For written notes on this lecture, please read chapter 3 of *The Practical Bioinformatician. Alternatively, please read* "Rule-Based Data Mining Methods for Classification Problems in Biomedical Domains", a tutorial at *PKDD04* by Jinyan Li and Limsoon Wong, September 2004. http://www.comp.nus.edu.sg/~wongls/talks/pkdd04/

CS2220: Introduction to Computational Biology

Unit 2: Essence of Knowledge Discovery

Li Xiaoli

NUS National University of Singapore

Outlines

- **1. Decision Tree Ensembles**
- **2. Other Machine Learning Approaches**
  - kNN
  - NB
  - SVM
- **3. Classification Model Evaluation**
- **4. Feature Selection**

NUS National University of Singapore

Motivating Example

- h_1, h_2, h_3 are indep classifiers w/ accuracy = 60%
- C_1, C_2 are the only classes
- t is a test instance in C_1
- $h(t) = \text{argmax}_{C \in \{C1,C2\}} |\{h_j \in \{h_1, h_2, h_3\} \mid h_j(t) = C\}|$
- Then prob($h(t) = C_1$)

Majority class

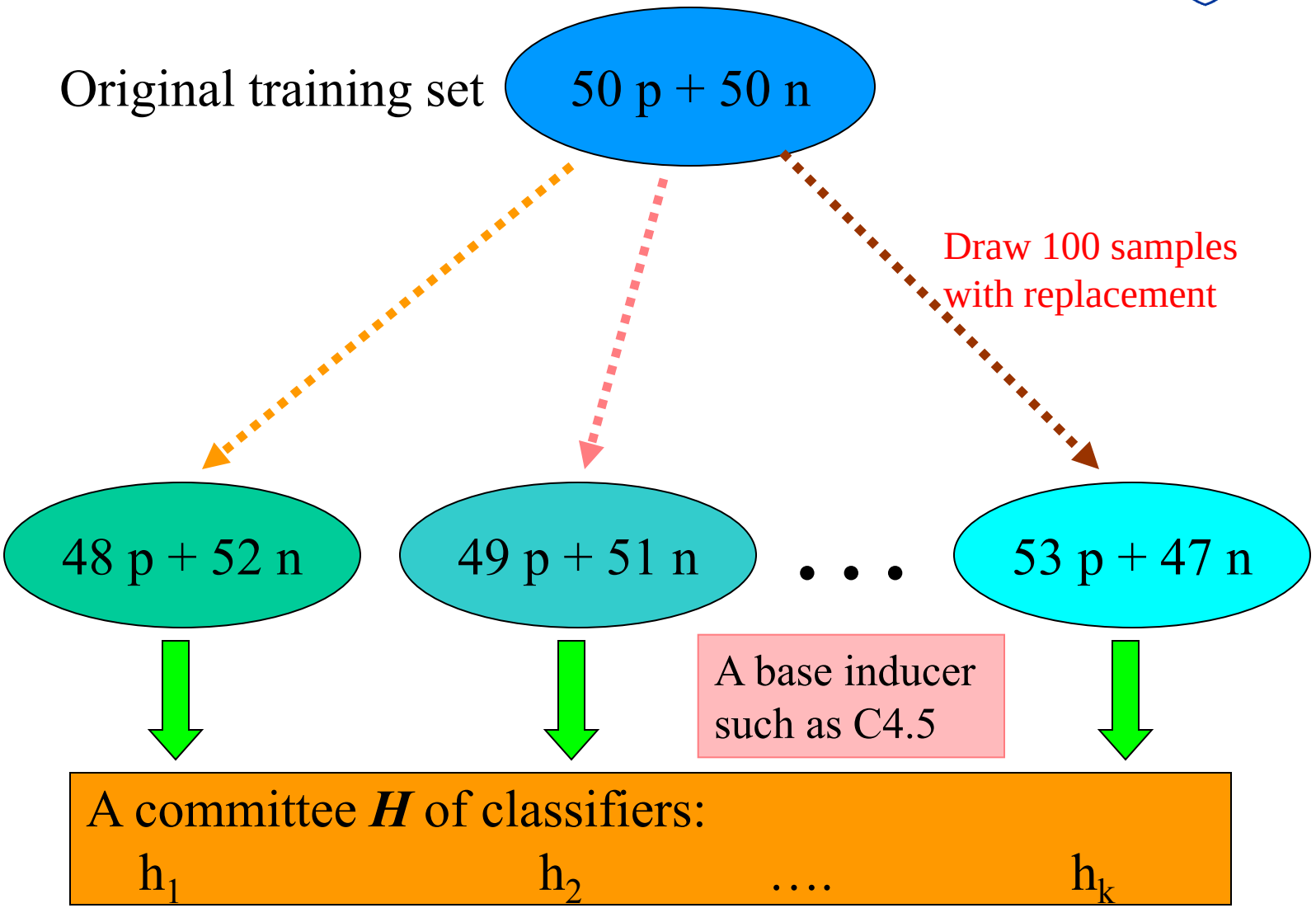
$$= \text{prob}(h_1(t)=C_1 \,\&\, h_2(t)=C_1 \,\&\, h_3(t)=C_1) +$$
$$\text{prob}(h_1(t)=C_1 \,\&\, h_2(t)=C_1 \,\&\, h_3(t)=C_2) +$$
$$\text{prob}(h_1(t)=C_1 \,\&\, h_2(t)=C_2 \,\&\, h_3(t)=C_1) +$$
$$\text{prob}(h_1(t)=C_2 \,\&\, h_2(t)=C_1 \,\&\, h_3(t)=C_1)$$
$$= 60\% * 60\% * 60\% + 60\% * 60\% * 40\% +$$
$$60\% * 40\% * 60\% + 40\% * 60\% * 60\% = 64.8\%$$

NUS National University of Singapore

Bagging

- **Proposed by Breiman (1996)**
- **Also called Bootstrap aggregating**
- **Make use of randomness injected to training data**

Main Ideas



Decision Making by Bagging

Given a new test sample T

Assign T with Majority class

$$bagged(T) = \text{argmax}_{C_j \in \mathcal{U}} |\{h_i \in \mathcal{H} \mid h_i(T) = C_j\}|$$

where $\mathcal{U} = \{C_1, ..., C_r\}$

In practice, we can build a random forest by building multiple decision trees. Each decision tree can be built by randomly choosing training examples and/or features

http://en.wikipedia.org/wiki/Random_forest

NUS National University of Singapore

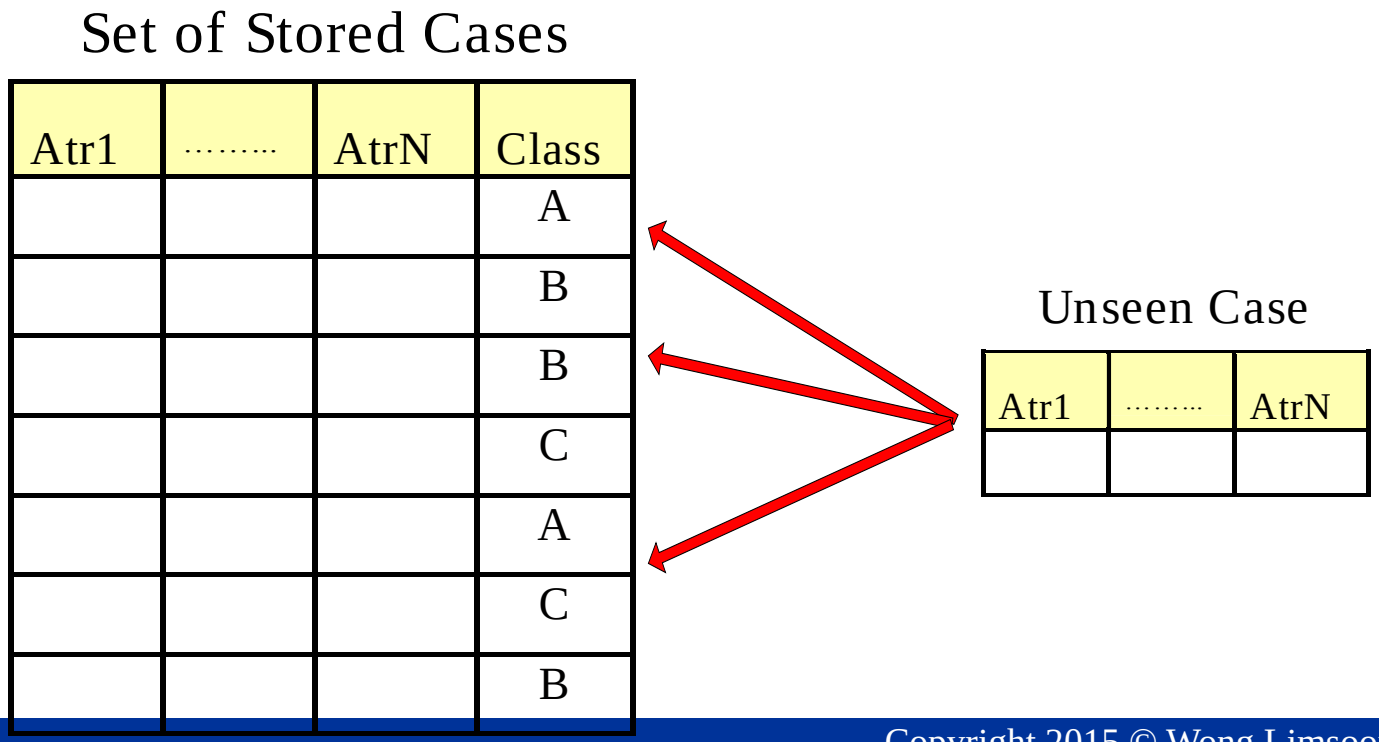
Outlines

- **1. Decision Tree Ensembles**
- **2. Other Machine Learning Approaches**
  - kNN (k-Nearest Neighbors)
  - NB (Naïve Bayesian Classifier)
  - SVM (Support Vector Machines)
- **3. Feature Selection**
- **4. Classification Model Evaluation**

