

Generative Modelling

Christos Dimitrakakis

November 7, 2025

Outline

Graphical models

- Graphical model

- Exercises

Classification

- Classification: Generative modelling

Density estimation

- Density estimation

Algorithms for latent variable models

- Gradient algorithms

- Expectation maximisation

Exercises

- Density estimation

- Classification

Graphical models

Graphical model

Exercises

Classification

Classification: Generative modelling

Density estimation

Density estimation

Algorithms for latent variable models

Gradient algorithms

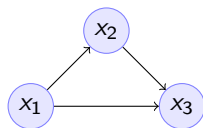
Expectation maximisation

Exercises

Density estimation

Classification

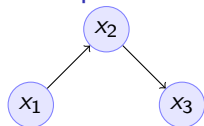
Graphical models



- ▶ Variables x_1, x_2, x_3
- ▶ Arrows denote dependencies between variables.

Conditional independence

Example



Graphical model for the factorisation $\mathbb{P}(x_3 \mid x_2) \mathbb{P}(x_2 \mid x_1) \mathbb{P}(x_1)$.

Definition

- ▶ Consider variables x_1, \dots, x_n .
- ▶ Let B, D be subsets of $[n]$.

We say x_i is **conditionally independent** of x_B given x_D and write

$$x_i \perp\!\!\!\perp x_B \mid x_D$$

if and only if:

$$\mathbb{P}(x_i, x_B \mid x_D) = \mathbb{P}(x_i \mid x_D) \mathbb{P}(x_B \mid x_D).$$

Directed graphical model

A collection of n random variables $x_i : \Omega \rightarrow X_i$, and let $X \triangleq \prod_i X_i$, with underlying probability measure P on Ω . Let $\mathbf{x} = (x_i)_{i=1}^n$ and for any subset $B \subset [n]$ let

$$\mathbf{x}_B \triangleq (x_i)_{i \in B} \quad (1)$$

$$\mathbf{x}_{-j} \triangleq (x_i)_{i \neq j} \quad (2)$$

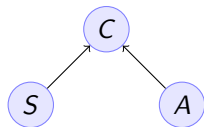
Model specification

$$x_1 \sim f \tag{3}$$

$$x_2 \mid x_1 = a \sim g(a) \tag{4}$$

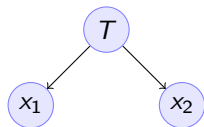
$$x_3 \mid x_2 = b \sim h(b), \tag{5}$$

Smoking and lung cancer



Smoking and lung cancer graphical model, where S : Smoking, C : cancer, A : asbestos exposure.

Time of arrival at work



Time of arrival at work graphical model where T is a traffic jam and x_1 is the time John arrives at the office and x_2 is the time Jane arrives at the office.

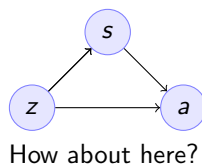
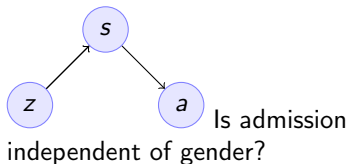
*Conditional independence:

- ▶ Even though x_1, x_2 are **not independent**, they become independent once you know T .

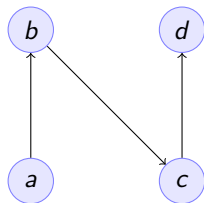
School admission

School	Male	Female
A	62	82
B	63	68
C	37	34
D	33	35
E	28	24
F	6	7

- z : gender
- s : school applied to
- a : admission

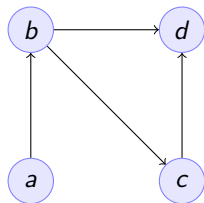


What is the model for this graph?



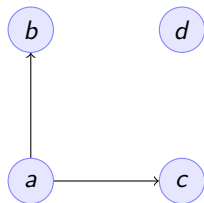
$$P(a, b, c, d) = \dots$$

What is the model for this graph?



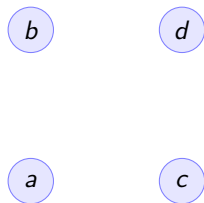
$$P(a, b, c, d) =$$

What is the model for this graph?



