Machine Intelligence:: Deep Learning
Week 8

Oliver Dürr

Institut für Datenanalyse und Prozessdesign

Zürcher Hochschule für Angewandte Wissenschaften

Winterthur, 10. April 2018

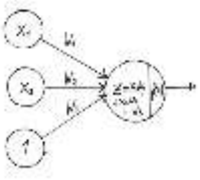
Organizational Issues: Projects

- 2 h lectures
- 2 h posters and presentations

Maximum Likelihood Principle

This is the prob. the model evaluates for the true class $y^{(i)}$ of training example $x^{(i)}$

Training Examples Y=1 or Y=0



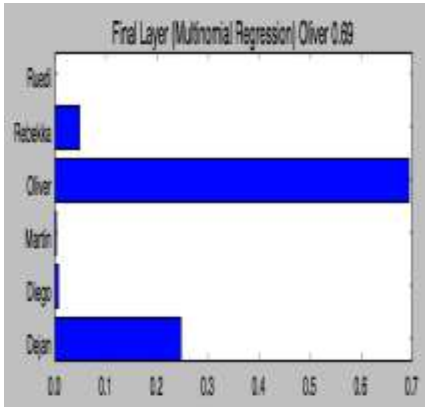
$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\log(p_{\text{model}}(y^{(i)}\,|\,x^{(i)};q))$$

Minimize Loss (Maximum Likelihood Principle)

N Training Examples classes (1,2,3,…,K)

$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\log(p_{\text{model}}(y^{(i)}\,|\,x^{(i)};q)) = -\frac{1}{N}\left(\sum_{i\in y_j=1}\log(p_1(x^{(i)})) + \sum_{i\in y_j=2}\log(p_2(x^{(i)})) + \ldots + \sum_{i\in y_j=K}\log(p_K(x^{(i)}))\right)$$

p_i



Output of last layer

Example: Look at class of single training example. Say it’s Dejan, if classified correctly p_dejan = 1 ➔ Loss = 0. Real bad classifier put’s p_dejan=0 ➔ Loss = Inf.

Alternative solution

```
w = tf.Variable(tf.random_normal([784, 10], stddev=0.01))
b = tf.Variable(tf.zeros([10]))
z = tf.matmul(x,w)+b #aka logits
loss = tf.reduce_mean(
        tf.nn.softmax_cross_entropy_with_logits(labels=y_true,logits=z)
)

#Old Solution
prob = tf.nn.softmax(z)
loss_old = tf.reduce_mean(-tf.reduce_sum(y_true * tf.log(prob),
reduction_indices=[1]))
```

For numerical stability, one should use

```
        tf.nn.softmax_cross_entropy_with_logits
```

There is also a sparse version (no one hot encoded needed)

```
        tf.nn.sparse_softmax_cross_entropy_with_logits
```

Why the hack they call it cross entropy?

Entropy and Cross Entropy

- The central loss function for classification is called cross entropy, why?

- This is a different viewpoint to the max-likelihood approach.

- Let’s start by defining the (information) entropy
  - It’s somewhat like the amount of surprise you get from a sample.
  - How many questions needs to be transmitted / how many questions need to be asked
    - Let’s first do an simple example of a coding book

Image: Wikipedia

Information Content of a single outcome

4 Balls each with same probability 25%

You want to tell your friend which ball has been picked. Or how many question does Partner need to ask.

Coding Scheme

Is the ball blue?

If yes finish?

If not

Is the ball red?

If yes finish?

If not continue

Number of Questions on average?

Can you do better?

Example form https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

Information Content of a single outcome

4 Balls each with same probability 25%

You want to tell your friend which ball has been picked?

Hier weiss man schon nach 3 das 4 übrig bleibt ➔ 2.25

Coding Scheme

Is the ball blue?

If yes finish?

If not

Is the ball red?

If yes finish?

If not continue

Number of Questions on average?

¼*1+1/4*2+1/4*3+¼*4 = 2.5

Can you do a better coding?

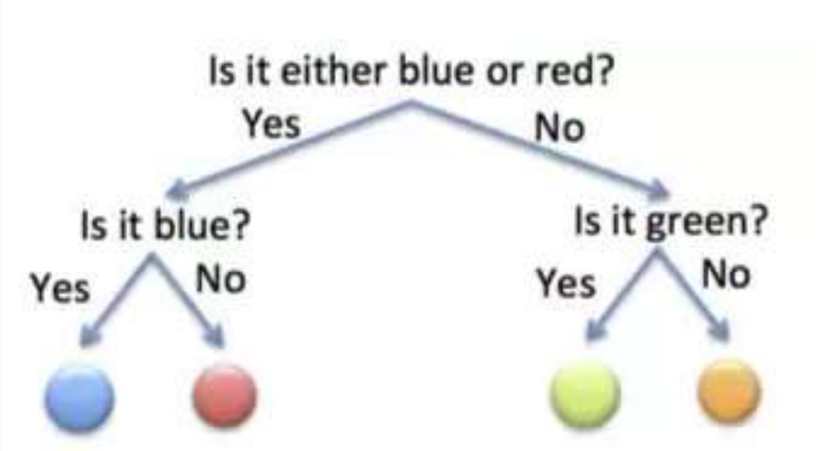
Example form https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

Information Content of a single outcome

Solution

4 Balls each with same probability 25%

How can your friend ask you which ball you picked, with minimum number of questions (on average)?



Let's say we have a red ball.
Two questions need to be ask.

Coding for red ball (yes=1)
10 // Information content 2 bits

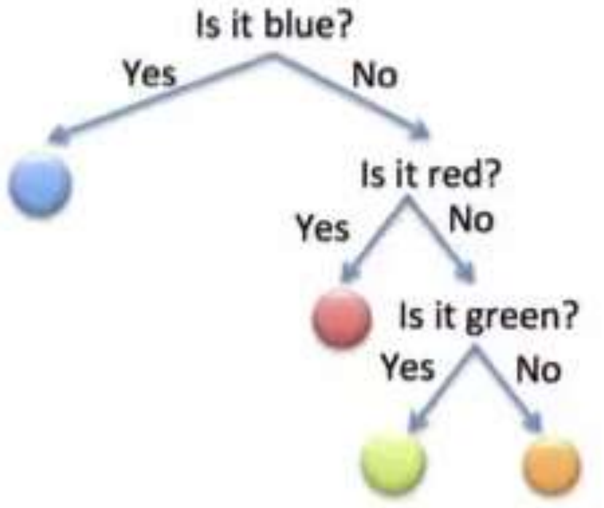
Coding for orange (your turn)
00 // Information content 2 bits

Example form https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

Information Content of a single outcome

- 4 Balls each with different probability 50%, 25%, 12.5%, 12.5%

- How can your friend ask you which ball you picked, with minimum number of questions (on average)?



Let's say we have a blue ball.
One questions need to be ask.

Coding for blue ball (yes=1)
1 // Information content 1 bit

Coding for red (2 questions)
01 // Information content 2 bit

Coding for green (your turn)
001 // Information content 3 bit

Example form https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

Information Content

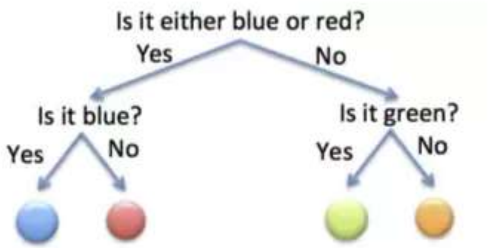
- For that easy example, we found the best coding by hand.
- Let's define the (self-) information (Turns out to be the minimal coding length "Shannon's source coding theorem")

- Requirement for Information (or surprise)
  - p_i the probability of event i (or prob. that symbol i occurs)
  - Seldom examples should have more surprise.
    - $I(p_i)$ should be monotonic decreasing (large p, no surprise, low I) function
  - Information should be non-negative
    - $I(p_i) \geq 0$
  - Uninformative, or sure events should have no Information
    - $I(p_i = 1) = 0$
  - Information of independent events i, j should add up
    - $I\left(p_{(i,j)}\right) = I\left(p_i p_j\right) = I(p_i) + I(p_j)$
- ➔ $\boldsymbol{I(p) = -\log_2(p)}$

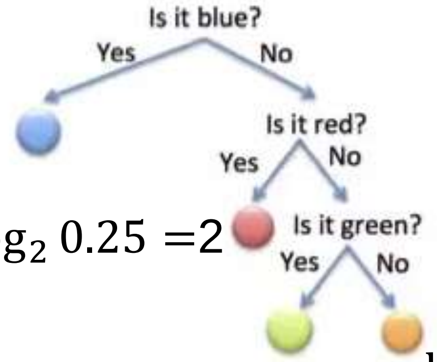
Defined up to basis, 2 is often chosen. Then "bits" if e than "nats"

