Introduction to graphical models and Bayesian networks

Joanna Matuszak, Joanna Wojciechowicz

Data Mining

2023

Probabilistic graphical models

- A simple way to visualize the structure of a probabilistic model.
- Insight into the model's properties - for example, conditional independencies that we can infer by looking at the graph.
- Complex computations that we typically need to perform for inference or model learning can be expressed as graphical manipulations carrying specific mathematical expressions.

Graph

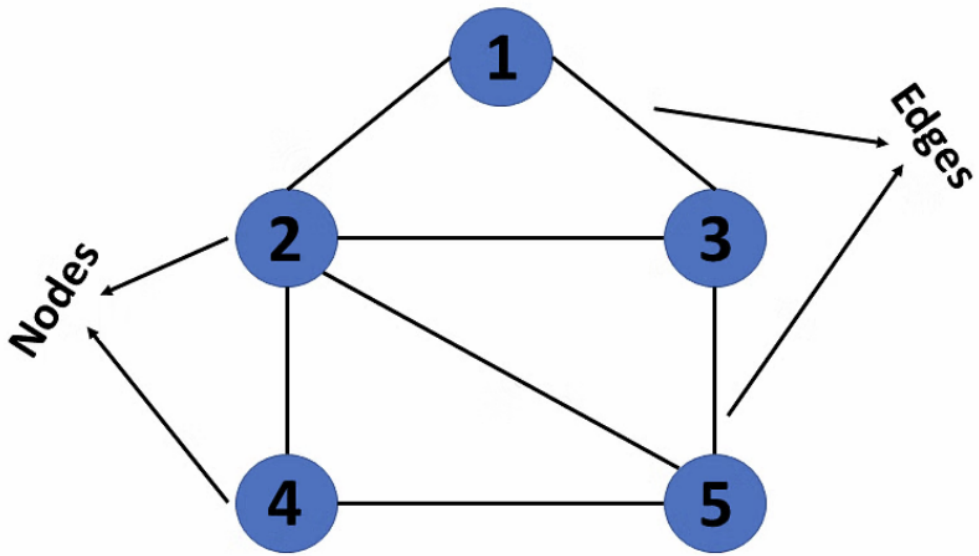

Figure: Source

In probabilistic graphical models

- Node - random variable or group of random variables.
- Edge - probabilistic relationship between connected vertices.
- Entire graph - represents how the joint distribution of all random variables can be decomposed into a product of factors, each depending only on a subset of variables.

In probabilistic graphical models

Two branches of graphical representations of distributions are commonly used

1. Bayesian networks - directed graphical models,
2. Markov random fields - undirected graphical models.

Joint distribution in Bayesian network

First consider an arbitraty joint distribution $p(a, b, c)$ over three variables a, b, c. At this stage, we do not need to specify anything further about these variables. By application of the product rule of probability, we can write the joint distribution in the form

$$p(a, b, c) = p(c|a, b)\, p(a, b) = p(c|a, b)\, p(b|a)\, p(a). \quad (1)$$

Bayesian networks

On the given graph

- a is a parent of node b and node c,
- b is a child of node a and a parent of node c,
- c is a child of node a and node b.

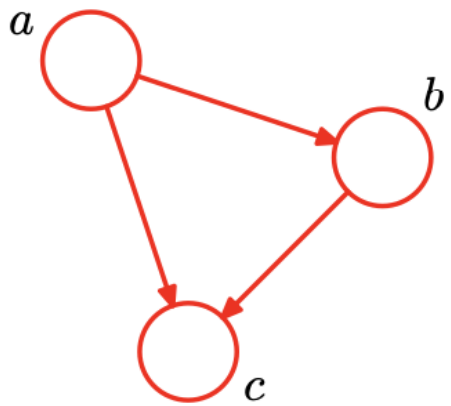

Figure: A directed graphical model representing the joint probability distribution over three variables a, b, and c, corresponding to the decomposition on equation (1). Source: [1].

