Machine Learning for Physics and Astronomy

Juan Rojo

VU Amsterdam & Theory group, Nikhef

Natuur- en Sterrenkunde BSc (Joint Degree), Honours Track

Lecture 6, 05/10/2020

Today's lecture

- Information Theory revisited (pre-recorded)
- Generative Models in ML (pre-recorded)
- Adversarial Learning and Generative Adversarial Networks
- Energy-based models
- Guest lecture by **Dr. Sascha Caron** on applications of ML to high-energy physics

Information Theory revisited

Information Theory

the main ideas of **information theory** are ubiquitous in machine learning. In order to discuss generative models and adversarial learning it is useful to review a few basic concepts here

consider a discrete random variable *x*. How much information is brought in if we measure an specific value of this variable (its **degree of surprise**)?

the greater the surprise, the bigger the amount of info

e.g. ``the sun did not rise this morning" contains more surprise than ``the sun rose this morning"!

clearly a measure of ``*how much information a new measurement brings"* depends on the probability distribution of *x,* ***p(x).*** Thus we look for an information-measuring function **h[p(x)]**

Information Theory

the main ideas of **information theory** are ubiquitous in machine learning. In order to discuss generative models and adversarial learning it is useful to review a few basic concepts here

consider a discrete random variable *x*. How much information is brought in if we measure an specific value of this variable (its **degree of surprise**)?

the greater the surprise, the bigger the amount of info
e.g. ``the sun did not rise this morning" contains more surprise than ``the sun rose this morning"!

clearly a measure of ``*how much information a new measurement brings"* depends on the probability distribution of *x,* ***p(x).*** Thus we look for an information-measuring function **h[p(x)]**

if two events x and y are independent, the information we gain from observing both of them is the sum of the information we gain from each of them separately

$$h(x, y) = h(x) + h(y)$$

But these two events are statistically independent, thus their probabilities obey

$$p(x, y) = p(x)p(y)$$

Information Theory

the main ideas of **information theory** are ubiquitous in machine learning. In order to discuss generative models and adversarial learning it is useful to review a few basic concepts

consider a discrete random variable x. How much information is brought in if we measure an specific value of this variable (its **degree of surprise**)?

the greater the surprise, the bigger the amount of info

e.g. ``the sun did not rise this morning" contains more surprise than ``the sun rose this morning"!

clearly a measure of ``*how much information a new measurement brings"* depends on the probability distribution of *x,* ***p(x).*** Thus we look for an information-measuring function **h[p(x)]**

if two events x and y are independent, the information we gain from observing both of them is the sum of the information we gain from each of them separately

$$h(x, y) = h(x) + h(y)$$

But these two events are statistically independent, thus their probabilities obey

$$p(x, y) = p(x)p(y)$$

information is positive!

$$h(x) = -\ln p(x)$$

$$0 \leq p(x) \leq 1$$

Information Theory

what is the **total amount of information** contained in a set of instances of ***x***?

Ii will be the expectation value of h(x) wrt the probability distribution p(x)

$$H[x] = \sum_x p(x)h(x) = -\sum_x p(x)\ln p(x)$$

which is known as the **entropy** associated to the random variable ***x***

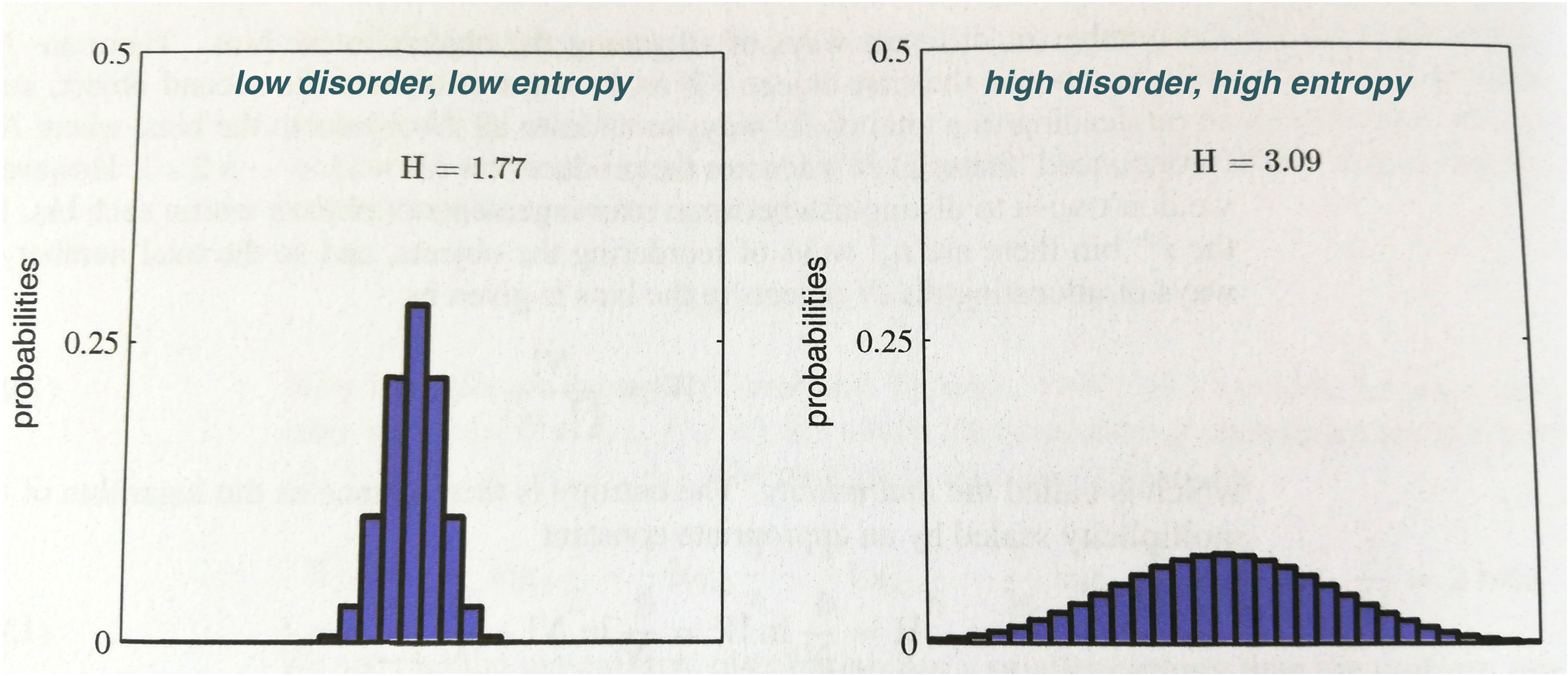
Note some **important properties** about the *surprise function h(x)* and the entropy *H[x]:*

- ☑ Low-probability events have associated a **high information content**
- ☑ However low-probability events carry a **small weight** in the total entropy
- ☑ One can show that steeply-varying probability distributions lead to smaller entropies than smooth distributions (largest entropy is for uniform distributions)
- ☑ Entropy can be understood to as the **average amount of information** needed to specify the state of a random variable

entropy is of course closely connected
to disorder: the higher the disorder, the largest the associated entropy

Information Theory

the broader the distribution, the higher the disorder and thus the higher the entropy



the highest possible value of the entropy corresponds to a uniform distribution

Statistical distances and similarity

In many applications for machine learning we want to assess the **degree of similarity** between two probability distributions: these metrics are known as **statistical distances**

The Kullback-Leibler divergence is a measure of the **similarity** between two probability distributions *p(**x**)* and *q(**x**)* and plays an important role in machine learning models

$$D_{KL}\left(p\,||\,q\right) = \int d\boldsymbol{x}\, p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \qquad D_{KL}\left(q\,||\,p\right) = \int d\boldsymbol{x}\, q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}$$

note that KL divergence is not symmetric (as usual with statistical distances)

$$D_{JS}\left(p\,||\,q\right) = \frac{1}{2}\left(D_{KL}\left(p\,\middle|\middle|\,\frac{p+q}{2}\right) + D_{KL}\left(q\,\middle|\middle|\,\frac{p+q}{2}\right)\right)$$

