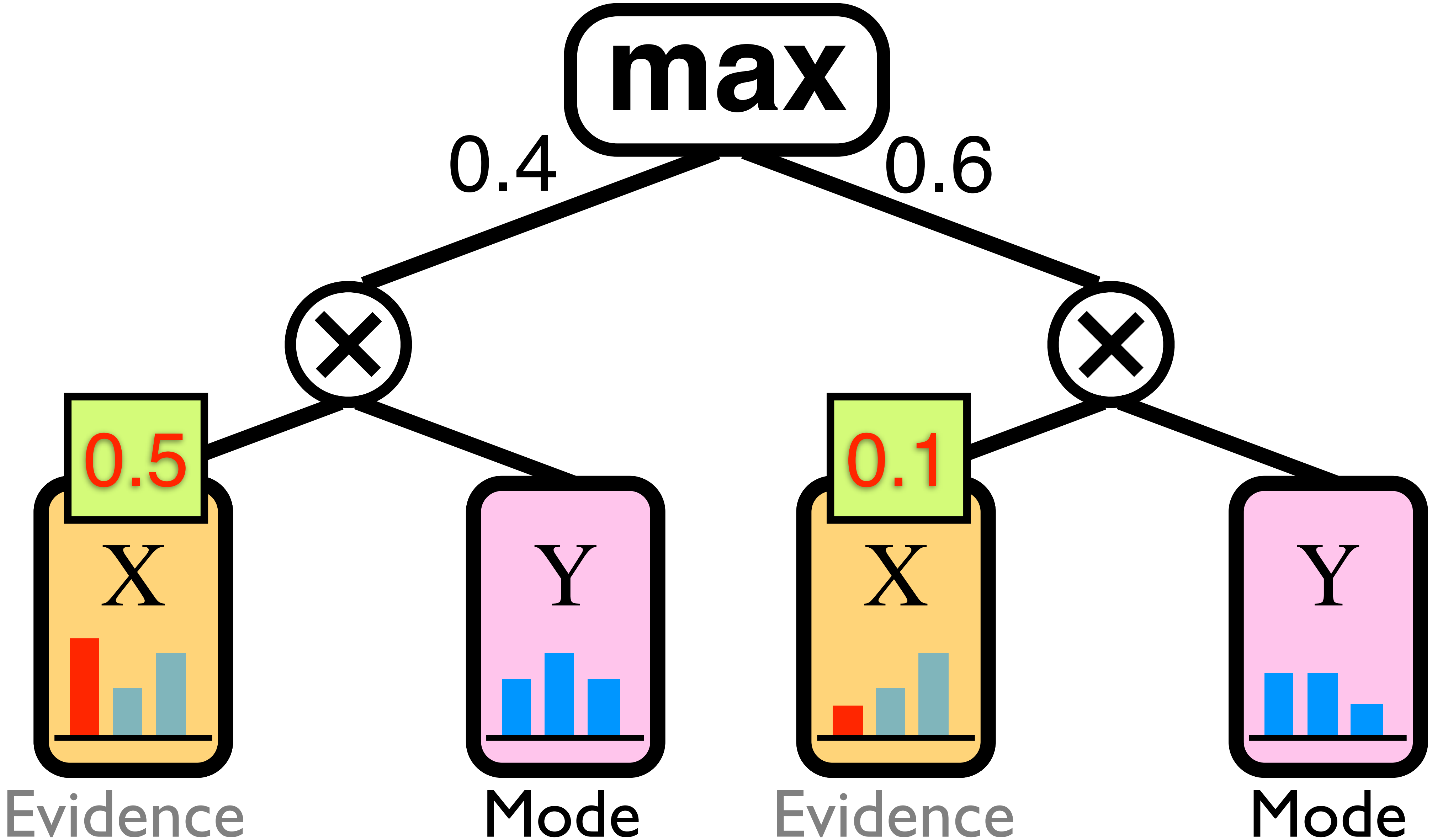
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X = 0, Y = y, H = h)\ ?$$



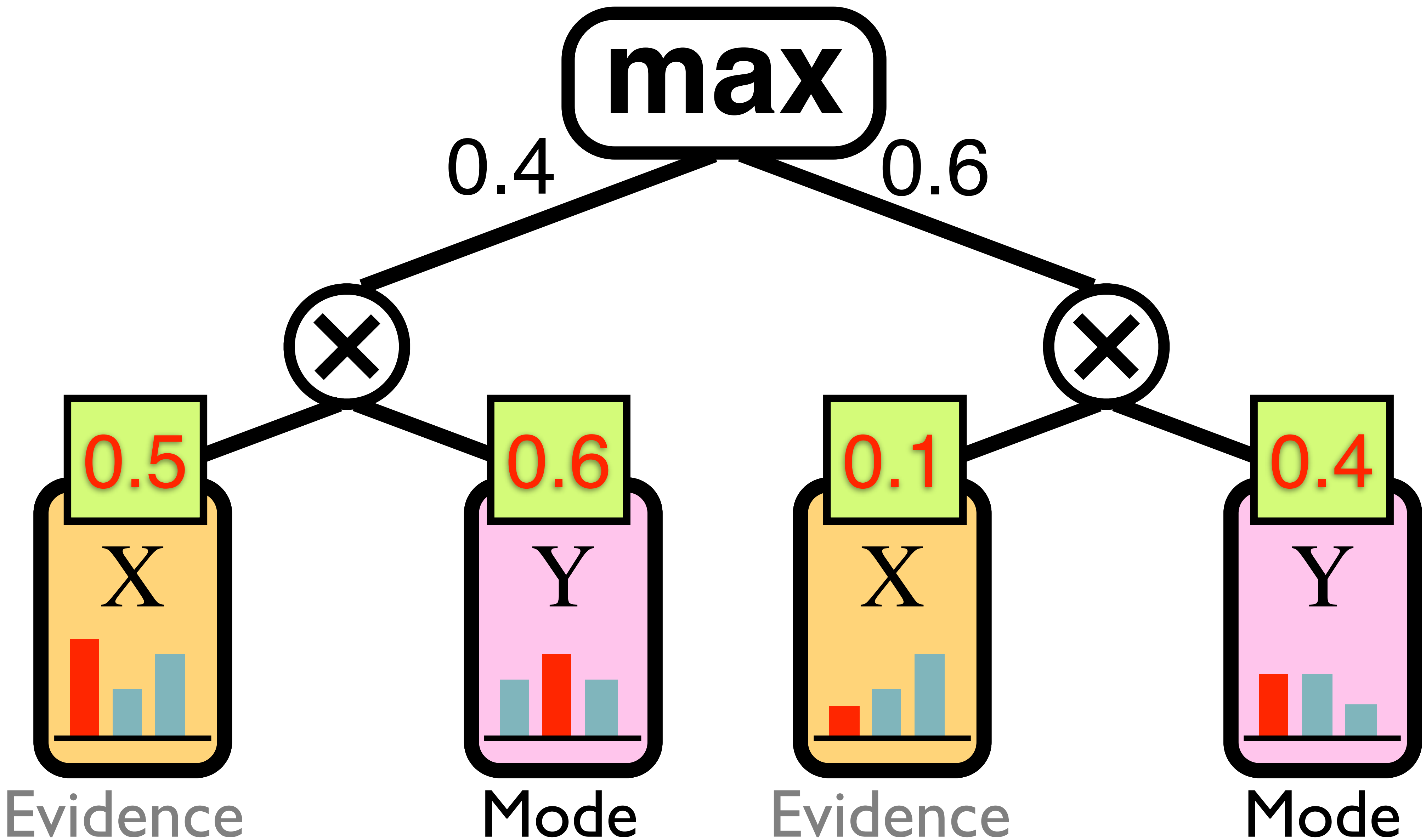
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h)\ ?$$



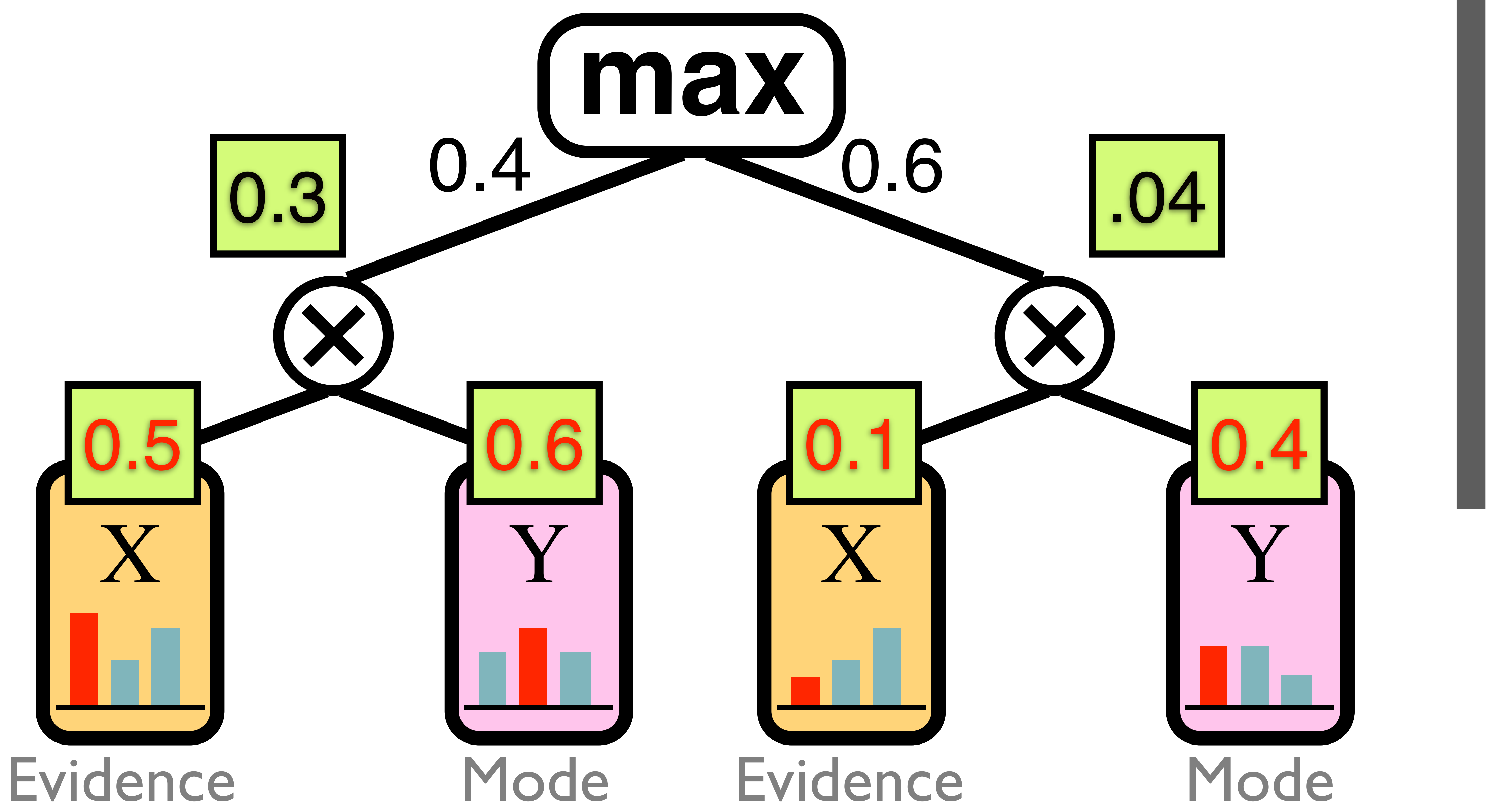
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h) \text{ ?}$$



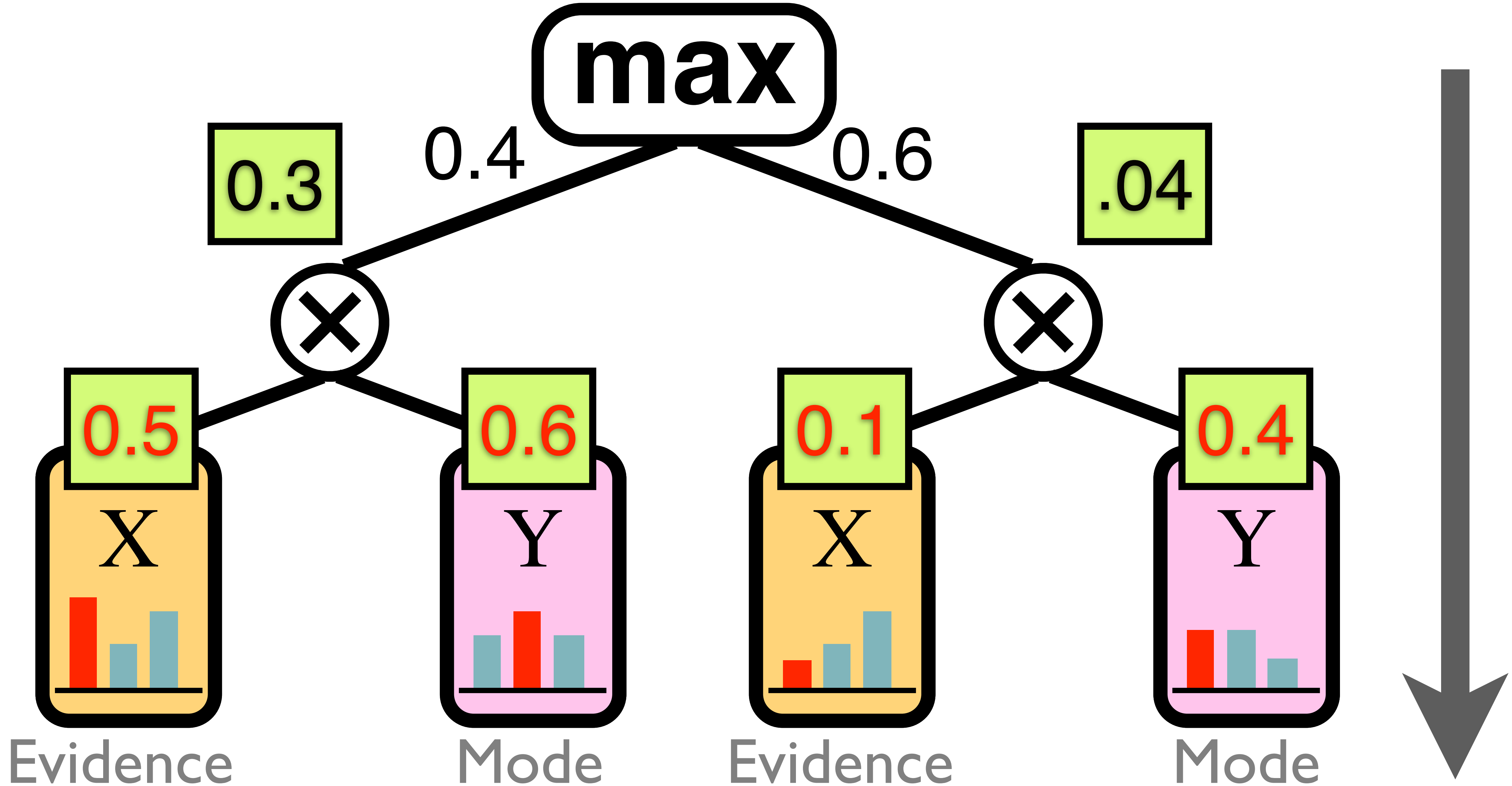
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h) = 0.12$$



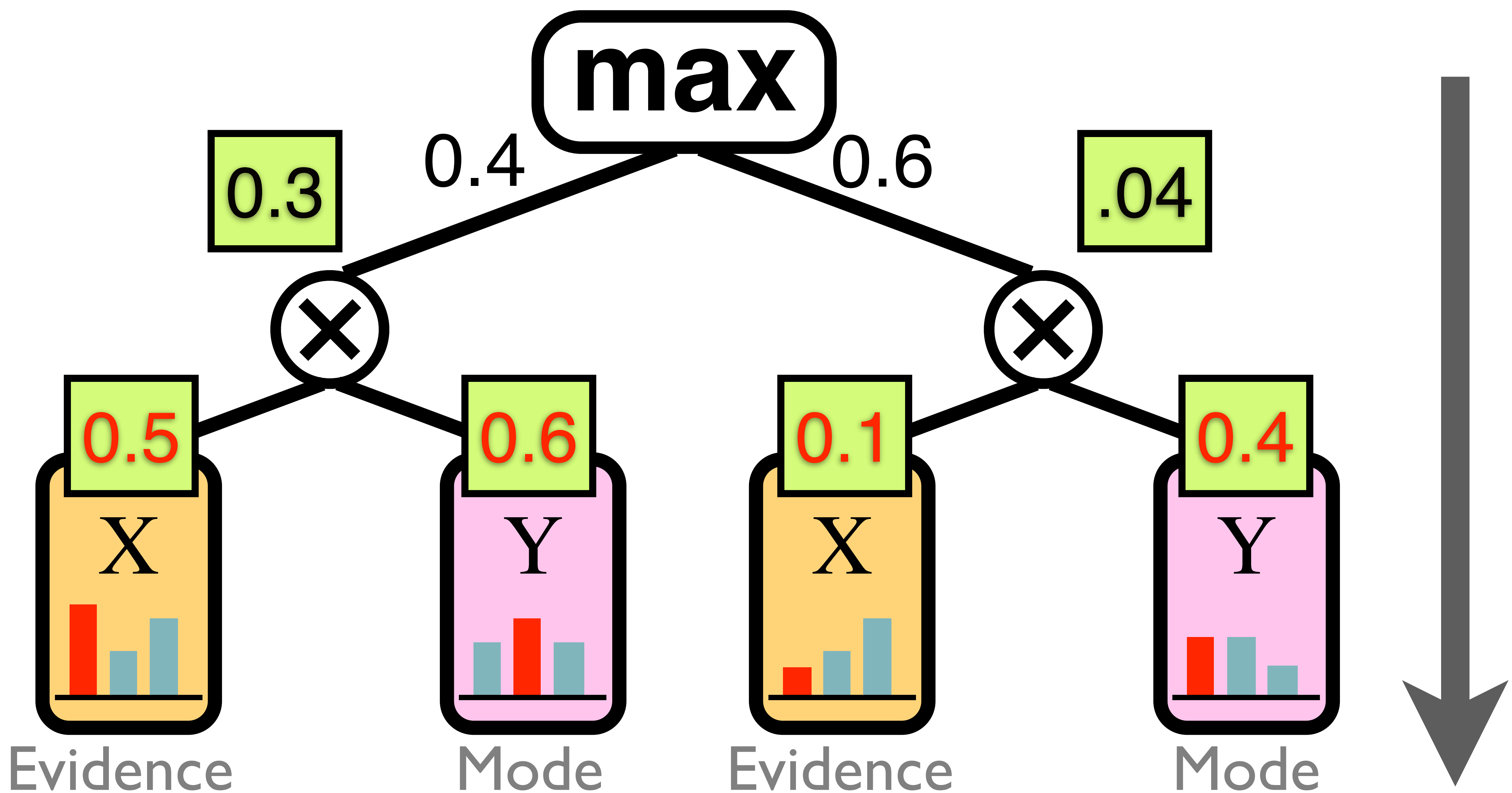
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h) = 0.12$$



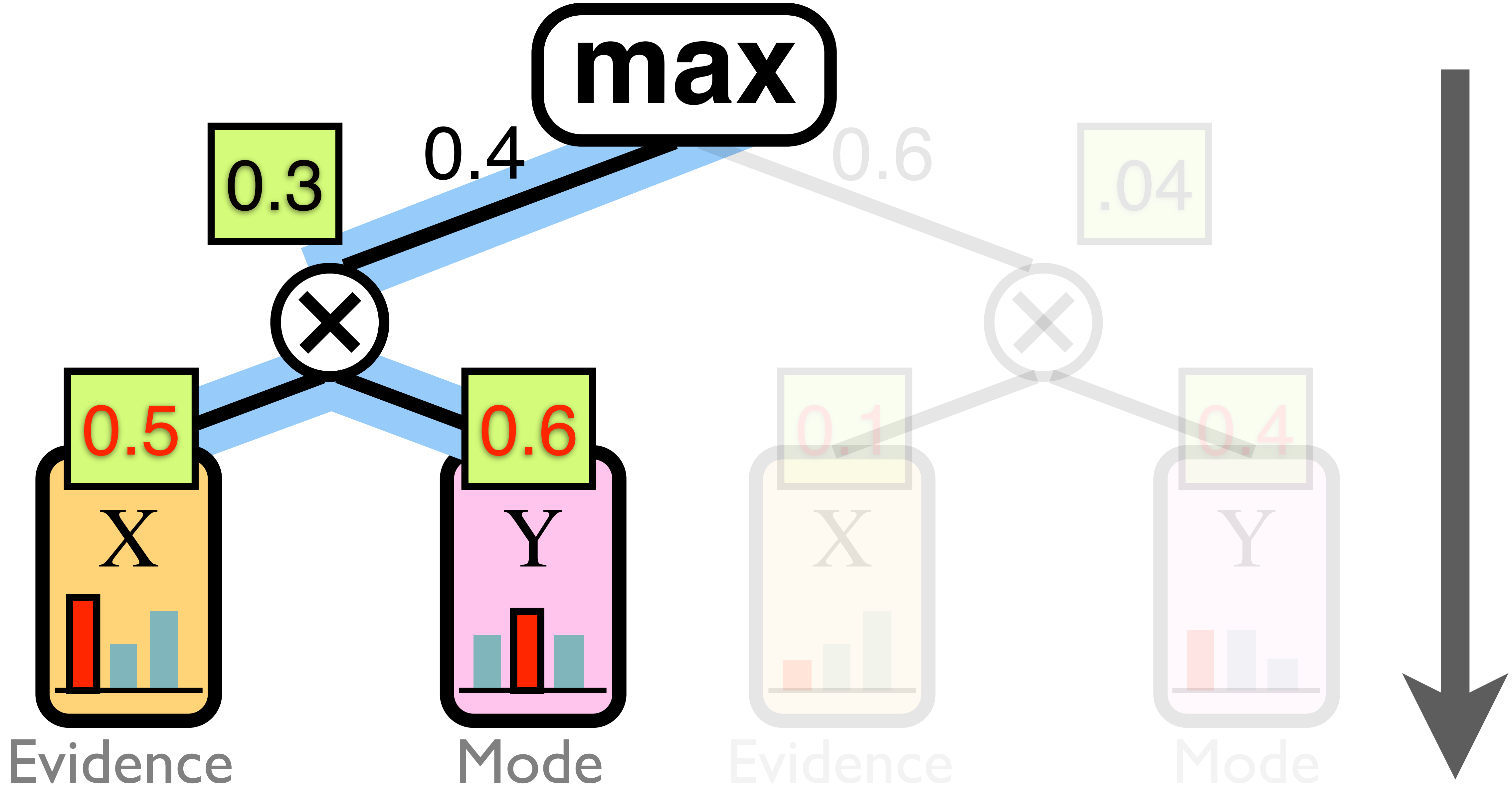
# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h) = 0.12$$



