ECE 657 Assignment 2

Spring 2023

Problem 1 (20 Marks)

Deriving Formulas for Stochastic Gradient-Based Method for Training an RBF NN

The first step in the development of the stochastic gradient-based supervised learning algorithm (seen in class) is to define the instantaneous error cost function as

$$J(n) = \frac{1}{2}|e(n)|^2 = \frac{1}{2}[y_d(n) - y(n)]^2$$

Where $y(n)$ is the actual output *and* $y_d(n)$ is the desired output at iteration (n) with

$$y(n) = \sum_{k=1}^{N} w_k(n)\emptyset\{\boldsymbol{x}(n), \boldsymbol{c}_k, \sigma_k\}$$

where $\boldsymbol{c}_k(n)$ being the center vector for the *k*-th radial function $\emptyset\{\boldsymbol{x}(n), \boldsymbol{c}_k, \sigma_k\}$ and σ_k is its spread parameter. Every other notation is similar to what we discussed in class. Then

$$J(n) = \frac{1}{2}|e(n)|^2 = \frac{1}{2}\left[y_d(n) - \sum_{k=1}^{N} w_k(n)\emptyset\{\boldsymbol{x}(n), \boldsymbol{c}_k(n), \sigma_k(n)\}\right]^2$$

If the RBF seed function $\emptyset\{\boldsymbol{x}(n), \boldsymbol{c}_k(n), \sigma_k(n)\}$ is chosen to be Gaussian kernel, $J(n)$ becomes

$$J(n) = \frac{1}{2}\left[y_d(n) - \sum_{k=1}^{N} w_k(n)\exp\left(-\frac{\|\boldsymbol{x}(n)-\boldsymbol{c}_k(n)\|^2}{\sigma_k^2(n)}\right)\right]^2$$

The update equation for the network parameters are then given by

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu_w \frac{\partial}{\partial \boldsymbol{w}} J(n)|_{\boldsymbol{w}=\boldsymbol{w}(n)}$$

$$\boldsymbol{c}_k(n+1) = \boldsymbol{c}_k(n) - \mu_c \frac{\partial}{\partial \boldsymbol{c}_k} J(n)|_{\boldsymbol{c}_k=\boldsymbol{c}_k(n)}$$

$$\sigma_k(n+1) = \sigma_k(n) - \mu_\sigma \frac{\partial}{\partial \sigma_k} J(n)|_{\sigma_k=\sigma_k(n)}$$

μ_w, μ_c, μ_σ are appropriate learning rate parameters.

Show that:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu_w e(n)\boldsymbol{\Psi}(n)$$

$$\boldsymbol{c}_k(n+1) = \boldsymbol{c}_k(n) + \mu_c \frac{e(n)w_k(n)}{\sigma_k^2(n)} \phi\{\boldsymbol{x}(n), \boldsymbol{c}_k(n), \sigma_k\}[\boldsymbol{x}(n) - \boldsymbol{c}_k(n)]$$

$$\sigma_k(n+1) = \sigma_k(n) + \mu_\sigma \frac{e(n)w_k(n)}{\sigma_k^3(n)} \phi\{\boldsymbol{x}(n), \boldsymbol{c}_k(n), \sigma_k\}\left|\left|\boldsymbol{x}(n) - \boldsymbol{c}_k(n)\right|\right|^2$$

Where

$$\boldsymbol{\Psi}(n) = [\phi\{\boldsymbol{x}(n), \boldsymbol{c}_1, \sigma_1\}, \phi\{\boldsymbol{x}(n), \boldsymbol{c}_2, \sigma_2\}, \dots, \phi\{\boldsymbol{x}(n), \boldsymbol{c}_N, \sigma_N\}]^T$$

Problem 2 (30 Marks)

Write a Python Code that creates an RBF Network to approximate the mapping defined by:

$$f(x_1, x_2) = \begin{cases} +1 & if\ x_1^2 + x_2^2 \leq 1 \\ -1 & if\ x_1^2 + x_2^2 > 1 \end{cases}$$

$$\text{over region } -2 < x_1 < 2 \text{ and } -2 < x_2 < 2$$

As a training set, use 441 randomly sampled data points defined as

$$x = (x_i, x_j)$$

Where

$$x_i = -2 + 0.2\,i \qquad i = 0,1,\dots,20$$

$$x_j = -2 + 0.2\,j \qquad j = 0,1,\dots,20$$

1. Carry out the design of RBF NN based on Gaussian kernel functions with constant spread function and using all the points in the training set as centers of the RB functions. Compare the performance results (mean square error) as you vary the spread parameter while keeping it the same for all kernel functions. Discuss your findings.
2. Perform the design of the RBF NN, using this time only 150 centers, choosing the centers using two approaches:
   a) Randomly select the centers from the input data.
   b) Use K-Means algorithm to find the centers. You can use a Kmeans function defined in sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html) or create your own.

Keep the spread parameter fix for all kernel functions. Compare the performance of this network to the one designed in part (1)

Problem 3 (30 Marks)

We need to design a Kohonen self organizing map (SOM), which gives as an output some shades of color mapped over 100 by 100 grid of neurones. The training input of the SOM are 24 colors (use shades of red, green, blue, with some yellow, teal and pink) which you can chose from the "RGB Color Table: Basic Colors" section of this page: http://www.rapidtables.com/web/color/RGB_Color.htm

Using a time varying learning rate $\boldsymbol{\alpha(k) = \alpha_0 exp(-\frac{k}{T})}$ where $\boldsymbol{k}$ is the current training epoch (starts with epoch 0), $\boldsymbol{\alpha_0 = 0.8}$, and T is the Total number of training epochs equal to 1000. Note that the epoch training involves all twenty four input samples for the 24 chosen colors to the network (hint: calibrate the color codes to values between 0 and 1, instead of being between 0 and 255). Initial weights are randomized. The topological neighbourhood $\boldsymbol{N_{i,j}(k)}$ of node (*j*) around the winning unit (*i*) is given by

$$N_{i,j}(k) = \exp\left(-\frac{d_{i,j}^2}{2\sigma^2(k)}\right)$$

Where

$$\sigma(k) = \sigma_0 \exp\left(-\frac{k}{T}\right)$$

And $d_{i,j}$ is the distance between the winning node *i* and surrounding node *j*.

a) Generate a figure of the original grid (random weights) followed by figures of the SOM after 20, 40, 100, 1000 epochs. Change the value of σ_0 = 1, 10, 30, 50, 70.
b) Draw your conclusions on how the output changes with σ_0 and the number of epochs.

Note: Basically, we need to have as the output of the SOM colors similar to: http://www.cs.hmc.edu/~kpang/nn/som.html (under “Demonstration”).