Data Mining I
Classification, Part 1

Heiko Paulheim
Outline

1. What is Classification?
2. k Nearest Neighbors and Nearest Centroids
3. Naïve Bayes
4. Evaluating Classification
5. Decision Trees
6. The Overfitting Problem
7. Other Classification Approaches
8. Hyperparameter Tuning
A Couple of Questions

• What is this?
• Why do you know?
• How have you come to that knowledge?
Introductory Example

- Learning a new concept, e.g., "Tree"

"tree"  "tree"  "tree"

"not a tree"  "not a tree"  "not a tree"
Introductory Example

• Example: learning a new concept, e.g., "Tree"
  – we look at (positive and negative) examples
  – ...and derive a model
    • e.g., "Trees are big, green plants"

• Goal: Classification of new instances

"tree?"

Warning: Models are only approximating examples! Not guaranteed to be correct or complete!
What is Classification?

- Classic programming:
  - if more than 10 orders/year and more than $100k spent
    set customer.isPremiumCustomer = true

- The prevalent style of programming computers
  - works well as long as we know the rules
  - e.g.: what makes a customer a premium customer?
What is Classification?

- Sometimes, it's not so easy
- E.g., due to missing knowledge
  - if customer is likely to order a new phone
    send advertisement for new phones
- E.g., due to difficult formalization as an algorithm
  - if customer review is angry
    send apology
What is Classification?

• A different paradigm:
  – User provides computer with examples
  – Computer finds model by itself
  – Notion: the computer learns from examples (term: machine learning)

• Example
  – labeled examples of angry and non-angry customer reviews
  – computer finds model for telling if a customer is angry
Classification: Formal Definition

- **Given:**
  - a set of labeled records, consisting of
    - data fields (a.k.a. attributes or features)
    - a class label (e.g., true/false)

- **Generate**
  - a function $f(r)$
    - input: a record
    - output: a class label
  - which can be used for classifying previously unseen records

- **Variants:**
  - single class problems (e.g., only true/false)
  - multi class problems
  - multi label problems (more than one class per record, not covered in this lecture)
  - hierarchical multi class/label problems (with class hierarchy, e.g., product categories)
The Classification Workflow

![Diagram showing the classification workflow with tables for training set and unseen records, and processes for learning, applying the model, and deduction.]
Classification Applications – Examples

• Attributes: a set of symptoms (cough, sore throat…)  
  – class: does the patient suffer from CoViD-19?

• Attributes: the values in your tax declaration  
  – class: are you trying to cheat?

• Attributes: your age, income, debts, …  
  – class: are you getting credit by your bank?

• Attributes: the countries you phoned with in the last 6 months  
  – class: are you a terrorist?
Classification Applications – Examples

• Attributes: words in a product review
  – Class: Is it a fake review?

• Attributes: words and header fields of an e-mail
  – Class: Is it a spam e-mail?
Classification Applications – Examples

- A controversial example
  - Class: whether you are searched by the police
  - Class: whether you are selected at the airport for an extra check

[Graph showing Lubbock traffic stop racial profiling report]
Classification Algorithms

• Recap:
  – we give the computer a set of labeled examples
  – the computer learns to classify new (unlabeled) examples

• How does that work?
k Nearest Neighbors

- Problem
  - find out what the weather is in a certain place
  - where there is no weather station
  - how could you do that?
k Nearest Neighbors

- Idea: use the average of the nearest stations
- Example:
  - 3x sunny
  - 2x cloudy
  - result: sunny
- Approach is called
  - “k nearest neighbors”
  - where k is the number of neighbors to consider
  - in the example: k=5
  - in the example: “near” denotes geographical proximity
k Nearest Neighbors

- Further examples:
  - Is a customer going to buy a product?
    → have similar customers bought that product?
  - What party are you going to vote for?
    → what party do your (closest) friends/family members vote for?
  - Is a film going to win an oscar?
    → have similar films won an oscar?