Information Content → Entropy

- Entropy (average Information Content)
  - $H(p) = \sum p_i I(p_i) = -\sum p_i \log_2(p_i)$



$-\log_2 0.25 = 2$

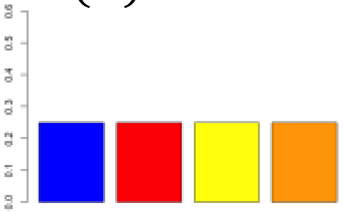


$-\log_2 0.5 = 1$

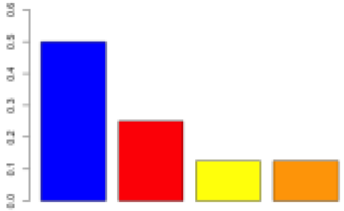
$-\log_2 0.25 = 2$

$-\log_2 0.125 = 3$

$H(X) = 2$



$H(X) = 0.5 + 0.25*2 + 0.25*3 = 1.75$



In general: Maximal Entropy if uniform, minimal if peaked (see also in physical Systems)

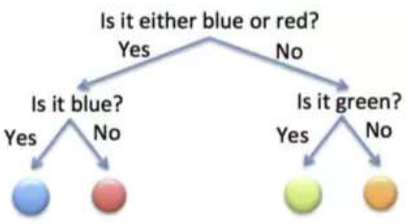
Cross Entropy

- If we know the distribution p, we can find the best coding and need H bits on average

- If we have **a “wrong” distribution q** how many bits do we need on average

$$H(p,q) = \sum p_i I(q_i) = -\sum p_i \log_2(q_i) \geq H(p)$$

- Example, we think symbols come uniform distributed q. But they come (0.5,0.25,0.125,0.125)



Optimal Coding Scheme
for Uniform q

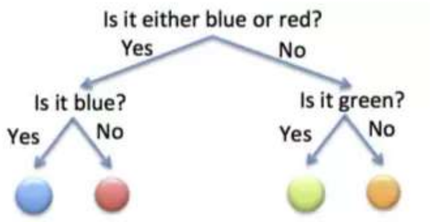
$$H(p,q) = 0.5\ 2 + 0.25\ 2 + 0.125\ \ast 2 + 0.125\ 2 = 2 > 1.75$$

KL-Divergence

- If we have a “wrong” distribution q how many bits do we have more than the minimal possible amount $H(p)$

$$-D_{KL}(p||q) = H(p,q) - H(p) = \sum p_i \ln(q_i/p_i) \geq 0$$

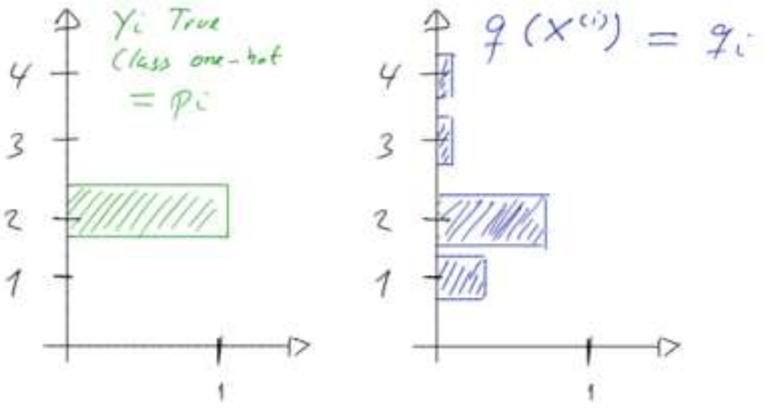
- Example, we think symbols come uniform distributed q. But they come (0.5,0.25,0.125,0.125)



Optimal Coding Scheme for Uniform q

$$D(p,q) = H(p,q) - H(p) = 2 - 1.75 = 0.25$$

Cross Entropy in DL



$H(p,q) = -\sum p_i lnq_i$ (for one example of the training set)

$H(p,q) = -\sum\sum p_i^{(j)} lnq_i^j$ (for the training set)

We minimize the cross entropy by changing q, the minimum is reached when q is identical to distribution of real labels p

Alternatively we could also minimize the KL-Divergence

Further Resources (cross entropy and information theory)

- https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/
- https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy
- https://www.khanacademy.org/computing/computer-science/informationtheory/moderninfotheory/v/information-entropy
- https://medium.com/swlh/shannon-entropy-in-the-context-of-machine-learning-and-ai-24aee2709e32

Summary: Foundations of Loss Fun

- Loss function can be derived from Maximum Likelihood Principle

$$\text{loss} = - \frac{1}{N} \overset{N}{\underset{n=1}{\mathring{a}}} \log(p_{\text{model}}(y^{(i)} \mid x^{(i)}; q))$$

- Loss function as minimal cross-entropy

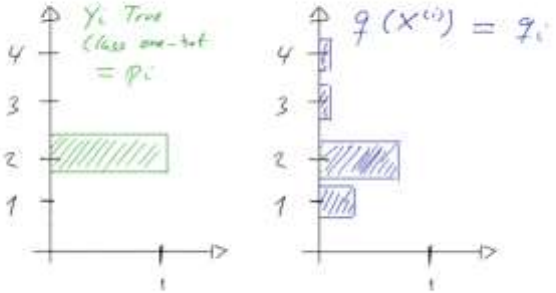
$$H(p,q) = -\sum p_i ln q_i$$

q_i Model Distribution for training data i

p_i True Distribution (“on—hot”)

- Loss function as minimal KL-distance between

$$D_{KL}(p||q)$$

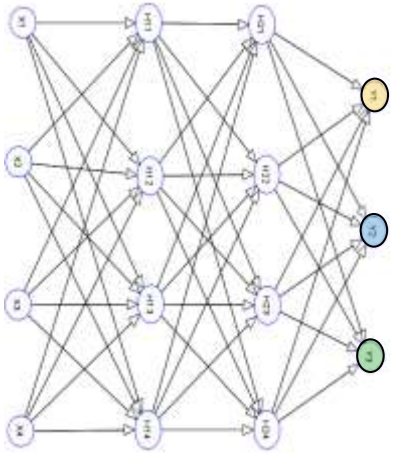


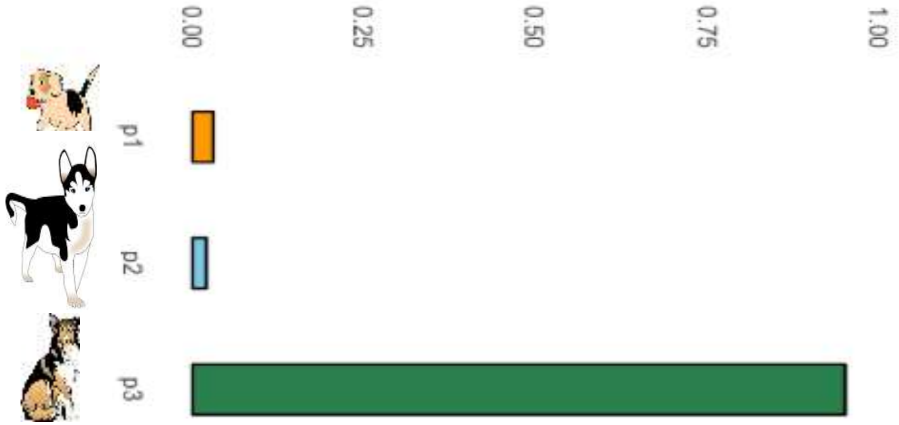
Monte Carlo Dropout

The usual deep learning success story…

A network trained to classify dog breads

Correct



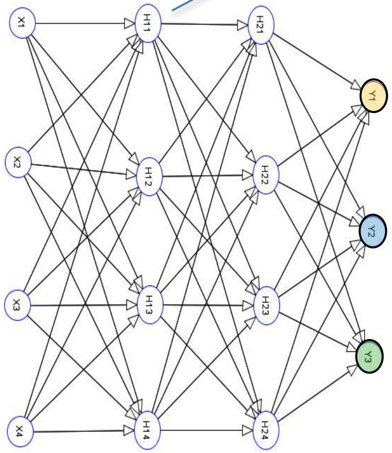


Task: What is this?

