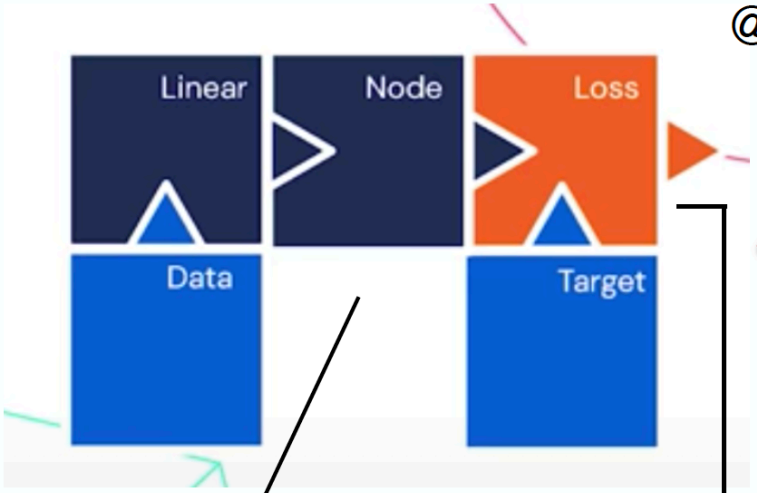
RECAP:

(deepmind @Czarnecki)

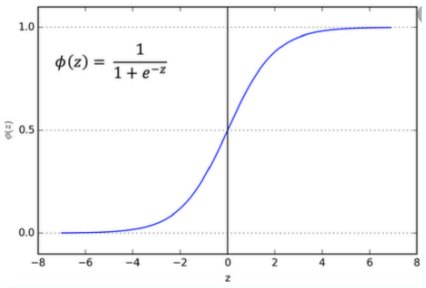


$$f_W(\vec{x}) = \vec{y}$$

$$\vdots$$

$$g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$

activation function



$$\frac{1}{N}\sum_{i=1}^{N}(y_i - f_w(x))^2$$

Minimized through gradient descent (backpropagation)

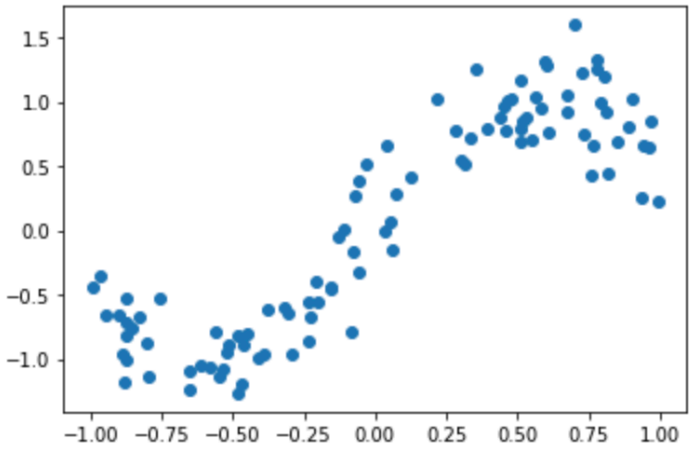
UNIVERSAL APPROXIMATION THEOREM

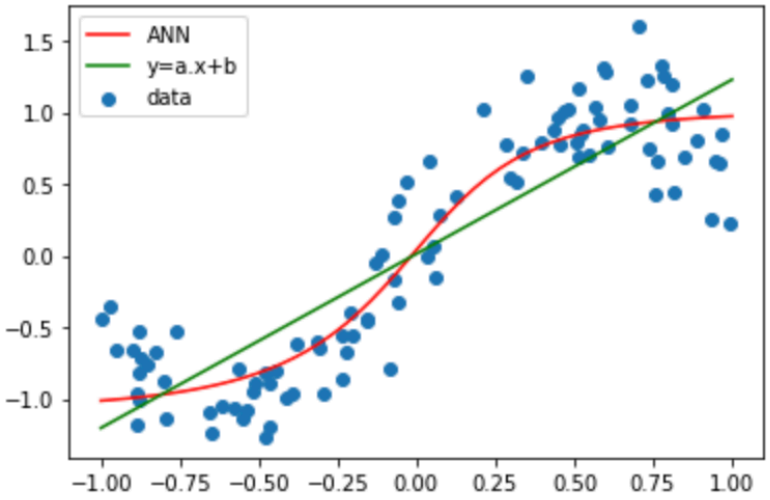
FOR ANY CONTINUOS FUNCTION FOR A HYPERCUBE [0,1]d TO REAL NUMBERS, NON-CONSTANT, BOUNDED AND CONTINUOUS ACTIVATION FUNCTION f, AND EVERY POSITIVE EPSILON, THERE EXISTS A1-HIDDEN LAYER NEURAL NETWORK USING f THAT OBTAINS AT MOST EPSILON ERROR IN FUNCTIONAL SPACE

Horvik+91

"BIG ENOUGH NETWORK CAN APPROXIMATE, BUT NOT REPRESENT ANY SMOOTH FUNCTION. THE MATH DEMONSTRATION IMPLIES SHOWING THAT NETWORS ARE DENSE IN THE SPACE OF TARGET FUNCTIONS"

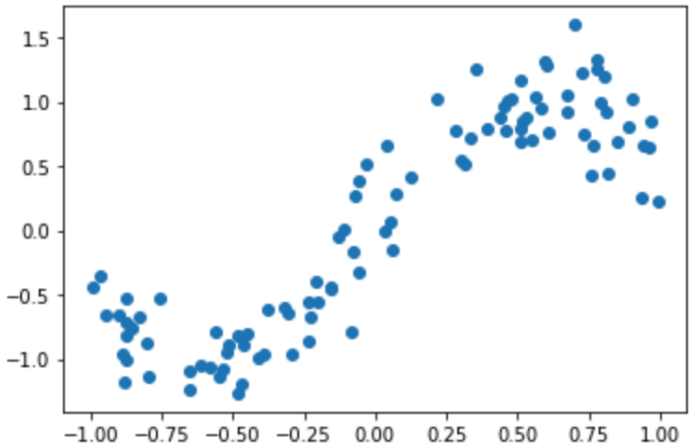
y=F(x;w)? w=(a,b)

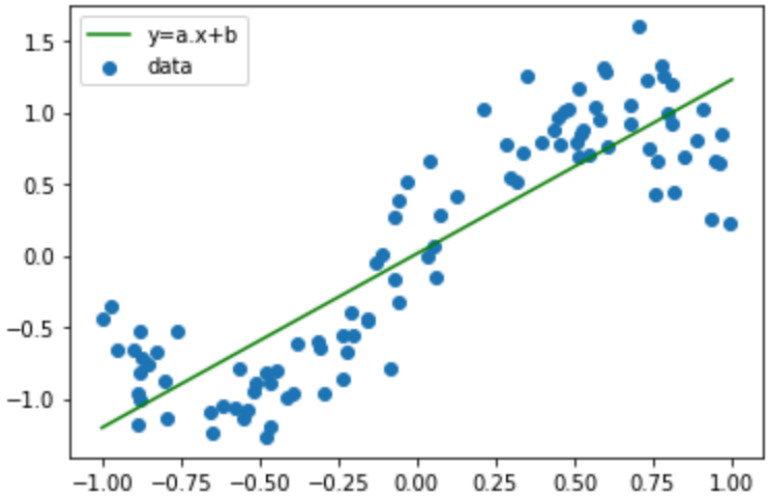


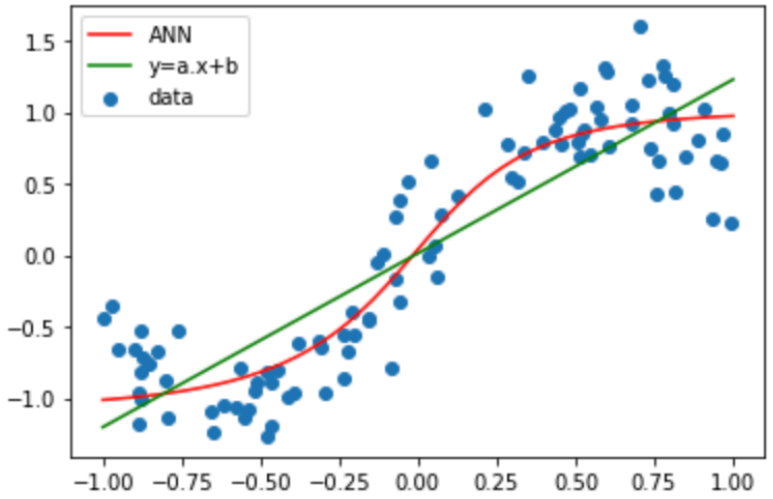


p is data driven, no physics

y=F(x;w)? w=(a,b)







p is derived from physical insight

p is data driven, no physics

Neural Networks as statistical models

$$f_W(\vec{x}) = \vec{y} \quad \Leftrightarrow \quad p(y|x;W)$$

we have a set of independent realizations:

$$(x_i, y_i); i = 1, ..., N$$

and want to find the underlying probability distribution of y given x - p(y|x):

