Review of some concepts in computational statistics

October 13, 2022

1 Reviewing some useful concepts in computational statistics

To quantify the properties of a random variable, we use the concept of **discrete distributions** (discrete r.v.) or **probability density functions** (continuous r.v.).

In the case of **discrete random variables x**:

- Bernoulli, described by $\pi \triangleq p(\mathbf{x} = 1)$.
- Categorical distribution, among K categories $1...K$, described by $\pi_k \triangleq p(\mathbf{x} = k)$ such that $\sum_{k=1}^{K} \pi_k = 1$.

In the case of **continuous random variables x**, $f_{\mathbf{x}}(\mathbf{x})$

$$p(\mathbf{x} \leq X) = \int_{-\infty}^{X} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}.$$

We need to satisfy:

$$\int_{-\infty}^{+\infty} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = 1.$$

Examples (in the mono-variate case):

- Gaussian
$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}\right).$$
- Exponential
$$f_{\mathbf{x}}(\mathbf{x}) = \begin{cases} \lambda \exp(-\lambda \mathbf{x}) & \text{for } \mathbf{x} \geq 0, \\ 0 \text{ otherwise.} \end{cases}$$

Joint distributions result from a collection of random variables, and evaluate the conjunction of realizations for these variables.

$$p(\mathbf{x}_1, \mathbf{x}_2).$$

How many parameters to describe this joint distribution of the pixel values for these binary $w \times h$ images?

Each pixel of the image can be seen as the realization of a **Bernoulli variable**.

To describe the possible $2^{w \times h}$ states:

$$2^{w \times h} - 1$$

parameters. **Combinatorial** nature of the joint distributions.

Now, a lost of efforts is done in controlling the modeling size and leverage **structure** through identifying **(in-)dependencies**.

An extreme case is **fully independent variables**:

$$p(\mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{x}_1)p(\mathbf{x}_2)...p(\mathbf{x}_m).$$

In the case of the $w \times h$ images, $m = w \times h$ and:

- again $2^{w \times h}$ possible states,
- only $w \times h$ parameters to describe this factorized distribution,
- but the assumption is probably **not very realistic**!

Bayes rule

$$p(\mathbf{x}_1|\mathbf{x}_2) = \frac{p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1)}{p(\mathbf{x}_2)} = \frac{p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1)}{\sum_{\mathbf{x}_1} p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1)}.$$

Chain rule

For a collection of r.v. $\mathbf{x}_1, ..., \mathbf{x}_m$,

$$p(\mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)...p(\mathbf{x}_m|\mathbf{x}_1, ..., \mathbf{x}_{m-1}).$$

This is an **exact factorization** of the joint distribution.

Note that, supposing $\mathbf{x}_i$ are binary variables,

- to define $p(\mathbf{x}_1, ..., \mathbf{x}_m)$ directly, you would need

$$2^m - 1$$

values.

- by using the chain rule: each of the **conditional probabilities** $p(\mathbf{x}_i|\mathbf{x}_1, ..., \mathbf{x}_{i-1})$ would be specified with a table with 2^{i-1} entries; hence in total:

$$1 + 2 + ... + 2^{m-1} = 2^m - 1.$$

(i.e you do not simplify anything!)

Now suppose **conditional independences** of the following form, for any i:

$$\mathbf{x}_{i+1} \perp\!\!\!\perp \mathbf{x}_1, ..., \mathbf{x}_{i-1}|\mathbf{x}_i,$$

then

$$p(\mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_2)...p(\mathbf{x}_m|\mathbf{x}_{m-1}).$$

(**Markov chain**)

How many parameters?

