▶ Notebook

Lecture 1 at a glance: five parts, twenty-two sections

Part 1 · Bayes' rule for a neural network

1.1 Bayes' rule ▶
1.2 A tiny network and its prior ▶
1.3 Likelihood and training ▶
1.4 The evidence is intractable ▶
1.5 Approximating the posterior ▶

Part 2 · How far apart are two distributions?

2.1 Surprise ▶
2.2 Entropy: average surprise ▶
2.3 Codes: entropy in bits ▶
2.4 Cross-entropy and KL ▶
2.5 KL at work ▶
  2.5.1 Classification ▶
  2.5.2 Knowledge distillation ▶
2.6 Closed forms and PyTorch ▶
2.7 KL by sampling ▶

Part 3 · Find the closest q

3.1 Search a grid ▶
3.2 Gradient descent ▶
3.3 Covariance in 2-D ▶
3.4 Mixtures: no formula ▶
3.5 The reparameterisation trick ▶
3.6 Which direction of KL? ▶

Part 4 · Back to Bayes: the ELBO

4.1 Deriving the ELBO ▶
4.2 Checking it on a coin ▶

Part 5 · Implement it

5.1 VI for the tiny network ▶
5.2 Wrap-up ▶

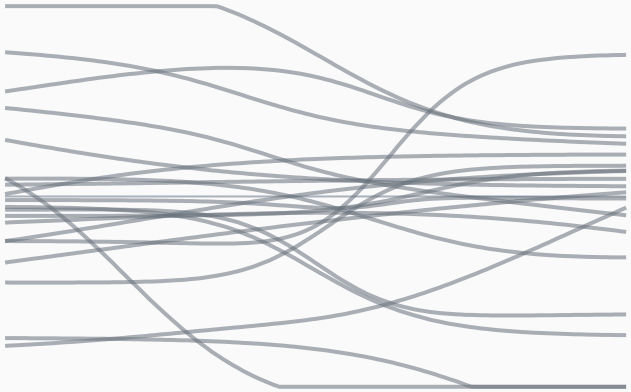
> Five parts, twenty-two numbered sections. The notebook uses the same numbers: the ▶ Notebook link at the top right of every slide opens its section in Colab.

▶ Notebook

Part 1 · Bayes' rule for a neural network

1 · Bayes for a network | 2 · Distance and KL | 3 · Find the closest q | 4 · Back to Bayes | 5 · Implement it

prior: networks before the data



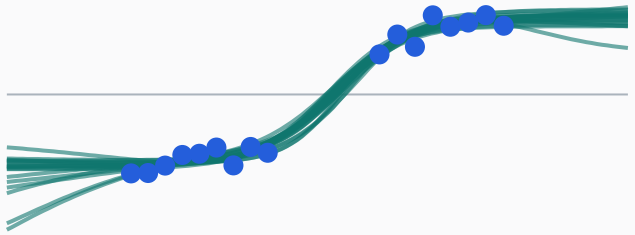
×

likelihood: the data



=

posterior: networks that fit



1.1 Bayes' rule ▶
1.2 A tiny network and its prior ▶
1.3 Likelihood and training ▶
1.4 The evidence is intractable ▶
1.5 Approximating the posterior ▶

> Part 1 asks what Bayes' rule says about a neural network's weights: start from a prior over networks, keep the ones that fit the data. The catch is one number, the evidence.

▶ Notebook 1.1

Bayes' rule: update what you believe, using data

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{p(B)}$$

> Bayes' rule: what to believe after seeing B, from what we believed before and how well each belief explains B.

▶ Notebook 1.1

Bayes' rule: update what you believe, using data

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{p(B)}$$

$$\underbrace{p(A \mid B)}_{\text{posterior}} = \frac{\overbrace{p(B \mid A)}^{\text{likelihood}}\ \overbrace{p(A)}^{\text{prior}}}{\underbrace{p(B)}_{\text{evidence}}}$$

> Four names for the whole lecture: posterior, likelihood, prior and evidence.

▶ Notebook 1.1

In machine learning, the unknown is the parameters

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}$$

A becomes θ, every weight and bias of the model; B becomes $\mathcal{D}$, the training data.

▶ Notebook 1.1

In machine learning, the unknown is the parameters

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}$$

Usual training

keeps one θ: the best fit to the data $\mathcal{D}$.

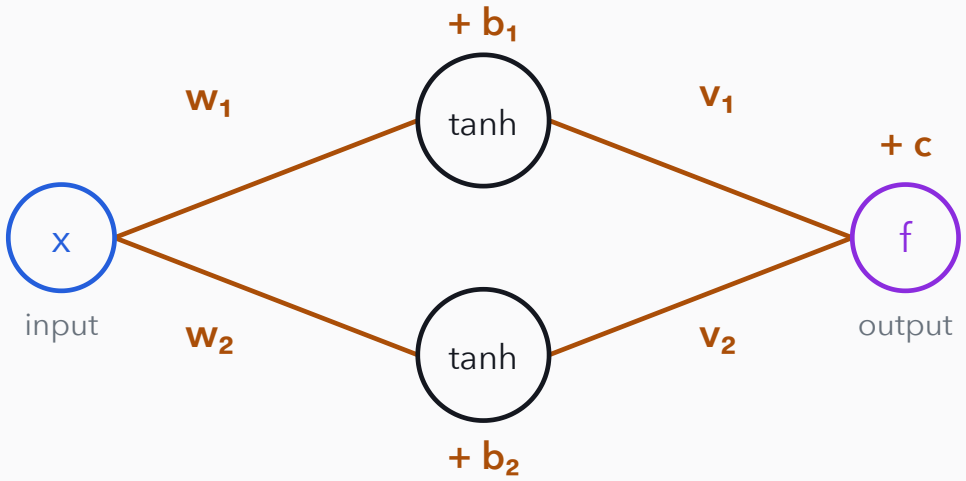
Bayesian learning

keeps every θ, weighted by $p(\theta \mid \mathcal{D})$.

> Usual training keeps one θ. The Bayesian answer keeps a whole distribution over θ.

▶ Notebook 1.2

A tiny network: one input, two hidden units, one output

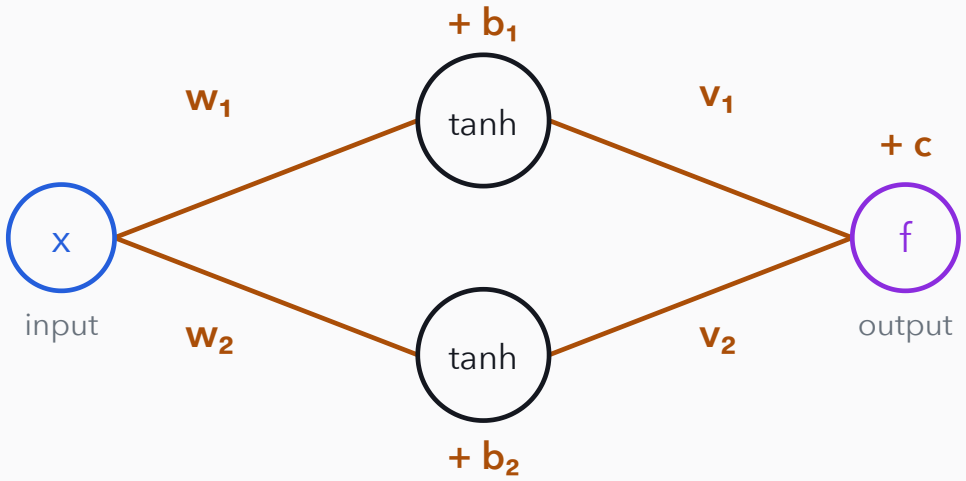


$$f_\theta(x) = v_1 \tanh(w_1 x + b_1) + v_2 \tanh(w_2 x + b_2) + c$$

Our running example: one input x, two tanh hidden units and one output f.

▶ Notebook 1.2

A tiny network: one input, two hidden units, one output



$$f_\theta(x) = v_1 \tanh(w_1 x + b_1) + v_2 \tanh(w_2 x + b_2) + c$$

$$\theta = (w_1, w_2, b_1, b_2, v_1, v_2, c) \in \mathbb{R}^7$$

Seven unknown numbers: four weights and three biases.

> Every weight and bias is unknown, so θ is seven numbers.

▶ Notebook 1.2

A prior on every parameter

$$\theta_j \sim \mathcal{N}(0, 1), \quad j = 1, \ldots, 7$$

$$p(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, 0,\, 1)$$

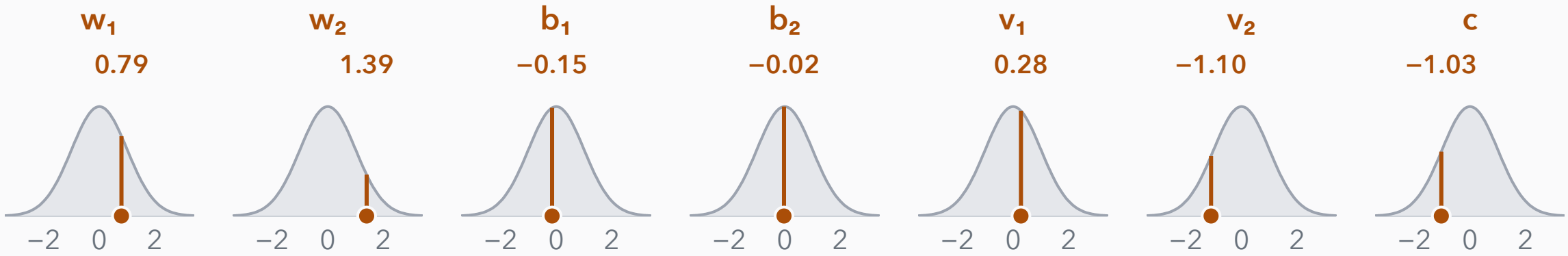
Before seeing data, every parameter gets its own prior: $\theta_j \sim N(0, 1)$, independently.

▶ Notebook 1.2

A prior on every parameter

$$\theta_j \sim \mathcal{N}(0, 1), \quad j = 1, \ldots, 7$$

$$p(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, 0,\, 1)$$



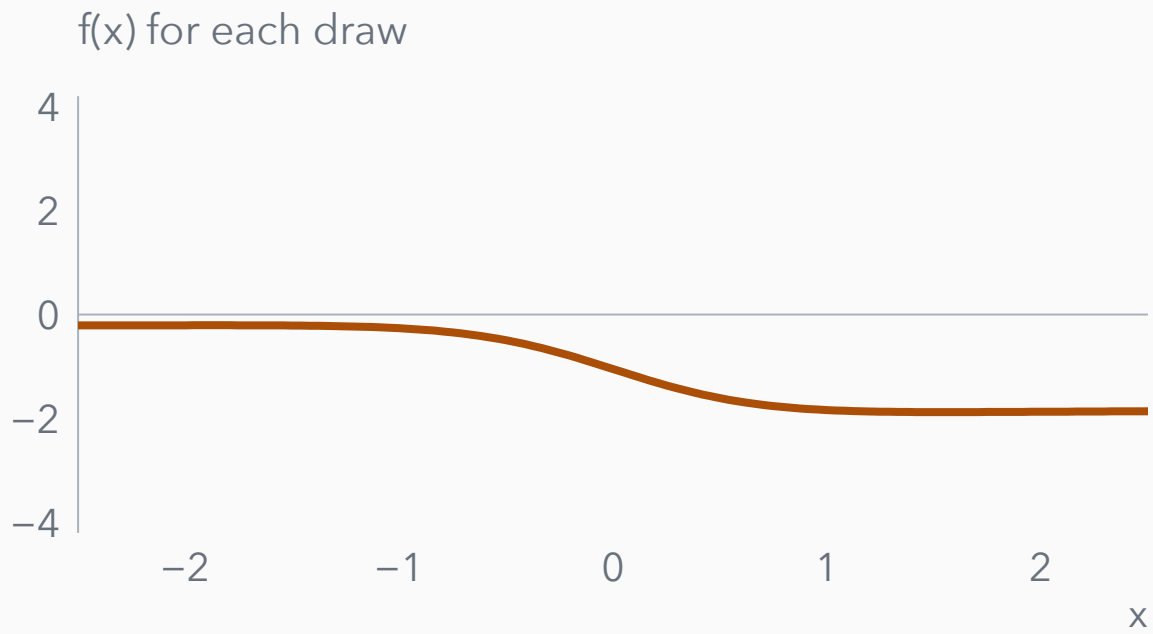
One draw from the prior is seven numbers, one from each bell: a single point θ in $\mathbb{R}^7$.

▶ Notebook 1.2

One draw of θ is one whole function

	w_1	w_2	b_1	b_2	v_1	v_2	c
$\theta^{(1)}$	0.79	1.39	−0.15	−0.02	0.28	−1.10	−1.03

Each row, put through the network, is one curve.



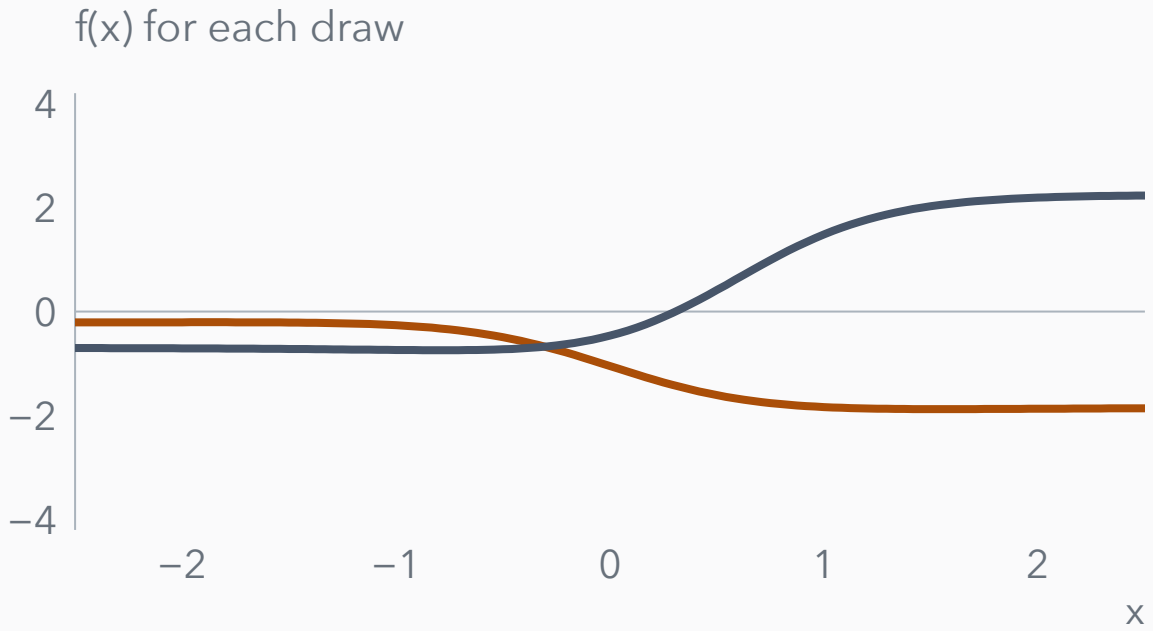
> Draw $\theta^{(1)}$ from the prior, put its seven numbers into the network, and evaluate f at every x: one curve.

▶ Notebook 1.2

One draw of θ is one whole function

	w_1	w_2	b_1	b_2	v_1	v_2	c
$\theta^{(1)}$	0.79	1.39	−0.15	−0.02	0.28	−1.10	−1.03
$\theta^{(2)}$	−1.07	−1.40	0.04	0.78	0.45	−1.93	0.77

Each row, put through the network, is one curve.



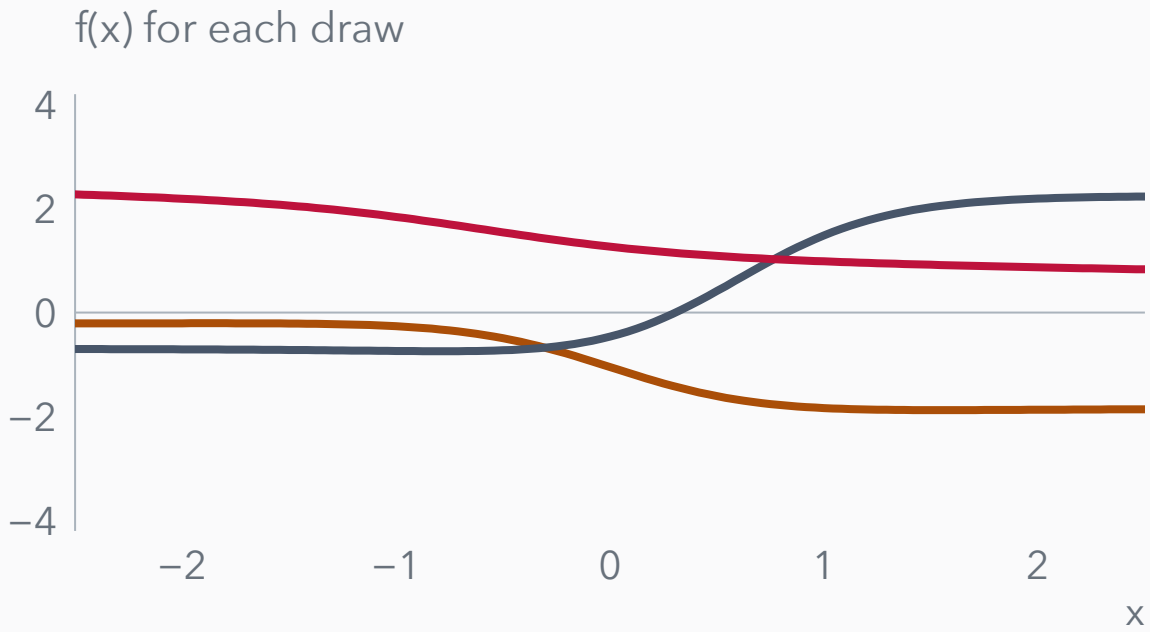
A second draw, $\theta^{(2)}$, gives a different curve.

▶ Notebook 1.2

One draw of θ is one whole function

	w_1	w_2	b_1	b_2	v_1	v_2	c
$\theta^{(1)}$	0.79	1.39	−0.15	−0.02	0.28	−1.10	−1.03
$\theta^{(2)}$	−1.07	−1.40	0.04	0.78	0.45	−1.93	0.77
$\theta^{(3)}$	−1.06	0.20	−0.63	0.22	0.48	−0.56	1.66

Each row, put through the network, is one curve.

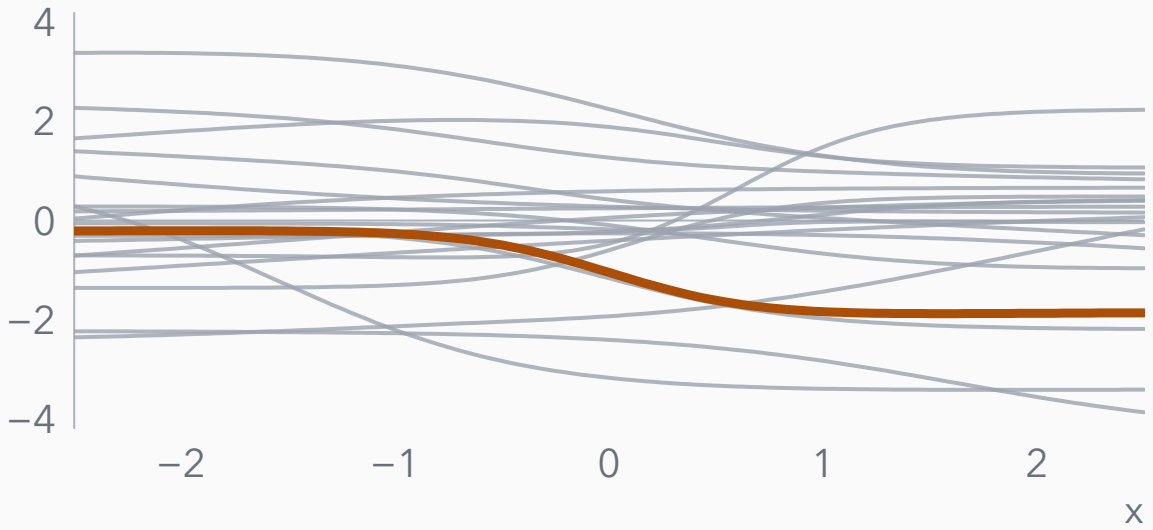


> Every point of $\mathbb{R}^7$ is a whole function. Drawing θ from the prior draws functions.

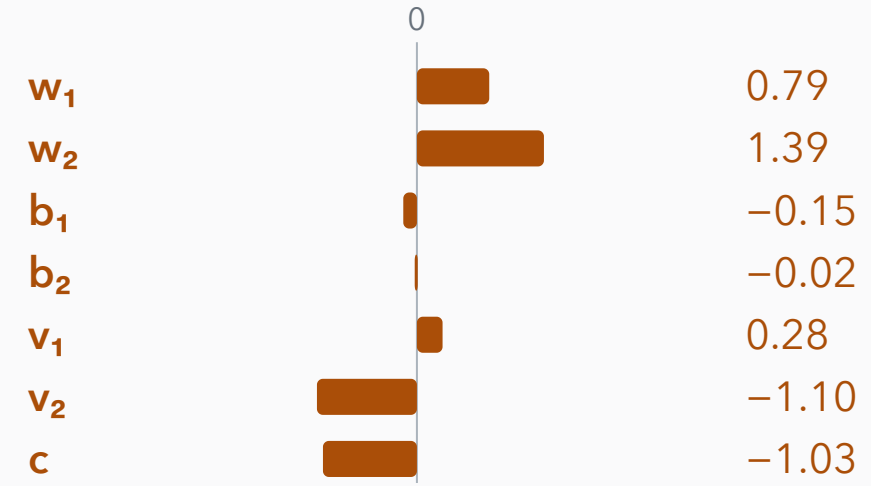
▶ Notebook 1.2

Twenty draws, and the prior predictive

f(x) for 20 draws of θ from the prior



draw 1 of 20: $\theta^{(1)} \in \mathbb{R}^7$



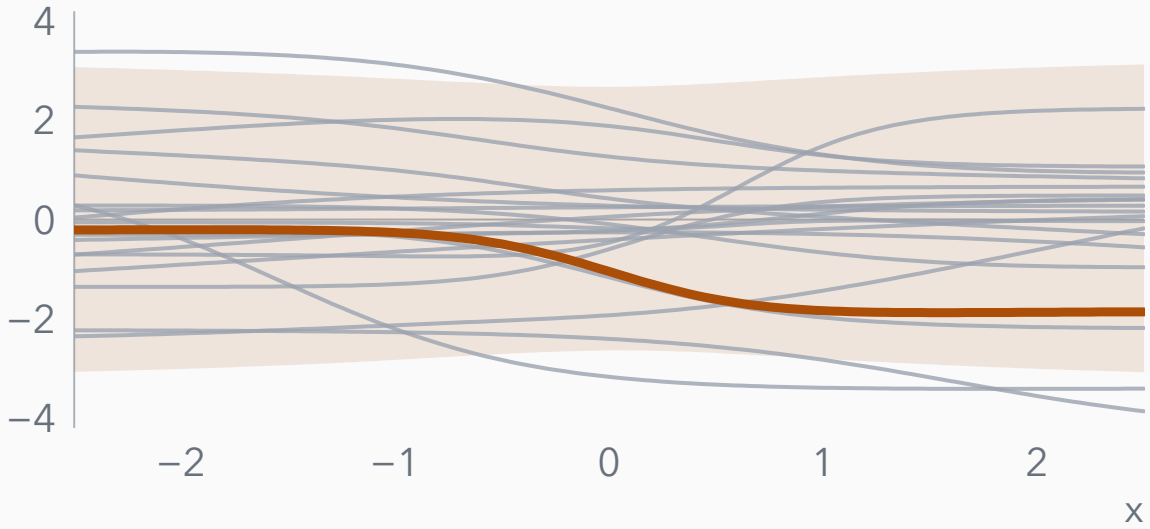
Hover over a line, or step through the draws.

Twenty draws of θ, twenty functions. Hover over a line, or step through, to see its point in $\mathbb{R}^7$.

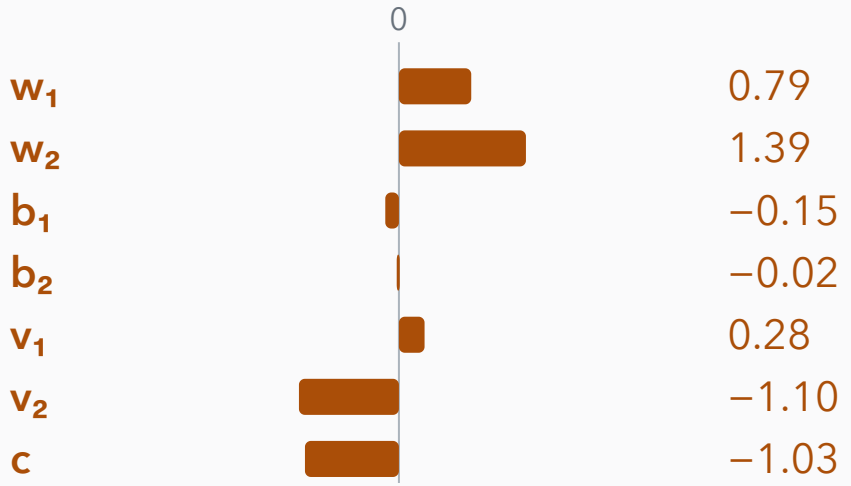
▶ Notebook 1.2

Twenty draws, and the prior predictive

20 draws, and the prior predictive: mean ± 2 sd



draw 1 of 20: $\theta^{(1)} \in \mathbb{R}^7$

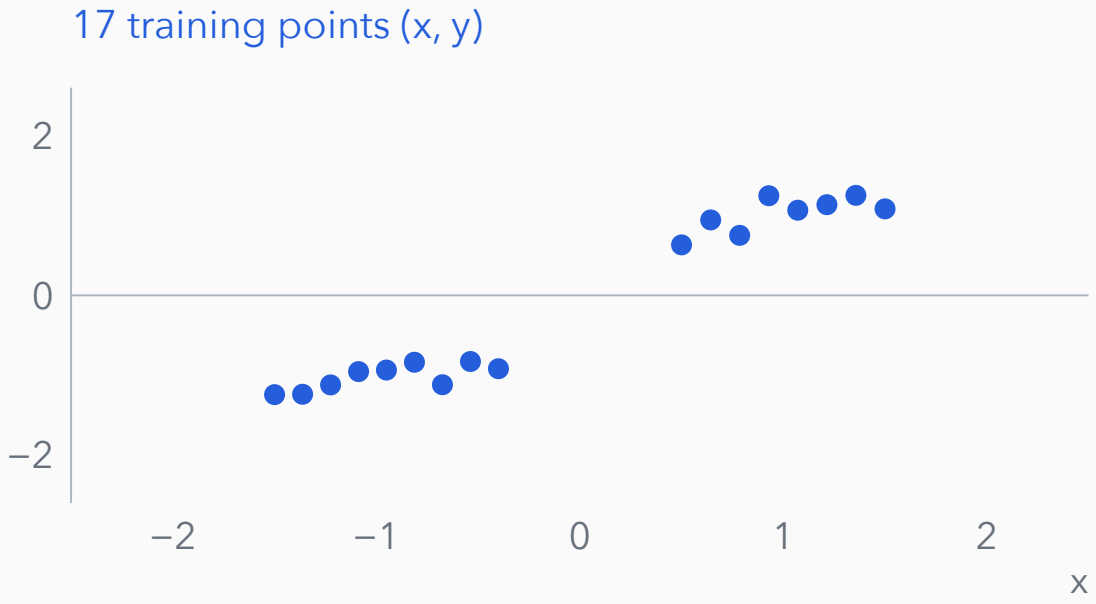


Hover over a line, or step through the draws.

> Average over every θ and add the 0.1 noise: the prior predictive, $p(y \mid x) = \int p(y \mid x, \theta)\, p(\theta)\, d\theta$. What the model expects before any data.

▶ Notebook 1.3

The data, and how well one θ explains them

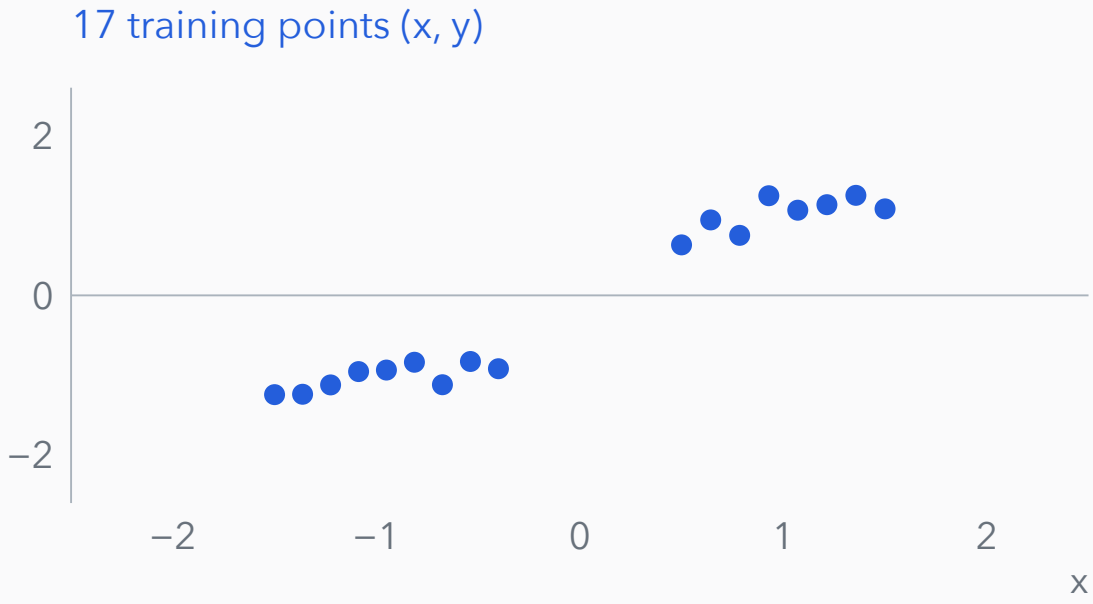


$$y_i \mid x_i, \theta \sim \mathcal{N}\left(f_\theta(x_i),\, 0.1^2\right)$$

> Seventeen training points with a gap in the middle. Each y is the network's output plus Gaussian noise of size 0.1.

▶ Notebook 1.3

The data, and how well one θ explains them



$$y_i \mid x_i, \theta \sim \mathcal{N}\big(f_\theta(x_i),\, 0.1^2\big)$$

$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^{17} \mathcal{N}\big(y_i;\, f_\theta(x_i),\, 0.1^2\big)$$

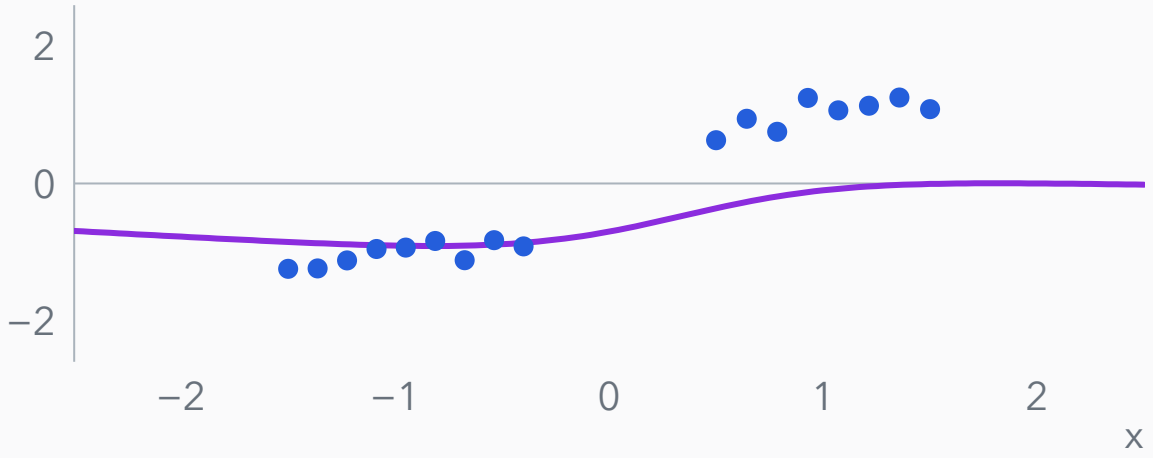
> The likelihood of θ multiplies all seventeen densities: how well this one θ explains the whole dataset.

▶ Notebook 1.3

Finding a good θ: train the network

$$\theta_A = \arg\max_{\theta} \left[\log p(\mathcal{D} \mid \theta) + \log p(\theta) \right]$$

the network during training



loss 559 · grey: the other nine starts



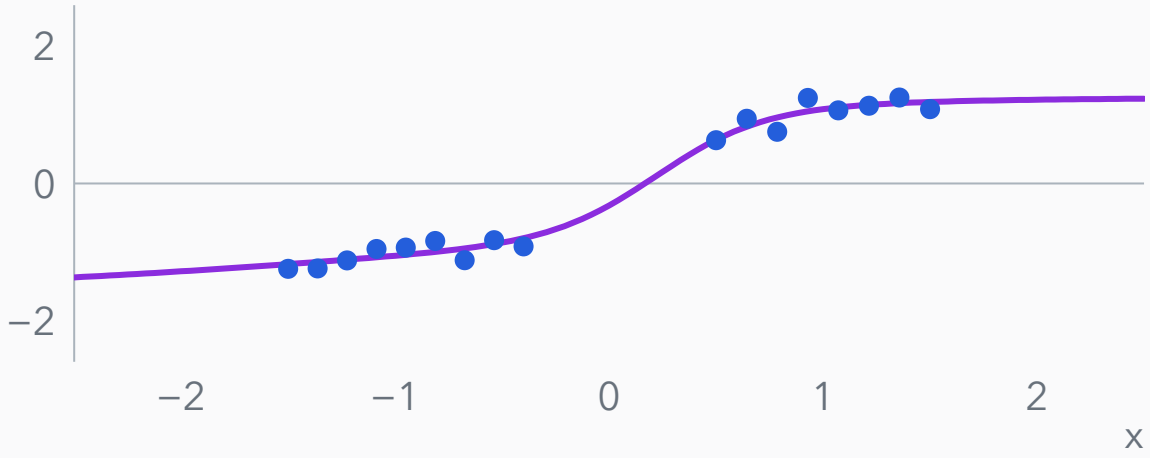
To pick one good θ, train the network the usual way: maximise log p($\mathcal{D}$ | θ) + log p(θ). That is squared error plus weight decay.

▶ Notebook 1.3

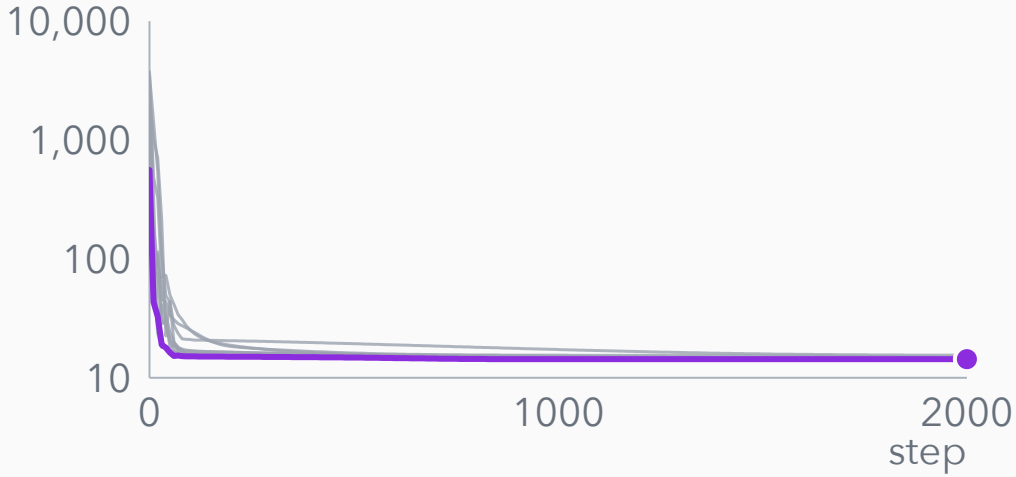
Finding a good θ: train the network

$$\theta_A = \arg\max_{\theta} \left[\log p(\mathcal{D} \mid \theta) + \log p(\theta) \right]$$

the network during training



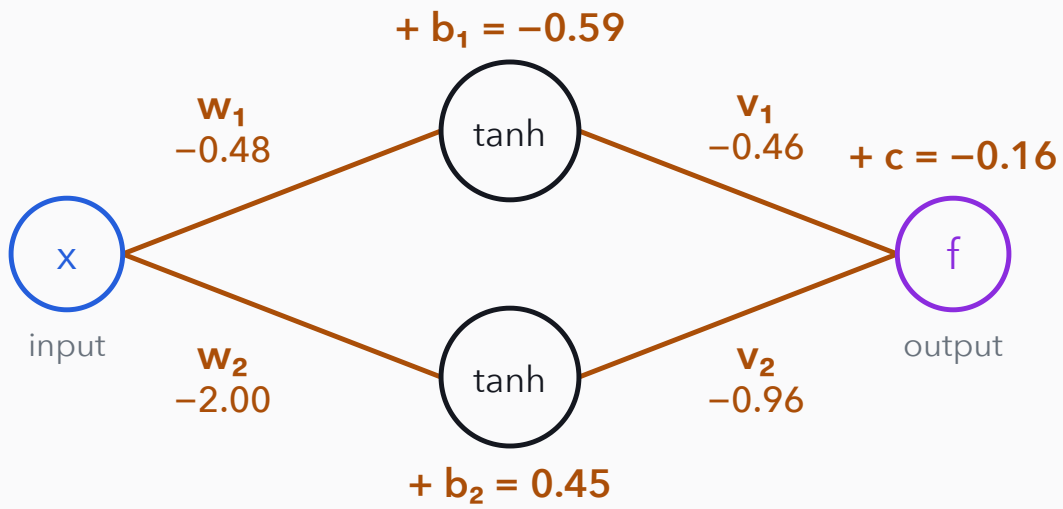
loss 14.3 · grey: the other nine starts



From a random start, gradient descent bends the curve onto the data. Ten starts finish at slightly different losses; the best finish is θ_A.

▶ Notebook 1.3

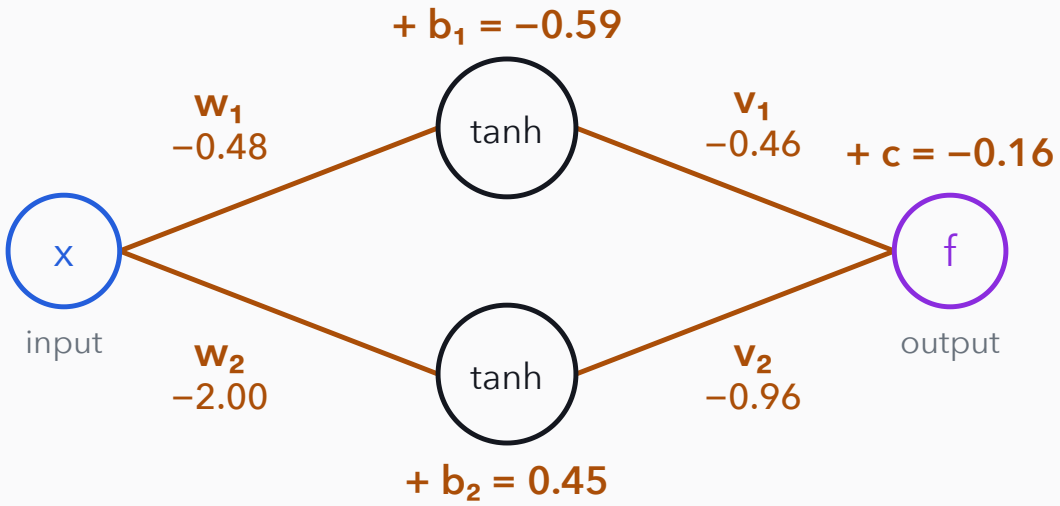
Meet θ_A: the network with its trained numbers

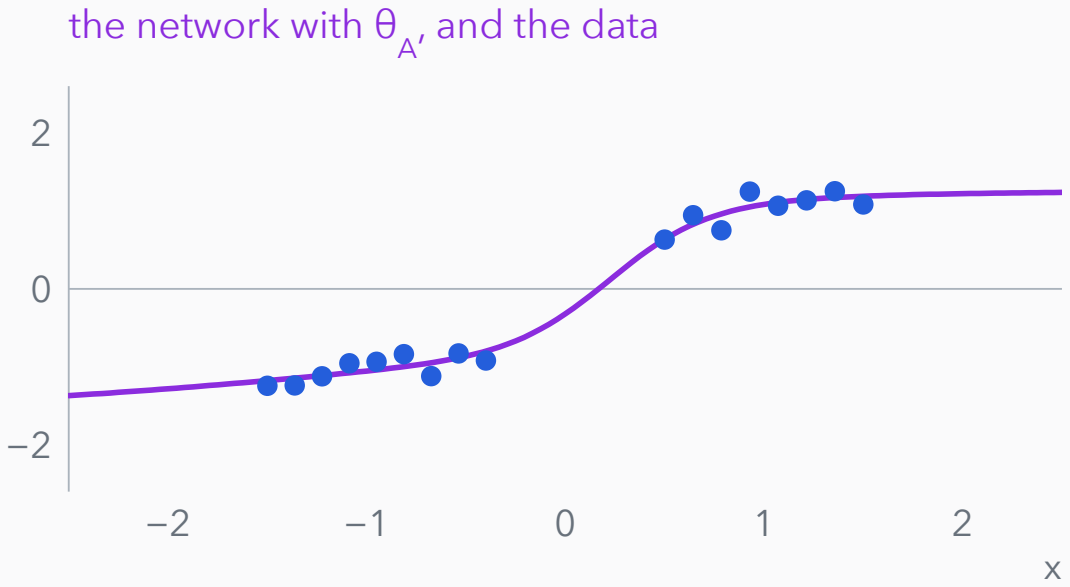


θ_A is one point in $\mathbb{R}^7$: these seven numbers, written on the network.

▶ Notebook 1.3

Meet θ_A: the network with its trained numbers





With these numbers the network's curve follows the data. Next, score θ_A with the likelihood and the prior.

▶ Notebook 1.3

The likelihood of θ_A: formula, then value

$$p(\mathcal{D} \mid \theta_A) = \prod_{i=1}^{17} \mathcal{N}\left(y_i;\ f_{\theta_A}(x_i),\ 0.1^2\right)$$

Fix $\theta = \theta_A$. The likelihood multiplies one Gaussian density per data point: how well the curve explains each y.

▶ Notebook 1.3

The likelihood of θ_A: formula, then value

$$p(\mathcal{D} \mid \theta_A) = \prod_{i=1}^{17} \mathcal{N}\left(y_i;\, f_{\theta_A}(x_i),\, 0.1^2\right)$$

$$\log p(\mathcal{D} \mid \theta_A) = -\frac{1}{2 \cdot 0.1^2} \sum_{i=1}^{17} \left(y_i - f_{\theta_A}(x_i)\right)^2 - \frac{17}{2} \log\left(2\pi \cdot 0.1^2\right)$$

Take logs: the product becomes a sum of squared errors, scaled by the noise, plus a constant.

▶ Notebook 1.3

The likelihood of θ_A: formula, then value

$$p(\mathcal{D} \mid \theta_A) = \prod_{i=1}^{17} \mathcal{N}\left(y_i;\, f_{\theta_A}(x_i),\, 0.1^2\right)$$

$$\log p(\mathcal{D} \mid \theta_A) = -\frac{1}{2 \cdot 0.1^2} \sum_{i=1}^{17} \left(y_i - f_{\theta_A}(x_i)\right)^2 - \frac{17}{2} \log\left(2\pi \cdot 0.1^2\right)$$

$$\log p(\mathcal{D} \mid \theta_A) = 12.17$$

Put in θ_A's curve and the seventeen points: log p($\mathcal{D}$ | θ_A) = 12.17.

▶ Notebook 1.3

The prior of θ_A, then the evidence

$$p(\theta_A) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, 0,\, 1)$$

The prior multiplies one N(0, 1) density per parameter.

▶ Notebook 1.3

The prior of θ_A, then the evidence

$$p(\theta_A) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, 0,\, 1)$$

$$\log p(\theta_A) = -\frac{1}{2}\sum_{j=1}^{7} \theta_j^2 - \frac{7}{2}\log 2\pi = -9.41$$

> Take logs: minus half the sum of squared parameters, plus a constant. For θ_A it is −9.41; the large w_2 costs the most.

▶ Notebook 1.3

The prior of θ_A, then the evidence

$$p(\theta_A) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, 0,\, 1)$$

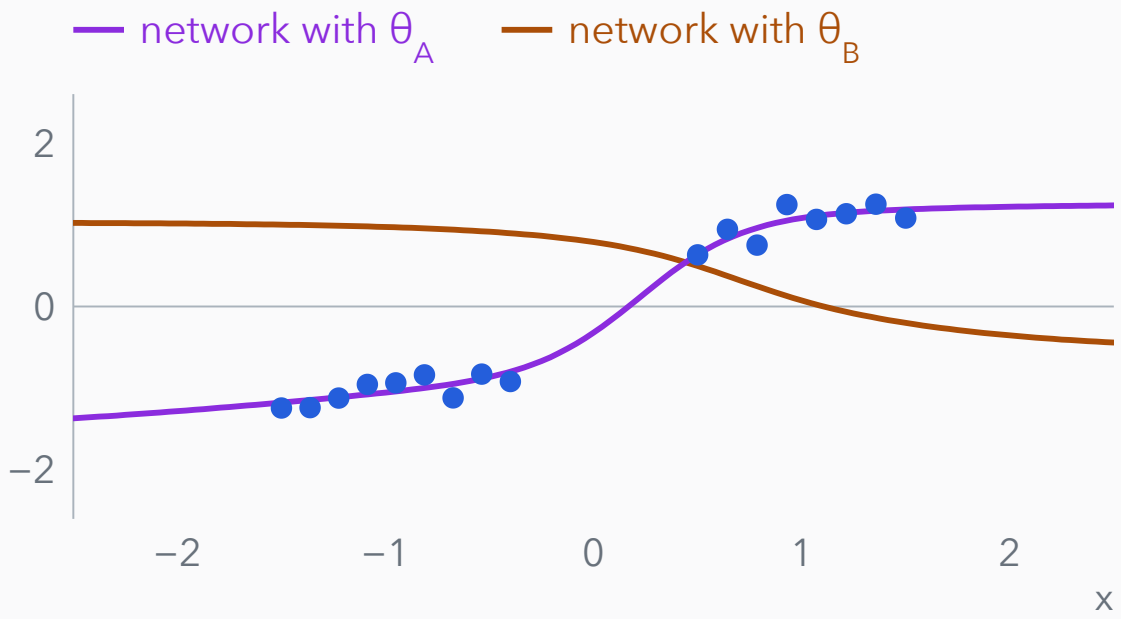
$$\log p(\theta_A) = -\frac{1}{2}\sum_{j=1}^{7} \theta_j^2 - \frac{7}{2}\log 2\pi = -9.41$$

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)\, p(\theta)\, d\theta = \ \mathbf{?}$$

> The third term, the evidence, is not about θ_A: it needs every θ at once.

▶ Notebook 1.3

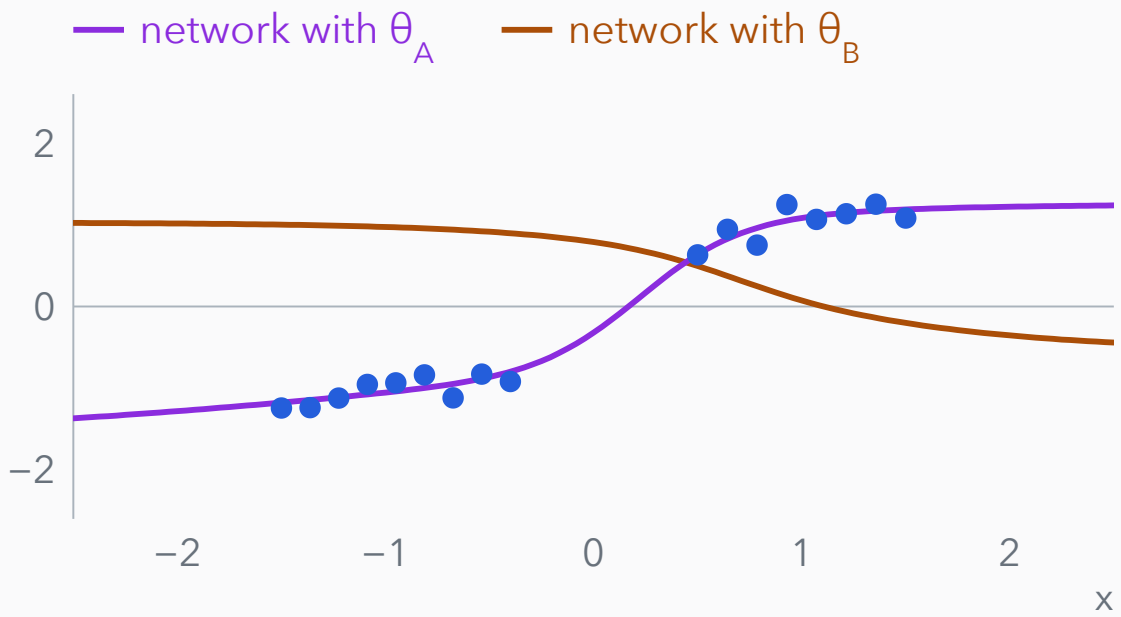
Two settings of θ, side by side



θ_A follows the data. θ_B, one random draw from the prior, does not.

▶ Notebook 1.3

Two settings of θ, side by side

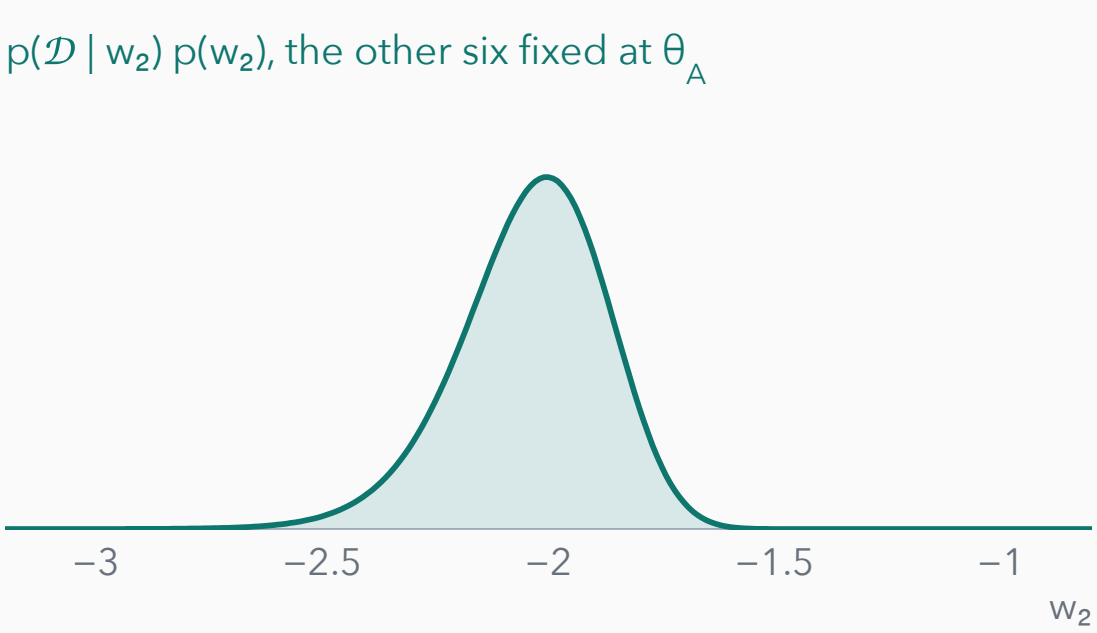


	θ_A	θ_B, a prior draw
$\log p(\mathcal{D} \mid \theta)$	12.2	−2,157.5
$\log p(\theta)$	−9.4	−9.9
sum of the two	2.8	−2,167.4

> Their priors are similar; their likelihoods differ by over 2,000 nats. A ratio of two posteriors does not even need p($\mathcal{D}$).

▶ Notebook 1.4

The evidence is an area: add up thin rectangles



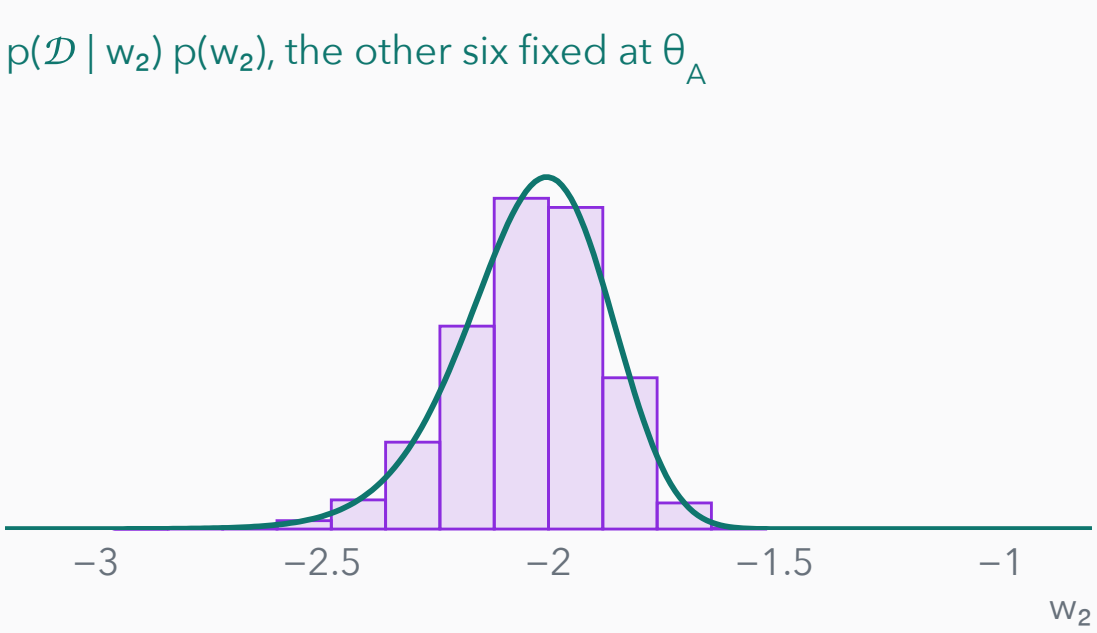
Suppose only w_2 were unknown, with the other six numbers fixed at θ_A:

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid w_2)\, p(w_2)\, dw_2$$

With one unknown, the evidence is the area under one curve: likelihood times prior.

▶ Notebook 1.4

The evidence is an area: add up thin rectangles



Suppose only w_2 were unknown, with the other six numbers fixed at θ_A:

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid w_2)\, p(w_2)\, dw_2$$

$$\approx \sum_{k=1}^{20} p(\mathcal{D} \mid w_{2,k})\, p(w_{2,k})\, \Delta w$$

> Chop the area into thin rectangles and add them up. Twenty rectangles already match a much finer grid to three decimals.

▶ Notebook 1.4

Intractable: why the rectangles do not scale

1 unknown: 100 points

d unknowns: $100 \times 100 \times \cdots \times 100 = 100^{d}$ points

unknowns	grid points	at 10^9 a second
1	100	instant

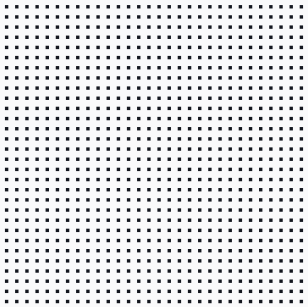
> To add up the evidence on a grid, put 100 points along each unknown. One unknown: 100 evaluations.

▶ Notebook 1.4

Intractable: why the rectangles do not scale

1 unknown: 100 points

2 unknowns



$100 \times 100 = 10^4$

d unknowns: $100 \times 100 \times \cdots \times 100 = 100^d$ points

unknowns	grid points	at 10^9 a second
1	100	instant
2	$100^2 = 10^4$	instant

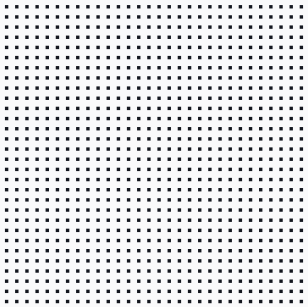
> Two unknowns: each of the 100 values of the first pairs with each of the 100 values of the second. $100 \times 100 = 10{,}000$.

▶ Notebook 1.4

Intractable: why the rectangles do not scale

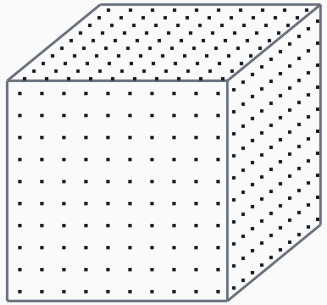
1 unknown: 100 points

2 unknowns



$100 \times 100 = 10^4$

3 unknowns



$100 \times 100 \times 100 = 10^6$

d unknowns: $100 \times 100 \times \cdots \times 100 = 100^d$ points

unknowns	grid points	at 10^9 a second
1	100	instant
2	$100^2 = 10^4$	instant
3	$100^3 = 10^6$	a millisecond

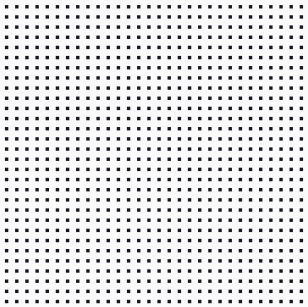
> Three unknowns: 100 × 100 × 100 = a million. Every extra unknown multiplies the count by another 100.

▶ Notebook 1.4

Intractable: why the rectangles do not scale

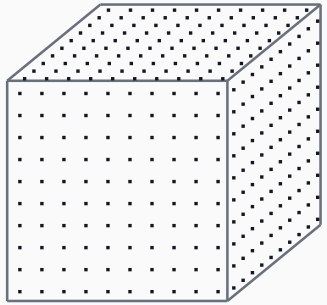
1 unknown: 100 points

2 unknowns



$100 \times 100 = 10^4$

3 unknowns



$100 \times 100 \times 100 = 10^6$

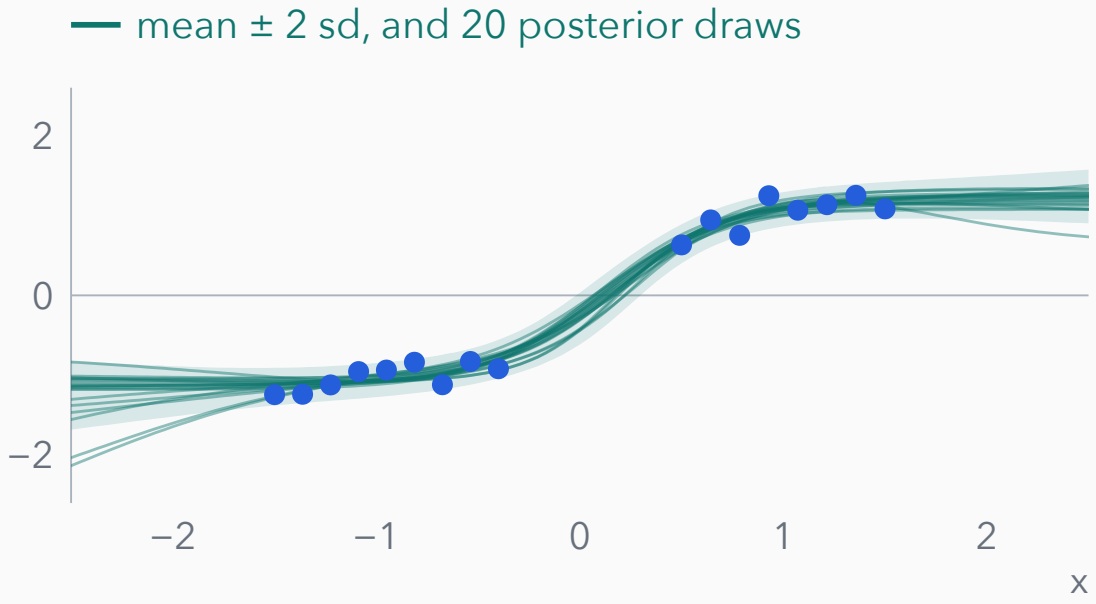
d unknowns: $100 \times 100 \times \cdots \times 100 = 100^d$ points

unknowns	grid points	at 10^9 a second
1	100	instant
2	$100^2 = 10^4$	instant
3	$100^3 = 10^6$	a millisecond
7 (our network)	$100^7 = 10^{14}$	≈ 28 hours
10^6 (a real one)	$10^{2,000,000}$	never

> Our seven: $100^7 = 10^{14}$, about 28 hours at a billion a second. A real network: $10^{2,000,000}$, never. That is what intractable means.

▶ Notebook 1.5

What we want in the end: predictions with error bars

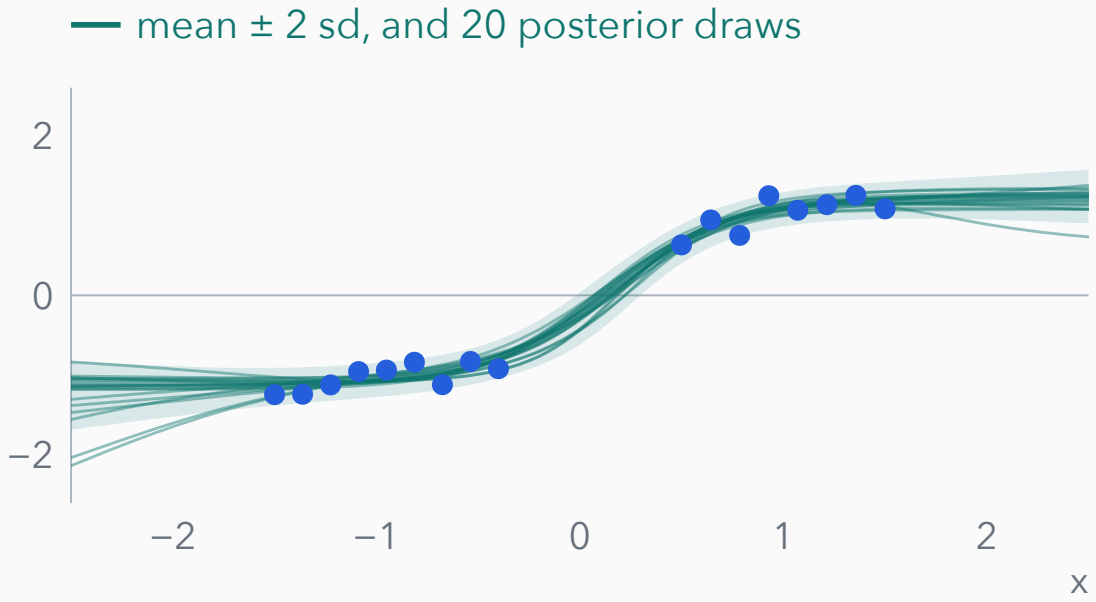


$$p(y^* \mid x^*, \mathcal{D}) = \int p(y^* \mid x^*, \theta) \times p(\theta \mid \mathcal{D})\, d\theta$$

> The posterior is a means to an end. The posterior predictive averages every network's prediction at a new x*.

