Lecture 6: Assessment of machine (supervised) learning systems

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

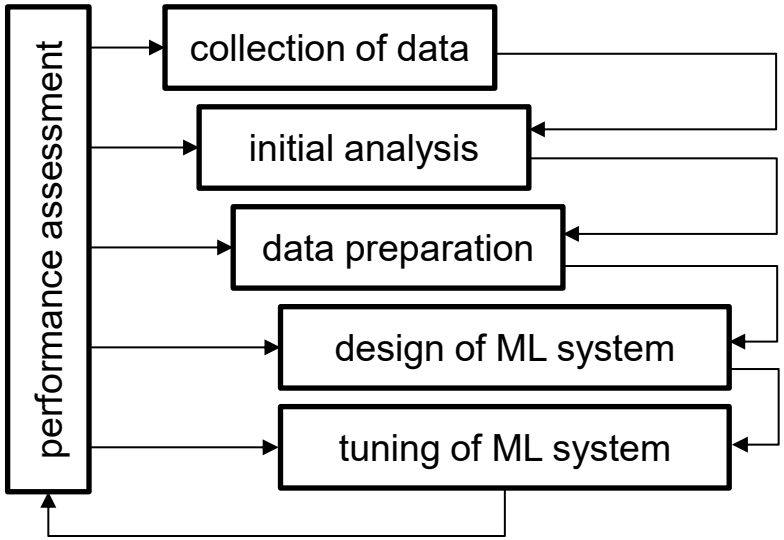
Alberto ORTIZ RODRÍGUEZ

Contents

Introduction

- The assessment of ML systems involves several aspects of its performance and may take the designer back to any of the developing stages
- Regarding the model itself, we may evaluate
  - its **adequacy**
    - is the model adequate? is it biased? which is the error rate?
  - its **generalization**
    - how well behaves the model with unseen data

underfitting vs overfitting

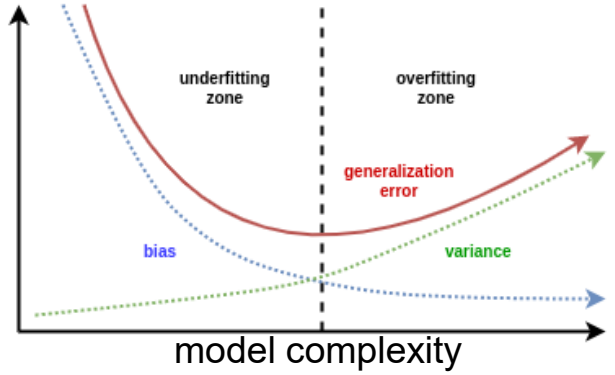


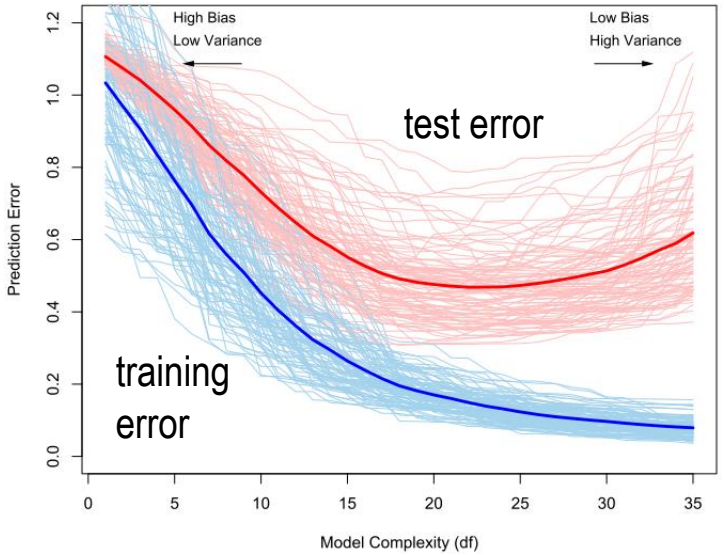
- ML systems evaluation tools are useful not only for assessing the performance of the system, but also
  - for **debugging** purposes, to figure out what is not working and make the necessary adjustments
  - to **compare among different ML models**
- The basic assessment protocol splits the dataset into:
  - **Training set**. Used to build the classifier/regressor.
  - **Test set**. To check the behaviour of the classifier with unseen examples.

