Lecture 14: Evaluation

COMP90049
Knowledge Technologies

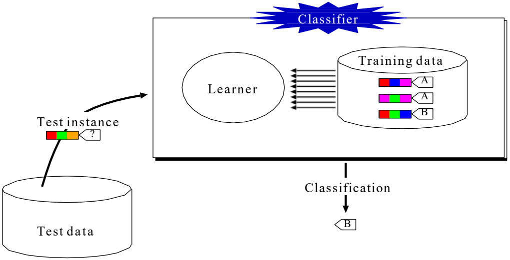
Hasti Samadi & Sarah Erfani & Karin Verspoor, CIS

Semester 2, 2019

THE UNIVERSITY OF MELBOURNE

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



What is a good Classifier?

- A good Classifier (in Supervised ML Framework)?

What is a good Classifier?

- A good Classifier (in Supervised ML Framework)?
  - Make correct predictions

What is a good Classifier?

- A good Classifier (in Supervised ML Framework)?
  - Make correct predictions

ID	*Outl*	*Temp*	*Humi*	*Wind*	PLAY	
TRAINING INSTANCES						
A	s	h	h	F	N	
B	s	h	h	T	N	
C	o	h	h	F	Y	
D	r	m	h	F	Y	
E	r	c	n	F	Y	
F	r	c	n	T	N	
TEST INSTANCES						
G	o	c	n	T	?	Y
H	s	m	h	F	?	N

What is a good Classifier?

- A good Classifier (in Supervised ML Framework)?
  - Make correct predictions

- The basic evaluation metric: **Accuracy**

$$\text{Accuracy} = \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}}$$

- Quantifies how frequently the classifier is correct in predicting labels, with respect to a fixed dataset with known labels

What is a good Classifier?

- Main idea:
  - Train (build classifier) using **training data**
  - Test (evaluate classifier) using **test data**
    - For instances in the test data, compare predicted class label with actual class label
- But often, we just have **data** – a collection of instances

Train vs Test Data

- The simplistic (and wrong!) idea can be:
  - Use all of the instances as training data
  - Build the classifier using all of the instances
  - Use all of the (same) instances as test data
  - Evaluate the classifier using all of the instances
- "Testing on the training data" tends **over-estimate** classifier performance.
- Effectively, we are telling the classifier what the correct answers are, and then checking 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 classifier using (only) the training instances
  - Evaluate the classifier 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 reproducibility
- **Disadvantages**:
  - size of the split affects estimate of the classifier'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)

Random Subsampling

Repeated Random Subsampling is slower, but somewhat better solution:

- 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

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)

m-Fold Cross Validation

Cross Validation is usually the preferred alternative:

- 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: Partitioning

Split up into *10* equal-sized partitions P_i :

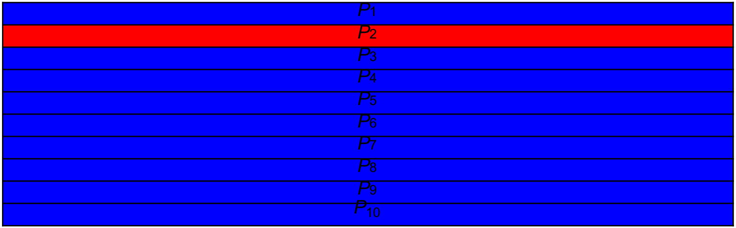
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
P_{10}

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

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

Cross Validation: Fold *i*

And so on ...

Cross Validation - Advantages

Why is this better than Holdout / 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
- Takes roughly the same amount of time as Repeated Random Subsampling
- Very reproducible
- Can be shown to minimise **bias** and **variance** of our estimates of the classifier's performance

Cross Validation – selecting *m*

- Number of folds directly impacts runtime and size of datasets:
  - Fewer folds: more instances per partition, *higher variance*
  - More folds: fewer instances per partition, *lower variance* but slower
- Most common choice of **m: 10** (occasionally, 5)
  - Mimics 90–10 Holdout, but far more reliable

Cross Validation – selecting *m*

- Best choice: m=N, the number of instances (also 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”)
  - Far too slow to use in practice

What is a good Classifier?

