

**COMP3308/3608, Lecture 6b**  
**ARTIFICIAL INTELLIGENCE**

**Evaluating and Comparing Classifiers**

**Reference: Witten, Frank, Hall and Pal, ch.5: p.161-194**  
**Russell and Norvig, p.695-697**

# Outline

- **Evaluating and comparing classifiers**
  - **Empirical evaluation**
    - **Measures: error rate and accuracy**
    - **Single holdout estimation, repeated holdout**
    - **Stratification**
    - **Cross-validation**
  - **Comparing classifiers – t-paired significance test**
  - **Other performance measures: recall, precision and F1**
  - **Cost-sensitive evaluation**
- **Inductive learning**

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the **University of Sydney** pursuant to Part VB of the Copyright Act 1968 (the Act).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice

# Evaluation: the Key to Success

- How to *evaluate* the performance of classifiers?
  - What performance measures to use?
  - What procedures to follow?
- How to *compare* the performance of 2 classifiers?
- Recall our goal: a classifier which *generalises well on new data*
  - i.e. classifies correctly new data, unseen during training

# Holdout Procedure

- Simple way to evaluate performance of a classifier
- Holdout procedure:
  - Split data into 2 **independent (non-overlapping)** sets: *training* and *test* (usually 2/3 and 1/3)
  - Use the training data to build the classifier
  - Use the test data to evaluate how good the classifier is
    - Calculate performance measures such as *accuracy* on test data

outlook	temp.	humidity	windy	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	73	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

training data: 9  
examples (2/3)

test data: 5  
examples (1/3)

# Accuracy and Error Rate

- **Accuracy:** proportion of **correctly** classified examples (i.e. their class is predicted correctly)
- **Error rate:** complimentary to accuracy - proportion of **incorrectly** classified examples (i.e. their class is predicted incorrectly)
- Accuracy and error rate sum to 1; typically given in % => sum to 100
- Evaluated on training and test set
  - **accuracy on training data** is overly optimistic, not a good indicator of performance on future data
  - **accuracy on test data** is the performance measure used

# Accuracy - Example

outlook	temp.	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

training data

- 1R classifier

if **outlook=sunny** then **play=no**

elseif **outlook=overcast** then **play=yes**

elseif **outlook=rainy** then **play=yes**

outlook	temp.	humidity	windy	play
sunny	hot	normal	false	no
overcast	mild	normal	false	yes
rainy	hot	normal	false	no
rainy	cool	high	true	no
sunny	mild	high	true	no

test data

Accuracy on training data=?

Accuracy on test data =?

# Validation Set

- Sometimes we need to use a 3rd set: validation set
- E.g. some classification methods (DTs, NNs) operate in two stages:
  - Stage 1: build the classifier
  - Stage 2: tune its parameters
- The test data can not be used for parameter tuning (stage 2)!  
Rule: the test data should not be used in any way to create the classifier
- Proper procedure uses 3 non-overlapping data sets
  - 1) Training set - to build the classifier
  - 2) Validation set - to tune parameters
  - 3) Test set - to evaluate accuracy
- Examples
  - DTs – training set is used to build the tree, validation set is used for pruning, test set – to evaluate performance
  - NNs – validation set is used to stop the training (to prevent overtraining)

# Making the Most of the Data

- **Generally**
  - The larger the training data, the better the classifier
  - The larger the test data, the better the accuracy estimate
- **Dilemma - ideally we want to use as much data as possible for**
  - training to get a good classifier
  - testing to get a good accuracy estimate
- **Once the evaluation is completed, all the data can be used to build the final classifier**
  - i.e. training, validation and test sets are joined together, a classifier is built using all of them for actual use (i.e. give to the customer)
  - But the accuracy of the classifier must be quoted to the customer based on test data