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: deduction step

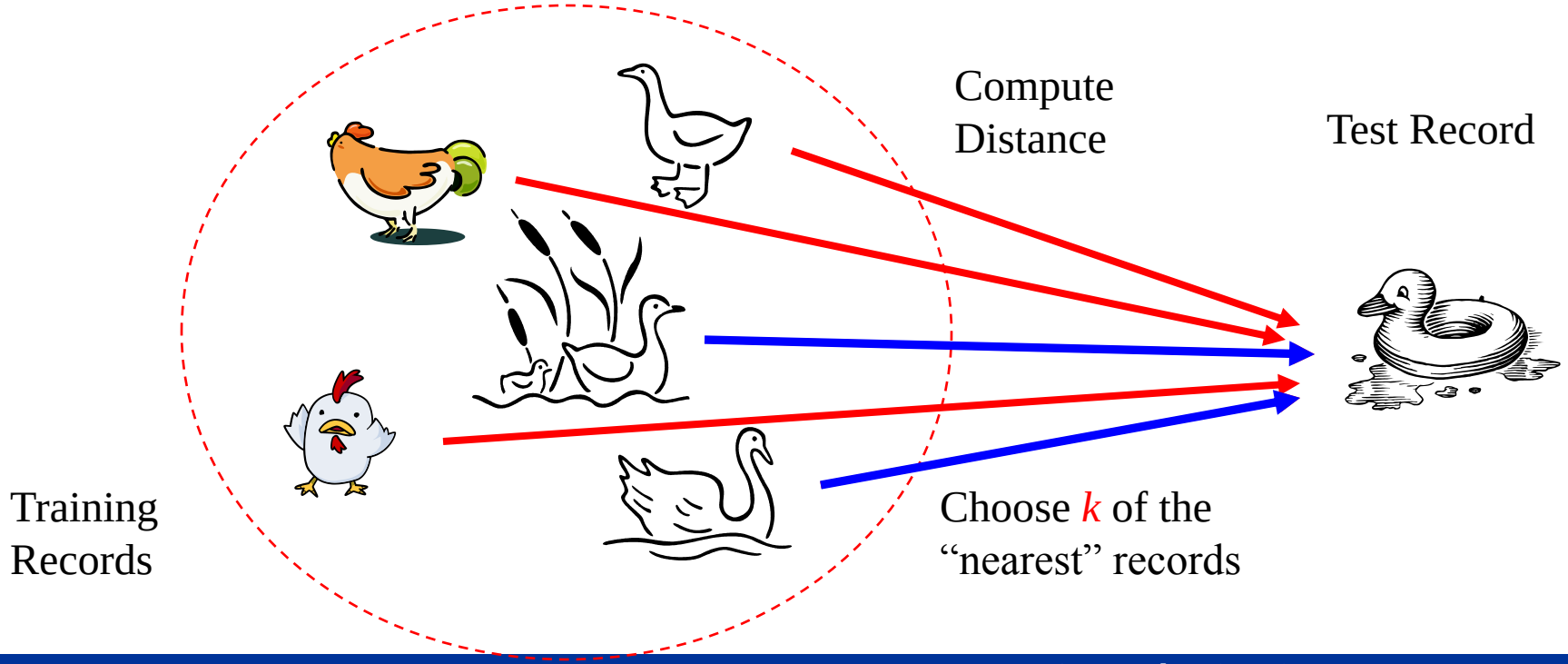


Instance Based Classifiers

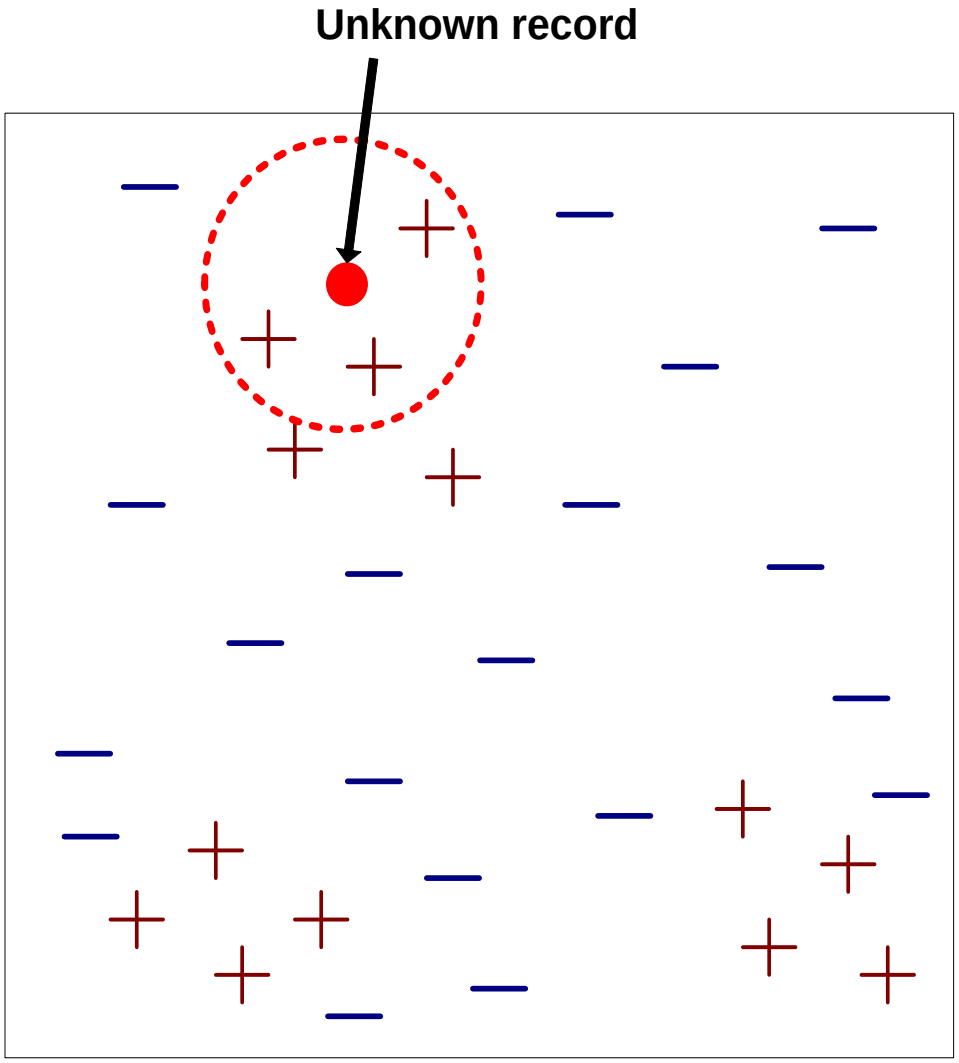
- **Rote-learner**
  - Memorizes entire training data and performs classification only if attributes of record **match** one of the training examples ***exactly***
- **k-Nearest Neighbors (k-NN)**
  - Uses *k* "closest" points (nearest neighbors) for performing classification

1) Nearest Neighbor Classifiers

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



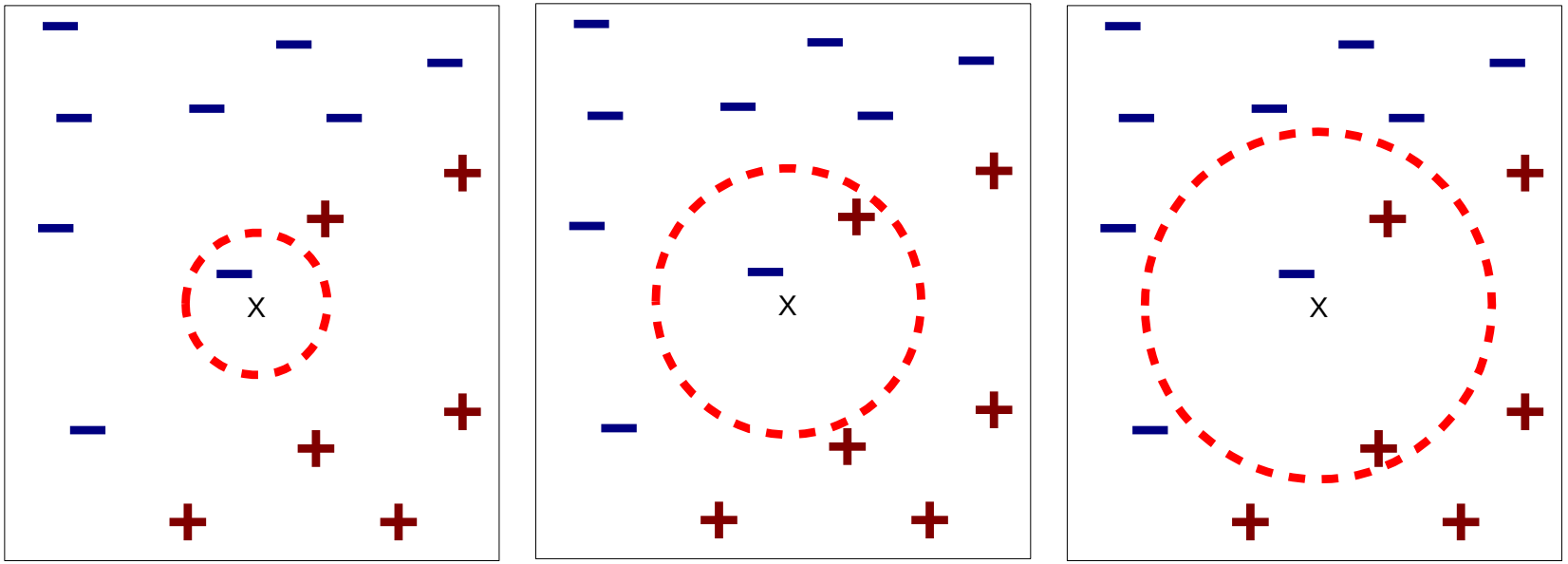
Nearest-Neighbor Classifiers



- **Requires three things**
  - The set of stored 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

Nearest Neighbor Classification

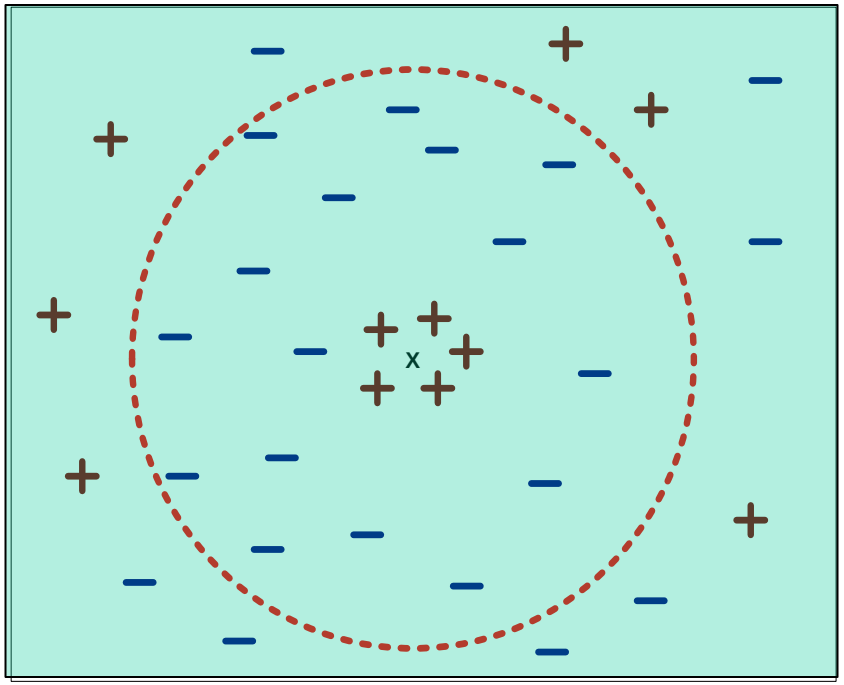
- **Compute distance between two points p & q:**
  - Euclidean distance:

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

- **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, $w = 1/d^2$**

Nearest Neighbor Classification

- **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

- **Scaling issues**
  - 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 1.4m to 2.4m**
    - **F2: weight of a person may vary from 40kg to 442kg**
    - **F3: Annual income of a person may vary from $10K to $5,000K**

$p = (p_1\, p_2\, p_3) = (1.65, 48, 6000)$

$q = (q_1\, q_2\, q_3) = (1.82, 75, 8000)$

F3 dominates the calculation of Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2} = \sqrt{(1.65-1.82)^2 + (48-75)^2 + (6000-8000)^2}$$

Normalization

- **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:

    Income range [$12,000, $98,000] normalized to [0.0, 1.0]. Then $73,000 is mapped to

$$\frac{73{,}000 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.71$$

$$\frac{12{,}000 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0 \qquad \frac{98{,}000 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 1$$

Nearest Neighbor Classification (cont'l)

NUS National University of Singapore

- **Pros of kNN**
  - Easy to implement
  - Incremental addition of training data trivial
- **Cons**
  - k-NN classifiers are **lazy** learners, which do not build models **explicitly**. This can be relatively 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.