Reduction **from exponential to linear**:

$$1 + 2(m - 1) = 2m - 1.$$

Bayesian networks generalize this idea of using conditional independences within a **directed graph**:

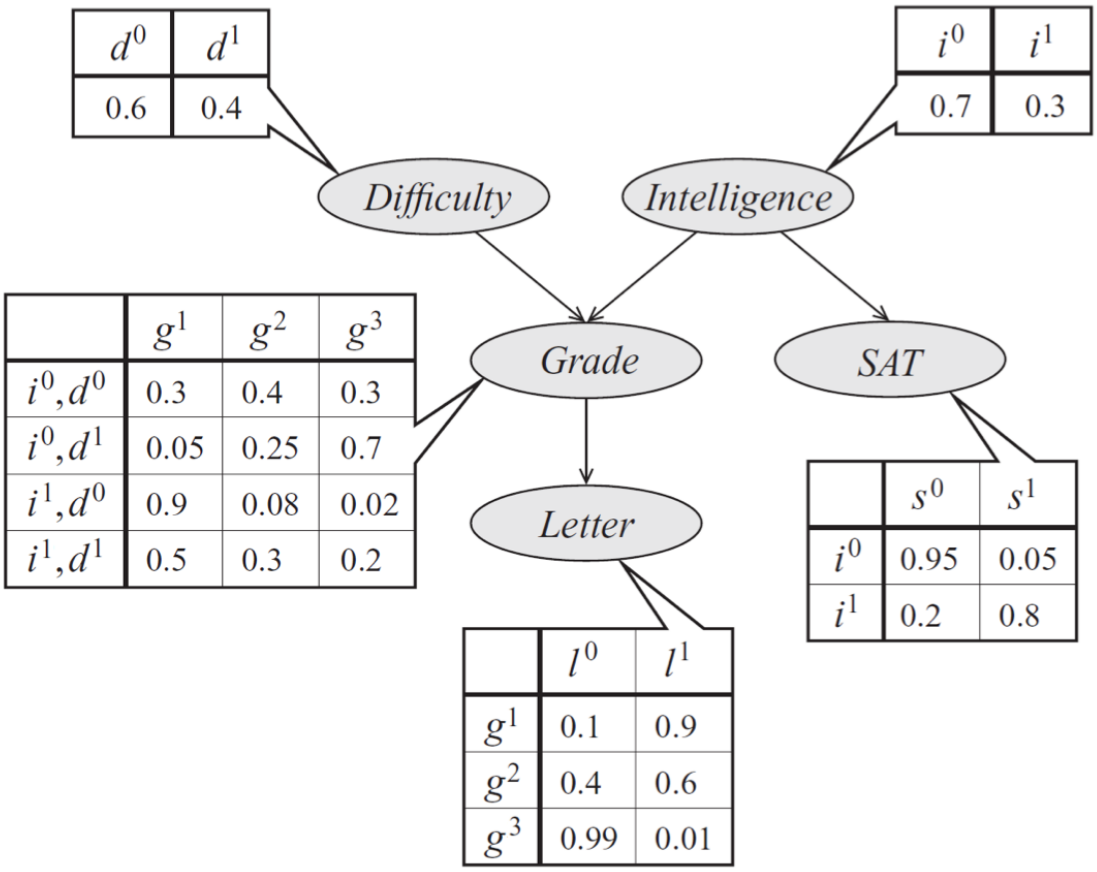
$$p(\mathbf{x}_1, ..., \mathbf{x}_m) = \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{x}_{\pi(i)})$$

where $\pi(i)$ describes the set of parents of r.v. i.

Remember:

$$p(\mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)...p(\mathbf{x}_m|\mathbf{x}_1, ..., \mathbf{x}_{m-1}).$$

The two should be consistent: a Bayesian netwok should correspond to **one (or maybe several) chain rule factorization**.



Chain rule factorization?

$$p(\mathbf{d}, \mathbf{i}, \mathbf{s}, \mathbf{g}, \mathbf{l}) = p(\mathbf{d})p(\mathbf{i})p(\mathbf{s}|\mathbf{i})p(\mathbf{g}|\mathbf{d}, \mathbf{i})p(\mathbf{l}|\mathbf{g}).$$

Is it the only one?

A Bayesian network corresponds to a **directed acyclic graph (DAG)** where the nodes are the random variables and the edges are the conditional dependencies.

To specify each variable, you use a **conditional probability distribution (CPD)**, $p(\mathbf{x}_i|\mathbf{x}_{\pi(i)})$, which can be given e.g., through **tables** in the discrete case.