Jensen-Shannon divergence (symmetric version of KL divergence)

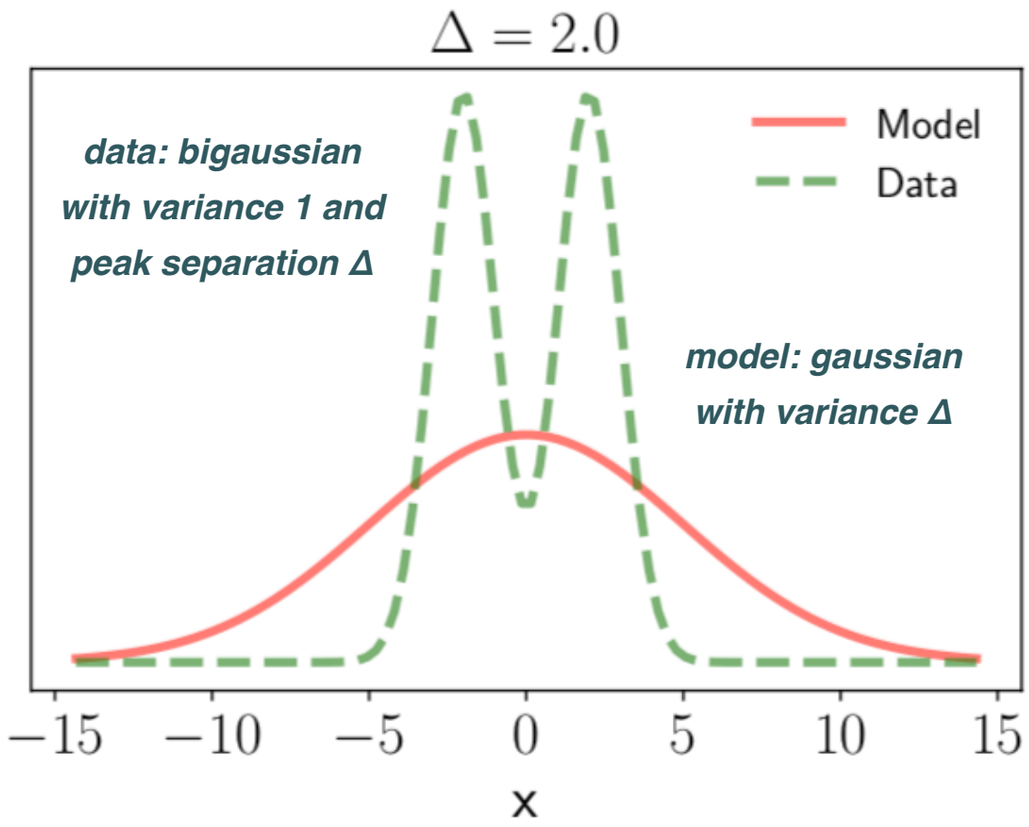
The KL-divergence is **positive-definite**, and only vanishes when *p(**x**)*= q*(**x**)*

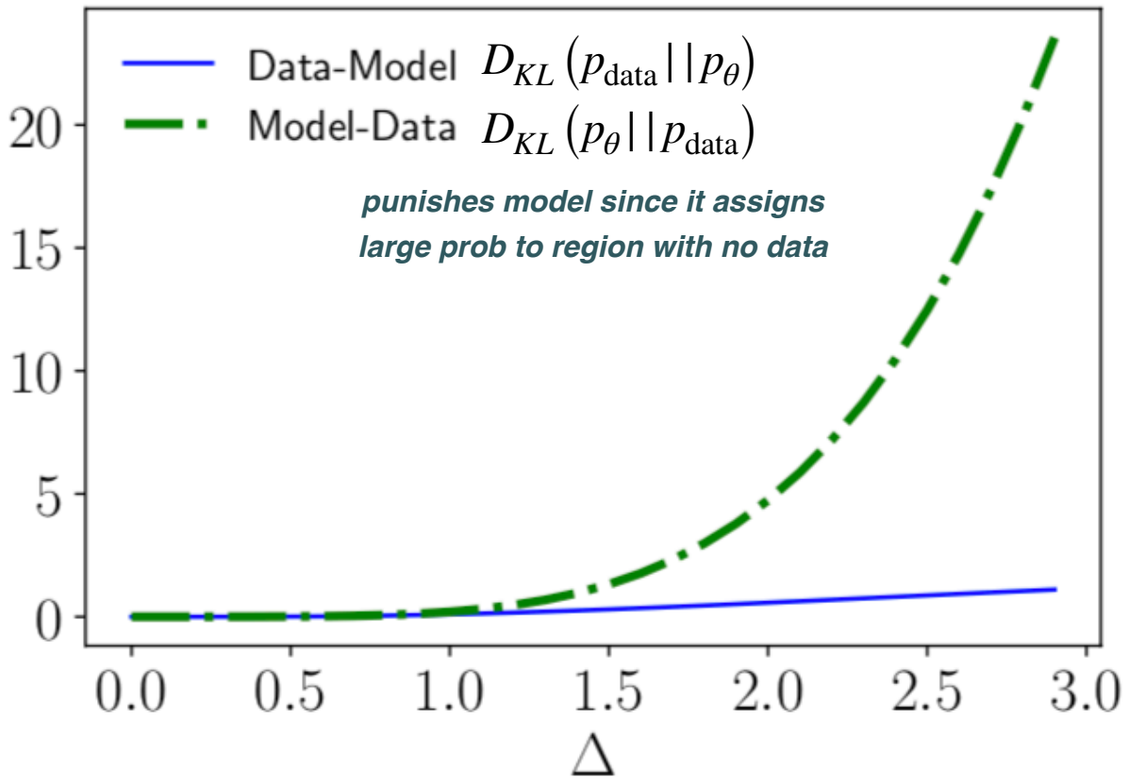
$$D_{KL}\left(p\,||\,q\right) \geq 0$$

in general the integral cannot be computed and one needs to sample the two probability distributions by means of a suitable binning

Statistical distances and similarity

Similarity between probability distributions is a subtle concept!





$D_{KL}\left(p_{\text{data}}||p_\theta\right)$ → misses important information when comparing the data and theory probability distributions

$D_{KL}\left(p_\theta||p_{\text{data}}\right)$ → accounts for non-trivial features beyond the lowest moments of the distribution

identifies when model gives too much wight to regions without data!

Statistical distances and similarity

another popular statistical distance is the **Kolmogorov-Smirnov statistic**

Assume we have *n* observations of the random variable *x*. The **empirical distribution function** is

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{[-\infty,x]}(x_i)$$

indicator function

$$I_{[-\infty,x]}(x_i) = 1\,, \text{for } x_i \leq x\,, \quad I_{[-\infty,x]}(x_i) = 0\,, \text{for } x_i > 0$$