- The basic evaluation metric: **Accuracy**

$$\text{Accuracy} = \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}}$$

- Quantifies how frequently the classifier is correct in predicting labels, with respect to a fixed dataset with known labels

What is a good Classifier?

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	no	no
rainy	mild	high	TRUE	no	yes

What is a good Classifier?

4 test instances; 2 correct predictions, 2 incorrect predictions

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

Confusion Matrix

- Confusion matrix for Binary classification

		Prediction	
		Yes	No
Actual	Yes	TP	FN
	No	FP	TN

Confusion Matrix

- Confusion matrix for Binary classification

		Prediction	
		Yes	No
Actual	Yes	TP	FN
	No	FP	TN

sunny	mild	normal	TRUE	yes	no	FN
overcast	mild	high	TRUE	yes	yes	TP
overcast	hot	normal	FALSE	no	no	TN
rainy	mild	high	TRUE	no	yes	FP

Accuracy & Error Rate

- **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**:

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

- It is clear that :

$$\mathrm{ER} = 1 - \mathrm{ACC}$$

Precision & Recall

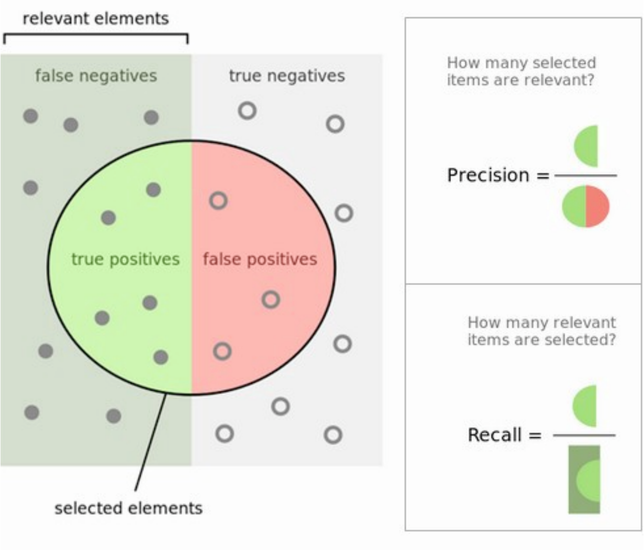
- **Precision:** How often are we correct, when we predict that an instance is positive

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

- **Recall:** What proportion of the Actually positive instances have we correctly identified?

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

Precision & Recall



Precision & 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**:

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

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

Multi-Class Confusion Matrix

		Prediction		
		A	B	C
Actual	A	TP	FN	FN
	B	FP	TN	TN
	C	FP	TN	TN

Confusion matrix for Class A

Multi-Class Confusion Matrix

		Prediction		
		A	B	C
Actual	A	TN	FP	TN
	B	FN	TP	FN
	C	TN	FP	TN

Confusion matrix for Class B

Multi-Class Confusion Matrix

		Prediction		
		A	B	C
Actual	A	TN	TN	FP
	B	TN	TN	FP
	C	FN	FN	TP

Confusion matrix for Class C

Multi-Class Confusion Matrix

Since Precision, Recall and F-score are calculated **per-class**, for evaluating the whole system we need to aggregate the results:

- **Macro-Averaging:**
    - calculate P, R per class and then take mean

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

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

Multi-Class Confusion Matrix

- **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 Confusion Matrix

- **Weighted Mean:**
  - calculate P, R per class and then take weighted mean, based on the proportion of instances in that class

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

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

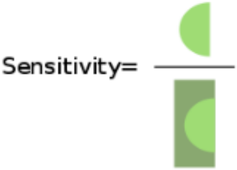
Sensitivity and Specificity

- **Sensitivity (True Positive Rate)**
  proportion of actual positives that are correctly identified

Sensitivity and Specificity

- **Sensitivity (True Positive Rate)**
  proportion of actual positives that are correctly identified → Recall

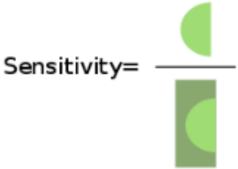
$$Sensitivity = Recall = \frac{TP}{TP + FN}$$