Example Use of kNN: Ovarian Cancer Diagnosis Based on SELDI Proteomic Data

- **Li et al, *Bioinformatics* 20:1638-1640, 2004**
- **Use kNN to diagnose ovarian cancers using proteomic spectra**
- **Data set is from Petricoin et al., *Lancet* 359:572-577, 2002**

2) Bayesian Classifier

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

$$P(C \mid A) = \frac{P(A,C)}{P(A)} \quad (1)$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)} \quad (2)$$

- **Bayes Theorem:**

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)} \quad (3)$$

The Basic Idea to Apply Bayes Theorem

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

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No

- C={Y, N}
- A is a test case, we want to predict its probability belonging to class C, e.g. given that

  A=(Refund=Yes, Marital=Married, Taxable=79), which class A belong to? Y or N

P(Y|Refund=?, Marital=?, Taxable=79)

P(N|Refund=?, Marital=?, Taxable=79)

We will choose the bigger one

- Bayes Theorem: to compute P(C|A), we need to estimate P(A|C).
- P(C) is easy to compute. There is no need to compute P(A)

The Basic Idea to Apply Bayes Theorem

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

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- C={Y, N}
- **Objective**

P(Y|Refund=?, Marital=?, Taxable=79)

P(N|Refund=?, Marital=?, Taxable=79)

- **What we need to compute?**

1) P(C)

P(Y) = 3/10, P(N) = 7/10

2) P(A|C): using indep assumption

P(Refund=?|Y), P(Refund=?|N)

P(Marital =?|Y), P(Marital l=?|N)

P(Taxable=79|Y), P(Taxable=79|N)

3) P(A|C) * P(C)

= $P(A_1|C) * P(A_2|C)$*… *P(C)

e.g. P(Marital=Married|N)= 4/7 (out of 7 N examples, we have 4 Married)

P(Taxable Income =79|Y) = ? P(Taxable Income =179|Y) = ?

How to Estimate Probabilities from Data for continuous attributes

- **Discretize the range into bins**
- **Probability density estimation:**
  - Assume attribute follows a normal distribution
  - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
  - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|C)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- **Normal distribution:**

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, C_i) pair

- **For (Taxable Income, Class=No):**

Class=No

  - **sample mean = 110**
  - **sample variance = 2975**
  - **Stand Deviation**

$$\sigma = \sqrt{2975} = 54.54$$

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record: $X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$

Naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K|Class=No)
  = 4/7 × 4/7 × 0.0072
  = 0.0024
- P(X|Class=Yes)= P(Refund=No|Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × **0** × 1.2 × 10^{-9} = 0

Clearly, P(X|**No**)P(**No**) > P(X|**Yes**)P(**Yes**)

Therefore P(**No**|X) > P(**Yes**|X)
=> **Class = No**

Naïve Bayes Classifier: Smoothing

- **If one of the conditional probability is zero, then the entire expression becomes zero**
- **Probability estimation:**

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

N_{ic}: The number of times of feature A_i occurred in C

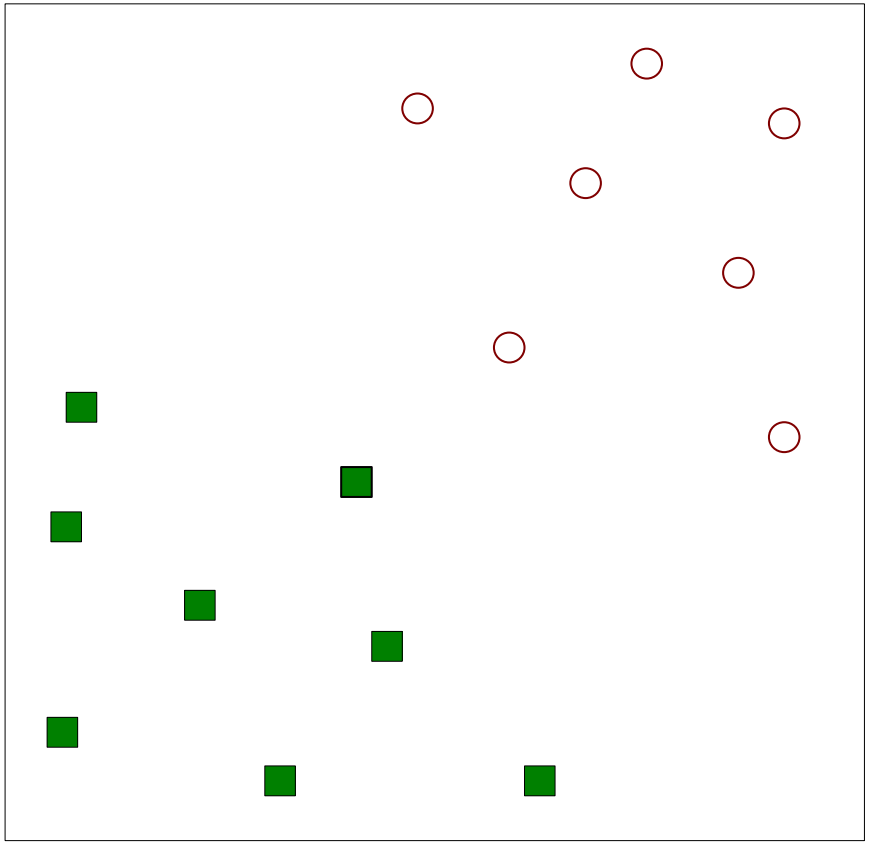
N_c : *the number of examples in C*

c: number of classes

Pros and Cons of Naïve Bayes Classifier

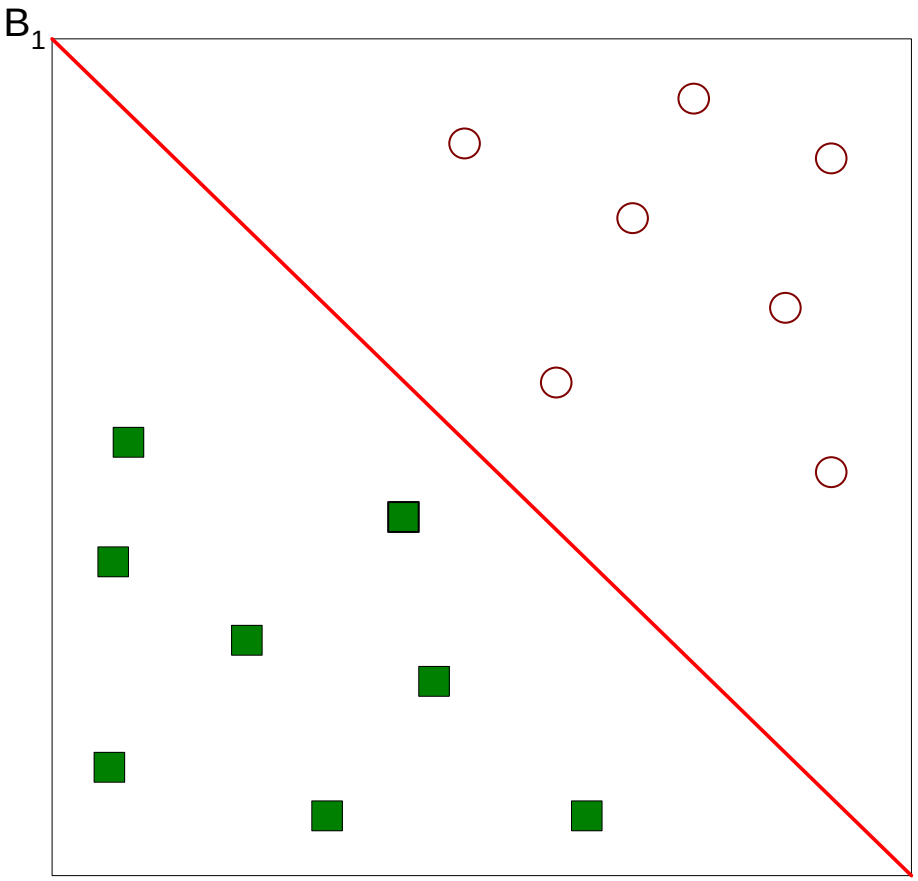
- **Pros**
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
  - Robust to isolated noise points
  - Handle missing values by ignoring the instance during probability estimate calculations
  - Robust to irrelevant attributes
- **Cons**
  - Independence assumption may not hold for some attributes (Could therefore loss of accuracy)
  - Use other techniques such as Bayesian Belief Networks (BBN)

3) Support Vector Machines



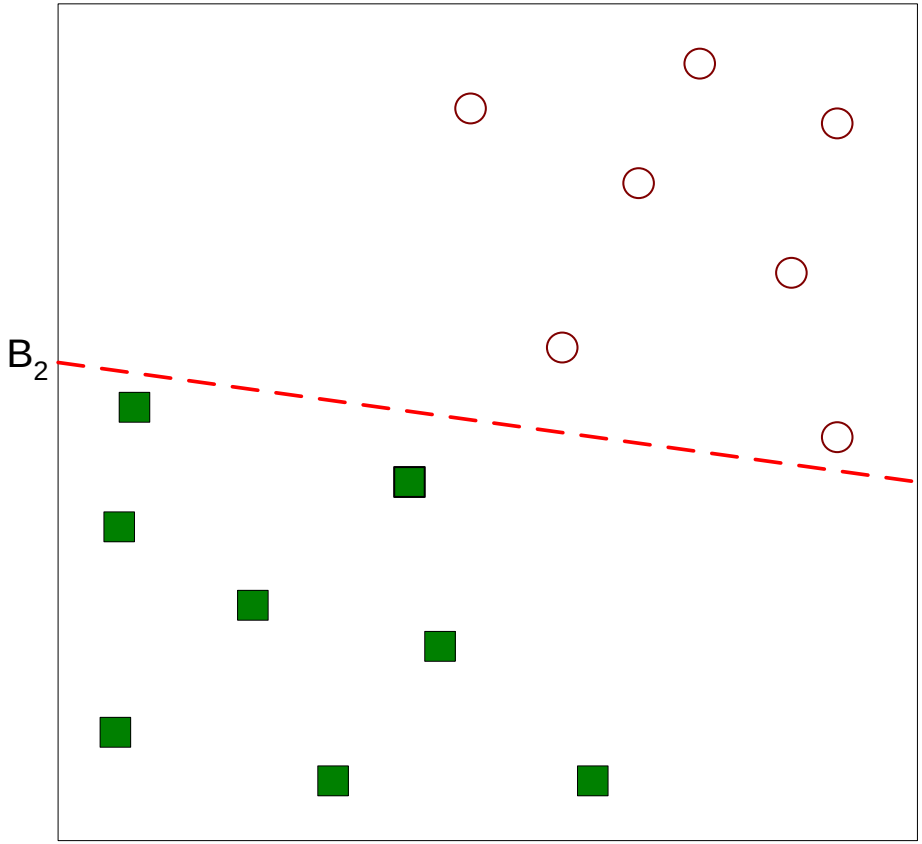
- **Find a linear hyperplane (decision boundary) that will separate the data**

