Undirected graphical models

and other unnormalised (aka energy-based) models

Pierre-Alexandre Mattei

Inria

UNIVERSITÉ CÔTE D'AZUR

3iA Côte d'Azur
Institut interdisciplinaire
d'intelligence artificielle

Menu for this lecture

1. Motivation for undirected models

2. Undirected graphical models, general theory

3. Examples of undirected models

4. Learning in undirected graphical models and energy-based models

$\mathbb{P}(\mathcal{G}|X)$

1

Motivation(s) for undirected models

What are directed graphical models?

- Let $G = (V, E)$ be a DAG whose nodes are denoted $V = \{1, ..., d\}$
- Let $X = (X_1, ..., X_d)$ be a random variable with density $p(x) = p(x_1, ..., x_d)$.
- Every node of the graph corresponds to a random variable

$\mathbb{P}(G \mid X)$

What are directed graphical models?

- Let $G = (V, E)$ be a DAG whose nodes are denoted $V = \{1, ..., d\}$
- Let $X = (X_1, ..., X_d)$ be a random variable with density $p(x) = p(x_1, ..., x_d)$.
- Every node of the graph corresponds to a random variable

Definition (Directed graphical model). *We say that p* ***factorises in*** *G (denoted $p \in \mathcal{L}(G)$) when, for all x,*

$$p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i}),$$

where, for all node i, pa_i denotes the set of parents of node i.

What are directed graphical models?

- Let $G = (V, E)$ be a DAG whose nodes are denoted $V = \{1, ..., d\}$
- Let $X = (X_1, ..., X_d)$ be a random variable with density $p(x) = p(x_1, ..., x_d)$.
- Every node of the graph corresponds to a random variable

Definition (Directed graphical model). *We say that p* ***factorises in*** *G (denoted $p \in \mathcal{L}(G)$) when, for all x,*

$$p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i}),$$

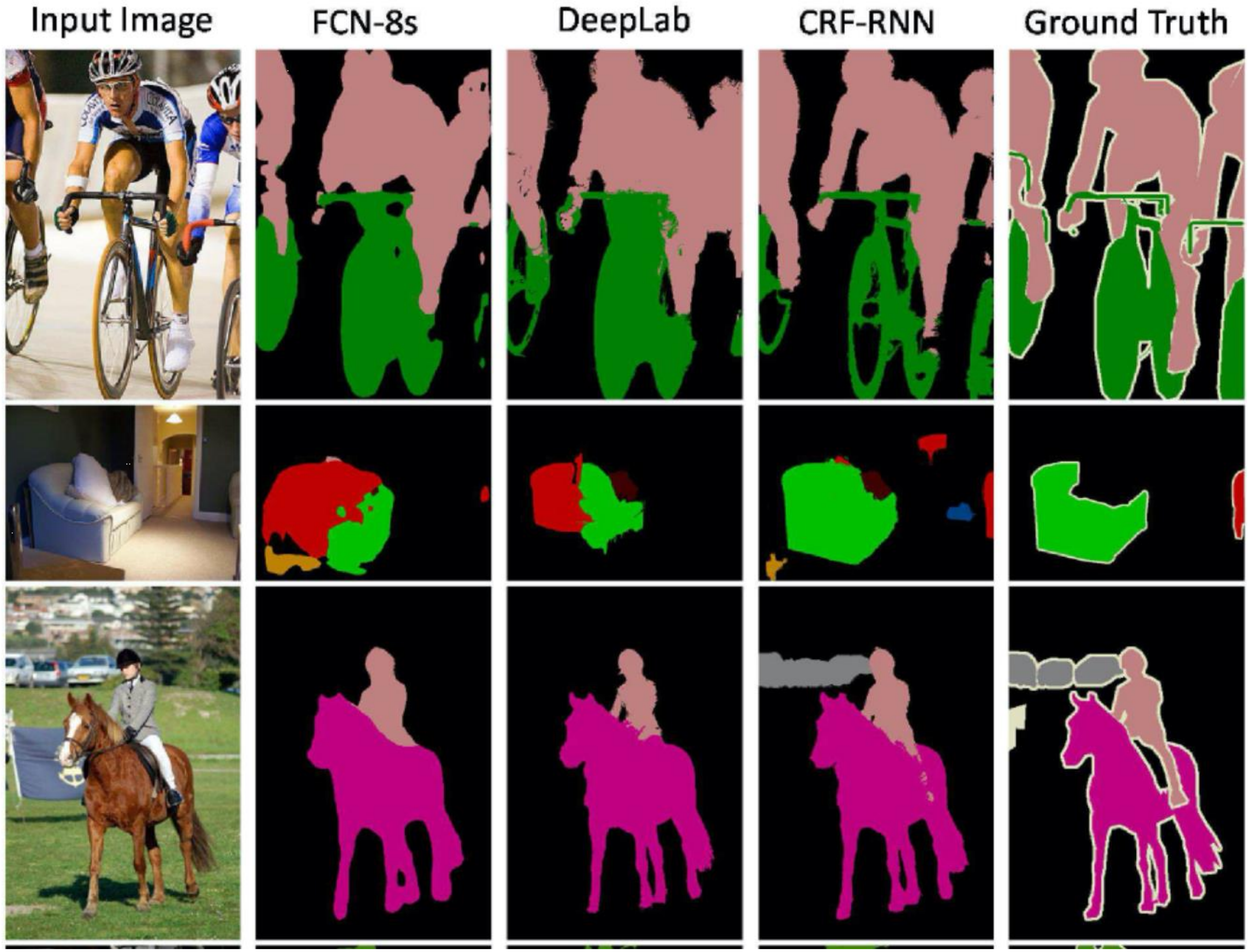
the likelihood (or bounds at least) can often be computed exactly

where, for all node i, pa_i denotes the set of parents of node i.

Motivating example: image segmentation with a neural net (eg Unet)

We have couples $(x_i, y_i)_{i \leq n}$

$\mathbb{P}(\theta | X)$



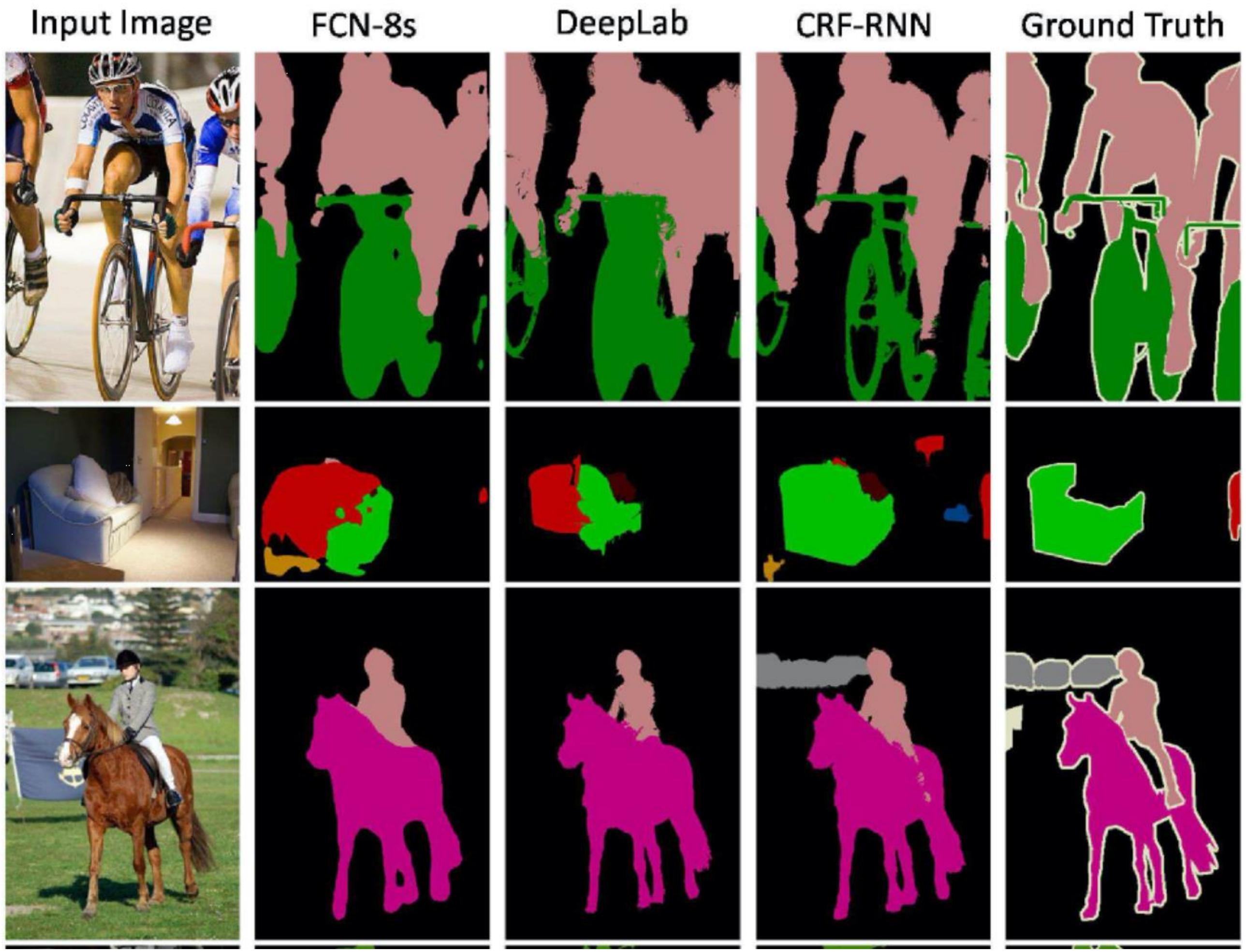
Conditional Random Fields as Recurrent Neural Networks

Motivating example: image segmentation with a neural net (eg Unet)

We have couples $(x_i, y_i)_{i \leq n}$

- $x_i \in \mathbb{R}^{n_{\text{pixels}}}$
- $y_i \in \{1, \ldots, C\}^{n_{\text{pixels}}}$

