

Neural Networks 2

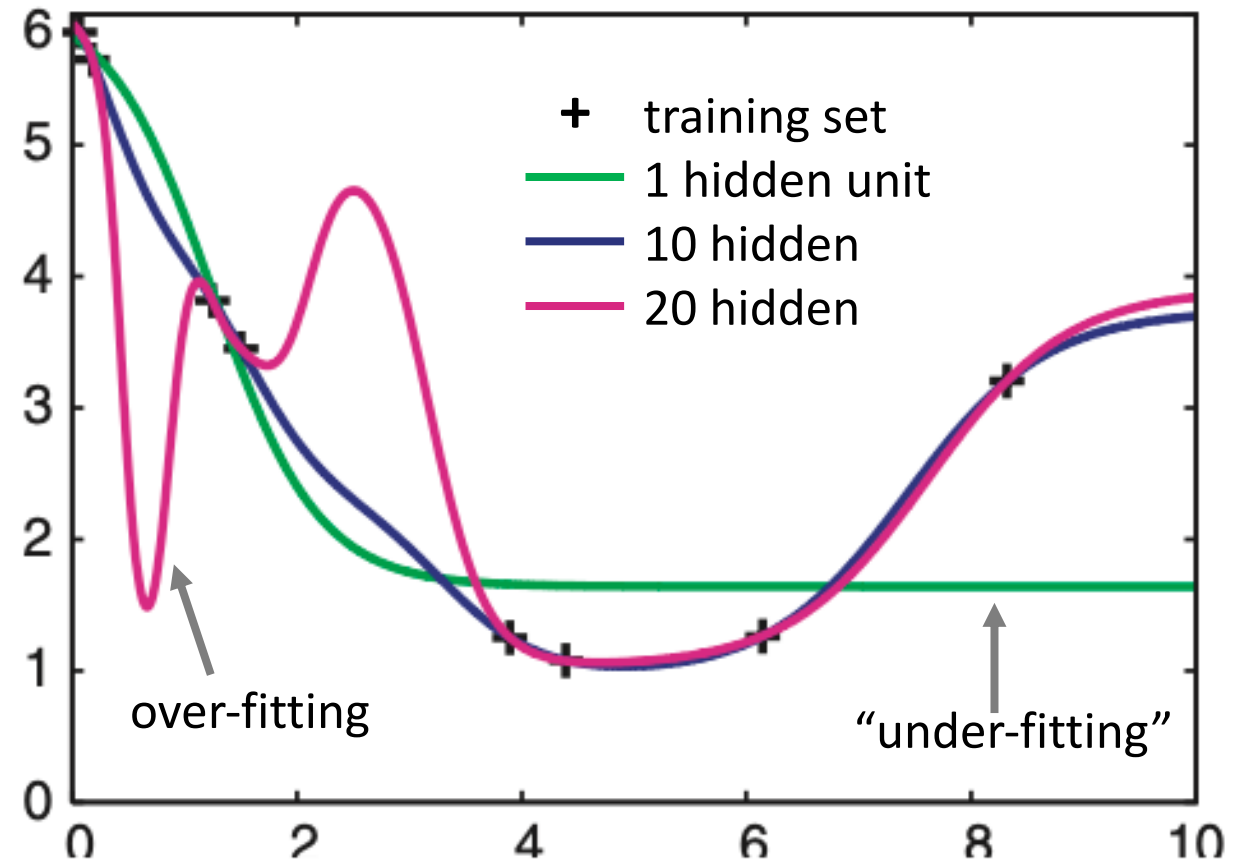
Anders Krogh

Center for Health Data Science

University of Copenhagen

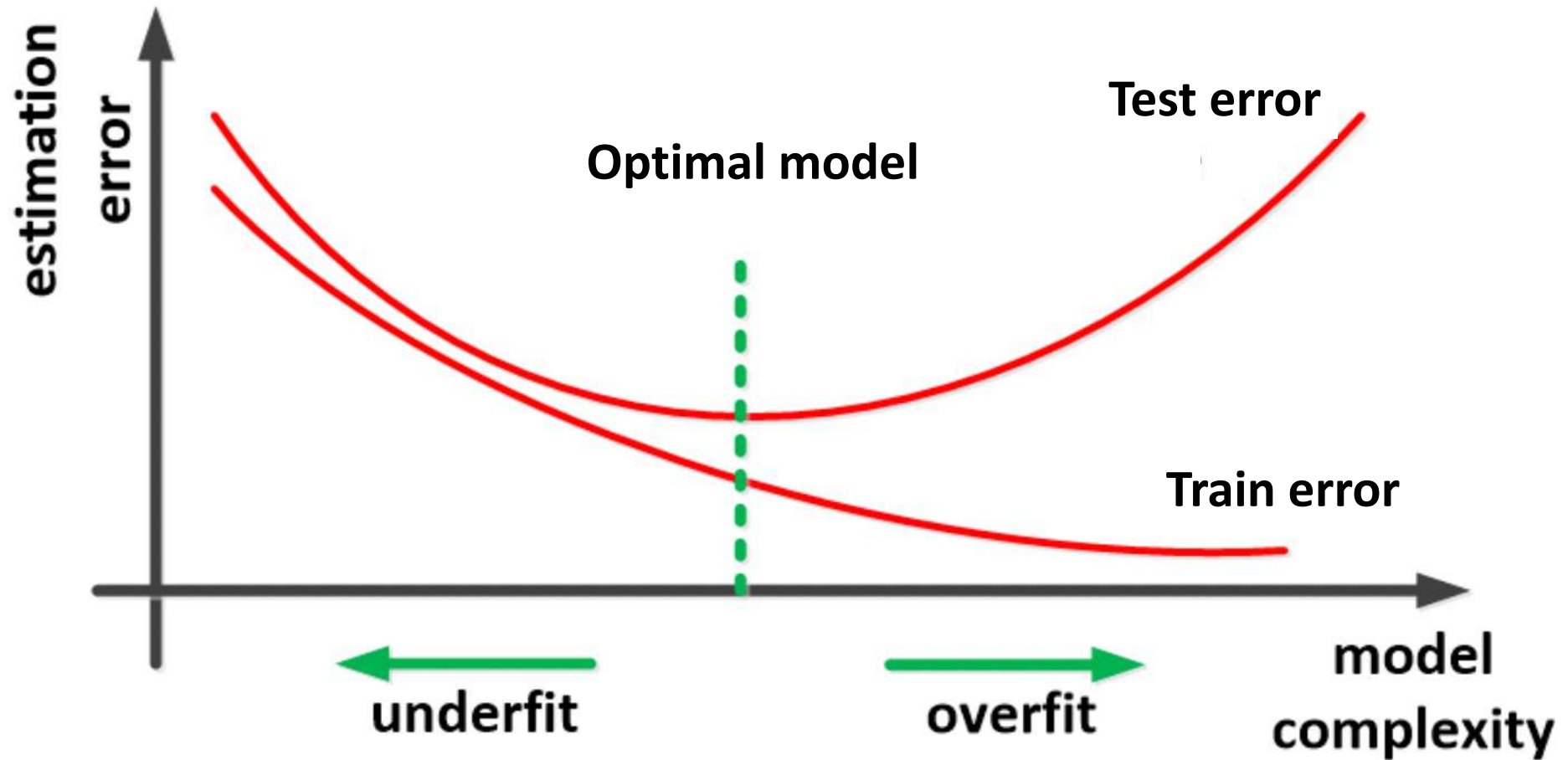
Over-fitting and generalization

- Many parameters and few training data leads to over-fitting
- If it over-fits, the network cannot **generalize**
- To generalize means to be able to predict on unseen (test) data