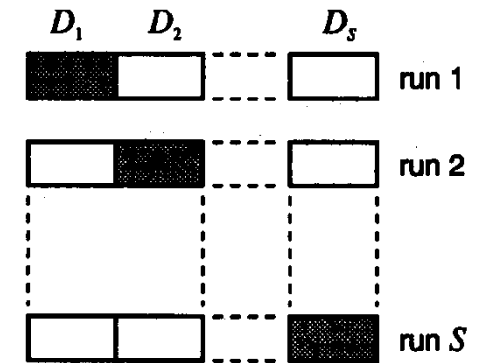
# Stratification

- An improvement of the holdout method
- The holdout method reserves a certain amount for testing and uses the reminder for training
- Problem: the examples in the training set might not be representative of all classes
  - E.g.: all examples with a certain class are missing in the training set => the classifier *cannot* learn to predict this class
- Solution: *stratification*
  - Ensures that each class is represented with approximately equal proportions in both data sets (training and testing)
- Stratification is used together with the evaluation method (e.g. holdout or cross validation)

# Repeated Holdout Method

- Holdout can be made more reliable by repeating the process
  - E.g. 10 times: In each of the 10 iteration, a certain proportion (e.g. 2/3) is randomly selected for training (possibly with stratification) and the reminder is used for testing
  - The accuracy on the different iterations are averaged to yield an overall accuracy
- This is called *repeated holdout method*
- Can be improved by ensuring that the test sets do not overlap -> *cross validation*

# Cross-Validation



from Neural Networks for Pattern Recognition,  
C. Bishop, Oxford Uni Press, 1995

- Avoids overlapping test sets
- *S-fold-cross validation:*

Step 1: Data is split into  $S$  subsets of equal size

Step 2: A classifier is built  $S$  times. Each time the testing is on 1 segment (black) and the training is on the remaining  $S-1$  segments (white)

Step 3: Average accuracies of each run to calculate overall accuracy.

- *10-fold cross-validation* – a standard method for evaluation

Split data into 10 non-overlapping sets  $set1, \dots, set10$  of approx. equal size

Run1: train on  $set1 + \dots + set9$ , test on  $set10$  and calculate accuracy ( $acc1$ )

Run2: train on  $set1 + \dots + set8 + set10$ , test on  $set9$  and calculate accuracy ( $acc2$ )

....

Run10: train on  $set2 + \dots + set10$ , test on  $set1$  and calculate accuracy ( $acc10$ )

final cross validation accuracy = average ( $acc1, acc2, \dots, acc10$ )

# More on Cross-Validation

- **Better to be used with stratification**
  - the subsets are stratified before the cross-validation is performed
  - Weka does this
- **Neither the stratification, nor the split into 10 folds needs to be exact**
  - 10 approximately equal sets, in each of which the class values are represented in approximately the right proportion
- **Even better: repeated stratified cross-validation**
  - e.g. 10-fold cross-validation is repeated 10 times and results are averaged
  - Reduces the effect of random variation in choosing the folds

# Leave-one-out Cross-Validation

- $n$ -fold cross-validation, where  $n$  is the number of examples in the data set



- How many times do we need to build the classifier?

- **Advantages**

- The greatest possible amount of data is used for training => increases the chance for building an accurate classifier
- Deterministic procedure – no random sampling is involved (no point in repeating the procedure – the same results will be obtained)

- **Disadvantage**

- high computational cost => useful for small data sets

# Comparing Classifiers

- Given two classifiers C1 and C2, which one is better on a given task?
- Step 1: Use 10-fold CV and then compare the 2 accuracies
- How much can we trust this comparison? Are the 10-fold CV estimates significantly different?
- Step 2: Run a statistical test to find if the differences are significant
  - paired t-test can be used

	C1	C2
Fold 1	95%	91%
...		
Fold 2	82%	85%
=====		
mean	91.3%	89.4%
(10-fold CV)		

Is C1 better than C2? Is this difference significant?

## Comparing Classifiers (2)

	C1	C2	
Fold 1	95%	91%	$d_1 =  95 - 91  = 4$
...			
Fold 2	82%	85%	$d_2 =  82 - 85  = 3$
=====			
mean	91.3%	89.4%	$d_{mean} = 3.5$

$$\sigma = \sqrt{\frac{\sum_i^k (d_i - d_{mean})^2}{k - 1}}$$

1. Calculate the differences  $d_i$
2. Calculate the standard deviation of the differences (an estimate of the true standard deviation)

