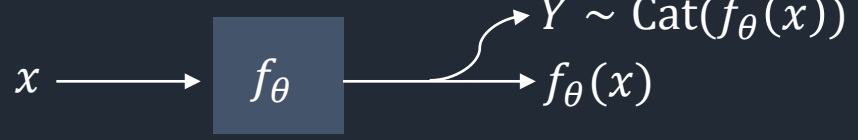
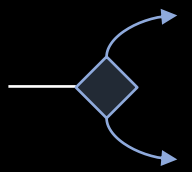
# Consider a simple network, e.g. a classifier

Training dataset: $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

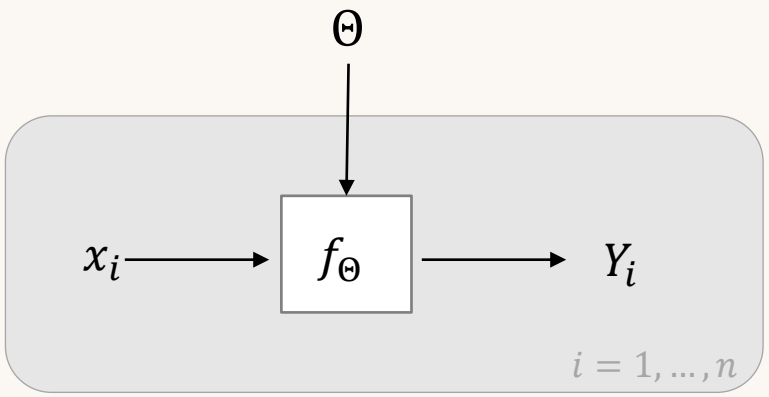
Model: 


*$f_\theta(x)$ is the model's total uncertainty about $Y$*

Earlier, we learnt to quantify uncertainty given a context $\mathcal{O}$,
from an oracle that taught us the link by showing us many example contexts.

But if all we have is a single dataset $\mathcal{D}$,
is there any way to get the epistemic / aleatoric split? 

# An embarrassingly simple approach to uncertainty: Bayesian neural networks




The classic Bayesianist approach:

1. Let Θ be all the parameters of the neural network, and invent a prior distribution

2. Find the posterior distribution $(\Theta|\mathcal{D})$
   — this *is* our epistemic uncertainty

3. Take many samples $\theta_j$ from the posterior distribution, and report an ensemble of predictions $f_{\theta_j}(x^*)$ for the readout at a new input $x^*$
   — this *is* prediction uncertainty

What prior would you use?

What about a prior for number of layers? activation functions?

How on earth might we find this?

# Dropout can be seen as approximating the posterior Gal and Ghahramani 2016

**Finding the true posterior is intractable.** Instead, let's approximate it: within a certain parametric family, find the distribution that best approximates the true posterior.

**The parametric family: dropout.** For any set of network weights $\phi$, where $\phi_{k\,k'}$ is the weight from node $k$ to node $k'$, define a distribution $r_\Theta^\phi$ on $\Theta$ by

$$r_\Theta^\phi(\theta) = \prod_{\text{nodes } k} \left[(1-\pi_0)\,\mathbb{1}_{\theta_{k*}=\phi_{k*}} + \pi_0\,\mathbb{1}_{\theta_{k*}=0}\right]$$

$\theta_{k*} = \{\theta_{k\,k'} : k \to k'\}$
i.e. all weights from node $k$

Equivalently,

$$\Theta_{k\,k'} = \phi_{k\,k'} Z_k \quad \text{where} \quad Z_k \sim \text{Bern}(1-\pi_0) \text{ independently}$$

Do you think this is a sensible family for approximating the posterior $p_{\Theta|\text{data}}$, for our given prior?

