

# Neural Networks 3

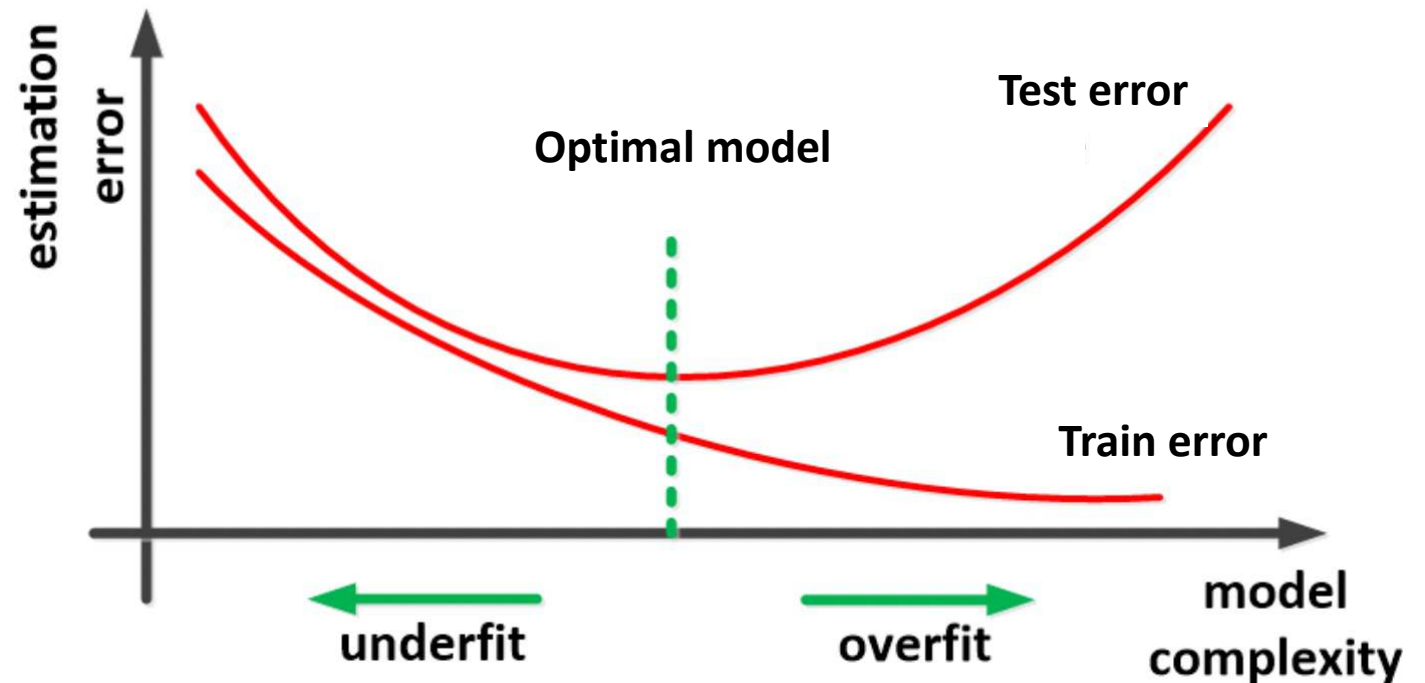
Anders Krogh

Center for Health Data Science

University of Copenhagen

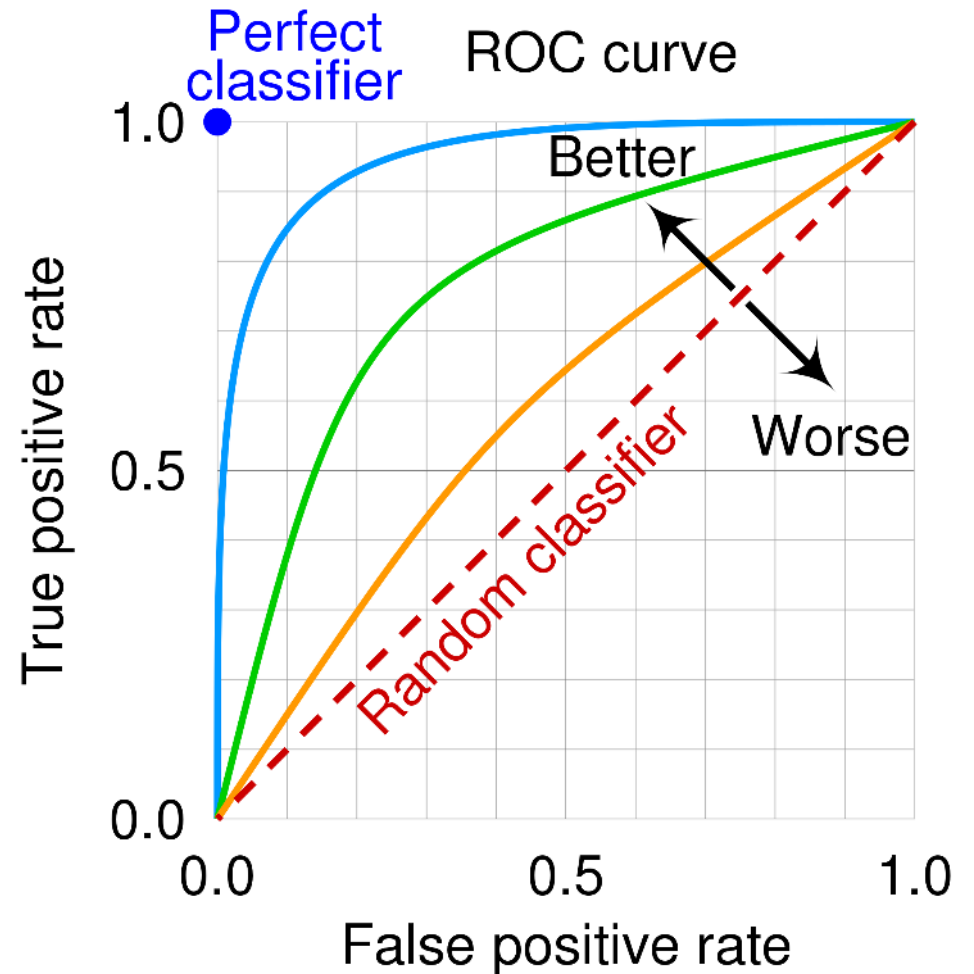
# Performance evaluation

- Neural networks should be evaluated like other machine learning methods on [independent test data](#)
- Additional danger with neural networks: Selection of network type/architecture using test set leads to biased test results
- Use **validation set** for model selection and “early stopping”
- Use **test set once** in the very end of the project



# ROC curves for evaluation of classification

- To classify you select a **cut-off** on the output (e.g. 0.5)
- You can then calculate accuracy
- ROC curves give a performance overview **independent of cut-off**
- **AUC**: area under the curve is an overall measure of performance that should be  $>0.5$  and  $<1$ .



TP = correctly predicted positives (cats)  
TP rate =  $TP / \text{real\_positives}$  (fraction correctly predicted cats)  
FP = incorrectly predicted as positives  
FP rate =  $FP / \text{real\_negatives}$

# Letters/sequences as input

What if the input DNA or protein sequences – how are they coded?

DNA sequence

AGTCACCACCGCTGGCAAGCCACATGTA

**BAD IDEA!**

1342122122324332113221214341

Use **one-hot encoding**