▶ Notebook 1.5

What we want in the end: predictions with error bars



$$p(y^* \mid x^*, \mathcal{D}) = \int p(y^* \mid x^*, \theta) \times p(\theta \mid \mathcal{D})\, d\theta$$

Average every network's prediction, weighted by the posterior. The band widens where there are no data.

> Here it comes from a long MCMC run, used only as a reference. The band widens beyond the data and in the gap.

▶ Notebook 1.5

The core problem of probabilistic machine learning

$$p(\theta \mid \mathcal{D}) = \frac{\overbrace{p(\mathcal{D} \mid \theta)}^{\text{easy}} \; \overbrace{p(\theta)}^{\text{easy}}}{\underbrace{p(\mathcal{D})}_{\text{intractable}}}$$

> The likelihood and the prior are easy to evaluate for any θ. The evidence, which turns them into the posterior, is not.

▶ Notebook 1.5

The core problem of probabilistic machine learning

$$p(\theta \mid \mathcal{D}) = \frac{\overbrace{p(\mathcal{D} \mid \theta)}^{\text{easy}} \; \overbrace{p(\theta)}^{\text{easy}}}{\underbrace{p(\mathcal{D})}_{\text{intractable}}}$$

How do we compute, or approximate, the posterior $p(\theta \mid \mathcal{D})$?

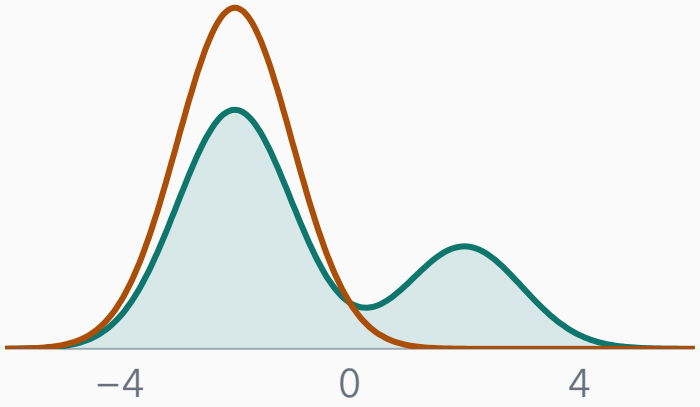
Bayesian neural networks, VAEs, mixture models and topic models all meet this same question.

> So the central question of probabilistic machine learning: how do we compute, or approximate, the posterior?

▶ Notebook 1.5

Three ways to approximate the posterior; we take the third

Laplace



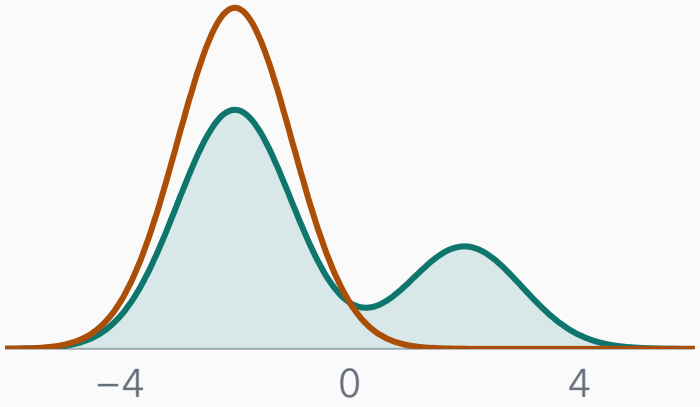
1 climb to the highest peak, θ_{MAP}. **2** measure how sharply log p bends there. **3** the Gaussian with that peak and that bend: $N(-2.0, 1.0^2)$. It never sees the second bump.

> Laplace: climb to the highest peak, measure how sharply it bends, and use the Gaussian with that peak and that bend. Fast, but it only sees the peak it climbed to.

▶ Notebook 1.5

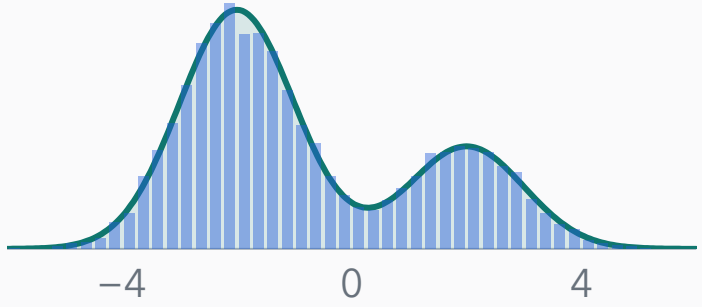
Three ways to approximate the posterior; we take the third

Laplace



1 climb to the highest peak, θ_{MAP}. **2** measure how sharply log p bends there. **3** the Gaussian with that peak and that bend: $N(-2.0, 1.0^2)$. It never sees the second bump.

MCMC



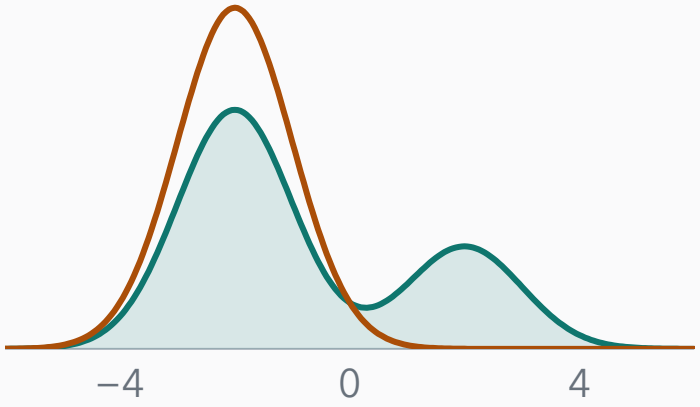
A walker proposes random steps. Uphill: always move; downhill: move with probability $p(\theta')/p(\theta)$, so the evidence cancels. Its visits follow the target: mean −0.80, true −0.80.

> MCMC: a walker proposes random steps. Uphill it always moves; downhill only sometimes, with probability $p(\theta')/p(\theta)$. Its visits follow the posterior: right in the long run, but slow.

▶ Notebook 1.5

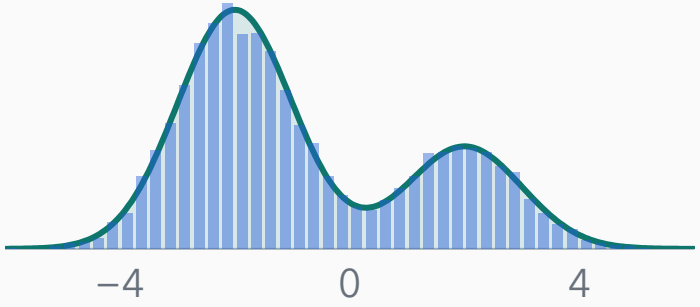
Three ways to approximate the posterior; we take the third

Laplace



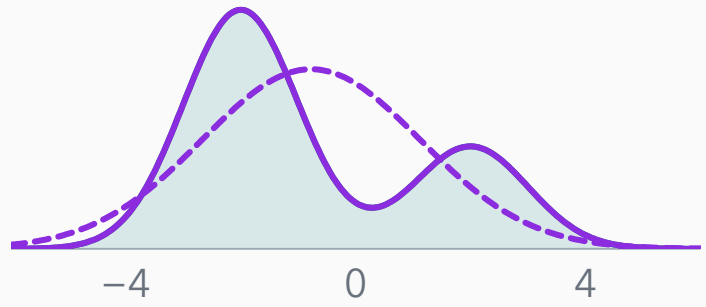
1 climb to the highest peak, θ_{MAP}. **2** measure how sharply log p bends there. **3** the Gaussian with that peak and that bend: $N(-2.0, 1.0^2)$. It never sees the second bump.

MCMC



A walker proposes random steps. Uphill: always move; downhill: move with probability $p(\theta')/p(\theta)$, so the evidence cancels. Its visits follow the target: mean −0.80, true −0.80.

Variational inference



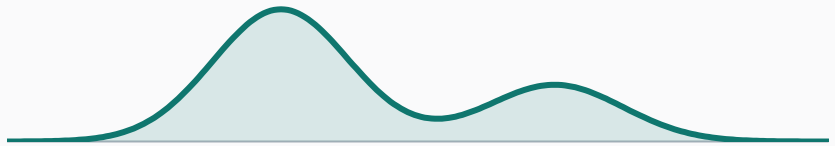
Choose a family, then tune its knobs to make KL(q ‖ p) small. One Gaussian (dashed) must stretch across both bumps; a mixture of two (solid) matches them.

> Variational inference: choose a family, then find its member closest to the posterior by optimisation. The family need not be Gaussian. The rest of the lecture develops this one.

▶ Notebook 1.5

Variational inference: choose q, make it close to the posterior

the posterior $p(\theta \mid \mathcal{D})$



what we need: predictions, error bars

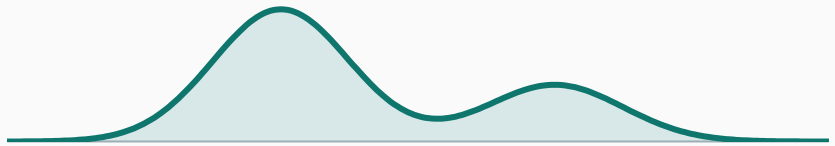
but $p(\mathcal{D})$ is intractable: we cannot compute it

> What we need is the posterior: every prediction with error bars averages over it. But we cannot compute it, because the evidence $p(\mathcal{D})$ is intractable.

▶ Notebook 1.5

Variational inference: choose q, make it close to the posterior

the posterior $p(\theta \mid \mathcal{D})$



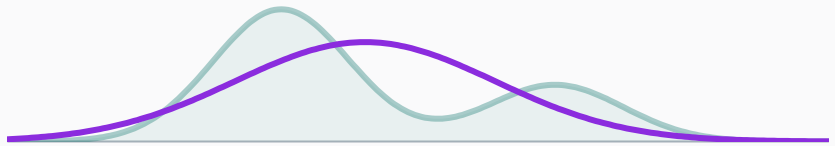
what we need: predictions, error bars

but $p(\mathcal{D})$ is intractable: we cannot compute it

tune ψ to make q

as close as possible

$q_\psi(\theta)$: a distribution we control



its knobs: $\psi = (\mu, \sigma)$; $\theta \sim q$ is easy to draw

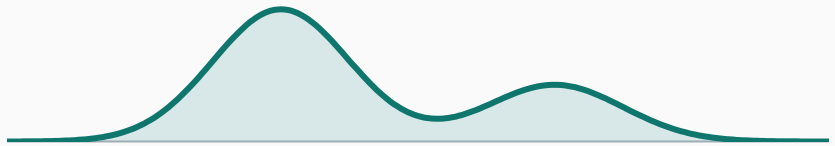
also called the guide (in Pyro)

So choose a distribution $q_\psi(\theta)$ that we control through its knobs ψ, and tune ψ until q_ψ is as close as possible to the posterior. Pyro calls q_ψ the guide.

▶ Notebook 1.5

Variational inference: choose q, make it close to the posterior

the posterior p(θ | $\mathcal{D}$)



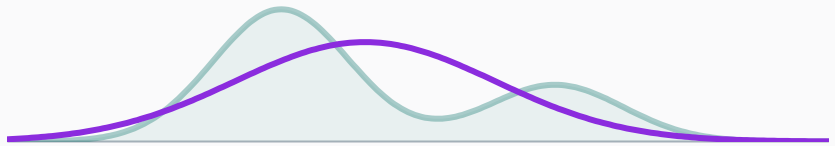
what we need: predictions, error bars

but p($\mathcal{D}$) is intractable: we cannot compute it

tune ψ to make q

as close as possible

q_ψ(θ): a distribution we control



its knobs: ψ = (μ, σ); θ ~ q is easy to draw

also called the guide (in Pyro)

$$p(y^* \mid x^*, \mathcal{D}) = \int p(y^* \mid x^*, \theta)\, p(\theta \mid \mathcal{D})\, d\theta \qquad \text{needs the posterior}$$

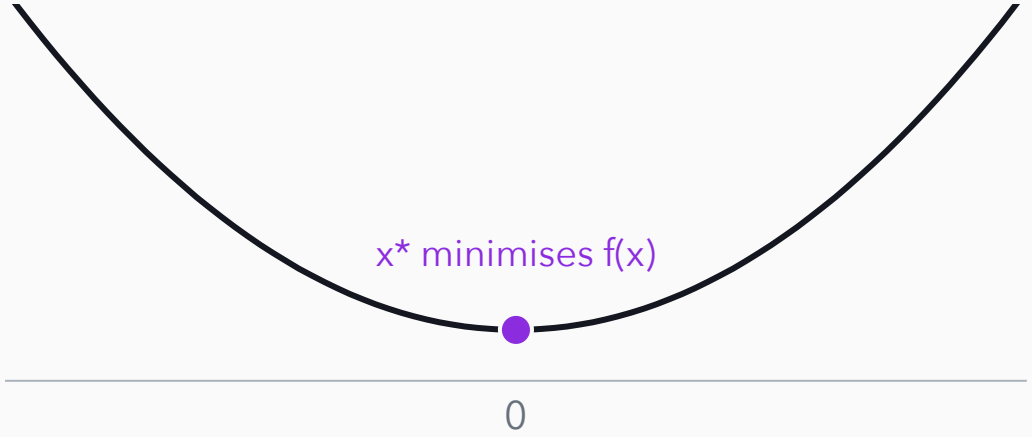
$$\approx \frac{1}{S} \sum_{s=1}^{S} p(y^* \mid x^*, \theta_s), \quad \theta_s \sim q_\psi(\theta) \qquad \text{needs only } q_\psi$$

> Then use q_ψ wherever the posterior was needed. For predictions: draw networks from q_ψ and average their outputs.

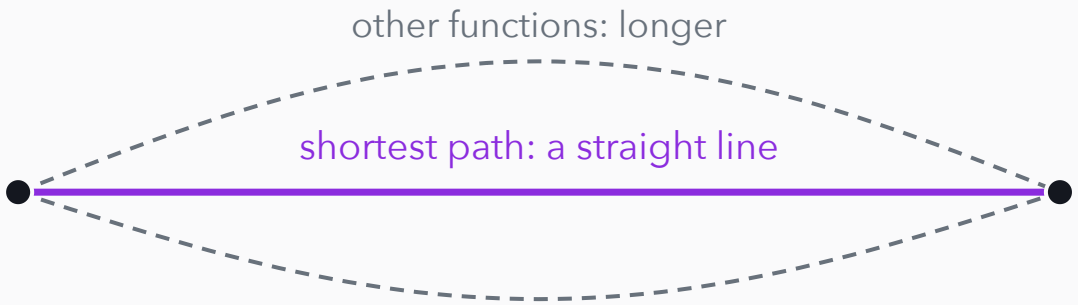
▶ Notebook

Why “variational”? Optimising over functions

calculus: the best number



calculus of variations: the best function

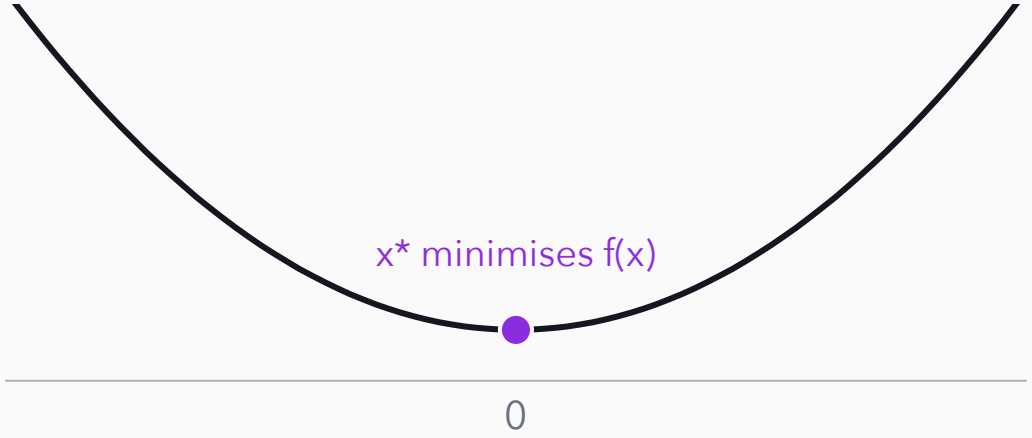


What is the unknown in ordinary calculus? In the calculus of variations?

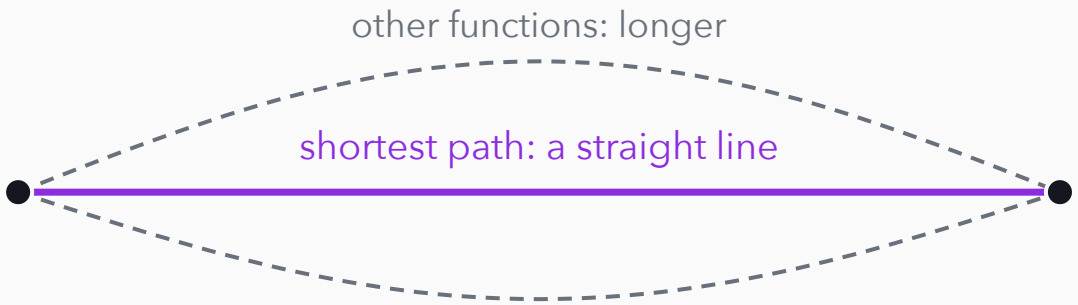
▶ Notebook

Why “variational”? Optimising over functions

calculus: the best number



calculus of variations: the best function



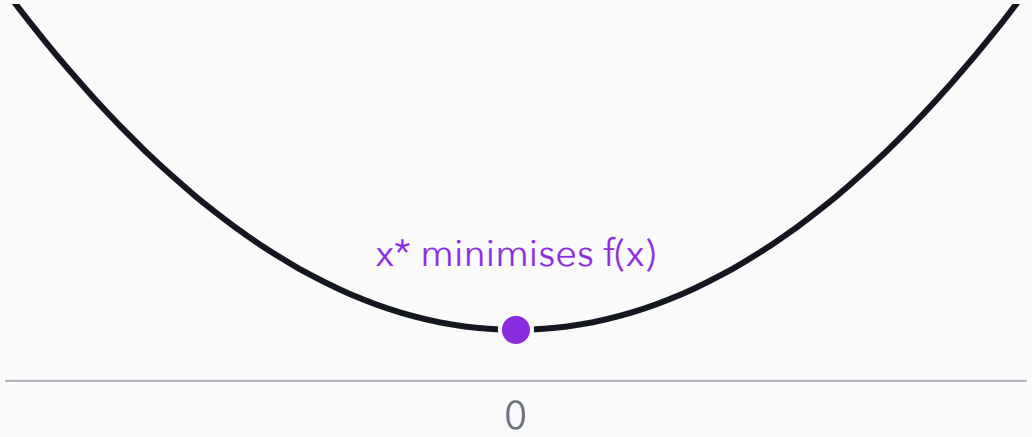
$$\text{VI:} \quad q^* = \arg\min_q \ \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) \qquad \text{over distributions } q\text{: functions}$$

What is the unknown in VI?

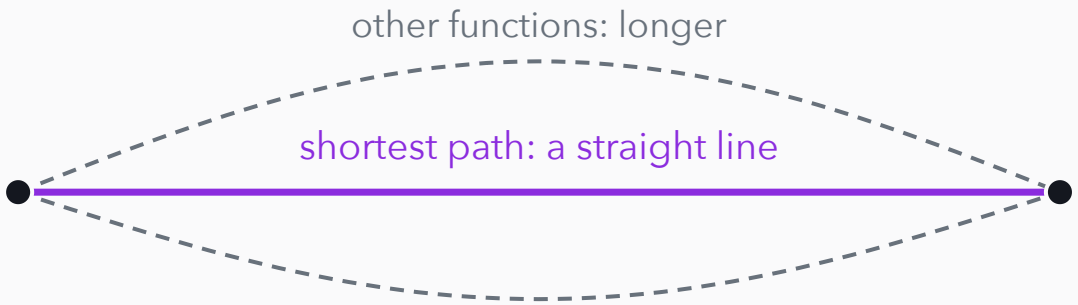
▶ Notebook

Why “variational”? Optimising over functions

calculus: the best number



calculus of variations: the best function



$$\text{VI:} \quad q^* = \arg\min_q \ \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) \qquad \text{over distributions } q\text{: functions}$$

$$\text{in practice:} \quad \psi^* = \arg\min_\psi \ \mathrm{KL}\big(q_\psi \,\|\, p(\theta \mid \mathcal{D})\big) \qquad \text{over the knobs } \psi\text{: numbers}$$

> How does VI turn a search over functions into one over numbers?

▶ Notebook

The vocabulary of variational inference

term	what it means here
variational distribution, approximate posterior, guide (Pyro)	$q_\psi(\theta)$: the distribution we use in place of the posterior
variational parameters	ψ: q's knobs, such as means and widths. Not the model's parameters θ
variational family	the set of all q we allow: one Gaussian, a mixture, a normalising flow
mean-field	q splits into one independent factor per parameter (Section 1.5.3, Part 5)
ELBO, or minus the variational free energy	the objective we maximise (Part 4)
amortised VI	a network predicts ψ for each data point: the encoder of a VAE (Lecture 2)

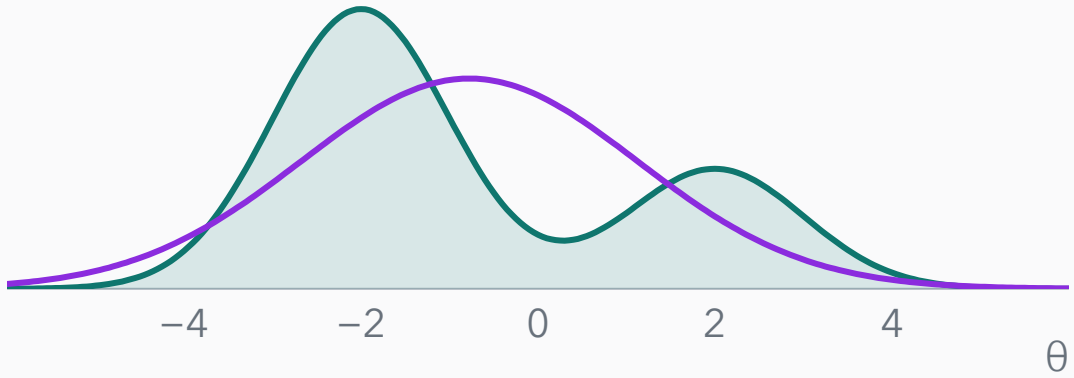
> Which parameters does VI train: θ or ψ?

▶ Notebook 1.5

The family is our choice: one Gaussian, or a mixture of two

family: one Gaussian

teal: the target p purple: the closest q



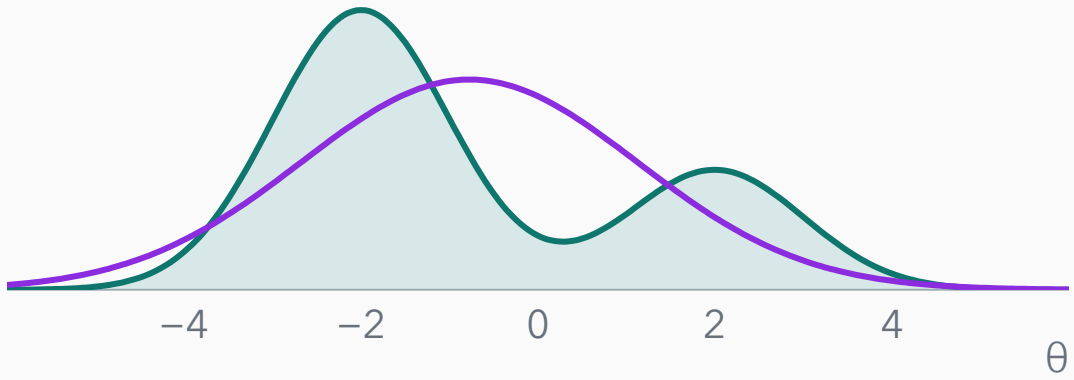
> One Gaussian has one bump, so it compromises: the closest one stretches across both bumps and puts mass in the gap between them.

▶ Notebook 1.5

The family is our choice: one Gaussian, or a mixture of two

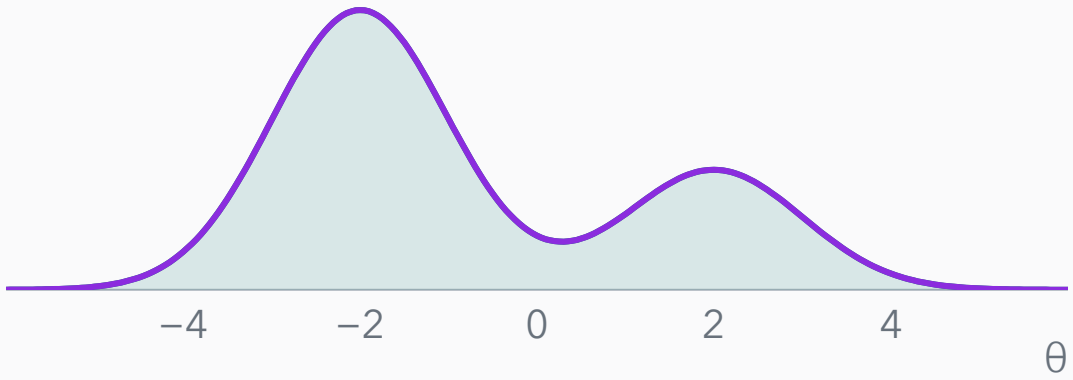
family: one Gaussian

teal: the target p purple: the closest q



family: mixtures of two Gaussians

teal: the target p purple: the closest q



With a mixture of two Gaussians, the closest q matches both bumps. A richer family fits better, at the price of more knobs ψ.

▶ Notebook 1.5

Our q for the network: one Gaussian per parameter

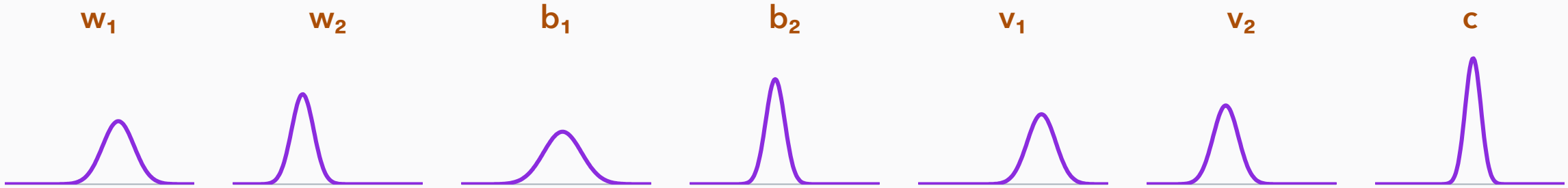
$$q_\psi(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, \mu_j,\, \sigma_j^2), \qquad \psi = (\mu_1, \ldots, \mu_7,\, \sigma_1, \ldots, \sigma_7)$$

> Pick a simple family: one Gaussian per parameter, each with its own mean and width. ψ is fourteen numbers.

▶ Notebook 1.5

Our q for the network: one Gaussian per parameter

$$q_\psi(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, \mu_j,\, \sigma_j^2), \qquad \psi = (\mu_1, \ldots, \mu_7,\, \sigma_1, \ldots, \sigma_7)$$

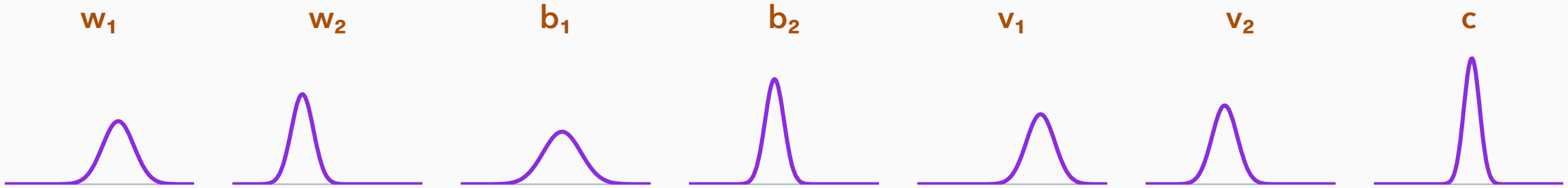


One bell per parameter. Together they define q(θ), and drawing θ from q is easy.

▶ Notebook 1.5

Our q for the network: one Gaussian per parameter

$$q_\psi(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j;\, \mu_j,\, \sigma_j^2), \qquad \psi = (\mu_1, \ldots, \mu_7,\, \sigma_1, \ldots, \sigma_7)$$



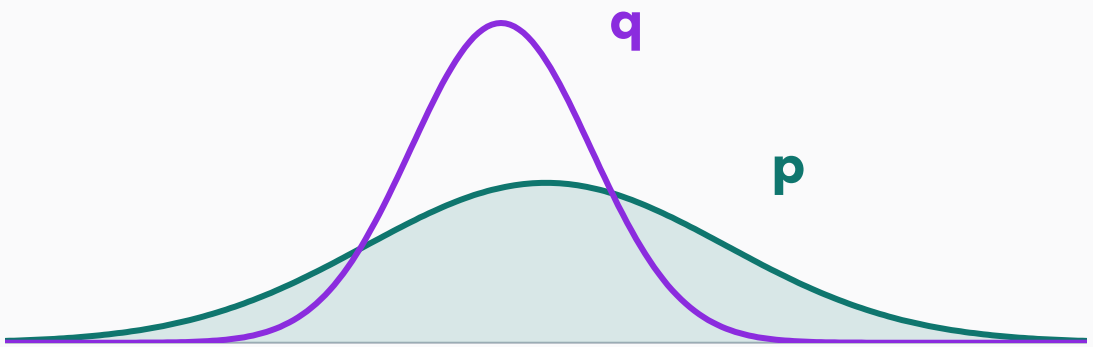
$$\psi^* = \arg\min_{\psi} \text{ distance}\big(q_\psi(\theta),\, p(\theta \mid \mathcal{D})\big)$$

> Tune ψ until q is as close as possible to the posterior. Two questions: what is "close", and how do we tune without the posterior?

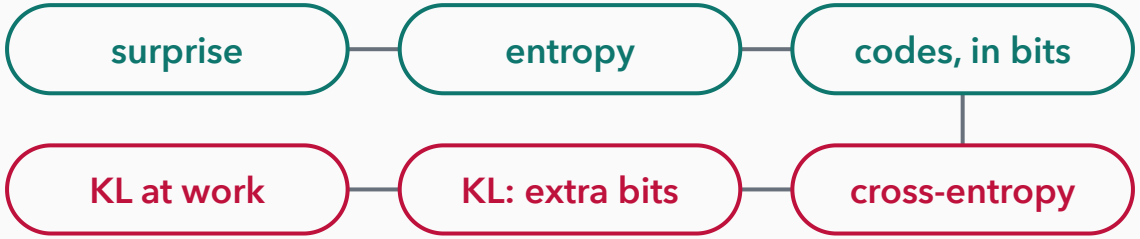
▶ Notebook

Part 2 · How far apart are two distributions?

how far apart are p and q? one number



the road through Part 2

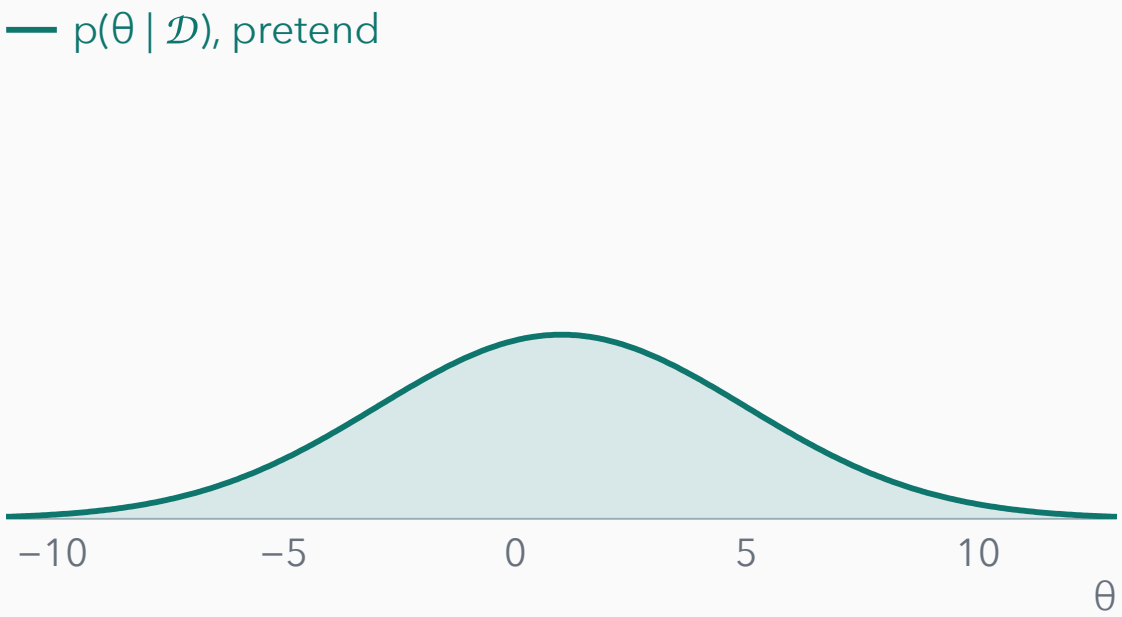


2.1 Surprise ▶
2.2 Entropy: average surprise ▶
2.3 Codes: entropy in bits ▶
2.4 Cross-entropy and KL ▶
2.5 KL at work ▶
2.6 Closed forms and PyTorch ▶
2.7 KL by sampling ▶

> Part 2 builds one number for how far apart two distributions are, from surprise, entropy and codes: the KL divergence.

▶ Notebook 2.1

For now, pretend we know the posterior



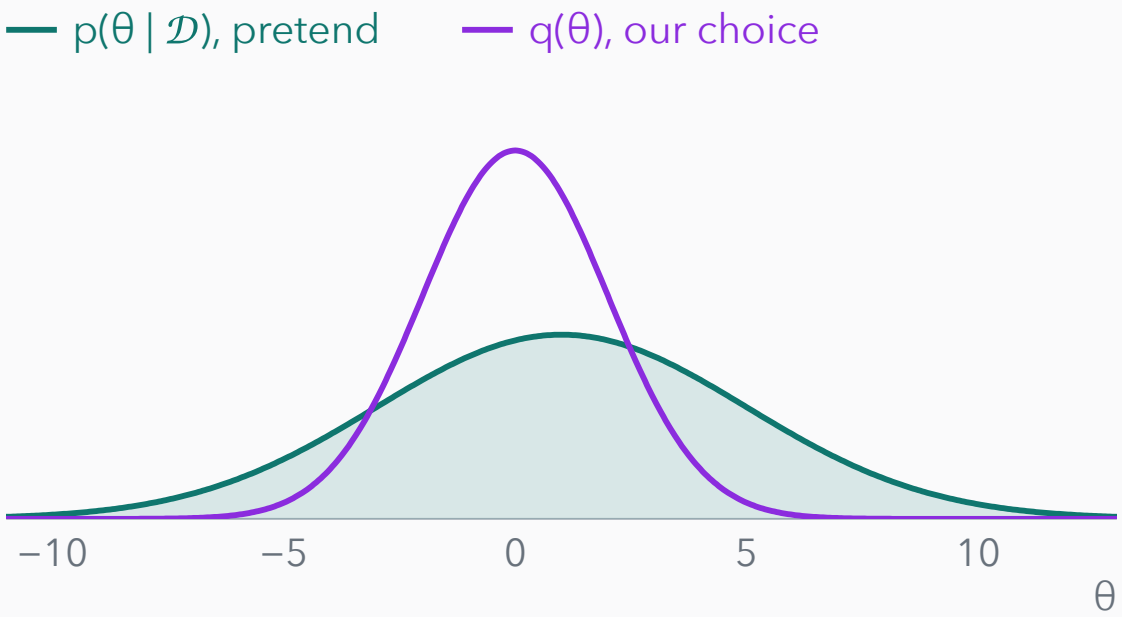
$$p(\theta \mid \mathcal{D}) = \mathcal{N}(1, 4^2)$$

A stand-in for the real posterior: known, one-dimensional, one bump.

To learn the tools, pretend for a while that we know the posterior: one Gaussian, in one dimension.

▶ Notebook 2.1

For now, pretend we know the posterior



$$p(\theta \mid \mathcal{D}) = \mathcal{N}(1, 4^2)$$

A stand-in for the real posterior: known, one-dimensional, one bump.

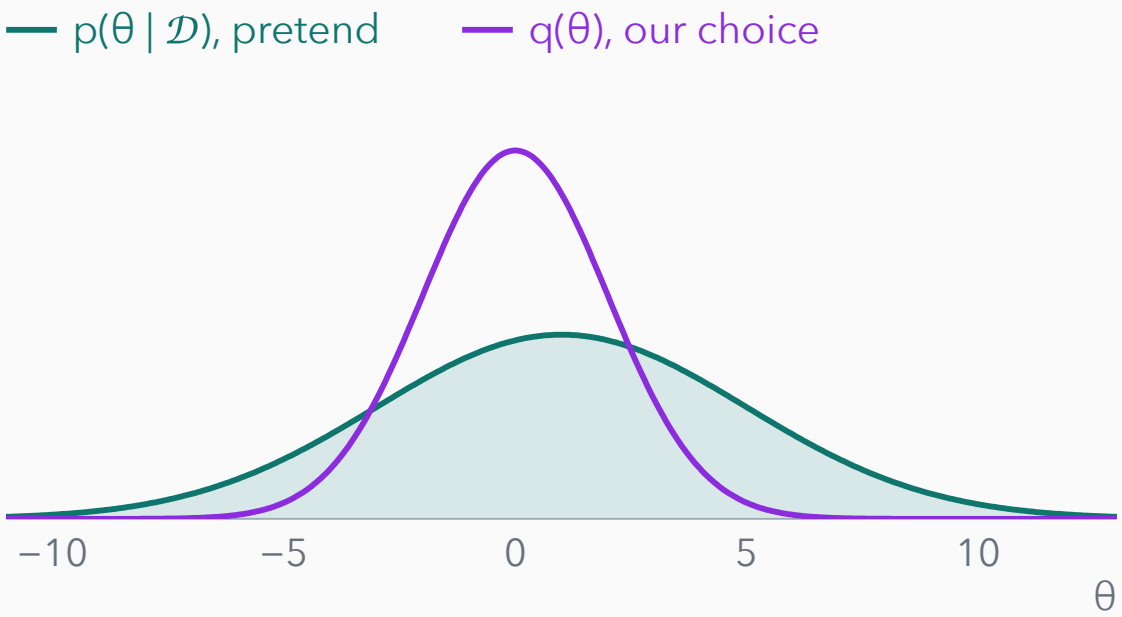
$$q(\theta) = \mathcal{N}(\mu, \sigma^2)$$

Our approximation: we choose μ and σ. Shown: μ = 0, σ = 2.

> Our approximation q is another Gaussian; we choose its mean and its width.

▶ Notebook 2.1

For now, pretend we know the posterior



$$p(\theta \mid \mathcal{D}) = \mathcal{N}(1, 4^2)$$

A stand-in for the real posterior: known, one-dimensional, one bump.

$$q(\theta) = \mathcal{N}(\mu, \sigma^2)$$

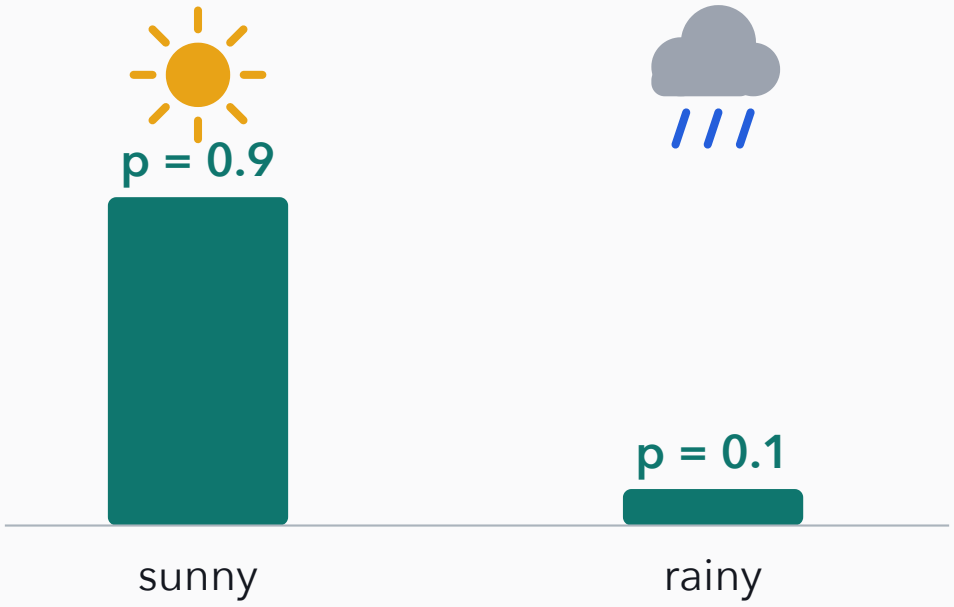
Our approximation: we choose μ and σ. Shown: μ = 0, σ = 2.

How far apart are p and q? We need one number, so that we can make it small.

> How far apart are p and q? We need one number, and we will build it from scratch, starting from surprise.

▶ Notebook 2.1

How surprised would you be? A forecast of 90% sun



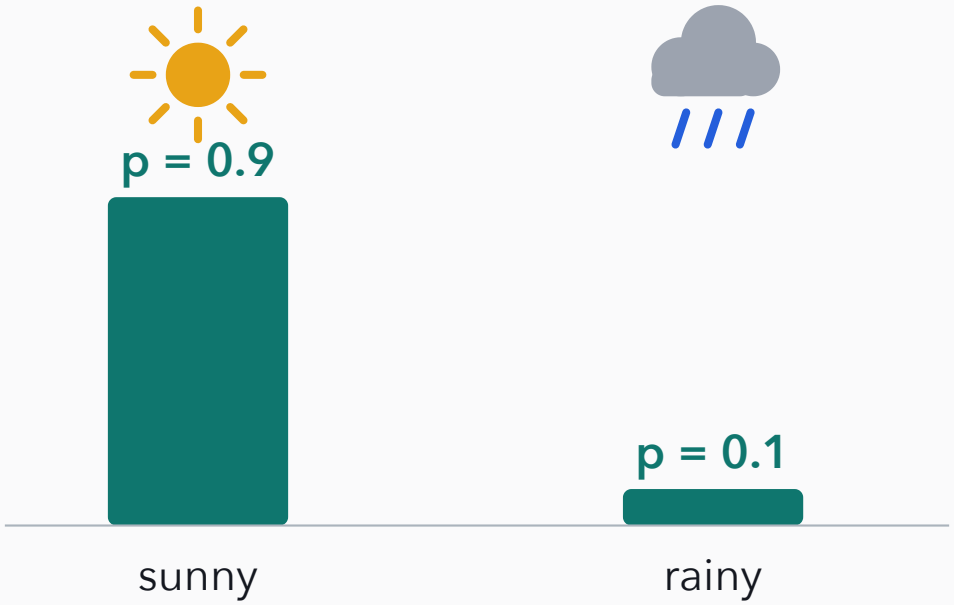
The forecast for tomorrow: sun with probability 0.9, rain with probability 0.1.

> Start with one outcome. The forecast says 90% sun, 10% rain.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

How surprised would you be? A forecast of 90% sun



The forecast for tomorrow: sun with probability 0.9, rain with probability 0.1.

If it rains you are surprised; if the sun shines, hardly at all. **Rare outcomes carry more surprise.**

Surprise is information: the outcome you did not expect tells you the most.

> Rain would surprise you; sun would not. We want a number for surprise: big for rare outcomes, small for common ones.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

What should surprise be? Three requirements

$$I(1) = 0$$

a certain outcome is no surprise

$$p \downarrow \Rightarrow \; I(p) \uparrow$$

rarer means more surprising

$$I(p_1\, p_2) = I(p_1) + I(p_2)$$

two independent days: probabilities multiply, surprises add

candidate	I(1) = 0	rarer, more	days add: I(½ · 0.1) vs I(½) + I(0.1)

> Surprise I(p) should be zero for a certain outcome, larger for rarer ones, and for two independent days, whose probabilities multiply, the surprises should add.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

What should surprise be? Three requirements

$$I(1) = 0$$

a certain outcome is no surprise

$$p \downarrow \Rightarrow \; I(p) \uparrow$$

rarer means more surprising

$$I(p_1\, p_2) = I(p_1) + I(p_2)$$

two independent days: probabilities multiply, surprises add

candidate	I(1) = 0	rarer, more	days add: I(½ · 0.1) vs I(½) + I(0.1)
$1 - p$	✓	✓	✗ 0.95 vs 1.4
$1/p$	✗ 1	✓	✗ 20 vs 12

> Test 1 − p and 1/p on two days with probabilities ½ and 0.1. Neither adds up, and 1/p even gives a certain outcome a surprise of 1.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

What should surprise be? Three requirements

$$I(1) = 0$$

a certain outcome is no surprise

$$p \downarrow \Rightarrow \; I(p) \uparrow$$

rarer means more surprising

$$I(p_1\, p_2) = I(p_1) + I(p_2)$$

two independent days: probabilities multiply, surprises add

candidate	I(1) = 0	rarer, more	days add: I(½ · 0.1) vs I(½) + I(0.1)
$1 - p$	✓	✓	✗ 0.95 vs 1.4
$1/p$	✗ 1	✓	✗ 20 vs 12
$\log_2 \frac{1}{p}$	✓	✓	✓ 4.32 = 1 + 3.32

> The logarithm passes all three: log $1/(p_1p_2)$ = log $1/p_1$ + log $1/p_2$. Only the log turns products into sums. So surprise is I(x) = log 1/p(x).

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

Five days: probabilities multiply, surprises add



p = 0.9	×	p = 0.9	×	p = 0.9	×	p = 0.9	×	p = 0.9	**= 0.590**
0.152 bits	+	0.152 bits	+	0.152 bits	+	0.152 bits	+	0.152 bits	**= 0.76 bits**

probabilities multiply

surprises add: $\log_2 1/0.590 = 0.76$

> Five sunny days: the probability multiplies to $0.9^5 = 0.59$, and the surprise adds: $5 \times 0.152 = 0.76$ bits.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

Five days: probabilities multiply, surprises add



p = 0.9	×	p = 0.9	×	p = 0.9	×	p = 0.9	×	p = 0.9	**= 0.590**	probabilities multiply
0.152 bits	+	0.152 bits	+	0.152 bits	+	0.152 bits	+	0.152 bits	**= 0.76 bits**	surprises add: $\log_2 1/0.590 = 0.76$



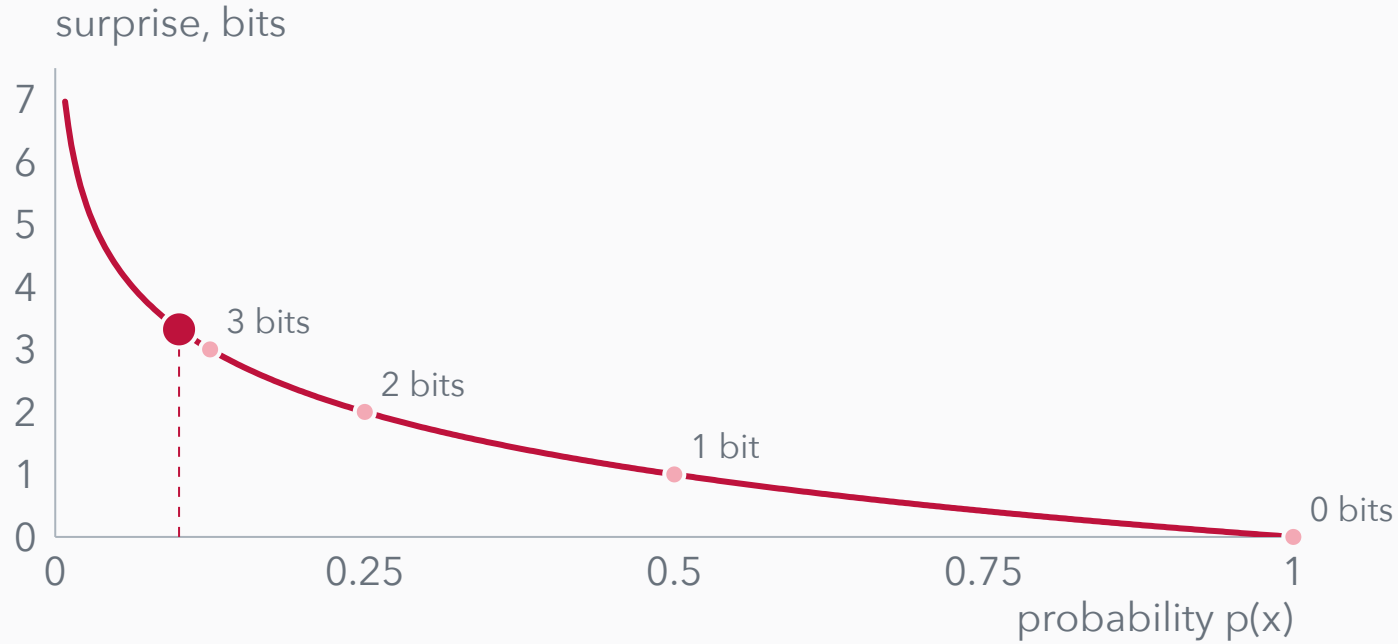
p = 0.9	×	p = 0.9	×	p = 0.1	×	p = 0.9	×	p = 0.9	**= 0.0656**	probabilities multiply
0.152 bits	+	0.152 bits	+	3.32 bits	+	0.152 bits	+	0.152 bits	**= 3.93 bits**	surprises add: $\log_2 1/0.0656 = 3.93$

> Make one day rainy: the probability drops to 0.066 and the surprise jumps to 3.93 bits. The single rainy day carries most of it.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

Surprise in bits: $I(x) = \log_2 1/p(x)$



p(x) = 0.10

surprise = $\log_2 1/p$ = 3.32 bits

= ln 1/p = 2.30 nats

a rare outcome: a real surprise

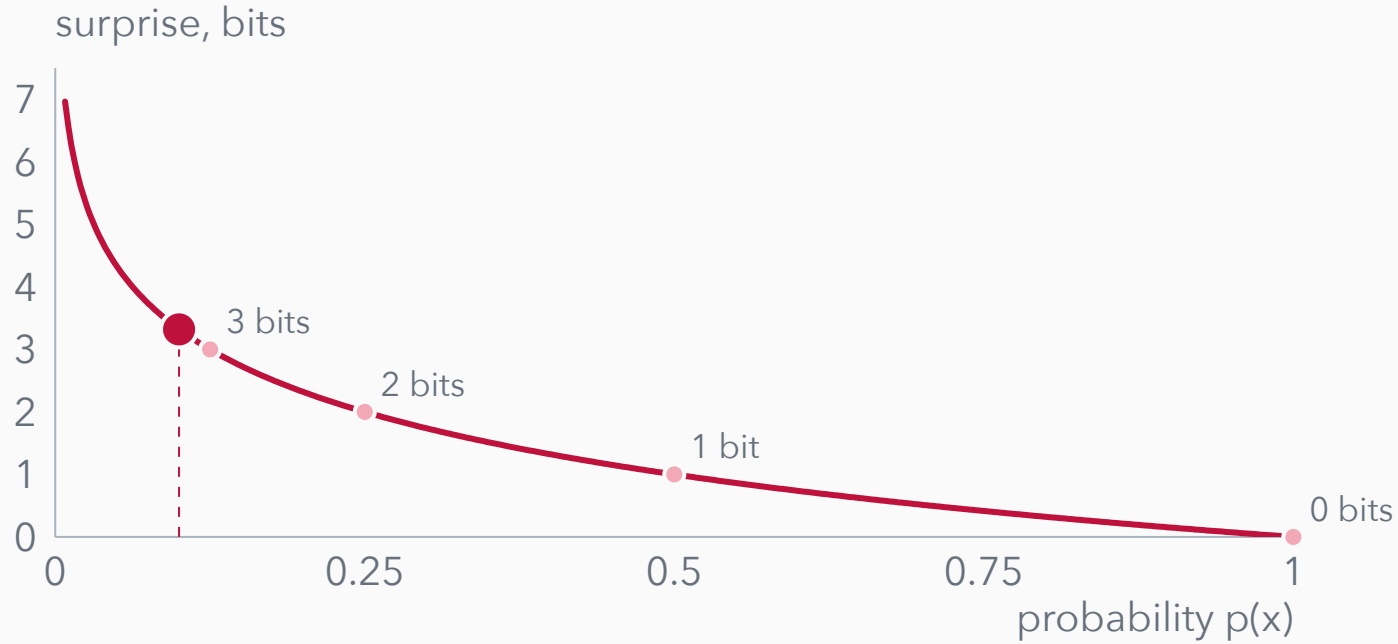
one bit = the surprise of a fair coin flip

A certain outcome gives 0 bits, and halving the probability adds exactly one bit. Drag p, or try the buttons.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.1

Surprise in bits: $I(x) = \log_2 1/p(x)$



p(x) = 0.10

surprise = $\log_2 1/p$ = 3.32 bits

= ln $1/p$ = 2.30 nats

a rare outcome: a real surprise

one bit = the surprise of a fair coin flip

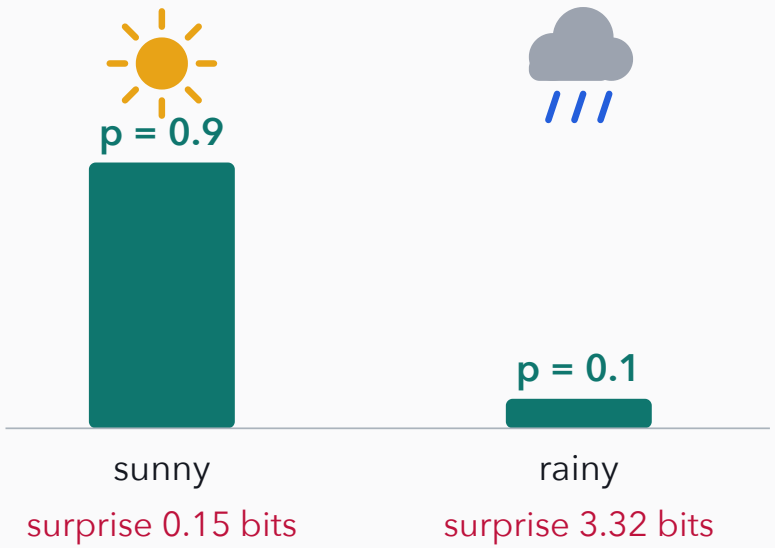
Base 2 gives bits; the natural log gives nats; 1 nat ≈ 1.44 bits.

> One bit is the surprise of a fair coin flip: one yes/no answer you could not guess. Information theory counts in bits, PyTorch in nats.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Entropy: how surprised are you, on average?



$$\text{sunny: } \log_2 \frac{1}{0.9} = 0.15 \text{ bits}$$

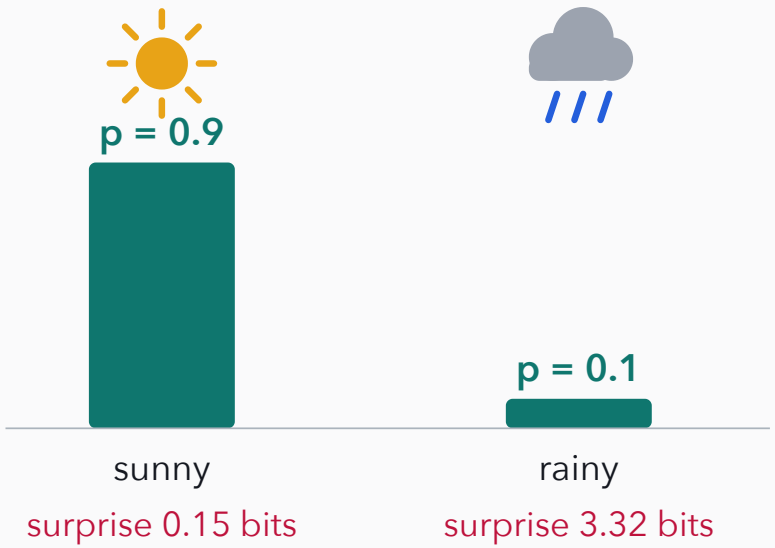
$$\text{rainy: } \log_2 \frac{1}{0.1} = 3.32 \text{ bits}$$

> Each kind of weather has its surprise: 0.15 bits for sun, 3.32 for rain.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Entropy: how surprised are you, on average?



$$\text{sunny: } \log_2 \frac{1}{0.9} = 0.15 \text{ bits}$$

$$\text{rainy: } \log_2 \frac{1}{0.1} = 3.32 \text{ bits}$$

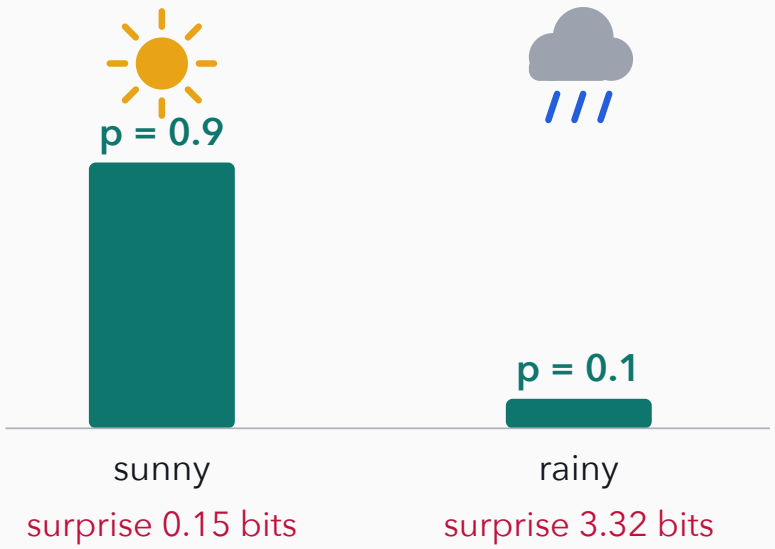
$$H = 0.9\,(0.15) + 0.1\,(3.32) = 0.47 \text{ bits}$$

> How surprised are you on a typical day? 90% of days give 0.15 bits and 10% give 3.32: on average 0.47 bits.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Entropy: how surprised are you, on average?



$$\text{sunny: } \log_2 \tfrac{1}{0.9} = 0.15 \text{ bits}$$

$$\text{rainy: } \log_2 \tfrac{1}{0.1} = 3.32 \text{ bits}$$

$$H = 0.9\,(0.15) + 0.1\,(3.32) = 0.47 \text{ bits}$$

$$H(p) = \sum_x p(x) \log_2 \frac{1}{p(x)} = \mathbb{E}_{x\sim p}\left[\log_2 \frac{1}{p(x)}\right]$$

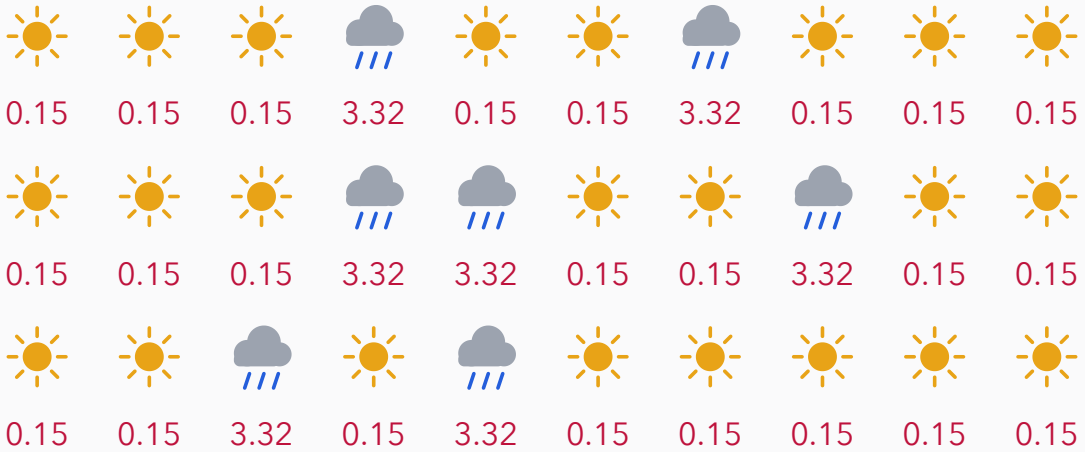
> That weighted average of surprise is the entropy: the surprise you expect from one draw.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

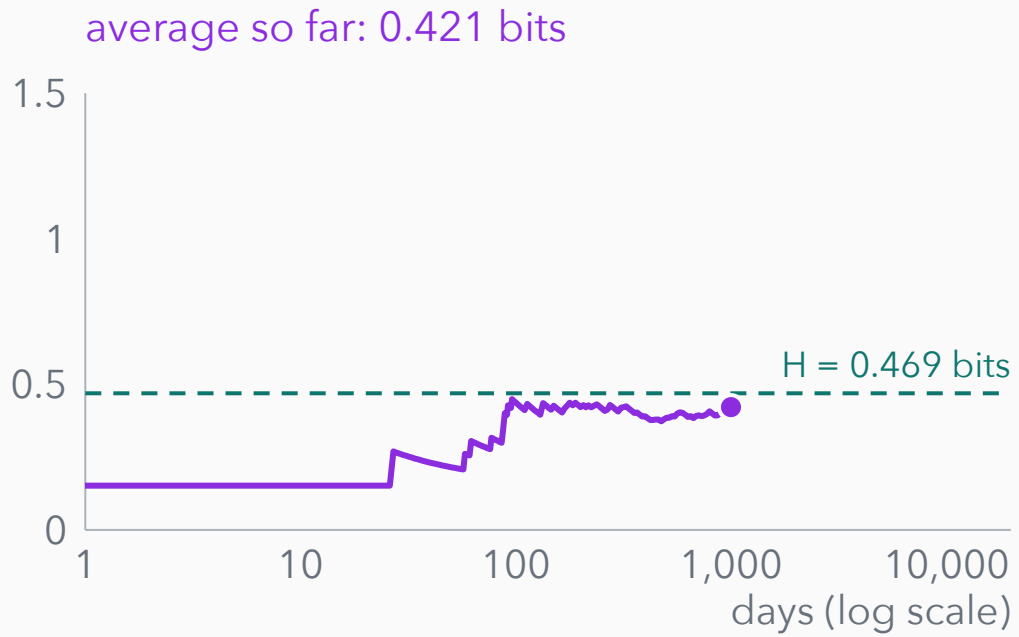
▶ Notebook 2.2

Why an expectation? It is the average over many days

days drawn from the forecast (latest 30), with their surprise



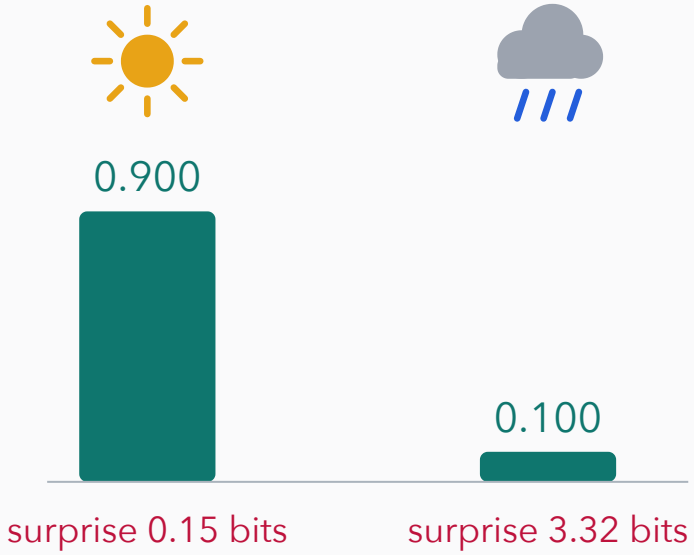
1,000 days



> Let days go by, each with its surprise: 0.15 bits if sunny, 3.32 if rainy. The running average settles on 0.47 bits. An expectation is a long-run average.

