Lecture 7: Model Evaluation I)

COMP90049
Introduction to Machine Learning

Semester 1, 2021

Lea Frermann, CIS

(parts adapted from slides by Karin Verspoor and Tim Baldwin)

Roadmap

So far

- Instance-based classification with KNN
- Probabilities and probabilistic modeling
- Optimization and MLE
- Probabilistic classification with Naive Bayes

Today... Evaluation

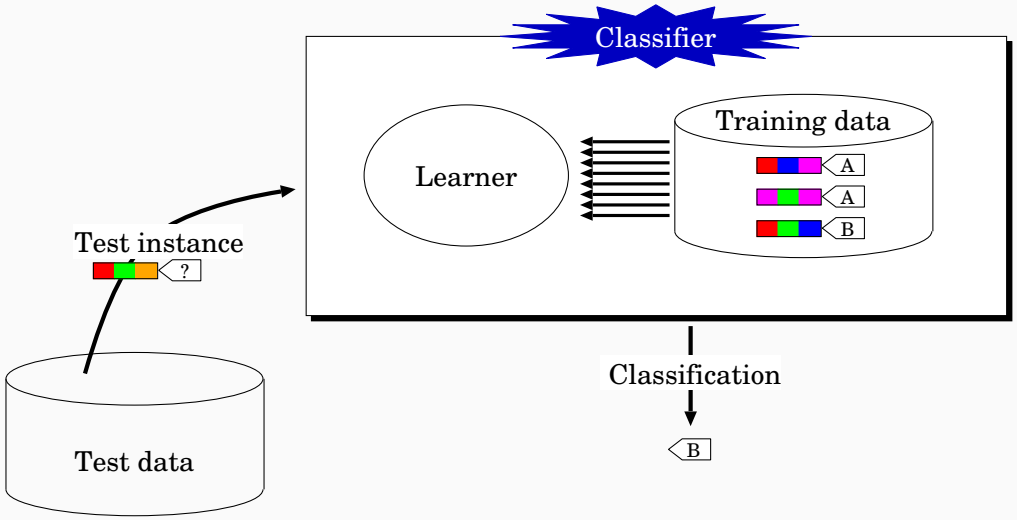
- How do we know that we succeeded in learning?
- Evaluation paradigms
- Evaluation methods

THE UNIVERSITY OF MELBOURNE

Classification Evaluation

The Nature of “Classification”

- Input: set of labelled training instances; set of unlabelled test instances
- Model: an estimate of the underlying target function
- Output: prediction of the classes of the test instances



Goal 1: Model Evaluation

Our goal (in a supervised Machine Learning framework):

- Have a perfect model?
  - Not necessarily clear what that means
  - More difficult than necessary
- Make predictions that are correct!
- Train and test machine learning models based on the limited data available
- Estimate the *true* performance on any data based on the errors made on the limited labelled data available

Goal 2: Model Selection

Choosing the best model from a number of possibilities

- Different machine learning algorithms
- Different parameterizations for the same algorithm
- Different training parameters
- Other considerations: Efficiency, explainability, fairness, ...

Evaluation Strategies

Evaluating Classification I

Main idea:

- Train (build model) using **training data**
- Test (evaluate model) on **test data**

But often, we just have **data** — a collection of instances

Evaluating Classification II

An obvious strategy, which is highly **not recommended**:

- Use all of the instances as training data
  - Build the model using all of the instances
- Use all of the (same) instances as test data
  - Evaluate the model using all of the instances

“Testing on the training data” tends to grossly over-estimate classifier performance.

Effectively, we are telling the classifier what the correct answers are, and then asking whether it can come up with the correct answers.

Holdout

One solution: **Holdout** evaluation strategy

- Each instance is randomly assigned as either a training instance **or** a testing instance
- Effectively, the data is **partitioned** — no overlap between datasets
- Evaluation strategy:
  - Build the model using (only) the training instances
  - Evaluate the model using (only) the (different) test instances

Very commonly used strategy; typical split sizes are approximately 50–50, 80–20, 90–10 (train, test)

Source(s): Tan et al. [2006, pp 186–7]

Holdout

Advantages

- simple to work with and implement
- fairly high reproducability

Disadvantages

- size of the split affects estimate of the model's behaviour:
  - lots of test instances, few training instances: learner doesn't have enough information to build an accurate model
  - lots of training instances, few test instances: learner builds an accurate model, but test data might not be representative (so estimates of performance can be too high/too low)