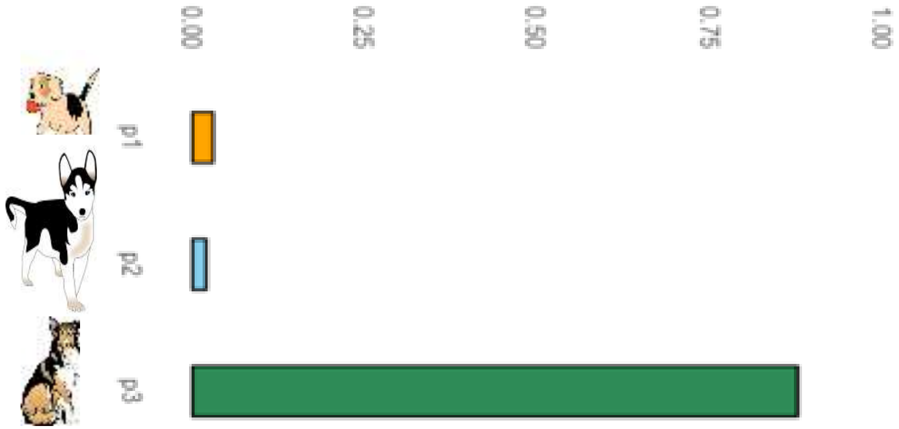
Deep neural networks can't voice their doubts…

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

A network trained to classify dog breads



Plain wrong



We need error bars!

If this doesn't scare you, think about self driving cars in an unknown environment.

A first experiment

- Let's do the experiment (with our data)
  - We remove a Mitochondria phenotype (cat) from the training set
  - Train the classifier w/o Mitochondria
  - Show Mitochondria (from validation set) to the trained classifier
    - It should tell you that it is unsure

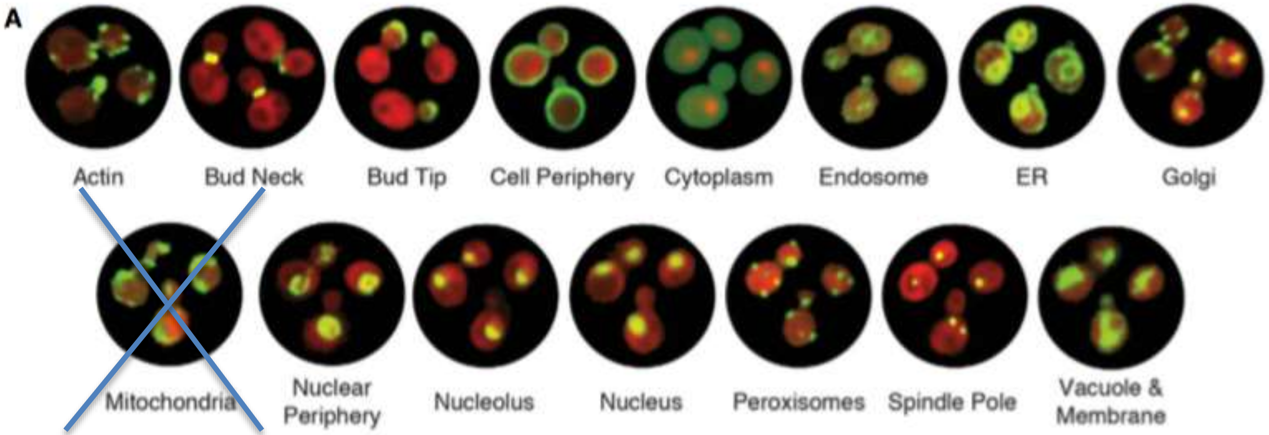
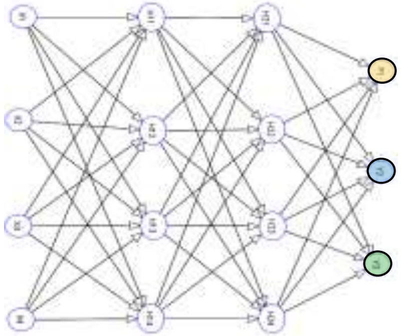
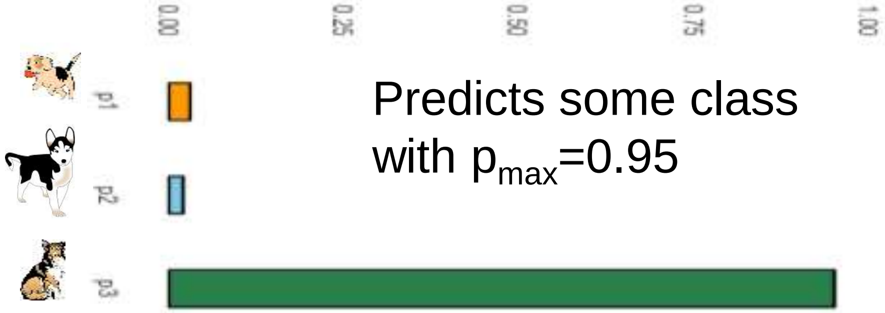


Image credit Kraus, O.Z. et all. , B.J. Molecular Systems Biology 13.4 (2017): 924

Results for the removed class

No mitochondria in training.



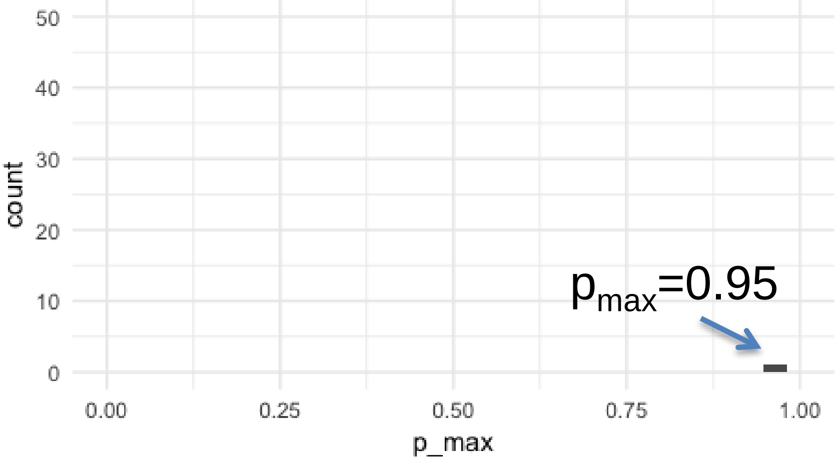


Predicts some class with p_{max}=0.95

Repeat this for all 620 Mito. cells in the validation set

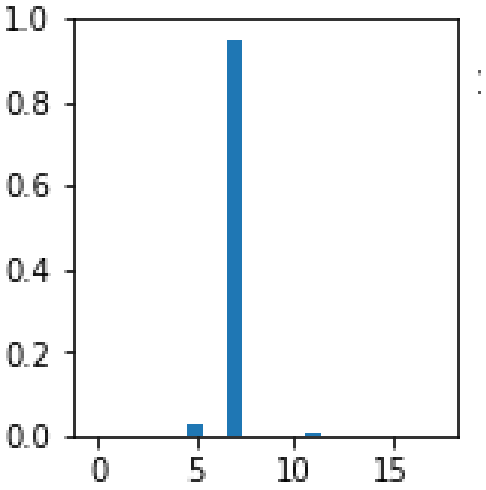
106 of 620 cells are from a class scored with over 90%!

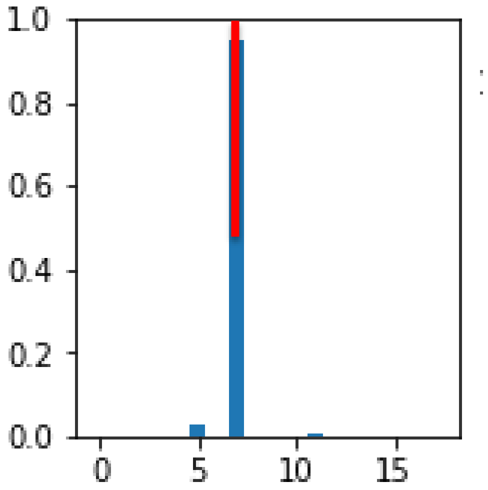
Not enough to have point estimates, we want error bars…

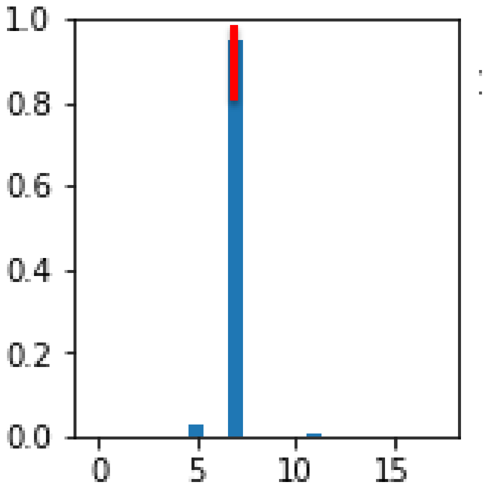


p_{max}=0.95

We want error bars (or even better a distribution)


95% class 6


95% class6 unsure


95% class6 sure

How to get error bars?

What would an experimenter do?

- Go in lab and repeat!

What would a kaggle script kid do?