Contents

Bias-variance tradeoff

- The source of prediction errors can be shown to be three-fold:
  - **Bias**
    - It is due to wrong assumptions, either by the designer or by the model
    - A high-bias model is most likely to **underfit** the training data
  - **Variance**
    - It is due to the model's **excessive sensitivity** to small variations in the training data
    - A model with many degrees of freedom, e.g. a high-degree polynomial, is likely to have high variance and thus to **overfit** the training data
  - **Irreducible error**
    - It is due to the **noisiness** of the data itself
    - the only way to reduce this error is to clean up the data, i.e. fix a broken sensor
- **Increasing a model's complexity** typically increases its variance and reduces its bias
- **Reducing a model's complexity** usually increases its bias and reduces its variance
- This is why it is called the **bias-variance tradeoff** (also BV dilemma)





Contents

Confusion matrix

- Many **performance metrics** can be calculated from the **confusion matrix**, e.g. for **two** classes, having defined first which is the positive class

		predicted class: **positive**	predicted class: **negative**	
true class	**positive**	TP	FN	$\text{Po} = TP + FN$
	negative	FP	TN	$\text{Ne} = FP + TN$
		$\widehat{\text{Po}} = TP + FP$	$\widehat{\text{Ne}} = FN + TN$	

accuracy (A)	$\frac{TP+TN}{TP+TN+FP+FN}$	**error rate** (E)	$1 - \text{A} = \frac{FP+FN}{TP+TN+FP+FN}$
false positive rate (FPR)	$\frac{FP}{\text{Ne}} = \frac{FP}{FP+TN}$	**true positive rate** (TPR)	$\frac{TP}{\text{Po}} = \frac{TP}{TP+FN}$
precision (P)	$\frac{TP}{\widehat{\text{Po}}} = \frac{TP}{TP+FP}$	**recall** (R)	$\frac{TP}{\text{Po}} = \frac{TP}{TP+FN} = TPR$
specificity	$\frac{TN}{\text{Ne}} = 1 - FPR$	F_1**-score**	$\frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R}$

FPR is also known as **false alarm rate**

TPR is also knwon as **sensitivity**

- **F_1 -score** combines in a single metric P and R, by means of their **harmonic mean**

$$\frac{2}{1/P + 1/R} = \frac{2PR}{P + R}$$

  - the regular mean treats all values equally, the harmonic mean penalizes more low values
  - a high F_1 score (which is better) results only if both P and R are high
  - also known as Sorensen-Dice coefficient or **Dice Similarity Coefficient** (DSC)

- F_1 is a particular case of the **F - score** or **F_β - score**:

$$F_\beta = (1+\beta^2)\frac{P\,R}{\beta^2 P + R} = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP}$$

$$F_1 = \frac{2TP}{2TP + FN + FP}, \quad F_2 = \frac{5TP}{5TP + 4FN + FP}, \quad F_{0.5} = \frac{1.25\,TP}{1.25\,TP + 0.25FN + FP}$$

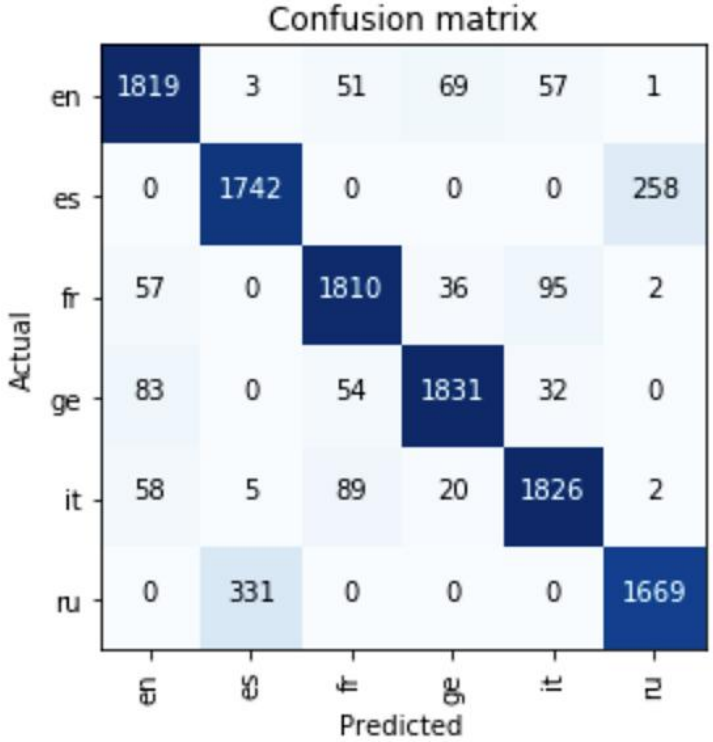
  - $\beta = 1$, F_1 -score, which weighs equally R and P: same emphasis on FN and FP
  - $\beta = 2$, **F_2 -score**, which weighs R lower than P: effect of FP is less noticeable
  - $\beta = 0.5$, **$F_{0.5}$ -score**, which weighs R higher than P: effect of FN is less noticeable