Repeated Random Subsampling

Slower, but somewhat better solution: **Repeated Random Subsampling**

- Like Holdout, but iterated multiple times:
  - A new training set and test set are randomly chosen each time
  - Relative size of training–test is fixed across iterations
  - New model is built each iteration
- Evaluate by averaging (chosen metric) across the iterations

Repeated Random Subsampling

Advantages:

- averaging Holdout method tends to produce more reliable results

Disadvantages:

- more difficult to reproduce
- slower than Holdout (by a constant factor)
- wrong choice of training set–test set size can still lead to highly misleading results (that are now very difficult to sanity–check)

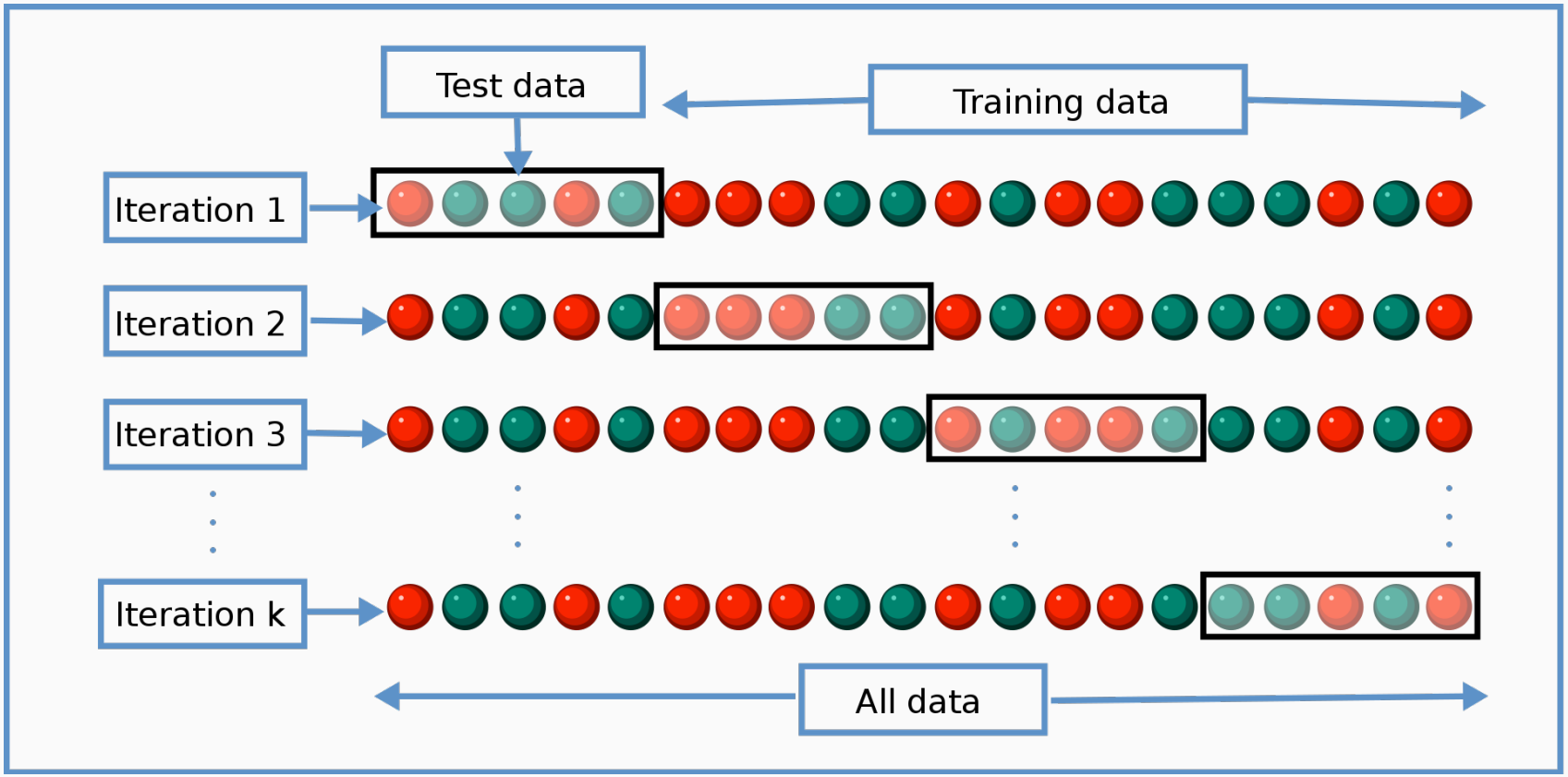
Cross–Validation

Usually preferred alternative: **Cross–Validation**

- Data is progressively split into a number of partitions m (≥ 2)
- Iteratively:
  - One partition is used as test data
  - The other $m - 1$ partitions are used as training data
- Evaluation metric is aggregated across m test partitions
  - This could mean averaging, but more often, counts are added together across iterations

Cross–Validation

Usually preferred alternative: **Cross–Validation**



https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.svg

Cross–Validation

Why is this better than Holdout/Repeated Random Subsampling?

