How to capture uncertainties with Neural Networks

A high level intro of probabilistic deep learning models

Beate Sick

Probabilistic Deep Learning is done in team work

The topic was started 2017 with Elvis and Oliver at ZHAW and with Lisa at UZH and since then a lot is going on…

Outline

- Why are traditional neural network models “not probabilistic”?
- How can a NN model capture the data inherent variation?

  → spoiler: by modeling a parametric **c**onditional **p**robability **d**istribution (CPD)
- How to capture the uncertainty of the parameters that fix the CPD?

  → spoiler: use Bayesian methods to get a distribution of plausible parameter values
- Aleatoric and epistemic uncertainty: The new buzzwords in the field
- Using epistemic uncertainty to identify novel classes in a real world classification task.

Probabilistic Neural Networks for simple regression

Can we predict the blood pressure from the age of a woman?

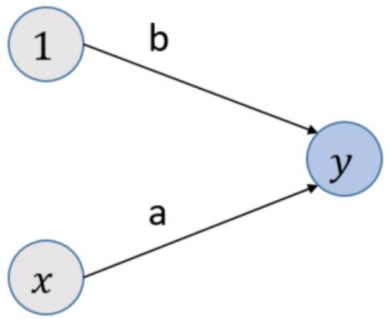
Hypothesis: Knowing the age of a woman helps us to predict the blood pressure.

People say that age is just a state of mind.
I say it's more about the state of your body.

Geoffrey Parfitt

The **sbp** (**s**ystolic **b**lood **p**ressure) tend to increase with **ag**e.

Simple regression via a NN: no probabilistic model in mind

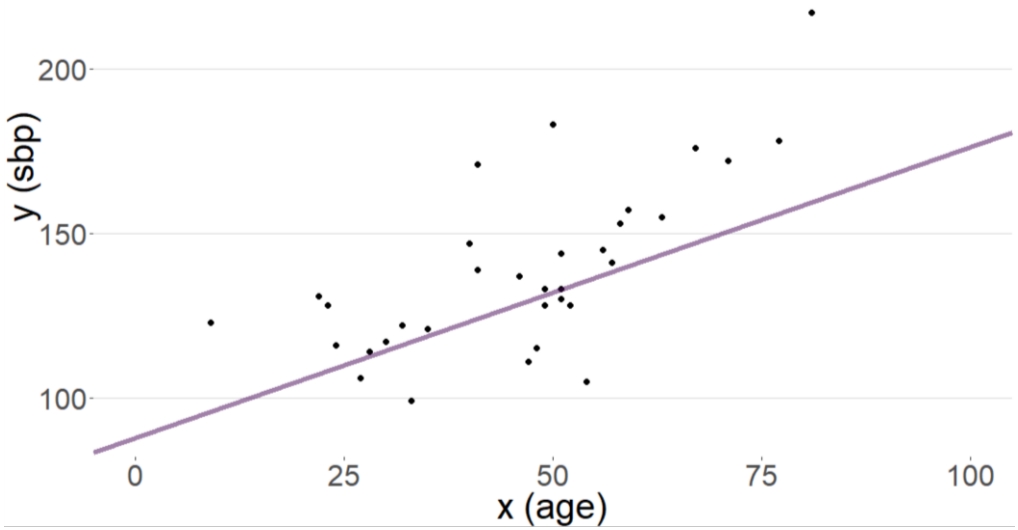



```
model = Sequential()
model.add(Dense(1,input_dim=1))
model.add(Activation('linear'))

opt = optimizers.SGD(lr=0.0004)

model.compile(loss='mean_squared_error',
              optimizer=opt)


for i in range(0,80000):
    model.fit(x=x,y=y,batch_size=33,
                      epochs=1,
                      verbose = 0)
```



$$y(x_i) = a \cdot x_i + b$$

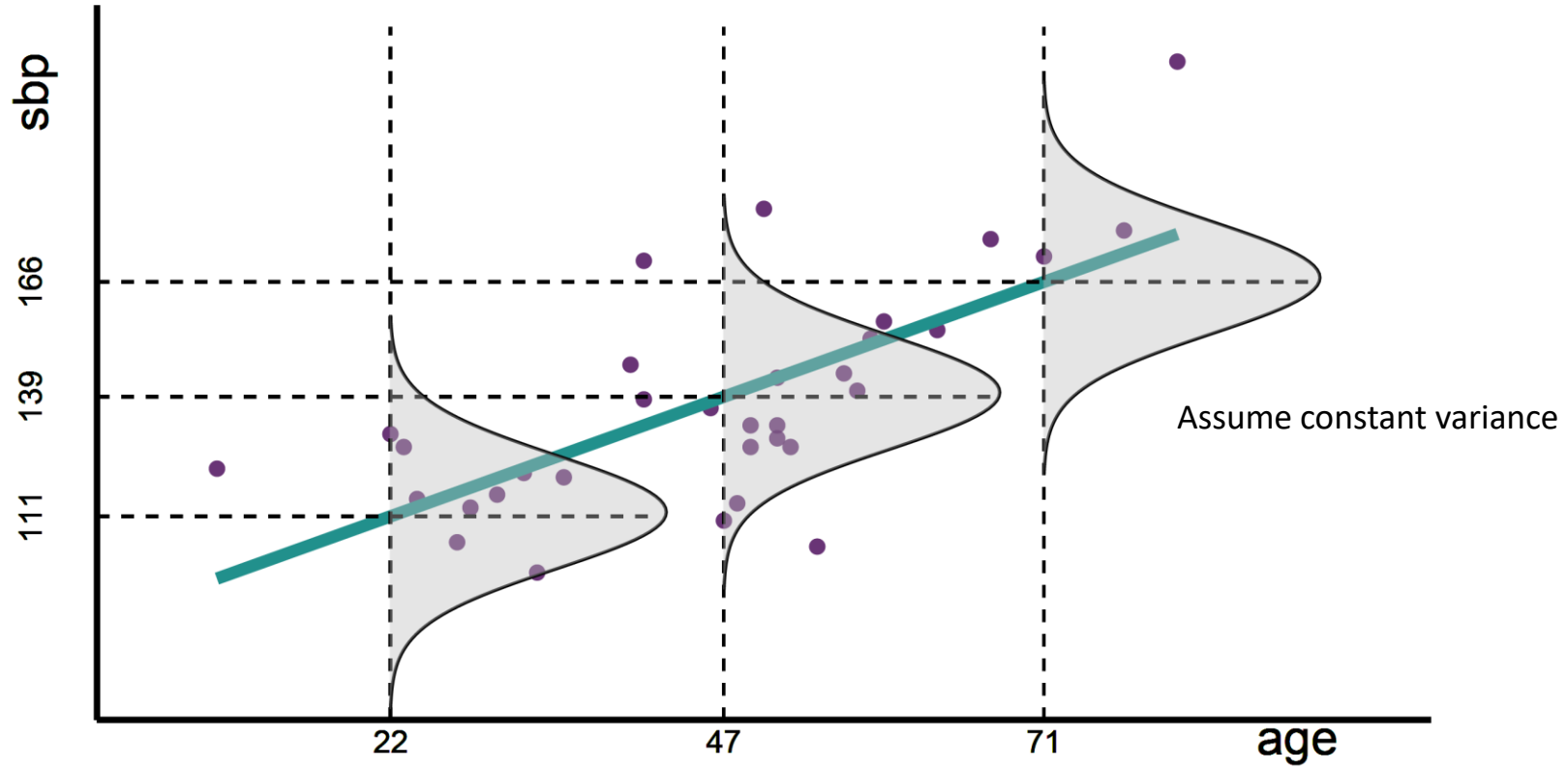
How to express uncertainty in statistics or machine learning?

- Probability is the language of uncertainty
- Probability distributions are the tool to quantify uncertainties

CPD

Conditional Probability Distribution

Fitting a probabilistic model allows to capture variations



For each age (x-position) we have fitted a whole distribution of possible sbp values (y-values or response values – here systolic blood pressure).

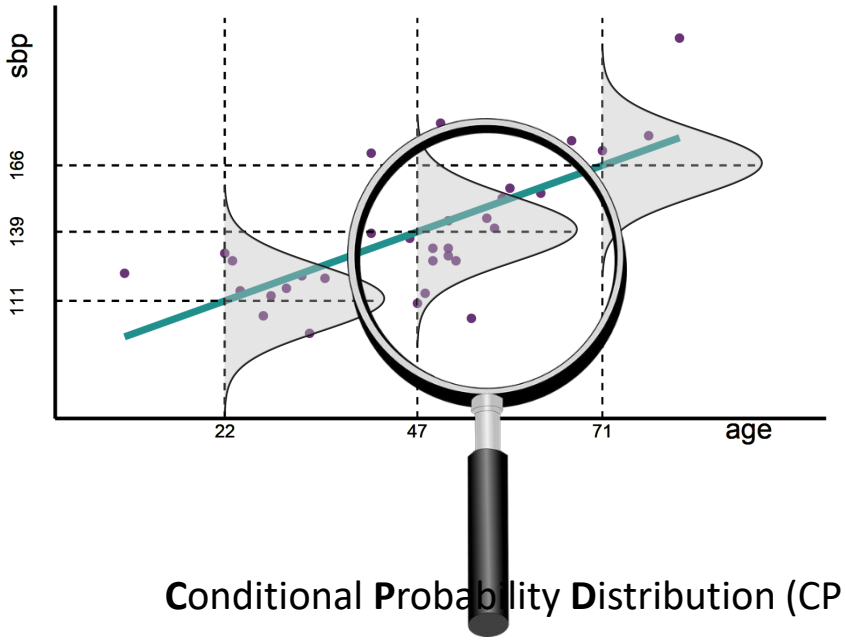
Simple linear regression models the μ_x-parameter of $N(\mu_x, \sigma^2)$

Traditional view
with focus on means and residuals:

$$y_i = \beta_0 + \beta_1 \cdot x_{i1} + \varepsilon_i \quad , \quad \varepsilon_i \sim N(0, \sigma^2)$$

We model the expected value:

$$\mu_{x_i} = a \cdot x_i + b$$



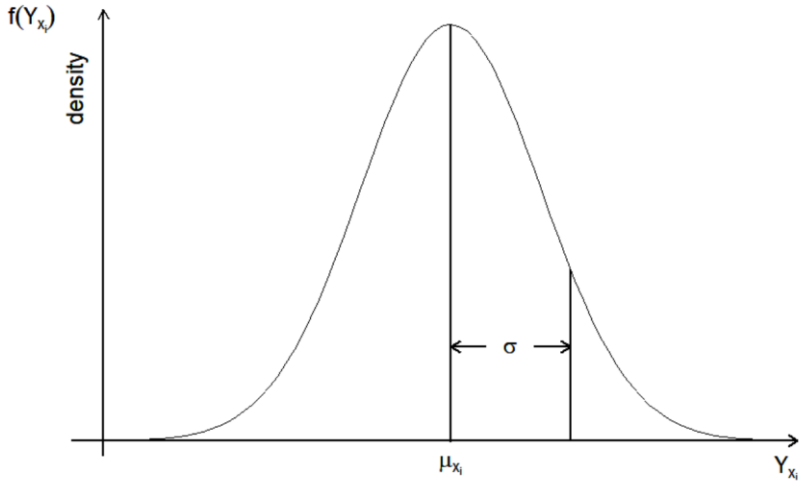
Conditional **P**roba**b**ility **D**istribution (CPD)

Slightly changed view
with focus on conditional distributions:

$$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2)$$

We model the μ_{x_i}-parameter of the CPD:

$$\mu_{x_i} = a \cdot x_i + b$$



Bird view on a probabilistic prediction model

Input

Artificial Intelligence ;-)

Outcome distribution

Input X

Parameter for CPD

x_i

μ_{x_i} σ_{x_i}

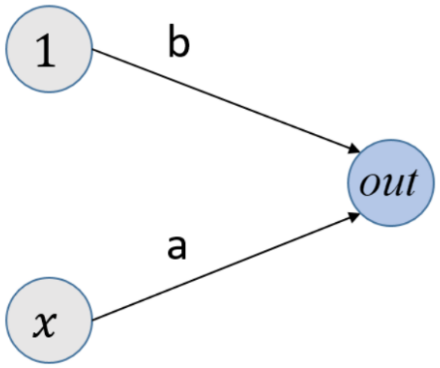
Using a neural network for simple linear regression

