Supervised Learning and Classification

COSC 757: Spring 2016

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numerical Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is
  - Remote Sensing

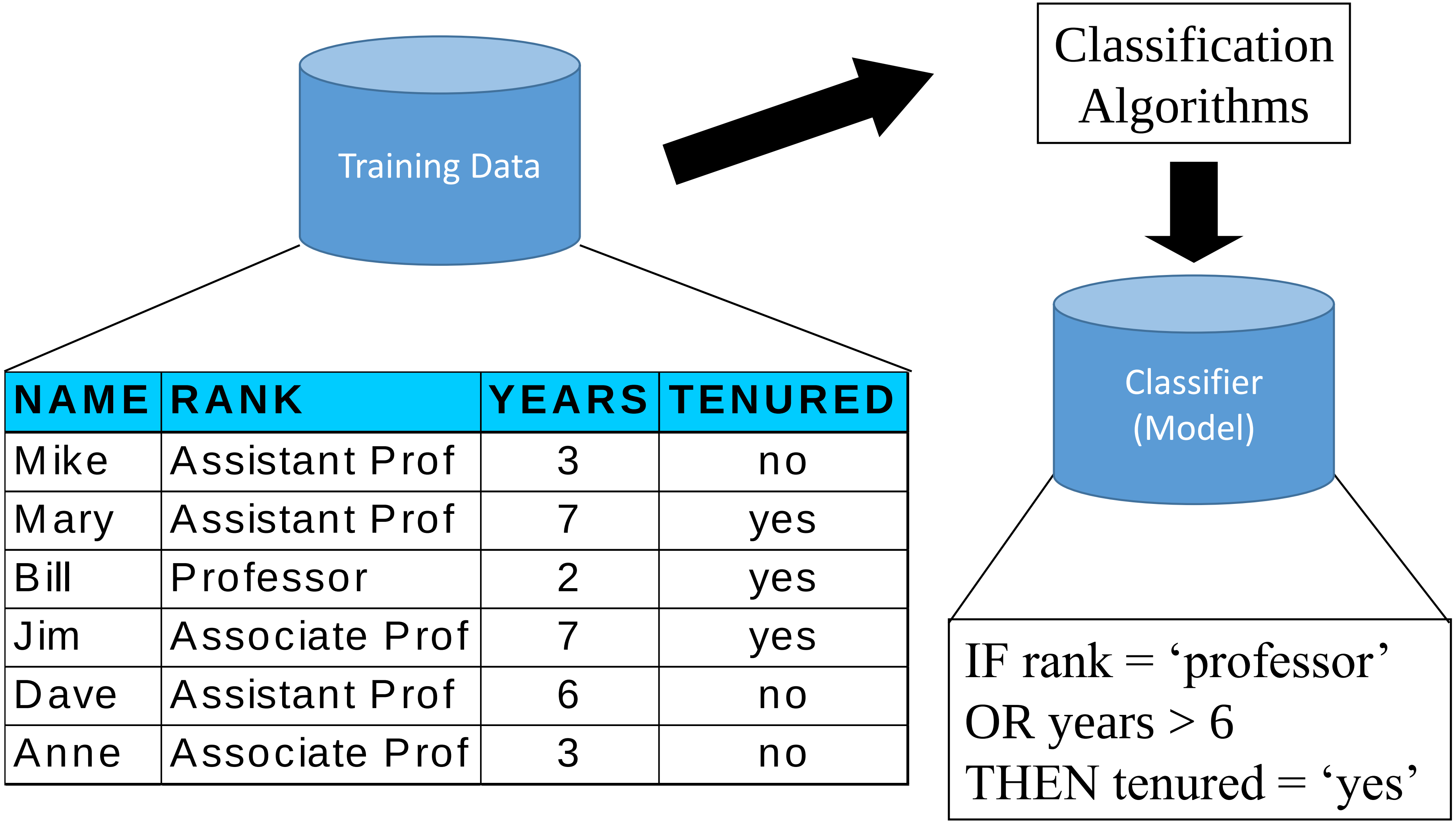
Classification – A Two-Step Process

- Model construction: describing a set of predetermined classes
    - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
    - The set of tuples used for model construction is training set
    - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
    - Estimate accuracy of the model
        - The known label of test sample is compared with the classified result from the model
        - Accuracy rate is the percentage of test set samples that are correctly classified by the model
        - Test set is independent of training set (otherwise overfitting)
    - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Classification Task

- First, the classification algorithm examines the data set values for the predictor and the already classified target variables in the *training set*
- This way, the algorithm “learns” which values of the predictor variables are associated with values of the target variable
- For example, older females may be associated with *income_bracket* values of “High”
- Now that the data model is built from the *training set*, the algorithm examines *new records* for which *income_bracket* is unknown
- According to classifications in the training set, the algorithm classifies the new records
- For example, a 63 year-old female might be classified in the “High” income bracket

Process 1: Model Construction



Training Data

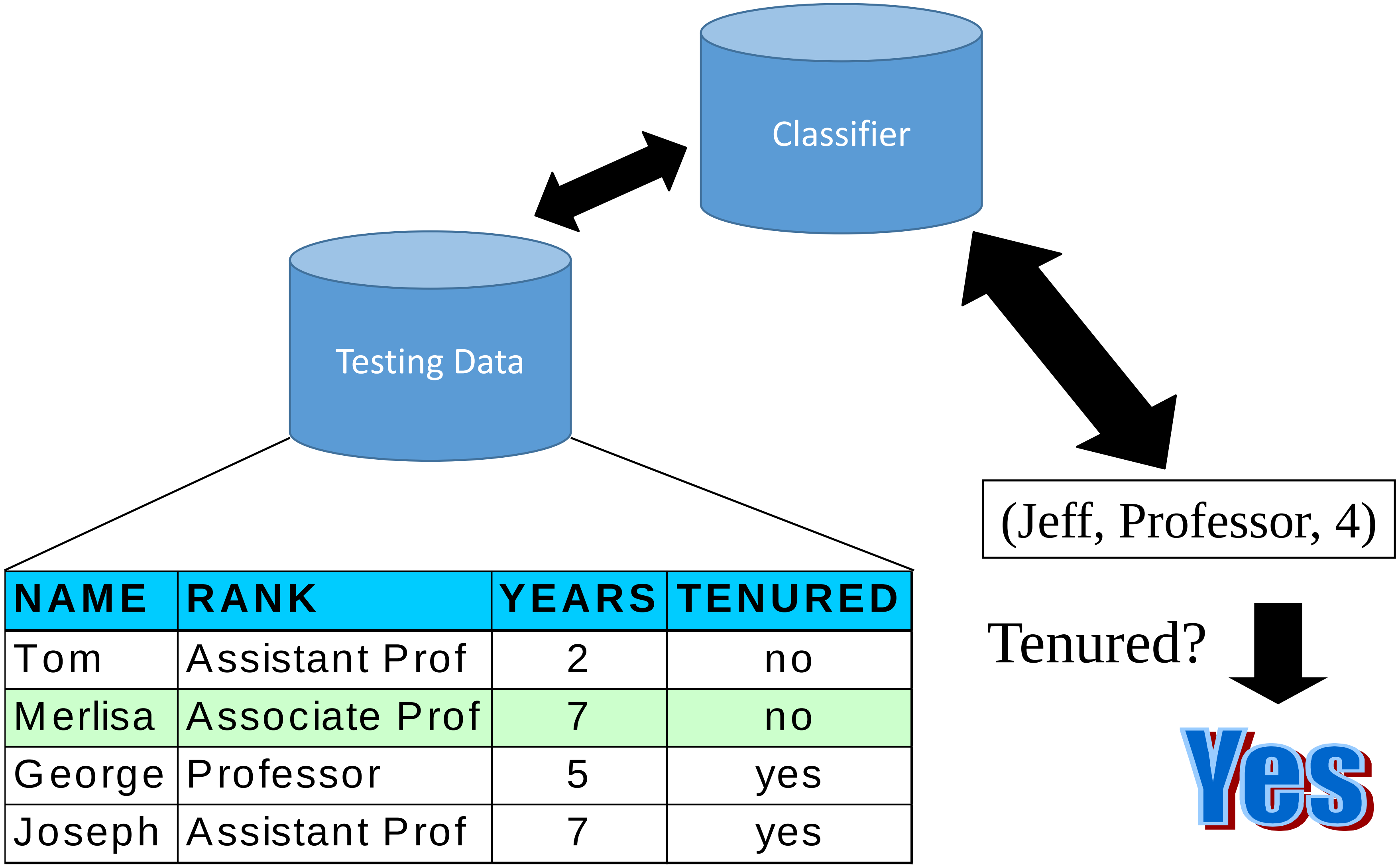
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification Algorithms

