PHY 835: Machine Learning in Physics Lecture 9: Deep Neural Network February 20, 2024

Gary Shiu

Why going deep?

- A shallow NN with only a single hidden layer can already approximate any continuous function to a specific precision, using piecewise linear functions.
- However, the network capacity (# hidden units) may be impractically large. A deep NN can produce more linear regions for a given # parameters.

Figure 3.5 Approximation of a 1D function (dashed line) by a piecewise linear Figure from Simon Prince "Understanding Deep Learning"

Composing Neural Networks functions, the required number of hidden units is impractically large. Deep networks can produce many more linear regions for a given number of parameters for a given number of parameters for a given ters. Hence, from a practical standpoint, they can be used to describe a broader family functions, the required number of hidden units is impractically large. Deep networks can produce many more linear regions than shallow networks for a given number of parameters. Hence, from a practical standpoint, they can be used to describe a broader family of functions.

- As we will see, compositions of NN are **special cases** of Deep NNs which are even more expressive. **y** = φ0 + φ3*h*² + φ3*h3*. (4.2) + φ3*h3*. (4.2) + φ3*h3*. (4.2) + φ3*h3.* diferent ranges of *x* are mapped to the same output range *y* ∈ [−1*,* 1], and the subsequent if Deep NNs which are even function defned by the second network is duplicated three times to create nine linear
- Generate piecewise linear functions. However, # linear regions is more than a single layer with 6 hidden units. To understand why, see next page. mapping from this range of *^y* to *^y*! is applied three times. The overall efect is that the Notebook 4.1 egions is more than a

Mapping multiple inputs to the same output

Figure 4.1 Composing two single-layer networks with three hidden units each. a) Three different ranges of x are mapped to the same output range $y \in [-1,1]$ and the T mapping from this range of y to y' is applied the subsequent mapping from this range of y to y' is applied three times.

> The composition creates **nine linear regions** The composition creates **nine linear regions**.

Folding Input Space . B) The frst network produces a function consisting of seven linear production consisting of seven linear production \mathcal{L} regions, one of the second network of the second network definition comprise a function comprise a function co two linear regions in *y* ∈ [−1*,* 1]. d) When these networks are composed, each of

The processes of an experimental commons conservative Commons Commons Commons Commons Commons Commons Commons C The first network "folds" the input space x back onto itself so that multiple inputs generate the same output. Then the second network applies a function, which is replicated at all points that were folded on top of one another.

Deep Neural Networks folder on the property of \mathbf{r} Then the second network applies a function, which is replicated at all points that were folded on top of one and the state \Box

• The composition of 2 shallow networks results in a 2-layer network: **4.2 From composing networks to deep networks**

• This is because we can eliminate the "mediator" y: **Fugure** we can eminique the **incutator** y. prior hooques we can oliminate the fimediates' y linear function. Substituting the expression for *y* into equation 4.3 gives:

$$
h'_1 = a[\theta'_{10} + \theta'_{11}y] = a[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1h_1 + \theta'_{11}\phi_2h_2 + \theta'_{11}\phi_3h_3]
$$

\n
$$
h'_2 = a[\theta'_{20} + \theta'_{21}y] = a[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1h_1 + \theta'_{21}\phi_2h_2 + \theta'_{21}\phi_3h_3]
$$

\n
$$
h'_3 = a[\theta'_{30} + \theta'_{31}y] = a[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1h_1 + \theta'_{31}\phi_2h_2 + \theta'_{31}\phi_3h_3],
$$

• However, a 2-layer network is more general since there are 9 unconstrained slope parameters instead of 6: ¹¹φ0*,* ψ¹¹ = θ! ¹¹φ1*,* ψ¹² = θ! 11 on. The result is a new sort is a new
11 on. The result is a new sort is a new ever. a 2-laver network $\frac{1}{2}$ ibut alifed slupe parameters instead of σ . α and α the set of the state of α which we can represent

