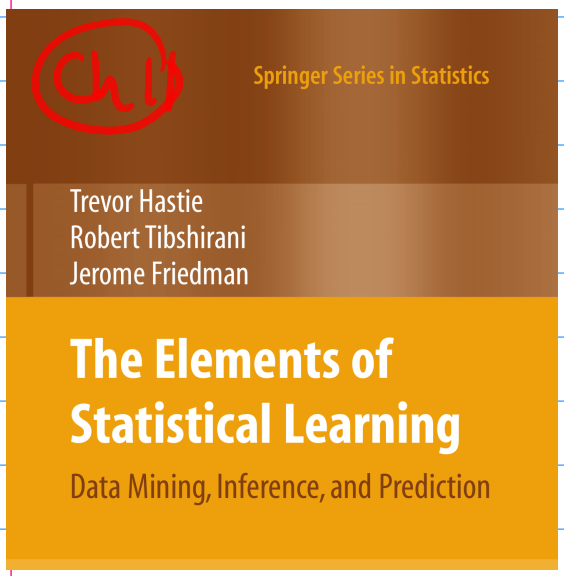
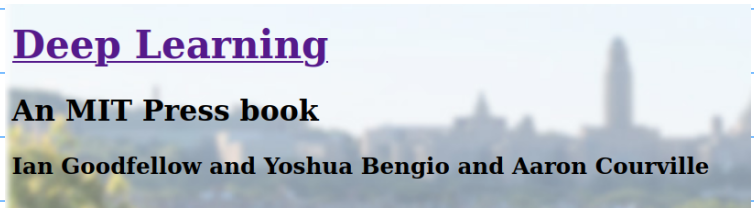
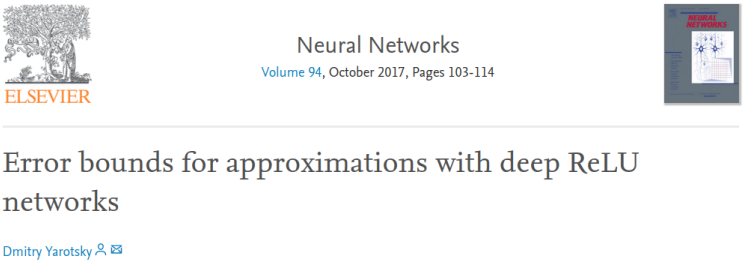
Lecture 26: Introduction to Neural Networks

In the next few lectures we will take a brief look at yet another family of models for machine learning & data analysis which have generated a lot of hype in recent years, ie, neural networks.

- Neural networks are not a new idea, despite the hype generated around them in the past 5-10 years.
  - + first IEEE int. conf. on Neural Networks was held in 1987.
  - + Idea of neural nets goes back to 1950's.



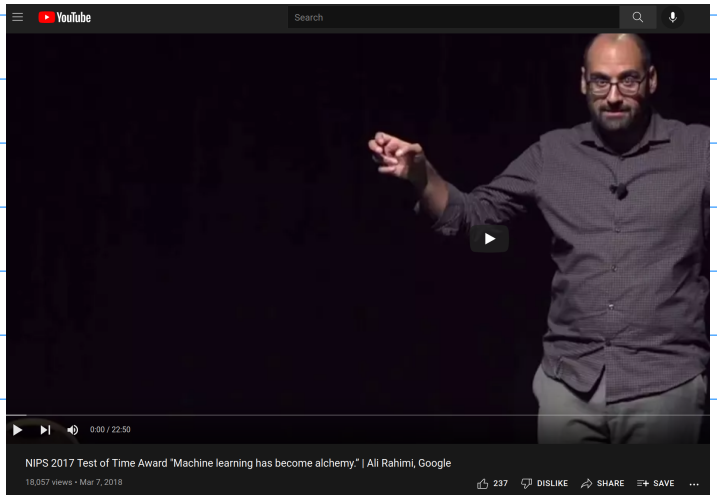




- Our goal in these few lectures is to introduce the idea of neural networks & try to gain some understanding of how they work.

