Where to go from our basic building block?

Features

- Image formation
- Gradients, edges, corners
- Descriptors

- Recognition
  - Scenes, places, objects,
  - ~ 5 weeks (inc. CNNs)

- Reconstruction
  - Geometric understanding
  - Last topic in class
Panorama stitching / instance recognition

Needs more geometric understanding…

…but we’ll see it later on.
Recognition

Often needs machine learning for compact descriptions of the visual world.

Scene recognition - City/forest/factory/…

Find pedestrians
Our approach

• We will look at ML as a tool. We will not detail the underpinnings of each learning method.

• Please take a machine learning course if you want to know more!
Machine Learning

• Learn from and make predictions on data.

• Arguably the greatest export from computing to other scientific fields.

• Statisticians might disagree with CompScis on the true origins...
ML for Computer Vision

- Face Recognition
- Object Classification
- Scene Segmentation
Data, data, data!

Peter Norvig
“The Unreasonable Effectiveness of Data”
(IEEE Intelligent Systems, 2009)

“... invariably, simple models and a lot of data trump more elaborate models based on less data”

[Norvig is Director of Research @ Google, and a Brown APMA alum!]
ImageNet

- Images for each category of WordNet
- 1000 classes
- 1.2mil images
- 100k test
- Top 5 error
<table>
<thead>
<tr>
<th>mite</th>
<th>container ship</th>
<th>motor scooter</th>
<th>leopard</th>
</tr>
</thead>
<tbody>
<tr>
<td>mite</td>
<td>container ship</td>
<td>motor scooter</td>
<td>leopard</td>
</tr>
<tr>
<td>black widow</td>
<td>lifeboat</td>
<td>go-kart</td>
<td>jaguar</td>
</tr>
<tr>
<td>cockroach</td>
<td>amphibian</td>
<td>moped</td>
<td>cheetah</td>
</tr>
<tr>
<td>tick</td>
<td>fireboat</td>
<td>bumper car</td>
<td>snow leopard</td>
</tr>
<tr>
<td>starfish</td>
<td>drilling platform</td>
<td>golfcart</td>
<td>Egyptian cat</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>grille</th>
<th>mushroom</th>
<th>cherry</th>
<th>Madagascar cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>convertible</td>
<td>agaric</td>
<td>dalmatian</td>
<td>squirrel monkey</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>grape</td>
<td>spider monkey</td>
</tr>
<tr>
<td>pickup</td>
<td>jelly fungus</td>
<td>elderberry</td>
<td>titi</td>
</tr>
<tr>
<td>beach wagon</td>
<td>gill fungus</td>
<td>afffordshire bullterrier</td>
<td>indri</td>
</tr>
<tr>
<td>fire engine</td>
<td>dead-man's-fingers</td>
<td>currant</td>
<td>howler monkey</td>
</tr>
</tbody>
</table>
ImageNet Competition

- Krizhevsky, 2012
- Google, Microsoft 2015
  - Beat the best human score in the ImageNet challenge.
<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Dimensionality reduction</td>
</tr>
<tr>
<td>Discrete</td>
<td>Clustering</td>
</tr>
<tr>
<td>Classification or categorization</td>
<td>Regression</td>
</tr>
</tbody>
</table>
## Machine Learning Problems

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>classification or categorization</td>
<td>clustering</td>
</tr>
<tr>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td>dimensionality reduction</td>
</tr>
</tbody>
</table>
Dimensionality Reduction

- **PCA, ICA, LLE, Isomap**
  - Principal component analysis
    - Creates a basis where the axes represent the dimensions of variance, from high to low.
    - Finds correlations in data dimensions to produce *best possible* lower-dimensional representation based on linear projections.
    - Subtract mean of data, compute eigendecomposition.
    - Eigenvectors are directions of variance; eigenvalues are magnitudes of variance.
PCA

A

B

1st principal component
Orthogonal 2nd principal component

Minor axis
Major axis

(Figure adapted from C. Beckmann, Oxford FMRIB)
Eigenfaces

The ATT face database (formerly the ORL database), 10 pictures of 40 subjects each
Eigenfaces

Mean face (not Jack)

Basis of variance (eigenvectors)

