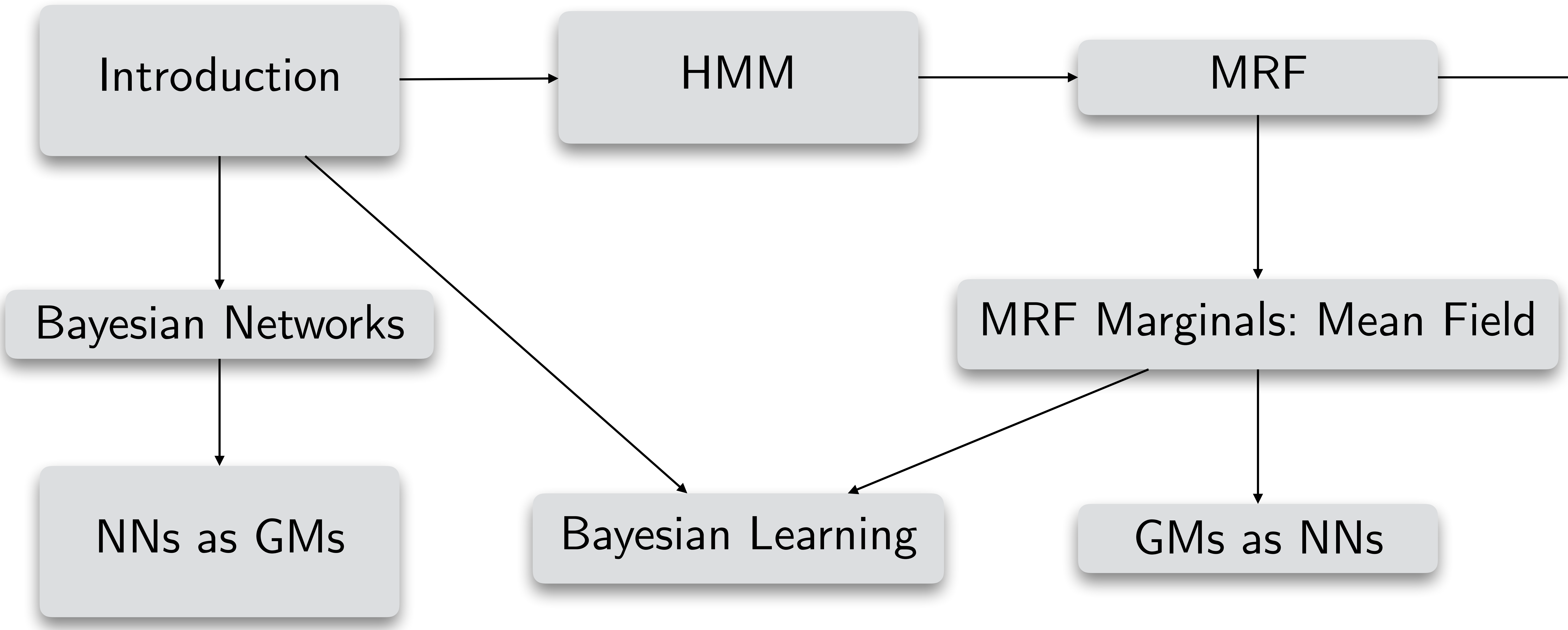
Introduction to Graphical Models

Alexander Shekhovtsov

2018

https://cw.fel.cvut.cz/wiki/courses/ucuss18

Roadmap


Introduction
HMM
MRF
MRF MAP
Bayesian Networks
MRF Marginals: Mean Field
NNs as GMs
Bayesian Learning
GMs as NNs

Introduction: What are Graphical Models

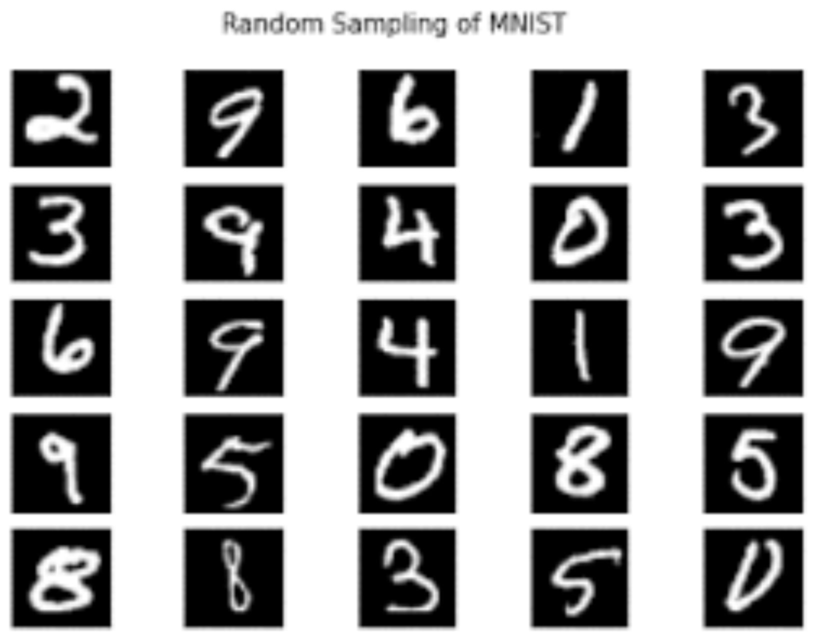
Basic Classification Problems

Two-class

$\{0, 1\}$

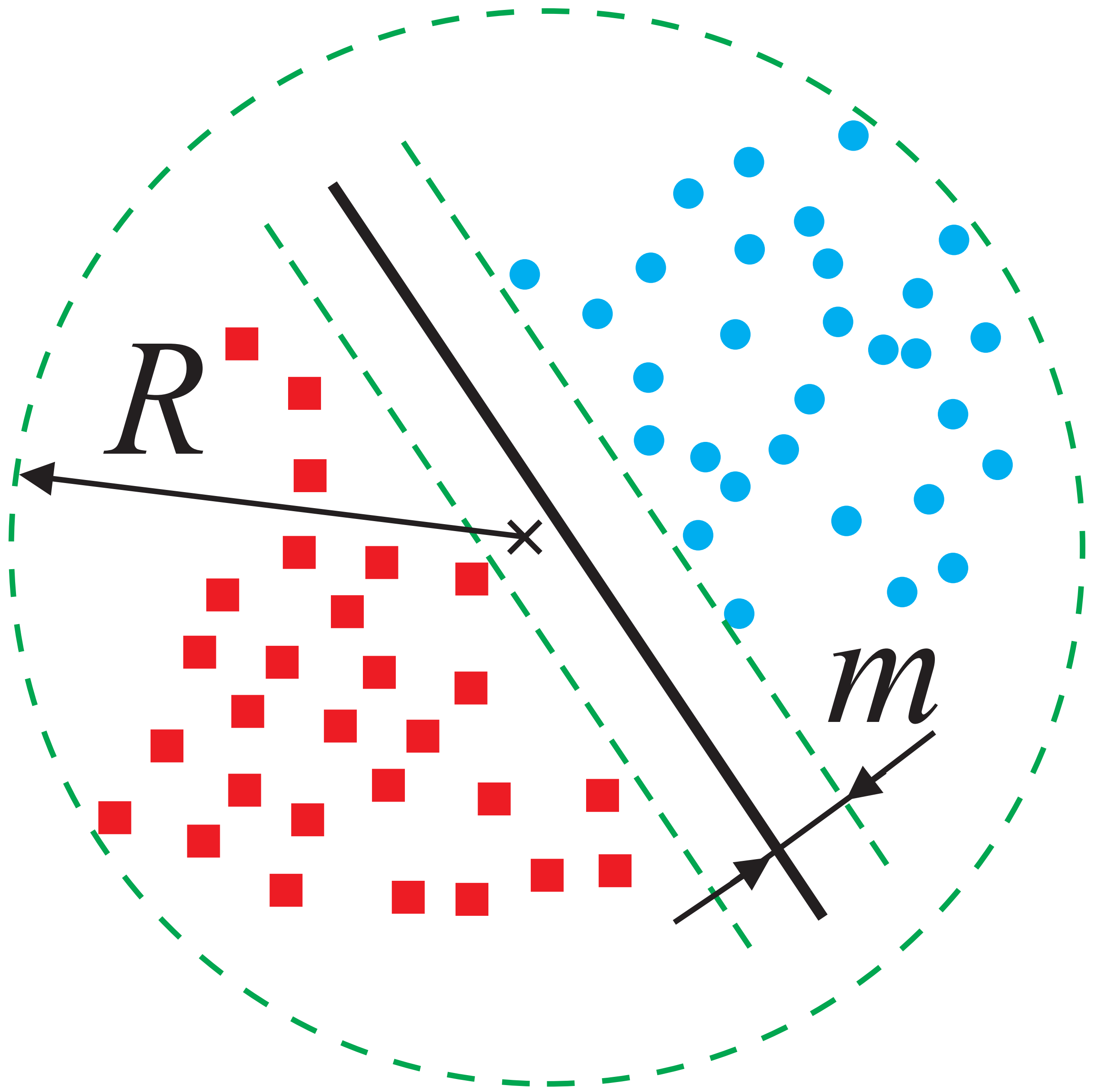
Multi-class

$\{1,..K\}$



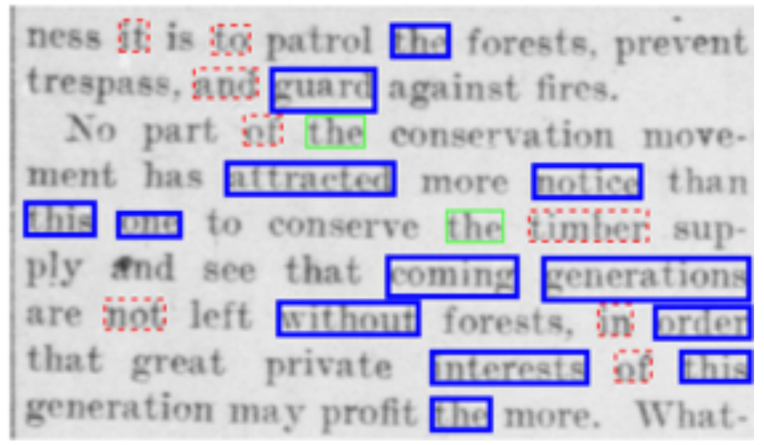
Classification Using Discriminant Functions

- SVMs
  - Design measurements, represent them as a feature vector
  - Learn the best discriminant function
- Deep NNs (simplified view)
  - Learn deep feature vectors
  - Apply SVM



Motivation for Graphical Models I: Structured Predictions

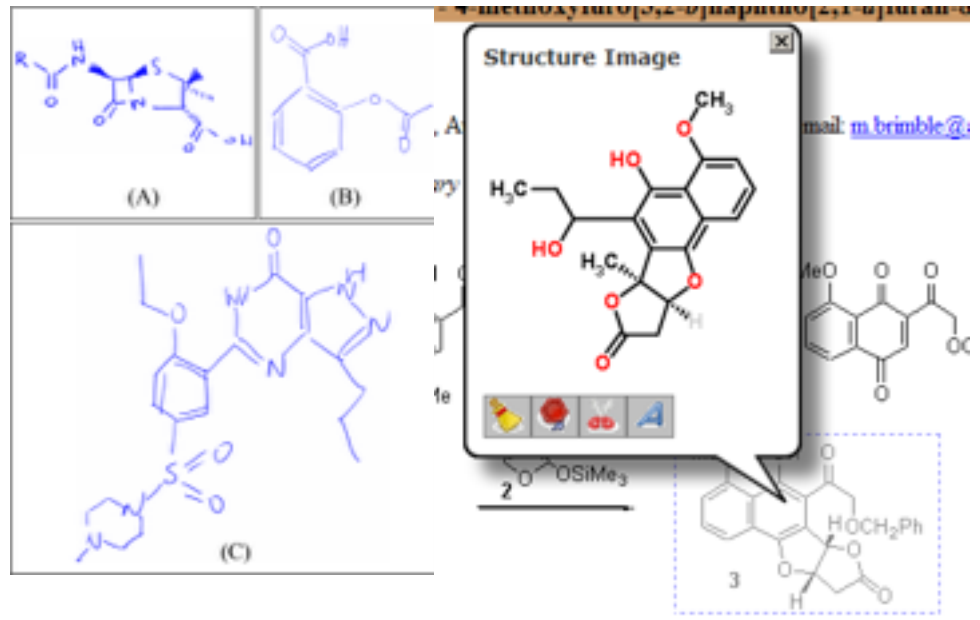
- Text Recognition



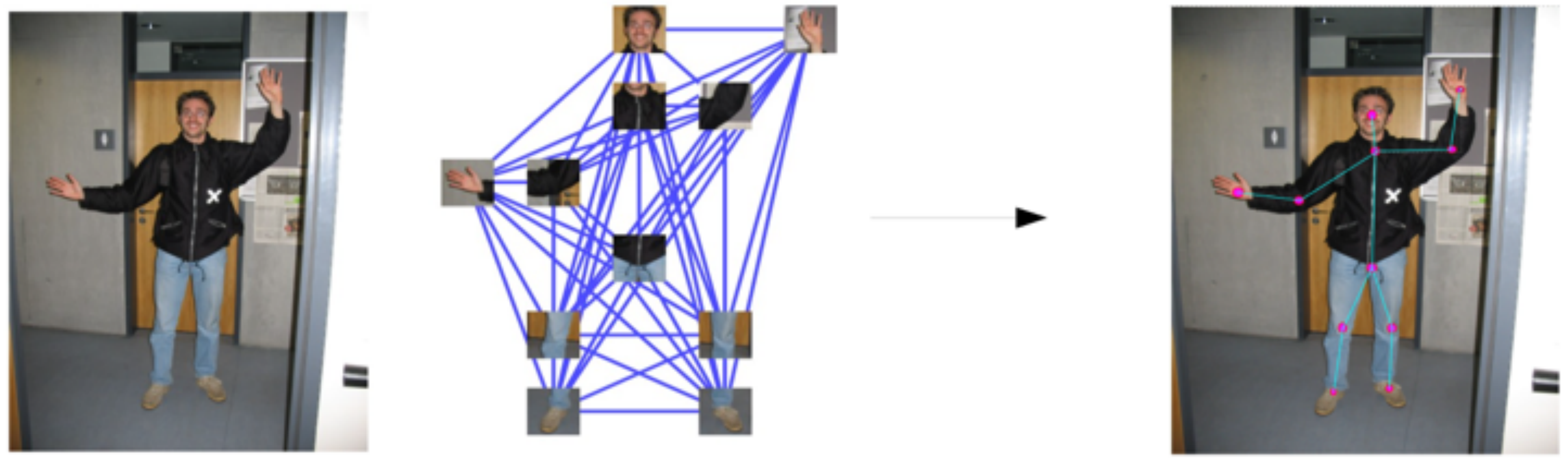
→ {space of text sentences}

- Image Segmentation

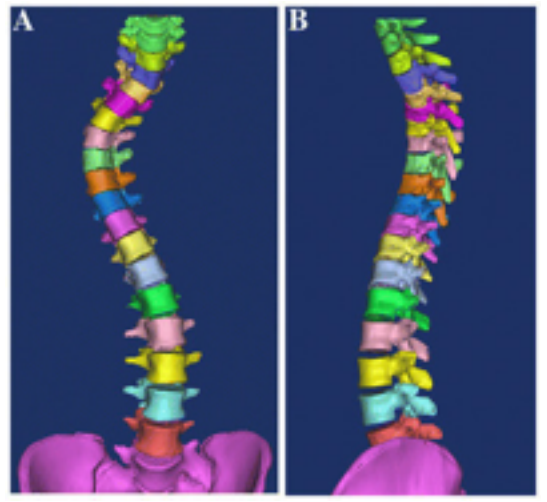
- Optical Structure Recognition

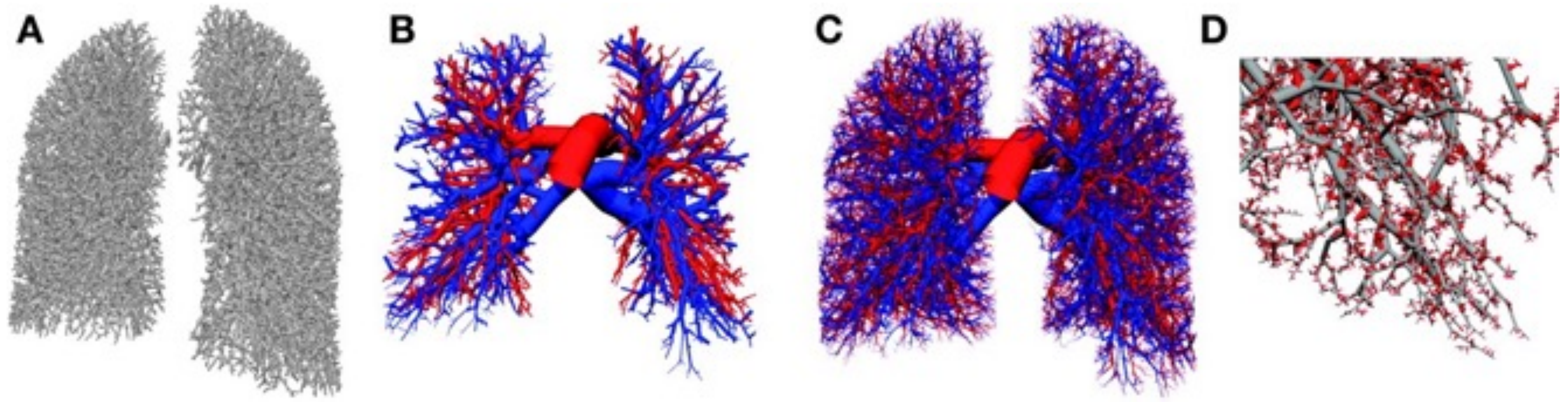


- Landmarks and Parts Detection



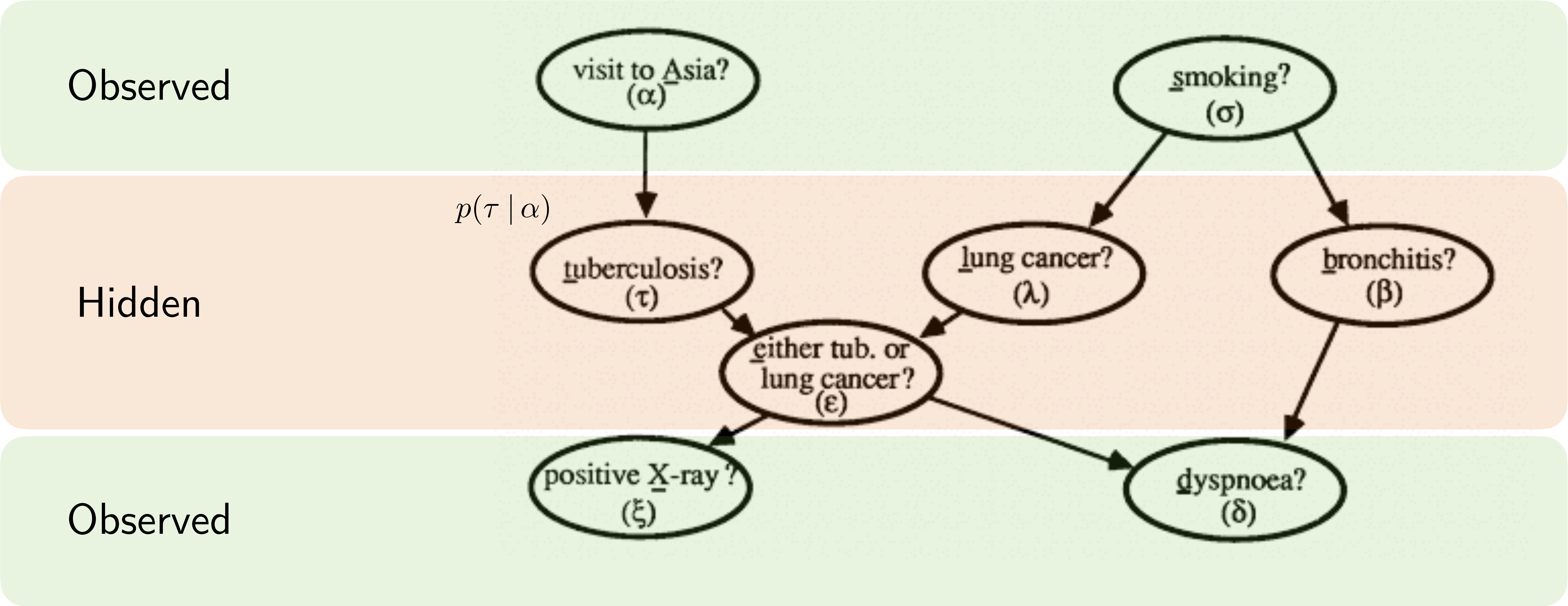
- Body Parts Segmentation





Motivation for Graphical Models II: Probabilistic Reasoning

- Example: Medical Diagnosis
  - Knowing the observed variables and conditional probabilities find the likely cause



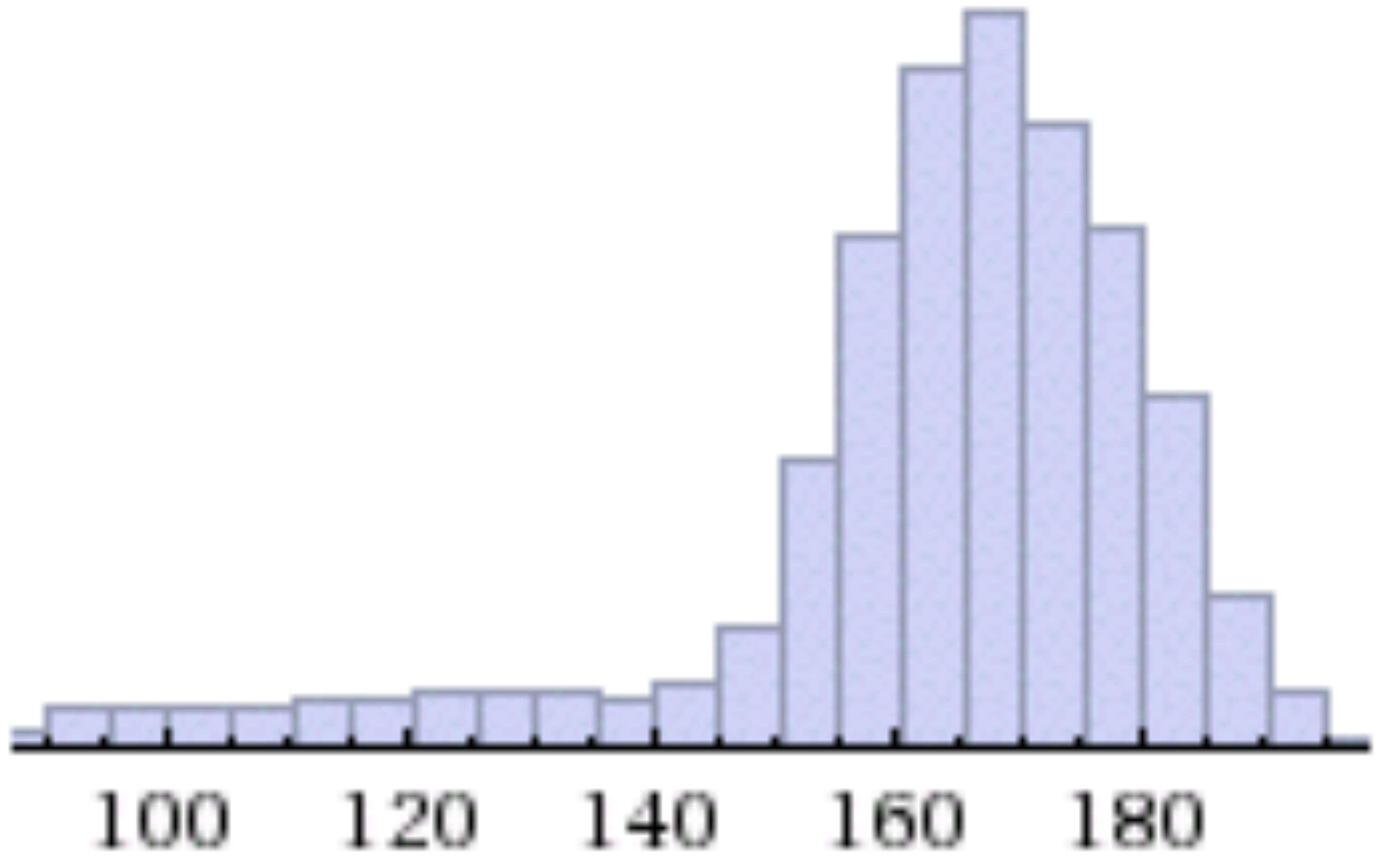
[Lauritzen and Speigelhalter 1988]

- Originally, such diagrams and methods were used by experts with pen and paper...

Statistical Models

- When do probabilities occur?
  - As a result of randomness such as thermal noise, but not only...
  - A way to represent information

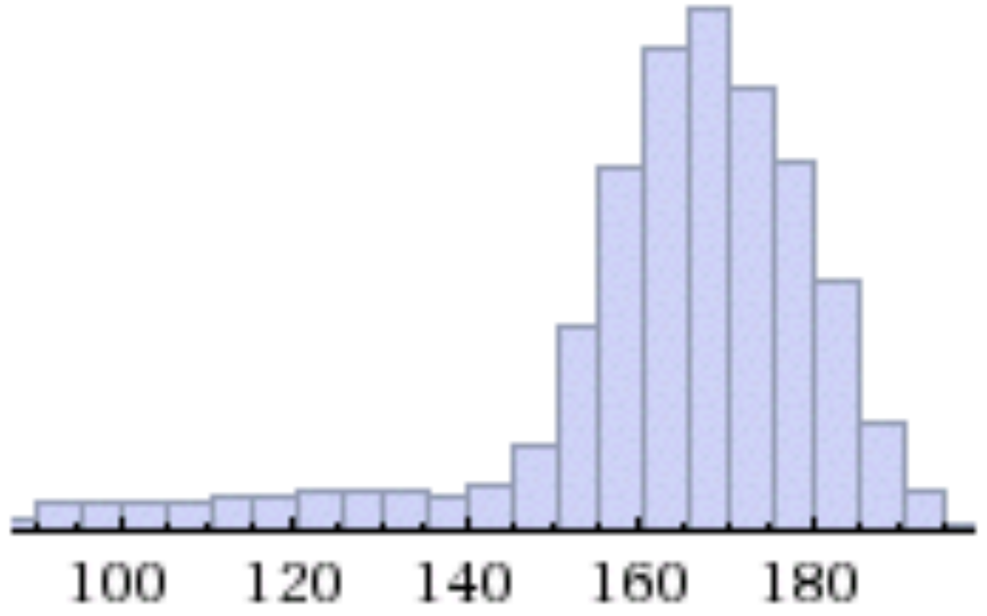
- Example 1: information about population height
  - Average human height is 162 cm (single number)
  - Human heigh is from 54 to 272 cm (interval)
  - Fraction of population of a given heigh
    - contains more information
    - more information => better solutions
    - defines a probability distribution



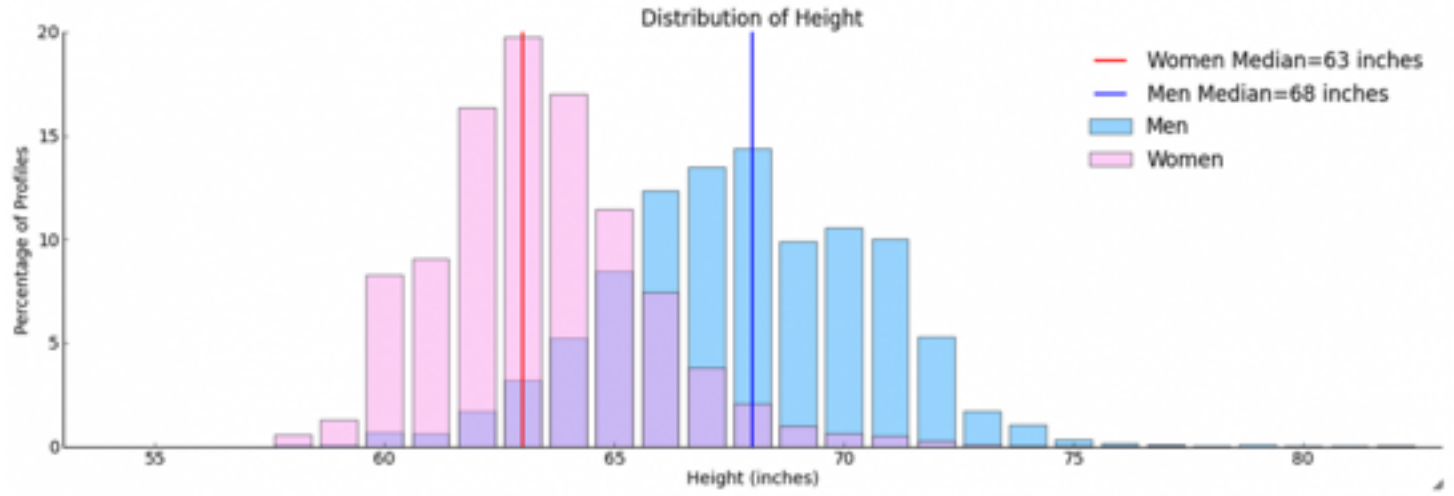
Statistical Models

- We represent the information with probabilities p(x)

random person in the world

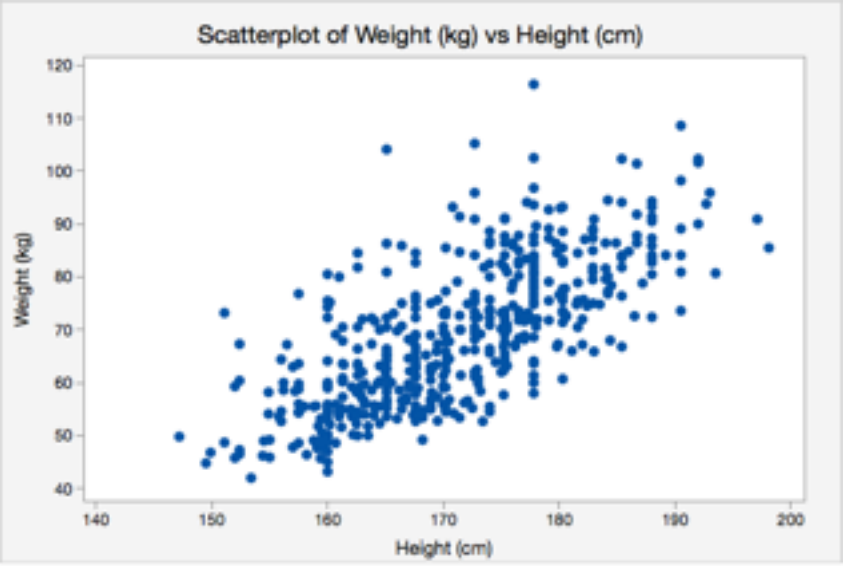


- Some new fact(s) need to be taken into account, e.g. male / female
- Refine the available information, p(x|A)



- Suppose also person weight is known

weight



hight

Refine further

Statistical Models

- Example 2: non-functional dependencies

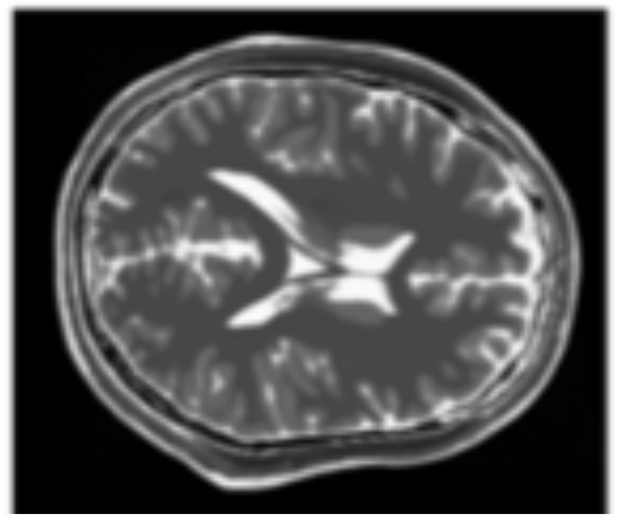
Hidden state = patient's brain

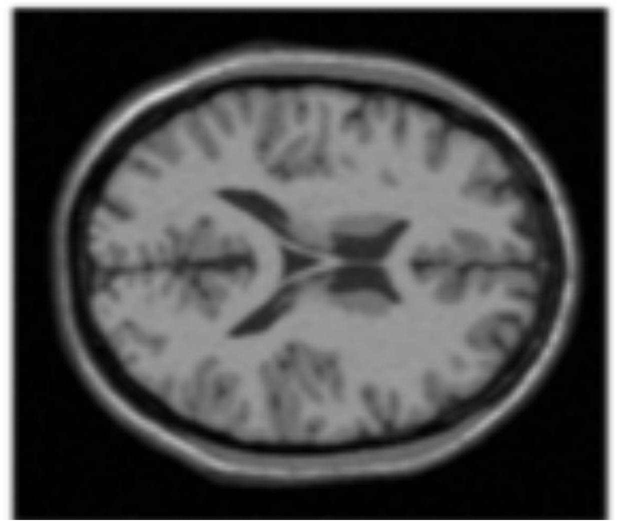
Z

MRI 1 — function

MRI 2 — function, different exposure time

X





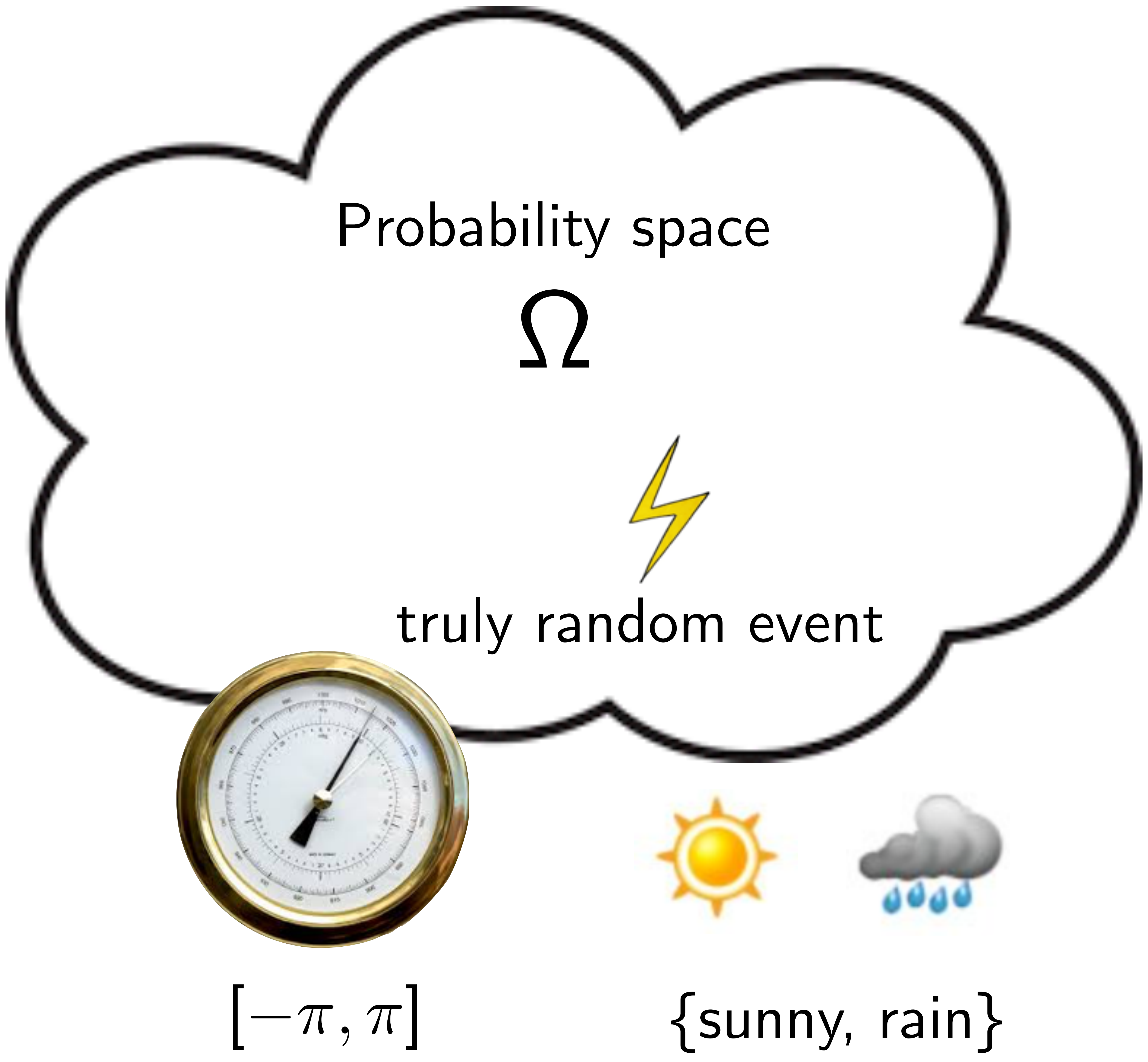
Y

Dependence of MRI 1 on MRI 2 is not a function! $Y = f(X)$?

$p(Y \mid X)$

Can be described as conditional probability distribution

Probability Cheat Sheet



$\omega \in \Omega$ – elementary event

$A \subset \Omega$ – event

$P: \Omega \to [0,1]$ – probability measure

$X: \Omega \to \mathcal{X}$ – random variable

$x \in \mathcal{X}$ – a value that r.v. X may take

$X = x$ – all elementary events that map to x:

$\{\omega \in \Omega \,|\, X(\omega) = x\}$ - an event

$P(A)$ ✓ $P(X)$ ✗ $P(X = x)$ ✓

$p_X : \mathcal{X} \to [0,1]$ – density (or p.m.f.) of X

$p_X(x)$ – density at point x

This lecture:

$P(X{=}y, Y{=}y, Z{=}z)$ is abbreviated as $P(x, y, z)$

$p_{X,Y|Z}(x, y|z)$ is abbreviated as $p(x, y|z)$ or as

$p(X{=}1, y|z)$ when ambiguous

$p(X)$ will denote $p_X(X)$ – the "whole density function"

(technically a composition: $\Omega \xrightarrow{X} \mathcal{X} \xrightarrow{p_X} [0,1]$)

Simple Classification Example

- Two classes to recognize: k in {0,1}
  - take some measurement x, for example thickness

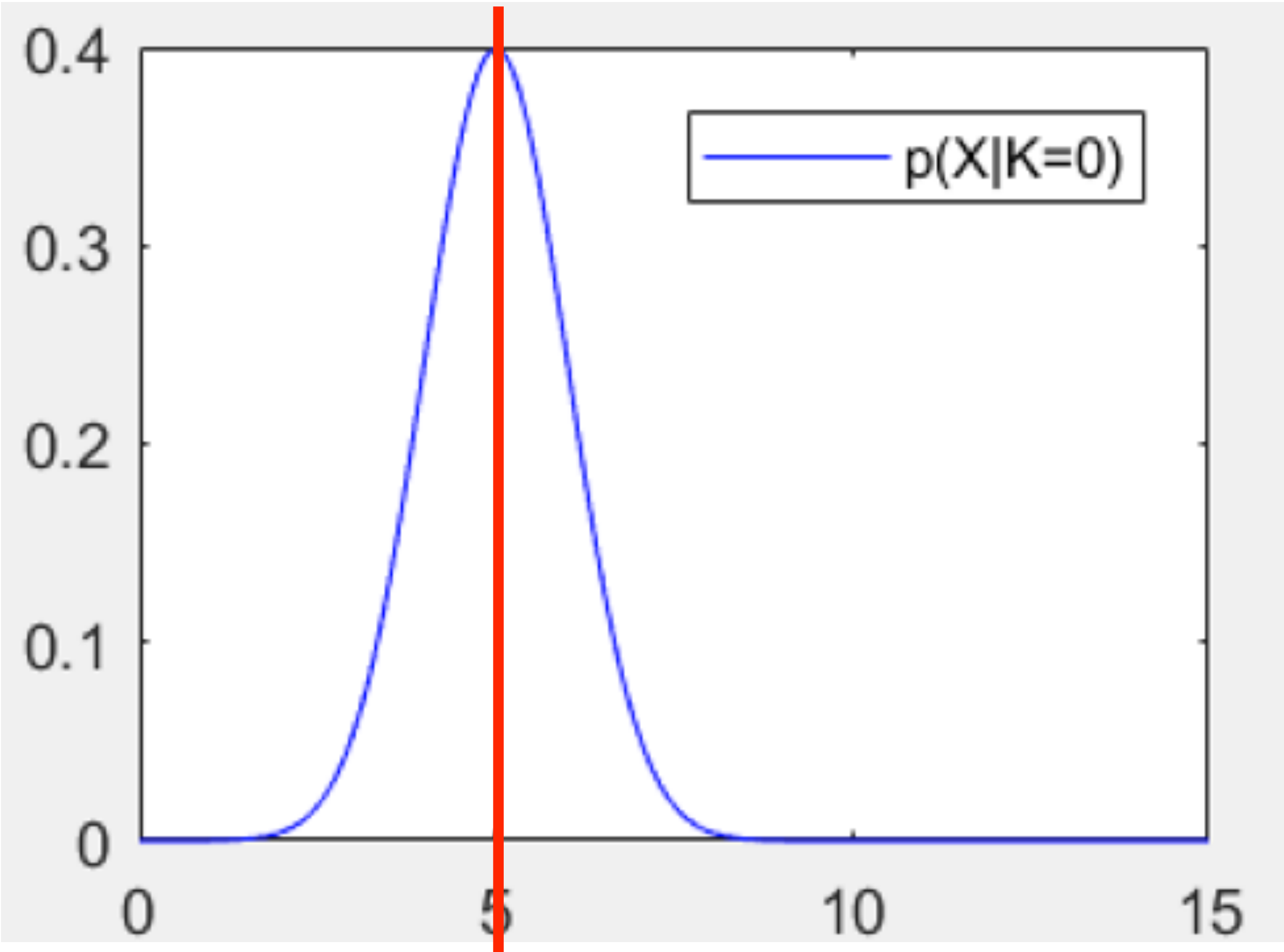
Salmon

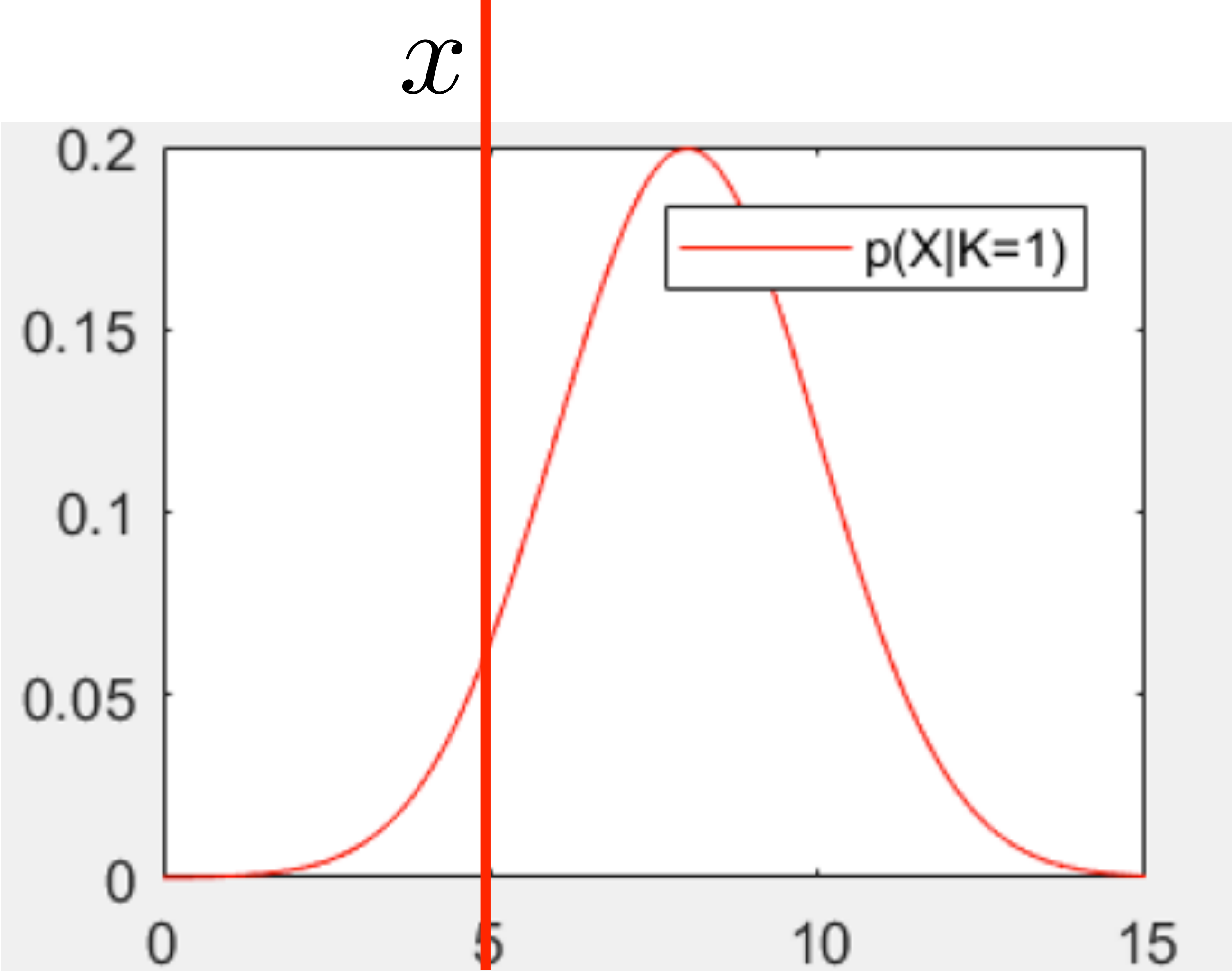
Sea Bass

- Observe X=5, which fish is it?
  - What if salmons are extremely rare in you lake?
  - Need to know probabilities p(K) of fish occurrence
  - p(K=0) = 0.15, p(K=1) = 0.85
- So what do we do with these numbers?

Known statistics p(X|k) for k=0,1

p(measurement | knowing the class)

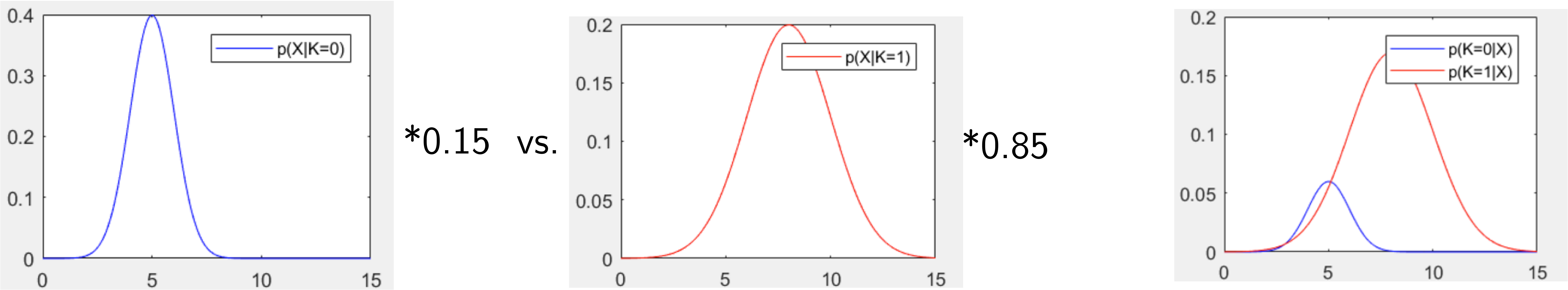




Bayes' Theorem

Theorem (Thomas Bayes, 1701–1761)

$$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}, \text{ where } A, B \text{ are events and } P(B) \neq 0$$



- For simple classification the denominator does not matter:

$$p(K{=}0 \mid x) \gtrless p(K{=}1 \mid x) \quad \Leftrightarrow \quad \frac{p(K{=}0 \mid x)}{p(K{=}1 \mid x)} \gtrless 1 \quad \Leftrightarrow \quad \frac{p(x \mid K{=}0)}{p(x \mid K{=}1)} \gtrless \frac{p(K{=}1)}{p(K{=}0)} = \theta$$

- If we have utilities (risks) or want to quantify uncertainty, need posterior probabilities:
  p(K=0|x) = 0.52, p(K=1|x) = 0.47

Parameters as Random Variables

- Experiment: flipping a coin

  $K \in \{\text{Heads}, \text{Tails}\}$

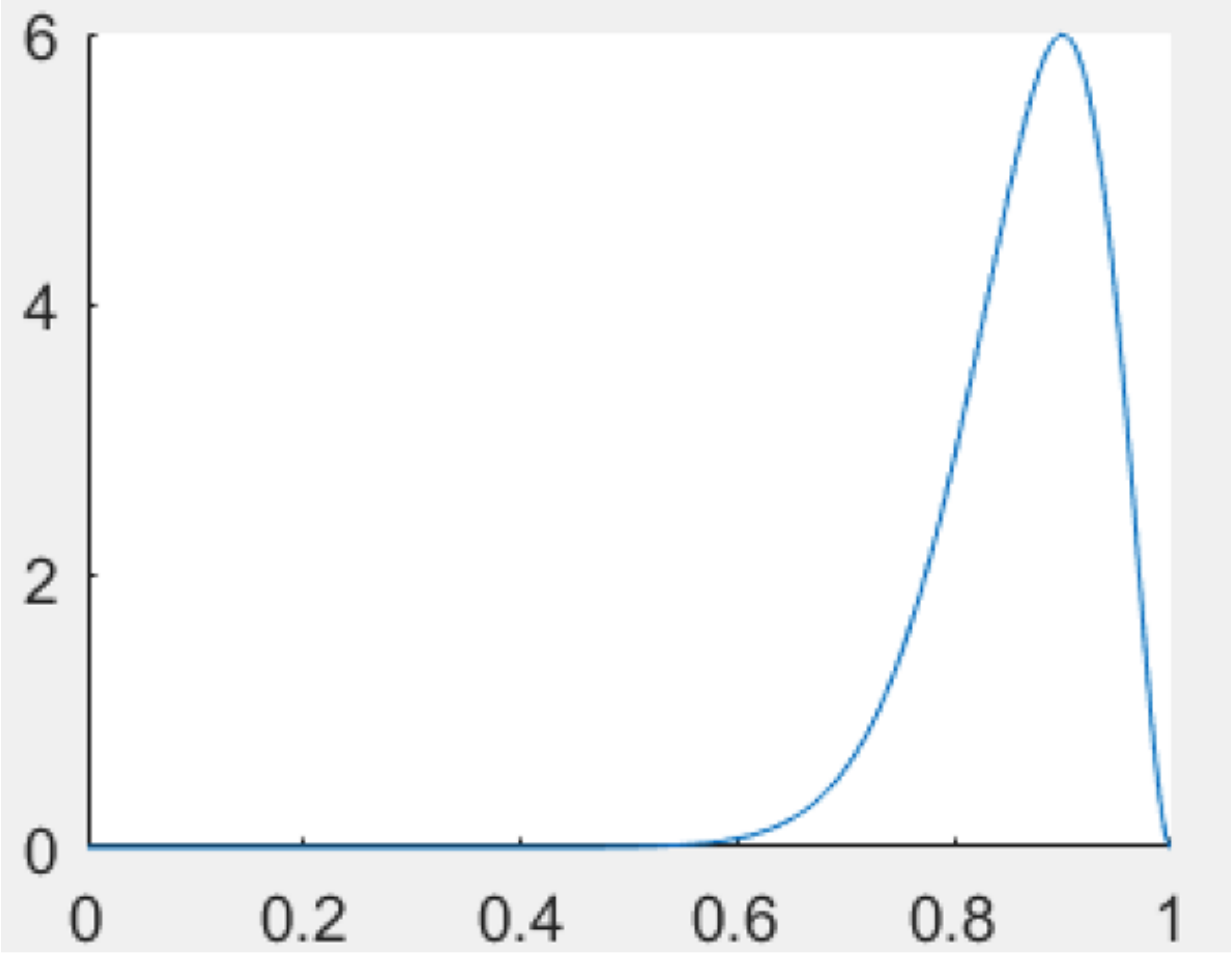
  $P(K{=}Heads) = p$

  $P(K{=}Tails) = 1 - p$

  p is unknown

- Suppose you tried 20 times and observed: 18 H and 2 T

- What you can say about p?
  - $0 < p < 1$ (strictly)
  - it is more likely that p is closer to 0.9
  - but other values of p, including 1/2 are not excluded...
- Bayes has proposed to assign probabilities to p considered as beliefs (the information that we have about p)



Bayes posterior of p (Beta distribution)

Axioms

- Recall axioms of the probability theory:

Axiom 1: $0 \leq P(A) \leq 1$, with $P(A) = 1$ if A is certain

Axiom 2: If events (A_i), $i = 1, 2, \ldots$ are pairwise incompatible (exclusive)

then $P(\bigcup_i A_i) = \sum_i P(A_i)$

Axiom 3: $P(A \cap B) = P(B \mid A) P(A)$

- Proofs exist that these rules are necessary
"if we want to assign numerical values to represent degrees of rational belief in a set of propositions" (Cox 1946).

Exercise

Axiom 1: $0 \leq P(A) \leq 1$, with $P(A) = 1$ if A is certain

Axiom 2: If events (A_i), $i = 1, 2, \dots$ are pairwise incompatible (exclusive)

then $P(\bigcup_i A_i) = \sum_i P(A_i)$

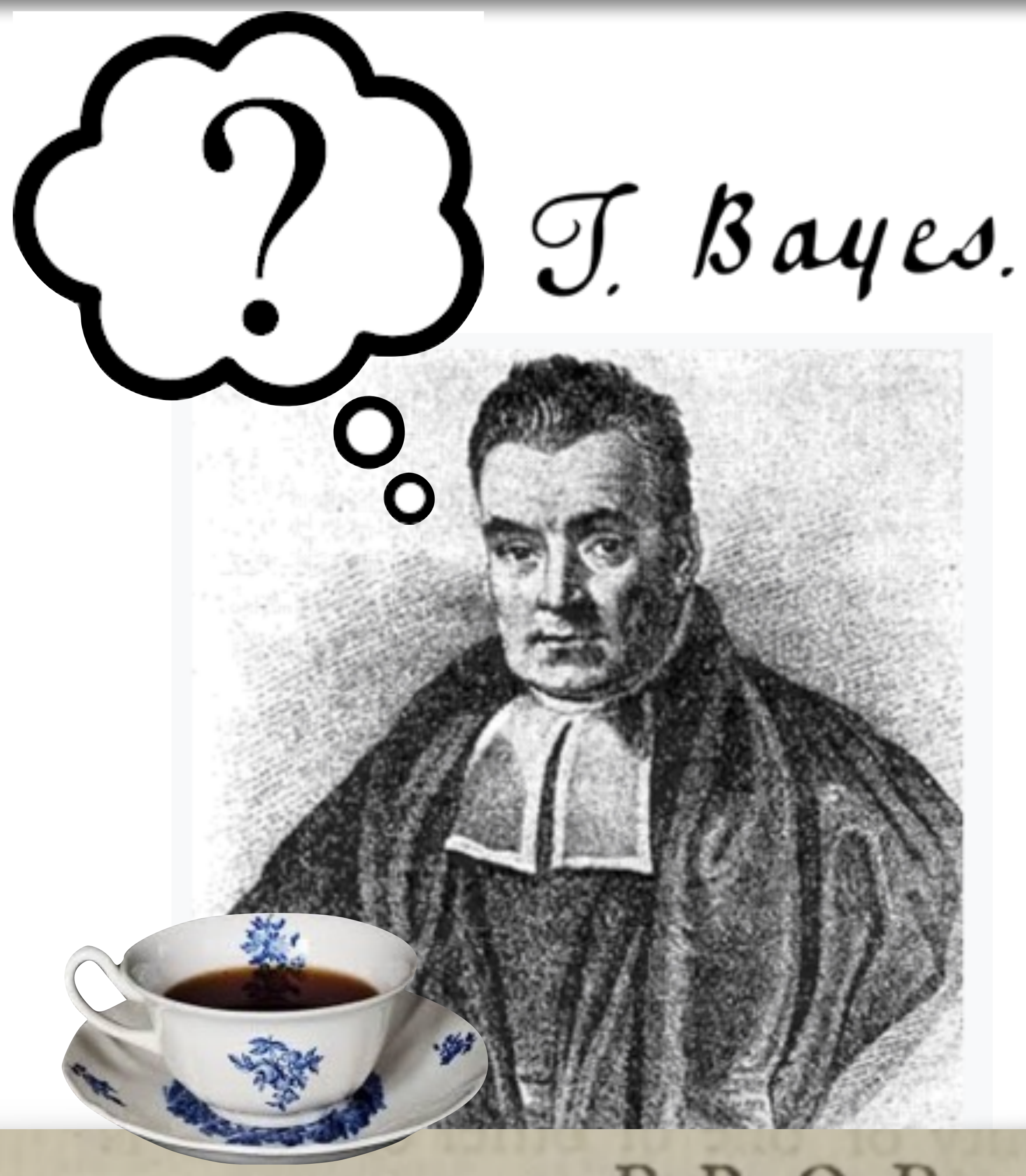
Axiom 3: $P(A \cap B) = P(B \mid A) P(A)$

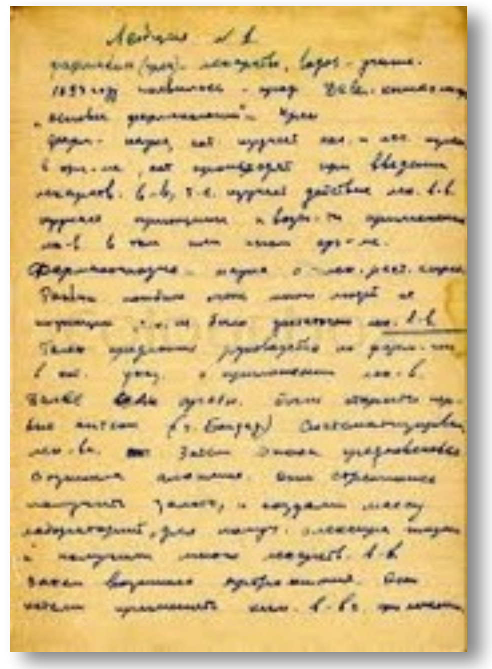
- Exercise: prove the Bayes' theorem:

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$

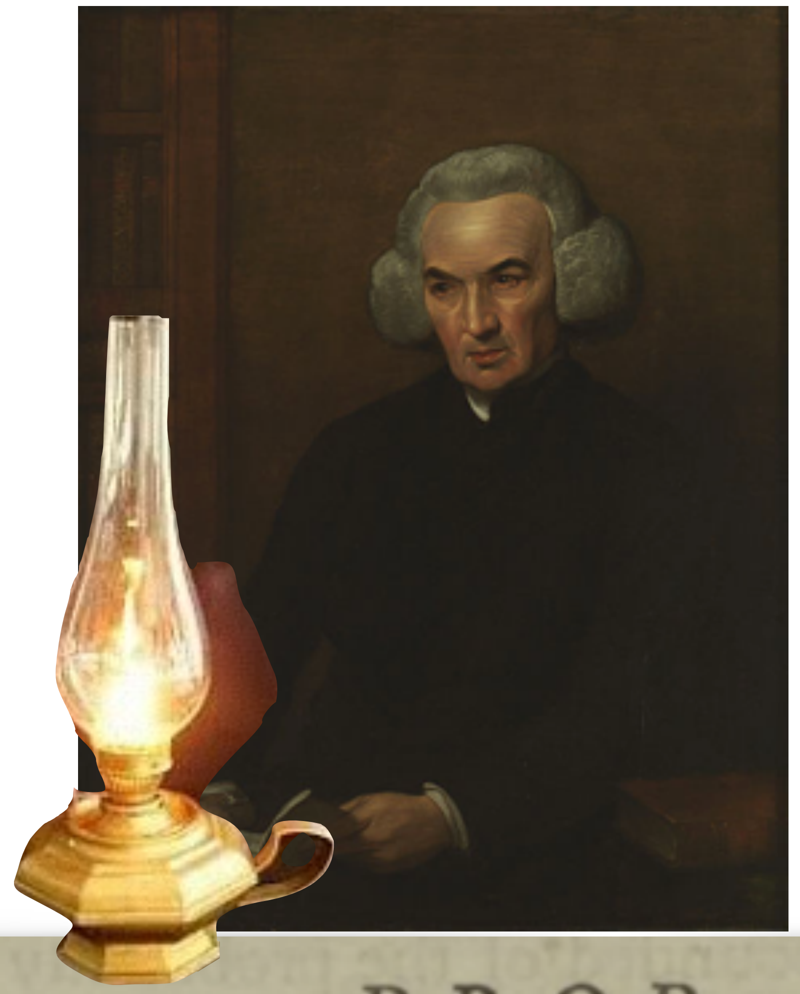
- Now prove it without axioms?