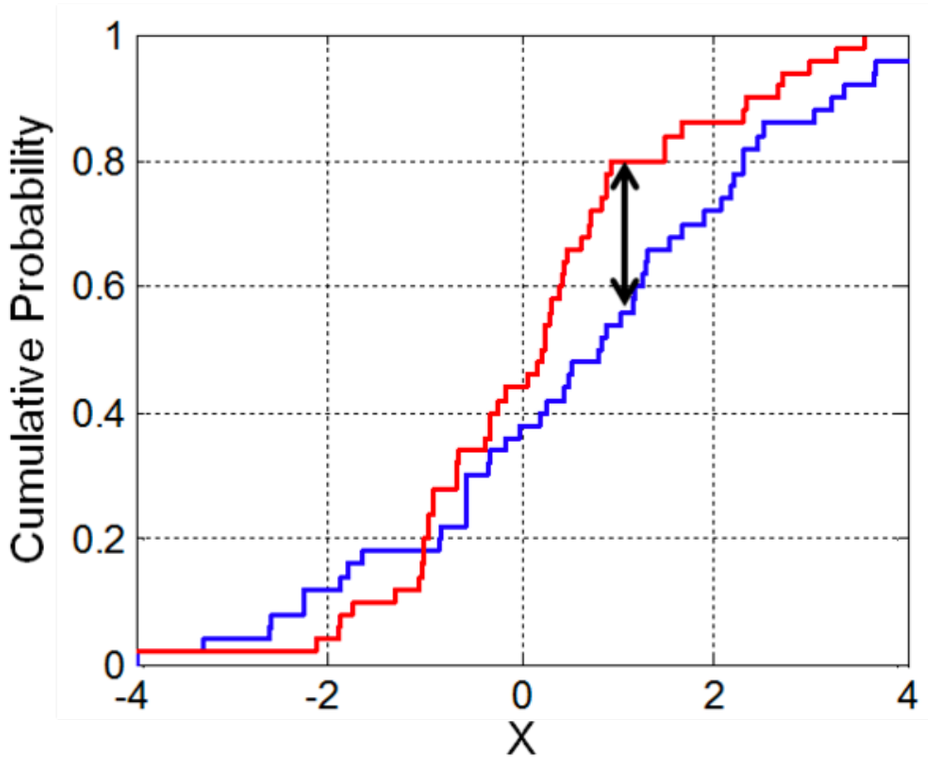
we can then define the **KS-statistic** between two probability distributions as

$$\mathrm{KS} = \sup_x \left[F_{1,n}(x) - F_{2,m}(x)\right]$$

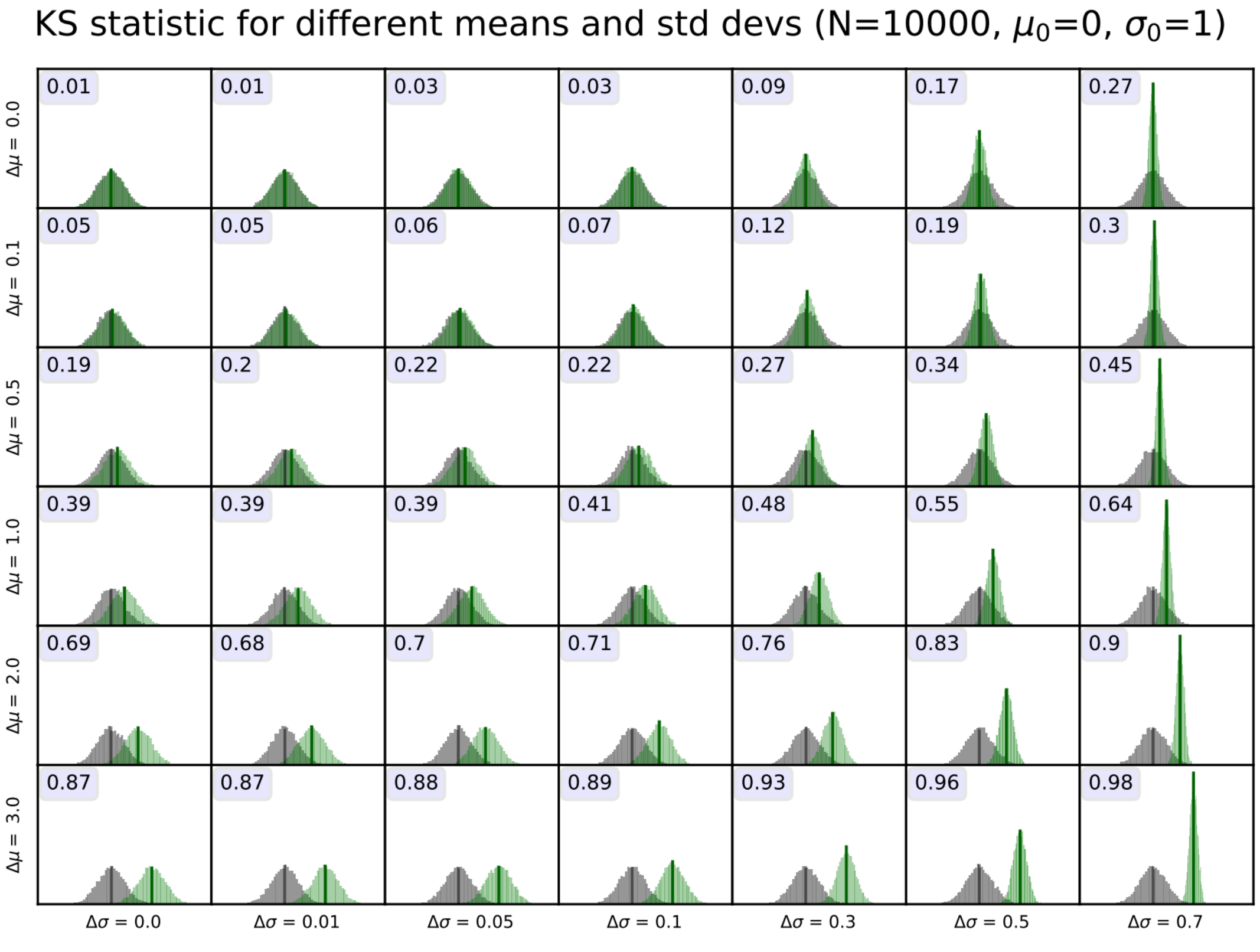
EDF sample 1 (n elements)

EDF sample 2 (m elements)

the KS statistics is sensitive to all possible differences between the two distributions (which might be excessive in some cases)



Statistical distances and similarity



Generative Models

Generative Models

let us go back to our discussion of **Decision Theory**. There we saw that a general classification problem can be separated into two distinct steps:

posterior class probabilities

- The **inference stage,** where a set of input examples is used to train a model for $p(\mathscr{C}_k|\boldsymbol{x})$
- The **decision stage,** where the information on these posterior probabilities is used to make **optimal class assignments**

So far we focused on **discriminative models,** where some criterion (*e.g.* minimise misclassification) is used together with the posterior probabilities to assign each new instance to a class

> In this lecture we will discuss **generative models** which aim to model the **distribution in the space of inputs *x*.** The name stems because using this distribution one can **generate synthetic data points** in the input space

one benefit of this approach is that we access the marginal density in the space of input data, *p(**x**),* which is specially useful to detect new data points that have low probability in the model: **outlier, anomaly, or novelty detection**

Generative Models

generative models can be better understood in the context of **classification**: assume we have a set of *n* instances ***{x}*** with *p* features each that we want to classify into categories C_k

- First of all, one must **solve the inference problem** and determine the **class-conditional probabilities** for each class individually

$$p\left(\boldsymbol{x} \mid \mathscr{C}_k\right)$$ → ***probability that given a class C_k the instance x is found in it***

- Then separately infer the **prior class probabilities** $p\left(\mathscr{C}_k\right)$

- Determine the **posterior class probabilities** by means of Bayes' Theorem

$$p\left(\mathscr{C}_k \mid \boldsymbol{x}\right) = \frac{p\left(\boldsymbol{x} \mid \mathscr{C}_k\right) p(\mathscr{C}_k)}{p(\boldsymbol{x})}$$

→ ***probability that x belongs to class C_k***

$$p(\boldsymbol{x}) = \sum_k p\left(\boldsymbol{x} \mid \mathscr{C}_k\right) p(\mathscr{C}_k)$$ → ***probability to generate the instance x***

Generative Models

$$p\left(\mathscr{C}_k \mid \boldsymbol{x}\right) = \frac{p\left(\boldsymbol{x} \mid \mathscr{C}_k\right) p(\mathscr{C}_k)}{p(\boldsymbol{x})}$$

