Homework Assignment 4

Lecturer: Kyunghyun Cho

April 4, 2017

1. (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:

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

SOLUTION:
Based on the specifications of the XOR problem, we know that points within quadrants 1 and 3 should be labeled in the negative class and points within quadrants 2 and 4 should be in the positive class. We also know that the centroids of each quadrant, and thus the basis vectors, are $[-1,1]^\top, [1,-1]^\top, [1,1]^\top, [-1,-1]^\top$. The radial basis function is also defined as $rbf(x,r) = \exp\{-(x-r)^2\}$ for all basis vectors. For each vector, this function converges to 1 as x approaches r and converges to 0 as x gets further from r. The output from this function will be a new vector corresponding to $[rbf(x,[-1,1]^\top), rbf(x,[1,-1]^\top), rbf(x,[1,1]^\top), rbf(x,[-1,-1]^\top)]$.
Since all input vectors are clustered around the basis vectors, then passing the input vectors through the rbf function will result in a one-hot vector corresponding to the quadrant it belongs to. This one-hot vector will be one of the following:
$[1,0,0,0]^\top$ in quadrant 2, $[0,1,0,0]^\top$ in quadrant 4, $[0,0,1,0]^\top$ in quadrant 1, and $[0,0,0,1]^\top$ in quadrant 3. However, we want to classify between quadrants 1 and 3 and quadrants 2 and 4. Since we earlier defined that points in quadrants 1 and 3 should be negative, we can negate their rbf vectors such that they are $[0,0,-1,0]^\top$ in quadrant 1 and $[0,0,0,-1]^\top$ in quadrant 3. The weight vector can be formed as the composition of the rbf of the inputs with the basis vectors and appended with a 0, giving us $\mathbf{w} = [1,1,-1,-1,0]^\top$.

We can then check our weight vector against the outputs from the rbf function. If we take the dot product of $\mathbf{w}$ with either $[0,0,1,0]^\top$ or $[0,0,0,1]^\top$, we get -1. Similarly if we take the dot product with either $[1,0,0,0]^\top$ or $[0,1,0,0]^\top$ we get 1, which is our desired output and thus proving that this weight vector will perfectly solve the XOR problem.

2. 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.

In multi-class classification, we said that each weight vector of the weight matrix corresponded to a particular class. Each value within that weight matrix described how much that particular feature of the input vector affected the classification for that particular class. When an input is passed through an RBF, we essentially trying to cluster our input by determining the distance from each basis vector, which are presumably the centroids of each cluster. Theoretically, the output will have a single value close to 1, corresponding to the closest basis vector, and several values close to 0. Therefore, when determining the likelihood of the input vector belonging to a specific class, we only care about the corresponding value in the output vector. In other words, to determine the likelihood of an input vector belonging to class i, we only care about the i^{th} entry in the output vector of the RBF. The weight matrix corresponding to this would thus simply be a composition of one-hot vectors for each class.

3. 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

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

assuming that M is a logistic regression classifier and that

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

SOLUTION:
$D(M(\phi(x)),M,\phi(x)) = -(y^* \log M(\phi(x)) + (1-y^*)\log(1-M(\phi(x))))$
This problem is very similar to exercise 3 of homework 1, in which we computed the gradient of the logistic regression distance function for binary classification. For that problem, the gradient was computed as such:
Let $a = -y^* \log M(\mathbf{x})$ and $b = \log(1-M(\mathbf{x}))$
We also know that $M(\mathbf{x}) = \sigma(\mathbf{w}^\top \tilde{\mathbf{x}})$
Based off this, we can say $a = -y^* \log(\frac{1}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}})$ and $b = \log(1-\frac{1}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}})$
$b = \log(1-\frac{1}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}) = \log(\frac{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}-1}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}) = \log(\frac{e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}) = \log(e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}) - \log(1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}})$
$\frac{db}{d\mathbf{w}^\top} = \frac{-\tilde{\mathbf{x}}e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}{e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}} + \frac{\tilde{\mathbf{x}}e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}} = \frac{-\tilde{\mathbf{x}}(1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}})+\tilde{\mathbf{x}}e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}} = \frac{\tilde{\mathbf{x}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}$
$\frac{da}{d\mathbf{w}^\top} = \frac{-y^*\tilde{\mathbf{x}}e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}$
Bringing this together:
$\nabla_{\mathbf{w}} D(M^*(\mathbf{x}),M,\mathbf{x}) = \frac{-y^*\tilde{\mathbf{x}}e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}} - \frac{y^*\tilde{\mathbf{x}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}} + \frac{\tilde{\mathbf{x}}}{1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}}$
$= -y^*\tilde{\mathbf{x}}e^{-\mathbf{w}^\top \tilde{\mathbf{x}}}\sigma(\mathbf{w}^\top \tilde{\mathbf{x}}) - y^*\tilde{\mathbf{x}}\sigma(\mathbf{w}^\top \tilde{\mathbf{x}}) + \tilde{\mathbf{x}}\sigma(\mathbf{w}^\top \tilde{\mathbf{x}}) = \tilde{\mathbf{x}}\sigma(\mathbf{w}^\top \tilde{\mathbf{x}})(-y^*(1+e^{-\mathbf{w}^\top \tilde{\mathbf{x}}})+1)$
$= \tilde{\mathbf{x}}\sigma(\mathbf{w}^\top \tilde{\mathbf{x}})(\frac{-y^*}{\sigma(\mathbf{w}^\top \tilde{\mathbf{x}})}+1) = \tilde{\mathbf{x}}(-y^* + \sigma(\mathbf{w}^\top \tilde{\mathbf{x}}))$
The only difference between the computations above and what we're looking for is that we need to substitute $\mathbf{x}$ for $\phi(x)$. The gradient of $\phi(x)$ with respect to r^k is simply $-2w_k\phi_k(x)(x-r^k)$. Due to the fact that the derivative of $e^{f(x)}$ is simply $f'(x)e^{f(x)}$, we can simply substitute $\tilde{\mathbf{x}}$ with the gradient of $\phi(x)$. This gives us:
$\nabla_{\mathbf{r}^k} D(y^*,M,\phi(\mathbf{x})) = (-y^* + \sigma(\mathbf{w}^\top \phi(x)))(2w_k\phi_k(x)(x-r^k))$
$= -2(y^* - \sigma(\mathbf{w}^\top \phi(x)))(w_k\phi_k(x)(x-r^k))$

