Deep Neural Networks

Part I: What are Deep Neural Networks

I. Introduction to Neural Networks

Deep neural networks are similar to two layer networks however there are now more layers. Consider our simple two-layer example

$$f(x) = W^\top \sigma(\Phi(\xi)) + \varepsilon(x)$$

where for $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{L \times n}$, $\sigma(\Phi(\xi)) \in \mathbb{R}^L$, $\Phi(\xi) \in \mathbb{R}^l$ is $\Phi(\xi) = V^\top \xi$, $V \in \mathbb{R}^{n+1 \times l}$, $\xi \in \mathbb{R}^{n+1}$ is $\xi = \begin{bmatrix} x \\ 1 \end{bmatrix}$, and $\varepsilon(x) \in \mathbb{R}^n$.

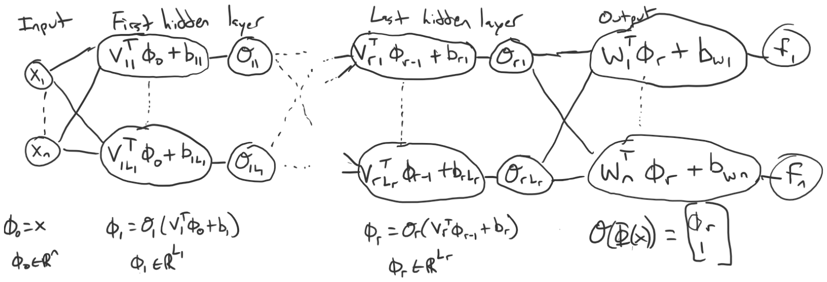
A deep network extends this idea to contain many inner weights (the hidden layers) enabling the network to estimate more complicated functions with higher accuracy (potentially). Now what we use is

$$f(x) = W^\top \sigma(\Phi(x)) + \varepsilon(x),$$

where

$$\Phi(x) \triangleq (\phi_r \circ \phi_{r-1} \circ \ldots \phi_2 \circ \phi_1)(x),$$

$\phi_l = \sigma_l\left(V_l^\top \phi_{l-1} + b_l\right)$, $l \in \{1, ...r\}$, $r \in \mathbb{Z}_{\geq 1}$ denotes the user-defined number of inner layers of the DNN, $\phi_0 = x$ is the input to DNN, $\sigma_l \in \mathbb{R}^{L_l}$ is the activation function for the lth inner layer, L_l is the size of the lth inner layer, $V_l \in \mathbb{R}^{L_{l-1} \times L_l}$ denotes the ideal constant weight matrix for the lth inner layer, $b_l \in \mathbb{R}^{L_l}$ denotes the ideal constant bias column matrix for the lth inner layer, $W \in \mathbb{R}^{L \times n}$ still denotes the ideal output layer weight matrix, $\sigma \in \mathbb{R}^L$ denotes the activation functions output layer including the bias, $L \in \mathbb{Z}_{>0}$ denotes the output layer size including the output bias $L = L_r + 1$, $\varepsilon \in \mathbb{R}^n$ denotes the unknown bounded function reconstruction error. Using all of this, $\Phi \in \mathbb{R}^{L_r}$ is essentially the features of the DNN. Consider the following graphic representation of this architecture.


Input
First hidden layer
Last hidden layer
Output
x_1
x_n
$V_{11}^\top \phi_0 + b_{11}$
$V_{1L_1}^\top \phi_0 + b_{1L_1}$
σ_{11}
σ_{1L_1}
$V_{r1}^\top \phi_{r-1} + b_{r1}$
$V_{rL_r}^\top \phi_{r-1} + b_{rL_r}$
σ_{r1}
σ_{rL_r}
$W_1^\top \phi_r + b_{w1}$
$W_n^\top \phi_r + b_{wn}$
f_1
f_n
$\phi_0 = x$
$\phi_0 \in \mathbb{R}^n$
$\phi_1 = \sigma_1(V_1^\top \phi_0 + b_1)$
$\phi_1 \in \mathbb{R}^{L_1}$
$\phi_r = \sigma_r(V_r^\top \phi_{r-1} + b_r)$
$\phi_r \in \mathbb{R}^{L_r}$
$\sigma(\Phi(x)) = \begin{bmatrix} \phi_r \\ 1 \end{bmatrix}$

