

# DSA5101

## Lecture 3 Alternative Classification

Li Xiaoli

National University of Singapore

# Outline

- 1. Instance-Based Classifier: KNN Classifier
- 2. Support Vector Machines
- 3. Random Forests
- 4. Imbalanced Learning
- 5. Semi-Supervised learning



Due to time constrain, we will NOT cover Neural Network, Bayesian Classifier etc

# 1. Instance-Based Classifiers

- Store the training records (without training explicit models) – no induction step
- Use training records directly to predict the class label of unseen cases/examples: deduction step needs to find training neighbors of unseen cases.

Set of Stored Cases

Atr1	.....	AtrN	Class
1		8	A
1		10	B
7		4	B
9		2	C
2		8	A
8		1	C
6		5	B

Training data

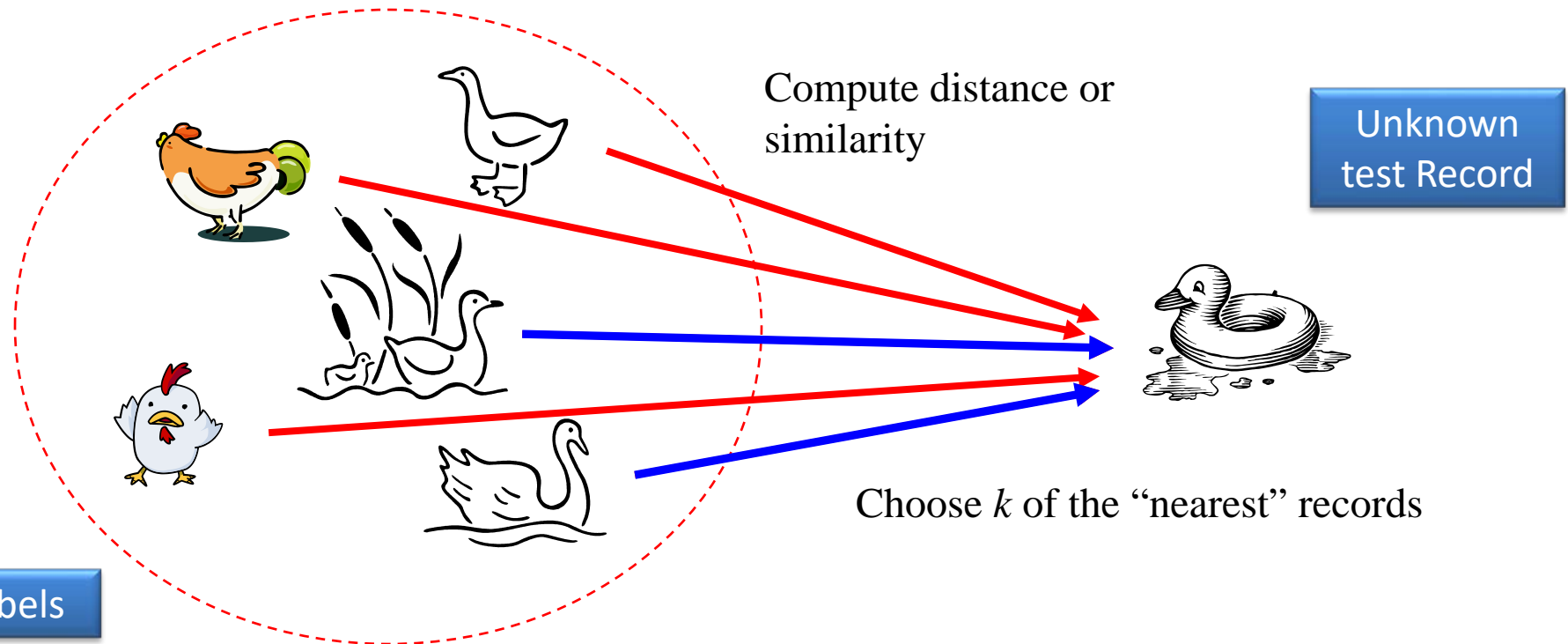
Unseen Case

Atr1	.....	AtrN
2		9

*k*-Nearest Neighbors (*k*-NN): uses *k* “closest” points (nearest neighbors) for performing classification

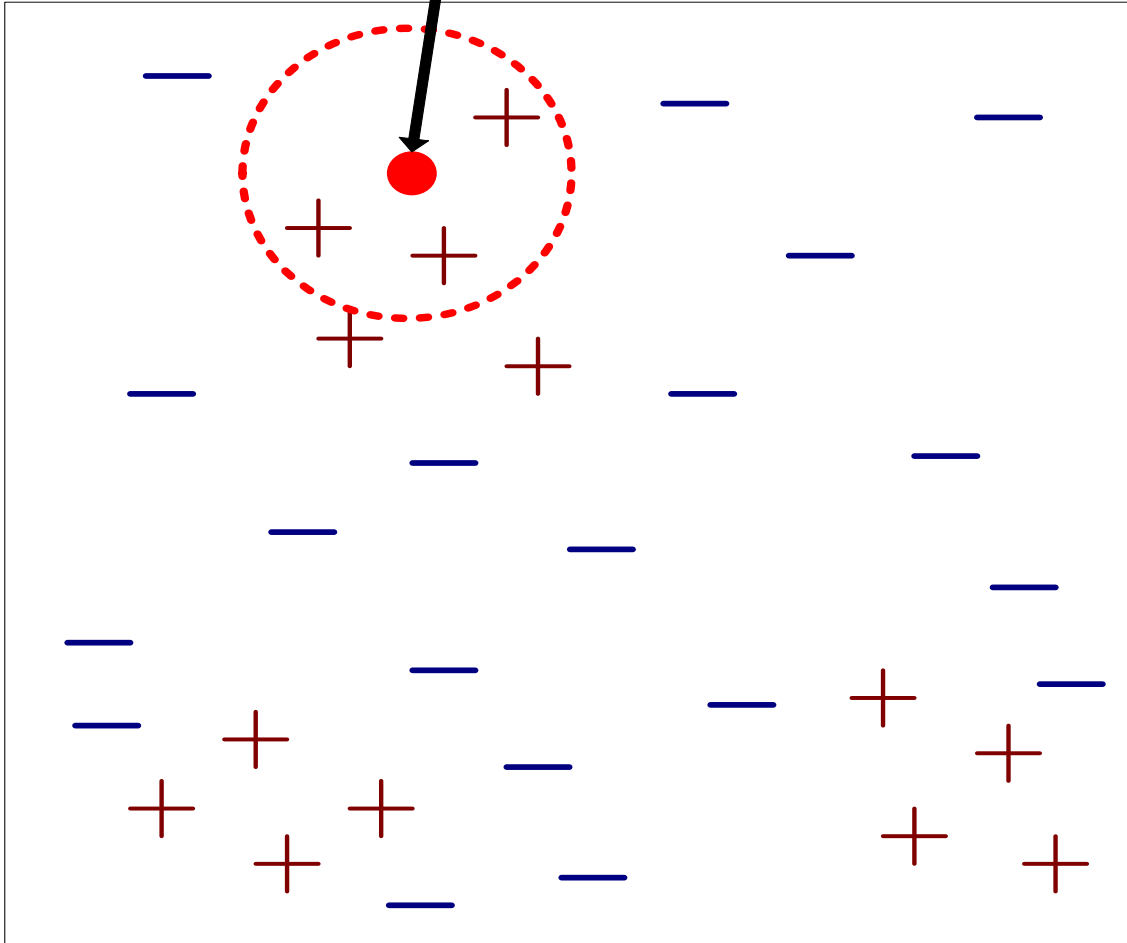
# Nearest Neighbor Classifiers

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



# Nearest-Neighbor Classifiers

Unknown record



## □ Requires three things

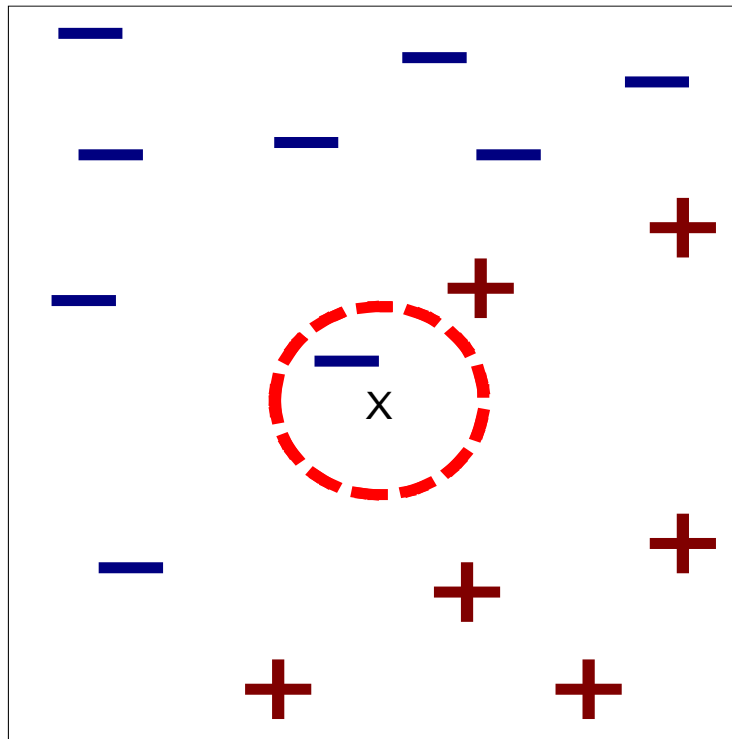
- The set of stored training records
- Distance metric to compute distance between records
- The value of  $k$ , the number of nearest neighbors to retrieve

## □ To classify an unknown record

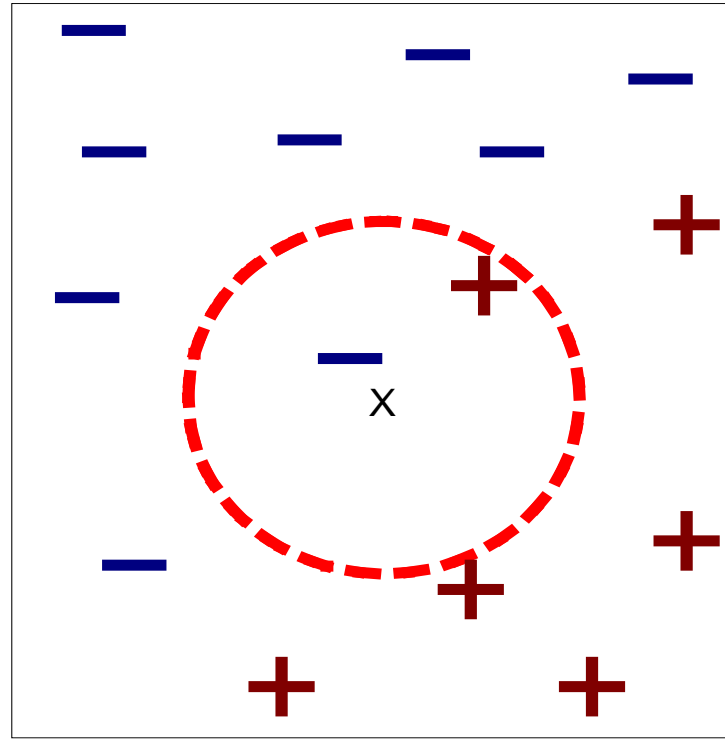
- Compute distance to other training records
- Identify  $k$  nearest neighbors
- Use class labels of the nearest neighbors to determine the class label of the unknown record (e.g., by taking majority vote)

