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

Table Data Visualization

Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing

Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression

Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.

Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**

Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models

ConceptNet
Case studies

Town Hall

Tyler Rablin @Mr_Rablin · 2d

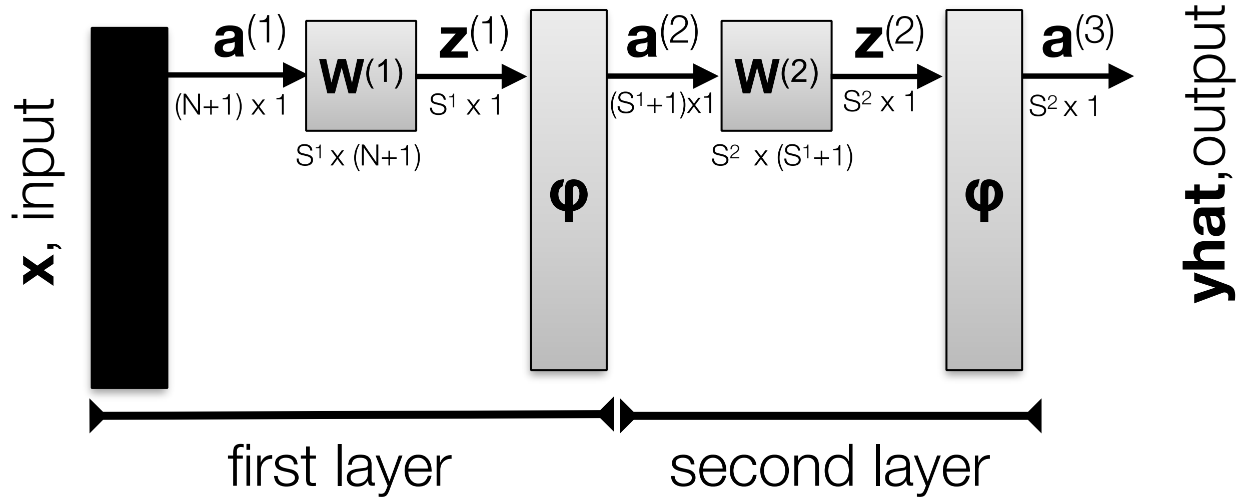
You're not grading assignments.

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

Make the mental switch. It matters.

Review: More Advanced Architectures: MLP

- The multi-layer perceptron (MLP):
  - two layers shown, but could be arbitrarily many layers



each row of **yhat** is no longer independent of the rows in **W** so we cannot optimize using one versus all!!!

$$\mathbf{yhat}^{(i)} = \begin{bmatrix} \varphi\left({}_{row=1}\mathbf{w}^{(2)} \cdot \varphi(\mathbf{W}^{(1)}\mathbf{a}^{(1)})\right) \\ \cdots \\ \varphi\left({}_{row=S}\mathbf{w}^{(2)} \cdot \varphi(\mathbf{W}^{(1)}\mathbf{a}^{(1)})\right) \end{bmatrix}$$

one hot

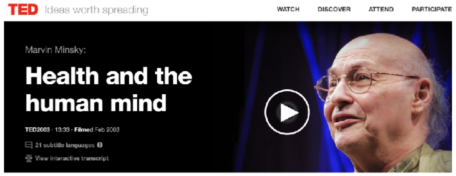
Review: The Rosenblatt-Widrow-Hoff Dilemma

- 1960's: Rosenblatt got into a public academic argument with Marvin Minsky and Seymour Papert

> "Given an elementary α-perceptron, a stimulus world W, and any classification C(W) for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to C(W) in finite time…"

- Minsky and Papert publish limitations paper, 1969:

> "the style of research being done on the perceptron is doomed to failure because of these limitations."



