Managing Big Data

Classification: Alternative Methods

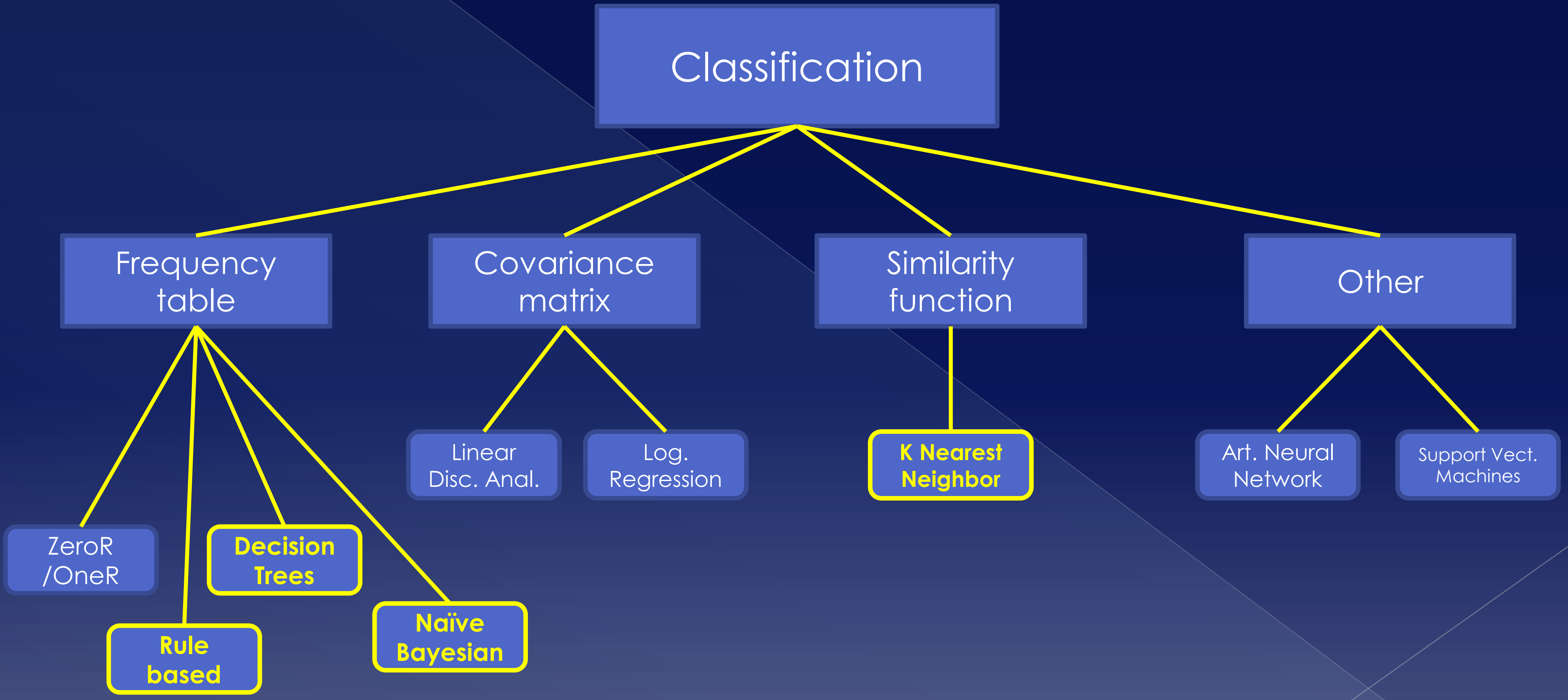
Manolis Tzagarakis
Assistant Professor
Department of Economics
University of Patras

tzagara@upatras.gr
2610 969845
google:tzagara
Facebook: tzagara
SkypeID: tzagara
QuakeLive: DeusEx

Reminder of the classification problem

- Given a collection of records **(training set)**
  - Each record contains a set of ***attributes***, one of the attributes is (always) 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 big picture of classification methods



Alternative methods

- Rule-based classifiers
- k-Nearest Neighbors (k-NN)
- Naïve Bayes

Rule-based classifiers

- **Classify records** by using a collection of **"if…then…" rules**
- General form of rule: **(*Condition*) $\rightarrow$ *y***
  - where
    - ***Condition*** is a **conjunctions (logical AND/$\wedge$)** of attributes
    - ***y*** is the **class label**
  - *LHS*: rule **antecedent or condition (Left-hand side)**
  - *RHS*: rule **consequent (Right-hand side)**
  - **Examples** of classification rules:
    - **(Blood Type=Warm) $\wedge$ (Lay Eggs=Yes)** $\rightarrow$ **Birds**

Antecedent/LHS **Consequent/RHS**

Rule-based classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

Records (also called *instances*)

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Set of rules for above dataset

Applying rule-based classifier

- Say that a rule *r* **covers an instance or record x** if the **attributes** of the instance **satisfy the condition of the rule (result TRUE)**

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Set of rules

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

Above rules cover instances (look at conditions):

The rule R1 covers a instance "hawk" => Bird (class)

The rule R3 covers the instance "grizzly bear" => Mammal (class)

Measuring "quality" of rule?

- **Assume classification rule A → y**, where A an expression of any number on conjunctions on attributes, y class attribute and a dataset D
- **Coverage of a rule:**
  - **Fraction of records** that **satisfy only the antecedent** (LHS or expression A) of a rule

$$Coverage(r) = \frac{|A|}{|D|},$$

$$|A| = number\ of\ inst.satisf.\ antecedent,$$

$$|D| = number\ of\ inst.\ in\ dataset$$

- **Accuracy of rule**
  - **Fraction of records** that **satisfy both the antecedent and consequent of a rule**

$$Accuracy(r) = \frac{|A \cap y|}{|A|},$$

$$|A \cap y| = number\ of\ inst.satisf.\ antecedent\ and\ consequent$$

Measuring "quality" of rule? Examples

- Rule: **(Status=Single) → No**
  - Coverage of rule = 4/10 = 40%
  - Accuracy of rule = 2/4 = 50%
- Rule: **(Refund=Yes) → Yes**
  - Coverage of rule = 3/10 = 30%
  - Accuracy rule = 0/3 = 0%
- Rule: **(Status=Married) ∧ (TaxInc. < 70k) → No**
  - Coverage of rule = 1/10 = 10%
  - Accuracy or fule = 1/1 = 100%
- Rule: **(Refund=No) ∧ (TaxInc. > 65K) → Yes**
  - Coverage of rule = 6/10 = 60%
  - Accuracy or rule = 3/6 = 50%

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**
10	No	Single	90K	**Yes**

Dataset, |D|=10

How do rules work?

- Rules are triggered by each instance i.e. starts the execution. Example:

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

What rules are triggered by the above data?

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

A dogfish shark triggers none of the above rules

Characteristics of rules

- **Mutually exclusive rules**
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule
- **Exhaustive rules**
  - Classifier (i.e. set of rules) has **exhaustive coverage** if it **accounts for every possible combination** of attribute values
  - Each record is covered by at least one rule

Examples of characteristics

- Example

name	Body Temp	Skin	Gives birth	Lives in water	Flys	Has legs	Hibernates	Type (class)
Lemour	warm	fur	yes	no	no	yes	yes	mammal
Turtle	cold	scales	no	sometimes	no	yes	Yes	Not mammal
Dogfish shark	cold	scales	yes	yes	no	no	no	Not mammal

R1: (Body Temp = cold) $\rightarrow$ Not mammal

R2: (Body Temp = warm) $\wedge$ (Gives birth = yes) $\rightarrow$ Mammal

R3: (Body Temp = warm) $\wedge$ (Gives birth = no) $\rightarrow$ Not mammal

This rule set is i) <u>mutual exclusive</u> and ii) <u>exhaustive</u>

Building classification rules

- Two ways
  - **Direct Method**
    - Extract rules directly from training data
    - Algorithms: RIPPER, CN2, Holte's 1R
  - **Indirect Method**
    - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
    - Algorithms: C4.5rules

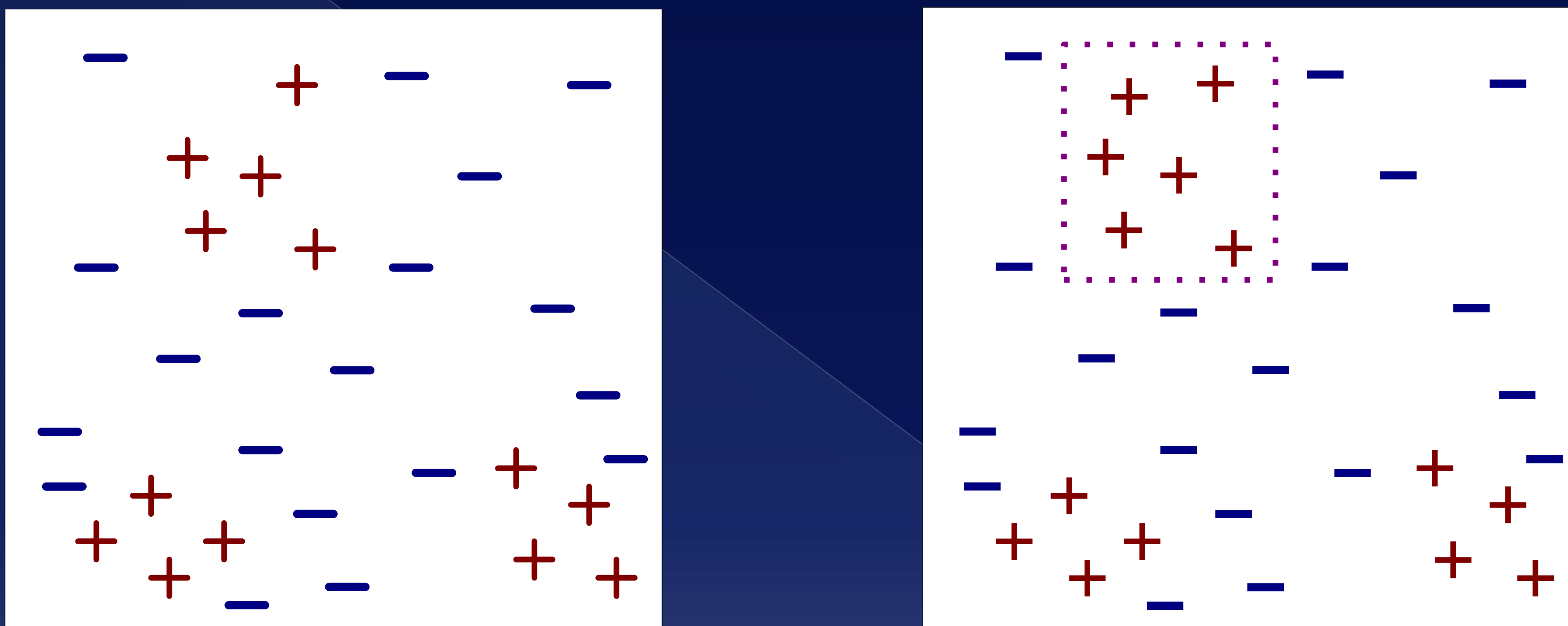
Direct method

- Sequential covering algorithm
  1. Start from an empty rule
  2. Grow a rule using **the Learn-One-Rule function**
     - Grow a rule, by **adding attributes in a greedy way**
  3. Remove training records covered by the rule
  4. Repeat Step (2) and (3) until stopping criterion is met

Direct method

- Sequential covering algorithm
  - Extracts rule for each value of class sequentially.
  - Learn-one-rule returns one new rule based on its efficiency:
    - Large positive rate
    - Small negative rate
- How are rules "grown"?

