ECE 6504: Deep Learning for Perception

Topics:

- Neural Networks
  - Backprop
  - Modular Design

Dhruv Batra

Virginia Tech

Administrativia

- Scholar
  - Anybody not have access?
  - Please post questions on Scholar Forum.
  - Please check scholar forums. You might not know you have a doubt.

- Sign up for Presentations
  - https://docs.google.com/spreadsheets/d/1m76E4mC0wfRjc4HRBWFdAlXKPIzlEwfw1-u7rBw9TJ8/edit#gid=2045905312

Plan for Today

- Notation + Setup
- Neural Networks
- Chain Rule + Backprop

Supervised Learning

- Input: x (images, text, emails…)
- Output: y (spam or non-spam…)
- (Unknown) Target Function
  - f: X → Y (the "true" mapping / reality)
- Data
  - $(x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N)$
- Model / Hypothesis Class
  - g: X → Y
  - $y = g(x) = \text{sign}(w^T x)$
- Learning = Search in hypothesis space
  - Find best g in model class.

Basic Steps of Supervised Learning

- **Set up** a supervised learning problem

- **Data collection**
  - Start with training data for which we know the correct outcome provided by a teacher or oracle.

- **Representation**
  - Choose how to represent the data.

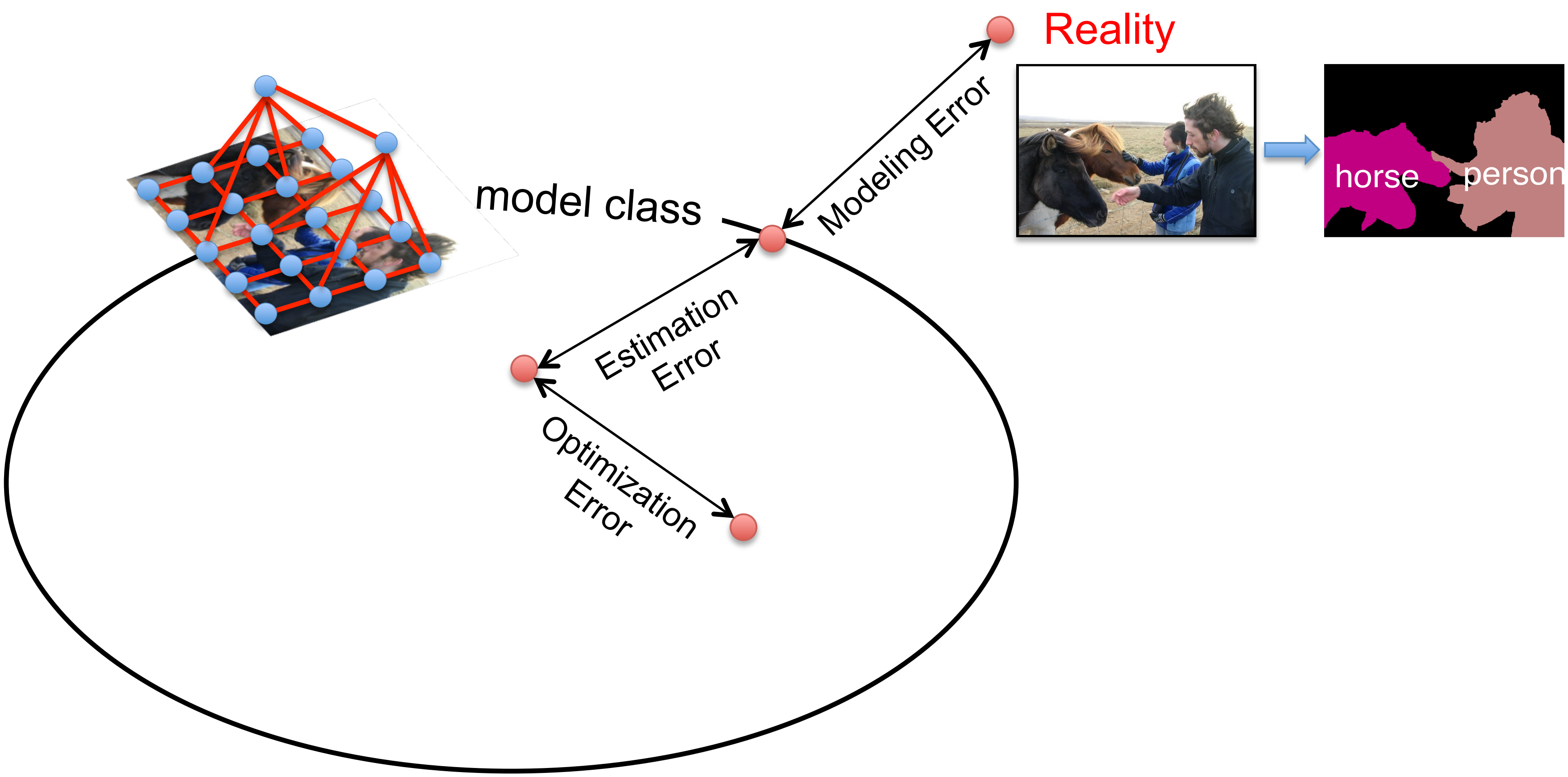
- **Modeling**
  - Choose a hypothesis class: $H = \{g: X \rightarrow Y\}$

- **Learning/Estimation**
  - Find best hypothesis you can in the chosen class.