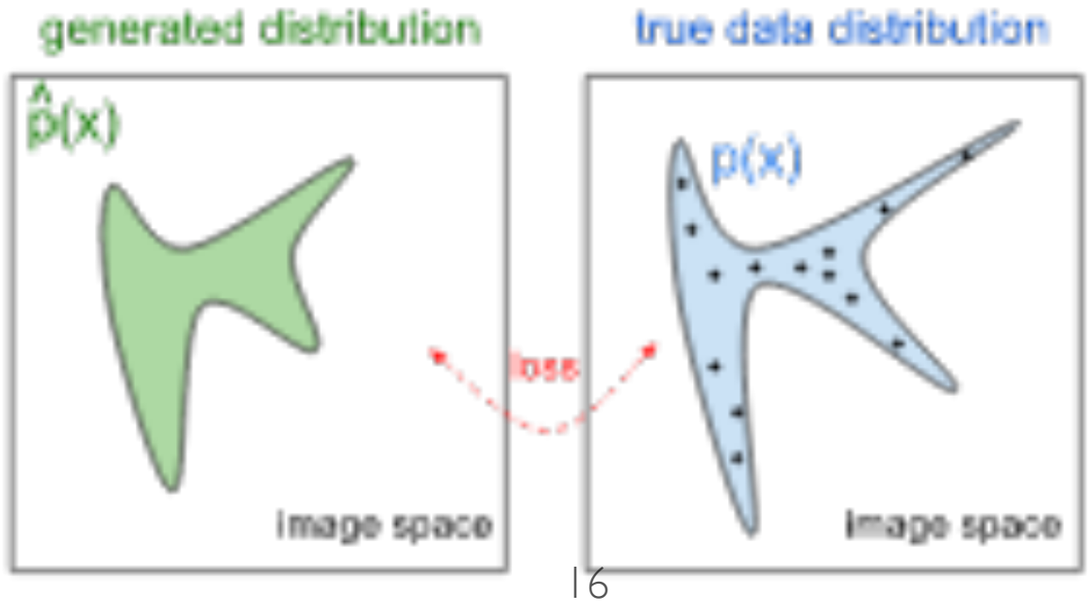
probability that x belongs to class C_k

$$p(\boldsymbol{x}) = \sum_k p\left(\boldsymbol{x} \mid \mathscr{C}_k\right) p(\mathscr{C}_k)$$

probability to generate the instance x

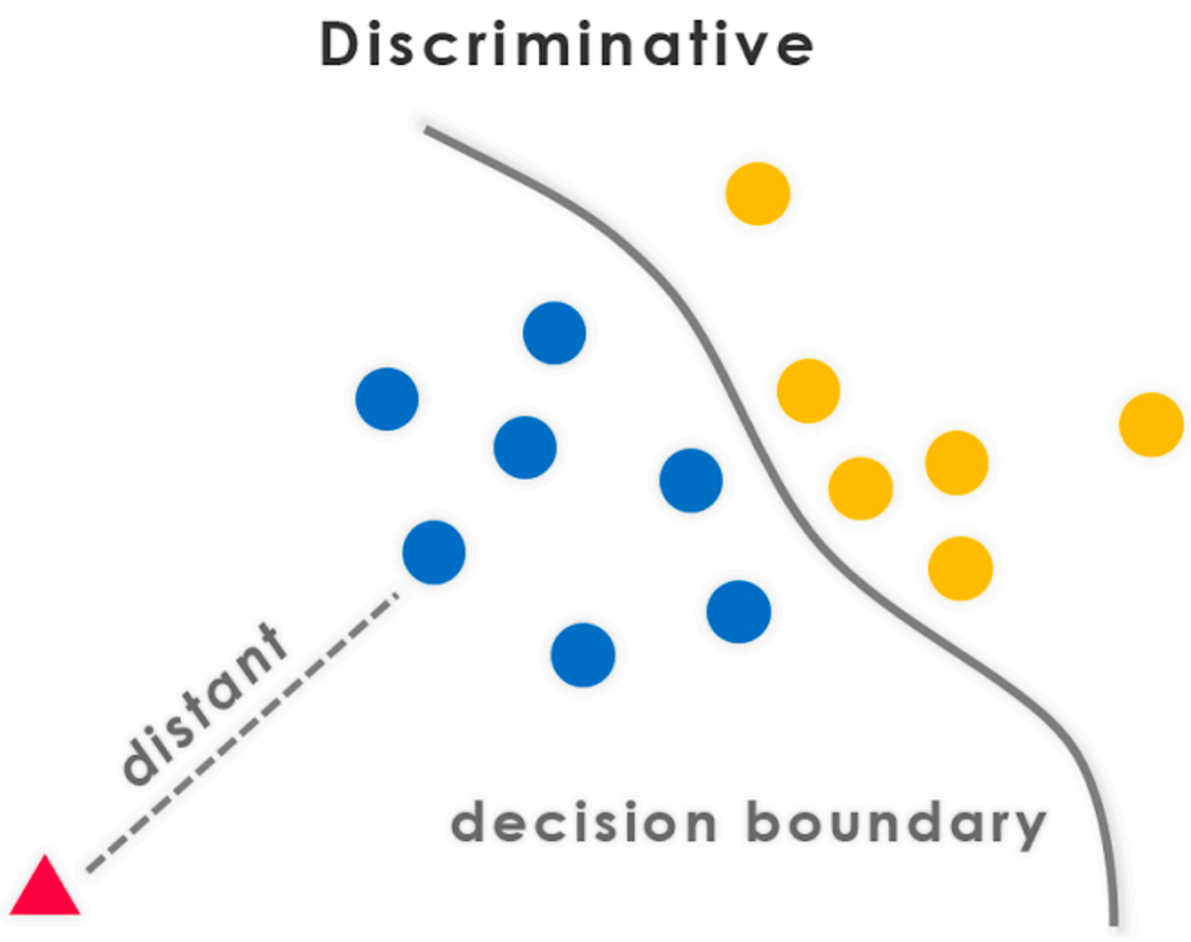
given that we now have the posterior probabilities, we can use **Decision Theory** to determine the class membership for each new instance ***x***

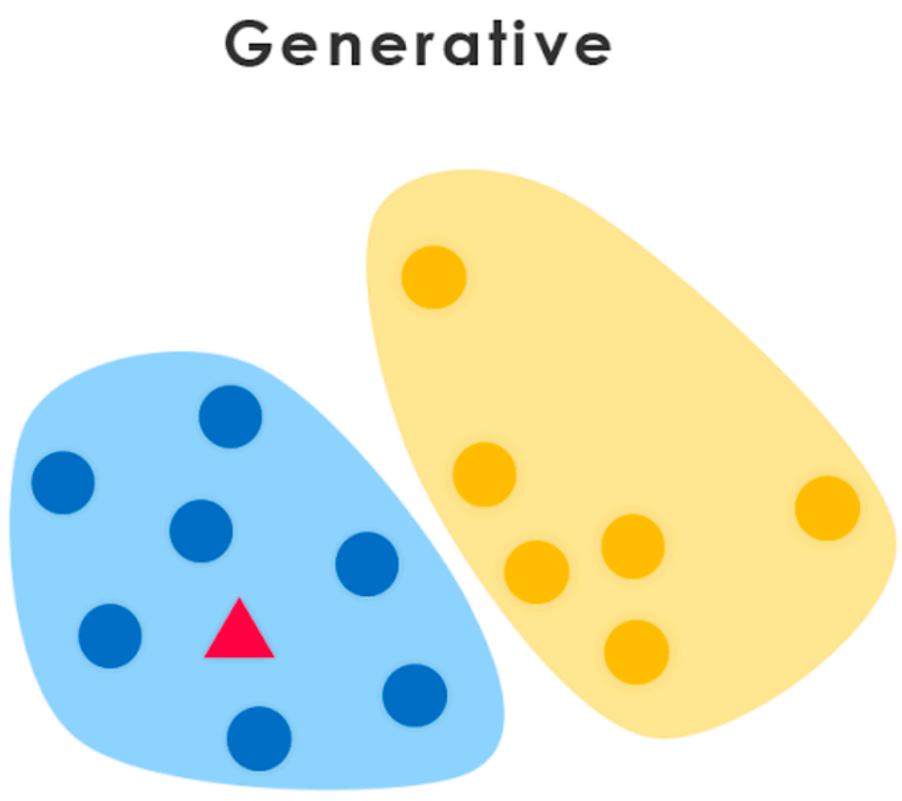
the key aspect of generative models is that we have access to ***p(x),*** the probability distribution in the input data space: sampling from ***p(x)*** one can then **generate new synthetic data points** in input space