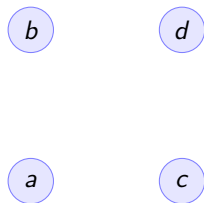
$$P(a, b, c, d) =$$

Draw the graph for this model



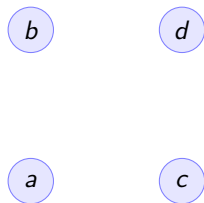
$$P(a, b, c, d) = P(a)P(b|a)P(c|b)P(d|b)$$

Draw the graph for this model



$$P(a, b, c, d) = P(a)P(b|a)P(d|c)P(c)$$

Draw the graph for this model



$$P(a, b, c, d) = P(a)P(b|a)P(c|a)P(d|b, c)$$

Graphical models

Graphical model

Exercises

Classification

Classification: Generative modelling

Density estimation

Density estimation

Algorithms for latent variable models

Gradient algorithms

Expectation maximisation

Exercises

Density estimation

Classification

Graphical models

Classification

Classification: Generative modelling

Density estimation

Algorithms for latent variable models

Exercises

Generative modelling

General idea

- ▶ Data (x_t, y_t) .
- ▶ Need to model $P(y|x)$.
- ▶ Model the **complete** data distribution: $P(x|y)$, $P(x)$, $P(y)$.
- ▶ Calculate $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$.

Generative modelling

General idea

- ▶ Data (x_t, y_t) .
- ▶ Need to model $P(y|x)$.
- ▶ Model the **complete** data distribution: $P(x|y)$, $P(x)$, $P(y)$.
- ▶ Calculate $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$.

Examples

- ▶ **Naive Bayes** classifier.
- ▶ **Gaussian mixture** model.
- ▶ Large language models.

Generative modelling

General idea

- ▶ Data (x_t, y_t) .
- ▶ Need to model $P(y|x)$.
- ▶ Model the **complete** data distribution: $P(x|y)$, $P(x)$, $P(y)$.
- ▶ Calculate $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$.

Examples


- ▶ **Naive Bayes** classifier.
- ▶ **Gaussian mixture** model.
- ▶ Large language models.

Modelling the data distribution in classification


- ▶ Need to estimate the density $P(x|y)$ for each class y .
- ▶ Need to estimate $P(y)$.

The basic graphical model


A discriminative classification model

Here $P(y|x)$ is given directly. 

A generative classification model

Here $P(y|x) = P(x|y)P(y)/P(x)$. 

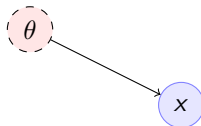
An unsupervised generative model

Here we just have $P(x)$. 

Adding parameters to the graphical model

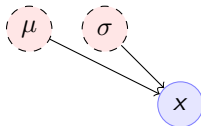
- ▶ Parameters are **latent** (hidden, unobserved)
- ▶ Data is **observed**
- ▶ We can learn about the parameters from the data.

A Bernoulli RV



Here, $x|\theta \sim \text{Bernoulli}(\theta)$

A normally distributed variable



Here $x|\mu, \sigma \sim \text{Normal}(\mu, \sigma^2)$

Classification: Naive Bayes Classifier

- ▶ Data (x, y)
- ▶ $x \in X$
- ▶ $y \in Y \subset \mathbb{N}$, N_i : amount of data from class i .

Classification: Naive Bayes Classifier

- ▶ Data (x, y)
- ▶ $x \in X$
- ▶ $y \in Y \subset \mathbb{N}$, N_i : amount of data from class i .

Separately model each class

- ▶ Assume each class data comes from a different normal distribution
- ▶ $x|y = i \sim \text{Normal}(\mu_i, \sigma_i I)$
- ▶ For each class, calculate
 - ▶ Empirical mean $\hat{\mu}_i = \sum_{t: y_t = i} x_t / N_i$
 - ▶ Empirical variance $\hat{\sigma}_i$.

Classification: Naive Bayes Classifier

- ▶ Data (x, y)
- ▶ $x \in X$
- ▶ $y \in Y \subset \mathbb{N}$, N_i : amount of data from class i .