- and so on...
Recap: Similarity and Distance

- **k Nearest Neighbors**
  - requires a notion of similarity (i.e., what is “near”?)
- **Review: similarity measures for clustering**
  - similarity of individual data values
  - similarity of data points
- **Think about scales and normalization!**
Nearest-Neighbor Classifiers

• Requires three things
  – The set of stored records
  – A distance metric to compute distance between records
  – The value of $k$, the number of nearest neighbors to retrieve
Nearest-Neighbor Classifiers

To classify an unknown record:
- Compute distance to each training record
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record
  - by taking majority vote
  - by weighing the vote according to distance
Definition of the k Nearest Neighbors

The k nearest neighbors of a record \( x \) are data points that have the \( k \) smallest distance to \( x \).

- (a) 1-nearest neighbor
- (b) 2-nearest neighbor
- (c) 3-nearest neighbor
Choosing a Good Value for 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: Test k values between 1 and 10.
Discussion of K-Nearest Neighbor

- Often very accurate
- ... but slow as training data needs to be searched

- Can handle decision boundaries which are not parallel to the axes
- Assumes all attributes are equally important
  - Remedy: Attribute selection or using attribute weights
Decision Boundaries of a k-NN Classifier

- $k=1$
- Single noise points have influence on model
Decision Boundaries of a k-NN Classifier

- $k = 3$
- Boundaries become smoother
- Influence of noise points is reduced
scaler = MinMaxScaler()
features_norm = scaler.fit_transform(features)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(features_norm, label)
Applying the Model

test_norm = scaler.transform(test)
model.predict(test_norm)
Contrast: Nearest Centroids

• a.k.a. Rocchio classifier
• Training: compute centroid for each class
  – center of all points of that class
  – like: centroid for a cluster
• Classification:
  – assign each data point to nearest centroid

• RapidMiner:
  – available in Mannheim RapidMiner Toolbox Extension
• Python:
  – scikit_learn.neighbors.NearestCentroid

Sounds pretty much just like k-NN, huh?
k-NN vs. Nearest Centroid

- Basic problem: two circles
  - Both k-NN and Nearest Centroid are rather perfect
k-NN vs. Nearest Centroid

- Some data points are mislabeled
  - k-NN loses performance
  - Nearest Centroid is stable
k-NN vs. Nearest Centroid

- One class is significantly smaller than the other
  - k-NN loses performance
  - Nearest Centroid is stable
k-NN vs. Nearest Centroid

- Outliers are contained in the dataset
  - k-NN is stable
  - Nearest Centroid loses performance
k-NN vs. Nearest Centroid

• k-NN
  – slow at classification time (linear in number of data points)
  – requires much memory (storing all data points)
  – robust to outliers

• Nearest Centroid
  – fast at classification time (linear in number of classes)
  – requires only little memory (storing only the centroids)
  – robust to label noise
  – robust to class imbalance

• Which classifier is better?
  – that strongly depends on the problem at hand!
Bayes Classifier

• Based on Bayes Theorem
• Thomas Bayes (1701-1761)
  – British mathematician and priest
  – tried to formally prove the existence of God
• Bayes Theorem
  – important theorem in probability theory
  – was only published after Bayes' death
Conditional Probability and Bayes Theorem

• A probabilistic framework for solving classification problems

• Conditional Probability:

\[
P(C \mid A) = \frac{P(A, C)}{P(A)}
\]

\[
P(A \mid C) = \frac{P(A, C)}{P(C)}
\]

• Bayes theorem:

\[
P(C \mid A) = \frac{P(A \mid C) P(C)}{P(A)}
\]
Conditional Probability and Bayes Theorem

• Bayes Theorem
  – Computes one conditional probability $P(\text{C}|\text{A})$ out of another $P(\text{A}|\text{C})$
  – given that the base probabilities $P(\text{A})$ and $P(\text{C})$ are known

• Useful in situations where $P(\text{C}|\text{A})$ is unknown
  – while $P(\text{A}|\text{C})$, $P(\text{A})$ and $P(\text{C})$ are known or easy to determine/estimate

• Example:
  – Given a symptom, what's the probability that I have a certain disease?
Example of Bayes Theorem

- PCR test for SaRS-CoV-2
  - exact quality is unknown

- Optimistic estimates\(^1\)
  - If you're infected, PCR shows a positive result with \(p=95\%\) (called “sensitivity”)
  - If you're not infected, PCR shows a negative result also with \(p=95\%\) (called “specificity”)

- Assume you have a positive test
  - What's the probability that you're infected with SARS-CoV-2?

