CS/ECE/ME 532 Period 22

Applications of artificial neural networks:

- natural language processing, translation, image processing, Alexa/Siri, self driving cars, …
- *Predicting age/gender/income from browsing behavior*

"Hey Siri"

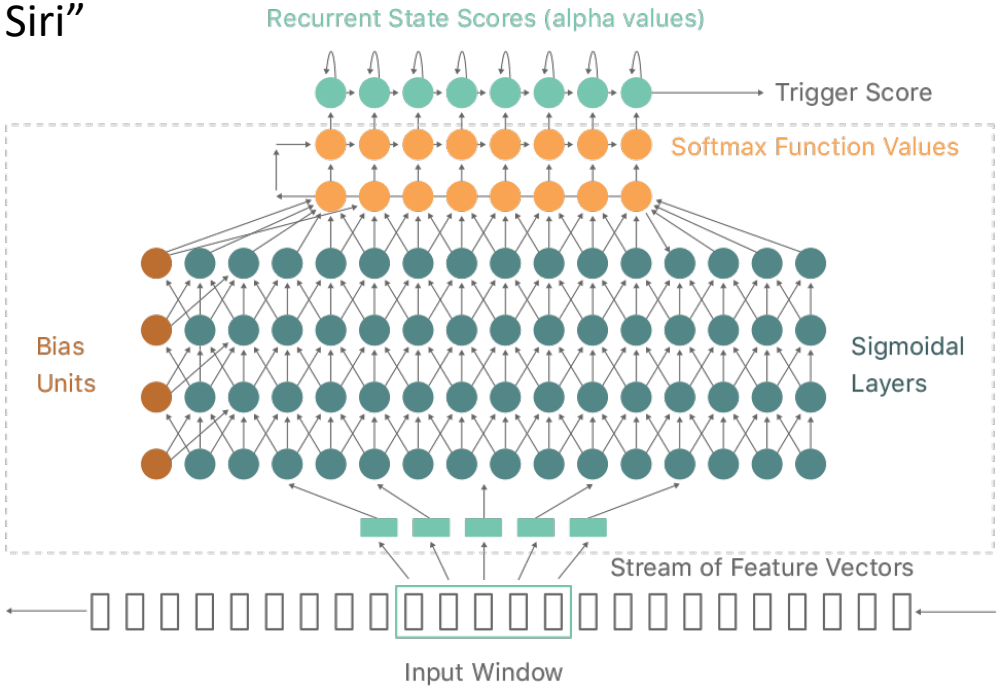


Figure 2. The Deep Neural Network used to detect "Hey Siri." The hidden layers are actually fully connected. The top layer performs temporal integration. The actual DNN is indicated by the dashed box.

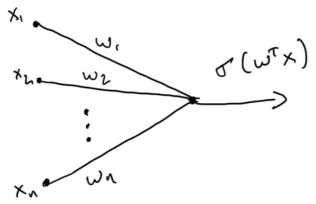
https://machinelearning.apple.com/2017/10/01/hey-siri.html

Neural Networks, SGD, backpropagation

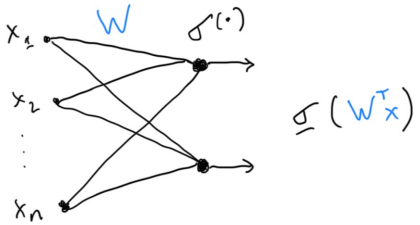
Linear regression: $\widehat{y} = \boldsymbol{x}^T \boldsymbol{w}$

Binary classification: $\widehat{y} = \text{sign}(\boldsymbol{x}^T \boldsymbol{w})$

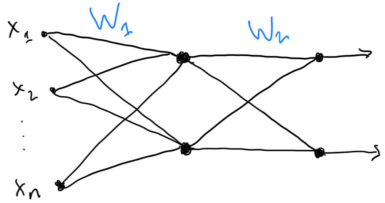
Single neuron: $\widehat{y} = \sigma(\boldsymbol{x}^T \boldsymbol{w})$



Multiple neurons, single layer: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}^T \boldsymbol{x})$



Two layers: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}_2^T \boldsymbol{\sigma}(\boldsymbol{W}_1^T \boldsymbol{x}))$



n-layers: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}_n^T \dots \boldsymbol{\sigma}(\boldsymbol{W}_2^T \boldsymbol{\sigma}(\boldsymbol{W}_1^T \boldsymbol{x})) \dots)$