Separately model each class

- ▶ Assume each class data comes from a different normal distribution
- ▶ $x|y = i \sim \text{Normal}(\mu_i, \sigma_i I)$
- ▶ For each class, calculate
 - ▶ Empirical mean $\hat{\mu}_i = \sum_{t: y_t = i} x_t / N_i$
 - ▶ Empirical variance $\hat{\sigma}_i$.

Decision rule

Use Bayes's theorem:

$$P(y|x) = P(x|y)P(y)/P(x),$$

choosing the y with largest posterior $P(y|x)$.

- ▶ $P(x|y = i) \propto \exp(-\|\hat{\mu}_i - x\|^2 / \hat{\sigma}_i^2)$

Graphical model for the Naive Bayes Classifier

When $x \in \mathbb{R}$

Assume k classes, then

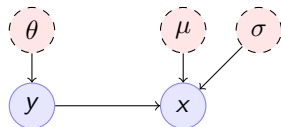
- ▶ $\mu = (\mu_1, \dots, \mu_k)$
- ▶ $\sigma = (\sigma_1, \dots, \sigma_k)$
- ▶ $\theta = (\theta_1, \dots, \theta_k)$

Graphical model for the Naive Bayes Classifier

When $x \in \mathbb{R}$

Assume k classes, then

- ▶ $\mu = (\mu_1, \dots, \mu_k)$
- ▶ $\sigma = (\sigma_1, \dots, \sigma_k)$
- ▶ $\theta = (\theta_1, \dots, \theta_k)$



- ▶ $y \mid \theta \sim \text{Mult}(\theta)$
- ▶ $x \mid y, \mu, \sigma \sim \text{Normal}(\mu_y, \sigma_y^2)$

Graphical models

Graphical model

Exercises

Classification

Classification: Generative modelling

Density estimation

Density estimation

Algorithms for latent variable models

Gradient algorithms

Expectation maximisation

Exercises

Density estimation

Classification

Density estimation

General idea

Parametric models

- ▶ Fixed histograms
- ▶ Gaussian Mixtures

Non-parametric models

- ▶ Variable-bin histograms
- ▶ Infinite Gaussian Mixture Model
- ▶ Kernel methods

Histograms

Fixed histogram

- ▶ Hyper-Parameters: number of bins
- ▶ Parameters: Number of points in each bin.

Variable histogram

- ▶ Hyper-parameters: Rule for constructing bins
- ▶ Generally \sqrt{n} points in each bin.

Gaussian Mixture Model

Hyperparameters:

- ▶ Number of Gaussian k .

Parameters:

- ▶ Multinomial distribution θ over Gaussians
- ▶ For each Gaussian i , center μ_i , covariance matrix Σ_i .

Algorithms:

- ▶ Expectation Maximisation
- ▶ Gradient Ascent
- ▶ Variational Bayesian Inference (with appropriate prior)

Details of Gaussian mixture models

Model. For each point x_t :

- ▶ $z_t \mid \theta \sim \text{Mult}(\theta_i), \theta \in \Delta^k$
- ▶ $x_t \mid z_t = i \sim \text{Normal}(\mu_i, \Sigma_i).$
- ▶ $\text{Mult}(\theta)$ is **multinomial**

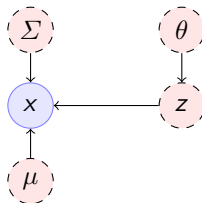
$$\mathbb{P}(z_t = i \mid \theta) = \theta_i$$

- ▶ $\text{Normal}(\mu, \Sigma)$ is **multivariate Gaussian**

$$p(x \mid \mu, \Sigma) \propto \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right)$$

- ▶ The generating distribution is

$$p(x \mid \theta, \mu, \Sigma) = \sum_{z \in [k]} p(x \mid \mu_z, \Sigma_z) P(z \mid \theta).$$



Graphical models

Graphical model

Exercises

Classification

Classification: Generative modelling

Density estimation

Density estimation

Algorithms for latent variable models

Gradient algorithms

Expectation maximisation

Exercises

Density estimation

Classification

Gradient ascent

In the following we use θ for all the parameters of the Gaussian mixture model, with $x = (x_1, \dots, x_T)$ and $z = (z_1, \dots, z_T)$