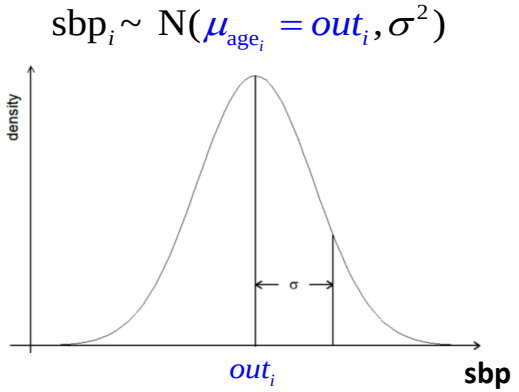
Input

Artificial Intelligence ;-)

Outcome distribution

Input: age_i of woman i

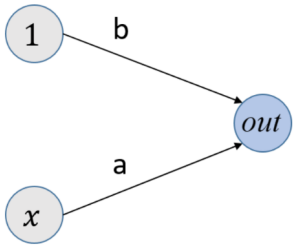




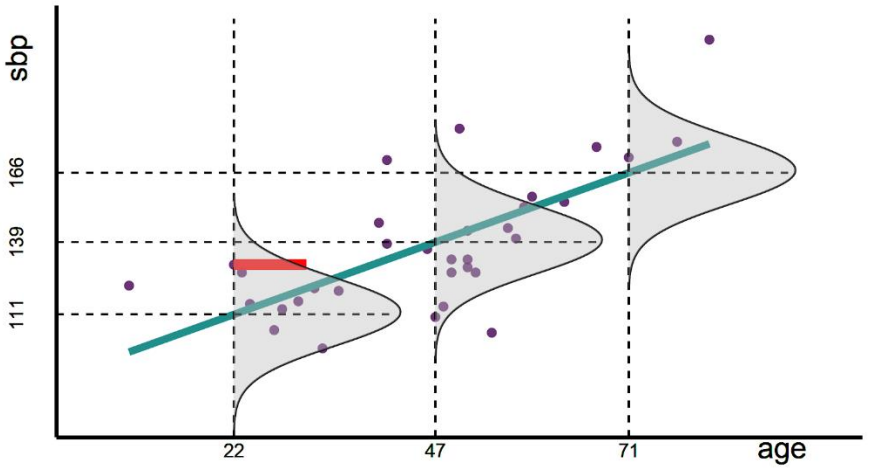
$$out_i = \mu_{x_i} = a \cdot x_i + b$$

Fitting the CPD parameter via Maximum Likelihood

How to find the weights in the NN that yield good μ-estimates?



$$Y_{x_i} \sim N\left(\mu_{x_i} = out_{x_i}, \sigma^2\right)$$



Maximum Likelihood principle:

$$\mathbf{w}_{\mathrm{ML}} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^{n} f\left(y_i \mid x_i, \mathbf{w}\right)$$

$$w = argmin_w \left\{ \sum_{i=1}^{n} - log\, f(y_i | x_i, w) \right\}$$

$$w = argmin_w \left\{ \sum_{i=1}^{n} - log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(y_i-\mu_{x_i})^2}{2\sigma^2}}\right) \right\}$$

$$w = argmin_w \left\{ \sum_{i=1}^{n} - log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(\mu_{x_i}-y_i)^2}{2\sigma^2} \right\}$$

$$w = argmin_w \left\{ \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\mu_{x_i} - y_i)^2 \right\}$$

$$(\hat{a}, \hat{b})_{\mathrm{ML}} = \underset{a,b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(a \cdot x + b - y_i\right)^2$$

gradient descent

$\hat{a}$ $\hat{b}$

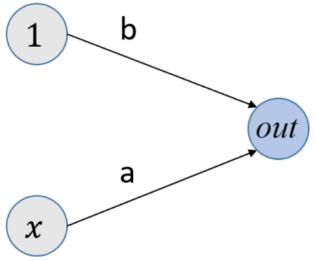
Analytically

$$\hat{a} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \text{and} \quad \hat{b} = \bar{y} - \hat{a} \cdot \bar{x}$$

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n} \left(\hat{a} \cdot x + \hat{b} - y_i\right)^2$$

Few code lines to fit the model

```
#Build model.
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(1,input_shape=(1,)))
model.compile(loss='mse', optimizer="adam", metrics=['accuracy'])
model.summary()
model.fit(x,y,batch_size=150,epochs=12000,verbose=0)
```



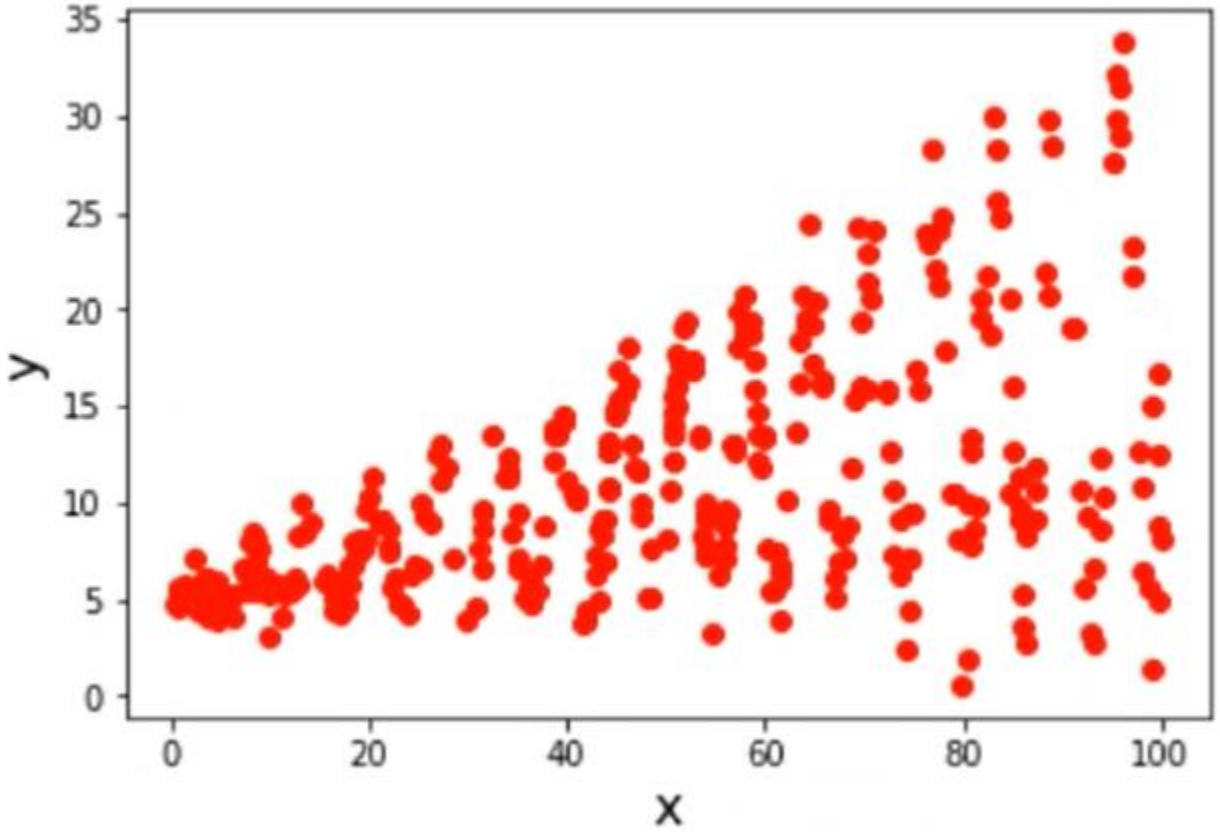

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
<tensorflow.python.keras.callbacks.History at 0x7f3aa5aedc50>
```

