Feature Classification using Machine Learning within Computer Vision

a whirlwind tour of key concepts for the uninitiated

integrating Vision & Language

Toby Breckon

School of Engineering and Computing Sciences

Durham University

www.durham.ac.uk/toby.breckon/mltutorial/

toby.breckon@durham.ac.uk

Image Understanding ...

- *“What does it mean, to see?*

The plain man's answer (and Aristotle's, too) would be, to know what is where by looking.” D. Marr, Vision (1982)

Image Understanding ...

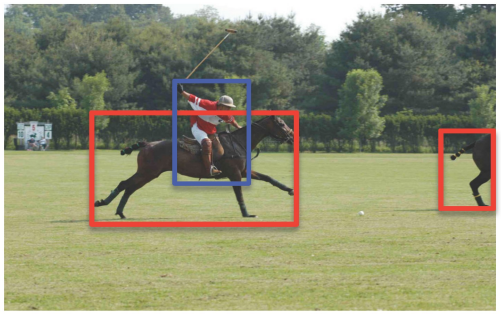
Test image

Scene:
Open-country

Event: Polo

Scene and Event Categorization

Depth estimation



Object detection

Geometric layout

Saliency detection

Source: http://chenlab.ece.cornell.edu/projects/FECCM/

VERY high level *(indeed!)*

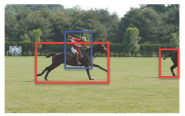
Feature Extraction & Representation

Decision Making

Scene: Open-country

Event: Polo

Scene and Event Categorization



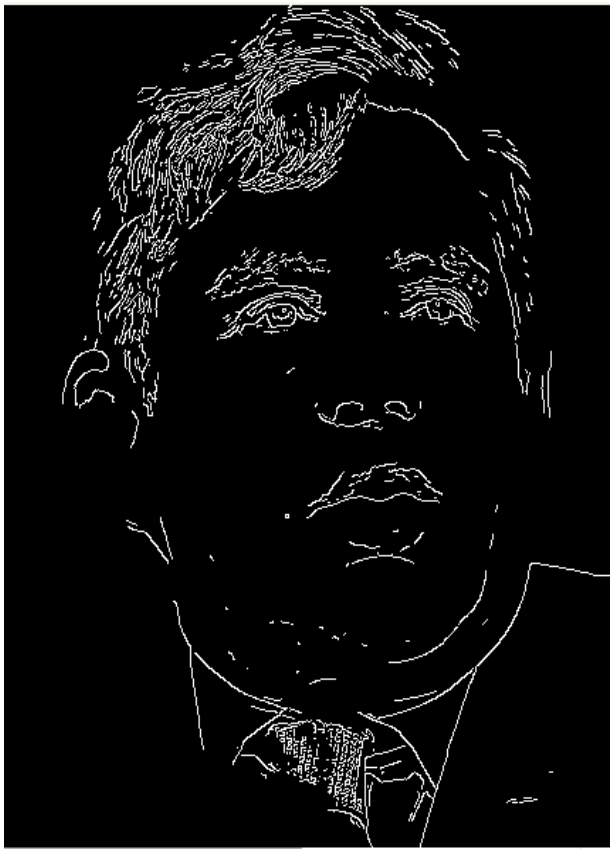
Object detection

Image

Understanding

Where we are today, with image features ….

Image Features



- Contemporary image features use **edge based information**
- **Colour provides context**, but is less important

Why … ? *(in general)*

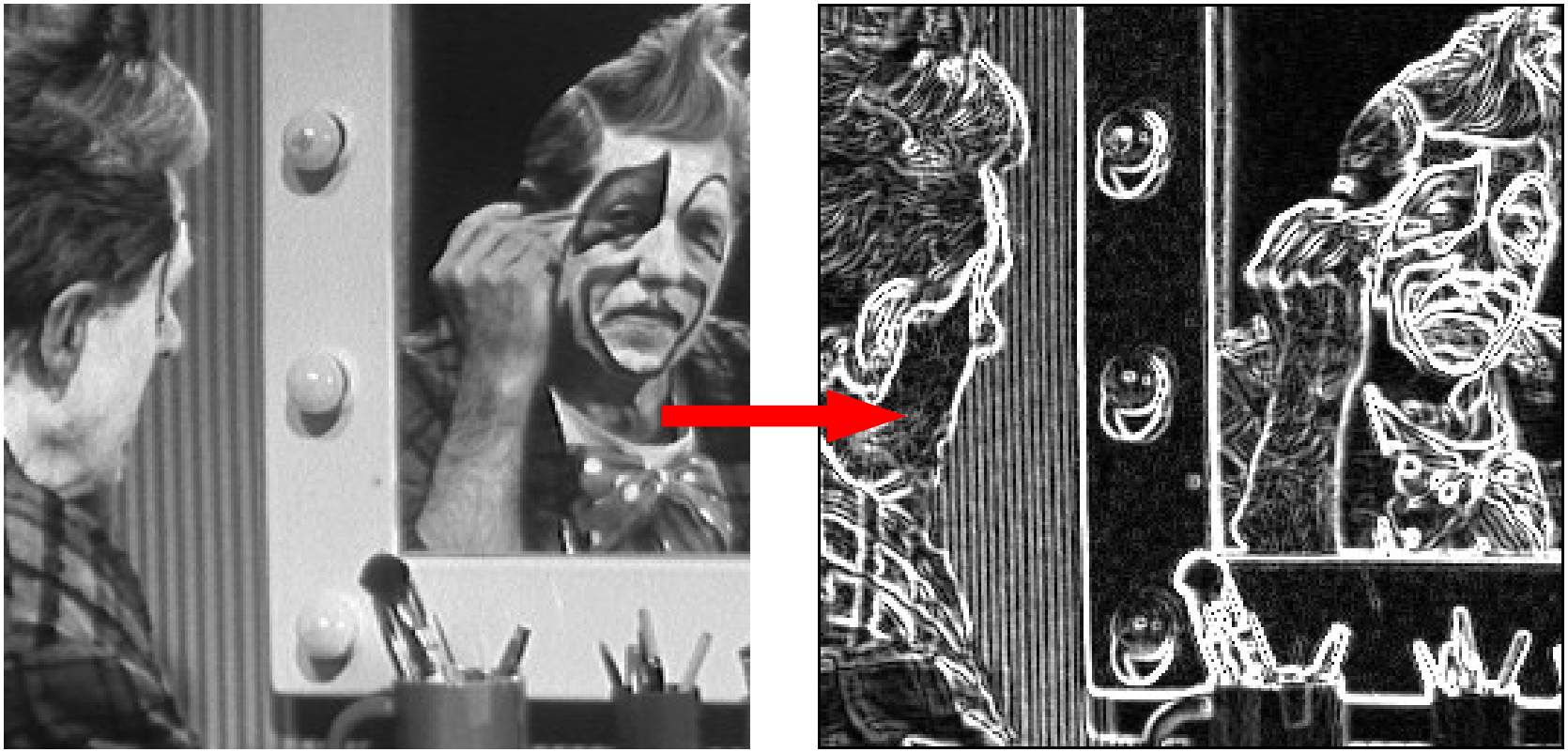
- Shape (edge information) has greater invariance intra-class variability …. (and lighting)

- Colour provides contextual information ….

  e.g. grass is green, (probabilistically) horses occur on grass

  e.g. wheels are round and dark, (probabilistically) cars have wheels

Edges are *gradients in the image*



source: HIPR2 © 2003/4

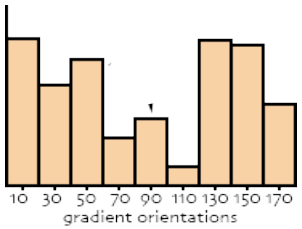
- If the image is $I(x,y)$ then the basic idea is to compute

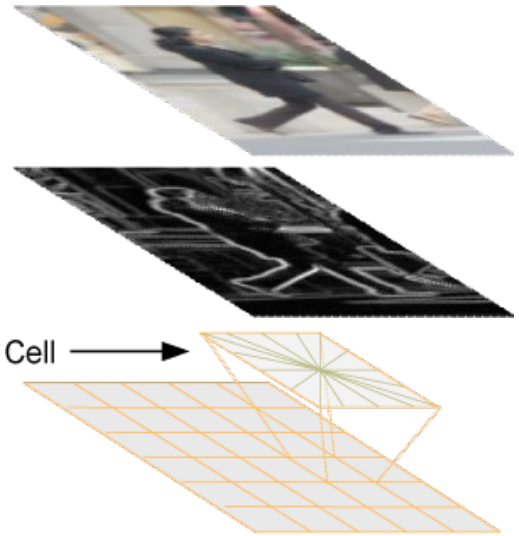
$$\nabla I(x,y) = \hat{\mathbf{x}}\frac{\partial}{\partial x}I(x,y) + \hat{\mathbf{y}}\frac{\partial}{\partial y}I(x,y)$$

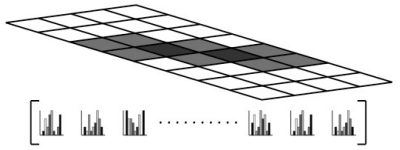
@ every pixel we get
{edge strength, edge orientation θ}

Gradient Based Features - Histograms

- **Calculate localized Histograms of Gradient** (Edge) **Orientations**







feature vector,

$V_{HOG} = \{\ldots, \ldots, \ldots.\}$

(dimension typically ~4000+)

- **Histogram of Orientated Gradients (HOG),** *[Dalal / Triggs, 2005]*
  - *localised distribution of gradient orientations from simple edge response filters*

HOG feature descriptor (visualisation)

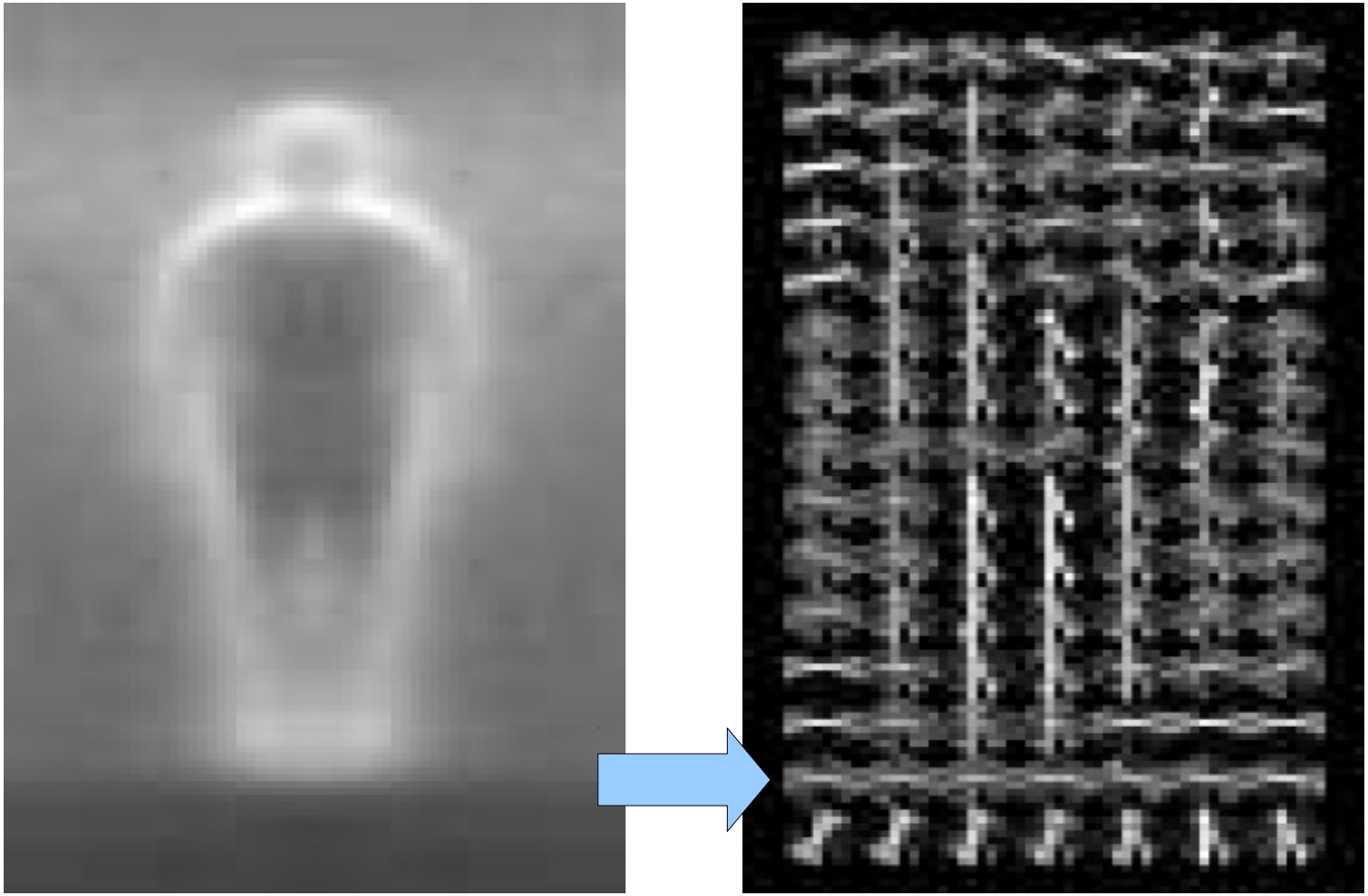
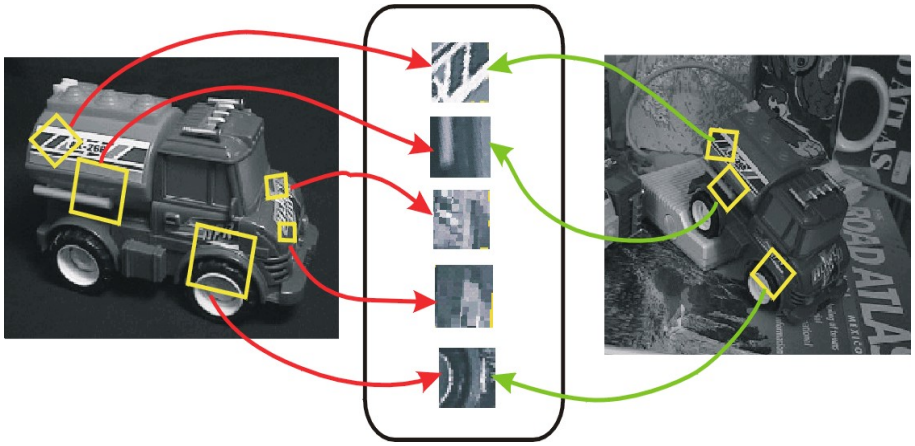


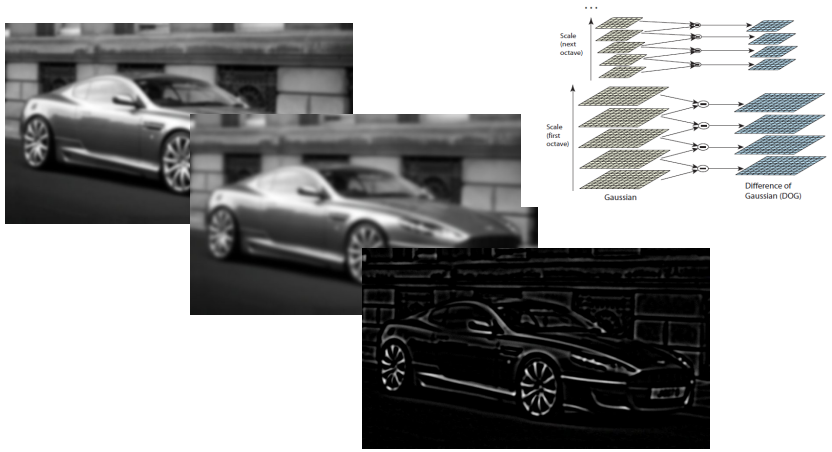
Image (gradient response)

Cell Histograms within HOG

Gradient Based Features - Points

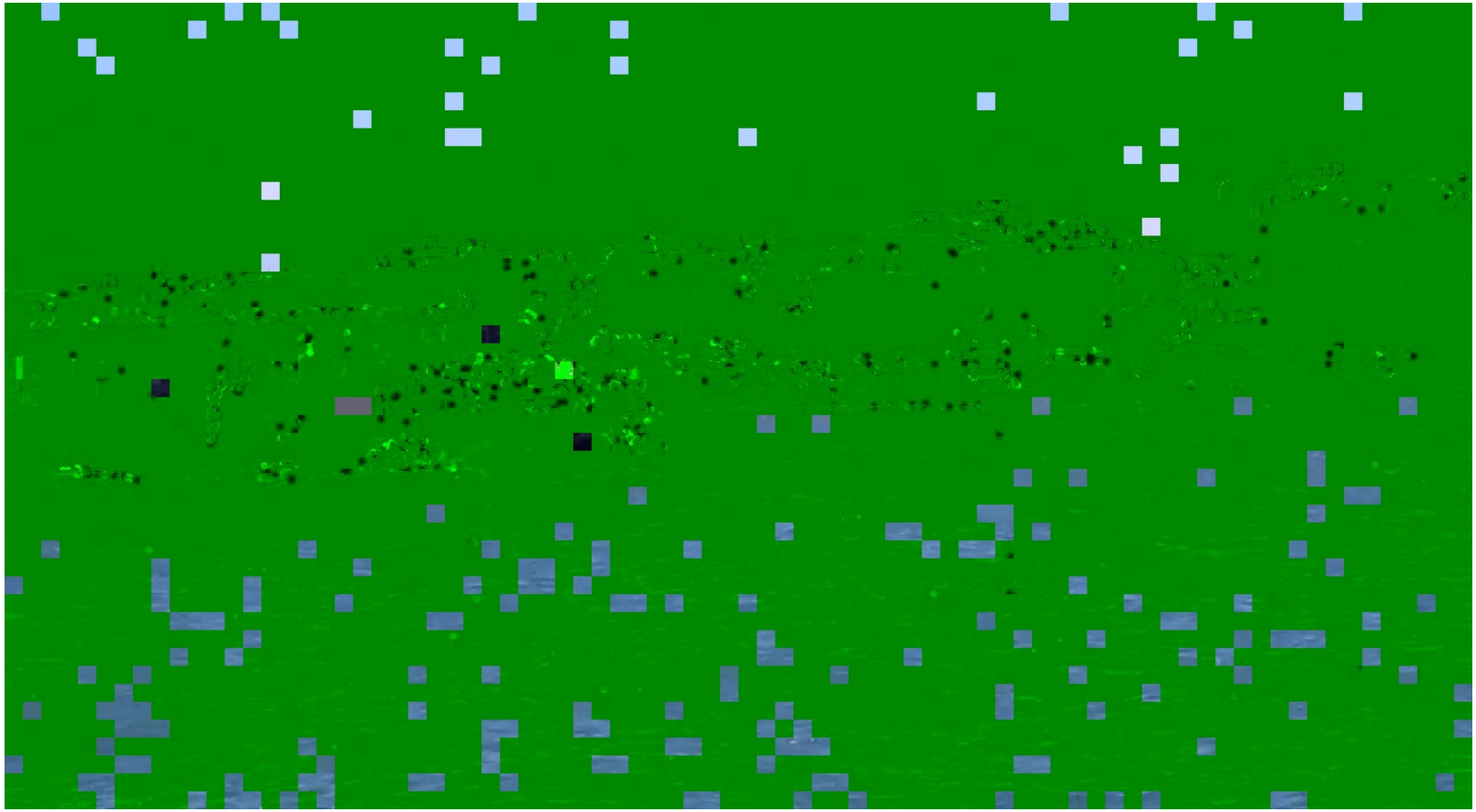
- Localized image regions (**feature points**) that are **invariant to transformations**
  - **geometric invariance**: translation, rotation, scale
  - **image intensity invariance**: brightness, exposure, *noise* ….



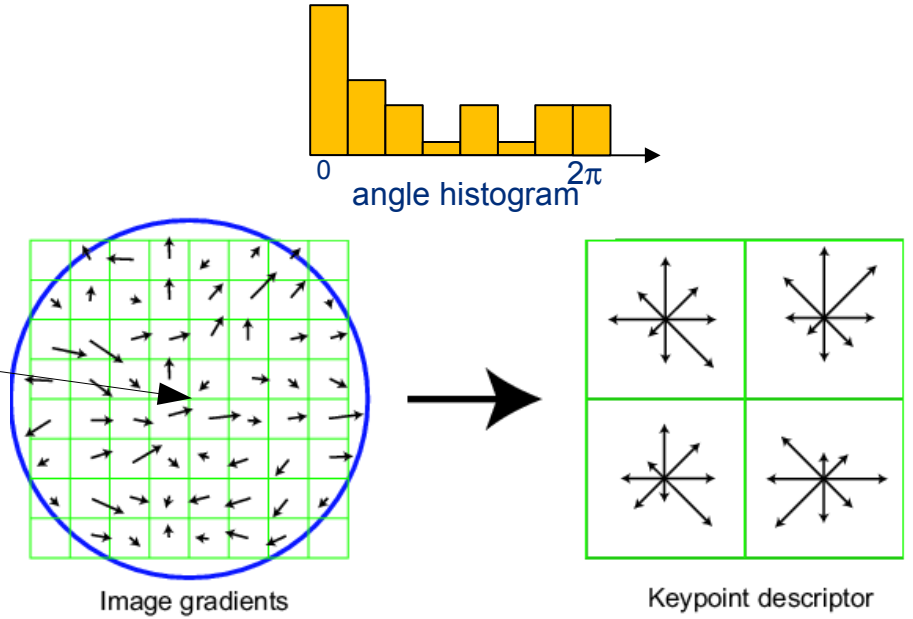


- Find **locally distinctive points** in image (e.g. corners) that are invariant under transformation

Feature Points



Feature Point Descriptors



→ describe region around each point using vector of gradient orientations (e.g. 64D or 128D vector)

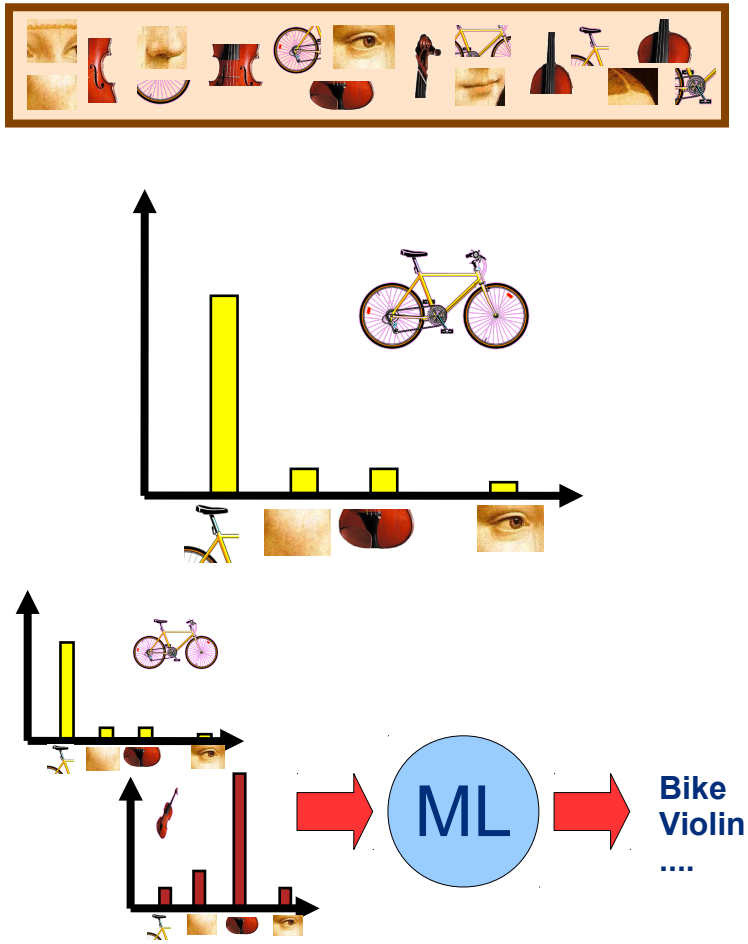
- **Scale Invariant Feature Transform (SIFT)** *[Lowe, 2004]*
  - many, many variants on concept exist (SURF, BRISK, BRIEF, ORB, FREAK, GIST ….)