# All MAP States Are Computable in Linear Time

$$\max_{y,h} P(X=0, Y=y, H=h) = 0.12$$



# What Does an SPN Mean?

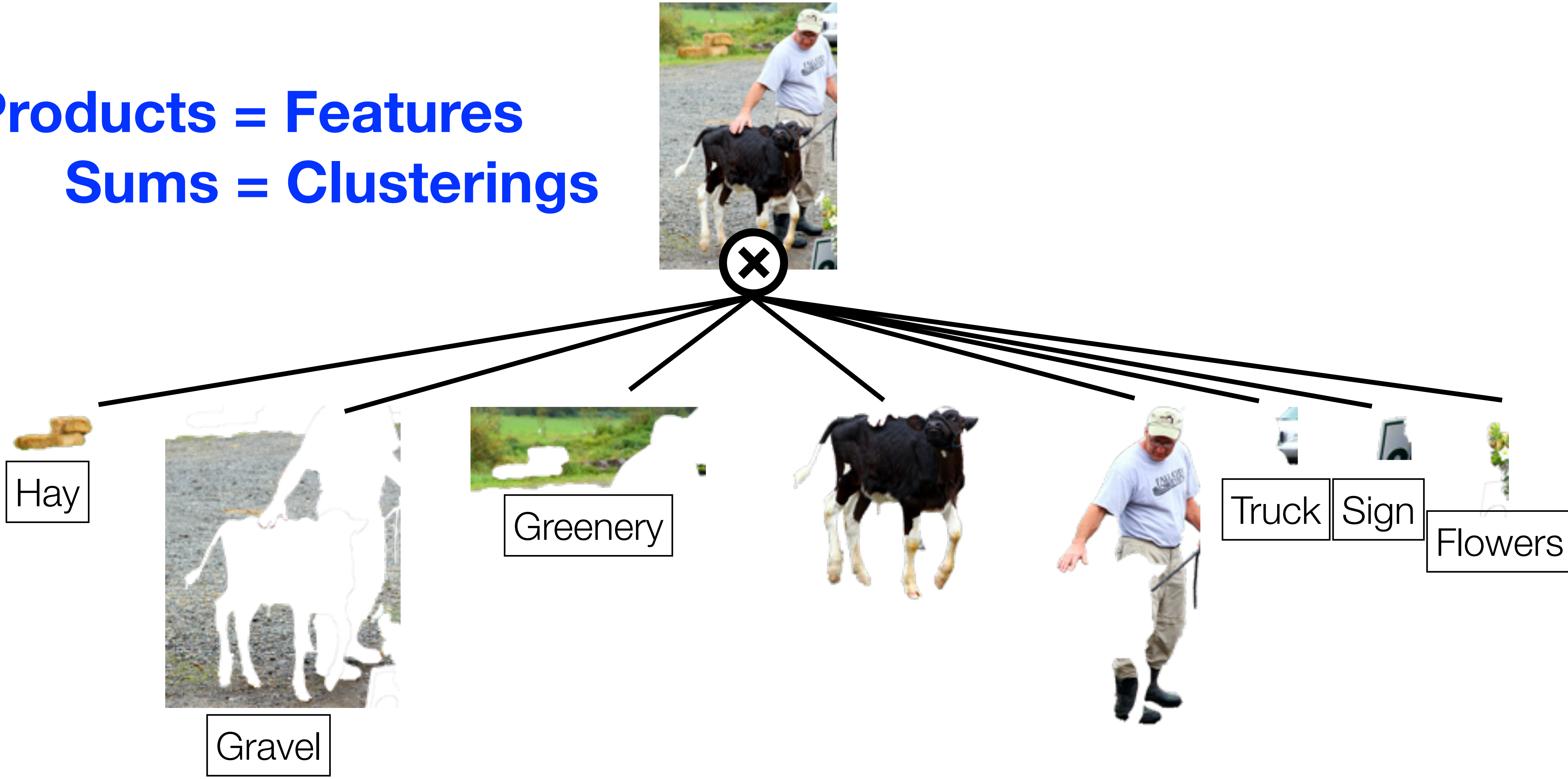
**Products = Features**

**Sums = Clusterings**

# What Does an SPN Mean?

**Products = Features**

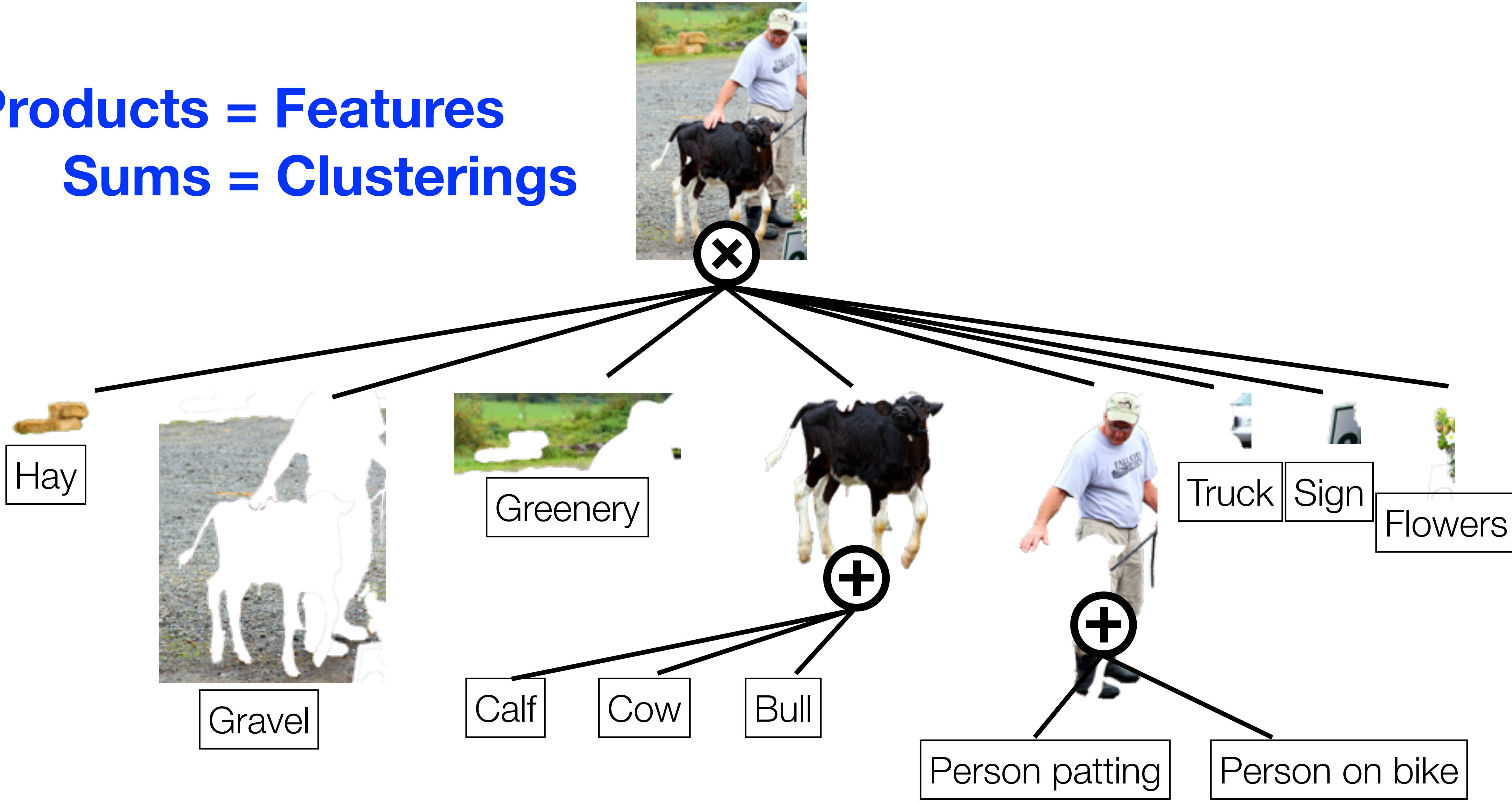
**Sums = Clusterings**



# What Does an SPN Mean?

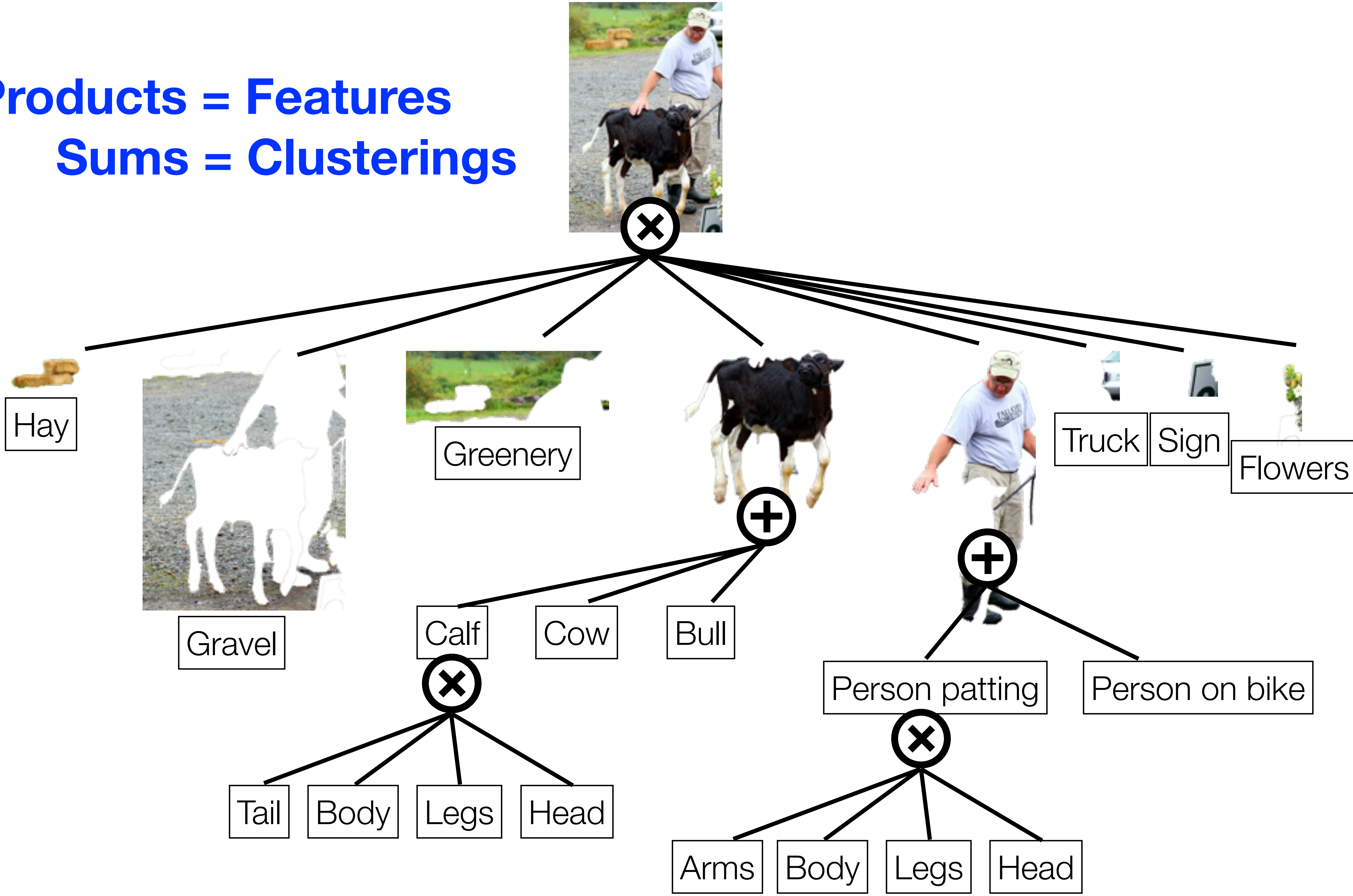
**Products = Features**

**Sums = Clusterings**



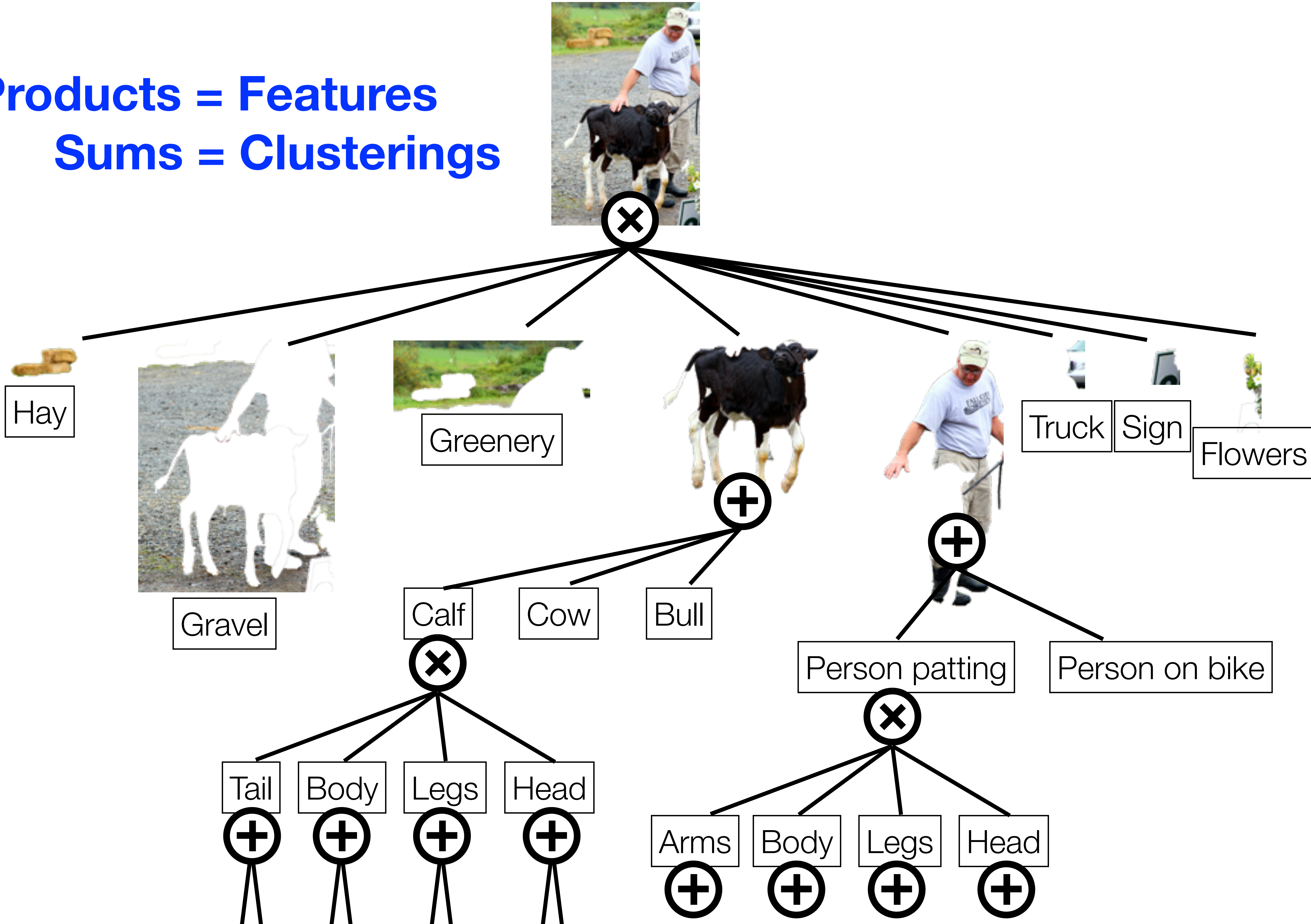
# What Does an SPN Mean?

**Products = Features**
**Sums = Clusterings**



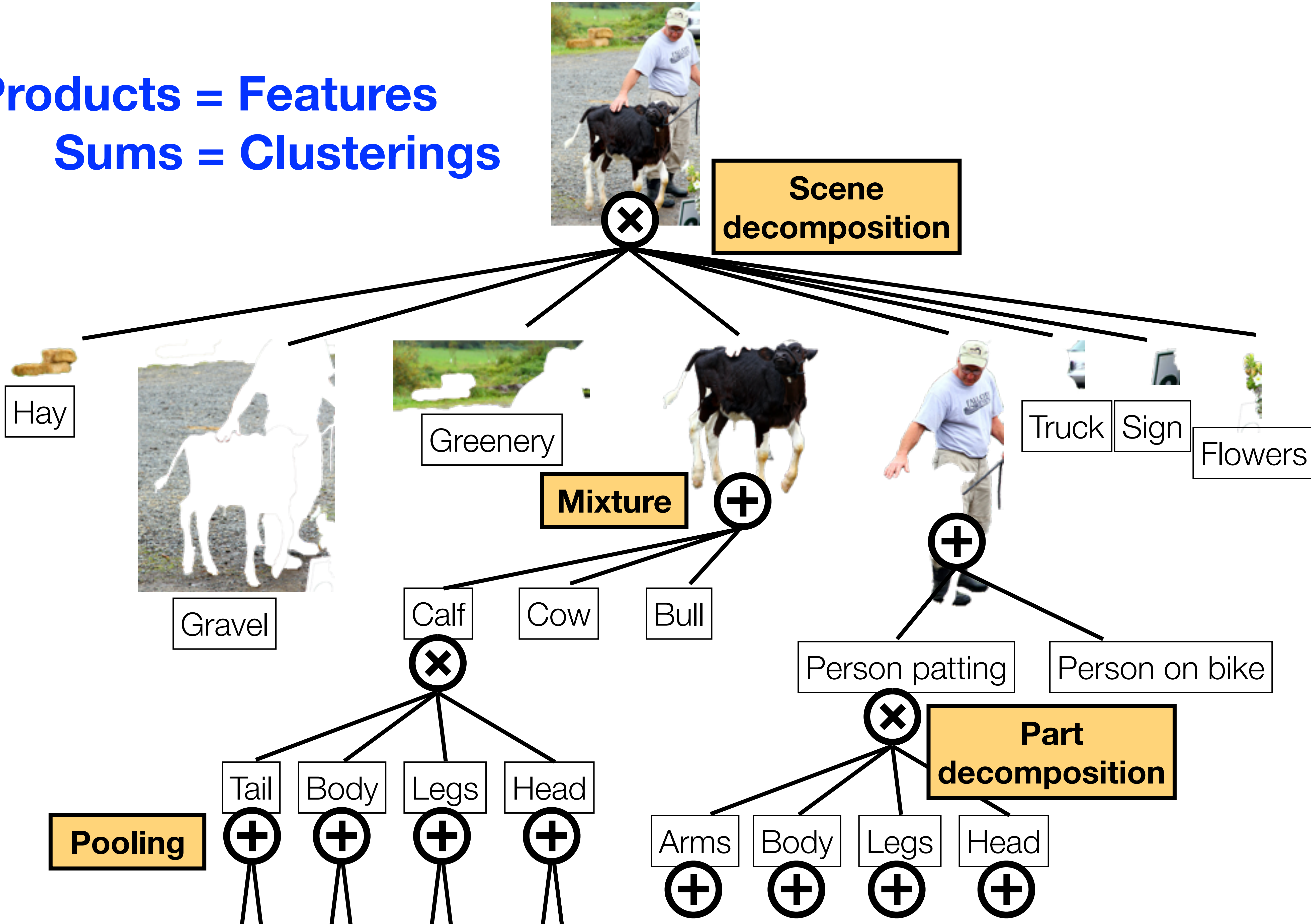
# What Does an SPN Mean?

**Products = Features**
**Sums = Clusterings**



# What Does an SPN Mean?

**Products = Features**
**Sums = Clusterings**



# Special Cases

- Hierarchical mixture models
- Thin junction trees (e.g.: hidden Markov models)
- Non-recursive probabilistic context-free grammars
- Etc.

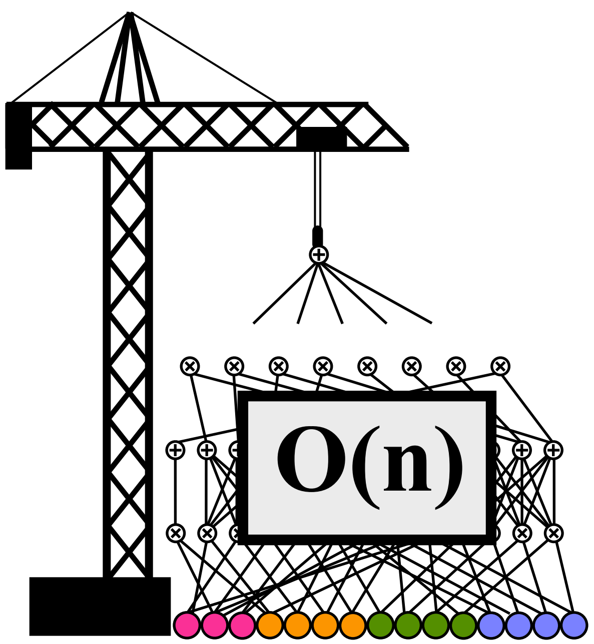
# Weight Learning

- **Generative** (Poon & Domingos, UAI 2011)
  UAI 2011 Best Paper Award
- **Discriminative** (Gens & Domingos, NIPS 2012)
  NIPS 2012 Outstanding Student Paper Award

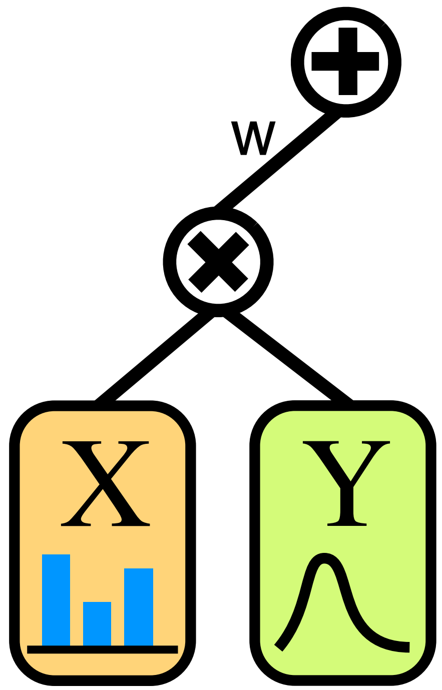
# Weight Learning

- **Generative** (Poon & Domingos, UAI 2011)
  UAI 2011 Best Paper Award
- **Discriminative** (Gens & Domingos, NIPS 2012)
  NIPS 2012 Outstanding Student Paper Award

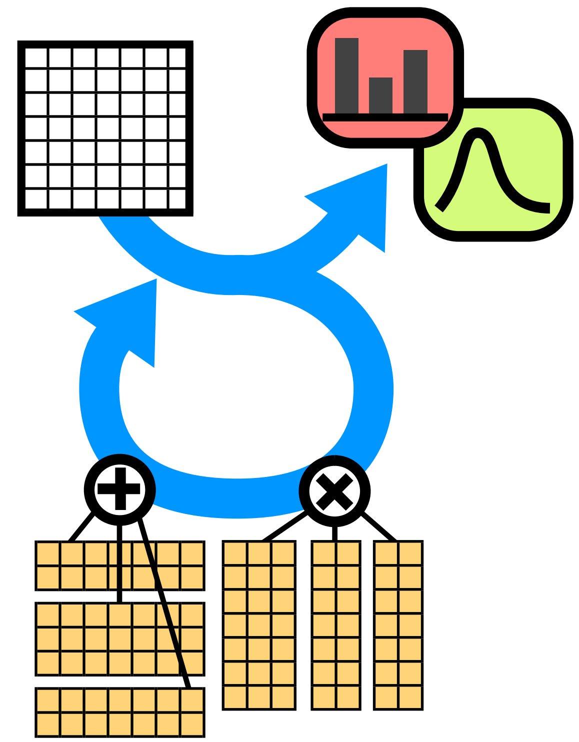
**Key limitation:** Requires a structure