- We will only discuss what are known as **feedforward neural networks** which encompass many interesting models but not all models used in practice.

- At the moment our theoretical understanding of NN's is far more limited than the scope of their applications. This means majority of NN research is focused on developing algorithms & involves a lot of expert knowledge, intuition, & trial & error



Ali Rahimi's talk at NIPS 2017.

26.1 Linear vs Compositional models

The methods that we saw in most of the course, (upto Lec 17) were based on models of the form

$$f(\underline{x}) = \sum_{j=0}^{J-1} \beta_j \cdot F_j(\underline{x})$$

we often refer to these as linear models because the function f depends linearly on the parameters β_j & the features $F_j : \mathbb{R}^d \to \mathbb{R}^J$

Recall in Lect 17 we discussed the idea of considering $\underline{z} = (F_0(\underline{x}), F_1(\underline{x}), \cdots, F_{J-1}(\underline{x}))$ then

$$f(\underline{z}) = \sum_{j=0}^{J-1} \beta_j \cdot z_j \quad , \underline{z} \in \mathbb{R}^J$$

so f is a linear function in the $\underline{z}$ space.

But you might wonder, why stop at linear functions of $\underline{z}$?

Consider new set of features

$$G_l : \mathbb{R}^J \to \mathbb{R}^M \quad , \quad l = 0, \cdots, L-1$$

& write

$$f(\underline{z}) = \sum_{l=0} \alpha_l G_l(\underline{z})$$

But this is equivalent to $F(x) = (F_0(x), \ldots, F_{J-1}(x))$

$$f(x) = \sum_{l=0}^{L-1} \alpha_l G_l(F(x))$$

& you can continue this approach as many times as you like.

The core idea here is that f is no longer a linear combination of features

$$f(x) = \sum_j \beta_j F_j(x)$$

But rather a combination of compositions of features

$$f(x) = \sum_l \alpha_l G_l(F_0(x), \ldots, F_{J-1}(x))$$

This compositional model is at heart of NNs & in particular feedforward NNs & Multi-layer perceptron (MLP)

26.2 Feed forward NNs (FNN)

FNNs consider a particular type of functions based on an activation function

$g: \mathbb{R} \to \mathbb{R}$ ex • Rectifier Linear Unit
$g(t) = \max(0, t)$
• sigmoid
$g(t) = (1 + \exp(-t))^{-1}$

Along with its vectorized extension

$g: \mathbb{R}^n \to \mathbb{R}^n$, $g(x_0, \dots, x_{n-1}) = (g(x_0), \dots, g(x_{n-1}))$.

Then an FNN is a function $f: \mathbb{R}^d \to \mathbb{R}^m$ (input $\underline{x}$, output $\underline{y}$) of the form