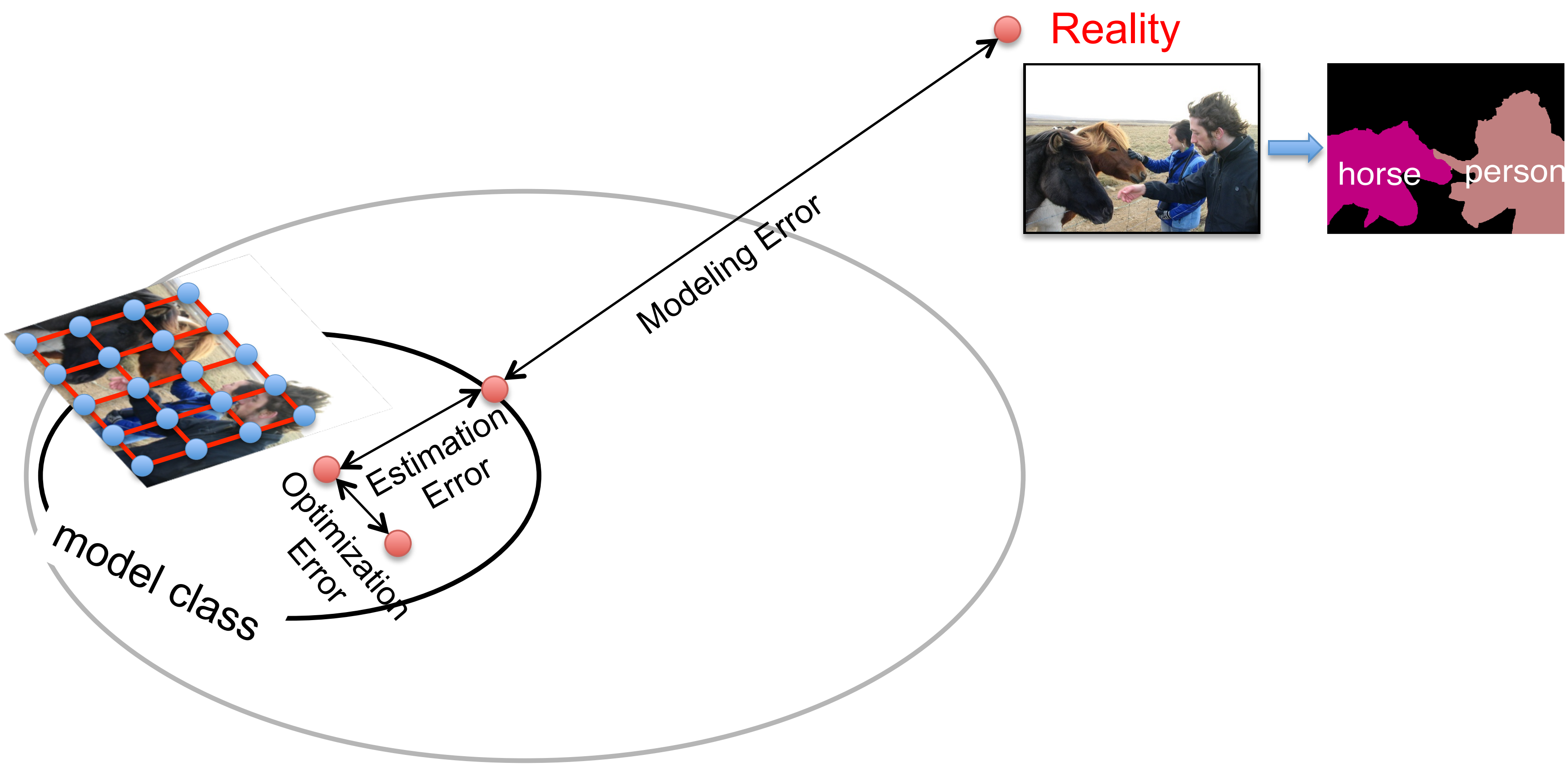
- **Model Selection**
  - Try different models. Picks the best one. (More on this later)