Sensitivity and Specificity

- **Sensitivity (True Positive Rate)**
  proportion of actual positives that are correctly identified → Recall

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

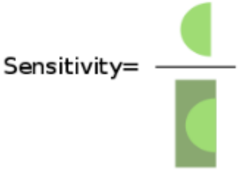


- **Specificity (True Negative Rate)**
  proportion of actual negatives that are correctly identified

Sensitivity and Specificity

- **Sensitivity (True Positive Rate)**
  proportion of actual positives that are correctly identified → Recall

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

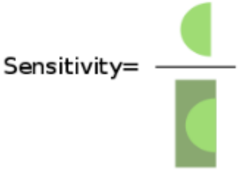


- **Specificity (True <u>Negative</u> Rate)**
  proportion of actual <u>negatives</u> that are correctly identified

Sensitivity and Specificity

- **Sensitivity (True Positive Rate)**
  proportion of actual positives that are correctly identified → Recall

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



- **Specificity (True <u>Negative</u> Rate)**
  proportion of actual <u>negatives</u> that are correctly identified

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

Sensitivity and Specificity

- **Sensitivity (True Positive Rate)**
  proportion of actual positives that are correctly identified → Recall

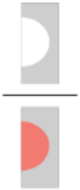
$$Sensitivity = Recall = \frac{TP}{TP + FN}$$

Sensitivity=

- **Specificity (True Negative Rate)**
  proportion of actual negatives that are correctly identified

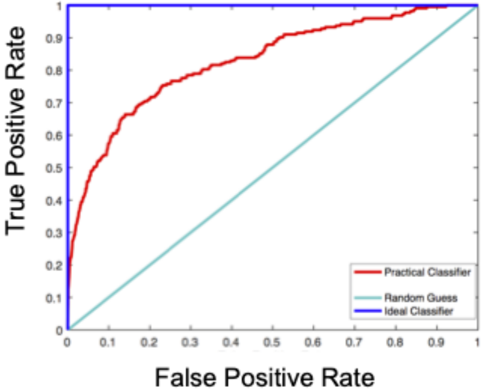
$$Specificity = \frac{TN}{TN + FP}$$

Specificity =



ROC and AUC

- **ROC** (Receiver Operating Characteristic) curve is an evaluation metric that typically used in binary classification.
- ROC curve illustrates the performance of a classifier as its discrimination threshold changes.
- y-axis represents the **True Positive Rate** ($\frac{TP}{TP+FP}$)
- x-axis represents the **False Positive Rate** ($\frac{FP}{FN+TN}$)

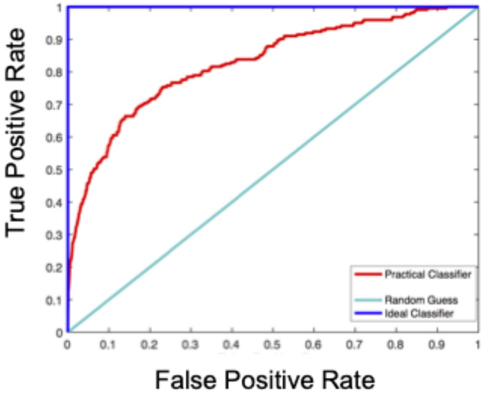


ROC and AUC

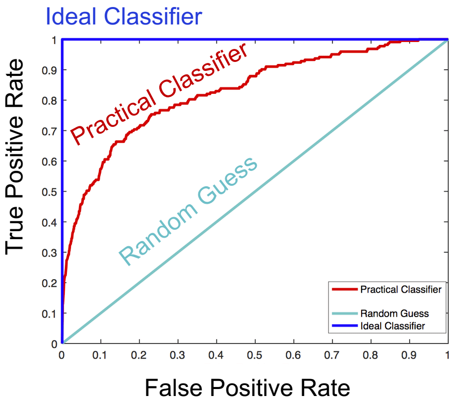
- **ROC** (Receiver Operating Characteristic) curve is an evaluation metric that typically used in binary classification.
- ROC curve illustrates the performance of a classifier as its discrimination threshold changes.
- y-axis represents the **True Positive Rate** ($\frac{TP}{TP+FP}$)

  sensitivity
- x-axis represents the **False Positive Rate** ($\frac{FP}{FN+TN}$)

