PROGRAMMING IN PYTHON II

Model Evaluation

Michael Widrich
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

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
8. Additional Measures: Binary Classifiers
9. Additional Measures: Regression
10. Additional Evaluation Strategies

MOTIVATION

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

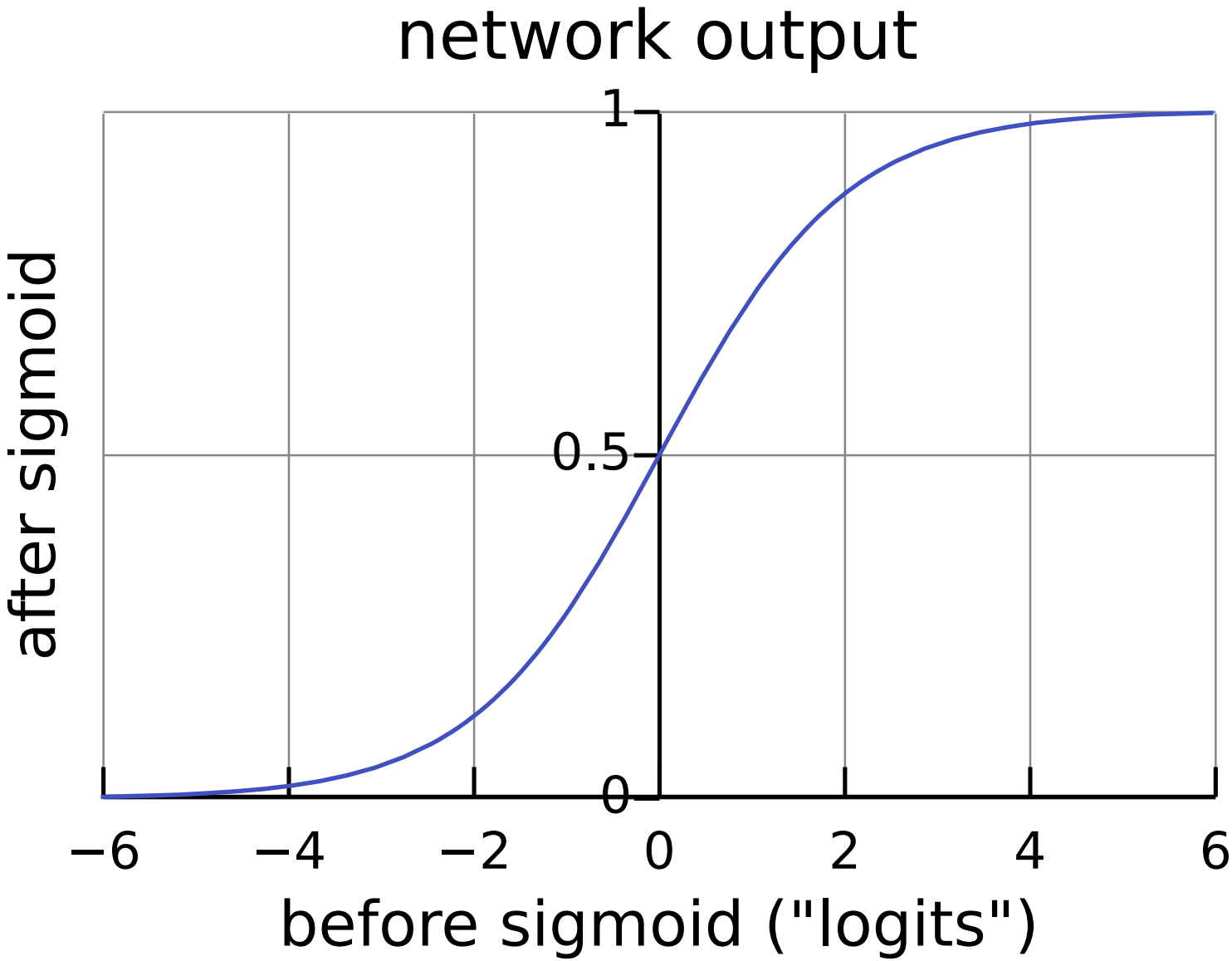
Motivation

- 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 our models
  - Typically not used for training (bad gradients or complete dataset required 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 sigmoid(logit) ≥ 0.5, otherwise predicted_class = negative
  - Generalization to multiple classes via softmax function