If  $k$  is sufficiently large,  $d_i$  is normally distributed

3. Calculate the confidence interval  $Z$ :

$$Z = d_{mean} \pm t_{(1-\alpha)(k-1)} \frac{\sigma}{\sqrt{k}}$$

$t$  is obtained from a probability table

$1-\alpha$  – confidence level

$k-1$  – degree of freedom

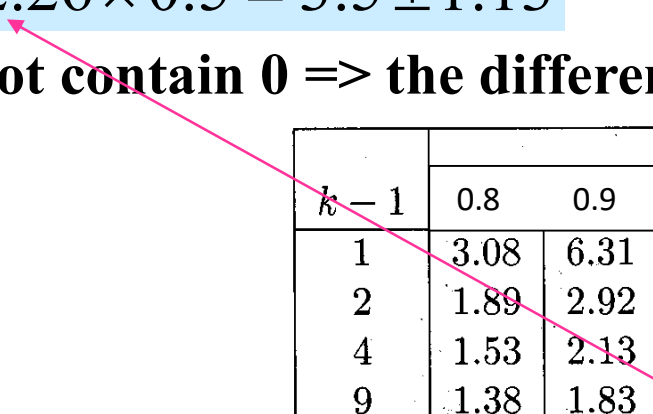
4. Interval contains 0 – difference not significant, else significant

# Comparing Classifiers - Example

Suppose that:

- We use 10 fold CV  $\Rightarrow k=10$
- $d_{mean}=3.5; \frac{\sigma}{\sqrt{k}} = 0.5$
- We are interested in significance at 95% confidence level
- Then:  $Z = 3.5 \pm 2.26 \times 0.5 = 3.5 \pm 1.13$

$\Rightarrow$  The interval does not contain 0  $\Rightarrow$  the difference is statistically significant



$k - 1$	$(1 - \alpha)$				
	0.8	0.9	0.95	0.98	0.99
1	3.08	6.31	12.7	31.8	63.7
2	1.89	2.92	4.30	6.96	9.92
4	1.53	2.13	2.78	3.75	4.60
9	1.38	1.83	2.26	2.82	3.25
14	1.34	1.76	2.14	2.62	2.98
19	1.33	1.73	2.09	2.54	2.86
24	1.32	1.71	2.06	2.49	2.80
29	1.31	1.70	2.04	2.46	2.76



# Confusion Matrix

- 2 class prediction (classes *yes* and *no*) – 4 different outcomes

- Confusion matrix:

examples	# assigned to class <i>yes</i>	# assigned to class <i>no</i>
# from class <i>yes</i>	true positives (tp)	false negatives (fn)
# from class <i>no</i>	false positives (fp)	true negatives (tn)



- How can we express *accuracy* in terms of *tp*, *fn*, *fp*, *tn*?
- Where are the correctly classified examples?
- Where are the misclassifications?
- Weka – iris data classification using 1R (3 classes); confusion matrix:
 

a	b	c	<-- classified as	
50	0	0	a = Iris-setosa	accuracy=?
0	44	6	b = Iris-versicolor	
0	3	47	c = Iris-virginica	

# Other Performance Measures: Recall, Precision, F1

- Information retrieval (IR) uses *recall (R)*, *precision (P)* and their combination *F1 measure (F1)* as performance measures

$$P = \frac{tp}{tp + fp} \quad R = \frac{tp}{tp + fn} \quad F1 = \frac{2PR}{P + R}$$

examples	# assigned to class <b>yes</b>	# assigned to class <b>no</b>
# from class <b>yes</b>	true positives (tp)	false negatives (fn)
# from class <b>no</b>	false positives (fp)	true negatives (tn)

## IR - Example

- **Blocking spam e-mails**

email	# classified as spam	# classified as not spam
# spam	24 (tp)	1 (fn)
# not spam	70 (fp)	5 (tn)

- **Spam precision** - the proportion of spam e-mails that were classified as spam, in the collection of all e-mails that were classified as spam:  
 $P = 24 / (24 + 70) = 25.5\%$ , low
- **Spam recall** - the proportion of spam e-mails that were classified as spam, in the collection of all spam e-mails:  
 $R = 24 / (24 + 1) = 96\%$ , high
- **Accuracy** –  $(24 + 5) / (24 + 1 + 70 + 5) = 29\%$ , low
- **Is this a good result?**