Classifier (Model)

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Process 2: Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Holdout Method & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - <u>Random sampling</u>: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained

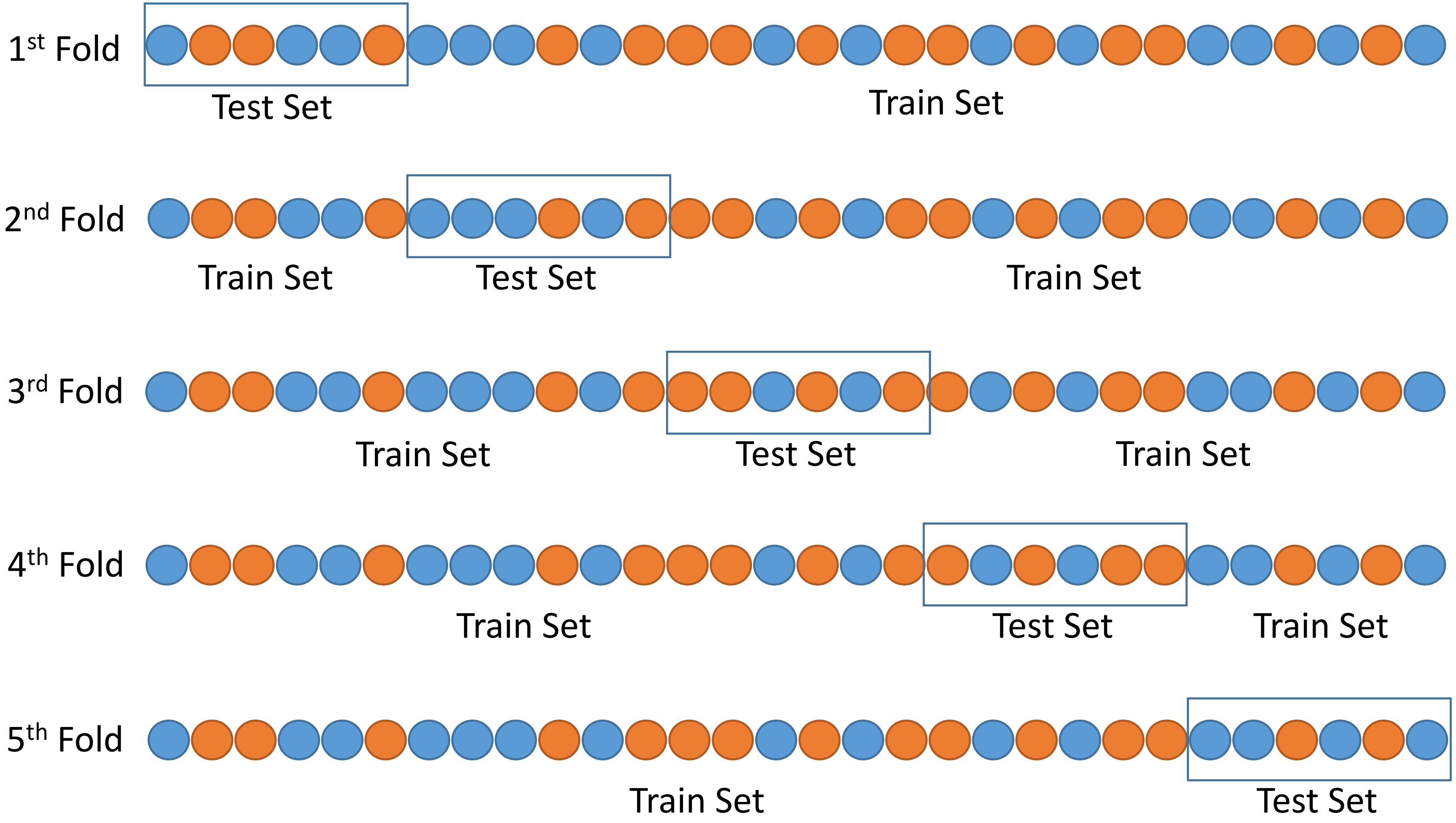
Cross Validation

- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Good choice if the number of tuples are small
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approximately the same as that in the initial data

Some Sampling Methods

- Simple Random Sampling
  - Samples are drawn randomly from the dataset
- Stratified Random Sampling
  - Samples are randomly drawn from sub-groups (e.g. age ranges)
- Sampling with replacement
  - Each time a tuple is selected, it is equally likely to be selected again and added to the training set
- Sampling without replacement
  - When a tuple is selected, it is removed from the

Simple K-Fold Cross-Validation Example (K=5)



Issues Affecting Model Selection

- **Accuracy**
    - classifier accuracy: predicting class label
- **Speed**
    - time to construct the model (training time)
    - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
    - understanding and insight provided by the model
- Other measures, e.g., decision tree size or compactness of classification rules

Bootstrap Sampling

- **Bootstrap**
  - Also Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 – 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