$\mathbb{P}(\theta|X)$



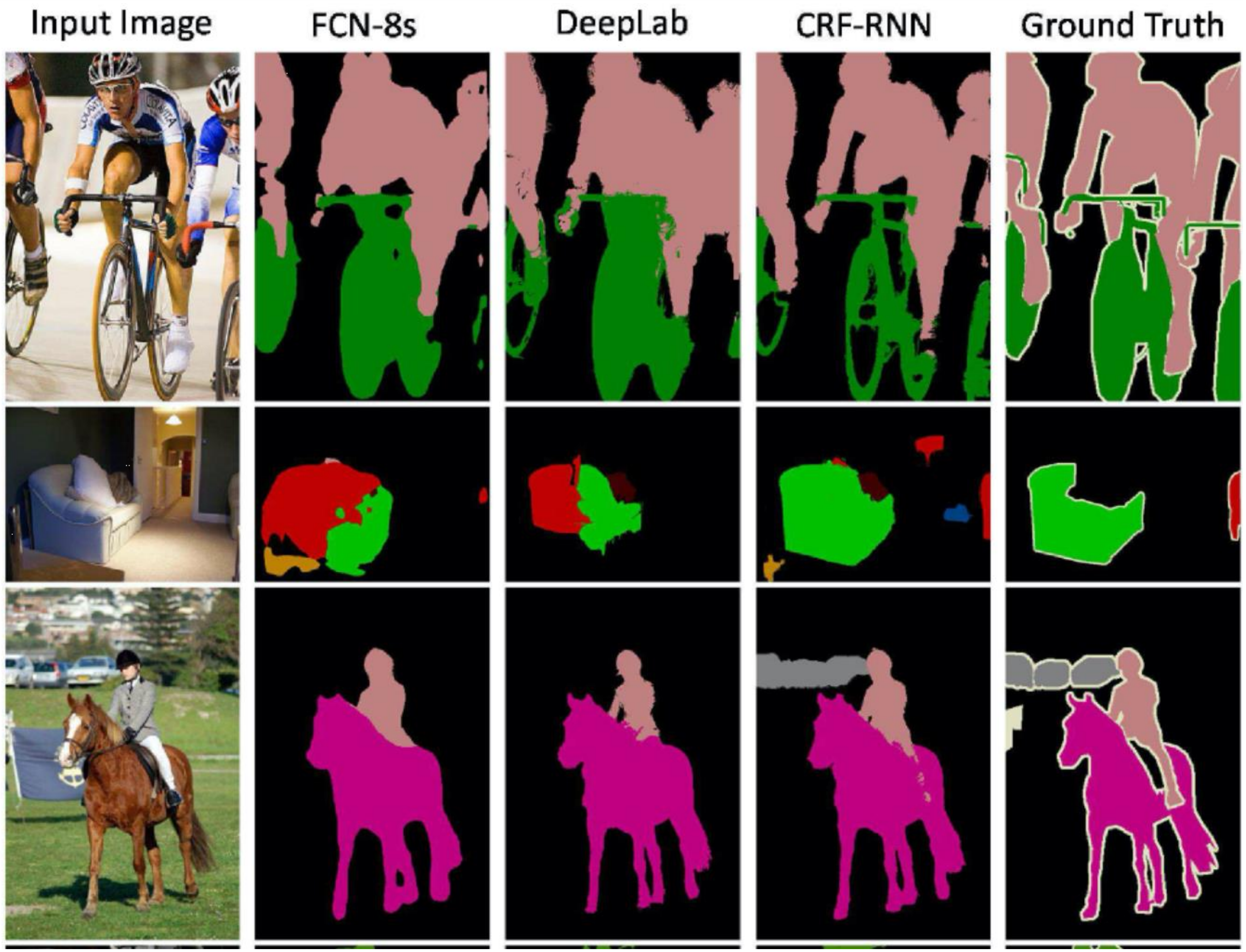
Conditional Random Fields as Recurrent Neural Networks

Motivating example: image segmentation with a neural net (eg Unet)

We have couples $(x_i, y_i)_{i \leq n}$

- $x_i \in \mathbb{R}^{n_{\text{pixels}}}$
- $y_i \in \{1, \ldots, C\}^{n_{\text{pixels}}}$

number of possible segmentation classes
(e.g. bycicle, cyclist, horse, background...)



Conditional Random Fields as Recurrent Neural Networks

Simple examples: image segmentation with a neural net (eg Unet)

$$p_\beta(x_1, \ldots, x_n, y_1 \ldots y_n) = \prod_{i=1}^{n} p(x_i) p_\beta(y_i | x_i)$$

We want to use a Unet that thakes as input an image and gives as output one probability vector per pixel.

$\mathbb{P}(\theta | X)$

Simple examples: image segmentation with a neural net (eg Unet)

$$p_\beta(x_1, \ldots, x_n, y_1 \ldots y_n) = \prod_{i=1}^{n} p(x_i) p_\beta(y_i|x_i)$$

We want to use a Unet that thakes as input an image and gives as output one probability vector per pixel.

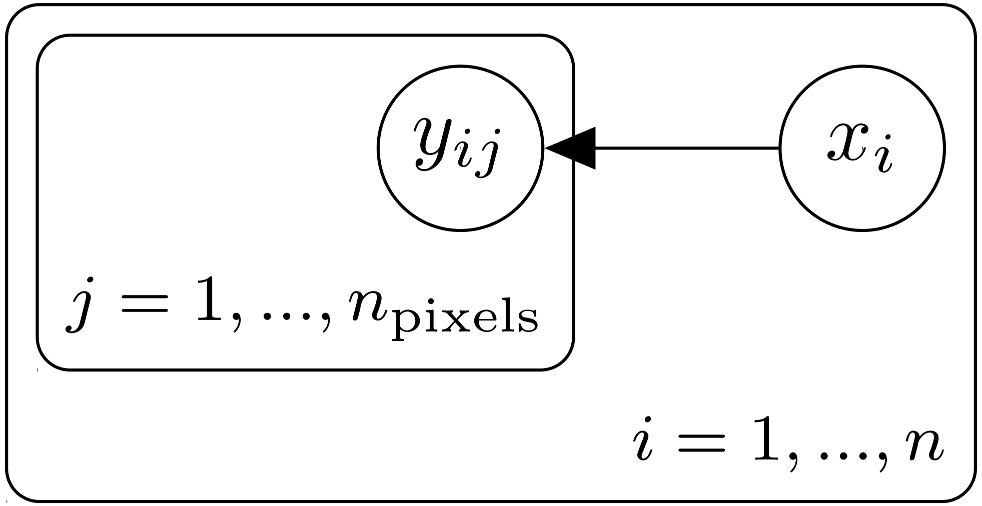
$$p_\beta(y_i|x_i) = \prod_{j=1}^{n_{\text{pixels}}} \mathcal{M}(y_{ij}|[\text{Unet}_\beta(x_i)]_j)$$

Simple examples: image segmentation with a neural net (eg Unet)

$$p_\beta(x_1, \ldots, x_n, y_1 \ldots y_n) = \prod_{i=1}^{n} p(x_i) p_\beta(y_i | x_i)$$

We want to use a Unet that thakes as input an image and gives as output one probability vector per pixel.

$$p_\beta(y_i | x_i) = \prod_{j=1}^{n_{\text{pixels}}} \mathcal{M}(y_{ij} | [\text{Unet}_\beta(x_i)]_j)$$

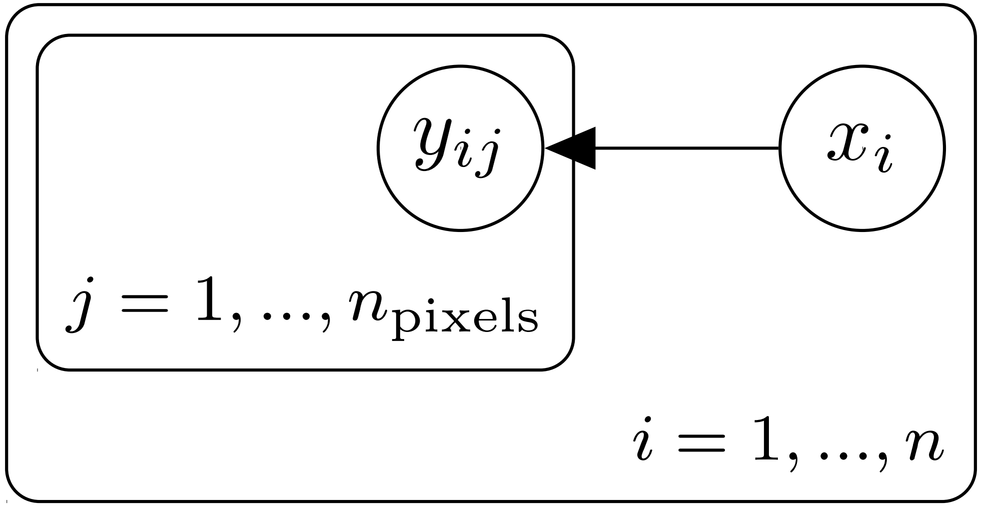


Simple examples: image segmentation with a neural net (eg Unet)

$$p_\beta(x_1, \ldots, x_n, y_1 \ldots y_n) = \prod_{i=1}^{n} p(x_i) p_\beta(y_i|x_i)$$

We want to use a Unet that thakes as input an image and gives as output one probability vector per pixel.

$$p_\beta(y_i|x_i) = \prod_{j=1}^{n_{\text{pixels}}} \mathcal{M}(y_{ij}|[\text{Unet}_\beta(x_i)]_j)$$



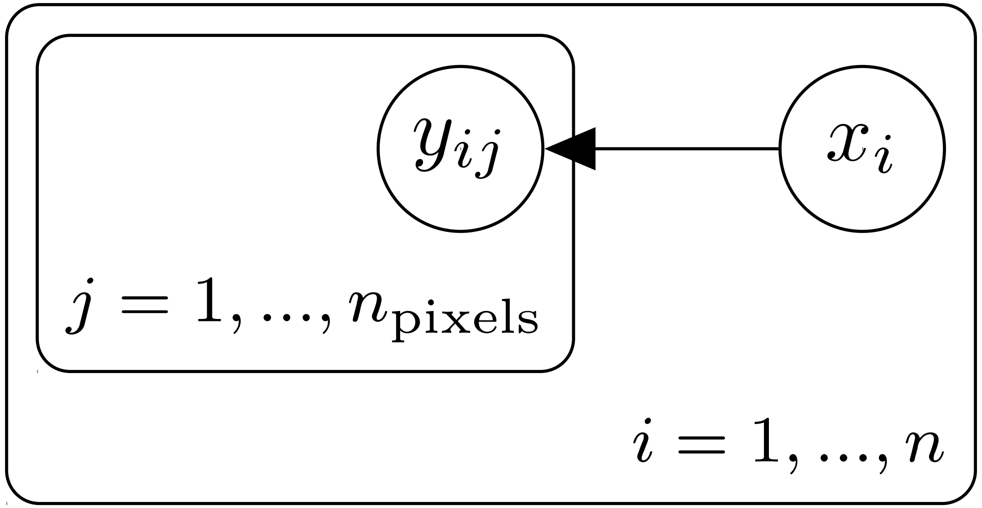
Q. Do you see issues with this model?

