

Deep learning

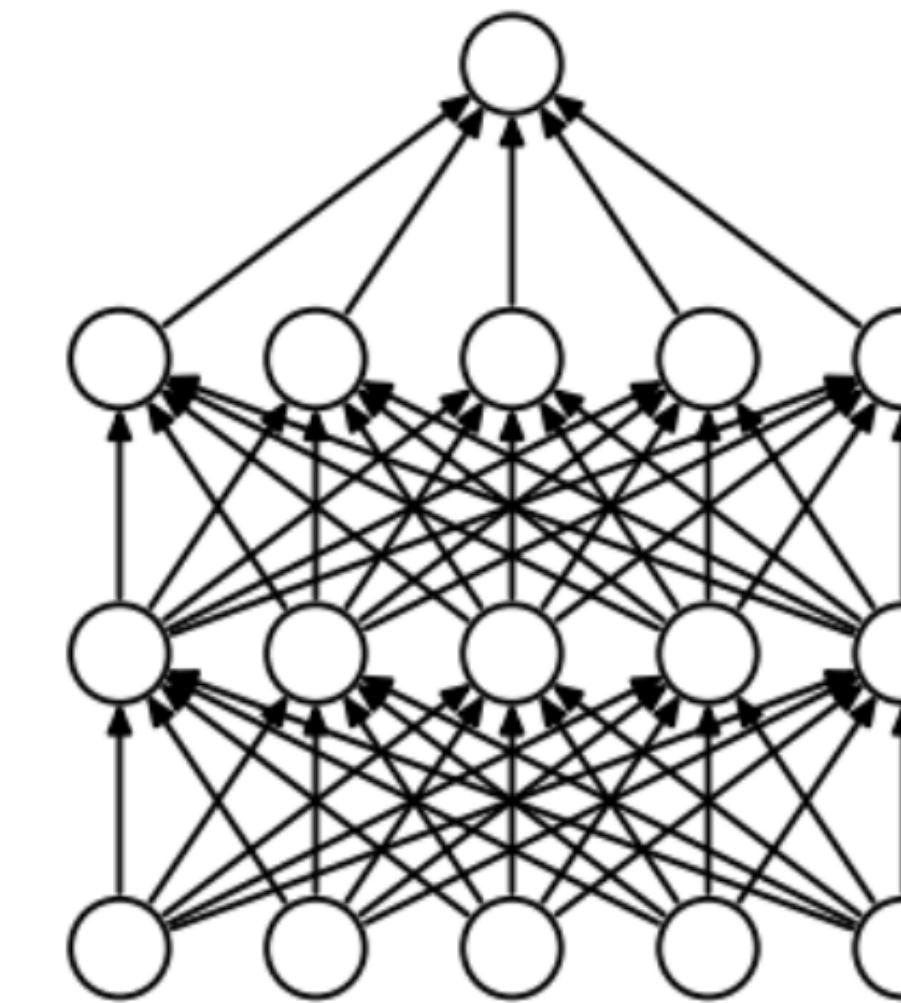
Part I: Introduction

Maurizio Pierini

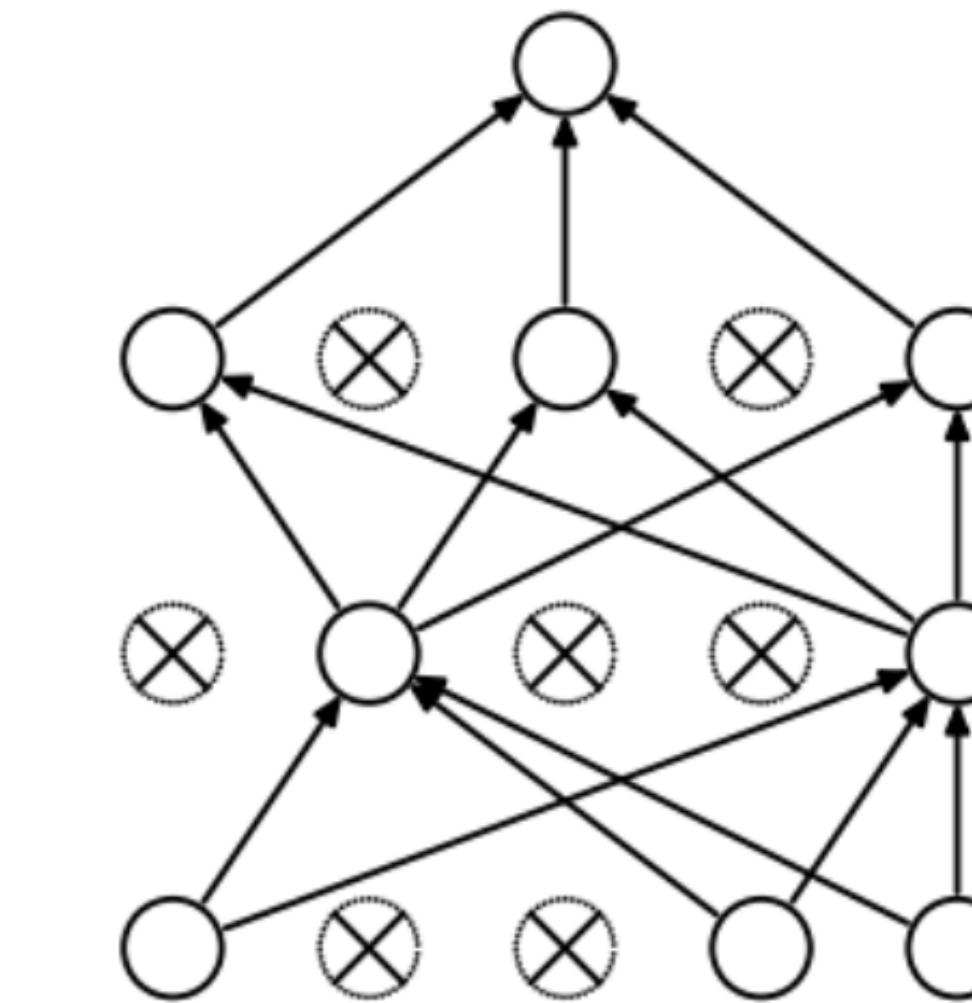


Dropout Layer

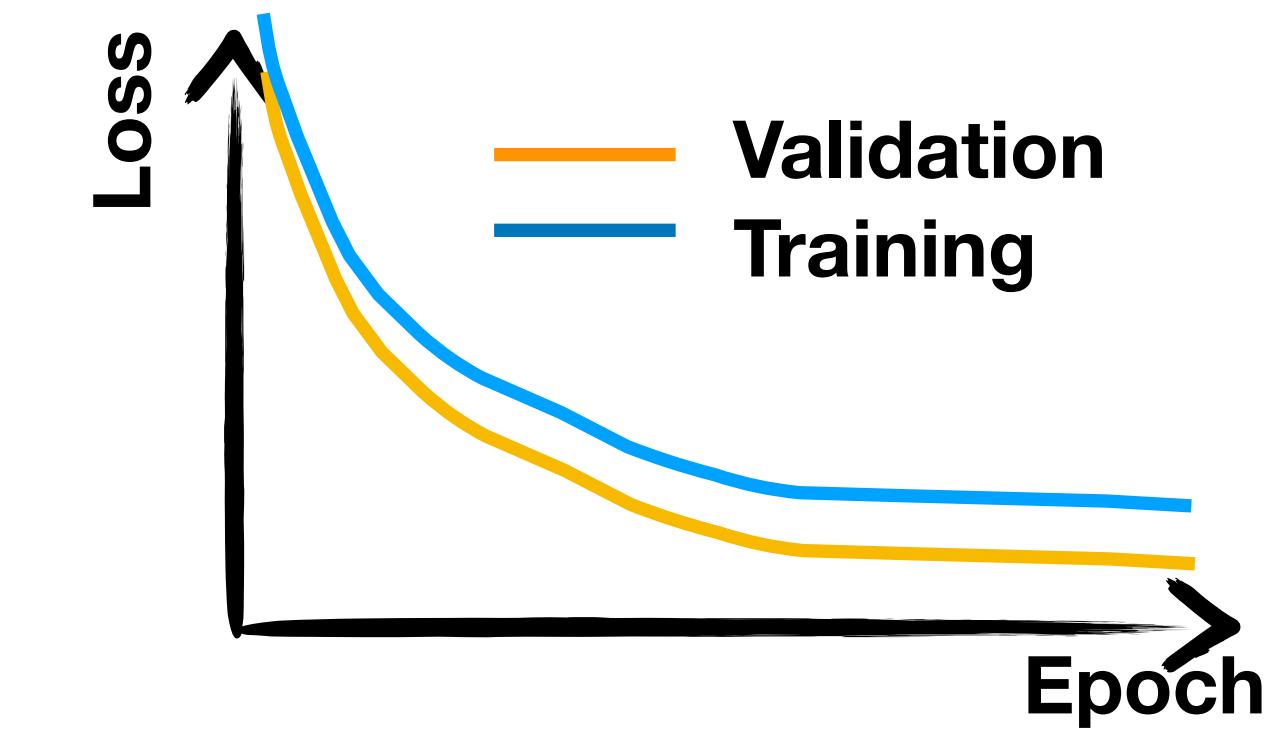