From A Krogh (2008) Nat. Biotech. 26, p. 195

Over-fitting and generalization



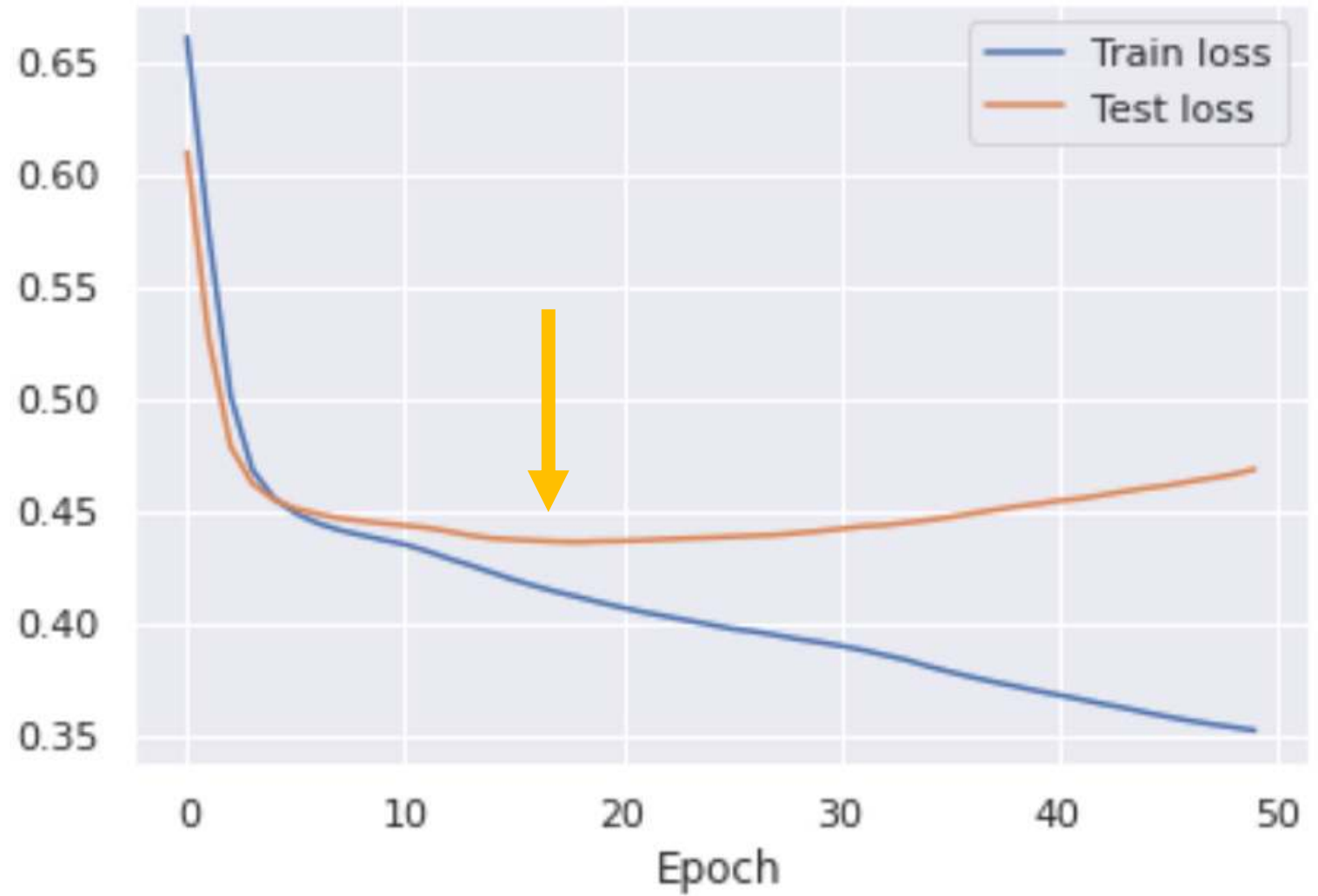
Over-fitting

Sign of over-fitting:

Test error starts to grow while training error decreases

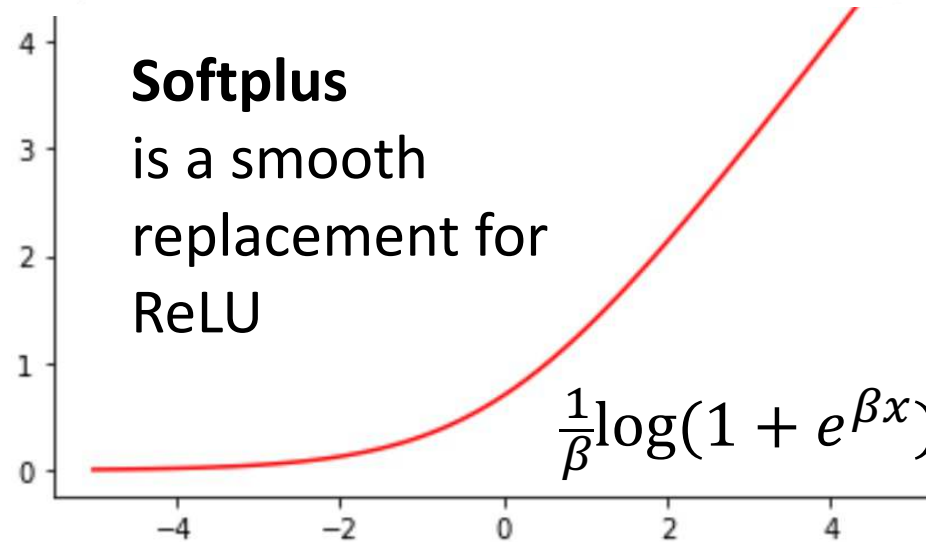
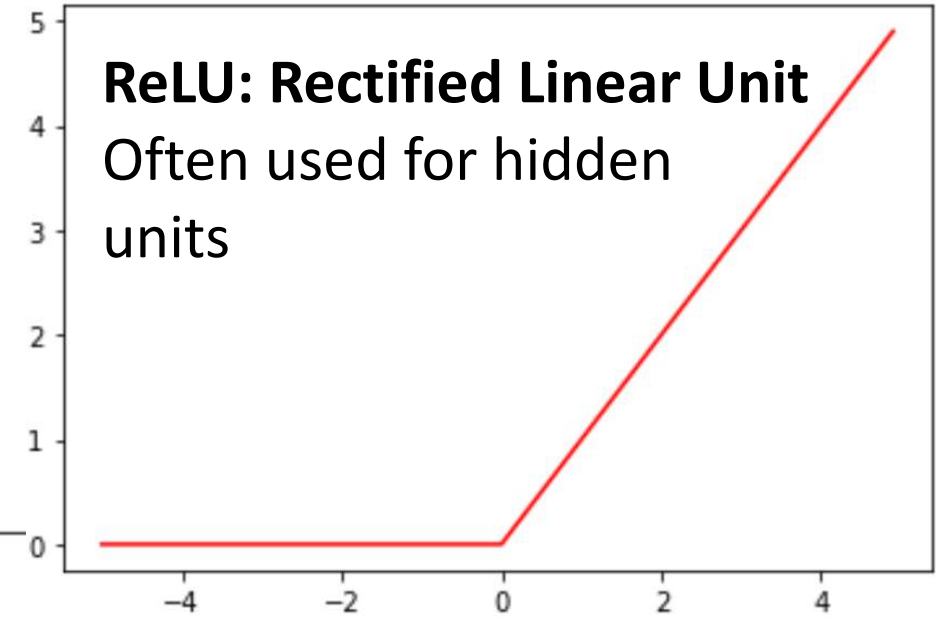
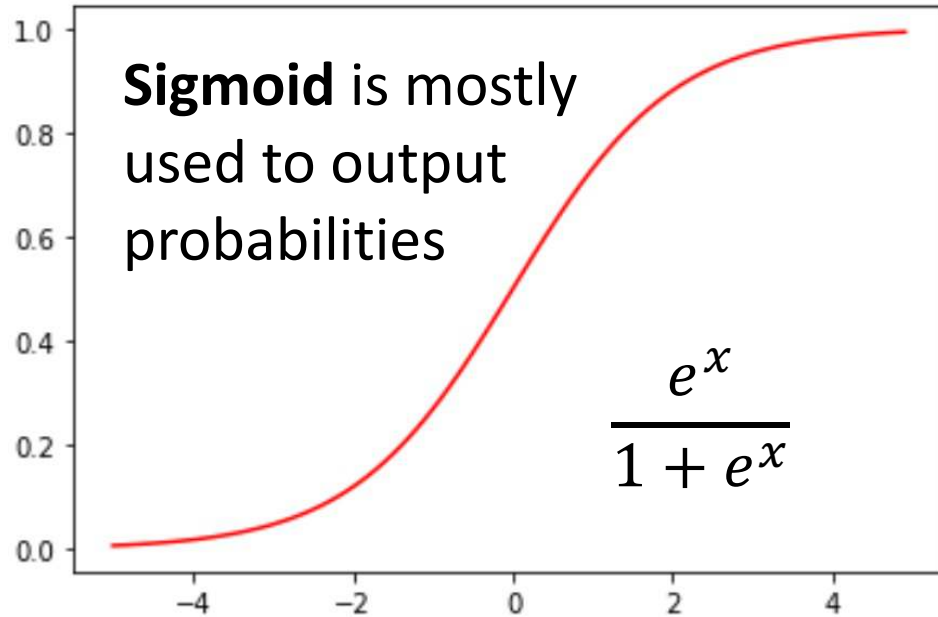
The network size can be decreased if it over-fits (e.g. fewer hidden units)

