1. (Haykin Problem 6.22) Equations (6.77), (6.78), and (6.79) describe three important properties of the inner product $\langle f, g \rangle$ defined in Eq. (6.75). Prove the properties described in those three equations.

   Proof of (6.77):

   $$\langle f, g \rangle = \sum_{i=1}^{l} \sum_{j=1}^{n} a_i k(\mathbf{x}_i, \tilde{\mathbf{x}}_j) b_j \qquad [Eq\,6.73]$$
   $$= \sum_{j=1}^{n} \sum_{i=1}^{l} b_j k(\tilde{\mathbf{x}}_j, \mathbf{x}_i) a_i$$
   $$= \langle g, f \rangle$$

   Proof of (6.78)

   $$\langle (cf + dg), h \rangle = \sum_{m=1}^{p} e_m k(\hat{\mathbf{x}}_m, \cdot)(c \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot) + d \sum_{j=1}^{n} b_j k(\tilde{\mathbf{x}}_j, \cdot))$$
   $$= \sum_{m=1}^{p} e_m k(\hat{\mathbf{x}}_m, \cdot)(c \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot)) + \sum_{m=1}^{p} e_m k(\hat{\mathbf{x}}_m, \cdot)(d \sum_{j=1}^{n} b_j k(\tilde{\mathbf{x}}_j, \cdot))$$
   $$= c \sum_{i=1}^{l} \sum_{m=1}^{p} a_i k(\mathbf{x_i}, \hat{\mathbf{x}}_m) e_m + d \sum_{j=1}^{n} \sum_{m=1}^{p} b_j k(\tilde{\mathbf{x}}_j, \hat{\mathbf{x}}_m) e_m$$
   $$= c \langle f, h \rangle + d \langle g, h \rangle$$

   Proof of (6.79)

   $$\|f\|^2 = \langle f, f \rangle = \sum_{i=1}^{l} \sum_{j=1}^{l} a_i k(\mathbf{x}_i, \mathbf{x}_j) a_j = \mathbf{a}^T \mathbf{K} \mathbf{a} \geq 0 \qquad [\text{since } \mathbf{K} \text{ is positive semidefinite}]$$

2. (Haykin Problem 7.16)
   a. Derive the cost functional of Eq. (7.124). Then use this functional to derive the optimized $\mathbf{a}^*$ of Eq. (7.125).

      To demonstrate this, start from Equation (7.119)

      $$E_\lambda(F) = \frac{1}{2} \sum_{i=1}^{l} (d_i - F(\mathbf{x}_i))^2 + \frac{1}{2} \lambda_A \|F\|_K^2 + \frac{1}{2} \lambda_I \mathbf{f}^T \mathbf{L} \mathbf{f}$$

      then substitute $F(\mathbf{x}) = \sum_{j=1}^{N} a_j k(\mathbf{x}, \mathbf{x}_j)$ into each term.

      In the first term, since there are only desired responses for the first *l* elements, we can only accommodate those elements if F in the expression, yielding

      $$\frac{1}{2} \sum_{i=1}^{l} (d_i - F(\mathbf{x}_i))^2 = \frac{1}{2} \sum_{i=1}^{l} (d_i - \sum_{j=1}^{l} a_j k(\mathbf{x}_i, \mathbf{x}_j))^2$$
      $$= \frac{1}{2} (\mathbf{d} - \hat{\mathbf{K}} \hat{\mathbf{a}})^T (\mathbf{d} - \hat{\mathbf{K}} \hat{\mathbf{a}})$$

      where $\hat{\mathbf{a}} = [a_1, a_2, \ldots, a_l]$ and $\hat{\mathbf{K}}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) \forall\, i \in \{1, \ldots, l\}$. And if we let J be as defined on page 354 of Haykin, this whole expression is $\frac{1}{2} (\mathbf{d} - \mathbf{J} \mathbf{K} \mathbf{a})^T (\mathbf{d} - \mathbf{J} \mathbf{K} \mathbf{a})$.

      For the second term, we get

$$\frac{1}{2}\lambda_A\left\|\sum_{i=1}^{N} a_i k(\cdot,\mathbf{x}_i)\right\|_K^2 = \frac{1}{2}\lambda_A \sum_{i=1}^{N} a_i k(\cdot,\mathbf{x}_i)\sum_{j=1}^{N} a_j k(\cdot,\mathbf{x}_j) = \frac{1}{2}\lambda_A \sum_{i=1}^{N}\sum_{j=1}^{N} a_i k(\mathbf{x}_i,\mathbf{x}_j) a_j$$
$$= \frac{1}{2}\lambda_A \mathbf{a}^T \mathbf{K}\mathbf{a}$$

Finally, in the third term we see

$\mathbf{f} = [F(\mathbf{x}_1), F(\mathbf{x}_2), \ldots, F(\mathbf{x}_N)]^T$ which now corresponds to
$[\sum_{i=1}^{N} a_i k(\mathbf{x}_{1,}\mathbf{x}_i), \sum_{i=1}^{N} a_i k(\mathbf{x}_{2,}\mathbf{x}_i), \ldots \sum_{i=1}^{N} a_i k(\mathbf{x}_N,\mathbf{x}_i)]^T$ which can more succinctly be written as $\mathbf{K}\mathbf{a}$ giving us the identity $\mathbf{f}^T\mathbf{L}\mathbf{f} = \mathbf{a}^T\mathbf{K}\mathbf{L}\mathbf{K}\mathbf{a}$ since K is symmetric.

