Data Mining I

Summer semester 2019

Lecture 6: Classification part 3

Lectures: Prof. Dr. Eirini Ntoutsi

TAs: Tai Le Quy, Vasileios Iosifidis, Maximilian Idahl, Shaheer Asghar, Wazed Ali
Overview of classification techniques covered in DM1

- Decision trees
- k nearest neighbours
- Bayesian classifiers
- Support vector machines

Evaluation of classifiers
Outline

- Lazy vs Eager Learners
- k-Nearest Neighbors classifiers (or, learning from your neighbors)
- Evaluation of classifiers
- Evaluation measures
- Evaluation setup
- Reading material
- Things you should know from this lecture
Lazy vs Eager learners

- **Eager learners**
  - Construct a classification model (based on a training set)
  - Learned models are ready and eager to classify previously unseen instances
  - e.g., decision trees, SVMs, MNB, neural networks

- **Lazy learners**
  - Simply store training data (with labels) and wait until a new instance arrives that needs to be classified
  - No model is constructed.
  - Known also as instance-based learners, because they store the training set
  - e.g., k-NN classifier

**Eager learners**
- Do lot of work on training data
- Do less work on classifying new instances

**Lazy learners**
- Do less work on training data
- Do more work on classifying new instances
Outline

- Lazy vs Eager Learners
- k-Nearest Neighbors classifiers (or, learning from your neighbors)
- Evaluation of classifiers
- Reading material
- Things you should know from this lecture
**k-Nearest Neighbor classifiers idea**

- Let $D$ be a training set of instances (i.e., with labels)
- Let $X$ be a new instance for which we cannot predict its class label
- Nearest-neighbor classifiers compare $X$ with its most similar training instances to decide on class
  - Basic idea: *If it walks like a duck, quacks like a duck, then it’s probably a duck*
**k-Nearest Neighbor (kNN) Classifiers**

**Input:**
- A training set $D$ (with known class labels)
- A distance measure to compute the distance between two instances
- The number of neighbors $k$

**Classification:** Given a new unknown instance $X$
- Compute distance to the training records, i.e., $\text{dist}(X,d) \in D$
- Identify the $k$ nearest neighbors
- Use the class labels of the $k$ nearest neighbors to determine the class label of $X$ (e.g., by taking majority vote)

**Complexity:** It requires $O(|D|)$ for each new instance
kNN algorithm

- Pseudocode:

```
Input:  
T      //training data  
K      //Number of neighbors  
t      //Input tuple to classify

Output:  
c      //Class to which t is assigned

KNN algorithm:  //Algorithm to classify tuple using KNN
begin
    N = ∅;
    //Find set of neighbors, N, for t
    for each d ∈ T do
        if |N| ≤ K
            then N = N ∪ {d};
        else if ∃ u ∈ N such that
            sim(t,u) ≤ sim(t,d) AND sim(t,u) ≤ sim(t,u’) ∀ u’ ∈ N
                then N = N − {u}; N = N ∪ {d};
    //Find class for classification
    c = class to which the most u ∈ N are classified
end
```

Think for 1’
Discuss with your neighbours
Discuss in the class

An example pseudocode is shown on the left.
How would you implement it?
Definition of nearest neighbors: parameter $k$

- Parameter $k$ defines the number of neighbors to be considered for the classification of an instance $X$ and therefore, its choice is critical for the performance of kNN
  - too small $k$: high sensitivity to outliers
  - too large $k$: many objects from other classes in the resulting neighborhood
  - average $k$: highest classification accuracy
Definition of nearest neighbors: parameter $k$

- An example with a too small $k$
  - Extreme case: $k=1$

$X$: unknown instance

Neighborhood for $k = 1$
Definition of nearest neighbors: parameter $k$

- An example with a too large $k$
  - Extreme case: $k = |D|$
Definition of nearest neighbors: parameter $k$

- An example with an average $k$

$X$: unknown instance

Neighborhood for $k = 7$

Why an average $k$ is best?
Definition of nearest neighbors: parameter $k$

- Overview of the effect of $k$

$X$: unknown instance

- Neighborhood for $k = 1$
- Neighborhood for $k = 7$
- Neighborhood for $k = 17$
Definition of nearest neighbors: distance function