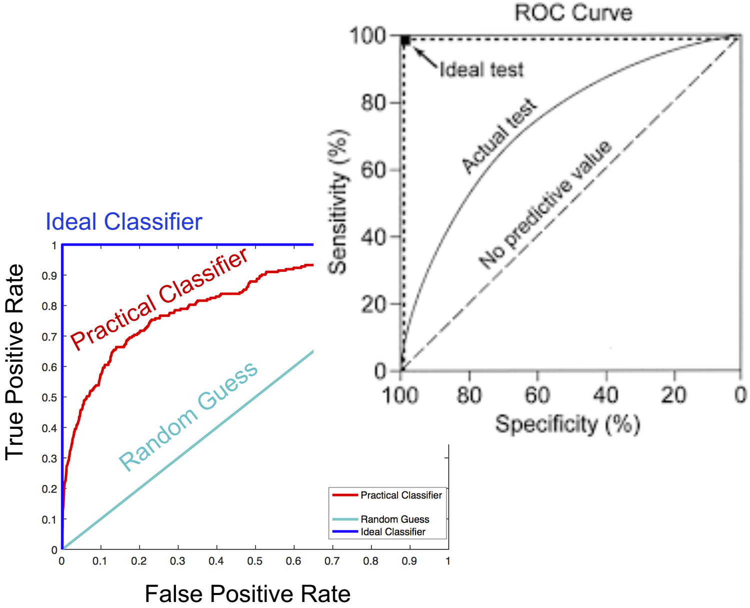
  1- specificity



ROC and AUC

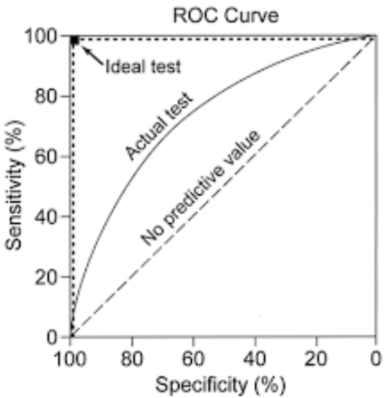

Ideal Classifier
Practical Classifier
Random Guess
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Practical Classifier
Random Guess
Ideal Classifier

ROC and AUC


Ideal Classifier
Practical Classifier
Random Guess
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Practical Classifier
Random Guess
Ideal Classifier
ROC Curve
Ideal test
Actual test
No predictive value
Sensitivity (%)
Specificity (%)
100
80
60
40
20
0

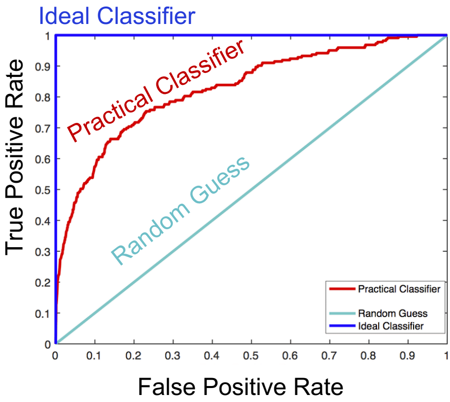
ROC and AUC

- The ideal prediction would yield a point in the upper left corner of the ROC space, representing **100% sensitivity** (no false negatives) and **100% specificity** (no false positives).



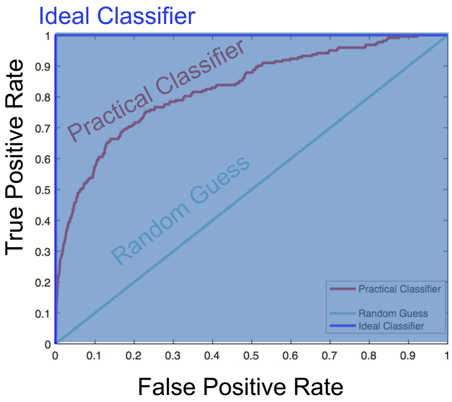
AUC = Area Under the Curve

- represents degree or measure of separability.
- how much model is capable of distinguishing between classes.