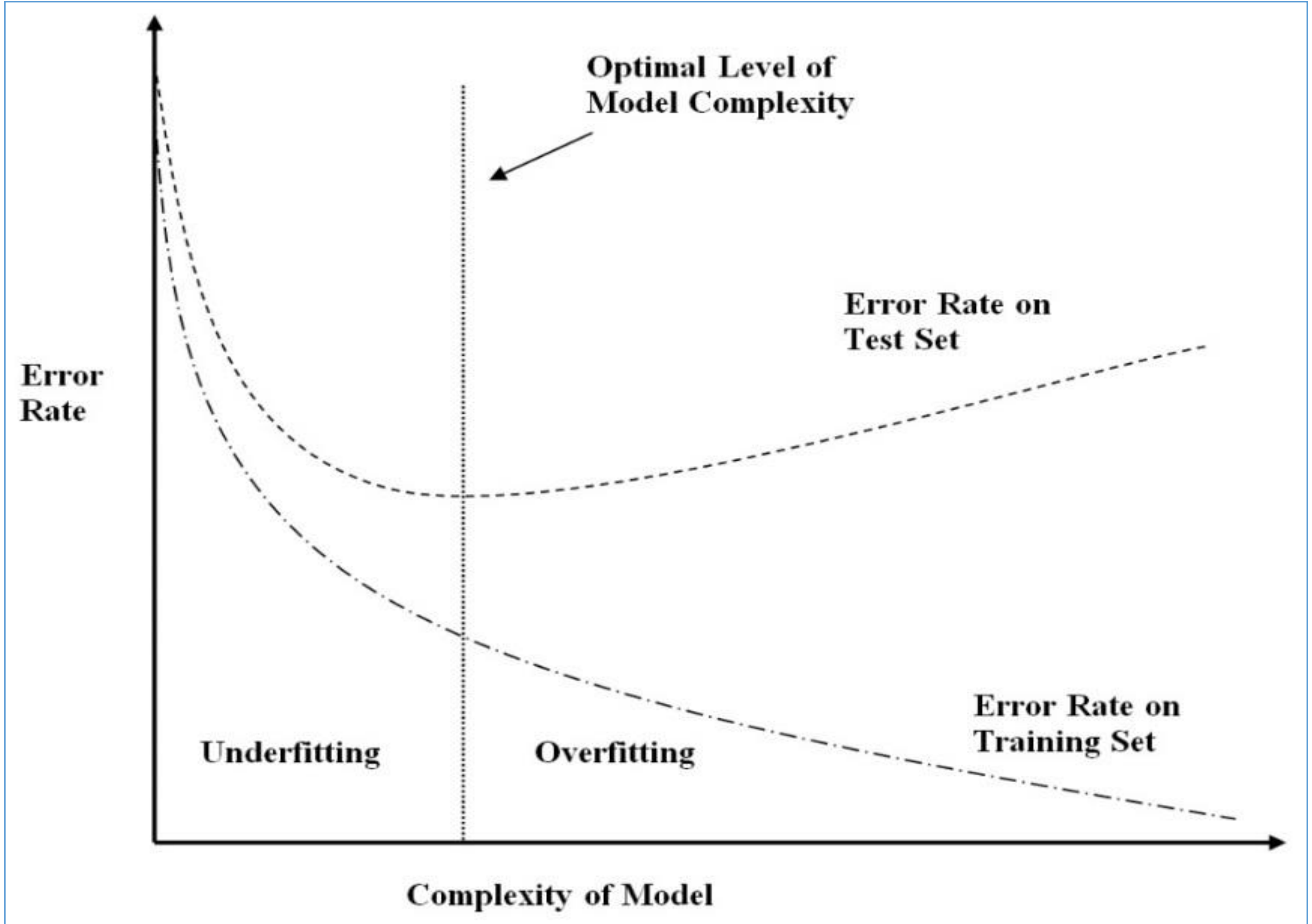
Validation of Partition

- Use methods discussed last week:
  - Confidence intervals
  - Two-sample T test for difference in means
  - Two-sample Z test for difference in proportions
  - X^2 test for homogeneity of proportions and goodness of fit
  - ANOVA to test whether the mean value of a continuous variable is the same across a set of partitions

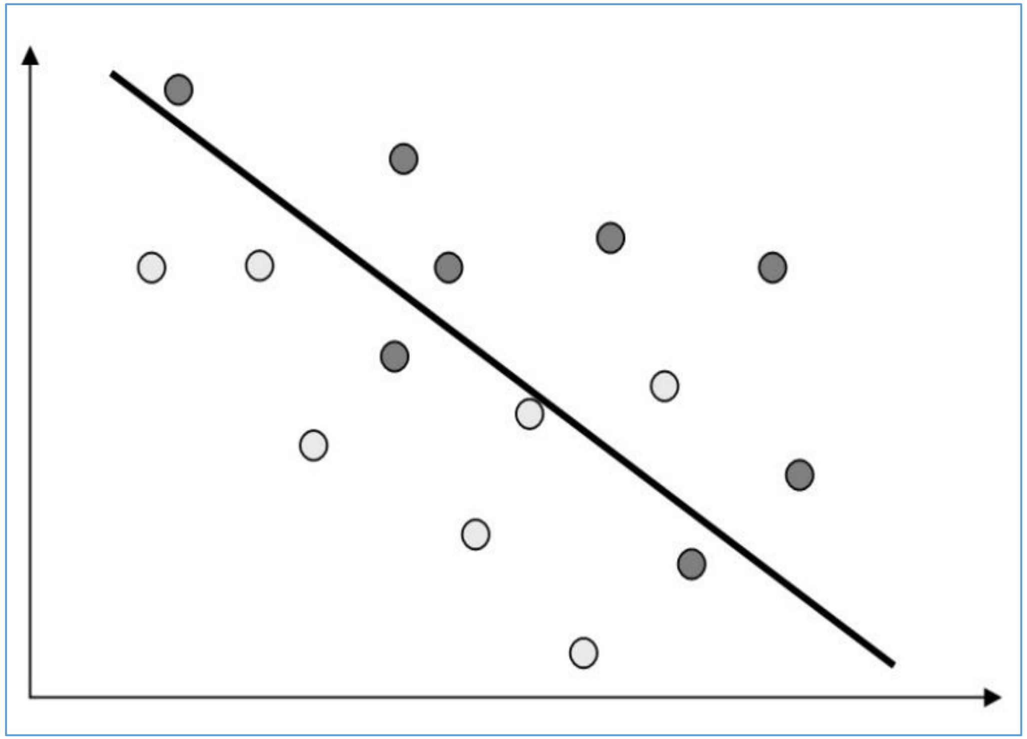
Overfitting

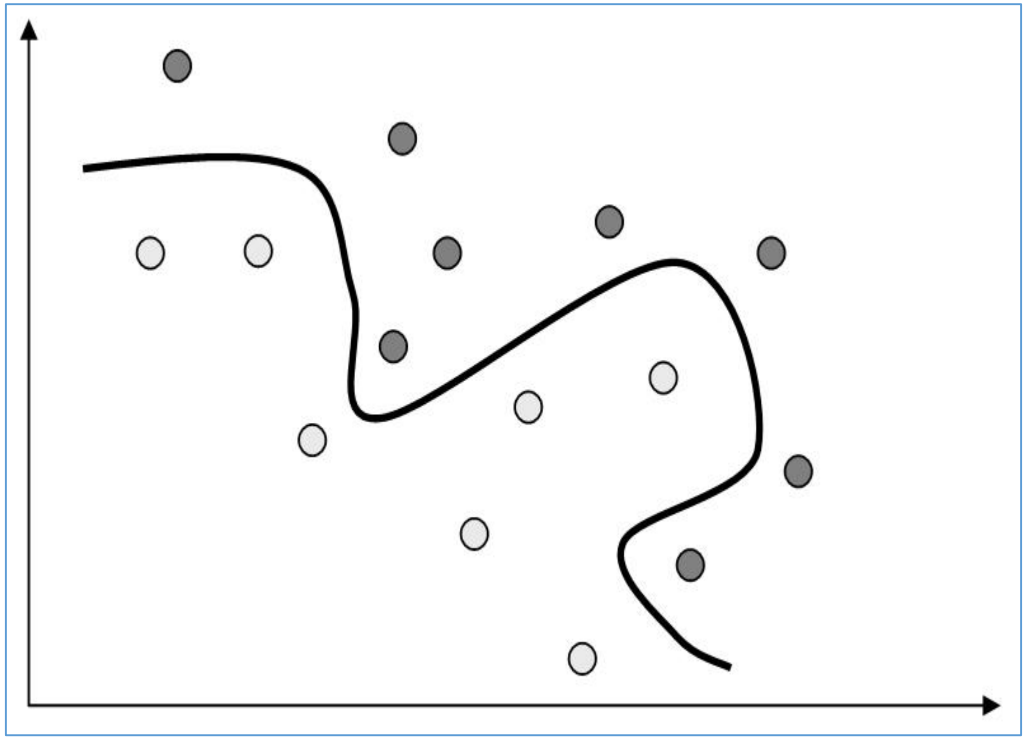
- Usually, the accuracy of model is not as high on the test set as it is on the training set
    - This might be caused by *overfitting* on the training set
- Overfitting occurs when the model tries to fit every possible trend/structure in the training set, like the spurious data issues mentioned above
- There is a need to balance the model complexity (resulting in high accuracy in training set) and generalizability to the test/validation sets
    - Increasing complexity leads to degradataion of generalizability of the model to the test set, as shown in Figure 6.1 (next slide)
- As the model grows in complexity, the error rates for both training sets starts to fall
- As the model complexity increases, the error in the training set continuous to fall as the error in the test set starts to increase
    - This is caused because the model has memorized the training set rather than leaving room for generalization to unseen data