Sigmoid activation function

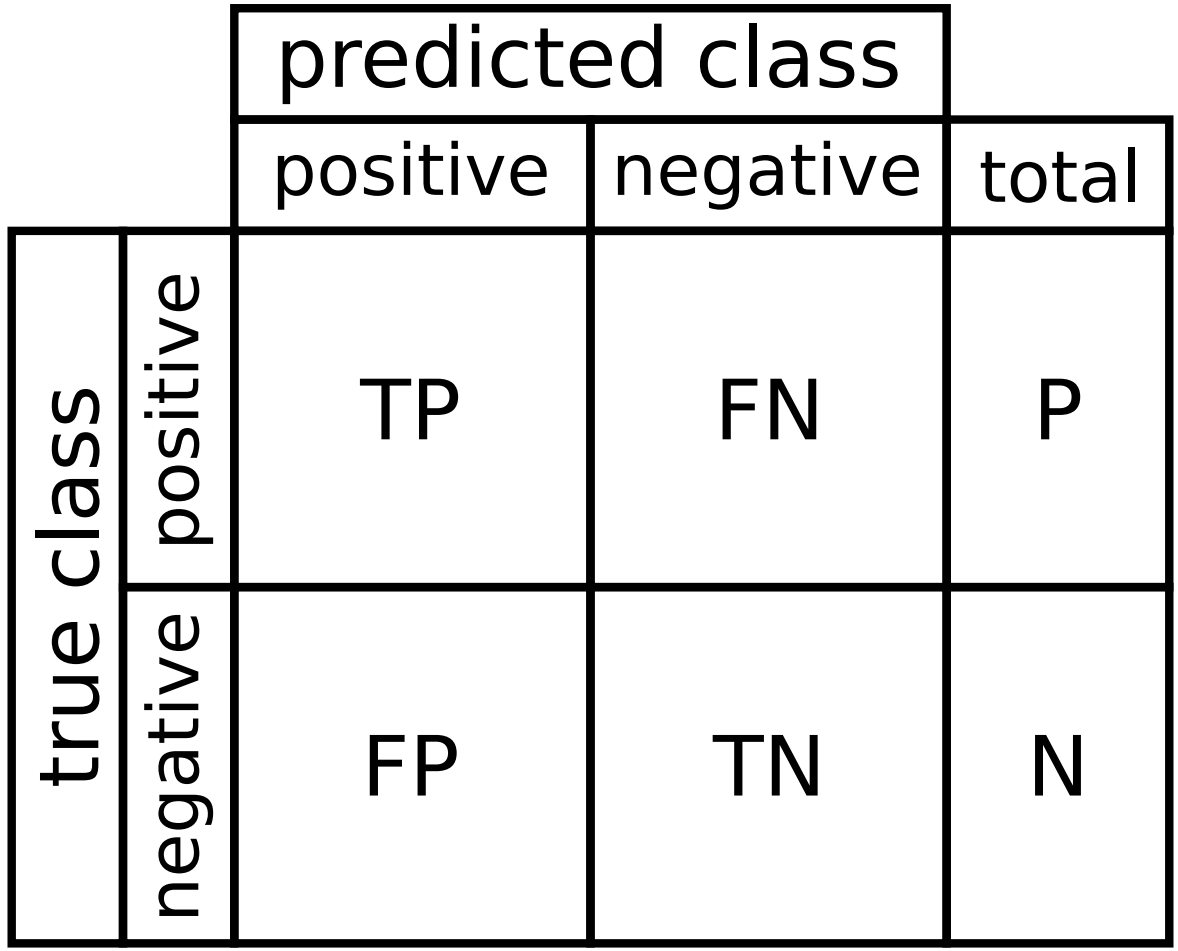


CONFUSION MATRIX

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



TRUE POSITIVE RATE AND TRUE NEGATIVE RATE

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

Balanced Accuracy

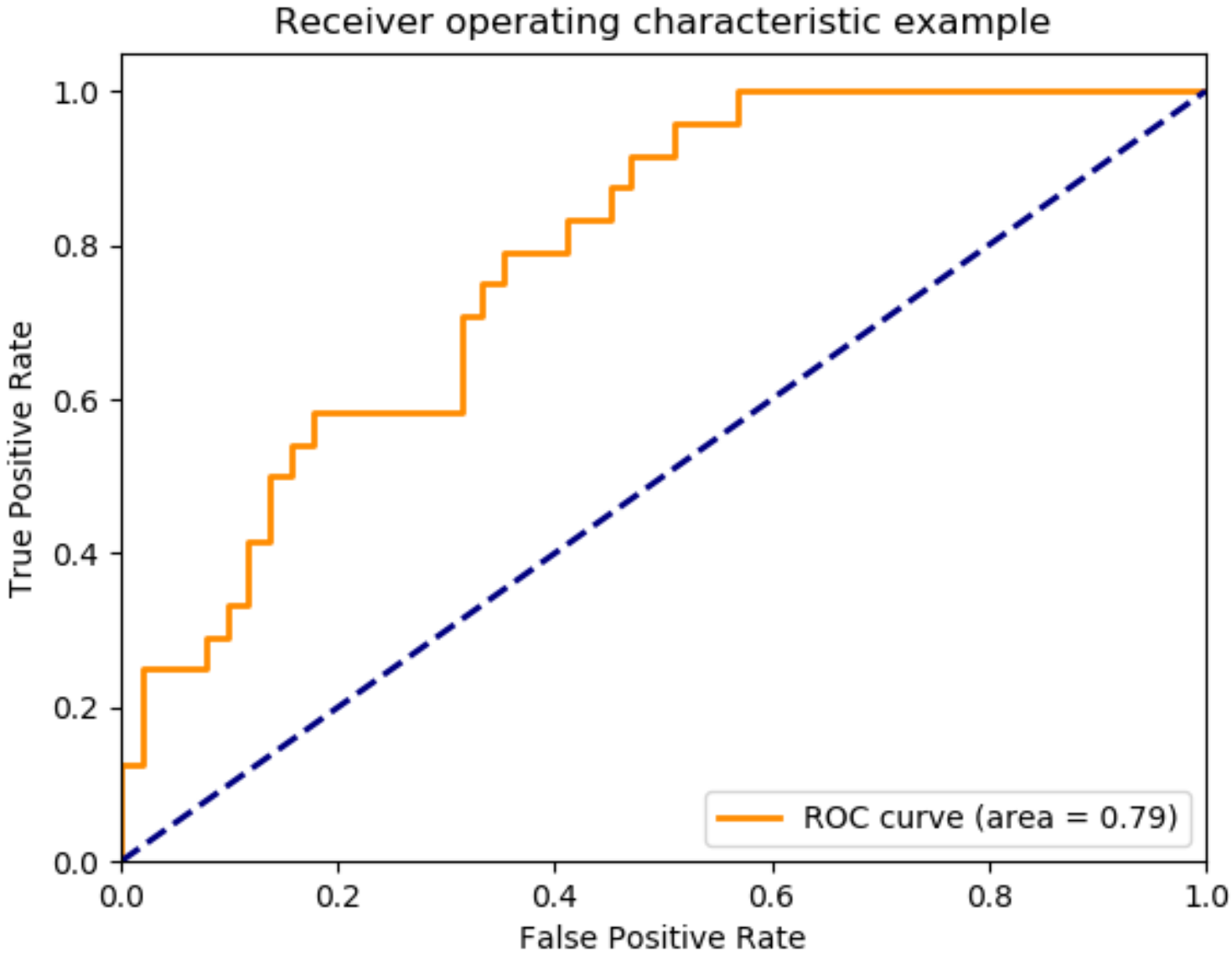
- Accuracy (ACC) (aka “fraction correct (FC)”)
  - Proportion of correctly classified samples
  - $ACC = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN}$
  - Simple measure but bad with imbalanced classes
- Balanced accuracy (BACC)
  - Normalizes TP and TN by the number of positive and negative samples
  - $BACC = \frac{TPR+TNR}{2}$
  - Suited for imbalanced classes

