EE6550 Machine Learning

Lecture Twelve – Deep Network I
Deep Feedforward Networks

Chung-Chin Lu

Department of Electrical Engineering

National Tsing Hua University

May 22, 2017

Neural Networks

- A neural network [a] can be described as a directed graph whose nodes correspond to neurons and edges correspond to links between them.
- Each neuron receives as input a weighted sum of the outputs of the neurons connected to its incoming edges.
- Feedforward network: a neural network whose underlying graph does not contain cycles.

[a] See Chapter 20 in S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithm*. New York: Cambridge University Press, 2014.

The Contents of This Lecture

- Feedforward neural networks
- VC dimensions of hypothesis sets of neural network predictors
- Expressive power of neural networks
- SGD and backpropagation

Feedforward Neural Networks

- A feedforward neural network is described by a directed acyclic graph, $G = (V, E)$, and a weight function over the edges, $w : E \to \mathbb{R}$.
  - Nodes of the graph correspond to neurons.
  - Each (directed) edge in the graph links the output of some neuron to the input of another neuron.
- Each single neuron is modeled as a simple scalar function, $\sigma : \mathbb{R} \to \mathbb{R}$, which is called the activation function of the neuron.
  - The activation function of a neuron describes the input-output relation of the neuron.
  - There are three general types of activation functions considered.

* $\sigma(a) = \mathrm{sgn}(a)$: the sign function;
* $\sigma(a) = 1_{[a>0]}$: the threshold function;
* $\sigma(a) = 1/(1+\exp(-a))$: the sigmoid function, which is a smooth approximation to the threshold function.

- The input of a neuron is obtained by taking a weighted sum of the outputs of all the neurons connected to it, where the weighting is according to the weight function w.

Layered Feedforward Neural Networks

- $V = \cup_{t=0}^{T} V_t$: a partition of the node set V.
  - Each V_t forms a layer.
- $E \subseteq \cup_{t=1}^{T}(V_{t-1} \times V_t)$: every edge in E connects some node in V_{t-1} to some node in V_t for some $t \in [1, T]$.
- V_0: the input layer, which consists of $n + 1$ neurons, where n is the dimensionality of the input space.
  - For every $i \in [1, n]$, the output of neuron i in V_0 is simply x_i.
  - The last neuron in V_0 is the constant neuron, which always outputs 1.
- $v_{t,i}$: the ith neuron of the tth layer.

- $o_{t,i}(\boldsymbol{x})$: the output of $v_{t,i}$ when the network is fed with the input vector $\boldsymbol{x}$.
  - For $i \in [1, n]$, we have $o_{0,i}(\boldsymbol{x}) = x_i$ and for $i = n + 1$, we have $o_{0,n+1}(\boldsymbol{x}) = 1$.
- Calculation in a layer by layer manner: suppose we have calculated the outputs of the neurons at layer t. Then, we can calculate the outputs of the neurons at layer $t + 1$ as follows.
  - Fix some $v_{t+1,j} \in V_t$. Let $a_{t+1,j}(\boldsymbol{x})$ denote the total input to $v_{t+1,j}$ when the network is fed with the input vector $\boldsymbol{x}$,

