Lecture 14: Neural Networks

Nisha Chandramoorthy

October 23, 2023

Last time

- A hypothesis class restricted to some C *shatters* C if any binary function on C is in the class

Last time

- A hypothesis class restricted to some C *shatters* C if any binary function on C is in the class
- VC dimension, $\mathrm{VCdim}(\mathcal{H})$: maximal size of $C \subseteq \mathcal{X}$ that is shattered by $\mathcal{H}$.

Last time

- A hypothesis class restricted to some C *shatters* C if any binary function on C is in the class
- VC dimension, $\mathrm{VCdim}(\mathcal{H})$: maximal size of $C \subseteq \mathcal{X}$ that is shattered by $\mathcal{H}$.
- Eg 1. VCdim of threshold functions on $\mathbb{R}$ is 1.

Last time

- A hypothesis class restricted to some *C* *shatters* *C* if any binary function on *C* is in the class
- VC dimension, $\mathrm{VCdim}(\mathcal{H})$: maximal size of $C \subseteq \mathcal{X}$ that is shattered by $\mathcal{H}$.
- Eg 1. VCdim of threshold functions on $\mathbb{R}$ is 1.
- Eg 2: VCdim of indicator functions on intervals of $\mathbb{R}$ is 2.

Last time

- A hypothesis class restricted to some C *shatters* C if any binary function on C is in the class
- VC dimension, $\mathrm{VCdim}(\mathcal{H})$: maximal size of $C \subseteq \mathcal{X}$ that is shattered by $\mathcal{H}$.
- Eg 1. VCdim of threshold functions on $\mathbb{R}$ is 1.
- Eg 2: VCdim of indicator functions on intervals of $\mathbb{R}$ is 2.
- Eg 3: VCdim of a finite class $\mathcal{H} \leqslant \log_2 |\mathcal{H}|$

Generalization bounds based on VC dimension

- $\mathcal{H} = \{h_\theta(x) = \sin(\theta x) : \theta \in \mathbb{R}\}$.

Generalization bounds based on VC dimension

- $\mathcal{H} = \{h_\theta(x) = \sin(\theta x) : \theta \in \mathbb{R}\}$.
- VCdim is ∞

Generalization bounds based on VC dimension

- $\mathcal{H} = \{h_\theta(x) = \sin(\theta x) : \theta \in \mathbb{R}\}$.
- VCdim is ∞
- Binary classification generalization for 0-1 loss over class $\mathcal{H}$ with VCdim = d: there exist constants $C_1, C_2 > 0$ such that

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leqslant m_{\mathcal{H}}(\epsilon, \delta) \leqslant C_2 \frac{d + \log(1/\delta)}{\epsilon}$$

(partial) History - trace back from transformers (source:Wikipedia)

- Transformer architecture: 2017, Google Brain [Vaswani et al]
- Deep learning, unsupervised learning 2010s (e.g., GANs 2014)...
- ImageNet: 2009, Fei Fei Li
- Long-short term memory (LSTM) architecture: 1997, [Hochreiter and Schmidhuber]
- Convolutional NNs: (inspired from) 1979 work by [Fukushima]; Recurrent neural networks: 1982 [Hopfield]
- ...
- Automatic Differentiation: 1970 [Linnainmaa]
- ...
- First neural networks: 1950s [Minsky and others]

Fully connected Neural Networks

- Neuron: input $\sum_j w_j h_j$; output $\sigma(\sum_j w_j h_j)$
- Organized into layers of depth l and width n
- Graph: V, E, σ, w; weight function.

