Image Blob Detection: A Machine Learning Approach

Andrew Colello
Undergraduate Thesis 2016
Union College Department of Computer Science
Faculty Advisor: Valerie Barr
Background: Golf Ball Problem

- Finding bright white ball on a normal day is a hassle.
- Lost golf balls account for $60 million each year.
- Smartphones contain software and hardware that can solve this problem.

Diagram:
- Photo of landing location from phone camera
- System Process
- Ball location info
Background: Alternative Solutions

- **GPS tracking**
  - The RadarGolf System
- **Special coating**
  - The Ballfinder Scout
- **Fixed cameras**
  - SwingShot Golf Video Cameras
Background: Related Research

Self-Driving Cars
- Image processed based on pixel values
- Output of first process is input for classification learning model

Blob Detection
- Picture analyzed for local maxima
  - Several types of averages taken
Research Evolution

• Initial: R and ImageJ
  – Issues with R: ease of use, portability, ecosystem
  – Issues with ImageJ: API limitations

• Making the switch to Python
  • Automatic training set generation
  • Image: automatically drawing circles with PIL

• Machine Learning Process
  – Neural networks
  – Scikit-learn

• Blob detection refinement
  – Tweaking settings for coherent blobs
  – Experimentation with different blob detection algorithms
System Process

- **Data Mining**
  - Flickr, ImageJ
  - Python: Scrapy, PIL

- **Blob Detection**
  - Python: Scikit-Image, Matplotlib

- **Blob Classification**
  - Java: weka
Data Mining

- Scrapy
  - Dataflow:
    - Items
    - Spiders
    - Pipelines
  - Spider:
    - FlickrAPI
    - PIL
  - Pipeline:
    - ARFF format
Blob Detection I

- Goal: Find 'blobs' of like pixels based on:
  - Contiguiousness
  - Contrast
  - Statistics
- Input: Array of pixels
- Output: several (y, x, radius) coordinates
Blob Detection II

- **Types of Blob Detection**
  - Laplacian of Gaussian
    - `blob_log(image_gray, min_sigma=15, max_sigma=50, num_sigma=10, threshold=.1, overlap=0.8)`
  - Determinant of Gaussian
  - Determinant of Hessian

- **Tools used:**
  - Python
    - Numpy
    - Scikit-Image
    - Matplotlib
Data I

- Training data
  - ImageJ

- Flickr data
  - Query
  - Sort conditions
Data II

• Attributes
  – Class
  – X-center
  – Y-center
  – Mean-px
  – Median-px
  – Mode-px
  – Radius
  – Radius-height-pct
  – Radius-width-pct
Classification I

• Broad definition: Use statistics to find patterns in data
  – Classification, Association, Clustering
  – Supervised vs Unsupervised

• Classification: Put various instances into distinct types, or 'classes'
  – Types of classification algorithms
    • Tree based
    • Naive Bayes
    • Neural Network
Classification II

• Weka
  – Inputs
  – Outputs

• Types of algorithms used
  – Cross validation(10x)
  – Top performer: Random Forest
  – Bottom performer: Naive Bayes
Results I

- Zero-R
- One-R
- JRip
- J48
- IBk
- Naive Bayes
- Random Forest

Percent Correct

Bars for each algorithm showing their performance.
## Results II

### Random Forest

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly Classified Instances</td>
<td>138</td>
<td>90.1961%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>15</td>
<td>9.8039%</td>
</tr>
<tr>
<td>Kappa statistic</td>
<td>0.7807</td>
<td></td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.1349</td>
<td></td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.2527</td>
<td></td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>29.4901%</td>
<td></td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>52.8691%</td>
<td></td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>Ignored Class Unknown Instances</td>
<td>61</td>
<td></td>
</tr>
</tbody>
</table>

### Naive Bayes

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly Classified Instances</td>
<td>131</td>
<td>85.6209%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>22</td>
<td>14.3791%</td>
</tr>
<tr>
<td>Kappa statistic</td>
<td>0.6742</td>
<td></td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.1431</td>
<td></td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.344</td>
<td></td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>31.281%</td>
<td></td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>71.9723%</td>
<td></td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>Ignored Class Unknown Instances</td>
<td>61</td>
<td></td>
</tr>
</tbody>
</table>
Next Steps

- Expand training set
- Create cloud-based I/O platform
- Write mobile app that interfaces with cloud
Questions?