AGTCACCACCGCTGGCAAGCCACATGTA

1000100100000000110001010001  
0001011011010001000110100000  
0100000000100110001000000100  
0010000000001000000000001010

2D tensor

Make 1D tensor (flatten)

10000001010000010010000010000100000100000100000010000010010010...  
A G T C A C C A C C G C T G ...

# One-hot tensor magic

```
seq='TGTAGCTCTCAG'  
onehot2d = one_hot_dna(seq)  
print(onehot2d)  
print(onehot2d.shape)
```

```
tensor([ [0, 0, 0, 1],  
         [0, 0, 1, 0],  
         [0, 0, 0, 1],  
         [1, 0, 0, 0],  
         [0, 0, 1, 0],  
         [0, 1, 0, 0],  
         [0, 0, 0, 1],  
         [0, 1, 0, 0],  
         [0, 0, 0, 1],  
         [0, 1, 0, 0],  
         [1, 0, 0, 0],  
         [0, 0, 1, 0]])  
torch.Size([12, 4])
```

```
# Turn it into a 1D tensor with flatten  
onehot1d=onehot2d.flatten()  
print(onehot1d)  
print(onehot1d.shape)  
# View can do the same with view (4*12=48)  
print(onehot2d.view(48))
```

```
tensor([0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,  
        0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0])  
torch.Size([48])  
tensor([0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,  
        0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0])
```

The method `view()` can  
change the shape of a  
tensor.