We have shown that each term of Eq. (7.124) corresponds to a term of Eq. (7.119) with the appropriate substitution of the definition of *F*. We also note, that **x** is no longer an unknown in this functional, however, **a** is, thus the change of argument from **x** to **a**.

b. Show the details of how this minimizer includes that of Eq. (7.74) for labeled examples as a special case.

If we set λ_I to 0, then Eq. (7.124) takes this form:

$$\frac{1}{2}(\mathbf{d}-\mathbf{J}\mathbf{K}\mathbf{a})^T(\mathbf{d}-\mathbf{J}\mathbf{K}\mathbf{a}) + \frac{1}{2}\lambda_A \mathbf{a}^T\mathbf{K}\mathbf{A}$$

Taking the derivative of this and setting it to 0 yields
$\frac{1}{2}(-\mathbf{d}\mathbf{J}\mathbf{K} + (\mathbf{J}\mathbf{K})^T\mathbf{J}\mathbf{K}\mathbf{a}) + \frac{1}{2}\lambda_A\mathbf{K}\mathbf{a} = 0$ and since only labeled samples are used, J = I, removing the factor ½, we get $-\mathbf{d}\mathbf{K} + \mathbf{K}\mathbf{K}\mathbf{a} + \lambda_A\mathbf{K}\mathbf{a} = 0$ or $-\mathbf{d} + \mathbf{K}\mathbf{a} + \lambda_A\mathbf{I}_N\mathbf{a} = 0$ yielding $(\mathbf{K} + \lambda_A\mathbf{I}_N)\mathbf{a} = \mathbf{d}$ or $\mathbf{a} = (\mathbf{K} + \lambda_A\mathbf{I}_N)^{-1}\mathbf{d}$ which is exactly the form of Eq. (7.74).

3. Write a formula describing the function defined by a one-hidden-layer (already trained) MLP with a single output. Write the formula describing the function defined a RBFN with a single output. How do they differ?

The formula describing the behavior of a one-hidden-layer MLP with a single output is
$F(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{W}\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$ where W is the weight matrix associated with the hidden layer, **w** are the weights associated with the output layer, and the **b**'s are the bias vectors and **φ** applies the activation function pointwise to its argument to yield a vector result.

The formula for an RBFN with a single output is
$F(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$ where **w** is the vector of weights associated with the RBF centers, and **φ** represents the vector yielded by applying each individual RBF to the input vector **x**.

There are several differences worth noting. First, the MLP forms a linear combination of the input vectors before applying the nonlinear activation function. The RBF on the other hand, does not form such a linear combination of the inputs. Second, the MLP applies the same nonlinear function to each of the nodes, whereas the RBF applies a nonlinear function of the same form, but having a different parameter (its center) at each node. Third the RBM forms a linear combination of the hidden layer responses to form an output, whereas the MLP applies the activation function to this linear combination. Finally, this formulation of an RBFN does not employ a bias.

4. (Haykin Problem 7.17) Compare the computational complexity of the Laplacian regularized least-squares algorithm with that of the regularized least-squares algorithm using labeled examples only.

   Solving a typical regularized least-squares problem requires forming the vector of coefficients **a** as follows: $\mathbf{a}=\left(\mathbf{X}+\lambda_N \mathbf{I}_N\right)^{-1}\mathbf{d}$. This requires one NxN matrix inversion, addition, and multiplication by an N vector. To perform Laplacian regularized least-squares, we must add to this the formation of the Gram matrix **K** and Laplacian matrix **L** of the data graph. To form the gram matrix requires N^2 operations. To form the Laplacian requires another N^2 operations unless the weight vector is formed using the same function as the Gram matrix.

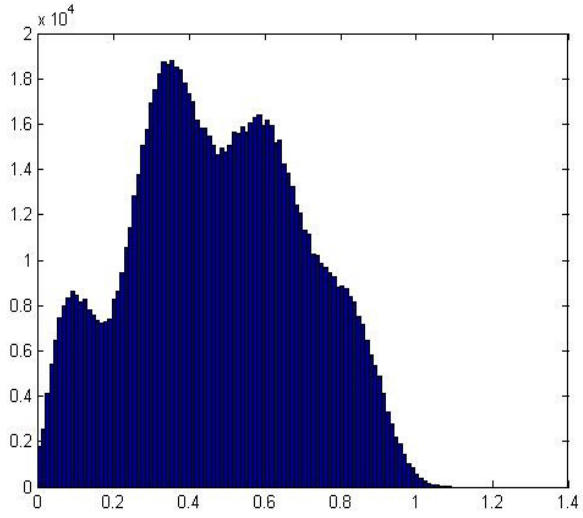
   .

