Ch 20 Neural Networks: approximation

Friday, March 27, 2020 3:05 PM

Neural Networks or "Artificial Neural Network" (ANN)

Originally inspired by brain, but now we don't need the physical link
i.e., ≠ Hodgkin-Huxley models ...

We'll focus on feed-forward ANN, and only cover basic theory

Not covering pooling, batch-normalization, dropout, recurrent ANN, auto-encoders, generative-adversarial networks (GAN), ...

Feed-forward ANN is described by its architecture (a directed acyclic (DAG) graph w/ nodes V, edges E)
and an activation function σ (node = neuron)
and the edge weights w. (hyperparameters (train via grad. student descent))
(parameters (train via SGD))

$w: E \to \mathbb{R}$, $w \in \mathcal{H}_{V,E,\sigma}$

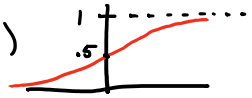
Assume a layered structure, $V = \bigcup_{t=0}^{T} V_t$, $|V_0| = n+1$ (for constant) is input layer

$\sigma: \mathbb{R} \to \mathbb{R}$, sometimes write $\sigma(\vec{a})$ to mean $(\sigma(\vec{a}))_i = \sigma(a_i)$, i.e., apply coordinate-wise
typical examples:

Ex 1a: $\sigma(a) = \text{sign}(a)$ "sign" fcn

Ex 1b: $\sigma(a) = \mathbb{1}_{[a>0]}$ "threshold" fcn

Ex 2: $\sigma(a) = \frac{1}{1+e^{-a}}$ "sigmoid" fcn (same as in logistic regression)



Ex 3: $\sigma(a) = [a]_+ = \max(0, a)$ "rectified linear unit" ReLU (not discussed in our book but very popular)

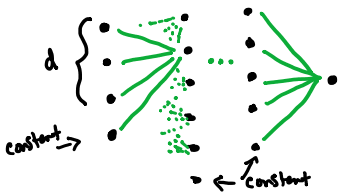
many more variants, but similar theoretical properties
(but nonlinearity usually essential)

to calculate the output, let V_{ti} be i^{th} neuron in t^{th} layer,
which has input a_{ti} and gives output $o_{ti} = \begin{cases} \sigma(a_{ti}) & i \neq 0 \\ 1 & i = 0 \end{cases}$ (special "constant" neuron)

where input is $a_{ti} = \sum_{r:(V_{t-1,r}, V_{ti}) \in E} w((V_{t-1,r}, V_{ti})) \cdot o_{t-1,r}$

(note: all input/output really depend on original input vector x)

Much simpler w/ a picture



T = #layers (excluding input), eg. 3

width = $\max_t |V_t|$, eg., 6

size = $|V|$

V_0 input | V_1, V_2 hidden layers (internal) | V_3 output

Different layers can have different sizes

output layer is 1 neuron for binary classification
but more for k-way classification (a major benefit of ANN is avoiding the ugly reductions of multiclass to binary via, eg., one-vs.-all or all-pairs, cf. ch 17)
or d for autoencoder, $x \in \mathbb{R}^d$

Fully connected means complete bipartite for adjacent layers
(vs a convolutional layer, for example)

Outline of our neural net unit:

I. Approximation Error

II. Estimation Error (generalization)

III. Optimization Error (new!)

Let $A(S)$ be output of neural net w/ training data S (like ERM)

$$L_D(A(S)) = \underbrace{\min_{h \in Hyp_{NN}} L_D(h)}_{\text{(I) Approx.}} + \underbrace{\min_{h \in Hyp_{NN}} \hat{L}_S(h) - \min_{h \in Hyp_{NN}} L_D(h)}_{\text{(II) Estimation}} + \underbrace{L_D(A(S)) - \hat{L}_S(A(S))}_{\text{also (II) Estimation}} + \underbrace{\hat{L}_S(A(S)) - \min_{h \in Hyp_{NN}} \hat{L}_S(h)}_{\text{(III) Optimization (=0 if ERM)}}$$

this is "classical". More "modern" approach incorporates training algorithm into analysis
(before, algorithm can only hurt you by giving large optimization error,
now, algo. can help by improving generalization (eg., stability analysis) and you don't have to include estimation error)

- Optimization Error hard to bound
- Estimation error of large ANN is huge if you do, eg, VC-dim analysis

"lack of theoretical understanding of ANN" is due to ↑ these pts.

