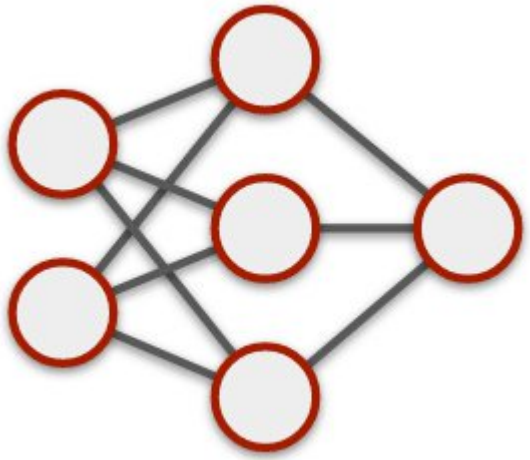


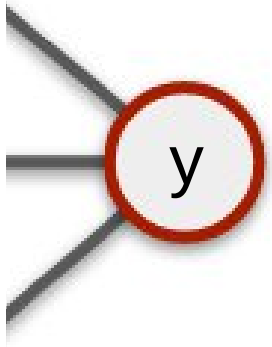
Neural Network Building Blocks (2/2)

Docent: Sebastian Wozniewski (Uni Göttingen)
Tutor: Steffen Korn (Uni Göttingen)

Deep Learning School — Basic Concepts
28.02.2023

Regression vs. Classification

Single output node



Regression of a single valued function

Range of y depends on problem, could be any range of numbers

Binary classifier

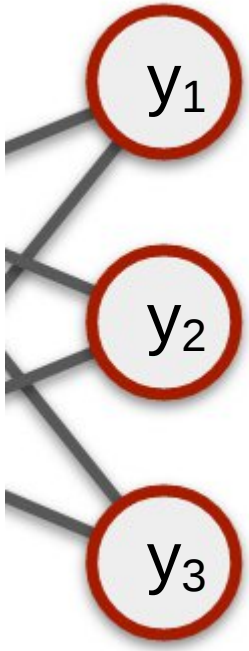
Usually y in [0, 1] or [-1, 1]

Use corresponding activation function of output node, e.g. sigmoid

Note: Classifier can be understood as regression of probability.
This is indeed imposed if using cross entropy as loss function.

Regression vs. Classification

Multiple output nodes



Regression of a vector function

Range of y_i depends on problem, could be any range of numbers

Multi-classifier

y_i in [0, 1]

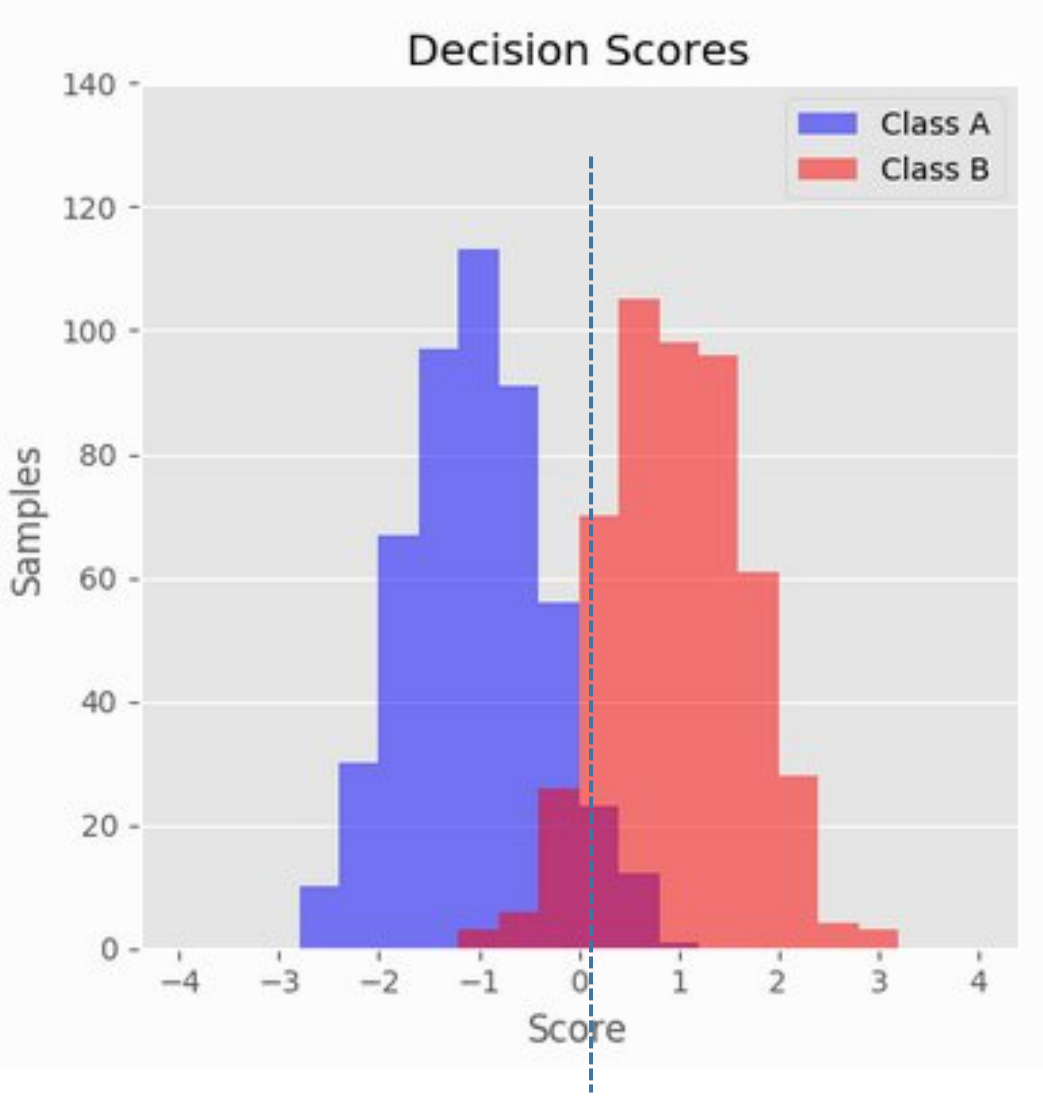
Use corresponding activation function of output node, e.g. softmax