Example Application

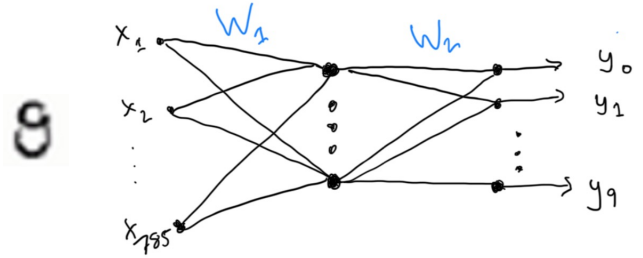
MNIST database: recognizing handwritten digits

- Input: 28x28 grayscale image, stacked as vector

$$\boldsymbol{x} \in \mathbb{R}^{785}$$

- Output: likelihood of being a 0, likelihood of being a 1, …

$$\widehat{\boldsymbol{y}} \in [0,1]^{10}$$



$$\boldsymbol{y} = \begin{bmatrix} 0.05 \\ 0.01 \\ \vdots \\ 0.95 \\ 0.02 \end{bmatrix}$$

Training

$$\min_{\boldsymbol{w}} \ell(\boldsymbol{w})$$

$$\min_{\boldsymbol{w}} \sum_i (\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2$$

$$\min \sum_i (\boldsymbol{\sigma}(\boldsymbol{W}_n^T \dots \boldsymbol{\sigma}(\boldsymbol{W}_2^T \boldsymbol{\sigma}(\boldsymbol{W}_1^T \boldsymbol{x}_i)) \dots) - \boldsymbol{y}_i)^2$$

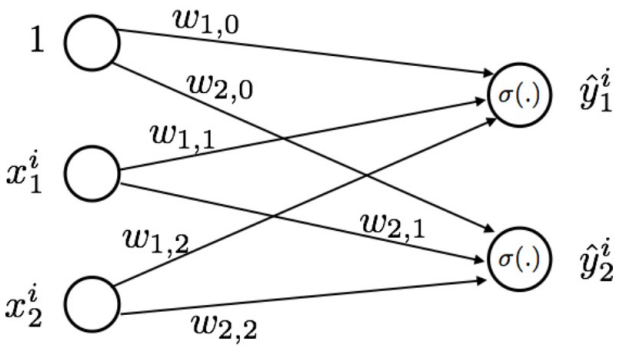
not convex, but we can still use gradient descent to find good W

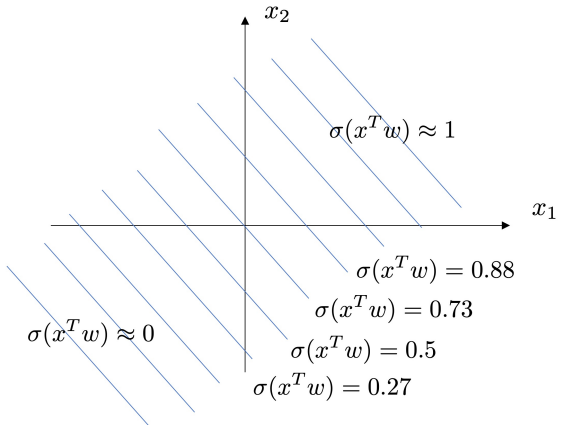
Two tricks:

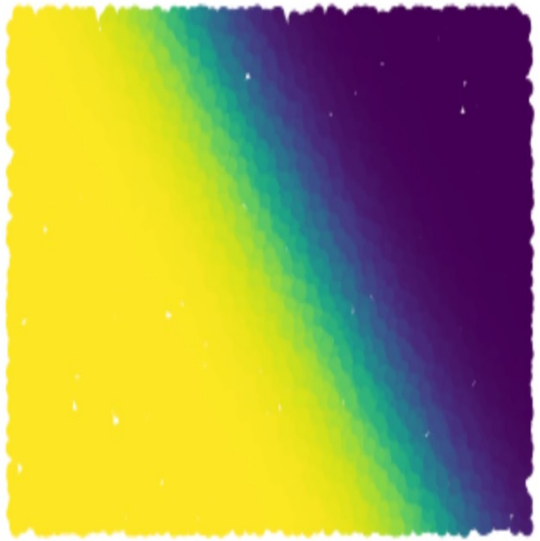
1) Stochastic GD
2) Backpropagation

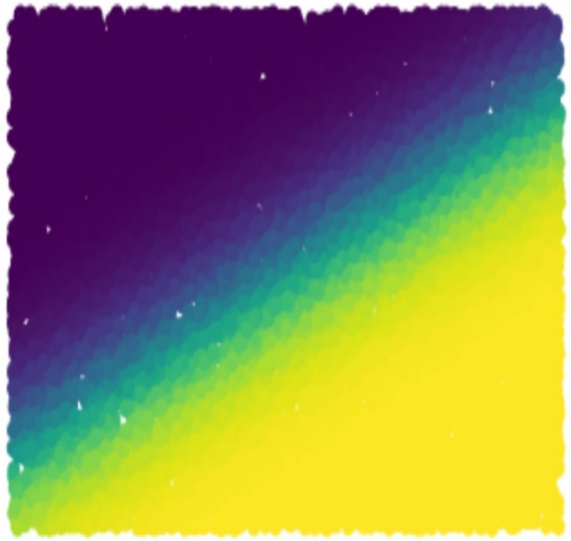
Derivatives for weights of each layer have simple expression (logistic activation function)

