CPSC 636-600 Homework 3 (Total 100 points)

See the course web page for the **due date** and **submission info**.

Instructor: Yoonsuck Choe

March 13, 2017

Problem 1 (Written: 10 pts): Consider the XOR problem:

x_0	x_1	d
0	0	0
0	1	1
1	0	1
1	1	0

Solve the XOR problem using RBF network that has four hidden units, each centered at one of the four inputs (0,0), (0,1), (1,0), and (1,1). Use the RBF function shown in slide05, page 8 (combine this with slide05 page 31 to make the standard deviation [spread] adjustable).

1. Visualize each RBF in 3D (x and y axes as the input, and the z axis as the $\phi_i(\mathbf{x})$ value).

```
x = meshgrid(-3:0.1:3); # generate matrix of x coords
y = x'; # generate matrix of y coords
imaegsc(x); # show x coord matrix
imaegsc(y); # show y coord matrix
imagesc(x,y,sqrt(x.^2+y.^2)); # show distance from center matrix
surf(x,y,exp(-(sqrt(x.^2+y.^2)).^2)); # visualize single RBF
```

2. Construct the matrix $\boldsymbol{\phi}$ for the RBF network (slide05, page 13) and compute the inverse $\boldsymbol{\phi}^{-1}$.

3. Calculate the linear weights ($\mathbf{w}$) of the output layer of the network (slide05, page 14).

Problem 2 (Written: 10 pts): Repeat problem 1 with only two hidden units, centered at $\mathbf{t}_1 = [0, 1]^T$ and $\mathbf{t}_2 = [1, 0]^T$.

Note: Instead of $\boldsymbol{\phi}^{-1}$, you need to find $\boldsymbol{\phi}^{+}$, the pseudo-inverse (slide05, page 15).

Problem 3 (Program: 40 pts): Write two Matlab/Octave programs (or Python, etc.) to conduct (1) RBF learning and (2) Generalized RBF learning, both that take a 2D input and generates a 1D output ($\lambda = 0$). That is, there are two input units and one output unit.

- For RBF, use the inputs themselves as the hidden unit center.
- For GRBF, compare two approaches: using fixed centers selected at random vs. self-organized selection of centers.
- For both RBF and GRBF, experiment with different standard deviations.

Programming hints:

It may be convenient to write two separate functions,

- one to figure out the hidden unit centers based on the input,
- and one that takes in as its arguments the input vectors, target values, hidden unit centers, and standard deviations, and returns the ϕ matrix, the weight vector, and the output values. Testing with new inputs can be done by generating a new ϕ matrix but using the weight vector obtained from the training run.

You can quickly calculate the ϕ matrix or the $\mathbf{G}$ matrix using matrix operations.

```
# The input: each row is one input vector
octave:1> x = [0 0 ; 0 1; 1 0 ; 1 1]
x =

  0  0
  0  1
  1  0
  1  1

# The target: each row is one target
octave:2> d = [0 1 1 0]'
d =

  0
  1
  1
  0

# Extract 1st vector component of the input vector
octave:3> x1 = x(:,1)
x1 =

  0
  0
  1
  1

# Extract 2nd vector component of the input vector
octave:4> x2 = x(:,2)
x2 =

  0
  1
  0
  1
```

```
# Center vectors: each row is one center vector
octave:5> t = [0 1; 1 0]
t =

  0  1
  1  0

# Extract 1st vector component of the center vector
octave:6> t1 = t(:,1)
t1 =

  0
  1

# Extract 2nd vector component of the center vector
octave:7> t2 = t(:,2)
t2 =

  1
  0


# Calculate input count and hidden layer count
octave:8> [m,junk] = size(t)
m = 2

octave:9> [N,junk] = size(x)
N = 4

# Make t1 matrix (N x m)
octave:10> T1=ones(N,1)*t1'
T1 =

  0  1
  0  1
  0  1
  0  1

# Make t2 matrix (N x m)
octave:11> T2=ones(N,1)*t2'
T2 =

  1  0
  1  0
  1  0
  1  0

# Make x1 matrix (N x m)
```

```
octave:12> X1= x1*ones(1,m)
X1 =

  0  0
  0  0
  1  1
  1  1

# Make x2 matrix (N x m)
octave:13> X2=x2*ones(1,m)
X2 =

  0  0
  1  1
  0  0
  1  1

# Set width of RBF
octave:14> sigma=0.4
sigma = 0.40000

# In one line, get the G matrix using matrix operations.
octave:105> G = exp(-((X1-T1).^2 + (X2-T2).^2)/(2*sigma^2))
G =

  0.0439369  0.0439369
  1.0000000  0.0019305
  0.0019305  1.0000000
  0.0439369  0.0439369

# Find pseudoinverse
octave:15> w = inv(G'*G)*G'*d
w =

  0.99045
  0.99045

# Verify answer (each row in G corresponds to one input,
# so the resulting column vector contains all outputs
# for all the inputs.
octave:16> G*w
ans =

  0.087035
  0.992367
  0.992367
  0.087035
```