```
model.get_weights()
```

```
[array([[0.13853021]], dtype=float32), array([4.9734774], dtype=float32)]
```

As machine learner we are interested in the outcome prediction, not in the parameter values!

How to do linear regression with non-constant variance?



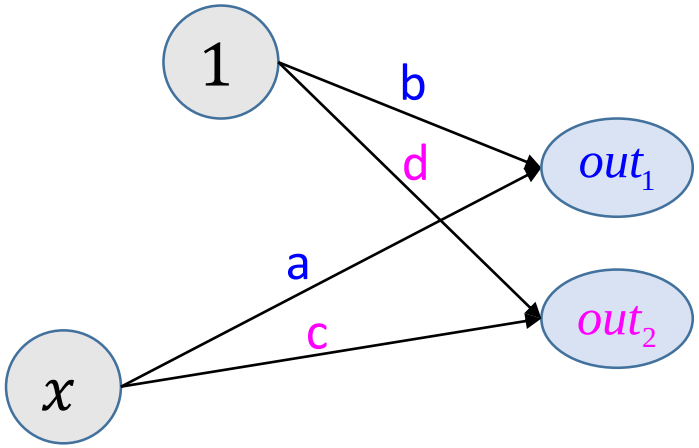
Idea 1: Transform data so that variance is stable.

Idea 2: Fit a conditional probability distribution where both parameters (μ, σ^2) depend on x.

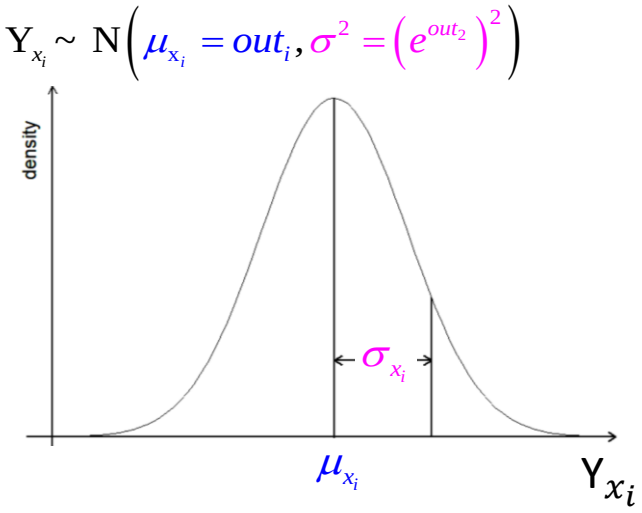
How to do linear regression with non-constant variance?

Input

Neural Network



Outcome distribution



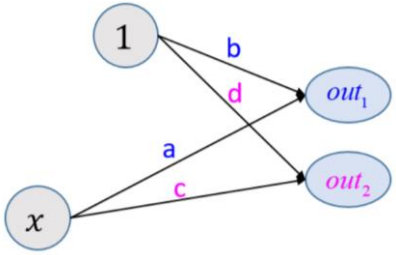
$$out_{1_i} = a \cdot x + b$$

$$out_{2_i} = c \cdot x_i + d$$

$$\mu_{x_i} = out_{1_i}$$

$$\sigma_{x_i} = e^{out_{2_i}}$$

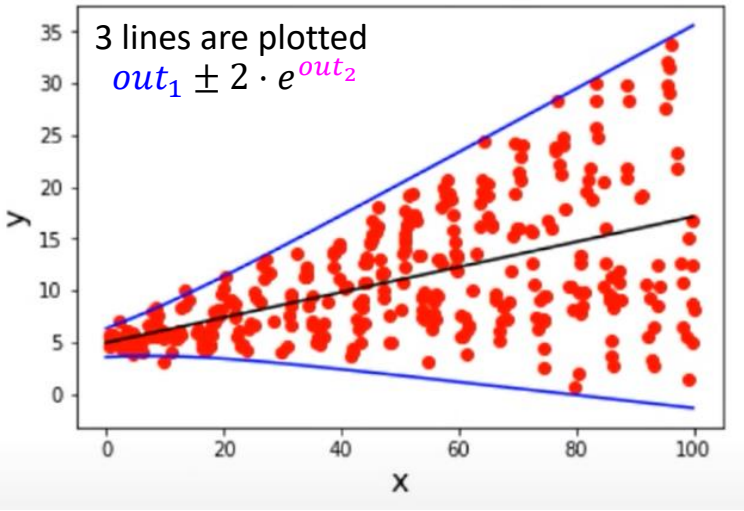
How to find the weights in the NN that yield good μ-estimates?



$$\mu_{x_i} = out_{1_i}$$

$$\sigma_{x_i} = e^{out_{2_i}}$$

$$Y_{x_i} \sim N\left(\mu_{x_i} = out_i, \sigma^2 = \left(e^{out_2}\right)^2\right)$$



Maximum Likelihood principle:

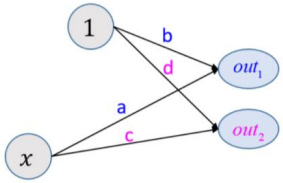
$$\mathbf{w}_{\mathrm{ML}} = \underset{\mathbf{w}}{\operatorname{argmin}} \prod_{i=1}^{n} f\left(y_i \mid x_i, \mathbf{w}\right)$$

$$w = argmin_w \left\{ \sum_{i=1}^{n} -log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(\mu_{x_i} - y_i)^2}{2\sigma^2} \right\}$$

gradient descent

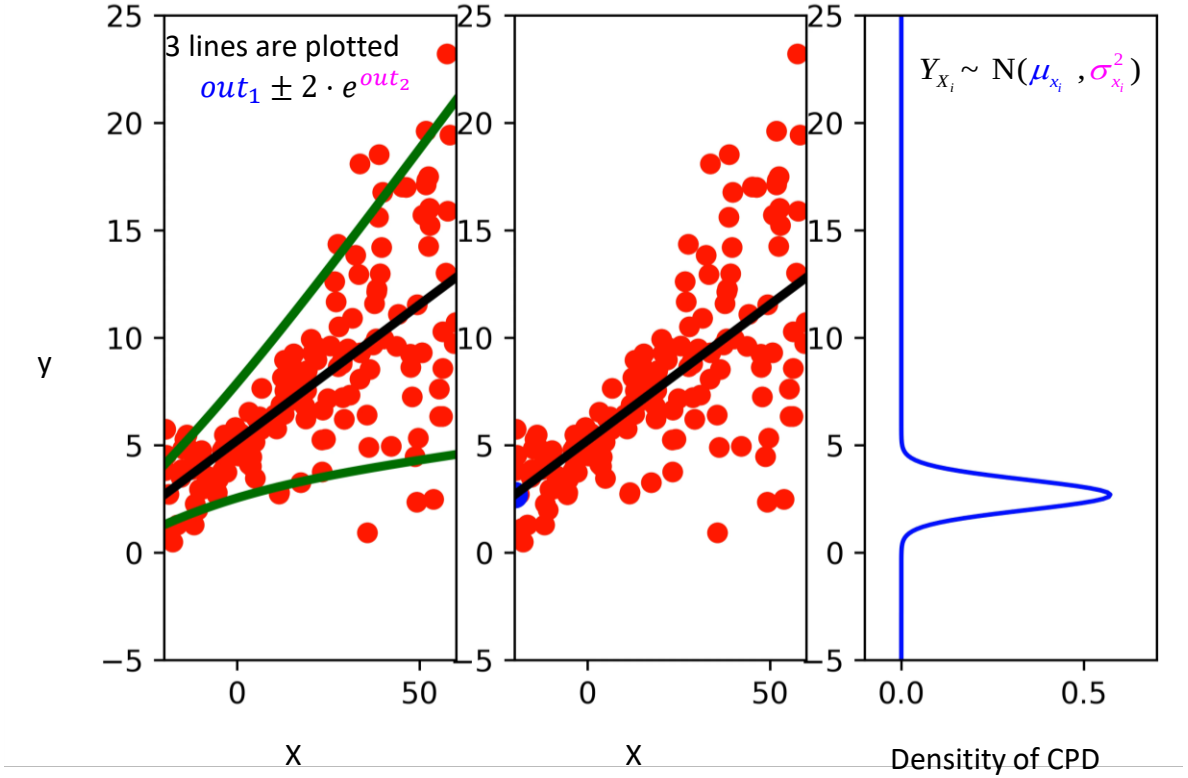
$$\hat{a}, \hat{b}, \hat{c}, \hat{d}$$

Outcome CPD is given by its parameter values



$\mu_{x_i} = out_{1_i}$

$\sigma_{x_i} = e^{out_{2_i}}$



In traditional linear regression we only model the μ-parameter of the CPD as dependent on x and assume that the variance σ^2 does not depend on x.

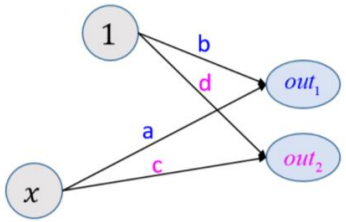
$$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2_{x_i}) \qquad \mu_{x_i} = a \cdot x_i + b \qquad \sigma_{x_i} = e^{c \cdot x + d}$$

depicted as blue dot

Credits for animated gif: https://crumplab.github.io/statistics/gifs.html

Lets use the same NN architecture for a different data set

