Homework Assignment 4

Kexin Huang kh2383

April 1, 2018

1. RBFN for XOR problem (20 points) (a) For the 2-dimensional XOR problem, we select the following four basis vectors:

$$\mathbf{r}^1 = [-1,-1]^\top$$
$$\mathbf{r}^2 = [1,1]^\top$$
$$\mathbf{r}^3 = [-1,1]^\top$$
$$\mathbf{r}^4 = [1,-1]^\top.$$

Show that the XOR problem is solved by the radial basis function network with the following weight vector:

$$\mathbf{w} = [1,1,-1,-1,0]^\top .$$

Answer: The XOR problem is defined as given a set of data points X: (x1,x2), if sign(x1)=sign(x2), then xor(X) is positive, otherwise, xor(X) is negative. let's see how this w solves this problem.
$f(X) = \sum_1^4 w_n * \phi_n + b$
$= e^{-[(x1+1)^2+(x2+1)^2]} + e^{-[(x1-1)^2+(x2-1)^2]} - e^{-[(x1+1)^2+(x2-1)^2]} - e^{-[(x1-1)^2+(x2+1)^2]}$
$= e^{-(x1^2+x2^2+2(x1+x2)+2)} + e^{-(x1^2+x2^2+2(-x1-x2)+2)} - e^{-(x1^2+x2^2+2(x1-x2)+2)} - e^{-(x1^2+x2^2-2(x1-x2)+2)}$
$= e^{-2*(x1^2+x2^2+2)} * [e^{x1+x2} + e^{-x1-x2} - e^{x1-x2} - e^{-x1+x2}]$
$= e^{-2*(x1^2+x2^2+2)} * (e^{x2} - e^{-x2}) * (e^{x1} - e^{-x1})$ note that we can denote as a constant $e^{-2*(x1^2+x2^2+2)}$ as it is larger than 0 and smaller than 1 because $(x1^2 + x2^2) > 0$, we denote as C.
Therefore, to determine the sign of the above equation, we only need to see the sign of $(e^{x2} - e^{-x2}) * (e^{x1} - e^{-x1})$.

We know there are 4 possible scenarios (excluding 0 which is the corner case) for X: when x1¿0, x2¿0, xor(X) is pos, and from the above equation we have $(e^{x1} - e^{-x1}) > 0$ and $(e^{x2} - e^{-x2}) > 0$, therefore, f(X)¿0 which accords with xor(X). Similarly, when x1¿0, x2¡0, xor(X) is neg, and from the above equation we have $(e^{x1} - e^{-x1}) > 0$ and $(e^{x2} - e^{-x2}) < 0$, therefore, f(X)¡0. When x1¡0, x2¡0, the above equation gives pos because $(e^{x1} - e^{-x1}) < 0$ and $(e^{x2} - e^{-x2}) < 0$, so $(e^{x2} - e^{-x2}) * (e^{x1} - e^{-x1}) > 0$ which accords with xor pos. When x1¡0, x2¿0, $(e^{x1} - e^{-x1}) < 0$ and $(e^{x2} - e^{-x2}) > 0$, f(x) gives neg also agrees with xor(X).

So from above, we can conclude this construction of W solve the problem of xor.

2. Nearest-neighbour classifier by RBFN (20 points) A nearest-neighbour classifier can be constructed as a radial basis function network by selecting all the input vectors in a training set as basis vectors. In a multi-class classification setting (i.e., there are more than two categories), provide a description on how a weight matrix could be built.

Ans: Let's denote all the training examples as $[r_1, r_2, ..., r_k]$, and its corresponding classification labels as $[(r_1, y_1), (r_2, y_2), ..., (r_k, y_k)]$, and we know $\phi(X)$ is a k X 1 matrix where i th element measures the distance between X and r_i. Let's denote $a = argmax\phi(X)$ that is $\phi(X)^a = max\phi(X)$ then we can construct $W = [0, 0, ..., \frac{1}{\phi(X)^a} * y_a, 0, 0....0]$ and the non-zero value is at the a th position. W is a 1 X k matrix. This construction of W gives $W * \phi(X)$ the right label for multi-class classification because nearest-neighbor is trying to find the closet r_i such that $dist(X, r_i)$is minimum, i.e. $\phi(X)^i$ is maximum. So by constructing W this way, we have, $W * \phi(X) = 0 * \phi(X)^1 + 0 * \phi(X)^2 + ... + \phi(X)^a * \frac{1}{\phi(X)^a} * y_a + ... + 0 * \phi(X)^k = \phi(X)^a * \frac{1}{\phi(X)^a} * y_a = y_a$ so this gives the label for the nearest neighbor of X in the whole training examples.