Support Vector Machines



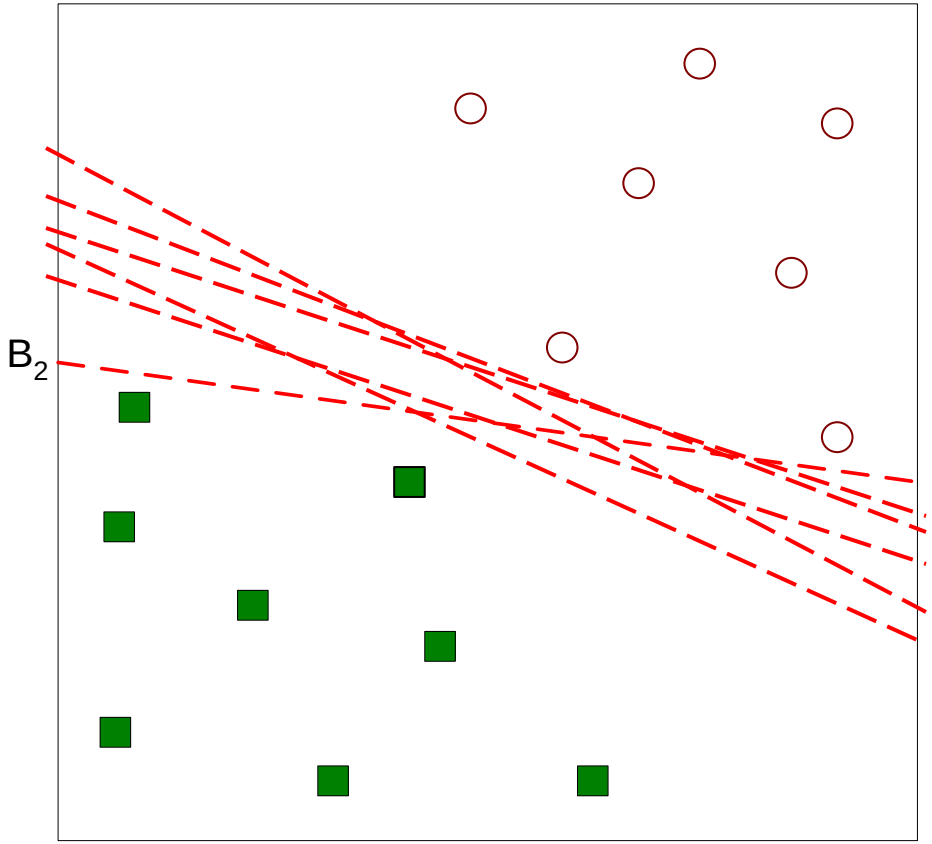
- **One Possible Solution**

Support Vector Machines



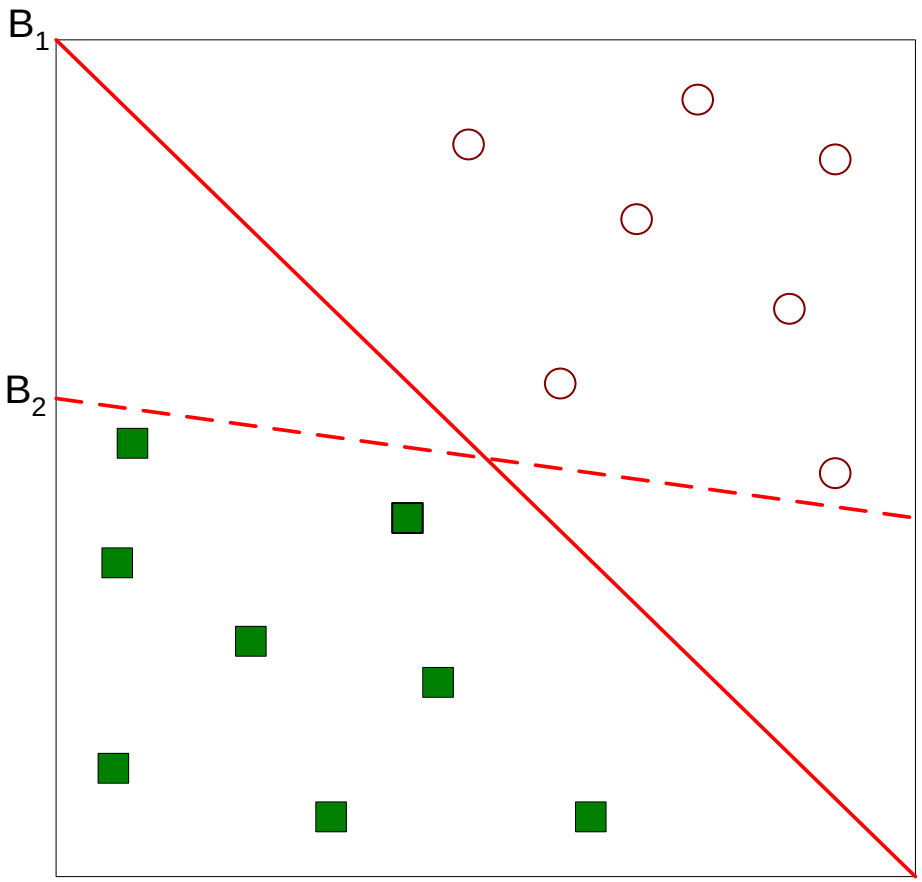
- **Another possible solution**

Support Vector Machines



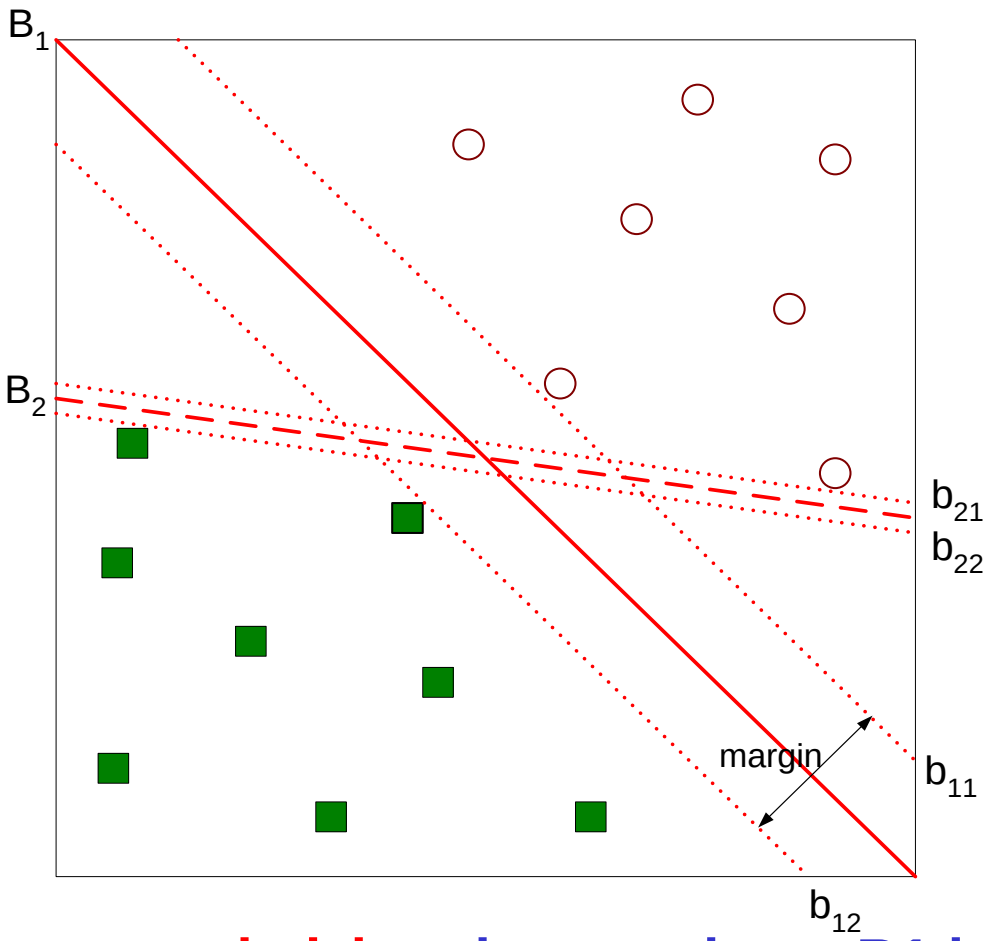
- **Many other possible solutions**

Support Vector Machines



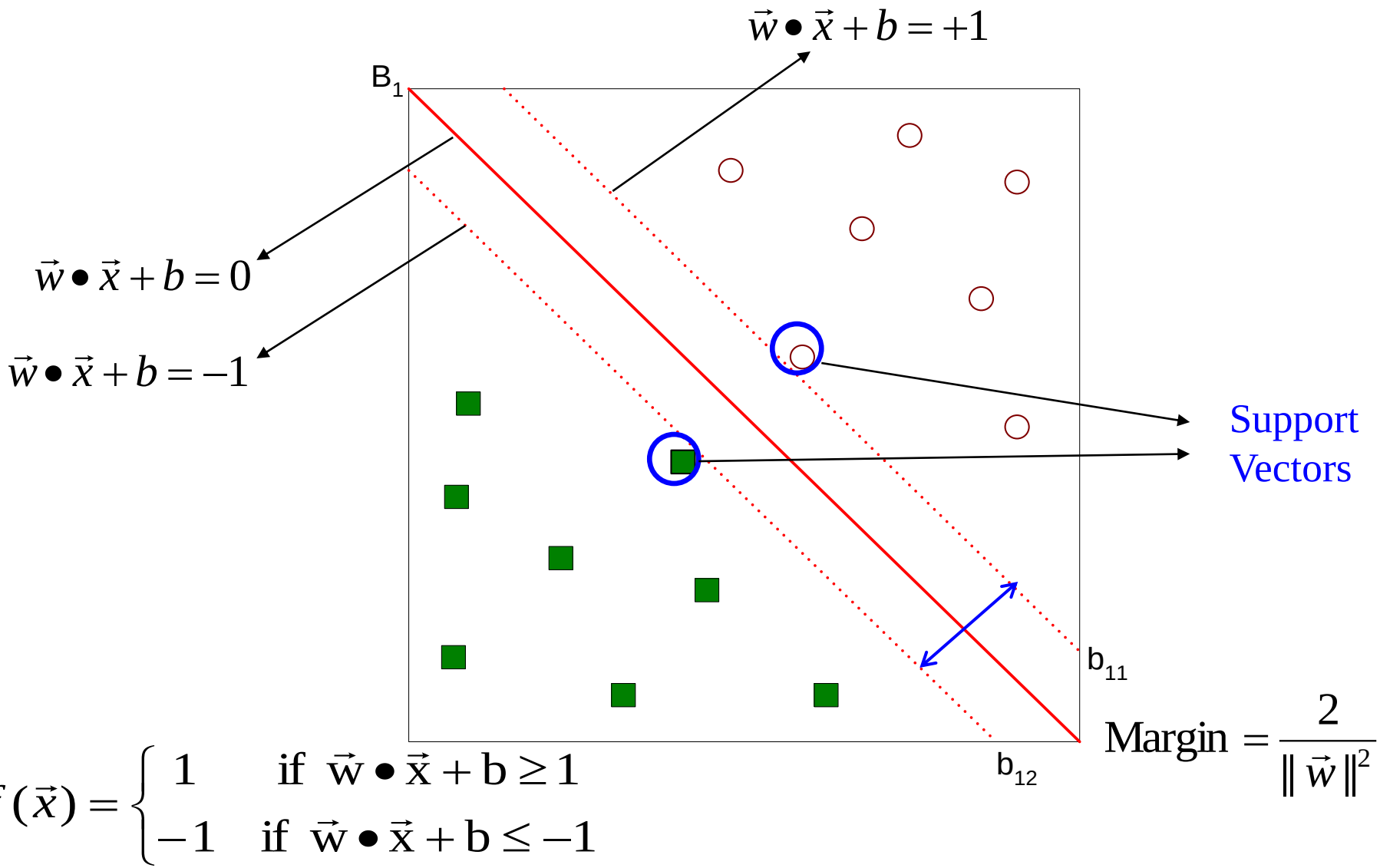
- **Which one is better? B1 or B2? How do you define better?**