$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2_{x_i})$

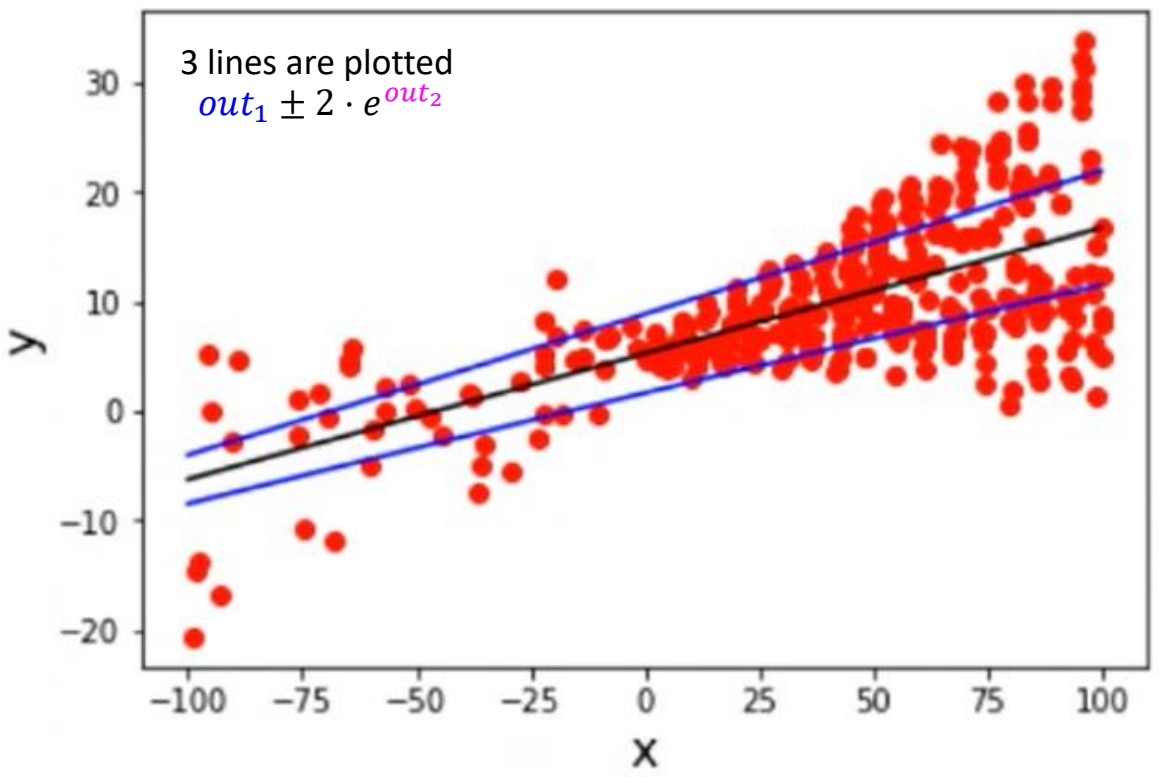


$\mu_{x_i} = out_{1_i}$

$\sigma_{x_i} = e^{out_{2_i}}$

$\mu_{x_i} = a \cdot x_i + b$

$\sigma_{x_i} = e^{c \cdot x + d}$



What is going wrong?

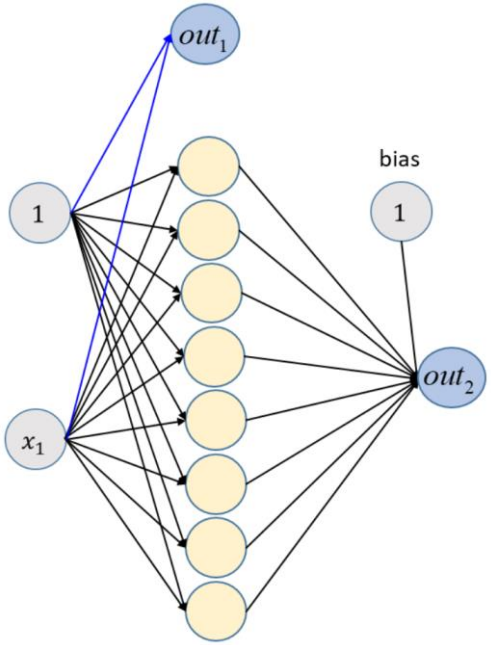
Without hidden layer out_2 is a linear function of x
→ Sigma is a monotone function of x!

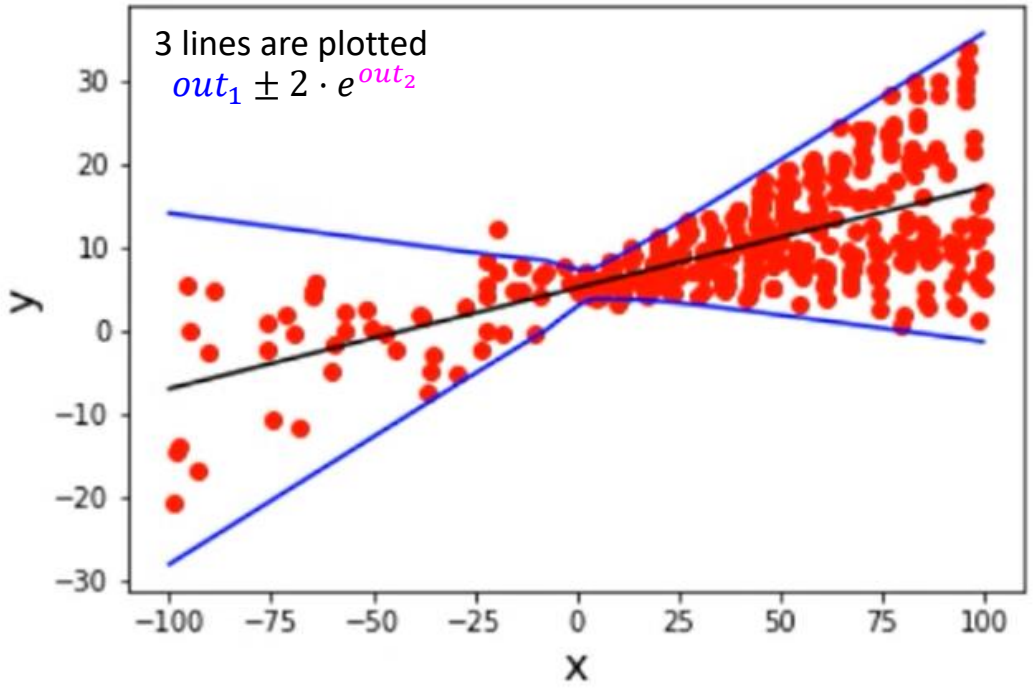
NN architecture for a non-monotone relation of input and variance

$$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2_{x_i})$$

$\mu_{x_i} = out_{1_i}$

$\sigma_{x_i} = e^{out_{2_i}}$



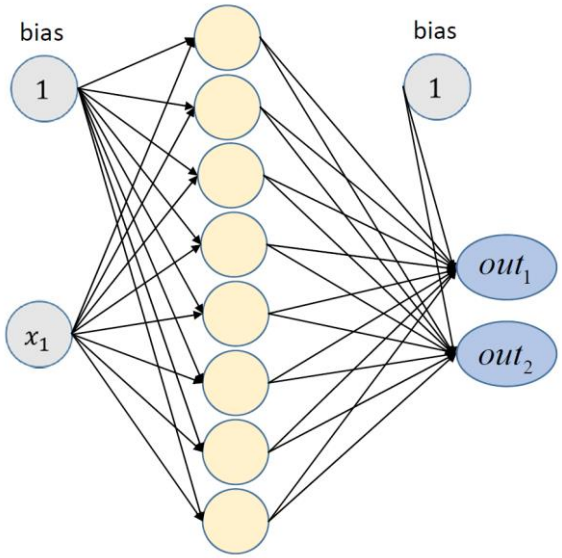


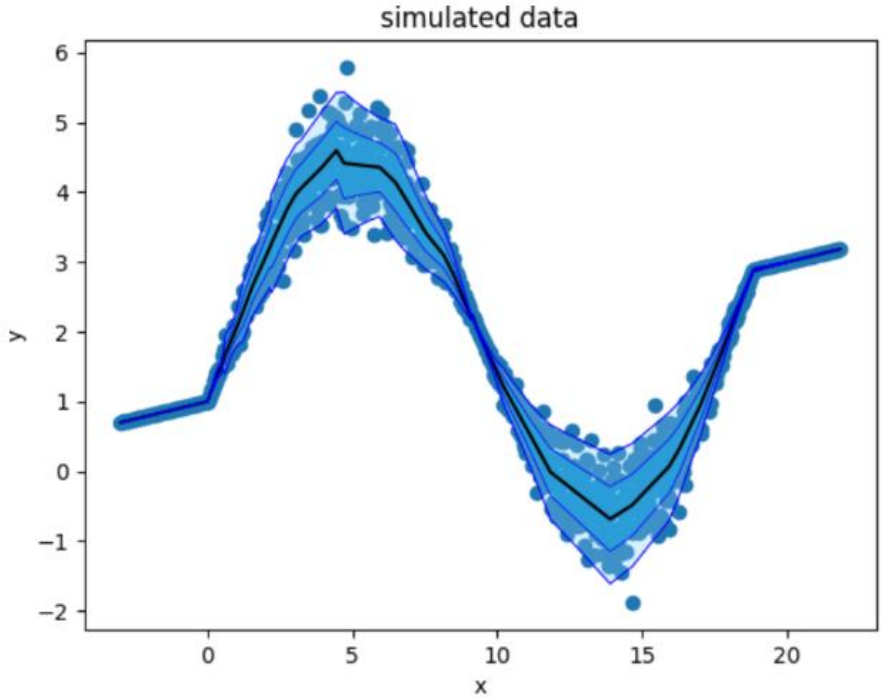
NN architecture to fit non-linear heteroscatic data

$$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2_{x_i})$$

$$\mu_{x_i} = out_{1_i}$$

$$\sigma_{x_i} = e^{out_{2_i}}$$





Parameter uncertainty via Bayesian approach

Recap: Bayesian models have distributions over their parameter

Bayesian fit of a polynomial function with 6 parameters

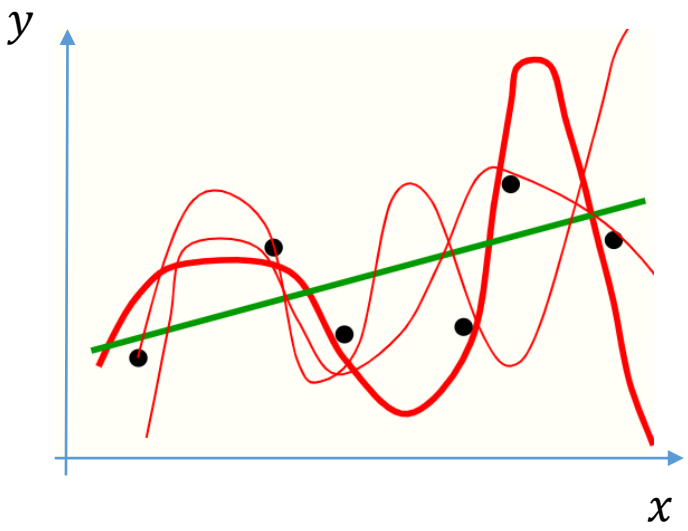
$$\hat{y} = \hat{\omega}_0 + \hat{\omega}_1 \cdot x + \hat{\omega}_2 \cdot x^2 + \hat{\omega}_3 \cdot x^3 + \hat{\omega}_4 \cdot x^4 + \hat{\omega}_5 \cdot x^5$$

with

$\hat{\omega}_0 \sim \text{Distribution}_{\boldsymbol{\theta}_0}$, e.g. $N\left(\mu_0, \sigma_0^2\right)$

...

$\hat{\omega}_5 \sim \text{Distribution}_{\boldsymbol{\theta}_5}$, e.g. $N\left(\mu_5, \sigma_5^2\right)$



A fitted Bayesian model represents not 1 polynomial but a distribution of polynomials.

Frequentist's vs Bayesian's answer when asked to do prediction

Frequentist's strategy:
You can only use a complex model if you have enough data!

Bayesian's strategy:

Use the model complexity you believe in.

Do not just use the best fitting model.

Do use the full posterior distribution over parameter settings leading to vague predictions since many different parameter settings have significant posterior probability.

$$p(y^*|x^*,X,Y) = \int p(y^*|x^*,\omega) \cdot p(\omega|X,Y)\, d\omega$$