(Visual) Feature Space

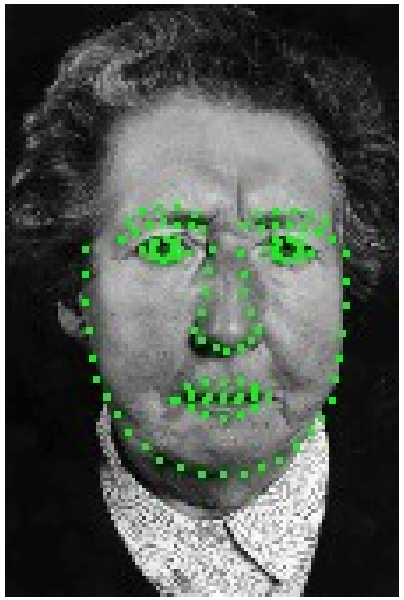
e.g. 128-D SIFT space

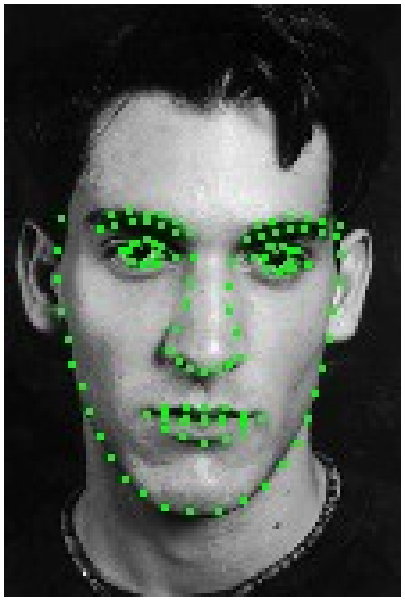
… → Bag of Visual Words

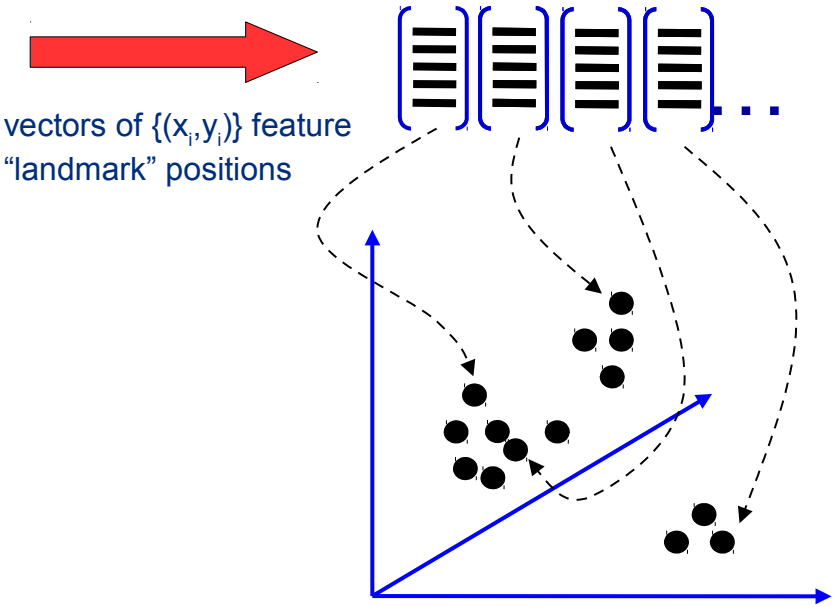
- **Cluster Feature Descriptors** in R^n
  - over example *(training)* images
  - K-means clustering
- Cluster *"membership"* for a given object example creates a **histogram of feature occurrence** (using a distance measure to individual clusters)
- Use **histogram of feature occurrence** as inputs to a machine learning classification algorithm



Gradient Based Features – Shape Models





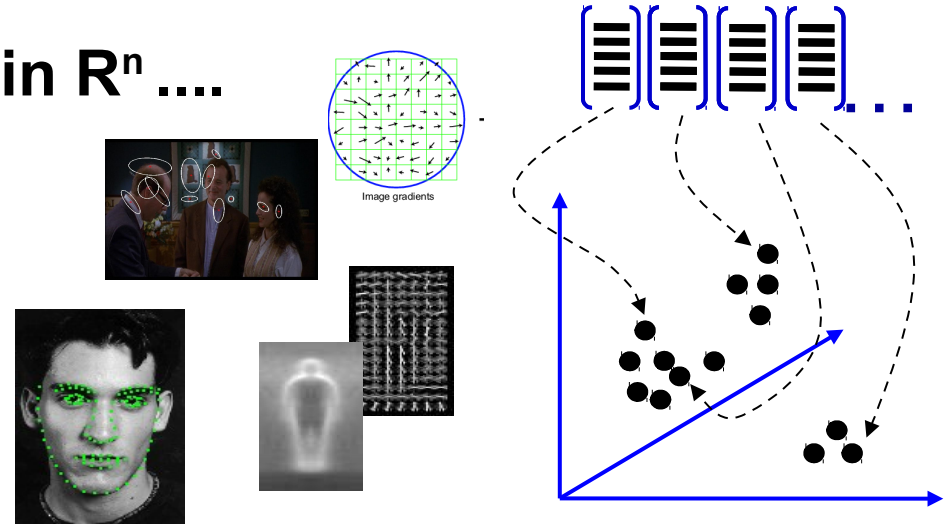


- combine shape and gray-level variation in single statistical appearance model
- shape **landmark positions derived from local gradient**

[Cootes, Taylor et al. 1995 → present]

Comparing Features

- Our features are **vectors in R^n ....**

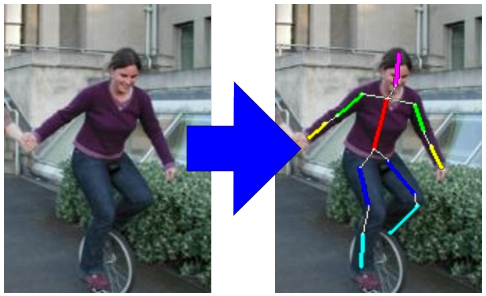


- Compare using:
  - numerical **distances** (Manhattan, Euclidean / L_1, L_2, L_n, *"rootSIFT"* *[Arandjelovic, 2012]*)
  - histograms of feature occurrence (**bag of visual words**)
  - **statistical comparison** of distributions (histograms)
    - Chi-squared, intersection, Bhattacharyya (image specific)

In practice, **complex feature representations** in **complex numerical spaces** are **difficult to reason with directly …..**

***Machine** Learning ?*

- **Why** Machine Learning?
  - we cannot program everything
  - some tasks are difficult to define algorithmically
    - especially in computer vision
      …. visual sensing has few rules



- Well-defined learning problems ?
  - easy to learn Vs. difficult to learn
    - ..... **varying complexity of visual patterns**

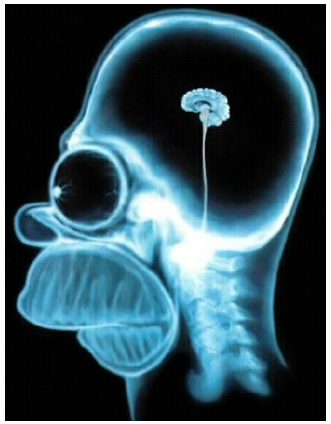
- An example: learning to recognise objects ...

Image: DK

Learning ? - in humans

Learning ? - in computers

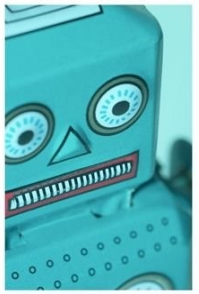
Learning …



- **learning** *verb*
  - –the *activity* of **obtaining knowledge**
  - –knowledge **obtained by study**

– **English Dictionary, Cambridge University Press**

Machine **Learning**



- **Definition:**

  – *A set of methods for the automated analysis of structure in data. …. two main strands of work, (i) **unsupervised** learning …. and (ii) **supervised** learning.*

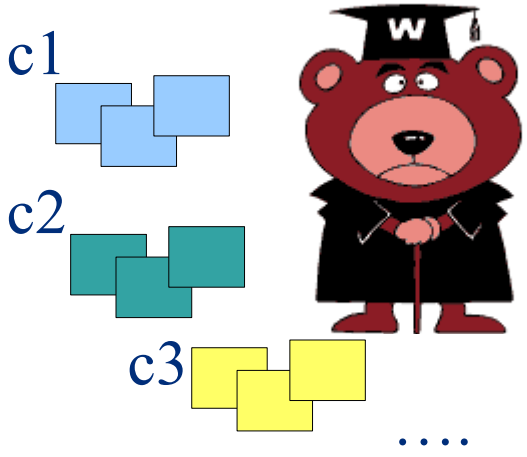
  *….similar to … **data mining**, but … **focus .. more on autonomous machine performance**, ….*

  rather than enabling humans to learn from the data.

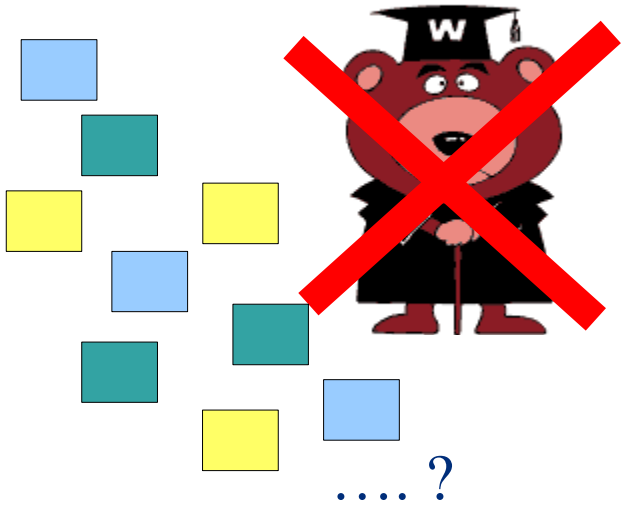
[Dictionary of Image Processing & Computer Vision, Fisher et al., 2014]

Supervised Vs. Unsupervised

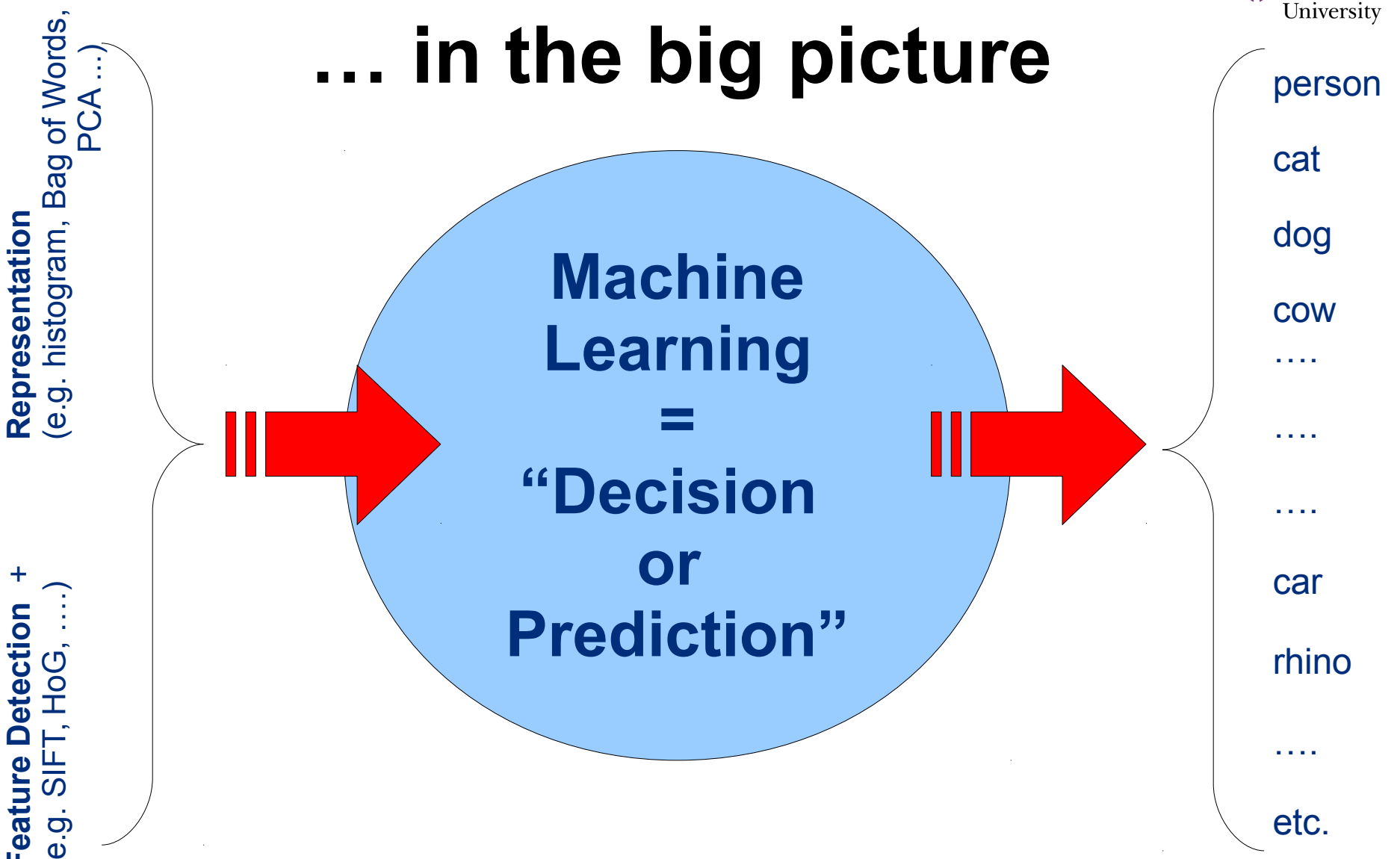
- **Supervised**
  - **knowledge of output -** learning with the presence of an "expert" / teacher
    - data is **labelled** with a class or value
    - **Goal:** predict class or value label
      - e.g. Neural Network, Support Vector Machines, Decision Trees, Bayesian Classifiers ....



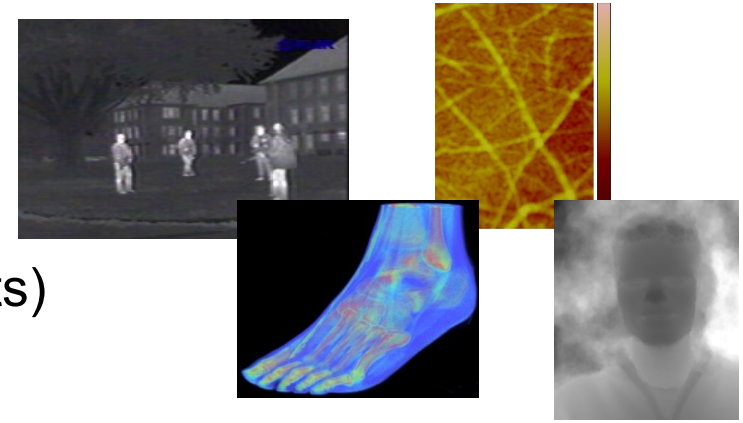
- **Unsupervised**
  - **no knowledge of output** class or value
    - data is **unlabelled** or value un-known
    - **Goal:** determine data patterns/groupings
  - Self-guided learning algorithm
    - (internal self-evaluation against some criteria)
    - e.g. k-means, genetic algorithms, clustering approaches ...



… in the big picture



ML Input "Examples" in Comp. Vis. ...

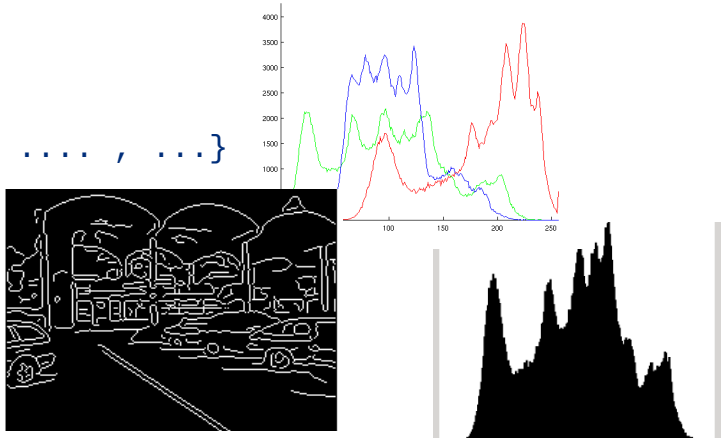


- **"Direct Sample" inputs**
  - Pixels / Voxels / 3D Points
  - sub-samples (key-points, feature-points)

… i.e. **"samples"** direct the imagery

- **Feature Vectors**
  - shape measures
  - edge distributions
  - colour distributions
  - texture measures / distributions

```
V = { 0.103900,
      120.102,
      30.10101, .... , ...}
```



[…. SIFT, SURF, HOG …. etc.]

… i.e. calculated **summary** "numerical" **descriptors**

Common ML Tasks in Comp. Vis. ...

- Object **Classification**
  what object ?



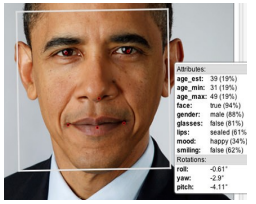

http://pascallin.ecs.soton.ac.uk/challenges/VOC/

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

{people | vehicle | … intruder ….}

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

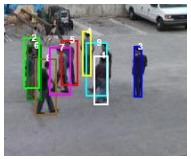

{face | vehicle plate| gait …. → biometrics}



{gender | type | species | age ……}

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

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

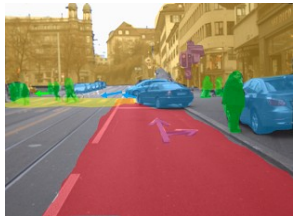



Types of ML Problem

- **Classification**
  - Predict (classify) sample → discrete set of class labels
    - e.g. classes {object 1, object 2 … } for recognition task
    - e.g. classes {object, !object} for detection task

- **Regression** **(traditionally less common in comp. vis.)**
  - Predict sample → associated numerical value (variable)
    - e.g. distance to target based on shape features
  - Linear and non-linear attribute to value relationships

- **Association or clustering**
  - grouping a set of instances by attribute similarity
    - e.g. image segmentation



[Ess et al, 2009]

Regression Example – Head Pose Estimation

- Input: image features (HOG)
- Output: { yaw | pitch }
- *varying illumination + vibration*

[Walger / Breckon, 2014]

Learning in **general**, from **specific examples**

- N.B.
  - E is **specific** : a specific **example** *(e.g. female face)*
  - T is **general** : a general **task** *(e.g. gender recognition)*

Program **P "learns"** task **T** from set of **examples {E}**

Thus if P improves learning must be of the following form:

– **"to learn a general**
[ability|behaviour|function|rules**]**
from specific examples"

…. ?

A simple learning example ....

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

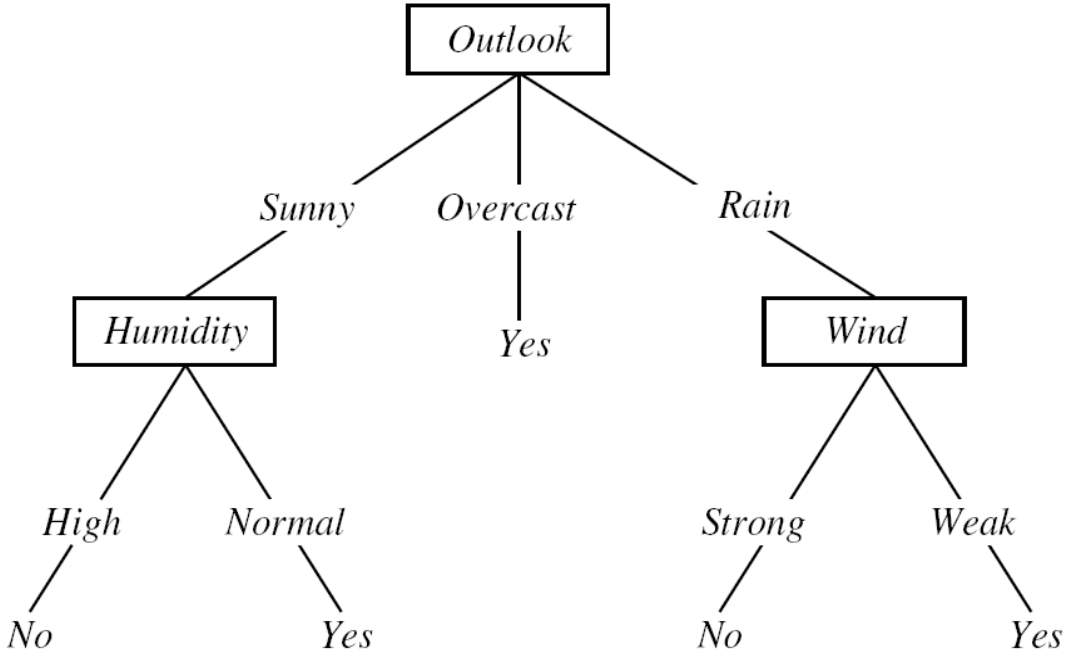
Attributes **Classification**

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 Trees

- Attribute based, decision logic construction
  - boolean outcome
  - discrete set of outputs

Safe conditions to fly ?



Decision Trees

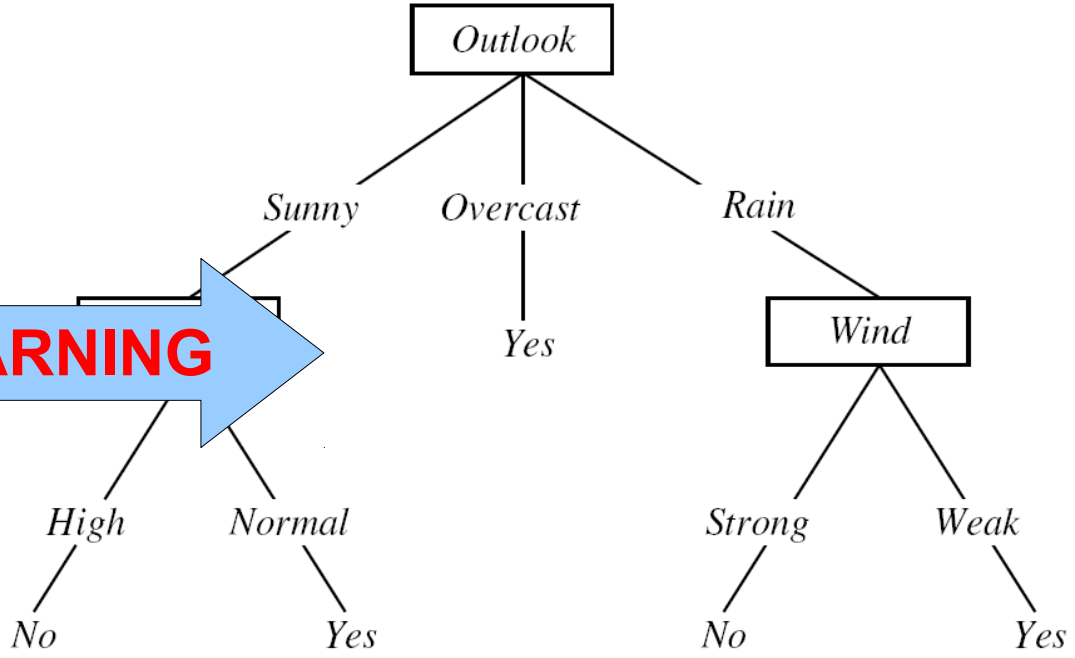
- Set of *Specific* **Examples** ...

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	
D12	Overcast	Mild	High	Strong	
D13	Overcast	Hot	Normal	Weak	
D14	Rain	Mild	High	Strong	No

(training data)

Safe conditions to fly ?