▶ Notebook 2.2

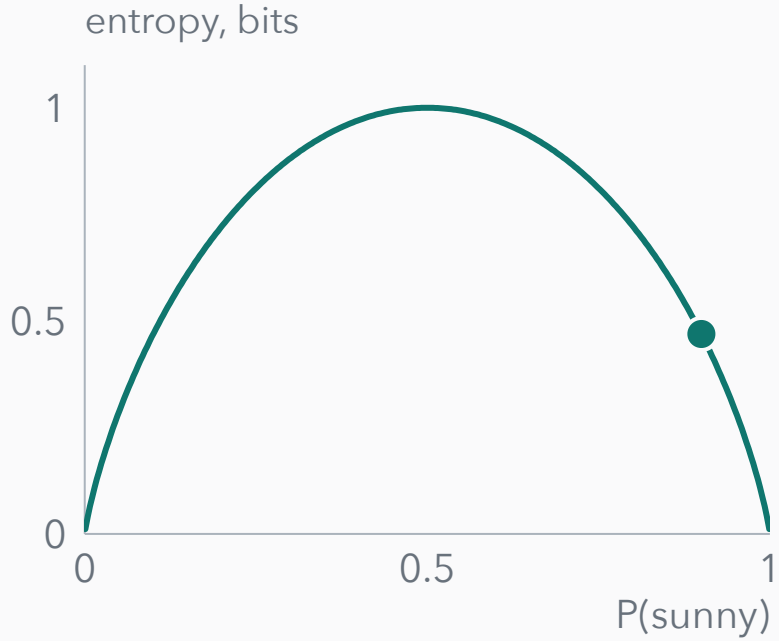
From certain to fully uncertain: between 0 and 1 bit



H = average surprise

$= 0.900 \times 0.15 + 0.100 \times 3.32$

$= 0.469$ bits

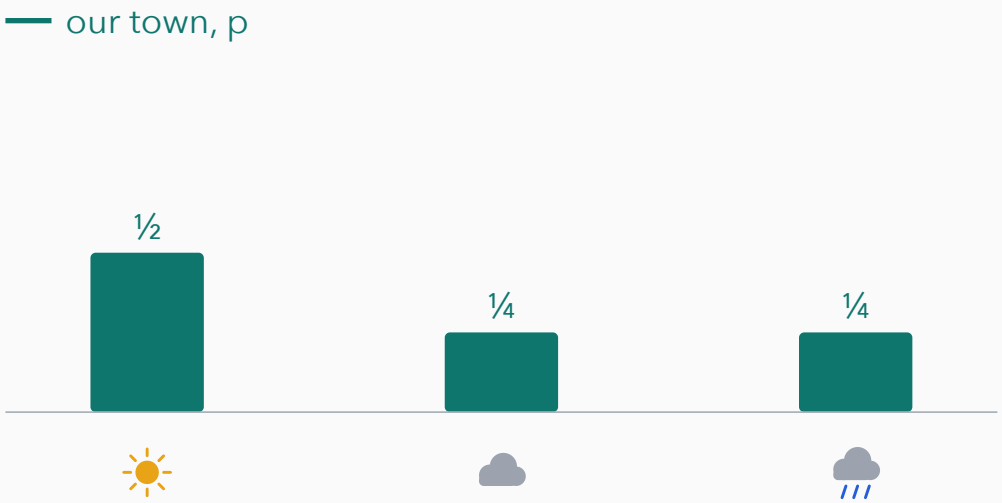


> Drag the chance of sun. A 50/50 forecast carries 1 bit, the most two outcomes can; 87.5/12.5 carries 0.54; a certain forecast, 0.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Three kinds of weather: 1.5 bits



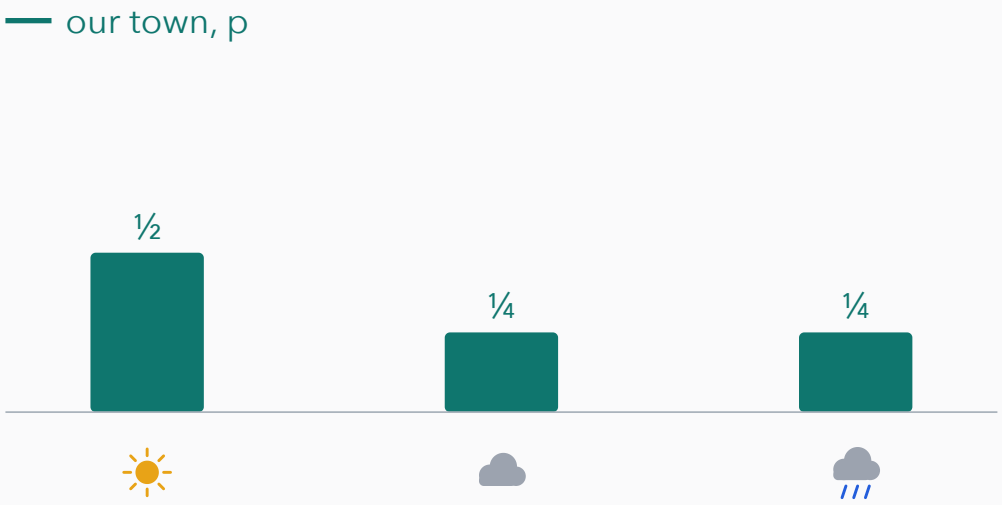
weather	p	$\log_2 \frac{1}{p}$
sunny	½	1 bit
cloudy	¼	2 bits
rainy	¼	2 bits

> Our town: sunny half the time, cloudy a quarter, rainy a quarter. Their surprises are 1, 2 and 2 bits.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Three kinds of weather: 1.5 bits



weather	p	$\log_2 \frac{1}{p}$
sunny	½	1 bit
cloudy	¼	2 bits
rainy	¼	2 bits

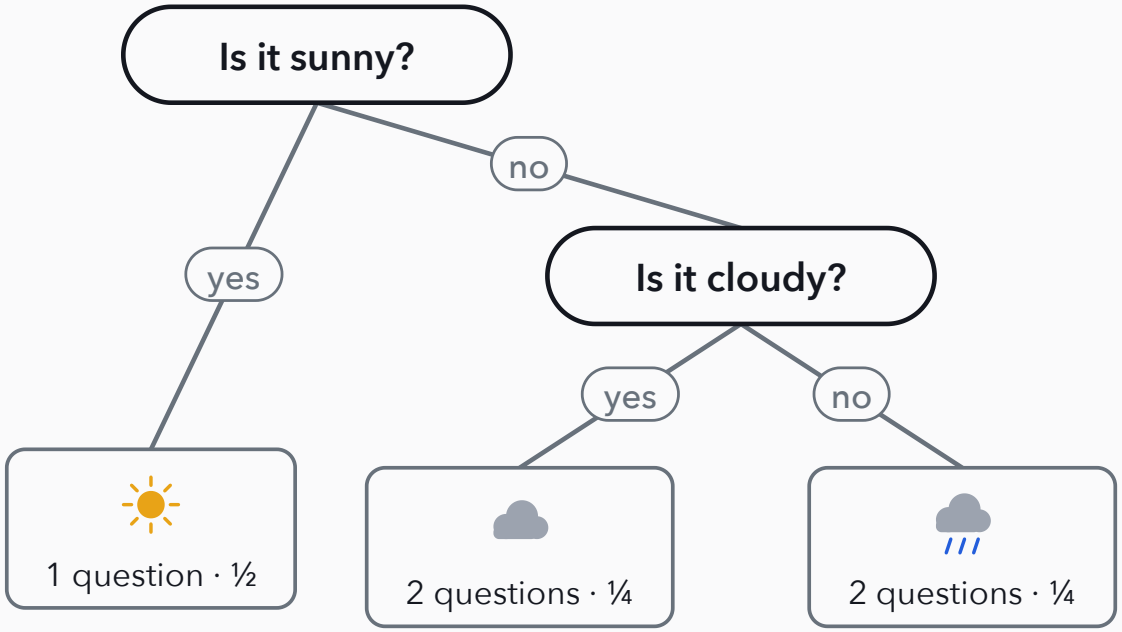
$$H = \frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{4}(2) = 1.5 \text{ bits}$$

> Average surprise: ½(1) + ¼(2) + ¼(2) = 1.5 bits per day.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Entropy counts yes/no questions



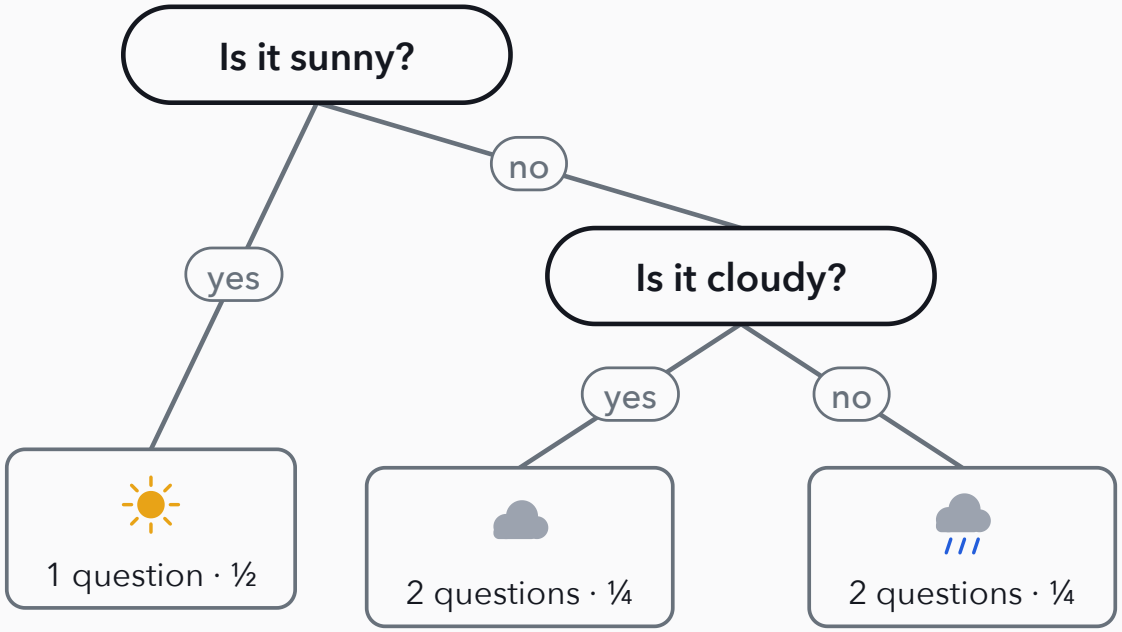
Guess today's weather with yes/no questions, asking about the likeliest first.

> Find today's weather with yes/no questions. Ask "Is it sunny?" first: half the time you are done.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Entropy counts yes/no questions



Guess today's weather with yes/no questions, asking about the likeliest first.

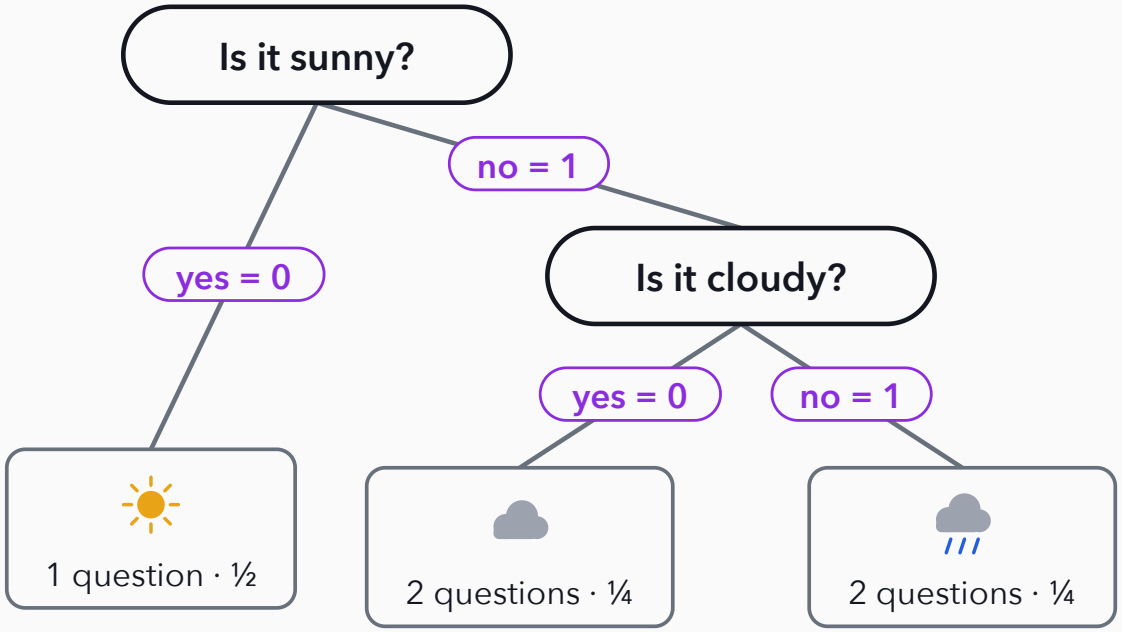
$$\tfrac{1}{2}(1) + \tfrac{1}{4}(2) + \tfrac{1}{4}(2) = 1.5 \text{ questions}$$

> Otherwise ask "Is it cloudy?". On average 1.5 questions: exactly the entropy.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.2

Entropy counts yes/no questions



Guess today's weather with yes/no questions, asking about the likeliest first.

$$\tfrac{1}{2}(1) + \tfrac{1}{4}(2) + \tfrac{1}{4}(2) = 1.5 \text{ questions}$$

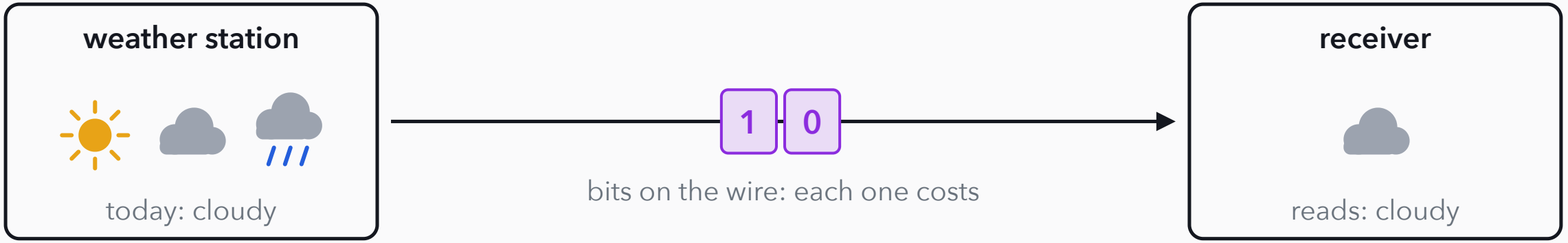
Write each answer as a bit, yes = 0 and no = 1: sunny is **0**, cloudy **10**, rainy **11**. That is a code.

> Each answer is one bit. Writing the answers down gives a code for the weather: Section 2.3.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.3

Way 2: send the weather, in bits

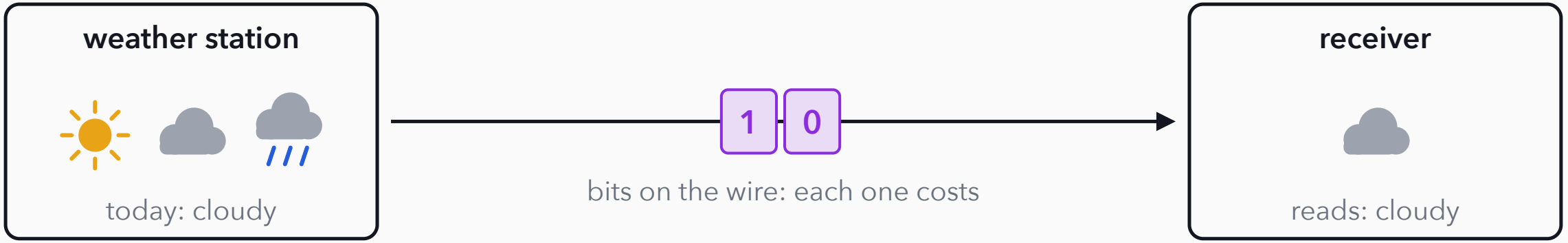


> Turn the question game into messages: a weather station sends each day's weather to a receiver as 0s and 1s. Every bit costs, so we want as few as possible.

Coding story after Aurélien Géron, "A Short Introduction to Entropy, Cross-Entropy and KL-Divergence" (YouTube)

▶ Notebook 2.3

Way 2: send the weather, in bits



	sunny	cloudy	rainy
how often, p	½	¼	¼

> Sunny on half the days, cloudy on a quarter, rainy on a quarter: p = (½, ¼, ¼).

Coding story after Aurélien Géron, "A Short Introduction to Entropy, Cross-Entropy and KL-Divergence" (YouTube)

▶ Notebook 2.3

A code: a string of bits for each kind of day

weather	p	uniform code	bits
sunny	½	`00`	2
cloudy	¼	`01`	2
rainy	¼	`10`	2

The simplest code gives every kind of day the same length: 2 bits.

A code assigns each kind of day a string of bits. The uniform code spends 2 bits on every day.

Coding story after Aurélien Géron, "A Short Introduction to Entropy, Cross-Entropy and KL-Divergence" (YouTube)

▶ Notebook 2.3

A code: a string of bits for each kind of day

weather	p	uniform code	bits
sunny	½	`00`	2
cloudy	¼	`01`	2
rainy	¼	`10`	2

The simplest code gives every kind of day the same length: 2 bits.

Fair, but wasteful: half the days are sunny, yet a sunny day costs as much as a rare one.

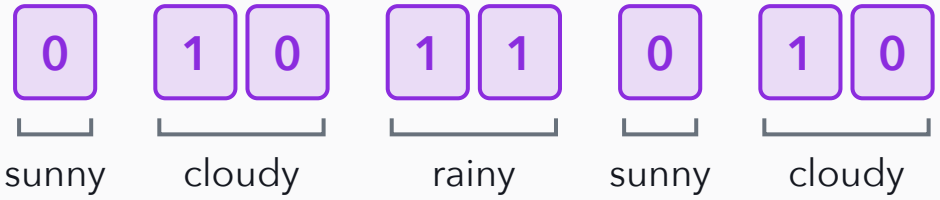
> Wasteful: common days cost as much as rare ones. Could common days get shorter strings?

Coding story after Aurélien Géron, "A Short Introduction to Entropy, Cross-Entropy and KL-Divergence" (YouTube)

▶ Notebook 2.3

Prefix codes: the bits split up without separators

prefix code: sunny 0 · cloudy 10 · rainy 11



→ the receiver reads left to right; each day is unambiguous

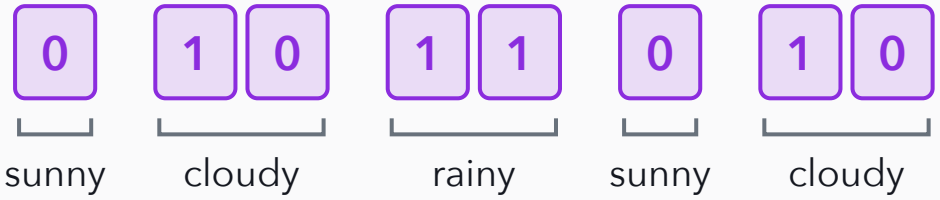
> Give sunny 0, cloudy 10, rainy 11. No codeword is the start of another, so the receiver reads left to right and each day pops out.

Coding story after Aurélien Géron, "A Short Introduction to Entropy, Cross-Entropy and KL-Divergence" (YouTube)

▶ Notebook 2.3

Prefix codes: the bits split up without separators

prefix code: sunny 0 · cloudy 10 · rainy 11



→ the receiver reads left to right; each day is unambiguous

not a prefix code: sunny 0 · cloudy 01 · rainy 1



= sunny, then rainy?

or cloudy?

0 is the start of 01, so the receiver cannot tell.

> Break that rule, say 0, 01 and 1, and "01" could mean cloudy, or sunny then rainy. A prefix code avoids the confusion.

Coding story after Aurélien Géron, "A Short Introduction to Entropy, Cross-Entropy and KL-Divergence" (YouTube)

▶ Notebook 2.3

Huffman's recipe: merge the two rarest, repeat

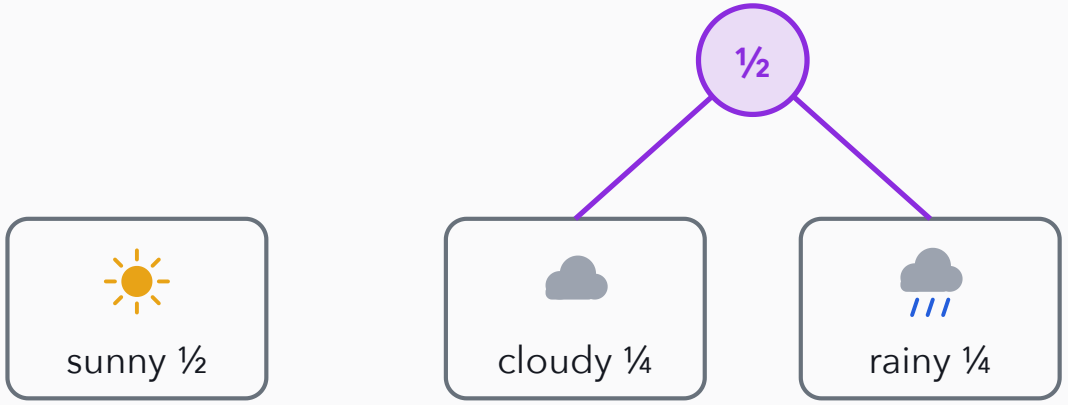
Huffman (1952) builds the best prefix code from the probabilities:



Start with the three outcomes and their probabilities.

▶ Notebook 2.3

Huffman's recipe: merge the two rarest, repeat



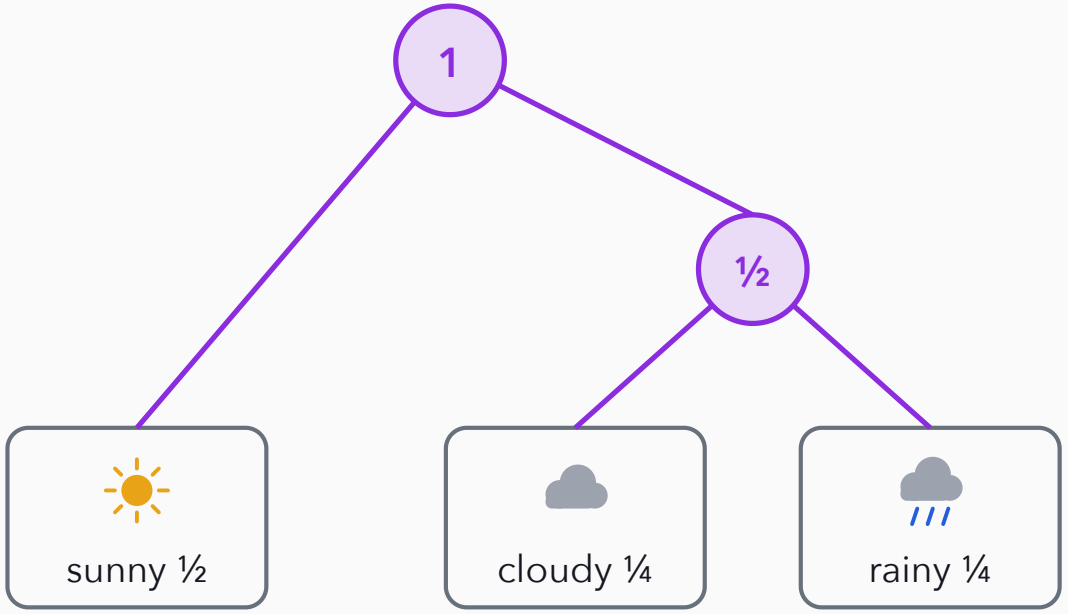
Huffman (1952) builds the best prefix code from the probabilities:

1. Merge the two least likely outcomes into one node.

Merge the two rarest, cloudy and rainy, into one node: ¼ + ¼ = ½.

▶ Notebook 2.3

Huffman's recipe: merge the two rarest, repeat



Huffman (1952) builds the best prefix code from the probabilities:

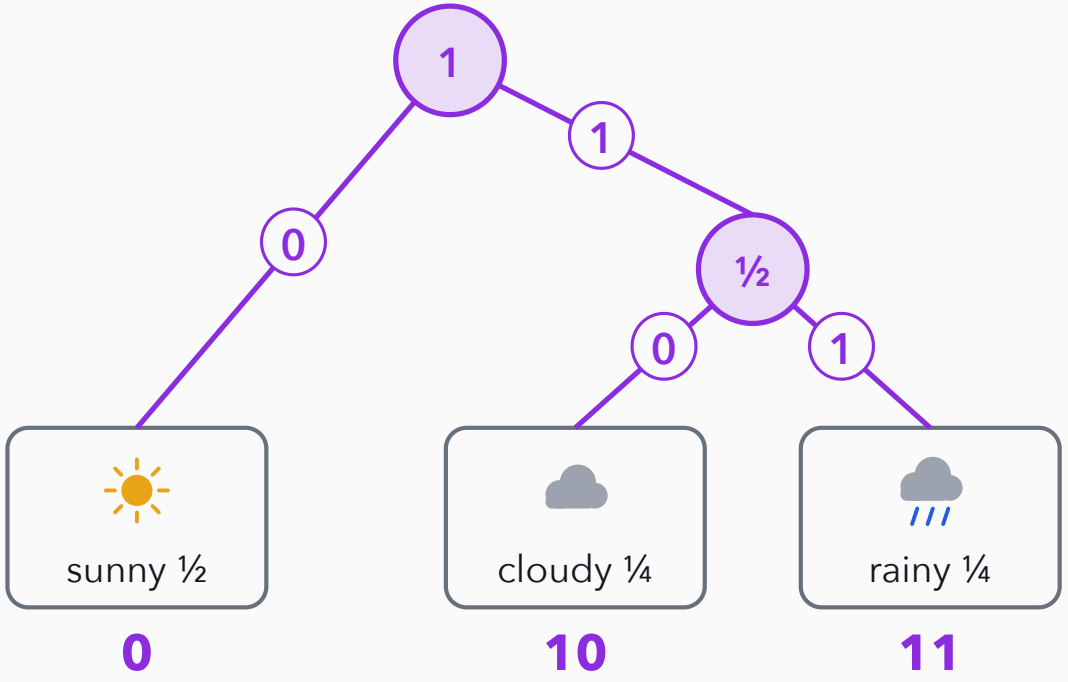
1. Merge the two least likely outcomes into one node.

2. Repeat until one node is left.

Merge again: sunny and the new node make the root.

▶ Notebook 2.3

Huffman's recipe: merge the two rarest, repeat



Huffman (1952) builds the best prefix code from the probabilities:

1. Merge the two least likely outcomes into one node.

2. Repeat until one node is left.

3. Read each codeword off its path: left 0, right 1.

Read the codes off the tree: sunny 0, cloudy 10, rainy 11. The question game's code.

▶ Notebook 2.3

Each codeword is exactly as long as its surprise

weather	p	code	length	surprise
sunny	½	`0`	1	1 bit
cloudy	¼	`10`	2	2 bits
rainy	¼	`11`	2	2 bits

> Compare the code lengths with the surprises: 1, 2, 2 and 1, 2, 2. The best code spends exactly the surprise on each outcome.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.3

Each codeword is exactly as long as its surprise

weather	p	code	length	surprise
sunny	½	`0`	1	1 bit
cloudy	¼	`10`	2	2 bits
rainy	¼	`11`	2	2 bits

$$\text{average length} = \tfrac{1}{2}(1) + \tfrac{1}{4}(2) + \tfrac{1}{4}(2) = 1.5 \text{ bits} = H(p)$$

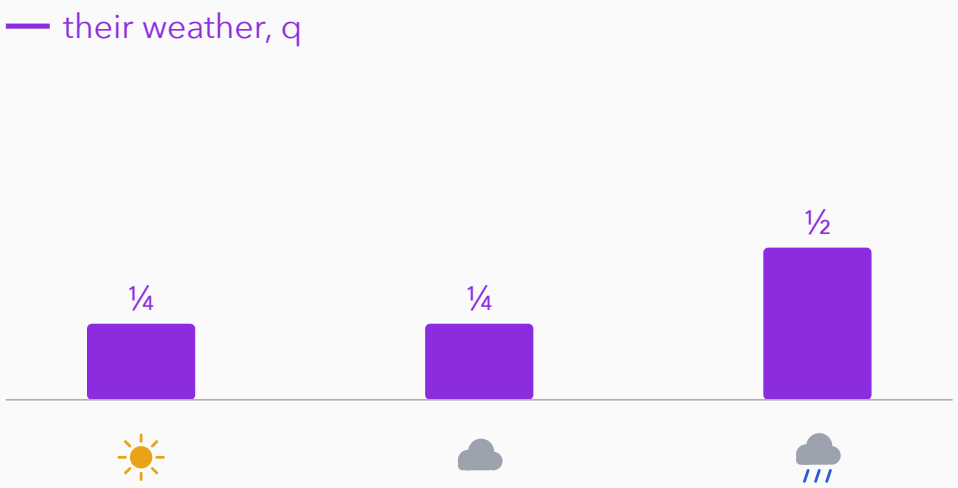
Shannon (1948): no code averages fewer than H(p) bits. Huffman reaches it when every probability is a power of ½, and comes within 1 bit otherwise.

> So the average message length is the average surprise, the entropy: 1.5 bits per day instead of 2.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Another town: weather q, and the best code for it



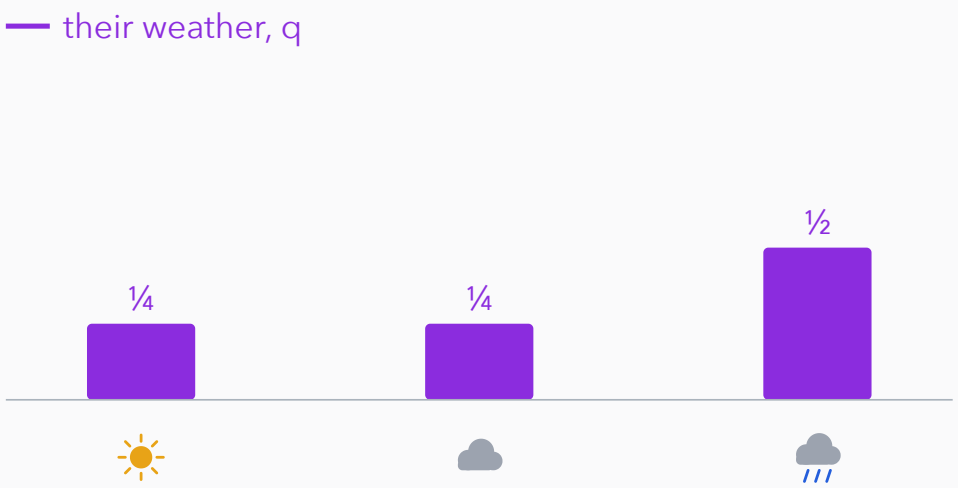
weather	q	Huffman code	length: $\log_2 1/q$
sunny	¼	10	2
cloudy	¼	11	2
rainy	½	0	1

> A town where rain is common has weather q = (¼, ¼, ½). Huffman on q gives rainy the short codeword: sunny 10, cloudy 11, rainy 0.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Another town: weather q, and the best code for it



weather	q	Huffman code	length: $\log_2 1/q$
sunny	¼	`10`	2
cloudy	¼	`11`	2
rainy	½	`0`	1

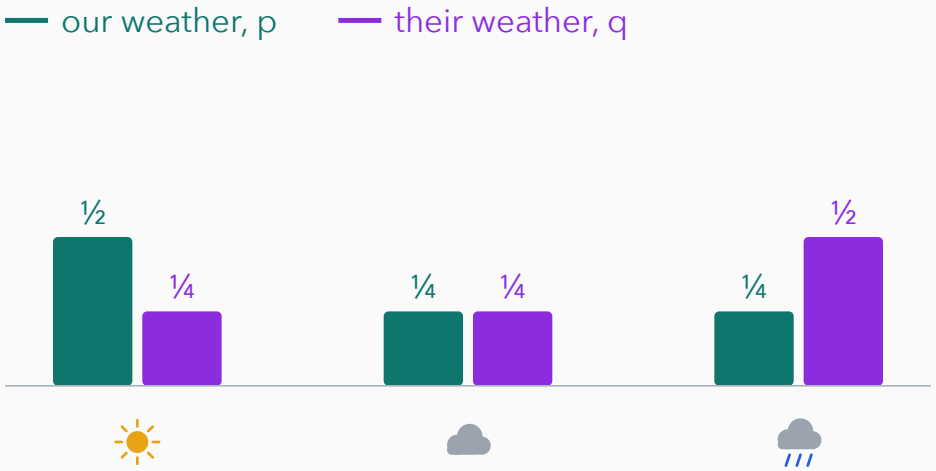
$$H(q) = \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{2} \cdot 1 = 1.5 \text{ bits}$$

> In their own town this code is perfect: it averages H(q) = 1.5 bits a day, the entropy of q. Each length is $\log_2 1/q$, the surprise under q.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Their code, our weather: a code that expects q



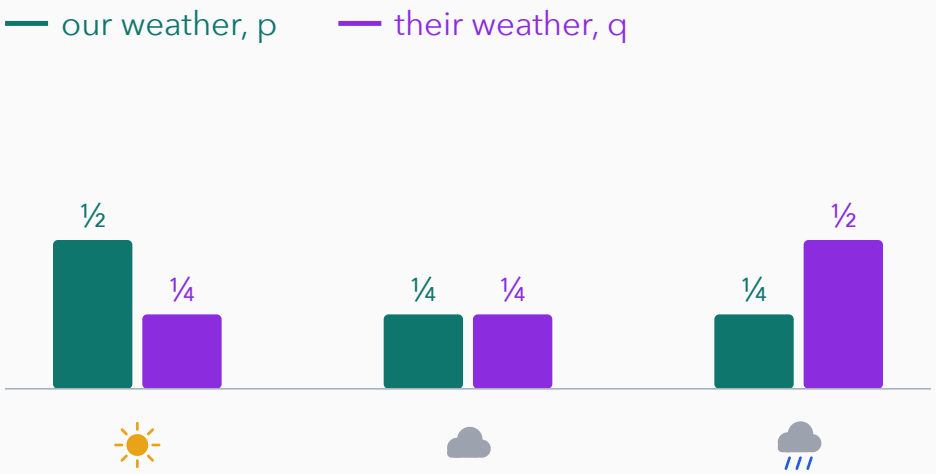
weather	how often here: p	their code	its length: $\log_2 1/q$
sunny	½	10	2
cloudy	¼	11	2
rainy	¼	0	1

> Now their station sends our weather. The days come from p, ours. The codeword lengths come from q, theirs.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Their code, our weather: a code that expects q



weather	how often here: p	their code	its length: $\log_2 1/q$
sunny	½	10	2
cloudy	¼	11	2
rainy	¼	0	1

The code **expects q**: its lengths are right only if days arrive with q's frequencies. Ours arrive with p's, so our common sunny days get a long codeword.

> "The code expects q" means it was built for days that follow q. Here sunny, half of our days, costs 2 bits instead of 1.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Cross-entropy: the average cost when the code expects q

weather	how often: p(x)	bits paid: $\log_2 1/q(x)$	contribution
sunny	½	2	
cloudy	¼	2	
rainy	¼	1	
average per day			

> How many bits a day does their code cost on our weather? For each kind of day: how often it happens here, p, and what it costs with their code, $\log_2 1/q$.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Cross-entropy: the average cost when the code expects q

weather	how often: p(x)	bits paid: $\log_2 1/q(x)$	contribution
sunny	½	2	½ × 2 = 1
cloudy	¼	2	¼ × 2 = ½
rainy	¼	1	¼ × 1 = ¼
average per day			**1.75 bits**

> Multiply and add: ½ × 2 + ¼ × 2 + ¼ × 1 = 1.75 bits per day.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Cross-entropy: the average cost when the code expects q

weather	how often: p(x)	bits paid: $\log_2 1/q(x)$	contribution
sunny	½	2	½ × 2 = 1
cloudy	¼	2	¼ × 2 = ½
rainy	¼	1	¼ × 1 = ¼
average per day			**1.75 bits**

$$H(p, q) = \sum_x p(x) \log_2 \frac{1}{q(x)} = \tfrac{1}{2} \cdot 2 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{4} \cdot 1 = 1.75 \text{ bits}$$

> That is the cross-entropy formula: teal weights from the truth p, purple lengths from the belief q.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Cross-entropy: the average cost when the code expects q

weather	how often: p(x)	bits paid: $\log_2 1/q(x)$	contribution
sunny	½	2	½ × 2 = 1
cloudy	¼	2	¼ × 2 = ½
rainy	¼	1	¼ × 1 = ¼
average per day			**1.75 bits**

$$H(p, q) = \sum_x p(x) \log_2 \frac{1}{q(x)} = \tfrac{1}{2} \cdot 2 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{4} \cdot 1 = 1.75 \text{ bits}$$

$$H(p, p) = H(p) = \tfrac{1}{2} \cdot 1 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{4} \cdot 2 = 1.5 \text{ bits}$$

> With our own code, q = p, the same formula gives the entropy, 1.5 bits. Their code costs 0.25 bits more per day.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Watch it happen: our weather through both codes

the latest days, and the bits each code sends



code for p	11	11	11	0	0	0	11	10
code for q	0	0	0	10	10	10	0	11

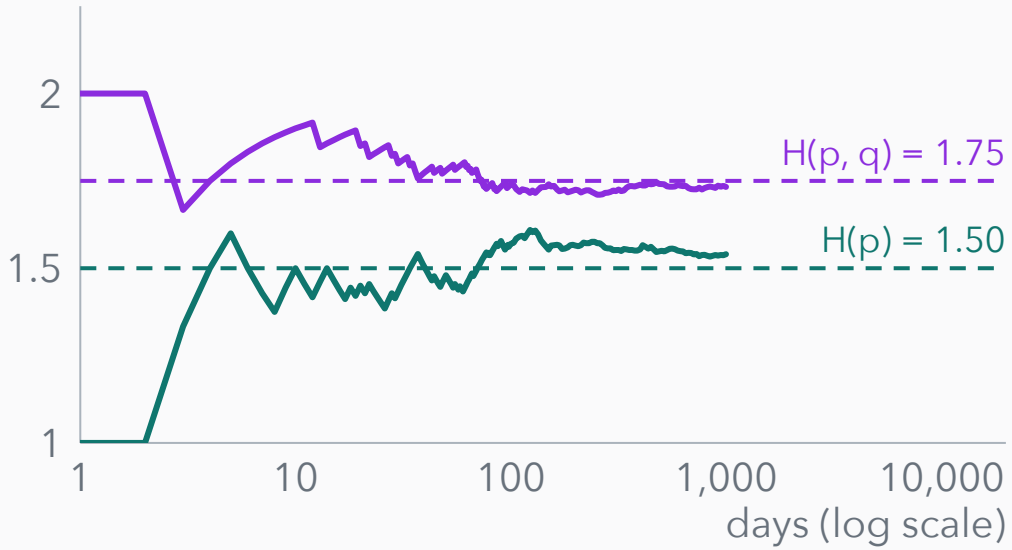
1,000 days sent

code for p: 1,540 bits, 1.540 per day

code for q: 1,733 bits, 1.733 per day

wasted: 0.193 bits per day

bits per day so far



> Send days of our weather through both codes. With 10, 100, 1,000 days the right code settles at 1.5 bits per day and the wrong one at 1.75: 0.25 bits per day wasted.

▶ Notebook 2.4

KL divergence: the extra bits we waste

weather	how often: p	their code: $\log_2 1/q$	our code: $\log_2 1/p$	extra bits
sunny	½	2	1	+1
cloudy	¼	2	2	0
rainy	¼	1	2	−1
average		**1.75**	**1.5**	**+0.25**

> Day by day: bits with their code, bits with ours, and the difference, the extra bits. A sunny day costs 1 extra, a cloudy day the same, a rainy day saves 1.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

KL divergence: the extra bits we waste

weather	how often: p	their code: $\log_2 1/q$	our code: $\log_2 1/p$	extra bits
sunny	½	2	1	+1
cloudy	¼	2	2	0
rainy	¼	1	2	−1
average		**1.75**	**1.5**	**+0.25**

$$\mathrm{KL}(p \,\|\, q) = H(p, q) - H(p) = \sum_x p(x)\left(\log_2 \frac{1}{q(x)} - \log_2 \frac{1}{p(x)}\right)$$

$$= 1.75 - 1.5 = \tfrac{1}{2}(+1) + \tfrac{1}{4}(0) + \tfrac{1}{4}(-1)$$

KL averages the extra bits over our weather, with the same weights p: ½(+1) + ¼(0) + ¼(−1) = 0.25 bits per day. Cross-entropy minus entropy.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

KL divergence: the extra bits we waste

weather	how often: p	their code: $\log_2 1/q$	our code: $\log_2 1/p$	extra bits
sunny	½	2	1	+1
cloudy	¼	2	2	0
rainy	¼	1	2	−1
average		**1.75**	**1.5**	**+0.25**

$$\begin{aligned} \mathrm{KL}(p \,\|\, q) &= H(p,q) - H(p) && = 1.75 - 1.5 \\ &= \sum_x p(x)\left(\log_2 \frac{1}{q(x)} - \log_2 \frac{1}{p(x)}\right) && = \tfrac{1}{2}(+1) + \tfrac{1}{4}(0) + \tfrac{1}{4}(-1) \\ &= \sum_x p(x)\, \log_2 \frac{p(x)}{q(x)} && = 0.25 \text{ bits per day} \end{aligned}$$

A difference of logs is the log of a ratio: KL(p ‖ q) = Σ p log p/q. Inside, p/q compares the two beliefs on each day; outside, p says how often that day comes.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Never negative, and zero only for the right code

$$H(p, q) \geq H(p) \quad \Longrightarrow \quad \mathrm{KL}(p \,\|\, q) \geq 0$$

No code averages fewer than H(p) bits, so a code built for q can only waste bits.

> The right code is the best possible, so any other code can only cost more: KL is never negative.

▶ Notebook 2.4

Never negative, and zero only for the right code

$$H(p, q) \geq H(p) \quad \Longrightarrow \quad \mathrm{KL}(p \,\|\, q) \geq 0$$

No code averages fewer than H(p) bits, so a code built for q can only waste bits.

$$\mathrm{KL}(p \,\|\, q) = 0 \quad \Longleftrightarrow \quad q = p$$

Single terms can be negative, as rainy's was; the weighted total cannot. An optional slide proves KL ≥ 0 for any distributions.

> And it is zero only when the belief matches the truth.

▶ Notebook 2.4

Reading KL(p ‖ q): data from the left, code from the right

$$\mathrm{KL}(p \,\|\, q) = \sum_x \underbrace{p(x)}_{\text{how often: the data}} \log_2 \frac{p(x)}{\underbrace{q(x)}_{\text{the code, or model}}}$$

	left: p	right: q
its role	the reference: where the data come from	the model: the code we use
in the sum	supplies the weights	supplies the lengths
our weather	our weather	their code

> The two arguments play different roles. The left one, p, is where the data come from: it supplies the weights. The right one, q, is the code or model we use: it supplies the lengths.

▶ Notebook 2.4

Reading KL(p ‖ q): data from the left, code from the right

$$\mathrm{KL}(p \,\|\, q) = \sum_x \underbrace{p(x)}_{\text{how often: the data}} \log_2 \frac{p(x)}{\underbrace{q(x)}_{\text{the code, or model}}}$$

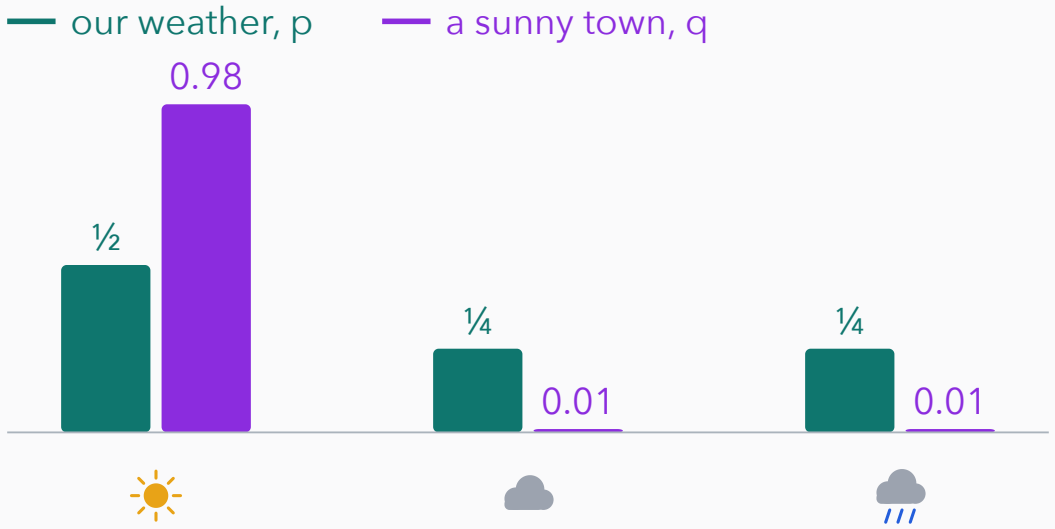
	left: p	right: q
its role	the reference: where the data come from	the model: the code we use
in the sum	supplies the weights	supplies the lengths
our weather	our weather	their code

Read KL(p ‖ q) as **data from p, coded with q**. In Section 2.5 the left slot holds the truth (a label, a teacher) and the right slot the model.

> So read KL(p ‖ q) as: data from p, coded with q. The reference on the left, the model on the right. Swap them and you ask a different question.

▶ Notebook 2.4

Which one is the truth? KL is not symmetric



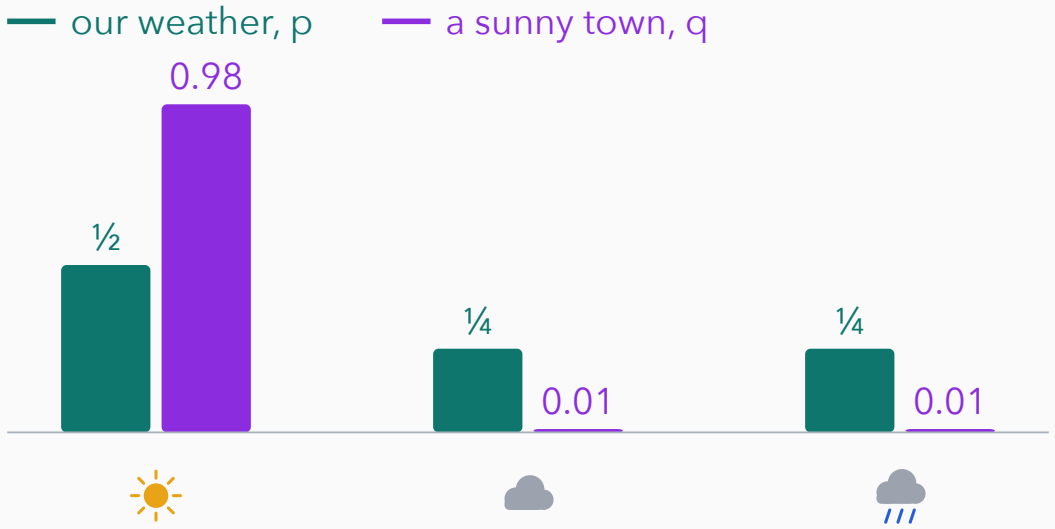
	bits wasted per day
our weather, their code: $\mathrm{KL}(p\|\|q)$	1.84

> A sunny town expects sun 98% of the time, so its code gives cloudy and rainy 6.6-bit codewords. On our weather, where they come half the time, that wastes 1.84 bits a day: KL(p‖q).

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.4

Which one is the truth? KL is not symmetric



	bits wasted per day
our weather, their code: $\mathrm{KL}(p\|\|q)$	1.84
their weather, our code: $\mathrm{KL}(q\|\|p)$	0.86

> Swap the roles: our code on their weather wastes only 0.86 bits a day, KL(q‖p). The truth supplies the weights, so swapping changes the answer.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5

KL at work: one recipe, two applications

Two losses you may know separately are one idea. Each goes through the same five steps:

	2.5.1 Classification	2.5.2 Knowledge distillation
① **the problem**	label an image	shrink a big model
② **data, model**	images; a classifier	inputs; a teacher and a student
③ **usual loss**	cross-entropy	cross-entropy with soft labels
④ **as KL**	KL(label ‖ prediction)	KL(teacher ‖ student)
⑤ **check**	PyTorch's loss = KL	the student copies the teacher

> Two applications, one recipe: state the problem, name the data and the model, write down the loss people use, rewrite it as a KL, and check with numbers.

▶ Notebook 2.5.1

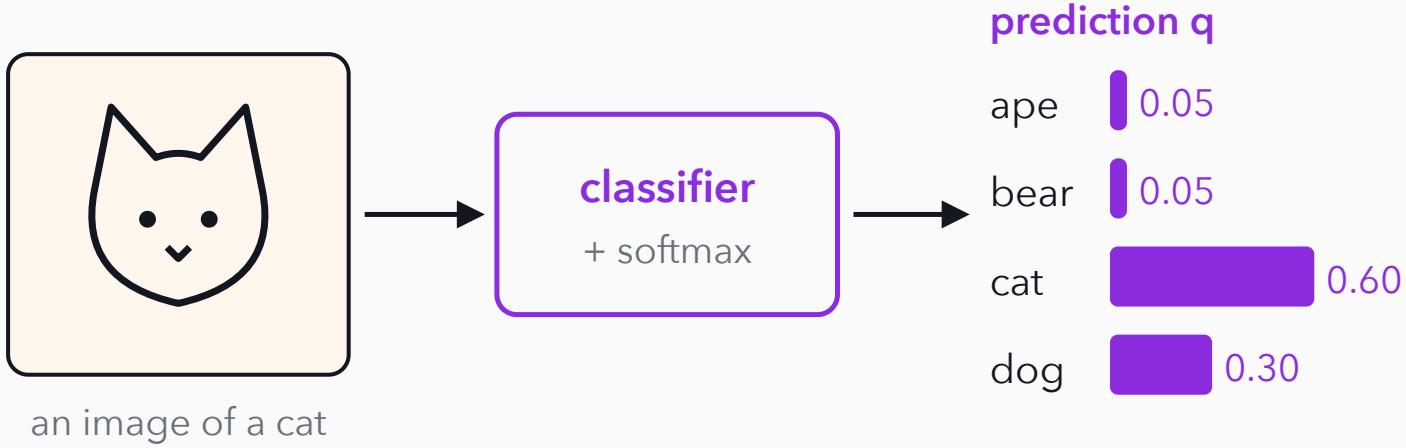
2.5.1 Classification: the plan

step	in this application
① **the problem**	Given an image, say what is in it: ape, bear, cat or dog.
② **data and model**	Images with their true labels. A network with a softmax outputs a distribution q over the four labels.
③ **the usual loss**	Cross-entropy: the loss you have used to train every classifier.
④ **the same loss as KL**	A one-hot label has no surprise, so cross-entropy = KL(label ‖ prediction).
⑤ **check**	PyTorch's loss for our cat is 0.511 nats, the KL; plotted, it is the surprise curve of 2.1.

> The first application, in five steps. You have trained classifiers before; now see which KL their loss is.

▶ Notebook 2.5.1

①② A classifier outputs a distribution

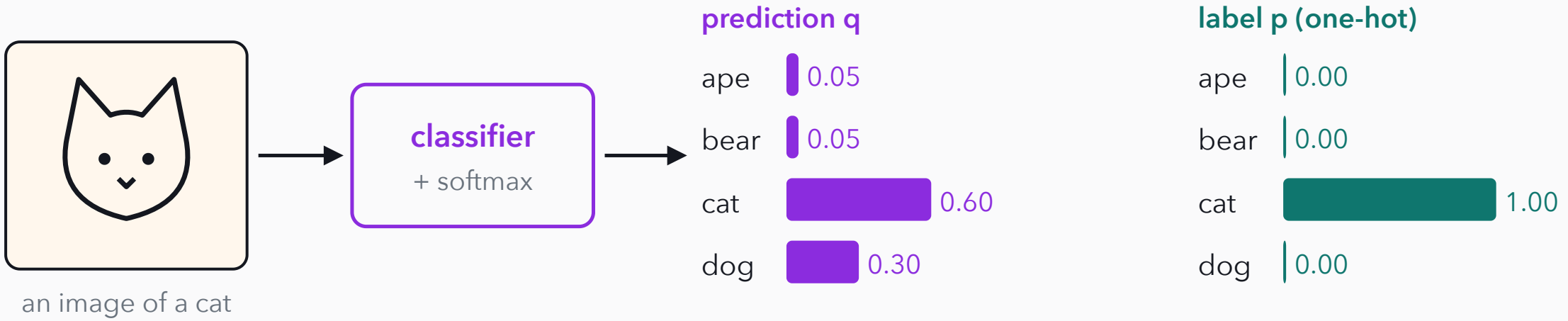


> An image goes into a classifier; out comes a distribution over four labels: ape 0.05, bear 0.05, cat 0.60, dog 0.30. Call it q.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

①② A classifier outputs a distribution



> The true label is a distribution too: all of its probability on cat. Call it p.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

③ The usual loss: surprise at the true label

$$H(p, q) = \sum_{y} p(y) \log \frac{1}{q(y)}$$

> The cross-entropy between the label p and the prediction q: the label supplies the weights.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

③ The usual loss: surprise at the true label

$$H(p, q) = \sum_y p(y) \log \frac{1}{q(y)}$$

$$= 0 \cdot \log \tfrac{1}{0.05} + 0 \cdot \log \tfrac{1}{0.05} + 1 \cdot \log \tfrac{1}{0.6} + 0 \cdot \log \tfrac{1}{0.3}$$

> Write out all four terms. Three are multiplied by zero.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

③ The usual loss: surprise at the true label

$$H(p, q) = \sum_y p(y) \log \frac{1}{q(y)}$$

$$= 0 \cdot \log \tfrac{1}{0.05} + 0 \cdot \log \tfrac{1}{0.05} + 1 \cdot \log \tfrac{1}{0.6} + 0 \cdot \log \tfrac{1}{0.3}$$

$$= -\log q(\text{cat}) = 0.51 \text{ nats} = 0.74 \text{ bits}$$

> What is left is −log q(cat): the model's surprise at the true label. That is the loss you have been minimising all along.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

④ As KL: with a one-hot label, cross-entropy = KL

$$\mathrm{KL}(\text{label} \,\|\, \text{prediction}) = H(p, q) - H(p)$$

> From Section 2.4: KL is cross-entropy minus entropy. The label is p, the truth; the prediction is q, the model.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

④ As KL: with a one-hot label, cross-entropy = KL

$$\mathrm{KL}(\text{label} \,\|\, \text{prediction}) = H(p, q) - H(p)$$

$$H(p) = 1 \cdot \log \tfrac{1}{1} + 0 + 0 + 0 = 0$$

A one-hot label is certain, so it has no surprise (0 · log 1/0 counts as 0).

> The entropy of a one-hot label is 0: a certain outcome carries no surprise.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

④ As KL: with a one-hot label, cross-entropy = KL

$$\mathrm{KL}(\text{label} \,\|\, \text{prediction}) = H(p, q) - H(p)$$

$$H(p) = 1 \cdot \log \tfrac{1}{1} + 0 + 0 + 0 = 0$$

A one-hot label is certain, so it has no surprise (0 · log 1/0 counts as 0).

```
F.cross_entropy(logits, target)   # 0.511 nats: here equal to KL(p ‖ q)
```

> So the classification loss is exactly KL(label ‖ prediction). PyTorch computes it in nats: 0.511 for our cat.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

④ As KL: with a one-hot label, cross-entropy = KL

$$\mathrm{KL}(\text{label} \,\|\, \text{prediction}) = H(p, q) - H(p)$$

$$H(p) = 1 \cdot \log \tfrac{1}{1} + 0 + 0 + 0 = 0$$

A one-hot label is certain, so it has no surprise (0 · log 1/0 counts as 0).

```
F.cross_entropy(logits, target)   # 0.511 nats: here equal to KL(p ‖ q)
```

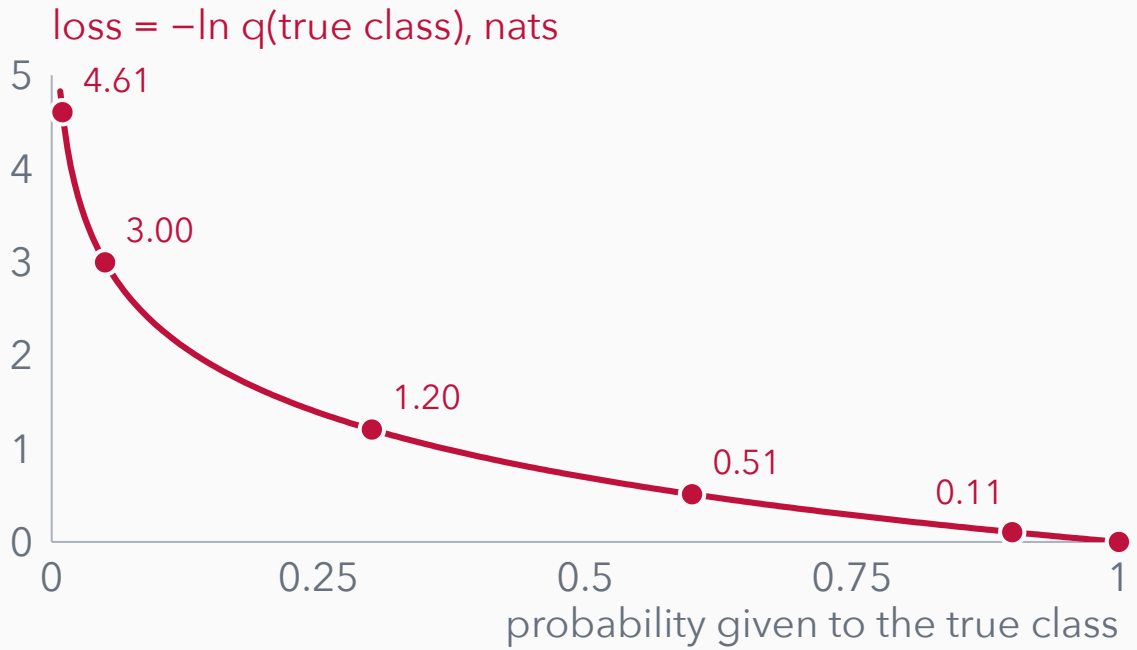
Why label ‖ prediction? The label is the reference: the data come from it, so it goes on the left. The prediction is the model, on the right. Swapped, KL(prediction ‖ label) would need log q/0 for every wrong class: infinite.

> Why this order? Data from the left, code from the right: the label generates the data, the prediction encodes it. The other order divides by the label's zeros.

Story after Jia-Bin Huang, "Fantastic KL Divergence and How to (Actually) Compute It" (YouTube)

▶ Notebook 2.5.1

⑤ Check: the loss is the surprise curve

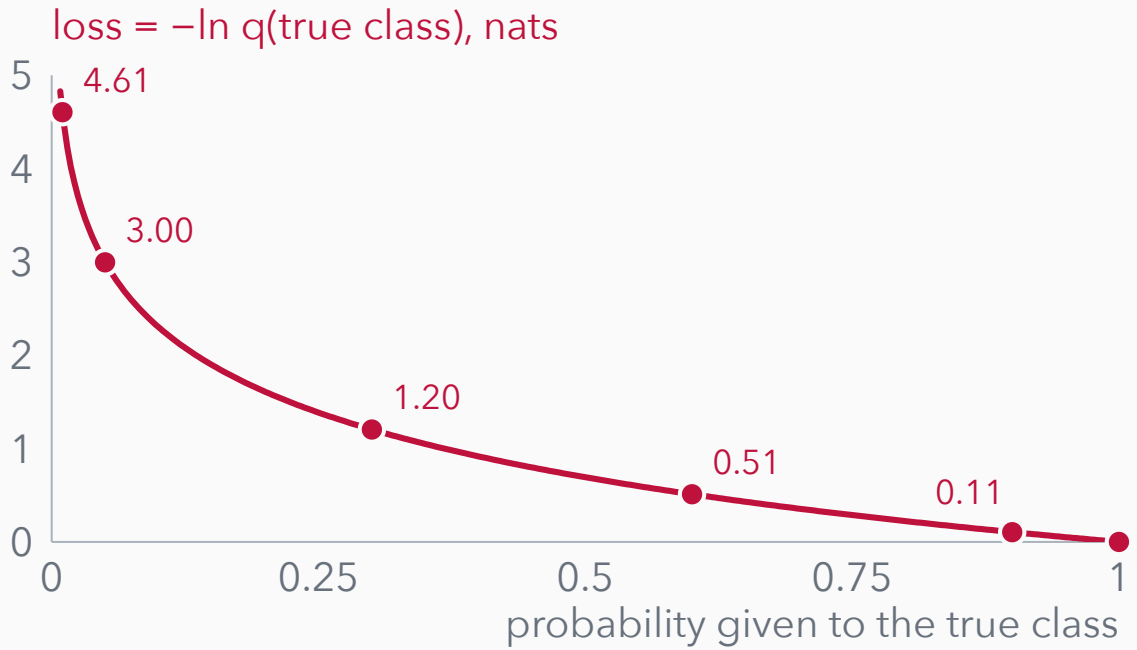


The loss is −ln q(true class): the surprise curve of Section 2.1, in nats.

Plot the loss against the probability the model gives to the true class: it is the surprise curve again.

▶ Notebook 2.5.1

⑤ Check: the loss is the surprise curve



The loss is −ln q(true class): the surprise curve of Section 2.1, in nats.

Confident and wrong is expensive: q(true) = 0.01 costs 4.6 nats; q(true) = 0.6 costs 0.51.

Training pushes q(true) up the curve towards 1. Being confidently wrong is punished hardest.