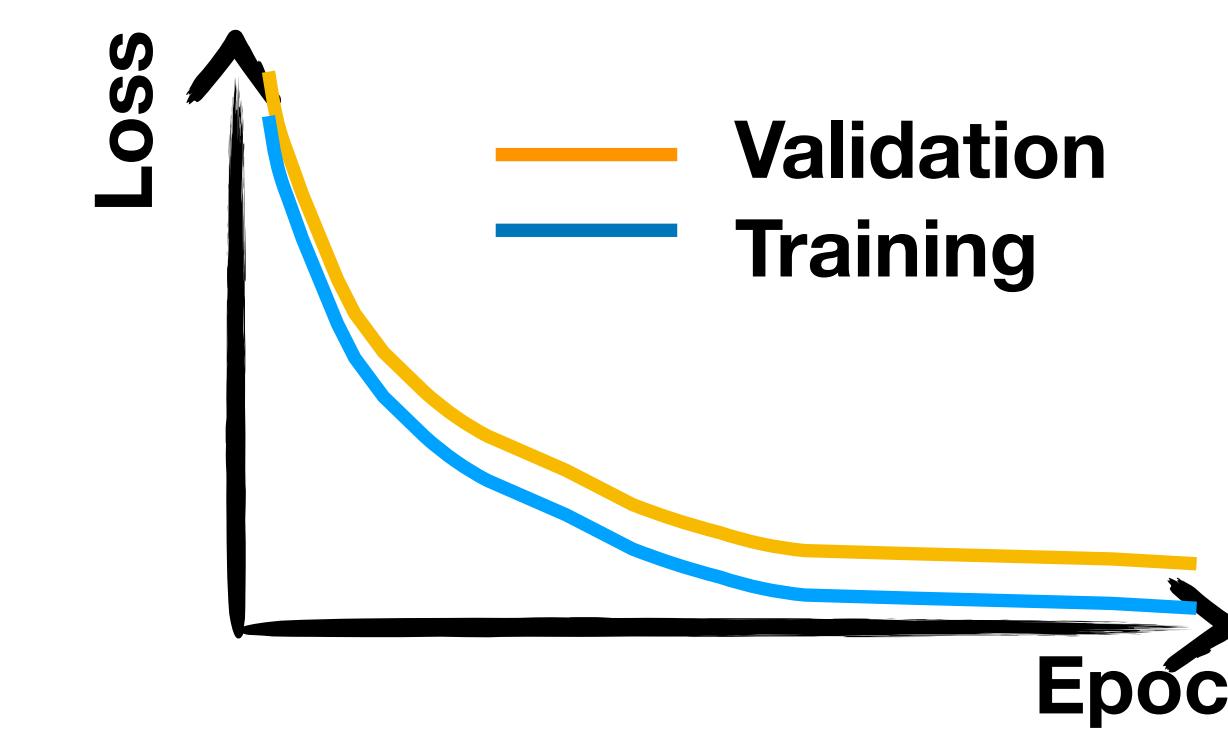
- A special kind of layer, introduced for regularisation purpose
- Randomly drop links between neurons, with probability p
- The connections are re-established during the validation and inference steps