- Widrow and Rosenblatt try to build bigger networks without limitations and fail
  - Neural Networks research **basically stops** for **17 years**
- **Until**: researchers revisit training bigger networks
  - neural networks with multiple layers

Review: More Advanced Architectures: history

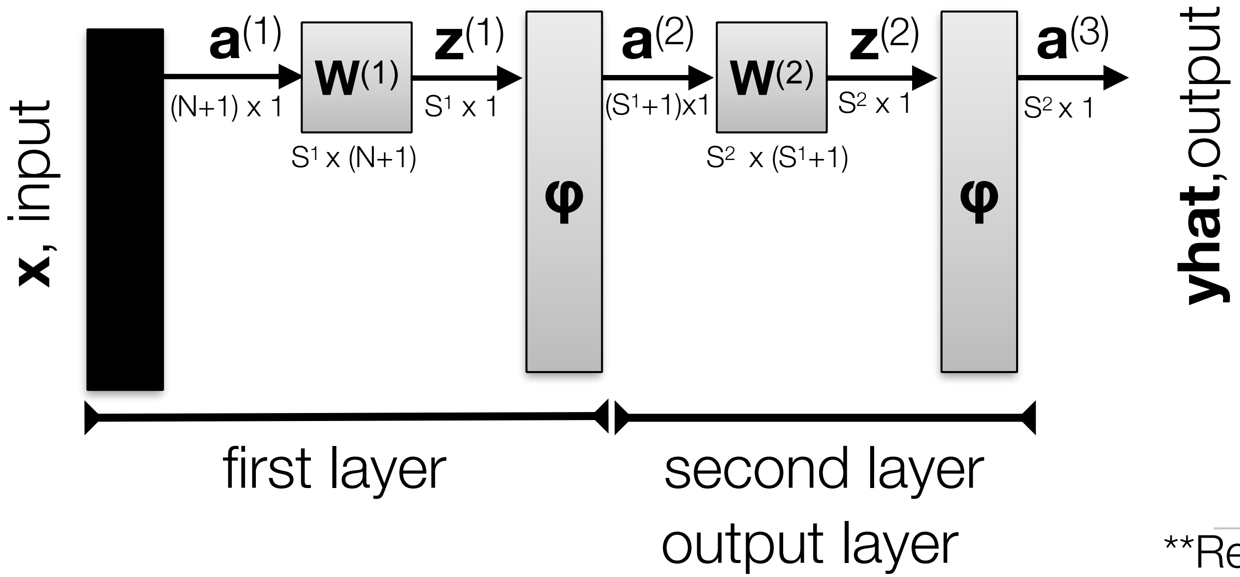
- 1986: *Rumelhart*, *Hinton*, and *Williams* popularize gradient calculation for multi-layer network
  - *actually* introduced by Werbos in 1982
- **difference**: Rumelhart *et al.* validated ideas with a computer
- until this point no one could train a multiple layer network consistently
- algorithm 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