- **Every** instance is a test instance, for some partition
  - Similar to testing on the training data, but without dataset overlap
  - Evaluation metrics are calculated with respect to a dataset that looks like the entire dataset (i.e. the entire dataset)
- Takes roughly the same amount of time as Repeated Random Subsampling (but see below)
- Very reproducible
- Can be shown to minimise **bias** and **variance** of our estimates of the classifier's performance (more on this in Evaluation II)

Cross–Validation

How big is *m*?

- Number of folds directly impacts runtime *and* size of datasets:
  - Fewer folds: more instances per partition, more variance in performance estimates
  - More folds: fewer instances per partition, less variance but slower
- Most common choice of *m*: 10 (occasionally, 5)
  - Mimics 90–10 Holdout, but far more reliable
- Best choice: *m*=*N*, the number of instances (known as **Leave–One–Out Cross–Validation**):
  - Maximises training data for the model
  - Mimics actual testing behaviour (every test instance is treated as an individual test "set")
  - Way too slow to use in practice

Inductive Biases and No Free Lunch

A learner that makes no **a priori assumptions** *regarding the identity of the target concept has no* **rational basis** *for classifying any unseen instances"*

[Mitchell, 1997, Ch. 2.7.3]

The No Free Lunch Theorem (Wolpert and Macready, 1997)

- Intuition: You don't get *something* for *nothing*. You won't get a *good classifier* (or any ML algorithm) without *making assumptions*.
- Averaged across all possible problems, the performance of any two algorithms is identical.

$\rightarrow$ If algorithm A does well on $p1$ it necessarily performs worse on some $p2$

- Averaged across all classification problems, the generalization error on a held-out test set of any two classifiers is identical. There is no universally superior classifier.

Inductive Biases and No Free Lunch

A learner that makes no **a priori assumptions** *regarding the identity of the target concept has no* **rational basis** *for classifying any unseen instances"*

[Mitchell, 1997, Ch. 2.7.3]

- **Inductive Learning Hypothesis:** Any hypothesis found to approximate the target function well over (a sufficiently large) training data set will also approximate the target function well over **unseen** test examples.

- Assumptions must be made about the data to build a model and make predictions (**inductive biases**)

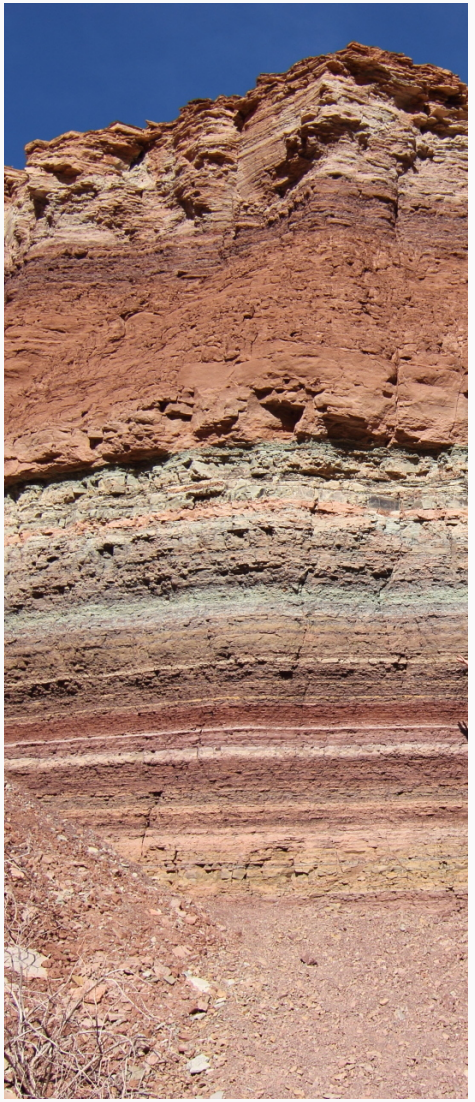
  $\rightarrow$ Different assumptions will lead to different predictions

Stratification

- Typical inductive bias in our evaluation framework (n.b. independent of model): **Stratification**

  → Assume that **class distribution** of unseen instances will be the same as distribution of seen instances