- The $k$-nearest neighbors are selected among instances of the training set $D$
- “Closeness” is defined in terms of a distance/similarity measure
- For example, Euclidean distance (or $L_2$ norm)

\[
\text{dist}(x, y) = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}
\]

\[
dist(x, y) = \sqrt{d}
\]

<table>
<thead>
<tr>
<th>point</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>p3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>p4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Distance matrix

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2.828</td>
<td>3.162</td>
<td>5.099</td>
</tr>
<tr>
<td>p2</td>
<td>2.828</td>
<td>0</td>
<td>1.414</td>
<td>3.162</td>
</tr>
<tr>
<td>p3</td>
<td>3.162</td>
<td>1.414</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p4</td>
<td>5.099</td>
<td>3.162</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Common proximity measures (for numerical attributes)

- **Manhattan distance or City-block distance (L₁ norm)**
  - \( dist_1 = |p_1 - q_1| + |p_2 - q_2| + ... + |p_d - q_d| \)
  - The sum of the absolute differences of the \( p,q \) coordinates

- **Euclidean distance (L₂ norm)**
  - \( dist_2 = ( (p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_d - q_d)^2 )^{1/2} \)
  - The length of the line segment connecting \( p \) and \( q \)

- **Supremum distance (L_{max} norm or L_{∞} norm)**
  - \( dist_{∞} = \max\{|p_1 - q_1|, |p_2 - q_2|, ..., |p_d - q_d|\} \)
  - The max difference between any attributes of the objects.

- **Minkowski Distance (Generalization of L_p-distance)**
  - \( dist_p = ( |p_1 - q_1|^p + |p_2 - q_2|^p + ... + |p_d - q_d|^p )^{1/p} \)

Proximity measures for numerical attributes: examples

**Example**

<table>
<thead>
<tr>
<th>point</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>p3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>p4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

**Point coordinates**

**L1 distance matrix**

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>p2</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>p3</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p4</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**L2 distance matrix**

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2.828</td>
<td>3.162</td>
<td>5.099</td>
</tr>
<tr>
<td>p2</td>
<td>2.828</td>
<td>0</td>
<td>1.414</td>
<td>3.162</td>
</tr>
<tr>
<td>p3</td>
<td>3.162</td>
<td>1.414</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p4</td>
<td>5.099</td>
<td>3.162</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**L∞ distance matrix**

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>p2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>p3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Nearest neighbor classification: voting in the neighborhood

- The class of an unknown instance $X$ is determined from its neighbors.
- If $k=1$, the class is that of the closest instance.
- In the general case ($k>1$)
  - **Majority voting**: take the majority vote of class labels among the neighbors.
    - All neighbors contribute equally to the classification.
    - The algorithm is very sensitive to the choice of $k$.
  - **Weighted voting**: each neighbor $d$ contributes with a weight proportional to its distance from the unknown instance $X$.
    - e.g., a possible weight factor: $w = 1/\text{dist}(X,d)^2$.

Compare majority to weighted voting in the above figure. What do you expect?
### Nearest neighbor classification: an example

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Height</th>
<th>Output1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kristina</td>
<td>F</td>
<td>1.6m</td>
<td>Short</td>
</tr>
<tr>
<td>Jim</td>
<td>M</td>
<td>2m</td>
<td>Tall</td>
</tr>
<tr>
<td>Maggie</td>
<td>F</td>
<td>1.9m</td>
<td>Medium</td>
</tr>
<tr>
<td>Martha</td>
<td>F</td>
<td>1.88m</td>
<td>Medium</td>
</tr>
<tr>
<td>Stephanie</td>
<td>F</td>
<td>1.7m</td>
<td>Short</td>
</tr>
<tr>
<td>Bob</td>
<td>M</td>
<td>1.85m</td>
<td>Medium</td>
</tr>
<tr>
<td>Kathy</td>
<td>F</td>
<td>1.6m</td>
<td>Short</td>
</tr>
<tr>
<td>Dave</td>
<td>M</td>
<td>1.7m</td>
<td>Short</td>
</tr>
<tr>
<td>Worth</td>
<td>M</td>
<td>2.2m</td>
<td>Tall</td>
</tr>
<tr>
<td>Steven</td>
<td>M</td>
<td>2.1m</td>
<td>Tall</td>
</tr>
<tr>
<td>Debbie</td>
<td>F</td>
<td>1.8m</td>
<td>Medium</td>
</tr>
<tr>
<td>Todd</td>
<td>M</td>
<td>1.95m</td>
<td>Medium</td>
</tr>
<tr>
<td>Kim</td>
<td>F</td>
<td>1.9m</td>
<td>Medium</td>
</tr>
<tr>
<td>Amy</td>
<td>F</td>
<td>1.8m</td>
<td>Medium</td>
</tr>
<tr>
<td>Wynette</td>
<td>F</td>
<td>1.75m</td>
<td>Medium</td>
</tr>
<tr>
<td>Pat</td>
<td>F</td>
<td>1.6m</td>
<td>Short</td>
</tr>
</tbody>
</table>
Decision boundaries

- Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries
  - in contrary to e.g. decision trees that result in axis parallel hyper rectangles
Efficiency of kNN classifiers

- Lazy learners: no model is built explicitly, like in eager learners
- Classifying unknown records is relatively expensive
- Possible solutions:
  - Use index structures to speed up the nearest neighbors computation
  - Partial distance computation (projected distances) based on a subset of attributes

Different attributes have different value ranges

- e.g., height in [1.5m-1.8m]; income in [$10K - $1M]
- Distance measures might be dominated by one of the attributes
- Solution: normalization
Nearest neighbor classification issues 2

- The “curse of dimensionality”
  - Ratio of \( (D_{\text{max},d} - D_{\text{min},d}) \) to \( D_{\text{min},d} \) converges to zero with increasing dimensionality \( d \)
    - \( D_{\text{max},d} \): distance to the nearest neighbor in the \( d \)-dimensional space
    - \( D_{\text{min},d} \): distance to the farthest neighbor in the \( d \)-dimensional space
  - This implies that:
    - all points tend to be almost equidistant from another in high dimensional spaces
    - the distances between points cannot be used to differentiate between them
The curse of dimensionality

- Pairwise distances example: a sample of $10^5$ instances drawn from a uniform $[0, 1]$ distribution, normalized $(1/\sqrt{d})$.

Source: Tutorial on Outlier Detection in High-Dimensional Data, Zimek et al, ICDM 2012
Nearest neighbor classification issues 3

The “curse of dimensionality”

- Ratio of \( (D_{\text{max}_d} - D_{\text{min}_d}) \) to \( D_{\text{min}_d} \) converges to zero with increasing dimensionality \( d \)
  - \( D_{\text{max}_d} \): distance to the nearest neighbor in the \( d \)-dimensional space
  - \( D_{\text{min}_d} \): distance to the farthest neighbor in the \( d \)-dimensional space

- This implies that:
  - all points tend to be almost equidistant from another in high dimensional spaces
  - the distances between points cannot be used to differentiate between them

- Possible solutions:
  - Dimensionality reduction (e.g., PCA)
  - Work with a subset of dimensions instead of the complete feature space

More on this topic in DM2 lecture next semester!
k-NN classifiers: overview

- (+-) Lazy learners: Do not require building a model, but testing is more expensive
- (-) Classification is based on local information in contrast to e.g. DTs that try to find a global model that fits the entire input space: Susceptible to noise
- (+) Incremental classifiers → training set can be updated
- (-) The choice of distance function and parameter $k$ is important
- (+) Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries, in contrary to e.g. decision trees that result in axis parallel hyper rectangles
Normalization

- Attributes with large ranges outweigh ones with small ranges
  - e.g. income [10.000-100.000]; age [10-100]

- To balance the “contribution” of an attribute $A$ in the resulting distance, the attributes are scaled to fall within a small, specified range.

- min-max normalization: rescales the feature values into a new range $[new_{\text{min}}_A, new_{\text{max}}_A]$

  \[
  v' = \frac{v - \text{min}_A}{\text{max}_A - \text{min}_A} (new_{\text{max}}_A - new_{\text{min}}_A) + new_{\text{min}}_A
  \]

  - $v$ is the current feature value

Normalize age = 30 in the [0-1] range, given $\text{min}_{\text{age}}=10$, $\text{max}_{\text{age}}=100$

new_{age}=(30-10)/(100-10))*(1-0)+0=2/9
z-score normalization also called zero-mean normalization or standardization: Transform the data by converting the values to a common scale with a mean of zero ($\mu=0$) and a standard deviation of one ($\sigma=1$) (unit variance).