RECEIVER OPERATING CHARACTERISTIC (ROC)

Receiver Operating Characteristic (ROC)

- 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

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

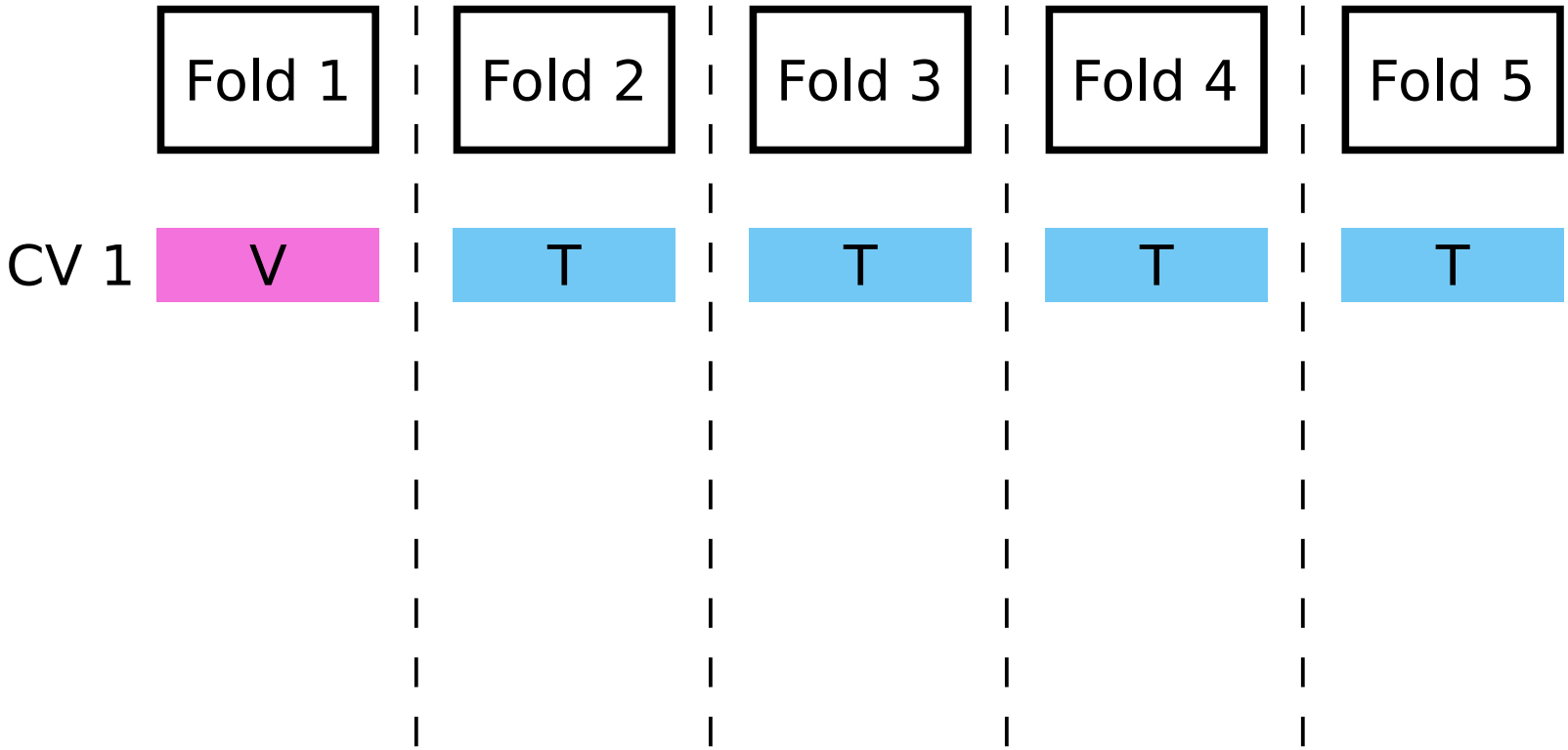
  `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics`
- For examples, see the code part of this Unit