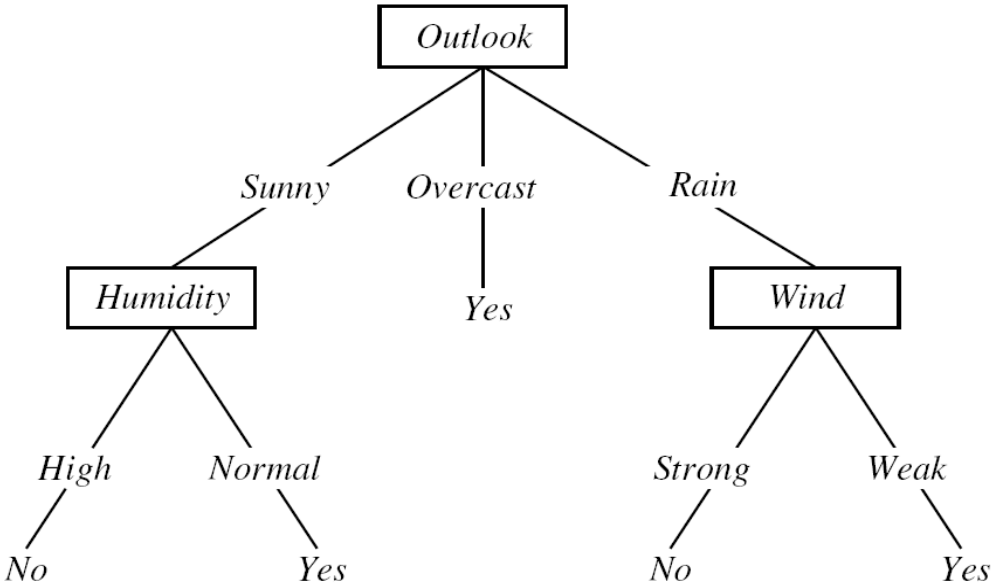
LEARNING



GENERALIZED RULE

Decision Tree **Logic**

(Outlook = Sunny AND Humidity = Normal)
OR (Outlook = Overcast)
OR (Outlook = Rain AND Wind = Weak)

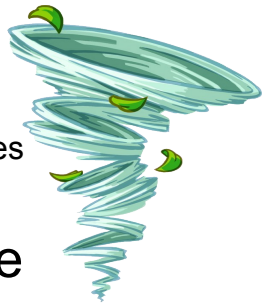


Growing Decision Trees

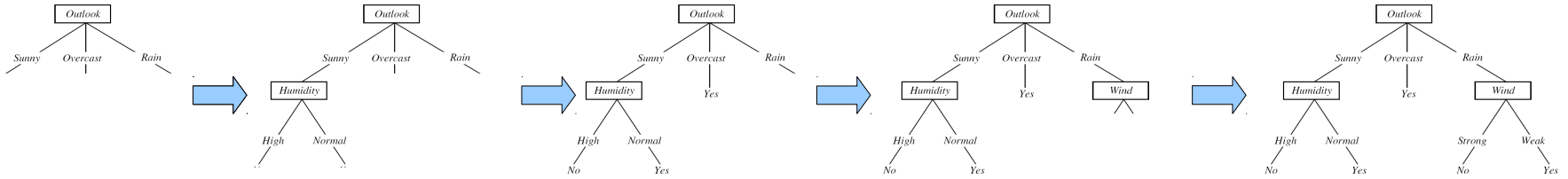
- **Construction** is carried out **top down** based on node splits that **maximise the reduction in the entropy** in each resulting sub-branch of the tree

[Quinlan, '86]

.. see extra slides



- **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

Extension : *Continuous Valued Attributes*

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

$$Temperature = 82.5$$

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

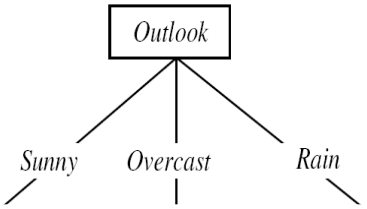
Temperature	40	48	60	72	80	90
Fly	No	No	Yes	Yes	Yes	No

Growing Decision Trees

(from data to "decision maker")

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	
D9	Sunny	Cool	Normal	Weak	
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

LEARNING



(training data)

- **"Safe conditions to fly ?"**

(Growing =) Learning *from* Data

- **Training data:** used to train the system
  - i.e. build the rules / learnt target function
  - **specific examples** *(used to learn)*

....

- **Test data:** used to test performance of the system
  - ***unseen*** by the system **during training**
  - also known as **validation data**
  - **specific examples** *(used to evaluate)*

....

e.g. facial gender classification

- **Training/test data** made up of **instances**
  - also referred to as examples/samples

N.B.

- **training data == training set**
- **test data == test/validation set**

Is it *really* that simple ?

Credit: Bekios-Calfa et al., *Revisiting Linear Discriminant Techniques in Gender Recognition*
IEEE Trans. Pattern Analysis and Machine Intelligence http://dx.doi.org/10.1109/TPAMI.2010.208

Must (Must, Must!)
avoid over-fitting …..

(i.e. over-learning)

Occam's Razor

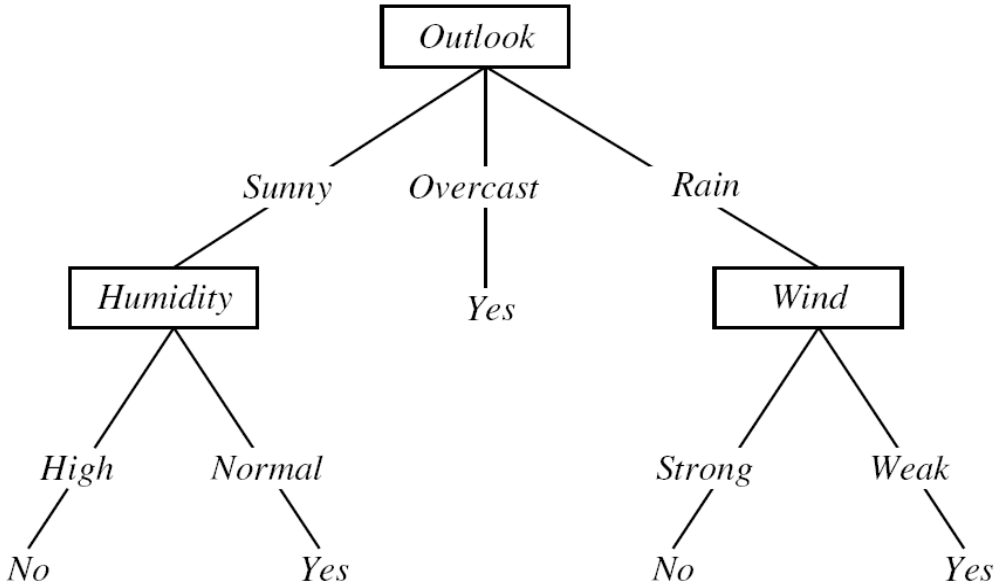
- Follow the principle of Occam's Razor
- Occam's Razor
  - “entia non sunt multiplicanda praeter necessitatem” **(latin!)**
  - “entities should not be multiplied beyond necessity” **(english)**
  - **“All things being equal, the simplest solution tends to be the best one”**
- **Machine Learning :** prefer the simplest {model | hypothesis | …. tree | network } that fits the data

14th-century English logician William of Ockham

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?

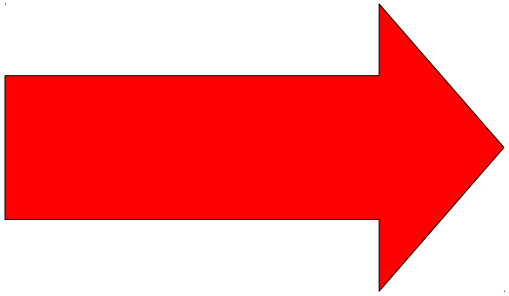


Problem of **Overfitting**

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

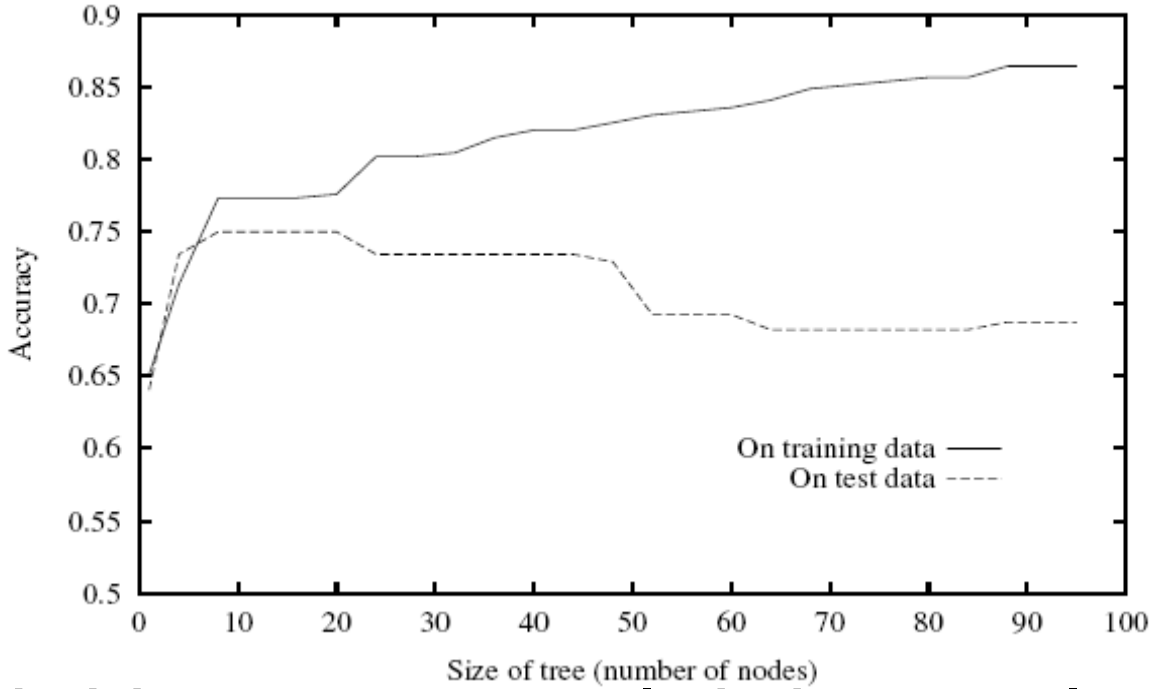
= wind!

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



Overfitting in general

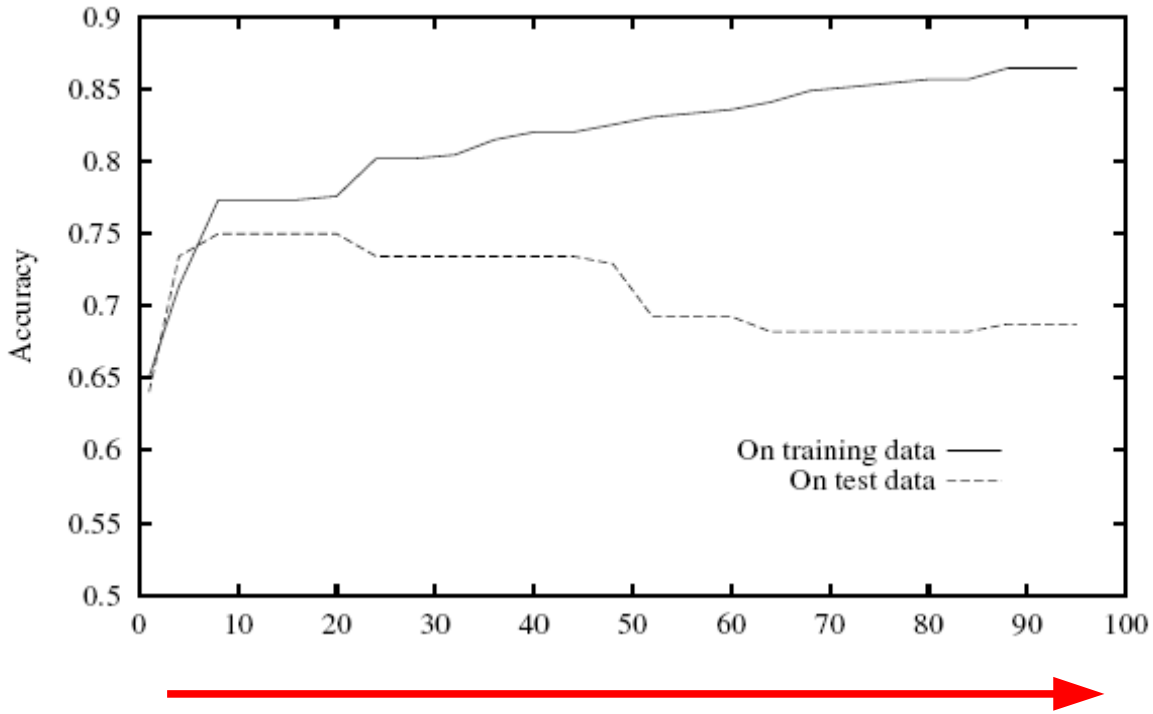
- **Performance** on the **training** data (with noise) **improves**
- **Performance** on the unseen **test** data **decreases**



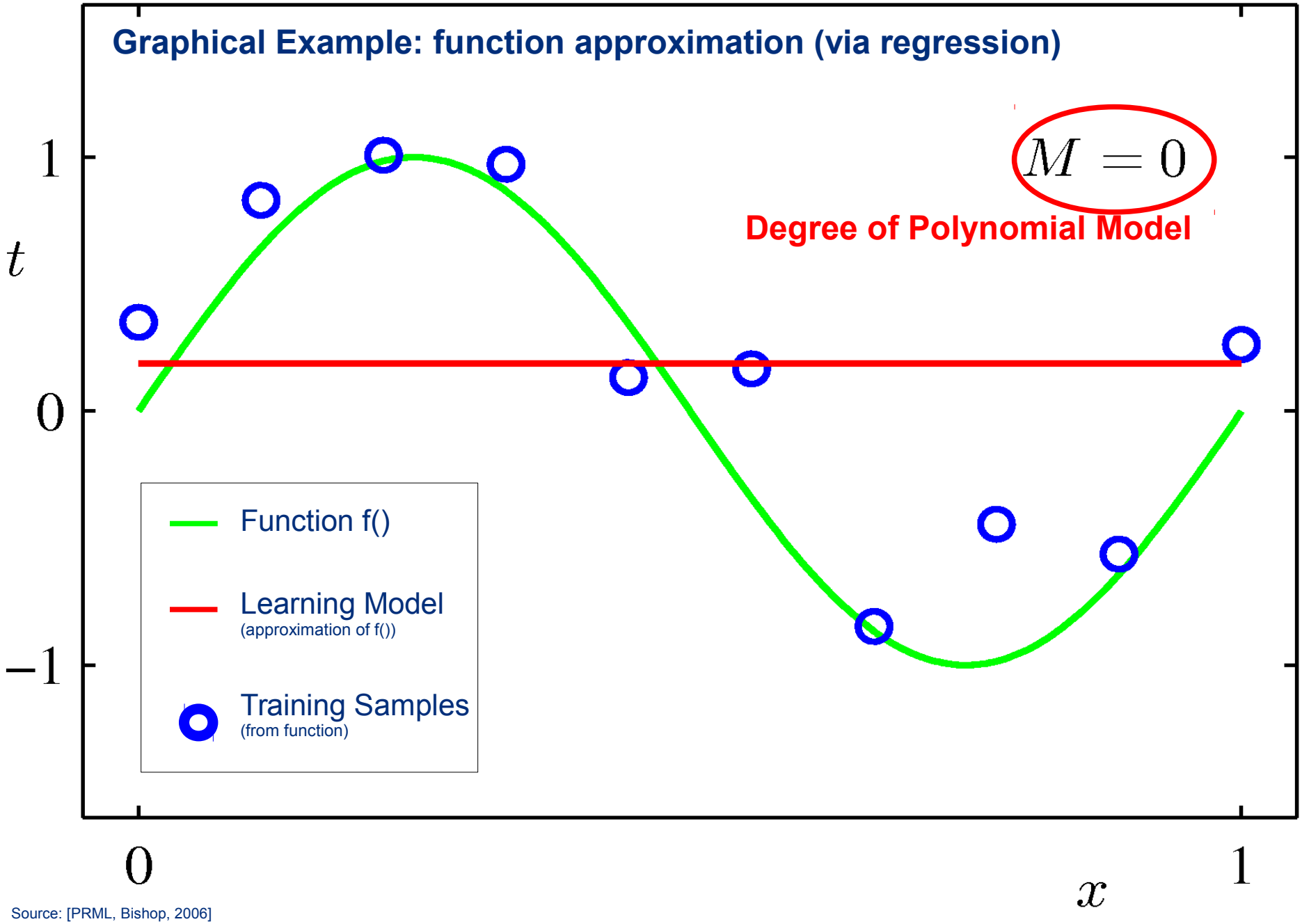
  - **For decision trees: tree complexity increases, learns training data too well! (over-fits)**

Overfitting in general

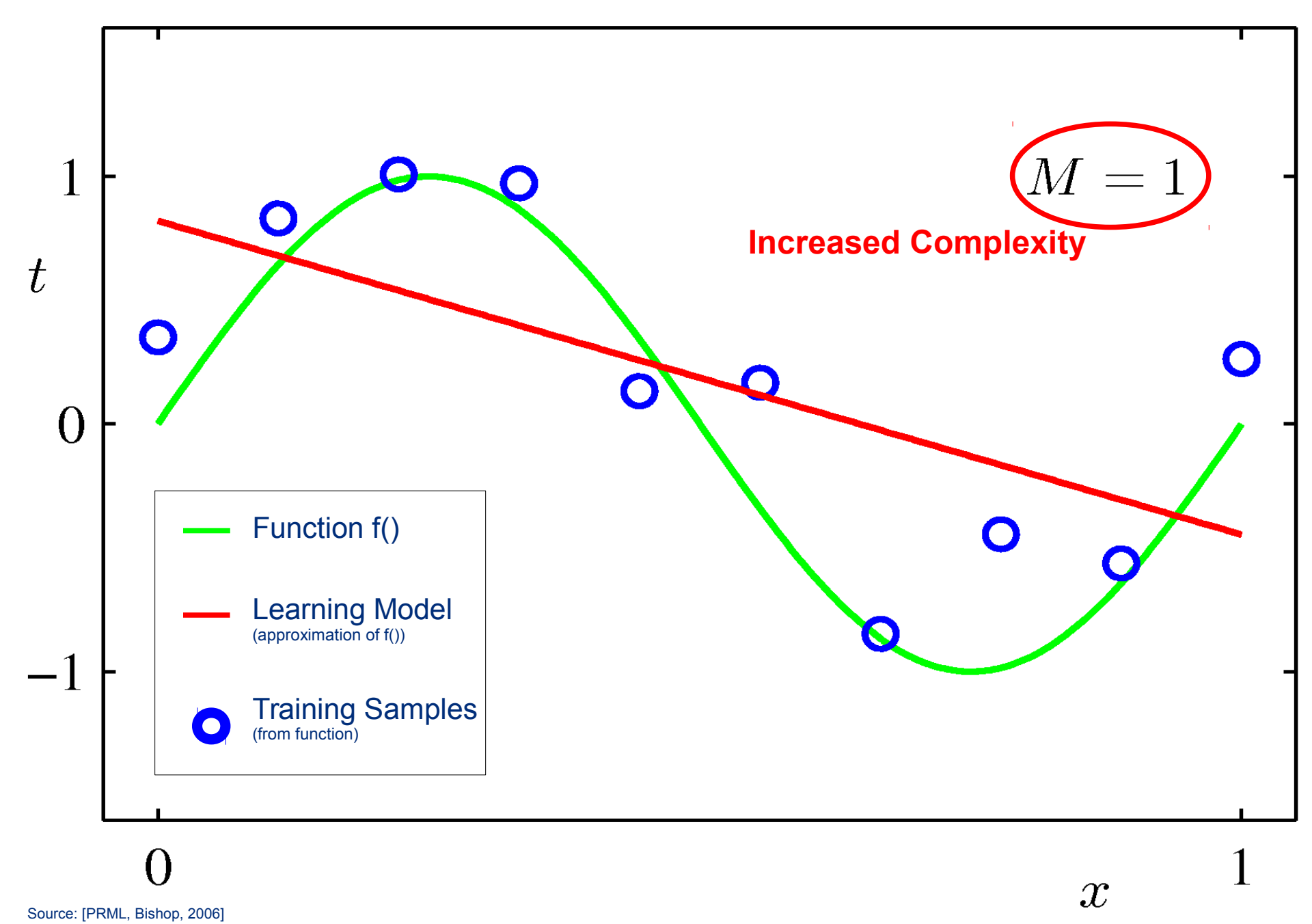
- **Hypothesis** is **too specific** towards training examples
- **Hypothesis not general enough** for test data



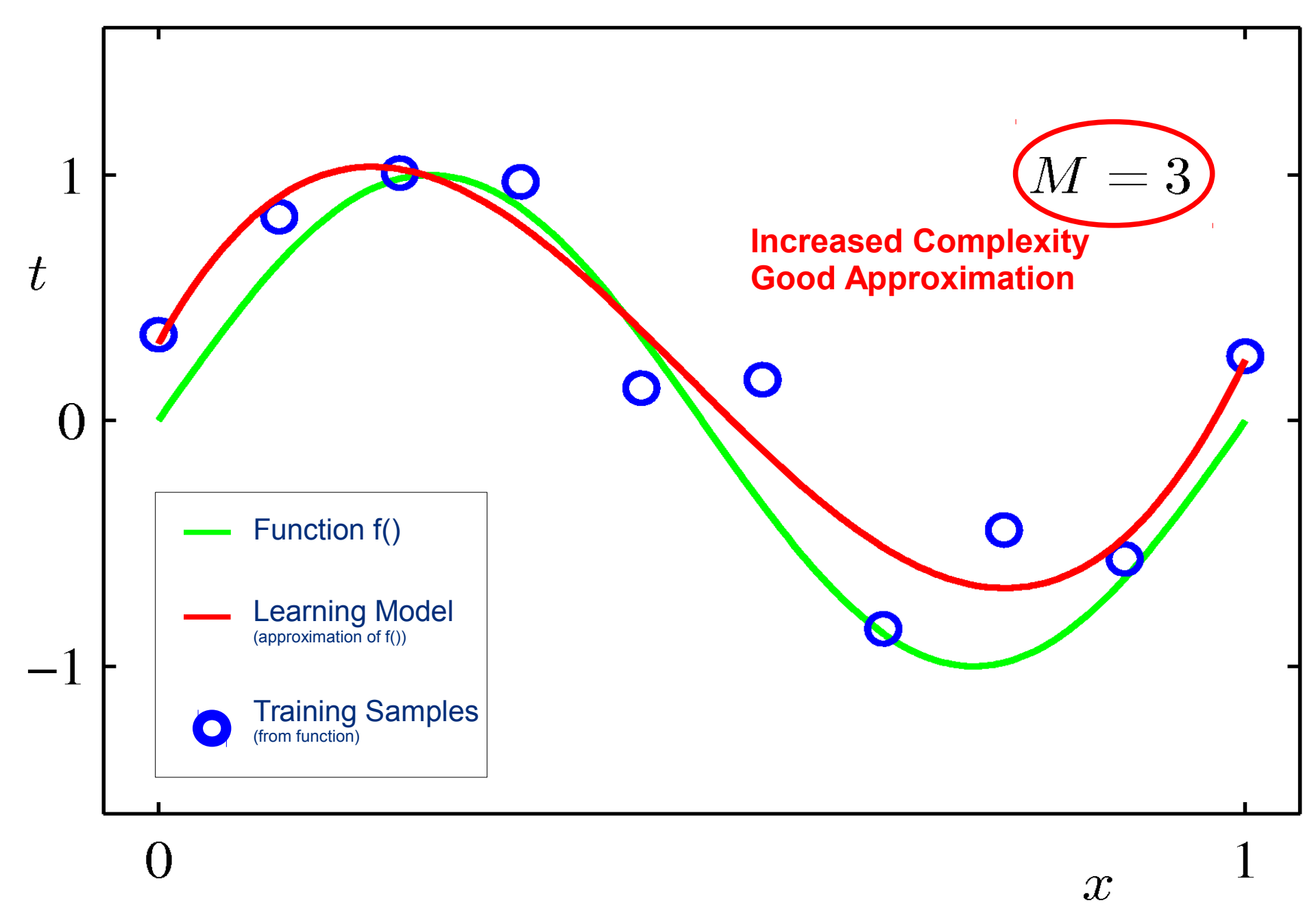
Graphical Example: function approximation (via regression)



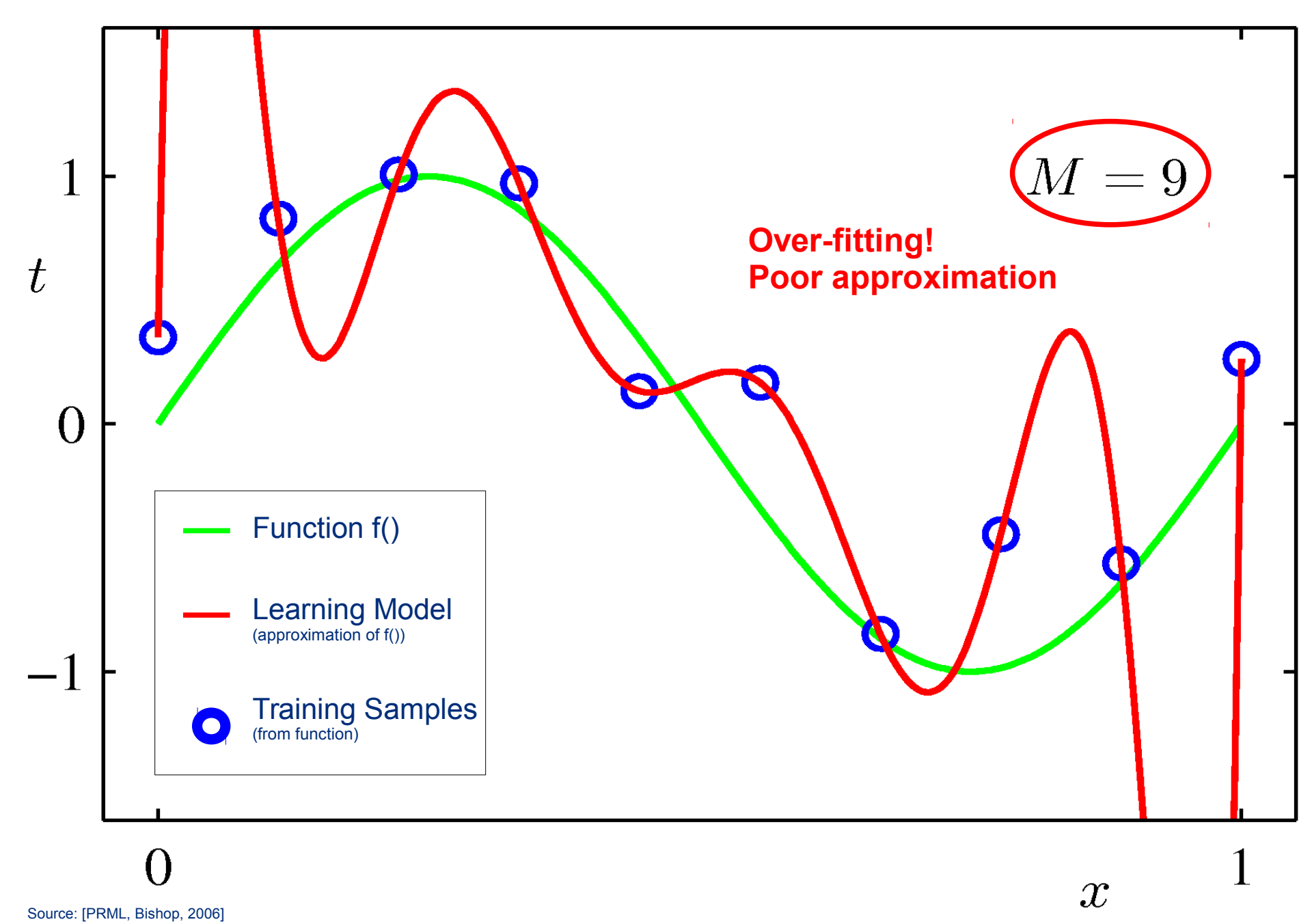
Source: [PRML, Bishop, 2006]