# Definition of Nearest Neighbor

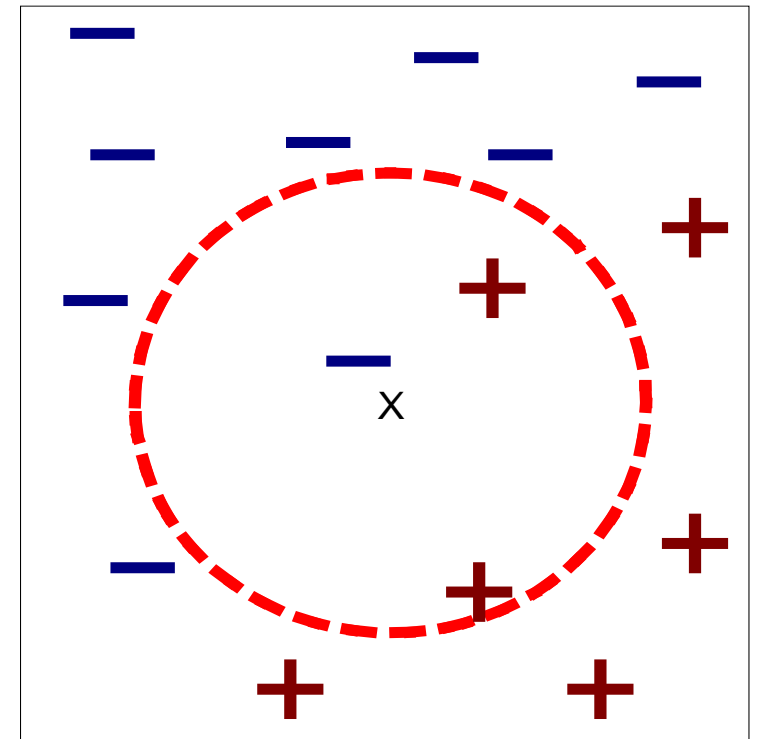
**$K$ -nearest neighbors** of a record  $x$  are data points that have the  $k$  smallest distance to  $x$



(a) 1-nearest neighbor



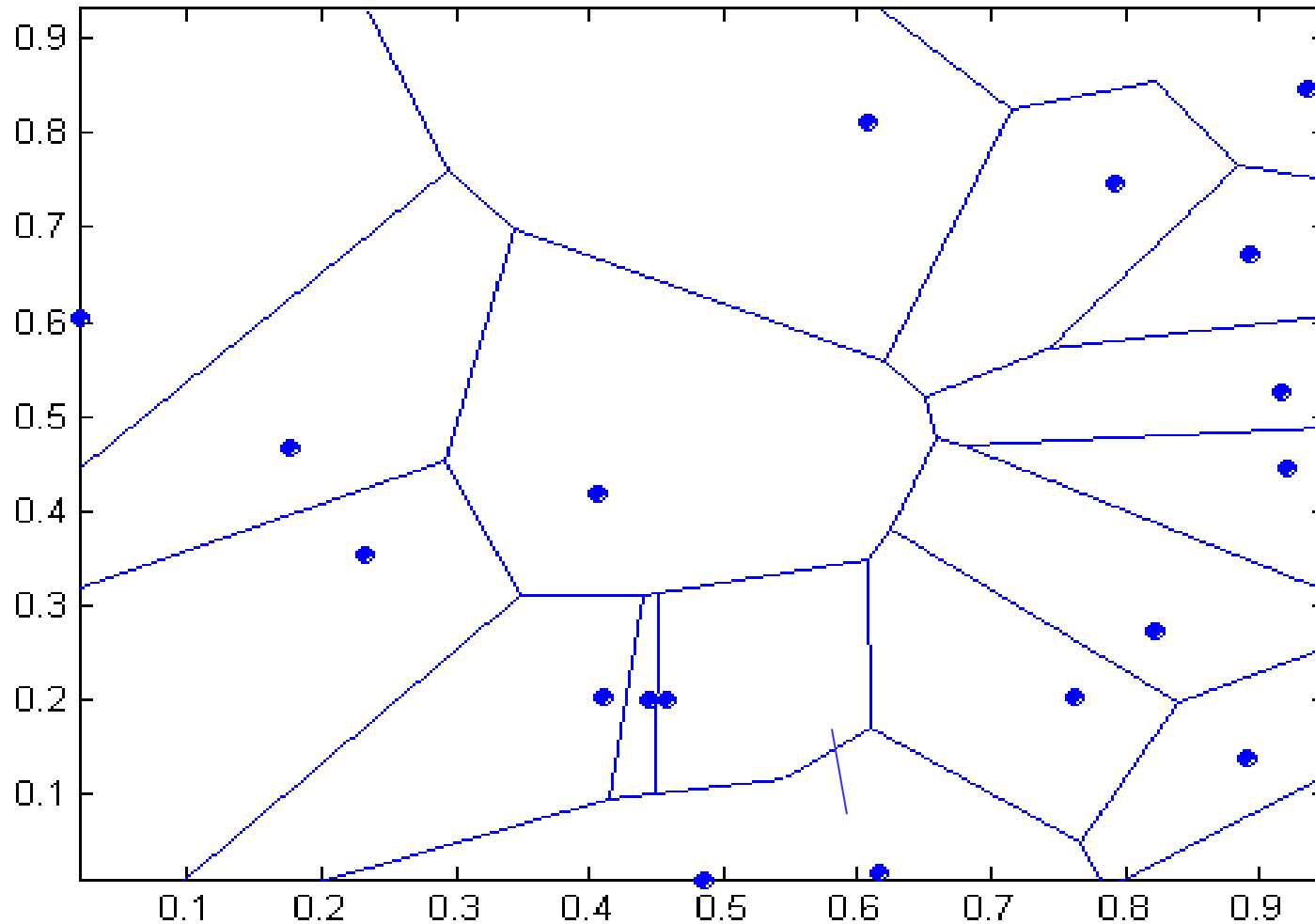
(b) 2-nearest neighbor



(c) 3-nearest neighbor

# 1 nearest-neighbor

- Voronoi Diagram (k-NN with  $k=1$ )
  - Boundaries denote 1-neighborhoods (from training set)



# Nearest Neighbor Classification

- Compute distance between two points  $p$  &  $q$ :
  - Euclidean distance:

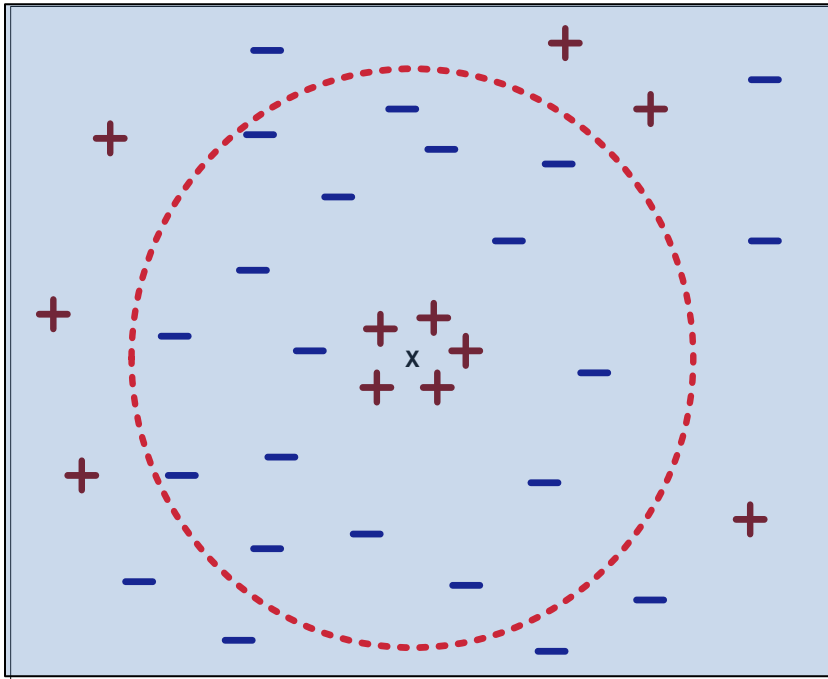
$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the  $k$ -nearest neighbors (**odd** vs even **number**)
  - Weigh the vote according to distance
    - weight factor,  $w = 1/d$  or  $1/d^2$



# Nearest Neighbor Classification (cont')

- Choosing the value of  $k$ :
  - If  $k$  is too small, sensitive to noise points (e.g.  $k=1, 2, 3$ )
  - If  $k$  is too large, neighborhood may include points from other classes



What if  $k=n$ , i.e. the number of all the data points?

Then it becomes majority class in the training data

If  $k$  is too large, the prediction will be depended on the data points that are not really related to my current data point

How to decide the value of  $k$ ? cross validation or separate validation set

# Nearest Neighbor Classification (cont')

- Pros of  $k$ NN
  - Easy to implement
  - Incremental addition of training data is trivial
- Cons of  $k$ NN
  - $k$ -NN classifiers are **lazy** learners, which do not build models **explicitly**. This can be more **expensive** than eager learners (such as decision tree) when **classifying** a test/unknown record.
  - Unlike decision tree that attempts to find a **global** model that fits the entire input space, nearest neighbor classifiers make the prediction based on **local information**, which can be more susceptible to noise.

# Euclidean Distance

- Euclidean Distance between two  $n$ -dimensional examples  $\mathbf{p}$  and  $\mathbf{q}$   
 $\mathbf{p}=\{p_1, p_2, \dots, p_k, \dots, p_n\}$ ,  $\mathbf{q}=\{q_1, q_2, \dots, q_k, \dots, q_n\}$ .

$$\mathit{dist} = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}$$

- where  $n$  is the number of dimensions (attributes), and  $p_k$  and  $q_k$  are the  $k^{\text{th}}$  attributes of data objects  $p$  and  $q$ , respectively.
- Feature normalization is usually necessary if scales are different.

# Normalization - Motivation

- Attributes may have to be scaled or normalized to prevent distance measures from being dominated by one of the attributes
- Example:
  - F1: height of a person may vary from 1.2m to 2.4m
  - F2: weight of a person may vary from 35kg to 442kg
  - **F3: Annual income of a person may vary from 10K to 50,000K**

$$p = (p_1 p_2 p_3) = (1.64, 48, 6000, A)$$

$$q = (q_1 q_2 q_3) = (1.82, 75, 10000, B)$$

**F3 dominates the calculation of Euclidean distance**

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2} = \sqrt{(1.65 - 1.82)^2 + (48 - 75)^2 + (6000 - 10000)^2}$$

# Normalization for each attribute

- Min-max normalization:

–  $[min_A, max_A] \dashrightarrow [new\_min_A, new\_max_A]$

$$v' = \frac{v - min_A}{max_A - min_A} (new\_max_A - new\_min_A) + new\_min_A$$

– Example of **Annual income**:

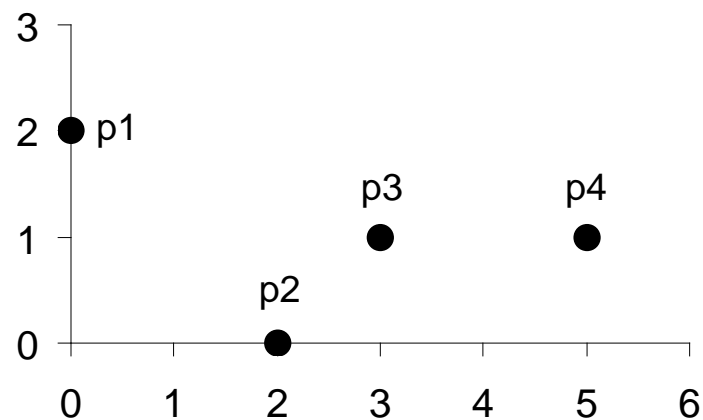
Annual income range [12,000, 300,000] normalized to [0.0, 1.0]. Then 73,000 is mapped to

$$\frac{73,000 - 12,000}{300,000 - 12,000} (1.0 - 0) + 0 = 0.21$$
$$\frac{12,000 - 12,000}{300,000 - 12,000} (1.0 - 0) + 0 = 0 \quad \frac{300,000 - 12,000}{300,000 - 12,000} (1.0 - 0) + 0 = 1$$

# Euclidean Distance in 2D

- Example:

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1



	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Euclidean Distance Matrix

# Minkowski Distance

- ▶ Minkowski Distance is a generalization of Euclidean Distance

$$\textit{dist} = \left( \sum_{k=1}^n |p_k - q_k|^r \right)^{\frac{1}{r}}$$

Where  $r$  is a parameter,  $n$  is the number of dimensions (attributes) and  $p_k$  and  $q_k$  are the  $k$ -th attributes (components) of data examples  $p$  and  $q$  respectively.