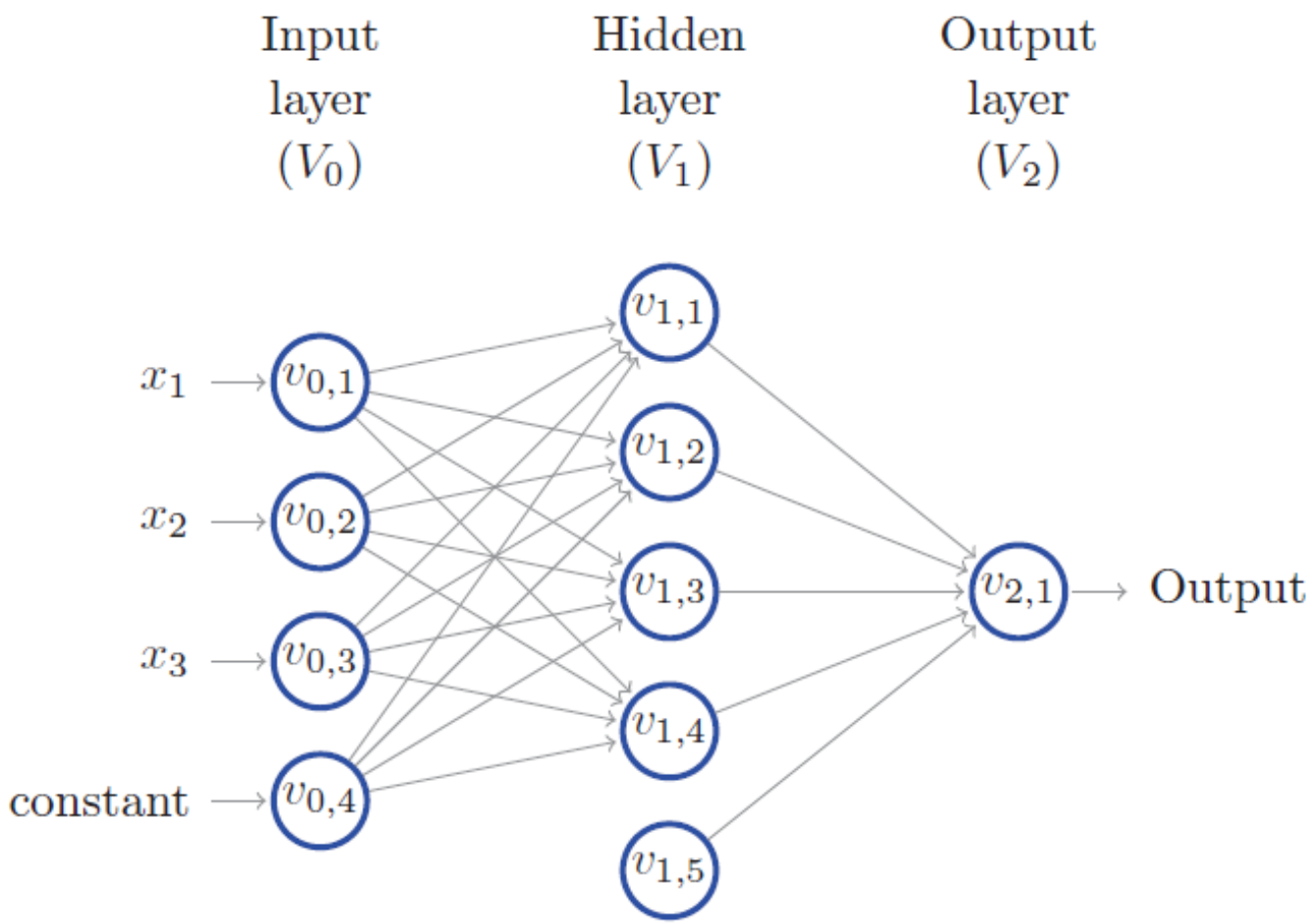
$$a_{t+1,j}(\boldsymbol{x}) = \sum_{r:(v_{t,r},v_{t+1,j})\in E} w((v_{t,r}, v_{t+1,j}))o_{t,r}(\boldsymbol{x}).$$

  - The output $o_{t+1,j}(\boldsymbol{x})$ of the neuron $v_{t+1,j}$ is

$$o_{t+1,j}(\boldsymbol{x}) = \sigma(a_{t+1,j}(\boldsymbol{x})).$$

- Layers $V_1, \ldots, V_{T-1}$: the $T - 1$ hidden layers.

- V_T: the output layer.
  - In simple prediction problems, the output layer contains a single neuron whose output is the output of the network.
- $|V| = \sum_{t=0}^{T} |V_t|$: the size of the network.
- T: the number of layers in the network (excluding V_0), called the depth of the network.
- $\max_{0 \le t \le T} |V_t|$: the width of the network.



A layered feedforward neural network of depth 2, size 10, and width 5.