Joint distribution in Bayesian network

We can now state in general terms the relationship between a given directed graph and the corresponding distribution over the variables. The joint distribution defined by a graph is given by the product, over all of the nodes of the graph, of a conditional distribution for each node conditioned on the variables corresponding to the parents of that node in the graph.
For a graph with K nodes, the joint distribution is given by

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|pa_k), \tag{2}$$

where pa_k - the set of parents of x_k and $\mathbf{x} = x_1, \ldots, x_k$.

Bayesian networks

1. Equation (2) expresses the **factorization properties** of the joint distribution for a directed graphical model.
2. We have considered each node to correspond to a single variable, but we can equally well associate sets of variables and vector-valued variables with the nodes of a graph.
3. The directed graphs that we are considering must be **directed acyclic graphs (DAG)**. In other words there are no closed paths within the graph such that we can move from node to node along links following the direction of the arrows and end up back at the starting node.

Example: Polynomial regression

The polynomial regression model

Polynomial regression model can be expressed as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & \vdots & x_1^m \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{3}$$

This can be written as

$$\vec{y} = \mathbf{X}\vec{w} + \vec{\epsilon}, \tag{4}$$

where $i = 1, 2, \dots, n$.

Example: Polynomial regression

1. Frequentist approach
   - Treats the parameters as **fixed, unknown constants** and estimates them using methods like Ordinary Least Squares.
   - We maximize likelihood, which is the probability of the data given the model parameter - $\max_w P(x, y|w)$.
2. Bayesian approach
   - Treats the parameters as **random variables** and incorporates prior knowledge through prior distributions. It estimates the parameters by combining prior information with observed data, resulting in a posterior distribution.

Bayesian approach

Bayes' theorem

Bayesian approach is based on Bayes' theorem:

$$\overbrace{P(A|B)}^{\text{posterior probability}} = \frac{\overbrace{P(B|A)}^{\text{likelihood}} \overbrace{P(A)}^{\text{prior probability}}}{\underbrace{P(B)}_{\text{prior probability}}} \propto P(B|A)\,P(A)\,. \quad (5)$$

Bayesian approach

- Often $P(B)$ is difficult to calculate as the calculation would involve sums or integrals that would be time-consuming to evaluate, so only the product of the prior and likelihood is considered, since the evidence does not change in the same analysis.
- We maximize posterior probability $P(A|B)$.
- The posterior can be approximated even without computing the exact value of $P(B)$ with methods such as Markov chain Monte Carlo.

Example: Polynomial regression

Bayes' theorem in polynomial regression

Bayes' theorem in our case is given by

$$\overbrace{P(w|x,y)}^{\text{posterior probability}} = \frac{\overbrace{P(x,y|w)}^{\text{likelihood}} \overbrace{P(w)}^{\text{prior probability}}}{\underbrace{P(x,y)}_{\text{data}}} \propto P(x,y|w)\,P(w)\,. \tag{6}$$

Example: Polynomial regression

The random variables in our Bayesian polynomial regression model are the vector of polynomial coefficients $\vec{w}$ and observed data $\vec{t}$. In addition, model contains the input data $\mathbf{x}$ and parameters σ^2, α from distributions of random variables.

The joint distribution of random variables

The joint distribution of random variables is given by

$$p\left(\vec{t}, \vec{w}\right) = p\left(\vec{w}\right) \prod_{n=1}^{N} p\left(t_n | \vec{w}\right). \tag{7}$$

Example: Polynomial regression

Joint distribution (7) can be represented by given graphical model.

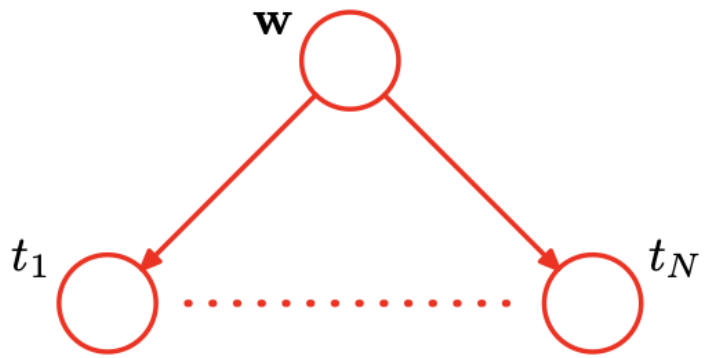

Figure: Directed graphical model representing the joint distribution corresponding to the Bayesian polynomial regression model. Source: [1]

Example: Polynomial regression

When we deal with more complex models, we do not want to write multiple nodes of the form $t_1, \ldots, t_N$. We can introduce more compact graphical notation as below.

