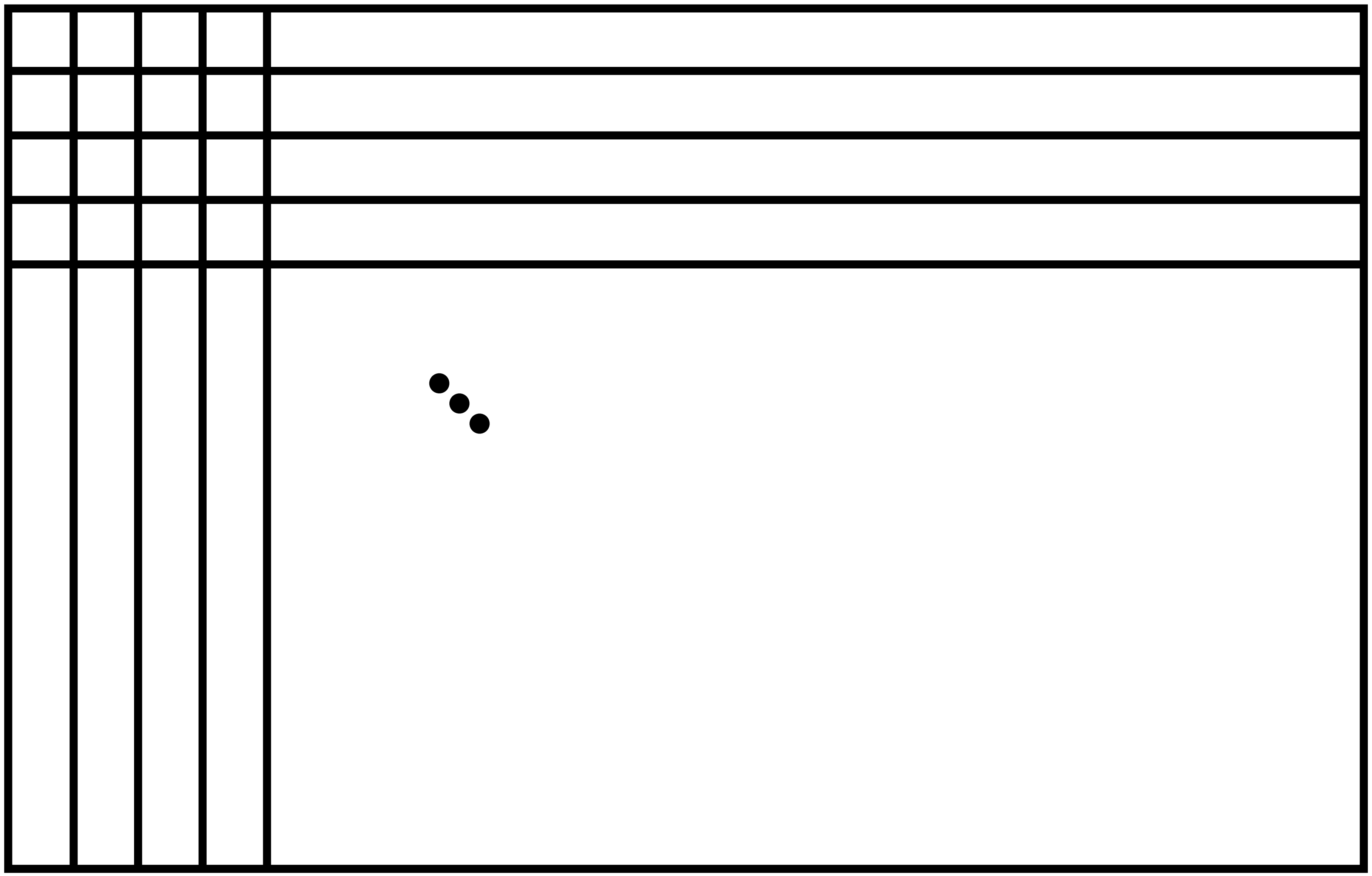
DEEP LEARNING

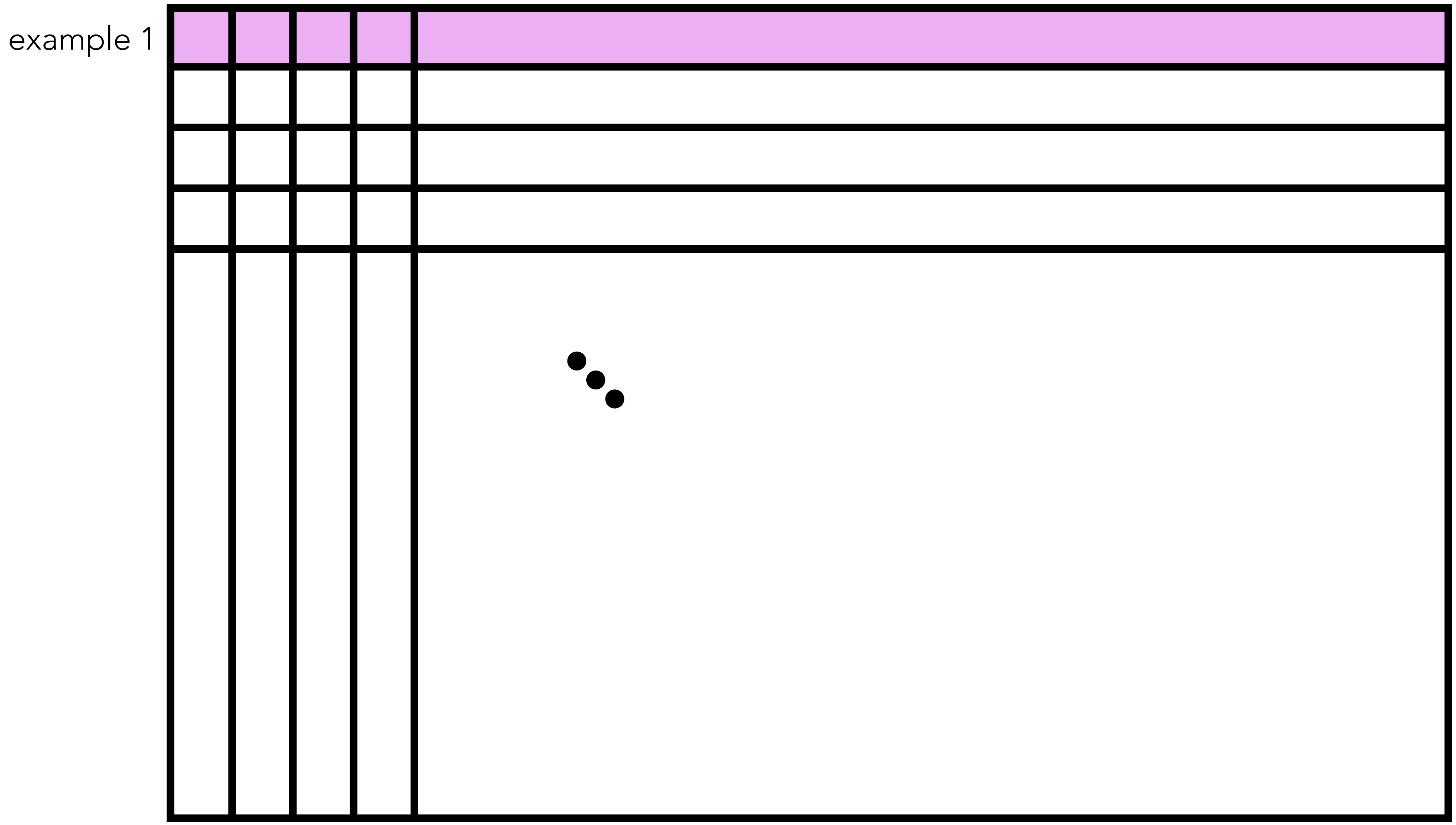
PART THREE - *DEEP GENERATIVE MODELS*

GENERATIVE MODELS

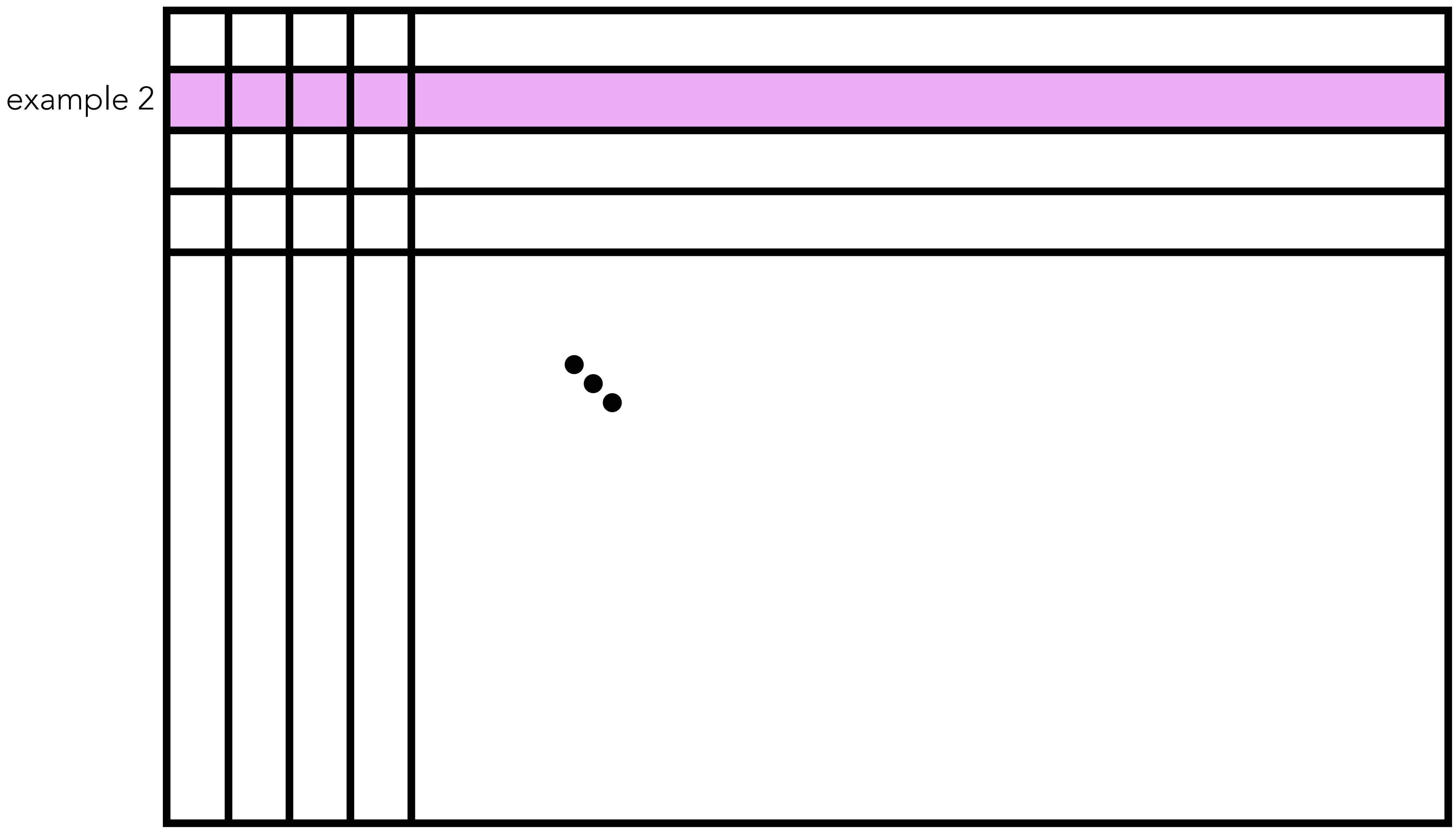
DATA



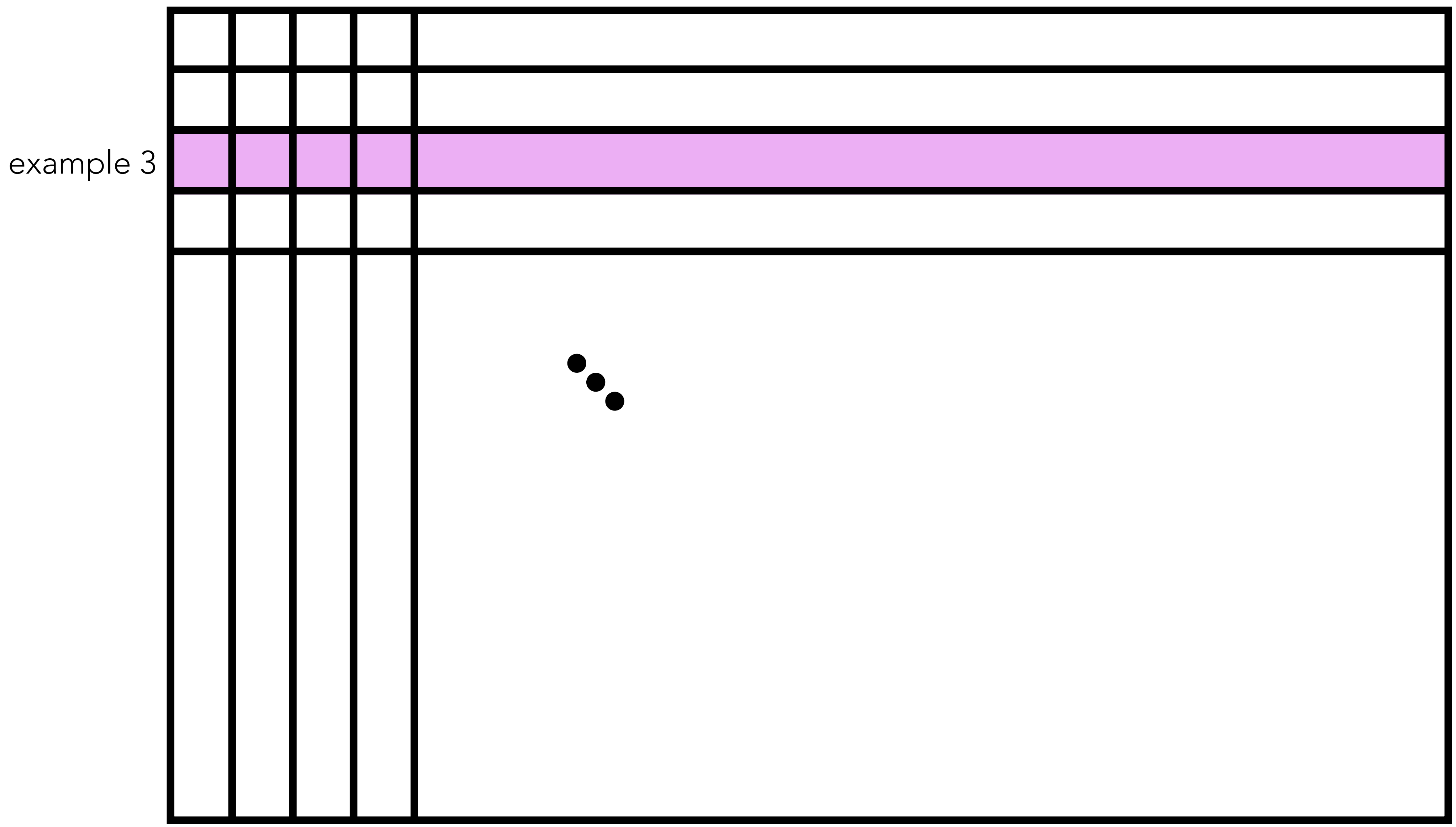
DATA



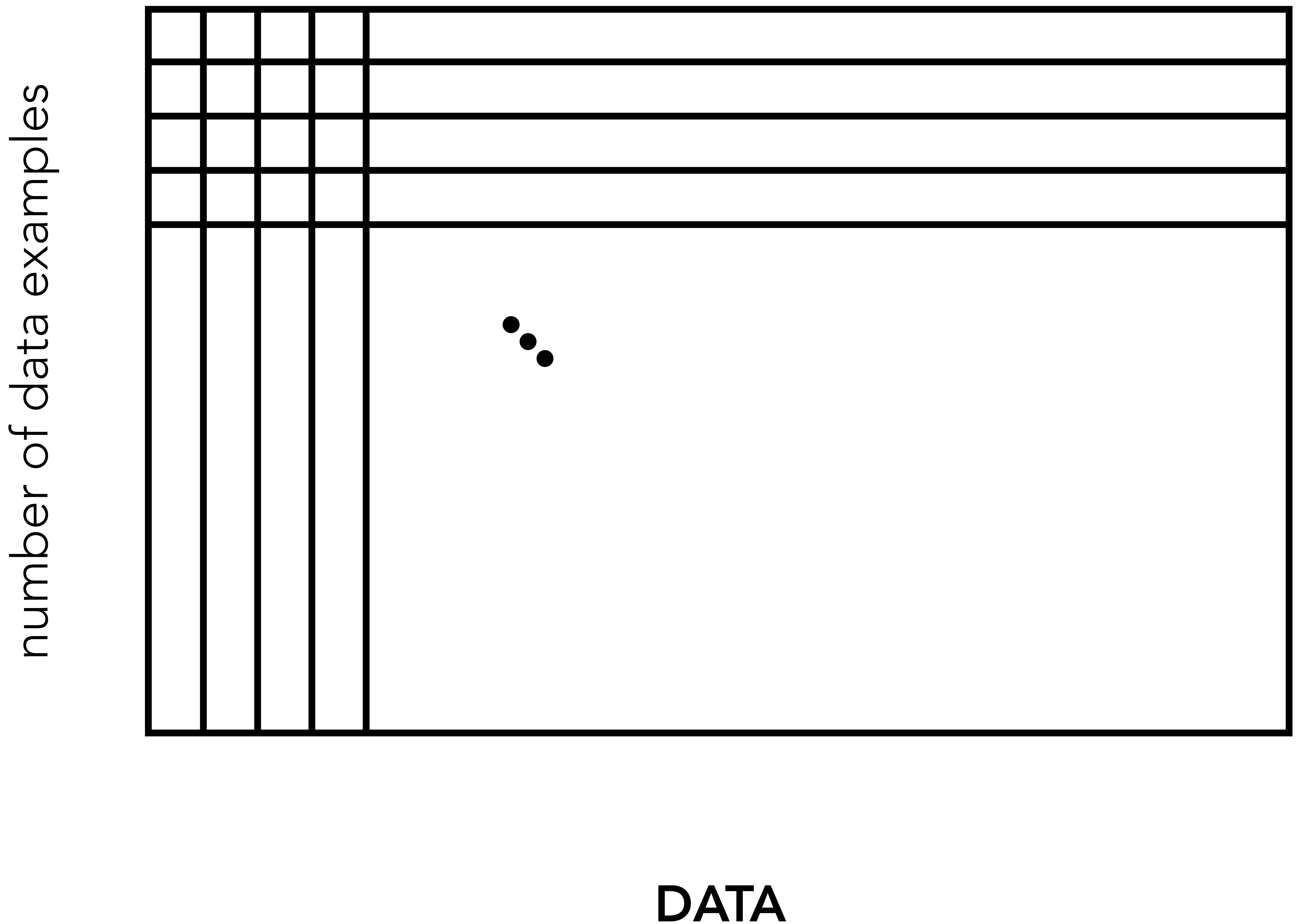
DATA

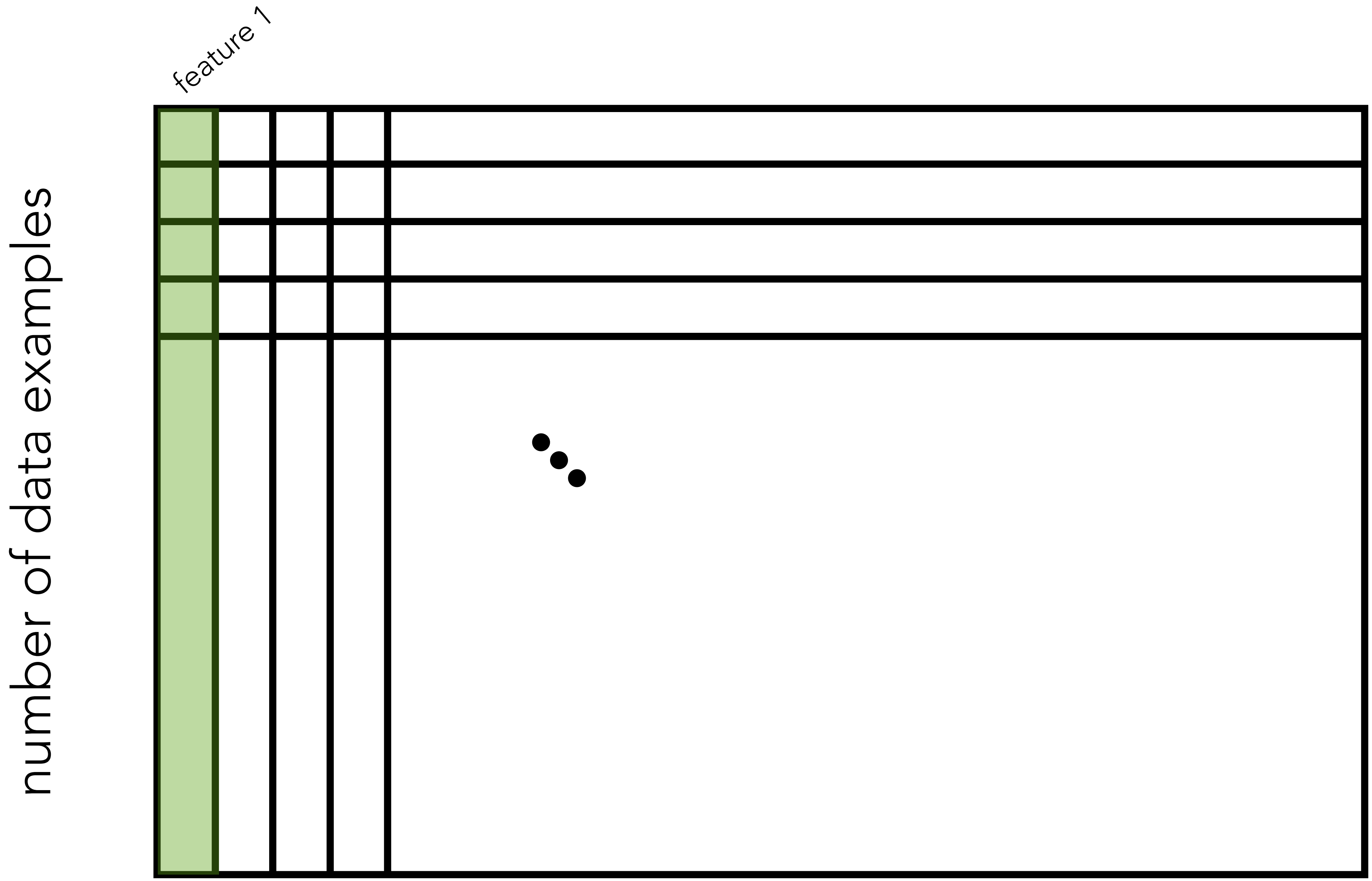


DATA

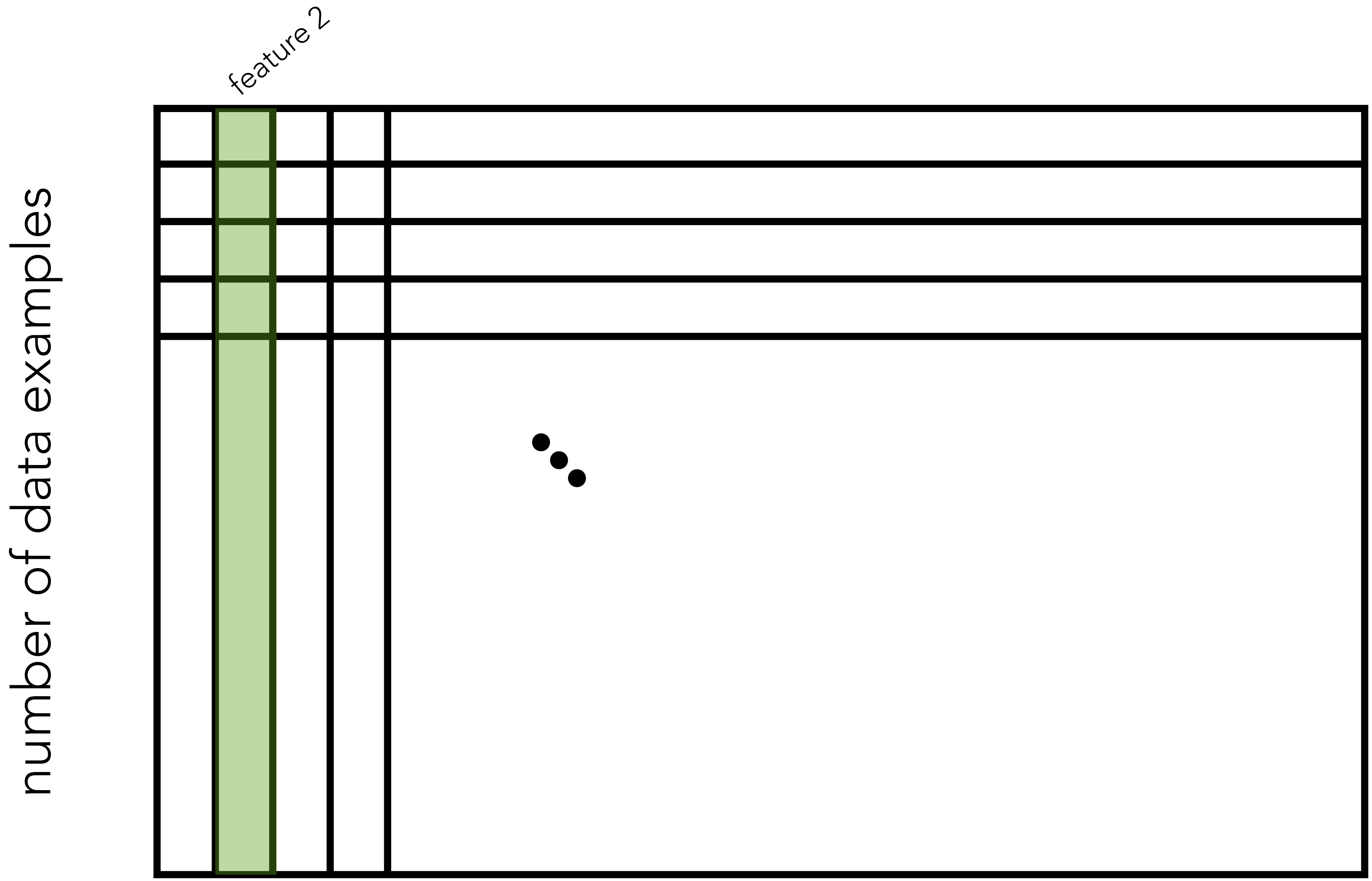


DATA

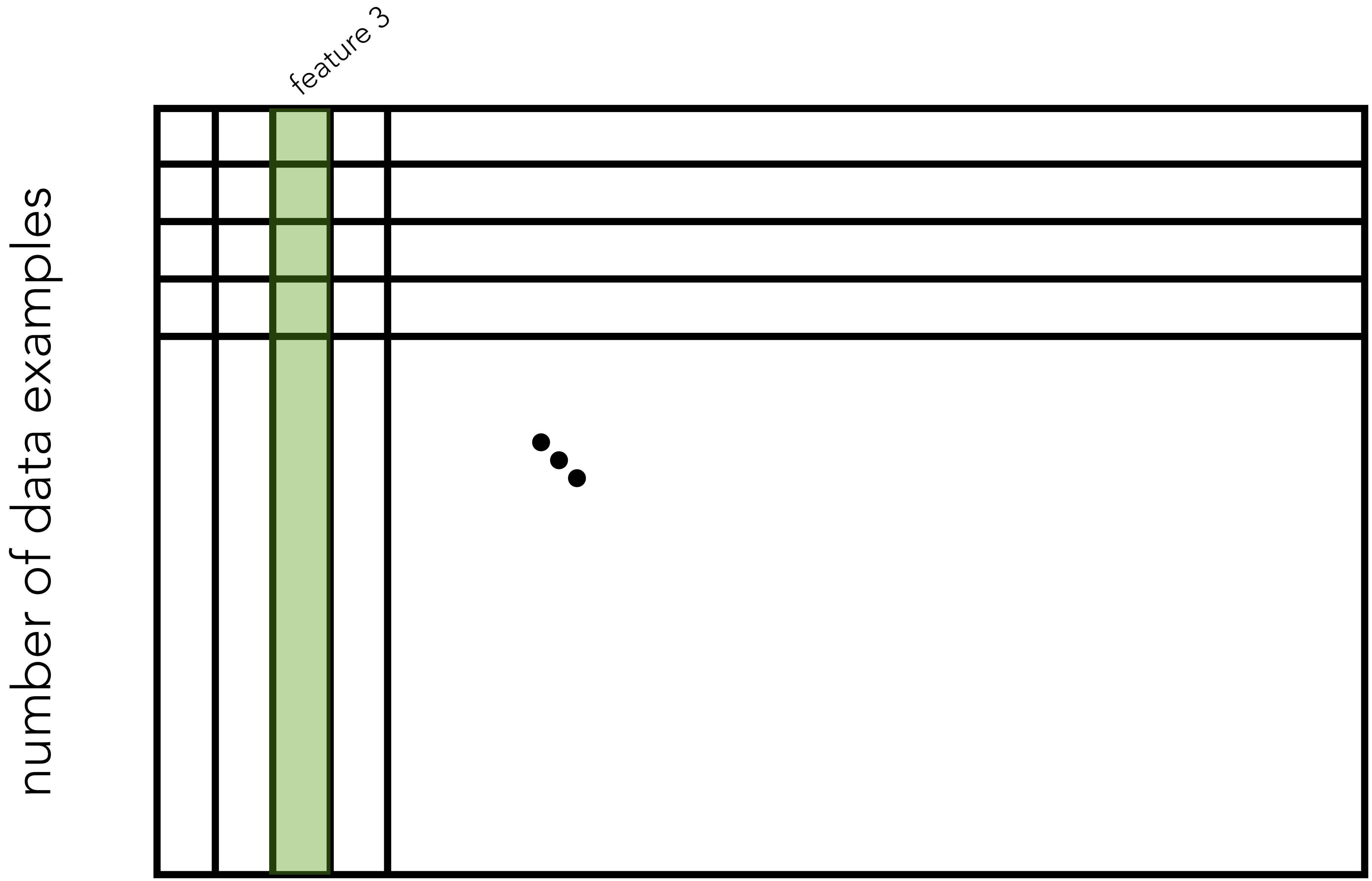

number of data examples
DATA



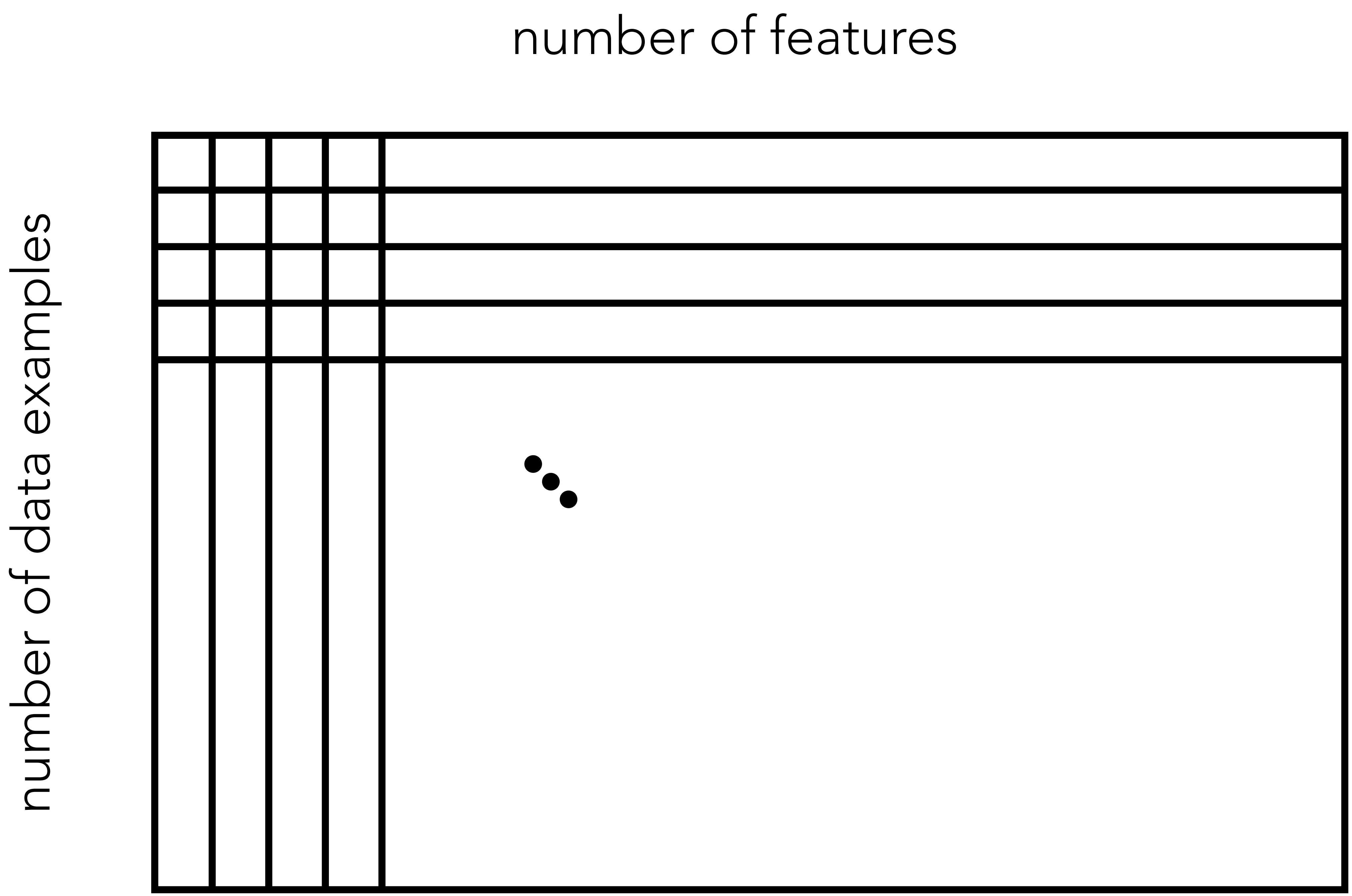
DATA



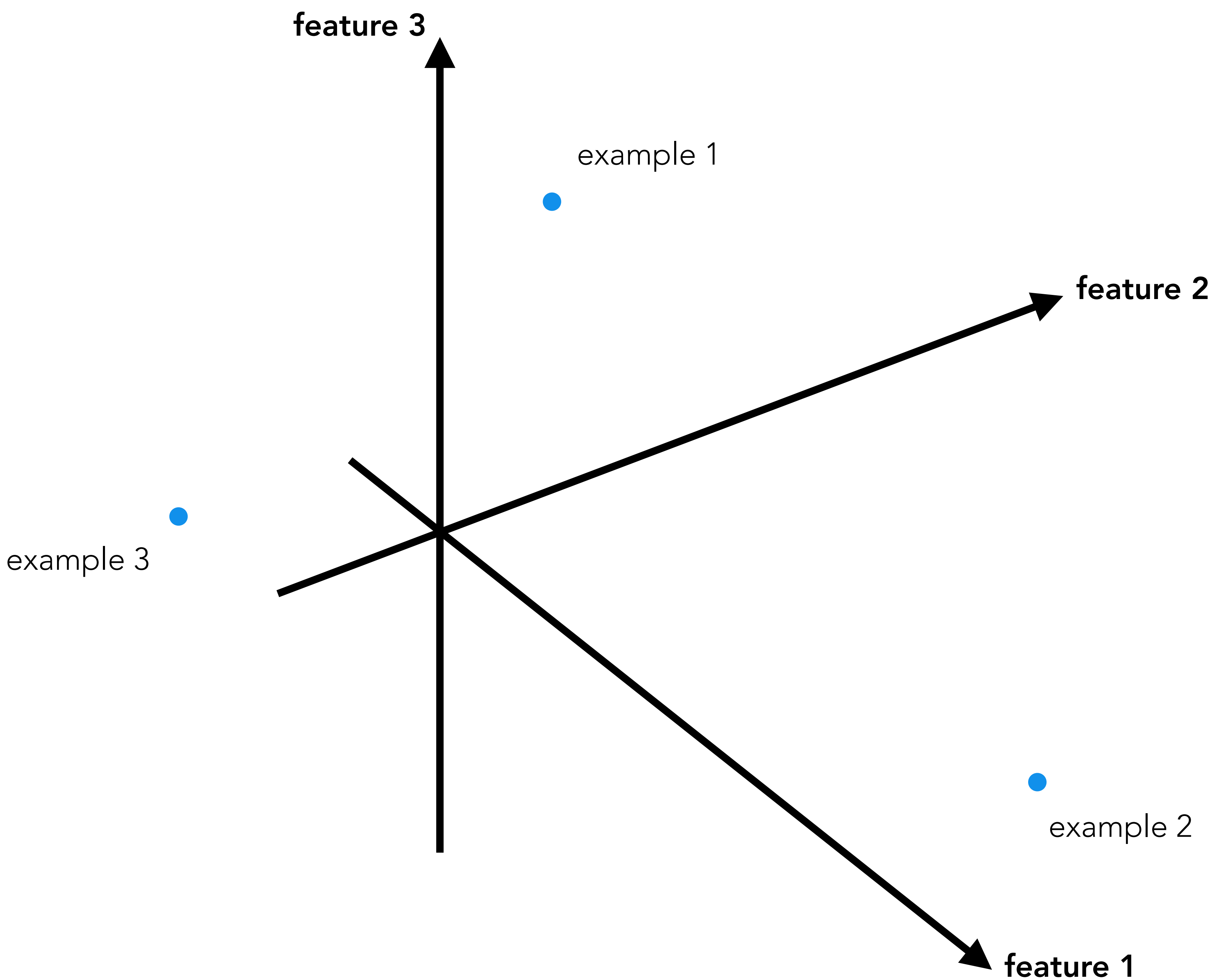
DATA



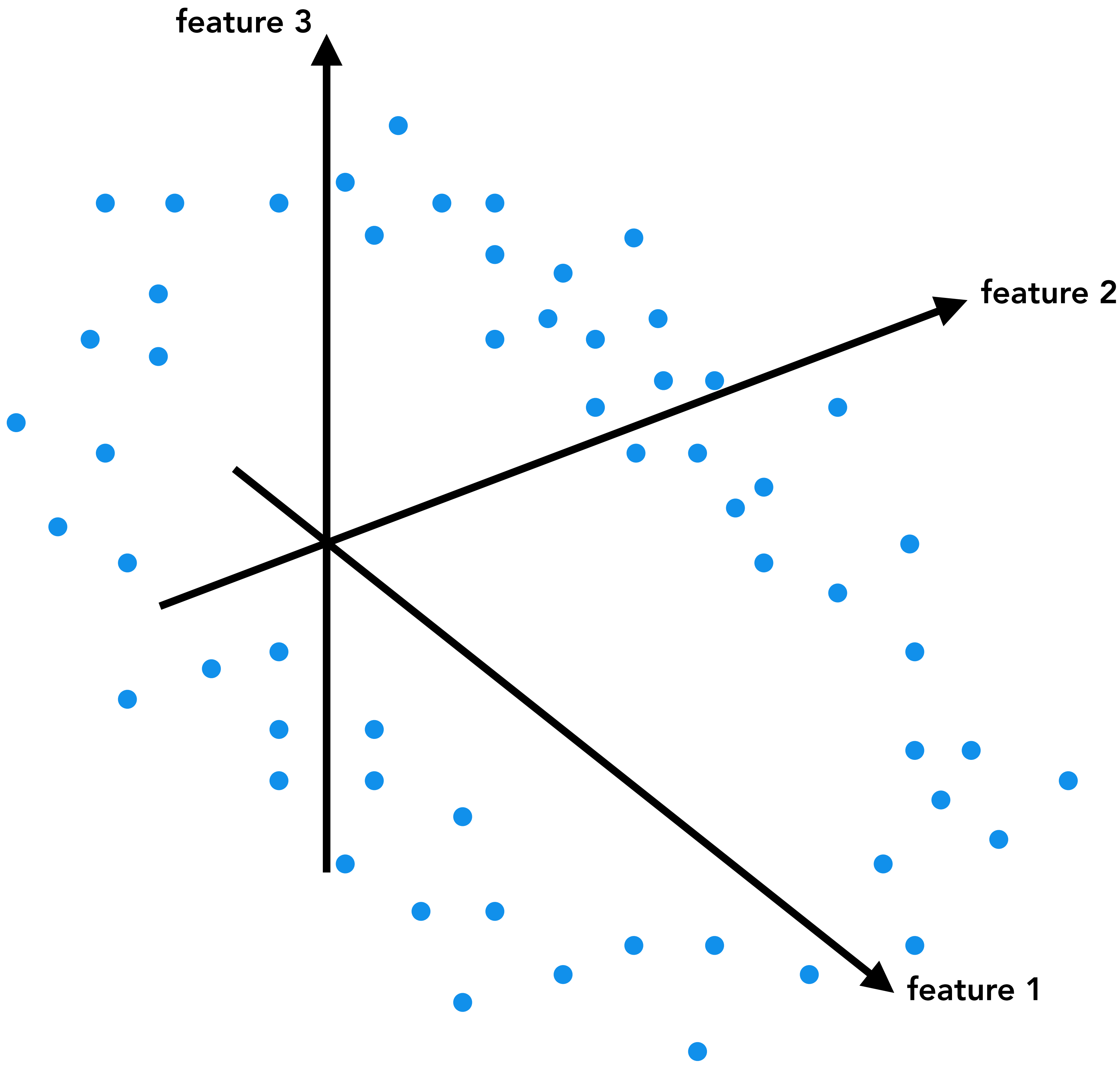
DATA



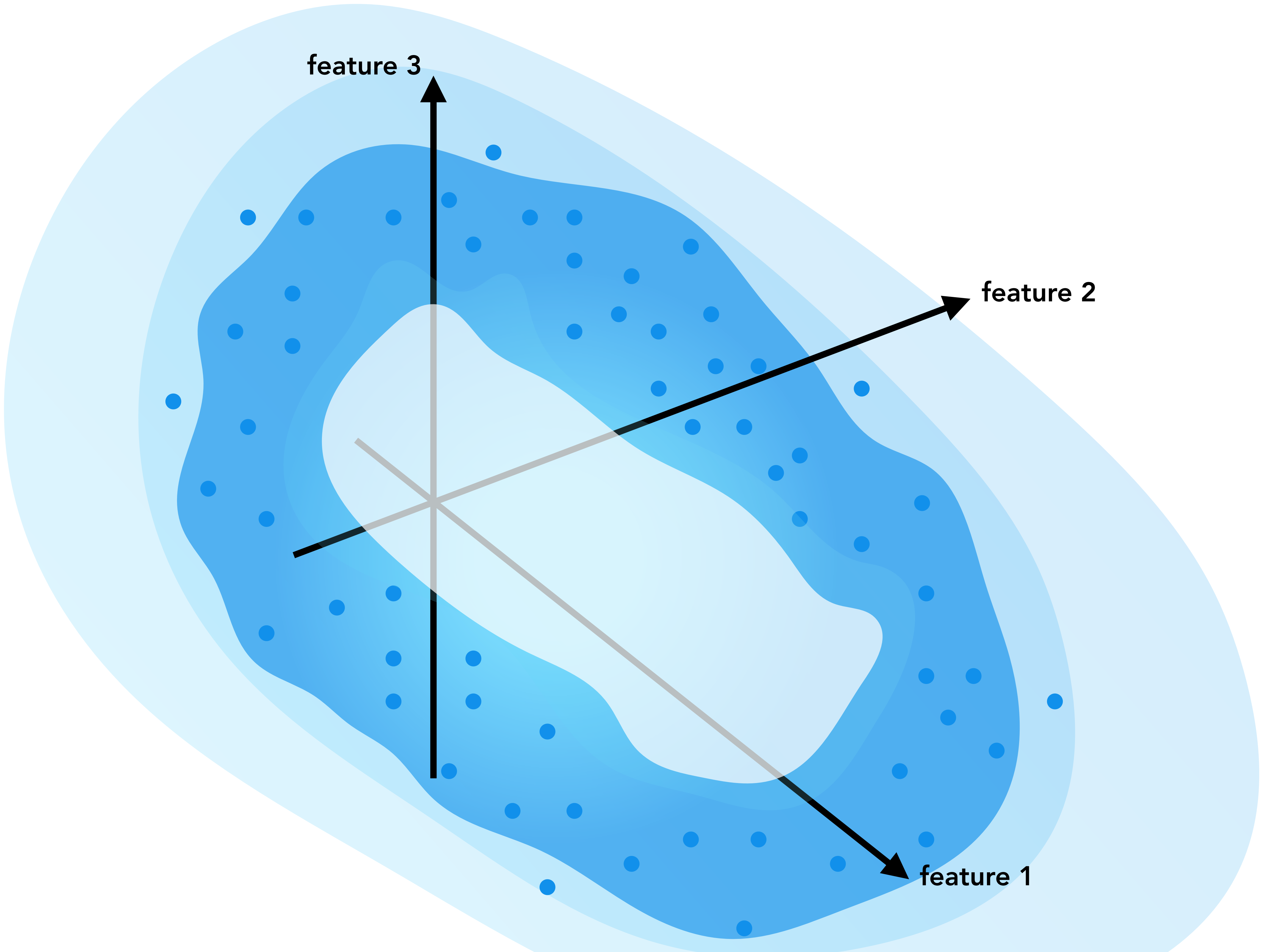
DATA



DATA DISTRIBUTION



DATA DISTRIBUTION



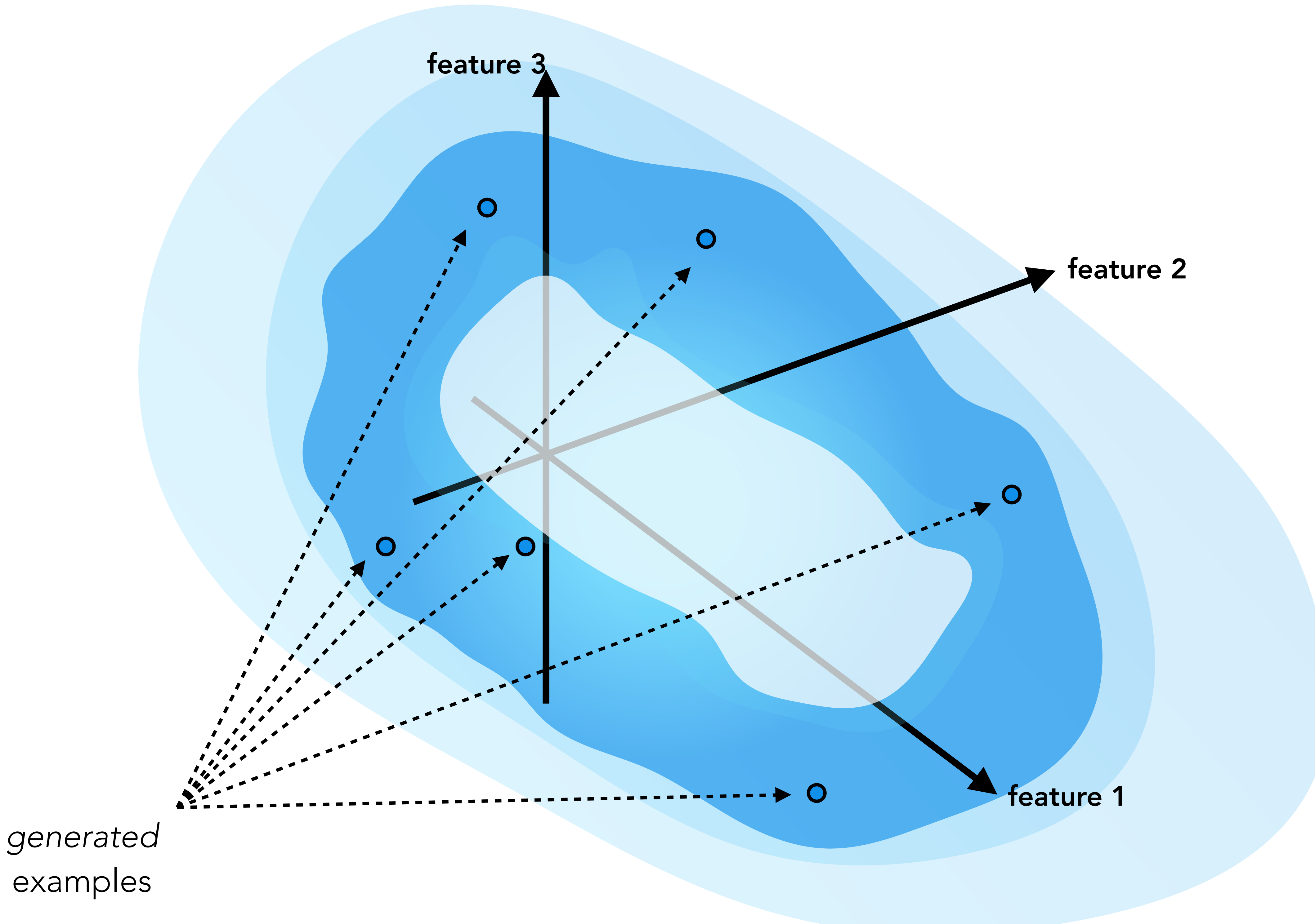
DENSITY ESTIMATION

estimating the density of the empirically observed data distribution

