DS310
Problem set 4
Zhixuan Yang

1.(a) Let $\varphi: R \to R$ be a non-constant, bounded, monotone, continuous function. Let I_N be the N-dimensional unit hypercube in R^N. Let $C(I_N) = \{f: I_N \to R\}$ be the set of all continuous functions with domain I_N and range R. Then for any function $f \in C(I_N)$ and any $\varepsilon > 0$, $\exists$ an integer L and a sets of real values $\theta, \alpha_j, \theta_j, W_{ji}$ $(1 \leq j \leq L; 1 \leq i \leq N)$ such that

$$F(X_1, X_2 \ldots X_N) \sum_{j=1}^{L} \alpha_j \phi\left(\sum_{i=1}^{N} W_{ji} X_i - \theta_j\right) - \theta$$

$\downarrow$ uniform approximation of f

$$\forall (X_1, \ldots X_N) \in I_N, |F(X_1 \ldots X_N) - f(X_1, \ldots X_N)| < \varepsilon$$

There are (1) $\varphi: R \to R$ (2) I_N in R^N (3) $C(I_N) = \{f: I_N \to R\}$ (4) the definition of Real-valued is a function whose values are real numbers $X \to f(X)$, $X \in R$. Above four points show that all of them have same domain (Real Numbers $\to R$), and then this Proof states the universal approximation theorem for ~~real~~- valued functions defined on the N-dimensional unit hypercube. Also, real-valued function meets the requirement of UFAT

The implication of this theorem is that (1) UFAT guarantees the existence of arbitraily accurate approximations of continuous functions defined over bounded subset of R^N. In other words, it can approximate any continous function on artificial neural networks (2) It tells us the representational power a certain class of multilayer networks relative to the set of continuous functions defined on bounded subsets of R^N, and it is another important implication for the design of artificial neural networks.

These two points are the implications of this theorm for the design of neural networks. Additionally Generalized delta rule allows nonlinear function to be learned from the training data, this feature also give the benefits to the design of neural networks. If we want to UFAT To learn an unknown function, we need an algorithm to search the hypothesis space of multilayer networks. So, we also should consider this requirement,

(b) i. $Z_{jp} = 1$ iff $h_{jp} \geq 0$ and $Z_{jp} = 0$ otherwise

No since $\varphi: R \to R$ be a non-constant, bounded, monotone, continuous. since Z_{jp} would be either 1 or 0, so this kind of Z_{jp} is NOT satisfied UFAT.

ii. $Z_{jp} = h_{jp}$

No. The weight will be updated based on the error, the weights will remain same value forever, since $Z_{jp} = h_{jp}$, the error will remain same. That leads this function becomes Linear function, thus this kind of function is NOT satisfied UFAT.

iii. $Z_{jp} = \frac{1}{1+e^{-h_{jp}}}$,

Yes because this is clearly a non-constant, bounded, monotone continuous function of the inputs). Thus, it meets the requirments of UFAT.

iv. $Z_{jp} = \tanh(h_{jp}) = \frac{1-e^{-h_{jp}}}{1+e^{-h_{jp}}}$

Yes similiar to the (iii) hyperbolic function is a non-constant, bounded, monotone, and continous function. Thus, it meets the requirments of UFAT.

v. $Z_{jp} = \frac{2}{\pi} \arctan(h_{jp})$

Yes Because this function is a non-constant, bounded, monotone, and continous function. Thus, it meets the requirment of UFAT.

2. (a) $E_a = \frac{1}{2}\sum_{p=1}^{P}(d_p - O_p)^2 = \sum_{K=1}^{P}\left[\left(d_p - \sum_{j=0}^{H} u_j \frac{2}{\pi}\arctan\left(\sum_{i=0}^{N} w_{ji} x_{ip}\right)\right)\right]^2$

the update equations for u_j (Hidden - to - output).

