Unit 7
Object Recognition

- Nearest Neighbor Classifier
- Bayesian Classifier
- Support Vector Machine
- Random Forest

Outline

- Fundamentals of Pattern Recognition
- K-Nearest-Neighbor Classifier
- Bayesian Classification
- Support Vector Machine
- Random Forest

What is Pattern Recognition?

- A pattern is an entity, vaguely defined, that could be given a name, e.g.,
  - fingerprint image,
  - handwritten word,
  - human face image,
  - speech signal,
  - DNA sequence,
  - document
  - ….
- Pattern recognition is the study of how machines can
  - observe the environment,
  - learn to distinguish patterns of interest,
  - make sound and reasonable decisions about the categories of the patterns.

Human Perception

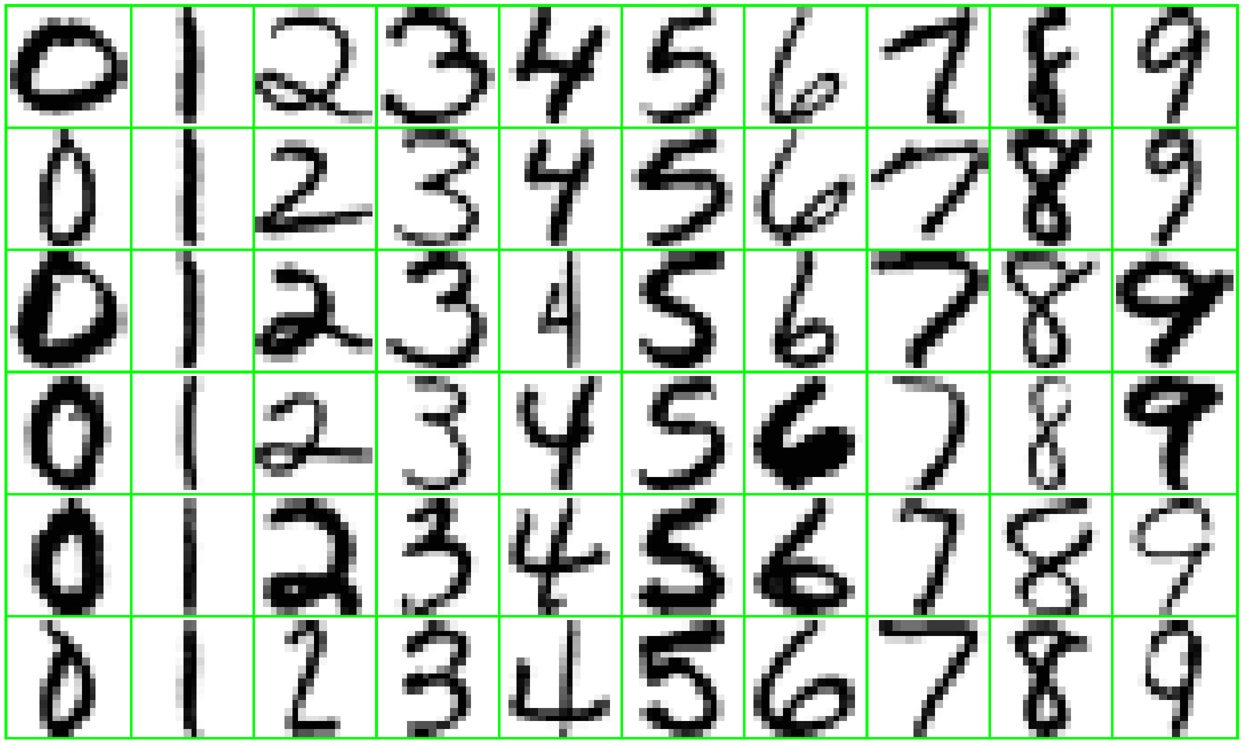
- Humans have developed highly sophisticated skills for sensing their environment and taking actions according to what they observe, e.g.,
  - recognizing a face,
  - understanding spoken words,
  - reading handwriting,
  - distinguishing fresh food from its smell.
- We would like to give similar capabilities to machines.

Goal of Pattern Recognition

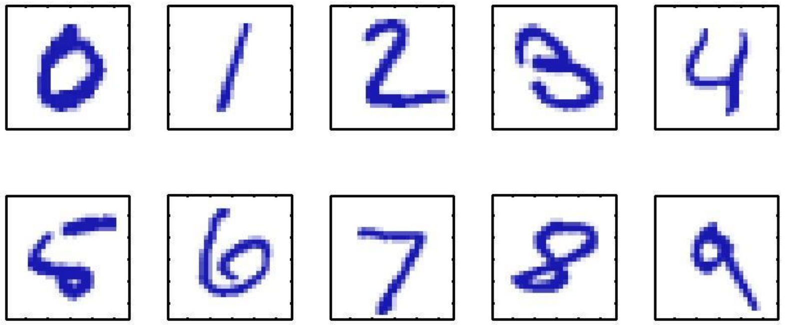
- Using past experiences to improve future performance.
- For a machine, experiences come in the form of data.
- What does it mean to improve performance?
- – Learning is guided by an objective, associated with a particular notion of loss to be minimized (or, equivalently, gain to be maximized).
- Why machine learning?
- We need computers to make informed decisions on new, unseen data.
- Often it is too difficult to design a set of rules “by hand”.
- Pattern recognition system is trained from a training data set to automatically extract relevant information from new data and make decisions.

