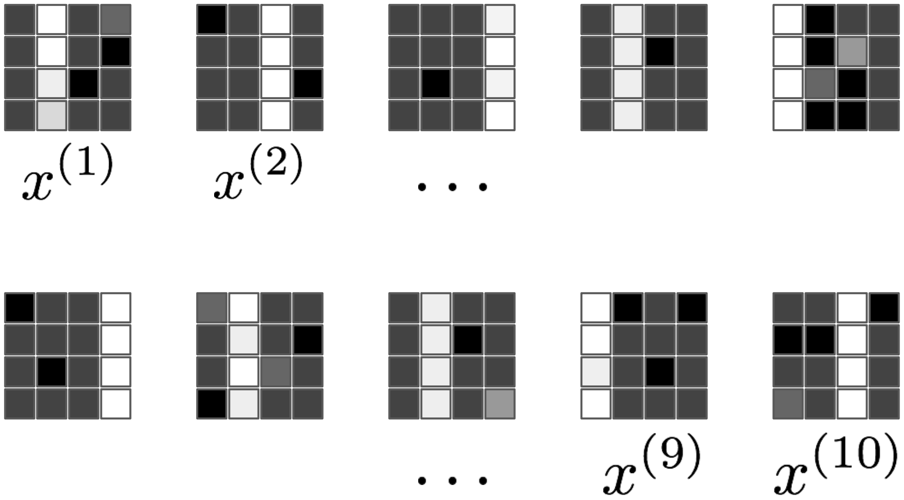
Deep Generative Models

Ulrich Paquet
DeepMind

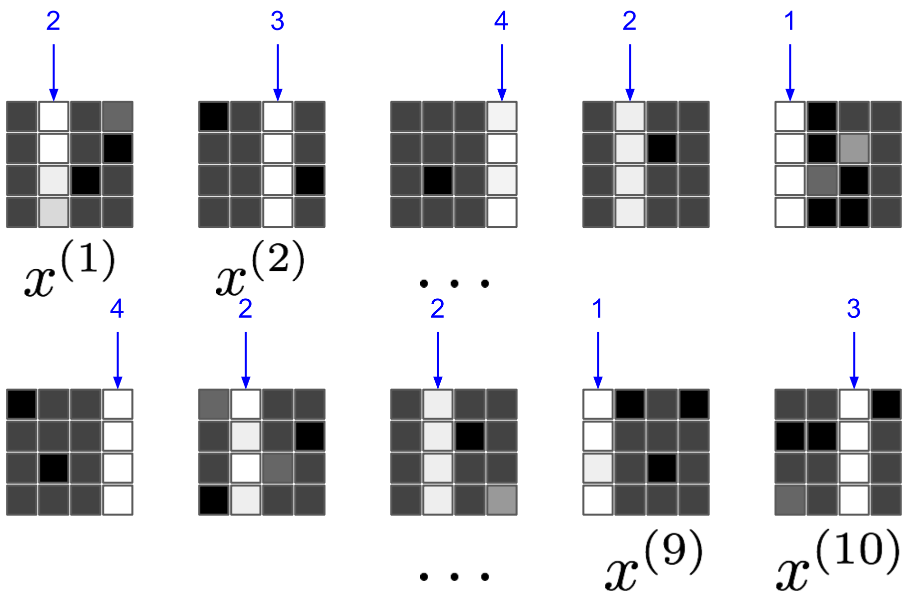
Version: 7 September 2017

2-minute exercise

Talk to your friend next to you, and tell him or her everything you can about this data set:



Data



Data manifold

We can capture most of the variability in the data through **one** number

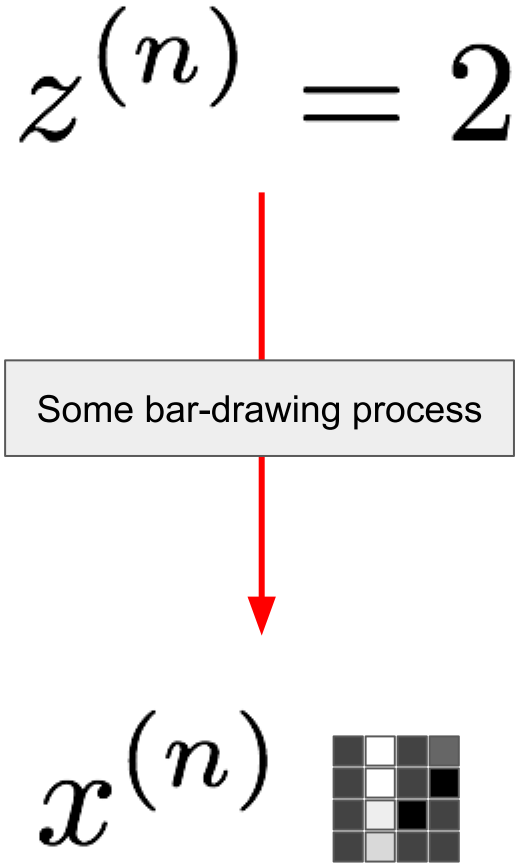
$$z^{(n)} = 1 \text{ or } 2, 3, 4$$

for each image *n*, even though each image is 16 dimensional

How?

How?

1. Take $z^{(n)} = 2$
2. Draw bar in column 2 of image
3. Et voila! You have $x^{(n)}$



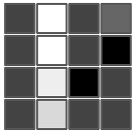
How?

1. Take $z^{(n)} = 2$
2. Draw bar in column 2 of image
3. Et voila! You have $x^{(n)}$

$$z^{(n)} = 2$$

Maybe some neural network, that takes z as input, and outputs a 16-dimensional vector x…?

$$x^{(n)}$$



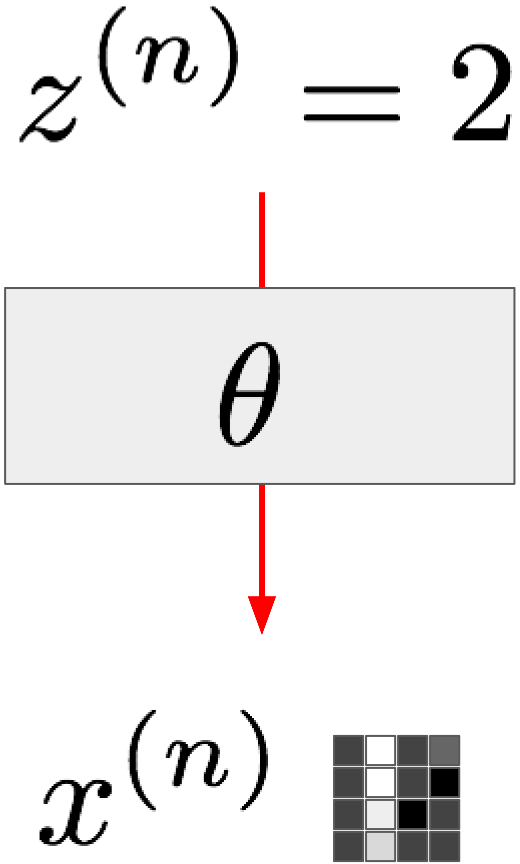
3-minute exercise

Write or draw a function (like a multi-layer perceptron) that takes $z \in \mathbb{R}$ and produces x

Is your input one-dimensional?

Is your output 16-dimensional?

Identify all the “tunable” parameters θ of your function



3-minute exercise

Write or draw a function (like a multi-layer perceptron) that takes $z \in \mathbb{R}$ and produces x

Is your input one-dimensional?

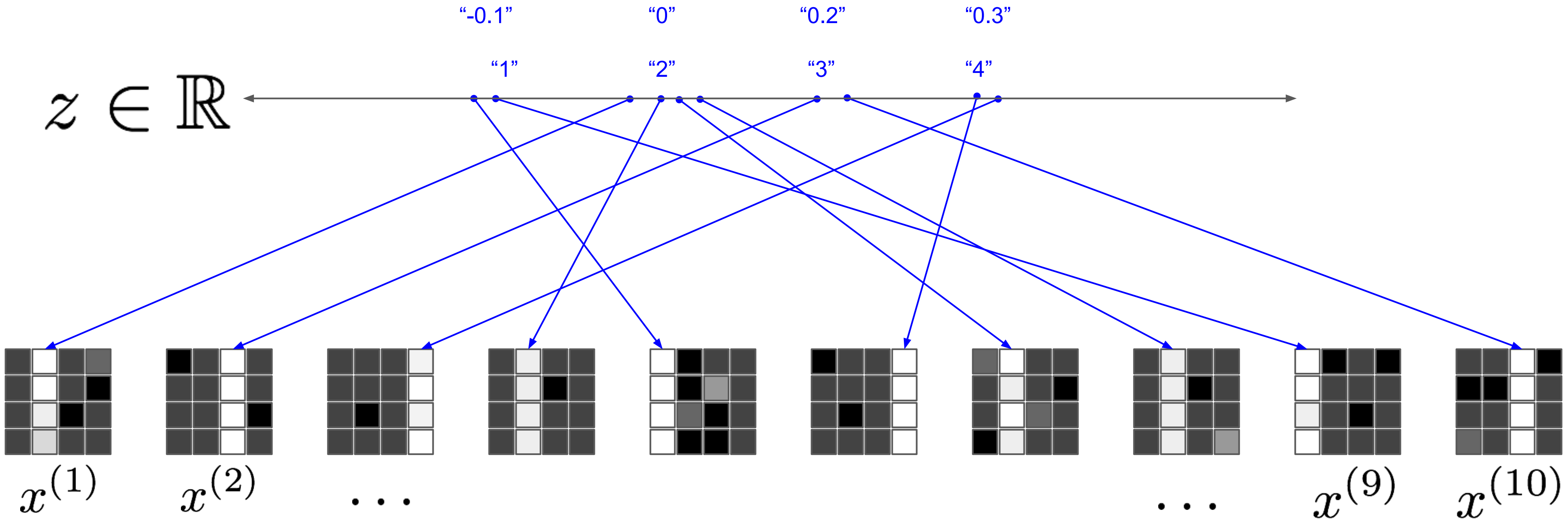
Is your output 16-dimensional?

Identify all the “tunable” parameters θ of your function

scratch space

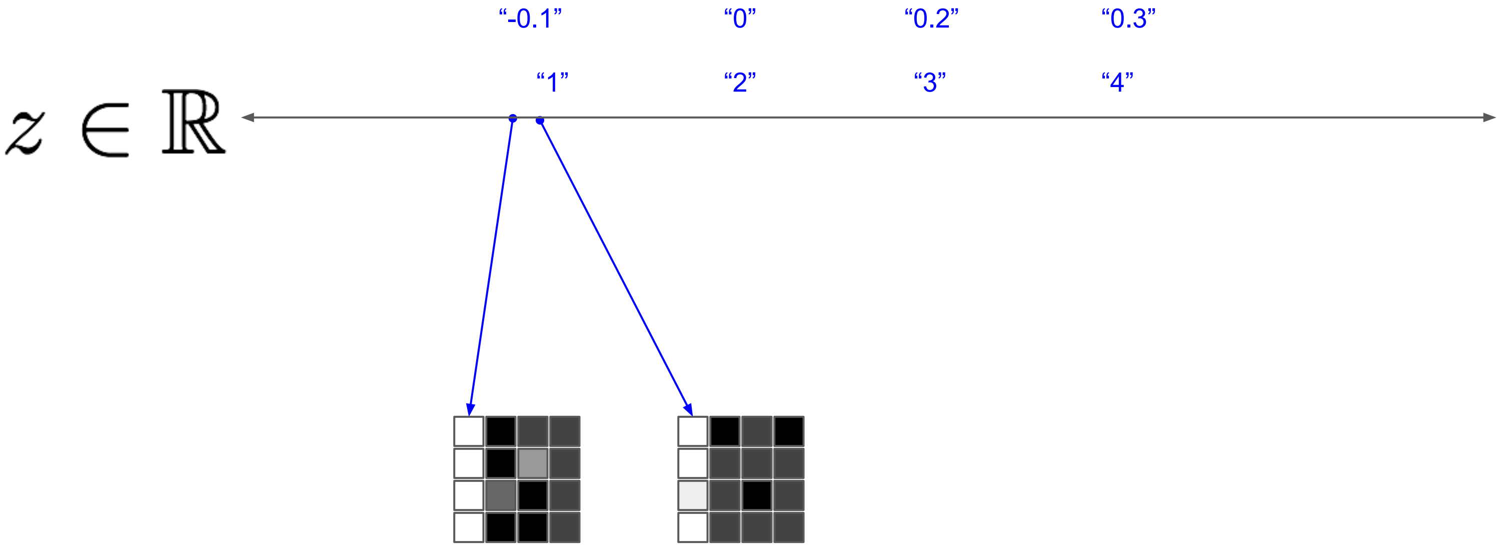
Data manifold

The 16-dimensional images live on a 1-dimensional manifold, plus some “noise”