Gradient ascent

In the following we use θ for all the parameters of the Gaussian mixture model, with $x = (x_1, \dots, x_T)$ and $z = (z_1, \dots, z_T)$

Objective function

One way to estimate θ is through maximising the likelihood $L(\theta) = P(x|\theta)$

Gradient ascent

In the following we use θ for all the parameters of the Gaussian mixture model, with $x = (x_1, \dots, x_T)$ and $z = (z_1, \dots, z_T)$

Objective function

One way to estimate θ is through maximising the likelihood $L(\theta) = P(x|\theta)$

Marginalisation over latent variable

However, we need to marginalise over all values z

$$L(\theta) = \sum_z P(z, x|\theta)$$

For T data points and k different values of z_t , there are k^T vectors z to sum over.

Gradient ascent

In the following we use θ for all the parameters of the Gaussian mixture model, with $x = (x_1, \dots, x_T)$ and $z = (z_1, \dots, z_T)$

Objective function

One way to estimate θ is through maximising the likelihood $L(\theta) = P(x|\theta)$

Marginalisation over latent variable

However, we need to marginalise over all values z

$$L(\theta) = \sum_z P(z, x|\theta)$$

For T data points and k different values of z_t , there are k^T vectors z to sum over.

Gradient ascent

If we can calculate the gradient of L , we can use gradient ascent to update our parameters:

$$\theta^{(n+1)} = \theta^{(n)} + \alpha \nabla_{\theta} L(\theta).$$

Gradient calculation

Here we use the **log trick**: $\nabla \ln f(\theta) = \nabla f(\theta)/f(\theta)$.

Gradient calculation

Here we use the **log trick**: $\nabla \ln f(\theta) = \nabla f(\theta)/f(\theta)$.

$$\nabla_{\theta} L(\theta) = \sum_z \nabla_{\theta} P(z, x | \theta) \quad (6)$$

$$= \sum_z P(z, x | \theta) \nabla_{\theta} \ln P(z, x | \theta) \quad (7)$$

$$= \sum_z P(x | z, \theta) P(z | \theta) \nabla_{\theta} \ln P(z, x | \theta) \quad (8)$$

$$\approx \frac{1}{m} \sum_{i=1}^m P(x | z^{(i)}, \theta) \nabla_{\theta} \ln P(z^{(i)}, x | \theta) \quad z^{(i)} \sim P(z | \theta) \quad (9)$$

Gradient calculation

Here we use the **log trick**: $\nabla \ln f(\theta) = \nabla f(\theta)/f(\theta)$.

$$\nabla_{\theta} L(\theta) = \sum_z \nabla_{\theta} P(z, x | \theta) \quad (6)$$

$$= \sum_z P(z, x | \theta) \nabla_{\theta} \ln P(z, x | \theta) \quad (7)$$

$$= \sum_z P(x | z, \theta) P(z | \theta) \nabla_{\theta} \ln P(z, x | \theta) \quad (8)$$

$$\approx \frac{1}{m} \sum_{i=1}^m P(x | z^{(i)}, \theta) \nabla_{\theta} \ln P(z^{(i)}, x | \theta) \quad z^{(i)} \sim P(z | \theta) \quad (9)$$

Hence, we can implement the following algorithm:

- ▶ For $i = 1, \dots, m$: $z^{(i)} \sim P(z | \theta^{(n)})$
- ▶ $d^{(n)} = \frac{1}{m} \sum_{i=1}^m P(x | z^{(i)}, \theta) \nabla_{\theta} \ln P(z^{(i)}, x | \theta^{(n)})$
- ▶ $\theta^{(n+1)} = \theta^{(n)} + \alpha d^{(n)}$.