Geoffrey Hinton

Review: Back propagation

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

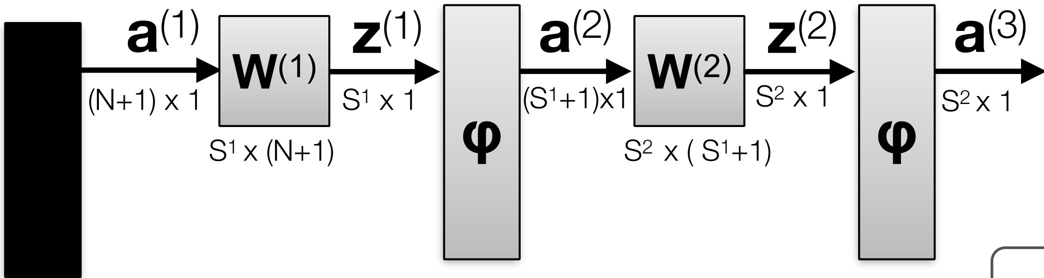


$$J(\mathbf{W}) = \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2$$

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

**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

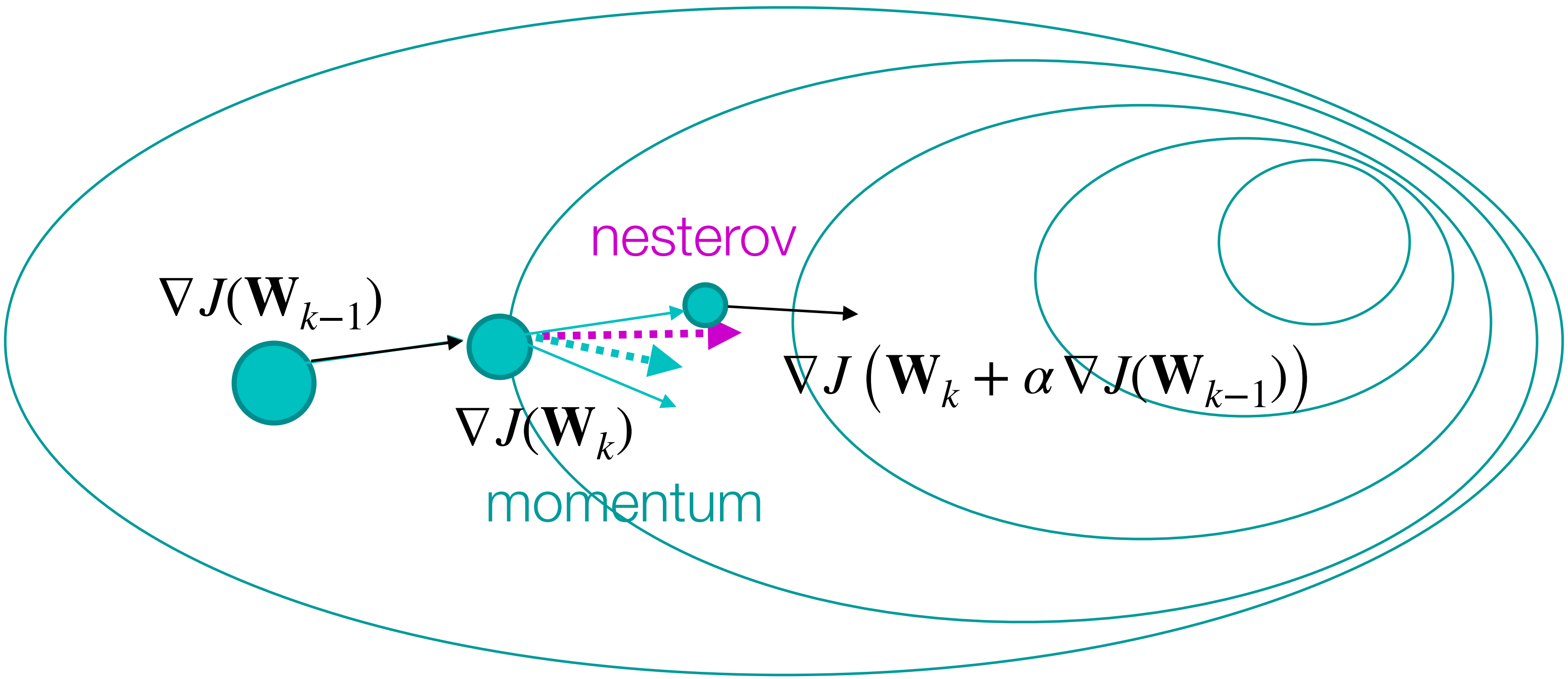
Problems with Advanced Architectures

- Numerous weights to find gradient update
  - minimize number of instances
  - **solution**: mini-batch
- **new problem**: mini-batch gradient can be erratic
  - **solution**: momentum
    - use previous update in current update

Common Adaptive Strategies $\mathbf{W}_{k+1} = \mathbf{W}_k - \rho_k$

- Momentum $\rho_k = \alpha \nabla J(\mathbf{W}_k) + \beta \nabla J(\mathbf{W}_{k-1})$

- Nesterov's Accelerated Gradient $\rho_k = \underbrace{\beta \nabla J\left(\mathbf{W}_k + \alpha \nabla J(\mathbf{W}_{k-1})\right)}_{\text{step twice}} + \alpha \nabla J(\mathbf{W}_{k-1})$



Adaptive Strategy: Cooling

- Space is no longer convex
    - **One solution**:
        - start with large step size
        - "cool down" by decreasing step size for higher iterations