prediction via marginalization over ω:

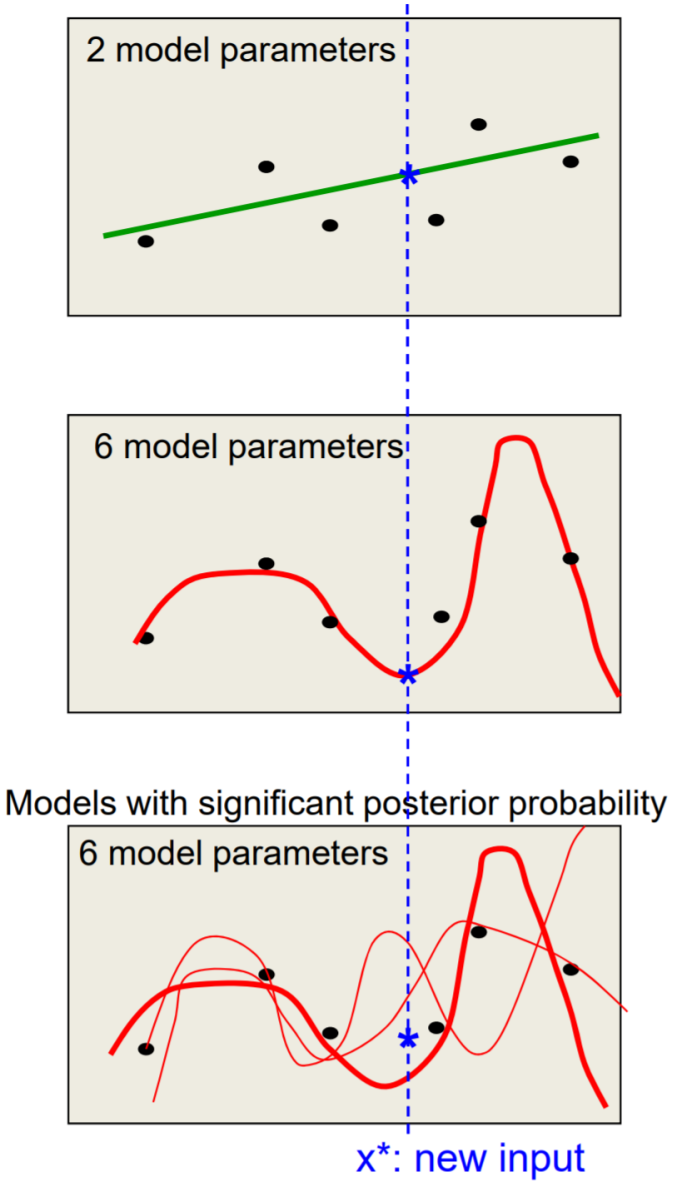


Image credits: Hinton coursera course

Bayesian framework

- In Bayesian Modeling we define a prior distribution over the parameter W: $W_i \sim N(0, I)$ defining p(w_i)

- For regression NN we have the likelihood:

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mu_{x_i} - y_i)^2}{2\sigma^2}}$$

- Given a dataset X, Y we then look for the posteriori distribution capturing the most probable model parameter given the observed data

likelihood prior

$$p(\omega \mid X, Y) = \frac{p(Y \mid \omega, X) \cdot p(\omega)}{p(Y \mid X)}$$

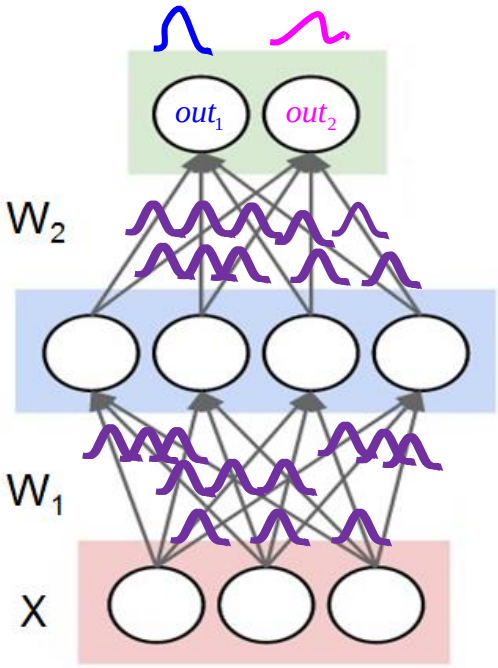
normalizer=marginal likelihood

not possible to derive exactly in case on NN

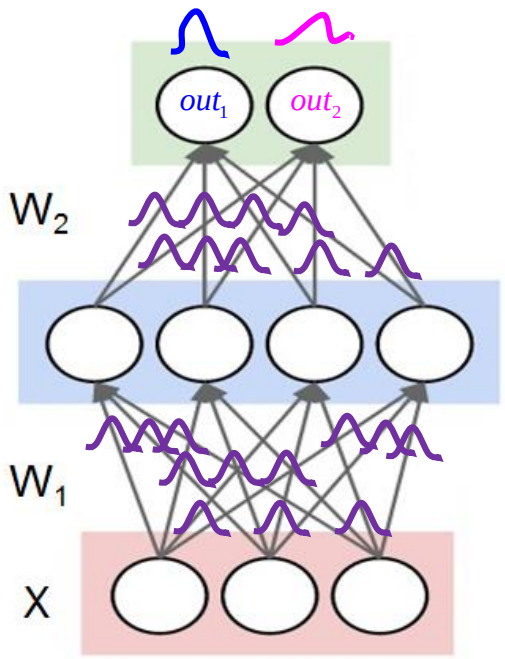
$$Y_{X_i} = (Y \mid X_i) \sim N(\mu_{x_i}, \sigma^2_{x_i})$$

$$\mu_{x_i} = out_{1_i}$$

$$\sigma_{x_i} = \ln(1 + e^{out_{2_i}})$$

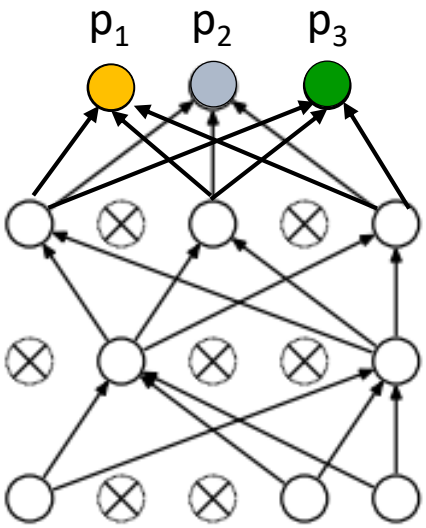


Approximate Bayesian NN by variational inference



TFp

→ We should sample from weight distribution during test time



MC Dropout

Randomly drop nodes in each run

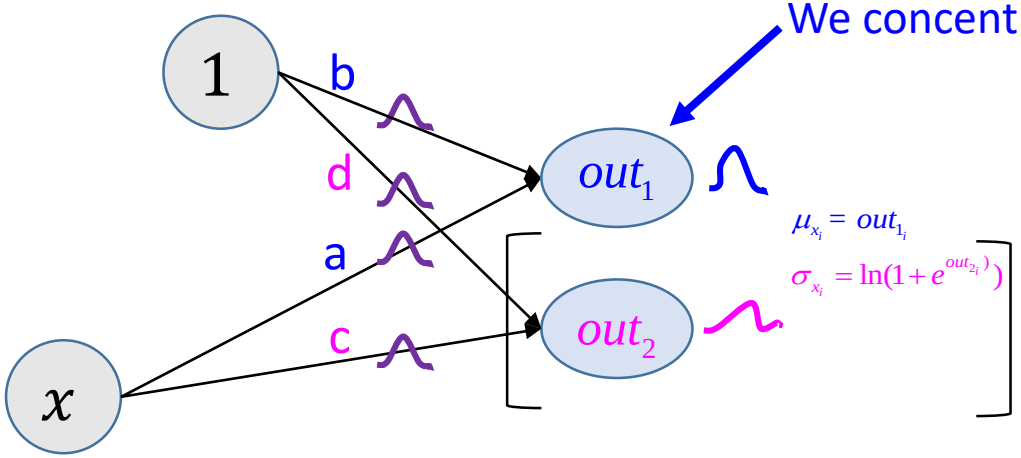
→ Ususally done during training

Oliver, Elvis and I plan to give additional BBSs on the theory and frameworks of Bayesian DL models.

There was a BBS on Dropout as Bayesian Approximation method: https://tensorchiefs.github.io/bbs/files/dropouts-brownbag.pdf

*Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning https://arxiv.org/abs/1506.02142

Bayesian NN provide uncertainty on the parameter values



We concentrate on the uncertainty of $\mu_{x_i} = out_{1_i}$

Predictive distribution

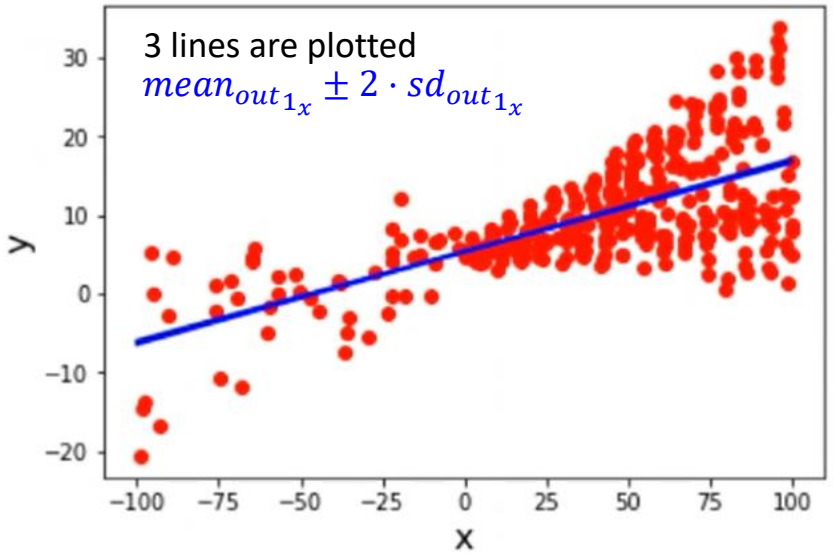
$$p(\mu \mid x, w = (a, b, c, d))$$

Can be approximated by sampling realized by repeated predictions

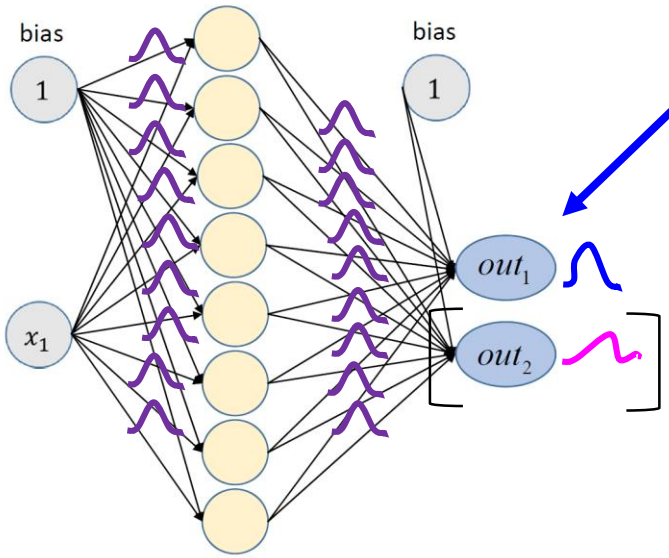
To indicate the uncertainty about the μ_{x_i} we plot here a curve for $mean_{out_{1_x}}$ and two curves for $mean_{out_{1_x}} \pm 2 \cdot sd_{out_{1_x}}$

We would expect higher uncertainty, if we have less data – here the uncertainty is incredible small – why?

