Principles of Very Large Scale Modeling

Pedro Domingos
Dept. Computer Science & Eng.
University of Washington

Students and Collaborators

Rakesh Agrawal
Corin Anderson
Phil Bernstein
Nilesh Dalvi
Jesse Davis
Robin Dhamankar
Thomas Dietterich
AnHai Doan
Oren Etzioni
Ali Farhadi
Dieter Fox
Abe Friesen

Vibhav Gogate
Carlos Guestrin
Rob Gens
Alon Halevy
Geoff Hulten
Henry Kautz
Kristian Kersting
Dennis Kibler
Chloe Kiddon
Stanley Kok
Tessa Lau
Yoonkyong Lee

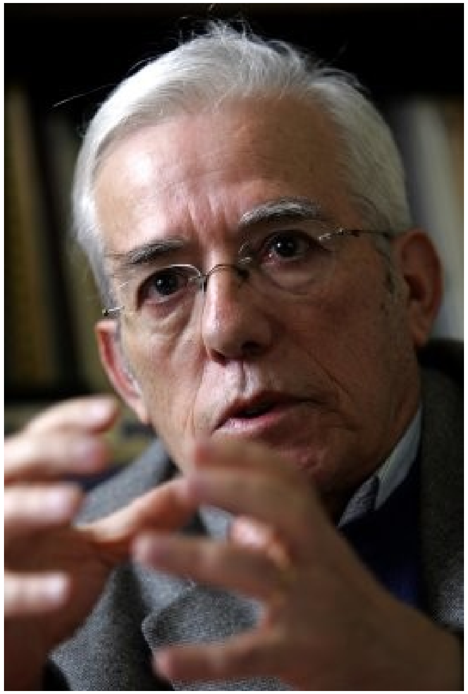
Daniel Lowd
Jayant Madhavan
Chris Manning
Mausam
Xu Miao
Tom Mitchell
Ray Mooney
Aniruddh Nath
Mathias Niepert
Michael Pazzani
Hoifung Poon
Foster Provost

Ross Quinlan
Matt Richardson
Parag Singla
Padhraic Smyth
Laurie Spencer
Josh Tenenbaum
Deepak Verma
Jue Wang
Austin Webb
Dan Weld
Steve Wolfman
Luke Zettlemoyer

. . . and many more

Thanks from all of us

IN MEMORIAM



J. J. DELGADO DOMINGOS
1935 - 2014

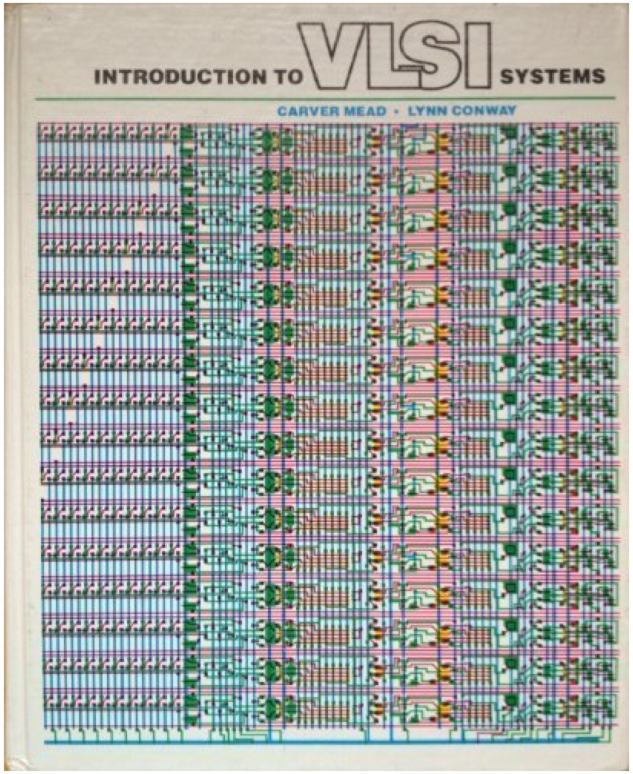
Road Map

- Very large scale modeling
- **First principle:**
  Model the whole, not just the parts
- **Second principle:**
  Tame complexity via hierarchical decomposition
- **Third principle:**
  Time and space should not depend on data size

Very Large Scale Modeling

Generation	Microchips	Models
Prehistory	Single components (pre-1960)	Descriptive statistics
First	Integration (1960-1970)	Small models (pre-1995)
Second	LSI (1970-1980)	Large models (1995-2015)
Third	VLSI (1980-now)	Very large models (2015 on)

The LSI-VLSI Transition



Before:

- Wire gates one by one
- Design tied to fabrication

After:

- Combine modules
- Design independent of fabrication
- CAD tools
- Hardware description languages

A Similar Transition Is Underway in KDD

Large Model	Very Large Model
Customer	Social network
Gene, protein	Metabolic pathway
Neuron	Brain
Service	City
Organism	Ecosystem
Atmosphere	Climate
Object recognition	Vision
Parser	Language
Recommender system	360º view of you

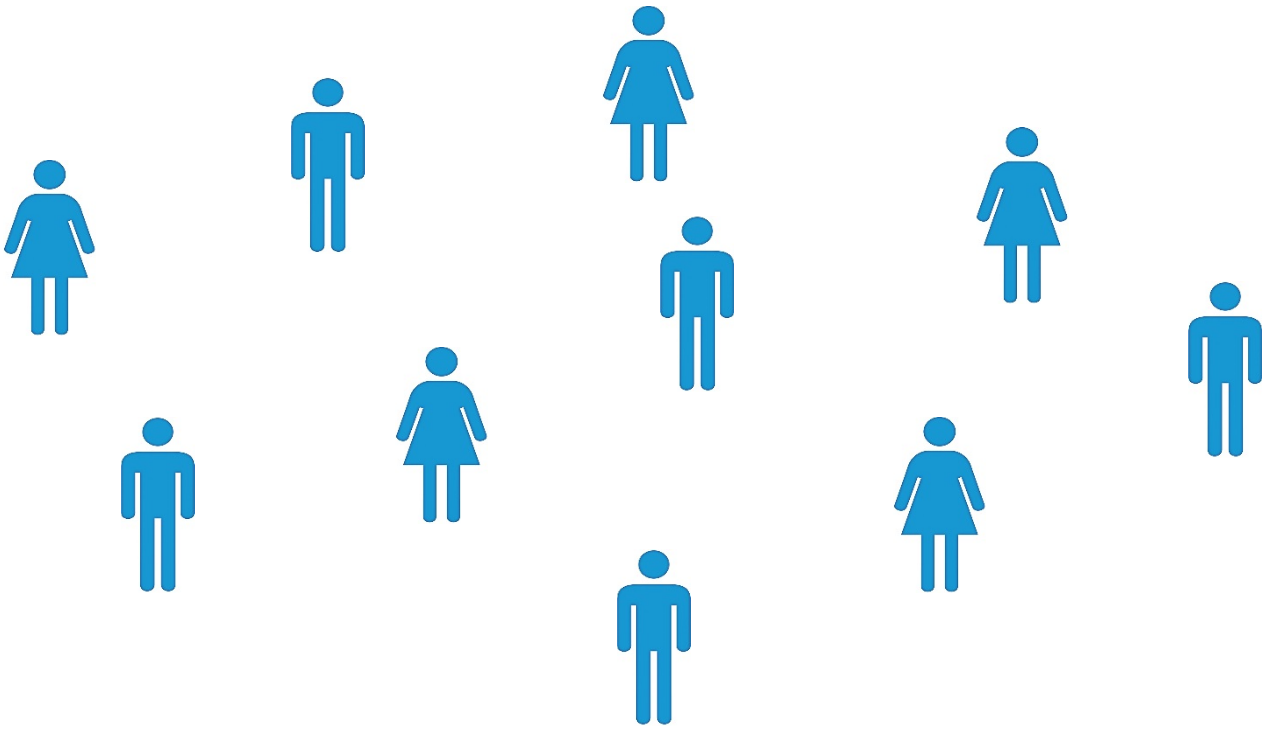
A Similar Transition Is Underway in KDD

- Not just more data, but modeling larger systems
- Poses host of new problems
- Requires new methodology
- This talk:
  - Three principles (of many)
  - Examples from my research