▶ Notebook 2.5.2

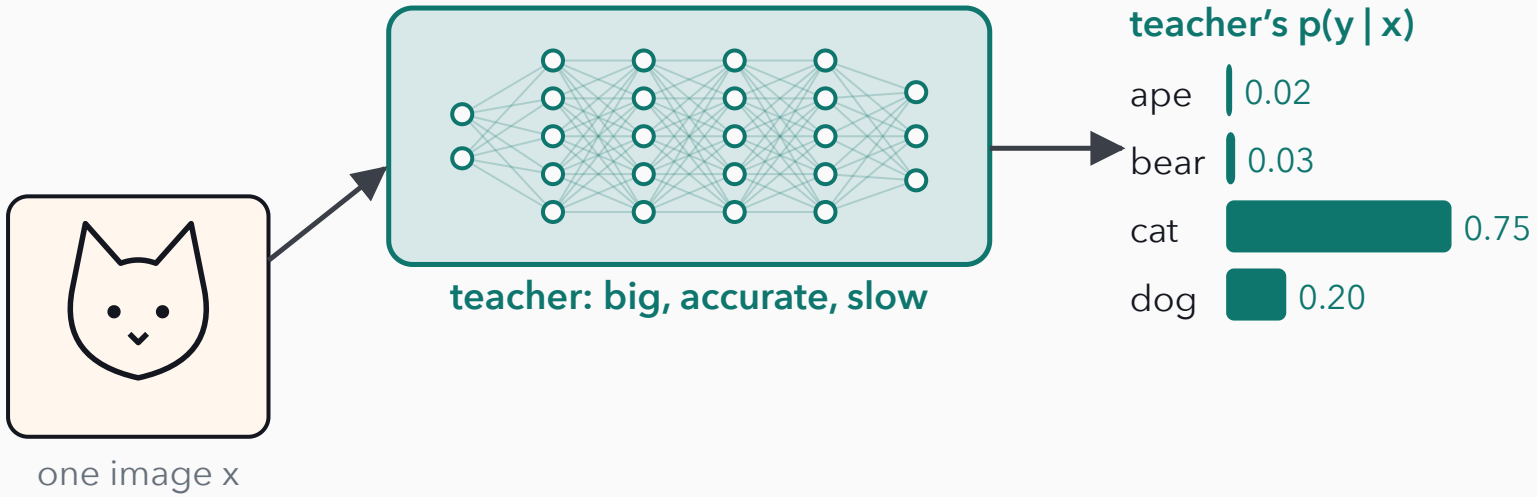
2.5.2 Knowledge distillation: the plan

step	in this application
① **the problem**	A big, accurate network is too slow and costly to deploy. We want a small network that behaves like it.
② **data and model**	Inputs, labelled or not, go to both networks. The teacher is frozen; only the student is trained.
③ **the usual loss**	Cross-entropy, with the teacher's probabilities as soft labels in place of the one-hot label.
④ **the same loss as KL**	Cross-entropy = H(teacher) + KL(teacher ‖ student); the teacher is fixed, so this minimises the KL.
⑤ **check**	For our cat 0.77 = 0.72 + 0.05 nats; a 27-parameter student learns a 4,547-parameter teacher.

> The third application: compressing a big model into a small one. The same five steps; now the "truth" is the teacher.

▶ Notebook 2.5.2

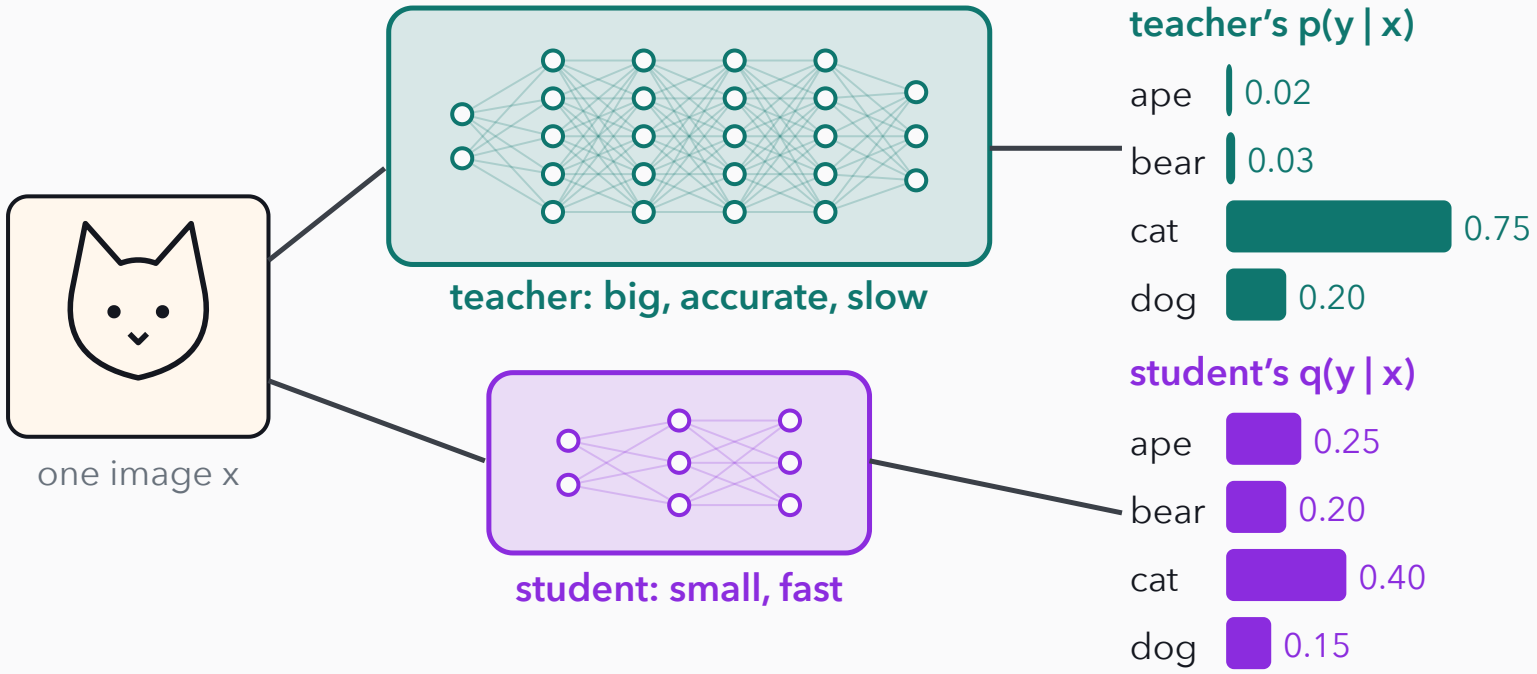
① The problem: a big teacher, a small student



> A big network, the teacher, is accurate but too slow and too expensive to run on a phone, or a million times a day.

▶ Notebook 2.5.2

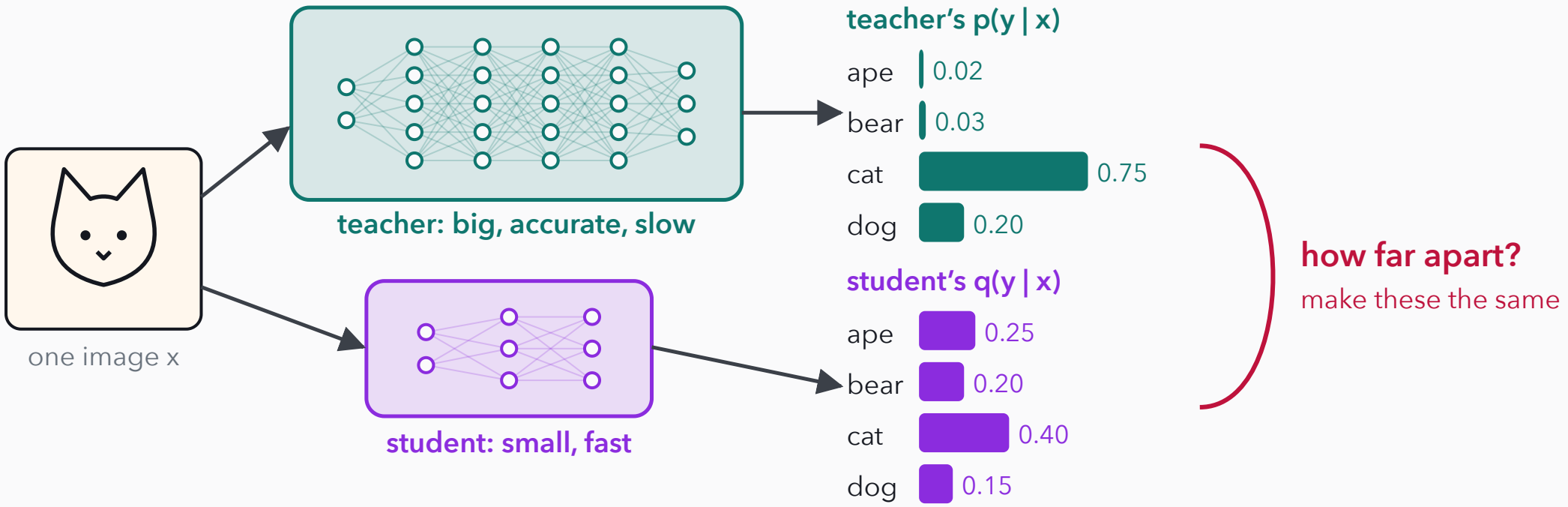
① The problem: a big teacher, a small student



> We want a small network, the student, that behaves like it. Give both the same image: the teacher says cat 0.75, dog 0.20; the untrained student is vaguer.

▶ Notebook 2.5.2

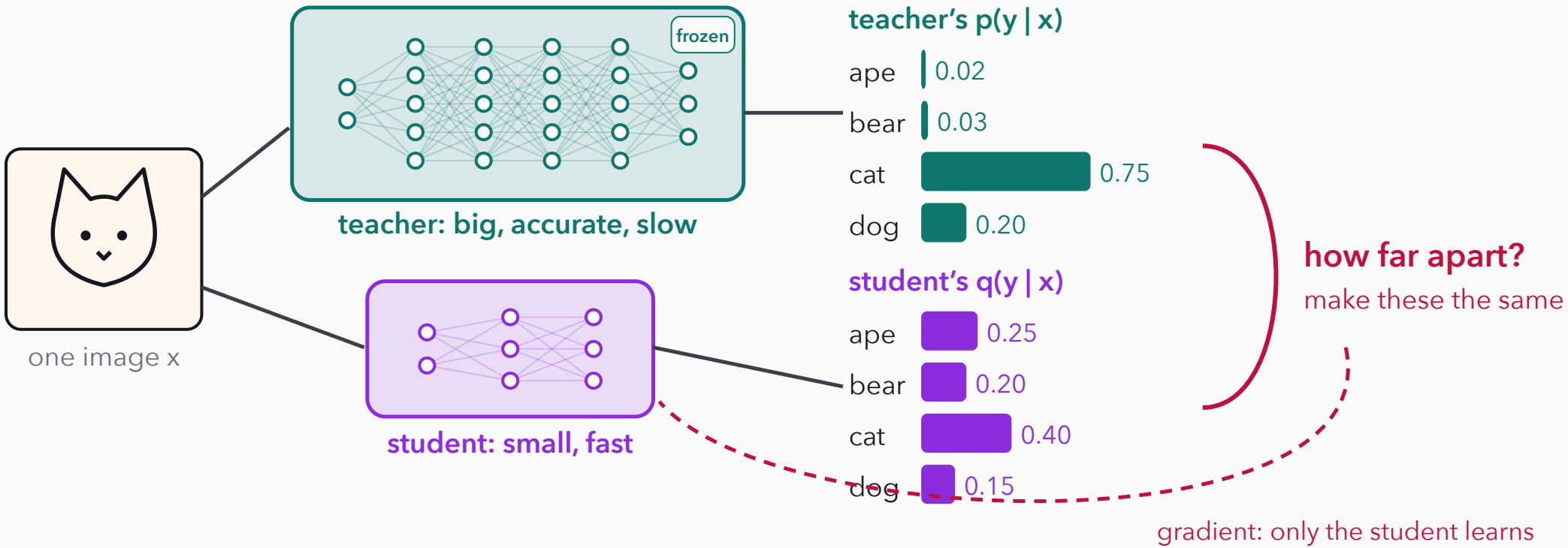
② The data and the model: same inputs, two networks



> The data: inputs only, such as images, labelled or not. Each goes through both networks; the teacher's distribution is the target for the student's.

▶ Notebook 2.5.2

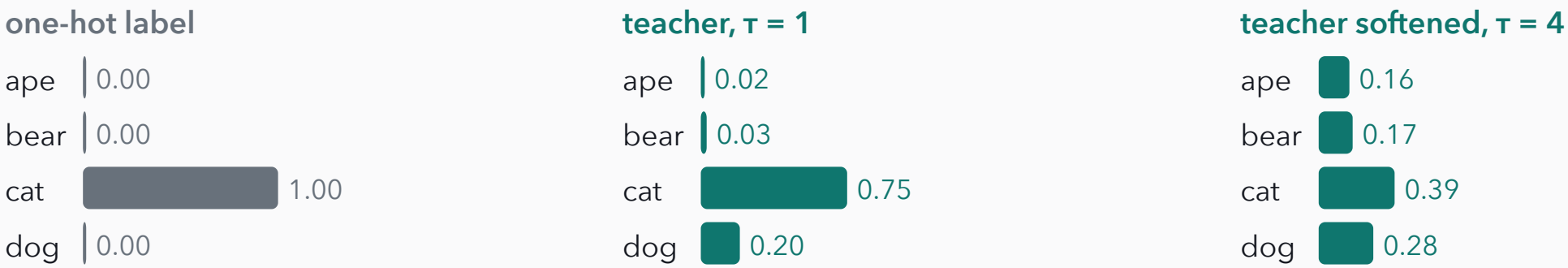
② The data and the model: same inputs, two networks



> The model: the student, the only network we train; the teacher is frozen. We need a loss that measures how far apart the two distributions are.

▶ Notebook 2.5.2

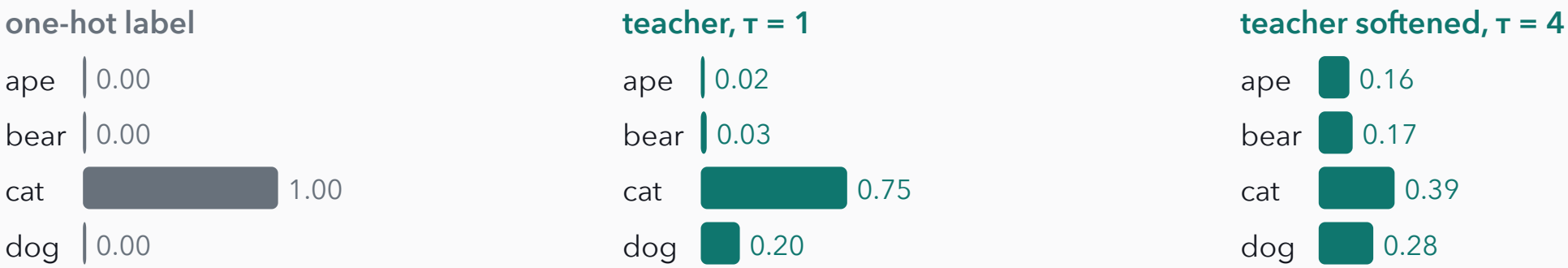
③ The usual loss: cross-entropy with soft labels



> The usual loss, from Hinton's paper: the classification loss of 2.5.1, but against the teacher's probabilities instead of the one-hot label. Compare the two targets.

▶ Notebook 2.5.2

③ The usual loss: cross-entropy with soft labels



$$\mathcal{L} = H(p_T, q_S) = -\sum_y p_T(y)\ \log q_S(y)$$

The classification loss of 2.5.1, with the teacher's probabilities in place of the one-hot label. Their small values say "a bit like a dog, not like an ape": Hinton's dark knowledge. A temperature τ = 4 turns them up.

> The loss is the cross-entropy $H(p_T, q_S)$: the student's surprise, weighted by the teacher's probabilities. The small ones say which mistakes are reasonable.

▶ Notebook 2.5.2

④ The same loss as KL, term by term

$$H(p_T, q_S) = \underbrace{H(p_T)}_{\text{teacher only}} + \mathrm{KL}(p_T \,\|\, q_S)$$

Split the cross-entropy of step ③, as in Section 2.4: the teacher's entropy plus KL(teacher ‖ student).

▶ Notebook 2.5.2

④ The same loss as KL, term by term

$$H(p_T, q_S) = \underbrace{H(p_T)}_{\text{teacher only}} + \mathrm{KL}(p_T \,\|\, q_S)$$

$$\text{teacher frozen} \;\Rightarrow\; \min_{\text{student}} H(p_T, q_S) \;\Longleftrightarrow\; \min_{\text{student}} \mathrm{KL}(p_T \,\|\, q_S)$$

> The teacher is frozen, so its entropy is a constant: minimising the cross-entropy is minimising the KL. Two formulations, one answer.

▶ Notebook 2.5.2

④ The same loss as KL, term by term

$$H(p_T, q_S) = \underbrace{H(p_T)}_{\text{teacher only}} + \mathrm{KL}(p_T \,\|\, q_S)$$

$$\text{teacher frozen} \;\Rightarrow\; \min_{\text{student}} H(p_T, q_S) \;\Longleftrightarrow\; \min_{\text{student}} \mathrm{KL}(p_T \,\|\, q_S)$$

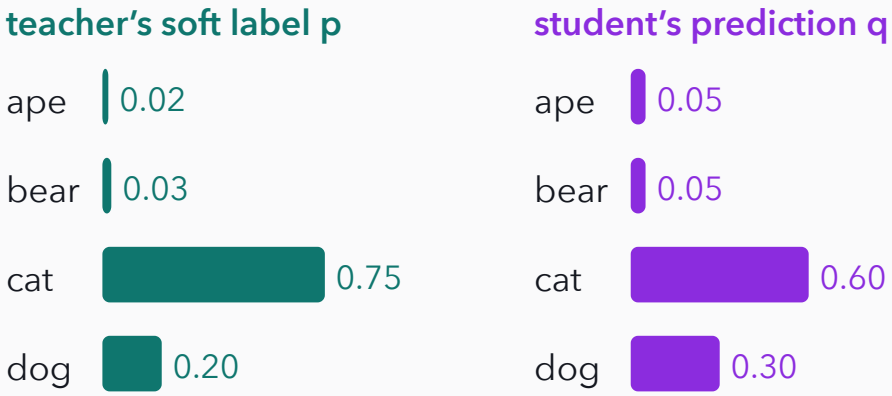
$$\mathcal{L} = (1-\alpha)\, H(y_{\text{true}}, q_S) + \alpha\, \tau^2\, \mathrm{KL}\big(p_T^{(\tau)} \,\|\, q_S^{(\tau)}\big)$$

In practice: $p^{(\tau)}$ = softmax(logits / τ) softens both distributions, τ^2 keeps the gradient scale comparable, and α blends in the true labels. Each term is averaged over the inputs.

> In practice both distributions are softened with a temperature τ, and a small term with the true labels is added.

▶ Notebook 2.5.1

⑤ Check with numbers: cross-entropy = entropy + KL

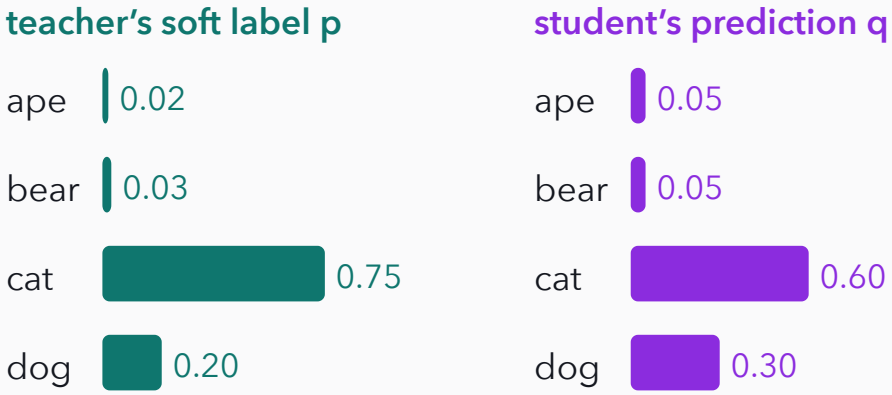


$$H(p, q) = \underbrace{H(p)}_{0.72} + \underbrace{\mathrm{KL}(p \,\|\, q)}_{0.05} = 0.77$$

> Our cat again. The teacher says cat 0.75, dog 0.20; the student predicts the q of 2.5.1. Cross-entropy 0.77 = entropy 0.72 + KL 0.05 nats.

▶ Notebook 2.5.1

⑤ Check with numbers: cross-entropy = entropy + KL



$$H(p, q) = \underbrace{H(p)}_{0.72} + \underbrace{\mathrm{KL}(p \,\|\, q)}_{0.05} = 0.77$$

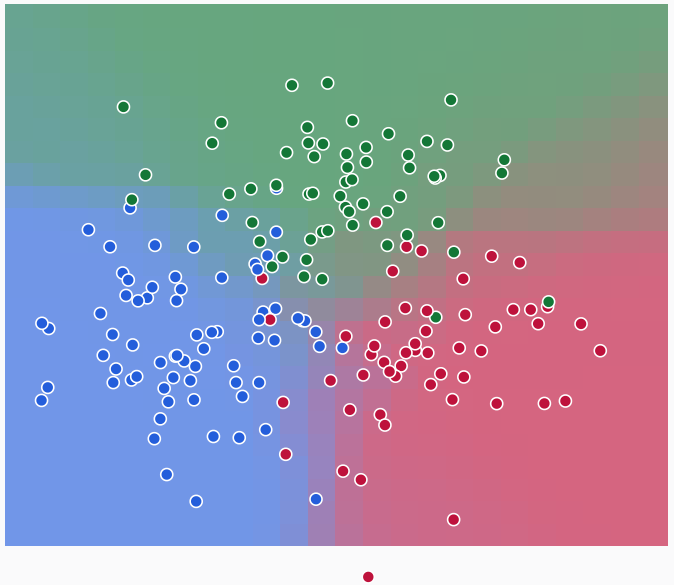
H(p) is fixed by the teacher, so lowering the cross-entropy lowers the KL by exactly as much. The best the student can reach is 0.72 nats: a copy of the teacher.

> Only the KL depends on the student, so lowering the cross-entropy lowers the KL by exactly as much.

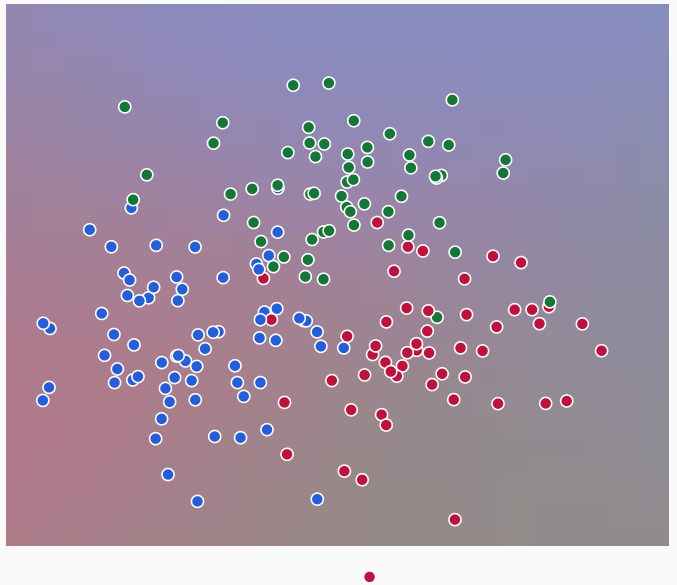
▶ Notebook 2.5.2

⑤ Check: a tiny student learns a big teacher

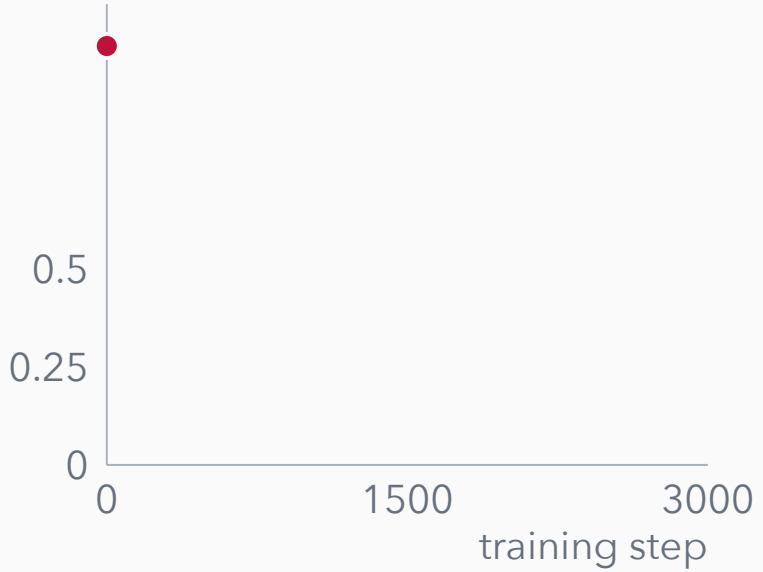
teacher · 4,547 parameters



student · 27 parameters · step 0



KL(teacher ‖ student) = 1.067

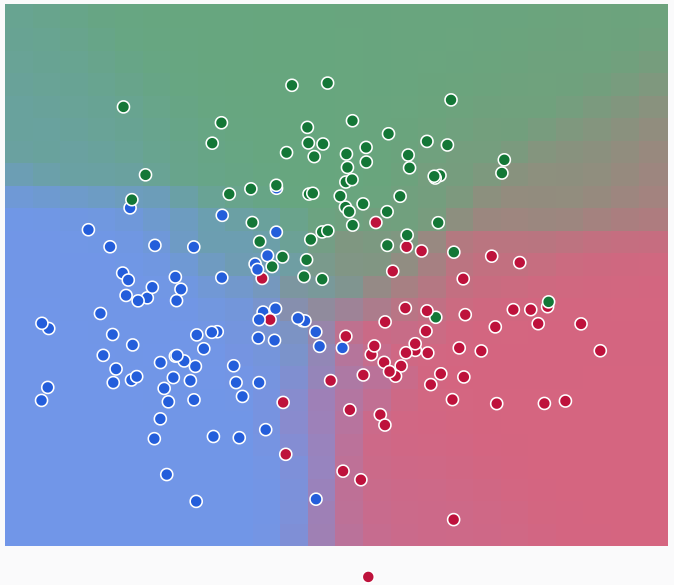


Three overlapping classes in the plane. The teacher has 4,547 parameters, the student 27. Colours show each network's class probabilities. Predict: can the student copy the teacher?

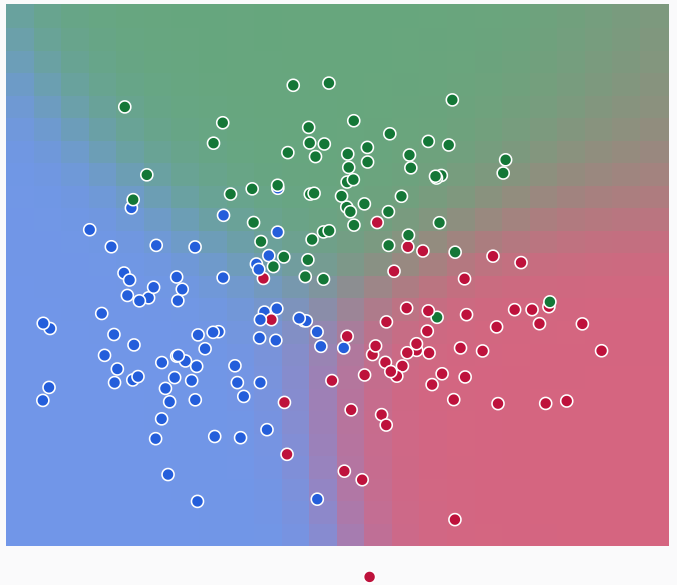
▶ Notebook 2.5.2

⑤ Check: a tiny student learns a big teacher

teacher · 4,547 parameters



student · 27 parameters · step 3000



KL(teacher ‖ student) = 0.009



Test accuracy: teacher 84.6%, student 85.0%; the student agrees with the teacher on 97.0% of test points, with 168× fewer parameters.

> Press play. The student's map turns into the teacher's, uncertain boundaries included, and KL(teacher ‖ student) falls from 1.07 to 0.009 nats.

▶ Notebook 2.5

One principle in many disguises

application	the usual loss	is the same as minimising	the truth
2.5.1 classification	cross-entropy	KL(label ‖ prediction)	the label
2.5.2 distillation	cross-entropy with soft labels	KL(teacher ‖ student)	the teacher

> Two losses you may have met separately are one idea: make the model match the truth, measured by KL with the truth first.

▶ Notebook 2.5

One principle in many disguises

application	the usual loss	is the same as minimising	the truth
2.5.1 classification	cross-entropy	KL(label ‖ prediction)	the label
2.5.2 distillation	cross-entropy with soft labels	KL(teacher ‖ student)	the teacher

Parts 3–5: variational inference minimises KL(q ‖ posterior), the same KL with the order flipped: our q comes first.

KL turns up in many other places too: for example, when chatbots are fine-tuned with human feedback (RLHF), a KL term keeps the model close to where it started.

> Variational inference flips the order: q comes first, because we can sample from q but not from the posterior. Section 3.6 shows what that changes.

▶ Notebook 2.6

From sums to integrals

$$\text{discrete:} \qquad \mathrm{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$\text{continuous:} \qquad \mathrm{KL}(p\|q) = \int p(\theta) \log \frac{p(\theta)}{q(\theta)}\, d\theta$$

For a continuous unknown like θ, the sum over grades becomes an integral over θ.

▶ Notebook 2.6

From sums to integrals

$$\text{discrete:} \qquad \mathrm{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$\text{continuous:} \qquad \mathrm{KL}(p\|q) = \int p(\theta) \log \frac{p(\theta)}{q(\theta)}\, d\theta$$

Probabilities become densities and the sum becomes an integral; the recipe is the same weighted average of log ratios.

> Nothing else changes. KL is never negative either; An optional slide proves it, by Jensen's inequality.

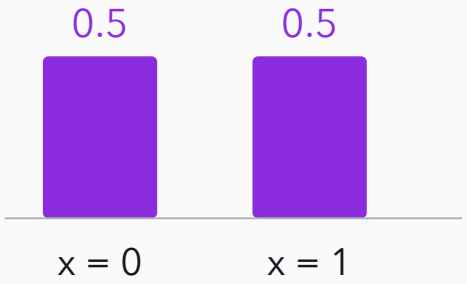
▶ Notebook 2.6

Bernoulli: a coin with P(x = 1) = a

p = Bernoulli(0.8)



q = Bernoulli(0.5)



$$p(x) = a^x(1-a)^{1-x}, \qquad x \in \{0, 1\}$$

$$p(1) = a, \qquad p(0) = 1 - a$$

Bernoulli(a): x is 1 with probability a and 0 otherwise. Here p = Bernoulli(0.8), q = Bernoulli(0.5).

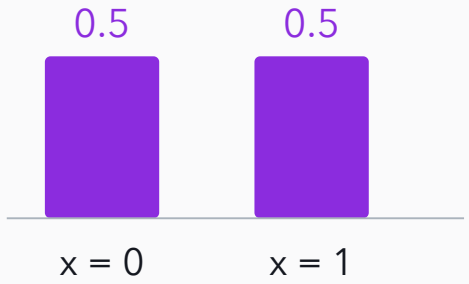
▶ Notebook 2.6

Bernoulli: a coin with P(x = 1) = a

p = Bernoulli(0.8)



q = Bernoulli(0.5)



$$p(x) = a^x(1-a)^{1-x}, \qquad x \in \{0, 1\}$$

$$p(1) = a, \qquad p(0) = 1 - a$$

Your turn: write $\mathrm{KL}(p \,\|\, q)$ for p = Bernoulli(a) and q = Bernoulli(b).

Your turn: put the PMF into Σ_x p(x) log p(x)/q(x). There are only two values of x.

▶ Notebook 2.6

KL between two Bernoullis, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{x \in \{0,1\}} p(x) \log \frac{p(x)}{q(x)}$$

Start from the definition: a sum over the values of x.

▶ Notebook 2.6

KL between two Bernoullis, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{x \in \{0,1\}} p(x) \log \frac{p(x)}{q(x)}$$

$$= p(1) \log \frac{p(1)}{q(1)} + p(0) \log \frac{p(0)}{q(0)}$$

There are two: write out the term for x = 1 and the term for x = 0.

▶ Notebook 2.6

KL between two Bernoullis, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{x \in \{0,1\}} p(x) \log \frac{p(x)}{q(x)}$$

$$= p(1) \log \frac{p(1)}{q(1)} + p(0) \log \frac{p(0)}{q(0)}$$

$$= a \log \frac{a}{b} + (1 - a) \log \frac{1 - a}{1 - b}$$

Substitute p(1) = a, p(0) = 1 − a, and b for q.

▶ Notebook 2.6

KL between two Bernoullis, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{x \in \{0,1\}} p(x) \log \frac{p(x)}{q(x)}$$

$$= p(1) \log \frac{p(1)}{q(1)} + p(0) \log \frac{p(0)}{q(0)}$$

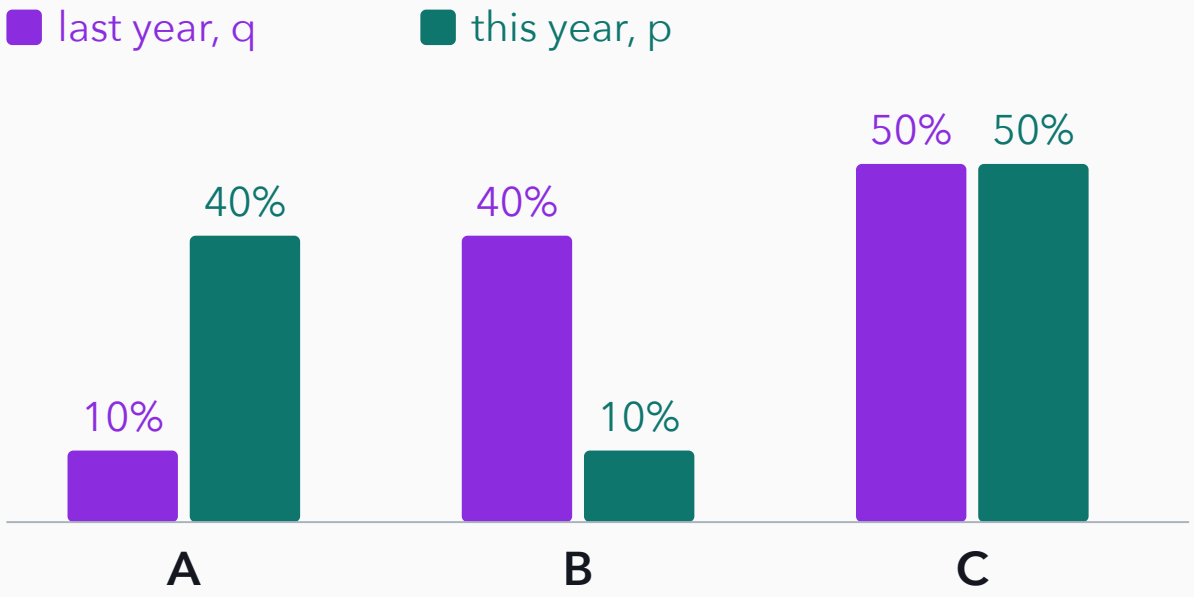
$$= a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b}$$

$$a = 0.8,\ b = 0.5: \quad 0.8 \log 1.6 + 0.2 \log 0.4 = 0.376 - 0.183 = 0.19$$

For a = 0.8 and b = 0.5: 0.38 − 0.18 = 0.19 nats.

▶ Notebook 2.6

Categorical: K outcomes with probabilities $\pi_1, \ldots, \pi_K$



$$p(x = k) = \pi_k, \quad k = 1, \ldots, K$$

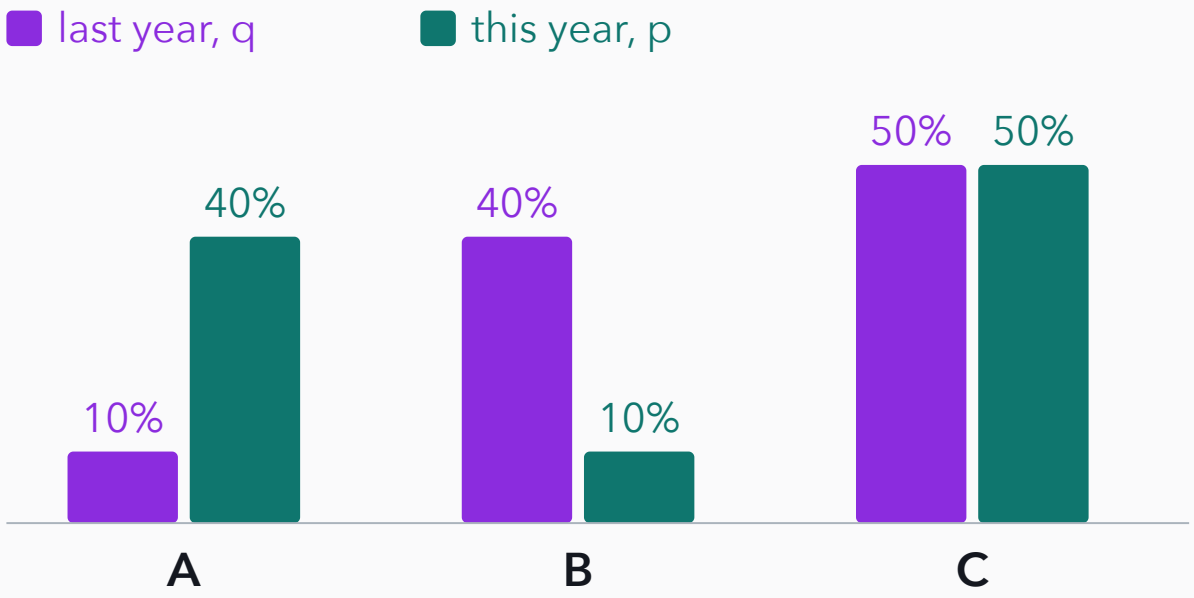
$$\pi_k \geq 0, \qquad \sum_k \pi_k = 1$$

Our grades: K = 3, with π = (0.4, 0.1, 0.5) this year and ρ = (0.1, 0.4, 0.5) last year.

Categorical(π): one of K outcomes, outcome k with probability π_k. Our grade distributions are categorical, with K = 3.

▶ Notebook 2.6

Categorical: K outcomes with probabilities $\pi_1, \ldots, \pi_K$



$$p(x = k) = \pi_k, \quad k = 1, \ldots, K$$

$$\pi_k \geq 0, \qquad \sum_k \pi_k = 1$$

Our grades: K = 3, with π = (0.4, 0.1, 0.5) this year and ρ = (0.1, 0.4, 0.5) last year.

Your turn: write $\mathrm{KL}(p \,\|\, q)$ for p = Categorical(π) and q = Categorical(ρ).

> Your turn: the same recipe, now with K terms.

▶ Notebook 2.6

KL between two categoricals, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{k=1}^{K} p(k) \log \frac{p(k)}{q(k)} = \sum_{k=1}^{K} \pi_k \log \frac{\pi_k}{\rho_k}$$

Substitute p(k) = π_k and q(k) = ρ_k: one term per outcome.

▶ Notebook 2.6

KL between two categoricals, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{k=1}^{K} p(k) \log \frac{p(k)}{q(k)} = \sum_{k=1}^{K} \pi_k \log \frac{\pi_k}{\rho_k}$$

$$K = 2: \quad \pi = (a,\, 1-a),\ \rho = (b,\, 1-b) \ \Rightarrow\ a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b}$$

With K = 2 this is exactly the Bernoulli formula: a Bernoulli is a categorical with two outcomes.

▶ Notebook 2.6

KL between two categoricals, step by step

$$\mathrm{KL}(p \,\|\, q) = \sum_{k=1}^{K} p(k) \log \frac{p(k)}{q(k)} = \sum_{k=1}^{K} \pi_k \log \frac{\pi_k}{\rho_k}$$

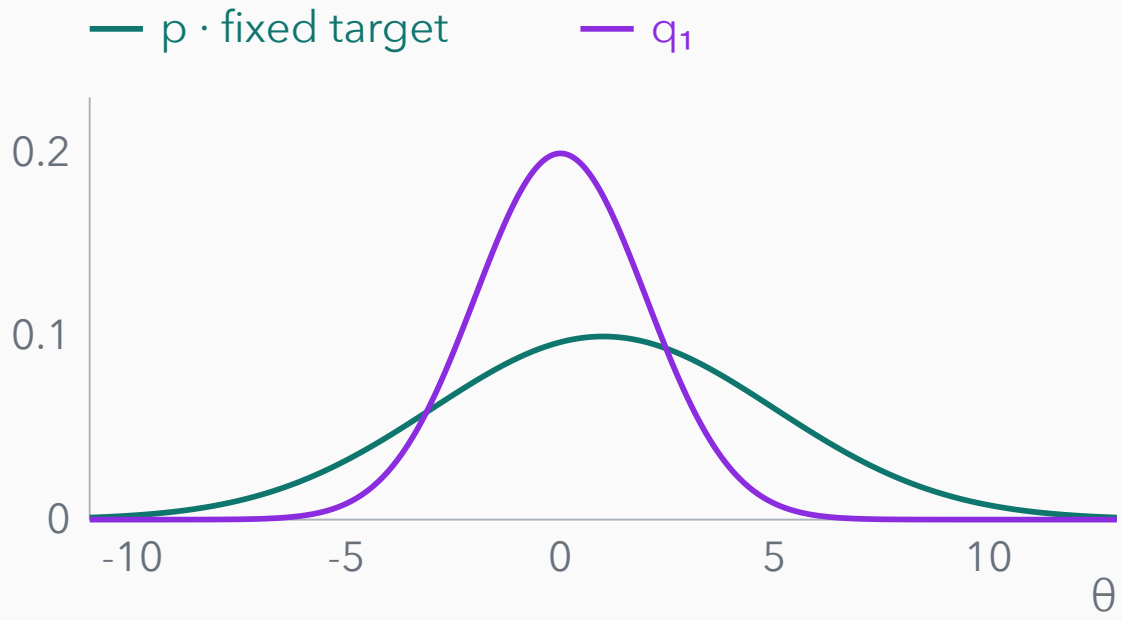
$$K = 2: \quad \pi = (a,\, 1-a),\ \rho = (b,\, 1-b) \ \Rightarrow\ a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b}$$

$$\text{grades: } 0.4 \log 4 + 0.1 \log \tfrac{1}{4} + 0.5 \log 1 = 0.42$$

For the grades: 0.42 nats, the number we built by hand in Way 1.

▶ Notebook 2.6

The Gaussian: a bell with mean μ and width σ



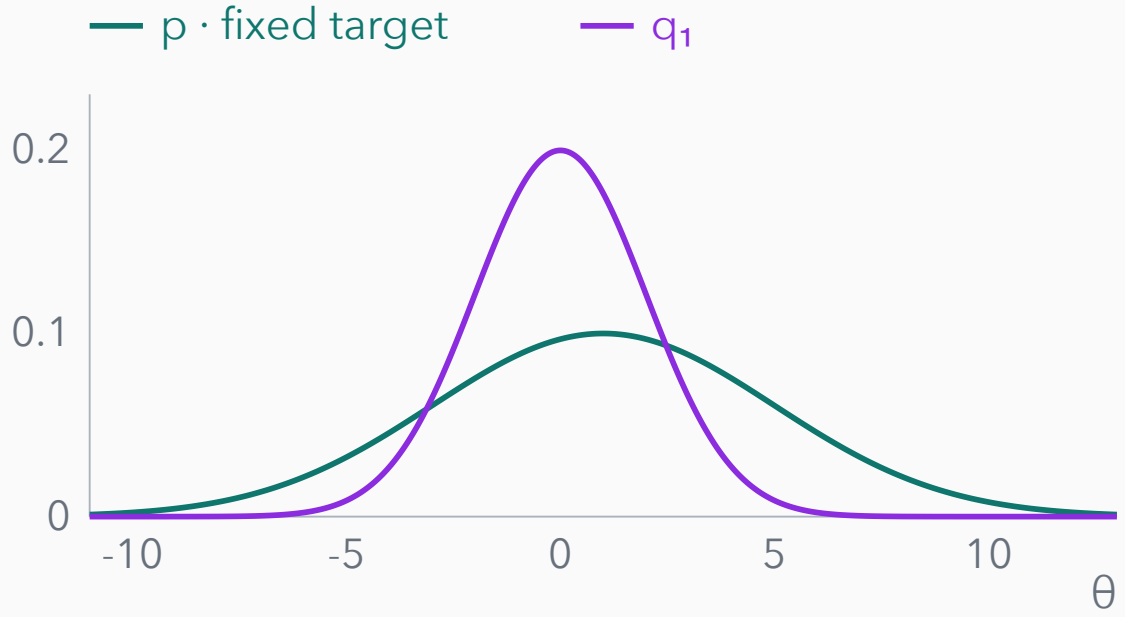
$$\mathcal{N}(\theta; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-(\theta-\mu)^2/(2\sigma^2)}$$

q = N(μ, σ²) is our approximation and p = N(m, s²) the pretend posterior. Shown: q = N(0, 2²), p = N(1, 4²).

> The Gaussian density: a normalising constant times e to the minus squared distance from the mean, scaled by the variance.

▶ Notebook 2.6

The Gaussian: a bell with mean μ and width σ



$$\mathcal{N}(\theta; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-(\theta-\mu)^2/(2\sigma^2)}$$

q = N(μ, σ²) is our approximation and p = N(m, s²) the pretend posterior. Shown: q = N(0, 2²), p = N(1, 4²).

Your turn: work out $\mathrm{KL}(q \,\|\, p)$. Start with log q(θ) − log p(θ).

> Your turn: work out KL(q ‖ p). The log makes the exponentials easy.

▶ Notebook 2.6

Gaussian KL, step 1: the log ratio

$$\log q(\theta) = -\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{(\theta-\mu)^2}{2\sigma^2}$$

Take the log of q's density: the exponential disappears.

▶ Notebook 2.6

Gaussian KL, step 1: the log ratio

$$\log q(\theta) = -\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{(\theta-\mu)^2}{2\sigma^2}$$

$$\log p(\theta) = -\tfrac{1}{2}\log(2\pi s^2) - \frac{(\theta-m)^2}{2s^2}$$

The same for p, with its own mean m and width s.

▶ Notebook 2.6

Gaussian KL, step 1: the log ratio

$$\log q(\theta) = -\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{(\theta-\mu)^2}{2\sigma^2}$$

$$\log p(\theta) = -\tfrac{1}{2}\log(2\pi s^2) - \frac{(\theta-m)^2}{2s^2}$$

$$\log q(\theta) - \log p(\theta) = \log\frac{s}{\sigma} - \frac{(\theta-\mu)^2}{2\sigma^2} + \frac{(\theta-m)^2}{2s^2}$$

Subtract. The 2π terms cancel, and the two log widths combine into log(s/σ).

▶ Notebook 2.6

Gaussian KL, step 2: average over q

$$\mathrm{KL}(q\|p) = \mathbb{E}_q\left[\log\frac{s}{\sigma} - \frac{(\theta-\mu)^2}{2\sigma^2} + \frac{(\theta-m)^2}{2s^2}\right]$$

Average the log ratio over θ drawn from q. The first term is a constant, so it stays as it is.

▶ Notebook 2.6

Gaussian KL, step 2: average over q

$$\mathrm{KL}(q\|p) = \mathbb{E}_q\left[\log\frac{s}{\sigma} - \frac{(\theta-\mu)^2}{2\sigma^2} + \frac{(\theta-m)^2}{2s^2}\right]$$

$$\mathbb{E}_q\left[(\theta-\mu)^2\right] = \sigma^2 \quad \Rightarrow \quad -\frac{\sigma^2}{2\sigma^2} = -\frac{1}{2}$$

The second: the average squared distance from q's own mean is its variance, σ². That term becomes −½.

▶ Notebook 2.6

Gaussian KL, step 2: average over q

$$\mathrm{KL}(q\|p) = \mathbb{E}_q\left[\log\frac{s}{\sigma} - \frac{(\theta-\mu)^2}{2\sigma^2} + \frac{(\theta-m)^2}{2s^2}\right]$$

$$\mathbb{E}_q\left[(\theta-\mu)^2\right] = \sigma^2 \quad \Rightarrow \quad -\frac{\sigma^2}{2\sigma^2} = -\frac{1}{2}$$

$$\mathbb{E}_q\left[(\theta-m)^2\right] = \mathbb{E}_q\left[\big((\theta-\mu)+(\mu-m)\big)^2\right] = \sigma^2 + (\mu-m)^2$$

The third: split θ − m into (θ − μ) + (μ − m). The cross term averages to zero, leaving σ² + (μ − m)².

▶ Notebook 2.6

Gaussian KL, step 2: average over q

$$\mathrm{KL}(q\|p) = \mathbb{E}_q\Big[\log\frac{s}{\sigma} - \frac{(\theta-\mu)^2}{2\sigma^2} + \frac{(\theta-m)^2}{2s^2}\Big]$$

$$\mathbb{E}_q\big[(\theta-\mu)^2\big] = \sigma^2 \quad\Rightarrow\quad -\frac{\sigma^2}{2\sigma^2} = -\frac{1}{2}$$

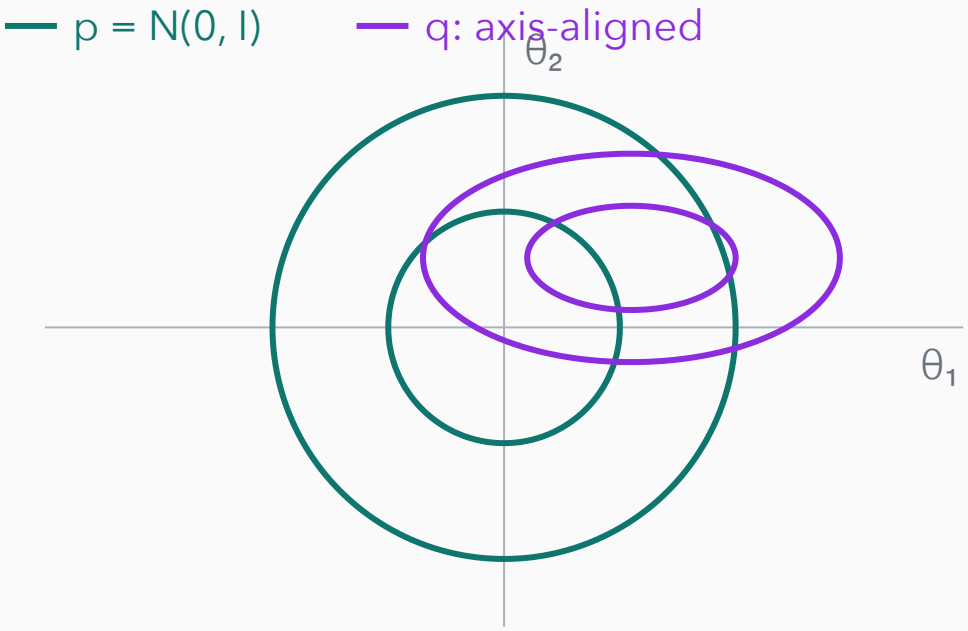
$$\mathbb{E}_q\big[(\theta-m)^2\big] = \mathbb{E}_q\big[\big((\theta-\mu)+(\mu-m)\big)^2\big] = \sigma^2 + (\mu-m)^2$$

$$\mathrm{KL}(q\|p) = \log\frac{s}{\sigma} + \frac{\sigma^2+(\mu-m)^2}{2s^2} - \frac{1}{2}$$

Together: a width term, the squared shift between the means, and −½. That is the Gaussian KL.

▶ Notebook 2.6

A diagonal Gaussian: one bell per coordinate



$$q(\boldsymbol{\theta}) = \prod_{j=1}^{d} \mathcal{N}(\theta_j;\, \mu_j,\, \sigma_j^2)$$

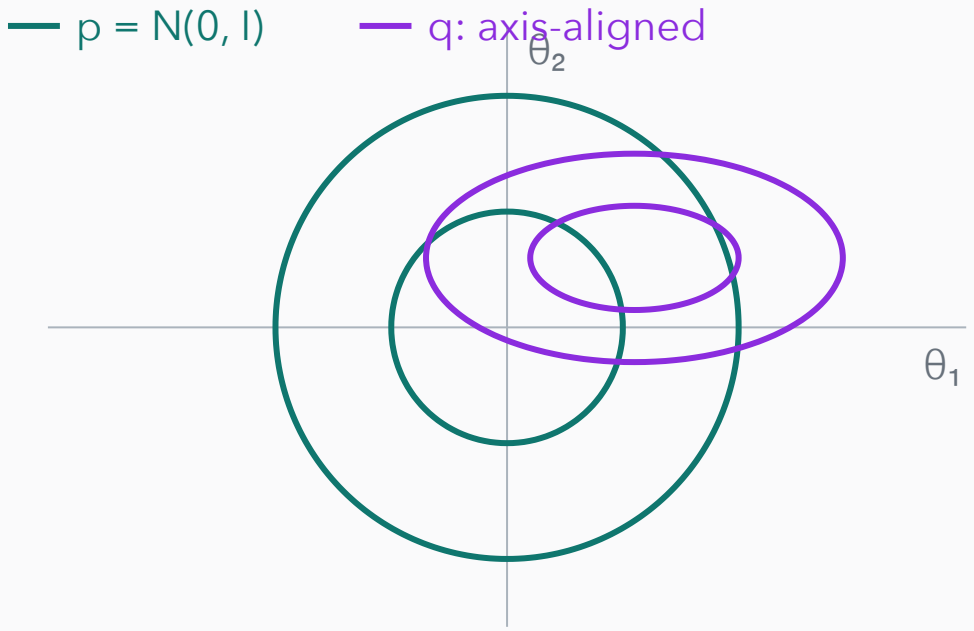
$$p(\boldsymbol{\theta}) = \prod_{j=1}^{d} \mathcal{N}(\theta_j;\, m_j,\, s_j^2)$$

Independent coordinates, so the contours are axis-aligned. Our network's q is exactly this, with d = 7.

> A diagonal Gaussian is a product of one-dimensional Gaussians: each coordinate has its own mean and width.

▶ Notebook 2.6

A diagonal Gaussian: one bell per coordinate



$$q(\boldsymbol{\theta}) = \prod_{j=1}^{d} \mathcal{N}(\theta_j;\, \mu_j,\, \sigma_j^2)$$

$$p(\boldsymbol{\theta}) = \prod_{j=1}^{d} \mathcal{N}(\theta_j;\, m_j,\, s_j^2)$$

Independent coordinates, so the contours are axis-aligned. Our network's q is exactly this, with d = 7.

Your turn: what is KL(q ‖ p) in d dimensions?

> Your turn: what happens to the log of a product?

▶ Notebook 2.6

KL between diagonal Gaussians, step by step

$$\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} = \sum_{j=1}^{d} \log \frac{\mathcal{N}(\theta_j; \mu_j, \sigma_j^2)}{\mathcal{N}(\theta_j; m_j, s_j^2)}$$

The log of a product is a sum of logs: the log ratio splits into one term per coordinate.

▶ Notebook 2.6

KL between diagonal Gaussians, step by step

$$\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} = \sum_{j=1}^{d} \log \frac{\mathcal{N}(\theta_j; \mu_j, \sigma_j^2)}{\mathcal{N}(\theta_j; m_j, s_j^2)}$$

$$\mathrm{KL}(q \| p) = \sum_{j=1}^{d} \mathrm{KL}\big(\mathcal{N}(\mu_j, \sigma_j^2) \,\|\, \mathcal{N}(m_j, s_j^2)\big)$$

Each term involves only θ_j, so its average under q is a one-dimensional Gaussian KL. The total is their sum.

▶ Notebook 2.6

KL between diagonal Gaussians, step by step

$$\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} = \sum_{j=1}^{d} \log \frac{\mathcal{N}(\theta_j; \mu_j, \sigma_j^2)}{\mathcal{N}(\theta_j; m_j, s_j^2)}$$

$$\mathrm{KL}(q\|p) = \sum_{j=1}^{d} \mathrm{KL}\big(\mathcal{N}(\mu_j, \sigma_j^2) \,\|\, \mathcal{N}(m_j, s_j^2)\big)$$

$$p = \mathcal{N}(0, I): \qquad \mathrm{KL}(q\|p) = \sum_{j=1}^{d} \Big[-\log \sigma_j + \frac{\sigma_j^2 + \mu_j^2}{2} - \frac{1}{2} \Big]$$

> Against a N(0, 1) prior on every coordinate: exactly the KL term in our network's ELBO, and in every VAE.

▶ Notebook 2.6

The same formulas in PyTorch: Bernoulli and categorical

```python
@register_kl(Bernoulli, Bernoulli)
def _kl_bernoulli_bernoulli(p, q):
    t1 = p.probs * (softplus(-q.logits) - softplus(-p.logits))
    t2 = (1 - p.probs) * (softplus(q.logits) - softplus(p.logits))
    return t1 + t2

@register_kl(Categorical, Categorical)
def _kl_categorical_categorical(p, q):
    t = p.probs * (p.logits - q.logits)
    return t.sum(-1)
```

From PyTorch's source, torch/distributions/kl.py: the Bernoulli and categorical KLs.

PyTorch source: torch/distributions/kl.py

▶ Notebook 2.6

The same formulas in PyTorch: Bernoulli and categorical

```python
@register_kl(Bernoulli, Bernoulli)
def _kl_bernoulli_bernoulli(p, q):
    t1 = p.probs * (softplus(-q.logits) - softplus(-p.logits))
    t2 = (1 - p.probs) * (softplus(q.logits) - softplus(p.logits))
    return t1 + t2

@register_kl(Categorical, Categorical)
def _kl_categorical_categorical(p, q):
    t = p.probs * (p.logits - q.logits)
    return t.sum(-1)
```

softplus(−logit) = −log(probability), so t1 = a log(a/b) and t2 = (1 − a) log((1 − a)/(1 − b)). For a categorical, logits are log-probabilities: $\Sigma\ \pi_k (\log \pi_k - \log \rho_k)$.

> Read them against our formulas: a log(a/b) + (1 − a) log((1 − a)/(1 − b)), and $\Sigma_k\ \pi_k \log(\pi_k/\rho_k)$. The same.

PyTorch source: torch/distributions/kl.py

▶ Notebook 2.6

The same formulas in PyTorch: Gaussian and diagonal

```python
@register_kl(Normal, Normal)
def _kl_normal_normal(p, q):
    var_ratio = (p.scale / q.scale).pow(2)
    t1 = ((p.loc - q.loc) / q.scale).pow(2)
    return 0.5 * (var_ratio + t1 - 1 - var_ratio.log())

@register_kl(Independent, Independent)
def _kl_independent_independent(p, q):
    result = kl_divergence(p.base_dist, q.base_dist)
    return _sum_rightmost(result, p.reinterpreted_batch_ndims)
```

> PyTorch's Gaussian KL, and the rule for independent coordinates.

PyTorch source: torch/distributions/kl.py

▶ Notebook 2.6

The same formulas in PyTorch: Gaussian and diagonal

```python
@register_kl(Normal, Normal)
def _kl_normal_normal(p, q):
    var_ratio = (p.scale / q.scale).pow(2)
    t1 = ((p.loc - q.loc) / q.scale).pow(2)
    return 0.5 * (var_ratio + t1 - 1 - var_ratio.log())

@register_kl(Independent, Independent)
def _kl_independent_independent(p, q):
    result = kl_divergence(p.base_dist, q.base_dist)
    return _sum_rightmost(result, p.reinterpreted_batch_ndims)
```

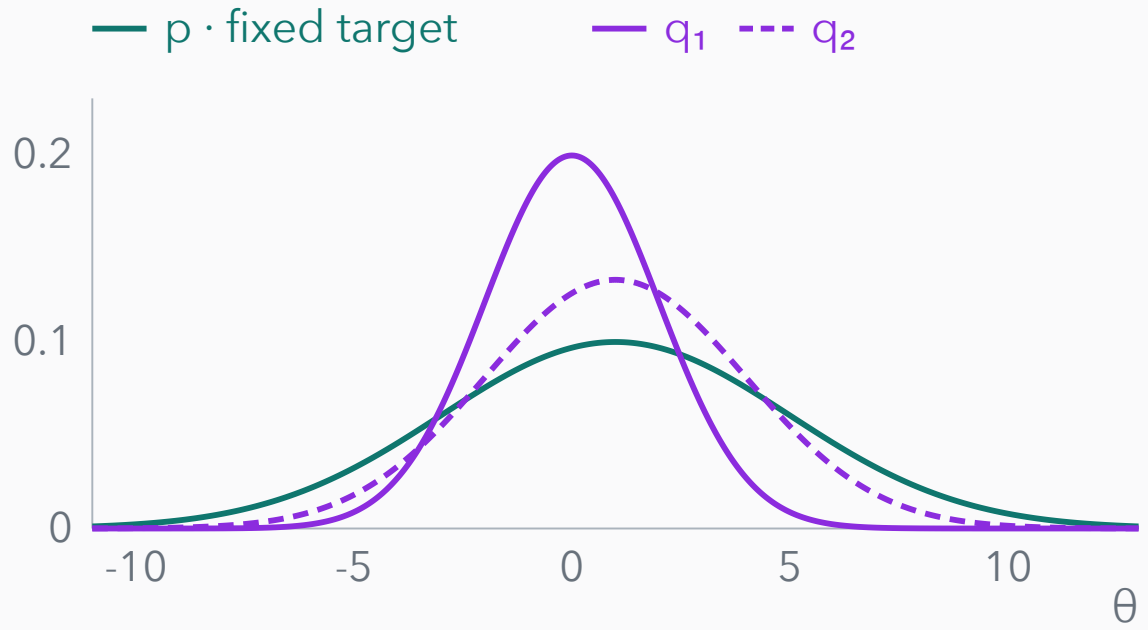
$\frac{1}{2}(\sigma^2/s^2 + (\mu - m)^2/s^2 - 1 - \log \sigma^2/s^2) = \log(s/\sigma) + (\sigma^2 + (\mu - m)^2)/(2s^2) - \frac{1}{2}$. Independent: the one-dimensional KLs, summed over coordinates.

> Expand the Normal line: it is our formula. Independent sums the per-coordinate KLs: the diagonal-Gaussian rule.

PyTorch source: torch/distributions/kl.py

▶ Notebook 2.6

Use the formula: which Gaussian is closer to p?