There is no hidden layer → linear fit over whole range, we can use all points to estimate this line (only 2 df)!



NN with hidden layers do not impose a linear model



We concentrate on the uncertainty of μ

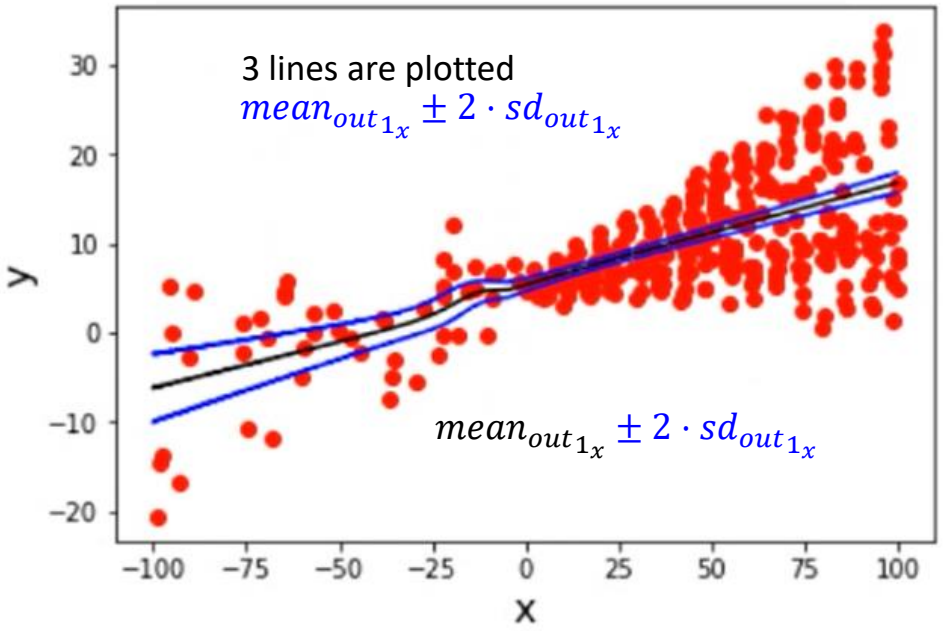
Predictive distribution

$$p(\mu \mid x, w = (a,b,c,d))$$

Can be approximated by sampling realized by repeated predictions

Hidden layers
→ no constrain that allow to borrow infromatin from whole range $meat_{out1_x}$ needs to be estimated locally from few data
→ large uncertaincy about μ_{x_i}

With 2 hidden layers (30,20)



Aleatoric uncertainty

&

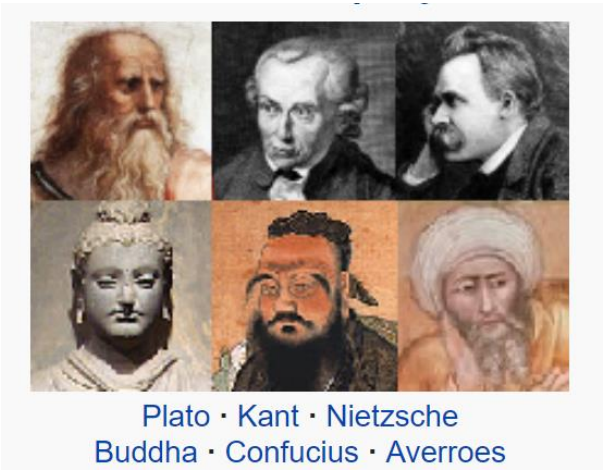
Epistemic uncertainty

Getting philosophical – Epistemology and Aleatoricism

Epistemology is the branch of philosophy concerned with the theory of knowledge.

The word derives from the Greek word epistēmē, meaning 'knowledge

https://en.wikipedia.org/wiki/Epistemology



Aleatoricism is the incorporation of chance into the process of creation.

The word derives from the Latin word alea, the rolling of dice.

https://en.wikipedia.org/wiki/Aleatoricism
https://en.wikipedia.org/wiki/Alea_iacta_est

Alea iacta est

A random experiment with two kinds of uncertainties

- We pull a poker chip out of a bag with mixed poker chips (black or blue) and toss it
  - Sometimes we get a black chip and sometimes a blue chip
  - In 50% of the tosses we see the side with the emoji pic otherwise the backside

1) Pull a chip

2) Toss the chip

Observe color and face of the chip

See http://www.stat.columbia.edu/~gelman/stuff_for_blog/ohagan.pdf

A random experiment with two kinds of uncertainties

1) Pull a chip

2) Toss the chip

Observe color and face of the chip

- The uncertainties on the color and the face of the drawn and tossed chip are different
  - w/o any knowledge we would assign a probability of 0.5 on both colors and faces
  - By collecting data we gain knowledge about the color probability: **epistemic**
  - The random tossing process gives always a 50% chance to both faces: **aleatoric**