# Minkowski Distance: Special Cases

Minkowski Distance is a generalization of Euclidean Distance

$$\text{dist} = \left( \sum_{k=1}^n |p_k - q_k|^r \right)^{\frac{1}{r}}$$

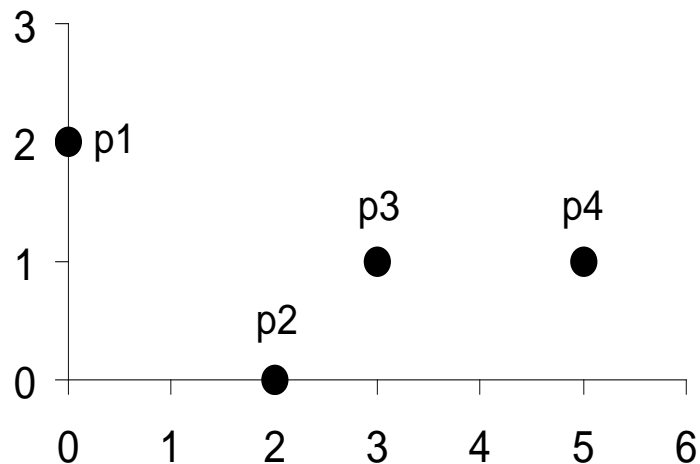
- $r = 1$ :

City block (Manhattan, taxicab,  **$L_1$  norm**) distance.

- A common example of this is the **Hamming distance**, which is just the number of bits that are different between two binary vectors (**Hamming distance** is only applied to binary vectors)

- $r = 2$ :

Euclidean distance ( **$L_2$  norm**)



point	x	y
p1	0	2
p2	2	0

**$L_1$  norm:  $\text{dist}(p1, p2) = |0-2| + |2-0| = 4$**

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

**$L_2$  norm:**



# Minkowski Distance: Special Cases

$$\mathbf{dist} = \left( \sum_{k=1}^n |p_k - q_k|^r \right)^{\frac{1}{r}}$$

- $r = 1$ :  
City block (Manhattan, taxicab, **L<sub>1</sub> norm**) distance.
- $r = 2$ :  
Euclidean distance (**L<sub>2</sub> norm**)
- $r \rightarrow \infty$ :  
“supremum” (**L<sub>max</sub> norm**, L<sub>∞</sub> norm) distance.
  - The **maximum difference** between any component of the two vectors:

$$\max(|p_1 - q_1|, \dots, |p_n - q_n|)$$

Do not confuse parameter  $r$  with dimensionality  $n$ , i.e., all these distances are defined for all the dimensions.

# Minkowski Distance

Distance Matrix

City block

Euclidean

Supremum

$$dist = \left( \sum_{k=1}^n |p_k - q_k|^r \right)^{\frac{1}{r}}$$

point	x	y
<b>p1</b>	<b>0</b>	<b>2</b>
p2	2	0
<b>p3</b>	<b>3</b>	<b>1</b>
p4	5	1

## An Example

Distance between P1 and P3

- $r=1$ ,  $L_1$  norm, City block distance

$$|0-3| + |2-1| = 4$$

- $r=2$ ,  $L_2$  norm, Euclidean distance

$$\sqrt{(0-3)^2 + (2-1)^2} = \sqrt{10} = 3.162$$

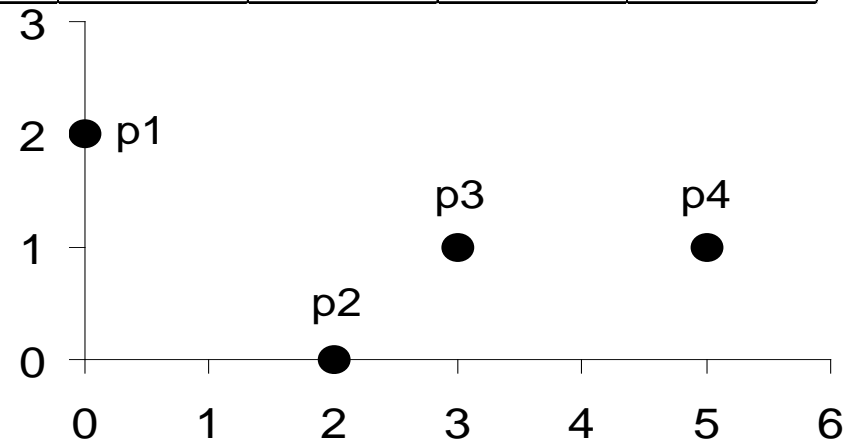
- $r \rightarrow \infty$ ,  $L_\infty$  norm, supremum distance

$$\text{Max}(|0-3|, |2-1|) = \text{Max}(3, 1) = 3$$

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

$L_\infty$	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0



# Cosine Similarity

- If  $d_1$  and  $d_2$  are two examples/vectors (e.g. document vectors), then

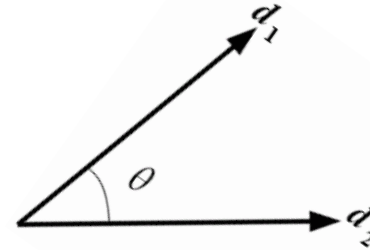
$$\cos( d_1, d_2 ) = (d_1 \bullet d_2) / ||d_1|| ||d_2||$$

where  $\bullet$  indicates vector dot product and  $||d||$  is the length of vector  $d$ .

- It is a measure of the *cosine* of the angle between the two vectors.
- Example:

$$d_1 = \mathbf{3\ 2\ 0\ 5\ 0\ 0\ 0\ 2\ 0\ 0}$$

$$d_2 = \mathbf{1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 2}$$




$$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$$

$$||d_1|| = (3*3 + 2*2 + 0*0 + 5*5 + 0*0 + 0*0 + 0*0 + 2*2 + 0*0 + 0*0)^{0.5} = (42)^{0.5} = 6.4807$$

$$||d_2|| = (1*1 + 0*0 + 0*0 + 0*0 + 0*0 + 0*0 + 0*0 + 1*1 + 0*0 + 2*2)^{0.5} = (6)^{0.5} = 2.4495$$

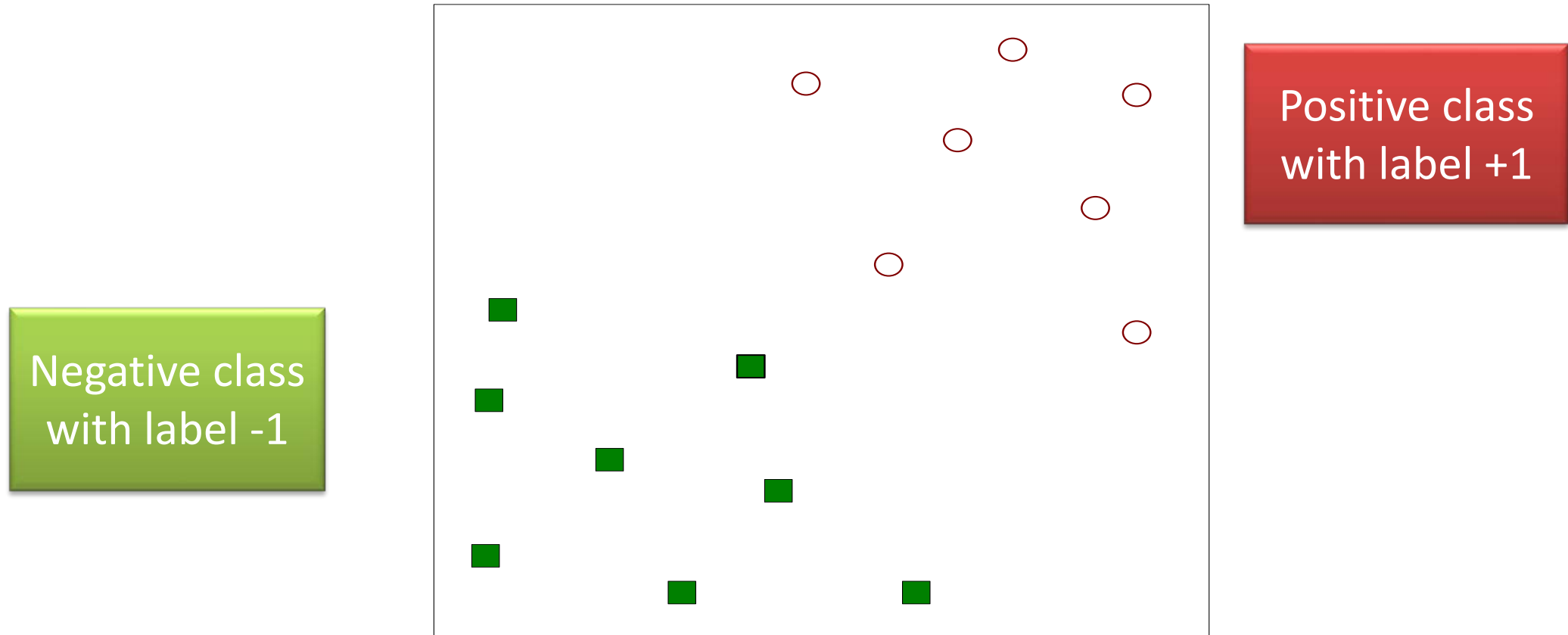
$$\cos( d_1, d_2 ) = 5 / (6.4807 * 2.4495) = 0.3150$$

# Outline

- 1. Instance-Based Classifiers: KNN Classifier
- 2. Support Vector Machines 
- 3. Random Forests
- 4. Imbalanced Learning
- 5. Semi-Supervised learning

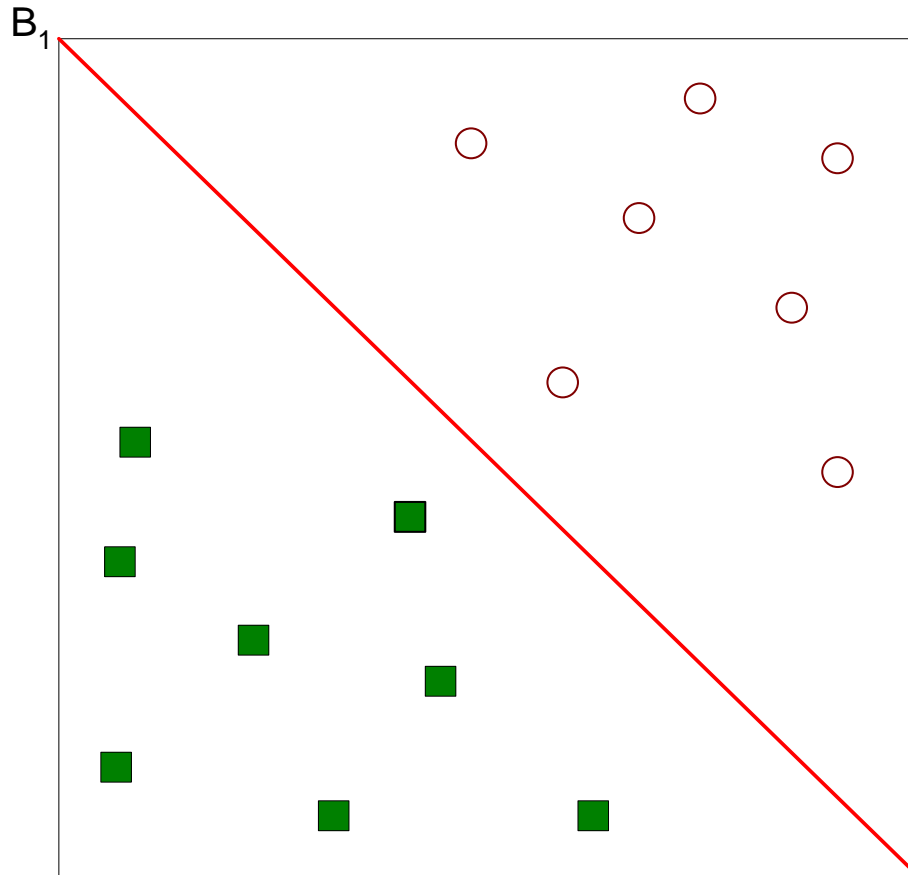
# Brief Introduction of Support Vector Machines

Handle binary classification only.



