Lecture 13: Evaluation

COMP90049
Knowledge Technologies

Sarah Erfani and Karin Verspoor, CIS

Semester 2, 2017

THE UNIVERSITY OF MELBOURNE

Tensions in Classification

- **Generalisation**: how well does the classifier generalise from the specifics of the training examples to predict the target function?
- **Overfitting:** has the classifier tuned itself to the idiosyncracies of the training data rather than learning its generalisable properties?
- **Consistency:** is the classifier able to flawlessly predict the class of all training instances?

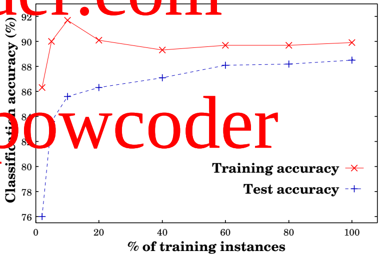
Generalisation Problem in Classification

- **Under-fitting:** model not expressive enough to capture patterns in the data.
- **Over-fitting:** model too complicated; capture noise in the data.
- **Appropriate-fitting** model captures essential patterns in the data.

Evaluating Classification

- Usually, the given data set is partitioned into two disjoint sets. The *training set* is used to build the model; the *test set* is used to validate it.
- We can measure the proportion of the time the class label is correctly discovered for test inputs.
- Learning curves represent the performance of a fixed learning strategy over different sizes of training data, relative to a fixed evaluation metric.

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 held-out *test examples*.



How to evaluate a classifier?

For a two class problem:

There are Positive and Negative cases

A classifier may classify

- a Positive instance as Positive (True Positive, TP)
- a Positive instance as Negative (False Negative, FN)
- a Negative instance as Positive (False Positive, FP)
- a Negative instance as Negative (True Negative, TN)

		Predicted	
		Y	***N***
Actual	***Y***	*true positive (TP)*	*false negative (FN)*
	N	*false positive (FP)*	*true negative (TN)*

Clustering accuracy

Outlook	Temperature	Humidity	Windy	Cluster	Play
sunny	hot	high	FALSE	0	no
sunny	hot	high	TRUE	0	no
overcast	hot	high	FALSE	0	yes
rainy	mild	high	FALSE	1	yes
rainy	cool	normal	FALSE	1	yes
rainy	cool	normal	TRUE	1	no
overcast	cool	normal	TRUE	1	yes
sunny	mild	high	FALSE	0	no
sunny	cool	normal	FALSE	1	yes
rainy	mild	normal	FALSE	1	yes
sunny	mild	normal	TRUE	1	yes
overcast	mild	high	TRUE	1	yes
overcast	hot	normal	FALSE	0	yes
rainy	mild	high	TRUE	0	no

Cluster 0 = “no”, Cluster 1 = “yes”

Clustering accuracy

Outlook	Temperature	Humidity	Windy	Cluster	Play
sunny	hot	high	FALSE	0	no
sunny	hot	high	TRUE	0	no
overcast	hot	high	FALSE	0	yes
rainy	mild	high	FALSE	1	yes
rainy	cool	normal	FALSE	1	yes
rainy	cool	normal	TRUE	1	no
overcast	cool	normal	TRUE	1	yes
sunny	mild	high	FALSE	0	no
sunny	cool	normal	FALSE	1	yes
rainy	mild	normal	FALSE	1	yes
sunny	mild	normal	TRUE	1	yes
overcast	mild	high	TRUE	1	yes
overcast	hot	normal	FALSE	0	yes
rainy	mild	high	TRUE	0	no

Cluster 0 = "no", Cluster 1 = "yes"

