Introduction to Deep Neural Networks

Numerical Methods for Deep Learning

March 12, 2018

Why deep models

- Fundamental theorem of NN suggests that we can fit **any** data by two layers.
- But - The width of the layer can be very large $\mathcal{O}(n_d)$ where n_d is the dimension of the data.
- Deeper architectures can lead to more efficient descriptions of the problem.
- No real proof but lots of practical experience.

Deep models

How deep is deep?
We will answer this question later ...

Until recently, the architecture was

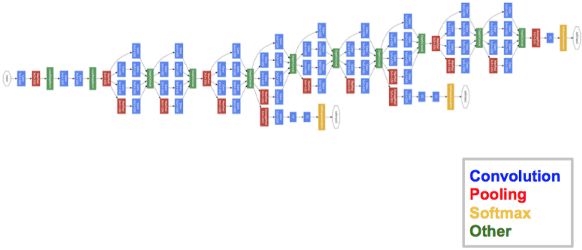
$$
\begin{aligned}
\mathbf{Y}_2 &= f(\mathbf{K}_1\mathbf{Y}_1 + b_1) \\
&\;\;... \\
\mathbf{Y}_N &= f(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + b_{N-1})
\end{aligned}
$$

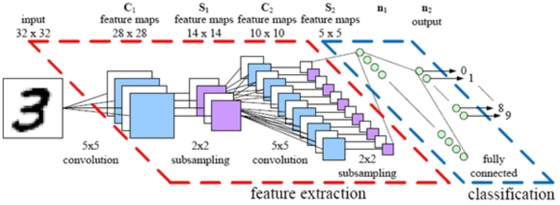
And use $\mathbf{Y}_N$ to classify

$$
\min \quad E\left(\mathbf{W}\mathbf{Y}_N(\mathbf{K}_1, \ldots, \mathbf{K}_{N-1}, b_1, \ldots, b_{N-1}), \mathbf{C}^{\rm obs}\right)
$$

Deep models

Very complex architectures





Deep models

- Architecture has many (billions) of parameters.
- Very difficult to design
- Strange computational behavior
- Very unpredictable

In 2015 He et-al came with a new architecture that solves many of the problems

Deep models

Residual Network

$$
\begin{aligned}
\mathbf{Y}_2 &= \mathbf{Y}_1 + f(Y_1\mathbf{K}_1 + b_1) \\
&\dots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + f(\mathbf{Y}_{N-1}\mathbf{K}_{N-1} + b_{N-1})
\end{aligned}
$$

And use $\mathbf{Y}_N$ to classify

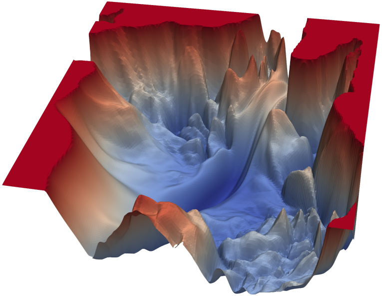
$$
\min \quad E\left(\mathbf{Y}_N(\mathbf{K}_1, \dots, \mathbf{K}_{N-1}, b_1, \dots, b_{N-1})\mathbf{W}, \mathbf{C}^{\mathrm{obs}}\right)
$$

Much more stable!

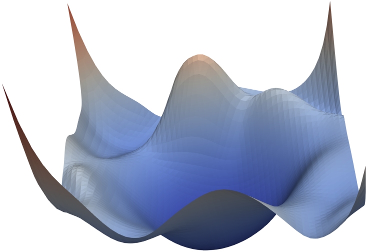
Deep models - Residual Network

A way to visualize the landscape, choose two random orthogonal directions $\delta\theta_1, \delta\theta_2$ and compute the function

$$g(h_1, h_2) = E(\theta + h_1\delta\theta_1 + h_2\delta\theta_2)$$



Regular NN



Resifual NN

Deep models - Residual Network

Why are ResNets more stable?
A small change

$$
\begin{aligned}
\mathbf{Y}_2 &= \mathbf{Y}_1 + hf(\mathbf{K}_1\mathbf{Y}_1 + b_1) \\
&\dots \\
\mathbf{Y}_N &= \mathbf{Y}_{N-1} + hf(\mathbf{K}_{N-1}\mathbf{Y}_{N-1} + b_{N-1})
\end{aligned}
$$

This is nothing but a forward Euler discretization of the ODE of the form

$$\dot{\mathbf{Y}} = f(\mathbf{K}(t)\mathbf{Y} + b(t))$$

We can understand the behavior by learning the dynamics of nonlinear evolution equation.

A crash review on ODE's

Given the ODE

$$\dot{\mathbf{y}} = f(t, \mathbf{y})$$

Assume f differentiable with

$$\mathbf{J}(t, \mathbf{y}) = \frac{\partial f}{\partial \mathbf{y}} \qquad \text{change slowly}$$

Then

- If $Re(\text{eig}(\mathbf{J})) > 0 \qquad \rightarrow$ Unstable
- If $Re(\text{eig}(\mathbf{J})) < 0 \qquad \rightarrow$ Stable (converge to a stationary point)
- If $Re(\text{eig}(\mathbf{J})) = 0 \qquad \rightarrow$ Stable, energy bounded

Residual Network as a path planning problem

$$\dot{\mathbf{Y}} = f(\mathbf{K}\mathbf{Y} + b) \quad \mathbf{Y}(0) = \mathbf{Y}_0$$

The goal is to plan a path such that the initial data can be linearly separated

Continuous vs discrete stability

Assume that we have a stable network in the continuous space

$$\dot{\mathbf{Y}} = f(\mathbf{K}\mathbf{Y} + b) \quad \mathbf{Y}(0) = \mathbf{Y}_0$$

And assume we use the forward Euler to discretize

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + hf(\mathbf{K}_j\mathbf{Y}_j + b_j)$$

Is the network stable?

Not always ...

Continuous vs discrete stability

Look at the simplest network ever

$$\dot{\mathbf{Y}} = \lambda \mathbf{Y}$$

And assume we use the forward Euler to discretize

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\lambda \mathbf{Y}_j = (1 + h\lambda)\mathbf{Y}_j$$

Then the method is stable only if

$$|1 + h\lambda| \leq 1$$

Not every network is stable!
Depends on our Jacobian