Overfitting

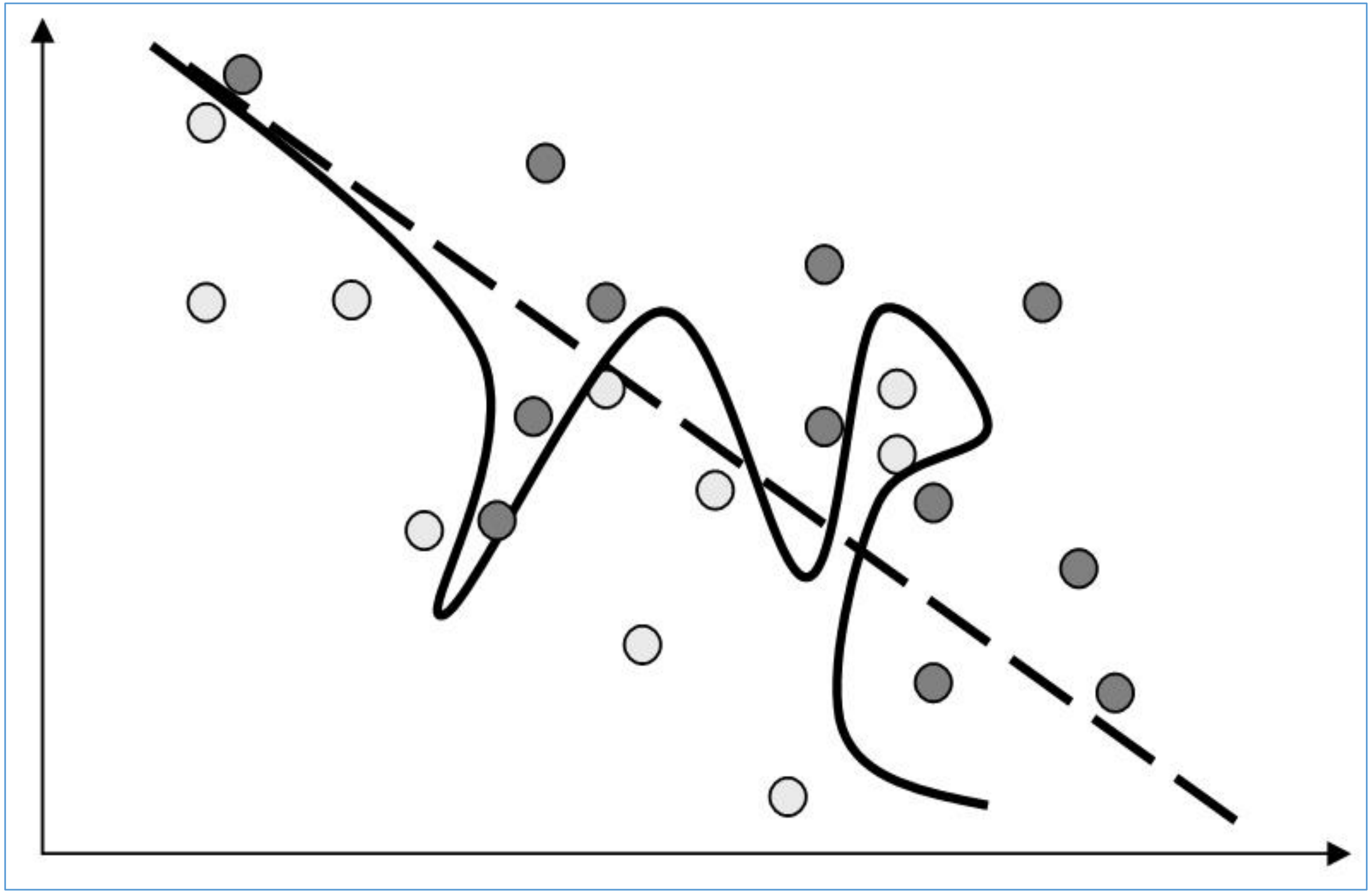


Bias-Variance Trade-Off





Bias-Variance Trade-Off



Bias-Variance Trade-Off

- Even though the high-complexity separator has *low bias* (low error rate on the training set), it has *high variance*
- And, even though the low-complexity model has a high *bias*, it has low *variance*
- This is known as the *bias-variance trade-off*
    - This is another way of describing the overfitting/underfitting
    - As model complexity increases, the bias of the training set decreases, but the variance increases
    - The goal is to construct a model in which neither bias nor variance is too high
- Example: A common evaluation for accuracy of models for continuous target variables is mean-squared error (MSE)
    - Why is MSE such a good evaluative measure?
    - Because it combines both bias and variance
    - MSE is a function of the estimation error (SSE) and the model complexity (degrees of freedom)
    - It can be shown (Hand et al. [1]) that the MSE can be partitioned using the following equation: MSE=variance+bias^2

Balancing the Training Data

- Balancing is recommended on classification models when one target variable class has much lower frequency than the other classes
  - The algorithm have a chance to learn about all types of records, not just those with high frequency
- Example: In fraud classification model with 100,000 transactions, only 1000 are fraudulent
  - The model could achieve 99% accuracy by labeling all transactions as “non-fraudulent” – this behavior is not desired
  - Instead, balance should be performed so that relative frequency of fraudulent transactions is increased
- There are two ways of doing balancing
  1. Resample a number of fraudulent recods
  2. Set aside a number of non-fraudulent records

Balancing the Training Data

- *Resampling* refers to sampling at random with replacement from a data set
- Example: Use Resampling so that the fraudulent records represent 25% of the balanced training set, rather than 1%
  - Solution: Add 32,000 resampled fraudulent records so that we had 33,000 fraudulent records, out of a total of 100,000+32,000=132,000 records in all

$$\frac{33{,}000}{132{,}000} = 0.25 = 25\% \text{ desired records}$$

- The formula to determine the number of records to resample is

$$x = \frac{p(records) - rare}{1 - p}$$

where:

x is the required number of resampled recods

p is the desired proportion of rare values in the balanced data set,

records represents the number of records in the unbalanced data set, and

rare represents the current number of rare target values

Balancing the Training Data

- Set aside a number of non-fraudulent records
  - When resampling is not desired, a number of non-fraudulent records would be set aside instead
- To achieve a 25% balance proportion, we would retain only 3000 non-fraudulent records
- We would need to discard 96,000 out of 99,000 non-fraudulent records from the analysis
  - It would not be surprising if the model suffer out of starving for data in this way
- When choosing a desired balancing proportion, recall the rationale for doing so: in order to allow the model a sufficiently rich variety of records to lean how to classify the rarer value of the target variable across a range of situations
  - The balancing proportion can be lower if the analyst if confident that the rare target value is exposes to a rich variety of records
  - The balancing proportion should be higher if the analyst is not confident of this

Balancing the Training Data

- **The test data set should never be balanced**
  - The test data set represents new data that the model have not seen yet
  - Real work data is unbalanced, therefore, he test data set shouldn’t be balanced either
  - All model evaluation will take place using the test data set, so that evaluative measures will be applied to unbalanced data

Establishing Baseline Performance

- Without a baseline, it is not possible to determine whether our results are any good
- Example: Suppose that we naively report that “only” 28.4% of the customers adopting the International Plan (see Table 3.3) will churn
  - This doesn’t sound too bad, until we notice that the overall churn rate is only 14.49%
  - This overall churn rate may be considered our *baseline*, against which any further results can be calibrated
  - Thus, belonging to the International Plan actually nearly doubles the churn rate – Not good!