First Principle

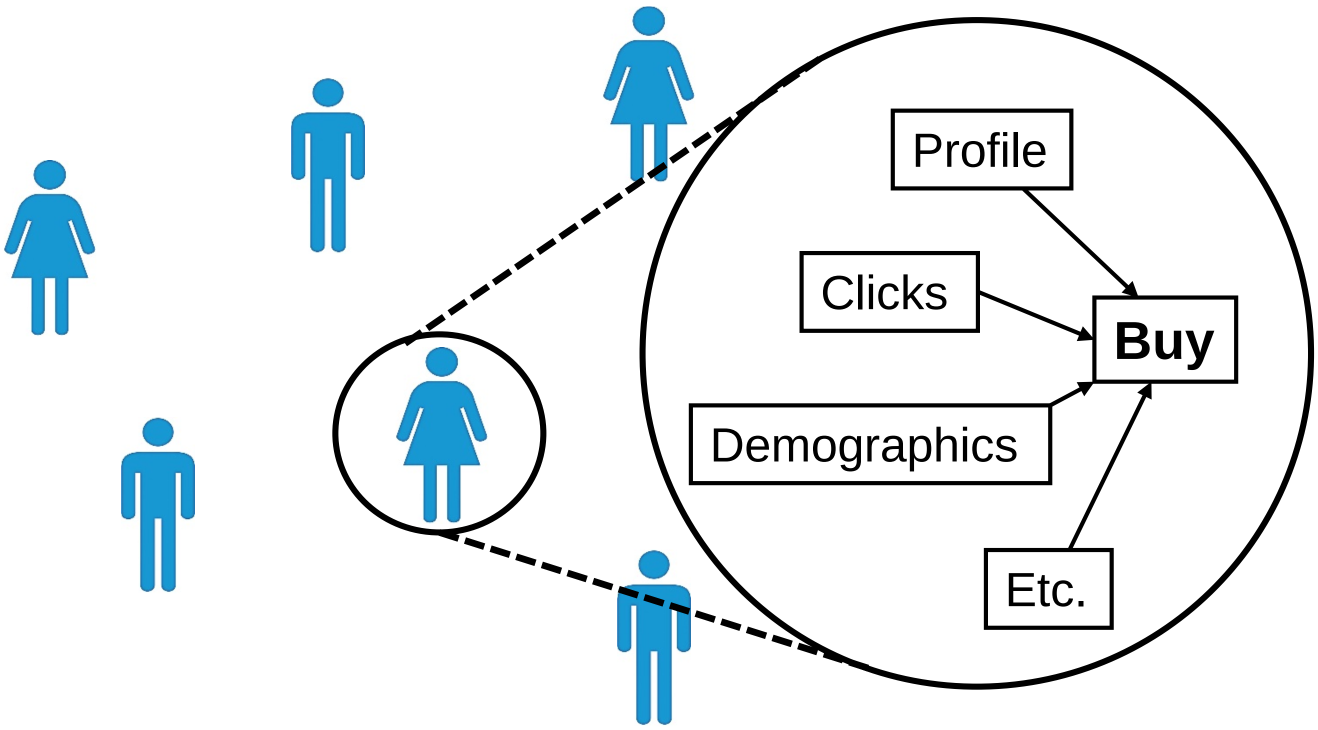
Model the whole, not just the parts

Example: Social Networks



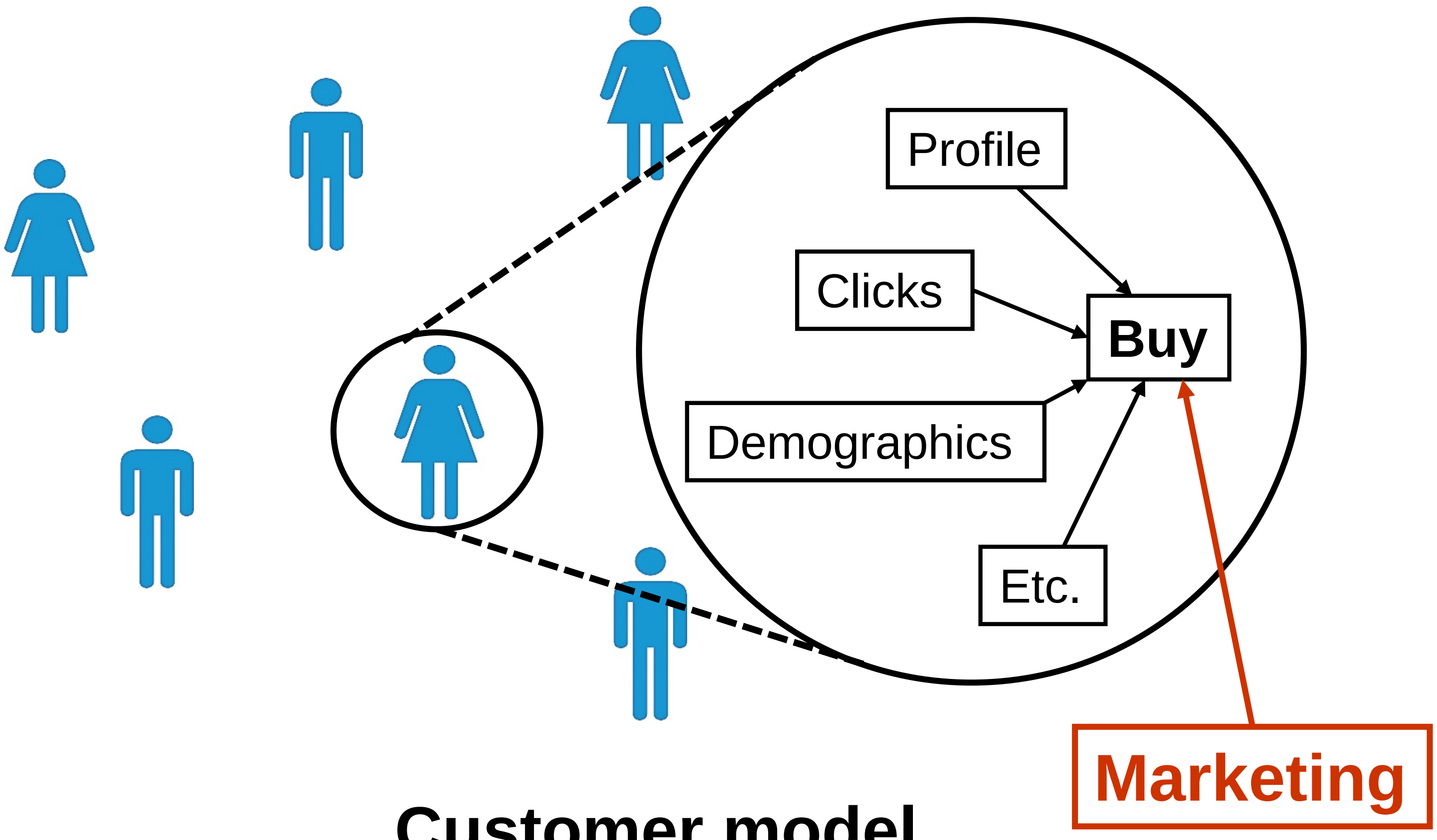
Customers

Example: Social Networks



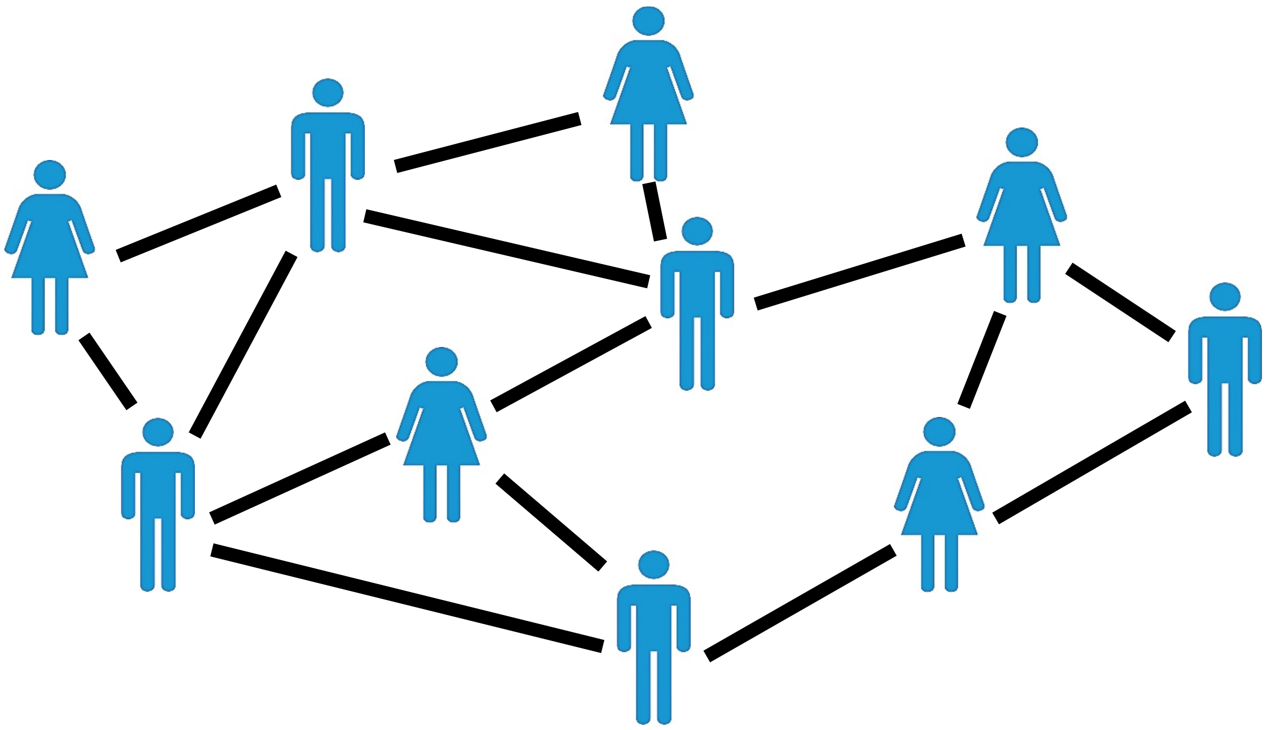
Customer model

Example: Social Networks



Customer model

Example: Social Networks



. . . but customers influence each other

Modeling the Whole Network

- Friends are often the largest influence on purchasing decisions
- If you don’t model the whole,
  you risk missing the forest for the trees
- But how do we model the whole?
- Traditional statistical models not applicable, because samples not independent
- Ad hoc methods don’t generalize,
  and don’t give you optimal actions

Markov Logic Networks

- Easy to represent interactions using logic:

$$Buys(x_1) \wedge Influences(x_1, x_2) \Rightarrow Buys(x_2)$$

$$Buys(Anna) \wedge Influences(Anna, Bob) \Rightarrow Buys(Bob)$$

- But logic rules are all-or-none;
  can’t model graded, uncertain behavior
- So treat them as feature templates for
  a log-linear model (Markov network)

From Log-Linear Models to MLNs

- Log-linear model:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x) \right)$$

Weight of Feature *i* (w_i) — Feature *i* ($f_i(x)$)

- Each instance of an MLN rule becomes a feature in the log-linear model
- If Anna influences Bob and both buy, probability goes up