Back in 1760s…

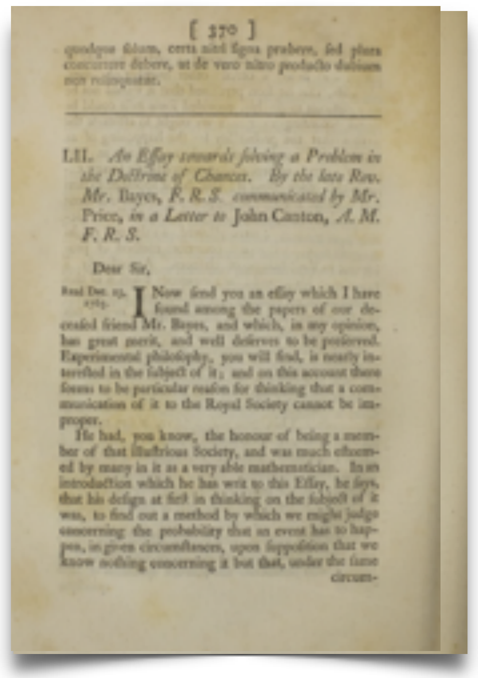




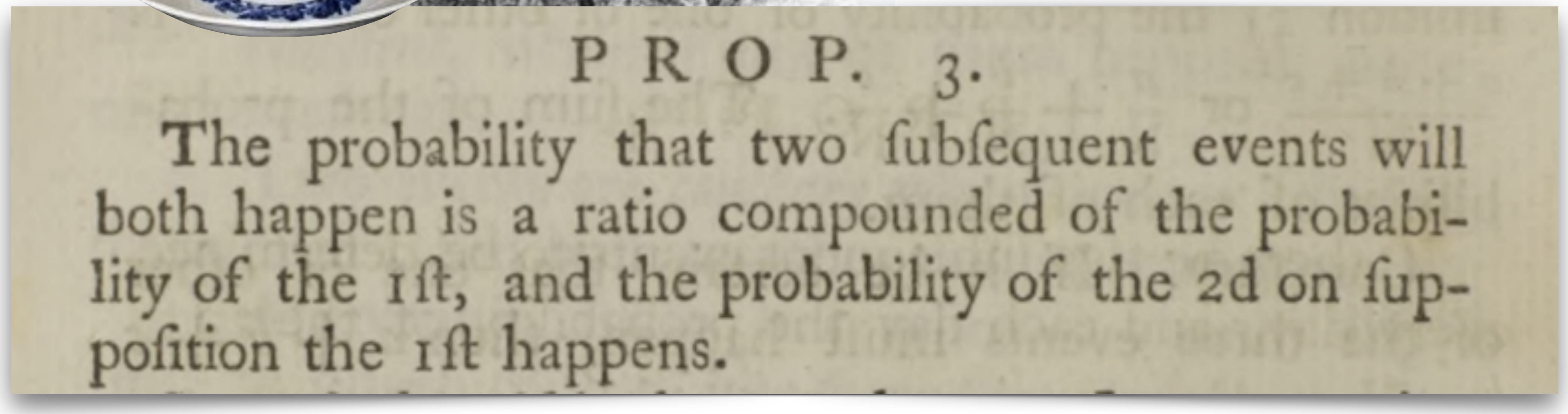
Richard Price



An Essay towards solving a Problem in the Doctrine of Chances, 1763

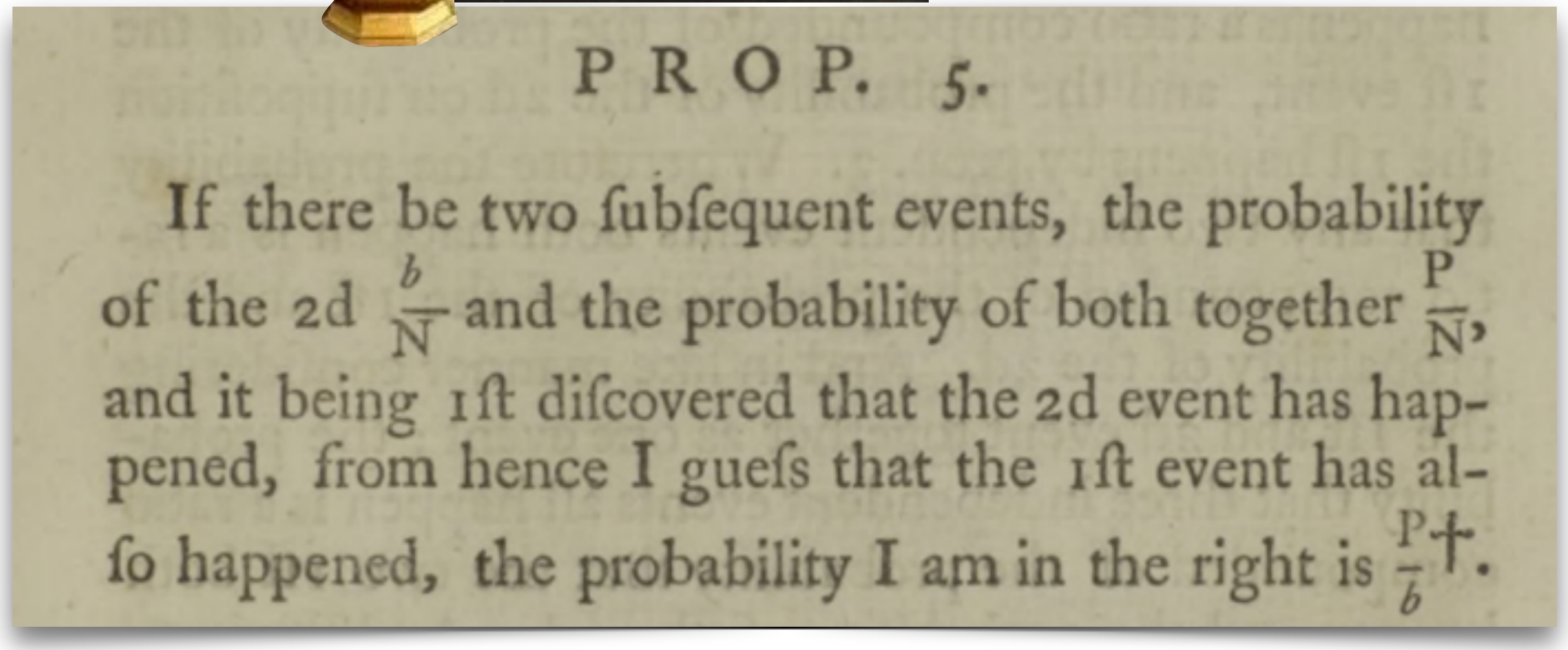


50 pages



PROP. 3.

The probability that two ſubſequent events will both happen is a ratio compounded of the probability of the 1ſt, and the probability of the 2d on ſuppoſition the 1ſt happens.



PROP. 5.

If there be two ſubſequent events, the probability of the 2d $\frac{b}{N}$ and the probability of both together $\frac{P}{N}$, and it being 1ſt diſcovered that the 2d event has happened, from hence I gueſs that the 1ſt event has alſo happened, the probability I am in the right is $\frac{P}{b}$†.

"…in the constitution of things fixt laws according to which things happen…
…and thus to confirm the argument taken from final causes for the existence of the Deity"

Richard Price

…Bicycle invented about 50 years later

Exercise

- Suppose we have a test for cancer with the following statistics:
  - The test was positive in 98% of cases when subjects had cancer
  - The test was negative in 97% of cases when subjects did not had cancer
- Suppose that 0.1% of the entire population has this disease

- A patient takes a test. Compute
  - The probability that a person who test positive has this disease?
  - The probability that a person who test negative does not have this disease?

Variables: $C \in \{y, n\}$, $T \in \{+, -\}$

Probabilistic Models

- Observed variables:
  $X_1, X_2, \ldots, X_n$; represented by vector $X = (X_i \mid i = 1, \ldots n)$; Event $X = x$ is denoted as x
- Hidden variables:
  $K_1, K_2, \ldots, K_m$; represented by vector $K = (K_i \mid i = 1, \ldots m)$

(The naming / roles may differ depending on the context)

Definition (Model)

A probabilistic *model* is the joint probability distribution over a set of random variables. We assume the density $p(X, K)$.

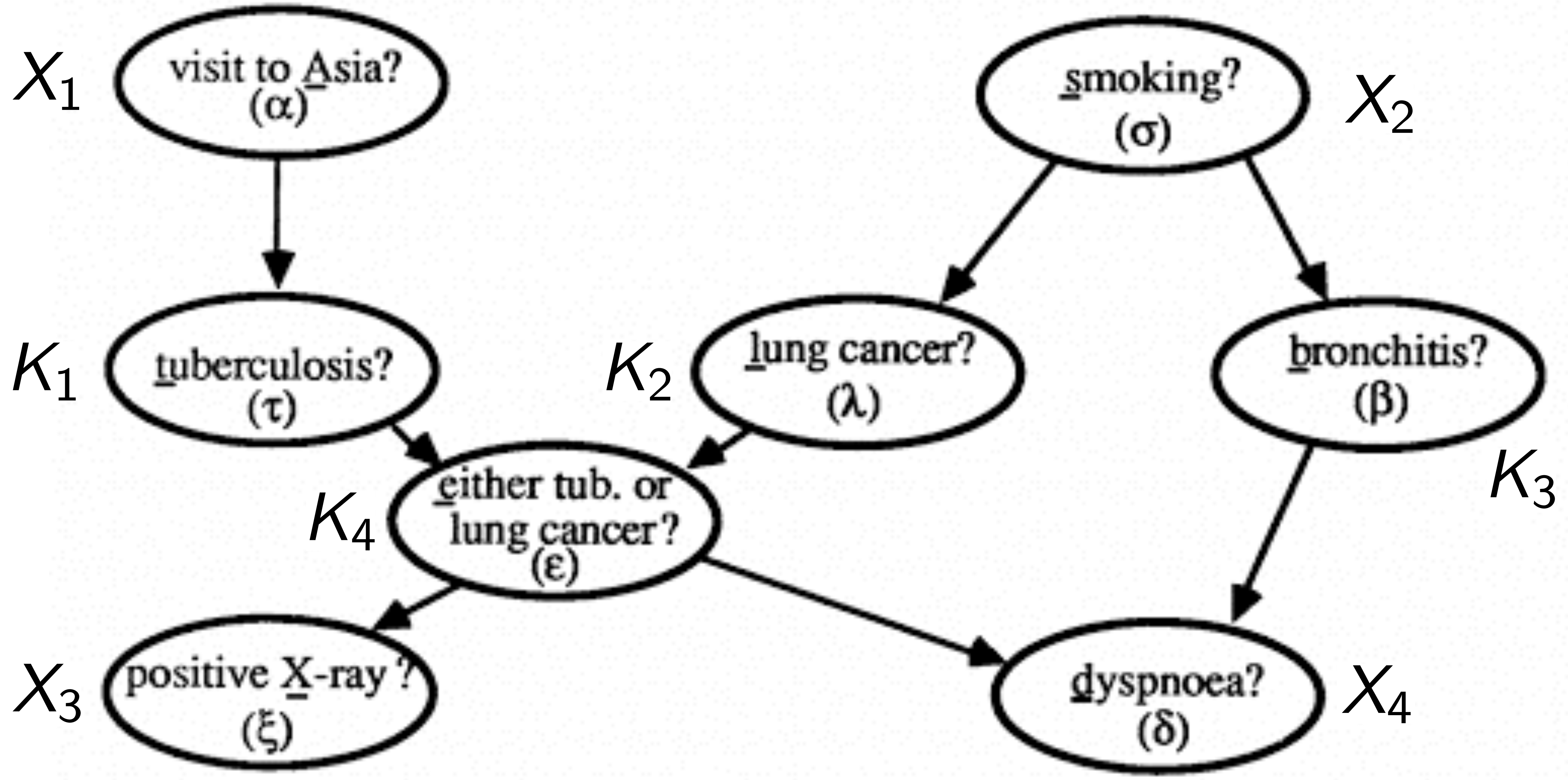
- Models describe how a part of the world works. Are always approximations or simplifications.

- Posterior inference task: Given $X = x$, compute $p(K \mid x)$
- Maximum a posterior task (recognition): $\operatorname{argmax}_K p(K \mid x)$
- Statistical decision making: $\operatorname{argmin}_d \sum_k \mathrm{Risk}(d, k) p(k \mid x)$

Example of Tasks

Model: $p(X, K)$

- Observation: x = (yes, yes, yes, no)
- Tasks:
  - Posterior: p(K3=yes | x)
    (belief in bronchitis)
  - MAP: most likely explanation:
    $$\max_k p(k \mid x)$$
  - Decision making:
    {do nothing, heal 1,2,3, new analysis}



- More general queries:
  - Suppose result of X-ray is not yet available,
    - what in the belief in bronchitis versus more serious problems?
    - what is the prediction for X-ray?
    - how much the belief in bronchitis depends on X-ray?

The Promise and The Catch

- Promises of probabilistic models:
  - A sound formulation for a system that can answer different kinds of queries:
    - recognition (likely cause)
    - handling missing data
    - prediction (likely symptoms)
    - "what if" queries
  - semi-supervised learning (parameters are random variables)
  - ...

- Obstacles:
  - Model representation
  - The problems that we can formulate mathematically are not necessarily solvable

- Looks like the right way to go, a major part in AI research
- With some hard work we get subclasses and approximations that are useful

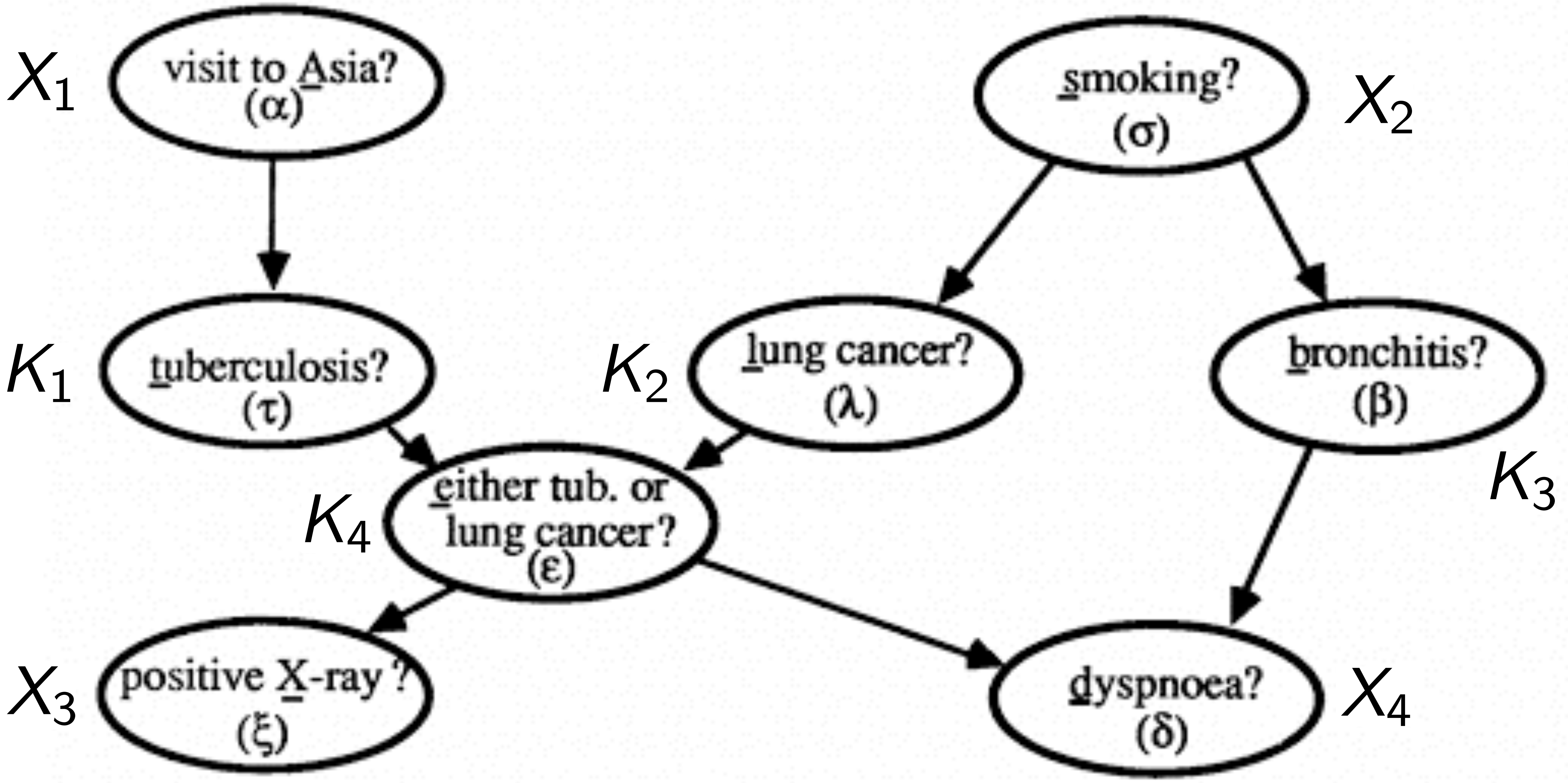
Model Complexity

- Probabilistic models are useful

- To represent the model in the example we need probabilities for all combinations of 8 Boolean variables:

  $p(X_1, X_2, X_3, X_4, K_1, K_2, K_3, K_4)$

- 2^8 =256 numbers
- Becomes quickly intractable
  - to store / to learn



Trivial observation: If all variables are independent, the distribution factors as:

$$p(X, K) = p(X_1)p(X_2)p(X_3)p(X_4)p(K_1)p(K_2)p(K_3)p(K_4)$$

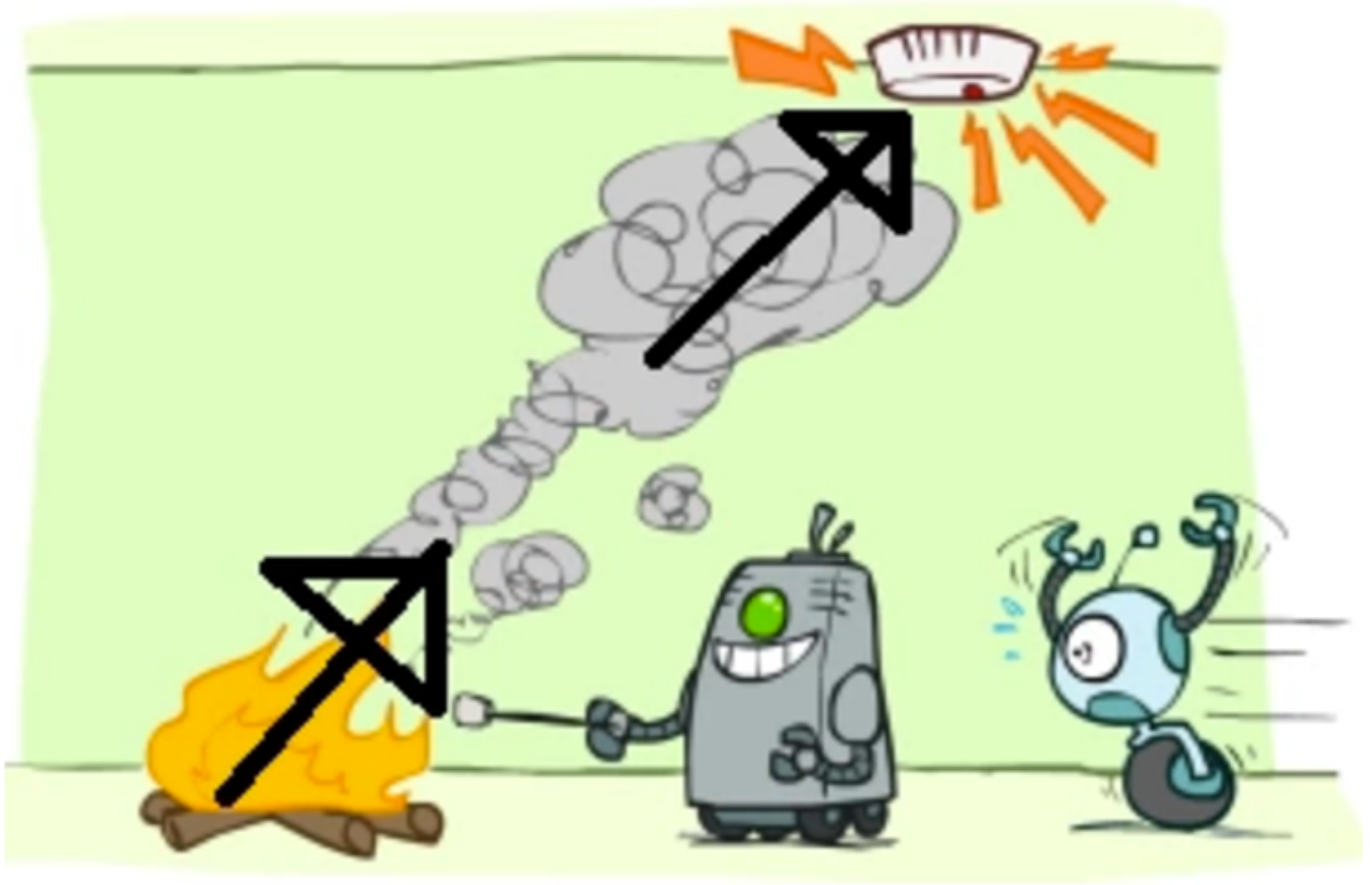
Can be described by just 8 parameters. Something in between?

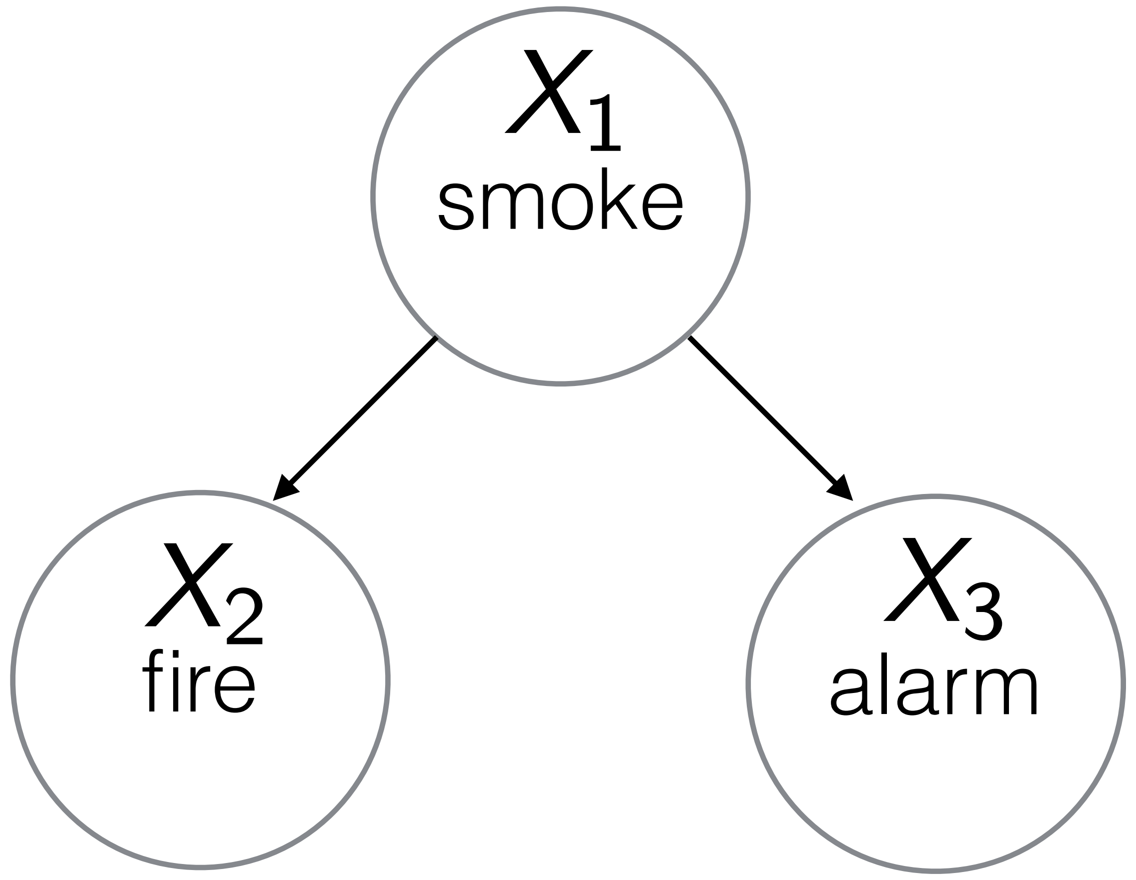
Conditional Independence

- Example: smoke, fire, alarm
  - all 3 correlated, but
  - given smoke => fire and alarm are independent

  X and Y with density $p(X, Y)$ are independent
  iff $p(x, y) = p(x)p(y)$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$

  - Conveniently represented with a graph diagram



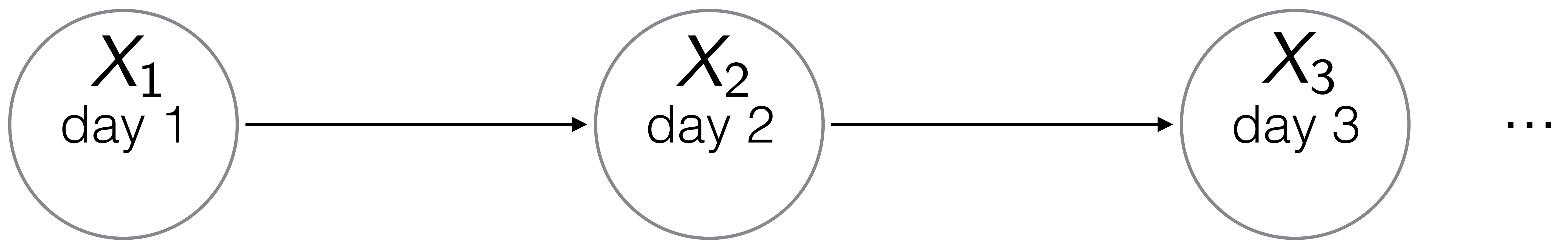


$$p(X_2, X_3 | X_1) = p(X_2 \mid X_1)p(X_3|X_1)$$

- Factorization: $p(X_1, X_2, X_3) = p(X_2, X_3 \mid X_1)p(X_1) = p(X_2 \mid X_1)p(X_3|X_1)p(X_1)$
- A directed graphical model (Bayes Network)

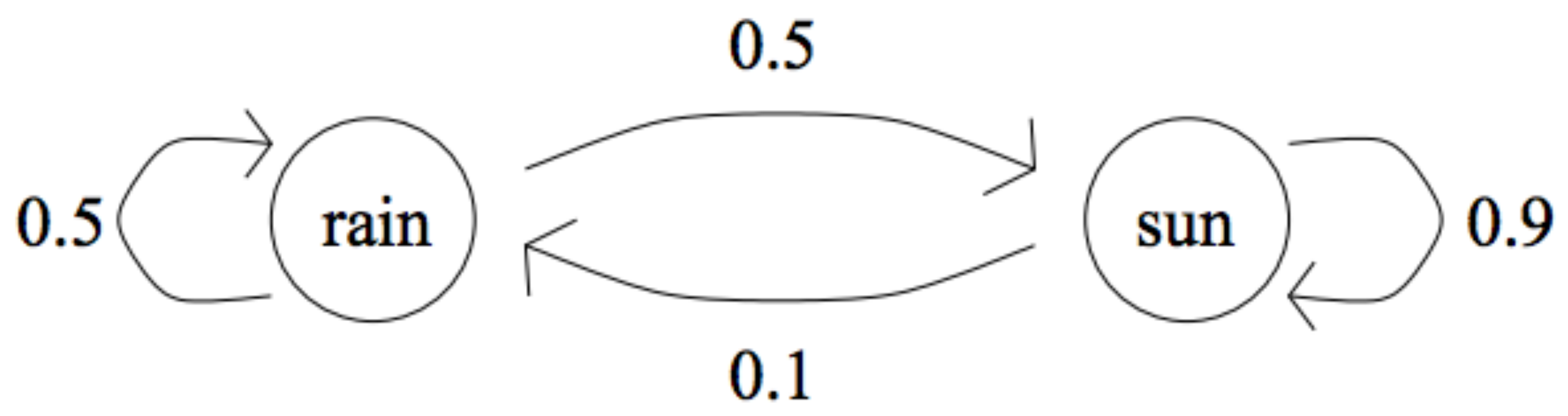
Markov Chain

- Example:
  - X_i — weather state on day i
  - Simplifying assumption: the weather on day i depends only on the state on day i-1, but not i-2, ...



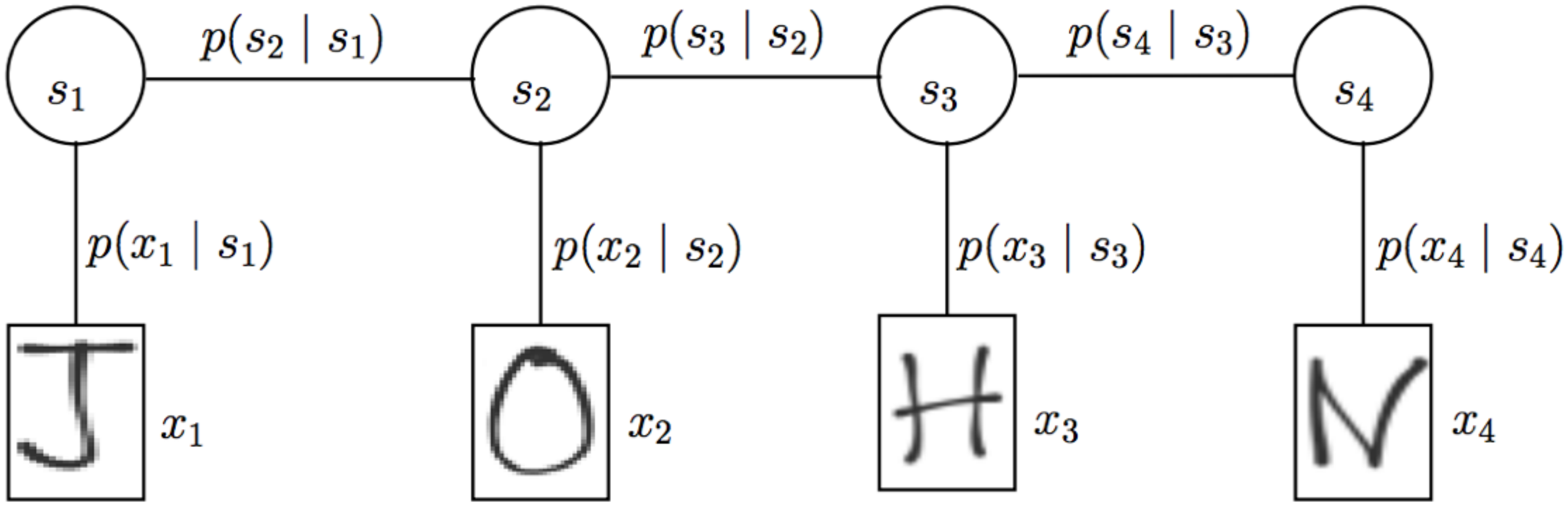
- Factorization: $p(X_1, X_2, X_3, \dots) = p(X_1)p(X_2 \mid X_1)p(X_3|X_2)\dots$

State transition diagram



Hidden Markov Model

- Example:
  - S_i — letter in a sequence (hidden)
  - X_i — observed images



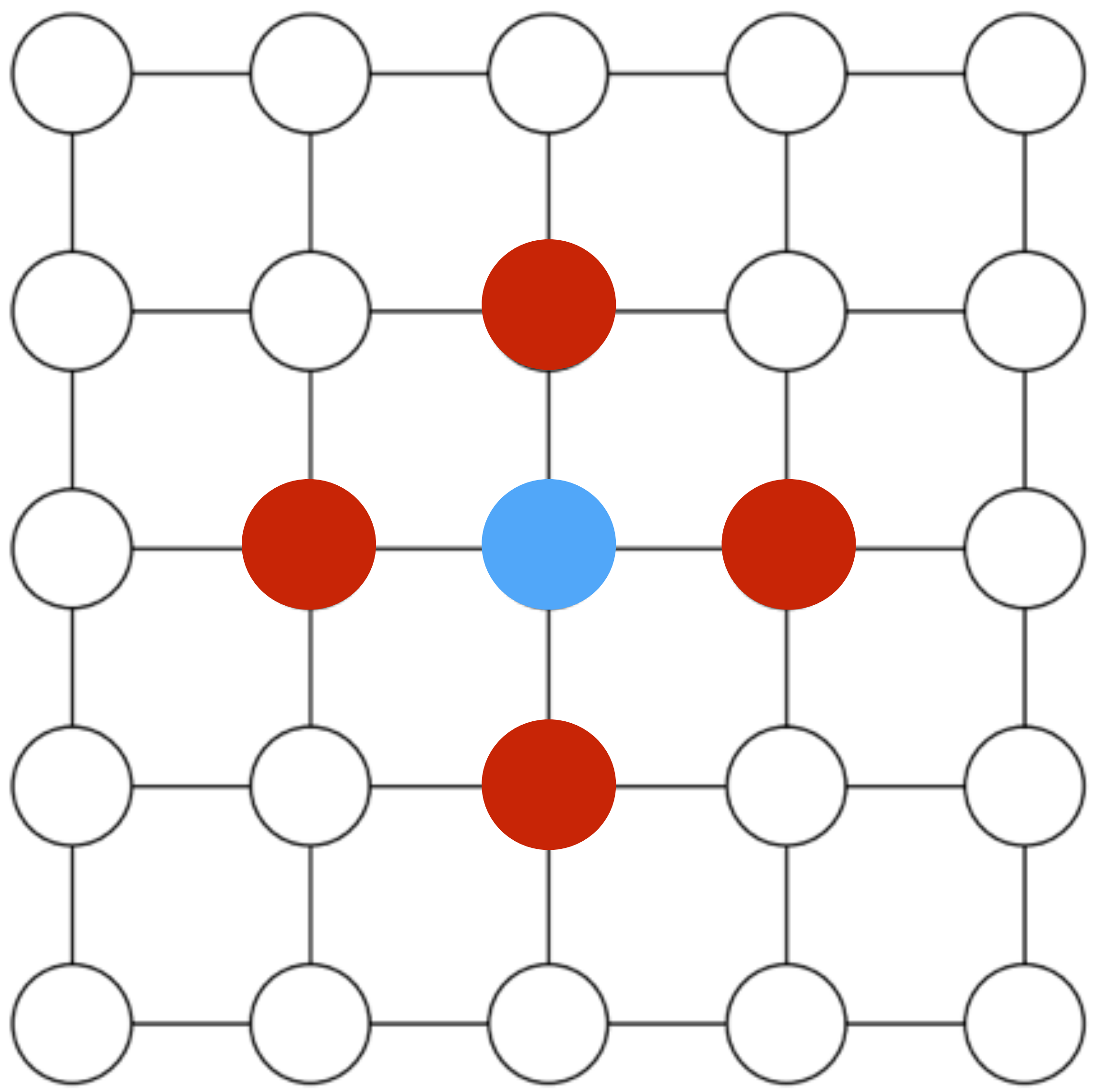
- Factorization: $$p(X, S) = p(S_1) \prod_{i=2}^{n} p(S_i \mid S_{i-1}) \prod_{i=1}^{n} p(X_i \mid S_i)$$

In Images

- A region is independent of the rest given some neighborhood

Markov Random Field

- Example: 2D spin glass:
  - X_i — spin orientation {-1,1}
  - Neighboring states "like" to be the same



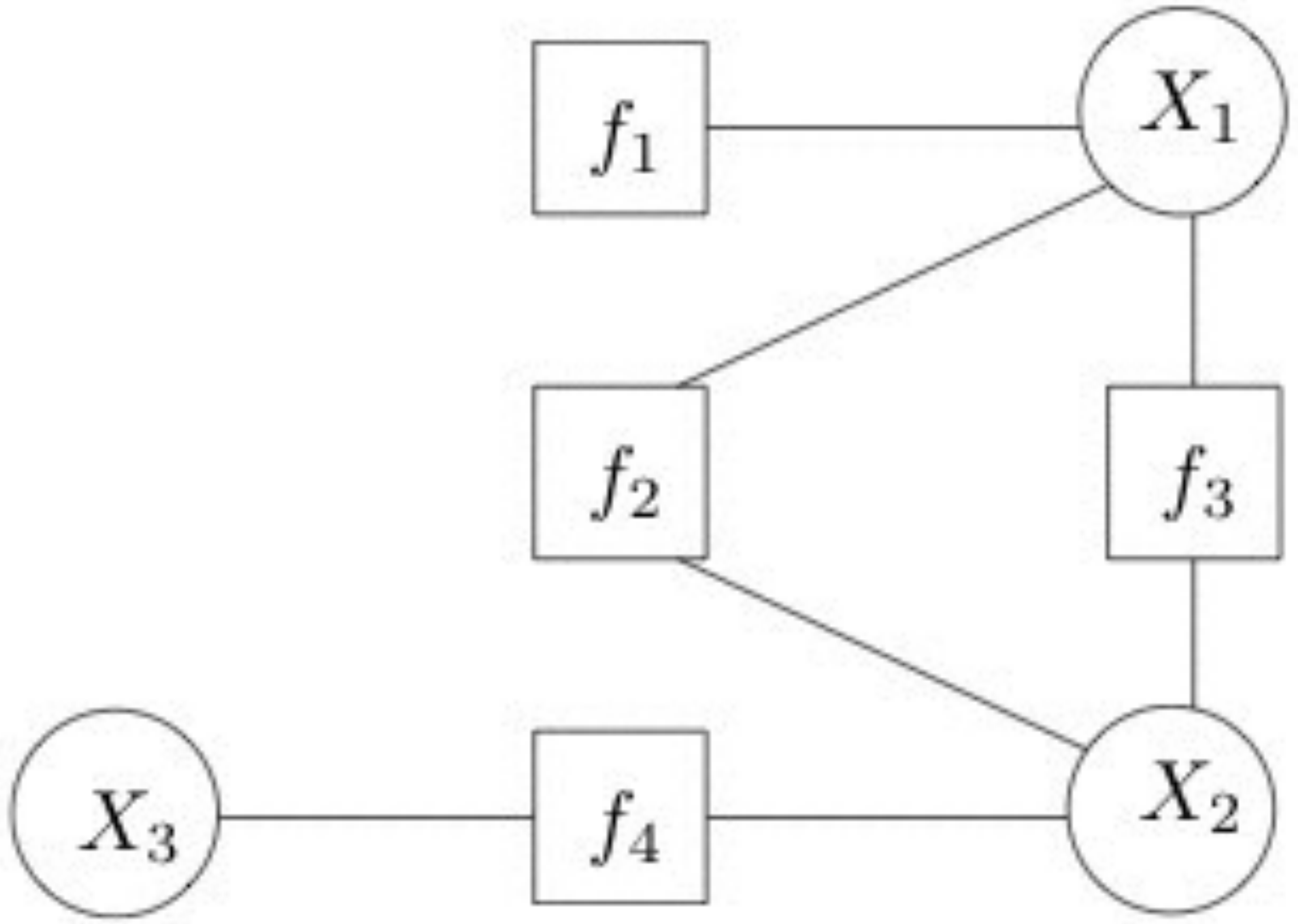
- Local Markov Property w.r.t. G:
  - Given neighbors of X_i, it is independent of the rest.
- Pairwise Markov Property w.r.t. G:
  - Absent edge (i, j) iff X_i and X_j are conditionally independent given the rest.
- Factorization: $p(x) = \prod_{c \in \mathcal{C}(G)} g_c(x_c)$ (over cliques of G, more on this later)

Factor Graphs

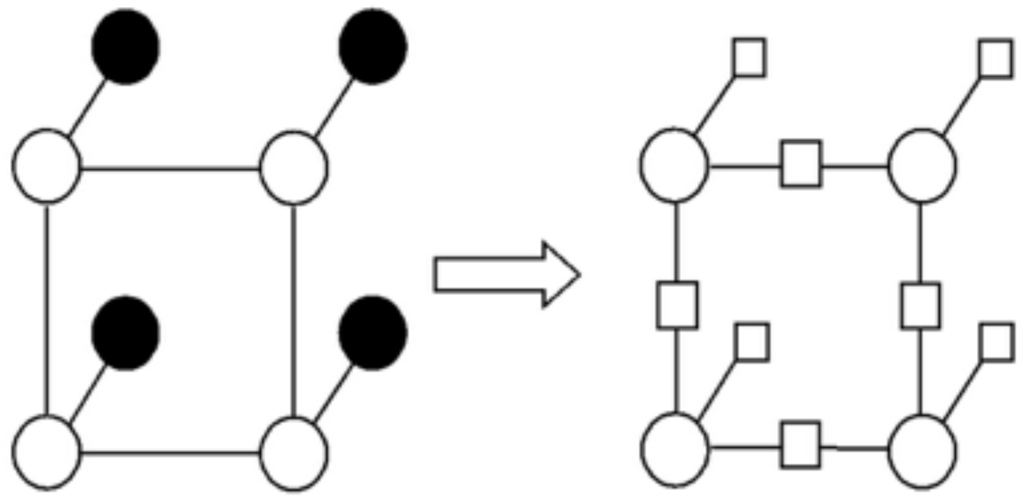
- Factorization is another constructive way to define joint probability distribution than conditional independence

$$p(X) = \frac{1}{Z} f_1(X_1) f_2(X_1, X_2) f_3(X_1, X_2) f_4(X_2, X_3)$$

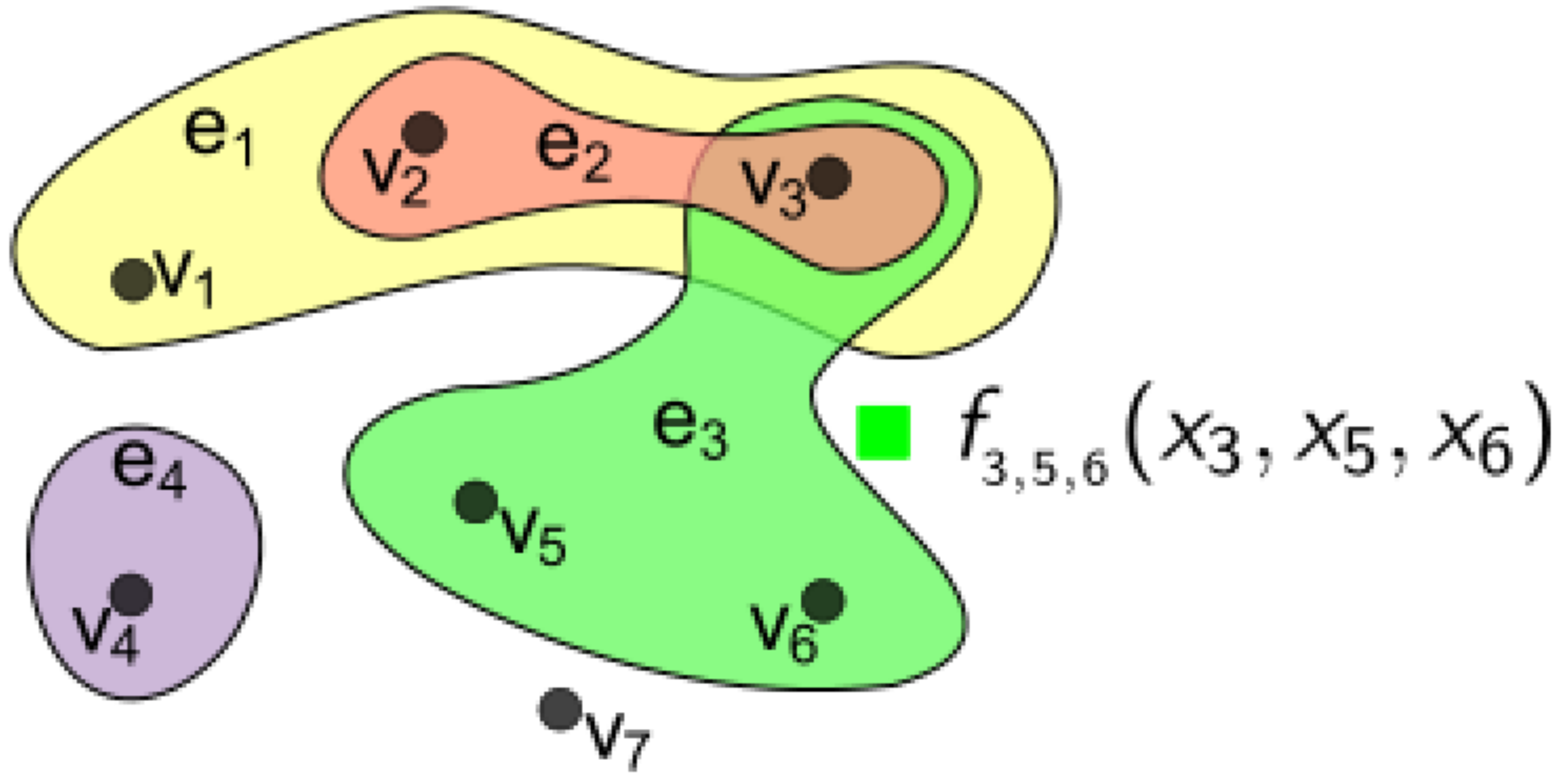
Z is the normalization factor, such that $\sum_X p(X) = 1$



- It is more general
- Inference algorithms often work directly with the factorization
- But:
  - more difficult to learn
    (c.f. conditional probabilities we could measure directly from the data)



Factor Graphs



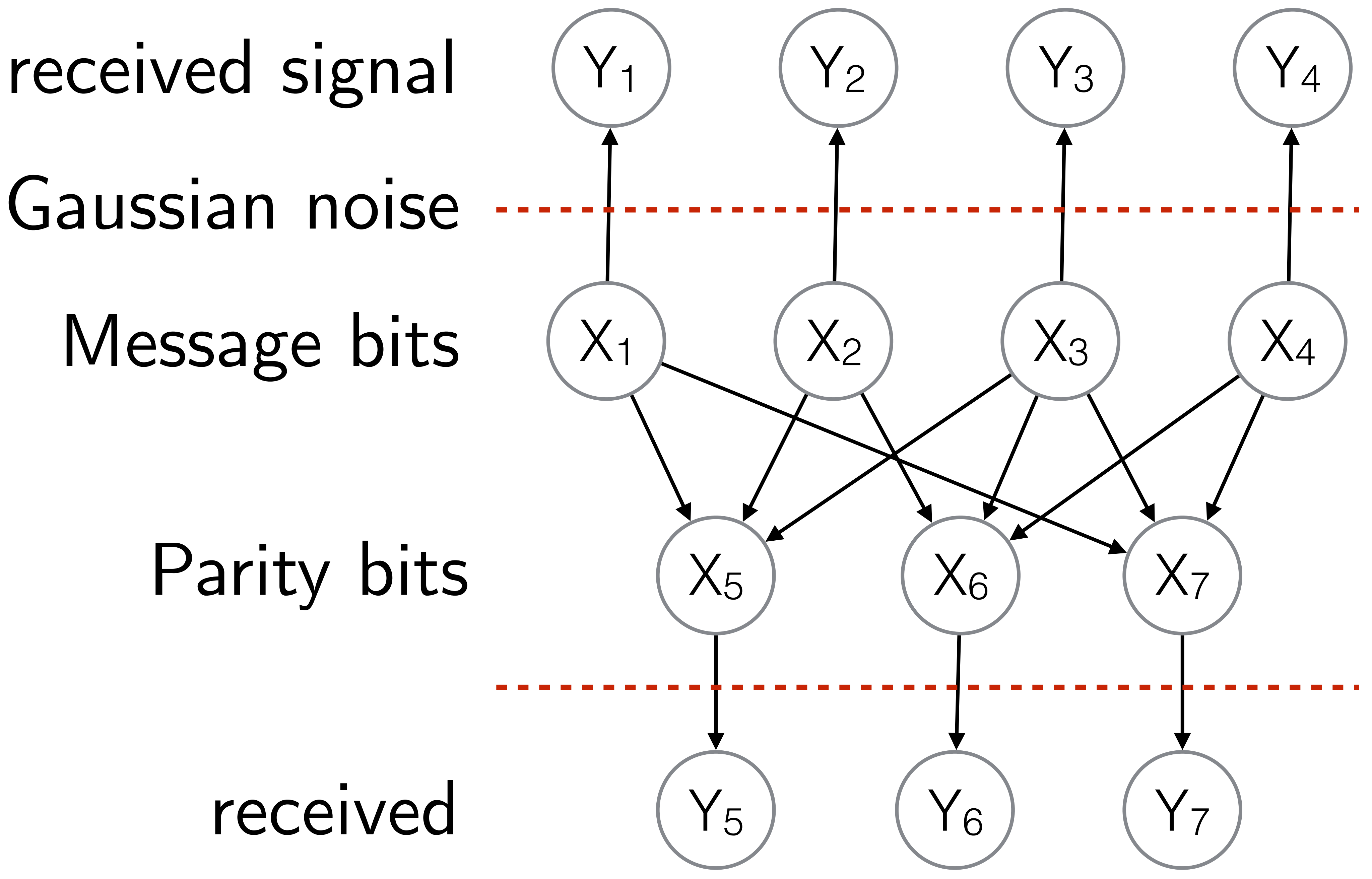
Low Density Parity Check Codes

- Coding
  - Sending N bits over a noisy channel to decode n bits
  - Shannon limit: codes exist with $n/N <$ channel capacity for arbitrary small error rate

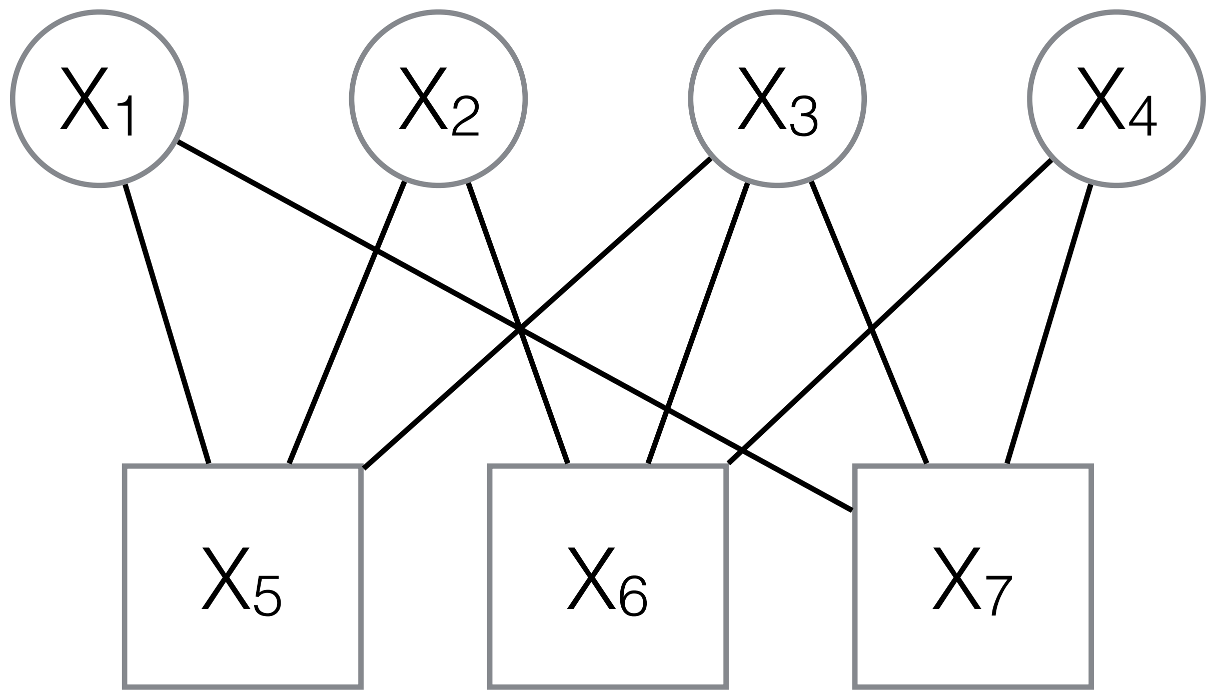
- LDPCs: proposed by Robert Gallager in 1962
- Good decoding algorithms found in 90's
- Appeared to be instances of Belief Propagation
- Motivated a lots of research on BP

- Turbo Codes and LDPCs
  - 3G and 4G mobile standards
  - digital video broadcasting
  - satellite communication systems
  - ...

  - Current codes coming closer and closer to Shannon limit



Reduced factor graph



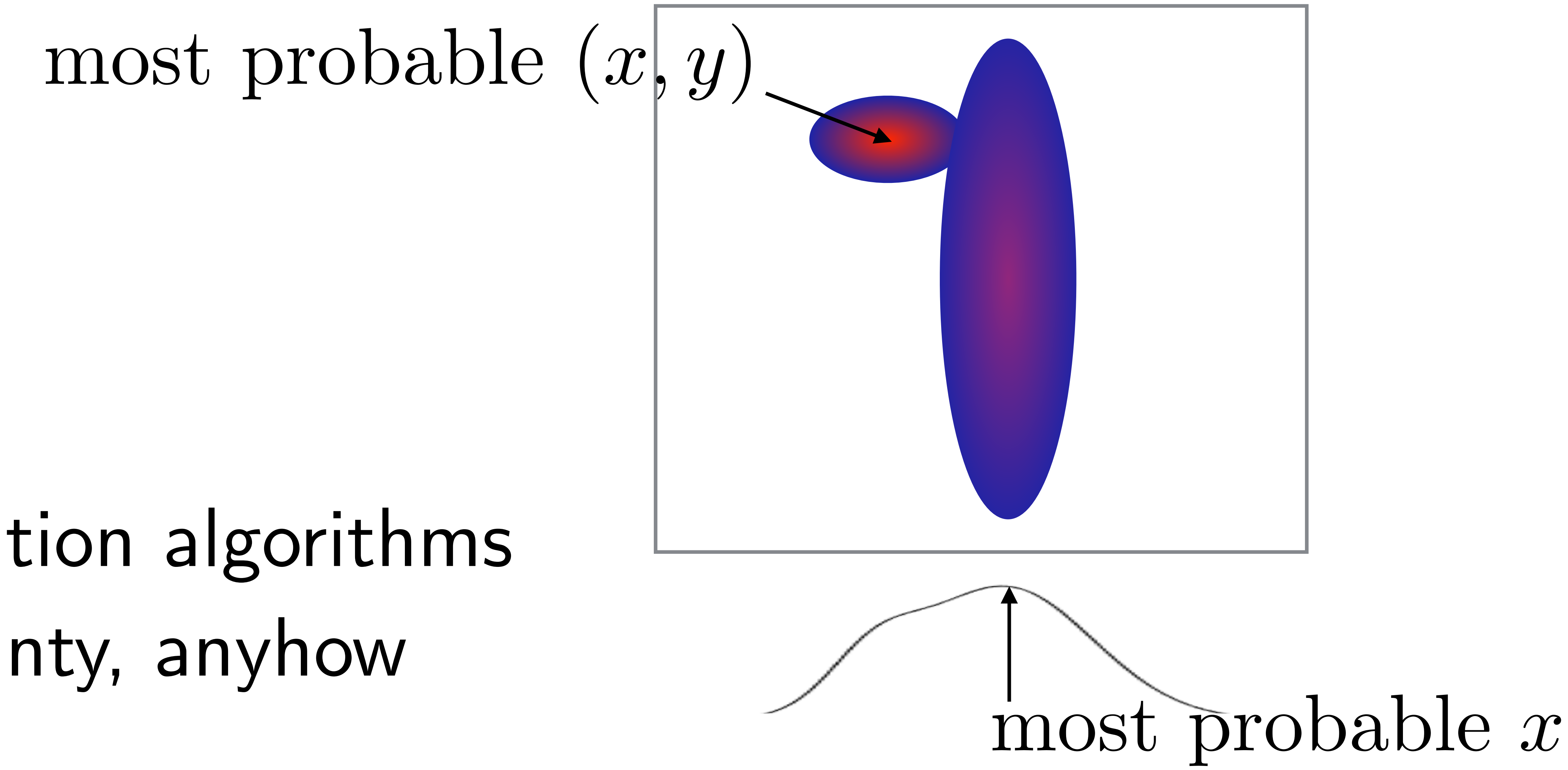
[Daphne Coller, Coursera]

Difference Between Recognizing Whole and a Part

- Example: joint probability p(X,Y):
  - p(A,A) = 0.4
  - p(A,B) = 0.1
  - p(B,A) = 0.3
  - p(B,B) = 0.2
- Goal: decide whether X is A or B (say we win 1$ if we guess right)
  - Approach 1: the most probable joint state is AA -> decide for A
  - Approach 2: compute marginal distribution p(X) -> decide based on that

- Continous example:
  - X - face position, Y - arm position
  - Want to know face position

- In practice, however we deal with approximation algorithms that behave poorly at high levels of uncertainty, anyhow



Conclusion

- Summary
  - Probabilistic models describe how some part of world works
  - Well suited for reasoning with uncertainty and posing many recognition problems
  - Graphical Models are probabilistic models
    - Have an underlying graph-like structure
      - The structure is a way of simplification and is related to the structure of an application
      - Modeling is needed to come up with a good structure
      - The space complexity is tractable
    - Solving the recognition problems (the time complexity) may be difficult
    - But still often possible, areas of applications of GMs:
      - Computer Vision
      - Bioinformatics
      - Communications
      - ...

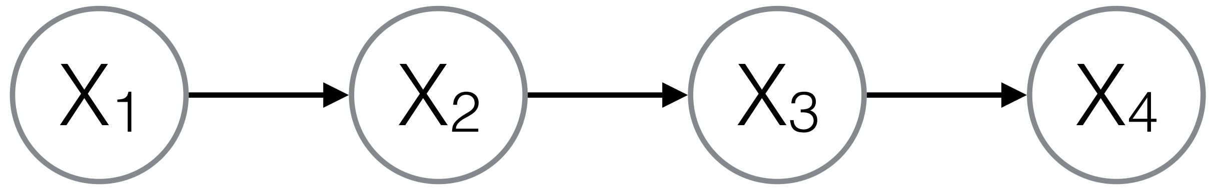
Hidden Markov Model

Goals

- Good for Classical Education
- Illustration of MAP and marginals problems that can be solved without hacks
- A very good starting point for understanding methods that work in general graphs (MRFs)
- In fact many methods are only understood as an extension of exact algorithms on trees
- There are actually many applications

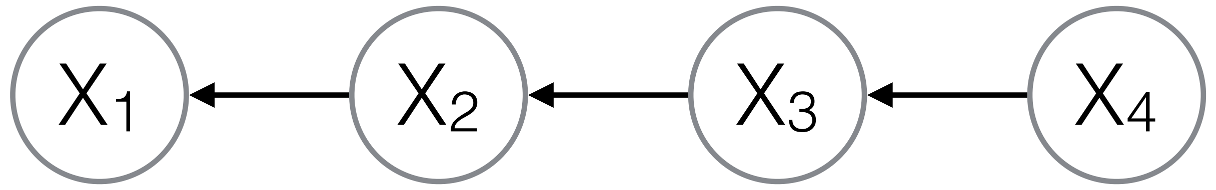
Markov Chain

Directed GM



$$p(x, y) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

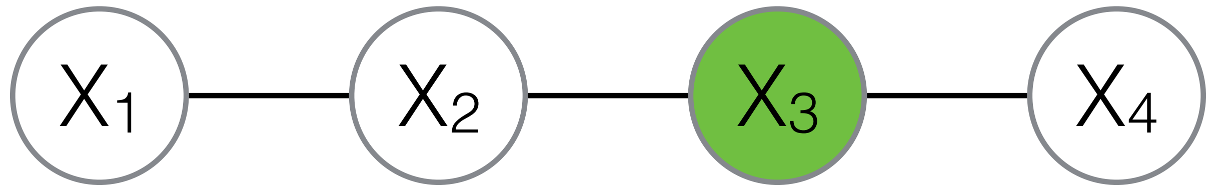
Equivalent directed GM



$$p(x, y) = p(x_n) \prod_{i=1}^{n-1} p(x_i \mid x_{i+1})$$

For converting between these forms, we will need an algorithm for computing marginals

Undirected GM



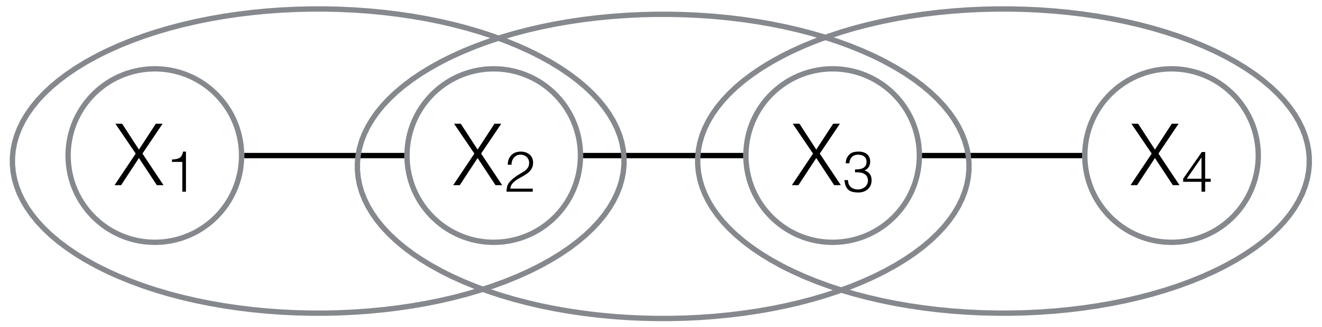
Given X_3, X_2 and X_4 are independent

...