Source: [PRML, Bishop, 2006]



Source: [PRML, Bishop, 2006]

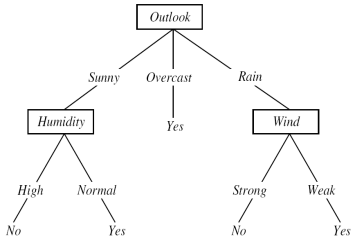


Source: [PRML, Bishop, 2006]

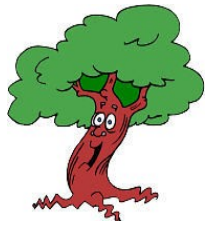
Avoiding Over-fitting

- **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 again)*



Fact 1: Decision Trees are **Simple**



Fact 2: Performance on Vision Problems is **Poor**

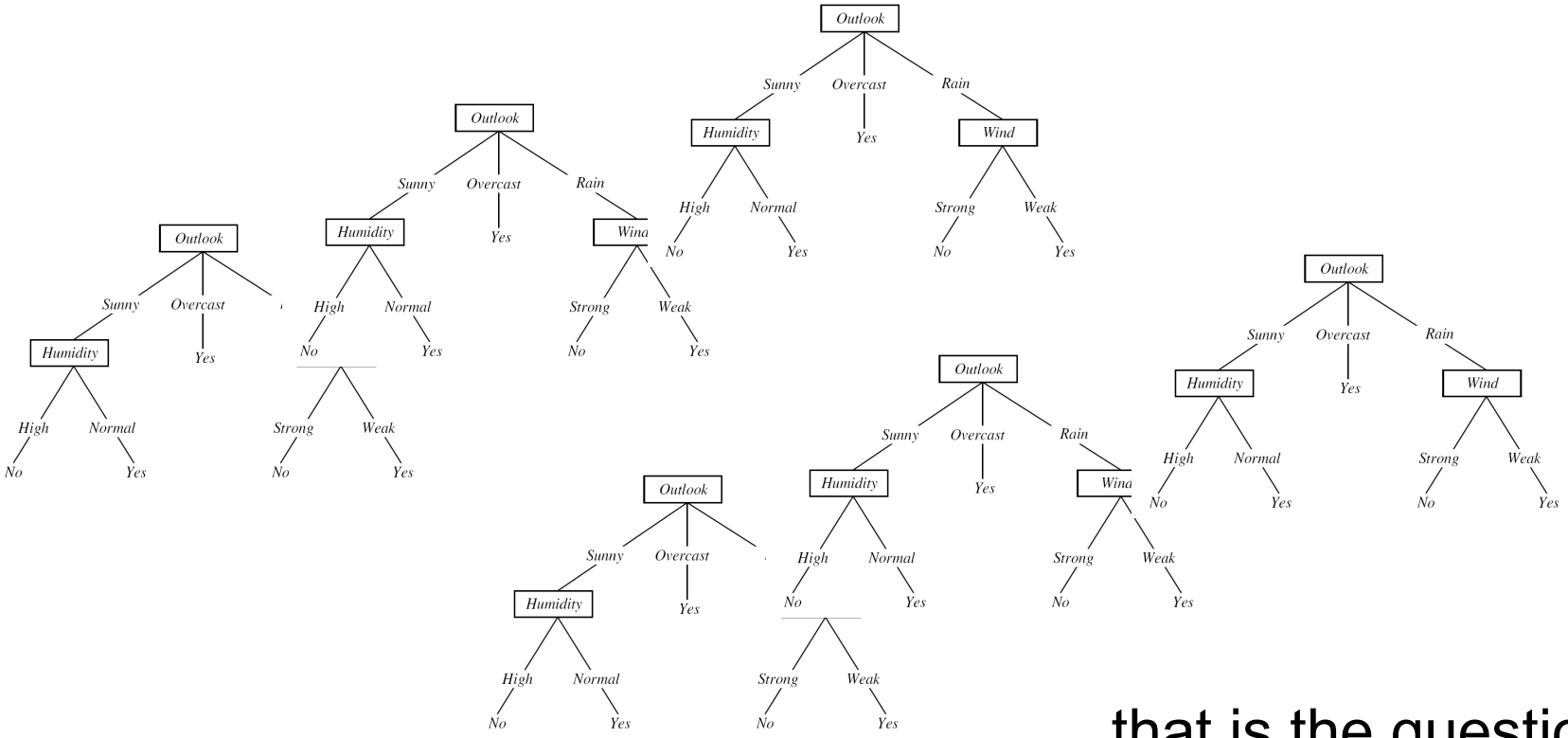
… unless we combine them in an **Ensemble Classifier**

Extending to **Multi-Tree Ensemble Classifiers**

- **Key Concept:** combining multiple 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:**
    - ***As per bagging***, 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!)*

Extending to **Multi-Tree Classifiers**

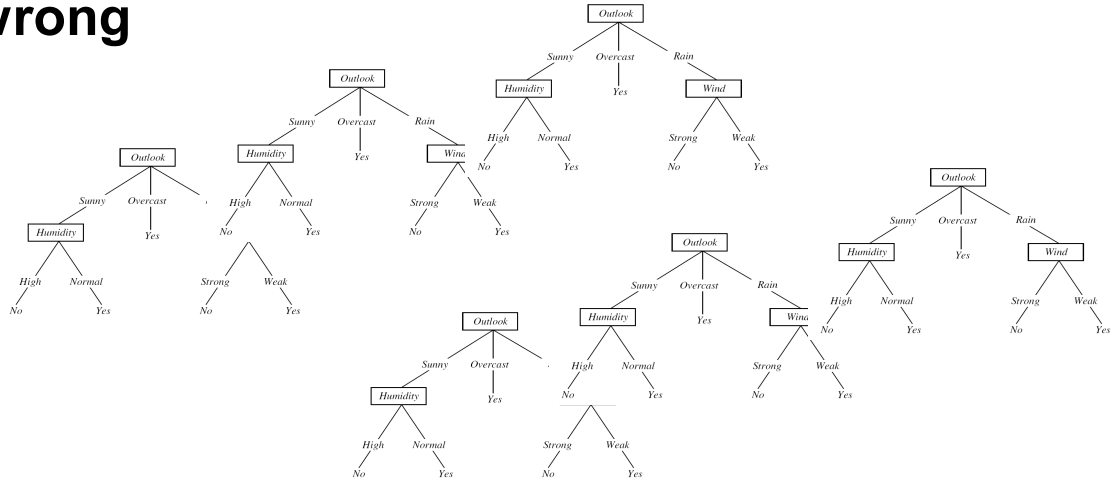
- To bag or to boost .....



..... that is the question.

Extending to **Multi-Tree Classifiers**

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

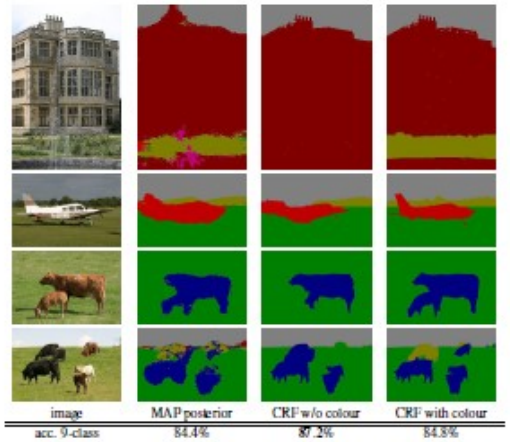


To bag or boost ? - *boosting generally works very well (but what about over-fitting ?)*

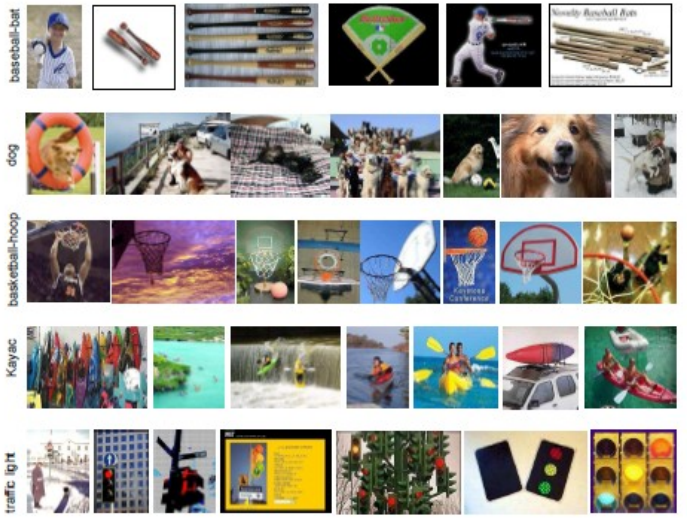
Decision Forests (a.k.a. Random Forests/Trees)

- **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]

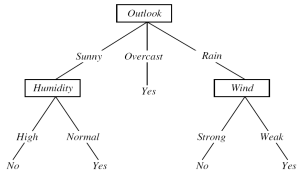


[Bosch 2007]

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

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

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



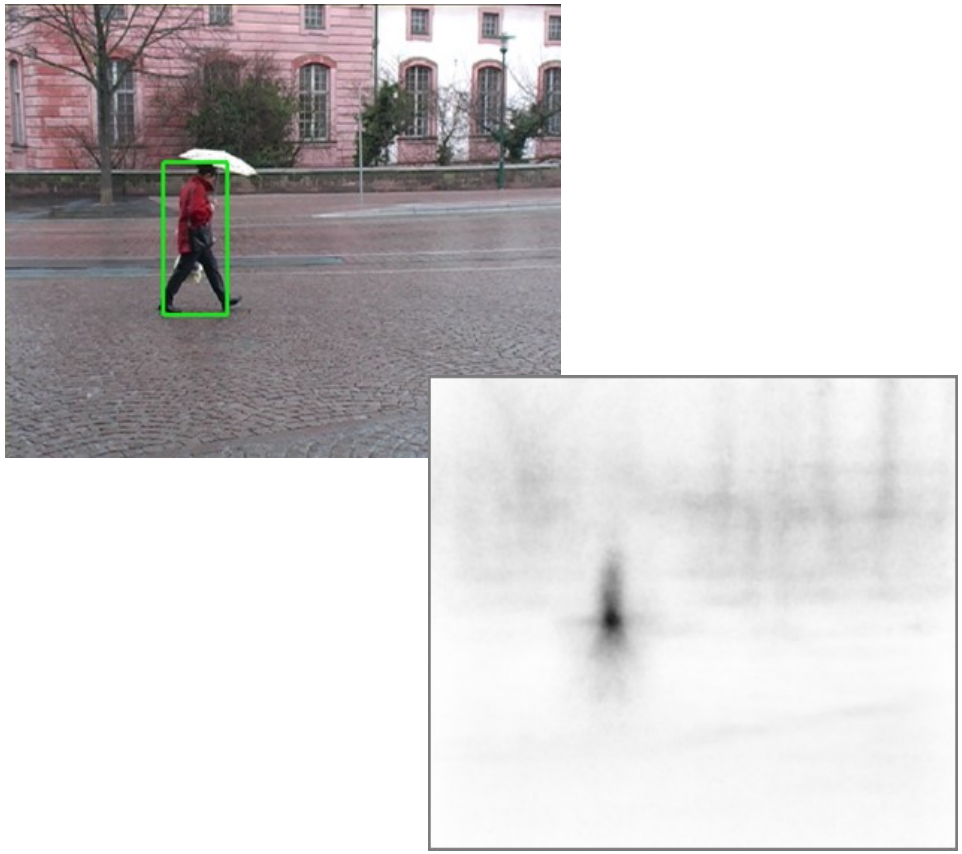
  - [alternatively weighted by number of training items assigned to the final leaf node reached in tree that have the same class as the sample (classification) or statistical value (regression)]

- **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"*

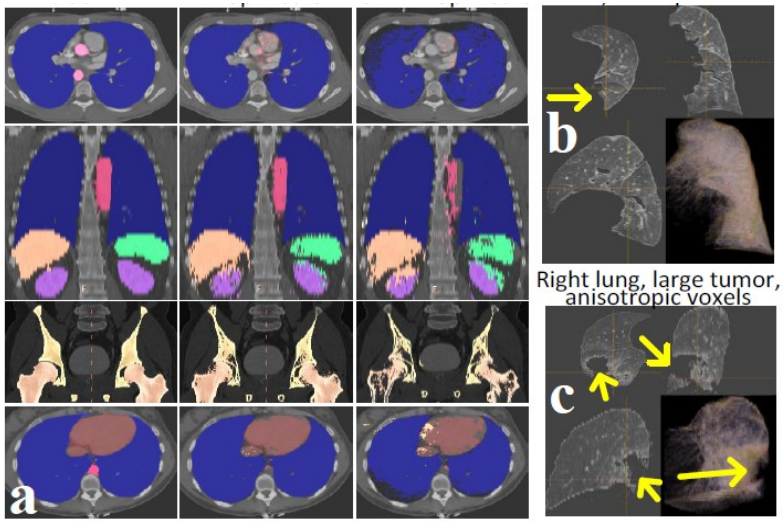
- **Drawbacks:** evaluation can be slow, lots of data for good performance, complexity of storage ...

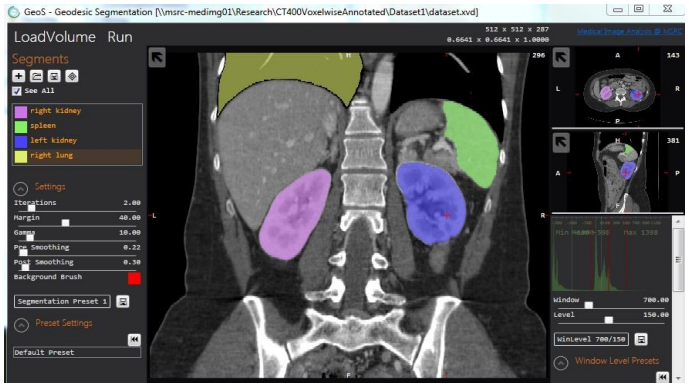
["Random Forests", Breiman 2001]

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

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!**

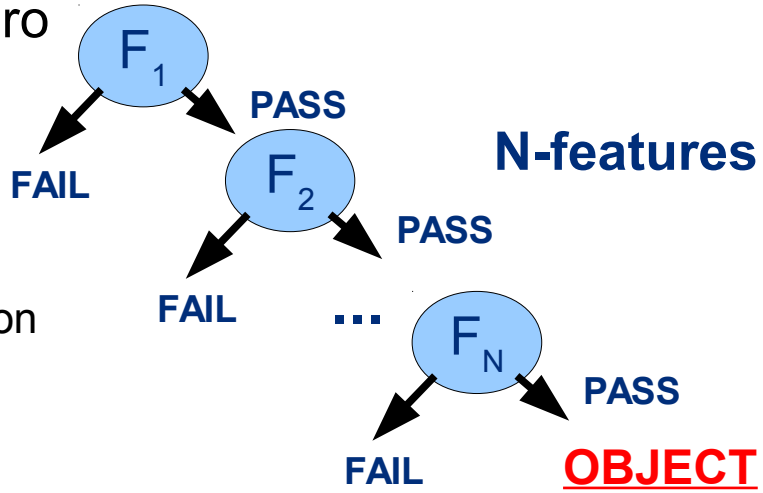
This is the concept of feature cascades.

Feature Cascading .....

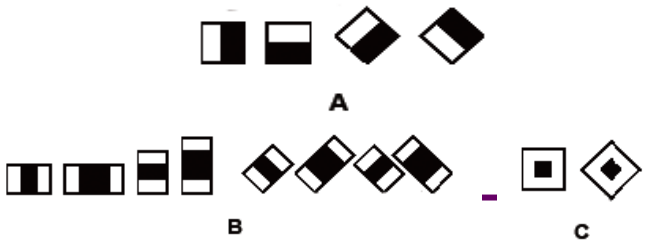
- Use boosting to **order image features from most to least discriminative for a given object** ....
  - allow high false positive per feature *(i.e. it's a weak classifier!)*

- 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]

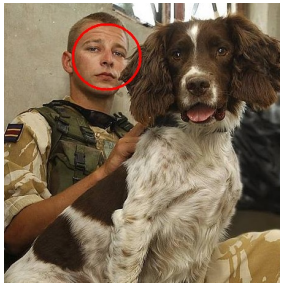


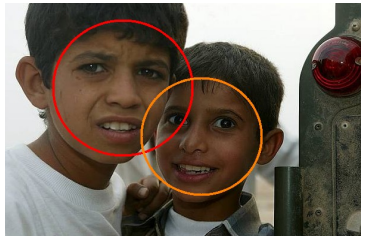
Haar Feature Cascades

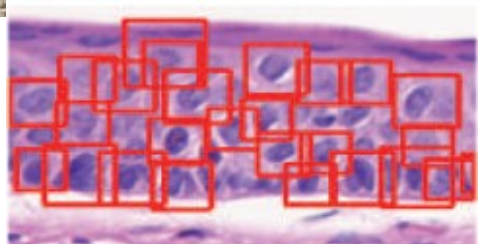
- **Real-time Generalised Object Recognition**

- **Benefits**
  - Multi-scale evaluation
    - scale invariant
  - Fast, real-time detection
  - “Direct” on image
    - no feature extraction
  - Haar features
    - contrast/ colour invariant

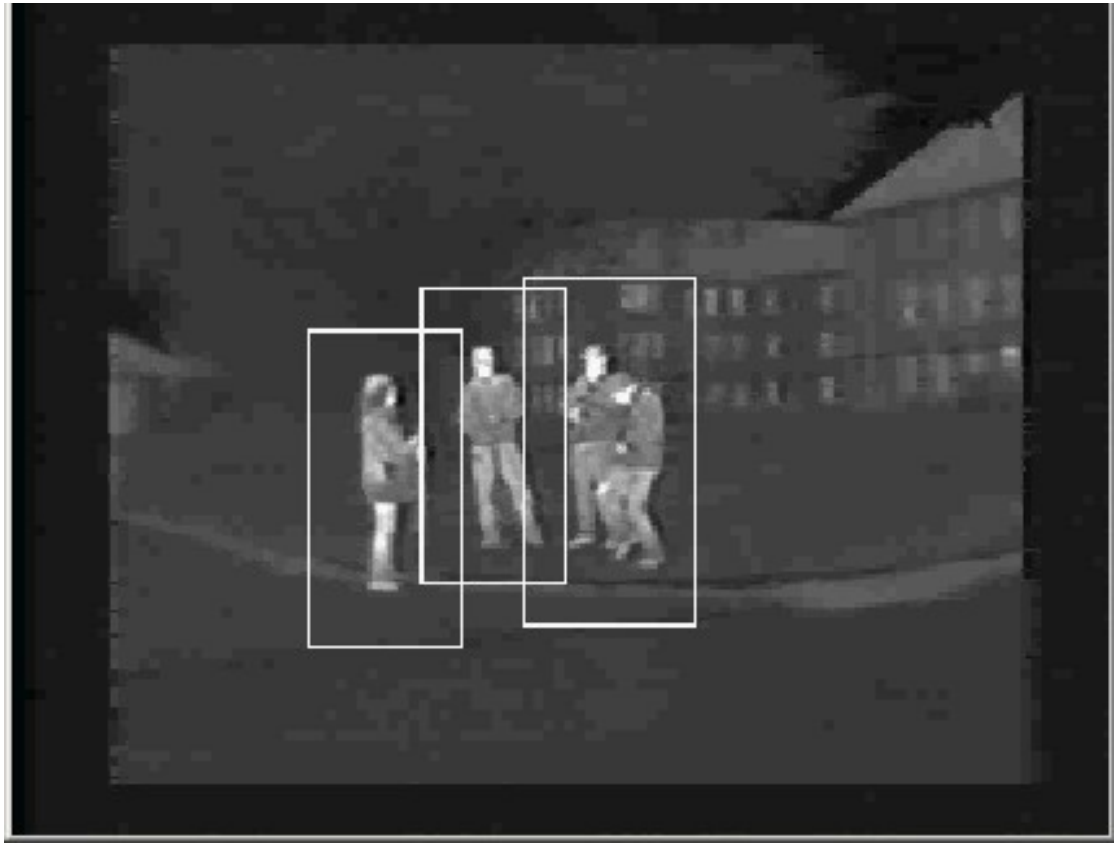
- **Limitations**
  - poor **performance on non-rigid objects**
  - object **rotation**

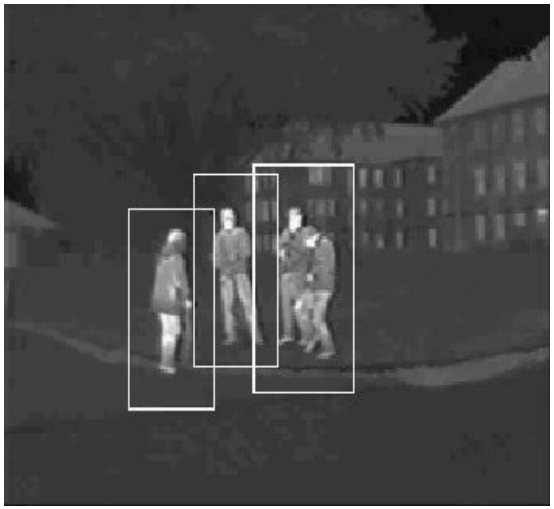






[Breckon / Eichner / Barnes / Han / Gaszczak 08-09]





http://www.durham.ac.uk/toby.breckon/demos/modgrandchallenge/

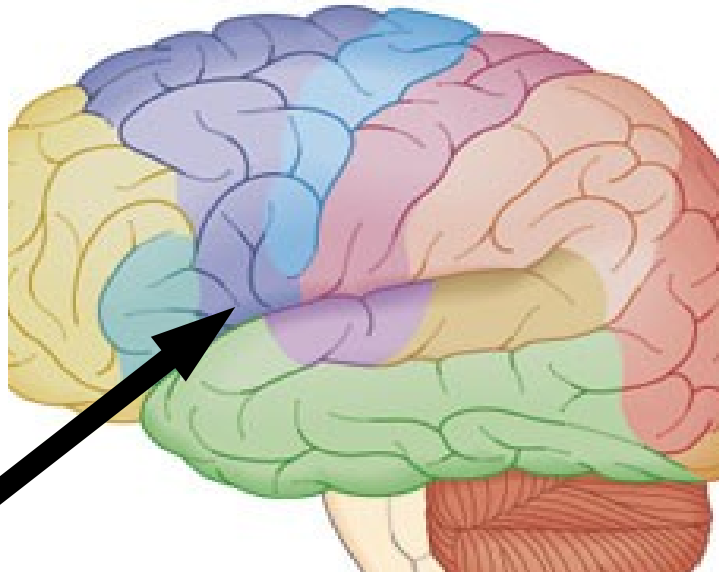
[Breckon / Eichner / Barnes / Han / Gaszczak 08-09]

