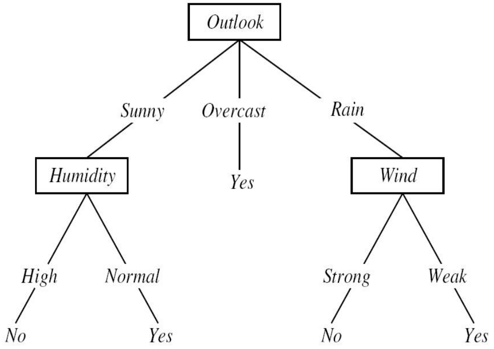
Back to the future:
Classification Trees Revisited
(Forests, Ferns and Cascades)



Toby Breckon
School of Engineering
Cranfield University

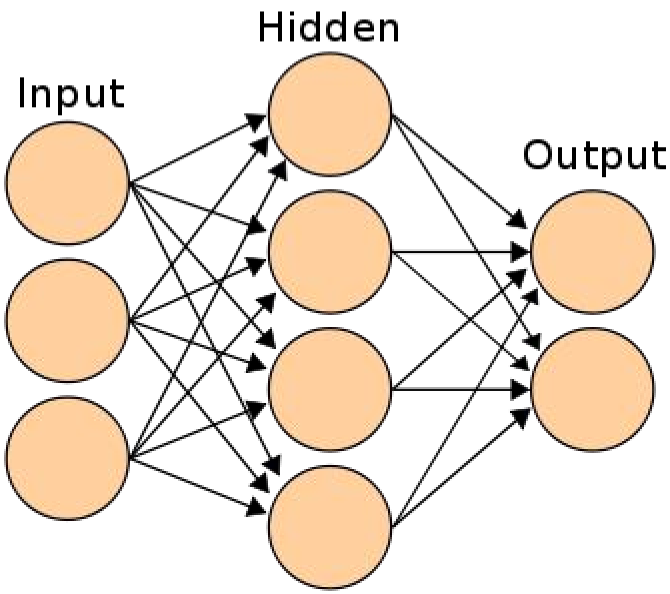
www.cranfield.ac.uk/~toby.breckon/mltutorial/
toby.breckon@cranfield.ac.uk

9th October 2013 - NATO SET-163 / RTG-90
Defence Science Technology Laboratory, Porton Down, UK

Cranfield UNIVERSITY

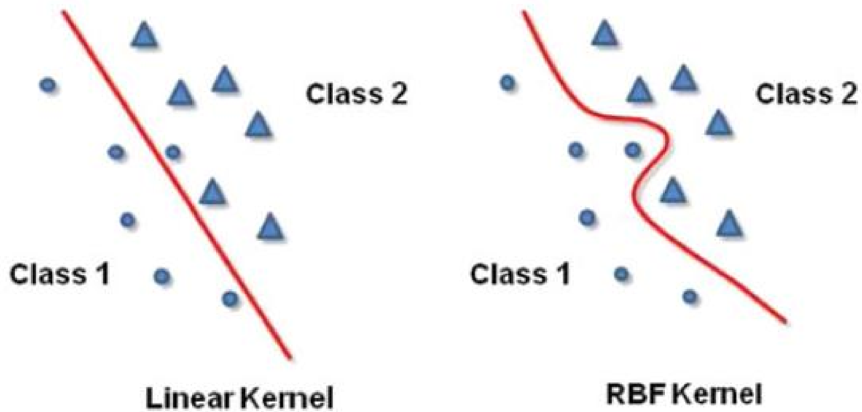
Neural Vs. Kernel

- Neural Network



  - over-fitting
  - complexity Vs. traceability

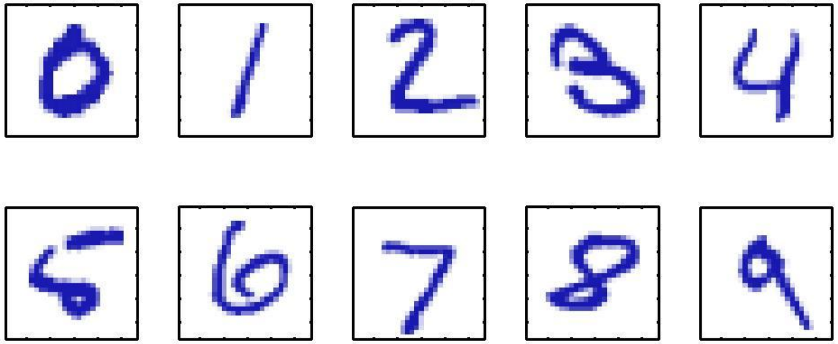
- Support Vector Machine



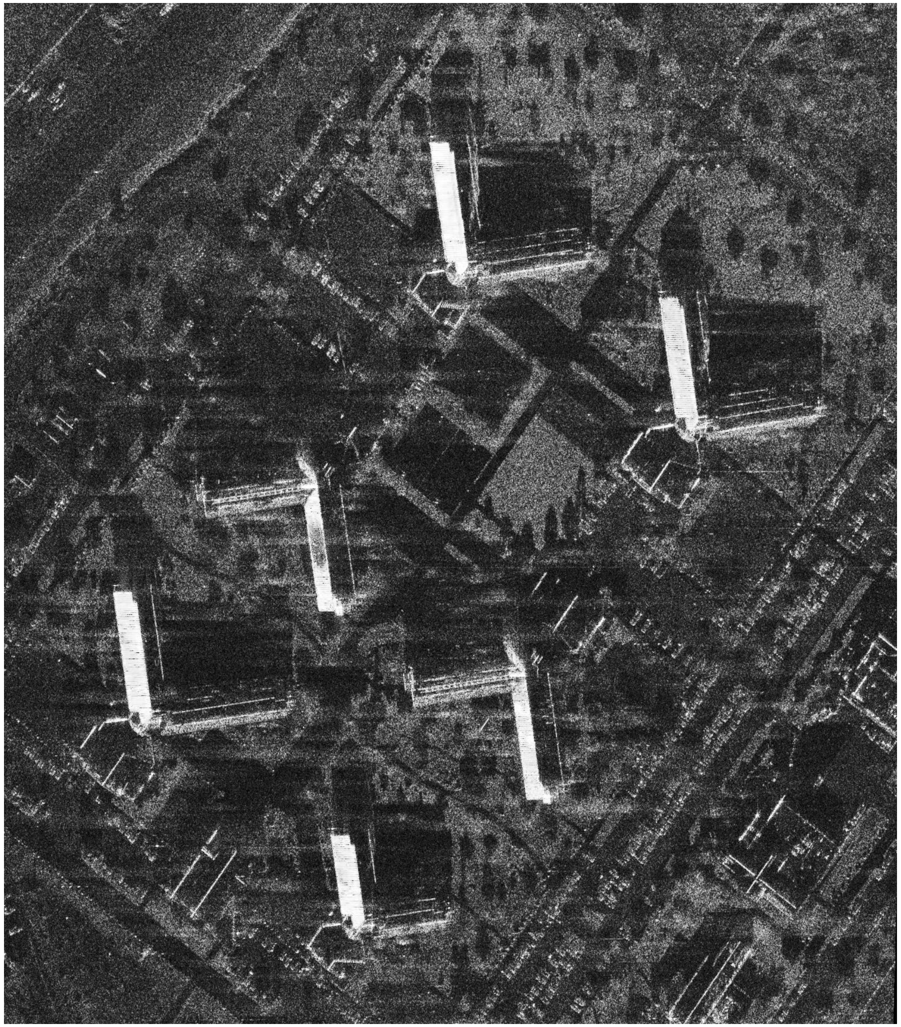
  - kernel choice
  - training complexity

Well-suited to classical problems ….

[Fisher/Brekcon et al. 2013]



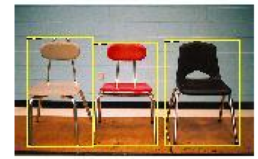
[Bishop 2006]



Cranfield UNIVERSITY

Common ML Sensing Tasks ...






- Object **Classification**
  - what object ?

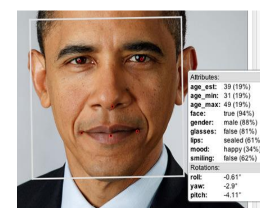
http://pascallin.ecs.soton.ac.uk/challenges/VOC/

- Object **Detection**
  - object or no-object ?

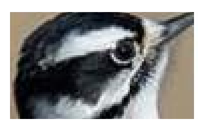
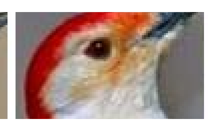
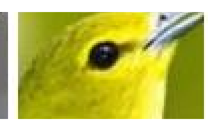
{people | vehicle | … intruder ….}

- Instance **Recognition** ?
  - who (or what) is it ?


{face | vehicle plate| gait …. → biometrics}

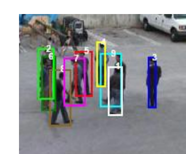


- **Sub-category** analysis
  - which object type ?

{gender | type | species | age …...}

- Sequence **{ Recognition | Classification }** ?
  - what is happening / occurring ?

… in the big picture

Features representation and/or **raw sensor samples ….**

Machine Learning = "Decision or Prediction"

person

building

tank

cattle

….

….

….

car

plane

….

etc.

A simple learning example ....

- Learn prediction of **"Safe conditions to fly ?"**
  - based on the weather conditions = *attributes*
  - classification problem, *class = {yes, no}*

Attributes / Features

Classification

Outlook	Temperature	Humidity	Windy	Fly
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
…	…	…	…	…

Decision Tree *Recap*

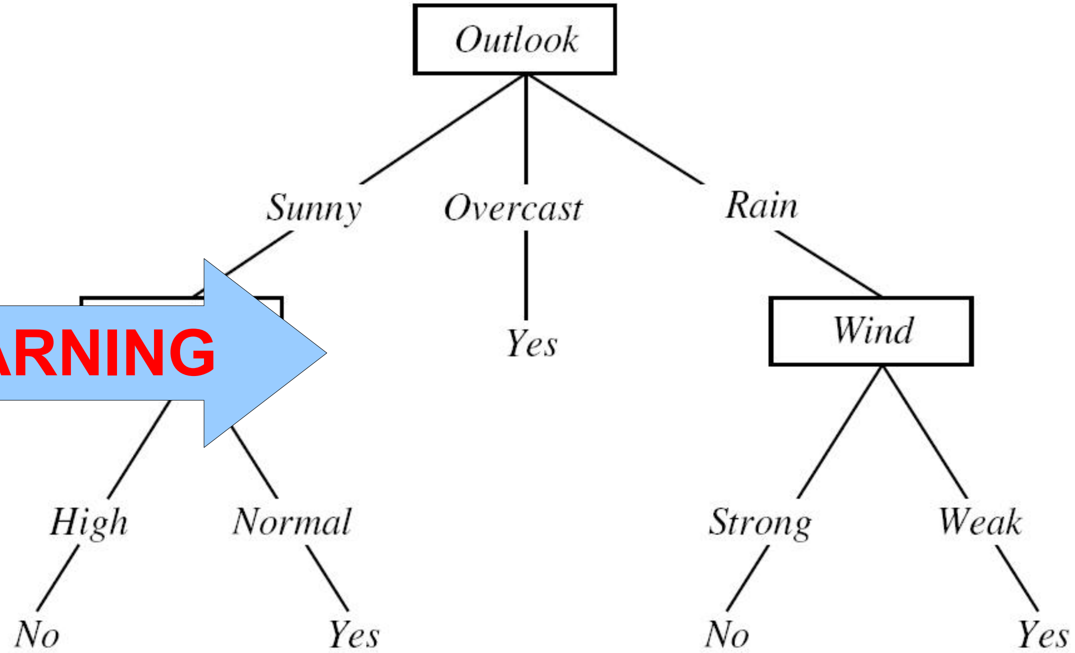
- Set of *Specific* **Examples** ...

Day	Outlook	Temperature	Humidity	Wind	**Fly**
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	[illegible]
D12	Overcast	Mild	High	Strong	[illegible]
D13	Overcast	Hot	Normal	Weak	[illegible]
D14	Rain	Mild	High	Strong	No

(training data)

LEARNING

Safe conditions to fly ?



GENERALIZED RULE

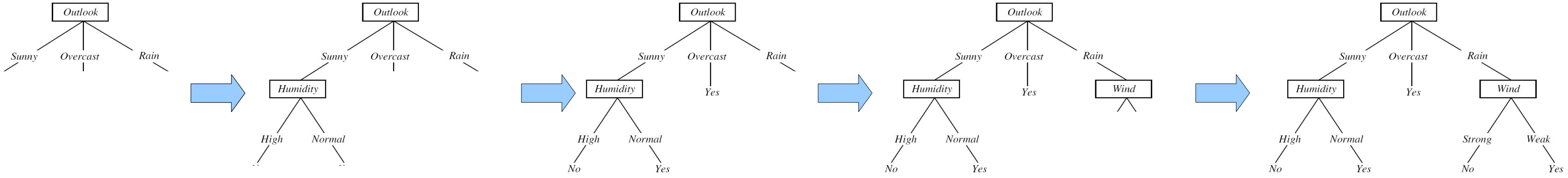
Cranfield UNIVERSITY

Growing Decision Trees

- **Construction** is carried out **top down** based on node splits that **maximise the reduction in the entropy** in each resulting sub-branch of the tree

[Quinlan, '86]

- **Key Algorithmic Steps**
  1. **Calculate** the **information gain** of splitting on each attribute
     (i.e. reduction in entropy (variance))
  2. **Select** attribute with **maximum information gain** to be a new node
  3. **Split** the **training data** based on this attribute



  4. Repeat recursively (step 1 → 3) for each sub-node until all

Cranfield UNIVERSITY

Extension : *Continuous Valued Attributes*

- Create a discrete attribute to test continuous attributes
  - chosen threshold that gives greatest **information gain**

$$Temperature = 82.5$$

$$(Temperature > 72.3) = t, f$$

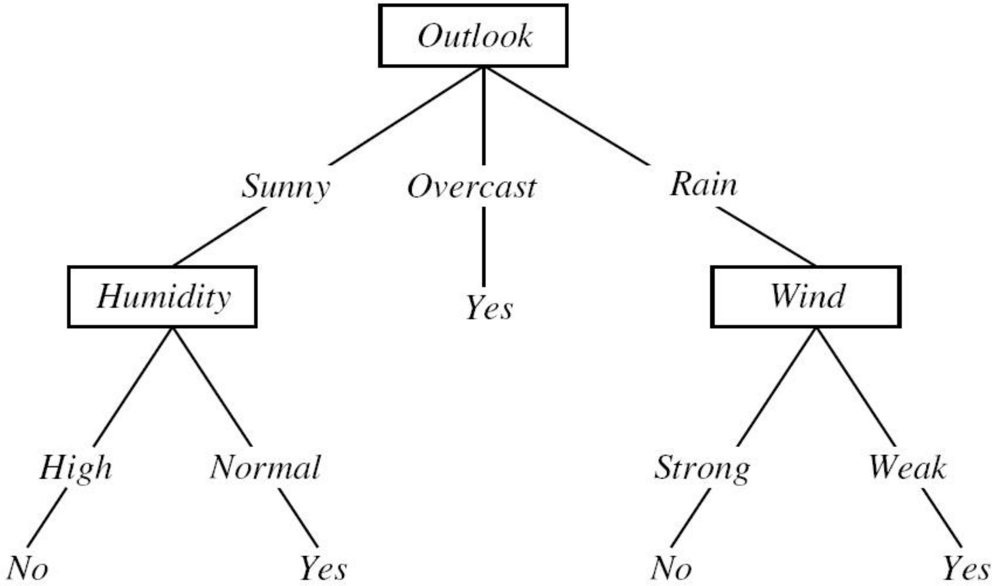
Temperature	40	48	60	72	80	90
Fly	No	No	Yes	Yes	Yes	No

Cranfield UNIVERSITY

Problem of **Overfitting**

- Consider adding **noisy** training example #15:
  - [Sunny, Hot, Normal, Strong, Fly=Yes] (WRONG LABEL)

- What training effect would it have on earlier tree?