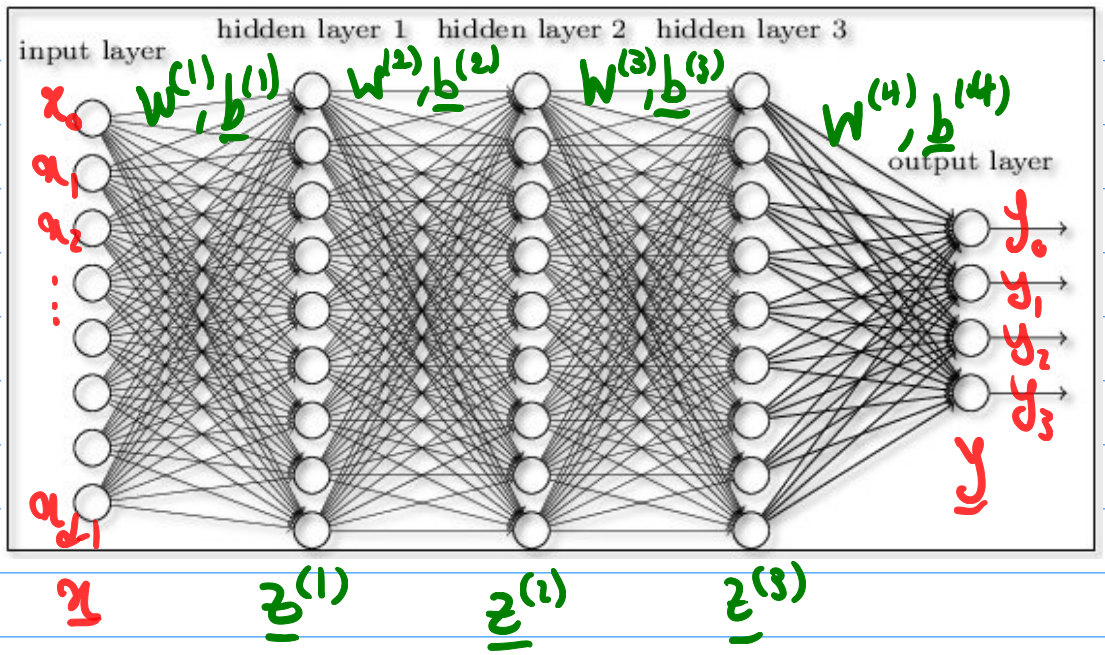
$$\begin{cases} f(\underline{x}) = W^{(L)} g(\underline{z}^{(L)}) + b^{(L)} \\ \underline{z}^{(l+1)} = W^{(l)} g(\underline{z}^{(l)}) + b^{(l)}, \quad l = 1, \dots, L-1 \\ \underline{z}^{(0)} = \underline{x} \end{cases}$$

Here $L \geq 0$ is an integer (depth) of the FNN, $W^{(l)}$ are weight matrices of each "layer", the $b^{(l)}$ are the bias vectors of each layer.

for simplicity we consider the setting when

$$W^{(1)} \in \mathbb{R}^{k \times d}, \quad \{W^{(l)}\}_{l=2}^{L-1} \in \mathbb{R}^{k \times k}, \quad W^{(L)} \in \mathbb{R}^{m \times k}$$

$$\{b^{(l)}\}_{l=1}^{L-1} \in \mathbb{R}^{k}, \quad b^{(L)} \in \mathbb{R}^{m}.$$