EXCURSION: CROSS VALIDATION

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

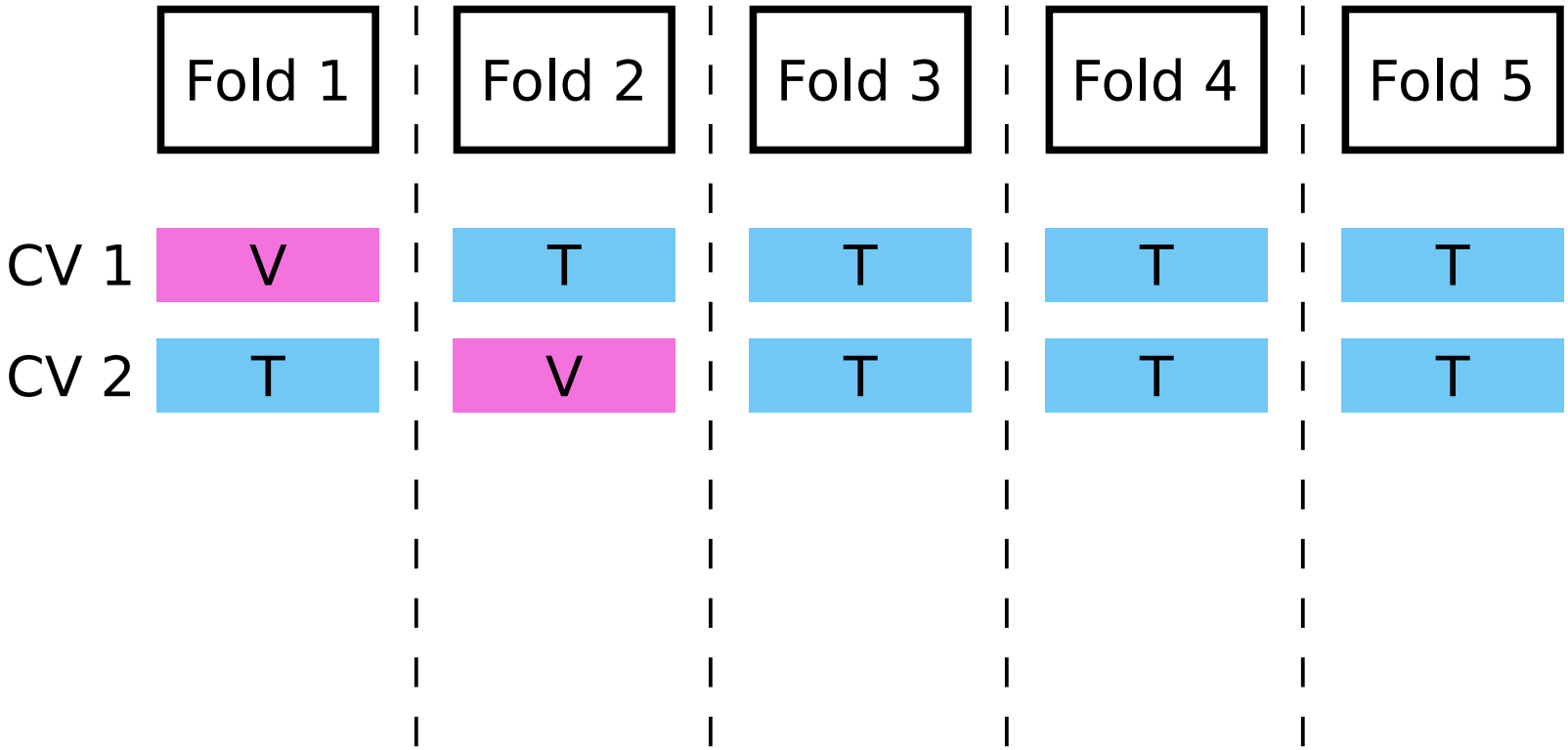
Cross Validation (1)