...and noise

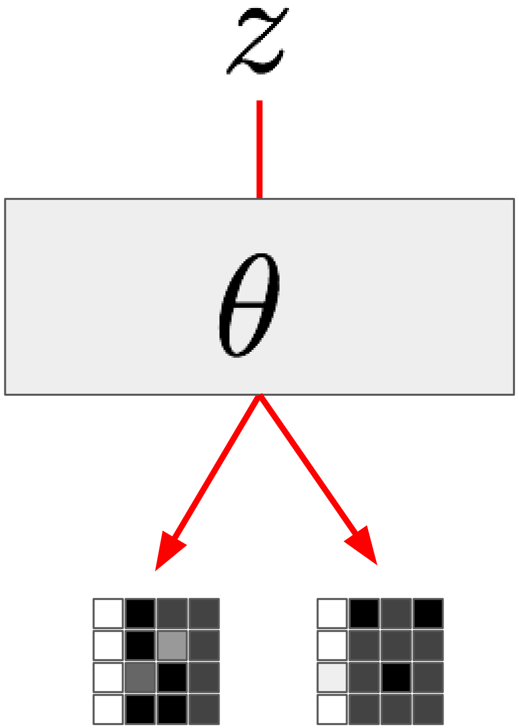
The 16-dimensional images live on a 1-dimensional manifold, plus some “noise”



3-minute exercise

Change your multi-layer perceptron to take z and produce a distribution over x

$$p_\theta(x|z)$$



3-minute exercise

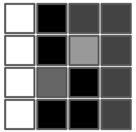
Change your multi-layer perceptron to take z and produce a distribution over x

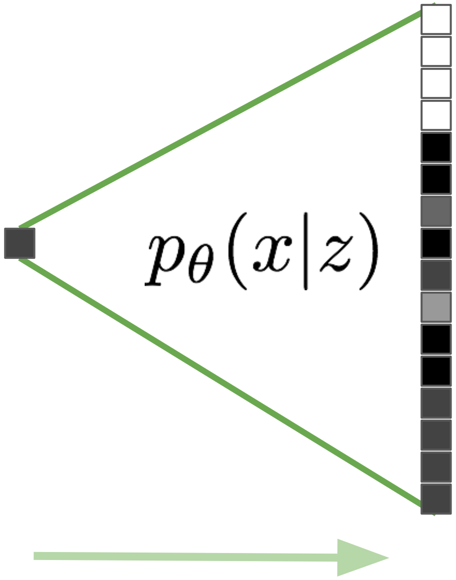
$$p_\theta(x|z)$$

scratch space

Decoder

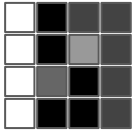
```python
def generative_network(z, ...):
    ...
    return bernoulli_logits    # for binary pixels
```

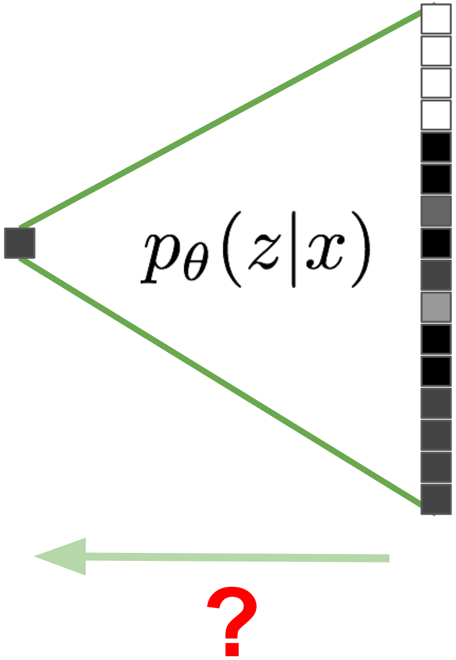




$$z \to x$$

Inference





Inversing our world

Two BIG problems to solve:

Inference

You wrote down $p_\theta(x|z)$ and can compute it.

Say I give you x. Keeping θ fixed, what was z ? Or $p_\theta(z|x)$?

Learning

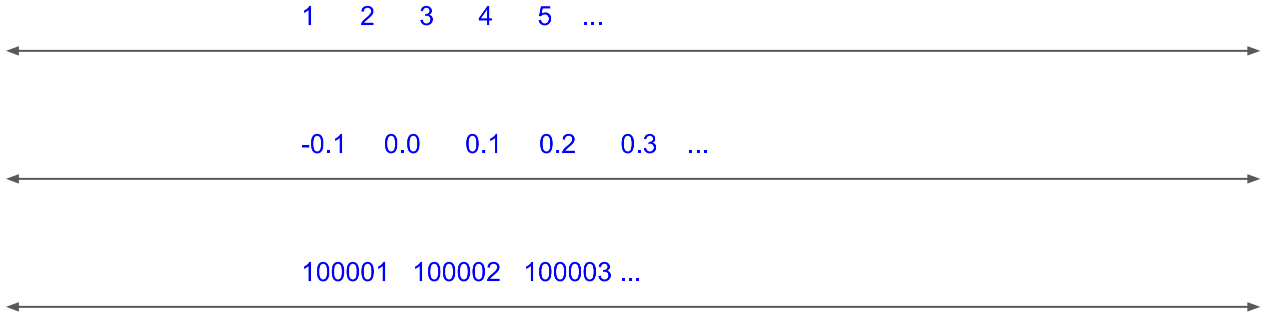
Is there a better (best) θ to generate the **observed** x from z ?

Inference

You wrote down $p_\theta(x|z)$ and can compute it.

Say I give you x. Keeping θ fixed, what was z? Or $p_\theta(z|x)$?

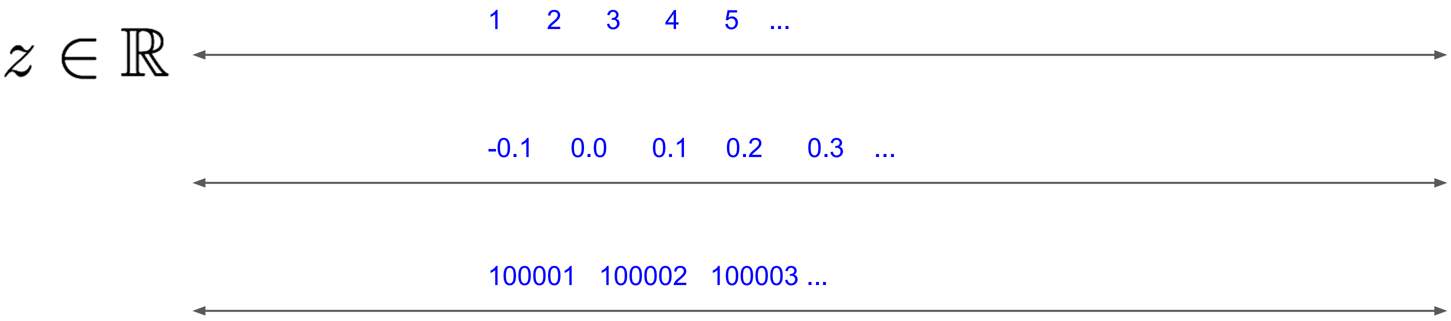
$z \in \mathbb{R}$



Inference

You wrote down $p_\theta(x|z)$ and can compute it.

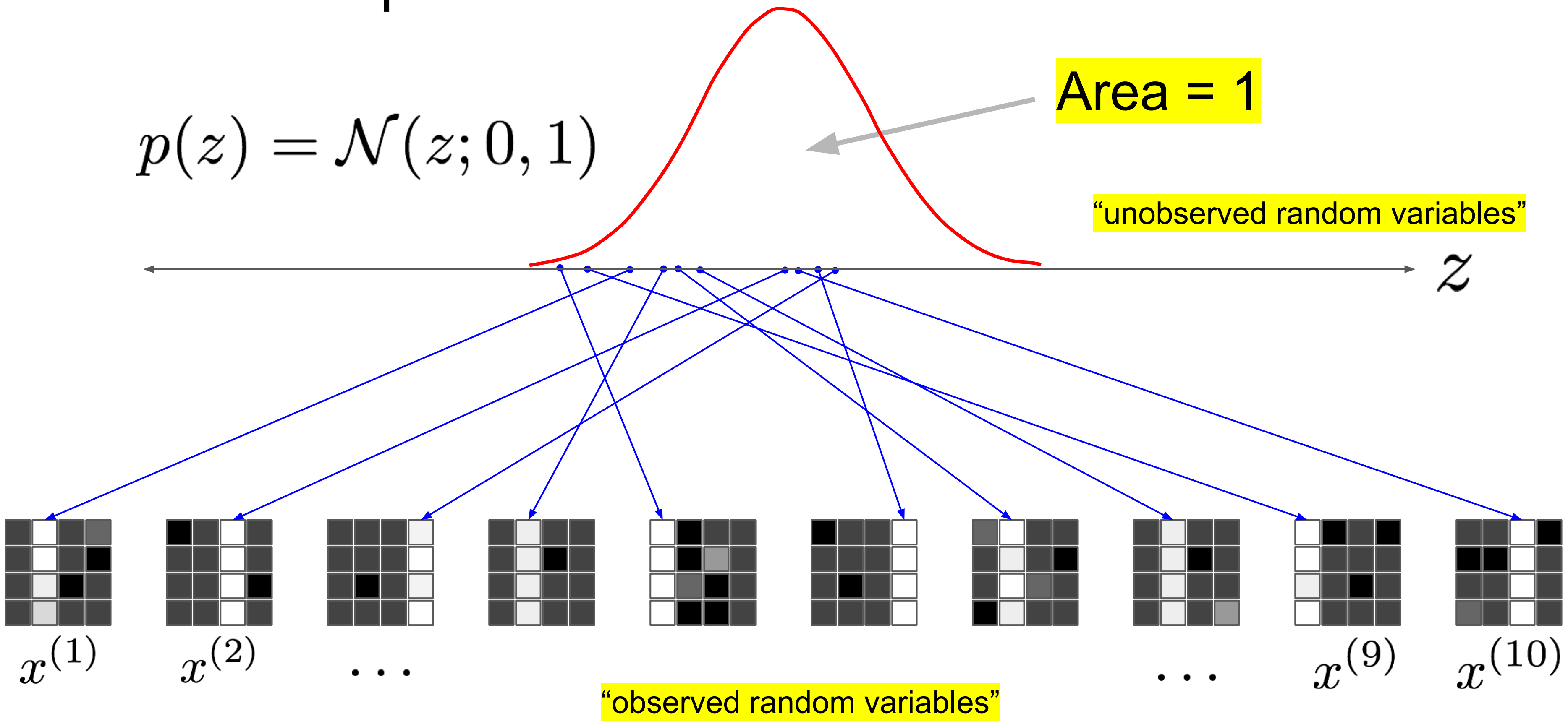
Say I give you x. Keeping θ fixed, what was z? Or $p_\theta(z|x)$?