- Simply spin up 100 AWS instances and repeat (train and predict)

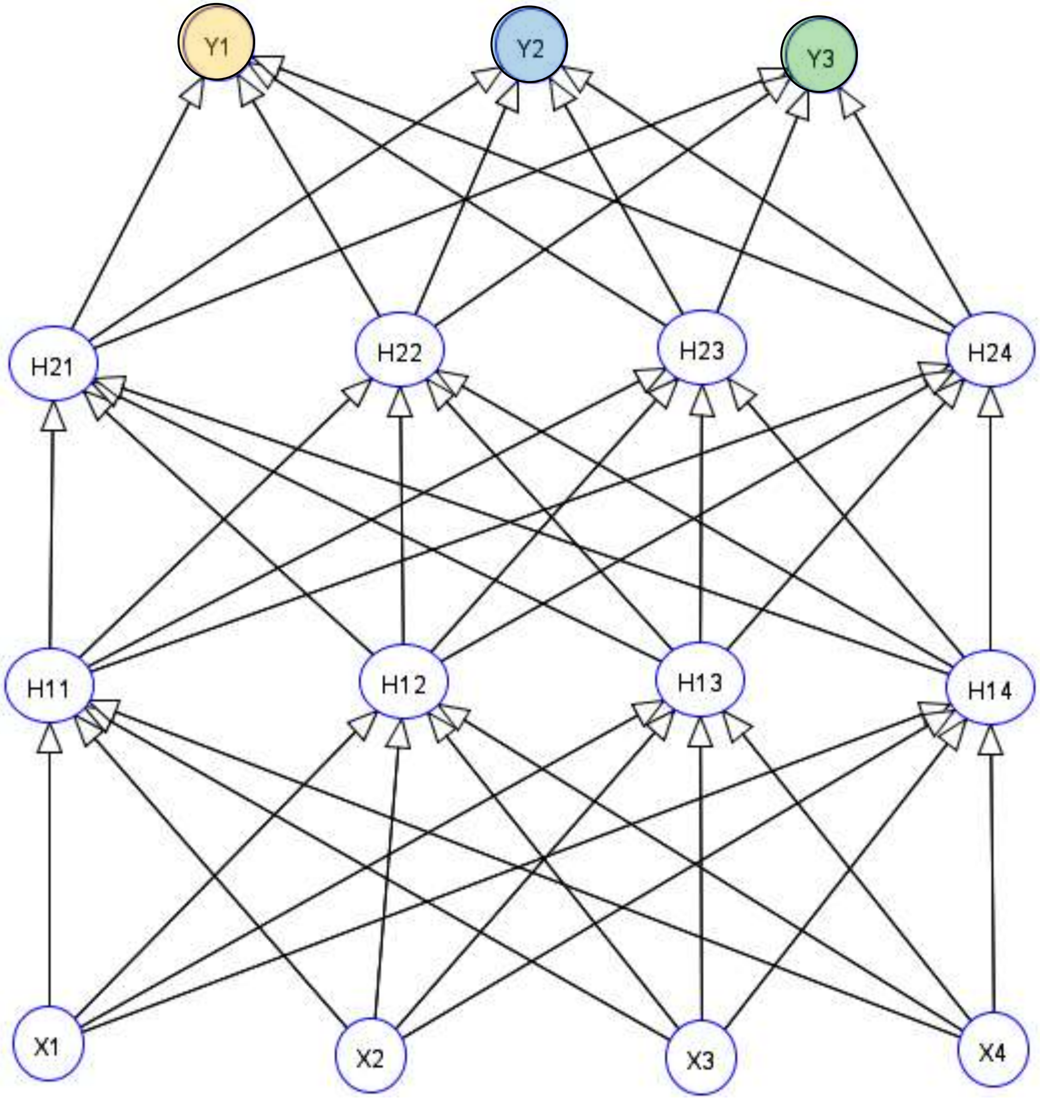
What would a computer scientist / statistician do?

- …

...Remember Dropout?