**Fitting.** The ELBO bound, with latent variable $\Theta$, says that

$$\log \Pr(\mathcal{D}) = \underbrace{\mathbb{E}_{\Theta\sim r_\Theta^\phi} \log \Pr(\text{data}|\Theta) - \text{KL}\left(r_\Theta^\phi \,\|\, p_\Theta\right)}_{\text{the ELBO}} + \text{KL}\left(r_\Theta^\phi \,\|\, p_{\Theta|\mathcal{D}}\right)$$

so we'll maximize this

We want this small ...
this is called Variational Inference

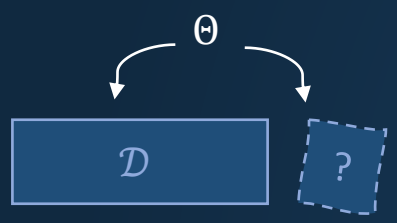
$$\text{ELBO} \approx \mathbb{E}_Z \log \Pr(\text{data}|\Theta = \phi Z) - \frac{1-\pi_0}{2\tau_0^2}\|\phi\|^2 - \text{const}$$

usual loglik, with dropout

If we use a Gaussian prior $p_\Theta$
we get "weight decay" regularization

**Readout.** Let $\hat{\phi}$ be the fitted parameters. Take parameter samples from $r^{\hat{\phi}}$ (i.e. keep dropout switched on at inference time) and report an ensemble of predictions.

"data" is either the context $\mathcal{O}$ or the training dataset $\mathcal{D}$.



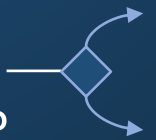

Our thinking so far has been Bayesianist:
epistemic uncertainty *is* the uncertainty of $(\Theta|\text{data})$.

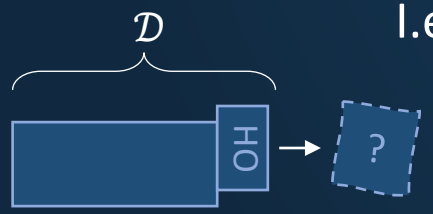
It depends on the prior.
Different Bayesianists will end up with different epistemic uncertainty.

$$\Pr(\theta|\text{data}) \propto \Pr(\theta)\,\Pr(\text{data}|\theta)$$

Standard KL training, with holdout, gives us the full predictive distribution $\Pr_Y(y|x)$ (and hence the total uncertainty).

Can it also tell us the epistemic / aleatoric split?
I.e. empirically, rather than by assuming a prior?





