

PROGRAMMING IN PYTHON II

Model Evaluation for Classification Tasks



Michael Widrich
Institute for Machine Learning

Copyright statement:

This material, no matter whether in printed or electronic form, may be used for personal and non-commercial educational use only. Any reproduction of this material, no matter whether as a whole or in parts, no matter whether in printed or in electronic form, requires explicit prior acceptance of the authors.

Outline

1. Motivation
2. Confusion matrix
3. True positive rate and true negative rate
4. Balanced Accuracy
5. Receiver Operating Characteristic (ROC)
6. Implementation
7. Excursion: Cross Validation

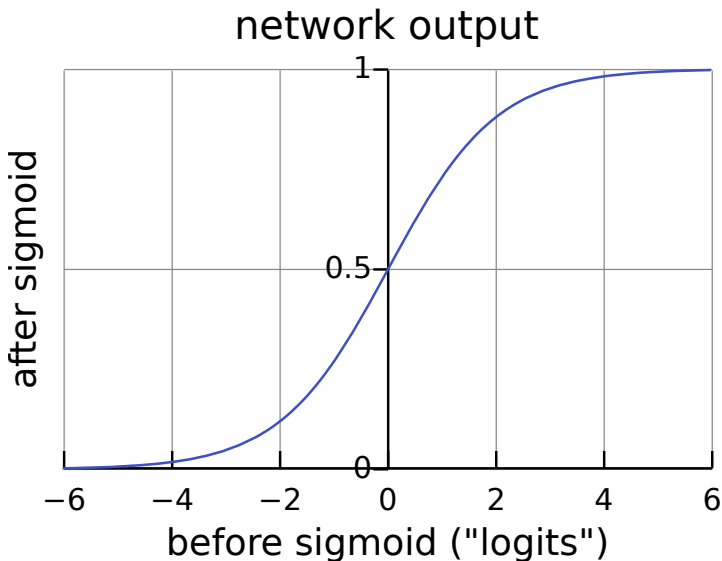
Motivation

- We have already learned about the **loss function**
 - Computes a **loss** given our model output and the target
 - The higher the loss, the farther away our output is to our target
 - We want to minimize the loss of our model during training
- In this Unit we will learn about some common methods for evaluation of (binary) classifiers
 - Typically not used for training (require complete dataset to compute scores)
 - Well suited for comparison of classification performance with different goals

Recap: Classification tasks and NNs

- Different classification tasks exist (see Unit 07)
- Binary classification task: Predict whether sample is in positive or negative class
- Typically done using a **sigmoid** activation function (aka logistic function) in the output layer
 - sigmoid function maps network output (“logit”) from range $(-\infty, \infty)$ to range $(0, 1)$
 - Activation after sigmoid can be interpreted as probability for belonging to positive class
 - Threshold to “classify” sample as positive or negative (threshold 0.5 is common):
predicted_class = positive if $\text{sigmoid}(\text{logit}) \geq 0.5$, otherwise
predicted_class = negative
 - Generalization to multiple classes via **softmax** function

Sigmoid activation function



Confusion matrix

- The **confusion matrix** is a central piece for many evaluation methods
- Assume a binary classification task with samples belonging to either positive (P) or negative (N) class
- The confusion matrix illustrates the number of samples being
 - **True positive (TP)**: correctly classified as positive (aka “hit”)
 - **True negative (TN)**: correctly classified as negative (aka “correct rejection”)
 - **False positive (FP)**: wrongly classified as positive (aka “false alarm”, “Type I error”)
 - **False negative (FN)**: wrongly classified as negative (aka “miss”, “Type II error”)

Confusion matrix for binary classification

		predicted class		
		positive	negative	total
true class	positive	TP	FN	P
	negative	FP	TN	N

True positive rate and true negative rate

■ True positive rate (TPR) (aka “sensitivity”, “recall”)

- Proportion of true positives that are correctly classified
- $TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$
- E.g. % of diseased people that are correctly classified as being diseased

■ True negative rate (TNR) (aka “specificity”)

- Proportion of true negatives that are correctly classified
- $TNR = \frac{TN}{N} = \frac{TN}{TN+FP}$
- E.g. % of healthy people that are correctly classified as not being diseased

■ False negative rate (FNR) (aka “miss rate”)

- $FNR = \frac{FN}{P} = 1 - TPR$

■ False positive rate (FPR) (aka “fall-out”)

- $FPR = \frac{FP}{N} = 1 - TNR$

Balanced Accuracy

■ Accuracy (ACC) (aka “fraction correct (FC)”)

- Proportion of correctly classified samples
- $ACC = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN}$
- Simple measure but bad with imbalanced classes