output: probabilities for each class

p_1 p_2 p_3



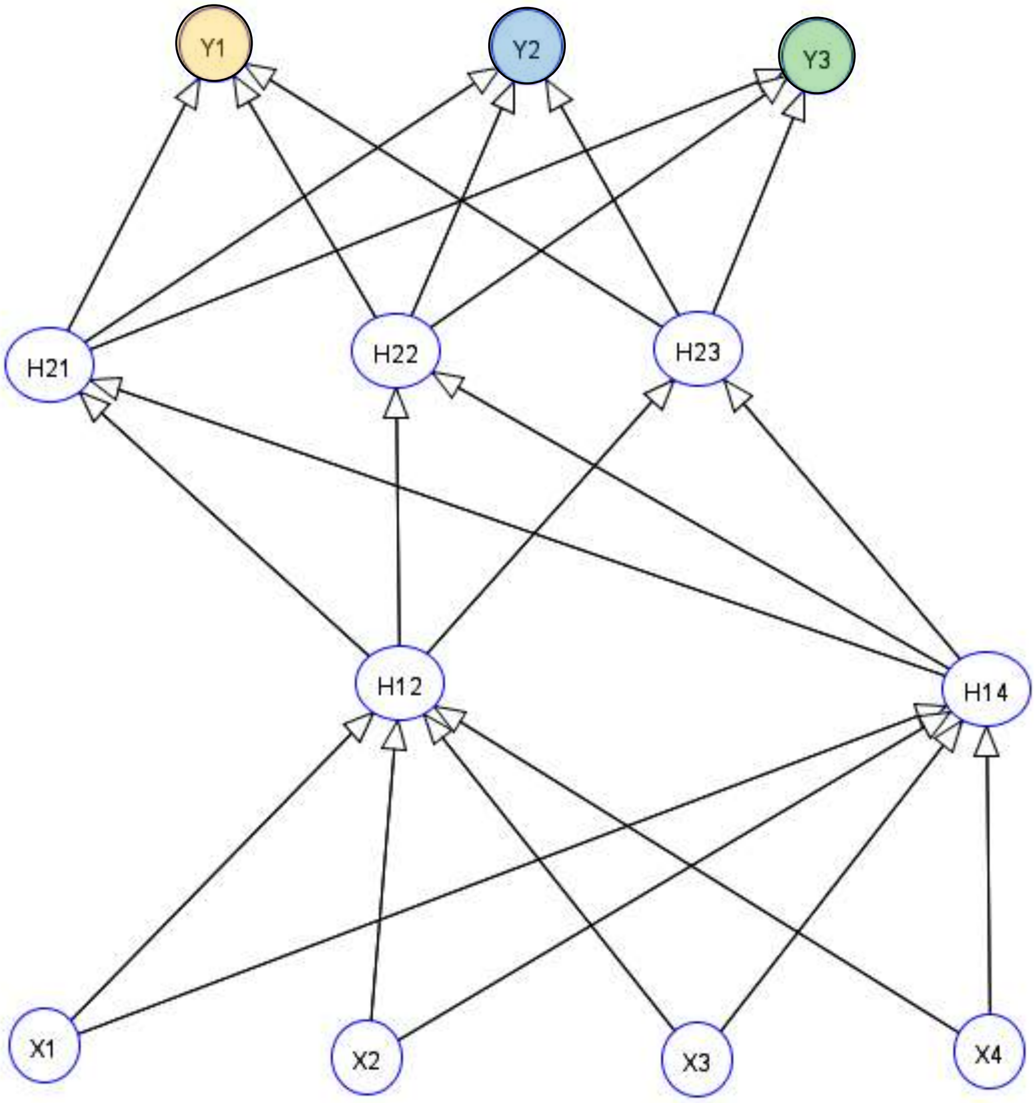
Done in training anyway

input: image pixel values

Use dropout also during testing

RUN 1

p_1=0.08 **p_2=0.89** p_3=0.03



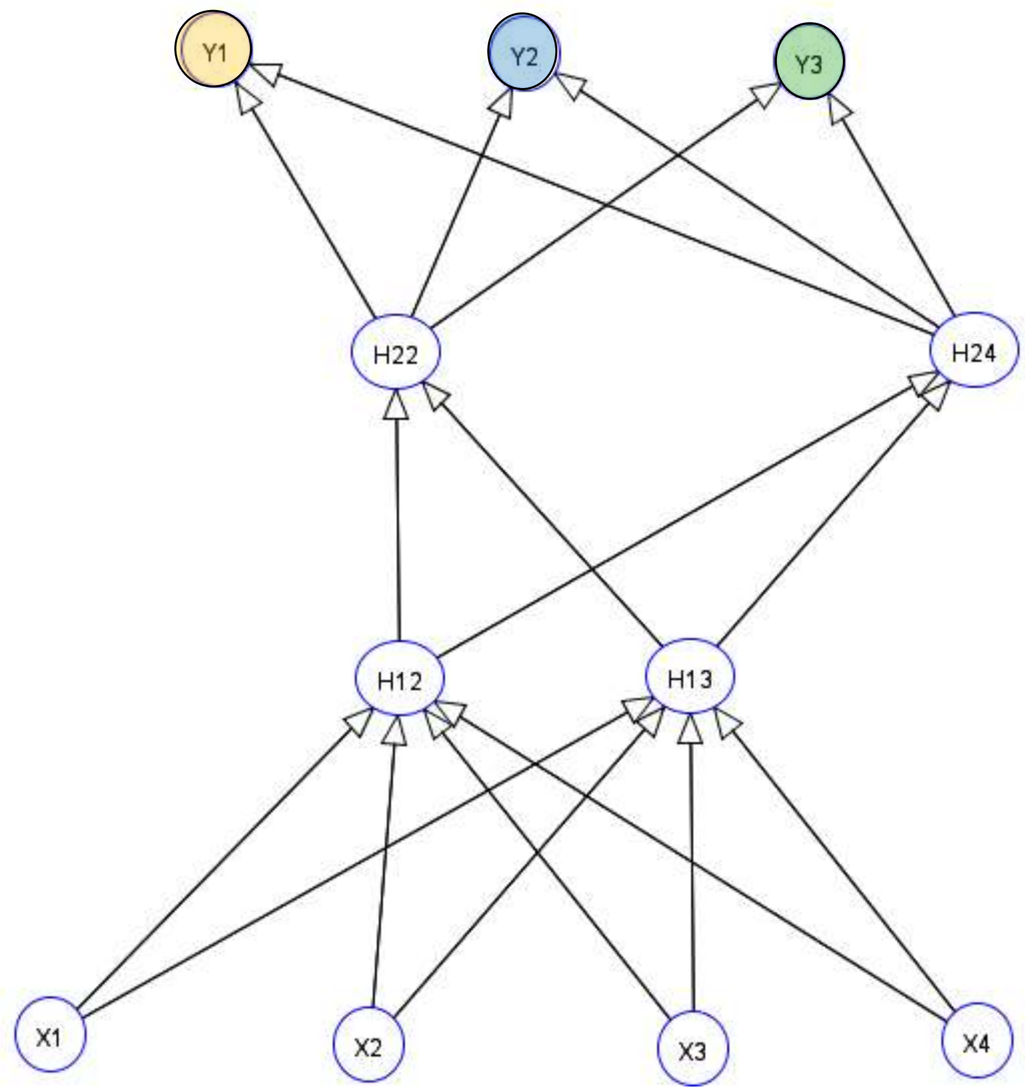
output depends on dropout

stochastic dropout of units

same input

RUN 2

p_1=0.11 **p_2=0.81** p_3=0.08



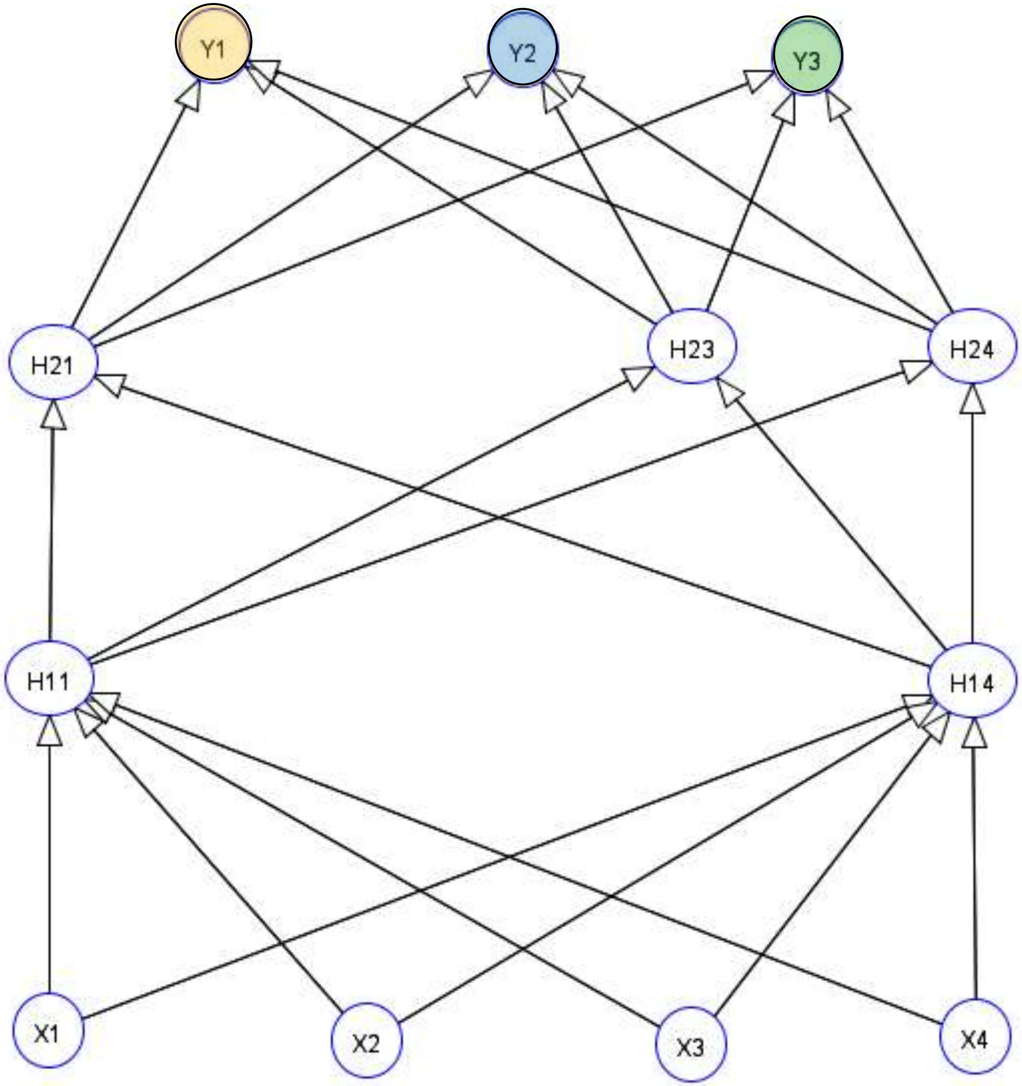
output depends on dropout

stochastic dropout of units

same input

RUN 3

$p_1=0.03$ **$p_2=0.94$** $p_3=0.03$



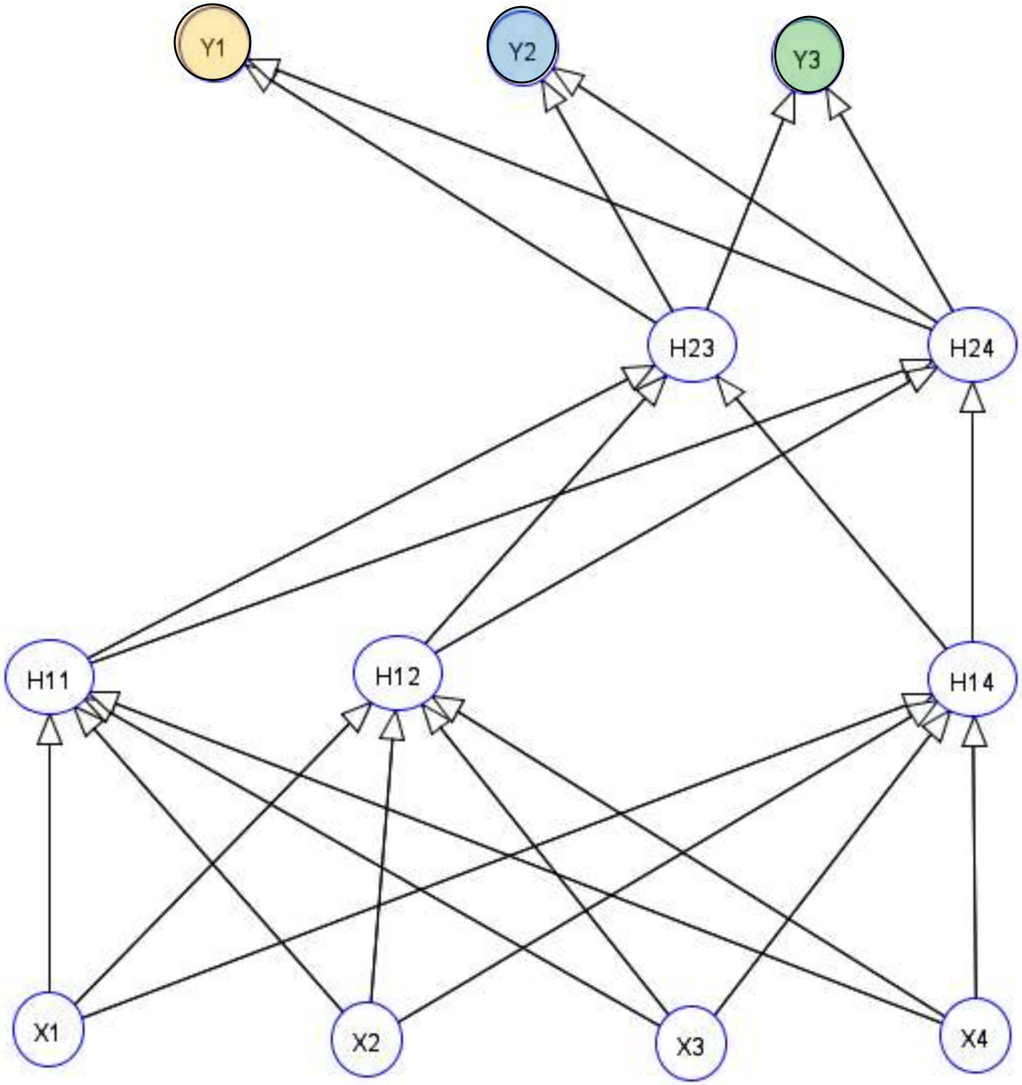
output depends on dropout

stochastic dropout of units

same input

RUN 4

p_1=0.16 **p_2=0.78** p_3=0.06



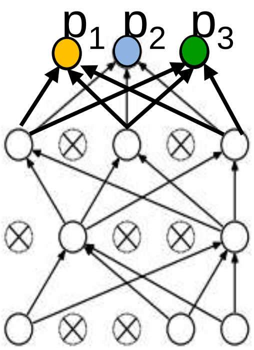
output depends on dropout

stochastic dropout of units

same input

...Repeat 1000 times