- Typical sign of it: invert hierarchy between training and validation loss



(a) Standard Neural Net

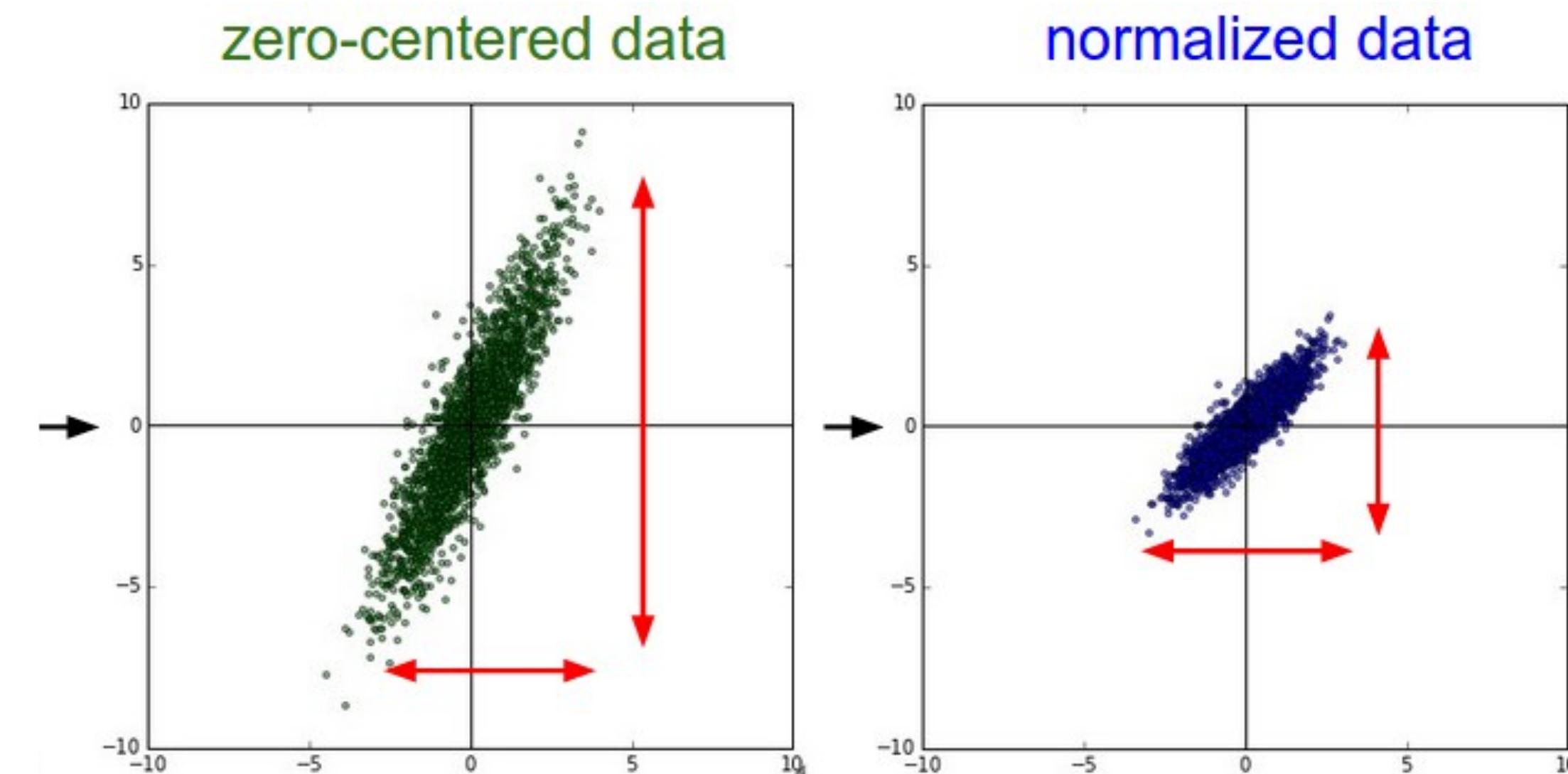
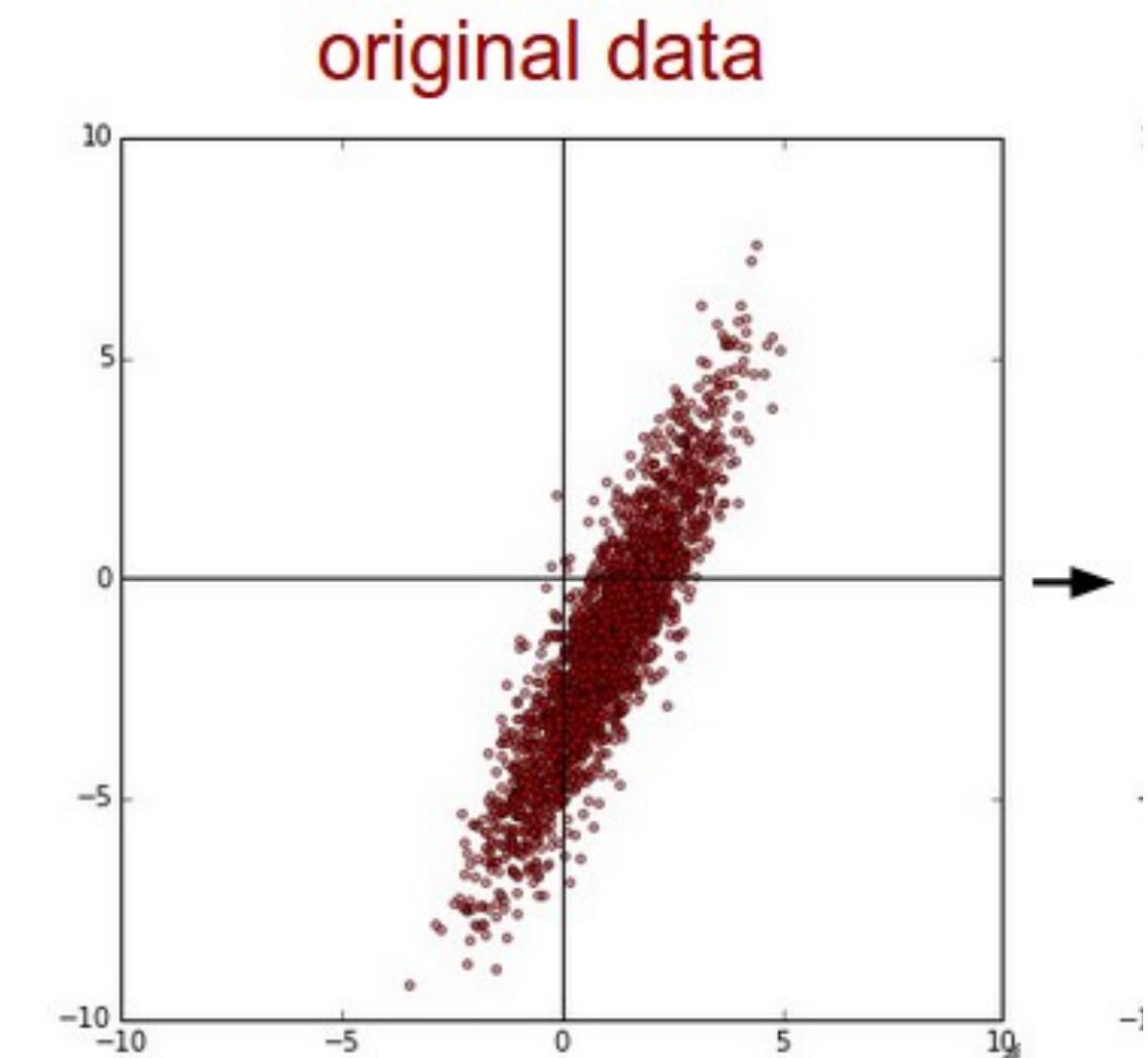