If we assume that p(y|x) is a Gaussian distribution (from Zejko's lecture yesterday):

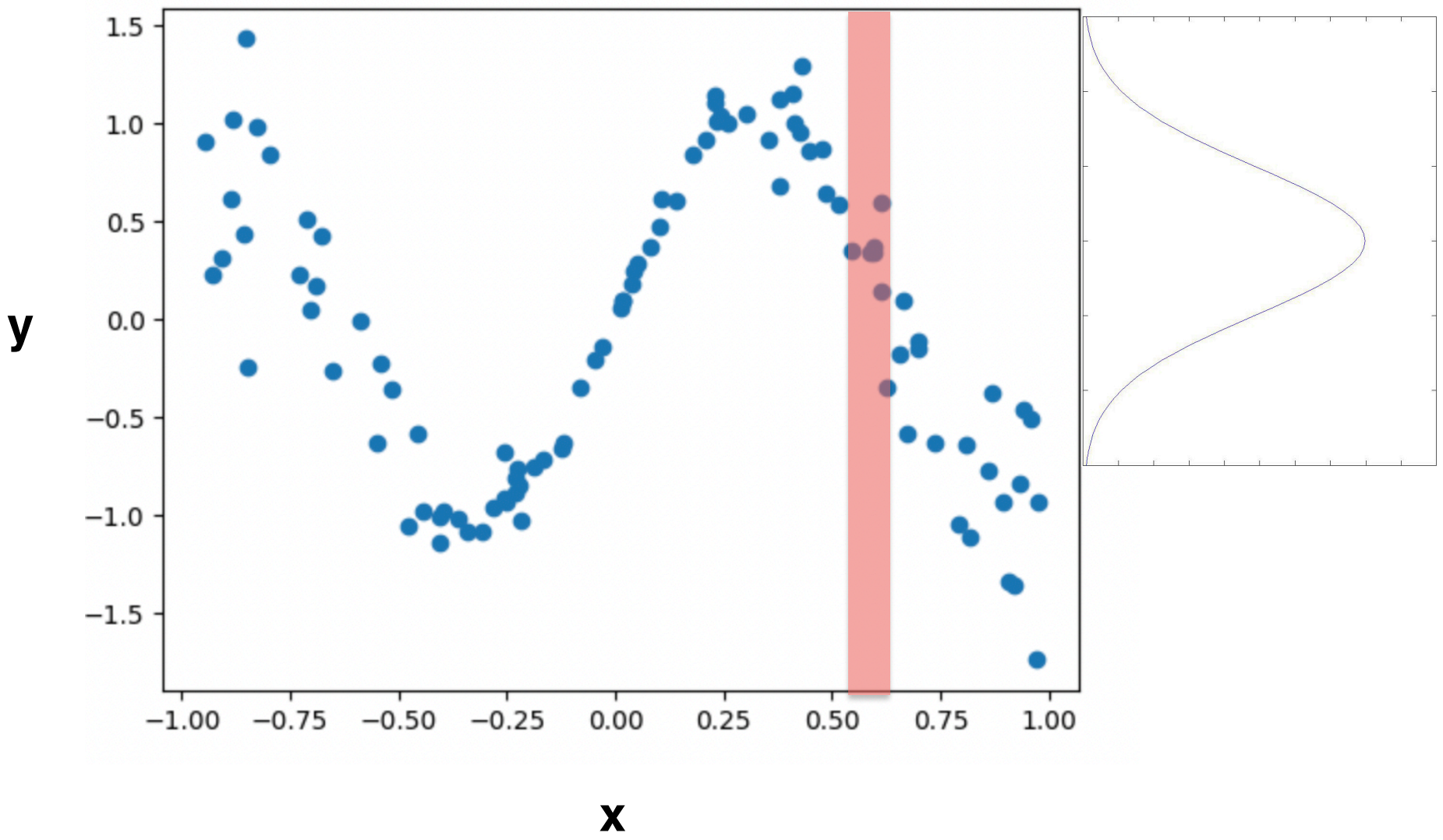
$$p_w(y|x) = L(w, \sigma^2) = \prod_{i=1}^{N} (2\pi\sigma^2)^{-N/2} exp(-\frac{1}{\sigma^2}(y_i - f_w(x_i))^2)$$

Network output

weights

So the goal is to find the values of w (weights) that estimate the mean of a Gaussian distribution for y given a set of N independent observations

$$p(y|x) = \mathcal{N}(\mu, \sigma)$$



we have a set of independent realizations:

$$(x_i, y_i); i = 1, ..., N$$

and want to find the underlying probability distribution of y given x [p(y|x)]:

Since y can only take 0 / 1 values, we assume it can be parametrized with a Bernouilli distribution:

$$P(y_i = 0|x_i) = 1 - sigm(f_w(x_i)) \qquad P(y_i = 1|x_i) = sigm(f_w(x_i))$$

sigmoid function

weights

So the goal is to find the values of w (weights) that generate a Bernouilli distribution for y given a set of N independent observations

Regressions

$$p_w(y|x) = L(w, \sigma^2) = \prod_{i=1}^{N} (2\pi\sigma^2)^{-N/2} exp(-\frac{1}{\sigma^2}(y_i - f_w(x_i))^2)$$

weights (model parameter)

neural network

And then take the log:

$$ln\ L = cst - \sum_{i=1}^{N} \frac{(x_i - f_w(x_i))^2}{2\sigma^2}$$

And now assume sigma = 1:

$$ln\ L = -\sum_{i=1}^{N} (x_i - f_w(x_i))^2 = MSE$$

Regressions

WHEN USING THE MEAN SQUARED ERROR, WE ARE ASSUMING THAT THE POSTERIOR P(y|x) IS A NORMAL DISTRIBUTION.

OUR ESTIMATOR IS THEREFORE THE MEAN OF THE NORMAL PDF WITH SIGMA = 1 THAT MAXIMIZES THE LIKELIHOOD (MAXIMUM LIKELIHOOD ESTIMATION).

What about classification?

$$H = -\frac{1}{N}\sum_{i=1}^{N} y_i.log(f_w(x_i)) + (1 - y_i).log(1 - f_w(x_i))$$

GROUND TRUTH

GROUND TRUTH

NETWORK OUTPUT

NETWORK OUTPUT

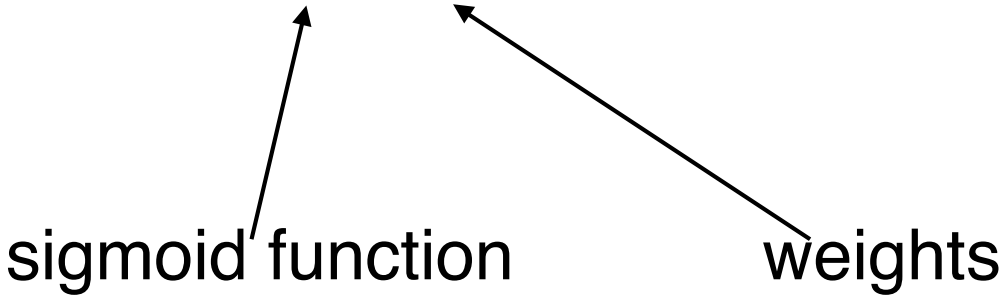
we have a set of independent realizations:

$$(x_i, y_i); i = 1, ..., N$$

and want to find the underlying probability distribution of y given x:

Since y can only take 0 / 1 values, we assume it can be parametrized with a **Bernouilli distribution**:

$$P(y_i = 0|x_i) = 1 - sigm(f_w(x_i)) \qquad P(y_i = 1|x_i) = sigm(f_w(x_i))$$



So the goal is to find the values of w (weights) that generate a Bernouilli distribution for y given a set of N independent observations

This can be achieved via Maximum Likelihood estimation:

The likelihood of a given observation under the Bernouilli assumption can be written as:

$$L(w; x_i, y_i) = [sigm(f_w(x_i))]^{y_i} [1 - sigm(f_w(x_i))]^{1-y_i}$$

which is equal to $[sigm(f_w(x_i))]$ if y=1 and $[1 - sigm(f_w(x_i))]$ if y=0

And the likelihood of the entire sample is the product of the likelihoods:

$$L(w; x_i, y_i) = \prod_{i=1}^{N} [sigm(f_w(x_i))]^{y_i} [1 - sigm(f_w(x_i))]^{1-y_i}$$

So, the log-likelihood:

$$l(w; x_i, y_i) = \sum_{i=1}^{N} y_i \times \log[sigm(f_w(x_i))] + (1 - y_i) \times \log[1 - sigm(f_w(x_i))]$$

THEREFORE EQUIVALENT TO THE CROSS-ENTROPY LOSS:

$$l(w; x_i, y_i) = \sum_{i=1}^{N} y_i \times \log[sigm(f_w(x_i))] + (1 - y_i) \times \log[1 - sigm(f_w(x_i))]$$

$$H = -\frac{1}{N} \sum_{i=1}^{N} y_i.log(f_w(x_i)) + (1 - y_i).log(1 - f_w(x_i))$$

WHEN YOU DO A BINARY CLASSIFICATION WITH NEURAL NETWORKS YOU ARE SIMPLY FINDING THE WEIGHTS OF YOUR MODEL THAT MAXIMIZE THE LIKELIHOOD OF A BERNOUILLI DISTRIBUTION

ASSUMES THAT THE POSTERIOR p(ylx) CAN BE MODELLED BY A BERNOUILLI DISTRIBUTION

For regressions: let's do some **"standard" bayesian inference**

goal of inference

posterior ***likelihood*** ***prior***

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

$$\ln p(X \mid \theta) = \ln \mathscr{L} = \left(X - m(\theta)\right)^{T} \mathbf{C}^{-1} \left(X - m(\theta)\right)$$

Mean of gaussians

model

covariance matrix

For regressions: let's do some **"standard" bayesian inference**

goal of inference

posterior *likelihood* *prior*

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

$$\ln p(X \mid \theta) = \ln \mathscr{L} = \left(X - m(\theta)\right)^T \mathbf{C}^{-1} \left(X - m(\theta)\right)$$

Mean of gaussians

model

covariance matrix

If all Gaussians are assumed to be independent and with standard deviation =1, then:

$$\mathbf{C}^{-1} = \mathbf{I} \qquad ln\ p(X|\theta) = (X - m(\theta))^2 = MSE$$

$$ln\ p(X|\theta) = (X - m(\theta))^2 = MSE$$

If you do not care about the posterior, one way to find the parameters of the model theta, is by maximising the likelihood: **maximum likelihood estimation, i.e. minimising the MSE**

When training a NN with an MSE loss, we are finding the parameters of the network w that maximise the likelihood, under a Gaussian assumption.

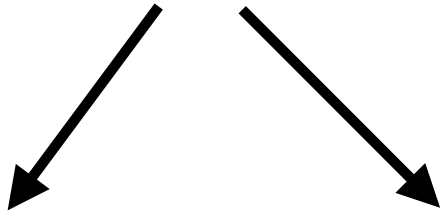
$$ln\ p(X|w) \sim (X - f_w(X)))^2 = MSE$$

Weights of NN

A standard NN performs Maximum Likelihood Estimation of the weights W

$$D = (x^{(i)}, y^{(i)})$$

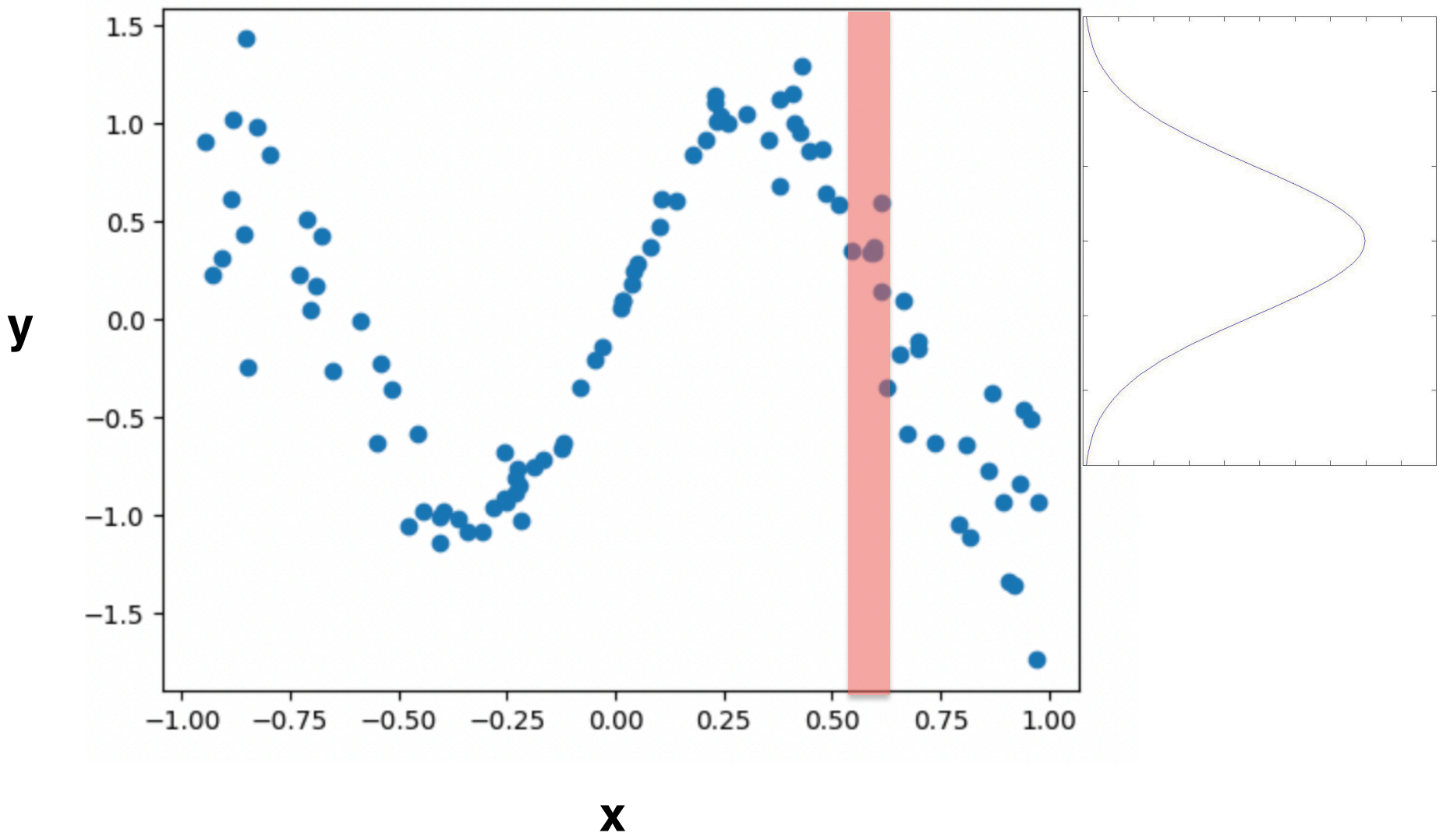
$$p(D|w) = \prod_i p(y^{(i)}|x^{(i)}, W) \sim p_w(y|x)$$