The Contents of This Lecture

- Feedforward neural networks
- VC dimensions of hypothesis sets of neural network predictors
- Expressive power of neural networks
- SGD and backpropagation

Hypothesis Sets of Neural Network Predictors

- A neural network can be specified by a quadruple (V, E, σ, w).
- $h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$: the hypothesis associated with a neural network specified by (V, E, σ, w).
- $\mathcal{H}_{V,E,\sigma} = \{h_{V,E,\sigma,w} \mid w \text{ is a mapping from } E \text{ to } \mathbb{R} \}$: the hypothesis set of neural network predictors by fixing the directed graph $G = (V, E)$ as well as the activation function σ but letting the weight function w arbitrary.
  - The triplet (V, E, σ) is often called the architecture of the hypothesis set of neural network predictors.
  - The parameters specifying a hypothesis in the hypothesis set $\mathcal{H}_{V,E,\sigma}$ are the weights over the edges of the network.

VC-Dimension of $\mathcal{H}_{V,E,\mathbf{sgn}}$ with Single Output Neuron

Theorem 1: The VC-Dimension of $\mathcal{H}_{V,E,\mathrm{sgn}}$ with single output neuron is $O(|E| \log |E|)$.

VC-Dimension of $\mathcal{H}_{V,E,\sigma}$ with Single Output Neuron

Theorem 2: The VC-Dimension of $\mathcal{H}_{V,E,\sigma}$ with single output neuron, where σ is the sigmoid function, is both $\Omega(|E|^2)$ and $O(|V|^2|E|^2)$.

The Contents of This Lecture

- Feedforward neural networks
- VC dimensions of hypothesis sets of neural network predictors
- Expressive power of neural networks
- SGD and backpropagation

Expressive Power of Neural Networks

- Fixing an architecture, (V, E, σ), what functions hypotheses in $\mathcal{H}_{V,E,\sigma}$ can implement ?
- Which type of Boolean functions (i.e., functions from $\{\pm 1\}^n$ to $\{\pm 1\}$) can be implemented by $H_{V,E,\text{sgn}}$?
  - Since real numbers are stored using b bits in every digital computer, whenever we calculate a function $f : \mathbb{R}^n \to \mathbb{R}$ on such a computer we in fact calculate a function $g : \{\pm 1\}^{nb} \to \{\pm 1\}^b$.
  - Therefore, studying which Boolean functions can be implemented by $H_{V,E,\text{sgn}}$ can tell us which functions can be implemented on a computer that stores real numbers using b bits.

Neural Networks Are Universal for Boolean Functions

Proposition 1: For every n, there exists a graph (V, E) of depth 2, such that $H_{V,E,\mathrm{sgn}}$ contains all functions from $\{\pm 1\}^n$ to $\{\pm 1\}$.

Proof.

- Construct a layered graph with node set $V = V_0 \cup V_1 \cup V_2$, where $|V_0| = n + 1$, $|V_1| = 2^n + 1$, and $|V_2| = 1$, and edge set E consisting of all possible edges between adjacent layers.
- $f : \{\pm 1\}^n \to \{\pm 1\}$: an arbitrary Boolean function.
- We need to show that $f = h_{V,E,\mathrm{sgn},w}$ by specifying a weight function w.
- $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$: all vectors in $\{\pm 1\}^n$ on which f outputs 1.
- Observe that for every $i \in [1, k]$ and every $\boldsymbol{x} \in \{\pm 1\}^n$, if $\boldsymbol{x} \neq \boldsymbol{u}_i$, then $\boldsymbol{x} \cdot \boldsymbol{u}_i \leq n - 2$ and if $\boldsymbol{x} = \boldsymbol{u}_i$, then $\boldsymbol{x} \cdot \boldsymbol{u}_i = n$.