\(^1\)see https://www.nejm.org/doi/full/10.1056/NEJMp2015897
Example of Bayes Theorem

- We want to know $P(\text{Corona}|\text{pos})$
  - Bayes theorem:
    $$P(\text{Cor}|\text{pos}) = \frac{P(\text{pos}|\text{Cor}) P(\text{Cor})}{P(\text{pos})}$$

- We still need $P(\text{pos})$
  - i.e., the probability that a test is positive
    $$P(\text{pos}) = P(\text{pos}|\text{Cor} \lor \neg \text{Cor})$$
    $$= P(\text{pos}|\text{Cor}) \cdot P(\text{Cor}) + P(\text{pos}|\neg \text{Cor}) \cdot P(\neg \text{Cor})$$

~3% in Germany
Example of Bayes Theorem

• Now: numbers

\[ P(\text{Corona} \mid \text{pos}) = \frac{P(\text{pos} \mid \text{Corona}) \cdot P(\text{Corona})}{P(\text{pos})} \]

\[ = \frac{P(\text{pos} \mid \text{Corona}) \cdot P(\text{Corona})}{P(\text{pos} \mid \text{Cor}) \cdot P(\text{Cor}) + P(\text{pos} \mid \text{\neg Cor}) \cdot P(\text{\neg Cor})} \]

\[ = \frac{0.95 \cdot 0.03}{0.95 \cdot 0.03 + 0.05 \cdot 0.97} = 0.37 \]

• That means:
  – at more than 65% probability, you are still healthy, given a positive test!

• Caveat:
  – numbers P(Cor) and (P\neg Cor) are different due to non-random testing!
The prior probability $P(C_j)$ for each class is estimated by

1. counting the records in the training set that are labeled with class $C_j$
2. dividing the count by the overall number of records

Example:
- $P(\text{Play}=\text{no}) = \frac{5}{14}$
- $P(\text{Play}=\text{yes}) = \frac{9}{14}$
Estimating the Conditional Probability $P(A \mid C)$

• Naïve Bayes assumes that all attributes are statistically independent
  – knowing the value of one attribute says nothing about the value of another
  – this independence assumption is almost never correct!
  – but … this scheme works well in practice

• The independence assumption allows the joint probability $P(A \mid C)$ to be reformulated as the product of the individual probabilities $P(A_i \mid C_j)$:

$$P(A_1, A_2, \ldots, A_n \mid C_j) = \prod P(A_n \mid C_j) = P(A_1 \mid C_j) \times P(A_2 \mid C_j) \times \ldots \times P(A_n \mid C_j)$$

$P(\text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool} \mid \text{Play}=\text{yes})$
  $= P(\text{Outlook}=\text{rainy} \mid \text{Play}=\text{yes}) \times P(\text{Temperature}=\text{cool} \mid \text{Play}=\text{yes})$

• Result: The probabilities $P(A_i \mid C_j)$ for all $A_i$ and $C_j$ can be estimated directly from the training data
Estimating the Probabilities $P(A_i | C_j)$