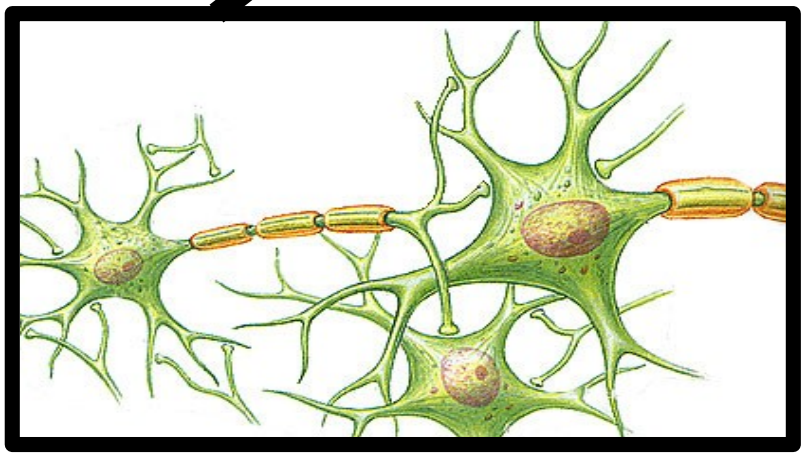
The big ones - “neural *inspired* approaches”

Biological *Motivation*

- **Real Neural Networks**
  - **human brain** as a collection of **biological neurons and synapses**

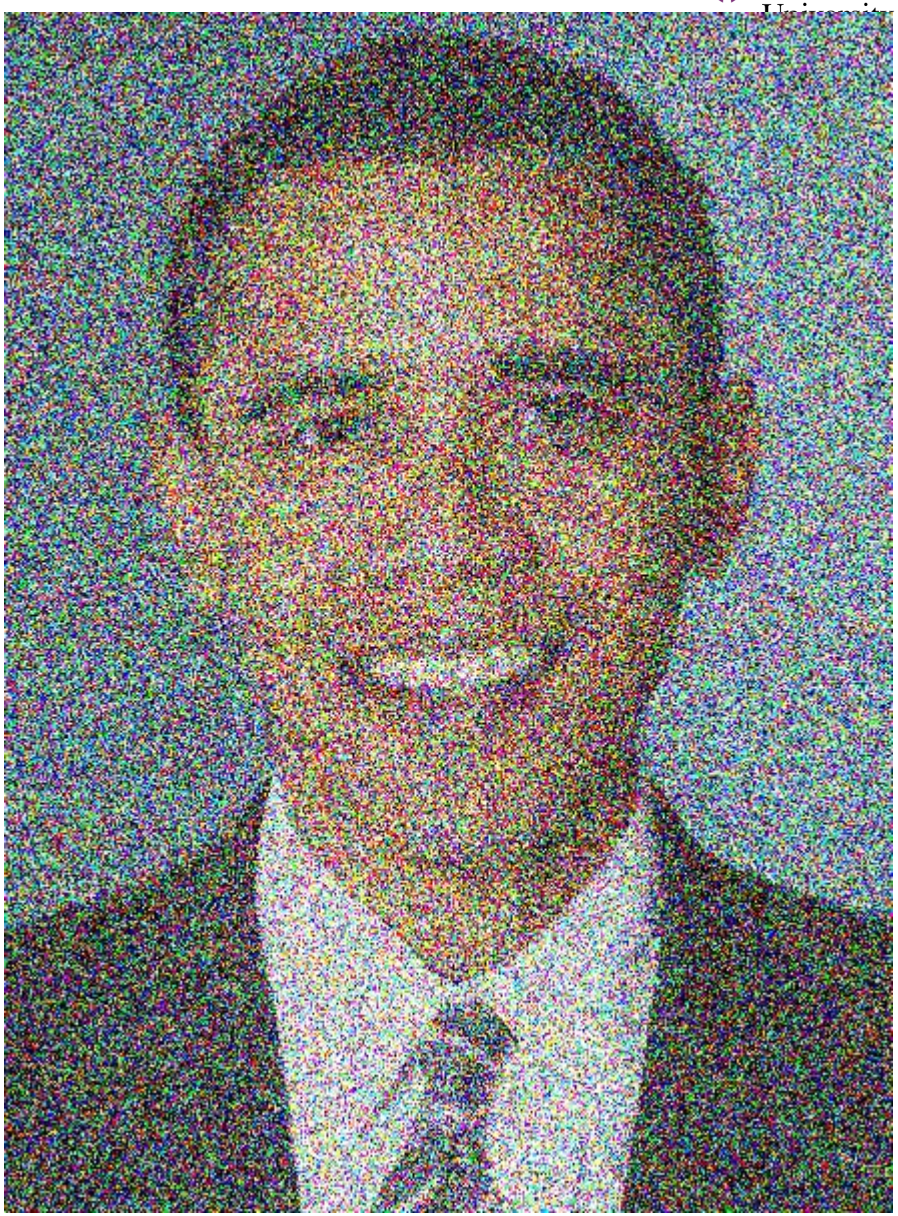
    (10^{11} neurons, 10^{4} synapse connections per neuron)
  - powerful, adaptive and noise *resilient* **pattern recognition**
  - combination of:
    - Memory
      - Memorisation
    - Generalisation
      - Learning “rules”
      - Learning “patterns”





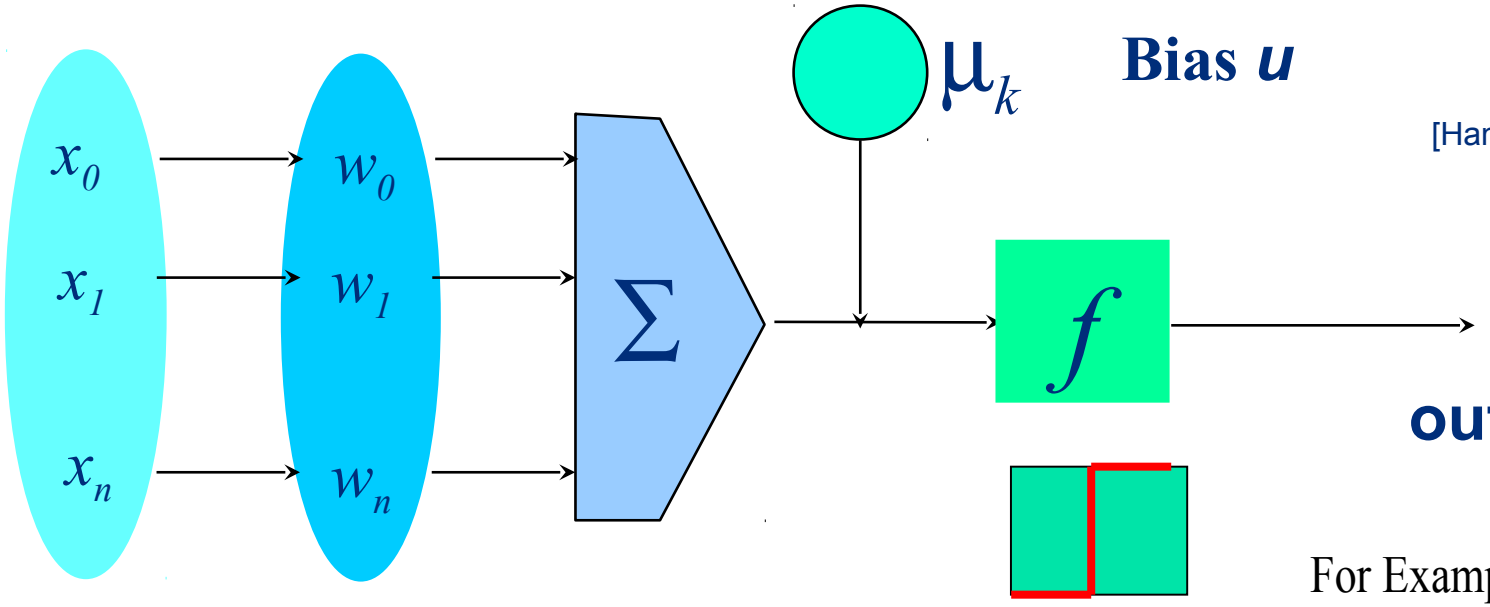
Images: DK Publishing

- Neural Networks (biological and computational) are good at **noise resilient pattern recognition**
  - the human brain can cope with **extreme noise** in pattern recognition



Images: Wikimedia Commons

Artificial Neurons (= a perceptron)



[Han / Kamber 2006]

Input vector *x* **weight vector *w*** **weighted sum** **Activation function**

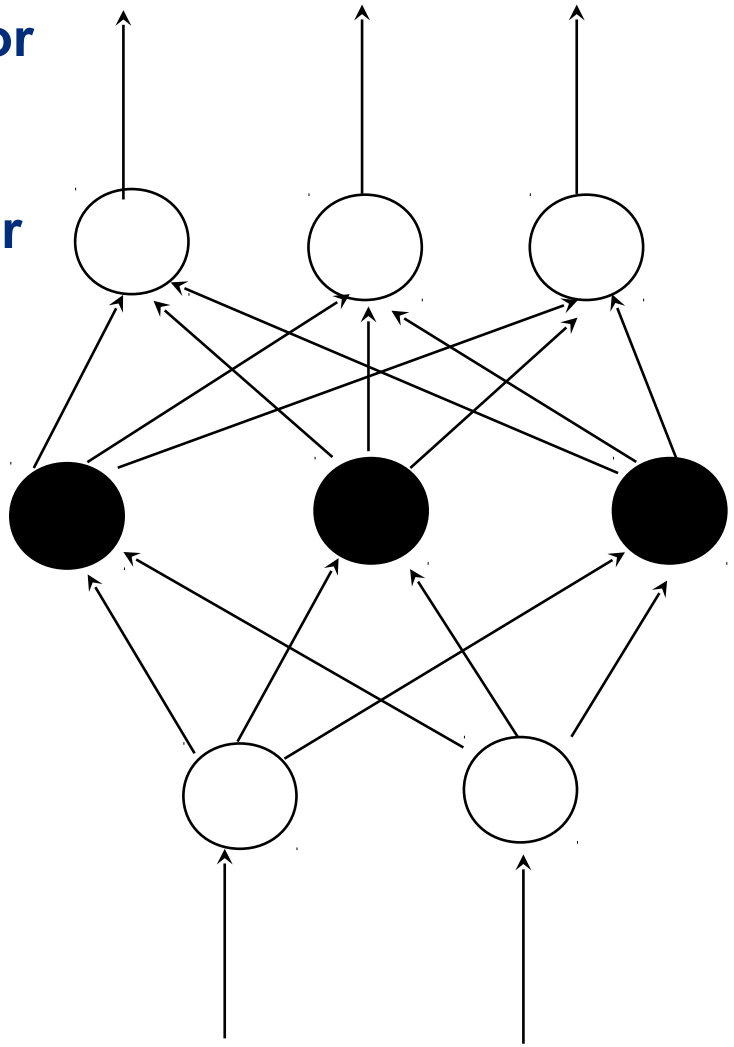
For Example let f() be:

$$o = \text{sign}\left(\sum_{i=0}^{n} w_i x_i + \mu_k\right)$$

- An *n*-dimensional **input vector *x*** is **mapped** to **output variable *o*** by means of the **scalar product and a nonlinear function mapping, *f***

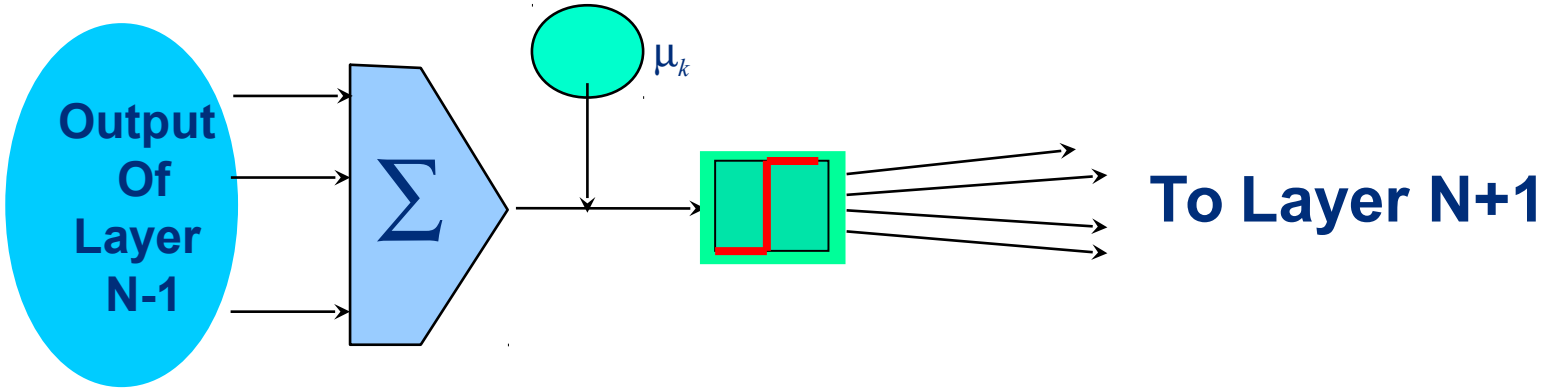
Artificial **Neural Networks** (ANN)

- **Multiple Layers of Perceptrons**
  - N.B. Neural Networks a.k.a Multi Layer Perceptons (MLP)
  - **N Layers of M Perceptrons**
    - Each fully connected (in graph sense to the next)
  - Every node of layer N takes outputs of all M nodes of layer N-1



[Han / Kamber 2006]

In **every node** we have ....



- **Input** to network = (numerical) **attribute vector** describing classification examples

- **Output** of network = vector representing classification
  - e.g. {1,0}, {0,1}, {1,1}, {0,0} for classes A,B,C,D
  - or alt. {1,0,0}, {0,1,0}, {0,0,1} for classes A, B C

Essentially, **input to output** is mapped as a **weighted sum** occurring at **multiple** (fully connected) **layers** in the network ....

… so the **weights** are key.

If everything else is constant

...

(i.e. activation function, network topology)

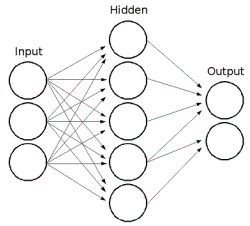
… the weights are only thing that changes

Thus …

setting the weights = training the network

Backpropagation Summary

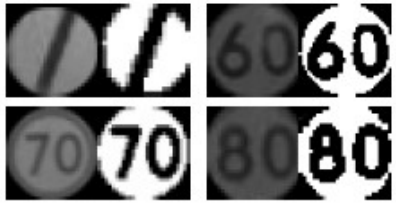
- **Modifications** are made in the **"backwards" direction**: *from the output layer, through each hidden layer down to the first hidden layer*, hence **"Backpropagation"**



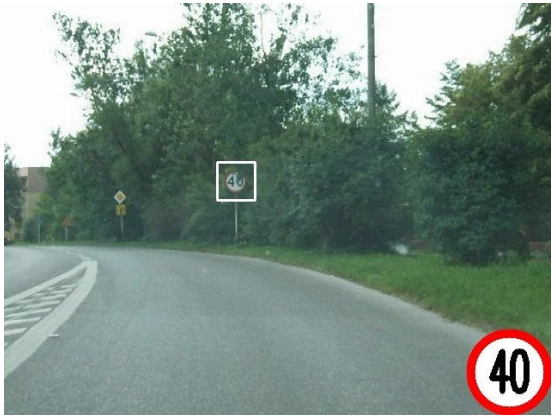
- **Key Algorithmic Steps**
  - **Initialize** weights (to small random values) in the network
  - **Propagate the inputs forward** (by applying activation function) at each node
  - **Backpropagate the error backwards** (by updating weights and biases)
  - **Terminating** condition (when error is very small or enough iterations)

Backpropogation details beyond scope/time (see Mitchell '97).

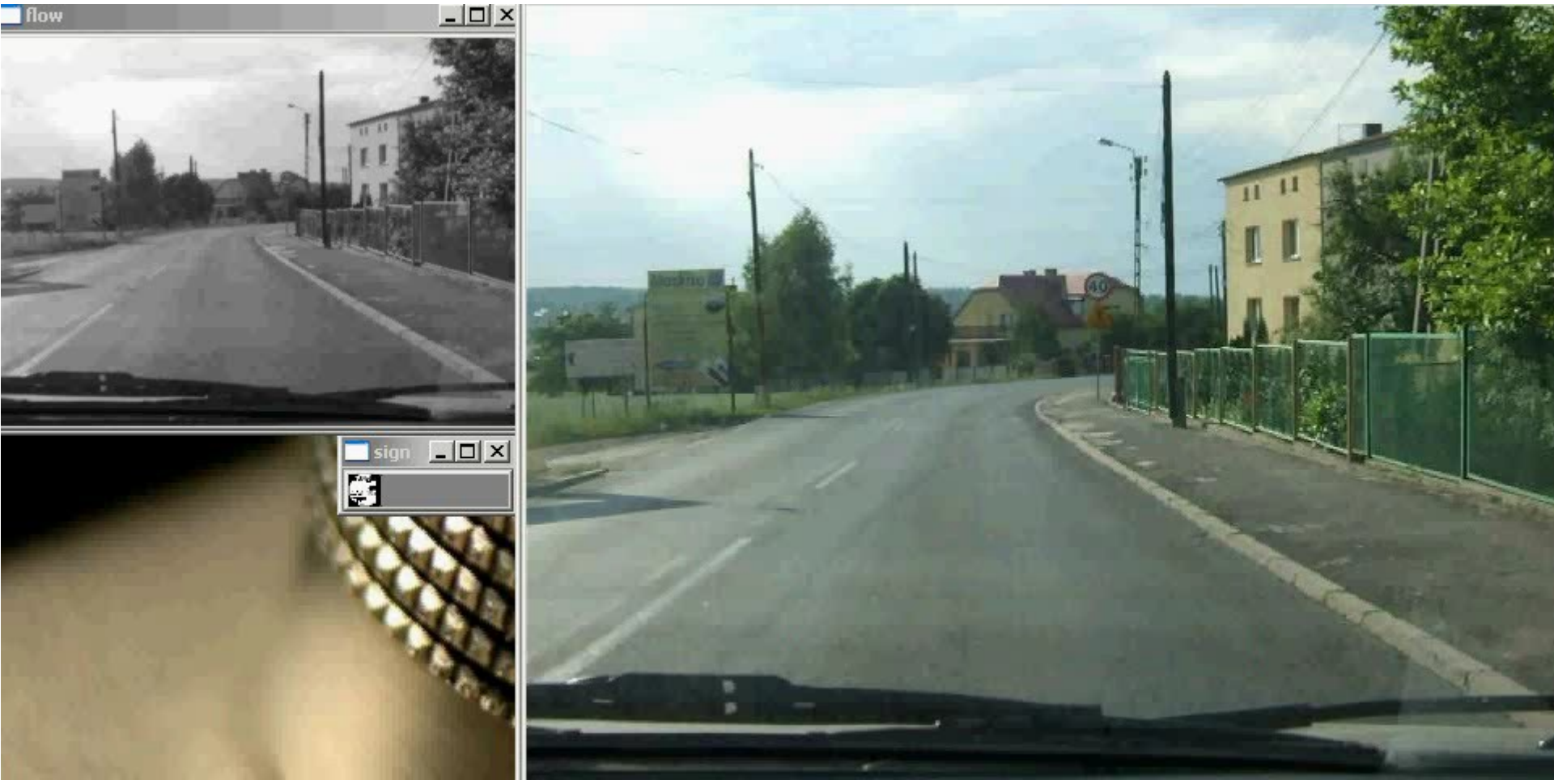
Example: **speed sign recognition**



- **Input:**
  - Extracted binary text image
  - Scaled to 20x20 pixels
- **Network:**
  - 30 hidden nodes
  - 2 layers
  - backpropogation
- **Output:**
  - 12 classes
  - {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, national-speed limit,non-sign}
- **Results:** ~97% (success)



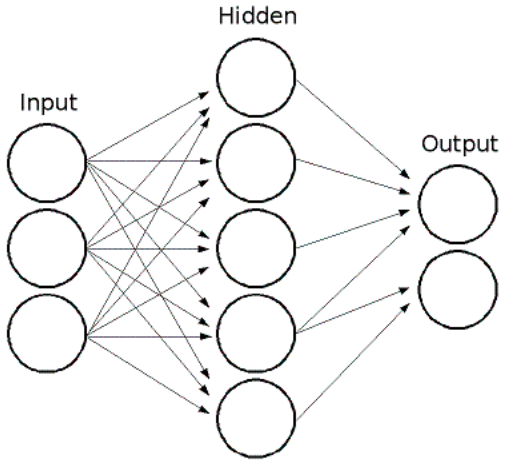
[Eichner / Breckon '08]



http://www.durham.ac.uk/toby.breckon/demos/speedsigns/

Problems suited to ANNs

- **Input** is **high-dimensional discrete or real-valued**
  - e.g. raw sensor input – signal samples or image pixels



- **Output**
  - discrete or real valued
  - is a vector of one or more values
- Possibly **noisy data**
- Form of **target function is generally unknown**
  - *i.e. don't know input to output relationship*
- Human readability of result is unimportant
  - *rules such as IF..THEN .. ELSE not required*

Problems **with** ANNs

- **Termination** of backpropagation
  - Too many iterations can lead to overfitting (to training data)
  - Too few iterations can fail to reduce output error sufficiently

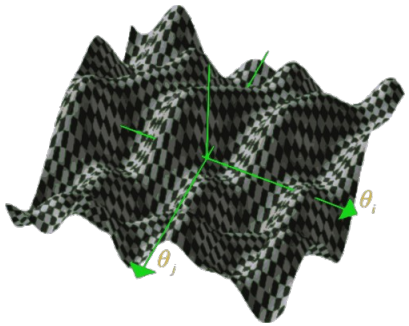
- Needs **parameter selection**
  - Learning rate (weight up-dates)
  - Network Topology (number of hidden nodes / number of layers)
  - Choice of activation function
  - ....
- **What is the network learning?**
  - How **can we be sure** the correct (classification) function is being learned ?

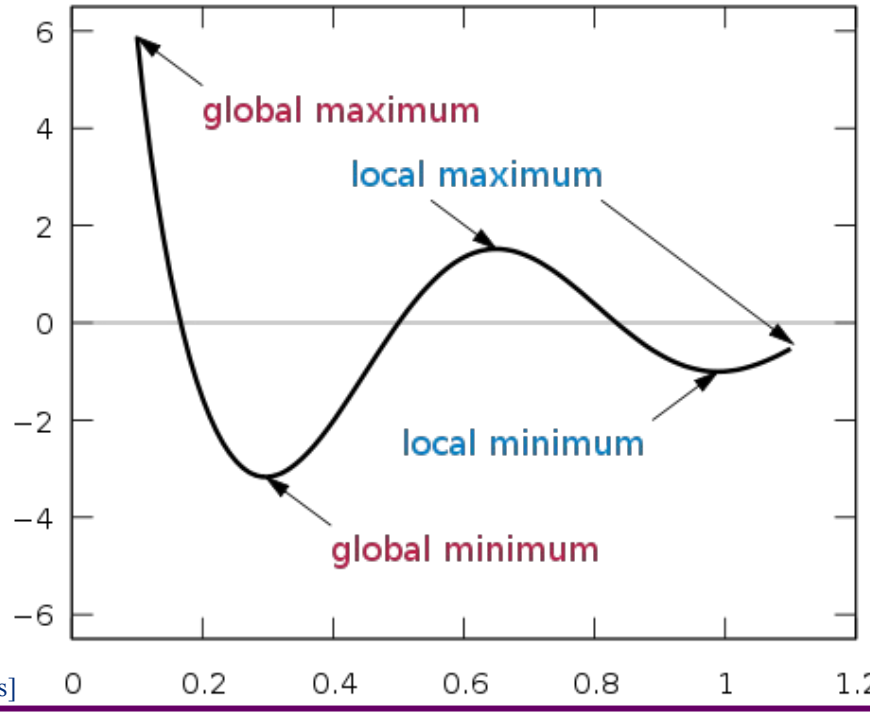
c.f. AI folk-lore “the tanks story”

Problems **with** ANNs

- May find **local minimum** within the search space of all possible weights *(due to nature of backprop. gradient decent)*
  - i.e. **backpropagation** is **not guaranteed to find the best weights** to learn the classification/regression function
  - Thus "learned" **neural network may not find optimal solution to the classification/regression problem**

Weight space $\{W_{ij}\}$





Images [Wikimedia Commons]

… towards the future state of the art

- **Deep Learning (Deep Neural Networks)**
  - multi-layer neural networks, varying layer sizes
    - varying levels of abstraction / intermediate feature representations
  - trained one layer at a time, followed by backprop.
  - complex and computationally demanding to train

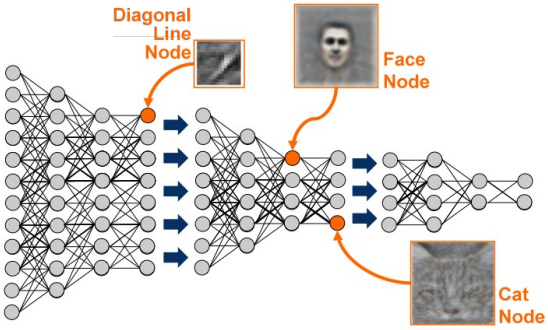
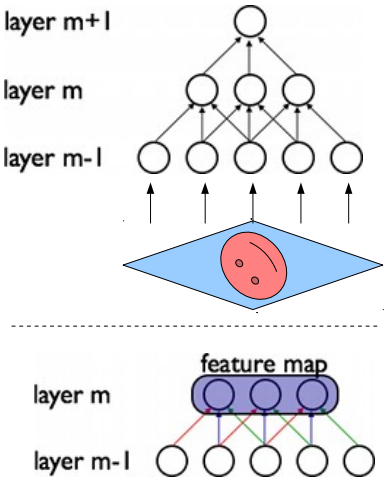


Image: http://theanalyticsstore.com/deep-learning/

- **Convolutional Neural Networks**
  - leverage **local spatial layout of features in input**
  - **locally adjacent neurons connected** layer to layer
    - instead of full layer to layer connectivity
    - units in ***m-th* layer connected to local subset of units in *(m-1)-th* layer** (which are spatially adjacent)



- **Often combined together : "state of the art" results in Large Scale Visual Recognition Challenge**
  - Image-Net Challenge - http://image-net.org/challenges/LSVRC/2013/ (1000 classes of object)

Deep Learning Neural Networks

Results are impressive.