To really answer that question, we need some notion of where we might have started! No inference without prior assumptions :)

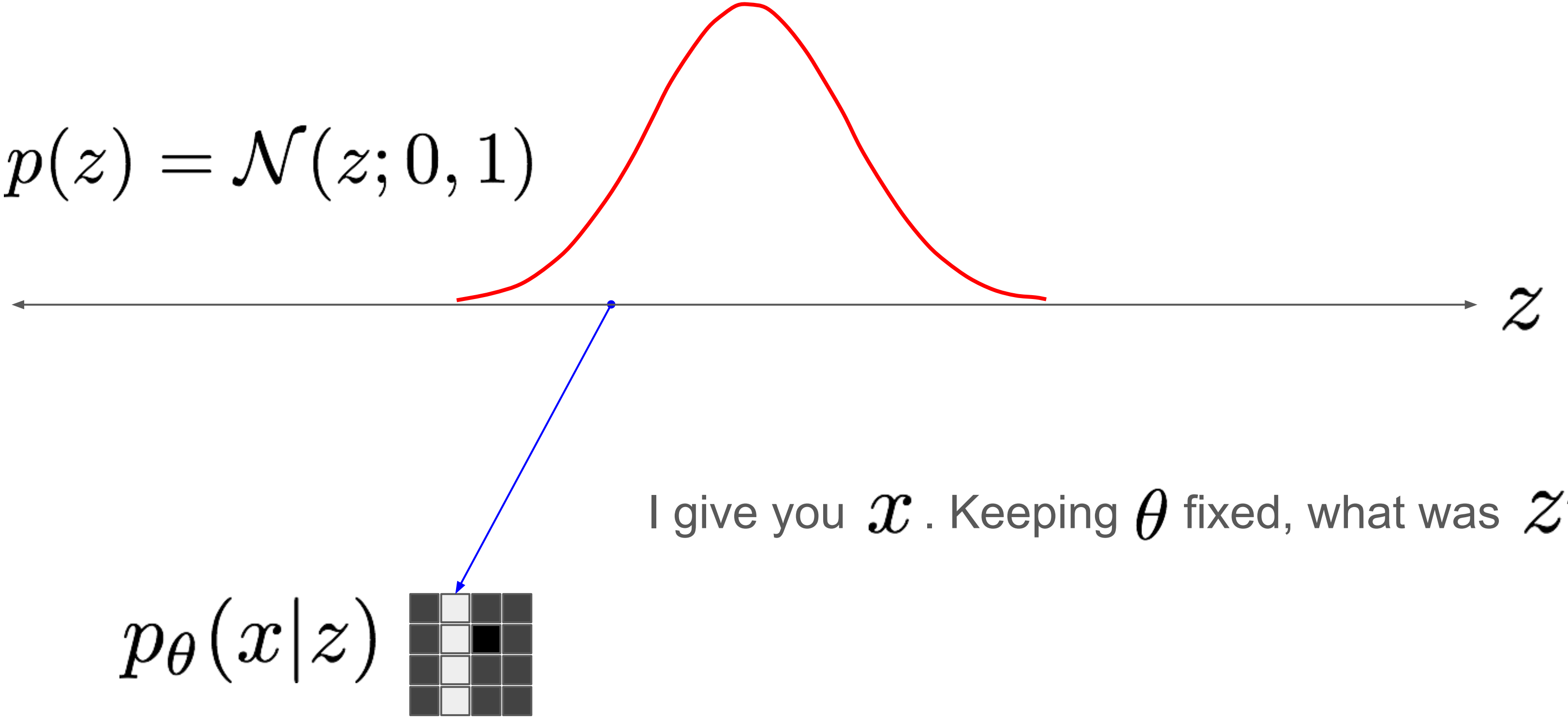
Prior assumptions

$$p(z) = \mathcal{N}(z; 0, 1)$$



Inference

$$p(z) = \mathcal{N}(z; 0, 1)$$

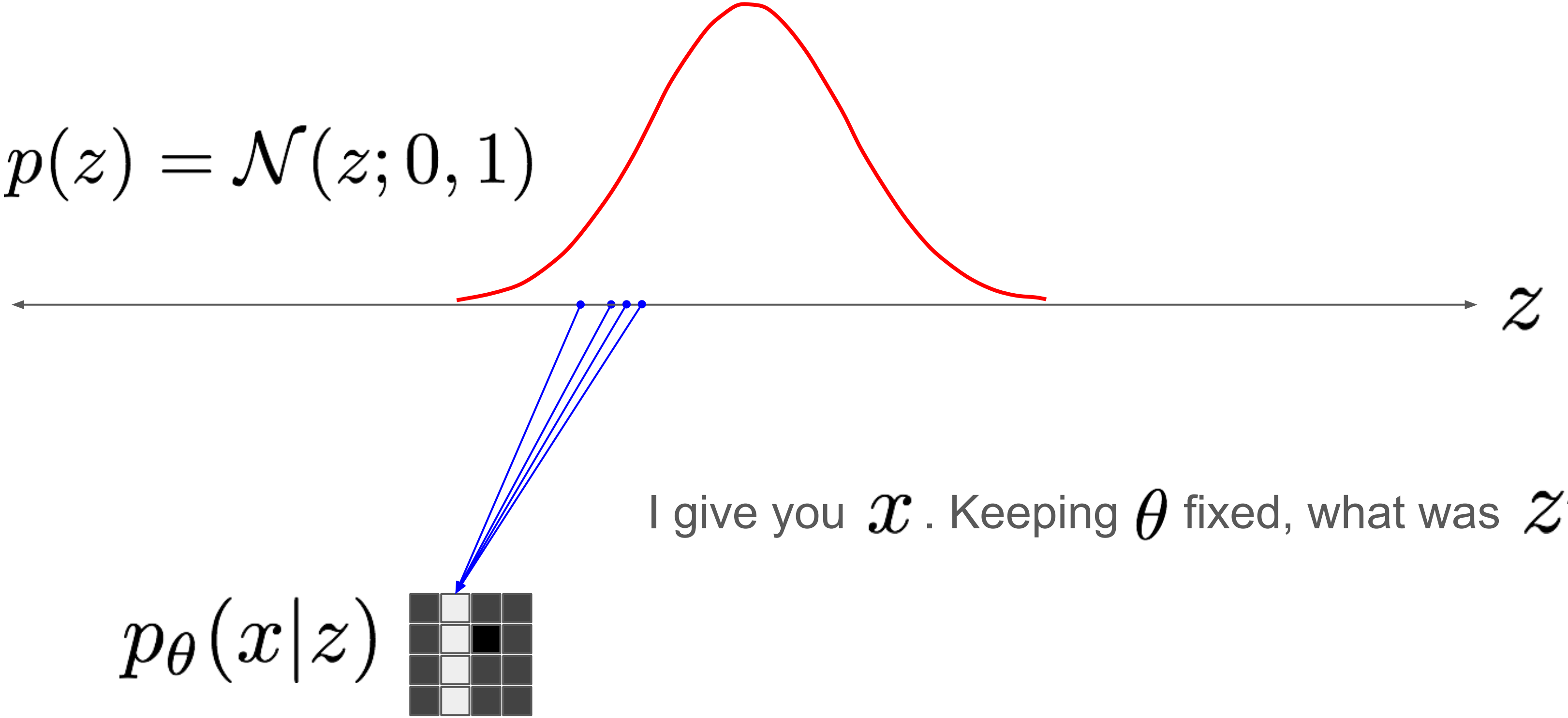


$$p_\theta(x|z)$$

I give you x . Keeping θ fixed, what was z?

Inference

$$p(z) = \mathcal{N}(z; 0, 1)$$

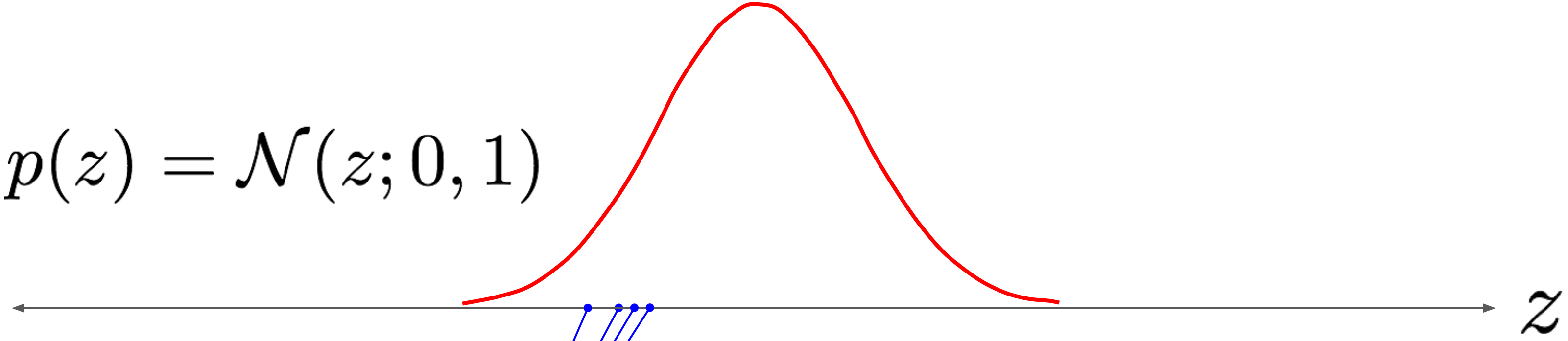


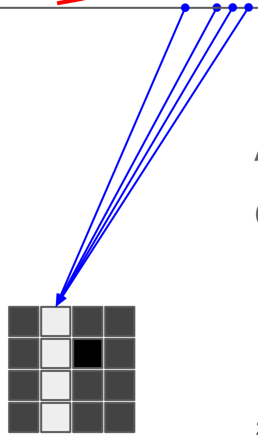
$$p_\theta(x|z)$$

I give you x. Keeping θ fixed, what was z?

3-minute exercise

$$p(z) = \mathcal{N}(z; 0, 1)$$



$p_\theta(x|z)$ 

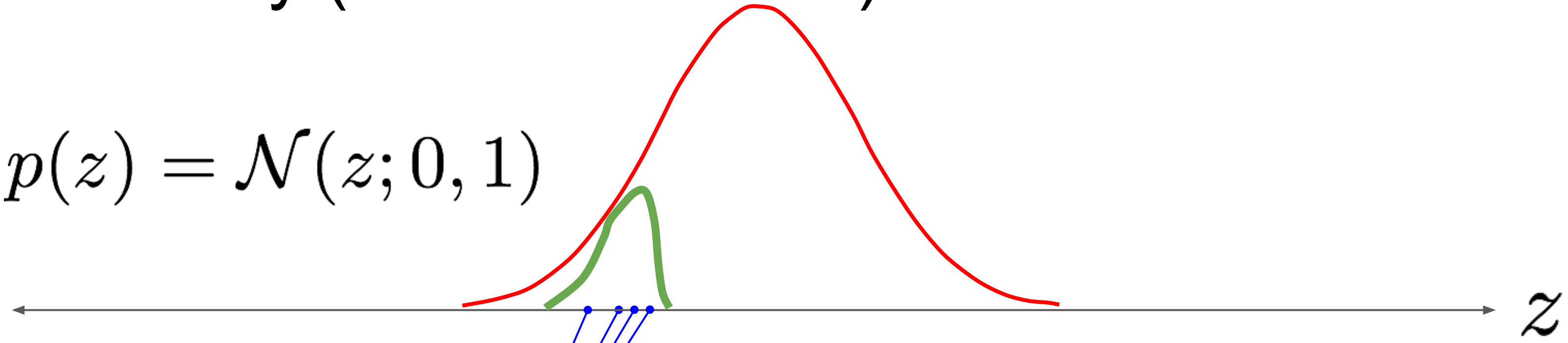
Assuming the largest value of $p_\theta(x|z)$ is 1, draw

$$p_\theta(x, z) = p_\theta(x|z)\, p(z)$$

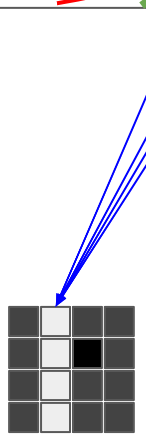
as a function of z on the same axis as above