Support Vector Machines



Find a hyperplane maximizing the margin => B1 is better than B2

Support Vector Machines



Support Vector Machines

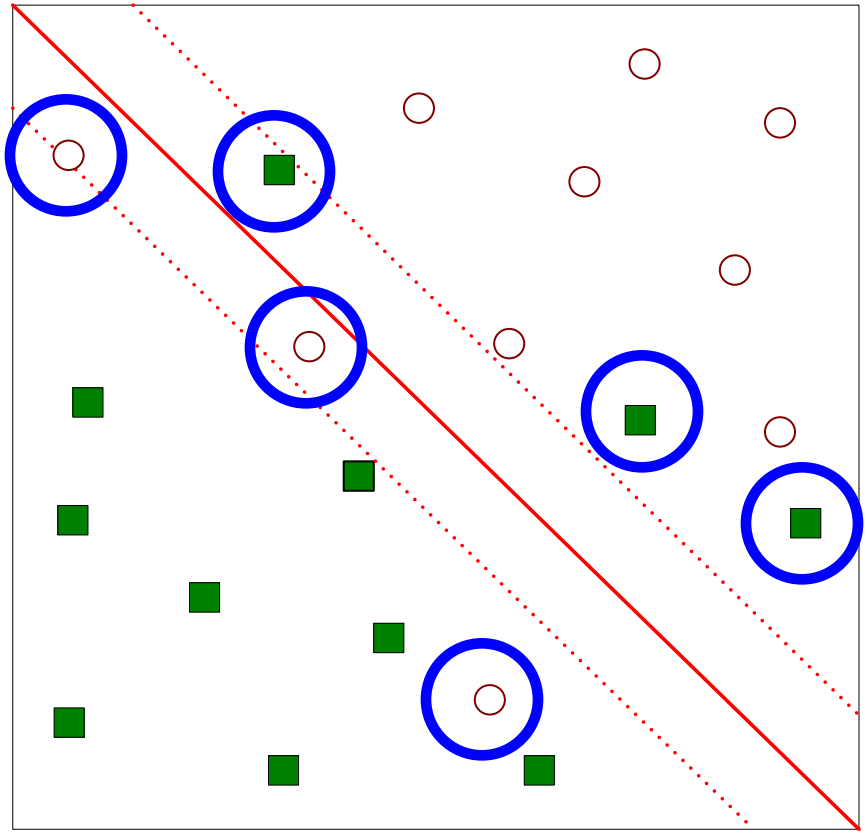
- **We want to maximize:** $\text{Margin} = \frac{2}{\| \vec{w} \|^2}$
  - Which is equivalent to minimizing: $L(w) = \frac{\| \vec{w} \|^2}{2}$
  - But subjected to the following constraints:

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

  - This is a constrained optimization problem, which can be solved by some numerical approaches, e.g., quadratic programming (QP)

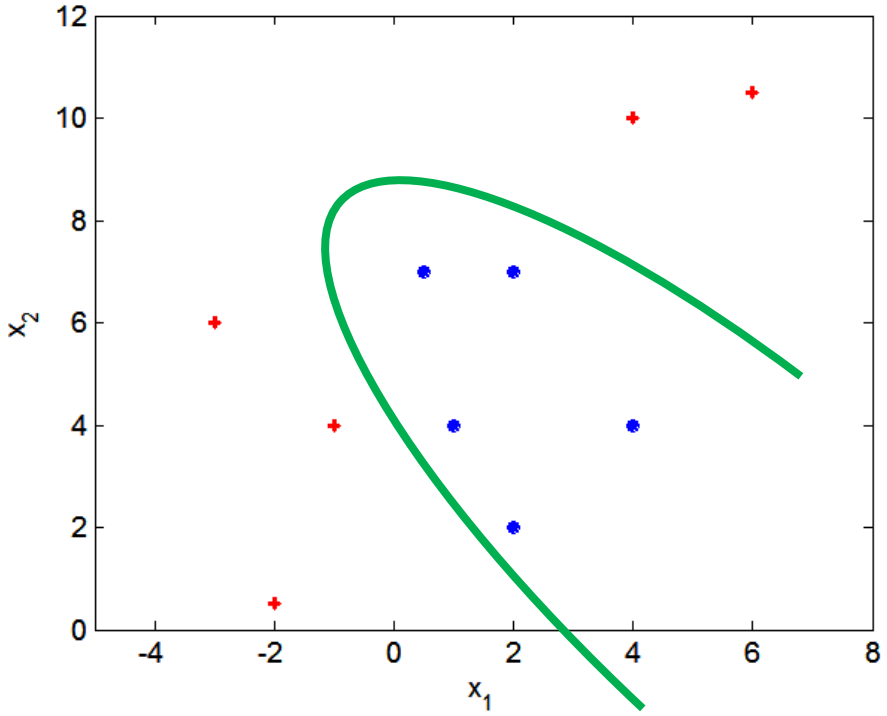
Support Vector Machines

- **What if the problem is not linearly separable?**



Nonlinear Support Vector Machin

- **What if decision boundary is not linear?**



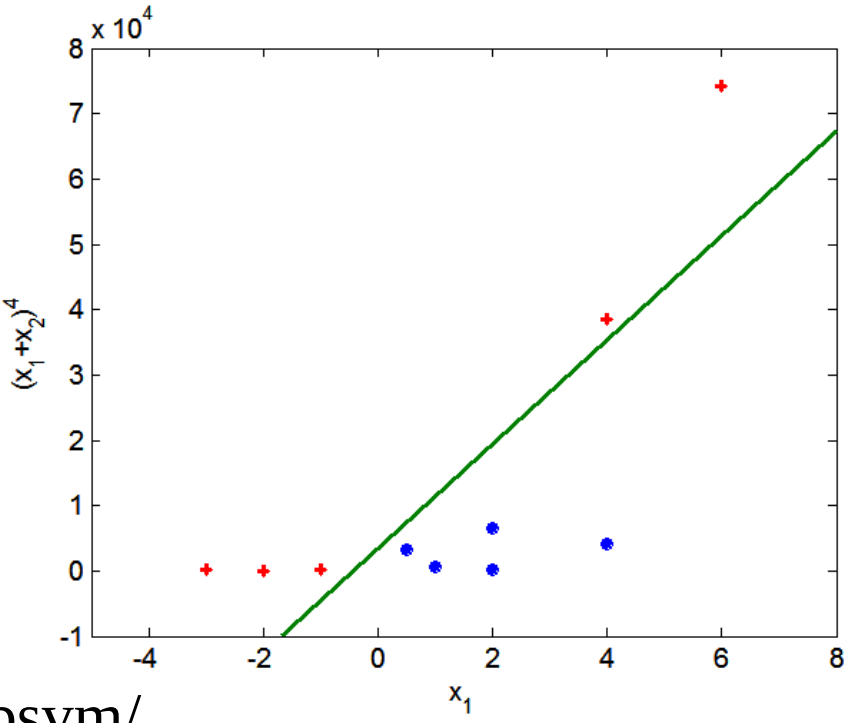
Nonlinear Support Vector Machines

- **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:**

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

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



http://svmlight.joachims.org/
http://www.csie.ntu.edu.tw/~cjlin/libsvm/

3. Classification Model Evaluation

All models are wrong, but some are useful!

- Wrong because it is a simplification of reality
- Useful if it may reach certain prediction accuracy

Model Evaluation

- **Metrics for Performance Evaluation**
  - How to evaluate the performance of a model?
- **Methods for Performance Evaluation**
  - How to obtain reliable estimates?
- **Methods for Model Comparison**
  - How to compare the relative performance among competing models?

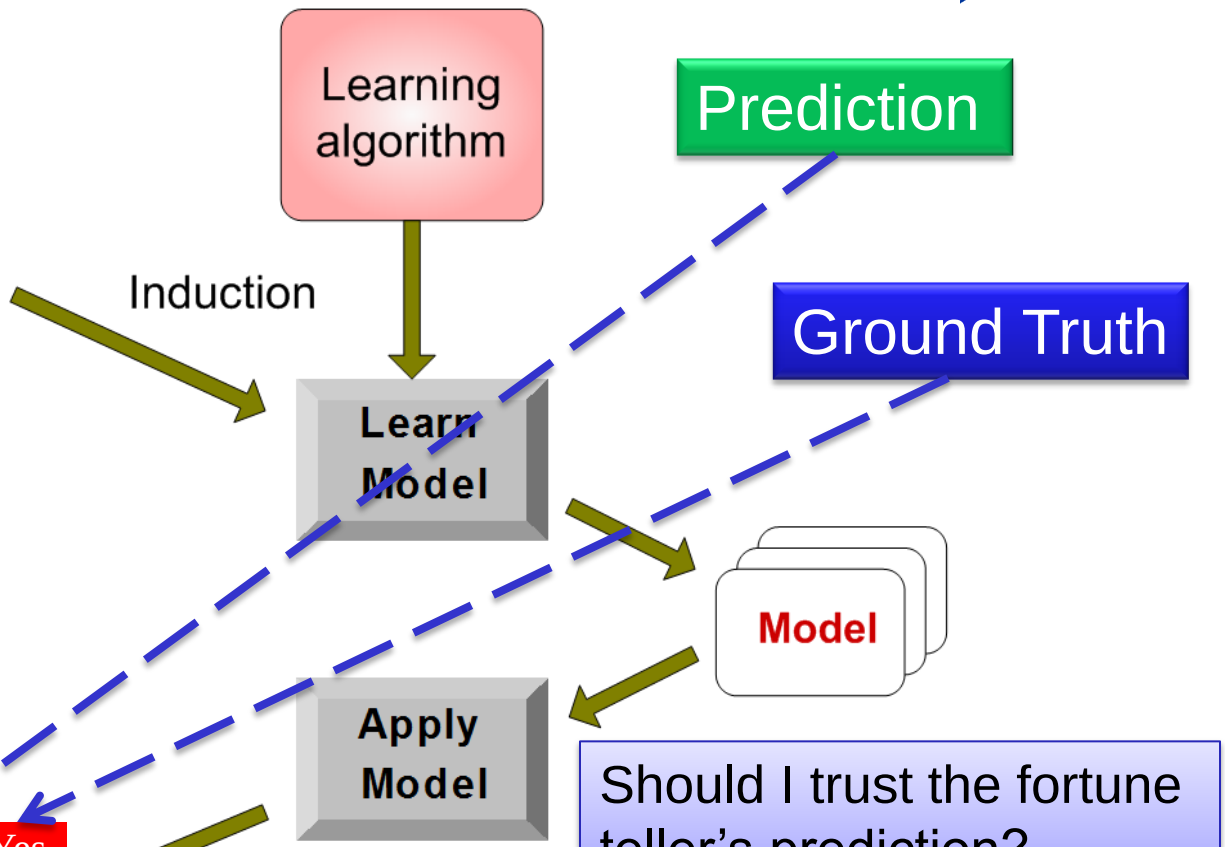
Metrics for Performance Evaluation

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class	
11	No	Small	55K	Yes	Yes
12	Yes	Medium	80K	No	No
13	Yes	Large	110K	Yes	No
14	No	Small	95K	No	Yes
15	No	Large	67K	Yes	Yes

Test Set



Should I trust the fortune teller's prediction?

