SAND2017-11179PE

Emily's favorite things about

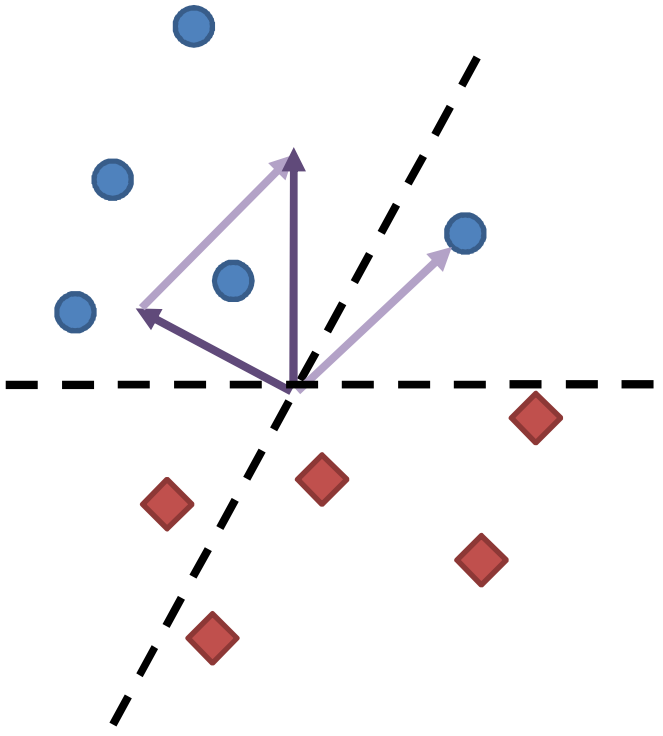
Convolutional Neural Networks

Emily Donahue

10/10/16

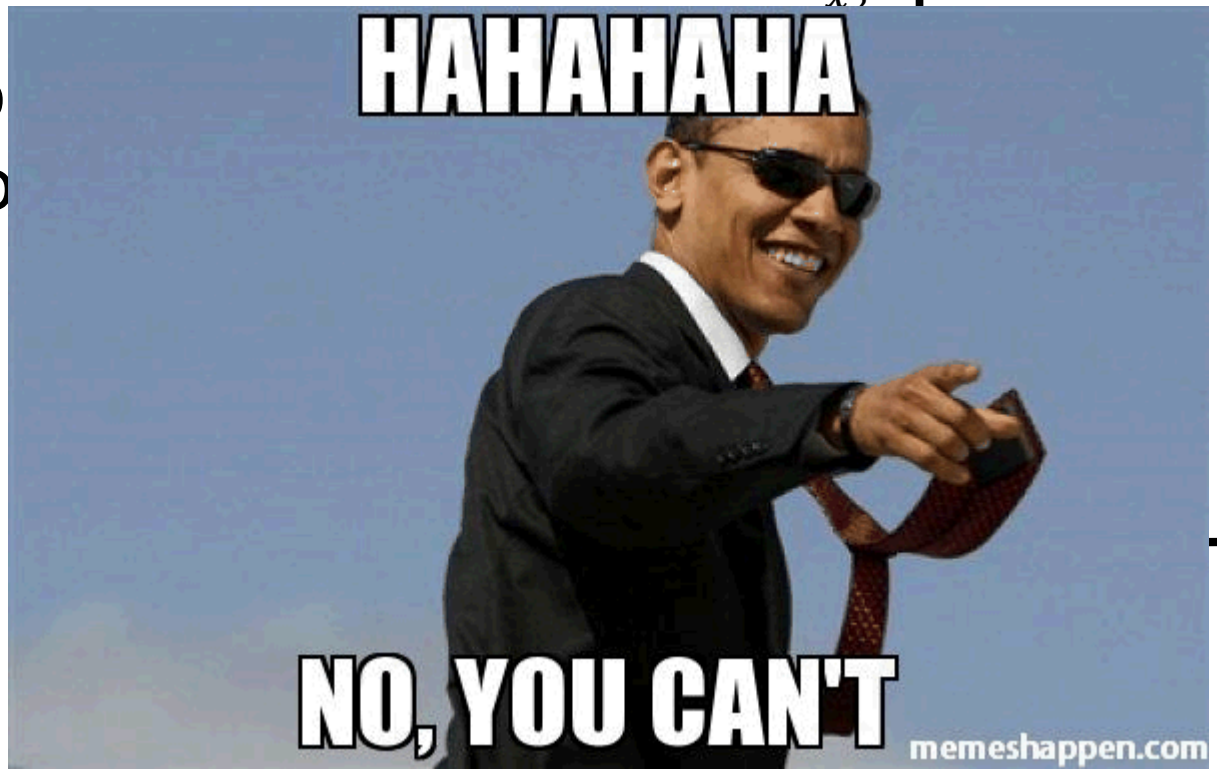
5-minute ML intro: The Perceptron

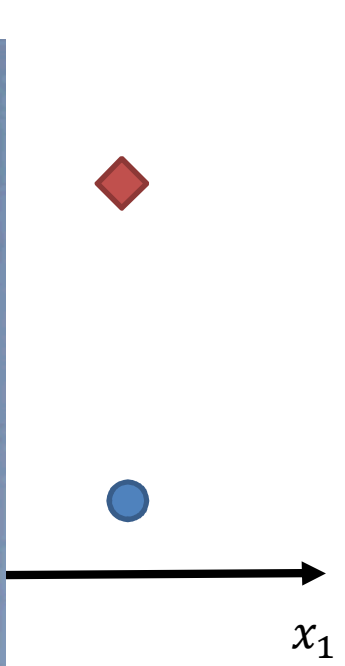
- Given a set of data points, how do we train an algorithm to classify them?
  - Assume data points are in 2d
    (e.g. $\overrightarrow{x_i} = (x_{i1}, x_{i2}) \in X_{data}$)
  - classify with weight vector:
    $y = sign(\vec{w}^T \vec{x})$
- Algorithm:
  - Initialize $\vec{w}$ with random values
  - While the Perceptron still misclassifies a point $\overrightarrow{x_j}$:
    - Update $\vec{w} = \vec{w} - y_j \overrightarrow{x_j}$
- Guaranteed convergence!



5-minute ML intro: XOR & the AI Winter

- How do
  the follo

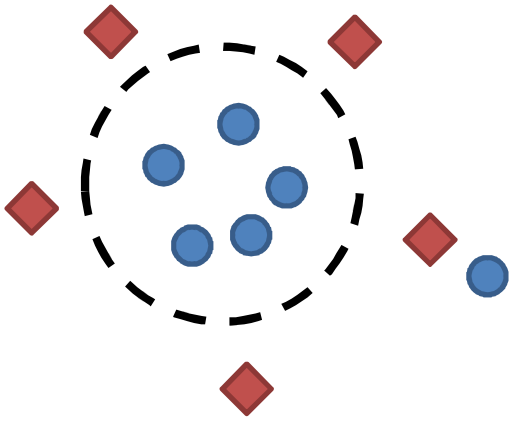




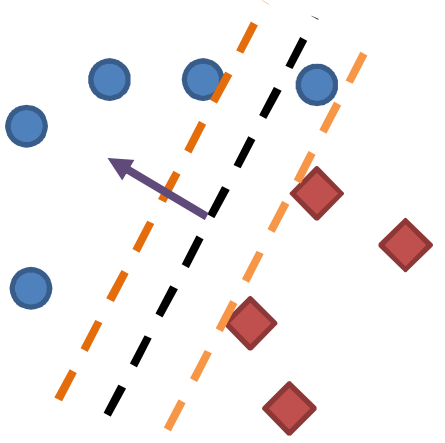
http://memeshappen.com/media/created/HAHAHAHA-No-you-Can39t--meme-8033.jpg