All the possible **topological orders** induced by the graph gives "valid" chain rule factorizations. For example:

$$p(\mathbf{i}, \mathbf{d}, \mathbf{g}, \mathbf{s}, \mathbf{l}) = p(\mathbf{i})p(\mathbf{d})p(\mathbf{g}|\mathbf{d}, \mathbf{i})p(\mathbf{s}|\mathbf{i})p(\mathbf{l}|\mathbf{g}).$$

Note that a given Bayesian network implies:

- in general, **several possible orders** of the random variables to produce a valid chain rule factorization,

- conditional independencies.

1.1 Discriminative versus generative models

In most modern ML problems, you have to deal with:

- Data $\mathbf{x}$, represented through m features/observations: $\mathbf{x} \triangleq (\mathbf{x}_1, .., \mathbf{x}_m)^T$.
- Target values (labels, continuous values): $\mathbf{y}$.

For example, consider images $\mathbf{x} \triangleq (\mathbf{x}_1, .., \mathbf{x}_m)^T$ (could be the pixels) and $\mathbf{y}$ a label (among K categories) that we assign to the images: image **classification**.

One approach to solve such a classification problem is **Naive Bayes**.

It assumes the following:

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}, \mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{y}) \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{y}).$$

i.e., that if I give you the label, the features are **independent**.

Prediction of the label of the image results from:

$$p(\mathbf{y} = k|\mathbf{x}_1, ..., \mathbf{x}_m) = \frac{p(\mathbf{y} = k, \mathbf{x}_1, ..., \mathbf{x}_m)}{p(\mathbf{x}_1, ..., \mathbf{x}_m)} = \frac{p(\mathbf{y} = k) \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{y} = k)}{\sum_{k'} p(\mathbf{y} = k') \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{y} = k')}.$$

In this context, this is a **generative model**, since it focuses on modeling and learning (through examples)

$$p(\mathbf{x}_i|\mathbf{y})$$

i.e., "explaining" how the data can be **produced given the label $\mathbf{y}$**.

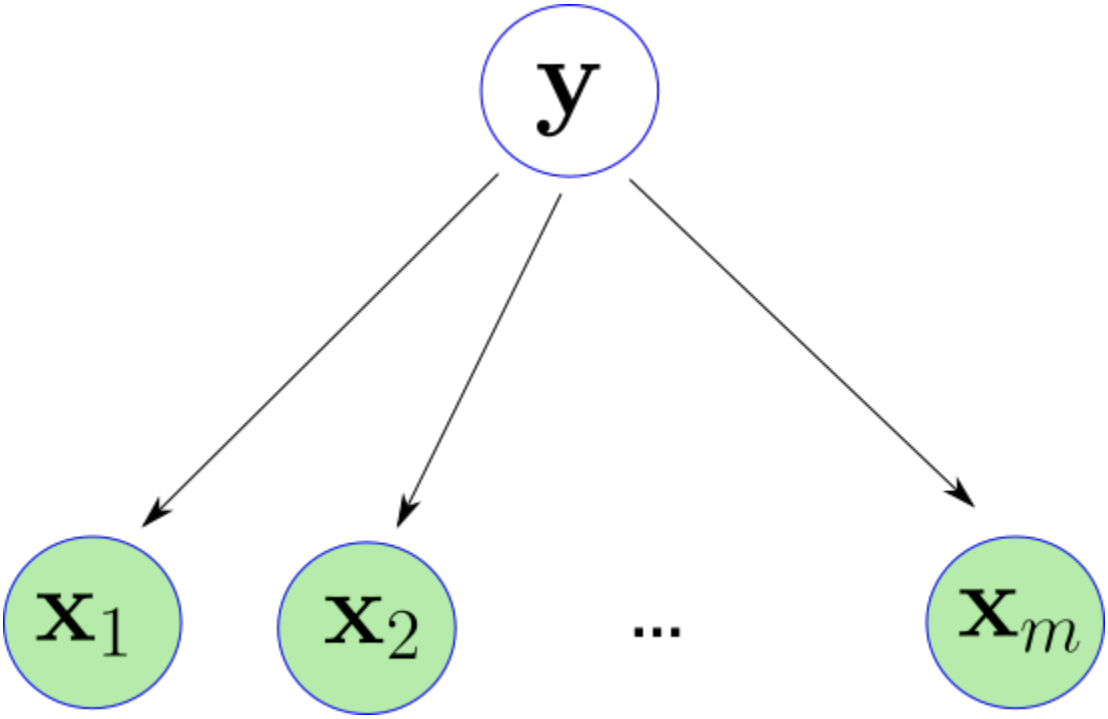
If you have access to $p(\mathbf{x}_i|\mathbf{y})$, then a priori, you may