$$P = \frac{TP}{\widehat{\text{Po}}} = \frac{TP}{TP + FP}$$

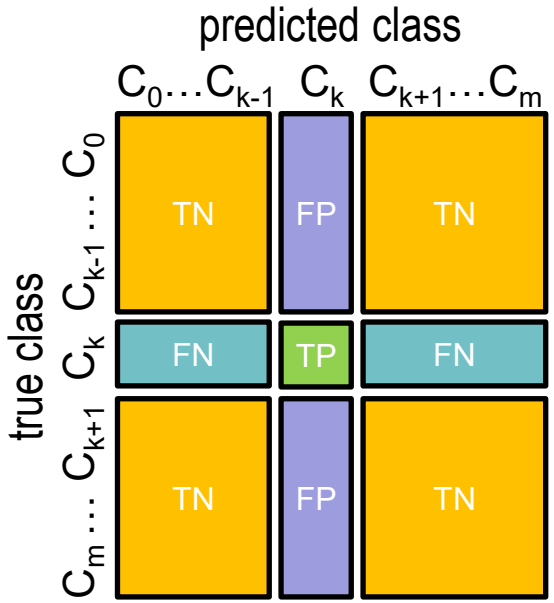
$$R = \frac{TP}{\text{Po}} = \frac{TP}{TP + FN}$$

Confusion matrix

- The confusion matrix can be generalized for **M-class problems**



TP, TN, FP, FN are calculated according to **one versus all (OvA) classification**



- Global metrics can be calculated following two approaches: $P_k = \frac{TP_k}{TP_k + FP_k}$, etc.
  - **macro-averages**: average scores for each class
    - each class is weighed equally
  - **micro-averages**: from individual TP, TN, FP, FN
    - each prediction is weighed equally

e.g. $P_{macro} = \frac{P_1 + \cdots + P_m}{m}$

e.g. $P_{micro} = \frac{TP_1 + \cdots + TP_m}{TP_1 + \cdots + TP_m + FP_1 + \cdots + FP_m}$

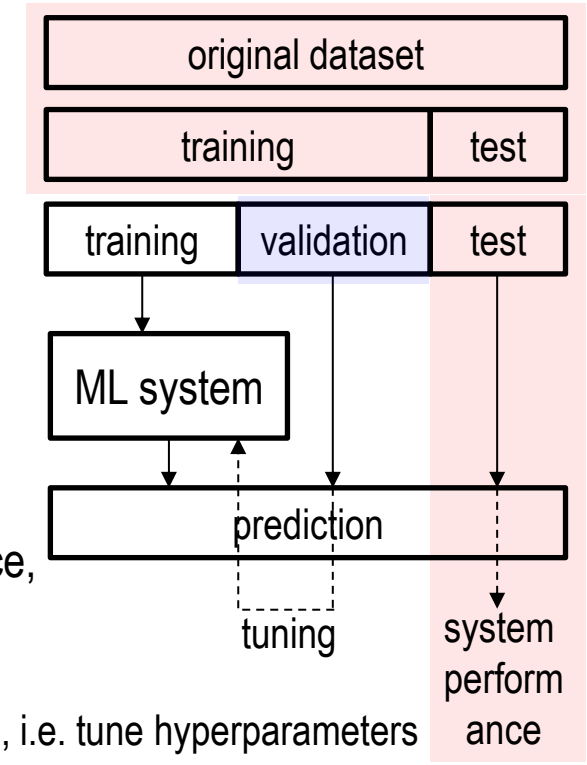
Contents

Cross-validation techniques

- From a **methodological** point of view, we need a way to obtain robust and bias-free performance measurements
- **Cross-validation** techniques can provide performance estimation values with low bias:
  - holdout cross validation
  - n-fold cross validation
  - stratified n-fold cross validation
  - leave-one-out cross validation (LOOCV)
  - nested cross validation

