CS 477/577 - Introduction to Computer Vision

Assignment Eleven

Due: 11:59pm (*) Wednesday, Nov 27.

(*) There is grace until 9AM Thursday morning, as the instructor will not grade assignment before then. However, once the instructor starts grading assignments, no more assignments will be accepted..

Weight: Approximately 5 points

This assignment can be done in groups of 2-3. You should create one PDF for the entire group, but everyone should hand in a copy of the PDF to help me keep track. Make it clear on the first page of your PDF who your group is, and a brief explanation of how you worked together. If the group consists of both graduate and undergraduate students, make it clear to what degree undergraduates contributed to any graduate parts or extra parts. Group reports are expected to be higher quality than individual ones.

General instructions

You can use any language you like for this assignment, but unless you feel strongly about it, you might consider continuing with Matlab.

You need to create a PDF document that provides needed meta information such as who was in your group, where you substantively deviated from the specification / suggests, conclusions that are not obvious from the images, etc.

Don't forget to hand in your code as well.

Learning goals

- Learn the K-means algorithm as a basic objective function based clustering method
- Understand segmentation as clustering
- Understand the effect of features such as color, location, and texture
- Exercise chicken and egg thinking to find lines (grads, choice B1)
- Learn about the texton representation of texture (grads, choice B2)
- Learn about using a neural network to learn to characterize texture (grads, **challenge** choice B3)

Part A (undergrads and grads)

This part is mostly about clustering and segmenting images using clustering. Some of you might have implemented clustering using K-means, or the slightly more sophisticated method of using EM to fit a GMM, in some other course. You are welcome to reuse parts of such code, provided you wrote it yourself. While there are many implementations of K-means (and EM+GMM) available online (even in Matlab!) it is instructive to implement this kind of clustering at least once. But having done so, you do not need to do it again, unless you want to.

For this part, unlike previous assignments, a detailed write is not needed for A1, A2, and A3 except to tell the instructor if you used EM+GMM instead of K-means. Otherwise, you only need to provide the images asked for with sufficient labeling for the grader to keep track. Detailed captions are not required. While learning to write good reports is very important, we have done a lot of it so far, and this part of the assignment supports a simpler format (mostly images).

1. Implement K-means clustering. You should make K easy to set. At every iteration you must print out the value of the objective function (which should go down). Recall that the objective function is the sum of the squared errors from the mean of the assigned cluster. Use your implementation to segment the following three images based on color taken as a 3D feature vector, using k=5 and k=10 (also on D2L).
   http://kobus.ca/teaching/cs477/data/sunset.tiff.
   http://kobus.ca/teaching/cs477/data/tiger-1.tiff.
   http://kobus.ca/teaching/cs477/data/tiger-2.tiff.

   For visualizing the segmentations, set each segment to its mean color. You could spice it up adding region borders. One way to do so is to make any pixel that is **not** surrounded by pixels from the same cluster a distinctive color, perhaps adding its neighbors in one pass to make the border thicker.

   > *Note for subsequent parts. This is a reasonable way to visualize your segments, even if you did not use to color to create the segments. For example, a segmentation based only on texture could be visualized this way, although there, using only the borders might be better as you can then see the texture within the region.*
   >
   > *Hint for debugging. Monitor the object function value. It must go down monotonically.*

   You will need to implement a stopping condition. To get started, you could use a reasonably large, fixed number, but once you have a sense for what a good stopping criterion might be, you should implement something more principled. The main choices are **1)** checking whether the relative change in the per-point likelihood does not change more than a threshold for some number of iterations; or **2)** waiting until assignments do not change. In either case, you should have a fallback large number of iterations in case something goes wrong. The disadvantage of the first choice is the need for thresholds, and the doubt that things could have improved one iteration after you stopped. The second choice is more expensive to compute and could lead you to drive the iterations longer than useful for your application–it could be better to spend that computation on a different starting point. Let us know what you choose to do here and any thoughts you have on this **($)**.

   Hand in code to generate the segmentations (6 total), and put the original and segmented images into your PDF "report" **($)**.

2. Add $\lambda*X$ and $\lambda*Y$ to your feature vector, where (X,Y) are the coordinates of the pixel scaled into the range (0,1), and λ is constant that you need to play with. Thus, you now have a five dimensional feature vector $(R, G, B, \lambda*X, \lambda*Y)$. If λ is zero, then you should get the same answer as for the previous question. For each image, provide two images that show the effect of increasing λ, which combined with the implicit result from question 1 for $\lambda=0$, tells a visual story **($)**. Assuming your R,G,B values are between 0 and 1, you could start with $\lambda=1$ and $\lambda=10$. You should be able to explain the results to yourself. You may have to push λ higher to have a significant effect.

   If are using EM+GMM, this will not quite work. Perhaps you can figure out how to weight features in this setting? If this is giving you trouble, ask!

3. Construct some texture features from a black and white version of the above three images. To get texture features, use your code from HW8 to find horizontal and vertical edge energy at two blurring sigmas (2 and 4). The responses of these four filters at a given pixel gives a four-dimensional feature vector for each pixel. Consider a window size, W, centered on each image pixel in turn. Compute the mean squared response of those four filters over the pixels of W. This will result in a derived feature vector that captures some notion of texture in the neighborhood, W, of the point centered at W. Explore **(a)** using this feature vector alone; **(b)** together with the full color (RGB); and **(c)** additional with the spatial location feature in the previous question, for some choice for W for the 4 texture features. You do not need to provide images for more than one choice of W, but make it easy to experiment with so you can provide results for a choice of W that you like (I do not have a good suggestion at this point, but I would start with 31). Provide results for the three images for the three different feature vectors just described **($)**. If the three images provided do not behave interestingly enough, try some other ones!