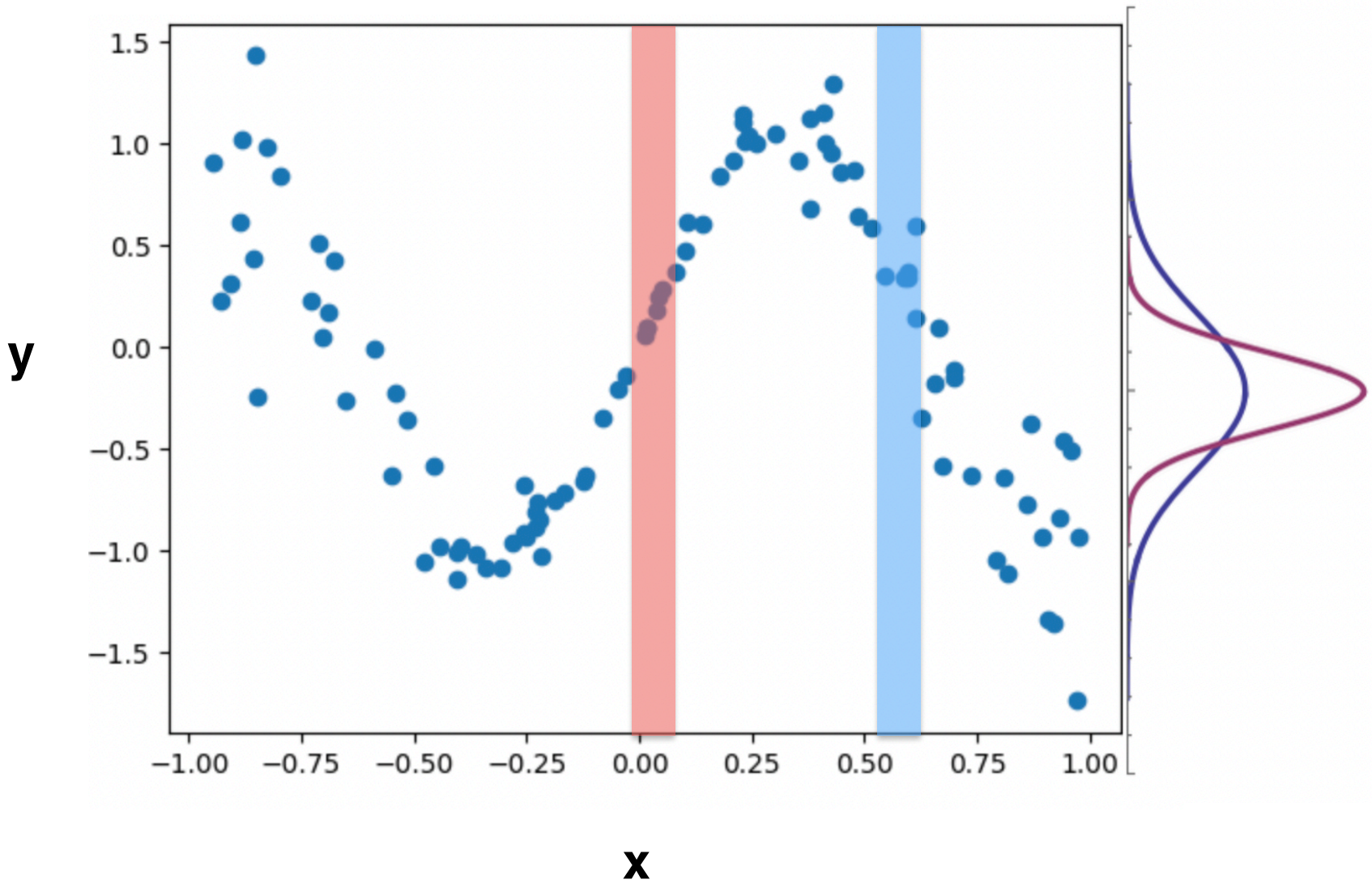
Crossentropy
(~ Bernouilli)

MSE
(~Normal(mu,1))

$$p(y|x) = \mathcal{N}(\mu, 1)$$



$$p(y|x) = \mathcal{N}(\mu, 1) \quad ?$$

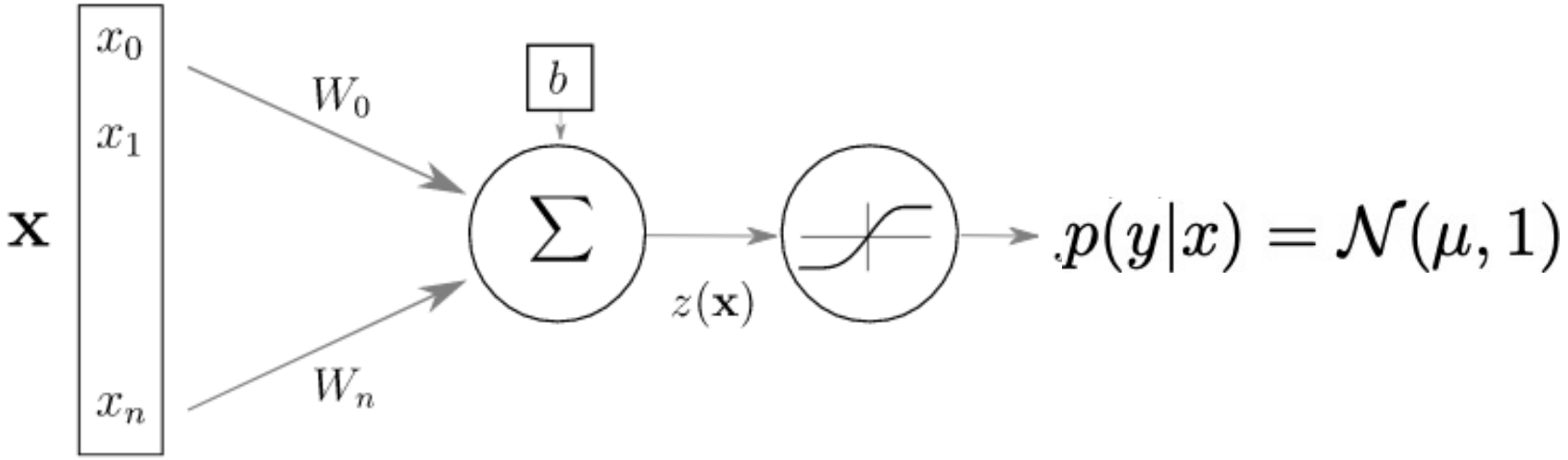


TWO TYPES OF UNCERTAINTY:

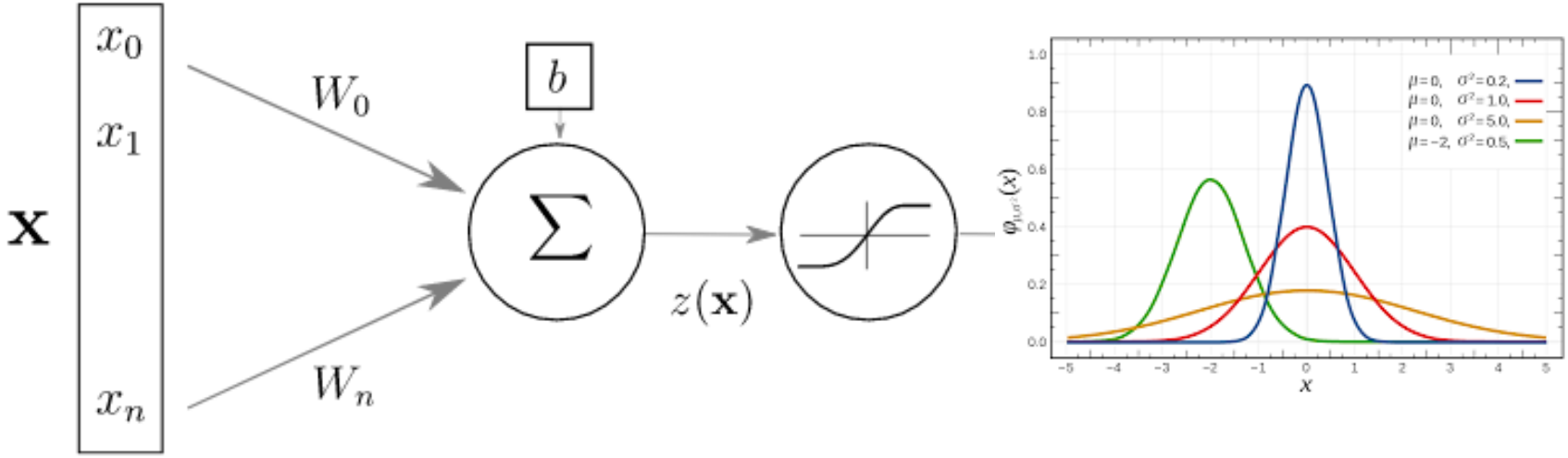
- **Aleatoric (Random)**:Uncertainty coming from the inherent noise in training data. It cannot be reduced if we get more data. Aleatoric uncertainty is covered by the probability distribution used to define the likelihood function

- **Epistemic (Systematics)**: This kind of uncertainty can be reduced if we get more data. Consequently, epistemic uncertainty is higher in regions of no or little training data and lower in regions of more training data