Factorization: $\prod_{ij}^{n} g(x_i, x_j)$

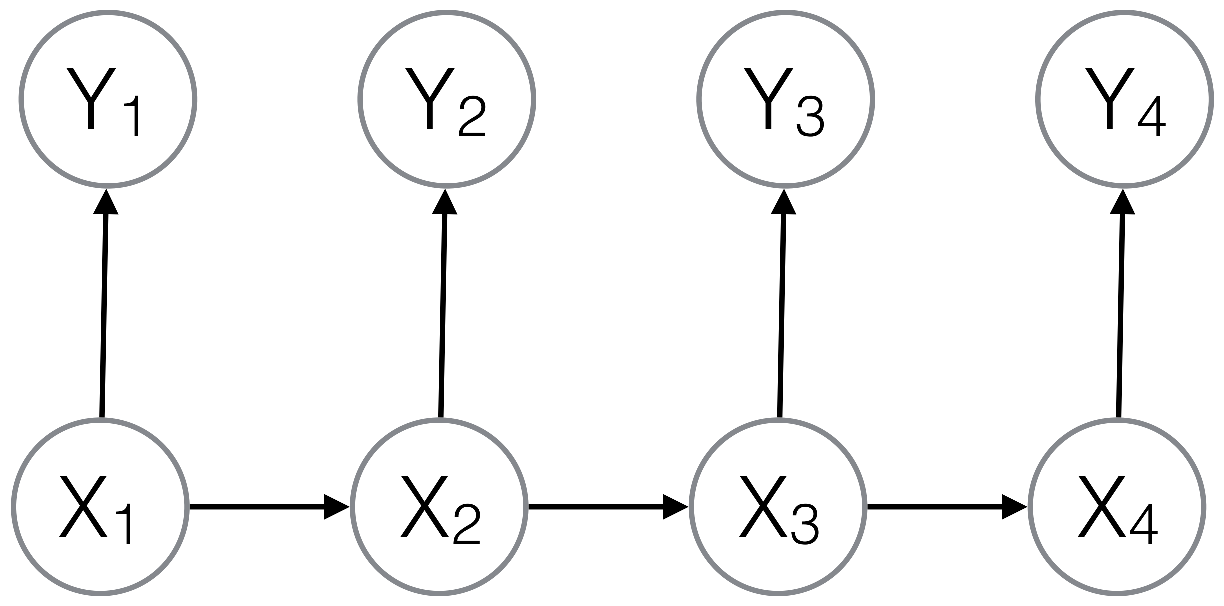
Factorization in marginals:

$$\prod_{ij}^{n} \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \prod_{i} p(x_i)$$



Hidden Markov Model

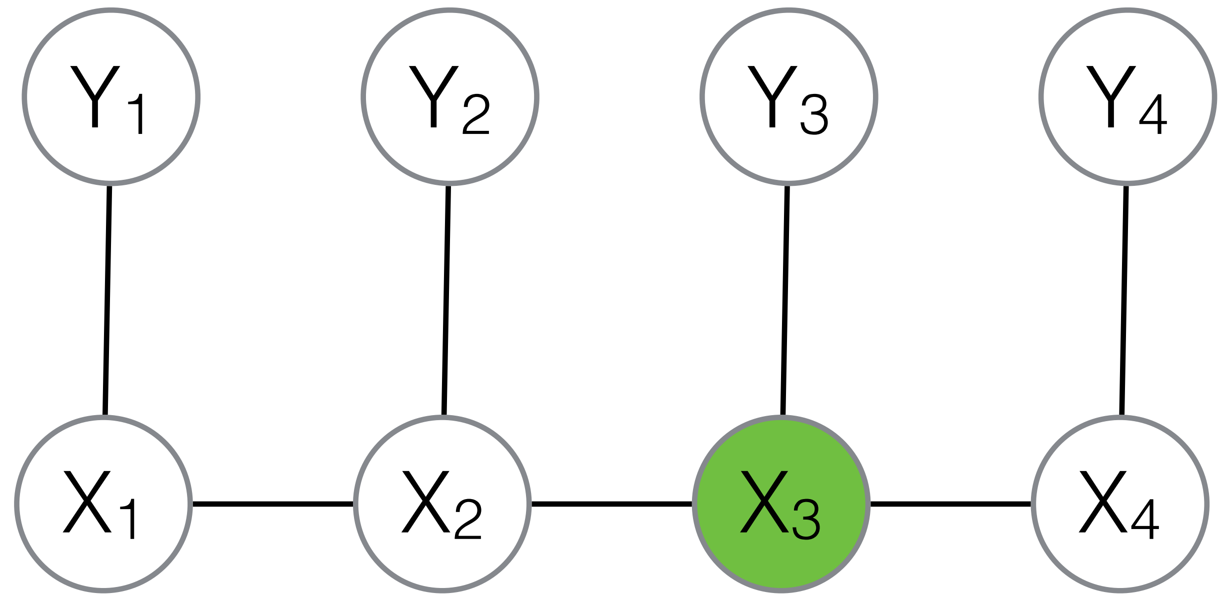
Directed GM



observed

hidden

Undirected GM



$$p(x, y) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}) \prod_{i=1}^{n} p(y_i \mid x_i)$$

Given X_3, Y_3 is independent of the rest

Given X_3, X_2 and X_4 are independent

$$\prod_{i=2}^{n} g(x_i, x_{i-1}) \prod_{i=1}^{n} f(y_i, x_i)$$

- Sequences (text, grammars)
- Time dependencies (speech, tracking, DNA)
- Good for understanding many things
- Basis for generalization of several algorithms

Observe that: $p(x) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$ – Markov chain

MAP Problem

Maximum a posteriori (MAP): given observation y we want to find the most probable hidden configuration x: $\max_x p(x \mid y)$

Recall $p(x \mid y) = p(x, y)/p(y)$

For fixed y, pdf $p(x \mid y)$ is a Markov chain on x:

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} = \frac{1}{p(y)} p(x) \prod_i p(y_i \mid x_i) = \frac{1}{p(y)} \prod_{ij} g_{ij}(x_i, x_j) \prod_i g_i(x_i)$$

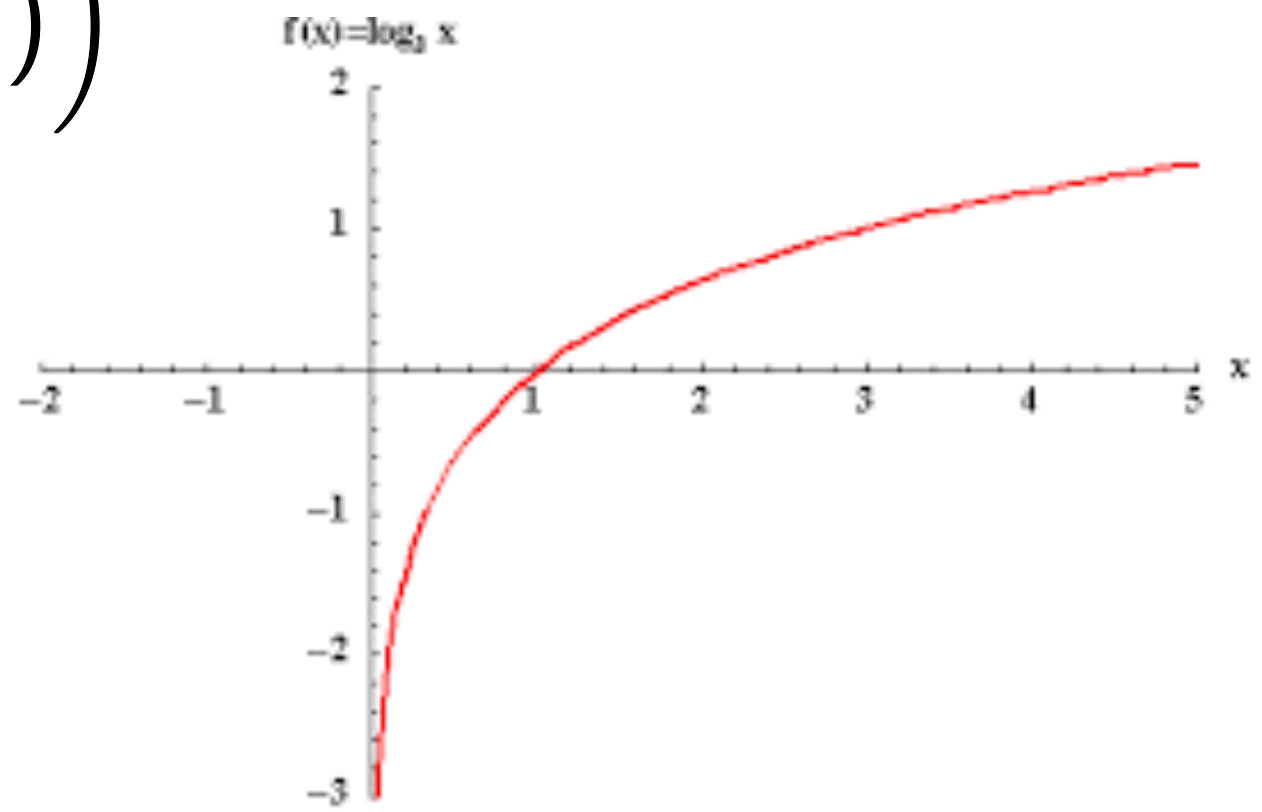
(We'll need marginalization computations to recover a directed or marginals factorization)

To find the MAP solution x we don't need to know $p(y)$:

$$\operatorname*{argmax}_x \underbrace{\prod_i g_i(x_i)}_{\text{data}} \underbrace{\prod_{ij} g_{ij}(x_i, x_j)}_{\text{prior}}$$

Energy Minimization

$$\underset{x}{\operatorname{argmax}} \prod_i g_i(x_i) \prod_{ij} g_{ij}(x_i, x_j) = \underset{x}{\operatorname{argmax}} \log \left(\prod_i g_i(x_i) \prod_{ij} g_{ij}(x_i, x_j) \right)$$



log is monotone, all factors non-negative

$$f_a(x_a) = -\log g_a(x_a)$$

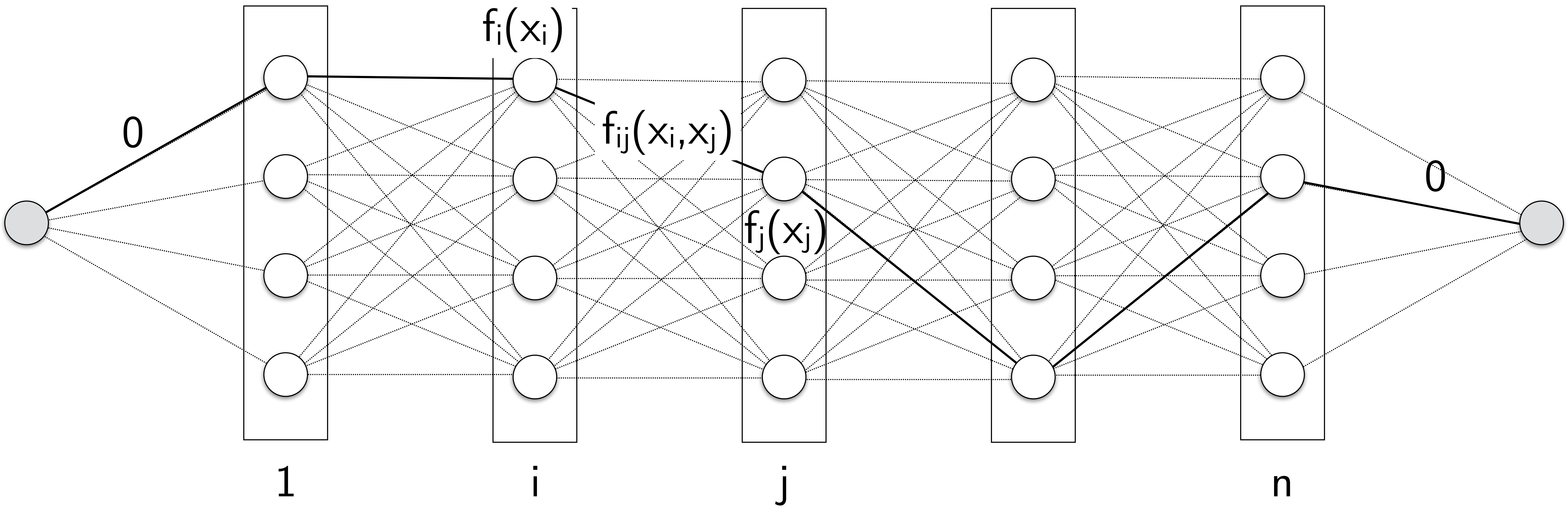
$$\underset{x}{\operatorname{argmin}} \left[E(x) = \underbrace{\sum_i f_i(x_i)}_{\text{data}} + \underbrace{\sum_{ij} f_{ij}(x_i, x_j)}_{\text{prior}} \right]$$

- Need to find a minimum of a function which is a sum of functions of one variable (unary terms) and two variables (pairwise terms)

As Shortest Path

$$\underset{x}{\operatorname{argmin}} \left[E(x) = \sum_i f_i(x_i) + \sum_{ij} f_{ij}(x_i, x_j) \right]$$

(Construction known as Trellis graph)

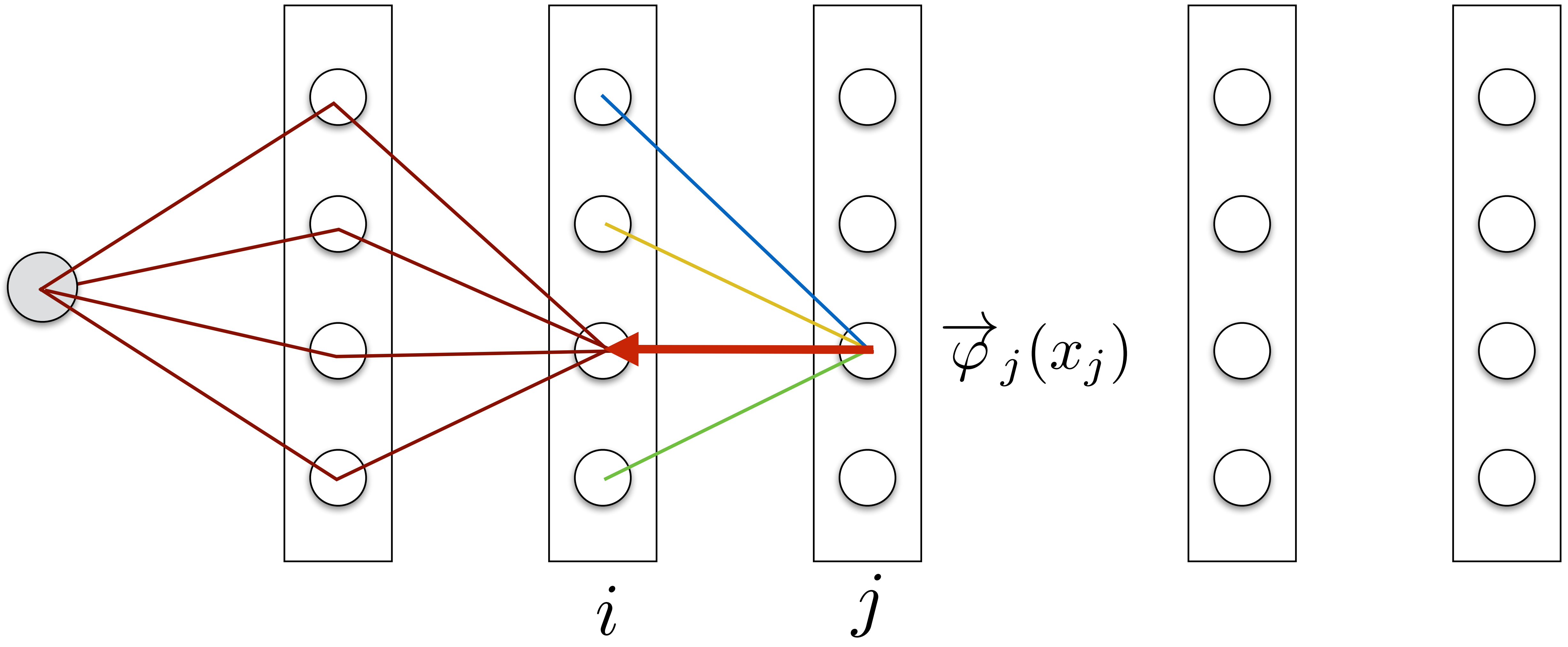


- Paths map one to one to labelings x; cost of a path equals E(x)
- Shortest path <=> MAP solution

Algebraic View / Viterbi Algorithm

- Problem:

$$\min_x \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j)$$



- Use distributivity:

$$\min(a + c, b + c) = min(a, b) + c$$

$$\min_{x_1,\ldots x_n} \left[f_{1,2}(x_1, x_2) + f_1(x_1) + \ldots\right] = \min_{x_2,\ldots x_n} \Big[\underbrace{\min_{x_1}[(f_{1,2}(x_1, x_2) + f_1(x_1))}_{\overrightarrow{\varphi}_2(x_2)} + \ldots \Big]$$

- Recurrent update:

$$\overrightarrow{\varphi}_1(x_1) = 0$$

$$\overrightarrow{\varphi}_j(x_j) = \min_{x_i} \left(f_{ij}(x_i, x_j) + f_i(x_i) + \overrightarrow{\varphi}_i(x_i)\right)$$

Viterbi Algorithm:

Forward pass: computes best path from the left

Backward pass: backtrack the minimizer

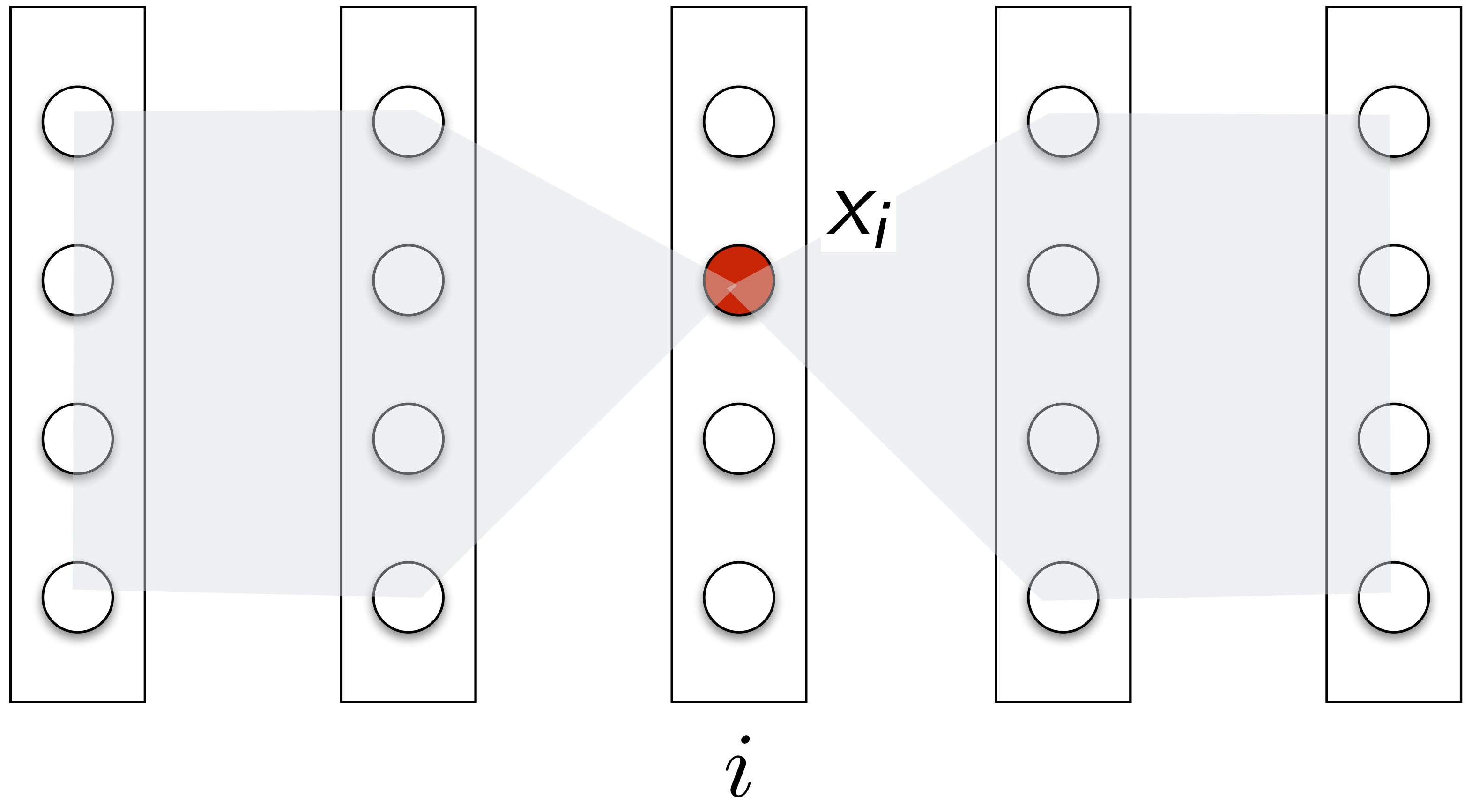
Shortest path from the left to every state. Core of all message passing algorithms

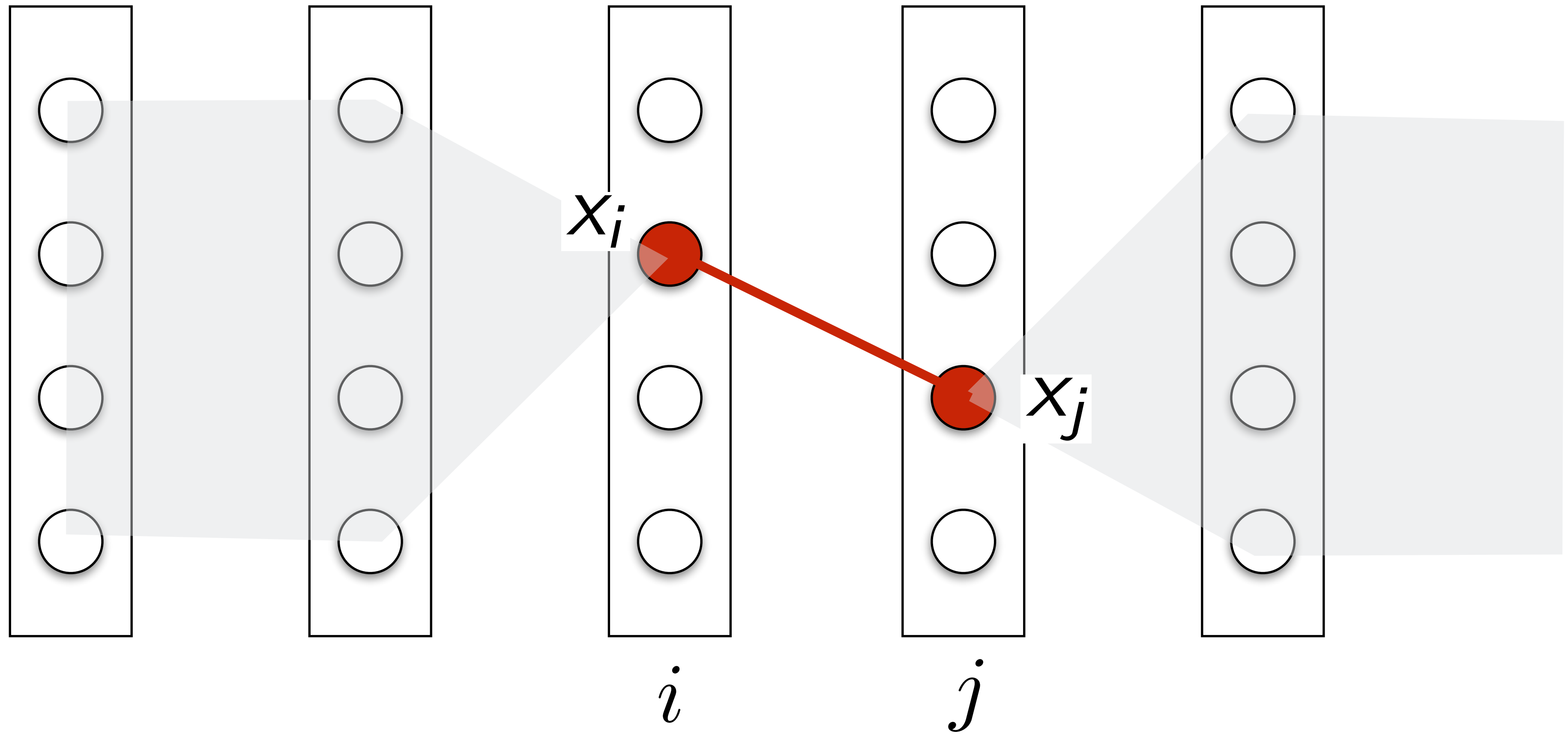
Marginals

Given factorization $p(x) = \frac{1}{Z} \prod_i g_i(x_i) \prod_{ij} g_{ij}(x_i, x_j)$

Compute $p(x_i)$, $p(x_i, x_j)$:

$$p(x_i) = \sum_{x_{\mathcal{V} \setminus \{i\}}} p(x) = \sum_{x_1, \ldots x_{i-1}, \square, x_{i+1} \ldots x_n} p(x); \qquad p(x_i, x_j) = \sum_{x_{\mathcal{V} \setminus \{i,j\}}} p(x)$$





$$p(x_i) \propto \overrightarrow{M}_i(x_i) g_i(x_i) \overleftarrow{M}_i(x_i)$$

$$\sum_{x_i} p(x_i) = 1$$

$$p(x_i, x_j) \propto \overrightarrow{M}_i(x_i) g_i(x_i) g_{ij}(x_i, x_j) g_j(x_j) \overleftarrow{M}_j(x_j)$$

$$\sum_{x_i, x_j} p(x_i, x_j) = 1$$

Marginals

Given factorization $p(x) = \frac{1}{Z} \prod_i g_i(x_i) \prod_{ij} g_{ij}(x_i, x_j)$

Compute $p(x_i)$, $p(x_i, x_j)$:

$$p(x_i) = \sum_{x_{\mathcal{V} \setminus \{i\}}} p(x) = \sum_{x_1, \dots x_{i-1}, \square, x_{i+1} \dots x_n} p(x); \qquad p(x_i, x_j) = \sum_{x_{\mathcal{V} \setminus \{i,j\}}} p(x)$$

- Use distributivity: $a \cdot c + b \cdot c = (a + b) \cdot c$,

$$\sum_{x_1, \dots x_{i-1}} [g_{12}(x_1, x_2) \cdot g_1(x_1) \cdot (\dots)] = \sum_{x_2, \dots x_{i-1}} [\underbrace{\sum_{x_1} [(g_{12}(x_1, x_2) \cdot g_1(x_1))}_{\overrightarrow{M}_2(x_2)} \cdot (\dots)]$$

Note: this is matrix-vector product

- Recurrent update:

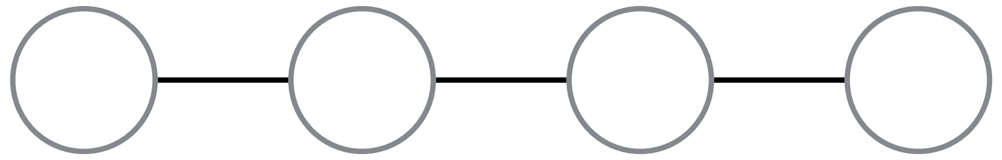
$$\overrightarrow{M}_1(x_1) = 1$$

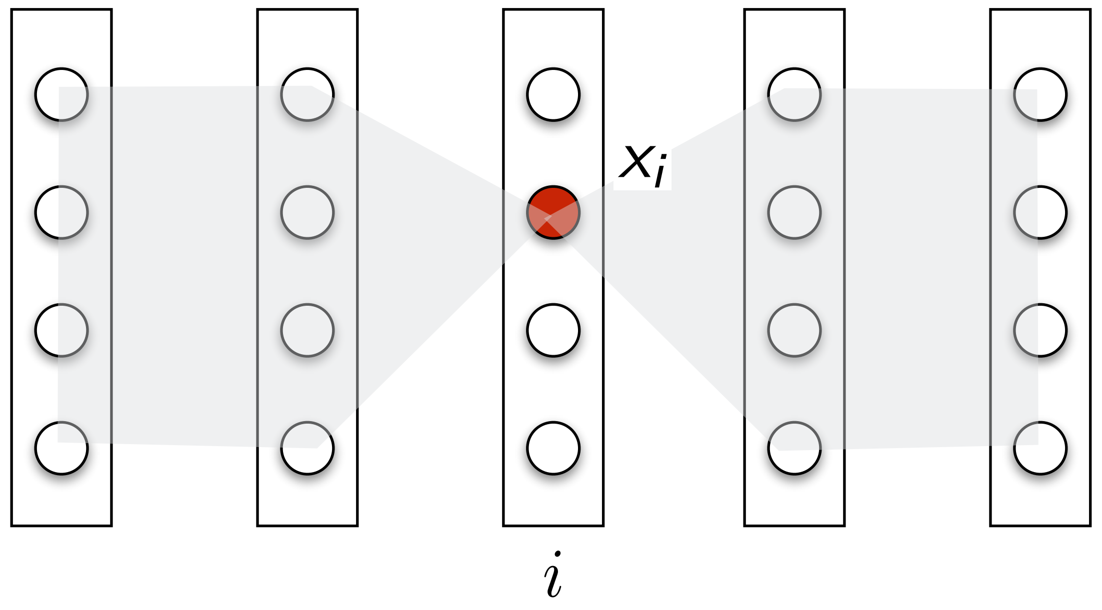
$$\overrightarrow{M}_j(x_j) = \sum_{x_i} (g_{ij}(x_i, x_j) \cdot g_i(x_i) \cdot \overrightarrow{M}_i(x_i))$$

Forward-Backward Algorithm

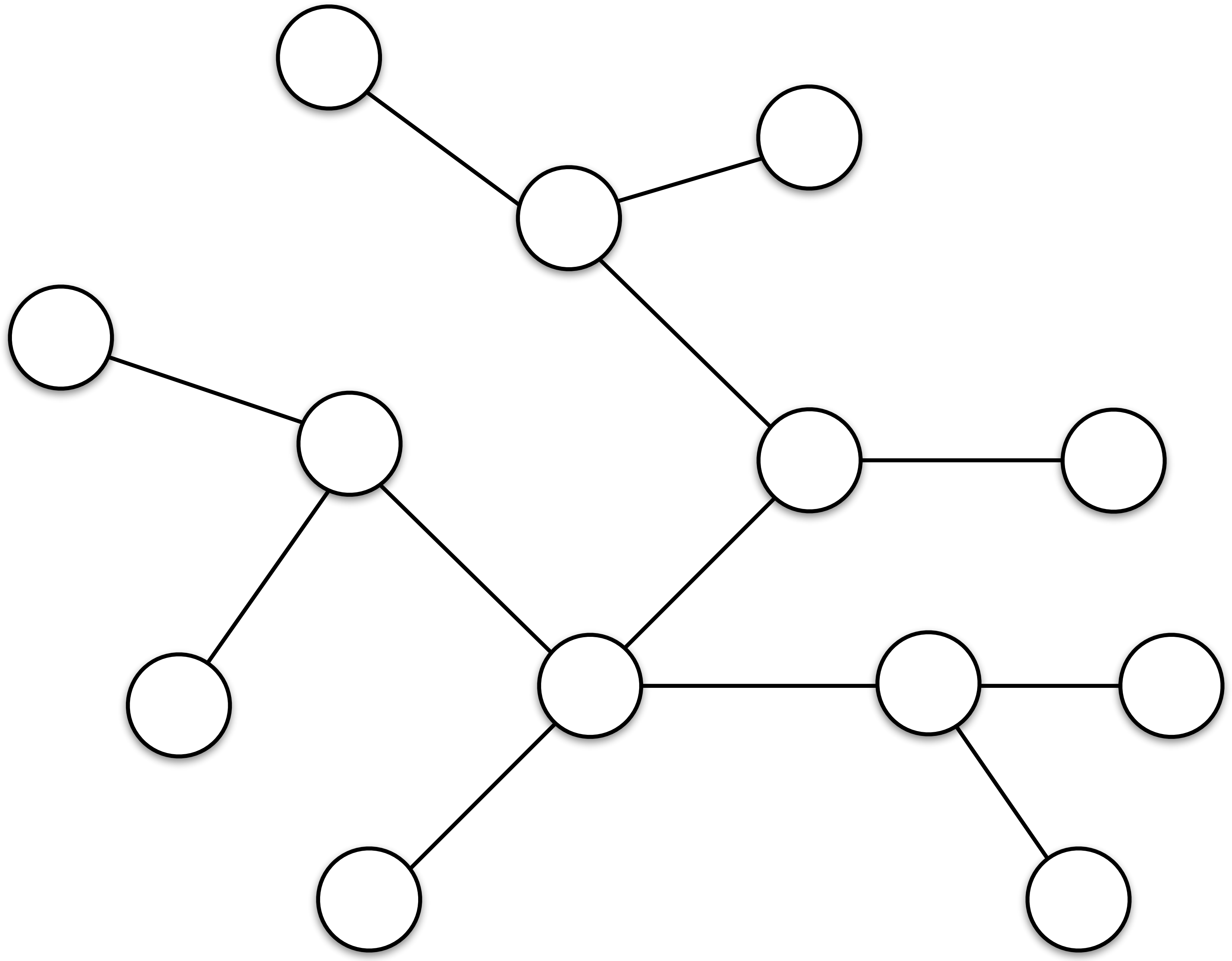
- **Forward**: compute left marginals recurrently: $\overrightarrow{M}_i(x_i)$
- **Backward**: compute right marginals recurrently $\overleftarrow{M}_i(x_i)$
- Compose marginals as $p(x_i) = \overrightarrow{M}_i(x_i) g_i(x_i) \overleftarrow{M}_i(x_i)$

Exercise: Extend to Trees





$$p(x_i) \propto \overrightarrow{M}_i(x_i) g_i(x_i) \overleftarrow{M}_i(x_i)$$



Generalized Algorithms

- Did you notice the similarity of computations in MAP and marginals problems?

Actually, for any semi-ring $(R, \oplus, \otimes)$ there holds distributivity:

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

$$(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$$

We can write a generalized algorithm for the problem of $\oplus\otimes$ marginals on a chain (tree):

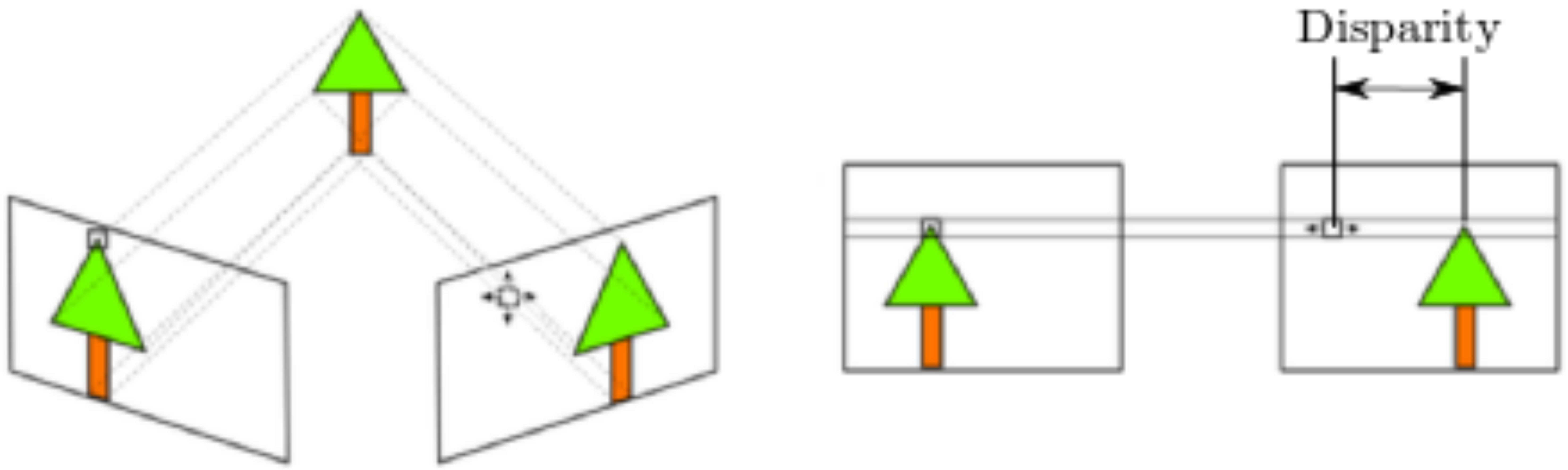
$$m_i(x_i) = \bigoplus_{x_{\mathcal{V} \setminus i}} \bigotimes_{ij} g_{ij}(x_i, x_j)$$

For example: $(\mathbb{B}, \vee, \wedge)$, $([0,1], \min, \max)$, $(\mathbb{R}, \mathrm{logsumexp}, +) \sim (\mathbb{R}_+, +, \times)$

[Schlesinger M.I. Ten lectures in statistical and structural pattern recognition]

Example: Scan-line Stereo

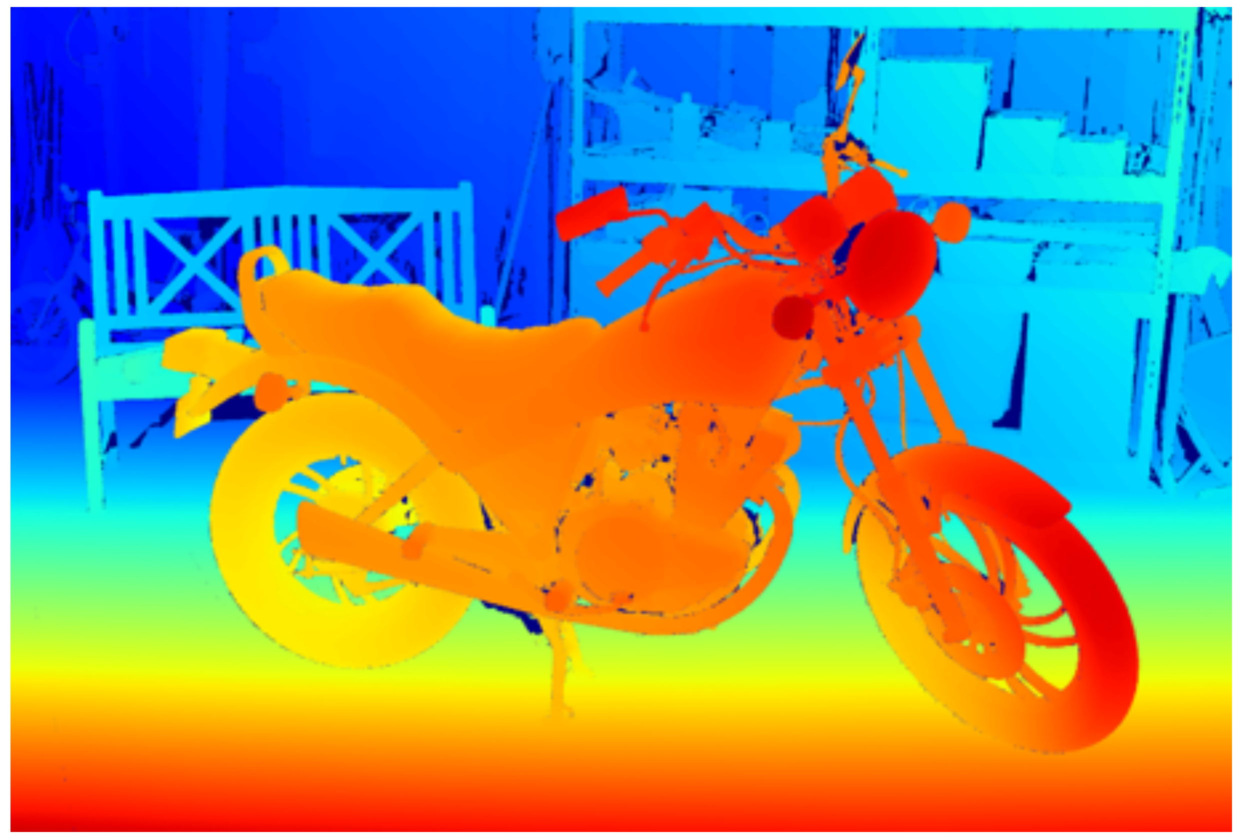
- Input
  - Two images from a calibrated camera pair
  - Rectified: epipolar lines correspond to image rows



- Problem
  - For each pixel in the left image find the corresponding pixel in the right image

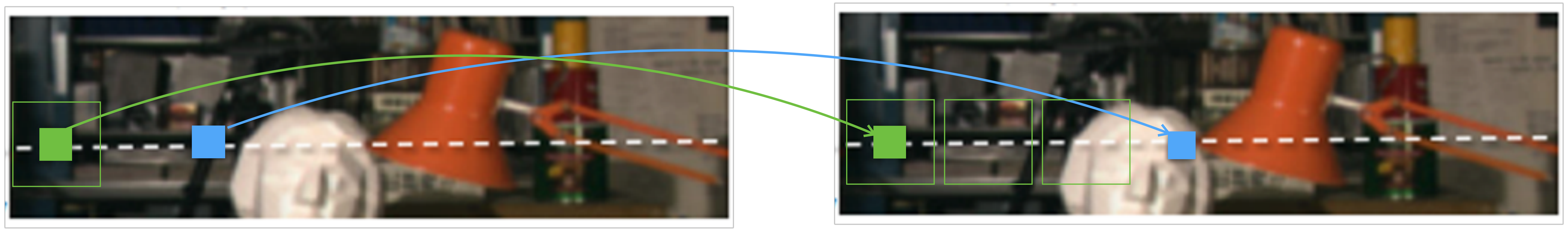
- Output
  - Dense depth (disparity) map

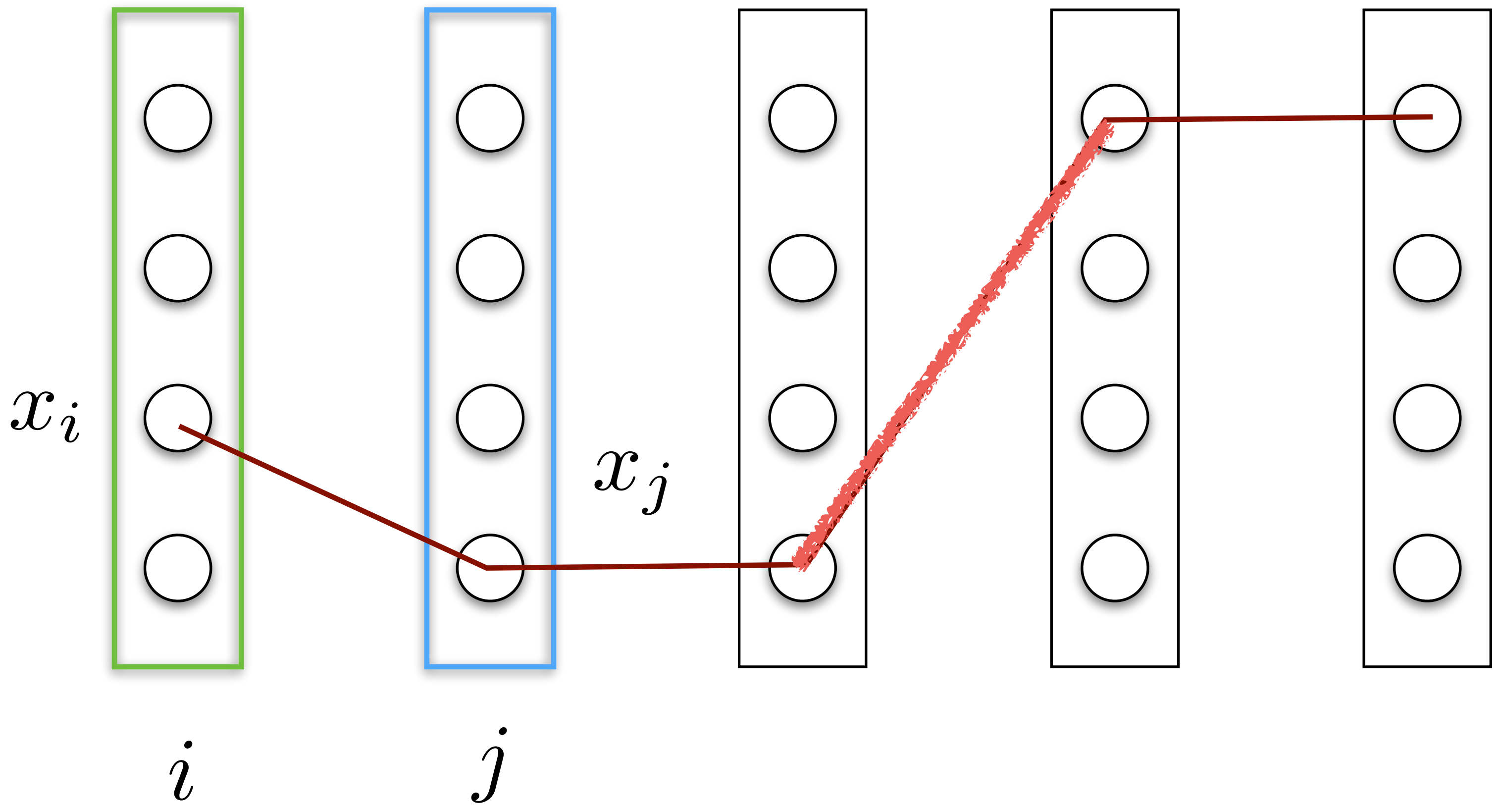
Input Pair



Disparity Map (GT)

Example: Scan-line Stereo





i - pixel

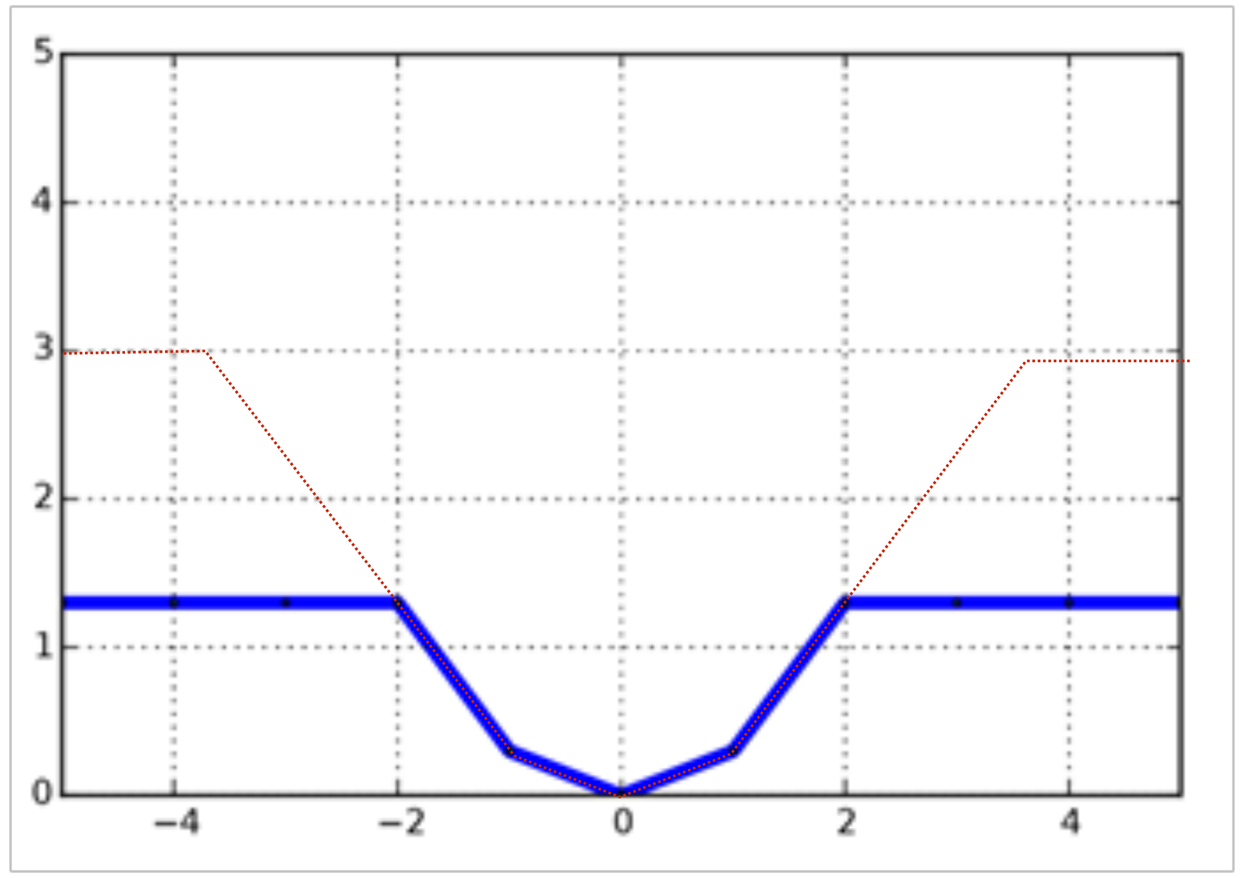
x_i - chosen disparity label

$x = (x_i \mid i \in \mathcal{V})$ - labeling

$f_i(x_i)$ - matching cost

$f_{ij}(x_i, x_j)$ - smoothness cost

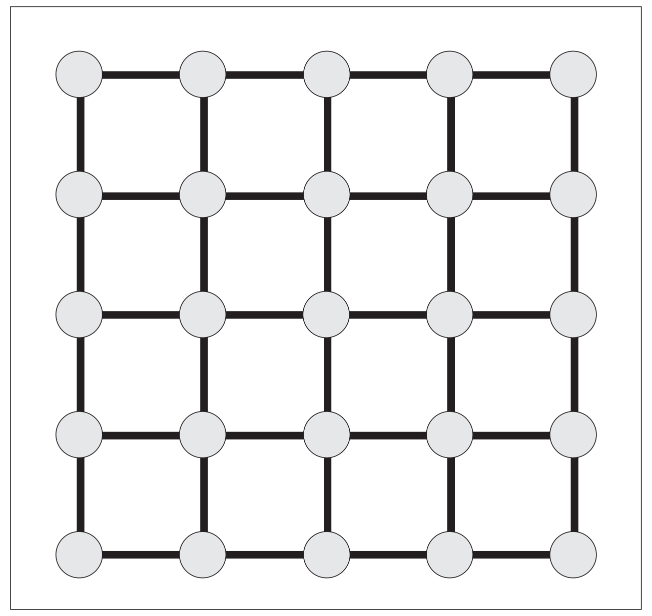
$$\min_x \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j)$$

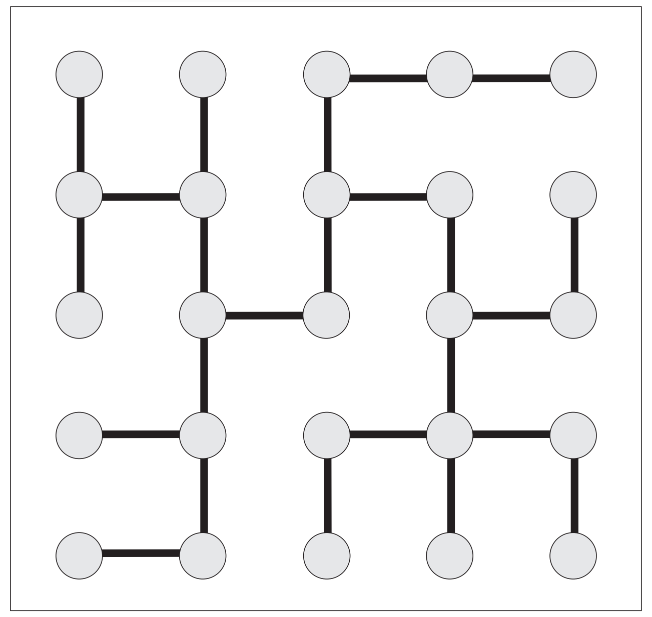


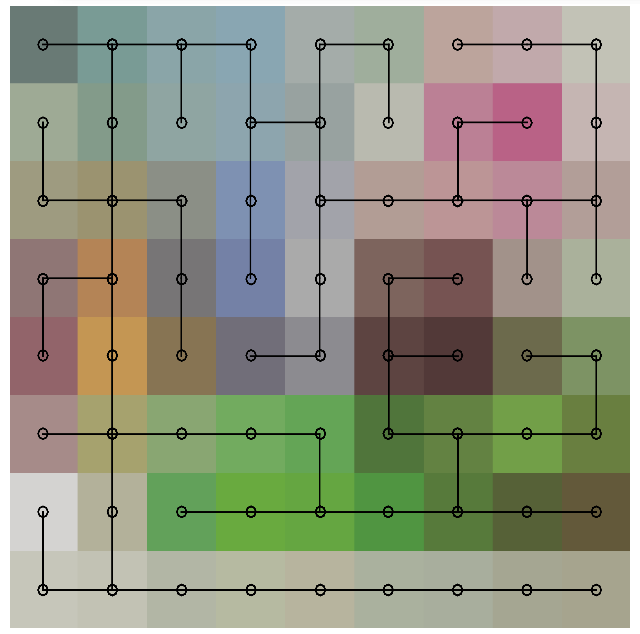
Tree-based Heuristics for Stereo

Veksler-05 → Psota et al. ICCV-15

full graph

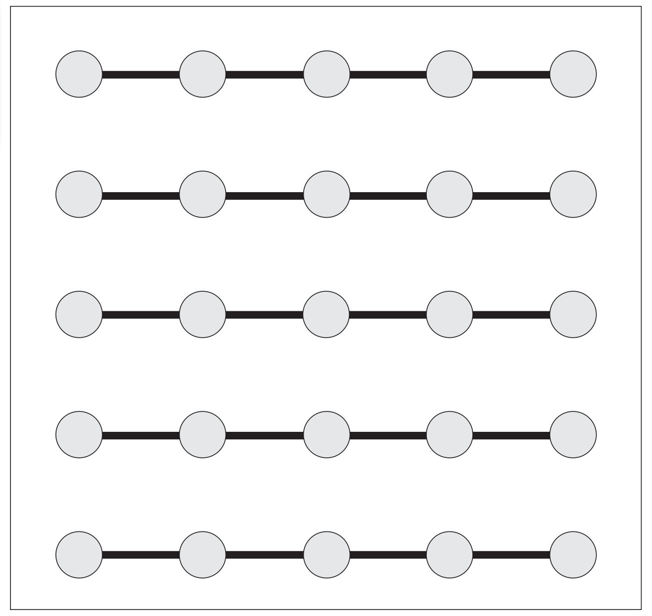


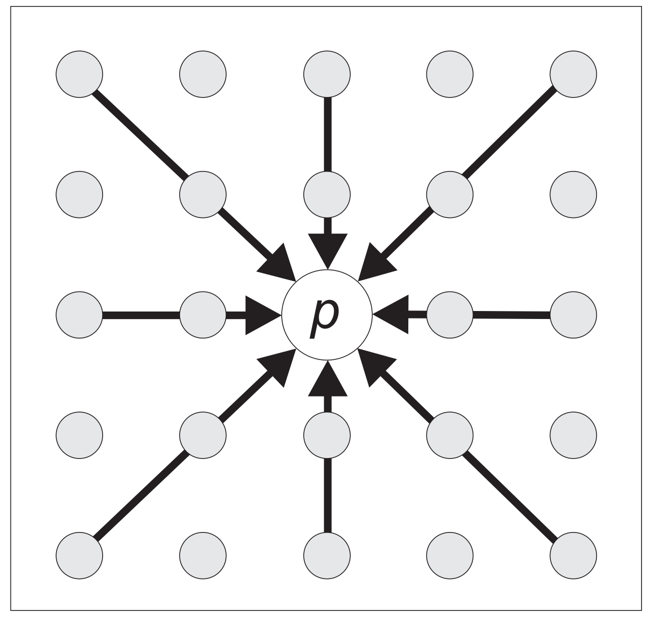




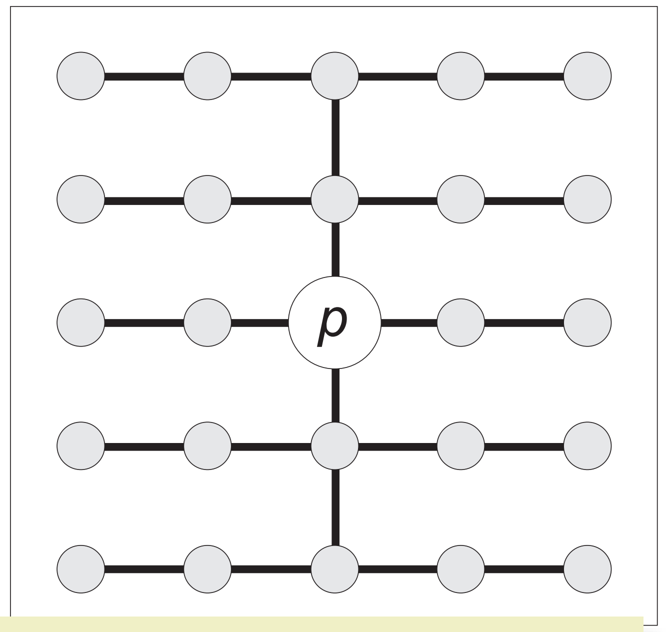
+ connect similar colors first
+ learned potentials

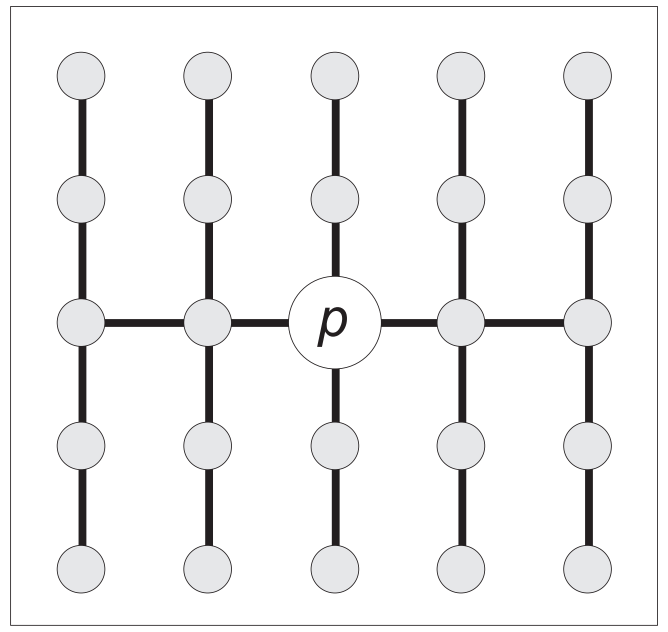
scanline DP





Hirschmüller-05 (SGM)
+ own tree for each pixel
+ reuse messages in DP





Bleyer & Gelautz-VISSAP-08
+ own tree for each pixel
+ larger coverage

Conclusion

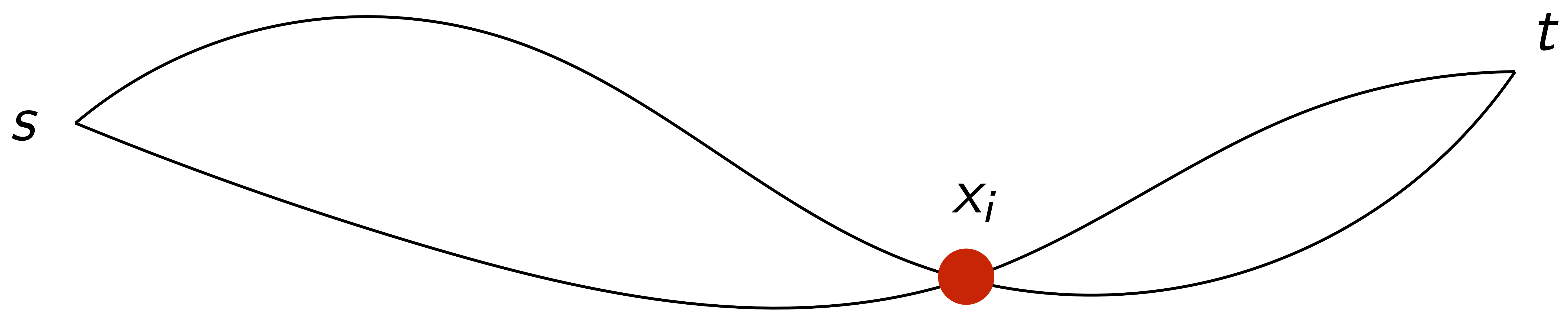
- Hidden Markov Model is very similar to Markov Chain
- All problems seem to be solvable with a kind of dynamic programming (but e.g. unsupervised learning isn't)
- In fact, trees seem to be important

Further Topics

- Junction Tree Algorithm
- Unsupervised learning (hidden states not observed) — Baum-Welsche algorithm
- Parallel algorithms O(n log(K)) time with K processors:
  - sum-product: Fourier transform
  - min-sum: lower envelopes, distance transform
- Kalman Filter
- Markov Chain Monte Carlo
  - Ergodicity and stationary distribution
- Finale state automata
- Markov Decision Processes

More on Dynamic Programming

Conditional Independence and Bellman Optimality



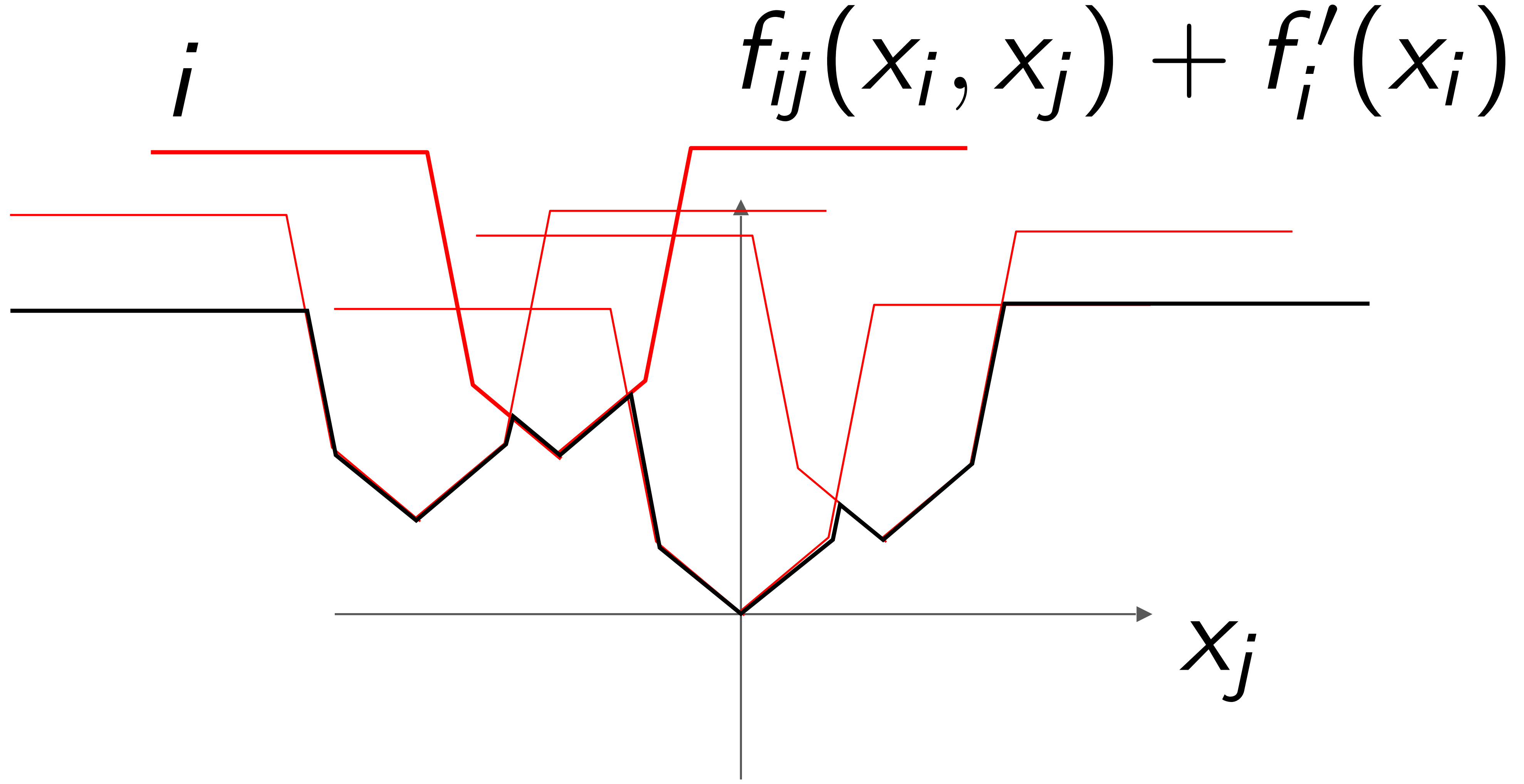
- Given x_i, the optimal solution consists of optimal solution (s to x_i) and (x_i to t)
- Variables ($X_{1,}$… X_{i-1}) and ($X_{i+1,}$… X_n) are conditionally independent given X_i

Lower Envelopes

One minimization of the form

$$\overrightarrow{\varphi}_j(x_j) = \min_{x_i}(\overrightarrow{\varphi}_i(x_i) + f_i(x_i) + f_{ij}(x_i, x_j))$$

is the problem of finding a lower envelope of a set of functions well studied in geometry / graphics



- Lower envelope (distance transform)

$f_{ij}(x_i, x_j) = w_{ij}\rho(x_i - x_j)$

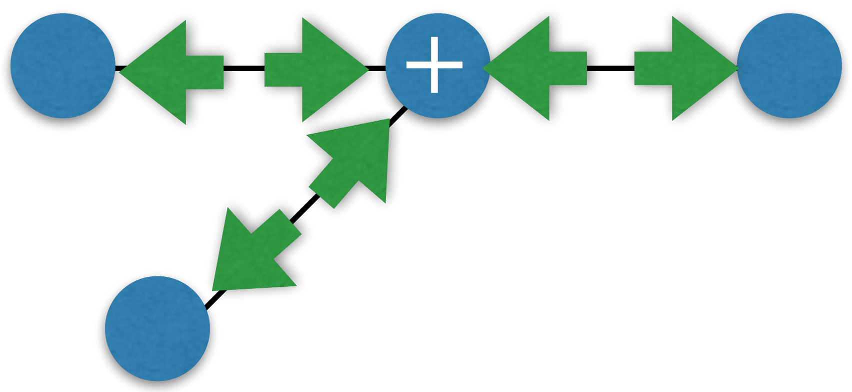
$O(nL^2)$ - naive approach, n variables, L labels

$O(nL)$ - efficient sequential algorithms [Hirata'96, Meijster'02] [Felzenszwalb&H.'06]

$O(n \log L)$ - efficient parallel algorithms, using L processors [Goodrich'86, Chen'02]

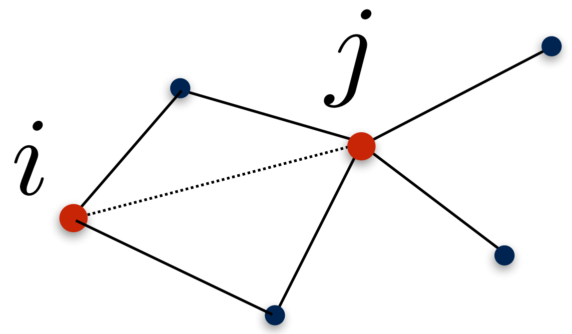
Max-Product BP, Tree-Reweighted[1]

- Can Run Message passing in parallel



$O(n)$ time, $O(n)$ processors

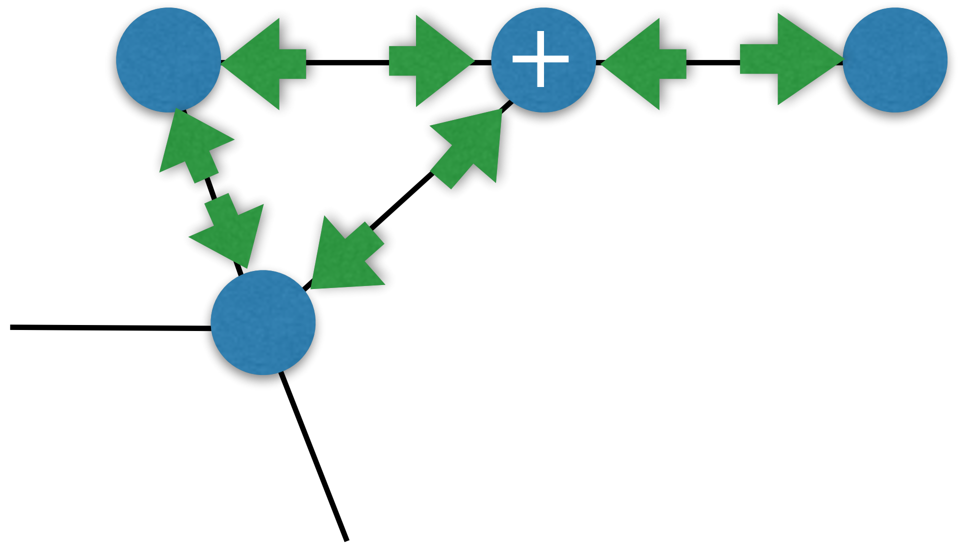
c.f. all shortest paths in a graph



$$d(i,j) := \min_k (d(i,k) + d(k,j))$$

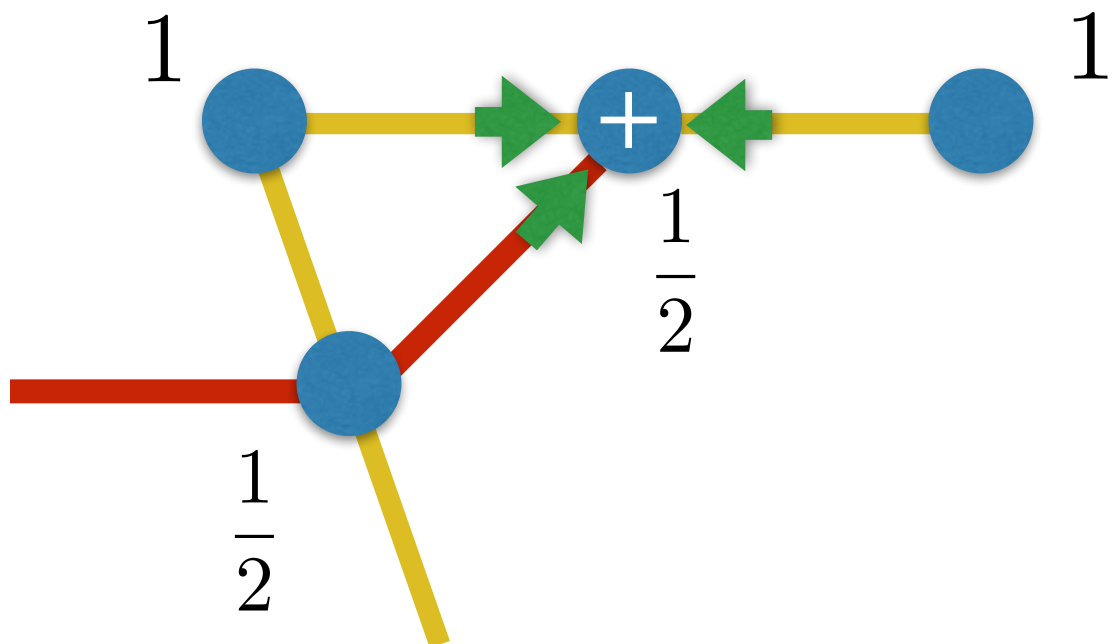
(Floyd–Warshall alg.)