5-minute ML intro: Solutions to XOR

- Kernel methods, 1964, but not popularized until 2000's
  - Represent the data in a high-dimensional space where it is linearly separable



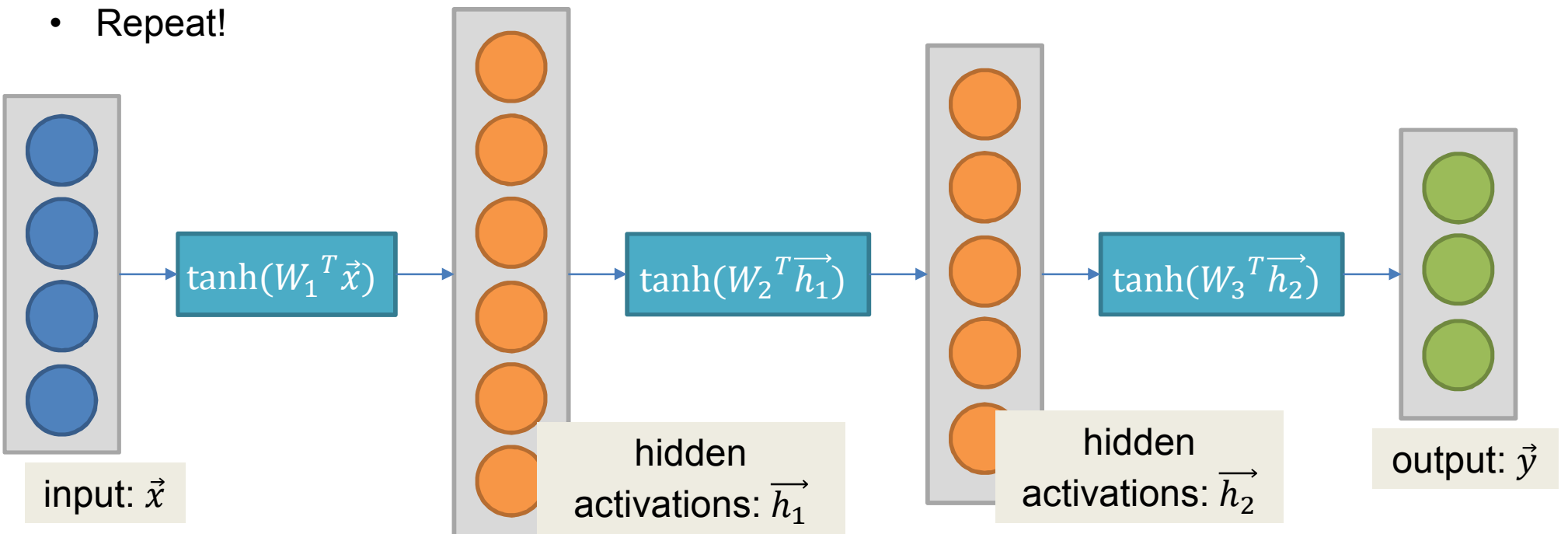
- Support Vector Machines (SVMs), 1993
  - Use a "soft margin" – i.e. accept some small errors in classification



- Multi-layered linear classifiers
  - Use one classifier, then apply a non-linear function ("activation" function)
  - Repeat…