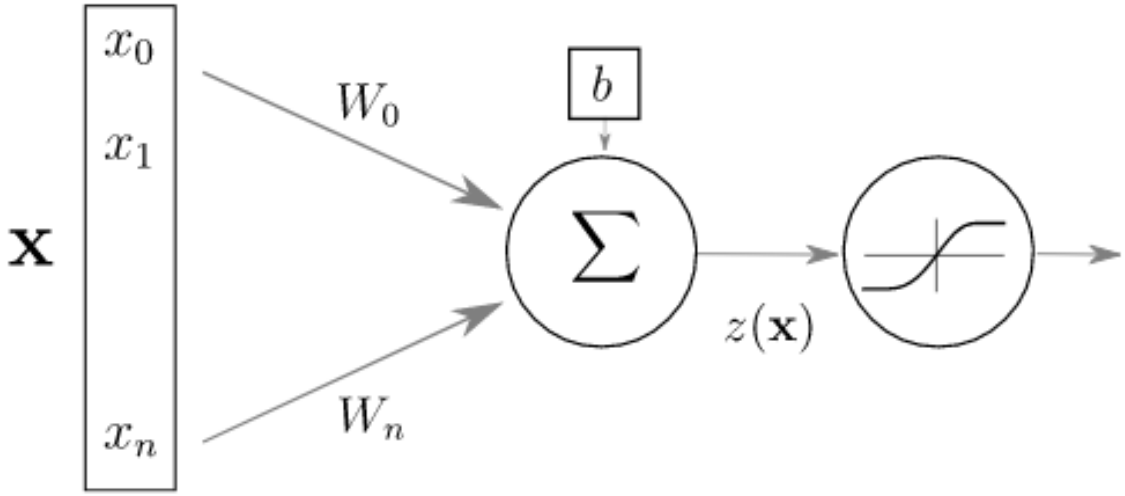
Mixture Density Network



$$p(y|x) = \mathcal{N}(\mu, \sigma)$$

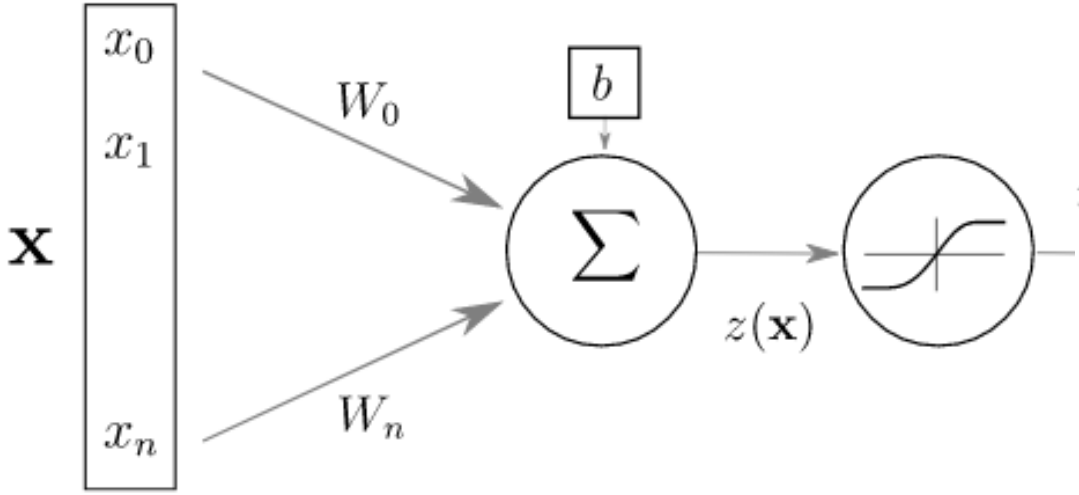


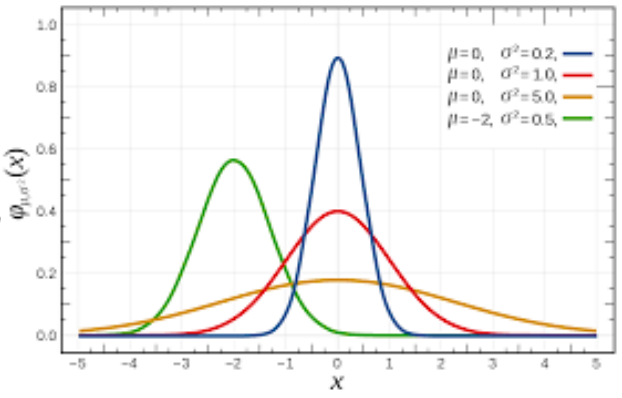
Mixture Density Network



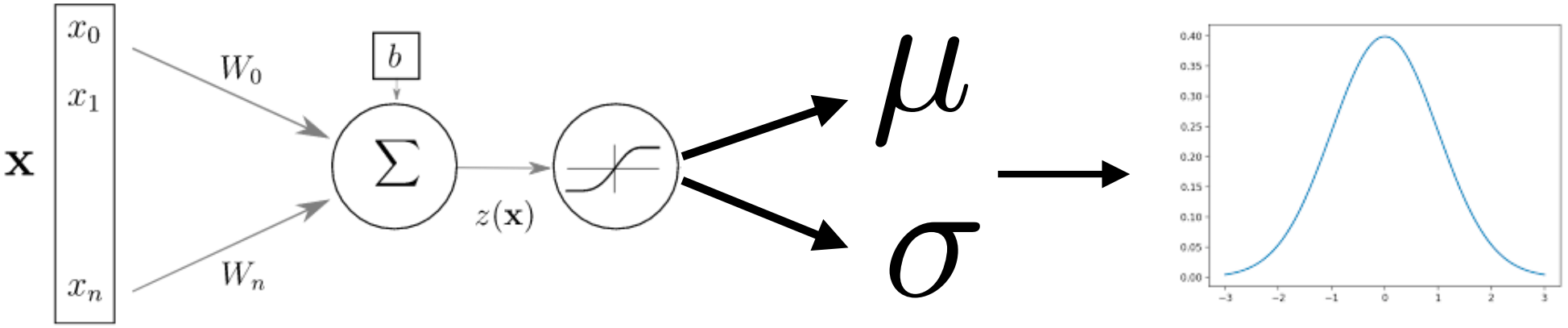
MSE = Max(p(D|w))

$p(D|w)$





Mixture Density Network

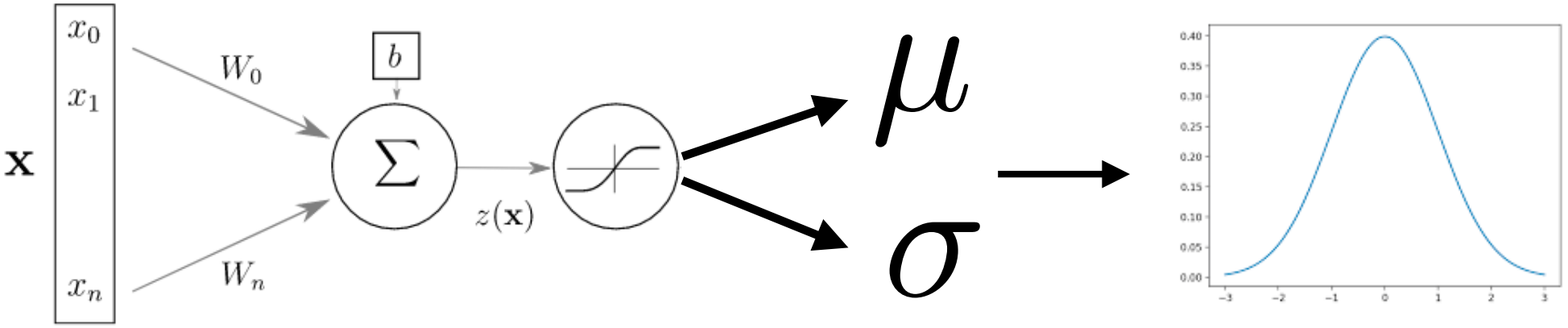


$p(D|w)$

[Likelihood of y under fw(x)]

Reparametrizarion trick: The neural network outputs the parameters of a distribution, on which one can back propagate.