# Let's investigate the joint distribution of a batch of predictions from a Gaussian process

Consider the Bayesian linear regression

$$Y^* \sim \alpha + \beta x^* + N(0, \sigma_0^2)$$

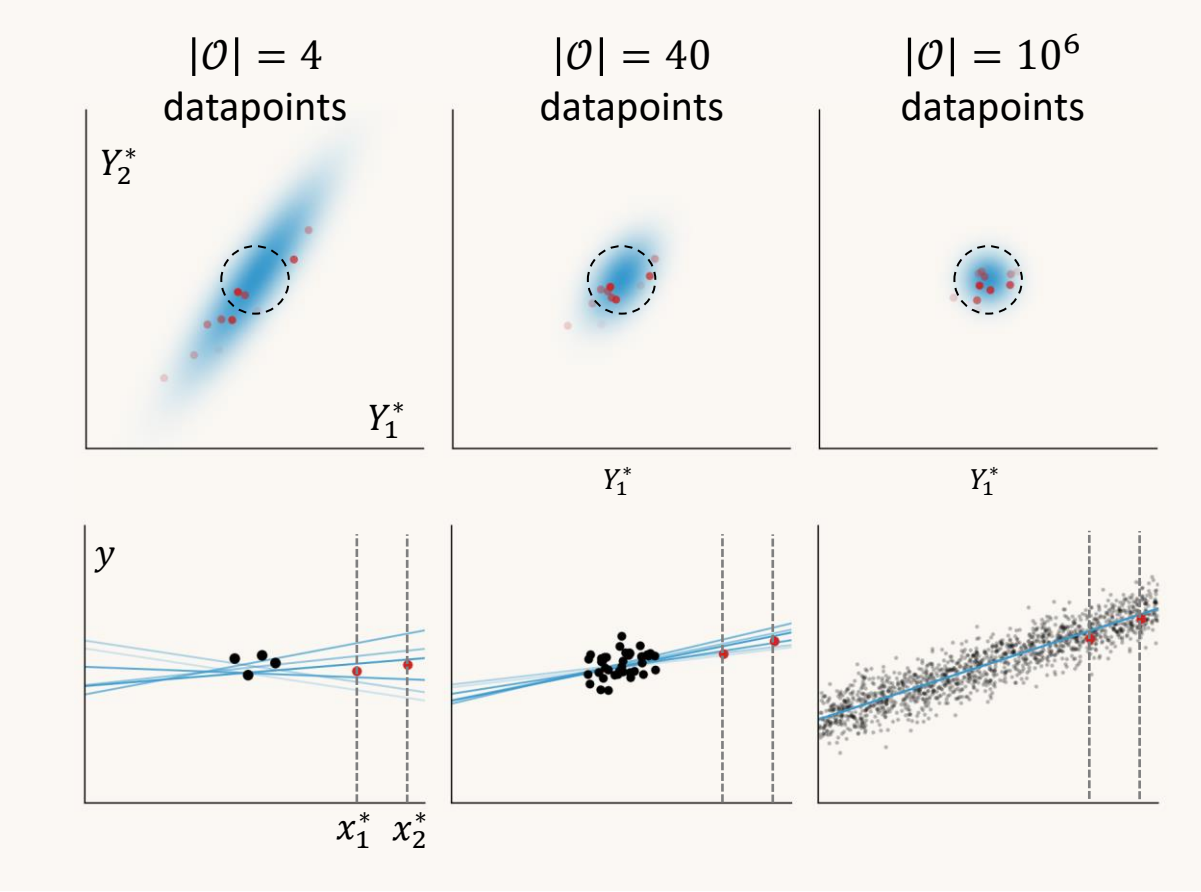
and look at a batch of predictions at $\mathcal{T} = [x_1^*, x_2^*]$

$$\begin{bmatrix} Y_1^* \\ Y_2^* \end{bmatrix} \sim \begin{bmatrix} 1 & x_1^* \\ 1 & x_2^* \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} N(0, \sigma_0^2) \\ N(0, \sigma_0^2) \end{bmatrix}$$

The joint prediction, given context $\mathcal{O}$, is multivariate Gaussian. It has covariance

$$\mathrm{Cov}\left( \begin{bmatrix} Y_1^* \\ Y_2^* \end{bmatrix} \middle| \mathcal{O} \right) = \begin{bmatrix} 1 & x_1^* \\ 1 & x_2^* \end{bmatrix} \mathrm{Cov}(\begin{bmatrix} \alpha \\ \beta \end{bmatrix} | \mathcal{O}) \begin{bmatrix} 1 & x_1^* \\ 1 & x_2^* \end{bmatrix}^T + \sigma_0^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The joint distribution of a batch of predictions depends on both epistemic and aleatoric uncertainty