Alternatively, a **weight decay** can mitigate over-fitting



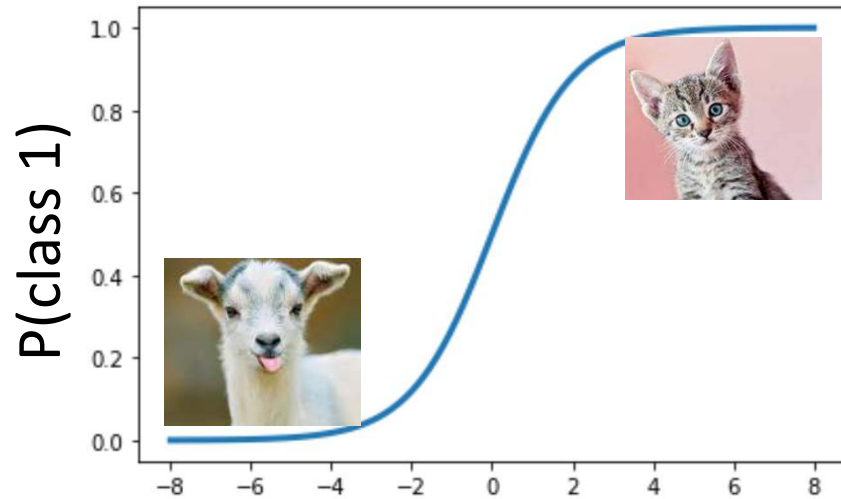
Weight decay: a term λw is subtracted from a weight w in each iteration. λ is normally small, 10^{-2} to 10^{-6}