Cranfield UNIVERSITY

Problem of **Overfitting**

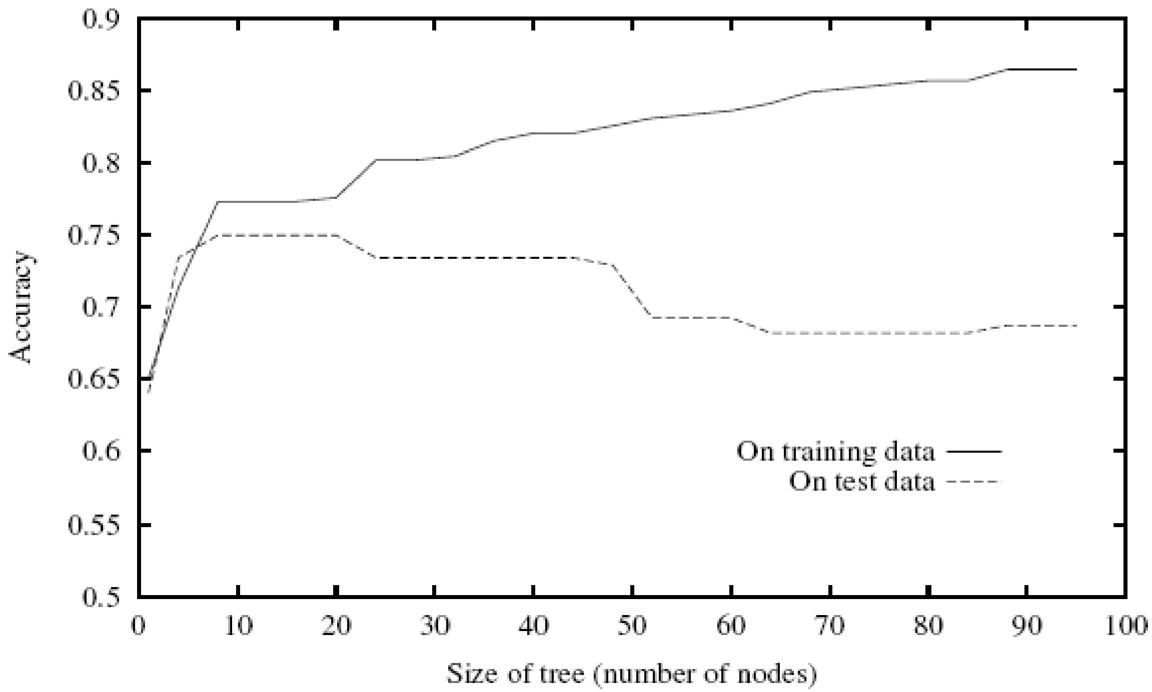
- Consider adding **noisy** training example #15:
  - [Sunny, Hot, Normal, Strong, **Fly=Yes**]

= wind!

- What effect on earlier decision tree?
  - error in example = **error in tree construction** !

Overfitting in general

- **Performance** on the **training** data (with noise) **improves**
- **Performance** on the unseen **test** data **decreases**

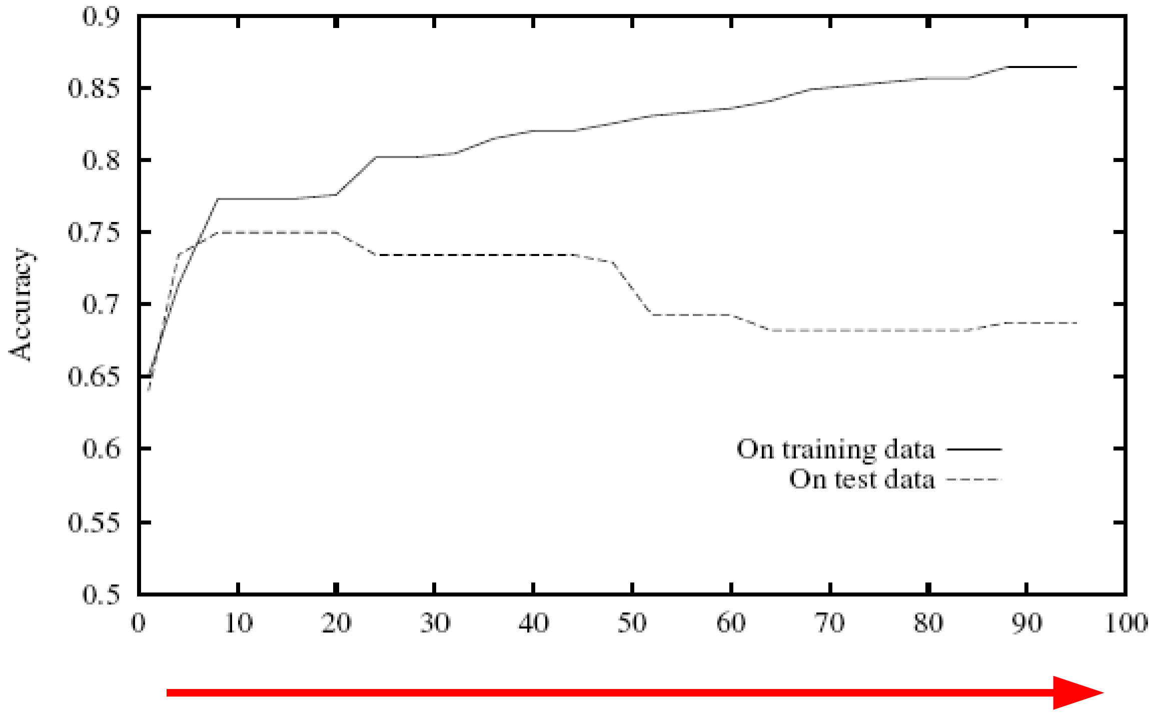


  – **For decision trees: tree complexity increases, learns training data too well! (over-fits)**

Cranfield UNIVERSITY

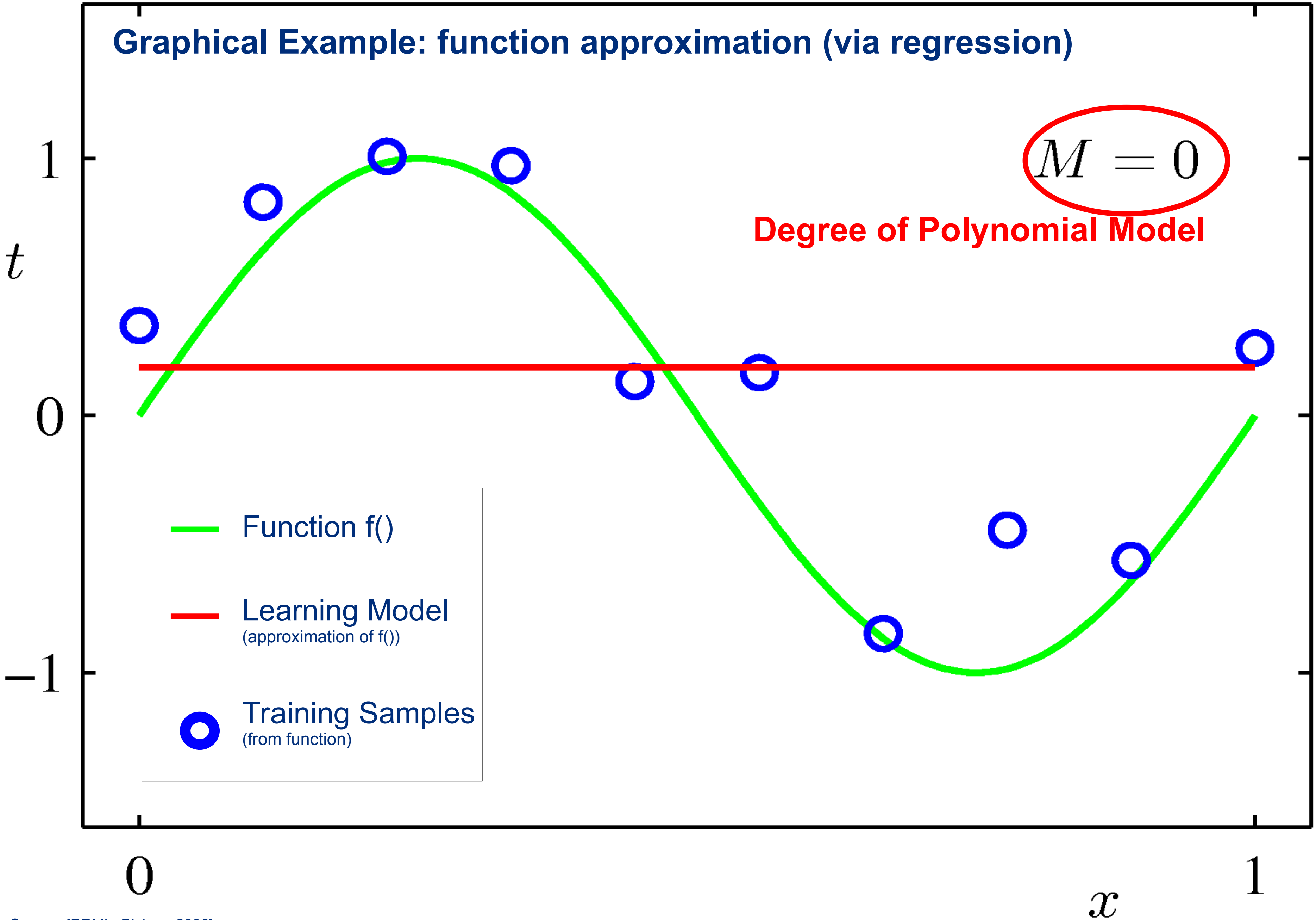
Overfitting in general

- **Hypothesis** is **too specific** towards training examples
- **Hypothesis not general enough** for test data