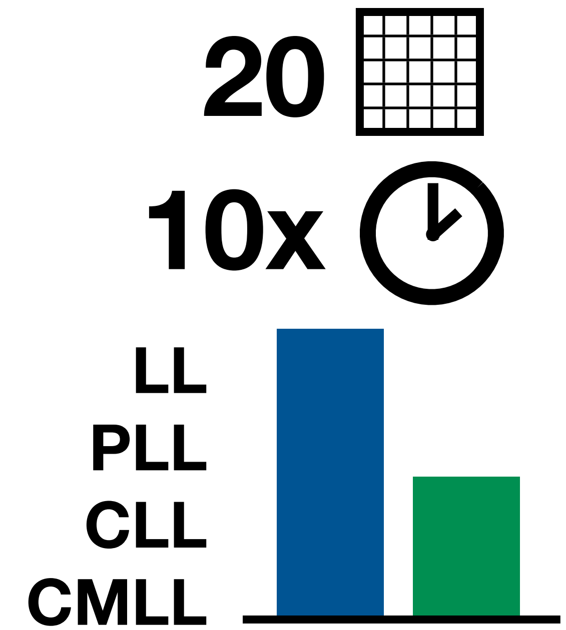

**Motivation**




**SPN Review**



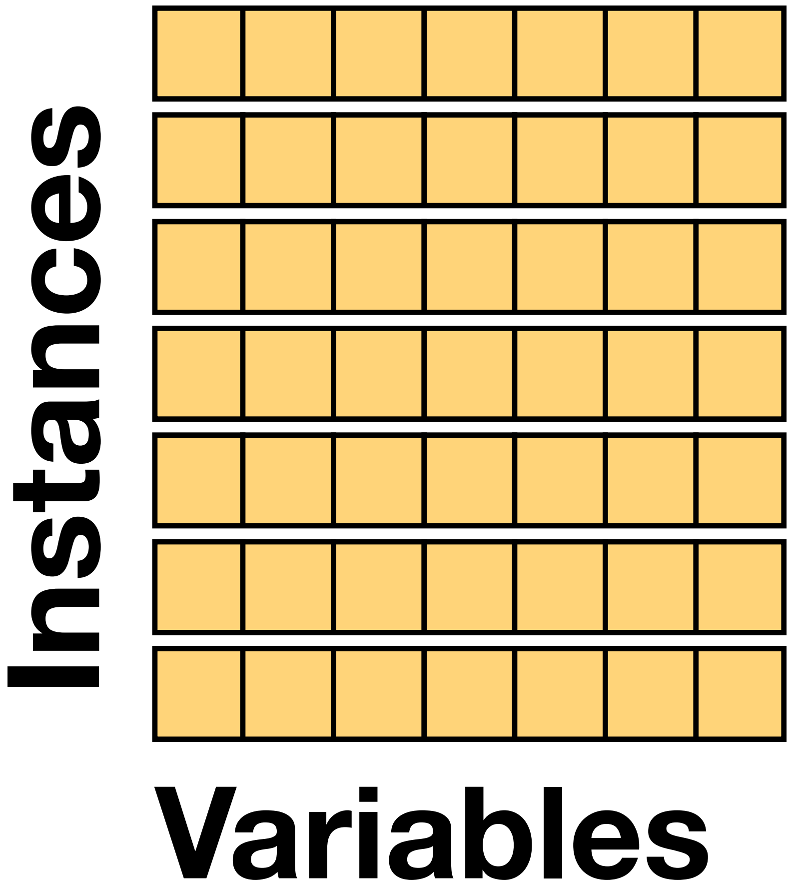
**Structure Learning**




**Experiments**

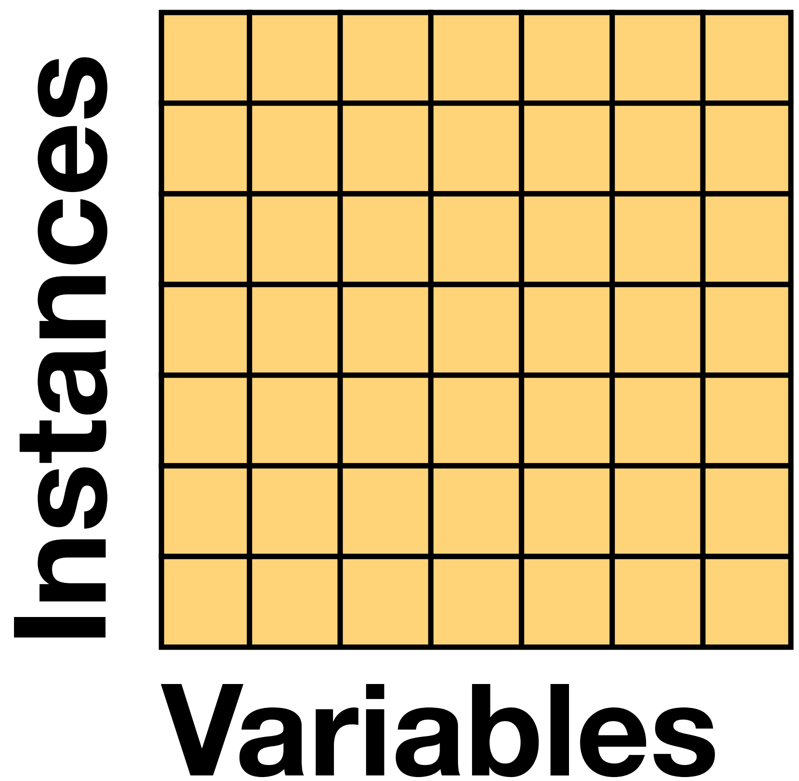
# LearnSPN

Recursive algorithm

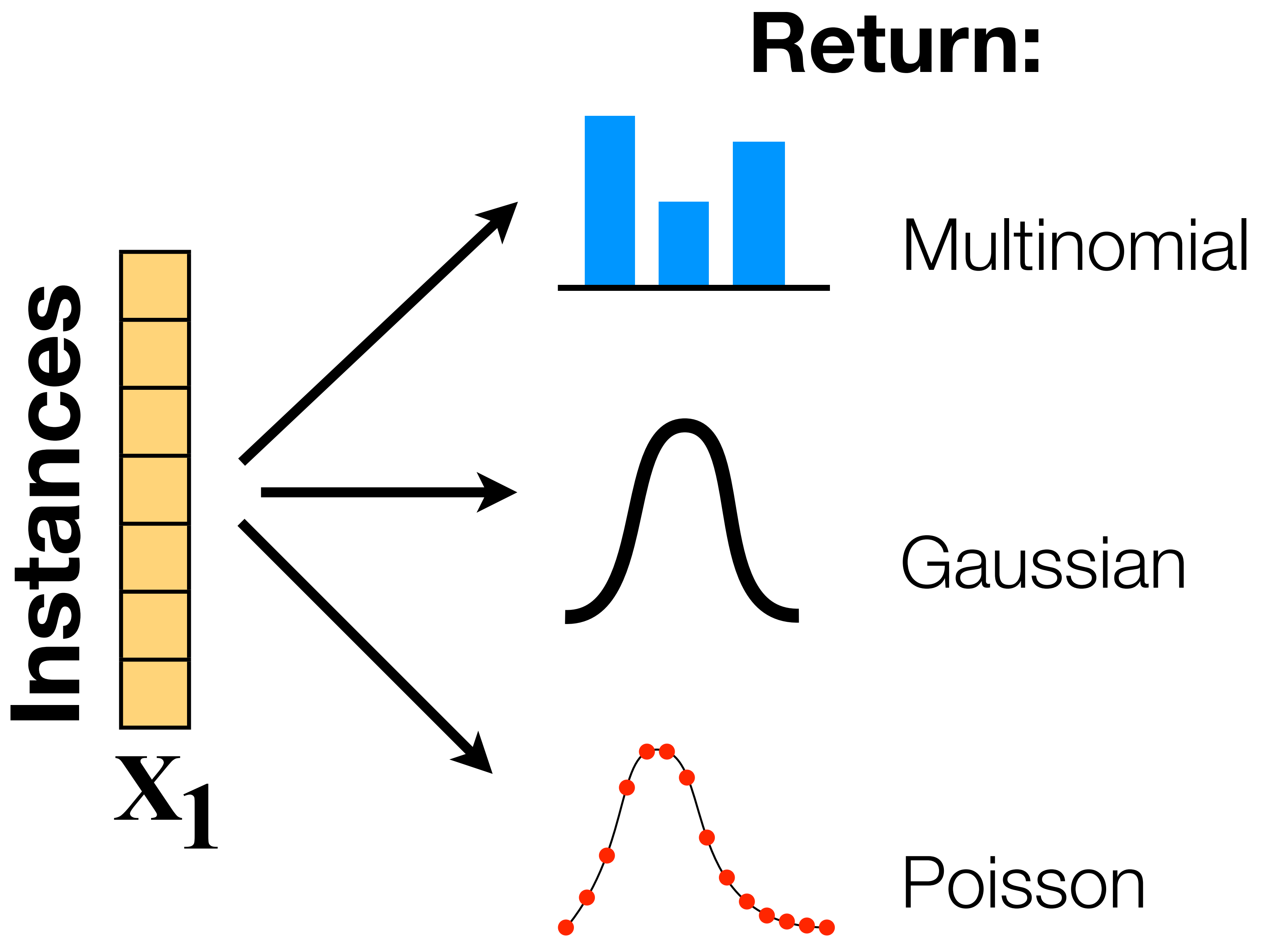


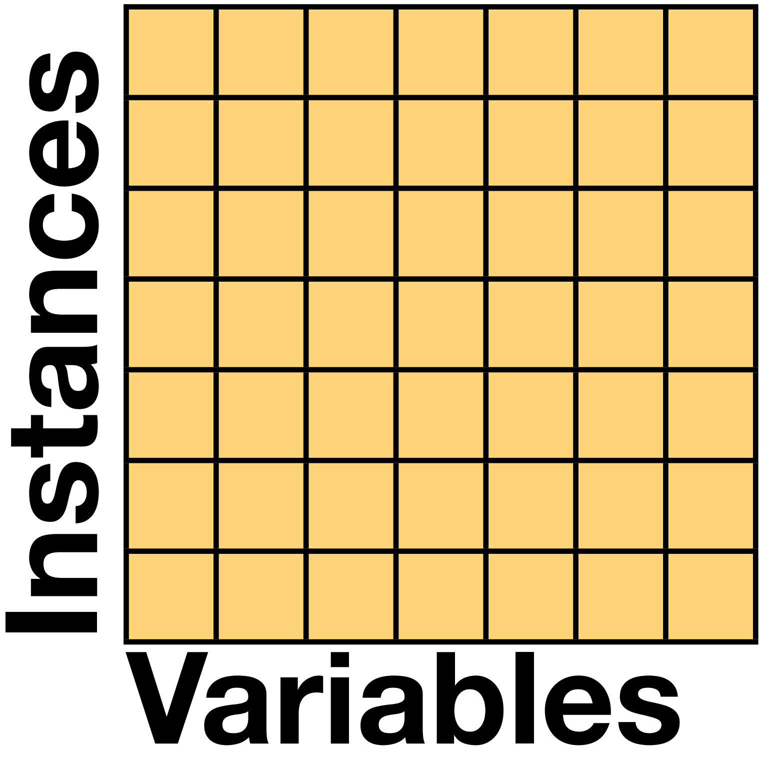
# LearnSPN

Recursive algorithm

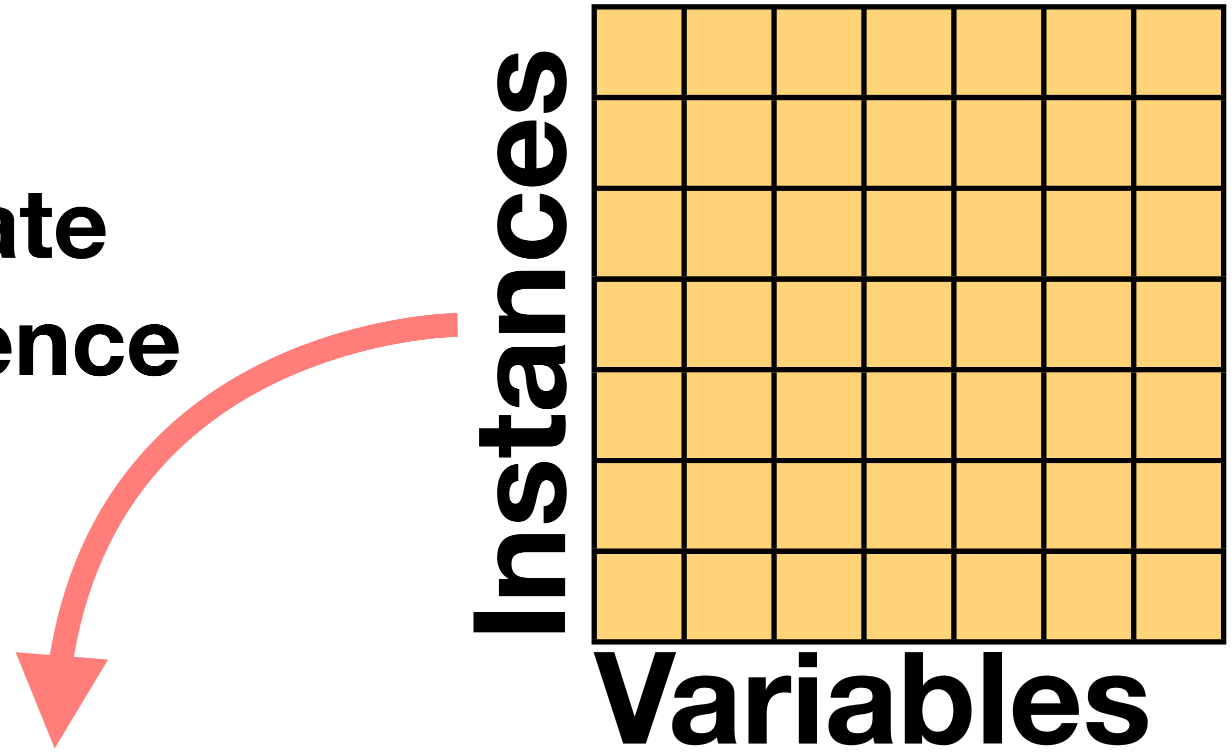


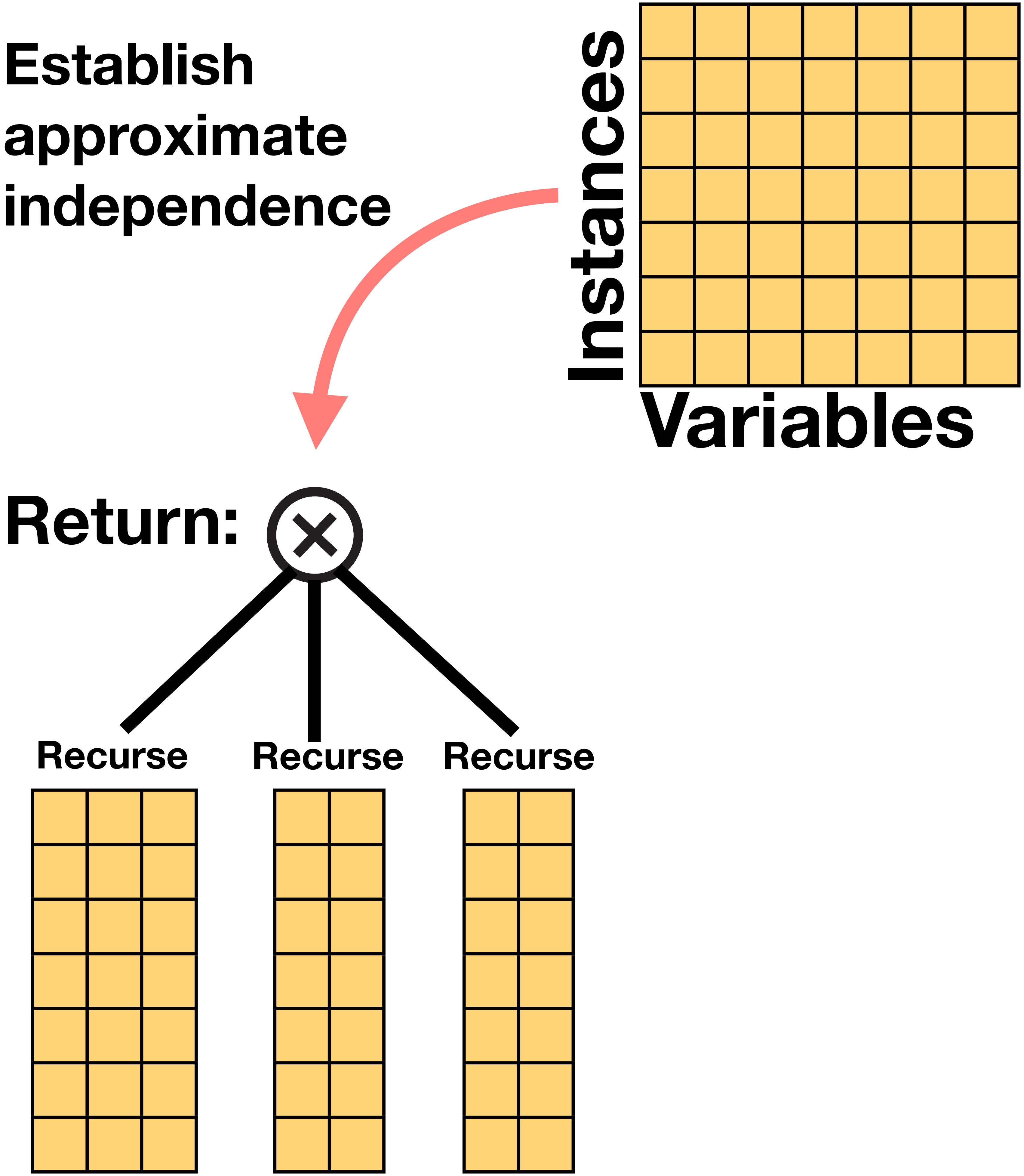
# LearnSPN

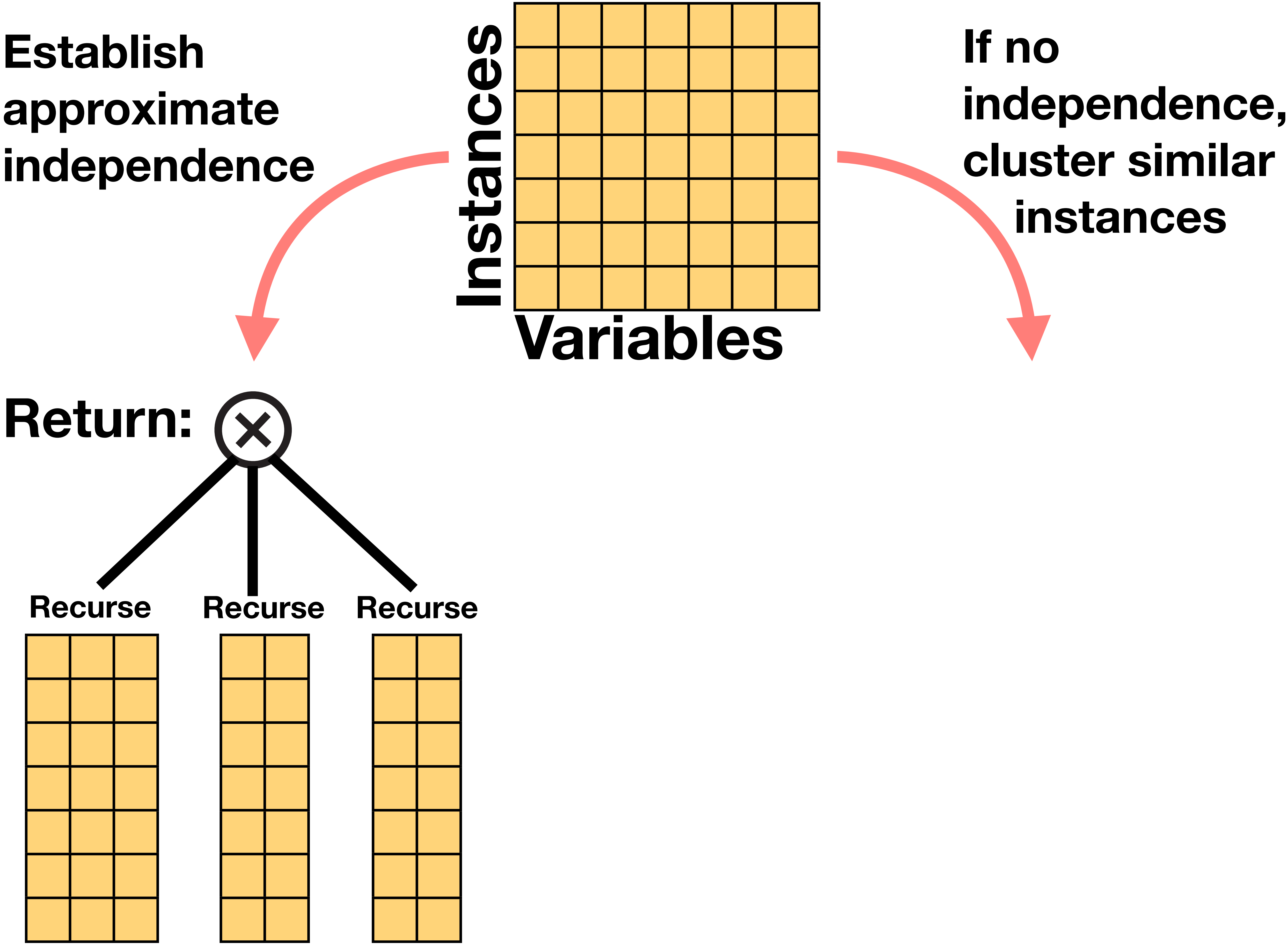


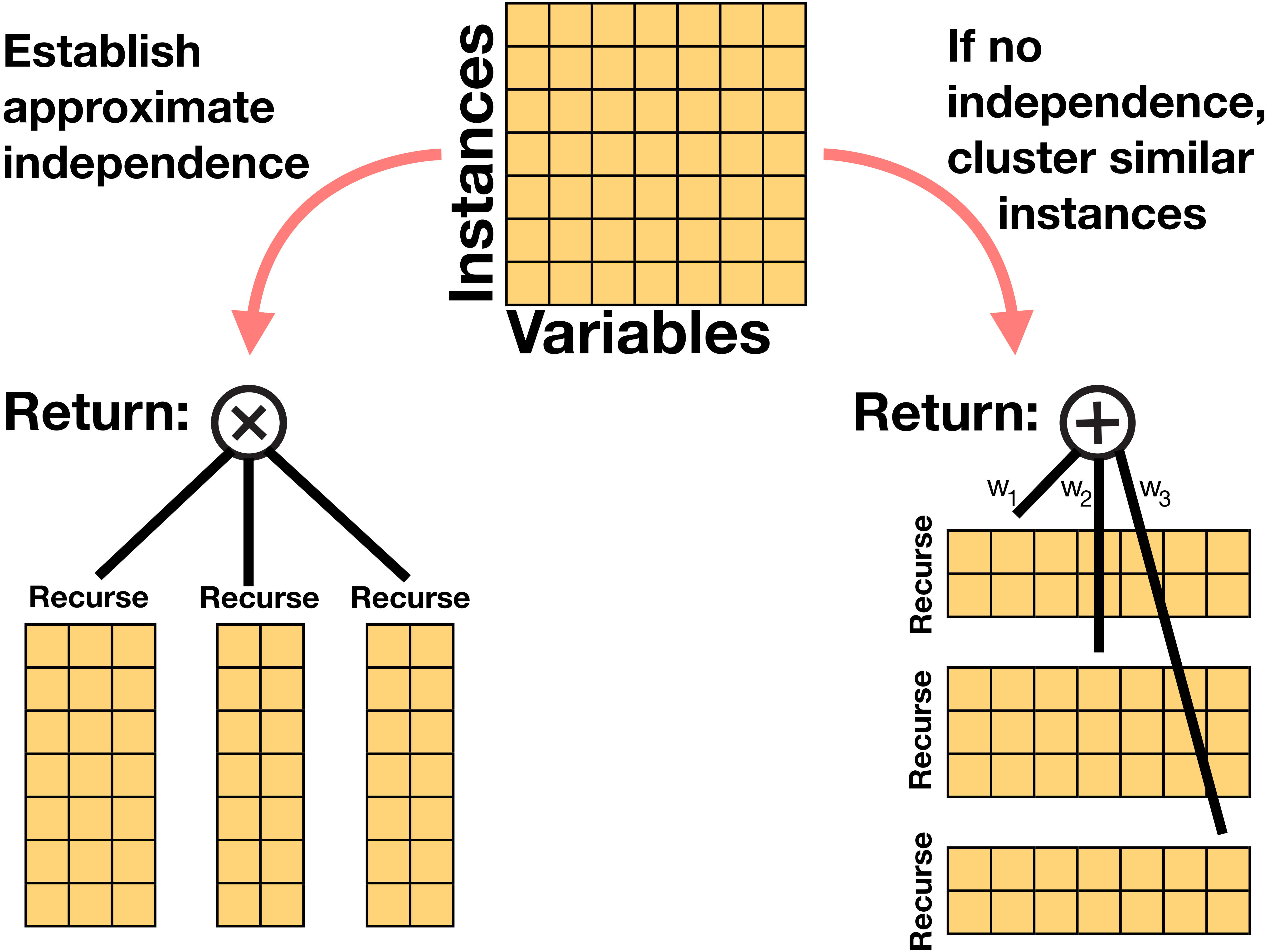

Instances
Variables

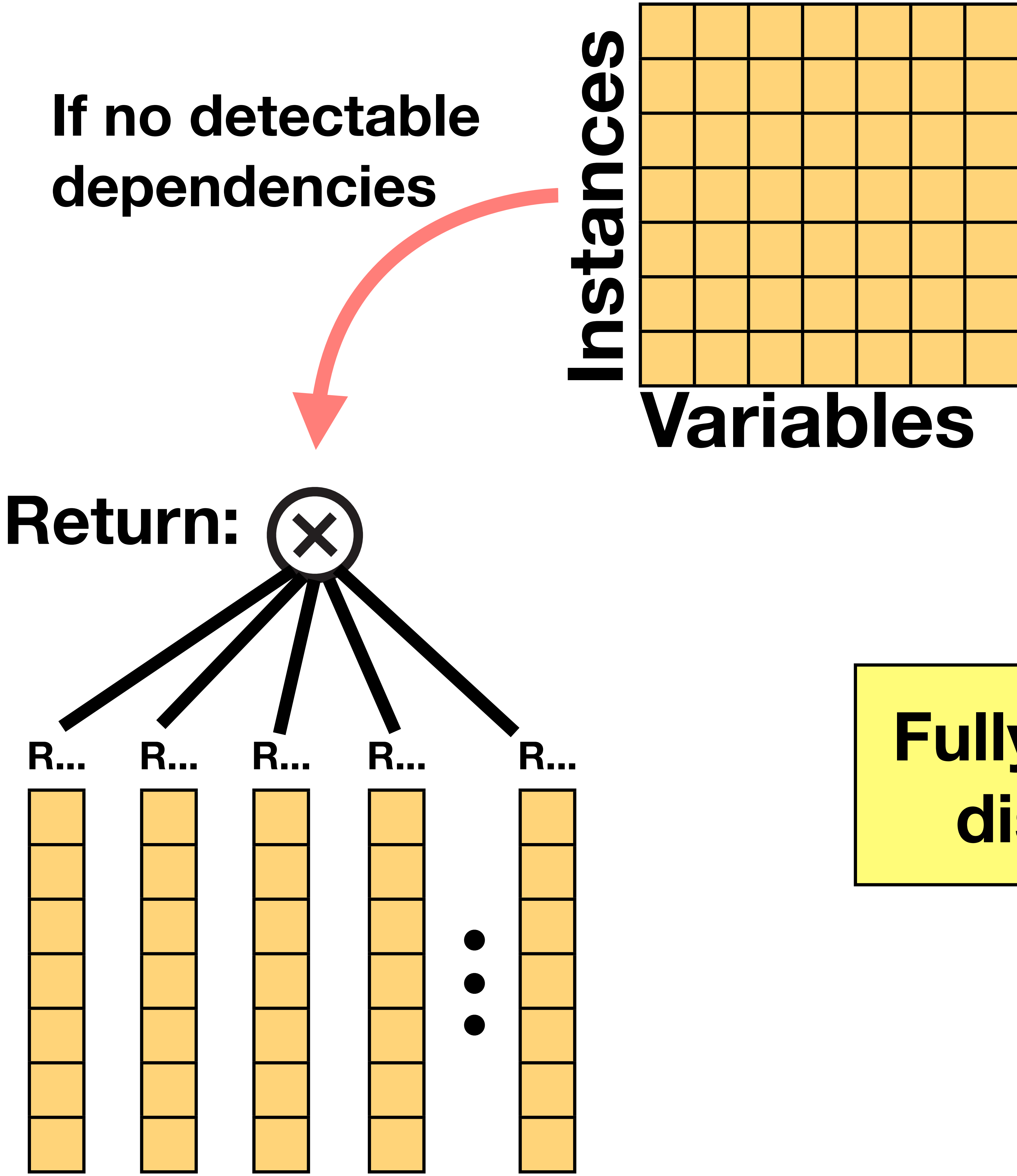
**Establish approximate independence**



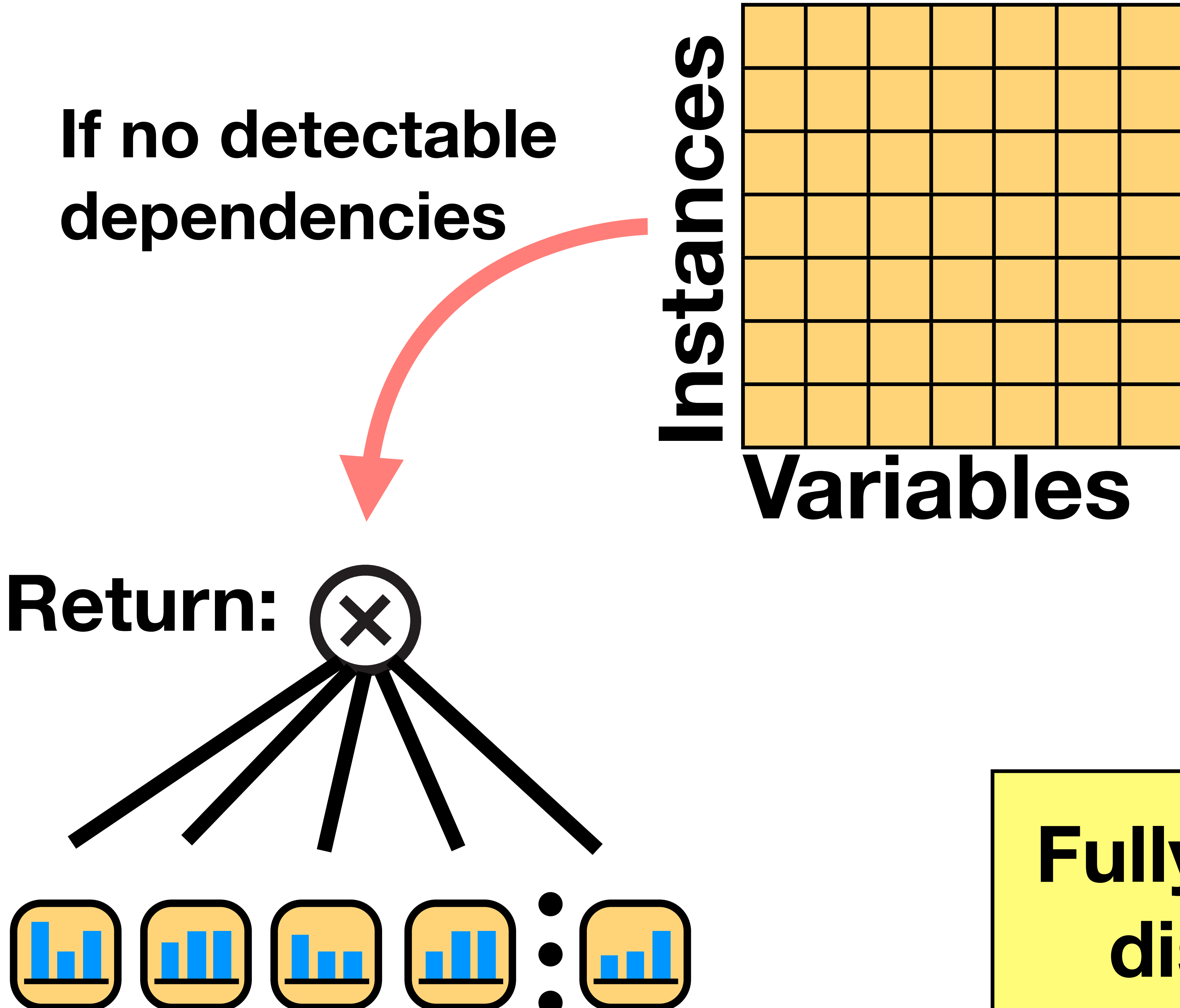

Establish approximate independence
Instances
Variables
Return:
Recurse
Recurse
Recurse


Establish approximate independence
Instances
Variables
If no independence, cluster similar instances
Return:
Recurse
Recurse
Recurse