   Don't forget that K-means is sensitive to the range of your data, and so you will want to scale your features accordingly.

   For more interesting results, you may want to apply the scaling trick to the color features (e.g., multiply the color features by some constant to tune the degree that color is used (much like tuning λ above to tune the degree that spatial adjacency is used).

(Required for grad students only, ugrads are eligible for a bonus).

Choose between B1, B2, and B3 below. If you do more than one of them, you will be eligible for modest extra credit. B1 is likeliest most straightforward. B3 is likeliest the most interesting. All these problems are guided exploration, and I do not know in advance what it will take to make things work for B2 and especially B3. Best of luck!

Unlike the previous section, you need to writeup questions in this section.

Part B1 (K-means for lines)

Part of the learning objective for this question is to structure exploration of computational ideas using synthetic data. Generally, if you are trying out new ideas, you should begin with synthetic experiments where you know the answer, and precisely how it relates to your assumptions. As things get complicated, mediocre results from real data are hard to distinguish from bugs. Generally, as we develop complex methods, we go through cycles of development, testing on synthetic data, and then testing on real data.

Explore using K-means to "segment" points into the lines they lie on. (If you know EM you can use EM instead of (or in addition to) K-means). You will have to generate K lines, and the points on them according to an error model. Let's assume that the distances from the line to generated points should follow a Gaussian distribution, which makes homogeneous line fitting a good choice. After you have worked with this basic noise model to your satisfaction, you may want to try adding outliers (e.g., points uniformly distributed over the image) — see things to explore below.

You will realize that generating points from lines requires more assumptions, and thus things are not so trivial. Lines could be anywhere and go off to infinity. Let's generate lines and points from them to be roughly in the box where x and y are both in [-1,1].

I highly encourage you to figure out line generating lines and points for them for yourself, but if you are stuck, or want to see another way to do this, you can have a look at the code provided with the assignment. Make a note in your writeup how you used it (e.g., copied it, translated it, read it for ideas, read it to check (even though there are various ways to it), read it to make sure the instructor got it right, or simply ignored it).

To do K-means, you will need to compute the perpendicular distance from a point to line, as well as fit lines with homogenous least squares. Recall that developing homogeneous least squares was based on a formula for the distance from a point to line.

You are implementing a classic chicken and egg algorithm driven by unknown correspondences (which points come from the same line). These algorithms invariably have two parts within the main iterative loop which we will the "E" step and the "M" step (borrowing from the "EM" algorithm). In the one step ("E") you deal with correspondence, and in the other step ("M") you deal with fitting the model parameters. Be sure you have these two blocks clear in your mind and in the code.

In your implementation, you should make it so that you can experiment with randomly initializing either (a) correspondences, or (b) the lines. One of your choices of initialization will skip the first block on the first iteration.

Questions to answer include:

1) Is it helpful to loop over multiple initializations and choose the best answer (lowest error)?
2) Which initialization strategy ((a) or (b) above) seems better.
3) What is the effect of varying K (e.g., fewer or more than the "truth" which you know since you generated the data)? Comment on the challenges of inferring K. Could be do this by consulting error on validation data? (A good answer is not trivial, but let me know your thoughts).
4) What is the effect of increasing noise?
5) What is the effect of outliers and increasing their number of them. Might it help to have a special "cluster" that simply keeps track of apparent outliers?
6) If we happen to find the true model (the one we generated), will that typically have the lowest error? Always?

You are welcome to state and address additional questions. The instructor would like to see a minimum of four stated and answered questions **($)** as well as some examples of finding lines by clustering **($)**.

Part B2 (Textons)

Create a texton representation of texture from a modest size data base of images. If you have some idea of what kind of images you would like to use (perhaps related to your research, or perhaps the kind of images you like), by all means create/provide your own training and test set, but tell the reader about your

choice. Otherwise, following a long tradition of texture in images work, let's use patterned animals, specifically tigers, as available here (also on D2L):

http://kobus.ca/teaching/cs477/data/tigers_small.tar.

If you find that more images might be helpful, you can use

http://kobus.ca/teaching/cs477/data/tigers.tar.

To create your textons you first have to get filter bank responses, which means you have to think about creating a filter bank, and then applying it. You can use functions provided by others to do so if you like. I suggest looking into the Matlab functions `gabor`() and `imgaborfilt`(). Having got your filter bank representations of your training images, you can use K-means for clustering to get textons. You will have to choose K via some combination of intuition and experimentation, where success might be defined based on the task below. I would start with K=50, but this is just a guess. Note that doing this might be slow on Matlab, so you may need to consider every tenth point or some other strategy to reduce computation.

Having found your textons, the characterization of texture in your test image at a given point is the histogram of textons in a window W around the point. Use this characterization to segment images based on texture with and without other features such as (x,y) and color.

Part B3 (Autoencoding textures)

This problem builds on problem B of the previous assignment 10. It may be a bit of an adventure, as I have no idea how well this will work, or what adjustments will need to be made. But it should be fun!

This problem is a bit like the B2 in spirit, except we want to use the middle layer encoding from an autoencoder as the texture descriptor instead of histogram over textons. You can use the suggested data provided for B2, or a different data set. Once you have an autoencoder to for windows of size W, you need to get those encodings for each pixel based on a window of size W centered the pixel. You can then use the encoding as descriptor in your K-means clustering. Compare your results to excluding texture and to what you got in A3. You can choose whether your autoencoder runs on black and white images, or perhaps try both!

What to Hand In

As PDF document that provides needed meta information such as who was in your group, where you substantively deviated from the specification / suggests, etc., and the images asked for. Please help the grader out, by checking that the order of the images is the order that they were asked for. Also, hand in your code that creates those images.