ECE 901: Large-scale Machine Learning and Optimization **Fall 2016**

Lecture 11 — 10/11

Lecturer: Dimitris Papailiopoulos *Scribe: Apul Jain*

Note: These lecture notes are still rough, and have only have been mildly proofread.

11.1 Neural Networks

Neural Networks were invented in 1940s as a model of computation inspired by the structure of neurons in the human brain. A neural network is a graph consisting of nodes and edges. During 1990s, Artificial Neural Networks (ANN) were designed as a hypothesis class for models learning from data. Broadly, ANNs gained attraction in two eras:

- 1st era: 1990s. During this time the focus was on expressivity and capacity of ANNs. It turned out that ANNs didn't work better than linear models and suffered from two major limitations:
  - Insufficient number of examples or in other words lack of training data
  - Limited computational power
- 2nd era: late 2000s. Nearly a decade later, ANNs gained traction again because of more than ever available compute power and large number of training examples. In fact sometime around 2006, image classification model designed using ANNs outperformed state of the art algorithm.

In this lecture we will focus on three topics: 1) Expressive power of neural networks 2) Computational hardness (Barriers) 3) Interesting/tractable questions.

11.2 Single-layer Neural Network

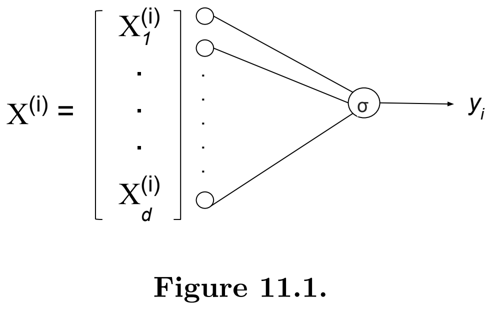

Figure 11.1.

Consider a single layer neural network as shown in figure 11.1. Here $X^{(i)}$ is the i-th d-dimensional feature vector, y_i is the ith label and σ is the activation function. Following are the examples of activation function.

$$\sigma(\alpha) = sign(\alpha)$$

$$\sigma(\alpha) = \frac{1}{1+e^{-\alpha}}$$

Suppose we take L-2 loss then ERM for the neural network will be

$$\min \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(w^T X^{(i)}))$$

If σ is invertible then problem is equivalent to

$$\min \frac{1}{n}\sum_{i=1}^{n}(\sigma^{-1}(y_i) - w^T X^{(i)})$$

Clearly single-layer neural network is equivalent to a linear classifier.

11.3 Feed-forward Neural Network

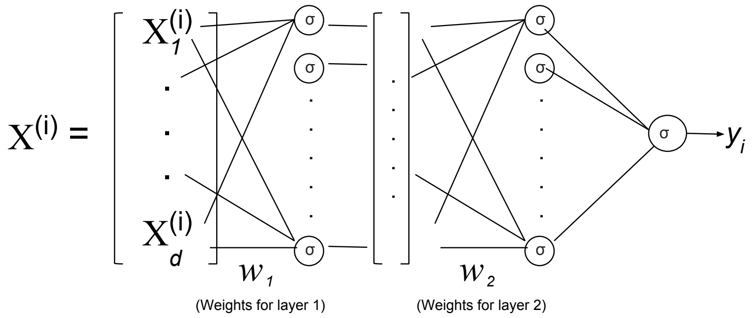

Figure 11.2.

Feed-forward neural networks have multiple hidden layers of nodes with no loops i.e., information flows in only one direction *forward* across the layers and no output goes to the previous layer.

11.4 Expressive power and Computational hardness

Consider a hypothesis class $\mathcal{H}_{V,E,\sigma}$ where V represents vertices (neurons) in the neural network and E represents directed edges between those vertices. As before, σ is the activation function applied at each neuron. Now given a hypothesis $h \in \mathcal{H}_{V,E,\sigma}$ ERM can be written as:

$$\min \frac{1}{n} \sum_{i=1}^{n} loss\left(h(X^{(i)}, w), y_i\right)$$

Here $X^{(i)}$ is the $d-$dimensional feature vector corresponding to i-th example.

Theorem 11.1. *For every dimension* d, $\exists\, h_{V,E,sign()}$ *such that it contains all binary functions* $f : \{+1, -1\}^d \to \{\pm 1\}$.