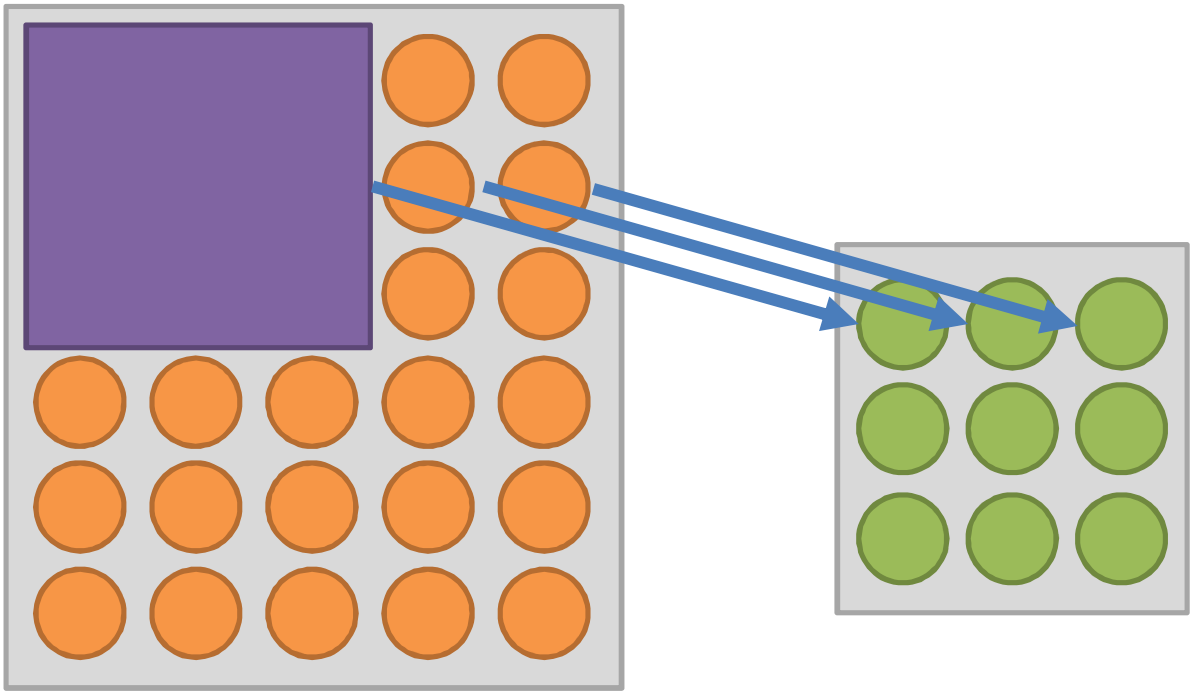
Neural Networks

- Multi-layered linear classifiers
  - One linear classifier, $\vec{w}^T \vec{x} = y$, produces one output
  - Produce multiple outputs with a matrix: $W\vec{x} = \vec{h}$
  - Apply a non-linear function to $\vec{h}$
    - e.g. tanh, sigmoid, ReLU
  - Repeat!



Convolutional Layers

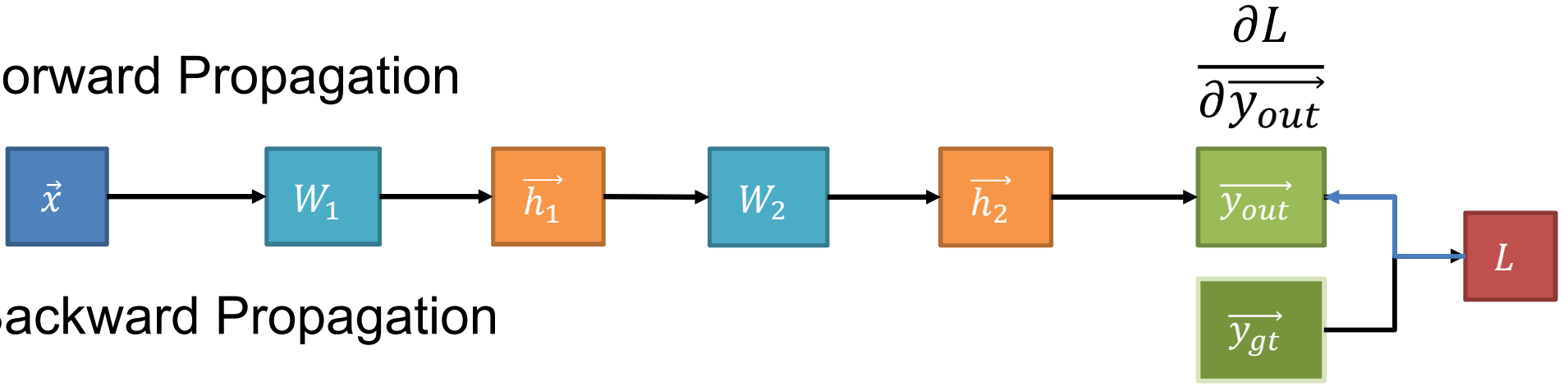
- The input, output, and hidden layers don't have to be 1-dimensional
- …and the layer functions don't have to be just matrix multiplications
- We use *convolutions* to exploit the spatial locality of features in the data
  - Use pooling afterwards to reduce data sizes