I need ground truth to compare with the prediction results to do fair evaluation

Metrics for Performance Evaluation

- **Focus on the predictive capability of a model**
  - Rather than how fast it takes to classify or build models, scalability, etc.
- **Confusion Matrix (element -> #cases in test set):**

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

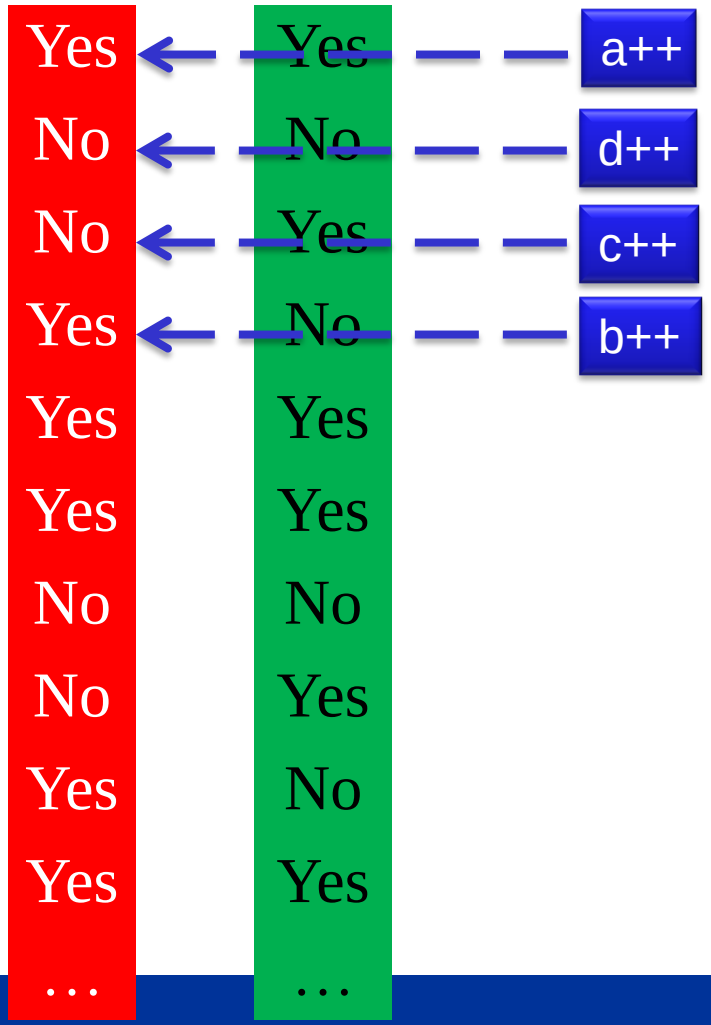
b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation

- **In the test set**



	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- **Most widely-used metric Accuracy:**

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Error Rate = 1- Accuracy**

Limitation of Accuracy

- **Consider a 2-class problem (imbalanced classification)**
  - spam detection
  - fraud detection
  - disease diagnostic
- **Usually negative class = OK class**

  positive class = not-OK class
- **Assume in the test set**
  - Number of negative examples = 9990
  - Number of positive examples = 10

Limitation of Accuracy

- **Number of negative examples = 9990**
- **Number of positive examples = 10**
- **If model predicts everything to be negative class, accuracy is 9990/(9990+10) = 99.9 %**

  (TP=0, TN=9990, FP=0, FN=10)

  – Accuracy is misleading because model does not detect any positive class example

- **In the *imbalanced* cases, accuracy is not really a reliable metric**

Cost Matrix

	PREDICTED CLASS		
ACTUAL CLASS	C(*i*|*j*)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(*i*|*j*): Cost of misclassifying class j example as class *i*

Cost/penalty means how much you need to pay if you suffer misclassification

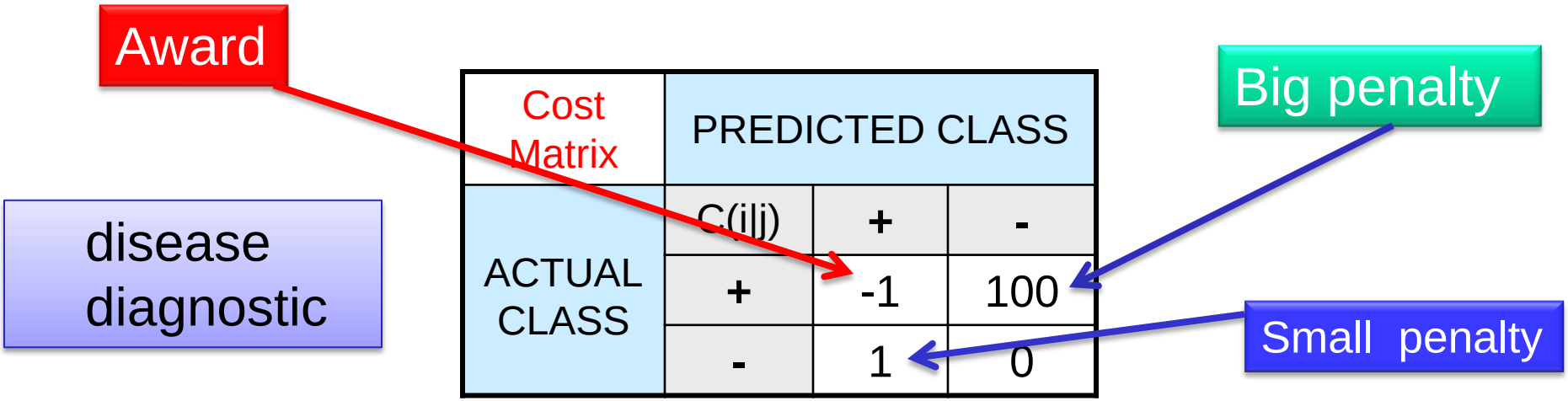
Cost Matrix in Medical Domain

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Cancer**	**Class=Normal**
	Class=Cancer	C(**Cancer|Cancer**)	C(**Normal|Cancer**) **99999?**
	Class=Normal	C(**Cancer|Normal**) **100?**	C(**Normal|Normal**)

It is **NOT acceptable** to misclassify cancer patients into normal, as it could delay the treatment

It is also **Not that acceptable** to misclassify normal patients into cancer – why?

Computing Cost of Classification



Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%
Cost = -1*150+100*40+60*1+0*250
=3910 M1 is better

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%
Cost = -1*250+100*45+1*5+0*200
=4255

Precision, Recall and F-measure

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

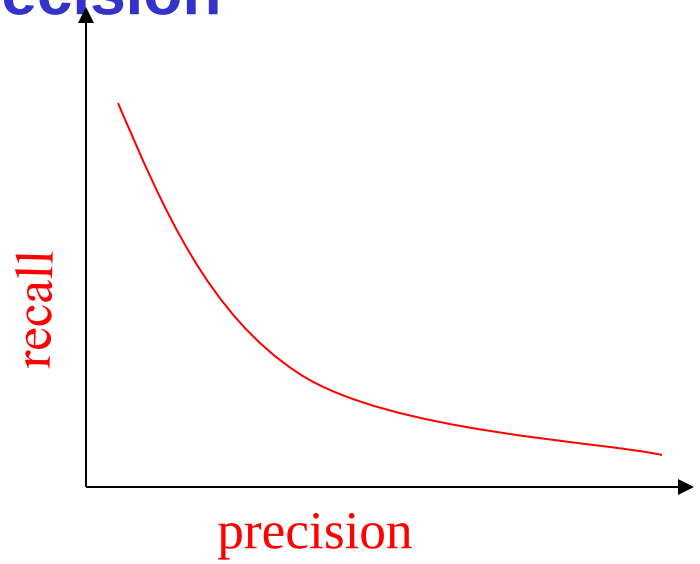
Precision: We predict *a+c* cases as positives, out of which *a* cases are correct

Recall: There are a+b positive cases, out of which *a* cases are classified as positive correctly.

What is the precision and recall (by default wrt positive/cancer class)?

Precision-Recall Trade-off

- **A predicts better than B if A has better recall and precision than B**
- **There is a trade-off between recall and precision**



- **In some apps, once you reach satisfactory precision, you optimize for recall**
- **In some apps, once you reach satisfactory recall, you optimize for precision**

Exercise: Why is there a trade off betw recall and precision?

Example of Precision, Recall and F-measure

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

		PREDICTED CLASS	
		Class=cancer	Class=Normal
ACTUAL CLASS	Class=cancer	40 (a)	60 (b)
	Class=Normal	160 (c)	5000 (d)

- **p= a/(a+c)=40/(40+160)=20%, r=a/(a+b)=40/(40+60)=40%**

- **r is also called *sensitivity* or *true positive rate* (TPR)**
- ***Specificity* or *true negative rate* =d/(c+d)=5000/(160+5000) =96.9%**

NUS National University of Singapore

Model Evaluation

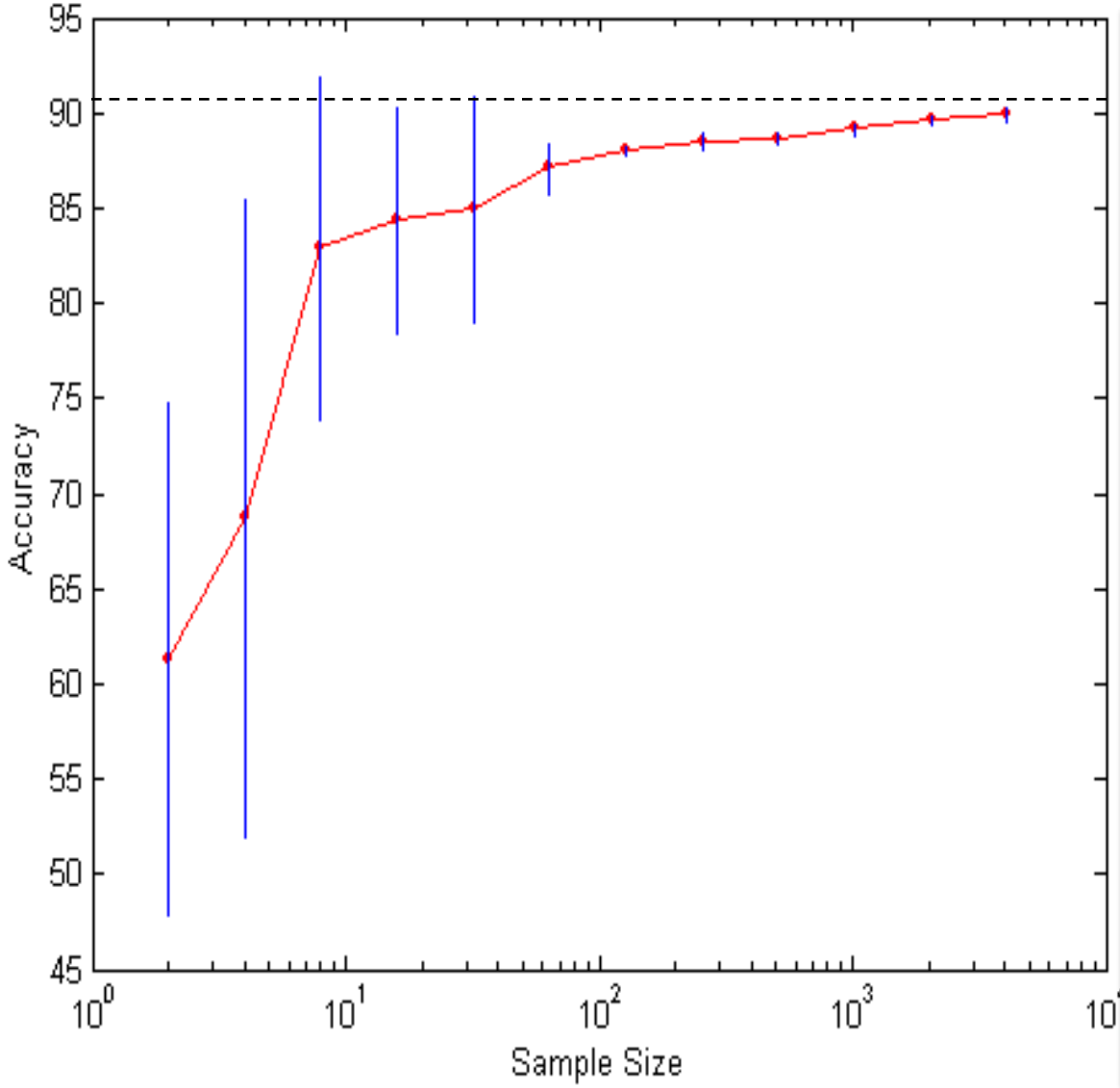
- **Metrics for Performance Evaluation**
  - How to evaluate the performance of a model?
- **Methods for Performance Evaluation**
  - How to obtain reliable estimates?
- **Methods for Model Comparison**
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- **How to obtain a reliable estimate of performance?**

- **Performance of a model may depend on other factors besides the learning algorithm:**
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al.), e.g. 10, 20, 30
  - Geometric sampling (Provost et al,), e.g. 2, 4, 8, 16, 32,…

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

Methods of Estimation

- **Holdout**
  - Reserve 2/3 for training and 1/3 for testing
- **Random subsampling**
  - Repeated holdout
- **Cross validation**
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- **Bootstrap**
  - Sampling with replacement