		Predicted	
		Y	***N***
Actual	***Y***	*TP (7)*	*FN (2)*
	N	*FP (1)*	*TN (4)*

Accuracy

- *Classification accuracy* is the proportion of instances for which we have correctly predicted the label, which corresponds to:

$$\mathrm{ACC} = \frac{TP + TN}{TP + FP + FN + TN}$$

- Alternatively, we sometimes talk about the *error rate*:

$$\mathrm{ER} = \frac{FP + FN}{TP + FP + FN + TN}$$

N.B. $\mathrm{ER} = 1 - \mathrm{ACC}$

- We also sometimes refer to the *error rate reduction*, comparing the error rate ER for a given method with that for a benchmark/baseline method ER_0:

$$\mathrm{ERR} = \frac{\mathrm{ER}_0 - \mathrm{ER}}{\mathrm{ER}_0}$$

Precision and Recall

- If we wish to know what we have positively identified **not** what we have correctly ignored (or equivalently performance relative to a single class of interest), we use *precision* and *recall*

$$\text{Precision} = \text{positive predictive value} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \text{sensitivity} = \frac{TP}{TP + FN}$$

- Precision: Proportion of positive predictions that are correct
- Recall: Accuracy with respect to positive cases; also called true positive rate
- *Specificity* is the accuracy with respect to negative cases

$$\text{Specificity} = \frac{TN}{TN + FP}$$

(sensitivity/specificity is often used in scientific applications)

Precision and Recall over Multiple Categories

- To compute an overall P/R value over multiple categories:
  1. *micro-averaging*

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

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

  2. *macro-averaging*

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

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

- In what situations are they the same/different?

Metrics, compared

		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 $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	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 $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	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+) $= \frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
		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−) $= \frac{FNR}{TNR}$	

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

F-score

- In applications where we make individual decisions for each data point rather than generating a monolithic ranking , *F-score* gives us an overall picture of system performance:

$$\text{F-score} = (1+\beta^2)\frac{PR}{R+\beta^2 P}$$

where P = precision and R = recall [**weighted harmonic mean**]

- Set β depending on how much we care about false negatives vs. false positives
- Conventionally $\beta = 1$, called the *F1-score*

$$\text{F1-score} = 2\frac{PR}{P+R}$$

ROC and AUC

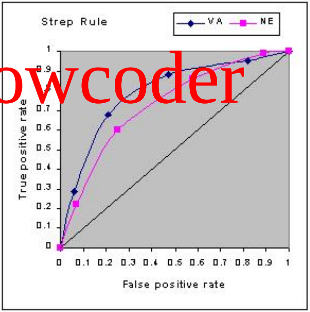
You may see people refer to AUC and ROC.

The **ROC** = Receiver Operating Characteristic

- A plot illustrating the performance of a classifier as its discrimination threshold is varied.
- Plotted in terms of True Positive Rate (Recall/Sensitivity) vs. False Positive Rate (1 − Specificity)
- The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives).

AUC = Area Under the Curve

- sometimes called **AUROC**
- equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one



Model Validation

We need some way to predict the fit of a model to a hypothetical validation set when an explicit validation set is not available.

- If we use all of our data to train a model, how do we test it?
- If we use all of our data to train a model, how can we be sure we haven't *overfit* our model to our data?

Bias and Variance in Evaluation

- The (training) *bias* of a classifier is the average distance between the expected value and the estimated value
  - Bias is large if the learning method produces classifiers that are consistently wrong.
  - Bias is small if (i) the classifiers are consistently right or (ii) different training sets cause errors on different documents or (iii) different training sets cause positive and negative errors on the same documents, but that average out to close to 0.
- The (test) *variance* of a classifier is the standard deviation between the estimated value and the average estimated value
  - Variance is large if different training sets give rise to very different classifiers.
  - It is small if the training set has a minor effect on the classification decisions made, be they correct or incorrect.
  - Variance measures how inconsistent the decisions are, not whether they are correct or incorrect.
- The *noise* in a dataset is the inherent variability of the training data
- In evaluation, we aim to minimise classifier bias and variance (but there's not a lot we can do about noise!)