## R vs P - Extreme Examples

- **Extreme example 1: all e-mails are classified as spam and blocked**

email	# classified as spam	# classified as not spam
# spam	25 (tp)	0 (fn)
# not spam	75 (fp)	0 (tn)

- **Spam precision = 25%, Spam recall = 100%, Accuracy = 25%**

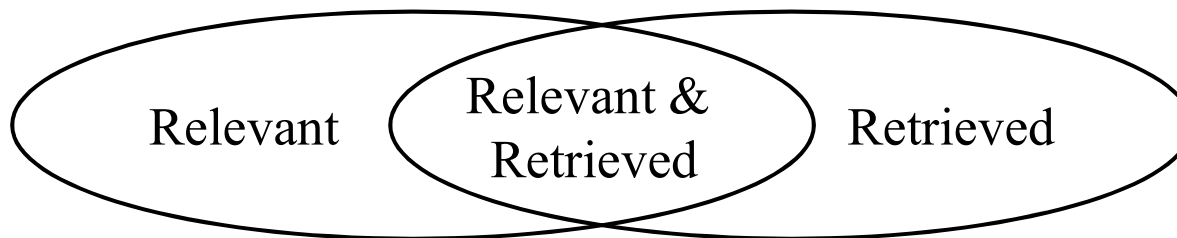
- **Extreme example 2: all but one e-mail are classified as not-spam**

email	# classified as spam	# classified as not spam
# spam	1 (tp)	79 (fn)
# not spam	0 (fp)	20 (tn)

- **Spam precision = 100%, Spam recall = 12.5%, Accuracy = 21%**
- **Trade-off between R and P - typically we can maximize one of them but not both**

## IR Example 2: Text Retrieval

- Given a query (e.g. “Cross validation”), a text retrieval system retrieves a number of documents
- *Retrieved* – the number of all retrieved documents
- *Relevant* – the number of all documents that are relevant



- Recall, Precision and F1 are used to assess the accuracy of the retrieval

$$precision = \frac{Relevant \& Retrieved}{Retrieved}$$

$$recall = \frac{Relevant \& Retrieved}{Relevant}$$

# Cost-Sensitive Evaluation

- **Misclassification may have different cost**



- **Which is higher?**

- **The cost of misclassifying a legitimate e-mail as spam and blocking it**
- **The cost of misclassifying spam e-mail as legitimate**

- **To reflect the different costs, we can calculate weighed (adjusted) accuracy, precision and F1: blocking a legitimate e-mail counts as X errors (e.g. 10, 100 , etc. – depends on the application)**

- **Other examples where the cost of different errors is different (most applications)**

- **Loan approval: the cost of lending to a defaulter > cost of refusing a loan to a non-defaulter**
- **Oil spills detection: the cost of failing to detect oil spills > false alarms**
- **Promotional mail: the cost of sending junk mail to a customer who doesn't respond < cost of sending mail to a customer who would respond**

# Inductive Learning

- Actually, why do we need to do empirical evaluation?
- Supervised learning is *inductive* learning
- Induction: inducing the universal from the particular  
*All apples I have seen are red.  $\Rightarrow$  All apples are red.*
- Given a set of examples  $(\mathbf{x}, f(\mathbf{x}))$ ,
  - $\mathbf{x}$  is the input vector
  - $f(\mathbf{x})$  is the output of a function  $f$  applied to  $\mathbf{x}$
  - we don't know what  $f(\mathbf{x})$  is
- Find: a function  $h$  (hypothesis) that is a good approximation of  $f$  (R&N, p.651)
- Ex.: Given:  $([1,1],2), ([2,1],3), ([3,1],4)$ , find  $h \approx f$