- **sample** from it;
- evaluate how likely are some features $\mathbf{x}_i$.

The corresponding (and more general) factorization **through chain rule** would be

$$p(\mathbf{y}, \mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{y}) \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{y}, \mathbf{x}_1, ..., \mathbf{x}_{i-1}).$$

We see again that Naive Bayes assumes the $\mathbf{x}_i$ as conditionally independent given the label $\mathbf{y}$.



Now, solving the **inference** problem is a bit complex because we have to use **the Bayes rule and normalize the obtained values into probabilities**:

$$p(\mathbf{y} = k|\mathbf{x}_1, ..., \mathbf{x}_m) = \frac{p(\mathbf{y} = k) \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{y} = k)}{\sum_{k'} p(\mathbf{y} = k') \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{y} = k')} \tag{1}$$

$$k^* = \arg\max_k p(\mathbf{y} = k|\mathbf{x}_1, ..., \mathbf{x}_m). \tag{2}$$

On the other side, note that something like a logistic model **does not assume any conditional independence** (not like naive Bayes): a priori more general in terms of the data structure of $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_m)^T$.

But the advantage is that you have access to all these distributions:

$$p(\mathbf{x}_i|\mathbf{y}),$$

$$p(\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_m|\mathbf{y}) = \prod_i p(\mathbf{x}_i|\mathbf{y}),$$

$$p(\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_m) = \sum_k p(\mathbf{y} = k) \prod_i p(\mathbf{x}_i|\mathbf{y} = k).$$

So, we have a **probabilistic model for our data** from which, for example, we could **sample** new data.

On the opposite are **discriminative models** that use this other factorization:

$$p(\mathbf{y}, \mathbf{x}_1, ..., \mathbf{x}_m) = \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{x}_1, ..., \mathbf{x}_{i-1}) p(\mathbf{y}|\mathbf{x}_1, ..., \mathbf{x}_m) = p(\mathbf{x}_1, ..., \mathbf{x}_m) p(\mathbf{y}|\mathbf{x}_1, ..., \mathbf{x}_m).$$

This is much more practical for **inference** since $p(\mathbf{x}_1, ..., \mathbf{x}_m)$ is a **constant** in this case, hence:

$$k^* = \arg\max_k p(\mathbf{y} = k|\mathbf{x}_1, ..., \mathbf{x}_m). \tag{3}$$

The problem with this focus is to **model** $p(\mathbf{y}|\mathbf{x}_1, ..., \mathbf{x}_m)$.

Logistic regression is a classical example of **discriminative model**: It uses a **closed form, parameterized function** to represent $p(\mathbf{y}|\mathbf{x}_1, ..., \mathbf{x}_m)$ (no table here)

$$p(\mathbf{y} = 1|\mathbf{x}_1, ..., \mathbf{x}_m) = \sigma(\alpha_0 + \sum_{i=1}^{m} \alpha_i \mathbf{x}_i)$$