Generative Models

Compare graphically Discriminative and Generative Models for classification





discriminative models works best when new instances are far from the decision boundary

in generative models new instances are degenerated in the bulk of the input space probability distribution

Probabilistic generative models

based on these ideas let us construct an explicit **probabilistic generative model** in the context of classification problems with **two categories**. In this case Bayes' Theorem reads

posterior class probabilities

$$p\left(\mathscr{C}_k|\boldsymbol{x}\right) = \frac{p\left(\boldsymbol{x}|\mathscr{C}_k\right)p(\mathscr{C}_k)}{p(\boldsymbol{x})}$$

$$p\left(\mathscr{C}_1|\boldsymbol{x}\right) = \frac{p\left(\boldsymbol{x}|\mathscr{C}_1\right)p(\mathscr{C}_1)}{p\left(\boldsymbol{x}|\mathscr{C}_1\right)p(\mathscr{C}_1) + p\left(\boldsymbol{x}|\mathscr{C}_2\right)p(\mathscr{C}_2)}$$

which can be simplified as

$$p\left(\mathscr{C}_1|\boldsymbol{x}\right) = \frac{1}{1+\exp(-a)} = \sigma(a) \qquad a = \ln\frac{p\left(\boldsymbol{x}|\mathscr{C}_1\right)p(\mathscr{C}_1)}{p\left(\boldsymbol{x}|\mathscr{C}_2\right)p(\mathscr{C}_2)}$$

in terms of the **logistic sigmoid function** which we discussed in the context of neural nets

Probabilistic generative models

Note that once we have trained such a model we can do two things:

- Use **posterior probabilities** with **decision theory** to determine **decision boundaries**

$$p\left(\mathscr{C}_k|\boldsymbol{x}\right) + \text{decision theory} \rightarrow \text{class boundaries}$$

- Use the **probability distribution in the space of input data** to generate synthetic samples

$$p\left(\boldsymbol{x}\right) \rightarrow \text{generate new instances of } \boldsymbol{x}$$

For *K>2* classes the posterior probabilities are given by

$$p\left(\mathscr{C}_k|\boldsymbol{x}\right) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = \ln\left(p(\boldsymbol{x}|\mathscr{C}_k)p(\mathscr{C}_k)\right)$$

normalised exponential of softmax function

(smoothed version of the maximum function)

Generative Models

we can summarise the main ideas **underlying generative models** as follows

Most ML models discussed here (Supervised NNs, logistic regression, ensemble models) are **discriminative:** designed to identify **differences between groups of data**

e.g. cats vs dogs discrimination

these models cannot carry some tasks such as **drawing new examples** from an unknown probability distribution: for this we need **generative models**

e.g. learn how to draw new examples of cat and dog images

e.g. generate new samples of a given phase of the Ising model

generative models are Machine Learning techniques that allows to learn **how to generate new examples** similar to those found in a training dataset

Adversarial Learning and Generative Adversarial Networks

Maximising similarity

In **generative models** one deals with two probability distributions (data and model), which we would like to have as similar as possible

$$p_\theta(\boldsymbol{x}) \qquad p_{\text{data}}(\boldsymbol{x})$$

from model — *from training data*

we aim to construct p_θ to model p_{data}

however subtleties about how we define **similarity** have large implications for the model training

maximising the **log-likelihood of the data under the model** is the same as **minimising the KL divergence** between the data distribution and the model distribution

Kullback-Leibler divergence

$$D_{KL}\left(p_{\text{data}}||p_\theta\right) = \int d\boldsymbol{x}\, p_{\text{data}}(\boldsymbol{x}) \log \frac{p_{\text{data}}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}$$

$$= \int d\boldsymbol{x}\, p_{\text{data}}(\boldsymbol{x}) \log p_{\text{data}}(\boldsymbol{x}) - \int d\boldsymbol{x}\, p_{\text{data}}(\boldsymbol{x}) \log p_\theta(\boldsymbol{x})$$

$$= S_p[p_{\text{data}}] - \left\langle \log p_\theta \right\rangle_{\text{data}}$$

Entropy

Expected value of model probability given the data

Maximising similarity

maximising the **log-likelihood of the data under the model** is the same as **minimising the KL divergence** between the data distribution and the model distribution

$$\left\langle \log p_\theta(\boldsymbol{x}) \right\rangle_{\text{data}} = S_p[p_{\text{data}}] - D_{KL}\left(p_{\text{data}} \,||\, p_\theta\right)$$

Log-likelihood of data under model

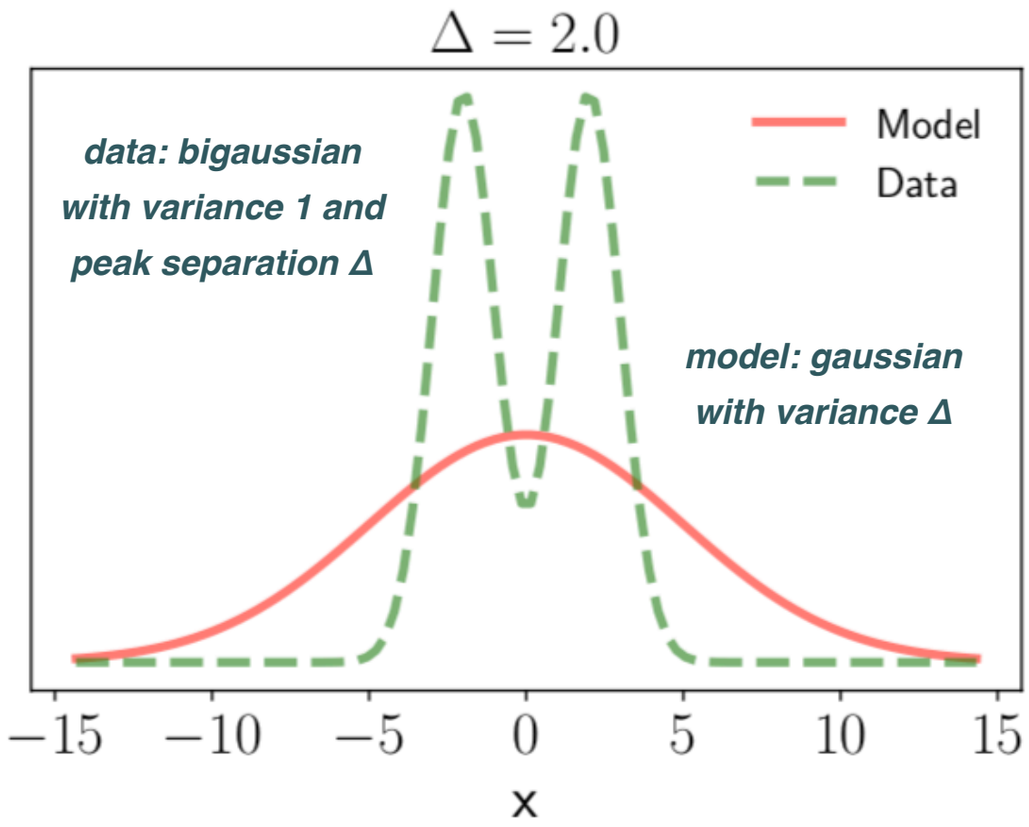
entropy of data: independent of model parameters