II. How to Use the Deep Network

Now the big question becomes how do we use and update the inner features? Recall we have

$$M(\phi)\ddot{\phi} + C\left(\phi, \dot{\phi}\right) + G(\phi) + \tau_d\left(\phi, \dot{\phi}\right) = \tau$$

$$M(\phi) \triangleq \begin{bmatrix} m_1 l_1^2 + m_2\left(l_1^2 + 2 l_1 l_2 c_2 + l_2^2\right) & m_2\left(l_1 l_2 c_2 + l_2^2\right) \\ m_2\left(l_1 l_2 c_2 + l_2^2\right) & m_2 l_2^2 \end{bmatrix},$$

$$C\left(\phi, \dot{\phi}\right) \triangleq \begin{bmatrix} -2 m_2 l_1 l_2 s_2 \dot{\phi}_1 \dot{\phi}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} -m_2 l_1 l_2 s_2 \dot{\phi}_2^2 \\ m_2 l_1 l_2 s_2 \dot{\phi}_1^2 \end{bmatrix},$$

$$G(\phi) \triangleq \begin{bmatrix} (m_1 + m_2) g l_1 c_1 + m_2 g l_2 c_{12} \\ m_2 g l_2 c_{12} \end{bmatrix}.$$

And we approximate $\tau_d\left(\phi, \dot{\phi}\right)$ as

$$\tau_d\left(\phi, \dot{\phi}\right) = W^\top \sigma\left(\Phi\left(\phi, \dot{\phi}\right)\right) + \varepsilon\left(\phi, \dot{\phi}\right)$$

A. *How we did it using two-layer*

Lets start by looking at how we did it for the two-layer we did

$$\Phi\left(\phi,\dot{\phi}\right)=V^{\top}\xi$$

$$\xi=\begin{bmatrix}\phi\\ \dot{\phi}\\ 1\end{bmatrix}.$$

Using this we ended up designing the input as

$$\tau=Y\widehat{\theta}+\widehat{W}^{\top}\widehat{\sigma}+e+\beta_r r+\beta_\delta\text{sgn}\left(r\right)$$

where we had

$$\widehat{\sigma}\triangleq\sigma\left(\widehat{V}^{\top}\xi\right).$$

Then we did updates for the output and inner weights as

$$\dot{\widehat{W}}=\text{proj}\left(\Gamma_W\widehat{\sigma}r^{\top}\right)$$

and

$$\dot{\widehat{V}}=\text{proj}\left(\Gamma_V\xi r^{\top}\widehat{W}^{\top}\widehat{\sigma'}\right)$$

where

$$\widehat{\sigma'}\triangleq\frac{\partial\sigma}{\partial\Phi}\left(\widehat{\Phi}\right)=\begin{bmatrix}\frac{\partial\sigma_1}{\partial\Phi_1}\left(\widehat{\Phi}_1\right) & 0 & \cdots\\ 0 & \ddots & \vdots\\ \vdots & \cdots & \frac{\partial\sigma_l}{\partial\Phi_l}\left(\widehat{\Phi}_l\right)\\ 0 & \cdots & 0\end{bmatrix}.$$

Now how do we extend this to deep networks?