- If happy stop
  - Else refine one or more of the above

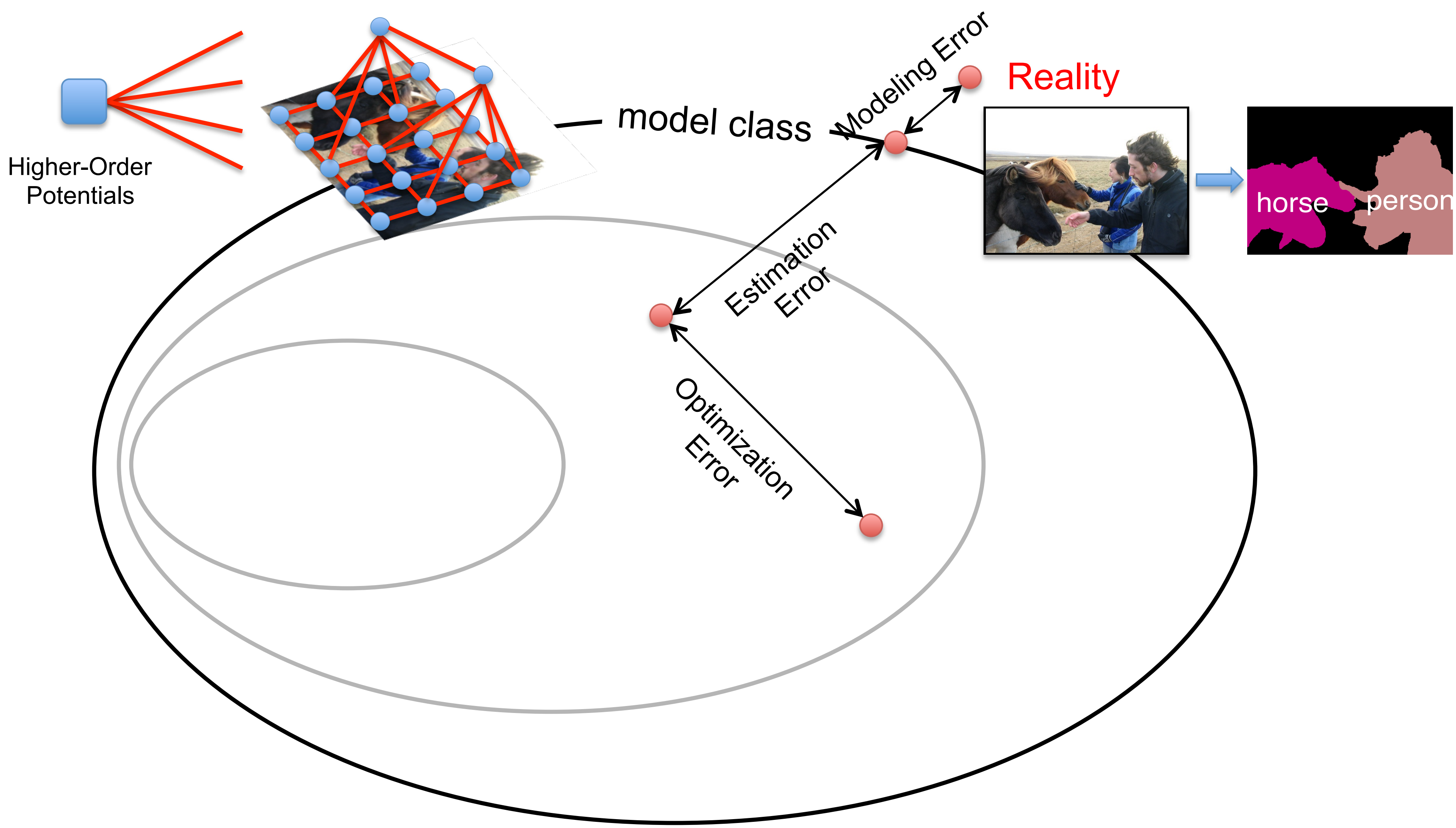
Error Decomposition



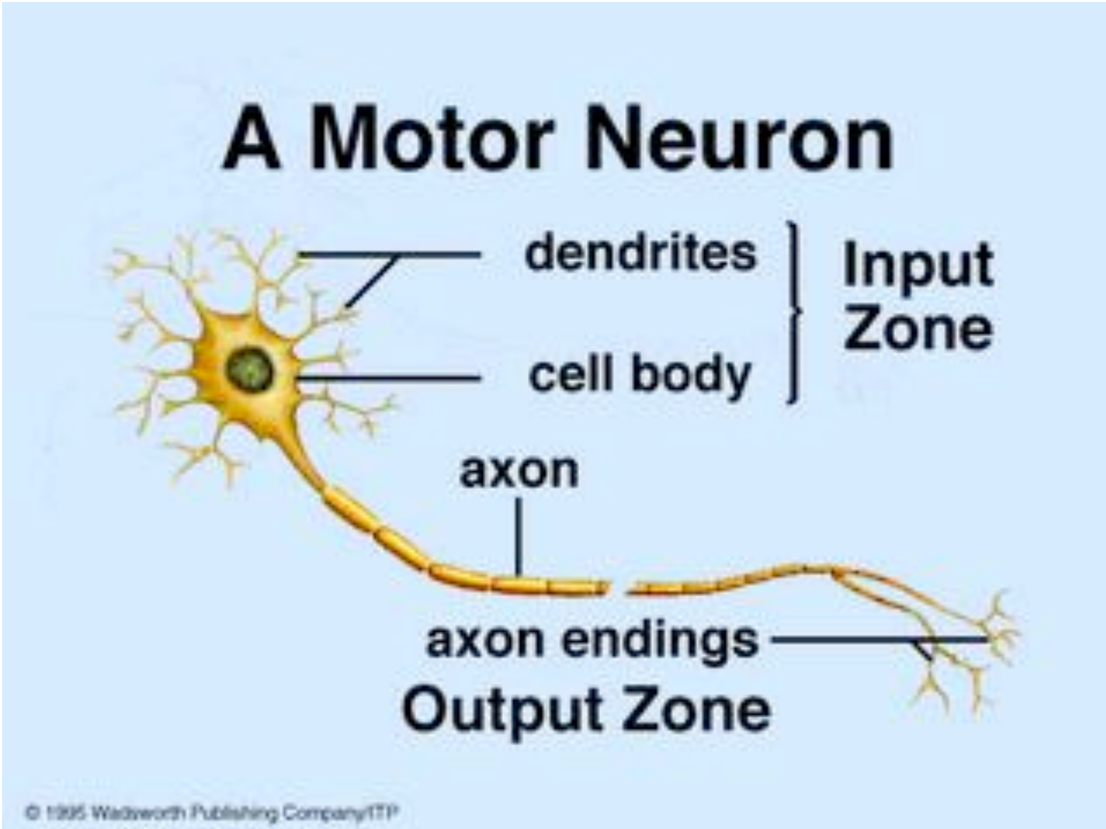
Error Decomposition



Error Decomposition

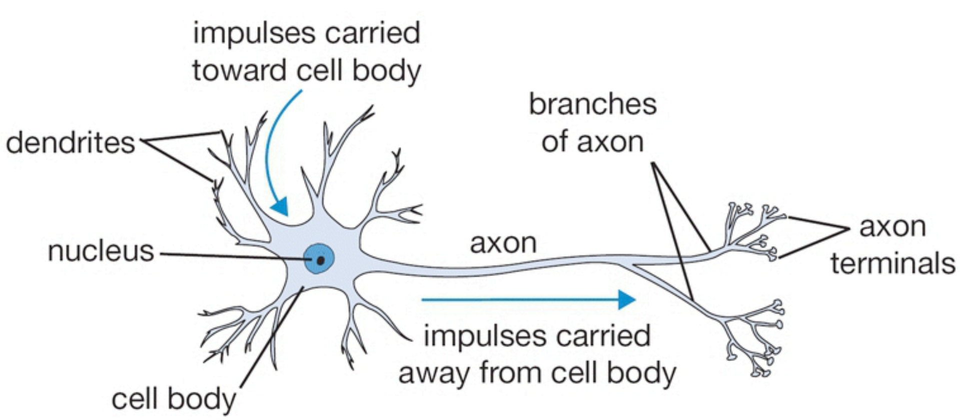


Biological Neuron



Recall: The Neuron Metaphor

- Neurons
    - accept information from multiple inputs,
    - transmit information to other neurons.
- Artificial neuron
    - Multiply inputs by weights along edges
    - Apply some function to the set of inputs at each node



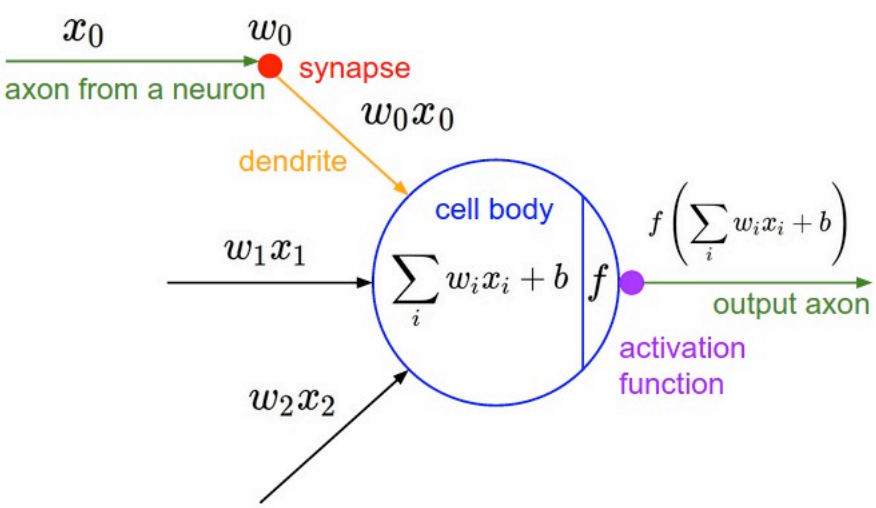
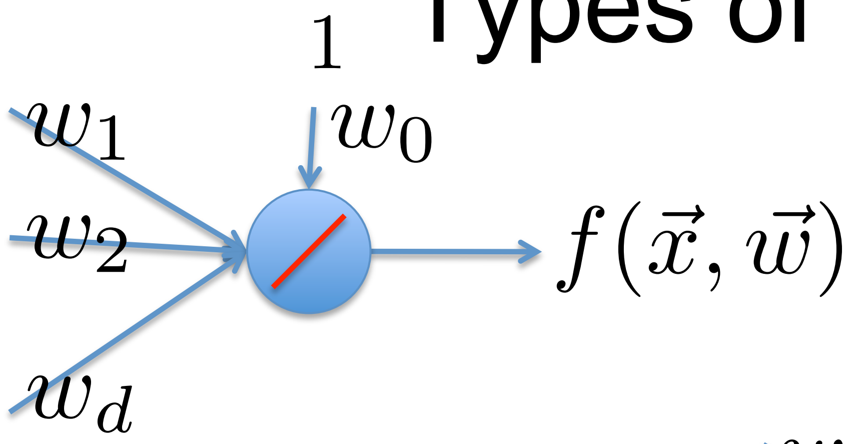
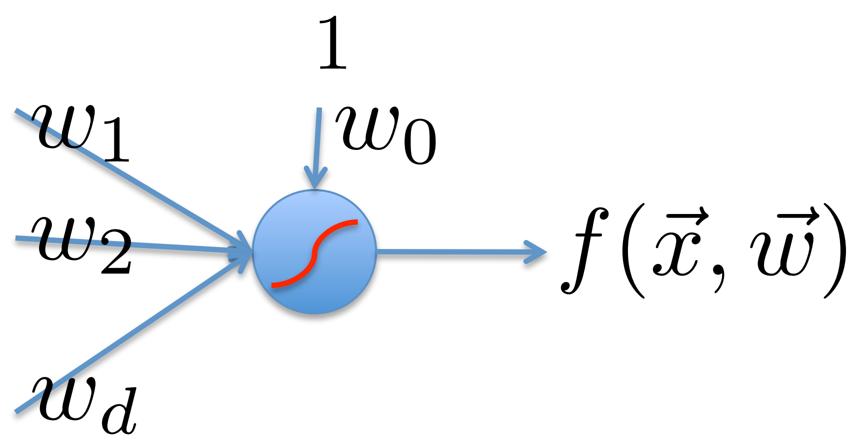