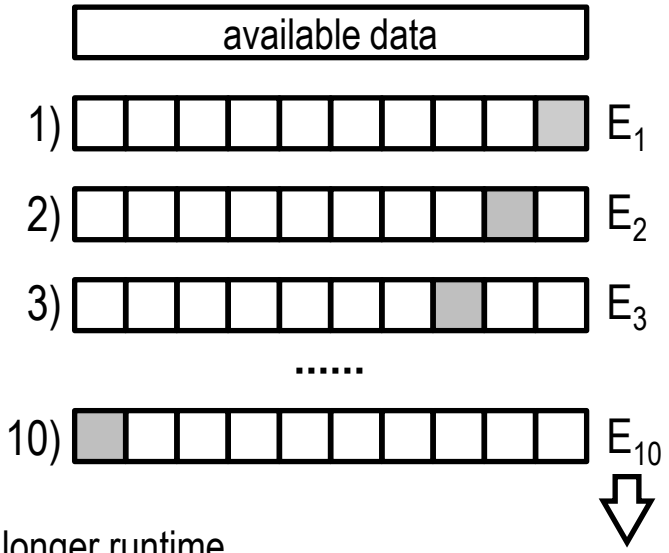
Cross-validation techniques

- **Holdout cross validation**
  - Simplest kind of cross validation
  - The data set is initially split into two sets: the **training set** and the **test set**
    1. ML system is built using the training set only
    2. ML system is asked to predict the output values for the data in the "unseen" test set
  - The accumulated errors give the **test set error**, used to evaluate the model
    - Maybe high variance in the results if the test is repeated
  - To tune the system appropriately and reduce this variance, we can split the training set in a **training subset** and a **validation subset**
    - The validation subset can be employed for model selection, i.e. tune hyperparameters
      - train the system
      - repeatedly evaluate it using the validation subset using different settings
  - Even in this way, the evaluation may depend heavily on **which data points end up in the training set and which end up in the test set**, and thus the evaluation may be significantly different depending on how the splitting is made



Cross-validation techniques

- **n-fold cross validation**
  - The available data is randomly split into n folds without replacement:
    - $n - 1$ folds are used for training
    - the remaining fold is used for testing
  - The procedure is repeated n times, choosing a different fold for testing each iteration (no overlapping)
  - A global performance measure is obtained by averaging the individual measurements
    - lower variance estimate than the holdout method
  - In most cases, $n = 5$ or 10
    - n large $\Rightarrow$ more data for training $\Rightarrow$ lower bias in E but longer runtime
  - A **high variance** indicates a situation of overfitting