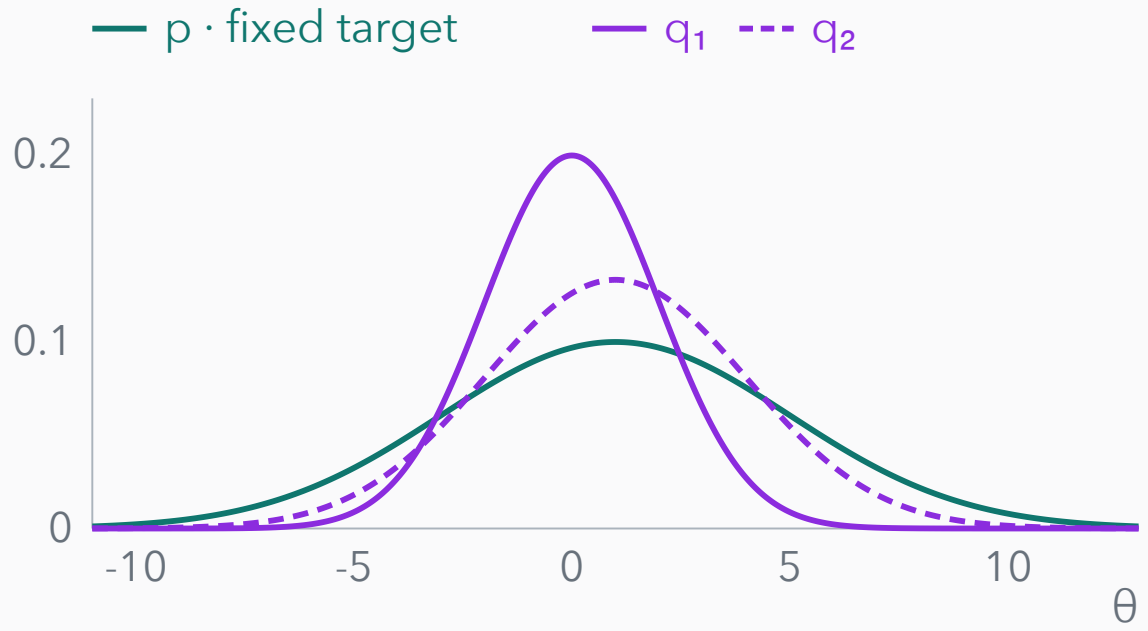
$$p = \mathcal{N}(1, 4^2)$$

	substitute	KL
$q_1 = \mathcal{N}(0, 2^2)$	$\log 2 + \frac{4+1}{32} - \frac{1}{2}$	
$q_2 = \mathcal{N}(1, 3^2)$	$\log \frac{4}{3} + \frac{9+0}{32} - \frac{1}{2}$	

The PML vote: which is closer to p, q_1 or q_2? Substitute into the formula.

▶ Notebook 2.6

Use the formula: which Gaussian is closer to p?



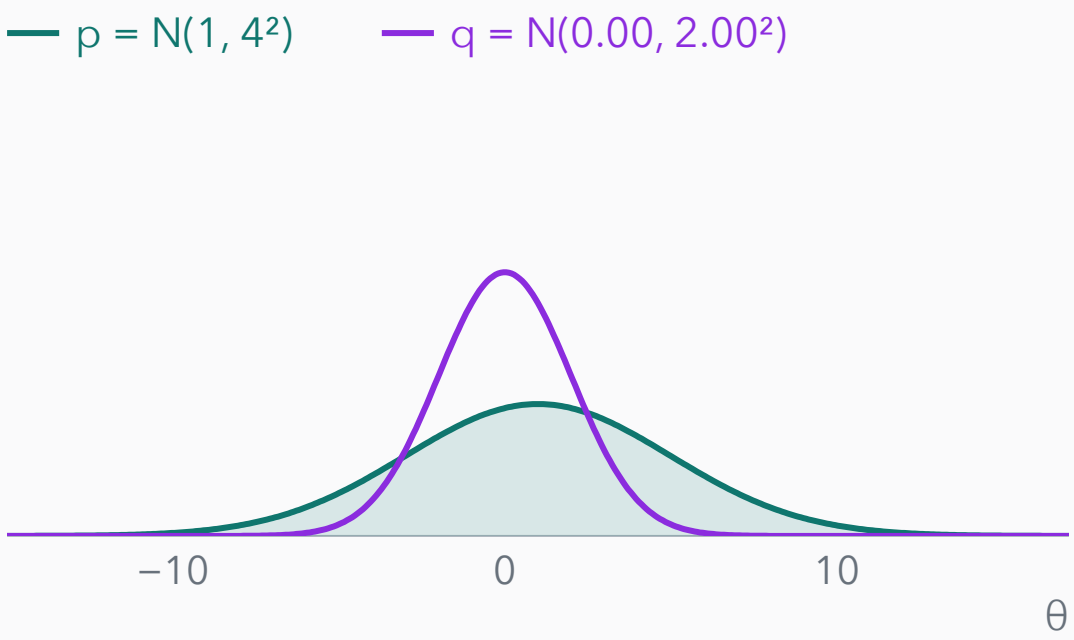
$$p = \mathcal{N}(1, 4^2)$$

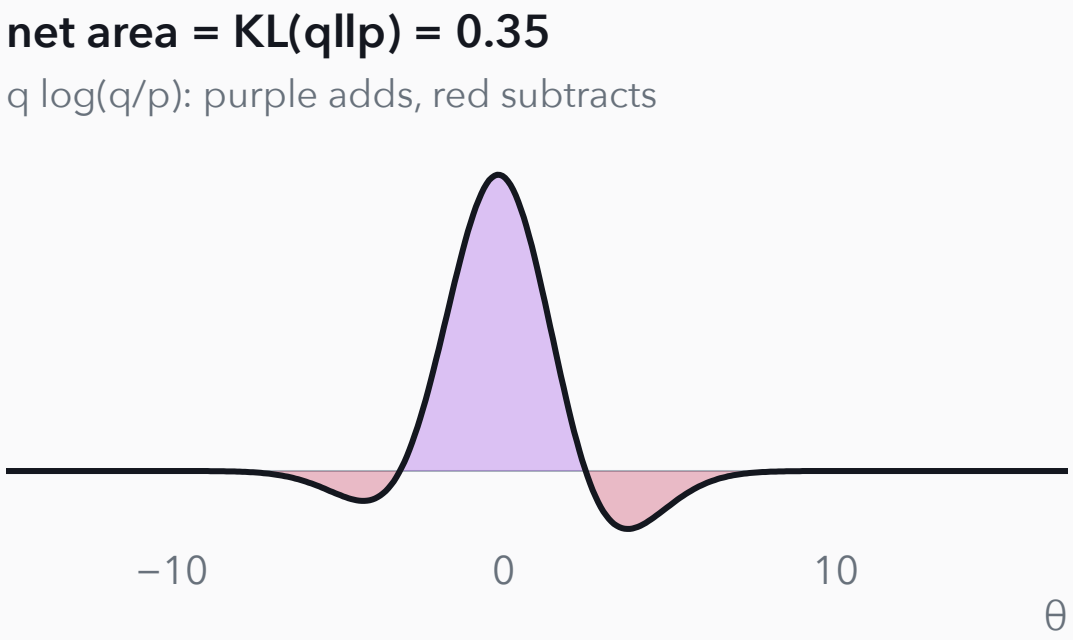
	substitute	KL
$q_1 = \mathcal{N}(0, 2^2)$	$\log 2 + \frac{4+1}{32} - \frac{1}{2}$	0.35
$q_2 = \mathcal{N}(1, 3^2)$	$\log \frac{4}{3} + \frac{9+0}{32} - \frac{1}{2}$	0.07

KL(q_1‖p) = 0.35 nats and KL(q_2‖p) = 0.07. The formula settles the vote.

▶ Notebook 2.6

KL is an area you can watch





Purple area adds and red area subtracts; the net area is the KL. Drag q, or flip the direction.

▶ Notebook 2.7

KL is an average: an expectation under p

$$\mathrm{KL}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Look again at the formula: each log ratio is weighted by p(x), and the weights add up to 1.

▶ Notebook 2.7

KL is an average: an expectation under p

$$\mathrm{KL}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$= \mathbb{E}_{x \sim p}\left[\log \frac{p(x)}{q(x)} \right]$$

E with x ~ p means: the average value when x is drawn from p.

So KL is an expectation: the average log ratio when x is drawn from p.

▶ Notebook 2.7

Two ways to compute an expectation

$$\mathbb{E}_{x\sim p}[f(x)] = \sum_{x} p(x)\, f(x)$$

For a discrete x, an expectation is a weighted sum: each value f(x) times its probability.

▶ Notebook 2.7

Two ways to compute an expectation

$$\mathbb{E}_{x\sim p}[f(x)] = \sum_{x} p(x)\, f(x)$$

$$\mathbb{E}_{\theta\sim p}[f(\theta)] = \int p(\theta)\, f(\theta)\, d\theta$$

For a continuous θ, the sum becomes an integral.

▶ Notebook 2.7

Two ways to compute an expectation

$$\mathbb{E}_{x\sim p}[f(x)] = \sum_{x} p(x)\, f(x)$$

$$\mathbb{E}_{\theta\sim p}[f(\theta)] = \int p(\theta)\, f(\theta)\, d\theta$$

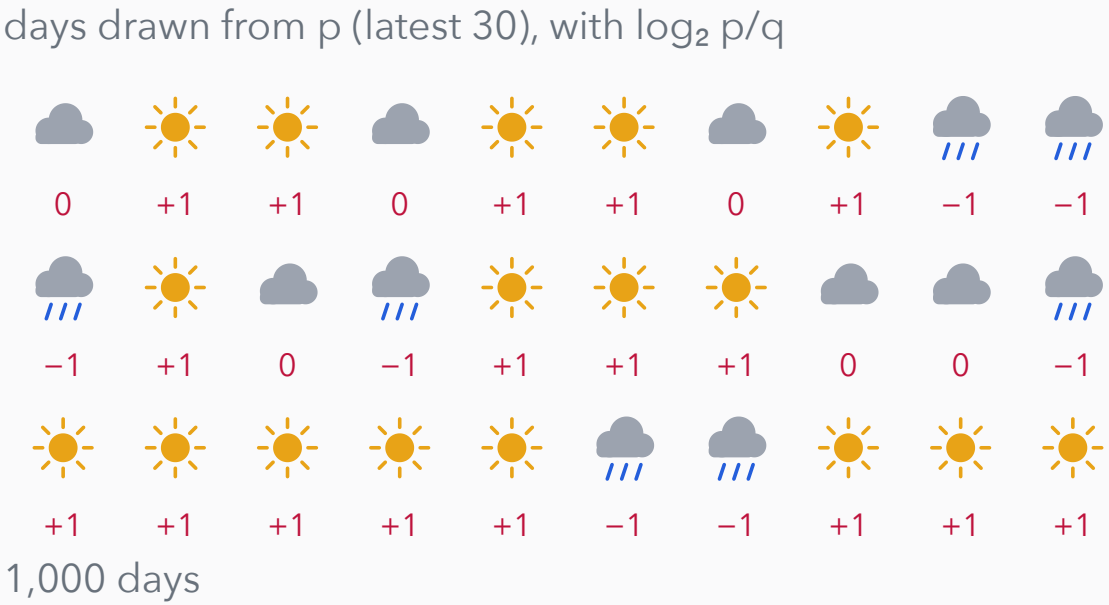
$$\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{S}\sum_{s=1}^{S} f(x_s), \qquad x_s \sim p$$

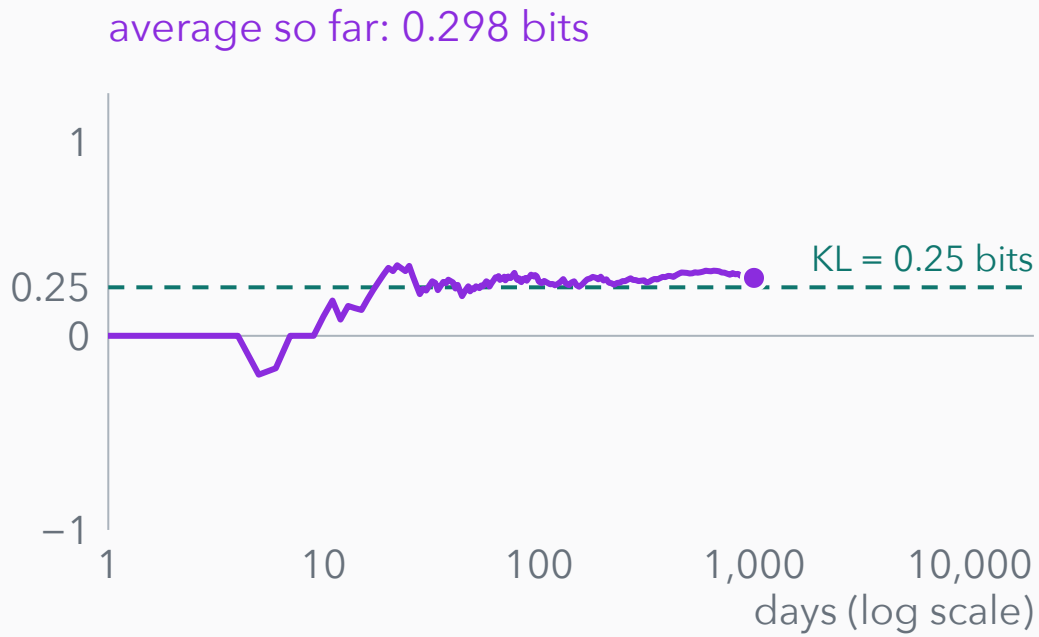
Draws land on each x in proportion to p(x), so a plain average over draws weights each f(x) by p(x) automatically.

> Or draw samples from p and take a plain average. Values that are more probable show up more often, so the weighting happens by itself.

▶ Notebook 2.7

Sample days and watch the average find the KL





Draw days from our weather p and average $\log_2$ p/q: +1 bit on sunny days, 0 on cloudy, −1 on rainy. The average settles on the KL, 0.25 bits, with no formula needed.

▶ Notebook 2.7

No formula needed: estimate KL by sampling

$$\mathrm{KL}(q\|p) = \mathbb{E}_{\theta\sim q}\big[\log q(\theta) - \log p(\theta)\big] \approx \frac{1}{S}\sum_{i=1}^{S}\big[\log q(\theta_i) - \log p(\theta_i)\big]$$

```
theta = q.sample((S,))
log_q = q.log_prob(theta)
log_p = p.log_prob(theta)
est = (log_q - log_p).mean()
exact = kl_divergence(q, p)
```

> Draw θ from q, score each draw with both log densities, and average. No formula needed.

▶ Notebook 2.7

No formula needed: estimate KL by sampling

$$\mathrm{KL}(q\|p) = \mathbb{E}_{\theta\sim q}\big[\log q(\theta) - \log p(\theta)\big] \approx \frac{1}{S}\sum_{i=1}^{S}\big[\log q(\theta_i) - \log p(\theta_i)\big]$$

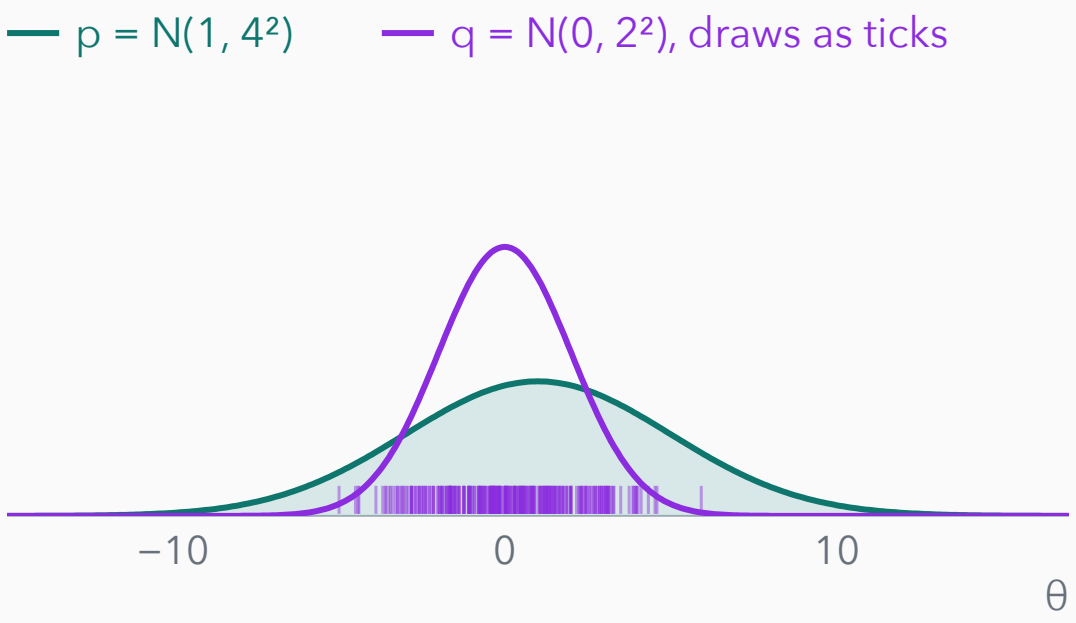
```
theta = q.sample((S,))
log_q = q.log_prob(theta)
log_p = p.log_prob(theta)
est = (log_q - log_p).mean()
exact = kl_divergence(q, p)
```

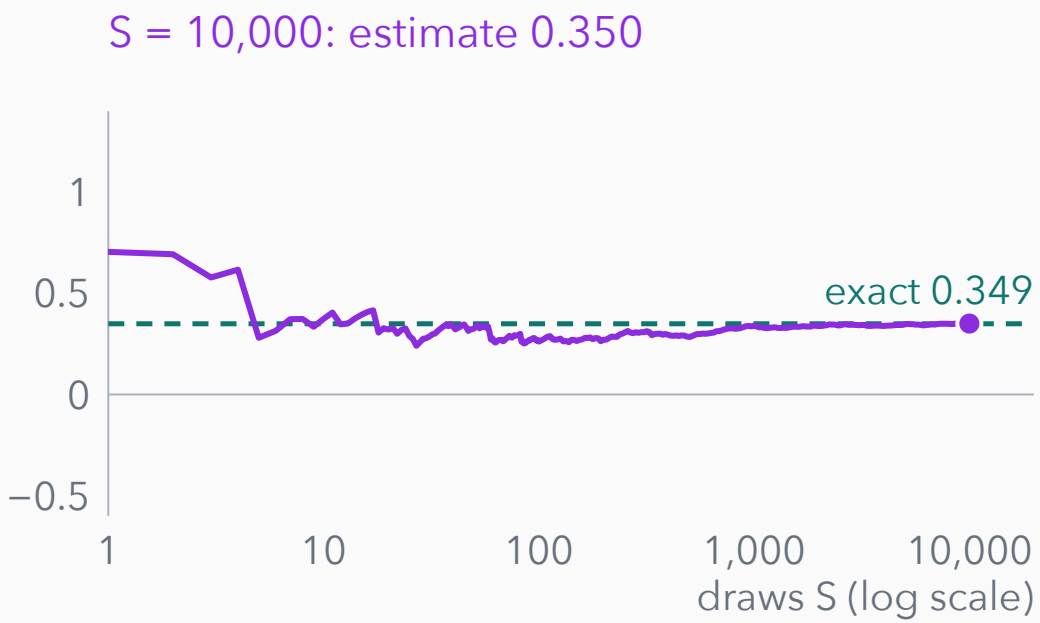
	KL(q₂‖p)
exact formula	0.069
estimate, S = 10,000	0.067

> Checked on a pair that has a formula: 0.067 from 10,000 draws, against the exact 0.069.

▶ Notebook 2.7

Watch the estimate settle



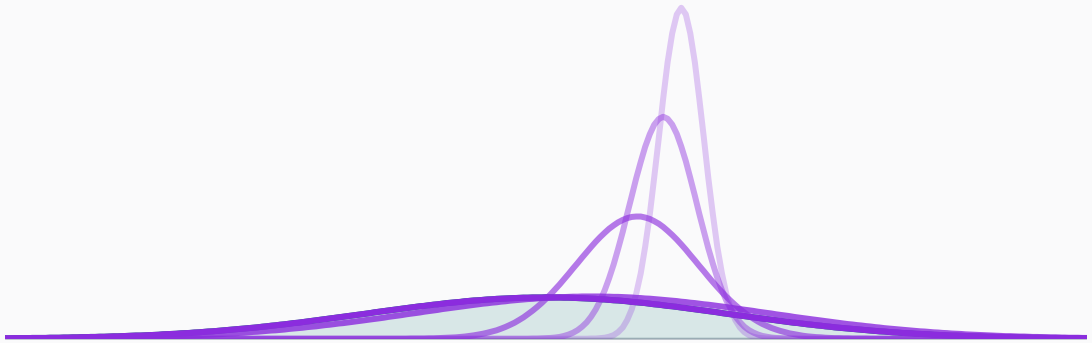


Add draws from q and watch the running average of log q − log p settle on the exact 0.349.

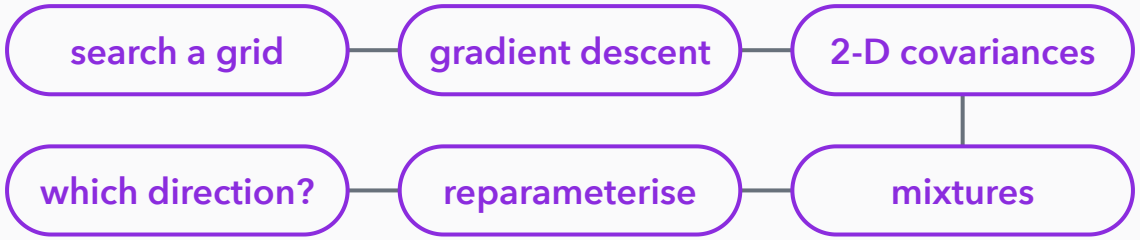
▶ Notebook

Part 3 · Find the closest q

q slides onto p, one gradient step at a time



the road through Part 3



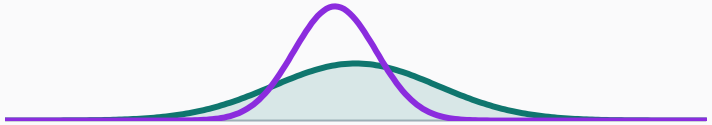
3.1 Search a grid ▶
3.2 Gradient descent ▶
3.3 Covariance in 2-D ▶
3.4 Mixtures: no formula ▶
3.5 The reparameterisation trick ▶
3.6 Which direction of KL? ▶

> Part 3 finds the closest q by gradient descent on KL, with a p we can evaluate: a grid, gradient descent, covariances, mixtures, the reparameterisation trick, and which direction of KL.

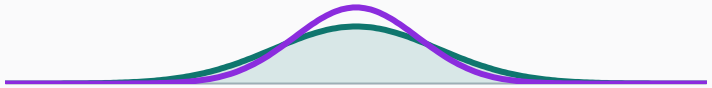
▶ Notebook 3.1

Six candidates: which q is closest to p?

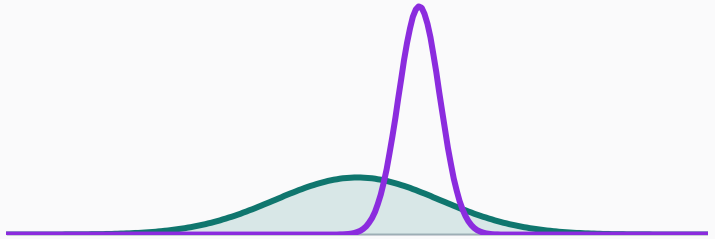
q = N(0, 2²)



q = N(1, 3²)



q = N(4, 1²)



q = N(−3, 4²)



q = N(1, 8²)



q = N(2, 5²)

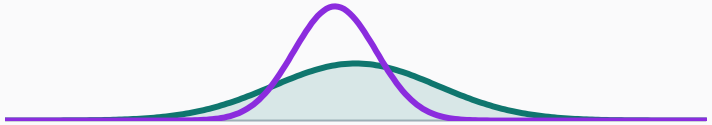


Six Gaussians against the same target p = N(1, 4²). Vote: which is closest?

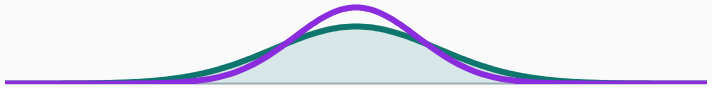
▶ Notebook 3.1

Six candidates: which q is closest to p?

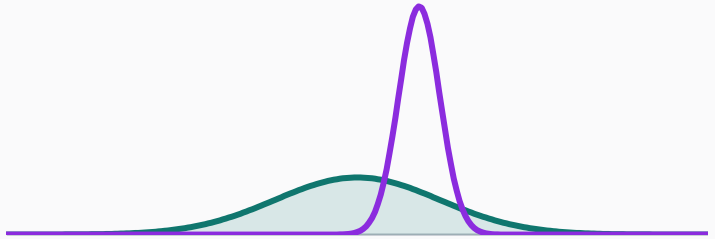
q = N(0, 2²) KL 0.35



q = N(1, 3²) **KL 0.07**



q = N(4, 1²) KL 1.20



q = N(−3, 4²) KL 0.50



q = N(1, 8²) KL 0.81



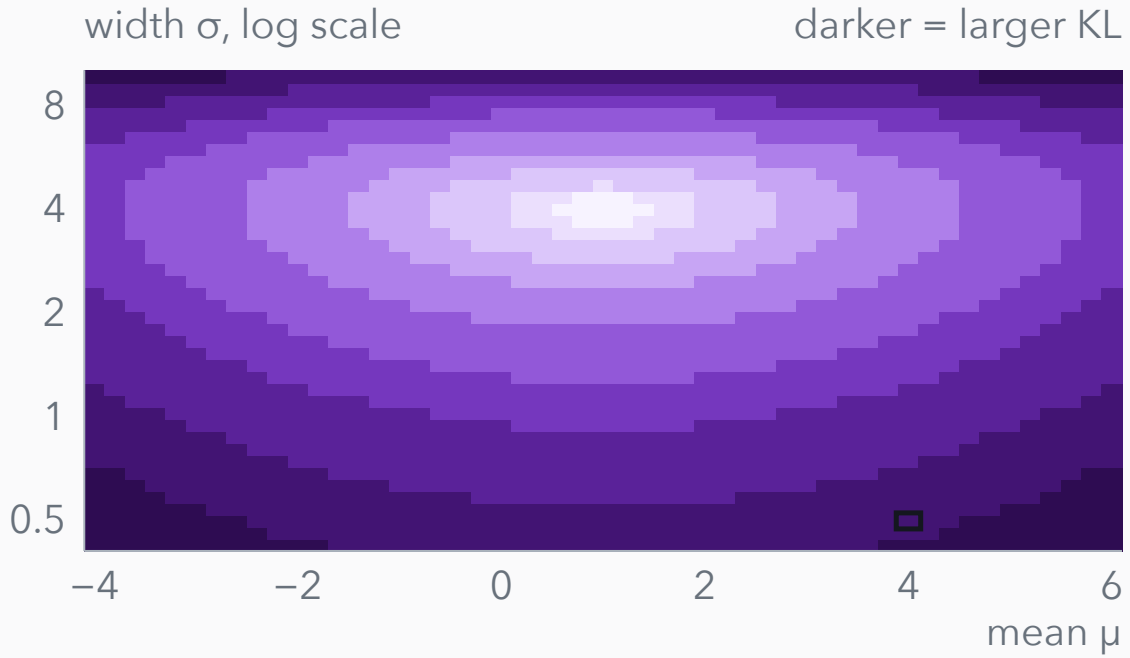
q = N(2, 5²) KL 0.09

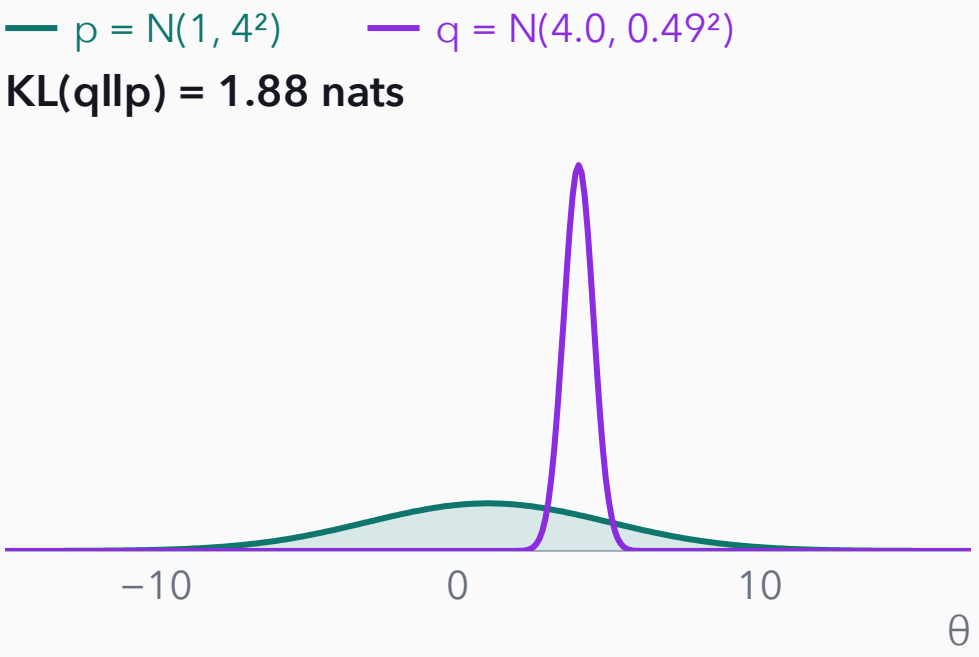


Smallest KL: N(1, 3²) at 0.07, then N(2, 5²) at 0.09. Could another q do better?

▶ Notebook 3.1

Search a grid of means and widths: hover over it

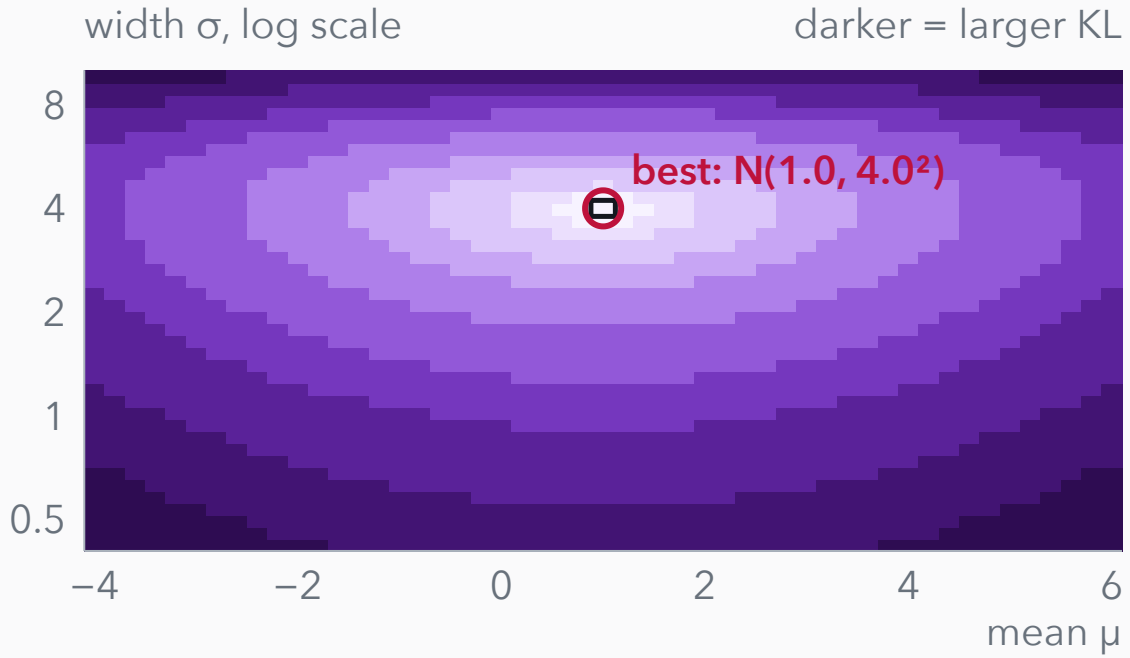


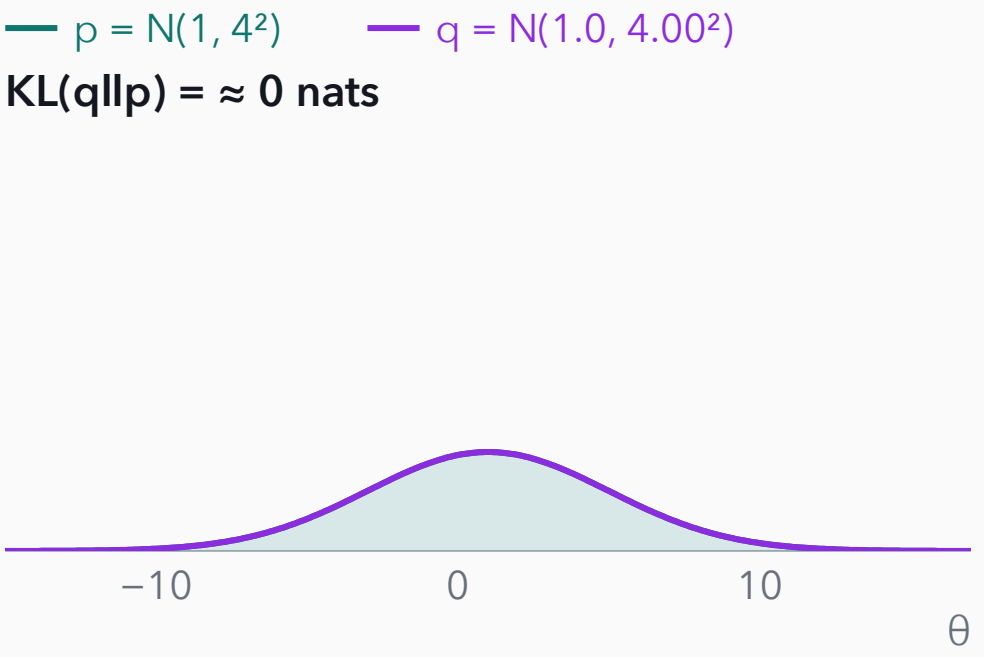


Every point of the grid is one q = N(μ, σ²), coloured by its KL to p. Hover over it: the right panel shows that q against p, and its score.

▶ Notebook 3.1

Search a grid of means and widths: hover over it

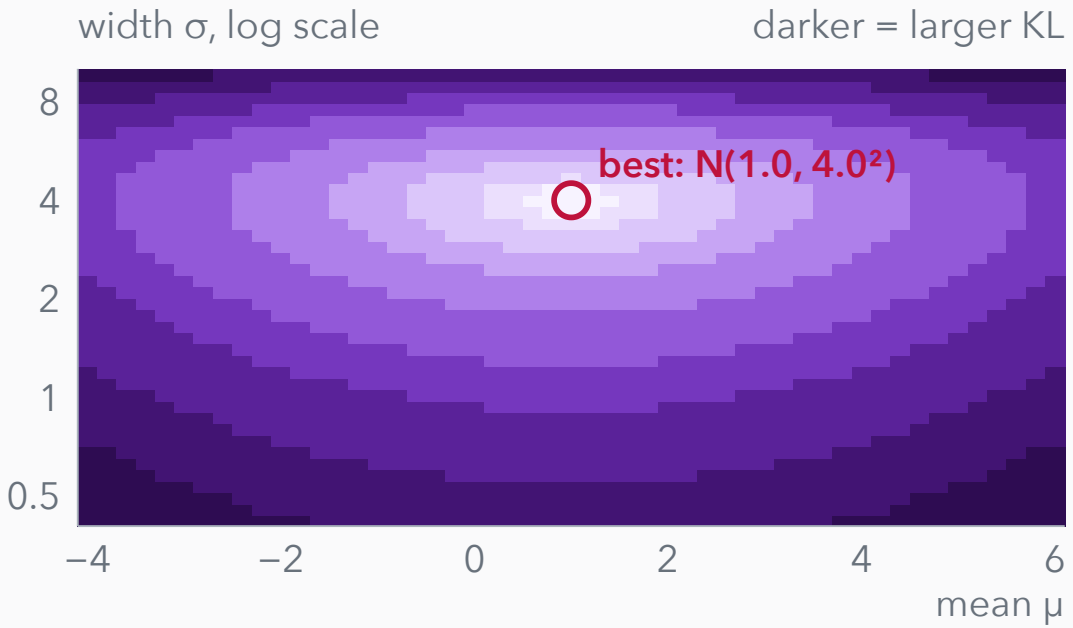




The lightest point is q = p: 2,040 evaluations. For our network's fourteen numbers a grid needs $40^{14} \approx 3 \times 10^{22}$. We need to walk downhill instead.

▶ Notebook 3.2

Instead of a grid, walk downhill



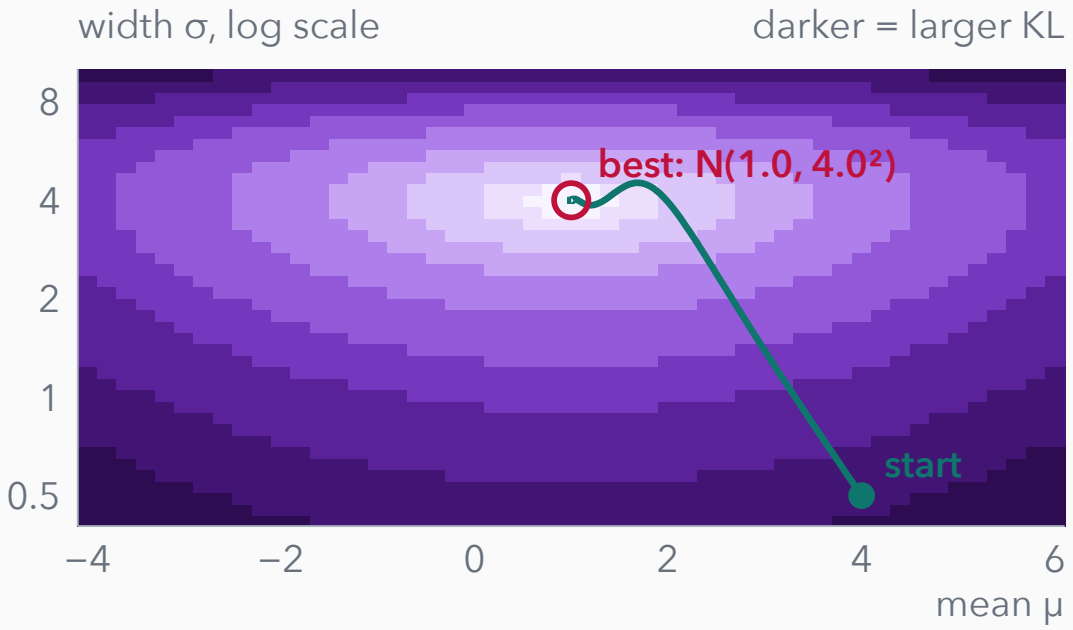
$$\psi^* = \arg\min_{\psi} \mathrm{KL}(q_\psi \,\|\, p)$$

$$\psi \leftarrow \psi - \eta \, \nabla_\psi \mathrm{KL}(q_\psi \,\|\, p)$$

Cast the search as optimisation: follow the gradient of KL with respect to q's parameters.

▶ Notebook 3.2

Instead of a grid, walk downhill



$$\psi^* = \arg\min_{\psi} \mathrm{KL}(q_\psi \,\|\, p)$$

$$\psi \leftarrow \psi - \eta \, \nabla_\psi \mathrm{KL}(q_\psi \,\|\, p)$$

Gradient descent from N(4, 0.5²): the teal path. About a hundred steps, and the same recipe works for millions of parameters.

From N(4, 0.5²) the optimiser walks downhill to p. It needs only gradients, which autograd provides.

▶ Notebook 3.2

The same search in code

```
mu  = Parameter(tensor(4.0))
rho = Parameter(tensor(log(0.5)))
opt = Adam([mu, rho], lr=0.05)
for step in range(200):
    q = Normal(mu, rho.exp())
    loss = kl_divergence(q, p)
    opt.zero_grad()
    loss.backward()
    opt.step()
```

$$\sigma = \exp(\rho) > 0 \text{ for every } \rho$$

Store ρ = log σ, so that σ = exp(ρ) stays positive whatever an update does.

▶ Notebook 3.2

The same search in code

```
mu  = Parameter(tensor(4.0))
rho = Parameter(tensor(log(0.5)))
opt = Adam([mu, rho], lr=0.05)
for step in range(200):
    q = Normal(mu, rho.exp())
    loss = kl_divergence(q, p)
    opt.zero_grad()
    loss.backward()
    opt.step()
```

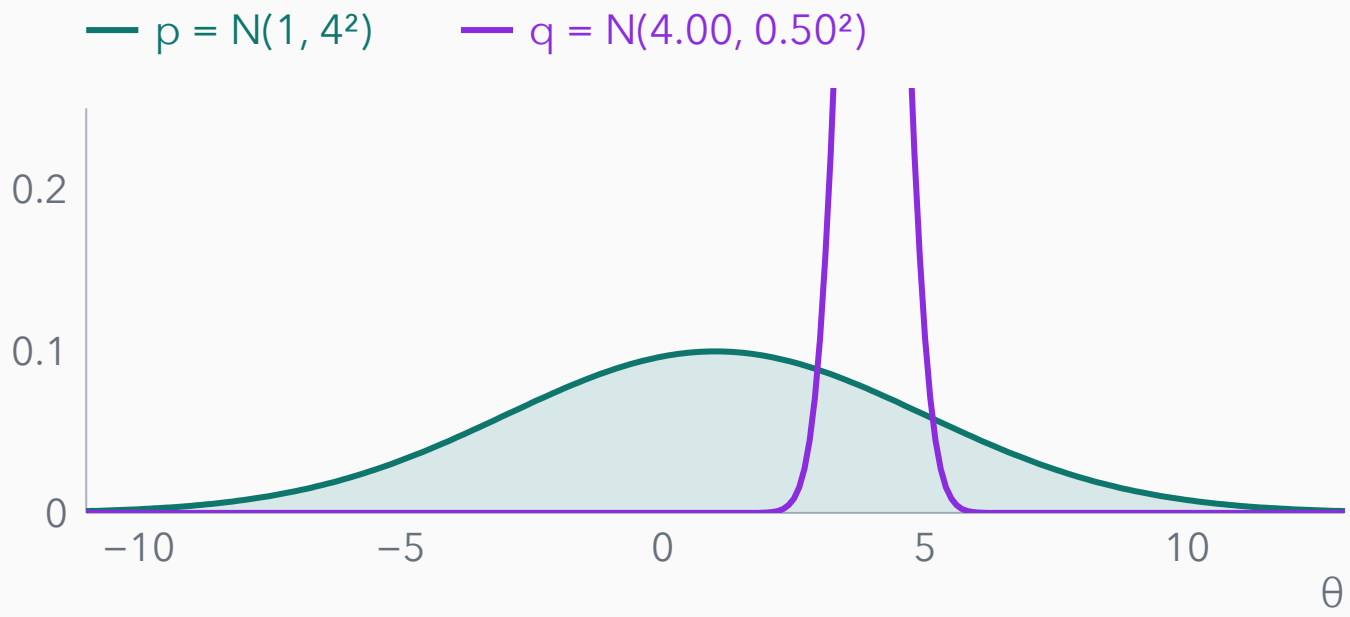
$$\sigma = \exp(\rho) > 0 \ \text{ for every } \rho$$

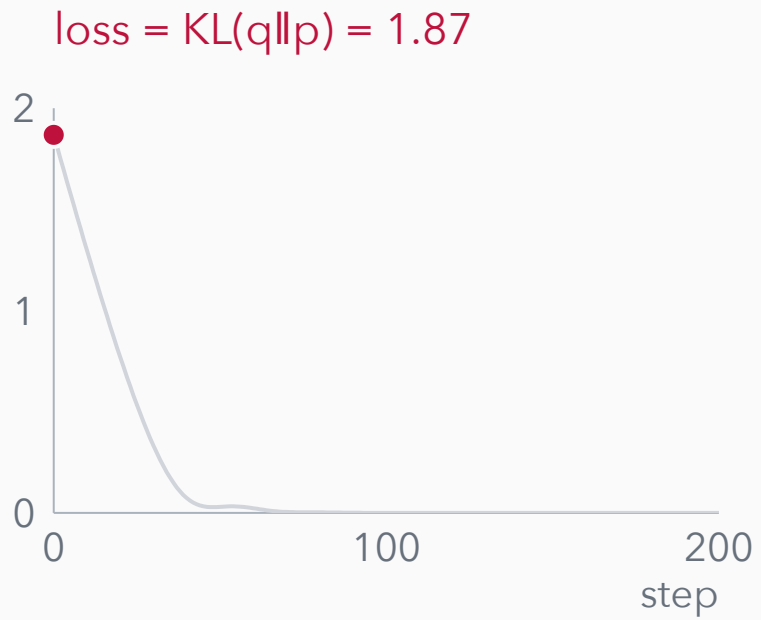
Rebuild q, score it, differentiate, update. The notebook runs exactly this loop and plots the loss, μ, σ and the learned curve.

> Rebuild q, score it, differentiate, update. The same five lines fit our network in Part 5.

▶ Notebook 3.2

Press play: the loss falls as q slides onto p

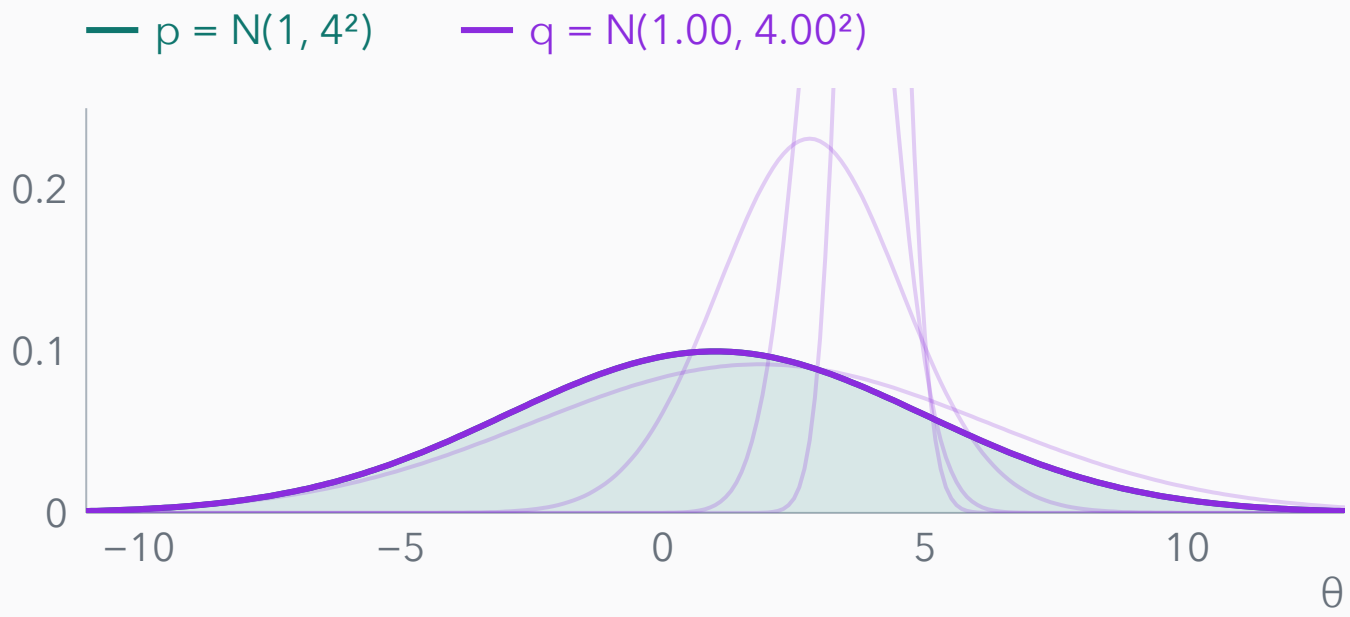


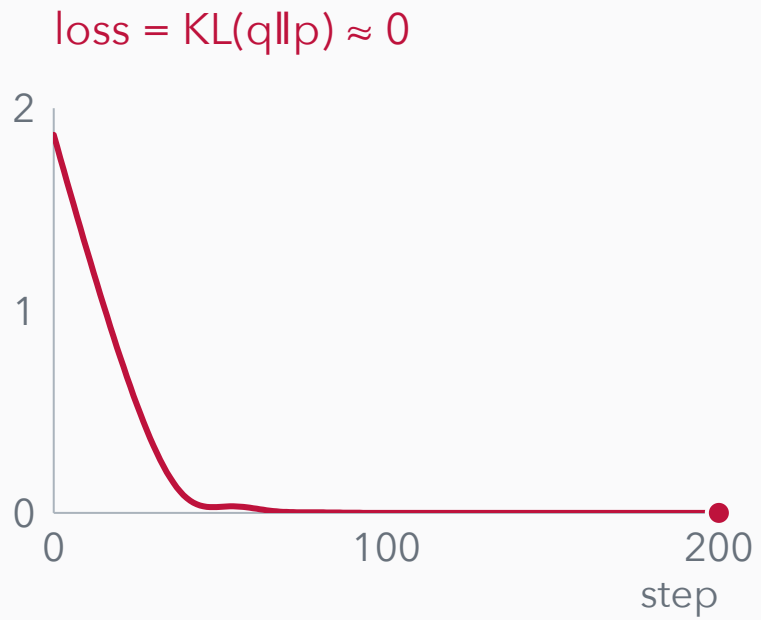


Start q far away, at N(4, 0.5²). Predict: which reaches the target first, the centre or the width?

▶ Notebook 3.2

Press play: the loss falls as q slides onto p



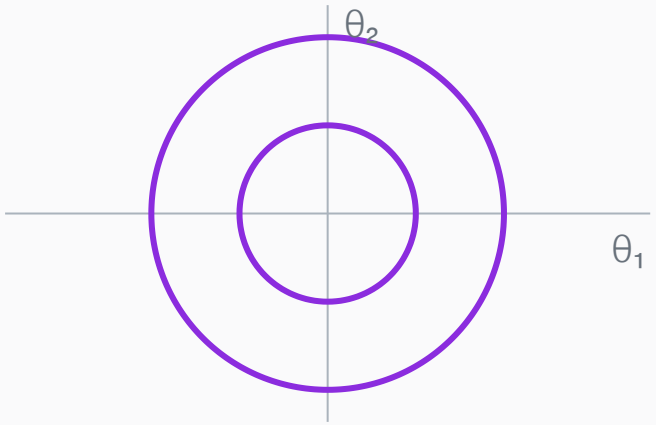


> The width gets there first and overshoots a little; the centre slides in by step 100. Then the loss, KL(q‖p), is about 0. Drag the slider to replay any step.

▶ Notebook 3.3

In two dimensions, q needs a covariance matrix: three families

isotropic: $s^2 I$



$$\Sigma = \begin{bmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix}$$

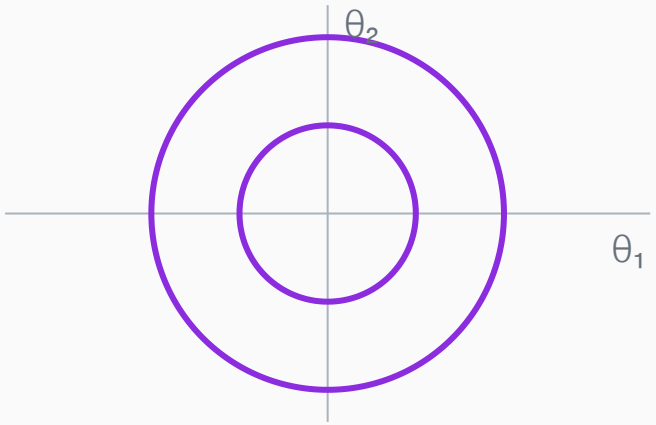
1 number, whatever the dimension

Isotropic: one width for every direction, so the contours are circles. $\Sigma = s^2 I$ is a single number.

▶ Notebook 3.3

In two dimensions, q needs a covariance matrix: three families

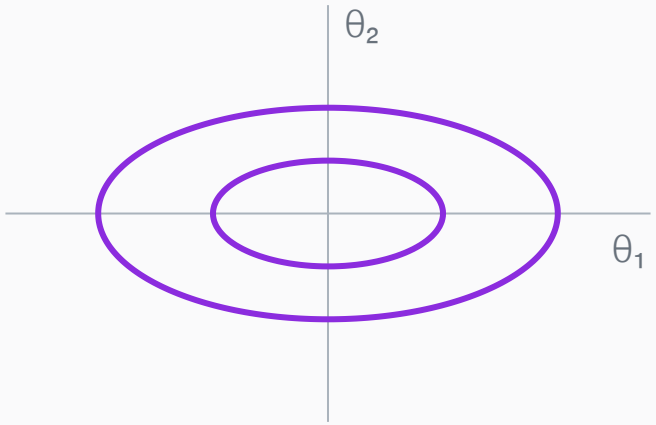
isotropic: $s^2 I$



$$\Sigma = \begin{bmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix}$$

1 number, whatever the dimension

diagonal: diag(s_1^2, s_2^2)



$$\Sigma = \begin{bmatrix} 1.70 & 0.00 \\ 0.00 & 0.36 \end{bmatrix}$$

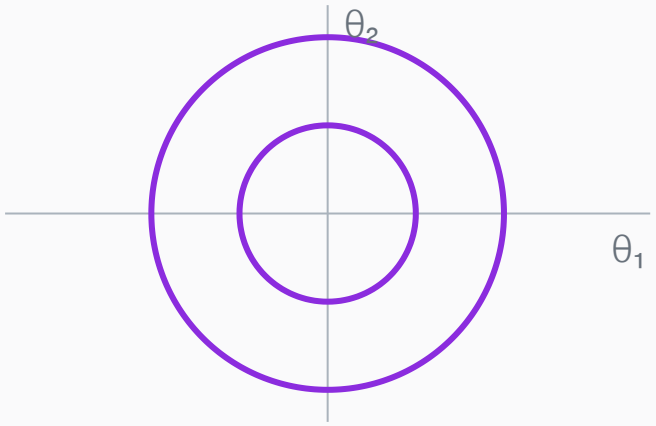
2 numbers; d in d dimensions

> Diagonal: its own width per coordinate but no correlation, so the ellipses are axis-aligned. This is mean-field: $q(\theta) = q(\theta_1)\, q(\theta_2)$.

▶ Notebook 3.3

In two dimensions, q needs a covariance matrix: three families

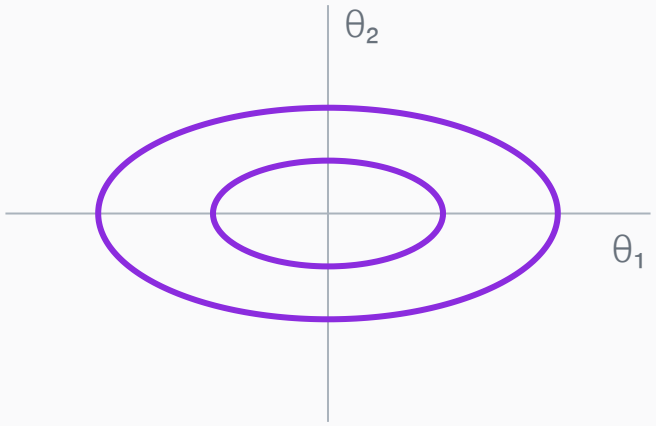
isotropic: $s^2 I$



$$\Sigma = \begin{bmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix}$$

1 number, whatever the dimension

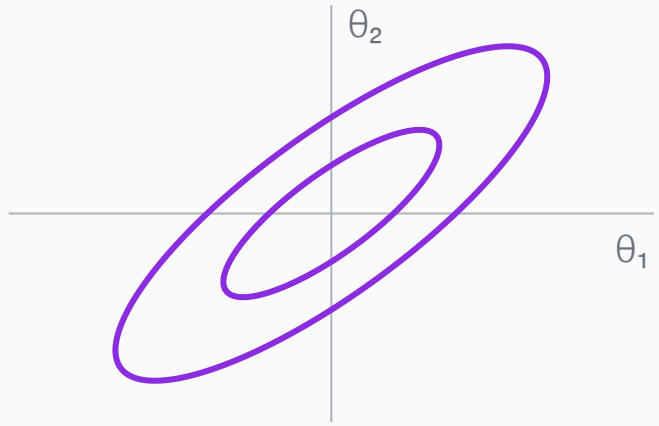
diagonal: diag(s_1^2, s_2^2)



$$\Sigma = \begin{bmatrix} 1.70 & 0.00 \\ 0.00 & 0.36 \end{bmatrix}$$

2 numbers; d in d dimensions

full: LL^T



$$\Sigma = \begin{bmatrix} 1.50 & 0.95 \\ 0.95 & 0.90 \end{bmatrix}$$

3 numbers; d(d + 1)/2 in general

Our network has d = 7 parameters: 1, 7 or 28 numbers. A network with a million weights: 1, a million, or 5×10^{11}.

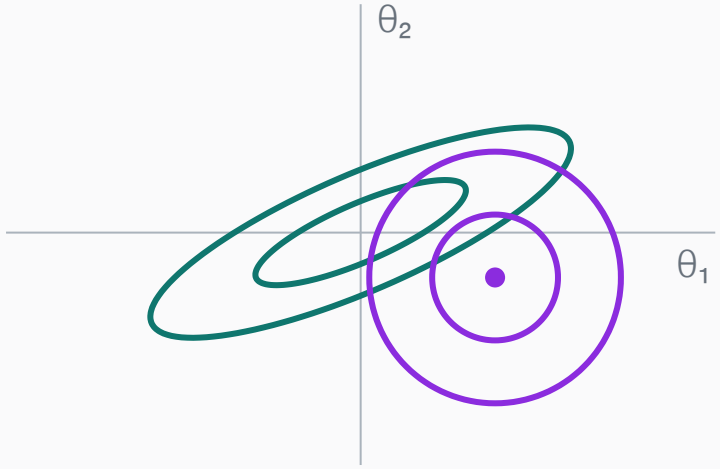
> Full: the ellipse can tilt. Writing $\Sigma = LL^T$ with L lower-triangular keeps Σ a valid covariance for any L. It costs d(d + 1)/2 numbers.

▶ Notebook 3.3

Fit all three to the same tilted target

isotropic KL 7.01

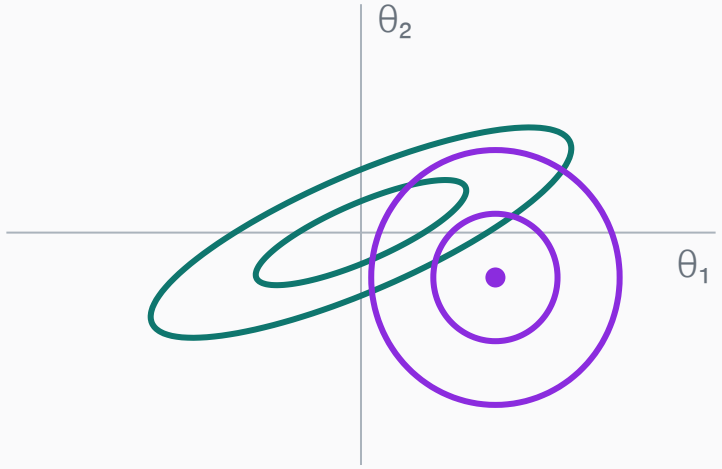
$s^2 I$ · 1 number



$$\Sigma_q = \begin{bmatrix} 0.80 & 0.00 \\ 0.00 & 0.80 \end{bmatrix}$$

diagonal KL 7.05

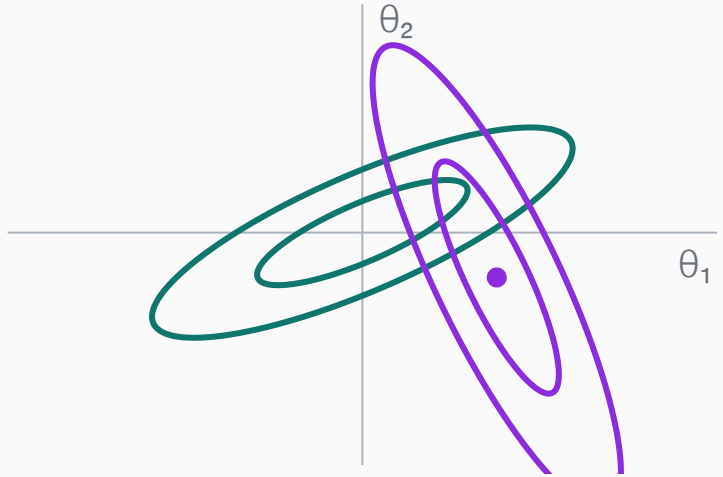
$\mathrm{diag}(s_1^2, s_2^2)$ · 2 numbers



$$\Sigma_q = \begin{bmatrix} 0.78 & 0.00 \\ 0.00 & 0.82 \end{bmatrix}$$

full KL 14

LL^T · 3 numbers



$$\Sigma_q = \begin{bmatrix} 0.78 & -1.23 \\ -1.23 & 2.74 \end{bmatrix}$$

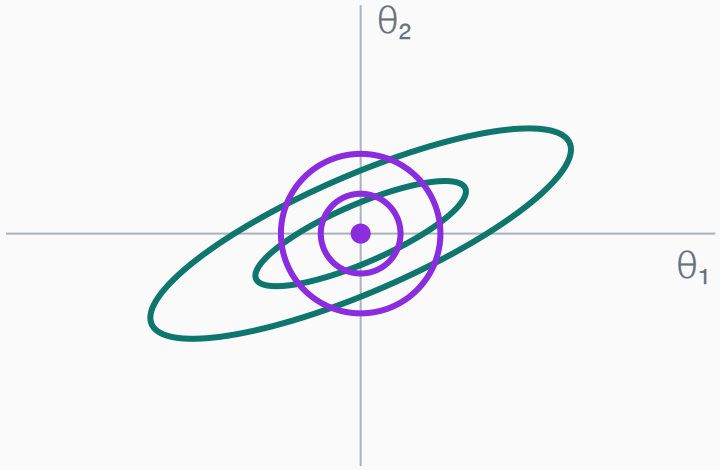
The target p (teal) has widths 1.5 and 0.75 and correlation 0.8. All three q start from the same random Gaussian, tilted the wrong way. Predict where each ends.

▶ Notebook 3.3

Fit all three to the same tilted target

isotropic KL 0.73

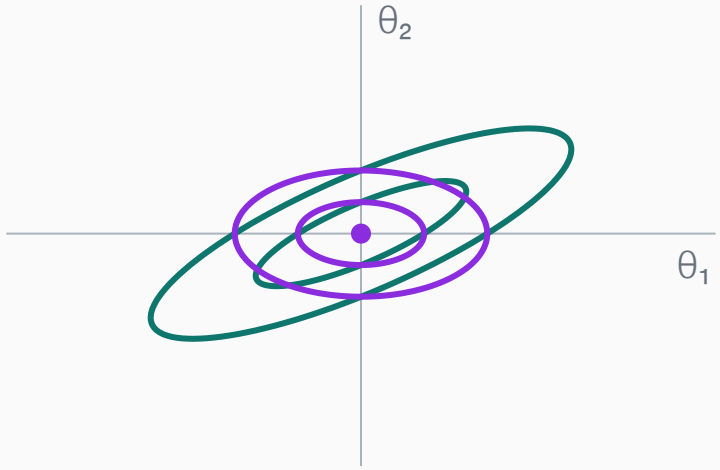
$s^2 I$ · 1 number



$$\Sigma_q = \begin{bmatrix} 0.32 & 0.00 \\ 0.00 & 0.32 \end{bmatrix}$$

diagonal KL 0.51

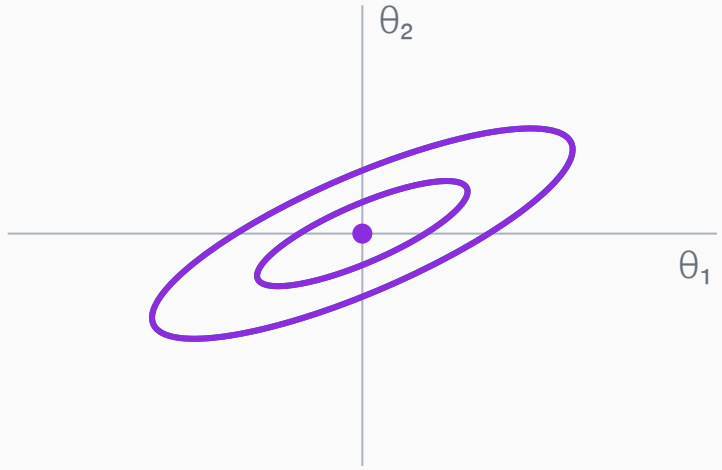
$\mathrm{diag}(s_1^2, s_2^2)$ · 2 numbers



$$\Sigma_q = \begin{bmatrix} 0.81 & 0.00 \\ 0.00 & 0.20 \end{bmatrix}$$

full KL ≈ 0

LL^T · 3 numbers



$$\Sigma_q = \begin{bmatrix} 2.25 & 0.90 \\ 0.90 & 0.56 \end{bmatrix}$$

Press play. The full q matches p exactly: KL → 0. The diagonal q cannot tilt; it shrinks to widths 0.90 and 0.45. The isotropic q shrinks to a circle of radius 0.57.