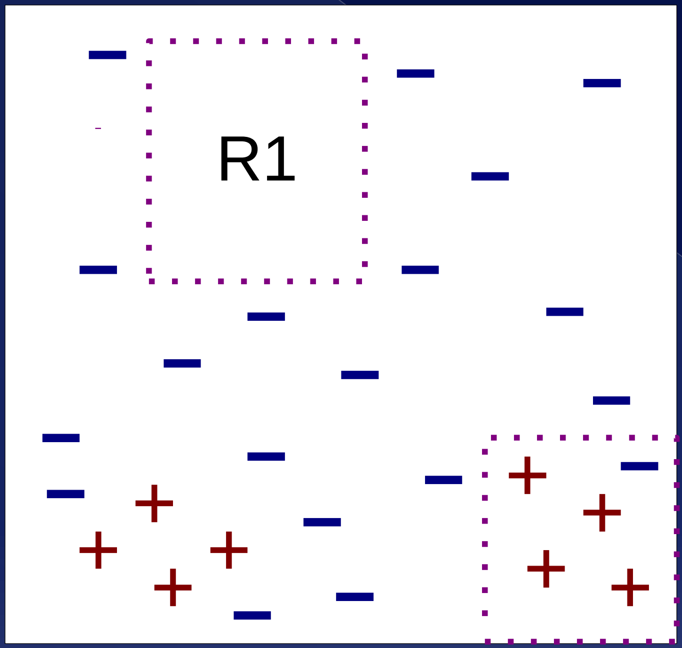
Example of Sequential Covering



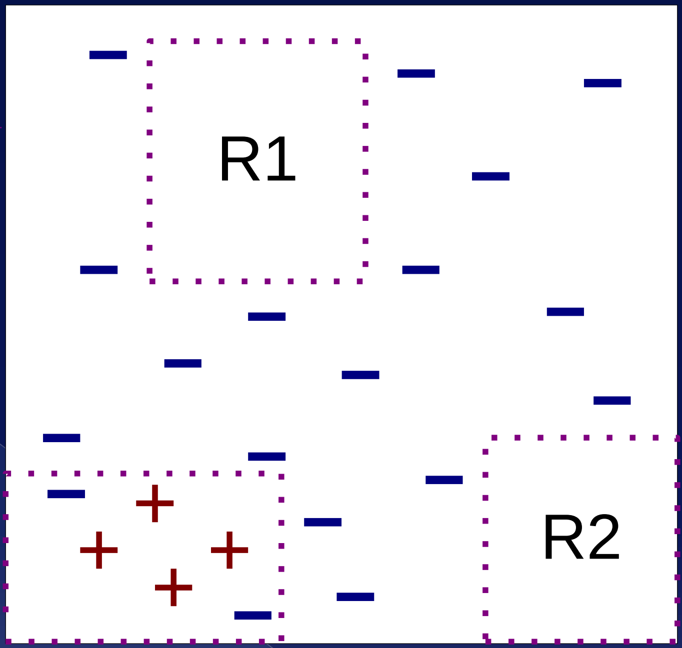
(i) Original Data (ii) Step 1

Start by finding instances with attributes/conditions that have great <u>positive</u> cover by rule. Remove these instances

Example of Sequential Covering



(iii) Step 2



(iv) Step 3

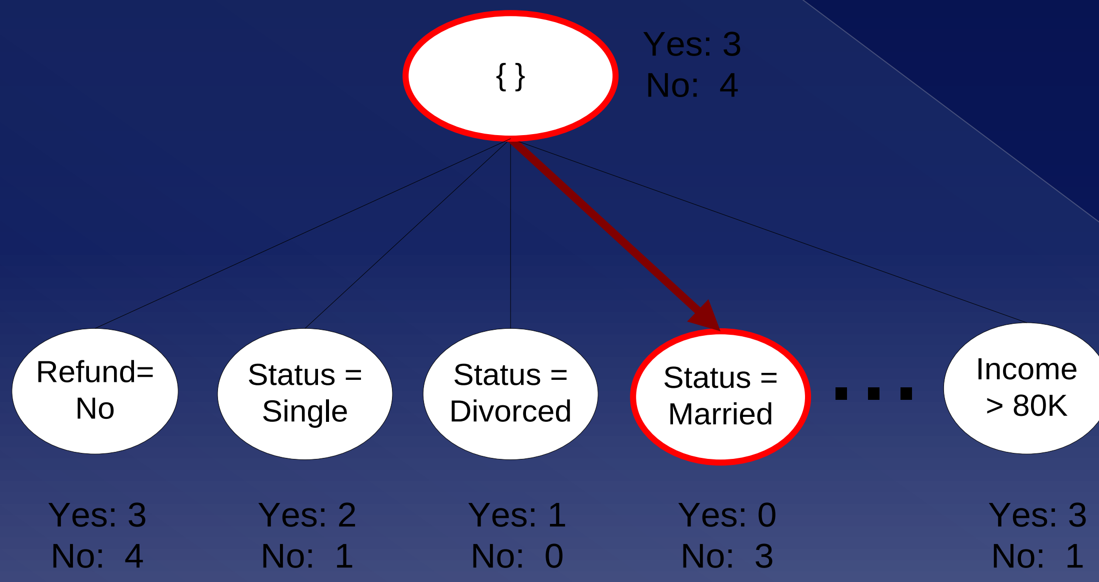
Continue finding the next rule which covers as much of the space as possible.

Strategies for growing rules

- How do rules grow i.e. get "bigger" by enclosing more conditions?
- Two Strategies
  - **General to specific**
  - **Specific to general**

Strategies for growing rules

- **General to specific**



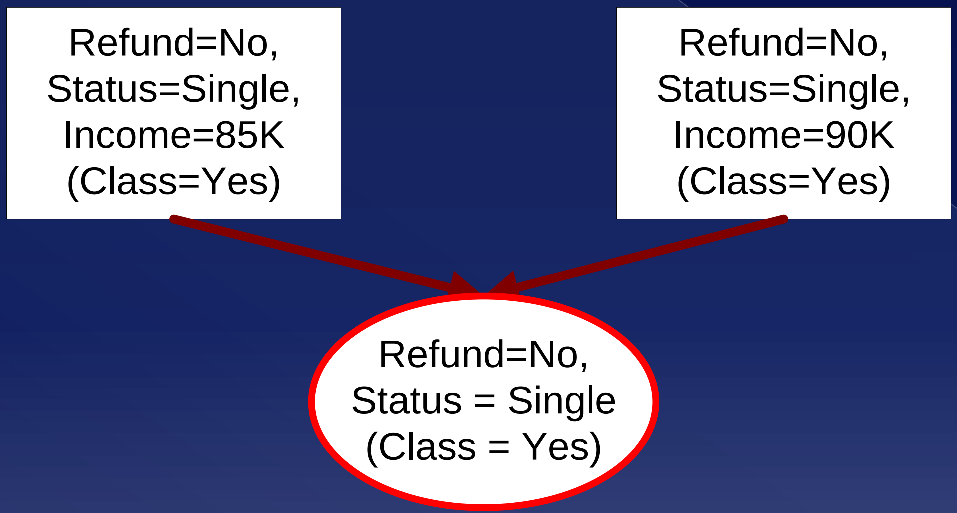
(a) General-to-specific

1. **Start with empty rule**

   $\{\} \rightarrow y$
2. **Add conjunctions** for various attributes
   - Select greedy the best attribute (calculating **gains**)
3. Continue **until gain not improved**

Strategies for growing rules

- **Specific to general**



(b) Specific-to-general

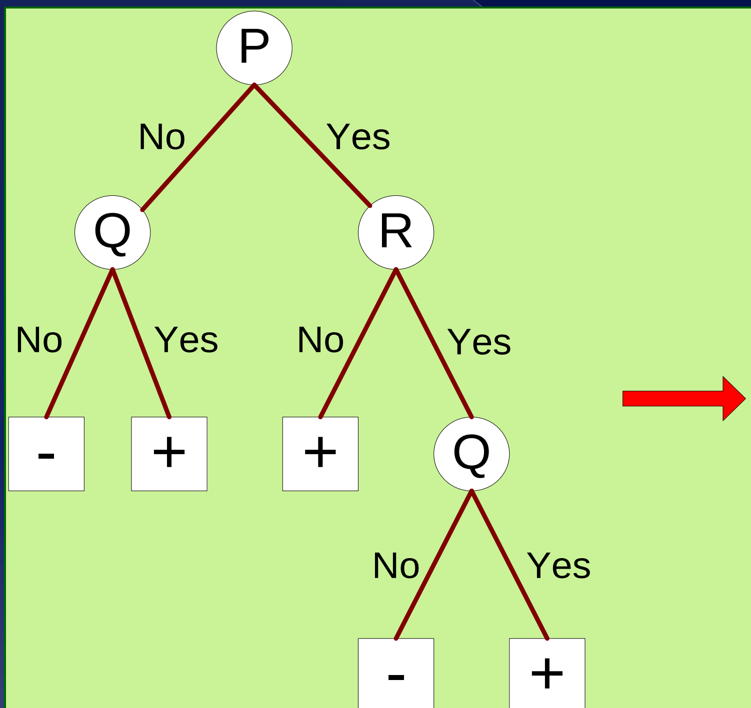
1. **Select one random instance**
2. Create rule out of instance, by forming conjunctions **from attributes of instance**
3. **Start removing conjunctions** until it starts covering negative instances

Summary of Direct Method

- Grow **a single rule**
  - Using general-to-specific or specific-to general strategies
- **Remove Instances** satisfying rule
- Prune the rule (if necessary)
- **Add rule** to Current Rule Set
- Repeat

Indirect Method

- Use Decision Tree to extract rules

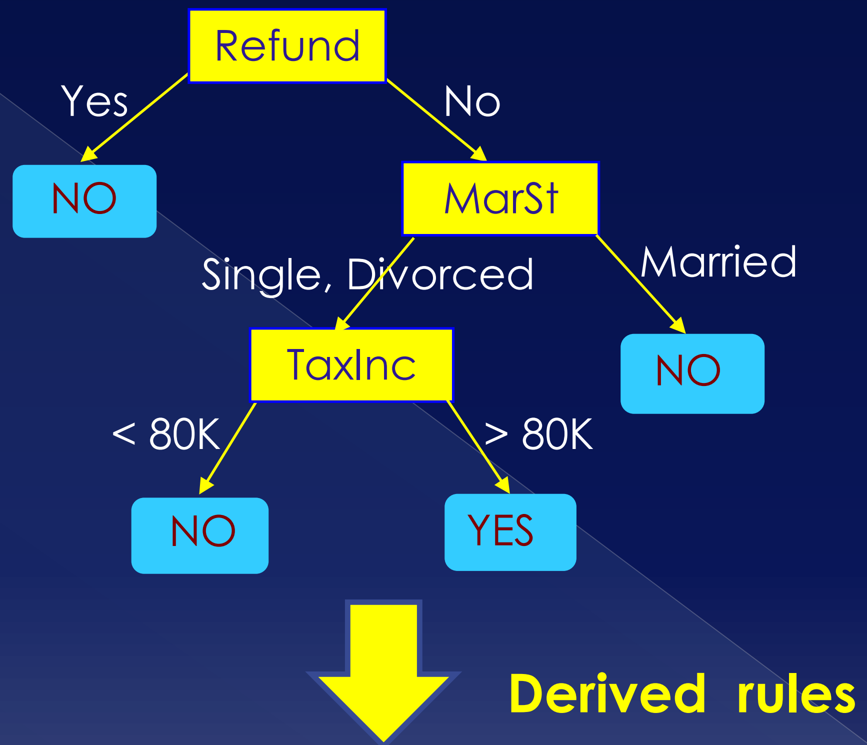


Rule Set

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

- Add rules by **forming conjunctions at each node. Braches/Paths to leaves represent one rule**

Indirect Method - Example



R1: (Refund=Yes) → No

R2: (Refund=No) ∧ (MarSt=Married) → No

R3: (Refund=No) ∧ (MarSt = Single, Divorced) ∧ (TaxInc < 80K) → Yes

....