Fully connected Neural Networks

- Neuron: input $\sum_j w_j h_j$; output $\sigma(\sum_j w_j h_j)$
- Organized into layers of depth l and width n
- Graph: V, E, σ, w; weight function.
- VC dim of $\mathcal{H}_{V,E,\mathrm{sign}} \leqslant C|E|\log|E|$

Fully connected Neural Networks

- Neuron: input $\sum_j w_j h_j$; output $\sigma(\sum_j w_j h_j)$
- Organized into layers of depth l and width n
- Graph: V, E, σ, w; weight function.
- VC dim of $\mathcal{H}_{V,E,\mathrm{sign}} \leqslant C|E|\log|E|$
  - Proof: Growth function $\tau_{\mathcal{H}}(m) = \max_{C,|C|=m} |\mathcal{H}|_C|$

Fully connected Neural Networks

- Neuron: input $\sum_j w_j h_j$; output $\sigma(\sum_j w_j h_j)$
- Organized into layers of depth l and width n
- Graph: V, E, σ, w; weight function.
- VC dim of $\mathcal{H}_{V,E,\mathrm{sign}} \leqslant C|E|\log|E|$
    - Proof: Growth function $\tau_{\mathcal{H}}(m) = \max_{C,|C|=m} |\mathcal{H}|_C|$ Fact 1: Let $\mathcal{H} = \mathcal{H}_l \circ \cdots \circ \mathcal{H}_1$. Then, $\tau_{\mathcal{H}}(m) \leqslant \prod_{t=1}^{l} \tau_{\mathcal{H}_t}(m)$.

Fully connected Neural Networks

- Neuron: input $\sum_j w_j h_j$; output $\sigma(\sum_j w_j h_j)$
- Organized into layers of depth l and width n
- Graph: V, E, σ, w; weight function.
- VC dim of $\mathcal{H}_{V,E,\mathrm{sign}} \leqslant C|E|\log|E|$
  - Proof: Growth function $\tau_{\mathcal{H}}(m) = \max_{C,|C|=m} |\mathcal{H}|_C|$ Fact 1: Let $\mathcal{H} = \mathcal{H}_l \circ \cdots \circ \mathcal{H}_1$. Then, $\tau_{\mathcal{H}}(m) \leqslant \prod_{t=1}^{l} \tau_{\mathcal{H}_t}(m)$.
  - Fact 2: Let $\mathcal{H} = \mathcal{H}^{(1)} \cdots \circ \mathcal{H}^{(n)}$. Then, $\tau_{\mathcal{H}}(m) \leqslant \prod_{t=1}^{l} \tau_{\mathcal{H}^{(t)}}(m)$.

Fully connected Neural Networks

- Neuron: input $\sum_j w_j h_j$; output $\sigma(\sum_j w_j h_j)$
- Organized into layers of depth l and width n
- Graph: V, E, σ, w; weight function.
- VC dim of $\mathcal{H}_{V,E,\mathrm{sign}} \leqslant C|E| \log |E|$
  - Proof: Growth function $\tau_{\mathcal{H}}(m) = \max_{C,|C|=m} |\mathcal{H}|_C|$ Fact 1: Let $\mathcal{H} = \mathcal{H}_l \circ \cdots \circ \mathcal{H}_1$. Then, $\tau_{\mathcal{H}}(m) \leqslant \prod_{t=1}^{l} \tau_{\mathcal{H}_t}(m)$.
  - Fact 2: Let $\mathcal{H} = \mathcal{H}^{(1)} \cdots \circ \mathcal{H}^{(n)}$. Then, $\tau_{\mathcal{H}}(m) \leqslant \prod_{t=1}^{l} \tau_{\mathcal{H}^{(t)}}(m)$.
  - Fact 3: Sauer's Lemma: $\tau_{\mathcal{H}}(m) = (em/d)^d$, where $d \geqslant$ VCdim $(\mathcal{H})$

Training

SGD update step: $w_{t+1} = w_t - \eta \tilde{\nabla} \hat{R}_S(w_t)$

- $\tilde{\nabla} \hat{R}_S(w_t)$ is an unbiased estimate of $\nabla \hat{R}_S(w_t)$
- Convergence for convex $\hat{R}_S$ and η small enough.
- Gradients implemented using backpropagation algorithm

Universal approximation theorems

Theorem [Park et al 2020, ICLR] (Informal) For $f \in L^p(\mathbb{R}^n, \mathbb{R}^m)$, and any $\epsilon > 0$, there exists a fully connected ReLU network F of width exactly $d = \max\{n+1, m\}$ such that $\|f - F\|_p^p < \epsilon$.