But the same problems remain.

(overfitting et al.)

The big ones - “kernel *driven* approaches”

Support Vector Machines

- Basic Approach:
  - **project** instances **into high dimensional space**
  - **learn linear separators** (in high dim. space) with **maximum margin**
  - learning as **optimizing bound on expected error**

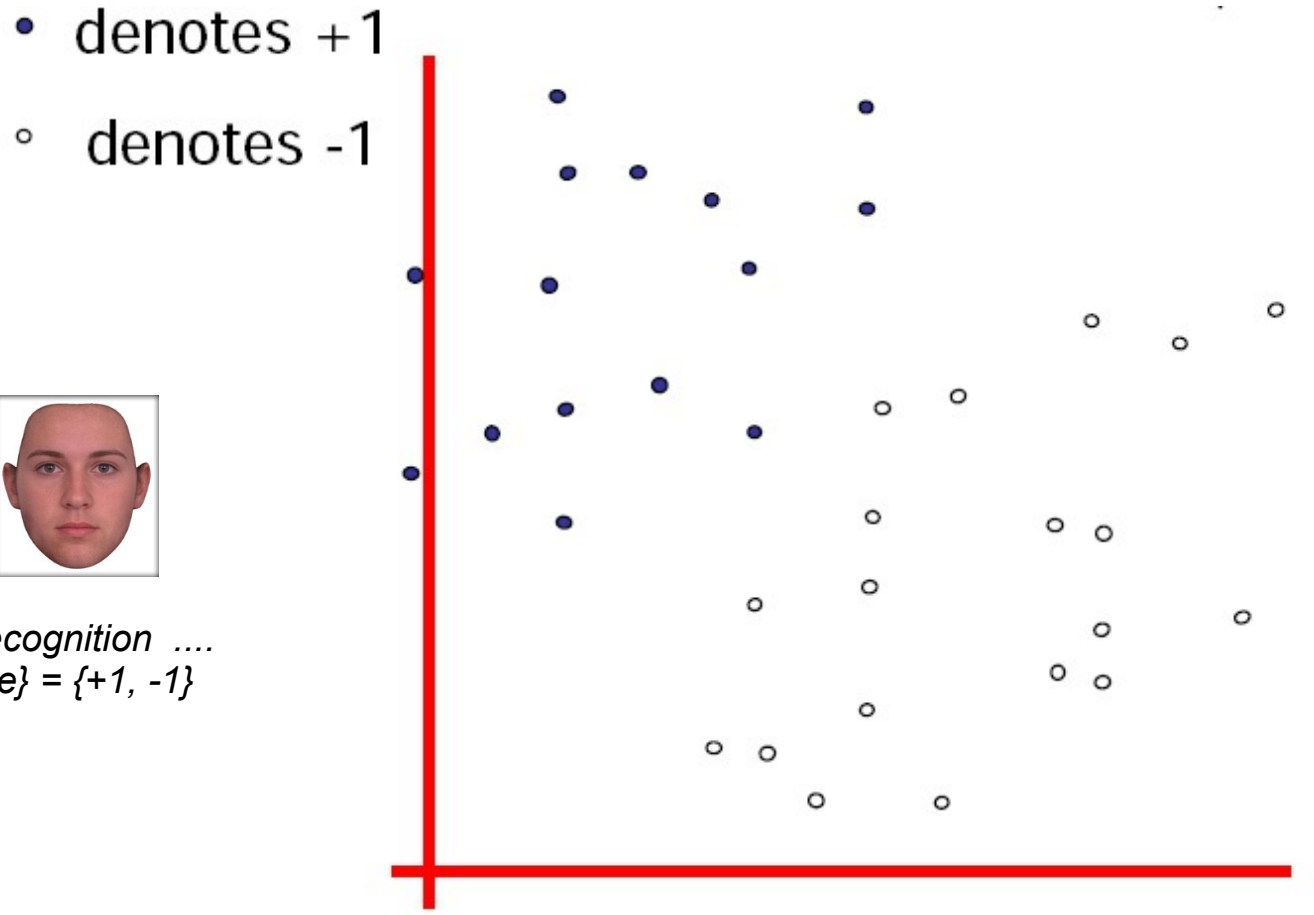
- Positives
  - **good performance** on character recognition, text classification, ...
  - "appears" to **avoid overfitting** in high dimensional space
  - **global optimisation**, thus avoids local minima

- Negatives
  - applying trained classifier *can be expensive* (i.e. query time)

N.B. "Machines" is just a **sexy name** (probably **to make them sound different**), *they are really **just computer algorithms** like everything else in machine learning!*

... so don't get confused by the whole "machines" thing :o)

Simple Example



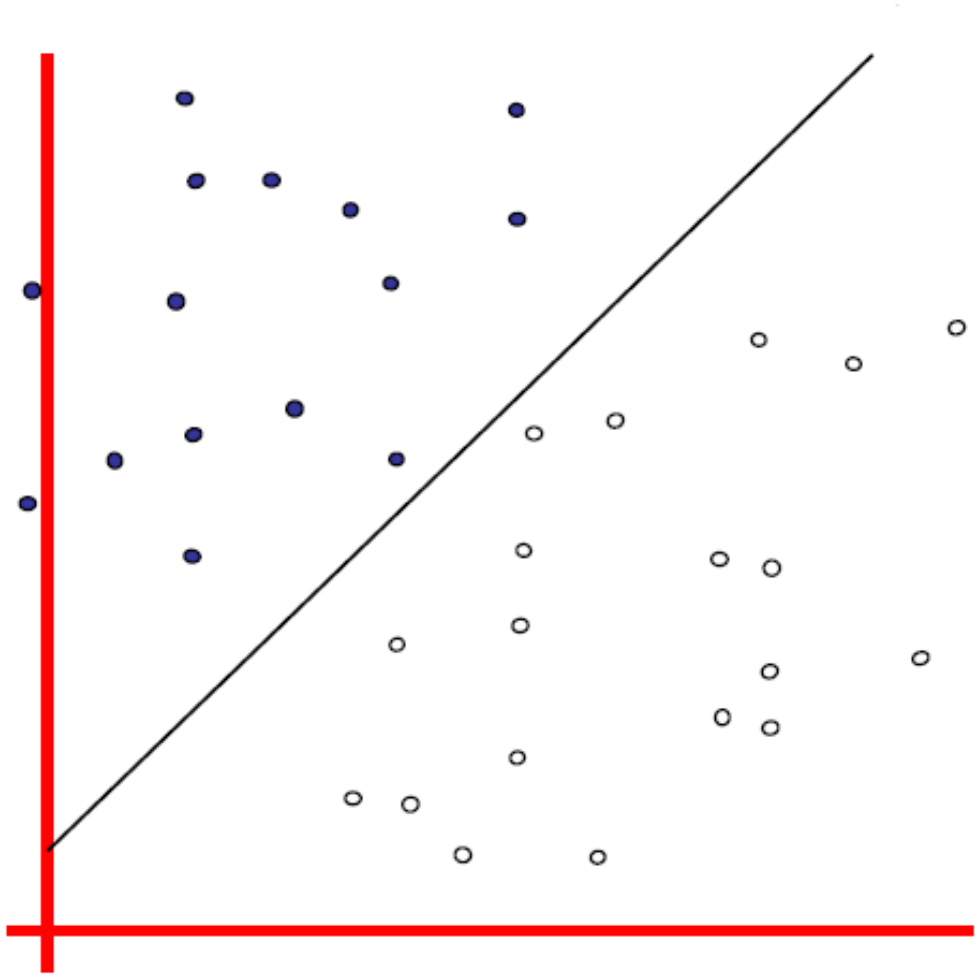
e.g. gender recognition ....
{male, female} = {+1, -1}

- How can we separate (i.e. classify) these data examples ? (i.e. learn +ve / -ve)

Simple Example

- denotes +1
- denotes -1

e.g. gender recognition ....
{male, female} = {+1, -1}

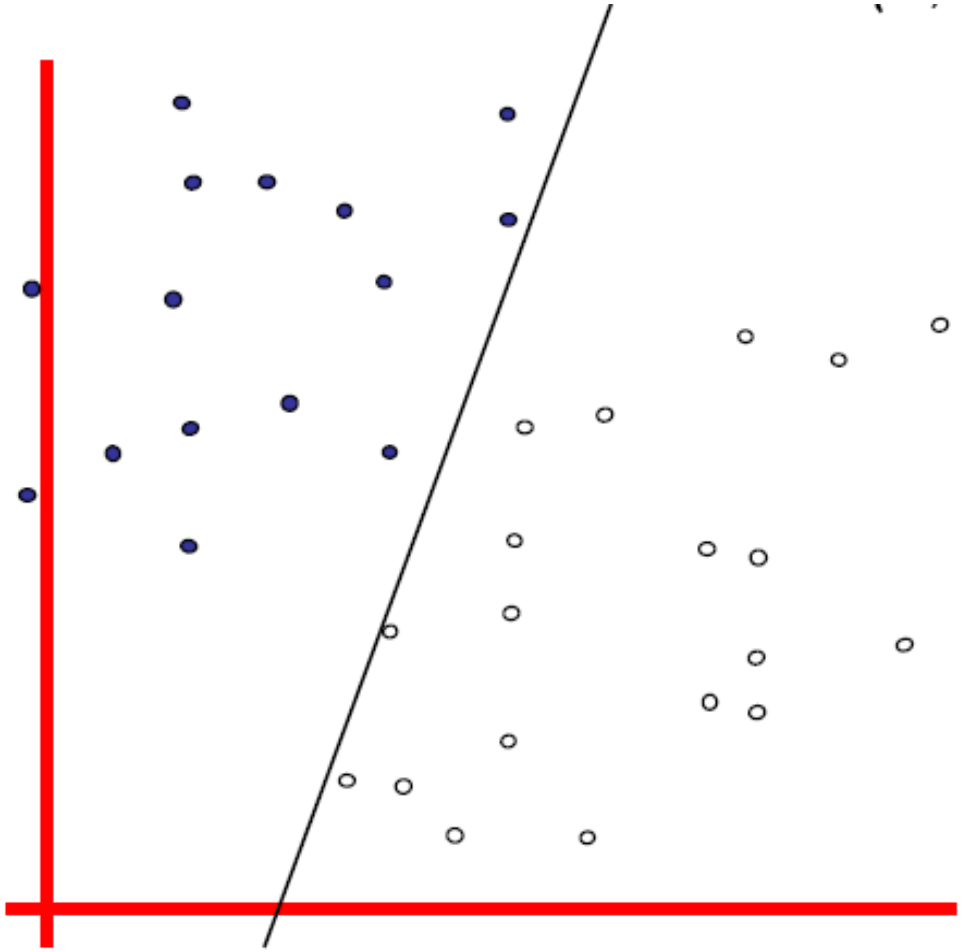


- Linear separation

Simple Example

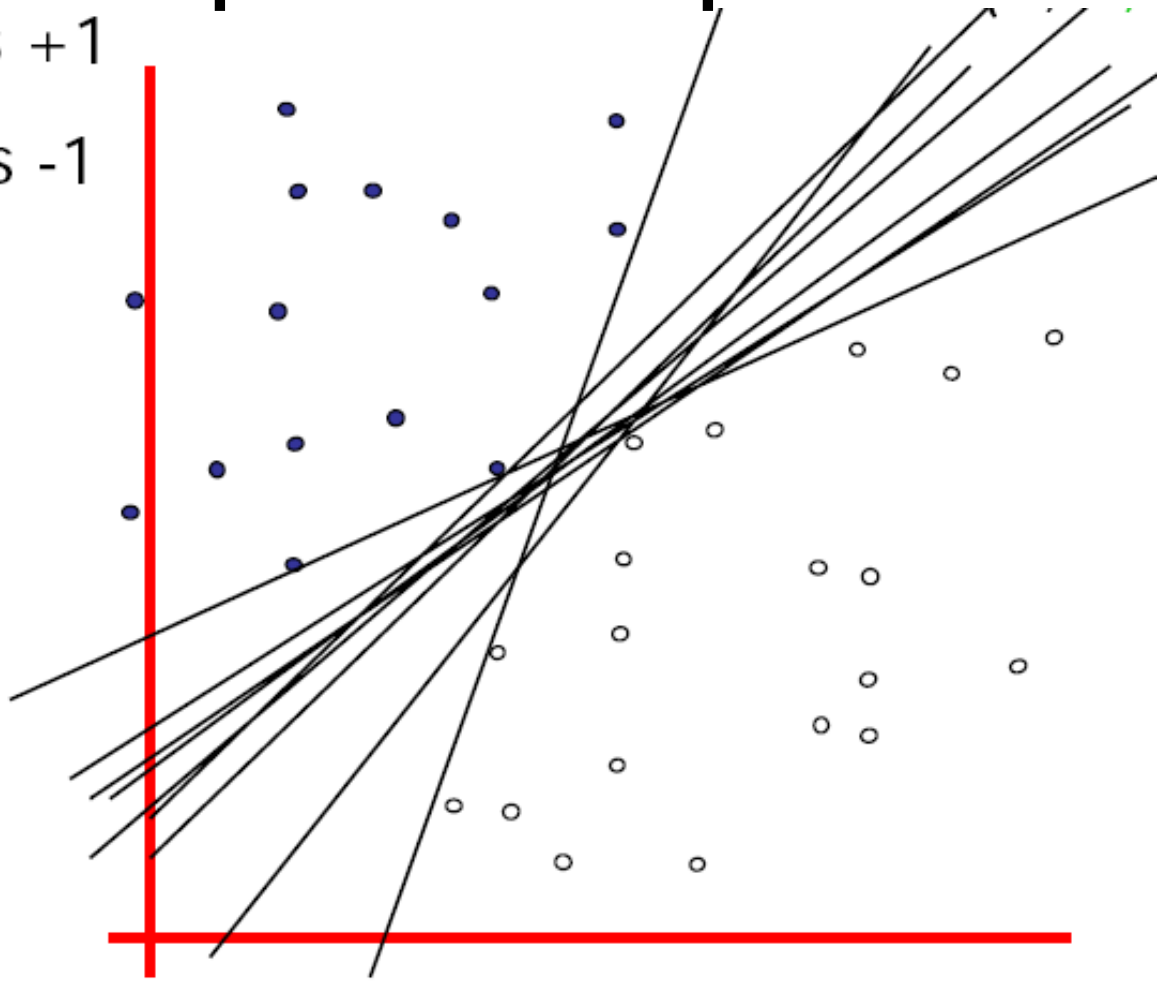
- denotes +1
- denotes -1

e.g. gender recognition ....
{male, female} = {+1, -1}



- Linear separators – *which one ?*

Simple Example

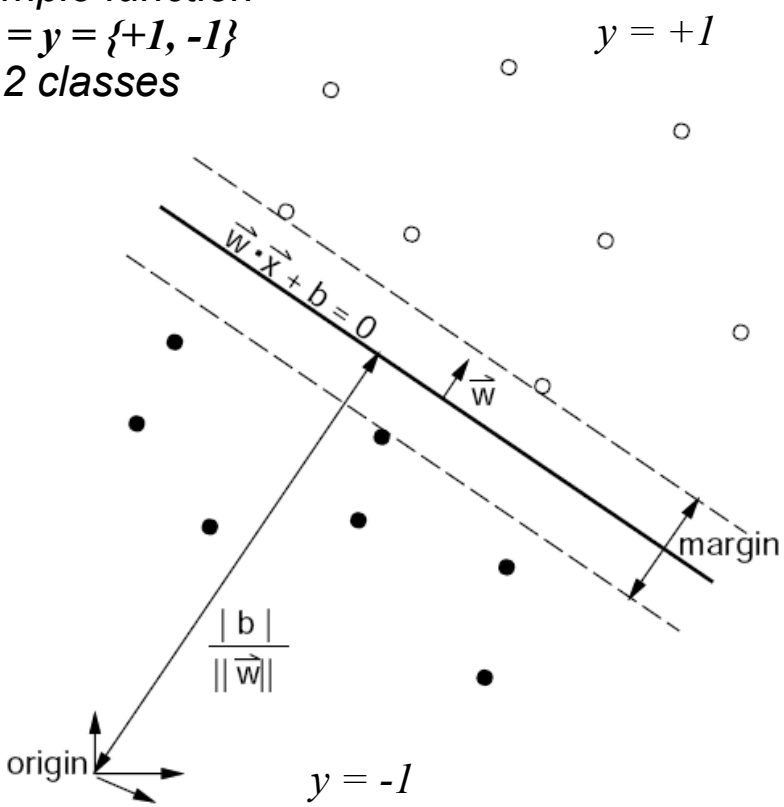


e.g. gender recognition ....
{male, female} = {+1, -1}

- Linear separators – *which one ?*

Linear Separator

Classification of example function ***f(x) = y = {+1, -1}*** *i.e. 2 classes*



- Instances (i.e, examples) $\{\mathbf{x_i}, \mathbf{y_i}\}$
  - $\mathbf{x_i}$ = point in instance space (R^n) made up of n attributes
  - $\mathbf{y_i}$ =class value for classification of $\mathbf{x_i}$

- Want a linear separator. Can view this as **constraint satisfaction problem:**

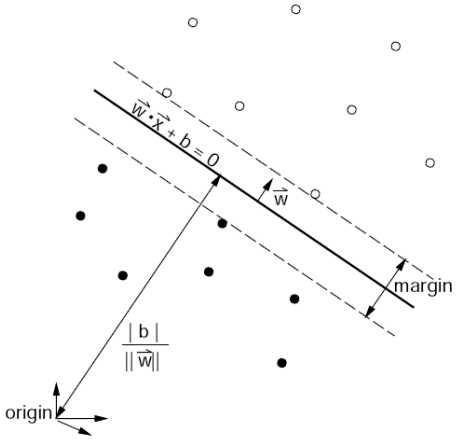
$$\vec{x_i} \cdot \vec{w} + b \geq +1 \quad \text{if } y_i \equiv f(\vec{x_i}) = +1$$
$$\vec{x_i} \cdot \vec{w} + b \leq -1 \quad \text{if } y_i = -1$$

- Equivalently,

$$y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0, (\forall i)$$

N.B. we have a vector of weights coefficients $\vec{w}$

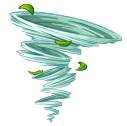
Linear Separator



"hyperplane" = separator boundary

hyperplane in R^2 == 2D line

- Now find the hyperplane (separator) with **maximum margin**
  - thus if size of margin is defined as $\frac{2}{||\vec{w}||}$ *(see extras)*



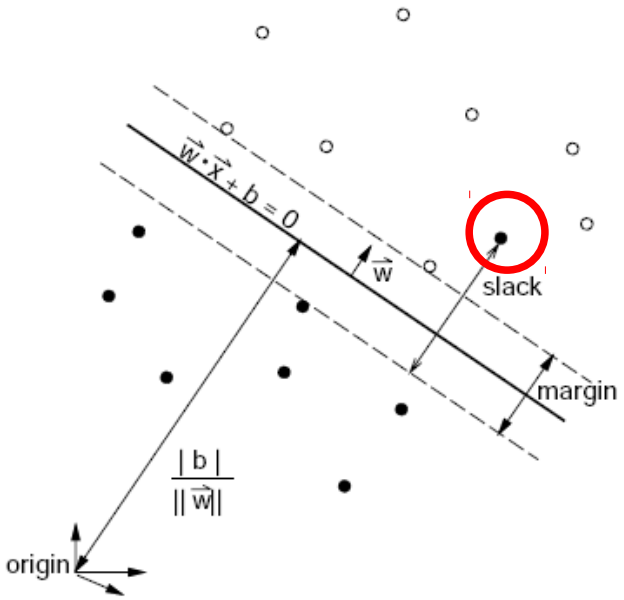
- So view our problem as a constrained optimization problem:

$$\text{Minimize } ||\vec{w}||^2, \text{ subject to}$$

$$y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0, (\forall i)$$

  - **Find "hyperplane" using computational optimization approach** (beyond scope)

What about Non-separable Training Sets ?