B. *Using the deep network for* Φ

So what is the change? While there are methods to expand the two-layer style approach to update inner features, we will focus on optimization-based methods. Specifically, we will use a switched approximation of the inner features as

$$\widehat{\Phi}_k=\left(\widehat{\phi}_{rk}\circ\widehat{\phi}_{r-1\,k}\circ\cdots\circ\widehat{\phi}_{2k}\circ\widehat{\phi}_{1k}\right)\left(\phi,\dot{\phi}\right)$$

$$\widehat{\phi}_{lk}\triangleq\sigma_l\left(\widehat{V}_{lk}^{\top}\widehat{\phi}_{l-1\,k}+\widehat{b}_{lk}\right)$$

where $\widehat{\Phi}_k$ is the kth estimate of the inner weights, $\widehat{\phi}_{lk}$ is the kth estimate of the lth layer output, $\widehat{V}_{lk}$ is the kth estimate of the lth layer weights, and $\widehat{b}_{lk}$ is the kth estimate of the lth layer bias, with $\widehat{\phi}_{0k}=\begin{bmatrix}\phi\\ \dot{\phi}\end{bmatrix}$. Now, using this approach we have

$$\widehat{\sigma}_k\triangleq\sigma\left(\widehat{\Phi}_k\right)$$

and in the analysis we no longer include the inner weights in the analysis. Instead we will later define an optimization approach to update the inner features. Lets examine the analysis for this.

C. *Error Systems*

Recall we used

$$r=\dot{e}+\alpha e$$

and were able to get the error system into the form

$$M\dot{r}=Y\theta+W^{\top}\sigma+\varepsilon-\tau-\frac{1}{2}\dot{M}r.$$

Previously in the two-layer we would do an approximate of σ which is now not used since we will not update the inner weights using the analysis.

D. Analysis

$$\zeta \triangleq \begin{bmatrix} e \\ r \\ \widetilde{\theta} \\ \text{vec}\left(\widetilde{W}\right) \end{bmatrix}$$

$$\text{vec}\left(\widetilde{W}\right) = \begin{bmatrix} \widetilde{W}_{11} \\ \vdots \\ \widetilde{W}_{L1} \\ \widetilde{W}_{12} \\ \vdots \\ \widetilde{W}_{L2} \end{bmatrix}$$

and

$$V(\zeta, t) \triangleq \frac{1}{2}e^\top e + \frac{1}{2}r^\top M(\phi) r + \frac{1}{2}\widetilde{\theta}^\top \Gamma_\theta^{-1}\widetilde{\theta} + \frac{1}{2}\text{tr}\left(\widetilde{W}^\top \Gamma_W^{-1}\widetilde{W}\right)$$

taking the time derivative

$$\dot{V}(\zeta, t) = e^\top \dot{e} + \frac{1}{2}r^\top \dot{M}(\phi) r + r^\top M(\phi)\dot{r} + \widetilde{\theta}^\top \Gamma_\theta^{-1}\dot{\widetilde{\theta}} + \text{tr}\left(\widetilde{W}^\top \Gamma_W^{-1}\dot{\widetilde{W}}\right)$$

then we can use

$$\begin{aligned} r &= \dot{e} + \alpha e \\ \implies \dot{e} &= r - \alpha e \end{aligned}$$

and

$$M\dot{r} = Y\theta + W^\top \sigma + \varepsilon - \tau - \frac{1}{2}\dot{M}r$$

$$\begin{aligned} \dot{V}(\zeta, t) = {} & e^\top (r - \alpha e) + \frac{1}{2}r^\top \dot{M}(\phi) r + r^\top \left(Y\theta + W^\top\sigma + \varepsilon - \tau - \frac{1}{2}\dot{M}r\right) \\ & - \widetilde{\theta}^\top \Gamma_\theta^{-1}\dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1}\dot{\widehat{W}}\right) \end{aligned}$$

$$\begin{aligned} \dot{V}(\zeta, t) = {} & -e^\top \alpha e + r^\top e + r^\top \left(Y\theta + W^\top\sigma + \varepsilon - \tau\right) \\ & - \widetilde{\theta}^\top \Gamma_\theta^{-1}\dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1}\dot{\widehat{W}}\right) \end{aligned}$$