## Machine Learning Problems

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>classification or categorization</td>
<td>clustering</td>
</tr>
<tr>
<td>regression</td>
<td>dimensionality reduction</td>
</tr>
</tbody>
</table>
Clustered by similar populations

http://fakeisthenewreal.org/reform/
The United States redrawn as Fifty States with Equal Population.
Clustering: color image segmentation

Goal: Break up the image into meaningful or perceptually similar regions
Segmentation for feature support or efficiency

[Hoiem et al. 2005, Mori 2005]

[Shi and Malik 2001]

[Felzenszwalb and Huttenlocher 2004]

Derek Hoiem
Types of segmentations

Oversegmentation

Undersegmentation

Hierarchical Segmentations
Clustering

Group together similar ‘points’ and represent them with a single token.

Key Challenges:
1) What makes two points/images/patches similar?
2) How do we compute an overall grouping from pairwise similarities?
Why do we cluster?

• **Summarizing data**
  – Look at large amounts of data
  – Patch-based compression or denoising
  – Represent a large continuous vector with the cluster number

• **Counting**
  – Histograms of texture, color, SIFT vectors

• **Segmentation**
  – Separate the image into different regions

• **Prediction**
  – Images in the same cluster may have the same labels
How do we cluster?

• K-means
  – Iteratively re-assign points to the nearest cluster center

• Agglomerative clustering
  – Start with each point as its own cluster and iteratively merge the closest clusters

• Mean-shift clustering
  – Estimate modes of pdf

• Spectral clustering
  – Split the nodes in a graph based on assigned links with similarity weights
K-means algorithm

1. Randomly select K centers

2. Assign each point to nearest center

3. Compute new center (mean) for each cluster

K-means algorithm

1. Randomly select $K$ centers

2. Assign each point to nearest center

3. Compute new center (mean) for each cluster

K-means

1. Initialize cluster centers: $c^0$; $t=0$

2. Assign each point to the closest center

$$\delta^t = \arg\min_\delta \frac{1}{N} \sum_j \sum_i \delta_{ij} (c^t_{i-1} - x_j)^2$$

3. Update cluster centers as the mean of the points

$$c^t = \arg\min_c \frac{1}{N} \sum_j \sum_i \delta^t_{ij} (c_i - x_j)^2$$

4. Repeat 2-3 until no points are re-assigned ($t=t+1$)
Think-Pair-Share

• What is good about k-means?
• What is bad about k-means?
• Where could you apply k-means?
K-means: design choices

• Initialization
  – Randomly select K points as initial cluster center
  – Or greedily choose K points to minimize residual

• Distance measures
  – Traditionally Euclidean, could be others

• Optimization
  – Will converge to a local minimum
  – May want to perform multiple restarts
K-means clustering using intensity or color
How to choose the number of clusters?

• Validation set
  – Try different numbers of clusters and look at performance
  • When building dictionaries (discussed later), more clusters typically work better.
How to initialize the clusters?

k-means++ initialization

– Make the initial cluster centers span the space

1. Choose one center uniformly at random from all data points.
2. For each point \( x \), compute the distance \( D(x) \) between \( x \) and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point \( x \) is chosen with probability proportional to \( D(x)^2 \).
4. Repeat Steps 2 and 3 until \( k \) centers have been chosen.
5. Proceed using standard k-means clustering.
K-Means pros and cons

• **Pros**
  • Finds cluster centers that minimize conditional variance (good representation of data)
  • Simple and fast*
  • Easy to implement

• **Cons**
  • Need to choose K
  • Sensitive to outliers
  • Prone to local minima
  • All clusters have the same parameters (e.g., distance measure is non-adaptive)
  • *Can be slow: each iteration is $O(KNd)$ for $N$ d-dimensional points

• **Usage**
  • Cluster features to build visual dictionaries
Building Visual Dictionaries

1. Sample features from a database
   - E.g., 128 dimensional SIFT vectors

2. Cluster to build dictionary
   - Cluster centers are the dictionary words

3. To match new features, assign to the nearest cluster to save rebuilding dictionary
Examples of learned codewords

Most likely codewords for 4 learned “topics”
EM with multinomial (problem 3) to get topics

http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic05b.pdf  Sivic et al. ICCV 2005
Agglomerative clustering