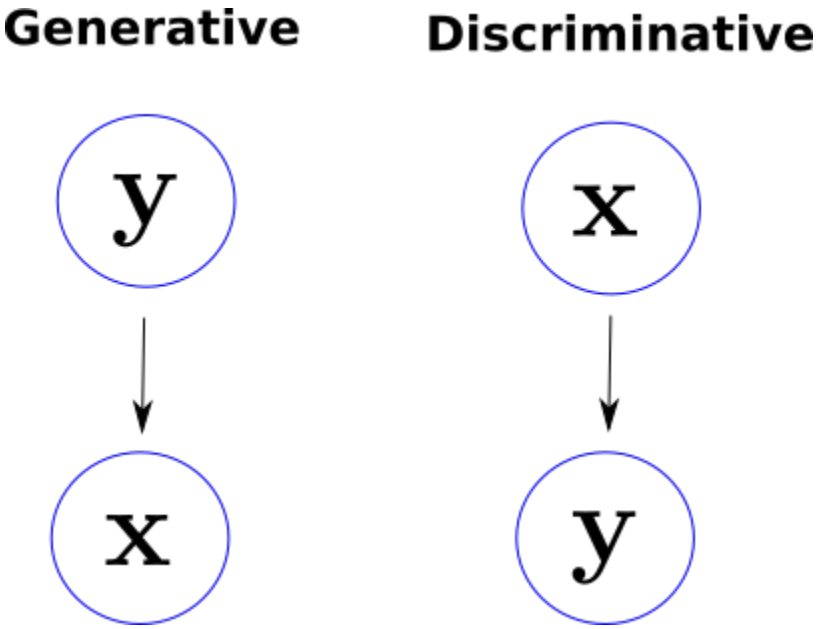
with

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

Directly specifies a discrimination rule (through a **hyperplane** in $\mathbb{R}^m$).

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{x}|\mathbf{y}) p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}).$$

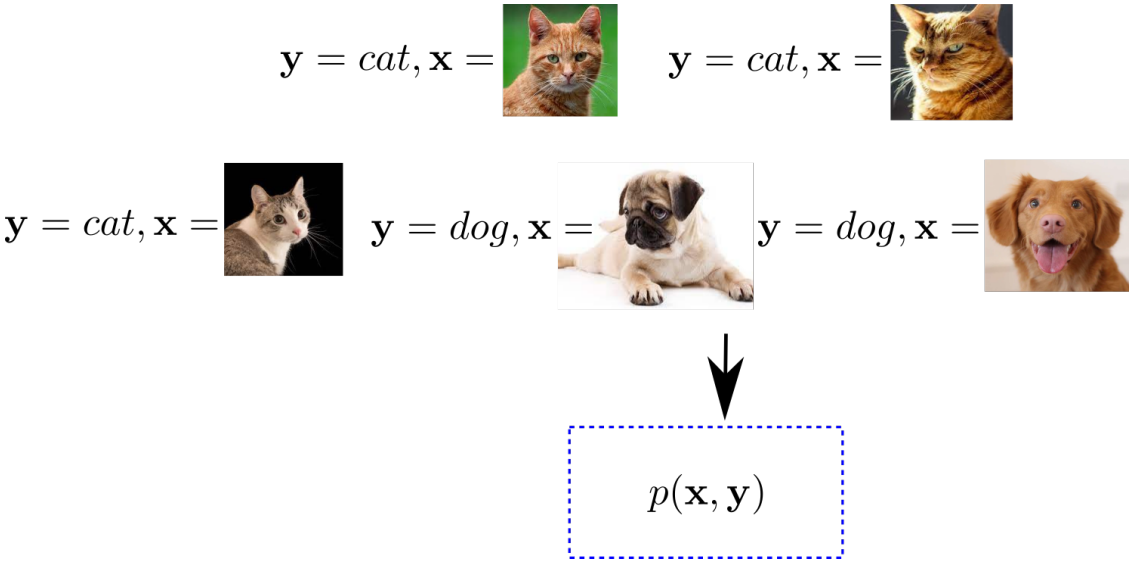
Two ways of factorizing, and two corresponding graphs:

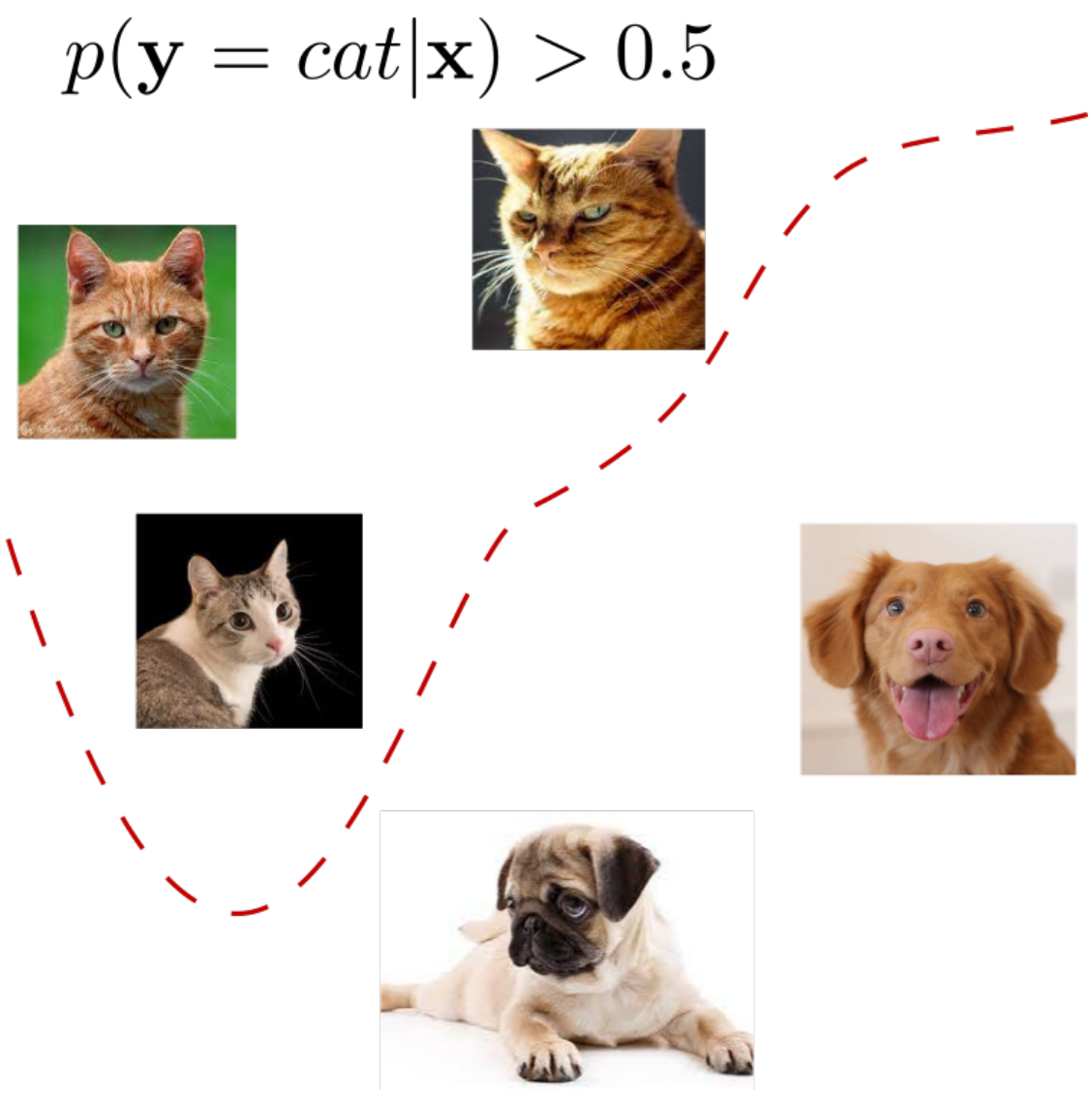


Sometimes, we just need $p(\mathbf{y}|\mathbf{x})$ since $\mathbf{x}$ is given and the required "answer" to the problem is about $\mathbf{y}$.

With a **generative model**, we need specify/learn both $p(\mathbf{y})$ and $p(\mathbf{x}|\mathbf{y})$, then we compute $p(\mathbf{y}|\mathbf{x})$ with the Bayes rule.

With a **discriminative model**, we simply evaluate $p(\mathbf{y}|\mathbf{x})$, with no need to have the distribution of $\mathbf{x}$.





However, sometimes:

- we may be interested by **mimicking some data distribution**, so we need a way to build a representation for

$$p(\mathbf{x})$$

from which we could sample new data.