A lower bound on the likelihood

For any distribution $G(z)$, and specifically for $G(z) = P(z|x, \theta^{(k)})$:

$$\begin{aligned}
 \ln P(x|\theta) &= \sum_z G(z) \ln P(x|\theta) = \sum_z G(z) \ln[P(x, z|\theta)/P(z|x, \theta)] \\
 &= \sum_z G(z) [\ln P(x, z|\theta) - \ln P(z|x, \theta)] \\
 &= \sum_z G(z) \ln P(x, z|\theta) - \sum_z G(z) \ln P(z|x, \theta) \\
 &= \sum_z P(z|x, \theta^{(k)}) \ln P(x, z|\theta) - \sum_z P(z|x, \theta^{(k)}) \ln P(z|x, \theta) \\
 &\geq \sum_z P(z|x, \theta^{(k)}) \ln P(x, z|\theta) - \sum_z P(z|x, \theta^{(k)}) \ln P(z|x, \theta^{(k)}) \\
 &= Q(\theta | \theta^{(k)}) + \mathbb{H}(z | x, \theta^{(k)}),
 \end{aligned}$$

where

$$\mathbb{H}(z | x, \theta^{(k)}) = \sum_z P(z | x, \theta^{(k)}) \ln P(z | x, \theta^{(k)})$$

is the entropy of z for a fixed $x, \theta^{(k)}$. As this is not negative,
 $\ln P(x|\theta) \geq Q(\theta | \theta^{(k)})$.

Some information theory

Information theory notation can be a bit confusing. Sometimes we talk about random variables ω , and sometimes about probability measures P . This is context-dependent.

Entropy

For a random variable ω under distribution P , we denote the entropy as

$$\mathbb{H}_P(\omega) \equiv \mathbb{H}(P) \equiv \mathbb{H}(\omega) = \sum_{\omega \in \Omega} P(\omega) \ln P(\omega).$$

KL Divergence

For two probabilities P, Q over random outcomes in the same space Ω , we define

$$D_{KL}(P\|Q) = \sum_{\omega \in \Omega} P(\omega) \ln \frac{P(\omega)}{Q(\omega)}$$

The Gibbs Inequality

$$D_{KL}(P\|Q) \geq 0, \text{ or } \sum_x \ln P(x)P(x) \geq \sum_x \ln Q(x)P(x)$$

EM Algorithm (Dempster et al, 1977)

- ▶ Initial parameter $\theta^{(0)}$, observed data x
- ▶ For $k = 0, 1, \dots$

– Expectation step:

$$Q(\theta \mid \theta^{(k)}) \triangleq \mathbb{E}_{z \sim P(z|x, \theta^{(k)})} [\ln P(x, z | \theta)] = \sum_z [\ln P(x, z | \theta)] P(z \mid x, \theta^{(k)})$$

– Maximisation step:

$$\theta^{(k+1)} = \arg \max_{\theta} Q(\theta, \theta^{(k)}).$$

See *Expectation-Maximization as lower bound maximization*, Minka, 1998

Minorise-Maximise

EM can be seen as a version of the minorise-maximise algorithm

- ▶ $f(\theta)$: Target function to **maximise**
- ▶ $Q(\theta|\theta^{(k)})$: surrogate function

Q Minorizes f

This means surrogate is always a lower bound so that

$$f(\theta) \geq Q(\theta|\theta^{(k)}), \quad f(\theta^{(k)}) \geq Q(\theta^{(k)}|\theta^{(k)}),$$

Algorithm

- ▶ Calculate: $Q(\theta|\theta^{(k)})$
- ▶ Optimise: $\theta^{(k+1)} = \arg \max_{\theta} Q(\theta|\theta^{(k)})$.

Graphical models

Graphical model

Exercises

Classification

Classification: Generative modelling

Density estimation

Density estimation

Algorithms for latent variable models

Gradient algorithms

Expectation maximisation

Exercises

Density estimation

Classification

GMM versus histogram

- ▶ Generate some data x from an arbitrary distribution in \mathbb{R} .
- ▶ Fit the data with a histogram for varying numbers of bins
- ▶ Fit a GMM with varying numbers of Gaussians
- ▶ What is the best fit? How can you measure it?

GMM Classifier

Base class: sklearn GaussianMixtureModel

- ▶ *fit()* only works for Density Estimation
- ▶ *predict()* only predicts cluster labels

Problem

- ▶ Create a GMMClassifier class
- ▶ *fit()* should take X, y, arguments
- ▶ *predict()* should predict class labels
- ▶ Hint: Use *predict_proba()* and multiple GMM models