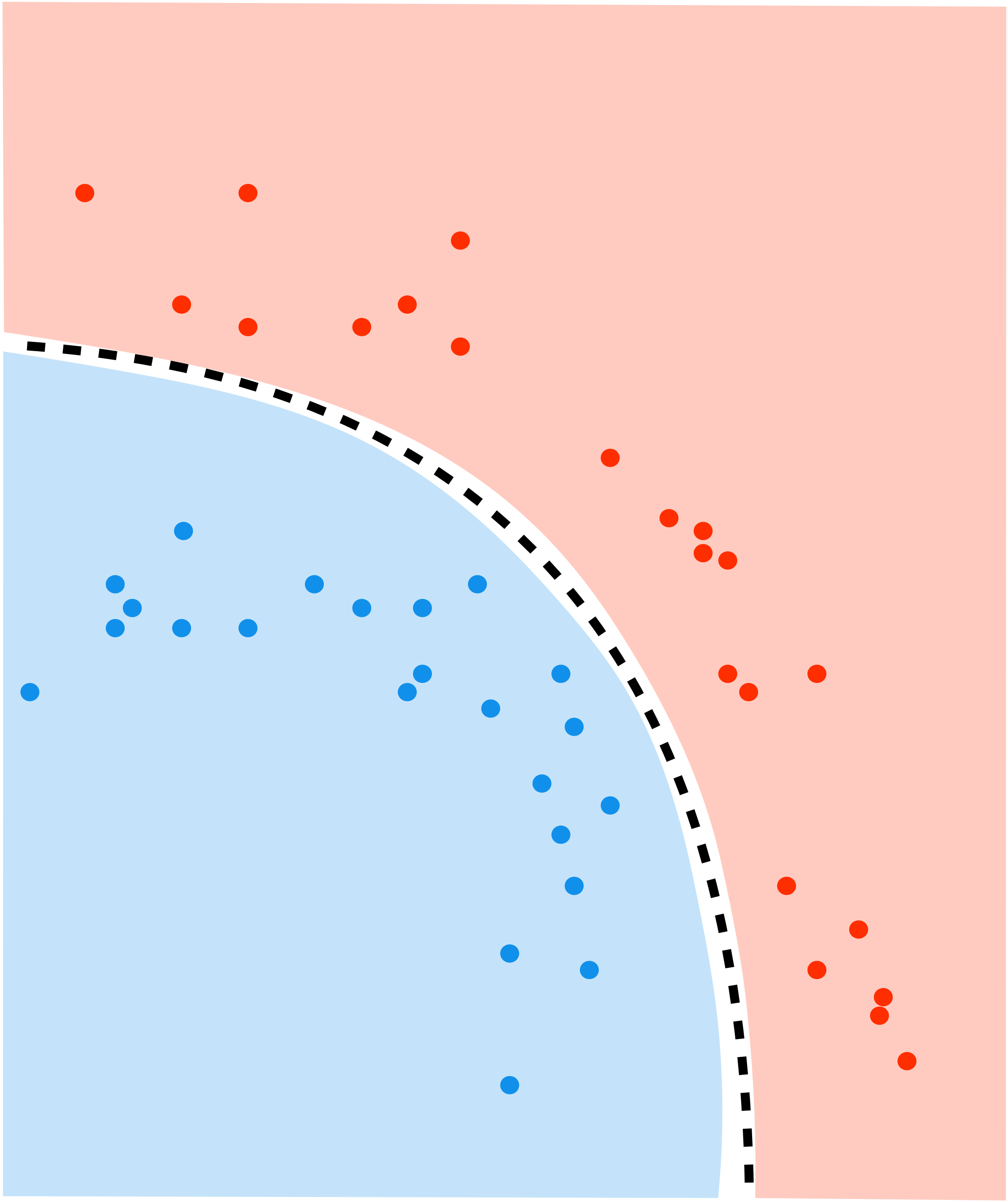
GENERATIVE MODEL

a model of the density of the data distribution

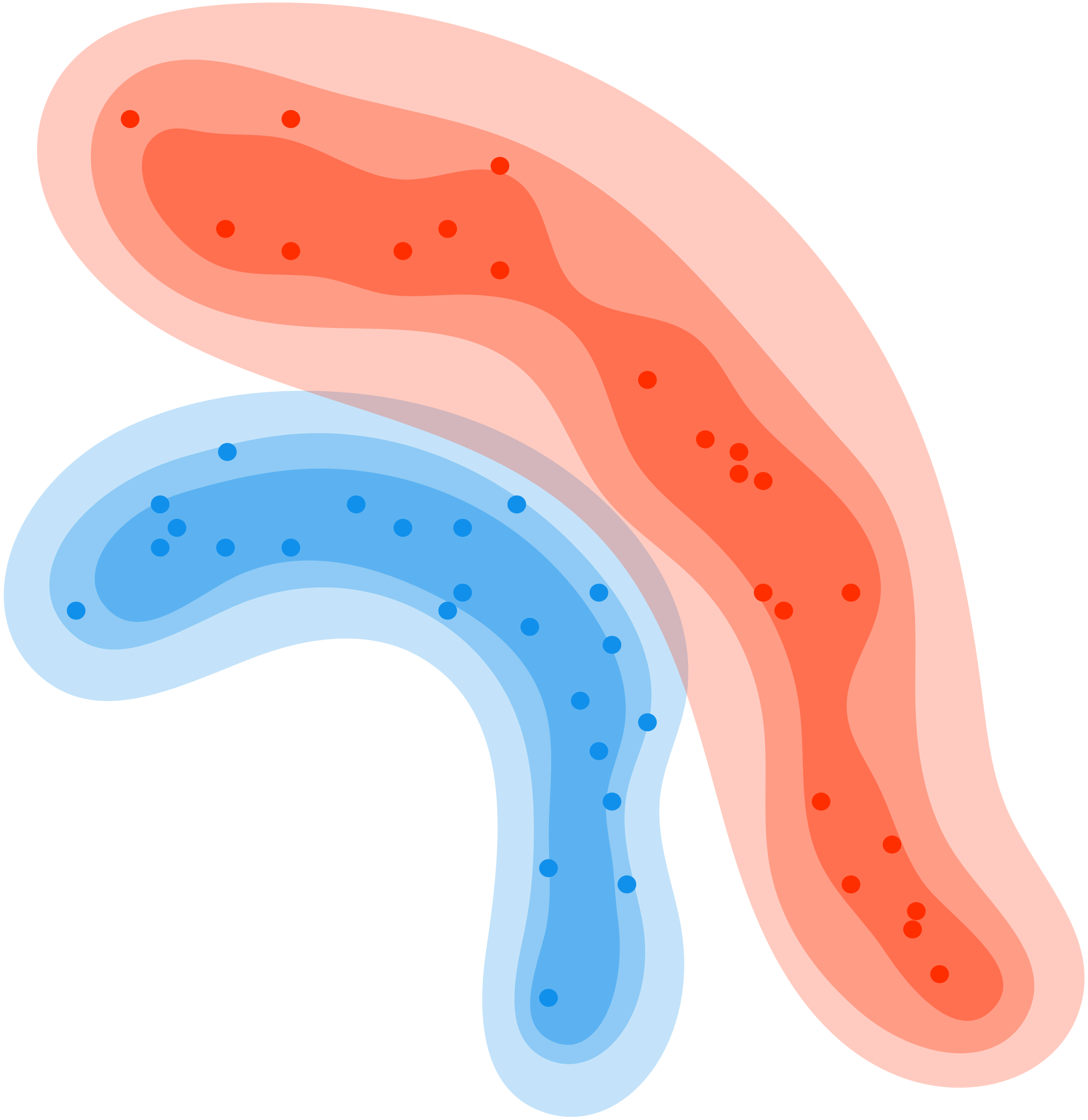
by modeling the data distribution,
*generative models are able to **generate** new data examples*



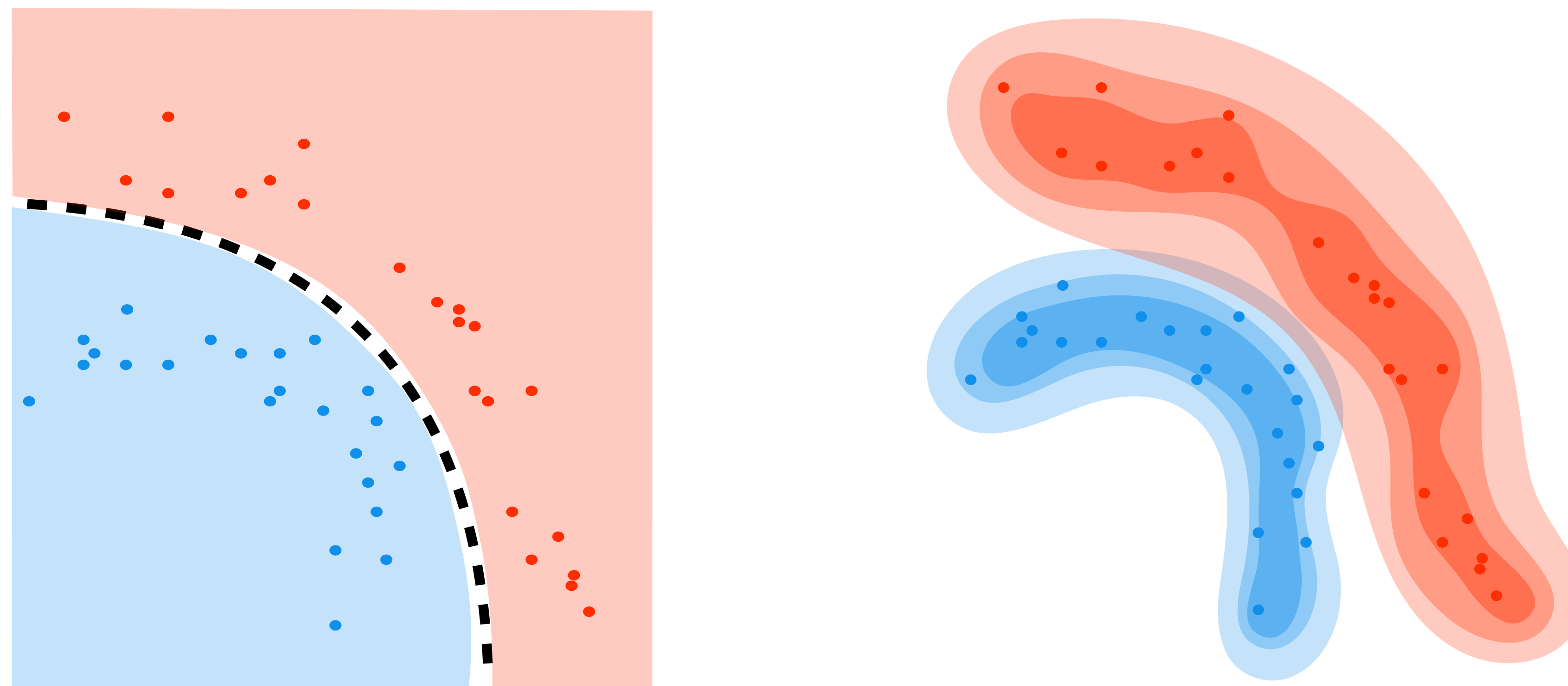
discriminative model



generative model



discriminative models vs. generative models



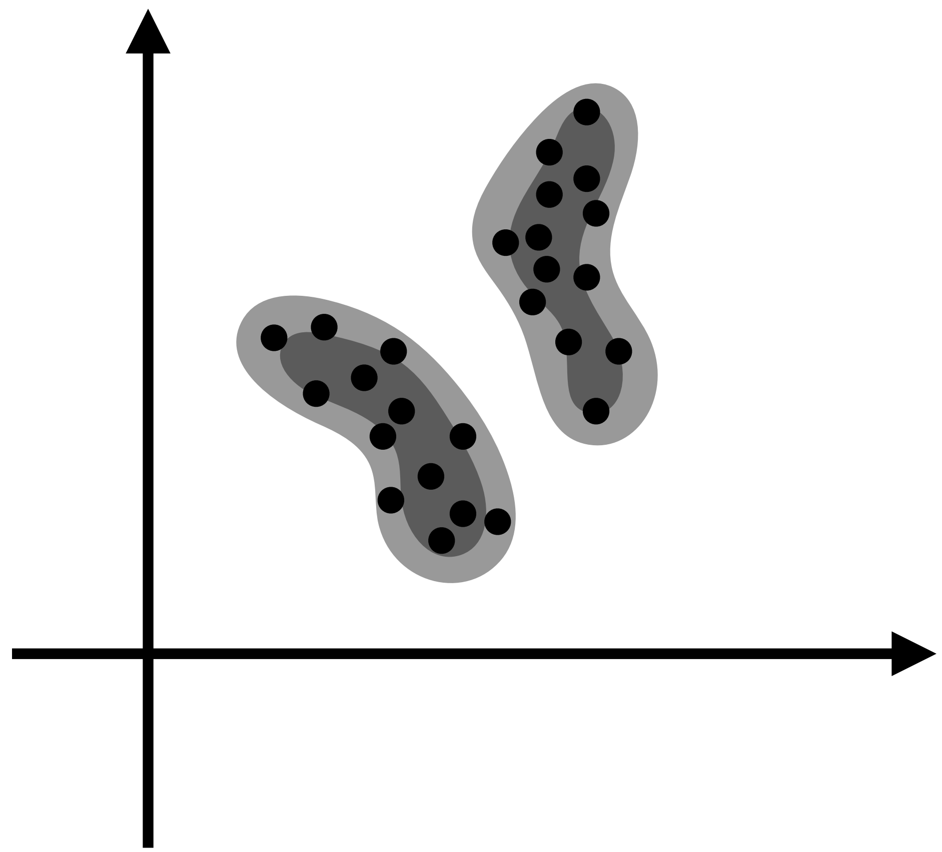
can both be trained using supervised learning

generative models are often easier to train with unsupervised methods

generative models typically require more modeling assumptions

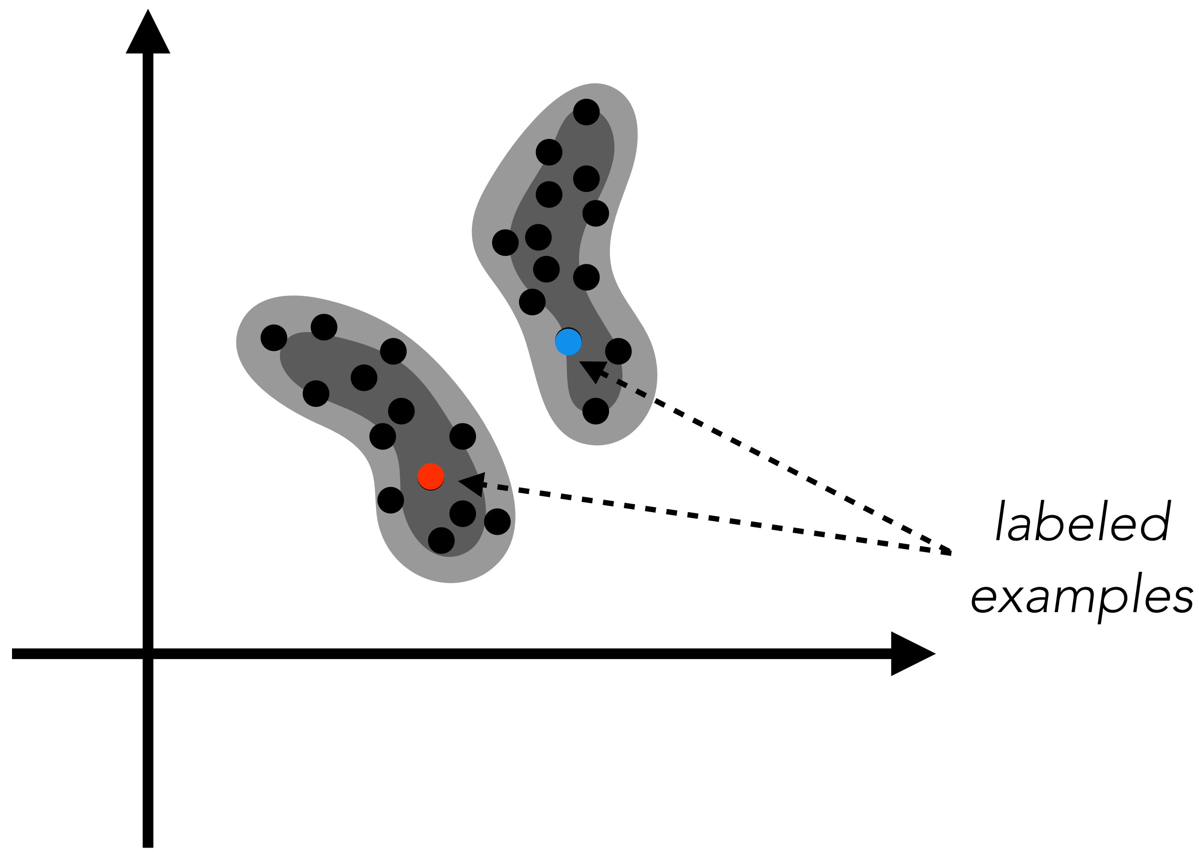
straightforward to quantify uncertainty with generative models

one of the main benefits of generative modeling is the ability to automatically extract structure from data