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Yes 2 No 3</td>
<td>Hot 2 Yes 3 No 4</td>
<td>False 6 Yes 2</td>
<td>Sunny 2/9 No 3/5</td>
</tr>
<tr>
<td>Overcast</td>
<td>Yes 4 No 0</td>
<td>Mild 4 Yes 2 No 1</td>
<td>True 3 Yes 3</td>
<td>Overcast 4/9 No 0/5</td>
</tr>
<tr>
<td>Rainy</td>
<td>Yes 3 No 2</td>
<td>Cool 3 Yes 1 No 1</td>
<td>False 6 No 2</td>
<td>Rainy 3/9 No 2/5</td>
</tr>
</tbody>
</table>

1. Count how often an attribute value co-occurs with class $C_j$
2. Divide by the overall number of instances in class $C_j$

Example:

“Outlook=sunny” occurs on 2/9 examples in class “Yes”

$\Rightarrow p(\text{Outlook=sunny}|\text{Yes}) = \frac{2}{9}$
Classifying a New Record

Unseen record

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>High</td>
<td>True</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
P(\text{yes} \mid E) = P(\text{Outlook} = \text{Sunny} \mid \text{yes}) \\
\times P(\text{Temperature} = \text{Cool} \mid \text{yes}) \\
\times P(\text{Humidity} = \text{High} \mid \text{yes}) \\
\times P(\text{Windy} = \text{True} \mid \text{yes}) \\
\times \frac{P(\text{yes})}{P(E)} \\
= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} \\
\]
Classifying a New Record (ctd.)

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>2/9</td>
<td>3/9</td>
<td>6/9</td>
<td>2/5</td>
</tr>
<tr>
<td>Overcast</td>
<td>4/9</td>
<td>0/5</td>
<td>2/5</td>
<td>1/5</td>
</tr>
<tr>
<td>Rainy</td>
<td>3/9</td>
<td>2/5</td>
<td>3/9</td>
<td>1/5</td>
</tr>
</tbody>
</table>

A new day:

Prior probability
Evidence

Choose Maximum

Likelihood of the two classes

For “yes” = \(2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053\)

For “no” = \(3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206\)

Conversion into a probability by normalization:

\[ P(“yes”) = \frac{0.0053}{0.0053 + 0.0206} = 0.205 \]

\[ P(“no”) = \frac{0.0206}{0.0053 + 0.0206} = 0.795 \]
Handling Numerical Attributes

- Option 1: **Discretize** numerical attributes before learning classifier.
  - Temp = 37°C → “Hot”
  - Temp = 21°C → “Mild”

- Option 2: Make assumption that numerical attributes have a **normal distribution** given the class.
  - Use training data to estimate parameters of the distribution (e.g., mean and standard deviation)
  - Once the probability distribution is known, it can be used to estimate the conditional probability $P(A_i|C_j)$
Handling Numerical Attributes

- The probability density function for the normal distribution is

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

- It is defined by two parameters:
  - **Sample mean** \( \mu \)
    \[ \mu = \frac{1}{n} \sum_{i=1}^{n} x_i \]
  - **Standard deviation** \( \sigma \)
    \[ \sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2} \]

- Both parameters can be estimated from the training data
Statistics for the Weather Data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>2</td>
<td>3</td>
<td>64, 68, 65, 71,</td>
<td>65, 70, 70, 85,</td>
</tr>
<tr>
<td>Overcast</td>
<td>4</td>
<td>0</td>
<td>69, 70, 72, 80,</td>
<td>70, 75, 90, 91,</td>
</tr>
<tr>
<td>Rainy</td>
<td>3</td>
<td>2</td>
<td>72, ... 85, ...</td>
<td>80, ... 95, ...</td>
</tr>
<tr>
<td>Sunny</td>
<td>2/9</td>
<td>3/5</td>
<td>μ =73  μ =75</td>
<td>μ =79  μ =86</td>
</tr>
<tr>
<td>Overcast</td>
<td>4/9</td>
<td>0/5</td>
<td>σ =6.2 σ =7.9</td>
<td>σ =10.2 σ =9.7</td>
</tr>
<tr>
<td>Rainy</td>
<td>3/9</td>
<td>2/5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example calculation:

\[
f(\text{temp} = 66 \mid \text{yes}) = \frac{1}{\sqrt{2\pi 6.2}} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340
\]
Classifying a New Record