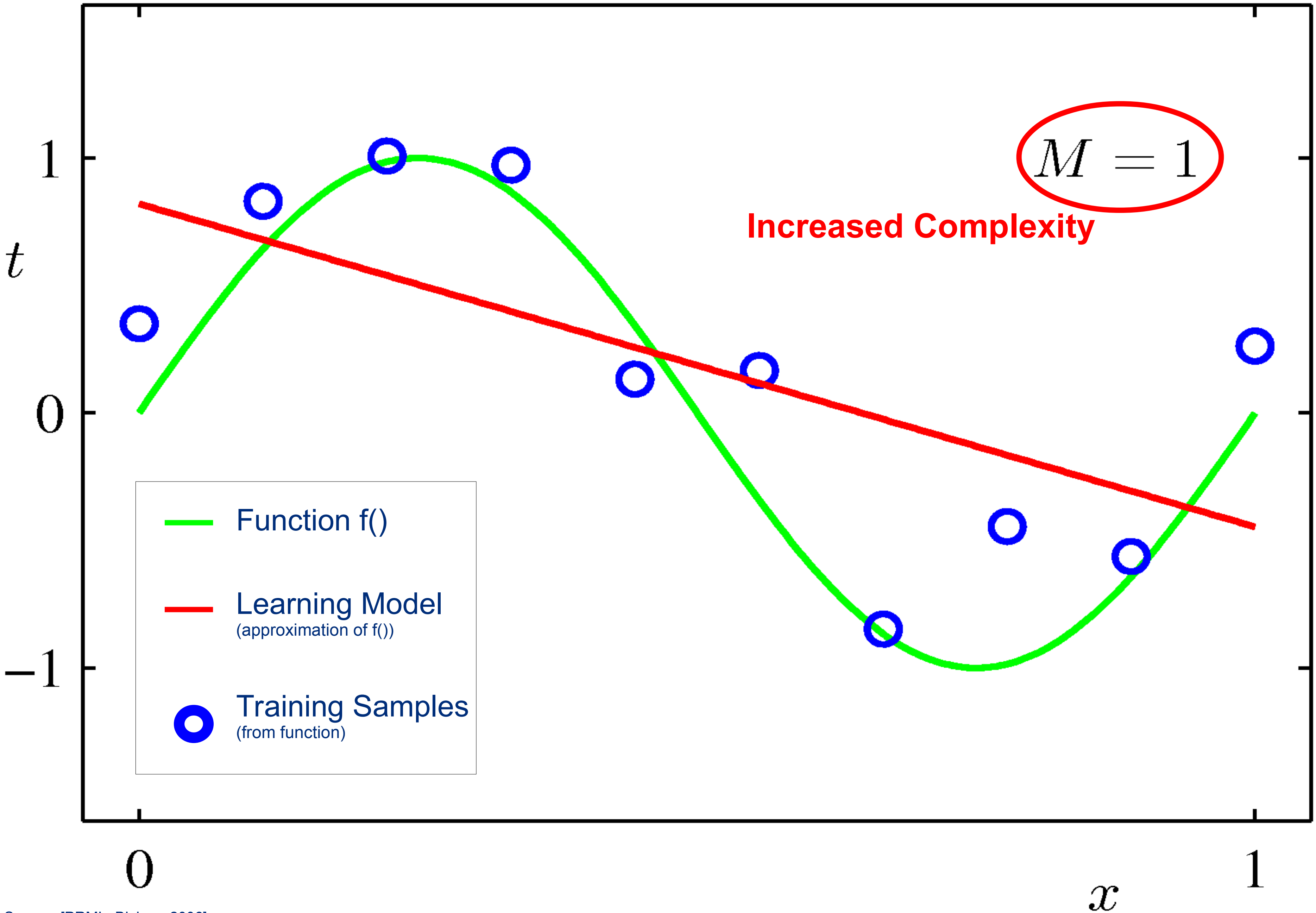


Increasing model complexity

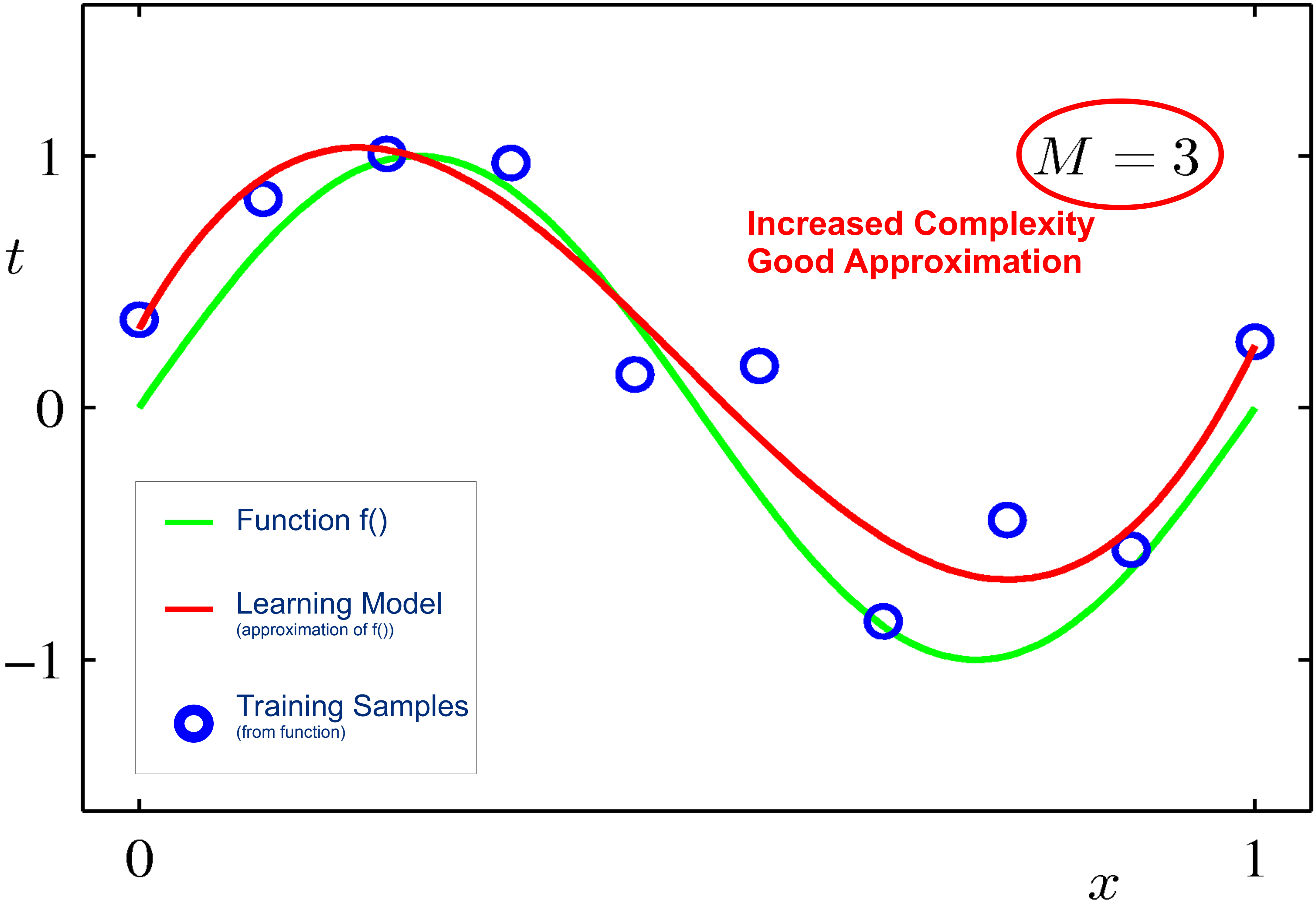
Graphical Example: function approximation (via regression)



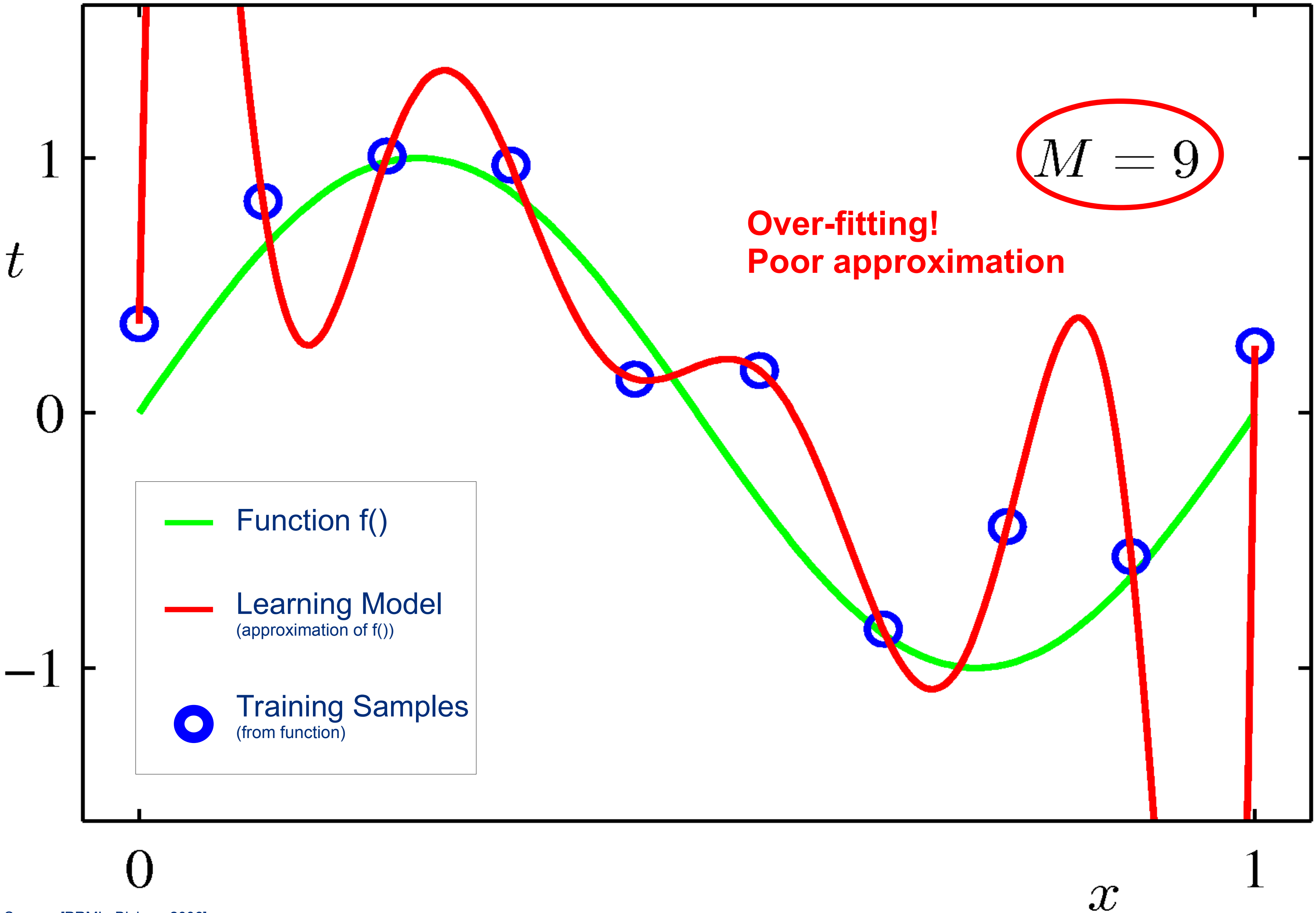
Source: [PRML, Bishop, 2006]



Source: [PRML, Bishop, 2006]



Source: [PRML, Bishop, 2006]

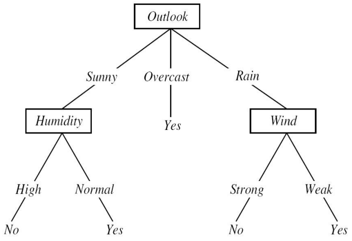


Source: [PRML, Bishop, 2006]

Avoiding Over-fitting

- **Robust Testing & Evaluation**
  - strictly separate training and test sets
    - train iteratively, test for over-fitting divergence
  - advanced training / testing strategies (K-fold cross validation)

- *For Decision Tree Case:*
  - control complexity of tree (e.g. depth)
    - stop growing when data split not statistically significant
    - grow full tree, then post-prune



  - minimize { *size(tree)* + *size(misclassifications(tree)* }
    - *i.e. simplest tree that does the job! (Occam)*

A *stitch* in time ...

Decision Tress

[Quinlan, '86]

and many others..

Ensemble Classifiers

Fact 1: Decision Trees are **Simple**

Fact 2: Performance on complex sensor interpretation problems is **Poor**

… unless we combine them in an **Ensemble Classifier**

Cranfield UNIVERSITY

Extending to **Multi-Tree Ensemble Classifiers**

- **Key Concept:** combining multiple WEAK classifiers
  - **strong classifier:** output strongly correlated to correct classification
  - **weak classifier:** output weakly correlated to correct classification
    - » *i.e. it makes a lot of miss-classifications (e.g. tree with limited depth)*

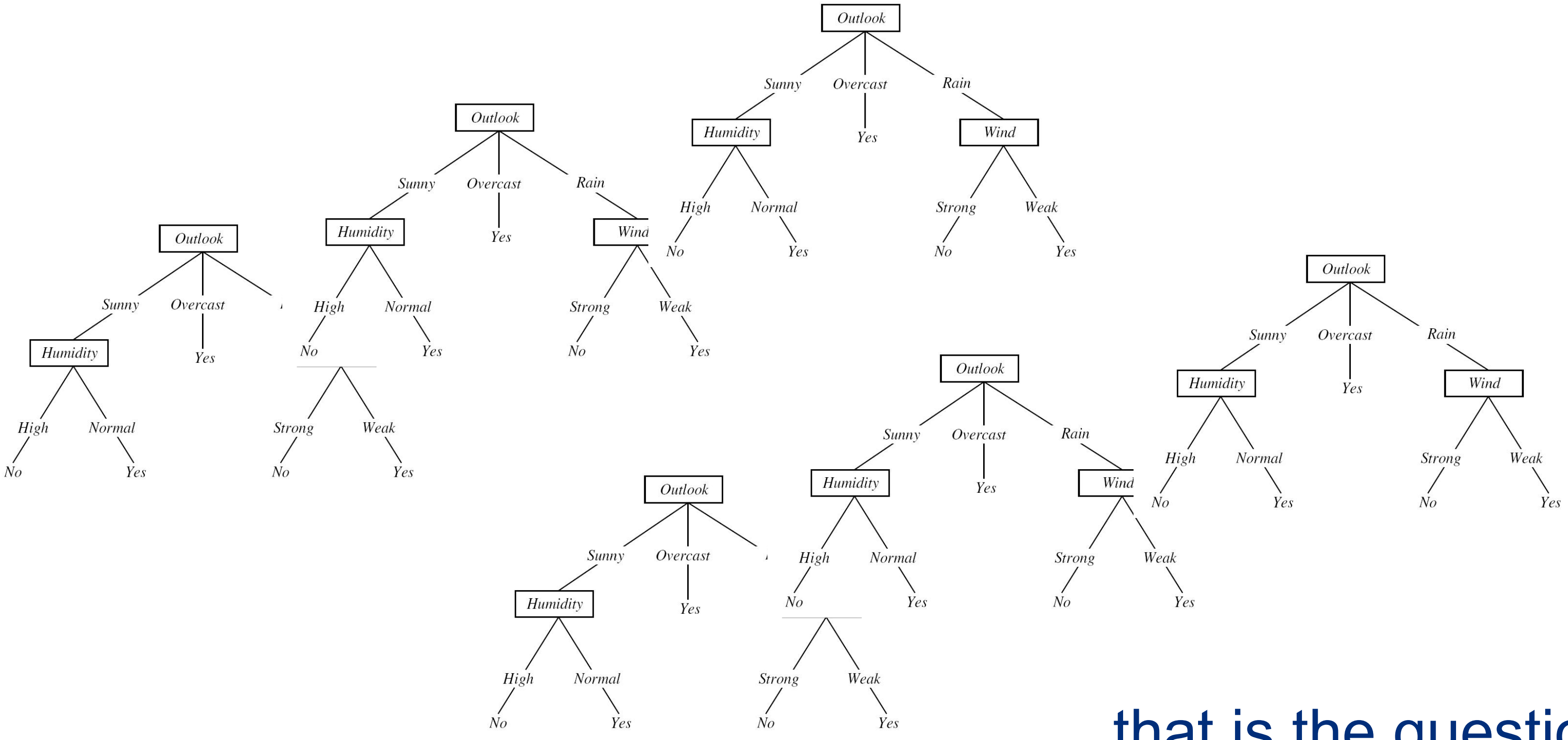
- *How to combine:*
  - **Bagging:**
    - train *N* classifiers on random sub-sets of training set; classify using majority vote of all *N* (and for regression use average of N predictions)
  - **Boosting:**
    - Use whole training set, but **introduce weights** for each classifier **based on performance** over the training set

- Two **examples**: *Boosted Trees + (Random) Decision Forests*
  - *N.B. Can be used with any classifiers (not just decision trees!)*

Cranfield UNIVERSITY

Extending to **Multi-Tree Classifiers**

- To bag or to boost .....