Handwritten Digit Recognition

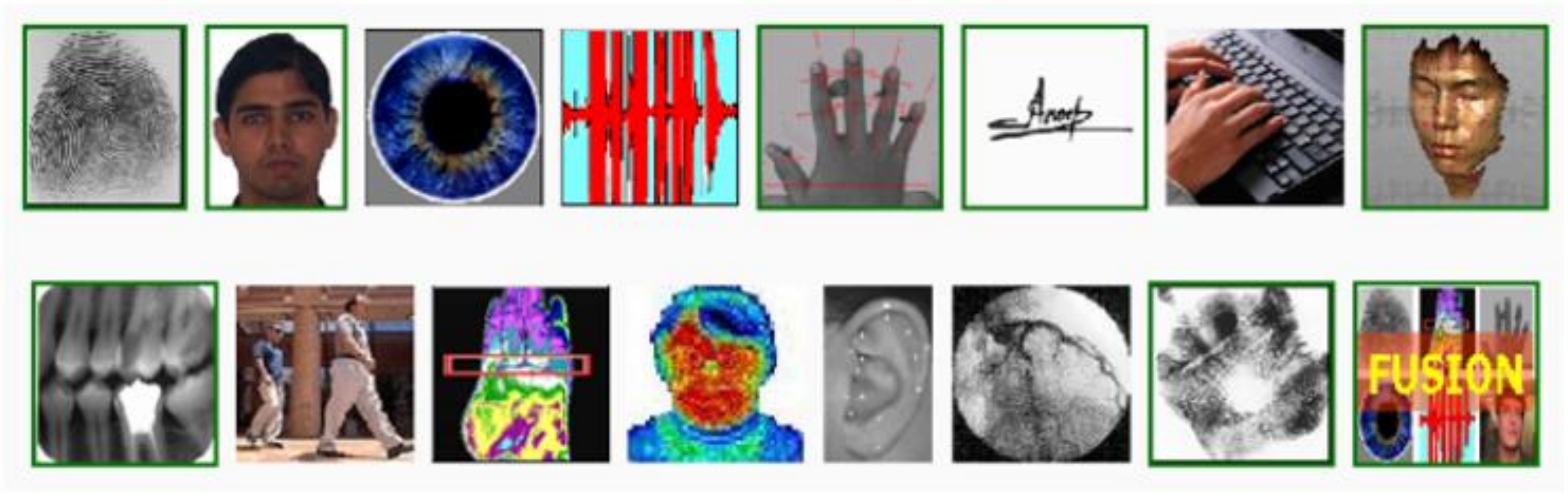
Training Data



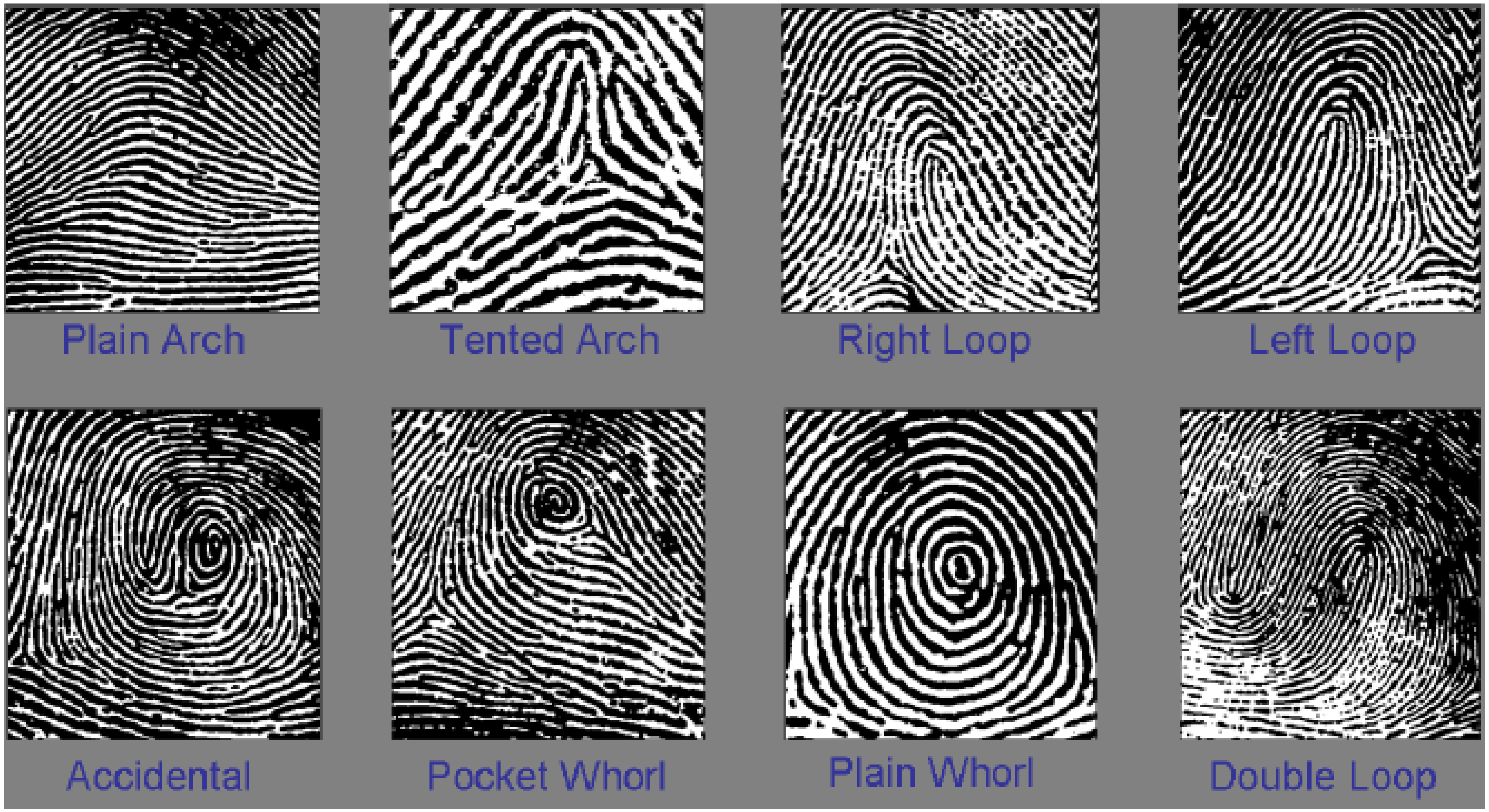
Testing Data



Biometric Recognition

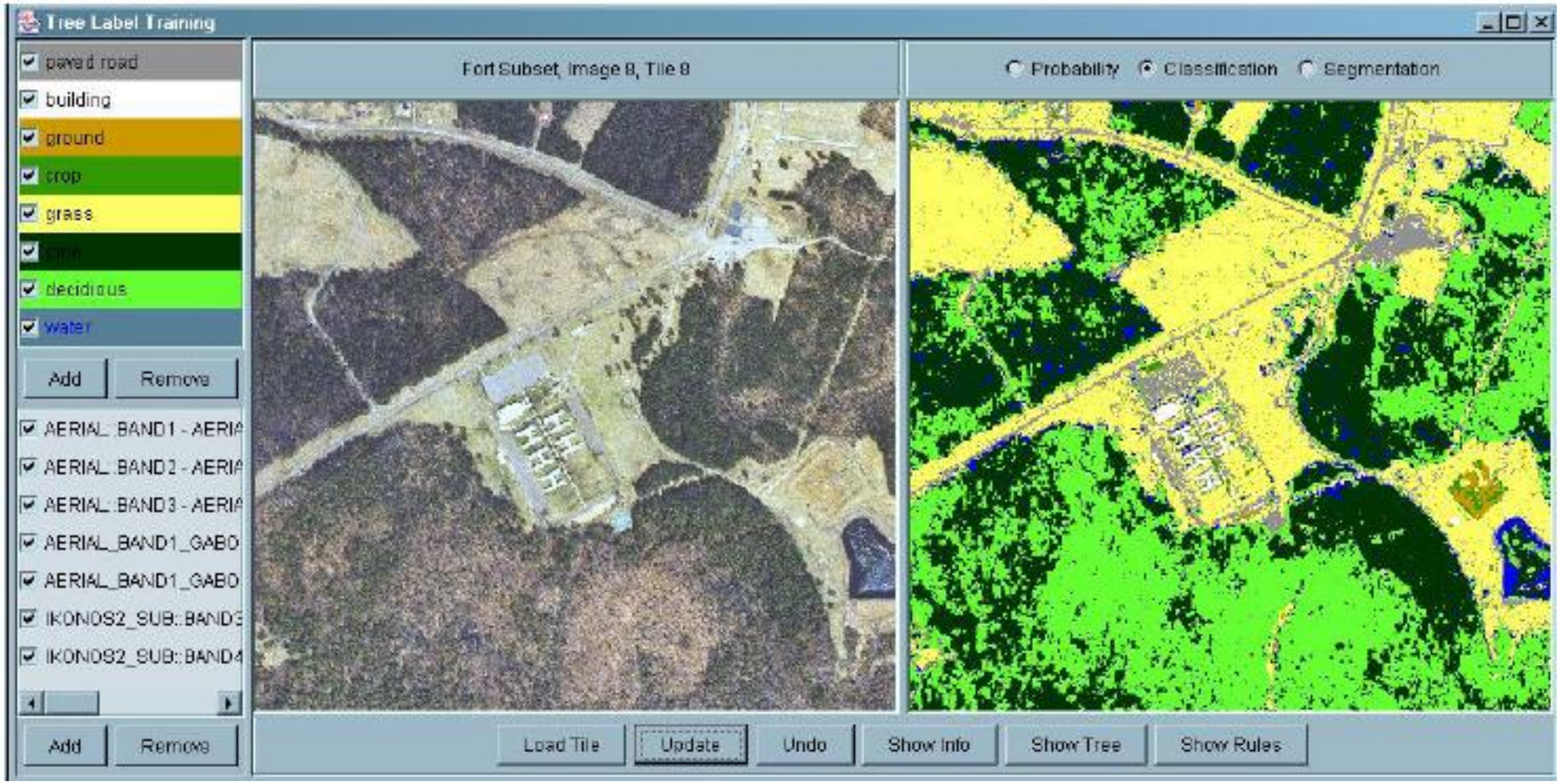


Fingerprint Recognition



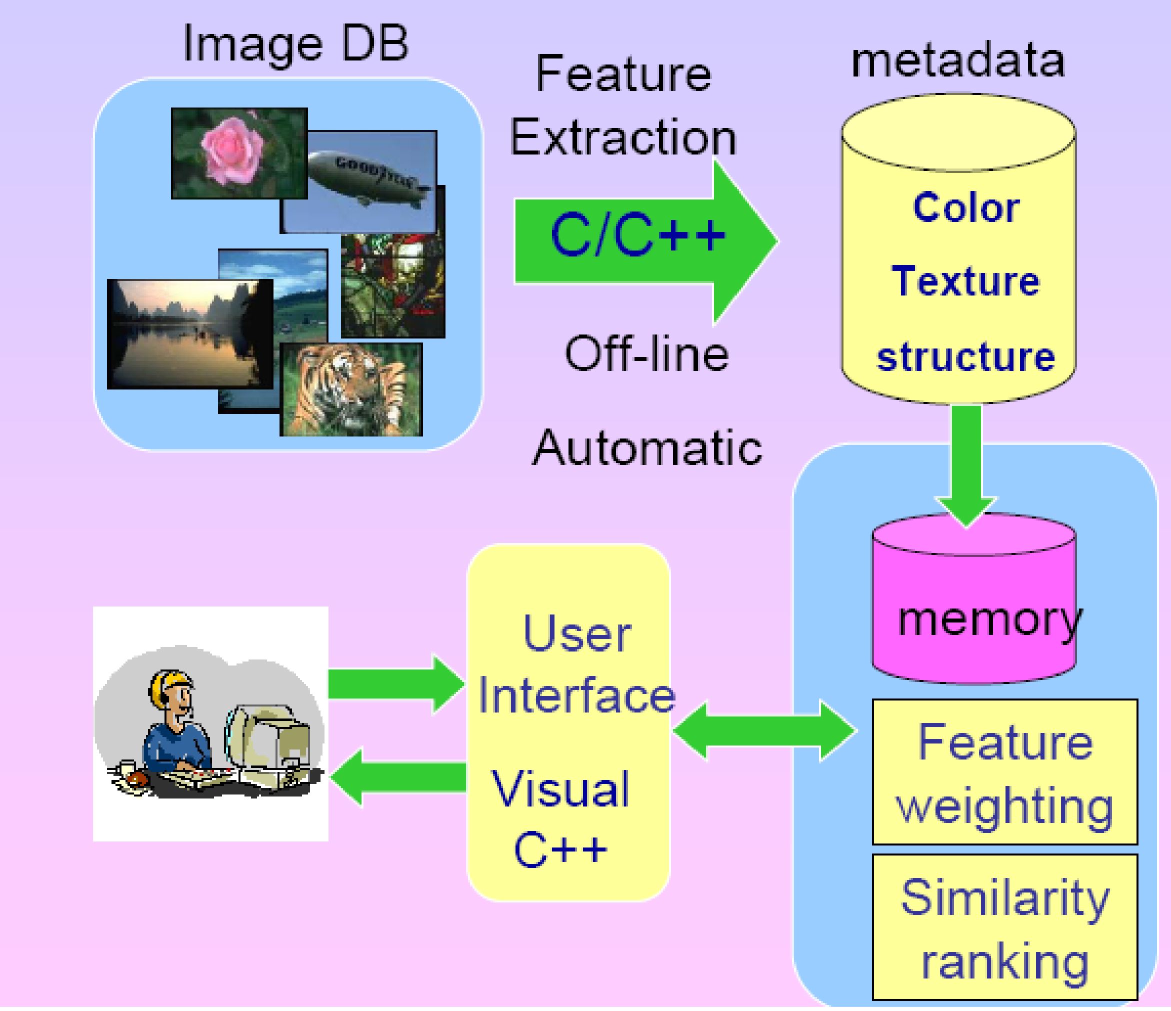
Face Recognition

Satellite Image Classification



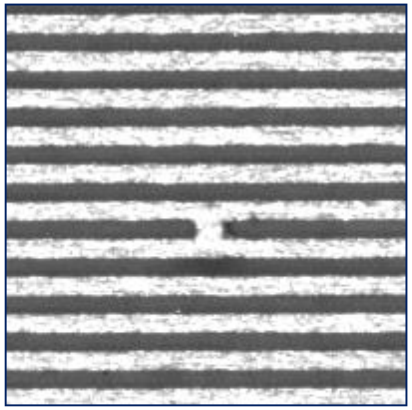
License Plate Recognition

Content-Based Image Retrieval

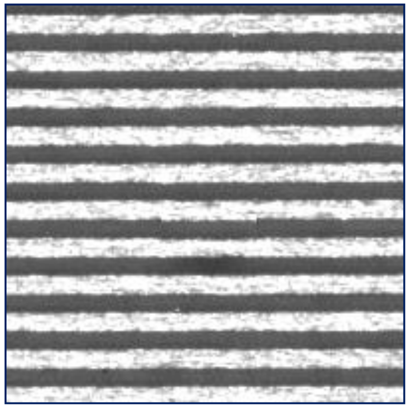


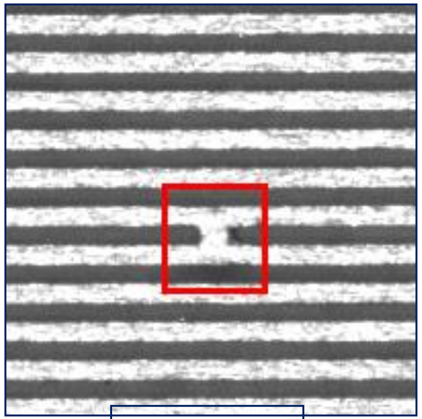
- **Color**
  - ✓Color histogram
  - ✓Color moments
  - ✓Color correlogram
- **Texture**
  - ✓Tamura texture
  - ✓Co-occurrence matrices
  - ✓Gabor features
  - ✓Wavelet moments
- **Shape**
  - ✓Fourier descriptor
- **Structure**
  - ✓Edge-based features

Automated Optical Inspection (AOI)



$$\begin{tikzcd}
\text{Inspection Image} \arrow[d] & \\
\text{Image Alignment} \arrow[d] & \text{Reference Image} \arrow[l] \\
\text{Defect Detection} \arrow[d] & \\
\text{Defect Classification} \arrow[d] & \\
\text{Inspection Result} &
\end{tikzcd}$$



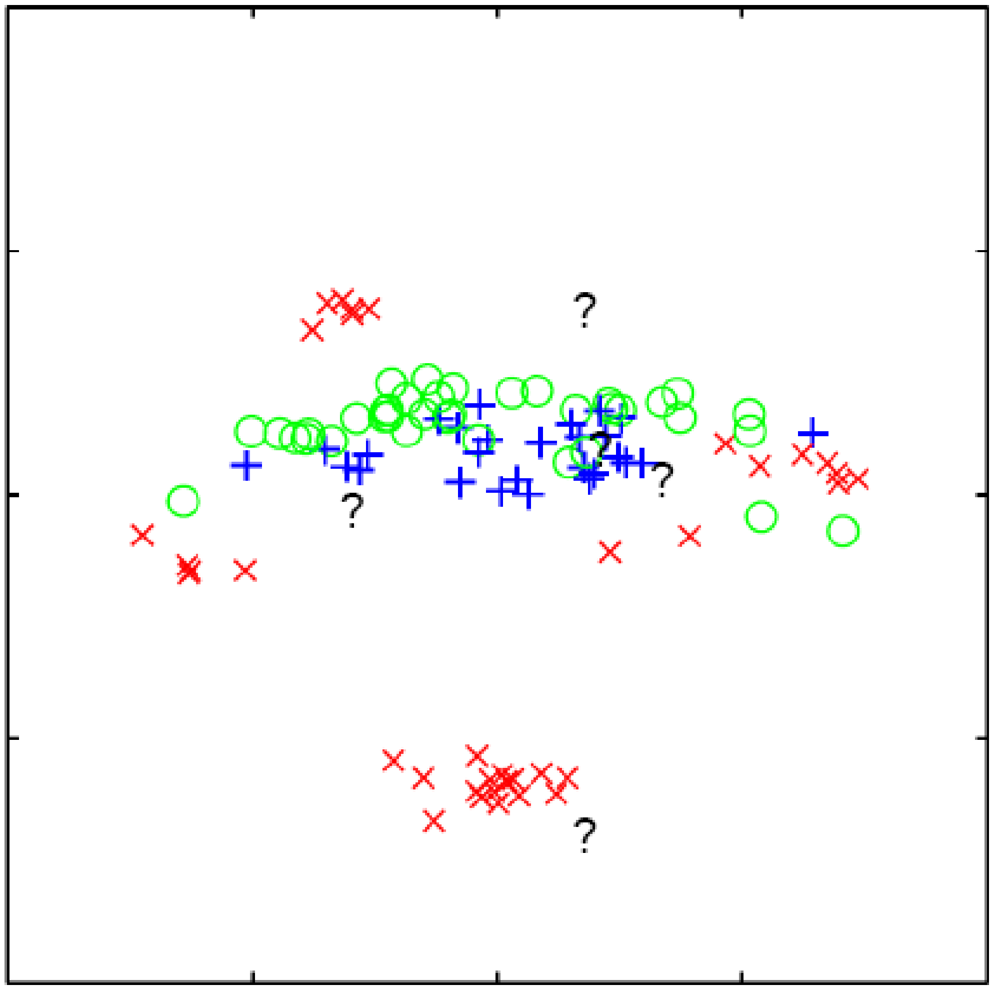


(short)

Classification: Definition

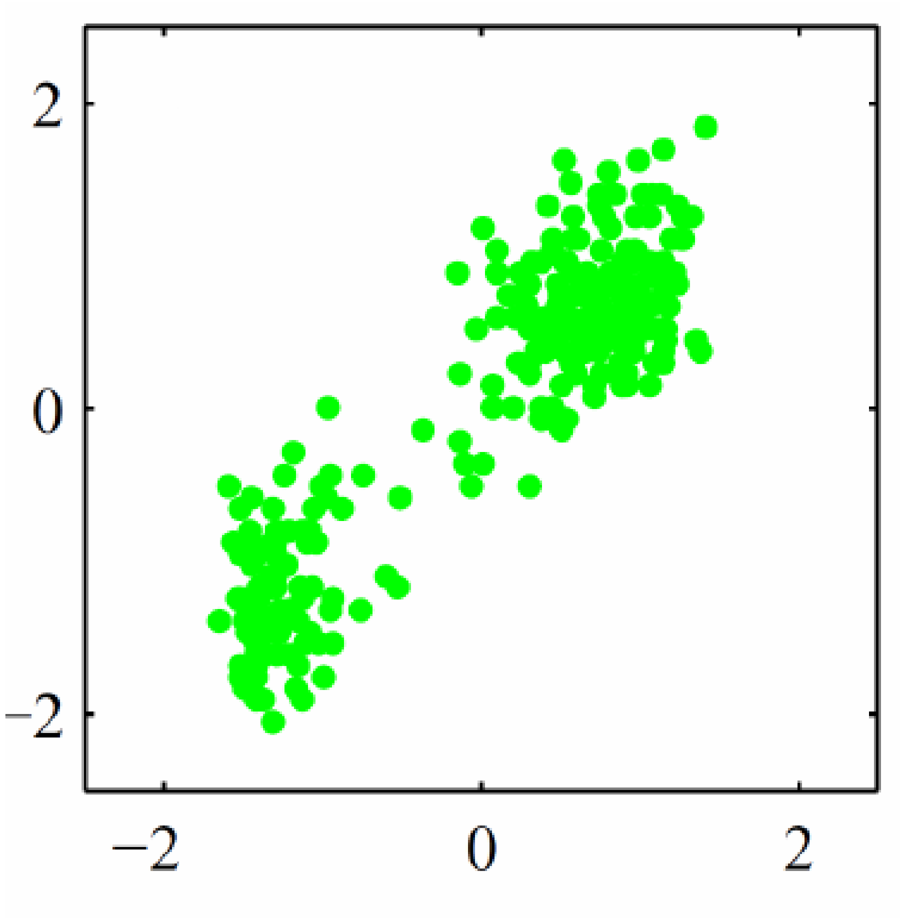
- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Supervised Classification