- The function $g_i(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{x} \cdot \boldsymbol{u}_i - n + 1)$ equals 1 if and only if $\boldsymbol{x} = \boldsymbol{u}_i$.
- We can adapt the weights between V_0 and V_1 so that for every $i \in [1, k]$, the neuron $v_{1,i}$ implements the function $g_i(\boldsymbol{x})$.
  - $w((v_{0,j}, v_{1,i})) = u_{i,j}$ for all $j \in [1, n]$ and $w((v_{0,n+1}, v_{1,i})) = -n + 1$.
  - The total input $a_{1,i}(\boldsymbol{x})$ of $v_{1,i}$ is $\boldsymbol{x} \cdot \boldsymbol{u}_i - n + 1$.
  - $o_{1,i}(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{x} \cdot \boldsymbol{u}_i - n + 1) = g_i(\boldsymbol{x})$.
- Observe that $f(\boldsymbol{x})$ is the disjunction of the functions $g_i(\boldsymbol{x})$, and therefore can be written as

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} g_i(\boldsymbol{x}) + k - 1\right)$$

  - $w((v_{1,i}, v_{2,1})) = 1 \ \forall \ i \in [1, k]$ and $w((v_{1,2^n+1}, v_{2,1})) = k - 1$.

This concludes the proof. □

Exponential Size Is Inevitable

Theorem 3: For every n, let $s(n)$ be the minimal integer such that there exists a graph (V, E) with $|V| = s(n)$ such that the hypothesis set $H_{V,E,\text{sgn}}$ contains all the functions from $\{\pm 1\}^n$ to $\{\pm 1\}$. (This implies that there are $n+1$ input neurons and a single output neuron.) Then, $s(n)$ is exponential in n. Similar results hold for $H_{V,E,\sigma}$ where σ is the sigmoid function.

Proof.

- Suppose that for some (V, E) with $|V_0| = n+1$ and $|V_T| = 1$, we have that $\mathcal{H}_{V,E,\text{sgn}}$ contains all functions from $\{\pm 1\}^n$ to $\{\pm 1\}$.
- Since $\mathcal{H}_{V,E,\text{sgn}}$ can shatter the set $\{\pm 1\}^n$ of 2^n vectors, the VC dimension of $\mathcal{H}_{V,E,\text{sgn}}$ is lower bounded by 2^n.
- We know that the VC dimension of $\mathcal{H}_{V,E,\text{sgn}}$ is $O(|E| \log |E|)$

and is upper bounded by $O(|V|^3)$. Thus we have

$$2^n \leq \text{VC-dim}(\mathcal{H}_{V,E,\text{sgn}}) \leq M|V|^3$$

for some constant $M > 0$.

- This implies that $|V|$ is $\Omega(2^{n/3})$.

This concludes the proof for the case of neural networks with the sign activation function. The proof for the sigmoid case is analogous. □

Remarks

- It is possible to derive a similar theorem for $H_{V,E,\sigma}$ with single output neuron for any activation function σ, as long as we restrict the weights so that it is possible to express every weight using a number of bits which is bounded by a universal constant.
- We can even consider hypothesis sets where different neurons can employ different activation functions, as long as the number of allowed activation functions is also finite.
- We will show that all Boolean functions that can be calculated in time $O(T(n))$ can also be expressed by a neural network of size $O(T(n)^2)$.

A Lemma of Logical Implementation

Lemma 1: Suppose that a neuron v, that implements the sign activation function, has k incoming edges, connecting it to neurons whose outputs are in $\{\pm 1\}$. Then, by adding one more edge, linking a constant neuron to v, and by adjusting the weights on the edges to v, the output of v can implement the conjunction or the disjunction of its inputs.

Proof.

- The conjunction function $f : \{\pm 1\}^n \to \{\pm 1\}$ is $f(\boldsymbol{x}) = \wedge_i x_i$, which can be implemented as $f(\boldsymbol{x}) = \text{sgn}(\sum_{i=1}^{k} x_i - k + 1)$.
- The disjunction function $f : \{\pm 1\}^n \to \{\pm 1\}$ is $f(\boldsymbol{x}) = \vee_i x_i$, which can be implemented as $f(\boldsymbol{x}) = \text{sgn}(\sum_{i=1}^{k} x_i + k - 1)$.