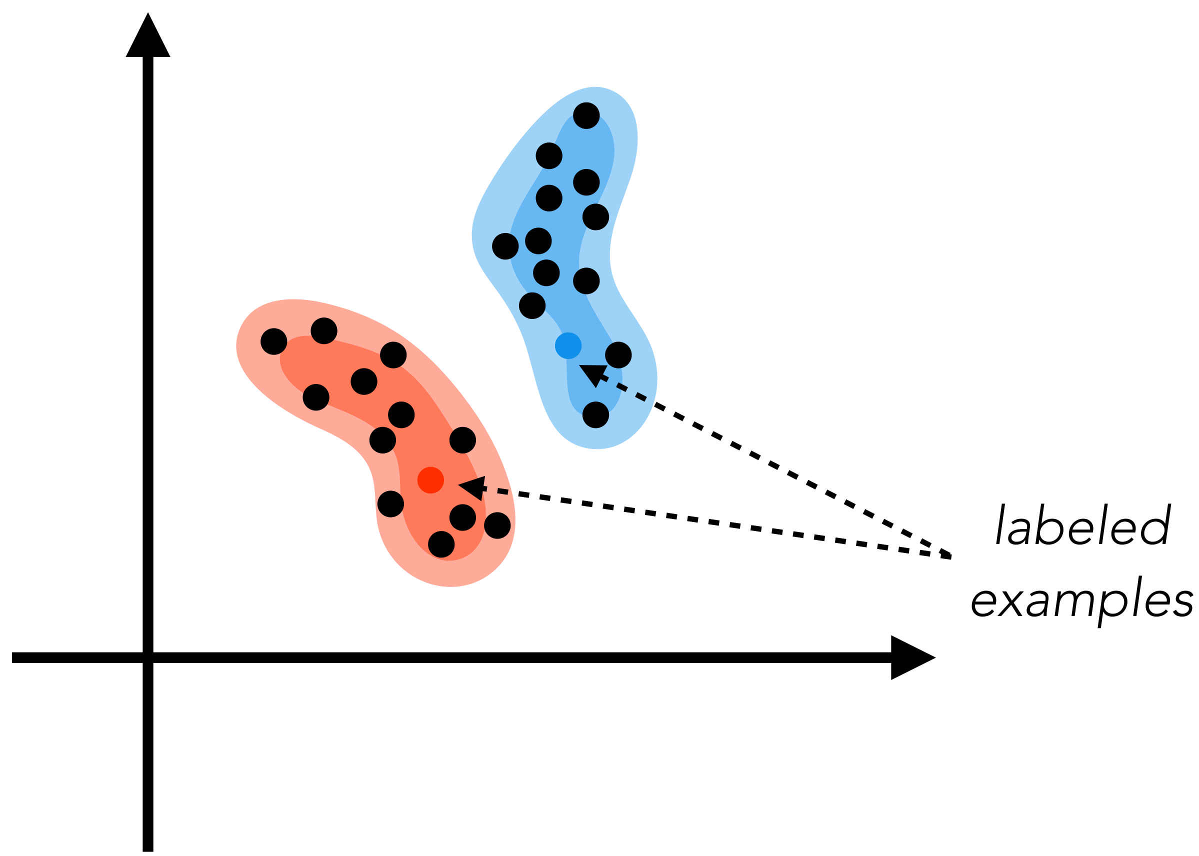
reducing the effective dimensionality of the data
can make it easier to learn and generalize on new tasks

one of the main benefits of generative modeling is the ability to automatically extract structure from data



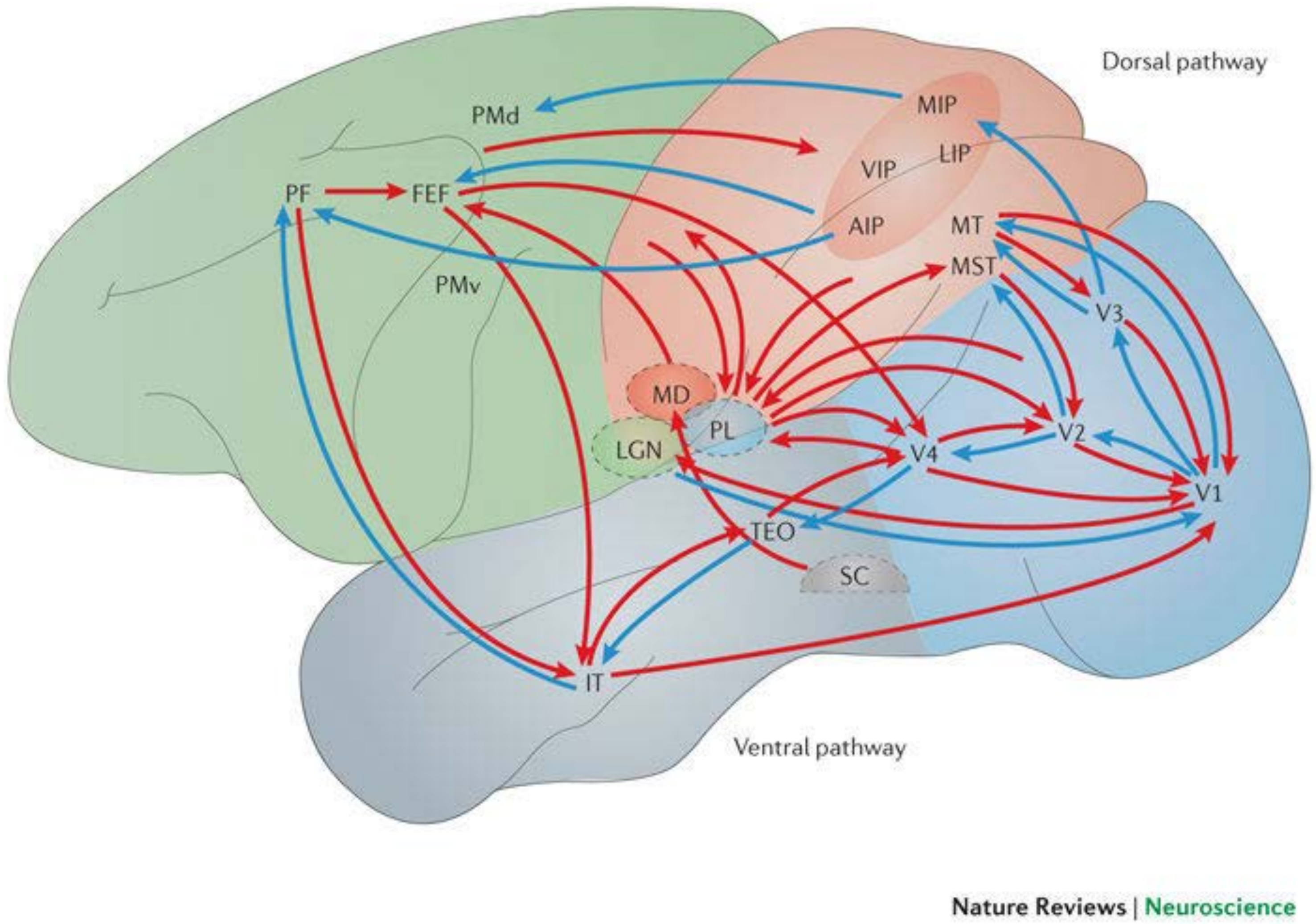
reducing the effective dimensionality of the data
can make it easier to learn and generalize on new tasks

one of the main benefits of generative modeling is the ability to automatically extract structure from data



reducing the effective dimensionality of the data can make it easier to learn and generalize on new tasks

any model that has an output in the data space
can be considered a generative model



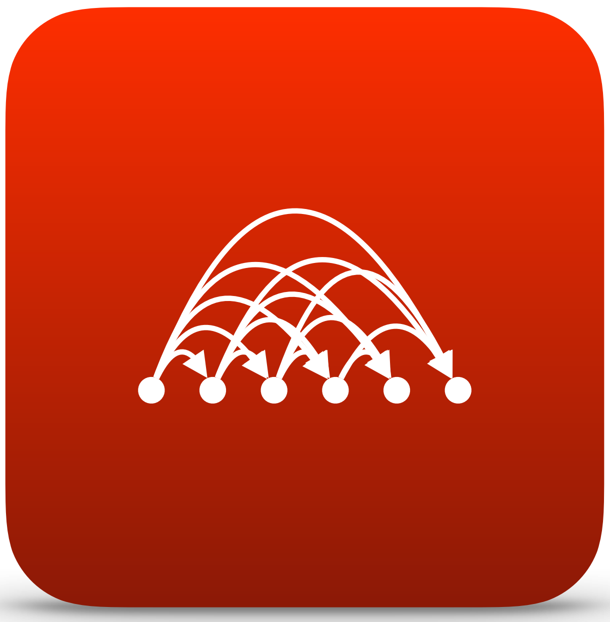
nervous systems appear to use this mechanism in part

prediction of sensory input using "top-down" pathways

deep generative model

a generative model that uses deep neural networks to model the data distribution

FAMILIES OF (DEEP) GENERATIVE MODELS



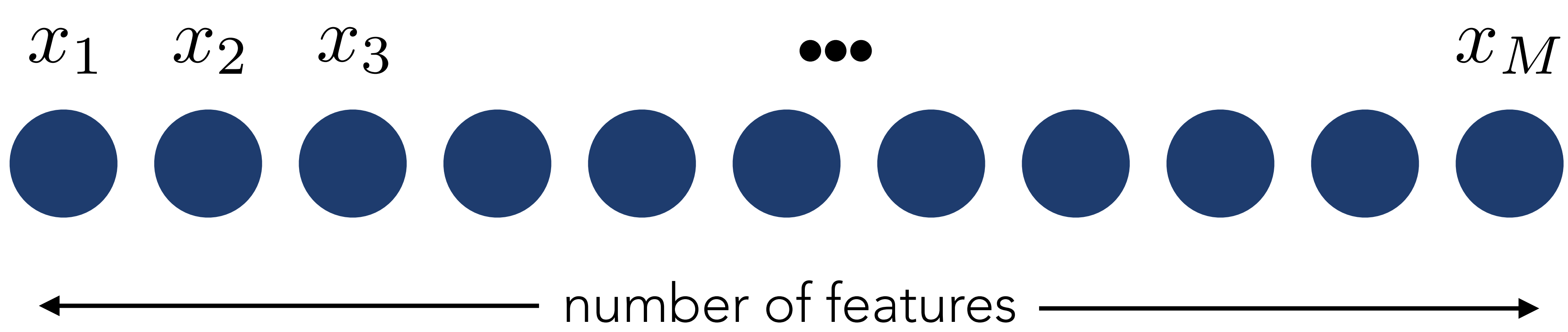
auto-regressive
models

latent variable
models

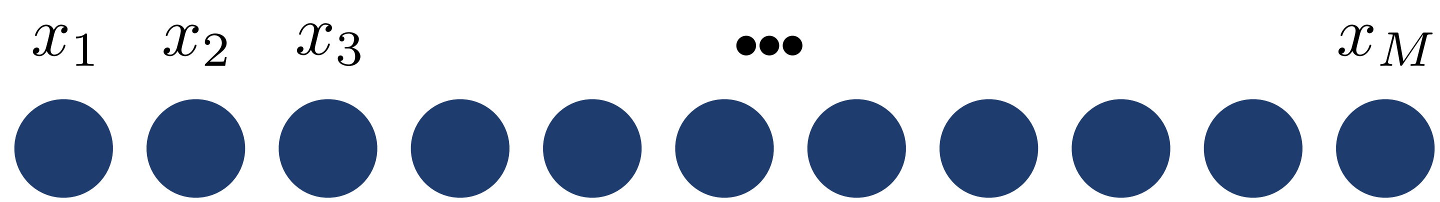
implicit
models

AUTO-REGRESSIVE MODELS

a data example



$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_M)$$



$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_M)$$

use *chain rule of probability* to split the joint distribution into a product of conditional distributions

definition of conditional probability

$$p(a|b) = \frac{p(a,b)}{p(b)} \longrightarrow p(a,b) = p(a|b)p(b)$$

recursively apply to $p(x_1, x_2, \ldots, x_M)$

$$p(x_1, x_2, \ldots, x_M) = p(x_1|x_2, \ldots, x_M)p(x_2, \ldots, x_M)$$

$$\vdots$$

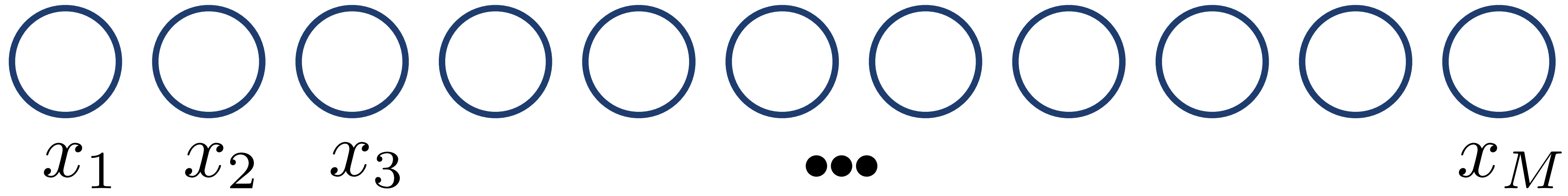
$$p(x_1, x_2, \ldots, x_M) = p(x_1|x_2, \ldots, x_M)p(x_2|x_3, \ldots, x_M) \ldots p(x_{M-1}|x_M)p(x_M)$$

note: conditioning order is arbitrary

$$p(x_1, \ldots, x_M) = \prod_{j=1}^{M} p(x_j|x_1, \ldots, x_{j-1})$$

model the conditional distributions of the data

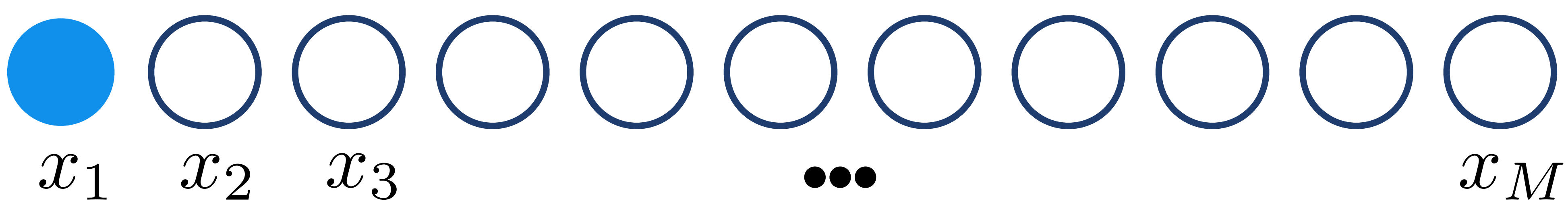
learn to ***auto-regress*** to the missing values



model the conditional distributions of the data

learn to ***auto-regress*** to the missing values

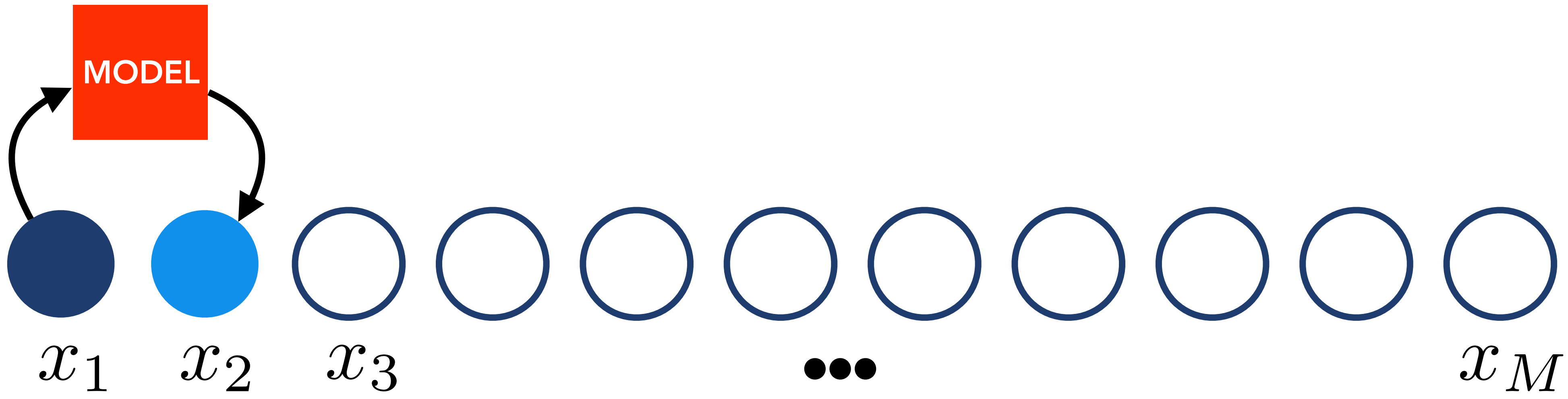
$$p(x_1)$$



model the conditional distributions of the data

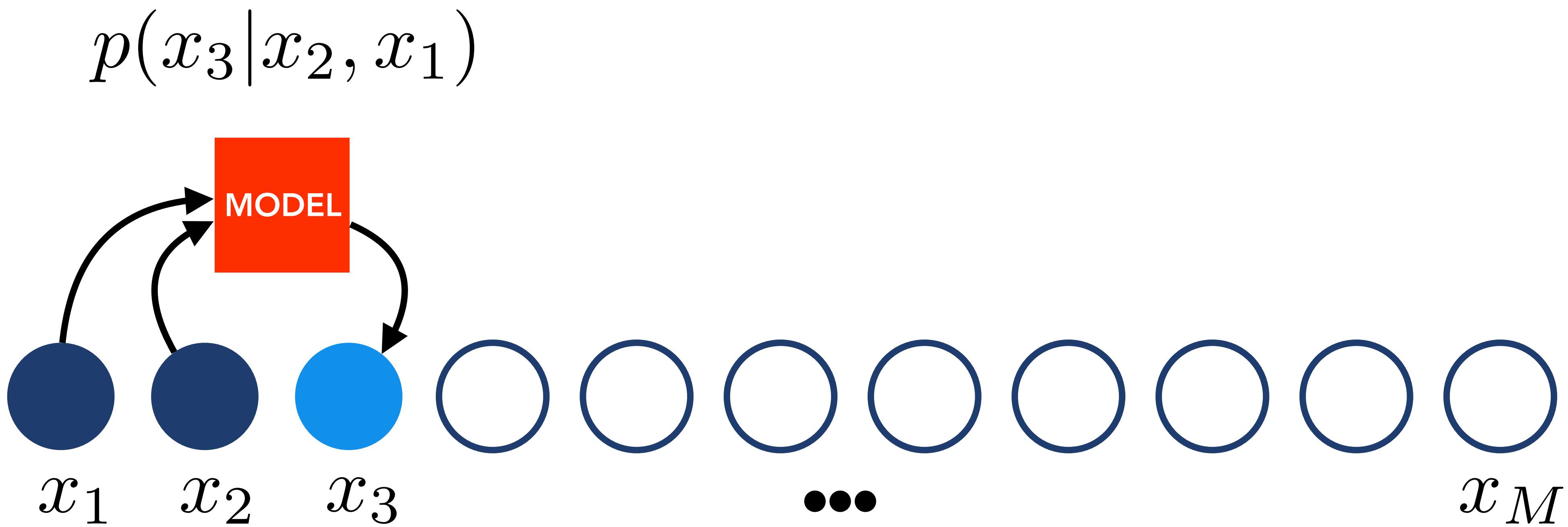
learn to ***auto-regress*** to the missing values

$p(x_2|x_1)$



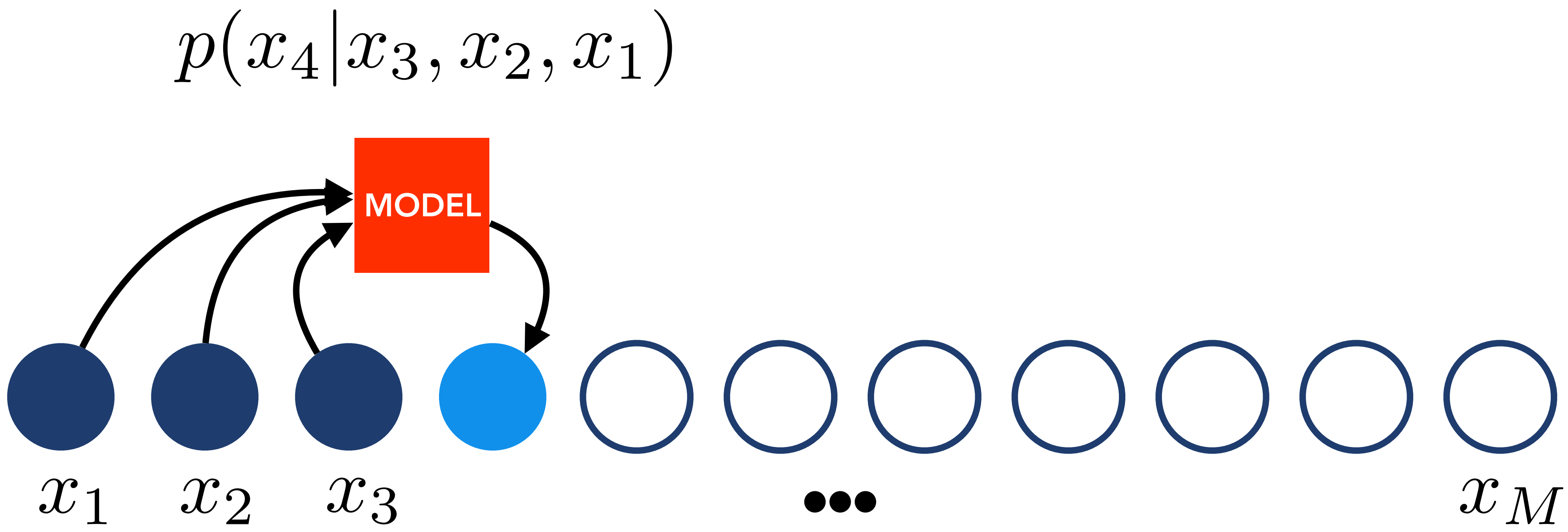
model the conditional distributions of the data

learn to ***auto-regress*** to the missing values



model the conditional distributions of the data

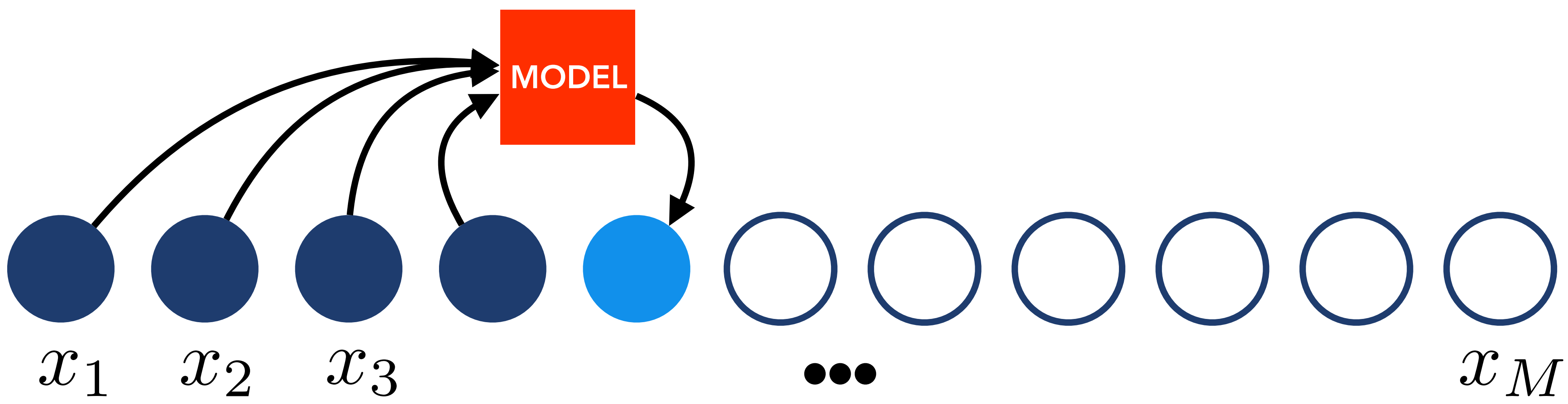
learn to ***auto-regress*** to the missing values



model the conditional distributions of the data

learn to ***auto-regress*** to the missing values

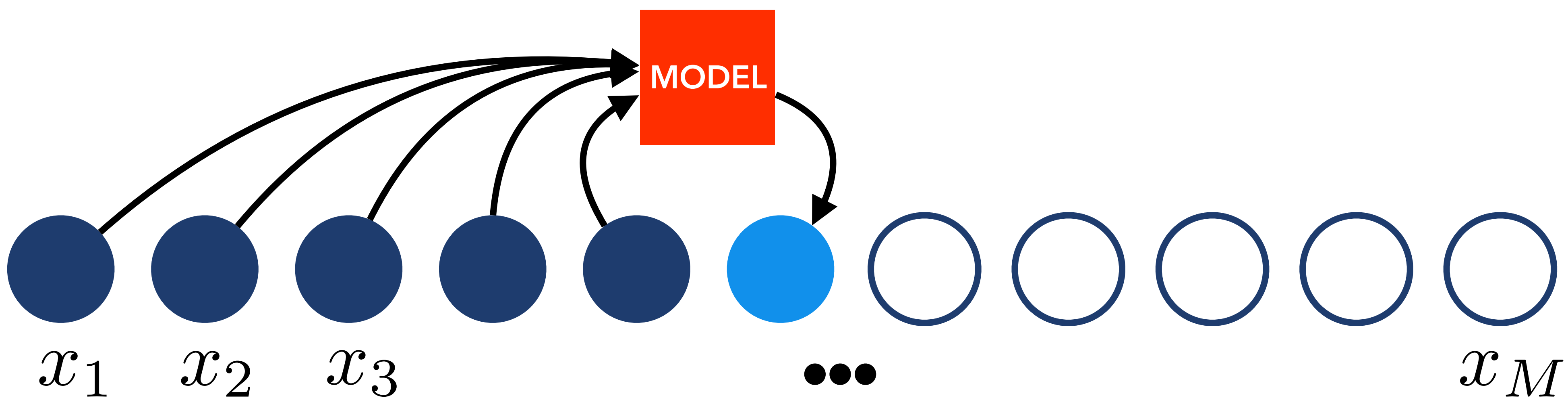
$$p(x_5|x_4, x_3, x_2, x_1)$$



model the conditional distributions of the data

learn to ***auto-regress*** to the missing values

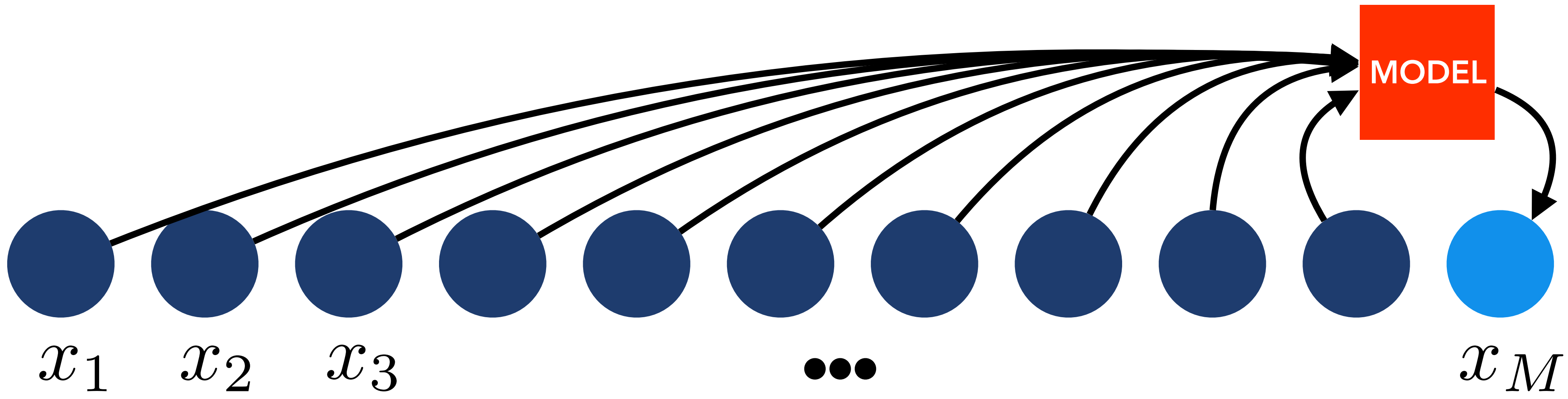
$$p(x_6 | x_5, x_4, x_3, x_2, x_1)$$



model the conditional distributions of the data

learn to ***auto-regress*** to the missing values

$$p(x_M | x_{M-1}, \ldots, x_1)$$



maximum likelihood

to fit the model to the empirical data distribution,
maximize the *likelihood* of the true data examples