Establishing Baseline Performance

- The type of baseline to use depends on the way the results are reported
  - For the churn example, we re interested in decreasing the overall churn rate
  - Thus, our objective should be to report a decrease in overall churn rate
- Example: Suppose the data mining model resulted in a predicted churn rate of 9.99%
  - This represents a 14.49%-9.99%=4.5% absolute decrease in the churn rate
  - But also a 4.5%/14.49%=31% in relative decrease in the churn rate
  - The analyst should make it clear for the client which comparison method is being used

Establishing Baseline Performance

- In an estimation task using a regression model, our baseline may take the form of a “$\bar{y}$ model”
  - $\bar{y}$ model - The model simply finds the mean of the response variable, and predicts that value for every record
- No data mining model should have a problem beating this $\bar{y}$ model
  - If your data mining model cannot outperform the $\bar{y}$ model, then something is clearly wrong
  - Recall that we measure the goodness of a regression model using the standard error of the estimate ***s***, along with the $\boldsymbol{r^2}$
- A more challenging baseline would be using results already existing in the field
  - If current algorithm your company uses succeeds identifying 90% of all fraudulent transactions, then your model would be expected to outperform this 90%

k-Nearest Neighbor Algorithm

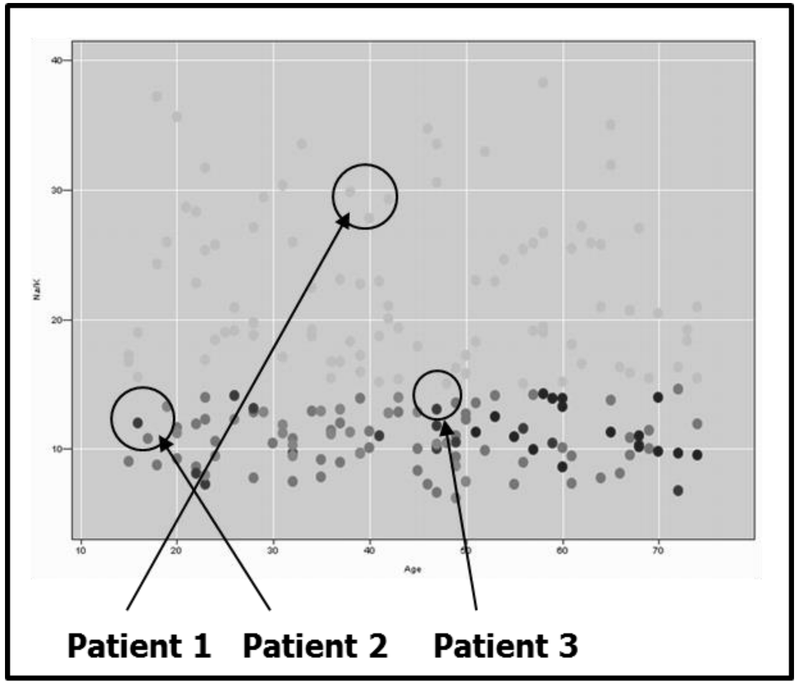
- The *k*-Nearest Neighbor algorithm is an example of instance-based learning where training set records are first stored
- Next, the classification of a new unclassified record is performed by comparing it to records in the training set it is most similar to
- *k*-Nearest Neighbor is used most often for classification, although it is also applicable to estimation and prediction tasks

k-Nearest Neighbor Algorithm

- Step 1: Determine parameter k = number of nearest neighbors.
- Step 2: Calculate the distance between the new instance and all the training examples.
- Step 3: Sort the examples by distance and determine nearest neighbors based on the k-th minimum distance.
- Step 4: Gather the category Y of the nearest neighbors.
- Step 5: Use simple majority of the category of the nearest neighbors as the prediction value of the query instance.

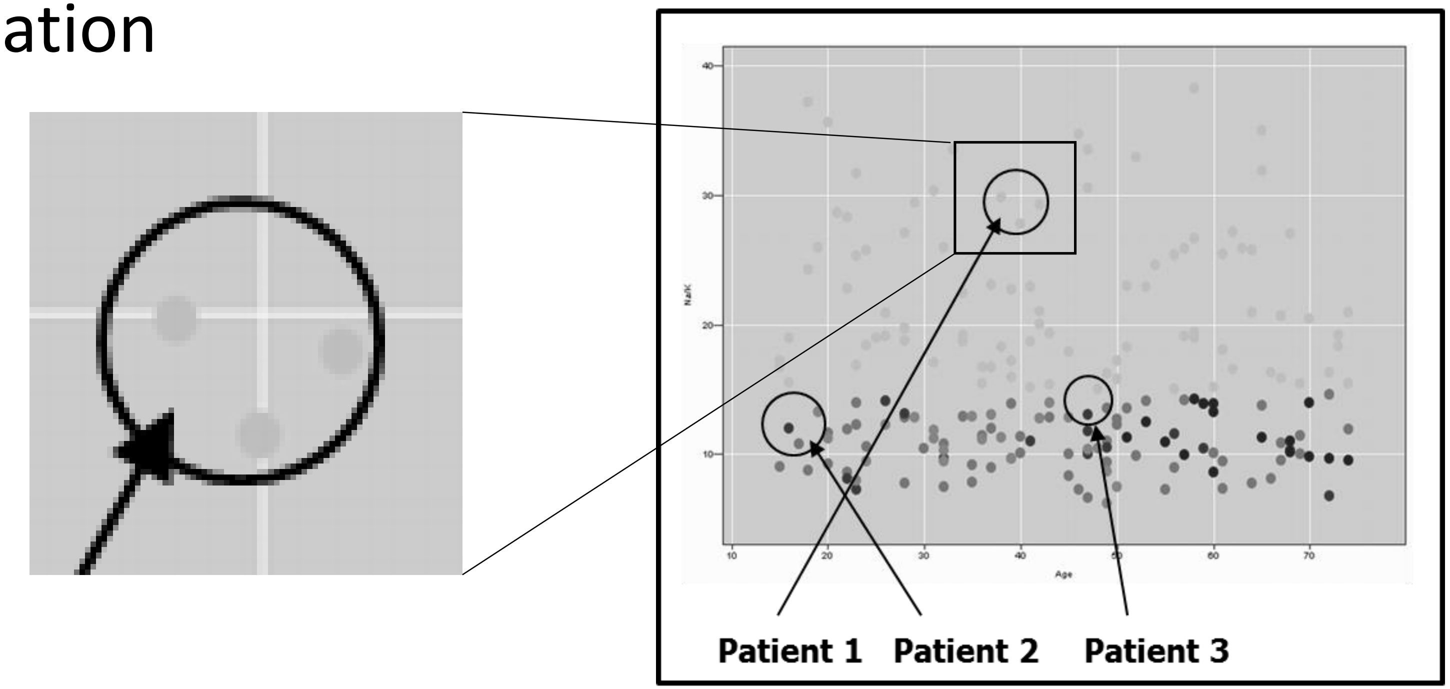
k-Nearest Neighbor Algorithm

- Example: This scatter plot of Na/K against Age shows the records in the training set that patients 1, 2, and 3 are most similar to
- A “drug” overlay is shown where Light points = drug Y, Medium points = drug A or X, and Dark points = drug B or C



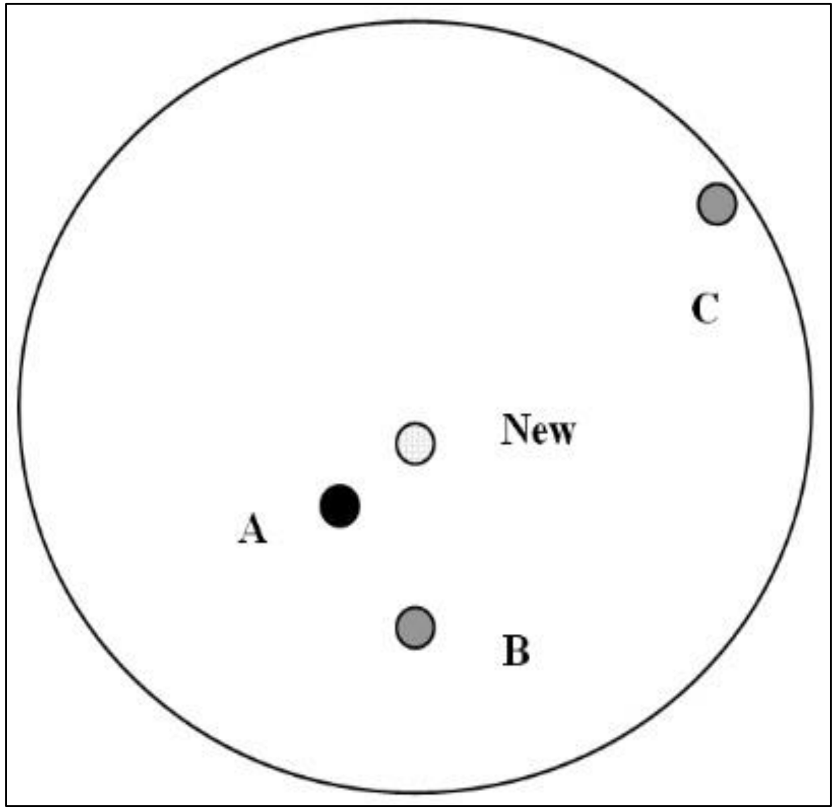
k-Nearest Neighbor Algorithm

- Which drug should Patient 1 be prescribed?
- Since Patient 1’s profile places them in the scatter plot near patients prescribed drug Y, we classify Patient 1 as drug Y
- All points near Patient 1 are prescribed drug Y, making this a straightforward classification