(b) After applying dropout.



BatchNormalization Layer

- It is good practice to give normalized inputs to a layer
- With all inputs having the same order of magnitude, all weights are equal important in the gradient
- Prevents explosion of the loss function
- This can be done automatically with BatchNormalization
- non-learnable shift and scale parameters, adjusted batch by batch



more complex structures

- Dense NN architectures can be made more complex

8|0|4|7|6|8|0 7|9|4|6|5|2|6 8|3|4|5|5|3|4

- Multiple inputs

- Multiple outputs

- Different networks branches

- This is possible thanks to layer-manipulation layers

- Add, Subtract, etc.

- Concatenation

- Flattening

- All these operations are usually provided with NN training libraries

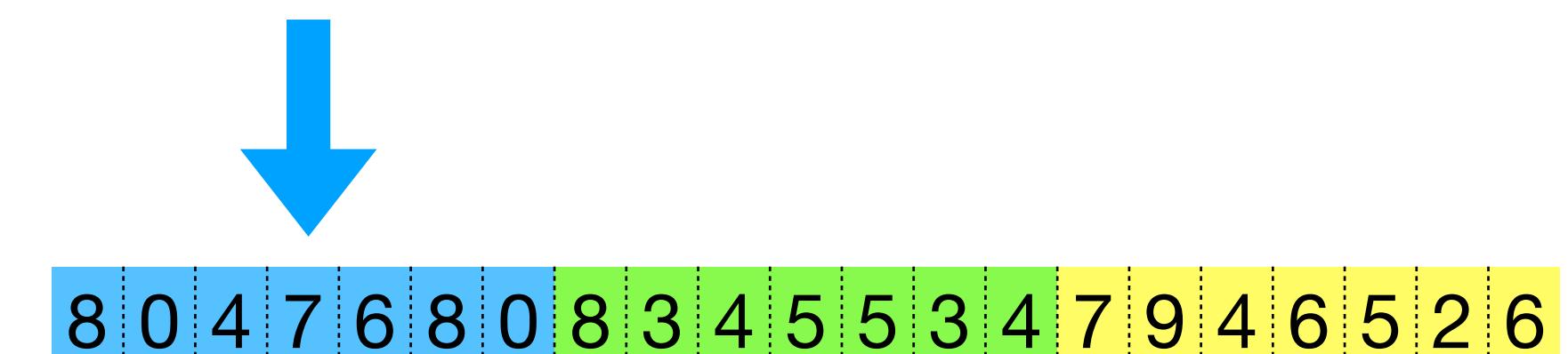


8|0|4|7|6|8|0|8|3|4|5|5|3|4|7|9|4|6|5|2|6

Concatenation

8|0|4|7|6|8|0
8|3|4|5|5|3|4
7|9|4|6|5|2|6

Flattening

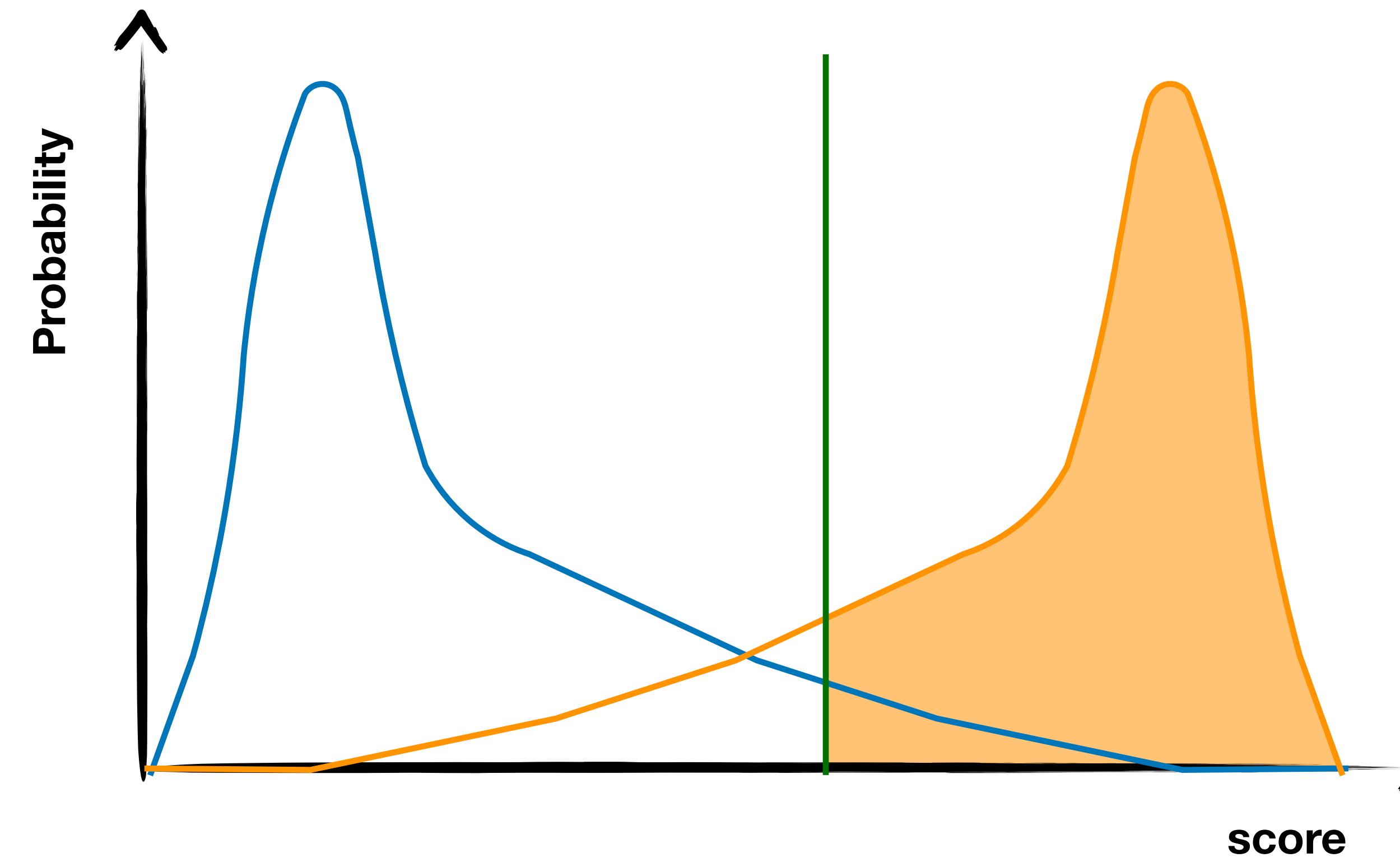


The diagram shows a 3x7 grid of numbers. The top row is blue and contains the numbers 8, 0, 4, 7, 6, 8, 0. The middle row is green and contains the numbers 8, 3, 4, 5, 5, 3, 4. The bottom row is yellow and contains the numbers 7, 9, 4, 6, 5, 2, 6. A large blue arrow points downwards from this grid towards the resulting flattened vector.

8|0|4|7|6|8|0|8|3|4|5|5|3|4|7|9|4|6|5|2|6

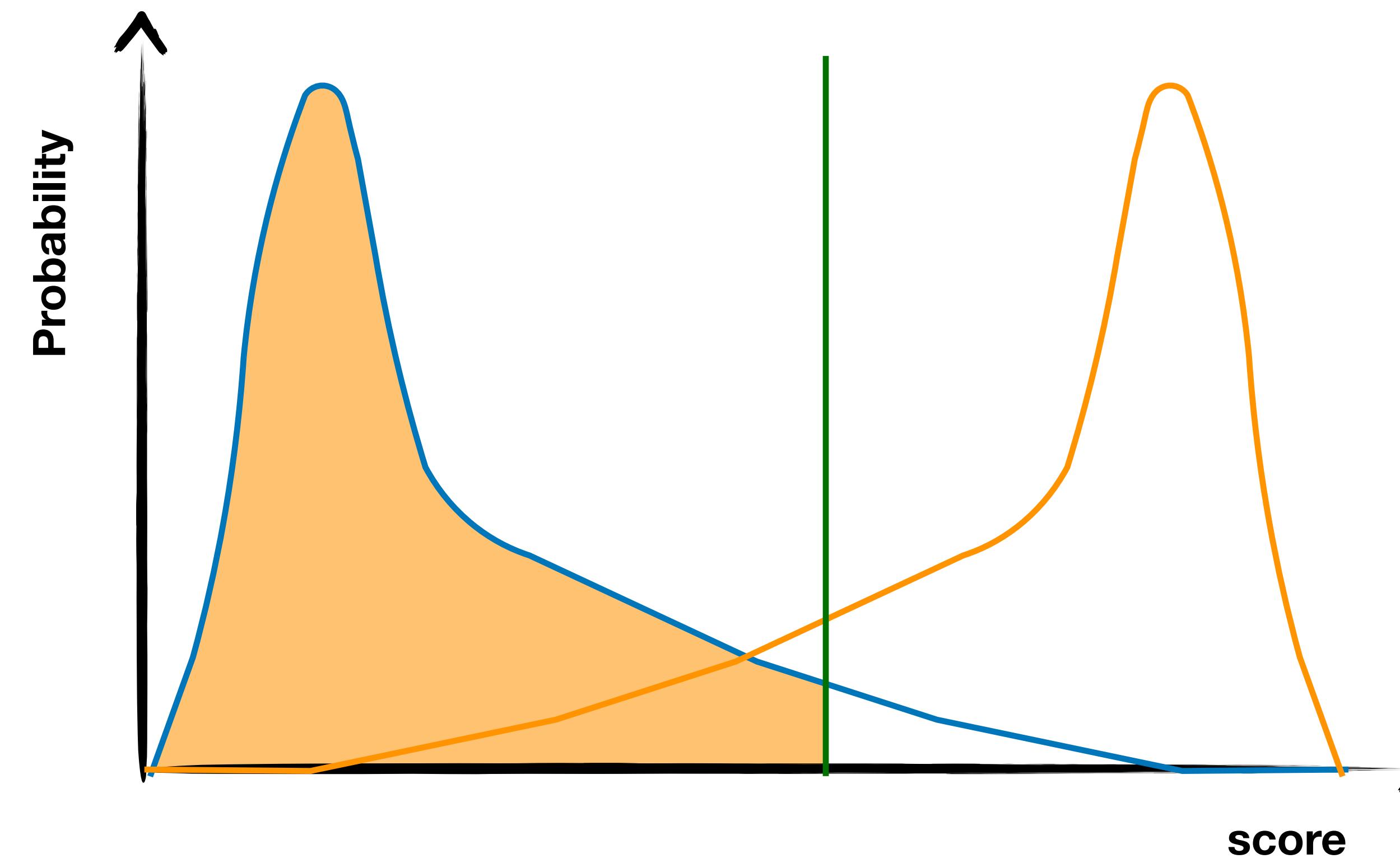
Classifier metrics

- A given threshold defines the following qualities
 - True-positives: Class-1 events above the threshold
 - True-negatives: Class-0 events below the threshold
 - False-positives: Class-0 events above the threshold
 - False-negatives: Class-1 events below the threshold