likelihood: $$p(\mathbf{x}) = \prod_{j=1}^{M} p(x_j | \mathbf{x}_{<j})$$

auto-regressive conditionals

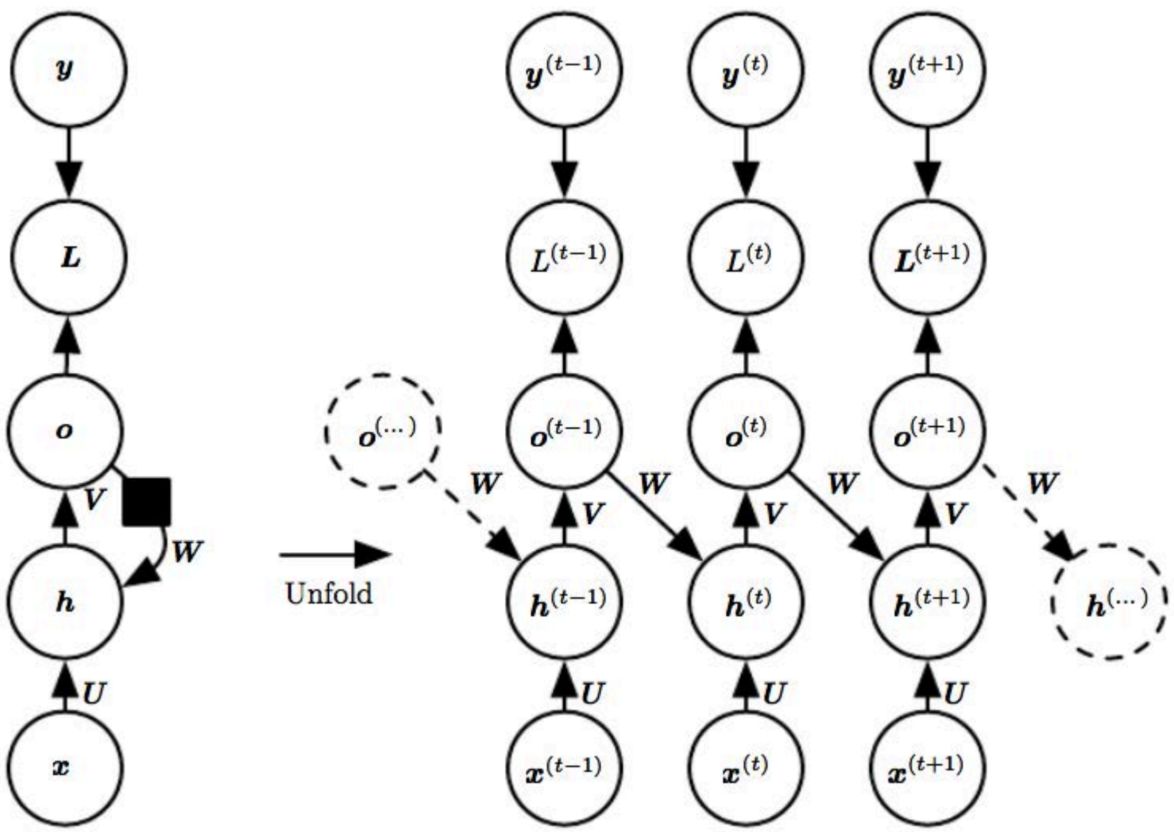
optimize the parameters to assign high (log) probability to the true data examples

learning: $$\theta^* = \mathrm{argmax}_\theta \log p(\mathbf{x})$$

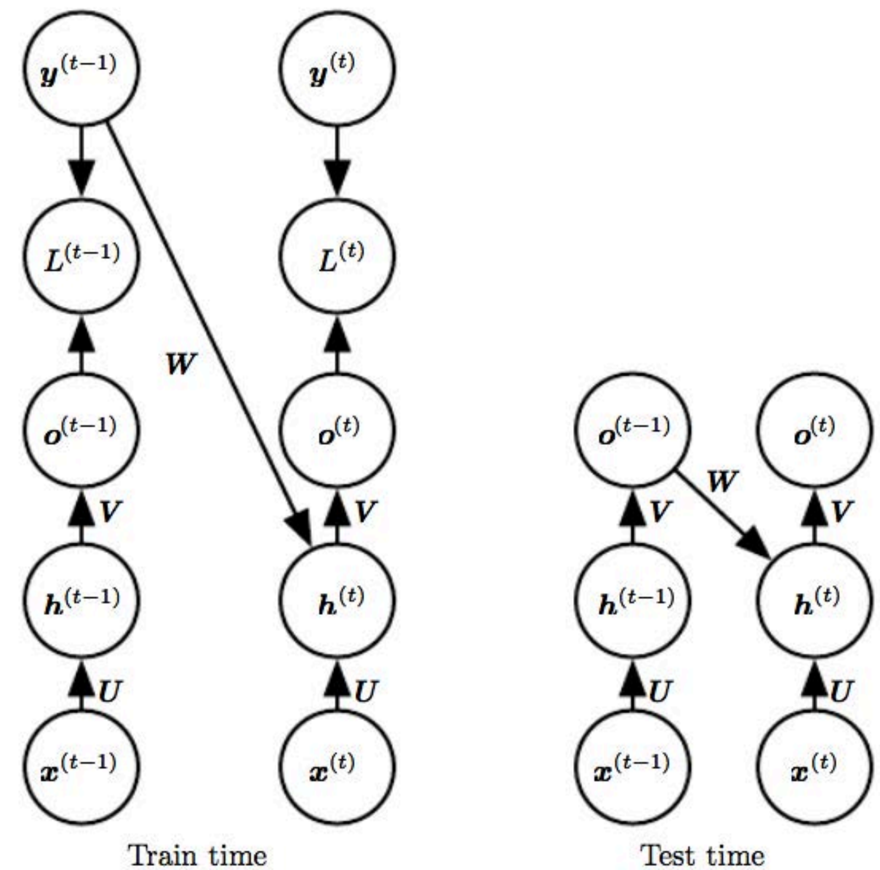
logarithm for numerical stability

models

can parameterize conditional distributions using a **recurrent neural network**



unrolling auto-regressive generation from an RNN



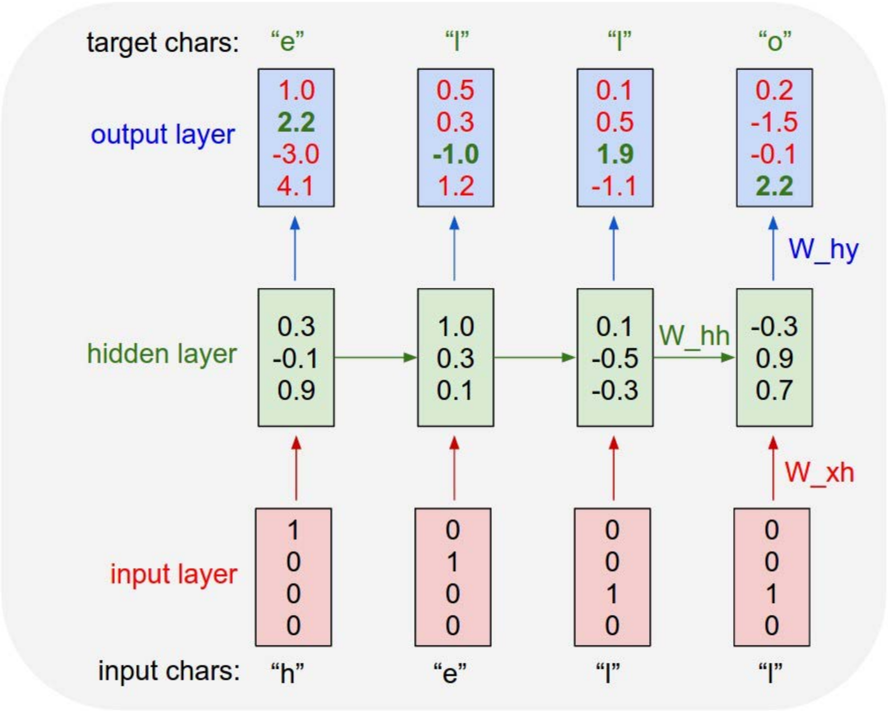
"teacher forcing"

Deep Learning, *Goodfellow et al.*, 2016

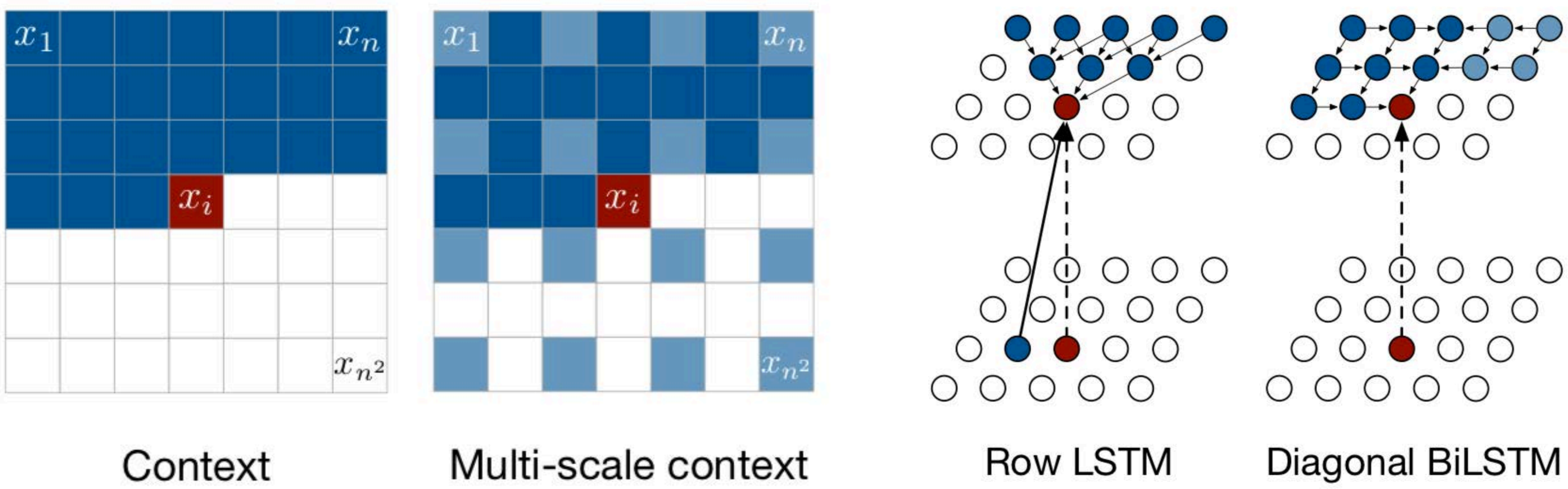
(chapter 10)

models

can parameterize conditional distributions using a **recurrent neural network**



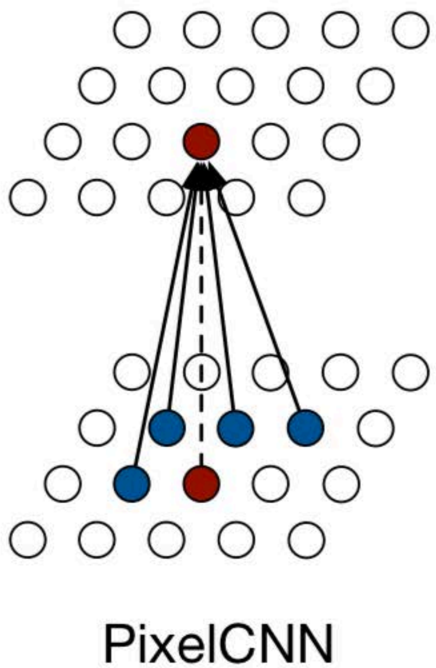
The Unreasonable Effectiveness of Recurrent Neural Networks, *Karpathy*, 2015



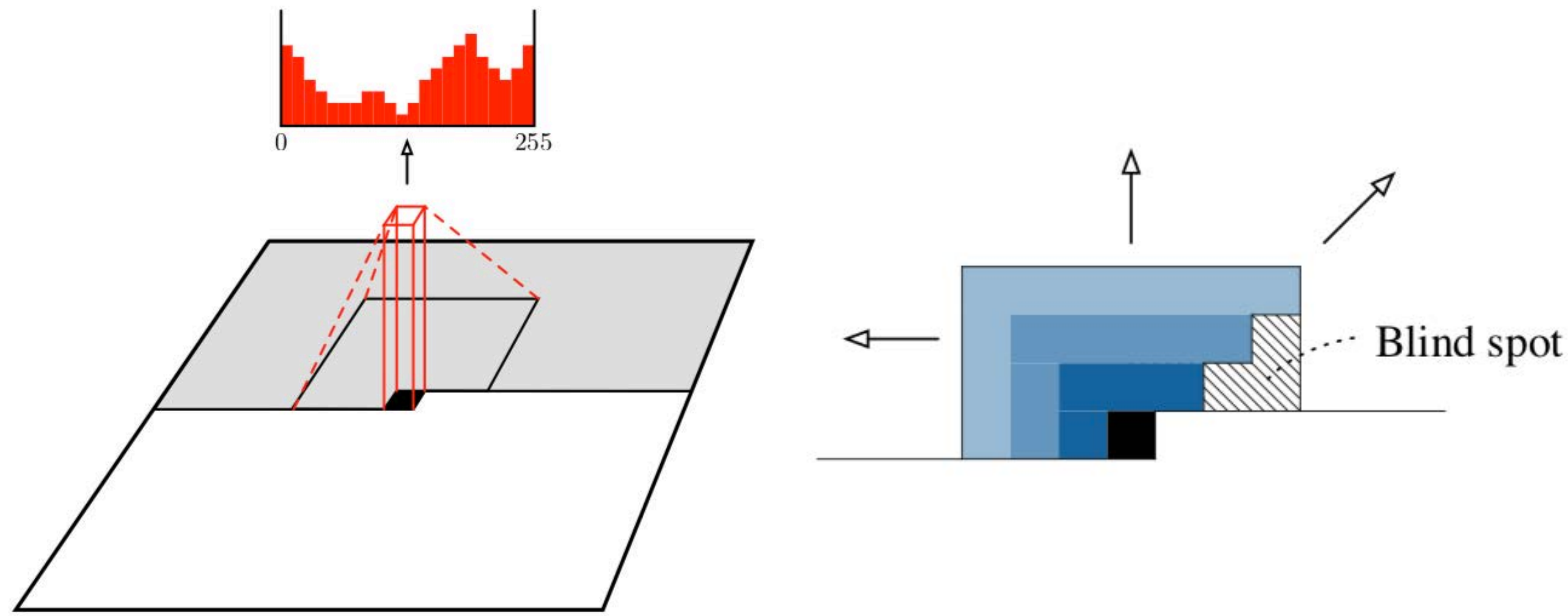
Pixel Recurrent Neural Networks, *van den Oord et al.*, 2016

models

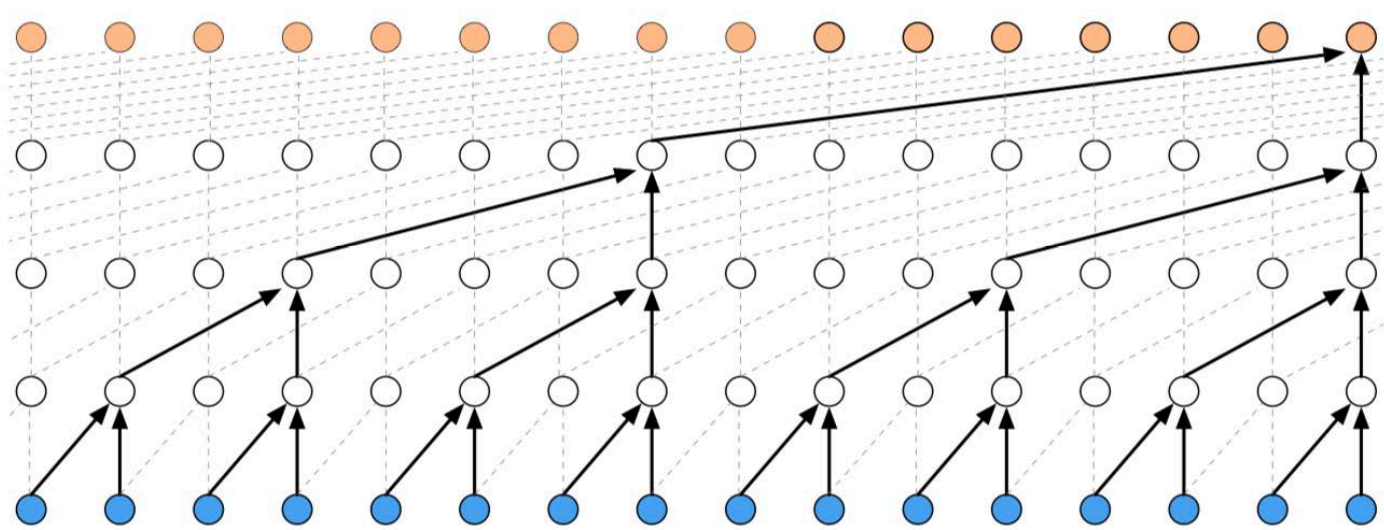
can also condition on a local window using **convolutional neural networks**



Pixel Recurrent Neural Networks, *van den Oord et al.*, 2016



Conditional Image Generation with PixelCNN Decoders, *van den Oord et al.*, 2016



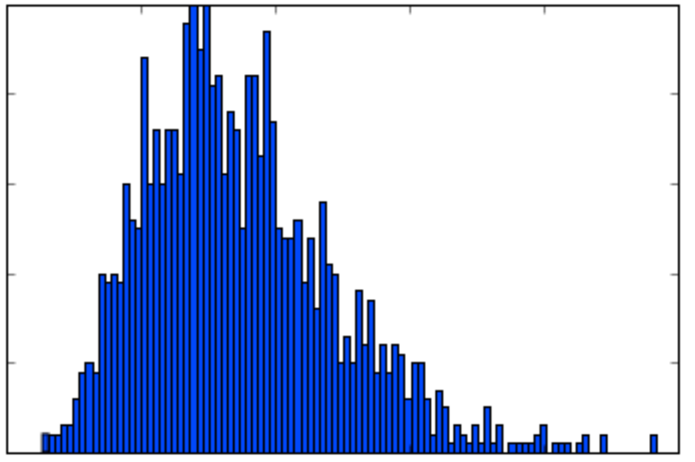
WaveNet: A Generative Model for Raw Audio, *van den Oord et al.*, 2016

output distributions

need to choose a form for the conditional **output distribution**,
i.e. how do we express $p(x_j | x_1, \ldots, x_{j-1})$?

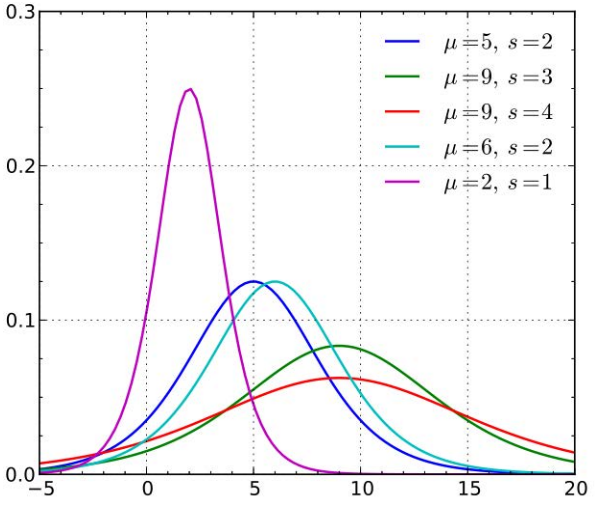
model the data as ***categorical*** *variables*

→ *multinomial output*



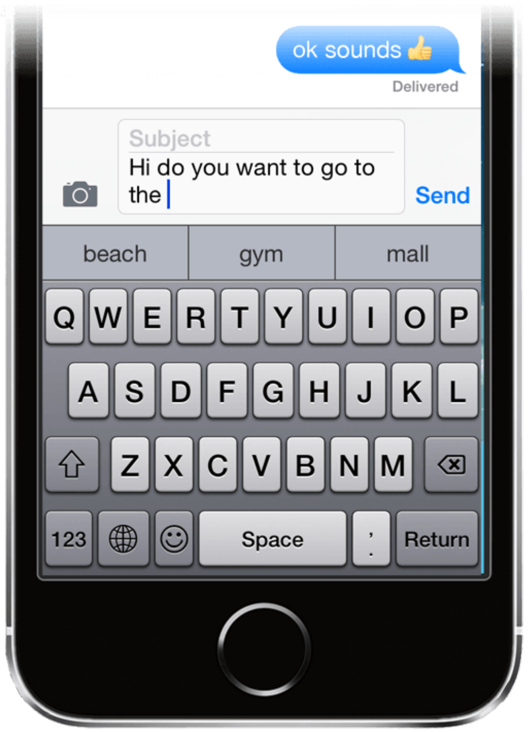
model the data as ***continuous*** *variables*

→ *Gaussian, logistic, etc. output*



example applications

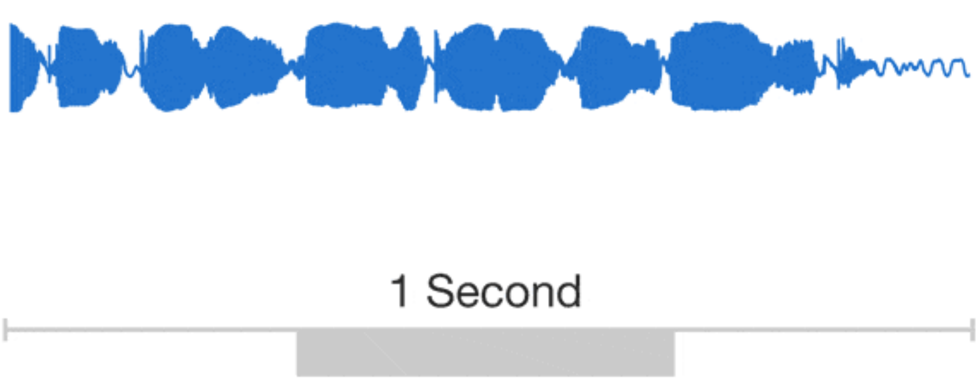
text



images

Pixel Recurrent Neural Networks, *van den Oord et al.*, 2016

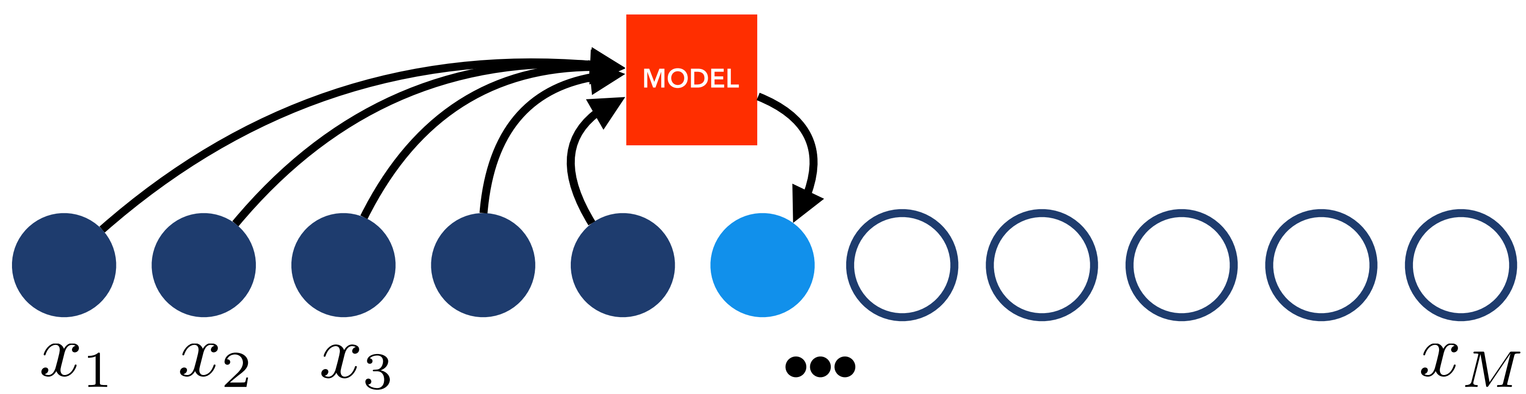
speech



WaveNet: A Generative Model for Raw Audio, *van den Oord et al.*, 2016

recap: auto-regressive models

$$p(x_6|x_5, x_4, x_3, x_2, x_1)$$



model conditional distributions to *auto-regress* to missing values

Pros

tractable and straightforward to evaluate the (log) likelihood

great at capturing details

superior quantitative performance

Cons

difficult to capture "high-level" global structure

need to impose conditioning order

sequential sampling is computationally expensive

EXPLICIT LATENT VARIABLE MODELS

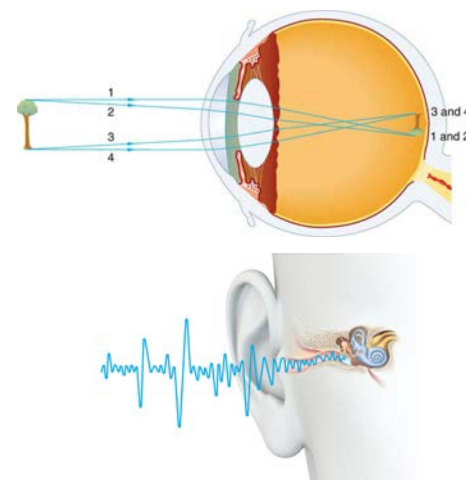
reality ***generates*** sensory stimuli from underlying latent phenomena

REALITY

matter
energy
forces
etc.

laws of nature →

STIMULI

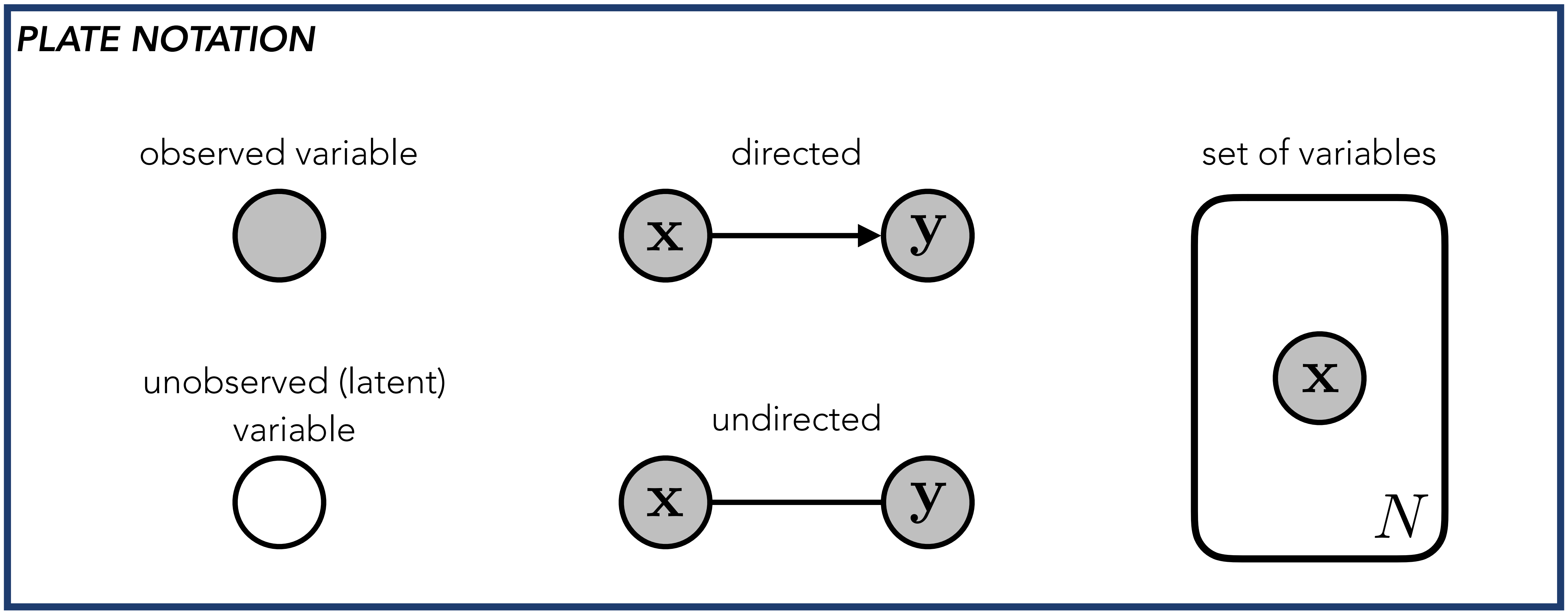


inference ⇢

PERCEPTION

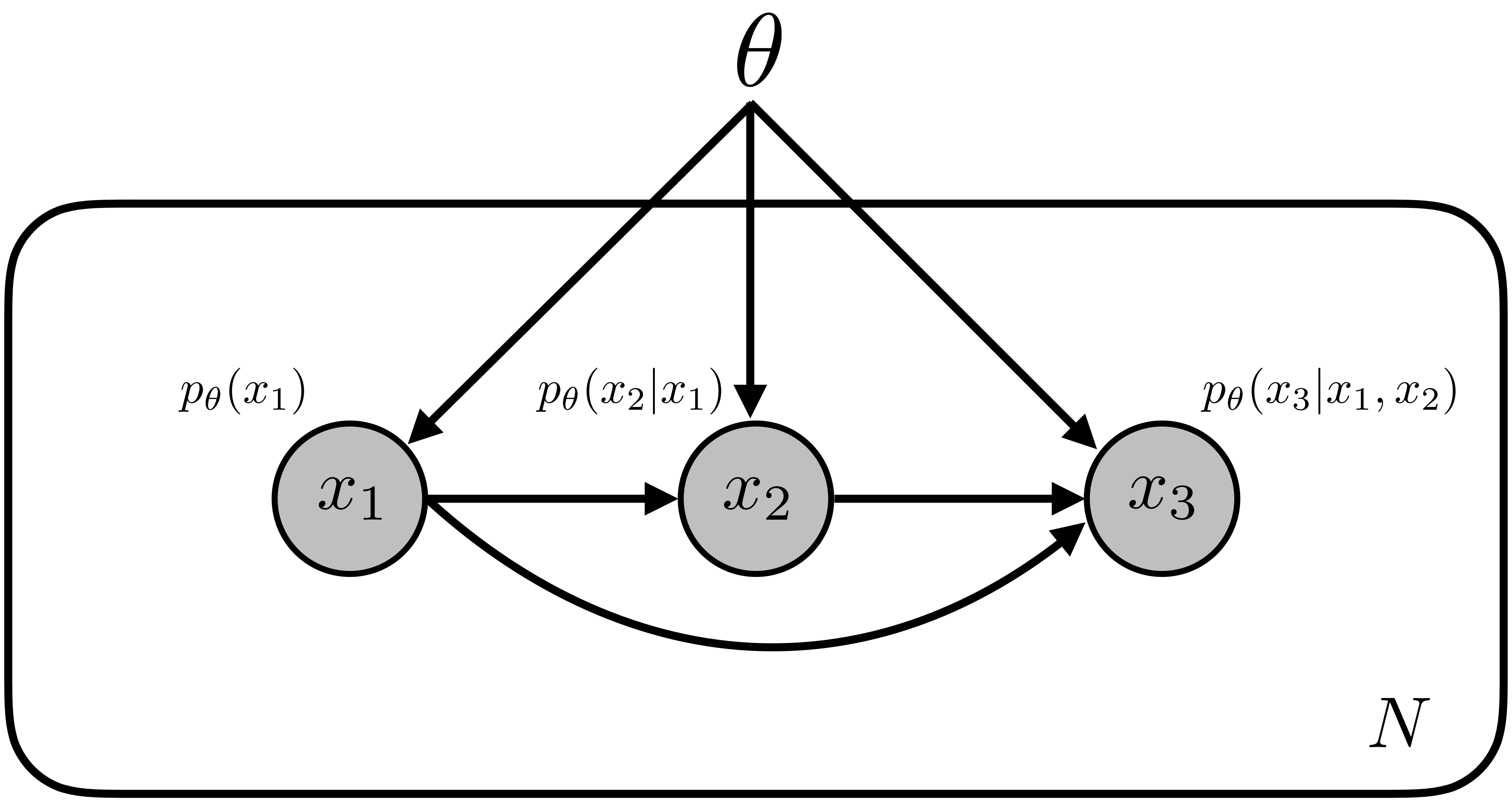
object identities
object locations
communication
etc.