- Can apply on graphs with loops (loopy BP)



- Over-counting
- May oscillate
- May diverge (unbounded)

- Tree-Reweighted [Wainwright'05]



- Decomposition into trees
- Connection to LP relaxation and its dual
- Parallel algorithm may still oscillate

Markov Random Fields

Goals

- Definitions
- Examples in Computer Vision
- Overview on MAP problem, one technique in detail
- Marginals problem — variational approach in detail

Random Field

- Collection of discrete random variables

$$X_1, X_2, \ldots X_n, \qquad X_i \in D$$

Definition

$p \colon D^n \to \mathbb{R}$ is a *random field* if $p(x) > 0 \ \forall x$, $\sum_x p(x) = 1$.

- Non-negativity is important for existence of conditional probabilities and other good reasons. Practically not a limitation.

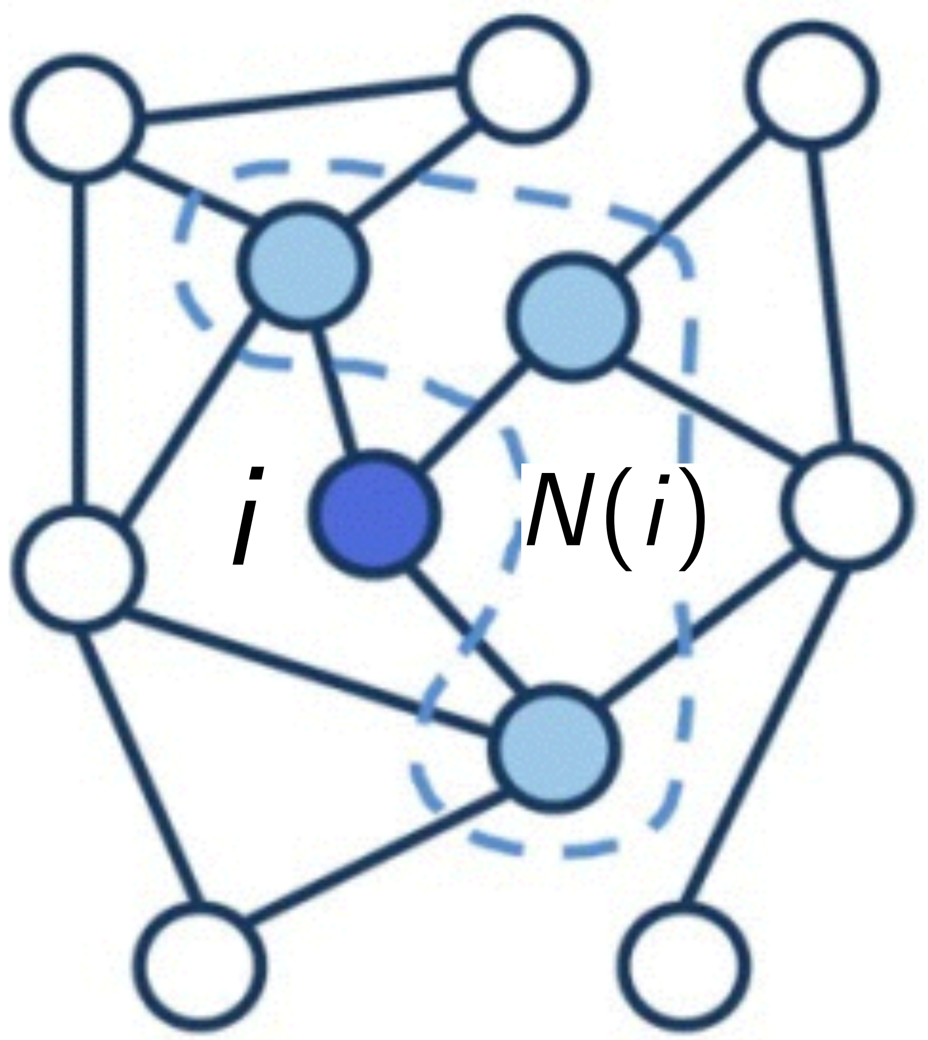
Definition

Random field p is a *Markov random field* if it satisfies some conditional independence (Markov) properties.

(Book: Lauritezen S.L.,"Graphical Models", 1996)

MRF w.r.t. a Graph

- Graph $G = (V, E)$
- Set of nodes V; random variables X_i, $i \in V$
- Set of edges E



- Local Markov Property w.r.t. G:
  - Given the neighbors of X_i, it is independent of the rest:

$$p(X_i \mid X_{V\setminus}) = p(X_i \mid X_{N(i)}), \forall i \in \mathcal{V}$$

- Pairwise Markov Property w.r.t. G:
  - Absent edge (i, j) in G iff X_i and X_j are conditionally independent given the rest of variables.

Theorem (Lauritzen 96)

Local and Pairwise Markov Properties are equivalent.

Definition

MRF w.r.t. graph G is a random field satisfying Markov property w.r.t. G

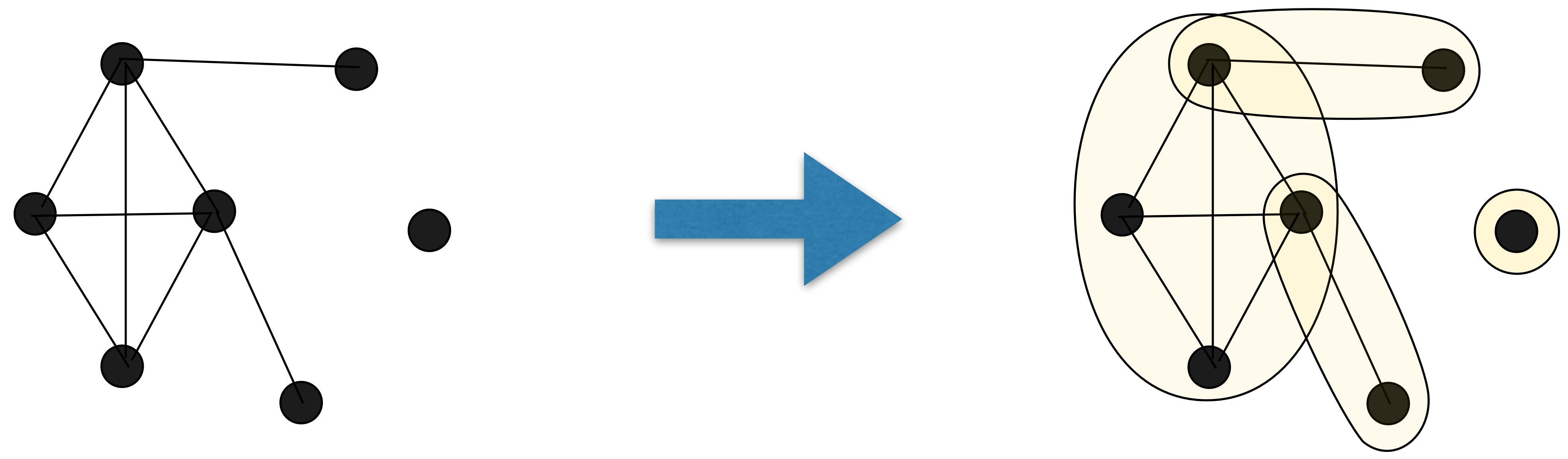
MRF factorization

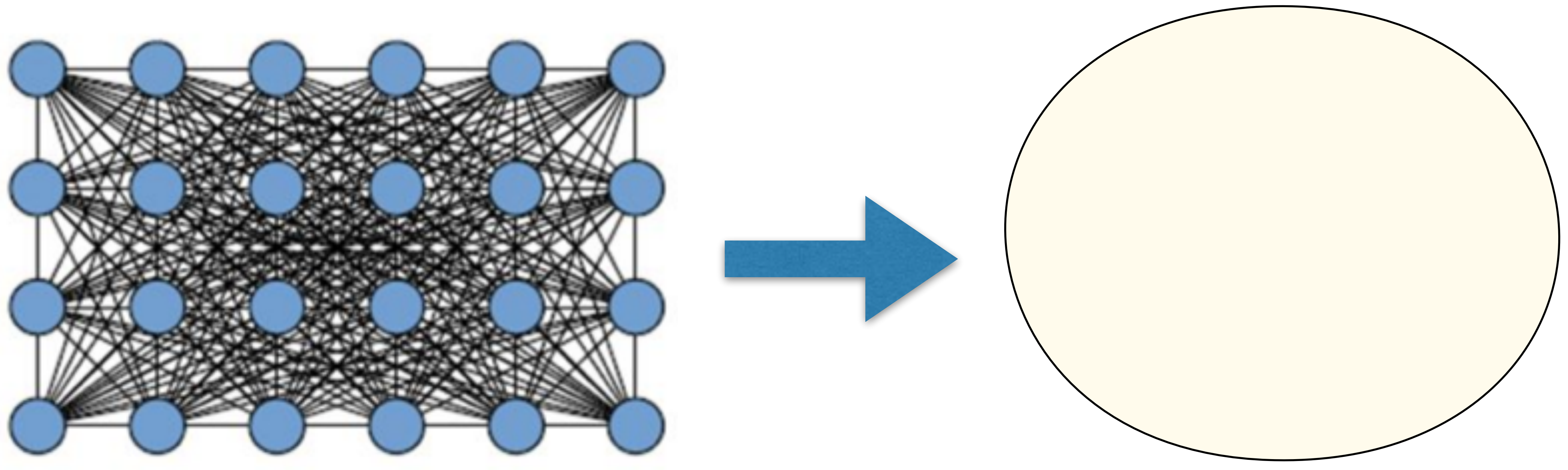
- Conditional independencies help to structure and simplify the distribution

Theorem (Hammersley-Clifford,1971)

MRF p w.r.t. graph G factors over cliques of G: $p(x) = \prod_{c \in C} f_c(x_c)$,

- C is the set of cliques – maximal fully connected subgraphs





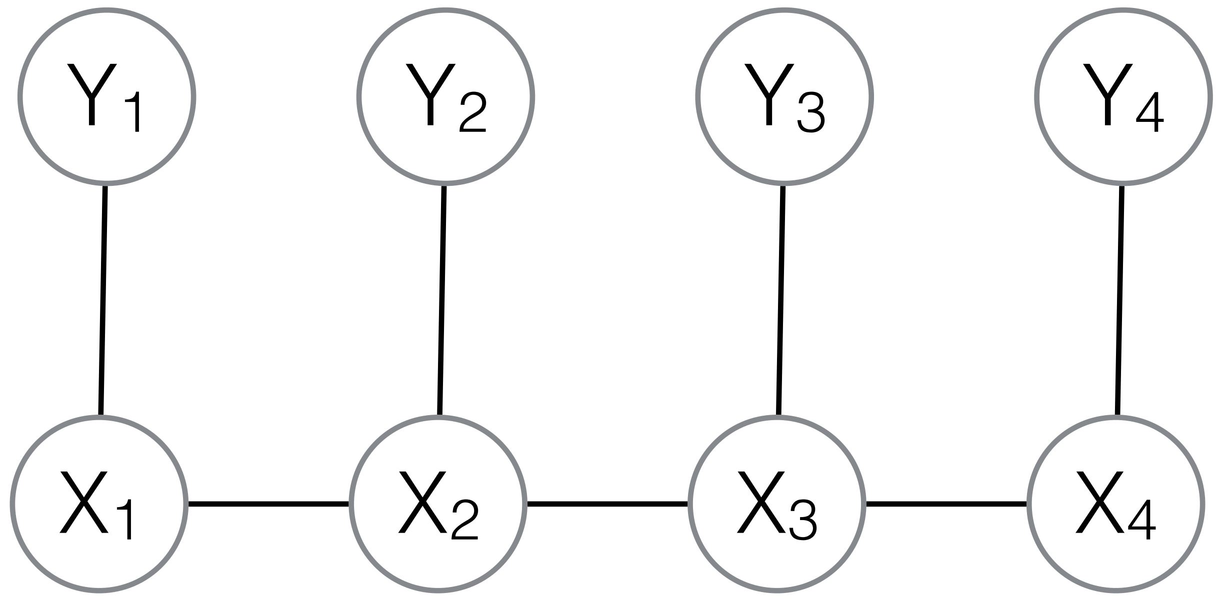
Definition

p is a *Gibbs Random field* if it factors as $p(x) = \prod_{c \subset S} f_c(x_c)$,

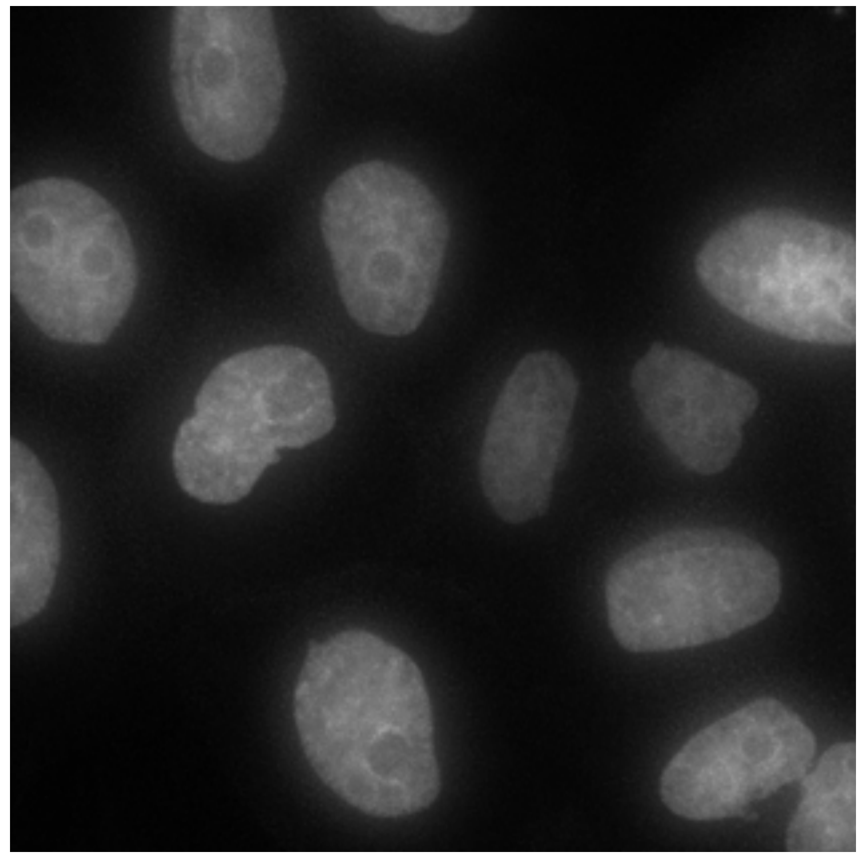
- Here we do not need c to be a clique in some graph
- Knowing factorization is more than knowing conditional independencies
- The factorization is what matters for the representation tractability and inference

Two-class Segmentation

MRF Model



Input Image

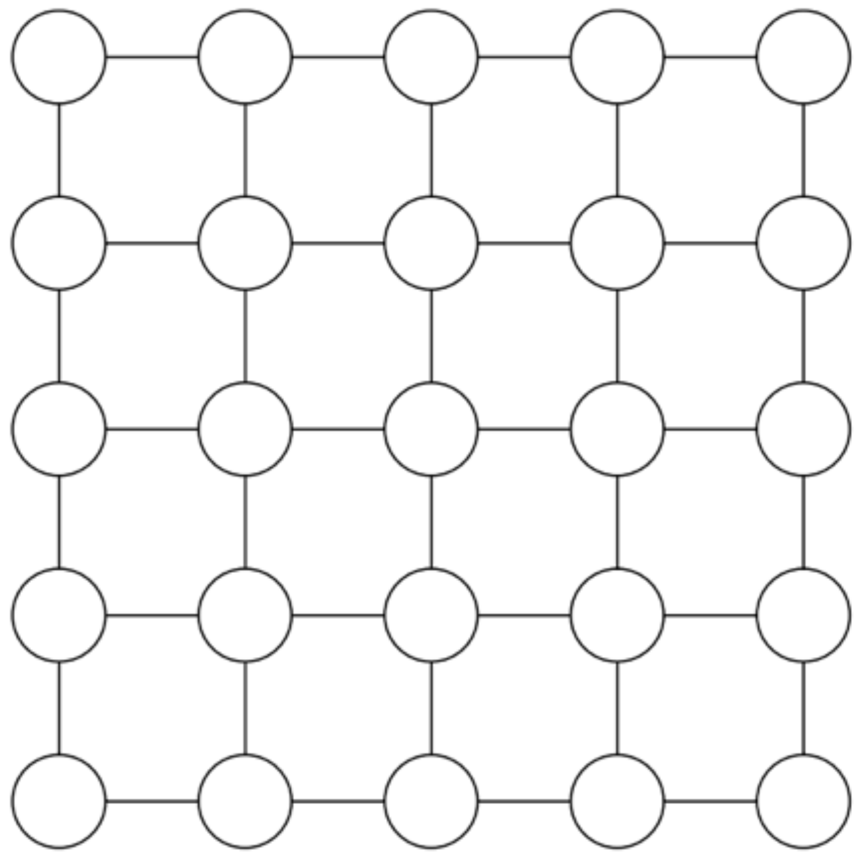


binary segmentation

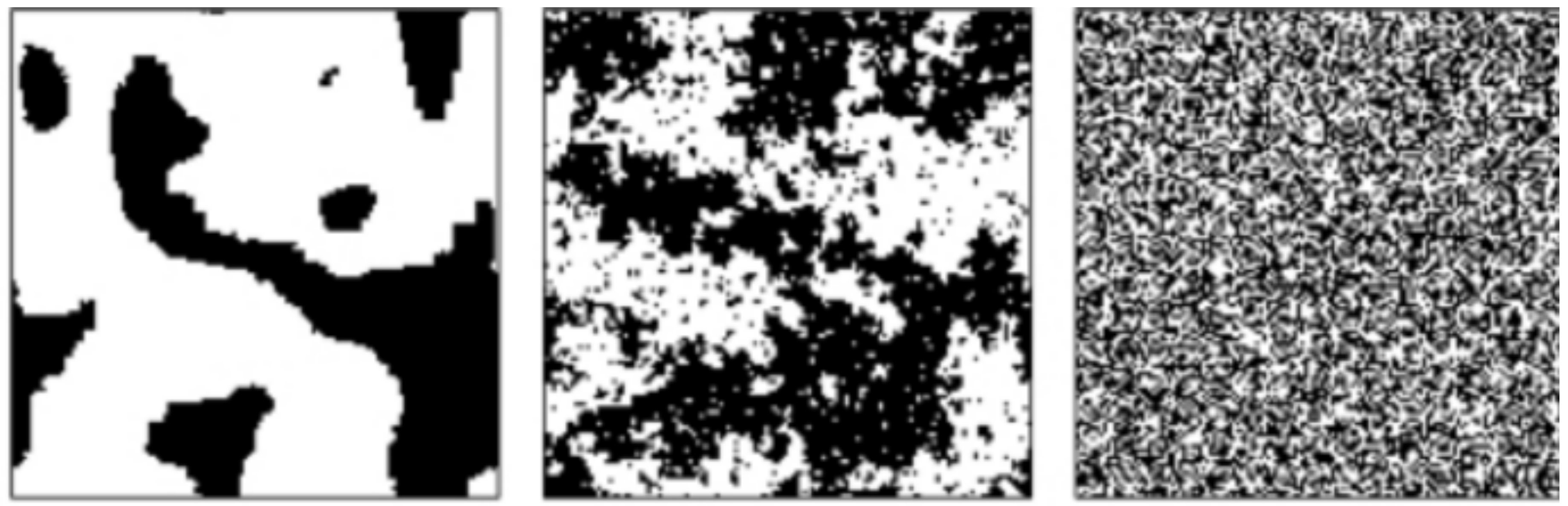
Observations: $p(y \mid x) = \prod_i p(y_i \mid x_i)$

Prior: $p(x) = \prod_{ij} \exp(-\lambda |x_i - x_j|)$ same neighbors are more probable

p(X)



Samples from the prior for varied lambda:



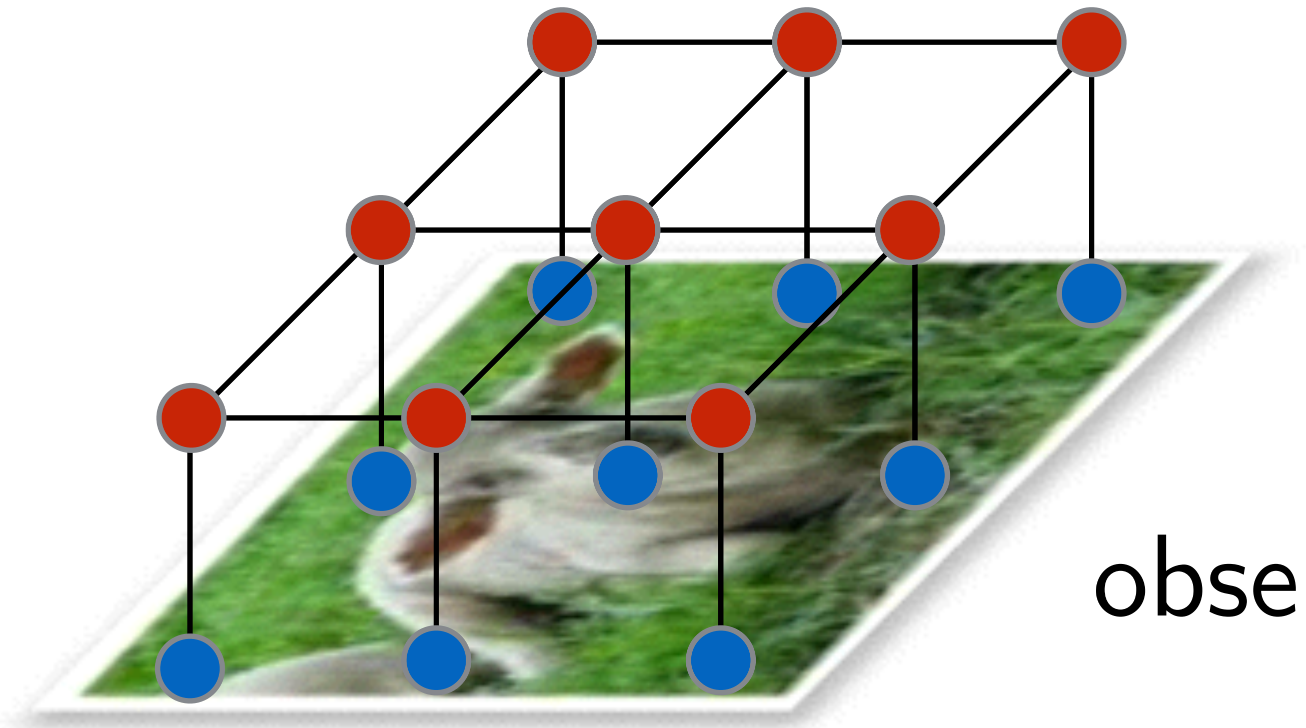
Conditional Random Field

- $x_i, \; i \in V$ - hidden random variables (segmentation)
- $y_j, \; j \in V'$ - observed random variables (Image)

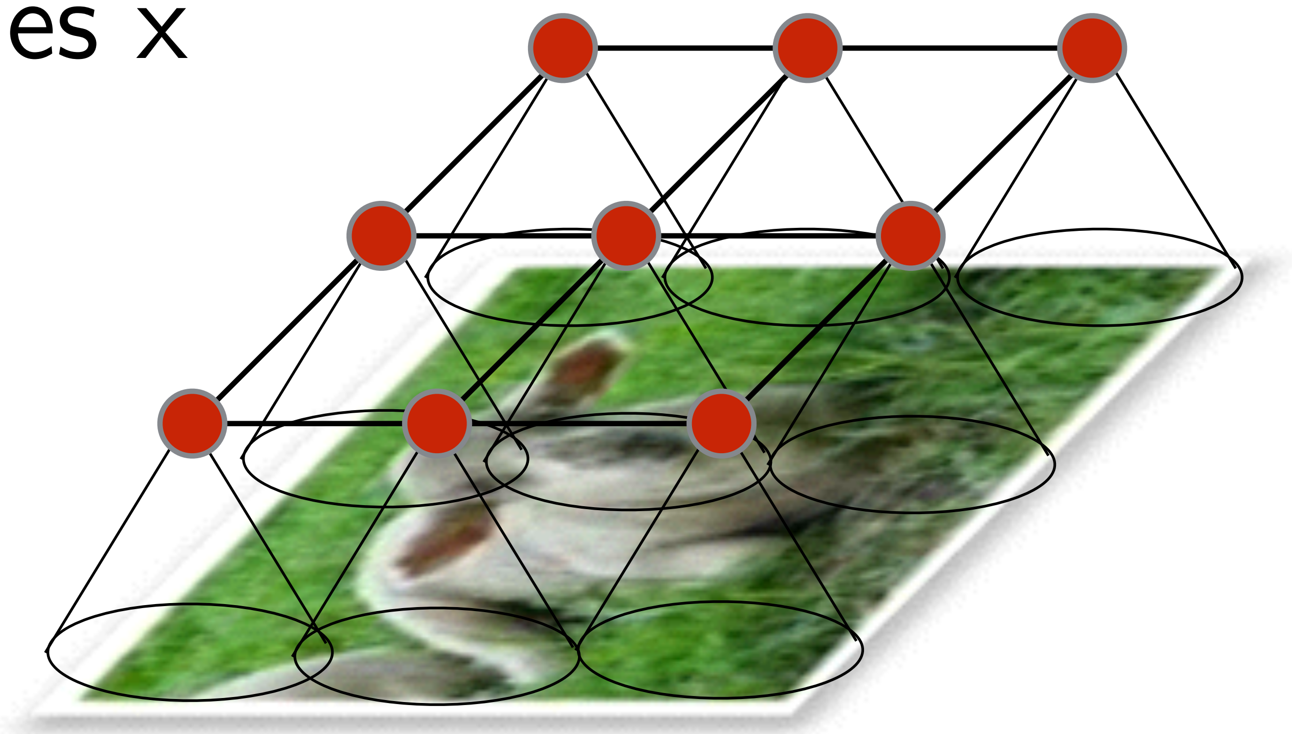
Definition (Lafferty *et al.* 01)

$p(x \mid y)$ is a conditional random field if it satisfies Markov properties w.r.t. x given y.

MRF p(x,y)



CRF p(x|y)



hidden variables x

observed variables y

Generative: $p(y) = \sum_x p(x, y)$

can be learned unsupervised

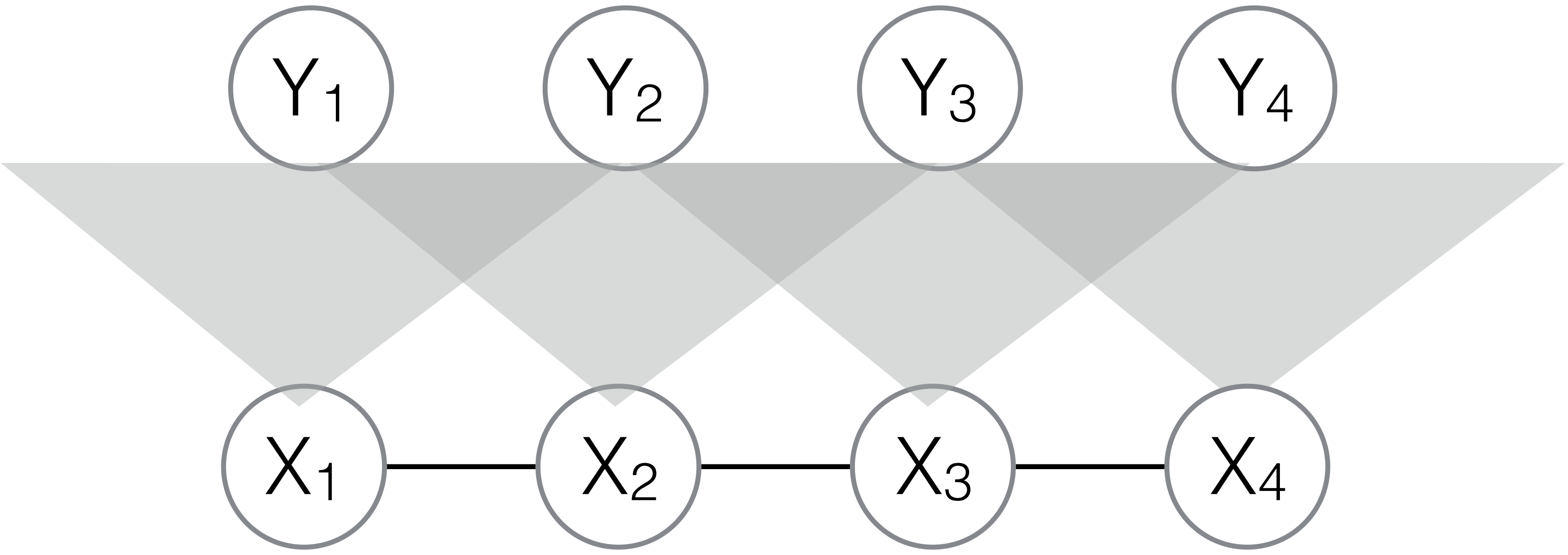
Discriminative, no model of p(y)

more flexible for recognition

Recognition is the same: $\mathrm{argmin}_x \, p(x, y) = \mathrm{argmin}_x \, p(x \mid y)$

Two-class Segmentation

CRF Model

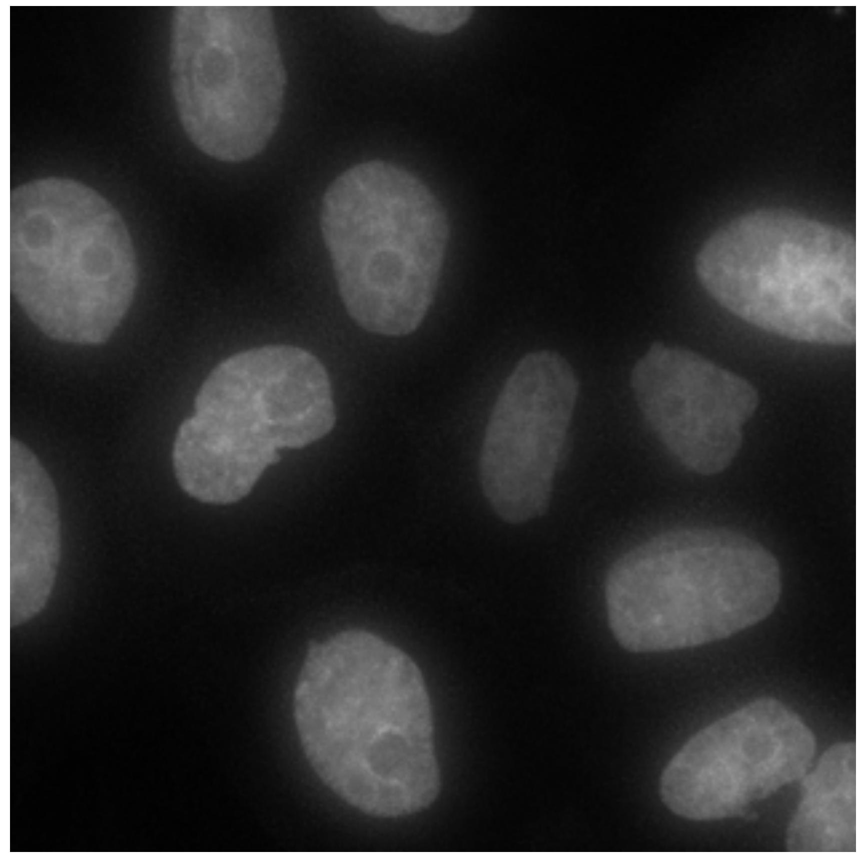


Image

Segmentation

p(X|y) is an MRF

Input Image

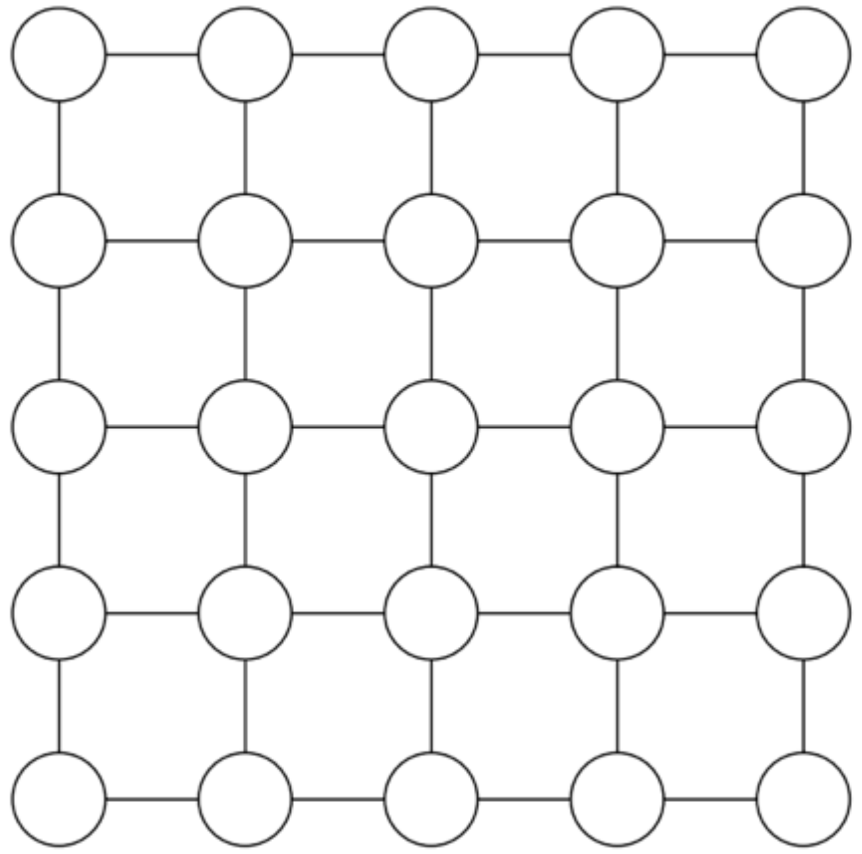


binary segmentation

CRF model: $p(y \mid x) = \prod_i g_i(y \mid x_i)$

$g_i(y|x_i)$ - could be a logistic model, decision tree, boosted classifier, etc.

p(X)



MAP of MRF — Energy Minimization

MAP of MRF

- Given the model $p(x) = \prod_{c \in S} g_c(x_c)$ find the most probable state:

$$\max_x p(x)$$

- Joint maximization in all variables
- Take negative logarithm:

$$\min_x \sum_{c \in S} -\log g_c(x_c) = \min_x E(x)$$

- Partially separable minimization problem, called Energy minimization
- Belongs to discrete optimization domain (combinatorial optimization, graph theory, ILP, relaxations, etc.)
- Many optimization techniques specifically suitable for computer vision

Pairwise Energy Minimization

Common scenario: only pairwise interactions:

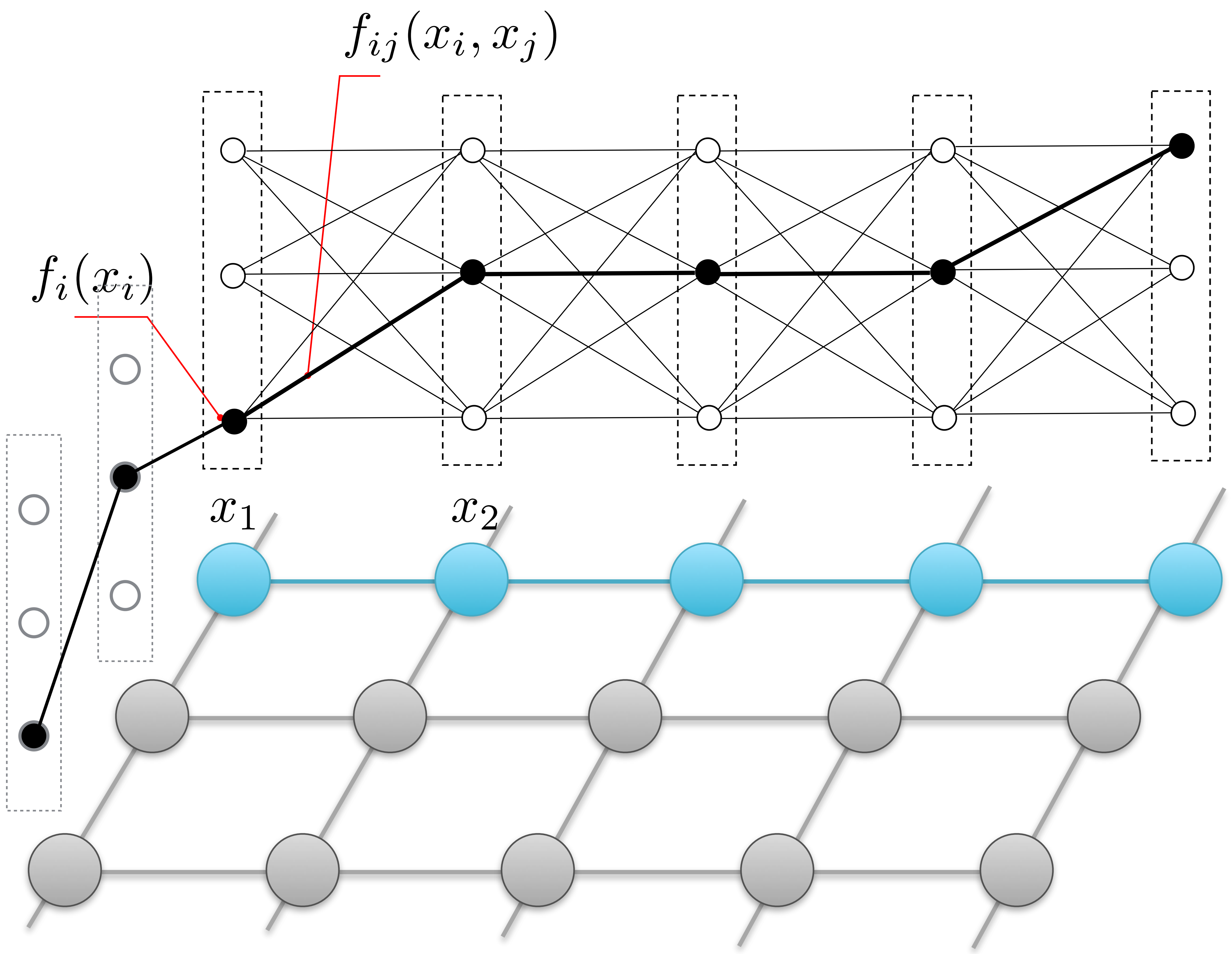
$$\min_{x} \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j)$$

$(\mathcal{V}, \mathcal{E})$ - graph

$\mathcal{V}$ - set of nodes

$\mathcal{E}$ - set of edges

$x = (x_i \mid i \in \mathcal{V})$ - labeling



- NP-hard (includes MAX-CUT, vertex packing, etc.)
- Two large groups of methods used in CV:
  - minimum cut (graph cuts)
  - LP relaxation / message passing
- There are much more

Pairwise Energy Minimization

Common scenario: only pairwise interactions:

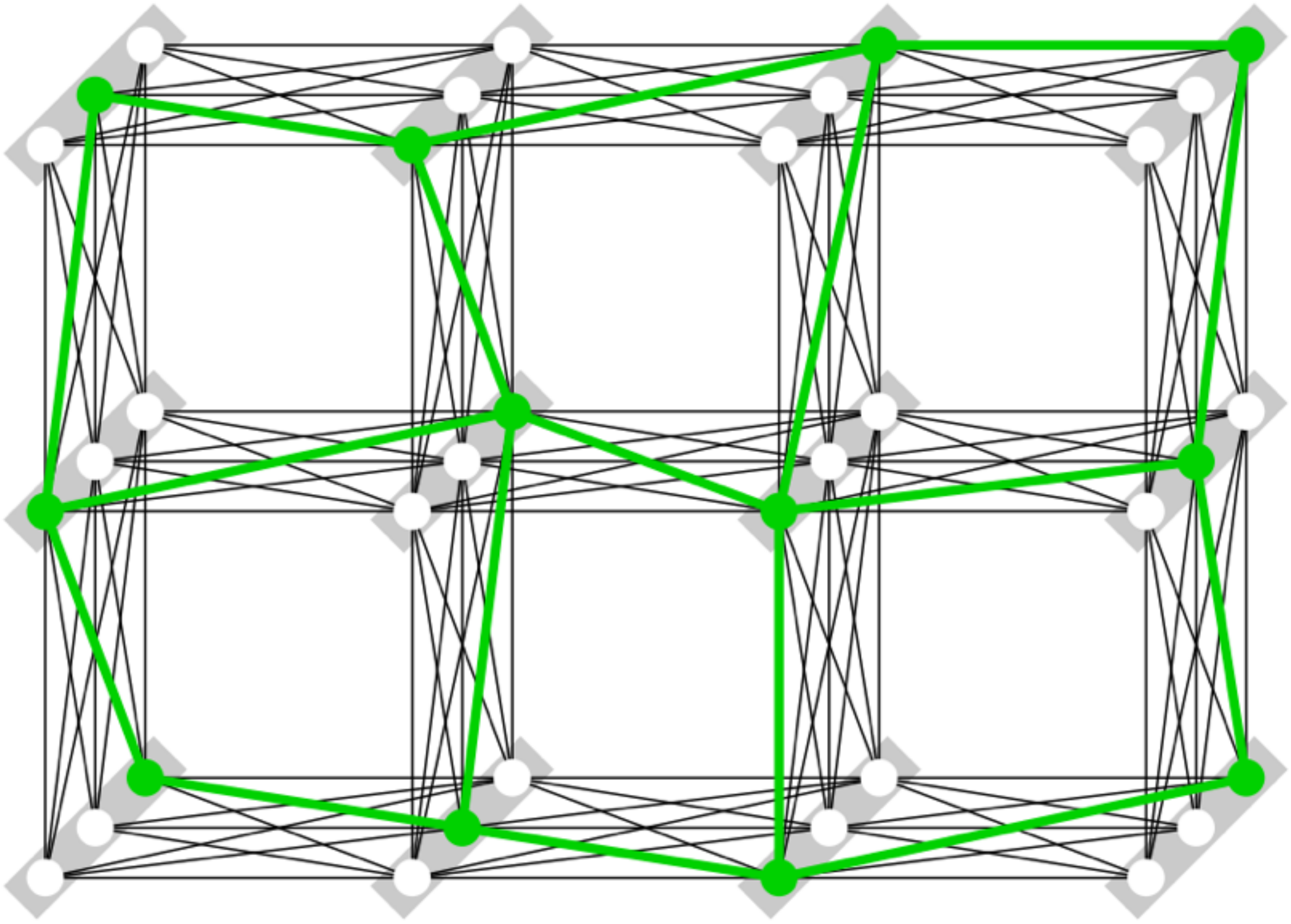
$$\min_x \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j)$$

$(\mathcal{V}, \mathcal{E})$ - graph

$\mathcal{V}$ - set of nodes

$\mathcal{E}$ - set of edges

$x = (x_i \mid i \in \mathcal{V})$ - labeling



- NP-hard (includes MAX-CUT, vertex packing, etc.)
- Two large groups of methods used in CV:
  - minimum cut (graph cuts)
  - LP relaxation / message passing
- There are much more

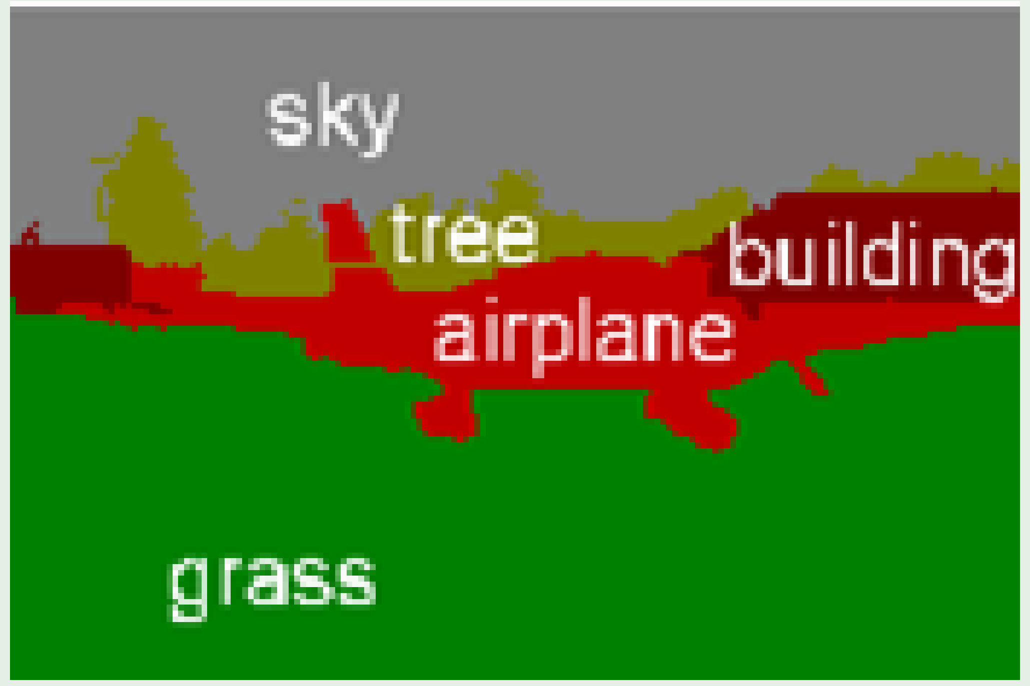
Example: Semantic Segmentation

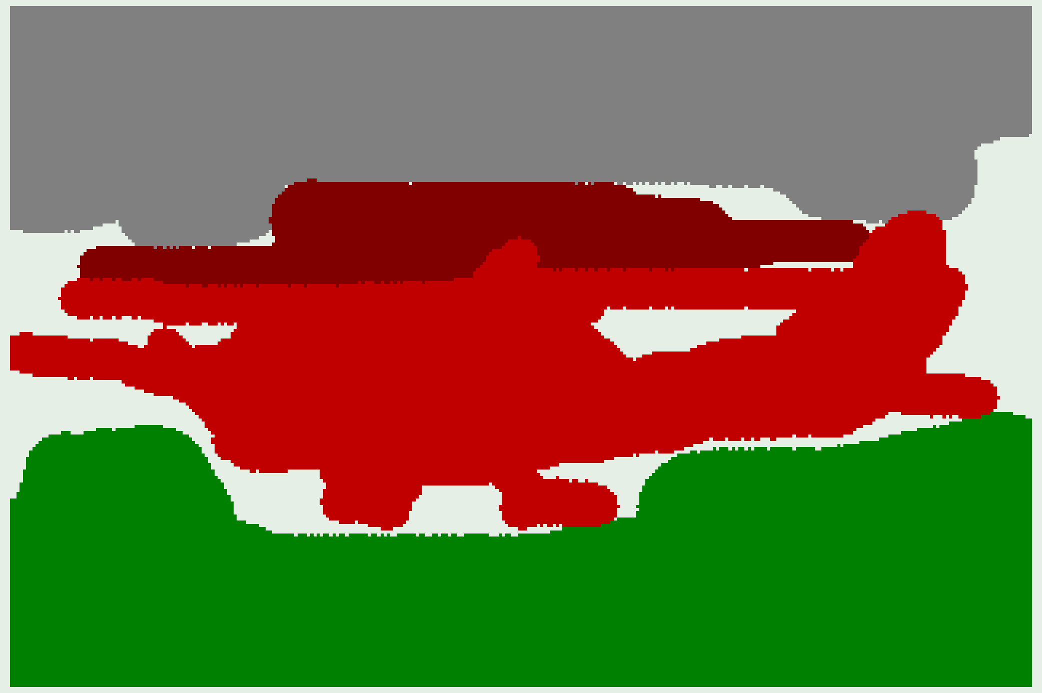
Example: Potts Model for Object Class Segmentation

- $\mathcal{V}$ - set of pixels; $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ neighboring pixels;
- $\mathcal{X}_s = \{1, \ldots K\}$ – class label;
- $E_f(x) = \sum_{s \in \mathcal{V}} f_s(x_s) + \sum_{st \in \mathcal{E}} \lambda_{st} [\![x_s \neq x_t]\!]$.

Image

Ground Truth





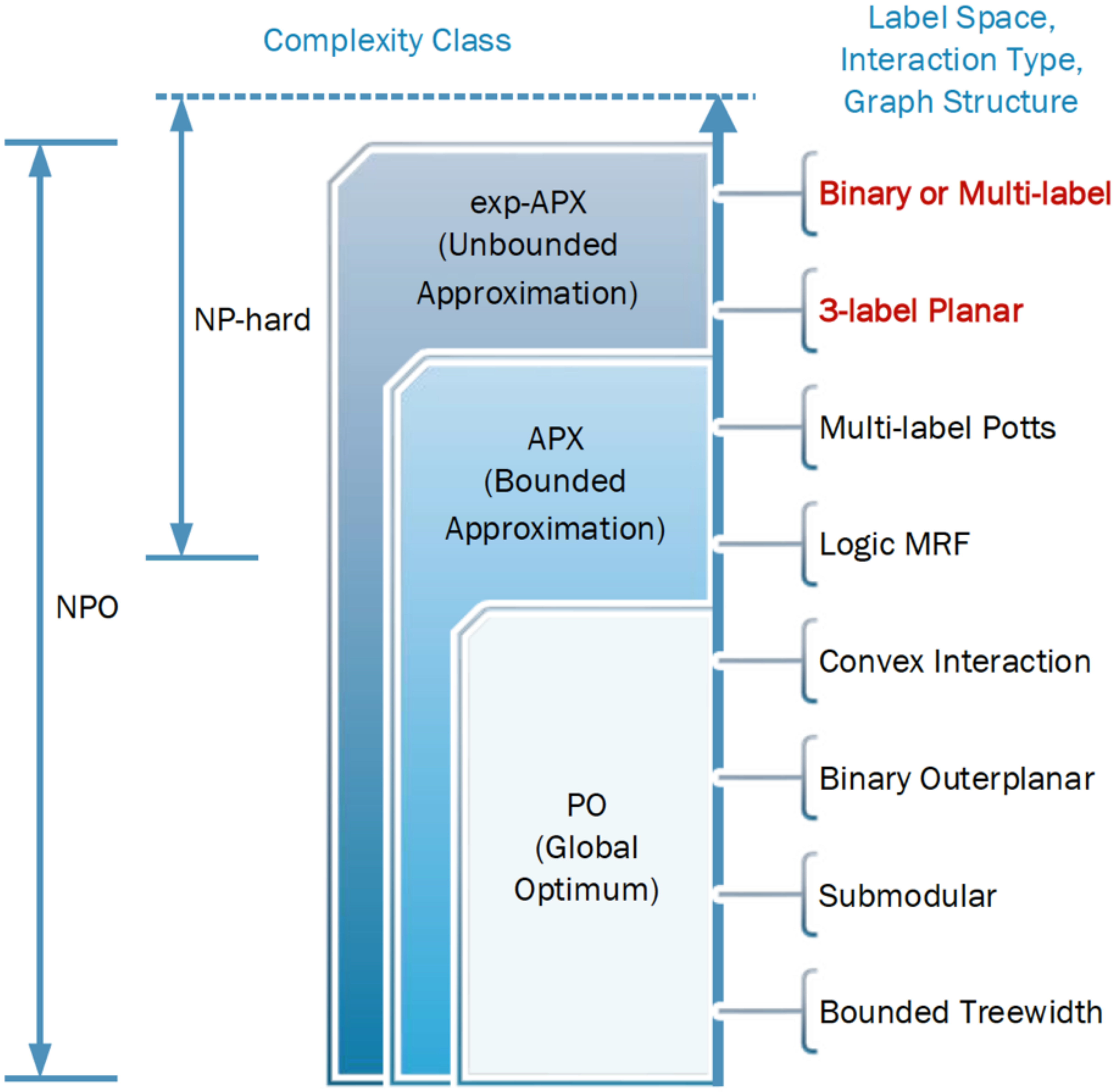
(MSRC object class segmentation)

Complexity of Energy Minimization

Cannot guarantee
$f(x) \leq P(n) f(x^*)$

$f(x) \leq C f(x^*)$

$f(x) = f(x^*)$



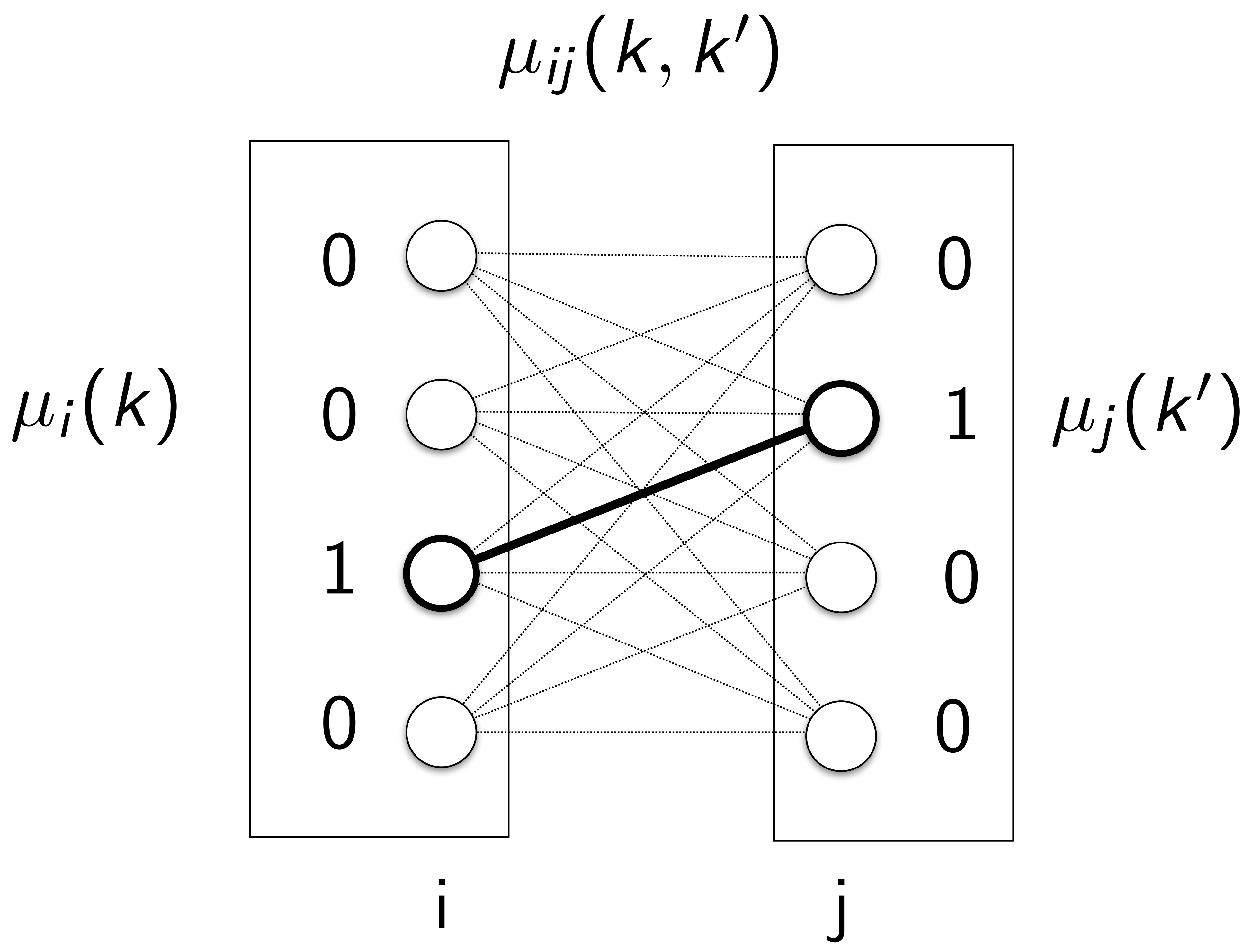
also used in applications but rely on "good " data

Directly used in applications
or as subproblem solvers

Overview in [Li et al. "Complexity of Discrete Energy Minimization Problems", 2016]

As Integer Linear Program

- Energy minimization: $\min_x \sum_i f_i(x_i) + \sum_{ij} f_{ij}(x_i, x_j)$
- For each i encode x_i with $\mu_i(k) \in \{0,1\}$, k – label
- For each ij encode (x_i, x_j) with $\mu_{ij}(k,k') \in \{0,1\}$
- The objective linearizes
- μ need to respect constraints



$$\min_{\mu} \sum_i \sum_k E_{f\,i}(k)\mu_i(k) + \sum_{ij} \sum_{k,k'} E_{f\,ji}(k,k')\mu_{i,j}(k,k')$$

$\mu \geq 0; \quad \mu \in \{0,1\}^{\mathcal{I}}$

$\sum_k \mu_i(k) = 1$

$\sum_{k,k'} \mu_{ij}(k,k') = 1$

$\sum_{k'} \mu_{ij}(k,k') = \mu_i(k)$

$\sum_k \mu_{ij}(k,k') = \mu_j(k')$

The Power of Basic LP Relaxation

- Consider a class C of problems specified by unrestricted graph structure and pairwise potentials from some set F.

Theorem (Thapper and Zivny 2012, Kolmogorov 2013)

(Roughly) Class C has a polynomial time algorithm **iff** *the Basic LP relaxation is tight for C.*

- This means LP relaxation is a rather universal tool
- It is also tight for many practical individual instances or provides a good approximation

Theorem (Prusa, Werner, 2017)

LP Relaxation of MAP MRF is as hard as any linear program. (Already for Potts model with 3 labels on a planar graph).