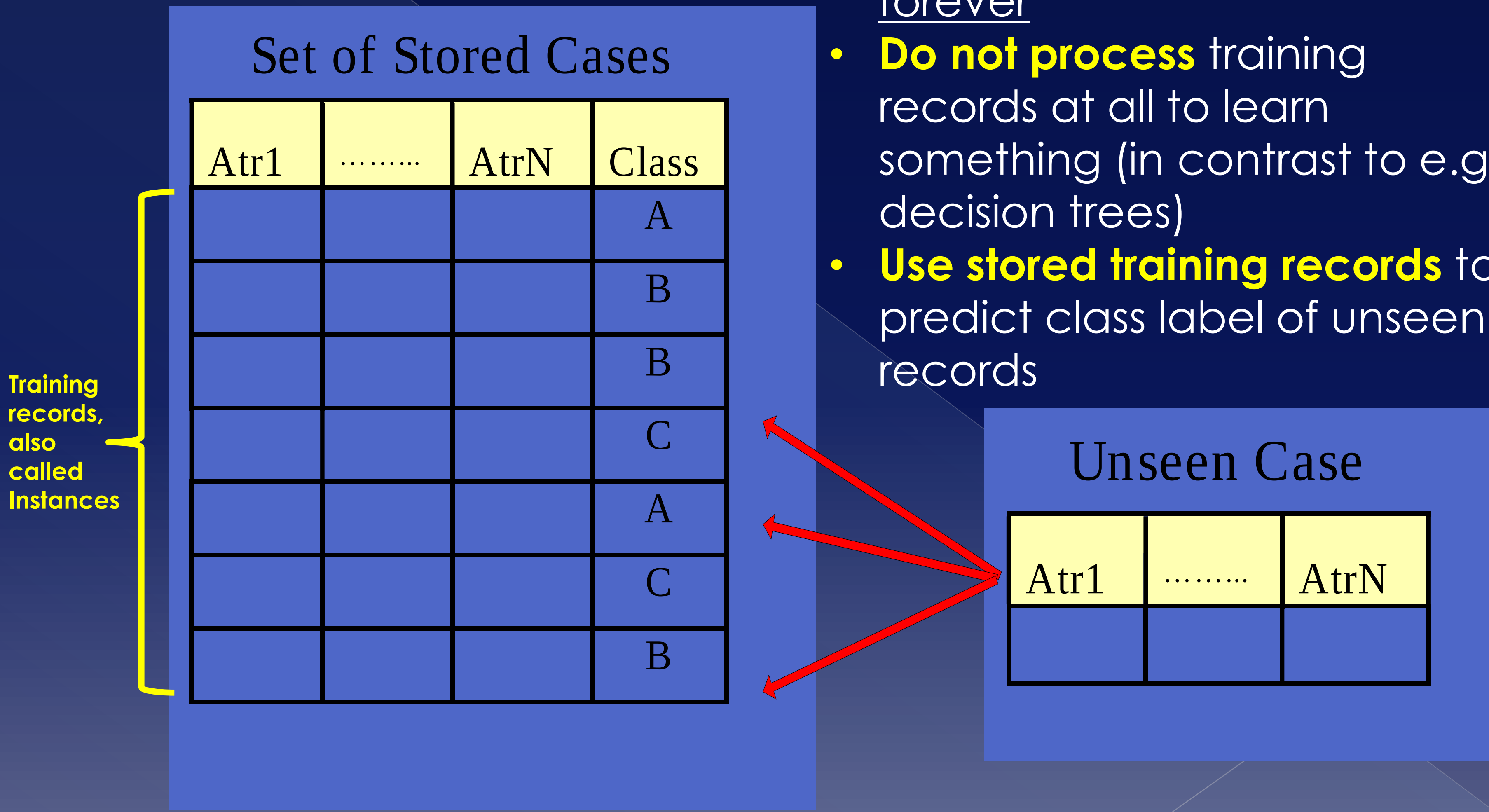
Advantages of Rule based classifiers

- As **highly expressive** as decision trees
- Easy to **interpret**
- Easy to **generate**
- Can classify new instances **rapidly**
- Performance **comparable to decision trees**

k-Nearest Neighbors

Instance-based classifiers

- **Instance based classifier?**
  - **<u>Just store</u>** the training records <u>forever</u>
  - **Do not process** training records at all to learn something (in contrast to e.g. decision trees)
  - **Use stored training records** to predict class label of unseen records

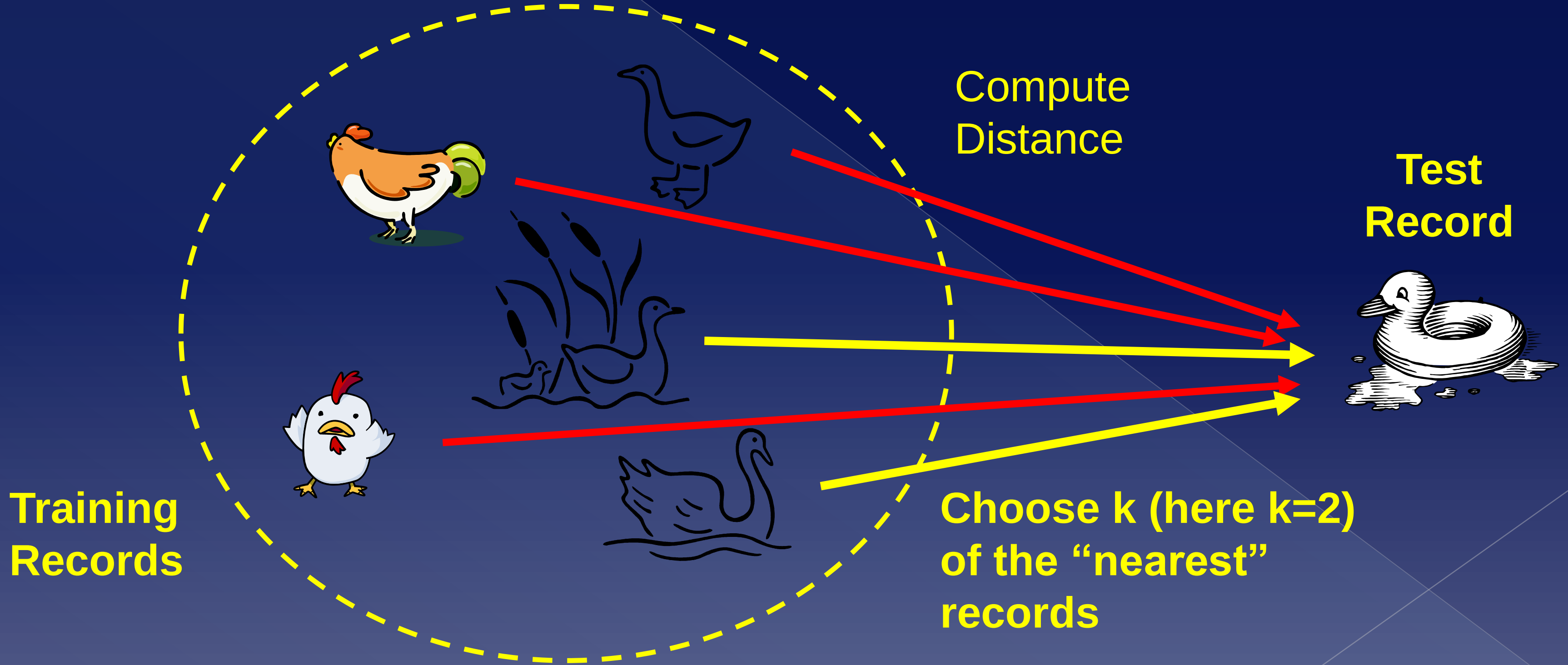


Instance-based classifiers

- Instance-based are among the simplest classifiers in existens
- Examples methods:
  - **Rote-learner**
    - **Memorizes (i.e. stores)** entire training data and performs **classification only** if attributes of record **match** one of the training examples **exactly**
  - **k-Nearest neighbor (k-NN)**
    - Uses **k "closest" points (nearest neighbors)** for performing classification
      - **For categorical class vars: majority vote** of k closest points

Nearest Neighbor Classifiers

- Basic idea of k-NN (aka duck-typing):
  - If it **walks like a duck**, **quacks like a duck**, then **it's probably a duck**