Training & Backprop

- Forward propagation: send input through a NN, compute objective
  - Loss function: objective function to minimize
  - e.g. Squared loss (or MSE)
- Backward propagation: send the gradient of the loss function back through the layers
  - update weights of matrices accordingly
  - start with the last layer, use chain rule for each previous layer

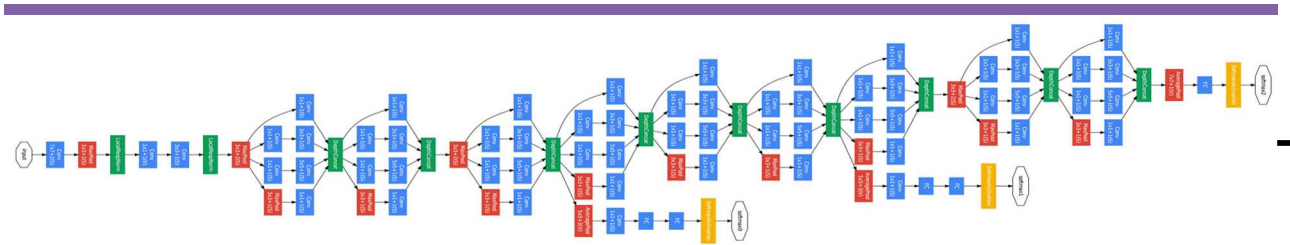
Forward Propagation



Backward Propagation

Convolutional Neural Networks

- How do we use computers to identify objects in images?
- Use Convolutional Neural Networks (CNNs)
  - Read images as matrices of numbers
  - Pass them through a series of functions (“layers”)
  - Train: Update parameters based on ground truth labels
  - Result is a trained image classifier



“dog”	“cat”	“boat”	“grass”	...
0.78	0.12	0.02	0.15	

“dog”

30-second corgi picture break

http://static.parade.com/wp-content/uploads/2016/01/Courtesy-of-eeveesevolution-on-tumblr-2.jpg

Questions?


Cool things (you might not know) about CNNs
https://i.imgur.com/ewEBLgA.jpg

Learning Style Designs

https://pdfs.semanticscholar.org/d8da/24a3496c6a902a05d866f3409c4de79e250e.pdf

Neural Nets as Artists

https://arxiv.org/pdf/1508.06576v2.pdf

DeepDream

Confusing Neural Networks

http://arxiv.org/pdf/1412.1897v2.pdf

Generative Art with MNIST

http://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/

Thanks!

- Acknowledgements:
  - David Wiegandt
    - Manager
  - Silpan Patel
    - Principal Investigator for Sasquatch
  - Jennifer Galasso
    - Mentor for the Caffe-OpenCL project
  - Kais Kudrolli
    - CCD intern and co-contributor

Sources

- Image credits:
  - http://www.nbb.cornell.edu/neurobio/ragusolab/images/cornell.gif
  - http://www.opencl.org/opencl_logo.jpg
  - https://developer.nvidia.com/sites/default/files/akamai/cuda/images/product_logos/NV_CUDA_wider.jpg
  - http://www.nvidia.com/docs/IO/67561/GeForce_GTX_280M_preview.jpg
  - http://3.bp.blogspot.com/-JqKMEYHEfjY/UfvZcNdwbII/AAAAAAAAM2c/JDPO2Uke1LU/s1600/ScreenLock.png
- References:
  - http://caffe.berkeleyvision.org/
  - https://www.khronos.org/opencl/