$$
h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]
$$

\n
$$
h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]
$$

\n
$$
h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3],
$$

Deep Neural Networks

Hyperparameters

- $F_{\rm eff} = 2.64 \times 2^{3}$ • Modern deep NNs might have $\gtrsim \mathcal{O}(10^2)$ layers with $\mathcal{O}(10^3)$ of hidden units in each layer. **D**₂ $\frac{1}{2}$ *D* $\frac{1}{2}$ *<i>D* $\frac{1}{2}$ *D* $\frac{1}{2}$ *<i>D* $\frac{1}{2}$ *D* $\frac{1}{2}$
- The number of layers $K =$ **depth**, & the number of hidden units in each layer (=width) $\frac{1}{2}$ and $\frac{1}{2}$ the subsequent intervent matrix $\frac{1}{2}$ the weight matrix $\frac{1}{2}$ the weight matrix $\frac{1}{2}$ that we have $\frac{1}{2}$ the weight matrix $\frac{1}{2}$ that we have $\frac{1}{2}$ the weight matrix $\frac{1}{2$ $D_1, D_2, ..., D_K$ are hyperparameters. The network **capacity** = # number of hidden units.
- 2 × 4. It is applied to the four hidden units in a four highest one and creates the inputs to the inputs to in • For fixed hyperparameters, the model describes a family of functions, and the parameters weights) determine a specific function $\overline{}$ wolght of actommed a opeenio function. θ (known as weights) determine a specific function.

General Formulation $\sum_{n=1}^{\infty}$ *<u>,</u> allegeness of the production of the production of the set of* This notation becomes cumbersome for networks with many layers. Hence, from now on, we will describe the vector of hidden units at layer *k* as h*k*, the vector of biases

• We can express the above 2-layer network in **matrix notation**: We can explose the above $\mathsf{L}\text{-}\mathsf{layer}$ helive where the same case, the above 2 lover petucide in matrix net • We can express the above 2-layer network in **matrix notation**. (intercepts) that contribute to hidden layer *k*+1 as β*k*, and the weights (slopes) that

$$
\begin{bmatrix}\nh_1 \\
h_2 \\
h_3\n\end{bmatrix} = \mathbf{a} \begin{bmatrix}\n\theta_{10} \\
\theta_{20} \\
\theta_{30}\n\end{bmatrix} + \begin{bmatrix}\n\theta_{11} \\
\theta_{21} \\
\theta_{31}\n\end{bmatrix},
$$
\n
$$
\begin{bmatrix}\nh'_1 \\
h'_2 \\
h'_3\n\end{bmatrix} = \mathbf{a} \begin{bmatrix}\n\psi_{10} \\
\psi_{20} \\
\psi_{30}\n\end{bmatrix} + \begin{bmatrix}\n\psi_{11} & \psi_{12} & \psi_{13} \\
\psi_{21} & \psi_{22} & \psi_{23} \\
\psi_{31} & \psi_{32} & \psi_{33}\n\end{bmatrix} \begin{bmatrix}\nh_1 \\
h_2 \\
h_3\n\end{bmatrix},
$$
\n
$$
\mathbf{h} = \mathbf{a} [\theta_0 + \theta x]
$$
\nand\n
$$
\mathbf{h}' = \mathbf{a} [\psi_0 + \Psi \mathbf{h}]
$$

and

$$
y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix},
$$

$$
\begin{bmatrix} h_1' \\ h_2' \\ h_3' \end{bmatrix} = \mathbf{a} \begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, \qquad \begin{aligned} \mathbf{h} &= \mathbf{a} \begin{bmatrix} \boldsymbol{\theta}_0 + \boldsymbol{\theta} x \end{bmatrix} \\ \mathbf{h}' &= \mathbf{a} \begin{bmatrix} \boldsymbol{\psi}_0 + \boldsymbol{\Psi} \mathbf{h} \end{bmatrix} \\ \mathbf{a}' &= \phi'_0 + \boldsymbol{\phi}' \mathbf{h}', \end{aligned}
$$

• More generally, a *K*-layer network: The gorloruly, a II layor riouvorm. \boldsymbol{V} leaven of the input. • More generally, a K -layer networ

$$
h_1 = a[\beta_0 + \Omega_0 x]
$$

\n
$$
h_2 = a[\beta_1 + \Omega_1 h_1]
$$

\n
$$
h_3 = a[\beta_2 + \Omega_2 h_2]
$$

\n
$$
\vdots
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

\n
$$
h_K = a[\beta_{K-1} + \Omega_{K-1} h_{K-1}]
$$

 $y = \beta_K + \Omega_K h_K$.