- Each training example is a feature vector with label, either +1 or -1
- Find a linear hyperplane (decision boundary) that will separate the data from two different classes

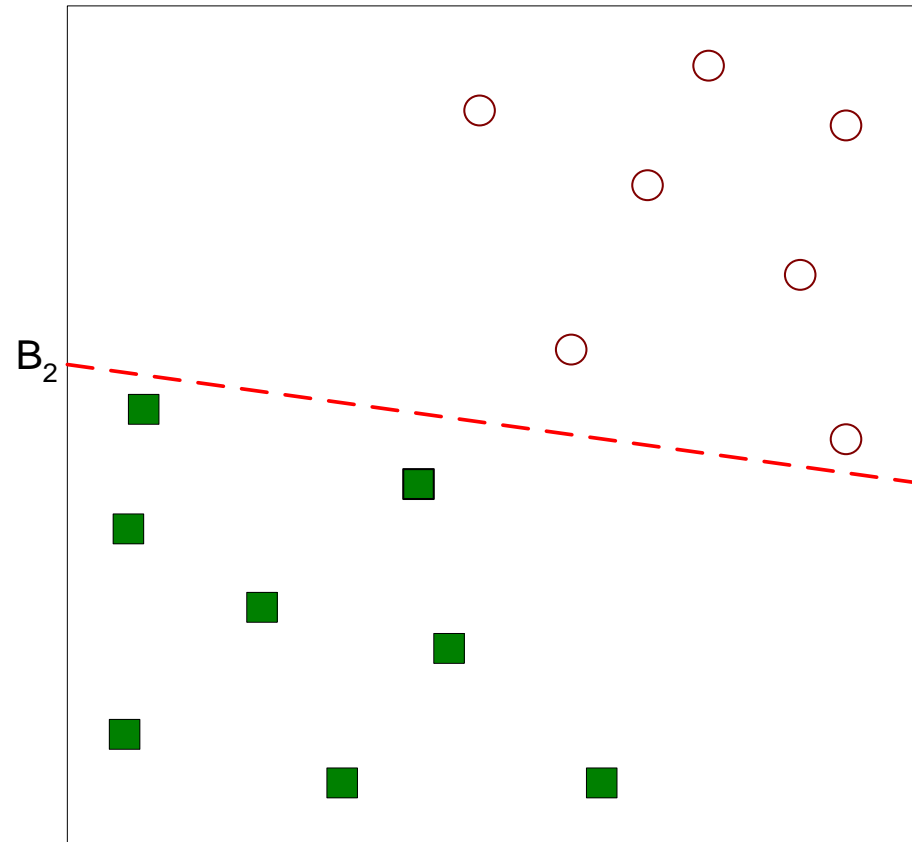
# Support Vector Machines



A hyperplane (or a line in 2d) separates the given training data from two different classes

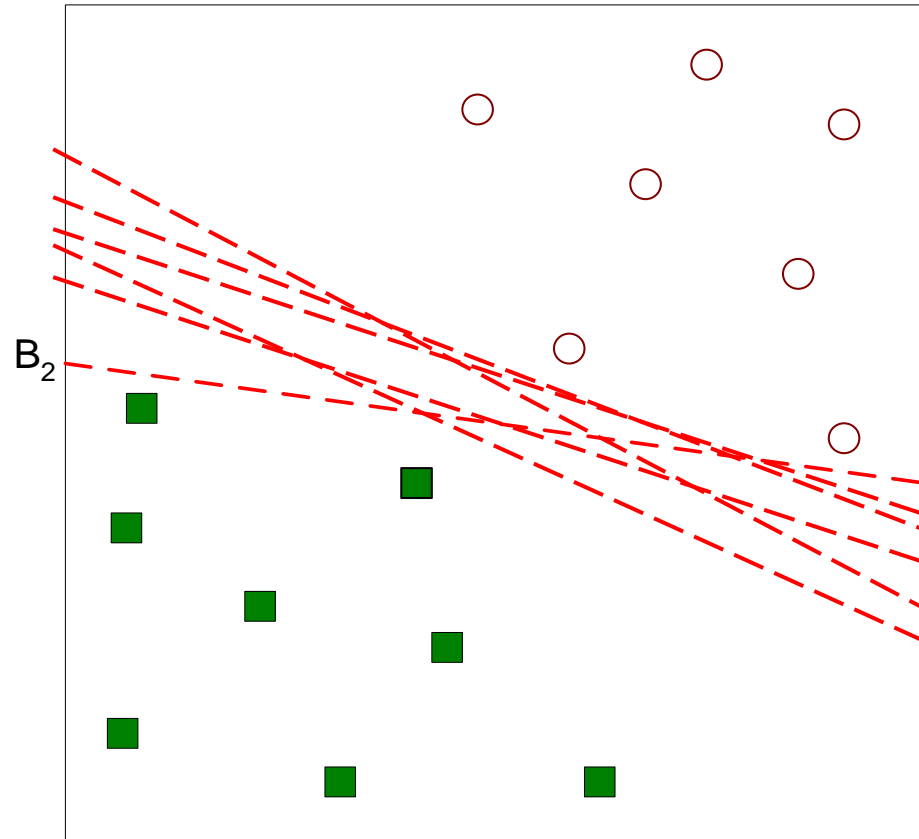
- One Possible Solution

# Support Vector Machines



- Another possible solution

# Support Vector Machines

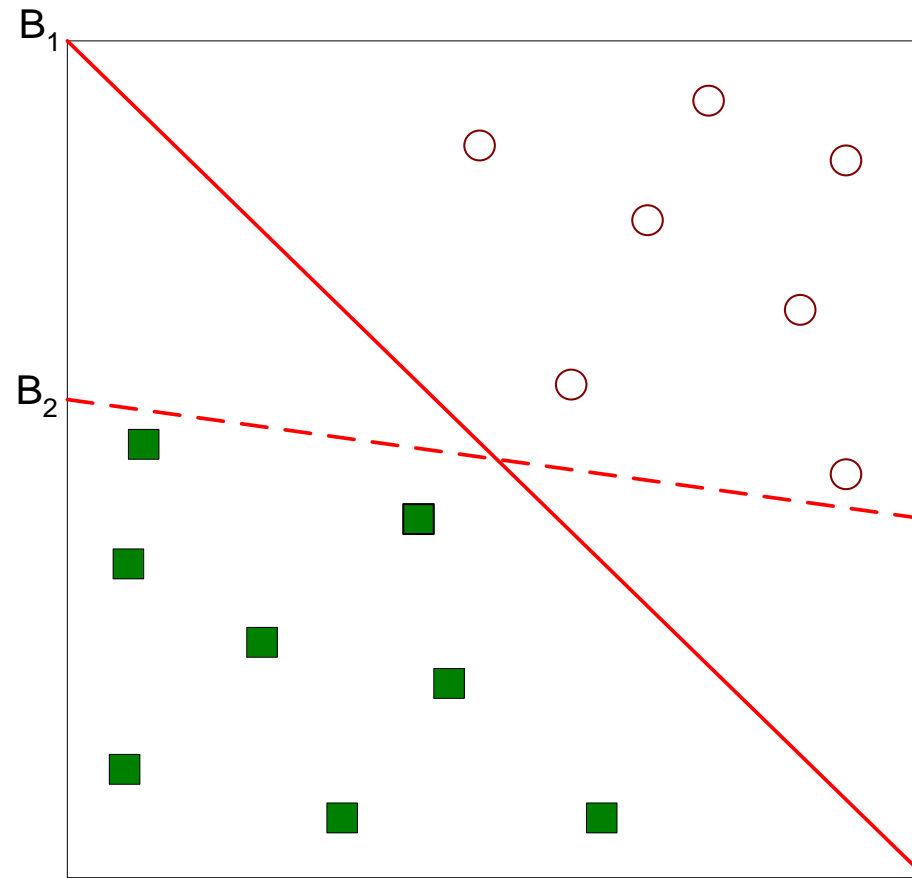


Many other possible solutions!

- We need a criteria to evaluate the quality of the hyperplane

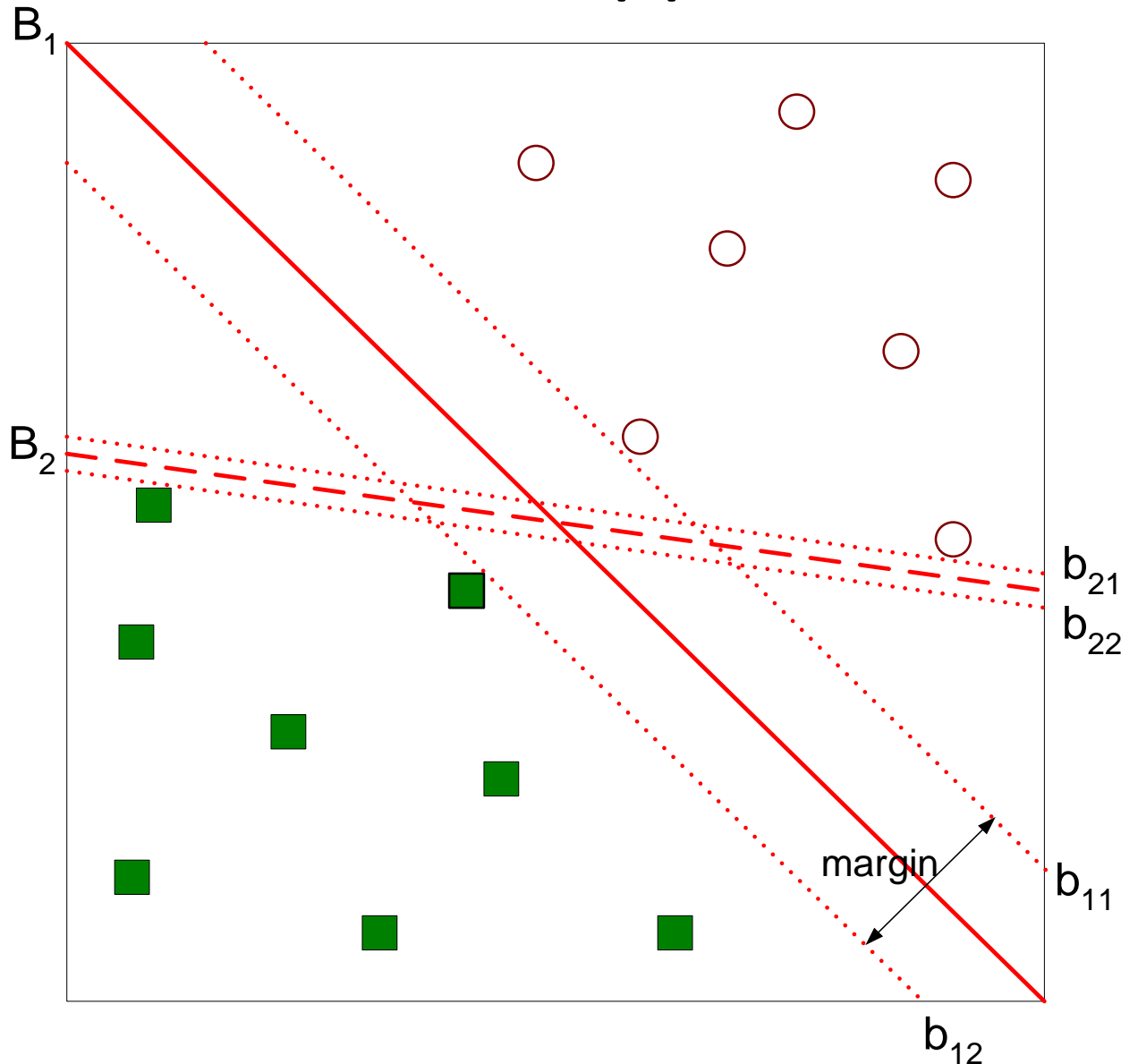


# Support Vector Machines



- Which one is better?  $B_1$  or  $B_2$ ? How do you define better?