Activation functions

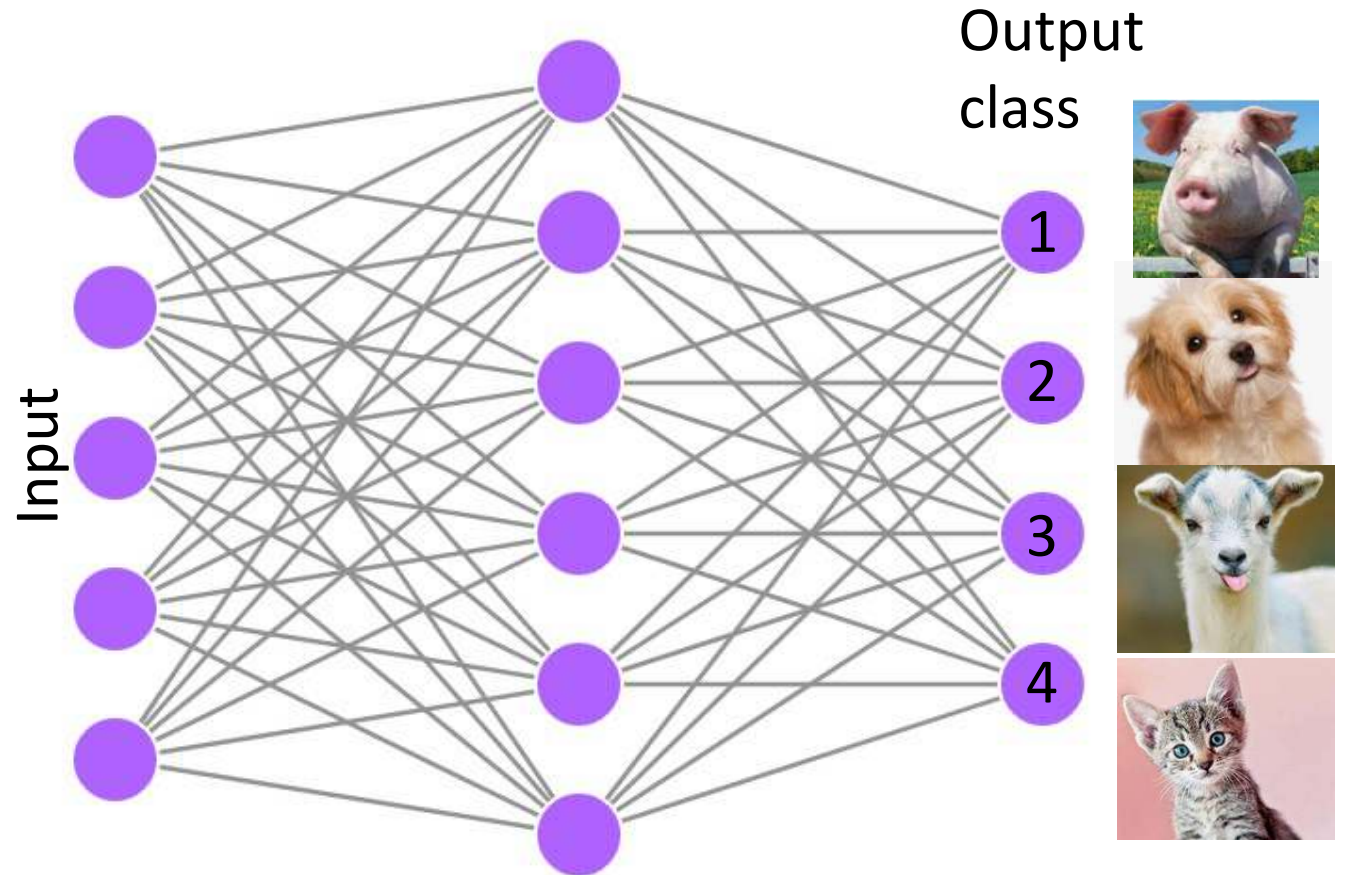


What if we have multiple classes?