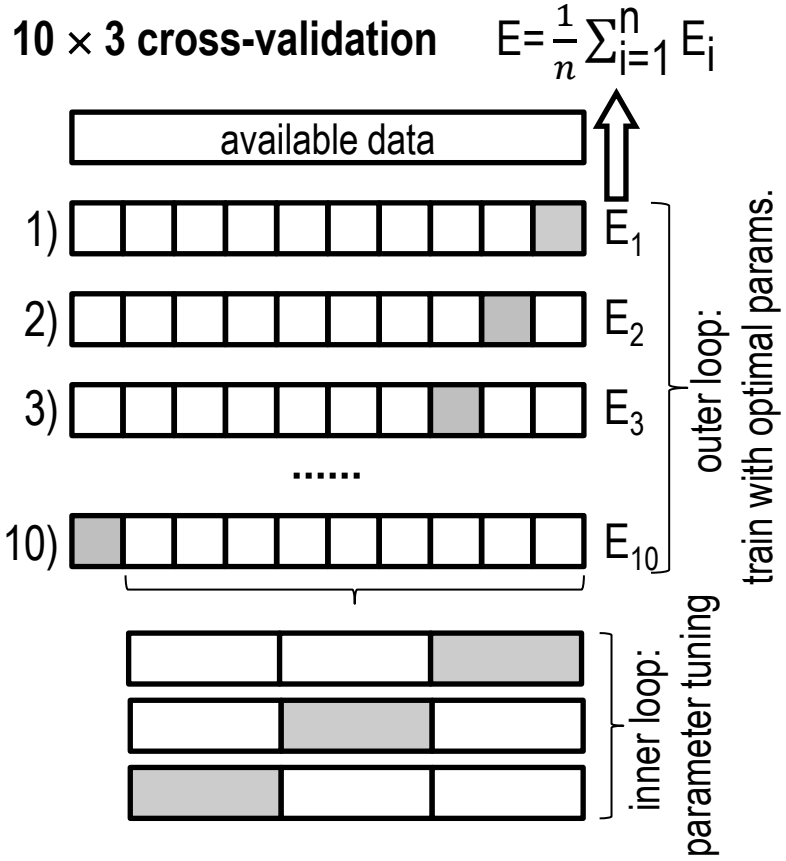
$$E = \frac{1}{n} \sum_{i=1}^{n} E_i$$

$$V = Var(E_i)$$

  - **Stratified n-fold cross validation**
    - Class proportions are preserved in each fold
    - Better performance estimates as for bias and variance
  - **Leave-one-out cross validation** (LOOCV)
    - $n = N$, the size of the dataset; hence, at each iteration, the test set comprises one single sample
    - recommended for small datasets

Cross-validation techniques

- **Nested cross validation**
  - Outer n-fold: performance estimation
    the available data is randomly split into
    n folds without replacement:
      - $n - 1$ folds are used for training
      - the remaining fold is used for test
    - The procedure is repeated n times, choosing a different fold for testing each iteration
  - Inner m-fold: model selection (hyperp. tuning)
    the set of training folds is split into $\boldsymbol{m}$ **folds** leaving one for validation
  - A global performance measure is obtained by averaging the individual measurements
  - This measure gives a good estimate of what to expect from unseen data

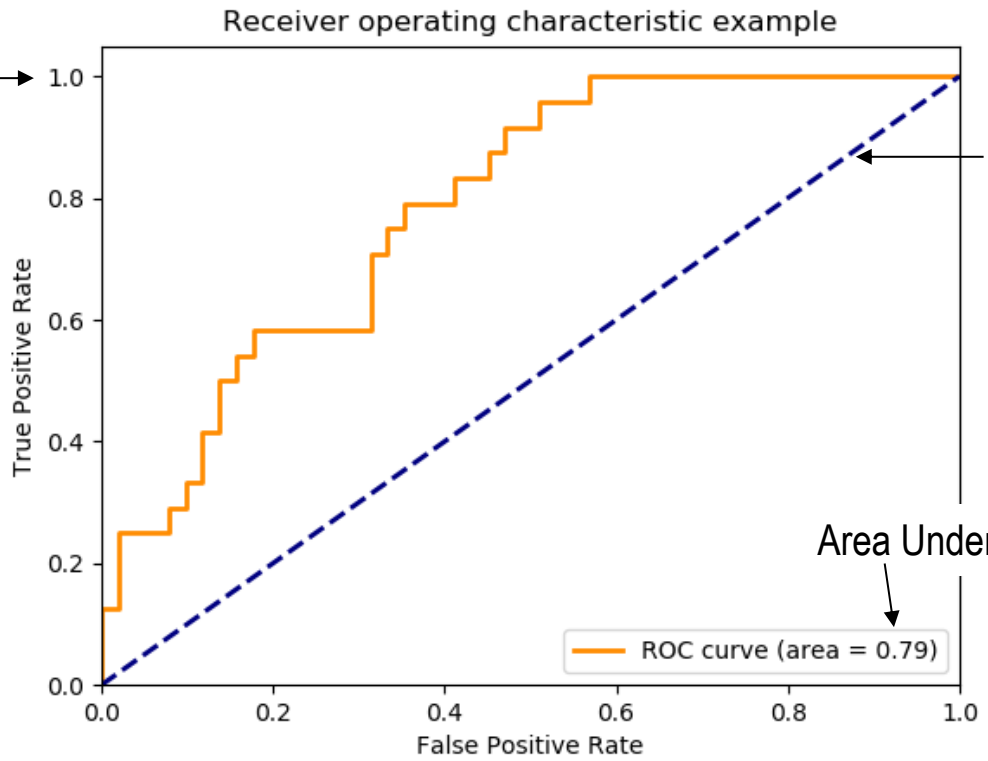


Contents

ROC curves

- **Receiver Operating Characteristic** curve (concept from early Radar days)
  - Set of (FPR, TPR) points obtained by varying the algorithm's parameters
    - Every (FPR, TPR) point is 1 classifier configuration, choose the closest to the (0,1) point