# Support Vector Machines

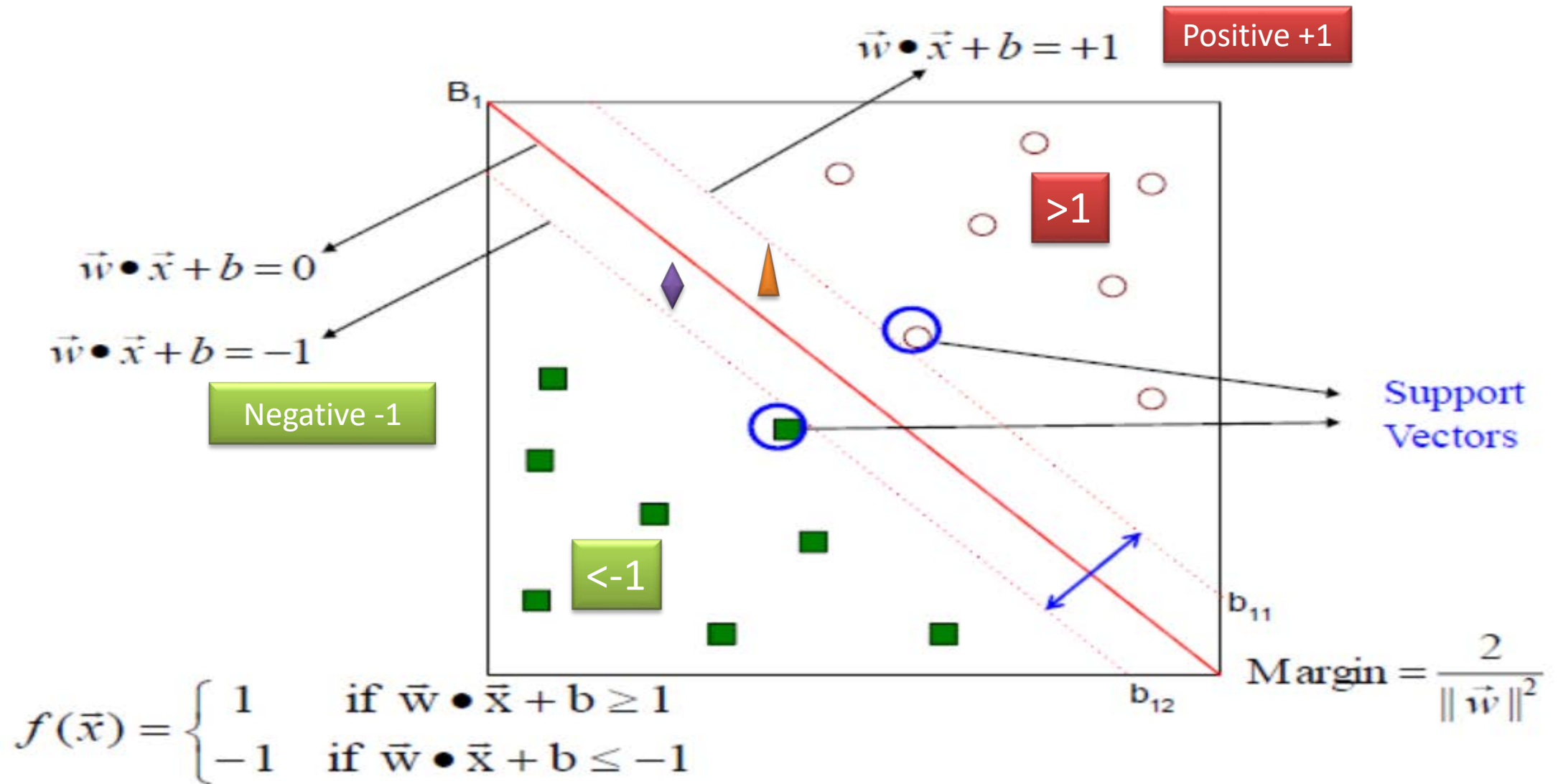


This can be modelled as an *optimization* problem, i.e. select a *best* hyperplane that can *best* separate the two classes, with largest margin

It seems boundary examples are very important to define hyperplane and margins, as we need to classify them correctly

- Find hyperplane **maximizing the margin** => B1 is better than B2, why?

# Support Vector Machines



Find  $w$  and  $b$ , so that all the training examples can be correctly classified

# Support Vector Machines

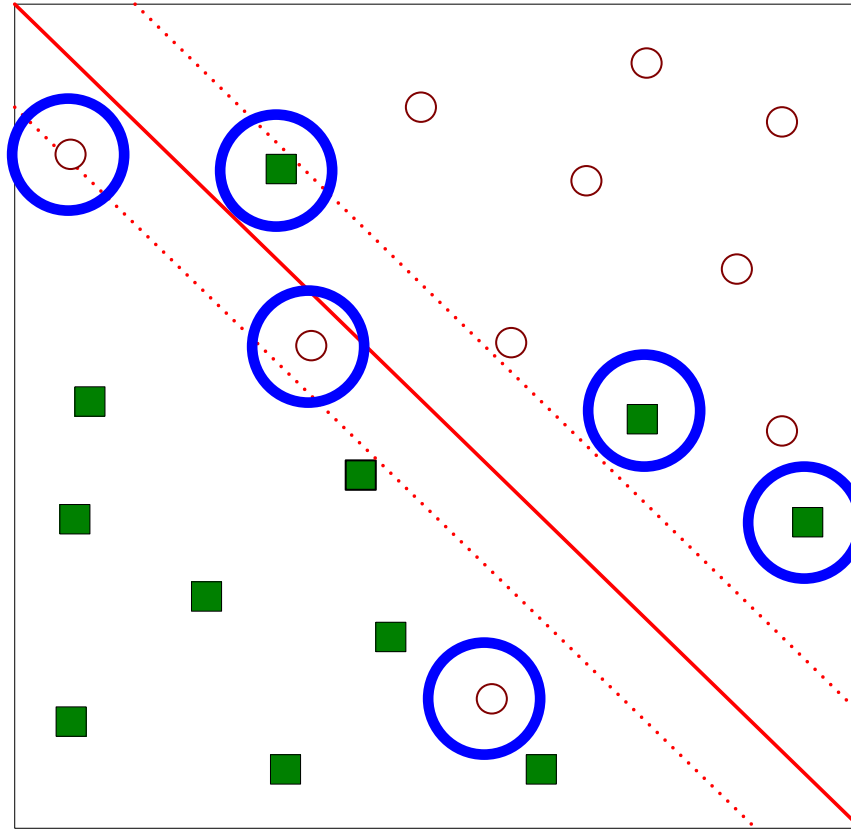
- We want to maximize:  $\text{Margin} = \frac{2}{\|\vec{w}\|^2}$ 
  - Which is equivalent to minimizing:  $L(w) = \frac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:
$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$
  - This is a constrained optimization problem, which can be solved by some numerical approaches, e.g., quadratic programming (QP)

$f(x_i)$  is a classifier that can classify any test example  $x_i$ :

If  $f(x_i) \geq 0$ , then  $x_i$  is positive class; negative class otherwise.

# Support Vector Machines

- What if the problem is not linearly separable?

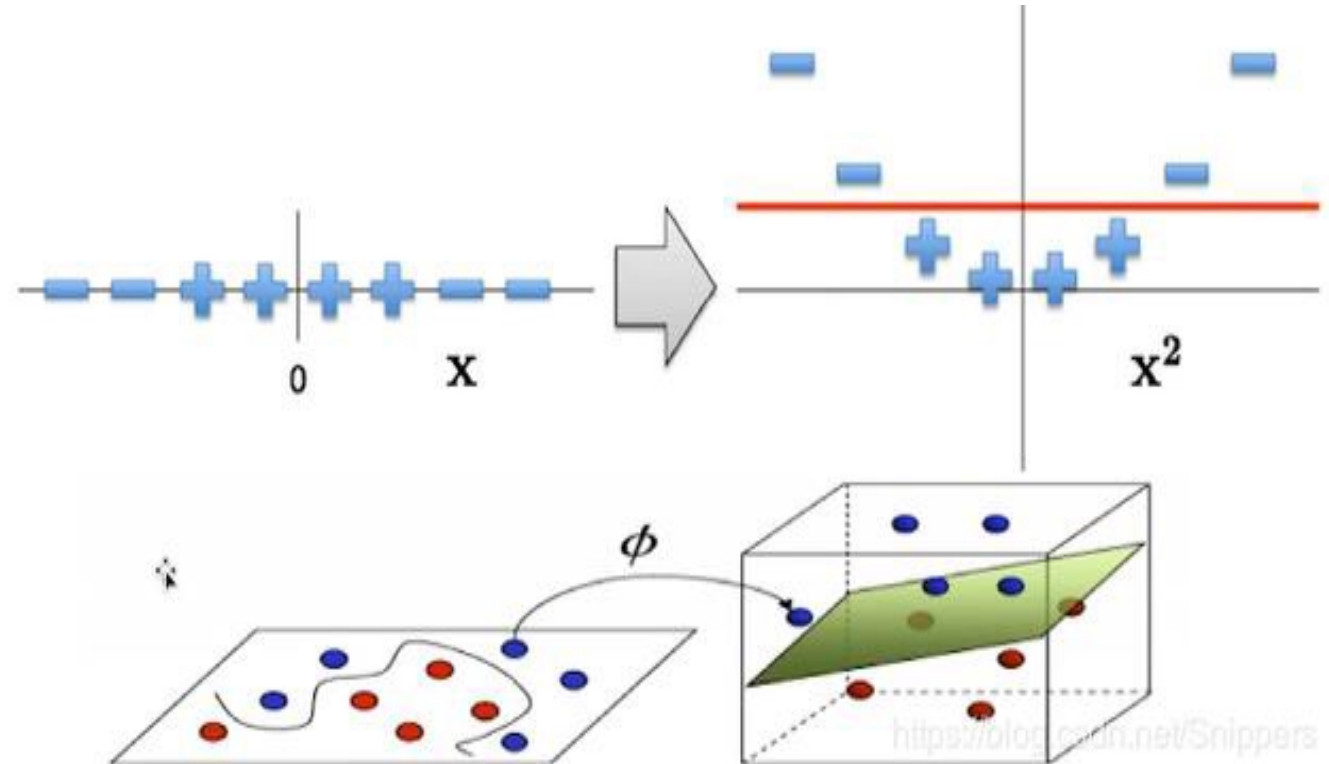


# High Level Idea for Nonlinear SVM

- Transform data into higher dimensional space
  - Using “kernel trick”, actual transformation need not be known
  - Just compute similarity between two vectors in original space
- Some Kernels:

$$K(x, y) = (x^T y + 1)^p$$

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



SVMlight <http://svmlight.joachims.org/>

LIBSVM <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

# SVMlight <http://svmlight.joachims.org/>

- **svm\_learn** train **model**
- **svm\_classify** test **model** predictions
- Train or test data format

Feature 2 and 3 are 0, so we skip.

1 1:0.58 4:0.34 6:0.78 7:0.9, ...

.....

-1 1:0.43 3:0.12 7:0.98 9:0.2, ...