Probabilistic Neural Network

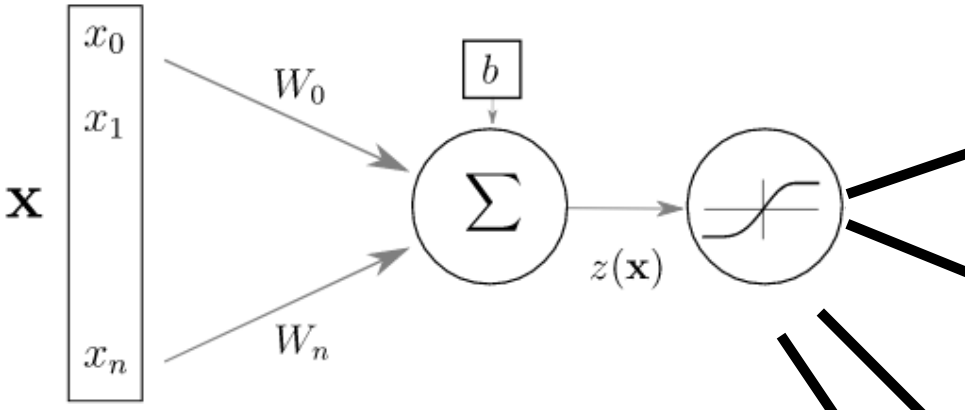


$p(D|w)$

[Likelihood of y under fw(x)]

The loss function becomes then the max of the (log) likelihood

What if the posterior is not gaussian?

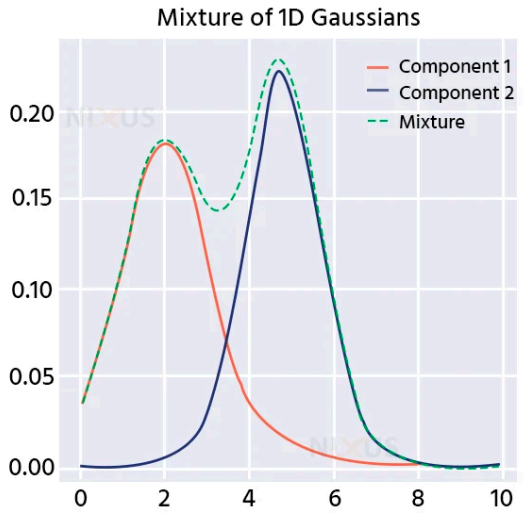


μ

σ

μ

σ



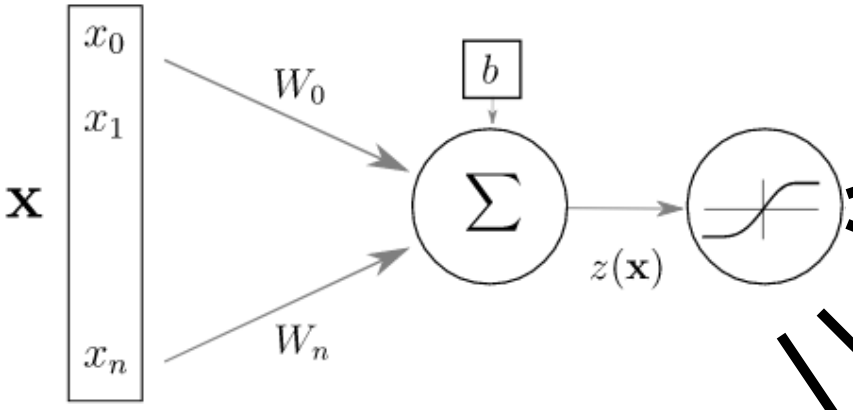
Assume any distribution than
can be parametrized with some parameters and add outputs to the NN

$p(D|w)$

[Likelihood of y under fw(x)]

The loss function remains the max of the (log) likelihood

What if the posterior is not gaussian?

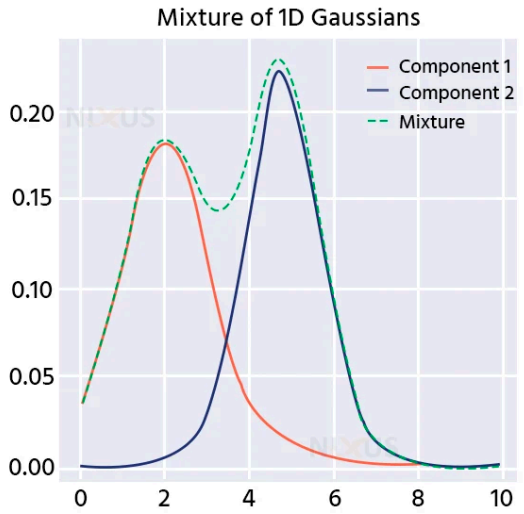


μ

σ

μ

σ



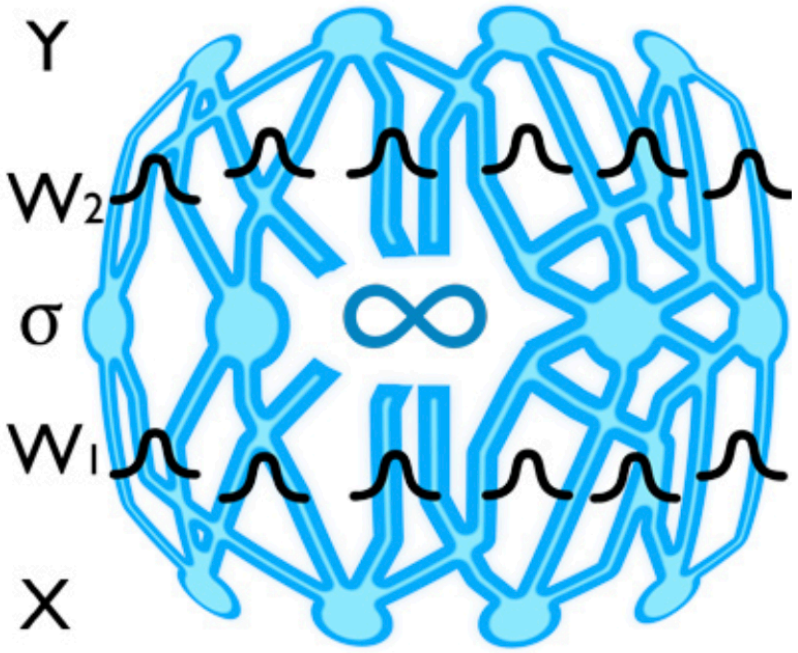
Assume any distribution than
can be parametrized with some parameters and add outputs to the NN

$p(D|w)$

[Likelihood of y under fw(x)]

*Neural Flows allow to estimate pdfs without assuming any parametrization (cycle3)