1) Input Design: Now we can design the input

$$\tau = Y\widehat{\theta} + \widehat{W}^\top \widehat{\sigma}_k + e + \beta_r r + \beta_\delta \text{sgn}(r)$$

which yields

$$\begin{aligned} \dot{V}(\zeta, t) = {} & -e^\top \alpha e + r^\top e + r^\top \left(Y\theta + W^\top\sigma + \varepsilon - \left(Y\widehat{\theta} + \widehat{W}^\top\widehat{\sigma}_k + e + \beta_r r + \beta_\delta \text{sgn}(r)\right)\right) \\ & - \widetilde{\theta}^\top \Gamma_\theta^{-1}\dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1}\dot{\widehat{W}}\right) \end{aligned}$$

$$\begin{aligned} \dot{V}(\zeta, t) = {} & -e^\top \alpha e - r^\top \beta_r r + r^\top \left(Y\widetilde{\theta} + W^\top\sigma - \widehat{W}^\top\widehat{\sigma}_k + \varepsilon - \beta_\delta \text{sgn}(r)\right) \\ & - \widetilde{\theta}^\top \Gamma_\theta^{-1}\dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1}\dot{\widehat{W}}\right) \end{aligned}$$

Now look at $W^\top\sigma - \widehat{W}^\top\widehat{\sigma}_k$ where we add and subtract a $W^\top\widehat{\sigma}_k$

$$W^\top\sigma - \widehat{W}^\top\widehat{\sigma}_k \pm W^\top\widehat{\sigma}_k$$

$$\to W^\top\widetilde{\sigma}_k + \widetilde{W}^\top\widehat{\sigma}_k$$

$$\widetilde{\sigma}_k = \sigma - \widehat{\sigma}_k$$

Using this we get

$$\begin{aligned}\dot{V}(\zeta,t) &= -e^\top \alpha e - r^\top \beta_r r + r^\top \left(Y\widetilde{\theta} + W^\top \widetilde{\sigma}_k + \widetilde{W}^\top \widehat{\sigma}_k + \varepsilon - \beta_\delta \text{sgn}(r) \right) \\ &- \widetilde{\theta}^\top \Gamma_\theta^{-1} \dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right)\end{aligned}$$

and let

$$\delta_k \triangleq W^\top \widetilde{\sigma}_k + \varepsilon$$

$$\begin{aligned}\dot{V}(\zeta,t) &= -e^\top \alpha e - r^\top \beta_r r + r^\top \left(Y\widetilde{\theta} + \widetilde{W}^\top \widehat{\sigma}_k + \delta_k - \beta_\delta \text{sgn}(r) \right) \\ &- \widetilde{\theta}^\top \Gamma_\theta^{-1} \dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right)\end{aligned}$$

$$\begin{aligned}\dot{V}(\zeta,t) &= -e^\top \alpha e - r^\top \beta_r r + r^\top Y\widetilde{\theta} + r^\top \widetilde{W}^\top \widehat{\sigma}_k + r^\top \delta_k - r^\top \beta_\delta \text{sgn}(r) \\ &- \widetilde{\theta}^\top \Gamma_\theta^{-1} \dot{\widehat{\theta}} + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right)\end{aligned}$$

2) Design Structured Update Law: Now we can design $\dot{\widehat{\theta}}$ using

$$r^\top Y\widetilde{\theta} - \widetilde{\theta}^\top \Gamma_\theta^{-1} \dot{\widehat{\theta}} = 0$$

$$\widetilde{\theta}^\top Y^\top r = \widetilde{\theta}^\top \Gamma_\theta^{-1} \dot{\widehat{\theta}}$$

$$\dot{\widehat{\theta}} = \text{proj}\left(\Gamma_\theta Y^\top r \right)$$

yielding

$$\begin{aligned}\widetilde{\theta}^\top Y^\top r &= \widetilde{\theta}^\top \Gamma_\theta^{-1} \Gamma_\theta Y^\top r \\ \widetilde{\theta}^\top Y^\top r &= \widetilde{\theta}^\top Y^\top r\end{aligned}$$

$$\begin{aligned}\dot{V}(\zeta,t) &= -e^\top \alpha e - r^\top \beta_r r + r^\top \widetilde{W}^\top \widehat{\sigma}_k + r^\top \delta_k - r^\top \beta_\delta \text{sgn}(r) \\ &+ \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right)\end{aligned}$$