...... that is the question.

Learning using Boosting

Learning Boosted Classifier (Adaboost Algorithm)

```
Assign equal weight to each training instance
For t iterations:
  Apply learning algorithm to weighted training set,
    store resulting (weak) classifier
  Compute classifier's error e on weighted training set
  If e = 0 or e > 0.5:
    Terminate classifier generation
  For each instance in training set:
    If classified correctly by classifier:
       Multiply instance's weight by e/(1-e)
  Normalize weight of all instances
```

e = error of classifier on the training set

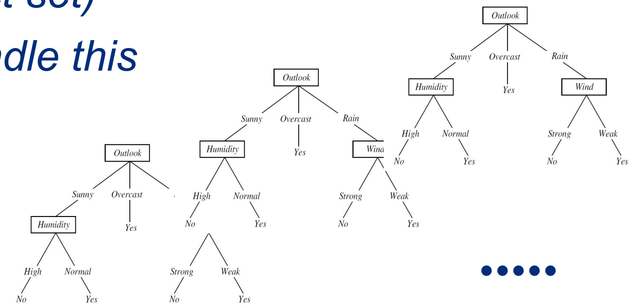
Classification using Boosted Classifier

```
Assign weight = 0 to all classes
For each of the t (or less) classifiers:
  For the class this classifier predicts
    add -log e/(1-e) to this class's weight
Return class with highest weight
```

Cranfield UNIVERSITY

Learning using **Boosting**

- **Some things to note:**
  - **Weight adjustment** means *t+1*th **classifier concentrates on the examples** tth classifier got **wrong**
  - Each classifier must be able to achieve greater than 50% success
    - (i.e. 0.5 in normalised error range {0..1})
  - Results in an **ensemble of *t* classifiers**
    - i.e. a boosted classifier made up of *t* weak classifiers
    - boosting/bagging classifiers often called **ensemble classifiers**
  - Training **error decreases exponentially** *(theoretically)*
    - **prone to over-fitting** *(need diversity in test set)*
      - *several additions/modifications to handle this*
  - ***Works best with weak classifiers***



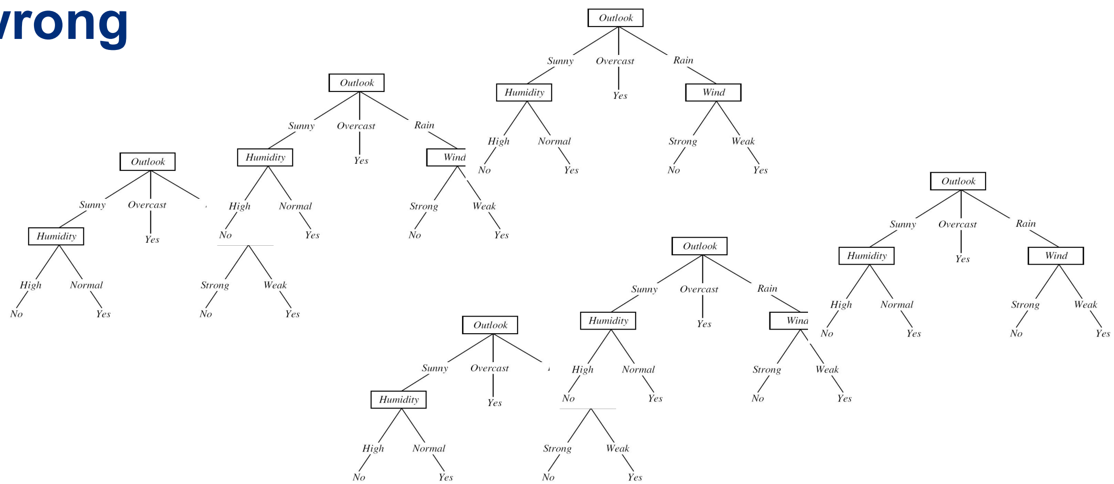
.....

- ***Boosted Trees***
  - *set of **t decision trees** of limited complexity (e.g. depth)*

Cranfield UNIVERSITY

Extending to **Multi-Tree Classifiers**

- Bagging = all equal *(simplest approach)*
- Boosting = classifiers weighted by performance
  - poor performers removed (zero or very low) weight
  - *t+1th* **classifier concentrates on the examples** tth classifier got **wrong**



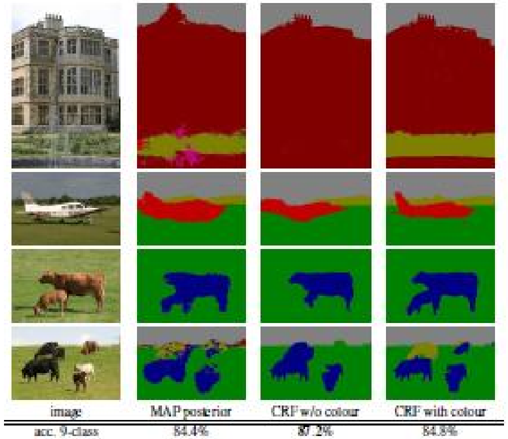
To bag or boost ? - *boosting generally works very well (but what about over-fitting ?)*

Cranfield UNIVERSITY

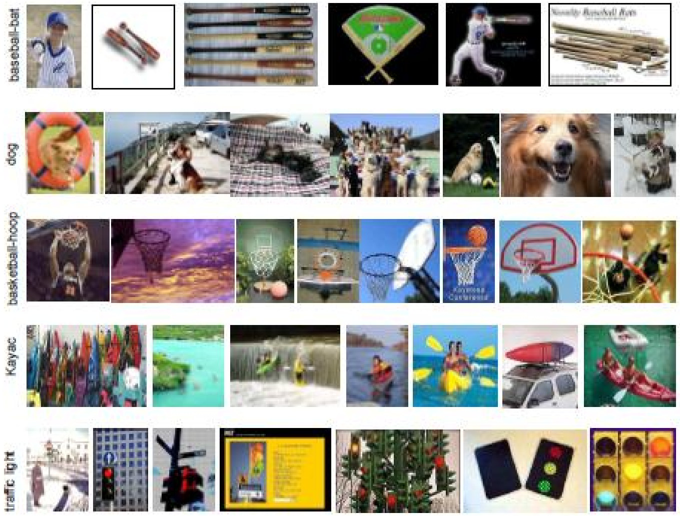
Decision Forests (a.k.a. Random Forests/Trees)

- **Bagging** using multiple decision trees where each tree in the **ensemble classifier ...**
  - is trained on a **random subsets of the training data**
  - computes a node split on a **random subset of the attributes**

[Breiman 2001]



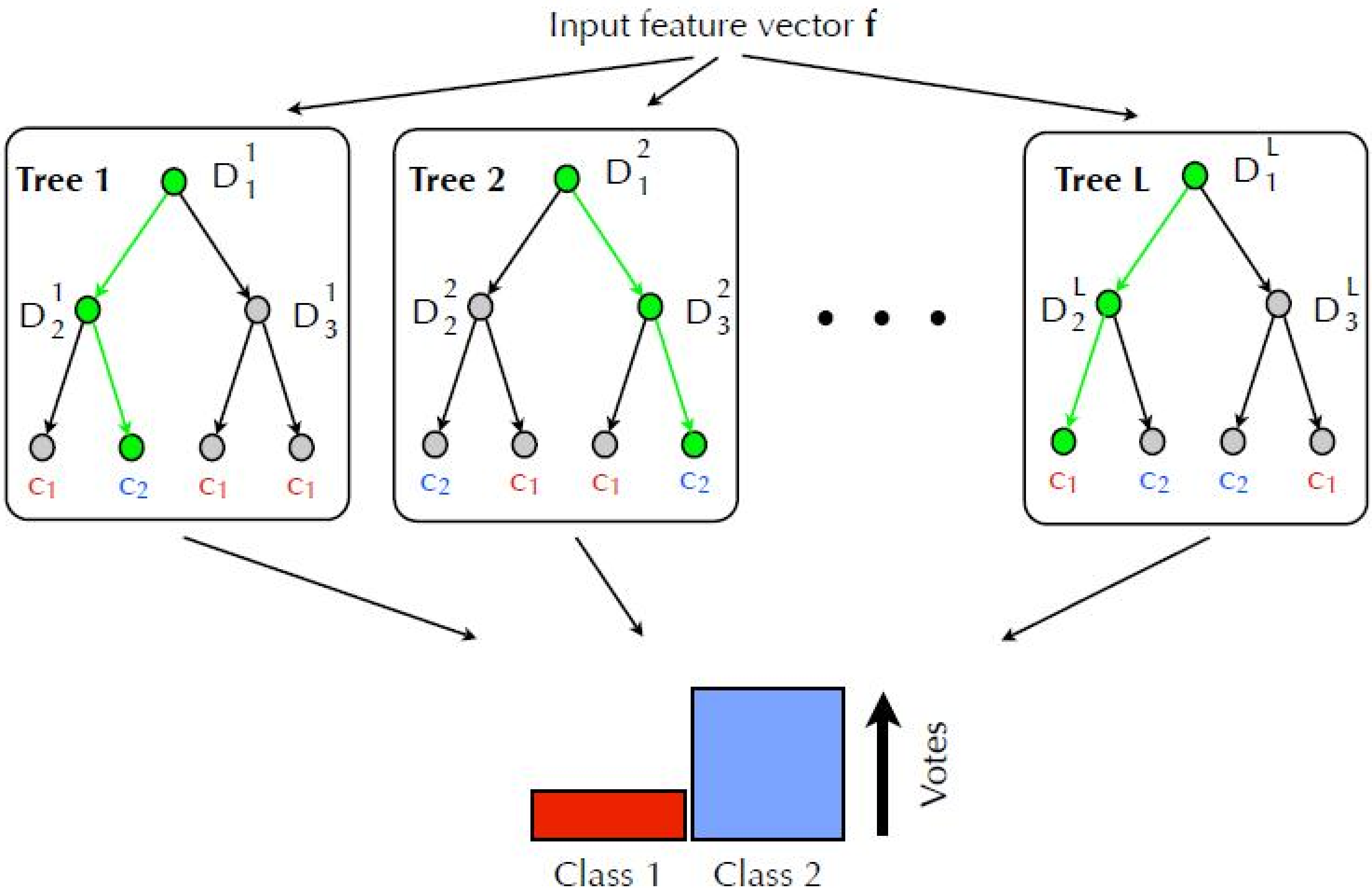
[schroff 2008]



[Bosch 2007]

  - close to “state of the art” for object segmentation / classification (inputs : feature vector descriptors)

Decision Forests (a.k.a. Random Forests/Trees)

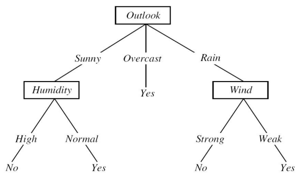


Images: David Capel, Penn. State.

Cranfield UNIVERSITY

Decision Forests (a.k.a. Random Forests/Trees)

- **Decision Forest = Multi Decision Tree Ensemble Classifier**
  – bagging approach used to return classification



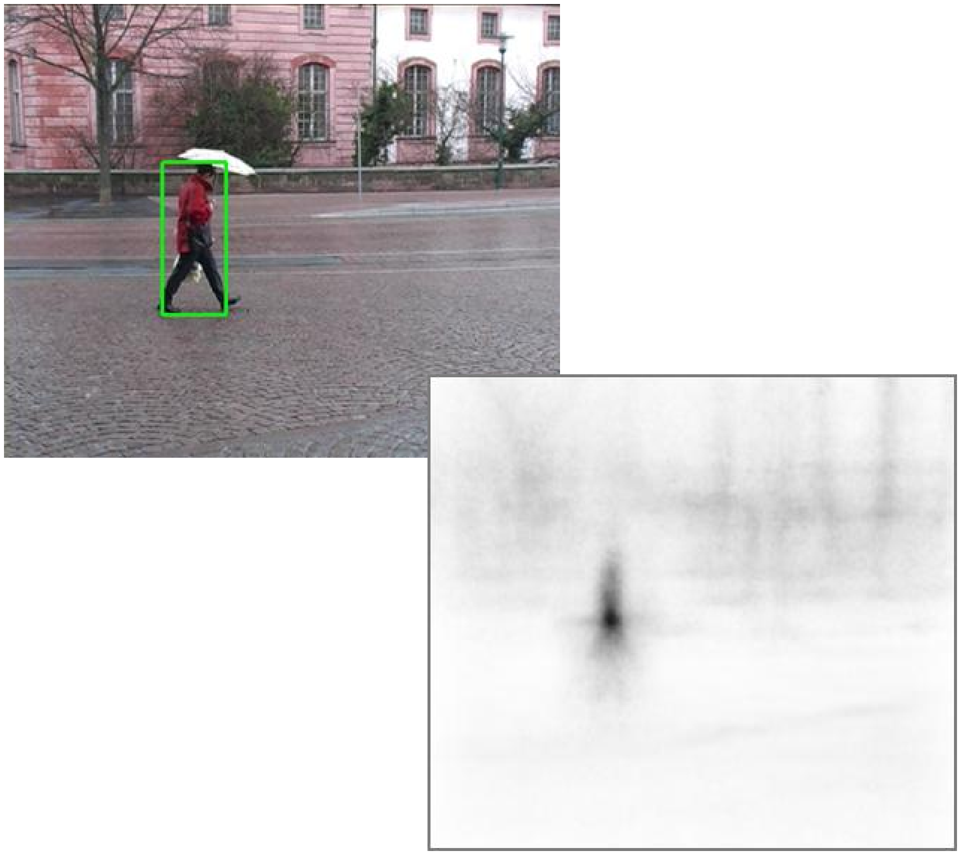
  – [alternatively weighted by number of training items assigned to the final leaf node reached in tree that have the same class as the sample (classification) or statistical value (regression)]

- **Benefits:** efficient on large data sets with multi attributes and/or missing data, inherent variable importance calc., unbiased test error ("out of bag"), *"does not overfit"*

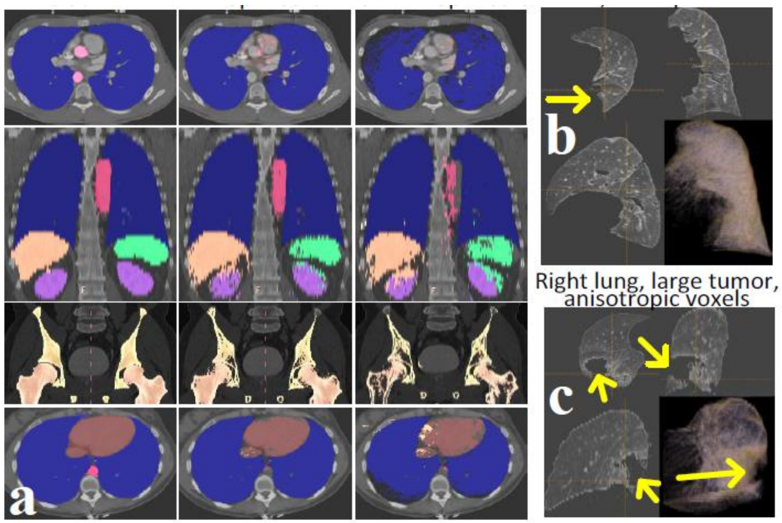
- **Drawbacks:** evaluation can be slow, lots of data for good performance, complexity of storage ...