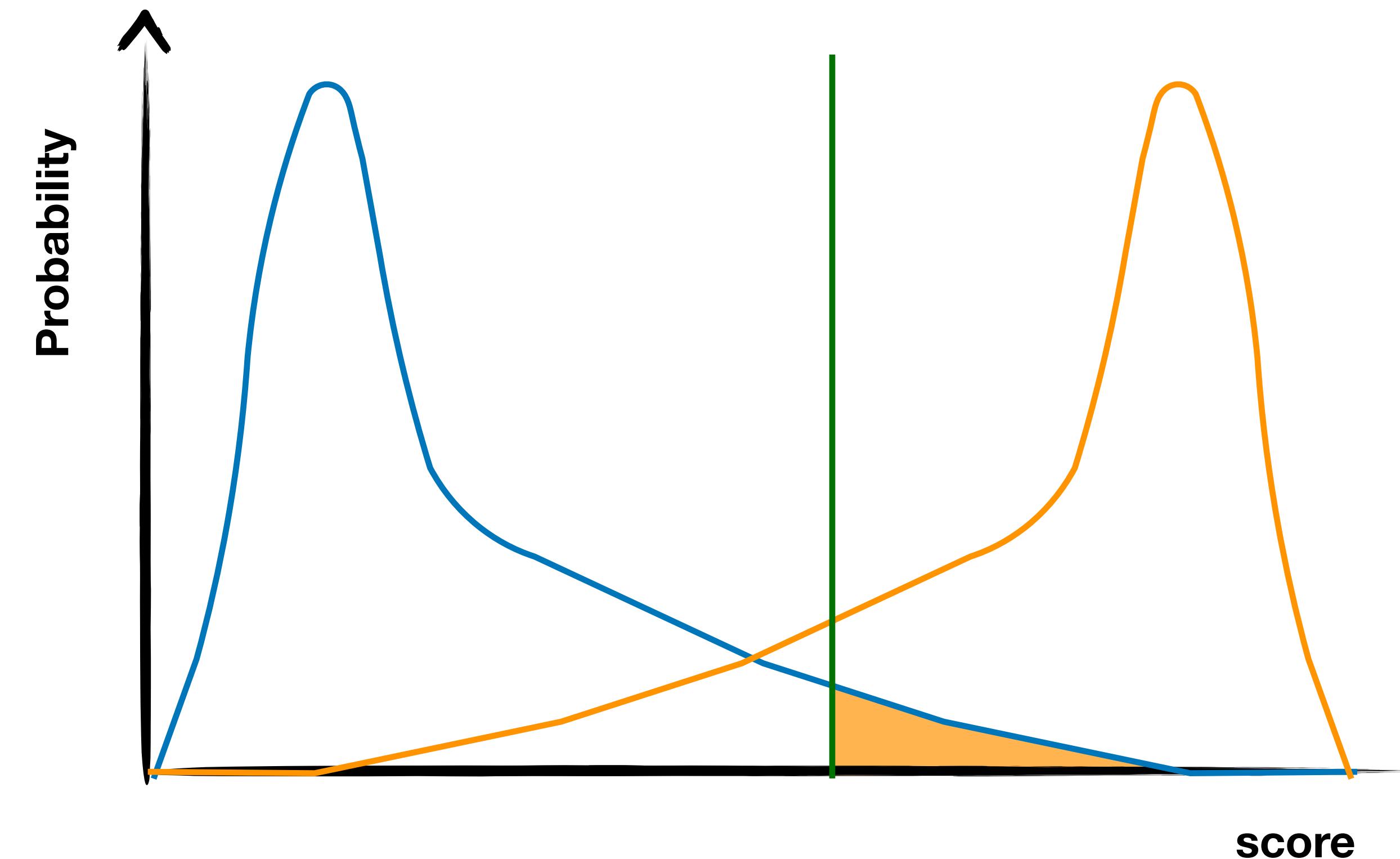
Classifier metrics

- A given threshold defines the following qualities
 - True-positives: Class-1 events above the threshold
 - True-negatives: Class-0 events below the threshold
 - False-positives: Class-0 events above the threshold
 - False-negatives: Class-1 events below the threshold



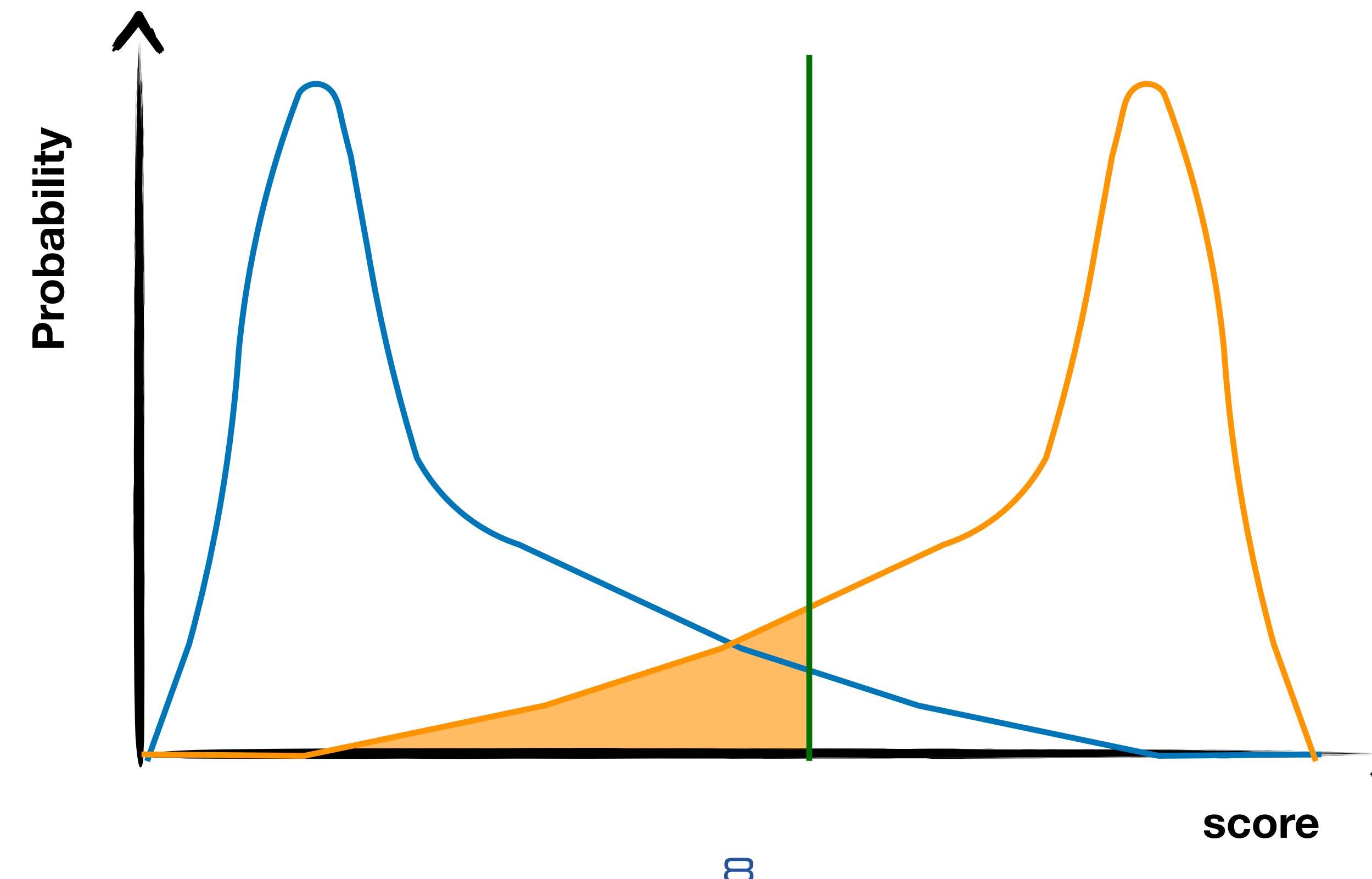
Classifier metrics

- A given threshold defines the following qualities
 - True-positives: Class-1 events above the threshold
 - True-negatives: Class-0 events below the threshold
 - False-positives: Class-0 events above the threshold
 - False-negatives: Class-1 events below the threshold