5. Consider the three data sets used with Program 3. Construct a histogram of the distances between all the points in each data set and display it using the Matlab `hist` function with at least 100 bins. Discuss how you might be able to choose a value for ε for spectral graph construction using data set 2. Discuss the challenges presented by data sets 1 and 3.

   To answer this problem, I loaded each of the datasets, one at a time, populating the matrix X with the sample points, executed the following Matlab instructions for each dataset:

```
N = size(X,1);
X1 = X(:,1);
X2 = X(:,2);
X1s = repmat(X1,[1,N]);
X2s = repmat(X2,[1,N]);
XX1 = X1s(:);
XX1t = X1s';
XX1t = XX1t(:);
XX2 = X2s(:);
XX2t = X2s';
XX2t = XX2t(:);
figure;
dists = (((XX1 - XX1t).^2+(XX2-XX2t).^2).^0.5);
hist(dists,100);
```

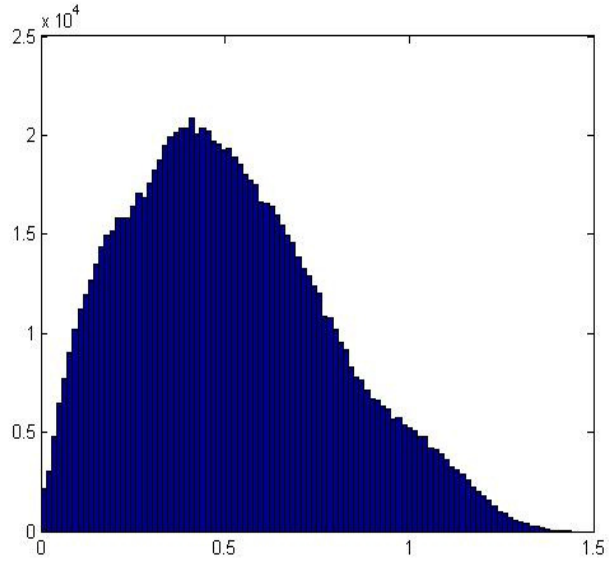
   This process yielded the following histogram for dataset 2:



   This looks like a mixture of Gaussians with several components. But it is not clear what aspects of the data give rise to these components. Clearly, there are points close together in each of the

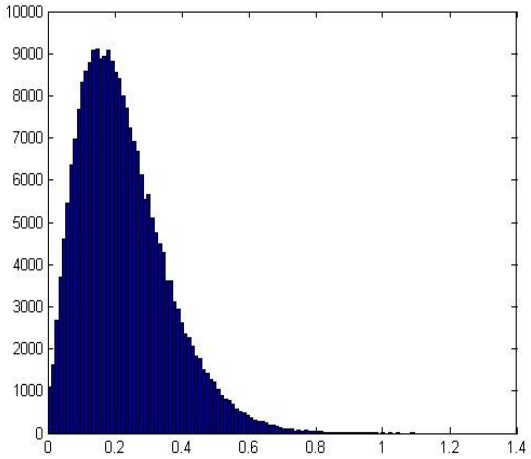
classes, the cross and the blobs. The greatest distances will arise from opposite blobs, but antipodal points in the cross will be nearly as far apart. This would likely be the cause of the two rightmost peaks at about .6 and .8. Points in each class are very near points in the same class, so it's likely that the leftmost peak arises due to points in all the classes. The second peak may be due to distances between points in blob pairs with like x or y values. To choose an ε for this dataset, one must find a value that will end up connect samples to samples in the same group, but not connect them to samples from another group. We should probably choose a value that is a relatively small value in the first Gaussian component, that is, a value between 0 and 0.1. And, of course, erring on the small side will likely yield fewer problems, so a value close to nearer 0 is probably.

The process yielded the following histogram for dataset 1



This appears somewhat similar to a mixture of Gaussian, however, it's not clear. Inspecting the data indicates that an ε on the order of about 0.05 might be desirable if we want to split the classes. There is no apparent feature in this histogram to indicate that this would be an appropriate value.

From dataset 3, we get the following histogram:



I see absolutely nothing here to indicate a reasonable choice of ε.

One this we might do is try to use the k-nearest neighbors to determine an appropriate distance (or use these neighbors directly to form a spectral graph. We can calculate the *k* nearest neighbors as follows, looking for the smallest distance to the *k*th nearest neighbor (in this case *k* is chosen to be 5:

```
D = zeros(N,N);
for i = 1:N, ind = (i-1)*N + 1; D(i,:) = sort(dists(ind:ind+N-1));end
f = min(D(:,6));
```

Doing this yields the distances 0.0111 for dataset 1, 0.0093 for dataset 2, and 0.0072 for the third dataset.