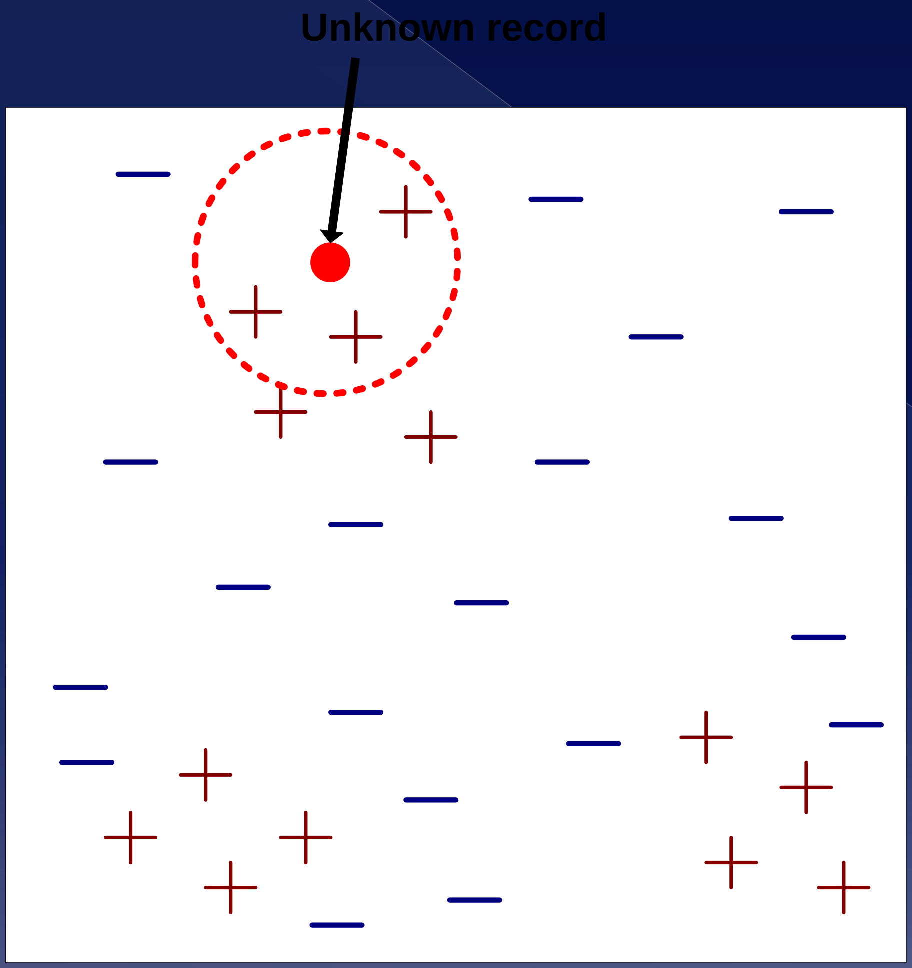
k-Nearest Neighbors Classifiers

- Outline of k-NN algorithm (quite simple)
  - **Calculate distance** between **new (unclassified) record** and **all other records in dataset (training set)**
  - Find the **k records in dataset with the closest distances to new record**
    - **k is always given, input of algorithm**
  - **Look** at the **class of these k nearest records**
  - If **majority** of the k records **belong to class C**, then **new record** is assigned **class C (one way)**
    - **k usually chosen odd**
  - k-NN is a **non-parametric method** (in contrast to decision trees)
    - **Non-parametric?** Does not care at all about he distribution of records

k-Nearest Neighbor Classifiers

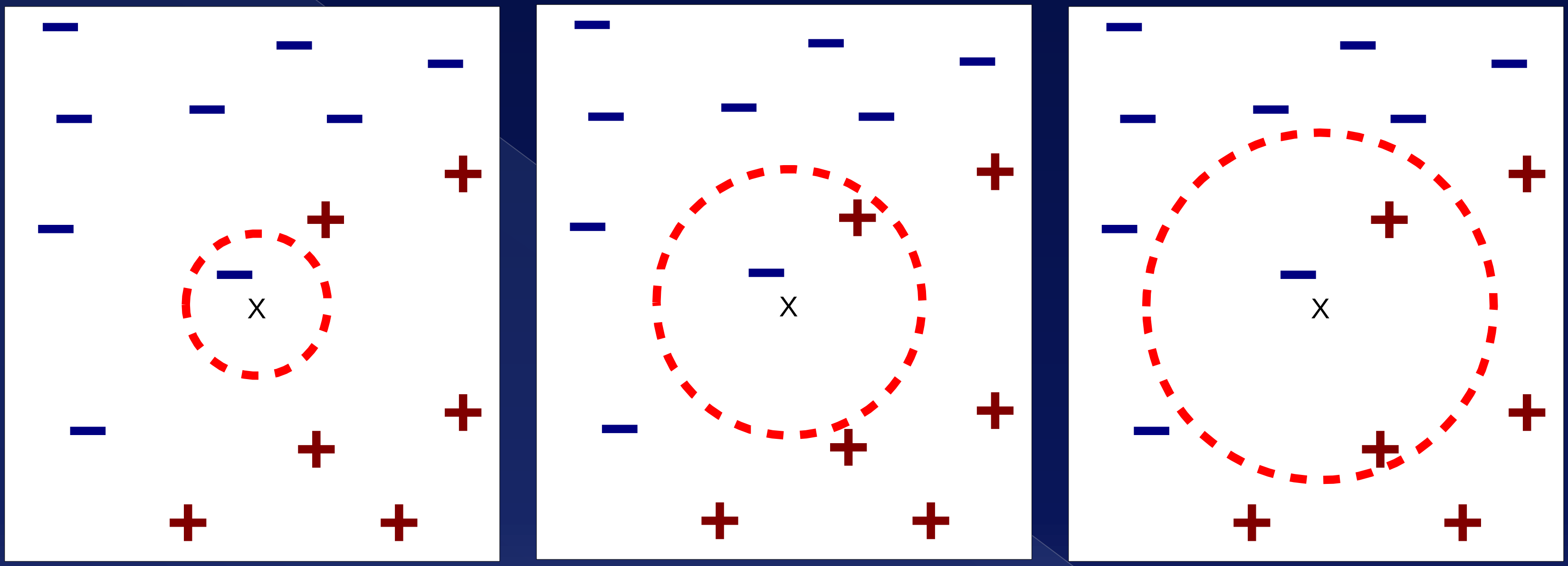
- Historical note
  - One of the **earliest methods** to classify records (and one of the simplest)
    - 1956

Nearest-Neighbor (NN) Classifiers



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

Defining Neighbors



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

k=1 **k=2** **k=3**

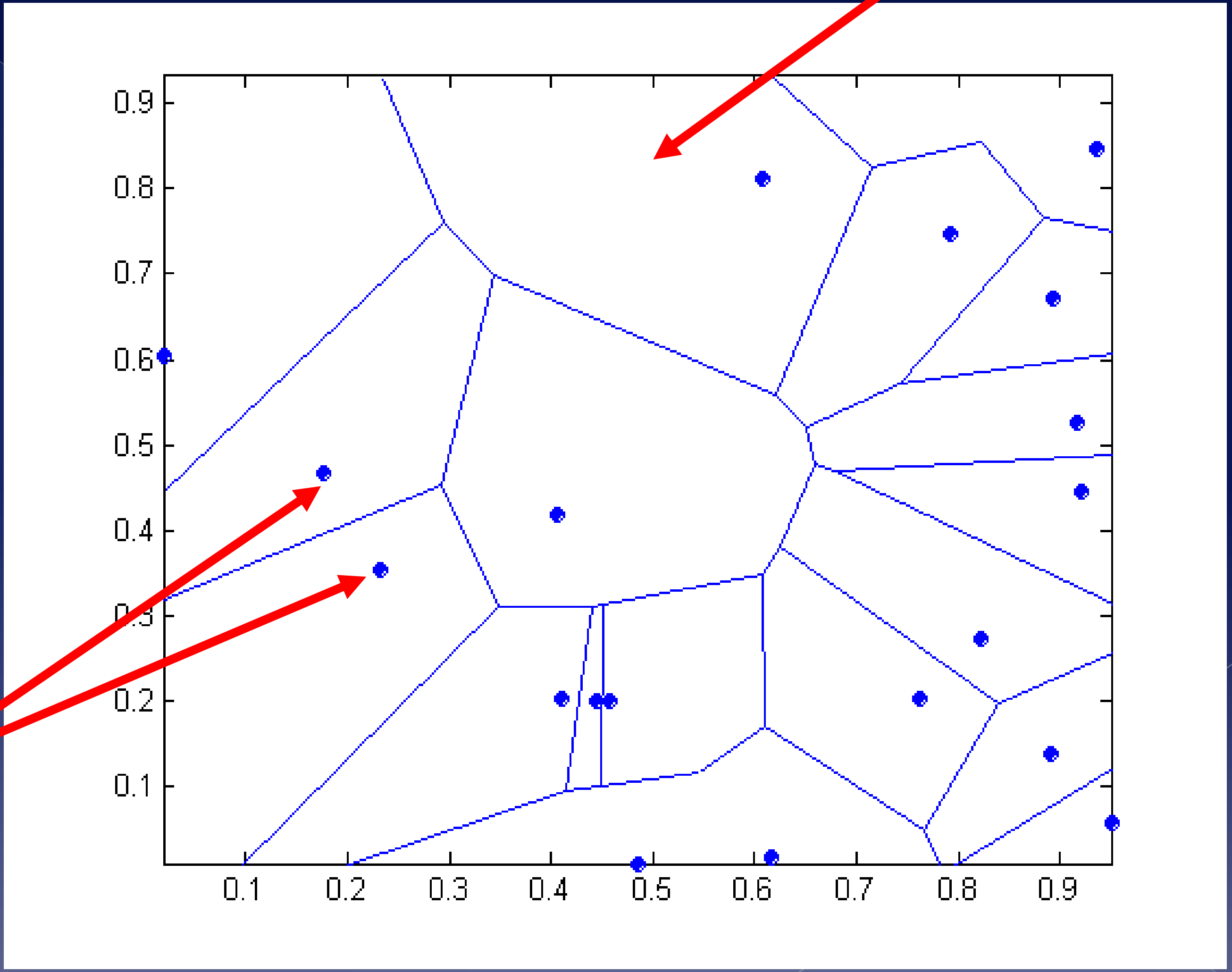
k-Nearest Neighbors of a record x are data points that have **the k smallest distance to x**

1 nearest-neighbor

If unknown record falls in this area, then class of unknown record is the class of training record defining the area

1-NN (k=1) **aka Voronoi Diagram**

In a Voronoi diagram:
-Points are training records
-Areas represent "regions" where the training record is the closest point and hence defines the class



Training records

Nearest Neighbor Classification

- Based on the **notion of distance** between records (or "points")
- How to **calculate distance between records?**
  - **Many, many ways depending on attribute types and objective**
- For **records <u>with numerical attributes</u>**, variations of the Minkowski distance:

$$d(x,y) = \sqrt[r]{\sum_{k=1}^{n} |(x_k - y_k)|^r}$$

Nearest Neighbor Classification

- If **r=1**, Minkowski distance becomes the Manhattan distance

$$d(p,q) = \sum_{i=1}^{k} |p_i - q_i|$$

… where **p, q** vectors i.e. **(p_1, p_2, p_3, …., p_k) of numericals**.

Nearest Neighbor Classification

- If **r=2**, Minkowski distance becomes the **Euclidean distance**

$$\mathbf{d(p,q)} = \sqrt{\sum_{\mathbf{i=k}}^{\mathbf{k}} (\mathbf{p_i} - \mathbf{q_i})^2}$$

… where **p, q** vectors i.e. **(p_1, p_2, p_3, …., p_k) of numericals**.

Nearest Neighbor Classification

- **3 most popular distance measures** in use for numerical records in k-NN:
  1. **Euclidean** distance
  2. **Manhattan** or **City block** distance
  3. **Supremum or L_∞ norm**, i.e. Minkowski distance when $r \to \infty$. What does that mean?
     - **As $r \to \infty$**, the Minkowski formula is **dominated by the term with the biggest difference**
     - A fancy way of saying: "*ignore all other attributes/dimensions except the ones with the biggest difference*".

Nearest Neighbor Classification

- Problem with Euclidean measure:
  - High dimensional data
    - **curse of dimensionality**
  - Can produce **counter-intuitive results**

1 1 1 1 1 1 1 1 1 1 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		**d = 1.4142**

- **Solution: Normalize the vectors to unit length**

Nearest Neighbor Classification

- Curse of dimensionality
  - In general when large number of attributes present, **all records become close i.e. distances shrink**.