can use *latent variables* to help model these phenomena probabilistic graphical models provide a framework for modeling relationships between random variables

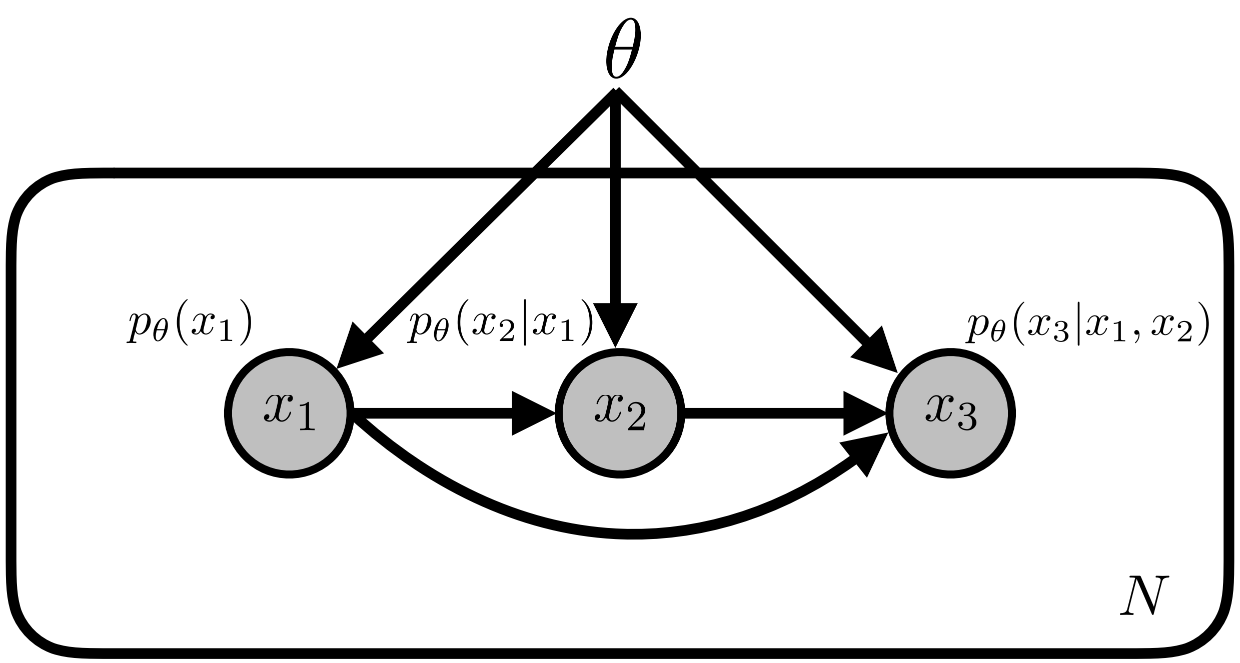


review exercise:

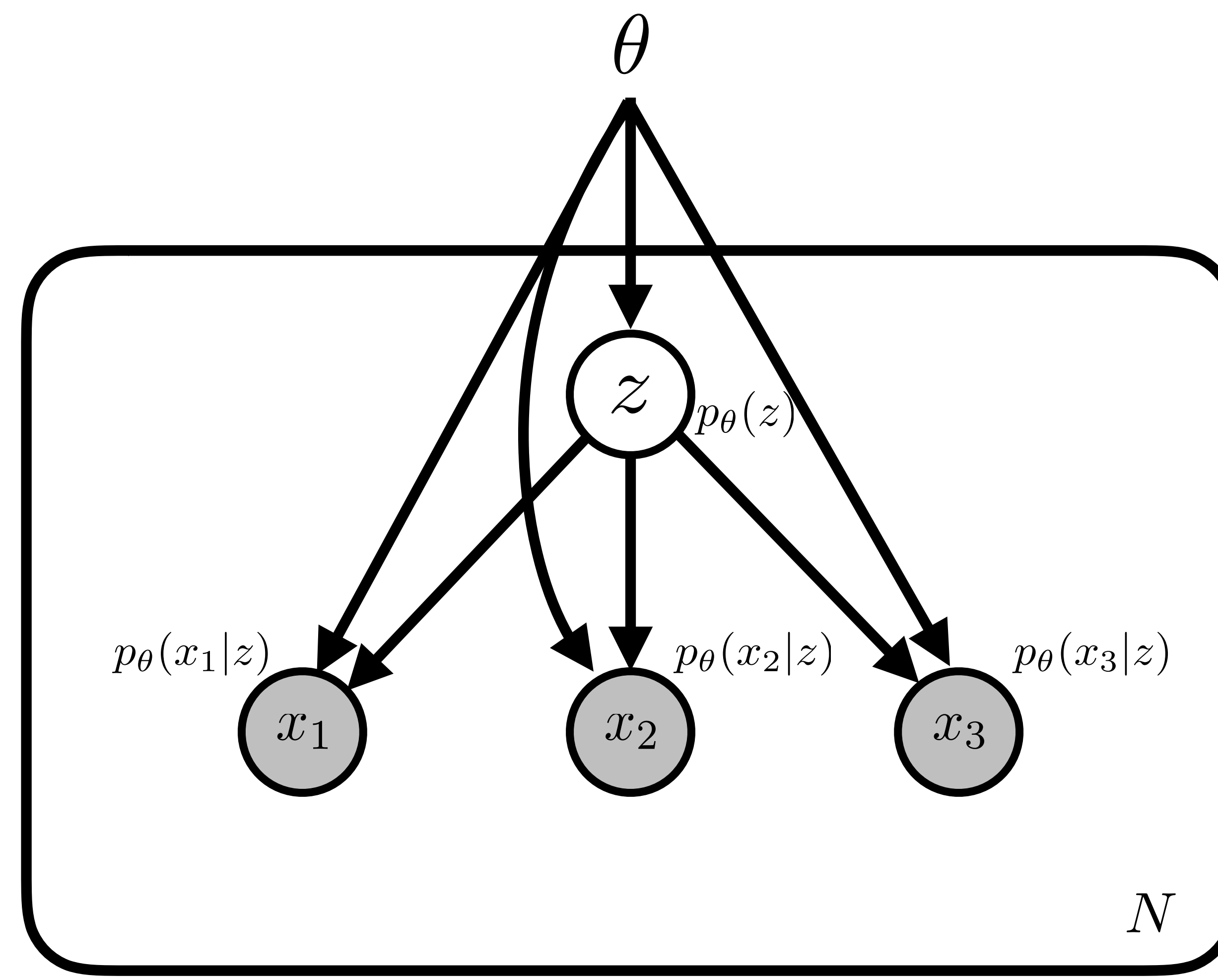
represent an auto-regressive model of 3 random variables with plate notation



comparing *auto-regressive models* and *latent variable models*



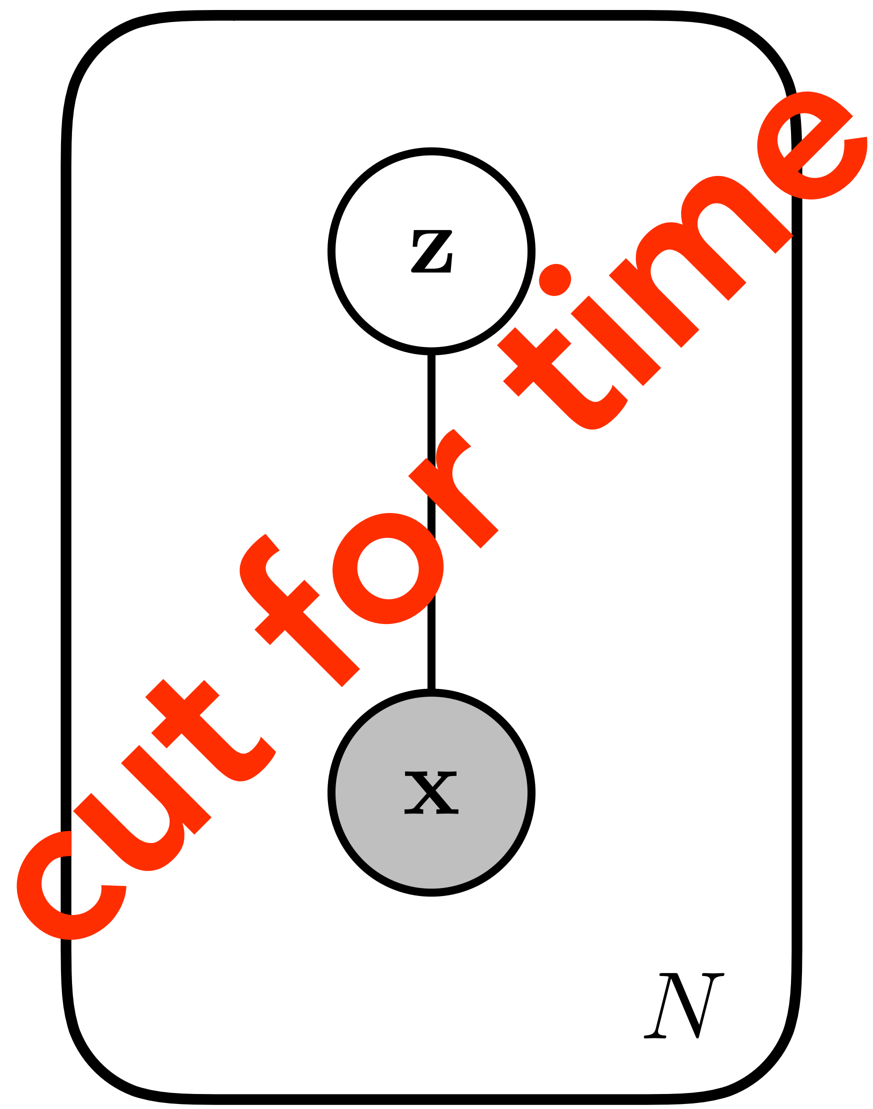
auto-regressive model



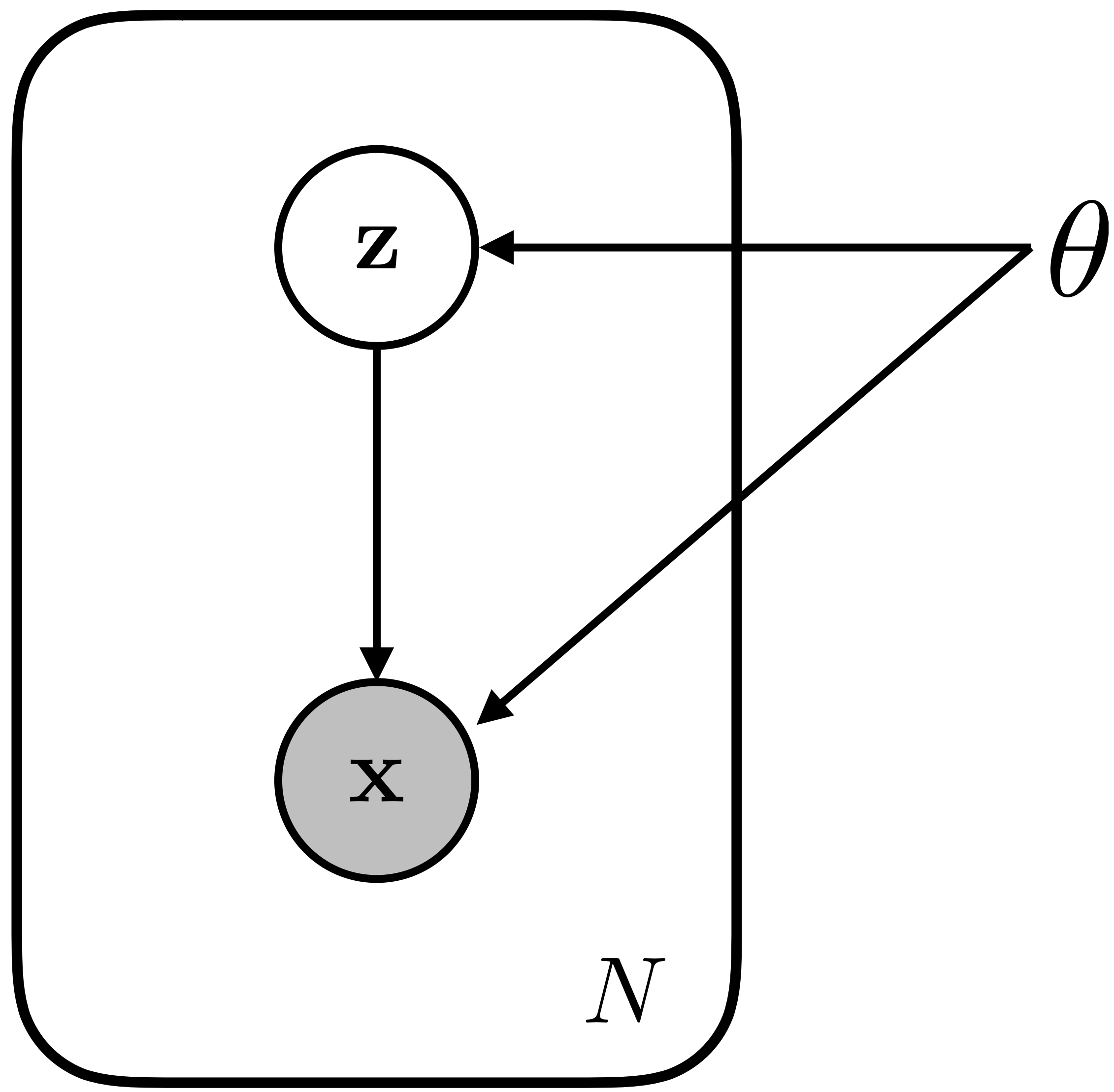
latent variable model

example: **<u>undirected</u> latent variable model**

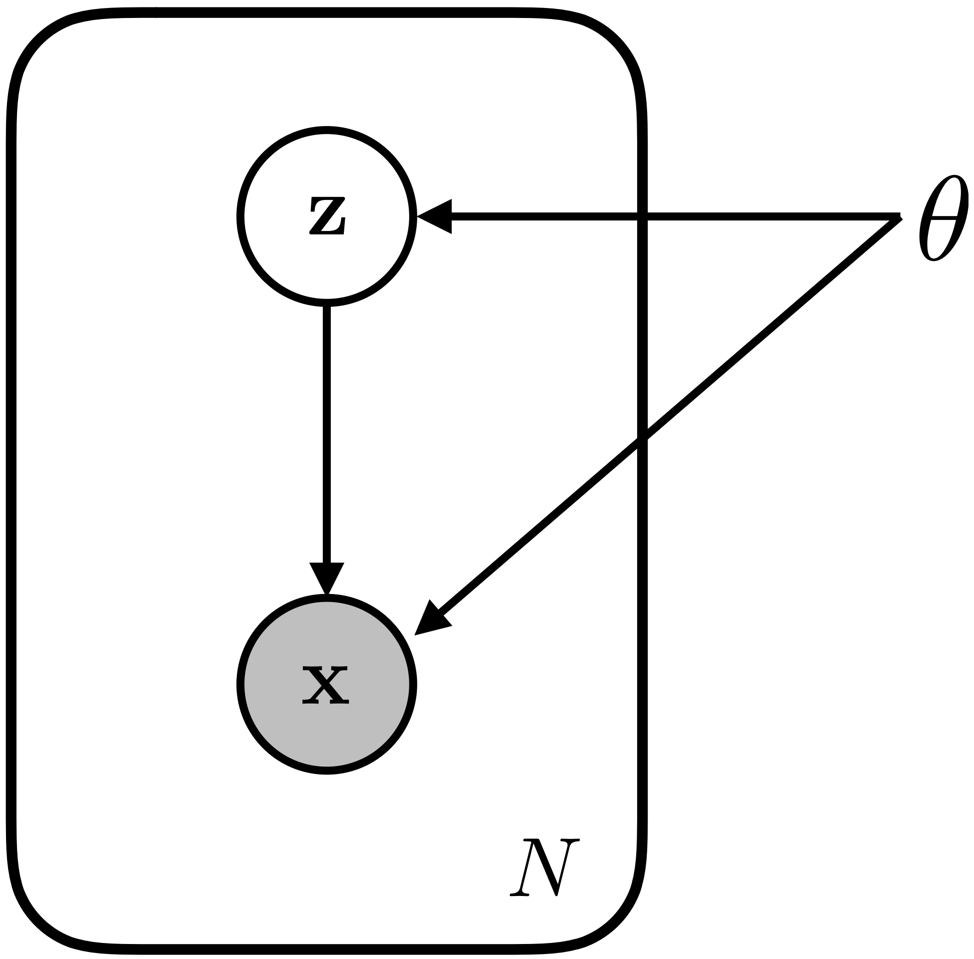
→ restricted Boltzmann machine



example: **<u>directed</u> latent variable model**



example: <u>**directed**</u> **latent variable model**



Generation

GENERATIVE MODEL

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

joint

conditional likelihood

prior

1. sample $\mathbf{z}$ from $p(\mathbf{z})$
2. use $\mathbf{z}$ samples to sample $\mathbf{x}$ from $p(\mathbf{x}|\mathbf{z})$

intuitive example

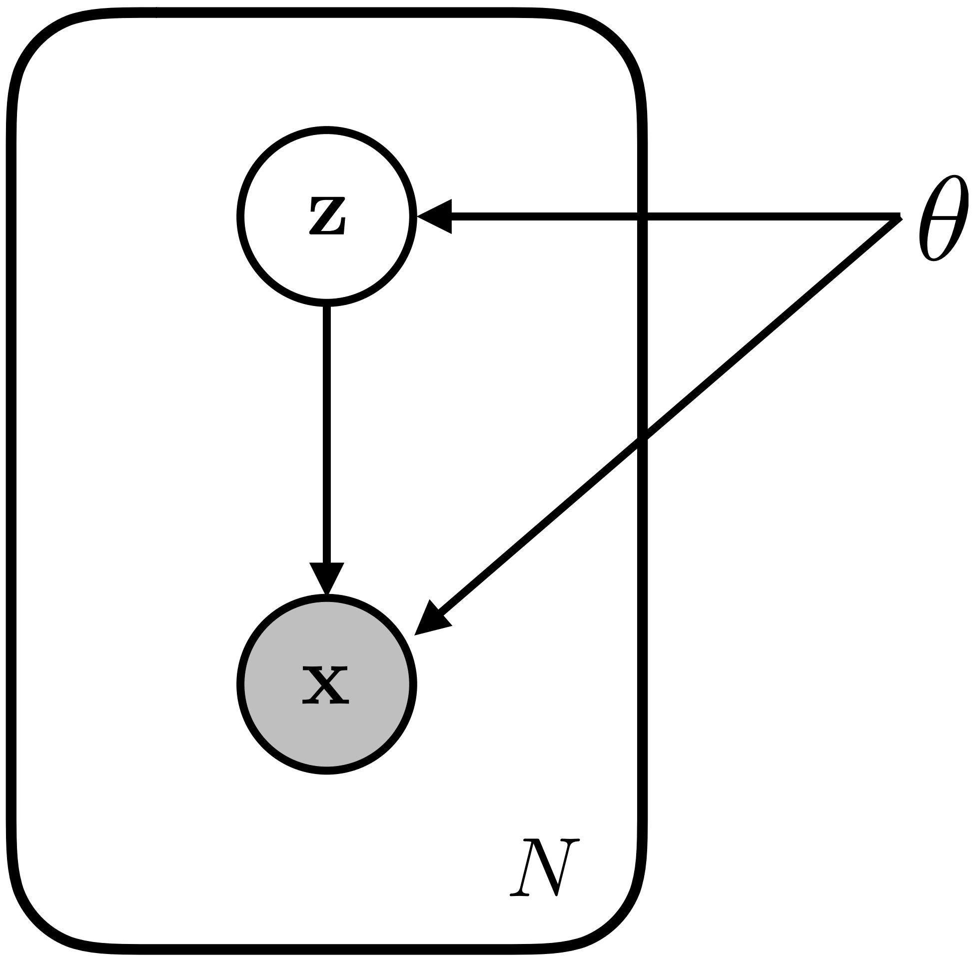
object ~ p(objects)

lighting ~ p(lighting)

background ~ p(bg)

RENDER

example: **<u>directed</u> latent variable model**



Posterior Inference

INFERENCE

joint

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$

posterior

marginal likelihood

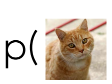
use Bayes' rule (def. of cond. prob.)

provides conditional distribution over latent variables

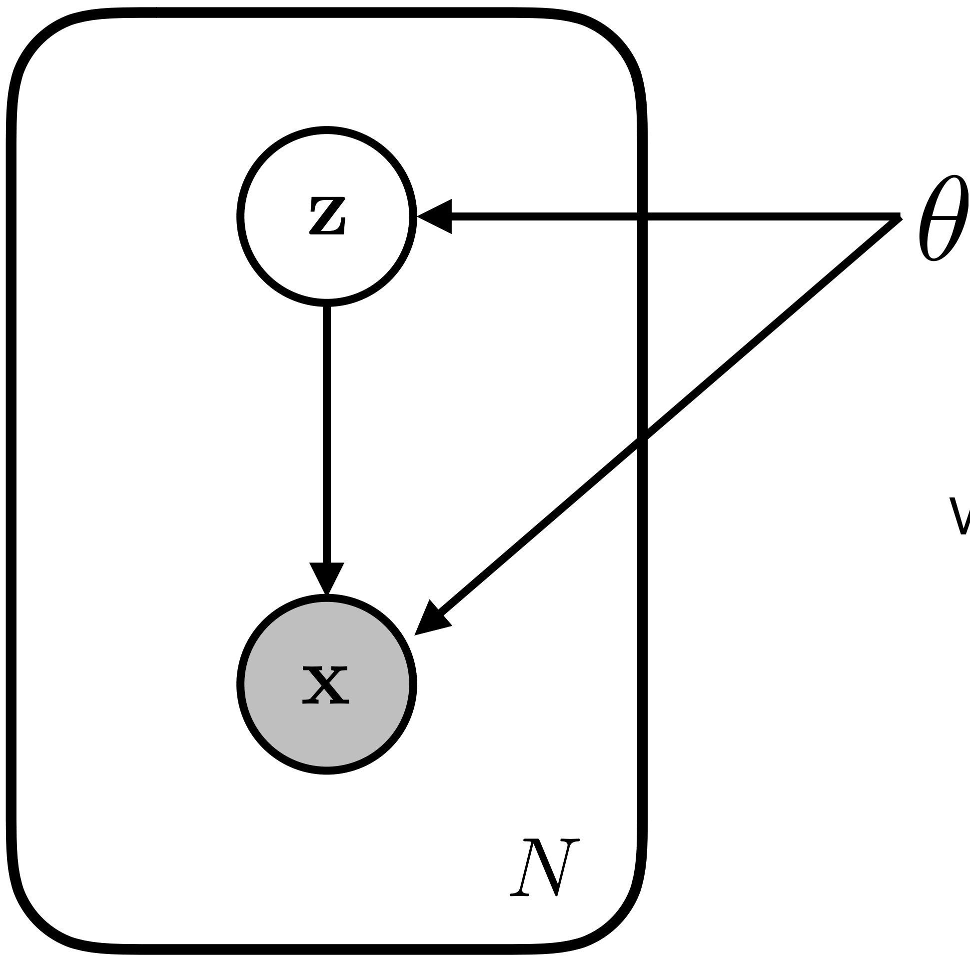
intuitive example

observation

what is the probability that I am observing a cat given these pixel observations?

p(cat | ) = p( |cat) p(cat) / p()

example: **<u>directed</u> latent variable model**



Model Evaluation

MARGINALIZATION

marginal likelihood

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

joint

to evaluate the likelihood of an observation, we need to *marginalize* over all latent variables

i.e. consider all possible underlying states

intuitive example

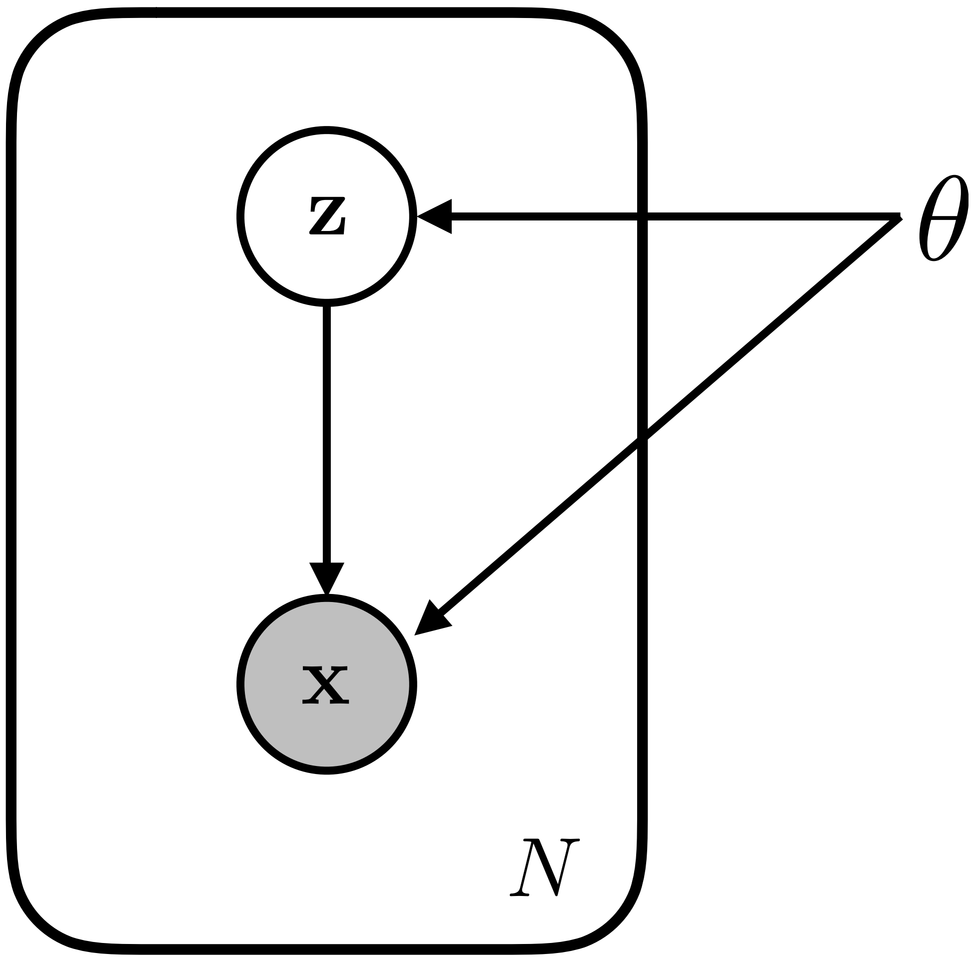
observation

how likely is this observation under my model?
(what is the probability of observing this?)

for all objects, lighting, backgrounds, etc.:

how plausible is this example?

example: **directed latent variable model**



to fit the model, we want to evaluate
the *marginal* (log) likelihood of the data

$$\theta^* = \mathrm{argmax}_\theta \log p(\mathbf{x})$$

however, this is generally *intractable*,
due to the integration over latent variables