Simple examples: image segmentation with a neural net (eg Unet)

$$p_\beta(x_1, \ldots, x_n, y_1 \ldots y_n) = \prod_{i=1}^{n} p(x_i) p_\beta(y_i | x_i)$$

We want to use a Unet that thakes as input an image and gives as output one probability vector per pixel.

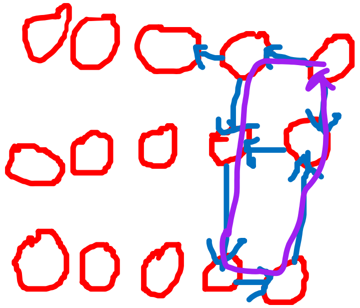
$$p_\beta(y_i | x_i) = \prod_{j=1}^{n_{\text{pixels}}} \mathcal{M}(y_{ij} | [\text{Unet}_\beta(x_i)]_j)$$

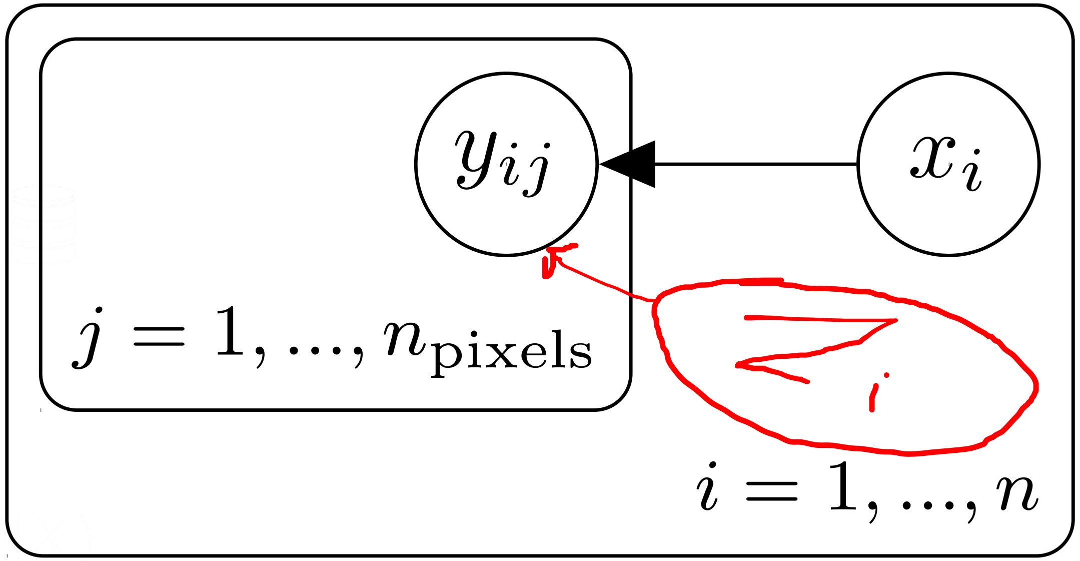


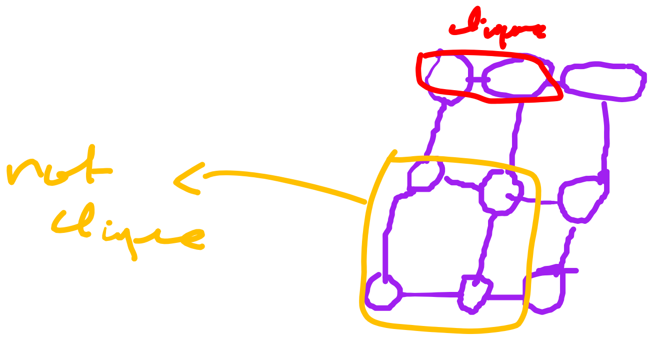
Q. Do you see issues with this model?

The pixels maybe shouldn’t be independent given the image…

How do we fix this model ?







Some issues with DAGs

- As we saw in the initial lectures, a graph is a DAG if and only if it can be topologically ordered
- Hence, DAGs are mostly well-suited for cases where there is a **natural ordering over modalities**

$\mathbb{P}(G \mid X)$

Some issues with DAGs

- As we saw in the initial lectures, a graph is a DAG if and only if it can be topologically ordered
- Hence, DAGs are mostly well-suited for cases where there is a **natural ordering over modalities**
- We saw previously that, **when there are two natural orderings (eg in VAEs or diffusion models), we could just use two DAGs**, but often that’s not the case

Some issues with DAGs

- As we saw in the initial lectures, a graph is a DAG if and only if it can be topologically ordered

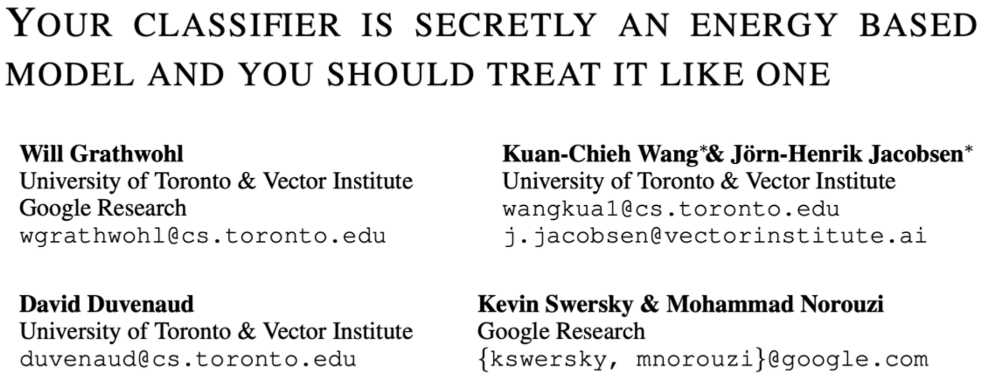
- Hence, DAGs are mostly well-suited for cases where there is a **natural ordering over modalities**

- We saw previously that, **when there are two natural orderings (eg in VAEs or diffusion models), we could just use two DAGs**, but often that’s not the case

- Even then, two DAGs means two models, which is not always satisfying.

Another motivating example

- This is a nice model if we're just interested in pure supervised learning
- But we can't sample images from this
- We can't compute $p(x)$, handle missing data
- It is tempting to reverse the arrow, that would allow us to do all that…
- Of course, we could do two DAGs, but can we do a single model?



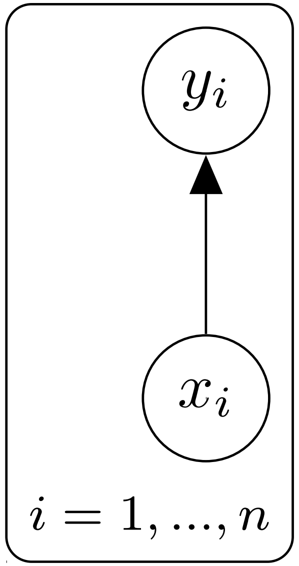
YOUR CLASSIFIER IS SECRETLY AN ENERGY BASED MODEL AND YOU SHOULD TREAT IT LIKE ONE

Will Grathwohl
University of Toronto & Vector Institute
Google Research
wgrathwohl@cs.toronto.edu

Kuan-Chieh Wang*& Jörn-Henrik Jacobsen*
University of Toronto & Vector Institute
wangkua1@cs.toronto.edu
j.jacobsen@vectorinstitute.ai

David Duvenaud
University of Toronto & Vector Institute
duvenaud@cs.toronto.edu

Kevin Swersky & Mohammad Norouzi
Google Research
{kswersky, mnorouzi}@google.com

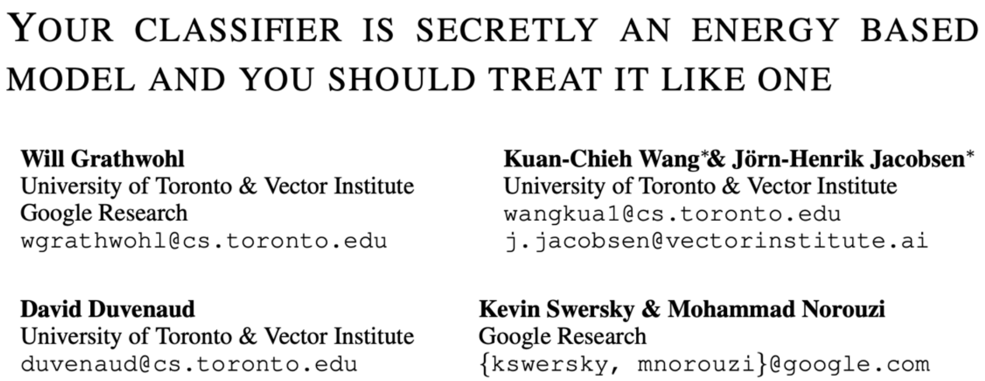


$$p_\beta(y_i|x_i) = \text{Categorical}(y_i|\text{CNN}_\beta(x))$$

Another motivating example

- This is a nice model if we’re just interested in pure supervised learning

$\mathbb{P}(y | X)$



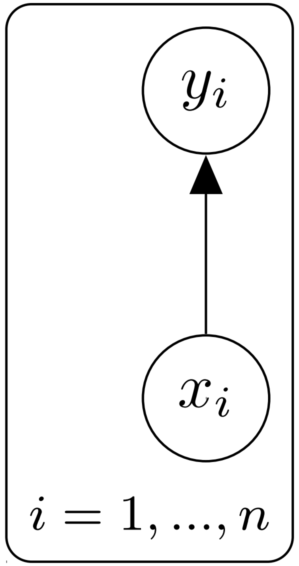
YOUR CLASSIFIER IS SECRETLY AN ENERGY BASED MODEL AND YOU SHOULD TREAT IT LIKE ONE