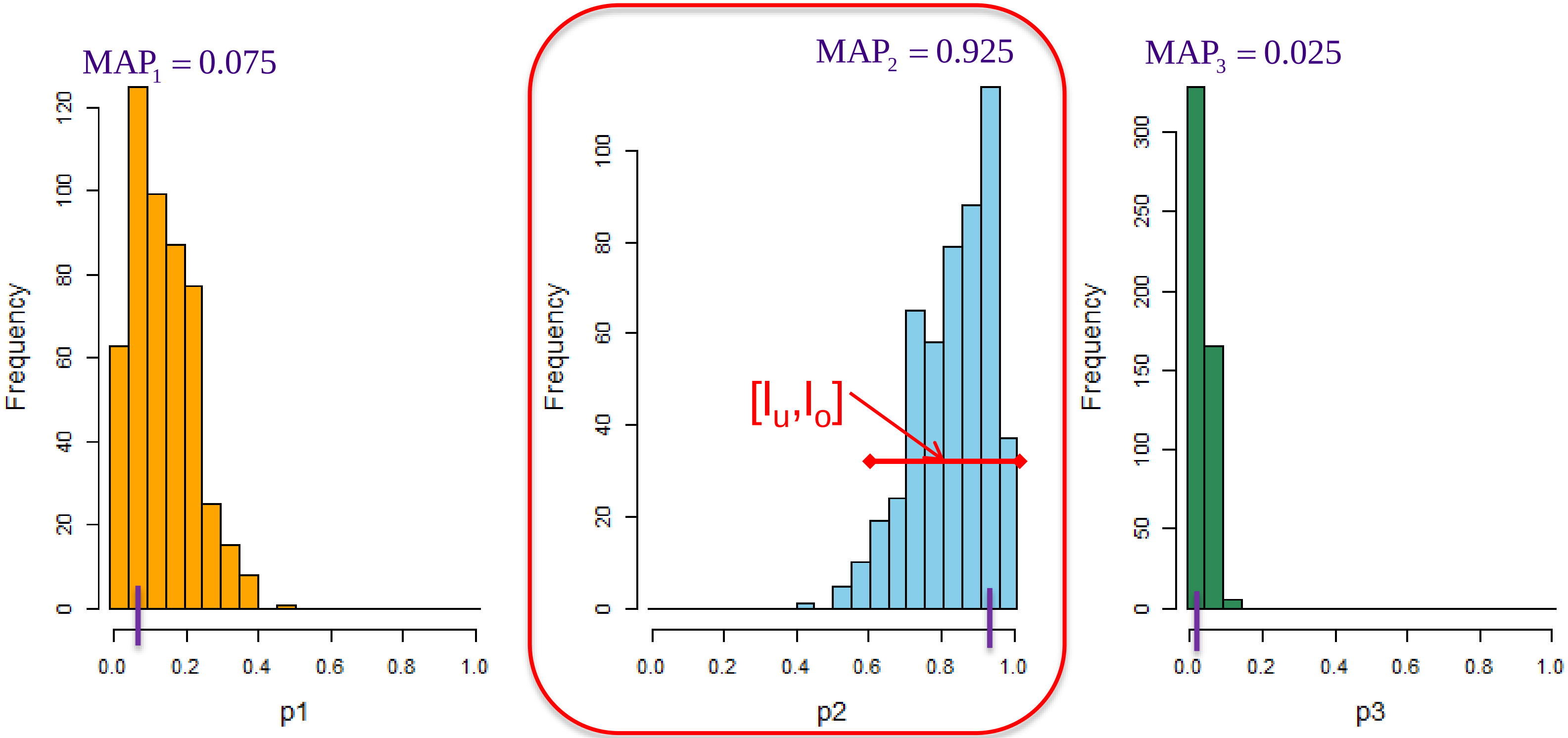
Distributions of predicted probabilities by dropout during test time



one image

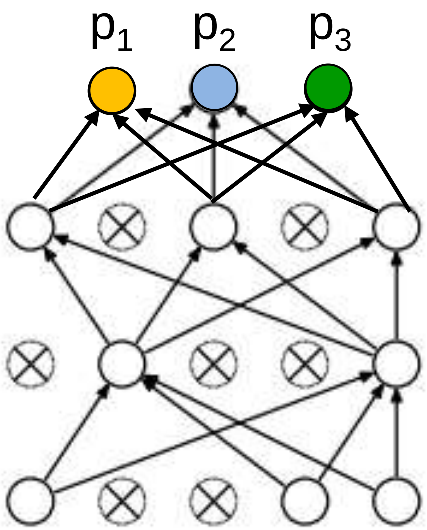
The class with highest probability at modus (MAP) chosen as predicted class.

Use 66% CI $[l_u,l_o]$ around MAP for confidence of the predicted probability.



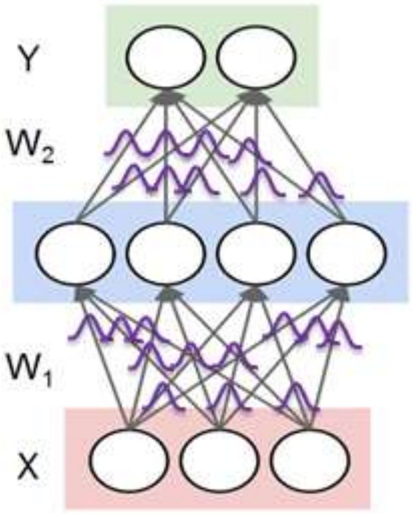
Note that we do not have independence in the p, therefore marginal histograms are strictly not allowed.

Does this really make sense?



Equivalence

Yarin Gal* (2015)



MC-Dropout
At each training and testing step we remove random nodes with a probability p

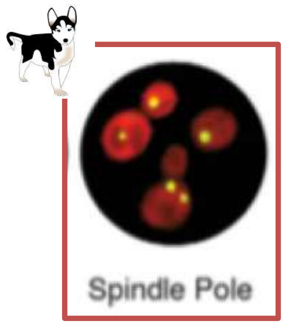
Bayesian Neural Networks

- Provides predictive probability distribution.

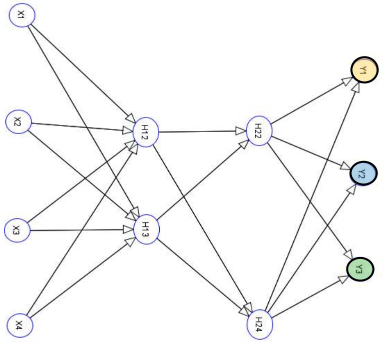
Get new experiments by simply doing dropout, also at testing.