Image Credit: Andrej Karpathy, CS231n

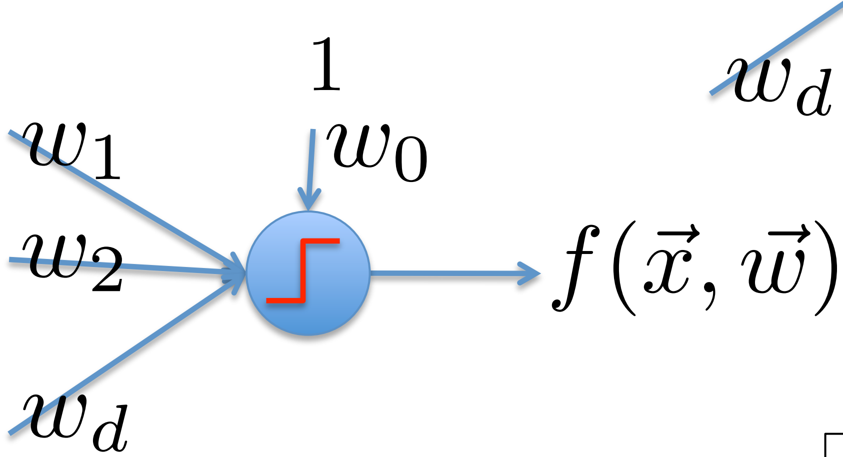
Types of Neurons



Linear Neuron



Logistic Neuron

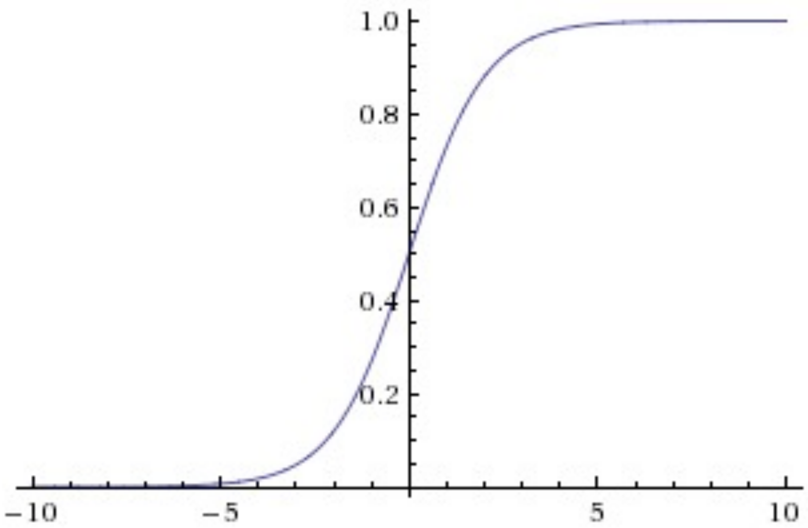


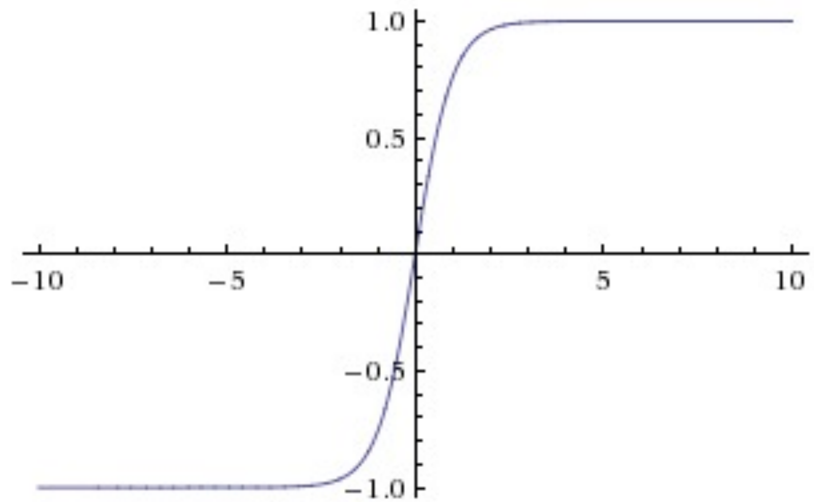
Perceptron

Potentially more. Require a convex
loss function for gradient descent training.