- we may be interested in filtering out some outlier data, this can be done by **evaluating** $p(\mathbf{x})$,
- we may want to **understand the structure of the data**.

Another advantage of generative modeling: We can sort out **incomplete** observations. Imagine you have $\mathbf{x}_1, ..., \mathbf{x}_p$ and do **not** observe $\mathbf{x}_{p+1}, ..., \mathbf{x}_m$

$$p(\mathbf{y}=k|\mathbf{x}_1,....,\mathbf{x}_p) = \frac{p(\mathbf{y}=k)p(\mathbf{x}_1,....,\mathbf{x}_p|\mathbf{y}=k)}{p(\mathbf{x}_1,....,\mathbf{x}_p)} \quad (4)$$

$$= \frac{p(\mathbf{y}=k)\sum_{\mathbf{x}_{p+1},...,\mathbf{x}_m} p(\mathbf{x}_1,....,\mathbf{x}_m|\mathbf{y}=k)}{\sum_{k'} p(\mathbf{y}=k')\sum_{\mathbf{x}_{p+1},...,\mathbf{x}_m} p(\mathbf{x}_1,....,\mathbf{x}_m|\mathbf{y}=k')} \quad (5)$$

We can perform inference in that case by **marginalizing over the unseen variables**.

Another motivation: Solve **inverse** problems.

Imagine some noisy projection process of our data from $\mathbb{R}^m$ to $\mathbb{R}^p$ with $p < m$

$$\mathbf{y} = \mathbf{P}\mathbf{x} + \eta.$$

A classical problem is, given $\mathbf{y}$, to find an $\mathbf{x}$ explaining well $\mathbf{y}$.

- Problem is **ill-posed**: lots of possible solutions.
- Needs **some priors** on the potential solutions.
- One (classical) way: enforce **sparsity** of the solution.
- With generative models, in particular latent-variable based models:

$$p(\mathbf{x})$$

is used **as a prior**.

Some generative models act as a **parameterized mapping from some latent space**

$$\mathbf{x} = g_\theta(\mathbf{z}) \text{ with } \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

then inverse problems for observations $\mathbf{y}$ can be formulated as

$$\arg\min_{\mathbf{z}} \|\mathbf{y} - \mathbf{P}g_\theta(\mathbf{z})\|.$$

The problem above is much less "open", since you search for latent variables, not directly for the data.

2 Generative models

A **statistical generative model** is a probability distribution $p(\mathbf{x})$ over some data $\mathbf{x}$.

To build this model, we will use:

- Data samples $\mathbf{x}^1, ..., \mathbf{x}^n$ (e.g., a collection of images of faces)
- Eventually, some explicit or implicit **priors**: parametric form, terms in loss function.

We may also try to learn **conditional generative models**:

$$p(\mathbf{x}|\mathbf{y}).$$

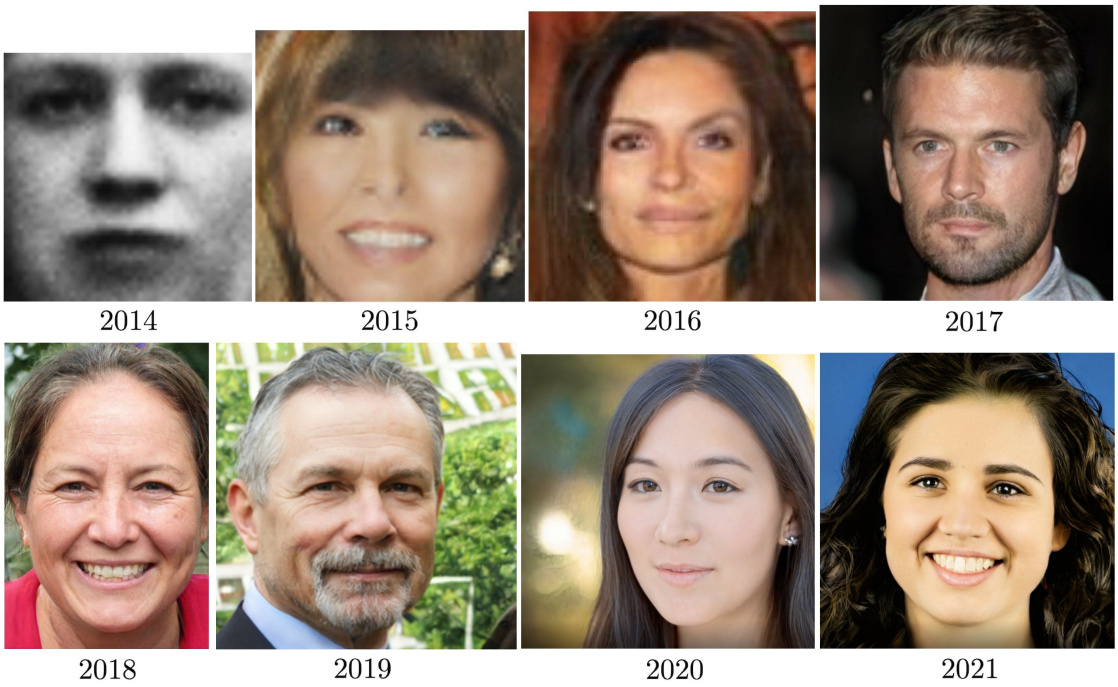
The way we will handle the generative model should ideally allow us to:

- Perform **sampling**: Generate new data $\mathbf{x}$ that "look like" the training data.
- Perform **density estimation**: produce an estimate of $p(\mathbf{x})$, that should be high if our data $\mathbf{x}$ looks like the training data, low otherwise (out-of-distribution detection).

One of the most important point: Learning to produce such a distribution typically implies, by design, **understanding well the structure of the data**: form of unsupervised representation learning; we should be able to learn features that represent what these data have in common.

We will see:

- Auto-regressive models.
- Normalizing flows.
- Variational auto-encoders.
- Generative adversarial networks.

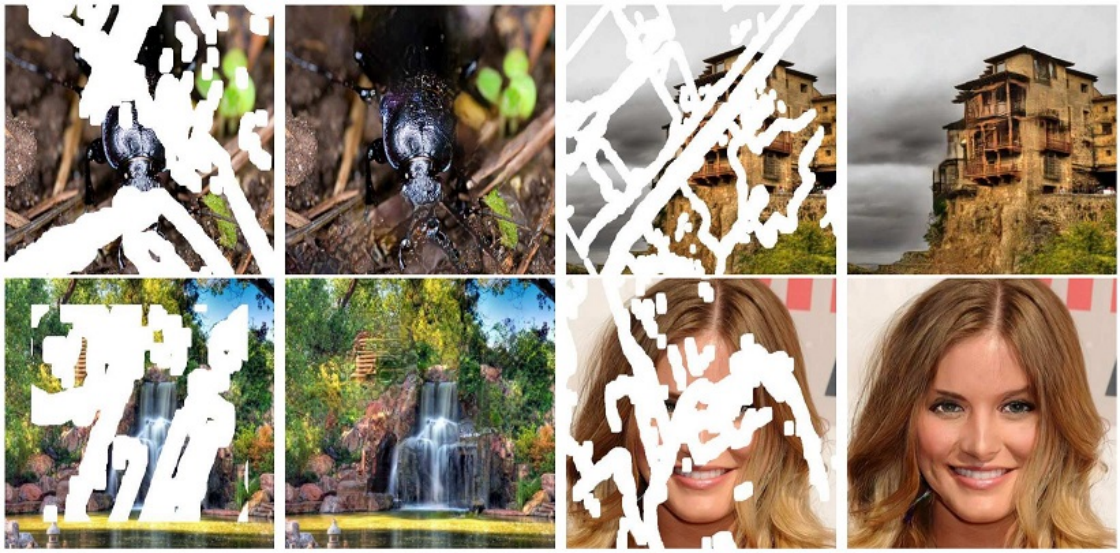


@tamaybes

—

—

[Menon et al, 2020]



[Liu et al, 2018]

[Antic et al, 2020]