Note: Classifier can be understood as regression of probability. This is indeed imposed if using cross entropy as loss function.

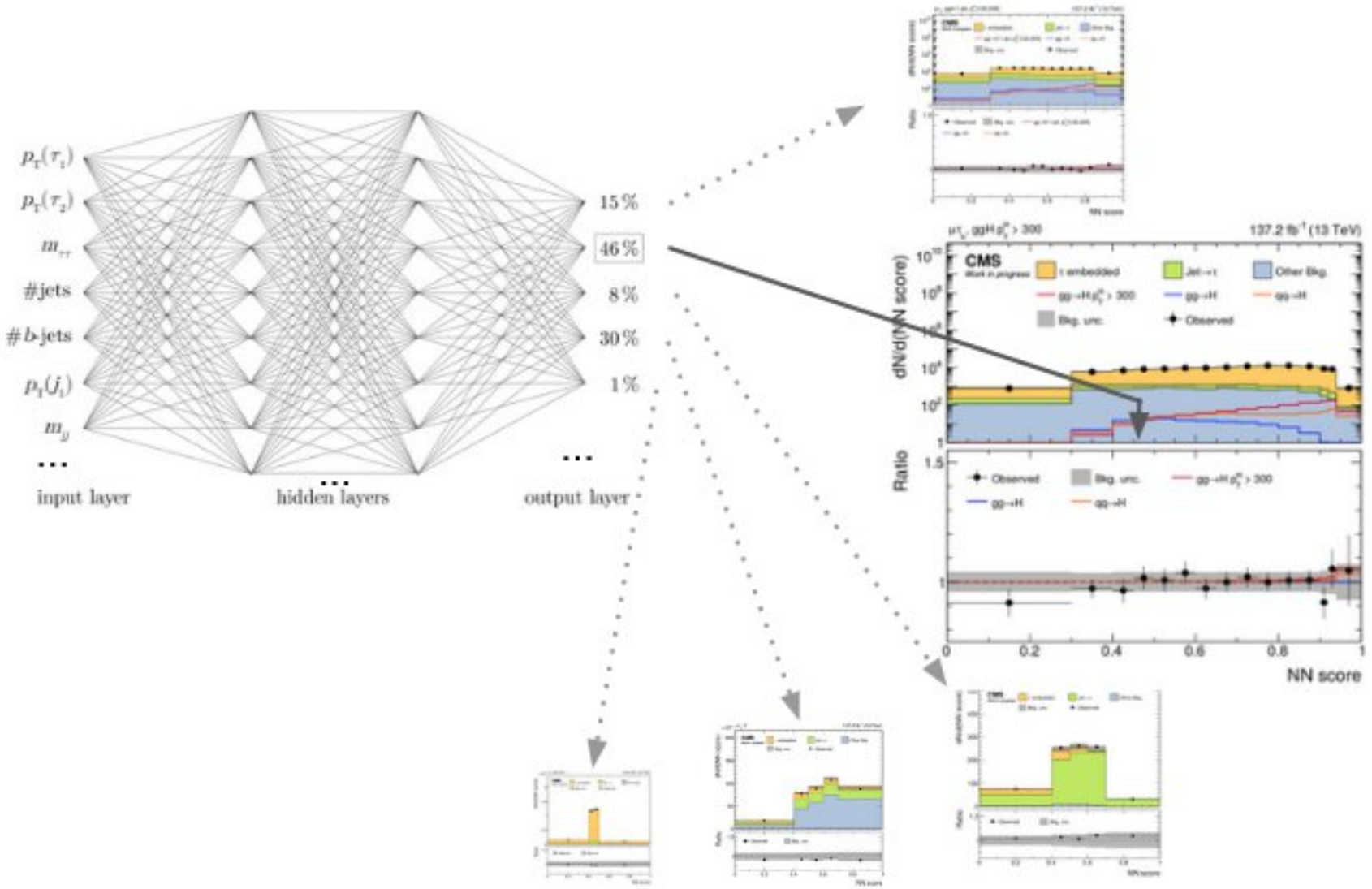
Classifier output - binary

Let's put it like this: The NN sorted the elements of the high-dimensional input space by the density ratio of both classes. Now you can easily choose a threshold for your selection!