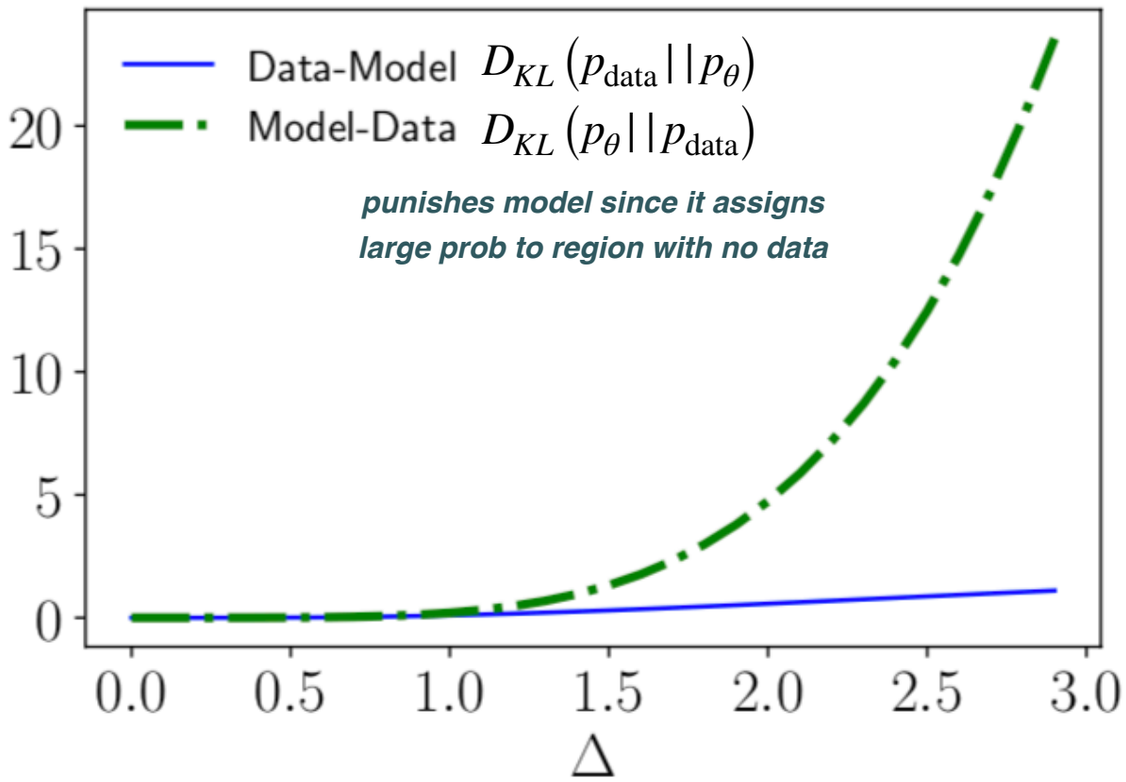
KL-divergence

by minimising the KL divergence between two distributions, we can build a model that reproduces the **input data distribution**

Statistical distances and similarity

Similarity between probability distributions is a subtle concept!





$D_{KL}\left(p_{\text{data}}||p_\theta\right)$ → misses important information when comparing the data and theory probability distributions

$D_{KL}\left(p_\theta||p_{\text{data}}\right)$ → accounts for non-trivial features beyond the lowest moments of the distribution

identifies when model gives too much wight to regions without data!

Adversarial Learning

A subtle point concerns which of the two versions of the KL-divergence to minimise:

(1) $D_{KL}\left(p_{\text{data}}||p_\theta\right)$ → ***(draw values of θ accordingly to p_θ and compare with data)*** → ***Calculable using sampling***

(2) $D_{KL}\left(p_\theta||p_{\text{data}}\right)$ → ***Large when model over-weights low-density regions near real peaks***

→ ***but not calculable since p_{data} unknown ….***

In **Adversarial Learning** we achieve a similar goal as that of minimising ***(2)*** by training a **discriminator** to distinguish between real data points and samples from the model

By punishing the model for generating points that can be easily discriminated from the data, Adversarial Learning decreases the **weight of regions in the model space that are far away from data points,** regions that inevitably arise when maximising the likelihood

Generative adversarial networks

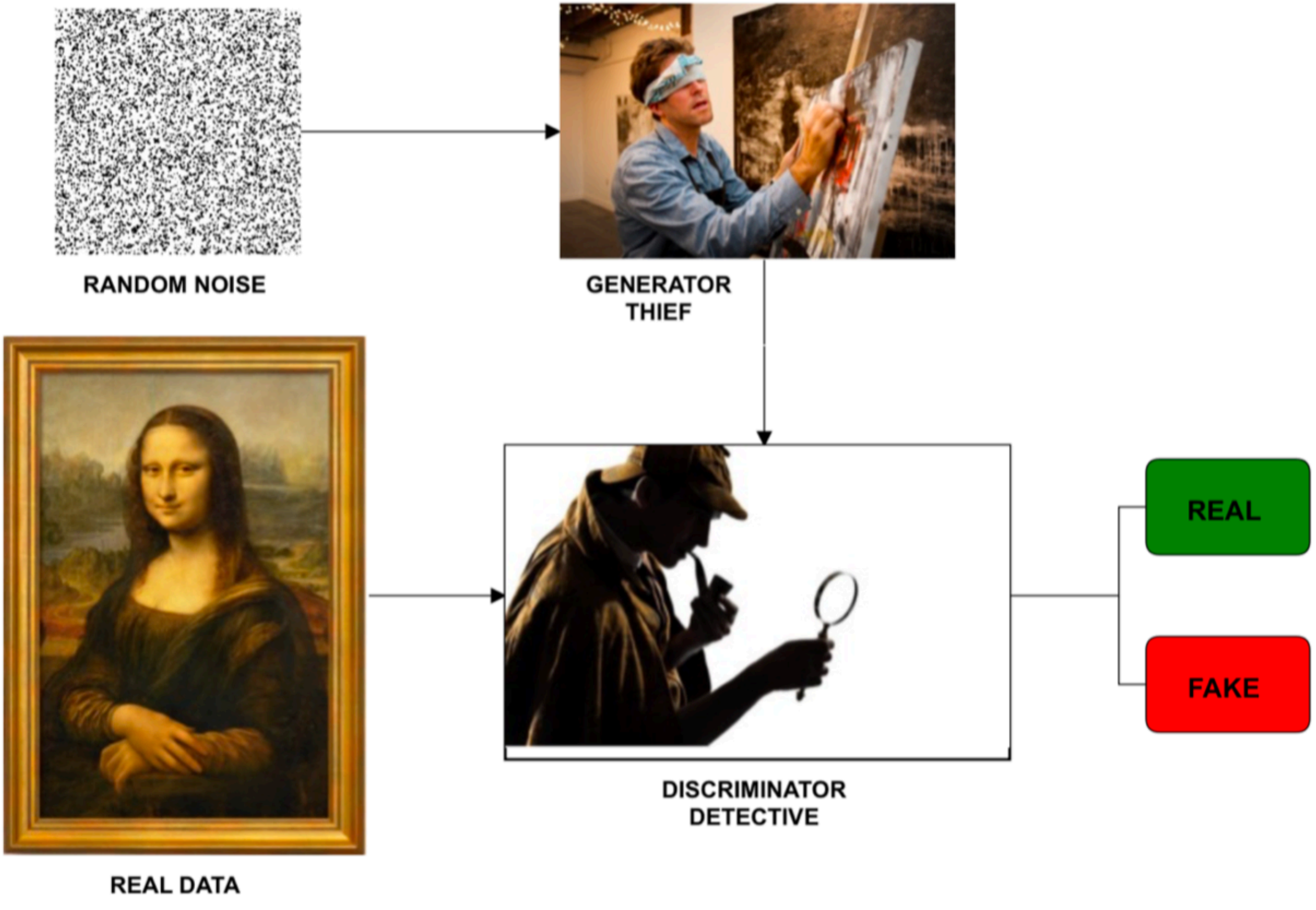
Generative Adversarial Networks (GANs) are deep neural network architectures, composed by two independent NNs which **compete against each other**