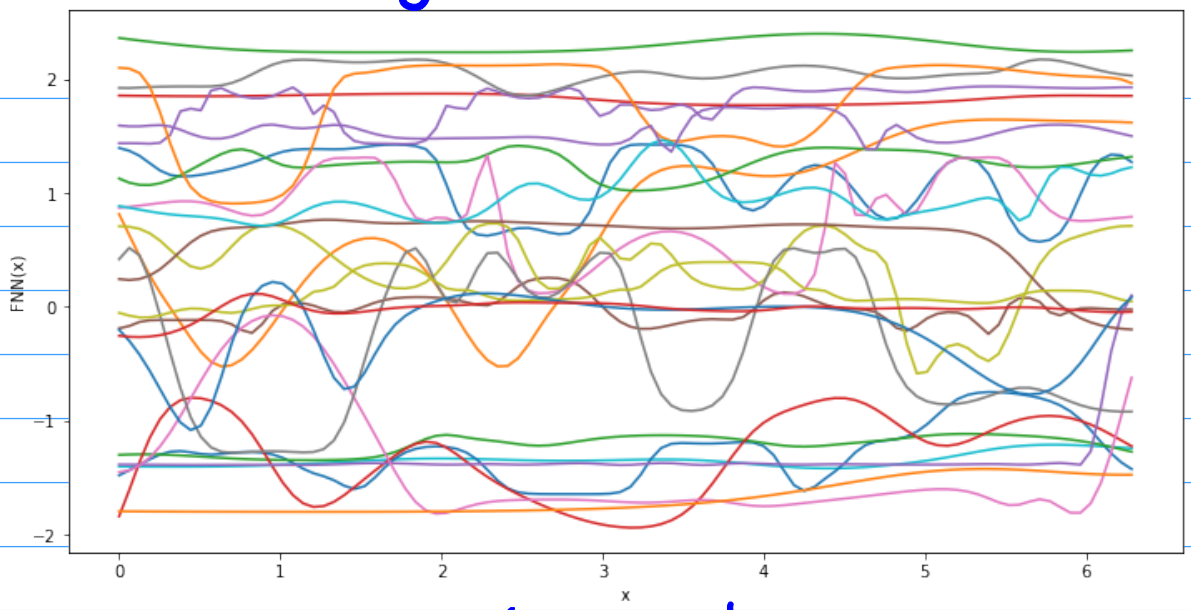


$$x \longmapsto f \longrightarrow y$$

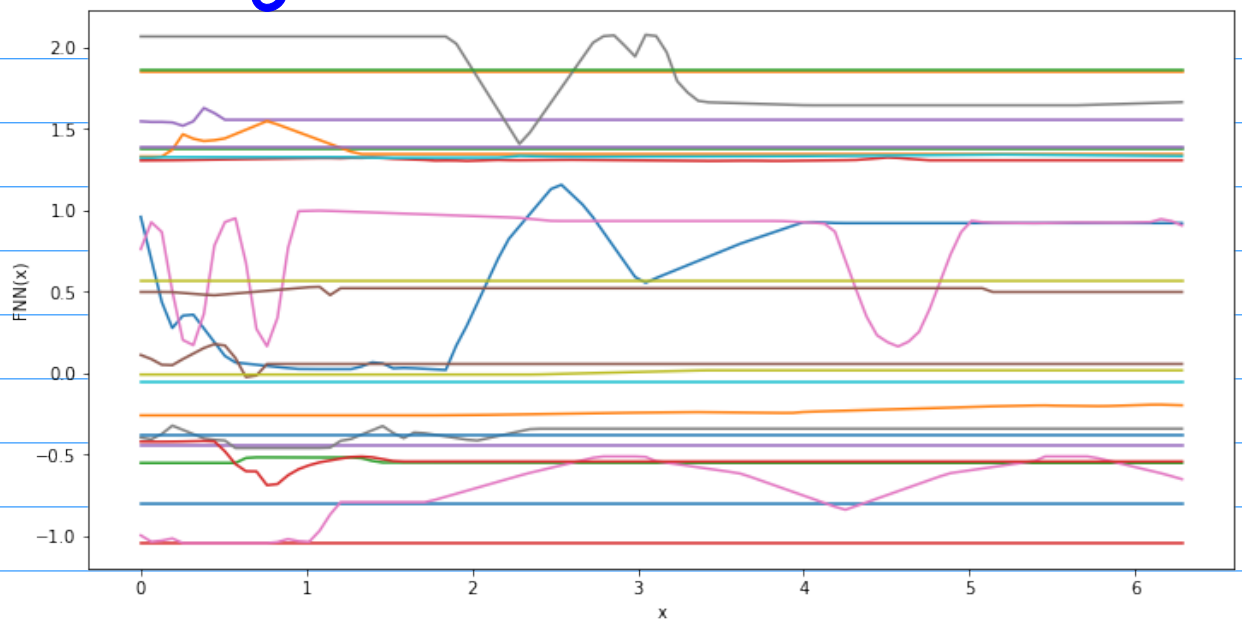
26.3 What do FNNs look like?

Before applying FNNs for ML tasks it is helpful to try to get a sense for what sort of functions they generate. To do this we will look at some simple NNs with random weights $W^{(l)}$ & biases $b^{(l)}$ mapping $\mathbb{R}$ to $\mathbb{R}$. (ie random functions).

$g(t) = \cos(t)$



$g(t) = \max(0, 1 - |t|)$



$g(t) = \max(0, t)$ ReLU