Q1

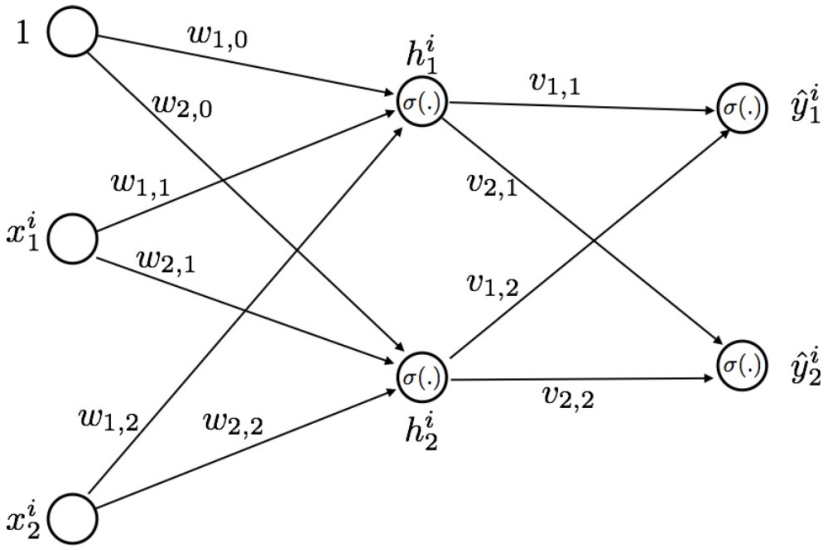

1
x_1^i
x_2^i
$w_{1,0}$
$w_{2,0}$
$w_{1,1}$
$w_{2,1}$
$w_{1,2}$
$w_{2,2}$
$\sigma(.)$
$\sigma(.)$
$\hat{y}_1^i$
$\hat{y}_2^i$


x_2
x_1
$\sigma(x^T w) \approx 1$
$\sigma(x^T w) = 0.88$
$\sigma(x^T w) = 0.73$
$\sigma(x^T w) = 0.5$
$\sigma(x^T w) = 0.27$
$\sigma(x^T w) \approx 0$