- When constructing Holdout/Cross–Validation partitions, ensure that training data and test data **both** have same class distribution as dataset as a whole



THE UNIVERSITY OF MELBOURNE

Evaluation Measures

Error E

The fraction of incorrect predictions.

$$E = \frac{1}{N}(1 - I(y_i, \hat{y}_i)), \textit{where} \quad I(a, b) \begin{cases} 1, & \text{if } a == b \\ 0, & \text{otherwise} \end{cases}$$

Error rate reduction: $\mathrm{ERR} = \frac{\mathrm{E}_0 - \mathrm{E}}{\mathrm{E}_0}$

Error E

The fraction of incorrect predictions.

$$E = \frac{1}{N}(1 - I(y_i, \hat{y}_i)), \textit{where} \quad I(a,b) \begin{cases} 1, & \text{if } a == b \\ 0, & \text{otherwise} \end{cases}$$

Error rate reduction: $\mathrm{ERR} = \frac{\mathrm{E}_0 - \mathrm{E}}{\mathrm{E}_0}$

Outlook	Temperature	Humidity	Windy	Actual	Classified
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	
sunny	cool	normal	FALSE	yes	
rainy	mild	normal	FALSE	yes	
sunny	mild	normal	TRUE	yes	no
overcast	mild	high	TRUE	yes	no
overcast	hot	normal	FALSE	yes	yes
rainy	mild	high	TRUE	no	yes

Two Types of Errors

Contingency Tables

$\downarrow y, \hat{y} \rightarrow$	1	0
1	true positive (TP)	false negative (FN)
0	false positive (FP)	true negative (TN)

Not all Errors are equal

- Some problems want to strongly **penalize false negative errors** e.g.,
- Other problems want to strongly **penalize false positive errors**,
- Attach a “cost” to each type of error

Accuracy

$\downarrow y, \hat{y} \rightarrow$	1	0
1	true positive (TP)	false negative (FN)
0	false positive (FP)	true negative (TN)

Accuracy: The basic evaluation metric

$$\text{Accuracy} = \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}} = \frac{TP + TN}{TP + FP + TN + FN}$$

- Same as $1 - E$
- Quantifies how frequently the classifier is correct

Accuracy

Outlook	Temperature	Humidity	Windy	Actual	Classified
sunny	hot	high	FALSE	no	
sunny	hot	high	TRUE	no	
overcast	hot	high	FALSE	yes	
rainy	mild	high	FALSE	yes	
rainy	cool	normal	FALSE	yes	
rainy	cool	normal	TRUE	no	
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	
sunny	cool	normal	FALSE	yes	
rainy	mild	normal	FALSE	yes	
sunny	mild	normal	TRUE	yes	no
overcast	mild	high	TRUE	yes	yes
overcast	hot	normal	FALSE	yes	yes
rainy	mild	high	TRUE	no	yes

Accuracy

4 test instances; 2 correct predictions, 2 incorrect predictions

$$\begin{aligned} \text{Accuracy} &= \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}} \\ &= \frac{2}{4} = 50\% \end{aligned}$$

Precision and Recall

$\downarrow y, \hat{y} \rightarrow$	1 (interesting)	0 (uninteresting)
1 (interesting)	true positive (TP)	false negative (FN)
0 (uninteresting)	false positive (FP)	true negative (TN)

With respect to **just the interesting class**:

- **Precision**: How often are we correct, when we predict that an instance is interesting?

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: What proportion of the truly interesting instances have we correctly identified as interesting?

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

Precision and Recall

Precision/Recall are typically in an **inverse relationship**. We can generally set up our classifier, so that:

- The classifier has high Precision, but low Recall
- The classifier has high Recall, but low Precision

But, we want **both** Precision and Recall to be high. A popular metric that evaluates this is **F-score**:

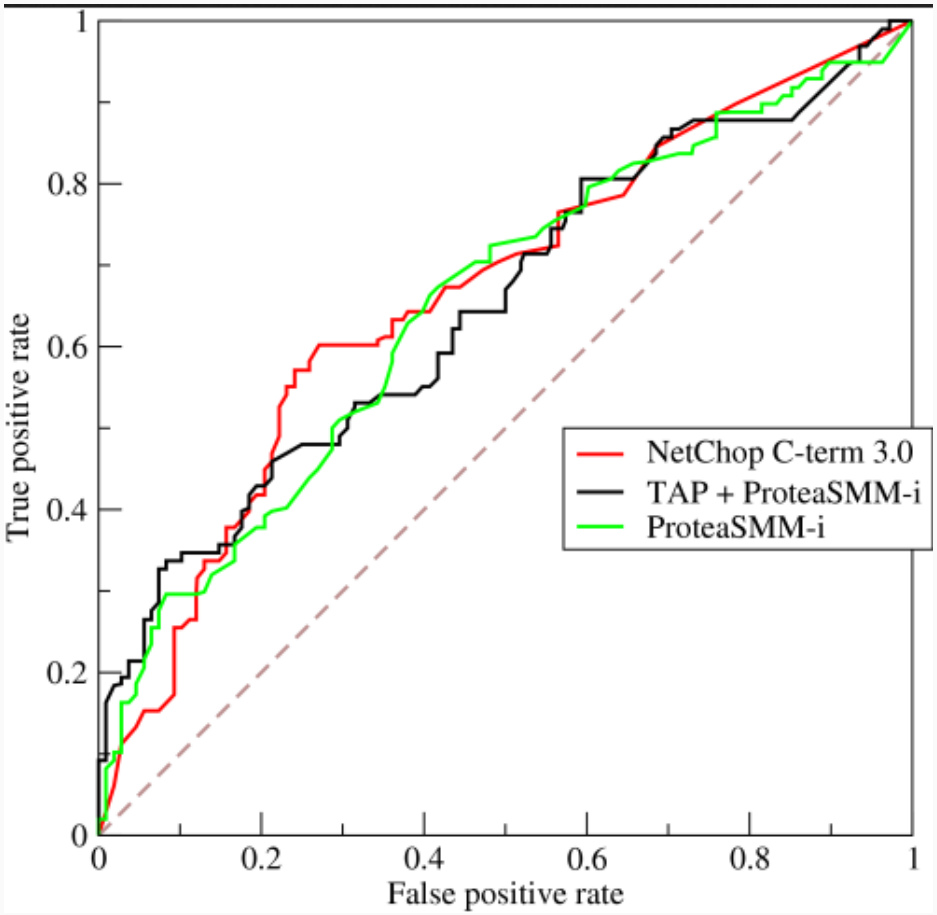
$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R}$$

$$F_1 = \frac{2PR}{P+R}$$

Capturing trade-offs between different objectives

Receiver Operating characteristics (ROC-curve)

- Goal: Maximize the area under the ROC curve: (a) minimize the **false alarms** and maximize the **true alarms**
- History: trade off between **false alarms** and **true alarms** in signal processing



- X-axis: percentage of false positives
  Y-axis: percentage of true positives

THE UNIVERSITY OF MELBOURNE

Many other Metrics!

		Predicted condition			
	Total population	Predicted Condition positive	Predicted Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
True condition	condition positive	**True positive**	**False Negative** (Type II error)	True positive rate (TPR), Sensitivity, Recall, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$
	condition negative	**False Positive** (Type I error)	**True negative**	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

From Wikipedia:

`https://en.wikipedia.org/wiki/Sensitivity_and_specificity`

Multi-class Evaluation

For a multi-class problem, assume an **Interesting** class (`I`) and several **Uninteresting** classes (`U1`, `U2`, ...).

		Predicted			
		I	***U1***	***U2***	...
Actual	***I***	TP	FN	FN	...
	U1	FP	TN	TN	...
	U2	FP	TN	TN	...
	...	...	...	...	...

A multi-class **Confusion Matrix**.

Typically, all classes are “interesting” in a multi-class context.

Multi-class Evaluation

- The natural definition of Accuracy still makes sense in a multi-class context,
- The technical definition behaves strangely: re-interprets the multi-class problem as a special case of a two-class problem, namely One-vs-Rest

- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
- **macro-averaging**: calculate P, R per class and then average

$$\text{Precision}_M = \frac{\sum_{i=1}^{c} \text{Precision}_i}{c}$$

$$\text{Recall}_M = \frac{\sum_{i=1}^{c} \text{Recall}_i}{c}$$

Multi-class Evaluation

- The natural definition of Accuracy still makes sense in a multi-class context,
- The technical definition behaves strangely: re-interprets the multi-class problem as a special case of a two-class problem, namely One-vs-Rest

- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
- **micro-averaging**: combine all test instances into a single pool

$$\text{Precision}_\mu = \frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c} TP_i + FP_i}$$

$$\text{Recall}_\mu = \frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c} TP_i + FN_i}$$

Multi-class Evaluation

- The natural definition of Accuracy still makes sense in a multi-class context,
- The technical definition behaves strangely: re-interprets the multi-class problem as a special case of a two-class problem, namely One-vs-Rest

- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
- **weighted averaging**: calculate P, R per class and then average, based on the proportion of instances in that class

$$\text{Precision}_W = \sum_{i=1}^{c} \text{Precision}_i \times \left(\frac{n_i}{N}\right)$$

$$\text{Recall}_W = \sum_{i=1}^{c} \text{Recall}_i \times \left(\frac{n_i}{N}\right)$$

How to average your predictions?