Joint density (with x observed)

$$p(z) = \mathcal{N}(z; 0, 1)$$



$$p_\theta(x|z)$$

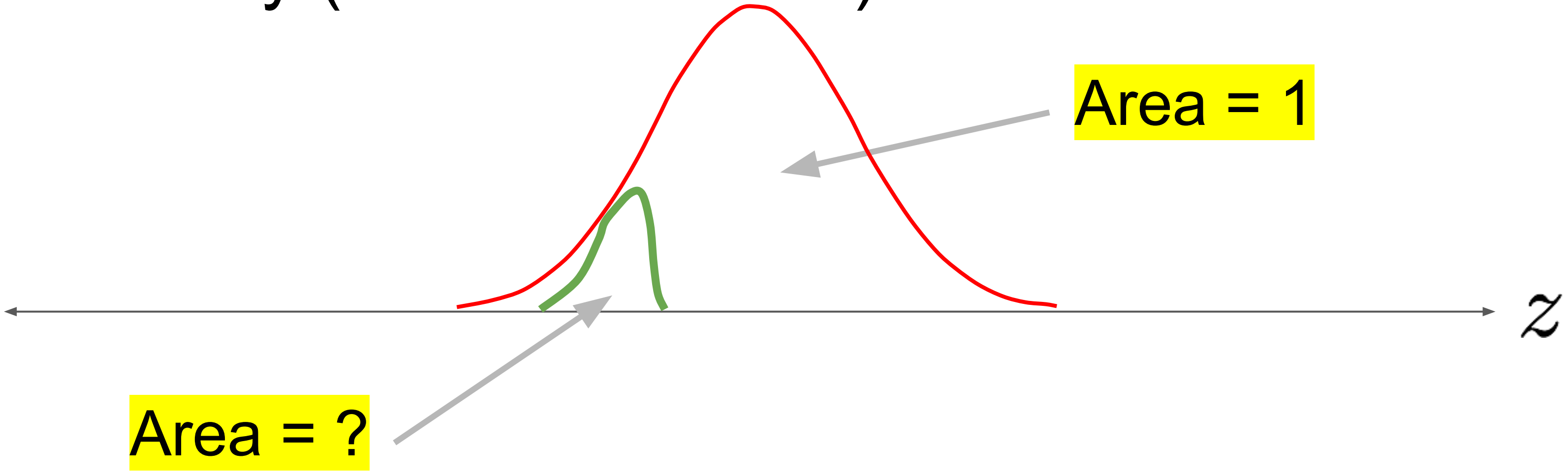


Assuming the largest value of $p_\theta(x|z)$ is 1, draw

$$p_\theta(x, z) = p_\theta(x|z)\, p(z)$$

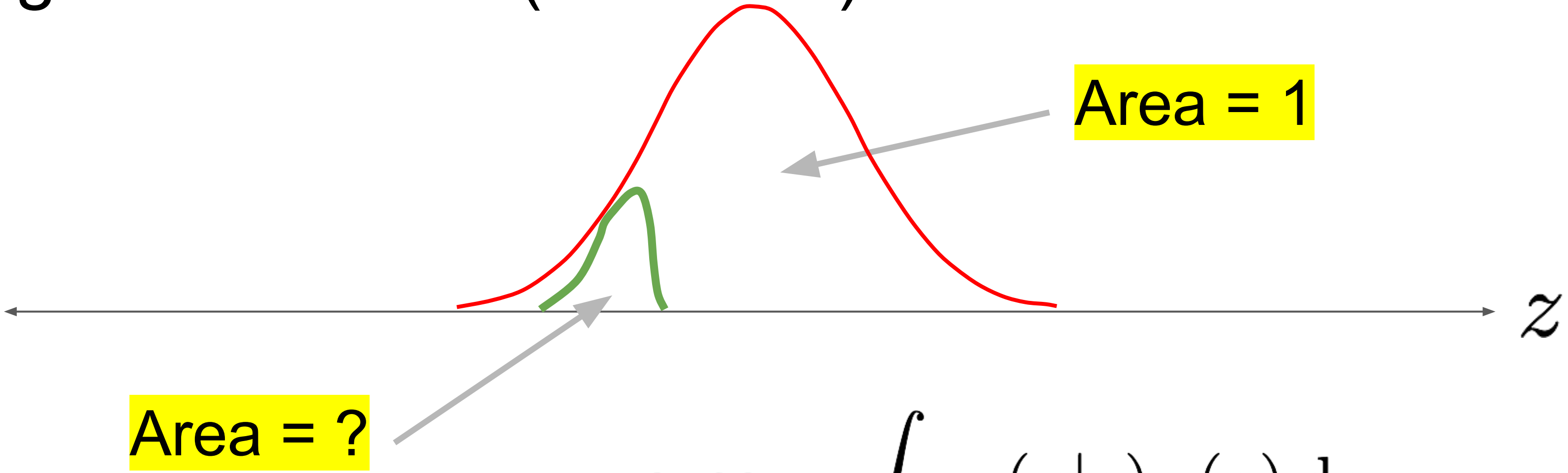
as a function of z on the same axis as above

Joint density (with x observed)



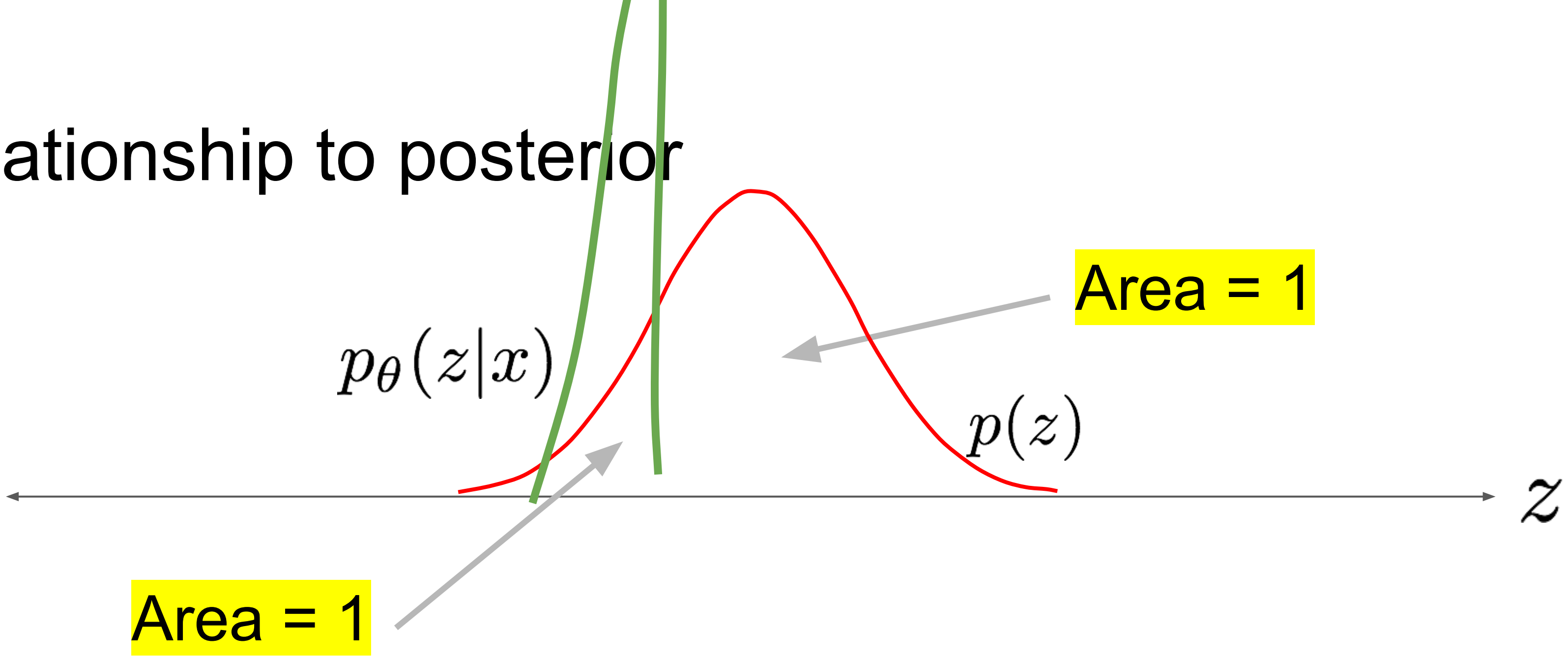
1-minute exercise:
what is the area?

Marginal likelihood (evidence)



$$\text{area} = \int p_\theta(x|z)\, p(z)\, \mathrm{d}z$$
$$= \int p_\theta(x, z)\, \mathrm{d}z$$
$$= p_\theta(x)$$

Relationship to posterior



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

Dividing by the marginal likelihood (evidence) scales the area back to 1...

Evidence, for all data points

$$X \equiv x^{(1)}, x^{(2)} \ldots, x^{(N)}$$

Area for data point *n*

$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x^{(n)})$$

Evidence, for all data points

$$X \equiv x^{(1)}, x^{(2)} \ldots, x^{(N)}$$

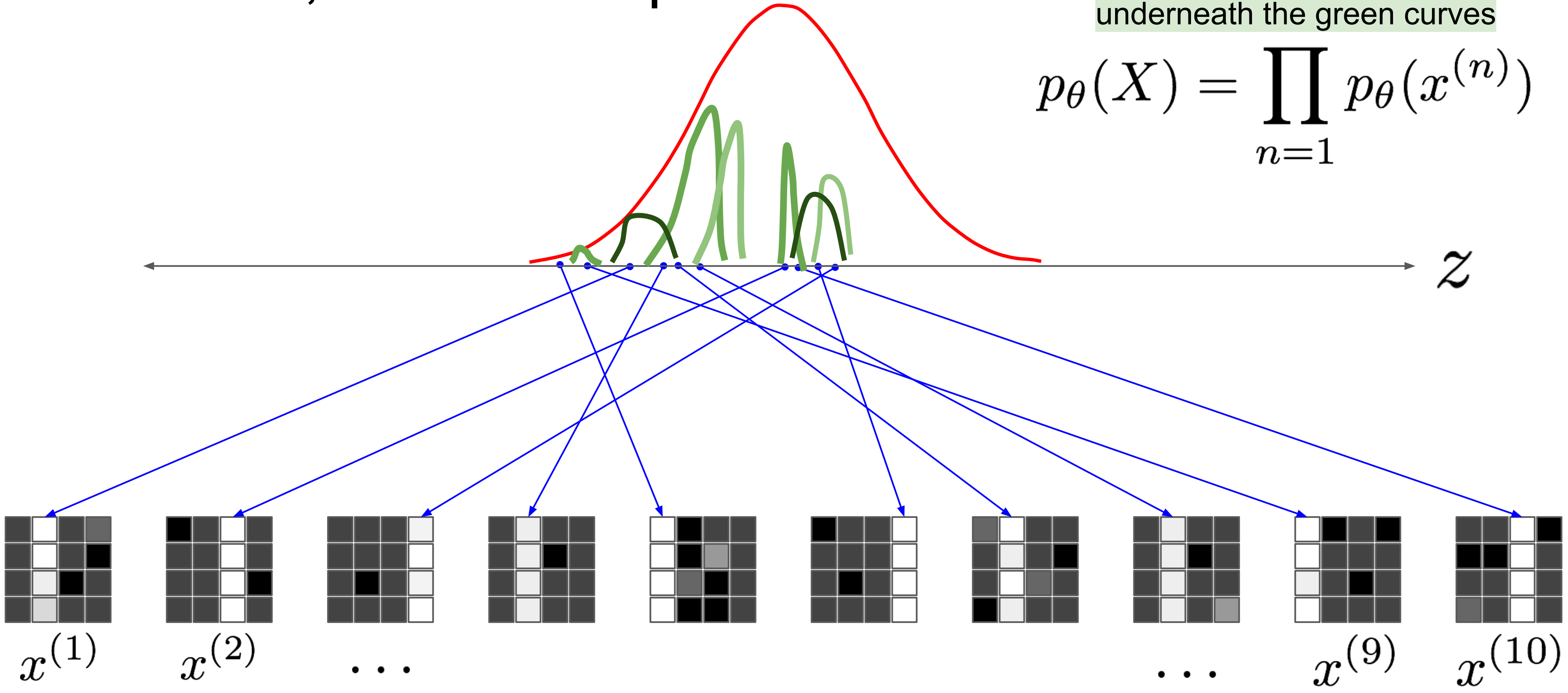
Area for data point *n*

$$\log p_\theta(X) = \sum_{n=1}^{N} \log p_\theta(x^{(n)})$$

Evidence, for all data points

The product of the areas underneath the green curves

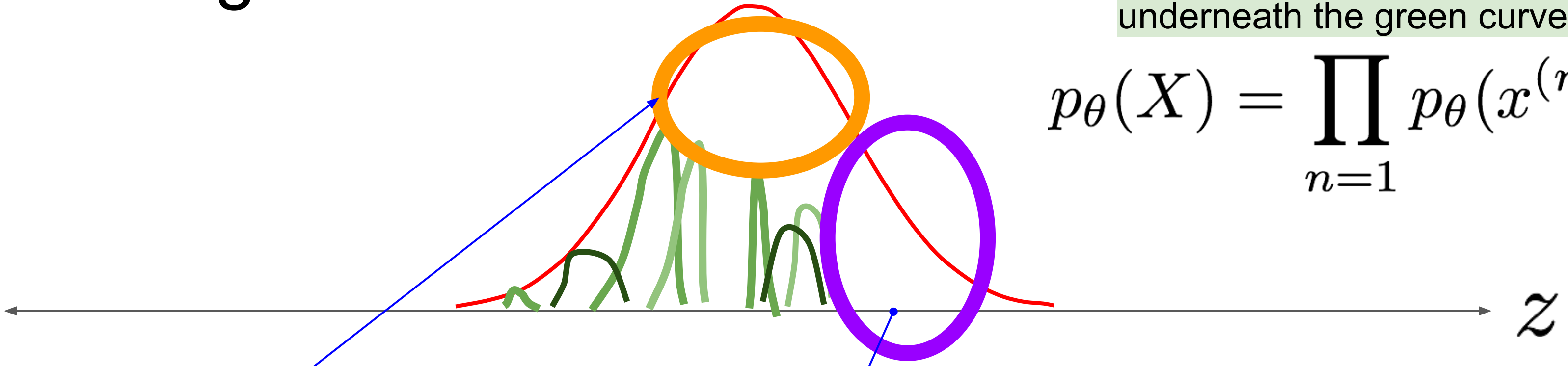
$$p_\theta(X) = \prod_{n=1} p_\theta(x^{(n)})$$



Maximizing the evidence

The product of the areas underneath the green curves

$$p_\theta(X) = \prod_{n=1} p_\theta(x^{(n)})$$



By changing θ we can make the evidence for these data points bigger...