Activation Functions

- sigmoid vs tanh





A quick note

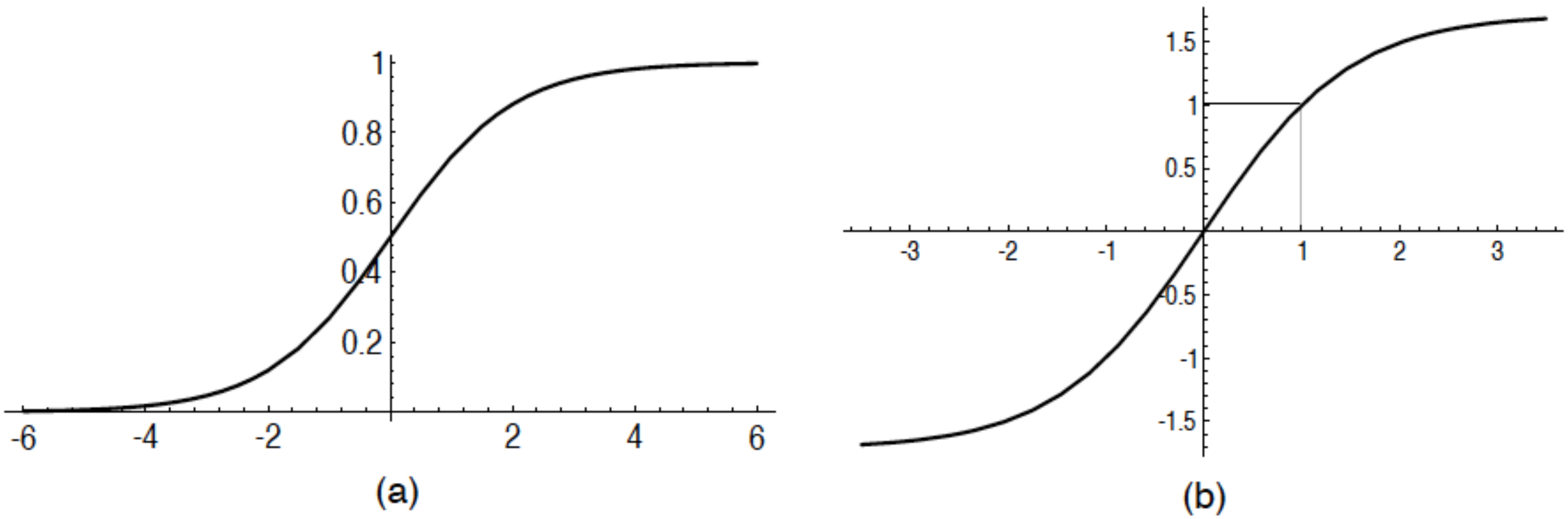
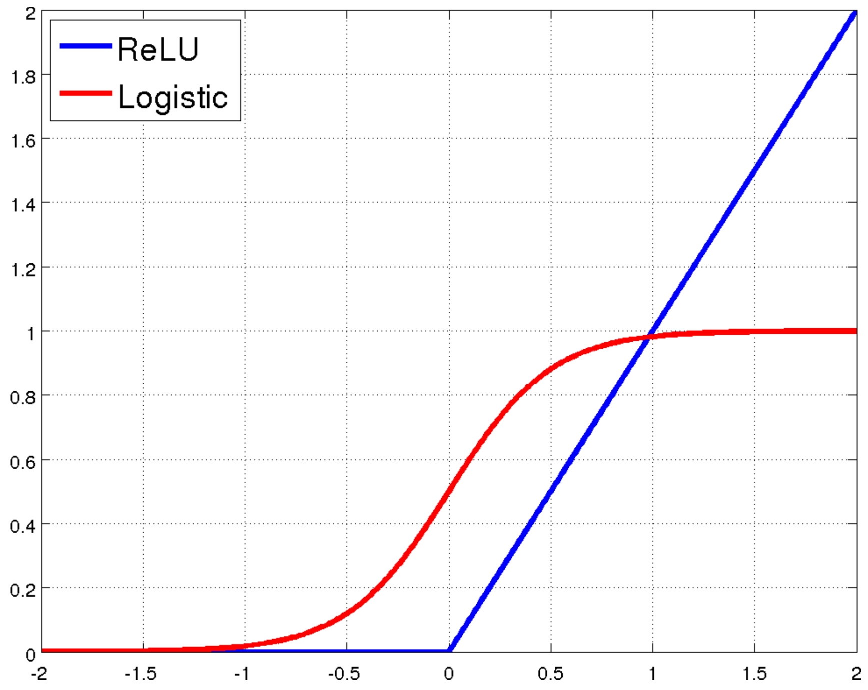
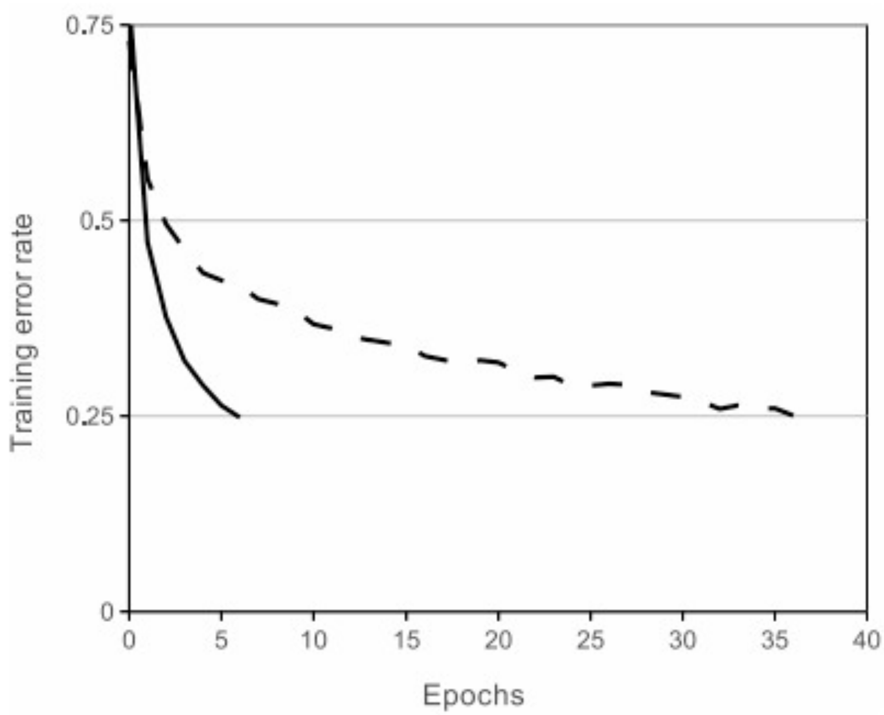


Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1 + e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159 \ \tanh\left(\frac{2}{3}x\right)$.

Rectified Linear Units (ReLU)





[Krizhevsky et al., NIPS12]

Limitation

- A single “neuron” is still a linear decision boundary
- What to do?
- Idea: Stack a bunch of them together!

Multilayer Networks

- Cascade Neurons together
- The output from one layer is the input to the next
- Each Layer has its own sets of weights