Class  
label

Feature  
index

Feature  
value

## Kernel options:

-t int	- type of kernel function:
	0: linear (default)
	1: polynomial $(s a*b+c)^d$
	2: radial basis function $\exp(-\gamma   a-b  ^2)$
	3: sigmoid $\tanh(s a*b + c)$
	4: user defined kernel from kernel.h
-d int	- parameter d in polynomial kernel
-g float	- parameter gamma in rbf kernel
-s float	- parameter s in sigmoid/poly kernel
-r float	- parameter c in sigmoid/poly kernel
-u string	- parameter of user defined kernel

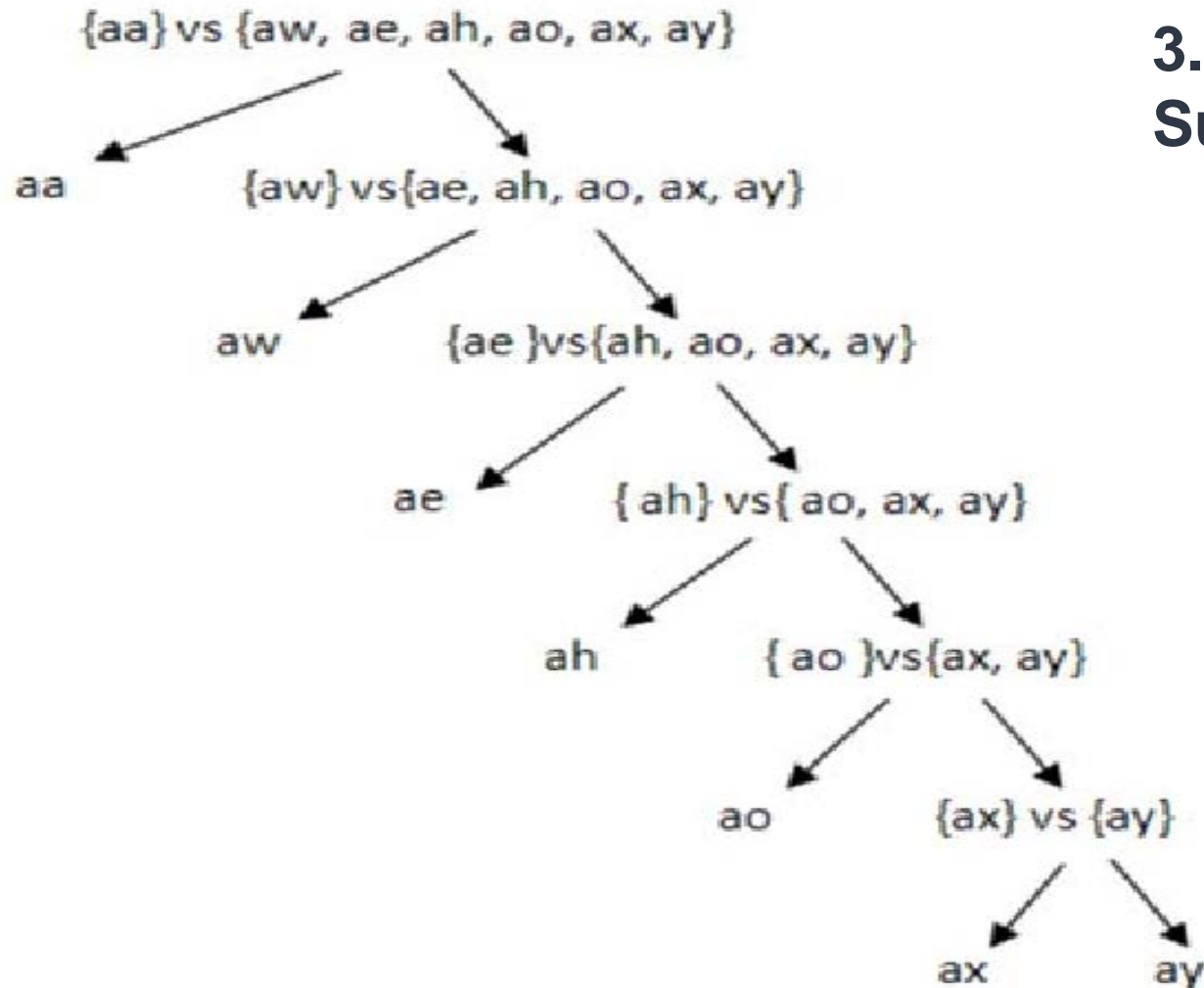
- **svm\_learn -t 1** train model
- Train non-linear SVM model with polynomial kernel

# Typical Manner to Apply SVM for Multiclass Classification

- Suppose we have  $n$  classes (in training data)
- 1. Most widely used method is **OVR (One vs. Rest)**. For each class  $i$ , you will need to build a classifier using class  $i$  as positive class and Rest ( $n-1$  classes, i.e. all class excluding class  $i$ ) as negative class. We will predict a test example as the class which has the highest positive score (or consider it belong to multiple classes based on the classifiers' decision).
- 2. Another option is **OVO (One vs. One)**. We will need to build  $n*(n-1)/2$  classifiers (all class pairs,  $C_1$  vs  $C_2$ , ...,  $C_1$  vs  $C_n$ ,  $C_2$  vs  $C_3$ , ...,  $C_2$  vs  $C_n$ , ...,  $C_{n-1}$  vs  $C_n$ ), and the resulting class is obtained by majority votes of all classifiers.




# Binary tree



## 3. OVA Tree Multiclass Framework for Support Vector Machine

<https://www.semanticscholar.org/paper/OVA-Tree-Multiclass-Framework-for-Support-Vector-Sidaoui/565157e2fcaeafca0f69fdd060edf8ffb5fcd154>

# Outline

- 1. Instance-Based Classifiers: KNN Classifier
- 2. Support Vector Machines
- 3. Random Forests 
- 4. Imbalanced Learning
- 5. Semi-Supervised learning

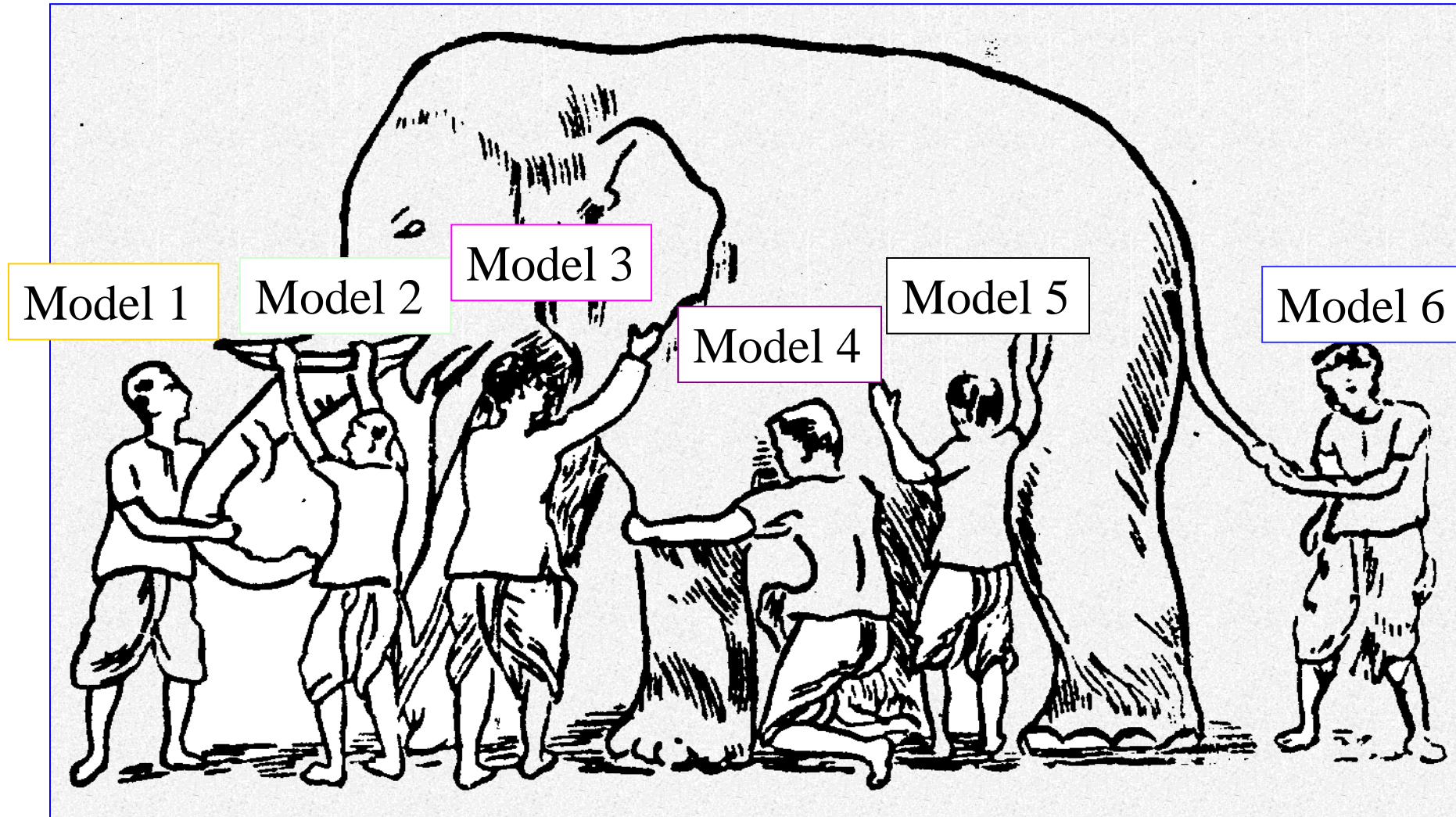


## 3. Random Forests

In financial, medical, social, or other domains, we often seek additional opinions before making a decision. In doing so, we weigh the individual opinions, and combine them through some thought process to reach a final decision that is presumably the most informed one.


















































The process of consulting “several experts” before making a final decision is perhaps second nature to us; yet, the extensive benefits of such a process in automated decision making applications have only recently been discovered by computational intelligence community.

# Why Ensemble Works?



**Each Model is good at a specific area and ensemble gives the global picture! More robust**

# An Example of Ensemble Weather Forecast

		Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	
	<b>Reality/ Ground Truth</b>								
	Model 1								
	Model 2								
	Model 3								
	Model 4								
	Model 5								
	<b>Ensemble results</b>								

Majority voting of 5 different prediction models over next 7 days

# Random Forests – an example of ensemble learning

- ***Random Forests*** is an ensemble method specifically designed for decision tree classifiers
- ***Random Forests*** grows many classification trees
- Ensemble of unpruned decision trees
- Each base classifier classifies a “new” test example (vector) and ***Random Forests*** chooses the classification having the most votes (over all the trees in the forest)



# Random Forests (Cont.)

Why is it called *Random* Forests?

Introduce two sources of *randomness*:

- Random example selection: Each tree is grown using a sample of training data
- Random feature selection: At each node, best split is chosen from random sample of  $m$  variables instead of all  $M$  input variables/features ( $m < M$ )

# Choose Part of Variables for Decision Tree

**variables**

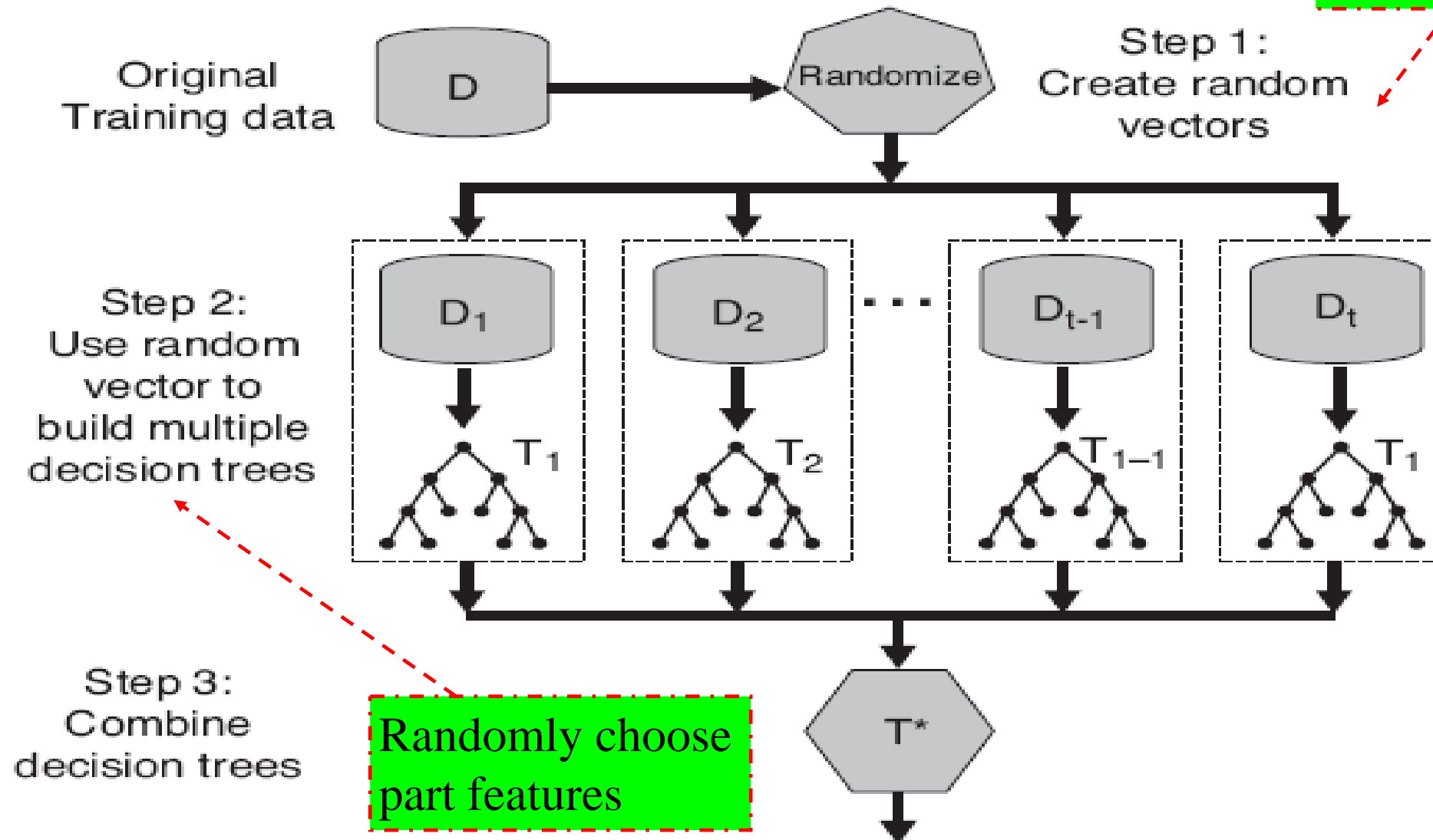
Age	Has_Job	Own_House	Credit_Rating	Class
young	false	false	fair	No
young	false	false	good	No
young	true	false	good	Yes
young	true	true	fair	Yes
young	false	false	fair	No
middle	false	false	fair	No
middle	false	false	good	No
middle	true	true	good	Yes
middle	false	true	excellent	Yes
middle	false	true	excellent	Yes
old	false	true	excellent	Yes
old	false	true	good	Yes
old	true	false	good	Yes
old	true	false	excellent	Yes
old	false	false	fair	No