- Training: Colored points are given and used for training (dividing feature space)
- Testing: Labels for test samples (question marks) can be inferred

Clustering (Unsupervised)

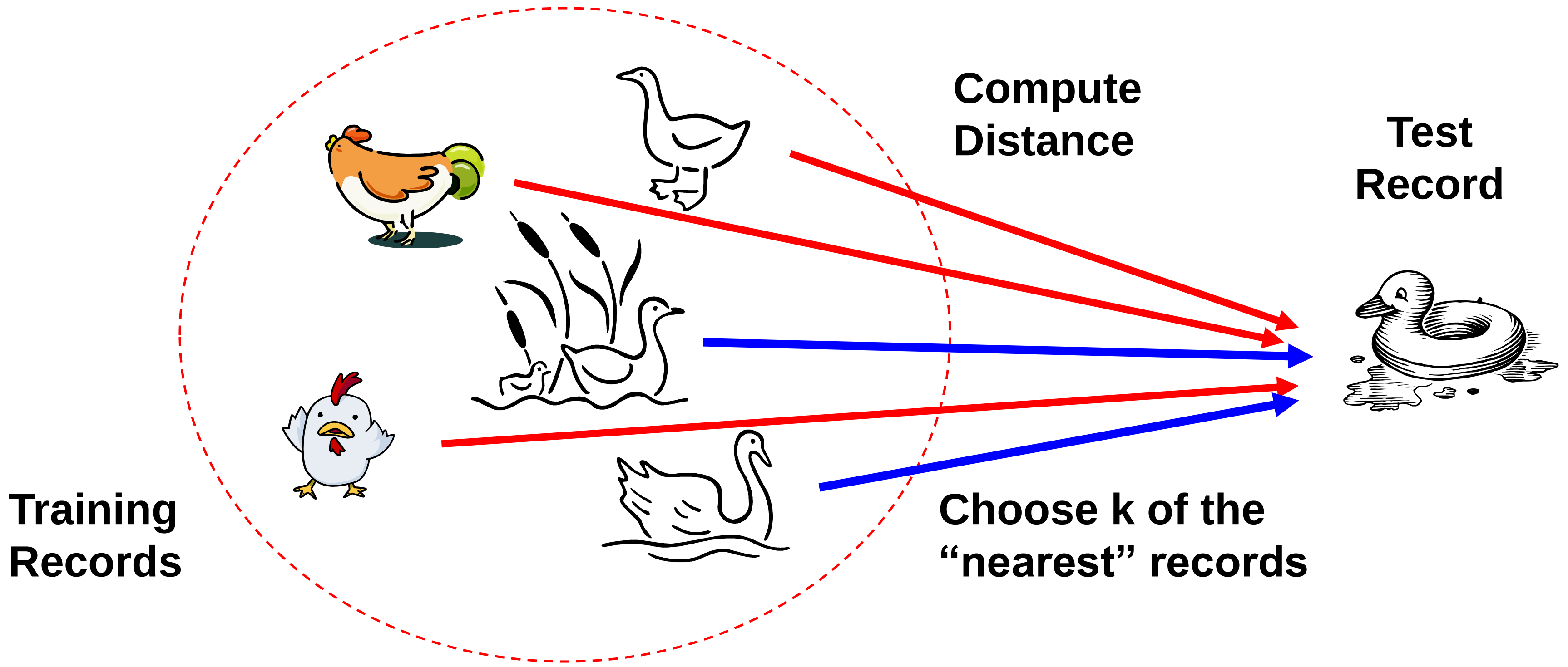


- Assign each data point to a cluster
- Data here suggests two clusters, some methods determine the number of clusters from the data, others use a predefined number of clusters

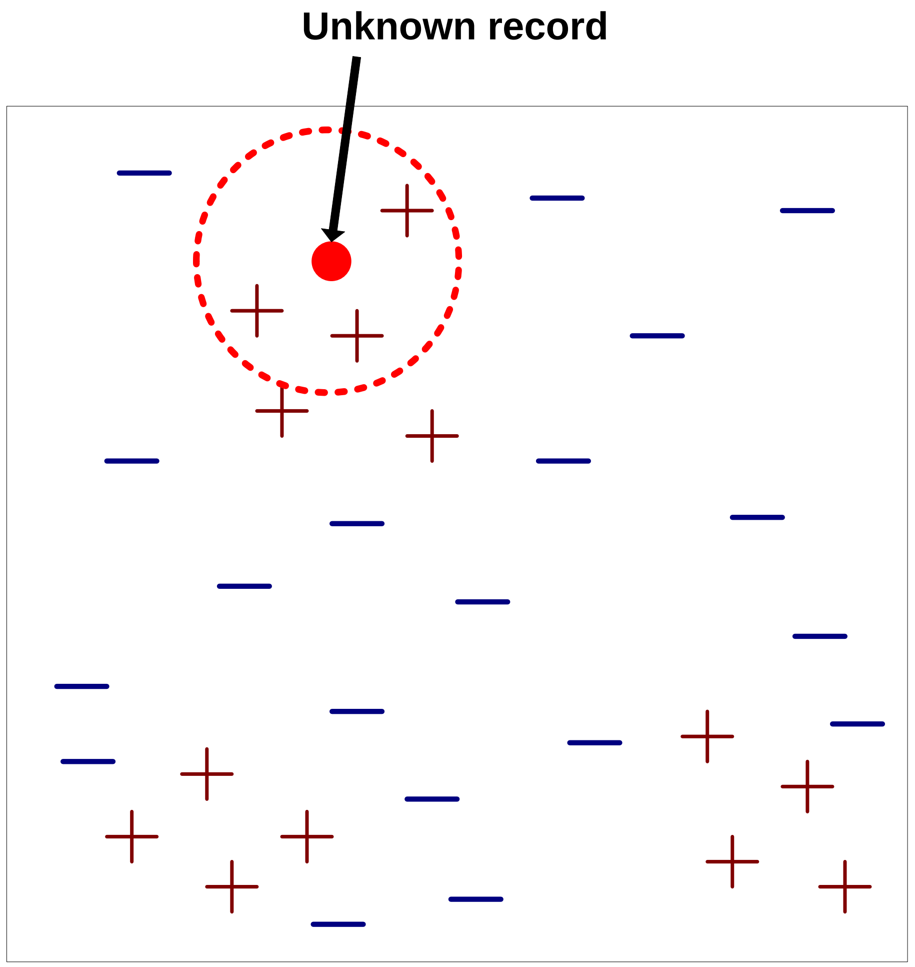
Nearest Neighbor Classifier

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



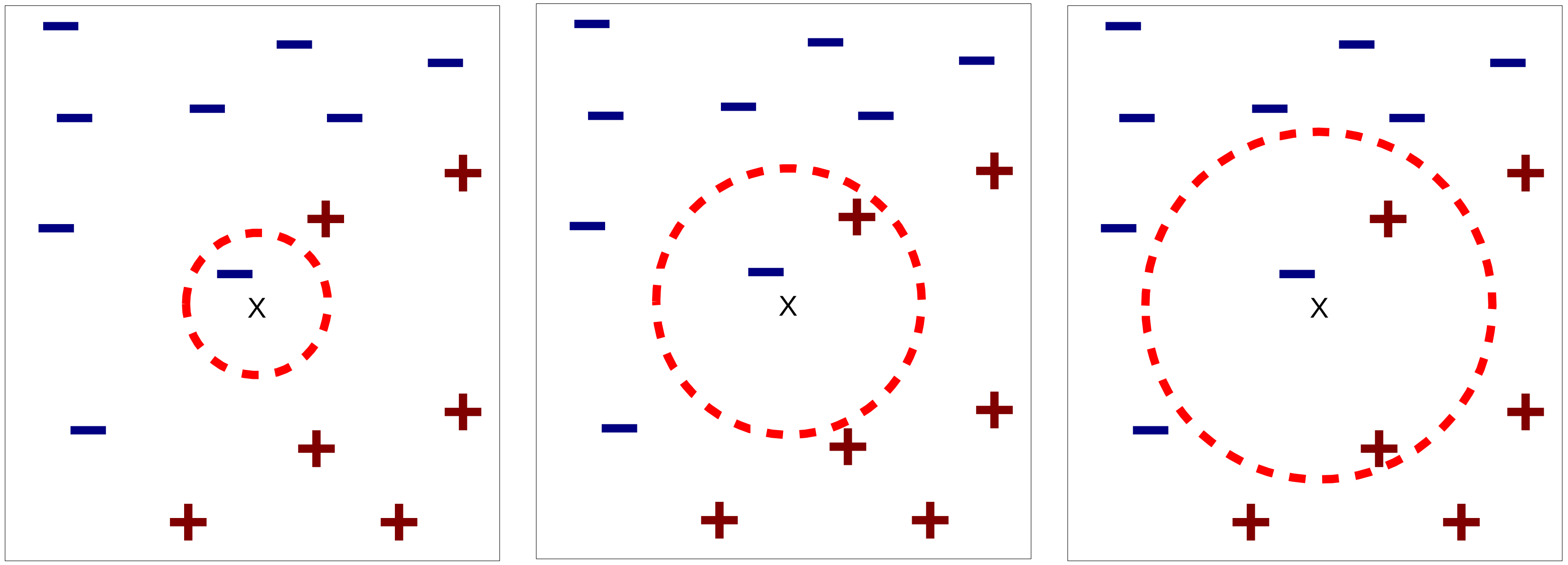
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification

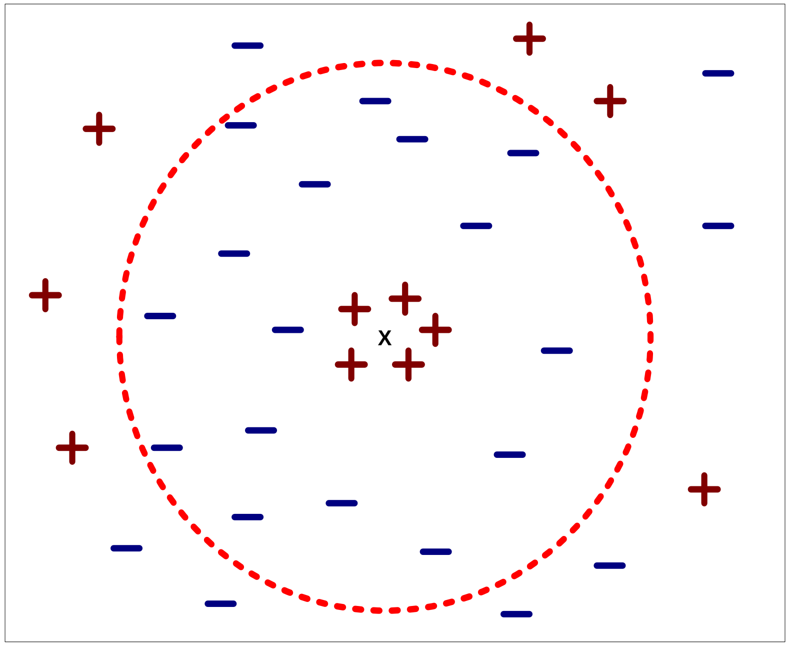
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive
- Selecting an appropriate distance metric for a specific classification problem is very crucial for k-NN classifiers

Bayesian Classification

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C \mid A) = \frac{P(A, C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A, C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C) P(C)}{P(A)}$$

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes $(A_1, A_2, \ldots, A_n)$
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes $P(C | A_1, A_2, \ldots, A_n)$

- Can we estimate $P(C | A_1, A_2, \ldots, A_n)$ directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$\boxed{P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$

  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C) P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

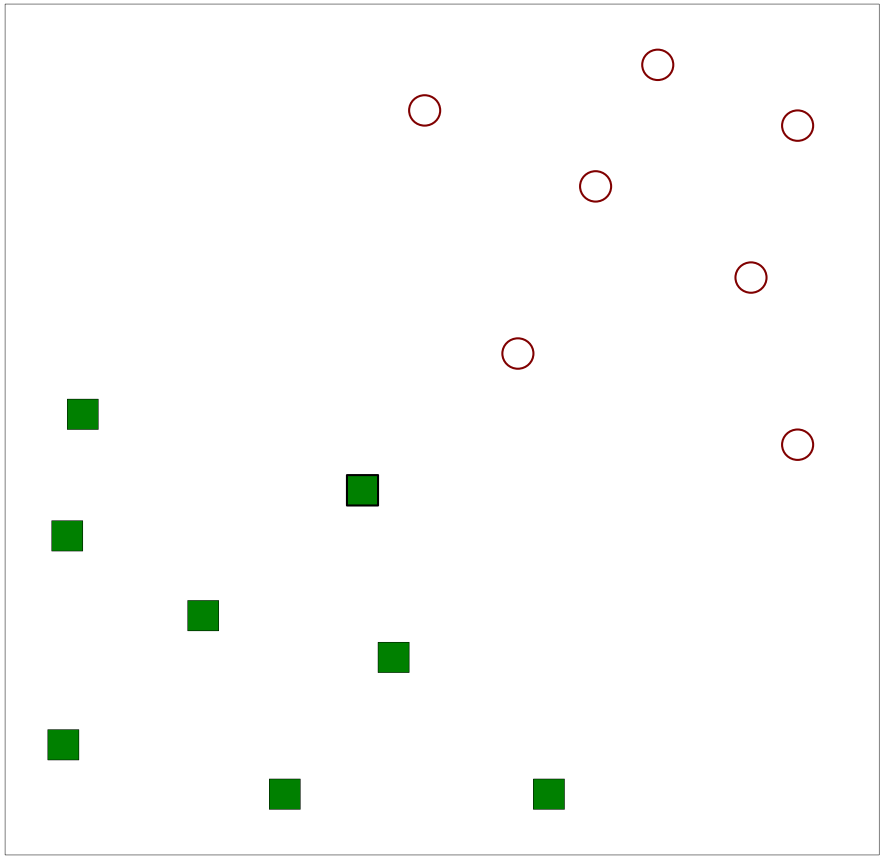
- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n | C) = P(A_1| C_j)\, P(A_2| C_j) \ldots P(A_n| C_j)$
  - Can estimate $P(A_i| C_j)$ for all A_i and C_j.
  - New point is classified to C_j if $P(C_j) \prod P(A_i| C_j)$ is maximal.