More on metrics later on…

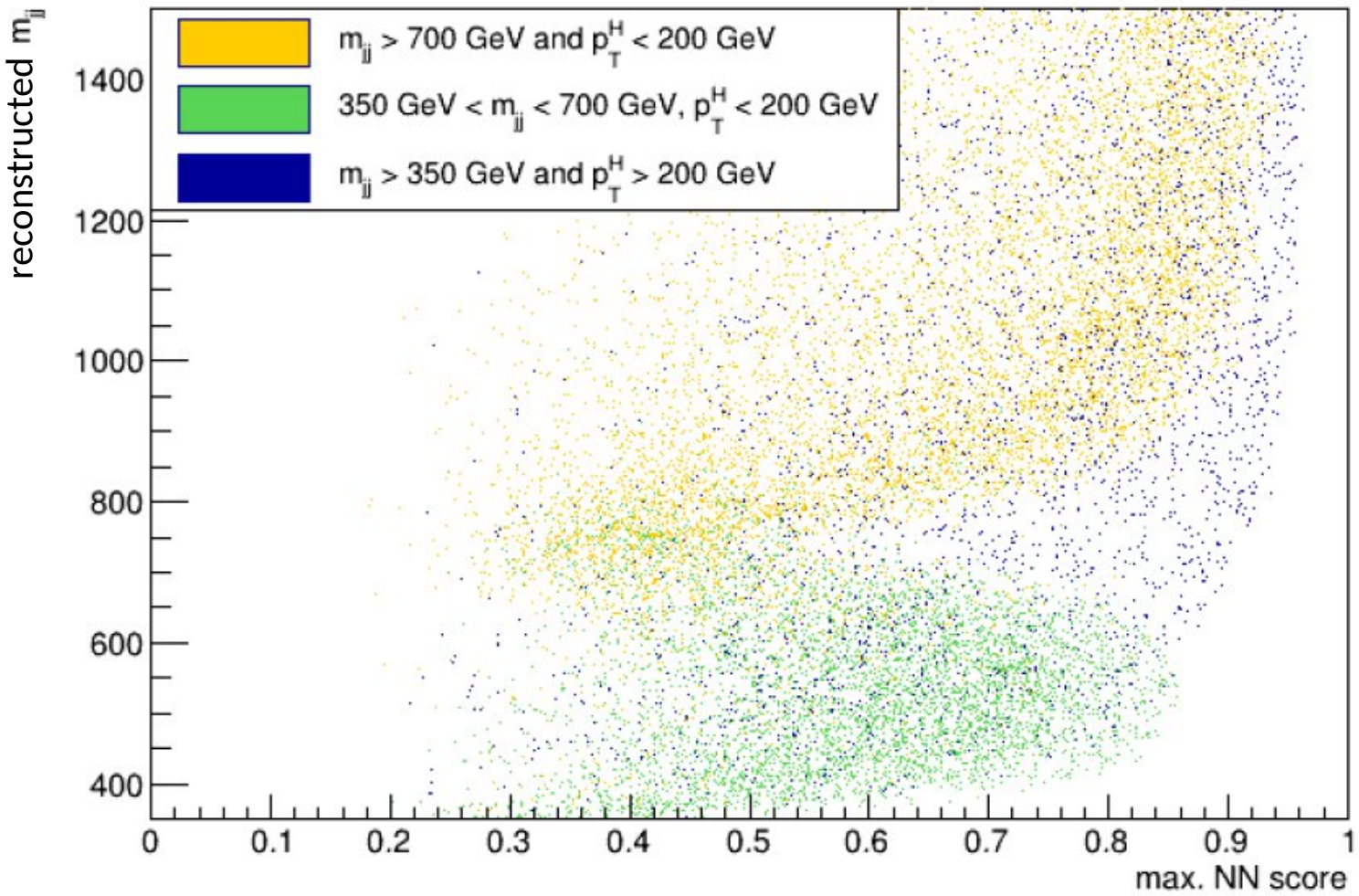


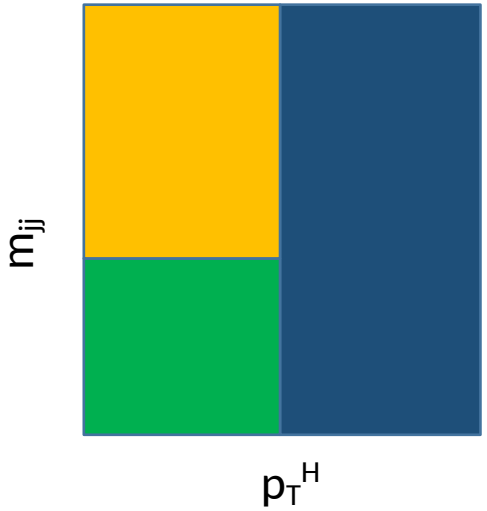
Classifier output - multi-class



Classifier output - multi-class

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737
Sebastian Wozniewski — 28.02.23





Class weights

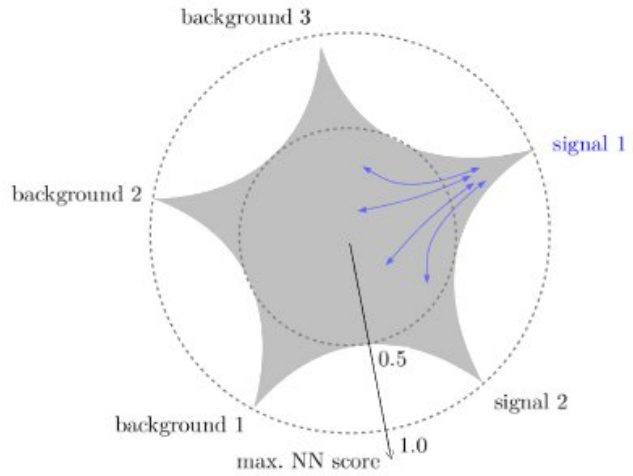
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737
Sebastian Wozniewski — 28.02.23

Are all classes in total equally likely or imbalanced (prior)?