- Macro average treats all classes equal: emphasizes small classes. Micro average is dominated by large classes.

- **Macro-averaged F-score**: is it the F-score of macro-averaged P (over classes) and macro-averaged R (over classes)? Or the macro-average (over classes) of the F-score for each class?

- If we are doing **Repeated Random Subsampling**, and want **weighted-averaged Precision**, do we average the weighted Precision (over classes) for each iteration of Random Subsampling? or do we take the weighted average (over classes) of the Precision averaged over the iterations of Subsampling? Or the weighted average over the instances aggregated over the iterations?

Model comparison

Baselines vs. Benchmarks

- **Baseline** = naive method which we would expect any reasonably well-developed method to better

  *e.g. for a novice marathon runner, the time to **walk** 42km*

- **Benchmark** = established rival technique which we are pitching our method against

  e.g. for a marathon runner, the time of our last marathon run/the world record time/3 hours/...

- "Baseline" often used as umbrella term for both meanings

The Importance of Baselines

- Baselines are important in establishing whether any proposed method is doing better than “dumb and simple”

  “dumb” methods often work surprisingly well

- Baselines are valuable in getting a sense for the intrinsic difficulty of a given task (cf. accuracy = 5% vs. 99%)

- In formulating a baseline, we need to be sensitive to the importance of positives and negatives in the classification task

  limited utility of a baseline of unsuitable for a classification task aimed at detecting potential sites for new diamond mines (as nearly all sites are unsuitable)

THE UNIVERSITY OF MELBOURNE

Random Baseline

Method 1: randomly assign a class to each test instance

- Often the only option in unsupervised/semi-supervised contexts

Method 2: randomly assign a class to each test instance, weighting the class assignment according to $P(C_k)$

- Assumes we know the prior probabilities
- Alleviate effects of variance by:
  - running method N times and calculating the mean accuracy

    OR
  - arriving at a deterministic estimate of the accuracy of random assignment = $\sum_i P(C_i)^2$

Zero-R (Zero rules)

Also known as **majority class** baseline

- **Method:** classify all instances according to the most common class in the training data
- The most commonly used baseline in machine learning
- Inappropriate if the majority class is `FALSE` and the learning task is to identify needles in the haystack

Outlook	Temperature	Humidity	Windy	y (Play)	$\hat{y}$
sunny	hot	high	FALSE	no	
sunny	hot	high	TRUE	no	
overcast	hot	high	FALSE	yes	
rainy	mild	high	FALSE	yes	
rainy	cool	normal	FALSE	yes	
rainy	cool	normal	TRUE	no	
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	?

One-R (One Rule)

Introduction

- Select **one** attribute and use it to predict an instance's class
- Test each attribute, and select the one with the smallest error rate
- Each attribute-specific test is often called "Decision stump" (more on that in the Trees lecture)

Intuition

- If there is a single, simple feature with which we can classify most of our features correctly – do we really need a sophisticated machine learner?

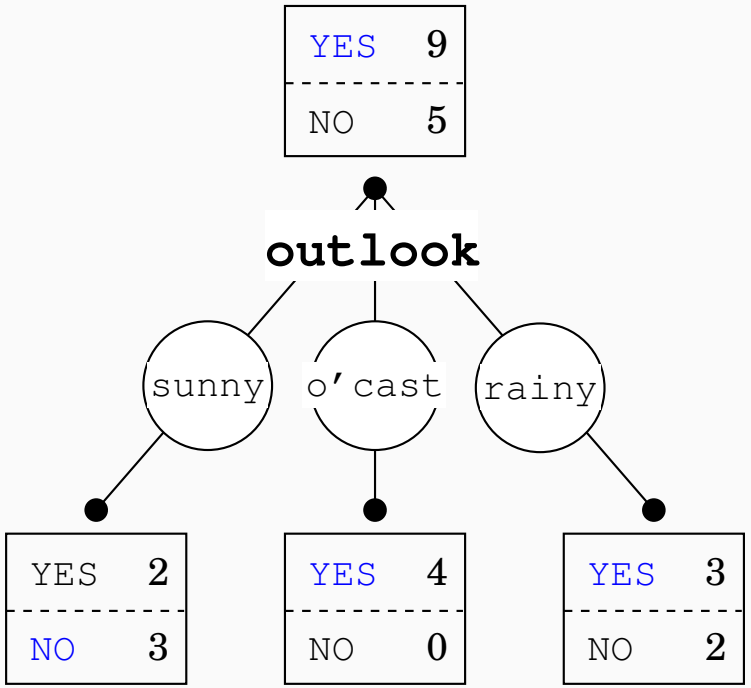
One-R pseudo-code

```
For each attribute

    For each value of the attribute, make a rule:

        (i)   count how often each class appears
        (ii)  find the most frequent class
        (iii) make the rule assign that class to this value

    Calculate the error rate of the rule

Choose the rules with the smallest error rate
```