Nearest Neighbor Classification

- What about records with **nominal attributes?**
  - E.g. (blue, Spaghetti, skirt)
  - Compute **Overlap (or Hamming) measure**:

$$\mathbf{d}(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{m} \mathbf{d}(\mathbf{p}_i,\mathbf{q}_i)$$

… where **p, q** vectors with nominal values
$d(p_i, q_i) = 1$ if $p_i <> q_i$ and **$d(p_i, q_i)=0$ if $p_i = q_i$**
i.e. distance is 1 if they differ and 0 otherwise

Nearest Neighbor Classification – Overlap measure example

1. (a, b, a, c) → A
2. (a, b, c, a) → B
3. (b, a, c, c) → C
4. (c, a, b, c) → A

Class!

Training records/dataset – stored permanently

5. (a, b, b, a)

New record. Class?

- Overlap distances
  - d(1,5) = d(a,a) + d(b,b) + d(a,b) + d(c,a) = 0+0+1+1 **= 2**
  - d(2,5) = 0+0+1+0 **= 1**
  - d(3,5) = 1+1+1+1 **= 4**
  - d(3,5) = 1+1+0+1 **= 3**
- Classification of new data? Based on overlap (Hamming) distances:
  - If k=1, then new record belongs to **class B**
  - If k=2, then new record belongs to **classes A or B (?)**
  - If k=3, then new record belongs to **class A**
  - If k=4, then new record belongs to **class A**

Nearest Neighbor Classification

- Overlap (Hamming) distance **not the only way** to measure distance **for nominal attributes**
- More clever ways to calculate distances of nominal attributes, which give much better results
  - For example **Value Difference Measure VDM**

Nearest Neighbor Classification

- Value Difference Measure
  - Takes into consideration the class(!)
  - Gives much better results than the Hamming distance
  - Defined as:

$$d(p,q) = \sum_{i=1}^{n} |P(c_i|p) - P(c_i|q)|^n$$

… where **p, q nominal <u>values</u>**, **n=number of classes**, **$P(c_i|p)$ = probability of class c_i given the presence of value p (Bayesian prob.)**

Note: The **Modified Value Difference Measure** does **<u>not raise to power n</u>** the difference.

Nearest Neighbor Classification

- Example of VDM
  - Assume cars sold in various colors: **red**, **green**, **blue** etc
  - For Overlap measure, difference between **red and green** is the same as the difference **red and blue**
  - However, **red and blue cars may sell more** than green cars which somehow implies that **red is closer to blue than to green**
    - VDM aims to capture this
    - Measures always depend on the objective!

Nearest Neighbor Classification

- How to calculate **distance** when vectors have **numerical and nominal attributes** (mixed data types) ?
  - e.g. **(43, blue, Married, 50901, Good, Peking Duck, 0.87)** ?
  - **Not always an easy task!**
  - **Find yourself** one measure that makes sense
    - You can come up with your own **that makes sense for your context**
    - Although there are many available e.g. Gower's General Similarity Coefficient
  - You may also **combine different measures into a single measure. E.g.**
    - Euclidean distance for numerical attributes
    - Other measure for nominal values
    - Combine (somehow) the above two

Nearest Neighbor Classification

- In general
  - **You design** your own distance measure function
  - Put your knowledge of the domain in
  - **Reason about what makes things similar and what not**
    - Depends on the domain, objective etc

Nearest Neighbor Classification

- Aren't there **any properties that the distance function must have**? Yes there are
- A metric or distance function $D(\cdot, \cdot)$ for all points x, y and z, must satisfy the following properties:
  - **Nonnegativity**: $D(x, y) \geq 0$
  - **Reflexivity:** $D(x, y) = 0$ if and only if $x = y$
  - **Symmetry**: $D(x, y) = D(y, x)$
  - **Triangle inequality:** $D(x,y)+D(y,z) \geq D(x,z)$

Example of k-nn: ordinal and nominal values

Training records (stored)

- Credit risk data

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Class

- How would Maria who is **single, high-income earner, and low in debt be classified**?
  - Record for Maria: **(Low, High, No)**

Example of k-nn: ordinal and nominal values

- Assume **k=3**
- Proper distance metric?
  - Using Overlap (Hamming): 0=same, 1=different and sum

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 1.

Calculate distance of Maria from all other records of the training set.

d(Maria, Joe) = 1 + 0 + 1 = 2
d(Maria, Amber) = 0 + 0 + 1 = 1
d(Maria, Harry) = 0 + 0 + 0 = 0
d(Maria, Lindsay) = 1 + 1 + 1 = 3
d(Maria, Kaley) = 0 + 1 + 0 = 1

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using Overlap (Hamming): 0=same, 1=different and sum

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 2.

Sort distances in ascending order

d(Maria, Harry) = 0
d(Maria, Amber) = 1
d(Maria, Kaley) = 1
d(Maria, Joe) = 2
d(Maria, Lindsay) = 3

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using Overlap (Hamming): 0=same, 1=different and sum

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 3.

Keep the k closest records to Maria (here, k=3)

d(Maria, Harry) = 0
d(Maria, Amber) = 1
d(Maria, Kaley) = 1

→ The **k-neighborhood of Maria**

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using Overlap (Hamming): 0=same, 1=different and sum

Name	**Debt?**	**Income?**	**Married?**	**Risk**
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 4.

Look at the class of the 3 closest record to Maria

d(Maria, Harry) → Poor (=class of Harry)
d(Maria, Amber) → Good (=class of Amber)
d(Maria, Kaley) → Poor (=class of Kaley)

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using Overlap (Hamming): 0=same, 1=different and sum

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, POOR)**

Step 5.

Assign to Maria the most frequent class (majority vote)

d(Maria, Harry) → Poor
d(Maria, Amber) → Good
d(Maria, Kaley) → Poor

➡ **Class of Maria: POOR**

Example MVDM

- Calculate distances using the **Modified** value difference metric (MVDM)

Class

Tid	Refund	Marital Status	Taxable Income	Cheat
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

d(Single, Married) = | P(Yes|Single) – P(Yes|Married) | + |P(No|Single) – P(No|Married) | = | (2/10)/(4/10) – (0/10)/(4/10)| + |(2/10)/(4/10) – (4/10)/(4/10)| = **1**

d(Single, Divorces) = | 2/4 – 1/2 | + | 2/4 – 1/2 | = **0**

d(Married, Divorced) = | 0/4 – 1/2 | + | 4/4 – 1/2 | = **1**

d(Refund=Yes, Refund=No) = | 0/3 – 3/7 | + | 3/3 – 4/7 | = ***6/7***

Example of k-nn: ordinal and nominal values

Training records (stored)

- Credit risk data

Name	Debt?	Income?	Married?	Class Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

- How would Maria who is **single, high-income earner, and low in debt be classified**? **NOTE: Using MVDM**
  - Record for Maria: **(Low, High, No)**

Example of k-nn: ordinal and nominal values

- Assume **k=3**
- Distance metric?
  - Using **MVDM**

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Step 1.

Calculate distance of Maria from all other records of the training set using MVDM. Hence calculate first differences between nominal values of all attributes.

Maria: **(Low, High, No, ????)**

Dept?: d(High, Low) = |P(Good|High) – P(Good|Low) | + |P(Poor|High) – P(Poor|Low)| = |1/2 – 1/3| + |1/2 – 2/3| = 0.33333

Incomme? d(High, Low) = |P(Good|High) – P(Good|Low)| + |P(Poor|High) – P(Poor|Low)| = |(2/5)-(0/5)| + |(1/5)-(2/5)| = 0.6

Married?d(Yes, No) = |P(Good|Yes) – P(Good|No)| + |P(Poor|Yes) – P(Poor|No)| = |(1/2) – 0| + ||(1/2) – 1 | = 1

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using **MVDM**

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 2.

Calculate actual distances. Based on previous slide

d(Maria, Harry) = d(Low, Low) + d(High, High) + d(No, No) = 0
d(Maria, Amber) = d(Low,Low) + d(High, High) + d(No,Yes) = 0 + 0 + 1 = 1
d(Maria, Kaley) = d(Low, Low) + d(High,Low) + d(No,Yes) = 0 + 0.6 + 1 = 1.6
d(Maria, Joe) = d(Low,High) + d(High,High) + d(No,Yes) = 0.333 + 0 + 1 = 1.333
d(Maria, Lindsay) = d(Low,High) + d(High,Low) + d(No,Yes) = 0.333 + 0.6 + 1 = 1.933

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using **MVDM**

Name	**Debt?**	**Income?**	**Married?**	**Risk**
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 3.

Sort distances in ascending order

d(Maria, Harry) = 0
d(Maria, Amber) = 1
d(Maria, Joe) = = 1.333
d(Maria, Kaley) = 1.6
d(Maria, Lindsay) = 1.933

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using **MVDM**

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 4.

Keep the k (=3) closest records to Maria

d(Maria, Harry) = 0
d(Maria, Amber) = 1
d(Maria, Joe) = = 1.333

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using **MVDM**

Name	Debt?	Income?	Married?	Risk
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, ????)**

Step 5.

Get class of the k (=3) closest records to Maria

d(Maria, Harry) = 0 → Poor (=class of Harry)
d(Maria, Amber) = 1 → Good (=class of Amber)
d(Maria, Joe) = 1.333 → Good (=class of Joe)

Example of k-nn: ordinal and nominal values (cont.)

- Assume **k=3**
- Proper distance metric?
  - Using **MVDM**

Name	**Debt?**	**Income?**	**Married?**	**Risk**
Joe	High	High	Yes	Good
Amber	Low	High	Yes	Good
Harry	Low	High	No	Poor
Lindsay	High	Low	Yes	Poor
Kaley	Low	Low	Yes	Poor

Maria: **(Low, High, No, *Good*)**

Step 6.