Neural nets generalize better than expected, and the type of algorithm (ex SGD w/ dropout) makes a difference

MUCH new theory since ~2015: one of biggest open problems

Basically, theory predicts a finite sample complexity, but it's too large / conservative

History:

Ideas, algorithms, analysis 1980's
Geoff Hinton, Yann LeCun, Yoshua Bengio (Mike Mozer at CU)

but not too much practical success (not bad, but not clear winner)

until ~2012,
1. conv nets showed excellent performance
2. "deep" (many layers), enabled by more computational power
3. GPUs, better computers
4. tricks (dropout, ...)
5. success => more success (more researches, more tricks/insight ...)
6. software (tensorflow, caffe, pyTorch, etc.)

(I) The expressive power of ANN (ie., approximation error)

Lemma For approximation, more-is-better: expressive power is nondecreasing as we add more layers or depth.

proof intuitive. If we add depth, can always set weights $=0$

If we add layers, for $\sigma(a) = \mathbb{1}_{[a>0]}$, note $\sigma^2(a) = \sigma(a)$, so make weights the identity

Approximation theorems

"Approximate all $f: \mathbb{R}^n \to \mathbb{R}^n$" is ill-posed

1. need a metric (eg, L^2, ptwise)
2. metric restricts type of functions f

(eg., need $\int f^2(x)dx < \infty$, or need f defined ptwise)

Classical thm: Weierstrass Approx. thm, says $P([a,b])$ dense in $C([a,b], \|\cdot\|_\infty)$ (or Stone-W.)
↑ all polynomials

For binary classification, $f: \mathbb{R}^n \to \{\pm 1\}$, ε-approx. wrt $\|\cdot\|_\infty$ doesn't make sense, but exact representation also undesirable (since uncountable cardinality)

Better: exact representation, but for $f: \{\pm 1\}^n \to \{\pm 1\}$ (Boolean Fcn.)

(and if $f: \mathbb{R}^d \to \mathbb{R}$, represent $\mathbb{R}$ w/ 64-bits on computer, then take $n = 64d$)

Claim 20.1

$(\forall n \in \mathbb{N})$ $(\exists V, E$ of depth 2, width $2^n+1)$ s.t. $\mathcal{H}_{V,E,\sigma=\text{sign}}$ contains all Boolean fcn $f: \{\pm 1\}^n \to \{\pm 1\}$

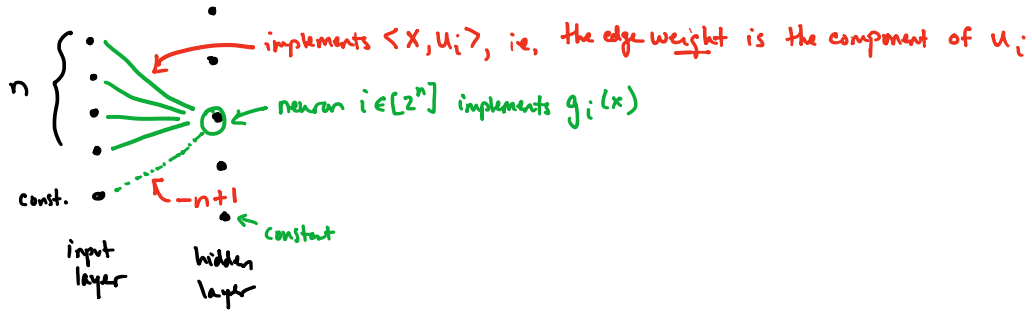
proof

Enumerate $\{\pm 1\}^n = \{u_i\}_{i \in [2^n]}$. Observe $\forall i$, $\forall x \in \{\pm 1\}^n$, $\langle x, u_i \rangle = \begin{cases} n & x = u_i \\ n' \le n-2 & x \neq u_i \end{cases}$

since $x \neq u_i \Rightarrow$ differ in at least one coordinate, so that "+1" is "−1", a difference of 2

for all $i \in [2^n]$, define

$$g_i(x) = \text{sign}(\langle x, u_i \rangle - n + 1) = \begin{cases} +1 & x = u_i \\ -1 & x \neq u_i \end{cases}$$