Holdout

- Train a classifier over a fixed training dataset, and evaluate it over a fixed held-out test dataset
- Advantages:
  - simple to work with
  - high reproducibility
- Disadvantages:
  - trade-off between more training and more test data (variance vs. bias)
  - representativeness of the training and test data

Random Subsampling

- Perform holdout over multiple iterations, randomly selecting the training and test data (maintaining a fixed size for each dataset) on each iteration

- Evaluate by taking the average across the iterations
- Advantages:
  - reduction in variance and bias over “holdout” method
- Disadvantages:
  - reproducibility

Evaluation strategies: Leave-One-Out

Let us assume we have N data points for which we know the labels.

We choose each data point as test case and the rest as training data.

This means we have to train the system N times and the average performance is computed across the N predictions.

Good points:

- There is no sampling bias in evaluating the system and the results will be unique and repeatable.
- The method also generally gives higher accuracy values as nearly all (N -1) points are used in training.
  (It is typically the case that having more data points means a more accurate classifier can be built.)

Bad point:

- It is infeasible if we have large data set and the training is itself very expensive.

Evaluation strategies: M-fold Cross-Validation

Let us assume we have N data points for which we know the labels.

We partition the data into M (approximately) equal size partitions.

We choose each partition for testing and the remaining M-1 partitions for training.

This means we have to train the system M times and the average performance is computed across the M runs.

Typical values for M: 5 or 10 (i.e. 5-fold cross-validation, 10-fold cross-validation)

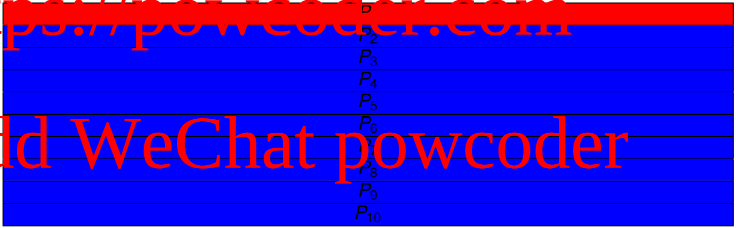
Cross Validation: Partitioning

- Split up into N equal-sized partitions P_i:

P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
P_{10}

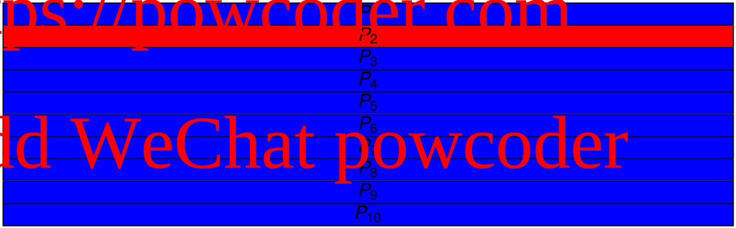
Cross Validation: Fold 1

- For each $i = 1 \ldots N$, take P_i as the test data and $\{P_j : j \neq i\}$ as the training data



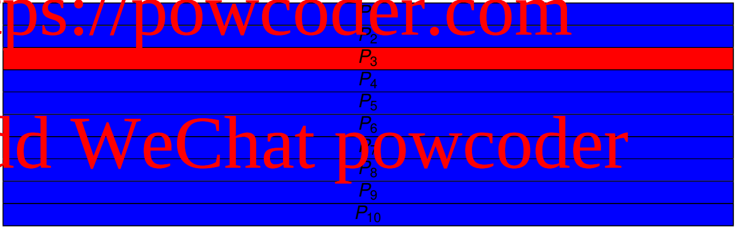
Cross Validation: Fold 2

- For each $i = 1 \ldots N$, take P_i as the test data and $\{P_j : j \neq i\}$ as the training data



Cross Validation: Fold 3

- For each $i = 1 \ldots N$, take P_i as the test data and $\{P_j : j \neq i\}$ as the training data



Cross Validation: Fold *i*

- And so on . . .