Will Grathwohl
University of Toronto & Vector Institute
Google Research
wgrathwohl@cs.toronto.edu

Kuan-Chieh Wang*& Jörn-Henrik Jacobsen*
University of Toronto & Vector Institute
wangkua1@cs.toronto.edu
j.jacobsen@vectorinstitute.ai

David Duvenaud
University of Toronto & Vector Institute
duvenaud@cs.toronto.edu

Kevin Swersky & Mohammad Norouzi
Google Research
{kswersky, mnorouzi}@google.com

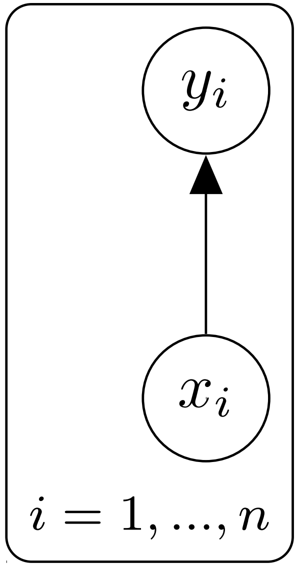


$$p_\beta(y_i|x_i) = \text{Categorical}(y_i|\text{CNN}_\beta(x))$$

Another motivating example

- This is a nice model if we're just interested in pure supervised learning
- But we can't sample images from this
- We can't compute $p(x)$, handle missing data

$\mathbb{P}(z|X)$

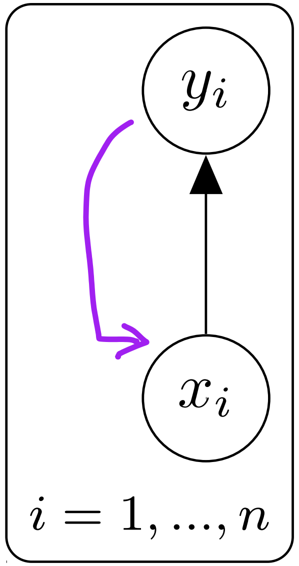


$$p_\beta(y_i|x_i) = \text{Categorical}(y_i|\text{CNN}_\beta(x))$$

Another motivating example

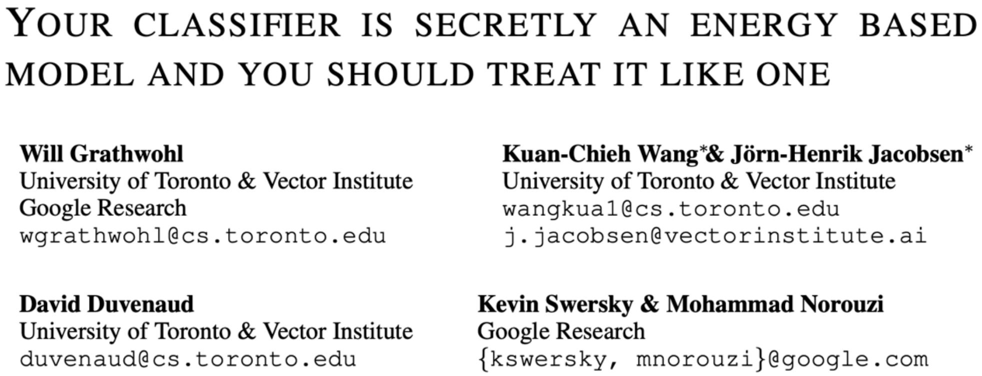
- This is a nice model if we're just interested in pure supervised learning
- But we can't sample images from this
- We can't compute $p(x)$, handle missing data
- It is tempting to reverse the arrow, that would

allow us to do all that…



$$p_\beta(y_i|x_i) = \text{Categorical}(y_i|\text{CNN}_\beta(x))$$

Another motivating example



YOUR CLASSIFIER IS SECRETLY AN ENERGY BASED MODEL AND YOU SHOULD TREAT IT LIKE ONE

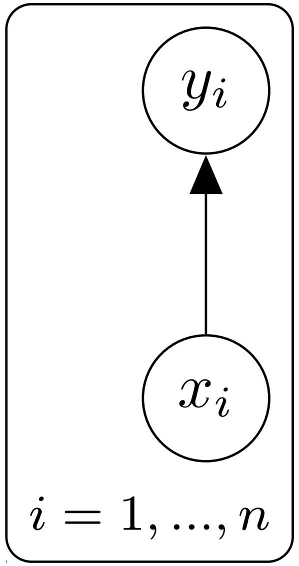
Will Grathwohl
University of Toronto & Vector Institute
Google Research
wgrathwohl@cs.toronto.edu

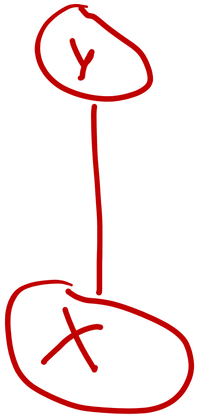
Kuan-Chieh Wang* & Jörn-Henrik Jacobsen*
University of Toronto & Vector Institute
wangkua1@cs.toronto.edu
j.jacobsen@vectorinstitute.ai

David Duvenaud
University of Toronto & Vector Institute
duvenaud@cs.toronto.edu

Kevin Swersky & Mohammad Norouzi
Google Research
{kswersky, mnorouzi}@google.com

- This is a nice model if we're just interested in pure supervised learning
- But we can't sample images from this
- We can't compute $p(x)$, handle missing data
- It is tempting to reverse the arrow, that would allow us to do all that…





$$p_\beta(y_i|x_i) = \text{Categorical}(y_i|\text{CNN}_\beta(x))$$

- Of course, we could do two DAGs, **but can we do a single model?**

2

Undirected graphical models **(aka Markov random fields)**

Image segmentation as a motivation for undirected models

- We would like to model some correlation between neighbourhoods of pixels after the image exits the convnet.

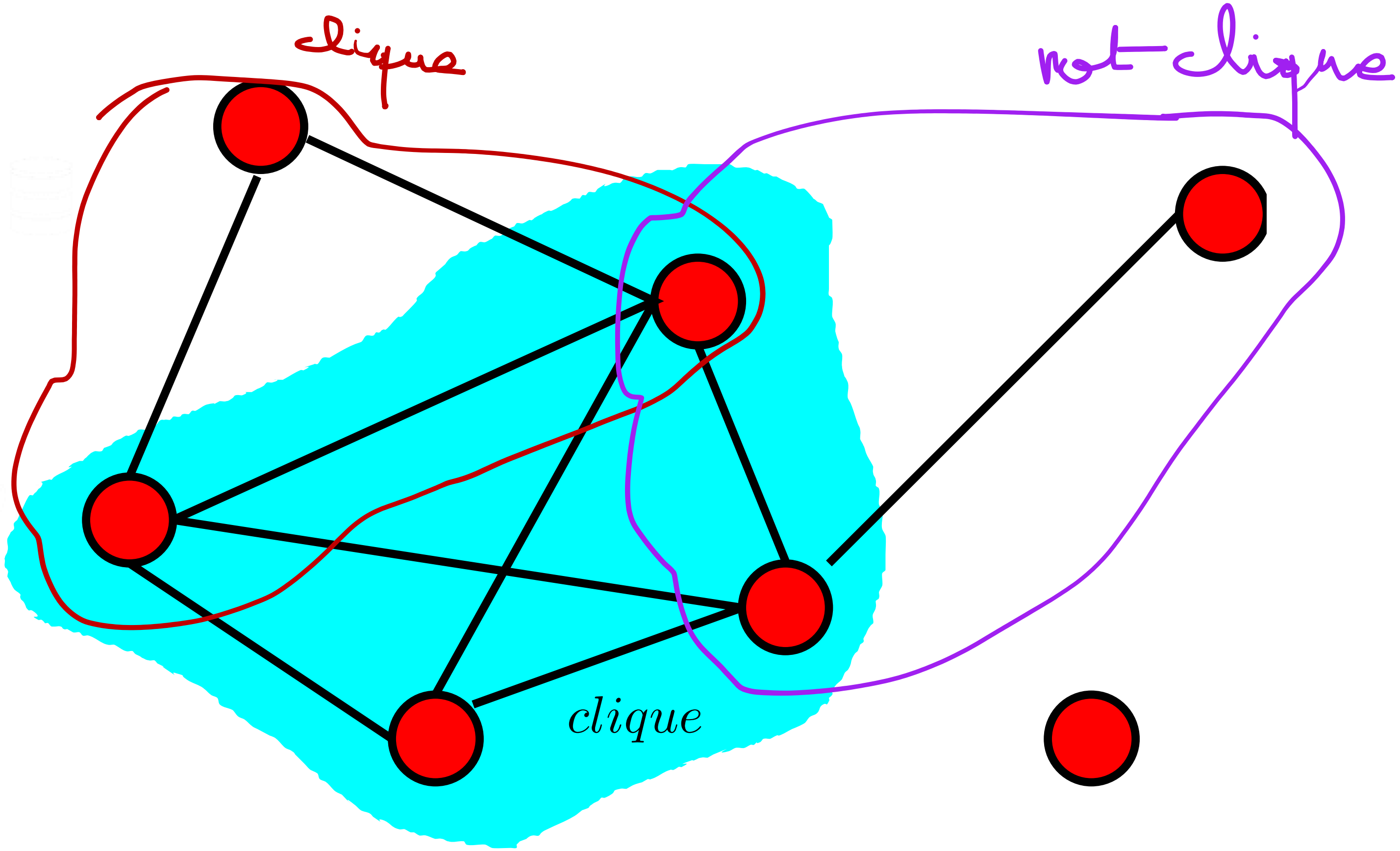
- Unfortunately, **it is a bit arbitrary to look at pixels as nodes of a DAG**: indeed, there is no obvious notion of ordering on the pixels of an image!