Does the representation in your training correspond to reality?
→ might be intended or not, depends…
Data can be weighted during the training. Global weights, i.e. class weights, can be applied to adjust prior.
→ higher impact on loss function, e.g. weight of 2 is like having the same event twice.

Binary classifier: Class weights not very important. Input space elements are sorted by ratio and class weights only shift scale.
→ You can still adjust the cut.

Multi-classification: Depending on how you exploit output scores, information content may change when changing class weights.



Performance metrics

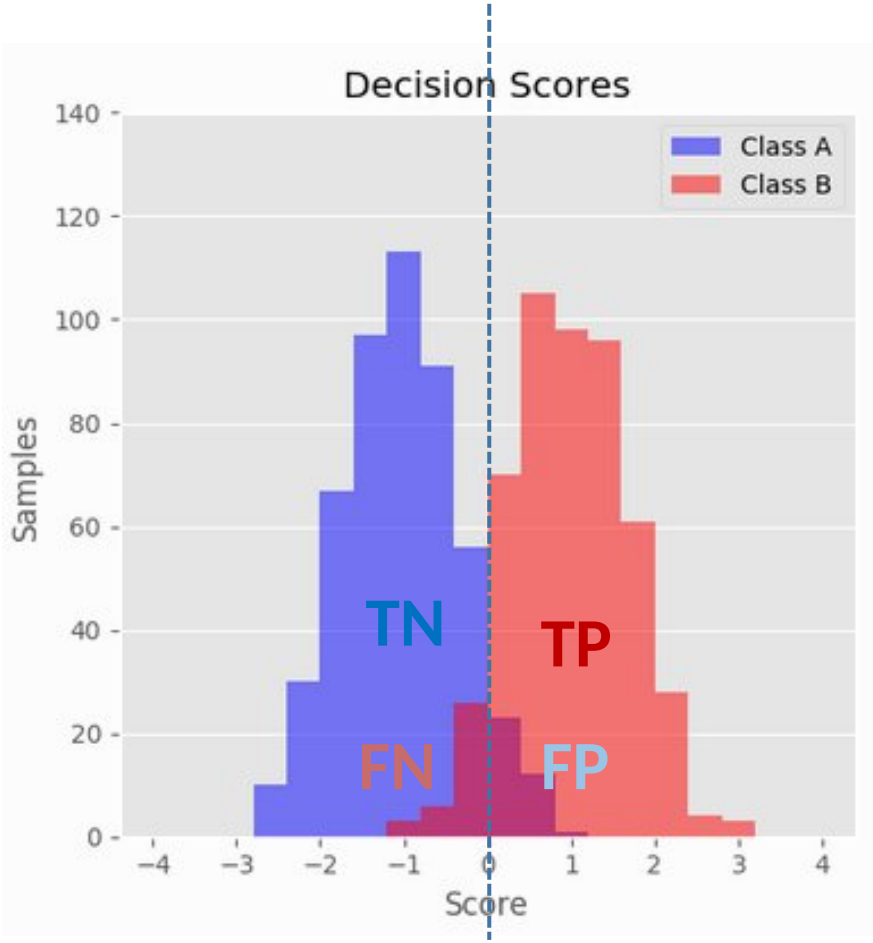
Key quantities:

- True positives (**TP**)
- False positives (**FP**)
- False negatives (**FN**)
- True negatives (**TN**)

Derived metrics:

- Precision = **TP** / (**TP** + **FP**), i.e. how pure is my predicted signal class?
- Recall or signal efficiency = **TP** / (**TP** + **FN**), i.e. what fraction of my signal is classified correctly?
- Accuracy = (**TP** + **TN**) / (**TP** + **FP** + **TN** + **FN**), i.e. what fraction of all data is classified correctly?
- …

Precision ↔ Recall
and don't mix them up!