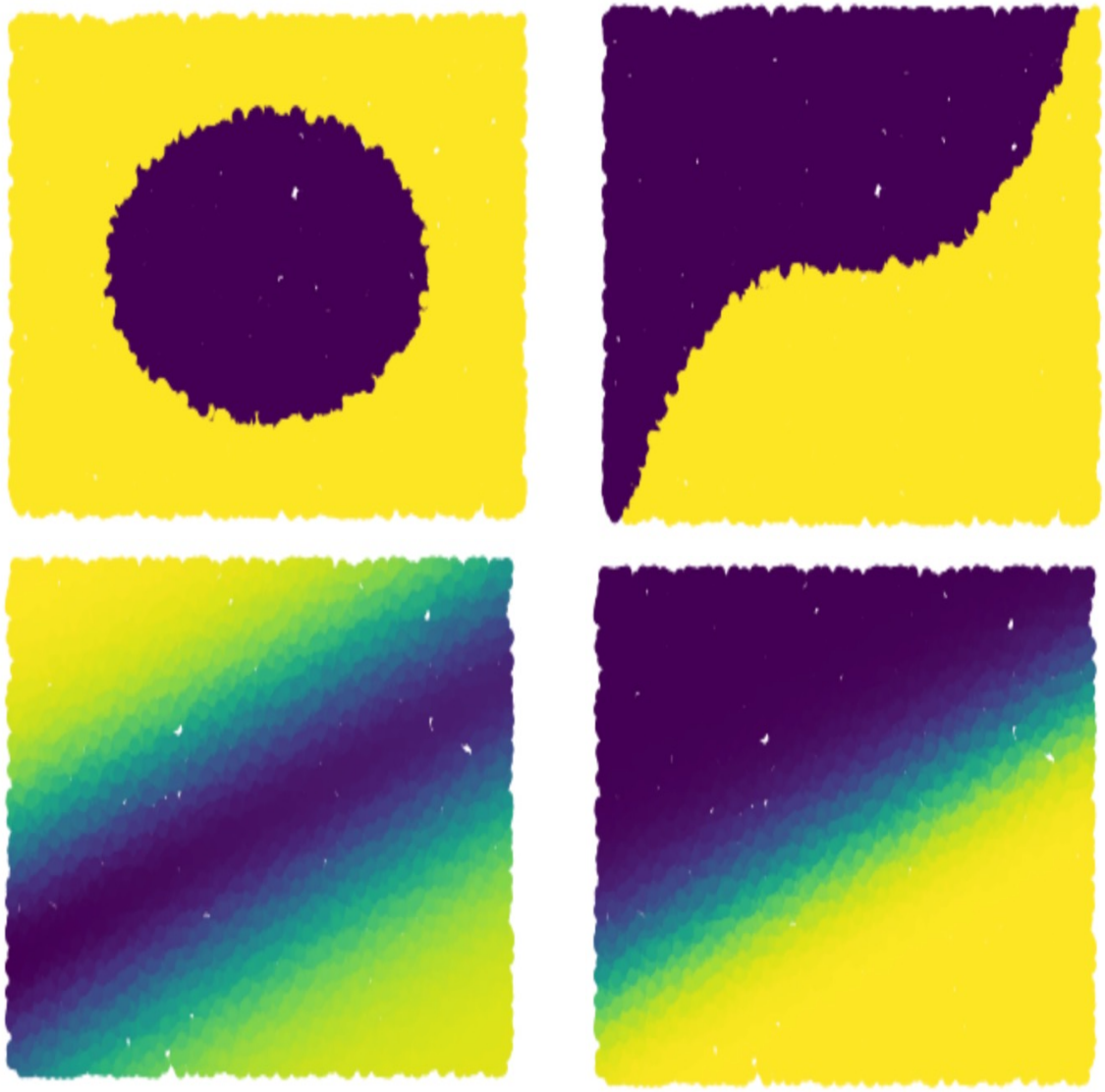
Q2



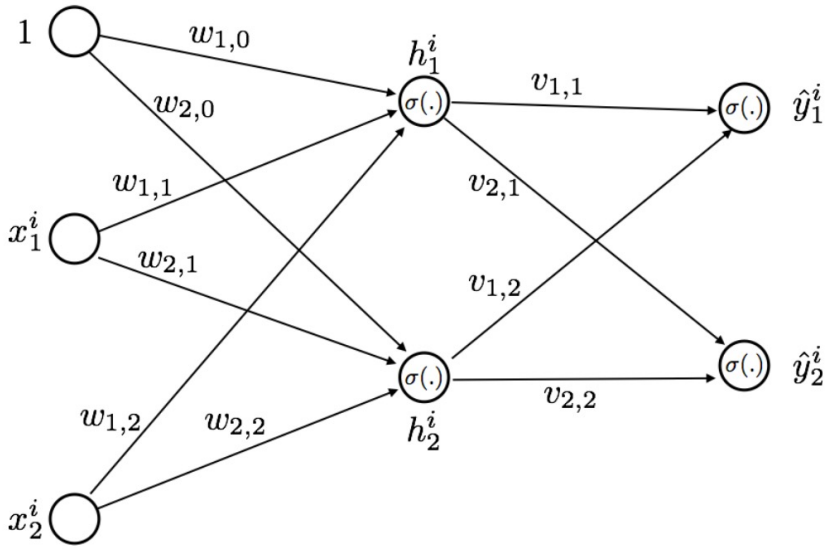
Gradient computation is complicated
$\Rightarrow$ Backpropagation

(we implemented it for you…)