5-fold Cross Validation

T: Training set; V: Test set

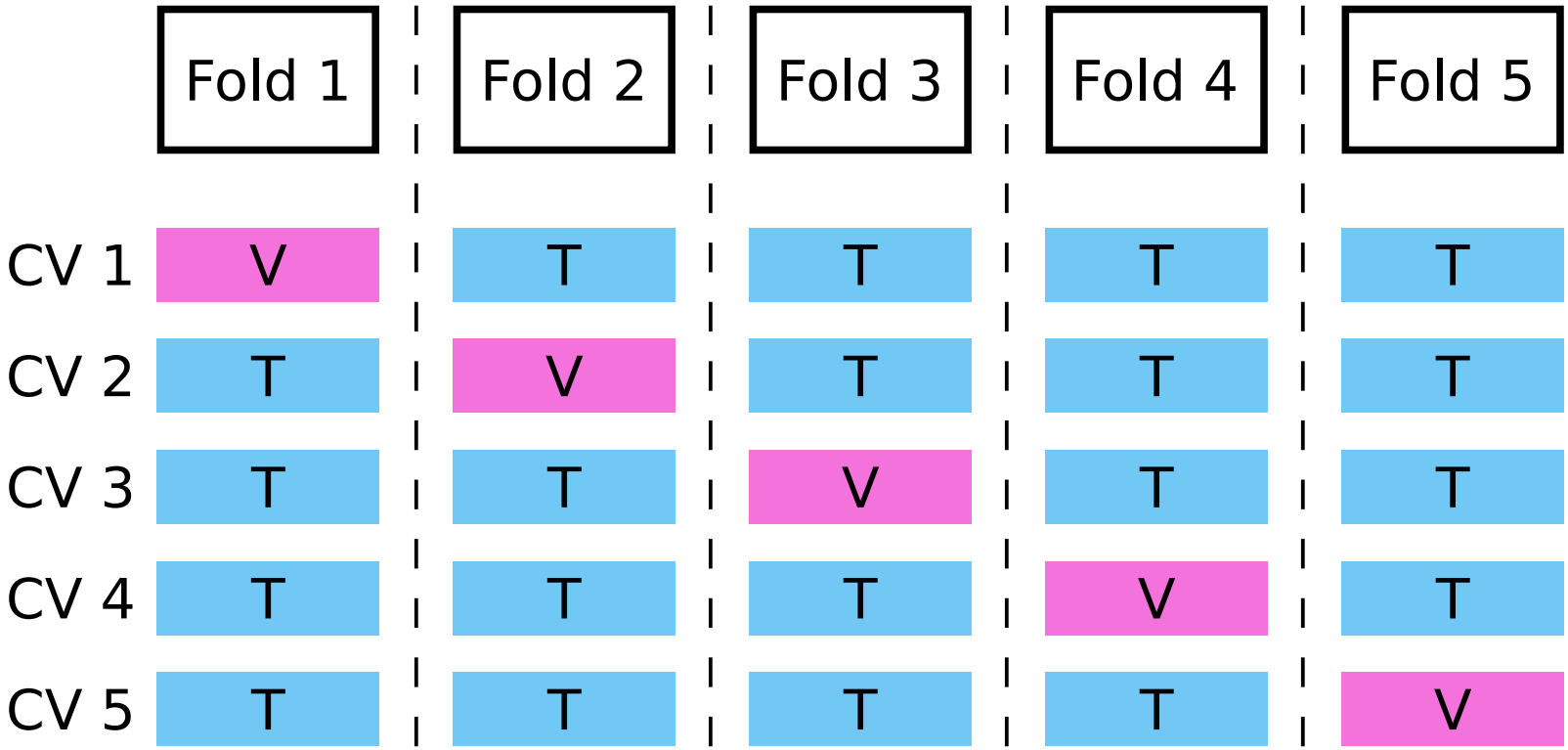
Cross Validation (1)



5-fold Cross Validation

T: Training set; V: Test set

Cross Validation (1)



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

  $\rightarrow$ allows for evaluation of model selection procedure

Cross Validation (2)

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

  $\rightarrow$ 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

Cross Validation (2)

- Nested Cross Validation
  - We can apply another (inner) CV procedure within each training-set of the original (outer) CV
  - $\rightarrow$ 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