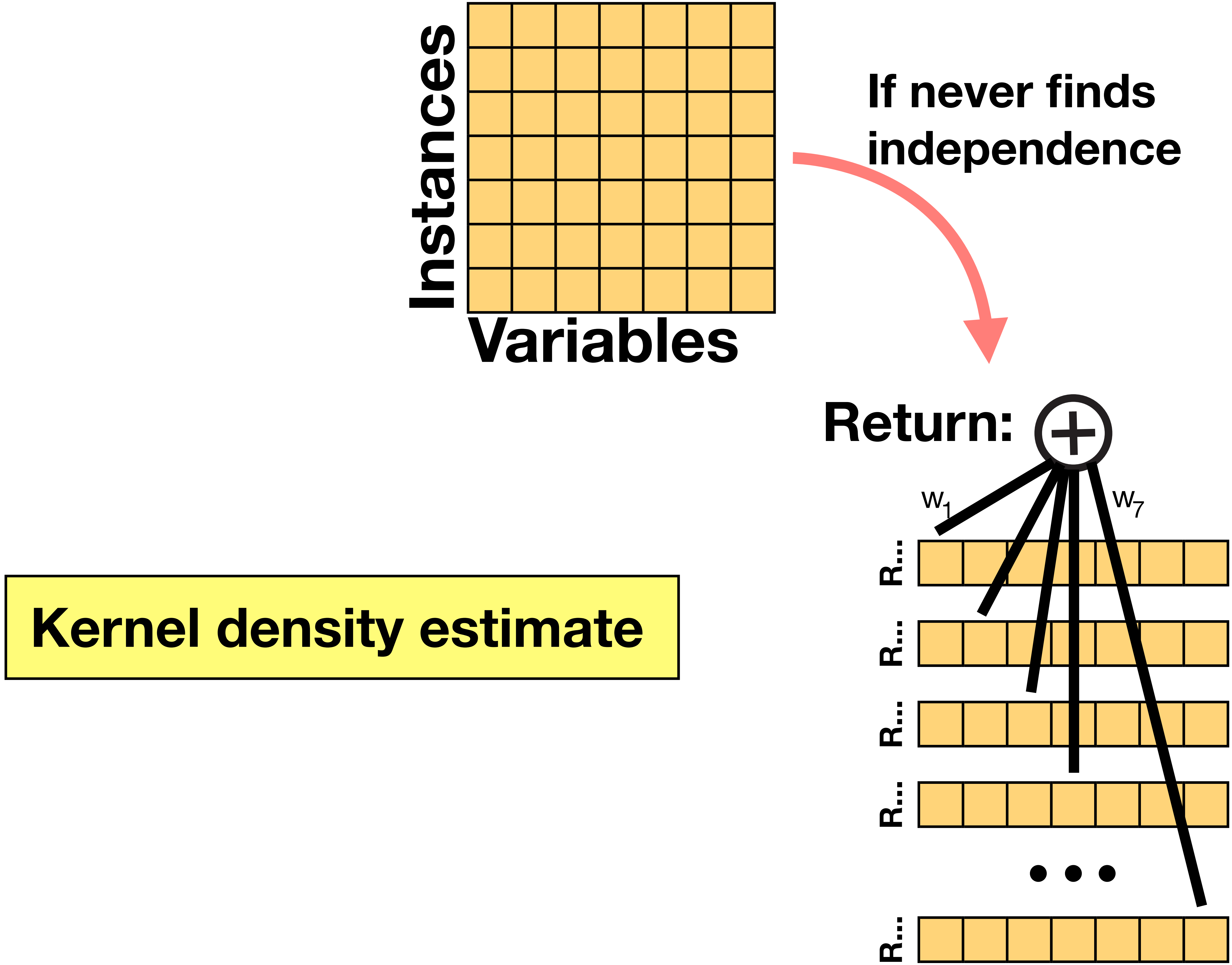

Establish approximate independence
Instances
Variables
If no independence, cluster similar instances
Return:
Recurse
Recurse
Recurse
Return:
$w_1$
$w_2$
$w_3$
Recurse
Recurse
Recurse

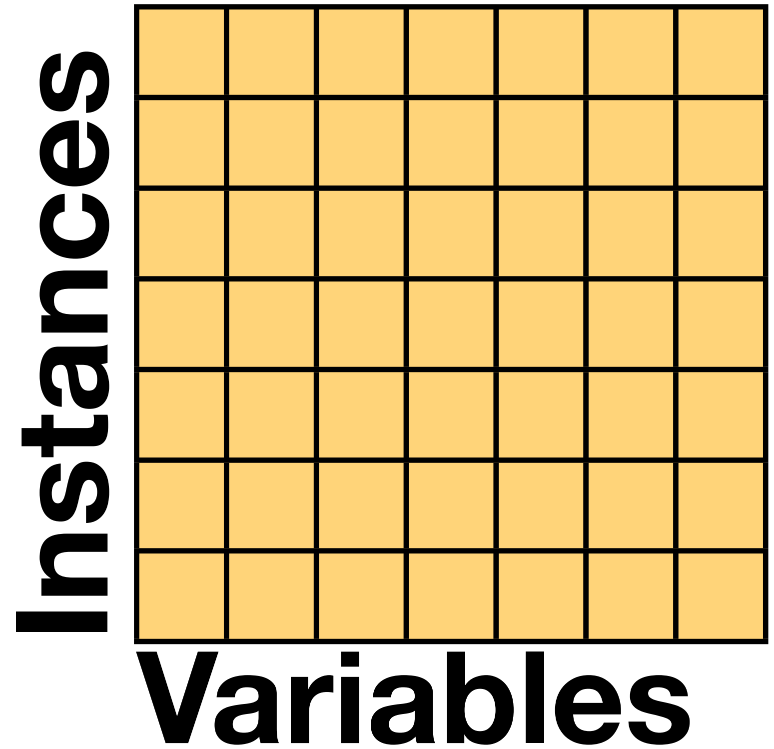



**Fully factorized distribution**



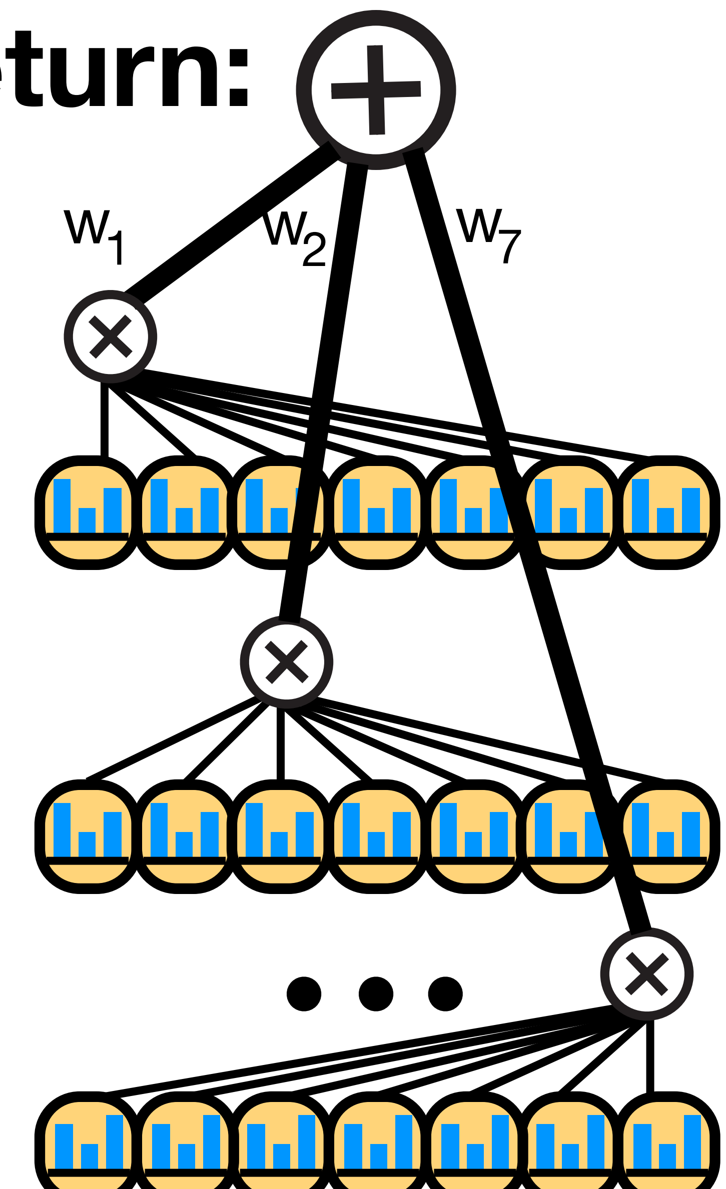

Fully factorized distribution


Instances
Variables
If never finds independence
Return:
+
w1
w7
R...
R...
R...
R...
R...
Kernel density estimate

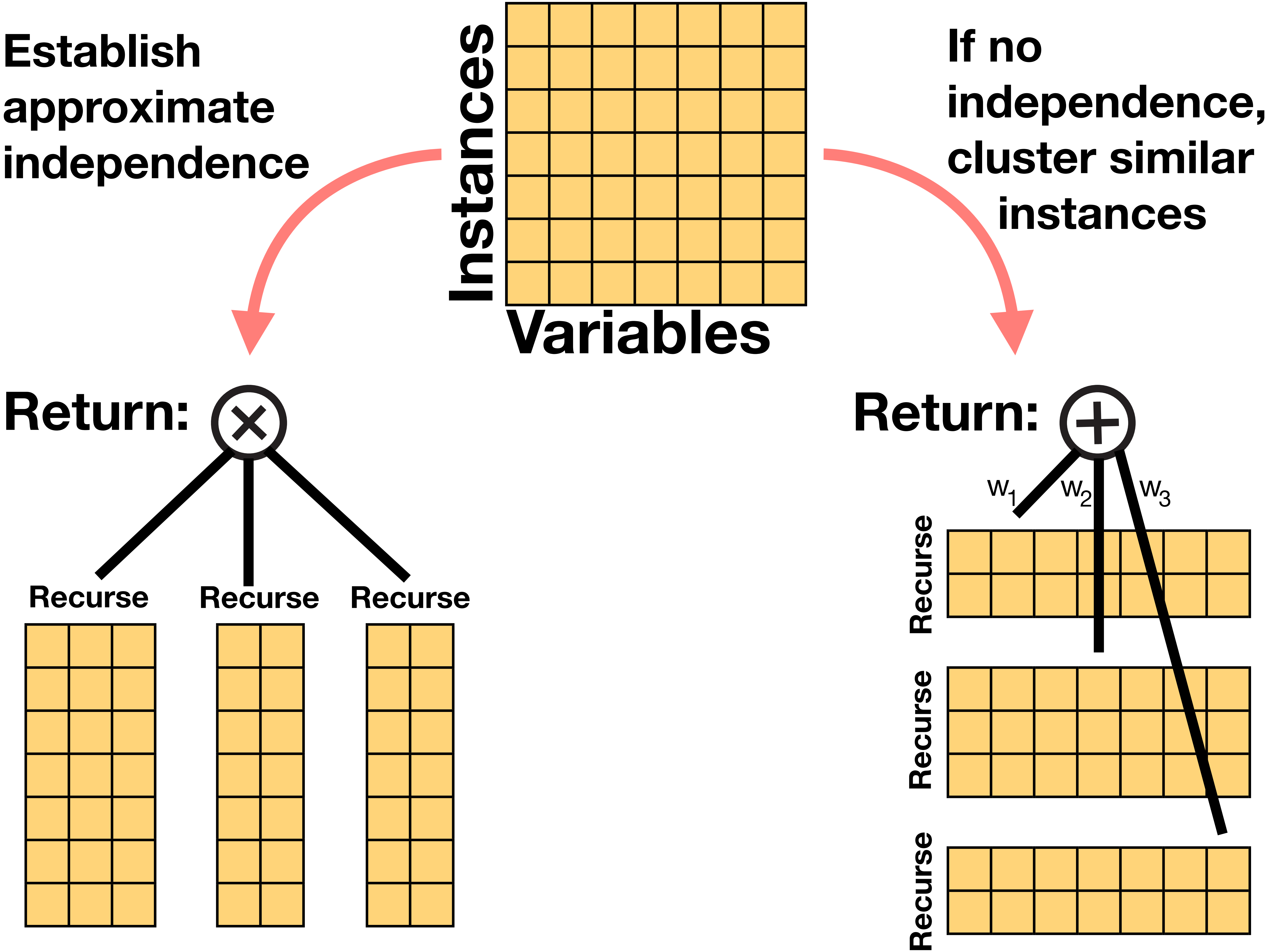


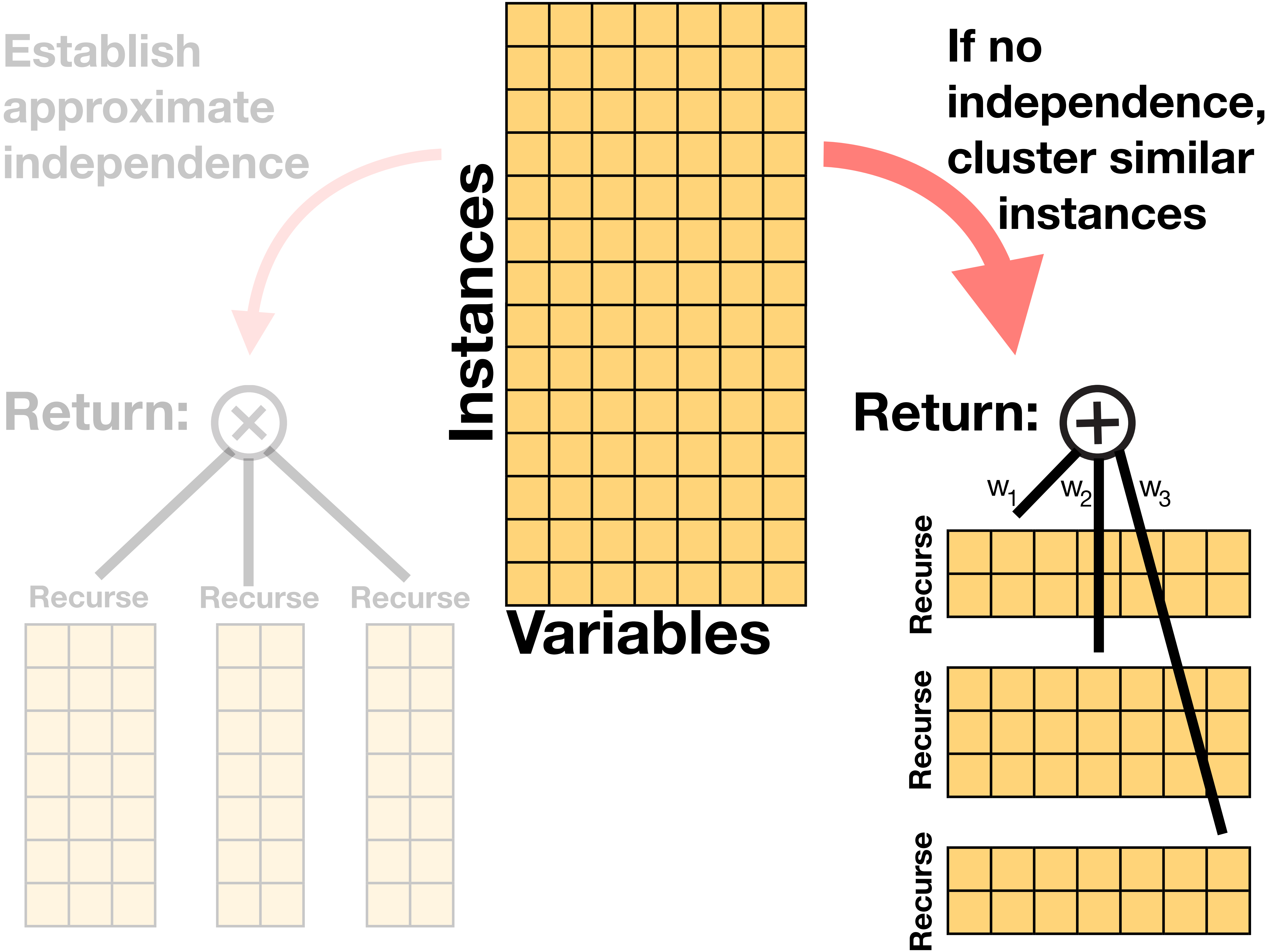

**If never finds independence**

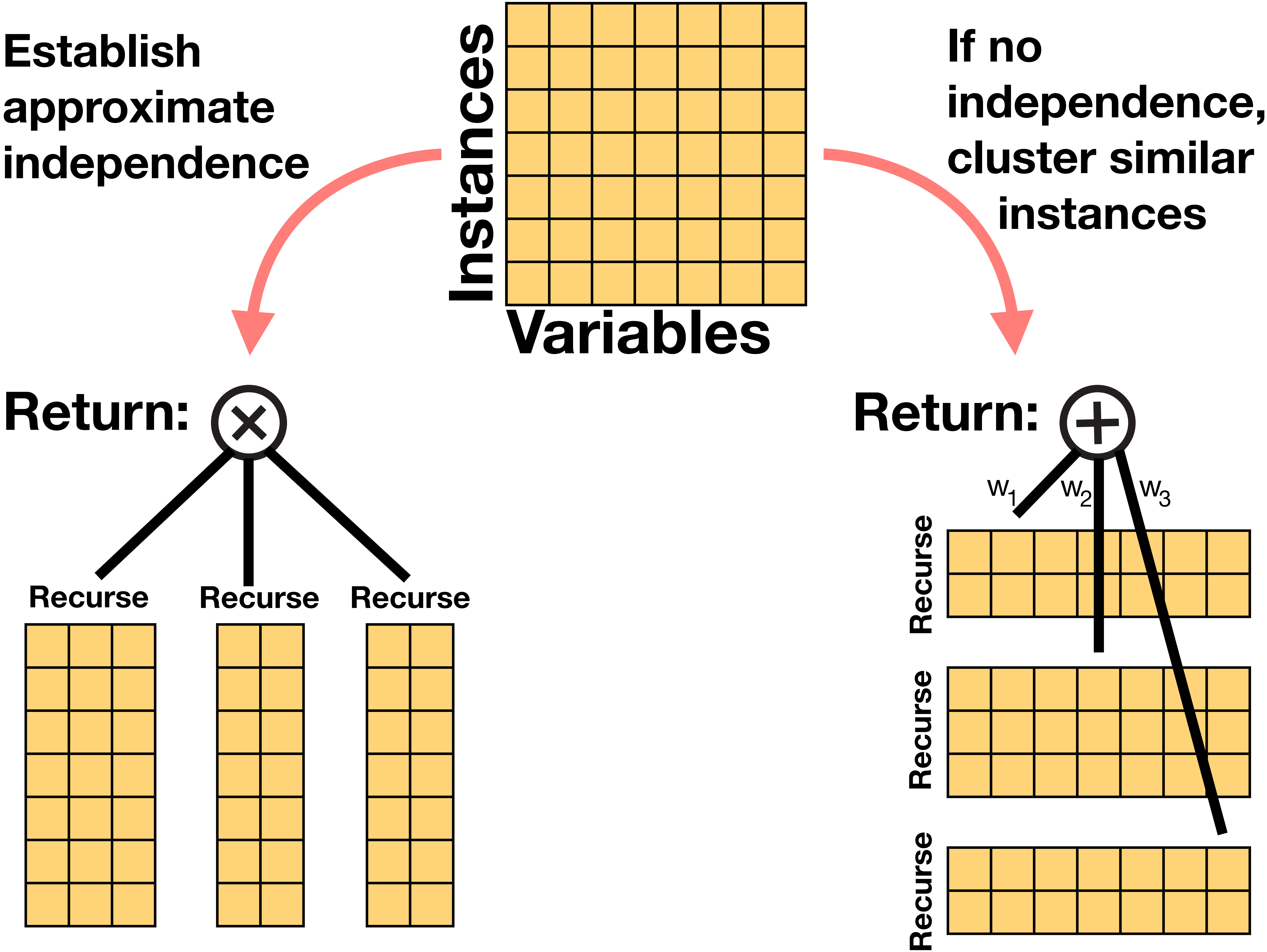
**Return:**

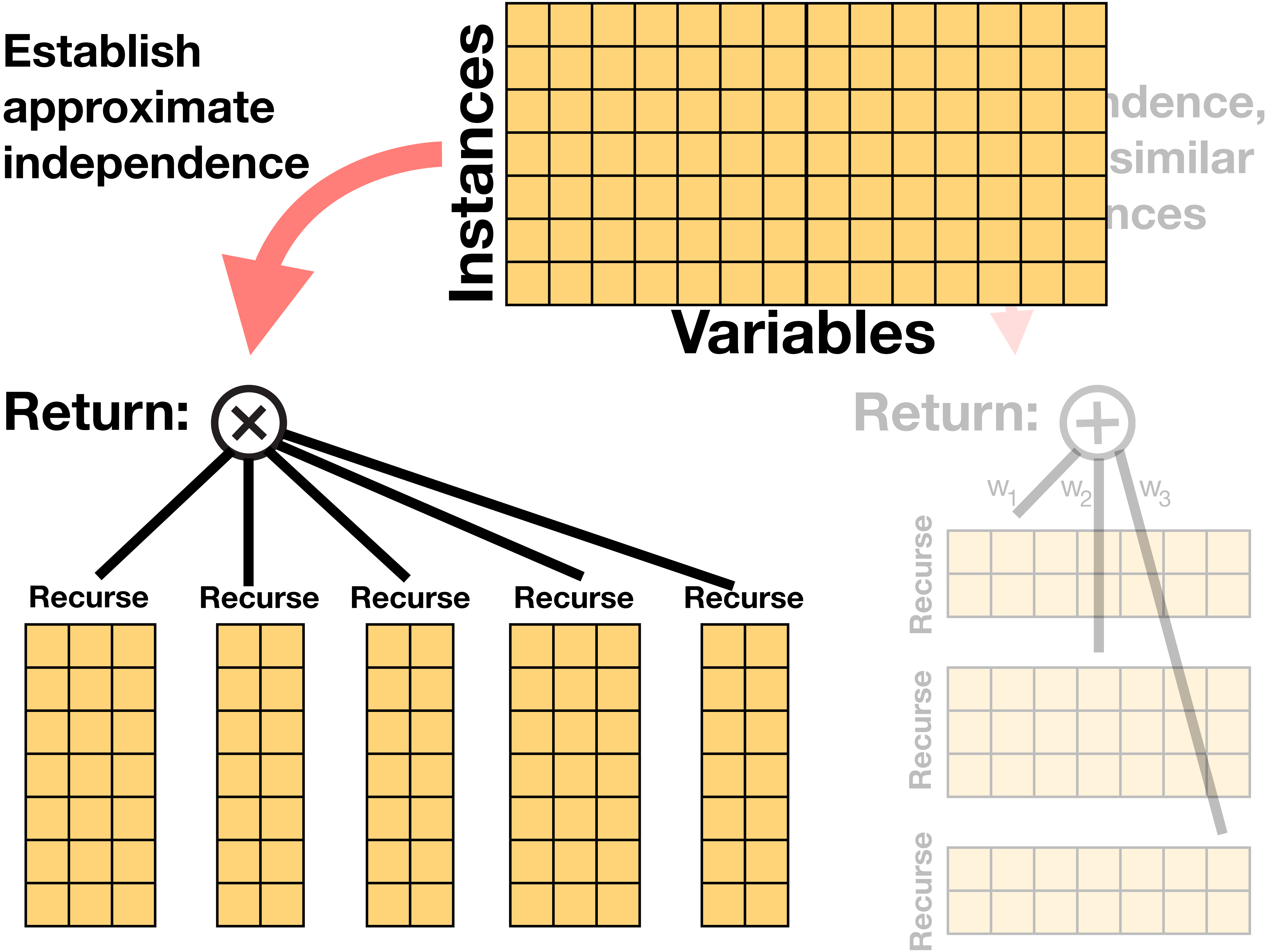



**Kernel density estimate**

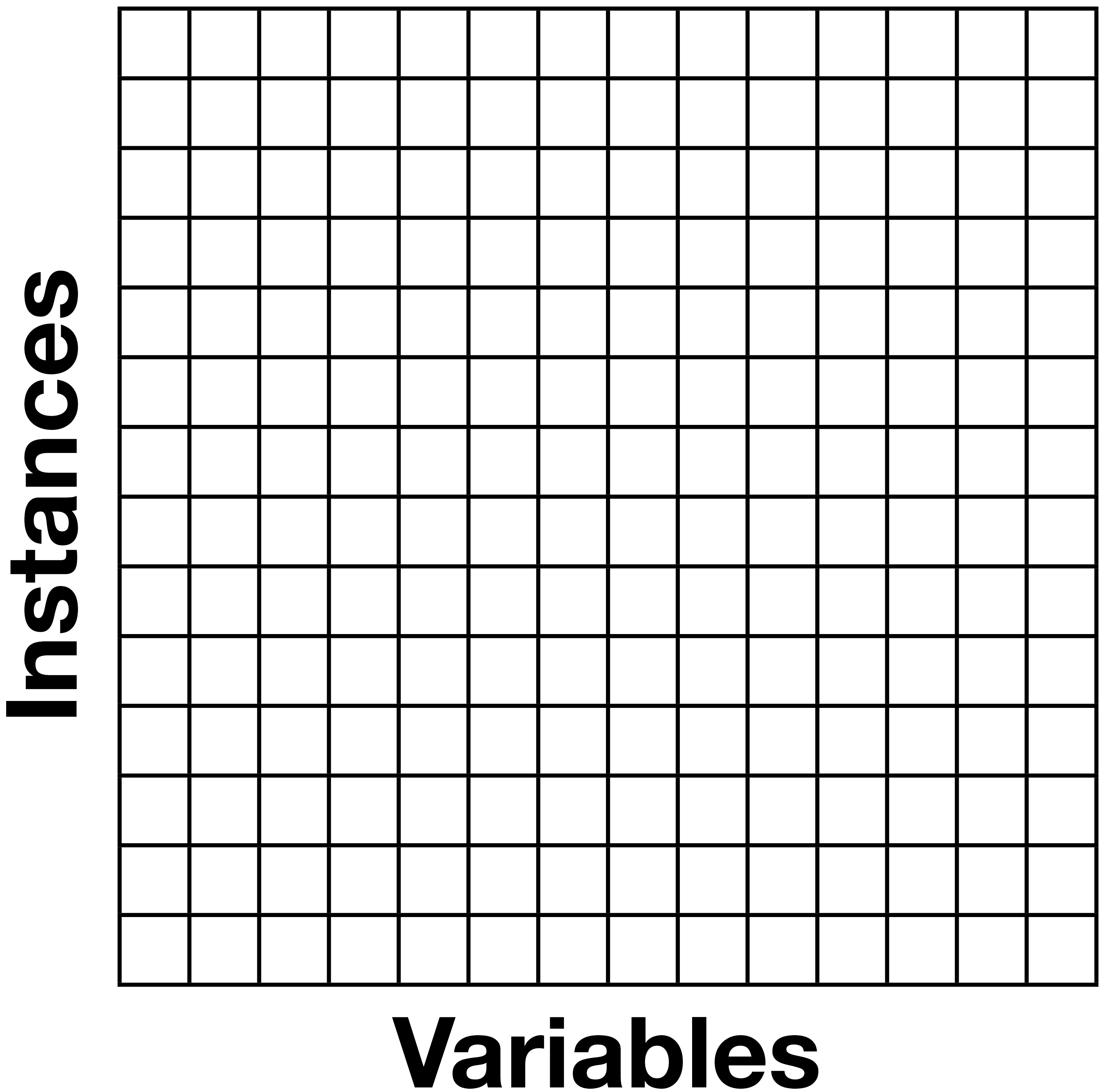

Establish approximate independence
Instances
Variables
If no independence, cluster similar instances
Return:
Recurse
Recurse
Recurse
Return:
w1
w2
w3
Recurse
Recurse
Recurse