MLN for Viral Marketing

$$\neg Buys(x)$$

$$MarketTo(x) \Rightarrow Buys(x)$$

$$Buys(x_1) \wedge Influences(x_1, x_2) \Rightarrow Buys(x_2)$$

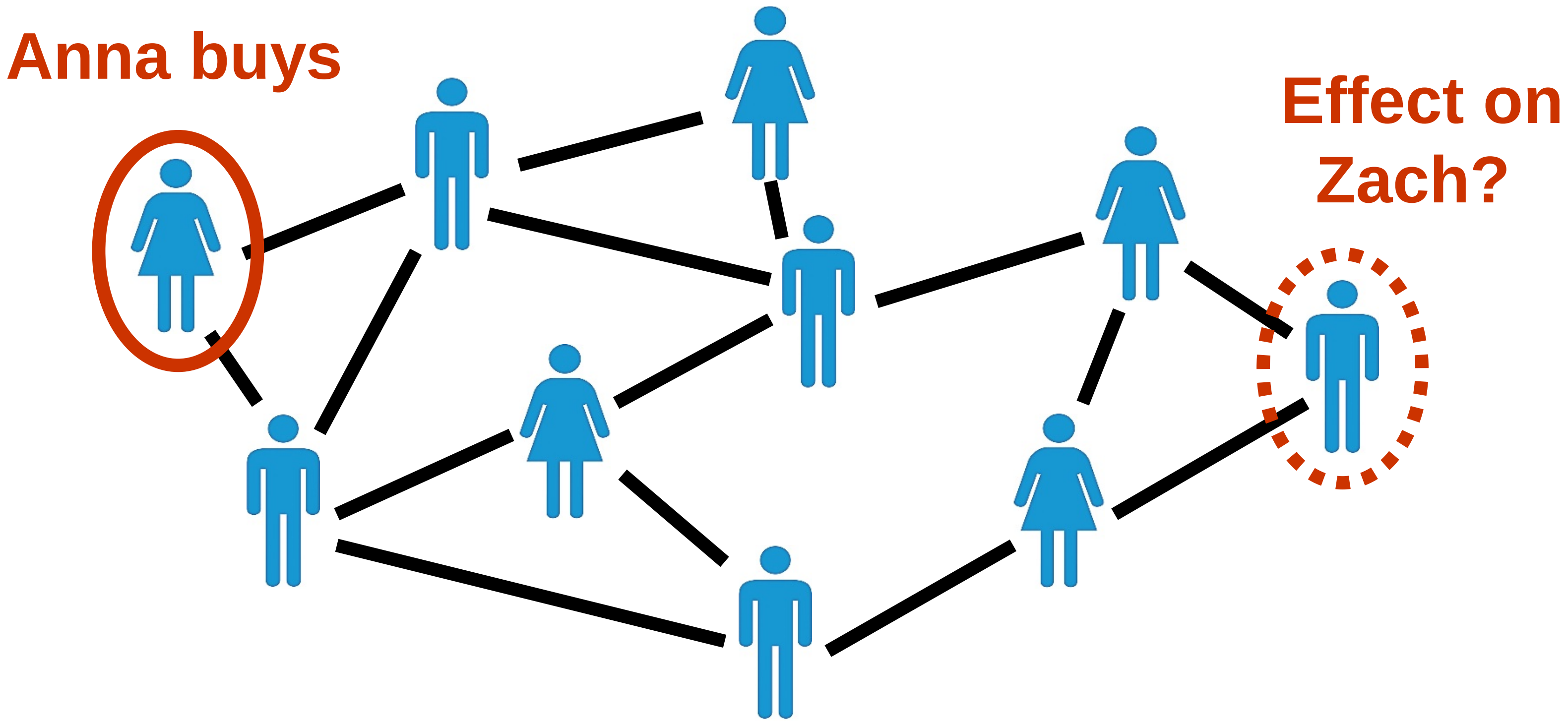
Richer MLNs

- Customer and product attributes
- Costs, prices and profits
- Multiple relations and entity types
- Choice of marketing actions
- Time
- Multiple products
- Multiple companies
- Etc.

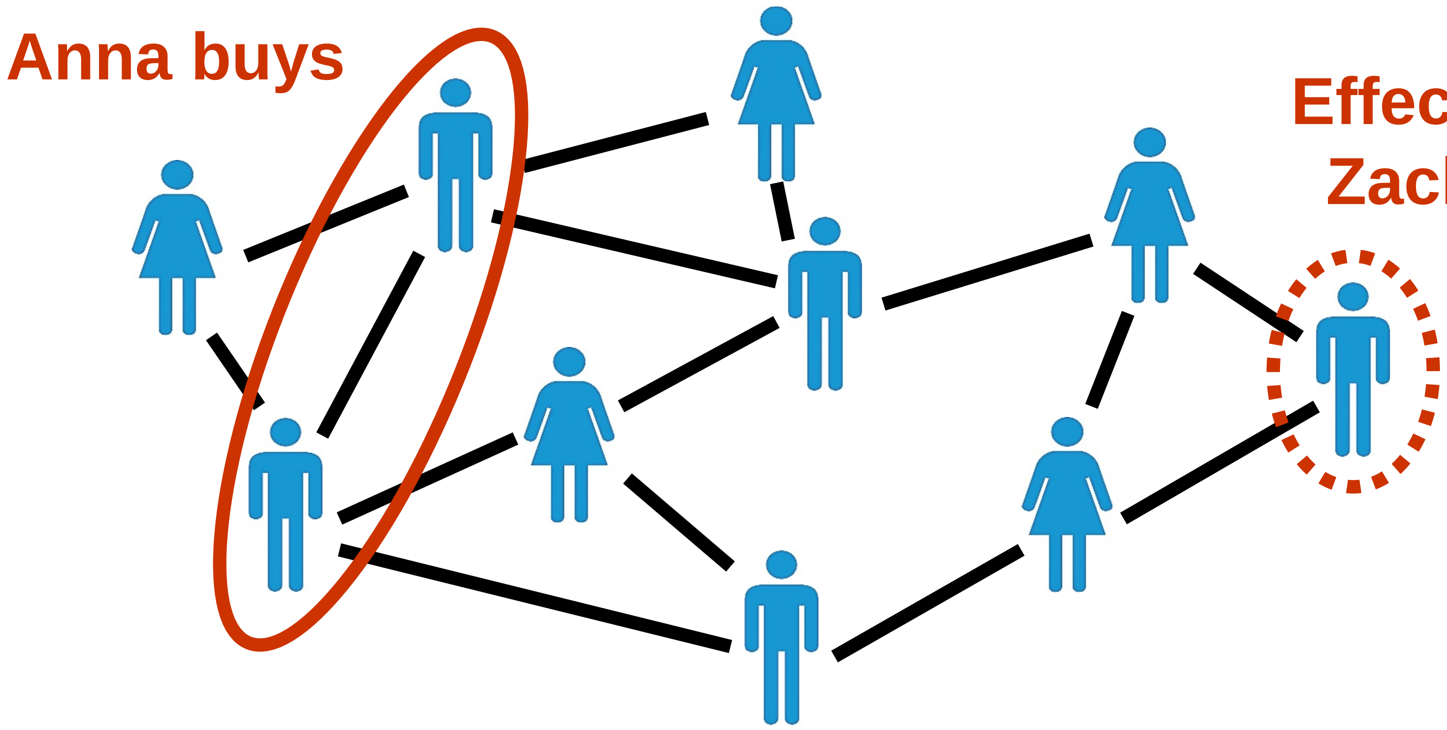
The Inference Problem

- But how do we use an MLN like this?
  - Choose initial set of customers to market to
  - Compute expected number of buyers
  - Search for optimal marketing strategy
- Highly intractable in general

