CS 231A CA Session: Problem Set 4 Review

Kevin Chen

May 13, 2016

PS4 Outline

- Problem 1: Viewpoint estimation
- Problem 2: Segmentation
  - Meanshift segmentation
  - Normalized cut

Problem 1: Viewpoint Estimation

Viewpoint estimation using Bag of Words

- Goal: Perform image classification
- Steps:
  - Build dictionary of codewords
  - Build features based on codewords
  - Train an SVM classifier on the features and labels

Viewpoint estimation using Bag of Words

- Goal: Perform image classification
- Steps:
  - **Build dictionary of codewords**
  - Build features based on codewords
  - Train an SVM classifier on the features and labels

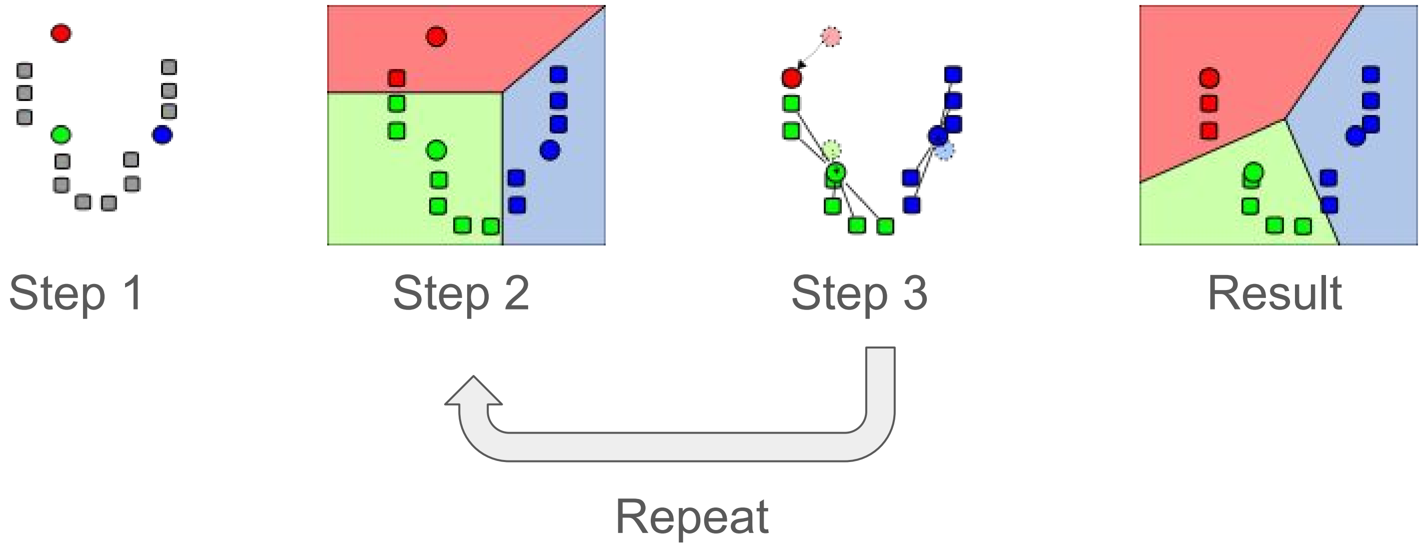
Build dictionary of codewords

- Goal: compute “words” for our bag of words model
  - Will be useful for computing a feature vector for an image
- Basic idea: perform k-means on dense SIFT features
- Cluster centers form the “words” in the bag of words model
- Algorithm
  - Subsample images in train set
  - Compute dense SIFT features
  - Find cluster centers by running k-means

K-means clustering

- Fixed number of clusters (k)
- Algorithm
  - Given an initial set of **k** means
  - Repeat until convergence
    - Assign each point to a *i*th cluster if it is closest to the *i*th mean (euclidean distance)
    - Recompute means
- Use VLFeat

k-means clustering



Viewpoint estimation using Bag of Words

- Goal: Perform image classification
- Steps:
    - Build dictionary of codewords
    - **Build features based on codewords**
    - Train an SVM classifier on the features and labels

Build features based on codewords

- Goal: Extract a feature from each image
- Algorithm
  - Extract dense SIFT features for every image in train set
  - For each image, build pyramid of dxd sub-images
  - For each pyramid depth
    - For every dense SIFT feature, find nearest codeword according to EUCLIDEAN distance
    - Build histogram of codewords
  - Concatenate sub-image histograms to build the final feature vector for the image