This completes the proof. □

Neural Networks of Polynomial Size

Theorem 4: Let $T : \mathbb{N} \to \mathbb{N}$ and for every n, let $\mathfrak{F}_n$ be the set of functions that can be implemented using a Turing machine using runtime of at most $T(n)$. Then, there exist constants $b, c > 0$ such that for every n, there is a graph (V_n, E_n) of size at most $cT(n)^2 + b$ such that $H_{V_n,E_n,\mathrm{sgn}}$ contains $\mathfrak{F}_n$.

The Contents of This Lecture

- Feedforward neural networks
- VC dimensions of hypothesis sets of neural network predictors
- Expressive power of neural networks
- SGD and backpropagation

Learning Neural Networks Is NP Hard

Theorem 5: Let $k \geq 3$. For every n, let (V, E) be a three-layered directed graph with $n+1$ input nodes, where one of them is the constant neuron, $k+1$ nodes at the (single) hidden layer, where one of them is the constant neuron, and a single output node. Then, it is NP hard to implement the ERM rule with respect to the hypothesis set $H_{V,E,\mathrm{sgn}}$.

Proof.

- The k-coloring problem can be reduced to the ERM problem with respect to $H_{V,E,\mathrm{sgn}}$.
- The k-coloring problem is known to be NP hard for all $k \geq 3$ [a].
- The ERM problem with respect to $H_{V,E,\mathrm{sgn}}$ is NP hard. □

[a] R.M. Karp, *Reducibility Among Combinatorial Problems*. Springer, 1972.

Neural Network Learning Problem

- (V, E, σ): the architecture of the hypothesis set $\mathcal{H}_{V,E,\sigma}$ of neural network predictors.
- $\boldsymbol{w} \in \mathbb{R}^{|E|}$: the weight vector which represents the weight function $w : E \to \mathbb{R}$.
- $n = |V_0|$: the number of input neurons.
- $k = |V_T|$: the number of output neurons.
- $\boldsymbol{h_w} : \mathbb{R}^n \to \mathbb{R}^k$: the vector function calculated by the neural network if the weight function is represented by $\boldsymbol{w}$.
- $\mathscr{I} = \mathbb{R}^n$: the input space, associated with a probability space $(\mathbb{R}^n, \mathcal{B}_n, P)$ where P is unknown.
- $\mathscr{Y} = \mathbb{R}^k$: the label space.

- $\boldsymbol{c} : \mathbb{R}^n \to \mathbb{R}^k$: an unknown target concept.
- $\Delta(\boldsymbol{h_w}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))$: the loss of predicting $\boldsymbol{h_w}(\boldsymbol{x})$ when the label is $\boldsymbol{c}(\boldsymbol{x})$.
  - $\Delta(\boldsymbol{h_w}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x})) = \frac{1}{2}\|\boldsymbol{h_w}(\boldsymbol{x}) - \boldsymbol{c}(\boldsymbol{x})\|^2$: squared loss will be assumed.
- $R(\boldsymbol{w}) = \underset{\boldsymbol{x}\sim P}{E}[\Delta(\boldsymbol{h_w}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))]$: the risk or generalization error or true error of the neural network predictor represented by $\boldsymbol{w}$.

Stochastic Gradient Descent Algorithm for Neural Networks

StochasticSubgradientDescent$(\boldsymbol{x}_0, T, \{\eta_t\}, \lambda, V, E, \sigma)$

1. $\boldsymbol{x}^{(1)} \leftarrow$ Random$(\boldsymbol{x}_0)$ ▷ a random initial point around $\boldsymbol{x}_0$
2. **for** $t \leftarrow 1$ **to** T **do**
3. $\quad \omega_t \leftarrow$ Sample(P)
4. $\quad \boldsymbol{v}_t \leftarrow$ Backpropagation$(\boldsymbol{x}^{(t)}, \omega_t, c(\omega_t), V, E, \sigma)$
5. $\quad \boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta_t(\boldsymbol{v}_t + \lambda \boldsymbol{x}^{(t)})$
6. **return** $\bar{\boldsymbol{x}}$ is the best performing $\boldsymbol{x}^{(t)}$ on a validation set