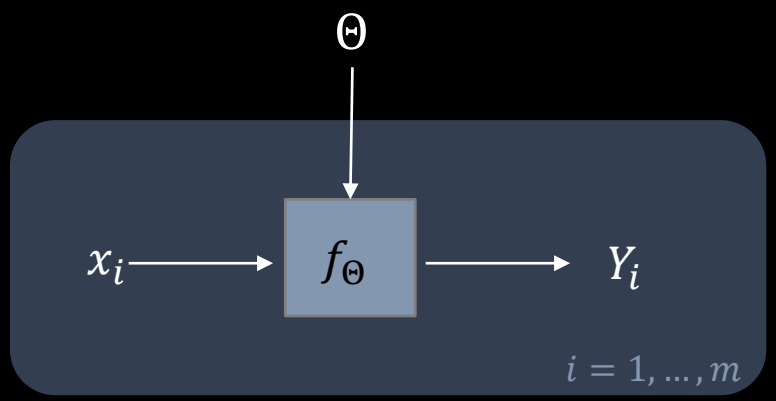



If $x_1^* = x_2^* =: x^*$ then

$$\mathrm{Var}(Y_1^* | \mathcal{O}) = \mathrm{Var}(\alpha + \beta x^* | \mathcal{O}) + \sigma_0^2$$
$$\mathrm{Cov}(Y_1^*, Y_2^* | \mathcal{O}) = \mathrm{Var}(\alpha + \beta x^* | \mathcal{O})$$

By asking for <u>two</u> readouts at a single x*, we learn about total <u>and</u> epistemic uncertainty

# De Finetti's folly



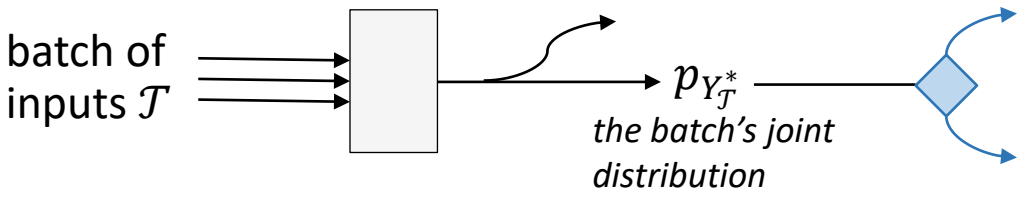

- A prior on Θ induces a joint distribution on a batch of data $(Y_1, \ldots, Y_m)$ at $(x_1, \ldots, x_m)$.
- Different priors lead to different models for the joint distribution.
- The "latent variable" story is underpants.
  All that matters, empirically, is having a model that proposes a joint distribution on batches.

THE UNDERPANTS PRINCIPLE

*"Model parameters are the underwear of modelling — good for your personal support, but not something to show off in public."*

Judge a model by what it says about observables.
Report what it says about observables.

# The empirical view of epistemic uncertainty: we just need a 'joint model'




- We can evaluate a joint model by the likelihood it gives to batches of holdout data.
- We can use it to read out epistemic and aleatoric uncertainty at a given input $x^*$, by asking for the batch $\mathcal{T} = [x^*, x^*]$
- RESEARCH QUESTION
  I speculate that a joint model is trainable by KL minimization

# A testbed for uncertainty Osband et al. 2022, *The Neural Testbed*

**Generator**
(a small neural network with random parameters $\Theta$)

*simulate*

**Dataset $\mathcal{D}$: a single coherent draw from the generator**

| *predictor* | *ground truth response* |
|---|---|
| $x$ | $y$ |
| ⋮ | ⋮ |

*train*

*with your proposed "uncertainty trainer"*

**A fitted joint model**
(any model that can report the joint likelihood of a batch)

*evaluate*

**Evaluate it as a batch-model,** using random batches $\mathcal{T}$:
$\mathbb{E}_{\mathcal{T}}\ \mathrm{KL}\left(q_{Y^*_{\mathcal{T}}|\mathcal{D}} \,\middle\|\, p_{Y^*_{\mathcal{T}}}\right)$

and average over many $\mathcal{D}$

RESULTS

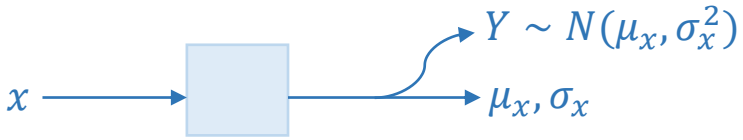
| **how the neural network was fitted (the "uncertainty trainer")** | **score for $\lvert\mathcal{T}\rvert = 1$** | **score for $\lvert\mathcal{T}\rvert = 10$** |
|---|---|---|
| an ordinary MLP, independent predictions | $0.129 - c_1$ | $1.367 - c_{10}$ |
| Gal's dropout | $0.128 - c_1$ | $1.347 - c_{10}$ |
| More refined approximation to the Bayesian neural network (using Langevin dynamics) | $0.122 - c_1$ | $0.947 - c_{10}$ |