- It means it is very unlikely to come up with an algorithm better than $O(n^{3.5}L)$
- Many approximate methods developed in Computer Vision

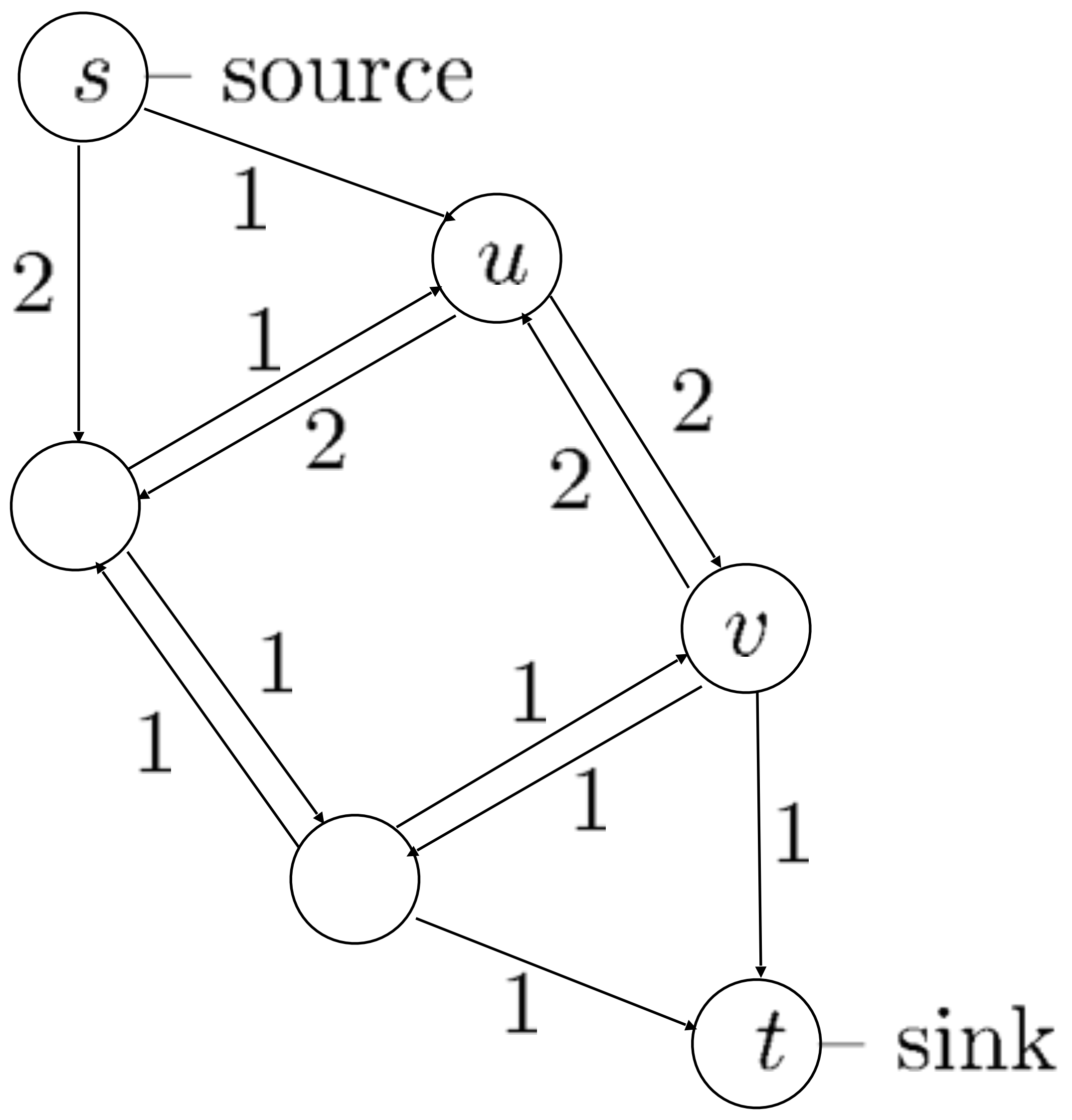
Minimum s-t Cut

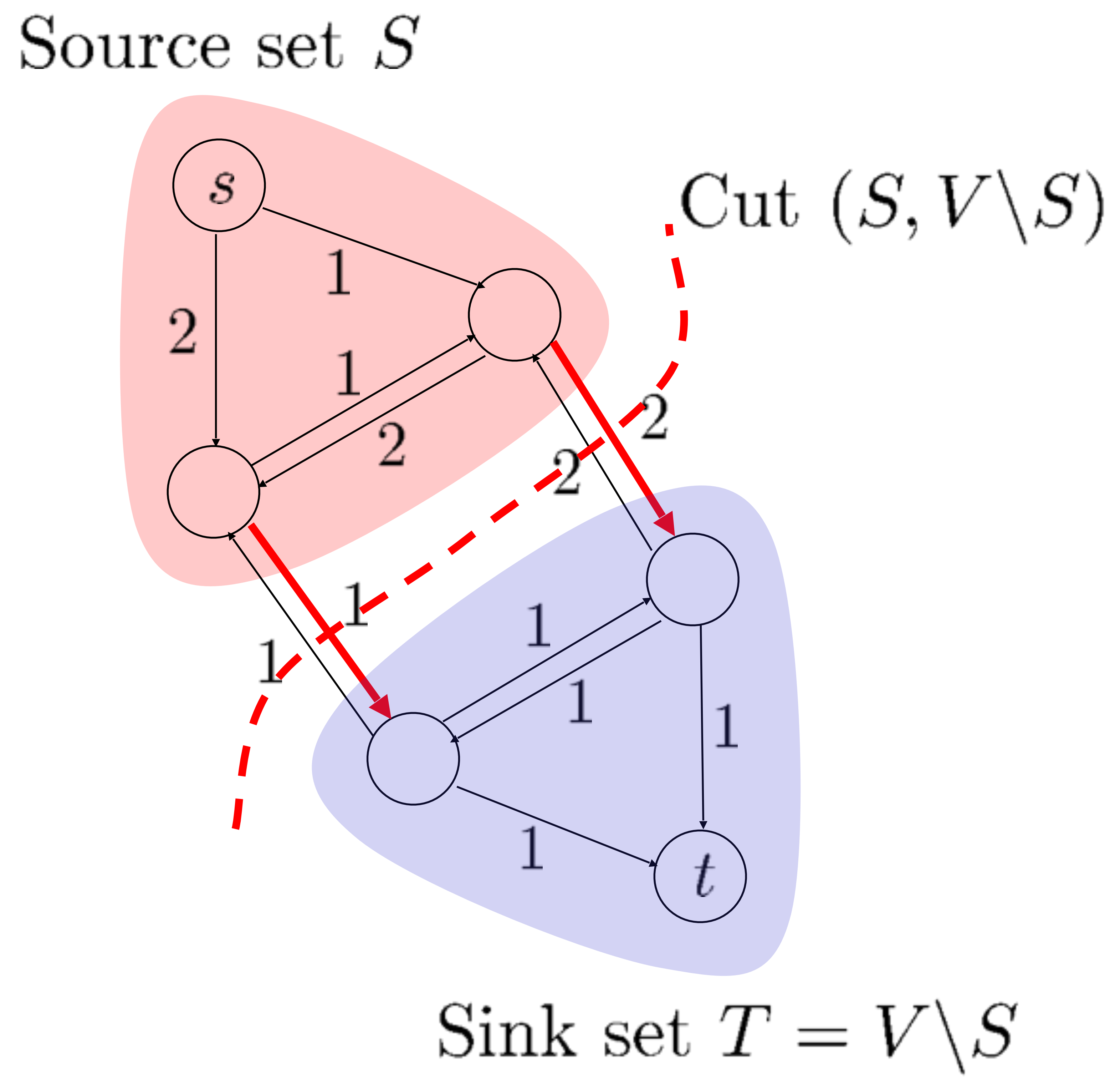
Capacitated network

$G = (V, E, c)$,

$c(u,v) \geq 0$ – arc capacities

Cut cost: $\sum_{\substack{(u,v)\in E \\ u\in S \\ v\notin S}} c(u,v) \rightarrow \min_{\substack{S \\ s\in S \\ t\notin S}}$





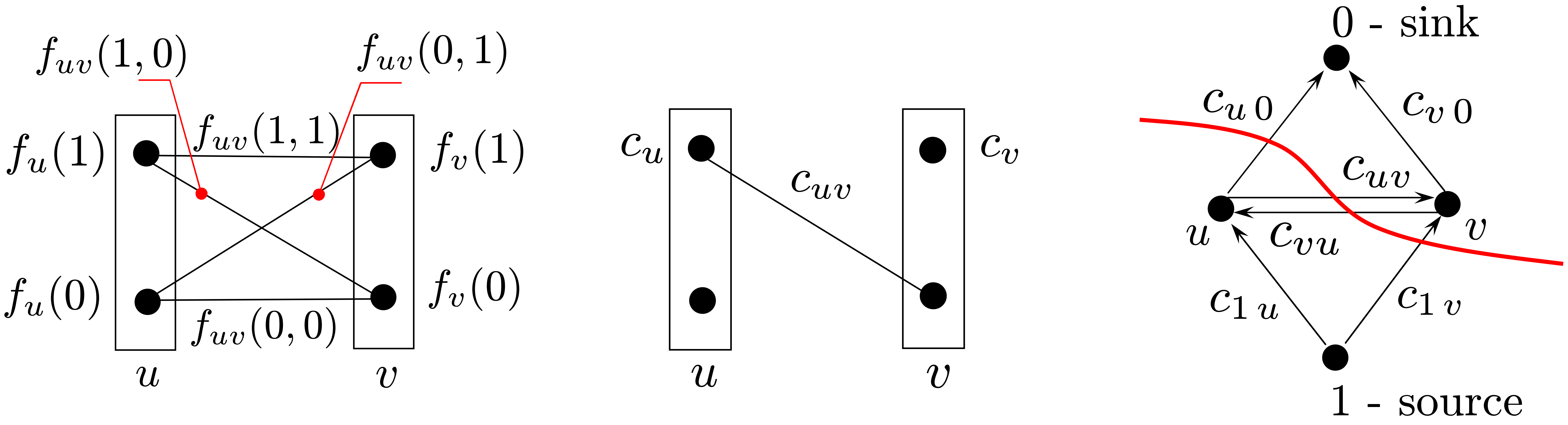
- Problem history: 30+ years
- Active research for better algorithms:
  - theoretical (Orlin'12: O(mn) algorithm), parallel algorithms
  - practical, esp. in computer vision

Reduction to Minimum s-t Cut

- Let $x_i \in \{0,1\}$
- Energy minimization: $\min_x \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j)$
- Expand as polynomial:

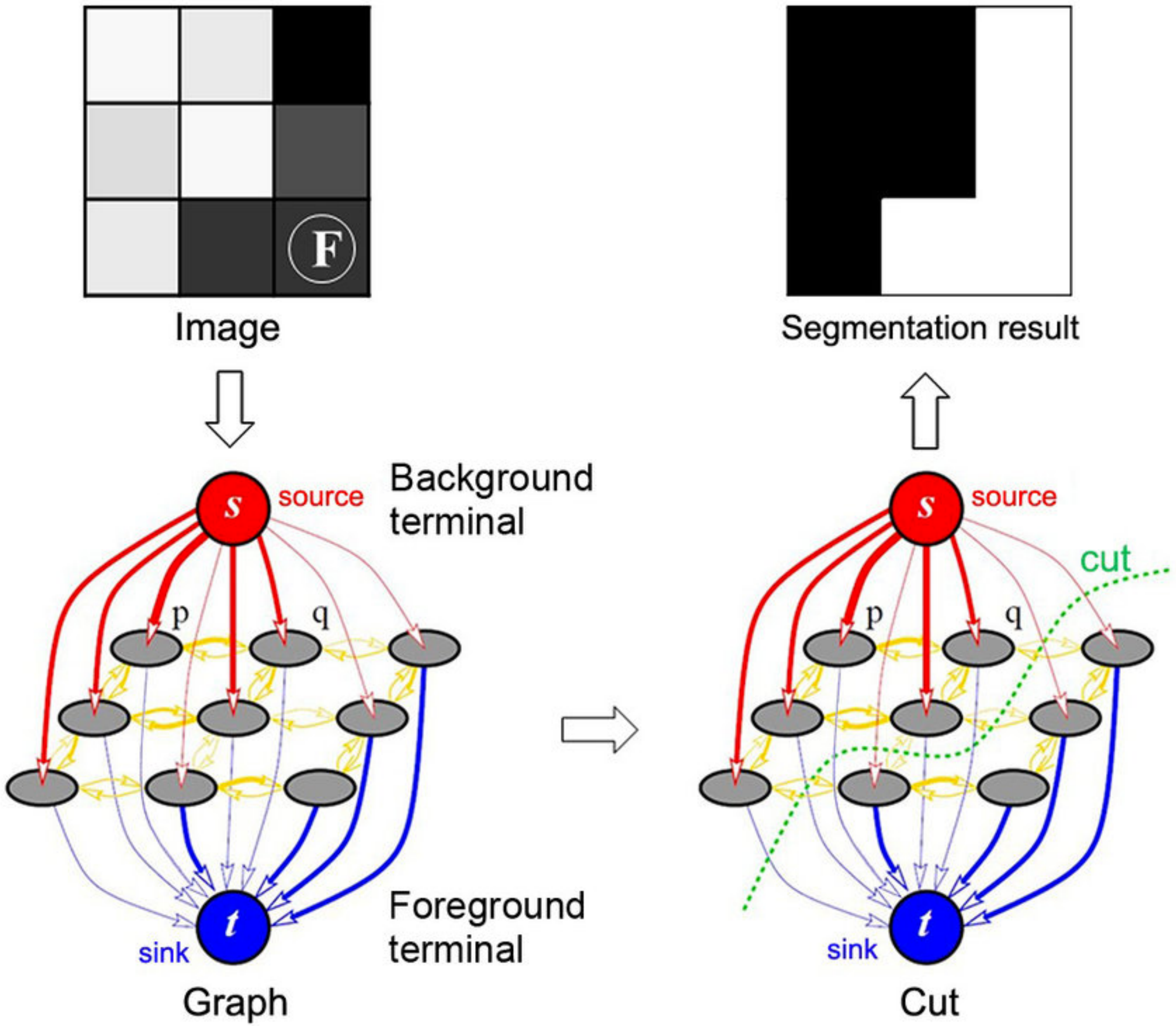
$$\begin{aligned} f_i(x_i) = f_i(1)x_i + f_i(0)(1 - x_i) \quad &= c_0 + c_i x_i; \\ f_{ij}(x_i, x_j) = \ldots \quad &= c_0' + c_i' x_i + c_j'' x_j + c_{ij} x_i (1 - x_j). \end{aligned}$$

- Minimum cut: $\min_{S \subset V} \sum_{ij \in (S, V \setminus S)} c_{ij}$



- Solvable in polynomial time if $c_{uv} >= 0$

Segmentation as Mincut



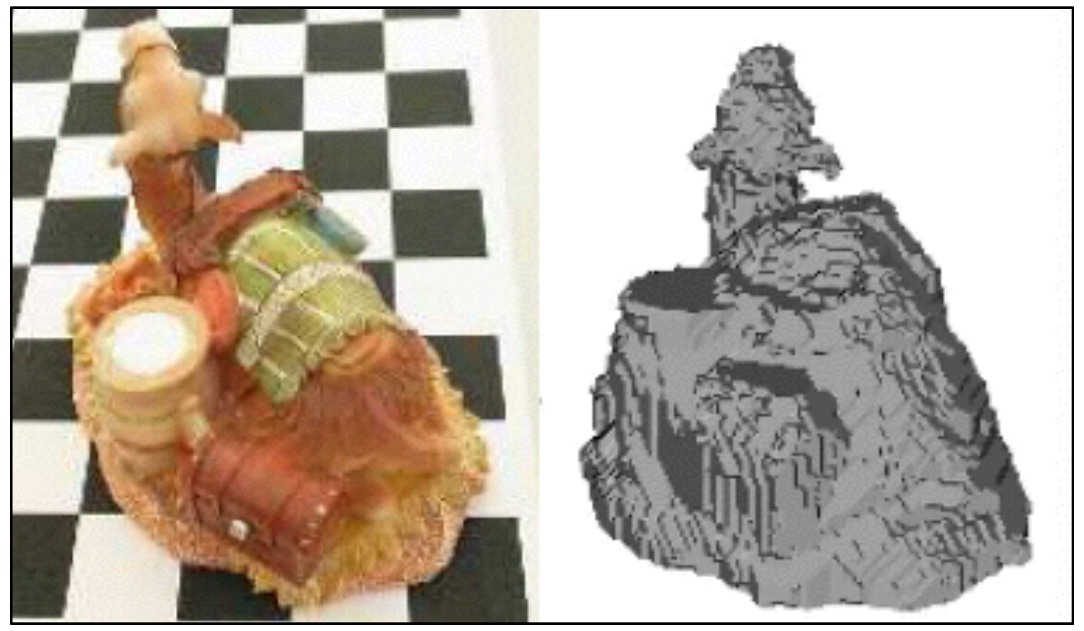
Exercise

Recall the segmentation model: $f_{ij}(x_i, x_j) = \lambda|x_i - x_j|$, $x_i x_j \in \{0, 1\}$

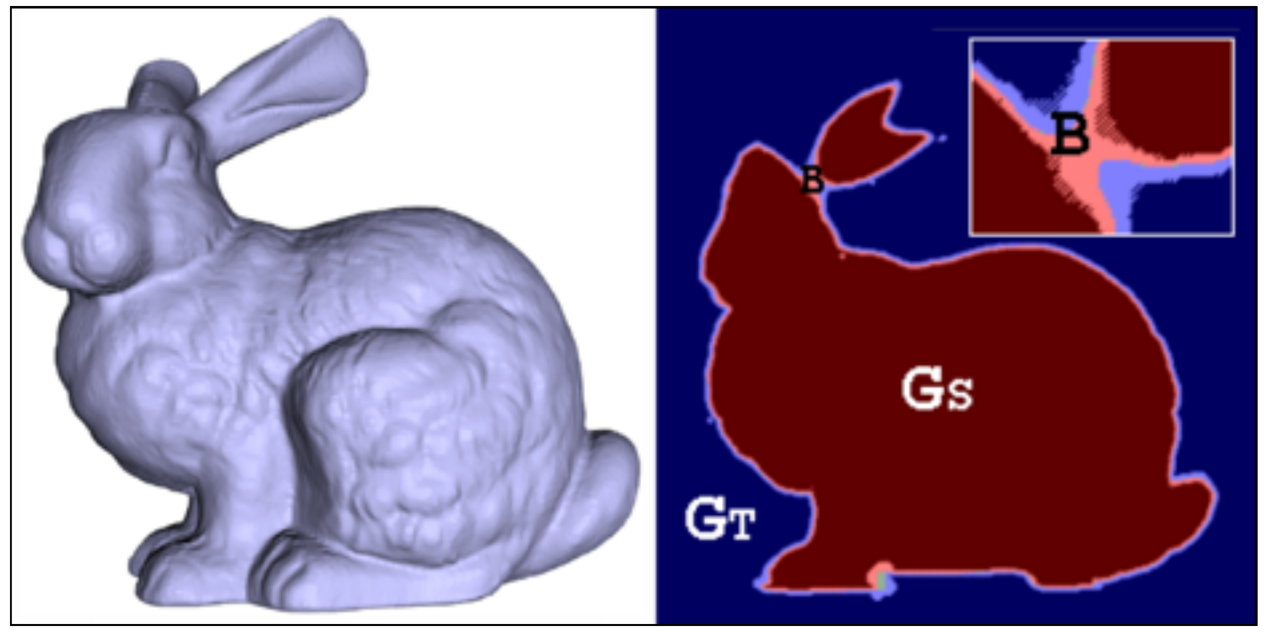
Derive c_{ij} such that f_{ij} expresses as

$c_0 + ax_i + bx_j + c_{ij}x_i(1 - x_j)$

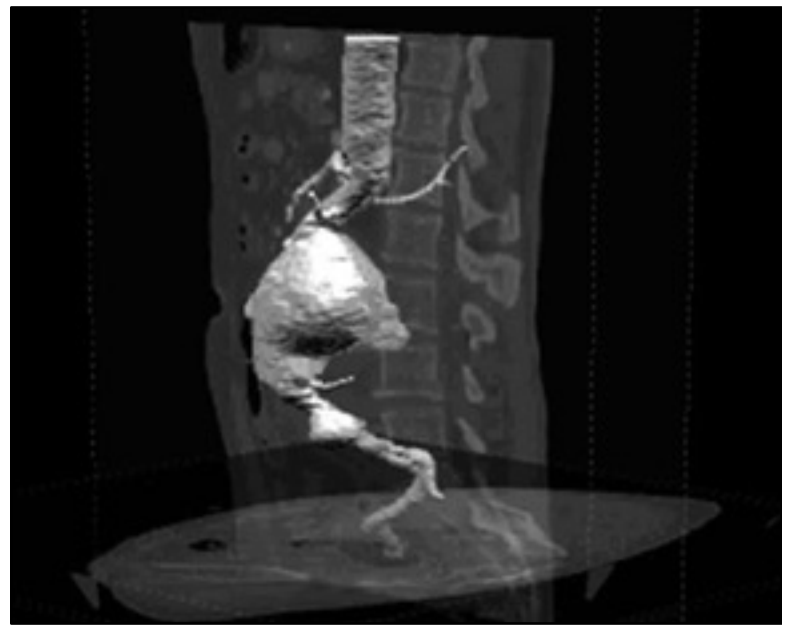
Applications of min-cut

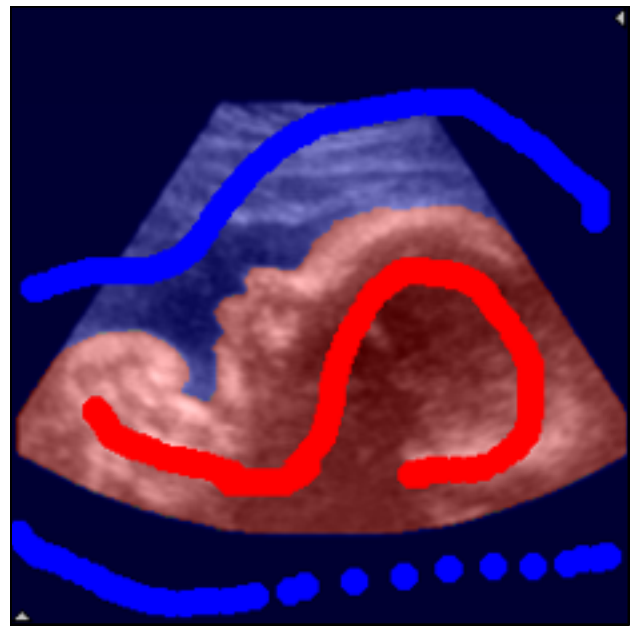


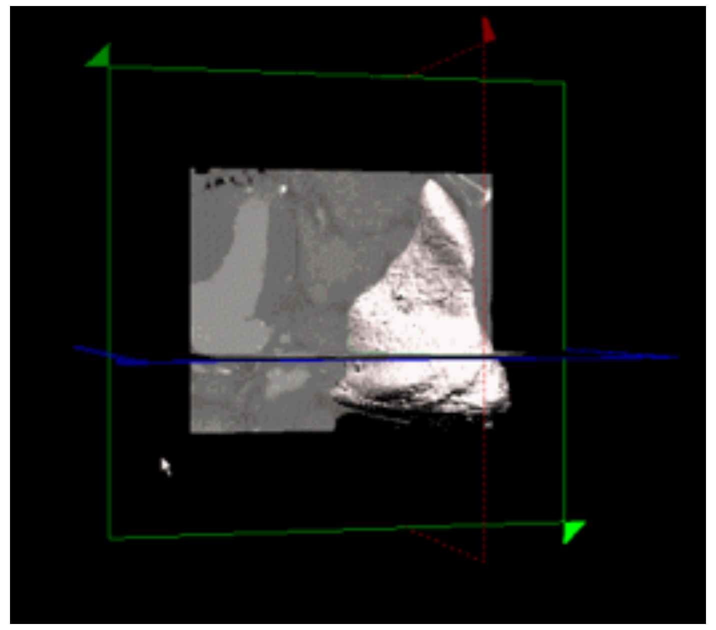
Multiview Reconstruction
Lempitsky et al. 2006
Boykov and Lempitsky 2006

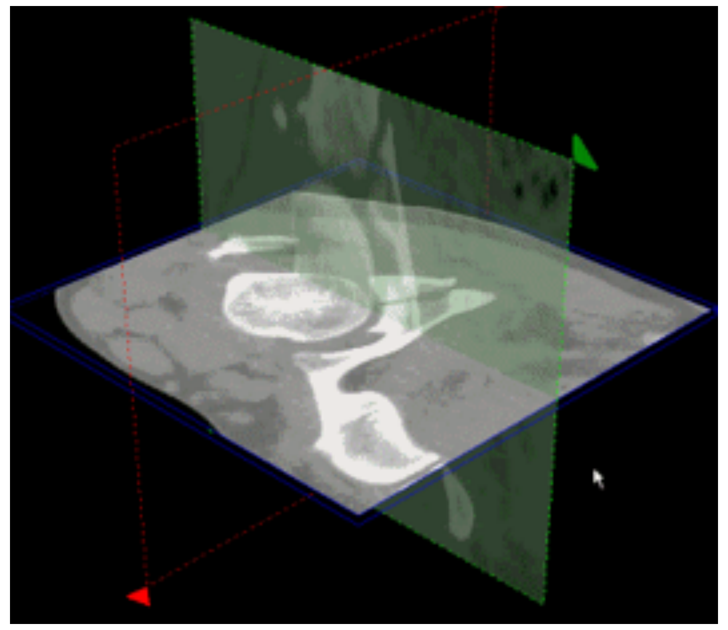


Surface Fitting
Lempitsky and Boykov 2007

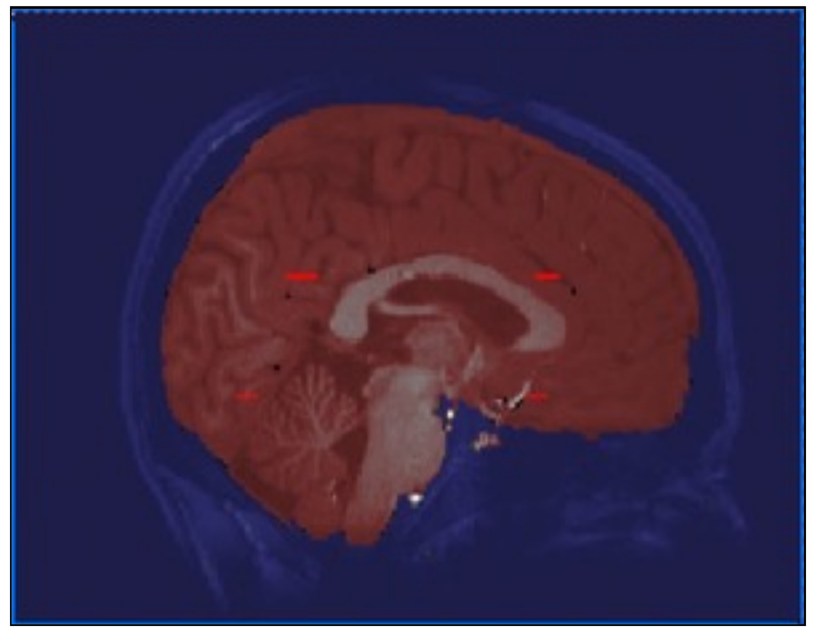




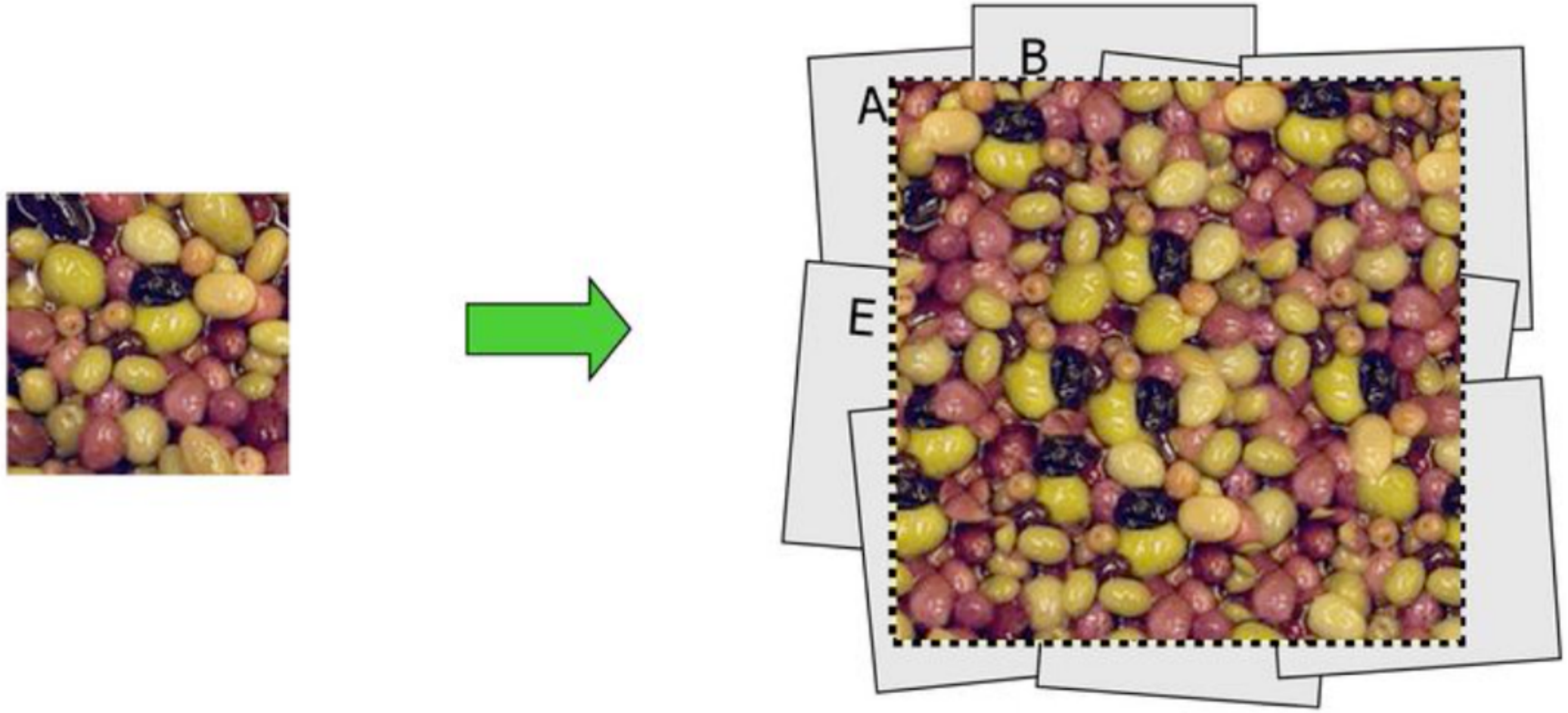


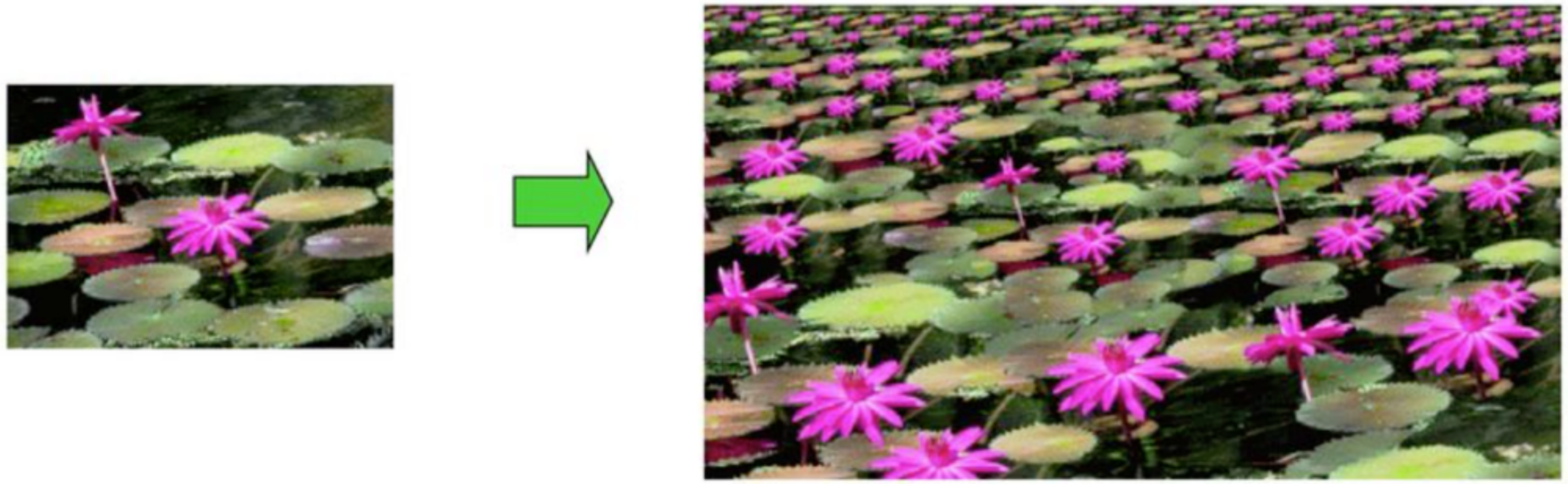


3D Segmentation
Boykov and Joly 2001
Boykov and Funka-Lea 2006
Boykov and Kolmogorov 2003



(More with further extensions)


A
B
E

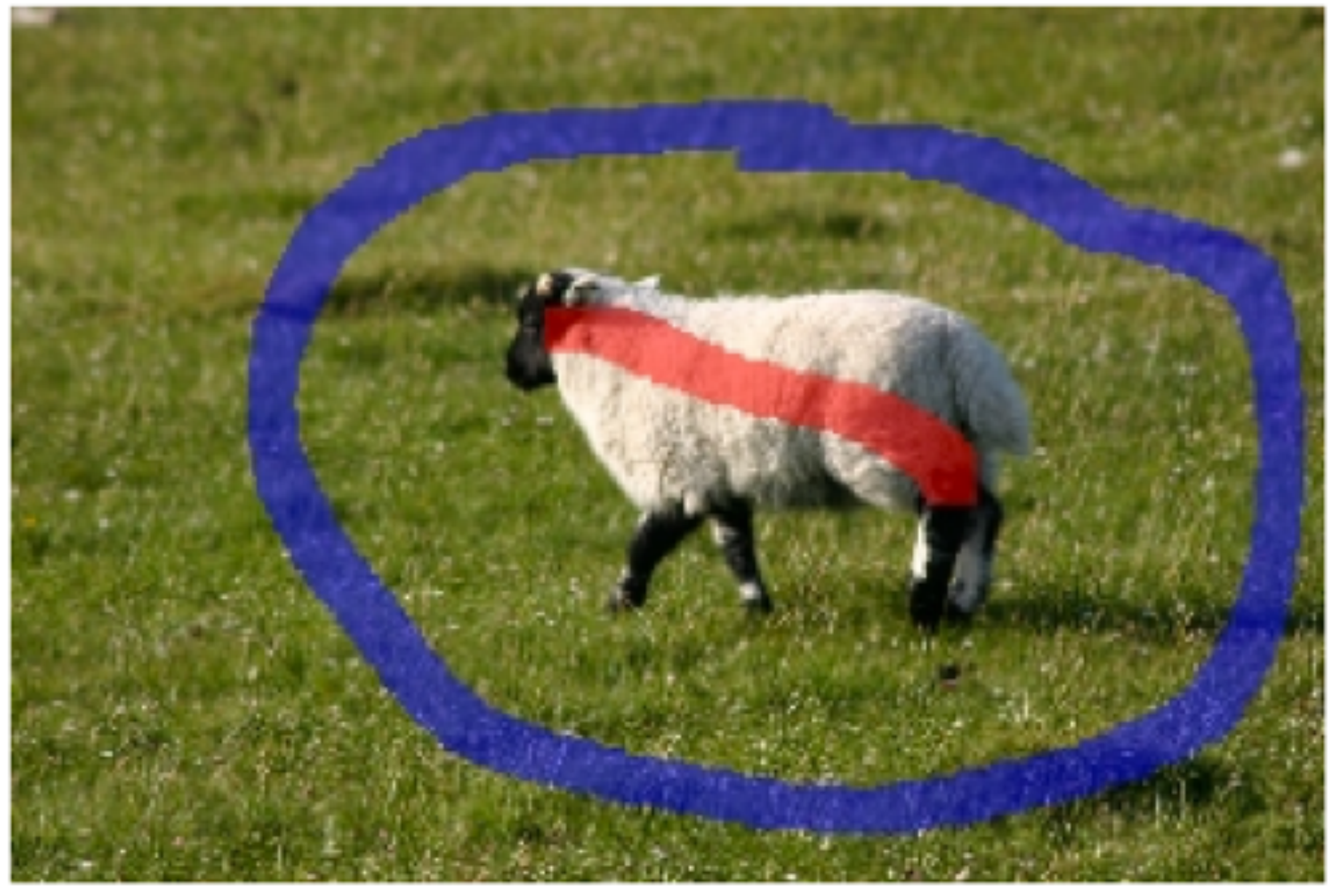


Example: Joint Segmentation and Parameter Estimation

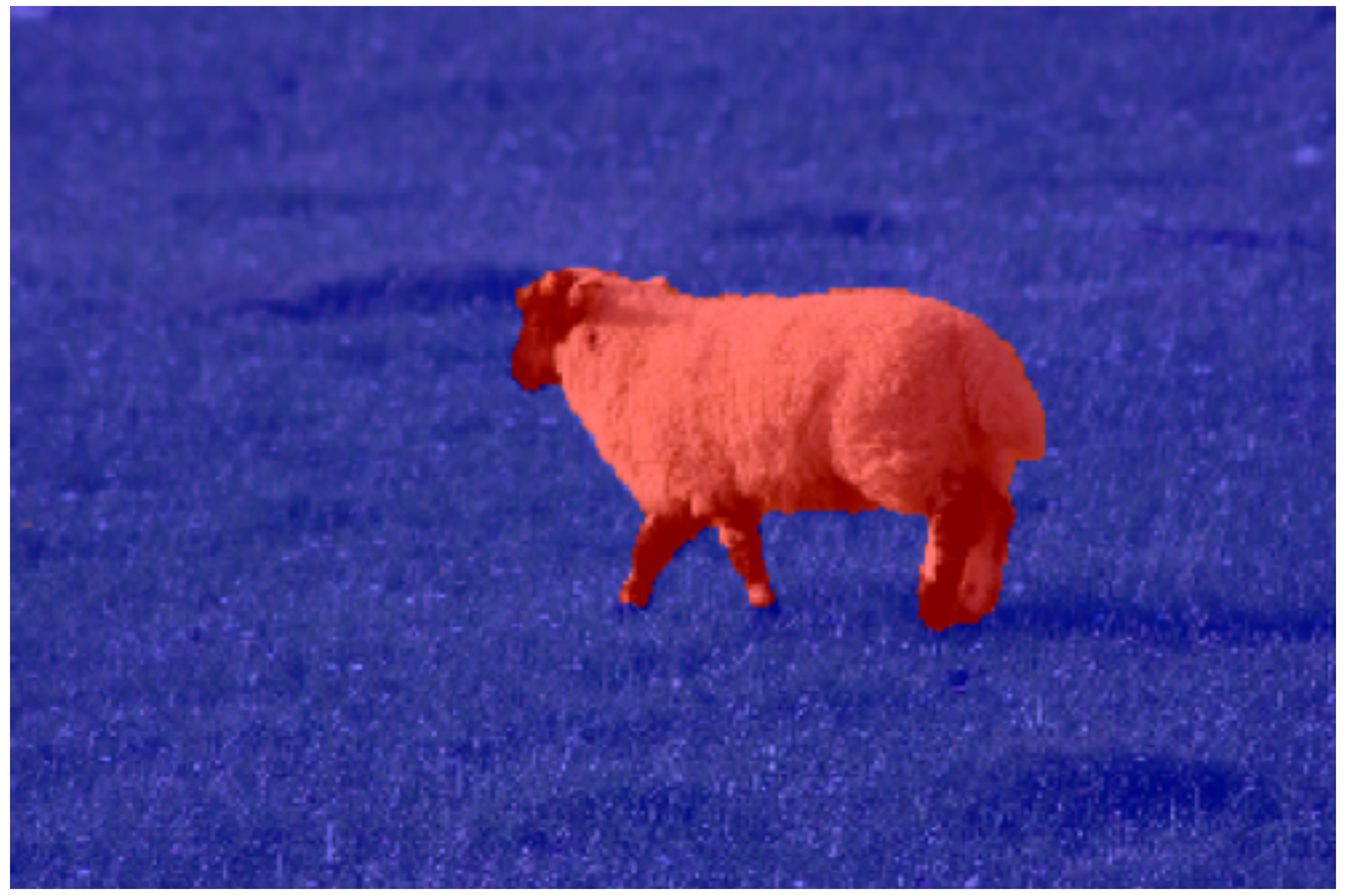
- Input:

Image

FG / BG brush

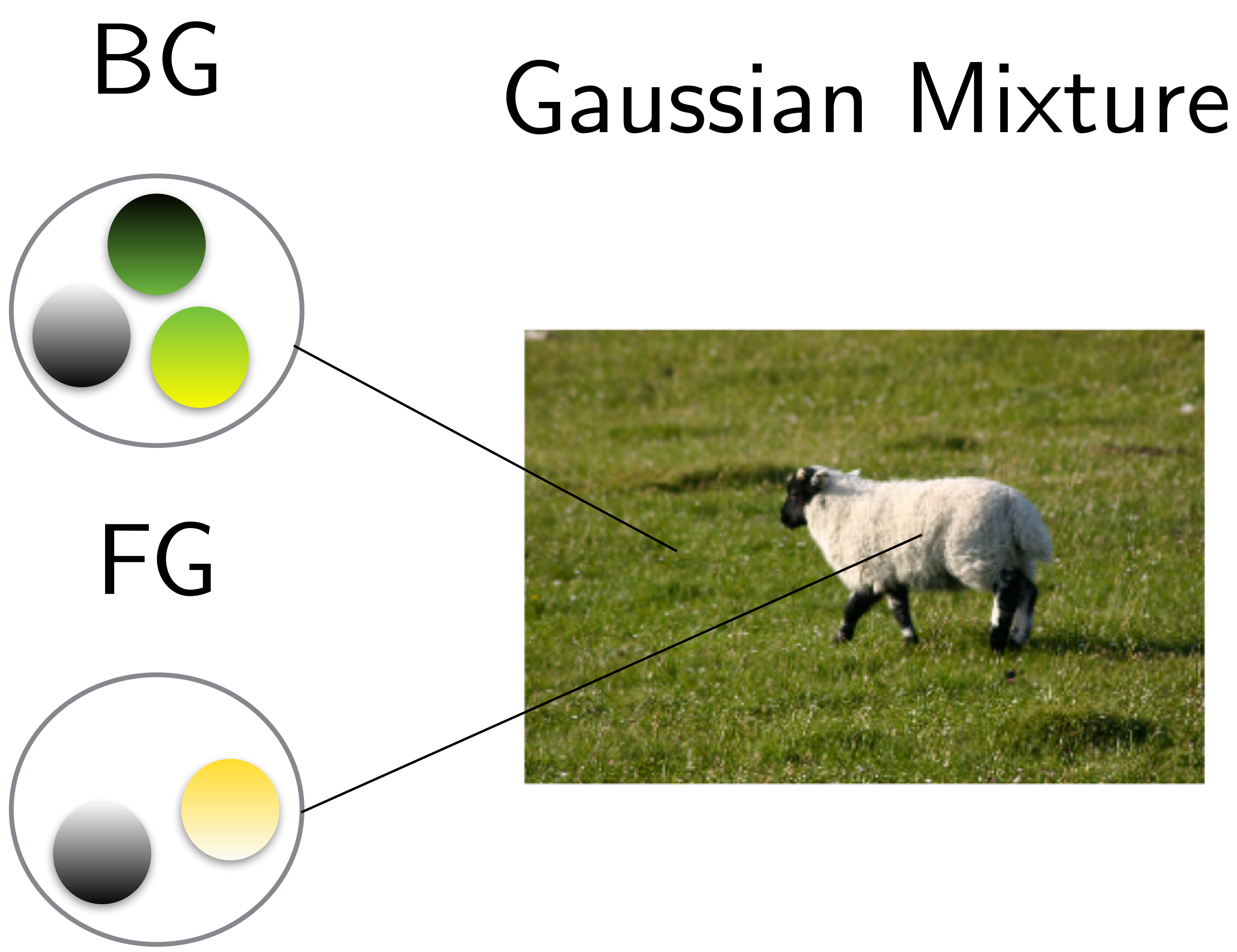


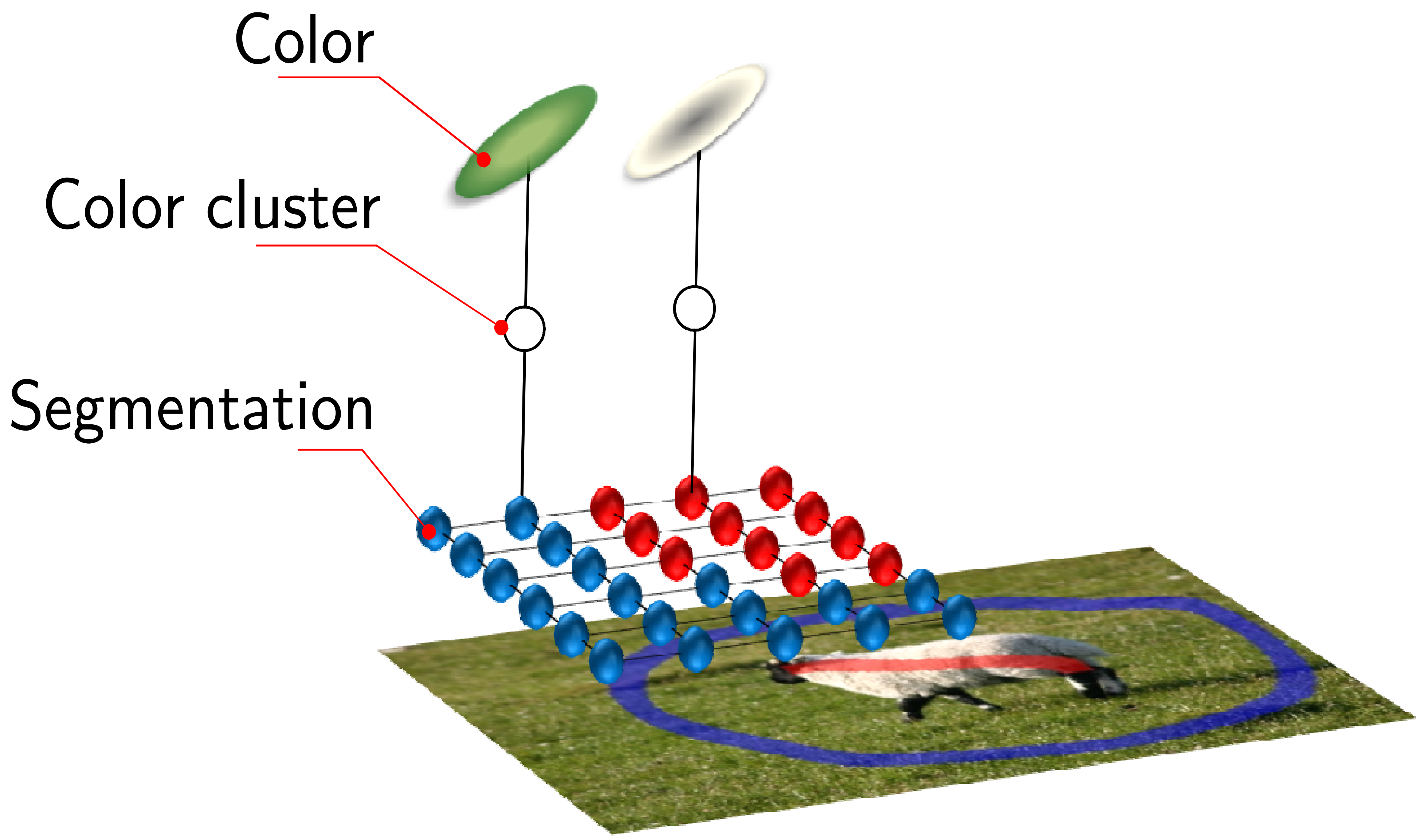
- Output:
  - Complete segmentation



Rother, Kolmogorov, Blake: "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Model





- Markov random field (generative) model:
- Segmentation $x\colon \Omega \to \{0,1\}$
  - Model: $p(x)$ - neighboring pixels are more likely to take the same segment
- Color clusters: $k\colon \Omega \to \{1,\dots K\}$
  - Model: $p(k|x)$ - conditionally independent for all pixels
- Image: $I\colon \Omega \to \mathbb{R}^3$ - color drawn from a color cluster
  - Model: $p(I|k)$ - conditionally independent for all pixels

Method

- Given appearance model find best segmentation (min-cut)
- Given segmentation refit the appearance model

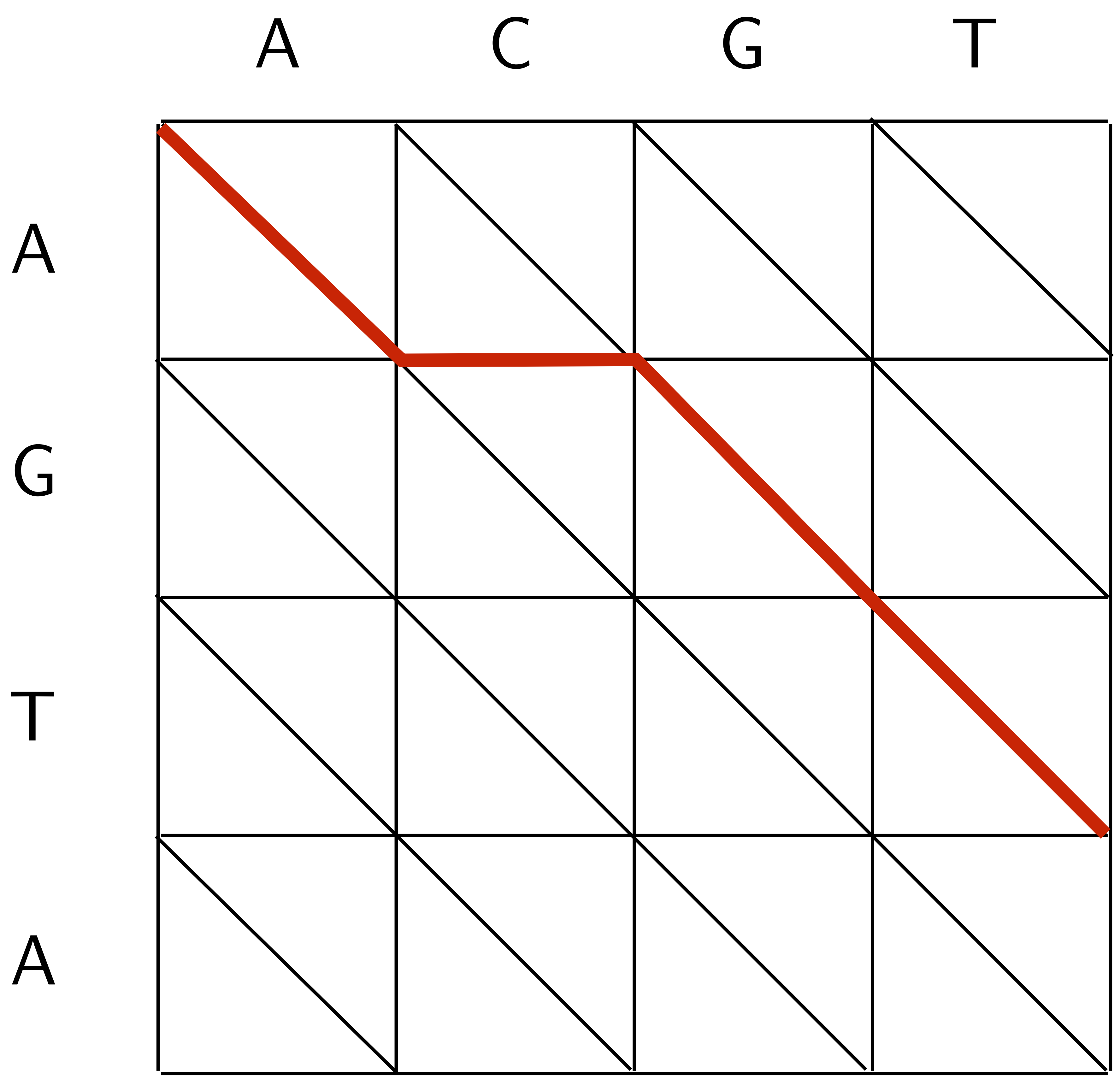
- Problem: fitting a Gaussian mixture is not closed form, may oscillate or get stuck
- Solution: Expectation Maximization algorithm

Stereo as Mincut

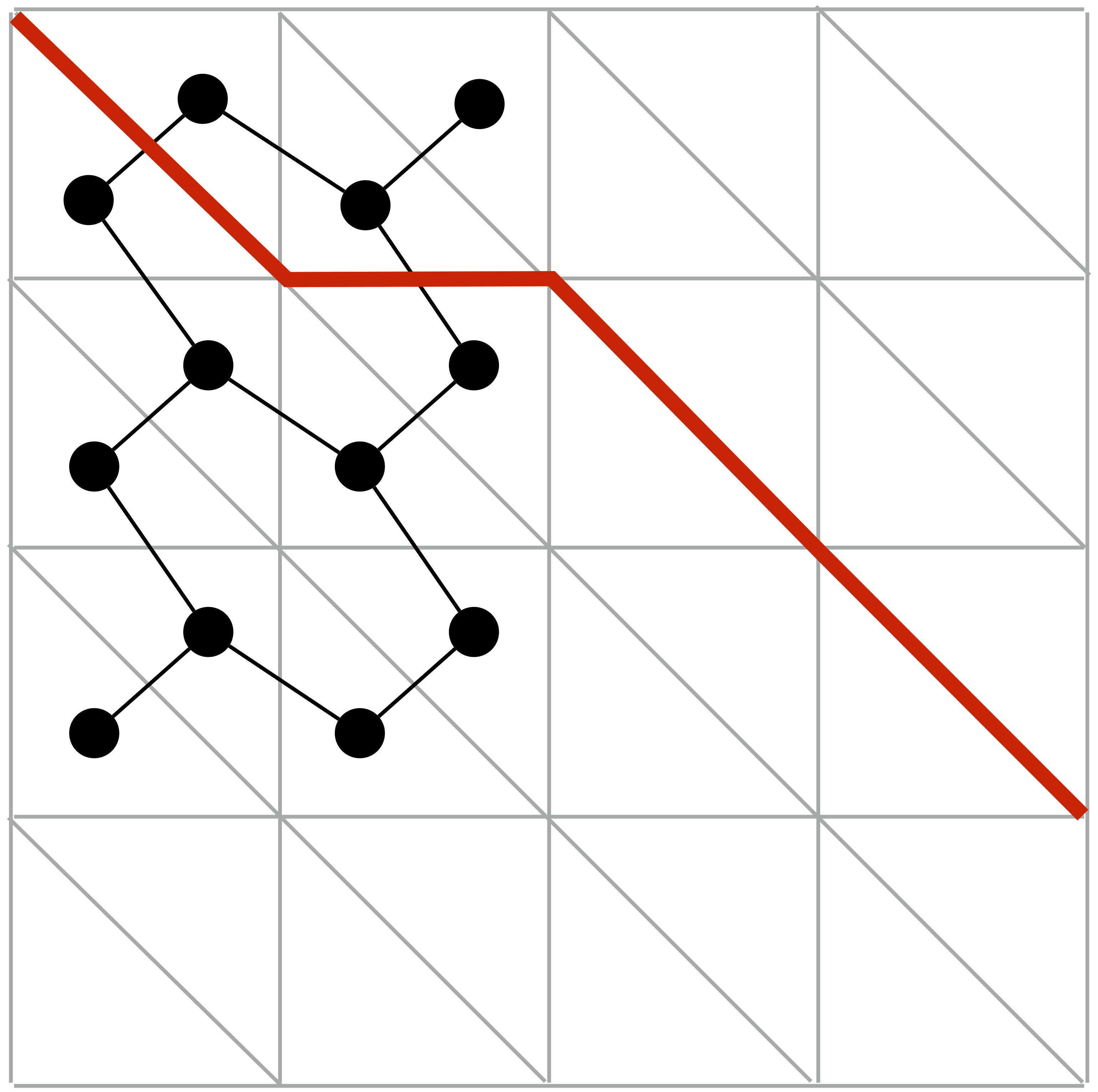
Sequence Alignment problem (bioinformatics), Needleman–Wunsch algorithm (1970)

Also good for scan-line stereo!

Shortest Path

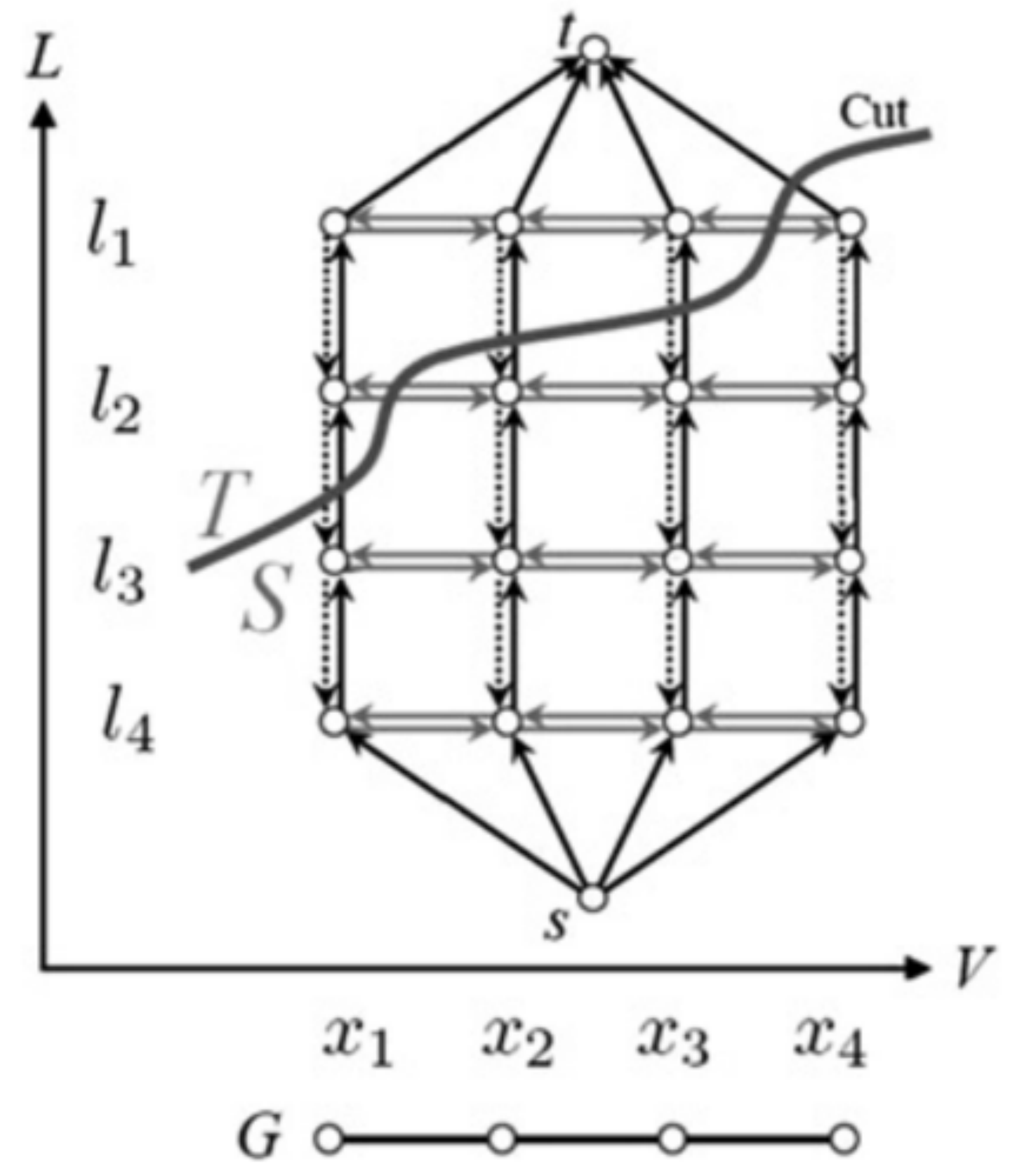


Minimum Cut — extends to surfaces



Hard to construct directly (one CV paper did)

Many other Problems Solvable with Min-cut

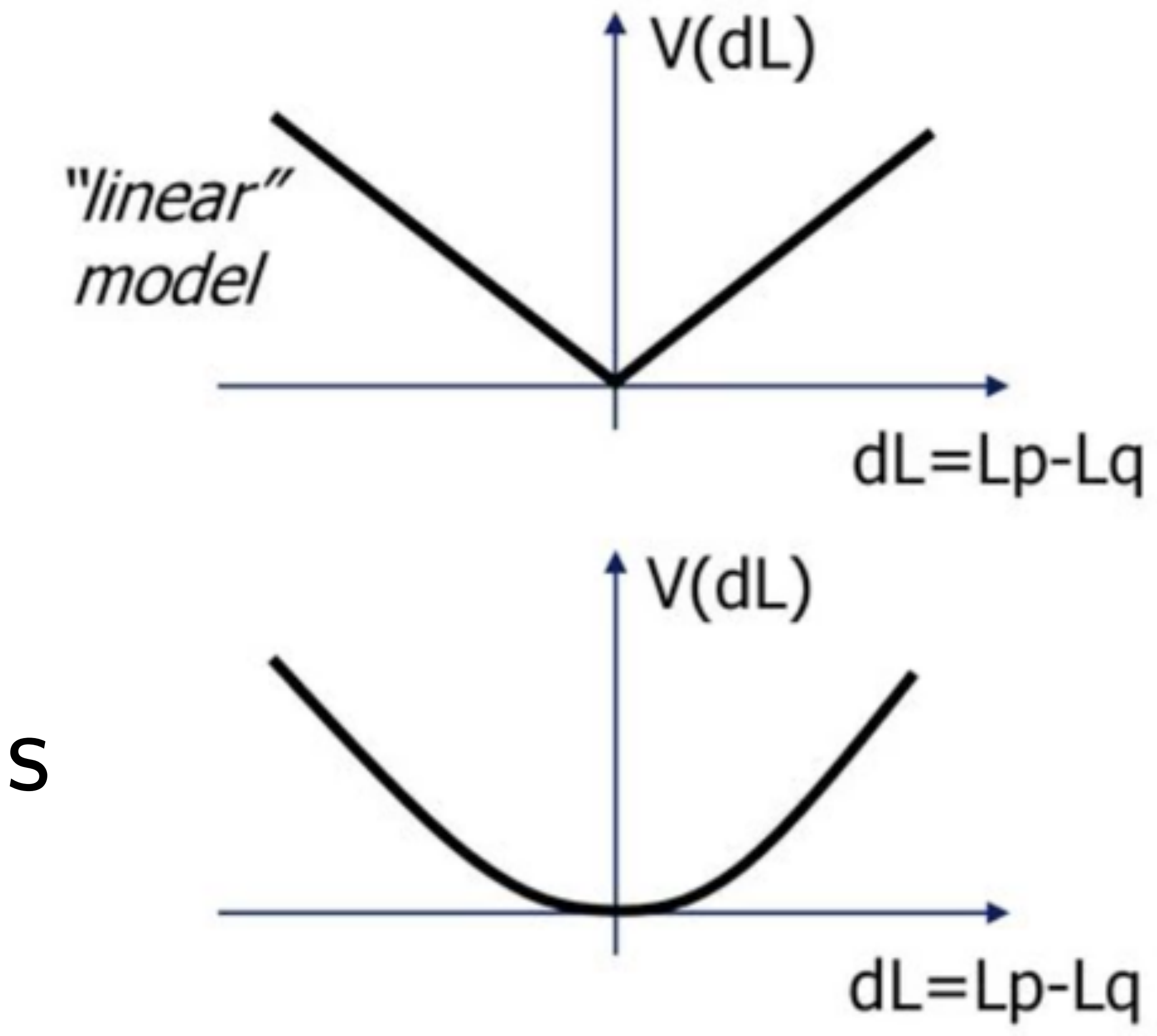


$f_{i,j}(x_i, x_j) = V(x_i - x_j)$, convex

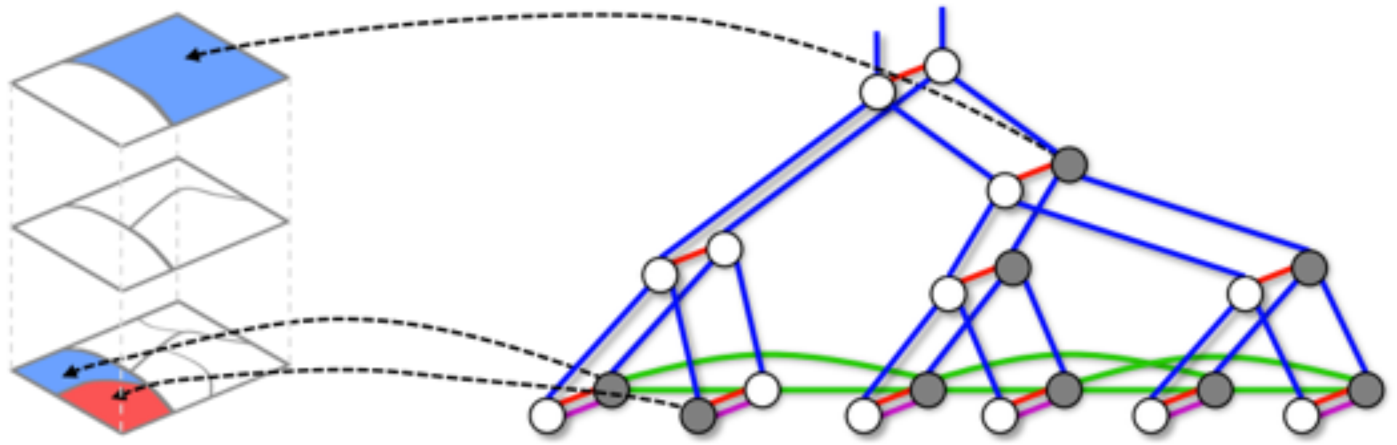
Moregenerally, *submodular*

... Class of

graph-cut representable problems



Multi-class segmentation for a hierarchy of nested candidate regions



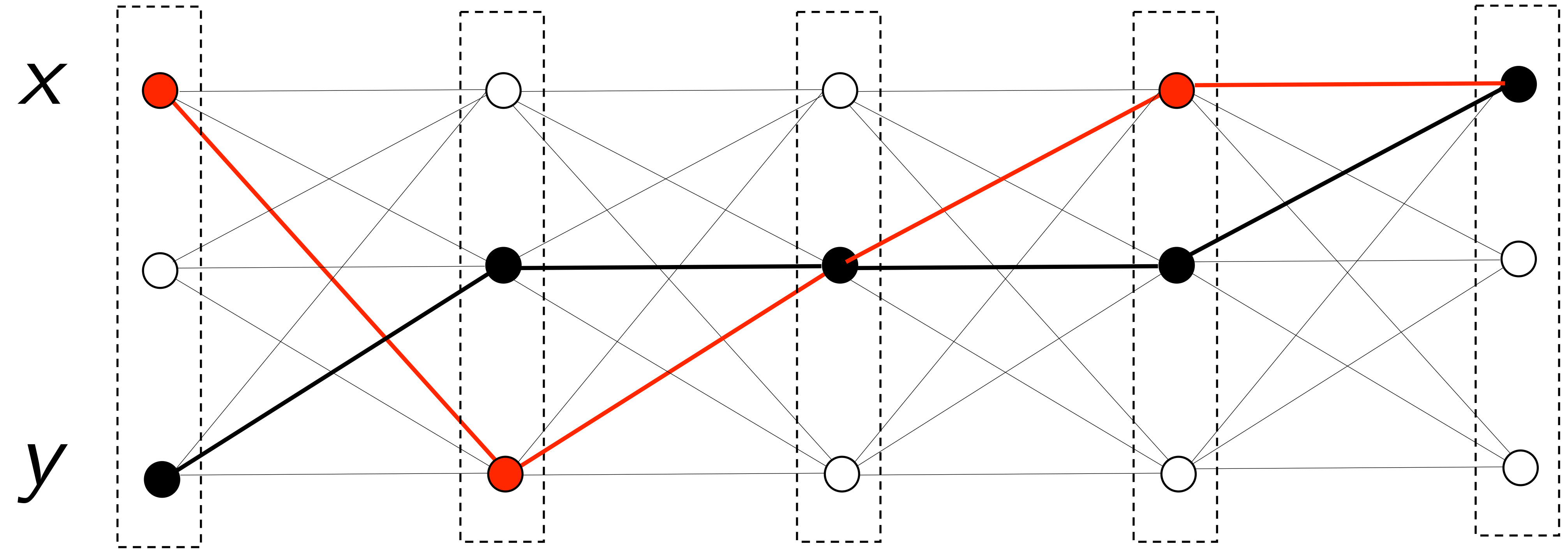
...