Kolmogorov-Arnold-Sprecher representation theorem: Any continuous multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ can be written as

$$f(x) = \sum_{i=0}^{2n} \Phi(\sum_{j=1}^{n} w_j \sigma(x_i + \eta i) + i),$$

where $\sigma : [0, 1] \to [0, 1]$.

Convolutional Neural Networks (source: cs231n.stanford.edu)

- Suitable for image recognition. Won the 2012 ImageNet competition and subsequent ones.
- Three types of layers: convolutional, FC, pooling
- Convolutional layer: accepts a volume of size $W_1 \times H_1 \times D_1$ and outputs a volume of size $W_2 \times H_2 \times D_2$ where $W_2 = (W_1 - F + 2P)/S + 1$ and $H_2 = (H_1 - F + 2P)/S + 1$ and $D_2 = K$.
- K is number of filters, F is filter size, S is stride, P is padding.
- Pooling layer: downsamples along width and height, and optionally along depth.
- FC layer: computes class scores, resulting in volume of size $1 \times 1 \times K$.

→ Theoretical & algorithmic aspects of NNs

→ ML engineering may be minimal

→ Working knowledge

<u>Plan</u> to <u>study NNs</u> (models + theory + algorithms)

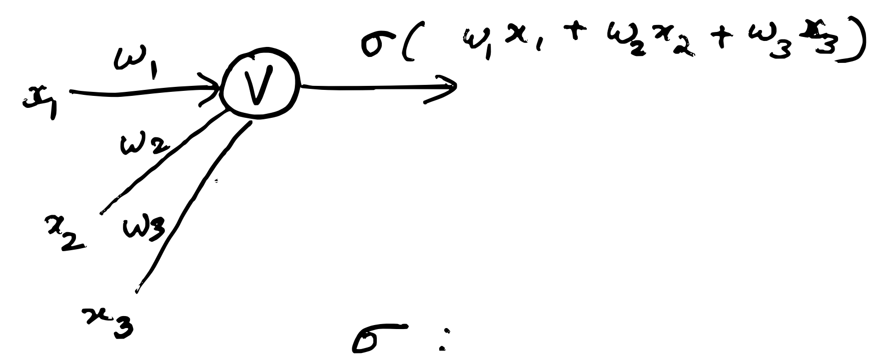
→ NN architectures / models: FC, Convolutional, ResNet, VAEs (dimension reduction) Bayesian perspective, GANs (Generative Adversarial Networks)

- LSTM ⇝ transformers (core of LLMs)
- RNNs (recurrent neural networks)

→ Approximation theory

→ Generalization (Sample complexity)