**Examples/  
Tuples**

**Class labels**



# Random Forests (cont)



Randomly choose examples

Randomly choose part features

# Random Forest Algorithm

$M$  input **variables**, a number  $m \ll M$  is specified such that at each node,  $m$  variables are selected at random out of the  $M$  and the best split on these  $m$  is used to split the node.


$m$  is held constant during the forest growing

Fast algorithm

Easily parallelized

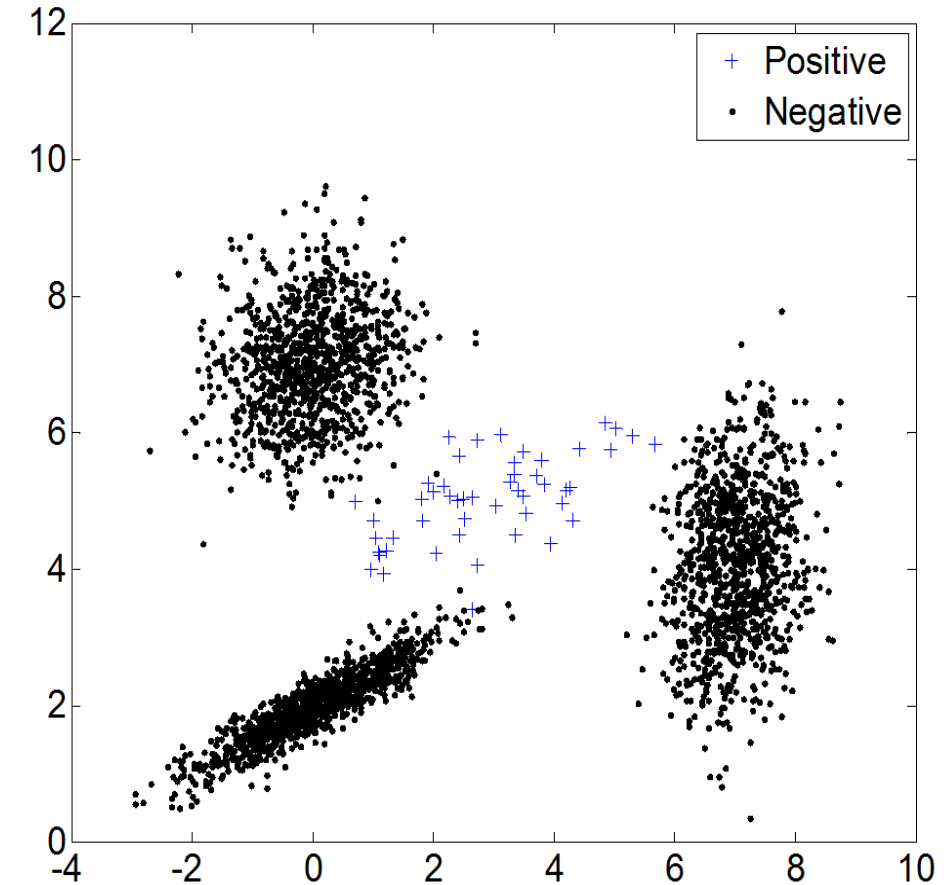
Handle high dimensional data without much problem

# Outline

- 1. Instance-Based Classifiers: KNN Classifier
- 2. Support Vector Machines
- 3. Random Forests
- 4. Imbalanced Learning 
- 5. Semi-Supervised learning

# Learning from Imbalanced Data

- ***Class imbalance is prevalent in many applications: fraud/intrusion detection, risk management, medical diagnosis/monitoring, spam email filtering etc.***
- ***Standard classifiers tend to be overwhelmed by the large classes and ignore the small ones, tend to produce high predictive accuracy over the majority class, but poor predictive accuracy over the minority class***



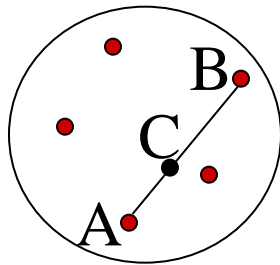
# Overall Solutions to Class Imbalance Problem

- At the data level (re-samplings)
  - **Under-sampling**: extract a smaller set of majority instances while preserving all the minority instances
  - **Over-sampling**: increases the number of minority instances by over-sampling them
- At the algorithmic level
  - **Cost-sensitive based**: adjust the costs of the various classes (cost matrix) so as to counter the class imbalance

# Over-sampling Techniques

- *Over-sampling by duplicating the minority examples*
- **SMOTE**: Synthetic Minority Over-sampling Technique

The minority class is over-sampled by taking each minority class sample and introducing **synthetic examples** along the line segments joining any/all of the  $k$  minority class nearest neighbors.



$$C = \alpha * A + (1 - \alpha) * B$$

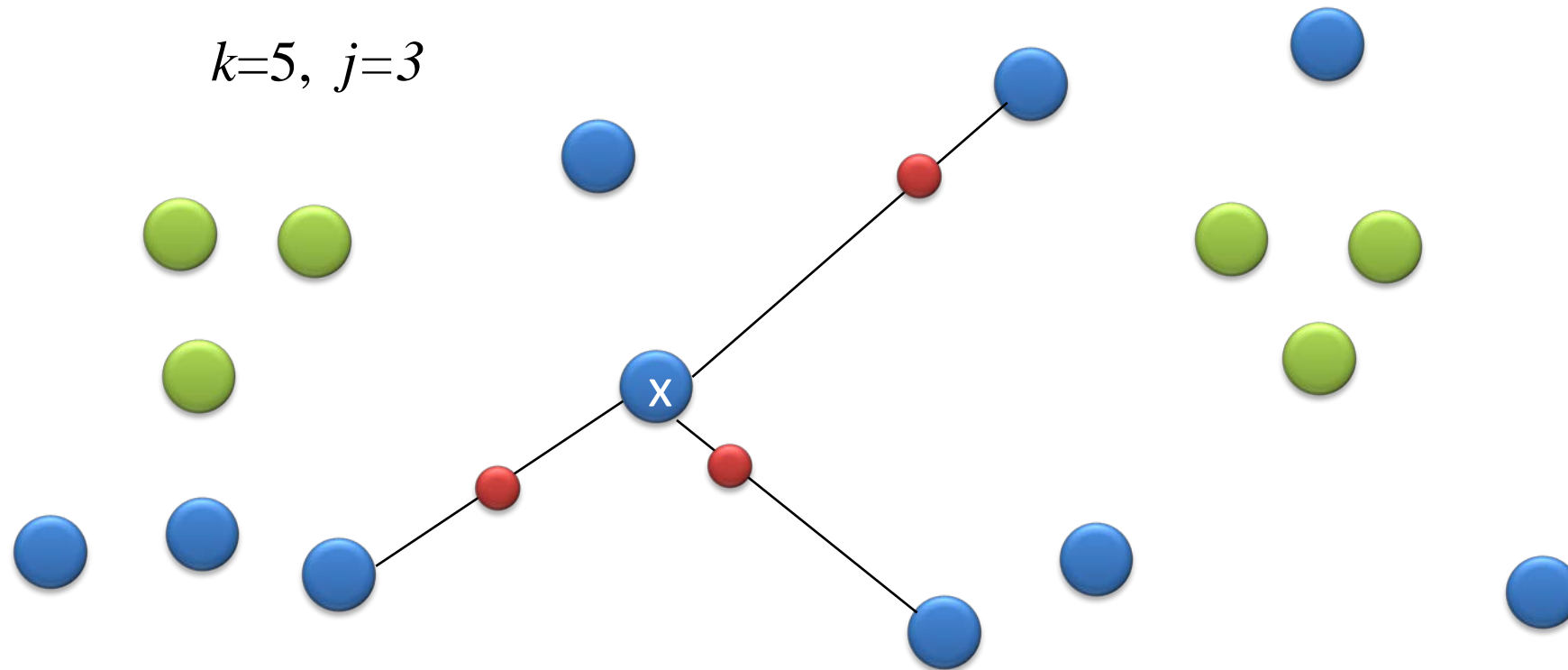
$$0 < \alpha < 1$$



# SMOTE Technique

- Proposed by Chawla, Hall, & Kegelmeyer in 2002.
  - For each minority sample, we will generate  $j$  more samples ( $k$  is a parameter and  $k > j$ )
    - Find its  $k$ -nearest minority neighbors
    - Randomly select  $j$  of these neighbors
    - Randomly generate synthetic samples along the lines joining the minority sample and its  $j$  selected neighbors
- $j$  depends on the number of oversampling examples desired.

# SMOTE illustration and problems



: Minority sample



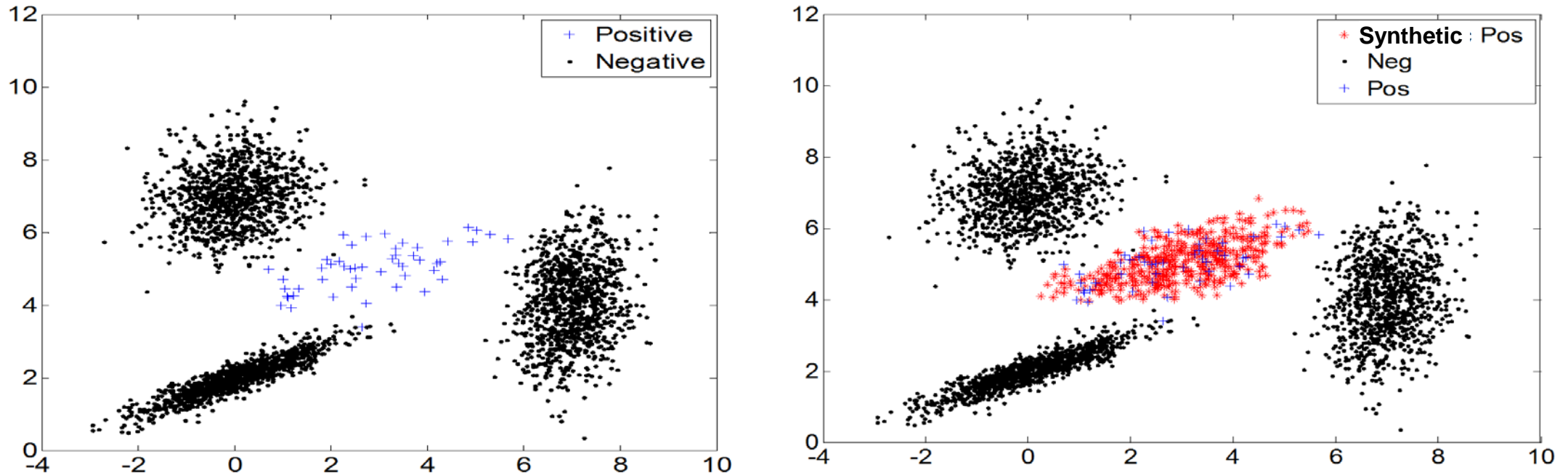
: Synthetic sample



: Majority sample




# Imbalanced learning: Oversampling



- The positive (Pos) class, containing only a few sparsely distributed samples, is outnumbered by the negative (Neg) class. Consequently, the learning algorithms tend to bias towards the less important Neg class with the larger population.