4. PROGRAMMING ASSIGNMENT

k-Nearest-Neighbors

When conducting this experiment, what first came to mind was the brute force method: testing every possible distance function against every possible k value. I quickly realized that this approach was infeasible given my limited computing power and time. I decided to separate the problem into 2 tasks. The first was to determine which distance function was the best. To do so, I simply set a fixed value of k and looped through every possible distance function, recording their outputs as it progressed. Based off this, I determined that the best distance function is "Correlation".

Now knowing the best distance function, the problem merely came down to the task of determining the best k. For this, I simply ran the correlation distance function against every k from 1 to 2000, the results of which can be seen in figure 1. I noticed immediately that there was a huge dropoff in the accuracy right from the beginning. I decided to focus solely on the first 10 k values as this was the area in the first experiment that had the highest accuracies. The results of this experiment can be seen in figure 2. From the results, it's clear that the best k value is 2 as that reported the highest accuracy for this particular distance function.

Figure 1

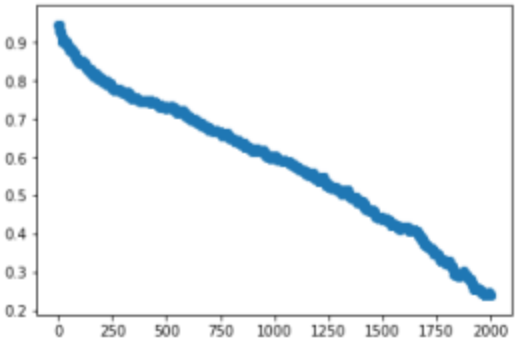
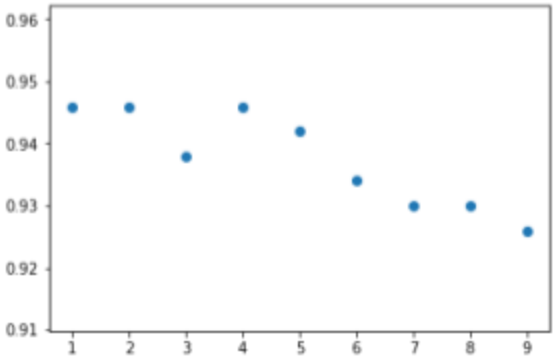


Figure 2



Radial Basis Function Network

I conducted this experiment in the same manner as KNN. I first found the optimal distance function for a fixed number of bases and then determined the optimal number of bases. After testing, I determined that the optimal distance function is canberra.

When determining the optimal number of bases, I had to modify my approach from KNN. Initially, I had tried to go through every possible value of the number of bases in increments of 10. I quickly realized that this approach was infeasible due to the tremendous computing power needed by the RBFN. I then modified it such that it increased in increments of 100. While this of course resulted in fewer data points, it still provided a sufficient amount to view an overall trend. As seen in figure 3, the results are rather interesting. The accuracies quickly rise until it hits around 1000 bases, at which point it stops increasing so much. By the time it reaches 2000, it totally plateaus. Even here though, the test set accuracies are strikingly low, with the max being

around 18%. Thus, despite low a low accuracy, the optimal number of bases is some value greater than 2000, likely at or very near the maximum number.

Figure 3