E0 306: Deep Learning: Theory and Practice 31^{st} January 2019

Scribe Notes for Lecture 8

Lecturer: Navin Goyal *Scribe*: Kapil Pathak

1 Decomposition of test error into approximation, optimization, generalization errors

Consider a set up with supervised with feed-forward networks where $\mathcal{X}$ is input distribution, $\mathcal{Y}$ target and $\mathcal{H}$ hypothesis class. Let $\mathcal{A}$ be an algorithm for learning the network.

$h_{A,S}$= Hypothesis output by $\mathcal{A}$ on input $\mathcal{S}$.

$h_S^* = \text{argmin}_{h\in\mathcal{H}} L_S(h)$

$h^* = \text{argmin}_{h\in\mathcal{H}} L_D(h)$

For good generalization, a neural network needs to approximate classification function (assuming a classification task). For that, the training error should be as small as possible. But that doesn't imply the neural network will have a good generalization over the unseen data.

According to above notation, $L_D(h_{A,S})$ is a generalization error due to a hypothesis obtained from algorithm $\mathcal{A}$ and training set $\mathcal{S}$. $L_D(h_{A,S})$ can be decomposed as,

$$L_D(h_{A,S}) = L_D(h_{A,S}) - L_S(h_{A,S}) + L_S(h_{A,S}) - L_S(h_S^*) + L_S(h_S^*) - L_D(h^*) + L_D(h^*)$$

where,
Approximation error(ϵ_{app}) =$\text{L}_D(h^*)$

Optimization error(ϵ_{opt}) =$\text{L}_S(h_{A,S}) - L_S(h_S^*)$

Generalization error(ϵ_{gen}) =$\sup_{h\in\mathcal{H}} \mid L_S(h) - L_D(h) \mid$

$L_S(h_S^*) - L_D(h^*) \leq L_S(h^*) - L_D(h^*) \leq$Generalization error

From above equations, we get

$$L_D(h_{A,S}) \leq \epsilon_{app} + \epsilon_{opt} + 2\epsilon_{gen}$$

Denote the Training Set $\mathcal{S} = \left\{ \left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right) \right\}$

where $x^{(i)} \in \mathbb{R}^{\mathrm{d}}$, denoted by $x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \ldots, x_{\mathrm{d}}^{(i)}\right)$, and $y^{(i)} \in \{0, 1\}$

2 Approximation Error (ϵ_{app})

2.1 Universal Approximation Theorem [4][5][3]

Let $P : \mathbb{R} \to \mathbb{R}$ be non constant, bounded and continuous. Then for any $\epsilon > 0$, positive integer m, $f : [0,1]^m \to \mathbb{R}$, $\exists$ integer N, real numbers a_i, b_i and vectors $w_i \in \mathcal{R}^{\mathrm{m}}$ for i $\in$\{1,2,...,N\},such that,$F(x) = \sum_{i \in [N]} a_i P(\langle w_i, x \rangle + b_i)$is an ϵ_{approx} of $f :\mid f(x) - F(x) \mid \leq \epsilon$ $\forall x \in [0,1]^m$.

But the question arises whether the depth increases expressive power of a neural network.

3 Generalization Error (ϵ_{gen})

Diagram pending: Size of one hidden layer network on small MNIST.

Here, the point to note is that no overfitting as we overparametrize the neural network. This contradicts the bias-variance trade-off.

CIFAR-10 dataset is an image dataset having training set size=5×10^4 Images of size $32 \times 32 \times 3 = 3072$

Architecture	No of parameters
AlexNet	1.3×10^6
Inception	1.6×10^6
Imagenet	1.2×10^6
Inception V4	4.2×10^7
ResNet-152	6×10^7
VGG-19	1.4×10^8
AmaebaNet -B (Top 1 accuracy=84.3 % and Top 5 accuracy=97 %)	5.5×10^8

3.1 A Convergence Theory for Deep Learning via Overparametrization [2]

3.2 Learning and Generalization in Overparametrized Neural Networks [1]

3.2.1 Informal Statement

- Training data = $\{(x_i, y_i^*)\}_{i \in [n]}$ where $x \in R^{\partial}, y_i^* \in R^d$
- Separability Assumption: $\forall$ distinct $i, j \in [n]$ $wehave \left\| x_i - x_j \right\| \geq \delta$
- L-hidden layer ReLU fully connected network with each other of size of m.
- He initialization
- $l_2 - loss$ for regression, cross-entropy for classification

- For regression, $m \geq poly(n, L, \frac{1}{\delta})$ then gradient descent or stochastic gradient descent, find $\epsilon - error$ global minimum for $l_2 - loss$ in $poly(n, L, \frac{1}{\delta} log(\frac{1}{\epsilon}))$ iterations.
- For classification, gradient descent or stochastic gradient descent find a classifier with 0 classification error in $poly(n, L, \frac{1}{\delta})$ iterations with number of samples $m \geq poly(n, L, \frac{1}{\delta})$
- Extensions to other Lip-smooth losses and architectures.

3.2.2 Remarks:

- It makes ReLU crucial.
- Independent of dimension d
- Overparametrization in theorem implies in practice
- Dependency on L (Not clear if it's correct)

Main insight in [1] and [2]

$ReLU(\langle w, x \rangle + b) = \langle w, x \rangle + b \; if \langle w, x \rangle + b \geq 0$

For highly overparametrized settings, activation pattern essentially remain constant, weights don't change much.

For $w^0, w^1, ..., w^t, \left\| w^0 - w^t \right\|$ is small. Still small change is sufficient as the network is over-parametrized.

Notation: Consider l_2-regression

$\phi(x) = max\{0, x\}$=$ReLU(x)$ For $v \in R^m, \phi(v) = (\phi(v_1), \phi(v_2), ..., \phi(v_m))$.

Data is normalized such that $\|x_i\| = 1$ and $(x_i)_\partial = \frac{1}{\sqrt{2}}$

We assume that there no biases added before applying activation functions.

Output from a neural network can be given as,

$y_i = N(x_i) = B\phi(W_L\phi(....W_2\phi(W_1\phi(Ax))))$ where B denotes output layer,

W_L denotes Hidden layer L, $A \in R^{m \times d}, W_l \in R^{m \times m}, B \in R^{d \times m}$

Initialize $A_{ij} \sim N(0, \frac{2}{m})$ for $i \in [m]$, $j \in [\partial]$

$[W_l^{(0)}] \sim N(0, \frac{2}{m})$ for $i, j \in [m]$, $l \in [L]$

$B_{ij} \sim N(0, \frac{1}{d})$ for $i \in [d]$, $j \in [m]$

$\overrightarrow{W} = (W_1, ..., W_L), \overrightarrow{W^{(0)}} = (W_1^{(0)}, ..., W_L^{(0)})$

$F(\overrightarrow{W}) = \sum_{i \in [n]} F_i(\overrightarrow{W})$

$F_i(\overrightarrow{W}) = \frac{1}{2}\|N(x_i) - y_i^*\|_2^2$ for i $\in [n]$

$\nabla F(\overrightarrow{W}) = (\nabla_{W_1} F(\overrightarrow{W}), ..., \nabla_{W_L} F(\overrightarrow{W}))$

Here, we are training with respect to $\overrightarrow{W} keeping A and B to initialize values$.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in over-parameterized neural networks, going beyond two layers. *CoRR*, abs/1811.04918, 2018.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *CoRR*, abs/1811.03962, 2018.

[3] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theor.*, 39(3):930–945, May 1993.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[5] Kurt and Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.