- It would make sense to **create a graph without order**, for which nearby pixels are connected. This is exactly the sort of thing we can do with undirected models!

$\mathbb{P}(G|X)$

What’s a clique?

Definition. *A totally connected subset of vertices is called a* ***clique****.*



What’s an undirected graphical model?

Definition. *Let $G = (V, E)$ be an undirected graph. We denote by $\mathcal{C}$ the set of all cliques of G . We say that **a probability distribution** p **factorizes in G** and write $p \in \mathcal{L}(G)$ if $p(x)$ is of the form:*

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \quad \textit{with} \quad \psi_C \geq 0,\ C \in \mathcal{C} \quad \textit{and} \quad Z = \int \prod_{C \in \mathcal{C}} \psi_C(x_C) dx.$$

$< +\infty$

*The functions ψ_C are called **factors** or **potentials**.*

What's an undirected graphical model?

Definition. *Let $G = (V, E)$ be an undirected graph. We denote by $\mathcal{C}$ the set of all cliques of G . We say that* ***a probability distribution*** p ***factorizes in G*** *and write $p \in \mathcal{L}(G)$ if $p(x)$ is of the form:*

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \quad \textit{with} \quad \psi_C \geq 0,\ C \in \mathcal{C} \quad \textit{and} \quad Z = \int \prod_{C \in \mathcal{C}} \psi_C(x_C) dx.$$

The functions ψ_C are called ***factors*** *or* ***potentials****.*

- Contrarily to directed models, **the factors are not necessarily probability densities!**
- Why can we be sure that, in spite of this, $p(x)$ will still be a proper density?

What’s an undirected graphical model?

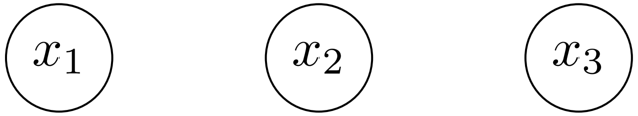
Definition. *Let $G = (V, E)$ be an undirected graph. We denote by $\mathcal{C}$ the set of all cliques of G . We say that* ***a probability distribution*** *p* ***factorizes in G*** *and write $p \in \mathcal{L}(G)$ if $p(x)$ is of the form:*

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \quad \textit{with} \quad \psi_C \geq 0,\ C \in \mathcal{C} \quad \textit{and} \quad Z = \int \prod_{C \in \mathcal{C}} \psi_C(x_C) dx.$$

The functions ψ_C are called ***factors*** *or* ***potentials****.*

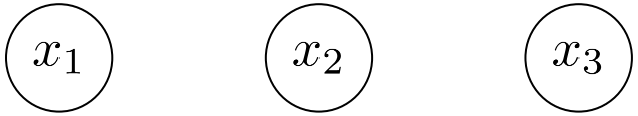
- Contrarily to directed models, **the factors are not necessarily probability densities!**
- Why can we be sure that, in spite of this, $p(x)$ will still be a proper density?
  - ➢ Because of the division by *Z*, that ensures that $p(x)$ sums to one.
- Because of this normalisation, multiplying the potentials by constants will not change anything.

The simplest example: the trivial graph



- What are all the possible cliques of this graph?

The simplest example: the trivial graph

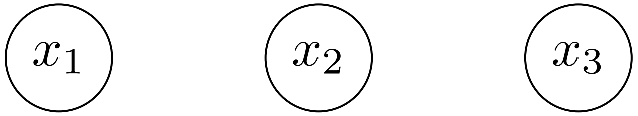


- What are all the possible cliques of this graph?
  - Just the three individual nodes!

- Therefore, the distributions that factorise in this trivial graph will be of the form

$$p(x) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)$$

The simplest example: the trivial graph



- What are all the possible cliques of this graph?
  - Just the three individual nodes!

- Therefore, the distributions that factorise in this trivial graph will be of the form

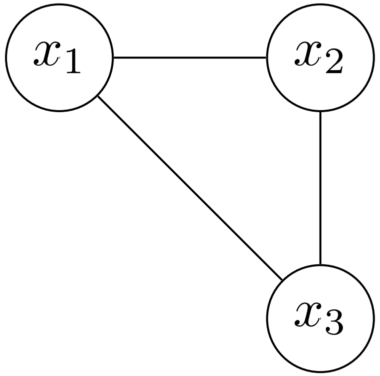
$$p(x) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)$$

- A direct consequence is that $\forall i \in \{1, 2, 3\},\ p(x_i) = \frac{1}{\int \psi_i(x)dx}\psi_i(x_i)$, and therefore

$$p(x) = p(x_1)p(x_2)p(x_3)$$

- Like in the directed case, **the trivial graph corresponds to independence.**

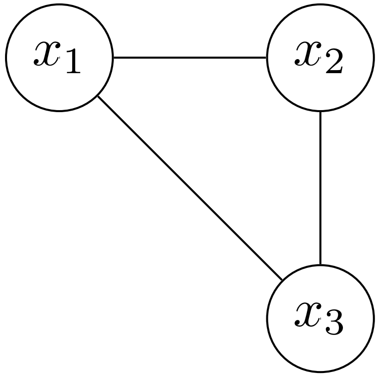
The other end of the spectrum: the complete graph



- Again, what are the possible cliques?
  - All the **individual nodes**, but also **the pairs**, and also **the full graph**!

$\mathbb{P}(\theta \mid X)$

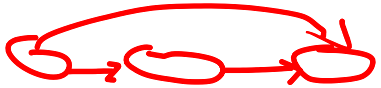
The other end of the spectrum: the complete graph



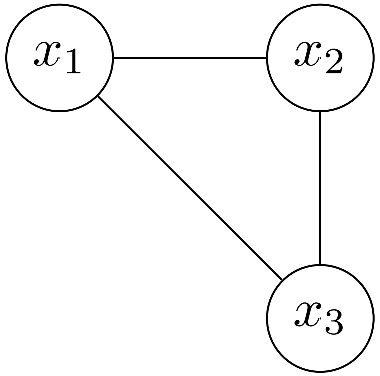
- Again, what are the possible cliques?
  - ➢ All the **individual nodes**, but also **the pairs**, and also **the full graph**!

$$p(x) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{13}(x_1,x_3)\psi_{123}(x_1,x_2,x_3)$$

$\mathbb{P}(\theta | X)$



The other end of the spectrum: the complete graph



- Again, what are the possible cliques?
  - All the **individual nodes**, but also **the pairs**, and also **the full graph**!

$$p(x) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{13}(x_1,x_3)\psi_{123}(x_1,x_2,x_3)$$

- By posing $\tilde{\psi}_{123}(x_1,x_2,x_3) = \psi_1(x_1)\psi_2(x_2)\psi_3(x_3)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{13}(x_1,x_3)\psi_{123}(x_1,x_2,x_3)$

this can be rewritten simply as $p(x) = \dfrac{1}{Z}\tilde{\psi}_{123}(x_1,x_2,x_3)$, so only looking at the largest clique

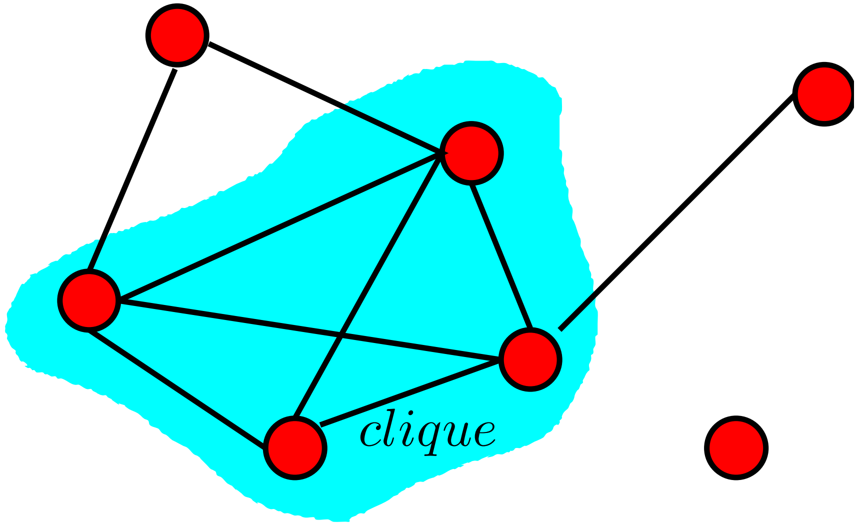
(called a **maximal clique**) will suffice.

- Like in the directed case, any distribution *p* will factorise in this complete graph by choosing

$$\tilde{\psi}_{123}(x_1,x_2,x_3) = p(x_1,x_2,x_3)$$

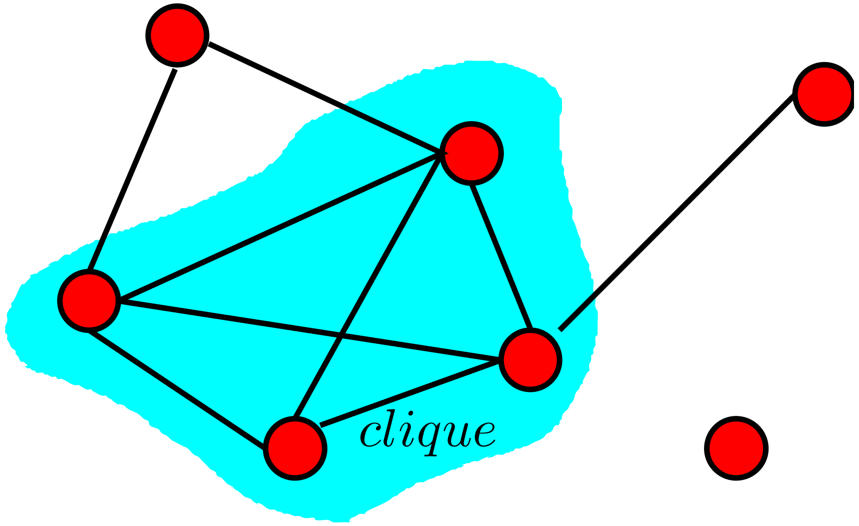
Should we limit ourservelves to maximal cliques

- **A maximal clique is a clique that cannot be included in a stricly larger clique.** Is this clique maximal?



Should we limit ourservelves to maximal cliques?