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

integration in
high-dimensions

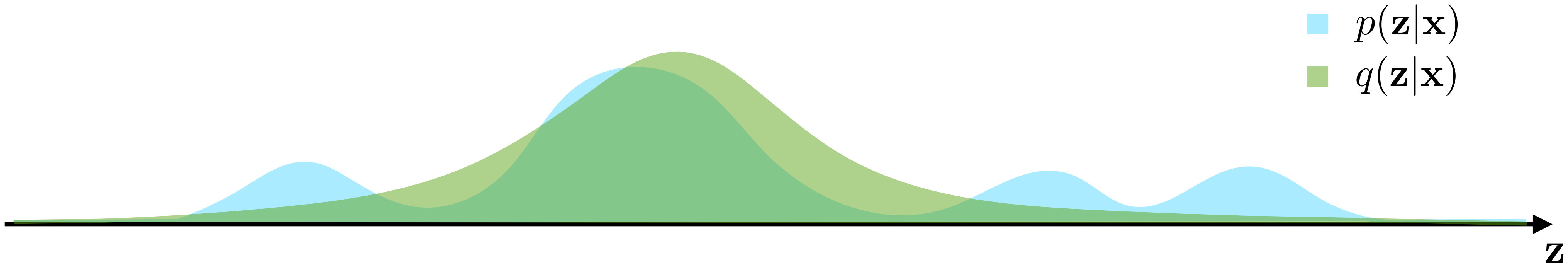
variational inference

<u>main idea</u>

instead of optimizing the (log) likelihood, optimize a ***lower bound*** on it

introduce an ***approximate posterior***, then minimize KL-divergence to the true posterior

$$q^*(\mathbf{z}|\mathbf{x}) = \text{argmin}_q D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$$



evaluating KL-divergence involves evaluating $p(\mathbf{z}|\mathbf{x})$, instead maximize $\mathcal{L}$:

$$q^*(\mathbf{z}|\mathbf{x}) = \text{argmax}_q \mathcal{L}$$

where $\mathcal{L}$ is the *evidence lower bound (ELBO), defined as* $\mathcal{L} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})]$

$\mathcal{L}$ provides a lower bound on $\log p(\mathbf{x})$, so we can use $\mathcal{L}$ to (approximately) fit the model

$$\tilde{\theta}^* = \text{argmax}_\theta \mathcal{L}$$

interpreting the ELBO

we can write the ELBO as

$$\begin{aligned}
\mathcal{L} &\equiv \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \\
&= \underbrace{\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]}_{\text{reconstruction}} - \underbrace{D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{regularization}}
\end{aligned}$$

$q(\mathbf{z}|\mathbf{x})$ is optimized to represent the data while staying close to the prior

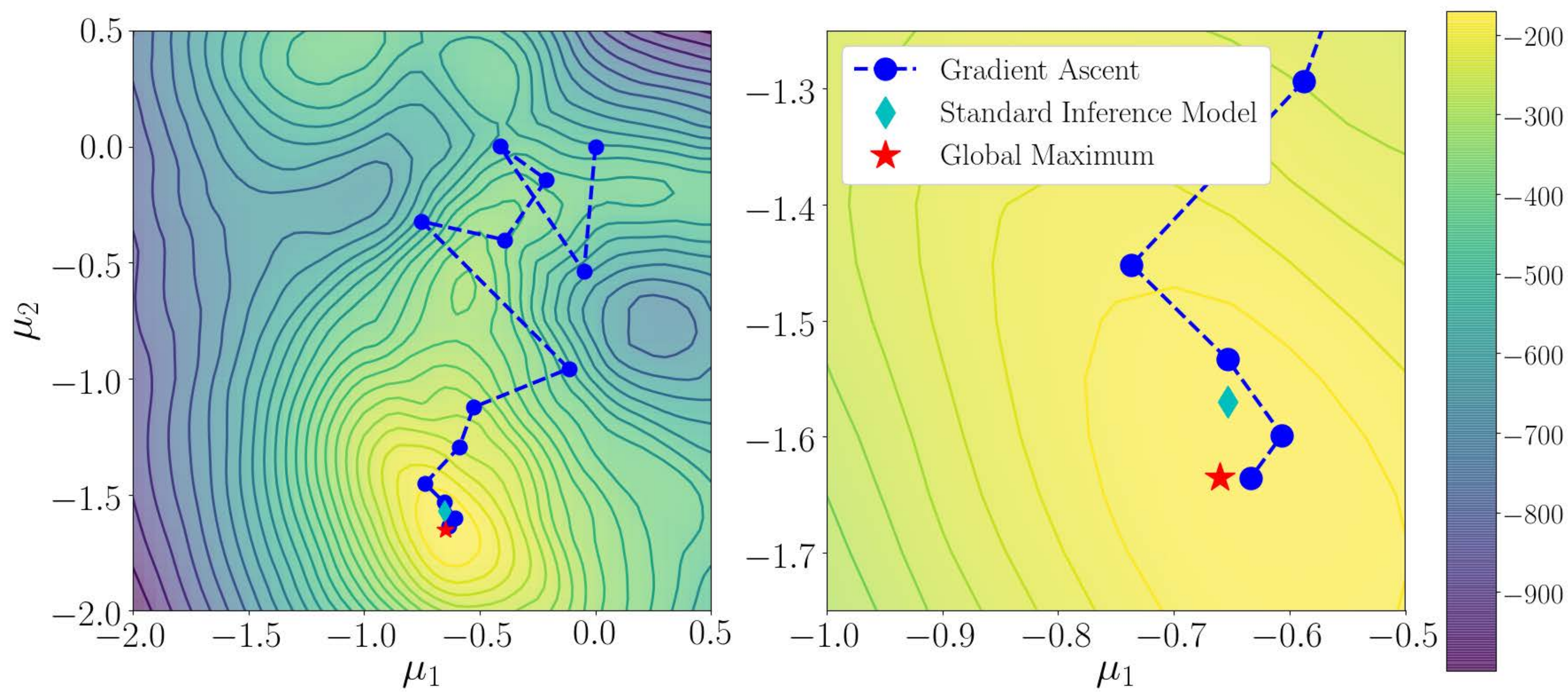
many connections to *compression, information theory*

resembles the "auto-encoding" framework

variational ***inference*** involves <u>optimizing</u>
the approximate posterior for each data example

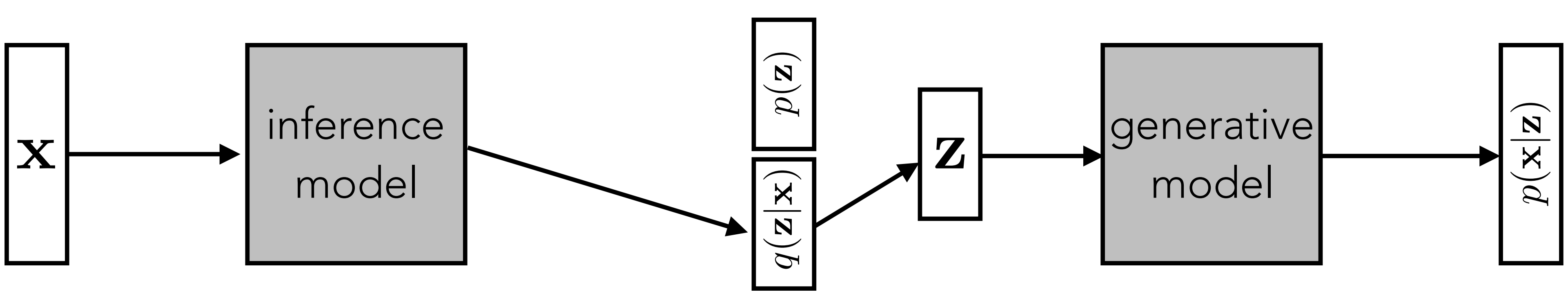
$$q^*(\mathbf{z}|\mathbf{x}) = \text{argmax}_q \mathcal{L}$$

can be solved using gradient ascent and (stochastic) backpropagation / REINFORCE,
but can be computationally expensive



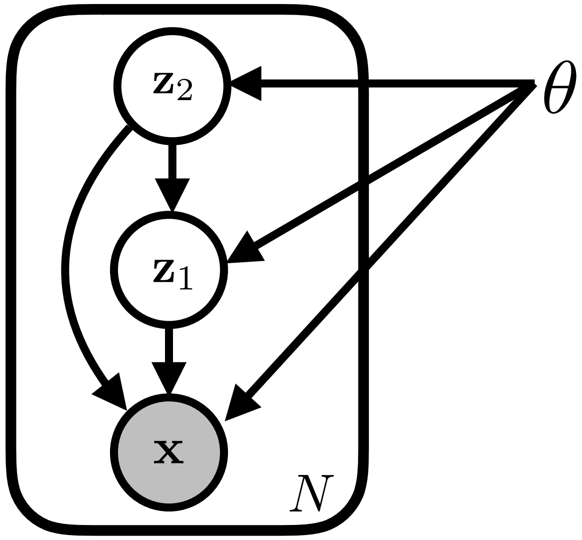
can instead *amortize* inference over data examples by learning
a separate ***inference model*** to output approximate posterior estimates

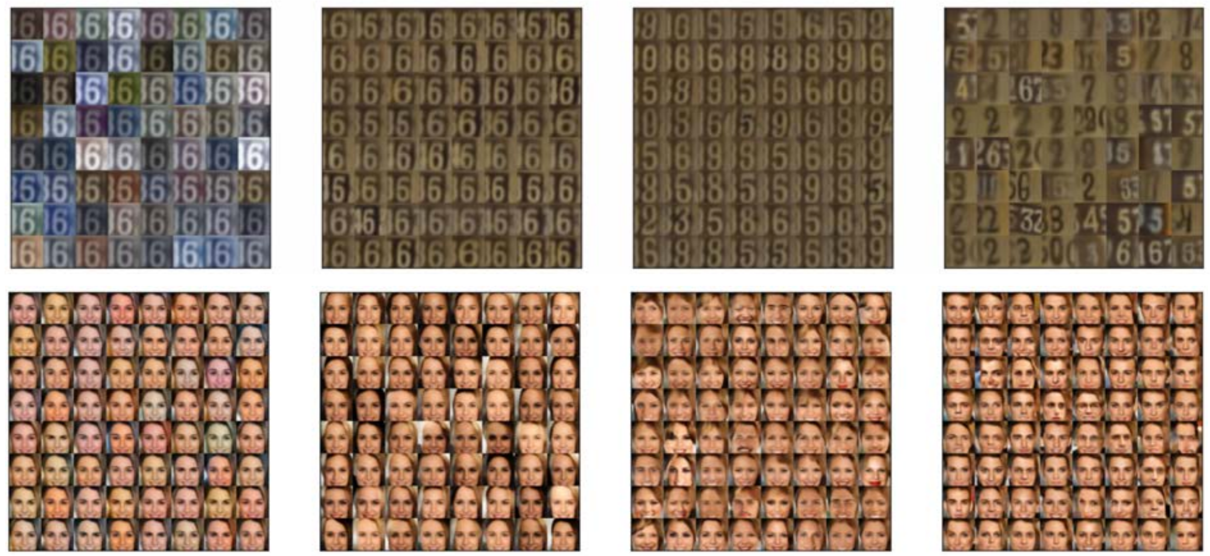
"variational auto-encoder"



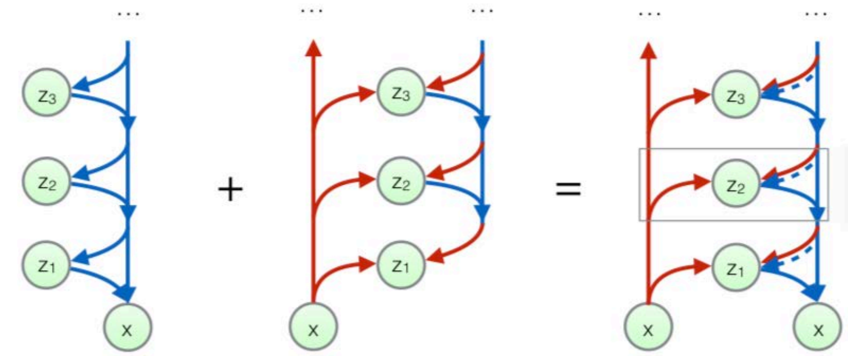
Learning to Infer, *Marino et al.*, 2017

<u>hierarchical</u> latent variable models



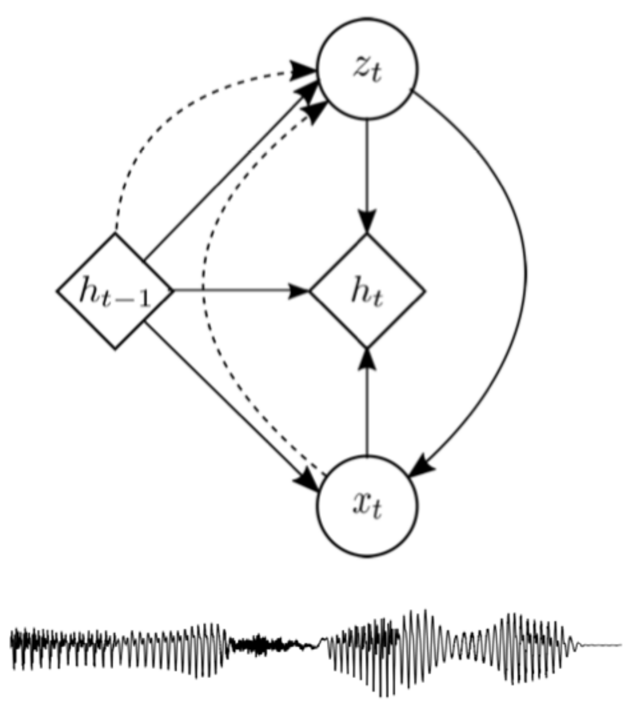


Learning Hierarchical Features from Generative Models, *Zhao et al.*, 2017

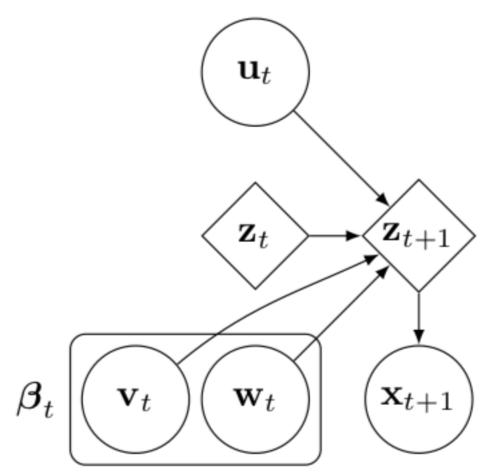


Improving Variational Inference with Inverse Auto-regressive Flow, *Kingma et al.*, 2016

<u>sequential</u> latent variable models

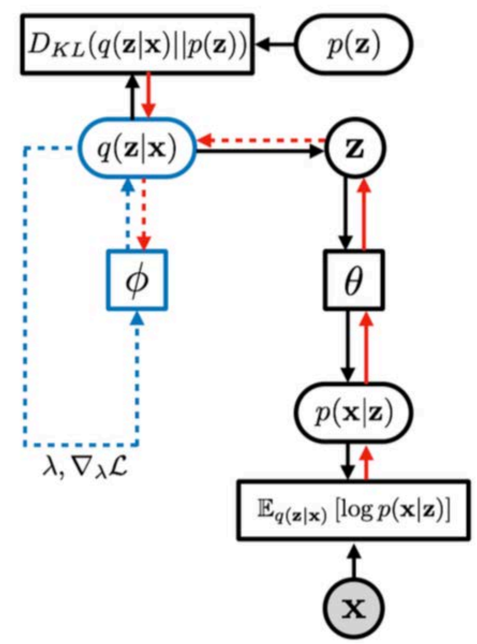


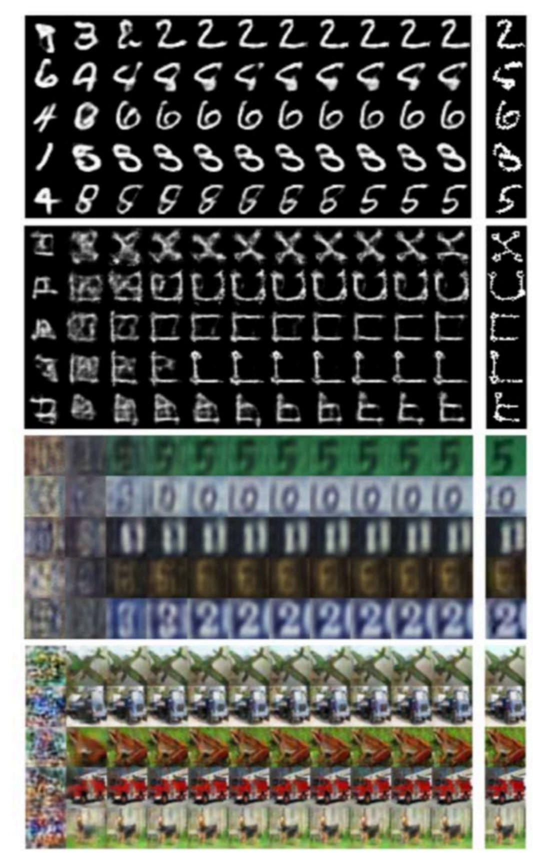
A Recurrent Latent Variable Model for Sequential Data, *Chung et al.*, 2015



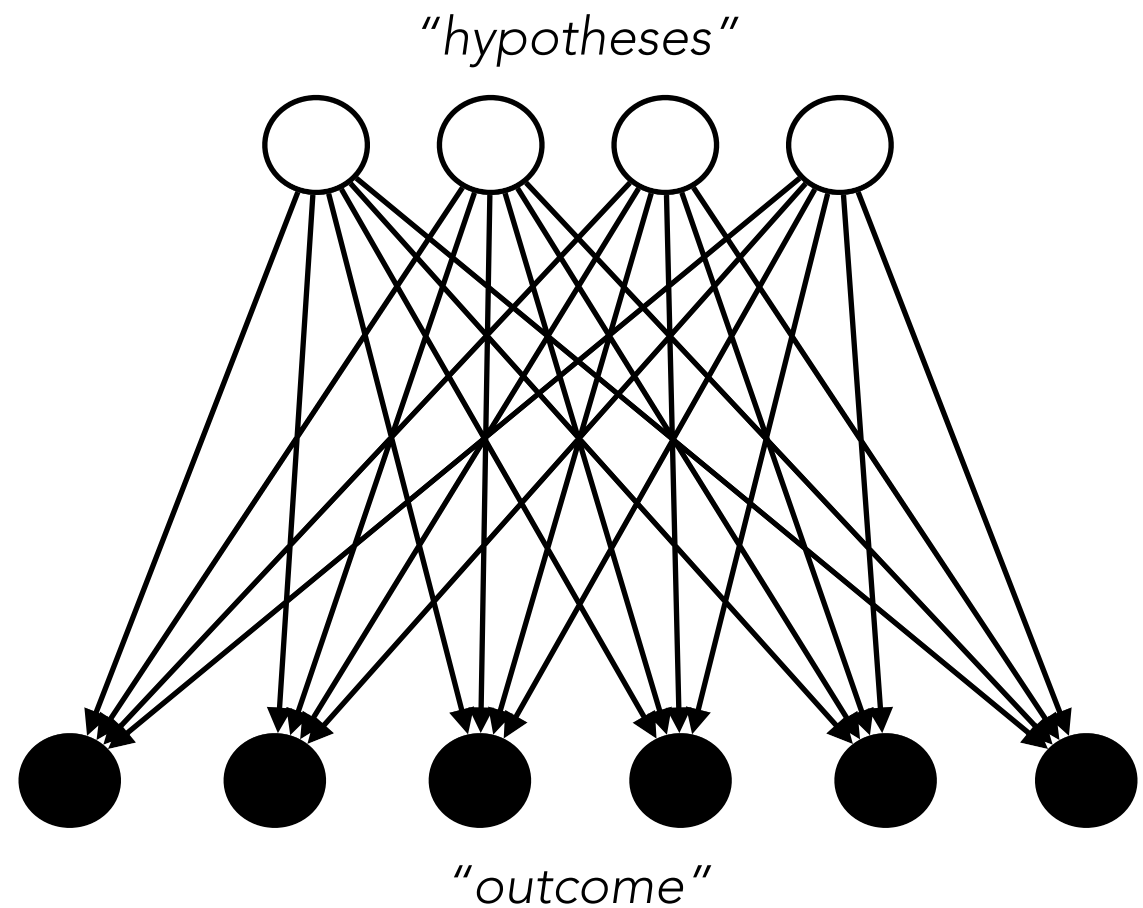
Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data, *Karl et al.*, 2016

iterative inference models





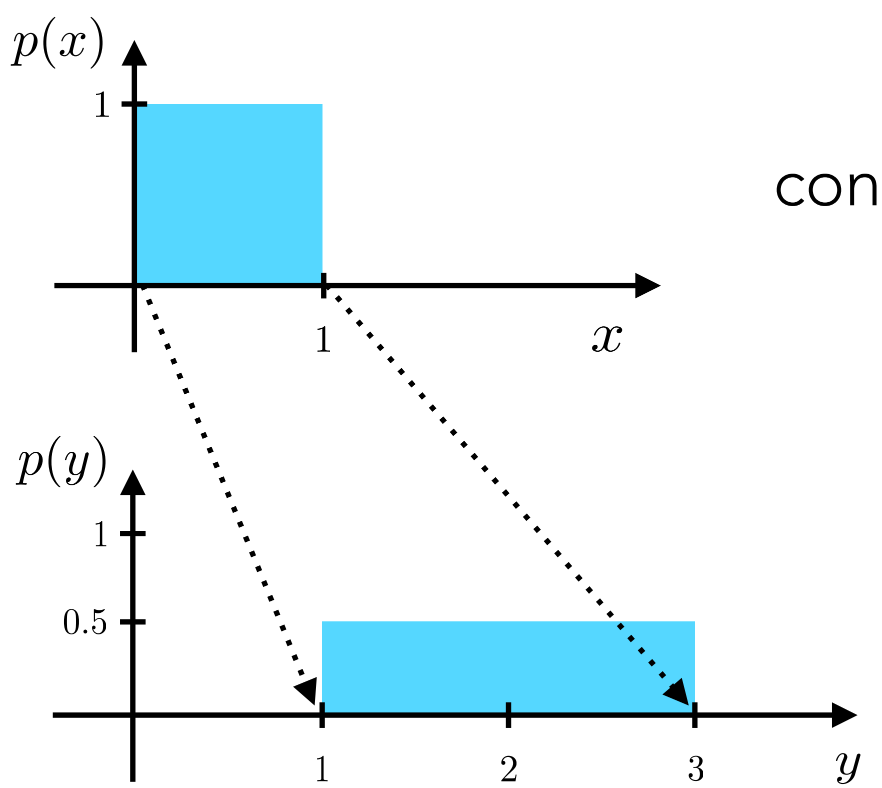
Learning to Infer, *Marino et al.*, 2017 introducing latent variables to a generative model generally makes evaluating the (log) likelihood intractable



need to consider all possible "hypotheses"
to evaluate (marginal) likelihood of the "outcome"

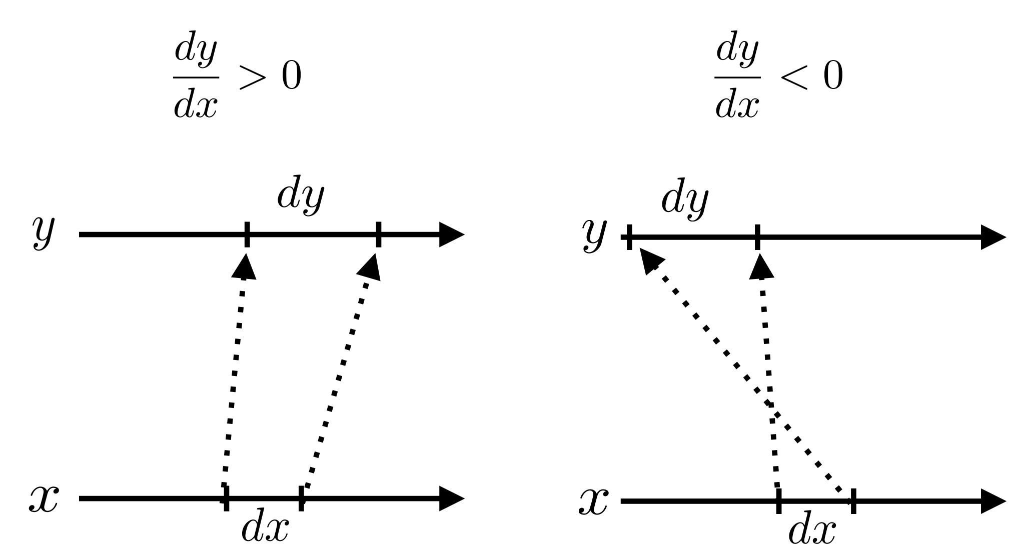
change of variables

under certain conditions, we can use the *change of variables* formula
to exactly evaluate the log likelihood



consider a variable in one dimension: $x \sim \mathrm{Uniform}(0, 1)$

then let y be an *affine transformation* of x,
e.g. $y = 2x + 1$.



to conserve probability mass, $p(y) = p(x) \left| \frac{dx}{dy} \right|$

Normalizing Flows Tutorial, *Eric Jang*, 2018

change of variables

in higher dimensions, conservation of probability mass generalizes to

CHANGE OF VARIABLES FORMULA

$$p(\mathbf{y}) = p(\mathbf{x}) \left| \det \frac{d\mathbf{x}}{d\mathbf{y}} \right| = p(\mathbf{x}) \left| \det \mathbf{J}^{-1} \right|$$

where $\mathbf{J}$ is the Jacobian matrix of the transformation, $\mathbf{J} = \frac{d\mathbf{y}}{d\mathbf{x}}$

$\left| \det \mathbf{J}^{-1} \right|$ *expresses the local distortion in volume from the linear transformation*

"law of the unconscious statistician" (LOTUS)

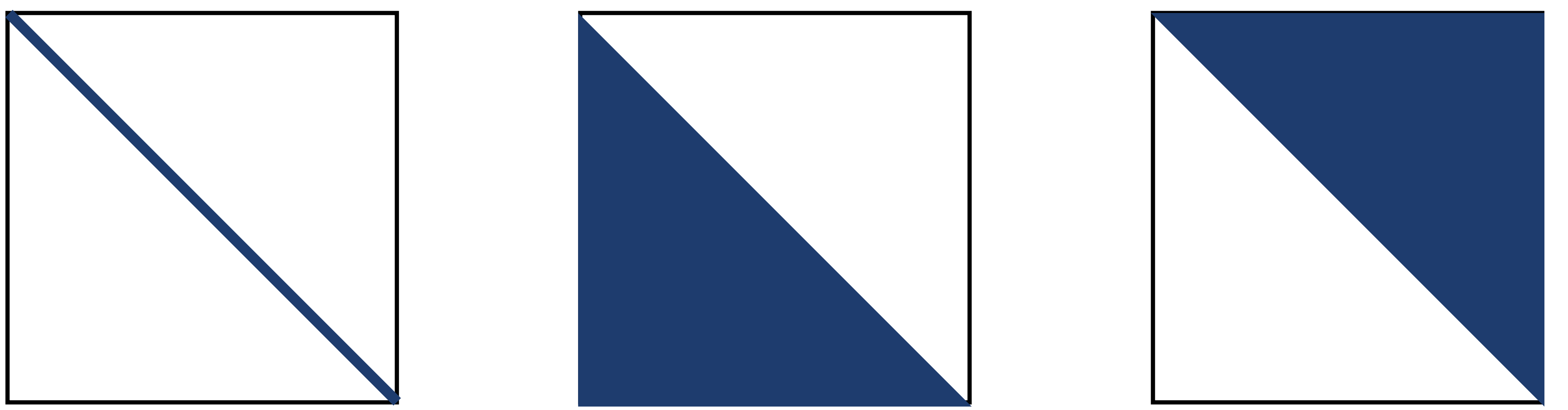
can evaluate the probability from one variable's distribution by evaluating the probability of a transformed variable and the volume transformation

for certain classes of transformations, this is *tractable* to evaluate

change of variables

to use the change of variables formula, we need to evaluate $\left|\det \mathbf{J}^{-1}\right|$

for an arbitrary $N \times N$ Jacobian matrix, this is worst case $O(N^3)$

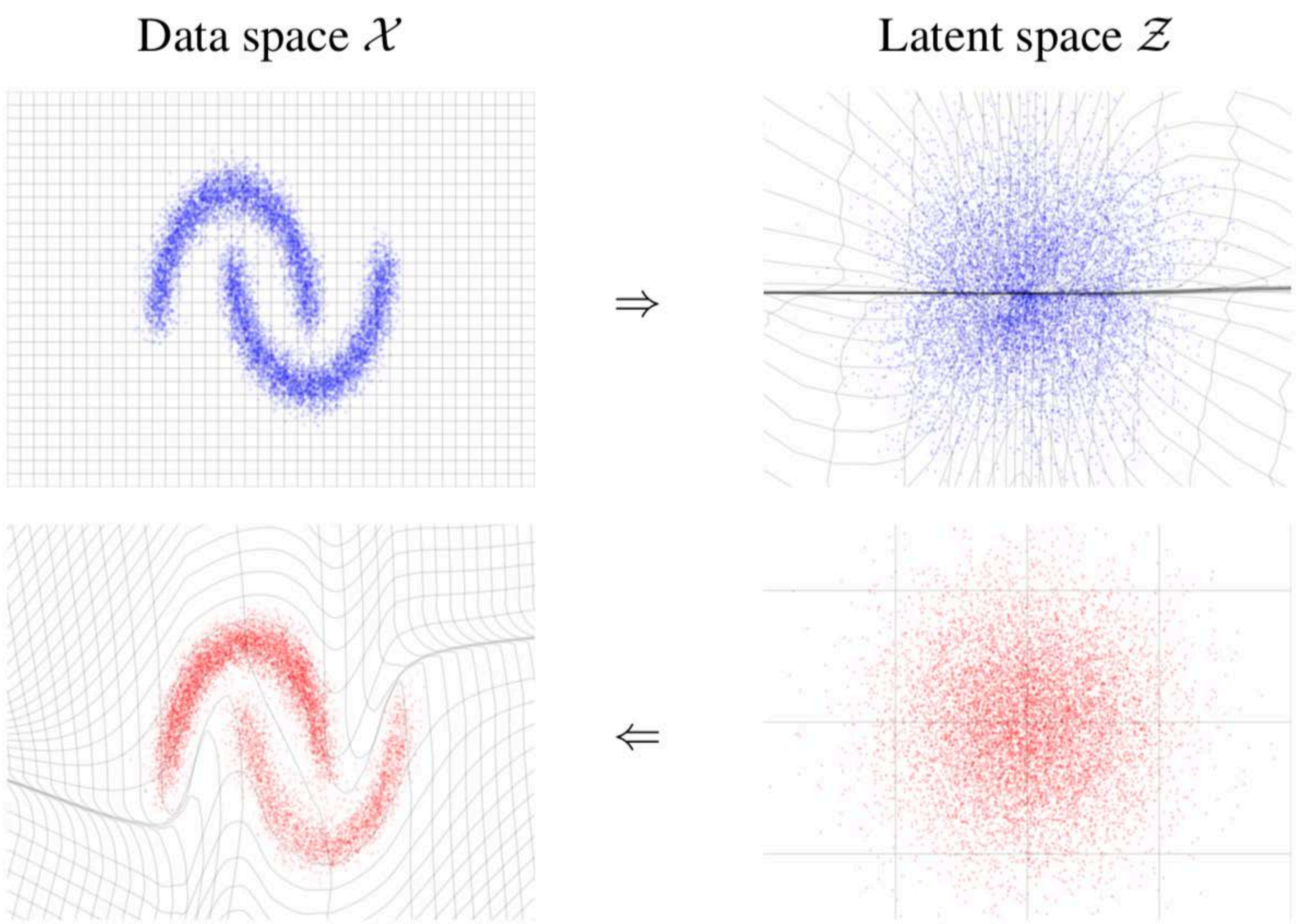


restricting the transformations to those with diagonal or triangular inverse Jacobians allows us to compute $\left|\det \mathbf{J}^{-1}\right|$ in $O(N)$.