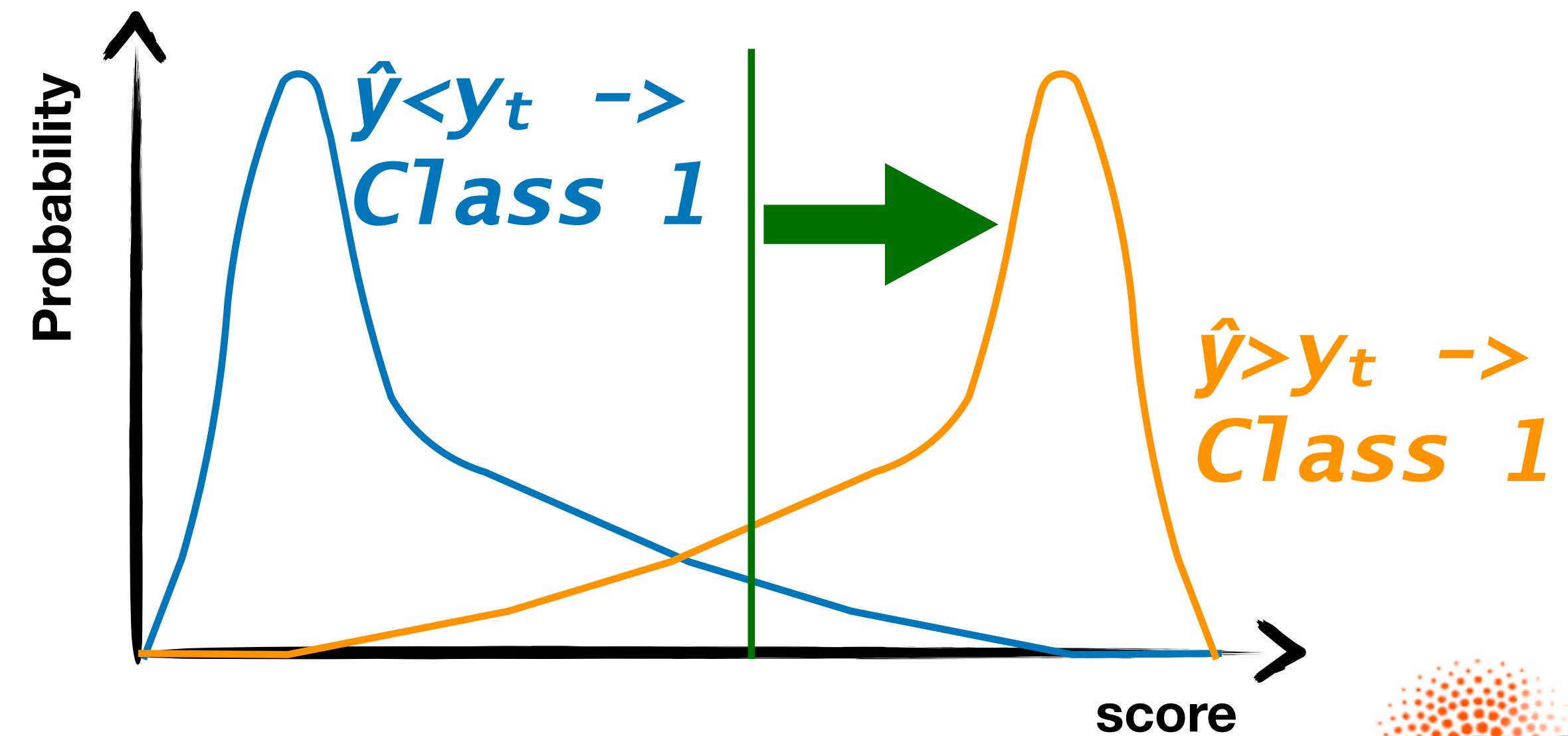
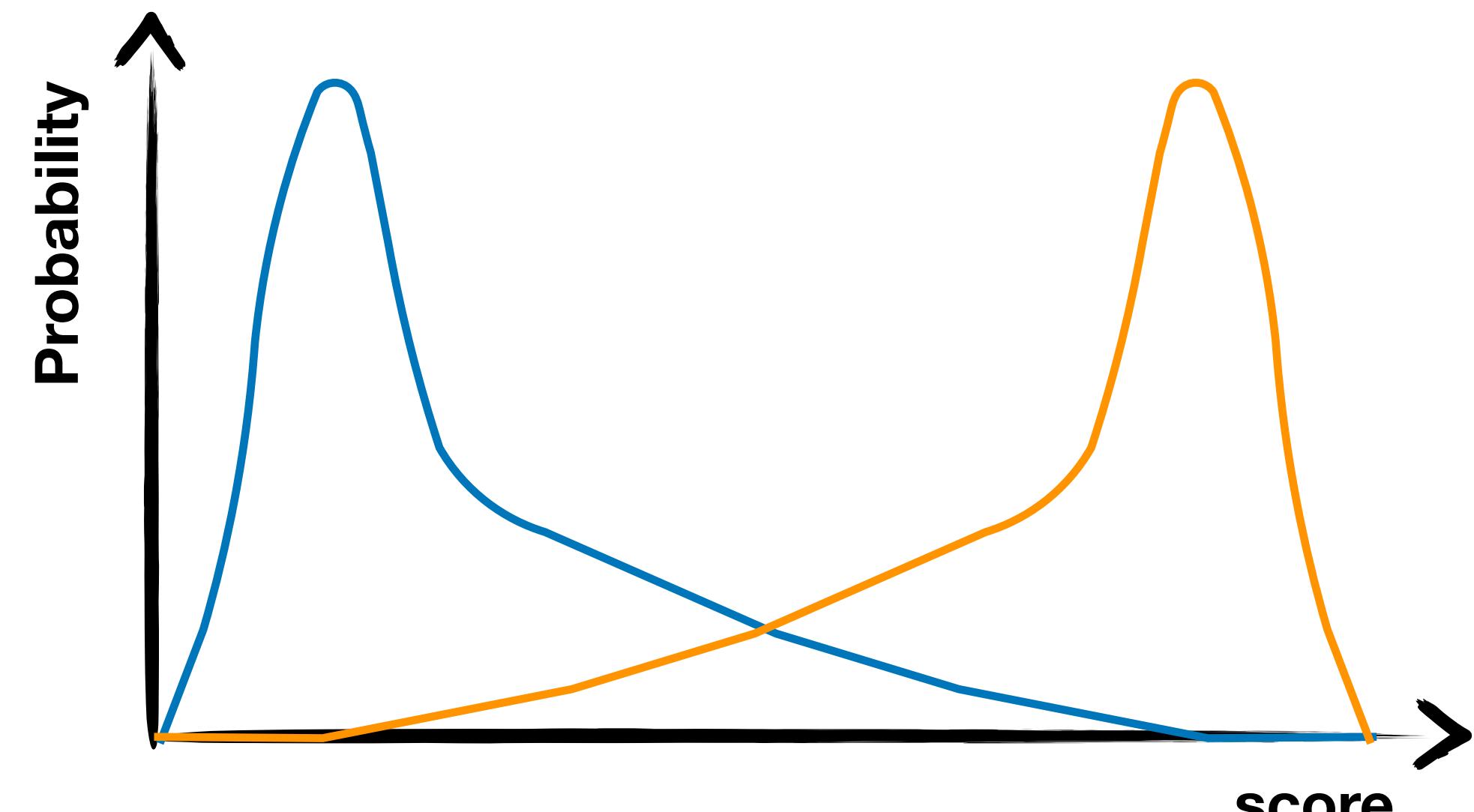
Classifier metrics

- A given threshold defines the following qualities
 - True-positives: Class-1 events above the threshold
 - True-negatives: Class-0 events below the threshold
 - False-positives: Class-0 events above the threshold
 - False-negatives: Class-1 events below the threshold