- **A maximal clique is a clique that cannot be included in a stricly larger clique.** Is this clique maximal?



- **Yes!** But all the cliques strictly included inside (e.g. the singletons or pairs) are not!

Should we limit ourservelves to maximal cliques?

- **A maximal clique is a clique that cannot be included in a stricly larger clique.**
- In the factorisation formula $p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$, we considered $\mathcal{C}$to be the set of all

possible cliques. But like we did for the complete graph, **we could consider only one factor for each maximal clique** (by including inside all the factors of the smaller cliques).

$\mathbb{P}(G|X)$

Should we limit ourservelves to maximal cliques?

- **A maximal clique is a clique that cannot be included in a stricly larger clique.**

- In the factorisation formula $p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$, we considered $\mathcal{C}$to be the set of all

possible cliques. But like we did for the complete graph, **we could consider only one factor for each maximal clique** (by including inside all the factors of the smaller cliques).

- This insight leads some authors **to change a bit the definition of undirected models, replacing $\mathcal{C}$ by the set of all maximal cliques**. For instance, [Bishop] does this.

- The maximal clique definition reduces the number of factors, but make them harder to interpret. For instance, in the previous complete graph example, maybe $\psi_{23}(x_2, x_3)$had a specific interesting form, but considering a single factor obscures this. This is why we did not chose this version of the definition.

Separation and undirected models

- For directed models, we had the nice concept of d-separation that helped us answer questions like « $X_A \perp\!\!\!\perp X_B | X_C ?$ ». Do we have something similar for undirected models?

- Yes! Now it’s just called **separation** (and the definition is much simpler).

Separation and undirected models

- For directed models, we had the nice concept of d-separation that helped us answer questions like « $X_A \perp\!\!\!\perp X_B | X_C?$ ». Do we have something similar for undirected models?

- Yes! Now it's just called **separation** (and the definition is much simpler).

- **Separation recipe:** we consider all chains between any node in A and any node in B. Any of these chains is said to be **blocked** if it passes through . C

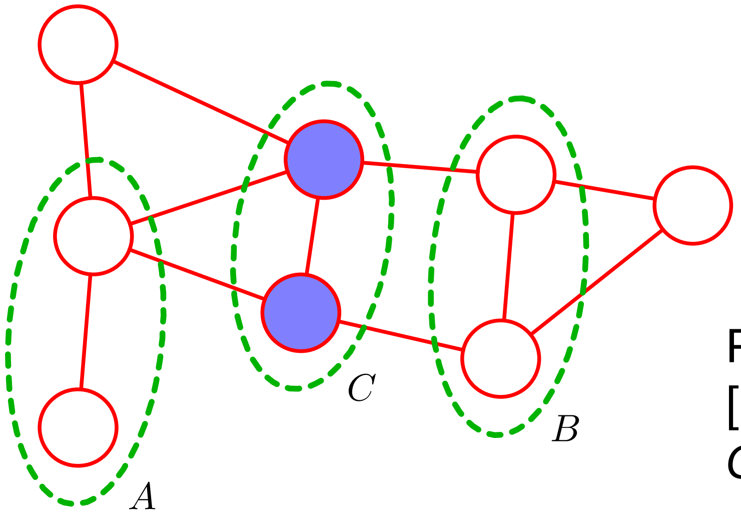
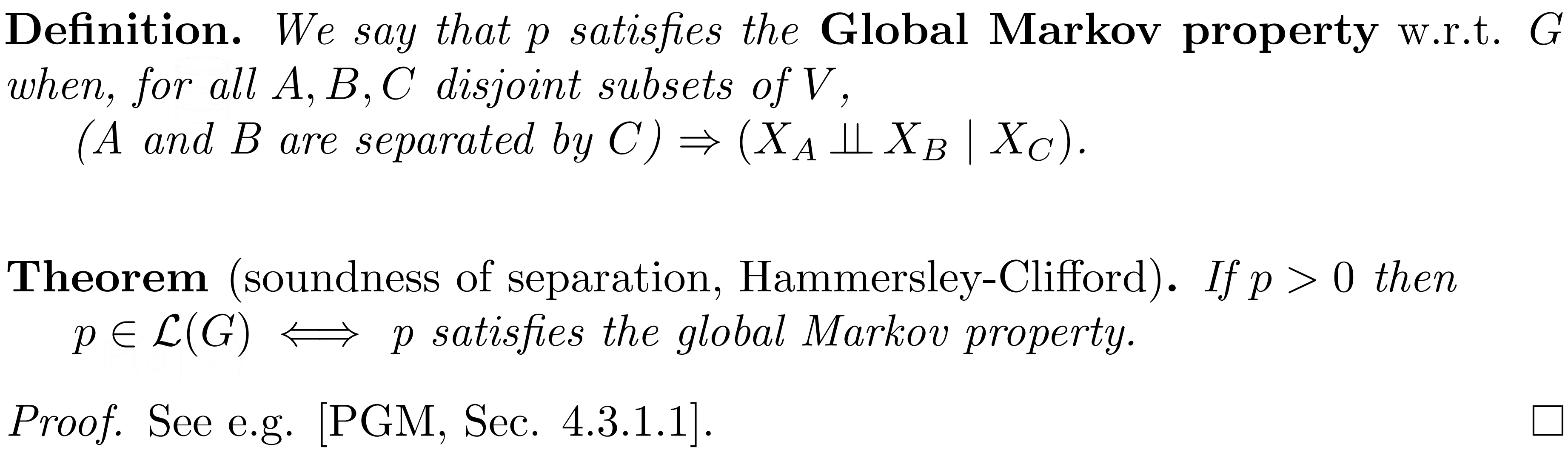


Fig. 8.27 from [Bishop]:
C separates *A* and *B*.

Properties of separation for undirected models

- Like in the directed case, the separation condition is sound and complete. The soundness constitutes a relatively famous theorem first proved by Hammersley and Clifford in 1971.



Definition. *We say that p satisfies the* **Global Markov property** w.r.t. G *when, for all A, B, C disjoint subsets of V,*

$$(A \textit{ and } B \textit{ are separated by } C) \Rightarrow (X_A \perp\!\!\!\perp X_B \mid X_C).$$

Theorem (soundness of separation, Hammersley-Clifford)**.** *If $p > 0$ then*

$$p \in \mathcal{L}(G) \iff p \textit{ satisfies the global Markov property.}$$

Proof. See e.g. [PGM, Sec. 4.3.1.1]. □

Properties of separation for undirected models

- Like in the directed case, the separation condition is sound and complete. The completeness result is similar to the one for d-separation.

Theorem (completeness of separation)**.** *If A and B are not separated by C, then there exist $p \in \mathcal{L}(G)$ such that $X_A \not\perp\!\!\!\perp X_B | X_C$.*

Proof. See e.g. [PGM, Sec. 4.3.1.2]. □

$P(G|X)$

From directed to undirected ?

- The general properties of directed and undirected models are quite similar, are there ways to go from one to the other?

$\mathbb{P}(G \mid X)$

From directed to undirected ?

- The general properties of directed and undirected models are quite similar, are there ways to go from one to the other?

- **Undirected seems more general than directed**, since the potentials are not required to be probability densities. So there should be a way to go from a directed graph to a more general undirected graph…

- We want to turn $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i}),$ into $p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$. How should we do this?

From directed to undirected ?

- The general properties of directed and undirected models are quite similar, are there ways to go from one to the other?

- **Undirected seems more general than directed**, since the potentials are not required to be probability densities. So there should be a way to go from a directed graph to a more general undirected graph…

- We want to turn $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i}),$ into $p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$. How should we do this?

1. We reverse all arrows to get an undirected graph. But that's not enough, since we need to turn each (x_i, x_{pa_i})into a clique.

2. We « marry the parents » by drawing edges between them.

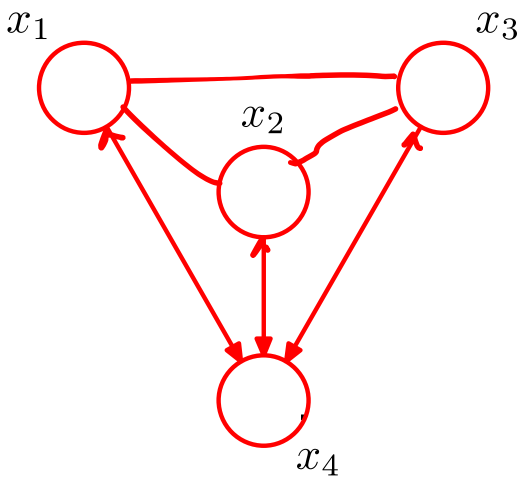
Graph « moralisation »

- This recipe is called « moralising » a graph (because we marry the parents):

  1. We reverse all arrows to get an undirected graph. But that's not enough, since we need to turn each (x_i, x_{pa_i})into a clique.

  2. We « marry the parents » by drawing edges between them.

What it the moralised version of this graph?
Fig. 8.33 in [Bishop].

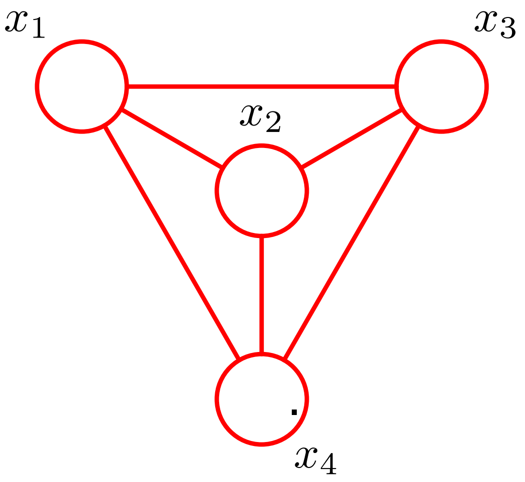


Graph « moralisation »

- This recipe is called « moralising » a graph (because we marry the parents):

1. We reverse all arrows to get an undirected graph. But that's not enough, since we need to turn each (x_i, x_{pa_i})into a clique.
2. We « marry the parents » by drawing edges between them.

Let's just apply the recipe!
Fig. 8.33 in [Bishop].