Unseen record

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>66</td>
<td>90</td>
<td>true</td>
<td>?</td>
</tr>
</tbody>
</table>

Likelihood of “yes” = \( \frac{2}{9} \times 0.0340 \times 0.0221 \times \frac{3}{9} \times \frac{9}{14} = 0.000036 \)

Likelihood of “no” = \( \frac{3}{5} \times 0.0291 \times 0.0380 \times \frac{3}{5} \times \frac{5}{14} = 0.000136 \)

\[ P(“yes”) = \frac{0.000036}{0.000036 + 0.000136} = 20.9\% \]

\[ P(“no”) = \frac{0.000136}{0.000036 + 0.000136} = 79.1\% \]

Caveat: Some numeric attributes are not normally distributed and you may thus need to choose a different probability density function or use discretization.
Handling Missing Values

- Missing values may occur in training and in unseen classification records

- **Training**: Record is not included into frequency count for attribute value-class combination

- **Classification**: Attribute will be omitted from calculation
  
  - Example:

    | Outlook | Temp. | Humidity | Windy | Play |
    |---------|-------|----------|-------|------|
    | ?       | Cool  | High     | True  | ?    |

    Likelihood of “yes” = \( \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0238 \)
    
    Likelihood of “no” = \( \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0343 \)
    
    \( P(\text{“yes”}) = \frac{0.0238}{0.0238 + 0.0343} = 41\% \)
    
    \( P(\text{“no”}) = \frac{0.0343}{0.0238 + 0.0343} = 59\% \)
Zero Frequency Problem

• If one of the conditional probabilities is zero, then the entire expression becomes zero
• And it is not unlikely that an exactly same data point has not yet been observed
• Probability estimation:

\[
\text{Original: } P(A_i|C) = \frac{N_{ic}}{N_c}
\]

\[
\text{Laplace: } P(A_i|C) = \frac{N_{ic} + 1}{N_c + c}
\]

\(c\): number of attribute values of A
Naïve Bayes in RapidMiner & Python

```python
model = GaussianNB()
model.fit(features, label)
```
# Anatomy of a Naïve Bayes Model

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Parameter</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook</td>
<td>value=rain</td>
<td>0.392</td>
<td>0.331</td>
</tr>
<tr>
<td>Outlook</td>
<td>value=overcast</td>
<td>0.014</td>
<td></td>
</tr>
<tr>
<td>Outlook</td>
<td>value=sunny</td>
<td>0.581</td>
<td></td>
</tr>
<tr>
<td>Outlook</td>
<td>value=unknown</td>
<td>0.014</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>mean</td>
<td>74.600</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>standard deviation</td>
<td>7.893</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>mean</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>standard deviation</td>
<td>9.618</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=true</td>
<td>0.589</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=false</td>
<td>0.397</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=unknown</td>
<td>0.014</td>
<td></td>
</tr>
</tbody>
</table>
Using Conditional Probabilities for Naïve Bayes

<table>
<thead>
<tr>
<th>Row No.</th>
<th>Play</th>
<th>confidence(no)</th>
<th>confidence(yes)</th>
<th>prediction(Play)</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>0.711</td>
<td>0.289</td>
<td>no</td>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>0.058</td>
<td>0.942</td>
<td>yes</td>
<td>overcast</td>
<td>80</td>
<td>90</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>0.014</td>
<td>0.986</td>
<td>yes</td>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>0.412</td>
<td>0.588</td>
<td>yes</td>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>0.460</td>
<td>0.540</td>
<td>yes</td>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>no</td>
<td>0.336</td>
<td>0.664</td>
<td>yes</td>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>0.010</td>
<td>0.990</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>no</td>
<td>0.596</td>
<td>0.404</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>0.248</td>
<td>0.752</td>
<td>yes</td>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
</tr>
<tr>
<td>10</td>
<td>no</td>
<td>0.407</td>
<td>0.593</td>
<td>yes</td>
<td>sunny</td>
<td>75</td>
<td>80</td>
<td>false</td>
</tr>
<tr>
<td>11</td>
<td>yes</td>
<td>0.496</td>
<td>0.504</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>yes</td>
<td>0.038</td>
<td>0.962</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>no</td>
<td>0.027</td>
<td>0.973</td>
<td>yes</td>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>true</td>
</tr>
<tr>
<td>14</td>
<td>yes</td>
<td>0.453</td>
<td>0.547</td>
<td>yes</td>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
</tr>
</tbody>
</table>

