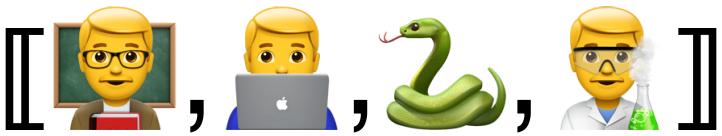
Lecture Notes for

Machine Learning in Python



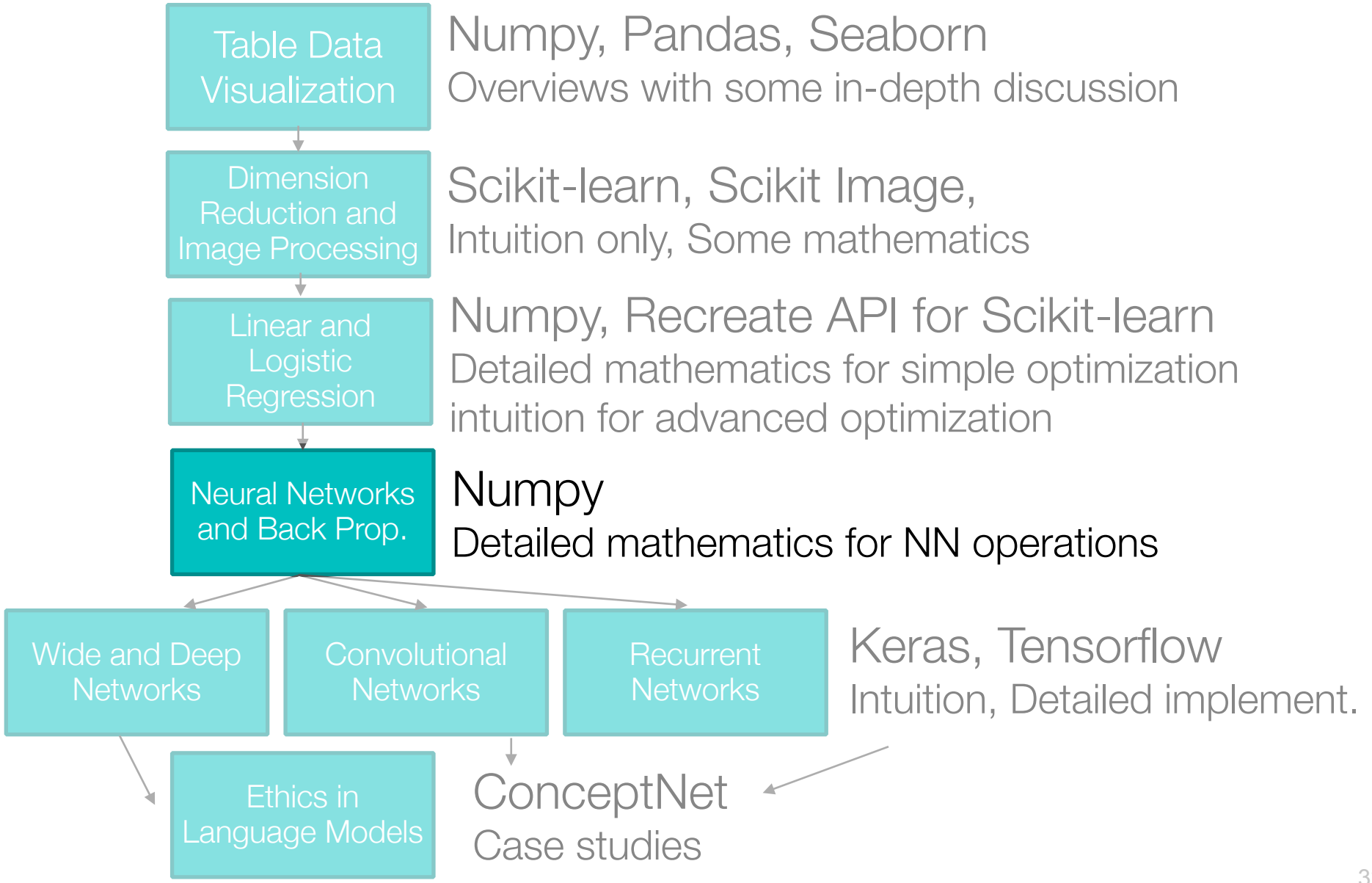
Professor Eric Larson

Optimizing Neural Networks

Class Logistics and Agenda

- Logistics
  - Grading
- Agenda:
  - "Finish" Town Hall
  - Practical Multi-layer Architectures
  - Programming Examples
- Next Time: More MLPs

Class Overview, by topic



Tyler Rablin @Mr_Rablin · 2d

You're not grading assignments.

You're collecting evidence to determine student progress and pointing them towards their next steps.

Make the mental switch. It matters.