▶ Notebook 3.3

A q that cannot tilt shrinks instead

q family	widths it finds	KL(q‖p)
target p	1.50 and 0.75, tilted	—
isotropic	0.57 and 0.57	0.73
diagonal (mean-field)	0.90 and 0.45	0.51
full	1.50 and 0.75, tilted	0

> The full q is exact. The diagonal q is narrower than p in both coordinates; the isotropic q is narrower still in the wide direction.

▶ Notebook 3.3

A q that cannot tilt shrinks instead

q family	widths it finds	KL(q‖p)
target p	1.50 and 0.75, tilted	–
isotropic	0.57 and 0.57	0.73
diagonal (mean-field)	0.90 and 0.45	0.51
full	1.50 and 0.75, tilted	0

KL(q‖p) charges heavily for q-mass where p is small, off the tilted ridge. So a q that cannot tilt shrinks: it underestimates the uncertainty. Our network's q is diagonal; Part 5 shows what that does to its error bars.

> Rather than spill mass off the ridge, a q that cannot tilt shrinks. Mean-field VI underestimates uncertainty.

▶ Notebook 3.4

A mixture of Gaussians: first pick a bump, then draw from it

step 1: pick a bump

bump 1
probability ½

bump 2
probability ½

z

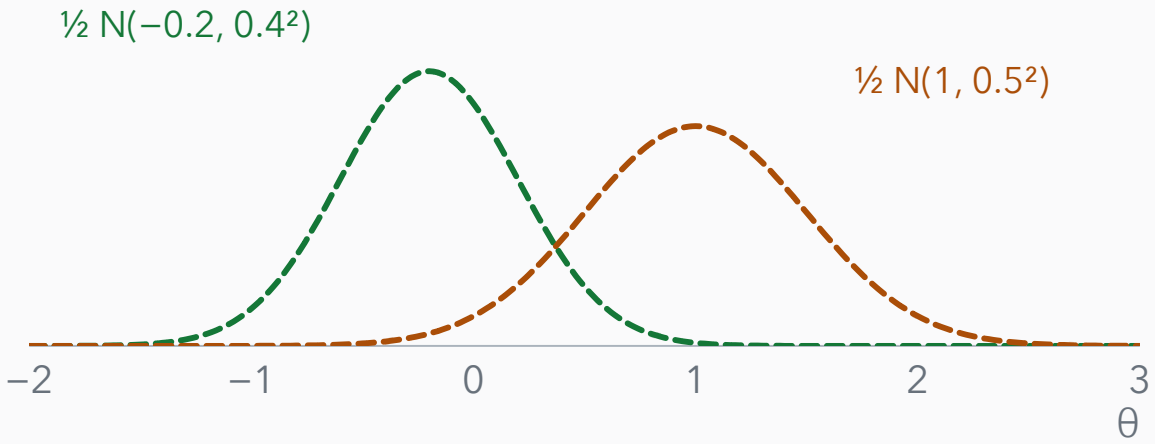
z ~ Categorical(½, ½)

step 2: draw θ from bump z

bump 1: N(−0.2, 0.4²)
bump 2: N(1.0, 0.5²)

θ

θ | z ~ N(μ_z, σ_z^2)



We see θ. We never see which bump made it.

> A mixture of Gaussians makes θ in two steps. First pick a bump, like tossing a fair coin: z = 1 or 2. Then draw θ from that bump's Gaussian.

▶ Notebook 3.4

A mixture of Gaussians: first pick a bump, then draw from it

step 1: pick a bump

bump 1
probability ½

bump 2
probability ½

z

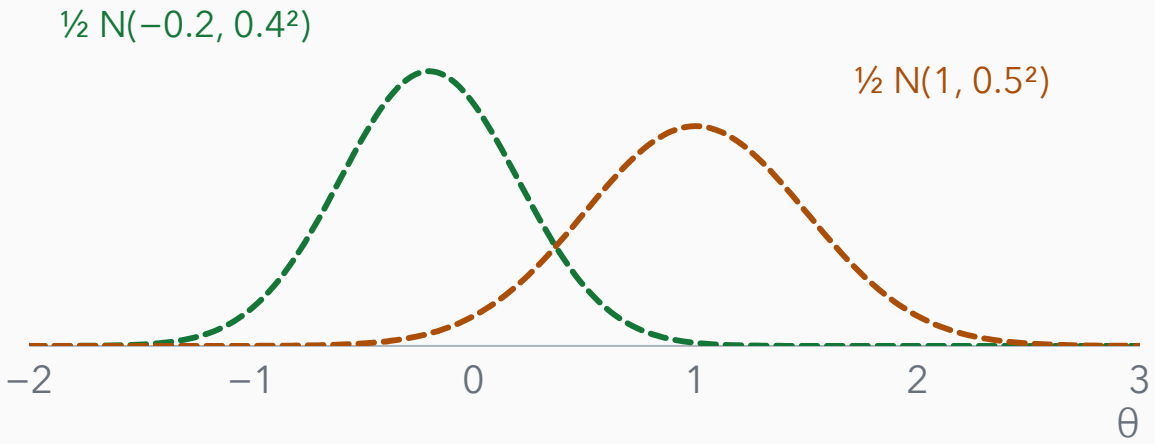
z ~ Categorical(½, ½)

step 2: draw θ from bump z

bump 1: N(−0.2, 0.4²)
bump 2: N(1.0, 0.5²)

θ

θ | z ~ N(μ_z, σ_z^2)



We see θ. We never see which bump made it.

> Each bump contributes its own bell, scaled by how often it is picked: ½ N(−0.2, 0.4²) and ½ N(1, 0.5²).

▶ Notebook 3.4

A mixture of Gaussians: first pick a bump, then draw from it

step 1: pick a bump

bump 1
probability ½

bump 2
probability ½

z

z ~ Categorical(½, ½)

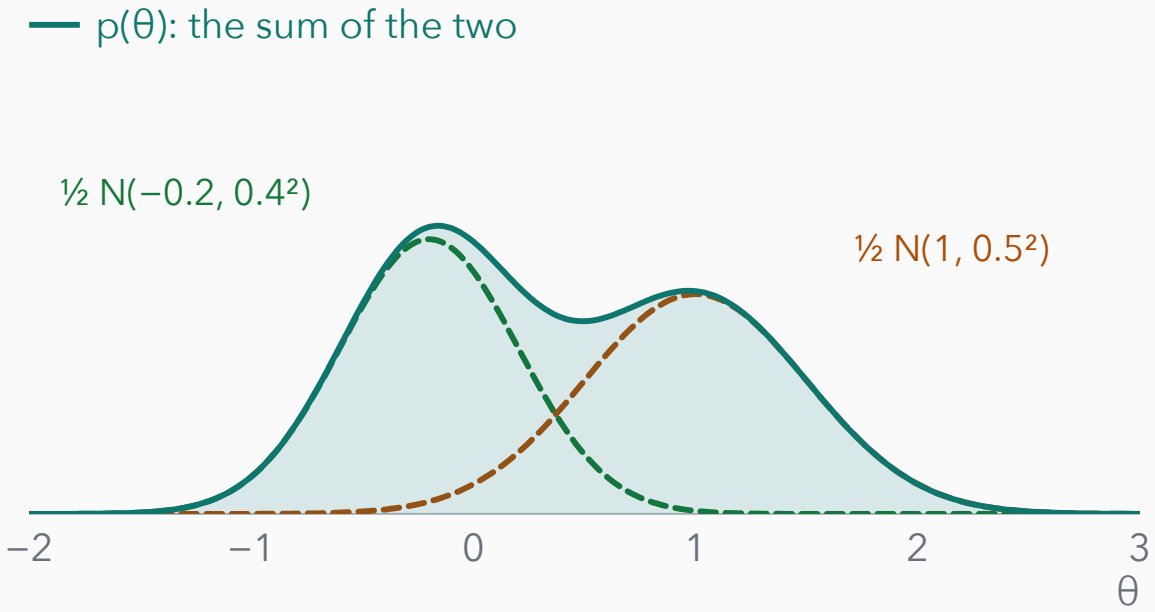
step 2: draw θ from bump z

bump 1: N(−0.2, 0.4²)
bump 2: N(1.0, 0.5²)

θ

θ | z ~ N(μ_z, σ_z^2)

We see θ. We never see which bump made it.



$$p(\theta) = \sum_k \pi_k \mathcal{N}(\theta; \mu_k, \sigma_k^2)$$

We see θ but not the bump that made it, so the density adds both bells: $p(\theta) = \Sigma_k \pi_k N(\theta; \mu_k, \sigma_k^2)$. Two bumps, one distribution.

▶ Notebook 3.4

Draw from the mixture yourself

step 1: pick a bump

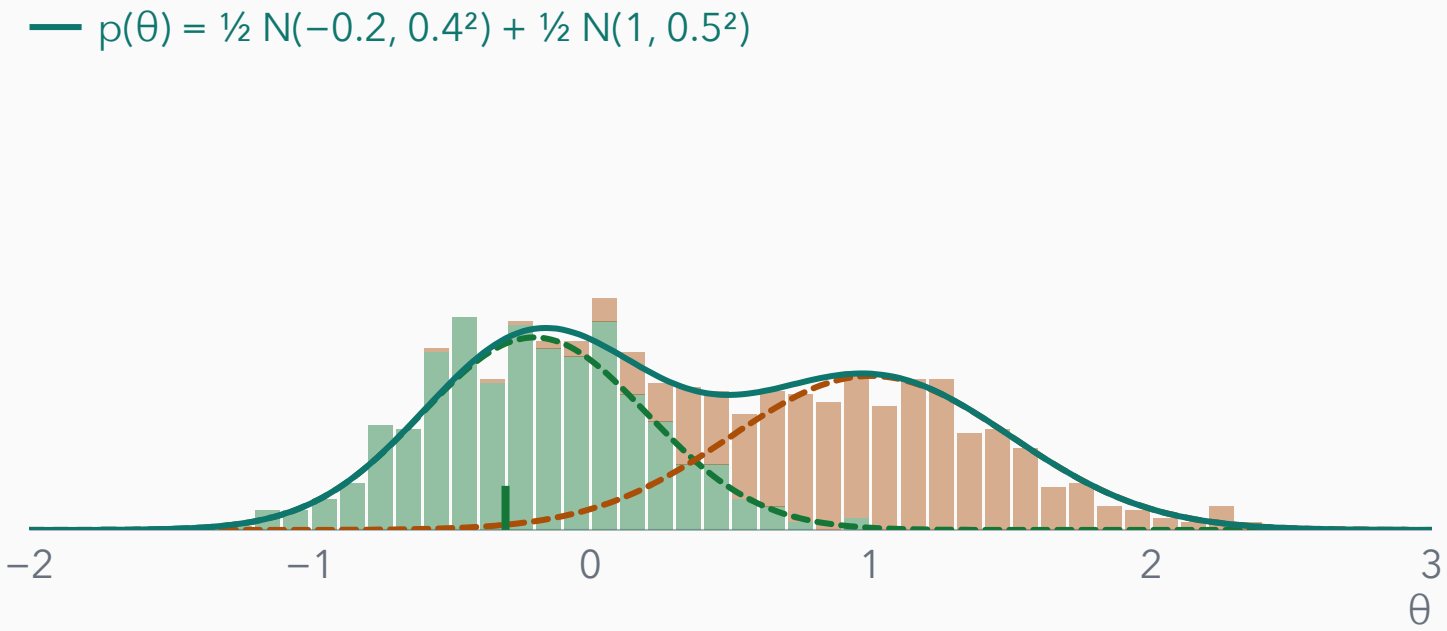
bump 1: probability ½
N(−0.2, 0.4²) · picked 539×

bump 2: probability ½
N(1.0, 0.5²) · picked 461×

step 2: draw θ from that bump

latest: bump 1, θ = −0.30

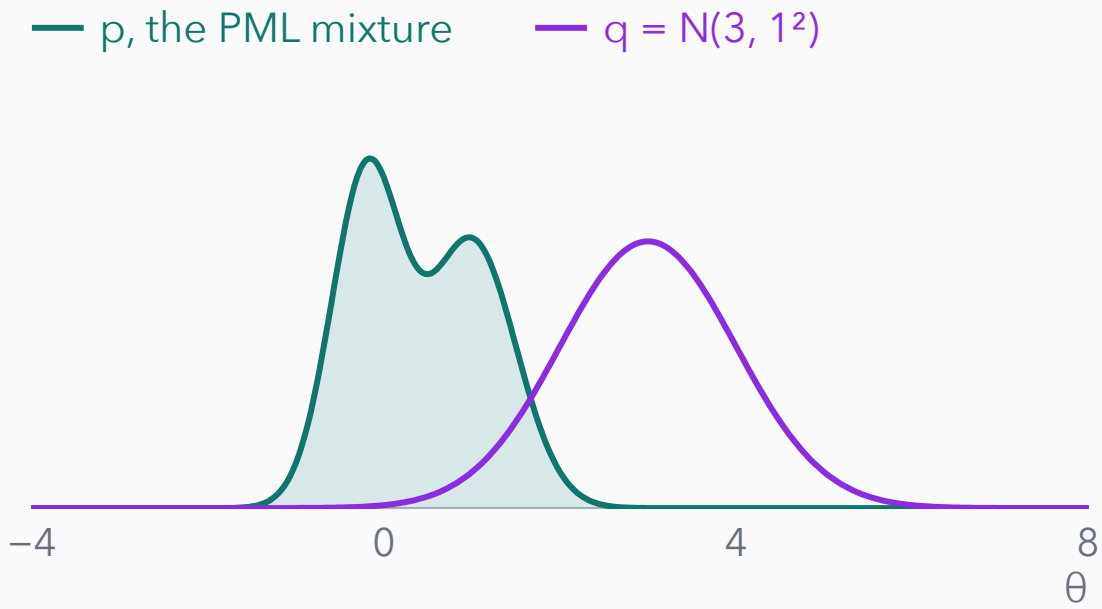
1,000 draws so far



> Each draw first picks a bump (its colour), then draws θ from it. Add 10, 100, then 1,000 draws and watch the histogram fill in the teal density.

▶ Notebook 3.4

PyTorch has no formula for this KL. The definition still works.

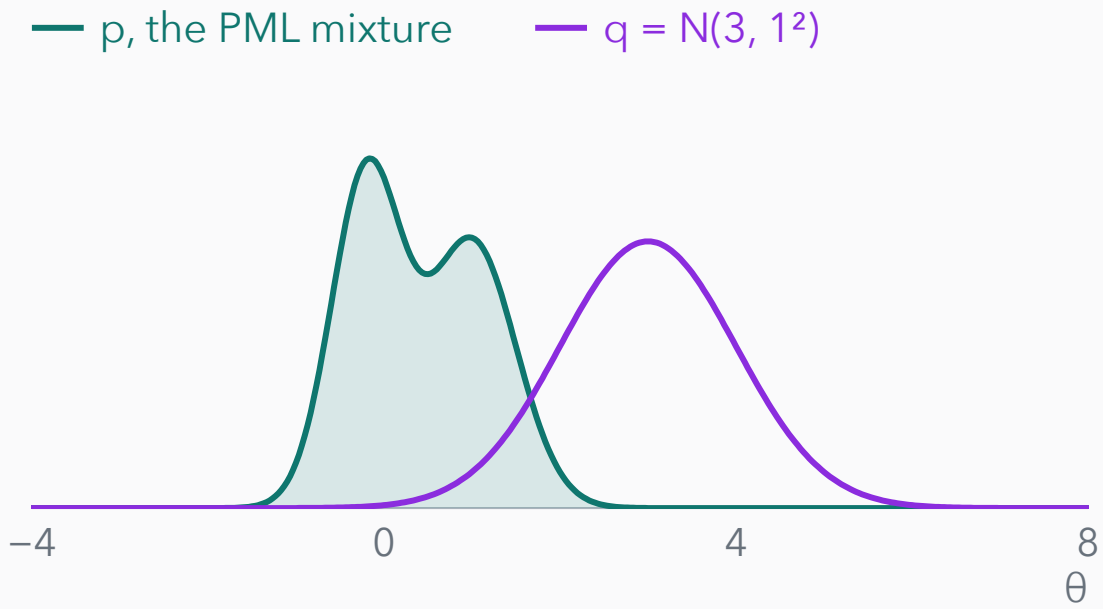



```
kl_divergence(q, mixture)
```

How far is a Gaussian q from this mixture? How would you compute KL(q ‖ p)? Then ask PyTorch.

▶ Notebook 3.4

PyTorch has no formula for this KL. The definition still works.




```
kl_divergence(q, mixture)
```

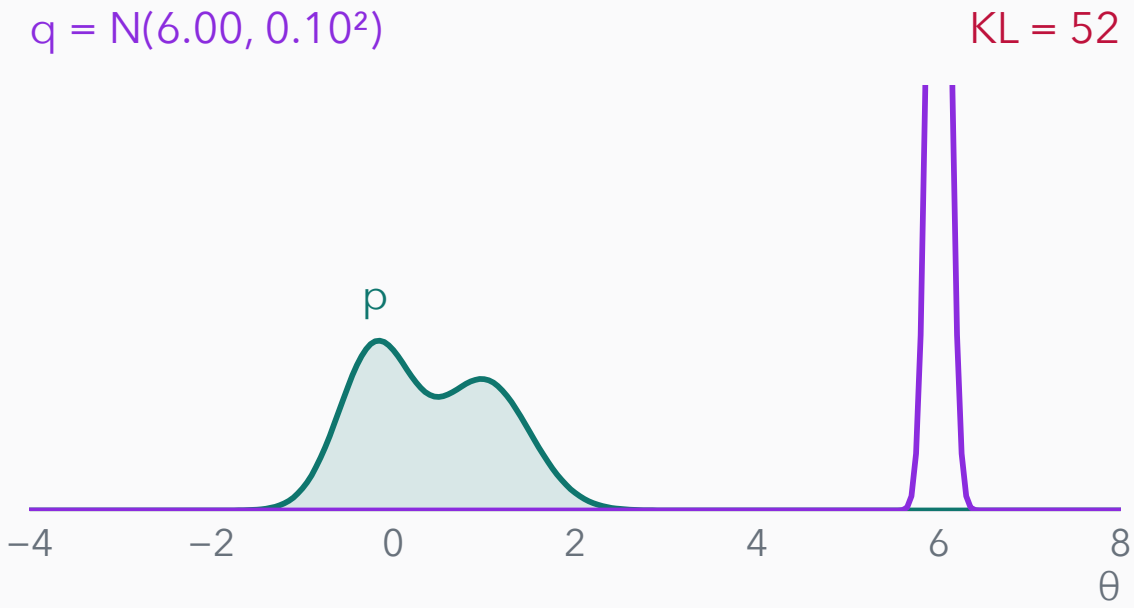
```
NotImplementedError:
No KL(p || q) is implemented
for p type Normal and
q type MixtureSameFamily
```

No formula exists for a Gaussian against a mixture. But KL(q‖p) is an average under q, and averages can be estimated by sampling, as in 2.7.

▶ Notebook 3.4

Now optimise it: the code runs, and q does not move

```
theta = q.sample((5000,))
log_q = q.log_prob(theta)
log_p = p.log_prob(theta)
loss = (log_q - log_p).mean()
opt.zero_grad()
loss.backward(); opt.step()
```

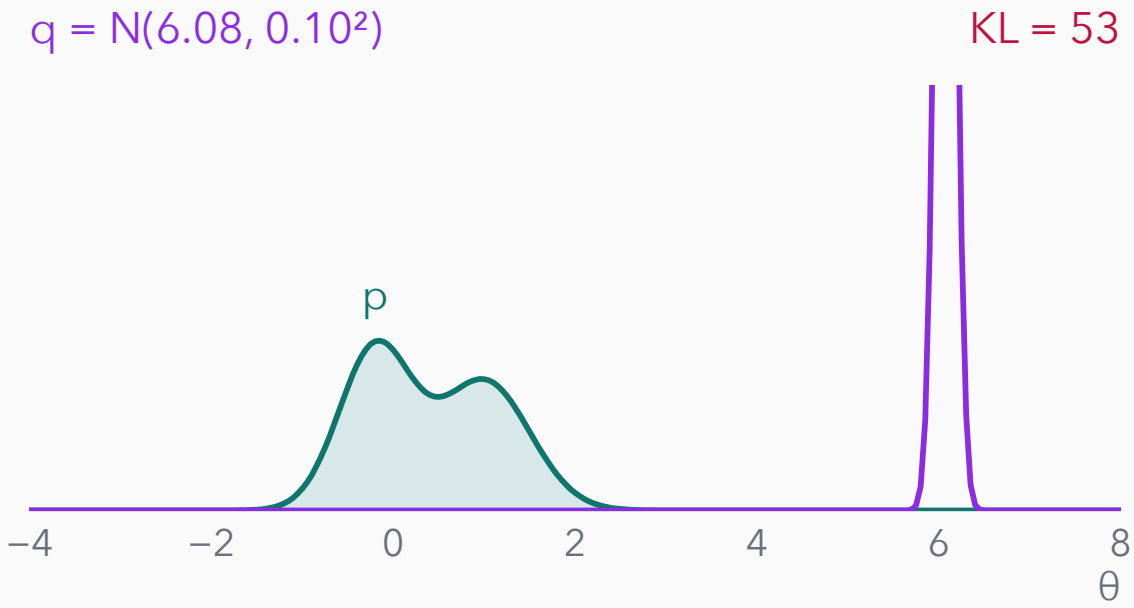


Use the sampled KL as the loss and run the same loop on the mixture. Predict: does q move?

▶ Notebook 3.4

Now optimise it: the code runs, and q does not move

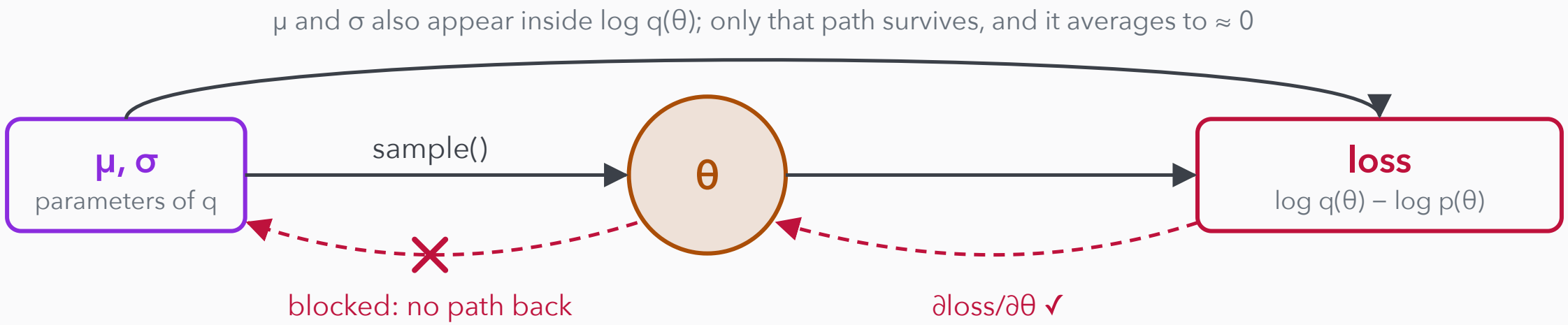
```
theta = q.sample((5000,))
log_q = q.log_prob(theta)
log_p = p.log_prob(theta)
loss = (log_q - log_p).mean()
opt.zero_grad()
loss.backward(); opt.step()
```



No error, no warning, and q stays at N(6, 0.1²): after 400 steps the KL is still about 53. Is this a bug? Hold that question: Section 3.5 answers it.

▶ Notebook 3.5

Why q does not move: the gradient stops at a random node

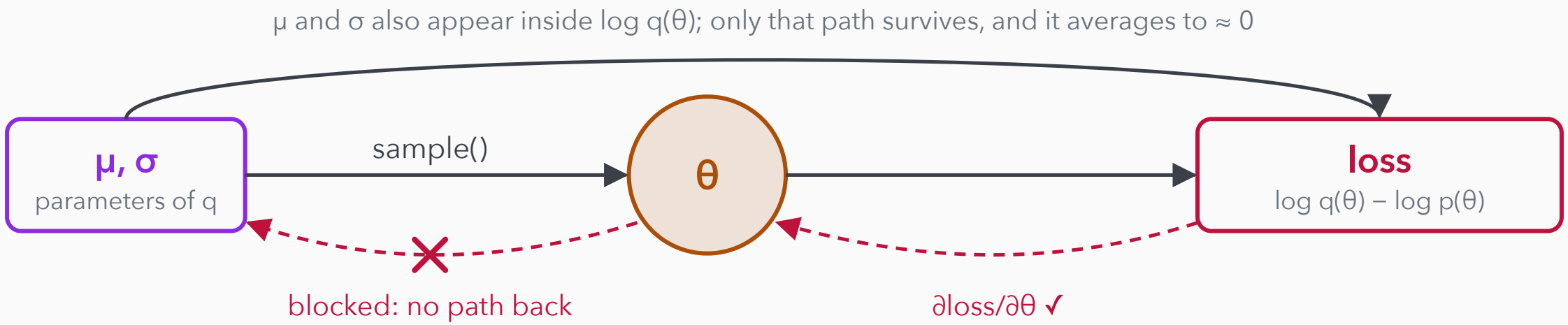


Boxes are deterministic nodes, computed from their inputs; circles are stochastic, drawn at random. With sample(), θ is stochastic, and the backward pass (red) cannot get from θ back to μ and σ.

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE", which follows Kingma's NIPS 2015 talk

▶ Notebook 3.5

Why q does not move: the gradient stops at a random node



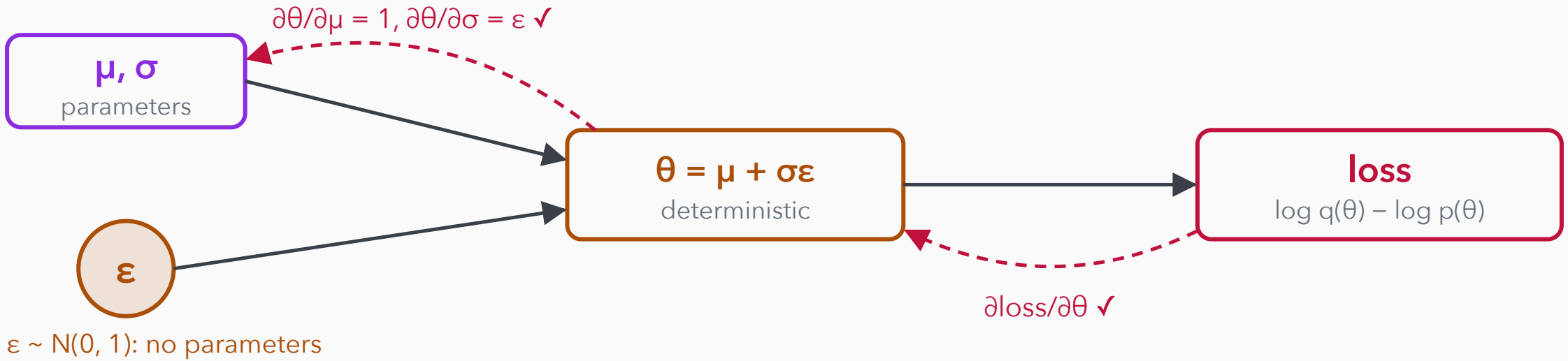
	$\partial \mathrm{KL}/\partial \mu$	$\partial \mathrm{KL}/\partial \sigma$
what autograd gets through sample()	0.0013	−0.0003
the true gradient	−0.063	−0.375

> Only the path through log q(θ)'s formula survives, and it averages to zero: 0.0013 and −0.0003 against −0.063 and −0.375. Not a bug: a missing path in the graph.

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE", which follows Kingma's NIPS 2015 talk

▶ Notebook 3.5

The fix: move the randomness into ε, and θ = μ + σε



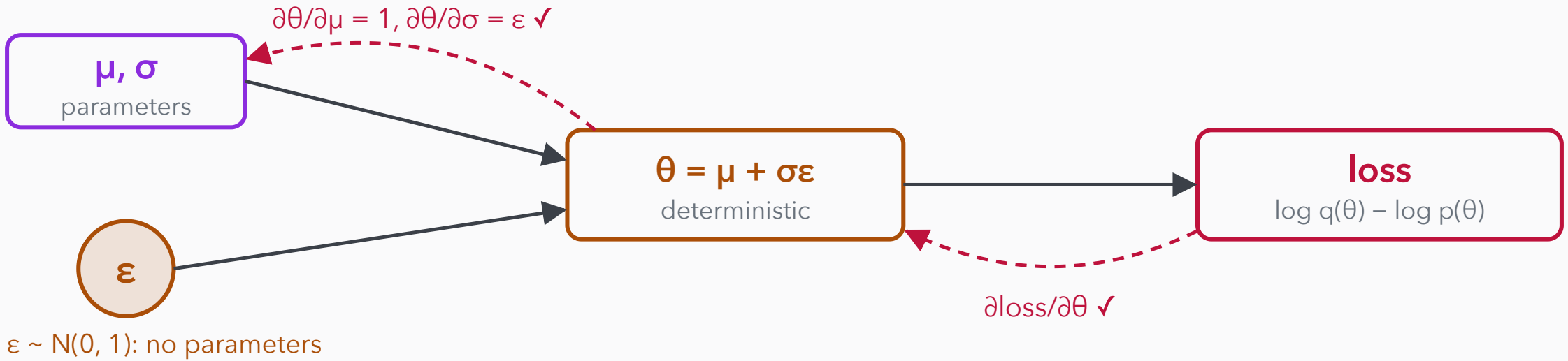
$$\epsilon \sim \mathcal{N}(0,1), \qquad \theta = \mu + \sigma\epsilon$$

> Draw the noise ε from a fixed N(0, 1), then shift and scale it. θ has the same distribution, N(μ, σ²), but it is now a deterministic node: a function of μ, σ and ε.

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE"

▶ Notebook 3.5

The fix: move the randomness into ε, and θ = μ + σε



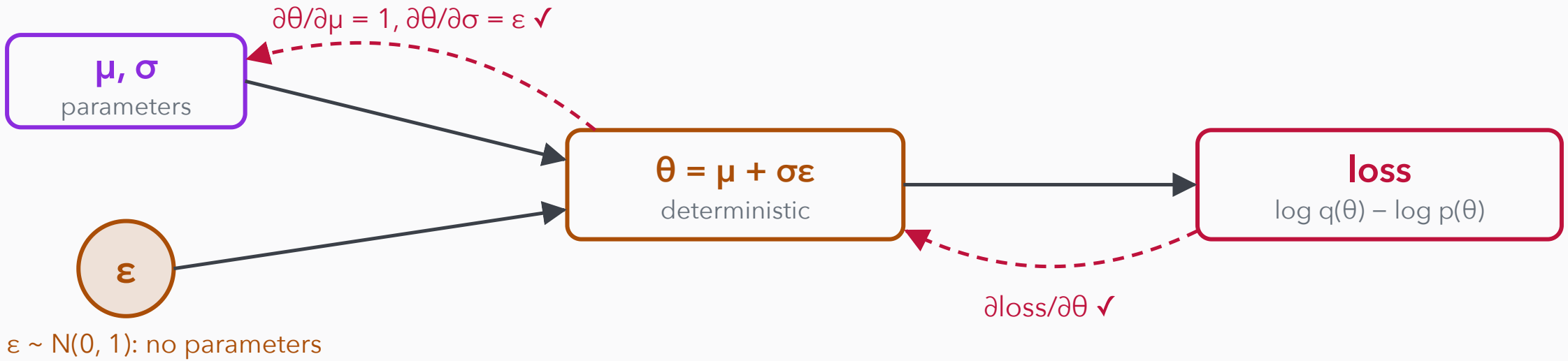
$$\epsilon \sim \mathcal{N}(0,1), \qquad \theta = \mu + \sigma\epsilon$$

Pause: with ε = −0.75, μ = 0 and σ = 2, find θ, ∂θ/∂μ and ∂θ/∂σ.

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE"

▶ Notebook 3.5

The fix: move the randomness into ε, and θ = μ + σε



$$\epsilon \sim \mathcal{N}(0,1), \qquad \theta = \mu + \sigma\epsilon \qquad\qquad \theta = -1.5, \quad \frac{\partial\theta}{\partial\mu} = 1, \quad \frac{\partial\theta}{\partial\sigma} = \epsilon = -0.75$$

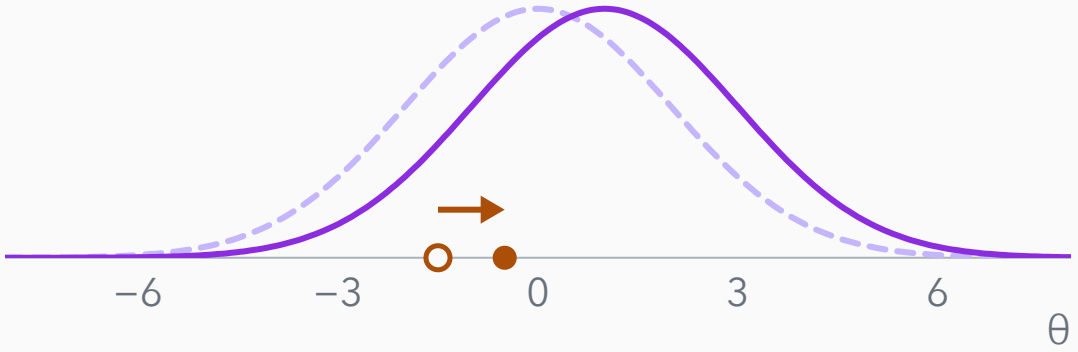
> θ = −1.5, ∂θ/∂μ = 1 and ∂θ/∂σ = ε = −0.75. The only random node, ε, has no parameters, so nothing blocks the path. In PyTorch this is rsample().

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE"

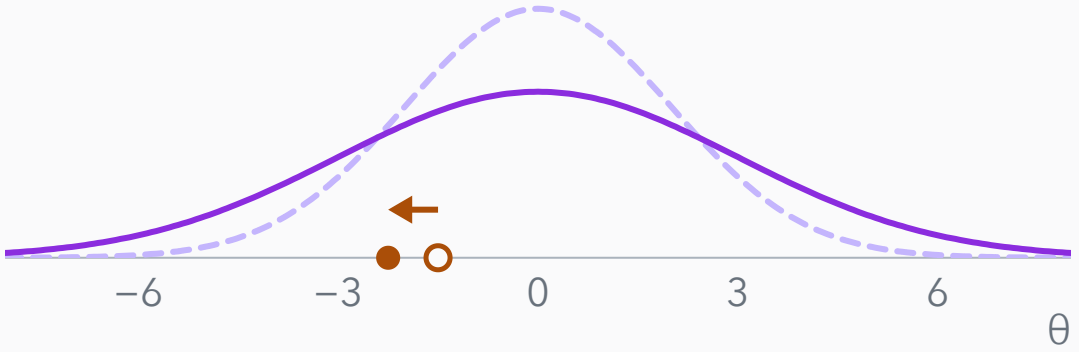
▶ Notebook 3.5

What a fixed ε does when μ or σ changes

move μ from 0 to 1 θ: −1.50 → −0.50



widen σ from 2 to 3 θ: −1.50 → −2.25

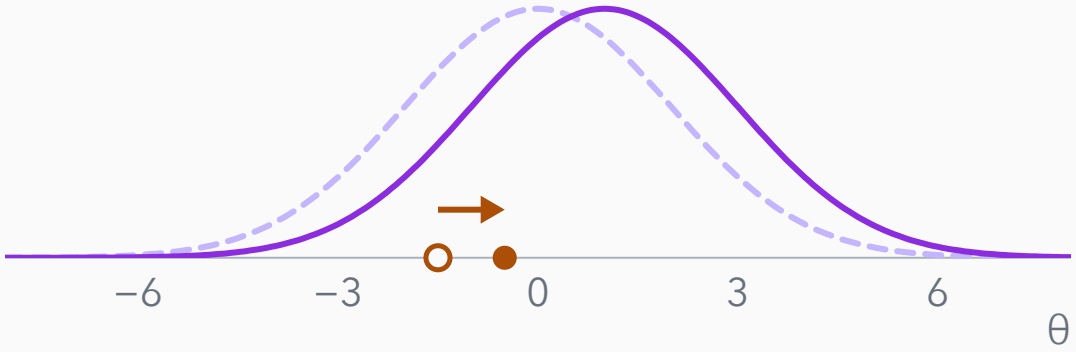


Keep the same noise draw, ε = −0.75. Shift μ by 1: the draw moves right by exactly 1.

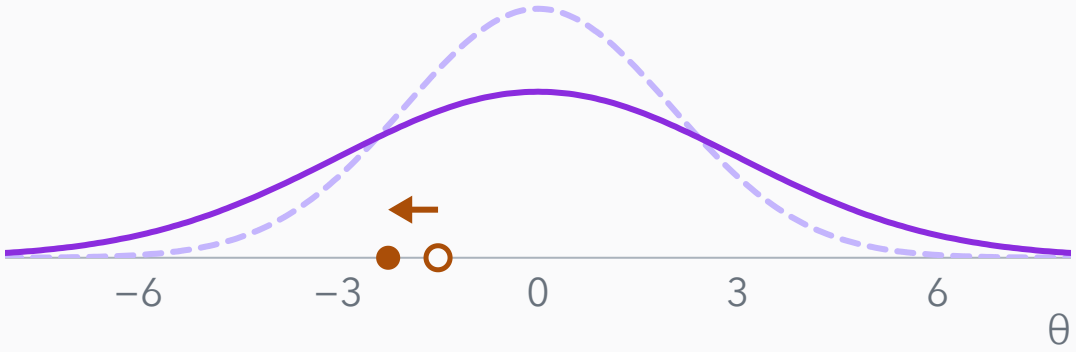
▶ Notebook 3.5

What a fixed ε does when μ or σ changes

move μ from 0 to 1 θ: −1.50 → −0.50



widen σ from 2 to 3 θ: −1.50 → −2.25

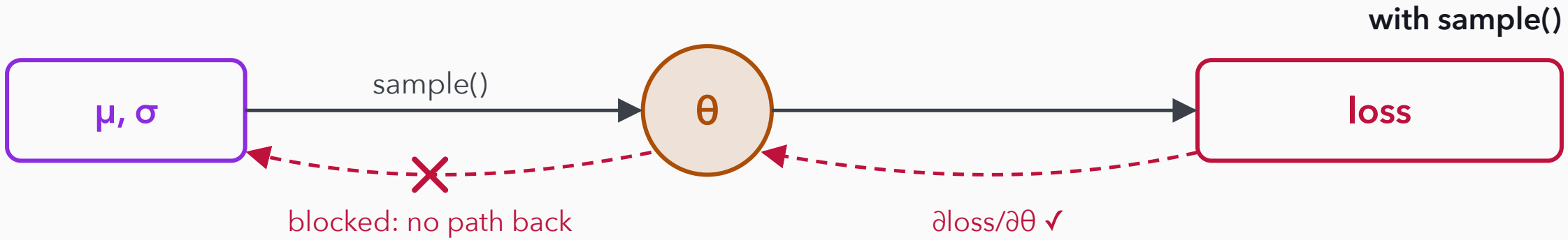


$$\frac{\partial \theta}{\partial \mu} = 1, \qquad \frac{\partial \theta}{\partial \sigma} = \epsilon = -0.75$$

> Widen σ by 1: the draw moves by ε = −0.75, further out on its side. These are the derivatives the gradient needs.

▶ Notebook 3.5

Side by side: where the randomness lives



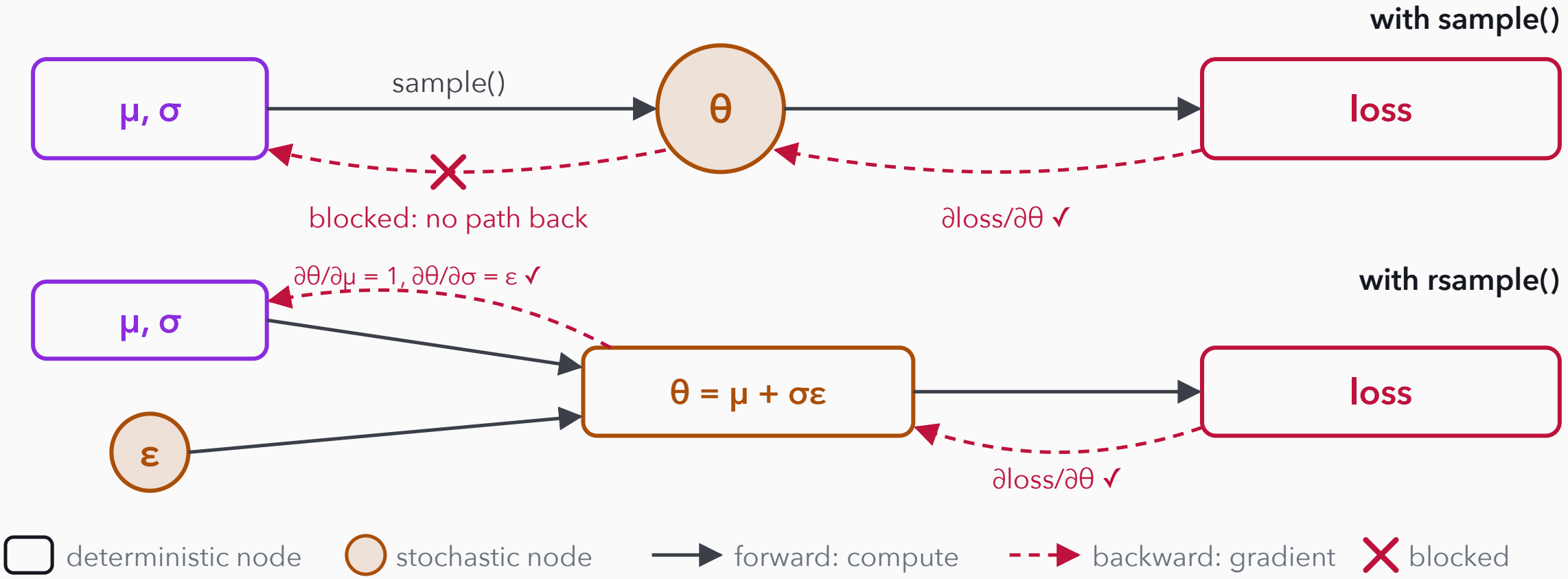
deterministic node · stochastic node · forward: compute · backward: gradient · blocked

> With sample(), the random node sits between the parameters and the loss, and the gradient cannot cross it.

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE"

▶ Notebook 3.5

Side by side: where the randomness lives



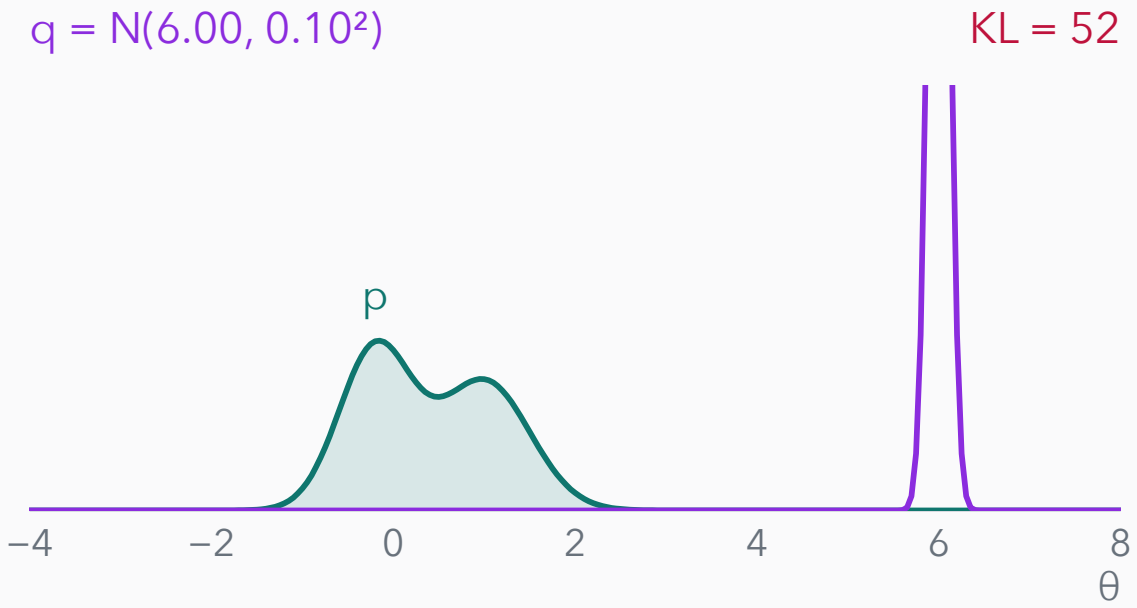
With rsample(), the randomness enters as a parameter-free input ε and θ = μ + σε is deterministic: the gradient flows from the loss to μ and σ. Same distribution, different graph.

Graph conventions after Lilian Weng, "From Autoencoder to Beta-VAE"

▶ Notebook 3.5

Change one word: rsample() fits the mixture

```
# one word changed
theta = q.rsample((5000,))
log_q = q.log_prob(theta)
log_p = p.log_prob(theta)
loss = (log_q - log_p).mean()
opt.zero_grad()
loss.backward(); opt.step()
```

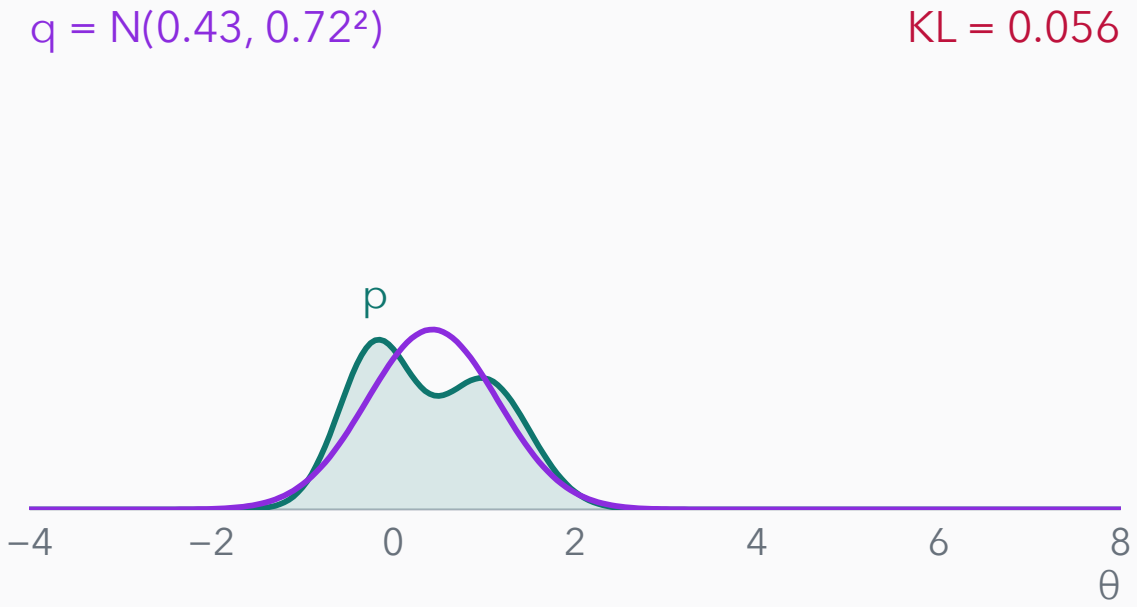


Same loop, same start, same seed, with q.rsample. Press play.

▶ Notebook 3.5

Change one word: rsample() fits the mixture

```
# one word changed
theta = q.rsample((5000,))
log_q = q.log_prob(theta)
log_p = p.log_prob(theta)
loss = (log_q - log_p).mean()
opt.zero_grad()
loss.backward(); opt.step()
```



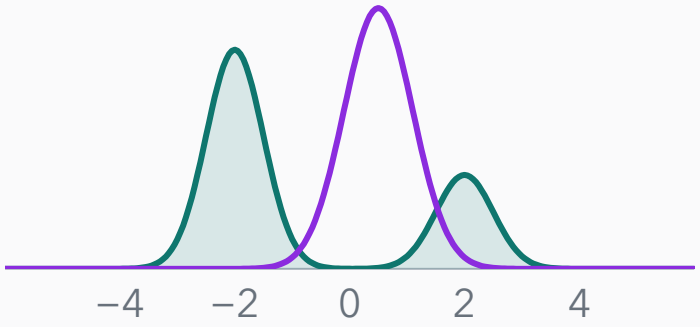
q settles at N(0.43, 0.72²), KL = 0.056: the best single Gaussian for this mixture. No formula was needed.

▶ Notebook 3.6

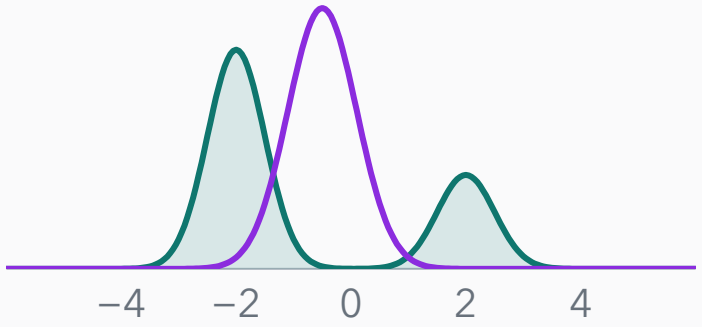
The price: reverse KL picks one bump

teal: the target p, fixed, with two bumps · **purple: q**, one Gaussian; we learn its mean and width

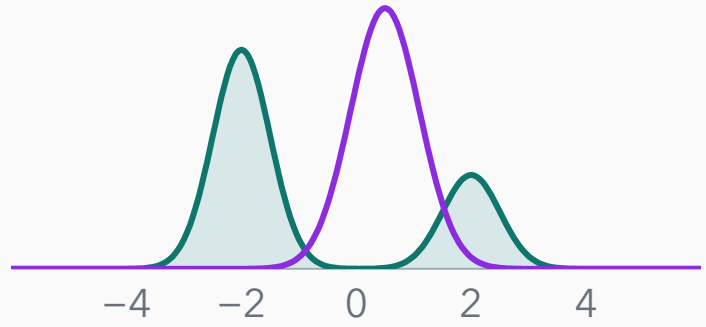
KL(p||q) 6.43



KL(q||p), start left 3.92



KL(q||p), start right 4.42



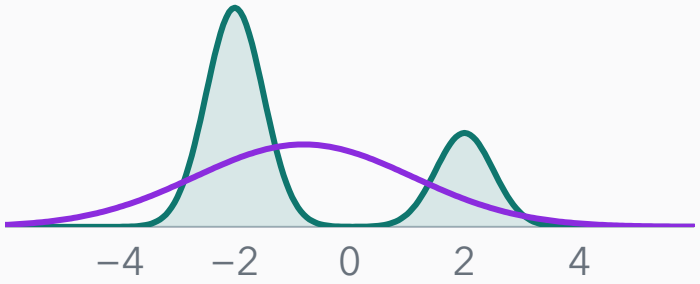
> Same target p, same family for q, same optimiser. Only the direction of the KL changes. We learn q's two knobs, mean and width, three times. Predict where each q settles.

▶ Notebook 3.6

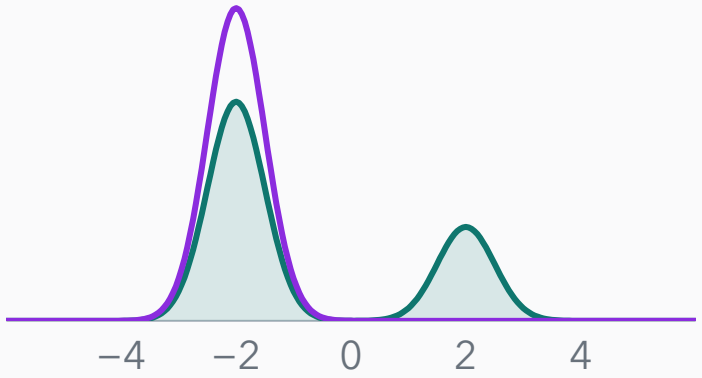
The price: reverse KL picks one bump

teal: the target p, fixed, with two bumps · **purple: q**, one Gaussian; we learn its mean and width

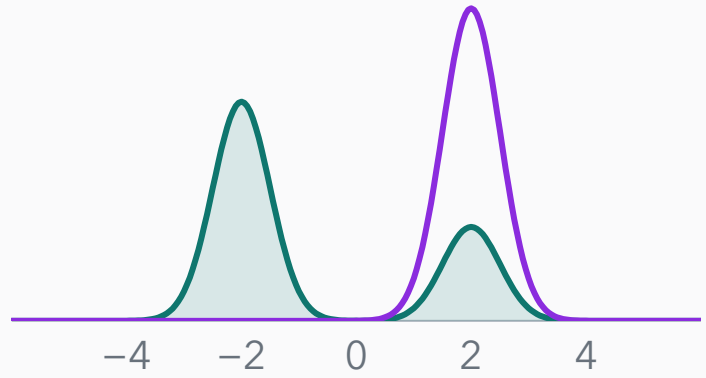
KL(p||q) 0.72



KL(q||p), start left 0.36



KL(q||p), start right 1.20



> KL(p ‖ q) spreads q over both bumps. KL(q ‖ p) locks onto the bump nearer its start, and reports it confidently. Why?

▶ Notebook 3.6

Why: reverse KL charges q for mass where p is empty

q stretched across both bumps



q on the left bump



$$\mathrm{KL}(q \,\|\, p) = \int q(\theta)\, \log \frac{q(\theta)}{p(\theta)}\, d\theta$$

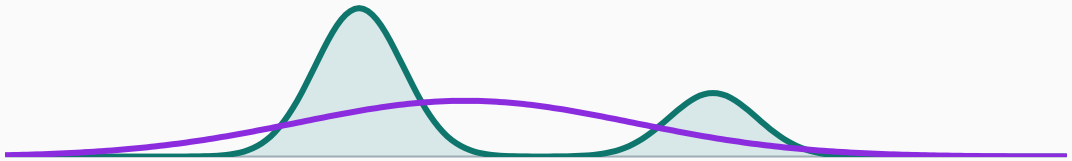
large where q is big and $p \approx 0$

Reverse KL averages under q. A q stretched across both bumps puts mass in the gap, where p is almost 0, and log q/p explodes there: cost 2.09. Sitting on one bump costs only 0.36.

▶ Notebook 3.6

Why: reverse KL charges q for mass where p is empty

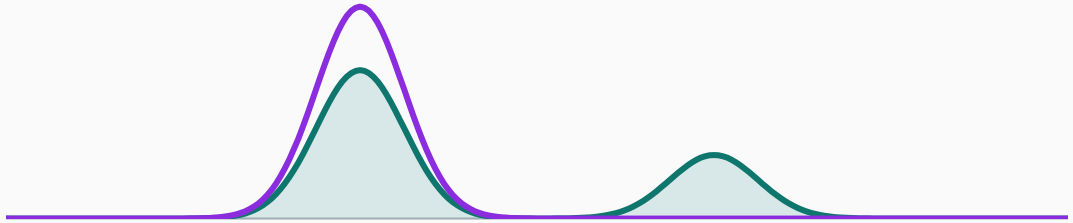
q stretched across both bumps



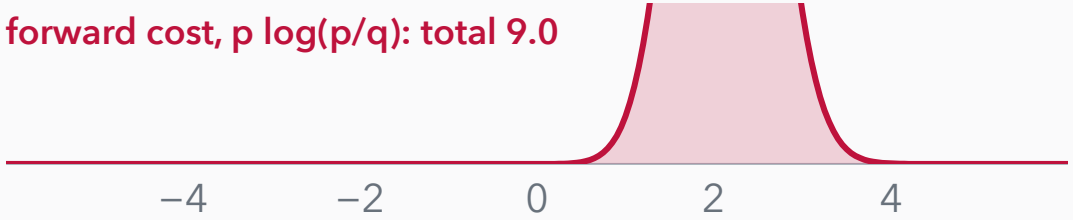
forward cost, p log(p/q): total 0.72



q on the left bump



forward cost, p log(p/q): total 9.0



$$\mathrm{KL}(p \,\|\, q) = \int p(\theta)\, \log \frac{p(\theta)}{q(\theta)}\, d\theta$$

large where p is big and $q \approx 0$

> Forward KL averages under p. Now the danger is the other way round: a q on one bump puts almost nothing where p has its second bump, and pays 9.0. Spreading out costs only 0.72.

▶ Notebook 3.6

What to learn: the direction decides what q is punished for

the fitted q	reverse: KL(q ‖ p)	forward: KL(p ‖ q)
on the left bump	**0.36** (best)	9.0
across both bumps	2.09	**0.72** (best)
on the right bump	1.20	21.7

Same three candidates, scored both ways: each direction prefers a different q.

> Score the three fitted q both ways. Reverse KL prefers the left bump; forward KL prefers the wide q. The same q can be best in one direction and worst in the other.

▶ Notebook 3.6

What to learn: the direction decides what q is punished for

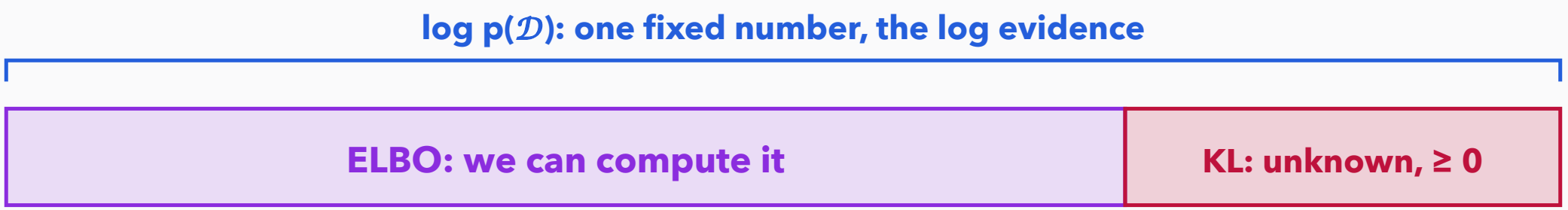
	forward: KL(p ‖ q)	reverse: KL(q ‖ p)
averages over	p, the target	q, our approximation
punishes q for	missing mass where p has it	putting mass where p has none
so q tends to	cover every bump, and spread	hug one bump, and be too narrow
needs	samples from p	samples from q, and p up to a constant
used in	classification, distillation	variational inference

The lesson: VI uses reverse KL, so a VI posterior can miss modes and be overconfident. Richer families (1.5.3) or MCMC help check it.

> The direction decides what q is punished for. VI must use reverse KL, because it averages under q, which we can sample. The price: VI tends to be overconfident and can miss modes.

▶ Notebook

Part 4 · Back to Bayes: the ELBO



raise the ELBO, and the KL to the posterior must fall

4.1 Deriving the ELBO ▶

4.2 Checking it on a coin ▶

> Part 4 returns to the real posterior, which we cannot evaluate. Four small steps turn KL to the posterior into the ELBO, which we can.

▶ Notebook 4.1

Back to Bayes: what we wanted

$$p(\theta \mid \mathcal{D}) = \frac{\overbrace{p(\mathcal{D} \mid \theta)}^{\text{easy}} \; \overbrace{p(\theta)}^{\text{easy}}}{\underbrace{p(\mathcal{D})}_{10^{14}\text{ grid points}}}$$

> Back to Part 1's problem. The posterior needs the evidence p($\mathcal{D}$), and for our seven numbers that is a grid of 10^{14} points.

▶ Notebook 4.1

Back to Bayes: what we wanted

$$p(\theta \mid \mathcal{D}) = \frac{\overbrace{p(\mathcal{D} \mid \theta)}^{\text{easy}} \; \overbrace{p(\theta)}^{\text{easy}}}{\underbrace{p(\mathcal{D})}_{10^{14}\text{ grid points}}}$$

$$\text{VI: choose } q_\psi(\theta), \text{ then } \psi^* = \arg\min_\psi \, \mathrm{KL}\big(q_\psi(\theta) \,\|\, p(\theta \mid \mathcal{D})\big)$$

> VI's plan from Section 1.5.3: choose a q we control and make it as close as possible to the posterior, measured by KL(q ‖ posterior).

▶ Notebook 4.1

Where we are: the tools, and the one missing piece

part	what it gave us
1 · Bayes for a network	the target, the posterior, and an evidence we cannot compute
2 · Distance and KL	a number for how far q is from p
3 · Find the closest q	gradient descent on KL: rsample, mixtures, covariances, with p known
4 · now	**the same, with p the posterior: without ever computing p($\mathcal{D}$)**

> Parts 2 and 3 built the tools, always with a p we could evaluate. Now p is the posterior, which we cannot evaluate. That is where we are.

▶ Notebook 4.1

Back to Bayes: minimise KL to the posterior directly?

$$\mathrm{KL}\big(q_\psi(\theta) \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_{\theta\sim q}\big[\log q_\psi(\theta) - \log p(\theta \mid \mathcal{D})\big]$$

> The plan from Part 1: make q close to the posterior. Use KL(q ‖ posterior): an average under q, which we can sample.

A careful written derivation: Gregory Gundersen, "The ELBO in Variational Inference"

▶ Notebook 4.1

Back to Bayes: minimise KL to the posterior directly?

$$\mathrm{KL}\big(q_\psi(\theta) \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_{\theta\sim q}\big[\log q_\psi(\theta) - \log p(\theta \mid \mathcal{D})\big]$$

the loss needs	available?
draws θ ~ q	yes: rsample, as in 3.5
$\log q_\psi(\theta)$	yes
$\log p(\theta \mid \mathcal{D})$	no: it needs p($\mathcal{D}$), the integral of 1.4

> Everything is available except log p(θ | $\mathcal{D}$), which needs the evidence p($\mathcal{D}$): the integral we cannot do. Four small steps get rid of it.

A careful written derivation: Gregory Gundersen, "The ELBO in Variational Inference"

▶ Notebook 4.1

Step 1: Bayes' rule, in logs

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}$$

> Bayes' rule from Part 1: the posterior (teal) is the likelihood (orange) times the prior (green), divided by the evidence (blue).

▶ Notebook 4.1

Step 1: Bayes' rule, in logs

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}$$

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D})$$

Rule: log(a · b / c) = log a + log b − log c.

> Take logs. A product becomes a sum, and a division a subtraction.

▶ Notebook 4.1

Step 1: Bayes' rule, in logs

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}$$

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D})$$

Rule: log(a · b / c) = log a + log b − log c.

Look at the last term: $\log p(\mathcal{D})$ has no θ in it. Whatever θ is, it is the same number, fixed by our model and our data.

> The last term does not contain θ: it is one fixed number. Remember that; it does the work in step 3.