[Lempitsky et al. A Pylon Model for Semantic Segmentation, 2011]

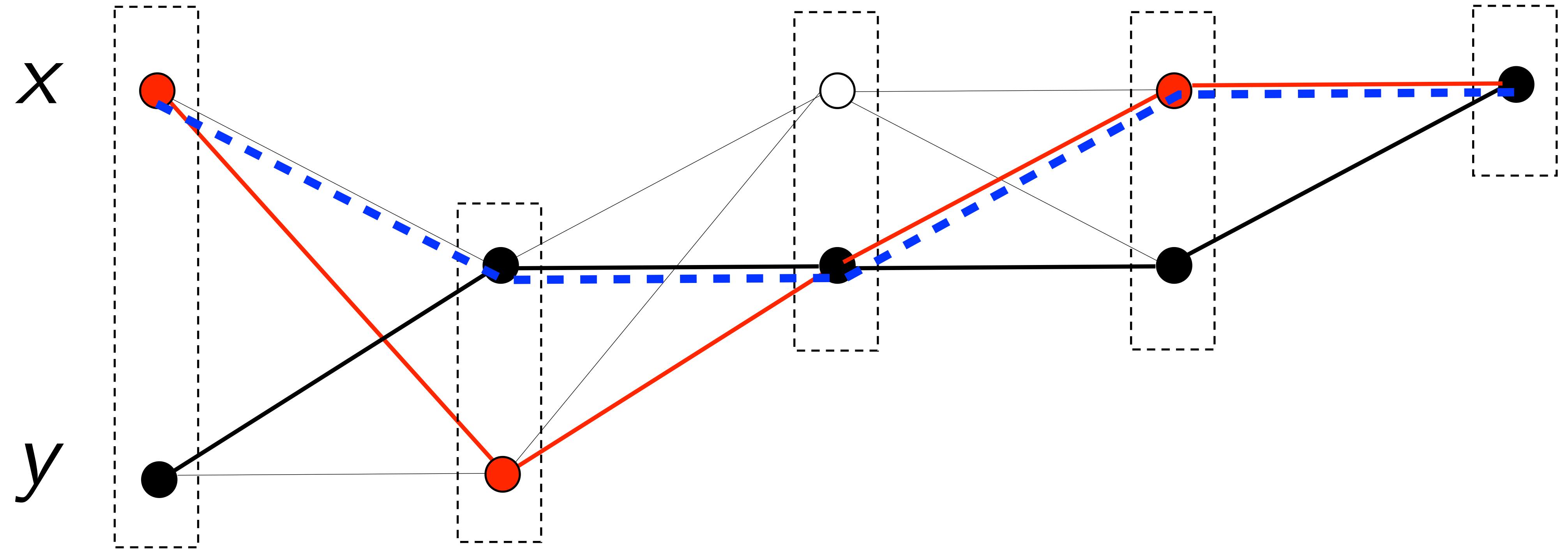
Optimized Crossover

Current best solution

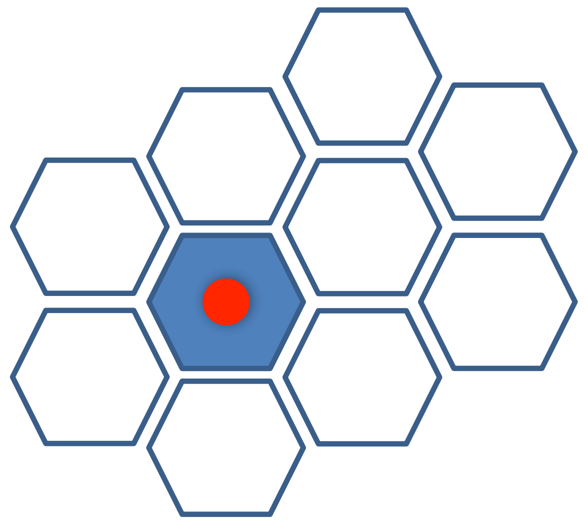
Proposal solution



Crossover (fusion problem)



Local Search in some combinatorial locality



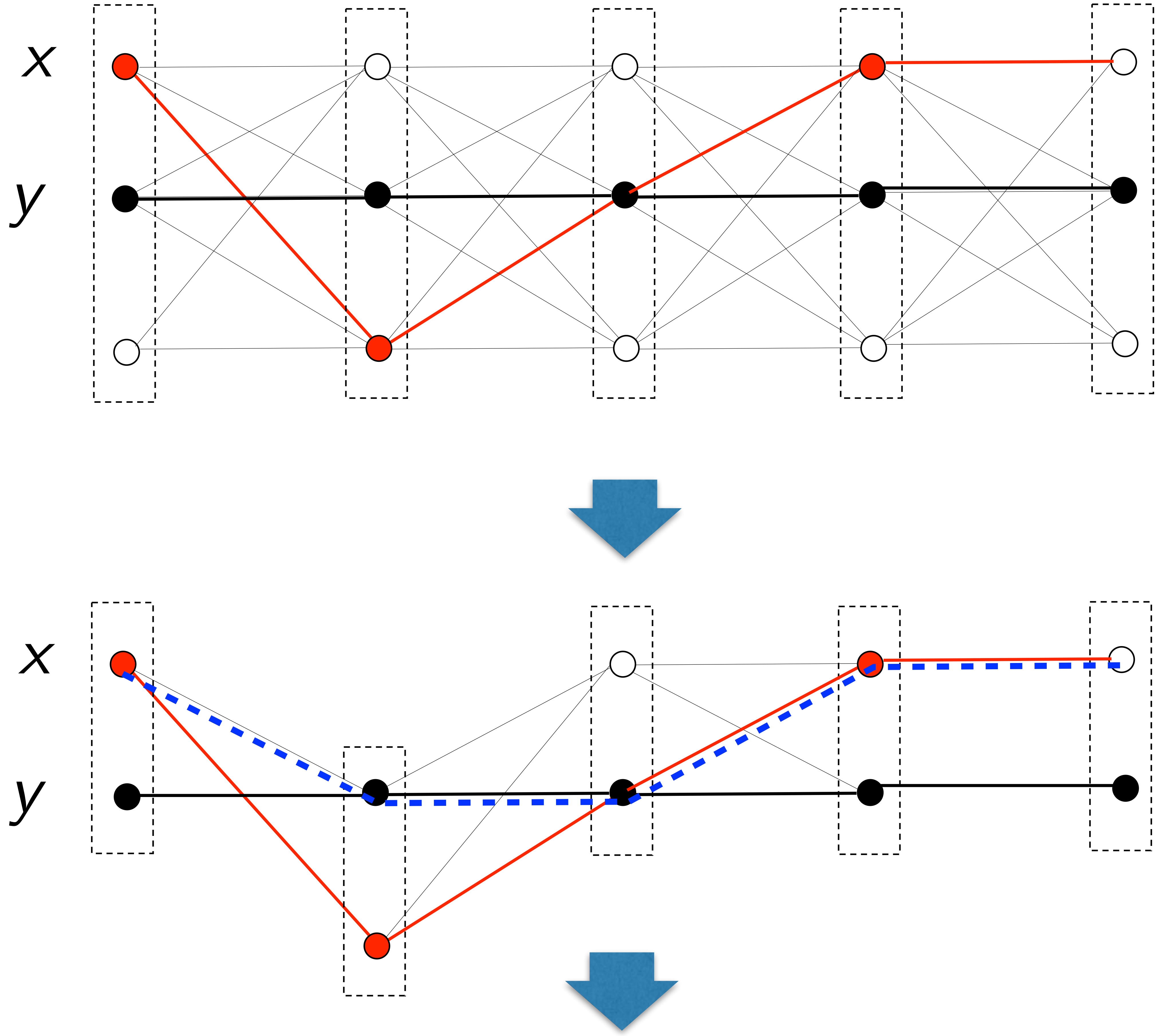
Expansion Move

[Boykov, Veksler, and Zabih: "Fast Approximate Energy Minimization via Graph Cuts", 1999]

Current best solution

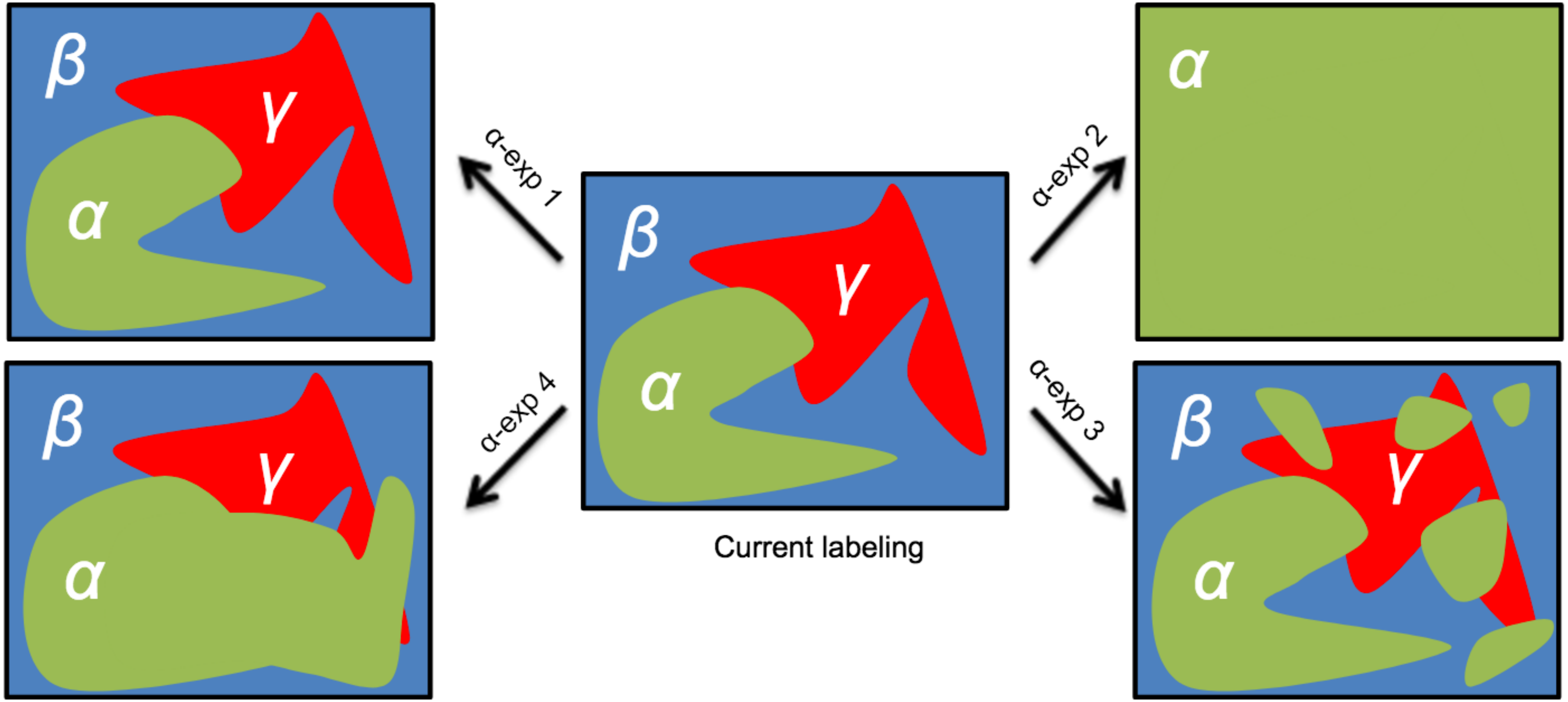
Proposal solution

Crossover (fusion problem)



Minimum Cut

Space of Possible Expansion of One Label

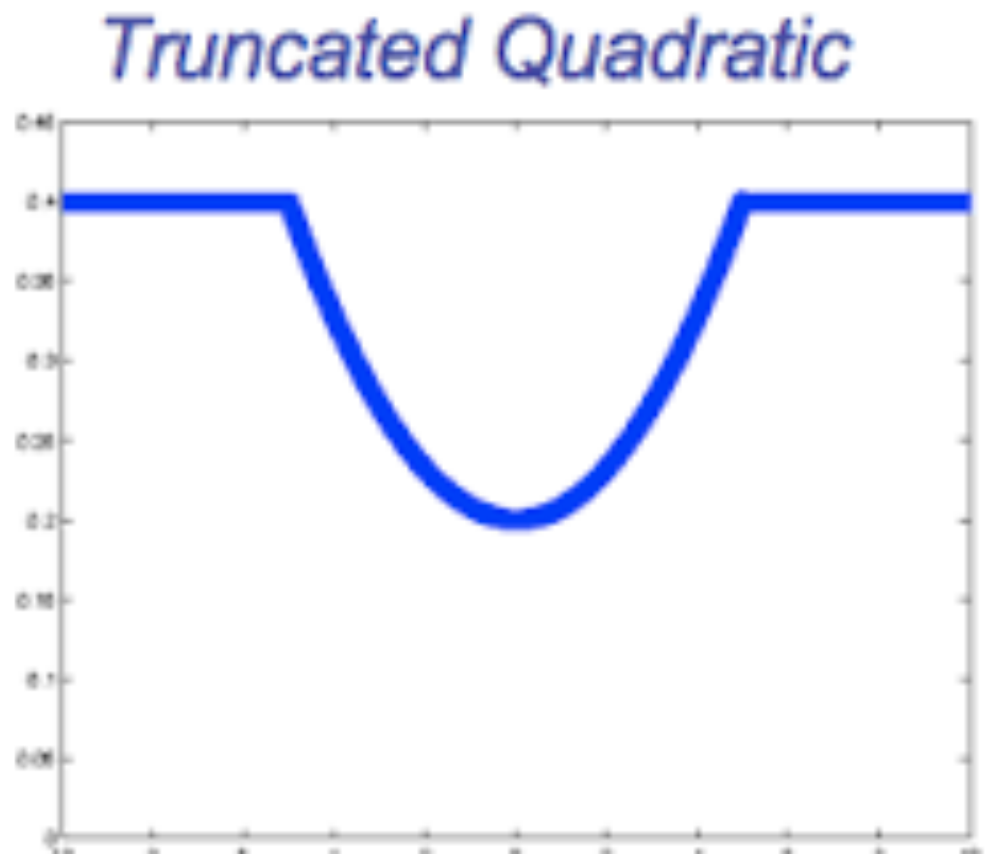


Expansion Move

- Start with initial solution x
- For each label a
  - Consider the Expansion-Move to a:
  - x_i stays or switches to a -> reduce to graph cut and solve
- Iterate until x stops changing

Semi-metric $f_{ij}(\alpha, \beta)$:

- $f_{ij}(\alpha, \beta) = 0$ iff $\alpha = \beta$
- $f_{ij}(\alpha, \beta) = f_{ij}(\beta, \alpha) \geq 0$
- $f_{ij}(\alpha, \beta) \leq f_{ij}(\alpha, \gamma) + f_{ij}(\gamma, \beta)$



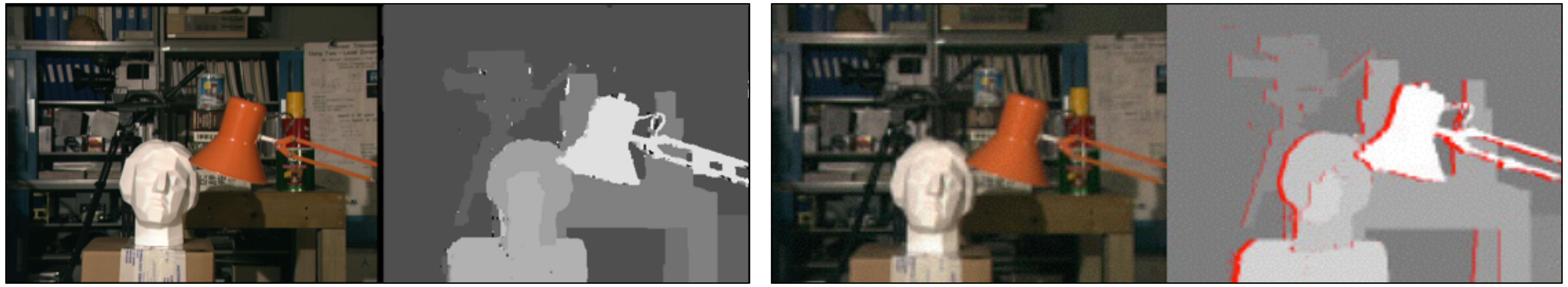
"robust" potentials:
outliers not over penalized

Theorem (Boykov, Veksler, Zabich, 1999)

For semi-metric problems, the expansion-move algorithm finds a solution with an approximation ratio:

$$2c = 2c \max_{ij} \frac{\max_{\alpha \neq \beta} f_{ij}(\alpha, \beta)}{\min_{\alpha \neq \beta} f_{ij}(\alpha, \beta)}$$

Applications of graph cuts



Stereo
Boykov et al. 1998
Kolmogorov and Zabih 2001

A general and fast technique

In 2011 received
Helmholtz Prize (Test of Time) Award

MRF Marginals — Mean Field Approximation

Computing Marginals

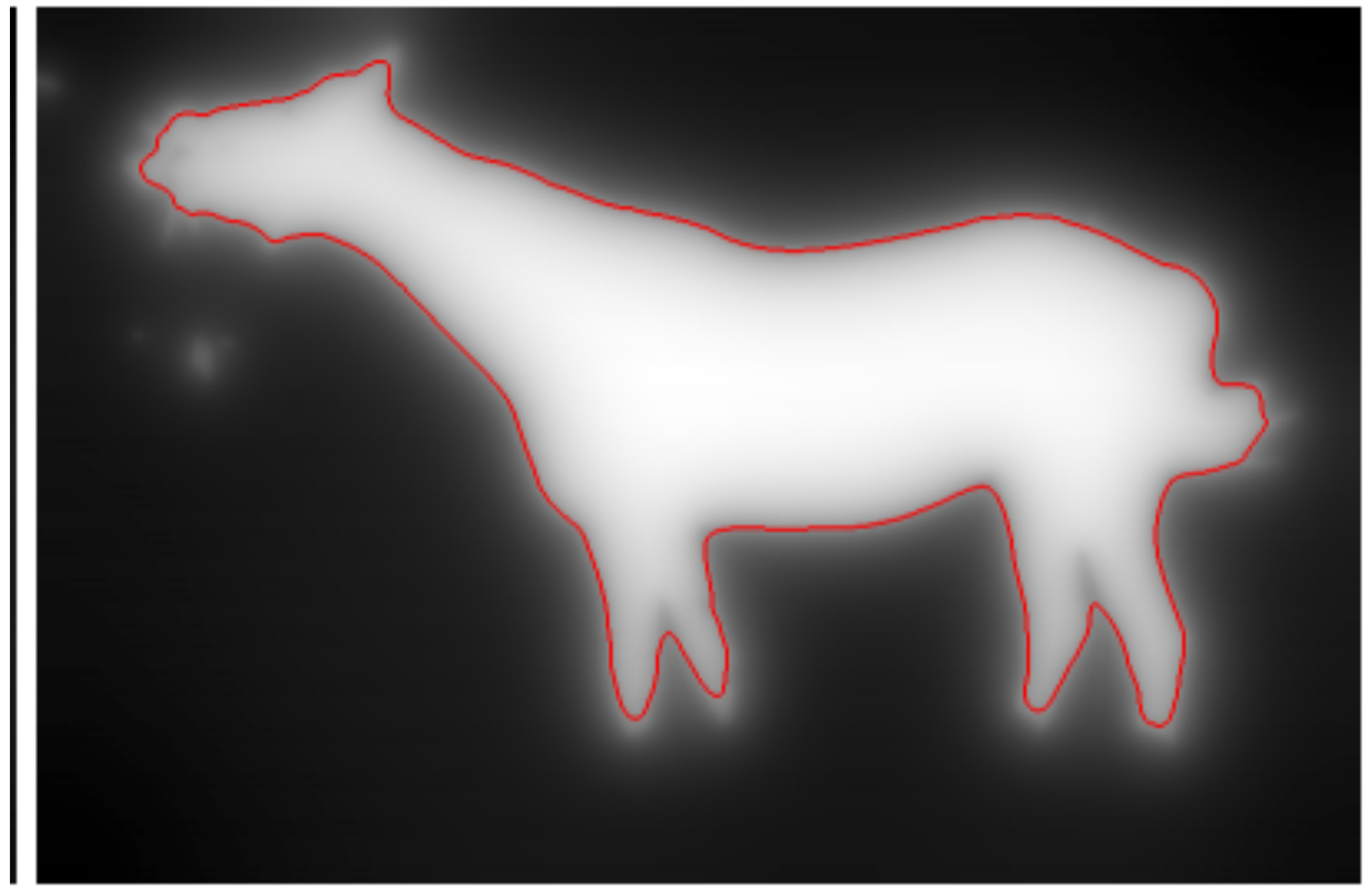
$$p(x \mid y) \propto \exp\Big(\sum_i -\phi_i(x_i, y_i) - \sum_{(i,j)} \phi_{ij}(x_i, x_j)\Big) \qquad \phi_{ji}(x_j, x_i) \equiv \phi_{ij}(x_i, x_j)$$

Posterior of the states given image

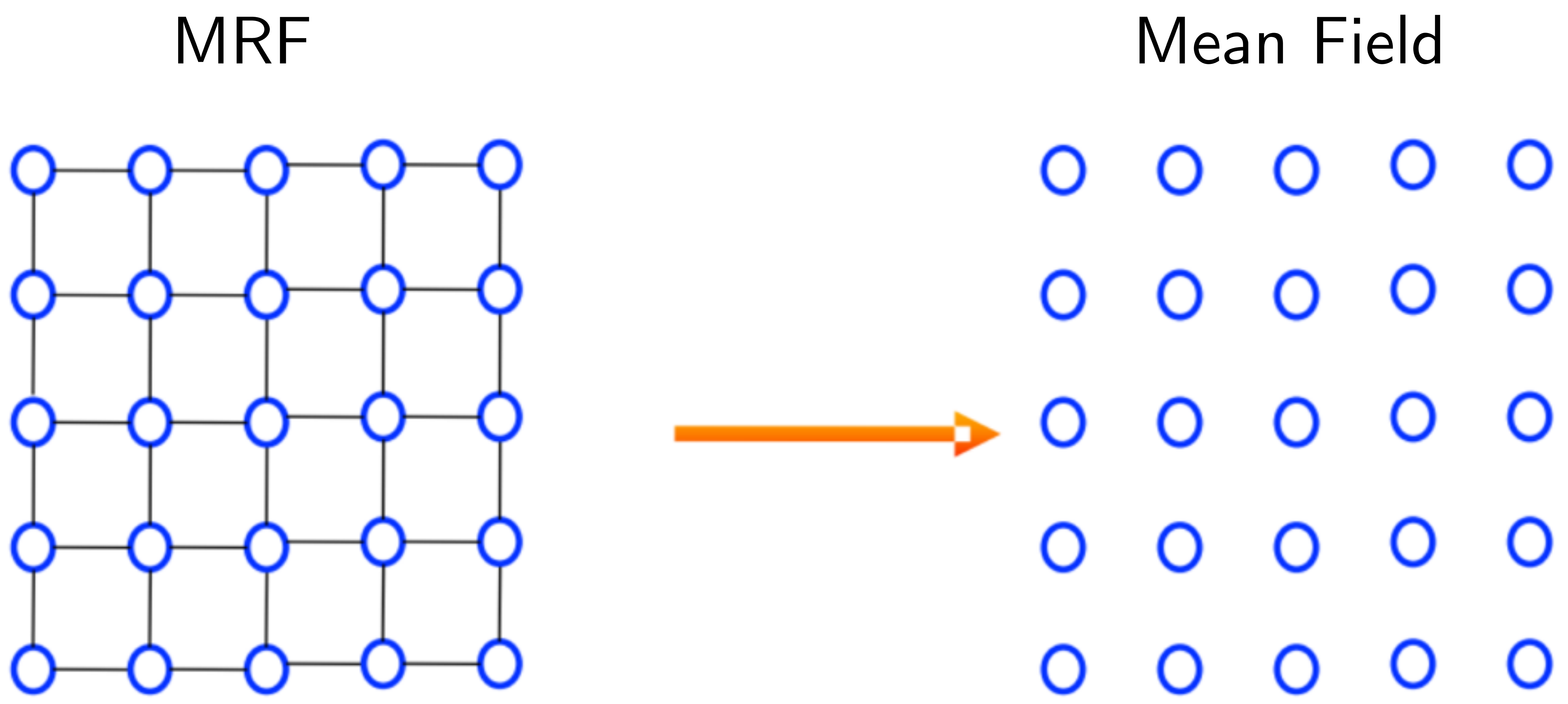
Want to estimate marginals $p(x_i \mid y)$

$$p(x_i \mid y) = \mathbb{E}_{x_{\mathcal{V}\setminus\{i\}}}\left[p(x \mid y)\right] \propto \sum_{x_{\mathcal{V}\setminus\{i\}}} \exp\Big(\sum_i -\phi_i(x_i, y_i) - \sum_{(i,j)} \phi_{ij}(x_i, x_j)\Big)$$

Example of Marginal Probabilities



Factorized Approximation of the Posterior



$$p(x_i \mid y) =\propto \sum_{x_{\mathcal{V}\setminus\{i\}}} \exp\left(\sum_i -\phi_i(x_i, y_i) - \sum_{(i,j)} \phi_{ij}(x_i, x_j)\right)$$

Posterior of the states given image

Want to estimate marginals $p(X_i \mid I)$

$$q(x) = \prod_i q_i(x_i)$$

Approximation of the posterior

(assume posterior distribution is concentrated around one configuration)

KL Divergence

Let $p(X)$ and $q(X)$ be two probability distributions.

Definition

Kullback–Leibler divergence (1951) of p and q is

$$KL(p(X)\|q(X)) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}$$

In the definition above $0\log\frac{0}{0} = 0\log\frac{0}{q} = 0$ and $p\log p0 = \infty$.

For continuous variables:

$$KL(p(X)\|q(X)) = \int p(x)\log\frac{p(x)}{q(x)}dx$$

The expected number of extra bits required to code samples from p using a code optimized for q

The amount of information lost when q is used to approximate p

Non-negative, KL(p||q) = 0 iff p = q

Non-negativity of KL

Assume $p(x) > 0$, $q(x) > 0$, $\sum_x p(x) = 1$, $\sum_x q(x) = 1$

Statement: $\sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0$

Proof

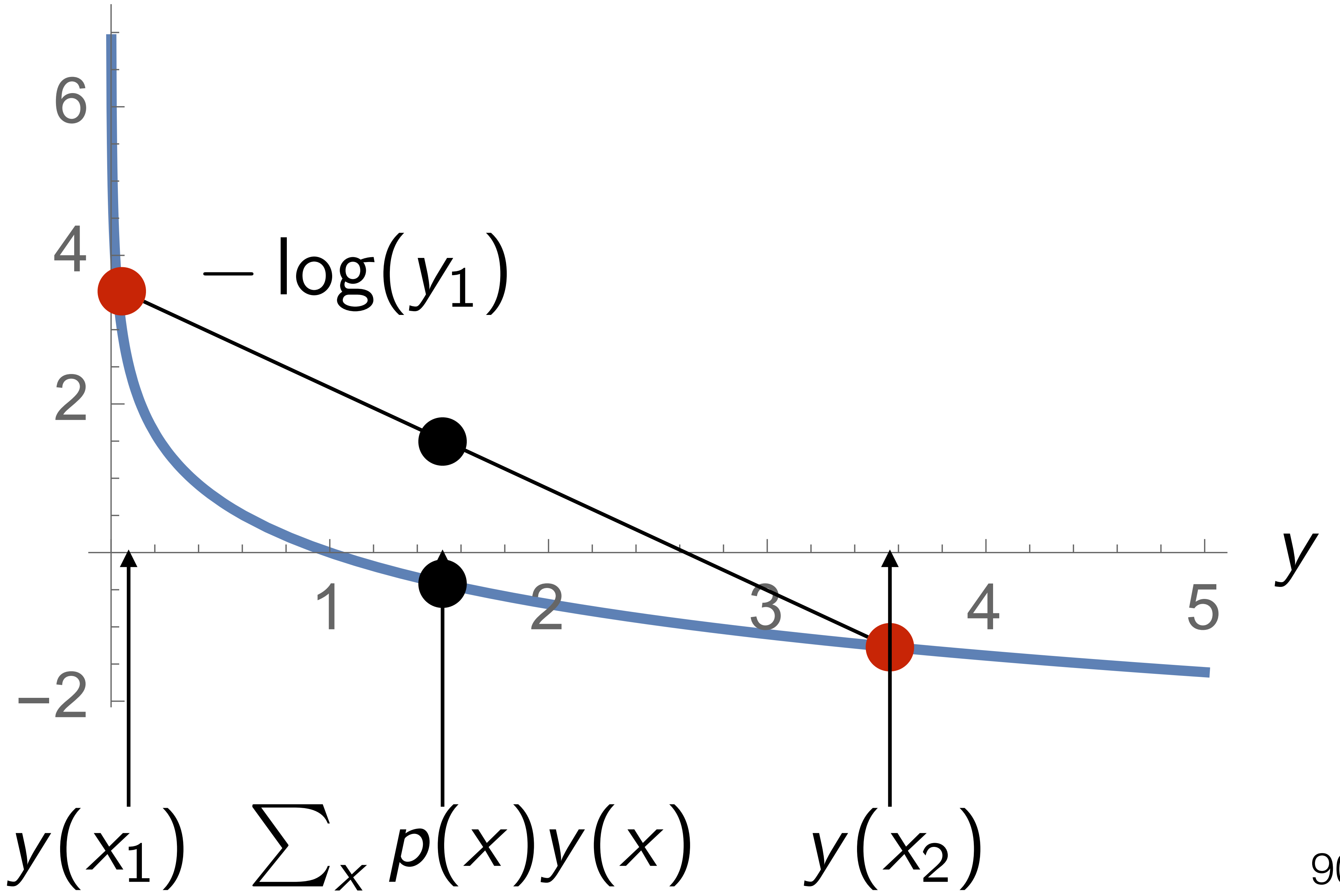
Denote $y(x) = \frac{q(x)}{p(x)}$, the inequality reads:

$\sum_x p(x)(-\log y(x)) \geq 0$

Observe that log is a convex function, apply Jensen's inequality:

$\sum_x p(x)(-\log y(x)) \geq -\log \sum_x p(x)y(x) = -\log 1 = 0$

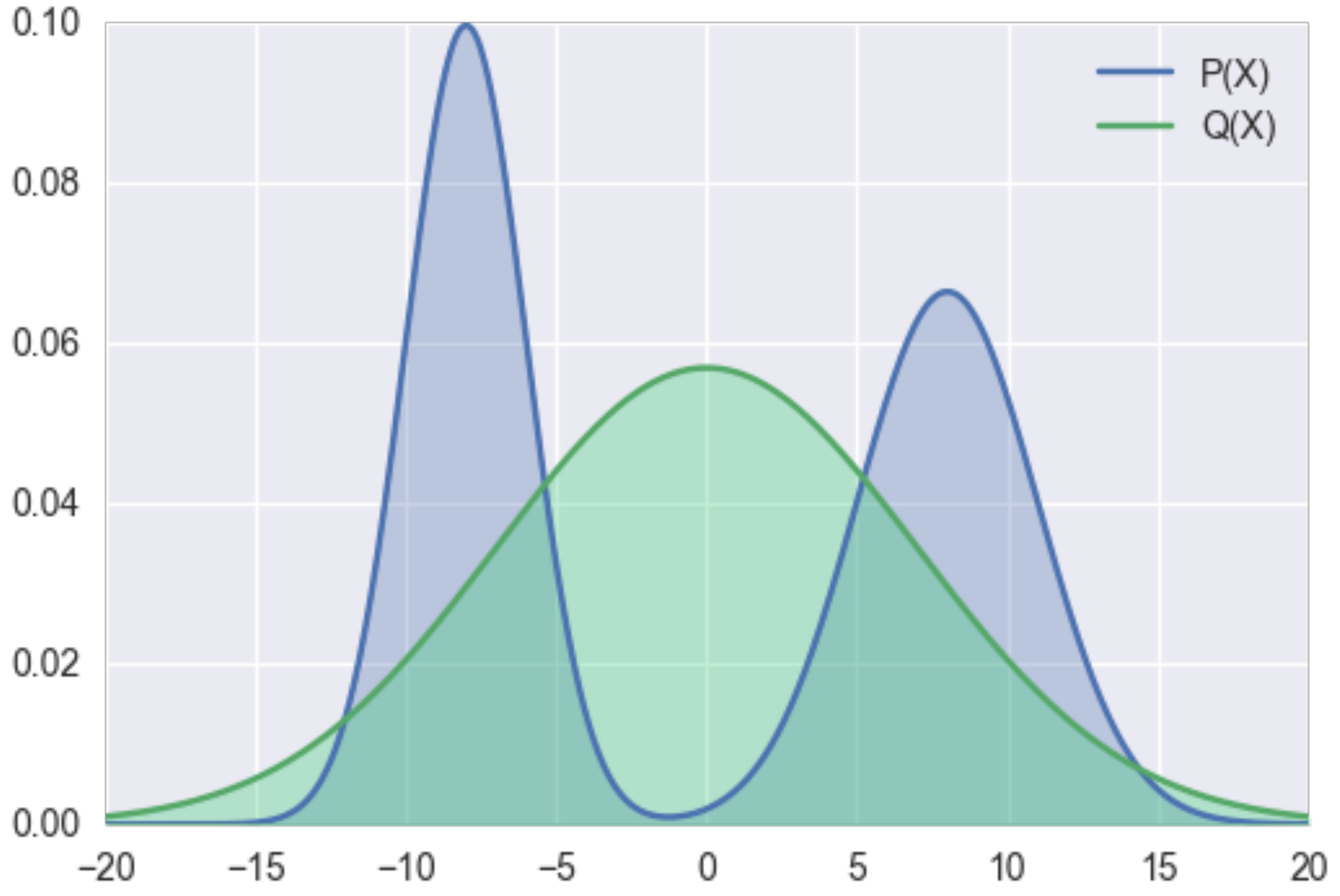
From strictly convexity: equality iff all y(x) are equal



Asymmetry

Minimizing forward KL divergence:

$$\min_q KL(p\|q) \quad \left(\int p(x) \log \frac{p(x)}{q(x)} dx \right)$$



Well on average in the expectation over p

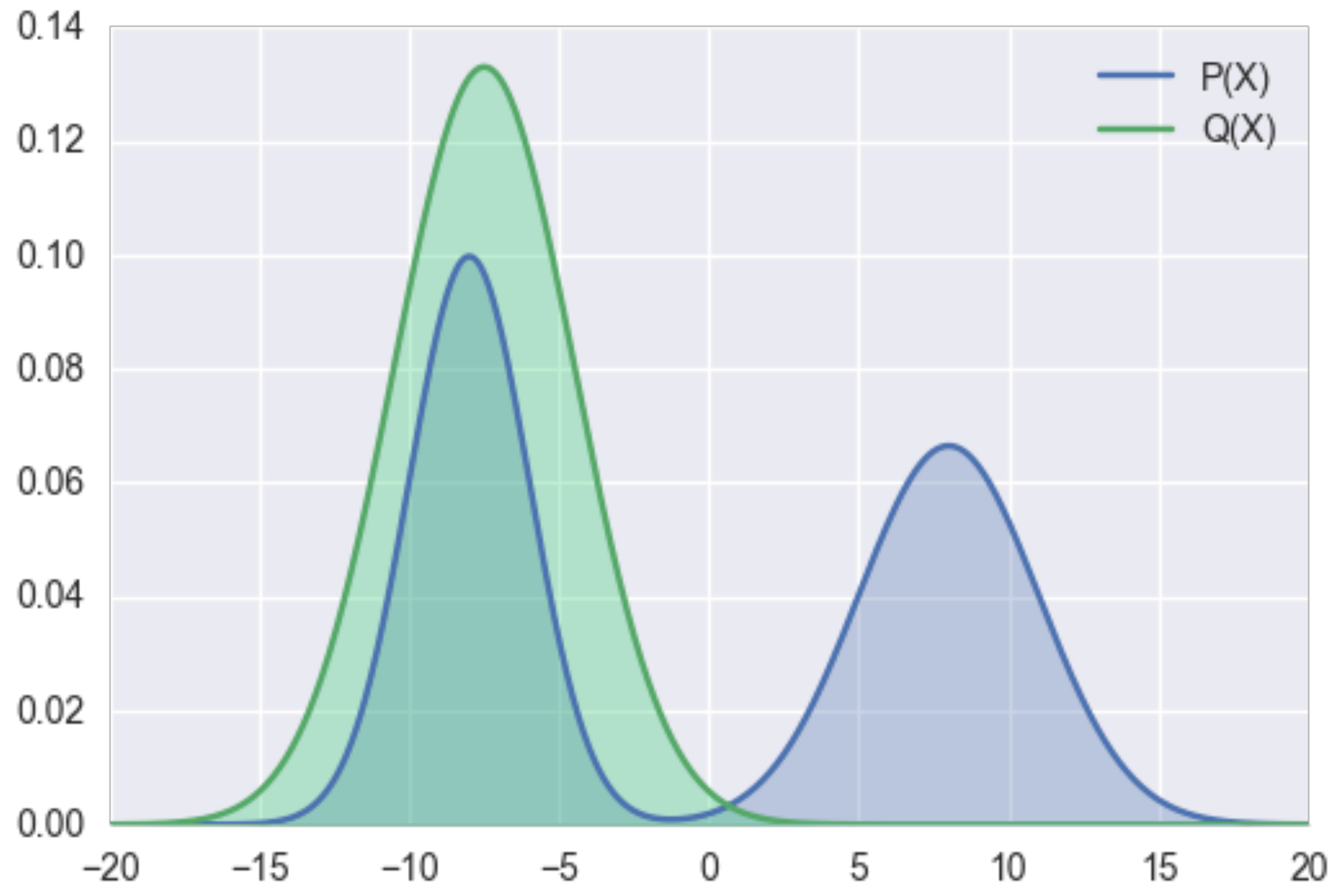
Example:

p - bimodal

q - Gaussian

Minimizing reverse KL divergence:

$$\min_q KL(q\|p) \quad \left(\int q(x) \log \frac{q(x)}{p(x)} dx \right)$$



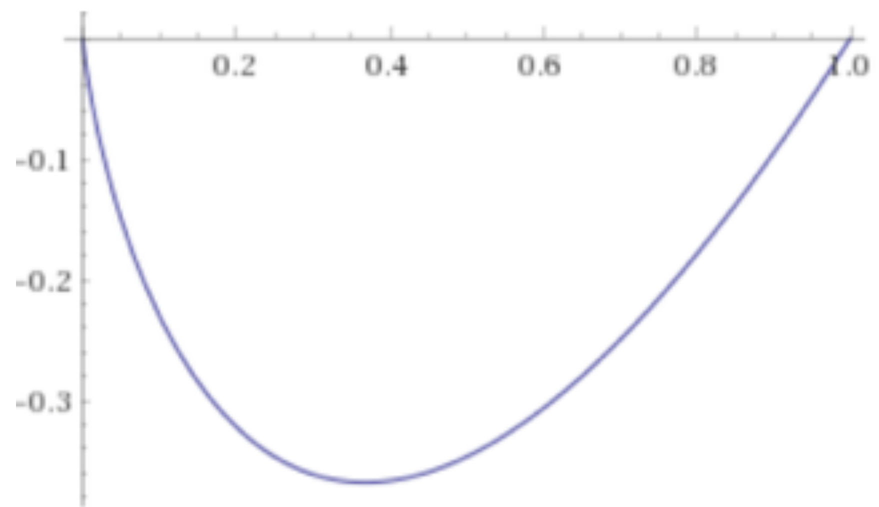
Well on average in the expectation over q — concentrating around a mode of p

- This gives rise to two families of variational methods

Reverse KL — Mean Field

$$KL(q\|p) = \sum_x q(x) \log \frac{q(x)}{p(x)} = -\sum_x q(x) \log p(x) + \sum_x q(x) \log q(x)$$

Cross-entropy / Evidence -Entropy



Entropy of independent variables is additive:

$$\sum_x q(x) \log q(x) = \sum_x \prod_{i'} q_{i'}(x_{i'}) \sum_i \log q_i(x_i) = \sum_x \sum_i \prod_{i'} q_{i'}(x_{i'}) \log q_i(x_i)$$

$$= \sum_i \sum_x \prod_{i'} q_{i'}(x_{i'}) \log q_i(x_i) = \sum_i \sum_{x_i} q_i(x_i) \log q_i(x_i) = \sum_i -H(q_i).$$

Cross-entropy decouples over pairwise terms:

$$\sum_x q(x) \log p(x) = -\sum_x \prod_{i'} q_{i'}(x_{i'}) (\sum_i \phi_i(x_i) + \sum_{ij} \phi_{ij}(x_i, x_j))$$

$$= -\sum_i \sum_{x_i} \phi_i(x_i) q_i(x_i) - \sum_{ij} \sum_{x_i, x_j} \phi_{ij}(x_i, x_j) q_i(x_i) q_j(x_j)$$

Mean Field

$$\min_q \sum_i \sum_{x_i} q_i(x_i)\Big(\log q_i(x_i) + \phi_i(x_i) + \sum_{j\in\mathcal{N}(i)} \sum_{x_j} q_j(x_j)\phi_{ij}(x_i,x_j)\Big)$$

$$\text{s.t.} \quad q_i \geq 0; \quad \sum_{x_i} q_i(x_i) = 1 \quad \forall i \quad | \quad \text{Lagrange multiplier } \lambda_i$$

Non-convex because of $q_i q_j$

$$0 = \frac{\partial}{\partial q_i(x_i)} = \log q_i(x_i) + \phi_i(x_i) + 1 + \sum_{j\in\mathcal{N}(i)} \sum_{x_j} q_j(x_j)\phi_{ij}(x_i,x_j) - \lambda_i$$

$$\log q_i(x_i) = -\phi_i(x_i) - \sum_{j\in\mathcal{N}(i)} \sum_{x_j} q_j(x_j)\phi_{ij}(x_i,x_j) - \lambda_i'$$

$$q_i(x_i) \propto \exp(-\phi_i(x_i)) \prod_{j\in\mathcal{N}(i)} \exp\Big(-\sum_{x_j} q_j(x_j)\phi_{ij}(x_i,x_j)\Big)$$

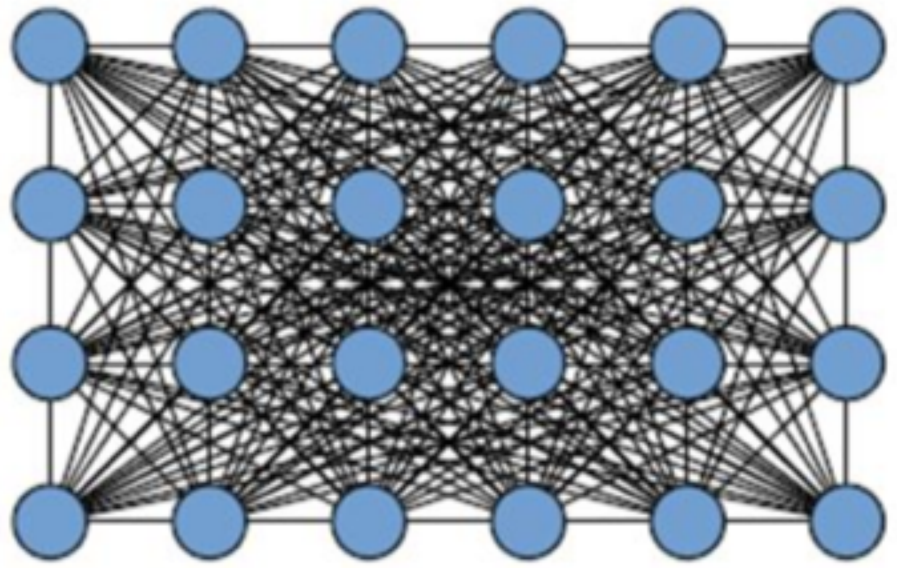
Algorithms:

sequential coordinate-wise minimization (convergent)

parallel coordinate-wise (may oscilate)

Fully Connected (Dense) CRFs

Assume potentials have the following structure: $\phi_{ij}(x_i, x_j) = \rho(x_i, x_j)k(i-j)$



$$\log q_i(x_i) = \phi_i(x_i) + \sum_{j \neq i} \sum_{x_j} q_j(x_j)\rho(x_i, x_j)k(i-j) - \lambda'$$

Parallel update can be implemented efficiently:

- For all labels l:
  - $s(j) := \sum_{l'} q_j(l')\rho(l, l')$
  - $\log q_i'(l) := \phi_i(l) + \sum_{j \neq i} s(j)k(i-j) = \phi_i(l) + s * k - s(i)k(0)$
- Renormalize all q_i'

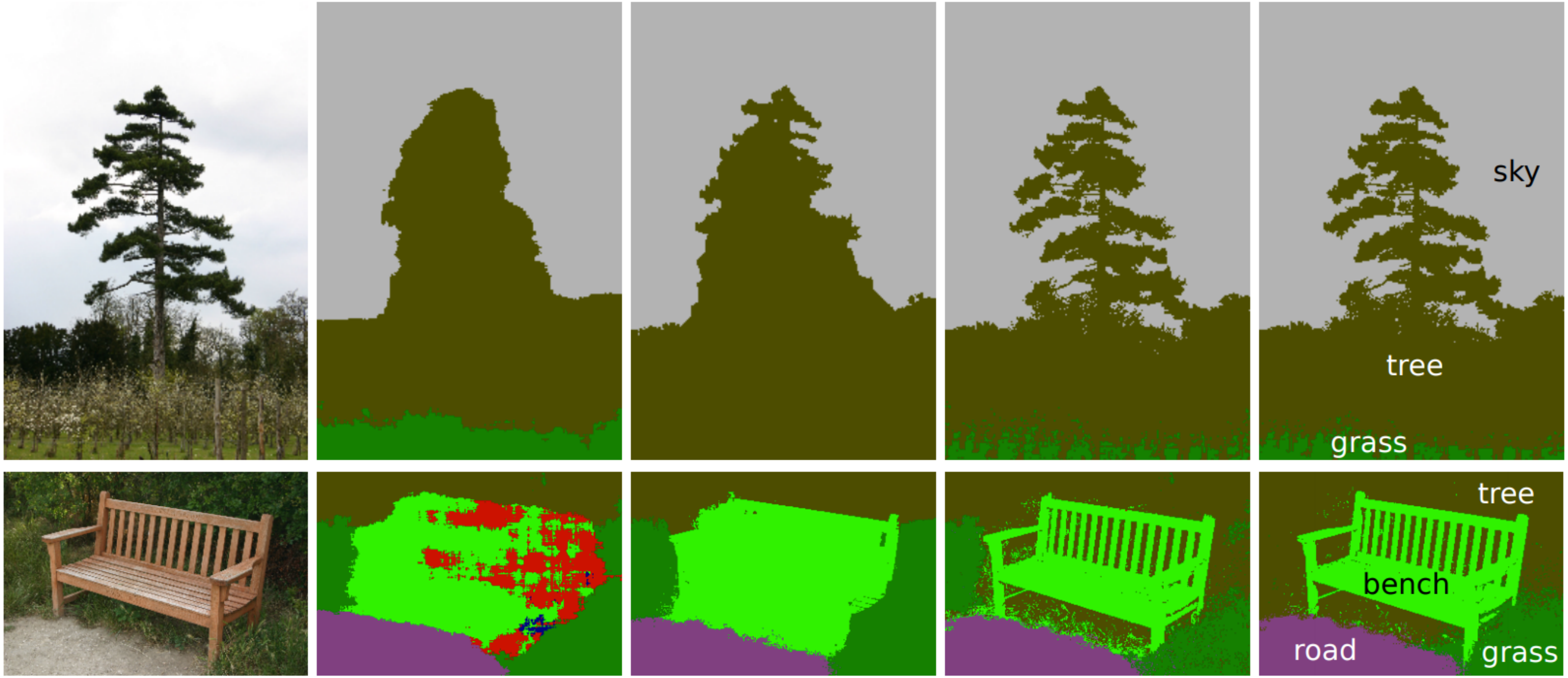
[Kraehenbuehl and Koltun: Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, 2012]

Potentials of the form: $\phi_{ij}(x_i, x_j) = \rho(x_i, x_j) \sum_m w_m k^m(f_i - f_j)$,

f -some features $\rightarrow$ bilateral filtering

Convergence with some assumptions, better algorithms than parallel coordinate-descent, other relaxations

[Kraehenbuehl and Koltun: Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, 2012]



(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF, MCMC inference, 36 hrs (e) Fully connected CRF, our approach, 0.2 seconds

Forward KL

$$KL(p\|q) = \sum_x p(x)\log\frac{p(x)}{q(x)} = \sum_x p(x)\log p(x) - \sum_x p(x)\log q(x)$$

-Entropy of p   Cross-entropy $-\mathbb{E}_{p(X)}\log q(X)$

When minimizing in q, H(p) does not matter

Cross-entropy simplifies using factorization of q:

$$\sum_x p(x)\log q(x) = \sum_x p(x)\sum_i \log q_i(x_i) = \sum_i \sum_{x_1,\ldots,x_i,\ldots,x_n} p(x)\log q_i(x_i) = \sum_i\sum_{x_i} p(x_i)\log q_i(x_i)$$

Turns out that we need to know marginals $p(X_i)$. But then:

$$\min_q -\sum_i\sum_{x_i} p(x_i)q_i(x_i) \quad \text{s.t.} \sum_i q_i = 1 \quad \Rightarrow \quad q_i(x_i) = p(x_i)$$

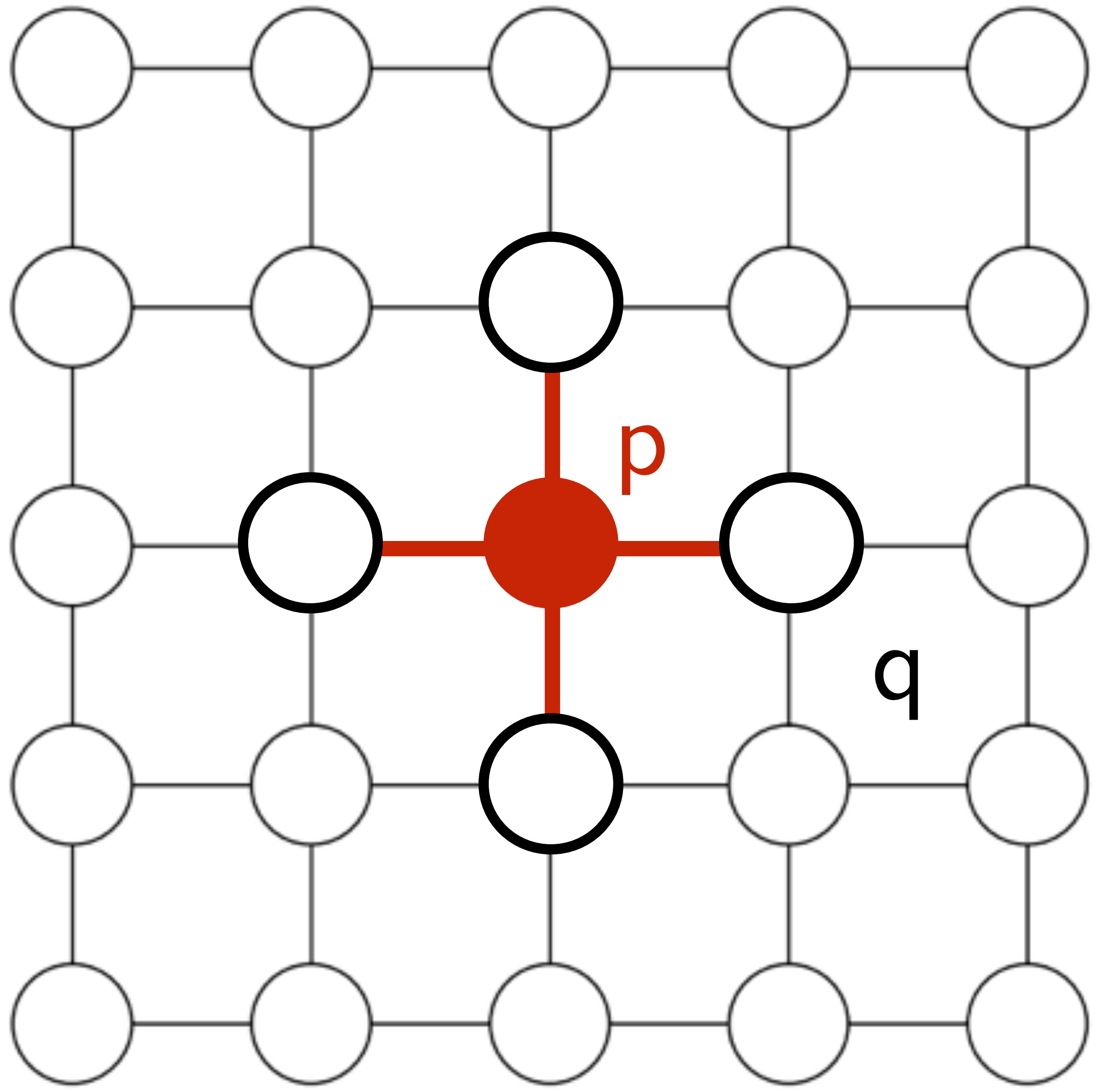
Forward divergence was the “right one” but we did not get a simplification

Mean Field as Approximation and Forward KL

$$q_i'(x_i) \propto \exp(-\phi_i(x_i)) \prod_{j \in \mathcal{N}(i)} \exp\Big(-\sum_{x_j} q_j(x_j)\phi_{ij}(x_i, x_j)\Big)$$

Terms from the original distribution

terms from current estimate



The iterative algorithm can be understood as follows. At each iteration

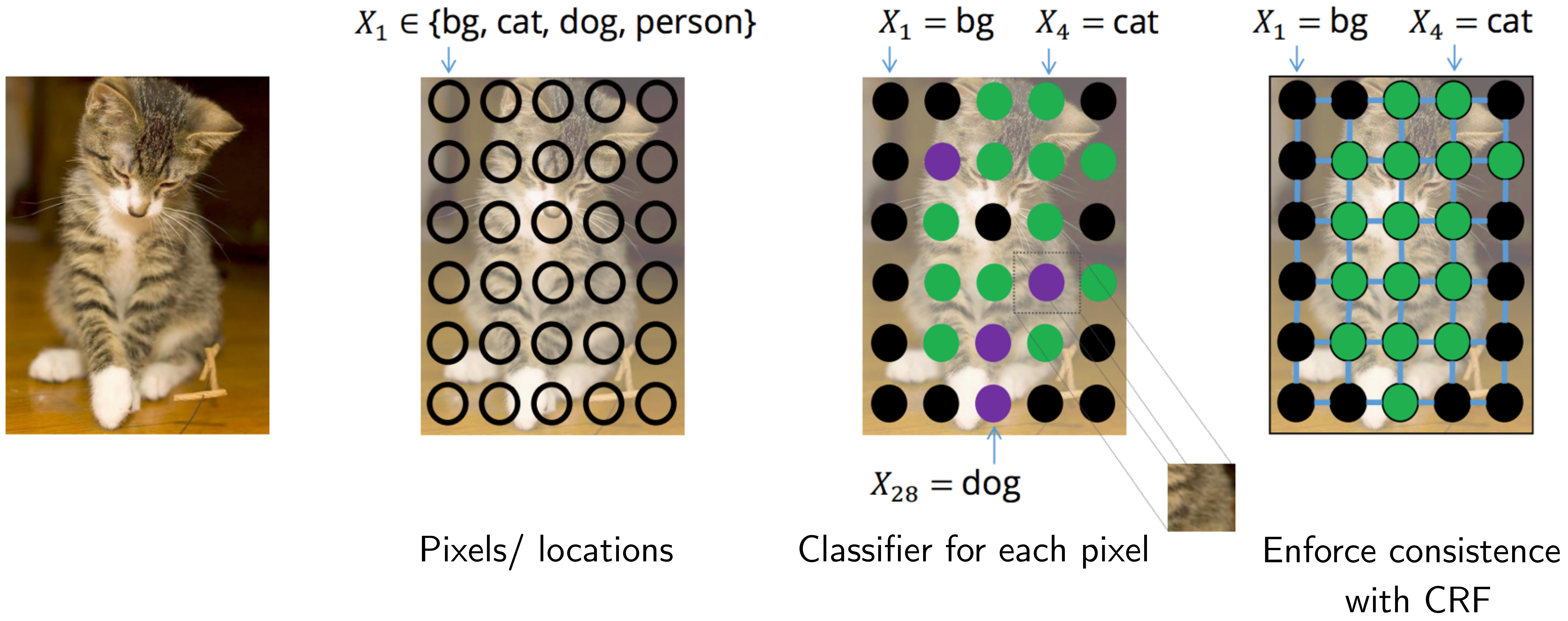
- Approximate $p(x) \approx \hat{p}(x) = p(x_i \,|\, x_{V \setminus \{i\}}) q(x_{V \setminus \{i\}})$
- Minimize $KL(\hat{p} \| q)$

Note, the second step efficiently means $q_i := \hat{p}(x_i) = \sum_{x_{\mathcal{N}(i)}} p(x_i \,|\, x_{\mathcal{N}(i)}) q(x_{\mathcal{N}(i)})$

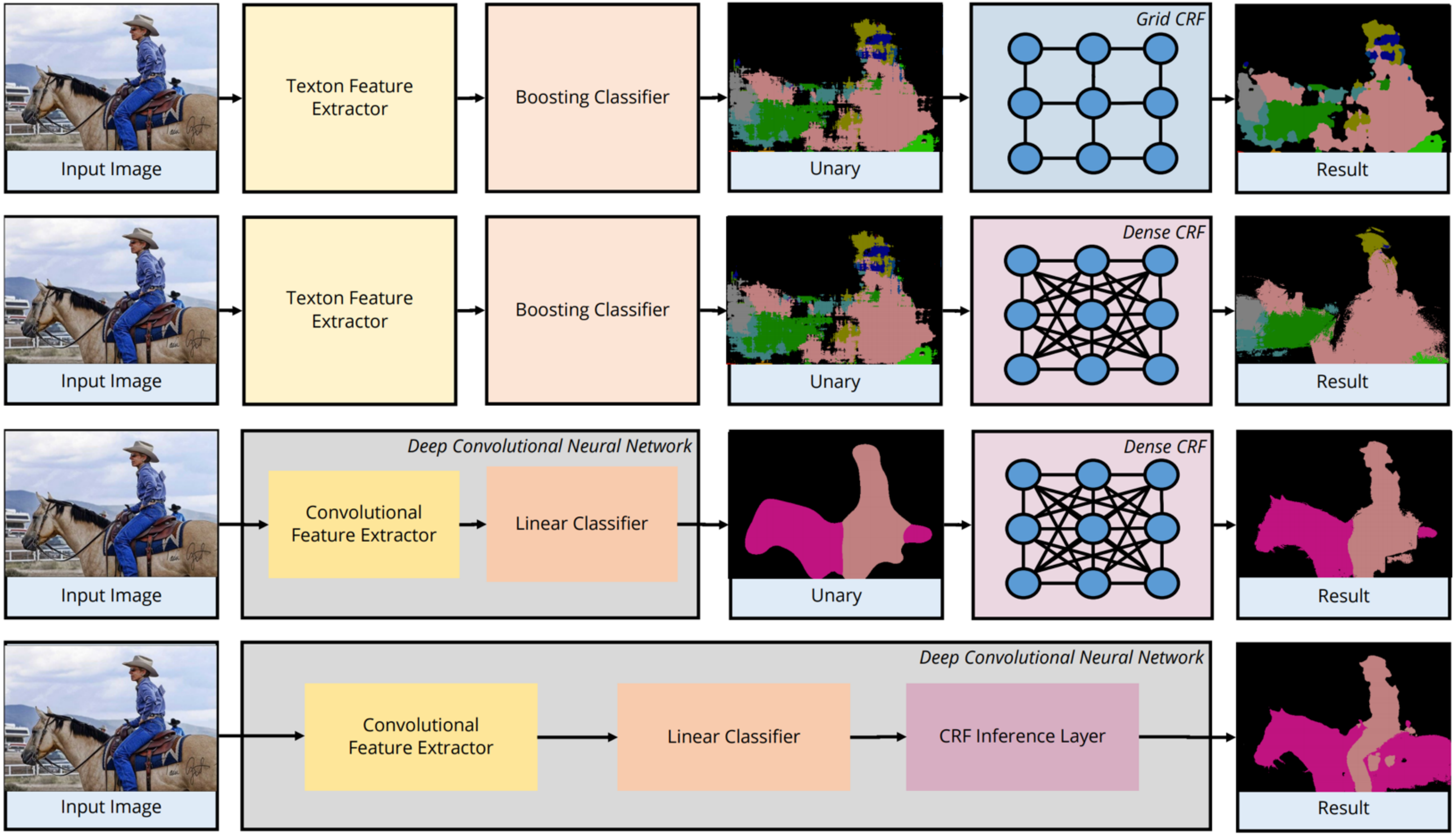
Graphical Models as Neural Networks

Materials: Arnab et al. “Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation”, 2018