- **Classifier is quite sure**
- **Classifier is not sure**
Decision Boundary of Naive Bayes Classifier

- Usually larger coherent areas
- Soft margins with uncertain regions
- Arbitrary (often curved) shapes
Naïve Bayes (Summary)

- Robust to isolated noise points
  - they have a small impact on the probabilities

- Handle missing values by ignoring the instance during probability estimate calculations

- Robust to irrelevant attributes

- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)
Why Naïve Bayes?

• Recap:
  – we assume that all the attributes are independent

• This does not hold for many real world datasets
  – e.g., persons: sex, weight, height
  – e.g., cars: weight, fuel consumption
  – e.g., countries: population, area, GDP
  – e.g., food: ingredients
  – e.g., text: word occurrences (“Donald”, “Trump”, “Duck”)
  – ...
Naïve Bayes Discussion

• Naïve Bayes works surprisingly well
  – even if independence assumption is clearly violated
  – Classification doesn’t require accurate probability estimates as long as maximum probability is assigned to correct class

• Too many redundant attributes will cause problems
  – Solution: Select attribute subset as Naïve Bayes often works as well or better with just a fraction of all attributes

• Technical advantages:
  – Learning Naïve Bayes classifiers is computationally cheap (probabilities are estimated in one pass over the training data)
  – Storing the probabilities does not require a lot of memory
Redundant Variables

• Consider two variables which are perfectly correlated
  – i.e., one is redundant
  – e.g.: a measurement in different units
• Violate independence assumption in Naive Bayes
  – Can, at large scale, skew the result
  – Consider, e.g., a price attribute in 20 currencies
    → price variable gets 20 times more influence
• May also skew the distance measures in k-NN
  – But the effect is not as drastic
  – Depends on the distance measure used
Irrelevant Variables

• Consider a random variable $x$, and two classes $A$ and $B$
  – For Naive Bayes: $p(x=v|A) = p(x=v|B)$ for any value $v$
  – Since it is random, it does not depend on the class variable
  – The overall result does not change

• For kNN:
Comparison kNN and Naïve Bayes

• Computation
  – Naïve Bayes is often faster

• Naïve Bayes uses *all* data points
  – Naive Bayes is less sensitive to label noise
  – k-NN is less sensitive to outliers

• *Redundant* attributes
  – are less problematic for kNN

• *Irrelevant* attributes
  – are less problematic for Naïve Bayes
  – attribute values equally distributed across classes
    → same factor for each class

• In both cases
  – attribute pre-selection makes sense (see Data Mining II)
Lazy vs. Eager Learning

• k-NN, and Naïve Bayes are all “lazy” methods
• They do not build an explicit model!
  – “learning” is only performed on demand for unseen records
• Nearest Centroid is a simple “eager” method

<table>
<thead>
<tr>
<th>Td</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>120K</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>95K</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>Yes</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Td</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</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>60K</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>
Lazy vs. Eager Learning

- We have seen three of the most common techniques for lazy learning
  - k nearest neighbors
  - Naïve Bayes
- ...and a very simple technique for eager learning
  - Nearest Centroids

- We will see more eager learning in the next lectures
  - where explicit models are built
  - e.g., decision trees
  - e.g., rule sets
Model Evaluation

• This week: metrics
  • how to measure performance?
  • here: quality of predictions, not: training time

• Next week: evaluation methods
  • how to obtain meaningful and reliable estimates?
Metrics for Performance Evaluation

- Looking at correctly/incorrectly classified instances
- Two class problem (positive/negative class):
  - true positives, false positives, true negatives, false negatives

Confusion Matrix:

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>TP</td>
</tr>
<tr>
<td>Class=No</td>
<td>FP</td>
</tr>
</tbody>
</table>
Metrics for Performance Evaluation

Most frequently used metrics:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

Error Rate = 1 – Accuracy
What is a Good Accuracy?