- ***(1)*** A **generator G** NN that creates (samples) pseudo-data by inferring the probability distribution associated to the training dataset

- ***(2)*** A **discriminator D** NN which determines the probability of a given sample arises from the actual training data rather than having been produced by **G**

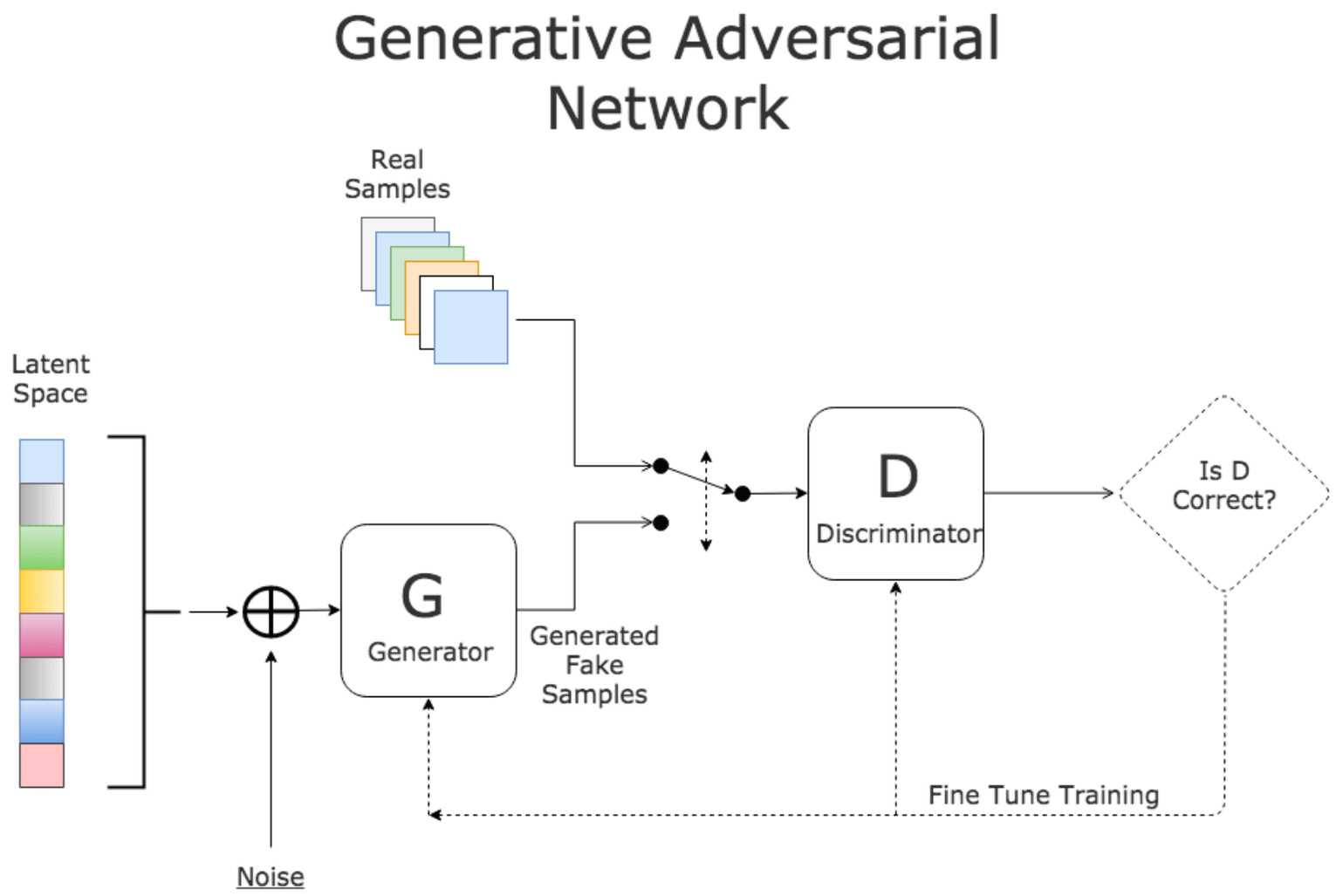
the generator network **G** should be trained to maximise the probability that the discriminator network **D** makes a mistake: that is, **G** should generate pseudo-data samples that are virtually **indistinguishable** from the actual data

Generative adversarial networks



Generative Adversarial Networks

- Architecture for an **unsupervised neural network training** (unlabelled samples)
- Based on two **independent nets** that work separately and act as **adversaries:**
  - the **Discriminator (D)** undergoes training and plays the role of classifier
  - the **Generator (G)** and is tasked to generate random samples that **resemble real samples** with a twist rendering them as fake samples.



Generative Adversarial Networks

GAN training

As with other NN architectures one uses GD to train GANs, but now one has to **update sequentially** the model parameters of both **G** and **D**

- Take a sample of *N* data points from the training set

$$\{\boldsymbol{x}_n\}_{n=1}^{N} \qquad \boldsymbol{x}_n = (x_{n,1}, x_{n,2}, \ldots, x_{n,p})$$

p = number of features per sample

- Produce a sample of *N* pseudo-data points from generator **G** (at ite$_0$ this is random noise)

$$\{\boldsymbol{z}_n\}_{n=1}^{N} \qquad \boldsymbol{z}_n = (z_{n,1}, z_{n,2}, \ldots, z_{n,p})$$

- Evaluate the cost function: since we are dealing with binary classification (true/false) the appropriate cost function is the **cross-entropy**

$$C\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\log D(\boldsymbol{x}_i) + \log(1 - D(G(\boldsymbol{z}_i)))\right)$$

NN params of D ***NN params of G*** ***output of D when input a real data sample*** ***output of D when input a ``fake'' data sample produced by G***

- Train **D** using GD to maximise its discrimination capability

GAN training

- Evaluate the cost function: since we are dealing with binary classification (true/false) the appropriate cost function is the **cross-entropy**

$$C\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\log D(\boldsymbol{x}_i) + \log(1 - D(G(\boldsymbol{z}_i)))\right)$$

- Train **D** using GD to maximise its discrimination capability

$$\mathbf{v}_t = \eta_t \nabla_{\theta_D} C(\boldsymbol{\theta}_{D,t}, \boldsymbol{\theta}_{G,t}), \qquad \boldsymbol{\theta}_{D,t+1} = \boldsymbol{\theta}_{D,t} - \mathbf{v}_t$$

- At this point **D** can tell apart data from pseudo-data pretty well, so we need to train **G** to generate better (closer to the training set) pseudo-data samples
- Produce a sample of *N* pseudo-data points from the generator **G** $\{\boldsymbol{z}_n\}_{n=1}^{N}$

$$C\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right) = \frac{1}{N}\sum_{n=1}^{N}\log(1 - D(G(\boldsymbol{z}_i)))$$

output of D (now with its parameters fixed)

$$\mathbf{v}_t = \eta_t \nabla_{\theta_G} C(\boldsymbol{\theta}_{D,t}, \boldsymbol{\theta}_G), \qquad \boldsymbol{\theta}_{G,t+1} = \boldsymbol{\theta}_{G,t} - \mathbf{v}_t$$

GAN training

the generator and discriminator are sequentially trained and iterated until convergence is achieved, at this point **D** cannot tell apart the pseudo-data from **G** from the real data