M = 2
& 10 epochs

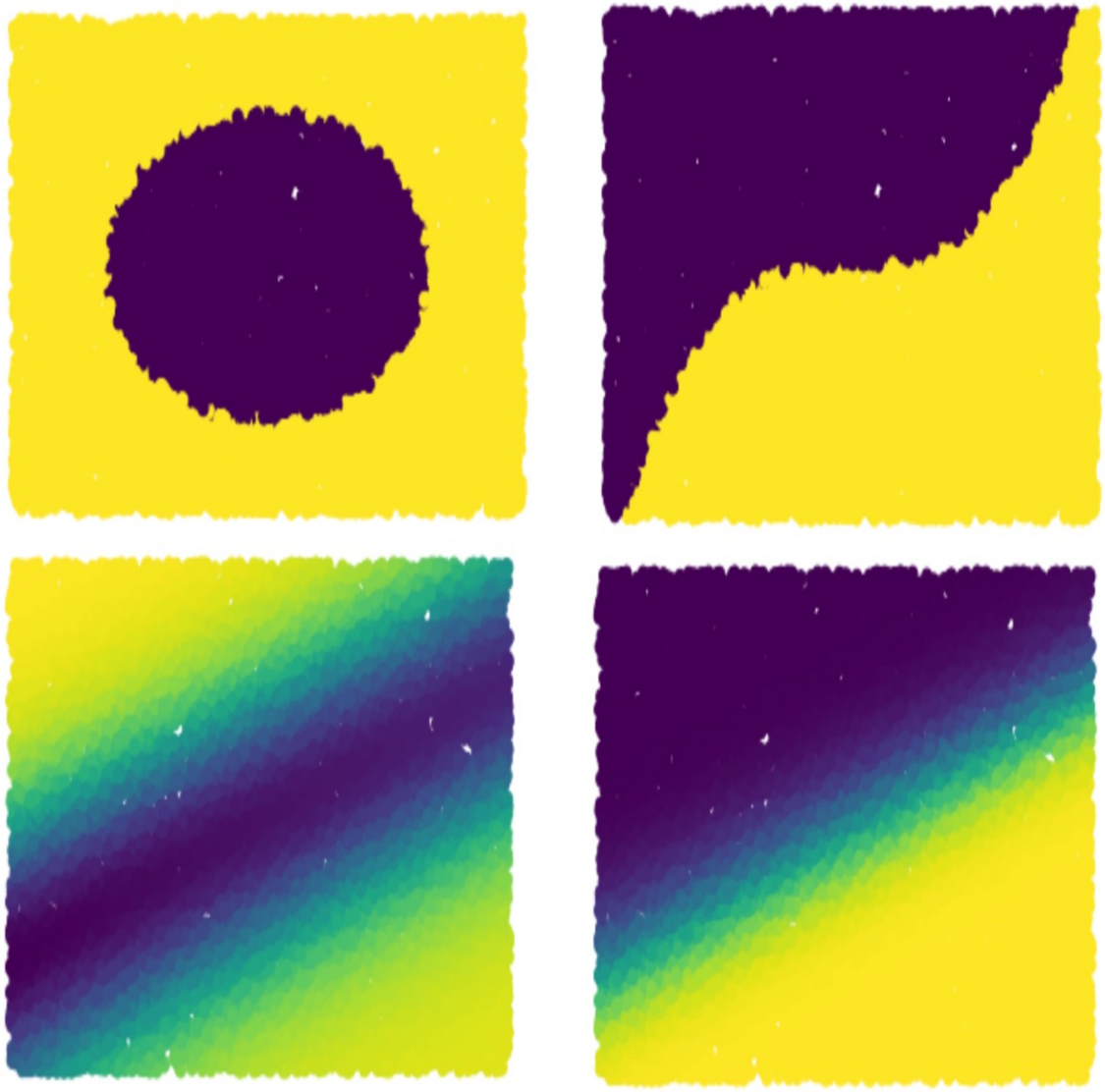


Q2

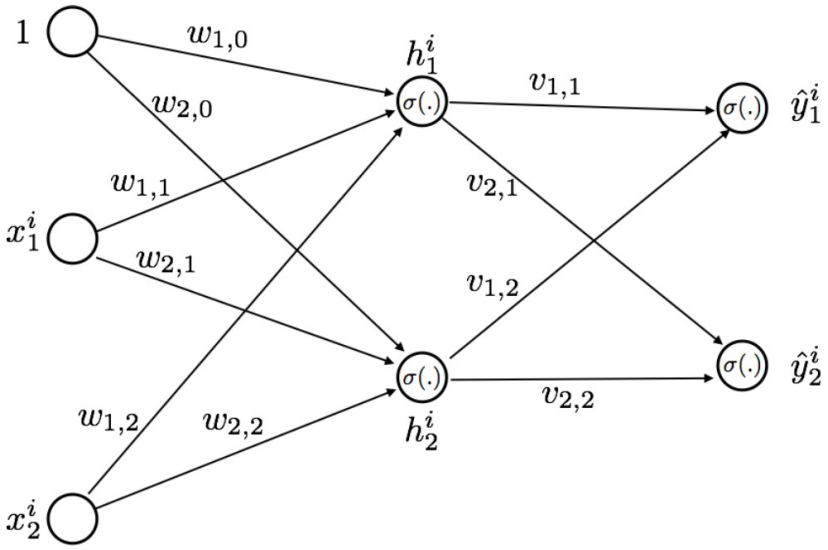


M = 3

& 10 epochs



Q2



M = 4

& 10 epochs