A dimension with -1  
will be filled to fit the  
remaining dimensions

```
# Turn a 1D tensor into 2D with view  
# You can use -1 for one dimension  
new2d = onehot1d.view(-1,4)  
print(new2d)
```

```
tensor([[0, 0, 0, 1],  
        [0, 0, 1, 0],  
        [0, 0, 0, 1],  
        [1, 0, 0, 0],  
        [0, 0, 1, 0],  
        [0, 1, 0, 0],  
        [0, 0, 0, 1],  
        [0, 1, 0, 0],  
        [0, 0, 0, 1],  
        [0, 1, 0, 0],  
        [1, 0, 0, 0],  
        [0, 0, 1, 0]])
```

# Convolution

Convolution is used when data are ordered like

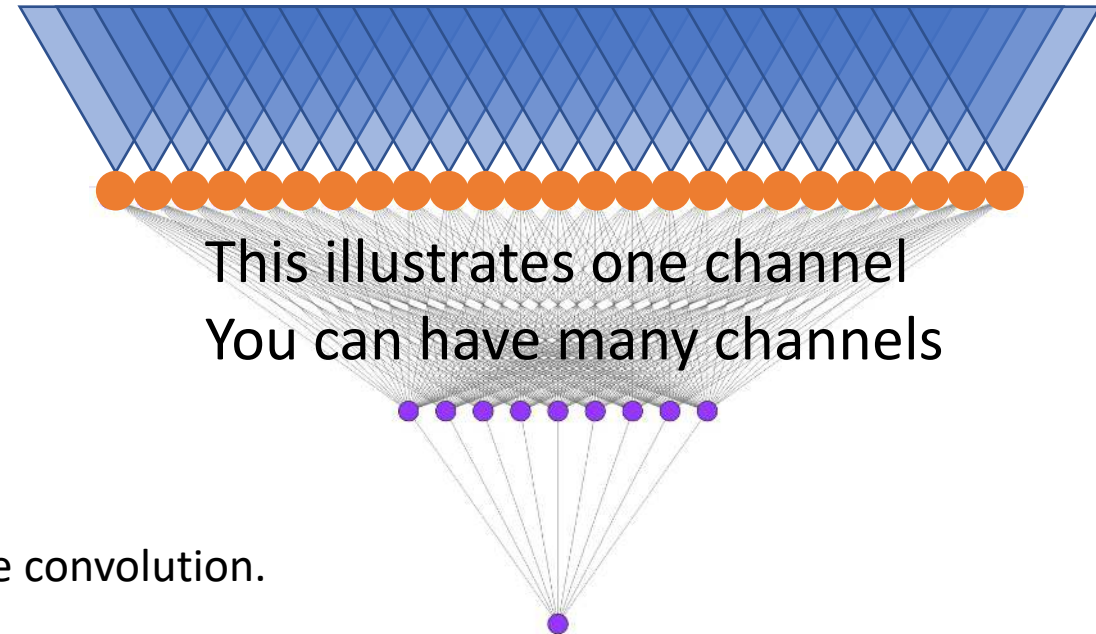
- Letters in a sequence
- Words in a sentence
- Pixels in an image

Cannot be used for gene expression values

- The **kernel size** is the input geometry for the convolution.
- In the illustration the kernel size is (5,4)
- **Stride** is the number of inputs you skip. stride=1 here.
- **Padding** can be used to add zeros to each side of the sequence. padding=0 here.