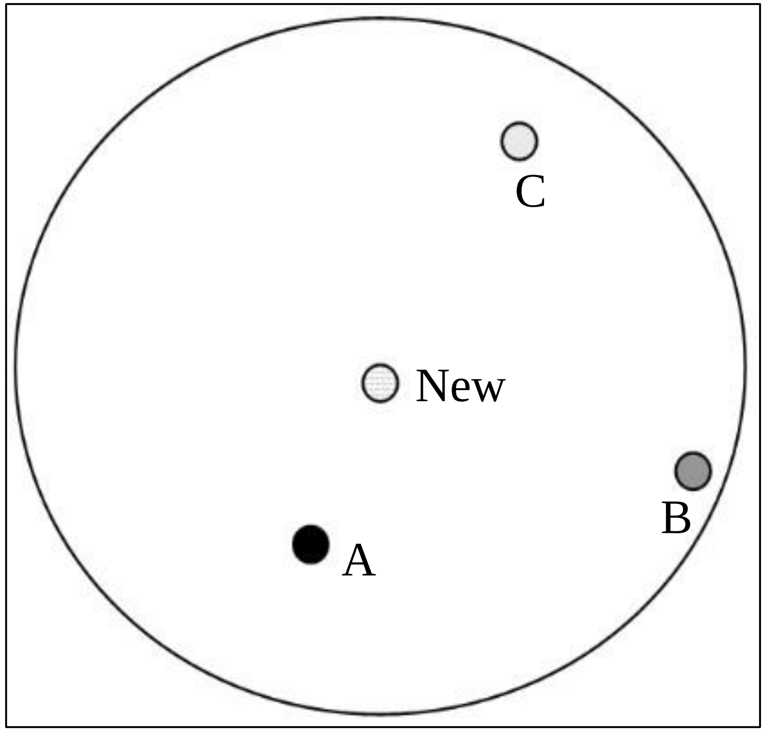
k-Nearest Neighbor Algorithm

- What if we want to prescribe a drug for a new patient who is 17-years-old with a Na/K ratio = 12.5?
- What if we set k = 1?
- What if we set k = 2?
- What if we set k = 3?



k-Nearest Neighbor Algorithm

- What if a new patient is 47-years-old and has a Na/K ratio of 13.5?
- What if we set k = 1?
- What if we set k = 2?
- What if we set k = 3?



Considerations for using k-Nearest Neighbor Algorithm

- How many neighbors should be used? *k* = ?
- How is the distance between points measured?
- How do we combine the information from more than one observation?
- Should all points be weighted equally, or should some points have more influence?

Distance Function

- How is similarity defined between an unclassified record and its neighbors?
  - Example: For a 50-year-old male, which patient is more similar, a 20-year-old male or a 50-year-old female
- A distance metric is a real-valued function d, such that for any coordinates x, y, and z:

1. $d(x,y) \geq 0$, and $d(x,y) = 0$ if and only if $x = y$

   Distance is always non-negative

2. $d(x,y) = d(y,x)$

   Commutative, distance from “A to B” is distance from “B to A”

3. $d(x,z) \leq d(x,y) + d(y,z)$

   Triangle inequality holds, introducing a third point can never shorten the distance between two other points

Distance Function

- The Euclidean Distance function is commonly-used to measure distance

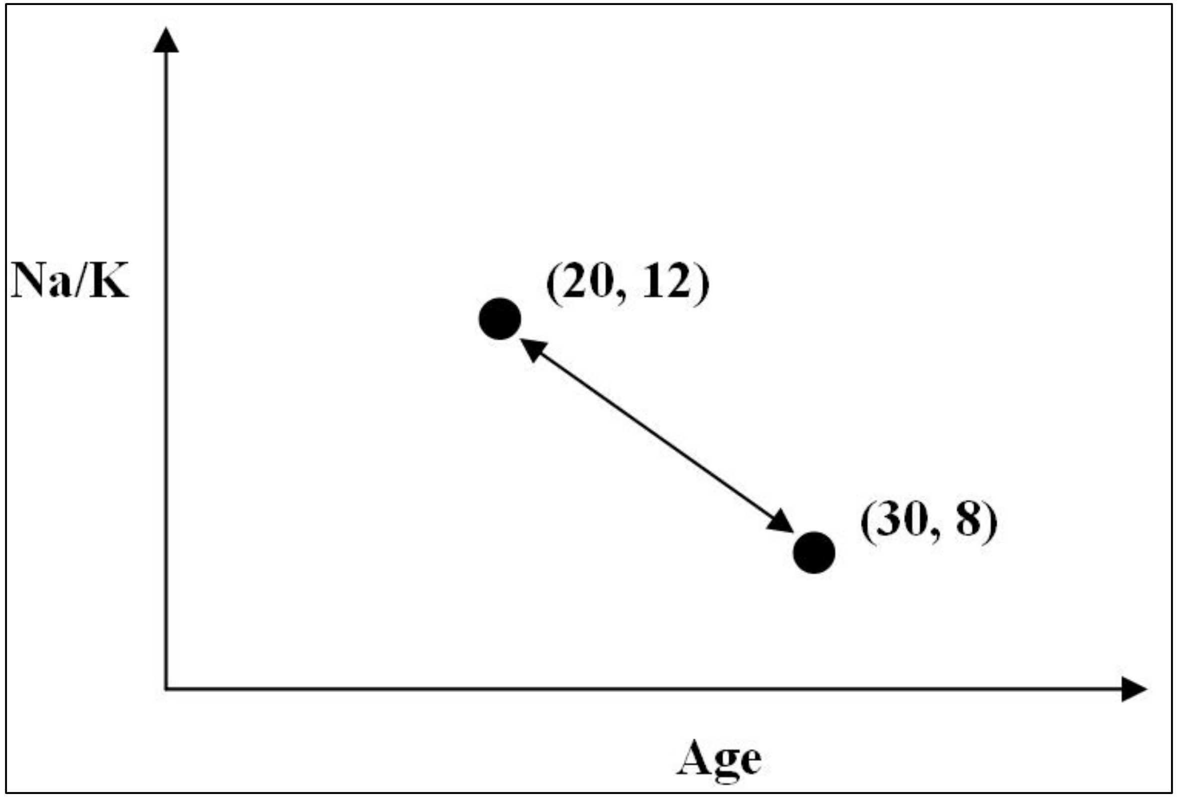
$$d_{Euclidean}(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

where **x** = $x_1, x_2, \ldots, x_m$, and **y** = $y_1, y_2, \ldots, y_m$ represent the *m* attribute values of two records

- Example
  - Suppose Patient A is 20-years-old and has a Na/K ratio = 12, and Patient B is 30-years-old and has a Na/K ratio = 8
  - What is the Euclidean distance between these instances?