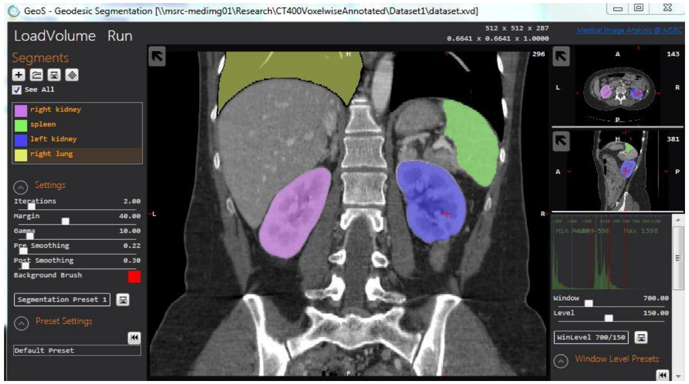
["Random Forests", Breiman 2001]

Decision Forests (a.k.a. Random Forests/Trees)



Gall J. and Lempitsky V., Class-Specific Hough Forests for Object **Detection**, IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), 2009.

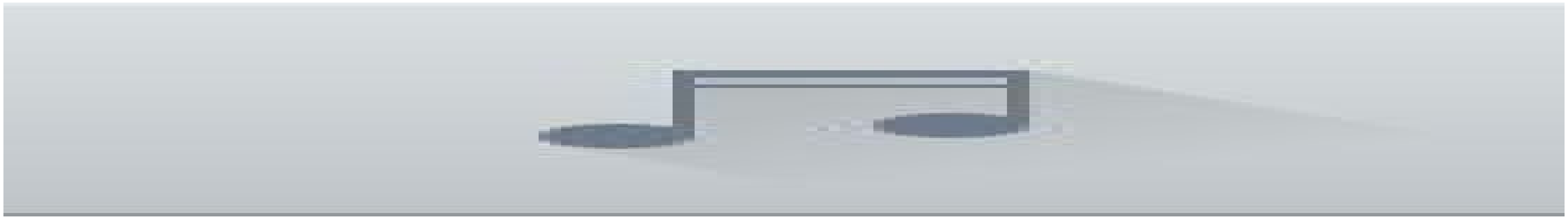




Montillo et al.. "Entangled decision forests and their application for semantic **segmentation** of CT images." In Information Processing in Medical Imaging, pp. 184-196. 2011.
http://research.microsoft.com/en-us/projects/decisionforests/

Microsoft Kinect ….

- **Body Pose Estimation** in Real-time From Depth Images
  - uses Decision Forest Approach



Shotton et al., Real-Time Human Pose Recognition in Parts from a Single Depth Image, CVPR, 2011 - http://research.microsoft.com/apps/pubs/default.aspx?id=145347

Why do they work so well ?

- **Optimal cut points depend** strongly on the **training set used** (high variance)
  - hence idea of using multiple trees voting for result

- **For multiple trees** to be most effective **the trees should be independent**
  - **splitting** using a random feature subset supports this

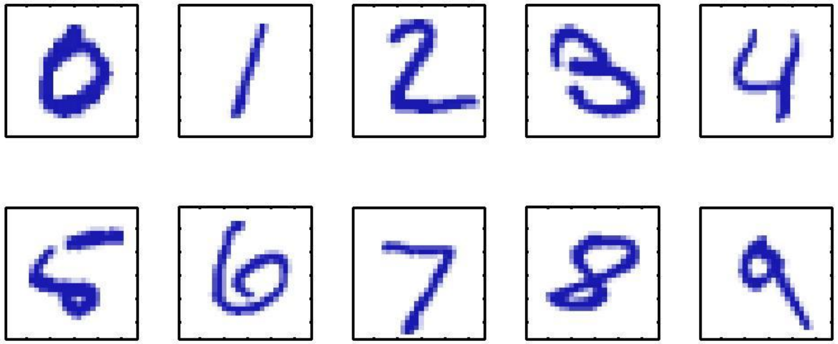
- **Averaging** the outputs of trees **reduces overfitting** to noise.
  - thus pruning (complexity reduction) is not needed

Cranfield UNIVERSITY

Comparison - Classical Problem

- Handwritten Digit Recognition
  - 10 class problem
  - 64 features / attributes

  Dataset: [Alpaydin / Kaynak, 98]



[Bishop 2006]

Technique	True Class.	False Class.	
Decision Tree	84.69%	15.3%	(depth <=25)
Boosted Trees	82.03%	17.97%	(100 trees)
Decision (Random) Forest	96.49%	3.5%	(100 trees)
Extreme Random Forest*	96.71%	3.28%	(100 trees)
Support Vector Machine (SVM)	96.10%	3.89%	(linear kernel)
Neural Network	71.56%	28.43%	(3-layer, 10 hidden nodes)
Naive Bayes	84.81%	15.19%	

* + random attribute split threshold

Comparison: clutter noise ….

Feature Vector	Accuracy	Precision	TNR	Recall
Isolated Zernike	93.65	88.52	90.14	98.18
Isolated HSI	98.41	96.72	97.01	100
Combined	98.41	96.72	97.01	100

Table I. Performance of Support Vector Machine classifier (%)

Feature Vector	Accuracy	Precision	TNR	Recall
Isolated Zernike	80.95	91.11	93.85	67.21
Isolated HSI	89.68	96.15	96.9	81.97
Combined	97.61	100	100	95.08

Table II. Performance of Neural Network classifier (%)

Feature Vector	Accuracy	Precision	TNR	Recall
Isolated Zernike	70.63	81.58	89.23	50.82
Isolated HSI	98.41	100	100	96.72
Combined	98.41	100	100	96.72

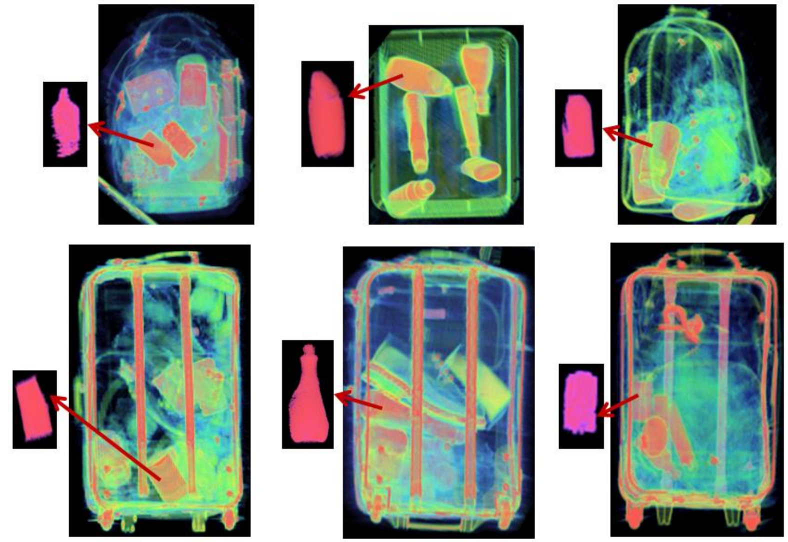
Table III. Performance of Decision Tree classifier (%)

Feature Vector	Accuracy	Precision	TNR	Recall
Isolated Zernike	89.68	92.86	93.85	85.25
Isolated HSI	98.41	100	100	96.72
Combined	98.41	100	100	96.72

Table IV. Performance of Boosted Decision Tree classifier (%)

Feature Vector	Accuracy	Precision	TNR	Recall
Isolated Zernike	89.95	93.02	95.38	65.57
Isolated HSI	100	100	100	100
Combined	98.41	100	100	96.72

Table V. Performance of Random Forest classifier (%)



A Comparison of Classification Approaches for Threat Detection in CT based Baggage Screening (N. Megherbi, J. Han, G.T. Flitton, T.P. Breckon), In Proc. Int. Conf. on Image Processing, pp. 3109-3112, 2012.

What if **every weak classifier was just** the presence/absence **of an image feature** ?

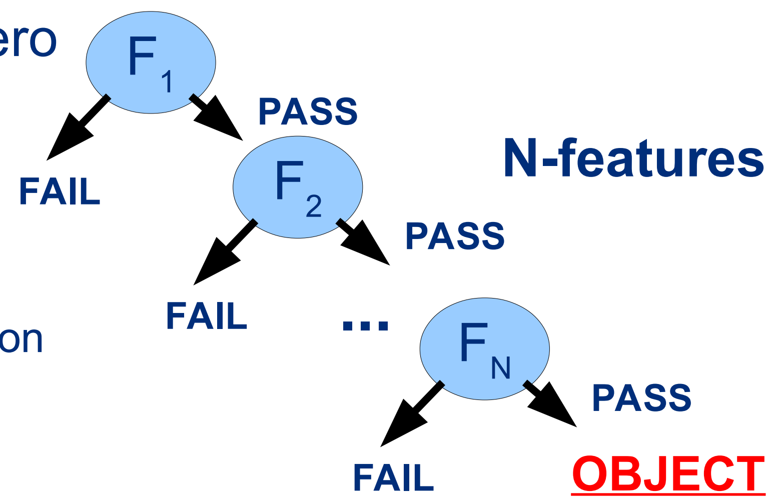
(i.e. feature present = {yes, no})

As the **number of features present from a given object**, in a given scene location, **goes up** the **probability of the object not being present goes down!**

This is the concept of feature cascades.

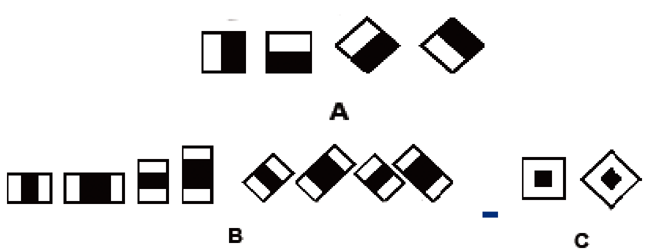
Feature Cascading .....

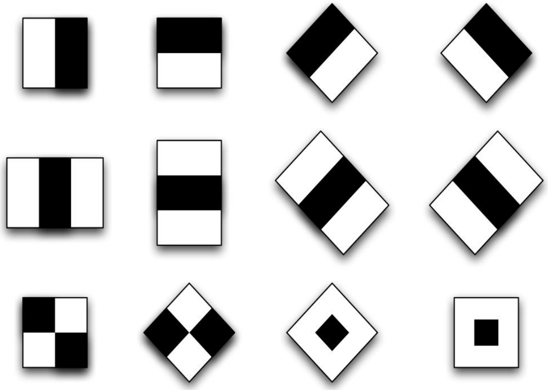
- Use boosting to **order image features from most to least discriminative for a given object** ....
  - allow high false positive per feature *(i.e. it's a weak classifier!)*
  - select features via boosting
- As feature F_1 to F_N of an object is present → probability of non-occurrence within the image tends to zero