M-fold Cross-Validation Pros/Cons

Good points:

- We need to train the system only M times unlike Leave-One-Out which requires training N times.
- We can measure the stability of the system across different training/test combinations.

Bad points:

- There can be a bias in evaluating the system due to sampling, how data is distributed among the M partitions.
- The results will not be unique unless we always partition the data identically. One solution is repeat the M Fold Cross Validation by randomly shuffling the data M/2 times.
- The results will give slightly lower accuracy values as only $\frac{M-1}{M}$ of the data is used for training.
- For small data sets it is not always possible to partition the data properly such that each partition represents the data IID (Identically Independently Distributed).

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)

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

- **Method:** classify all instances according to the most common class in the training data
- The most commonly used baseline in machine learning
- Also known as *majority class* baseline
- Inappropriate if the majority class is `FALSE` and the learning task is to identify needles in the haystack
- For `weather.nominal`, zero-R class = yes

One-R (One Rule)

Creates one rule for each attribute in the training data, then selects the rule with the smallest error rate as its one rule

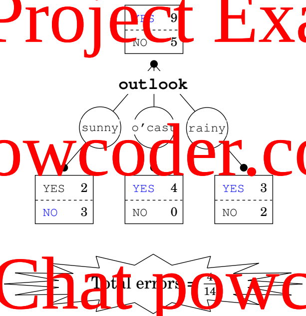
- **Method:** create a "decision stump" for each attribute, with branches for each value, and populate the leaf with the majority class at that leaf; select the decision stump which leads to the lowest error rate over the training data
- Pseudo-code:


```
For each attribute,
  For each value of the attribute, make a rule as follows:
    1 count how often each class appears
    2 find the most frequent class
    3 make the rule assign that class to this attribute-value
  Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

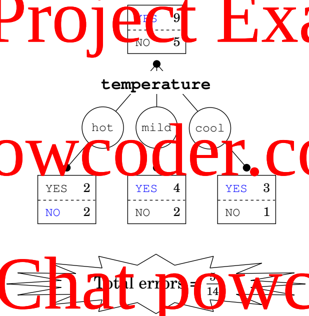
Clustering accuracy

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

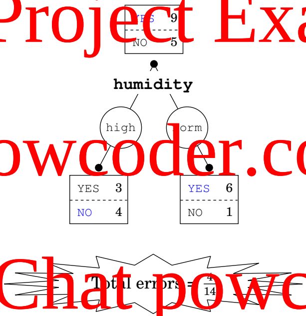
Decision Stump (`outlook`)



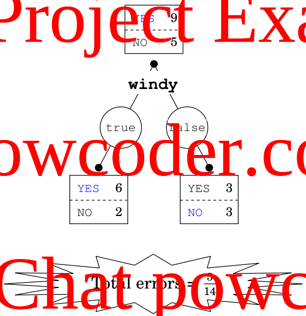
Decision Stump (temperature)



Decision Stump (`humidity`)



Decision Stump (windy)



One-R: Reflections

- Advantages:
  - simple to understand and implement
  - simple to comprehend
  - surprisingly good results
- Disadvantages:
  - unable to capture attribute interactions
  - bias towards high-arity attributes (attributes with many possible values)

Summary

- 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?

Further Reading

Evaluation in IR (unranked retrieval): Manning, Raghavan and Schtze Introduction to Information Retrieval, Cambridge University Press. 2008. **Section 8.** `http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html`

Bias/Variance tradeoff: Manning, Raghavan and Schtze Introduction to Information Retrieval, Cambridge University Press. 2008. **Section 14.6.** `http://nlp.stanford.edu/IR-book/html/htmledition/the-bias-variance-tradeoff-1.html`

ROC: Tom Fawcett, “An introduction to ROC analysis”, Pattern Recognition Letters 27 (2006) `https://ccrma.stanford.edu/workshops/mir2009/references/ROCintro.pdf`