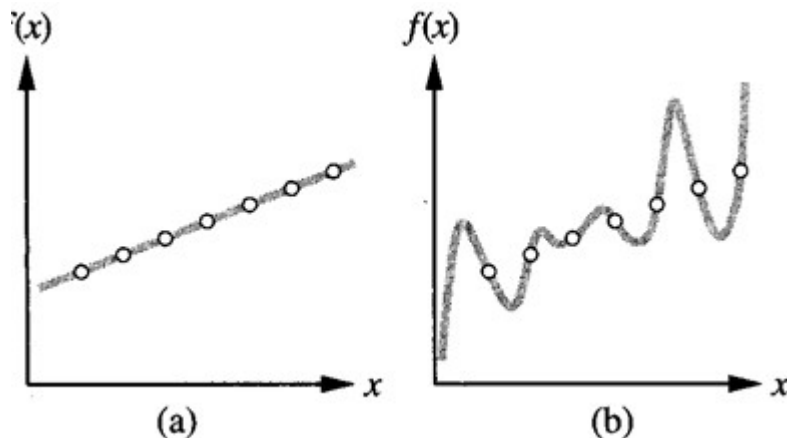
# Inductive Learning - Difficulty

- We can generate many hypotheses  $h$ 
  - the set of all possible hypotheses  $h$  form the hypothesis space  $H$
- How to tell if a particular  $h$  is a good approximation of  $f$ ?
  - a good  $h$  will **generalize** well, i.e. will predict new examples correctly
  - That's why we measure its performance on new examples (testing set)
- A good  $h$  does not necessary imply fitting the given samples  $x$  perfectly - we want to extract patterns not to memorize the given data



# Which Consistent Hypothesis is Best?

- **Given:** a set of 7 examples  $(x, f(x))$ ,  $x$  and  $f(x)$  are real numbers
- **Task:** find  $h \approx f$  such as
  - $h$  is a function of a single variable  $x$
  - the hypothesis space is the set of polynomials of degree at most  $k$ , e.g.  $2x+3$  – a degree-1 polynomial;  $6x^2+3x+1$  – degree 2
- Consider these 2 solutions:



fit with a line  
(degree-1)

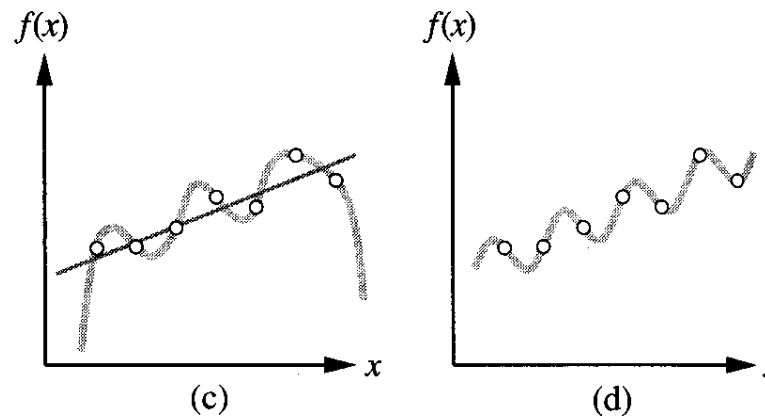
fit with a degree-7 polynomial

- Both hypotheses are *consistent* with training data (agree with it, fit the examples perfectly)
- Which one is better? How do we choose from several consistent hypotheses?

# Multiple Consistent Hypotheses

- Recall that we'd like to *extract pattern* from data
- Ockham razor: prefer the simplest hypothesis consistent with training data
  - It is also simpler to compute
- How to define simplicity? Not easy in general.
  - Easy in this case - obviously degree-1 polynomial (line) is simpler than a higher degree polynomial

# Importance of Hypotheses Space



fit with a degree-6  
Polynomial  $ax+b+c$

fit with  $\sin x$

- (d) shows the true function:  $\sin(x)$
- In (c) we are searching the hypothesis space  $H$  consisting of polynomial functions –  $\sin$  is not there

- (c): a degree-1 polynomial function (line) cannot perfectly fit data
- (c): a degree-6 polynomial perfectly fits data but is this a good choice?



- How many parameters? Does it seem to find any pattern in data?

- The choice of hypotheses space is important
  - *The  $\sin$  function cannot be learnt accurately using polynomials*
- A learning problem is *realizable* if  $H$  contains the true function
- In practice we can't tell if the problem is realizable because we don't know the true hypothesis! (we are trying to learn it from examples)

# Empirical Evaluation

- **This is the best thing we can do**
- **Empirical evaluation - review**
  - **Examples used to build the model are called *training set***
  - **Success (how good the fit is) is typically measured on fresh data for which class labels are known (*test data*) as the proportion of correctly classified examples (*accuracy*)**