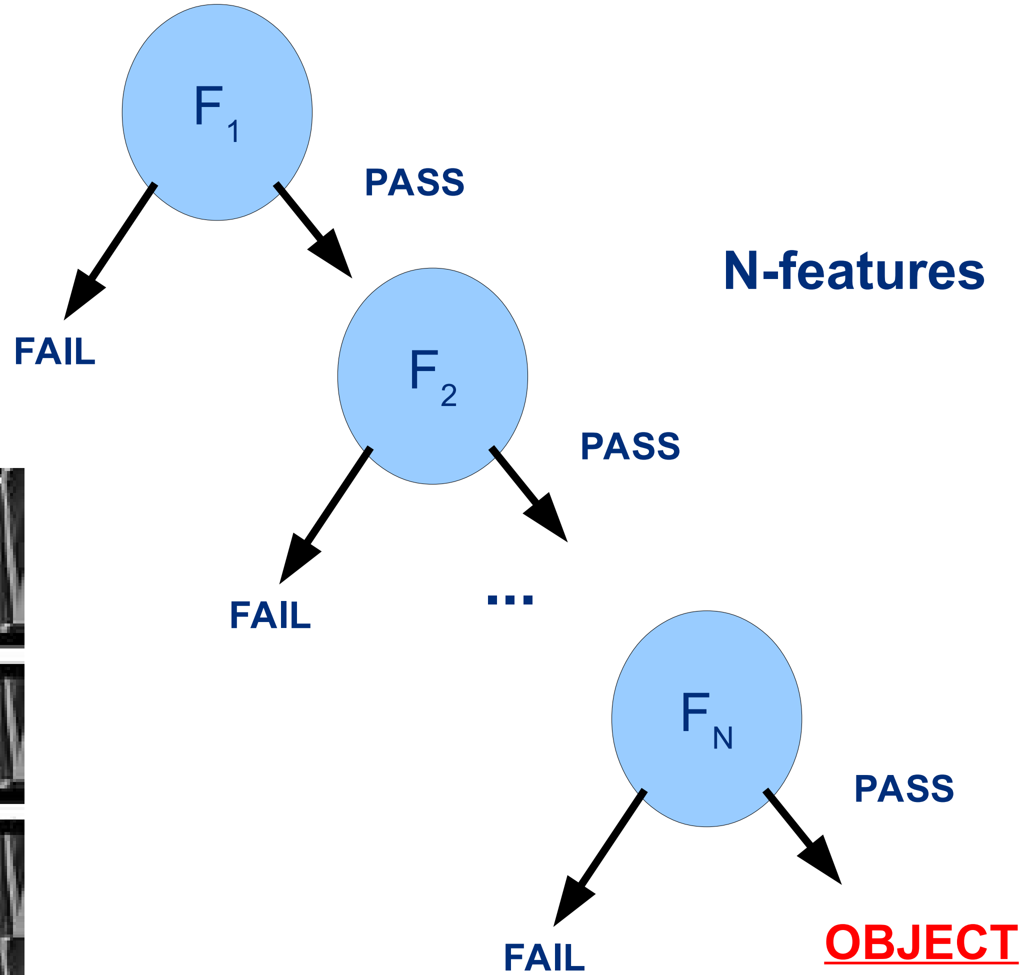
- e.g. Extended **Haar features**
  - set of differences between image regions
  - rapid evaluation (and non-occurrence) rejection

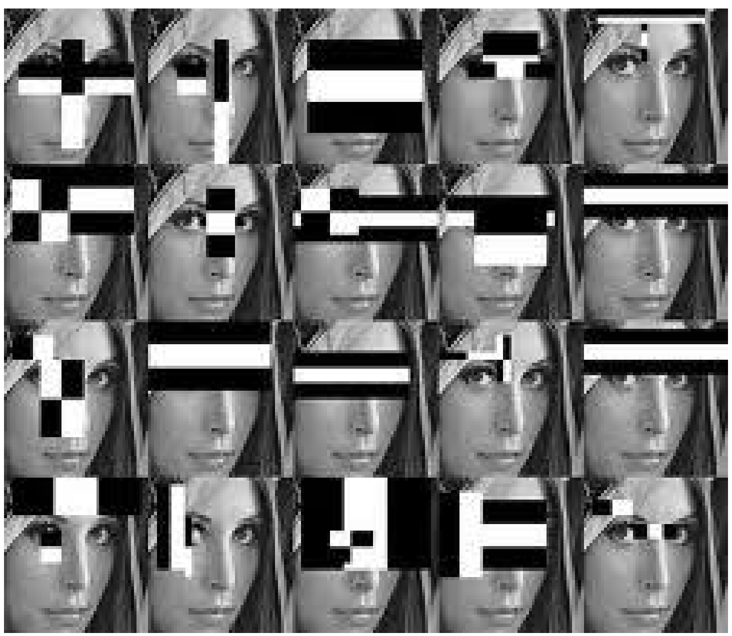
[Volia / Jones 2004]



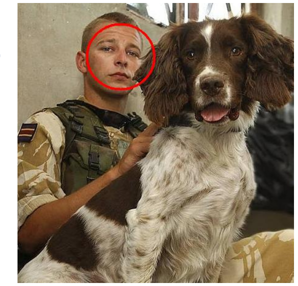


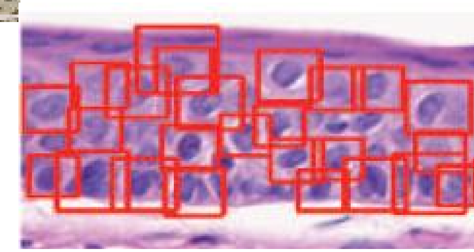
. . . . . .


F1
PASS
FAIL
F2
PASS
FAIL
...
FN
PASS
FAIL
OBJECT
N-features



Haar Feature Cascades

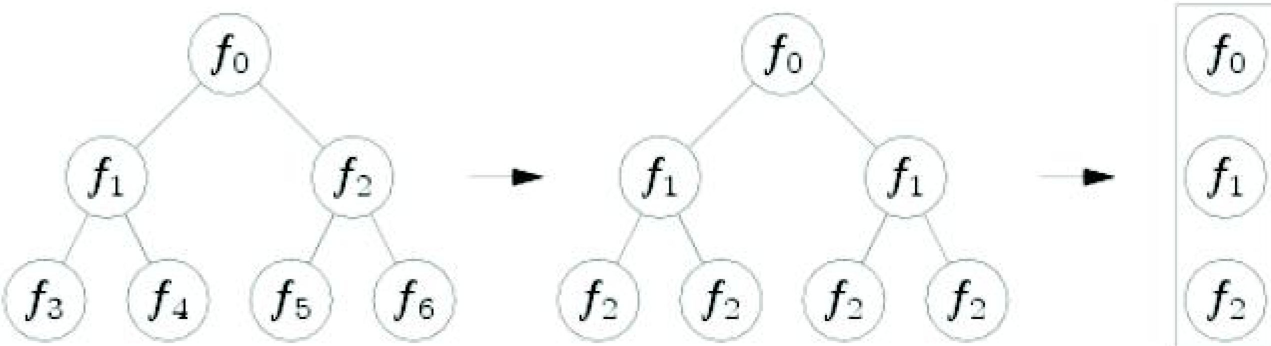




- **Real-time Generalised Object Recognition**
- **Benefits**
  - Multi-scale evaluation
    - scale invariant
  - Fast, real-time detection
  - “Direct” on image
    - no feature extraction
  - Haar features
    - contrast/ colour invariant
- **Limitations**
  - poor **performance on non-rigid objects**
  - object **rotation**

[Breckon / Eichner / Barnes / Han / Gaszczak 08-09]

Ferns ...

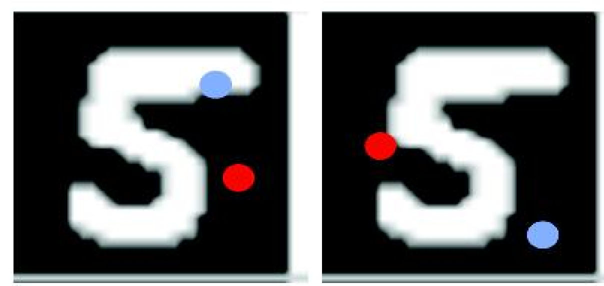


- Concept: “a constrained tree where a simple binary test is performed at each level”

e.g relative intensities of a pair of pixels:

$f_1(\mathbf{I}) = I(x_a, y_a) > I(x_b, y_b)$ -> true

$f_2(\mathbf{I}) = I(x_c, y_c) > I(x_d, y_d)$ -> false



Images: David Capel, Penn. State.

Ferns = "Semi-Naive" Bayes

- Class C_k & feature set $\{ f_i \}$
- Posterior probability : $$\underset{k}{\operatorname{argmax}}\ P(C_k | f_1, f_2, \ldots, f_N)$$
- Via Bayes rule : $$\underset{k}{\operatorname{argmax}}\ P(f_1, f_2, \ldots, f_N | C_k) P(C_k)$$ (likelihood x prior)
- Naive Bayes $$P(f_1, f_2, \ldots, f_N | C_k) = \prod_{i=1}^{N} P(f_i | C_k)$$
  - assume features are independent
  - often invalid assumption

Cranfield UNIVERSITY

Ferns = “Semi-Naive” Bayes

- Group features into sets, F_l, *of size S*

$$F_l = \{f_{l,1}, f_{l,2}, \ldots, f_{l,S}\}$$

- Assume groups are conditional independent

$$P(f_1, f_2, \ldots, f_N | C_k) = \prod_{l=1}^{L} P(F_l | C_k)$$

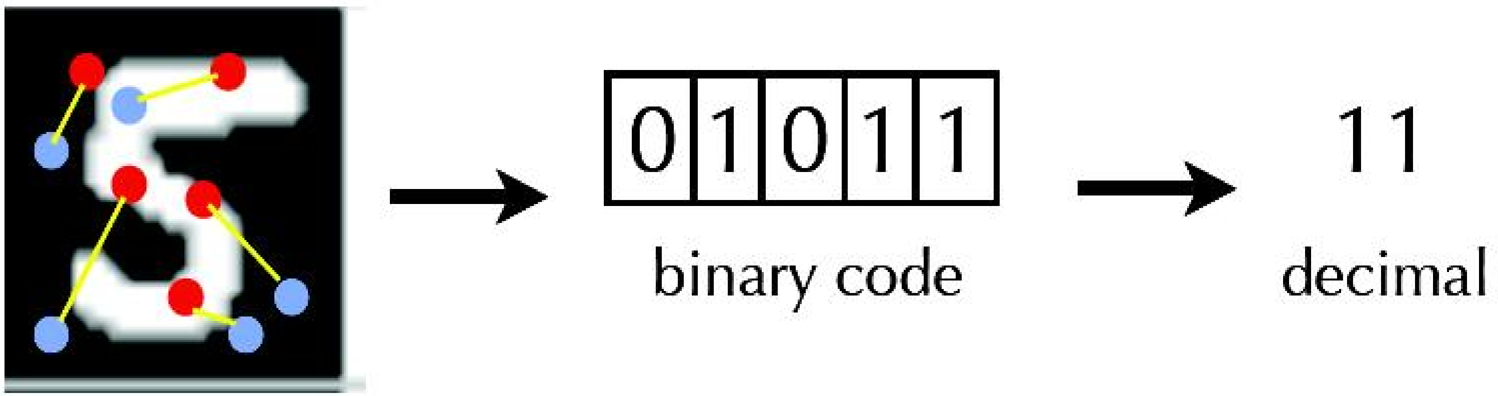
- Perform classification via “Semi-Naive” Bayes approach

$$\text{Class}(\mathbf{f}) \equiv \underset{k}{\text{argmax}}\ P(C_k) \prod_{l=1}^{L} P(F_l | C_k)$$

Cranfield UNIVERSITY

Ferns ...

- Result = S-digit binary code for a given set of S tests



… to be interpreted as an decimal value $0 \rightarrow 2^S$

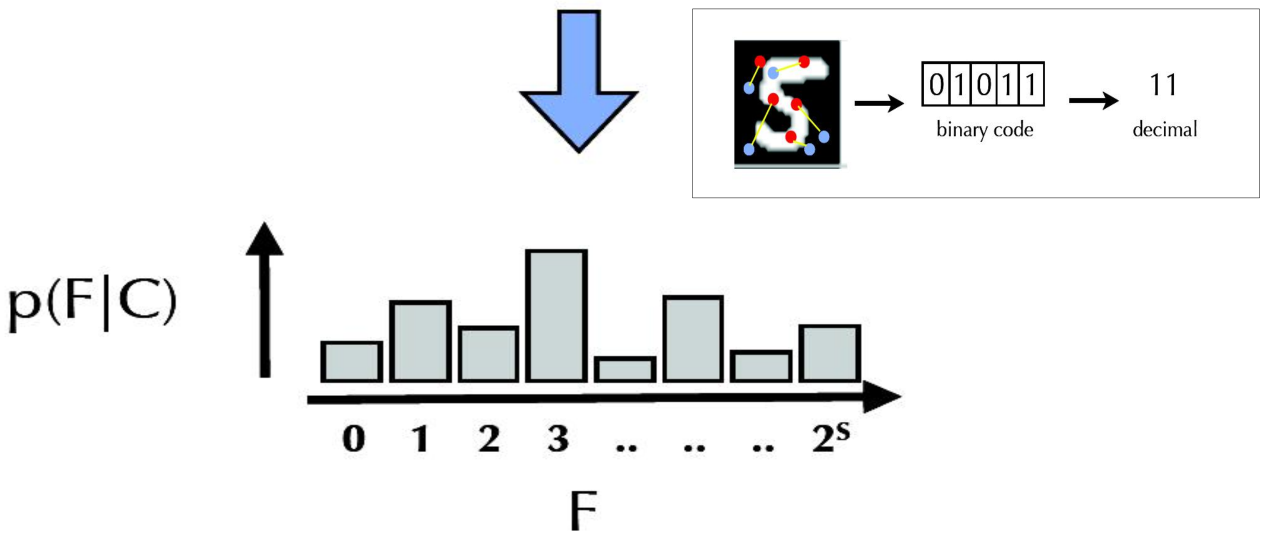
- *Essentially* a "hash" (lookup) of S-digit binary value to $0 \rightarrow 2^S$

Images: David Capel, Penn. State.

Cranfield UNIVERSITY

Ferns ...