Hyperparameters:

$$
K, D_1, D_2, \ldots, D_K
$$

 $\mathbf{a}[\mathbf{\beta}_2 + \mathbf{\Omega}_2 \mathbf{h}_2]$ **Parameters:** biases and weights **CLCIS.** DIASES AND WEIGHTS

$$
\pmb{\phi} = \{\pmb{\beta}_k, \pmb{\Omega}_k\}_{k=0}^K.
$$

 α is α **k** α **k** α **k** α **k** α **l n** α **k what are the sizes of the biases &** If the *^kth* layer has *^D^k* hidden units, then the bias vector ^β*^k*−¹ will be of size *^Dk*. *What are the sizes of the biases &*
weights in terms of the hyperparameters? Deep networks size *D*¹ × *Dⁱ* where *Dⁱ* is the size of the input. The last weight matrix Ω*^K* is *D^o* × *DK*,

- **Universal approximation theorem:** deep NNs can approximate any continuous function arbitrarily closely given suficient capacity.
	- We can reproduce a shallow network if all but one layer is the identity function. Since we showed that a shallow NN can approximate any continuous function, deep NNs also work.
- **More expressive** (more linear regions per parameter):
	- A shallow NN with 1 input, 1 output, $D > 2$ hidden units can create up to $D + 1$ linear regions using $3D + 1$ parameters.
	- A deep NN with 1 input, 1 output, $D > 2$ hidden units can create up to $(D + 1)^K$ linear regions using $3D + 1 + (K - 1)D(D + 1)$ parameters (more next page).

This exponential growth in linear regions is what makes deep NN more expressive.

- The counting of parameters for shallow NNs goes as follows:
	- There are D hidden units, each has two parameters (bias, weight). The output layer has D weights and one bias. # parameter=2D+D+1.
- The counting of parameters for deep NNs goes as follows:
	- There are *D* weights between the input and the first hidden layer, $K - 1$ lots of $D \times D$ inputs between adjacent hidden layers, and D weights between the last hidden layer and the output. There are D biases at each of the K hidden layers and 1 bias for the output. This gives $D + (K - 1)D^2 + D + KD + 1 =$ $3D + (K - 1)D^2 + (K - 1)D + 1$ parameters.

Shallow vs Deep Figure 4.2 Composing neural networks with a 2D input. a) The frst network a scalar output *y*. This is passed into a second network with two hidden units to . B) The fraction consisting of seven linear produces a function consisting of seven linear production consisting of seven linear production \mathbf{r}_i

• Deep NNs create much more linear regions for a fixed parameter budget, but they contain **complex dependence and symmetries**. the six non-fat regions for a fixed into the final into the fractions into the f

- The greater number of regions is an advantage if: **Figure 1.3** Deep networks in a proportional interest in a proportion of the multiple interests in a proportion σ is the input space back on top it σ
	- 1. there are similar symmetries in the function to approximate;
	- 2. the input \rightarrow output map is a composition of simpler functions.
- **Depth efficiency** refers to the phenomenon that a shallow NN needs exponentially more hidden units to achieve an equivalent approximation to that of a deep NN.

• **Large, structured inputs:** We have discussed fully connected networks where every element of each layer contributes to every element of the subsequent one.

- not practical for large structured inputs like images $\sim 10^6$ pixels
	- no point in independently learning to recognize the same object at every position in the image.

• CNN (which we will discuss later): process local image regions in parallel and integrate information from increasing large regions. Difficult to do this local-to-global processing with a single layer.

• Training and generalization: It is easier to train moderately deep networks than to train shallow ones.

- Deep NNs also seem to generalize to new data better than shallow ones. datasets (including CIFAR-100 and MNIST) almost perfectly with very large batches of the control of th
- Empirically, one finds best results for most tasks using networks with tens to hundreds of layers. MNIST-1D examples with randomized labels using full-batch (i.e., non-stochastic) gra-