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

where $mean_A$, $stand\_dev_A$ are the mean and standard deviation of the feature

Normalize income = 70,000 if $mean_{\text{income}}=50,000$, $stand\_dev_{\text{income}}=15,000$

$$new\_value = \frac{70,000 - 50,000}{15,000} = 1.33$$
What is the inductive bias for kNN classification?

- Think for 1’
- Discuss with your neighbours
- Discuss in the class

*Inductive bias:* the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances.

(Recall Lecture 5)
Outline

- Lazy vs Eager Learners
- k-Nearest Neighbors classifiers (or, learning from your neighbors)
  - Evaluation of classifiers
- Evaluation measures
- Evaluation setup
- Reading material
- Things you should know from this lecture
Evaluation of classifiers: The role of the test set

Data Mining I @SS19, Lectures 6: Classification part 3

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrb1</th>
<th>Attrb2</th>
<th>Attrb3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrb1</th>
<th>Attrb2</th>
<th>Attrb3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Evaluation of classifiers

- The quality of a classifier is evaluated over a test set, different from the training set
  - For each instance in the test set, we know its true class label
  - We compare the predicted class (by some classifier) with the true class of the test instances

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Test set

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D16</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D17</td>
<td>Overcast</td>
<td>High</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D18</td>
<td>Sunny</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D19</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Fundamental assumptions

- Training and testing sets come from the same distribution
  - The data stationarity assumption
- Testing instances are not seen during training
  - the training and testing sets are disjoint

Why the stationarity assumption is important?
What if it is violated? E.g. Twitter stream

What if they are not disjoint? How this would affect evaluation?

More on this topic in DM2 lecture next semester!
A useful tool for analyzing how well a classifier performs is the **confusion matrix**

- The confusion matrix is built from the **test set** and the predictions of the classifier
- For an $m$-class problem, the matrix is of size $m \times m$

**Terminology**

- Positive tuples: tuples of the main class of interest
- Negative tuples: all other tuples

An example of a matrix for a 2-class problem:

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>D16</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D17</td>
<td>Overcast</td>
<td>High</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>D18</td>
<td>Sunny</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>D19</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C₁</strong></td>
<td><strong>C₂</strong></td>
<td>totals</td>
</tr>
<tr>
<td>TP (true positive)</td>
<td>FN (false negative)</td>
<td>P</td>
</tr>
<tr>
<td>FP (false positive)</td>
<td>TN (true negative)</td>
<td>N</td>
</tr>
<tr>
<td>Totals</td>
<td>P'</td>
<td>N'</td>
</tr>
</tbody>
</table>
Outline

- Lazy vs Eager Learners
- k-Nearest Neighbors classifiers (or learning from your neighbors)
- Evaluation of classifiers
  - Evaluation measures
- Evaluation setup
- Reading material
- Things you should know from this lecture
Classifier evaluation measures

- Accuracy
- Error rate
- Sensitivity
- Specificity
- Precision
- Recall
- F-measure
- $F_\beta$-measure
- ...

Data Mining I @SS19, Lectures 6: Classification part 3
Classifier evaluation measures 1/4

- **Accuracy/ Recognition rate:**
  
  % of test set instances correctly classified

  \[
  accuracy(M) = \frac{TP + TN}{P + N}
  \]

- **Error rate/ Missclassification rate:** \( error\_rate(M) = 1 - accuracy(M) \)

  \[
  error\_rate(M) = \frac{FP + FN}{P + N}
  \]

- These measures are more effective when the class distribution is relatively balanced

---

### Data Mining I @SS19, Lectures 6: Classification part 3

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
<th>totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>TP (true positive)</td>
<td>FN (false negative)</td>
<td>( P )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>FP (false positive)</td>
<td>TN (true negative)</td>
<td>( N )</td>
</tr>
<tr>
<td>Totals</td>
<td>( P' )</td>
<td>( N' )</td>
<td></td>
</tr>
</tbody>
</table>