- Apply to a large number of (training) examples to learn a multinomial distribution of this “hash” value $0 \rightarrow 2^S$

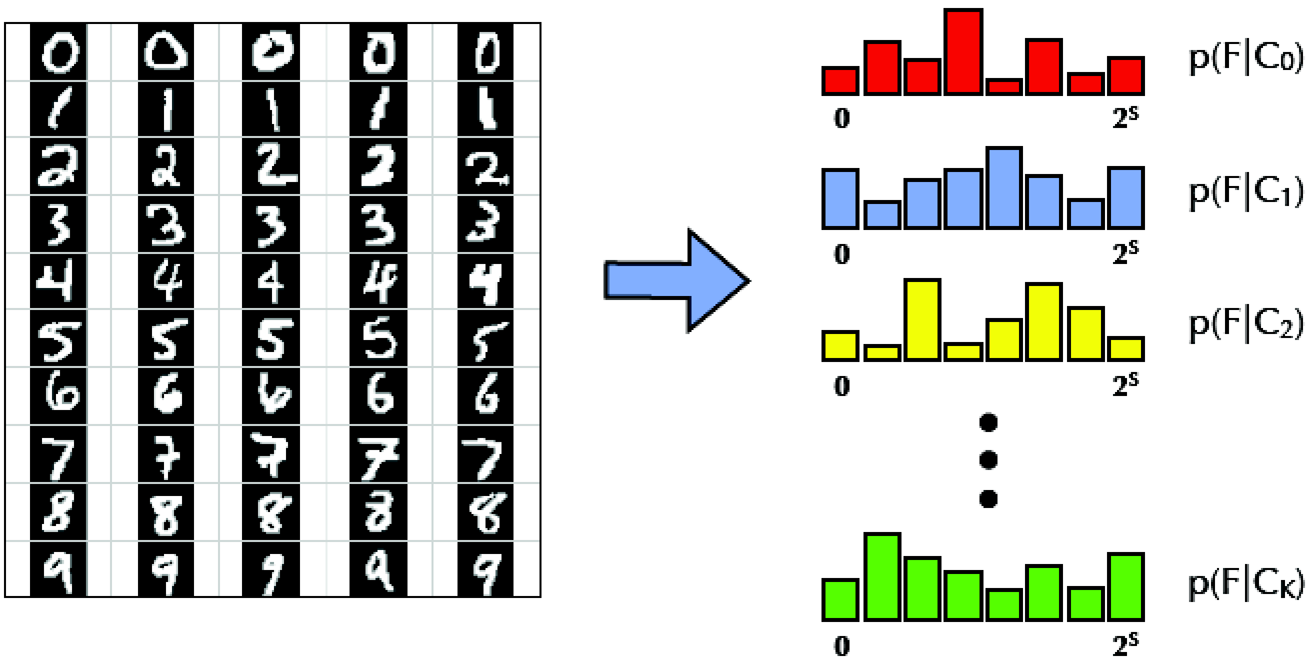


Images: David Capel, Penn. State.

Cranfield UNIVERSITY

Ferns ….

- Repeat for all classes ….

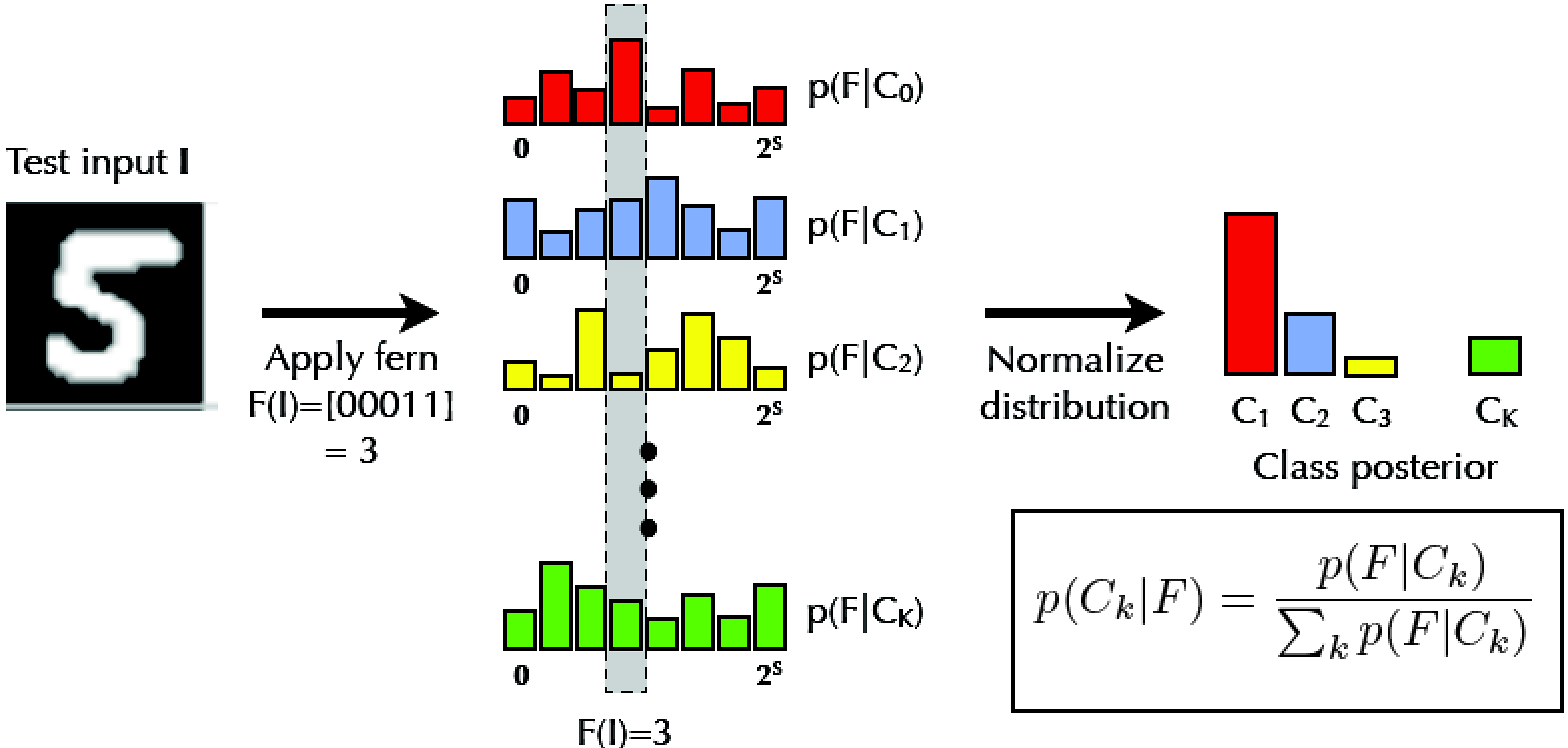


… obtain one distribution per class

Images: David Capel, Penn. State.

Fern Based Classification

- For an **unseen example, I:**
  - construct **fern**
  - **perform lookup** via decimal “hash”
  - **compute posterior probability** for class



$$p(C_k|F) = \frac{p(F|C_k)}{\sum_k p(F|C_k)}$$

Images: David Capel, Penn. State.

Cranfield UNIVERSITY

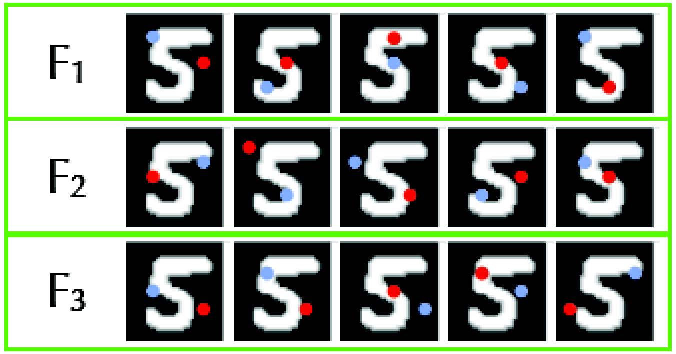
Random Ferns

- **Construct** *L* ferns from **random feature subsets**

e.g. $F_1 = \{f_2, f_7, f_{22}, f_5, f_9\}$

$F_2 = \{f_4, f_1, f_{11}, f_8, f_3\}$

$F_3 = \{f_6, f_{31}, f_{28}, f_{11}, f_2\}$



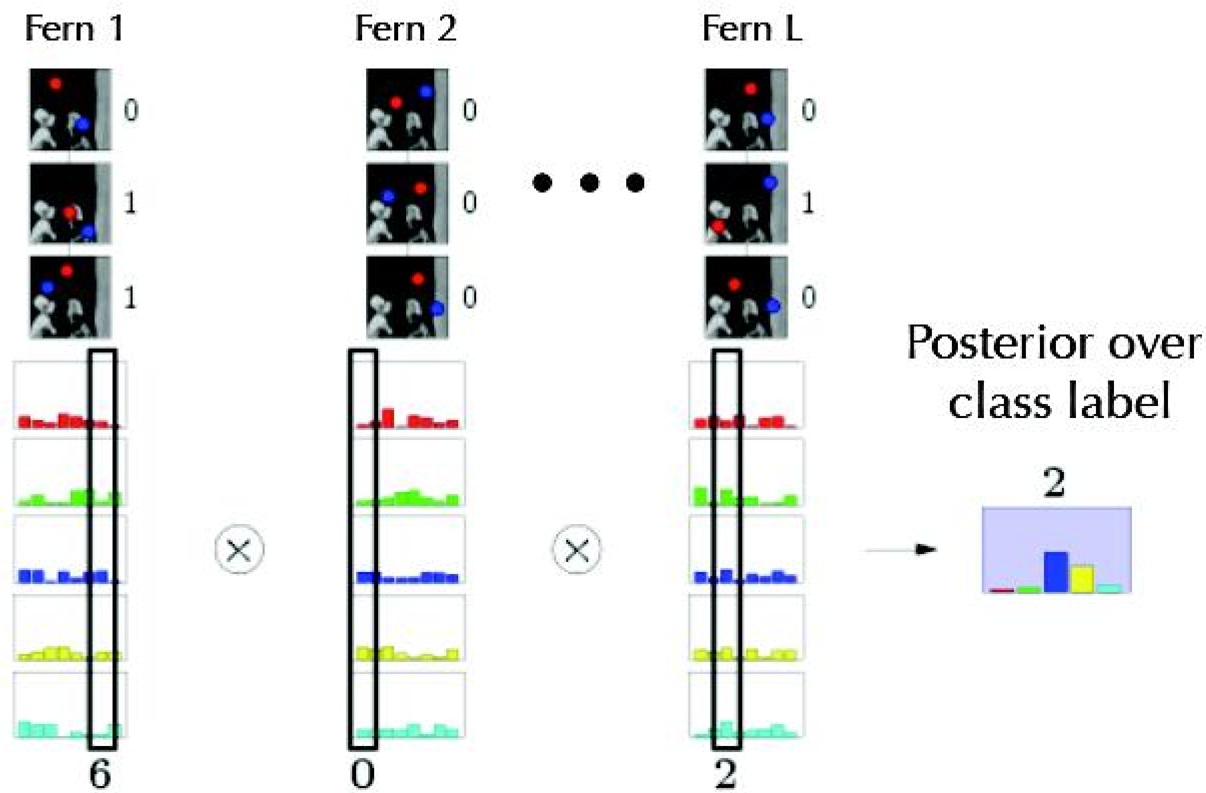
- **Classify** using whole set
- **Compute** most **probable class**, C_k, as:

$$\mathrm{Class}(\mathbf{f}) \equiv \underset{k}{\mathrm{argmax}}\, P(C_k) \prod_{l=1}^{L} P(F_l|C_k)$$

Images: David Capel, Penn. State.

Random Ferns

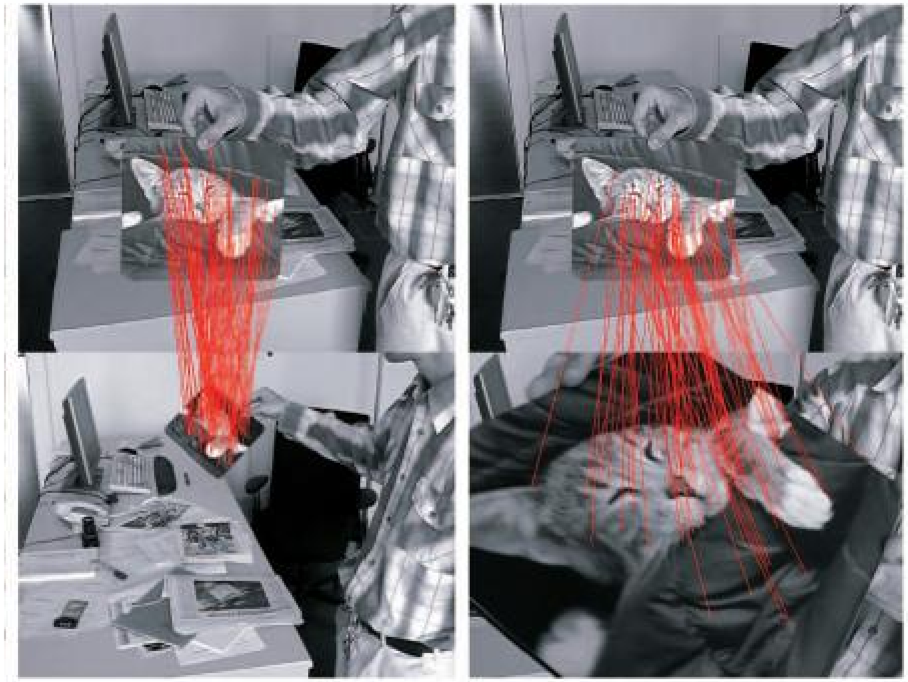
- Classification now only involves "fast lookup":



Images: David Capel, Penn. State.