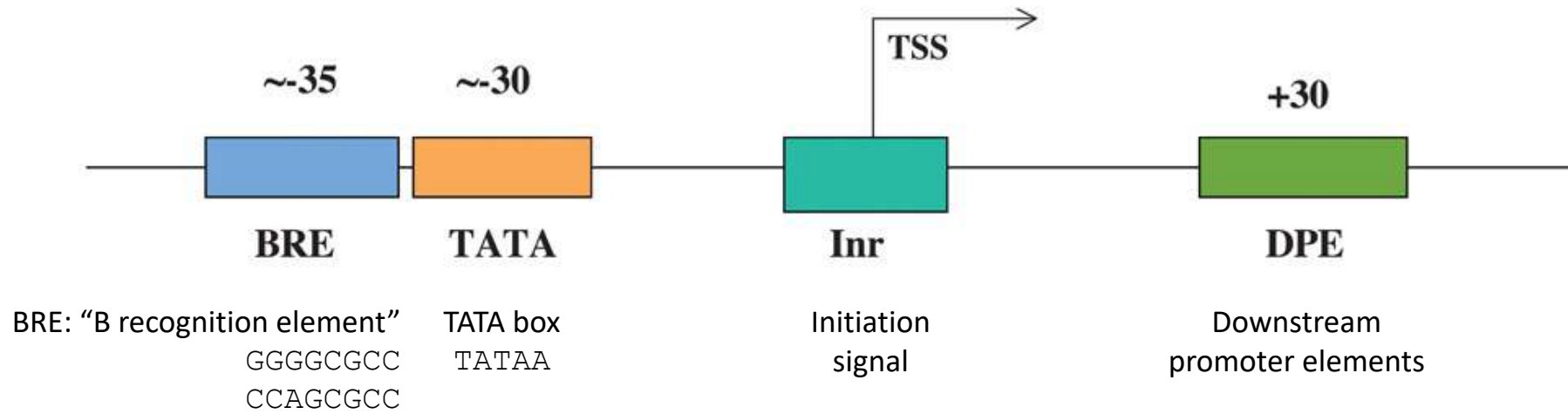
AGTCACCAACCGCTGGCAAGCCACATGTA

100010001000000000110001010001  
0001011011010001000110100000  
0100000000100110001000000100  
0010000000000010000000000001010



# Exercise with human core promoters

The core promoter is the region around the **transcription start site (TSS)**

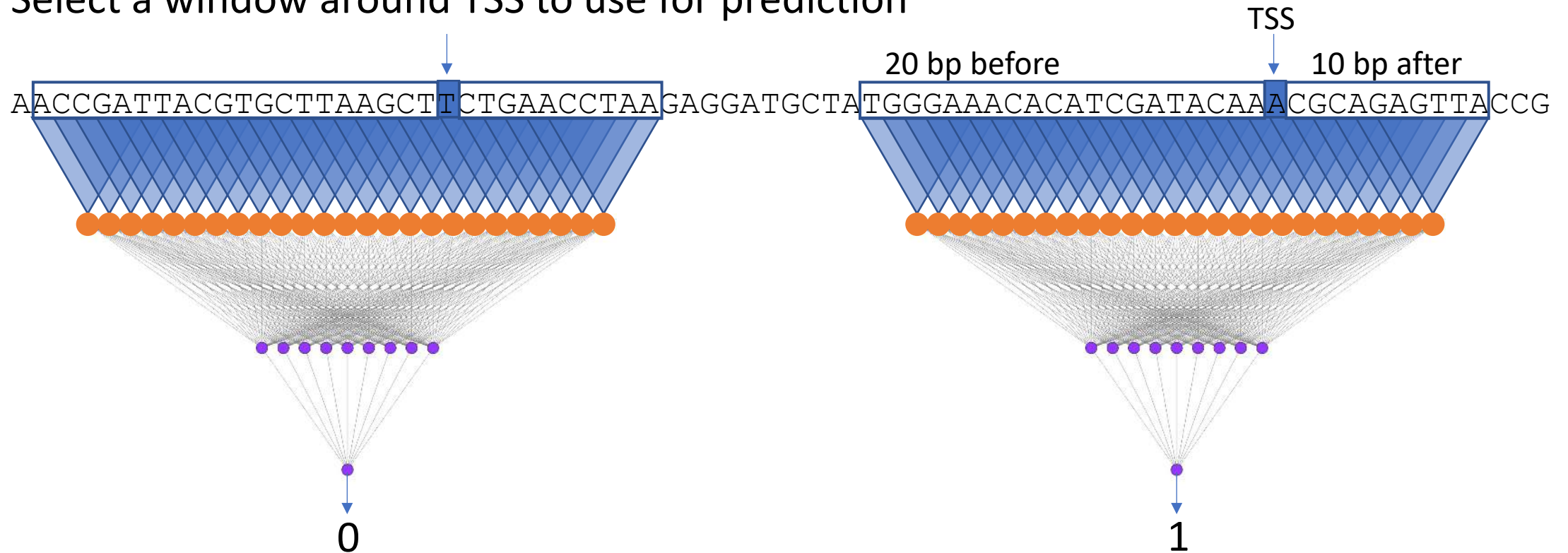


In the exercise we will try to predict the TSS from the DNA sequence around it

figure from Roy A.L. (2005) Core Promoters. In: Encyclopedic Reference of Genomics and Proteomics in Molecular Medicine. Springer, Berlin, Heidelberg . [https://doi.org/10.1007/3-540-29623-9\\_2210](https://doi.org/10.1007/3-540-29623-9_2210)

# Predict TSS from DNA

Select a window around TSS to use for prediction



- We use one-hot encoding
- Negative examples are randomly sampled from +/- 1000 bases around TSS
- You can compare convolution and fully connected networks