Assign to Maria the most frequent class (majority vote)

d(Maria, Harry) = 0 → Poor
d(Maria, Amber) = 1 → Good
d(Maria, Joe) = 1.333 → Good

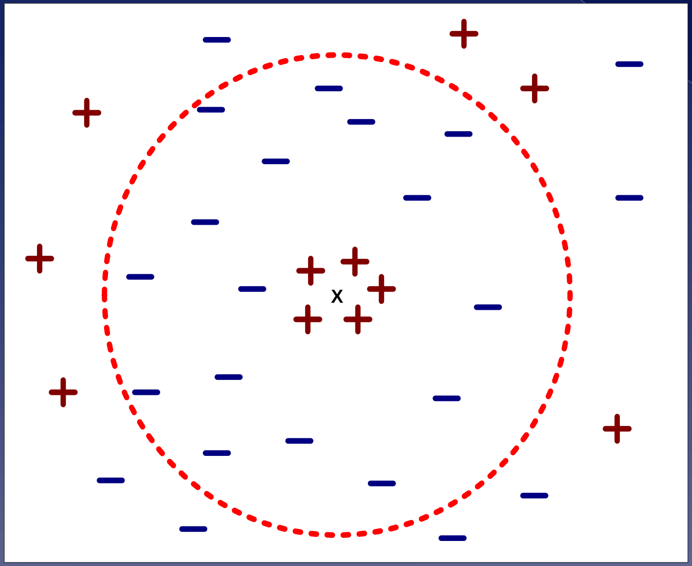
⇒ **Class of Maria: GOOD**

Nearest Neighbor Classification

- How to reach decision based on k nearest neighbors?
  - Many different approaches
    - **Un-weighted votes**: count simply the most frequent class among k nearest neighbors
    - **Distance weighted votes**: weigh each vote by some factor that takes into consideration e.g. the distance
      - Hence, records further away have less influence in the voting process

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**
  - **Rule of thumb: k ~= sqrt(number of observations)**



Nearest Neighbor Classification

- **Very important! <u>Scaling issues for numerical attributes</u>**
  - Attributes may **have to be scaled** to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M
    - Assume **<u>all above in record data</u>**. Difference in **income a lot greater** than **height difference**, which **influences (dominates) the distance measure**.
  - In **practice**, all numerical attributes are usually **scaled to the (0,1) range (other ranges possible too)**

Nearest Neighbor Classification

- Why scaling – Example
  - Assume vector representing visitors/customer to a website with attributes: **Age, Income, Number of visits .**

Jim: (75, 55000, 35) => Jim is old and visits site often

k-NN distance measure may calculate that Alice is closest to Jim, because <u>income dominates</u>

Alice: (22, 54000, 0) => Alice is young and never visited the site

But this does not sound "reasonable". Other attributes differences have been masked by income

Nearest Neighbor Classification

- How to scale? Many different ways
  - Calculate what portion of the range a value accounts for (called **min-max normalization**)

$$\textbf{New value} = \frac{\textbf{Old value} - x_{minimum}}{x_{maximum} - x_{minimum}}$$

…where **"Old value"** current value of an attribute, $x_{minimum}$ = the minimum value of the attribute and $x_{maximum}$ = the maximum value of the attribute

Ranges from 0 to 1.

Do this for all attributes.

Nearest Neighbor Classification

- How to scale? Many different ways
  - Express each value in **terms of z scores** i.e. how many std. deviations σ it is away from the mean of attribute (called **standardization**):

$$z_{val} = \frac{(val - \overline{y_{val}})}{\sigma}$$

…where **"val"** a value of an attribute, $\bar{y}_{val}$ **= mean of attribute y for which val is a value** and **σ the std deviation of attribute y**

Ranges aprox. from -3 to 3

Apply this to all numerical attributes

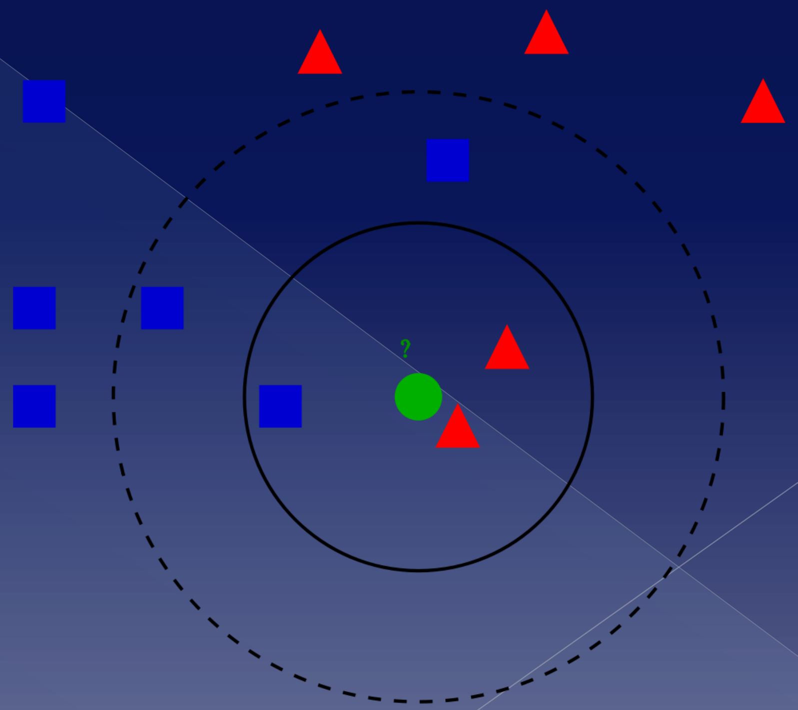
Nearest Neighbor Classification

- k-NN classifiers are **lazy learners**
  - It does **not build models explicitly**
    - **Instead stores training data and computes distances every time (wtf!). Does not build model.**
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records **is relatively expensive**
    - **Always compute again distances instances for each new record**
- Decision Trees/Rule-based are **eager learners**
  - They **build model out of training data**

Nearest Neighbor Classification

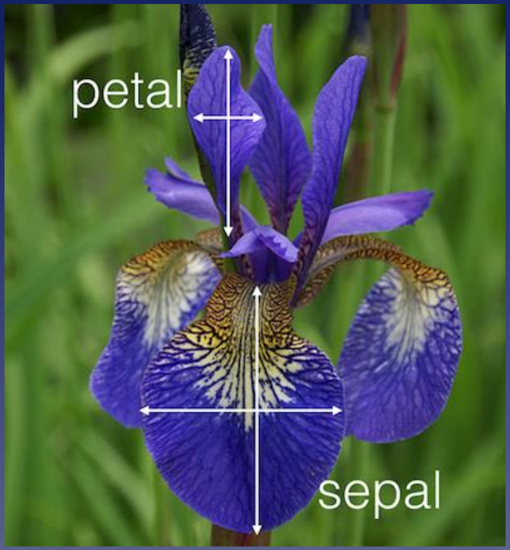
- kNN prone to **overfitting**
  - Overfitting when k small (k=1). Why?

When k small then classification prone to noise. E.g consider a space where most points are class A, few class B. Test records **happens to be near record of class B.** If k=1 probably misclassified as B, which is counterintuitive.



Nearest Neighbor implementation in R

- Using the iris dataset to classify different species of iris plant
  - 3 species: **virginica**, **setosa**, **versicolor**
  - Determine species (class) based on some characteristics, length and width- of petals and sepals. **Will use k-nn algorithm**



Iris virginica

Iris setosa

Iris versicolor

k-NN in R

- **The k-NN algorithm in R**

```
#includes the class package, containing the necessary functions for k-nn
library(class)

#Add the Iris dataset. Note: iris dataset is build-in and comes with R
data(iris)

#Take a quick look at the data (peek at data)
head(iris)

# Something strange things can be seen. The iris dataset is sorted on class. i.e.
# 50 first records are all the same species, next 50 are the same etc.
# This does not help us in getting a good training dataset which should contain
# a good mix of each species.
# Hence, first shuffle iris dataset. There are many ways to do it, but this is one

# Initialize random number generator
set.seed(9850)
# Get 150 random numbers from 0 to 1 from a uniform distribution
gp <- runif(nrow(iris))
# Now, use the outcome of order on gp, to get the rankings of the random numbers
# and use these to shuffle iris records
iris <-iris[order(gp),]
#Take a look at the values of each attribute.
summary(iris)
# There is an issue. Attributes have different ranges. This may introduce bias. So,
# try to normalize each value of attribute in the range 0 to 1. One easy way to do
# this is to normalize is to use min-max normalization :
# new_value = (old_value – col_min())/ (col_max() – col_min() ) . To do this, we will use a
# function. Makes things easier. …Continued on next slide…
```

k-NN in R

```
# Continued from previous slide
# Define a function to normalize all attributes. Yes, in R you can define function and
# store the definition of function in variables (whaaaat???)
norm <- function(x){ return( (x-min(x)) / (max(x)-min(x)) ) }

# Now normalize each attribute by applying norm i.e. make each value from 0 to 1
iris[,"Sepal.Length"] <- norm(iris$Sepal.Length)
iris[,"Sepal.Width"] <- norm(iris$Sepal.Width)
iris[,"Petal.Length"] <- norm(iris$Petal.Length)
iris[,"Petal.Width"] <- norm(iris$Petal.Width)

#Take a look at the data again
summary(iris)

# Looks cool! Create now training dataset We will use first 129 records as training
# set and the rest as testing set
iris_train <- iris[1:129,]
iris_test <- iris[130:150,]

# Single out, i.e. keep separately the class of each record. This will help us make
# some tests easier. Also, the R function for knn requires it.
iris_train_target <-iris[1:129, 5]
iris_test_target <-iris[130:150, 5]

# This is advance ninja techniques vol 4: without the next lines, R;s knn function
# goes berserk.
iris_train <-iris_train[, -5]
iris_test <-iris_test[, -5]

# Now we are ready to apply the k-NN algorithm. See next slide…
```

k-NN in R

```
# Continued from previous slide
# Call R's knn function which does our job. Set as k the square root of
# number of records in dataset (a rule of thumb)

model1 <- knn(train=iris_train, test=iris_test, cl=iris_train_target, k=13)

# Ok done! At this point we have calculated the k-nearest neighbors
# and assigned class by majority vote for each one of the records in the iris_test
# dataset.  I.e. we have predicted a class for each record in the testing set.

# Print the confusion matrix to see how our model has performed
table(iris_test_target, model1)
```

Complexity of k-NN?

- Suppose we have **training set of size d** and **dimension d and require k closest neighbors**
  - Complexity to compute distance to one training record: **O(d)**
  - Complexity to compute distance to all training records: **O(nd)**
  - Complexity to find k closest distances: **O(nk)**
  - Total time (complexity): **O(nd + nk)**
  - **For large training set (usually the case) expensive!**

K-NN: Possible if class is continuous

- Classification

$$\hat{y} = \text{ most common class in set } \{y_1, \ldots, y_K\}$$

- Regression

$$\hat{y} = \frac{1}{K}\sum_{k=1}^{K} y_k$$

In regression, take the **average of values of k nearest neighbors**. This will be the "class" or dependent value for unknown record

K-NN: Possible if class is continuous

- Weighted by distance
  - Classification

$$\hat{y} = \text{ most common class in wieghted set}$$
$$\{D(\mathbf{x},\mathbf{x}_1)y_1,\ldots,D(\mathbf{x},\mathbf{x}_K)y_K\}$$

  - Regression

$$\hat{y} = \frac{\sum_{k=1}^{K} D(x, x_k) y_k}{\sum_{k=1}^{K} D(x, x_k)}$$

Advantages/Disadvantages of k-NN?