CAPTURING "MODEL UNCERTAINTY", OR EPISTEMIC UNCERTAINTY, OR UNCERTAINTY ON THE WEIGHTS

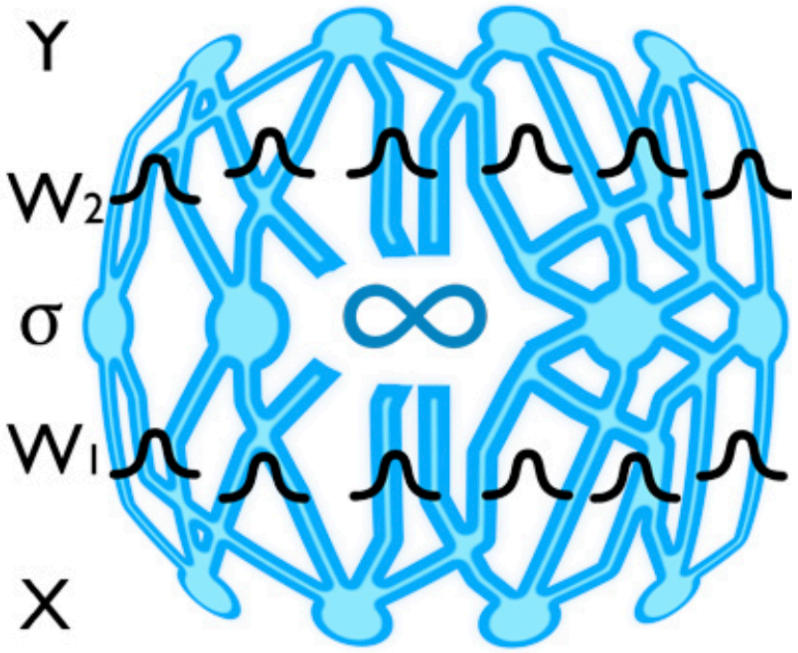


In the previous approach, we are choosing one model among all possible models

IDEALLY, WE WOULD LIKE TO MARGINALIZE OVER ALL MODELS AND OVER ALL POSSIBLE WEIGHT VALUES

$$p(y|x, D) = \int p(y|x, W).p(W|D)dW$$

CAPTURING "MODEL UNCERTAINTY", OR EPISTEMIC UNCERTAINTY



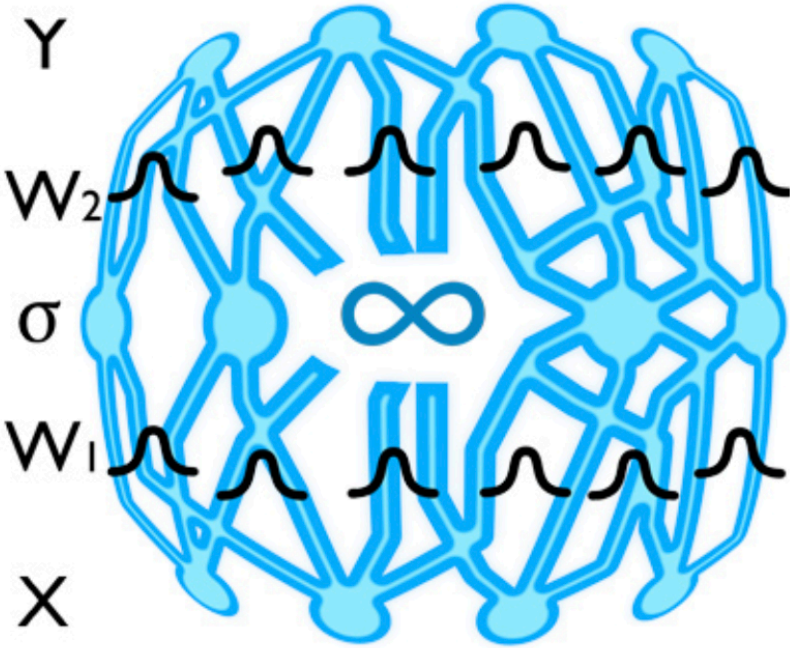
In the previous approach, we are choosing one model among all possible models

IDEALLY, WE WOULD LIKE TO MARGINALIZE OVER ALL MODELS AND OVER ALL POSSIBLE WEIGHT VALUES

$$p(y|x, D) = \int p(y|x, W)\,p(W|D)\,dW$$

This is typically intractable

Bayesian Neural Networks are an approach for epistemic uncertainty estimation



BNNs ADD A PRIOR DISTRIBUTION TO EACH WEIGHT - <u>HARD TO TRAIN</u>

Deep Ensembles

$$p(y|x) = M^{-1} \sum_{m=1}^{M} p_{w_m}(y|x, w_m)$$

For classification, this corresponds to averaging the predicted probabilities

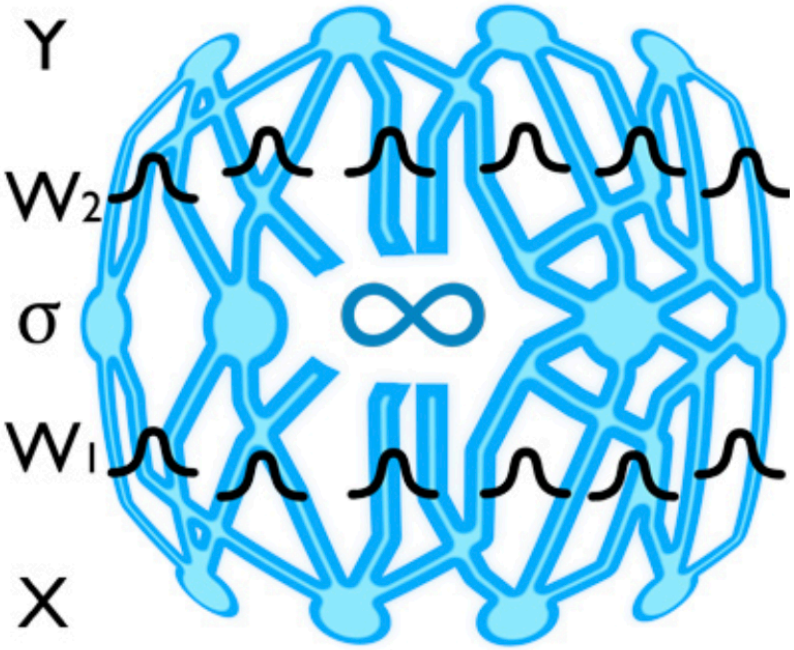
For regression, the prediction is a Mixture of Gaussians.
(One Gaussian / model)

Which can be described with the following summary statistics

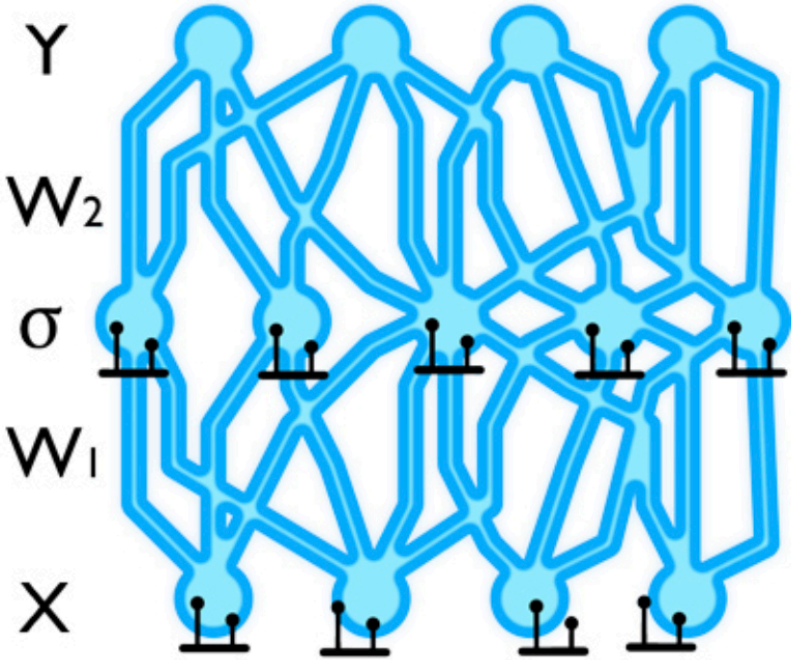
$$\mu_* = M^{-1} \sum_m \mu_{w_m} \qquad \sigma_*^2 = M^{-1} \sum_m (\sigma_{w_m}^2 + \mu_{w_m}^2) - \mu_*^2$$

DROPOUT AS EPISTEMIC UNCERTAINTY ESTIMATION

Denker&LEcun91, Neal+95, Graves+11, Kingma+15, **Gal+15…**



BNNs ADD A PRIOR DISTRIBUTION TO EACH WEIGHT - HARD TO TRAIN



GAL+15 SHOW THAT DROPOUT CAN BE USED TO ESTIMATE UNCERTAINTY

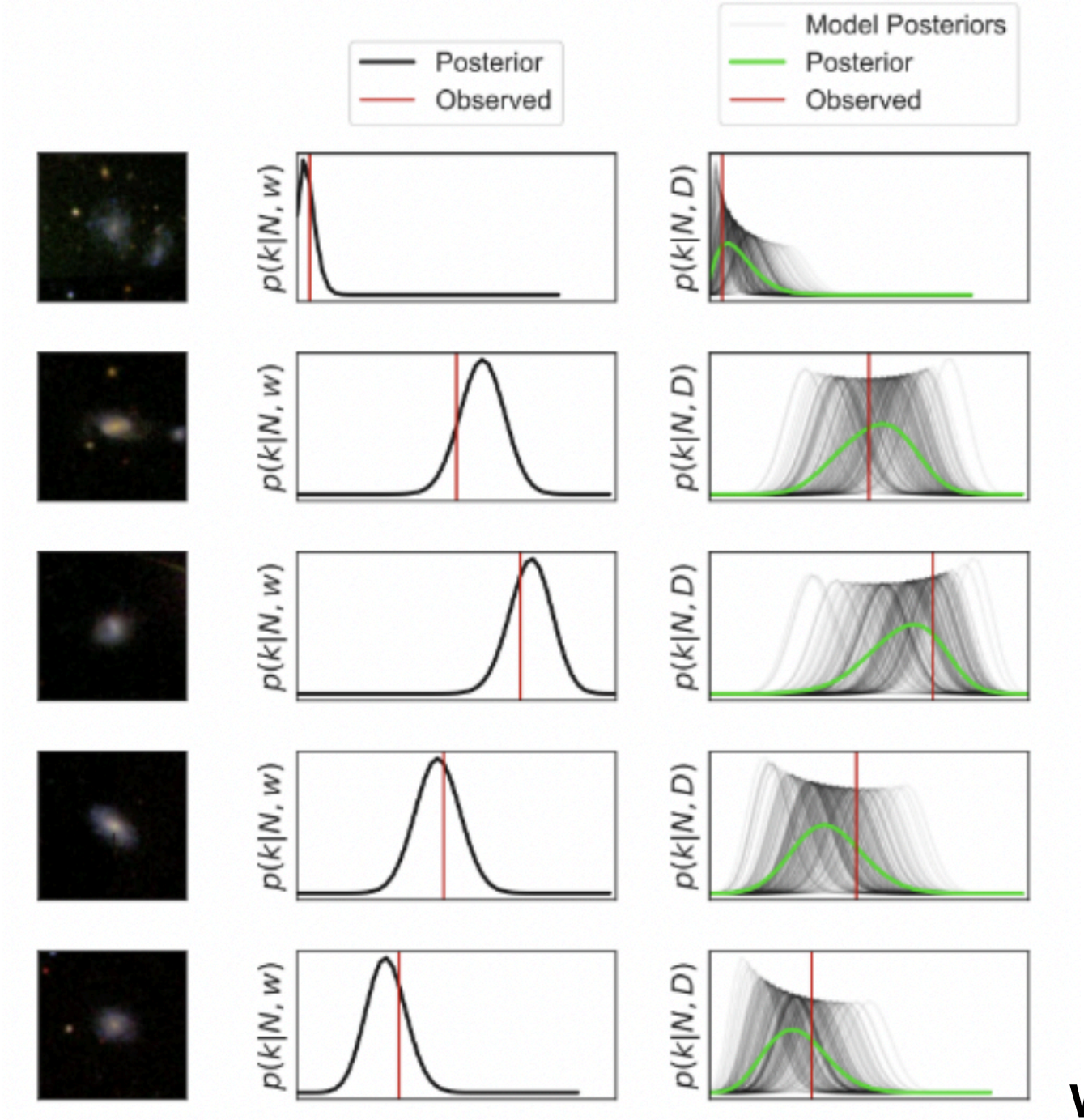
DROPOUT AS EPISTEMIC UNCERTAINTY ESTIMATION