Distance Function

$$d_{\text{Euclidean}}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2} = \sqrt{(20-30)^2 + (12-8)^2} = 10.77$$



More on Distance Metrics

- Euclidean is a type of Minkowski distance

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and h is the order (the distance so defined is also called L-*h* norm)

- Properties
  - d(i, j) > 0 if i ≠ j, and d(i, i) = 0 (Positive definiteness)
  - d(i, j) = d(j, i) (Symmetry)
  - d(i, j) ≤ d(i, k) + d(k, j) (Triangle Inequality)
- A distance that satisfies these properties is a metric

Applications of Minkowski Distance

- *h* = 1: Manhattan (city block, L_1 norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

- *h* = 2: (L_2 norm) Euclidean distance

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

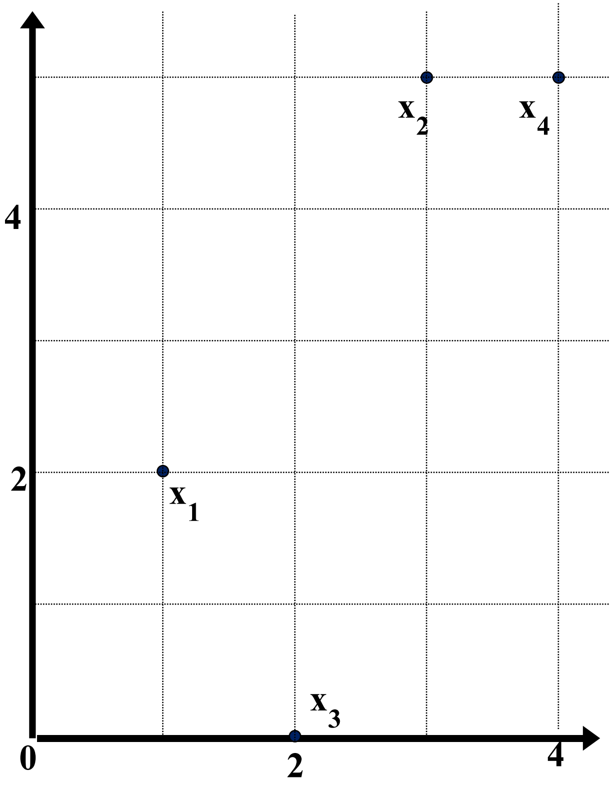
- $h \to \infty$. "supremum" (L_{max} norm, L_{∞} norm, Chebyshev) distance.
  - This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h \to \infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

$$D_{\text{Chess}} = \max\left(|x_2 - x_1|, |y_2 - y_1|\right).$$

Example: Minkowski Distance

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum (L_∞)

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

Distance Metrics in Practice

- Euclidean Distance: By far most common
  - Our intuitive notion of distance
- Manhattan Distance: Sometimes
- Rest: Very rare

What about categorical data?

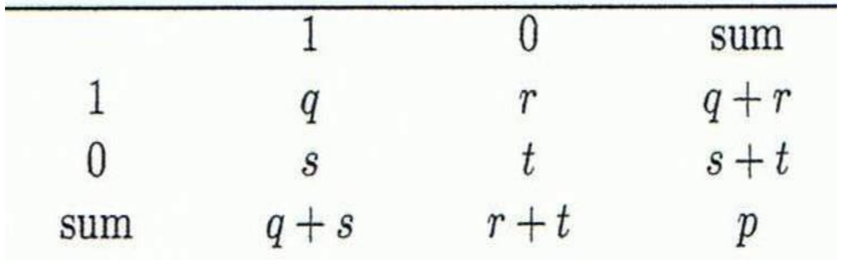
- For categorical attributes, the Euclidean Distance function is not appropriate
- Instead, we define a function called “different” (as seen with gender in previous example)

$$\text{Different}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

- We substitute *different(x_i, y_i)* for the *i*th term in the Euclidean distance metric above
- Example
  - Which patient is more similar to a 50-year-old male: a 20-year-old male or a 50-year-old female?

Proximity Measure for Binary Attributes

- A contingency table for binary data
- Count attributes = 1 and attributes = 0 for each object



		Object *j*		
		1	0	sum
Object *i*	1	q	r	$q+r$
	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

- Distance measure for symmetric binary variables:

$$d(i, j) = \frac{r+s}{q+r+s+t}$$

- Distance measure for asymmetric binary variables:

$$d(i, j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i, j) = \frac{q}{q+r+s}$$

Dissimilarity Between Binary Variables

- Example

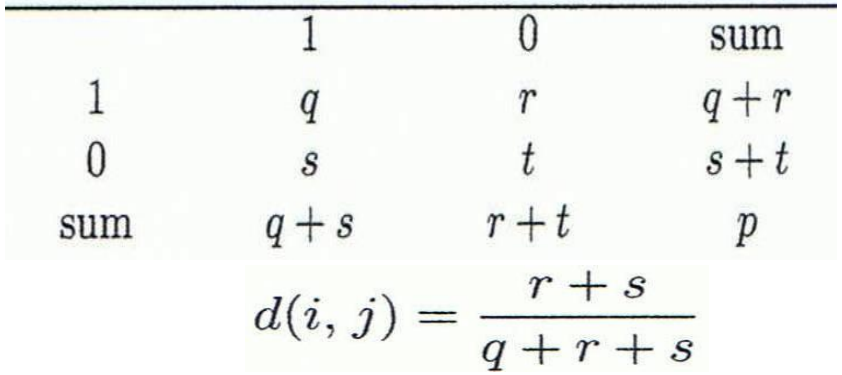
Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	Y	N	N	N	N

- Gender is a symmetric attribute
- The remaining attributes are asymmetric binary
- Do we need to use gender?
- Let the values Y and P = 1, and the value N = 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$



	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

$$d(i, j) = \frac{r+s}{q+r+s}$$

Normalization

- When measuring distance, one or more attributes can have very large values, relative to the other attributes
- For example, *income* may be scaled 30,000-100,000, whereas *years_of_service* takes on values 0-10
- In this case, the values of *income* will overwhelm the contribution of *years_of_service*
- To avoid this situation we use normalization
- Continuous data values should be normalized using Min-Max Normalization or Z-Score Standardization

Distance Example

- Let Patient A = 50-year-old male, Patient B = 20-year-old male, and Patient C = 50-year-old female
- Suppose that the *Age* variable has a range = 50, minimum = 10, mean = 45, and standard deviation = 15
- The table contains original, Min-Max Normalized, and Z-Score Standardized values for *Age*

Patient	Age	Age_{MMN}	$\text{Age}_{\text{Zscore}}$	Gender
A	50	$\frac{50-10}{50}=0.8$	$\frac{50-45}{15}=0.33$	Male
B	20	$\frac{20-10}{50}=0.2$	$\frac{20-45}{15}=-1.67$	Male
C	50	$\frac{50-10}{50}=0.8$	$\frac{50-45}{15}=0.33$	Female

Distance Function and Scaling

- Age not normalized
  - Assume we do not normalize *Age* and calculate the distance between Patient A and Patient B, and Patient A and Patient C

$$d(A,B)=\sqrt{(50-20)^2+0^2}=30$$

$$d(A,C)=\sqrt{(50-50)^2+1^2}=1$$

  - We determine, although perhaps incorrectly, that Patient C is nearest Patient A
  - Is Patient B really 30 times more distant than Patient C is to Patient A?
  - Perhaps neglecting to normalize the values of *Age* is creating this discrepancy?