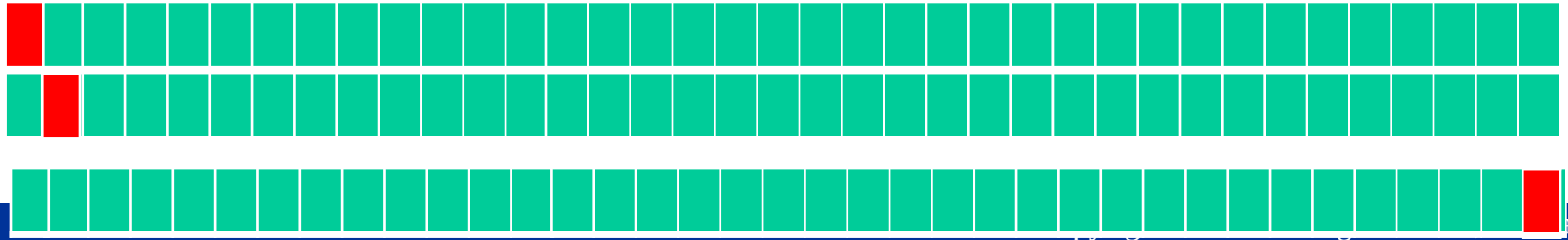
Cross Validation

5-fold cross validation

1.Test	2.Train	3.Train	4.Train	5.Train
1.Train	2.Test	3.Train	4.Train	5.Train
1.Train	2.Train	3.Test	4.Train	5.Train
1.Train	2.Train	3.Train	4.Test	5.Train
1.Train	2.Train	3.Train	4.Train	5.Test

- **Divide samples into k roughly equal disjoint parts**
- **Each part has similar proportion of samples from different classes**
- **Use each part to test other parts**
- **Average accuracy and F-measure etc**

Leave-one-out cross validation



Requirements of Biomedical Classification

- **High accuracy/sensitivity/specificity/precision**
- **High comprehensibility**

Model Evaluation

- **Metrics for Performance Evaluation**
  - How to evaluate the performance of a model?
- **Methods for Performance Evaluation**
  - How to obtain reliable estimates?
- **Methods for Model Comparison**
  - How to compare the relative performance among competing models?

All the measures that we have mentioned can be used

- **Accuracy**
- **Error Rate**
- **Precision**
- **Recall/sensitivity**
- **Specificity**
- **F-measure**
- **ROC**

ROC (Receiver Operating Characteristic)

- **Developed in 1950s for signal detection theory to analyze noisy signals**
  - Characterize the trade-off between **positive hits** and **false alarms** [You hope you can escape from fire but do not want to be disturbed by false alarm]
- **ROC curve plots TP rate (on the y-axis) against FP rate (on the x-axis)**
- **Performance of each classifier represented as a point on the ROC curve**
  - Changing the threshold of algorithm (prob) changes the location of the point

How to Construct an ROC curve

Probabilistic classifier (IR system) can generate a probability value to indicate how likely a case/record belongs to positive class

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold *t* at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

TP rate and FP rate

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	TP	FN
	Class=No	FP	TN

Ideal:
TP=1
FP=0

- TP rate, TPR = TP/(TP+FN) =Recall (pos)

  fraction of positives that I get back

- FP rate (**false alarm ratio)**, FPR = FP/(FP + TN)

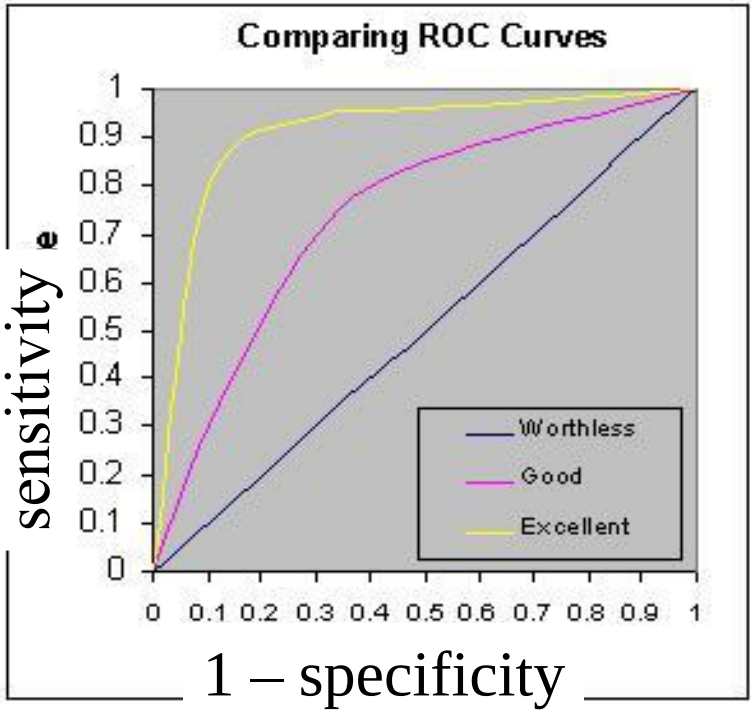
  out of all the negatives, what is the fraction of mistakes (they are classified as positives)

ROC Curves

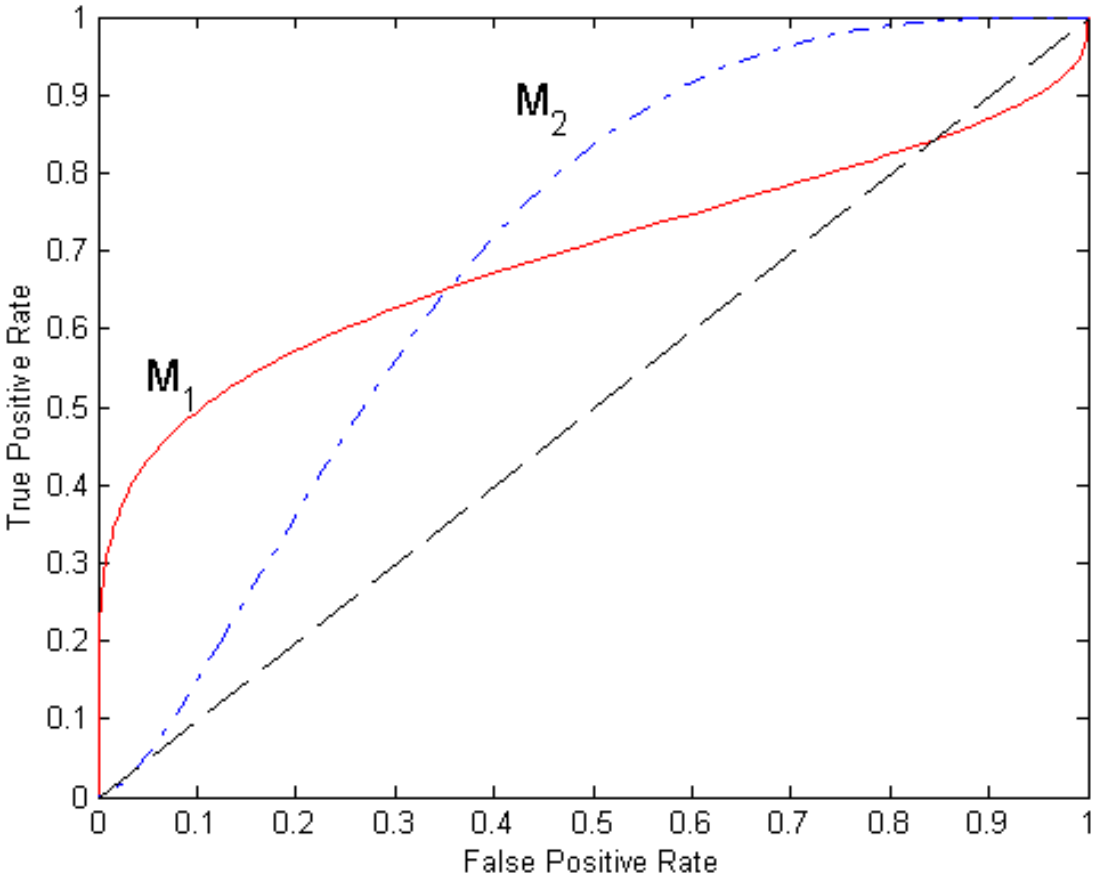
- By changing t, we get a range of sensitivities and specificities of a classifier
- A predicts better than B if A has better sensitivities than B at most specificities
- Leads to ROC curve that plots sensitivity vs. (1 – specificity)

Exercise: Draw a typical curve of sensitivity vs specificity

- Then the larger the area under the ROC curve, the better



Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

NUS National University of Singapore

Outlines

- **1. Decision Tree Ensembles**
- **2. Other Machine Learning Approaches**
  - kNN
  - NB
  - SVM
- **3. Classification Model Evaluation**
- **4. Feature Selection**

Recall kNN …

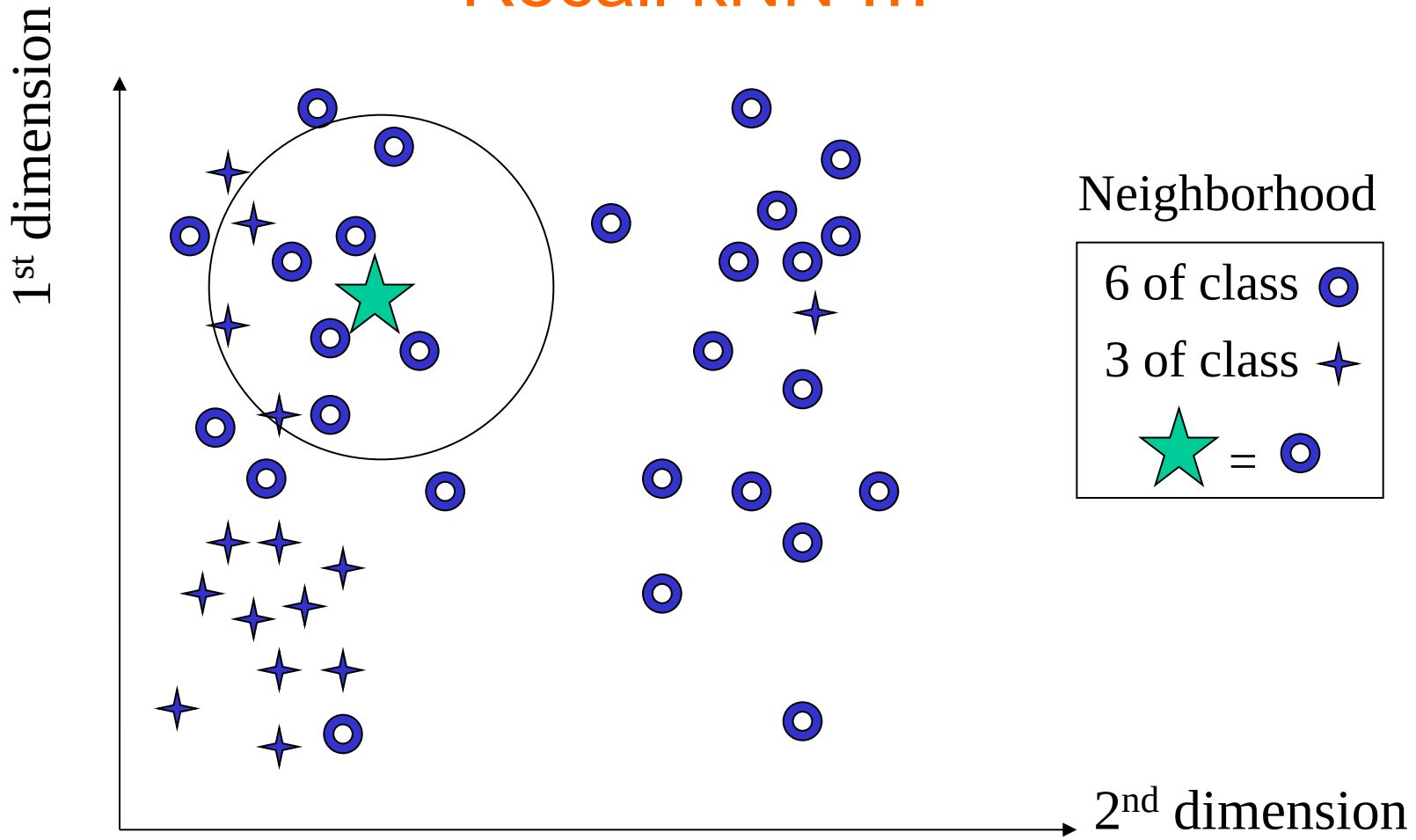
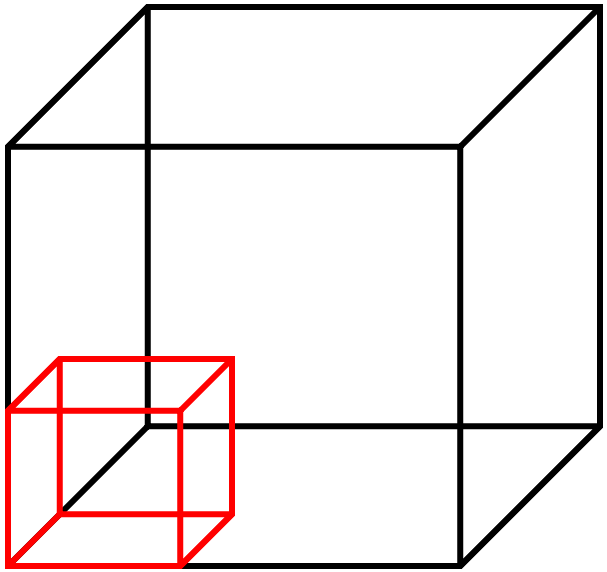