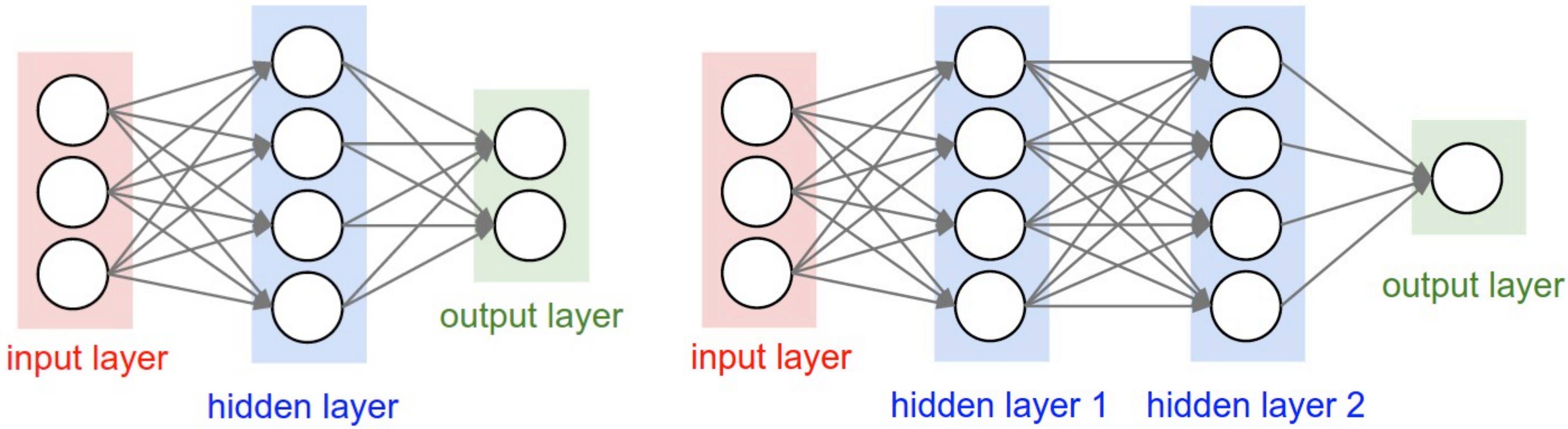


Image Credit: Andrej Karpathy, CS231n

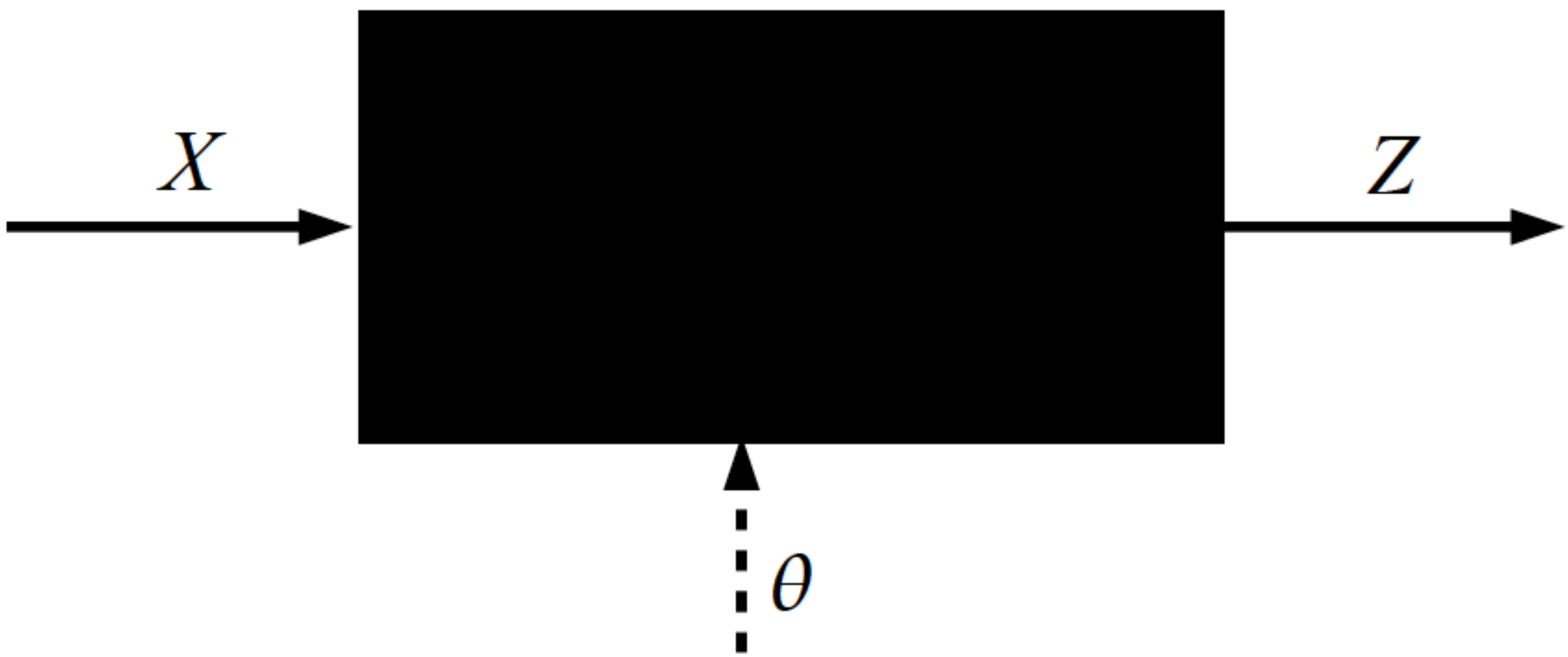
Universal Function Approximators

- Theorem
  - 3-layer network with linear outputs can uniformly approximate any continuous function to arbitrary accuracy, given enough hidden units [Funahashi ’89]