Town Hall

Review: Back propagation history

- 1986: *Rumelhart*, *Hinton*, and *Williams* popularize gradient calculation for multi-layer network
  - *actually* introduced by Werbos in 1982
- **difference**: Rumelhart *et al.* validated ideas with a computer
- until this point no one could train a multiple layer network consistently
- algorithm is popularly called **Back-Propagation**
- wins pattern recognition prize in 1993, becomes de-facto machine learning algorithm until: SVMs and Random Forests in ~2004
- would eventually see a resurgence for its ability to train algorithms for Deep Learning applications: **Hinton is widely considered the founder of deep learning**

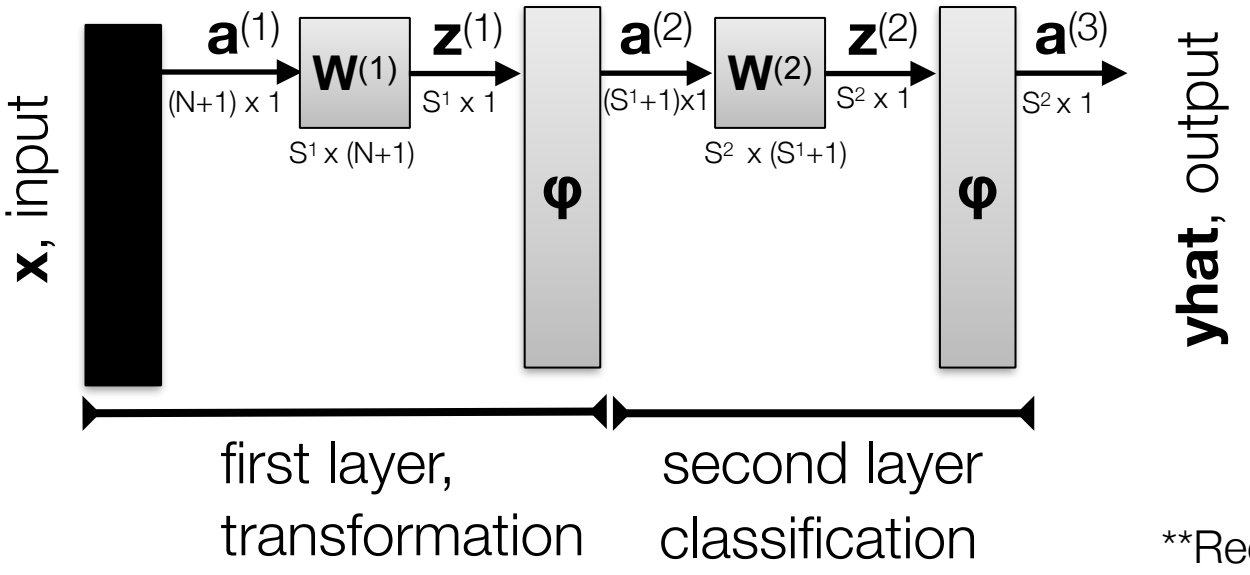
David Rumelhart

1942-2011

Geoffrey Hinton

Review: Back propagation

- Steps:
  - propagate weights forward
  - calculate gradient at final layer
  - back propagate gradient for each layer
    - via recurrence relation

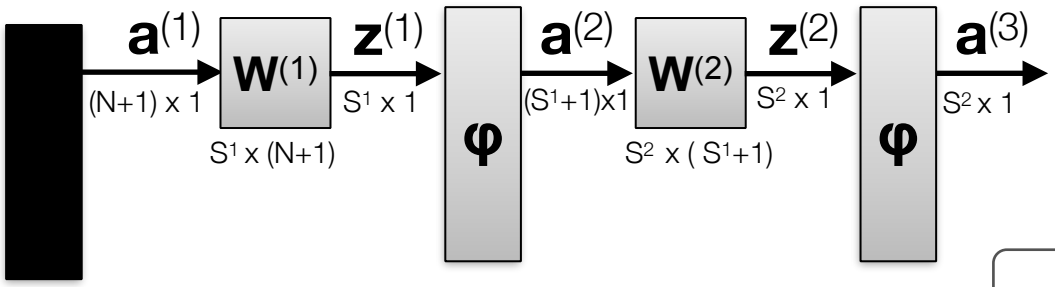


$$J(\mathbf{W}) = \left\| \mathbf{Y} - \hat{\mathbf{Y}} \right\|^2$$

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \eta \frac{\partial J(\mathbf{W})}{\partial w_{i,j}^{(l)}}$$

**Recall from Flipped Assignment!

Review: Back Propagation Summary



1. Forward propagate to get **Z**, **A**
2. Get final layer gradient
3. Back propagate sensitivities
4. Update each $\mathbf{W}^{(l)}$

$$\mathbf{V}^{(2)} = -2(\mathbf{Y} - \mathbf{A}^{(3)}) * \mathbf{A}^{(3)} * (1 - \mathbf{A}^{(3)})$$

$$\nabla^{(2)} = \mathbf{V}^{(2)} \cdot [\mathbf{A}^{(2)}]^T$$

$$\mathbf{V}^{(1)} = \mathbf{A}^{(2)} * (1 - \mathbf{A}^{(2)}) * [\mathbf{W}^{(2)}]^T \cdot \mathbf{V}^{(2)}$$

$$\nabla^{(1)} = \mathbf{V}^{(1)} \cdot [\mathbf{A}^{(1)}]^T$$

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \nabla^{(l)}$$

Where is the problem of **vanishing gradients** introduced?

**Recall from Flipped Assignment!

Lightning Demo

`07. MLP Neural Networks.ipynb`

same as Flipped Assignment!
with regularization
and vectorization
and mini-batching

Self test: Should we see examples where:

A. $\mathbf{z} = \mathbf{W} \cdot \mathbf{a}_{bias}$ where bias is concatenated, and $\mathbf{W}$ incorporates bias term?

B. $\mathbf{z} = \mathbf{W} \cdot \mathbf{a} + \mathbf{b}$ where we separate out the bias explicitly ?