1. Say “Every point is its own cluster”
Agglomerative clustering

1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

How to define cluster similarity?
- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids

How many clusters?
- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges
Conclusions: Agglomerative Clustering

Good

• Simple to implement, widespread application
• Clusters have adaptive shapes
• Provides a hierarchy of clusters

Bad

• May have imbalanced clusters
• Still have to choose number of clusters or threshold
• Need to use an “ultrametric” to get a meaningful hierarchy
Mean shift segmentation


• Versatile technique for clustering-based segmentation
Mean shift algorithm

Try to find *modes* of a non-parametric density.
Attraction basin

- **Attraction basin**: the region for which all trajectories lead to the same mode
- **Cluster**: all data points in the attraction basin of a mode
Attraction basin
Mean shift

Region of interest
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Region of interest

Center of mass

Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift
Mean shift

Region of interest

Center of mass

Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Region of interest
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Region of interest
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Region of interest

Center of mass

Slide by Y. Ukrainitz & B. Sarel
Kernel density estimation

Kernel density estimation function

\[ \hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{x - x_i}{h} \right) \]

Gaussian kernel

\[ K \left( \frac{x - x_i}{h} \right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x - x_i)^2}{2h^2}}. \]
Computing the Mean Shift

Simple Mean Shift procedure:
• Compute mean shift vector \( \mathbf{m}(\mathbf{x}) \)
• Iteratively translate the kernel window by \( \mathbf{m}(\mathbf{x}) \) until convergence.

\[
\mathbf{m}(\mathbf{x}) = \left[ \frac{\sum_{i=1}^{n} \mathbf{x}_i g \left( \frac{\| \mathbf{x} - \mathbf{x}_i \|^2}{h} \right)}{\sum_{i=1}^{n} g \left( \frac{\| \mathbf{x} - \mathbf{x}_i \|^2}{h} \right)} \right] - \mathbf{x}
\]
Mean shift clustering

The mean shift algorithm seeks *modes* of the given set of points

1. Choose kernel and bandwidth
2. For each point:
   a) Center a window on that point
   b) Compute the mean of the data in the search window
   c) Center the search window at the new mean location
   d) Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster
Segmentation by Mean Shift

• Compute features for each pixel (color, gradients, texture, etc.).
• Set kernel size for features $K_f$ and position $K_s$.
• Initialize windows at individual pixel locations.
• Perform mean shift for each window until convergence.
• Merge windows that are within width of $K_f$ and $K_s$. 
Mean shift segmentation results

Comaniciu and Meer 2002
Mean shift pros and cons

• Pros
  – Good general-practice segmentation
  – Flexible in number and shape of regions
  – Robust to outliers

• Cons
  – Have to choose kernel size in advance
  – Not suitable for high-dimensional features

• When to use it
  – Oversegmentation
  – Multiple segmentations
  – Tracking, clustering, filtering applications
Spectral clustering

Group points based on graph structure & edge costs. Captures “neighborhood-ness” or local smoothness.

Image: Hassan et al.
Spectral clustering

Group points based on links in a graph
Cuts in a graph

Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

\[ N\text{cut}(A, B) = \frac{\text{cut}(A, B)}{\text{volume}(A)} + \frac{\text{cut}(A, B)}{\text{volume}(B)} \]

- volume(A) = sum of costs of all edges that touch A

Source: Seitz
Normalized cuts for segmentation
Segmentation as a result

GrabCut, Rother et al. 2004
Which algorithm to use?

• Quantization/Summarization: K-means
  – Aims to preserve variance of original data
  – Can easily assign new point to a cluster

Summary of 20,000 photos of Rome using “greedy k-means”
http://grail.cs.washington.edu/projects/canonview/
Which algorithm to use?

• Image segmentation: agglomerative clustering
  – More flexible with distance measures (e.g., can be based on boundary prediction)
  – Adapts better to specific data
  – Hierarchy can be useful

http://www.cs.berkeley.edu/~arbelaez/UCM.html
Things to remember

• K-means useful for summarization, building dictionaries of patches, general clustering

• Agglomerative clustering useful for segmentation, general clustering

• Spectral clustering useful for determining relevance, summarization, segmentation