3) Design Unstructured Update Law: Now we can design $\dot{\widehat{W}}$ using the trace trick for two vectors as before $a, b \in \mathbb{R}^n$

$$\text{tr}\left(ba^\top \right) = a^\top b$$

$$\begin{aligned}r^\top \widetilde{W}^\top \widehat{\sigma}_k + \text{tr}\left(-\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right) &= 0 \\ r^\top \widetilde{W}^\top \widehat{\sigma}_k &= \text{tr}\left(\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right)\end{aligned}$$

$$\underbrace{r^\top}_{a^\top} \underbrace{\widetilde{W}^\top \widehat{\sigma}_k}_{b} = \text{tr}\left(\underbrace{\widetilde{W}^\top \widehat{\sigma}_k}_{b} \underbrace{r^\top}_{a^\top} \right)$$

$$\text{tr}\left(\widetilde{W}^\top \widehat{\sigma}_k r^\top \right) = \text{tr}\left(\widetilde{W}^\top \Gamma_W^{-1} \dot{\widehat{W}} \right)$$

$$\dot{\widehat{W}} = \text{proj}\left(\Gamma_W \widehat{\sigma}_k r^\top \right)$$

yielding

$$\text{tr}\left(\widetilde{W}^\top \widehat{\sigma}_k r^\top \right) = \text{tr}\left(\widetilde{W}^\top \Gamma_W^{-1} \Gamma_W \widehat{\sigma}_k r^\top \right)$$

$$\text{tr}\left(\widetilde{W}^\top \widehat{\sigma}_k r^\top \right) = \text{tr}\left(\widetilde{W}^\top \widehat{\sigma}_k r^\top \right)$$

$$\dot{V}(\zeta,t) = -e^\top \alpha e - r^\top \beta_r r + r^\top \delta_k - r^\top \beta_\delta \text{sgn}(r)$$

4) Bound Approximation Errors: Now we can bound

$$r^\top \delta_k - r^\top \beta_\delta \mathrm{sgn}\left(r\right)$$

as

$$-\underline{\beta_\delta}\|r\| + \overline{\delta}\|r\| \leq 0$$
$$\underline{\beta_\delta} > \overline{\delta}$$

yielding

$$\dot{V}\left(\zeta, t\right) \leq -e^\top \alpha e - r^\top \beta_r r$$

Now here you might ask, but this is a switching update, don't we need to have dwell-time constraints or a switching analysis like we did with CL type updates to avoid instability from switches? Technically you could do an analysis on each switch but in the end you essentially have a common bound so each is bounded as above. Also, zeno behavior is excluded since it takes time to do each update to the basis (we collect data and then perform the optimization) so each switch is on a slow time scale so that is has a type of "dwell-time" that is inherently in this update. Next we will look at how we perform the optimizations.

E. Inner Feature Updates

Now lets examine how we do the optimization for the inner features. Consider the ways we were doing CL before where we formed the data-based learning regressor

$$\mathcal{Y}\left(\phi, \dot{\phi}, \ddot{\phi}\right)\theta = M\left(\phi\right)\ddot{\phi} + C\left(\phi, \dot{\phi}\right) + G\left(\phi\right).$$
$$\mathcal{Y}\left(\phi, \dot{\phi}, \ddot{\phi}\right) \triangleq Y_M\left(\phi, \ddot{\phi}\right) + Y_C\left(\phi, \dot{\phi}\right) + Y_G\left(\phi\right)$$

where

$$Y_M\left(\phi, \ddot{\phi}\right) \triangleq \begin{bmatrix} \ddot{\phi}_1 & \left(2c_2\ddot{\phi}_1 + c_2\ddot{\phi}_2\right) & \ddot{\phi}_2 & 0 & 0 \\ 0 & c_2\ddot{\phi}_1 & \left(\ddot{\phi}_1 + \ddot{\phi}_2\right) & 0 & 0 \end{bmatrix},$$