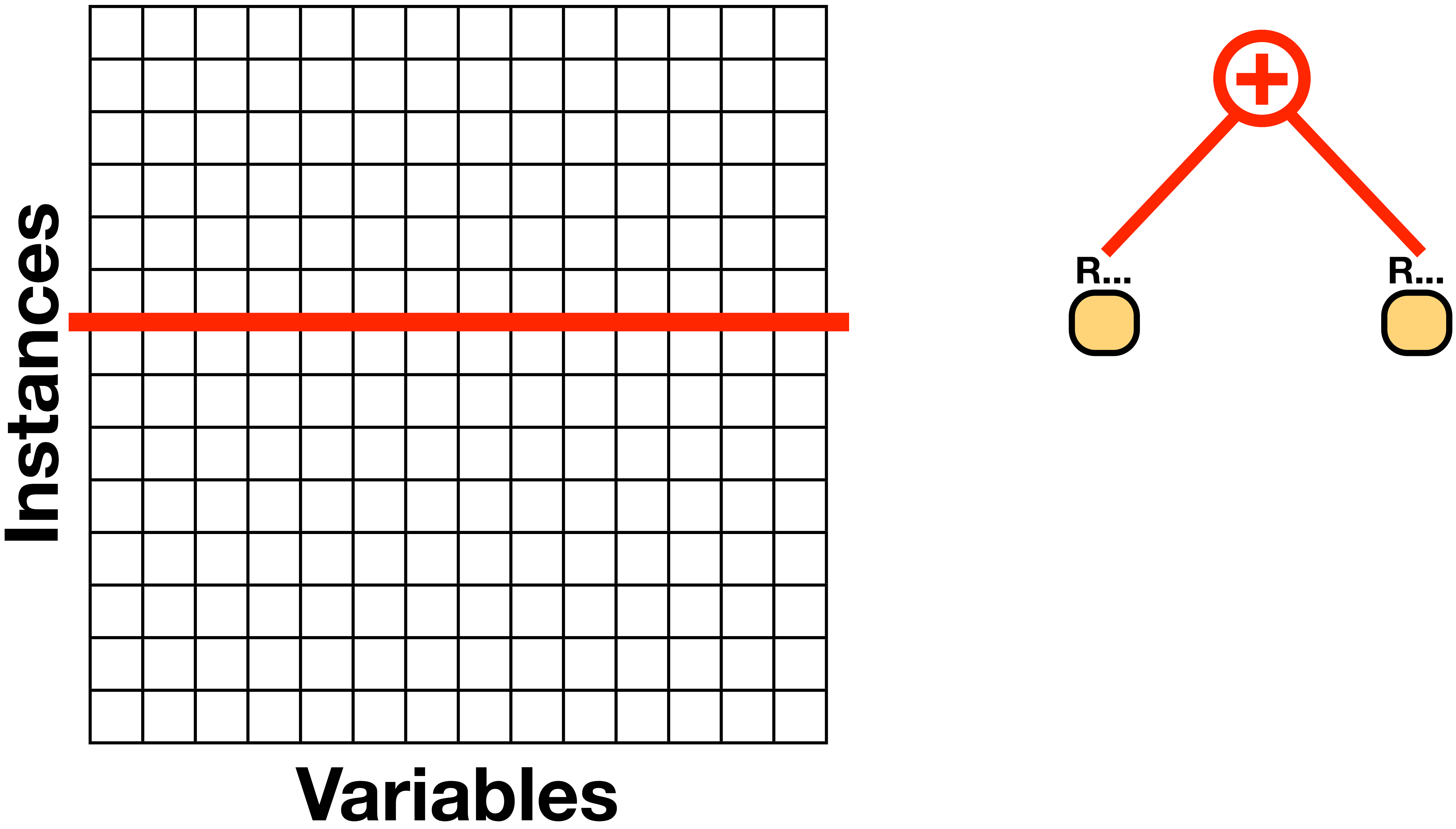

Establish approximate independence
Instances
Variables
If no independence, cluster similar instances
Return:
Return:
Recurse
Recurse
Recurse
w1
w2
w3
Recurse
Recurse
Recurse

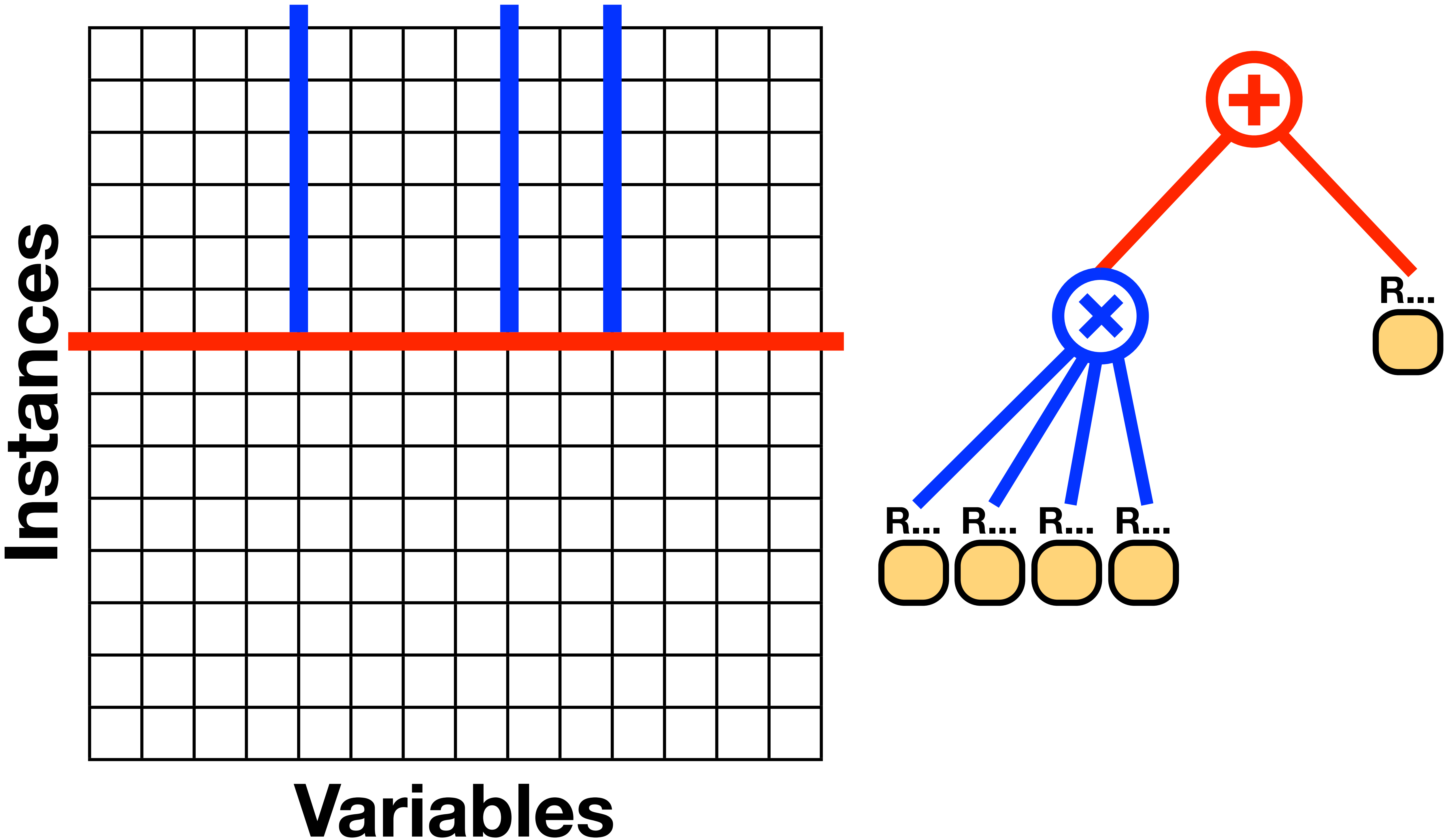

Establish approximate independence
Instances
Variables
If no independence, cluster similar instances
Return:
Recurse
Recurse
Recurse
Return:
w1
w2
w3
Recurse
Recurse
Recurse



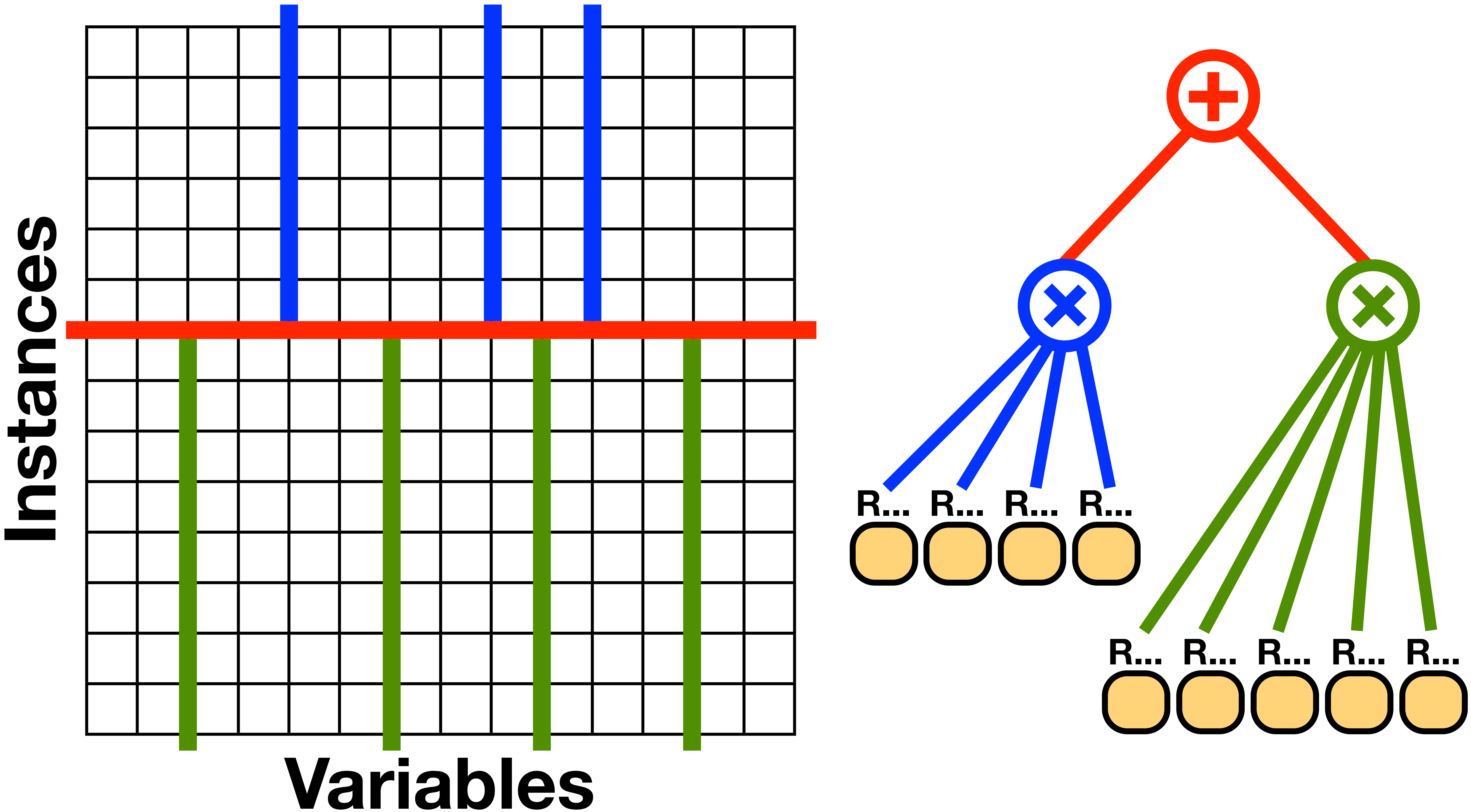
Establish approximate independence
Instances
Variables
ndence,
similar
nces
Return:
Recurse
Recurse
Recurse
Recurse
Recurse
Return:
$w_1$
$w_2$
$w_3$
Recurse
Recurse
Recurse


Instances
Variables

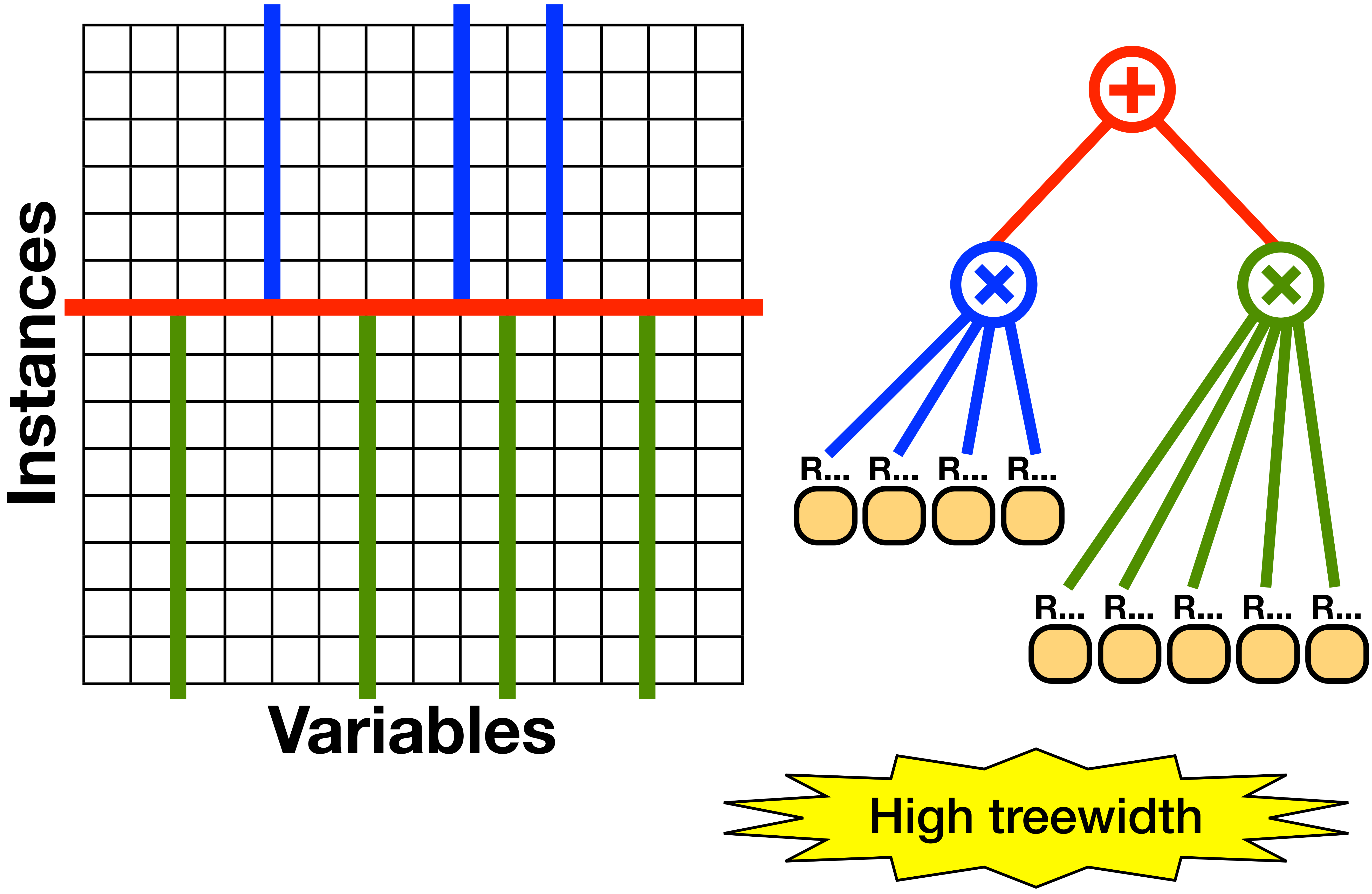

Instances
Variables
R...
R...

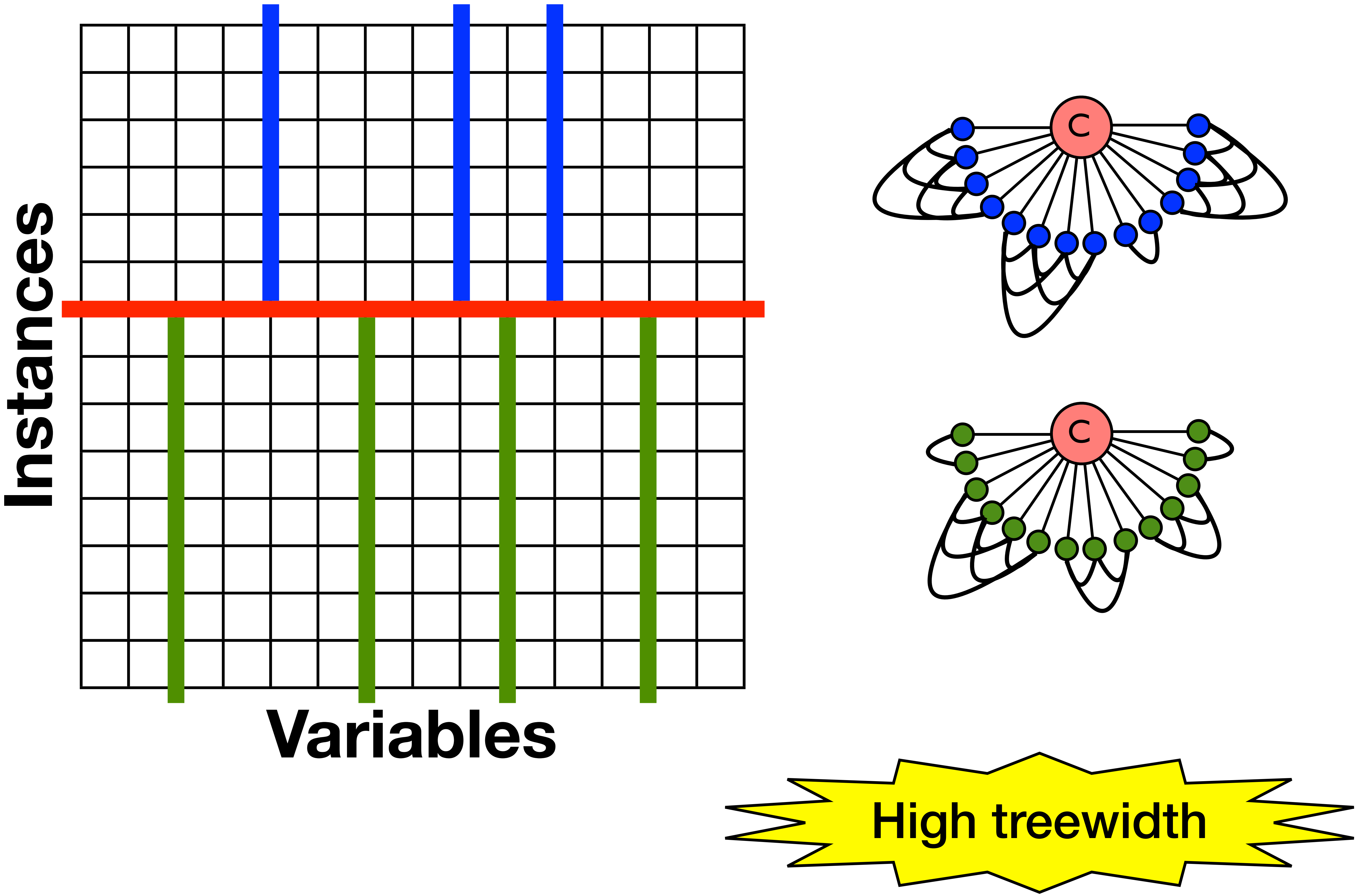


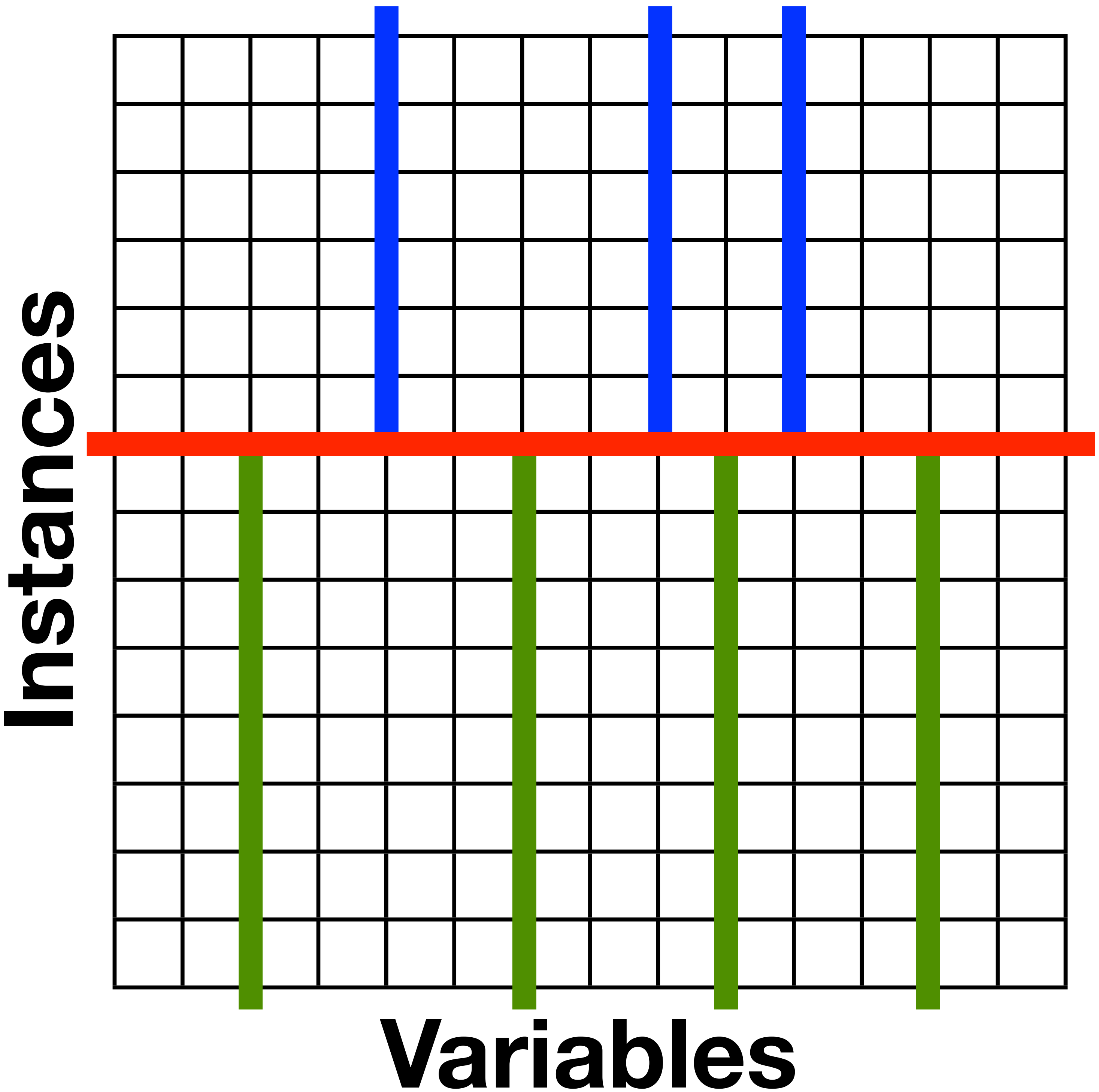
Instances
Variables
R...
R...
R...
R...
R...

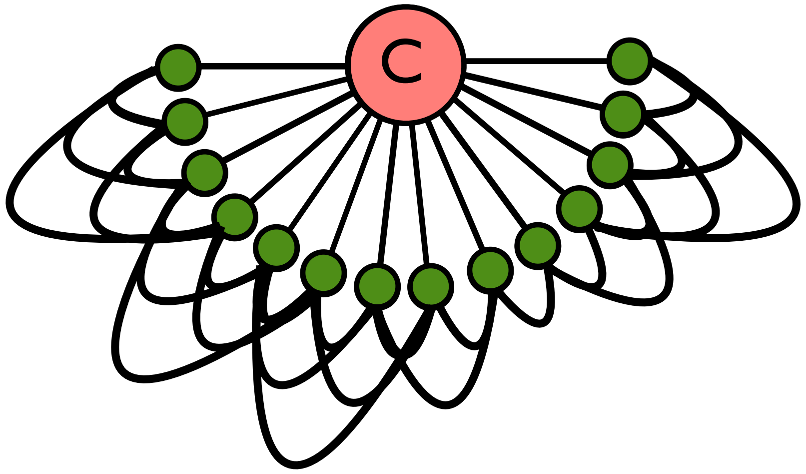


Instances
Variables
R...
R...
R...
R...
R...
R...
R...
R...
R...