See http://www.stat.columbia.edu/~gelman/stuff_for_blog/ohagan.pdf

"Aleatoric" and "Epistemic" in probabilistic models - my understanding

«aleatoric uncertainty» = inherent data uncertainty

due to the stochastic process that generates the data

captured by the fitted parametric distribution (CPD)

→ cannot be reduced by providing more training data

«epistemic uncertainty» = uncertainty about the parameter values

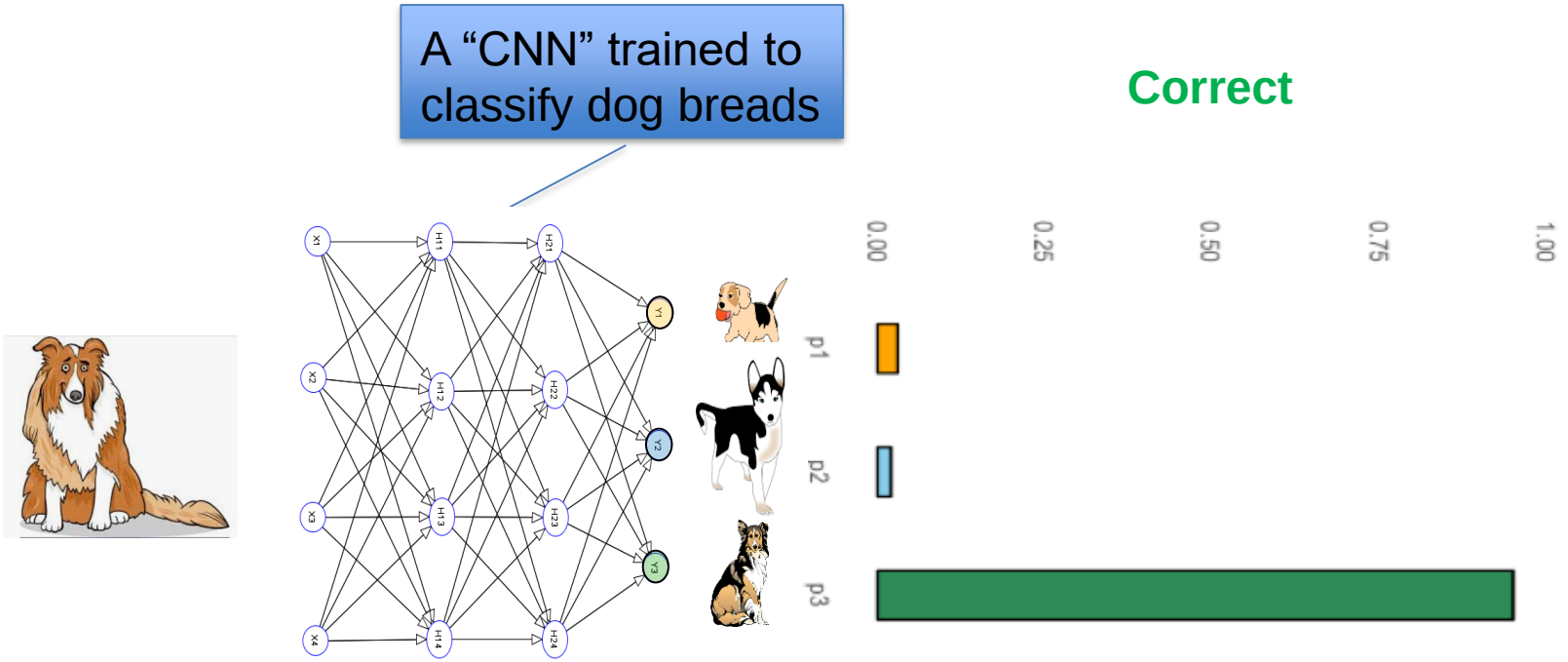
Need to learn about the composition of the bag

captured by CIs or predictive distributions

→ can be reduced by providing more training data

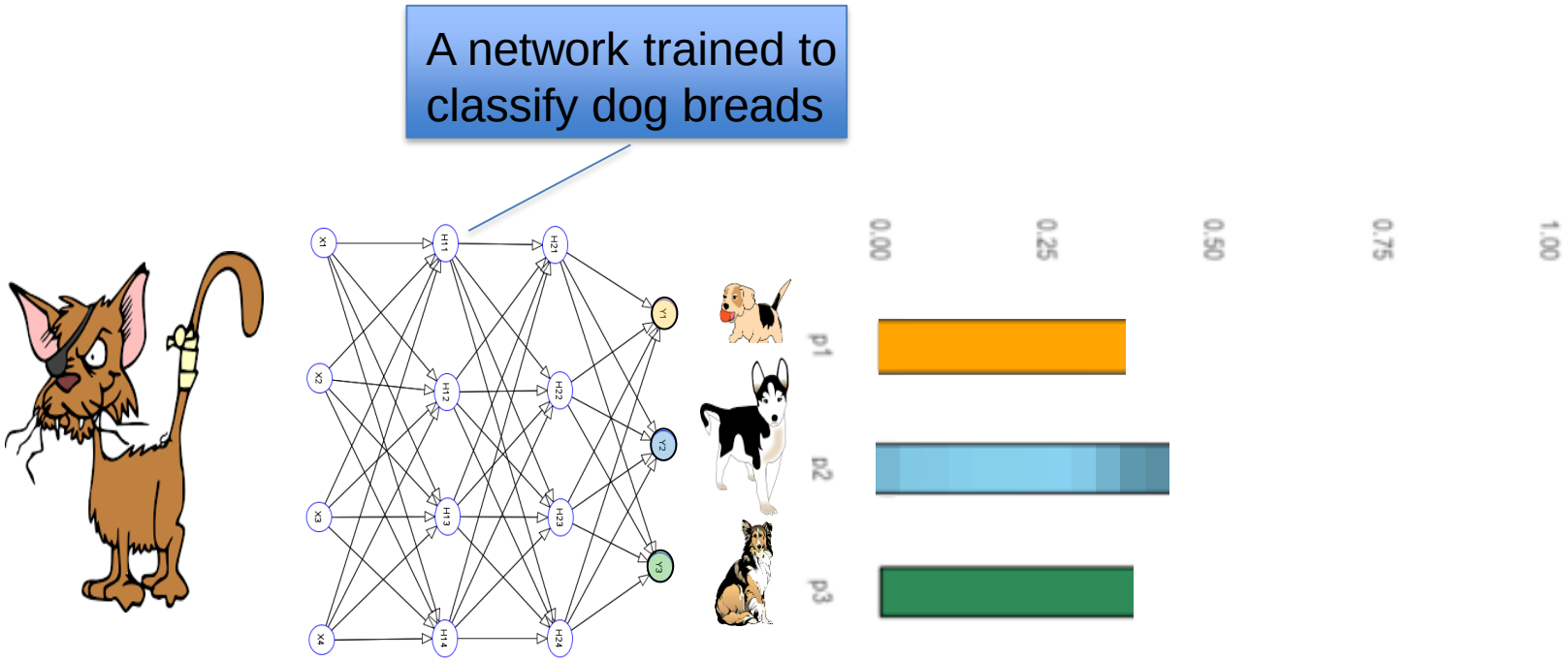
Probabilistic Deep Learning for image classification

DL revolutionized the field of image classification



We fit a probabilistic model:
We got a CPD (multinomial) which captures the aleatoric variablility.
The parameter (p_1, p_2, p_3) of the CPD are estimated by a NN.

What happens if we present an image of a novel class to the CNN



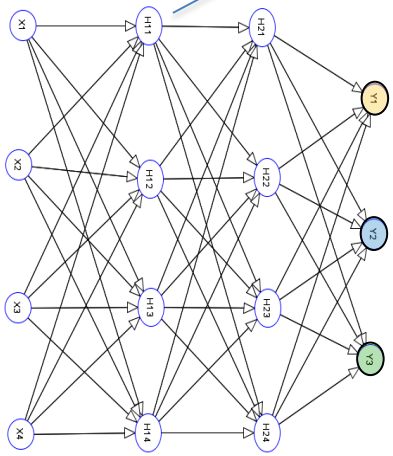
This is what you would expect, right?

Best would be all zero
(not possible due to softmax)

Traditional Deep NN tend to be over-confident…

You call me a collie ? #@*$!! Are you serious?

A network trained to classify dog breads

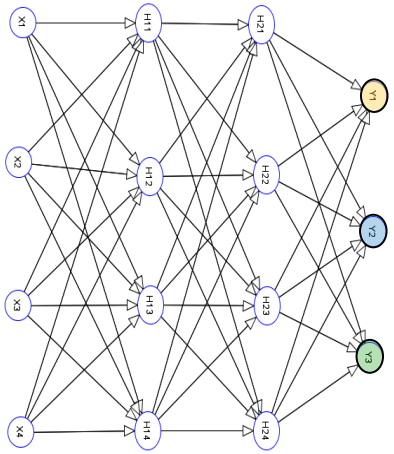


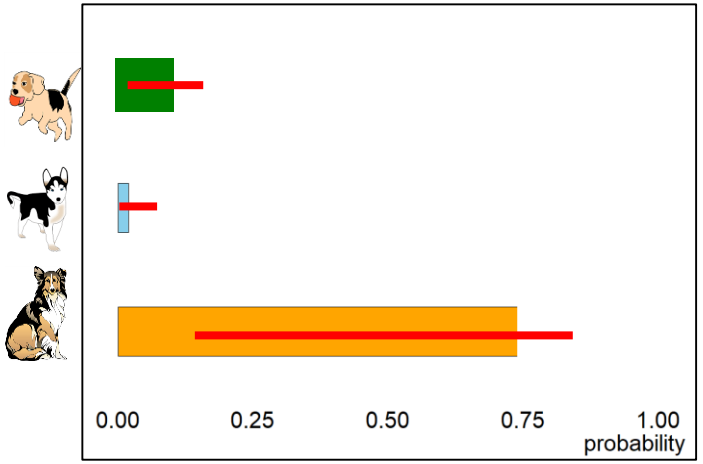
Plain wrong

This is what you get!

Remedy?: Bayesian Approaches to DL

The epistemic (parameter) uncertainty is missing

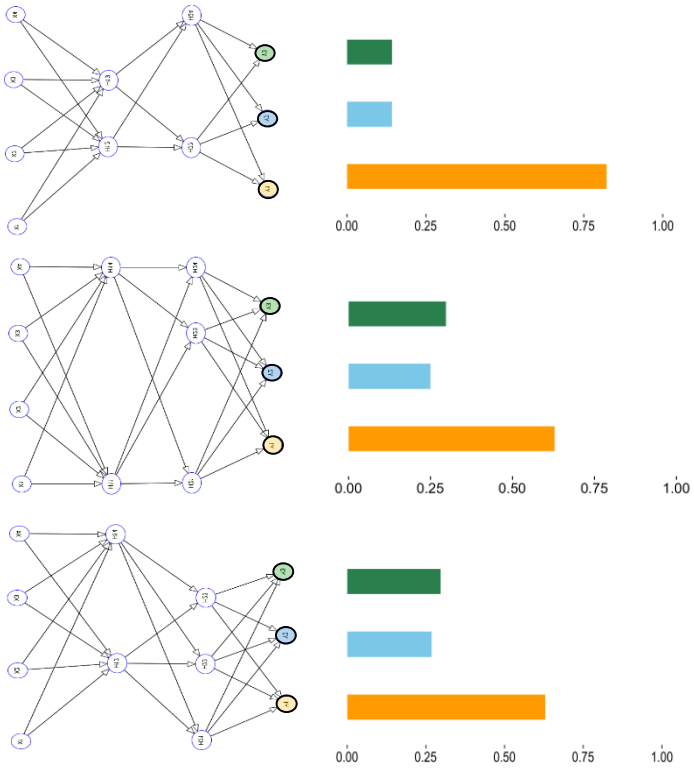




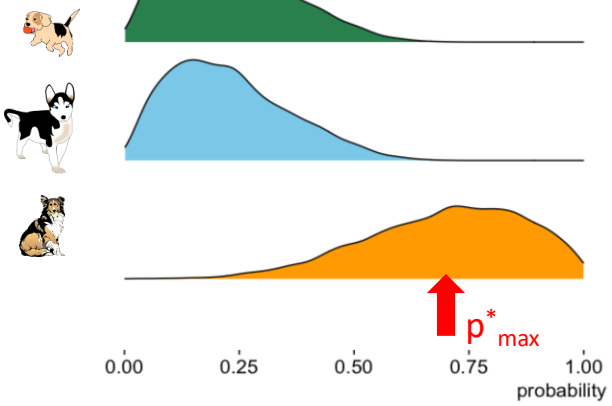
We need some error bars!

MC probability prediction

Many Dropout Runs in forward pass



use dropout also during prediction



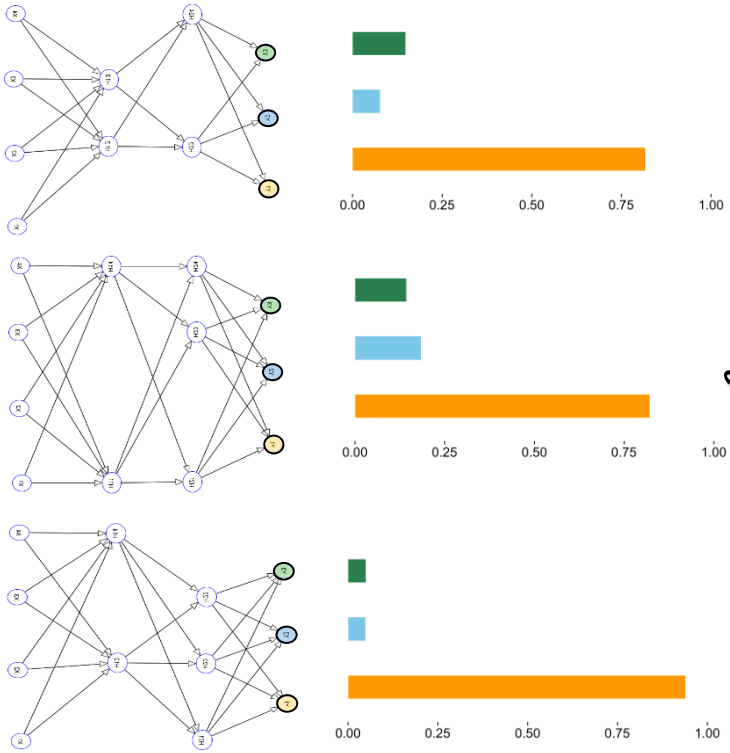
CNN predicts class “collie” but with high uncertainty

...

Remark: Mean of marginal give components of mean in multivariate distribution.

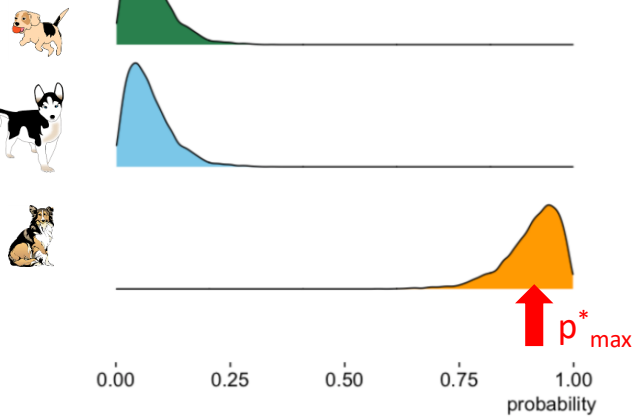
MC probability prediction provides epistimic uncertainty

Many Dropout Runs in forward pass



...

use dropout also during prediction

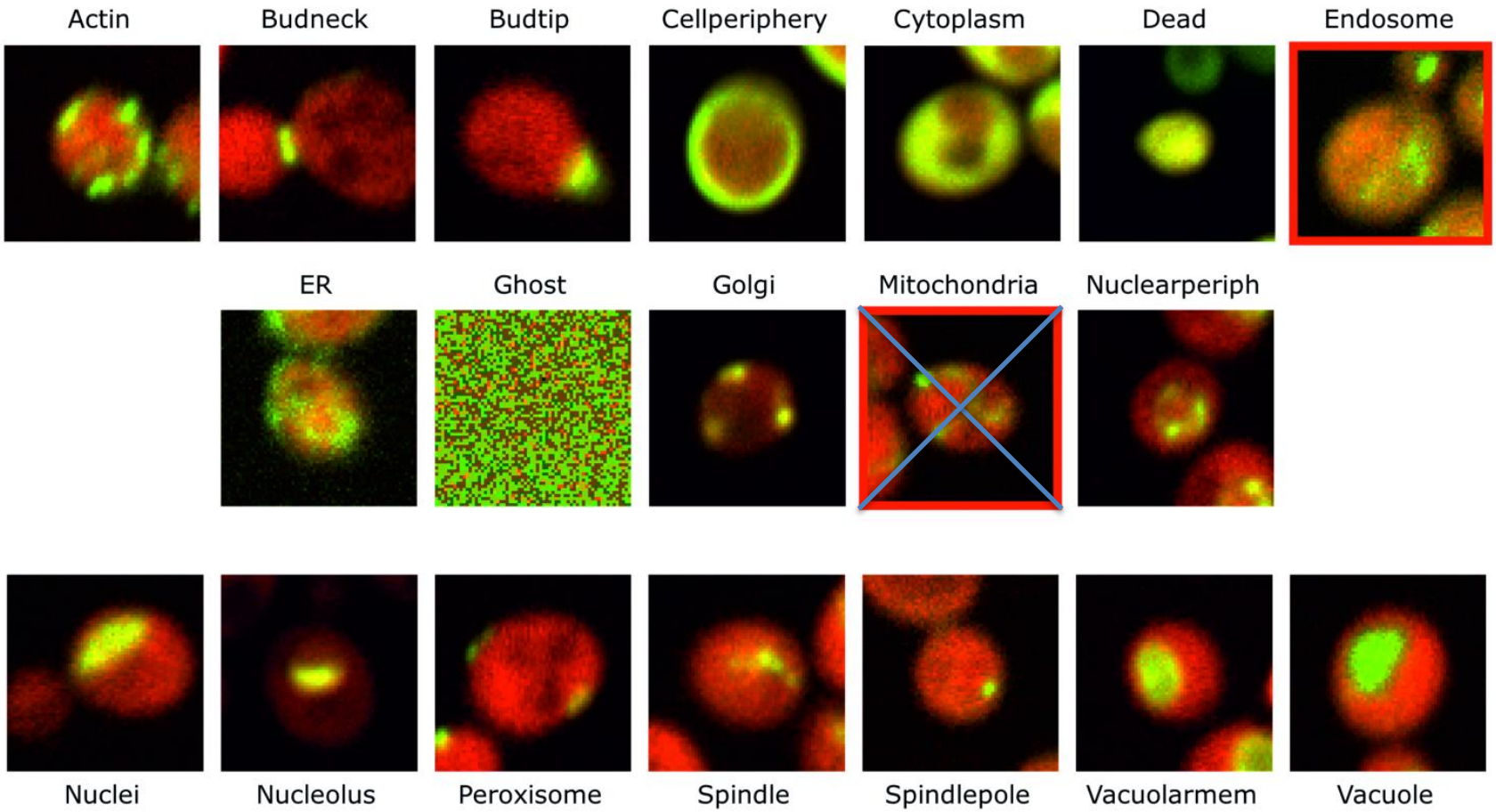