Problem 4 (Written: 20 pts): Test RBF and GRBF ($\lambda = 0$) on a classification task as described below. First, construct the positive (class 1) and negative inputs (class 2):

```
pos=[randn(100,1),randn(100,1)];
neg=[randn(100,1)*sqrt(5)+5,randn(100,1)*sqrt(5)];
plot(pos(:,1),pos(:,2),'r+',neg(:,1),neg(:,2),'bx');

x=[pos;neg];
d=[ones(100,1);zeros(100,1)];
```

You can see that the two point clouds are scattered near $[0, 0]^T$ and $[5, 0]^T$, with a Gaussian distribution. For classification tasks using RBF, usually multiple output units are used, but here, let us just use 1 output unit for simplicity (class 1 if output > 0.5, and class 2 otherwise).

You can also construct a separate *test set* using the same commands and saving the results in a different variable (running `randn()` multiple times will give you a new set of points). Do not use the test set during training.

1. Test performance (classification accuracy) using RBF on the training set and the test set. Experiment with various standard deviations (spread; slide05, page 31).
2. Test performance using GRBF with bias ($\lambda = 0$), with two random centers picked from the training input set, and std 2 for both centers, on both the training set and the test set. Experiment with various standard deviations (spread).
3. Test performance using GRBF with bias ($\lambda = 0$), with two self-organized centers (see slide05, page 44), and std 2 for both centers, on both the training set and the test set. Experiment with various standard deviations (spread). Report the learned centers and compare those to the ideal centers.
4. Test performance using GRBF with bias ($\lambda = 0$), with centers $[0, 0]^T$ and $[5, 0]^T$, and std 2 for both centers, on both the training set and the test set. Compare the performance with (3). Experiment with various standard deviations (spread).
5. Based on your experiments, summarize the strengths and weaknesses of the various approaches above.

Problem 5 (Program: 20 pts): Experiment with a GRBF network that has one input unit and one output unit and n hidden units (no bias, $\lambda = 0$). If you are lazy, you can just reuse the code from problem 3 by zeroing out the 2nd column in the input and in the hidden unit centers. Make it learn $f(x) = \exp(-x^2/16)\cos(x)$ for input $-3\pi \leq x \leq 3\pi$. Set the hidden unit centers to be at an equal interval across the full range of x ($-3\pi \leq x \leq 3\pi$). Experiment with various combinations of n values (number of hidden units) and standard deviations (slide05, page 31).

1. With a fixed standard deviation of 0.5, start with 1 hidden unit and gradually increase the number of hidden units (2, 3, 4, 5, 10, 20, ...). Before you experiment, guess which combination of (standard deviation, hidden unit count) will give the best match? Explain why you think so and compare with the results. Explain why you think the result came out the way it did.
2. With a fixed number of hidden units ($n = 15$), start with a small standard deviation $= 0.2$ and gradually increase it (0.4, 0.8, 0.16, 0.32, 0.64, 1.28, ...). Before you experiment, guess which combination of (standard deviation, hidden unit count) will give the best match? Explain why you think so and compare with the results. Explain why you think the result came out the way it did.