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

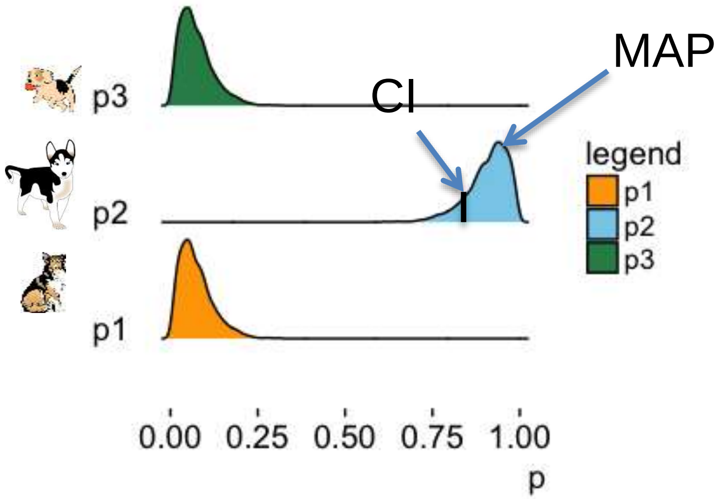
Phenotype in training

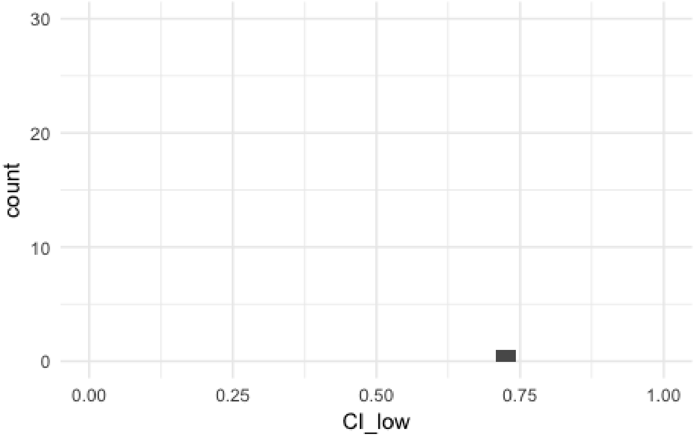


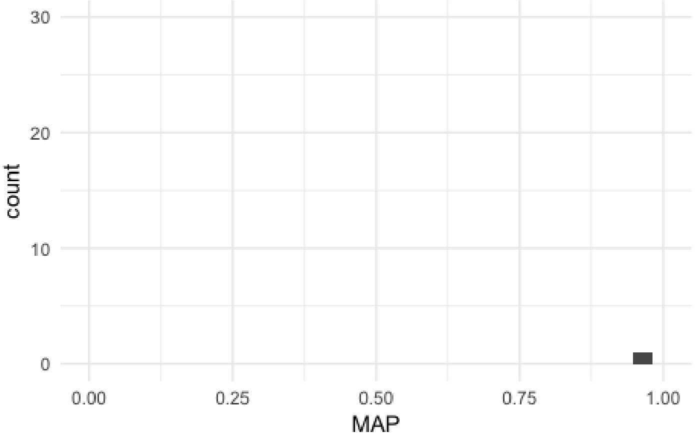
This phenotype is in training!



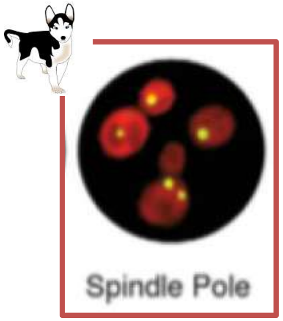
Many Dropout Runs



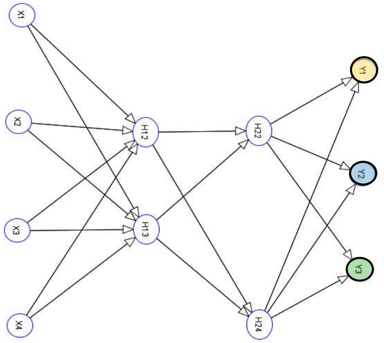




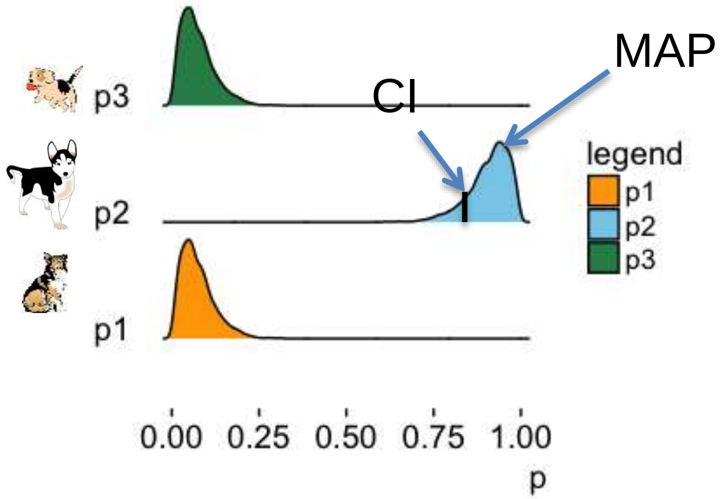
Phenotype in training



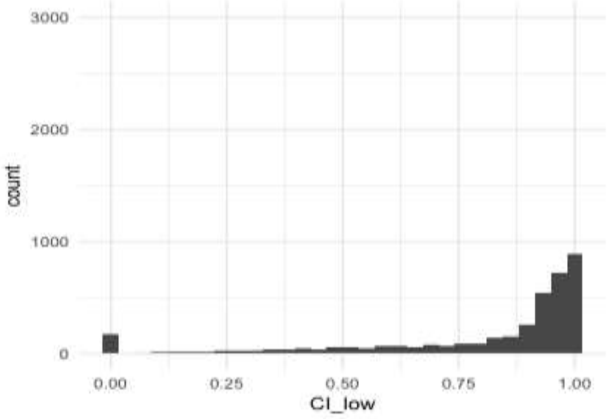
This phenotype is in training!

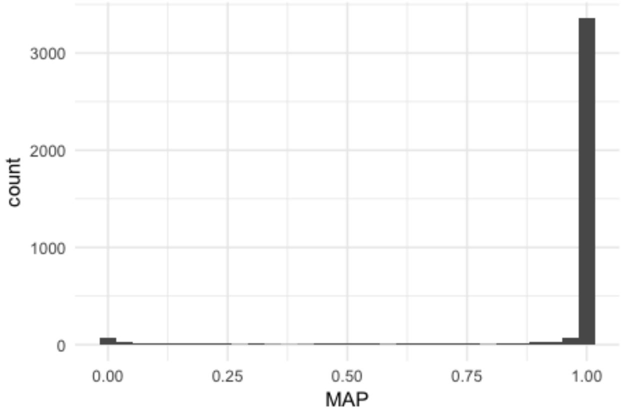


Many Dropout Runs

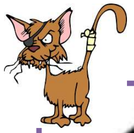


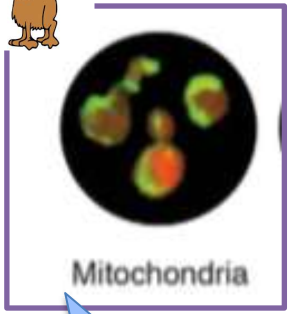
Repeat this for all 3849 non mitochondria cells in the validation set



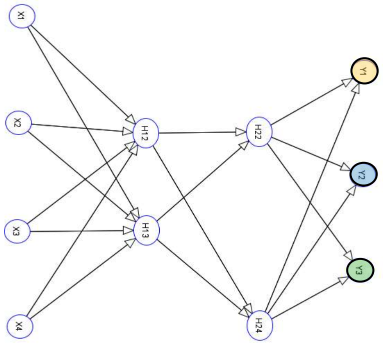


Phenotype not in training

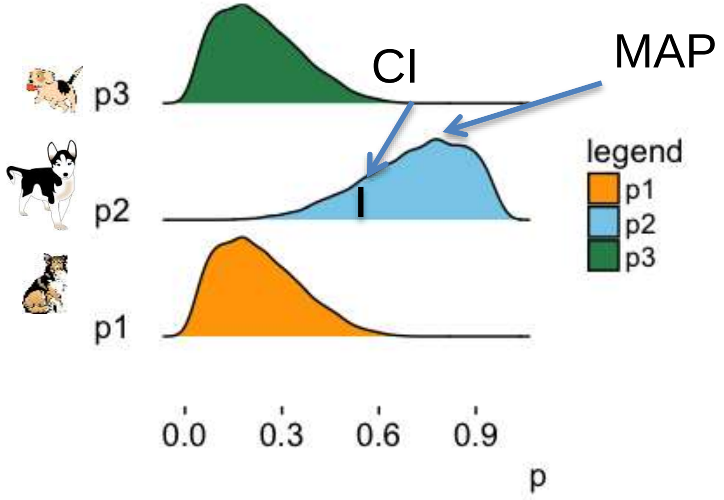




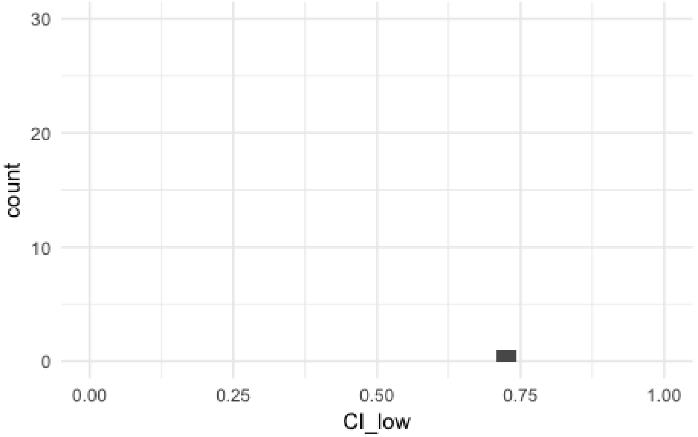
This phenotype is not in training!