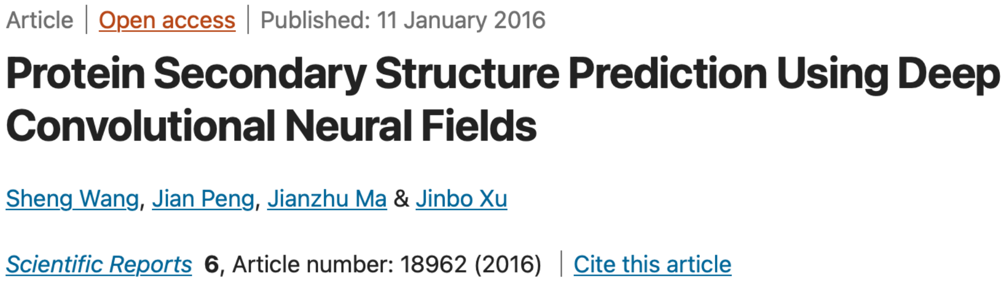


3

Some examples

Conditional random fields for protein structure prediction

- The idea of using the outputs of a neural net as univariate potentials for images or sequence data is generally called a **conditional random field**

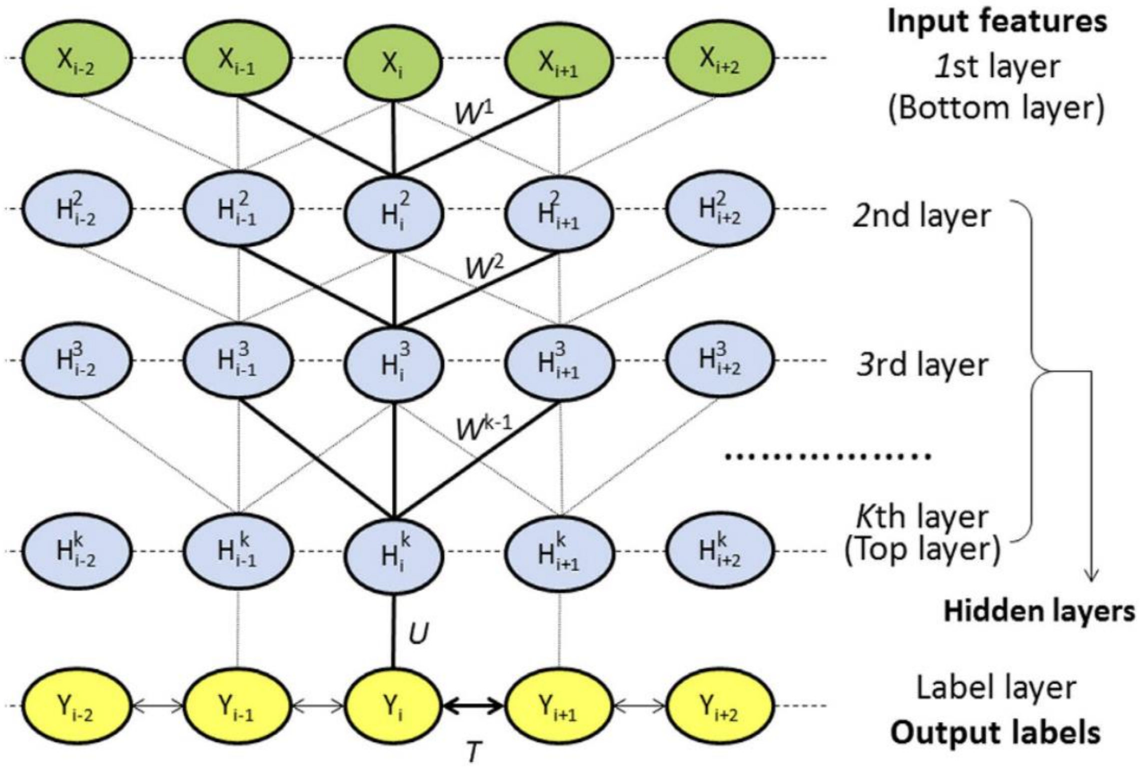


Article | Open access | Published: 11 January 2016

Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields

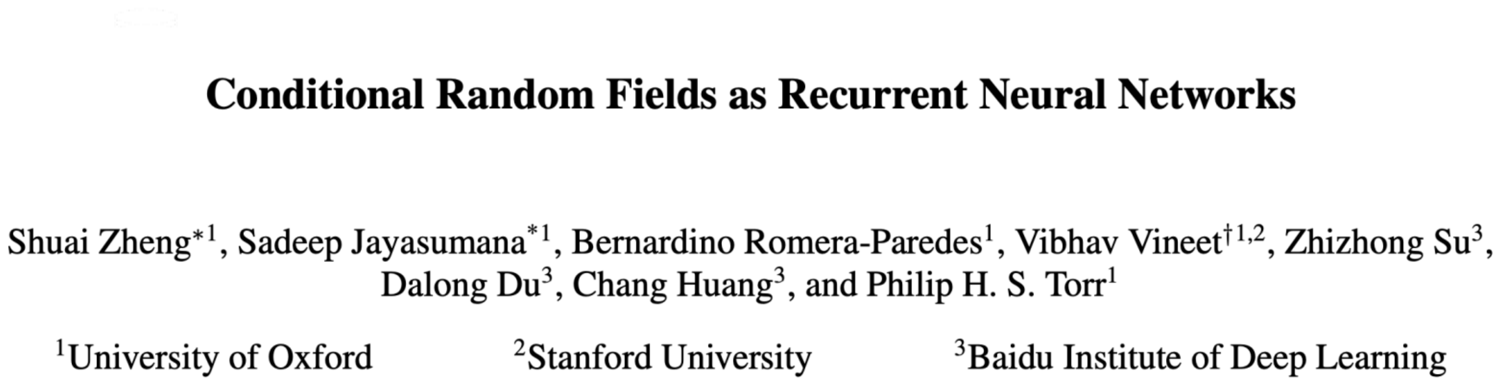
Sheng Wang, Jian Peng, Jianzhu Ma & Jinbo Xu

Scientific Reports **6**, Article number: 18962 (2016) | Cite this article



Conditional random fields for image segmentation

- The idea of using the outputs of a neural net as univariate potentials for images or sequence data is generally called a **conditional random field**



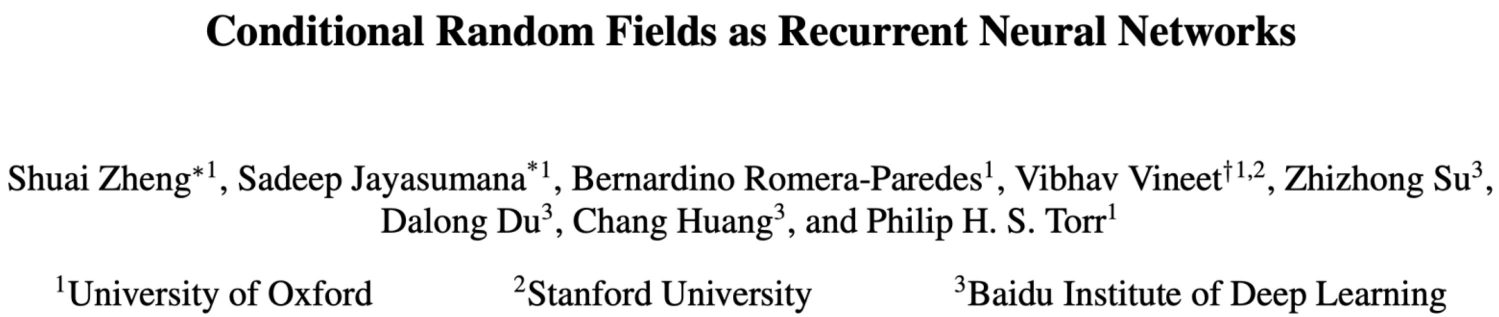
Conditional Random Fields as Recurrent Neural Networks

Shuai Zheng*[1], Sadeep Jayasumana*[1], Bernardino Romera-Paredes[1], Vibhav Vineet†[1,2], Zhizhong Su[3],
Dalong Du[3], Chang Huang[3], and Philip H. S. Torr[1]

[1]University of Oxford [2]Stanford University [3]Baidu Institute of Deep Learning

Conditional random fields for image segmentation

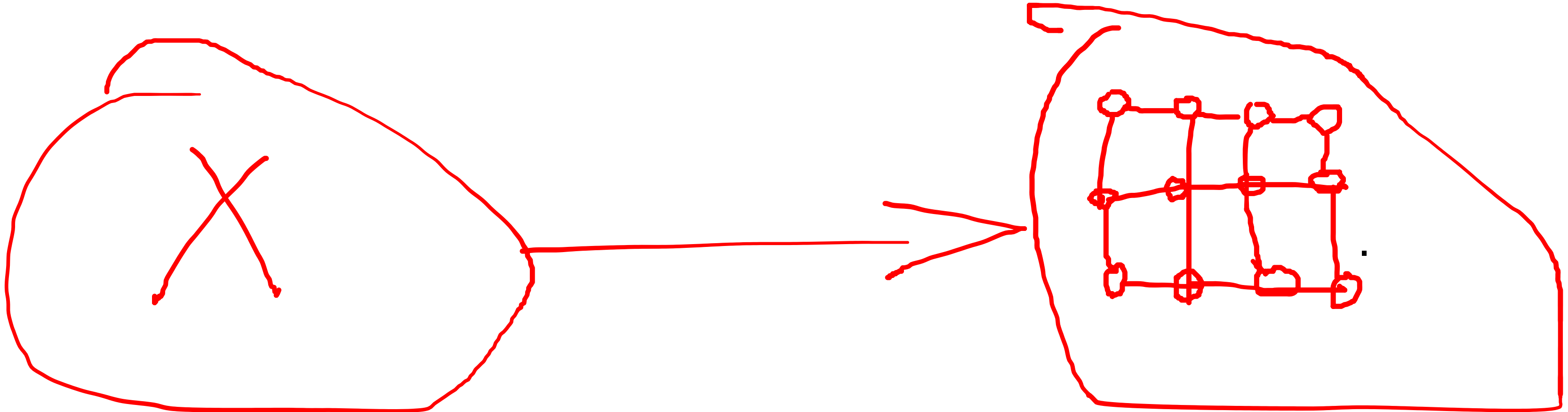
- The idea of using the outputs of a neural net as univariate potentials for images or sequence data is generally called a **conditional random field**



Conditional Random Fields as Recurrent Neural Networks

Shuai Zheng*[1], Sadeep Jayasumana*[1], Bernardino Romera-Paredes[1], Vibhav Vineet†[1,2], Zhizhong Su[3],
Dalong Du[3], Chang Huang[3], and Philip H. S. Torr[1]

[1]University of Oxford [2]Stanford University [3]Baidu Institute of Deep Learning



Getting rid of a lot of constraints

- Exact likelihood models rely on complex constraints on the types of architectures used.
- **Can we get rid of these constraints, and consider super general deep generative models? Something like** $p_\theta(\mathbf{x}) = \mathrm{NN}_\theta(x)$ **with any neural net?**
- Does this make sense?