These z's don't generate images like the ones in the data set…

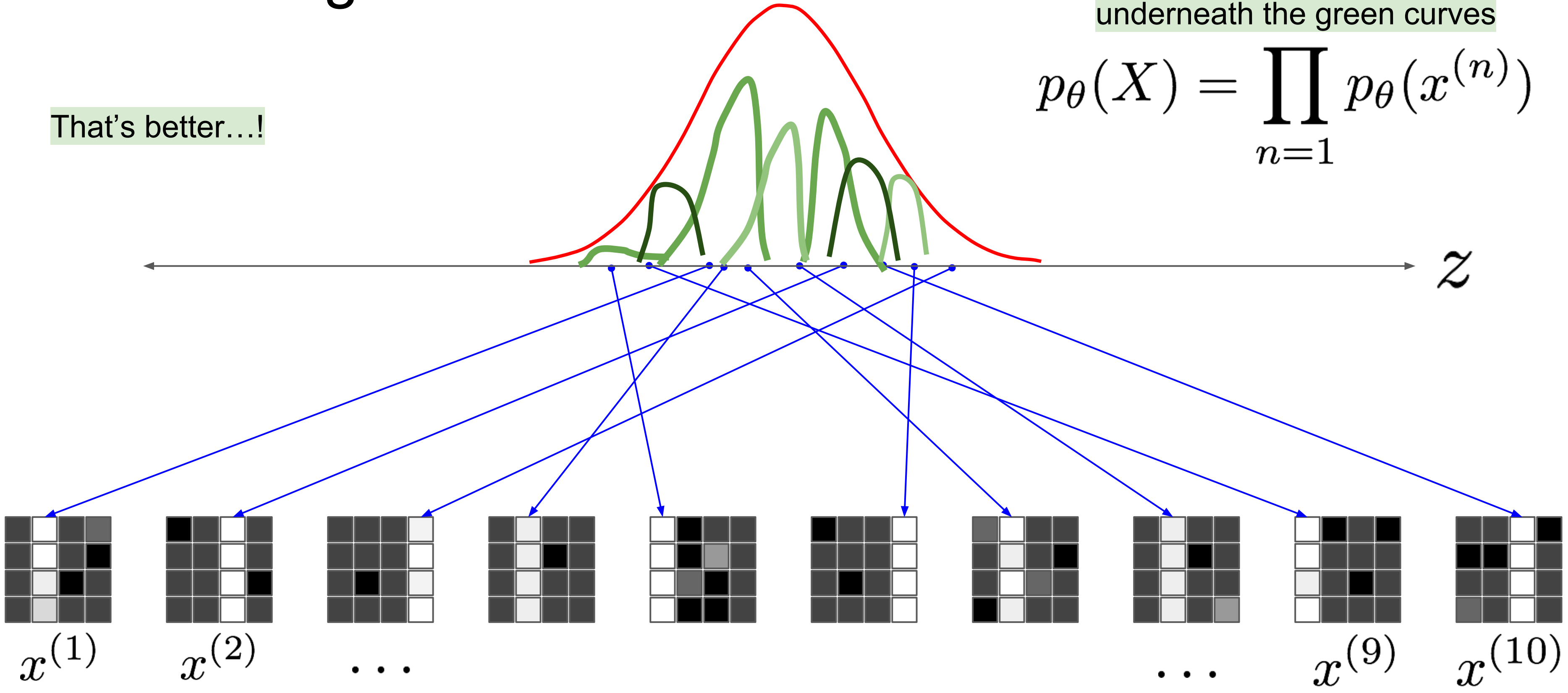
(With this θ, the prior doesn't capture the data manifold well)

Maximizing the evidence

That's better…!

The product of the areas underneath the green curves

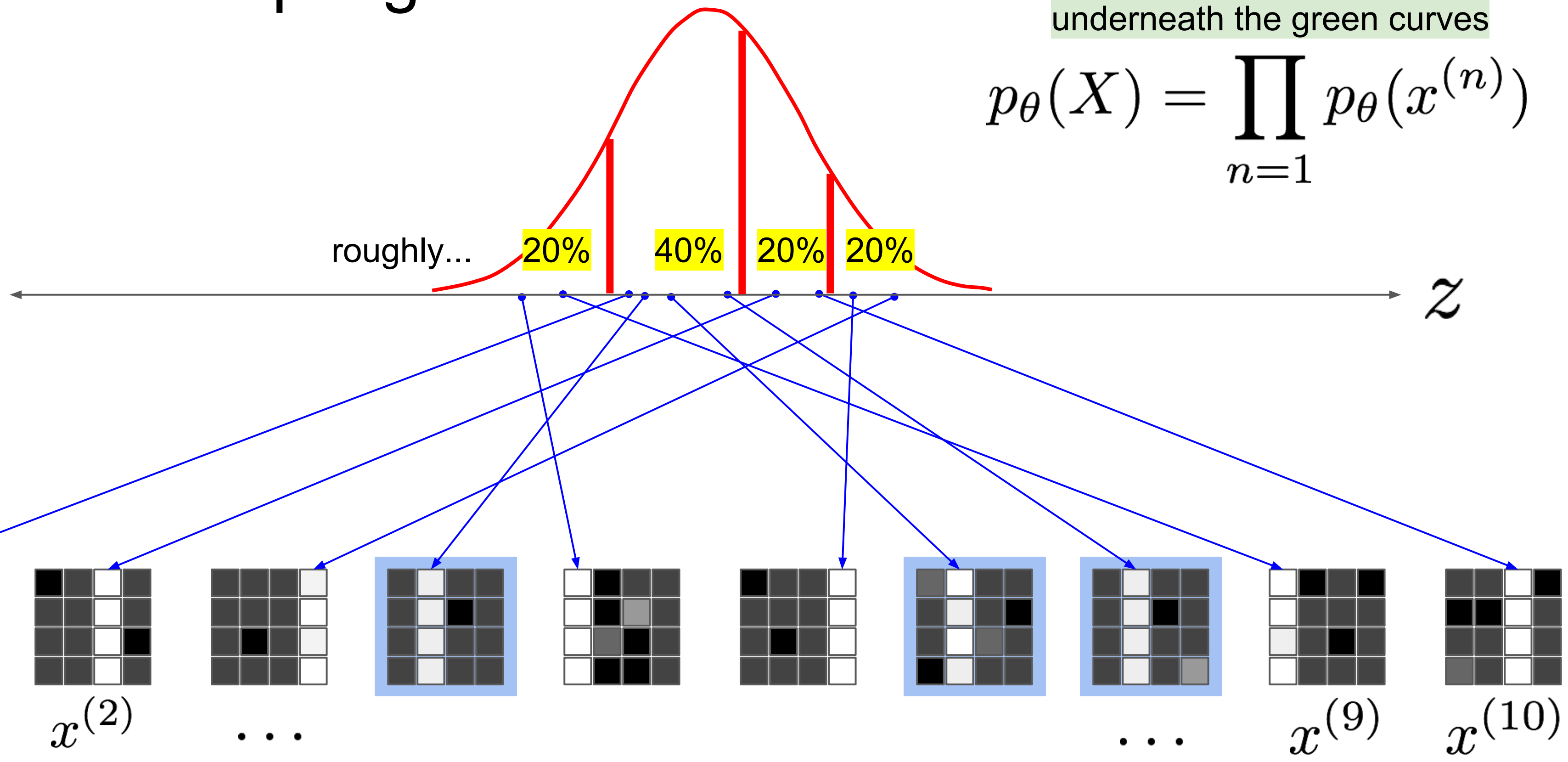
$$p_\theta(X) = \prod_{n=1} p_\theta(x^{(n)})$$



For the sharp-sighted

The product of the areas underneath the green curves

$$p_\theta(X) = \prod_{n=1} p_\theta(x^{(n)})$$



Learning

We want to maximize

$$\max_{\theta}\left[\log p_{\theta}(X)\right]$$

$$=\max_{\theta}\left[\sum_{n=1}^{N}\log p_{\theta}(x^{(n)})\right]$$

Area for data point *n*

except that we cannot write down an analytically tractable expression for the area.

Strategies: Stochastic (Monte Carlo samples + gradients) or deterministic (approximate inference). We'll follow the "deterministic" path next...

Approximate inference

We want to use this quantity for “learning”, but cannot compute it in an analytically tractable way:

$$\log p_\theta(x) = \log \int p_\theta(x|z)\, p(z)\, \mathrm{d}z$$

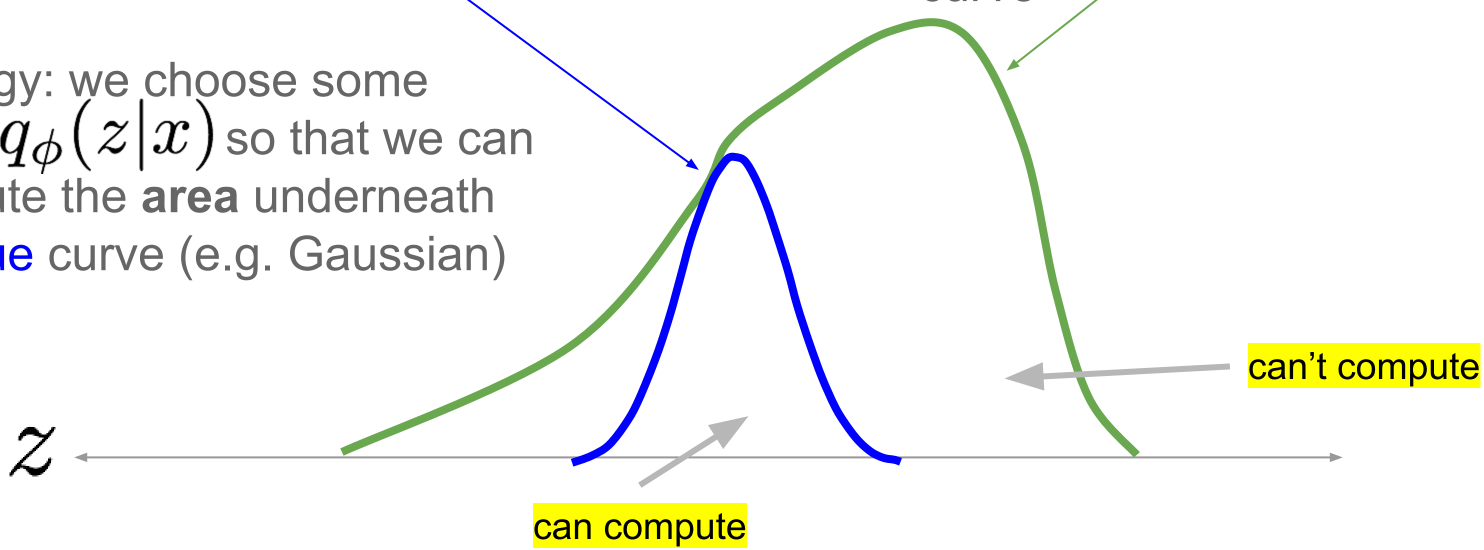
$$= \log \int p_\theta(x, z)\, \mathrm{d}z$$