Weather dataset

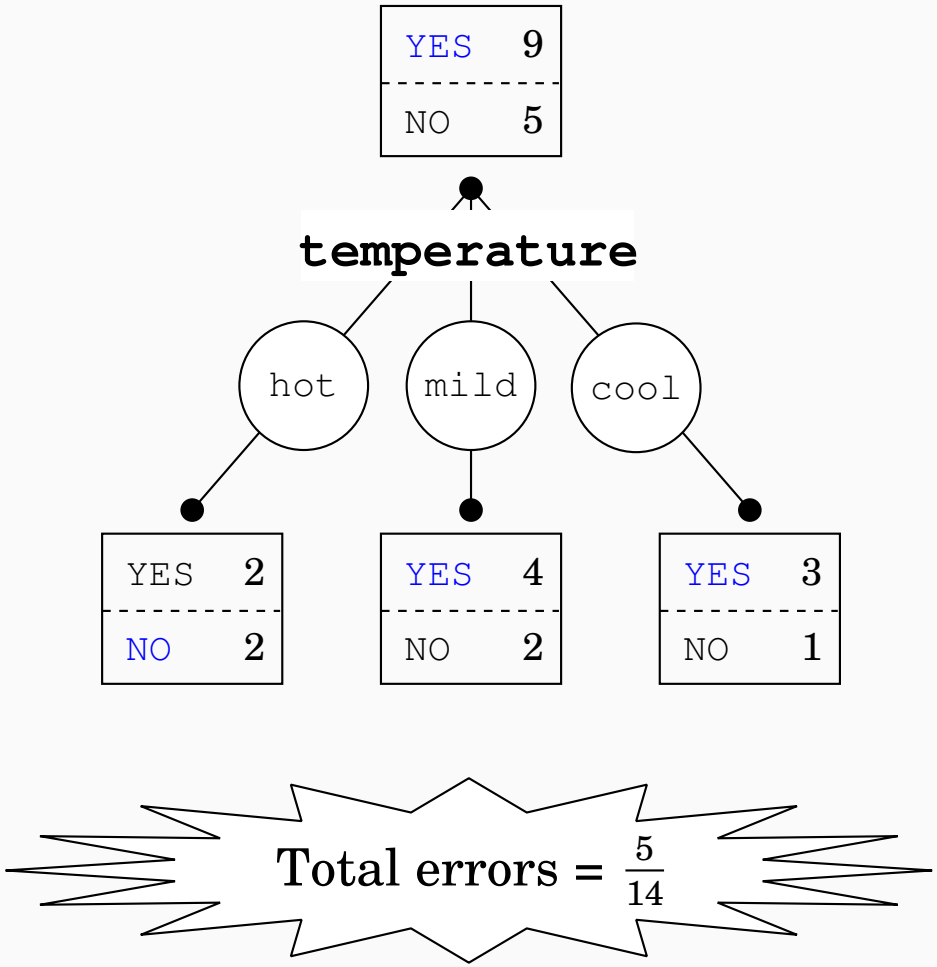
Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Attribute Option 1 (outlook)