### Table

<table>
<thead>
<tr>
<th>Actual class</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 ) buy_computer = yes</td>
<td>6954</td>
<td>46</td>
<td>7000</td>
</tr>
<tr>
<td>buy_computer = no</td>
<td>412</td>
<td>2588</td>
<td>3000</td>
</tr>
<tr>
<td>total</td>
<td>7366</td>
<td>2634</td>
<td>10000</td>
</tr>
</tbody>
</table>

- **Accuracy of the model:**
  
  \( \text{Accuracy}(M) = 95.42\% \)

- **Error rate of the model:**
  
  \( \text{Error\_rate}(M) = 4.58\% \)

---

What is the accuracy of the model?

Accuracy\( (M) = 95.42\% \)

What is the error rate of the model?

Error\_rate\( (M) = 4.58\% \)
Limitations of accuracy and error rate

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9%
  - Accuracy is misleading because model does not detect any class 1 examples

!!! Accuracy and error rate are more effective when the class distribution is relatively *balanced*
Classifier evaluation measures 2/4

If classes are imbalanced:

- **Sensitivity/ True positive rate/ recall:**
  % of positive tuples that are correctly recognized

\[
sensitivity(M) = \frac{TP}{P}
\]

- **Specificity/ True negative rate:** % of negative tuples that are correctly recognized

\[
specificity(M) = \frac{TN}{N}
\]

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>Actual class</th>
<th>buy_computer = yes</th>
<th>buy_computer = no</th>
<th>total</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy_computer = yes</td>
<td>6954</td>
<td>46</td>
<td>7000</td>
<td>99.34</td>
<td></td>
</tr>
<tr>
<td>buy_computer = no</td>
<td>412</td>
<td>2588</td>
<td>3000</td>
<td>86.27</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>7366</td>
<td>2634</td>
<td>10000</td>
<td>95.42</td>
<td></td>
</tr>
</tbody>
</table>

What is the sensitivity/specificity of the model?

- Accuracy(M)=95.42%
- sensitivity(M)=99.34%
- specificity(M)=86.27%

Data Mining I @SS19, Lectures 6: Classification part 3
Classifier evaluation measures 3/4

- **Precision**: % of tuples labeled as positive which are actually positive
  
  \[ \text{precision}(M) = \frac{TP}{TP + FP} \]

- **Recall**: % of positive tuples labeled as positive
  
  \[ \text{recall}(M) = \frac{TP}{TP + FN} = \frac{TP}{P} \]

  - Precision biased towards TP and FP
  - Recall biased towards TP and FN
  - Higher precision → less FP
  - Higher recall → less FN

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>C1</th>
<th>C2</th>
<th>totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>TP (true positive)</td>
<td>FN (false negative)</td>
<td>P</td>
</tr>
<tr>
<td>C2</td>
<td>FP (false positive)</td>
<td>TN (true negative)</td>
<td>N</td>
</tr>
<tr>
<td>Totals</td>
<td>P'</td>
<td>N'</td>
<td></td>
</tr>
</tbody>
</table>

Recall the definition of precision/recall in IR:
- Precision: % of selected items that are correct
- Recall: % of correct items that are selected

What is the precision/recall of the model?

- \( \text{precision}(M) = 94.41\% \)
- \( \text{recall}(M) = 99.34\% \)
Classifier evaluation measures 4/4

- **F-measure / $F_1$ score / F-score** combines both

\[
F(M) = \frac{2 \times \text{precision}(M) \times \text{recall}(M)}{\text{precision}(M) + \text{recall}(M)}
\]

It is the harmonic mean of precision and recall

- **$F_\beta$-measure** is a weighted measure of precision and recall

\[
F_\beta(M) = \frac{(1 + \beta^2) \times \text{precision}(M) \times \text{recall}(M)}{\beta^2 \times \text{precision}(M) + \text{recall}(M)}
\]