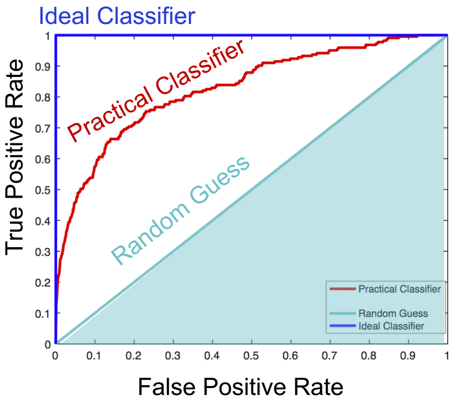
AUC = Area Under the Curve

- represents degree or measure of separability.
- how much model is capable of distinguishing between classes.



AUC = Area Under the Curve

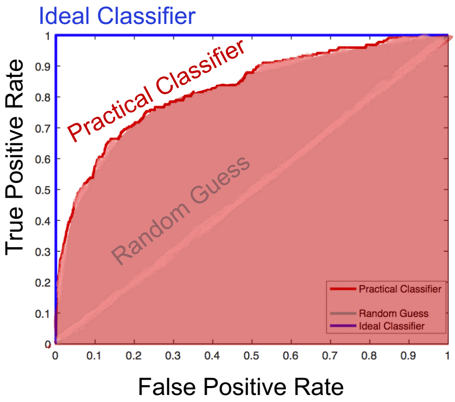
- represents degree or measure of separability.
- how much model is capable of distinguishing between classes.



ROC and AUC

AUC = Area Under the Curve

- represents degree or measure of separability.
- how much model is capable of distinguishing between classes.

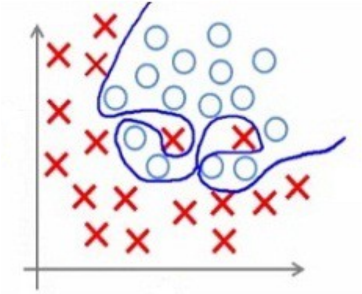


Tensions in Classification

- **Generalization**: 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 idiosyncrasies of the training data rather than learning its generalisable properties?
- **Consistency:** is the classifier able to flawlessly predict the class of all training instances?

Over-fitting

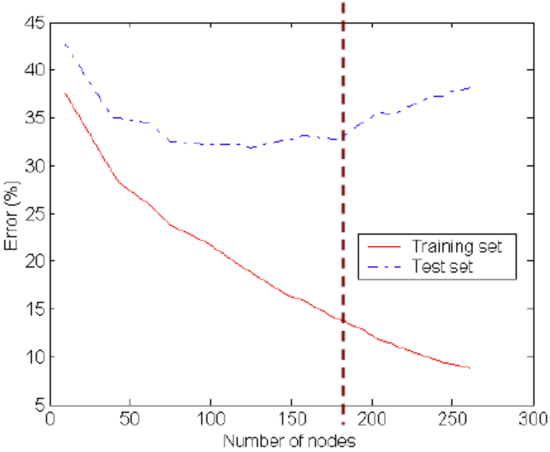
- A model that fits the training data too well can have poorer generalisation than a model with higher training error.



Over-fitting

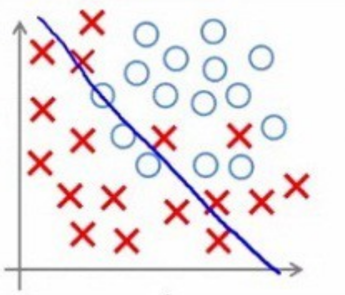
Over-fitting

- Possible evidence of overfitting: large gap between training and test performance



Under-fitting

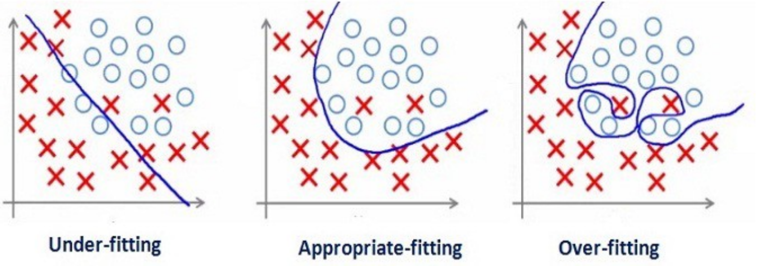
- A model that fits neither the training data nor the training data. The model is too simplistic and does not follow the pattern in the data.