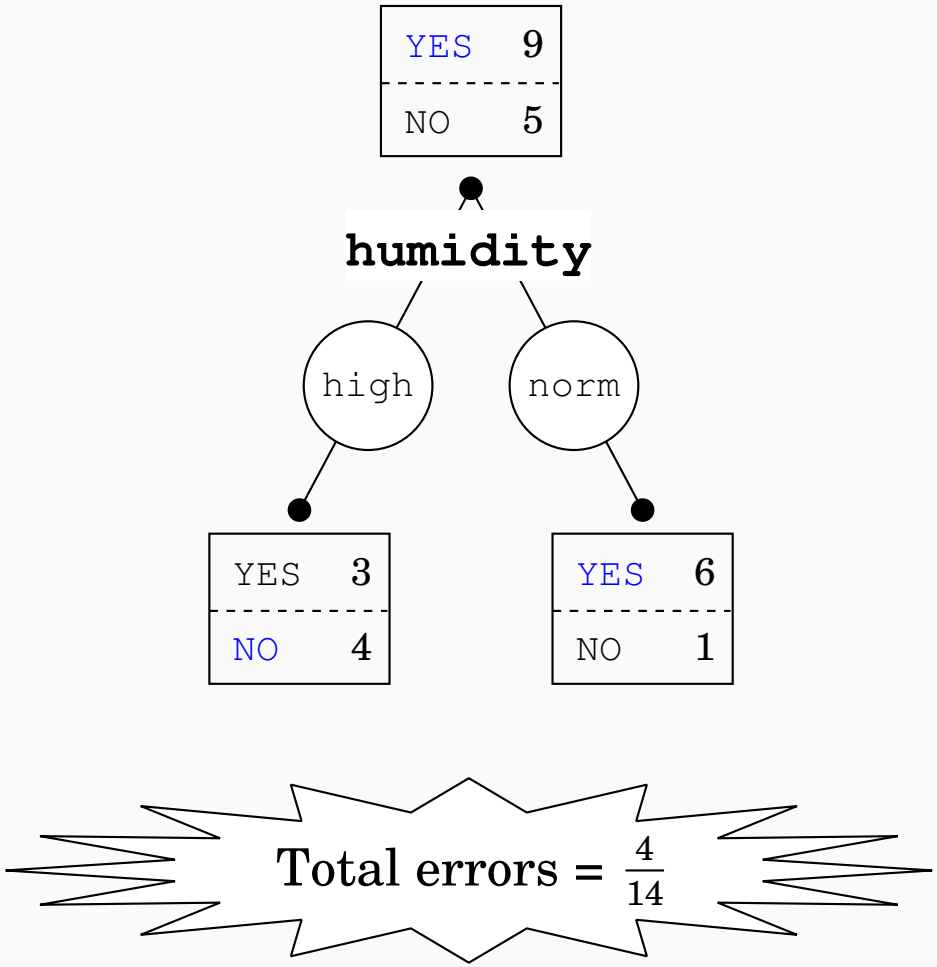


Total errors = $\frac{4}{14}$

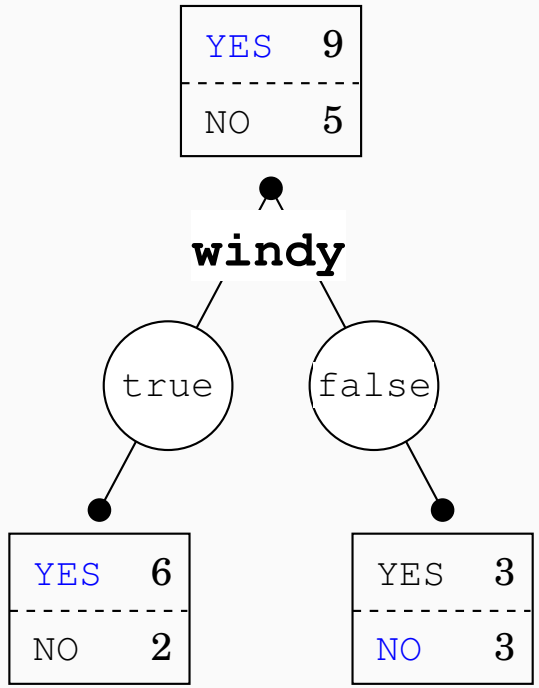
Attribute Option 2 (temperature)



Attribute Option 3 (humidity)



Attribute Option 4 (windy)



Total errors = $\frac{5}{14}$

One-R: Reflections

Advantages:

- simple to understand and implement
- simple to comprehend the results
- surprisingly good results

Disadvantages:

- unable to capture attribute interactions
- bias towards high-arity attributes (attributes with many possible values)

Summary

Today

- How do we set up an evaluation of a classification system?
- What are the measures we use to assess the performance of the classification system?
- What is a baseline? What are some examples of reasonable baselines to compare with?

Next lecture

- Optimization (part 2)
- Logistic Regression

References

Data Mining: Concepts and Techniques, 3rd ed., Jiawei Han and Micheline Kamber, Morgan Kaufmann, 2006. Chapter 8.5

Chris Bishop. Pattern Recognition and Machine Learning. Chapters: 1.3

Addison Wesley, 2006.David Wolpert and William Macready. No free lunch theorems for optimization.IEEETransactions on Evolutionary Computation, 1:67–82, 1997.

Jacob Eisenstein. Natural Language Processing. Chapter 4.4