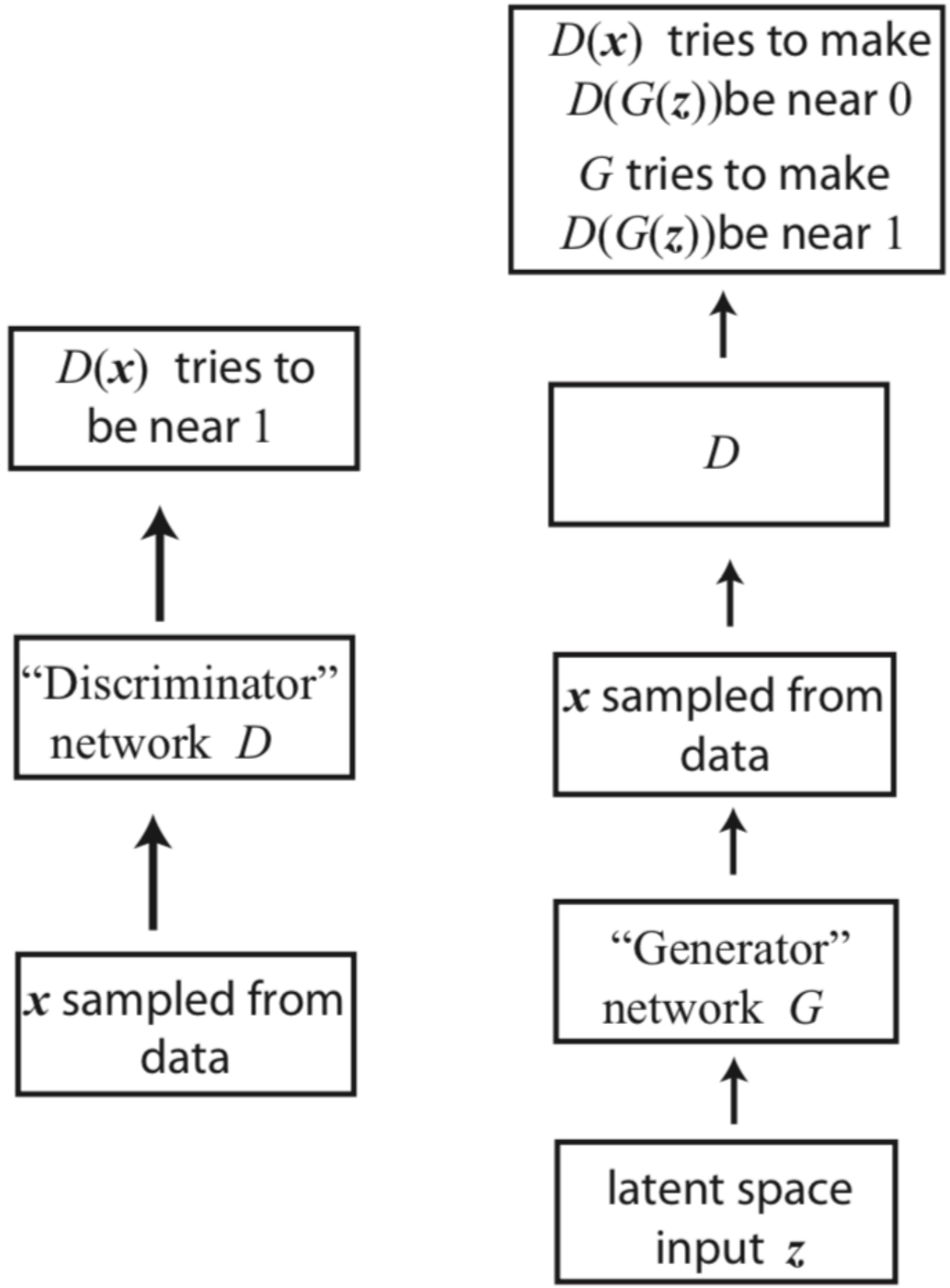


Image generation with GANs

https://thispersondoesnotexist.com/

Image generation with GANs

https://thispersondoesnotexist.com/

Generative Adversarial Networks

Energy-based Models

Maximum Entropy Generative Models

basic concept is the **Shannon information-theoretic entropy**, which quantifies the statistical uncertainty one has about a random variable drawn from a probability distribution

$$S_p = \mathrm{Tr}_{\boldsymbol{x}} p(\boldsymbol{x}) \log p(\boldsymbol{x})$$

sum/integral over all possible values of variable

variables whose prob dist I would like to estimate

assume we have a **set of models**, functions of ***x,*** whose average should coincide with some observed values. What should be their **underlying prob dist**?

$$\{f_i(\boldsymbol{x})\} \qquad \langle f_i \rangle_{\mathrm{obs}} \qquad i = 1, \ldots, n_{\mathrm{dat}}$$

model for sample i as function of inputs x

models ***observations***

Principle of Maximum Entropy: choose the probability distribution with the largest uncertainty (Shannon entropy) subject to the observational constraints

$$\langle f_i \rangle_{\mathrm{model}} = \int d\mathbf{x} f_i(\mathbf{x}) p(\mathbf{x}) = \langle f_i \rangle_{\mathrm{obs}}$$

the selected distribution is the one that makes admits the most ignorance beyond the stated data

Maximum Entropy Generative Models

this condition can be expressed as a **Lagrange Multiplier problem** by minimising:

$$\mathscr{L}[p] = -S_p + \sum_i \lambda_i \left(\langle f_i \rangle_{\text{obs}} - \int d\boldsymbol{x}\, f_i(\boldsymbol{x}) p(\boldsymbol{x}) \right) + \gamma \left(1 - \int d\boldsymbol{x} p(\boldsymbol{x}) \right)$$

Shannon entropy ***observational constraints*** ***normalisation***

whose solution gives us the **Maximum Entropy distribution**

$$p(\boldsymbol{x}) = \frac{\exp\left(\sum_i \lambda_i f_i(\boldsymbol{x}) \right)}{\int d\boldsymbol{x}\ \exp\left(\sum_i \lambda_i f_i(\boldsymbol{x}) \right)} = \frac{\exp\left(\sum_i \lambda_i f_i(\boldsymbol{x}) \right)}{Z}$$

partition function

$$E/k_B T = -\sum_i \lambda_i f_i(\boldsymbol{x})$$

which is nothing but the **Boltzmann distribution in statistical mechanics,** and where the parameters of the distribution are fixed by the observations

$$\partial_{\lambda_i} \log Z = \langle f_i \rangle_{\text{data}}$$

Energy-based Generative Models

these MaxEnt models can be used to **infer the underlying probability distributions** from a finite set of observations, which subsequently can be used to **generate new instances**

training an energy-based generative model: using the data to infer the model parameters

$$E\left(\boldsymbol{x};\boldsymbol{\theta}\right) = -\sum_i \theta_i f_i(\boldsymbol{x})$$

model parameters

as in Supervised Learning, we need to specify a **cost function,** which however in the case of generative models is much subtler: what defines a good model?

the most useful method is to **maximise the log-likelihood** of the training set

$$\hat{\boldsymbol{\theta}} = \arg\,\min_\theta \mathscr{L}(\{\boldsymbol{\theta}\})$$

$$\mathscr{L}(\{\boldsymbol{\theta}\}) = \left\langle \log p_\theta(\boldsymbol{x}) \right\rangle_{\text{data}} = -\left\langle E(\boldsymbol{x};\boldsymbol{\theta}) \right\rangle_{\text{data}} - \log Z(\{\boldsymbol{\theta}\})$$

where we have used that the **generative probability distribution** is of the Boltzmann form and that the partition function does not depend on the data

Boltzmann machines

The training of **energy-based generative models** proceeds usually via SGD

MaxEnt generative models are defined by the choice of the **energy**

model parameters

$$p_\theta(\boldsymbol{x}) = \frac{\exp\left(\sum_i \theta_i f_i(\boldsymbol{x})\right)}{\int d\boldsymbol{x}\, \exp\left(\sum_i \theta_i f_i(\boldsymbol{x})\right)} = \frac{\exp\left(\sum_i \theta_i f_i(\boldsymbol{x})\right)}{Z} = \frac{\exp\left(-E(\boldsymbol{x};\boldsymbol{\theta})\right)}{Z}$$

one can construct various **other generative models** with different choices of the energy

e.g. Restricted Boltzmann machines

$$E(\boldsymbol{x};\boldsymbol{\theta}) = -\sum_i a_i(v_i) - \sum_\mu b_\mu(h_\mu) - \sum_{i\mu} W_{i\mu} v_i h_\mu$$