Assignment 4: Object Classification
<table>
<thead>
<tr>
<th>airplane</th>
<th>butterfly</th>
<th>camera</th>
<th>helicopter</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Airplane" /></td>
<td><img src="image2.jpg" alt="Butterfly" /></td>
<td><img src="image3.jpg" alt="Camera" /></td>
<td><img src="image4.jpg" alt="Helicopter" /></td>
</tr>
</tbody>
</table>

Also: lotus, panda, pizza, pyramid, snoopy, yin-yang
Your Task:

What am I?

10 categories, 10x10 train images, 10x20 test images
Results

Overall success rate: 0.50

Individual success rates:
- airplane: 0.85
- butterfly: 0.15
- camera: 0.75
- helicopter: 0.40
- lotus: 0.60
- panda: 0.10
- pizza: 0.55
- pyramid: 0.60
- snoopy: 0.50
- yin_yang: 0.50
How?
Train

Detect Features
Random
Harris
Sift
Etc.

Compute Descriptors
Window
Sift
Etc.

Should be familiar by now...
Train

Descriptors

kmeans
Train

Descriptors

Codewords

airplane

butterfly

Codeword 1
Codeword 2
Codeword 3
Codeword 4
**API Tips**

\[ \text{IDX, C} = \text{kmeans}(X, k) \]

\[ \text{[...]} = \text{kmeans}(..., \text{param1, val1, param2, val2, ...}) \]

<table>
<thead>
<tr>
<th>'emptyaction'</th>
<th>Action to take if a cluster loses all its member observations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>'error'</td>
<td>Treat an empty cluster as an error (default).</td>
</tr>
<tr>
<td>'drop'</td>
<td>Remove any clusters that become empty. kmeans sets the corresponding return values in C and D to NaN.</td>
</tr>
<tr>
<td>'singleton'</td>
<td>Create a new cluster consisting of the one point furthest from its centroid.</td>
</tr>
</tbody>
</table>
Classify

- Detect Features
- Compute Descriptors
- Naive Bayes

Codewords

Histograms

Pizza!!!
Naive Bayes

\[ p(C_i | p_1, p_2, \ldots) = \frac{p(C_i)}{p(p_1, p_2, \ldots)} \]  

(1)

\[ \propto p(p_1, p_2, \ldots | C_i) \]  

(2)

\[ = \prod_j p(p_j | C_i) \]  

(3)

(1) - Bayes formula
(2) - Assume equal class priors, don't care about normalization.
(3) - Assume independent descriptors
Naive Bayes

\[ p(C_i | p_1, p_2, \ldots) \propto \prod_j p(p_j | C_i) \]

Estimated class:

\[ \arg \max_i \prod_j p(p_j | C_i) \]

(Using the fact that

\[ p(p_j | C_i) \propto \text{count}(p_j \in \text{training}(C_i)) \]

2 more practical modifications...
Naive Bayes

Prevent zero counts

\[ p(p_j | C_i) \propto \text{count}(p_j \in \text{training}(C_i)) \]

Prevent overflow

\[ p(p_j | C_i) \propto 1 + \text{count}(p_j \in \text{training}(C_i)) \]

Estimated class:

\[ \arg\max_i \sum_j \log(1 + \text{count}(p_j \in \text{training}(C_i))) \]