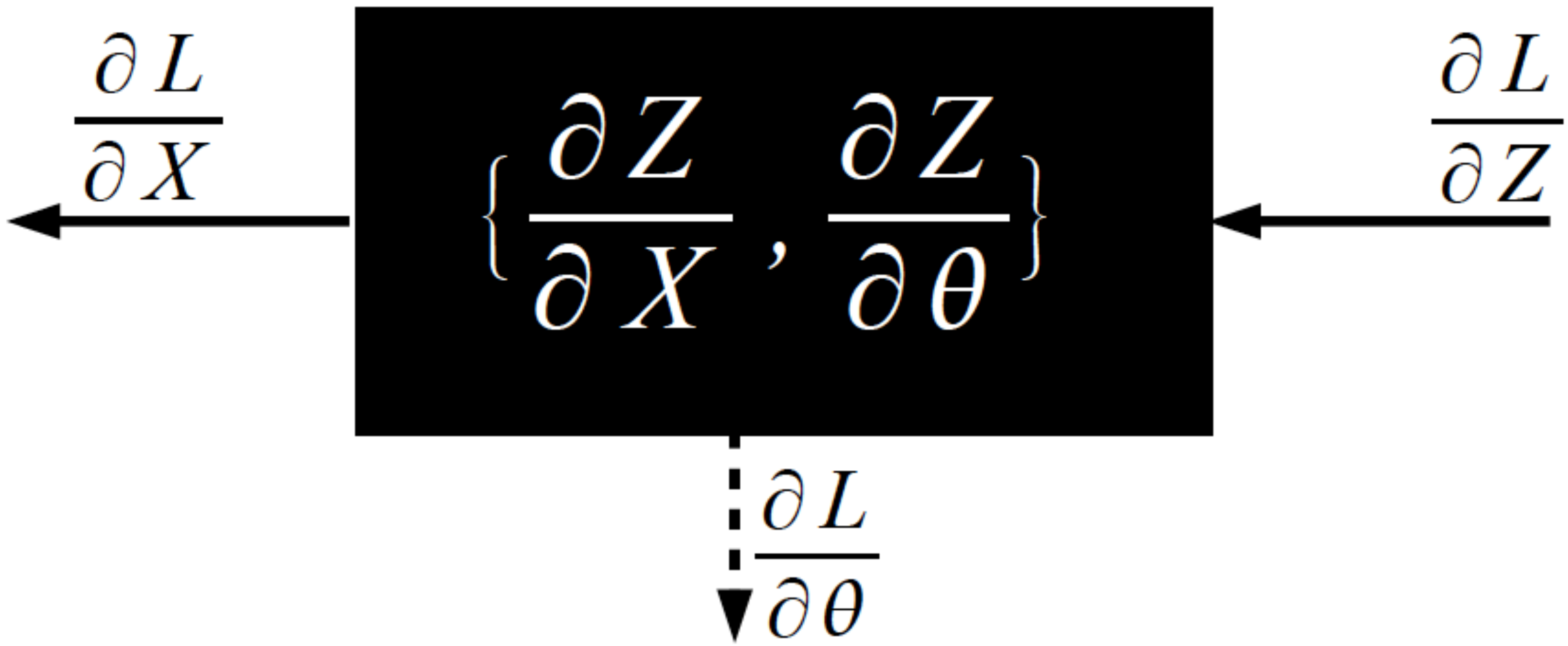
Neural Networks

- Demo
  - http://neuron.eng.wayne.edu/bpFunctionApprox/bpFunctionApprox.html

Key Computation: Forward-Prop



Key Computation: Back-Prop



Linear Classifier: SVM

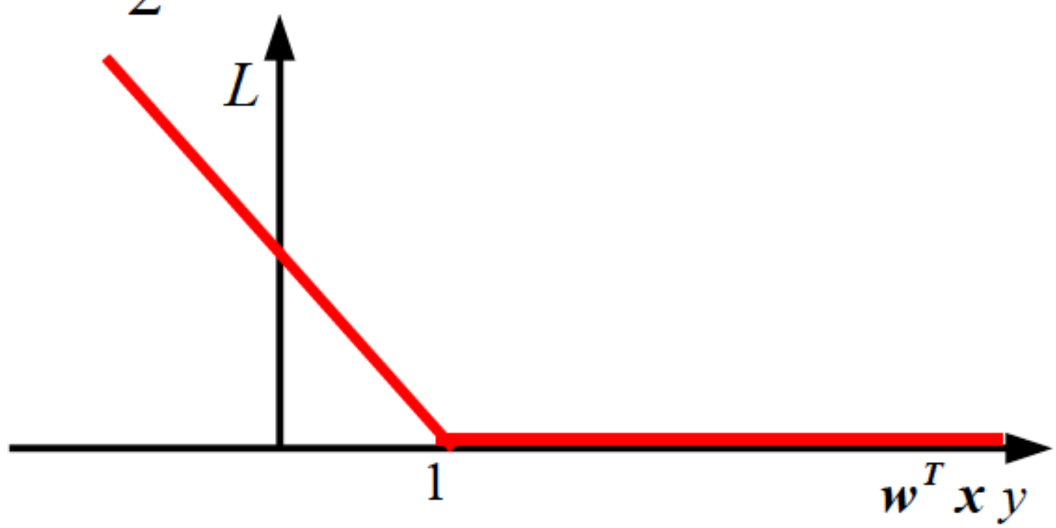
Input: $\boldsymbol{x} \in R^D$

Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $\boldsymbol{w}^T \boldsymbol{x}$

Loss: $L = \frac{1}{2} ||\boldsymbol{w}||^2 + \lambda \, max[0, 1 - \boldsymbol{w}^T \boldsymbol{x} \, y]$



Hinge Loss

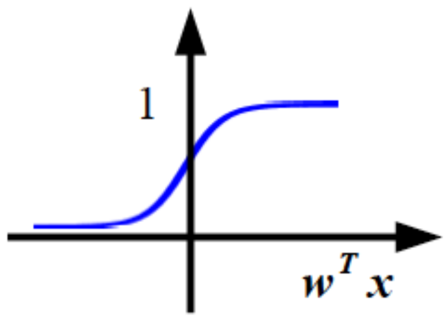
Linear Classifier: Logistic Regression

Input: $\boldsymbol{x} \in R^D$

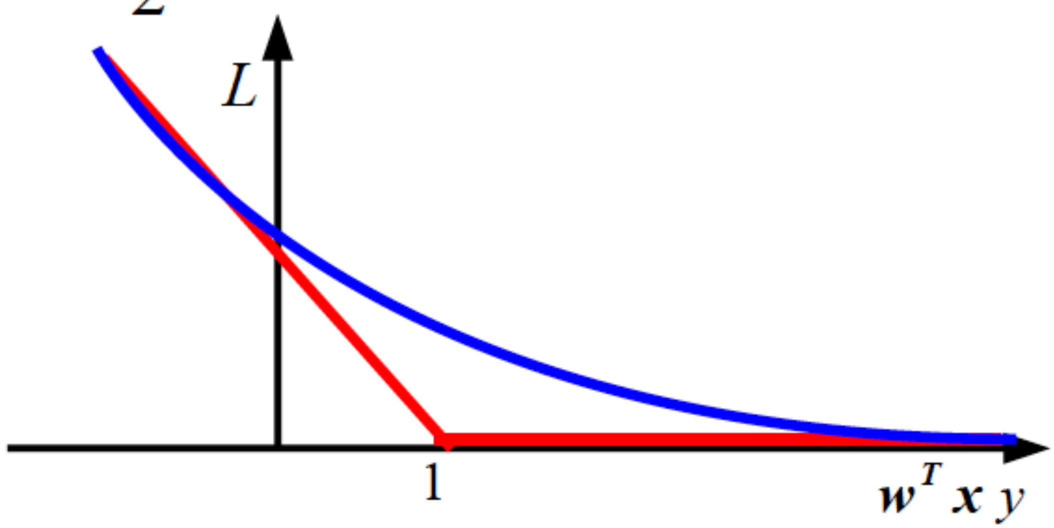
Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $p(y=1|\boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$



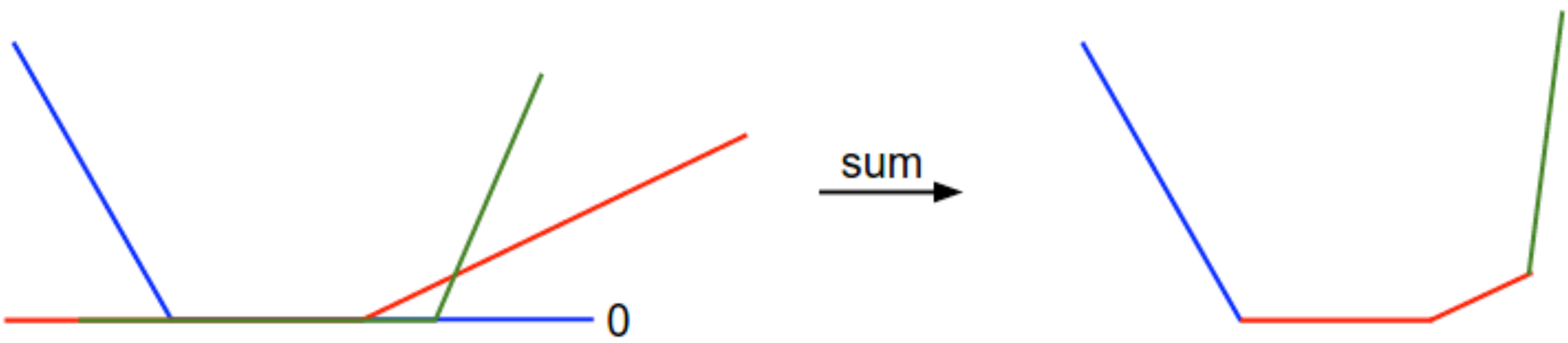
Loss: $L = \dfrac{1}{2}\|\boldsymbol{w}\|^2 - \lambda \log(p(y|\boldsymbol{x}))$