"Variational lower bound"

Unnormalized $q_\phi(z|x)$

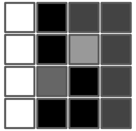
Strategy: we choose some other $q_\phi(z|x)$ so that we can compute the **area** underneath the blue curve (e.g. Gaussian)

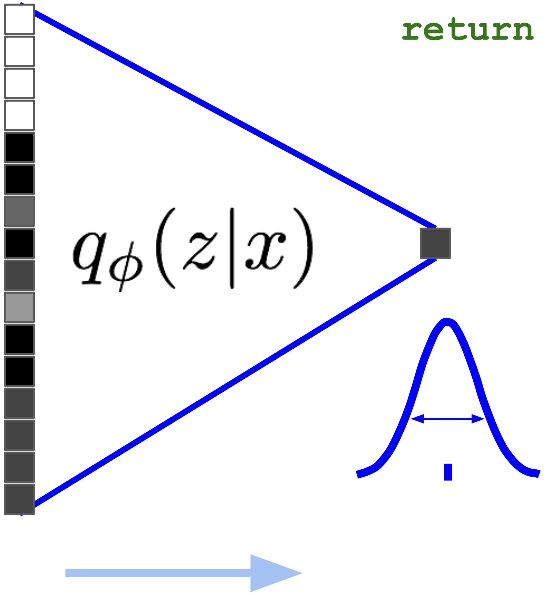
Unnormalized $p_\theta(z|x)$
We cannot (tractably) compute the **area** underneath the green curve



Encoder

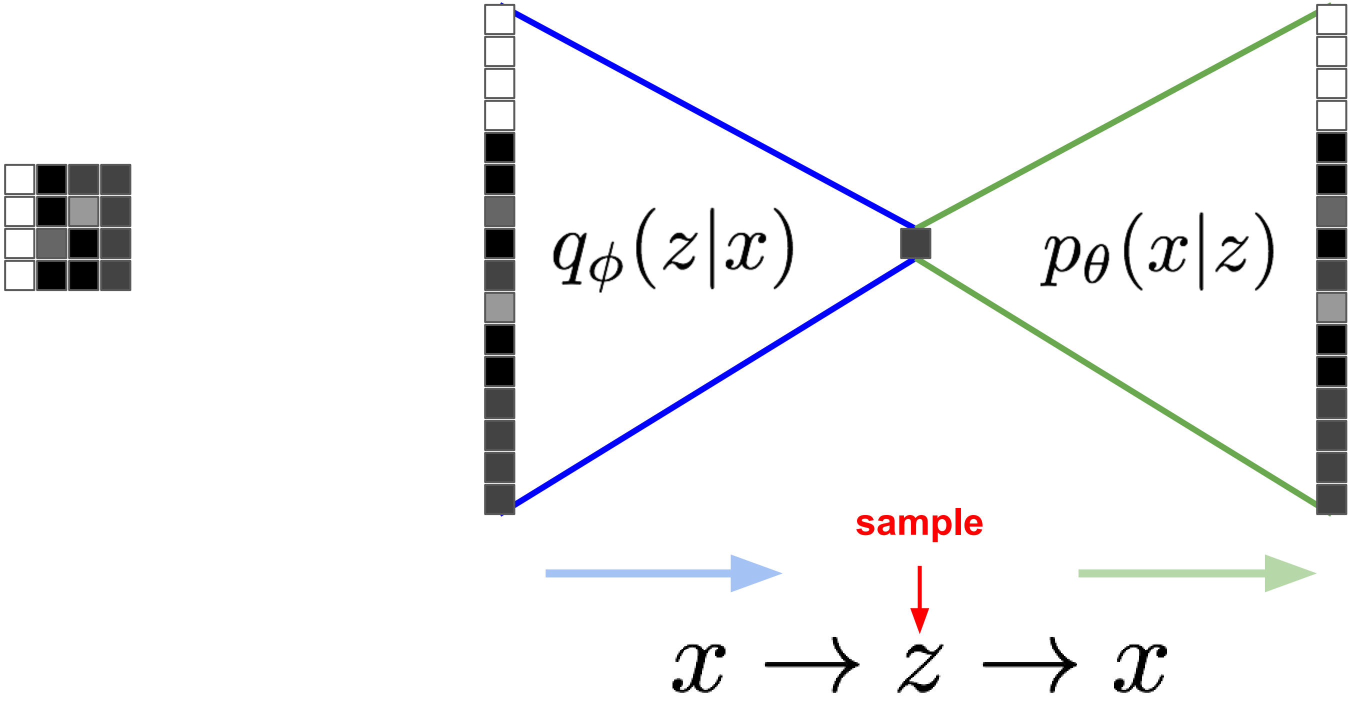
```
def inference_network(x, latent_dim=1):
    ...
    return mu, sigma
```





$$x \rightarrow \mu_\phi(x), \sigma^2_\phi(x)$$

Encoder decoder

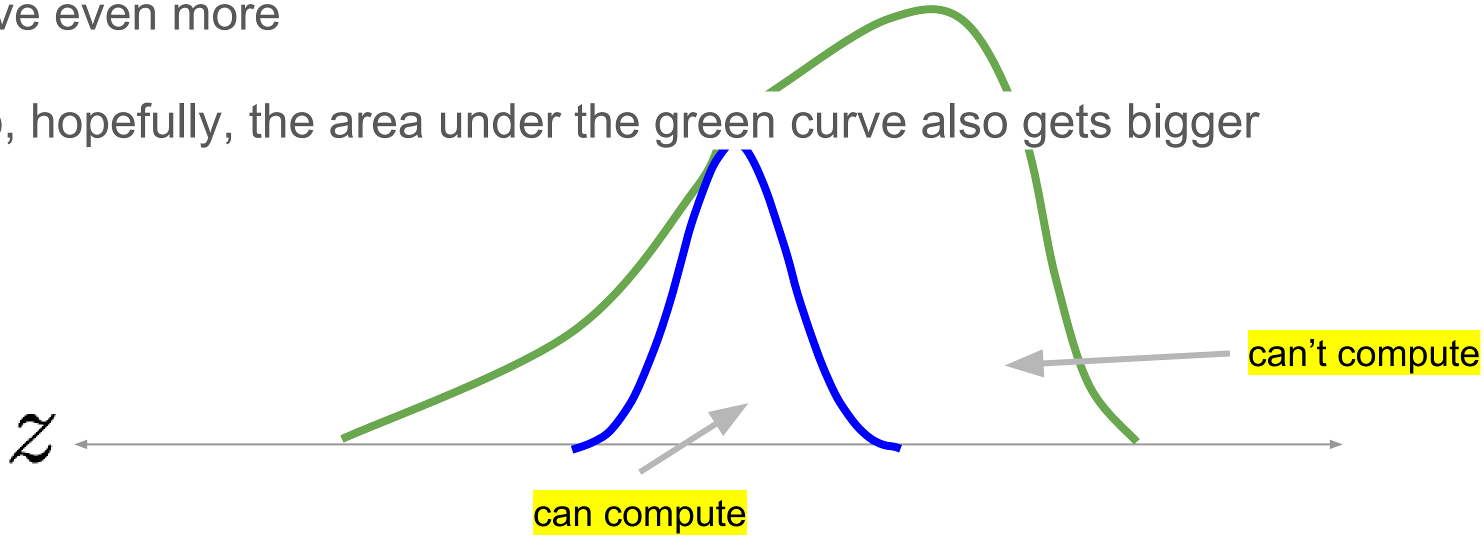


Strategy

Change ϕ to inflate the area under the blue curve. We can do that!

Change θ to change the green curve, so that we can inflate the area under the blue curve even more

...and so, hopefully, the area under the green curve also gets bigger



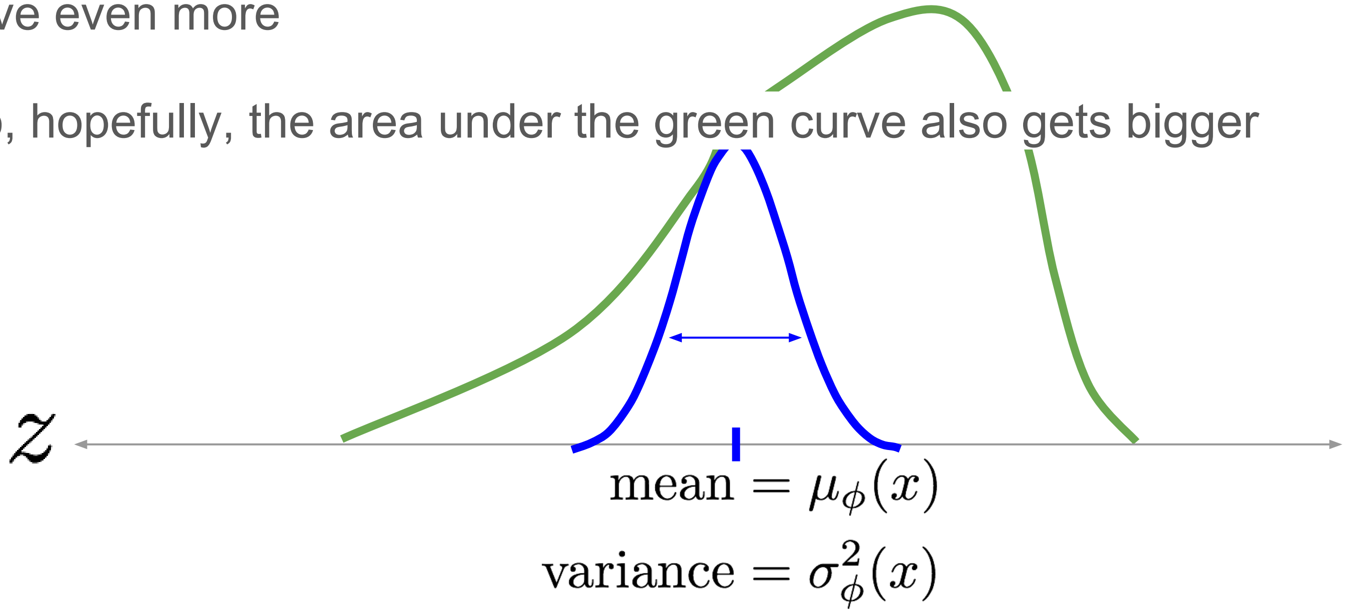
Whhaaaatttt?

Strategy

Change ϕ to inflate the area under the blue curve. We can do that!

Change θ to change the green curve, so that we can inflate the area under the blue curve even more

...and so, hopefully, the area under the green curve also gets bigger

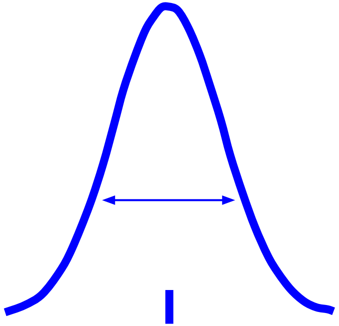


3-minute exercise

Create and draw $q_\phi(z|x) = \mathcal{N}\left(z; \mu_\phi(x), \sigma^2_\phi(x)\right)$ as a function.