Common values for $\beta$:
- $\beta=1 \rightarrow F_1$
- $\beta=0.5$

- For our example, $F(M)=2\times94.41\%\times99.34\%/(94.41\%+99.34\%)=96.81\%$

Data Mining I @SS19, Lectures 6: Classification part 3

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>TP (true positive)</td>
<td>FN (false negative)</td>
</tr>
<tr>
<td>FP (false positive)</td>
<td>TN (true negative)</td>
</tr>
<tr>
<td>Totals</td>
<td>$P'$</td>
</tr>
</tbody>
</table>

More on harmonic mean: [http://mathworld.wolfram.com/HarmonicMean.html](http://mathworld.wolfram.com/HarmonicMean.html)
Classifier evaluation measures: ROC curve 1/3

- ROC: Receiver Operating Characteristic curve
  - TPR is plotted on the Y axis
    - TPR (TP rate): % of positive tuples that are correctly recognized
      - TPR = TP/P = TP/(TP+FN), i.e., sensitivity
  - FPR is plotted on the X axis
    - FPR (FP rate): % of negative tuples that are incorrectly classified
      - FPR = FP/N = FP/(FP + TN)
  - It depicts relative trade-offs between benefits (true positives) and costs (false positives).

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_1$</td>
</tr>
<tr>
<td>$C_1$</td>
<td>TP (true positive)</td>
</tr>
<tr>
<td>$C_2$</td>
<td>FP (false positive)</td>
</tr>
<tr>
<td>Totals</td>
<td>$P'$</td>
</tr>
</tbody>
</table>
Classifier evaluation measures: ROC curve 2/3

- **Perfect classification:**
  - TPR = 1.0 (no false negatives)
  - FPR = 0.0 (no false positives)

- **Random classification** (diagonal = *line of no-discrimination*):
  - TPR = 0.5
  - FPR = 0.5

- **In practice:**
  - Points above the diagonal represent good classification results (better than random)
  - Points below the line represent poor results (worse than random)

- Used to evaluate the performance of different classifiers

Adopted by
https://en.wikipedia.org/wiki/Receiver_operating_characteristic
How to create a ROC curve?
A classifier produces a single ROC point.
If the classifier has a “sensitivity” parameter, varying it produces a series of ROC points (confusion matrices).
Alternatively, a series of ROC points can be generated by varying the class ratio in the training set.

A ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. [Wikipedia]
Lazy vs Eager Learners
k-Nearest Neighbors classifiers (or, learning from your neighbors)
Evaluation of classifiers
Evaluation measures
Evaluation setup
Reading material
Things you should know from this lecture
Evaluation setup

- How to create the training and test sets out of a dataset?
  - We don’t want to make unreasonable assumptions about our population

- Many approaches
  - Holdout
  - Cross-validation
  - Bootstrap
  - ....
**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
- (+) It takes no longer to compute
- (-) it depends on how data are divided
- (-) You have less data for training, problematic for small datasets.

**Random sampling**: a variation of holdout
- Repeat holdout $k$ times, accuracy is the *avg* accuracy obtained

---

Data Mining I @SS19, Lectures 6: Classification part 3
Cross-validation ($k$-fold cross validation, $k = 10$ usually)

- Randomly partition the data into $k$ mutually exclusive subsets $D_1, \ldots, D_k$ each approximately equal size
- Training and testing is performed $k$ times
  - At the $i$-th iteration, use $D_i$ as test set and rest as training set
- Each point is in a test set 1 time and in a training set $k-1$ times
- Accuracy is the avg accuracy over all iterations
- (+) Does not rely so much on how data are divided
- (-) The algorithm should re-run from scratch $k$ times

Leave-one-out: $k$-folds with $k = \#$of tuples, so only one sample is used as a test set at a time;
  - for small sized data
Stratified sampling vs random sampling

- Stratified sampling creates a mini-reproduction of the population in terms of the class labels. E.g., if 25% of the population belongs to the class “blue”, 25% to class “green” and 50% to class “red” then 25% of the sample is drawn randomly from class “blue”, 25% from class “green” and 50% from class “red”.