Sigmoid for two classes



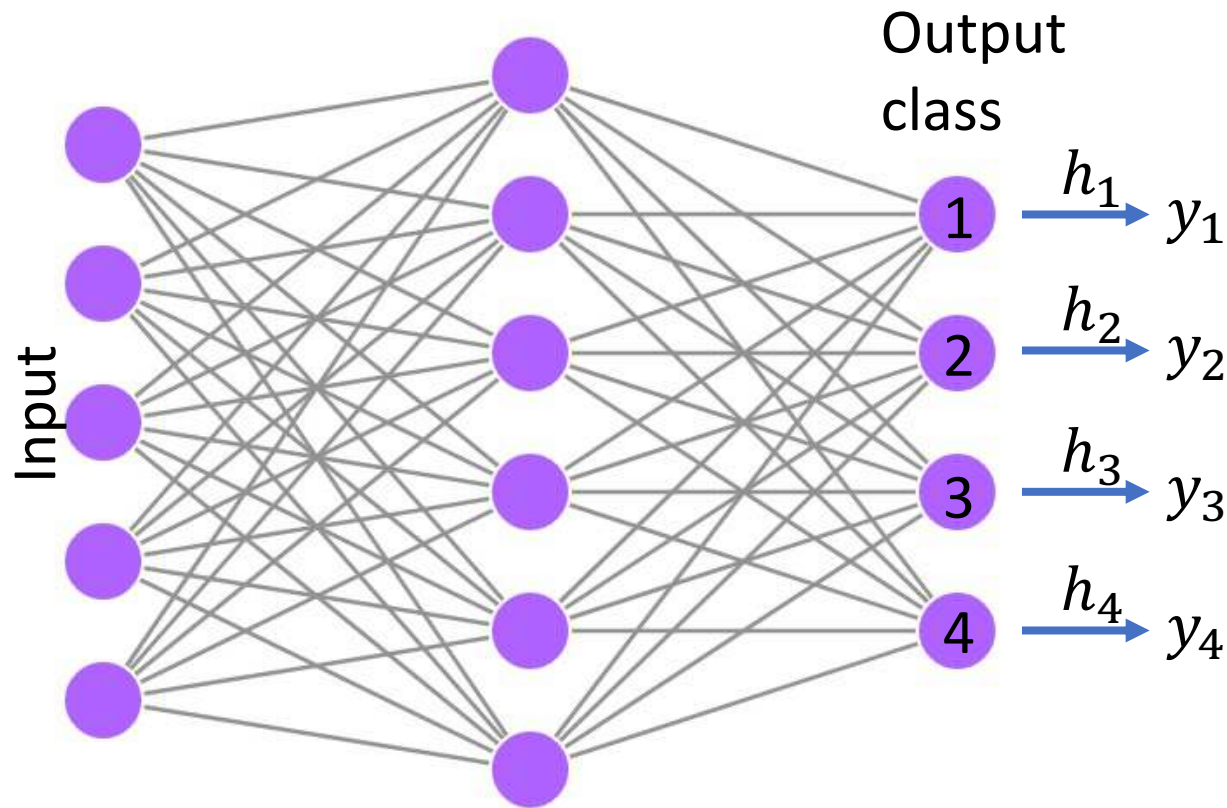
$$P(\text{class } 0) = 1 - P(\text{class } 1)$$



We would like the network to output the probability of each class

Multiple classes: Softmax

Use the softmax function to ensure probabilities sum to one



Weighted sum for the last layer is called h_i for class i

Output for class i :

$$y_i = \text{softmax}(h_i) = \frac{e^{h_i}}{\sum_{j=1}^n e^{h_j}}$$

- **Loss function:** cross entropy *
- In pytorch, the softmax is built into the loss function: it takes h_i instead of y_i .

*) Cross entropy loss is similar to binary case.

$$E(w) = -\sum_i t_i \log y_i \quad \text{where target } t_i \text{ is 1 or 0}$$

The optimizer

- In plain stochastic gradient descent (torch.optim.SGD) you need to set parameters (learning rate and momentum)
- The Adam optimizer is usually a better choice
 - It automatically adapts the learning rate and momentum in clever ways
 - It is based on SGD and uses mini-batches
 - you can set a weight decay

Example of code using the Adam optimizer:

```
optimizer = torch.optim.Adam(nn.parameters())  
for epoch in range(nepochs):  
    for x,t in train_loader:  
        optimizer.zero_grad()  
        y = nn(x)  
        loss = lossfunc(y,t)  
        loss.backward()  
        optimizer.step()
```