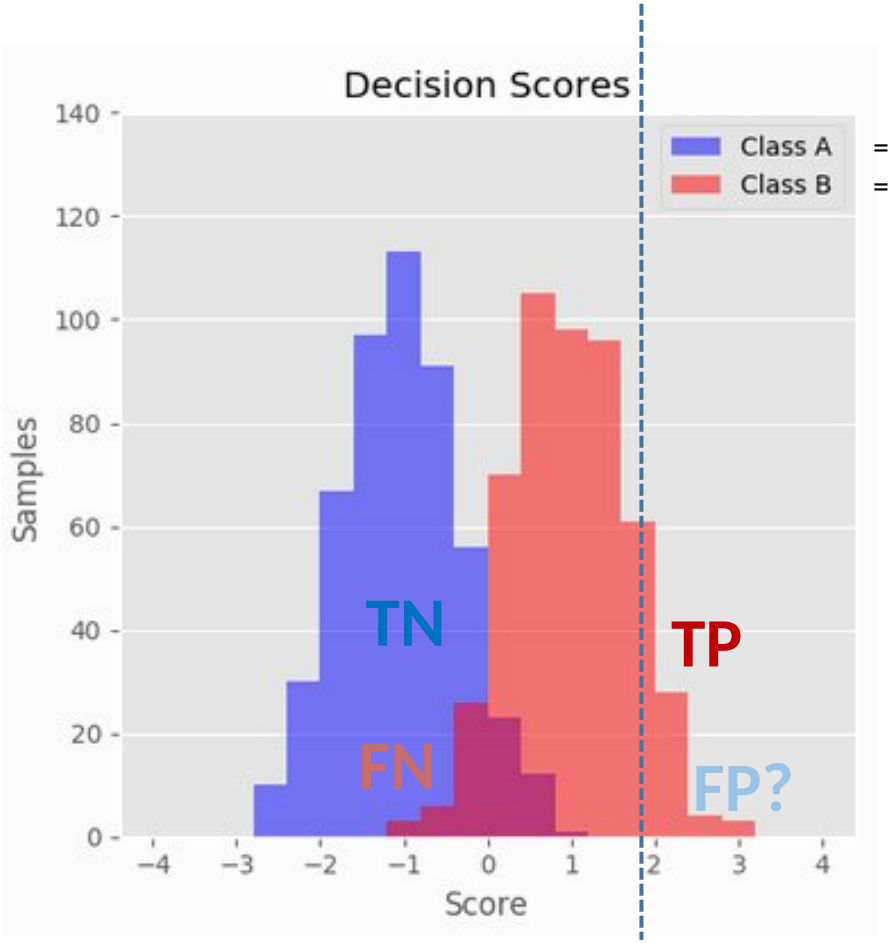


Understand your needs

What is more important precision or recall?

e.g. in
- detection of explosives at the airport
- who is admitted to visit the White House?
- forensic proof
- b-tagging (or selection of physics signal in general)

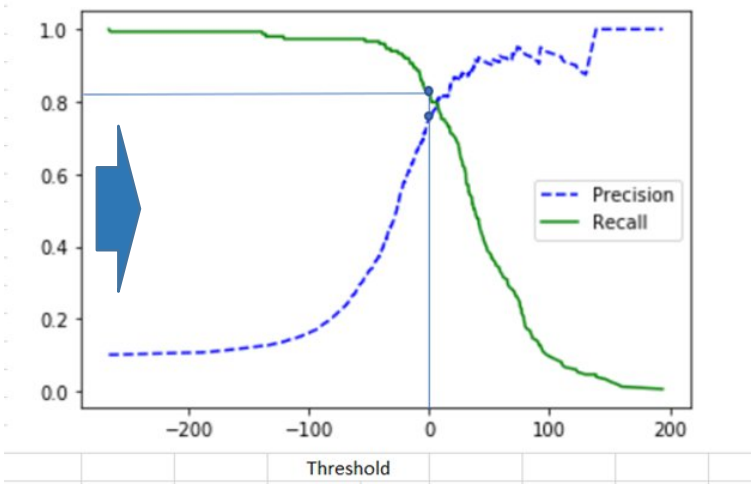
The answer may also influence your choice of class weights!

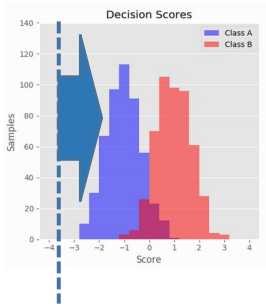


Performance visualization

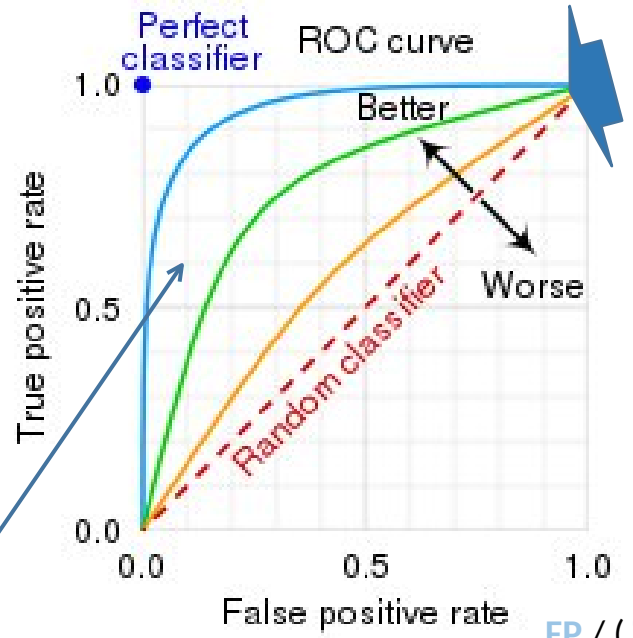
Performance metrics usually visualized for the full range of possible thresholds, e.g.:

- plot metrics versus threshold





- ROC (receiver operating characteristic) curve, i.e. plot signal and background efficiencies versus each other (threshold not explicitly shown)