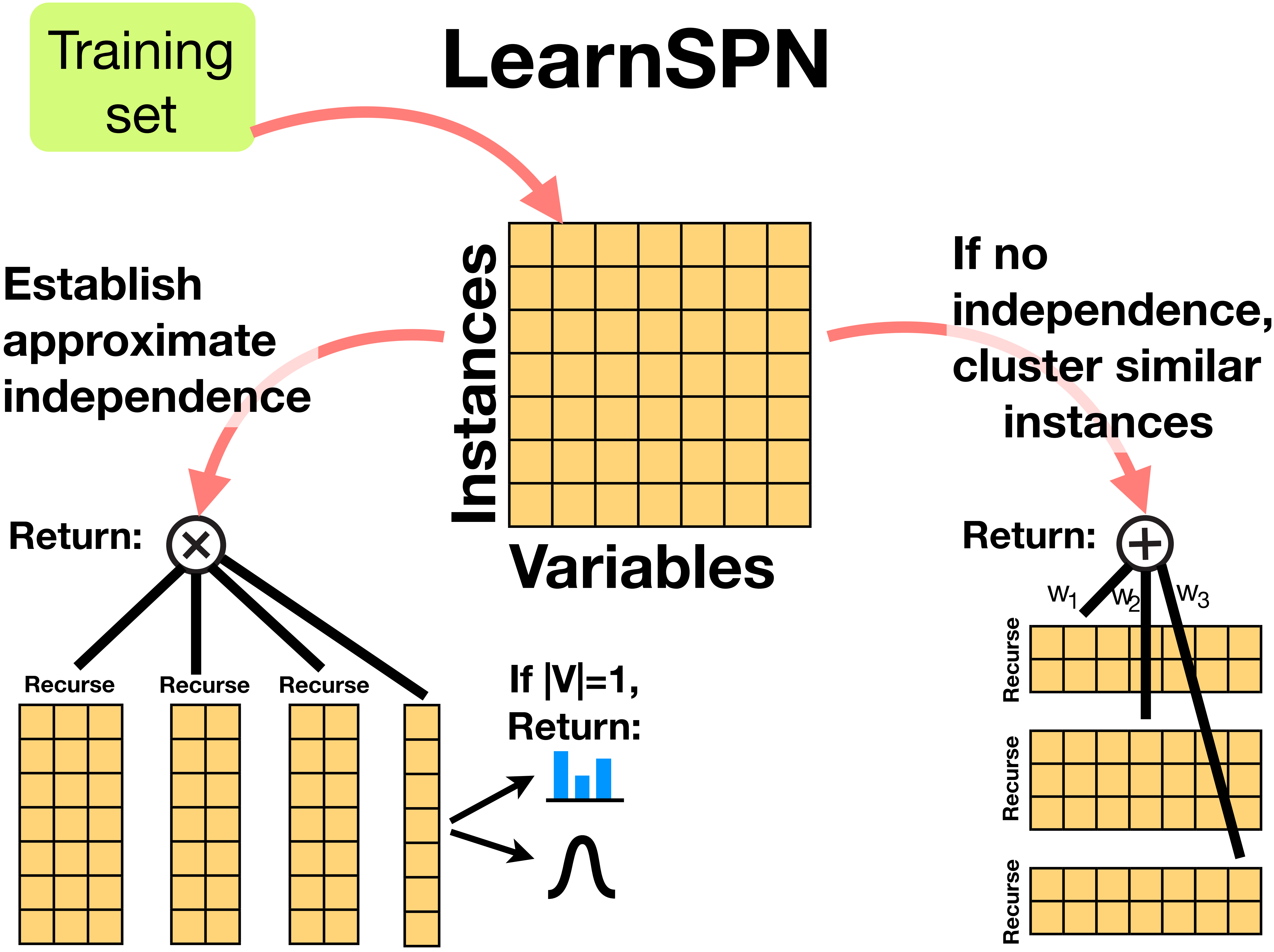

Instances
Variables
R...
R...
R...
R...
R...
R...
R...
R...
R...
High treewidth


Instances
Variables
C
C
High treewidth



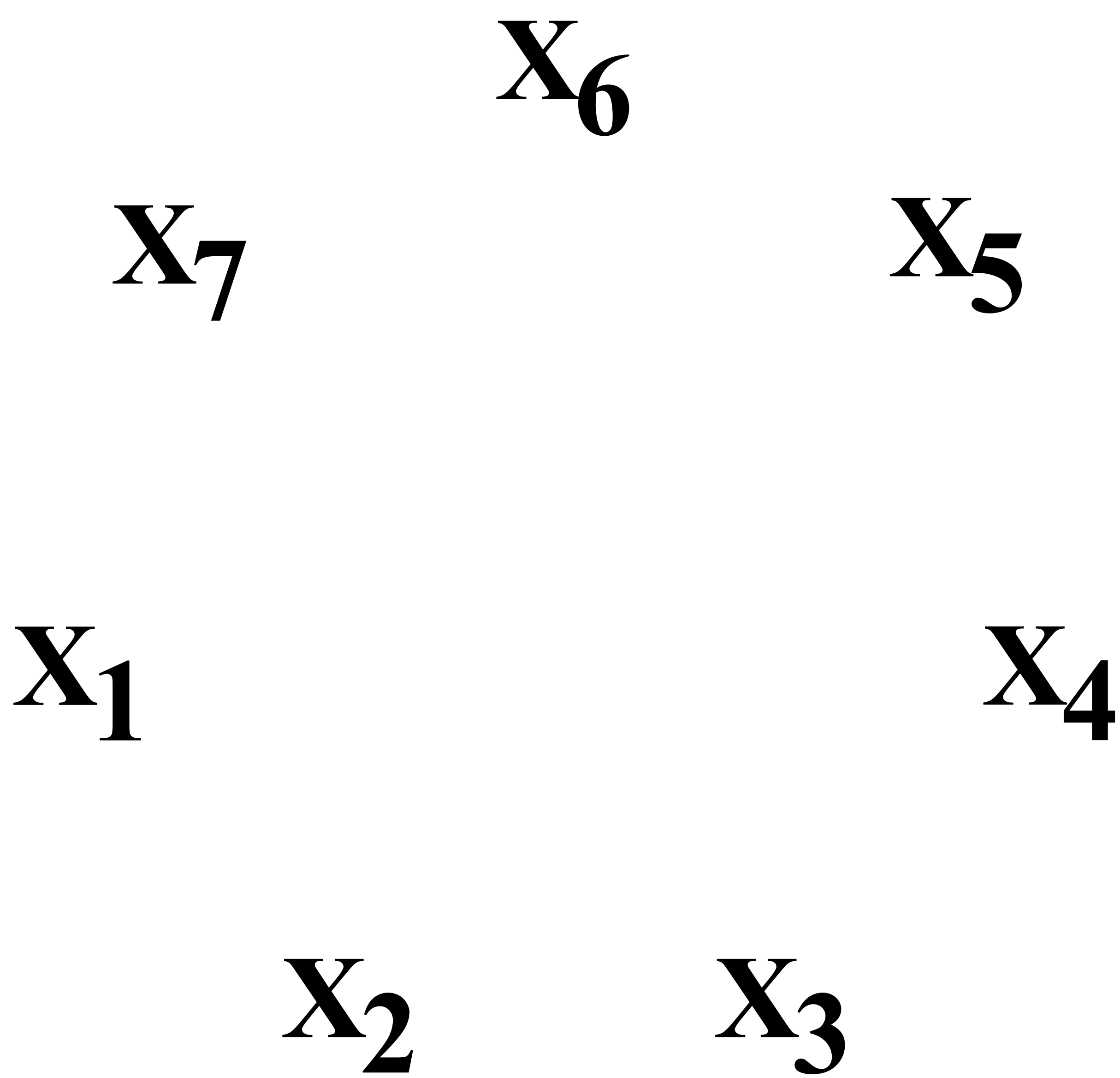




High treewidth


Training set
LearnSPN
Instances
Variables
Establish approximate independence
If no independence, cluster similar instances
Return:
Return:
w1
w2
w3
Recurse
Recurse
Recurse
If |V|=1, Return:
Recurse
Recurse
Recurse

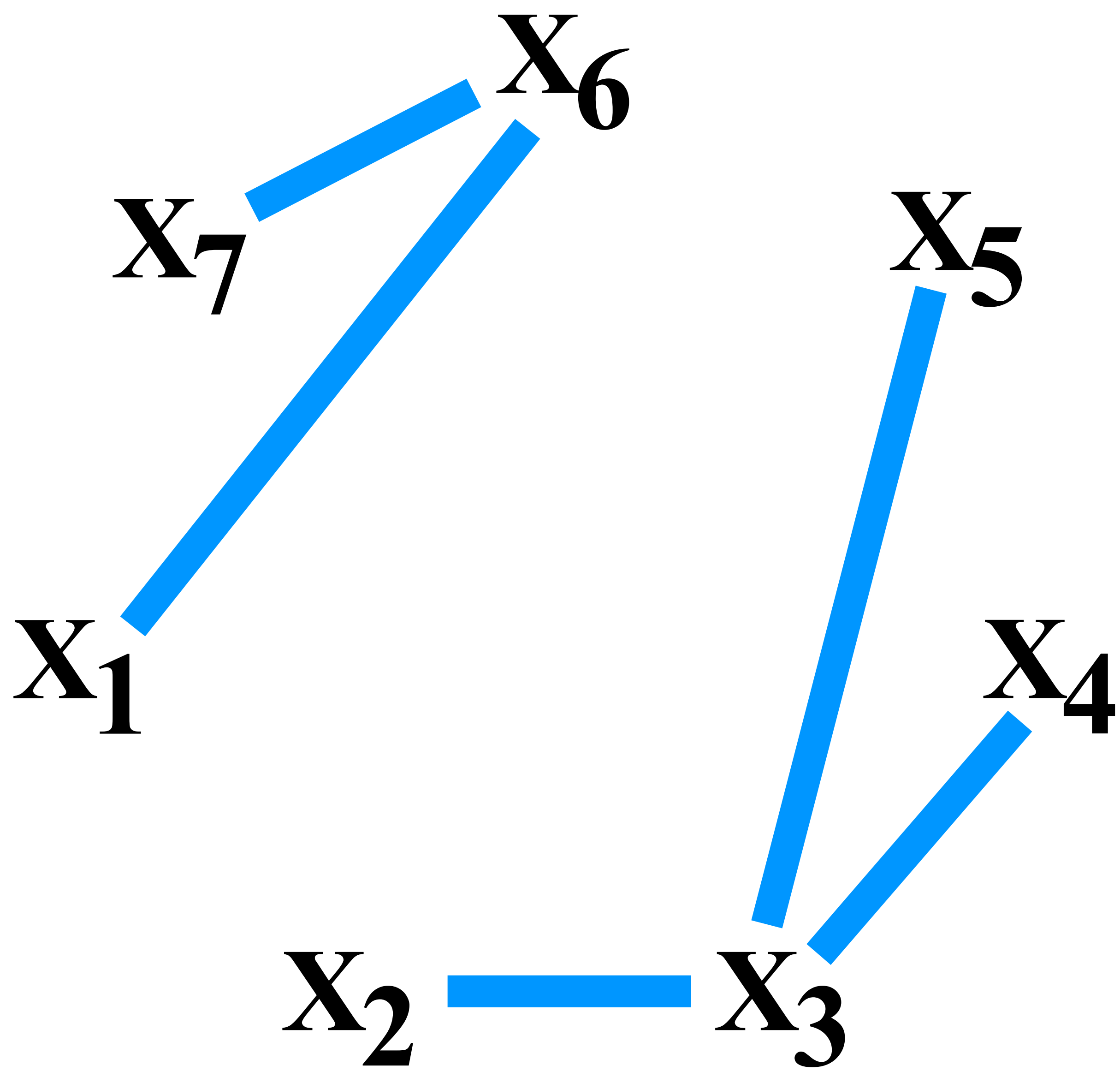
# Variable Splits (×)

- Pairwise independence test
- p-value (validation set)



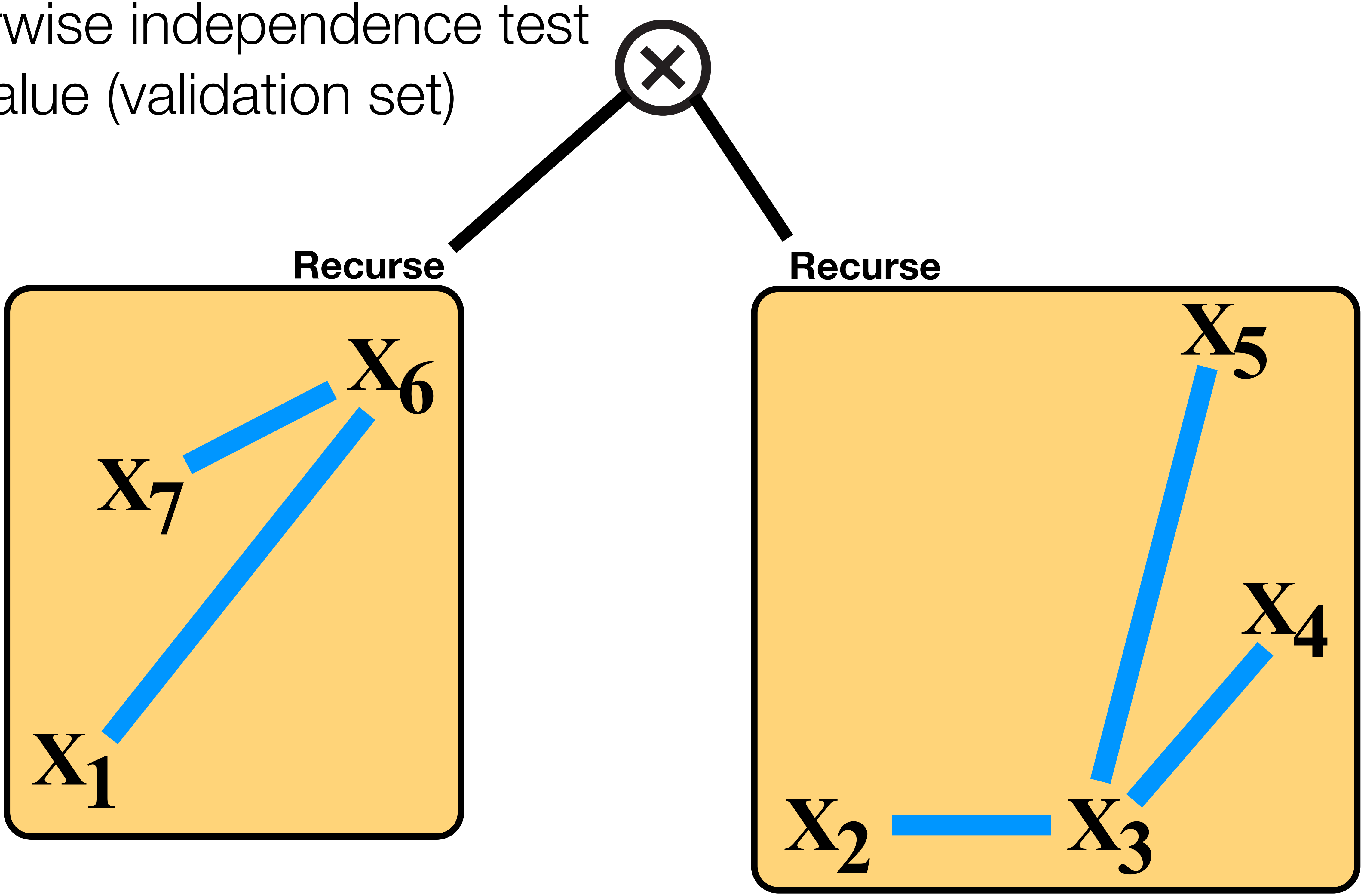
# Variable Splits (×)

- Pairwise independence test
- p-value (validation set)



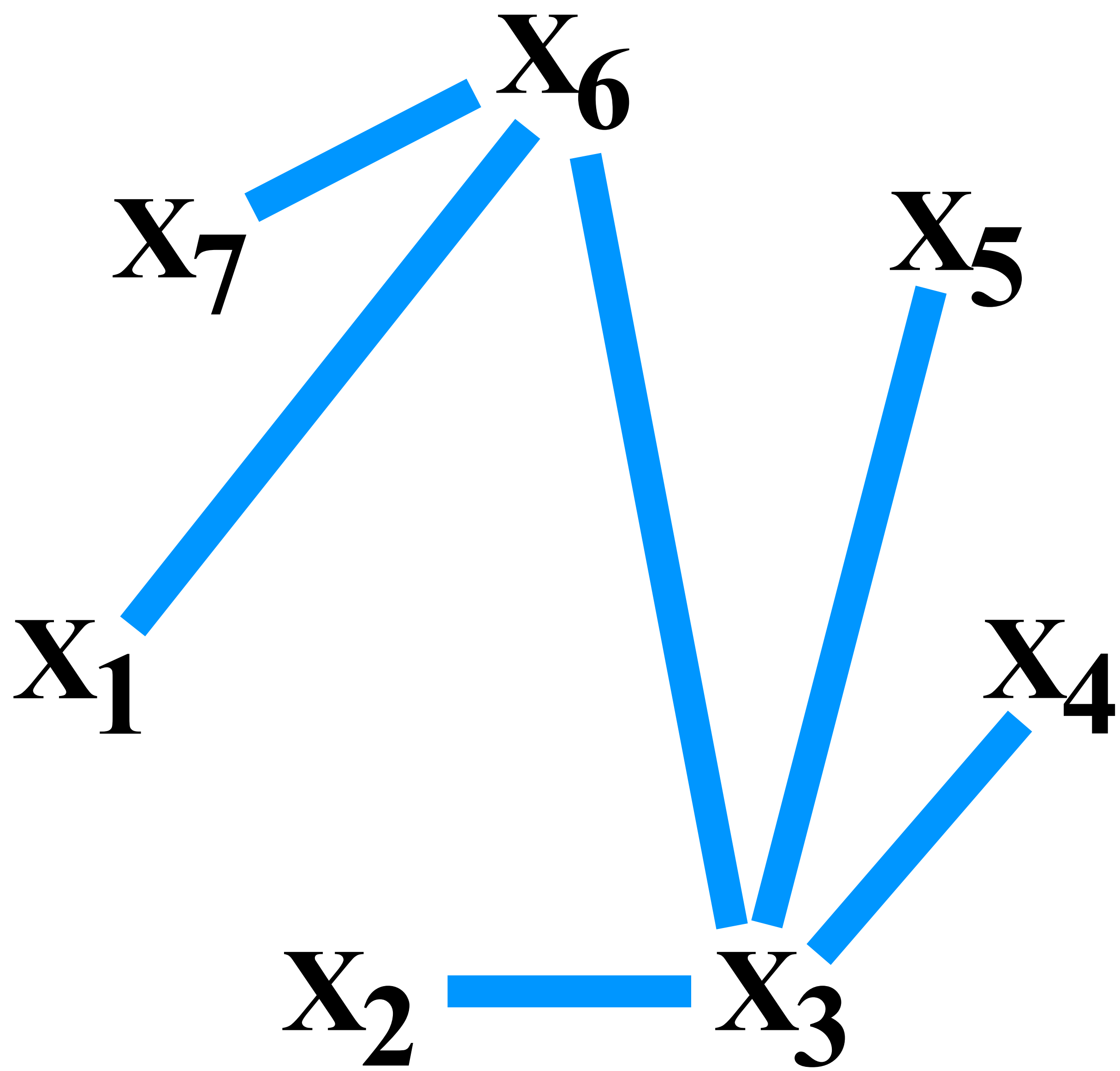
# Variable Splits (×)

- Pairwise independence test
- p-value (validation set)

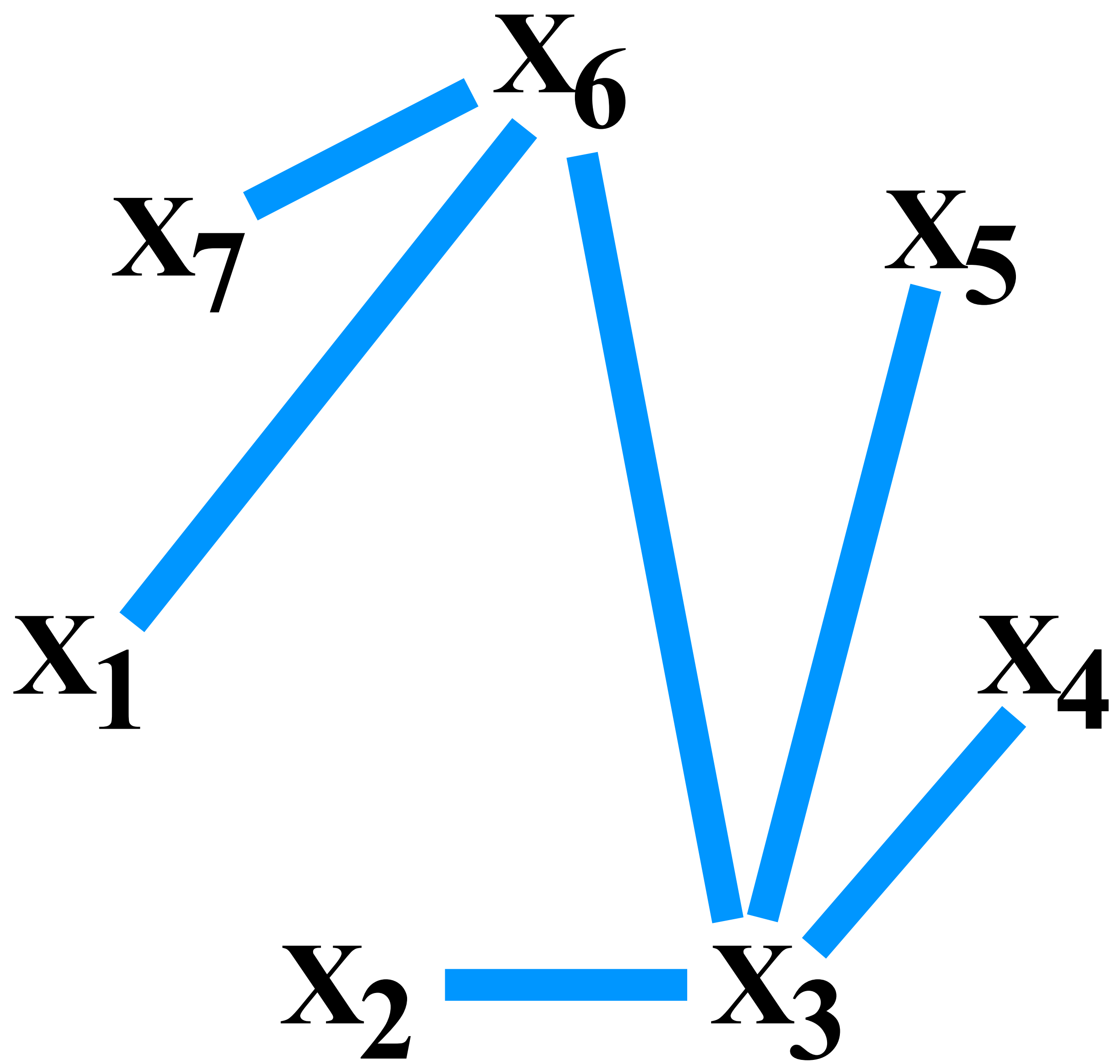


# Variable Splits (×)

- Pairwise independence test
- p-value (validation set)



# Instance Clustering (+)



# Instance Clustering (+)

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\mathbf{x}_5$ $\mathbf{x}_6$ $\mathbf{x}_7$

# Instance Clustering (+)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $m_1$ | | | | | | | |
| $m_2$ | | | | | | | |
| $m_3$ | | | | | | | |
| $m_4$ | | | | | | | |
| $m_5$ | | | | | | | |
| $m_6$ | | | | | | | |
| $m_7$ | | | | | | | |
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |

# Instance Clustering (+)

$\mathbf{m_1}$
$\mathbf{m_2}$
$\mathbf{m_3}$
$\mathbf{m_4}$
$\mathbf{m_5}$
$\mathbf{m_6}$
$\mathbf{m_7}$

# Instance Clustering (+)

- Online hard EM
- Naive Bayes mixture model
- Cluster penalty (validation set)
- Learned weights are the mixture priors

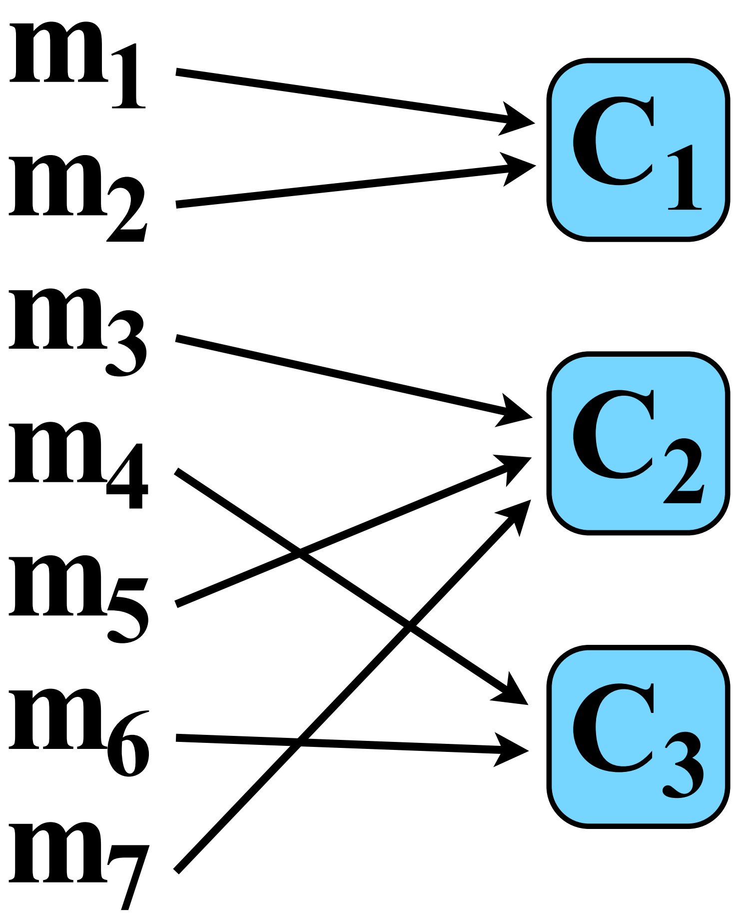
$$P(V) = \sum_i P(C_i) \prod_j P(X_j|C_i)$$

$\mathbf{m_1}$
$\mathbf{m_2}$
$\mathbf{m_3}$
$\mathbf{m_4}$
$\mathbf{m_5}$
$\mathbf{m_6}$
$\mathbf{m_7}$

# Instance Clustering (+)

- Online hard EM
- Naive Bayes mixture model
- Cluster penalty (validation set)
- Learned weights are the mixture priors

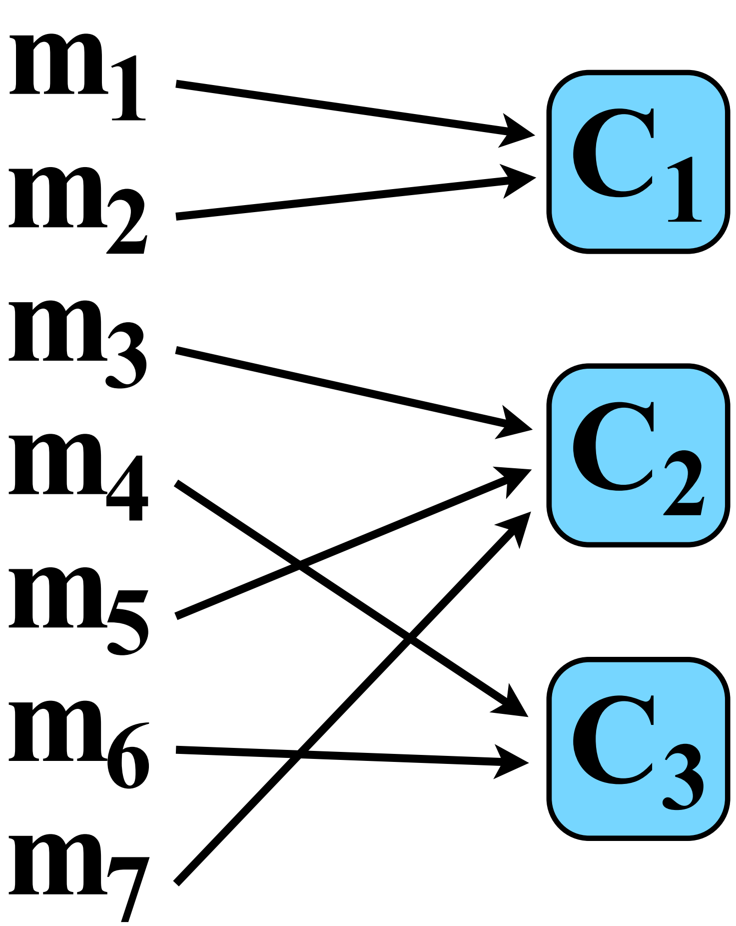
$$P(V) = \sum_i P(C_i) \prod_j P(X_j|C_i)$$



# Instance Clustering (+)

- Online hard EM
- Naive Bayes mixture model
- Cluster penalty (validation set)
- Learned weights are the mixture priors