You can set parameters in Adam, such as

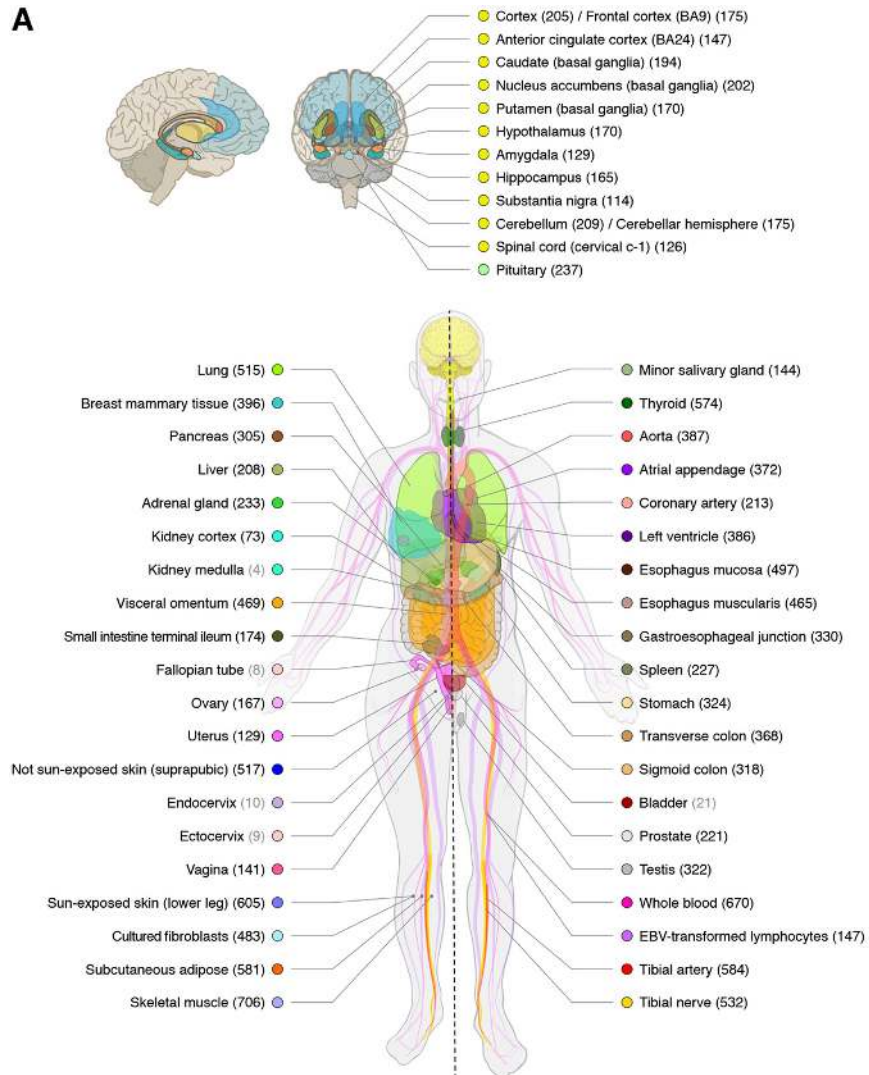
- learning rate (e.g. “lr=1.e-4”)
- “weight_decay=1.e-5”

All the choices you have to make ...

- There are many parameters you can vary in a Neural Network.
- It is a good idea to make an initial “grid search” where you systematically test performance by varying
 - the number of hidden layers and their size
 - other parameters one by one
- This is sometimes done on a reduced data set and or with quite few iterations

Exercise with gene expression data

A



- RNA-seq is a great use-case for Machine Learning algorithms:
 - High dimensional data
 - Many cellular pathways are highly correlated
 - Tissue specific
 - Big datasets available (GTEx)
- The exercise:
 - Can a neural network capture the information encoded on gene expression and detect tissues?
 - In the exercise we will build a neural network that learns the gene expression profile and is able to guess the tissue of origin