→ SGM variants
→ used for non-convex optimization



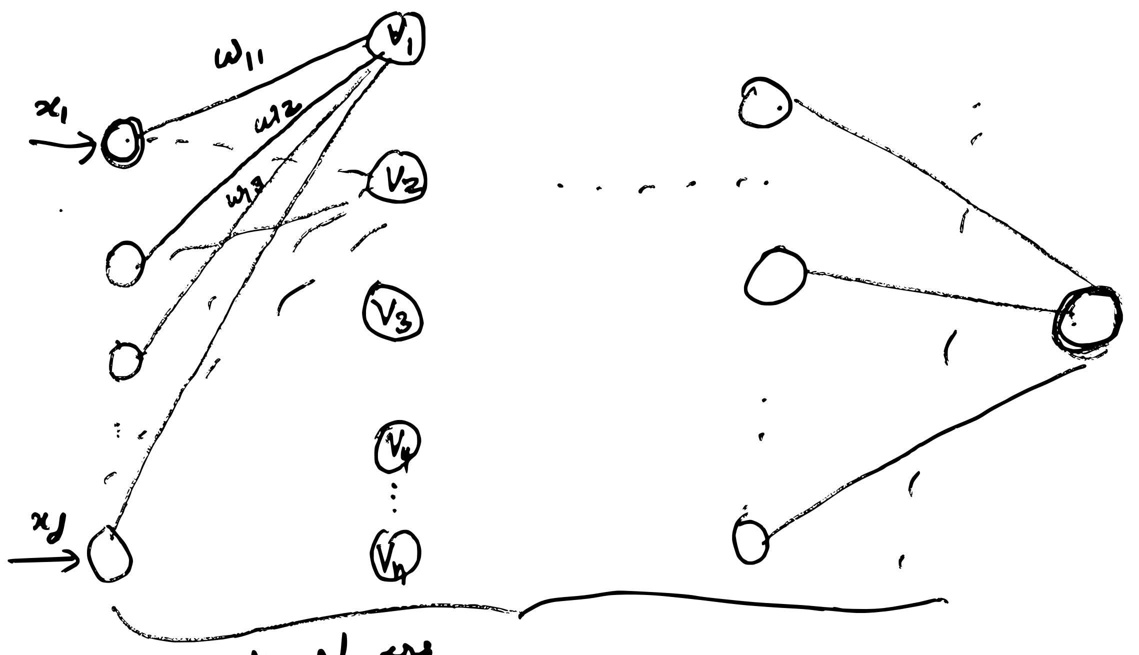
σ :

ReLU $\sigma(x) = \max\{0, x\}$

Sigmoid $\sigma(x) = \dfrac{1}{1+e^{-x}}$

sgn $\sigma(x) = \begin{cases} 1 & x > 0 \\ -1 & x \le 0 \end{cases}$

n = width of layer



depth: # of layers

$w_l \in \mathbb{R}^{n_{l-1} \times n_l}$

dimension of input layer = d
$(x \in \mathbb{R}^d)$

> 1 hidden layer → DNN

Graphical representation of DNNs.

→ V, E, σ, w

V: neuron, E: edges, σ: activation function

$$V = \bigcup_{l=1}^{depth} V_l$$

$$w(E_{ij}) \in \mathbb{R}$$

$$\rightarrow \mathcal{H}_{V,E,\sigma} = \mathop{\circ}_{l=1}^{depth} \mathcal{H}_{V_l, E_l, \sigma}$$

$$\mathcal{H}_{V_l, E_l, \sigma} = \mathop{\times}_{i=1}^{n_l} \mathcal{H}_{V_{li}, E_{li}, \sigma}$$

n_l : width of layer l

-

-

→ Midterm - 19^{th}

70 minutes

Neural Networks

Last time: → Feedforward

→ FF in Pytorch

→ Theory: UA

→ Optimization using SGD

Today:

→ Template for training, testing

→ Convolutional NN ✓

→ Algorithm: backpropagation ✓

→ 6.2, 7.12 ("Questions to focus on" + Lecture)

Ben-David Ch 20

Where do backprop?

SGD update

$$w_{t+1} = w_t - \eta_t \hat{\nabla}\hat{R}_S(w_t)$$

$\eta_t \sim O\left(\frac{1}{\sqrt{t}}\right)$ for Convergence convex functions

$$\hat{R}_S(w) = \frac{1}{m} \sum_{(x,y) \in S} \ell(y, h(x,w))$$

$$\nabla \hat{R}_S(w) = \frac{1}{m} \sum_{(x,y) \in S} \nabla_w \ell(y, h(x,w))$$

(MSE)

$$= \frac{1}{m} \sum_{(x,y) \in S} -2(y - h(x,w)) \nabla_w h(x,w)$$

Ex: $h(x, w) = f_{2,w} \circ \sigma \circ f_{1,w}(x)$

model(x)