Under-fitting

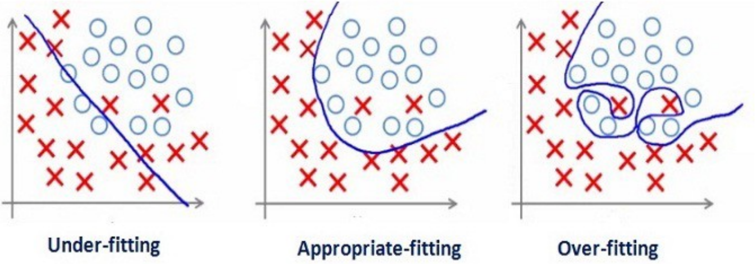
Over-fitting vs Under-fitting

- **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.



Over-fitting vs Under-fitting

- **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.



- Model complexity is a major factor that influences the ability of the model to generalize

Bias & Variance

- The other two important factors are:
    - **(training) Bias**
    - **Variance**

Bias & Variance

- **(training) Bias**

  The tendency of our classifier to make systematically wrong predictions

- **Variance**

Bias & Variance

- **(training) Bias**

  The tendency of our classifier to make systematically wrong predictions

  - Average distance between the expected value and the estimated value

$$\mathrm{Bias}(\hat{\theta};\theta) = \mathrm{E}_x[\hat{\theta}(x) - \theta(x)]$$

- **Variance**

- **(training) Bias**

  The tendency of our classifier to make systematically wrong predictions

  - Average distance between the expected value and the estimated value

$$\mathrm{Bias}(\hat{\theta};\theta) = \mathrm{E}_x[\hat{\theta}(x) - \theta(x)]$$

- **Variance**

  The tendency of different training sets to produce different models/predictions

Bias & Variance

- **(training) Bias**

  The propensity of our classifier to make systematically wrong predictions

  - Average distance between the expected value and the estimated value

$$\mathrm{Bias}(\hat{\theta};\theta) = \mathrm{E}_x[\hat{\theta}(x) - \theta(x)]$$

- **Variance**

  The propensity of different training sets to produce different models/predictions

  - Standard deviation between the estimated value and the average estimated value

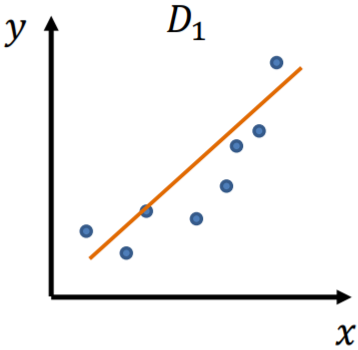
$$\mathrm{Var}(\hat{\theta};\theta) = \mathrm{E}_x[\hat{\theta}(x)^2] - \mathrm{E}_x[\hat{\theta}(x)]^2$$

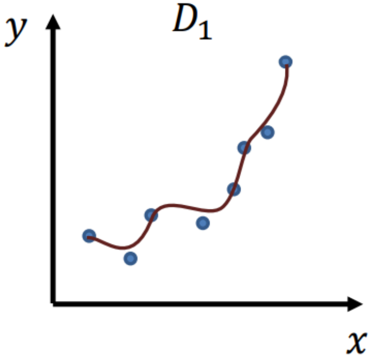
Bias & Variance

- **(training) Bias**
  - <u>Bias is large if</u>

    the learning method produces classifiers that are consistently wrong.
  - <u>Bias is small if</u>
    1. the predictions are consistently right or
    2. different training sets cause positive and negative errors on the same documents, but that average out to close to 0.
- We can have unbiased systems with very poor performance; or a biased system with relatively strong performance. (Bias is usually a secondary evaluation metric)