(FPR=0, TPR=1)
perfect classifier



random guessing:
FPR = TPR ⇒
FP / Ne = TP / Po ⇒
FP / TP = Ne / Po
Ne / Po = 1 ⇒ TP = FP

$$FPR = \frac{FP}{\text{Ne}} = \frac{FP}{FP + TN}$$

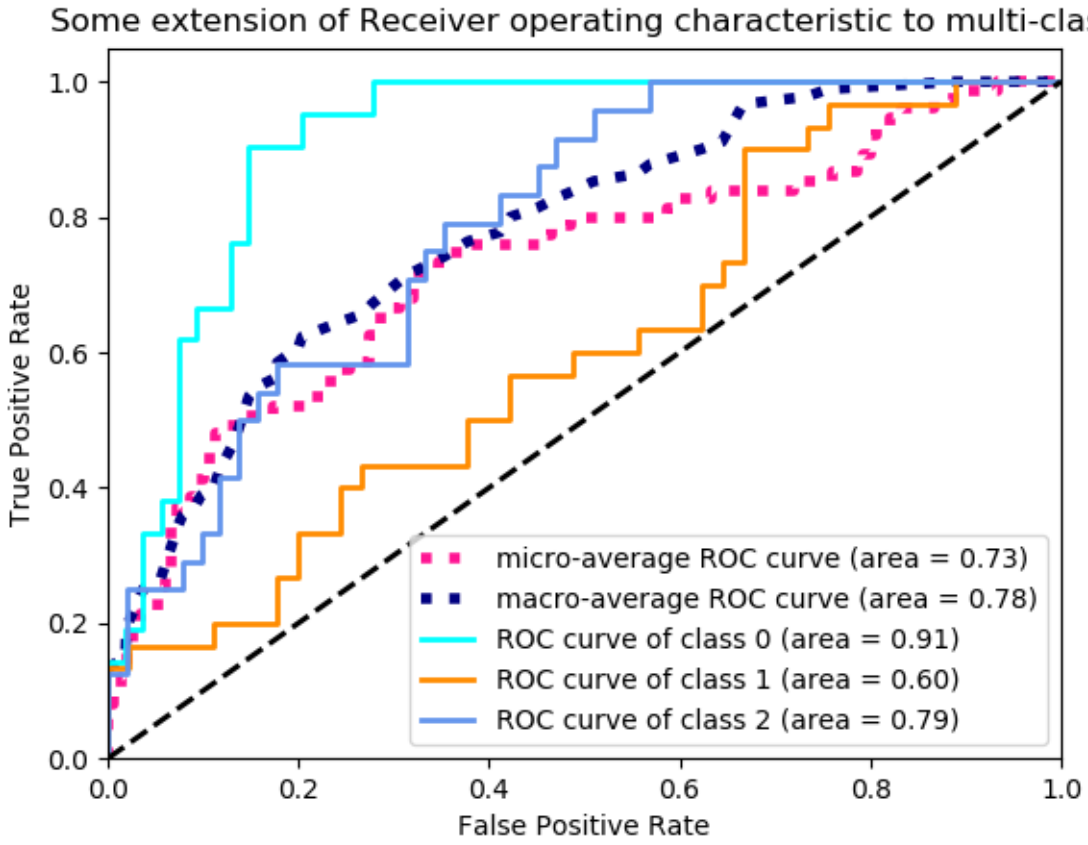
$$TPR = \frac{TP}{\text{Po}} = \frac{TP}{TP + FN}$$

Area Under the Curve (AUC)

    - Can also be shown as sensitivity (TPR) vs 1 – specificity (FPR)
  - **Area Under the Curve** (AUC)
    - Global measure of classifier performance, the higher the better