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

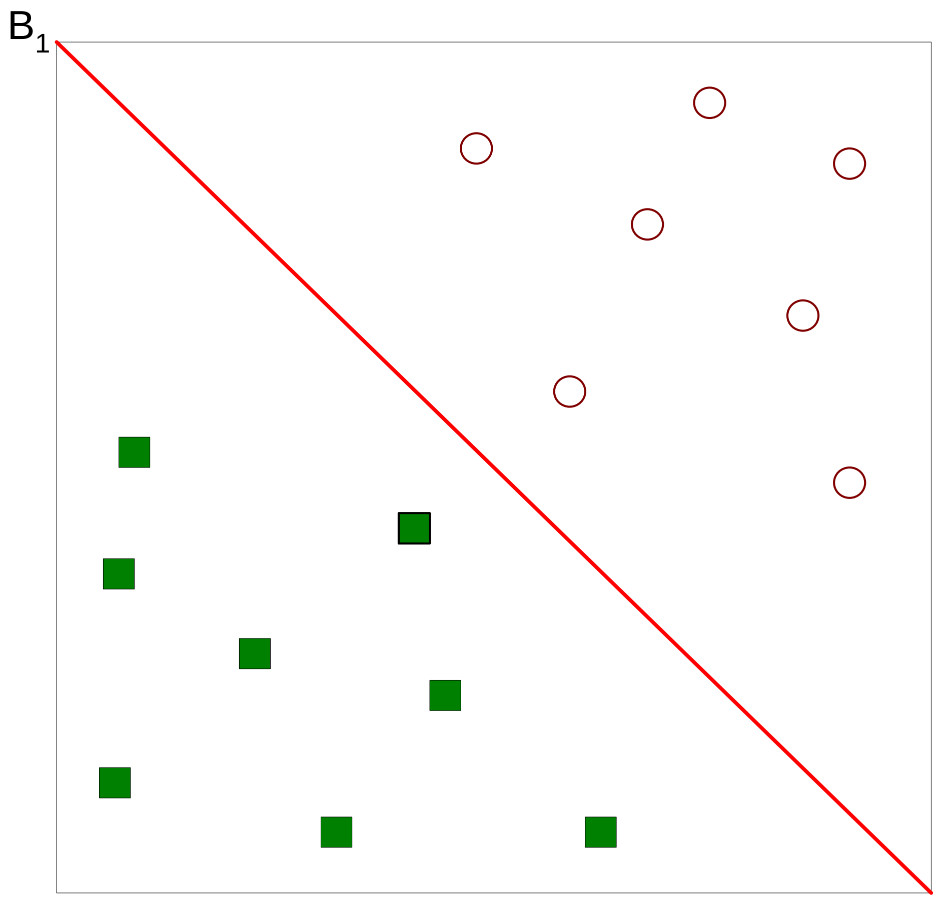
SVM (Support Vector Machine) Classification

Support Vector Machines



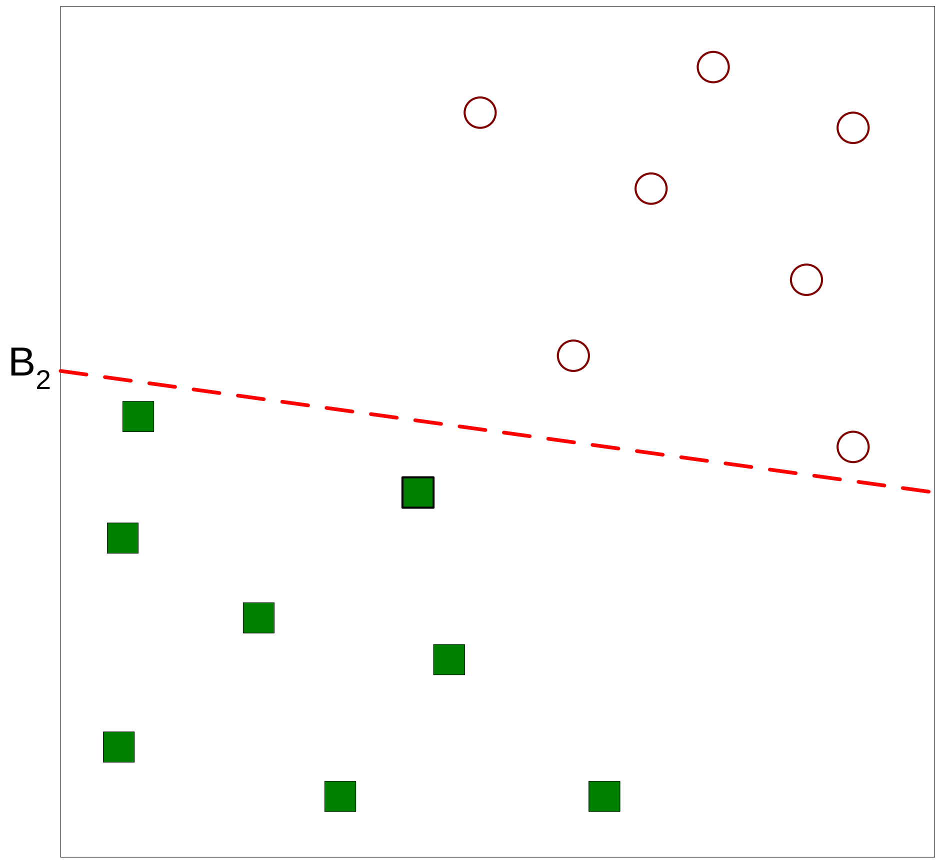
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