Semantic Segmentation



Gradual "Neuralization" of CRF approaches



FCN + Mean Field CRF

Mean Field CRF inference as common CNN operations

$Q_u(l) \leftarrow \frac{1}{\sum_{l'} \exp(U_u(l'))} \exp(U_u(l))$ ▷ Initialization

while not converged **do**

$\tilde{Q}_u^{(m)}(l) \leftarrow \sum_{v \neq u} k^{(m)}(\mathbf{f}_u, \mathbf{f}_v) Q_v(l)$ for all m

▷ Message Passing

$\check{Q}_u(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_u^{(m)}(l)$

▷ Weighting Filter Outputs

$\hat{Q}_u(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \check{Q}_u(l')$

▷ Compatibility Transform

$\breve{Q}_u(l) \leftarrow U_u(l) - \hat{Q}_u(l)$

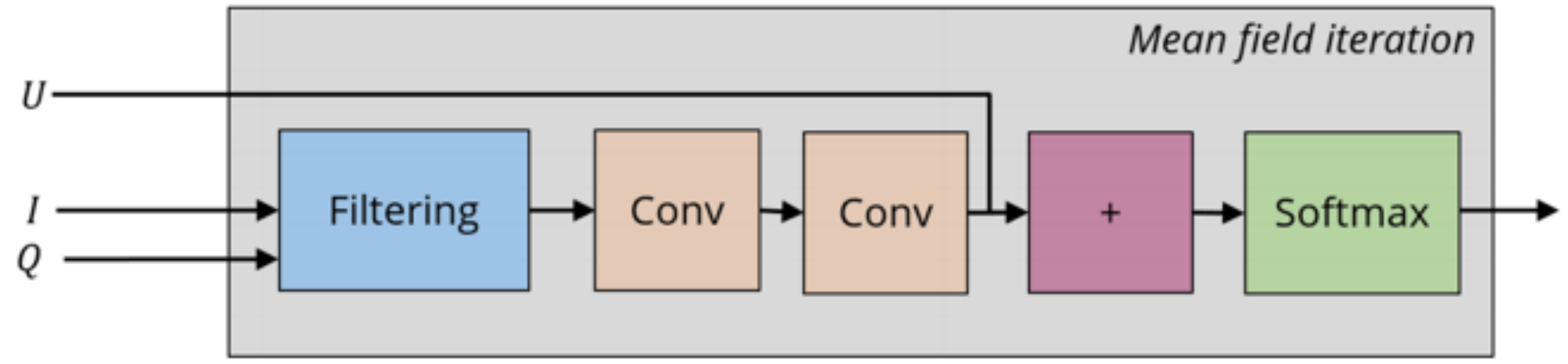
▷ Adding Unary Potentials

$Q_u(l) \leftarrow \frac{1}{\sum_{l'} \exp(\breve{Q}_u(l'))} \exp\left(\breve{Q}_u(l)\right)$

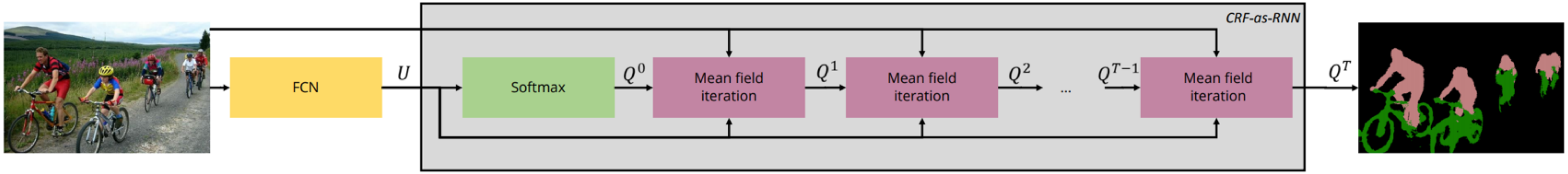
▷ Normalizing

end while

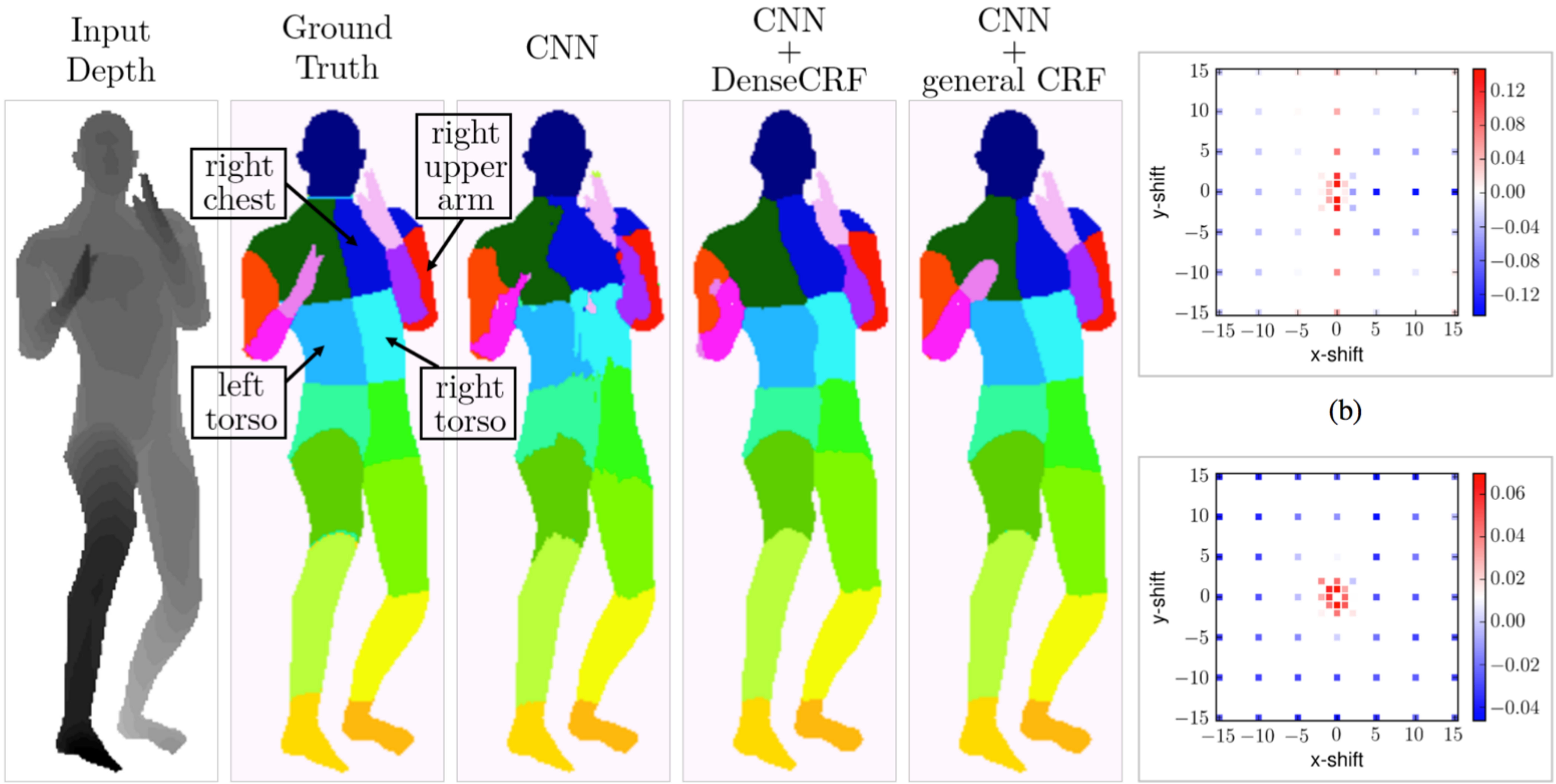
Mean Field Iteration



Conditional random fields as recurrent neural, networks (Zheng et al., 2015)



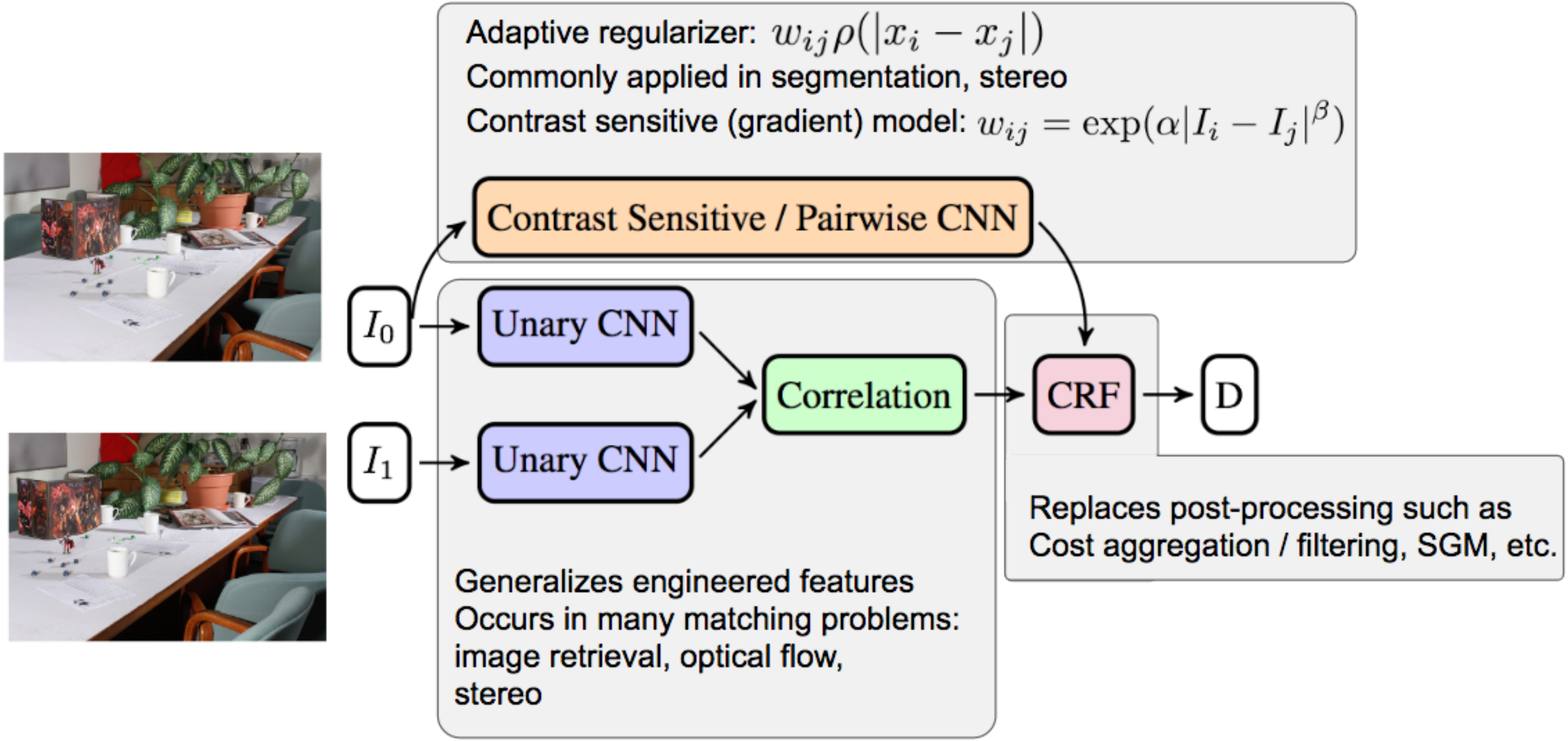
Another Example: CRF with Learned Potential Structure



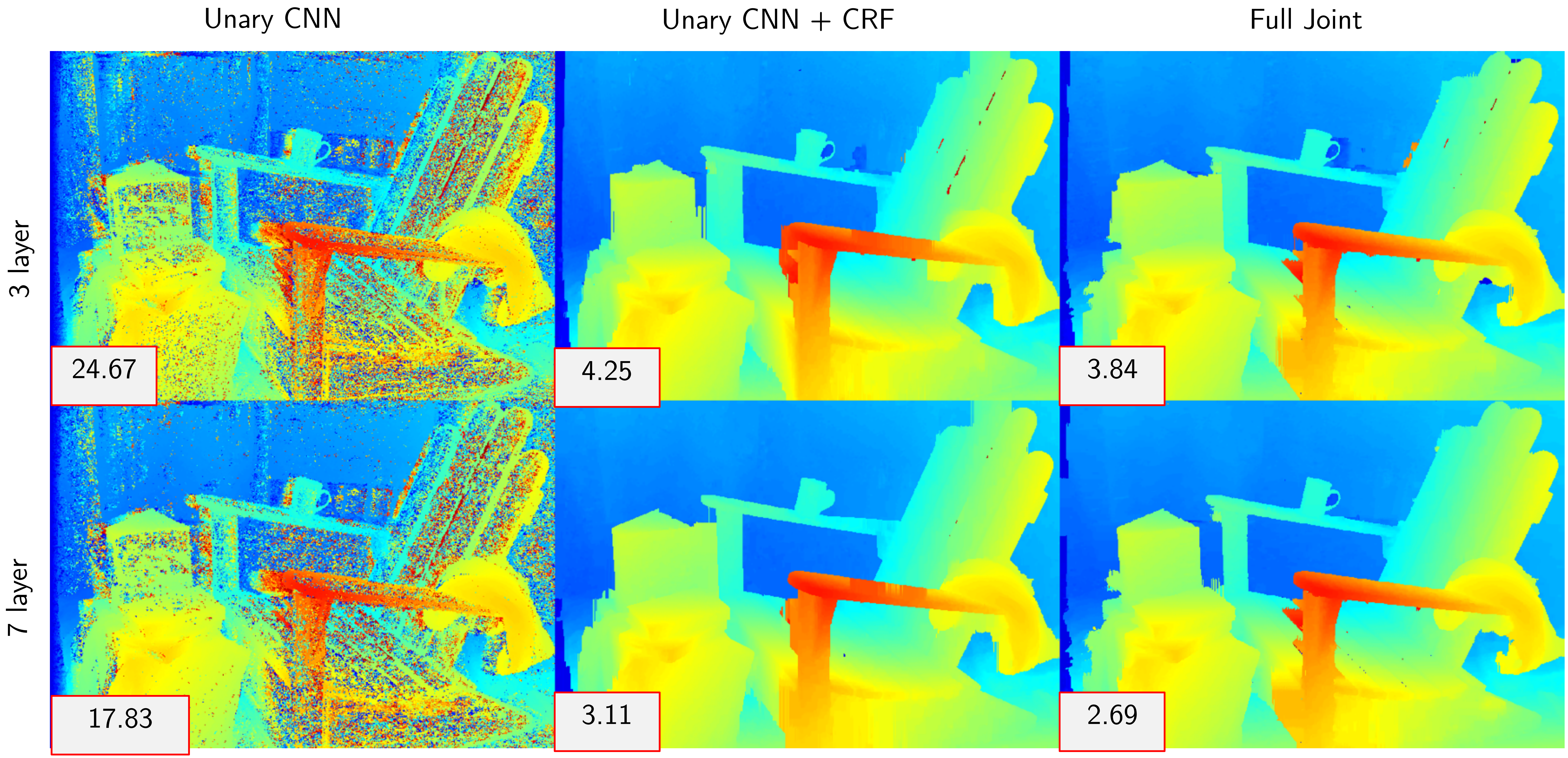
Improved results compared to DenseCRF, based on Gibbs sampling (training and test time)

CNN+CRF Stereo

Knöbelreiter et al. End-to-End Training of Hybrid CNN+CRF Models for Stereo, 2017



Effect of Joint Training



5 iterations of DMM

Discussion

- CRF could improve the results
- But also, we practically implemented it with CNN-like elements
- It means that in fact we have designed specialized CNN layers with a special structure
  - allowing for more spatial interactions
  - enforcing clustering of neighboring predictions
  - adjusting to image edges
- Does it matter that these layers were derived from MAP CRF?

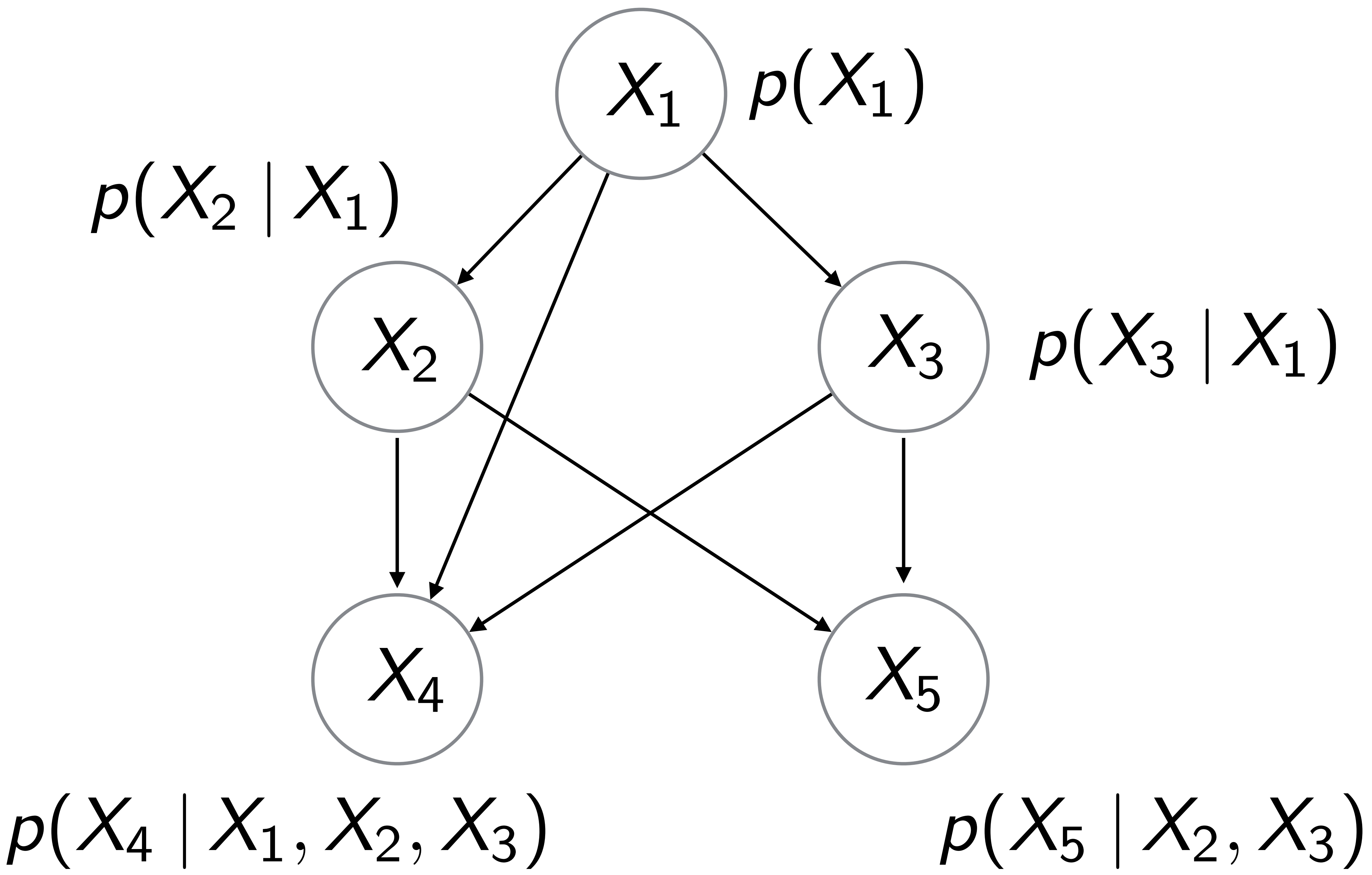
- Further Topics
  - Deep Boltzman machine, Deep Bayesian network

Bayesian Networks

Directed Graphical Model (Bayesian Network)

- Directed Acyclic Graph
  - Graph $G = (V, E)$
  - Set of nodes V; random variables X_i, $i \in V$
  - Set of directed edges $E \subset V \times V$
  - There are no directed loops in G
  - *Parents* of i is the set $\mathrm{Pa}(i) = \{j \in \mathcal{V} \mid (j, i) \in E\}$

Edges encode "direct dependencies"



Definition

Bayesian network w.r.t. graph G is a random field that factorizes as

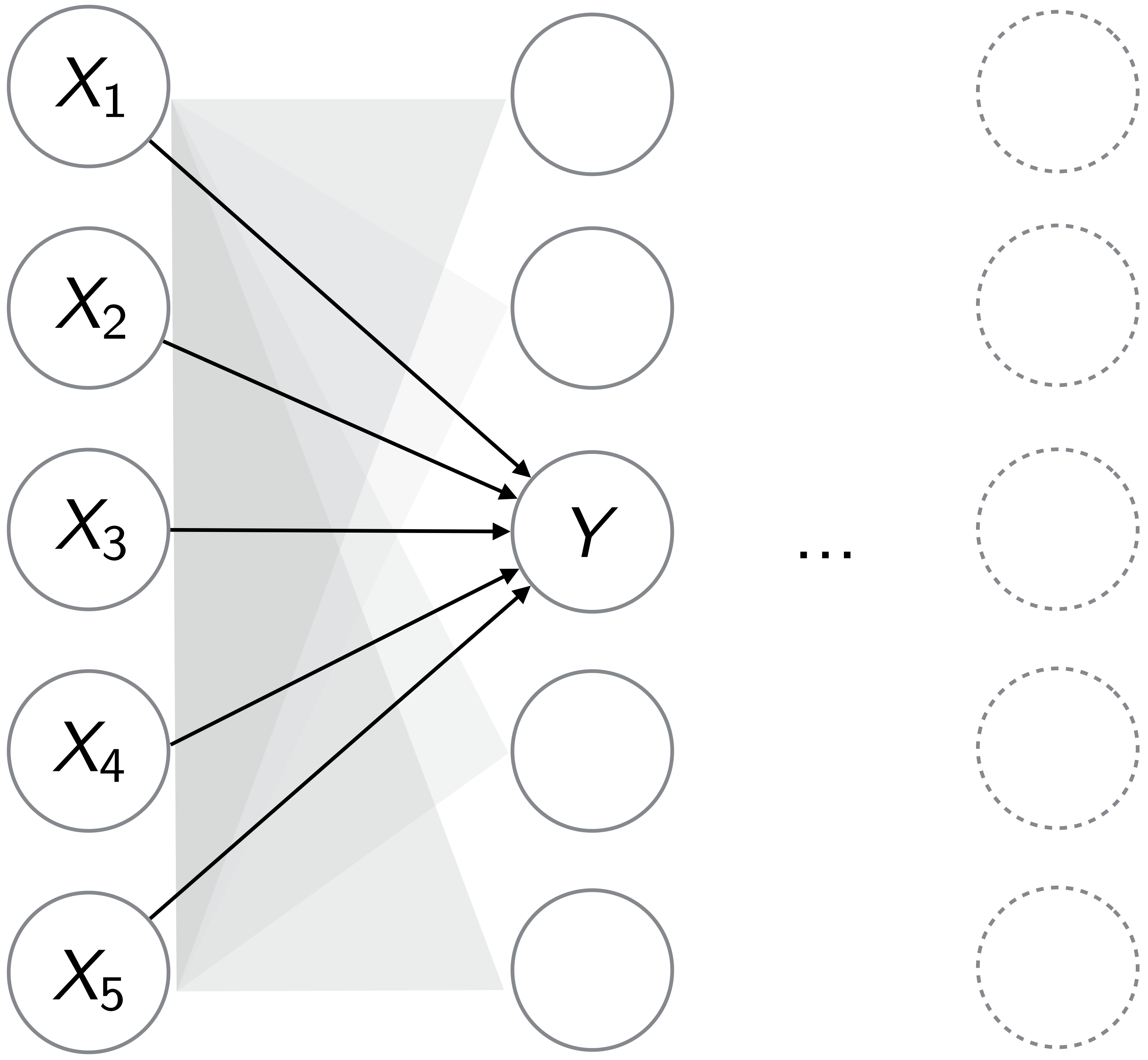
$$p(X) = \prod_{i \in V} p(X_i \mid X_{\mathrm{Pa}(i)})$$

Sigmoid Belief Network

- As considered by Neal (1992)
  - Binary variables
  - Conditional probabilities using logistic model:

$$p(Y_j{=}1 \mid X) = \frac{1}{1 + exp(-\sum_i w_i X_i)}$$

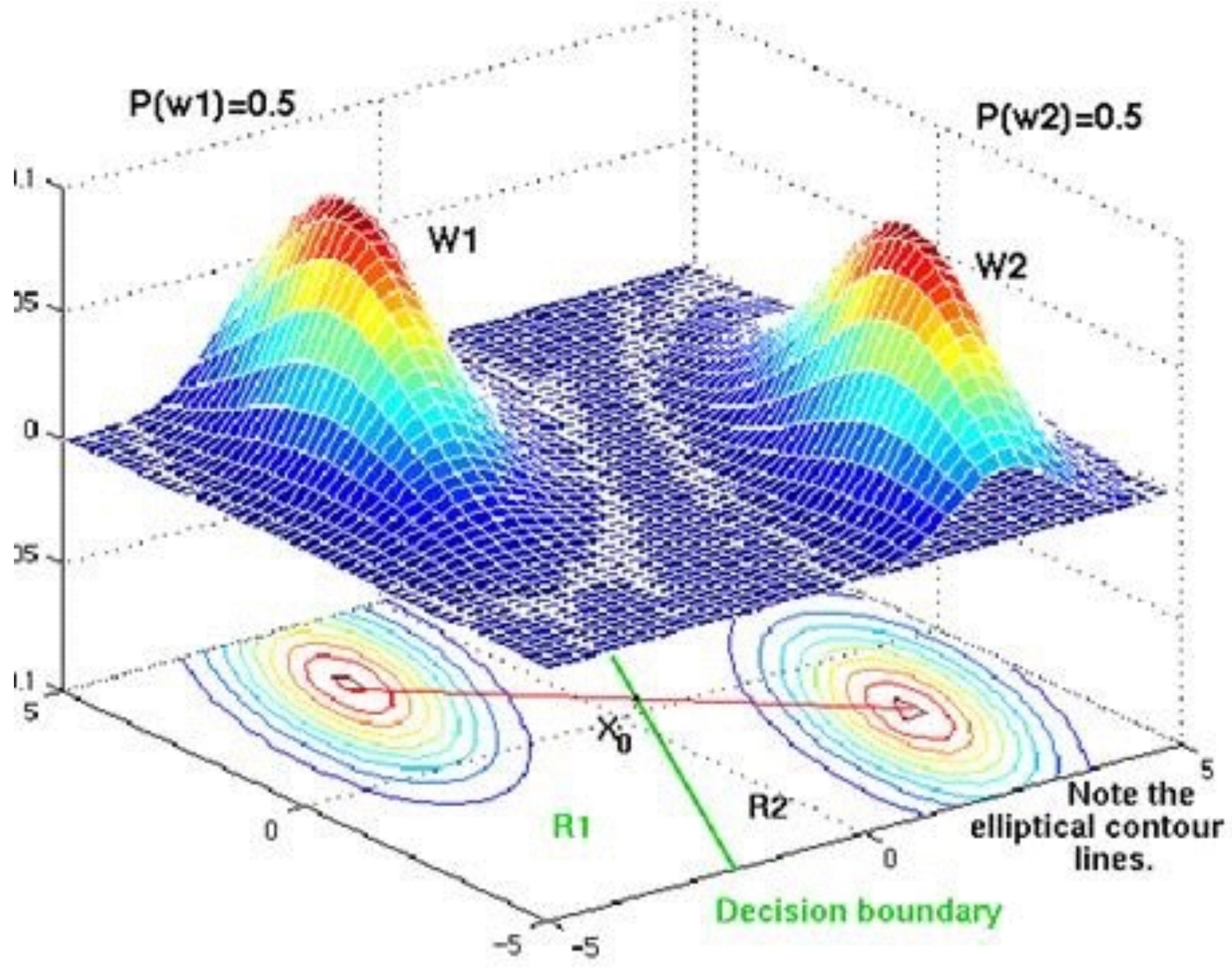
$$p(Y \mid X) = \prod_j p(Y_j \mid X)$$



- Logistic conditional probabilities:
  - the probability model that has linear discriminant function
  - can be also derived assuming the factorization
- Same conditional probabilities in:
  - restricted Bolzman machine, deep Bolzman machine, deep Bayesian network

Logistic Function from Linear Discriminants

- X - observed feature vector
- K in {0,1} - hidden class label (face / not face)



The optimal Bayesian classifier is given by

$$\frac{p(K=1|x)}{p(K=0|x)} \lessgtr \theta$$

Equivalently, with *log-odds*:

$$f(x) := \log p(K=1|x) - \log p(K=0|x) \lessgtr \eta$$

What is the form of conditional distribution $p(K|X)$ such that $f(x)$ is linear: $f(x) = w^\top x$?

Sigmoid Belief Network from Factorization

Consider a joint model $p(X, Y) = p(Y \mid X)p(X)$

Conditional distribution $p(Y \mid X)$ is *strongly conditionally independent* if it factors as:

$$p(y \mid x) = \frac{1}{Z(x)} \prod_{i,j} g_{ij}(x_i, y_j)$$

$$p(y \mid x) = \frac{1}{Z(x)} \exp \sum_{i,j} u_{ij}(x_i, y_j) = \prod_j \frac{1}{Z_j(x)} \exp \sum_i u_{ij}(x_i, y_j) = \prod_j p(y_j \mid x)$$

Any function $u_{ij}(x_i, y_j)$ of binary variables can be written as $u_{ij}(x_i, y_j) = y_j W_{ij} x_j + b_j y_j + c_i x_i + d$

Terms $c_i x_i + d$ cancel in the normalization of $p(Y \mid X)$

$$p(Y_j = 1 \mid x) = \tfrac{1}{Z_j(x)} \exp(\textstyle\sum_i W_{ij} x_j + b_j), \quad p(Y_j = 0 \mid x) = \tfrac{1}{Z_j(x)} \exp(0) = \tfrac{1}{Z_j(x)}$$

$$p(Y_j = 1 \mid x) = \frac{1}{1 + \exp\{-(\sum_i W_{ij} x_j + b_j)\}}$$

Further Topics

- Global conditional independencies — Markov Blanket
- Local conditional independencies — Moral Graph
- Optimal approximations by trees — Chow-Liu trees

- Other names for BN:
  - belief network,
  - directed graphical model
  - (probabilistic network, causal network, knowledge map)

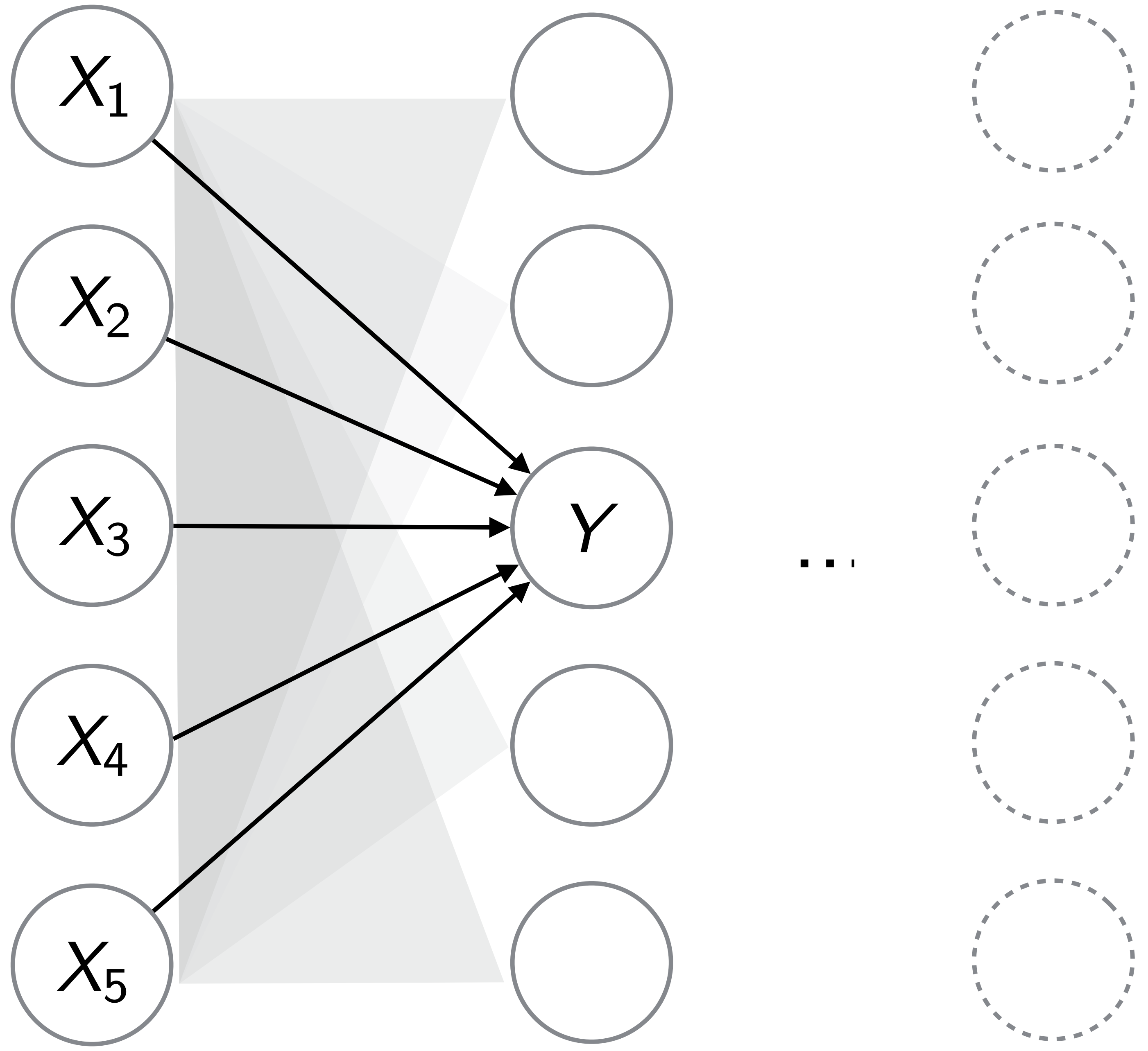
Neural Networks as Graphical Models

Materials: Shekhovtosv, Flach, Busta: “Feed-forward Uncertainty Propagation in Belief and Neural Networks”, 2018

Recall: Sigmoid Belief Network

$$p(Y_j{=}1 \mid X) = \frac{1}{1 + exp(-\sum_i w_i X_i)}$$

$$p(Y \mid X) = \prod_j p(Y_j \mid X)$$



Assume input $X^0 = x^0$ is given,

Model: $p(X^n, X^{n-1}, \ldots, X^1 \mid x^0)$

First level posterior: $p(X^1 = 1 \mid x^0) = \mathcal{S}(W^1 x^0)$

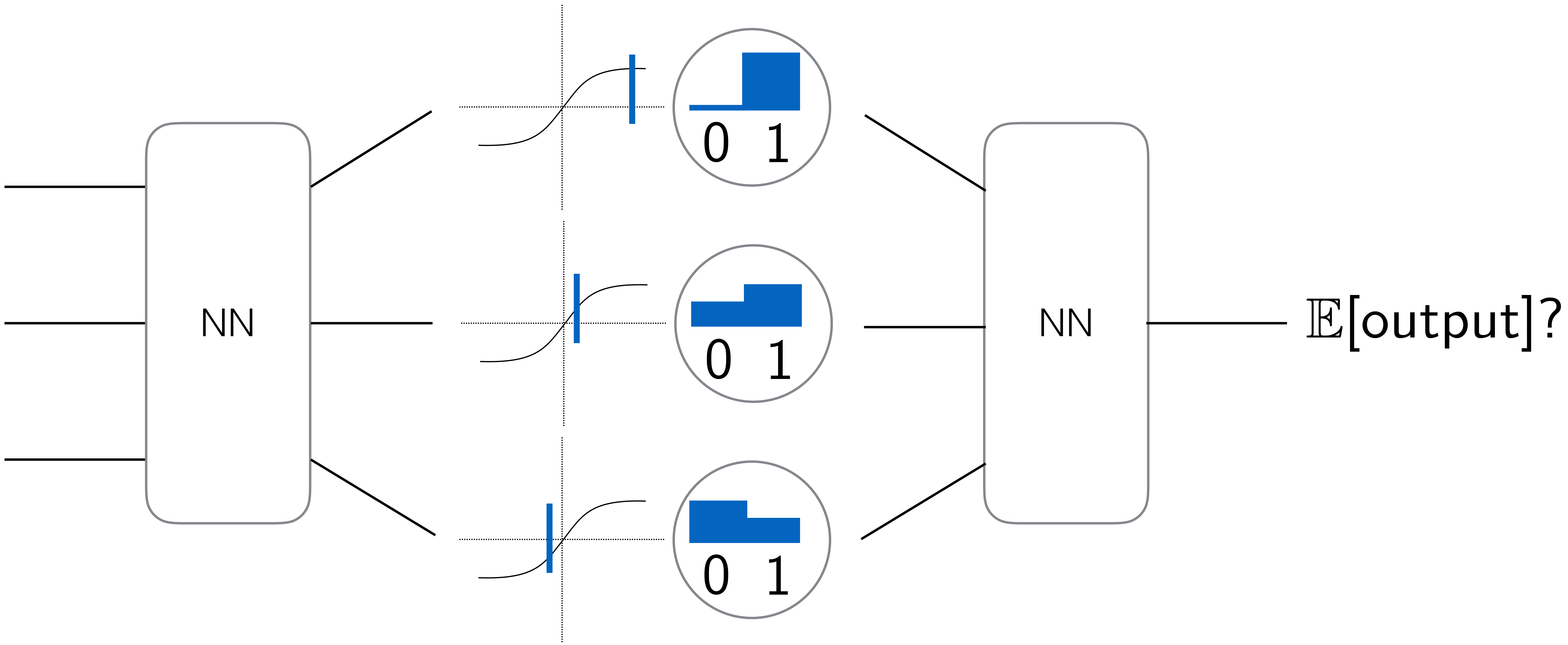
Second level posterior: $p(X^2 = 1 \mid x^0) = \sum_{x^1} p(X^2 = 1 \mid x^1) p(x^1 \mid x^0)$

...

Network output: $p(X^n \mid x^0) = \mathbb{E}_{X^1, X^2, \ldots X^{n-1}} p(X^n, X^{n-1}, \ldots, X^1 \mid x^0)$

In Sigmoid NNs Expectations Replaced

- Sigmoid output is often interpreted as probability (e.g. part detectors, hierarchy of logistic models)
- NNs do not compute the expectation (substitute it inside)



- Use cases for computing the expectation:
  - Improve stability (robustness) of neural networks
  - Training networks with binary activations / weights

Sigmoid NN as Approximation

For two consecutive layers X, Y

Apply the first order Taylor approximation for the moments of functions of random variables:

$p(Y = 1 \mid x^0) = \mathbb{E}_{X \sim P(X \mid x^0)}[\mathcal{S}(w^\mathsf{T} X)] \approx \mathcal{S}(E_X[w^\mathsf{T} X]) = \mathcal{S}(w^\mathsf{T} E_X[X])$

Note that for Bernoulli variables $E_Y[Y] = p(Y{=}1 \mid x^0)$.

We obtained standard NN propagation rules where activations are the "means"

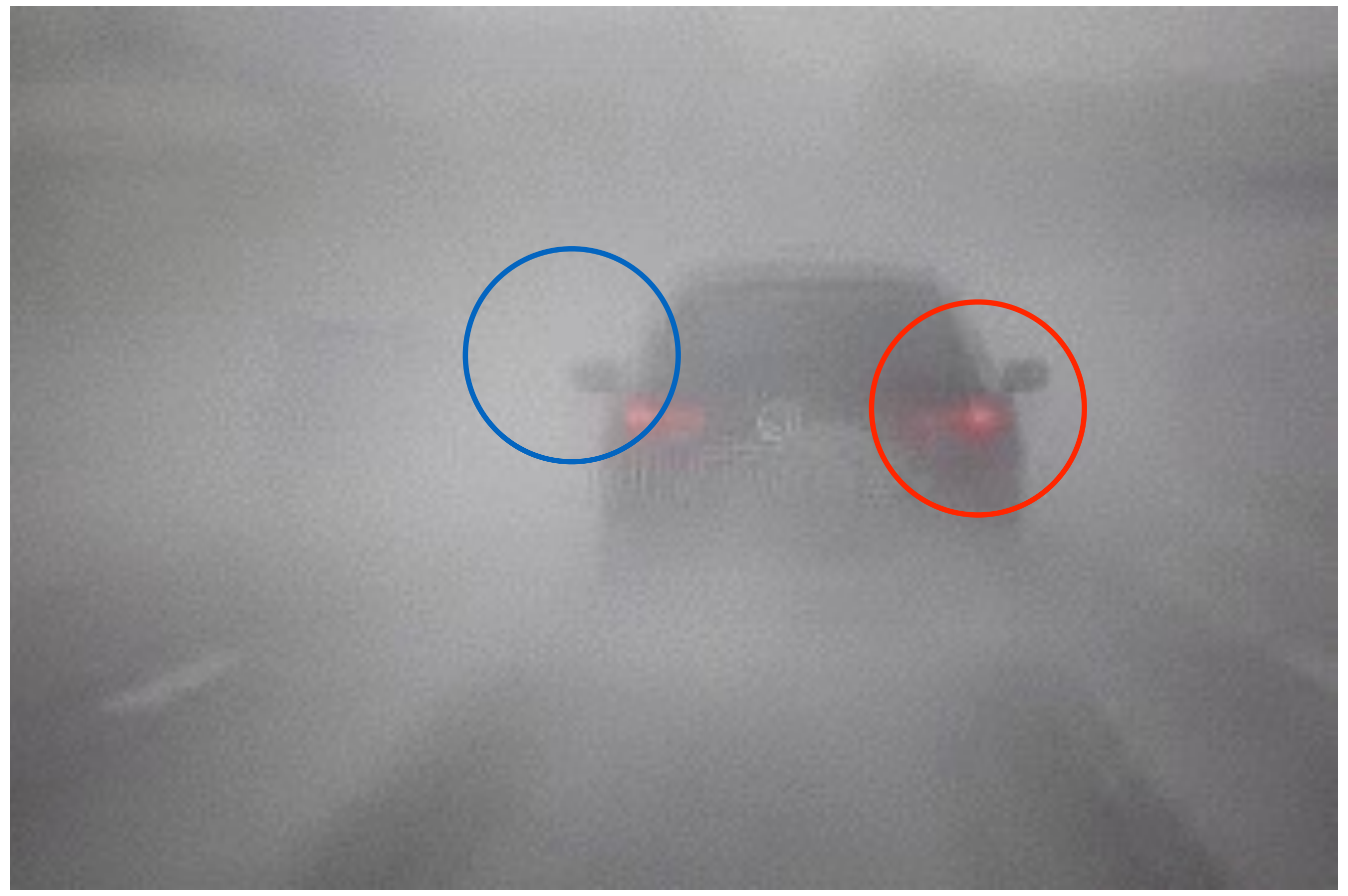
- Is this difference important?

Example: Logic Gates

- For example, composition of parts:

  X_1 =1 if seeing "car mirror"

  X_2 =1 if seeing "car stop light"



- If X_1=1 with probability 0.3 and X_2=1 with probability 0.2 what is the probability that booth are present: X_1&X_2?

Let us fit logistic model

$p(Y=1 \mid X) = \mathcal{S}(a(X_1 + X_2) + b)$

And compare $\mathbb{E}_X \mathcal{S}(a(X_1 + X_2) + b)$

with $\mathrm{AP1} = \mathcal{S}(a\mathbb{E}_X[X_1 + X_2] + b)$

$p(X_1{=}1)$	$p(X_2{=}1)$	$\mathbb{E}[X_1 \wedge X_2]$	$\mathbb{E}[Y]$	AP1
0	0	0	0.00015	0.00015
0	1	0	0.05	0.05
1	1	1	0.95	0.95
0.25	0.25	0.0625	0.077	0.0027
0.5	0.5	0.25	0.26	0.05
0.75	0.75	0.56	0.55	0.5

Parameters a, b are set such that: $S(a(1+1)+b)>0.95$, $S(a(0+0)+b)<0.05$

Logistic model is ok, but NN severely underestimates the probability of X_1 AND X_2.

Similarly, for X_1 OR X_2, NN overestimates the probabilities.

Could it Be One of the Reasons for Instability?

original semantic segmentation framework

compromised semantic segmentation framework

Houdini: Fooling Deep Structured Prediction Models, Cisse et al. Cisse 2017

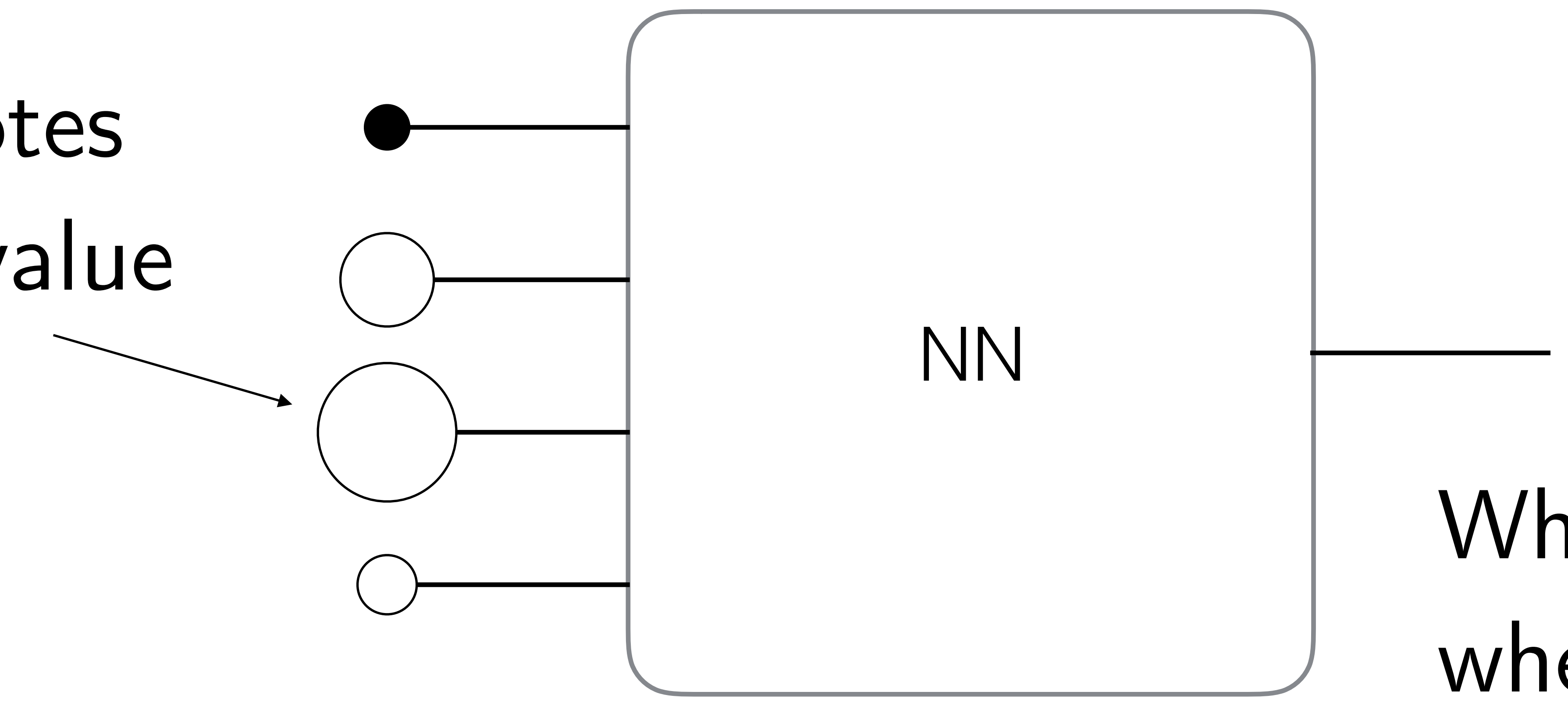
CNNs are sensitive to random noise and to adversarial attacks (structured noise optimized to compromise a given network)

The other reasons could be:
Lack of regularization (overfitting)?
CNN structure?

Uncertain / Missing Inputs

- Uncertain input may be:
  - Sensor noise (noisy image, lidar, computational sensors, etc.)
  - an input from other networks

A circle denotes
an uncertain value

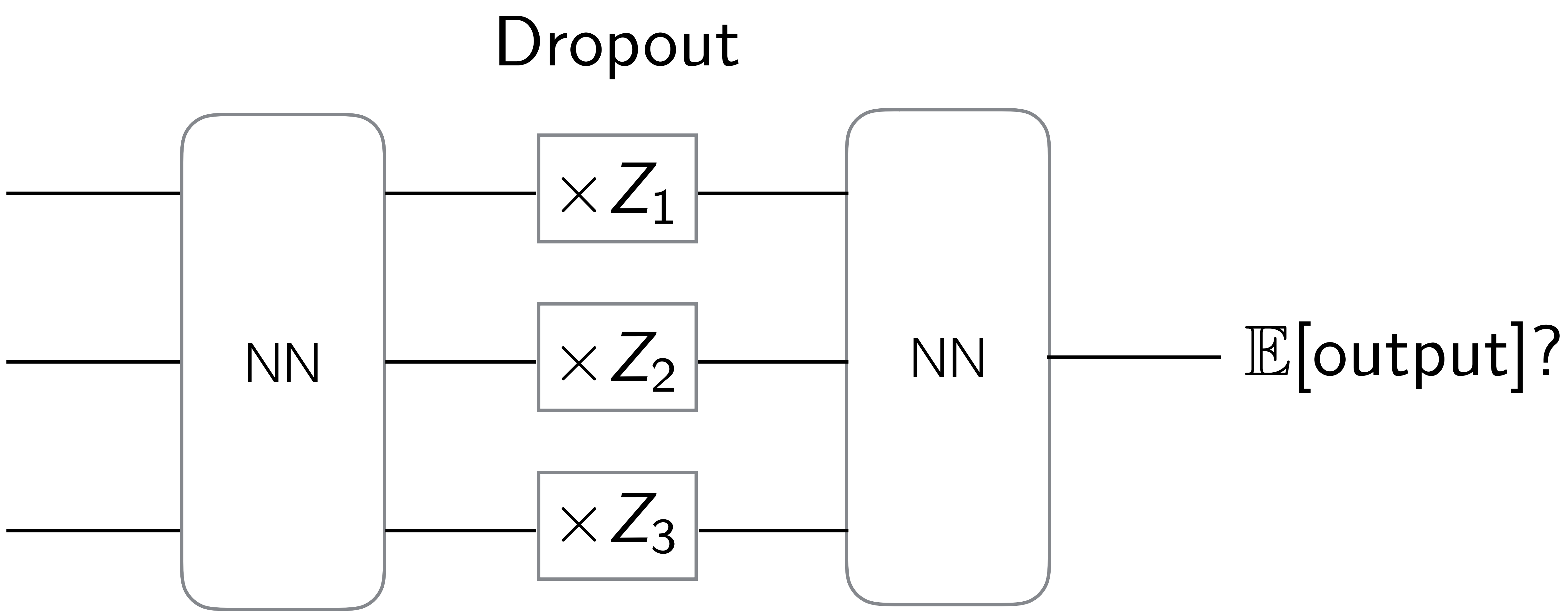


$\mathbb{E}$[output]?

What is the average output
when input is random with
the given uncertainty?

Networks with Dropout

- Known[1] to improve generalization of NNs
- Usually sampled at training time and replaced with means at test time



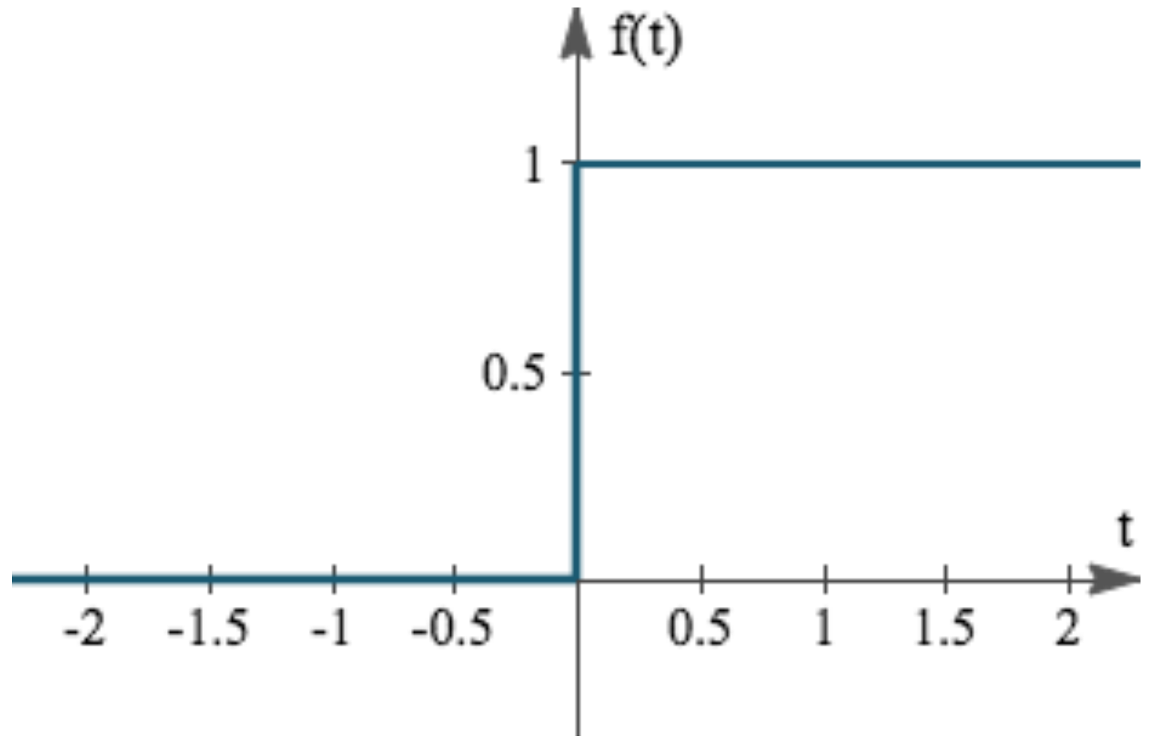
$$Z_i \sim \text{Bernoulli}(0.3)$$

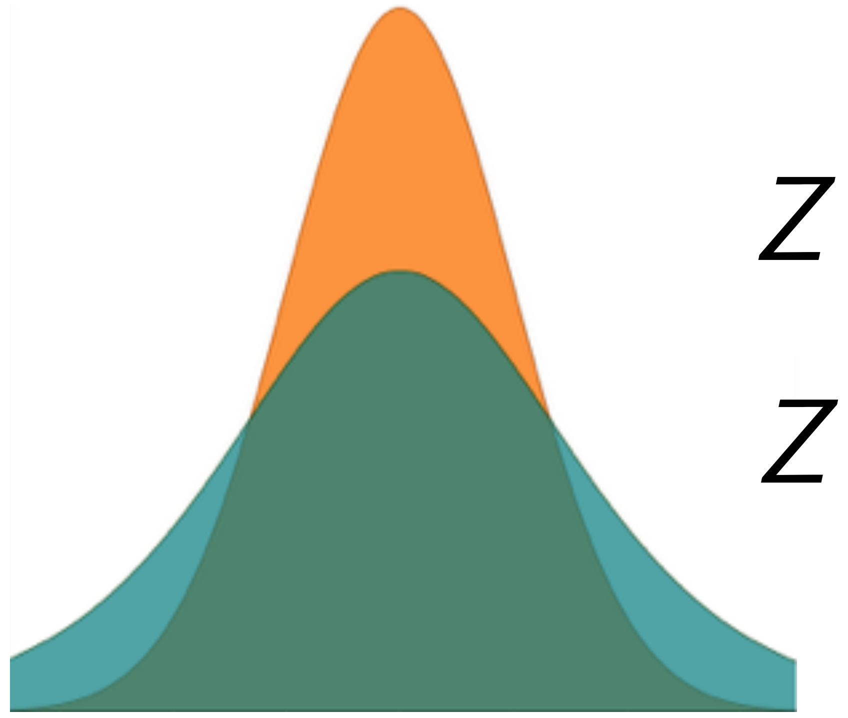
- Another case for statistical treatment

[1] Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting
[2] Wang, S. and Manning, C. (2013). Fast dropout training. In ICML

Equivalence of Injected Noise and Probabilistic Models

$[\![x \geq 0]\!]$

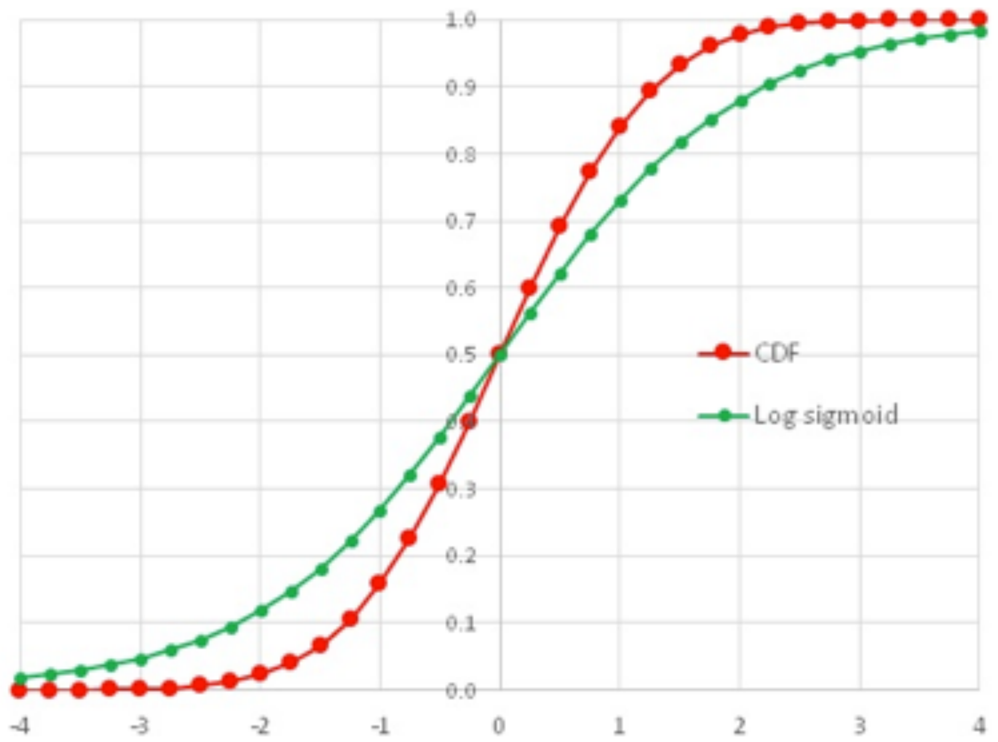




$Z \sim$ logistic

$Z \sim$ normal

$\mathbb{E}_Z[\![x + Z \geq 0]\!]$



More generally, let $Y = f(X, Z)$

Then c.d.f. of Y given X, $F_Y(y \mid X) = \mathbb{E}_Z[\![f(X, Z) \leq y]\!]$

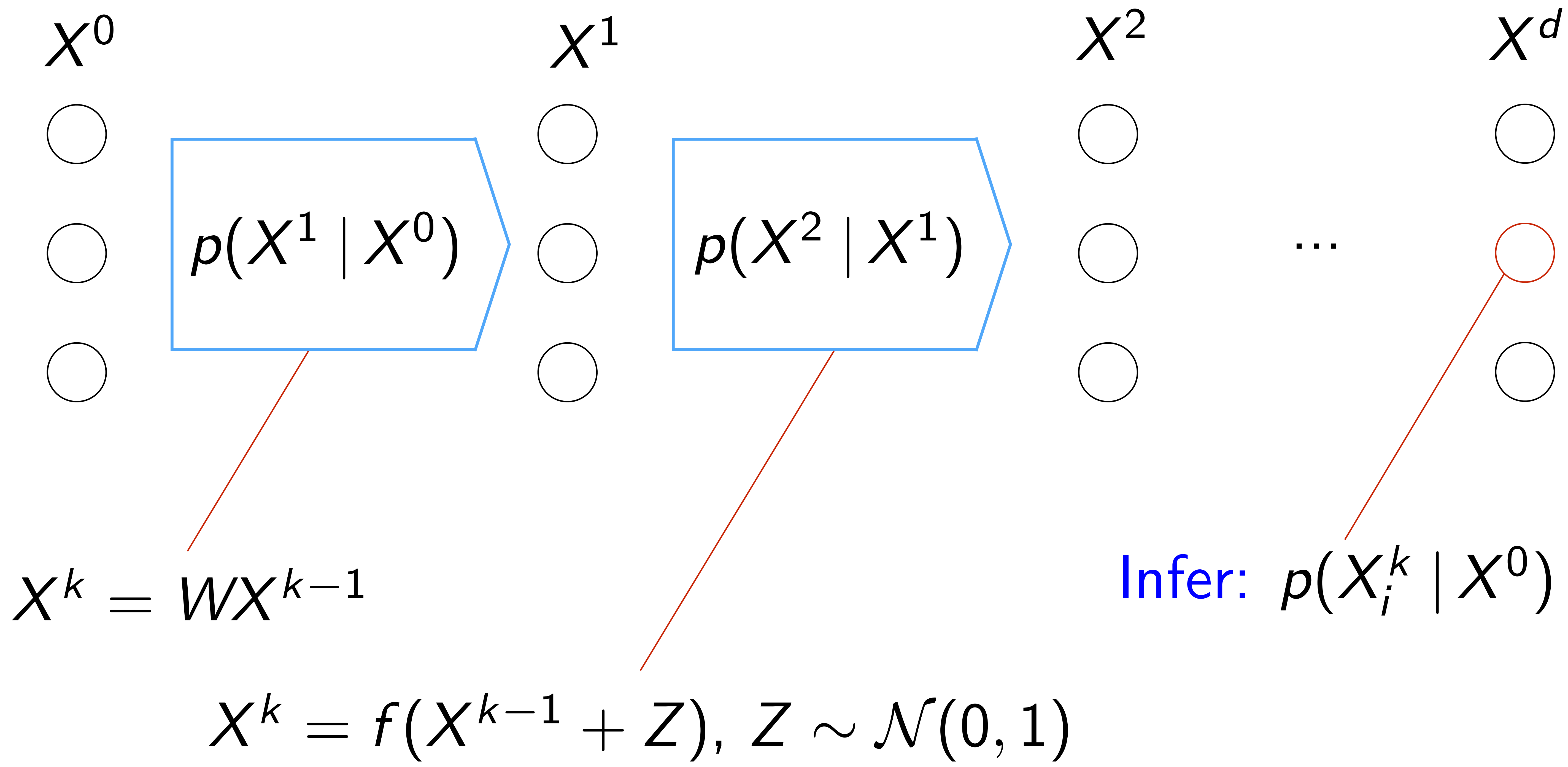
We can in principle reconstruct $p(Y \mid X)$

