Anthony Poerio (adp59@pitt.edu)

CS1674: Homework 7 - Written

Due: 10/31/2016, 11:59pm

QUESTIONS

1) **Name three examples of recognition problems in computer vision. For each, explain in 1-2 sentences what the task that we're trying to solve in each of these problems is (i.e. – elaborate on what the problem means).**

Three problems are:

1. The Generic Categorization Problem
   a. The generic categorization problem concerns situations like this one. Given an image, determine **if and where** there are examples of some object in within the image.
   b. For example, if we are given images of a city, we may desire to recognize exactly ***where*** there are cars in that city.
   c. This problem concerns **being able to generalize** and infer what an object is, given abstract attributes that our algorithm associates with it.

2. The Instance-Level Recognition Problem
   a. The instance-level recognition problem concerns situations where we are looking to find ***a specific instance of an object***.
   b. For example, we may desire to recognize "John's Car", or "Jane's Dog".
   c. These are **specific objects to identify**, not general ones.

3. Attribute Based Search Problem
   a. For an attribute based search problem—we are given a set of abstract attributes (i.e.—'red dress').
   b. Then we are tasked with **finding images** that contain *concrete representations of the abstract attributes* the user is searching for.
   c. The goal here is to **find specific visual examples, given abstract data.**

2) **Consider generic categorization versus instance recognition. Why might the former be more challenging than the latter? Why might the reverse be true in some cases?**

Generic categorization <u>might be harder</u> because there might be examples of items in our category that look very different from each other. Then, to a model which is not highly trained on a wide variety of data, it might be impossible to correctly identify all variations of the object type we are looking for.

For example, both **pickup trucks** and **convertibles** are types of cars. But these car types look very different, and unless a model has been exposed to both, it would not necessarily know that both objects are related and contained within the abstract type 'car'.

Instance recognition <u>might be harder</u> because the *indicators* that identify the very specific object we are interested in could be *extremely subtle*.

For example, assume that my dog and your dog are both siblings from the same litter—and the goal of our model is to identify <u>*my dog*</u>, specifically. Unless one of our dogs has some marking which is highly identifiable, it might be nearly impossible to algorithmically differentiate between these **two sibling dogs** with a high degree of accuracy.

3) **Briefly describe what we do during recognition at training time, and what we do at test time.**

<u>At training time:</u> We take a large set of images that have been labeled with the objects/attributes we want to recognize. Then, we extract the features from each attribute. And finally, we apply a prediction function on these feature representations and attempt to minimize the errors we make each time.

After each iteration of this process, we compare our prediction with the labeled information, and store data regarding whether our prediction was 'correct' or 'incorrect'. Each time we do this, the goal is to learning from our errors (or lack thereof). Over time, we are able to create a prediction model that becomes increasingly accurate.

<u>At test time:</u> We take an unseen set of images and process them, making new predictions based on 1) their features; and 2) the prediction model that we built during training.

Ideally, because the prediction model has been exposed to so much similar data during training, it will be able to more accurately predict (or **recognize**) features during our test.

4) **How does a K-Nearest neighbor classifier work?**

The K-Nearest neighbor classifier works by following these steps.

First, map all of our feature vectors into a feature space (which we can define through an algorithm of our choosing).

Second, given a new feature vector—identify the k-closest vectors to it in the feature space we have defined.

Third, look at the labels attached to each of those k-closest vectors. Each of these labels can be seen as a 'vote' to **classify** the new feature vector that we are examining.

Finally, ***find the label with the most votes***—this becomes our classifier for the new vector.

5) **Why might a Spatial Pyramid Match make sense for scene recognition but not object recognition?**

The key feature of a Spatial Pyramid Match is that it divides the image into a set of equal sized 'bins', and then computes a BOW histogram for each—so that the algorithm can weigh each equal-sized part of the image equally as it performs recognition.

This is **good for a 'scene'**, where we are interested in the image as a whole—because, in this case, the whole image is divided into equal parts and weighed equally.

However, this is **bad for object recognition** because object recognitions is NOT interested in the image itself as a whole (at least not necessarily).

Instead, in object recognition, we are interested in some very specific part of the image. The problem is this: let's say that some object we are interested in is contained *fully within* one 'bin'. We won't recognize the object because that bin is considered **only as a fractional part** of the entire recognition process.

And therefore, it's probable that **we won't be able to identify the object correctly**—unless we have a close-up image of the object, specifically one that *occupies the entire image*.