ADDITIONAL MEASURES: BINARY CLASSIFIERS

F-Score

- F-score (F_1-score, F-measure)
  - F-score = $2 * \frac{Precision * Recall}{Precision + Recall}$
  - Used for binary classification models
  - Defined as the harmonic mean of Precision and Recall (TPR, Sensitivity) of a model

- F_β-score (generalized version of F-score)
  - F_β-score = $(1 + \beta^2) * \frac{Precision * Recall}{\beta^2 * Precision + Recall}$
  - Used for binary classification models
  - β indicates how much Recall is more important than Precision

Accuracy for multi-class classification

Accuracy (binary classes) (ACC)	**Balanced Accuracy (binary classes)** (BACC)
$\frac{TP+TN}{P+N} = \frac{total\ hit\ count}{total\ samples}$	$\frac{1}{2}(TPR + TNP)$
	It's suitable for imbalanced classes

Accuracy (multi-classes) (ACC)	**Balanced Accuracy (multi-classes)** (BACC)
$\frac{\sum TP_i}{\sum P_i} = \frac{total\ hit\ count}{total\ test}$	$\frac{1}{n}\sum \frac{TP_i}{P_i}$
	It's suitable for imbalanced classes

- TP_i and P_i are respectively the numbers of true positive and actual positive examples for class i
- n is the number of classes
- Not suitable for multi-label classification

ADDITIONAL MEASURES: REGRESSION

Measures for Regression Models

- Mean Absolute Error (MAE): $\frac{\sum_{i=1}^{n} |y_i - y'_i|}{n}$
- Mean Squared Error (MSE): $\frac{\sum_{i=1}^{n} (y_i - y'_i)^2}{n}$ $\rightarrow$this measure exaggerates the presence of outliers
- Root Mean Squared Error (RMSE): Square root of MSE
- Relative Absolute Error: $\frac{\sum_{i=1}^{n} |y_i - y'_i|}{\sum_{i=1}^{n} |y_i - \overline{y}_i|}$
- Relative Squared Error: $\frac{\sum_{i=1}^{n} (y_i - y'_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2}$
- Root Relative Squared Error: Squared root of Relative Squared Error

y_i and y'_i are respectively the true and predicted value of the depending variable y in the test set; $\overline{y}_i$ is the mean value of y_i in the training set.

ADDITIONAL EVALUATION STRATEGIES

Hold-out test set

- Hold-out test set
  - Randomly partitioned a data set into a training set, (validation set,) and a test set
    - e.g. 2/3 of the data for training set, remaining 1/3 for test set
  - Training set used to train a model, test set used for performance estimate
  - Performance estimate not reliable if test set is small
  - Used for:
    - Large datasets where CV isn't feasible
    - Challenges (often with fine-tuning on full dataset for more training samples)

Bootstrapping

- Bootstrapping methods
  - A class of evaluation methods in which
    - training set is formed by randomly picking samples from the data set
    - samples in the data set may be selected more than once
  - Commonly use bootstrapping method: .632 bootstrap
    - Training set is formed by n-times randomly picking a sample from data set of n examples
    - Samples which were not picked for training set form the test set
    - If n is large, 63.2% of samples in data set form the training set and the remaining 36.8% form the test set*

*) The probability an example selected from the given data set is $\frac{1}{n}$ so the probability of not being selected is $(1 - \frac{1}{n})$. n times of selections are done so $(1 - \frac{1}{n})^n$ is the probability that an example will not be selected. If n is large, this probability approximates $\lim_{n\to\infty}(1 - \frac{1}{n})^n = e^{-1} = 0.368$

Bagging (Bootstrap Aggregation)

- Bagging (bootstrap aggregation)
  - Build k models from data set
  - Each model is learned from a training set which is formed by a bootstrapping
  - The composite (ensemble) model works based on
    - Majority votes on predicted classes of the member models in case of classification
    - The average value of predicted values of member models in case of regression
  - Properties
    - Greater predictive performance since using more training samples
    - More robust to the effects of noisy data
    - Composite model reduces variance of the individual classifiers