TP / (TP + FN)

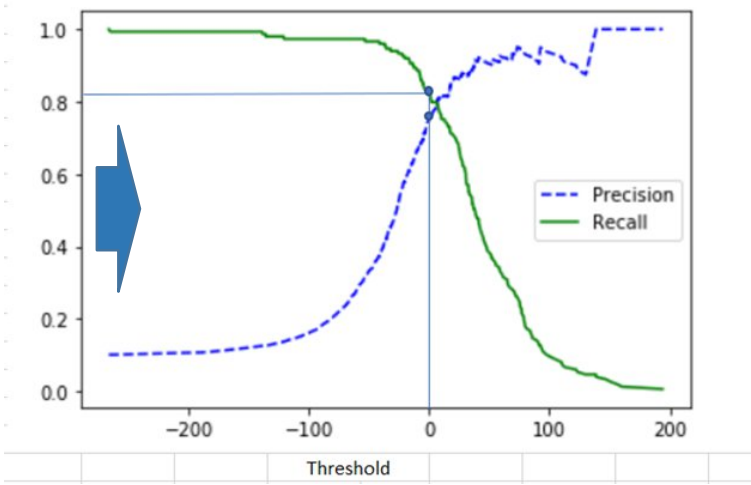
FP / (TN + FP)

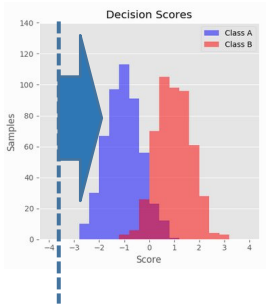
AUC (Area Under Curve) ofter quoted as performance metric

Performance visualization

Performance metrics usually visualized for the full range of possible thresholds, e.g.:

- plot metrics versus threshold

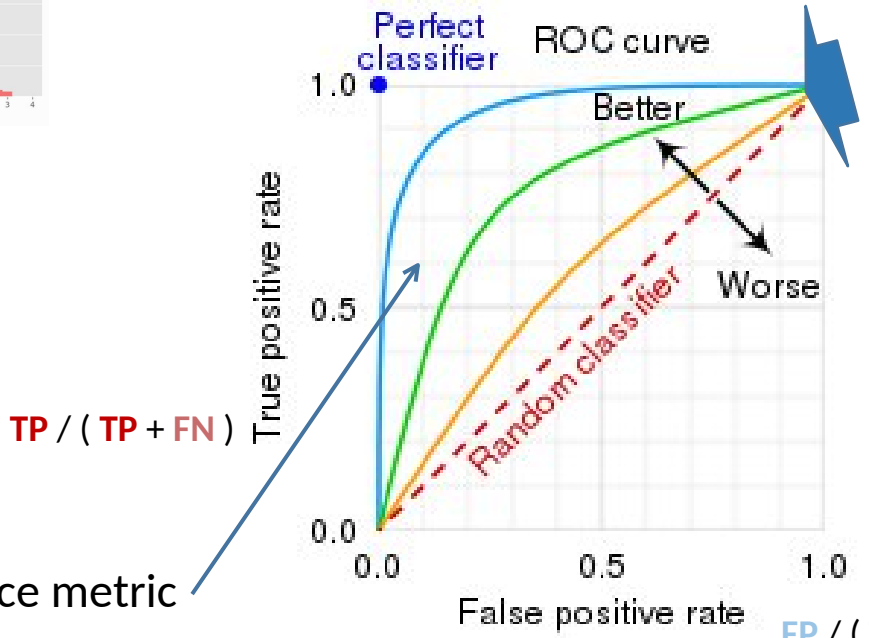




- ROC (receiver operating characteristic) curve, i.e. plot signal and background efficiencies versus each other (threshold not explicitly shown)

In some cases ROC curves of different classifiers may cross such that choice of the classifier depends on working point. But also an improved ‘universal’ classifier should be possible in that case!

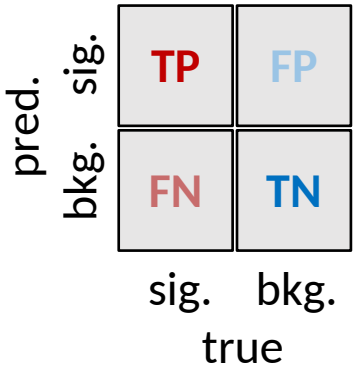
AUC (Area Under Curve) ofter quoted as performance metric



Migration matrices (Confusion matrices)