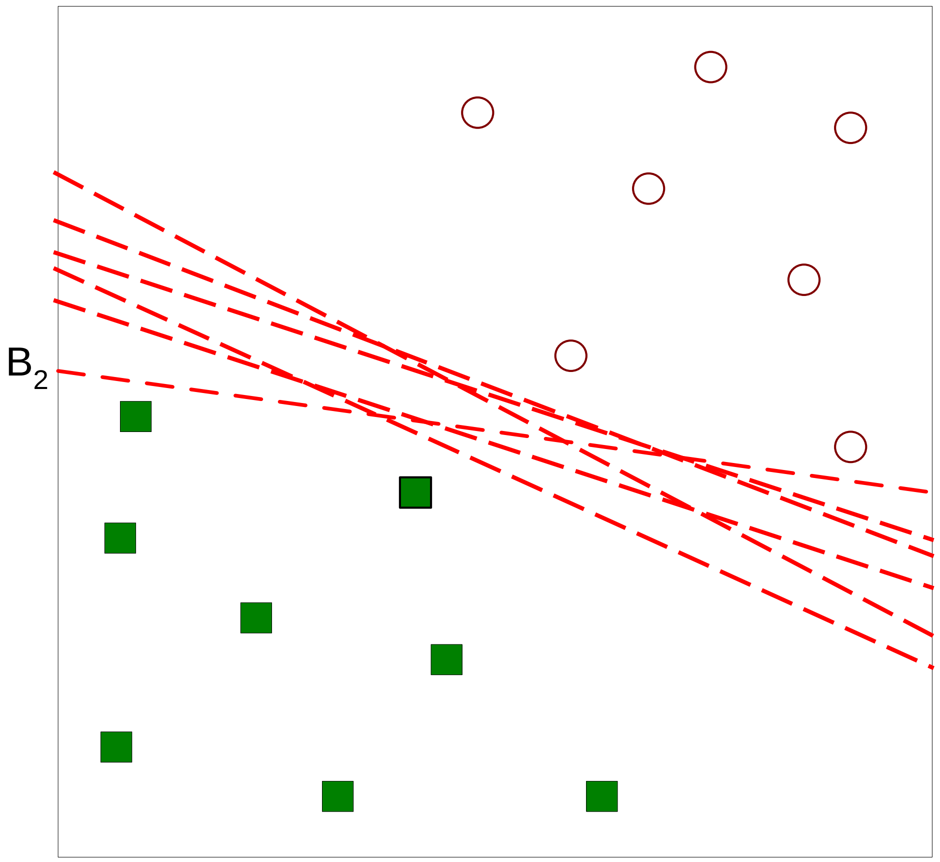
- One Possible Solution

Support Vector Machines



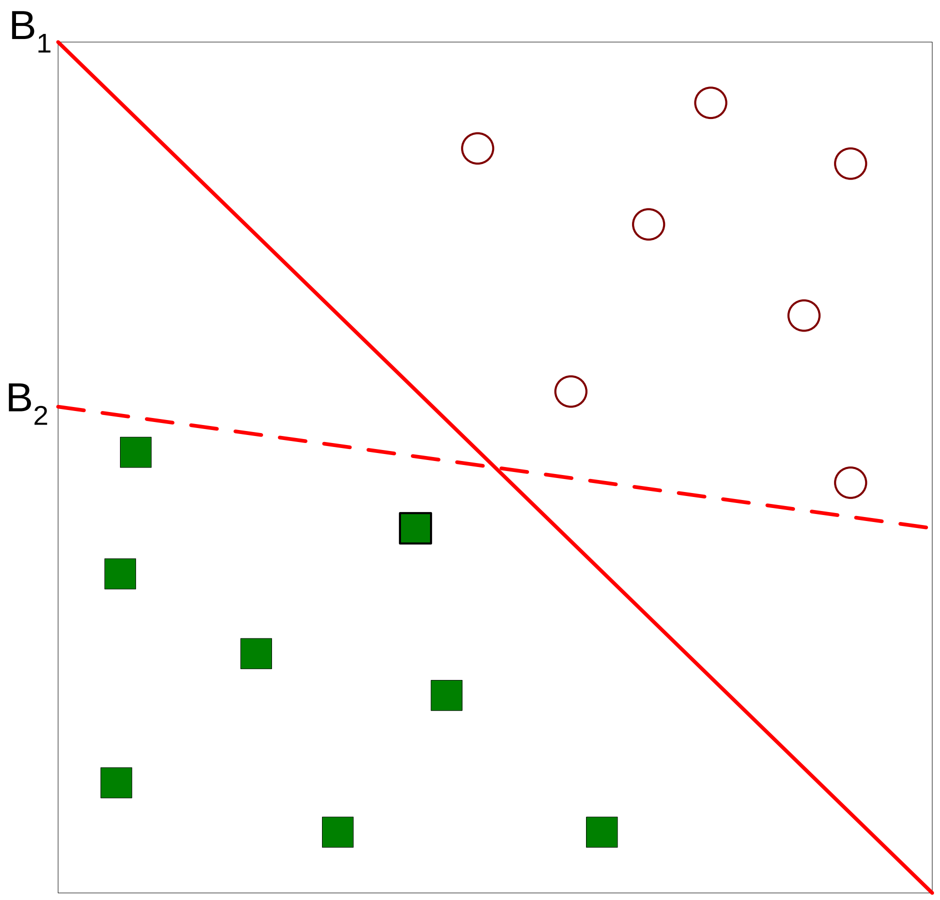
- Another possible solution

Support Vector Machines



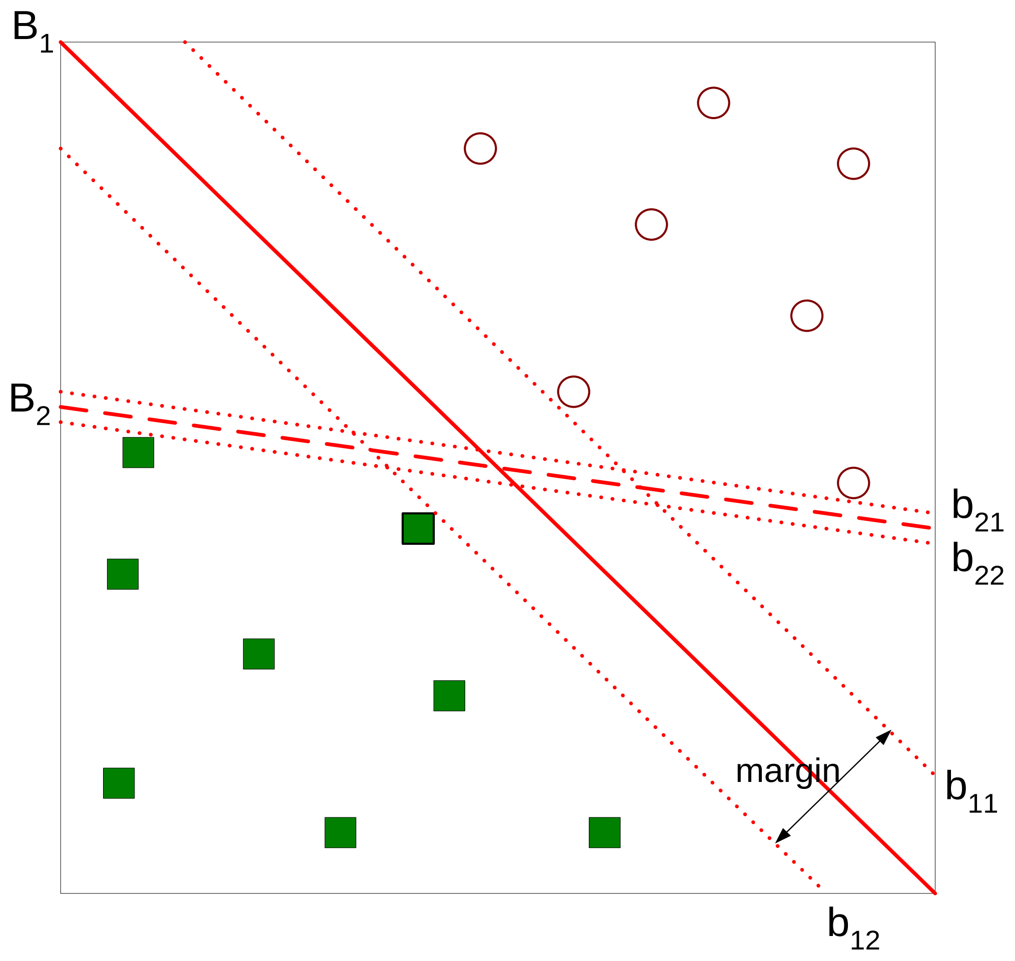
- Other possible solutions

Support Vector Machines



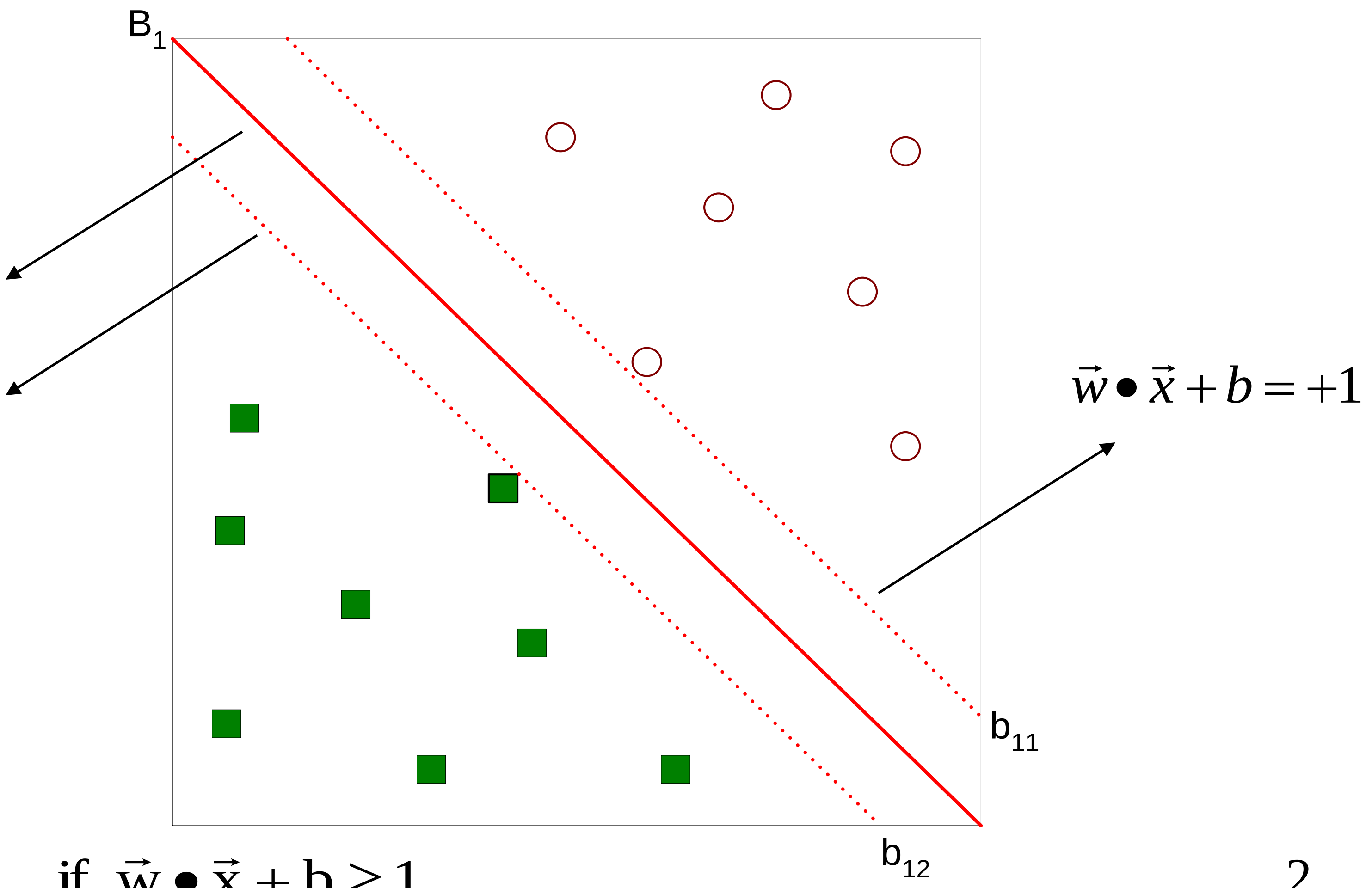
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines


B$_1$
$\vec{w}\bullet\vec{x}+b=0$
$\vec{w}\bullet\vec{x}+b=-1$
$\vec{w}\bullet\vec{x}+b=+1$
b$_{11}$
b$_{12}$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w}\bullet\vec{x}+b \geq 1 \\ -1 & \text{if } \vec{w}\bullet\vec{x}+b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

Support Vector Machines

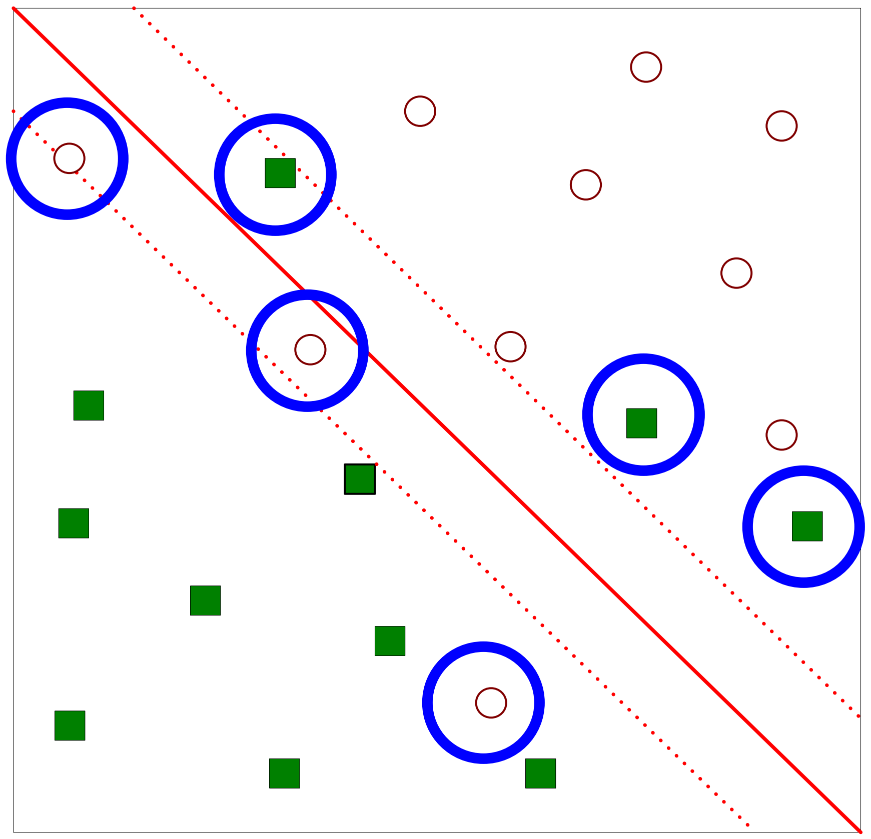
- We want to maximize: $\text{Margin} = \frac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing: $L(w) = \frac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 \end{cases}$$

    - This is a constrained optimization problem
      - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:

$$L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

    - Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq \boxed{1 - \xi_{\mathrm{i}}} \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq \boxed{-1 + \xi_{\mathrm{i}}} \end{cases}$$

Soft margin hyperplane

- By minimizing $\sum_i \xi_i$, ξ_i can be obtained by

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - ξ_i are "slack variables" in optimization
    - ◆Note that ξ_i=0 if there is no error for $\mathbf{x}_i$

- The optimization problem becomes (*C* : tradeoff parameter between error and margin)

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

The optimization problem

- The dual of the problem is

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **w** is recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now
- Once again, a QP solver can be used to find α_i

Extension to non-linear decision boundary

- Non-Linear Classifiers
  - One solution: Multiple layers of threshold linear function → multi-layer neural network

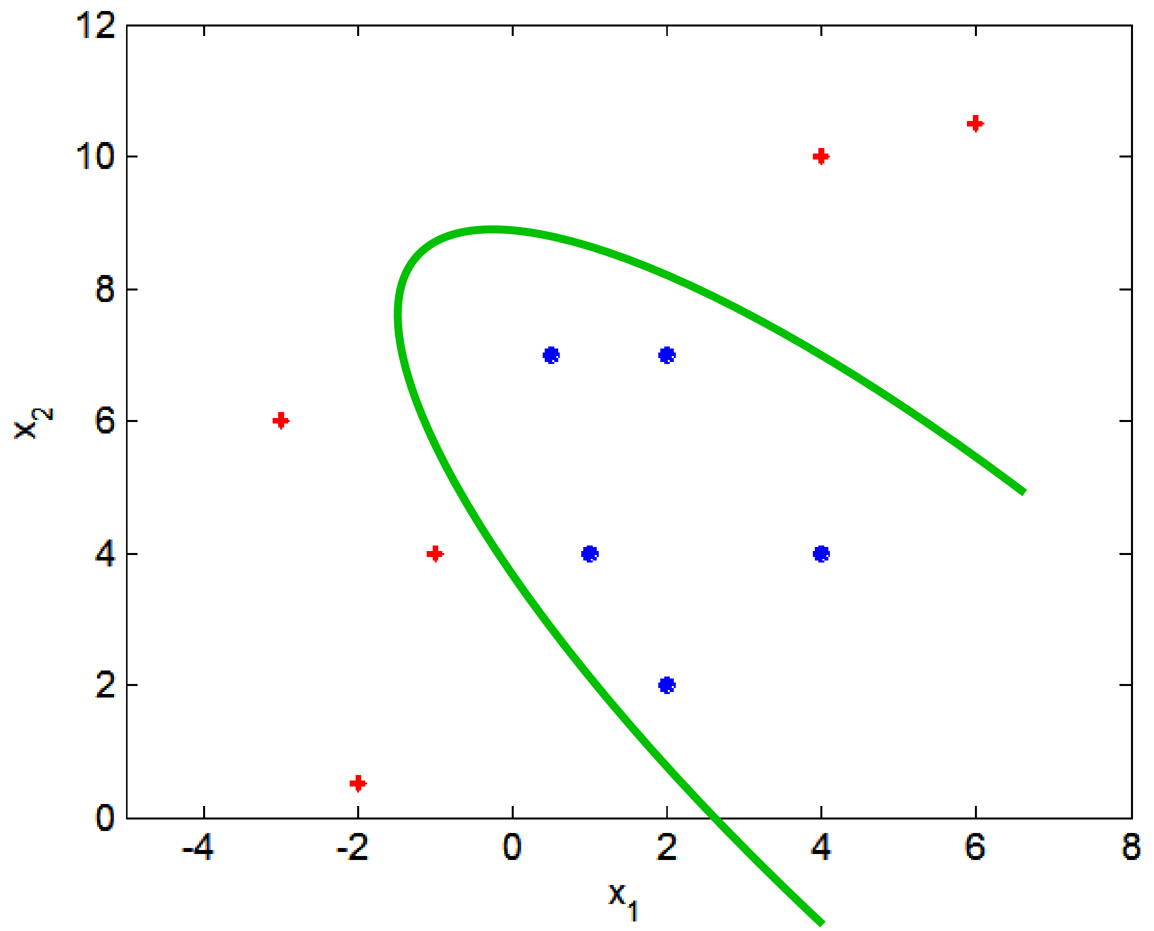
    (problems: local minima; many parameters; heuristics needed to train …etc)
  - Other solution: project the data into a high dimensional feature space to increase the computational power of the linear learning machine.