Getting rid of a lot of constraints

- Exact likelihood models rely on complex constraints on the types of architectures used.

- **Can we get rid of these constraints, and consider super general deep generative models? Something like $p_\theta(\mathbf{x}) = \mathrm{NN}_\theta(x)$ with any neural net?**

- Does this make sense? Not really like this! We have at least **two very basic constraints that we can't get rid of…**
  - The density must **be non-negative**
  - The density must **sum to one**

Getting rid of a lot of constraints

- Exact likelihood models rely on complex constraints on the types of architectures used.
- **Can we get rid of these constraints, and consider super general deep generative models? Something like** $p_\theta(\mathbf{x}) = \mathrm{NN}_\theta(x)$ **with any neural net?**
- Does this make sense? Not really like this! We have at least **two very basic constraints that we can't get rid of…**
    - ➢ The density must **be non-negative**
    - ➢ The density must **sum to one**
- This motivates the very general framework of **energy-based models**

$$p_\theta(\mathbf{x}) \propto \exp(f_\theta(x))$$

Any sort of neural net

Getting rid of a lot of constraints

- EBMs are **undirected graphical models**, with density

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(x))}{Z_\theta}$$

Where $Z_\theta = \int \exp(f_\theta(x))dx$ is the **normalising constant.**

➢Aka **Boltzman distribution** in statistical physics, and $E_\theta(x) = -f_\theta(x)$ is the **energy**

➢When E_θ is linear, this is called an **exponential family** in statistics

4

Training undirected models and other unnormalised models

MLE for EBMs?

- EBMs are **undirected graphical models**, with density

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(x))}{Z_\theta}$$

Where $Z_\theta = \int \exp(f_\theta(x))dx$ is the **normalising constant.**

$\mathbb{P}(\theta | X)$

- Computing the normalising constant seems daunting… Can we really do (approximate) maximum likelihood for these models?

Getting rid of a lot of constraints

- EBMs are **undirected graphical models**, with density

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(x))}{Z_\theta}$$

Where $Z_\theta = \int \exp(f_\theta(x))dx$ is the **normalising constant.**

$\mathbb{P}(\theta | X)$

- Computing the normalising constant seems daunting… Can we really do (approximate) maximum likelihood for these models?

- Remember that we do not need to actually compute the likelihood, but just to optimise it.
- So the only thing we really need are **gradients of the log-likelihood!**

Getting rid of a lot of constraints

- EBMs are **undirected graphical models**, with density

$$p_\theta(\mathbf{x}) = \frac{\exp(f_\theta(x))}{Z_\theta}$$

Where $Z_\theta = \int \exp(f_\theta(x))dx$ is the **normalising constant.**

- Computing the normalising constant seems daunting… Can we really do (approximate) maximum likelihood for these models?

- Remember that we do not need to actually compute the likelihood, but just to optimise it.
- So the only thing we really need are **gradients of the log-likelihood!**

Computing log-likelihood gradients

Problematic term!!

- Let’s try to compute our gradients!

$$\nabla \log p_\theta(\mathbf{x}) = \nabla f_\theta(x) - \nabla \log Z_\theta$$

- We’ll now focus on the gradient of the log-normaliser,

$$\nabla \log Z_\theta = \frac{\nabla Z_\theta}{Z_\theta} = \frac{1}{Z_\theta} \int \nabla \exp(f_\theta(x)) dx$$

which gives

$$\nabla \log Z_\theta = \overbrace{\frac{1}{Z_\theta} \int \exp(f_\theta(x))}^{= p(x)} \nabla f_\theta(x) dx$$

and finally

$$\nabla \log Z_\theta = \int p(x) \nabla f_\theta(x) dx = \mathbb{E}_{x \sim p_\theta}[\nabla f_\theta(x)]$$

Computing log-likelihood gradients

$$\nabla \log Z_\theta = \int p(x) \nabla f_\theta(x) dx = \mathbb{E}_{x \sim p_\theta}[\nabla f_\theta(x)]$$

gives us a way to estimate log-likelihood gradients, provided that we can sample from the model! It’s often called **contrastive divergence**.

- It is possible to sample from the model using MCMC, for instance **Langevin Monte Carlo.**

$\mathbb{P}(\theta | X)$

Aparté: Langevin Monte Carlo

- **Langevin Monte Carlo** provides a general way of sampligng from distributions of the form

$$p(\mathbf{x}) \propto \exp(U(x))$$

- The idea is that this distribution is the stationnary distribution of the Langevin diffusion equation $X' = \nabla U(X) + \sqrt{2}W'$, where *W* is a Brownian motion

- Solving the equation by discretising it gives

$$x^{k+1} = x^k + \tau \overbrace{\nabla U(x^k)}^{\text{Stein score}} + \sqrt{2\tau}\varepsilon_k$$

where τ is essentially a learning rate, and ε_k is standard normal.

- This allows to sample approximatively according to $p(\mathbf{x}) \propto \exp(U(x))$

How to train EBMs? Hyvärinen’s score matching

- Key idea: the gradient of $\log p_\theta(x)$ does not depend on the normalising constant!

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x)$$

- One way of having $p_{\widehat{\theta}} \approx p_{\text{data}}$ is tho have $\nabla_{\mathrm{x}} \log p_{\widehat{\theta}} \approx \nabla_{\mathrm{x}} \log p_{\text{data}}$, so Hyvärinen suggested to minimise

$$\mathbb{E}\left[||\nabla_x \log p_{\mathrm{data}}(x) - \nabla_x \log p_\theta(x)||^2\right]$$

Fisher score $= \nabla_\theta \log p_\theta(x)$

How to train EBMs? Hyvärinen’s score matching

- Key idea: the gradient of $\log p_\theta(x)$ does not depend on the normalising constant!

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x)$$

- One way of having $p_{\hat{\theta}} \approx p_{\text{data}}$ is tho have $\nabla_{\text{x}} \log p_{\hat{\theta}} \approx \nabla_{\text{x}} \log p_{\text{data}}$, so Hyvärinen suggested to minimise

$$\mathbb{E}\left[|| \nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x) ||^2 \right]$$

Hyvärinen score or Stein score of the model

How to train EBMs? Hyvärinen's score matching

- Key idea: the gradient of $\log p_\theta(x)$ does not depend on the normalising constant!

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x)$$

- One way of having $p_{\widehat{\theta}} \approx p_{\text{data}}$ is tho have $\nabla_{\mathrm{x}} \log p_{\widehat{\theta}} \approx \nabla_{\mathrm{x}} \log p_{\text{data}}$, so Hyvärinen suggested to minimise

$$\mathbb{E}\left[||\nabla_x \log p_{\mathrm{data}}(x) - \nabla_x \log p_\theta(x)||^2 \right]$$

Hyvärinen score or Stein score of the data

How to train EBMs? Hyvärinen’s score matching

- Key idea: the gradient of $\log p_\theta(x)$ does not depend on

$$\nabla_x \log p_\theta(x) = -$$

This looks difficult because we dont know

$$\nabla_x \log p_{\text{data}}(x)$$

- One way of having $p_{\hat\theta} \approx p_{\text{data}}$ is tho have $\nabla_{\mathrm{x}} \log p_{\hat\theta} \approx \nabla_{\mathrm{x}} \log p_{\text{data}}$, Hyvärinen suggested to minimise

$$\mathbb{E}\left[||\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x)||^2 \right]$$

Hyvärinen score or Stein score of the data

How to train EBMs? Hyvärinen's score matc

- Key idea: the gradient of $\log p_\theta(x)$ does not depend on

$$\nabla_x \log p_\theta(x) = -$$

This looks difficult because we dont know

$$\nabla_x \log p_{\text{data}}(x)$$

- One way of having $p_{\hat{\theta}} \approx p_{\text{data}}$ is tho have $\nabla_x \log p_{\hat{\theta}} \approx \nabla_x \log p_{\text{data}}$, yvärinen suggested to minimise

$$\mathbb{E}\left[||\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x)||^2\right]$$

Hyvärinen score or Stein score of the data

How to train EBMs? Hyvärinen’s score matching

Stein

$s_\theta(x) =$ score (u-not)

- Hyvärinen managed to attack $\mathbb{E}\left[||\nabla_x \log p_{\text{data}}(x) - \boxed{\nabla_x \log p_\theta}(x)||^2\right]$ using integration by parts, but this requires the Hessian of $\log p_\theta(x)$, it is pretty much impossible to use for deep models…

- Hyvärinen, Estimation of Non-Normalized Statistical Models by Score Matching, JMLR 2005

- Another solution was found by Pascal Vincent, who suggested to replace $\nabla_\text{x} \log p_{\text{data}}$ by $\nabla_\text{x} \log p_{\text{data},\sigma}$, where $p_{\text{data},\sigma}$ is a kernel density estimate of $\log p_{\text{data}}$

How to train EBMs? Vincent's denoising score matching

- Vincent, A Connection Between Score Matching and Denoising Autoencoders, Neural Computation, 2011

- Idea: replace $\nabla_x \log p_{\text{data}}$ by $\boldsymbol{\nabla_x \log p_{\text{data},\sigma}}$, where $p_{\text{data},\sigma}$ is a kernel density estimate of $\log p_{\text{data}}$

$$p_{\text{data},\sigma}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{N}(x|x_i, \sigma^2 I)$$

$p_{\sigma_1} \cdots\cdots p_{\sigma_T}$