The Cost of Inference

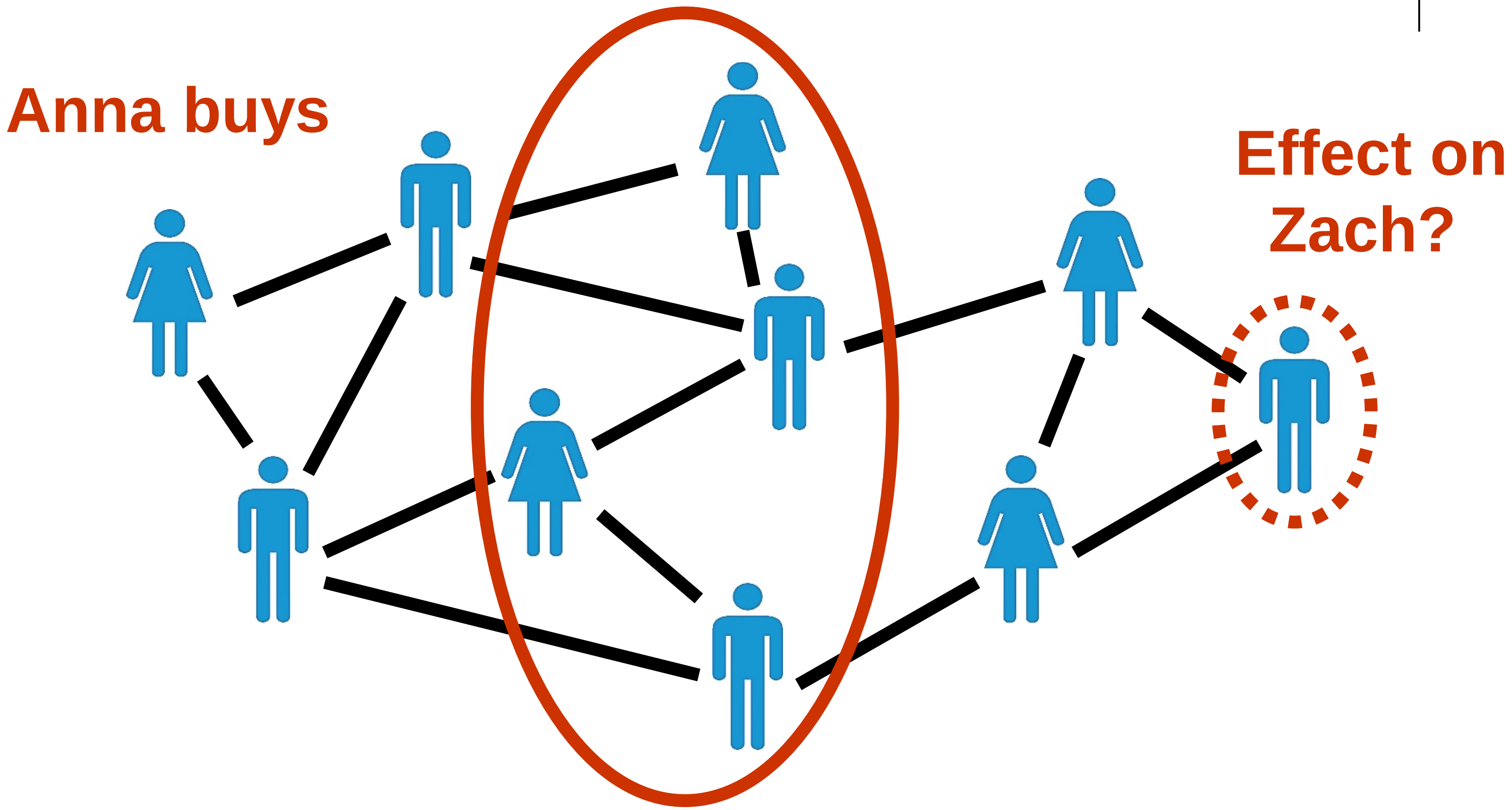


The Cost of Inference



Prob(Bob, Chris buy): 2^2 = 4 states

The Cost of Inference



Prob(Di, Ed, Fran, Glenn buy): 2^4 = 16 states

The Cost of Inference

- Cost of inference is exponential in width of network
- Same for every probabilistic model (etc.)
- Ad hoc inference can give arbitrarily bad results
