6/12
One-shot Recognition

Kevin Tang

Project Review

- Motivated by the paper "Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer" by Lampert et al. at CVPR 2009
  - Animals with Attributes dataset
- Problem:
  - Achieving better one-shot recognition performance
- Potential Applications:
  - Content-based image retrieval
  - Scalable system for object recognition

Potential Solution

- Solution:
  - Train a system in a way such that it "UNDERSTANDS" that it must perform one-shot recognition
  - Related Work:
    - Neighborhood Components Analysis (Goldberger et al.)
    - Deep Belief Nets (Hinton et al.)
    - Semantic Hashing-based Image Retrieval (Torralba et al.)

Potential Solution

- How do we do this?
- Train the system by simulating one-shot recognition
- Given training data, probabilistically divide classes into multiple sets of few training/many testing examples
- Using sets, learn a dimensionality reduction to reduce data to lower dimension
  - Lower dimension = Attributes
  - Learning important attributes

Potential Solution

- Testing phase:
  - Given one example to learn from
  - Project onto lower dimension
  - Use K-nearest neighbors (KNN) for classification
- Similar to:
  - Learning distance metrics
  - Dimensionality reduction
- Idea is that there is an inherent structure to one-shot recognition systems that can be learned

Solution Details

- Pose learning as an optimization problem:
  - Maximize the probability that features of the same class are close in lower dimension (facilitates KNN)
  - Optimize over the multiple sets

$$p_s = \sum_{i \in Te(s)} \frac{\exp(||\mathbf{o}(\mathbf{x}_i) - \mathbf{o}(\mathbf{x}_{Tr(s,C(i))})||^2)}{\sum_{j=1}^{N_C} \exp(||\mathbf{o}(\mathbf{x}_i) - \mathbf{o}(\mathbf{x}_{Tr(s,j)})||^2)},$$

$$\max_{\theta} \sum_{l=1}^{S} p_{s_l}$$

Project Progress

- Implemented methods to generate sets by randomly sampling
- Implemented linear transformation version
  - Using same training/testing classes as Lampert
  - Train from: 150 different sets, 20 different classes, 1/10 train/test
  - Test from: 150 different sets, 10 different classes, 1/10 train/test
  - Mean accuracy: ~17-18%
  - Chance accuracy: 10%
  - Lampert accuracy: ~40%

Project Progress

- Implemented to use both conjugate gradient and stochastic gradient descent
  - Stochastic gradient descent works better
  - Also suits our needs more intuitively
- Implemented Principle Component Analysis (PCA) as base-line
  - PCA works by selecting eigenvectors with greatest variance
  - Basic method of dimensionality reduction
  - Mean accuracy: ~16-17%

Project Progress

- Not doing any better than PCA
- Implemented nonlinear transformation version (logistic function)
  - Similar training parameters as linear version
  - Preliminary results better: ~20%, but not promising enough
- Idea: Randomize sets at every iteration of stochastic gradient descent
  - Potentially better performance because more randomized
  - With randomized sets, number of iterations more important than number of sets

Project Progress

- Awaiting results from set randomization
- Currently running both linear and nonlinear versions with various parameters tweaked

Next Steps

- Target accuracy: ~30%
- Implementation for Animals with Attributes dataset done, as efficiently optimized as possible
- Vary parameters, keep running tests
- Problem: too many parameters to vary
  - Number of sets
  - Number of iterations
  - Number of training/testing examples

Next steps

- Hard to compare with Lampert's accuracy, try different setups to match other one-shot recognition setups
  - Cross-generalization (Bart, Ullman)
  - One-shot recognition (Fei-Fei Li)
- Extend this work into trying to learn the good/bad attributes from Lampert's set
  - Incorporate cost terms to optimization to push O(x) to resemble attributes
- Consult Dr. Sukthankar, Intel Research
  - Incorporating pre-defined semantic attributes (Lampert, Intel-218)