▶ Notebook 4.1

Step 2a: the KL we want, piece by piece

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_{\theta\sim q}\big[\, \log q(\theta) - \log p(\theta \mid \mathcal{D}) \,\big]$$

> The quantity to make small: KL(q ‖ posterior). It is an average, over draws θ from q, of log q minus log posterior.

▶ Notebook 4.1

Step 2a: the KL we want, piece by piece

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_{\theta\sim q}\big[\, \log q(\theta) - \log p(\theta \mid \mathcal{D}) \,\big]$$

$$\mathbb{E}_{\theta\sim q}\Big[\quad \underbrace{\log q(\theta)}_{\text{ours: computable}} \quad - \quad \underbrace{\log p(\theta \mid \mathcal{D})}_{\text{needs } p(\mathcal{D})\text{: not computable}} \quad\Big]$$

$E_{\theta\sim q}[\,\cdot\,]$ means: draw θ from q, evaluate what is in the bracket, average over the draws.

> Two pieces. log q is ours: we chose q, so we can evaluate it. The log posterior is the problem: it needs the evidence. Step 1 splits it.

▶ Notebook 4.1

Step 2b: swap in step 1's three pieces

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D})$$

Step 1 wrote the log posterior as three pieces: log-likelihood, plus log-prior, minus log-evidence.

▶ Notebook 4.1

Step 2b: swap in step 1's three pieces

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D})$$

$$\mathrm{KL} = \mathbb{E}_q \Big[\log q(\theta) - \Big(\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D}) \Big) \Big]$$

> Put those three pieces where the log posterior was, inside a teal bracket. Nothing has changed: the same number, written differently.

▶ Notebook 4.1

Step 2b: swap in step 1's three pieces

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D})$$

$$\mathrm{KL} = \mathbb{E}_q\Big[\, \log q(\theta) - \Big(\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log p(\mathcal{D}) \Big) \Big]$$

colour	term	what it is
purple	$\log q(\theta)$	our approximation
orange	$\log p(\mathcal{D} \mid \theta)$	the likelihood
green	$\log p(\theta)$	the prior
blue	$\log p(\mathcal{D})$	the evidence

> From here on each piece keeps its colour, so you can follow it to the end of step 4.

▶ Notebook 4.1

Step 2c: open the bracket, one sign at a time

$$-(a+b-c) = -a-b+c$$

A minus sign in front of a bracket flips the sign of every term inside it.

▶ Notebook 4.1

Step 2c: open the bracket, one sign at a time

$$-(a+b-c) = -a-b+c$$

$$-(5+3-2) = -6 \qquad \text{and} \qquad -5-3+2 = -6$$

Check with numbers: minus (5 + 3 − 2) is −6, and so is −5 − 3 + 2.

▶ Notebook 4.1

Step 2c: open the bracket, one sign at a time

$$-(a+b-c) = -a-b+c$$

$$-(5+3-2) = -6 \qquad \text{and} \qquad -5-3+2 = -6$$

$$\mathrm{KL} = \mathbb{E}_q\big[\,\log q(\theta) - \log p(\mathcal{D}\mid\theta) - \log p(\theta) + \log p(\mathcal{D})\,\big]$$

Apply it: the likelihood and the prior now enter with minus signs, and the evidence with a plus.

▶ Notebook 4.1

Step 2, done: four terms inside one average

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_q\Big[\, \underbrace{\log q(\theta)}_{\text{ours}} - \underbrace{\log p(\mathcal{D} \mid \theta)}_{\text{likelihood}} - \underbrace{\log p(\theta)}_{\text{prior}} + \underbrace{\log p(\mathcal{D})}_{\text{evidence}} \,\Big]$$

After step 2 the KL is one average of four terms.

▶ Notebook 4.1

Step 2, done: four terms inside one average

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_q\Big[\ \underbrace{\log q(\theta)}_{\text{ours}} - \underbrace{\log p(\mathcal{D} \mid \theta)}_{\text{likelihood}} - \underbrace{\log p(\theta)}_{\text{prior}} + \underbrace{\log p(\mathcal{D})}_{\text{evidence}}\ \Big]$$

term	changes with θ?	can we compute it?
$\log q(\theta)$	yes	yes: we chose q
$\log p(\mathcal{D} \mid \theta)$	yes	yes: run the network on the data
$\log p(\theta)$	yes	yes: the prior is our choice
$\log p(\mathcal{D})$	**no: one number**	**no: the integral of 1.4**

> Three change with θ and are computable. The blue one, log p($\mathcal{D}$), is not computable, but it has no θ in it: the same number for every draw. Step 3 uses exactly that.

▶ Notebook 4.1

Step 3: a constant comes out of an average

$$\mathbb{E}_q\big[f(\theta) + c\big] = \mathbb{E}_q\big[f(\theta)\big] + c \qquad \text{when } c \text{ does not depend on } \theta$$

> A rule about averages: add the same number c to every value, and the average goes up by exactly c.

▶ Notebook 4.1

Step 3: a constant comes out of an average

$$\mathbb{E}_q\big[f(\theta) + c\big] = \mathbb{E}_q\big[f(\theta)\big] + c \qquad \text{when } c \text{ does not depend on } \theta$$

	three draws	their average
$f(\theta)$	2, 4, 6	4
$f(\theta) + 10$	12, 14, 16	14 = 4 + 10

> Check it: the average of 2, 4 and 6 is 4. Add 10 to each and the average is 14.

▶ Notebook 4.1

Step 3: a constant comes out of an average

$$\mathbb{E}_q\big[f(\theta) + c\big] = \mathbb{E}_q\big[f(\theta)\big] + c \qquad \text{when } c \text{ does not depend on } \theta$$

	three draws	their average
$f(\theta)$	2, 4, 6	4
$f(\theta) + 10$	12, 14, 16	14 = 4 + 10

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = \mathbb{E}_q\big[\log q(\theta) - \log p(\mathcal{D} \mid \theta) - \log p(\theta)\big] + \log p(\mathcal{D})$$

> log p($\mathcal{D}$) is such a constant: step 1 showed it has no θ. So it leaves the average.

▶ Notebook 4.1

Step 4: rearrange, and name the part we can compute

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = -\mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big] + \log p(\mathcal{D})$$

> Take a minus sign outside the average: inside, the likelihood and the prior now have plus signs, and log q a minus.

▶ Notebook 4.1

Step 4: rearrange, and name the part we can compute

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = -\mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big] + \log p(\mathcal{D})$$

$$\log p(\mathcal{D}) = \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big] + \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

Move that average to the other side: log p($\mathcal{D}$) = average + KL.

▶ Notebook 4.1

Step 4: rearrange, and name the part we can compute

$$\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big) = -\mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big] + \log p(\mathcal{D})$$

$$\log p(\mathcal{D}) = \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big] + \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

$$\log p(\mathcal{D}) = \underbrace{\mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big]}_{\mathrm{ELBO}(q)} + \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

> Give the average a name: the ELBO, the evidence lower bound. Every term in it is computable: draws from q, the likelihood, the prior and log q. No p($\mathcal{D}$) anywhere.

▶ Notebook 4.1

One fixed number = what we can compute + what we cannot

$$\underbrace{\log p(\mathcal{D})}_{\text{fixed}} = \underbrace{\mathrm{ELBO}(q)}_{\text{computable}} + \underbrace{\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)}_{\text{unknown, but } \geq 0}$$

> Read the identity as a budget: one fixed number, split into a part we can compute and a part we cannot.

▶ Notebook 4.1

One fixed number = what we can compute + what we cannot

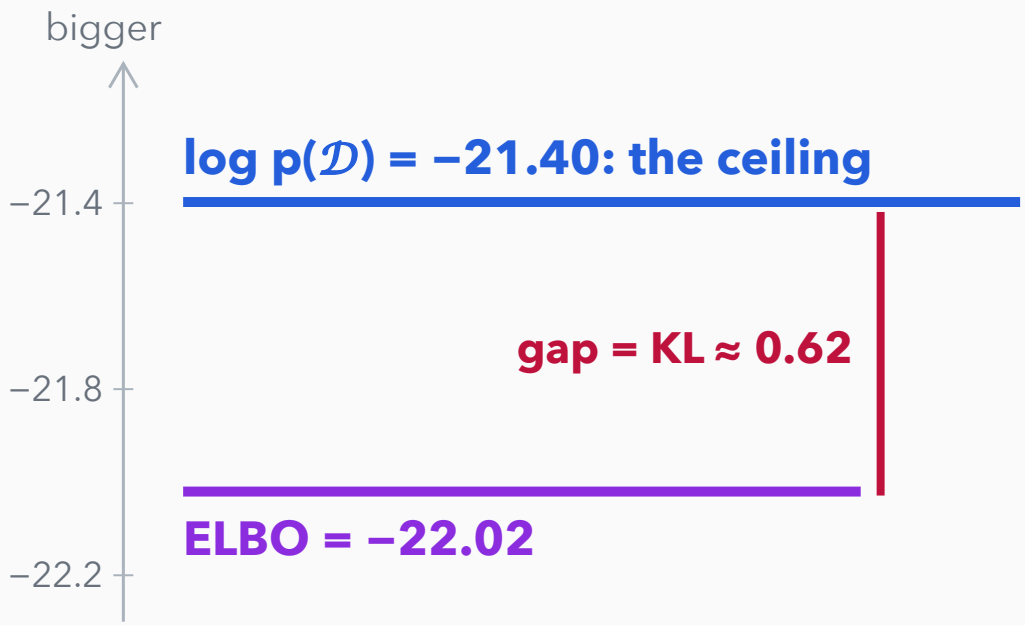
$$\underbrace{\log p(\mathcal{D})}_{\text{fixed}} = \underbrace{\mathrm{ELBO}(q)}_{\text{computable}} + \underbrace{\mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)}_{\text{unknown, but } \geq 0}$$

piece	what it is	changes with q?
$\log p(\mathcal{D})$	the log evidence: the data's probability, averaged over every θ	no: fixed
$\mathrm{ELBO}(q)$	average under q of log likelihood + log prior − log q	yes; computable
the KL	how far q is from the posterior	yes; ≥ 0, and 0 only at the posterior

> The KL is the part we want small. We cannot compute it, but we know one thing about it: it is never negative.

▶ Notebook 4.1

The ELBO is a lower bound: it sits below log p($\mathcal{D}$)



$$\log p(\mathcal{D}) = \mathrm{ELBO}(q) + \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

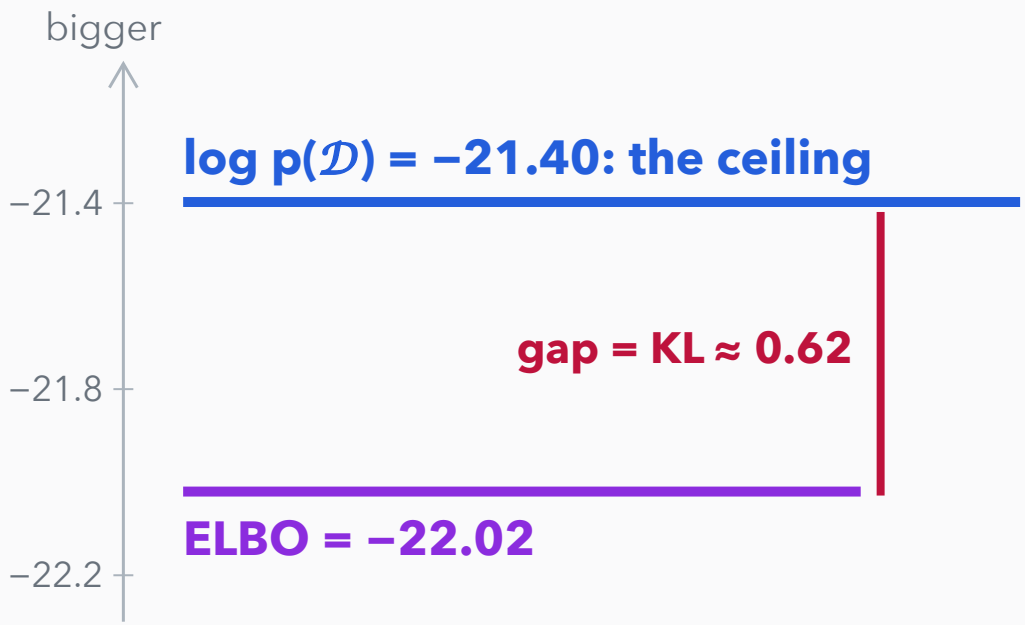
$$\mathrm{KL} \geq 0 \quad \Longrightarrow \quad \mathrm{ELBO}(q) \leq \log p(\mathcal{D})$$

Whatever q we choose, the **ELBO** is **below** (or at) the **log evidence**, never above: a **lower bound** on log p($\mathcal{D}$). **E**vidence **L**ower **BO**und.

> Picture a ceiling: log p($\mathcal{D}$), one fixed number set by the model and the data. Every q gives an ELBO, and every ELBO sits below the ceiling, by exactly the KL. That is why it is a lower bound.

▶ Notebook 4.1

The ELBO is a lower bound: it sits below log p($\mathcal{D}$)



$$\log p(\mathcal{D}) = \mathrm{ELBO}(q) + \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

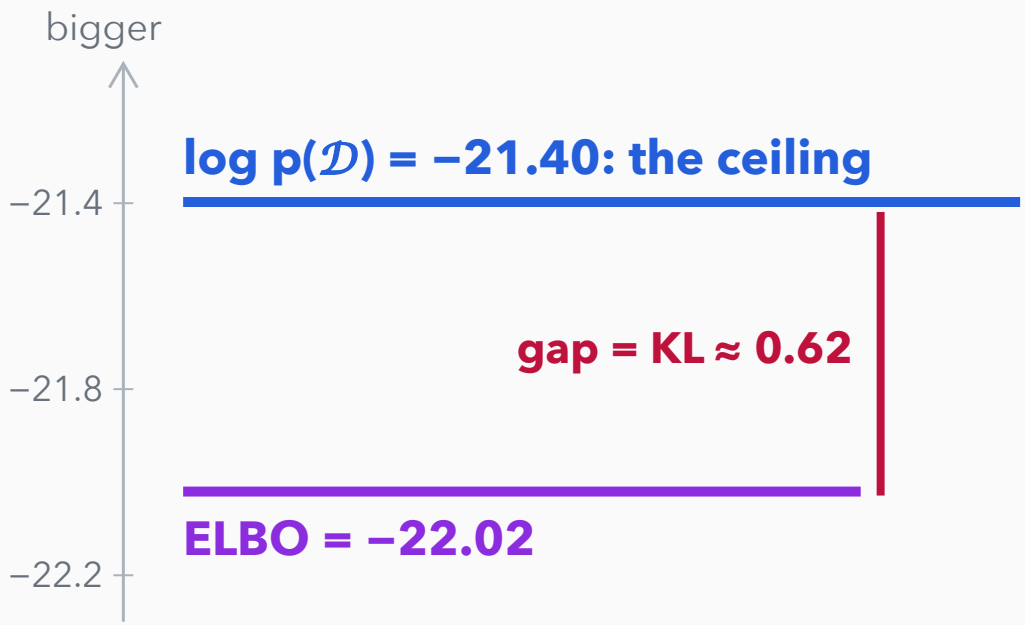
$$\mathrm{KL} \geq 0 \quad \Longrightarrow \quad \mathrm{ELBO}(q) \leq \log p(\mathcal{D})$$

Both numbers are negative. −22.02 is further below zero than −21.40, so it is the smaller one: 0.62 lower, exactly the KL.

The coin, with q = Beta(10, 10): log p($\mathcal{D}$) = −21.40 and the ELBO = −22.02. With negative numbers, “lower” means further below zero: the ELBO is 0.62 lower.

▶ Notebook 4.1

The ELBO is a lower bound: it sits below log p($\mathcal{D}$)



$$\log p(\mathcal{D}) = \mathrm{ELBO}(q) + \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

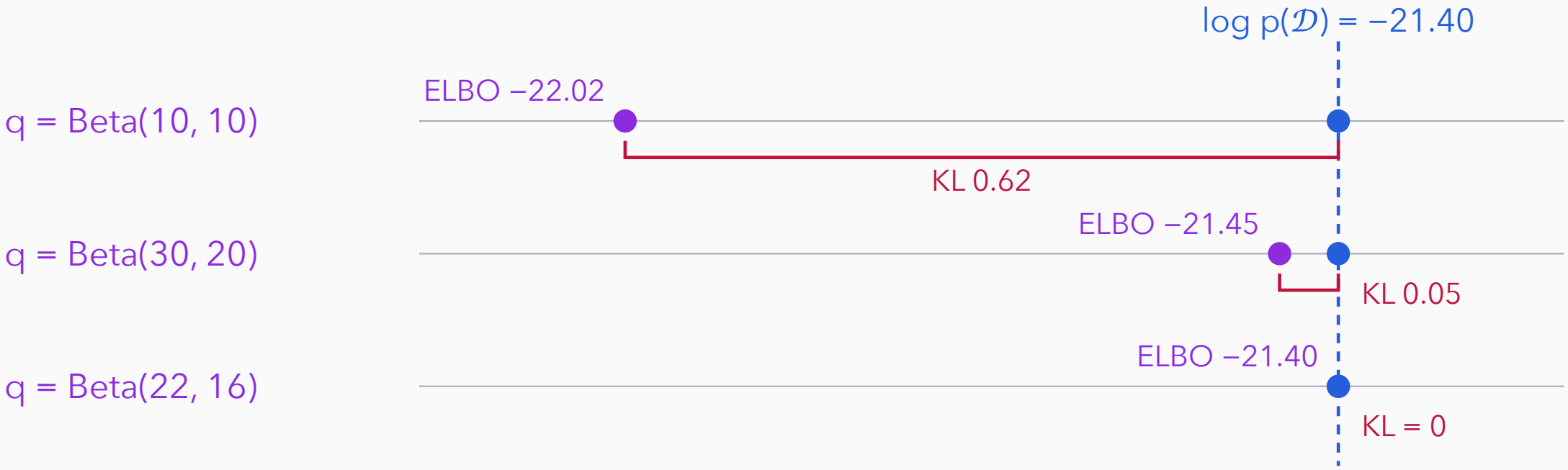
$$\mathrm{KL} \geq 0 \quad \Longrightarrow \quad \mathrm{ELBO}(q) \leq \log p(\mathcal{D})$$

	log p($\mathcal{D}$)	it is the log of …
the coin: discrete data	−21.40	a probability: never above 0
our network: continuous data	−1.43	a density: either sign

"Lower bound" says nothing about signs: a probability is at most 1, but a density can exceed it (log p($\mathcal{D}$ | θ_A) = +12.2). Only the order is guaranteed.

▶ Notebook 4.1

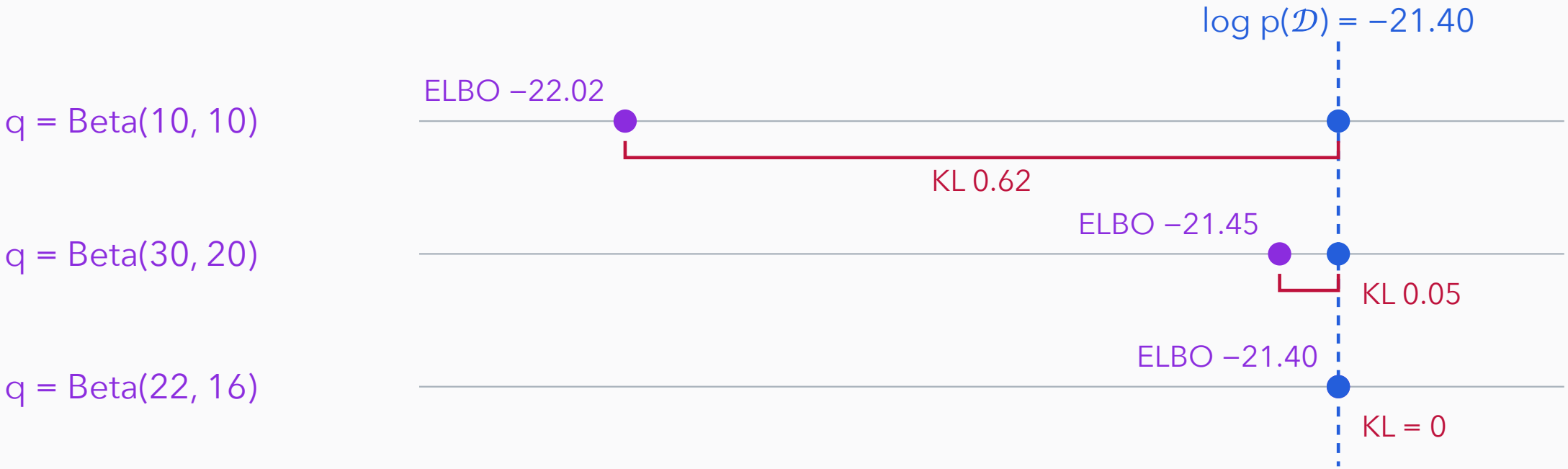
Raise the ELBO, and the KL falls by exactly as much



Three q's for the same coin. log p(𝒟) does not move. As q improves, the ELBO climbs and the gap, the KL, shrinks by exactly the same amount.

▶ Notebook 4.1

Raise the ELBO, and the KL falls by exactly as much



$$\max_q \ \mathrm{ELBO}(q) \quad \Longleftrightarrow \quad \min_q \ \mathrm{KL}\big(q \,\|\, p(\theta \mid \mathcal{D})\big)$$

> So maximising the ELBO, which we can compute, minimises the KL to the posterior, which we cannot. At the best q the gap is 0.

▶ Notebook 4.1

The ELBO in two terms: fit the data, stay near the prior

$$\mathrm{ELBO}(q) = \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big]$$

Take the ELBO apart.

▶ Notebook 4.1

The ELBO in two terms: fit the data, stay near the prior

$$\mathrm{ELBO}(q) = \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big]$$

$$= \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta)\big] + \mathbb{E}_q\big[\log p(\theta) - \log q(\theta)\big]$$

> The average of a sum is the sum of the averages: a likelihood term and the rest.

▶ Notebook 4.1

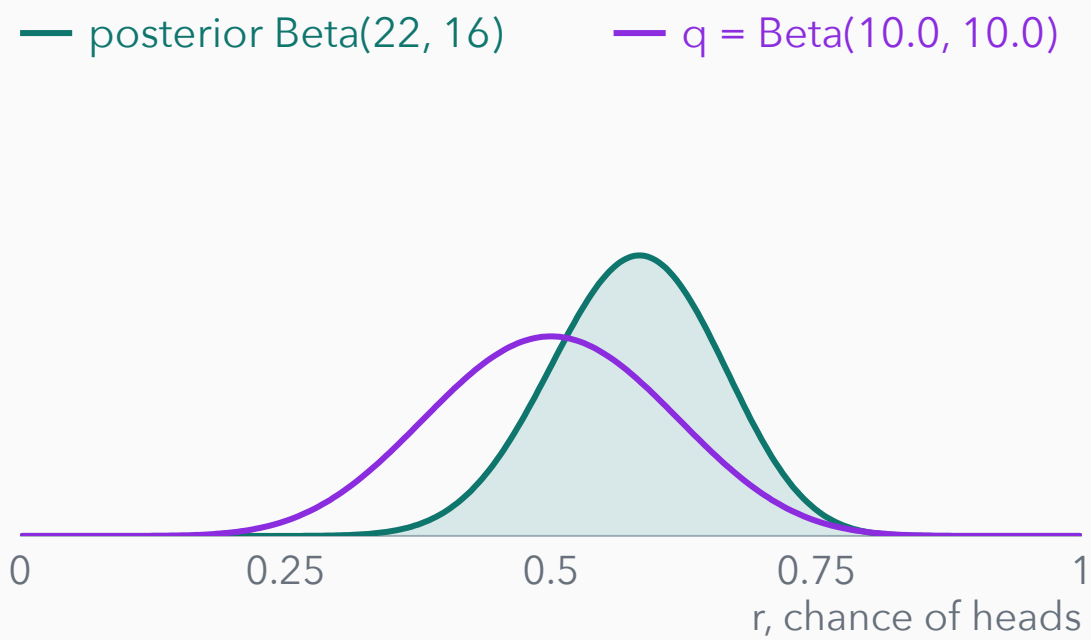
The ELBO in two terms: fit the data, stay near the prior

$$\begin{aligned}\mathrm{ELBO}(q) &= \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta) + \log p(\theta) - \log q(\theta)\big] \\ &= \mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta)\big] + \mathbb{E}_q\big[\log p(\theta) - \log q(\theta)\big] \\ &= \underbrace{\mathbb{E}_q\big[\log p(\mathcal{D} \mid \theta)\big]}_{\text{fit the data}} - \underbrace{\mathrm{KL}\big(q(\theta) \,\|\, p(\theta)\big)}_{\text{stay near the prior}}\end{aligned}$$

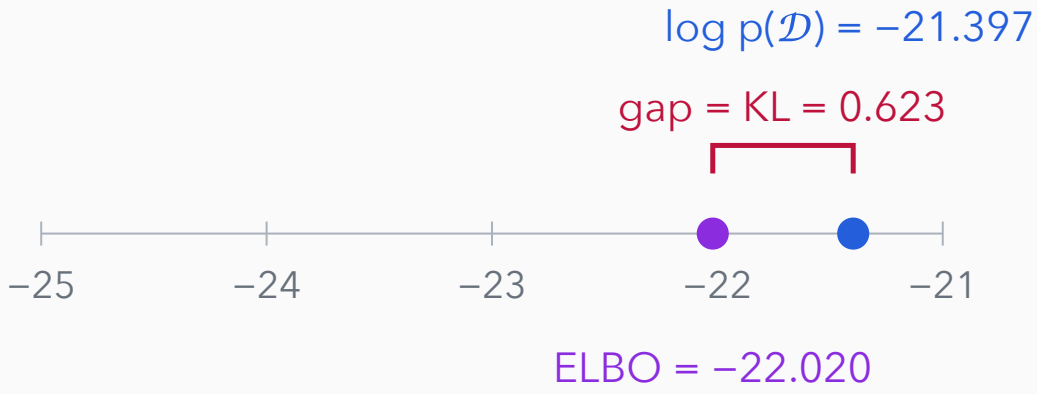
> The rest is minus a KL, from q to the prior. Fit the data, but stay near the prior. For our network the second term is the diagonal-Gaussian formula of 2.6; the first needs samples.

▶ Notebook 4.2

Check the identity where p($\mathcal{D}$) is known: a coin



One fixed total



−22.020 + 0.623 = −21.397

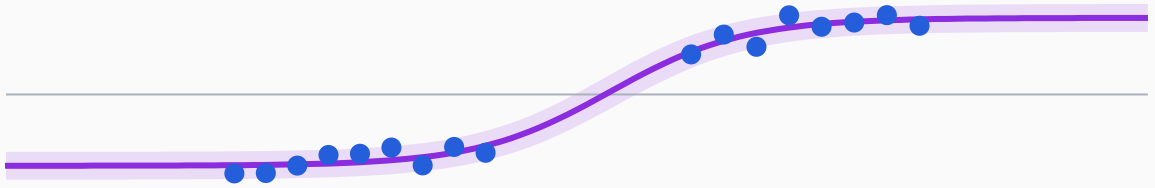
A coin with a Beta(5, 3) prior and 17 heads in 30 tosses has an exact posterior and an exact p($\mathcal{D}$). Drag q: the ELBO and the KL trade off, and their sum never changes.

▶ Notebook

Part 5 · Implement it


```
q = Normal(mu, rho.exp())
theta = q.rsample()
elbo = log_lik(theta) + log_prior(theta)
       - q.log_prob(theta).sum()
(-elbo).backward(); opt.step()
```

the network's predictions, with error bars



5.1 VI for the tiny network ▶

5.2 Wrap-up ▶

> Part 5 puts it in code: VI for our network, in a few lines of PyTorch, and its predictions with error bars against the MCMC reference.

▶ Notebook 5.1

The ELBO for our network, in code

```
mu  = Parameter(0.5 * randn(7))          # 7 means
rho = Parameter(full((7,), log(0.1)))    # 7 log-widths
q = Normal(mu, rho.exp())                # one bell per parameter
theta = q.rsample((16,))                 # 16 sampled networks
log_lik = log_likelihood(theta).mean()   # forward pass
kl = kl_divergence(q, prior).sum()       # closed form
loss = -(log_lik - kl)                   # minus the ELBO
```

Fourteen parameters, sixteen sampled networks per step, and the closed-form KL to the prior.

▶ Notebook 5.1

The ELBO for our network, in code

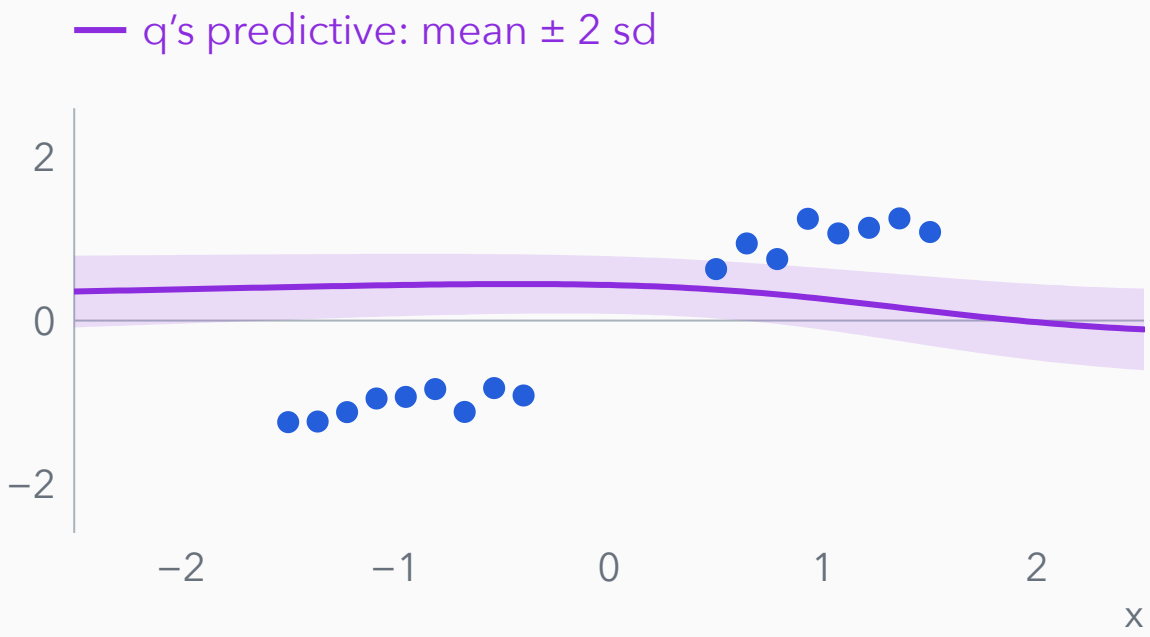
```
mu  = Parameter(0.5 * randn(7))          # 7 means
rho = Parameter(full((7,), log(0.1)))    # 7 log-widths
q = Normal(mu, rho.exp())                # one bell per parameter
theta = q.rsample((16,))                 # 16 sampled networks
log_lik = log_likelihood(theta).mean()   # forward pass
kl = kl_divergence(q, prior).sum()       # closed form
loss = -(log_lik - kl)                   # minus the ELBO
```

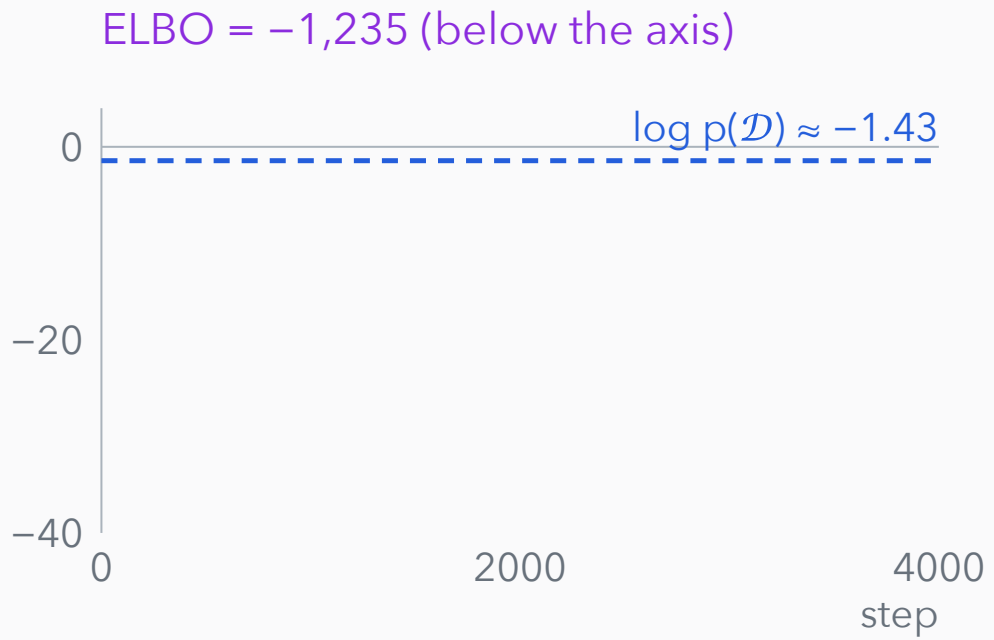
Adam, learning rate 0.02, 4,000 steps; best of five random starts. No p($\mathcal{D}$) anywhere.

> Every line is computable: rsample, a forward pass and a formula. Never p($\mathcal{D}$).

▶ Notebook 5.1

Press play: q learns the network

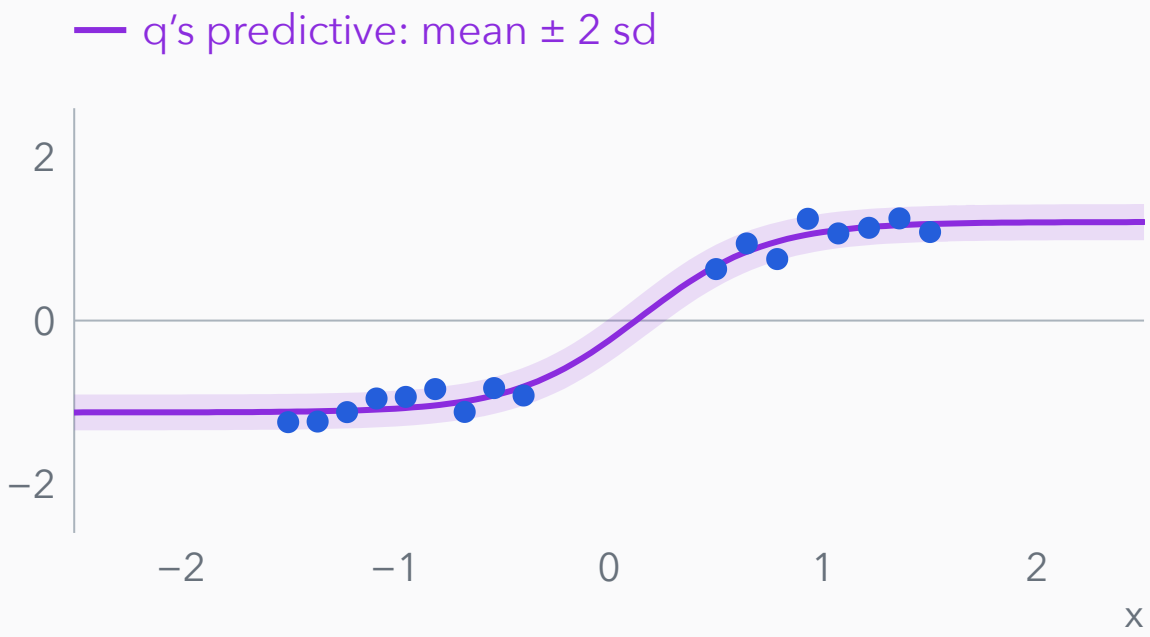


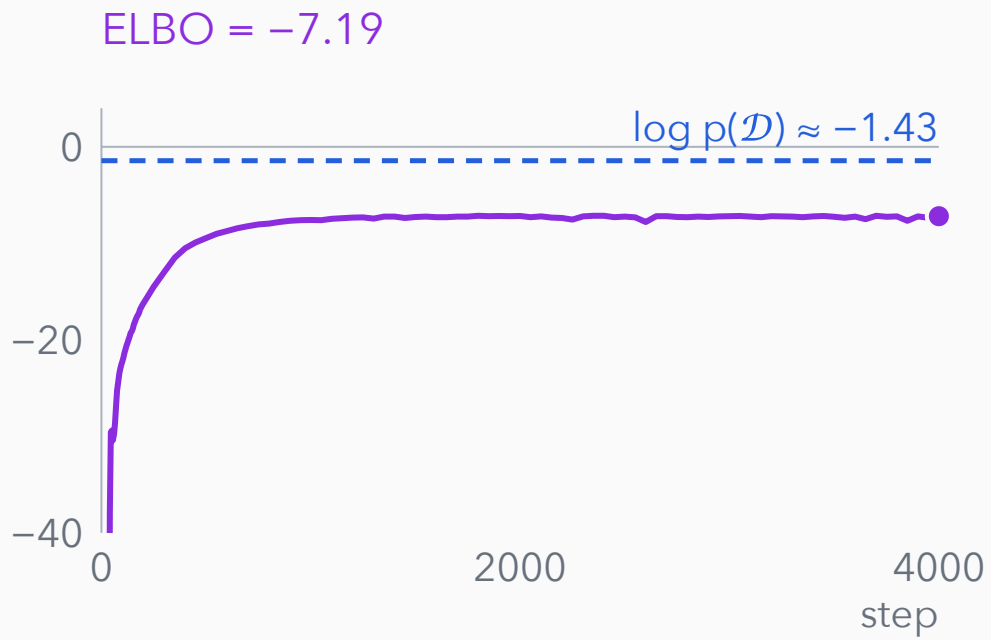


Start from random means and narrow widths. Watch q's predictive band and the ELBO.

▶ Notebook 5.1

Press play: q learns the network

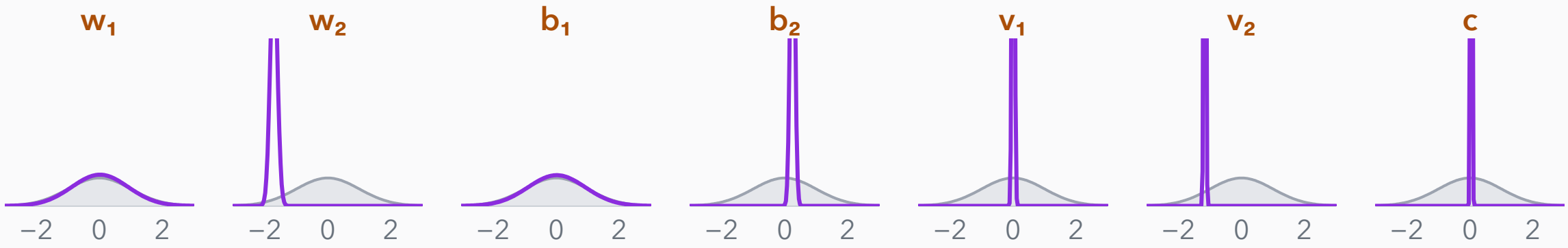




> The ELBO climbs from −1,235 to about −7.2 and the band settles onto the data. It stays below log p(𝒟) ≈ −1.43.

▶ Notebook 5.1

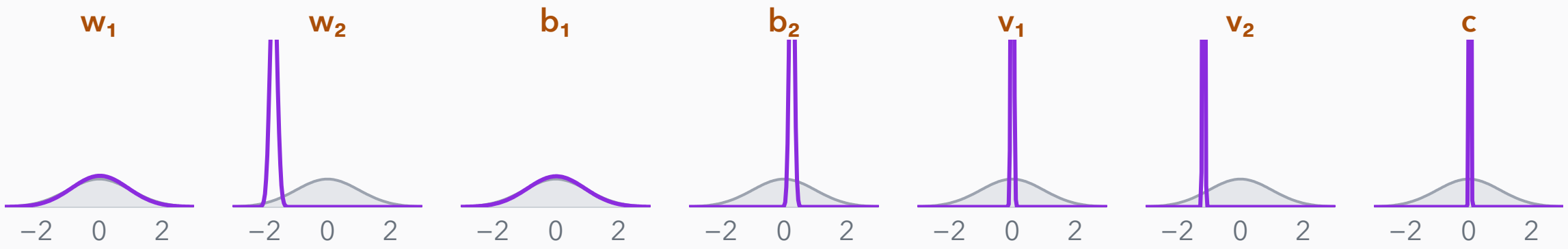
What q learned: one hidden unit switched off



The fitted bell for each parameter, against the N(0, 1) prior in grey.

▶ Notebook 5.1

What q learned: one hidden unit switched off



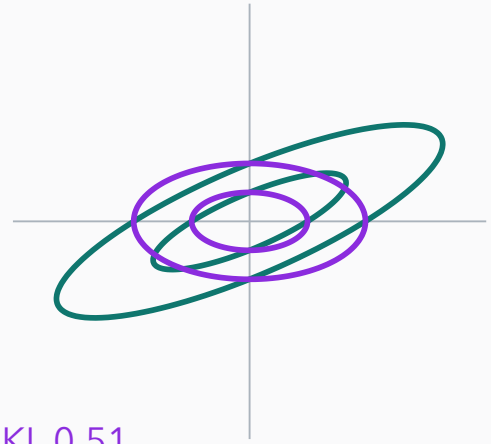
$v_1 \approx 0.01$ with width 0.035: the first hidden unit is switched off, and its w_1 and b_1 fall back to the prior.

> VI switched the first hidden unit off. Mean-field VI often prunes units like this: every parameter kept away from the prior costs KL.

▶ Notebook 5.1

Mean-field VI, and full-covariance VI

diagonal q (mean-field)



KL 0.51

full-covariance q



KL ≈ 0

$$\text{mean-field: } q(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j; \mu_j, \sigma_j^2)$$

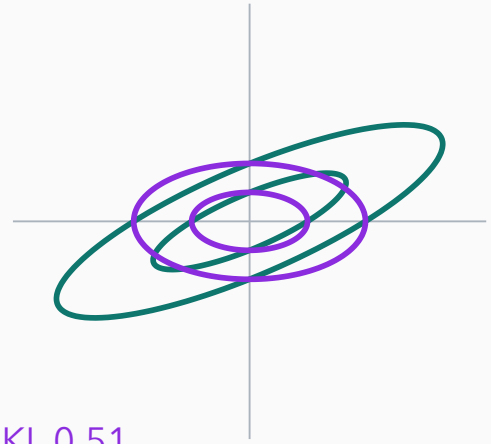
One independent bell per parameter: the diagonal family of 3.3. 7 + 7 = 14 numbers.

> Our q is mean-field: q factorises into one bell per parameter, so it cannot express correlations. In 3.3 that made a diagonal q shrink.

▶ Notebook 5.1

Mean-field VI, and full-covariance VI

diagonal q (mean-field)



KL 0.51

full-covariance q



KL ≈ 0

$$\text{mean-field: } q(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j; \mu_j, \sigma_j^2)$$

One independent bell per parameter: the diagonal family of 3.3. 7 + 7 = 14 numbers.

$$\text{full covariance: } q(\theta) = \mathcal{N}\left(\mu,\, LL^\top\right)$$

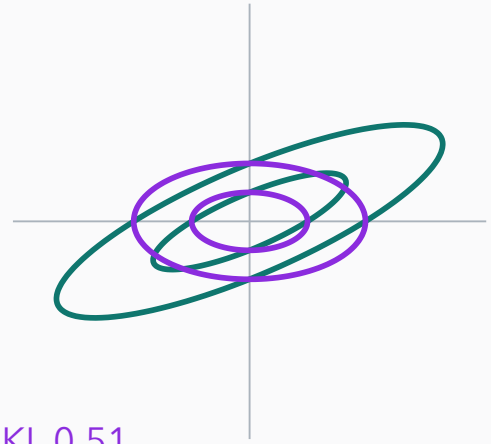
One tilted bell in $\mathbb{R}^7$: 7 + 28 = 35 numbers; d(d + 1)/2 in general.

> Full-covariance VI uses one tilted Gaussian over all seven parameters: Σ = LLᵀ, 35 numbers instead of 14. Fine for 7 parameters; hopeless for a million.

▶ Notebook 5.1

Mean-field VI, and full-covariance VI

diagonal q (mean-field)



KL 0.51

full-covariance q



KL ≈ 0

$$\text{mean-field: } q(\theta) = \prod_{j=1}^{7} \mathcal{N}(\theta_j; \mu_j, \sigma_j^2)$$

One independent bell per parameter: the diagonal family of 3.3. 7 + 7 = 14 numbers.

$$\text{full covariance: } q(\theta) = \mathcal{N}\left(\mu,\, LL^\top\right)$$

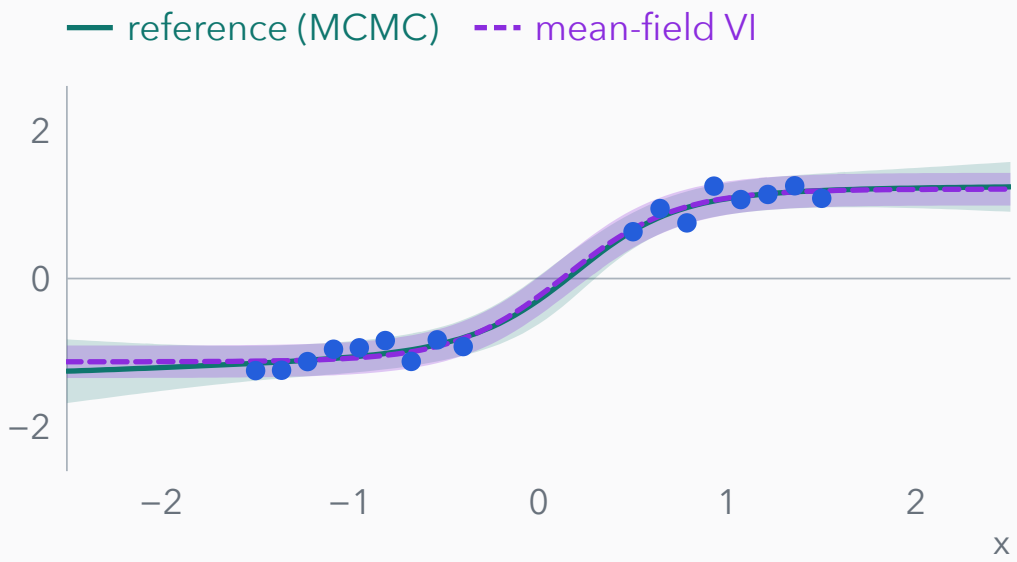
One tilted bell in $\mathbb{R}^7$: 7 + 28 = 35 numbers; d(d + 1)/2 in general.

Full q: correlation −0.76 between w_2 and v_2; ELBO −7.18 → −6.32.

> Starting from the mean-field answer, it learns that a steeper w_2 goes with a smaller v_2, and the ELBO rises by 0.86 nats: the KL to the posterior falls by exactly that much.

▶ Notebook 5.1

Against the reference: the right curve, bands still too narrow

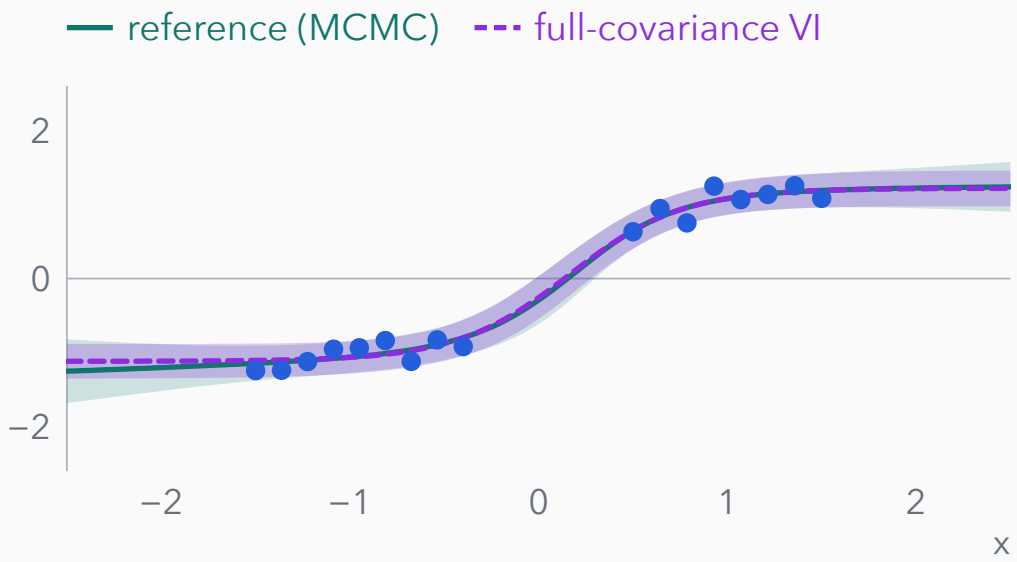


	ELBO	sd at −2.5	sd at 2.5
mean-field	−7.18	0.110	0.110
MCMC reference	$\log p(\mathcal{D}) \approx -1.43$	0.215	0.168

Mean-field VI gets the mean curve right, but its bands are too narrow beyond the data: the diagonal q of 3.3 shrinking, in seven dimensions.

▶ Notebook 5.1

Against the reference: the right curve, bands still too narrow



	ELBO	sd at −2.5	sd at 2.5
mean-field	−7.18	0.110	0.110
full covariance	−6.32	0.117	0.120
MCMC reference	$\log p(\mathcal{D}) \approx -1.43$	0.215	0.168

> Full covariance helps a little. The rest of the gap is the family: the posterior has several modes (swap the hidden units, flip signs), and one Gaussian sits on one.

▶ Notebook 5.2

Six obstacles, six tools

obstacle	tool	where
$p(\mathcal{D})$ is an impossible integral	approximate the posterior with q	1.4 – 1.5
“close” needs a number	KL(q ‖ p)	2.1 – 2.4
a grid cannot search 14 numbers	gradient descent	3.1 – 3.2
no formula for the average	Monte Carlo	2.5, 3.4
no gradient through a draw	$\theta = \mu + \sigma\varepsilon$, rsample()	3.5
KL to the posterior needs $p(\mathcal{D})$	maximise the ELBO	4.1

> Every obstacle today earned a tool, and our network used them all. In Lecture 2 a VAE does the same with one q per image.

▶ Notebook 5.2

Generative models create this inference problem again

A hidden description can generate an observation.

> In an image model, a hidden code z can generate an observed image x, just as the mixture's bump z generated θ in 3.4.

▶ Notebook 5.2

Generative models create this inference problem again

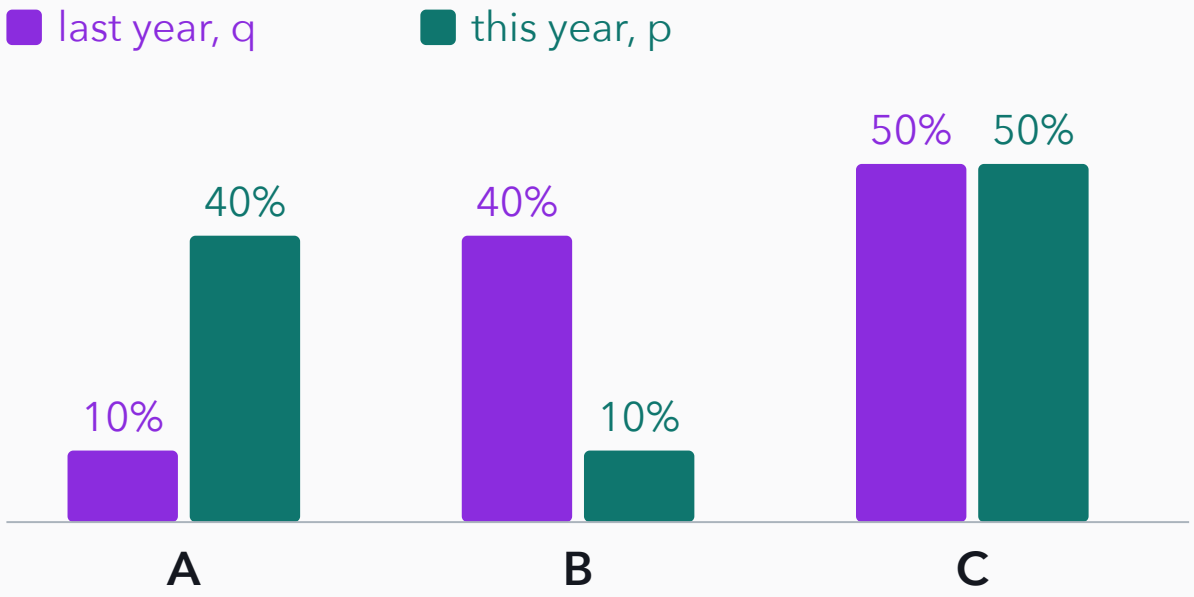
A hidden description can generate an observation.

Next: learn representations for images, then ask whether they can generate new ones.

> Lecture 2 builds autoencoders, then returns to VI with an encoder that outputs q for each image: the VAE.

▶ Notebook

By hand: two runs of this course, two grade distributions



Appendix: KL by hand. A second route to the same formula.

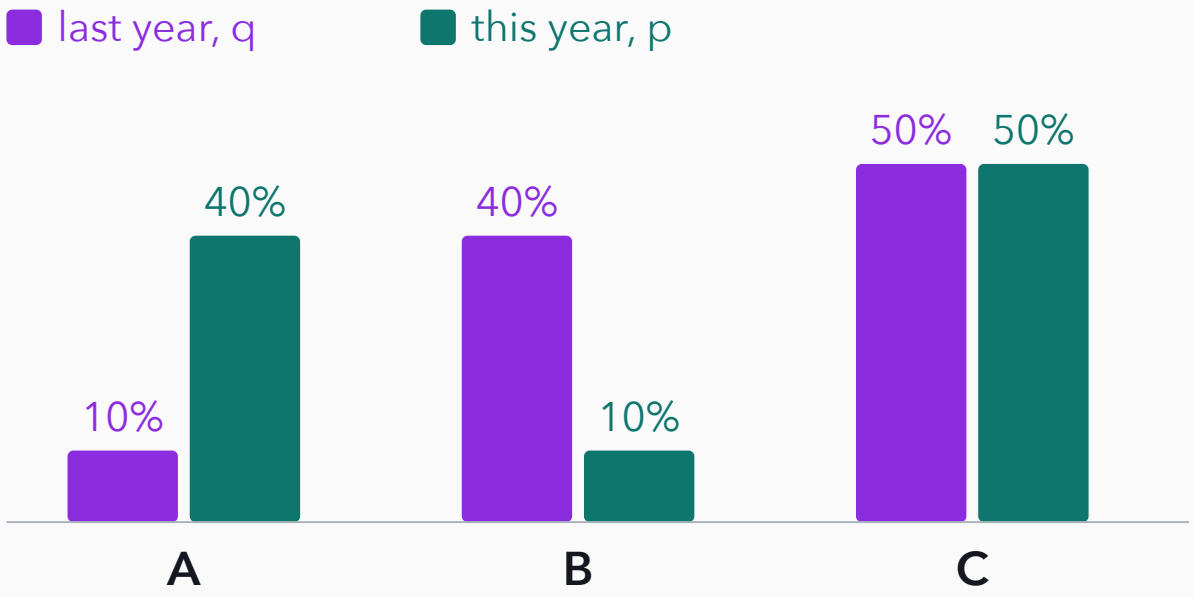
Grades in last year's run (q) and this year's (p).

> A second route to KL, by hand: the grade distributions of two runs of this course. How different are they, as one number?

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

▶ Notebook

By hand: two runs of this course, two grade distributions



Appendix: KL by hand. A second route to the same formula.

Grades in last year's run (q) and this year's (p).

A: 10% → 40%. B: 40% → 10%. C: 50% both years.

> Four times as many A's, a quarter as many B's, the same share of C's.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

Compare grade by grade: ratios

grade	p	q	p/q
A	0.4	0.1	4
B	0.1	0.4	¼
C	0.5	0.5	1



> How many times more common is each grade this year? A: 4. B: ¼. C: 1.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

Compare grade by grade: ratios

grade	p	q	p/q
A	0.4	0.1	4
B	0.1	0.4	¼
C	0.5	0.5	1



Average the ratios? (4 + ¼ + 1) / 3 = 1.75. Identical years would give 1, not 0, and ×4 sits 3 above 1 while ÷4 sits only 0.75 below.

> Averaging raw ratios is lopsided: a fourfold rise counts four times as much as a fourfold fall.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

What should a ratio count for? Find the function

We want a function f of the ratio r with:

$$f(1) = 0$$

$$f(4) = +y, \qquad f(\tfrac{1}{4}) = -y$$

$$f(2 \cdot 2) = f(2) + f(2)$$

$f(r)$	$f(\frac{1}{4})$	$f(1)$	$f(4)$	$f(2) + f(2)$

> What should a ratio count for? No change should count 0; a fourfold rise and a fourfold fall should cancel; and changes should add up.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

What should a ratio count for? Find the function

We want a function f of the ratio r with:

$$f(1) = 0$$

$$f(4) = +y, \qquad f(\tfrac{1}{4}) = -y$$

$$f(2 \cdot 2) = f(2) + f(2)$$

$f(r)$	$f(\frac{1}{4})$	$f(1)$	$f(4)$	$f(2) + f(2)$
$r - 1$	−0.75	0	3	2 ≠ 3 ✗
$r - \frac{1}{r}$	−3.75	0	3.75	3 ≠ 3.75 ✗

> Try a few functions. r − 1 is lopsided. r − 1/r balances ×4 against ÷4, but two doublings do not add up to a quadrupling.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

What should a ratio count for? Find the function

We want a function f of the ratio r with:

$$f(1) = 0$$

$$f(4) = +y, \qquad f(\tfrac{1}{4}) = -y$$

$$f(2 \cdot 2) = f(2) + f(2)$$

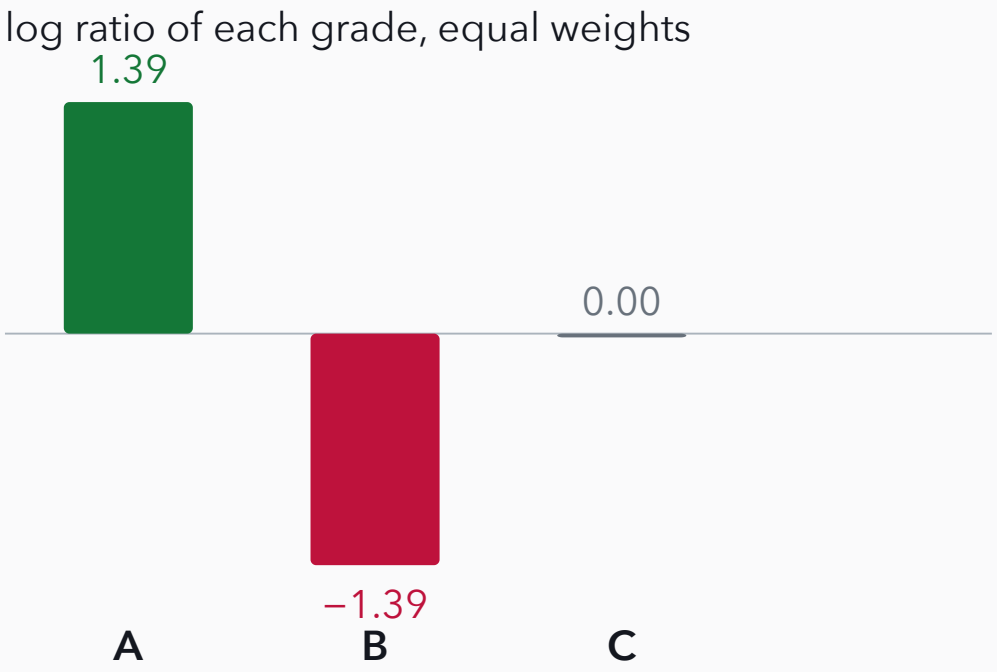
$f(r)$	$f(\frac{1}{4})$	$f(1)$	$f(4)$	$f(2) + f(2)$
$r - 1$	−0.75	0	3	2 ≠ 3 ✗
$r - \frac{1}{r}$	−3.75	0	3.75	3 ≠ 3.75 ✗
$\log r$	−1.39	0	1.39	1.39 = 1.39 ✓

> The logarithm passes every test: log 4 = 1.39, log ¼ = −1.39, and log 2 + log 2 = log 4.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

▶ Notebook

Weight each grade by this year's share: that is KL



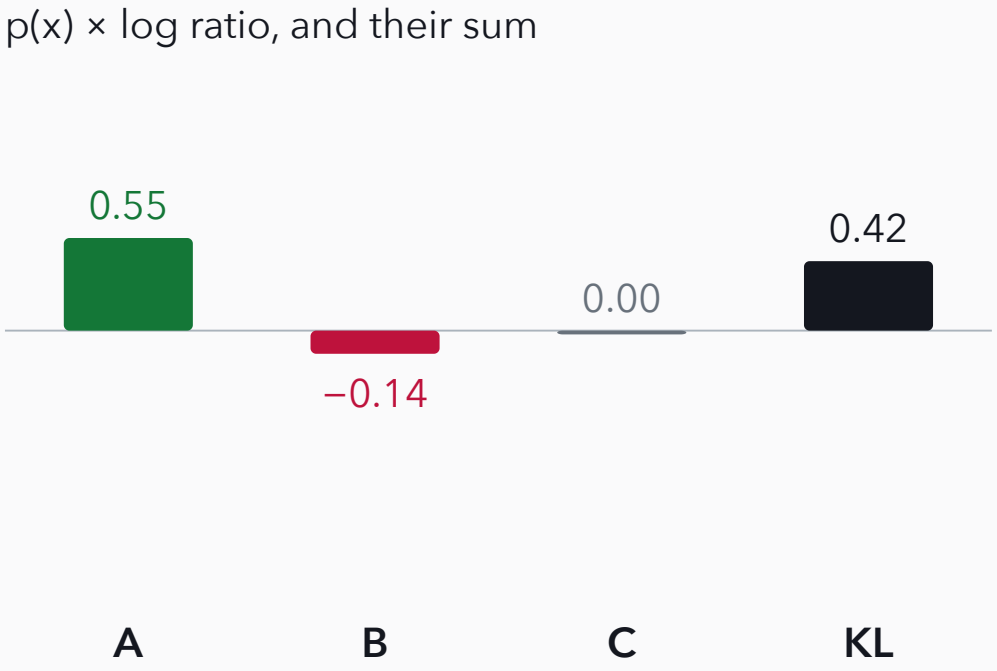
$$\text{plain average: } \tfrac{1}{3}\left(1.39 - 1.39 + 0\right) = 0$$

> Why not simply average the logs? Here the plain average is exactly 0, yet the two years clearly differ.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

▶ Notebook

Weight each grade by this year's share: that is KL



$$\text{plain average: } \tfrac{1}{3}\left(1.39 - 1.39 + 0\right) = 0$$

$$\text{weighted: } 0.4(1.39) + 0.1(-1.39) + 0.5(0) = 0.42$$

> Weight each grade by how many students got it this year: the A's, now common, count most. The weighted sum is 0.42.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

▶ Notebook

Weight each grade by this year's share: that is KL

p(x) × log ratio, and their sum



$$\text{plain average: } \tfrac{1}{3}\left(1.39 - 1.39 + 0\right) = 0$$

$$\text{weighted: } 0.4(1.39) + 0.1(-1.39) + 0.5(0) = 0.42$$

$$\mathrm{KL}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

That weighted average of log ratios is the Kullback–Leibler divergence of p from q.