We can treat the many different networks (with different neurons dropped out) as Monte Carlo samples from the space of all available models. This provides mathematical grounds to reason about the model's uncertainty and, as it turns out, often improves its performance.

We simply apply dropout at test time, that's all! Then, instead of one prediction, we get many, one by each model. We can then average them or analyze their distributions.

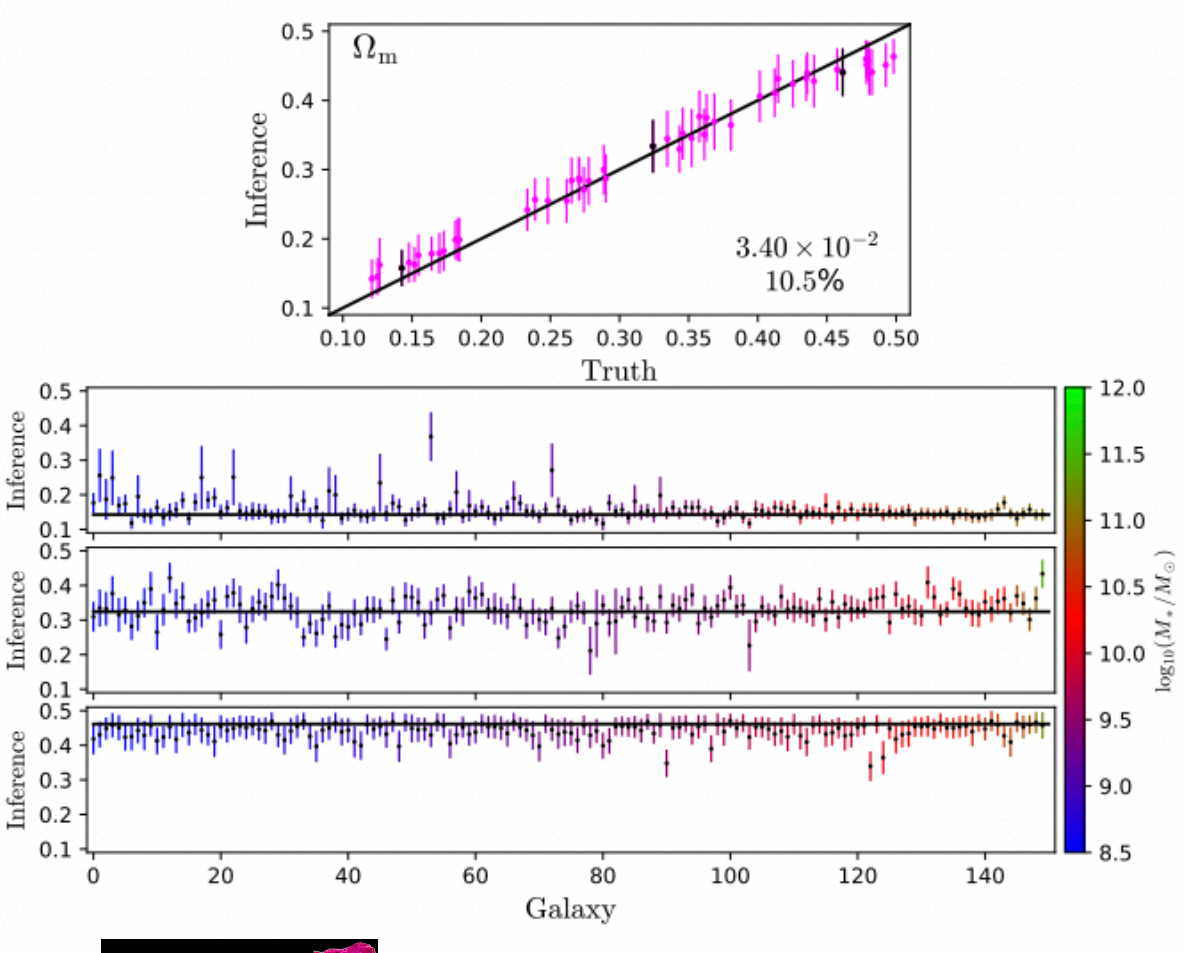
Aleatoric

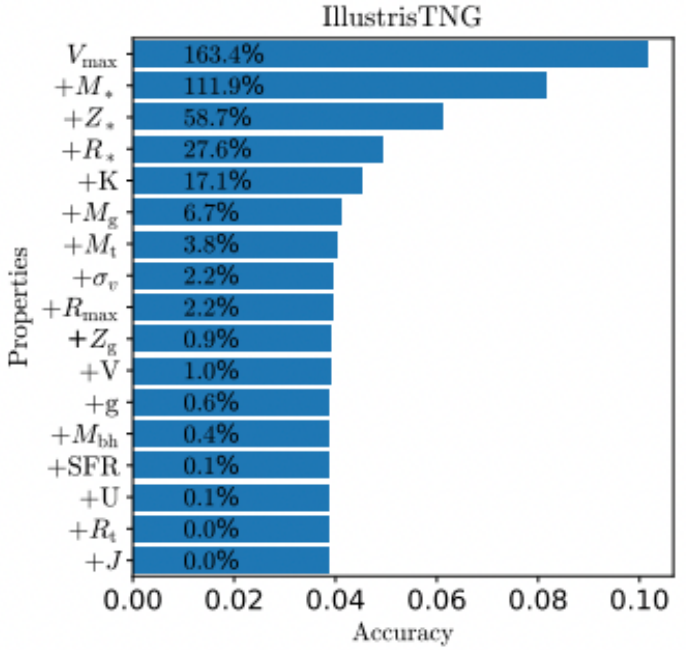
Aleatoric+Epistemic

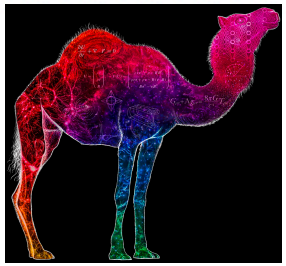


Walmsley+20

Cosmological Inference: beyond summary statistics







(Cosmology and Astrophysics with Machine Learning Simulations)

CAMELS

Villaescusa-Navarro+22

Ravanbakhsh+17, Brehmer+19, Ribli+19, Pan+19, Ntampaka+19, Alexander+20, Arjona+20, Coogan+20, Escamilla-Rivera+20, Hortua+20, Vama+20, Vernardos+20, Wang+20, Mao+20, Arico+20, Villaescusa_navarro+20, Singh+20, Park+21, Modi+21, Villaescusa-Navarro+21ab, Moriwaki+21, DeRose+21, Makinen+21, Villaescusa-Navaroo+22