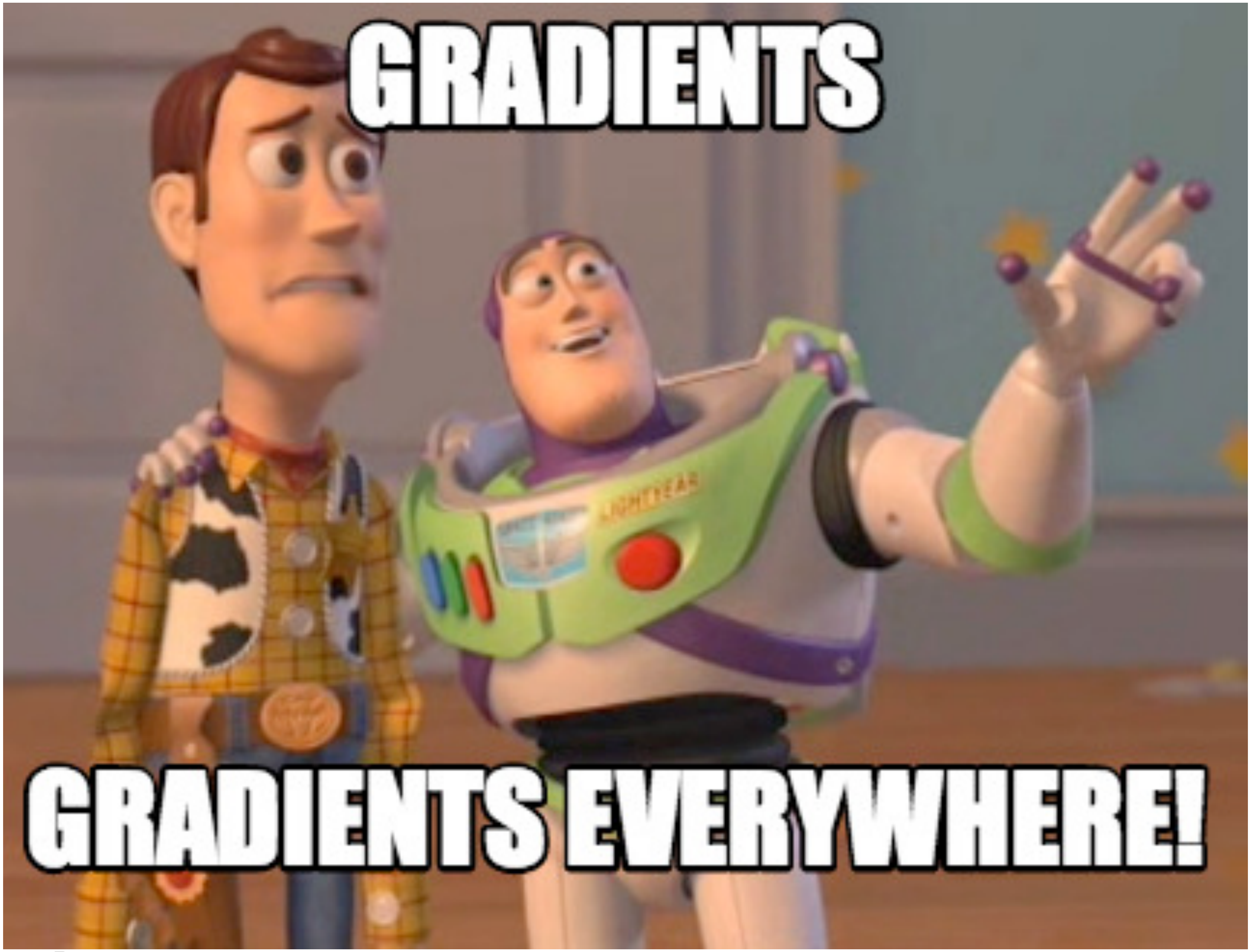
Log Loss

Visualizing Loss Functions

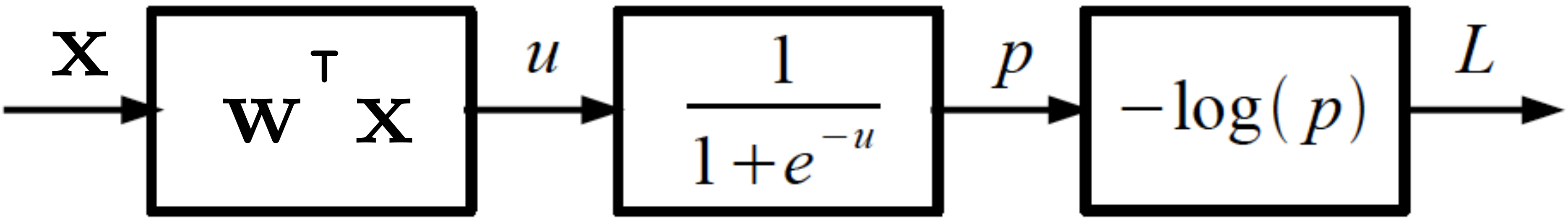
- Sum of individual losses

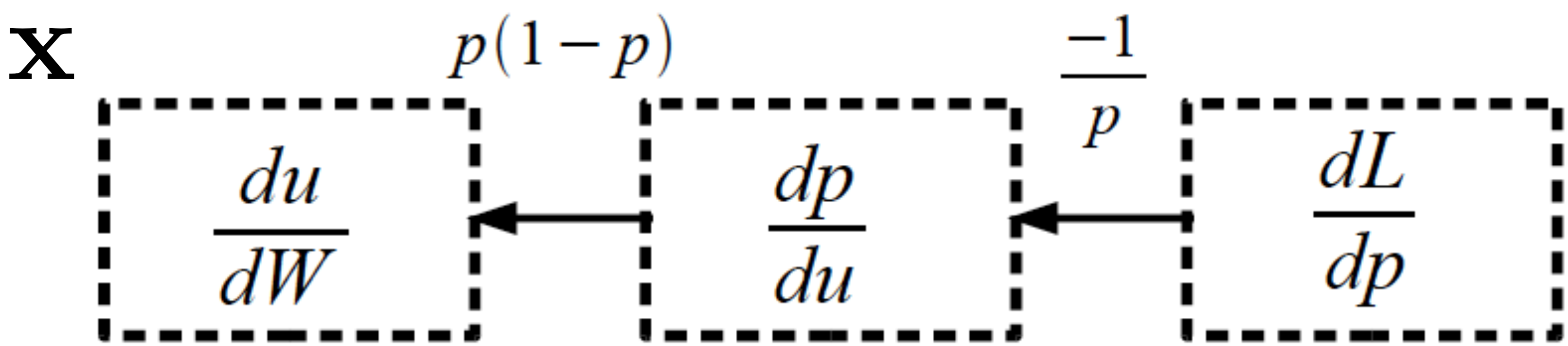


Detour



Logistic Regression as a Cascade





$$\frac{dL}{dW} = \frac{dL}{dp} \cdot \frac{dp}{du} \cdot \frac{du}{dW} = (p-1)\mathbf{x}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Forward Propagation

- On board