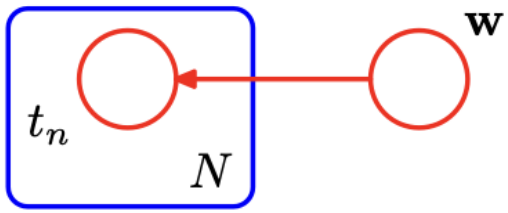

Figure: An alternative, more compact, representation of the graph.
Source: [1]

Example: Polynomial regression

We will make parameters of the model explicit. Now the formula (7) will be different.

The joint distribution of random variables with explicit parameters

The joint distribution of random variables with explicit parameters is given by

$$p\left(\vec{t}, \vec{w} | \mathbf{X}, \alpha, \sigma^2\right) = p\left(\vec{w}, |\alpha\right) \prod_{n=1}^{N} p\left(t_n | \vec{w}, x_n, \sigma^2\right). \qquad (8)$$

Example: Polynomial regression

- We can make $\mathbf{X}$ and α explicit in our graphical representation. We will adapt the convention that random variables will be denoted by open circles, and deterministic parameters will be denoted by smaller solid circles.
- In a graphical model, we will denote observed variables t_n from training set by shading the corresponding nodes.

Example: Polynomial regression

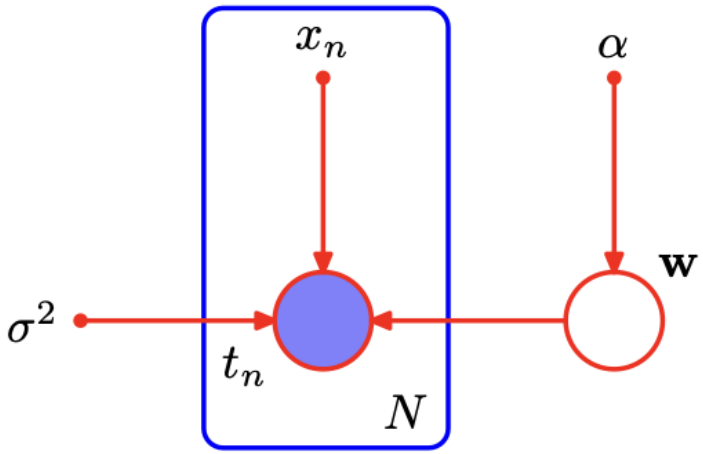

Figure: The same directed graph model, but with the deterministic parameters shown explicitly by the smaller solid nodes. Source: [1]

Example: Polynomial regression

```
import numpy as np
import pymc as pm
import arviz as az
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Generating artificial data
np.random.seed(42)

N = 100
xs = np.linspace(-np.pi, np.pi, num=N)
ys = np.sin(xs) + np.random.normal(0, 0.2, size=N)
```