Introducing kernels

- So far, we only consider large-margin classifier with a linear decision boundary
  - How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point x_i are located
  - Feature space: the space of $f(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - Kernel function

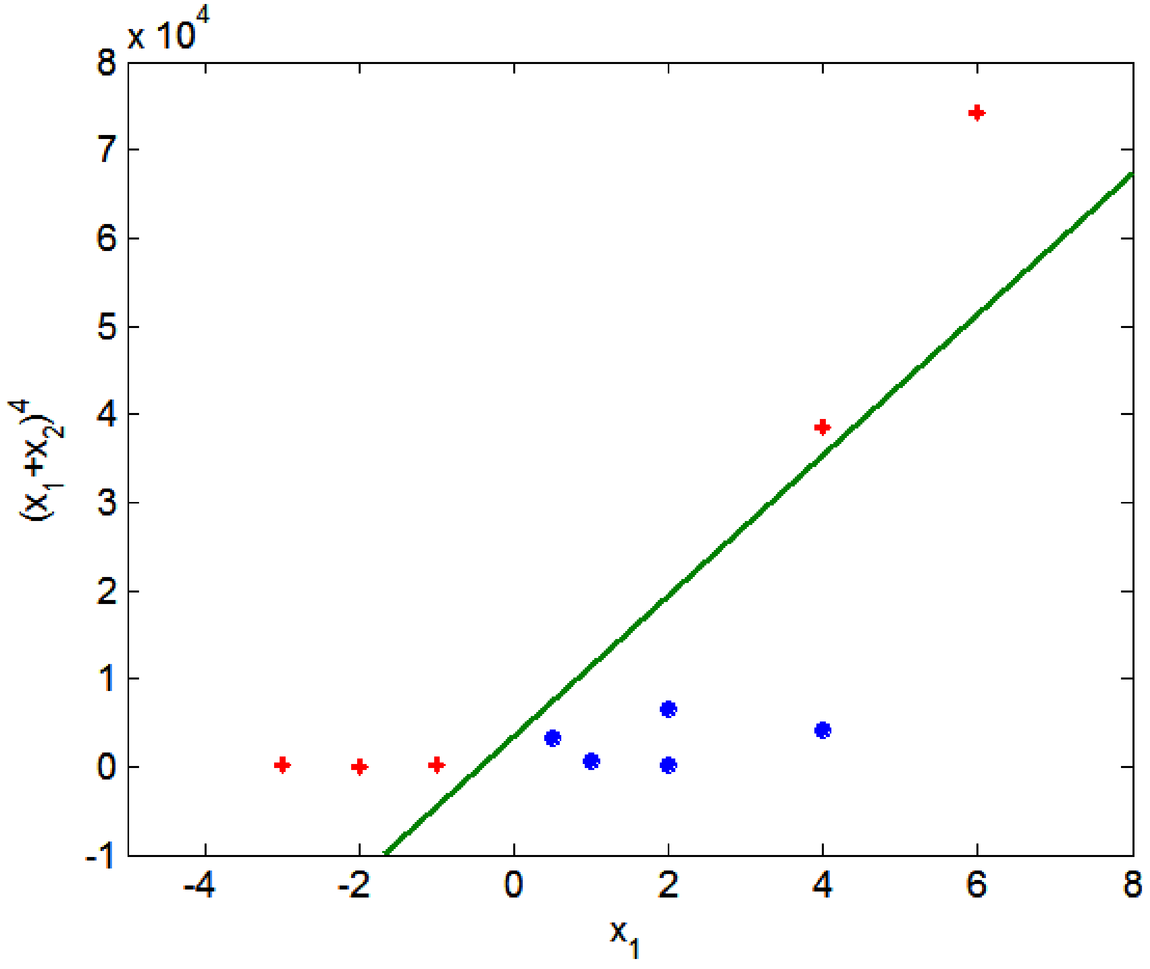
Nonlinear Support Vector Machines

- What if decision boundary is not linear?

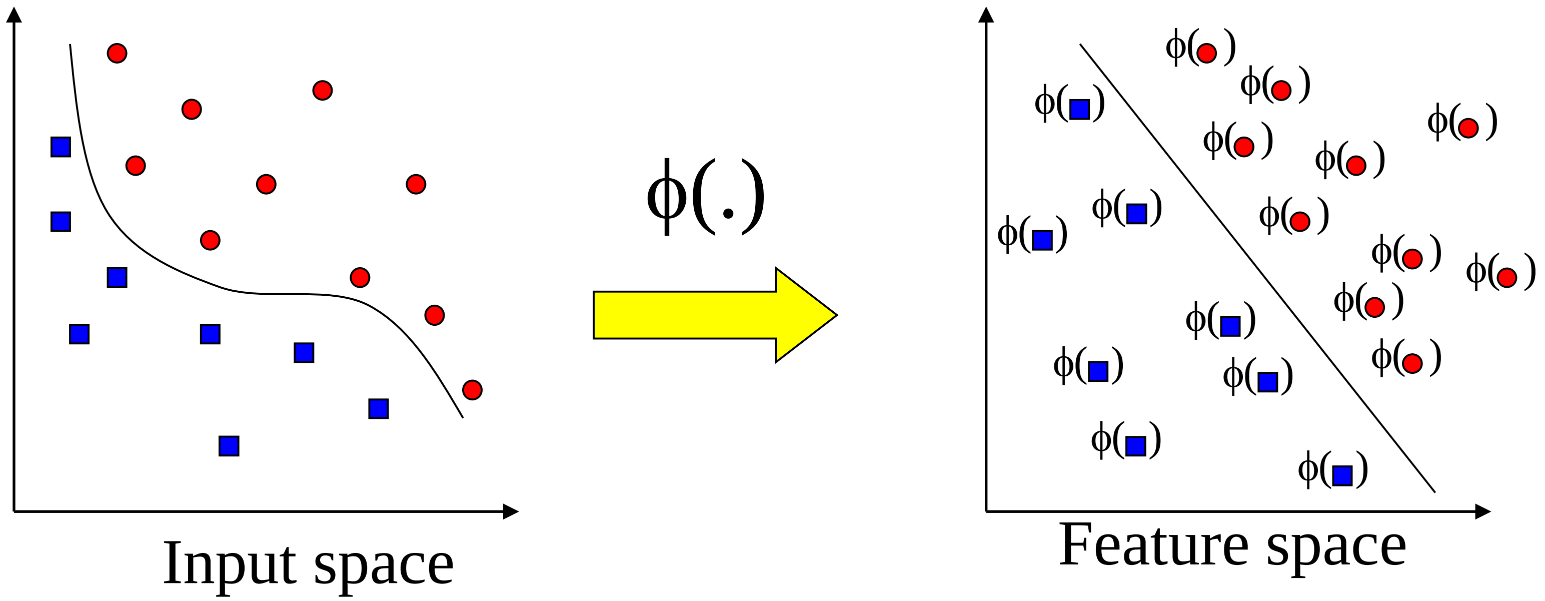


Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Transforming the data



The kernel trick

- Recall the SVM optimization problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Making kernels

- The mapping function must be symmetric,

$$K(x \bullet z) = \langle \phi(x) \bullet \phi(z) \rangle$$
$$= \langle \phi(z) \bullet \phi(x) \rangle = K(z \bullet x)$$

- and satisfy the inequalities that follow from the Cauchy-Schwarz inequality.

$$K(x,z)^2 = \langle \phi(x) \bullet \phi(z) \rangle^2 \leq \|\phi(x)\|^2 \|\phi(z)\|^2$$
$$= \langle \phi(x) \bullet \phi(z) \rangle \langle \phi(x) \bullet \phi(z) \rangle$$
$$= K(x,x) K(z,z)$$

An example for $\phi(\bullet)$ and $K(\bullet,\bullet)$

- Suppose $\phi(.)$ is given as follows

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- An inner product in the feature space is

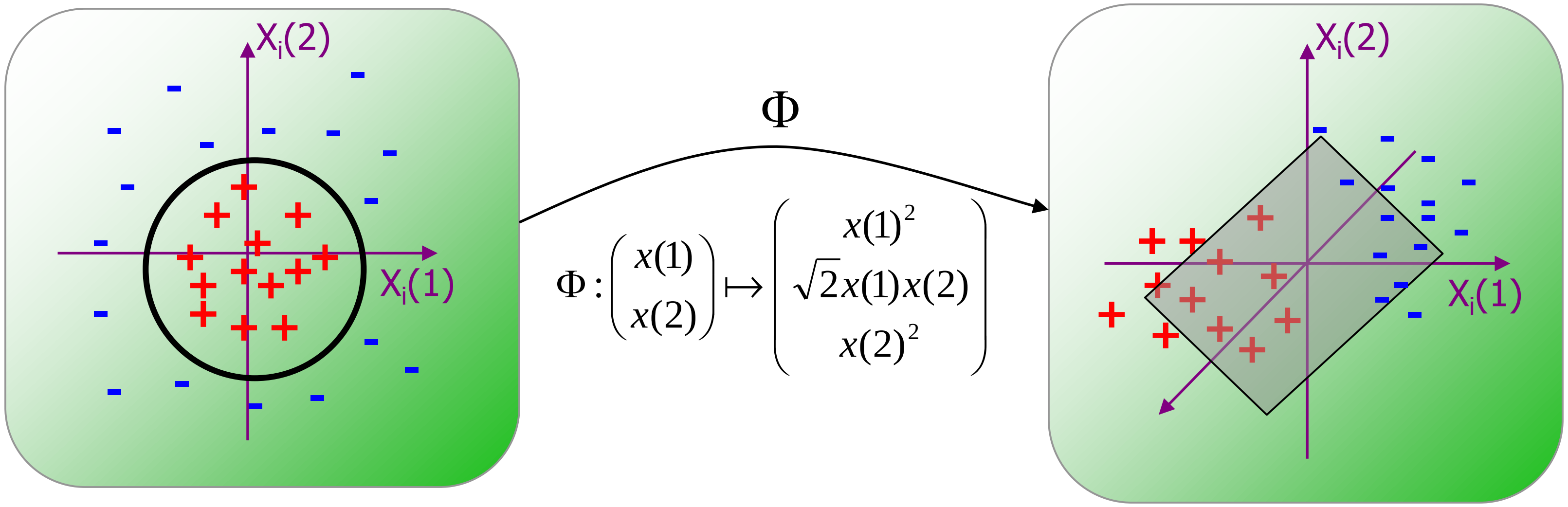
$$\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out $\phi(.)$ explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out $\phi(.)$ explicitly is known as the kernel trick

Support Vector Machines - Nonlinear case



$$\Phi : \begin{pmatrix} x(1) \\ x(2) \end{pmatrix} \mapsto \begin{pmatrix} x(1)^2 \\ \sqrt{2}x(1)x(2) \\ x(2)^2 \end{pmatrix}$$

- Nonlinear mapping $\Phi : F \rightarrow \Phi(F)$ into a higher dimension space
- $\mathrm{k}(\mathbf{x}, \mathbf{x}') := \left(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')\right)$

Examples of kernel functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x}-\mathbf{y}||^2/(2\sigma^2))$$

- Sigmoid with parameter κ and θ

$$K(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

Modification due to kernel function

- Change all inner products to kernel functions
- For training,

Original

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Modification due to kernel function

- For testing, the new data **z** is classified as class 1 if *f* ≥0, and as class 2 if *f* <0

Original

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

A Simple SVM Example

- Suppose we have 5 1D data points
  - $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, ..., 5) by

$$\text{max.} \quad \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

A Simple SVM Example

- By using a QP solver, we get
  - $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$\begin{aligned} & f(y) \\ &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on $y_i(\mathbf{w}^T\phi(z)+b)=1$ and all give b=9

$$\Rightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Choosing the kernel function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarize all the data
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
- Note that SVM with RBF kernel is closely related to RBF neural networks

Strength and weakness of SVM

- Strength
  - Training is relatively easy
    - Globally optimal solution, unlike neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weakness
  - Need to choose a “good” kernel function

Random Forest

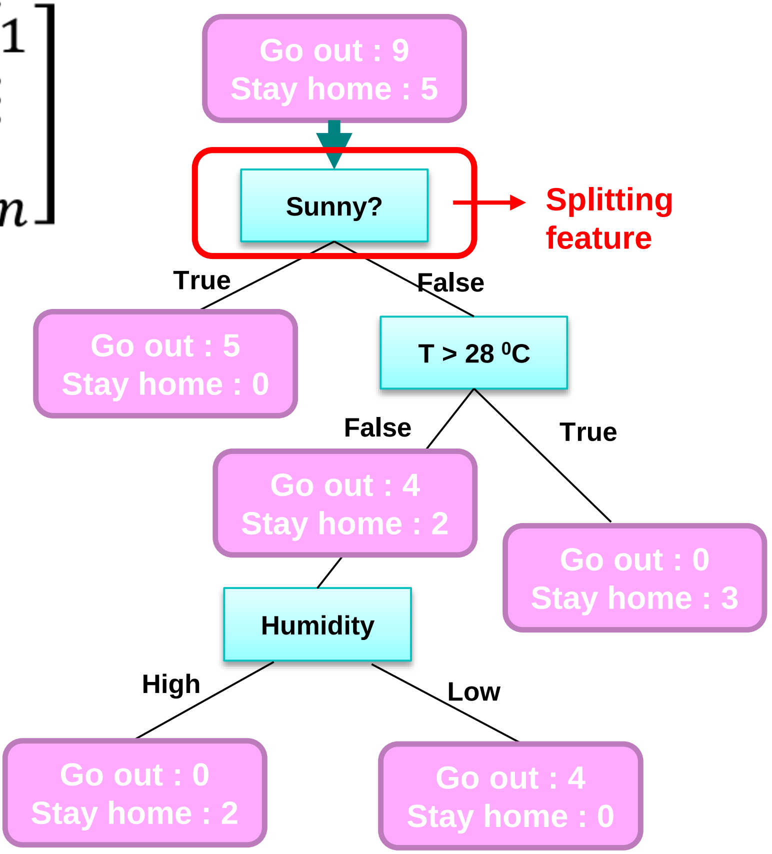
What? Why? and How?

- Each tree in the random forest is a classification tree.
- The objective is solving the classification problems.
- It improves the performance of traditional trees because it generalizes well.
- It employs the **ensemble results** (majority vote) of all random trees.

Decision tree (classification case)

- n data samples with features x and decisions y
- $$\begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \qquad \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

1	True	25	Low	Go out
2	True	22	Low	Go out
3	False	29	Low	Stay home
4	True	33	Low	Go out
5	False	26	High	Stay home
⋮				
14	False	19	Low	Go out



Feature selection

- How to decide which feature to be used in each stage(node), that will produce the best effect in classification result?
- There are to terms to **evaluate the performance of split node** in decision tree:
  - Shannon entropy
  - Information Gain

Shannon entropy & Information gain

- **Entropy**
- The expected value (average) of the information contained in each sample.
- Entropy thus characterizes our uncertainty about our source of information

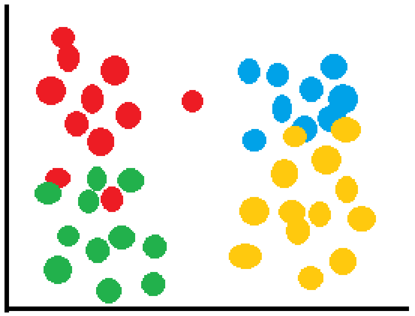
$$H(x) = -\sum_{i=1}^{n} p_i(x) log_2(p_i(x))$$

- **Information gain**
- The decreasing amounts of the entropy on the data source after we selected a certain feature for splitting data.

