UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

ECE 544NA Pattern Recognition

Homework 2

Fall 2013

Assigned: Thursday, September 5, 2013 Due: Thursday, September 12, 2013

Reading: NNPR Chapter 2, or PRML Chapters 2 and 9

Problem 2.1

A diagonal-covariance Generalized Gaussian pdf has a form similar to a Gaussian, but decays as the L_p-norm of the distance from the mean, not the L_2-norm. This can be written as

$$p(\vec{x}) \propto \frac{1}{\prod_{d=1}^{D} \lambda_{(d)}} \exp\left(- \sum_{d=1}^{D} \left(\frac{x_{(d)} - \mu_{(d)}}{\lambda_{(d)}} \right)^p \right)$$

(a) Suppose you are given N vector observations, $\vec{x} \in \Re^D$. Find the maximum likelihood estimators of the parameters $\vec{\mu}$ and $\vec{\lambda}$.

(b) Suppose that you have prior information about the centroid vector $\vec{\mu}$: suppose you are told that it's most *a priori* probable value is $\vec{\mu} \approx \vec{\nu}$, with confidence $1/\alpha_{(d)}$ in the d^{th} dimension, thus

$$p(\vec{\mu}) \propto \frac{1}{\prod_{d=1}^{D} \alpha_{(d)}} \exp\left(- \sum_{d=1}^{D} \left(\frac{\mu_{(d)} - \nu_{(d)}}{\alpha_{(d)}} \right)^p \right)$$

In addition to this prior information, you are also given N observations, $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$. Find the MAP estimator of $\vec{\mu}$.

Matlab Exercises

Problem 2.2

Humans naturally group similar colors together; for example, we might describe a particular outfit as "a maroon sweater with navy blue shorts," ignoring the many subtle variations of color tone present in the sweater. Machines are not so good at clustering colors.

One of the problems is that machines are more sensitive to small variations in luminance than humans are; humans are relatively more sensitive to chrominance changes, and less sensitive to luminance. We can make the machine completely *insensitive* to luminance by normalizing out the luminance, thus if $\vec{x}_{\vec{n}} = [r_{\vec{n}}, g_{\vec{n}}, b_{\vec{n}}]^T$ is the RGB color vector at pixel $\vec{n} = [n_1, n_2]$ of an image, we can create a pure chrominance vector as, for example,

$$\vec{y}_{\vec{n}} = \begin{bmatrix} y_r[\vec{n}] \\ y_b[\vec{n}] \end{bmatrix} = \frac{1}{r_{\vec{n}} + g_{\vec{n}} + b_{\vec{n}}} \begin{bmatrix} r_{\vec{n}} \\ b_{\vec{n}} \end{bmatrix}$$

Find an image you like, and which has two or three obviously dominant colors. Create an electronic document that you will e-mail to me as your solution to this problem, so that you don't have to waste money on color printouts. Paste a copy of your image into the electronic document.

Calculate a chrominance vector for each pixel in the image. Plot a scatter plot of the chrominance vectors, $\vec{y}_{\vec{n}}$ (e.g., `plot(y(1,:),y(2,:),'x');`). Include this scatter plot in your solution set.

Write two matlab functions called `estep` and `mstep` that compute, respectively, the E-step and M-step for training a Gaussian mixture model. Specifically, the function `estep` should accept model parameters and data, and should compute $\gamma_{\vec{n}}(j) = p(j|\vec{y}_{\vec{n}}, \theta)$ for each Gaussian component number j and for each sample vector $\vec{y}_{\vec{n}}$. The function `mstep` should take the values of $\gamma_{\vec{n}}(j)$ and the data vectors, and should return new estimates of the model parameters.

Use your code to estimate a three-component, full-covariance Gaussian mixture model of the chrominance vectors in your image. Start with centroid vectors $\vec{\mu}_j$ that are reasonable based on your histogram. Start with relatively small covariance matrices, perhaps $\Sigma_j = 0.01I$. Iterate `estep` and `mstep` several times, until the values of the parameters stop changing.

Plot one-sigma ellipses for the three components of your GMM, in a vector space with the same axes as your scatter plot. If `[Vj,Dj]` are the eigenvector and eigenvalue matrices of Σ_j, and `muj` is its centroid vector, then its one-sigma ellipse can be plotted as

```
phi = [0:0.01:(2*pi)];
ellipse = muj+sqrt(Dj(1,1))*Vj(:,1)*cos(phi)+sqrt(Dj(2,2))*Vj(:,2)*sin(phi);
plot(ellipse(1,:),ellipse(2,:),'-');
hold on;
```

where the last command holds the plot, so that you can put all three ellipses onto the same plot. Hand in this plot. You should find that the ellipses are close to the densest regions in your scatter plot.

Create a figure with a 3×5 array of sub-plots (`subplot(3,5,j);` for $1 \leq j \leq 15$). In the first column of sub-plots, use `image` or `imagesc` to show a solid-color image containing the centroid color from your first Gaussian component. You can do this by converting the chrominance vector into an RGB vector using some default luminance (perhaps `x=[muj(1);1-sum(muj);muj(2)]*255`), then creating a $100 \times 100 \times 3$ image with every pixel set equal to the default color (perhaps `A=permute(repmat(x,[1 100 100]),[3 2 1]);`, then showing the image using `image` or `imagesc`. In the first column of your figure, each row should contain a solid-color image showing the centroid color from one of the three Gaussian components.

The second column should contain solid-color images showing the color that is one standard deviation away from the centroid along the first eigenvector of the covariance, thus the j^{th} row contains the color $\mu_j + V_j(:,1) * \sqrt{D_j(1,1)}$. The third column similarly contains $\mu_j - V_j(:,1) * \sqrt{D_j(1,1)}$, the fourth column contains $\mu_j + V_j(:,2) * \sqrt{D_j(2,2)}$, and the fifth column contains $\mu_j - V_j(:,2) * \sqrt{D_j(2,2)}$.

Now train another GMM using the same data, but with a different starting point (for example, try initializing the centroids randomly according to a uniform distribution). Create an ellipse-plot showing the one-sigma ellipses that result. How different is the result? Does the result still capture the color distribution of the original image?