Bias & Variance

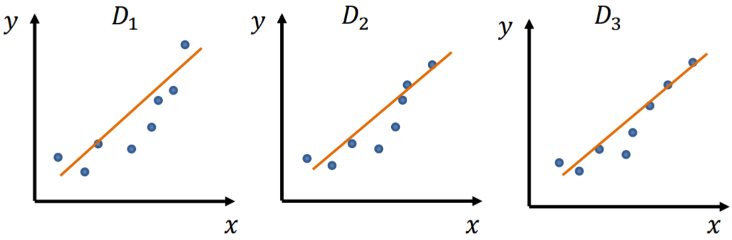




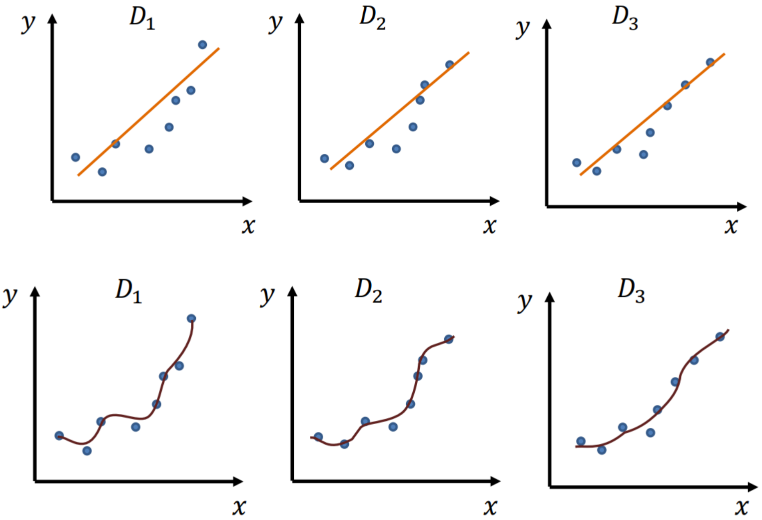
Bias & Variance

- **Variance**
  - Variance is large if
    - different training sets lead to very different predictions for the test dataset
  - Variance 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.

Bias & Variance



Bias & Variance



Bias & Variance

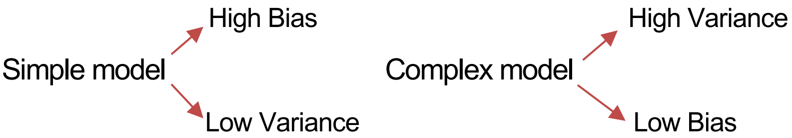
- There is always a **trade-off** between Bias and Variance

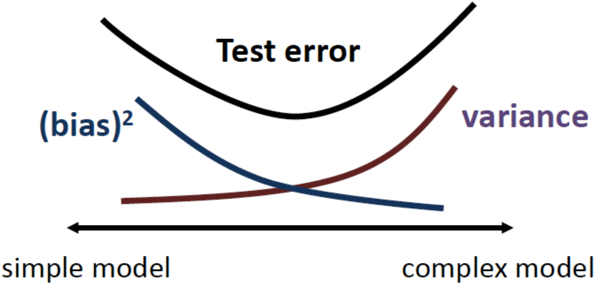
Simple model → High Bias
Simple model → Low Variance

Complex model → High Variance
Complex model → Low Bias

Bias & Variance

- There is always a **trade-off** between Bias and Variance





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 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 c_k to each test instance, weighting the class assignment according to $P(c_k)$

- Assumes we know the class prior probabilities
- Reduce 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

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

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

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

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

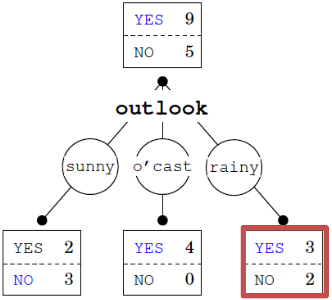
- **Method:** create a "decision stump" for each attribute, with branches for each value, and populate the leaf with the majority class at that leaf; select the decision stump which leads to the lowest error rate over the training data

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)