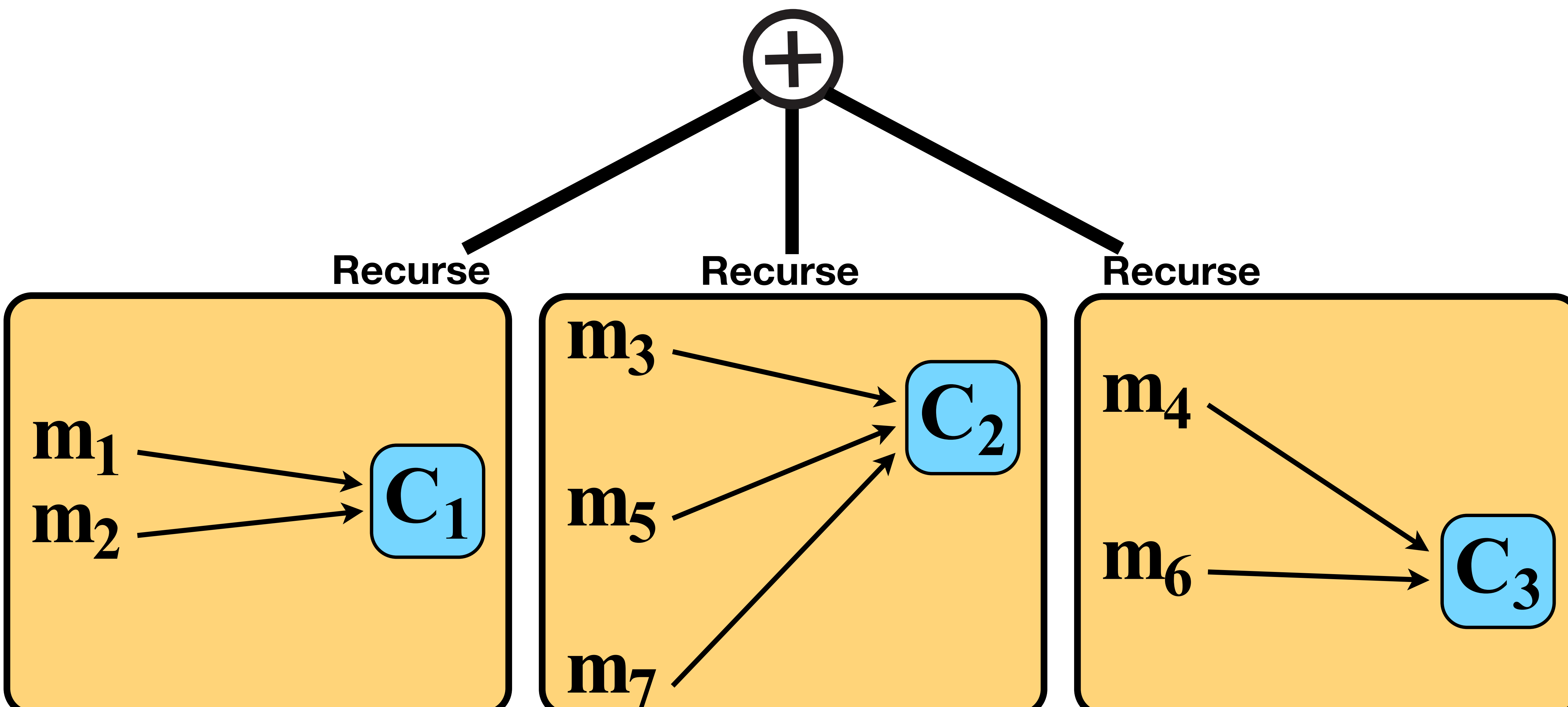
$$P(V) = \sum_i P(C_i) \prod_j P(X_j|C_i)$$



# Instance Clustering (+)

- Online hard EM
- Naive Bayes mixture model
- Cluster penalty (validation set)
- Learned weights are the mixture priors

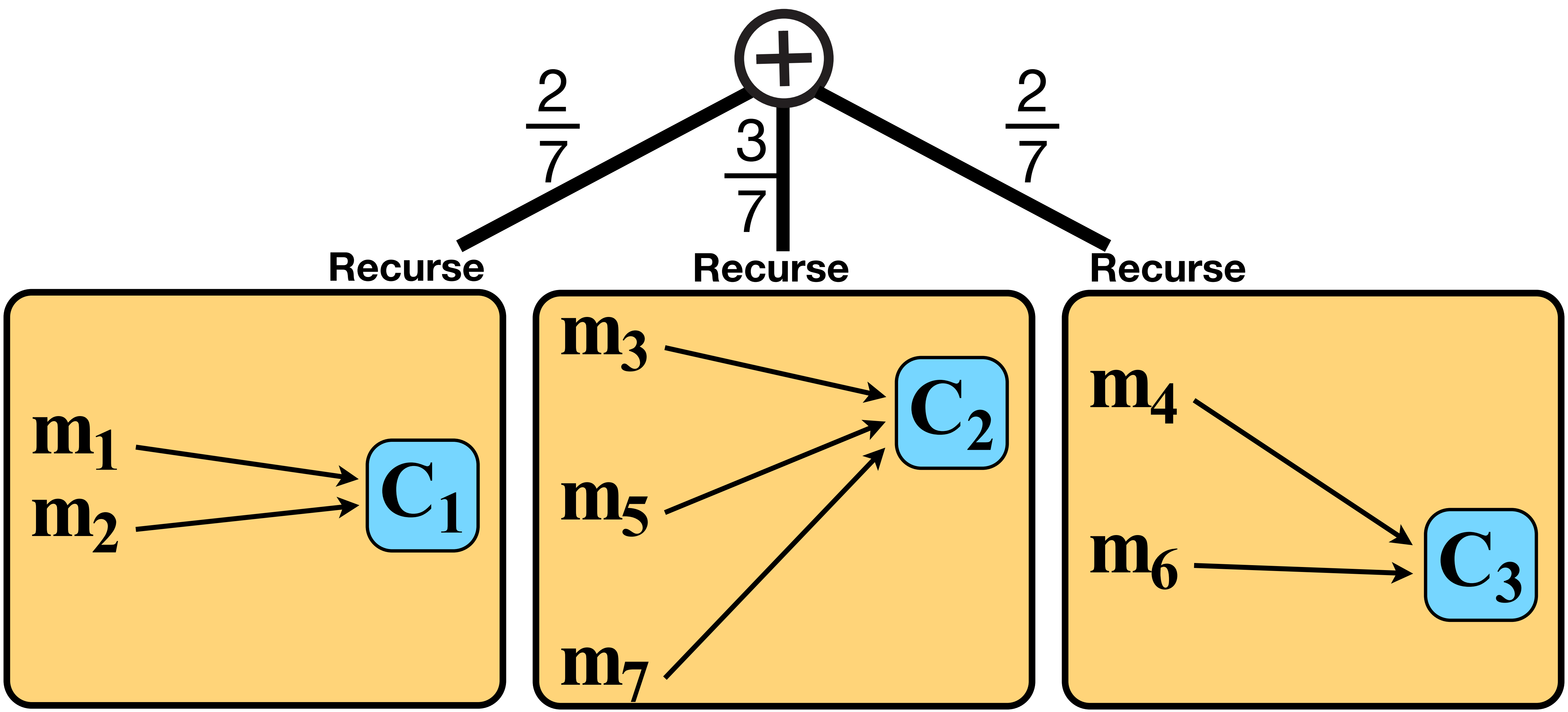
$$P(V) = \sum_i P(C_i) \prod_j P(X_j | C_i)$$



# Instance Clustering (+)

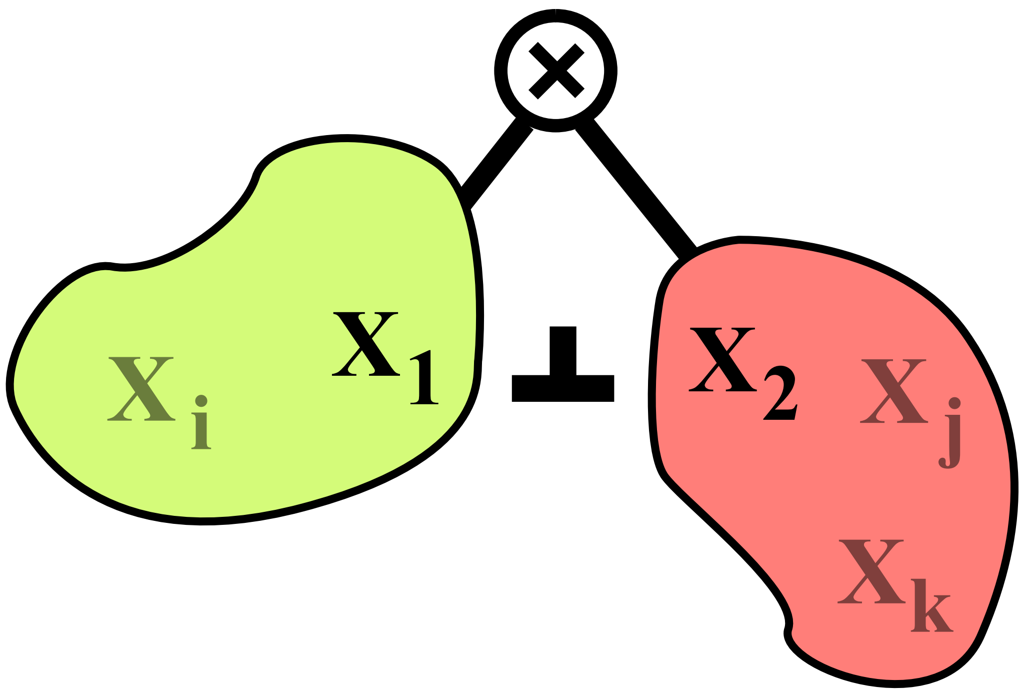
- Online hard EM
- Naive Bayes mixture model
- Cluster penalty (validation set)
- Learned weights are the mixture priors

$$P(V) = \sum_i P(C_i) \prod_j P(X_j|C_i)$$



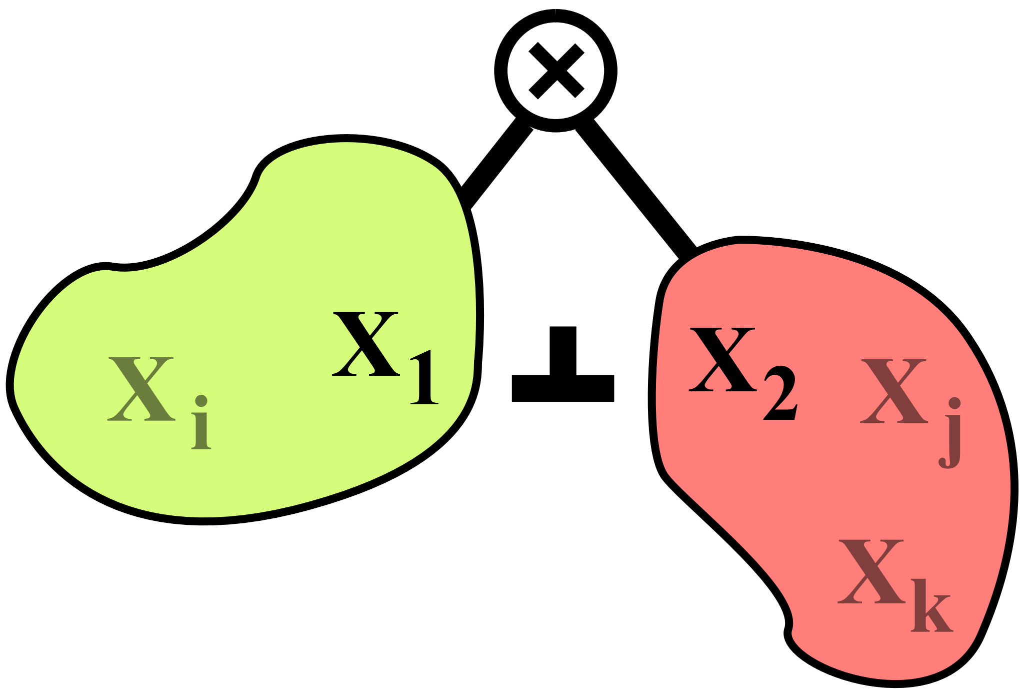
# LearnSPN Locally Optimizes Likelihood

# LearnSPN Locally Optimizes Likelihood



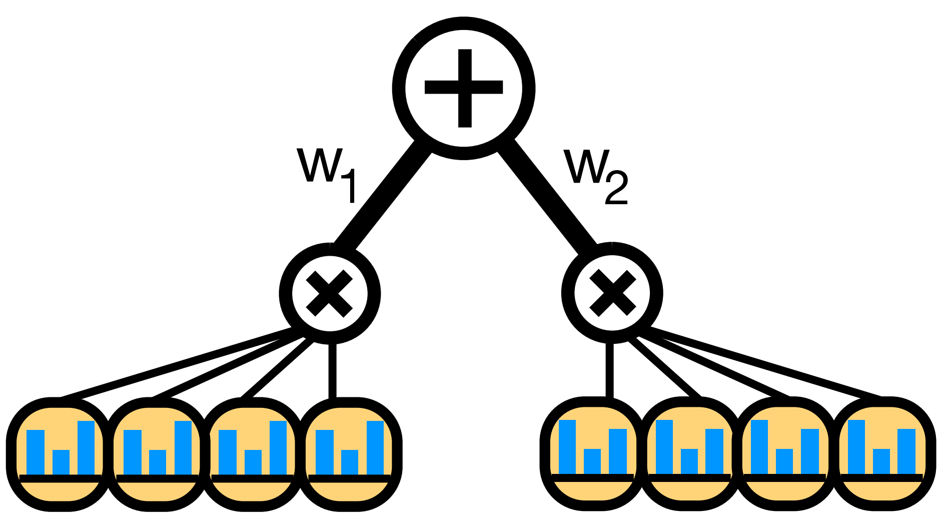

No loss of likelihood
if truly independent

# LearnSPN Locally Optimizes Likelihood



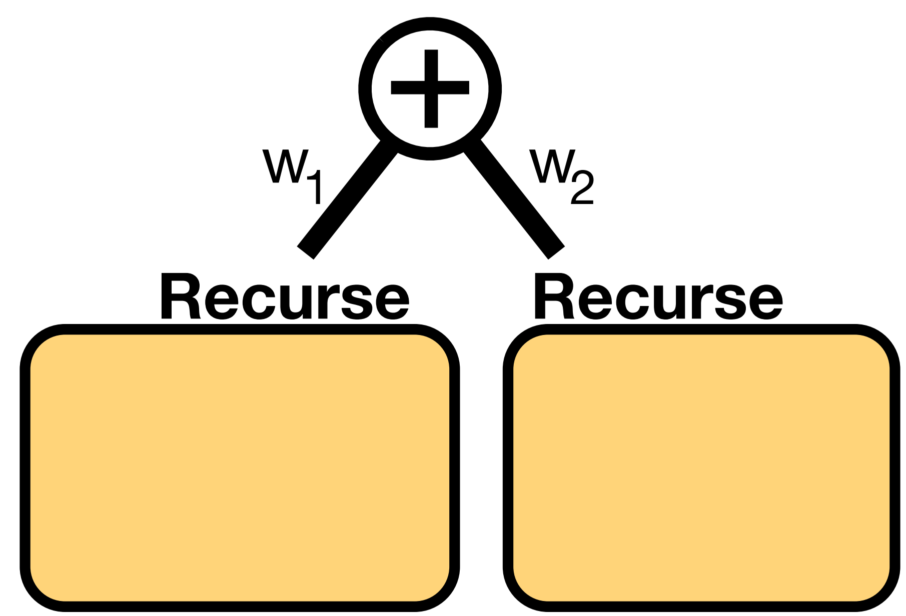

No loss of likelihood if truly independent

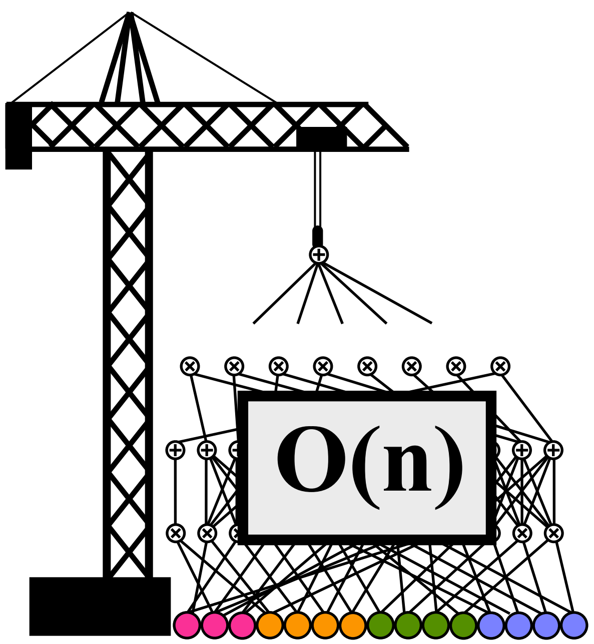
Naive Bayes likelihood



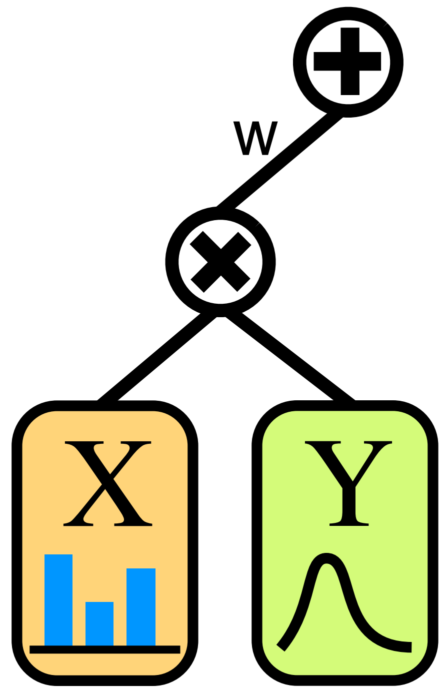

$\leq$

LearnSPN likelihood

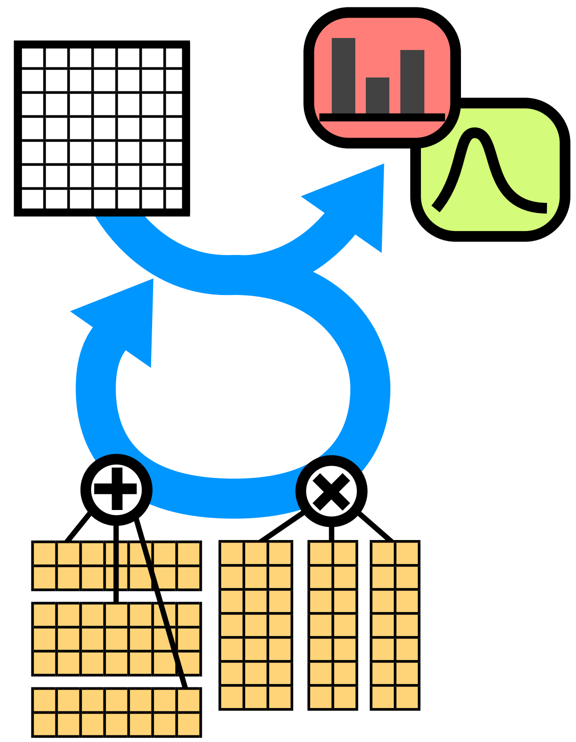




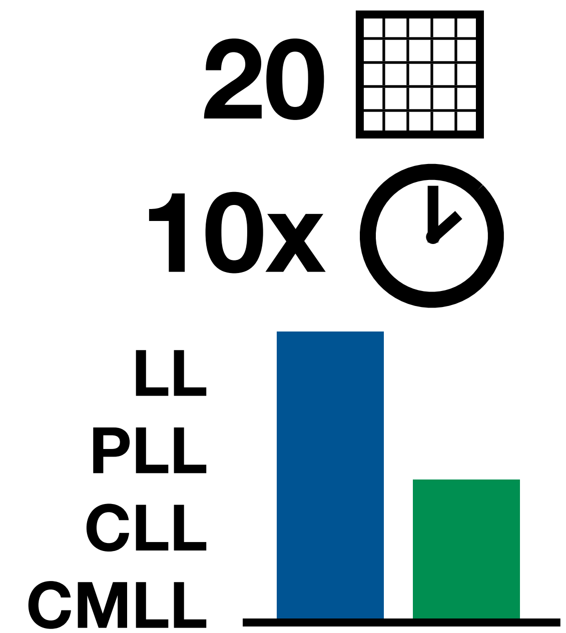

**Motivation**




**SPN Review**



**Structure Learning**



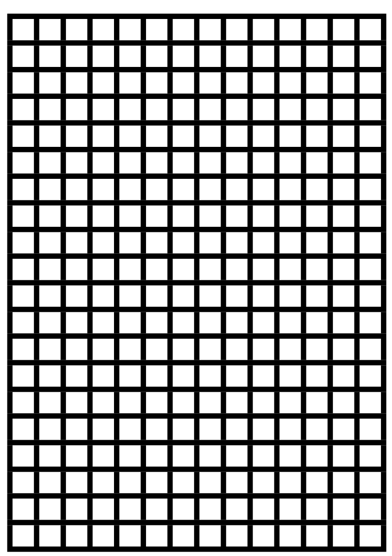

**Experiments**

# Experiments

## 20 Datasets

collaborative filtering
click-through logs
nucleic acid sequences
...

**Instances**
2k-388k



**Variables**
16-1556

| Representation | Learning | Inference |
|---|---|---|
| SPN | LearnSPN | Exact |
| Bayesian Network | WinMine | Gibbs |
| | | Loopy BP |
| Markov Network | Della Pietra | Gibbs |
| | | Loopy BP |
| | L1 | Gibbs |
| | | Loopy BP |

**<u>Total: 1680 Experiments</u>**

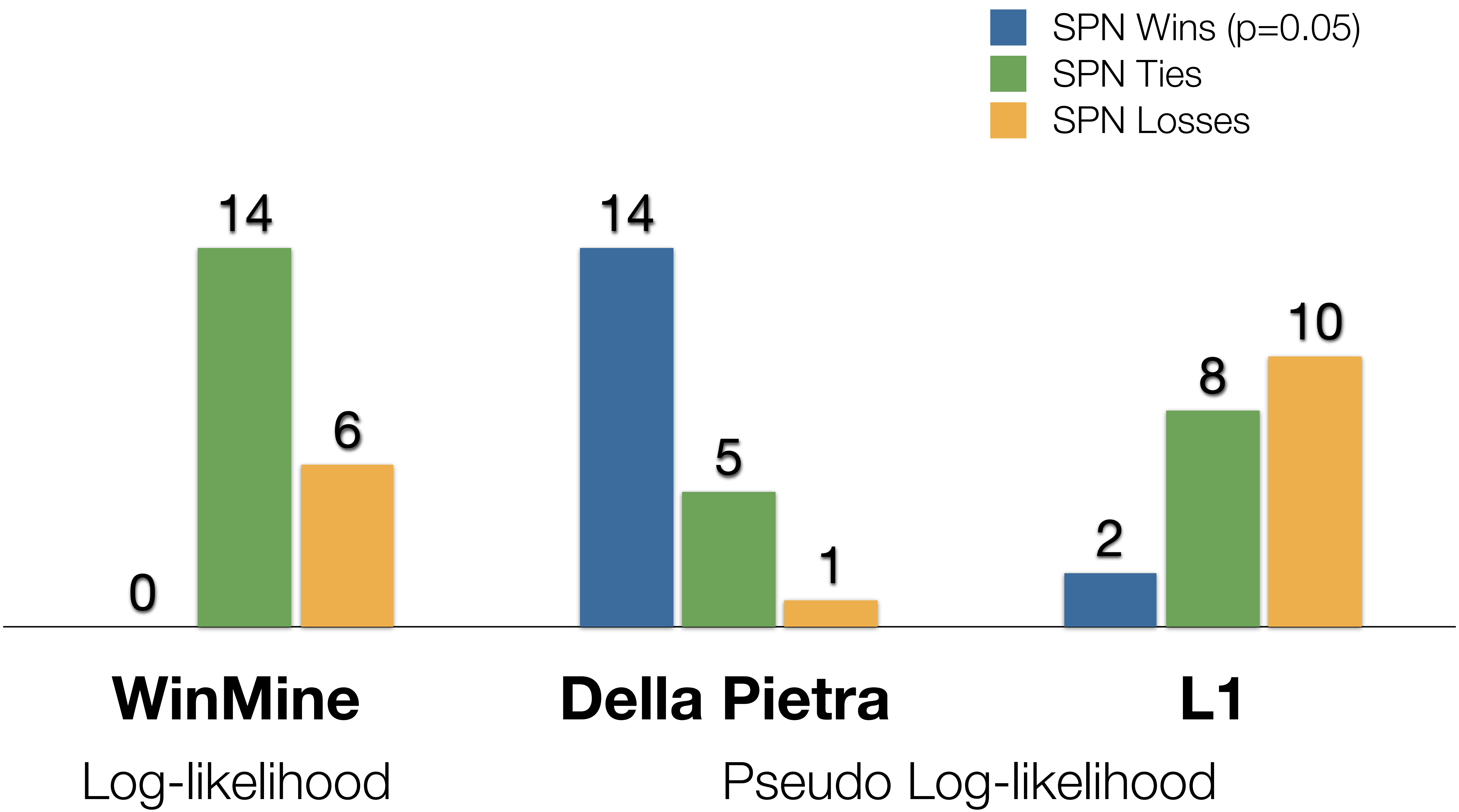
# Learning Results

SPN Wins (p=0.05)
SPN Ties
SPN Losses

**WinMine** **Della Pietra** **L1**

Log-likelihood Pseudo Log-likelihood

# Learning Results



# Inference

# Inference

$P(\, Query \mid Evidence \,)$

# Inference

*P( Query | Evidence )*

| Q | E |
|---|---|
| 10% | 30% |
| 20% | 30% |
| 30% | 30% |
| 40% | 30% |
| 50% | 30% |
| 30% | 0% |
| 30% | 10% |
| 30% | 20% |
| 30% | 40% |
| 30% | 50% |

For each proportion,
generate 1000
queries from test set

# Inference

*P( Query | Evidence )*

| Q | E |
|---|---|
| 10% | 30% |
| 20% | 30% |
| 30% | 30% |
| 40% | 30% |
| 50% | 30% |
| 30% | 0% |
| 30% | 10% |
| 30% | 20% |
| 30% | 40% |
| 30% | 50% |