1. $f_{1,w}(x)$ ✓
2. $\sigma(f_{1,w}(x))$ ✓
3. $f_{2,w}(\sigma(f_{1,w}(x)))$ ✓

To calculate "Adjoint"; use information from forward pass efficiently

Program	Adjoint
function $f(x, a)$	Goal: $\frac{dg}{da}$
$f(x,a) = a \sin x$	"$\frac{\partial f(x_i,a)}{\partial x_i}$" $= a \cos x$
return $f(x,a)$	"$\frac{\partial f(x_i,a)}{\partial a}$" $= \sin x$
$g(x) = \frac{1}{n} \sum_{i=1}^{n} f(x_i, a)$	✓ $\frac{dg}{dg} = 1$
	✓ $\frac{\partial g}{\partial f(x_i,a)} = \frac{1}{n}$

Adjoint: → calculating derivatives in reverse

→ when (param dim) > dim(g)
(a)

$$\frac{\partial g}{\partial f(x_i,a)} = \frac{\partial g}{\partial g} \cdot \frac{\partial g}{\partial f(x_i,a)} \frac{\partial f(x_i,a)}{\partial a}$$

$$\frac{\partial g}{\partial a} = \frac{\partial g}{\partial g} \frac{\partial g}{\partial f(x_1,a)} \frac{\partial f(x_1,a)}{\partial a} + \cdots + \frac{\partial g}{\partial g} \frac{\partial g}{\partial f(x_n,a)} \frac{\partial f(x_n,a)}{\partial a}$$

$$= \boxed{}$$

"Convolutional Layer" adjoint = Convolution

6.2

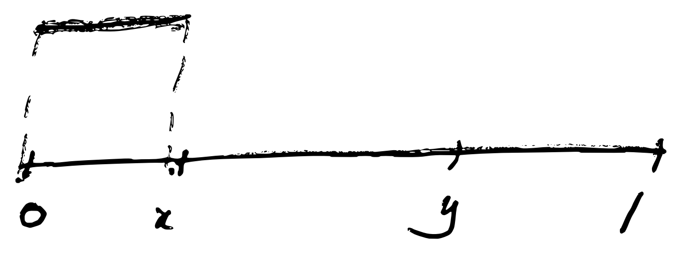
h) $k(x,y) = \min(x,y) - xy$

$[0,1] \times [0,1]$ is PDS

$\rightarrow k(x,y) = k(y,x)$

$\rightarrow I_1(x,y) = \int_0^1 1_{t\in[0,x]} \, 1_{t\in[0,y]} \, dt$

$I_2(x,y) = \int_0^1 1_{t\in[x,1]} \, 1_{t\in[y,1]} \, dt$



$I_1(x,y) = \min(x,y)$

$I_2(x,y) = 1 - \max(x,y)$

$I_1(x,y) = \langle 1_{[0,x]} \, , \, 1_{(0,y]} \rangle$

$k(x,y) = I_1(x,y) \times I_2(x,y)$

$= \min(x,y) - \min(x,y) \times \max(x,y)$

$= \min(x,y) - xy$

Closure properties

$\rightarrow$ PDS kernels are closed under $+$, $*$, $x \rightarrow \sum_n a_n x^n$, $a_n \geq 0$

(k) $k(x,y) = e^{\left(\frac{\sum_{i=1}^{N} \min(|x_i|, |y_i|)}{\sigma^2}\right)}$

- $k_1(x,y) = \min(|x|, |y|)$ is PDS.
- $k_2(\underset{\mathbb{R}^N}{x, y}) = \sum_{i=1}^{N} \min(|x_i|, |y_i|)$ is PDS (closure prop under addition)
- $k(x,y) = e^{k_2(x,y)/\sigma^2}$

$= \sum_n \underbrace{\left(\frac{k_2(x,y)}{\sigma^2}\right)^n \frac{1}{n!}}$

(closure property under series expansion).