- What to do?

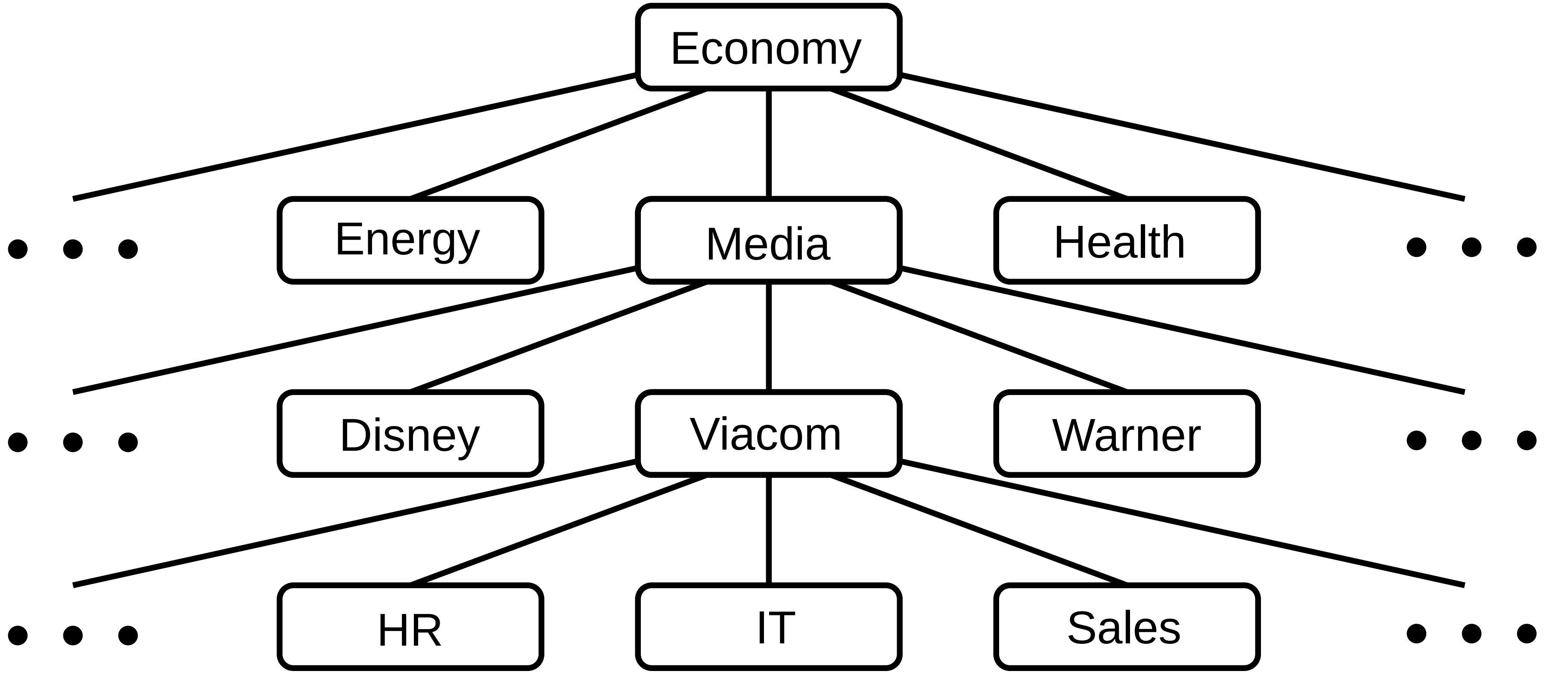
Second Principle

Tame complexity by hierarchical decomposition

Hierarchical Decomposition

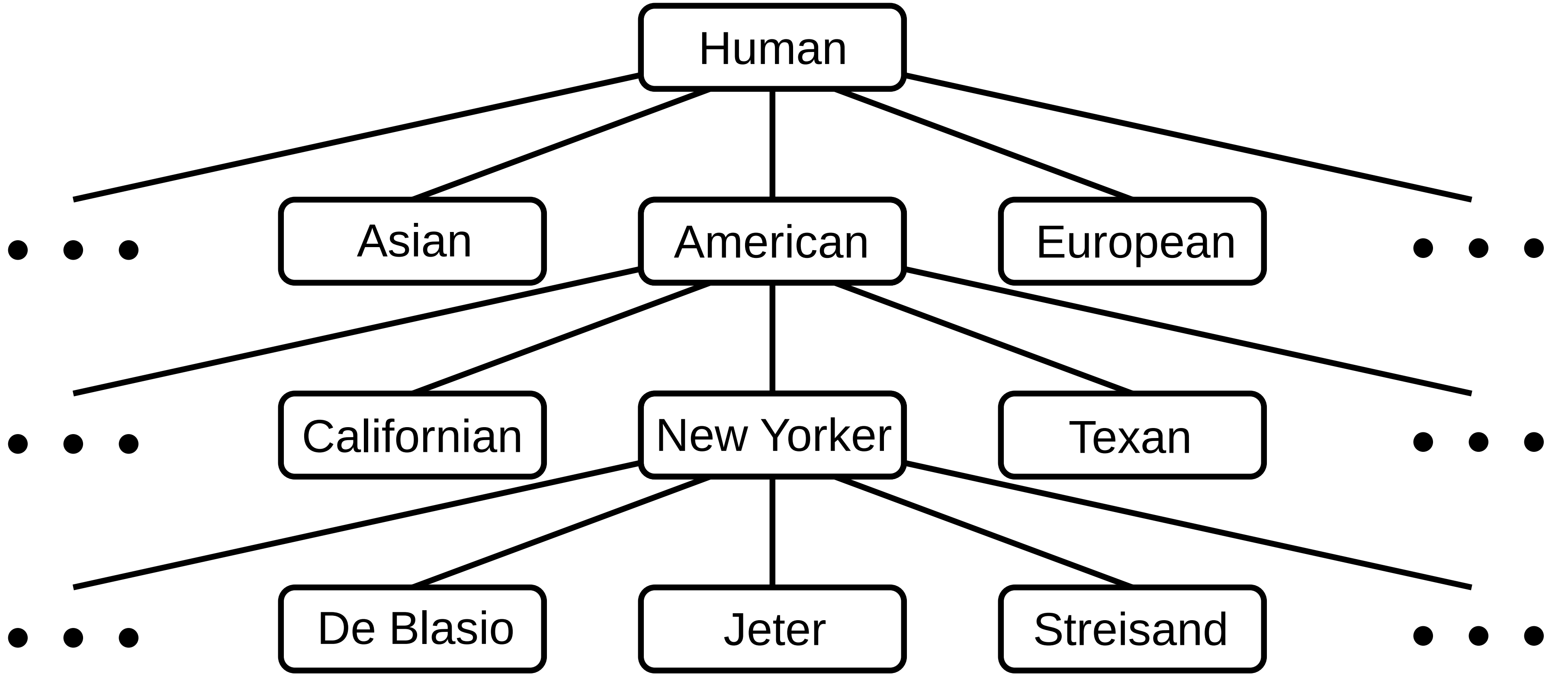
- Everyone does it . . . except us
- E.g.: VLSI, programming
- Most models are not hierarchical
- Why?
  - Little need so far
  - Phenomena are not hierarchical?