For each proportion, generate 1000 queries from test set

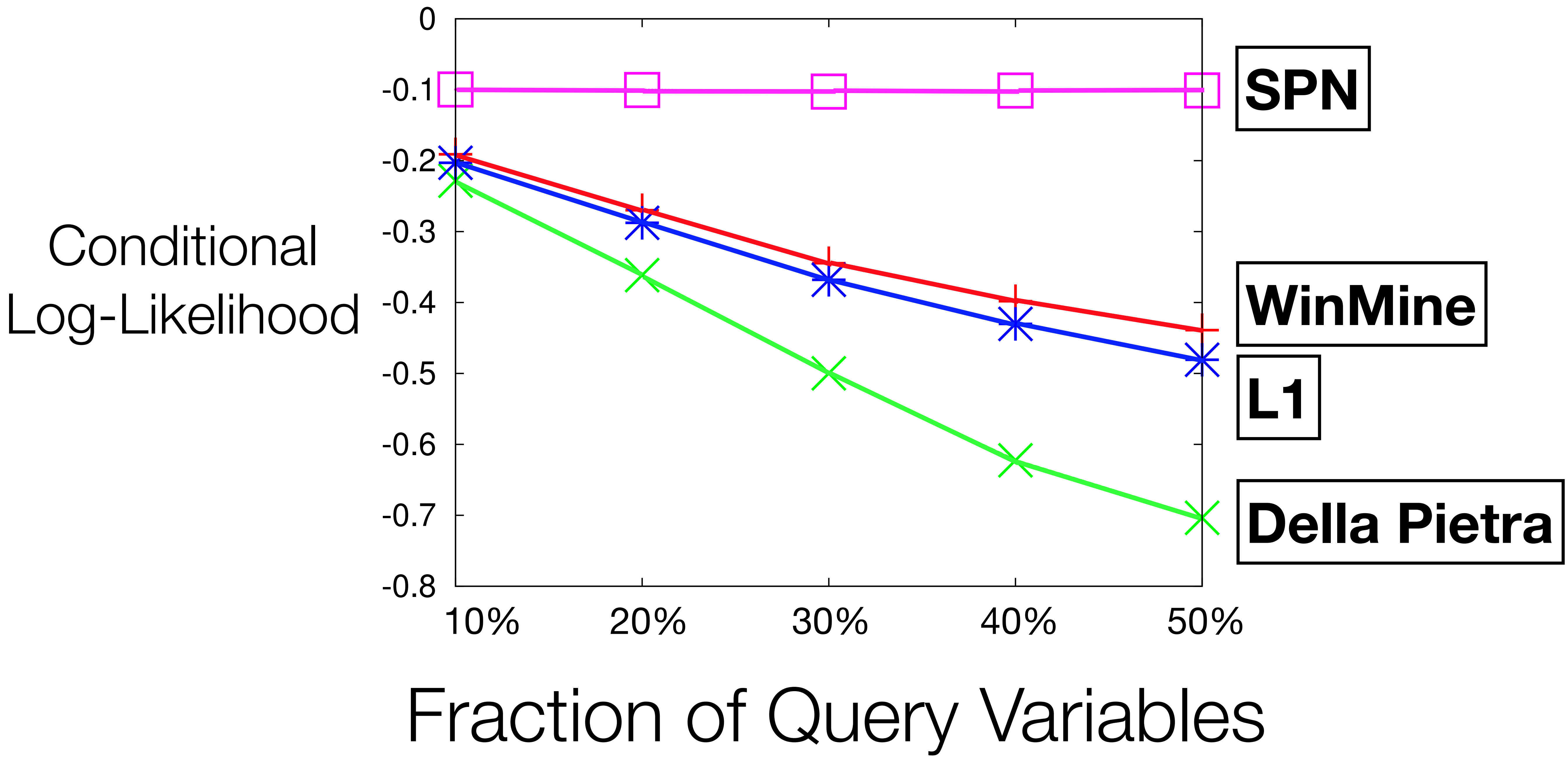
{ WinMine, Della Pietra, L1 } **X** 20 Datasets **X** 10 Variable proportions

**X** { Gibbs, Loopy BP } **= 1200 Experiments**

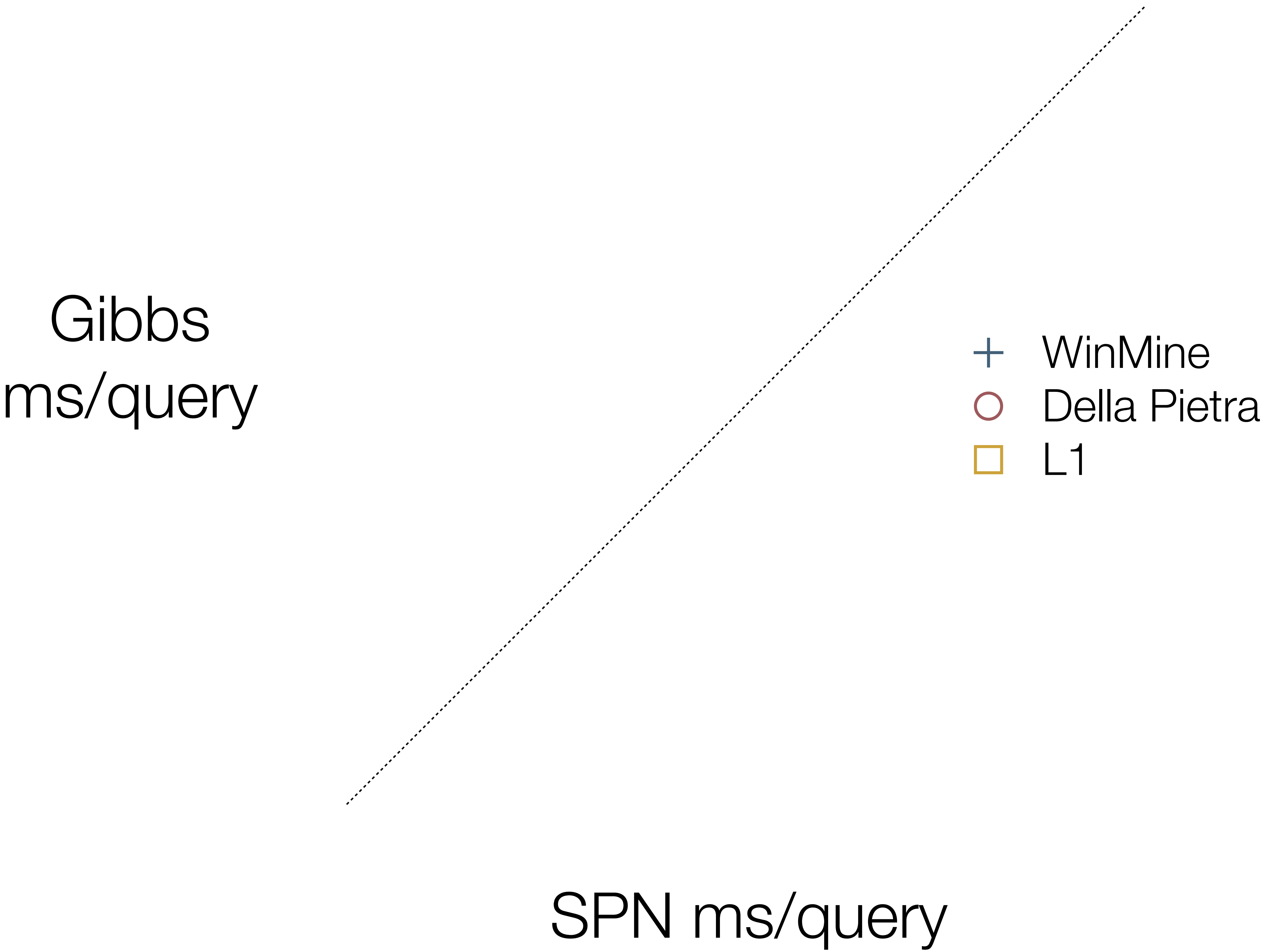
(In addition to 400 for SPNs)

# Inference Accuracy

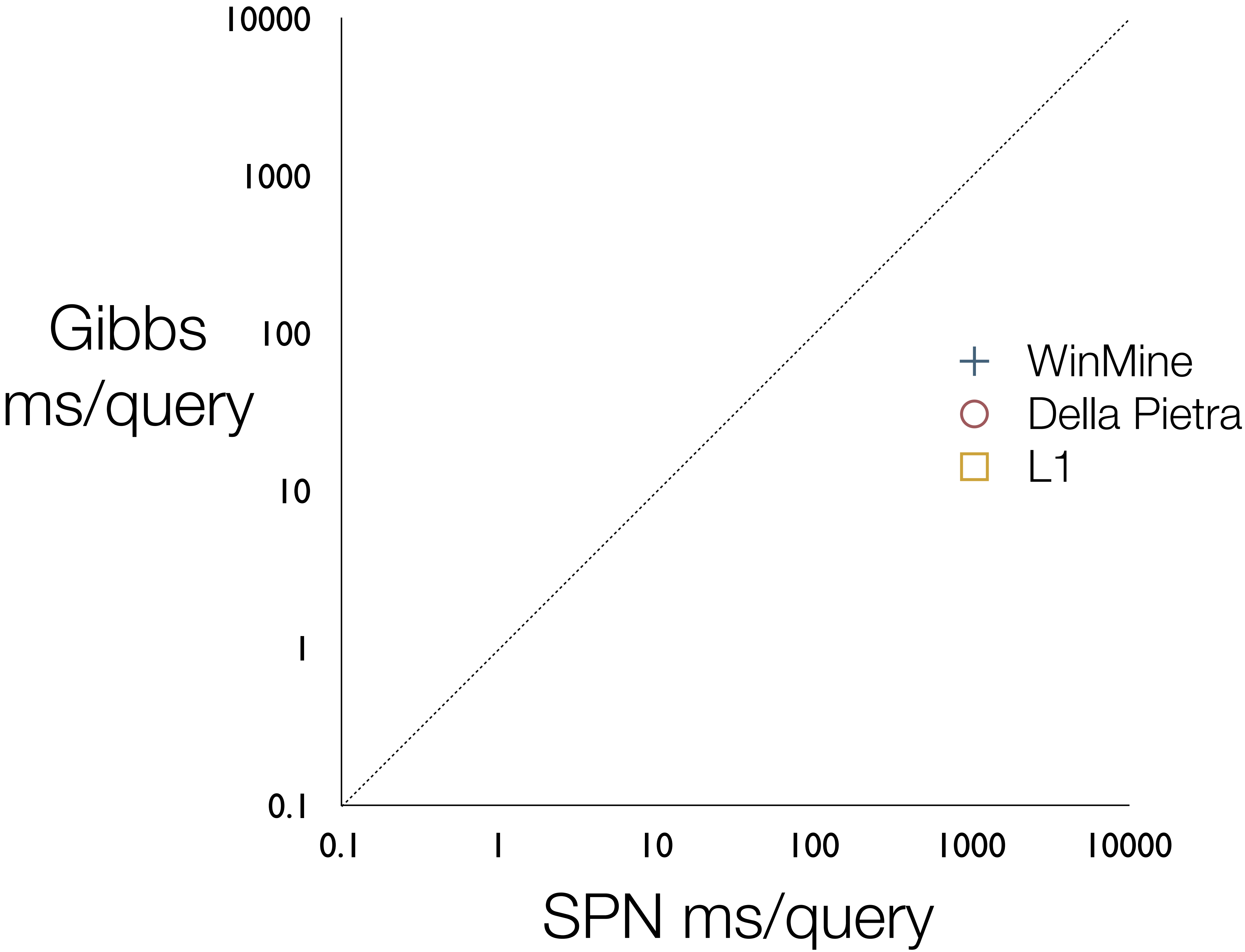
"EachMovie", 500 variables



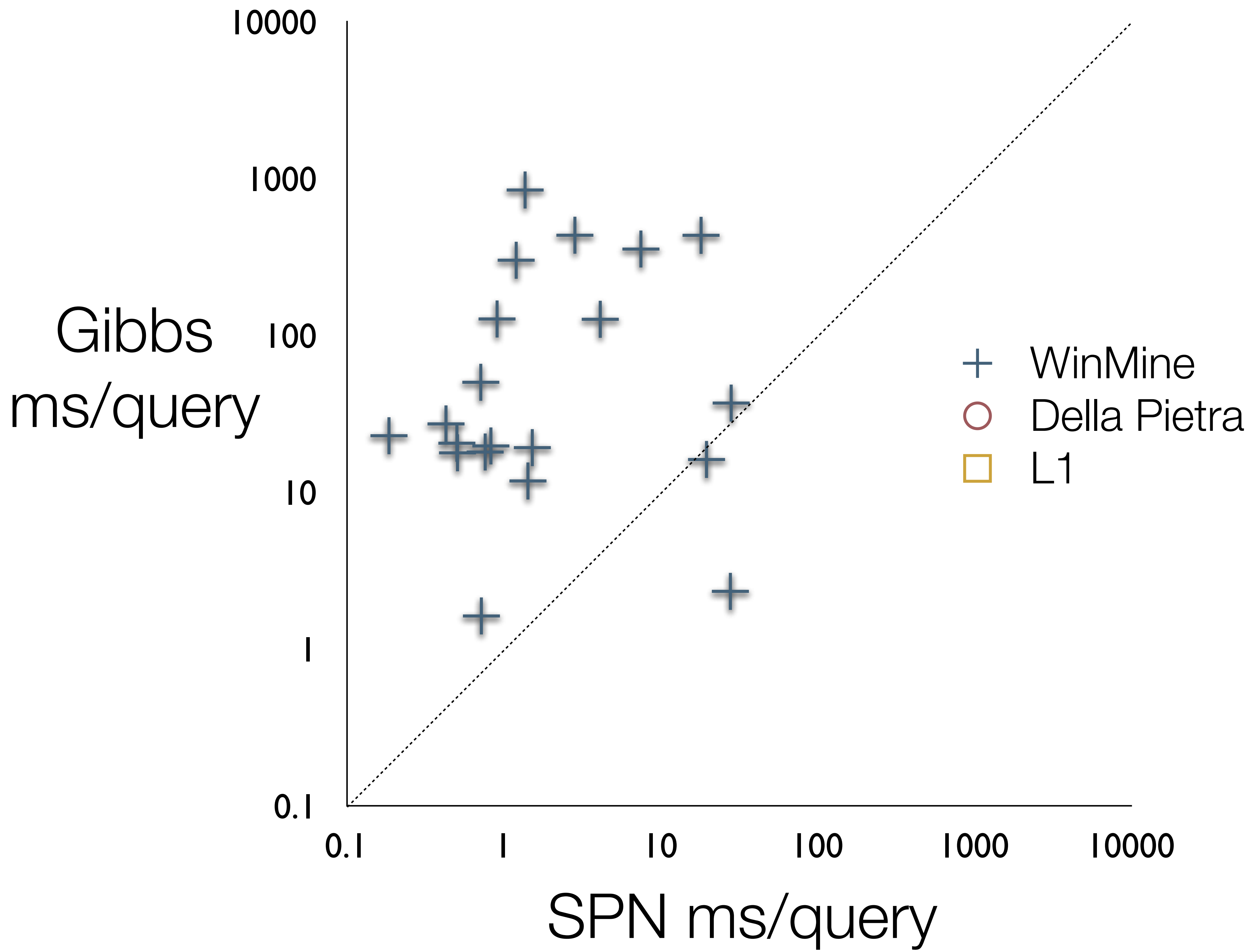
# Inference Time



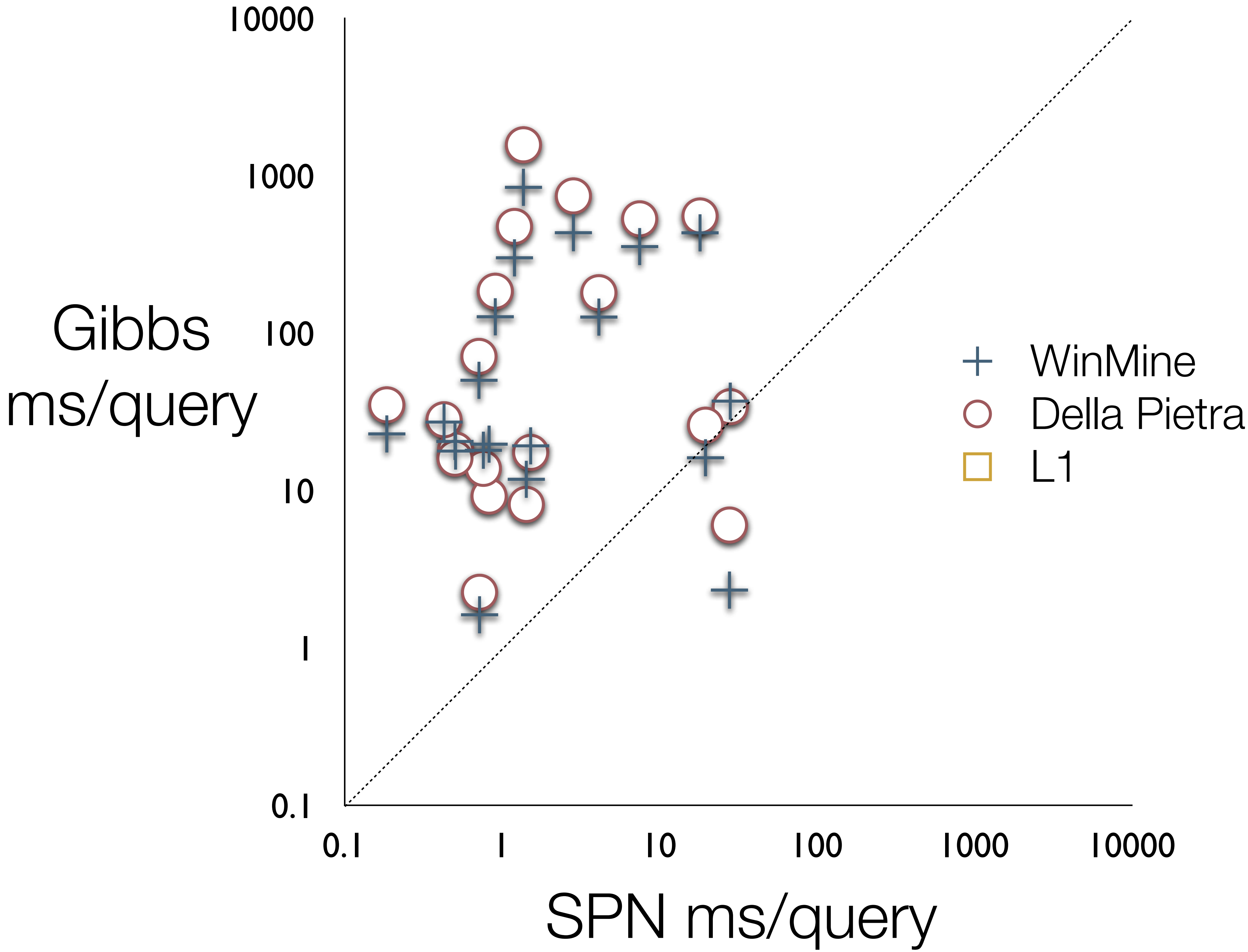
# Inference Time



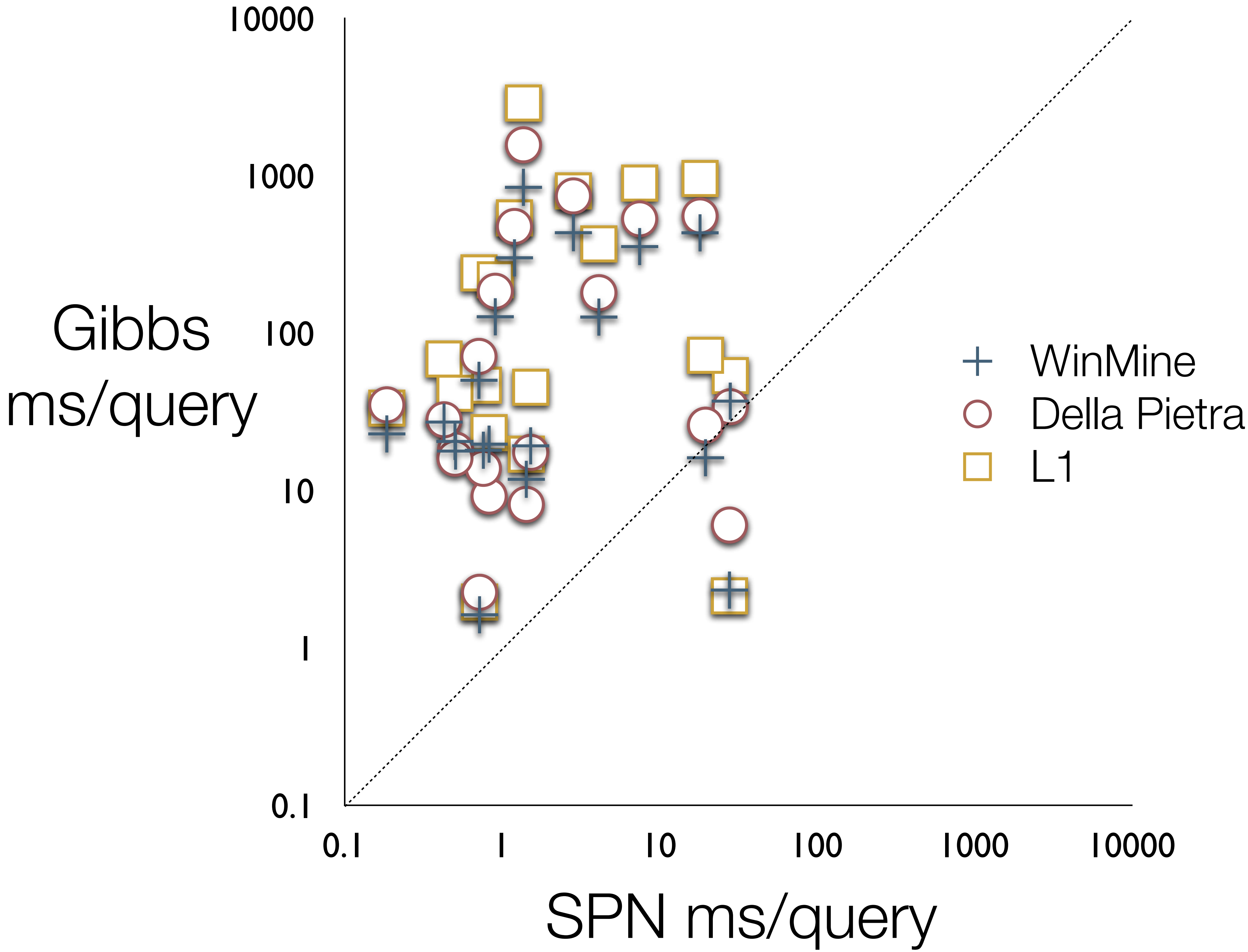
# Inference Time



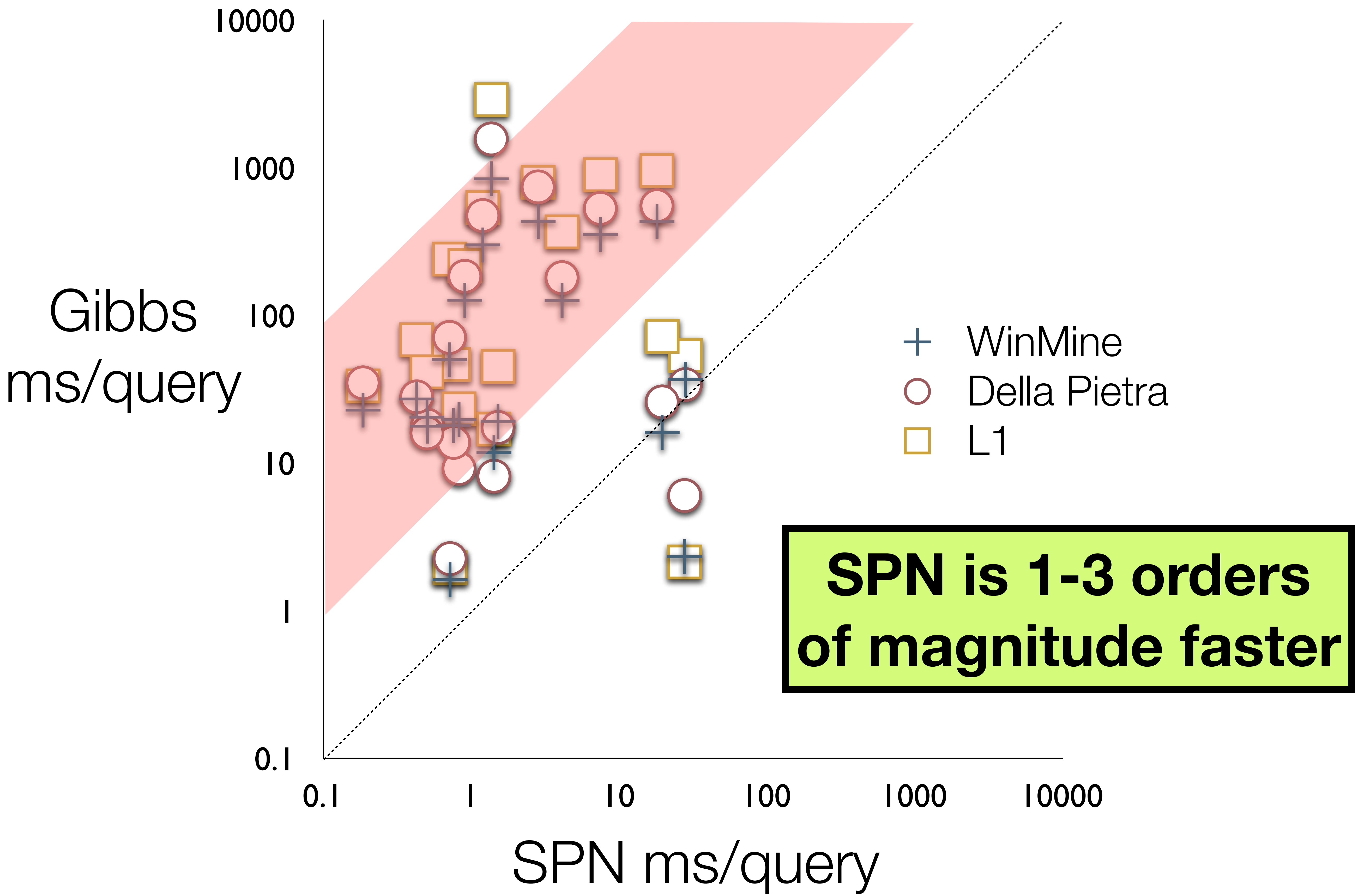
# Inference Time



# Inference Time



# Inference Time



# Inference Time

10000

Gibb

ms/qu

## Loopy Belief Propagation

Single variable marginals

{ WinMine, Della Pietra, L1 } X 20 Datasets X 10 Variable proportions

**= 600 Experiments**

**SPNs had higher conditional marginal log-likelihood on 78% of experiments**

(95% higher CLL vs. Gibbs)

SPN ms/query

# Experiment Conclusions

- SPN learning accuracy is comparable
- SPN exact inference is 1-3 orders of magnitude faster, and more accurate
- Inference does not involve tuning settings or diagnosing convergence
- Inference takes a predictable amount of time
- Can now apply SPNs to many domains

# Future Work

- Other SPN structure and weight learning algorithms
- Approximating intractable distributions with SPNs
- Parallelizing SPN learning and inference

Code and supplemental results available at spn.cs.washington.edu/learnspn/