Training with dropout, it turns out, doesn't do any better at uncertainty than naïve training with an ordinary MLP

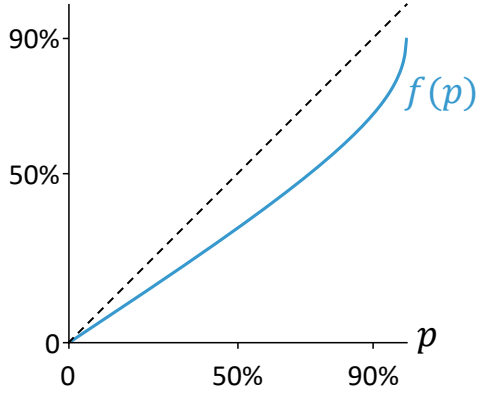
# Lots of papers talk about ‘calibration of uncertainty’. What is this?

Standard fitting should, in principle, learn the full distribution $p_{Y;x}$ — but does it actually learn it well?
(Do our class probabilities match the real class probabilities? Does our standard deviation match the real spread?)

To measure if the output distribution fits well, we can measure its calibration on holdout data $(x_i, y_i)_{i=1,\dots,n}$.

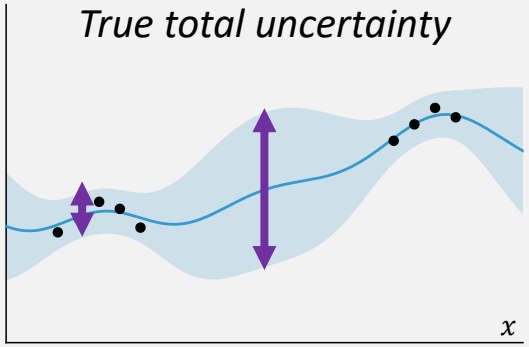



If the model fits, we expect $y_i \in \mu_{x_i} \pm 1.96\sigma_{x_i}$ in 95% of cases. Let’s measure the actual frequency $f(p)$ and compare to the target frequency $p = 0.95$.



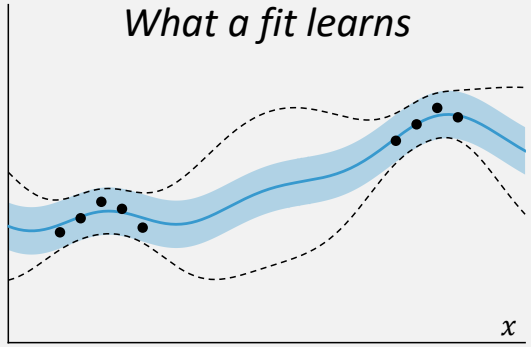

Networks trained for accuracy (i.e. weakly regularized) are typically over-confident. Why?

*True total uncertainty*




The true total uncertainty ↕ (total = epistemic + aleatoric) is
- large where we have little data
- small where we have more data

*What a fit learns*




The fit learns (correctly) that the total uncertainty is small — on the data that it was trained on.

Since we’re using weak regularization, it learns to do well on training data with little regard for holdout data. It’s easy for it to extrapolate “small total uncertainty” to regions without much training data. The model is therefore overconfident on holdout data.