$$\frac{\partial E_a}{\partial u_j} = \frac{\partial E_a}{\partial n_{aj}} \cdot \frac{\partial n_{aj}}{\partial u_j} \qquad \frac{\partial n_{aj}}{\partial u_j} = Z_{ja}$$

$$\frac{\partial E_a}{\partial u_a} = \frac{\partial E_a}{\partial Z_a} \cdot \frac{\partial Z_a}{\partial u_a} = -(d_a - O_a)(1)$$

$$u_j \leftarrow u_j - \eta \frac{\partial E_a}{\partial u_j} = u_j + (d_a - O_a) Z_{ja} = u_j + \delta_{aj} Z_{ja}$$

Above the update equations for u_j to minimize E_a.

the update equations for w_{ji} (input - to - hidden)

$$\frac{\partial E_a}{\partial w_{ji}} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial O_p} \frac{\partial O_p}{\partial w_{ji}} = \sum_{p=1}^{P} \frac{\partial E_a}{\partial O_p} \frac{\partial O_p}{\partial Z_p} \cdot \frac{\partial Z_p}{\partial h_{jp}} \frac{\partial h_{jp}}{\partial w_j}$$

$$= -\left(\sum_{p=1}^{P} \delta_{jp}(w_{ji}) \frac{2}{\pi} \arctan(X_{jp} - u_j)\right)(X_{ip})$$

$$= -\left(\sum_{p=1}^{P} d_{jp}(w_{ji}) \frac{2}{\pi} \arctan(X_{jp} - u_j)\right)(X_{ip})$$

$$w_{ji} \leftarrow w_{ji} + \eta\, d_{jp}(w_{ji}) \frac{2}{\pi} \arctan(X_{jp} \cdot w_{ji})(x_{ip})$$

Above are update equations for u_j and w_{ji} so as to minimize E_a

2(b) i. Use of a second order Taylor-series approximation of the error function is a very good choice to instead of first order approximation because taylor-series can converse any function to polynomial. So, it helped neural networks to solve a lot of computation, so I agree with this recommendations. good

ii. It is good because use of momentum term allows the effective learning rate for each weight to adapt as needed and helps speed up convergence. So, it is good.

iii. I think it is a good recommendation because the standard error function could reduce the variance, in other words, the robust of this neural network is increased, so it is good

iv. Randomize the order of presentation of training examples from one pass to the next helps avoid local minima. Thus, this suggestion is very good

V. Introduce small amounts of noise in the weight updates during training helps improve generalization - minimizes over-fitting, makes the learned approximation more robust to noise, and helps avoid local minima, Overall, above statement shows this suggestion is very good.

3. (a) i. Since $E = E_a + \lambda E_b$, where λ is a user-defined non-negative constant and $E_b = \sum_{p=1}^{P} \sum_{i=0}^{N} \left(\frac{\partial E_a}{\partial x_i}\right)^2$, ~~thus the~~ the error is becoming large, which means final output will not very accurate to desired output. For this situation, the sensitivity of the network output is small, which means the noise would not influence the output dramatically. Overall, we can see that this modified error function "increased" the robust of this neural network.

ii. This modified error function generalized the weights and bias, in other words, if the error is a constant, the neural network would high likely to have a risk of over-fitting. To solve this problem, this new error function provide the ability that deal with different inputs to this neural works. However, the new error function generalized the capability of the network, and reduce the risk of over-fitting.

(b) $u_j \leftarrow u_j - \eta \frac{\partial(E_a + E_b)}{\partial u_j} = u_j + (d_a - o_a) z_{ja} + (d_b - o_b) z_{jb}$

$= u_j + \delta_{aj} z_{ja} + \delta_{bj} z_{bj}$.

Above is the function to update. u_j

$\frac{\partial(E_a + E_b)}{\partial w_{ji}} = \sum_{p=1}^{P} \frac{\partial(E_a + E_b)}{\partial o_p} \frac{\partial o_p}{\partial w_j} = \sum_{p=1}^{P} \frac{\partial E_a}{\partial o_p} \cancel{\frac{\partial o_p}{\partial z_a}} \cdot \frac{\partial z_p}{z_{njb}} = \frac{\partial n_j}{\partial w_j}$

$w_{ji} \leftarrow w_{ji} \quad \eta \delta_{jp}(w_{ji}) \frac{2}{\pi} \arctan(x_{jp} \cdot x_{ip})(x_{ip})$ $+''$

$E(\sigma)$

$+E_b$