- Advantages
  - "Learning" is very, very fast
    - If you can call it "learning" (well, it's not learning actually)
  - Can "learn" complex target functions/models easily
    - Because **there is no model** to learn.
  - Does not lose any information
    - Compare with decision tree
- Disadvantages
  - **Computationally expensive**, slow query time i.e. slow to classify unknown records
    - Due to **number of times the distance** has to be calculated training sets generally large
  - Requires lots of storage
    - Not a problem anymore (in 1956 was great problem)
  - Easily **fooled by irrelevant attributes** (most important problem) – Curse of dimensionality.
    - **Signal of important attributes may be masked by the noise of many irrelevant attributes**

Naïve Bayes Classifiers

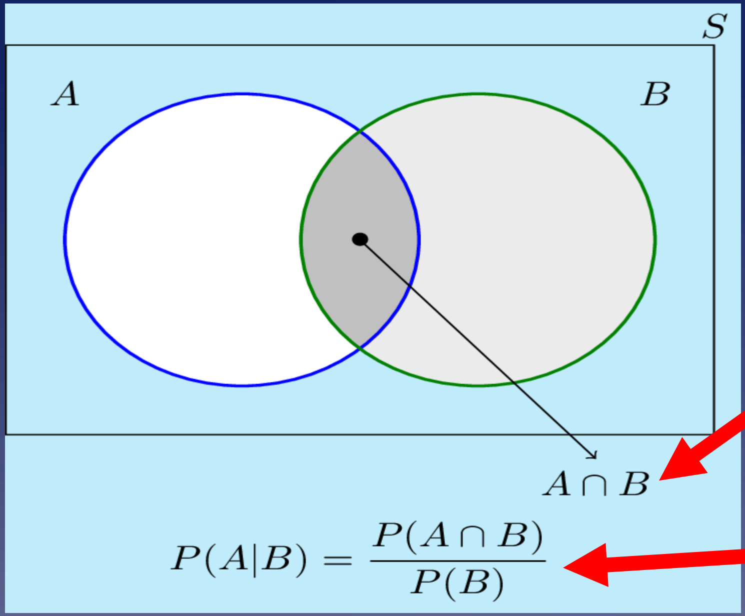
Naïve Bayes classifiers

- Naïve Bayes classifiers?
  - A **probabilistic framework** for solving classification problems
    - **Probabilistic?** Calculate some probabilities and decide class based on these
    - Record belongs to a class with some probability
      - Can calculate probability for any class (!)
      - We don't assert with certainty the class a record belongs to
  - Based on **conditional probabilities**
  - Based on **Bayes theorem**

Preliminaries

- Conditional probability
  - **Probability of event A** given that **<u>event B has occurred</u>**

$$P(A|B) = \frac{P(A\ \cap B)}{P(B)}$$



Tells us how the probability of events A and B occurring wrt Ω (S). Measures how big/pct the common area is between A and B <u>with respect to Ω (S)</u>

Tells us the probability of event A given that B occurred. Measures how big/pct the common area between A and B <u>with respect to B</u>

Bayes Theorem

- Bayes theorem
  - Tells us how **conditions** are **related to events:**

$$\boldsymbol{P(A|B) = \frac{P(B|A)P(A)}{P(B)}}$$

- Derives straight from conditional probabilities:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, but\ since\ P(B|A) = \frac{P(A \cap B)}{P(B)} \rightarrow \mathrm{P(A \cap B) = P(B|A)P(A)}$$

$$hence\ \boldsymbol{P(A|B) = \frac{P(B|A)P(A)}{P(B)}}$$

Bayes Theorem

- Intuitively understanding the elements in Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- **P(A)**: Prior belief. Probability of event A before seeing any data. The hypothesis.
- **P(B|A)**: Likelihood. Probability of the data if event B is true
- **P(B)**: Data evidence. Marginal probability of the data
- **P(A|B)**: Posterior probability. Probability of event A after having seen data of event B

Bayes Theorem

- Bayes theorem
  - Alternative forms of Bayes' theorem, based on **Law of Total Probability**
  - **Law of total probability?** Assume space Ω is partitioned into n partitions A_i such that **$A_i \cap A_j$ = ø** (**mutual exclusive**) and ∪ **A_i = Ω** (**exhaustive**), then probability of event B occurring is:

$$P(B) = \sum_{i=1}^{n} P(A_i \cap B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Bayes Theorem

- Bayes theorem
  - Visualizing **Law of Total Probability**

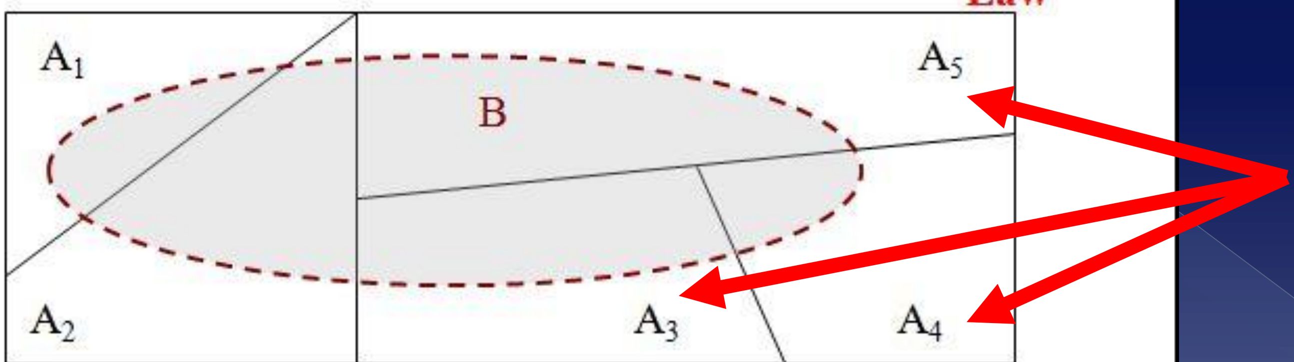
Law of Total Probability

$P(B) = \Sigma_i P(B \cap A_i) = \Sigma_i P(B|A_i)P(A_i)$

Makes sense geometrically

Makes sense conceptually

Tied together by Multiplicative Law



Partitions of space. Event B spawns/crosses partitions.

$A_i \cap A_j = \varnothing$ (Mutually Exclusive), and

$\cup A_i = S$ (Collectively Exhaustive)

Bayes Theorem

- Bayes theorem
  - **Based on Law of total probability**, Bayes theorem becomes:

$$P(A|B) = \frac{P(B|A)P(A)}{\sum_{i=1}^{n} P(B|A_i)P(A_i)}$$

…assuming that **space B is partitioned in n partitions A_i** with the same properties (i.e. A_i mutual exclusive and exhaustive)

Examples of Bayes theorem (quite counterintuitive)

- Assume the following:
  - Breast cancer affects 1% of women
  - A test (e.g. mammogram) detects breast cancer 80% if the person has breast cancer
  - 9.1% of the test detects breast cancer when the person has not cancer (false positive)
  - **Question: Given that a woman takes the test and the test reveals cancer (i.e. indicated that the woman has cancer) what is the probability that she really has cancer?**

Examples of Bayes theorem (quite counterintuitive)

- What do we have?
  - **P(has breast cancer)** = 0.01
  - **P(Test says br. cancer | has br. cancer)** = 0.8
  - **P(Test says br. cancer | has not br. cancer)** = 0.096
  - For the woman which took the test and it revealed breast cancer, we are actually looking for this:
    - **P(has breast cancer | test says br. cancer) = ?**
      - The basic idea here is that **it's different for a test to say it** and **actually having it!**

Examples of Bayes theorem (quite counterintuitive)

- Applying Bayes Theorem

$$P(has\ br.cancer \mid test\ says\ br.cancer) = \frac{P(test\ says\ br.cancer \mid has\ br.cancer)P(has\ br.cancer)}{P(test\ says\ br.cancer)}$$

P(test says br. Cancer | has br. Cancer) known = 0.8, **P(has br. cancer)** also known = 0.01 . For **P(test says br. cancer)** we can **apply the Law of total probability**: we have two partitions -br. Cancer and no br. Cancer- that are mutual exclusive and exhaustive and "test says br. cancer" crosses both areas. Hence

P(test says br. cancer) = P(test says br. cancer|has cancer)P(has cancer) + P(test says br. cancer| has no cancer)P(has no cancer) = 0.8*0.01 + 0.096*(1-0.01) = 0. 10304

Answering the question:

P(has br. Cancer | test says br. Cancer) = (0.8*0.01)/0.10304 = 0.077 (or 7.7%)

NOTE: Quite small chance, even test came out positive for breast cancer.

Bayesian classifier: basic idea

- Assume that **X** are all the values of some attributes of a record r **($x_1, x_2, x_3, \ldots, x_d$)** and **Y** the class label of the record
  - Note: **X** will stand in next sections as the **set of all values of attributes of a record (except class of course)**
- Further assume that the **class label Y is non-deterministically related to X**
  - **Non-deterministically?** Simply means you can not associate values of X with a particular value of Y with certainty 100%

Bayesian classifier: basic idea

- Then, we can **treat X and Y as random variables and calculate P(Y | X)** i.e. the probability that record r, with these values X on its attributes belongs to class Y.
  - The problem of **classifying record r** becomes **then to find Y (class) that maximizes P(Y|X)**
- This is the **main idea of Bayesian classifiers**

Bayesian classifier: basic idea

- **P(Y|X)** known as **posterior probability**
- **P(Y)** known as **prior probability**
- As with all classification methods, 2 phases
  - **Training phase**: Try to calculate P(Y|X) based on the records of the training set
  - **Testing phase:** Given a record X' with unknown class, and **find P(Y'|X') which maximizes this probability**. If Y' maximizes, say record X' belongs to class Y'

Bayesian classifier: basic idea

- How to compute P(Y|X) ?
  - More clearly **P(Y|X)** is actually **$P(Y | x_1 \cap x_2 \cap x_3 \cap \ldots \cap x_d)$ where xi values on attributes of record with dimension d. (Remember X attribute values)**
  - Use **Bayes Theorem** to **calculate posterior probability**:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Bayesian classifier: basic idea

- How to compute P(Y|X) ?

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

…calculates posterior probability **based on prior probability P(Y), conditional dependence P(X|Y)** and **P(X) called the evidence.**

Now, **given a record r**, with **unknown class Y**, in order to see which class it belongs, try to **maximize P(Y|X)** or **maximize P(X|Y)P(Y)** since P(X) for record r always constant (and not always computable). Record belongs to class Y that maximizes this probability,

Bayesian classifier: basic idea

- How to calculate P(Y | X)?
  - It's **easy to calculate prior P(Y)** based on training set
  - Calculating **P(X | Y) not that easy**.
    - Note: For record r with dimension d and attribute values (x1, x2, x3, … ,xd), **$P(X|Y) = P(x_1 \cap x_2 \cap x_3 \cap \ldots \cap x_d \mid Y)$**
  - Bayes classifiers **differ in their way** they deal with calculating P(X | Y) and what assumptions they make
    - **Naïve Bayes classifier**
    - Artificial Neural Networks (ANN)

Naïve Bayes Classifier

- **Naïve Bayes classifier assumes** that the attributes X of record r are **conditionally independent of class Y**. **I.e.**

$$P(X|Y=y) = P(x_1 \cap x_2 \cap \cdots \cap x_d | Y=y) = \prod_{i=1}^{d} P(x_i|Y=y)$$

In Naïve Bayes, new record r is classified to **class y if P(y) Π P(x_i| Y=y) is maximal.**

Naïve Bayes Classifier

- Conditional independence
  - Let there be **three events X, Y, Z**. We say that event **X is conditionally independent of Z given Y** when:

$P(X|Y \cap Z) = P(X|Y)$ or equivalent
$P(X \cap Z \mid Y) = P(X|Y)P(Z|Y)$

Naïve Bayes Classifier for discrete attributes - Example

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**

- For discrete attributes:
  - $P(Y) = N_y / N$, Y class attribute
    - **P(Yes) = 3/10**
    - **P(No) = 7/10**
  - **$P(X_i | Y_k) = |X_{ik}| / N_{yk}$** Where $|X_{ik}|$ number of attributes having value x_i and belong to class Y_k . E.g.
    - **P(Status=Married|No) = 4/7**
    - **P(Refund=Yes|Yes) = 0**

Naïve Bayes Classifier for qualitative attributes – applying naïve Bayes example

Training Data

Class!