• i.e., when are you done?
  – at 75% accuracy?
  – at 90% accuracy?
  – at 95% accuracy?

• Depends on difficulty of the problem!
• Baseline: naive guessing
  – always predict majority class

• Compare
  – Predicting coin tosses with accuracy of 50%
  – Predicting dice roll with accuracy of 50%
  – Predicting lottery numbers (6 out of 49) with accuracy of 50%
Limitation of Accuracy: Unbalanced Data

• Sometimes, classes have very unequal frequency
  – Fraud detection: 98% transactions OK, 2% fraud
  – eCommerce: 99% don’t buy, 1% buy
  – Intruder detection: 99.99% of the users are no intruders
  – Security: >99.99% of Americans are not terrorists

• 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 example
Precision and Recall

How many examples that are classified positive are actually positive?

\[ p = \frac{TP}{TP + FP} \]

Which fraction of all positive examples is classified correctly?

\[ r = \frac{TP}{TP + FN} \]

All positives

Ignored majority

false negatives

true negatives

true positives

false positives

Classified as positives
Precision and Recall Example

This confusion matrix gives us:

- **precision** $p = 100\%$ and
- **recall** $r = 1\%$

because we only classified one positive example correctly and no negative examples wrongly.

We want a measure that combines precision and recall.
F₁-Measure

- It is hard to compare two classifiers using two measures
- F₁-Score combines precision and recall into one measure
  - by using the harmonic mean

\[
F_1 = \frac{2 \frac{1}{p} + \frac{1}{r}}{\frac{1}{p} + \frac{1}{r}} = \frac{2 pr}{p+r}
\]

- The harmonic mean of two numbers tends to be closer to the smaller of the two
- For F₁-value to be large, both p and r must be large
F$_1$-Measure Graph

Low threshold: Low precision, high recall
Restrictive threshold: High precision, low recall
## Alternative for Unbalanced Data: Cost Matrix

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

$C(i|j)$: Cost of misclassifying class $j$ example as class $i$
Computing Cost of Classification

### Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(i</td>
</tr>
<tr>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

### Model M₁

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>162</td>
</tr>
<tr>
<td>-</td>
<td>160</td>
</tr>
</tbody>
</table>

Accuracy = 67%
Cost = 3960

### Model M₂

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>155</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

Accuracy = 92%
Cost = 4505
ROC Curves

• Some classification algorithms provide confidence scores
  - how sure the algorithms is with its prediction
  - e.g., Naive Bayes: the probability
  - e.g., Decision Trees: the purity of the respective leaf node

• Drawing a ROC Curve
  - Sort classifications according to confidence scores
    (e.g.: predicted probabilities in Naive Bayes)
  - Evaluate
    • correct prediction: draw one step up
    • incorrect prediction: draw one step to the right
ROC Curves

- Drawing ROC Curves in RapidMiner & Python

```python
fpr, tpr, thresholds = roc_curve(actual, predictions)
plt.plot(fpr, tpr)
```
Example ROC Curve of Naive Bayes
Interpreting ROC Curves

• Best possible result:
  – all correct predictions have higher confidence than all incorrect ones

• The steeper, the better
  – random guessing results in the diagonal
  – so a decent algorithm should result in a curve significantly above the diagonal

• Comparing algorithms:
  – Curve A above curve B means algorithm A better than algorithm B

• Frequently used criterion
  – area under curve (aka ROC AUC)
  – normalized to 1
Questions?