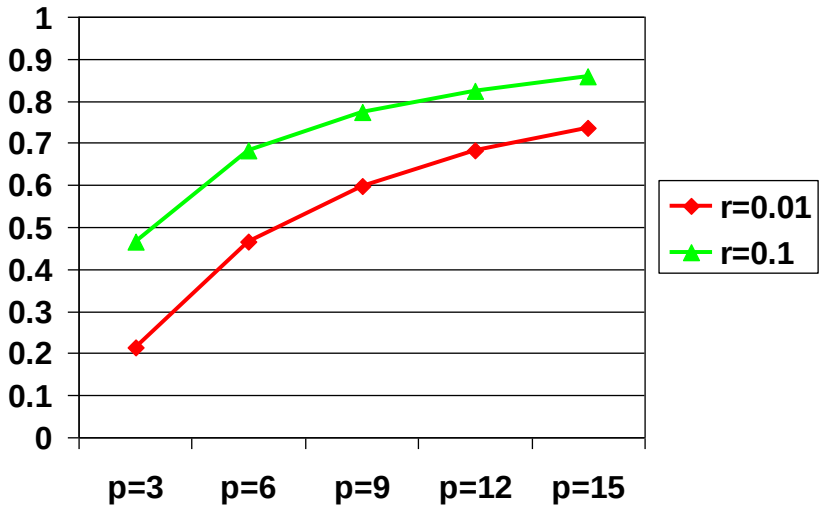
Image credit: Zaki

Curse of Dimensionality

- **How much of each dimension is needed to cover a proportion r of total sample space?**



- **Calculate by $e_p(r) = r^{1/p}$**
- **So, to cover 10% of a 15-D space, need 85% of each dimension!**



Exercise: Why $\mathbf{e_p(r) = r^{1/p}}$?

Consequence of the Curse

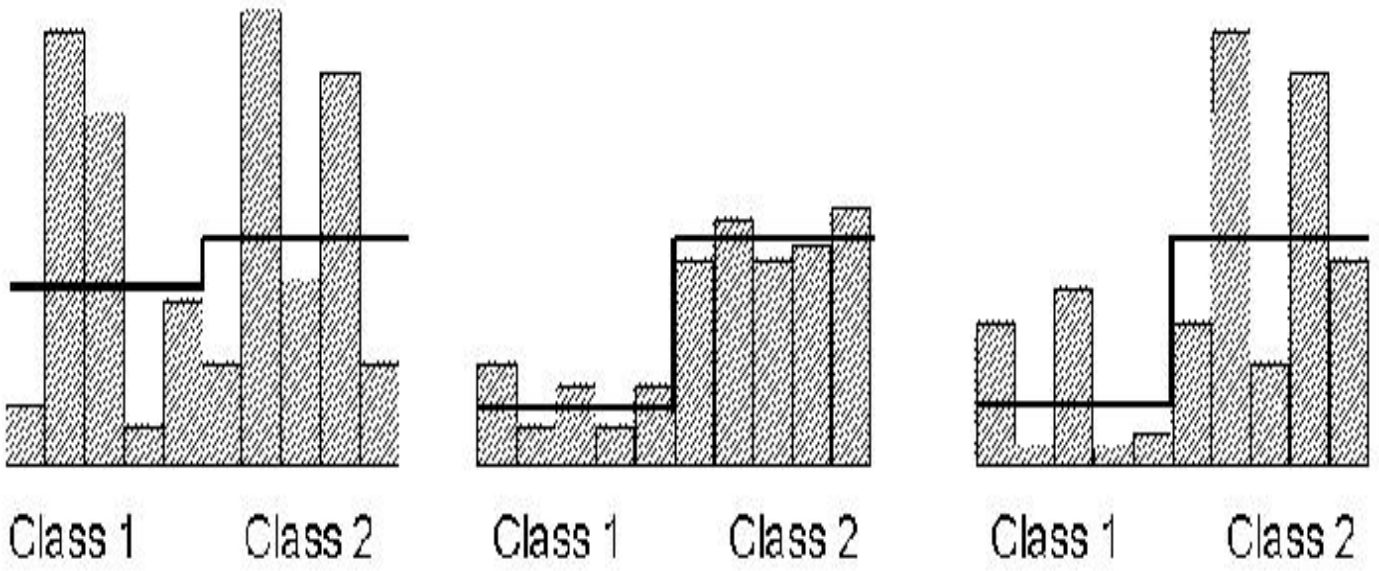
- **Suppose the number of samples given to us in the total sample space is fixed**
- **Let the dimension increase**
- **Then the distance of the k nearest neighbours of any point increases**
- **Then the k nearest neighbours are less and less useful for prediction, and can confuse the k-NN classifier**

Feature Selection

- **Given a sample space of p dimensions**
- **It is possible that some dimensions are irrelevant**
- **Need to find ways to separate those dimensions (aka features) that are relevant (aka signals) from those that are irrelevant (aka noise)**

Signal Selection (Basic Idea)

- **Choose a feature w/ low intra-class distance (variance is smaller)**
- **Choose a feature w/ high inter-class distance (mean difference is bigger)**



Signal Selection (e.g., t-statistics)

The t-stats of a signal is defined as

$$t = \frac{|\mu_1 - \mu_2|}{\sqrt{(\sigma_1^2/n_1) + (\sigma_2^2/n_2)}}$$

where σ_i^2 is the variance of that signal in class i, μ_i is the mean of that signal in class i, and n_i is the size of class i.

A feature f can be considered better than a feature f' if $t(f, C_1, C_2) > t(f', C_1, C_2)$. Thus given a collection of candidate features in samples of C_1 and C_2, we simply sort them by their t-test statistical measure, and pick those with the largest t-test statistical measures

Self-fulfilling Oracle

- Construct **artificial dataset** with 100 samples, each with 100,000 randomly generated features and **randomly assigned class labels**
- Select 20 features with the best t-statistics (or other methods)
- Evaluate accuracy by cross validation using the 20 selected features
- The resulting accuracy can be **~90%**
- But the true accuracy should be 50%, as the data were derived randomly

What Went Wrong?

- **The 20 features were selected from whole dataset**
- **Information in the held-out testing samples has thus been "leaked" to the training process**

- **The correct way is to re-select the 20 features at each fold; better still, use a totally new set of samples for testing**

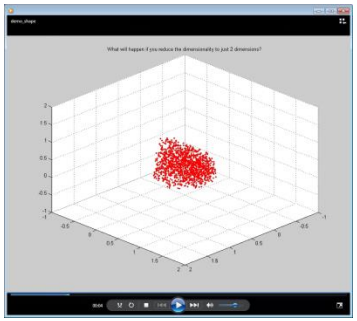
While **dimensionality reduction** is an important tool in machine learning/data mining, we must always be aware that it can *distort* the data in misleading ways.

Above is a two dimensional projection of an intrinsically three dimensional world….

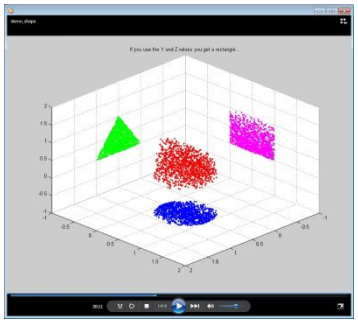
Original photographer unknown/
See also www.cs.gmu.edu/~jessica/DimReducDanger.htm

© Eamonn Keogh

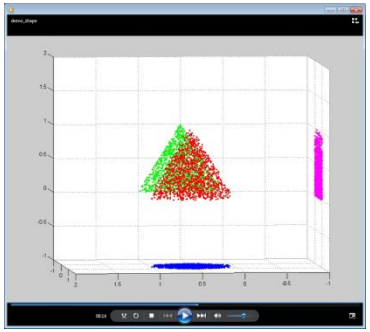
A cloud of points in 3D



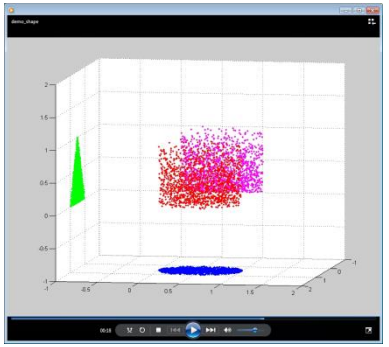
Can be projected into 2D
XY or XZ or YZ



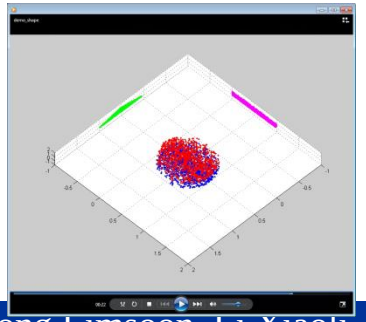
In 2D XZ we see
a triangle



In 2D YZ we see
a square



In 2D XY we see
a circle



Screen dumps of a short video from
www.cs.gmu.edu/~jessica/DimReducDanger.htm

NUS National University of Singapore

Thank You

Contact: xlli@i2r.a-star.edu.sg if you have questions

References

- L. Breiman, et al. Classification and Regression Trees. Wadsworth and Brooks, 1984
- L. Breiman, Bagging predictors, Machine Learning, 24:123-140, 1996
- L. Breiman, Random forests, Machine Learning, 45:5-32, 2001
- J. R. Quinlan, Induction of decision trees, Machine Learning, 1:81-106, 1986
- J. R. Quinlan, C4.5: Program for Machine Learning. Morgan Kaufmann, 1993
- C. Gini, Measurement of inequality of incomes, The Economic Journal, 31:124-126, 1921
- Jinyan Li et al., Data Mining Techniques for the Practical Bioinformatician, *The Practical Bioinformatician*, Chapter 3, pages 35-70, WSPC, 2004

References

- Y. Freund, et al. Experiments with a new boosting algorithm, ICML 1996, pages 148-156
- T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Machine Learning, 40:139-157, 2000
- Naïve Bayesian Classification, *Wikipedia*, http://en.wikipedia.org/wiki/Naive_Bayesian_classification
- Hidden Markov Model, Wikipedia, http://en.wikipedia.org/wiki/Hidden_Markov_model