$$Y_C\left(\phi, \dot{\phi}\right) \triangleq \begin{bmatrix} 0 & -\left(2s_2\dot{\phi}_1\dot{\phi}_2 + s_2\dot{\phi}_2^2\right) & 0 & 0 & 0 \\ 0 & s_2\dot{\phi}_1^2 & 0 & 0 & 0 \end{bmatrix},$$

$$Y_G\left(\phi\right) \triangleq \begin{bmatrix} 0 & 0 & 0 & gc_1 & gc_{12} \\ 0 & 0 & 0 & 0 & gc_{12} \end{bmatrix}.$$

Now using this with our new dynamics yields

$$M\left(\phi\right)\ddot{\phi} + C\left(\phi, \dot{\phi}\right) + G\left(\phi\right) + \tau_d\left(\phi, \dot{\phi}\right) = \tau$$
$$\mathcal{Y}\left(\phi, \dot{\phi}, \ddot{\phi}\right)\theta + \tau_d\left(\phi, \dot{\phi}\right) = \tau.$$

Now lets use our approximation of $\tau_d\left(\phi, \dot{\phi}\right)$

$$\tau_d\left(\phi, \dot{\phi}\right) = W^\top \sigma\left(\Phi\left(\phi, \dot{\phi}\right)\right) + \varepsilon\left(\phi, \dot{\phi}\right)$$

to yield

$$\mathcal{Y}\left(\phi, \dot{\phi}, \ddot{\phi}\right)\theta + W^\top \sigma\left(\Phi\left(\phi, \dot{\phi}\right)\right) + \varepsilon\left(\phi, \dot{\phi}\right) = \tau$$

Now we know that θ, W, Φ, and ε are unknown so how can we use this to perform the optimization? Well consider we know τ, we have $\widehat{\theta}$ and $\widehat{W}$, and the objective of our kth optimization is to improve the kth estimate of Φ, $\widehat{\Phi}_k$. Using these we can show our approximate of the left side of the above relationship

$$\mathcal{Y}\left(\phi, \dot{\phi}, \ddot{\phi}\right)\widehat{\theta} + \widehat{W}^\top \sigma\left(\widehat{\Phi}_k\left(\phi, \dot{\phi}\right)\right)$$

Considering this, we could combine them as

$$\tau - \mathcal{Y}\left(\phi, \dot{\phi}, \ddot{\phi}\right)\widehat{\theta} + \widehat{W}^\top \sigma\left(\widehat{\Phi}_k\left(\phi, \dot{\phi}\right)\right).$$

Say we save a history of the data required to calculate this relationship as a buffer

$$\mathcal{B}\left(t\right) = \left\{\tau\left(t_h\right), \phi\left(t_h\right), \dot{\phi}\left(t_h\right), \ddot{\phi}\left(t_h\right), \mathcal{Y}\left(\phi\left(t_h\right), \dot{\phi}\left(t_h\right), \ddot{\phi}\left(t_h\right)\right)\right\}_{h=1}^{b(t)}$$

where we use the current estimates of the weights $\widehat{\theta}(t)$ and $\widehat{W}(t)$ to form a very commonly used mean squared error loss function to optimize $\widehat{\Phi}_k$ to determine $\widehat{\Phi}_{k+1}$ as

$$\widehat{\tau}(t_h) = \mathcal{Y}\left(\phi(t_h), \dot{\phi}(t_h), \ddot{\phi}(t_h)\right)\widehat{\theta}(t) + \widehat{W}^{\top}(t)\,\sigma\left(\widehat{\Phi}_k\left(\phi(t_h), \dot{\phi}(t_h)\right)\right)$$

$$\mathcal{L}_{k+1}(t) = \frac{1}{b(t)}\sum_{h=1}^{b(t)} \left\|\tau(t_h) - \widehat{\tau}(t_h)\right\|^2 .$$

Is mean squared error the only loss function we could use? No, there are many choices but this is often a good first choice for supervised training like what we are doing here.