Add a "slack variable" $\xi_i \geq 0$ for each $\langle x_i, y_i \rangle$.
New optimization problem is

Minimize $||\vec{w}||^2 + C(\Sigma_i \xi_i)^k$, subject to

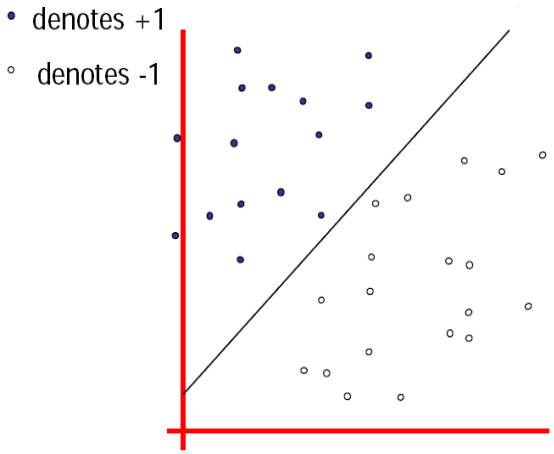
$$\vec{x_i} \cdot \vec{w} + b \geq +1 - \xi \quad \text{if } y_i = +1$$
$$\vec{x_i} \cdot \vec{w} + b \leq -1 + \xi \quad \text{if } y_i = -1$$

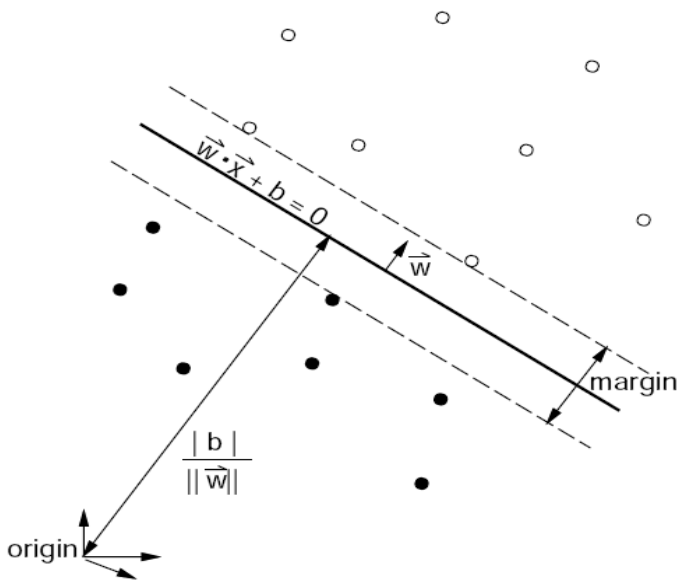
C picked by hand

Penalty term > 0
for each "wrong side of the boundary" case

– **Find "hyperplane" via computational optimization**

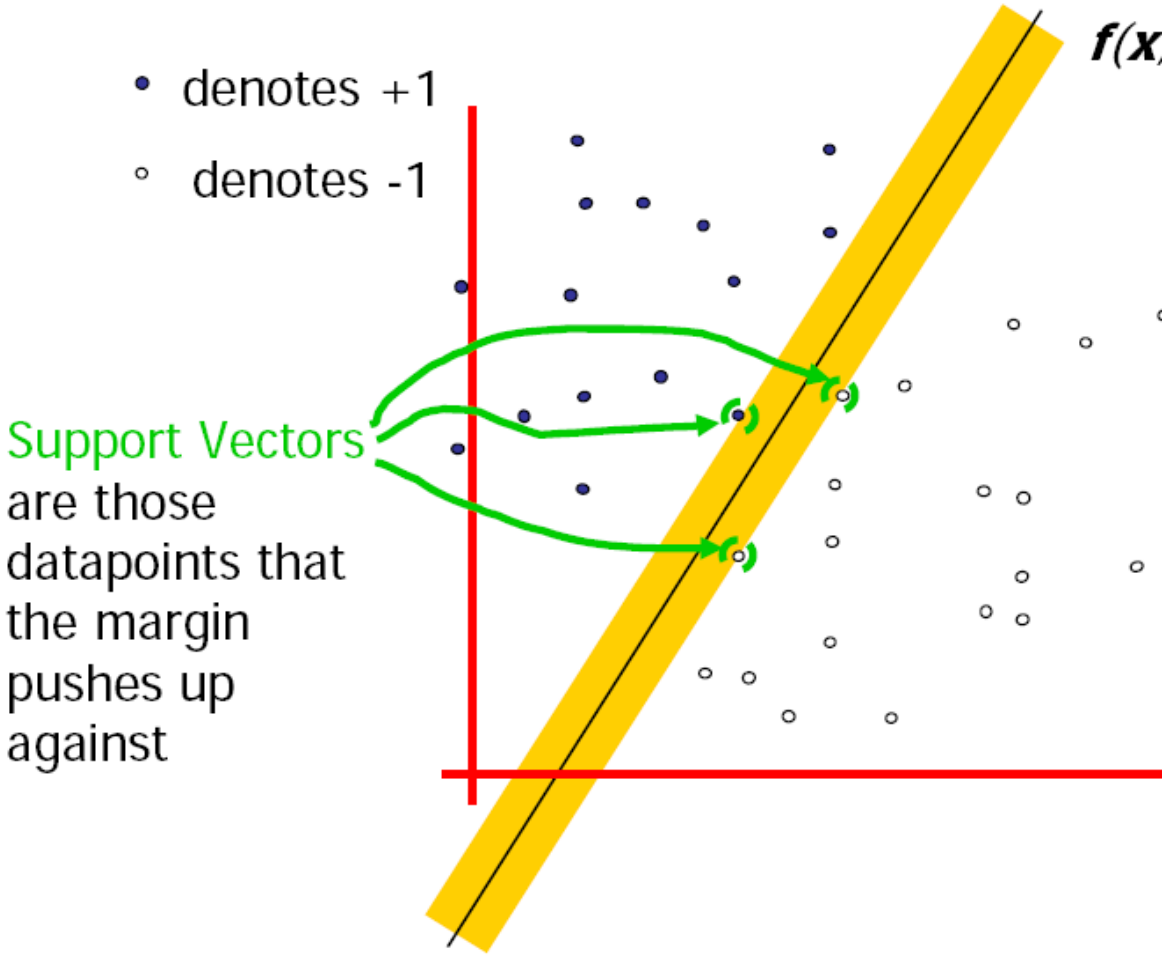
Linear Separator





- separator margin determined by just a few examples
  - call these **support vectors**
  - can define separator in terms of support vectors and classifier examples *x* as:

$$f(\vec{x}) \leftarrow sgn(\sum_{s_i \in \text{support vectors}} w_i \vec{s_i} \cdot \vec{x} + b)$$

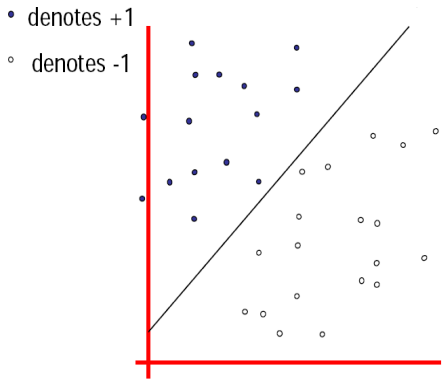


- **Support vectors** = *sub-set of training instances that define decision boundary between classes*

This is the simplest kind of Linear SVM (LSVM)

How do we **classify a new example ?**

- New unseen (test) example attribute vector *x*
- *Set of support vectors* $\{s_i\}$ *with weights* $\{w_i\}$, *bias* b
  - *define the "hyperplane" boundary in the original dimension (this is a linear case)*

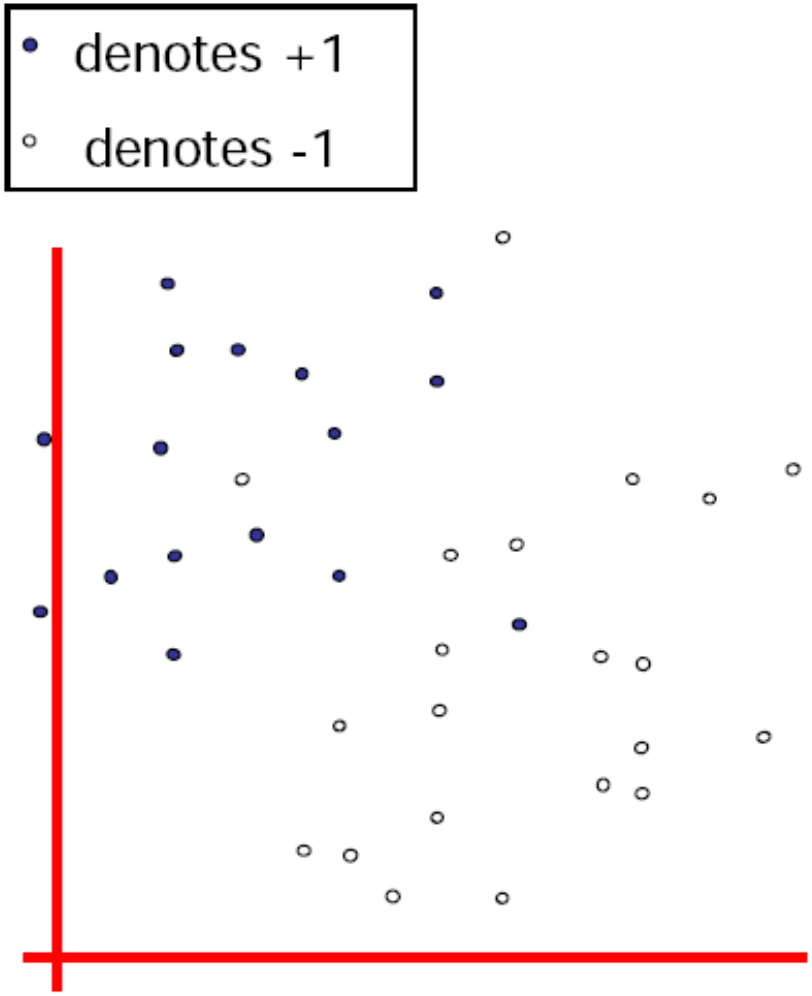


- *The output of the classification f(x) is:*

$$f(\vec{x}) \leftarrow sgn\Big(\sum_{s_i \in \text{support vectors}} w_i \vec{s_i} \cdot \vec{x} + b\Big)$$

  - *i.e. f(x) = {-1, +1}*

What about this ..... ?



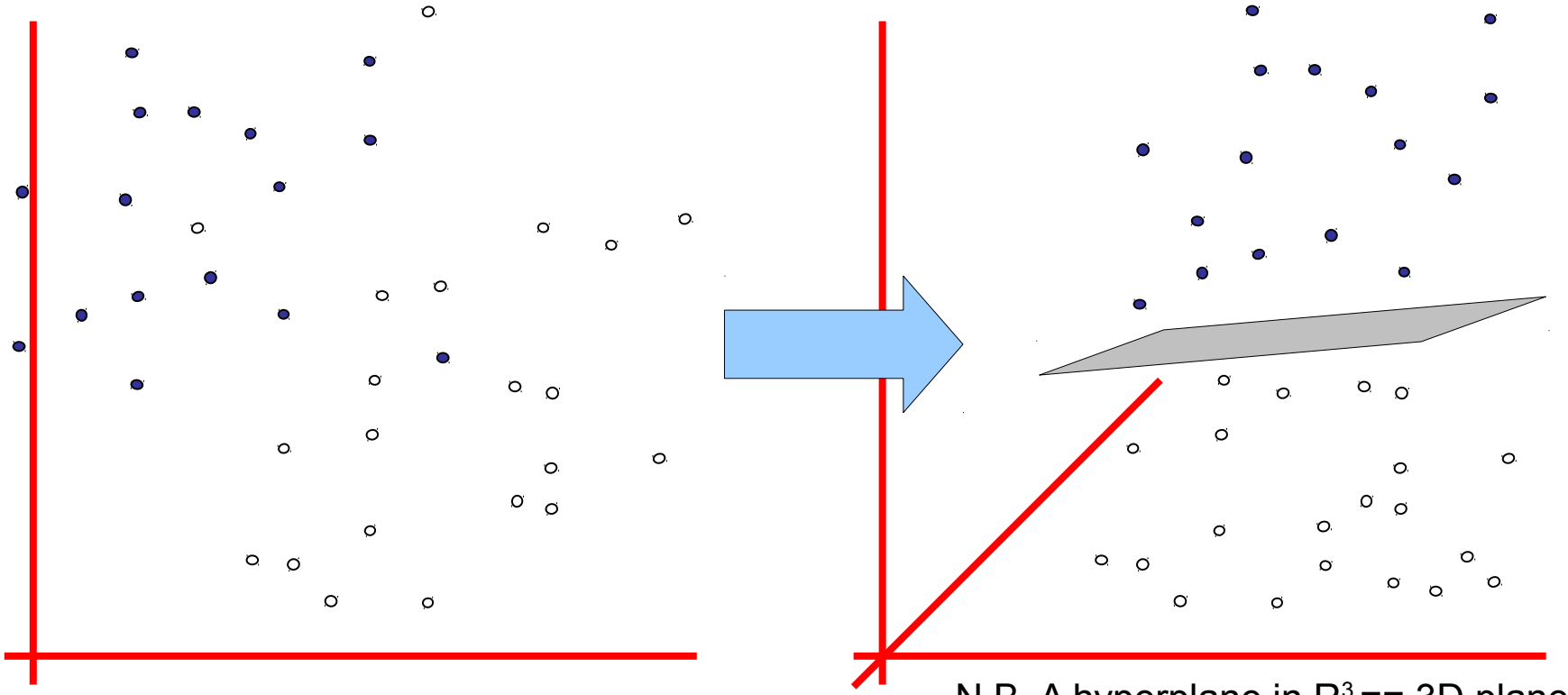
Not linearly Separable !

e.g. *2D to 3D*

denotes +1

denotes -1

- Projection from 2D to 3D allows separation by hyperplane (surface) in R^3



N.B. A hyperplane in R^3 == 3D plane

SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

Non Linear SVMs

- Just as before but now **with the kernel ....**

$$f(\vec{x}) \leftarrow \sum_{s_i \in \text{support vectors}} w_i y_i K(\vec{s_i}, \vec{x}) + b$$

  - For the (earlier) linear case **K** is the identity function (or matrix)
    - This is often referred to the “linear kernel”

Non Linear SVMs

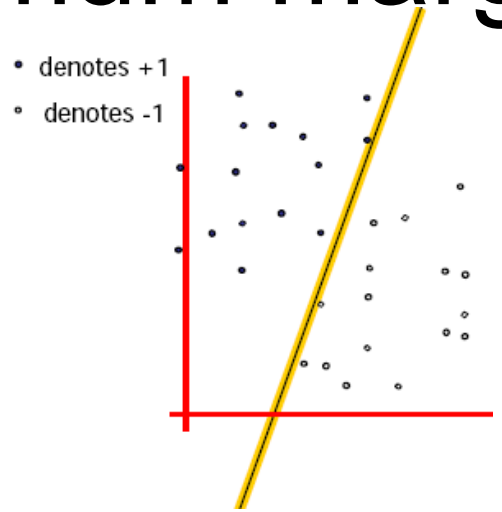
- Suppose we have instance space $X = R^n$, need non-linear separator.
  - **project** X **into some higher dimensional space** $X' = R^m$ where data will be linearly separable
  - let $\Phi : X \rightarrow X'$ be this projection.
- Interestingly,
  - Training depends only on dot products of form $\Phi(x_i) \bullet \Phi(x_j)$
    - i.e. dot product of instances in R^n gives divisor in R^n
  - So we can train in R^m with same computational complexity as in R^n, provided we can find a **kernel basis function** ***K*** such that:
    - $K(x_i, x_j) = \Phi(x_i) \bullet \Phi(x_j)$ ***(kernel trick)***
  - Classifying new instance x now requires calculating sign of:

$$f(\vec{x}) \leftarrow \sum_{s_i \in \text{support vectors}} w_i y_i K(\vec{s_i}, \vec{x}) + b$$

This is how SVMs solve difficult problems without linear separation boundaries (hyperplanes) occurring in their original dimension.

"project data to higher dimension where data is separable"

Why use maximum margin?



- Intuitively this **feels safest.**
- **a small error in the location of the boundary** gives us **least chance of causing a misclassification.**
- Model is **immune to removal of any non support-vector data-points**
- **Some related theory** (using V-C dimension) (beyond scope)

- **Distance from boundary ~= measure of "good-ness"**
- **Empirically it works very well**

*Choosing **Kernels** ?*

- **Kernel functions** *(commonly used):*
  - Polynomial function of degree p

$$K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^p$$

  - Gaussian radial basis function (size σ)

$$K(\vec{x_i}, \vec{x_j}) = e^{-||\vec{x_i} \cdot \vec{x_j}||^2 / 2\sigma^2}$$

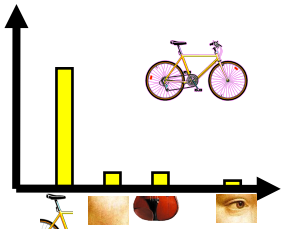
  - 2D sigmoid function (as per neural networks)

Commonly chosen by grid search of parameter space
"glorified trial and error"

Application to *Image Classification*

- Common Model - **Bag of Visual Words**
  1. build histograms of feature occurrence over training data (features: SIFT, SURF, MSER ....)
  2. Use histograms as input to SVM *(or other ML approach)*

- **Cluster Features** in R^n space



- Cluster *"membership"* creates a **histogram of feature occurrence**

Caltech objects database 101 object classes

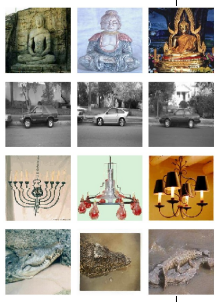
Features:

SIFT detector / PCA-SIFT descriptor, *d*=10

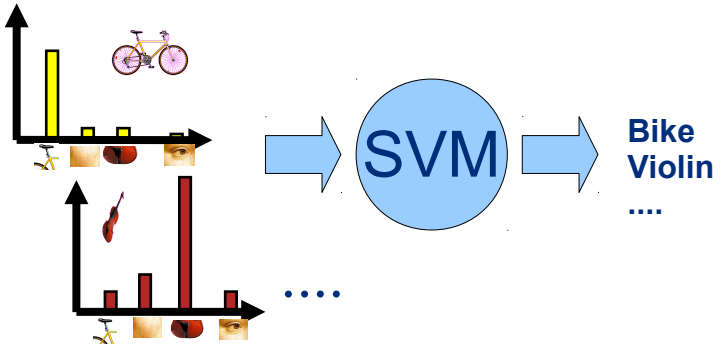
30 training images / class

43% recognition rate

(1% chance performance)



Example: Kristen Grauman



- **Bag of Words Model**
  - SURF features
  - SVM {people | vehicle} detection
  - Decision Forest - sub-categories

[Breckon / Han / Richardson, 2012]

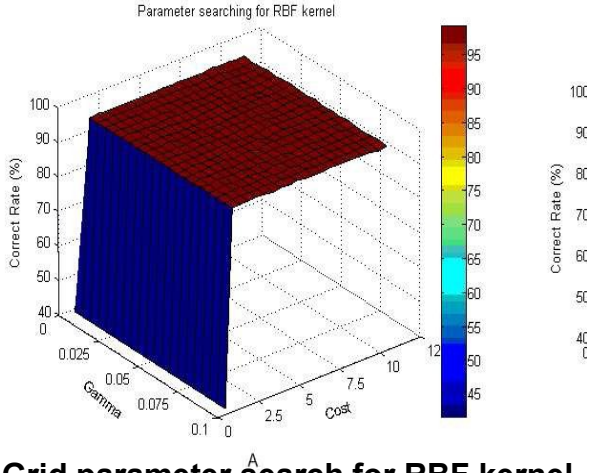
www.durham.ac.uk/toby.breckon/demos/multimodal/

Application to *Image Classification*

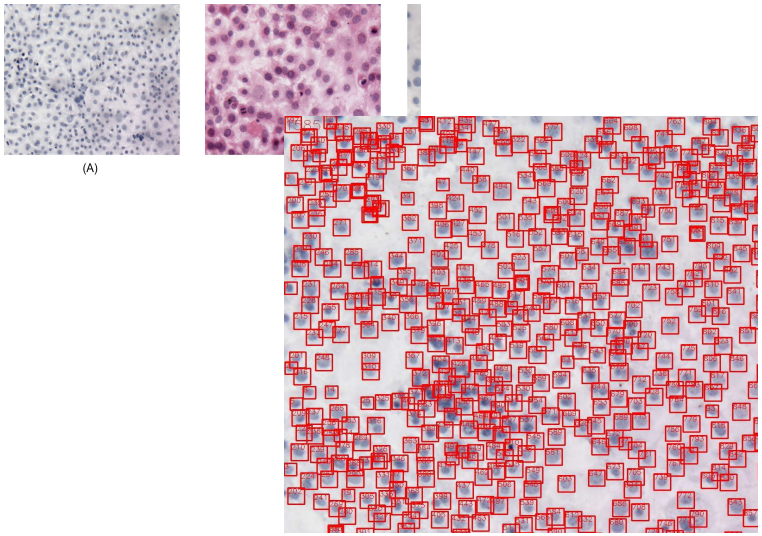
- Searching for Cell Nuclei Locations with SVM

  [Han / Breckon et al. 2010]

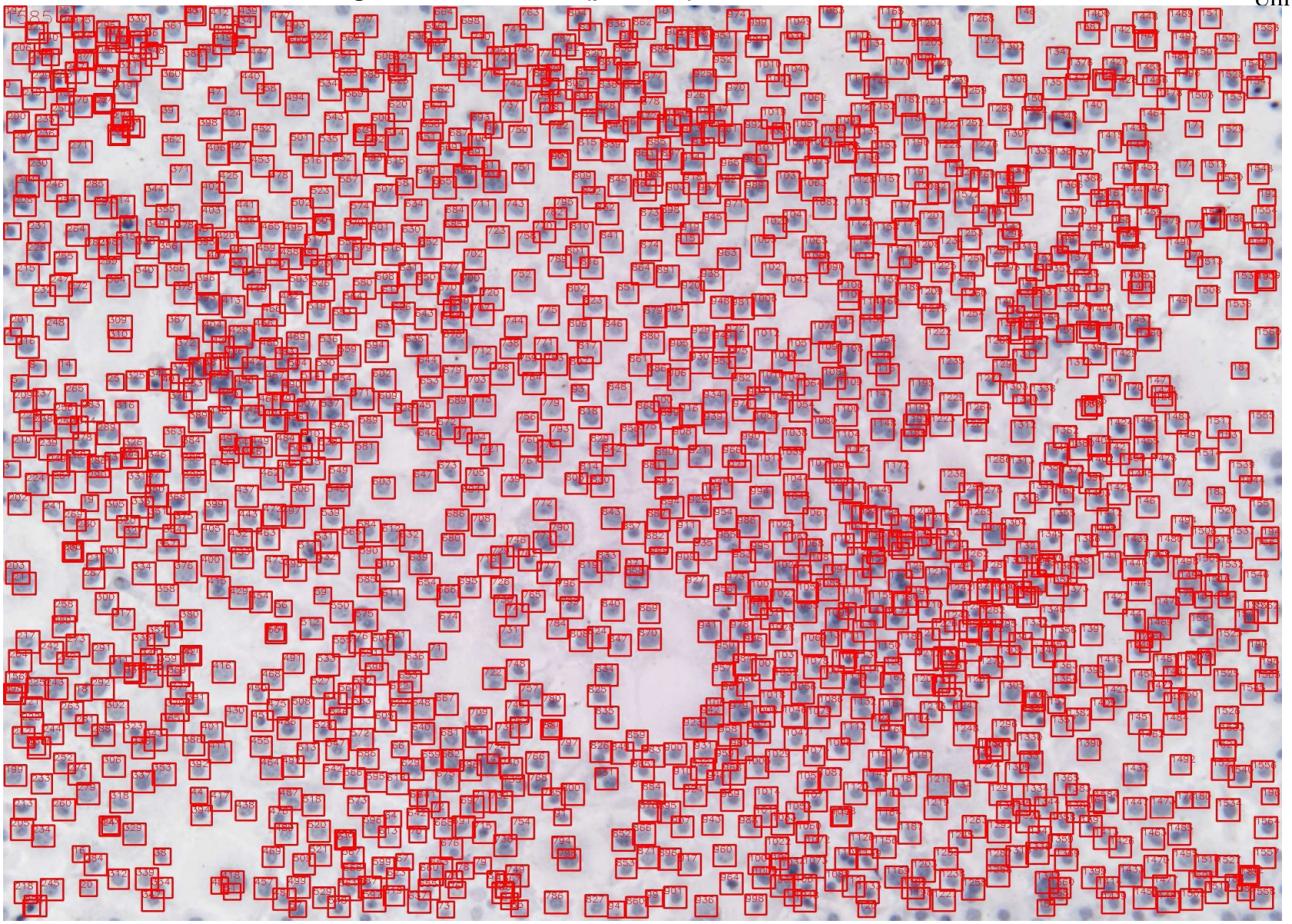
  - **input:** “laplace” enhanced pixel values as vector
    - scaled to common size
  - **process:**
    - exhaustively extract each image neighbourhood over multiple scales
    - pass pixels to SVM
      - Is it a cell nuclei ?
  - **output**: {cell, nocell}

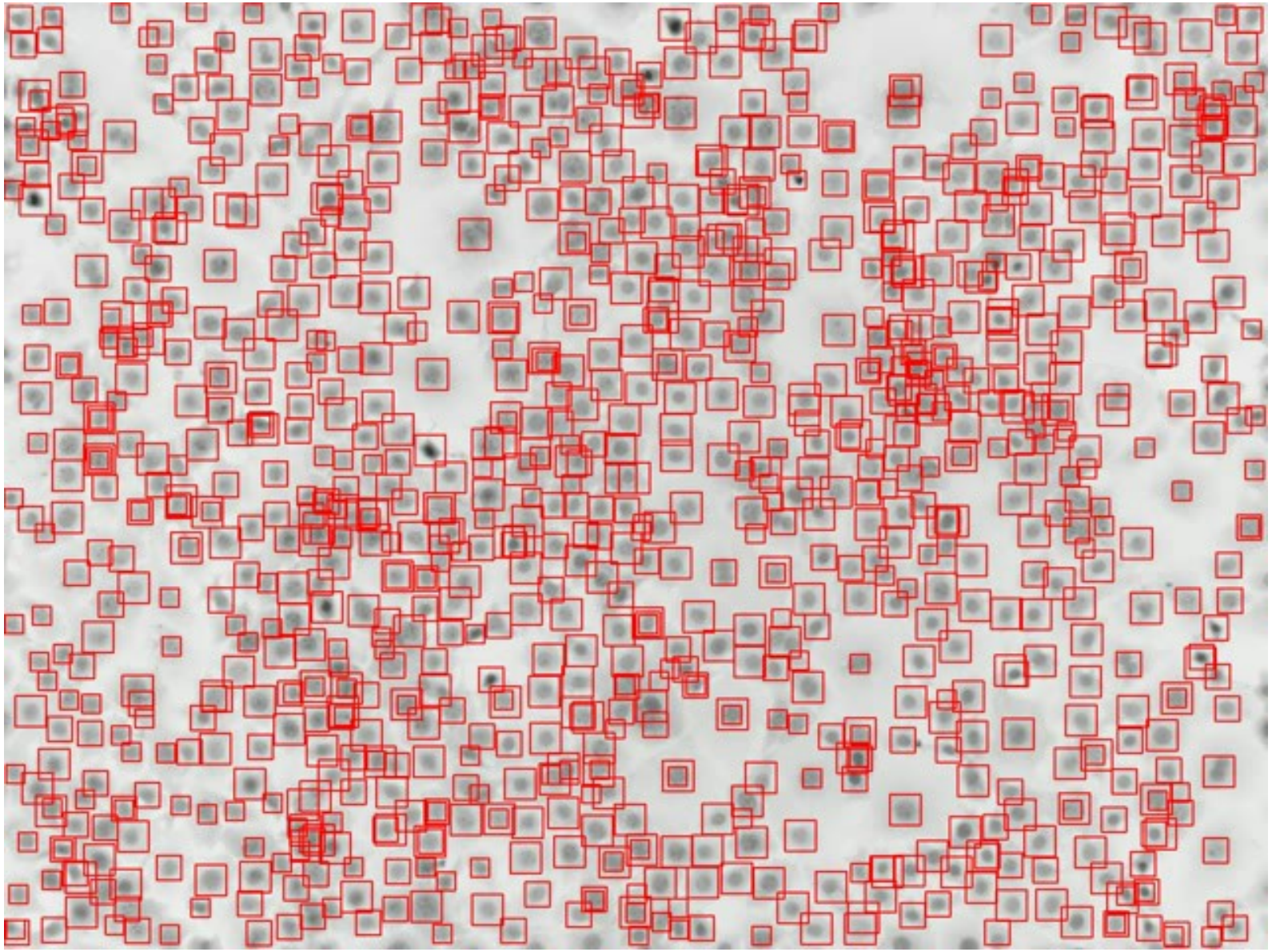


Grid parameter search for RBF kernel



Application: automatic cell counting / cell architecture (position) evaluation





http://www.durham.ac.uk/toby.breckon/demos/cell

A *probabilistic* interpretation ….