Example: Polynomial regression

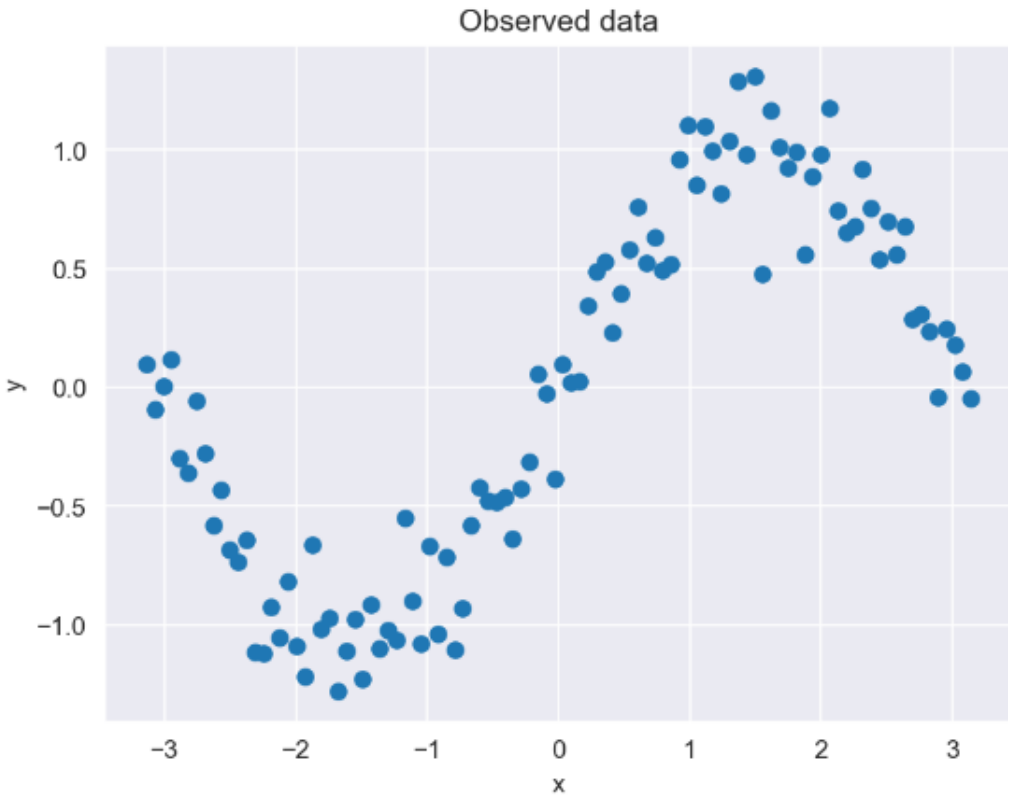

Figure: Generated data

Example: Polynomial regression

```
# Polynomial Regression - Frequentist approach (Least Squares)
poly_model = np.poly1d(np.polyfit(xs, ys, 3))
ys_poly = poly_model(xs)
```