KL intuition after ritvikmath, "The KL Divergence: Data Science Basics" (YouTube)

▶ Notebook

The unit: nats, or bits

$$\mathrm{KL}(p\|q) = \sum_x p(x)\,\ln\frac{p(x)}{q(x)} = 0.42 \text{ nats}$$

$$\sum_x p(x)\,\log_2\frac{p(x)}{q(x)} = 0.60 \text{ bits}$$

> The log's base sets the unit. Natural log: nats. Log base 2: bits. Here 0.42 nats is exactly 0.60 bits; 1 nat ≈ 1.44 bits.

▶ Notebook

The unit: nats, or bits

$$\mathrm{KL}(p\|q) = \sum_x p(x)\,\ln\frac{p(x)}{q(x)} = 0.42 \text{ nats}$$

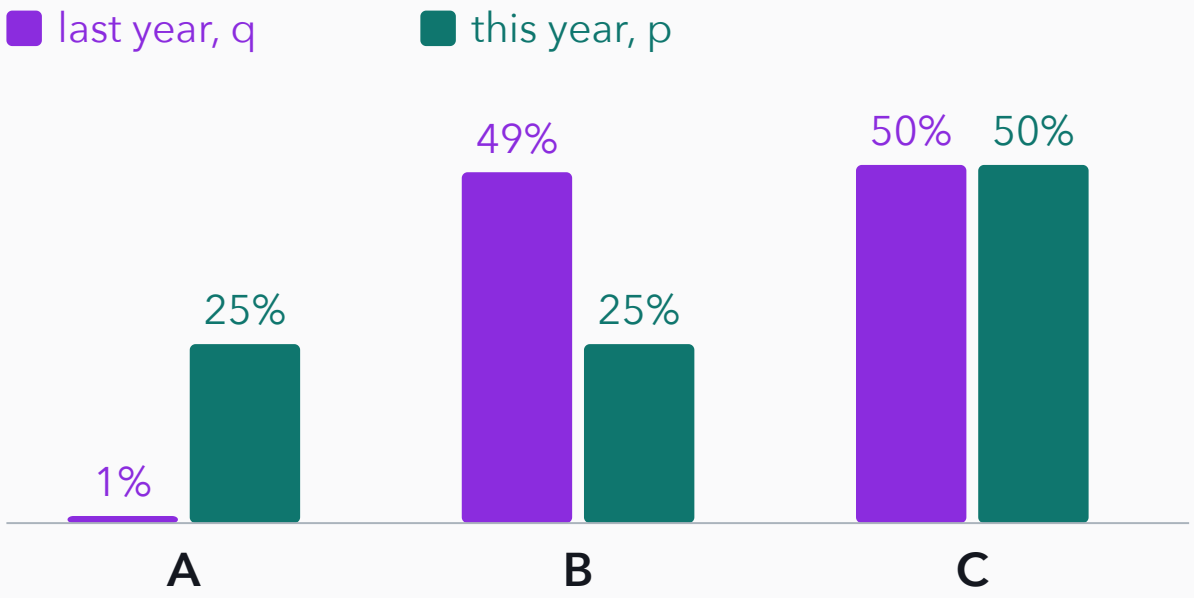
$$\sum_x p(x)\,\log_2\frac{p(x)}{q(x)} = 0.60 \text{ bits}$$

Read 0.42 nats as $e^{0.42} \approx 1.5$: on average, a grade from this year is 1.5 times more probable under this year's distribution than under last year's.

> 0.42 nats means a typical this-year grade is about 1.5 times more probable under this year's distribution than under last year's.

▶ Notebook

Swap the reference year: a different number

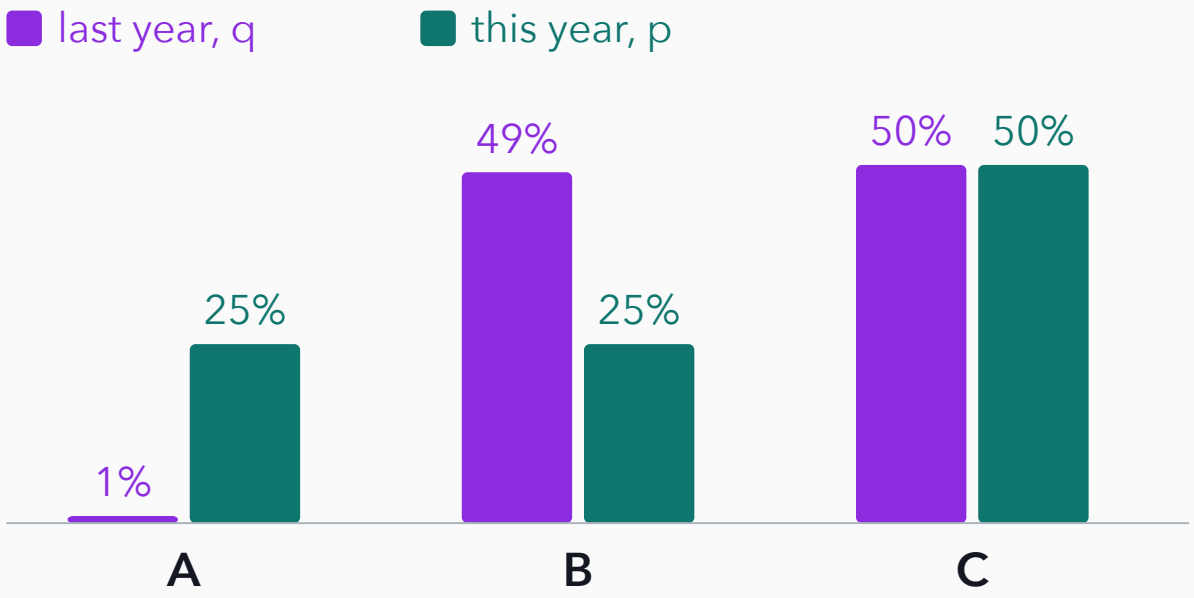


$$\mathrm{KL}(\text{this} \,\|\, \text{last}) = 0.64$$

A different pair of years: last year 1% got an A, this year 25% did. Measured from this year: 0.64 nats.

Notebook

Swap the reference year: a different number



$$\mathrm{KL}(\text{this} \,\|\, \text{last}) = 0.64$$

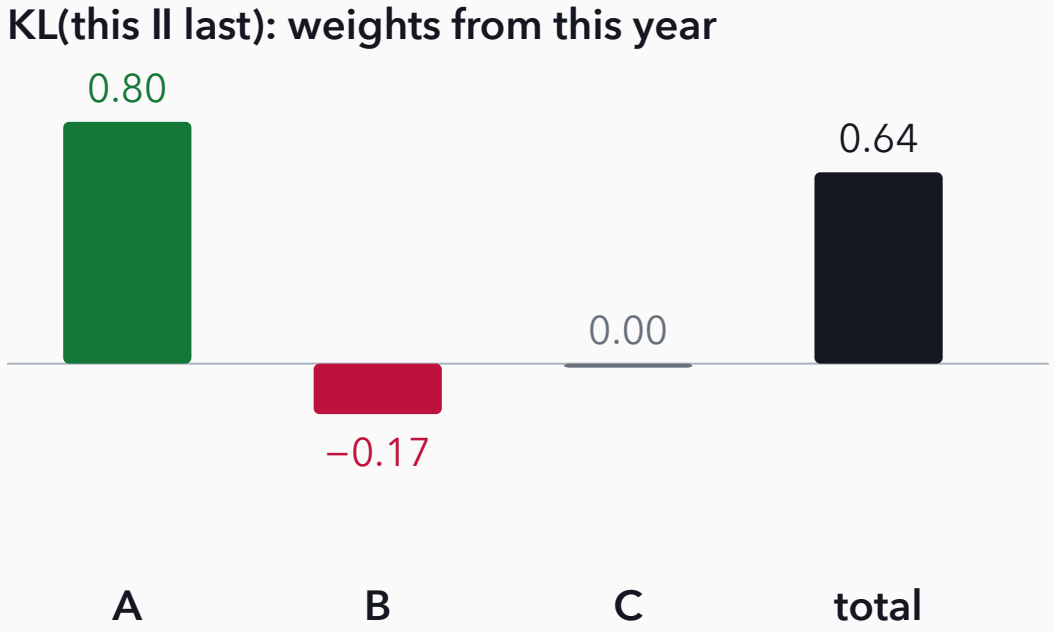
$$\mathrm{KL}(\text{last} \,\|\, \text{this}) = 0.30$$

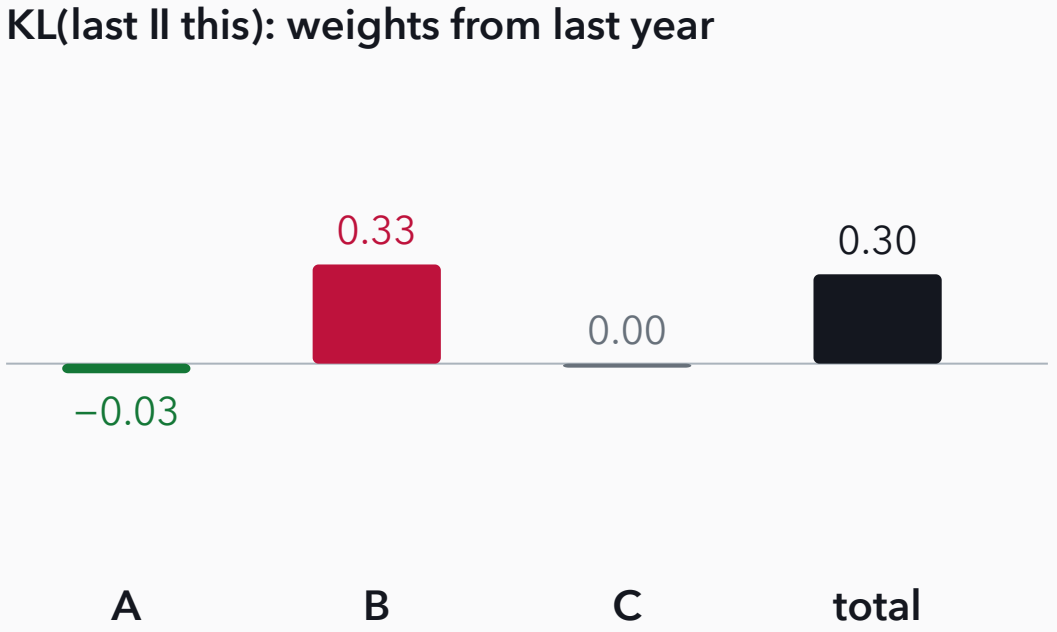
Which distribution comes first matters: KL is not symmetric.

> Measured from last year instead: 0.30 nats. Changing the reference changes the answer: KL is not symmetric.

Notebook

Why the reference matters

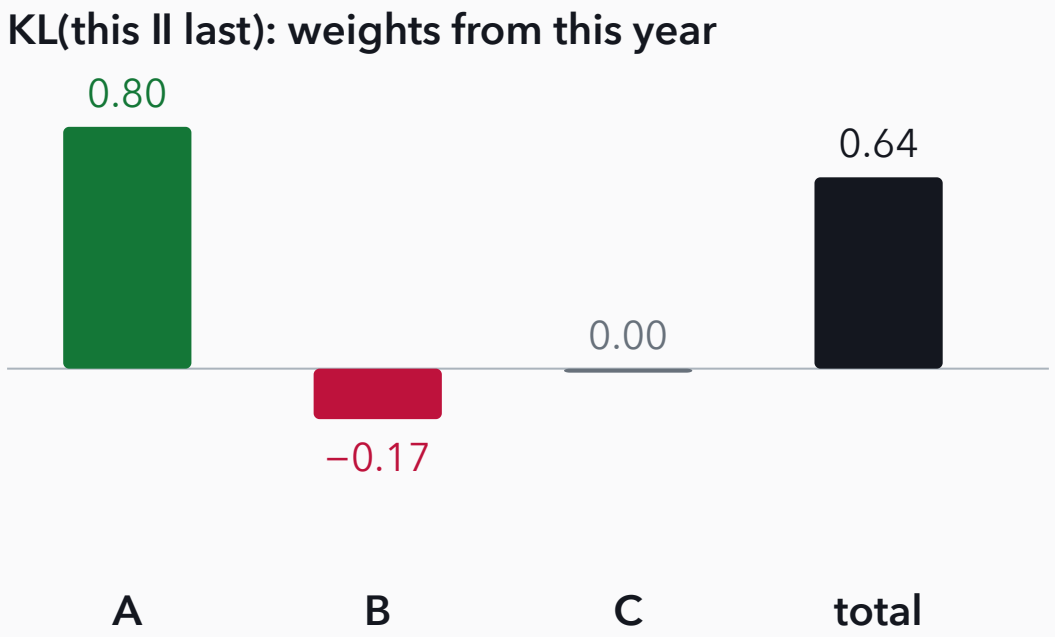


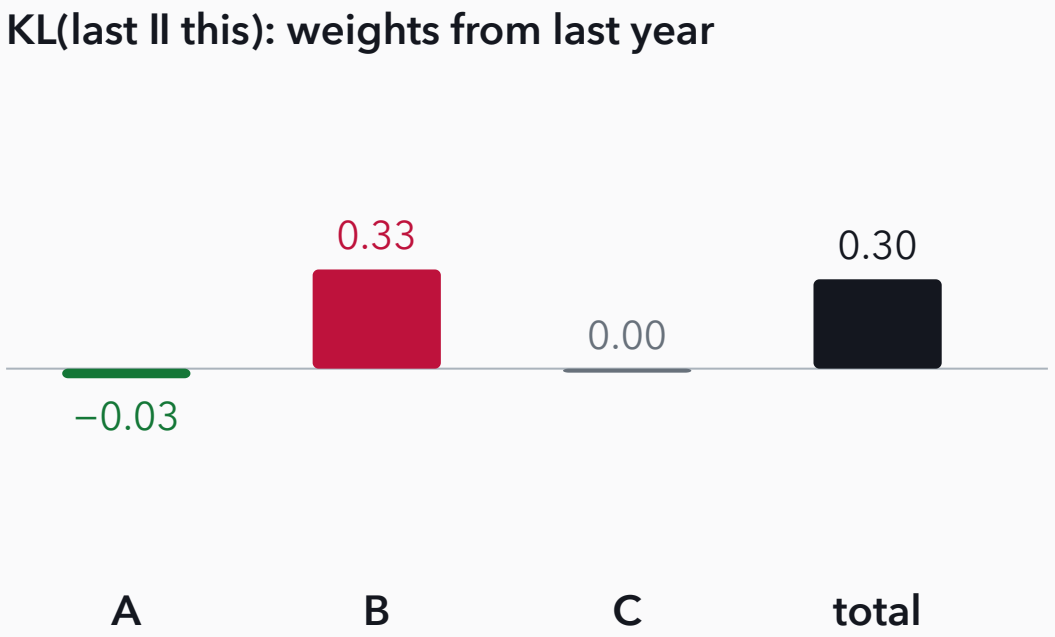


> From this year's view, A is 25 times more common than before, and a quarter of the class has it: A contributes 0.80.

▶ Notebook

Why the reference matters





The first distribution supplies the weights, so it decides which changes count.

> From last year's view A was rare, so its change barely counts (−0.03); the drop in B dominates instead.

▶ Notebook

Sample grades and watch the average find the KL

grades drawn from this year's p (latest 30)

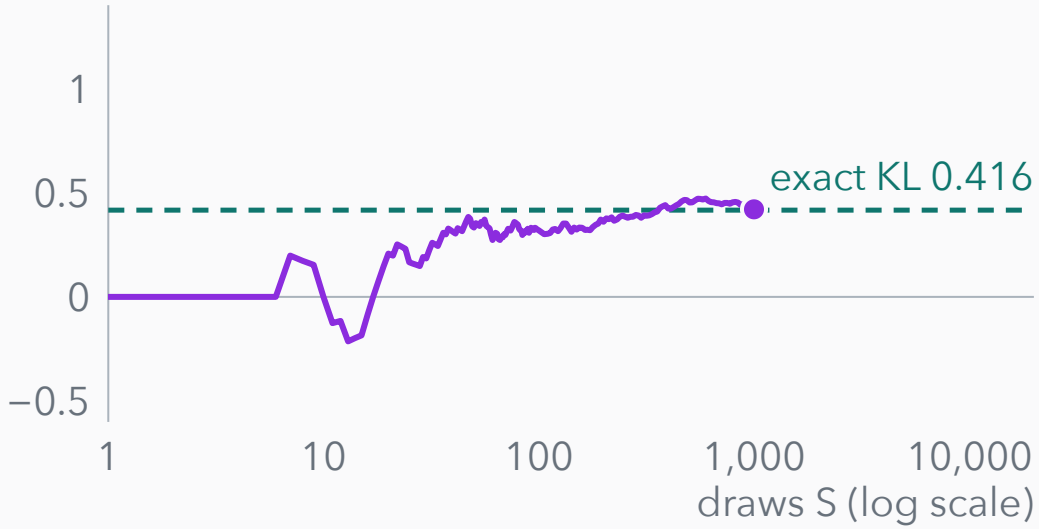
C	A	A	C	B	A	C	B	C	C
C	B	C	C	B	A	A	C	C	C
A	A	B	B	A	C	C	A	A	B

A: log ratio +1.39 — 411 draws, 41%

B: log ratio −1.39 — 108 draws, 11%

C: log ratio +0.00 — 481 draws, 48%

average of 1,000 log ratios: 0.420



> Draw grades from this year's distribution and average their log ratios. The counts approach 40%, 10%, 50%, and the average approaches the exact 0.42.

▶ Notebook

A KL estimator that is never negative

$$r = \frac{p(\theta)}{q(\theta)},\ \theta \sim q: \qquad \underbrace{-\log r}_{\text{plain}} \qquad \text{vs} \qquad \underbrace{r - 1 - \log r}_{\geq 0 \text{ for every draw}}$$

Why does adding r − 1 not change the average?

After Jia-Bin Huang and John Schulman, "Approximating KL divergence"

▶ Notebook

A KL estimator that is never negative

$$r = \frac{p(\theta)}{q(\theta)},\ \theta \sim q: \qquad \underbrace{-\log r}_{\text{plain}} \qquad \text{vs} \qquad \underbrace{r - 1 - \log r}_{\geq 0 \text{ for every draw}}$$

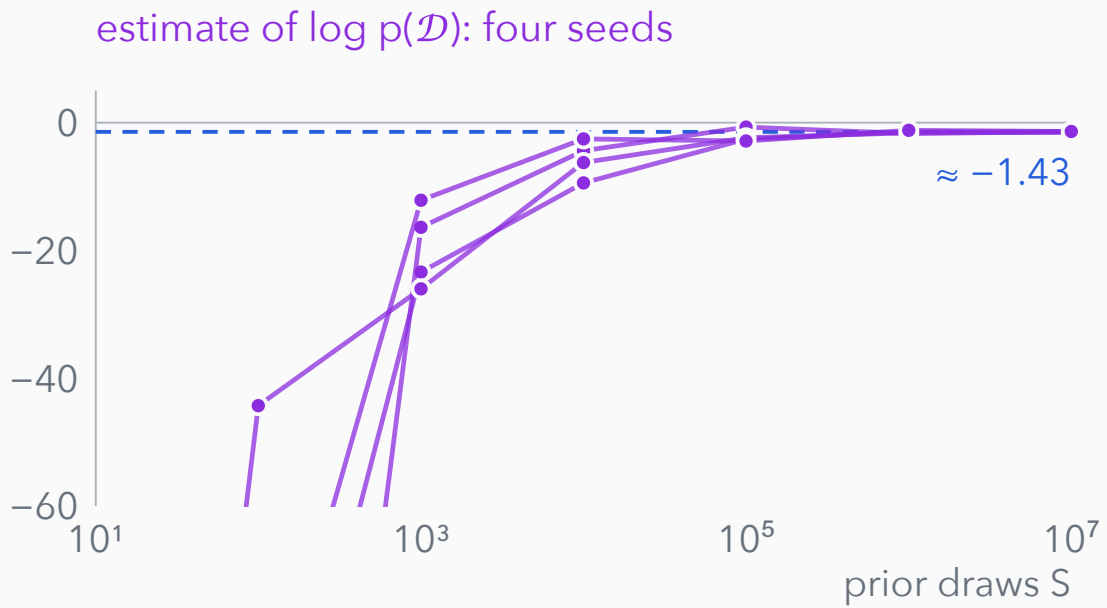
draws S	plain: spread	plain: negative	r − 1 − log r: spread	negative
5	0.299	34%	0.154	0%
10	0.219	24%	0.119	0%
100	0.072	0%	0.039	0%
1000	0.023	0%	0.013	0%

> When can the better estimator be worse?

After Jia-Bin Huang and John Schulman, "Approximating KL divergence"

▶ Notebook

Sampling from the prior: a few lucky draws carry everything



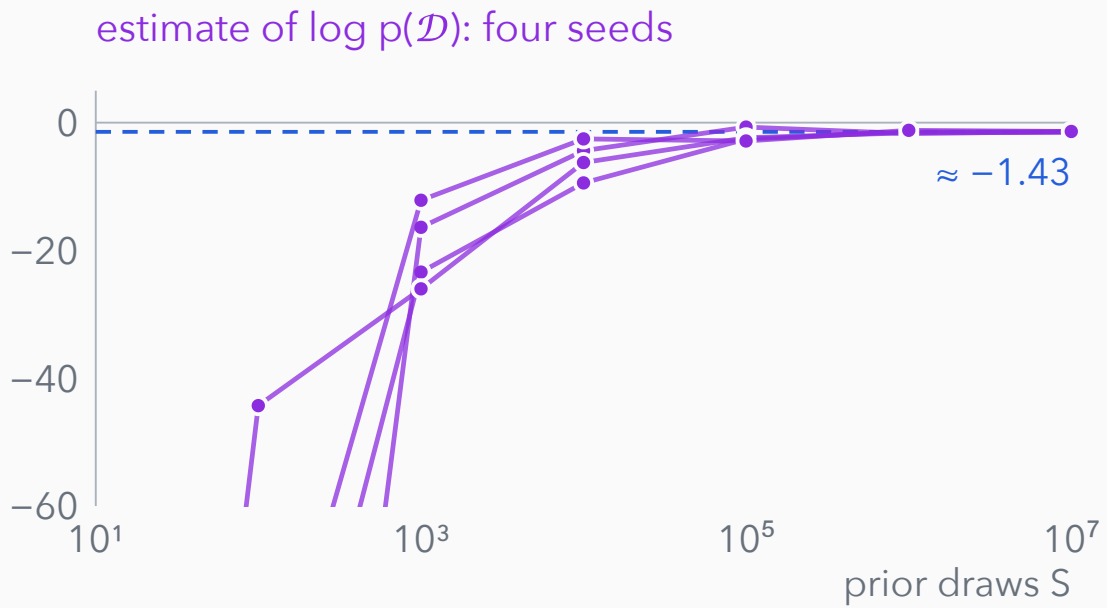
$$p(\mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(\mathcal{D} \mid \theta_s)$$

$$\theta_s \sim p(\theta)$$

Average the likelihood over draws from the prior: four seeds, up to ten million draws each.

▶ Notebook

Sampling from the prior: a few lucky draws carry everything



$$p(\mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(\mathcal{D} \mid \theta_s)$$

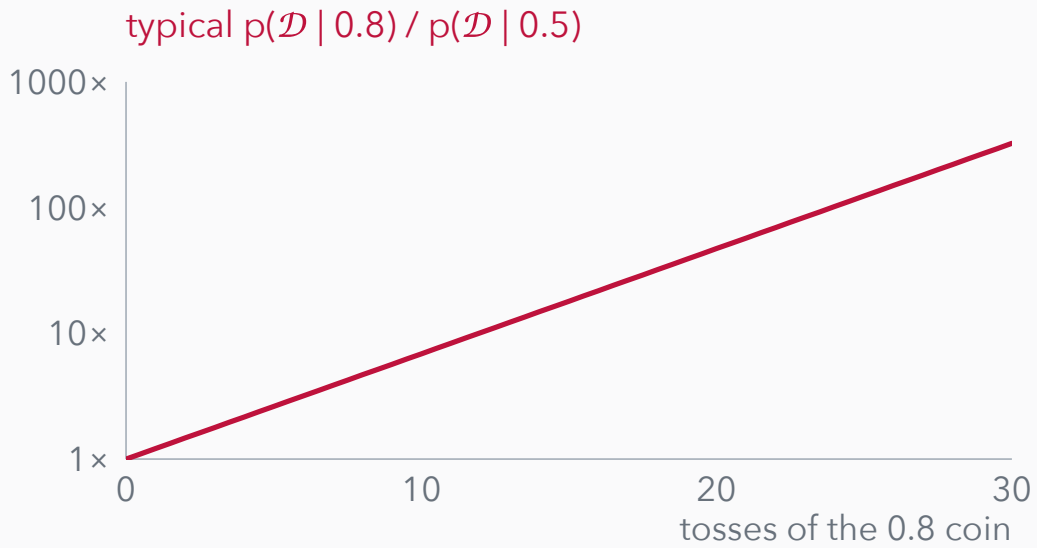
$$\theta_s \sim p(\theta)$$

Ten draws: off by about 1,000 nats. Ten million: about 103 draws carry the estimate, −1.43.

> Almost every prior draw explains the data terribly. Seven parameters just about work; more parameters or more data make it hopeless.

▶ Notebook

A KL value is how fast evidence piles up

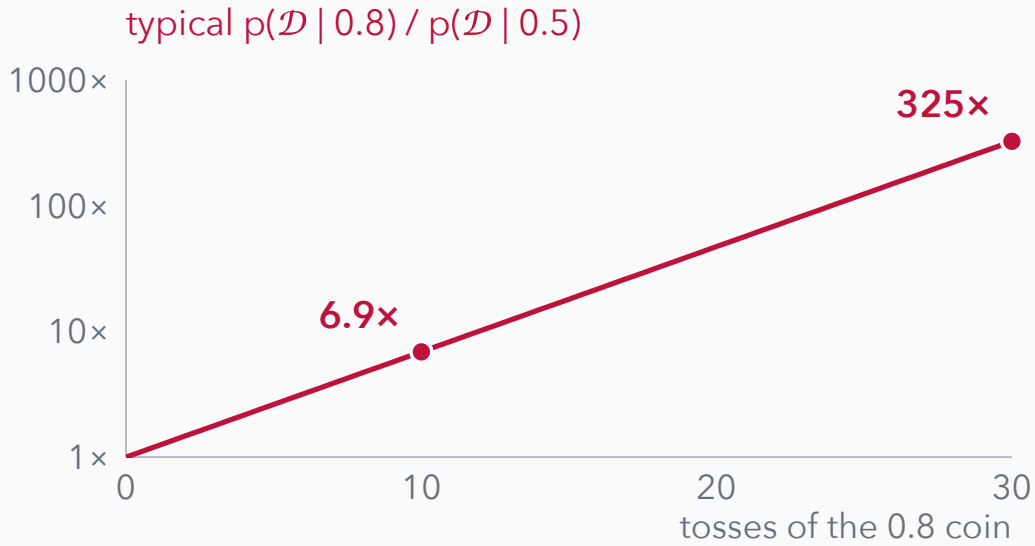


$$\mathrm{KL}\big(\mathrm{Bern}(0.8) \,\|\, \mathrm{Bern}(0.5)\big) \approx 0.19 \text{ nats}$$

> A coin lands heads 80% of the time. Each toss favours “0.8” over “fair” by 0.19 nats on average. After 30 tosses, by what factor?

▶ Notebook

A KL value is how fast evidence piles up



$$\mathrm{KL}\big(\mathrm{Bern}(0.8) \,\|\, \mathrm{Bern}(0.5)\big) \approx 0.19 \text{ nats}$$

$$\frac{p(\mathcal{D} \mid 0.8)}{p(\mathcal{D} \mid 0.5)} \approx e^{0.19\,n}$$

The reverse direction, KL(0.5‖0.8) = 0.22, is a different number.

> About e^(30 × 0.19) ≈ 325. KL is the rate at which evidence piles up: 6.9× after 10 tosses, 325× after 30.

▶ Notebook

A sensor has an unknown constant bias

$$\theta \sim \mathcal{N}(0, 1), \qquad y_i \mid \theta \sim \mathcal{N}(\theta, 0.5^2)$$

A sensor reads a hidden bias θ, with noise. Name the unknown, the data and a q.

▶ Notebook

A sensor has an unknown constant bias

$$\theta \sim \mathcal{N}(0, 1), \qquad y_i \mid \theta \sim \mathcal{N}(\theta, 0.5^2)$$

$$p(\mathcal{D}, \theta) = p(\theta) \prod_i p(y_i \mid \theta)$$

> The joint is the prior times the likelihood of each reading: available.

▶ Notebook

A sensor has an unknown constant bias

$$\theta \sim \mathcal{N}(0, 1), \qquad y_i \mid \theta \sim \mathcal{N}(\theta, 0.5^2)$$

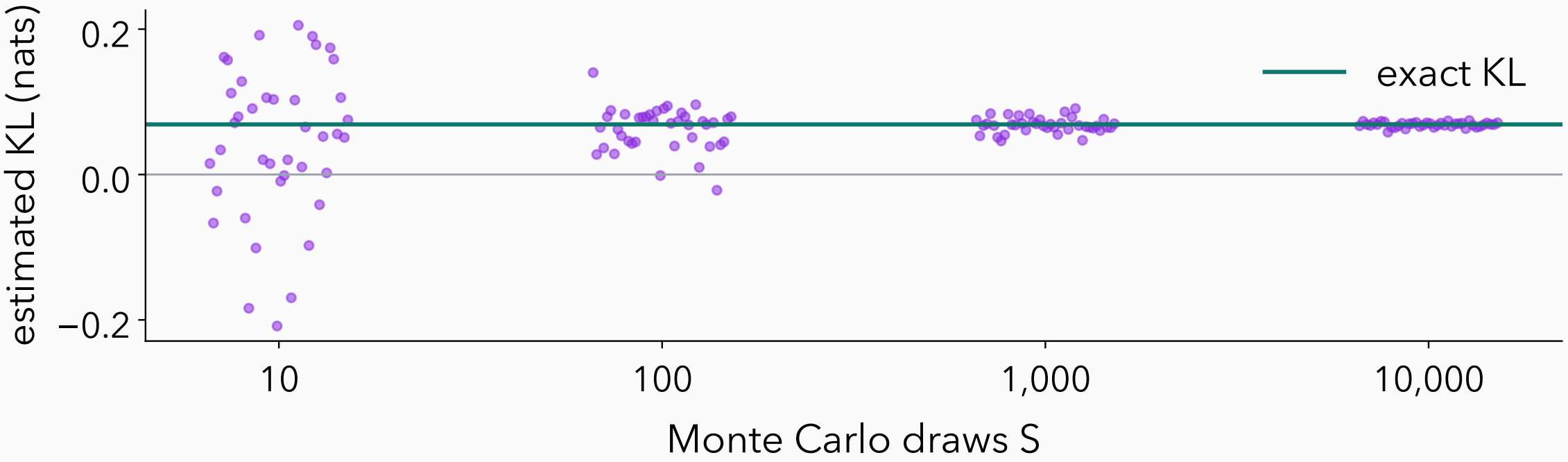
$$p(\mathcal{D}, \theta) = p(\theta) \prod_i p(y_i \mid \theta)$$

$$\max_{\psi} \mathbb{E}_{q_\psi(\theta)}[\log p(\mathcal{D}, \theta) - \log q_\psi(\theta)]$$

Maximise the ELBO over q's parameters. The same tools apply.

▶ Notebook

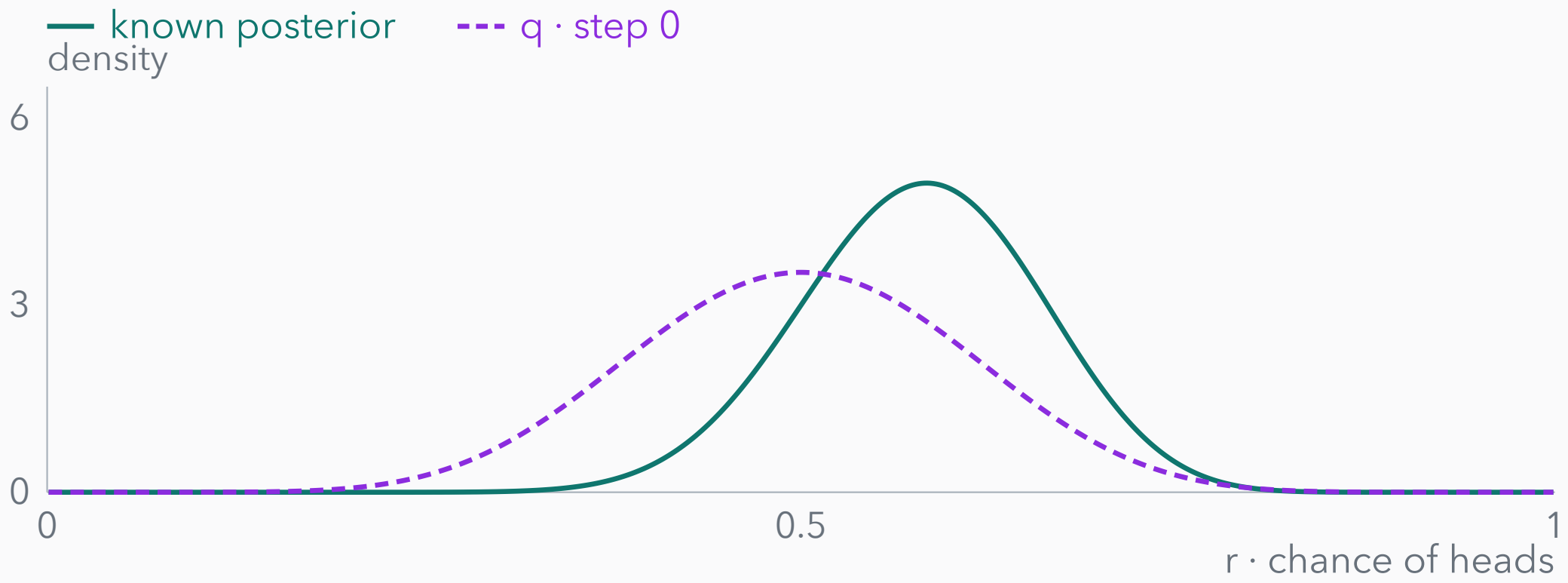
More samples, tighter estimates



Forty repeats at each sample size. The spread shrinks like $1/\sqrt{S}$; the exact value never moves.

▶ Notebook

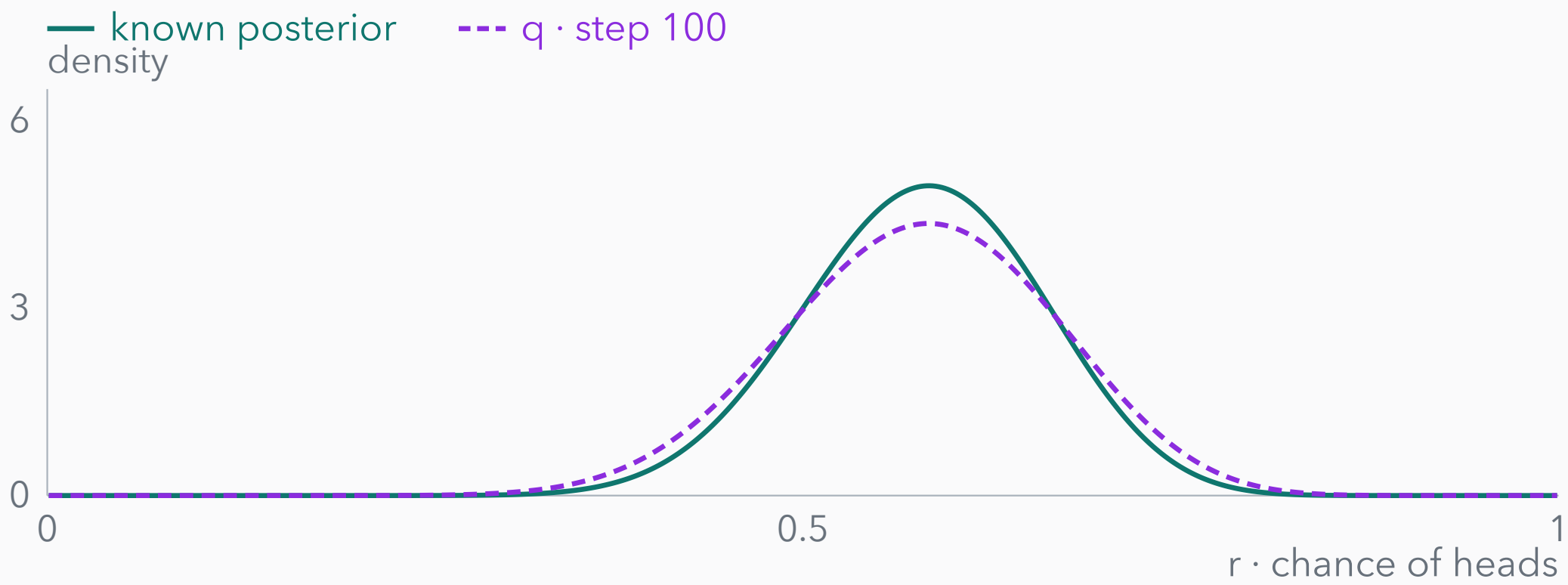
A check with an answer key: the Beta coin



With the Beta(5, 3) prior we know the posterior: Beta(22, 16). Fit a Beta q from Beta(10, 10) using only the joint.

▶ Notebook

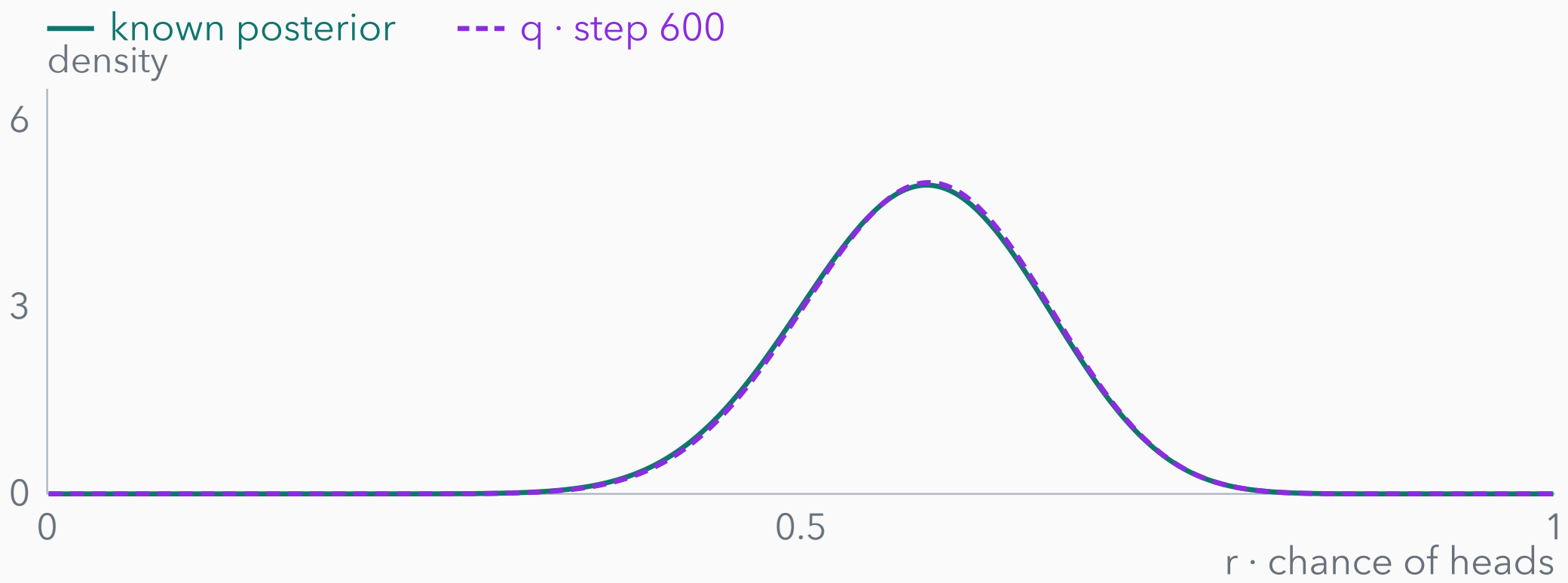
A check with an answer key: the Beta coin



By step 100, q has moved most of the way.

▶ Notebook

A check with an answer key: the Beta coin



After 600 steps: Beta(22.4, 16.2), with KL ≈ 0.0002 to the true posterior.

▶ Notebook

One literal update of the mean

$$\sigma = 2 \text{ fixed}: \qquad L(\mu) = \text{const} + \frac{(\mu - 1)^2}{32}$$

Which term of the Gaussian KL depends on μ?

▶ Notebook

One literal update of the mean

$$\sigma = 2 \text{ fixed}: \qquad L(\mu) = \text{const} + \frac{(\mu - 1)^2}{32}$$

$$\frac{\partial L}{\partial \mu} = \frac{\mu - 1}{16} = -0.0625 \ \text{ at } \mu = 0$$

Differentiate it at μ = 0.

▶ Notebook

One literal update of the mean

$$\sigma = 2 \text{ fixed}: \qquad L(\mu) = \text{const} + \frac{(\mu - 1)^2}{32}$$

$$\frac{\partial L}{\partial \mu} = \frac{\mu - 1}{16} = -0.0625 \ \text{ at } \mu = 0$$

$$\mu^+ = 0 - 4(-0.0625) = 0.25: \quad \text{KL } 0.349 \to 0.336$$

Take one step with η = 4. What is the best KL with σ stuck at 2?

▶ Notebook

KL ≥ 0 by Jensen's inequality

$$\mathrm{KL}(q\|p) = -\mathbb{E}_q\left[\log \frac{p(\theta)}{q(\theta)}\right]$$

Which function makes Jensen's inequality useful?

▶ Notebook

KL ≥ 0 by Jensen's inequality

$$\mathrm{KL}(q\|p) = -\mathbb{E}_q\left[\log \frac{p(\theta)}{q(\theta)}\right]$$

$$\geq -\log \mathbb{E}_q\left[\frac{p(\theta)}{q(\theta)}\right]$$

Why does the inequality point this way?

▶ Notebook

KL ≥ 0 by Jensen's inequality

$$\mathrm{KL}(q\|p) = -\mathbb{E}_q\left[\log \frac{p(\theta)}{q(\theta)}\right]$$

$$\geq -\log \mathbb{E}_q\left[\frac{p(\theta)}{q(\theta)}\right]$$

$$= -\log 1 = 0$$

What makes the final expectation one?

▶ Notebook

Five draws can give a negative KL estimate

θ_i	$\log q(\theta_i) - \log p(\theta_i)$
5.623	−0.232
0.120	0.269
−5.536	−0.751
2.705	0.217
−2.254	0.030

> Some rows are negative. Could their average be negative?

▶ Notebook

Five draws can give a negative KL estimate

θ_i	$\log q(\theta_i) - \log p(\theta_i)$
5.623	−0.232
0.120	0.269
−5.536	−0.751
2.705	0.217
−2.254	0.030

$$\widehat{\mathrm{KL}}_5 \approx -0.093, \qquad \mathrm{KL}(q\|p) \approx 0.069$$

> Does this contradict KL ≥ 0?

▶ Notebook

The two PML candidates, scored by Monte Carlo

candidate	MC estimate ± 1 s.e.	numerical integral
$N(3, 1^2)$	9.54 ± 0.03	9.49
$N(3, 4.5^2)$	45.02 ± 0.21	45.15

> How large is the difference relative to the sampling uncertainty?

▶ Notebook

The two PML candidates, scored by Monte Carlo

candidate	MC estimate ± 1 s.e.	numerical integral
$N(3, 1^2)$	9.54 ± 0.03	9.49
$N(3, 4.5^2)$	45.02 ± 0.21	45.15

The narrow candidate scores far better. Both sit away from the mixture's mass, so both are poor.

> What role does the numerical integral play?

▶ Notebook

A fixed-noise finite difference checks the path

```
q = Normal(mu, sigma)
theta = q.rsample((S,))
loss = (q.log_prob(theta) - p.log_prob(theta)).mean()
```

What is different about rsample?

▶ Notebook

A fixed-noise finite difference checks the path

```
q = Normal(mu, sigma)
theta = q.rsample((S,))
loss = (q.log_prob(theta) - p.log_prob(theta)).mean()
```

derivative	autograd	finite difference
μ	−0.062466	−0.062466
σ	−0.372223	−0.372223

> What must stay fixed on both sides of the finite difference?

Notebook

Pathwise gradients are one route, not the only one

$$\nabla_\psi \mathbb{E}_{q_\psi}[f_\psi(\theta)] = \mathbb{E}_{q_\psi}[f_\psi(\theta)\nabla_\psi \log q_\psi(\theta) + \nabla_\psi f_\psi(\theta)]$$

Which term was missing from the naive sample() gradient?

▶ Notebook

Pathwise gradients are one route, not the only one

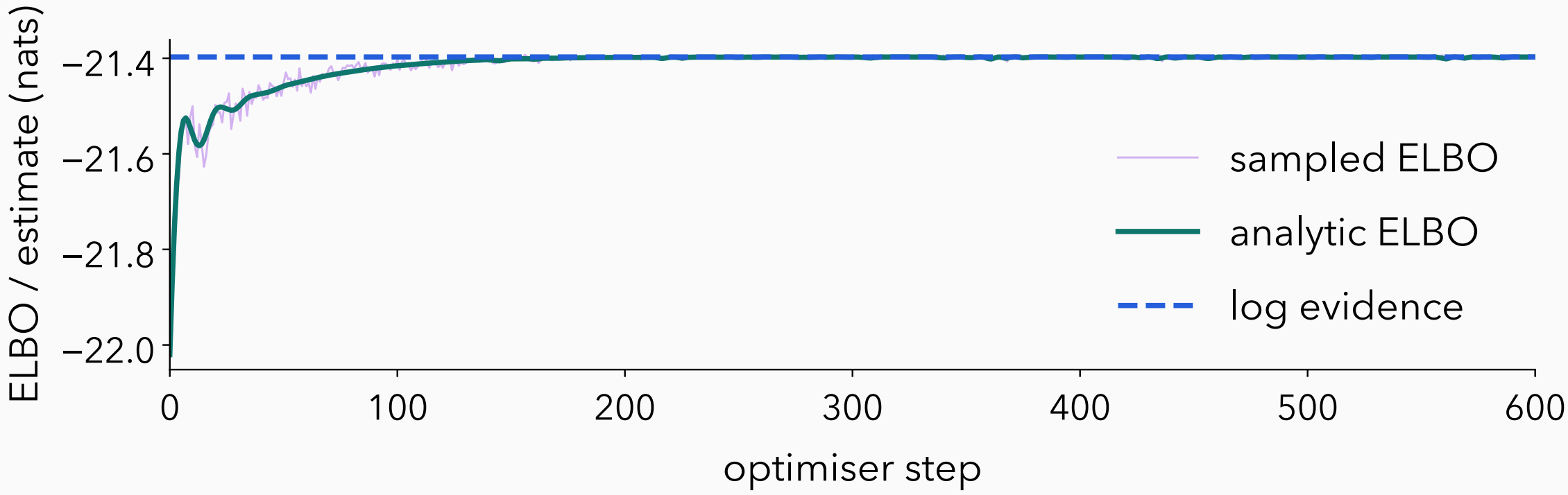
$$\nabla_\psi \mathbb{E}_{q_\psi}[f_\psi(\theta)] = \mathbb{E}_{q_\psi}[f_\psi(\theta)\nabla_\psi \log q_\psi(\theta) + \nabla_\psi f_\psi(\theta)]$$

A score-function estimator accounts explicitly for the changing sampling distribution.

> Does this identity require θ = μ + σε?

▶ Notebook

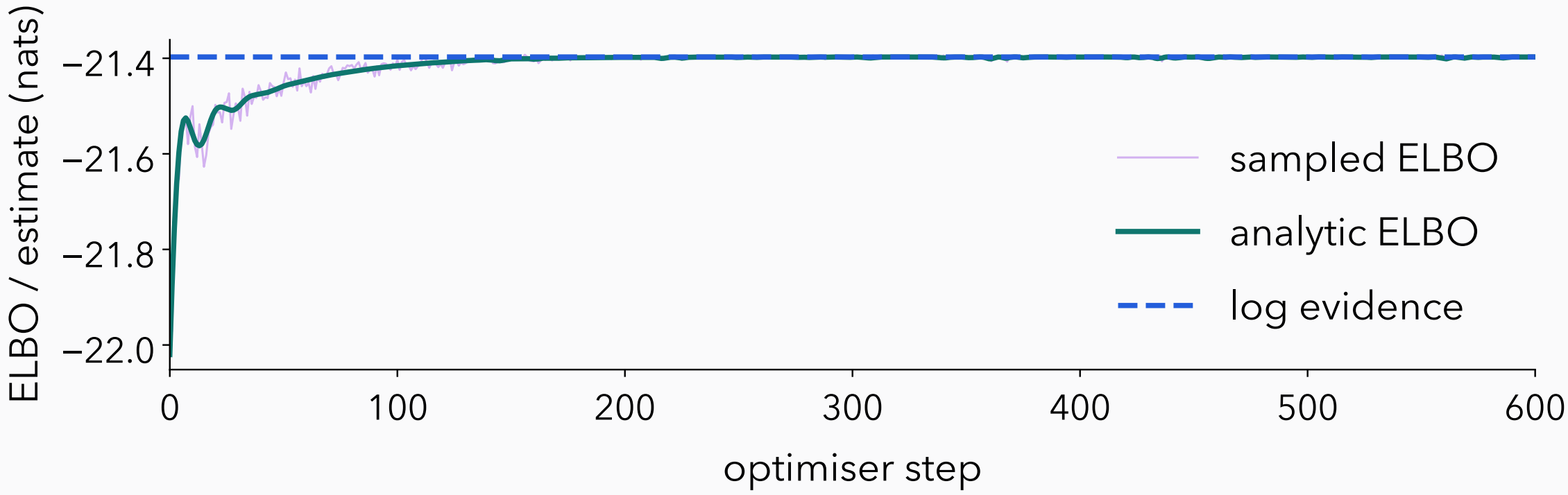
A noisy ELBO estimate can cross the evidence



Can a finite-sample ELBO estimate cross the reference line?

▶ Notebook

A noisy ELBO estimate can cross the evidence



162 of 600 sampled estimates lie above the known evidence in this run.

> Which object remains a lower bound?

▶ Notebook

A higher ELBO does not prove a zero gap

$$\mathrm{ELBO}(q_{\mathrm{new}}) > \mathrm{ELBO}(q_{\mathrm{old}})$$

What does a true increase in the exact ELBO establish?

▶ Notebook

A higher ELBO does not prove a zero gap

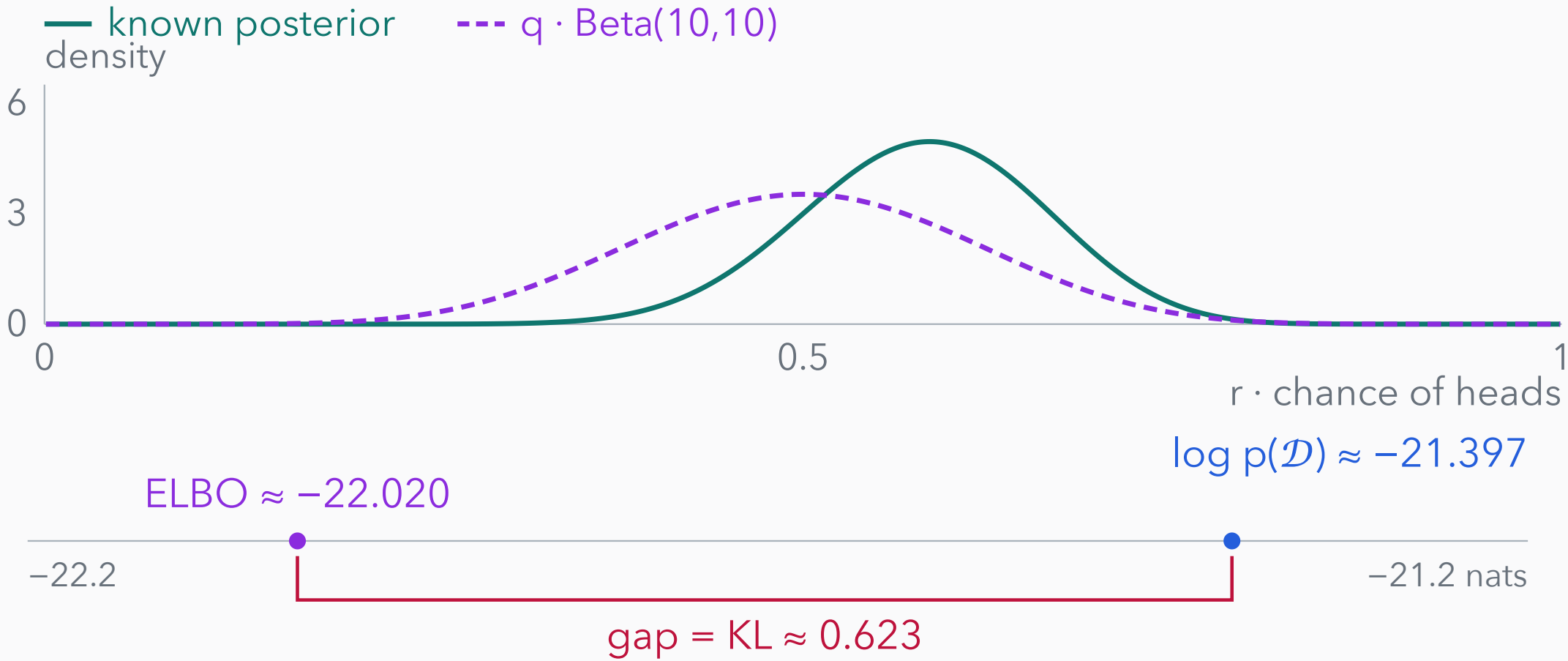
$$\mathrm{ELBO}(q_{\mathrm{new}}) > \mathrm{ELBO}(q_{\mathrm{old}})$$

The family may not contain the posterior, and optimisation may stop short.

> Does a stable loss establish q = p(θ | $\mathcal{D}$)?

▶ Notebook

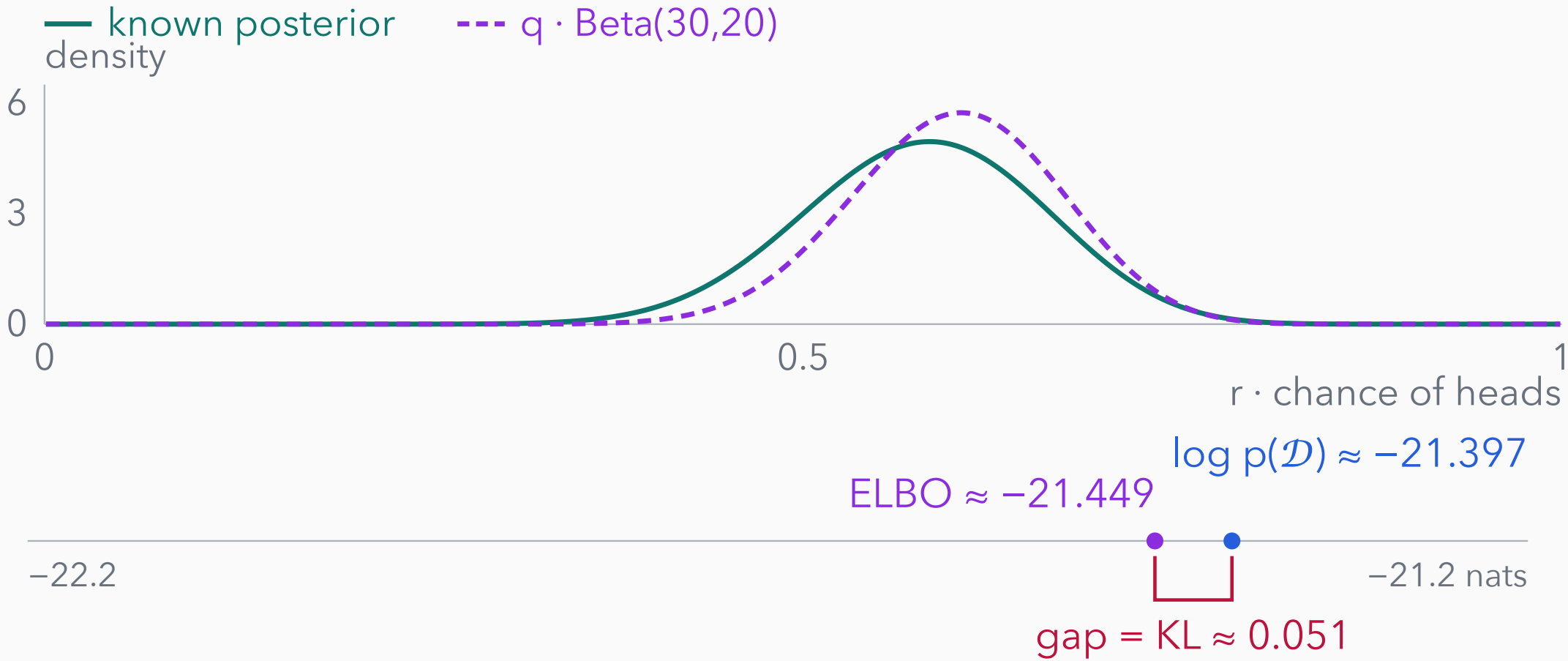
Check the fixed total with three Beta candidates



Add the ELBO and the gap. What is fixed?

▶ Notebook

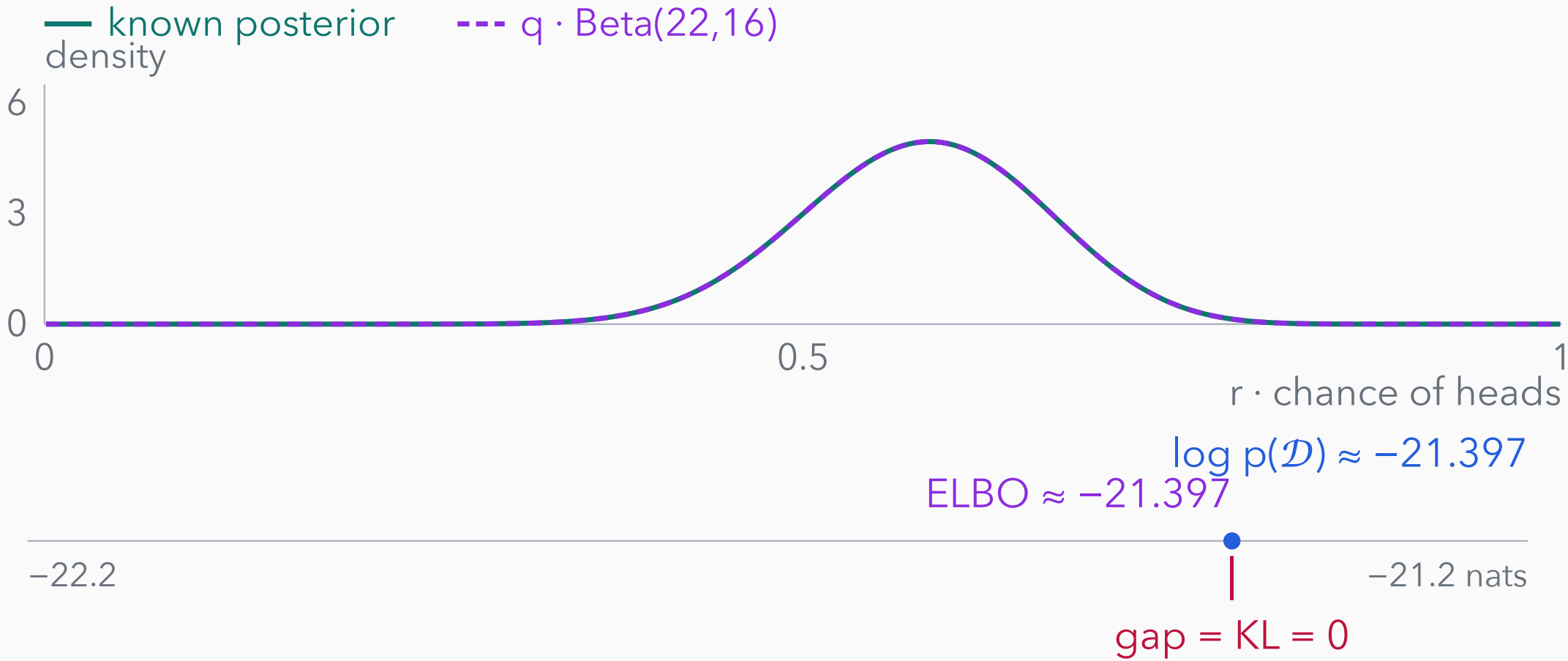
Check the fixed total with three Beta candidates



Does the better candidate raise the ELBO by the amount its KL falls?

▶ Notebook

Check the fixed total with three Beta candidates



What happens at the exact posterior?

▶ Notebook

A mean must say which dimension it averages

object	shape	correct reduction
S scalar Beta draws	[S]	mean over S
S d-dimensional joint log densities	[S]	mean over S
S factorised coordinate log densities	[S, d]	sum over d, then mean over S

> Does every log_prob output still have an event dimension?

▶ Notebook

A mean must say which dimension it averages

object	shape	correct reduction
S scalar Beta draws	[S]	mean over S
S d-dimensional joint log densities	[S]	mean over S
S factorised coordinate log densities	[S, d]	sum over d, then mean over S

MultivariateNormal.log_prob already sums the event dimension. Do not sum the sample axis as well.

> What goes wrong if you sum over S before averaging?

▶ Notebook

An ordered sequence and a count have different evidence

$$p(H = 17 \mid r) = \binom{30}{17} r^{17}(1-r)^{13}$$

> Which event contains more possible sequences?

▶ Notebook

An ordered sequence and a count have different evidence

$$p(H = 17 \mid r) = \binom{30}{17} r^{17}(1-r)^{13}$$

observed object	log evidence
one ordered sequence	−21.397
only the head count	−2.796

> Does this constant change the posterior or its KL gap?

▶ Notebook

Check the sampled objective with analytic Beta moments

$$\mathbb{E}_q[\log r] = \mathrm{digamma}(a) - \mathrm{digamma}(a+b)$$

> Which two expectations determine the coin ELBO?

▶ Notebook

Check the sampled objective with analytic Beta moments

$$\mathbb{E}_q[\log r] = \text{digamma}(a) - \text{digamma}(a+b)$$

$$\mathbb{E}_q[\log(1-r)] = \text{digamma}(b) - \text{digamma}(a+b)$$

> Why write digamma rather than ψ here?

Notebook

Check the sampled objective with analytic Beta moments

$$\mathbb{E}_q[\log r] = \mathrm{digamma}(a) - \mathrm{digamma}(a+b)$$

$$\mathbb{E}_q[\log(1-r)] = \mathrm{digamma}(b) - \mathrm{digamma}(a+b)$$

Insert these into expected log likelihood + expected log prior − expected log q.

> How was the identity checked non-circularly?