The World Is Hierarchical



Part Hierarchy

The World Is Hierarchical



Class Hierarchy

Hierarchical Decomposition

- Most phenomena are approximately hierarchical
- Even if not, we need to approximate them as such
- Better than assuming complete independence
- Better than intractable models

Exploiting Hierarchy

- Subparts are independent given part

  P(Co. buys, HR buys, IT buys, . . .) =

  P(Co. buys) P(HR buys | Co. buys)

  $\times$ *P(IT buys | Co. buys)* $\times$ *. . .*

- Probability for class is average over subclasses

  P(Buys | American) =

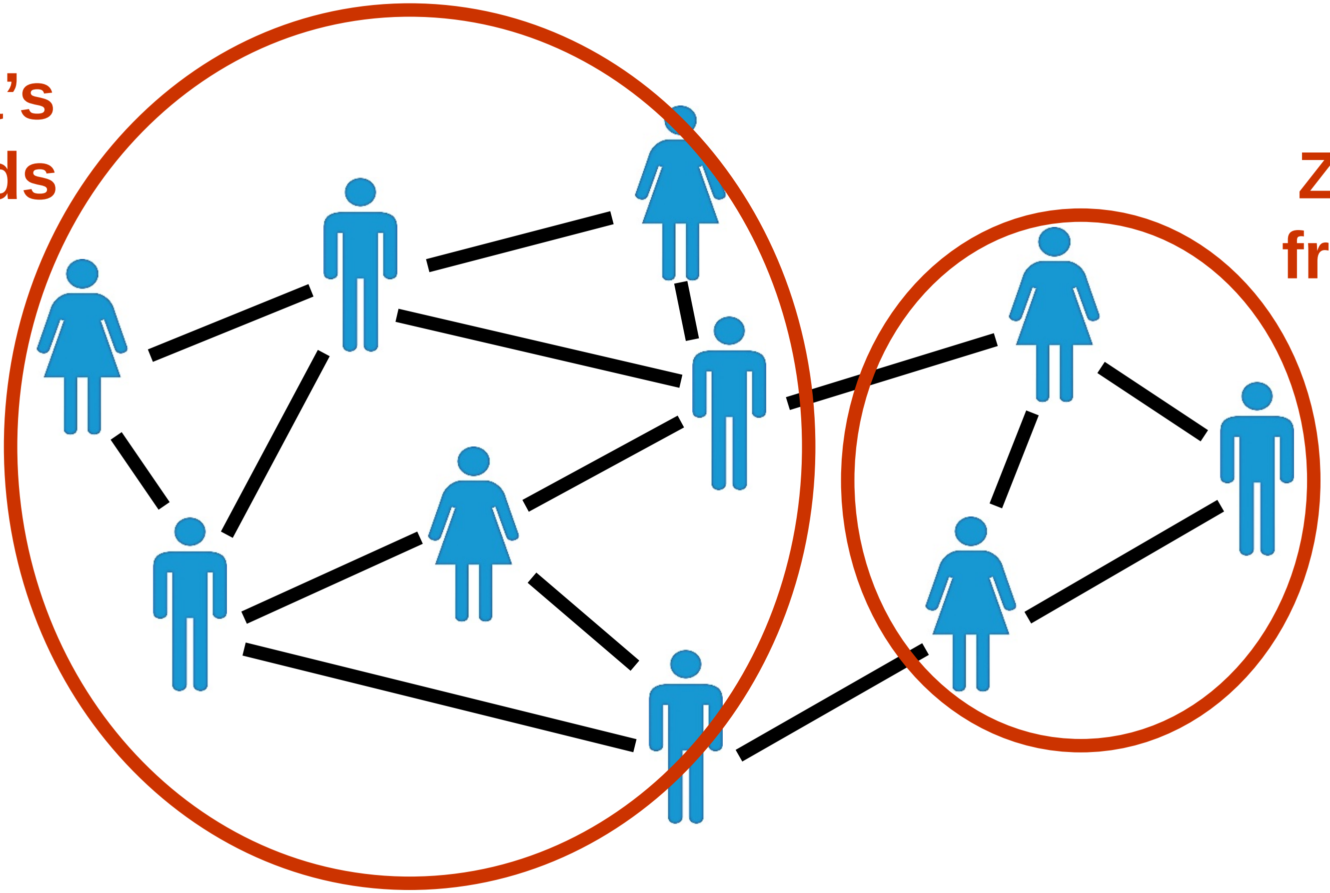
  P(New Yorker) P(Buys | New Yorker)

  + P(Californian) P(Buys | Californian) + . . .