It could be a multi-layer perceptron (MLP) that takes 16-dimensional x, and produces two 1-dimensional quantities,

$$\text{mean} = \mu_\phi(x)$$
$$\text{variance} = \sigma^2_\phi(x)$$

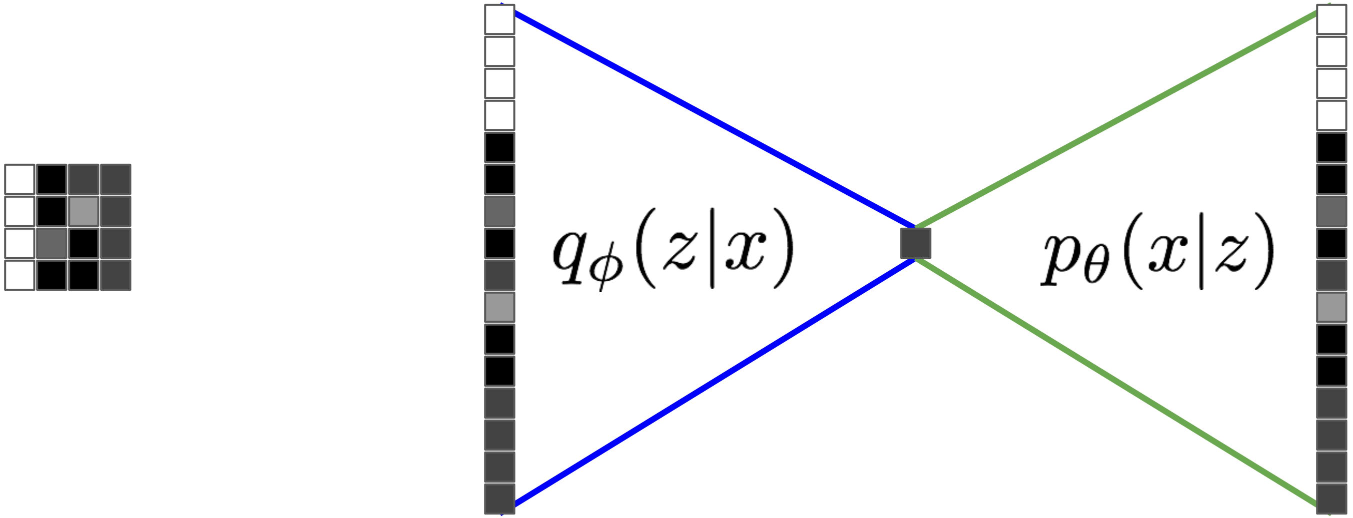


scratch space

What are your parameters ϕ?

Objective function discussion

maximize (for all data points)...



KL ≥ 0

$$\mathbb{E}_{q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] - \mathrm{KL}\big(q_\phi(z|x)\,\|\,p(z)\big)$$

Evidence lower bound (ELBO) for one data point

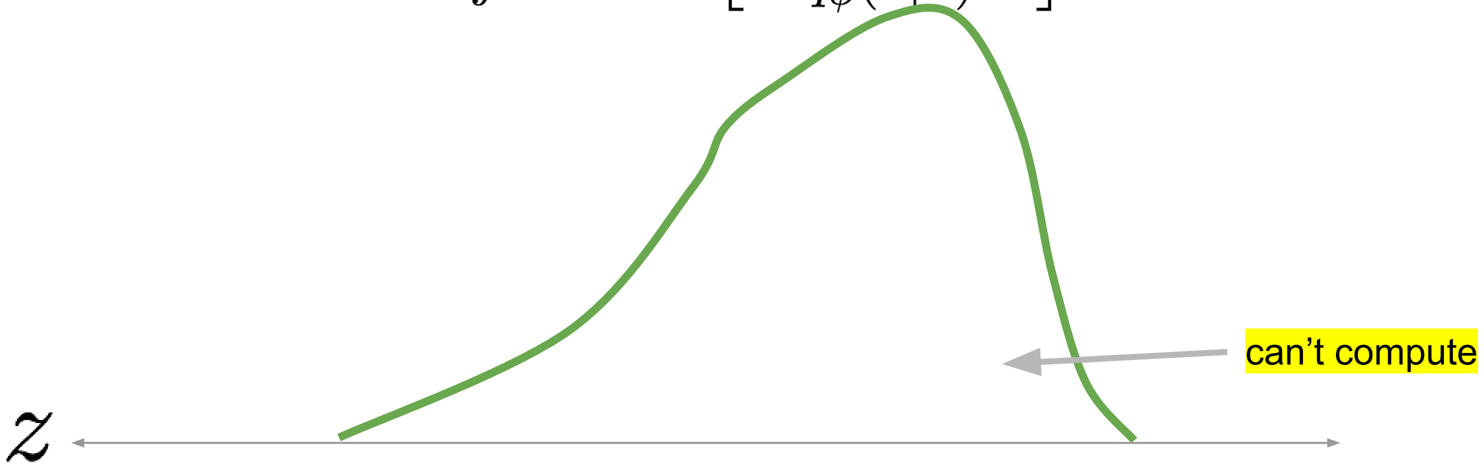
$$
\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)\, p(z)\, \mathrm{d}z \\
&= \log \int q_\phi(z|x) \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)} \right] \mathrm{d}z \\
&\geq \int q_\phi(z|x) \log \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)} \right] \mathrm{d}z \qquad \text{[Jensen]} \\
&= \int q_\phi(z|x) \log p_\theta(x|z)\, \mathrm{d}z - \int q_\phi(z|x) \log \left[\frac{q_\phi(z|x)}{p(z)} \right] \mathrm{d}z \\
&= \mathbb{E}_{q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big] - \mathrm{KL}\big(q_\phi(z|x) \,\|\, p(z)\big) \\
&\equiv \mathcal{L}(x; \theta, \phi)
\end{aligned}
$$

ELBO → $\mathcal{L}(x; \theta, \phi)$

Evidence lower bound (ELBO) for one data point

$$\log p_\theta(x) = \log \int p_\theta(x|z)\, p(z)\, \mathrm{d}z$$

$$= \log \int q_\phi(z|x) \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)} \right] \mathrm{d}z$$



Evidence lower bound (ELBO) for one data point

$$\log p_\theta(x) = \log \int p_\theta(x|z)\, p(z)\, \mathrm{d}z$$

$$= \log \int q_\phi(z|x) \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)} \right] \mathrm{d}z$$

concave function

$$\geq \int q_\phi(z|x) \log \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)} \right] \mathrm{d}z \qquad \text{[Jensen]}$$

expectation

concave function

z

$$\log \mathbb{E}_{q_\phi(z|x)} \Big[f(z) \Big] \geq \mathbb{E}_{q_\phi(z|x)} \Big[\log f(z) \Big]$$

3-minute exercise

Jensen's inequality

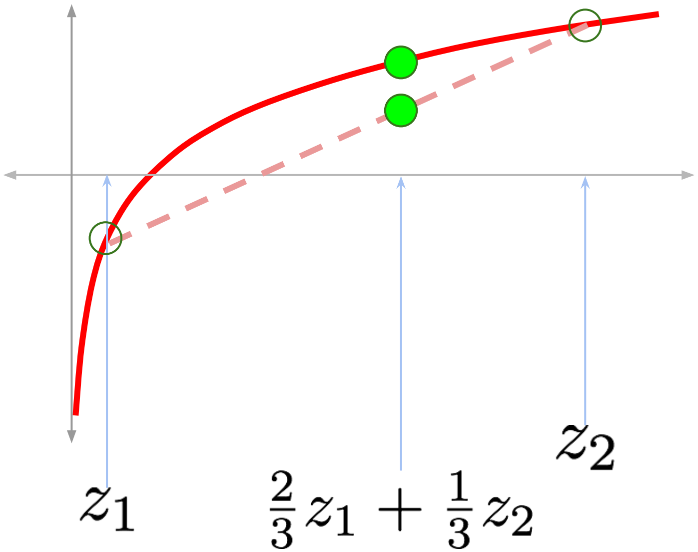
Draw log(...) as a function, and convince yourself that

$$\log\left(\underline{\tfrac{2}{3}}z_1 + \underline{\tfrac{1}{3}}z_2\right) \geq \underline{\tfrac{2}{3}}\log(z_1) + \underline{\tfrac{1}{3}}\log(z_2)$$

is always true for any (nonnegative) setting of z_1 and z_2.

Logarithm (concave)

$$\log\left(\tfrac{2}{3}z_1 + \tfrac{1}{3}z_2\right) \geq \tfrac{2}{3}\log(z_1) + \tfrac{1}{3}\log(z_2)$$



Evidence lower bound (ELBO) for one data point

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)\, p(z)\, \mathrm{d}z \\
&= \log \int q_\phi(z|x) \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)}\right] \mathrm{d}z \\
&\geq \int q_\phi(z|x) \log \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)}\right] \mathrm{d}z \\
&= \int q_\phi(z|x) \log p_\theta(x|z)\, \mathrm{d}z - \int q_\phi(z|x) \log \left[\frac{q_\phi(z|x)}{p(z)}\right] \mathrm{d}z \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \mathrm{KL}\left(q_\phi(z|x)\,\|\,p(z)\right) \\
&\equiv \mathcal{L}(x; \theta, \phi)
\end{aligned}$$

Reconstruction

Expected log likelihood. Cannot compute in closed form, and will have to get a Monte Carlo estimate (with SGD)

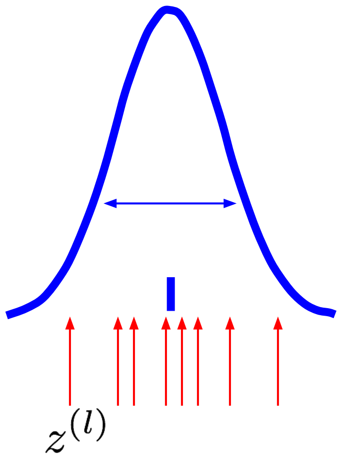
Kullback-Leibler divergence between two Gaussian distributions (here). Can compute in closed form

ELBO →

Expected log likelihood

We can estimate the expected log likelihood with a Monte Carlo estimate:

Draw L samples $z^{(l)} \sim \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x))$ ...



Expected log likelihood

We can estimate the expected log likelihood with a Monte Carlo estimate:

Draw L samples $z^{(l)} \sim \mathcal{N}(z; \mu_\phi(x), \ \sigma_\phi^2(x))$ and use them to estimate the average:

$$\mathbb{E}_{q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] = \mathbb{E}_{z \sim \mathcal{N}(z;\mu_\phi(x), \ \sigma_\phi^2(x))}\Big[\log p_\theta(x|z)\Big]$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x|z^{(l)})$$

Expected log likelihood

We can estimate the expected log likelihood with a Monte Carlo estimate:

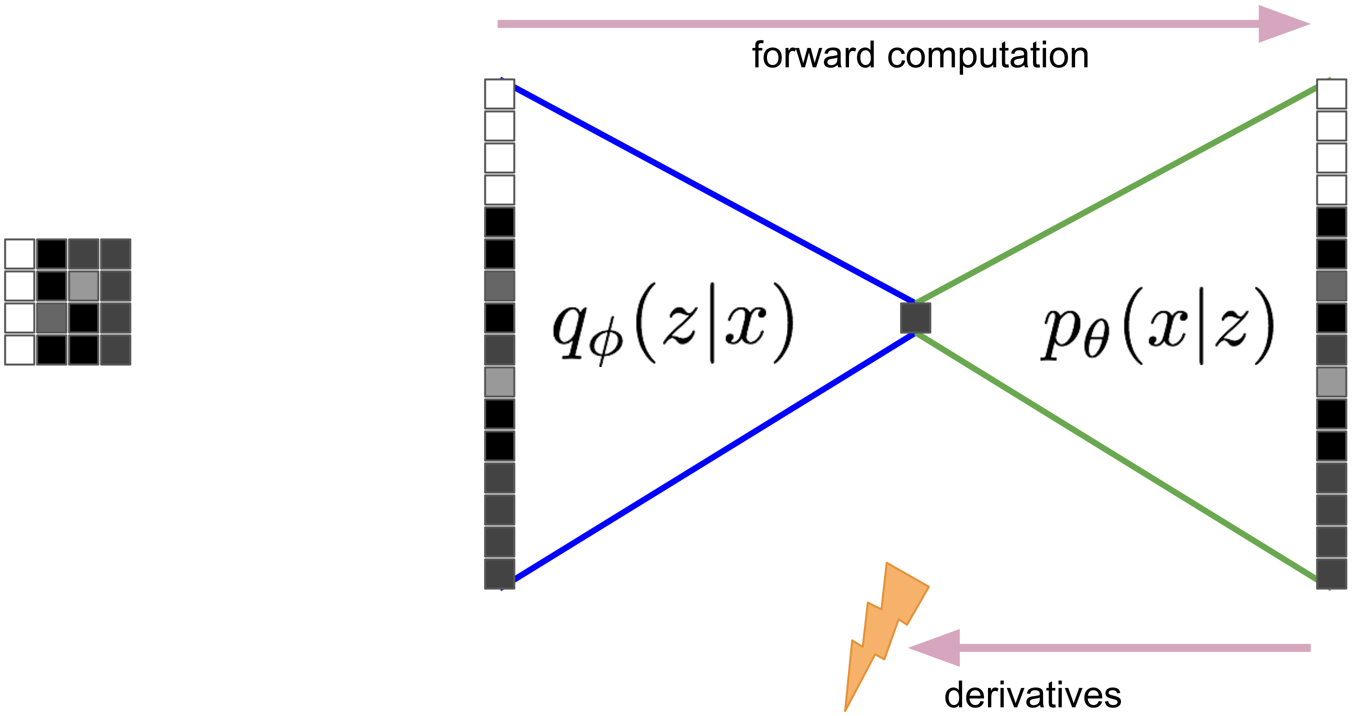
Draw L samples $z^{(l)} \sim \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x))$ and use them to estimate the average:

$$\mathbb{E}_{q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] = \mathbb{E}_{z\sim\mathcal{N}(z;\mu_\phi(x),\sigma^2_\phi(x))}\Big[\log p_\theta(x|z)\Big]$$

$$\approx \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x|z^{(l)})$$

Using samples in *this* way removes ϕ from part of the objective function, and even though we can evaluate it, we can't take derivatives / get the gradients!