Now, let $f : \{\pm 1\}^n \to \{\pm 1\}$ be arbitrary. It is characterized by the inputs that give an output of $+1$. Let $\{u_i\}_{i \in F}$ be these inputs, $|F| = k \leq 2^n$

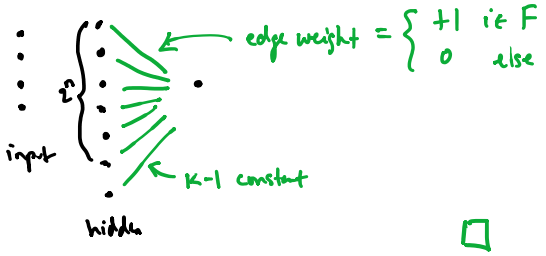
Observe $f(x) = \begin{cases} +1 & \text{if } x \in \bigcup_{i \in F} u_i \\ -1 & \text{else} \end{cases}$

$$= \text{sign}\left(\sum_{i \in F} g_i(x) + k - 1 \right)$$

Since $x \in \bigcup_{i \in F} u_i \Rightarrow \sum_{i \in F} g(x) = 1 - (k-1) = 2-k \Rightarrow 2-k+k-1 > 0$

$x \notin \bigcup_{i \in F} u_i \Rightarrow \sum_{i \in F} g(x) = -k \Rightarrow -k+k-1 < 0$

So for our last layer we implement



□

What about real-valued functions?

Example result

Exercise 20.1 Let $f : [-1,1]^n \to [-1,1]$ be ρ-Lipschitz. Then $(\forall \varepsilon > 0)$

$\exists$ ANN f_{NN} w/ σ = sigmoid s.t. $\| f - f_{NN} \|_\infty < \varepsilon$

proof sketch

wlog let f be ρ-Lipschitz wrt $\|\cdot\|_\infty$ norm on $\mathbb{R}^n$
(ie., adjust value of ρ if need be)

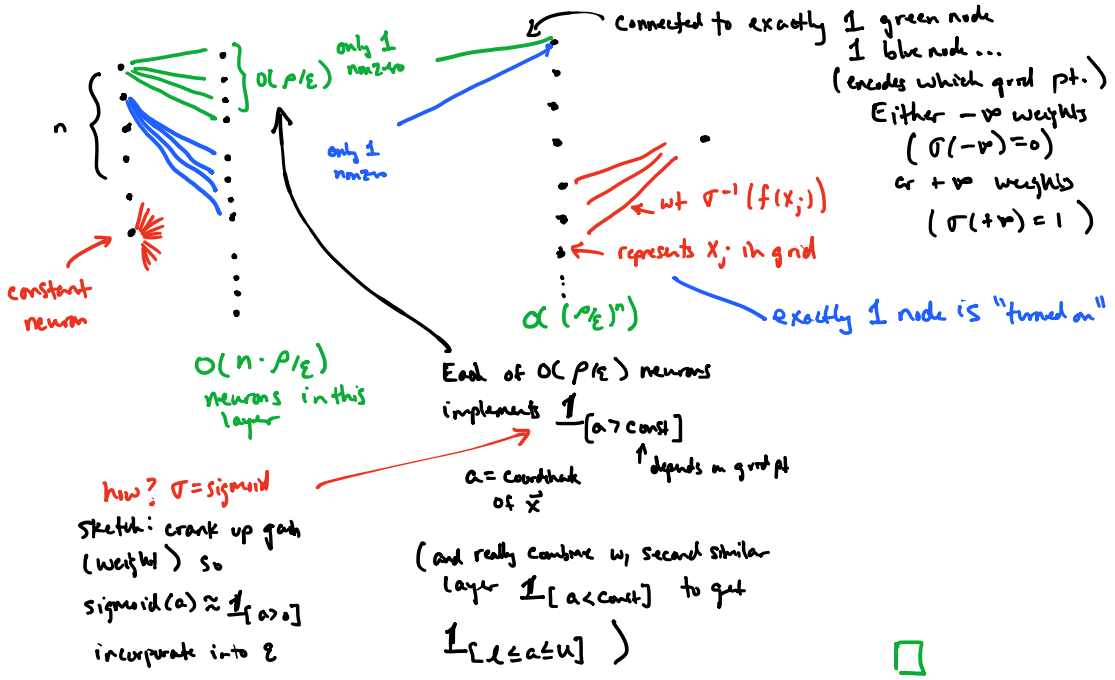
Make grid on $[-1,1]^n$, spacing of $O(\varepsilon/\rho)$ in each dimension