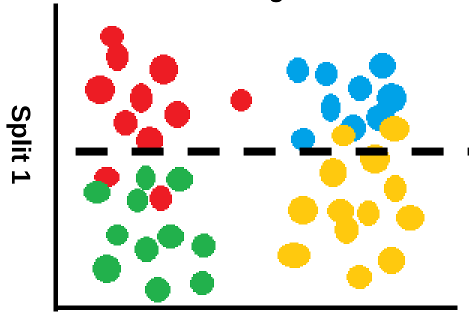
$$IG(S,a) = \mathrm{H}(S) - H(S|a)$$

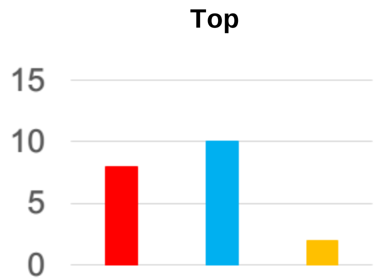
Shannon entropy & Information gain

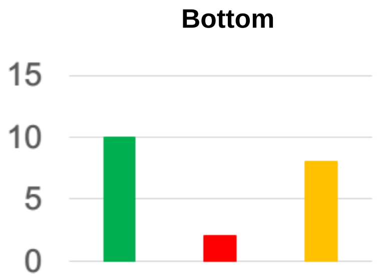
Data before splitting



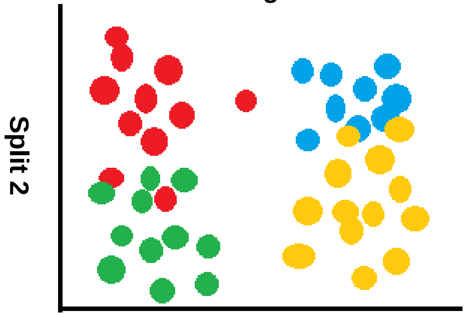
Information gain=0.4

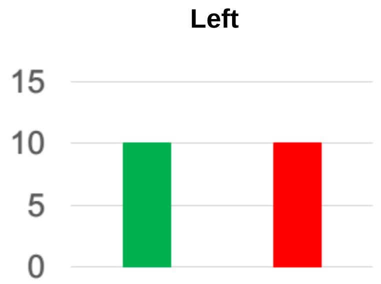


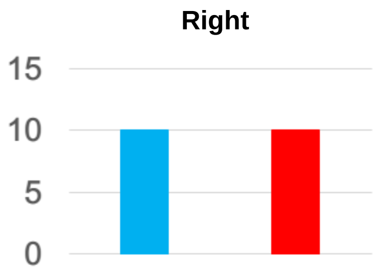




Information gain=0.69







Example

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

$$H(S) = H(9play, 5no\ play) = -\left(\frac{9}{14} * log_2\frac{9}{14} + \frac{5}{14} * log_2\frac{5}{14}\right) \approx 0.94$$

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	FALSE	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	TRUE	3	3		
rainy	3	2	cool	3	1								

$$H(Sunny) = H(2+, 3-) = -\left(\frac{2}{5} * log_2\frac{2}{5} + \frac{3}{5} * log_2\frac{3}{5}\right) \approx 0.971$$

$$H(Overcast) = H(4+, 0-) = -\left(\frac{4}{4} * log_2\frac{4}{4} + \frac{0}{4} * log_2\frac{0}{4}\right) = 0$$

$$H(Sunny) = H(3+, 2-) = -\left(\frac{3}{5} * log_2\frac{3}{5} + \frac{2}{5} * log_2\frac{2}{5}\right) \approx 0.971$$

$$IG(S, outlook) = H(S) - \left(\frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971\right) \approx \mathbf{0.247}$$

$IG(S, Humidity) \approx \mathbf{0.151}$ $IG(S, Windy) \approx \mathbf{0.048}$

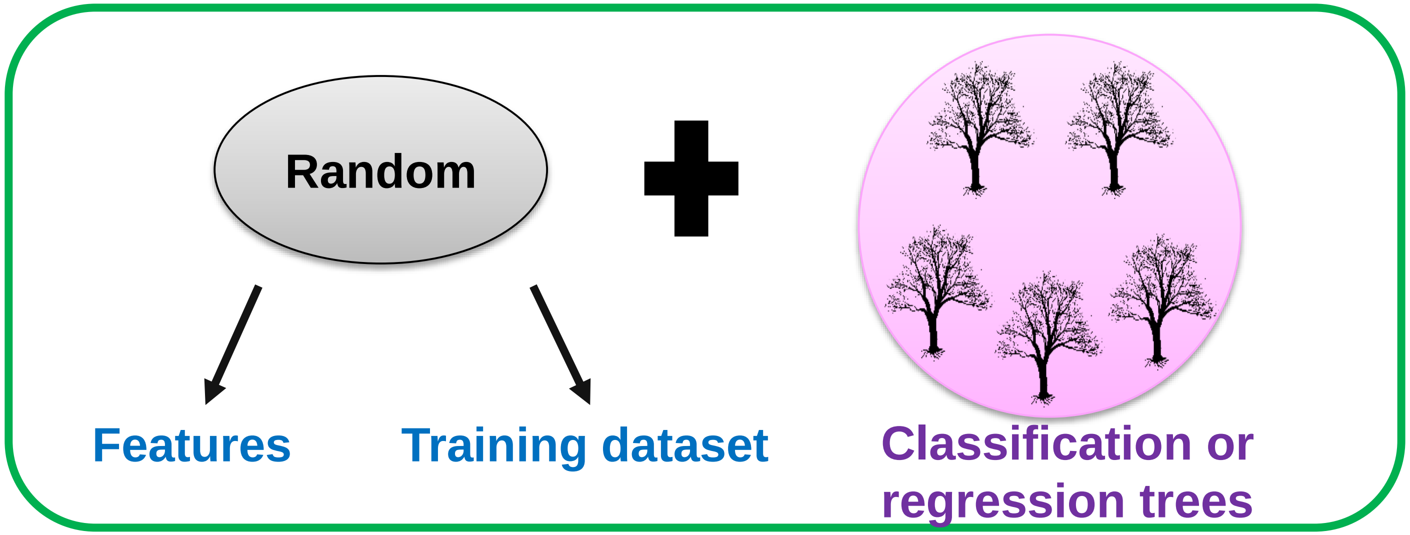
$IG(S, Temperature) \approx \mathbf{0.029}$

Classification & Regression Tree (CART)

- Use other function to decide the splitting features, called Impurity Function (IF), ex. Entropy, **Gini Index.**
  - $Gini(p) = 1 - \sum_k p_j^2$
- **Regression tree**
  - For solving regression problem (number, not class).
  - Use minimize square error to evaluate splitting performance.
  - Determine the final result by averaging the ensemble trees results.

Random forest

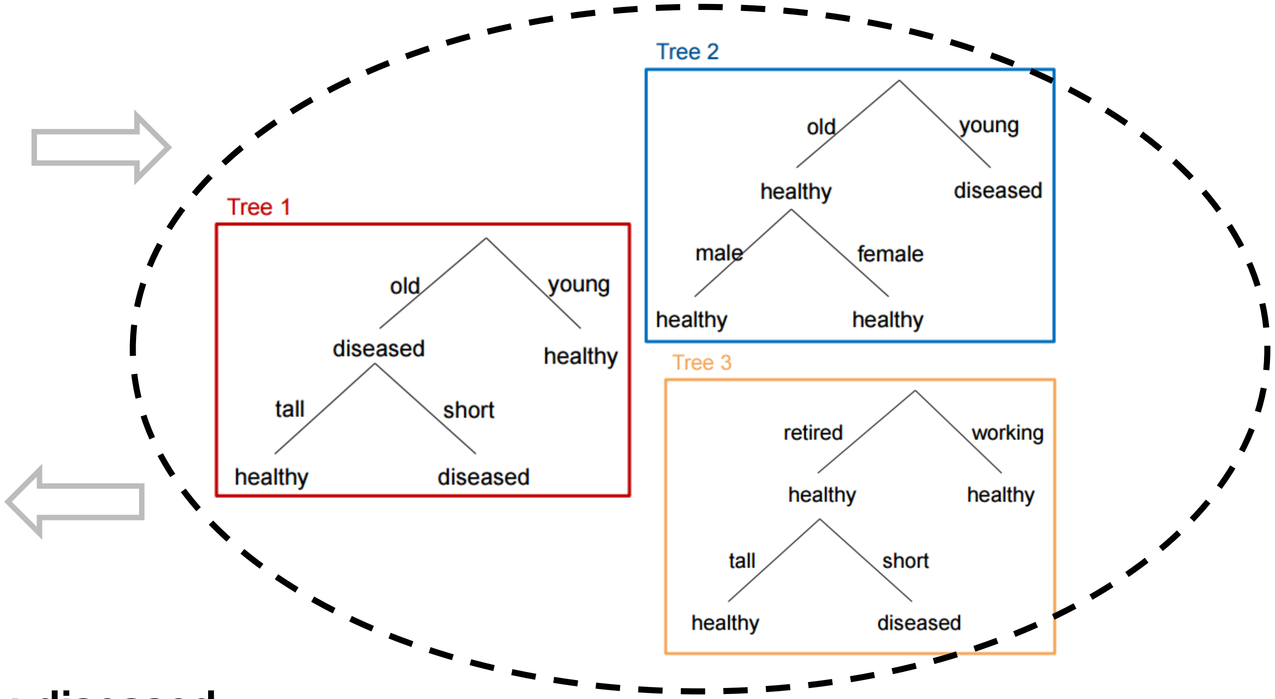
- Combine several classification trees or regression trees.
- Randomly choose sub-data samples on each tree.
- Randomly select features at each node.



Ensemble (classification)

- For a new data sample, predict the class result by voting.

New sample:
old, retired, male, short



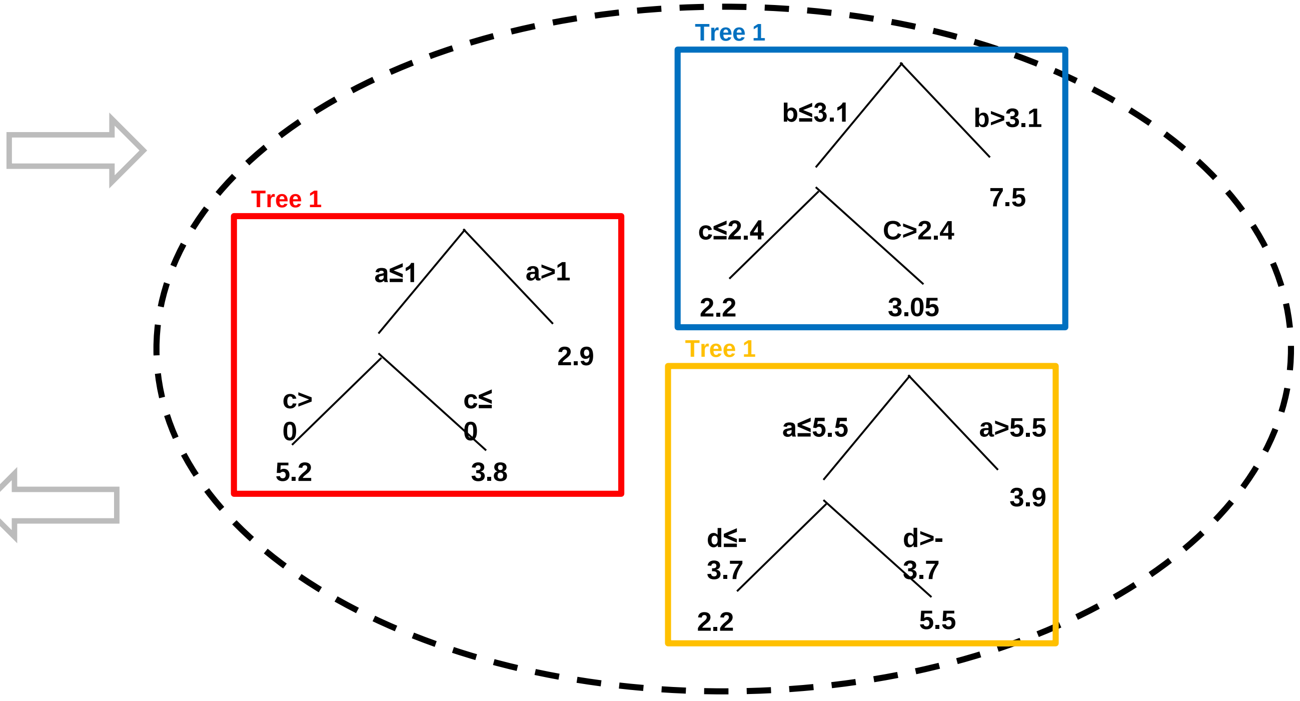
Prediction results:
diseased, healthy, diseased

→ **Majority rule: diseased**

Ensemble (regression)

- For a new data sample, predict the class result by averaging.