Proof: Let $\mathcal{X}$ be the set of feature vectors such that

$$\mathcal{X} = \{\pm 1\}^d$$

$$\mathcal{X}^+ = \{X \in \mathcal{X} : f(X) = 1\}$$

$$\mathcal{X}^- = \{X \in \mathcal{X} : f(X) = -1\}$$

Let's assume $|\mathcal{X}^+| = k$
We aim to design an architecture of neural network which will represent the above boolean function f.

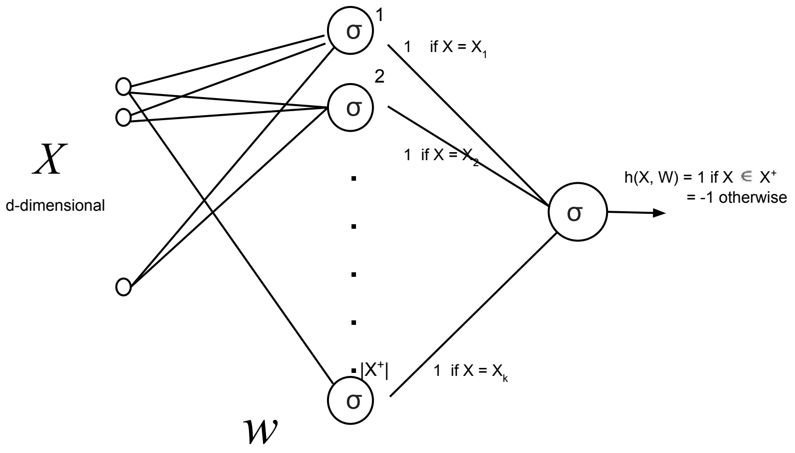

Figure 11.3.

Construct a neural network having input layer with $d+1$ neurons, one hidden layer with $2^d + 1$ neurons and one output neuron. Now we will try to make each neuron in the hidden layer recognize if $X \in \mathcal{X}^+$ or $\mathcal{X}^-$.

Take $X_1 \in \mathcal{X}^+$ then we can make the 1st neuron in the inner layer recognize if input is X_1 or not by following trick:

Set $w_1 = X_1$, where w_1 are the weights corresponding to neuron-1 in the inner layer. Then,

$$X_1^T w_1 = d$$

$$X_2^T w_1 \leq d - 1 \quad \text{if } X_2 \neq X_1$$

Hence, neuron-1 will output

$$sign\left(X^T w_1 - d + 1\right) \begin{array}{ll} = 1 & \text{if } X = w_1 \\ = -1 & \text{otherwise} \end{array}$$

Similarly we can design all the neurons of inner layer to recognize each individual $X_i \in \mathcal{X}^+$. Final output layer neuron will combine outputs of inner layer as follows:

$$sign\left(\sum_i \sigma_i(w_i^T X) + k\right) \begin{array}{ll} = 1 & \text{if } X \in \mathcal{X}^+ \\ = -1 & \text{otherwise} \end{array}$$

This construction gives us a set of weights for a neural network which represents the given function f.

Let's analyze the complexity of this network. Since VC-dimension of $\mathcal{H}_{V,E,sign()} = 2^d$ and $\mathcal{H}_{V,E,sign()} \leq O(Elog|V|) \leq O(V^3)$. This implies $|V| \geq 2^d$. Hence the network is exponential in size.

$\square$

Theorem 11.2. *Let $f : \{\pm 1\} \to \{\pm 1\}$. If complexity to evaluate $f(X)$ is O(T). Then there exists a neural network $h_{V,E,sign()}$ with size O(T^2).*

Proof: : This can be proved using the relation between the time complexity of programs and the circuit complexity which measures the size of Boolean circuits required to calcualte functions. However, learning complexity of such a network can't be bounded. For further discussion, see Theorem 20.3 in ref [1]. $\square$

Solving ERM with 0-1 loss for the class of neural networks $\mathcal{H}_{V,E,sign()}$ is as hard as $k \geq 3$ coloring problem. So although in the worst case we are doomed, things are not as bad in practice!

11.5 Interesting/tractable questions

In subsequent lectures we will discuss following things:

$$\min_w \sum_{i=1}^{n} \left(h(X_i, w) - y_i\right)^2$$

- What can we promise if we run SGD using back propagation. We will see if $E||\nabla f(X_k)||^2 \to 0$ as $k \to \infty$.
- What about overparameterization. Does having more weights than samples help? Can we make loss = 0?
- Can we solve it in poly time? Specifically we will look into generalization property of SGD for non-convex functions as well.
- How do you scale to parallel and distributed architecture?

References

[1] Understanding Machine Learning: From Theory to Algorithms, by Shai Shalev-Shwartz and Shai Ben-David