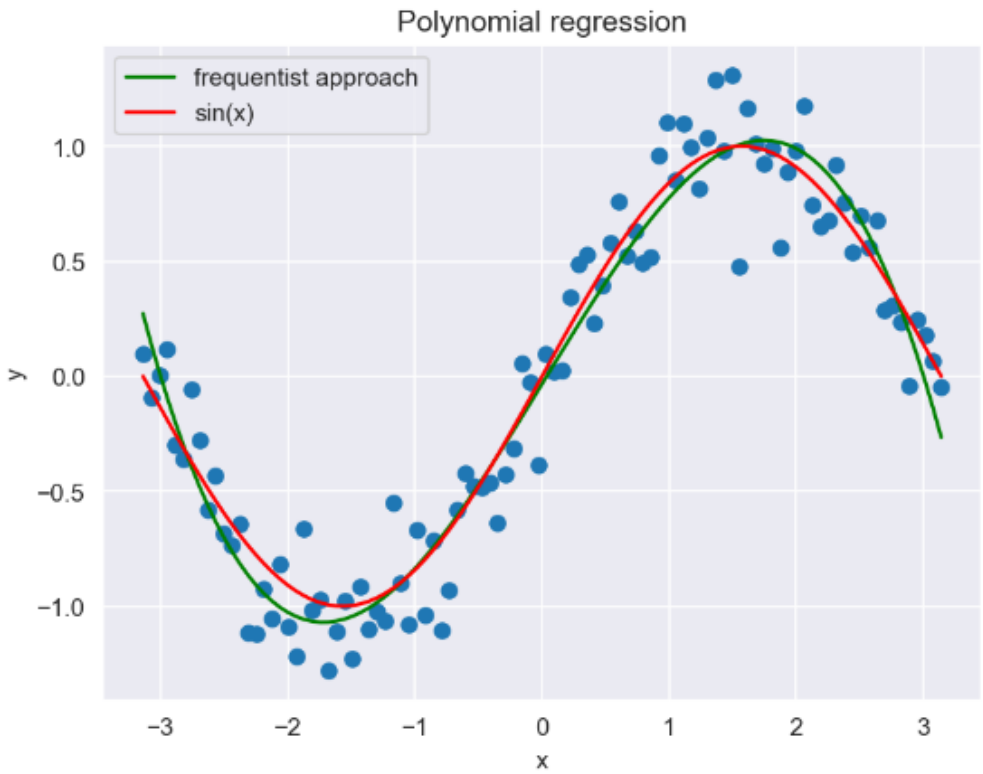

Figure: Generated data with fitted polynomial of 3rd order by frequentist approach

Example: Polynomial regression

```
# Polynomial Regression - Bayesian Network
alpha = 0.1
sigma = 0.1

with pm.Model() as model:
    # Define priors
    w = pm.Normal('w', mu=0, sigma=1/alpha, shape=4)

    # Define likelihood
    mu = w[0] + w[1] * xs + w[2] * xs**2 + w[3] * xs**3
    y = pm.Normal('y', mu=mu, sigma=sigma, observed=ys)

    # Markov chain Monte Carlo (MCMC) to draw posterior samples
    trace = pm.sample(10000, tune=1000, random_seed=42)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [w]
```

100.00% [44000/44000 01:30<00:00
Sampling 4 chains, 0 divergences]

```
Sampling 4 chains for 1_000 tune and 10_000 draw iterations (4_000 + 40_
000 draws total) took 161 seconds.
```

Example: Polynomial regression

```
# Analyzing the model
az.plot_trace(trace)
```

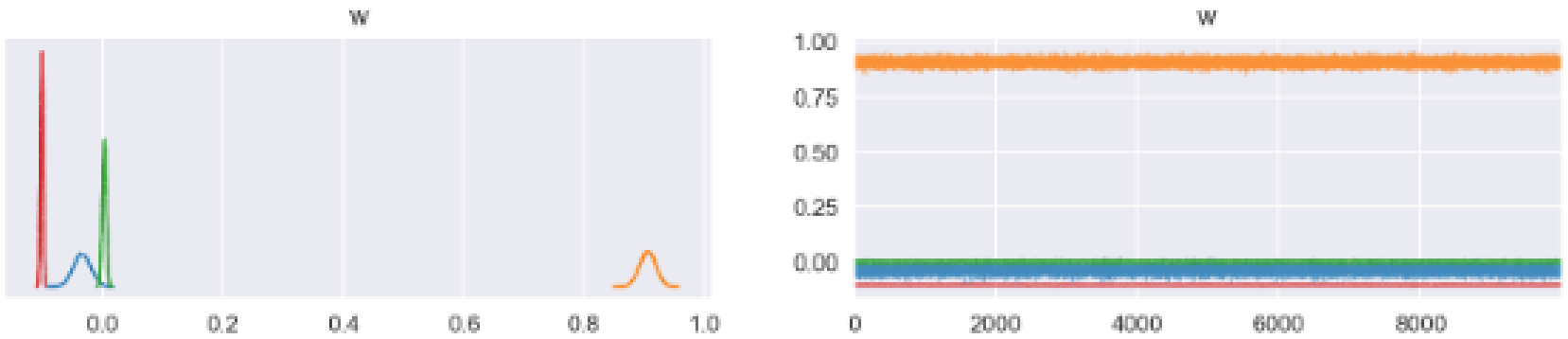

Figure: Marginal posterior distribution for each parameter

Example: Polynomial regression

```
# Analyzing the model
az.summary(trace, kind="stats")
```

	mean	sd	hdi_3%	hdi_97%
w[0]	-0.033	0.015	-0.061	-0.005
w[1]	0.906	0.014	0.880	0.932
w[2]	0.004	0.003	-0.002	0.010
w[3]	-0.101	0.002	-0.104	-0.097