New sample:
1.5, 2.6, 5.5, -3.2

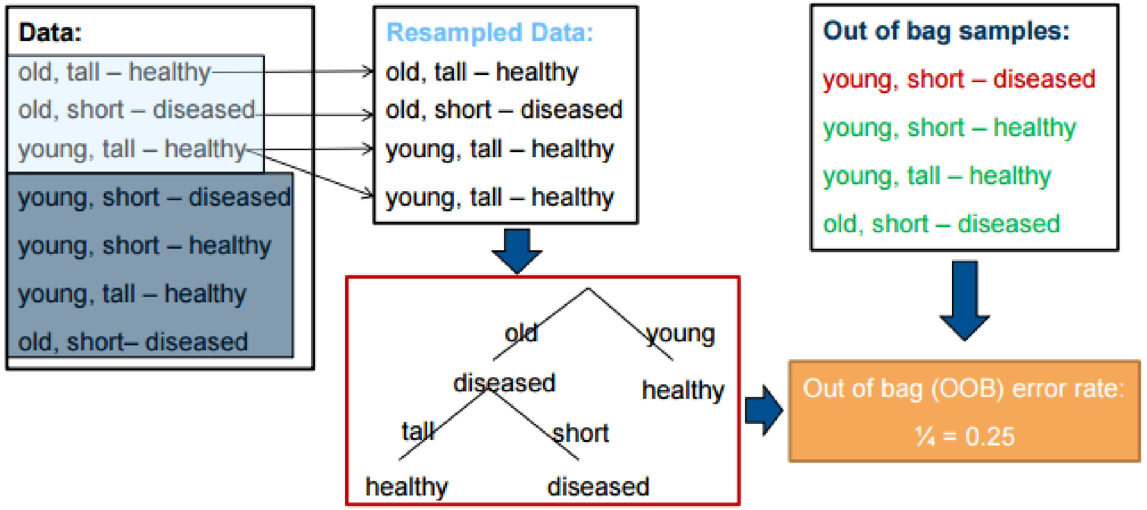


Prediction results:
2.9, 3.05, 5.5

→ average: 3.82

Out-of Bag (OOB)

- Out-of-bag estimates the generalization error and is similar to leave-one-out cross-validation.
- Sample the unused data subsets from the original data source as OOB samples.

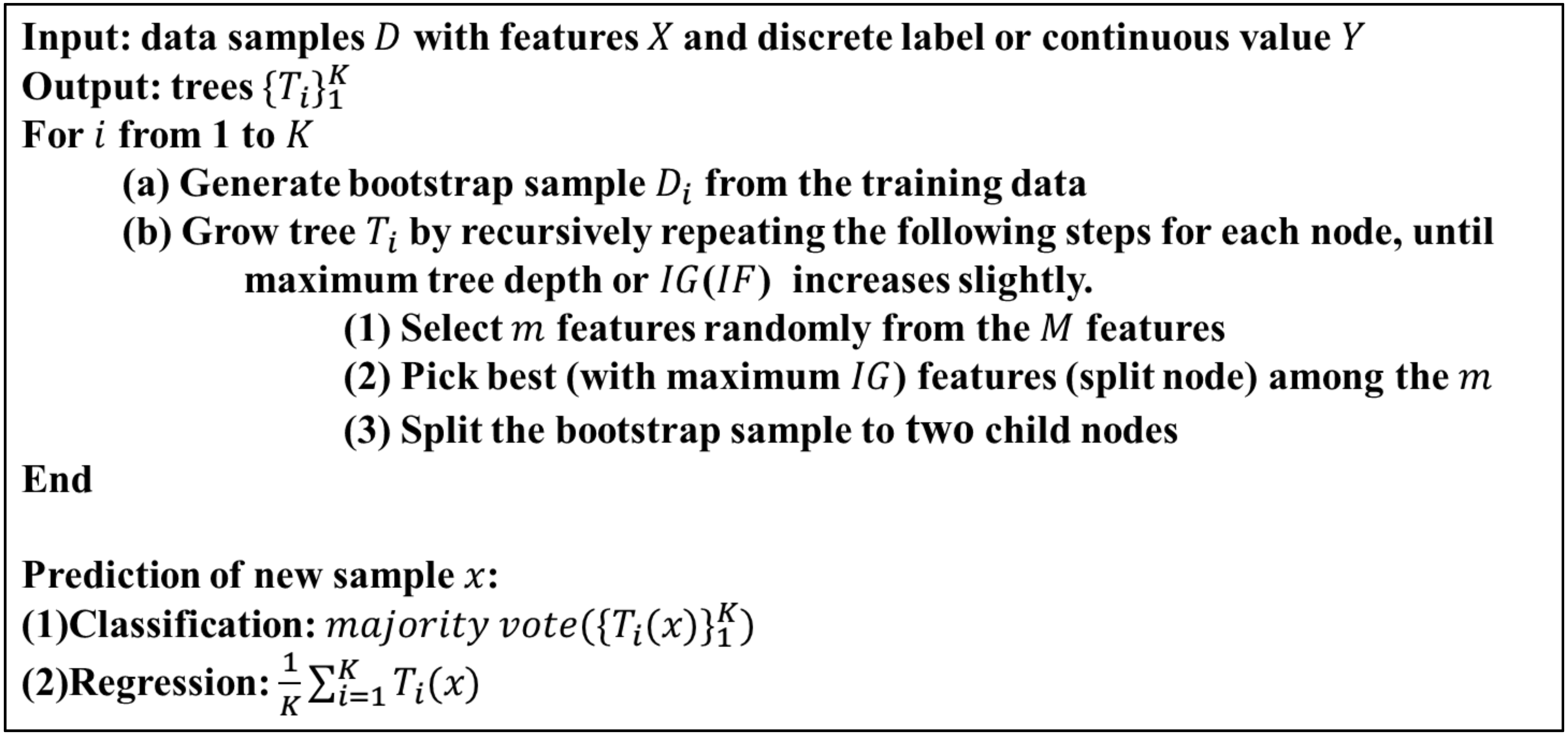


Determine parameters in advance

- There are some important parameters in RF need to be decided first:
  - Total number of trees K
  - Number of features to used in a node m
- Following are some advices about these two parameters:
  - 1. In general, the more number of trees, the better the performance is.

    But, it may cost time in testing stage.
  - 2. Typically, $m = \sqrt{M}$ or $log_2 M$, where M is total number of all features.
  - 3. OOB is a common way to evaluate the performances of these factors.

Random forest algorithm

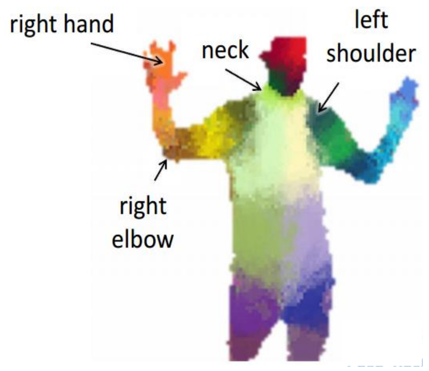


```
Input: data samples $D$ with features $X$ and discrete label or continuous value $Y$
Output: trees $\{T_i\}_1^K$
For $i$ from 1 to $K$
        (a) Generate bootstrap sample $D_i$ from the training data
        (b) Grow tree $T_i$ by recursively repeating the following steps for each node, until
                maximum tree depth or $IG(IF)$ increases slightly.
                        (1) Select $m$ features randomly from the $M$ features
                        (2) Pick best (with maximum $IG$) features (split node) among the $m$
                        (3) Split the bootstrap sample to two child nodes
End

Prediction of new sample $x$:
(1)Classification: $majority\ vote(\{T_i(x)\}_1^K)$
(2)Regression: $\frac{1}{K}\sum_{i=1}^{K} T_i(x)$
```

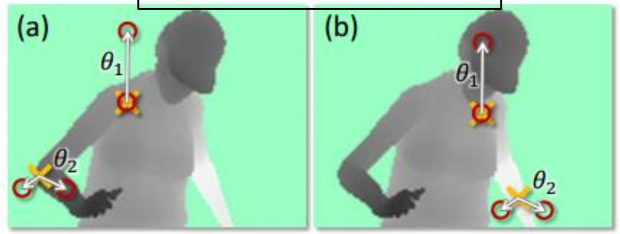
Applications in computer vision (1)

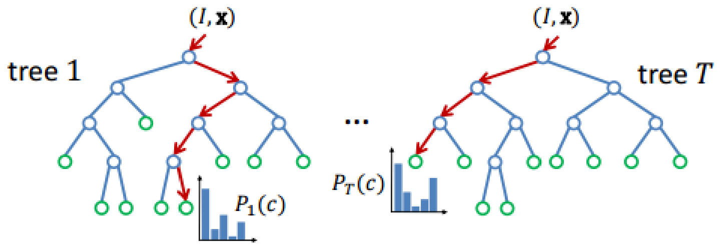
- Random forest as the classifier or regression estimator has been widely used in the computer vision applications.
- Following are some examples :
- **Body part classification** [1]



- **Body part labels shown in different colors.**

Features for splitting





- **Data samples to train the trees**

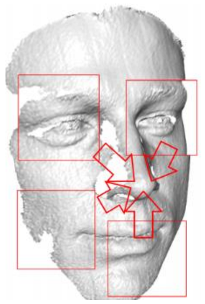
- **θ_1 in (a),(b) have different depth value**
- **θ_2 in (a) give a large depth difference response.**

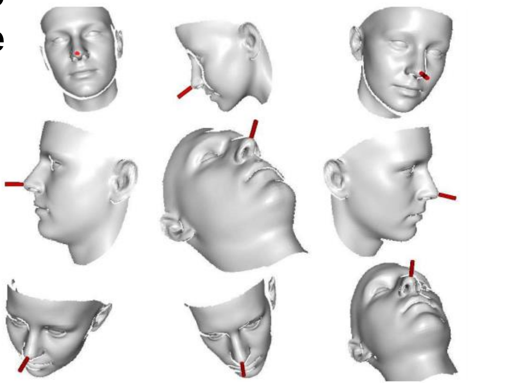
[1] J. Shotton et al. Real-Time Human Pose Recognition in Parts from a Single Depth Image. CVPR 2011

Applications in computer vision (2)

- **Head pose estimation** (Regression) [2]

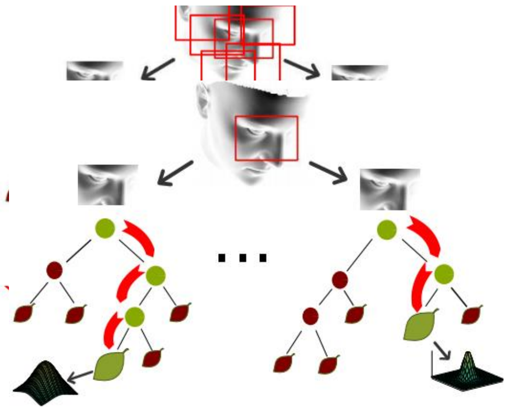
- **Different patches were used to estimate the nose location**



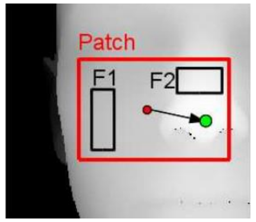


- **Synthetic 3D face models with different angles of $\theta_x, \theta_y, \theta_z, \theta_{yaw}, \theta_{pitch}, \theta_{roll}$**

Leaf node: real-valued of multivariate distribution of the output parameters



Ensemble result: probabilistic prediction in the output space



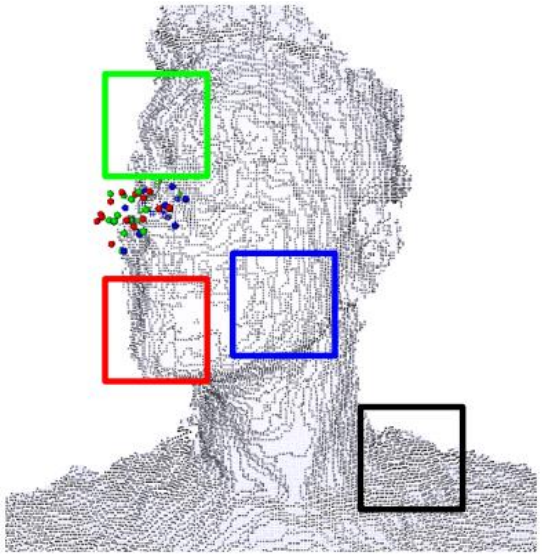
- **Red point: center of the patch**
- **Green point: groundtruth of the nose location**

- **F1 and F2 represent a possible choice for the regions over which to compute a binary test, i.e., the difference between the average values computed over the two boxes**

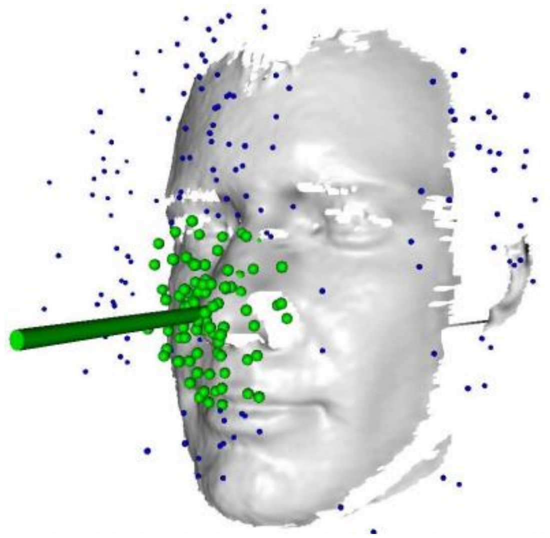
[2] G. Fanelli et al. Random Forests for Real Time 3D Face Analysis. IJCV 2013

Applications in computer vision (3)

- **Head pose estimation** (Regression) (Cont.)



- **Different patches generate different votes for the estimated nose location.**



- **Filtering outliers by mean shift; and the remains (bigger sphere) determine the final result.**

Summary

- K-NN classifier
- Bayesian classifier
- SVM classifier
- Native Bayes Classifier
- Random Forest