ID	M	N	Q	R
1	M1	N3	Q2	R2
2	M2	N3	Q2	R1
3	M2	N2	Q1	R1
4	M1	N2	Q1	R2
5	M2	N1	Q3	R2
6	M1	N1	Q3	R2

Assume all attributes categorical. Assume conditional independence of class

- Given record **X =(M2, N3, Q1).** What is it's class?
  - P(R1) = 1/3
  - P(R2) = 2/3
  - **Calculate P(R1|X)** . Note: Can't and don't need to calculate P(X). P(X|R1)P(R1)= P(M2|R1)P(N3|R1)P(Q1|R1)P(R1) =1*(1/2)*(1/2)(1/3)= *0.083*
  - **Calculate P(R2|X).** P(M2|R2)P(N3|R2)P(Q1|R2)P(R2) = (1/4)*(1/4)*(1/4)*(2/3)=*0.0104*
  - **Since P(R1|X) > P(R2|X), record X belongs to class R1**

Naïve Bayes Classifier for qualitative attributes – applying naïve Bayes example

Training data

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Unknown record. Find class

- Assume
  - A: all attributes of unknown record
  - M: Mammals
  - N: Non-Mammal
- **P(A|M) =** (6/7)*(6/7)*(2/7)*(2/7)= **0.06**
- **P(A|N)** = **0.0042**
- **P(A|M)*P(M) = 0.021**
- **P(A|N)*P(N) = 0.0027**
- Hence, since **P(A|M)*P(M) > P(A|N)*P(N)** unknown is classified as "Mammal".

Naïve Bayes Classifier: Continuous attributes

- For continuous attributes:
  - **Discretize** the range into bins
    - one **ordinal attribute** per bin (e.g. poor, good, better, very good etc. **Note: have ordering**)
    - violates independence assumption
      - Discretization may mask discriminating factors of attribute (loss of information)
  - **Two-way split: (A < v) or (A > v)**
    - choose only one of the two splits as new attribute
  - **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 $P(A_i|c)$**

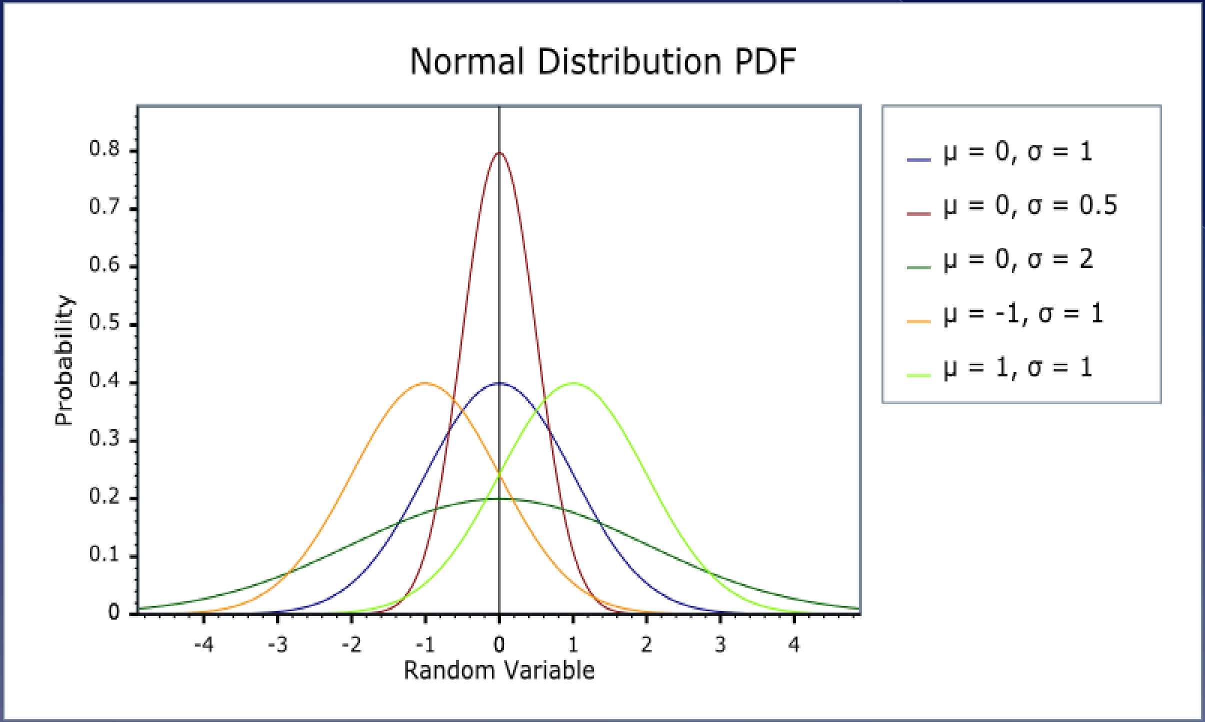
Naïve Bayes Classifier: Continuous attributes

- Probability density estimation method
  - Assume every continuous attribute normally distributed.
  - Calculate mean, variance **for each attribute given class**
  - Calculate **$P(x_i | Y_j)$ for each (x_i, Y_j)** pair as follows, using the normal distribution's PDF:

$$P(x_i|Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

Naïve Bayes Classifier: Continuous attributes

- PDF of normal distribution of attributes assumption in Naïve Bayes



Some notes:
The **PDF (Probability Density Function)** does not calculate probabilities(!). For continuous variable X **$P(X=x_0) = 0$**. The PDF tells us the "density" at this point i.e. **how common are samples (i.e. observed values) at exactly this value $X=x_0$**

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Naïve Bayes Classifier: Continuous attributes - Example

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

Training Data

- Assume Income (cont. var) normally distributed
- Calculate $P(x_i \mid c_j)$ for each pair $(x_i \mid c_j)$ using normal distribution's PDF
- To do this, calculate for each class (yes/no), mean and variance
  - For Class=no
    - **Sample mean μ of "no" class = 110K** (add all income where class=no and divide by # of "no" classes). Note: will use value 110, as we take each income as measured in K. i.e. 125 instead of 125000 (125K)
    - **Sample variance σ^2 of "no" class = 2975**

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

Naïve Bayes Classifier: Example with unknown record

Conditional prob for discrete attributes from training data

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

Data needed to calculate continuous variables assuming normal distribution, via PDF

- Assume previous training data
- We are given **new, unclassified record X=(No, Married, 120K). Class=?? Mixed: discrete and continuous attributes.**
- Calculate P(X | Class=No) and P(X | Class=Yes) for record X
- **P(X | Class=No) =** P(Ref=No | No)*P(Married | No)*P(120 K | Class=No) = (4/7)*(4/7)*0.0072 = **0.0024**
- **P(X | Class=Yes) =** P(Ref=No | Class=Yes)*P(Married | Class=Yes)*P(Income=120K | Class=Yes) = $1 \times 0 \times 1.2 \times 10^{-9} = 0$
- Calculate now P(X | Class=No)*P(No) and P(X | Class=Yes)*P(Yes)
  - P(X | Class=No)*P(No) = 0.0024 *(7/10) = 0.00168
  - P(X | Class=Yes)*P(Yes) = 0
- **Since P(X | Class=No)*P(No) > P(X | Class=Yes)*P(Yes), given record X is classified as "No"**

Naïve Bayes Classifier - improvements

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

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

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

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

Naïve Bayes in R

```
#Includes functions for Naïve Bayes
library(e1071)

#We will be using the Congressional Voting Records Data Set
#From: http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

#First read the data. Note the dataset HAS NO headers, hence set header to FALSE.
#We well add headers later. NOTE: Change your path to data appropriately!
voteData = read.csv("house-votes-84.data", header=FALSE)
attach(voteData)

#Add headers to data. Makes working with dataset easier
colnames(voteData) <- c("party", "infants", "water-cost", "budgetRes", "PhysicianFr",
"ElSalvador", "ReligSch", "AntiSat", "NicarAid", "Missile", "Immigration", "CorpCutbacks",
"EduSpend", "RightToSue", "Crime", "DFExports", "SAExport")

#Take a quick look at the data. Is everything ok?
head(voteData)

#Looks fine. We are now ready to train our model and derive our Naïve Bayes
#classifier. We want to predict the party based on how a congress delegate
#has voted on various issues.
NaiveBayesModel <- naiveBayes (party ~ ., data = voteData)

#Done! Model created. Variable NaiveBayesModel contains now our naïve bayes model
#as it resulted from the training data (voting records dataset)
#Now, try to predict the party based on the voting history of some congressman. See next
slide
```

Naïve Bayes in R

```
#Now, try to apply the Naïve Bayes model to an unknown record.

#Add a new unknown record to existing voteData. Note that first attribute (party) has
#value ? meaning we don't know it and try to guess it from all the other
#attributes. NOTE: we will get a warning but we ignore it.
voteData[nrow(voteData)+1, ] <-
c("?","n","n","y","y","y","n","n","y","n","n","y","n","y","y","y","y")

#Apply Naive Bayes model to unknown record i.e. to last record that was
#added to voteData
unknownRecordClass = predict(NaiveBayesModel, voteData[nrow(voteData), ])

#Now unknownRecordClass has the class i.e. party predicted for unknown record.
#Let's see it
unknownRecordClass

#You can also plot it (sigh)
plot(unknownRecordClass)
```

Naïve Bayes - Summary

- **Advantages**
  - **Robust** to **isolated noise points**
  - Can handle missing values by ignoring the instance during probability estimate calculations
  - **Robust** to **irrelevant attributes**
- **Disadvantages**
  - Assumption: **class conditional independence**, which may cause loss of accuracy
  - **Independence assumption may not hold** for some attribute. Practically, dependencies exist among variables
    - Use other techniques such as Bayesian Belief Networks (BBN)

Appendices

Appendix B: Bibliography

- Cover, T., and P. Hart. 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory. 13(1): 21-7.
- S. Arya, D. N. Mount, S. Netanyahu, R. Silverman, A. Y. Wu, An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM, 1998
- A. Broder, S. Glassman, M. Manasse, G. Zweig, Syntactic clustering of the Web , WWW6.
- S. B. Imandoust, M. Bolandraftar, Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background, Int. Journal of Engineering Research and Applications, 2013
- D. Guegan, P. Rakotomarolahy, The Multivariate k-Nearest Neighbor Model for Dependent Variables: One-Sided Estimation and Forecasting, Working paper http://econpapers.repec.org/paper/msecesdoc/09050.htm

Appendix B: Bibliography

- Brandts, Jordi, Jacob K. Goeree, and Charles A. Holt "Naive Bayesian Learning and Adjustment to Equilibrium in Signaling Games," University of Virginia, Discussion Paper, (1996, revised 1999)
- Kim J, Le DX, Thoma GR. Naive Bayes Classifier for Extracting Bibliographic Information From Biomedical Online Articles Proc 2008 International Conference on Data Mining. Las Vegas, Nevada, USA. July 2008;II:373-8
- Ng, A., M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in Advances in Neural Information Processing Systems 14. Cambridge, MA: MIT Press, 2002.