Figure: Summary table of each parameter

Example: Polynomial regression

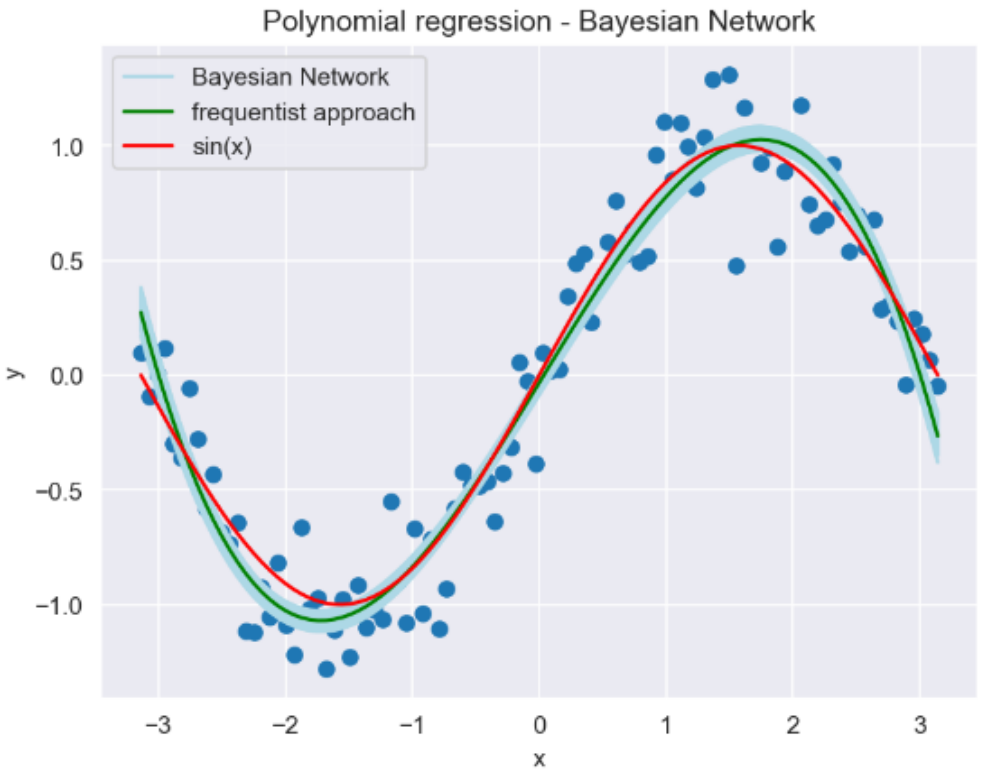

Figure: Generated data with fitted polynomial of 3rd order by frequentist approach and 200 randomly chosen regression lines generated by Bayesian Network

Conclusions

- The key point of the Bayesian approach is that we do not receive a single point estimate for a regression line, i.e. "a line of best fit", as in the frequentist case. Instead we receive a distribution of likely regression lines.
- Bayesian graphs are just a way of visualizing Bayesian statistics and conditional probability, we do not need to build an actual graph.

Discriminative and generative models

- Discriminative algorithms estimate conditional posterior probabilities $P(Y|X)$ whereas generative algorithms model underlying joint probability distributions $P(X, Y)$.
- Discriminative models focus on a direct solution, generative models try to populate the dataset.
- Discriminative models can be used for tasks such as image classification, text categorization. Generative models are useful for image generation, text generation, clustering.
- Generative models provide a complete model of data and can generate new samples by sampling form the learned joint distribution.