⟶ *product of diagonal entries*

change of variables

can transform the data into a space that is easier to model

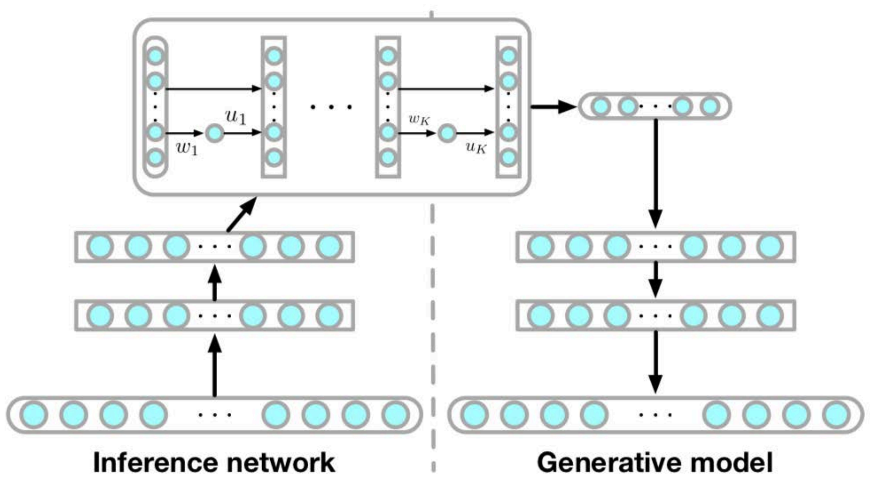


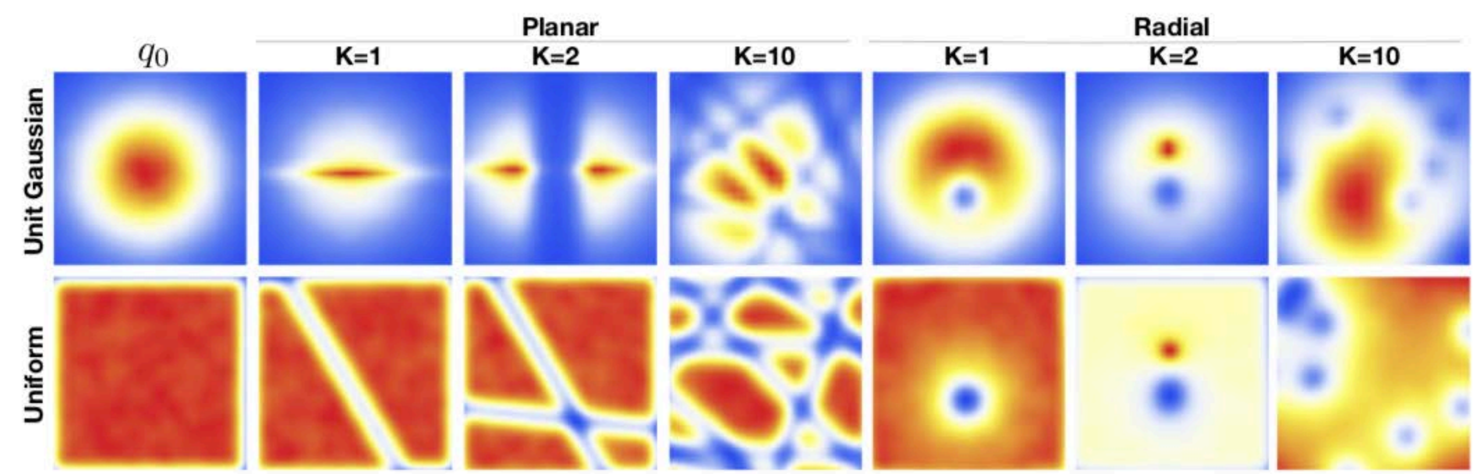
Density Estimation Using Real NVP, *Dinh et al.*, 2016

change of variables for variational inference: ***normalizing flows***

use more complex approximate posterior, but evaluate a simpler distribution

transform $q(\mathbf{z}|\mathbf{x})$

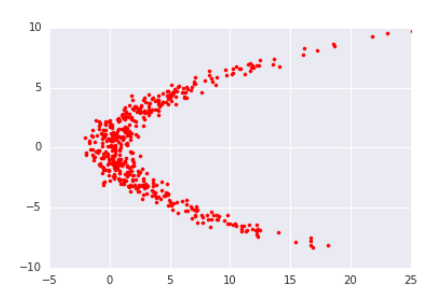


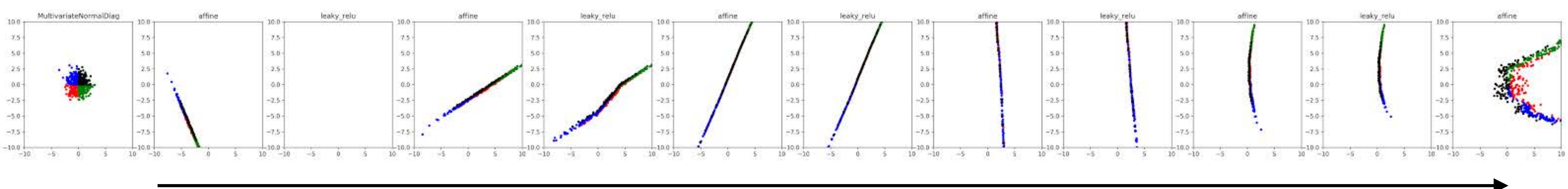


Variational Inference with Normalizing Flows, *Rezende & Mohamed, 2015*

chain together multiple transforms to get more expressive model

target distribution:





transforms

Normalizing Flows Tutorial, *Eric Jang*, 2018

transforms

additive coupling layer Dinh et al., 2014

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} + f(\mathbf{x}_{1:d})$$

planar flow Rezende & Mohamed, 2015

$$\mathbf{y} = \mathbf{x} + f(\mathbf{x}) \odot g(h(\mathbf{x})^\intercal \mathbf{x} + b(\mathbf{x}))$$

affine coupling layer Dinh et al., 2016

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(f(\mathbf{x}_{1:d})) + g(\mathbf{x}_{1:d})$$

inverse auto-regressive flow (IAF) Kingma et al., 2016

$$\mathbf{y} = \frac{\mathbf{x} - f(\mathbf{x})}{\exp(g(\mathbf{x}))}$$

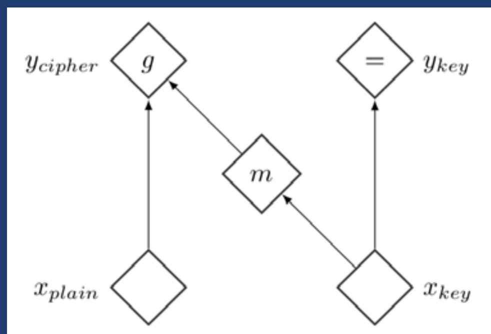
masked auto-regressive flow (MAF) Papamakarios et al., 2017

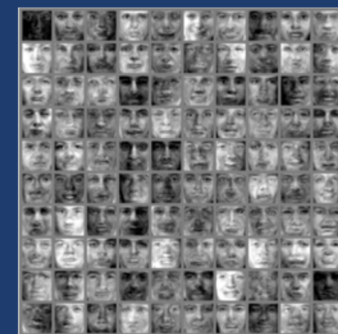
$$\mathbf{y} = \mathbf{x} \odot \exp(g(\mathbf{x})) + f(\mathbf{x})$$

recent work

NICE: Non-linear Independent Components Estimation

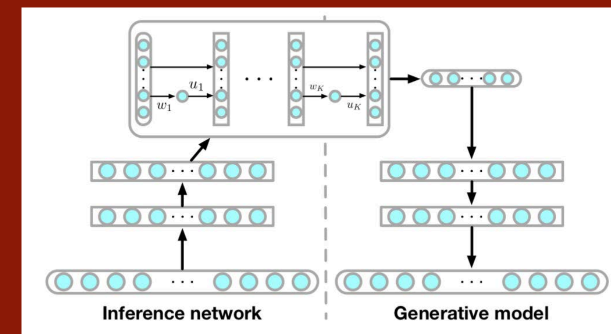
Dinh et al., 2014

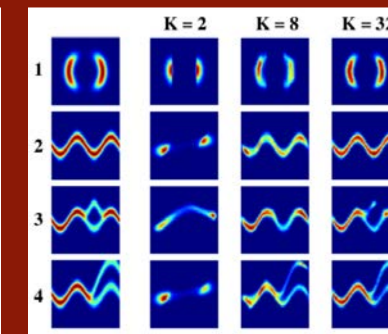


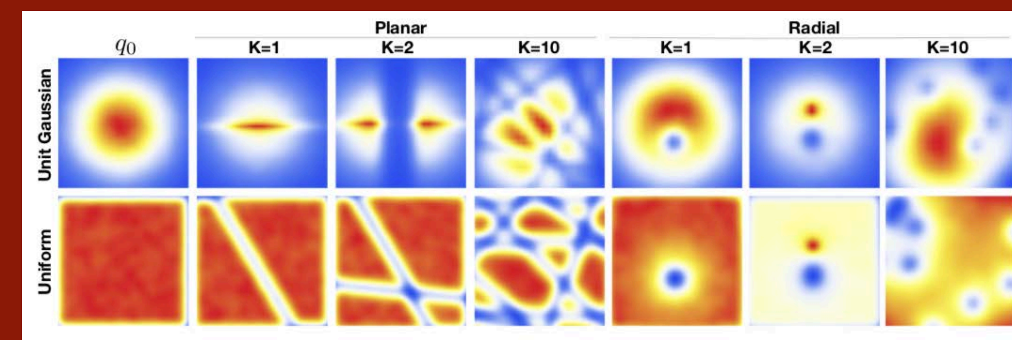


Variational Inference with Normalizing Flows

Rezende & Mohamed, 2015

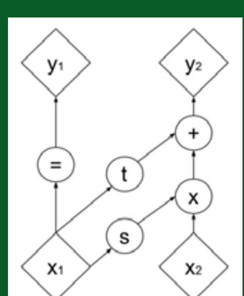


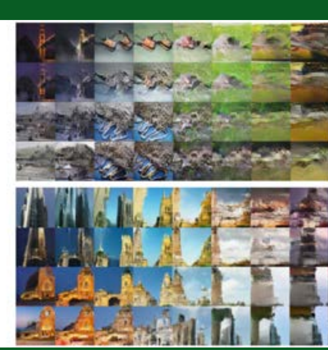




Density Estimation Using Real NVP

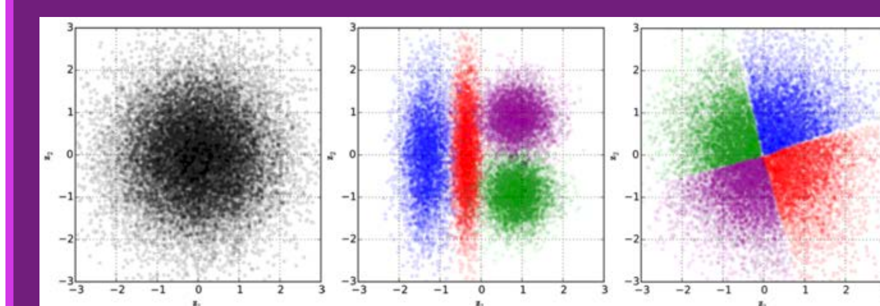
Dinh et al., 2016

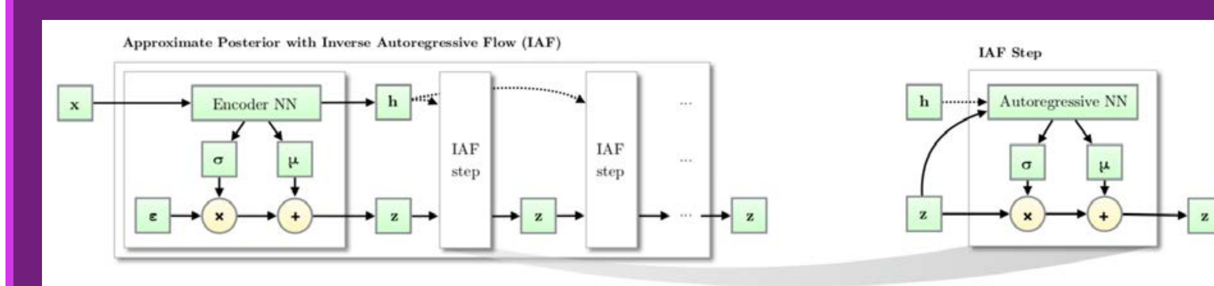




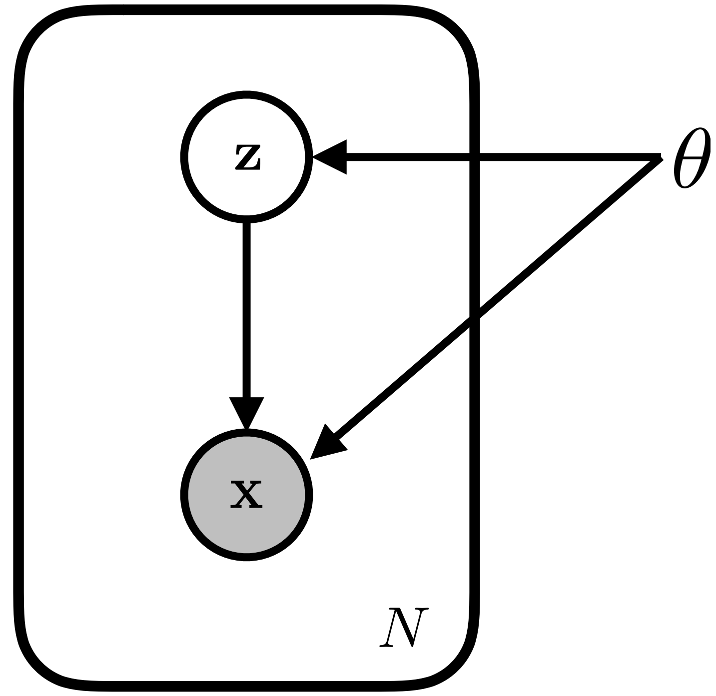
Improving Variational Inference with Inverse Autoregressive Flow

Kingma et al., 2016





recap: explicit latent variable models



model the data through *latent variables*

Pros

can capture abstract variables,
good for semi supervised learning

relatively fast sampling / training

theoretical foundations
from info. theory

Cons

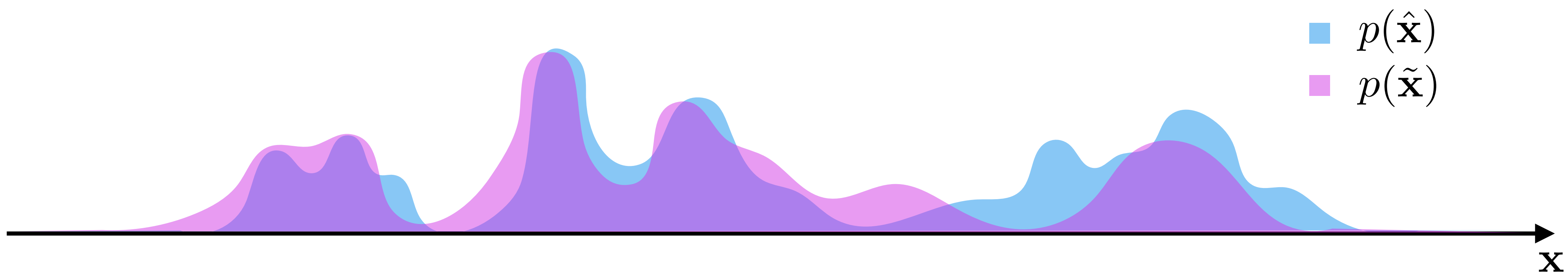
likelihood evaluation / inference
often intractable

requires additional assumptions
on latent variables

difficult to capture details

IMPLICIT LATENT VARIABLE MODELS

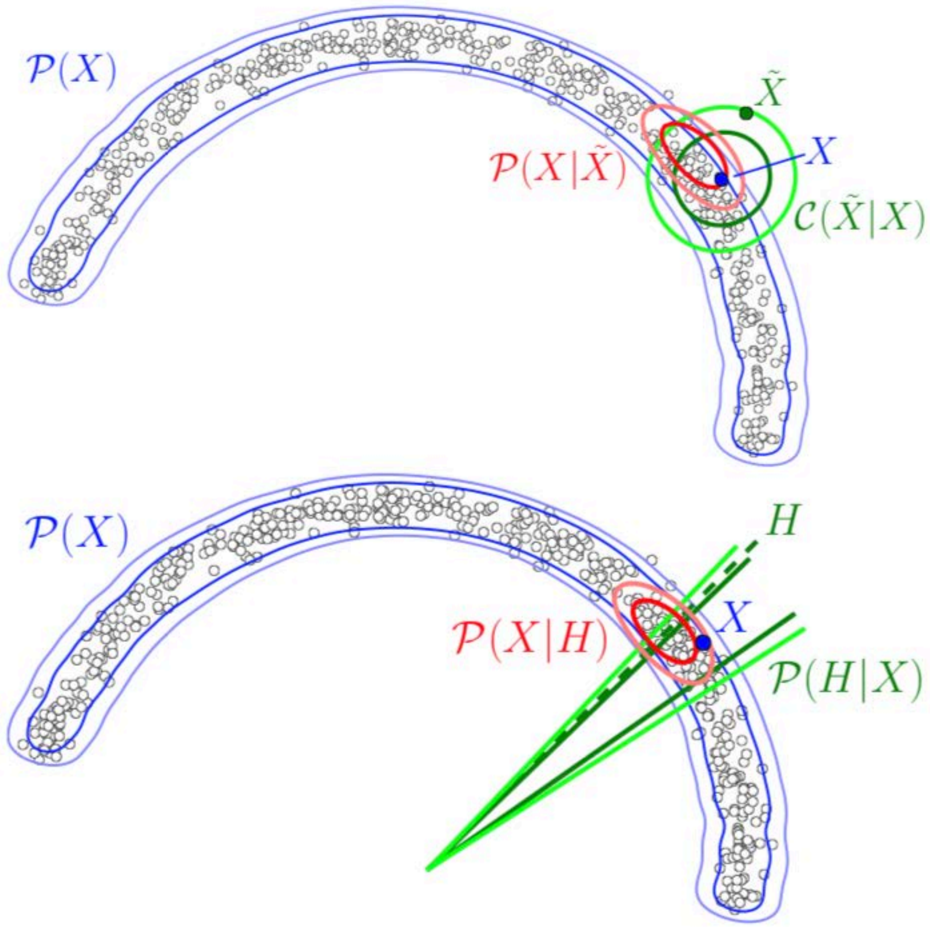
instead of using an *explicit* probability density, learn a model that defines an *implicit density*

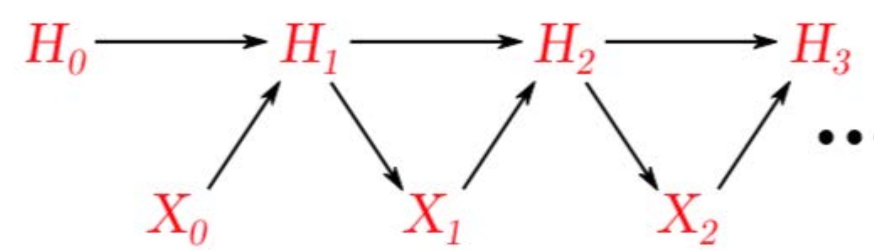


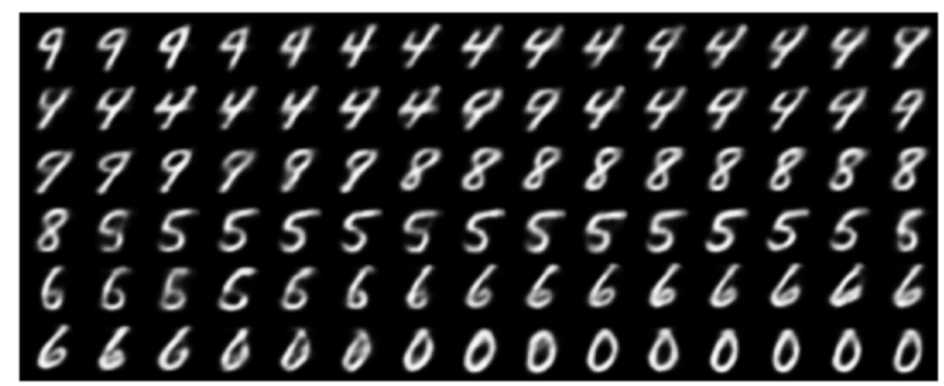
specify a stochastic procedure for generating the data that does not require an explicit likelihood evaluation

Learning in Implicit Generative Models, *Mohamed & Lakshminarayanan*, 2016

Generative Stochastic Networks (GSNs)



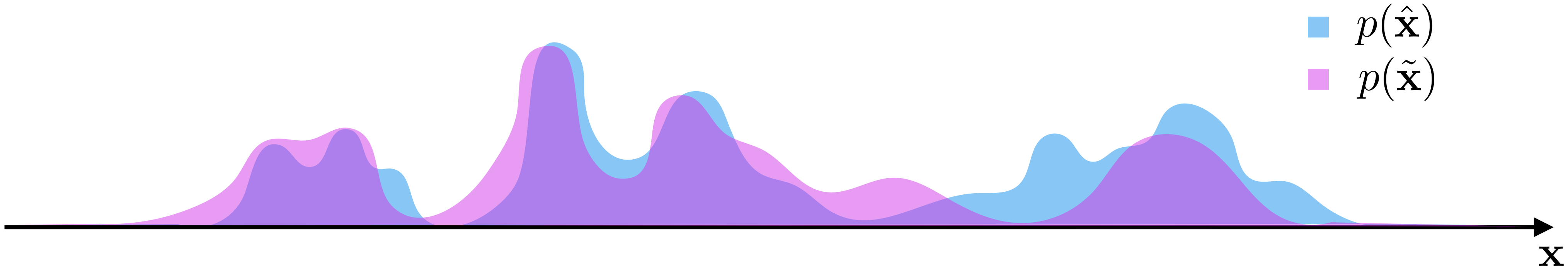




Deep Generative Stochastic Networks Trainable by Backprop, *Bengio et al.*, 2013

train an auto-encoder to learn Monte Carlo sampling transitions

the generative distribution is *implicitly* defined by this transition



estimate density ratio through *Bayesian two-sample test*

data distribution $p(\hat{\mathbf{x}})$

generated distribution $p(\tilde{\mathbf{x}})$

$$\frac{p(\hat{\mathbf{x}})}{p(\tilde{\mathbf{x}})} = \frac{p(\mathbf{x}|\text{data})}{p(\mathbf{x}|\text{gen.})}$$

$$\frac{p(\hat{\mathbf{x}})}{p(\tilde{\mathbf{x}})} = \frac{p(\text{data}|\mathbf{x})p(\mathbf{x})/p(\text{data})}{p(\text{gen.}|\mathbf{x})p(\mathbf{x})/p(\text{gen.})}$$

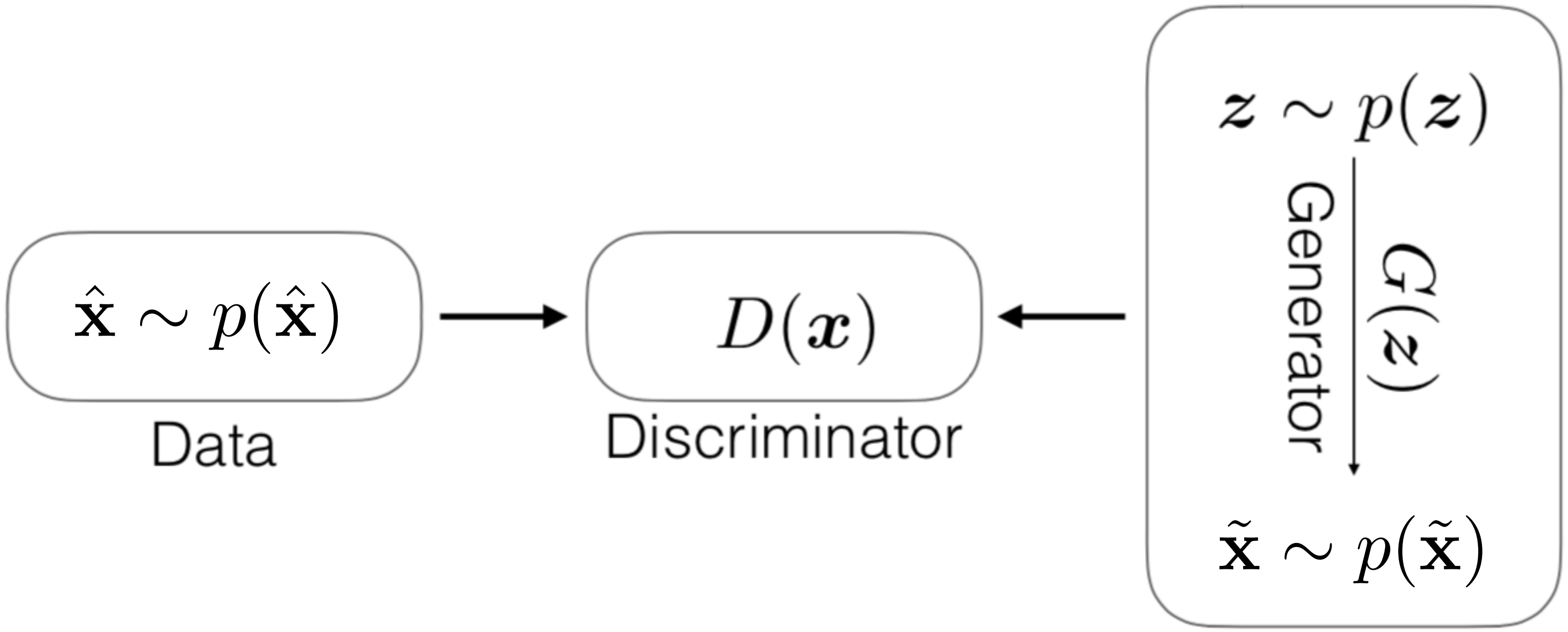
(Bayes' rule)

$$\frac{p(\hat{\mathbf{x}})}{p(\tilde{\mathbf{x}})} = \frac{p(\text{data}|\mathbf{x})}{p(\text{gen.}|\mathbf{x})}$$

(assuming equal dist. prob.)

density estimation becomes a sample discrimination task

Generative Adversarial Networks (GANs)



learn the discriminator:

$$p(\mathrm{data}|\mathbf{x}) = D(\mathbf{x}) \qquad p(\mathrm{gen.}|\mathbf{x}) = 1 - D(\mathbf{x})$$

Bernoulli outcome: $y \in \{\mathrm{data}, \mathrm{gen.}\}$

$$\log p(y|\mathbf{x}) = \log D(\hat{\mathbf{x}}) + \log(1 - D(\tilde{\mathbf{x}}))$$

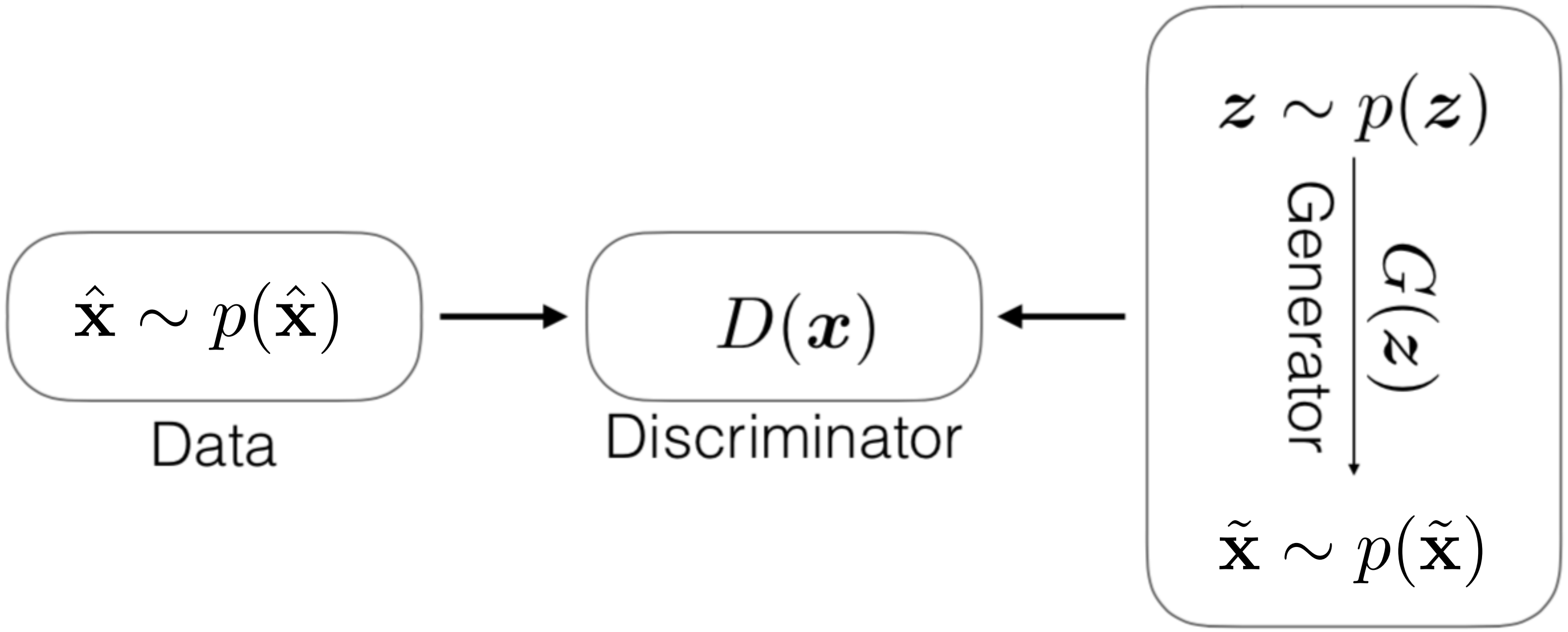
two-sample criterion:

$$\min_G \max_D \mathbb{E}_{p(\hat{\mathbf{x}})} \left[\log D(\hat{\mathbf{x}})\right] + \mathbb{E}_{p(\tilde{\mathbf{x}})} \left[\log(1 - D(\tilde{\mathbf{x}}))\right]$$

Goodfellow, 2016

Mohamed, 2016

Generative Adversarial Networks (GANs)



two-sample criterion:

$$\min_G \max_D \mathbb{E}_{p(\hat{\mathbf{x}})} \left[\log D(\hat{\mathbf{x}})\right] + \mathbb{E}_{p(\tilde{\mathbf{x}})} \left[\log(1 - D(\tilde{\mathbf{x}}))\right]$$

in practice:

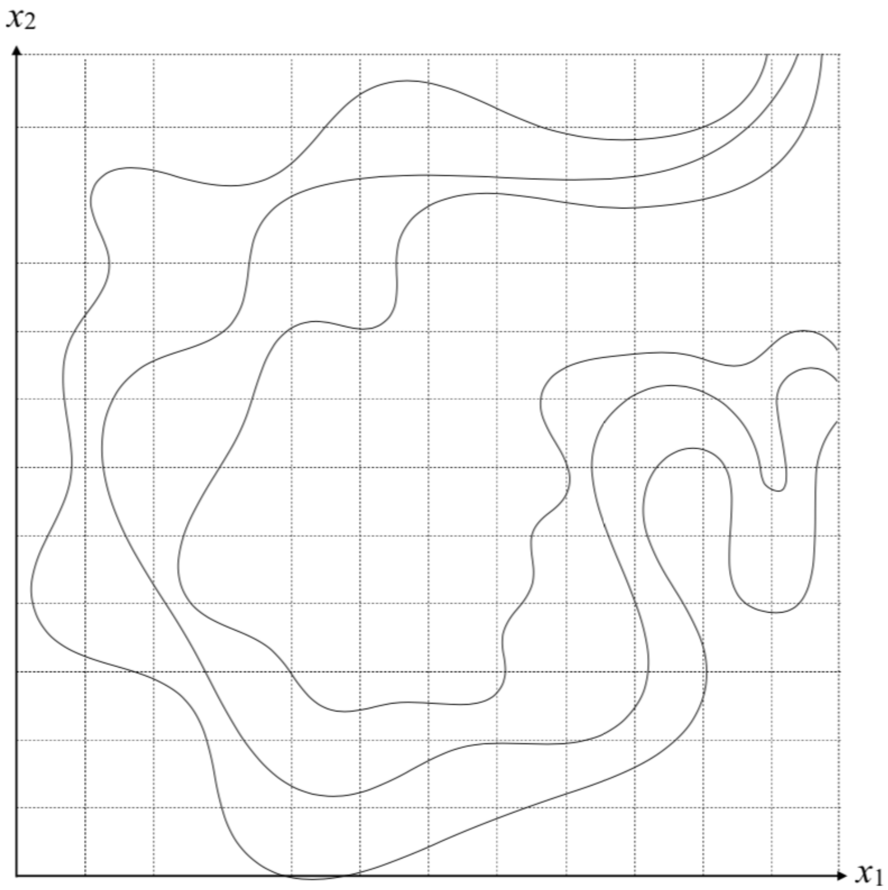
$$\max_D \mathbb{E}_{p(\hat{\mathbf{x}})} \left[\log D(\hat{\mathbf{x}})\right] + \mathbb{E}_{p(\tilde{\mathbf{x}})} \left[\log(1 - D(\tilde{\mathbf{x}}))\right]$$

$$\max_G \mathbb{E}_{p(\tilde{\mathbf{x}})} \left[\log D(\tilde{\mathbf{x}})\right]$$

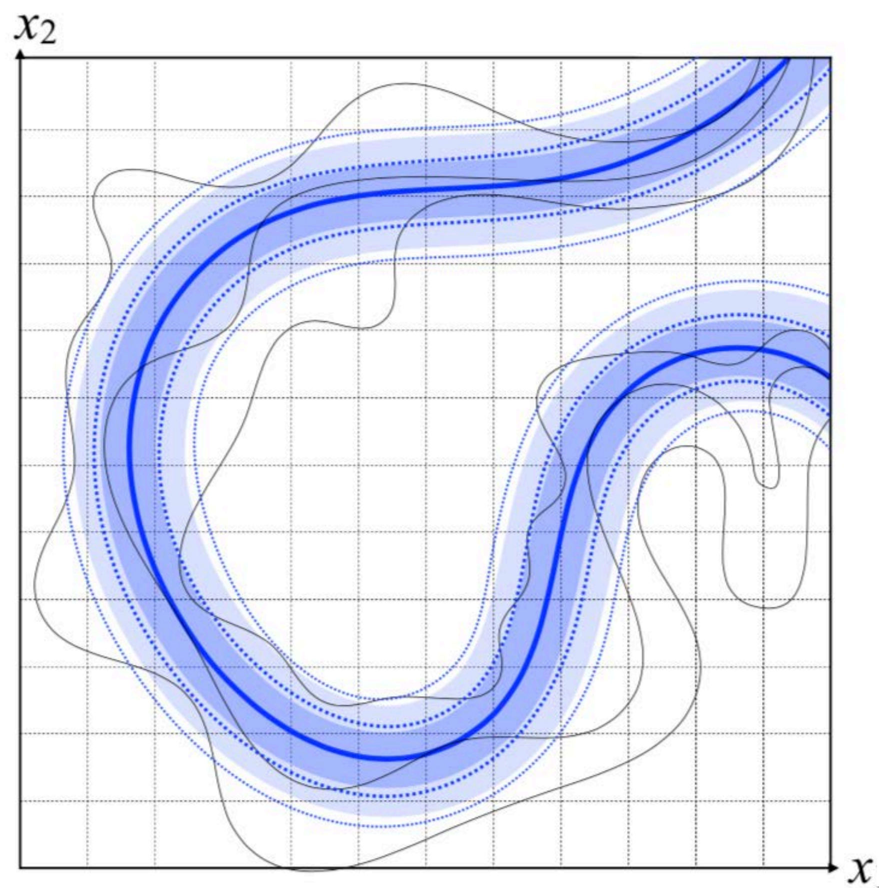
Goodfellow, 2016

Generative Adversarial Networks, *Goodfellow et al.*, 2014

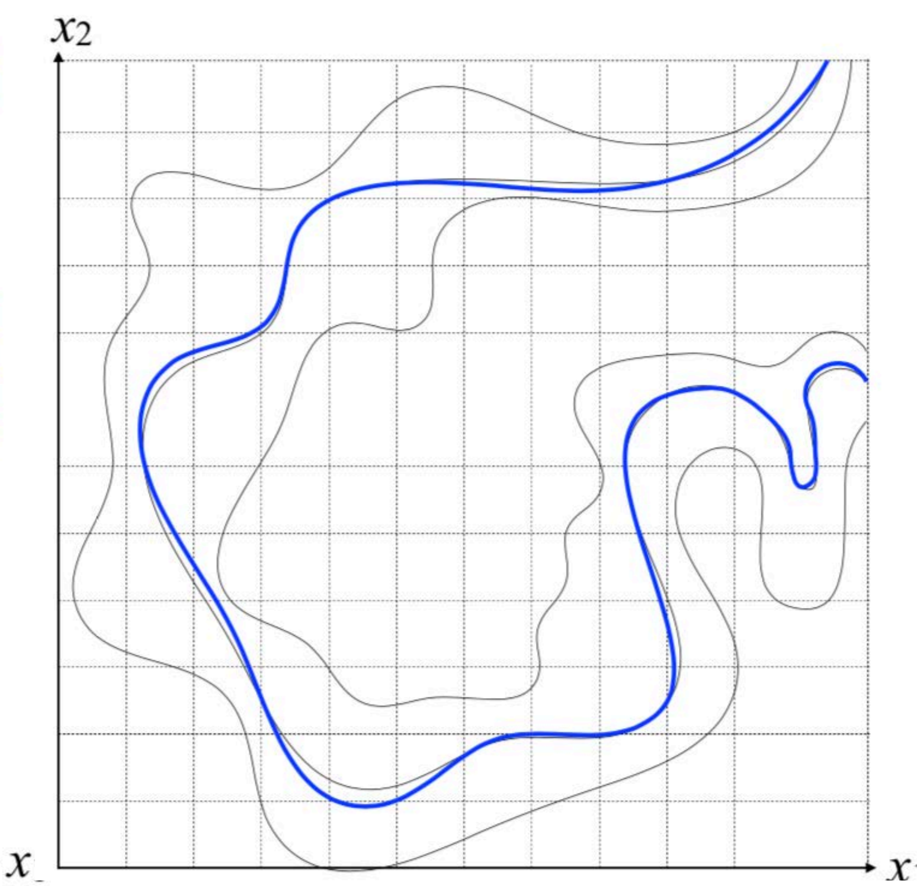
interpretation



data manifold



explicit model



implicit model

explicit models tend to cover the entire data manifold, but are constrained

implicit models tend to capture part of the data manifold, but can neglect other parts

→ *"mode collapse"*

Generative Adversarial Networks (GANs)

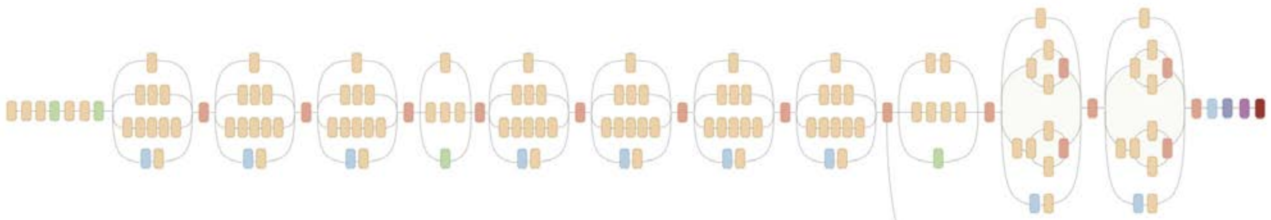
GANs can be difficult to optimize

Improved Training of Wasserstein GANs, Gulrajani et al., 2017

evaluation

without an explicit likelihood, it is difficult to quantify the performance

inception score



use a pre-trained Inception v3 model to quantify class and distribution entropy

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{p(\tilde{\mathbf{x}})} D_{KL}(p(y|\tilde{\mathbf{x}})||p(y))\right)$$

$p(y|\tilde{\mathbf{x}})$ is the class distribution for a given image

$\longrightarrow$ should be highly peaked (low entropy)

$p(y) = \int p(y|\tilde{\mathbf{x}}) d\tilde{\mathbf{x}}$ is the marginal class distribution

$\longrightarrow$ want this to be uniform (high entropy)

Improved Techniques for Training GANs, *Salimans et al.*, 2016
A Note on the Inception Score, *Barratt & Sharma*, 2018

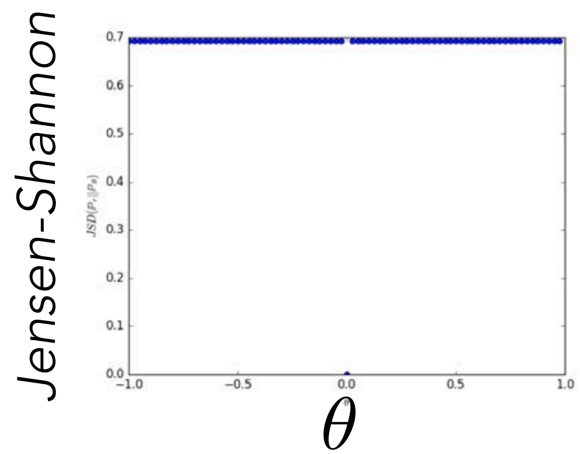
extensions: Wasserstein GAN

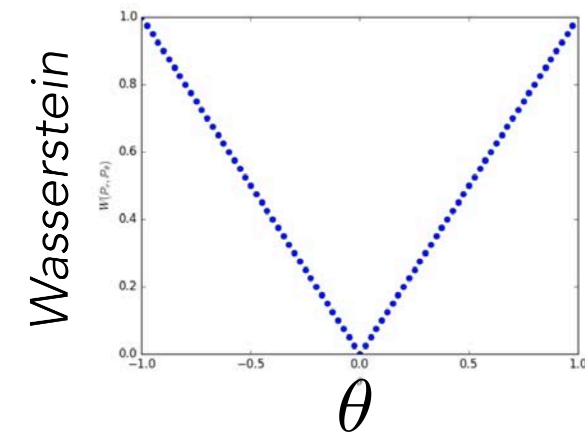
under an *"ideal"* discriminator, the generator minimizes the Jensen-Shannon divergence

$$D_{JS}(p(\hat{\mathbf{x}})||p(\tilde{\mathbf{x}})) = \frac{1}{2}D_{KL}(p(\hat{\mathbf{x}})||\frac{1}{2}(p(\hat{\mathbf{x}}) + p(\tilde{\mathbf{x}}))) + \frac{1}{2}D_{KL}(p(\tilde{\mathbf{x}})||\frac{1}{2}(p(\hat{\mathbf{x}}) + p(\tilde{\mathbf{x}})))$$

however, this metric can be **discontinuous**, making it difficult to train

θ is a gen. model parameter





can instead use the Wasserstein (Earth Mover's) distance (which is continuous and diff. almost everywhere):

$$W(p(\hat{\mathbf{x}}), p(\tilde{\mathbf{x}})) = \inf_{\gamma \in \Pi(p(\hat{\mathbf{x}}), p(\tilde{\mathbf{x}}))} \mathbb{E}_{(\hat{\mathbf{x}}, \tilde{\mathbf{x}}) \sim \gamma} \left[||\hat{\mathbf{x}} - \tilde{\mathbf{x}}|| \right]$$

think of it as the "minimum cost of transporting points between two distributions"

intractable to actually evaluate Wasserstein distance, but by constraining the discriminator, can evaluate

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{p(\hat{\mathbf{x}})} \left[D(\hat{\mathbf{x}}) \right] - \mathbb{E}_{p(\tilde{\mathbf{x}})} \left[D(\tilde{\mathbf{x}}) \right]$$

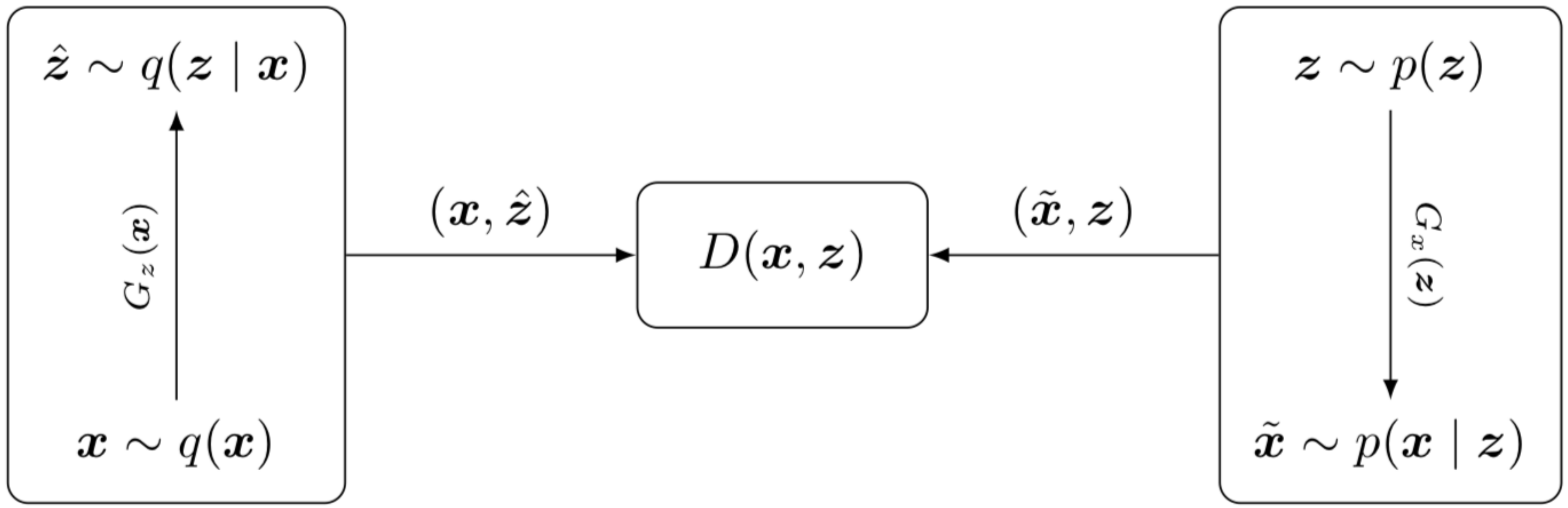
$\mathcal{D}$ is the set of Lipschitz functions, which can be enforced through weight clipping or gradient penalty

Wasserstein GANs, *Arjovsky et al.*, 2017

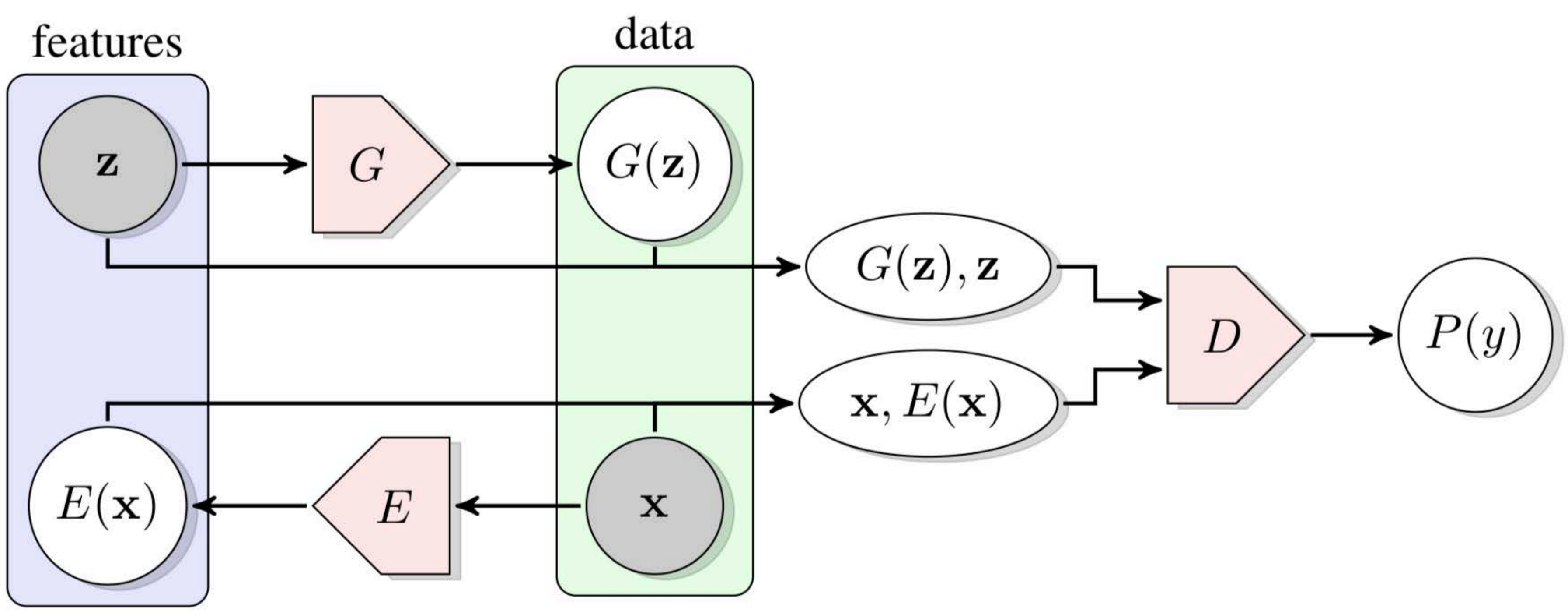
Improved Training of Wasserstein GANs, Gulrajani et al., 2017

extensions: inference

can we also learn to ***infer a latent representation****?*



Adversarially Learned Inference, *Dumoulin et al.*, 2017



Adversarial Feature Learning, *Donahue et al.*, 2017

applications

image to image translation

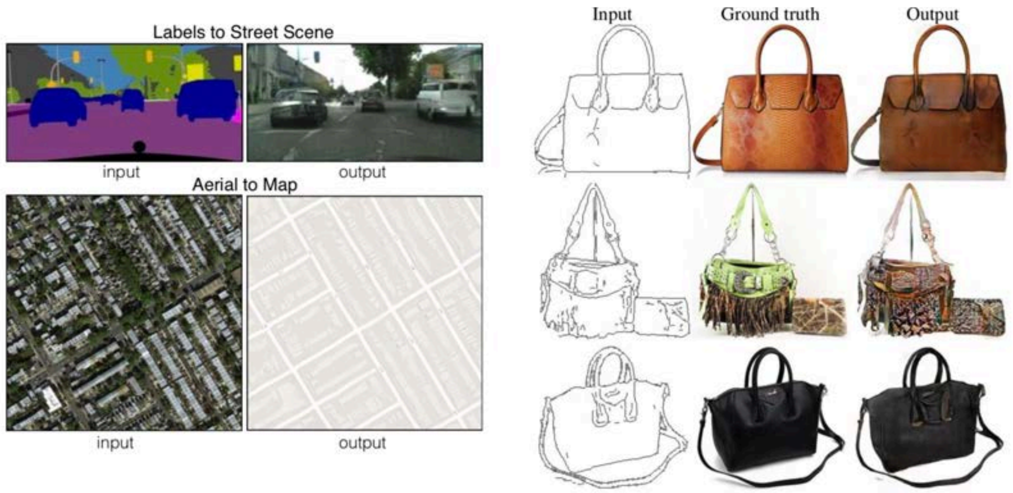
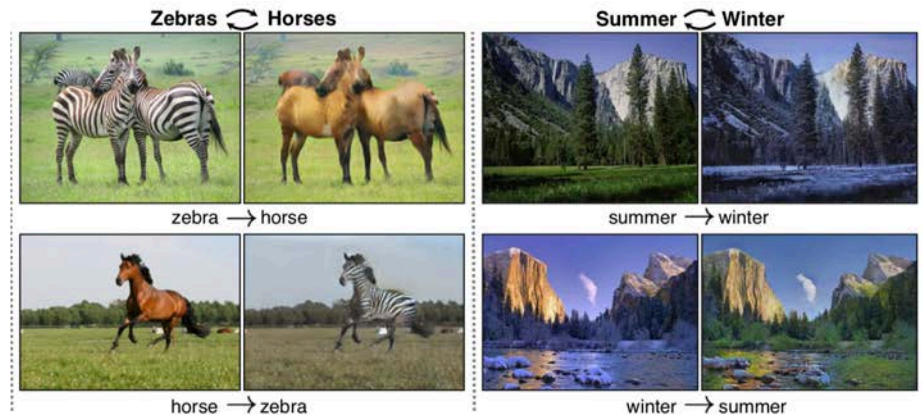
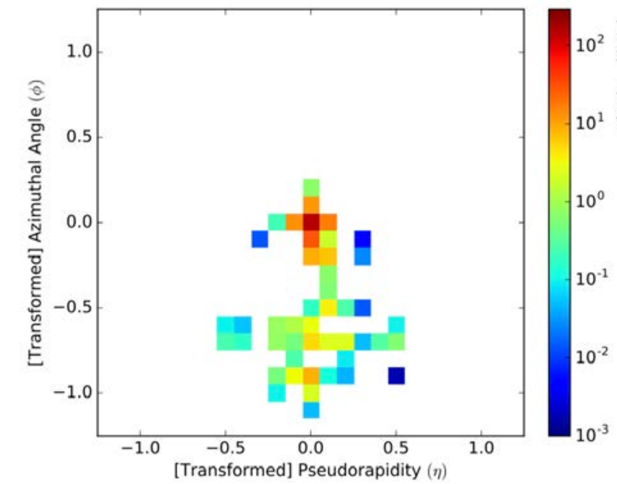


Image-to-Image Translation with Conditional Adversarial Networks, *Isola et al.*, 2016



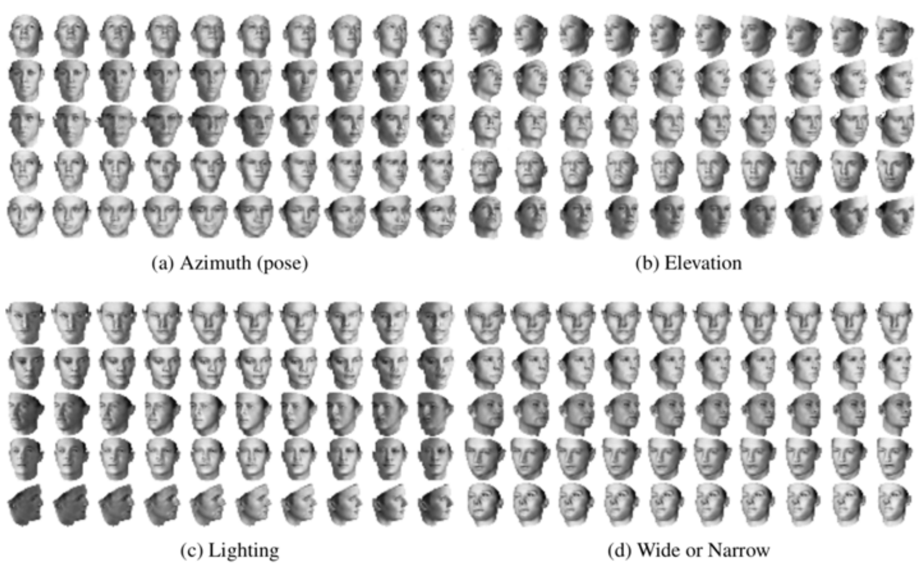
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, *Zhu et al.*, 2017

experimental simulation



Learning Particle Physics by Example, *de Oliveira et al.*, 2017

interpretable representations

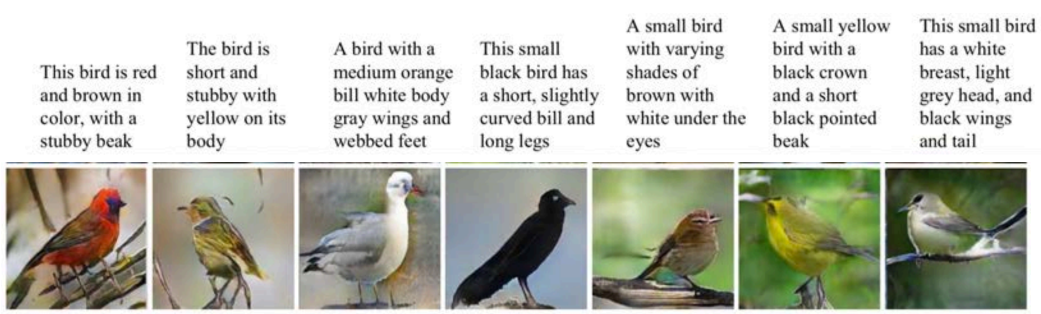


InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, *Chen et al.*, 2016

music synthesis

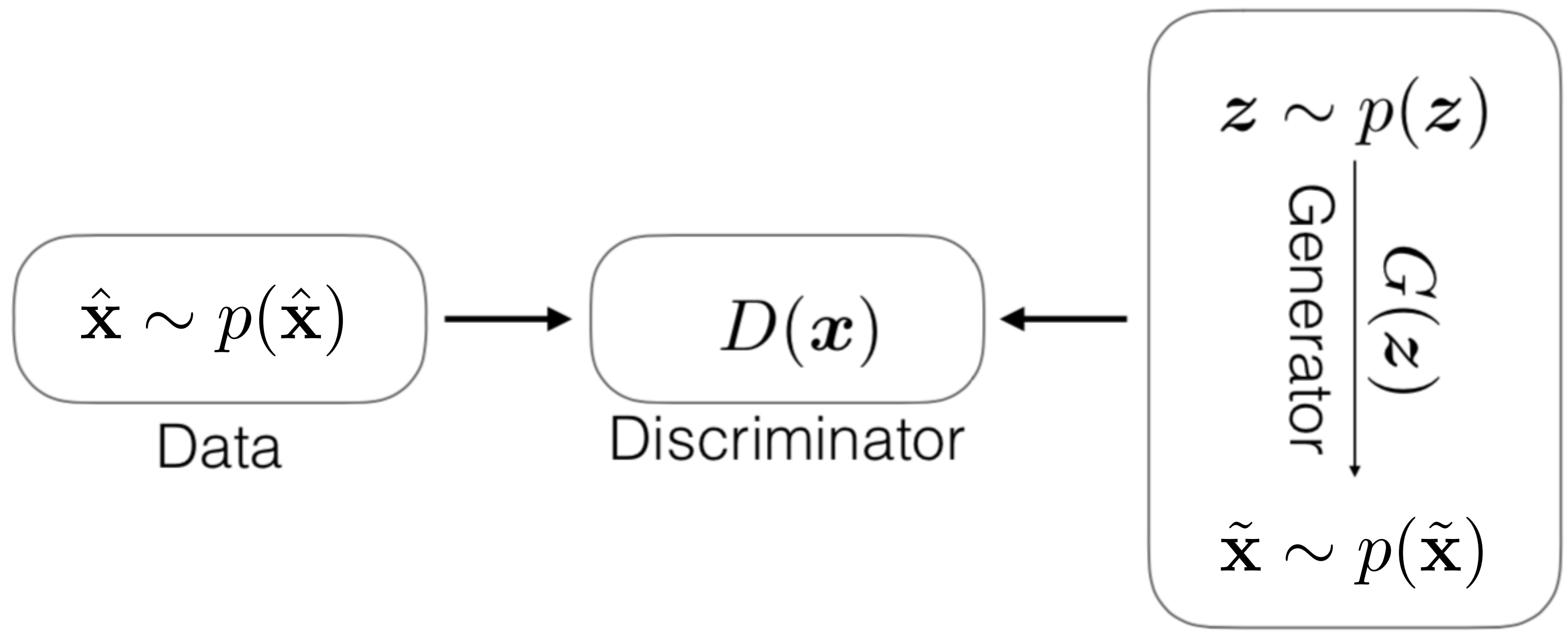
MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION, *Yang et al.*, 2017

text to image synthesis



StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, *Zhang et al.*, 2016

recap: implicit latent variable models



Pros

able to learn flexible models

requires fewer modeling assumptions

capable of learning latent representation

Cons

difficult to evaluate

sensitive, difficult to optimize

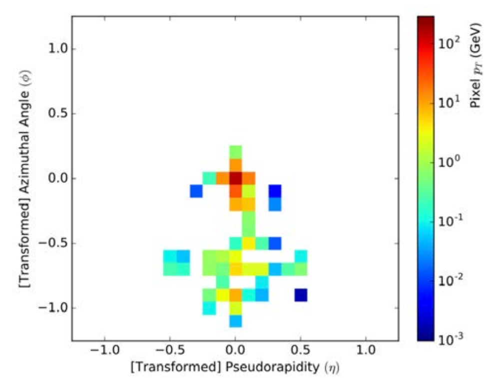
can be difficult to incorporate model assumptions

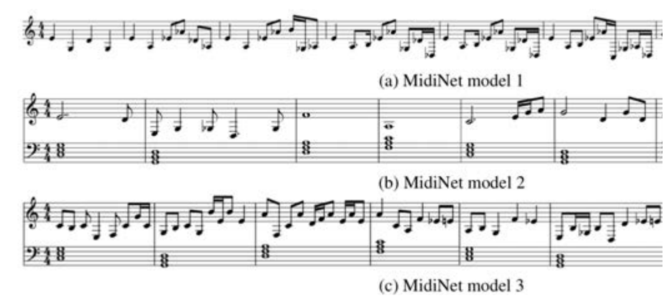
DISCUSSION

generative models: *what are they good for?*

generative models model the data distribution

1. can generate and simulate data





2. can extract structure from data

generative models: *what's next?*

applying generative models to new forms of data

incorporating generative models into *complementary* learning systems