- Combining the two ensures tractability

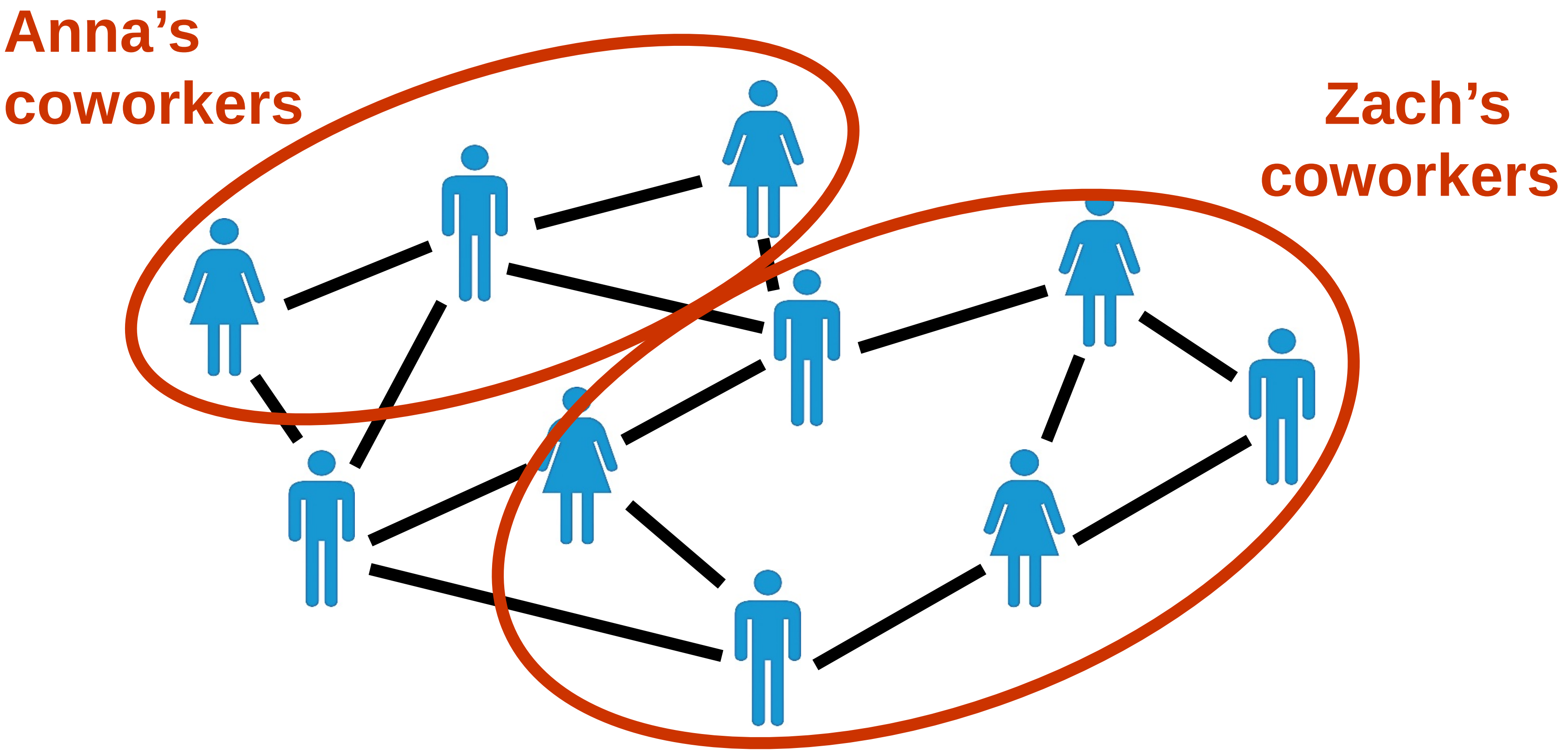
Buying an Item

Anna's friends

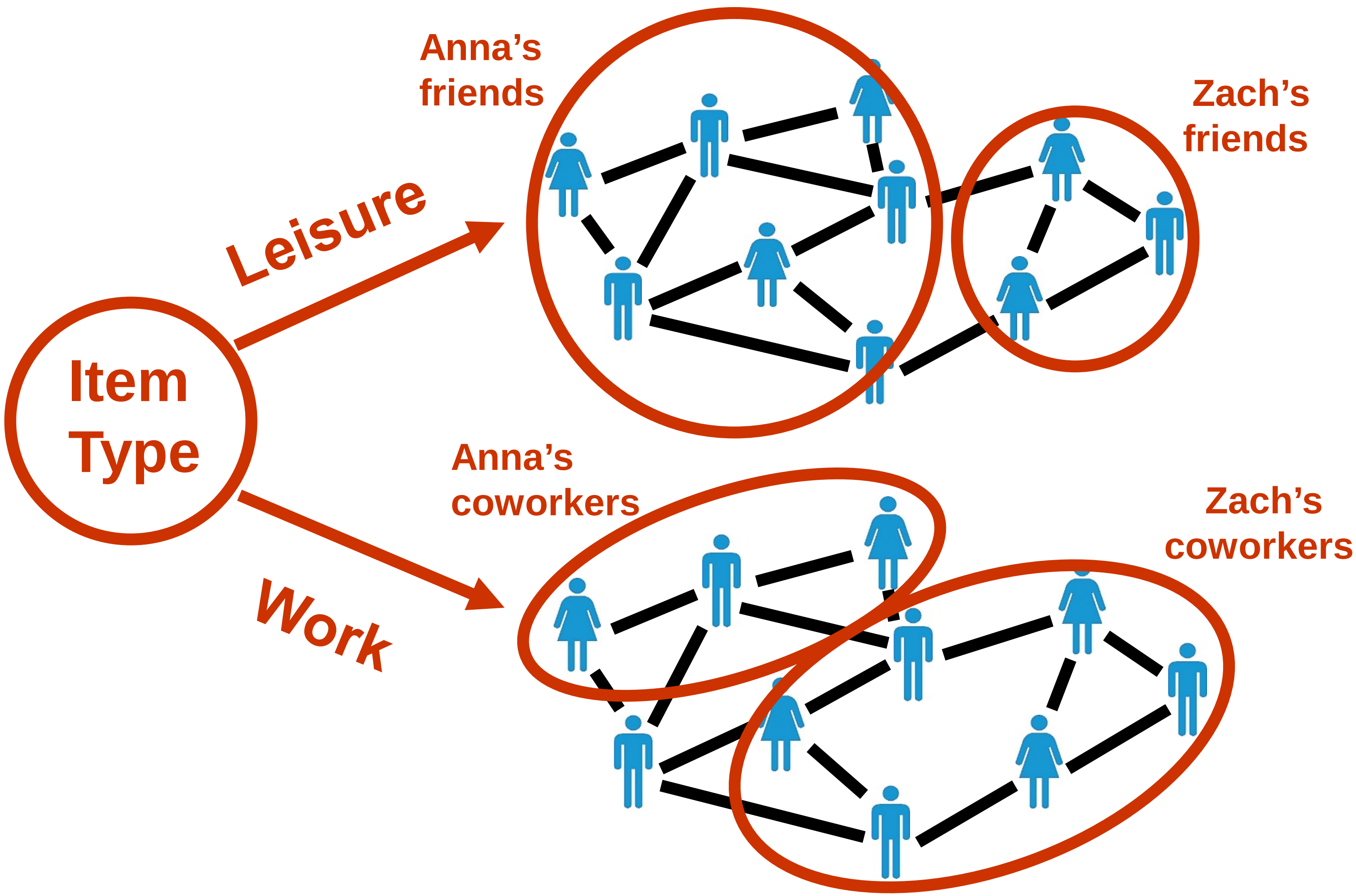


Zach's friends

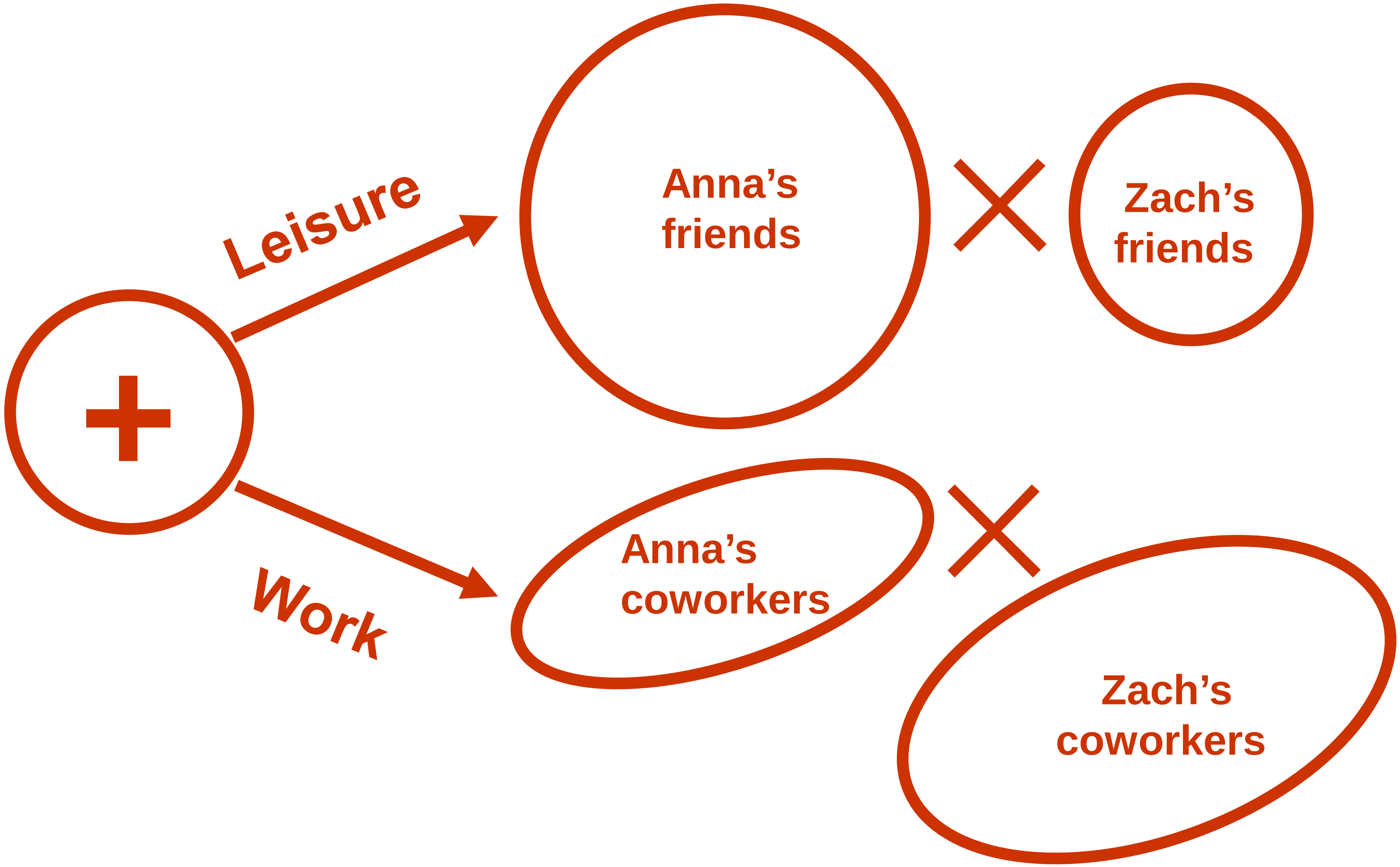
Buying an Item



Buying an Item



Buying an Item



The Sum-Product Theorem

A marginal probability can be efficiently computed if it is either:

- A weighted sum of efficiently computable marginals over the same variables

$$\sum_A (f(A) + g(A)) = \sum_A f(A) + \sum_A g(A)$$

- A product of efficiently computable marginals over disjoint variables

$$\sum_{A,B} f(A)g(B) = \left(\sum_A f(A)\right)\left(\sum_B g(B)\right)$$

The Sum-Product Theorem

- Recurse through any number of levels
- Allows many tractable wide networks
- Not just probabilities (any function, semiring)
- Decomposition can be learned, encoded, or approximation

Markov Logic Networks

\+ Sum-Product Theorem

Tractable Markov Logic

Third Principle

Time and space should not depend on data size

From Big Data to Big Models

- The purpose of big data is to learn a big model (or many small ones)
- Otherwise it's wasted
- Size of model dictates size of data you need
- Excess data can be ignored
- Just how much data is enough?

Streaming Bound Algorithms

- Infinite data stream
- Constant time and memory
- **Goal:** Learn approximately same model as we would with infinite time and memory

Why Is This Possible?

- To predict election winner, we only need to poll a few thousand voters
- All we have to do is generalize this to models with complex structure and many parameters
- Data in stream must be in random order

Learning an MLN

- Data is relational database
- Maximize likelihood
- Weight learning: Gradient descent

$$\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w\left[n_i(x)\right]$$

No. of true instances of rule *i* in data

Expected no. true instances according to MLN

- Structure learning: Greedy search

Hoeffding Bounds

- Model depends on data only through sufficient statistics (counts n_i)

- **Hoeffding bound:**
  With probability at least $1 - \delta$, true frequency p_i is within ϵ of empirical one $\hat{p}_i = \frac{n_i}{n}$, where

$$\epsilon = |p_i - \hat{p}_i| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}$$

Gradient Descent

- Log-linear model: $P(x) = \frac{1}{Z}\exp\left(\sum_{i=1}^{d} w_i f_i(x)\right)$