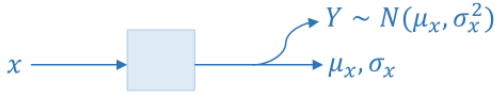
**Proper holdout validation gives us a calibrated network, but weak regularization leads to overconfidence.**

Guo et al. “On Calibration of Modern Neural Networks”, 2017. Kuleshov et al. “Accurate uncertainties for deep learning”, 2018

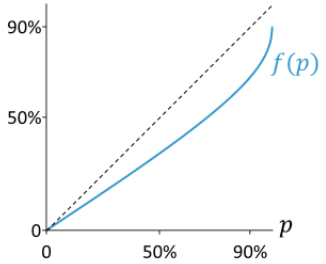
# Lots of papers talk about ‘calibration of uncertainty’. What is this?

Standard fitting should, in principle, learn the full distribution $p_{Y;x}$ — but does it actually learn it well?
(Do our class probabilities match the real class probabilities? Does our standard deviation match the real spread?)

To measure if the output distribution fits well, we can measure its calibration on holdout data $(x_i, y_i)_{i=1,\dots,n}$.

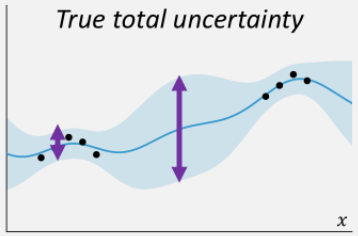



If the model fits, we expect $y_i \in \mu_{x_i} \pm 1.96\sigma_{x_i}$ in 95% of cases. Let’s measure the actual frequency $f(p)$ and compare to the target frequency $p = 0.95$.



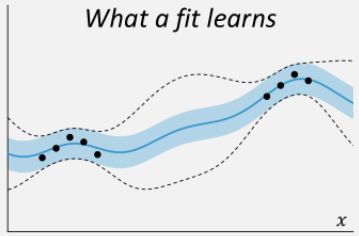

Networks trained for accuracy (i.e. weakly regularized) are typically over-confident. Why?

*True total uncertainty*




The true total uncertainty ↕ is large where we have little data, small where we have more data

*What a fit learns*




The fit learns (correctly) that the total uncertainty is small — on the data that it was trained on.

Since we’re using weak regularization, it learns to do well on training data with little regard for holdout data. It’s easy for it to extrapolate “small total uncertainty” to regions without much training data. The model is therefore overconfident on holdout data.

**Proper holdout validation gives us a calibrated network, but weak regularization leads to overconfidence.**

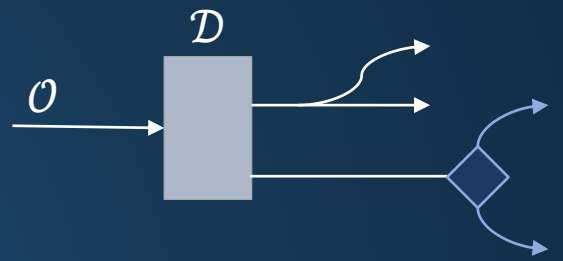
8

Guo et al. “On Calibration of Modern Neural Networks”, 2017. Kuleshov et al. “Accurate uncertainties for deep learning”, 2018

- Properly regularized networks are automatically calibrated.
  Calibration is only a story about weak regularization.
- Calibration is about total uncertainty, not the epistemic / aleatoric split.
  *It is inherently incapable of addressing epistemic uncertainty,*
  *since it’s only about networks that make independent predictions.*

LLMs get data from two sources, training data $\mathcal{D}$ and context data $\mathcal{O}$.
They handle

- uncertainty ‘in their weights’ given $\mathcal{D}$
- uncertainty ‘in their processing’ given $\mathcal{O}$




The distinction between $\mathcal{D}$ and $\mathcal{O}$ is artificial,
but there is not yet a grand unified theory of uncertainty.
Prompt baking, Bhargava et al. 2024

Aligned LLMs seem to have ‘circuits’ for epistemic certainty.
The biology of LLMs, Lindsey et al. 2025

*By default, Anthropic’s Haiku 3.5 declines to answer questions. But it has circuits that detect if the question is something it knows about and, when these circuits fire, it will respond. When these ‘I know’ circuits misfire, it hallucinates.*