Learning Probability Distributions

- **Bayes** Formula

$$P(\omega_j|x) = \frac{p(x|\omega_j)\,P(\omega_j)}{p(x)} = \frac{p(x|\omega_j)\,P(\omega_j)}{\sum_j p(x|\omega_j)P(\omega_j)}$$

in words:

$$posterior = \frac{likelihood \times prior}{evidence}$$

- Assign observed feature vector x to maximally probable class ω_k for $j=\{1 \rightarrow k\ classes\}$

Thomas Bayes (1701-1761)

- Captures:
  - **prior probability** of occurrence
    - e.g. from training examples
  - **probability** of each class **given the evidence** (feature vector)
  - Return most probable **(Maximum A Posteriori – MAP)** class
  - Optimal (costly) Vs. Naive (simple)

"Approx" State of the **Art**

- Recent approaches – **SVM (+ variants), Decision Forests (+ variants), Boosted Approaches, Bagging** ....
  - – *outperform* Neural Networks
  - – some renaissance in **Deep Learning**
  - – can find **maximally optimal solution (SVM)**
  - – **less prone to over-fitting**
  - – **allow for extraction of meaning**
    (e.g. if-then-else rules for tree based approaches)

- Several other ML approaches
  - – **clustering** – k-NN, k-Means in multi-dimensional space
  - – **graphical models**
  - – **Bayesian** methods
  - – **Gaussian Processes** (largely regression problems – sweeping generalization!)

But how do we evaluate how well it is working* ?

* and produce convincing results for our papers and funders

Evaluating **Machine Learning**

- *For classification problems ....*

True Positives (TP) 

- example **correctly classified as** an +ve instance of given class A

False Positives (FP)

- example **wrongly classified** as an +ve instance of given class A
  - i.e. it is not an instance of class A

True Negatives (TN) 

- example **correctly classified** as an -ve instance of given class A

False Negatives (FP)

- example **wrongly classified** as an -ve instance of given class A
  - i.e. classified as not class A but is a true class A

Evaluating **Machine Learning**

- **Confusion Matrices**
  - Table of TP, FP, TN, FN
  - e.g. 2 class labels {yes, no}

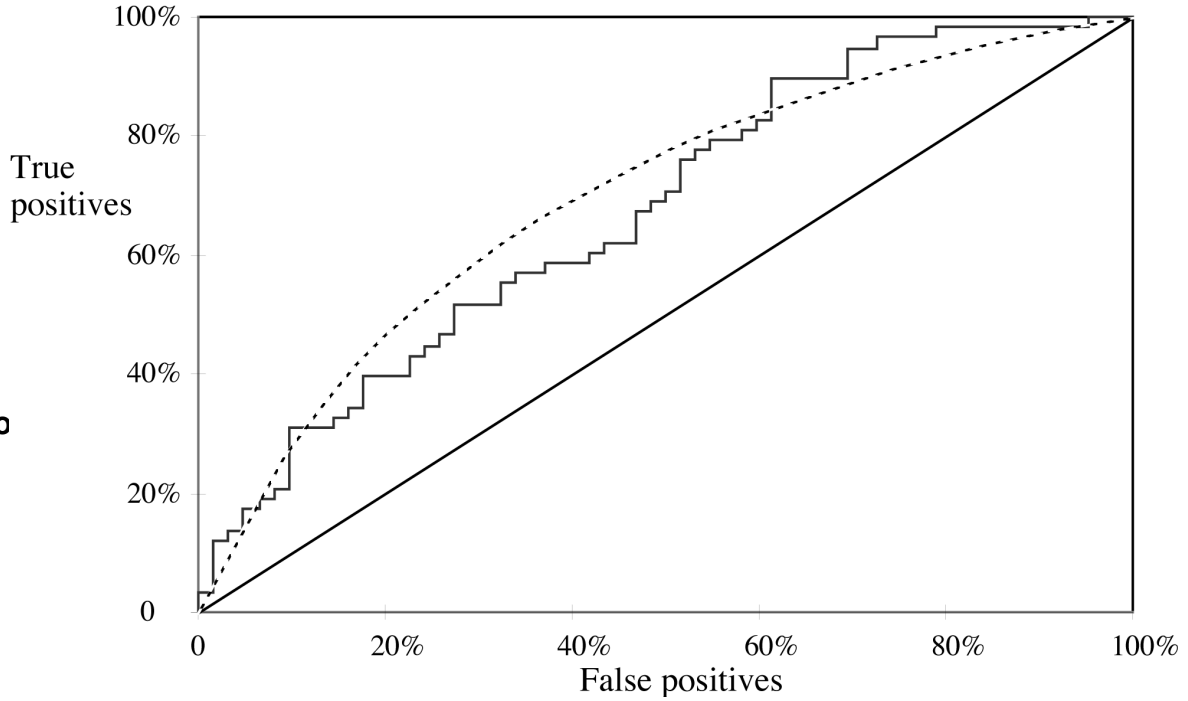
		Predicted class	
		Yes	No
Actual class	Yes	True positive	False negative
	No	False positive	True negative

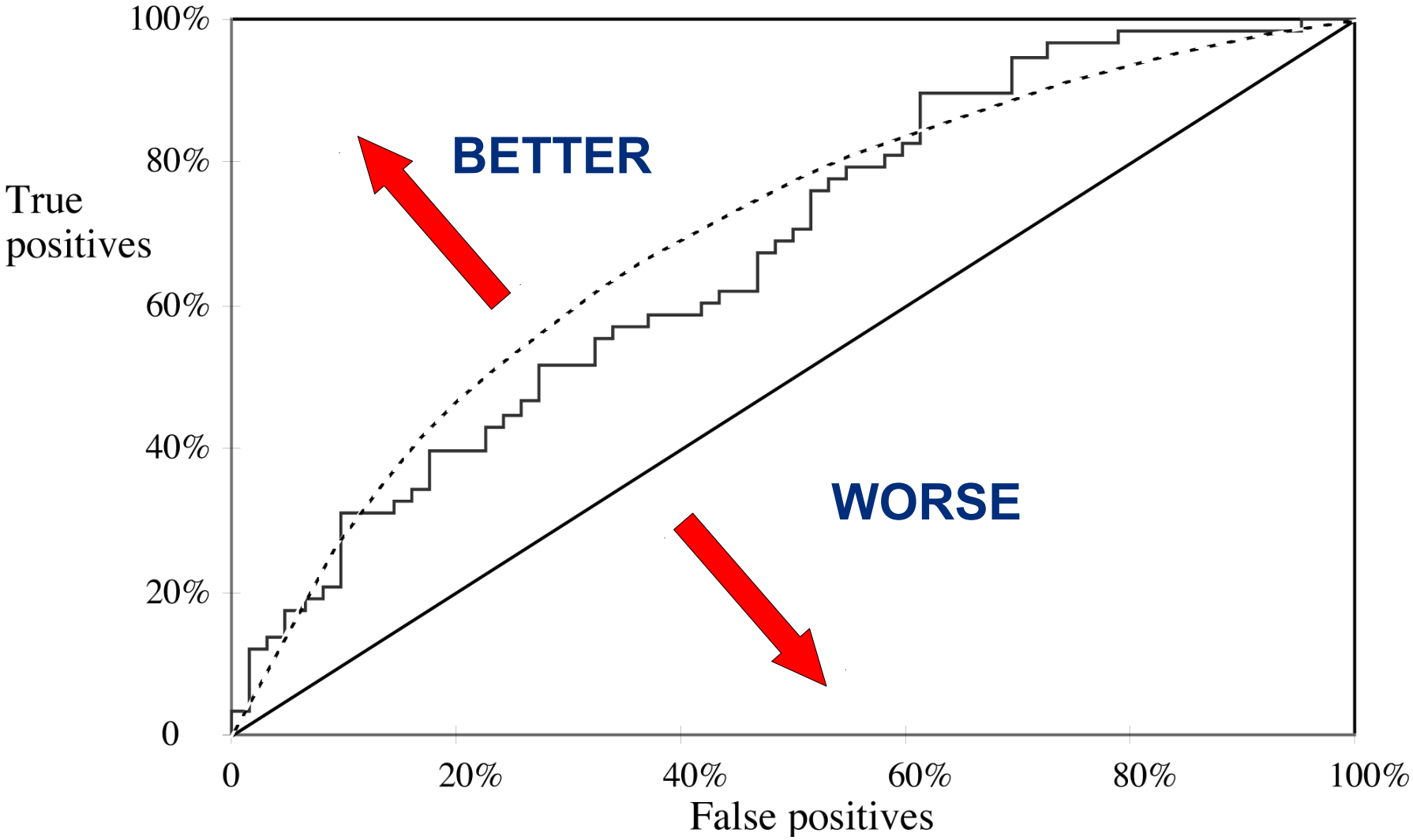
  - can also show TP, FP, TN, FN weighted by cost of mis-classification Vs. true-classification

N.B. A common name for "actual class" (i.e. the true class) is "ground truth" or "the ground truth data".

Evaluating **Machine Learning**

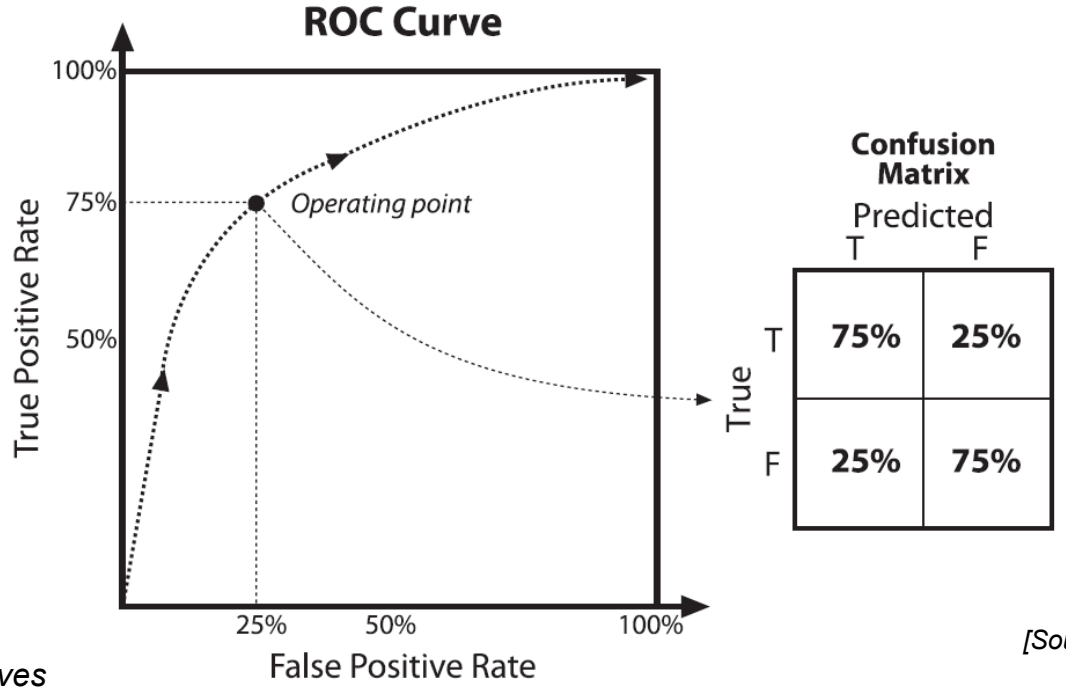
- **Receiver Operating Characteristic (ROC) Curve**
  - used to show **trade-off between hit rate and false alarm rate** over noisy channel (in communications originally)
    - here a "noisy" (i.e. error prone) classifier
  - **% TP Vs. %FP**
  - **"jagged steps" = actual data**
  - **- - - - = averaged result (over multiple cross-validatio folds) or best line fit**




100%
80%
60%
40%
20%
0
True positives
BETTER
WORSE
0
20%
40%
60%
80%
100%
False positives

Evaluating **Machine Learning**

- **Receiver Operating Characteristic (ROC) Curve**
  - Used to **compare different classifiers** on common dataset
  - Used to **tune a given classifier** on a dataset
    - by **varying a given threshold or parameter** of the learner that effects the TP to FP ratio



[Source: Bradski '09]

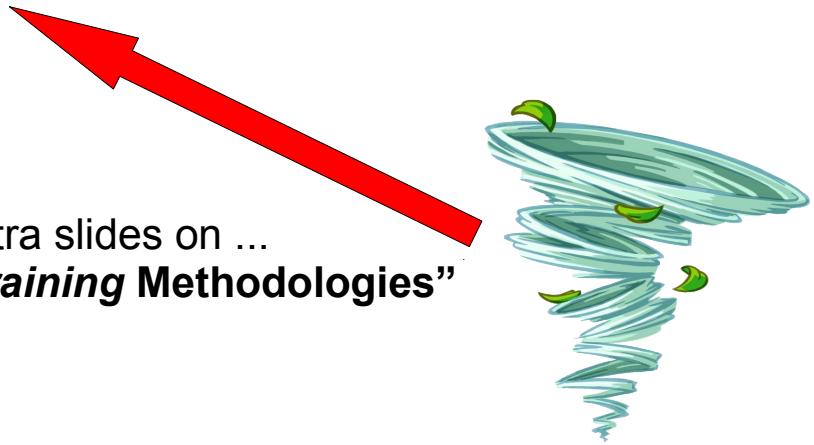
See also: precision/recall curves

Evaluating **Machine Learning**

- The ***key*** is:

Robust experimentation on **independent training and testing sets**

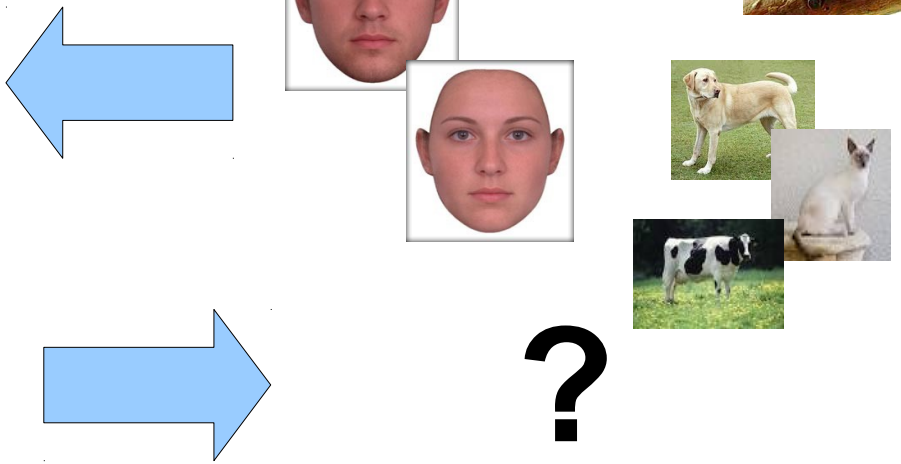
– *Perhaps using cross-validation or similar*



.. see extra slides on ...
"*Data Training* Methodologies"

Many, many ways to perform machine learning ....

ML "classifier"



… we have seen *(only)* some of them.

(very briefly!)

Which one is "the best" ?

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 … }**

What we have seen ...

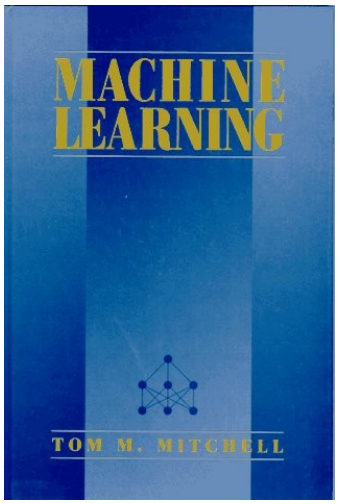
- The power of combining *simple* things ….
  - **Ensemble Classifiers**
    - Decision Forests / Random Forests
      - *concept extends to all ML approaches*

- **Neural** Inspired Approaches
  - Neural Networks
    - *many, many variants*

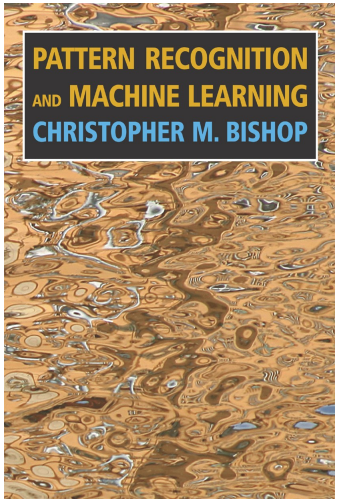
- **Kernel** Inspired Approaches
  - Support Vector Machines
    - *beginning of the story*

Further Reading - *textbooks*

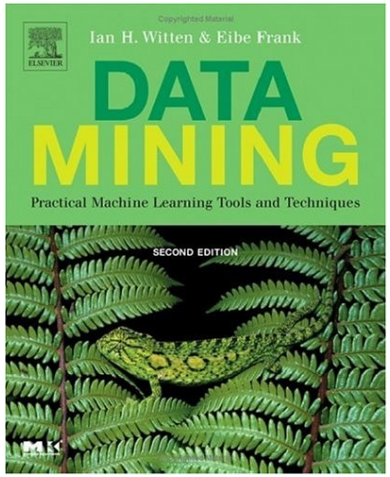
- **Machine Learning - Tom Mitchell (McGraw-Hill, 1997)**



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



- *Data Mining - Witten / Frank (Morgan-Kaufmann, 2005)*



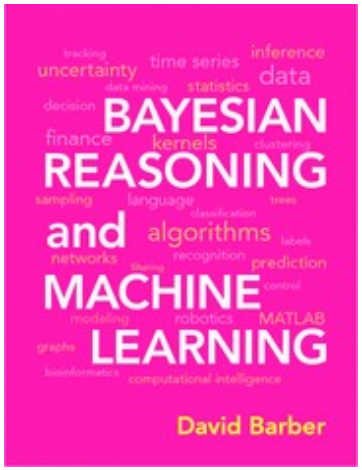
Further Reading - *textbooks*

- **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***

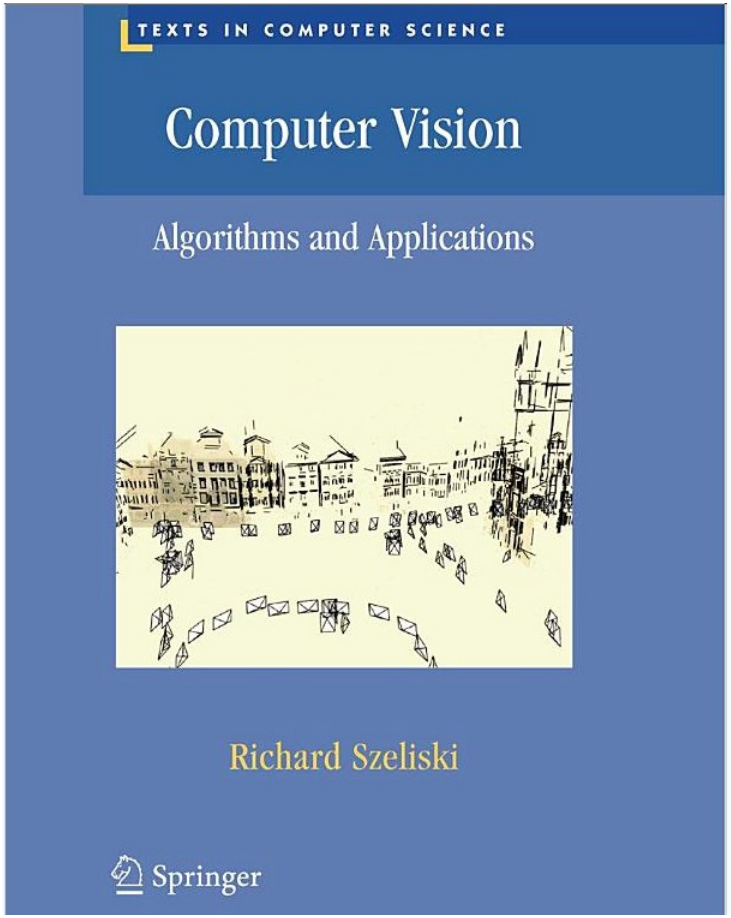
  – Simon Prince

  (Springer, 2012)

  http://www.computervisionmodels.com/

  *… both very **probability driven**, both available as **free PDF online** (woo, hoo!)*

Computer Vision – general overview



- Computer Vision: Algorithms and Applications
  - *Richard Szeliski, 2010 (Springer)*

- PDF download:
  - http://szeliski.org/Book/

Supporting specific content from this machine learning lecture:

- see Chapter 14

Publications – *a selection*

- **Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning** - A. Criminisi et al., in Foundations and Trends in Computer Graphics and Vision, 2012.
- **Three things everyone should know to improve object retrieval.** Arandjelovic, R., & Zisserman,. Computer Vision and Pattern Recognition (CVPR), 2012.
- **OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,** P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun: International Conference on Learning Representations 2014.

- **Image Classification using Random Forests and Ferns**

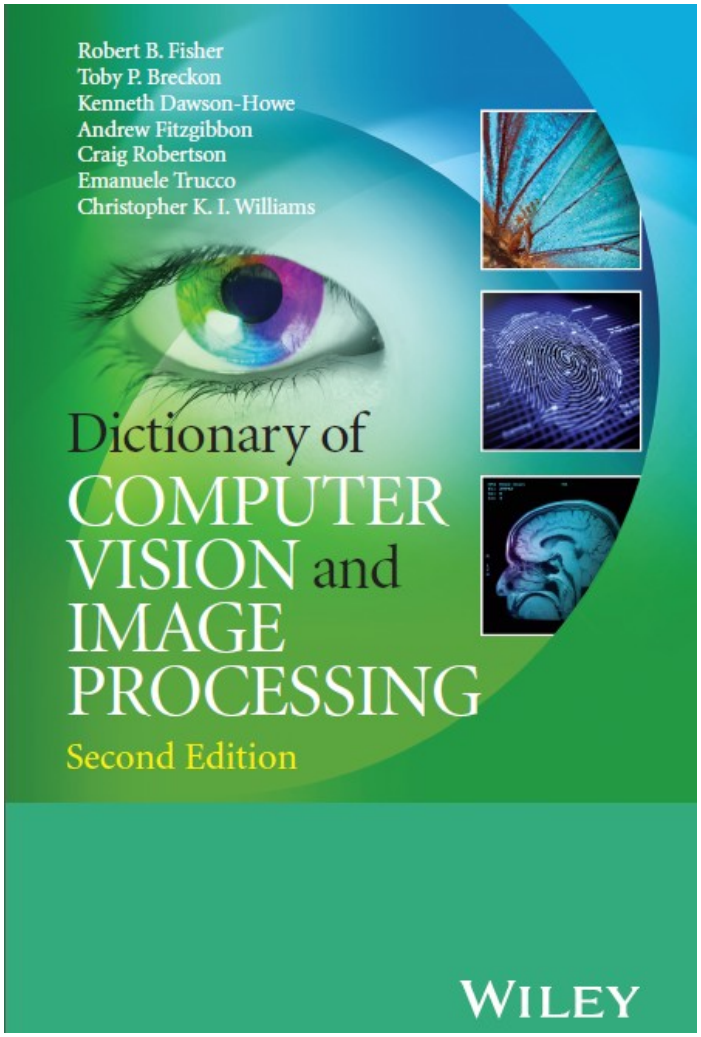
  A Bosch, A Zisserman, X Munoz – Int. Conf. Comp. Vis., 2007.
- **Robust Real-time Object Detection**

  P Viola, M Jones - International Journal of Computer Vision, 2004.

- **The PASCAL Visual Object Classes (VOC) challenge**

  M Everingham, L Van Gool, C Williams, J Winn, A Zisserman - International Journal of Computer Vision, 2010.

Final *shameless* plug ….



- Dictionary of Computer Vision and Image Processing

  R.B. Fisher, T.P. Breckon, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, E. Trucco, C.K.I. Williams, Wiley, 2014.

  – ***… maybe it will be useful!***

That's all folks ...

Slides, examples, demo code,links + ***extra supporting slides*** *@ www.durham.ac.uk/toby.breckon/mltutorial/*