- Gradient: $\frac{\partial}{\partial w_i}\log P_{\mathbf{w}}(\mathbf{x}) = n_i - E_{\mathbf{w}}[n_i]$
- Gradient descent: $w_i \leftarrow w_i - \frac{r}{n}(n_i - E_{\mathbf{w}}[n_i])$

Gradient Descent Error Bound

- Error: $\epsilon = \sum_{i=1}^{d} |w_i - \hat{w}_i|$
- Initialization: $\epsilon_0 = 0$
- First step: $\epsilon_1 = \sum_{i=1}^{d} r \left|\frac{n_i}{n} - \frac{\hat{n}_i}{\hat{n}}\right| \leq dr\sqrt{\frac{\ln(2/\delta)}{2n}}$
- Second step:

$$|w_i - \hat{w}_i| \leq \epsilon_{i,1} + r\sqrt{\frac{\ln(2/\delta)}{2n}} + r|E_{\mathbf{w}}[p_i] - E_{\hat{\mathbf{w}}}[p_i]|$$

Gradient Descent Error Bound

- Error in unnormalized probabilities φ:
$$e^{-\epsilon}\varphi \leq \varphi \leq e^{\epsilon}\varphi$$
- Expectation error:
$$|E_{\mathbf{w}}[p_i] - E_{\hat{\mathbf{w}}}[p_i]| \leq \left|\frac{\hat{p}_i}{\hat{p}_i - e^{\pm 2\epsilon}(1-\hat{p}_i)} - \hat{p}_i\right|$$
- Error in sth step:
$$\epsilon_s \leq \epsilon_{s-1} + dr\sqrt{\frac{\ln(2/\delta)}{2n}} + r\left|\frac{\hat{p}_i}{\hat{p}_i - e^{\pm 2\epsilon_{s-1}}(1-\hat{p}_i)} - \hat{p}_i\right|$$
- Error grows roughly linearly with # steps

Streaming Gradient Descent

- Start with optimistic sample size
- If bound exceeded, restart with $2\times$ sample
- Time and space independent of data size
- Much tighter than PAC bounds
- Lets you do with one CPU what might otherwise take thousands
- Structure learning: divide δ by # models tried

What If Data Changes Over Time?

- Maintain sufficient statistics over sliding window
- Monitor difference between current statistics and statistics model was learned on
- If difference exceeds bound for some statistic, relearn corresponding part of model

Applications to Date

- Viral marketing
- Web knowledge bases
- Object recognition
- Web caching
- Semantic parsing
- Etc.

Applications to Date

- Viral marketing
- Web knowledge bases
- Object recognition
- Web caching
- Semantic parsing
- Etc.

Open source software:
Alchemy, SPN, VFML, etc.

Principles of Very Large Scale Modeling

1. Model the whole, not just the parts
   → *Markov logic networks*
2. Tame complexity via hierarchical decomposition
   → *Sum-product theorem*
3. Time and space should not depend on data size
   → *Streaming bound algorithms*

The Master Algorithm

Machine Learning and the
Big Data Revolution

Pedro Domingos

Basic Books