(So $O(\rho/\varepsilon)$ grid pts / dimension, or $O((\rho/\varepsilon)^n)$ grid pts overall)

hence $x, x' \in$ same grid cell

$\Rightarrow |f(x) - f(x')| \leq \rho \|x - x'\|_\infty \leq \varepsilon/2$

Now, given $x \in [-1,1]^n$, can we determine which grid cell it belongs to?



□

Back to Claim 20.1, the width is 2^n+1 (not good!)

Not just due to proof technique

Theorem 20.2 $(\forall n\in\mathbb{N})$ if $\mathcal{H}_{V,E,\sigma=\text{sign}}$ contains all Boolean fcn $f:\{0,1\}^n\to\{0,1\}$ (or $\{\pm1\}$, book's typo) then $|V|$ is exponential in n. Similar results if σ = sigmoid

Proof

let $\mathcal{H}_{V,E,\text{sign}}$ contain all such Boolean functions, thus it can shatter

$m=2^n$ vectors $\Rightarrow$ VCdim$(\mathcal{H}) \ge 2^n$

As we'll show later, VCdim$(\mathcal{H}) \le O(|E|\log(|E|))$
$\le O(|V|^3)$

So $2^n \le O(|V|^3) \Rightarrow$ $|V| \ge \Omega(2^{n/3})$ □

Similarly for $\mathbb{R}$-valued functions:

Thm 20.5 $\forall\varepsilon\in(0,1)$, let $\mathcal{H}_{V,E,\sigma=\text{sigmoid}}$ be large enough to approximate all 1-Lipschitz functions $f:[-1,1]^n\to[-1,1]$ to within ε. Then $|V|$ is exponential in n.

Related theorems in the literature

Zhou Lu et al. "The Expressive Power of Neural Networks: A View from the Width" (NIPS 2017)

Thm (density in $L^1(\mathbb{R}^n)$) Let $f \in L^1(\mathbb{R}^n)$ then $\forall \varepsilon > 0$ $\exists$ ANN $f_{NN}: \mathbb{R}^n \to \mathbb{R}$ (fully connected, ReLU activation, depth $4n+1$, width $n+4$) s.t. $\|f - f_{NN}\|_{L^1} < \varepsilon$

G. Cybenko. "Approximation by superpositions of a sigmoidal function" (1989). DOI: 10. 1007/BF02551274

Thm (density in $C(\Omega, \|\cdot\|_\infty)$) Let Ω be any compact subset of $\mathbb{R}^n$, then ANN w/ one hidden layer, arbitrary ~~depth~~ width, any continuous sigmoidal activation ($\lim_{t\to+\infty} \sigma(t) = 1$, $\lim_{t\to-\infty} \sigma(t) = 0$) are dense in $C(\Omega, \|\cdot\|_\infty)$

(Issue: f_{NN} w/ σ sigmoidal is bounded, but let σ at last layer be identity)

Dmitry Yarotsky. "Optimal approximation of continuous functions by very deep ReLU networks" (2018) arXiv: 1802.03620