Classifier metrics

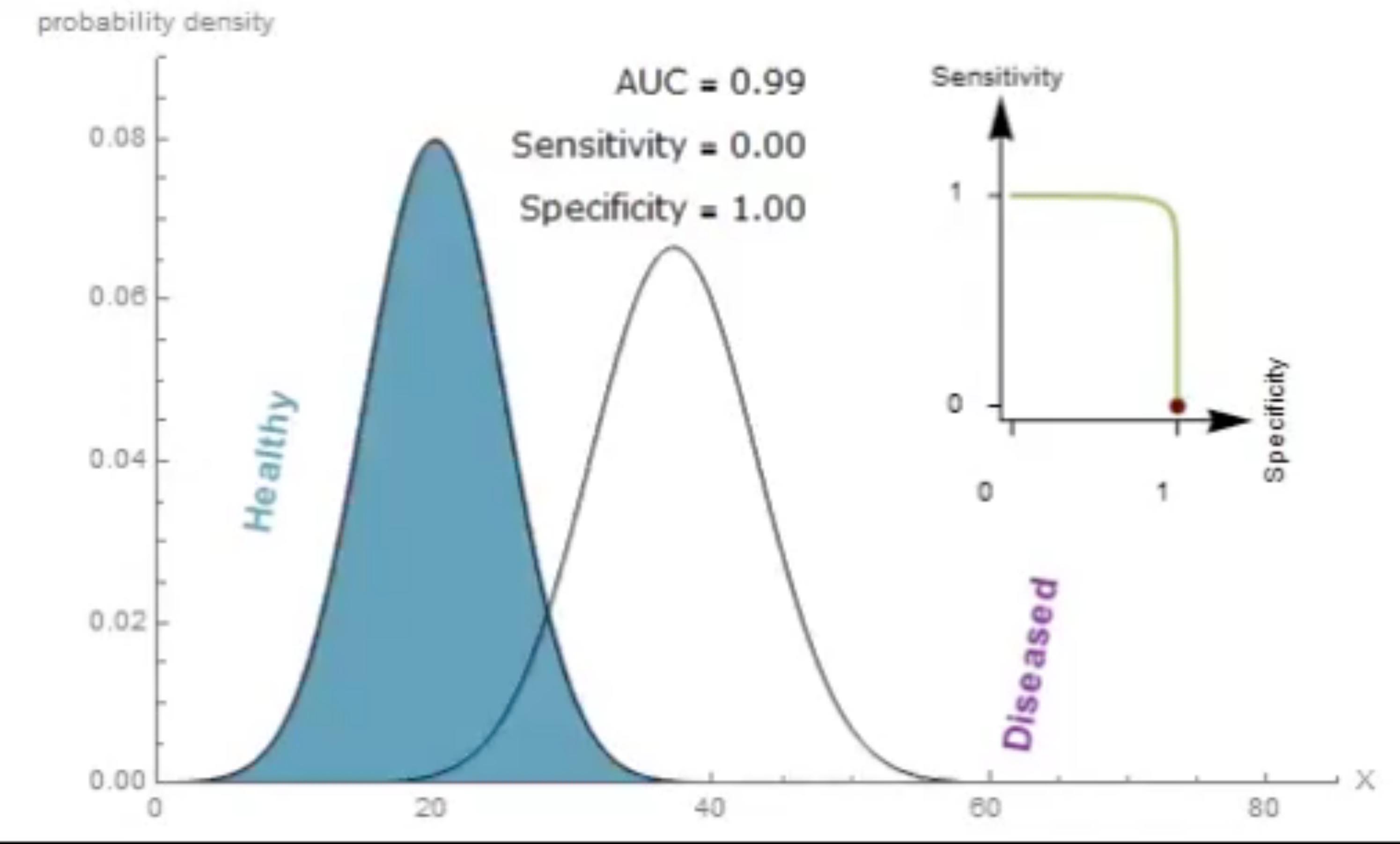
- Consider a binary classifier
- Its output \hat{y} is a number in $[0, 1]$
- If well trained, value should be close to 0 (1) for class-0 (class-1) examples
- One usually defines a threshold y_t such that:
 - $\hat{y} > y_t \rightarrow \text{Class 1}$
 - $\hat{y} < y_t \rightarrow \text{Class 0}$



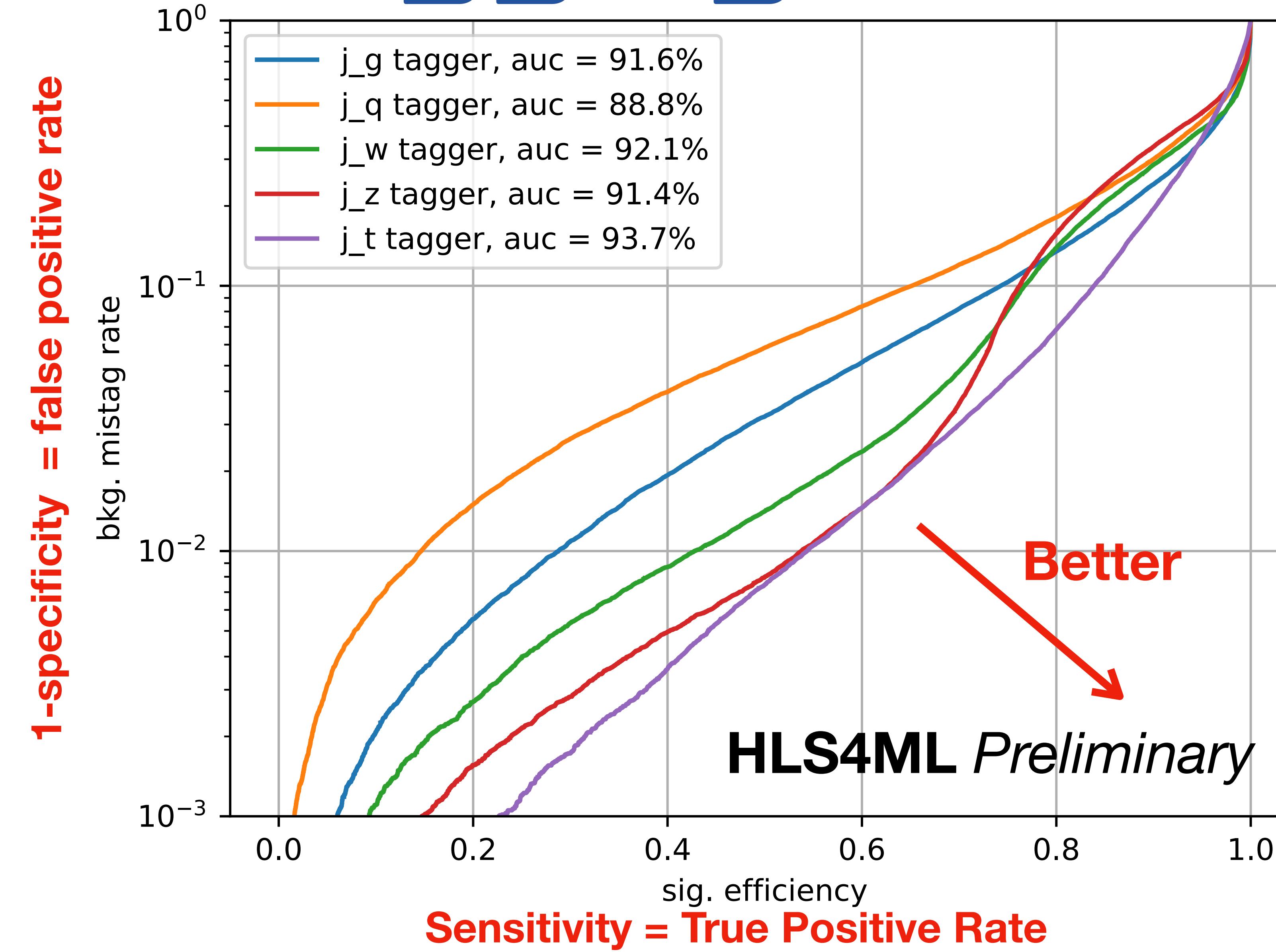
Classifier metrics

- Starting ingredients are true positive (TP) and true negative (TN) rates
- Accuracy: $(TP+TN)/Total$
 - The fraction of events correctly classified
- Sensitivity: $TP/(Total \text{ positive})$
 - AKA signal efficiency in HEP
- Specificity: $TN/(Total \text{ negative})$
 - AKA mistag rate in HEP