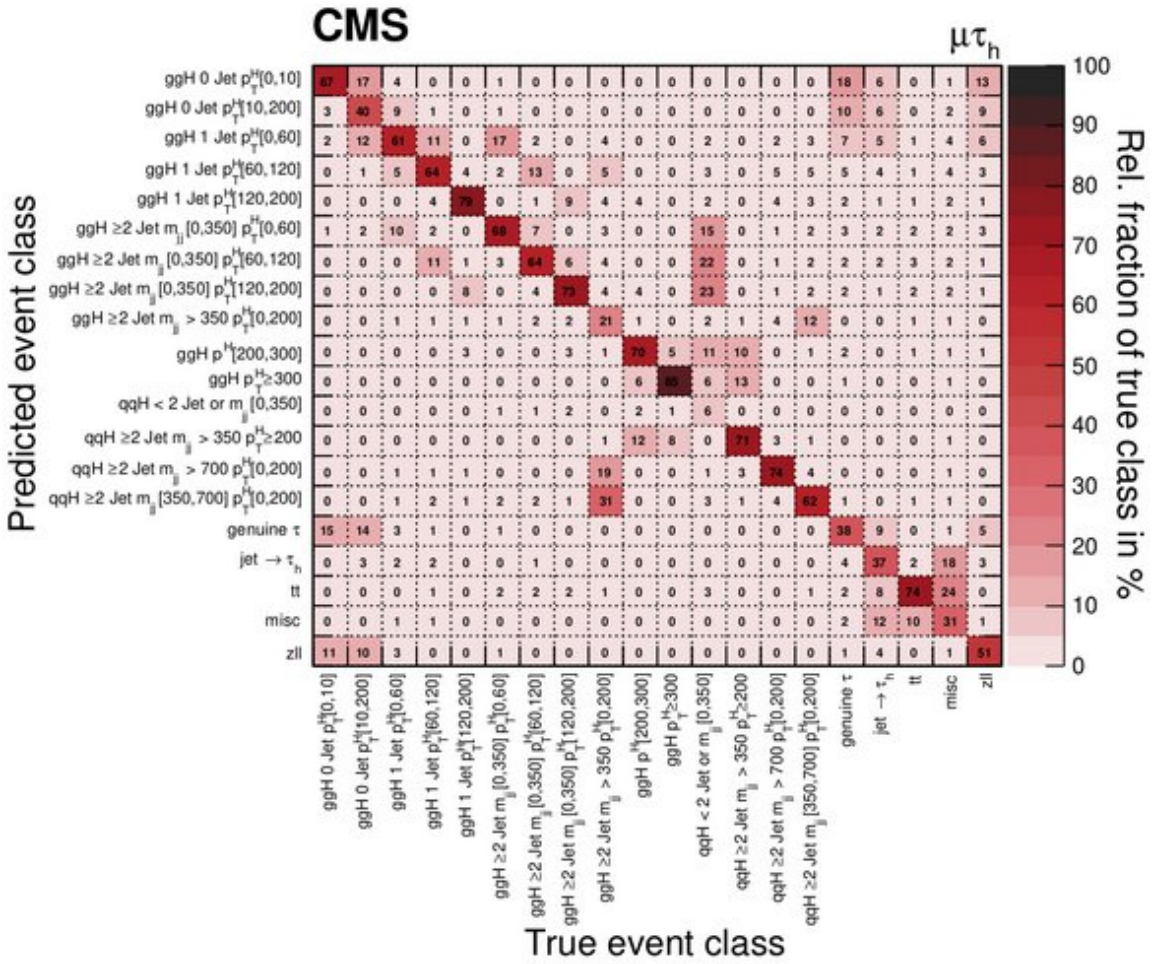
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737
Sebastian Wozniewski — 28.02.23

Shows migration from true classes to predicted classes,
e.g. binary case:



…expandable to many classes.

- can show raw event numbers,
- can be normalized to true classes (efficiency matrix),
- can be normalized to predicted classes (purity matrix)