Discriminative and generative models

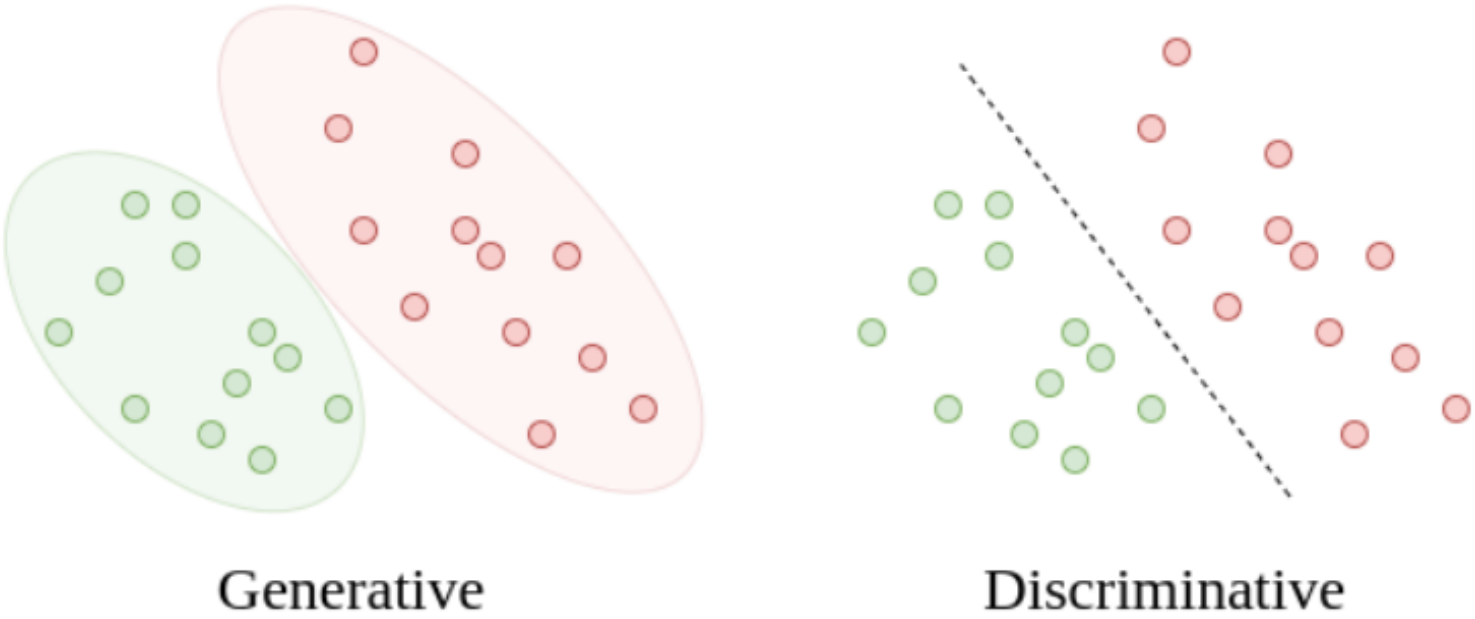

Figure: The difefrence between generative and discriminative algorithms. Source

Discriminative and generative models

Bayesian networks are usually considered as generative models because they allow us to model joint probablility distributions. However, they can be adapted for certain discriminative tasks as well.

Applications of graphical models

- Speech recognition
- Computer vision
- Graphical models for protein structure
- Computer graphics
- Information retrieval
- Manufacturing

Example: Ancestral sampling

Consider a joint distribution $p(x_1, \ldots, x_K)$ over K variables that factorizes to

$$p(x) = \prod_{k=1}^{K} p(x_k|pa_k).$$

In order to draw a sample $\hat{x_1}, \ldots, \hat{x_K}$ from the joint distribution we can follow the algorithm[1]:

- make sure that there are no links from any node to any lower numberd one,
- draw a sample $\hat{x_1}$ from $p(x_1)$,
- go through each node in order and draw $\hat{x_n}$ from $p(x_n|pa_n)$

Example: Ancestral sampling

Once we sample the final variable $\hat{x_K}$ we have the sample from the joint distribution.
We can also obtain a sample from marginal distribution corresponding to subset of variables by discarding the samples from remaining nodes.

Useful libraries

Python:

- PyMC3,
- NiLearn,
- pgmpy,
- PyBN,
- Orange3-Bayesian-Networks.

R:

- bnlearn,
- gRain,
- deal.

Literature

1. Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
2. Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning, Data Mining, Inference, and Prediction, Jan 2017.