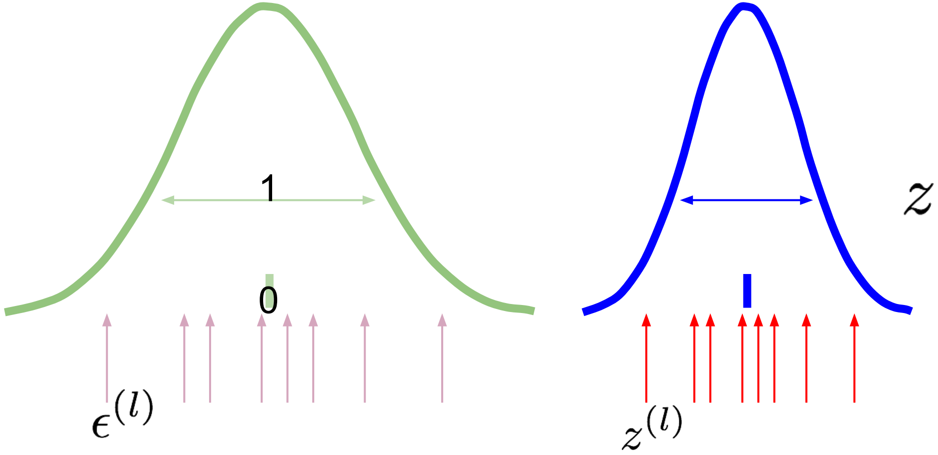
Naive sampling



Expected log likelihood: reparameterization trick

We can estimate the expected log likelihood with a Monte Carlo estimate:

Draw L samples $\epsilon^{(l)} \sim \mathcal{N}(\epsilon; 0,\ 1)$ and transform them!



$$z^{(l)} = \mu_\phi(x) + \epsilon^{(l)} * \sigma_\phi(x)$$

Expected log likelihood

We can estimate the expected log likelihood with a Monte Carlo estimate:

Draw L samples $\epsilon^{(l)} \sim \mathcal{N}(\epsilon; 0,\ 1)$ and use them to estimate the average:

$$\mathbb{E}_{q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] = \mathbb{E}_{\epsilon \sim \mathcal{N}(\epsilon;0,1)}\Big[\log p_\theta(x \mid z = \mu_\phi(x) + \epsilon * \sigma_\phi(x))\Big]$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x \mid z^{(l)} = \mu_\phi(x) + \epsilon^{(l)} * \sigma_\phi(x))$$

Expected log likelihood

We can estimate the expected log likelihood with a Monte Carlo estimate:

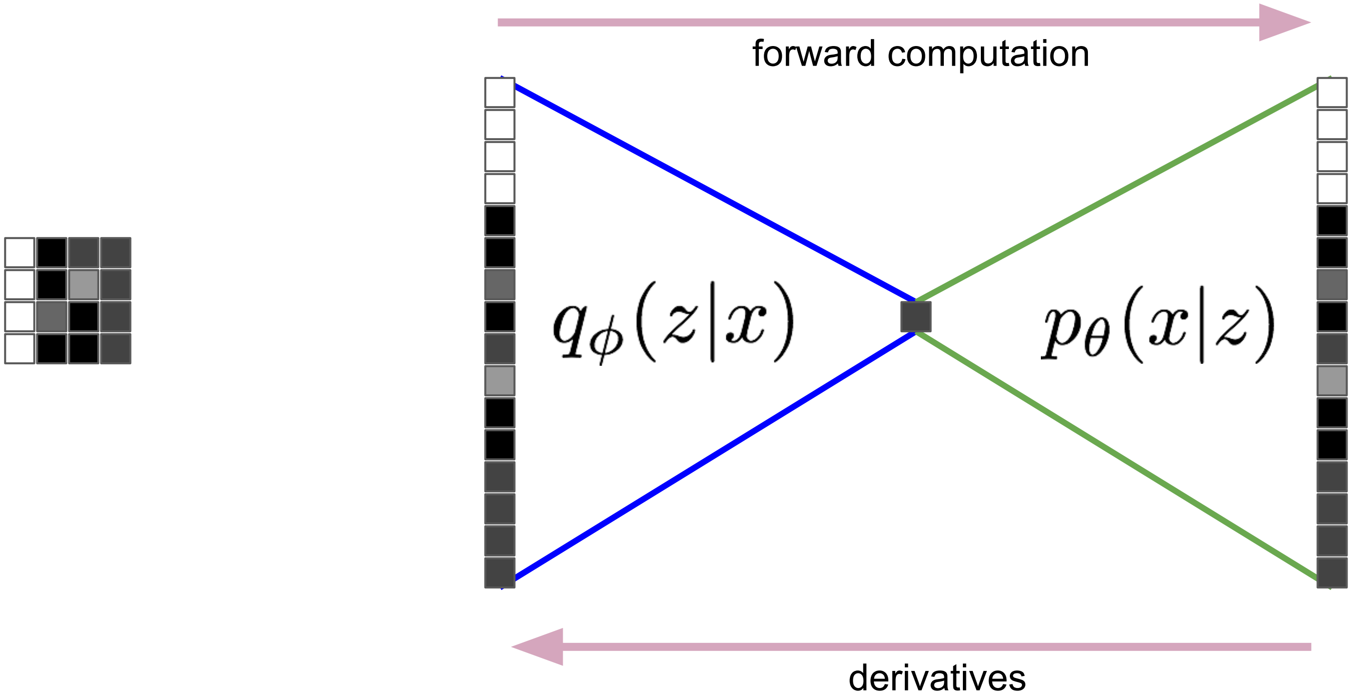
Draw L samples $\epsilon^{(l)} \sim \mathcal{N}(\epsilon; 0,\ 1)$ and use them to estimate the average:

$$\mathbb{E}_{q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] = \mathbb{E}_{\epsilon\sim\mathcal{N}(\epsilon;0,1)}\Big[\log p_\theta(x \mid z = \mu_\phi(x) + \epsilon * \sigma_\phi(x))\Big]$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x \mid z^{(l)} = \mu_\phi(x) + \epsilon^{(l)} * \sigma_\phi(x))$$

The noise is introduced “from outside” the computation graph, and we can evaluate the objective function **and** take derivatives / get the gradients!

Reparameterization trick



ELBO for full data set

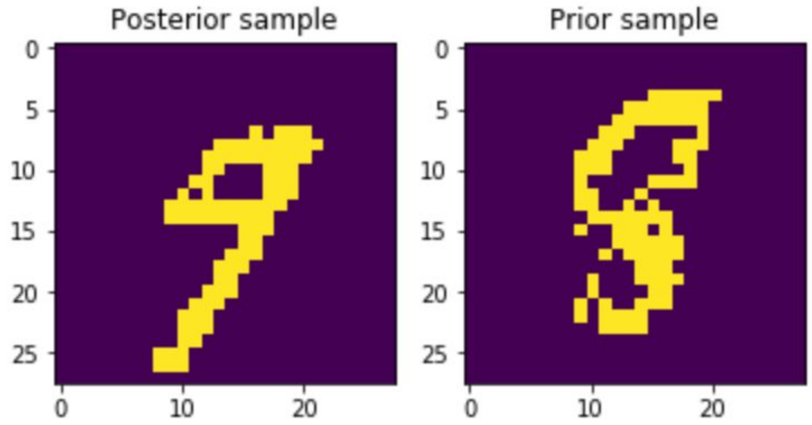
You now have all the tools to estimate the ELBO for a whole data set,

$$\mathcal{L}(X;\theta,\phi) = \sum_{n=1}^{N} \left\{ \mathbb{E}_{q_\phi(z^{(n)}|x^{(n)})}\Big[\log p_\theta(x^{(n)}|z^{(n)})\Big] - \mathrm{KL}\big(q_\phi(z^{(n)}|x^{(n)}) \,\|\, p(z^{(n)})\big) \right\}$$

take mini-batch subsamples, and use stochastic gradient ascent to maximize it.

Practical

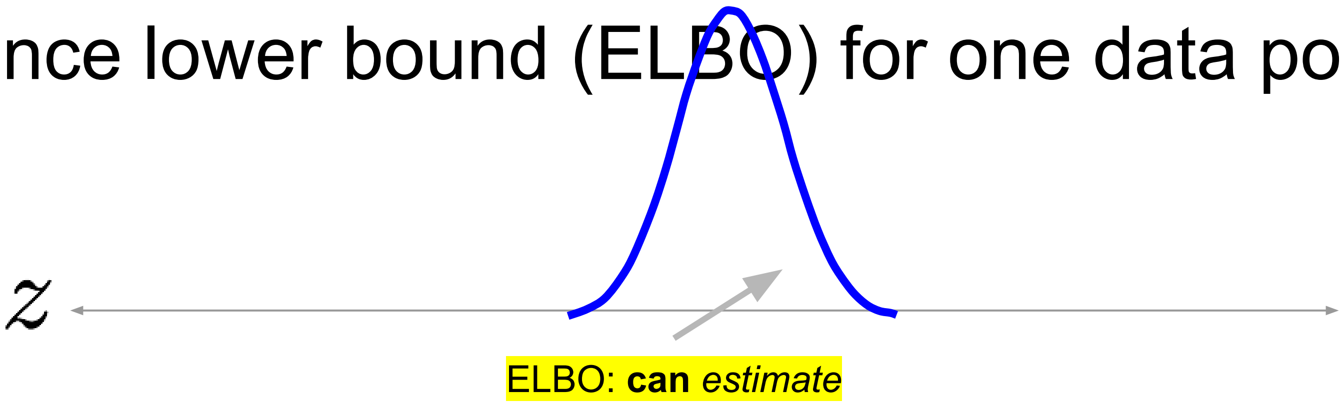
```
Iteration: 7700 ELBO: -85.999 Examples/s: 1.158e+07
Iteration: 7800 ELBO: -90.856 Examples/s: 1.155e+07
Iteration: 7900 ELBO: -85.855 Examples/s: 1.155e+07
Iteration: 8000 ELBO: -88.127 Examples/s: 1.190e+07
```




```
Iteration: 8100 ELBO: -90.874 Examples/s: 1.118e+07
Iteration: 8200 ELBO: -92.233 Examples/s: 1.166e+07
Iteration: 8300 ELBO: -95.609 Examples/s: 1.148e+07
Iteration: 8400 ELBO: -85.463 Examples/s: 1.128e+07
```

The end

Evidence lower bound (ELBO) for one data point



$$\geq \int q_\phi(z|x) \log \left[\frac{p_\theta(x|z)\, p(z)}{q_\phi(z|x)} \right] \mathrm{d}z \qquad \text{[Jensen]}$$

$$= \int q_\phi(z|x) \log p_\theta(x|z)\, \mathrm{d}z - \int q_\phi(z|x) \log \left[\frac{q_\phi(z|x)}{p(z)} \right] \mathrm{d}z$$

$$= \mathbb{E}_{q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big] - \mathrm{KL}\big(q_\phi(z|x) \,\|\, p(z) \big)$$

$$\equiv \mathcal{L}(x; \theta, \phi)$$