CNN predicts class “collie” this time with low uncertainty

Remark: Mean of marginal give components of mean in multivariate distribution.

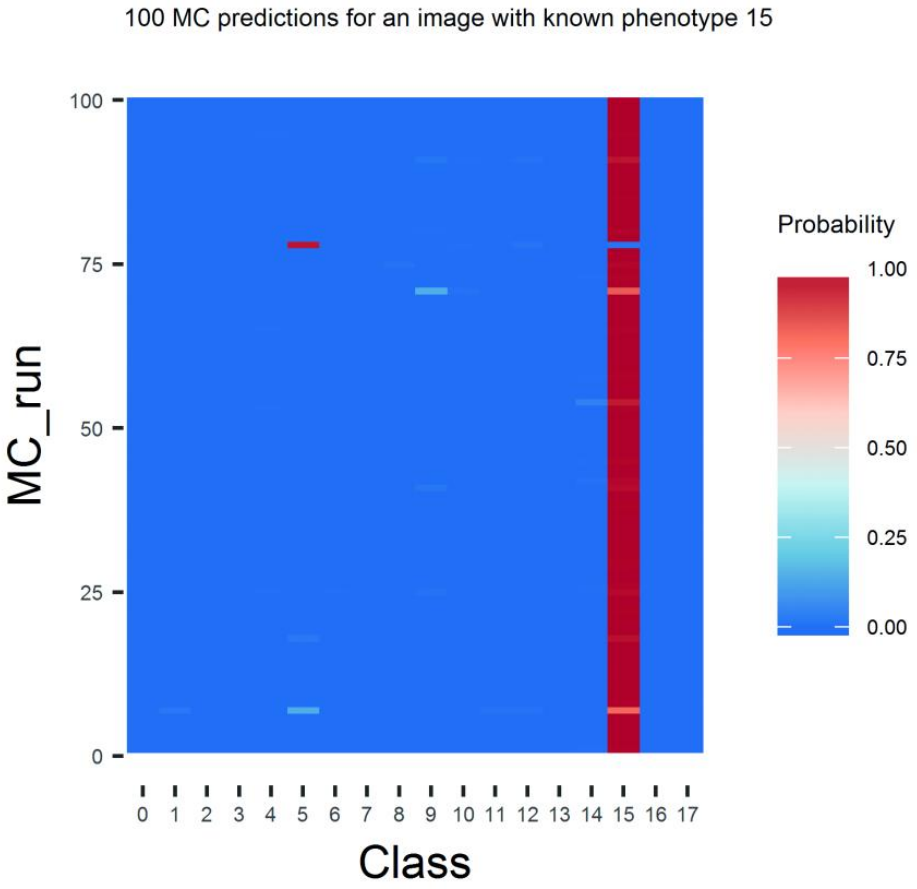
Novelty detection via epistemic uncertainty

Experiment with novel classes in test set

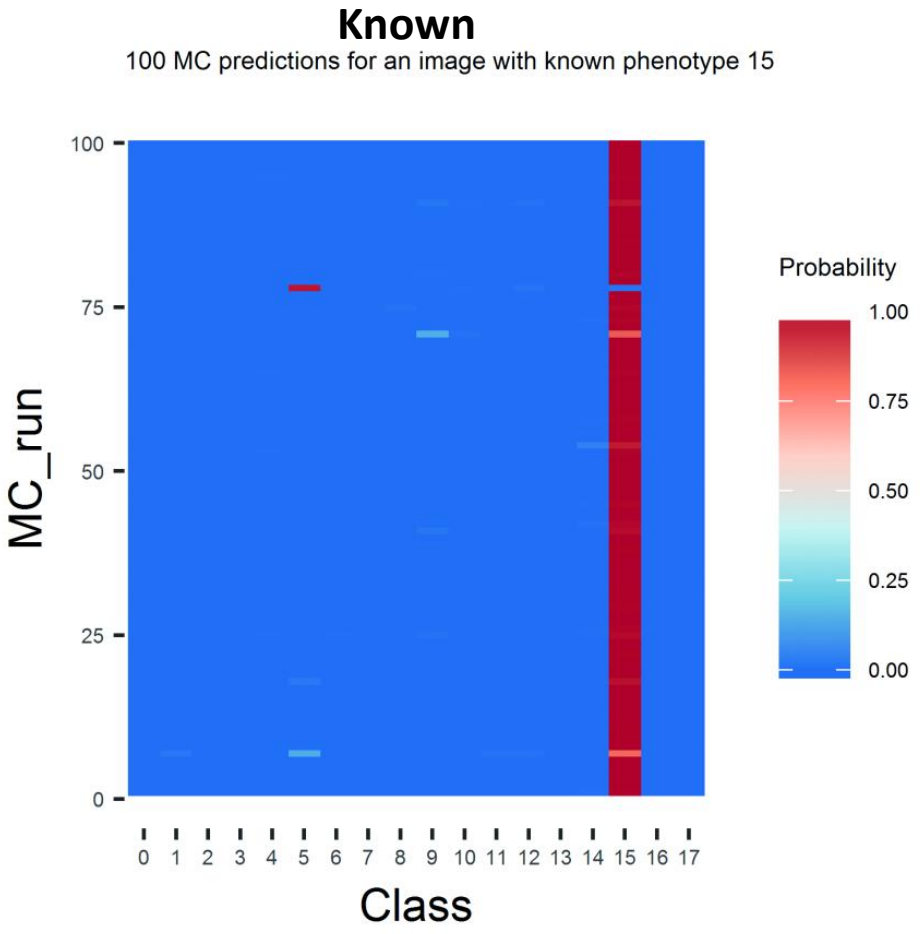


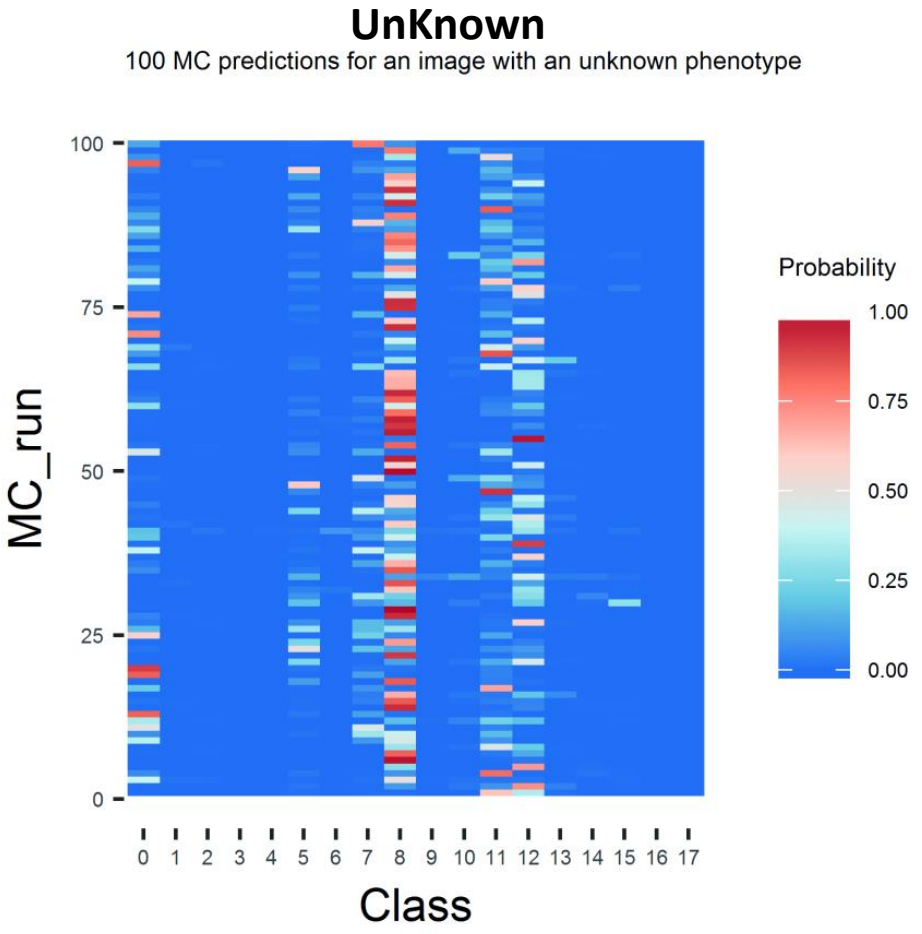
Dürr O, Murina E, Siegismund D, Tolkachev V, Steigele S, Sick B. Know when you don't know, Assay Drug Dev Technol. 2018

Classifying images of known and unknown phenotype



Classifying images of known and unknown phenotype





How to quantify the spread of a MC* probability distribution

The center of mass quatifies the predicted probablilty

p^*_{max}

The spread quantifies in addition the uncertainty of the predicted probability

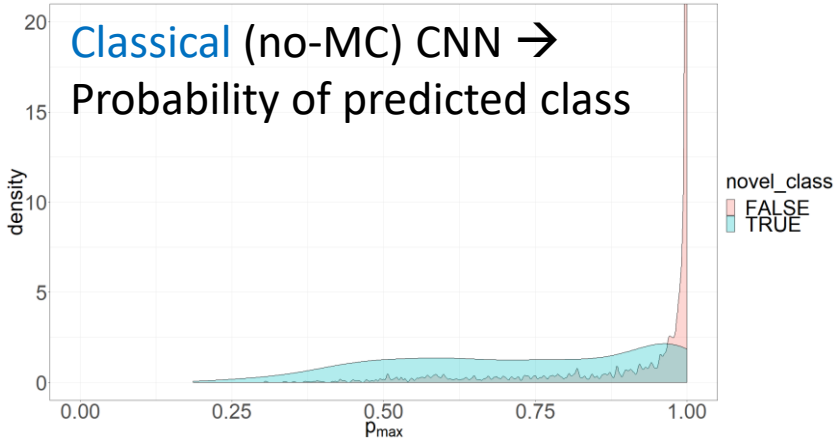
σ^* total standard deviation

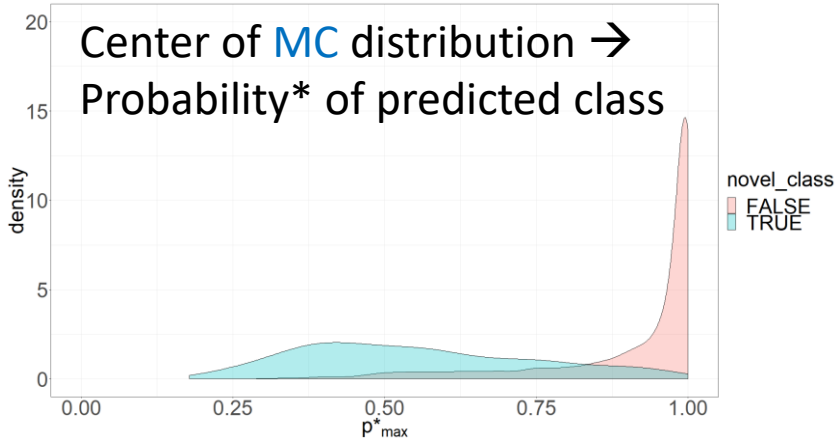
PE^* entropy,

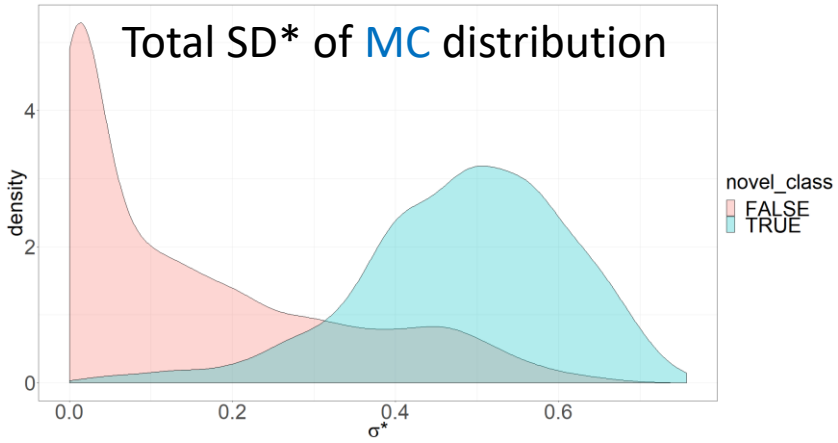
MI^* mutual information

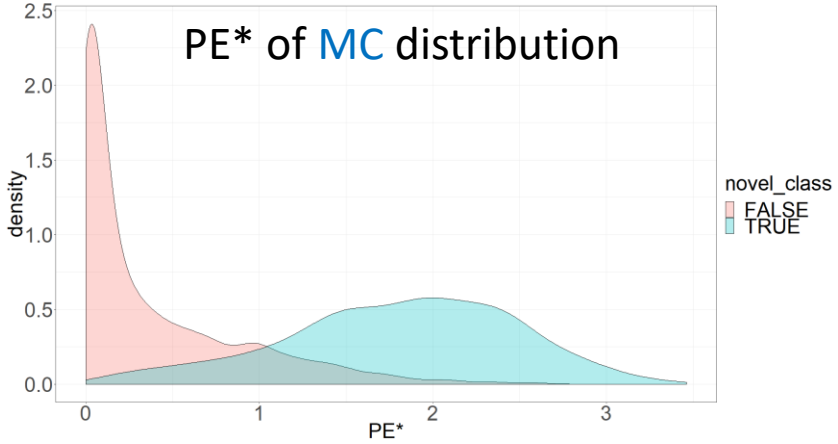
VR* variation ratio

Experiments with images of known and unknown phenotype

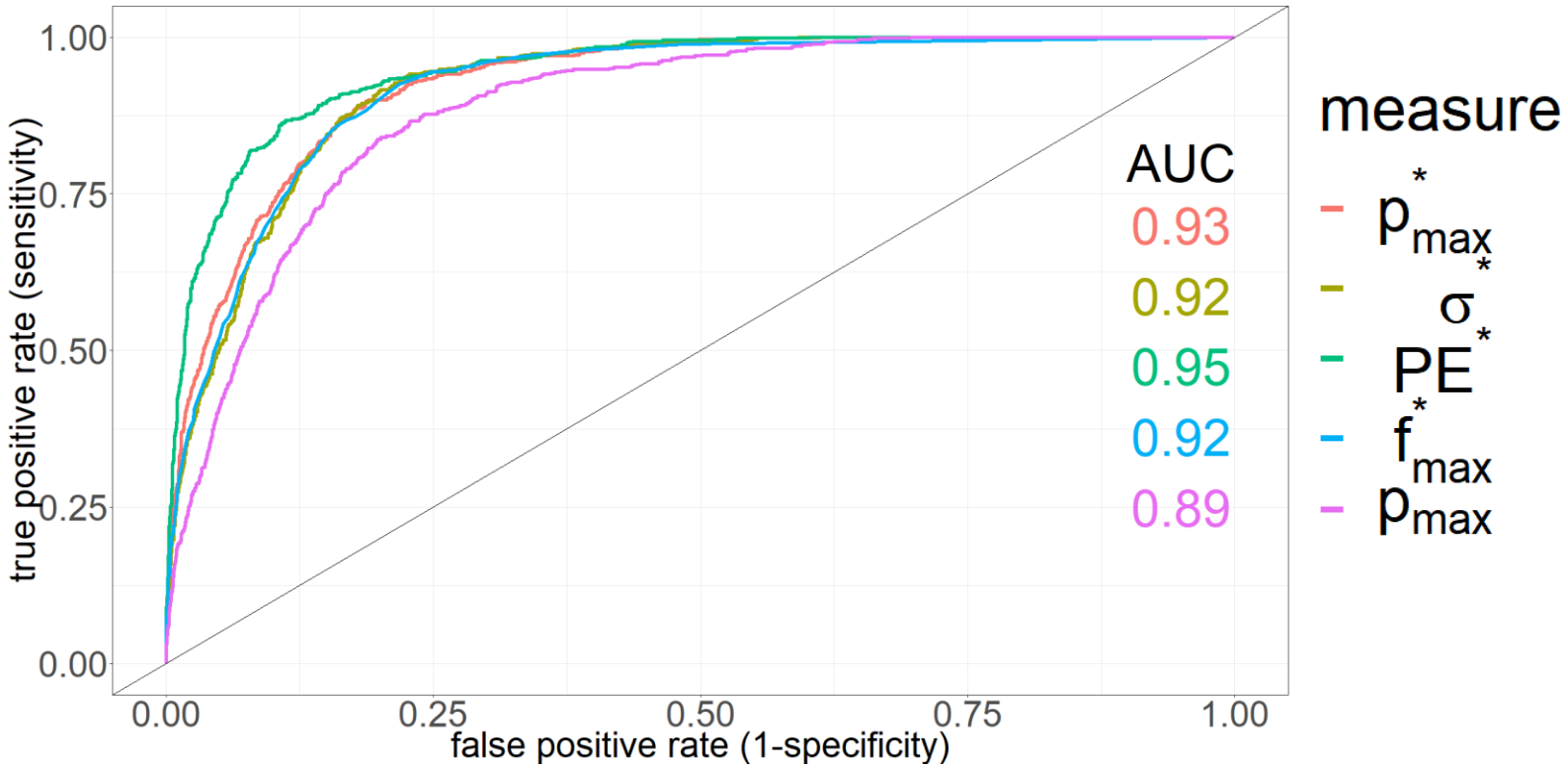






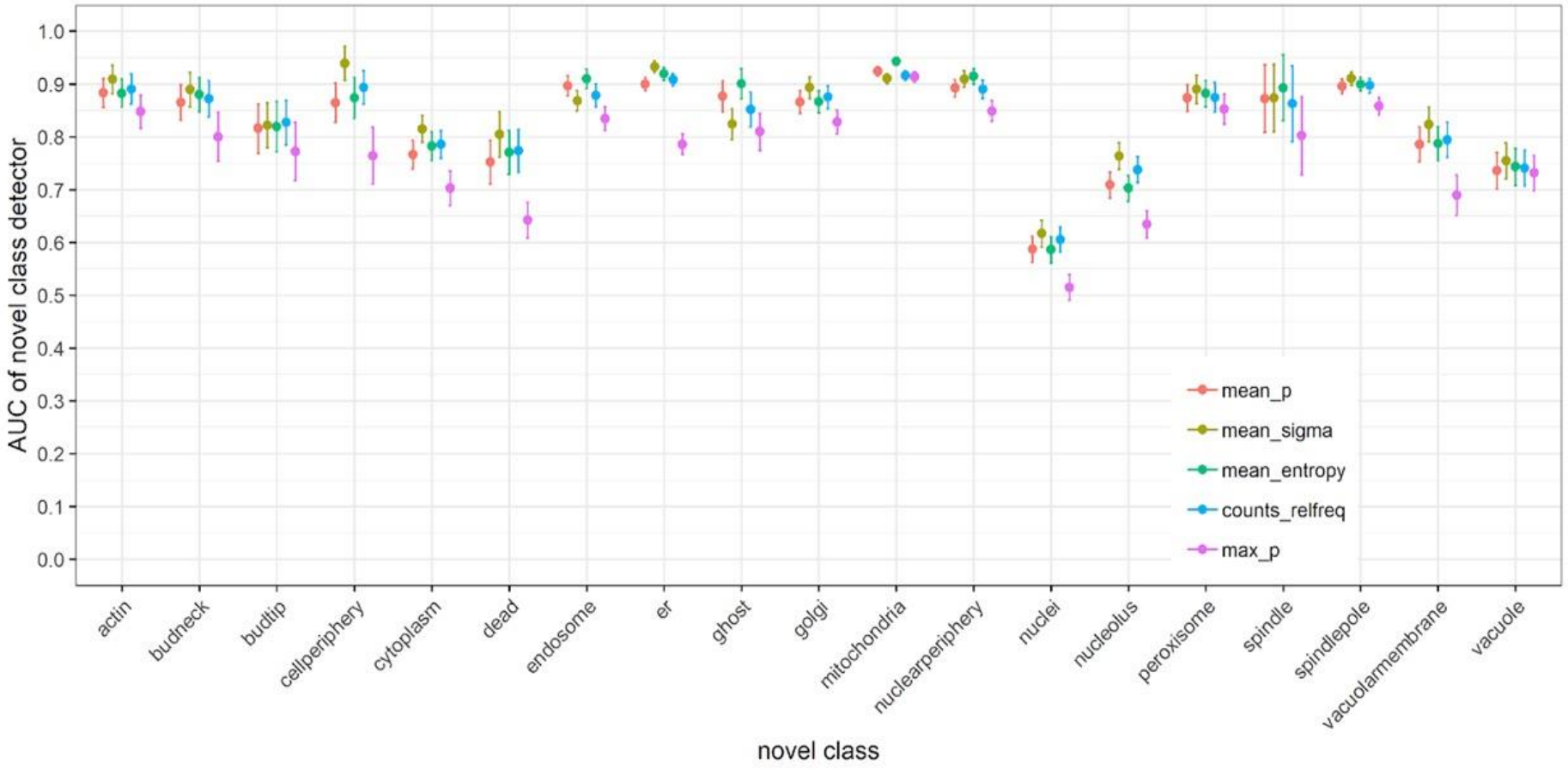


Experiments with images of known and unknown phenotype



All **MC Dropout based appoaches are superior compared** to the non-MC approach.

Experiments with images of known and unknown phenotype



In all cases the Bayesian epistemic uncertainty measures are better suitable to identify novel classes than tradition point estimates of the class probability.

Summary

→ A probabilistic NN fits a whole conditional outcome distribution (CPD)

→ We need to choose a parametric model for this CPD (e.g. a Gaussian for regression)

→ We use the ML or Bayes to estimate the parameters of the CPD

→ We use Bayes methods to estimate the uncertainty of the parameter values

→ Aleatoric uncertainty is the data-inherent variability captured by the CPD

→ Epistemic uncertainty is the uncertainty about the parameter values

→ Epistemic uncertainty can be used for novelty detection