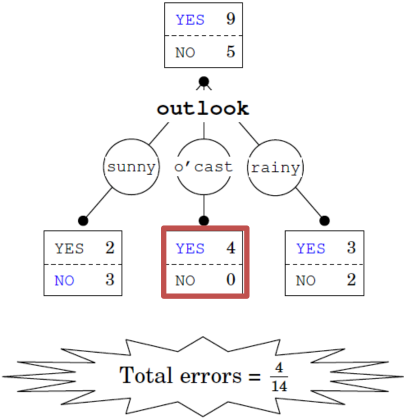
Outlook	Play
sunny	no
sunny	no
overcast	yes
rainy	yes
rainy	yes
rainy	no
overcast	yes
sunny	no
sunny	yes
rainy	yes
sunny	yes
overcast	yes
overcast	yes
rainy	no



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

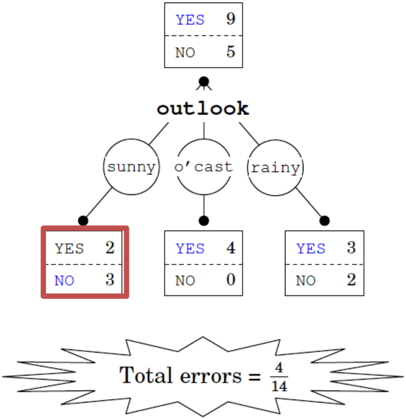
Decision Stump (outlook)

Outlook	Play
sunny	no
sunny	no
overcast	yes
rainy	yes
rainy	yes
rainy	no
overcast	yes
sunny	no
sunny	yes
rainy	yes
sunny	yes
overcast	yes
overcast	yes
rainy	no

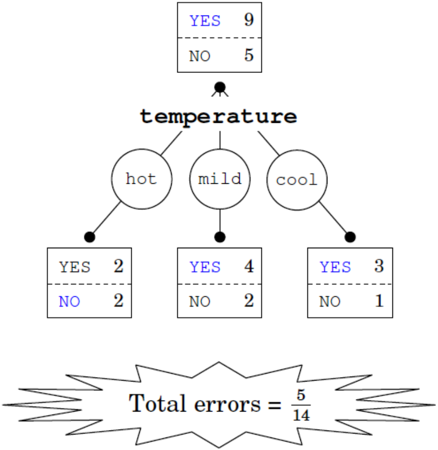


Decision Stump (outlook)

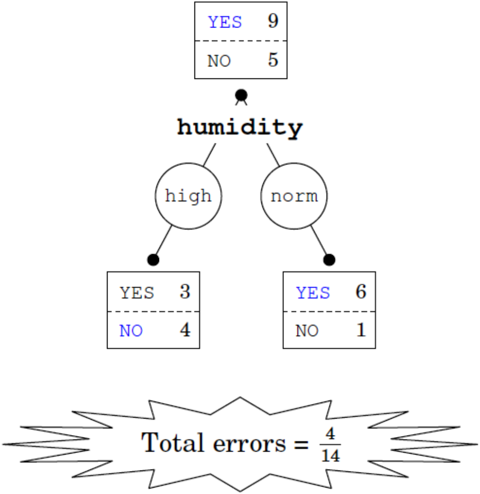
Outlook	Play
sunny	no
sunny	no
overcast	yes
rainy	yes
rainy	yes
rainy	no
overcast	yes
sunny	no
sunny	yes
rainy	yes
sunny	yes
overcast	yes
overcast	yes
rainy	no



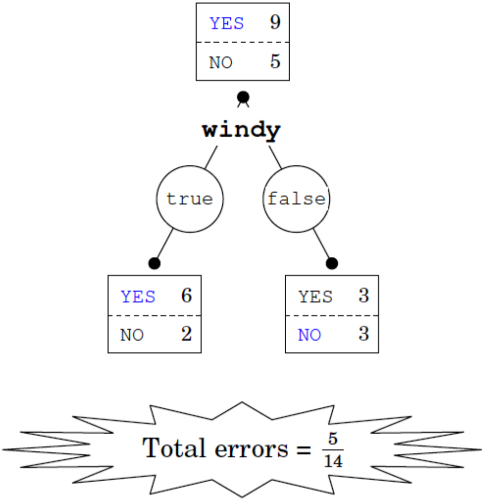
Decision Stump (temperature)



Decision Stump (humidity)



Decision Stump (windy)



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 rules

Choose the attribute whose rules produce the smallest error rate

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