Hong Cao, Xiao-Li Li, Yew-Kwong Woon and See-Kiong Ng, "Integrated Oversampling for Imbalanced Time Series Classification", IEEE Transactions on Knowledge and Data Engineering (TKDE), 2013

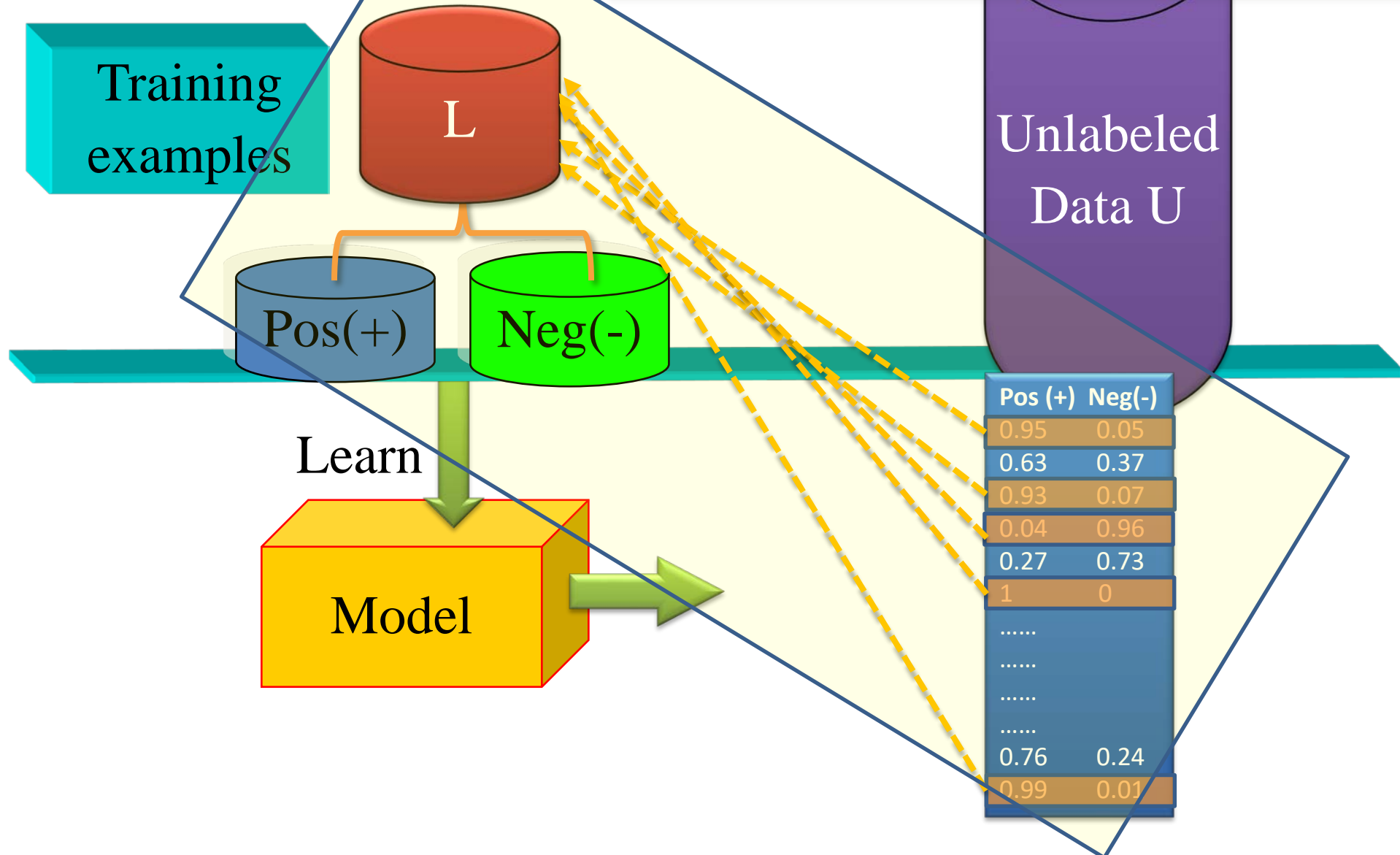
# Outline

- 1. Instance-Based Classifiers: KNN Classifier
- 2. Support Vector Machines
- 3. Random Forests
- 4. Imbalanced Learning
- 5. Semi-Supervised learning 

# Semi-Supervised Learning

## LU learning

Using L and U (with soft labels) to build a new model



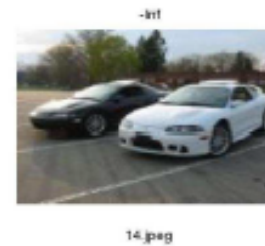
# Semi-Supervised Learning

also known as Self-training ..

Self-training example: image categorization

1. Train a naïve Bayes classifier on the two initial labeled images

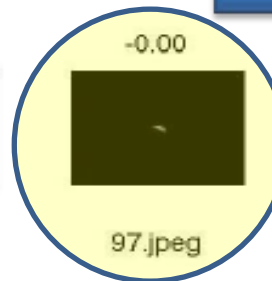
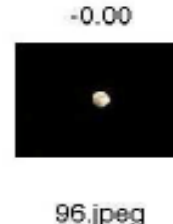
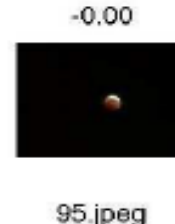
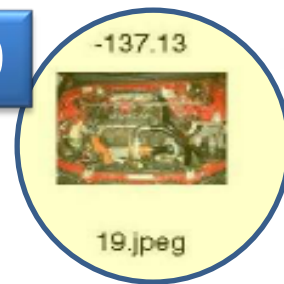
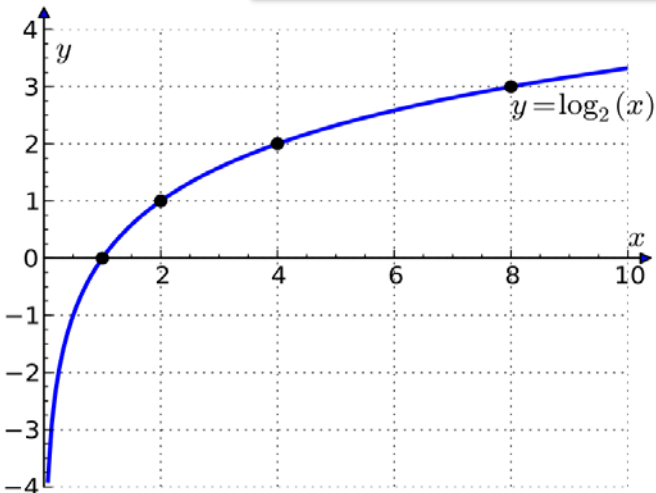
Class 1: astronomy



Class 2: Car

2. Classify unlabeled data, sort by confidence  $\log p(y = \text{astronomy} | x)$

$p(y = \text{astronomy}) \approx 0$

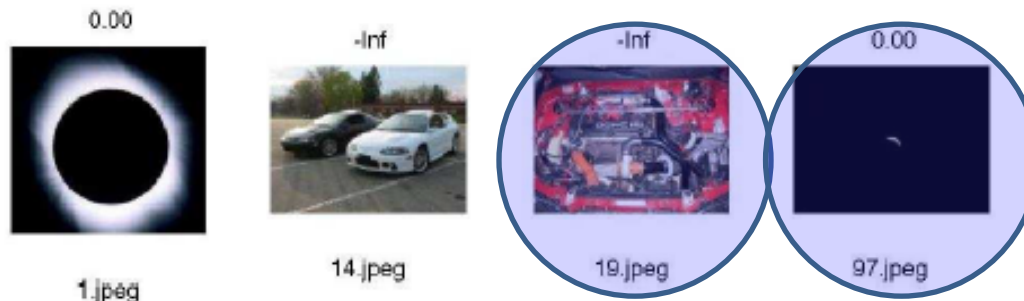


$p(y = \text{astronomy}) \approx 1$

# Self-training ..

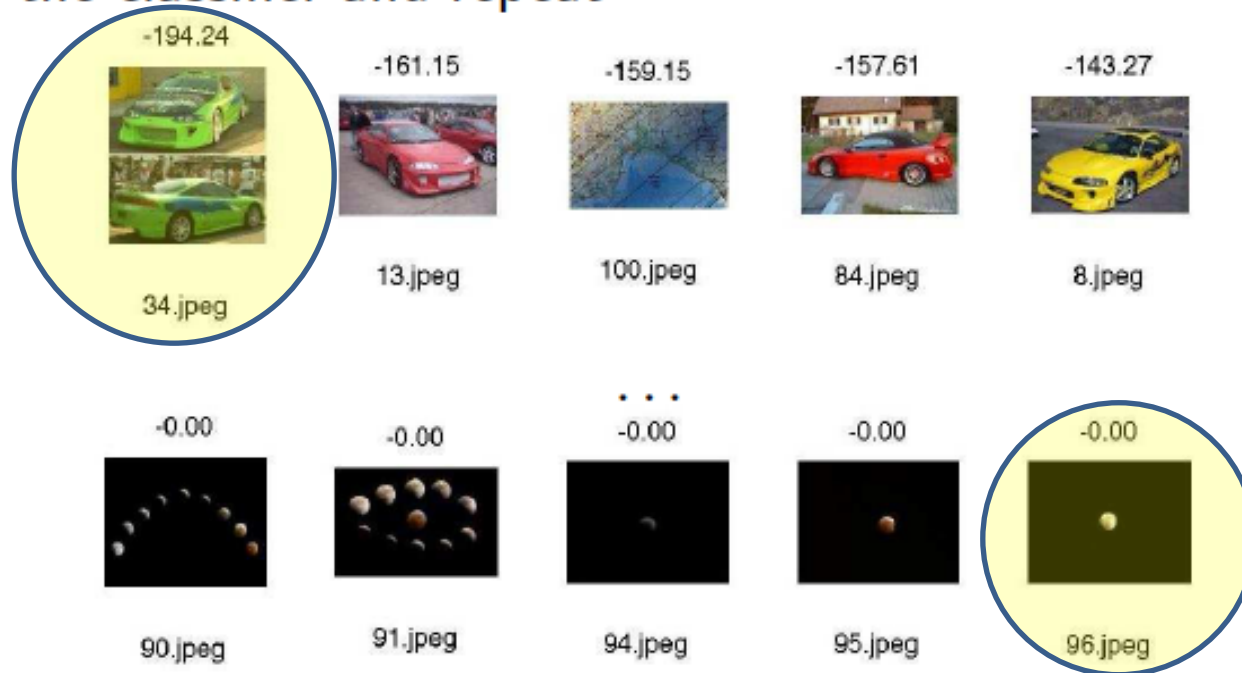
## Self-training example: image categorization

3. Add the most confident images and **predicted** labels to labeled data



Good to control the amount (adding to training data) to avoid introducing noisy examples

4. Re-train the classifier and repeat



Characteristic:  
Adding quality/reliable examples to increase the size of training examples

# Thank You

Contact: [xli@i2r.a-star.edu.sg](mailto:xli@i2r.a-star.edu.sg) if you have questions