■ Balanced accuracy (BACC)

- Normalizes TP and TN by the number of positive and negative samples
- $BACC = \frac{TPR+TNR}{2}$
- Suited for imbalanced classes

Receiver Operating Characteristic (ROC)

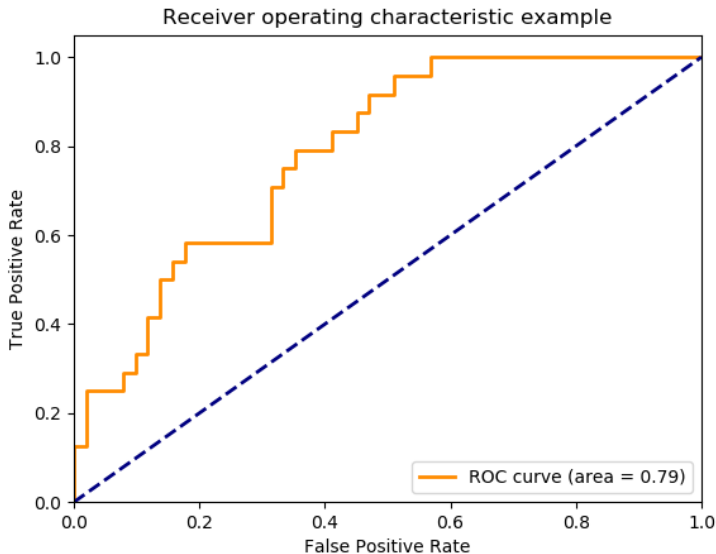
■ Receiver operating characteristic (ROC) curve

- Shows TPR and FPR while varying classification threshold
- Classification performance w.r.t. FN and FP costs

■ Area under the receiver operating characteristic curve (AUC)

- Reduces ROC curve to single value (the area under the curve)
- Probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- Ranges from 0 to 1, with 1 being perfect and 0.5 being random
- Suitable for imbalanced classes

Receiver Operating Characteristic (ROC)



[Source: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html]

Implementation

- Many more evaluation methods exist, some of which with generalizations to tasks with multiple classes
- The `scikit-learn` (`sklearn.metrics`) module provides a vast selection of evaluation methods

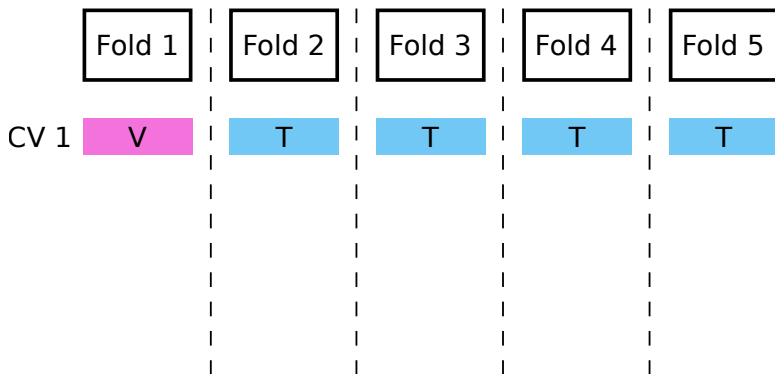
`https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics`

- For examples, see the code part of this Unit

Excursion: Cross Validation

- To get an estimate for the model performance on future data, we have to use the test set to compute the scores
- For small datasets, the requirement that training and test set must not overlap is painful
- Solution: **Cross Validation (CV)**
 - Split dataset into n disjoint folds
 - Use $n - 1$ folds as training set, left-out fold as test set
 - Train n times, every time leaving out a different fold as test set
 - Average over n estimated risks on test sets to get better estimate of generalization capability

Cross Validation (1)



5-fold Cross Validation
T: Training set; V: Test set

Cross Validation (1)

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
CV 1	V	T	T	T	T
CV 2	T	V	T	T	T

5-fold Cross Validation
T: Training set; V: Test set

Cross Validation (1)

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
CV 1	V	T	T	T	T
CV 2	T	V	T	T	T
CV 3	T	T	V	T	T
CV 4	T	T	T	V	T
CV 5	T	T	T	T	V

5-fold Cross Validation
T: Training set; V: Test set

Cross Validation (2)

- Nested Cross Validation

- We can apply another (inner) CV procedure within each training-set of the original (outer) CV

- allows for evaluation of model selection procedure

- Getting a risk estimate on selected model:

1. Apply cross validation on training set (withhold test set)
2. Use test set to estimate risk for the model selected via CV

- In practice, the found model is often trained further or re-trained on complete dataset for best performance