Distance Function and Scaling

- Age Normalized using Min-Max
    - *Age* is normalized using Min-Max Normalization. Values lie in the range [0, 1]
    - Again, we calculate the distance between Patient A and Patient B, and Patient A and Patient C

$$d_{MMN}(A,B) = \sqrt{(0.8-0.2)^2 + 0^2} = 0.6$$

$$d_{MMN}(A,C) = \sqrt{(0.8-0.8)^2 + 1^2} = 1.0$$

- In this case, Patient B is now closer to Patient A

Distance Function and Scaling

- Age Standardized using Z-Score
  - This time, *Age* is standardized using Z-Score Standardization

$$d_{Zscore}(A,B) = \sqrt{(0.33-(-1.67))^2 + 0^2} = 2.0$$

$$d_{Zscore}(A,C) = \sqrt{(0.33-0.33)^2 + 1^2} = 1.0$$

  - Using Z-Score Standardization, most values are typically contained in the range [-3, 3]
  - Now, Patient C is nearest Patient A. This is different from the results obtained using Min-Max Normalization

Distance Function and Scaling

- Conclusion
    - The use of different normalization techniques resulted in Patient A being nearest to different patients in the training set
    - This underscores the importance of understanding which technique is being used
    - Note that the *distance(x,y)* and Min-Max Normalization functions produce values in the range [0, 1]
    - **Perhaps, when calculating the distance between records containing both numeric and categorical attributes, the use of Min-Max Normalization is preferred**

To Scale or Not to Scale

- If variables are not scaled
  - Variable with largest range has most weight
  - Distance depends on scale
- Scaling gives every variable equal weight
- Scale if
  - Variables measured in different units (kg, meter, sec)
  - You explicitly want to have equal weight for each variable
- Don’t scale if units are the same for all variables
- Most often: Better to scale…

Back to k-Nearest Neighbor

- We now have a method of determining which record are similar to new, unclassified records
- How should the most similar (*k*) records combine to provide a classification?
- Simple Unweighted Voting
  - This is the most simple combination function
  - Decide on the value for *k* to determine the number of similar records that “vote”
  - Compare each unclassified record to its *k* nearest (most similar) neighbors according to the Euclidean Distance function
  - Each of the *k* similar records vote

Back to k-Nearest Neighbor

- Recall that we classified a new Patient 2, 17-years-old with a Na/K ratio = 12.5, using *k* = 3 (Figure 7.2)
- Simple unweighted voting determined that two of the three closet points to Patient 2 are Medium
- Therefore, Patient 2 is classified as drug A or X with a <u>confidence</u> of 2/3 = 66.67%

- We also classified a new Patient 3, 47-years-old that has a Na/K ratio of 13.5, using *k* = 3 (Figure 7.3)
- However, simple unweighted voting did not help and resulted in a tie
- Perhaps weighted voting should be considered?

Weighted Voting

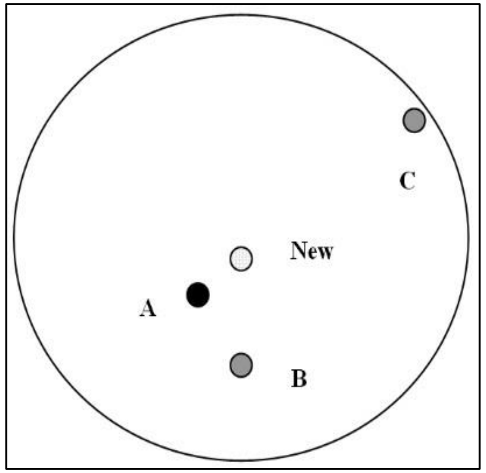
- Weighted Voting
  - In this case, the closer the neighbor, the more influence it has in the classification decision
  - This method assumes a closer neighbor is more similar, and therefore its vote should be weighted more heavily, as compared that of more distant neighbors
  - The weight of particular record is inversely proportional to its distance to the unclassified record
  - A “tie” is unlikely to occur using this approach

Weighted Voting

- Example
  - Again, recall that we classified a new patient 17-years-old with a Na/K ratio = 12.5, using *k* = 3
  - We determined, using unweighted voting, two of the closest points were Medium, and the third was Dark
  - However, the Dark point is the most similar to the new patient
  - Now, we reclassify the new patient using a weighted voting scheme using values from the table below

Table 7.3

Record	Age	Na/K	Age_{MMN}	Na/K_{MMN}
New Patient	17	12.5	0.05	0.25
A (Dark)	16.8	12.4	0.0467	0.2471
B (Med)	17.2	10.5	0.0533	0.1912
C (Med)	19.5	13.5	0.0917	0.2794



Weighted Voting

- The distance of records A, B, and C to the new patient are:

$$d(new, A) = \sqrt{(.05 - .0467)^2 + (.25 - .2471)^2} = .004393$$
$$d(new, B) = \sqrt{(.05 - .0533)^2 + (.25 - .1912)^2} = .058893$$
$$d(new, C) = \sqrt{(.05 - .0917)^2 + (.25 - .2794)^2} = .051022$$

- Next, the votes of these records are weighted according to the inverse square of their distance to the new record
- Record A votes to classify the new patient as Dark (drug B or C)

$$Votes\,(Dark\ \ Gray) = \frac{1}{d(new, A)^2} = \frac{1}{.004393^2} \cong 51{,}818.$$

Weighted Voting

- Records B and C vote to classify the new patient as Medium (drug A or X)

$$Votes(Medium\ Gray) = \frac{1}{d(new,B)^2} + \frac{1}{d(new,C)^2} = \frac{1}{.058893^2} + \frac{1}{.051022^2} \cong 672.$$

- Convincingly (51,818 vs. 672) the weighted voting method classifies the new patient as Dark (drug B or C)
- Note that this procedure reverses our classification decision determined using unweighted voting, *k* = 3
- The inverse distance of 0 is undefined using weighted voting
- Theoretically, the value of *k* could be increased, such that all training records participate in voting; however, the computational complexity may result in poor performance

Quantifying Attribute Relevance: Stretching the Axes

- Not all attributes may be relevant to classification
- For example, Decision Trees only include attributes that contribute to improving classification accuracy
- In contrast, *k*-Nearest Neighbor's default behavior is to calculate distances using all attributes
- A relevant record may be proximate for important variables, while at the same time very distant for other, unimportant variables
- Taken together, the relevant record may now be moderately far away from the new record, such that it does not participate in the classification decision

Quantifying Attribute Relevance: Stretching the Axes

- Perhaps, we should consider restricting the algorithm to using the most important fields for classification
- However, rather than making this determination *a priori*, we can make attributes either more, or less important
- This is accomplished using cross-validation or applying domain knowledge expertise
- Stretching the Axes
  - Stretching the Axes finds the coefficient z_j by which to multiply the j_{th} axis. Larger values of z_j are associated with the more important variable axes
- Cross-validation
  - Cross-validation selects a random subset of data from the training set and determines the set of z_1, z_2, ..., z_m that minimize the classification error on the test set

Quantifying Attribute Relevance: Stretching the Axes

- Repeating the process leads to a more accurate set of values for $z_1, z_2, ..., z_m$
- Domain Expertise
  - Alternately, we may call upon domain experts to recommend values for $z_1, z_2, ..., z_m$
  - Using either approach the *k*-Nearest Neighbor algorithm may be made more precise
- Example
  - Suppose that the Na/K ratio was determined to be <u>3 times more important</u> than the Age attribute, for performing drug classification

Quantifying Attribute Relevance: Stretching the Axes

- The distance of the records A, B, and C to the new record are calculated as follows:

$$\text{where } z_{Na/K} = 3,\ z_{Age} = 1$$

$$d(new, A) = \sqrt{(.05 - .0467)^2 + ((3)(.25 - .2471))^2} = .009305$$
$$d(new, B) = \sqrt{(.05 - .0533)^2 + ((3)(.25 - .1912))^2} = .17643$$
$$d(new, C) = \sqrt{(.05 - .0917)^2 + ((3)(.25 - .2794))^2} = .097561$$

- The classification does not change by stretching the axes for Na/K ratio
- In many situations, stretching the axes leads to improved accuracy by **quantifying the relevance** of each variable used in the classification decision

Database Considerations

- Instance-based learning methods benefit from having access to learning examples composed of many attribute value combinations
- The data set should be balanced to include a sufficient number of records with common, as well as less-common, classifications
- One approach to balancing the data set is to reduce the proportion of records with more common classifications
- Restrictions on main memory space may limit the size of the training set used
- The training set may be reduced to include only those records that occur near a classification “boundary”

Choosing *k*

- What value of *k* is optimal?
- There is not necessarily an obvious solution
- Smaller *k*
    - Choosing a small value for *k* may lead the algorithm to overfit the data
    - Noise or outliers may unduly affect classification
- Larger *k*
    - Larger values will tend to smooth out idiosyncratic or obscure data values in the training set
    - It the values become too large, locally interesting values will be overlooked
- Cross-validation alternative
    - Following procedure similar to the one mentioned earlier about finding optimal values for axis stretching
    - Iterating instead for finding the k that minimizes classification/estimation error