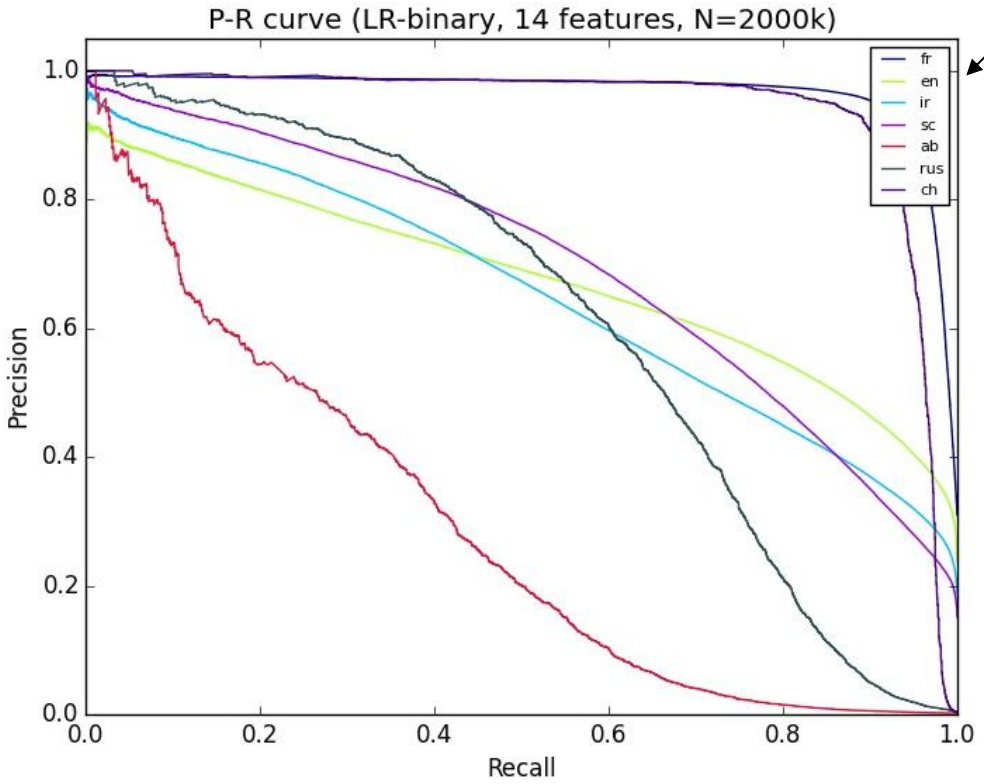
ROC curves

- **Receiver Operating Characteristic** curve (concept from early Radar days)
  - can also be calculated for multi-class problems



ROC curves

- Other usual curves for classifier performance characterization are the **Precision-Recall** curves



$(R = 1, P = 1)$
perfect classifier

$$P = \frac{TP}{\widehat{\text{Po}}} = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{\text{Po}} = \frac{TP}{TP + FN}$$

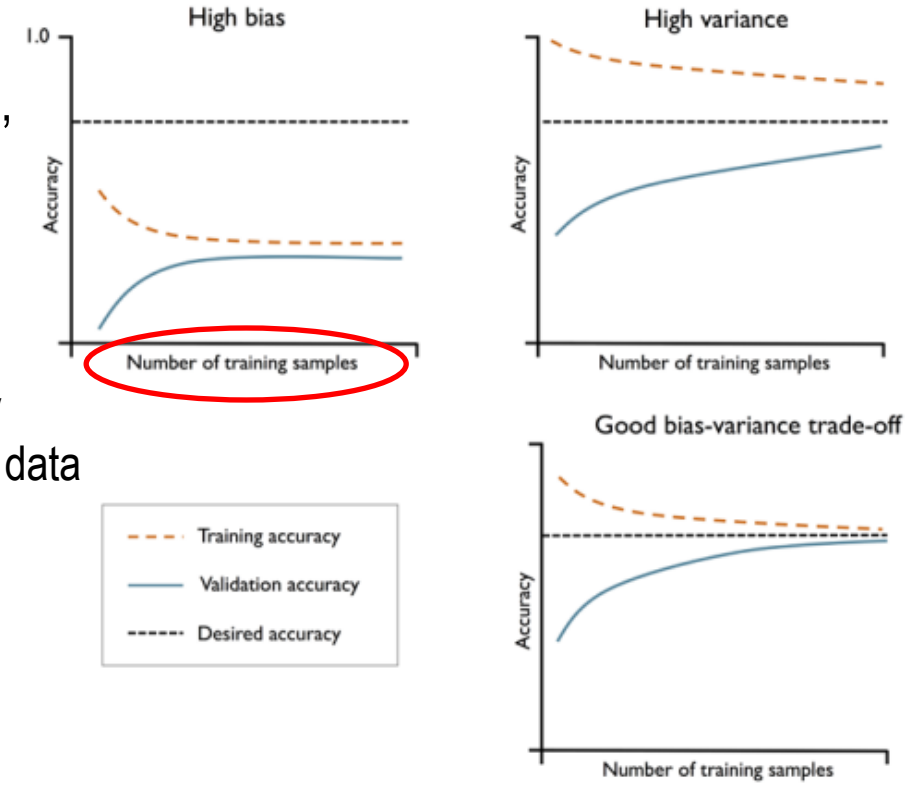
  - The AUC is also of application in this case, also maximum R for P = 1, etc.
  - Note: TN are not accounted for in this curve, it is the kind of problem where negatives are not as relevant as positives, e.g. inspection systems