3. Adaptive RBFN (20 points) Unlike a fixed basis function network, an adaptive basis function network adapts basis vectors so as to maximize the classification accuracy (i.e. to minimize the empirical cost.) In order to do so, we need to be able to compute the gradient of the (logistic regression) distance function with respect to each and every basis vector. Derive this gradient

$$\nabla_{\mathbf{r}^k} D(y^*, M, \phi(\mathbf{x})),$$

assuming that M is a logistic regression classifier and that

$$\phi(\mathbf{x}) = \begin{bmatrix} \exp\left(-(\mathbf{x} - \mathbf{r}^1)^2\right) \\ \vdots \\ \exp\left(-(\mathbf{x} - \mathbf{r}^K)^2\right) \end{bmatrix}.$$

Ans: Let's first define the distance function: $D(y^*, M, \phi(\mathbf{x})) = -(y^* logM(\phi(\mathbf{x}))) + (1 - y^*)log(1 - M(\phi(\mathbf{x})))$ where $M(\phi(\mathbf{x})) = \sigma(W^T\phi(\mathbf{x}) + b)$ and we denote $W^T\phi(\mathbf{x}) + b = \mathbf{a}$. We want $\nabla_{\mathbf{r}^k} D(y^*, M, \phi(\mathbf{x})) = \frac{\partial D}{\partial \mathbf{a}} * \frac{\partial \mathbf{a}}{\partial \phi(\mathbf{x})} * \frac{\partial \phi(\mathbf{x})}{\partial \mathbf{r}^k}$

Let's see each term:

from homework 1, problem(4), we learn that $\frac{\partial D}{\partial \mathbf{a}} = -(y^* log'M) * M'(\mathbf{a}) + (1 - y^*)log'(1 - M) * M' = -(y^* \frac{1}{\sigma(W^T\phi(\mathbf{x})+b)}) * \sigma(W^T\phi(\mathbf{x}) + b) * (1 - \sigma(W^T\phi(\mathbf{x}) + b)) + (1 - y^*)(\frac{1}{1-\sigma(W^T\phi(\mathbf{x})+b)}) * \sigma(W^T\phi(\mathbf{x}) + b) * (1 - \sigma(W^T\phi(\mathbf{x}) + b))$, let's denote $\sigma(W^T\phi(\mathbf{x}) + b) = \mathbf{e}$, the above equation is $-y^*(1 - e) + (1 - y^*)e = -y^* + y^*e + e - y^*e = -y^* + e = -y^* + \sigma(W^T\phi(\mathbf{x}) + b)$

And we can see easily that $\frac{\partial \mathbf{a}}{\partial \phi(\mathbf{x})} = W^T$ and as $\phi(\mathbf{x})^k$ doesn't depend on any row vectors other than kth in the W matrix, $\frac{\partial \mathbf{a}}{\partial \phi(\mathbf{x})} = \frac{\partial \mathbf{a}}{\partial \phi(\mathbf{x})^k} = W_k$

Finally, $\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{r}^k}$. Because for $\phi(x)_i$ such that i not k, the term is like a constant, so the gradient is 0. So we can see $\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{r}^k} = \frac{\partial \phi(\mathbf{x})_k}{\partial \mathbf{r}^k} = \frac{de^{-(x-\mathbf{r}^k)^2}}{d\mathbf{r}^k} = e^{-(x-\mathbf{r}^k)^2} * (-2 * (x - \mathbf{r}^k)) *$

$(-1) = \phi(\mathbf{x})_k * (2 * (x - \mathbf{r}^k))$
Now, let's combine together, $\nabla_{\mathbf{r}^k} D(y^*, M, \phi(\mathbf{x})) = (-y^* + \sigma(W^T\phi(\mathbf{x}) + b)) * W^T * \phi(\mathbf{x})_k * (2 * (x - \mathbf{r}^k)) = -2(y^* - \sigma(W^T\phi(\mathbf{x}) + b))W_k * \phi(\mathbf{x})_k (x - \mathbf{r}^k)$

4. Programming Assignment (40 points in total) Please open `https://github.com/nyu-dl/Intro_to_ML_Lecture_Note/blob/master/homeworks/hw4.ipynb` and follow the instructions there.

See code