Viewpoint estimation using Bag of Words

- Goal: Perform image classification
- Steps:
  - Build dictionary of codewords
  - Build features based on codewords
  - **Train an SVM classifier on the features and labels**

Problem 2: Segmentation

Segmentation

Meanshift

Meanshift segmentation

- Compute features for each pixel, e.g. (u, v, R, G, B)
- Keep track of visited pixels (initialize all to unvisited)
- Keep track of cluster centers (initially there are none)
- Repeat until all pixels have been visited
    - Pick a random, unvisited pixel and get its corresponding feature vector
    - Perform meanshift procedure to get cluster center
        - Make sure to update visited pixels
    - Update list of cluster centers
        - Check if the newly computed cluster center is sufficiently close to any previous cluster centers
            - If so, ignore newly computed cluster center (say it's part of a previous cluster)
            - If not, add new cluster to cluster list
- Assign pixels to nearest cluster (in feature space)

Meanshift procedure

- Repeat until convergence
  - Find all features within bandwidth distance from the current feature vector
  - Compute mean of these features
    - All features with bandwidth circle (at any iteration) should have their corresponding pixels marked as visited
  - Update “current feature vector” as the newly computed mean
- Return the final mean as a new cluster center

Normalized Cut

Motivation

- Graph can be constructed with nodes as pixels and edges between pixels
    - Weights are associated with each edge
- Typically want to partition graph into two parts
    - Find partition corresponding to minimum cut cost
- Often ends up choosing cuts that partition small sets of nodes
- Use norm cut instead
    - Norm cut - cost is computed as a fraction of weights of edges connecting to the partition
- See paper for details

Basic ideas

- Vector **x** defines a partition
  - $\mathbf{x}_i$ = 1 if node *i* is in subset A
  - $\mathbf{x}_i$ = -1 if node *i* is in subset B
- We want to solve the following problem:

$$\min_x \operatorname{ncut}(x, W, D)$$

- Solve for partition by trying to solve an equivalent problem

$$\min_x \operatorname{ncut}(x, W, D) = \min_y \frac{y^T(W-D)y}{y^T D y}$$

$$\text{subject to} \quad y_i \in \{1, -b\},\ y^T D 1 = 0$$

- **y** is a reformulation of the partition indicator vector **x**
  - We can retrieve A, B from **y**

We ask you to implement...

- The computation of norm cut cost
- A recursive algorithm for partitioning the graph

We ask you to implement...

- **The computation of norm cut cost**
- A recursive algorithm for partitioning the graph

Computation of norm cut

- Given a partition value **ϵ** and vector **v**, we can compute **x**
  - If $\mathbf{v}_i > \boldsymbol{\epsilon}$, then $\mathbf{x}_i = 1$
  - If $\mathbf{v}_i <= \boldsymbol{\epsilon}$, then $\mathbf{x}_i = -1$
- Use definitions to compute **D**, **k**, **b**, and **y**
- Cost can then be computed by

$$\text{ncut} = \frac{y^T(D-W)y}{y^TDy}$$

Definitions

- Given a weight matrix **W** in which **W**(*i*, *j*) corresponds to edge weight between node *i* and node *j*
- Compute diagonal matrix **D** in which **D**(*i, i*) corresponds to sum of weights connecting to node *i*
- Vector **x** defines a partition
    - $\mathbf{x}_i$ = 1 if node *i* is in subset A
    - $\mathbf{x}_i$ = -1 if node *i* is in subset B
- Others
    - Define **k** to be sum of edge weights which connect to nodes in A
    - Define **b** to be ratio of edge weights in A to edge weights in B

We ask you to implement...

- The computation of norm cut cost
- **A recursive algorithm for partitioning the graph**

Recursive algorithm for partitioning the graph

- Compute **D**
- Approximately solve for **y** in the problem shown before

$$\min_x \text{ncut}(x, W, D) = \min_y \frac{y^T(W - D)y}{y^T D y}$$

$$\text{subject to} \quad y_i \in \{1, -b\}, \; y^T D 1 = 0$$

  - To do this, solve generalized eigensystem

$$(D - W)y = \lambda D y$$

    by finding the eigenvector **v** associated with the second smallest eigenvalue of the generalized eigensystem

- This does not satisfy first constraint, so find best partition point using *fminsearch* on norm cut cost
- Once a partition point is found, we can get A and B
- Recurse on partitions A and B unless end recursion conditions are true

When ending recursion...

- Suppose we want to partition V into A and B
- End recursion conditions
  - Norm cust cost is above threshold
  - Proposed A partition is too small
  - Proposed B partition is too small
- If any of the above are true…
  - Return V as the segmentation

Questions?