Stratified cross-validation: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data

- Stratified 10 fold cross-validation is recommended!!!
Evaluation setup 4/5

- **Bootstrap**: Samples the given training data 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
  - Works well with small data sets

- Several bootstrap methods, and a common one is **.632 bootstrap**
  - Suppose we are given a dataset of $n$ tuples. The data set is sampled $n$ times, with replacement, resulting in a training set of $n$ samples (known also as bootstrap sample):
  - The data tuples that did not make it into the training set end up forming the test set.
  - Each sample has a probability $1/n$ of being selected and $(1-1/n)$ of not being chosen. We repeat $n$ times, so the probability for a tuple to not be chosen during the whole period is $(1-1/n)^n$.
    - For large $d$: $\left(1 - \frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$
  - So on average, 36.8% of the tuples will not be selected for training and thereby end up in the test set; the remaining 63.2% will form the train test
Evaluation setup 5/5

- Repeat the sampling procedure $k$ times $\rightarrow k$ bootstrap datasets
- Report the overall accuracy of the model:

$$acc_{boot}(M) = \frac{1}{k} \sum_{i=1}^{k} (0.632 \times acc(M_i)_{testSet_i} + 0.368 \times acc(M_i)_{train\_set_i})$$

Accuracy of the model obtained by bootstrap sample $i$ when it is applied on test set $i$.

Accuracy of the model obtained by bootstrap sample $i$ when it is applied over all labeled data

- often called training error
The role of the validation set

- If model selection and error estimates are to be computed simultaneously, the data should be divided into three disjoint datasets.

  - **Training set** is a set of examples used for learning a model.
  - **Validation set** is a set of examples that cannot be used for learning the model but can help tune model parameters (e.g., selecting $K$ in K-NN).
  - **Test set** is used only to assess the performance of the final model and provide an estimation of the generalization error.

- **Why separate test and validation sets?**
  - The error rate estimate of the final model on validation data will be biased (smaller than the true error rate) since the validation set is used to select the final model.
  - After assessing the final model on the test set, you MUST NOT tune the model any further.
The role of the validation set

- **Procedure outline (for holdout)**
  1. Divide the available data into training, validation and test set.
  2. Select parameters
  3. Train the model using the training set
  4. Evaluate the model using the validation set
  5. Repeat steps 2 through 4 using different parameters
  6. Select the best model and train it using data from the training and validation sets
  7. Assess this final model using the test set

- If cross validation or bootstrap are used, steps 3,4 have to be repeated for each of the $k$ folds.
Train, Validation and Test in practice

Source: CSCE 666 Pattern Analysis | Ricardo Gutierrez-Osuna | CSE@TAMU
Classifier evaluation summary

- Evaluation measures
  - accuracy, error rate, sensitivity, specificity, precision, F-score, $F_{\beta}$, ROC

- Train – test splitting
  - Holdout, cross-validation, bootstrap,...

- Other parameters that might influence our decision on which model is best for our problem
  - Speed (construction time, usage time)
  - Robustness to noise, outliers and missing values
  - Scalability for large data sets
  - Interpretability (by humans)
Outline

- Lazy vs Eager Learners
- k-Nearest Neighbors classifiers (or learning from your neighbors)
- Evaluation of classifiers
- Evaluation measures
- Evaluation setup
- Reading material
- Things you should know from this lecture
Reading material

- Reading material for this lecture:
  - Lazy learners – KNN – Section 5.2, Tan et al book
  - Evaluation of classifiers – Section 4.5, Tan et al book

- Reading material for next week:
  - Bayesian classifiers Section 5.3
  - Artificial Neural Networks – Perceptron Section 5.4
Lazy vs Eager Learners

k-Nearest Neighbors classifiers (or learning from your neighbors)

Evaluation of classifiers

Evaluation measures

Evaluation setup

Reading material

Things you should know from this lecture
Things you should know from this lecture

- Lazy vs Eager classifiers
- kNN classifiers
  - Effect of k
  - Distance function and its effect
- Evaluation measures
- Evaluation setup
- How to choose a classifier for your problem at hand
The slides are based on

- KDD I lecture at LMU Munich (Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Eirini Ntoutsi, Jörg Sander, Matthias Schubert, Arthur Zimek, Andreas Züfle)

- Introduction to Data Mining book slides at http://www-users.cs.umn.edu/~kumar/dmbook/

- Pedro Domingos Machine Lecture course slides at the University of Washington


- Thank you to all TAs contributing to their improvement, namely Vasileios Iosifidis, Damianos Melidis, Tai Le Quy, Han Tran.