Thm (density in $C^k(\Omega, \|\cdot\|_\infty)$, quantitative) ANN w/ σ=ReLU is dense in $C^k(\Omega, \|\cdot\|_\infty)$ (Ω compact in $\mathbb{R}^n$) and for error ε, need $O(\log(\frac{1}{\varepsilon}))$ layers and $O((\frac{1}{\varepsilon})^{K/n}\log(\frac{1}{\varepsilon}))$ ~~depth~~. $|E|$ weights

David Rolnick and Max Tegmark. "The power of deeper networks for expressing natural functions" (2017) arXiv: 1705.05502

Thm lower bounds For approximation of $\mathbb{R}^n$ function, ANN needs

width $\approx$ $\begin{cases} 2^n & 1 \text{ hidden layer} \\ n^{1/k} & k \text{ hidden layers} \quad (1 < k < \text{cutoff}) \\ n & \text{many hidden layers} \end{cases}$

Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. "Small ReLU networks are powerful memorizers: a tight analysis of memorization" (2018) arXiv: 1810.07770

Instead of learning functions, can talk about memorizing datasets
= good, from approx. perspective = bad, from generalization perspective

Thm (Shallow) A 3-layer ReLU network w/ $\Omega(\sqrt{m})$ hidden nodes can perfectly memorize a dataset w/ m points ($\Omega(\sqrt{m})$ is necessary and sufficient) (2 hidden layers of $\sqrt{m}$ each, or, product is m)

Back to our book

We saw some Boolean functions require an ANN of exponential size.
Can some be expressed in ANN of polynomial size? Yes.

Thm 20.3 Let $T: \mathbb{N} \to \mathbb{N}$. $\exists c, b \in \mathbb{R}_+$ s.t. $(\forall n \in \mathbb{N})$ $\exists (V,E)$ of size $|V| \le c \cdot T(n)^2 + b$ s.t. any function f which can be implemented on a Turing machine in runtime $T(n)$ or less is inside the class $\mathcal{H}_{V,E,\sigma=\text{sign}}$

No proof, but key idea is that an ANN w/ sign can implement the basics of a circuit (binary logic)
i.e., a set of universal gates, like {NAND}, {NOR}, {AND, NOT}

In particular, ANN can recreate
- Conjunction $\wedge$ (AND)
- disjunction $\vee$ (OR)
- negation $\neg$ (NOT)

Lemma 20.4

Let $\sigma = \text{sign}$, and let a neuron v have k incoming edges with values $o_i \in \{\pm 1\}$, $i \in [k]$
Then (allowing for a constant input neuron) we can adjust weights w_i s.t.

the output $\sigma\left(\sum_i w_i o_i + c\right) = \bigwedge_i o_i$

or $\bigvee_i o_i$

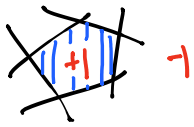
Proof $\bigwedge_{i \in [k]} o_i = \text{sign}\left(1 - k + \sum_{i=1}^{k} o_i\right)$ and $\bigvee_{i \in [k]} o_i = \text{sign}\left(k - 1 + \sum_i o_i\right)$

Geometric Intuition

$f: \mathbb{R}^2 \to \{\pm 1\}$ w/ $\sigma = \text{sign}$. What can we represent?

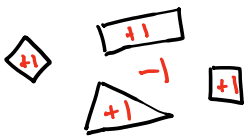
Depth 2 (1 input layer, 1 hidden layer, 1 output layer)

Every neuron in hidden layer is of form $\text{sign}(\langle w, x \rangle + b)$
So it's a halfspace predictor. Then the output can be a conjunction ($\wedge$) or disjunction ($\vee$) of these, so intersection / union of halfspaces.
(If $|V_1| = k+1$, (for constant) intersection of k halfspaces i.e., convex polytope w/ k faces (width $k+1$))



Depth 3

Take depth 2 and also do conjunctions / disjunctions
So... union of polytopes (or intersection... so nonconvex polytopes)



So you can see it's quite expressive compared to the classes we've studied before (half-spaces, SVM, decision stumps, decision trees)