CSE 6390D 2010 (F)
Prof. J. Elder
Assignment 1

Overview
In this assignment, we will apply our understanding of probabilistic models and supervised learning to the problem of shape representation. We will restrict our attention to planar (two-dimensional) shape, in particular to the boundaries of animal models.

In the visual world, objects are often obscured or occluded by intervening objects, resulting in fragmented boundaries and a loss of shape information. A good visual system must be robust to such problems.

One advantage of generative models is that they can fill in missing data based upon partial observation. In the context of our problem, this means that the missing portion of the boundary can be estimated. This is the problem of *shape completion*.

In particular, we will evaluate our models by occluding a contiguous 10% portion of an object boundary, and then using our models to estimate these missing data. The root-mean-square Euclidean error between the true locations of the missing points and our estimates will be computed. Our goal is to minimize this error.

Dataset
The dataset is drawn from the Hemera database of 150,000 blue-screened photo-objects. From these I have selected 350 animal objects and randomly partitioned them into training and test datasets of 175 objects each. The boundary of each object has been down-sampled to a vector of $D = 128$ points. Each point of a shape is a 2D Euclidean coordinate. We represent this as a complex number $x + iy$. The data and code I provide uses this representation, as it makes the code a little simpler.

Each shape has been normalized to a unit circle using a Procrustes transformation. For the purposes of this assignment it is not necessary to understand this normalization process, but there are two crucial implications:

1. There is a 1:1 correspondence between the 128-element vectors representing each shape, which facilitates analysis.
2. The expected position of a point on a shape is given by the corresponding point on the unit circle: $E\left[\left(x_i, y_i\right)\right] = \left(\cos\theta_i, \sin\theta_i\right)$, where $\theta_i = \frac{2\pi i}{D}$.

You can access the training dataset now from the course website.

Code
I have provided code for 3 models:

Model 1. This is a very simple generative model that assumes shape vectors are drawn from an isotropic multivariate normal distribution. (In other words the covariance matrix is a diagonal matrix with a constant diagonal.) There is a single scalar parameter: the variance.
Functions:

- ShapeModel1ML.m - computes maximum likelihood estimate of the parameter
- ShapeModel1Sample.m - generates and displays random samples from the model
- ShapeModel1Complete.m - estimates missing portion of a given shape

Model 2. In this generative model, shape vectors are assumed to be samples from a general multivariate normal distribution. There is only one parameter, the covariance matrix, but this represents $D(D+1)/2$ degrees of freedom (i.e., scalar unknowns).
Functions:

- ShapeModel2ML.m - computes maximum likelihood estimate of the parameters
- ShapeModel2Sample.m - generates and displays random samples from the model
- ShapeModel2Complete.m - estimates missing portion of a given shape

Model 3. This model is not generative: it simply uses linear interpolation to estimate the missing points.
Functions:

- ShapeModel3Complete.m - estimates missing portion of a given shape

An additional function EvaluateShapeModelsOnCompletion.m is also provided. This function accepts training and test datasets, uses the training dataset to estimate the maximum likelihood parameters, and then evaluates average RMS error on the test dataset. Note that you will have to partition the training dataset I provide you into training and test portions in order to use this function.

This code can now be downloaded from the website.

Tasks

1. First start by downloading the dataset and code. Make sure that you can run the code and that you are not missing any required toolbox functions. First run the ML functions to estimate the parameters. Then try running the Sample functions to visualize the information contained in Models 1 and 2. Do they generate plausible shapes? Finally, try running the three Completion functions. Do they do a good job in filling in the missing data?
2. Now go through the code in detail and make sure you understand each step. It is often helpful to use the debugger to step through line-by-line, examining data structures along the way.
3. Note that if you train these algorithms using all 175 training shapes, and then test on one of these shapes, performance may be better than on data you have never seen (overlearning). Thus you should subdivide the training data into your own training and test portions to get a more realistic idea of performance.
4. Now using these training and test partitions, you can use EvaluateShapeModelsOnCompletion.m to compare each model. Which do you think is the best of the three?
5. Now for the fun part: your goal is to create a new algorithm that performs better than any of the 3 models I have given you. You can create as many as you like, and document them in your report, but only one will be evaluated on the test set. In the near future I will distribute a template for the function you will use to evaluate your method on the test set.

Grading: This assignment is worth 20% of the course mark. Grades will be determined by the quality of the report submitted and the accuracy of your algorithms. You also get points if you find any bugs in code I provide!!