Contents

Bias and variance problems

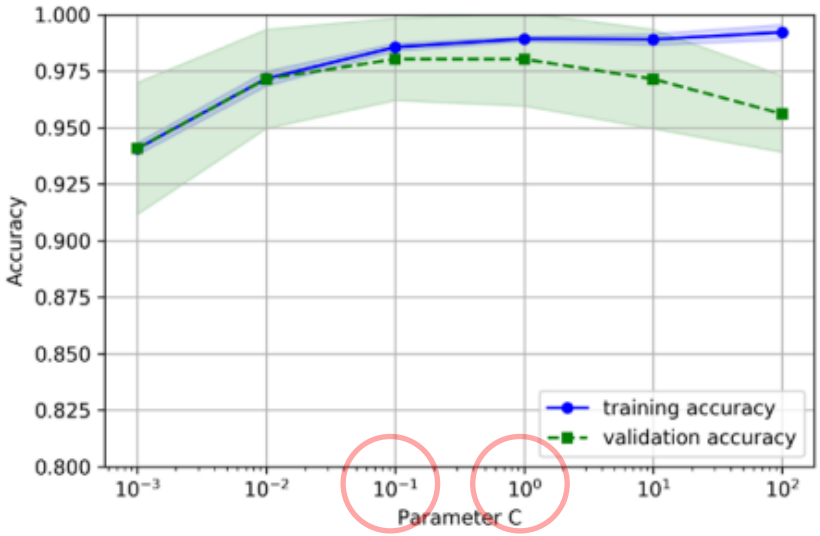
- By plotting the model training and validation accuracies as functions of the training set size (**learning curves**), we can easily detect whether the model suffers from high variance or high bias
  - **high bias**
    - symptom:
      low training and validation accuracy and hence the model **underfits** the data
    - solution:
      increase model complexity either with more parameters or more features
  - **high variance**
    - symptom: large gap between training and validation accuracy, the model captures too well the training data (**overfitting**)
    - solution: collect more training data (care with noisy data, performance will not improve), reduce the model complexity (e.g. regularization) or remove some features (via feature selection or extraction)

progressively increase training set size, train and test



Model tuning

- In machine learning, we have two types of parameters:
  - those that are learned from training data, e.g. separating hyperplane
  - those that are optimized separately, e.g. decision tree depth or C in soft SVM
- The latter are known as **hyperparameters**
  - They can be set one by one, using **performance** (train and validation) **learning curves**



  - or via **grid search**
    - **brute-force** exhaustive search paradigm where we specify a list of values for different hyperparameters and the computer evaluates the model performance for each combination, e.g. when optimizing SVM
      - kernel type: linear, (in)homogeneous polynomial, rbf
      - kernel parameters: polynomial degree q, coefficient γ
      - soft SVM hyperparameter C

$$K_{ln}(x,z) = x^T z$$
$$K_{hp}(x,z) = (x^T z)^q, \ q > 0$$
$$K_{ip}(x,z) = (\gamma x^T z + 1)^q, \ q > 0$$
$$K_{rbf}(x,z) = e^{-\gamma \|x-z\|^2}$$

$$\min J(w,\xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$$

Lecture 6: Assessment of machine (supervised) learning systems

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