In dropout training objective we have something like:

$$\mathbb{E}_Z\Big[\log \mathrm{softmax}(W^n \mathrm{ReLU}(W^{n-1} \mathrm{ReLU}(\ldots W^1 x^0 \ldots) Z_{n-2}) Z_{n-1}))$$

NN as Bayesian / Belief Network

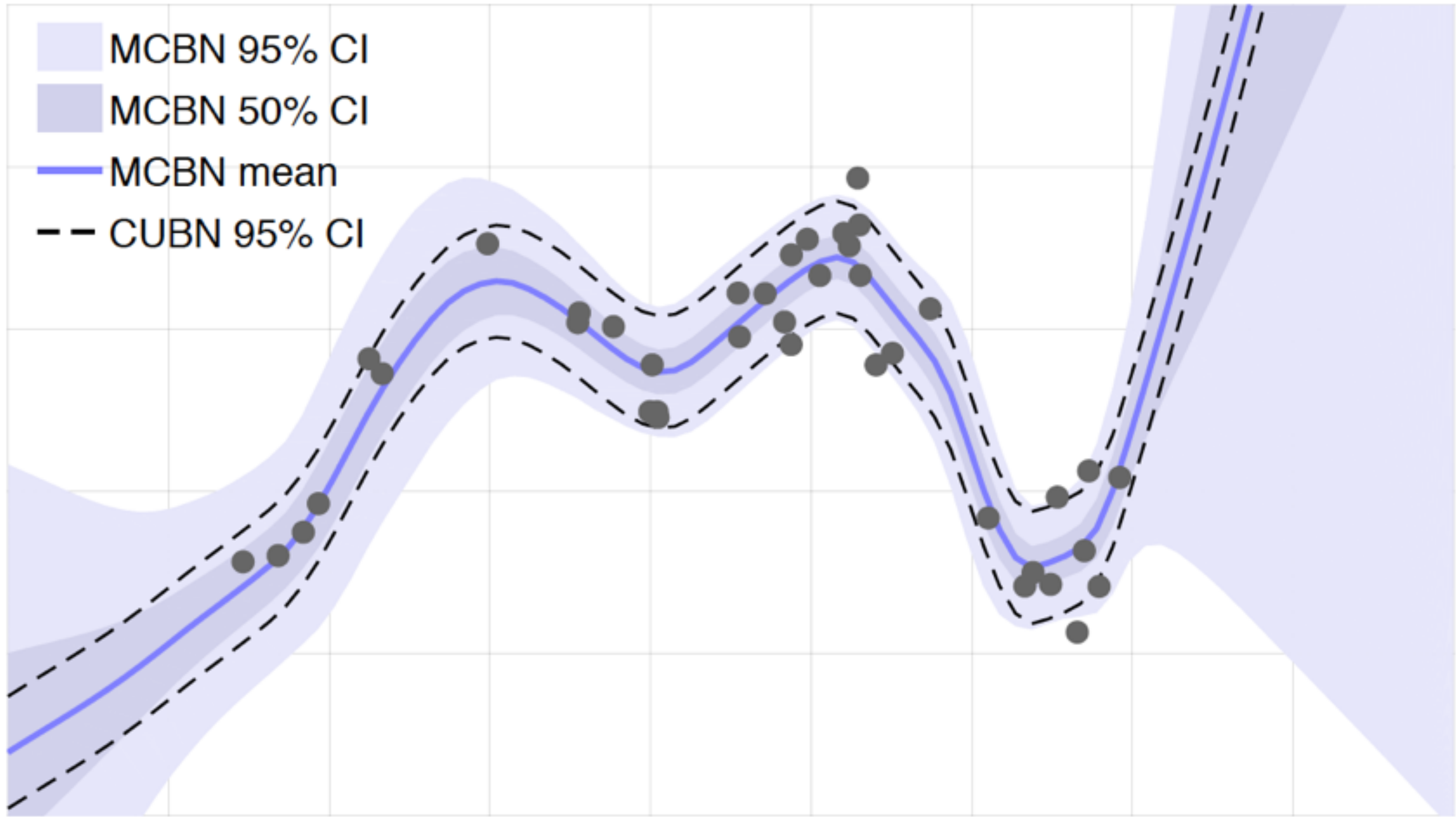
- All neurons are random variables
- Feed-forward network = directed graphical model



Output Uncertainty

- Goal: if we take into account all stochastic components, we should be able:
  - in classification: compute better likelihoods (confidence estimates)
  - in regression: output with uncertainty

Something like this:

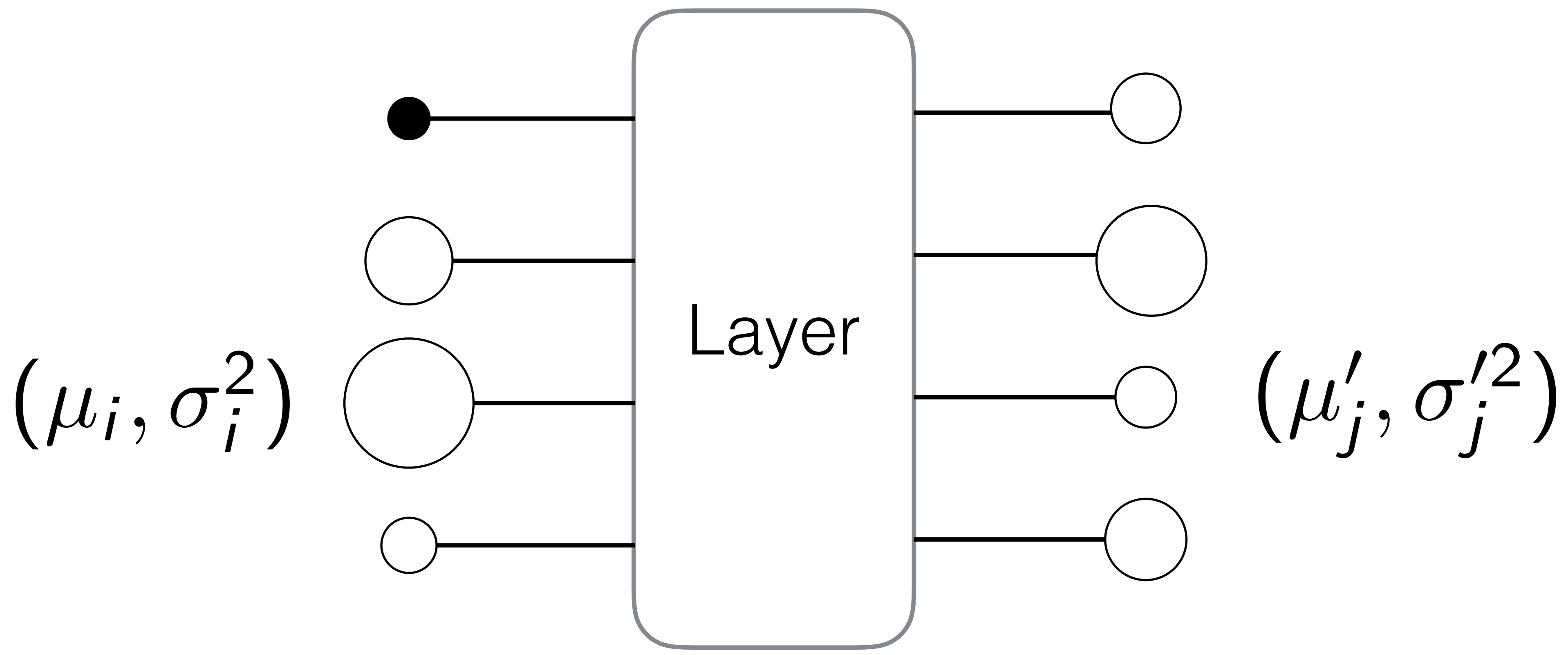


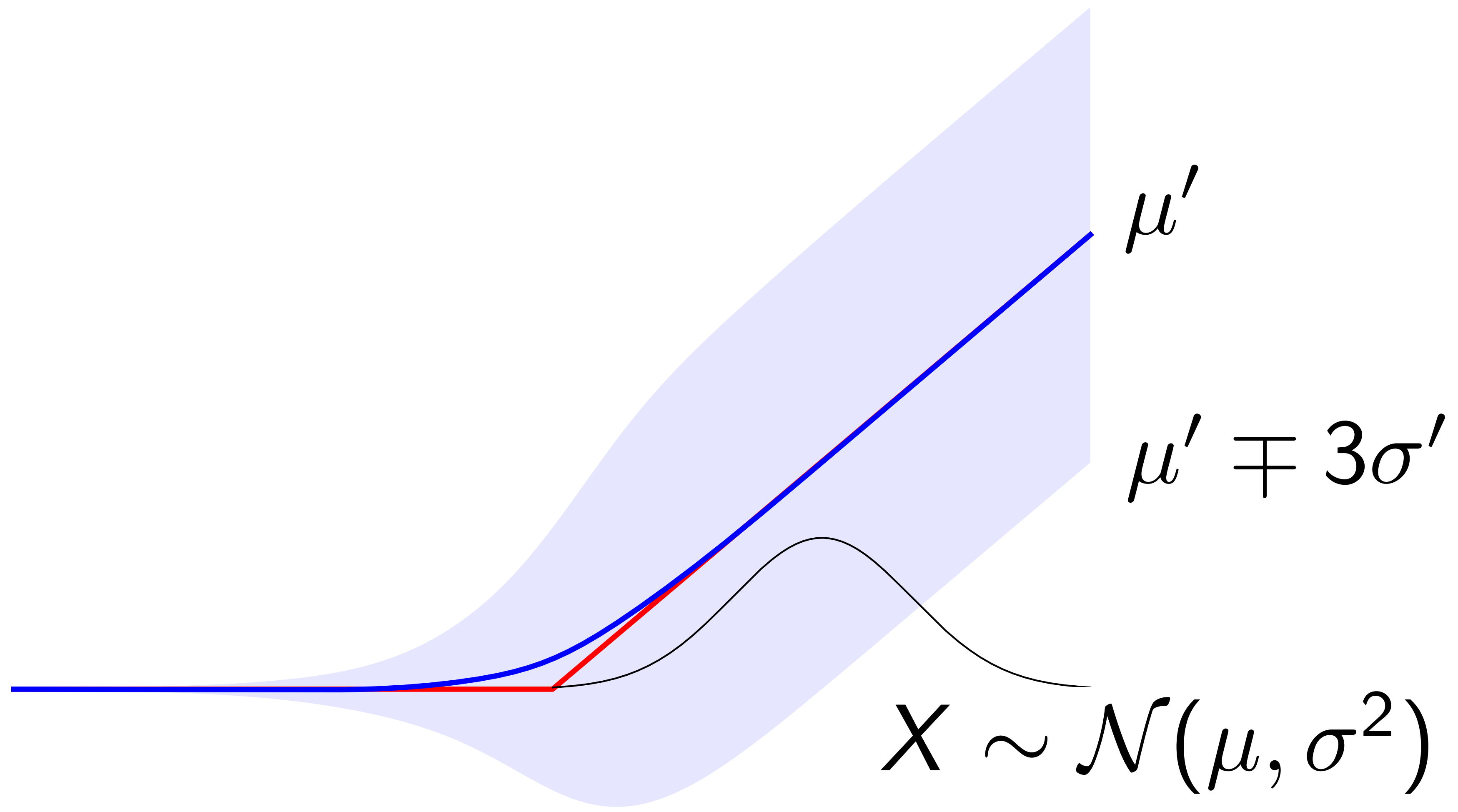
Sampling techniques [Some paper]

Several methods exist, but not widely used and many open research questions

Feed-forward Uncertainty Propagation

General diagram for all layers





Linear: $Y = w^\top X$

$$\mu' = \mathbb{E}[Y] = w^\top \mathbb{E}[X] = w^\top \mu,$$

$$\sigma'^2 = \sum_{ij} w_i w_j \mathrm{Cov}[X] \approx \sum_i w_i^2 \sigma_i^2,$$

ReLU: $Y = \max(X, 0)$

Assume $X \sim \mathcal{N}(\mu, \sigma^2)$

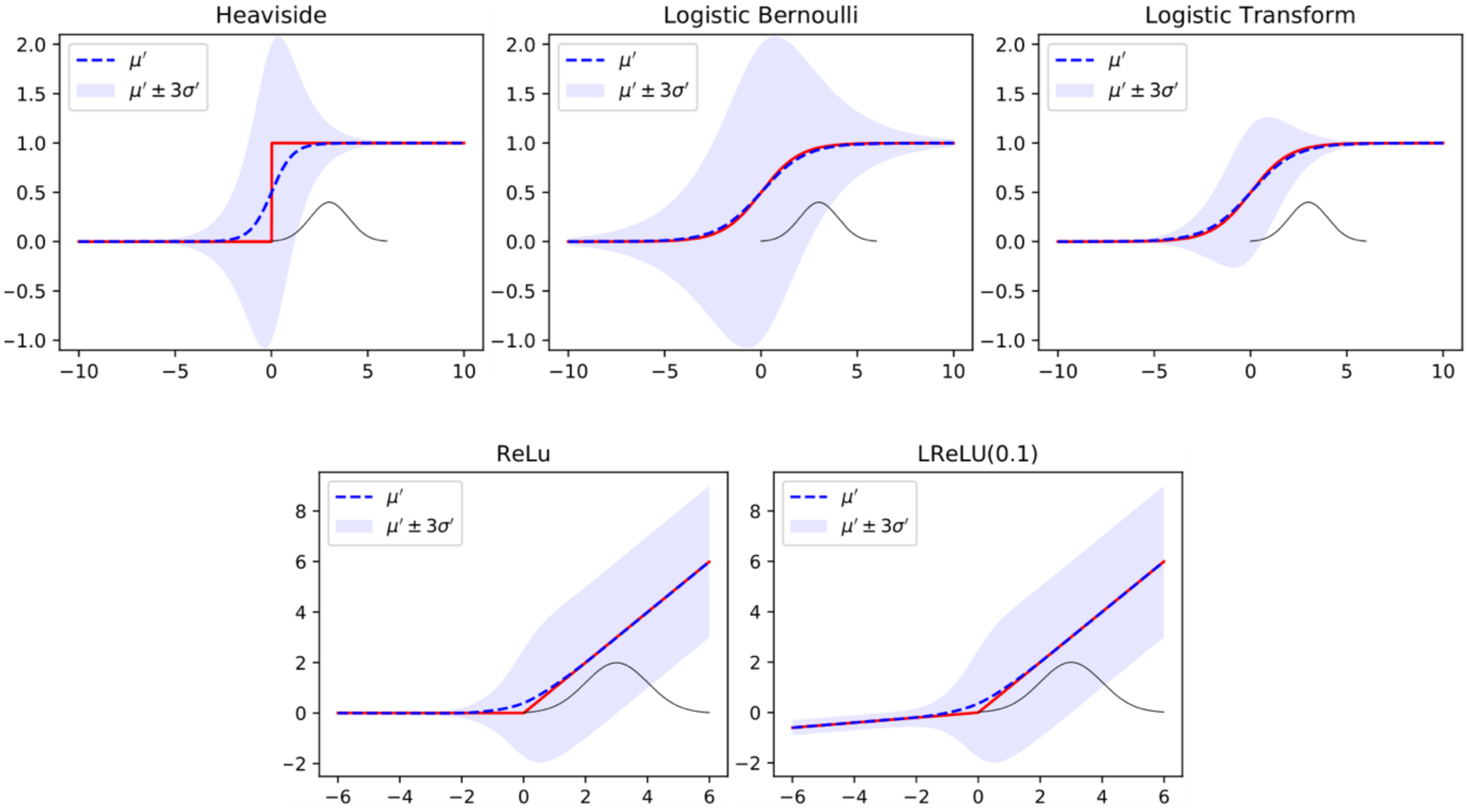
$\mu' = \int_{-\infty}^{\infty} p(X) f(X) dx$

$\sigma' = \int_{-\infty}^{\infty} p(X) f(X)^2 dx - \mu'^2$

- Also supporting: sigmoid, softmax, max-poolig, maxOut, dropout, ...

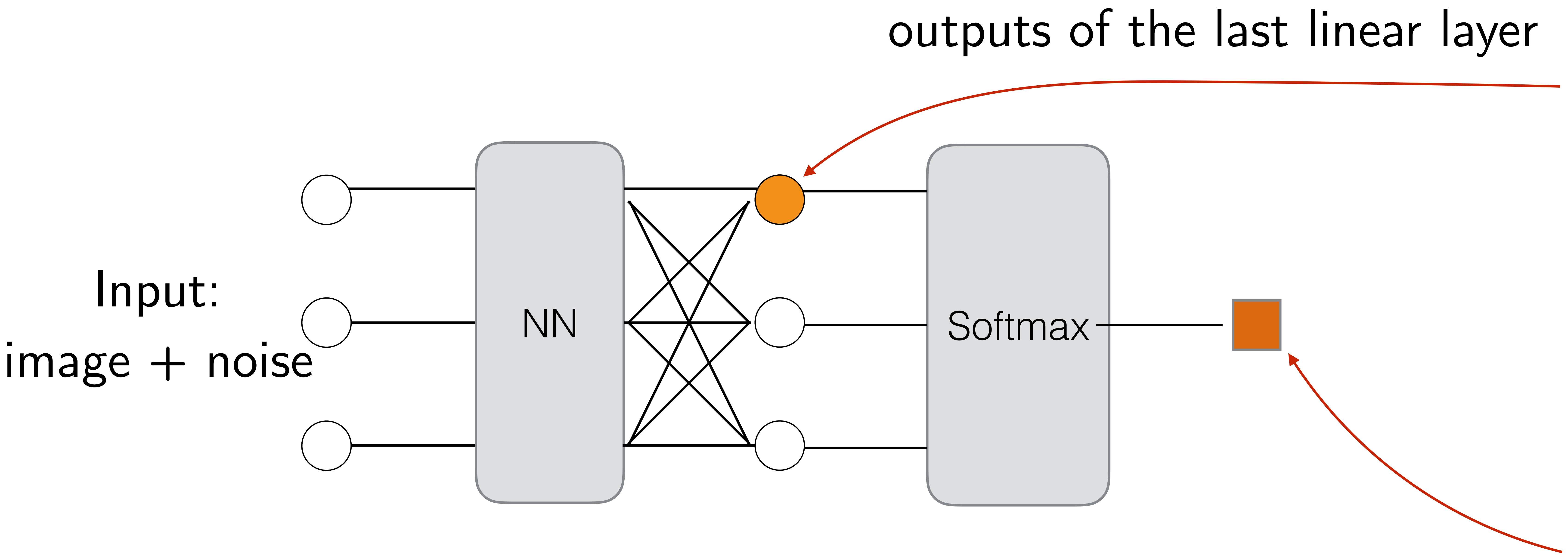
Different Coordinate-wise Functions

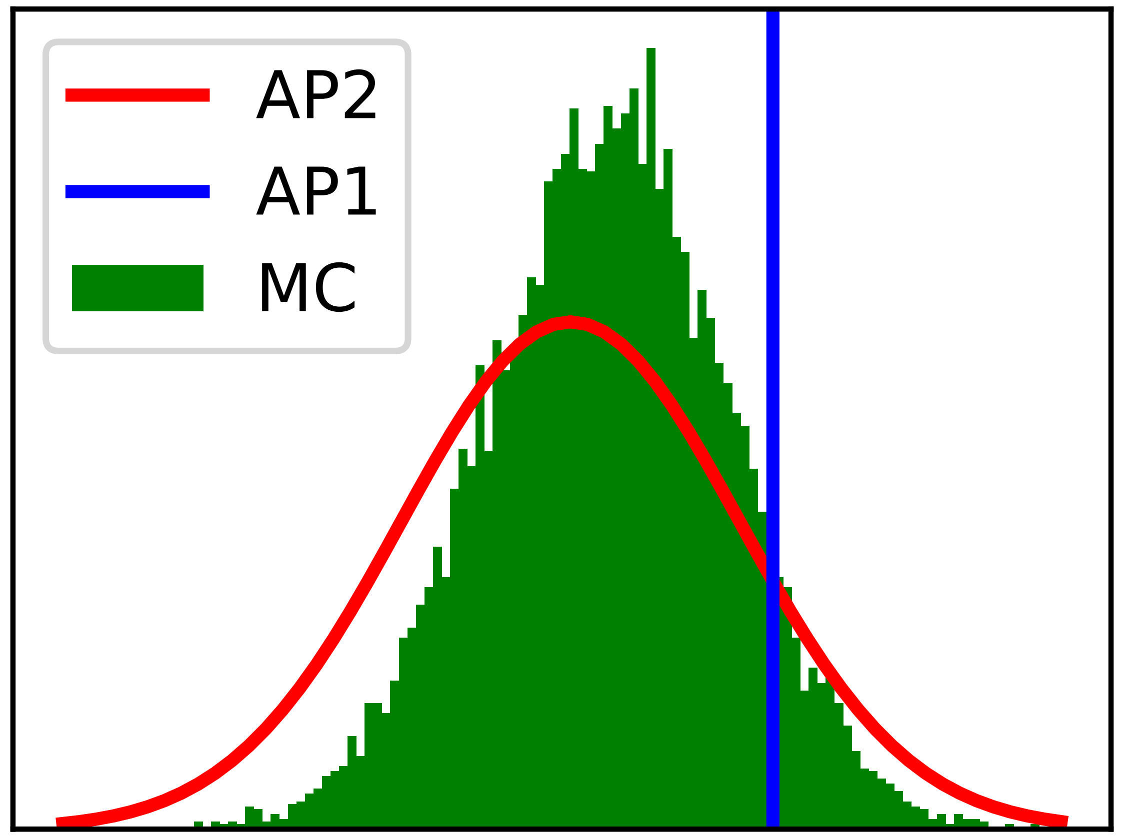
- Expectations are always smooth



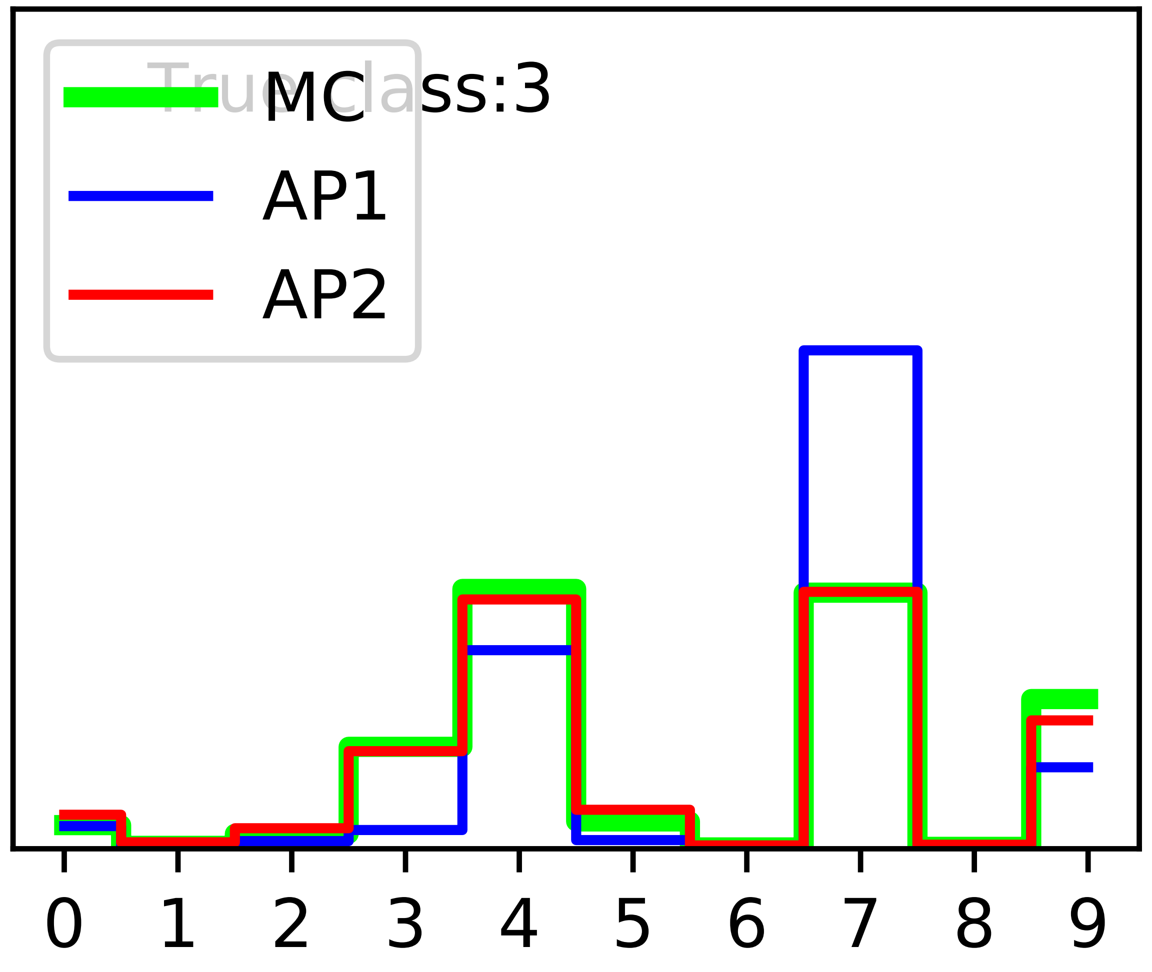
Propagation Methods: Example

- AP1: take clean image and propagate with standard rules
- MC: take several samples of noise and collect statistics from propagating image+noise
- AP2: propagating mean and variance





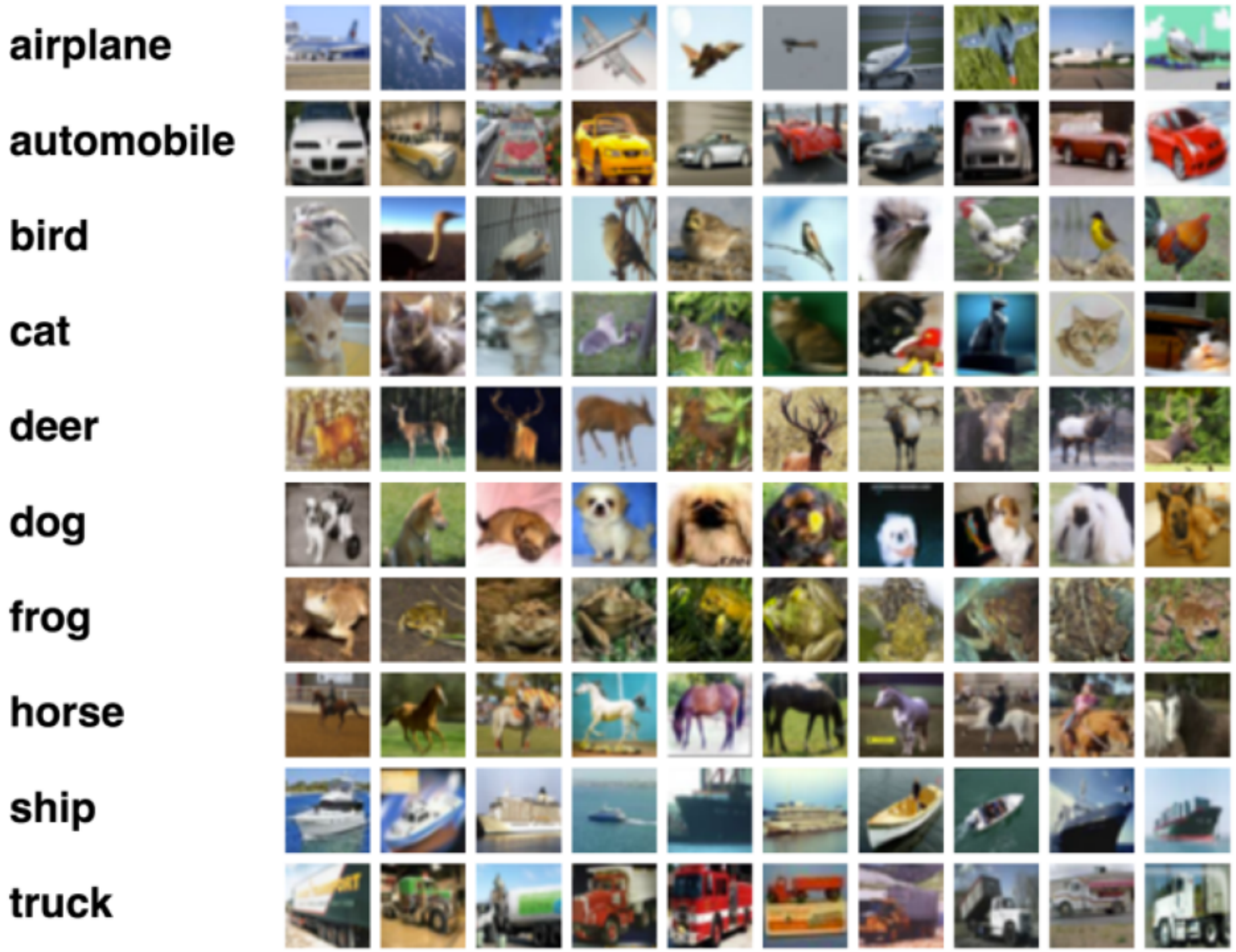
posterior class probability



Under noisy input, estimates may differ
If the output variance is low - standard method is Ok,
Bit if it is high we would not know about it

Experiments on CIFAR-10

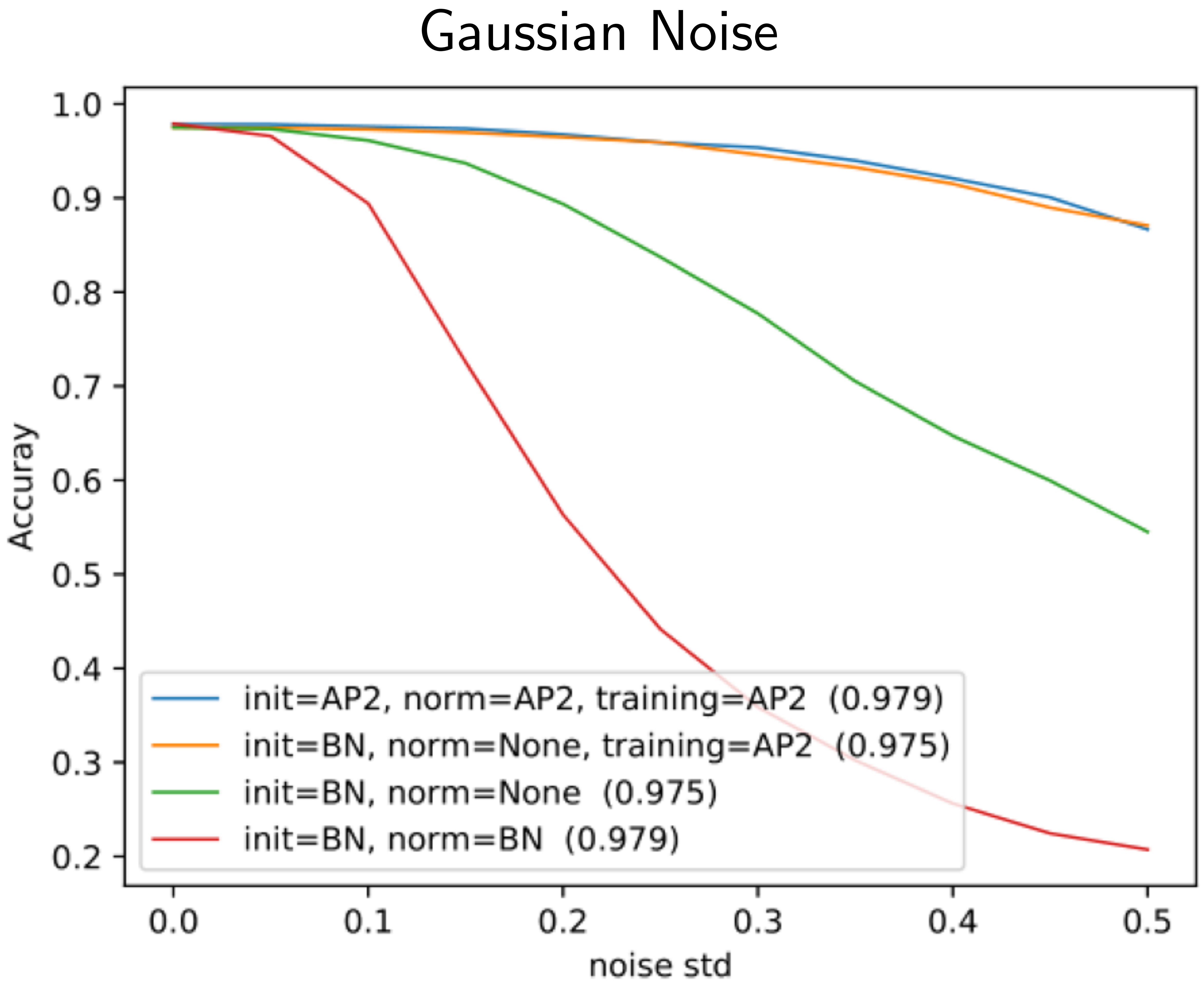
Data: CIFAR10

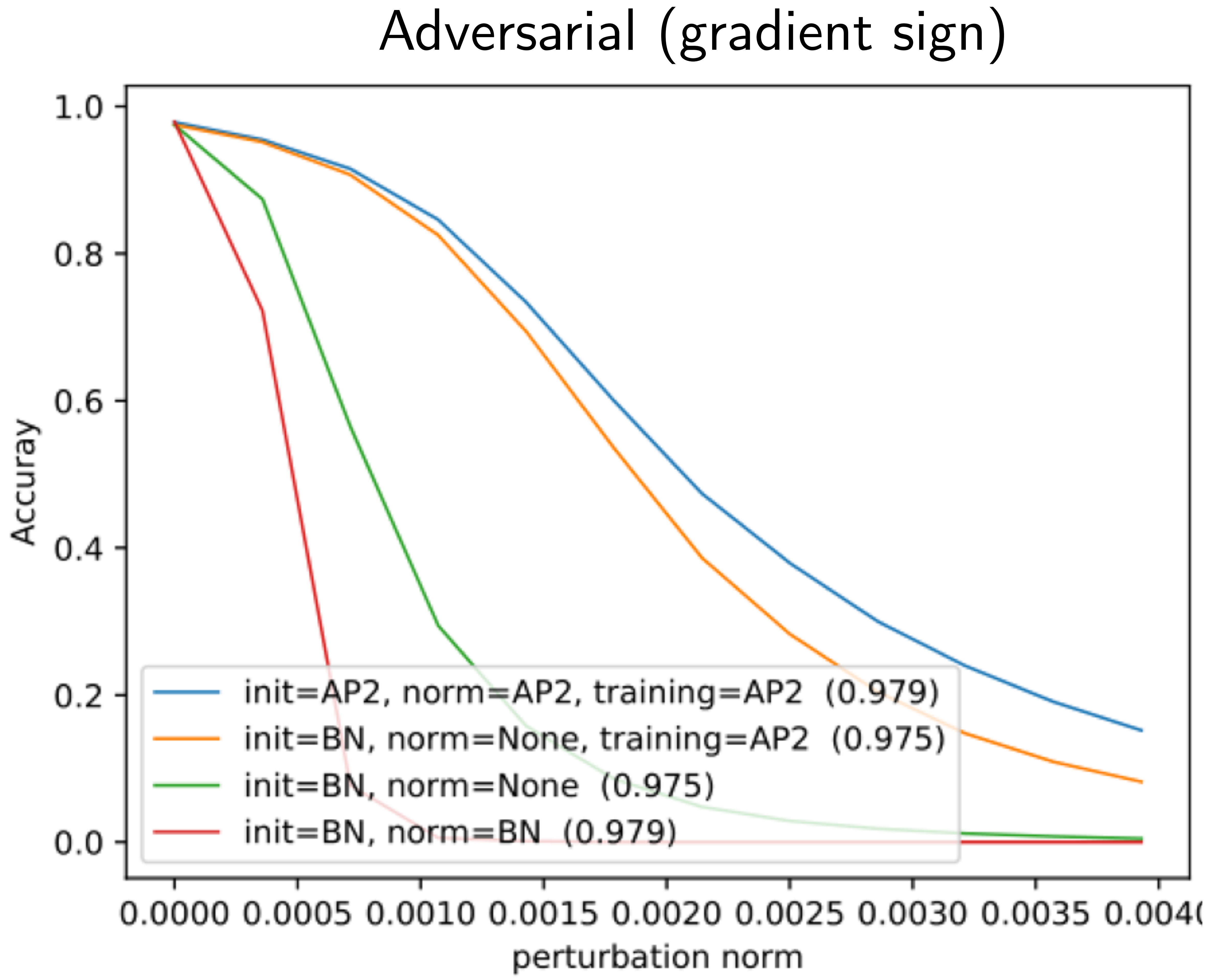


Network: 9 convolutional layers + last layer: average pooling, softmax

Better Stability

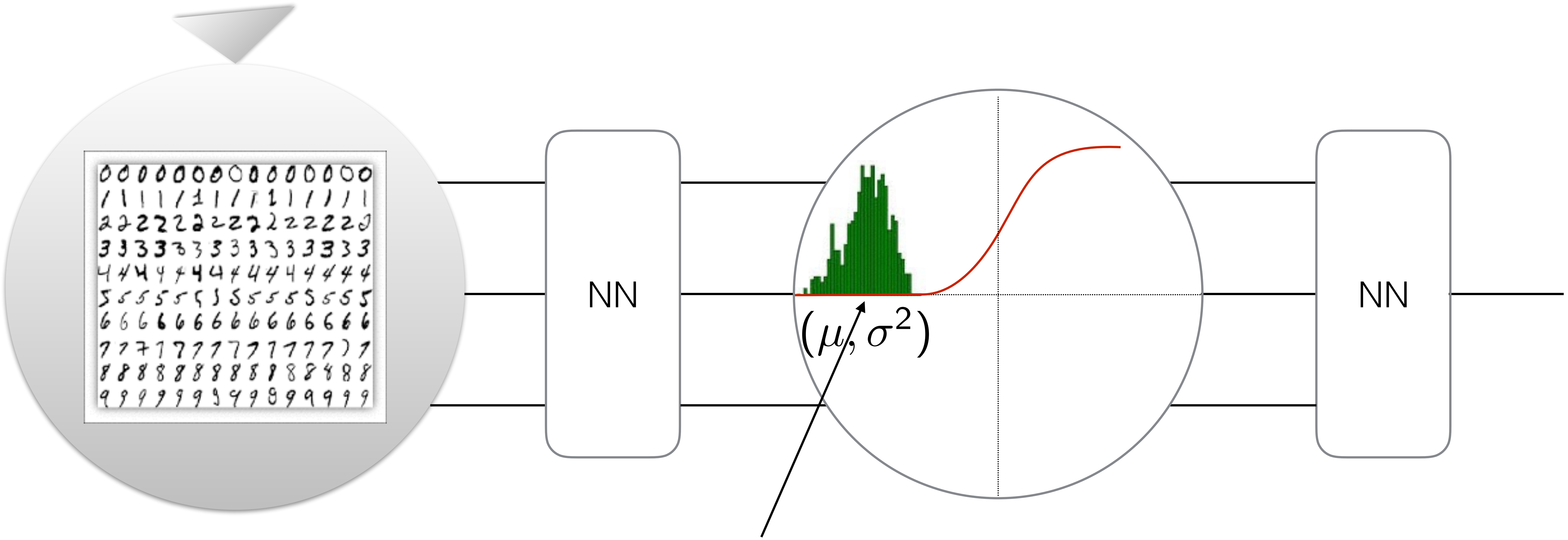
- Currently only for shallow networks, working on improving it





Statistics over the Dataset

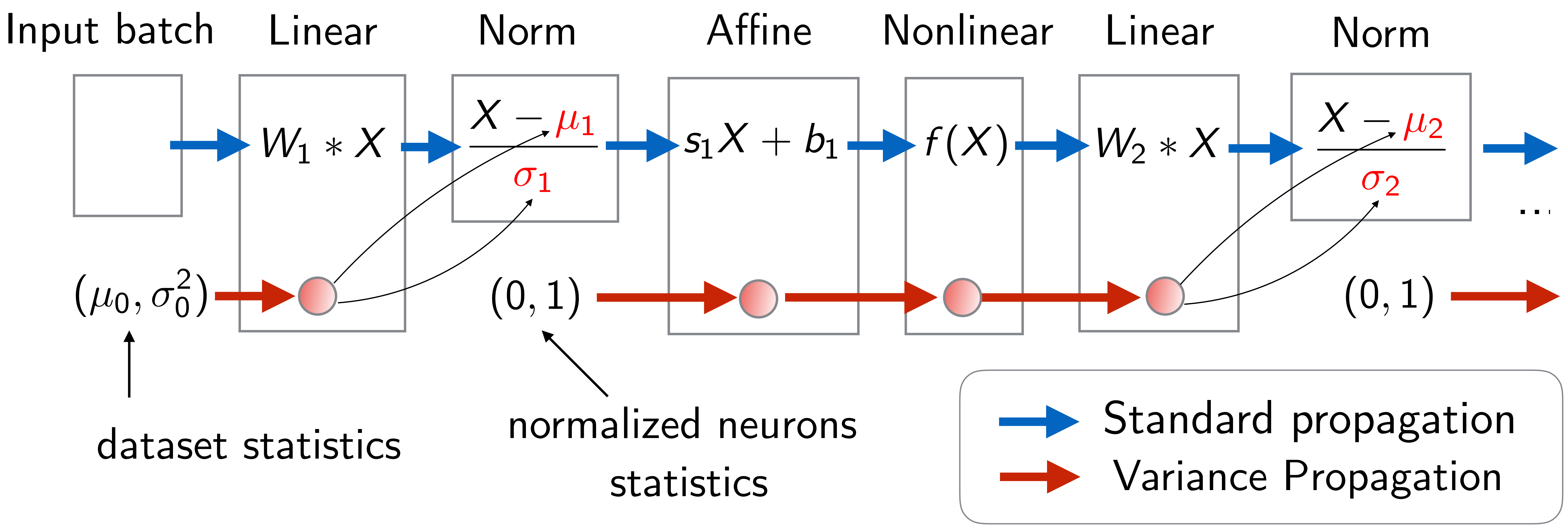
- Problem:
  compute expectations of neurons (mean and variance) over the dataset
- Used for: (same as in Batch Normalization)
  - initialization (start in a non-saturated regime)
  - normalization (a reparametrization better conditioning gradient descent)

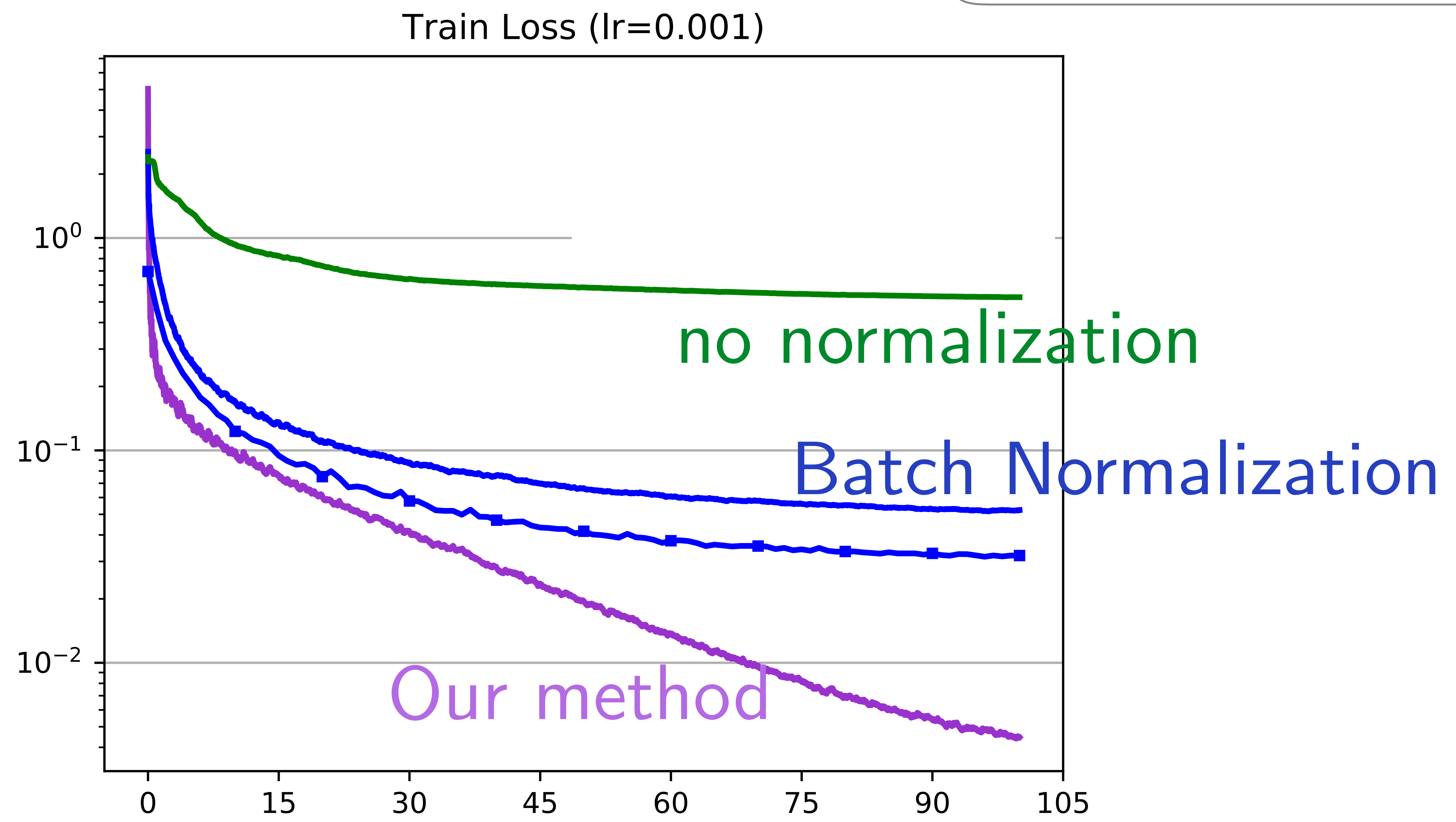


Poor initialization: all inputs to a neuron are in a saturated part

Statistics for Normalization

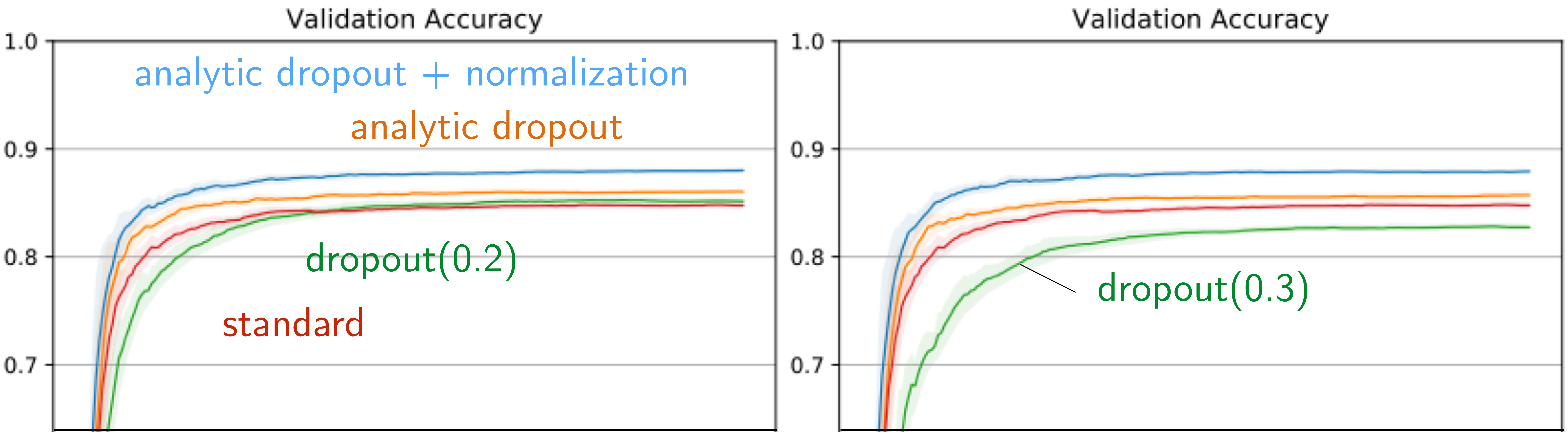
- Shekhovtsov and Flach: Neural Network Normalization using Analytic Variance Propagation





Analytic Dropout

- Can give a better generalization than standard dropout and trains faster
- Related work: Wang and Manning "Fast dropout training"



(BN performs better in this plot)

Take Away Message

- Lots of things to improve in NNs understanding them as probabilistic models
  - uncertain inputs, stability of NNs under perturbations
  - uncertain outputs for regression
  - initialization and normalization
  - improving training with dropout and other noisy regularizers
  - generative models
  - better learning models

Variational Bayesian Learning

Maximum Likelihood

Let x be an input and y the prediction or class label we want to recognize.

Consider a conditional model $p(y \mid x; \theta)$ parametrized by θ.

Let $D = \{(x^t, y^t) \mid t = 1, \dots T\}$ be a set of training samples.

Recall the maximum likelihood approach:

- Training: find the maximum conditional likelihood estimate of θ:

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \prod_{t} p(y^t \mid x^t; \theta)$$

- Testing: recognize new input x using $\hat{\theta}$:

$$y = \operatorname*{argmax}_{y} p(y \mid x; \hat{\theta})$$

- The confidence is given by the posterior $p(y \mid x; \hat{\theta})$

Bayesian Learning

Bayesian approach

- Consider θ as a random variable, with a priori distribution $p(\theta)$
- The conditional model becomes $p(y \mid x, \theta)$
- Training: the posterior estimate of θ given D is:

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{p(D)} = \frac{\prod_t p(y^t|x^t, \theta)p(\theta)p^*(x)}{p(D)},$$

  where $p^*(x)$ is the true distribution of inputs, which we will not be estimating and assume that x is independent of θ.

- Up to normalization: $p(\theta \mid D) \propto \prod_t p(y^t|x^t, \theta)p(\theta)$. Can compute for a given θ using all the data.
- Testing: given x, integrate out θ:

$$p(y \mid x) = \int p(y \mid x, \theta)p(\theta \mid D)d\theta \quad \text{— in general intractable}$$

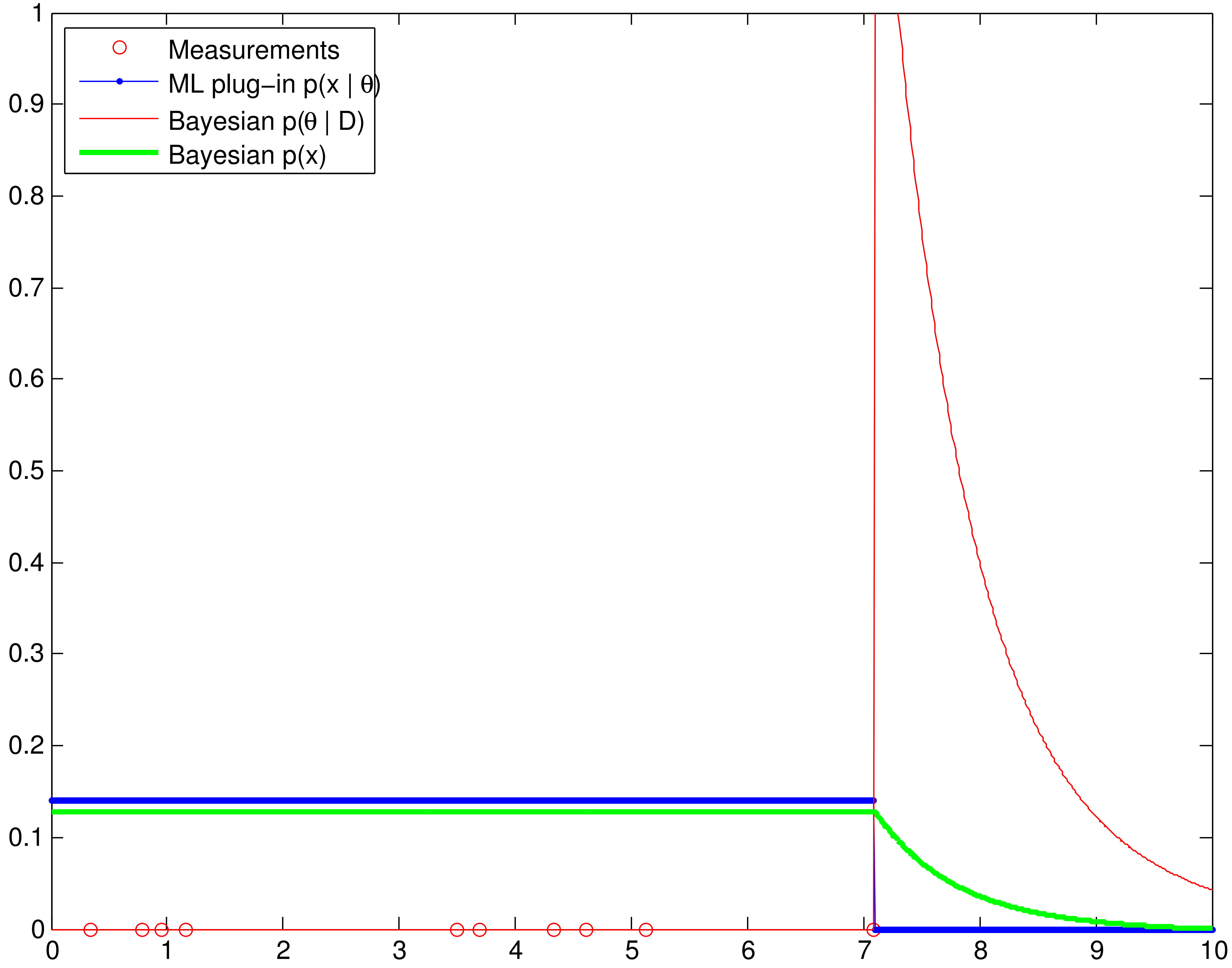
Example: Uniform Distribution

- Let $p(x;\theta)$ be a uniform distribution in $[0,\ \theta]$.
- Want to estimate θ.
- Suppose we know a priori $\theta \in [0,10]$, choose $p(\theta)$ uniform in $[0,\ 10]$.

Given a sample $\mathcal{D} = \{x_1, x_2, \ldots x_n\}$, compute Bayesian estimate of $p(\theta \mid \mathcal{D})$:

$$p(\theta \mid \mathcal{D}) \propto \prod_{i=1}^{n} p(x_i \mid \theta) p(\theta) = \prod_{i=1}^{n} \frac{1}{\theta} [\![x_i \leq \theta]\!] p(\theta).$$

Example: Uniform Distribution



Variational Bayesian Learning

- Proposition: compute approximation to $p(\theta \mid D)$ by a simpler distribution $q(\theta)$.
- Let for example $\theta \in \mathbb{R}^d$ and

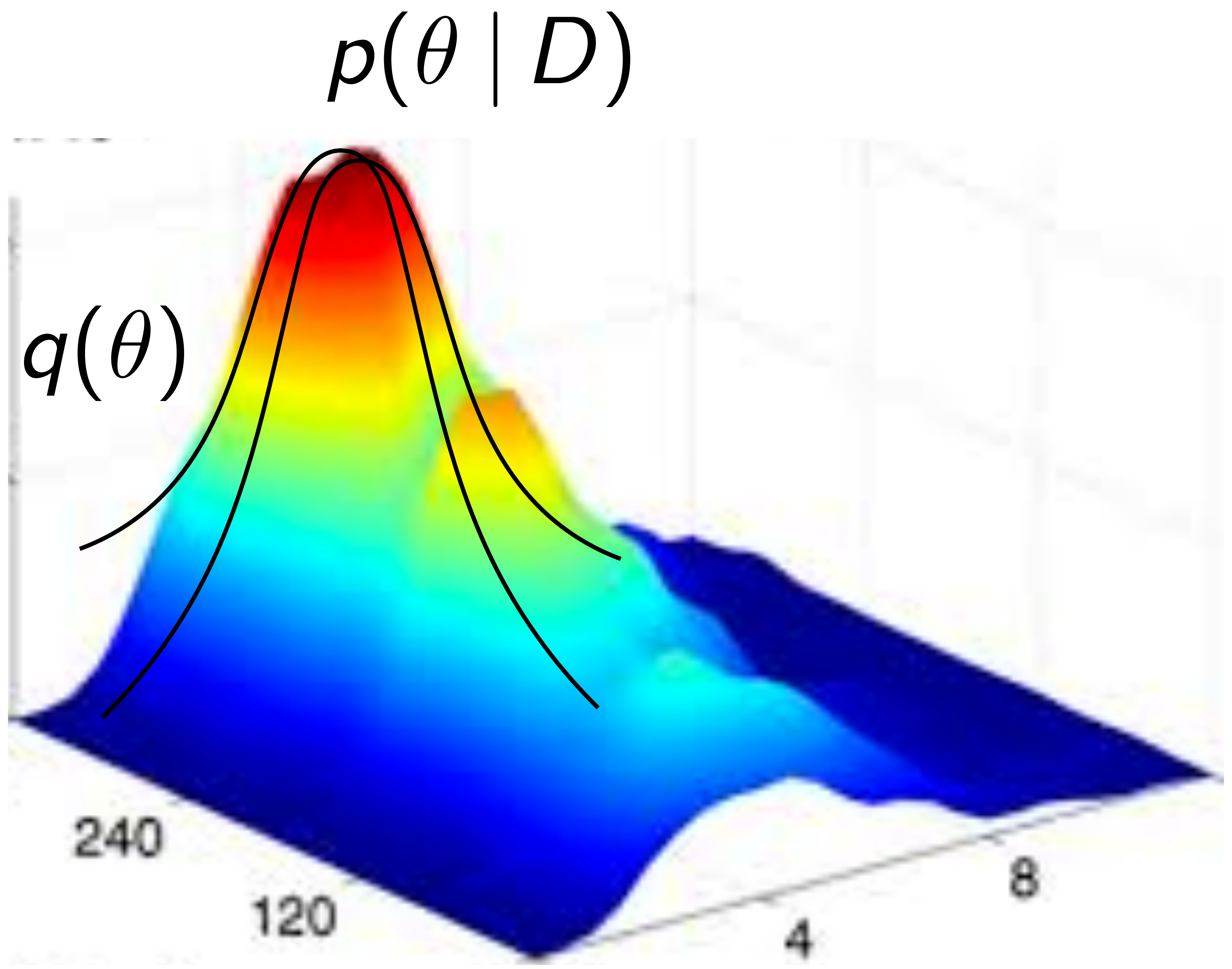
$$q(\theta) = \prod_{i=1}^{d} p_{\mathcal{N}}(\theta_i; \hat{\theta}, \hat{\sigma}^2)$$

- For each coordinate of θ we would like to estimate mean and variance.
- Recall the mean field approach:

$$\min_q KL(q(\theta) \| p(\theta \mid D))$$

- Only this time θ is continuous.

Sensible if expect the posterior to be concentrated around some point



Variational Bayesian Learning

Having q, The Bayesian posterior is approximated using distribution q in place of $p(\theta \mid D)$:

$$p(y \mid x, D) \approx \int p(y \mid x, \theta) q(\theta) d\theta.$$

Solving the variational problem. Expand KL:

$$KL(q(\theta)\|p(\theta \mid D)) = \mathbb{E}_{\theta\sim q(\theta)} \log \frac{q(\theta)}{p(\theta \mid D)} = \mathbb{E}_{\theta\sim q(\theta)} \log \frac{q(\theta)}{\prod_t p(y^t \mid x^t, \theta) p(\theta)/p(D)}$$

$$= \underbrace{\mathbb{E}_{\theta\sim q(\theta)}\Big[-\sum_t \log p(y^t \mid x^t, \theta)\Big]} + \underbrace{KL(q(\theta)\|p(\theta))} + \log p(D).$$

log likelihood, expected over parameters, data evidence — data-independent regularization

Special case I:

When we choose q to be the delta-function at $\hat{\theta}$ (fix a tiny $\hat{\sigma}$) and the prior $p(\theta)$ as $\mathcal{N}(0, \sigma_0^2 I)$, the variational optimization becomes, up to constants,

$$\min_{\hat{\theta}}[-\sum_t \log p(y^t \mid x^t, \hat{\theta})] + \frac{\|\hat{\theta}\|^2}{2\sigma_0^2},$$

I.e., we recover the maximum likelihood, with a weight regularization.

Variational Bayesian Learning with SGD

$$KL(q(\theta)\|p(\theta \mid D)) = \mathbb{E}_{\theta\sim q(\theta)}\Big[-\sum_t \log p(y^t \mid x^t, \theta)\Big] + KL(q(\theta)\|p(\theta)) + \log p(D).$$

$$\operatorname*{argmin}_q KL(q(\theta)\|p(\theta \mid D)) = \operatorname*{argmin}_q -|D|\mathbb{E}_{\substack{\theta\sim q \\ (x,y)\sim D}}\Big[-\log p(y \mid x, \theta)\Big] + KL(q(\theta)\|p(\theta))$$

Special case II: $q(\theta) = q(\theta \mid \phi)$ is Gaussian with parameters ϕ

- $KL(q(\theta)\|p(\theta))$ is closed form for several types of priors $p(\theta)$
- Gradient in q of the data evidence expresses as:

$$\frac{\partial}{\partial\phi}\mathbb{E}_{\substack{\theta\sim q \\ (x,y)\sim D}}\Big[-\log p(y \mid x, \theta)\Big] = \mathbb{E}_{\substack{\theta\sim q \\ (x,y)\sim D}}\Big[-\frac{\partial}{\partial\phi}\log p(y \mid x, \theta)\Big]$$

A stochastic estimate of the gradient an be made from few samples of the data and parameters — means we can apply SGD

Variational Bayesian Learning with SGD

Stochastic gradient in q:

- pick a random training sample (x^t, y^t) (or a batch)
- *sample* parameters θ from current posterior: $\theta \sim q(\theta)$
- Evaluate usual log likelihood $\log p(y^t \mid x^t, \theta)$
- add *regularizer*
- back propagate and perform a gradient descent step *in parameters of q*

Looks similar to training with dropout, doesn't it?

References:

- Graves A.: Practical Variational Inference for Neural Networks, 2009
- Schulman, J., Heess, N., Weber, T., Abbeel, P.: Gradient estimation using stochastic computation graphs, 2015

Books

- Cowell et al. "Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks", 2007
- S. L. Lauritzen. Graphical models., 1996
- Sebastian Nowozin and Christoph H. Lampert. Structured Learning and Prediction in Computer Vision