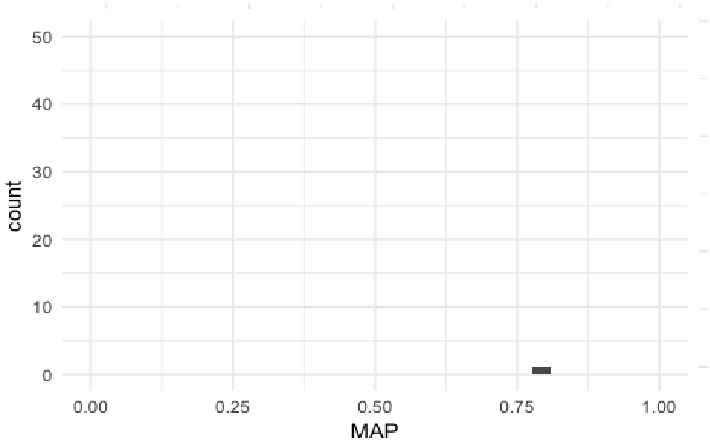


Many Dropout Runs

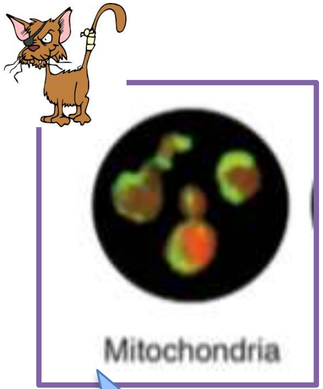


Repeat this for all 620 mitochondria cells in the validation set

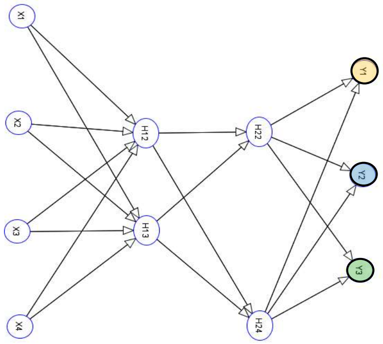




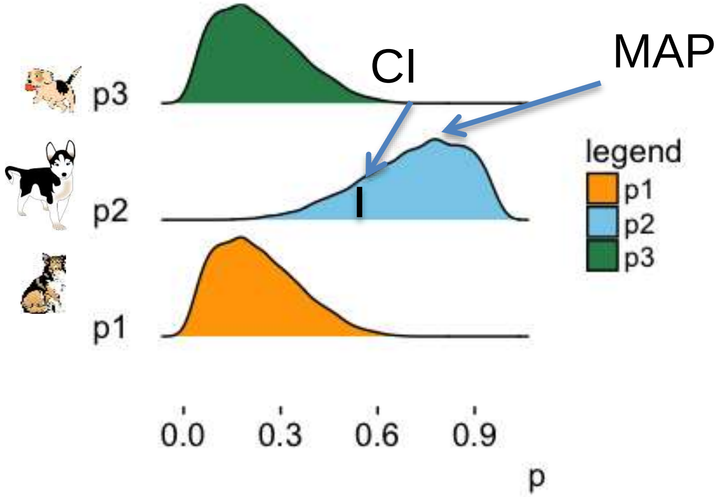
Phenotype not in training



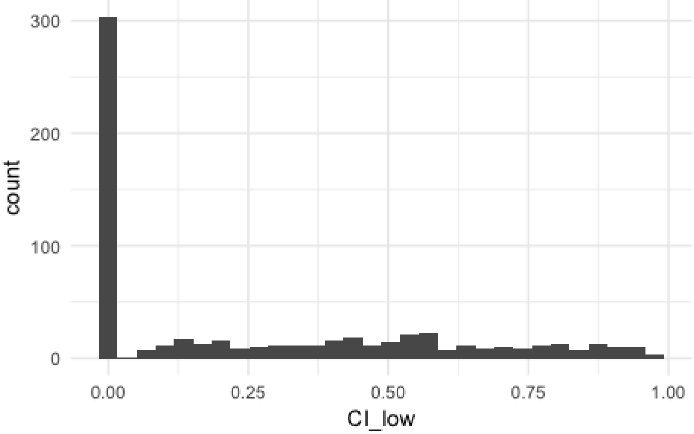
This phenotype is not in training!

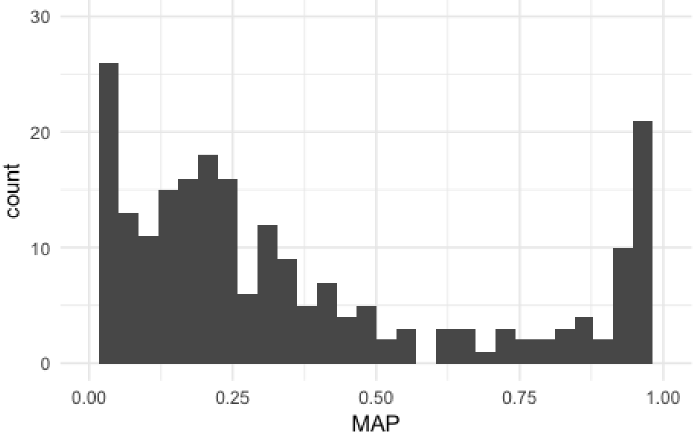


Many Dropout Runs

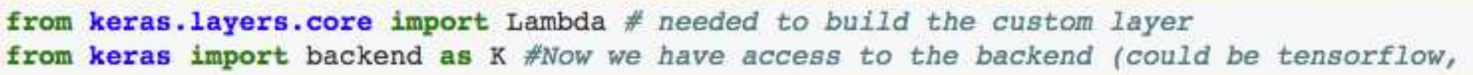


Repeat this for all 620 mitochondria cells in the validation set



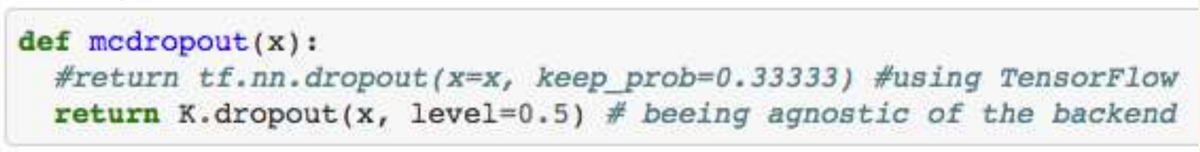


Creating custom layers in keras



```
from keras.layers.core import Lambda # needed to build the custom layer
from keras import backend as K #Now we have access to the backend (could be tensorflow,
```

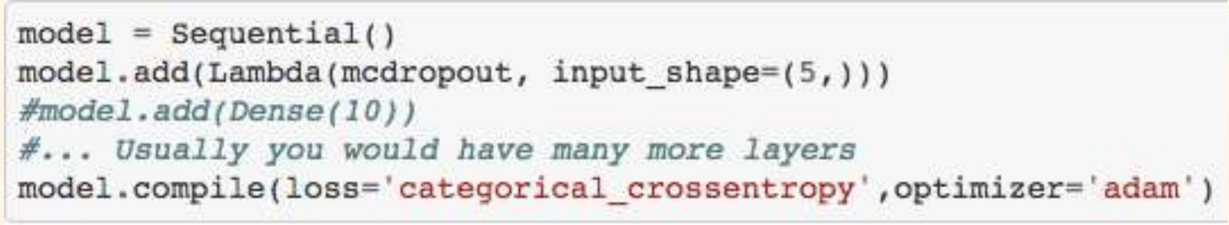
Define your custom function



```
def mcdropout(x):
  #return tf.nn.dropout(x=x, keep_prob=0.33333) #using TensorFlow
  return K.dropout(x, level=0.5) # beeing agnostic of the backend
```

Here you could do anything possible in TF
`tf.add(x, 10)`
…

Include your custom function as a layer



```
model = Sequential()
model.add(Lambda(mcdropout, input_shape=(5,)))
#model.add(Dense(10))
#... Usually you would have many more layers
model.compile(loss='categorical_crossentropy',optimizer='adam')
```

https://github.com/oduerr/dl_tutorial/blob/master/tensorflow/keras/using_tf_in_keras.ipynb