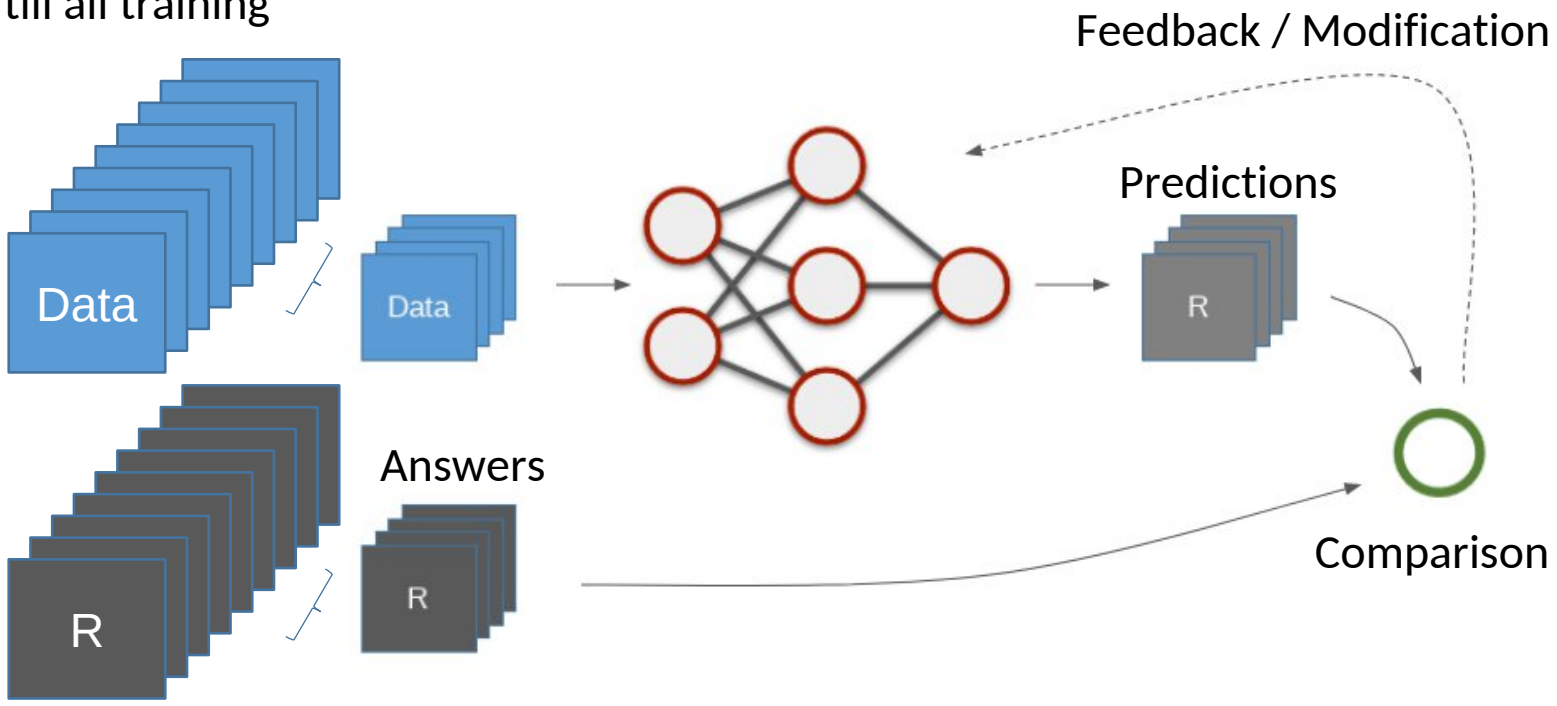


Training batches and epochs

Nowadays training data are usually too large to all fit in memory at a time, which would be needed for a perfect gradient step.
→ random subsets of the training data (**batches**) are processed in one gradient step. Repeat this till all training data has been processed once (**epoch**)

Depending on chosen method, model is updated after each batch or just after each epoch.

Strongly imbalanced classes can lead to problems: Small class may not even show up in every batch.
If you apply equalizing class weights, you could instead use data multiple times, i.e. select fixed number of data from each class per batch!



Subsets of the labeled data

Training set: The sample of labeled data used to fit the model.

Validation set: The sample used for frequent validation and optimization of hyper-parameters.

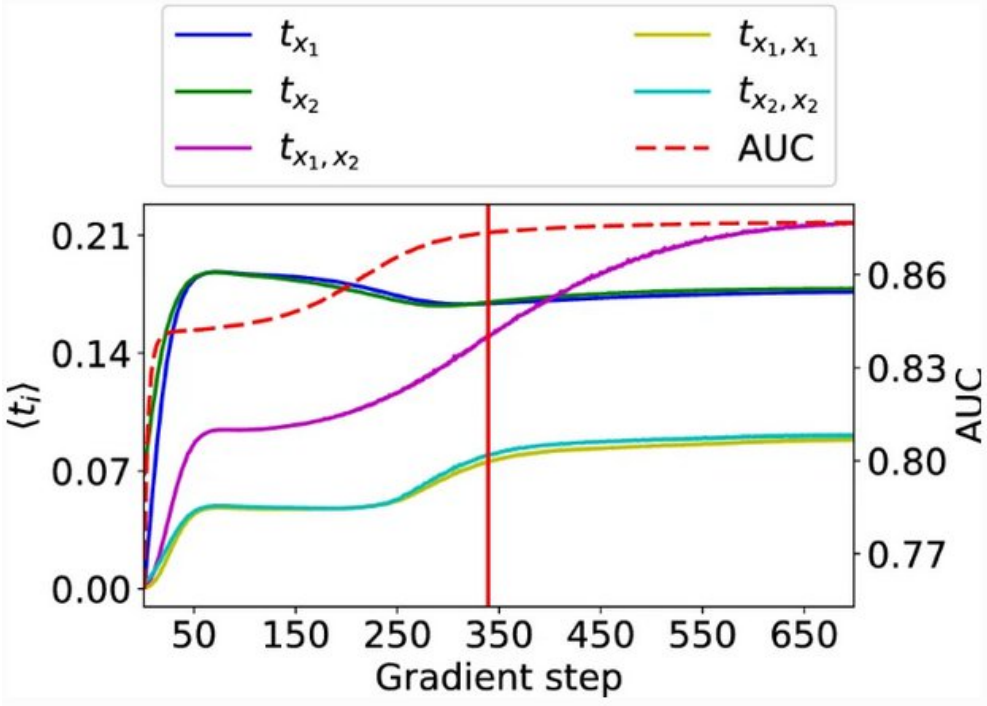
Test set: The sample used for an unbiased estimate of the performance of the final model.

Investigating what the NN learns

In the training, we calculate the **gradient wrt. the NN parameters** for the given training data.
We can turn this around and calculate the **gradient wrt. some data** at the points of the learned NN parameters.
→ **Tells us how much we depend on certain input features.** Can use higher order derivatives as well.

From Wunsch et.al., Identifying the Relevant Dependencies of the Neural Network Response on Characteristics of the Input Space. Comput Softw Big Sci 2, 5 (2018) :

Gradient analysis can be used in various ways.
In a given example, it tells us that first order features are learned quicker while higher order features take more time to be learned.



Similar example to try yourself: tensorflow_playground_example