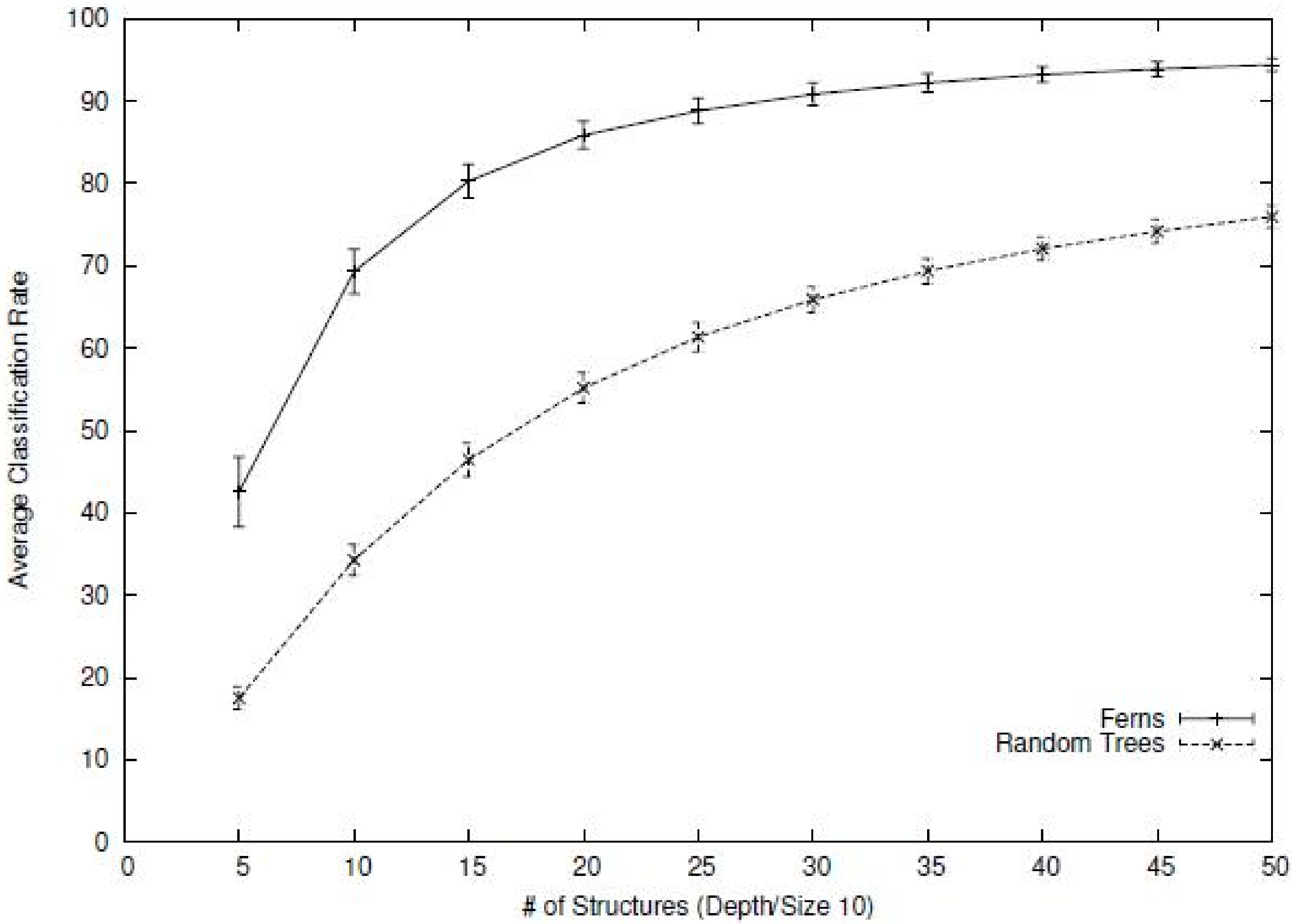
Comparison ...

- fast key-point matching
  - each point is a class
  - trained on 1000s affine transforms of same patch
  - fast, robust
  - S = 10
  - ensembles of 5-50 ferns



Ozuysal, Mustafa, et al. "Fast keypoint recognition using random ferns." Pattern Analysis and Machine Intelligence, IEEE Transactions on 32.3 (2010): 448-461.

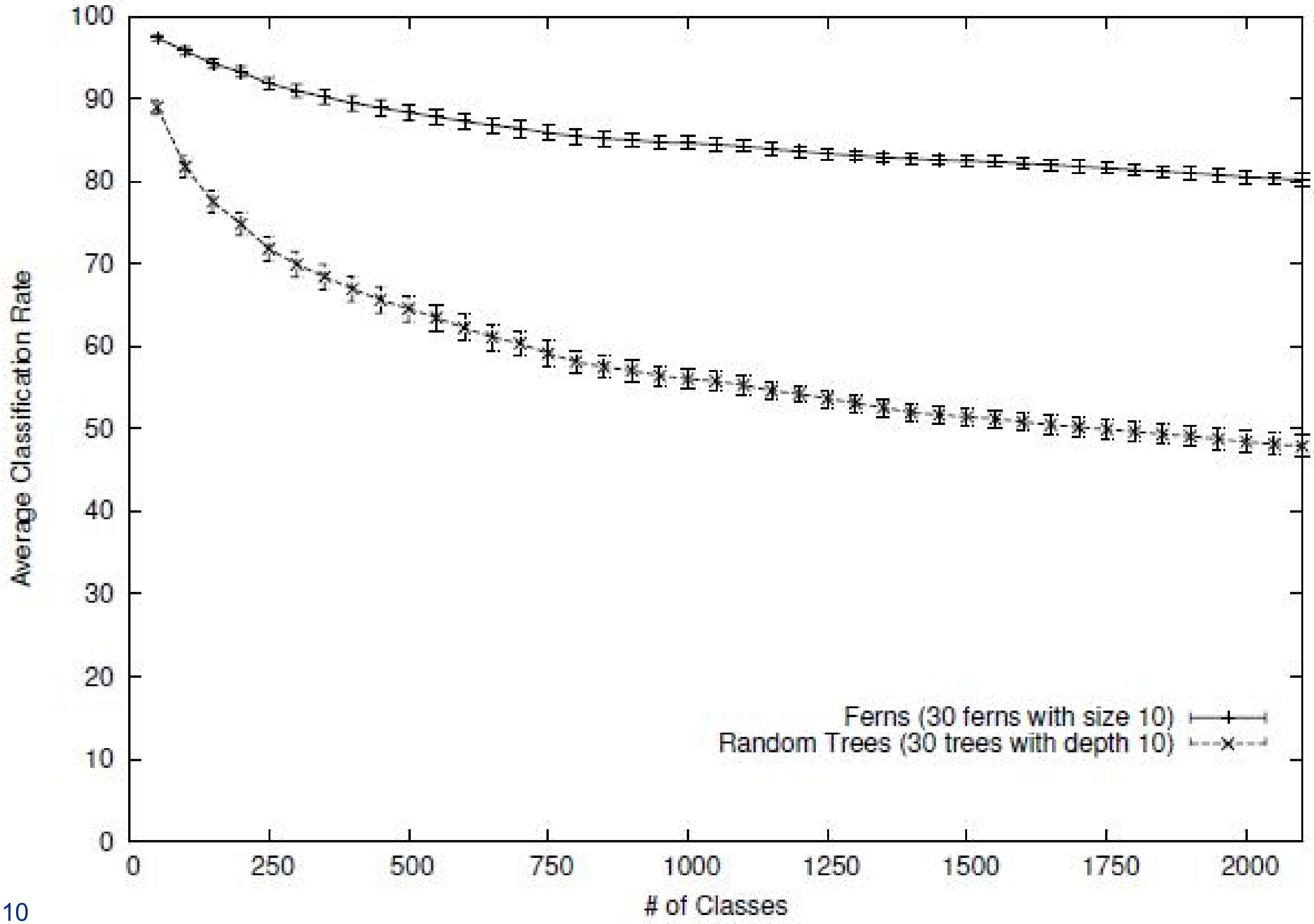
Comparison ...



Images: David Capel, Penn. State.

Cranfield UNIVERSITY

Comparison ...



30 ferns with S = 10

Images: David Capel, Penn. State.

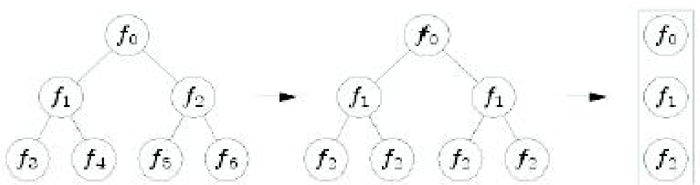
Comparison ...

Random Forests

- decision trees directly **learn the posterior *$P(C_k|F)$***
- **different sequence of tests** in each child node
- **training** time grows **exponentially with tree depth**
- **combine** tree hypotheses **by averaging**

Ferns

- **learn class-conditional distributions *$P(F|C_k)$***
- **same sequence of tests** to every input vector
- **training** time grows **linearly with fern size S**
- **combine** hypothesis using **Bayes rule (multiplication)**



Tree classifier Fern classifier

Images: David Capel, Penn. State.

Cranfield UNIVERSITY

Comparison ...

- Fern classifiers can be very memory hungry, e.g.
  - Fern size = 11
  - Number of ferns = 50
  - Number of classes = 1000

RAM = 2^S * sizeof(float) * NumFerns * NumClasses

= 2048 * 4 * 50 * 1000

= 400 Mbytes!

Example: David Capel, Penn. State.

- …… BUT so can Random Forests

  BUT both easy to parallelize

No Free Lunch! *(Theorem)*

- .... the idea that it is **impossible to get something for nothing**
- This is very **true in Machine Learning**
  - **approaches** that **train quickly** or **require little memory** or **few training examples** produce **poor results**
    - and vice versa .... !!!!!
  - **poor data = poor learning**
    - **problems with data = problems with learning**
    - **problems = {not enough data, poorly labelled, biased, unrepresentative … }**

Cranfield UNIVERSITY

What we have seen ...

- The power of combining *simple* things ….
  - **Ensemble Classifiers**

– *concept extends to all ML approaches*

- **Decision Forests**
  - Decision Trees back from the grave (or the '80s)

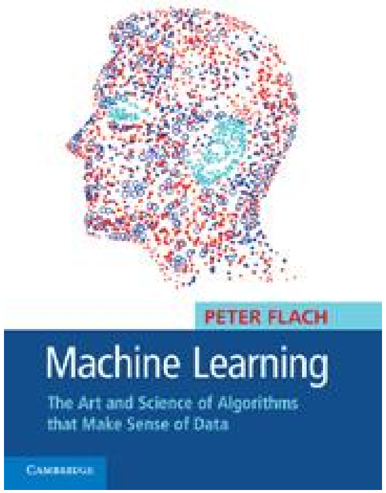
– *many, many variants*

- **Ferns**
  - simplified trees, fast, powerful

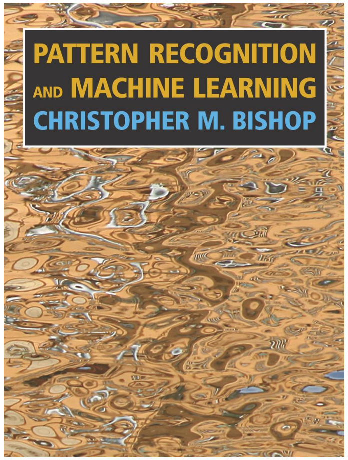
– *beginning of the story*

Further Reading - *textbooks*

- **Machine Learning (P. Flach), Cambridge University Press, 2012.**

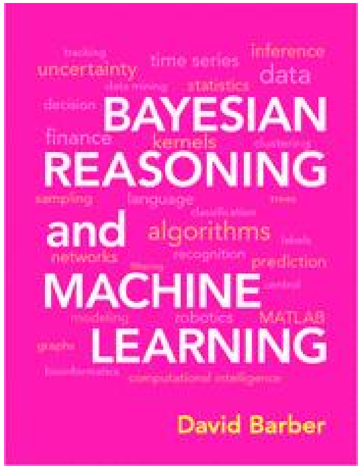


- **Pattern Recognition & Machine Learning - Christopher Bishop (Springer, 2006)**

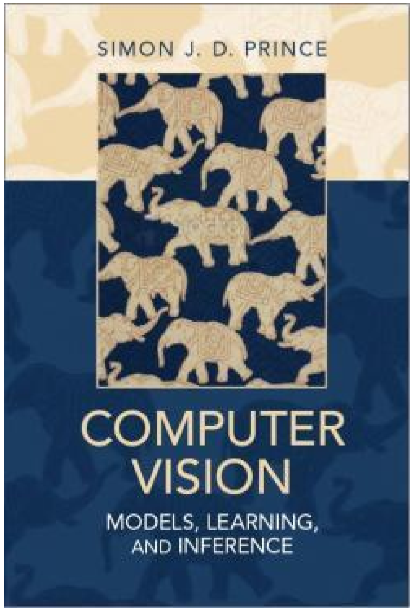


Further Reading - *textbooks*

- **Bayesian Reasoning and Machine Learning**
  – David Barber
  http://www.cs.ucl.ac.uk/staff/d.barber/brml/
  (Cambs. Univ. Press, 2012)



- ***Computer Vision: Models, Learning, and Inference***
  – Simon Prince
  (Springer, 2012)
  http://www.computervisionmodels.com/



*… both very **probability driven**, both available as **free PDF online***

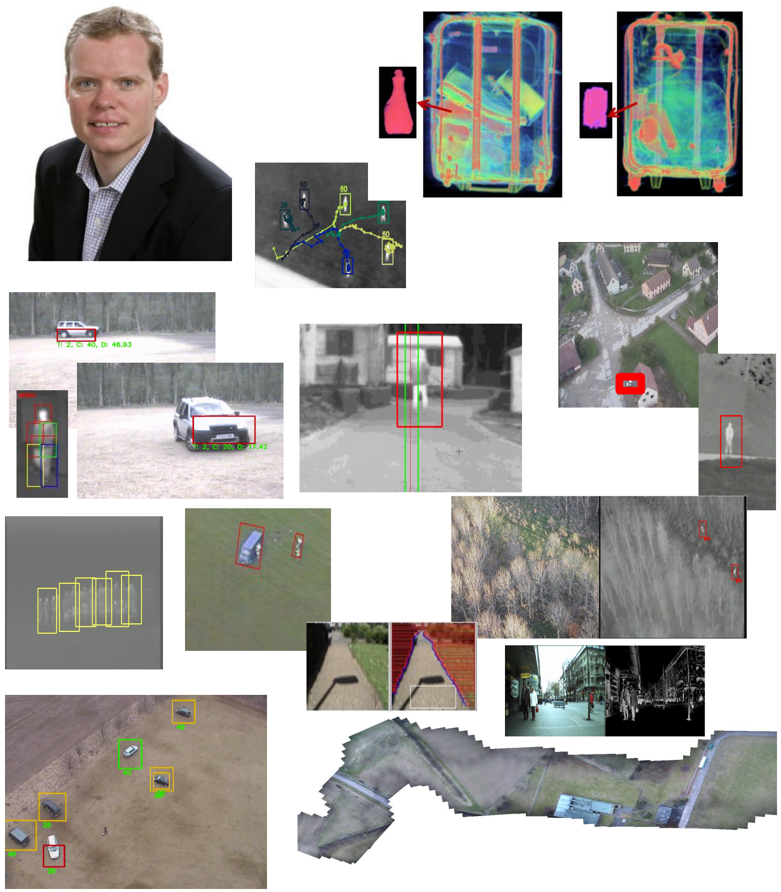
Cranfield UNIVERSITY

Thanks ...

www.cranfield.ac.uk/~toby.breckon/mltutorial/

toby.breckon@cranfield.ac.uk

Thanks ...



www.breckon.eu/toby/mltutorial/

toby@breckon.eu