Receiver operating characteristic

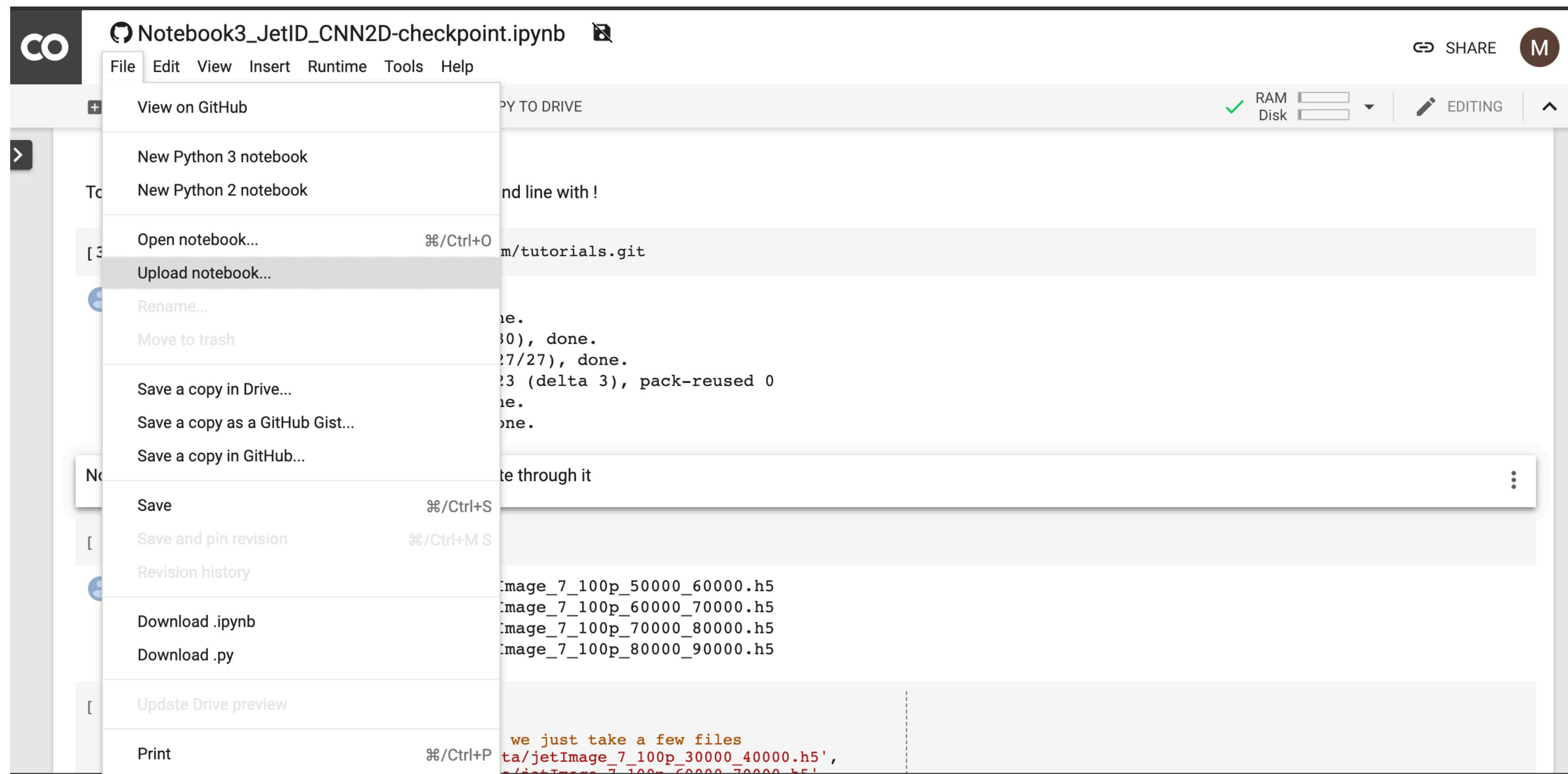


Jet tagging ROC curve



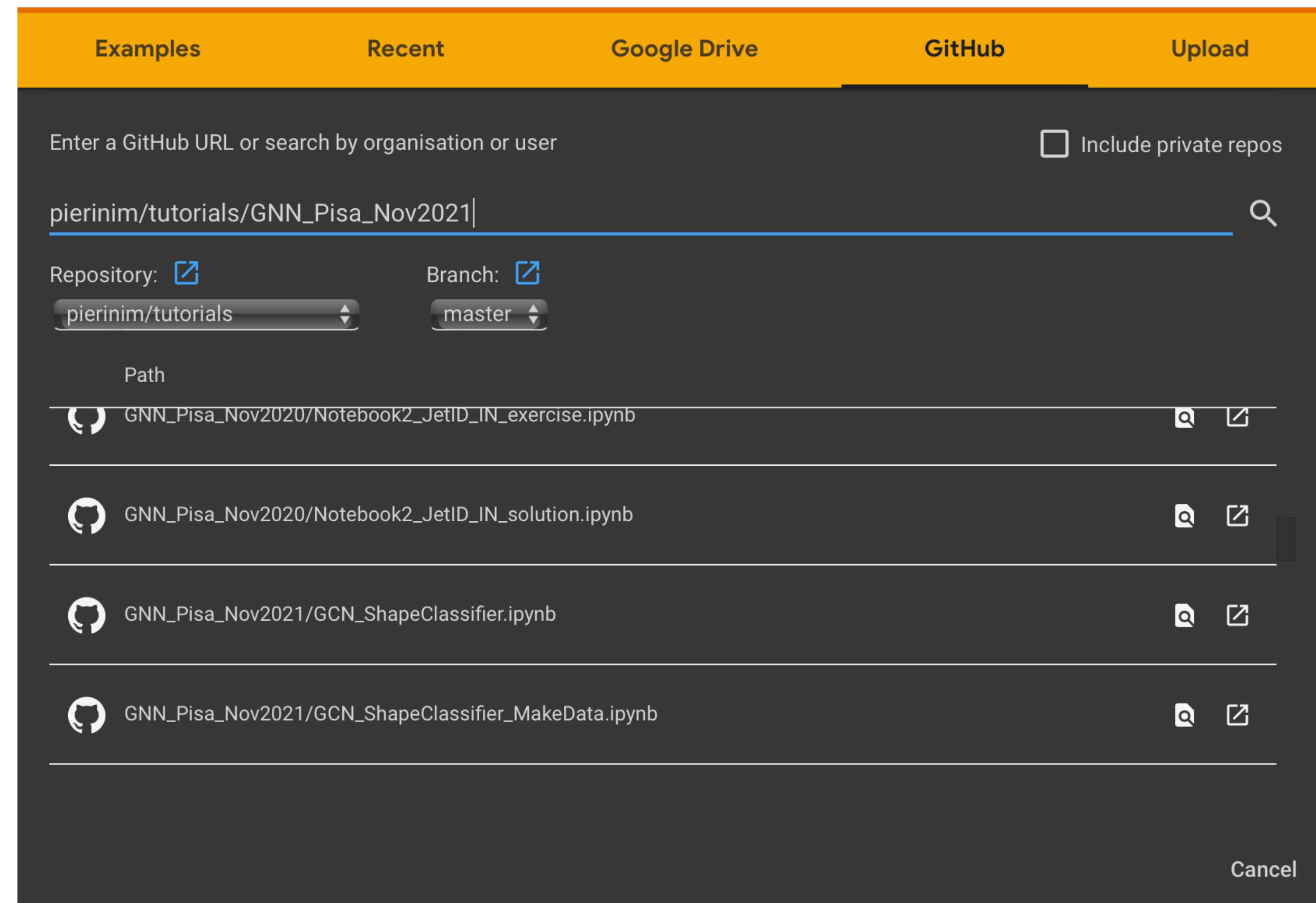
Step1: Open Notebook on Colab

- Go to <https://colab.research.google.com>



Step2: import the Tutorial from gitlab

- Click on the *GITHUB* tab
- Specify the repository *pierinim/tutorials*
- Click on the notebook



Set your resources to use GPUs

Notebook3_JetID_CNN2D-checkpoint.ipynb

File Edit View Insert Runtime Tools Help

CODE TEXT

Run all $\text{\textcircled{z}/}\text{Ctrl+F9}$

Run before $\text{\textcircled{z}/}\text{Ctrl+F8}$

Run the focused cell $\text{\textcircled{z}/}\text{Ctrl+Enter}$

Run selection $\text{\textcircled{z}/}\text{Ctrl+Shift+Enter}$

Run after $\text{\textcircled{z}/}\text{Ctrl+F10}$

Interrupt execution $\text{\textcircled{z}/}\text{Ctrl+M I}$

Restart runtime... $\text{\textcircled{z}/}\text{Ctrl+M .}$

Restart and run all...

Reset all runtimes...

Change runtime type

Manage sessions

View runtime logs

```
[3] ! git clone https://github.com/.../HiggsSchool.git
Cloning into 'tutorials/HiggsSchool...'...
remote: Enumerating objects...
remote: Counting objects...
remote: Compressing objects...
remote: Total 3042
Unpacking objects...
Checking out files...
Now that the gitub repository is cloned, we can load the data
[ ] ! ls tutorials/HiggsSchool/
jetImage_7_100p_0_10000.h5      jetImage_7_100p_50000_60000.h5
jetImage_7_100p_10000_20000.h5  jetImage_7_100p_60000_70000.h5
jetImage_7_100p_30000_40000.h5  jetImage_7_100p_70000_80000.h5
jetImage_7_100p_40000_50000.h5  jetImage_7_100p_80000_90000.h5
[ ] target = np.array([])
jetImage = np.array([])
# we cannot load all data on Colab. So we just take a few files
datafiles = ['tutorials/HiggsSchool/data/jetImage_7_100p_30000_40000.h5',
```

Set your resources to use GPUs

