

ENGR-E 533

Deep Learning Systems

Module 01

Shallow Neural Network Basics

Minje Kim

Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: <http://minjekim.com>

Research Group: <http://saige.sice.indiana.edu>

Meeting Request: <http://doodle.com/minje>



INDIANA UNIVERSITY

**SCHOOL OF INFORMATICS,
COMPUTING, AND ENGINEERING**

The Last Layer

- Function approximation

- Don't worry, we're going to cover the first layer, too
 - And the layers in the middle
- But, why the last layer first?
 - Everything we do with neural networks is about approximating a function

The output of the function $\in \mathbb{R}^{K \times 1}$ $\rightarrow \mathbf{y} = \mathcal{F}(\mathbf{x})$ Your observed data sample $\in \mathbb{R}^{D \times 1}$

The mapping function you want to know (but you can never know its exact form)

The predicted output $\rightarrow \hat{\mathbf{y}} = \mathcal{G}(\mathbf{x}; \mathbb{W})$ Parameters

The estimate of the mapping function

Error function of your choice

- The objective function? $\arg \min_{\mathcal{G}} \mathcal{D}(\mathbf{y} || \hat{\mathbf{y}})$
 - What's wrong with it?
- It's easier to work with a parametric function $\arg \min_{\mathbb{W}} \mathcal{D}(\mathbf{y} || \mathcal{G}(\mathbf{x}; \mathbb{W}))$
- The actual optimization is done to reduce the sum of all error network $\arg \min_{\mathbb{W}} \sum_t \mathcal{D}(Y_{:,t} || \mathcal{G}(X_{:,t}; \mathbb{W}))$



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Function approximation

- Today we focus on the function approximation
 - Forget about neural networks just for now
- What I want to show you is
 - We can solve this potentially nonlinear function approximation problem linearly
 - In a certain condition
 - We'll transform the raw input data into feature space
- What kind of functions?
 - Basically any kind of functions
 - Scalar-to-scalar mapping
 - Vector-to-scalar mapping
 - Vector-to-[bounded scalar] mapping
 - Vector-to-vector mapping
 - Vector-to-[bounded vector] mapping
- Hold on.. what about differentiation, optimization, backpropagation, etc?
 - Forget about them
 - For now I want to give you an intuition that things are about template matching



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #1: scalar-to-scalar mapping, a linear approximation

- Suppose the mapping function between scalar variables x and y is defined by

$$y = \mathcal{F}(x) = \sin(x)$$

- And we don't know it

- What happens if we try to approximate it linearly?

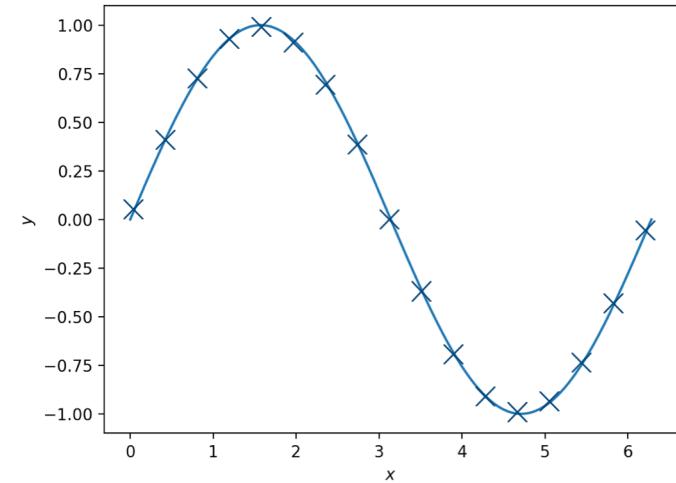
$$\hat{y}_t = \mathcal{G}(x_t; \mathbb{W}) = a_1 x_t + a_0$$

$$\mathbb{W} = \mathbf{a} = [a_1, a_0]^\top$$

$$\hat{y}_t = [a_1, a_0][x_t, 1]^\top$$

- This relationship should hold for all pairs of input and output samples

$$[\hat{y}_0 \ \hat{y}_1 \ \hat{y}_2 \ \cdots \ \hat{y}_{T-1}] = [a_1 \ a_0] \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{T-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$
$$\Leftrightarrow \hat{\mathbf{y}}^\top = \mathbf{a}^\top \mathbf{X}$$



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #1: scalar-to-scalar mapping, a linear approximation

$$\hat{\mathbf{y}}^\top = \mathbf{a}^\top \mathbf{X}$$

- What are we going to do with this?

$$\begin{aligned}\arg \min_{\mathbb{W}=\mathbf{a}} \sum_t \mathcal{D}(\hat{y}_t || y_t) &= \arg \min_{\mathbf{a}} \sum_t (\hat{y}_t - y_t)^2 = (\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}}) \\ &= \arg \min_{\mathbf{a}} (\mathbf{y}^\top - \mathbf{a}^\top \mathbf{X})(\mathbf{y}^\top - \mathbf{a}^\top \mathbf{X})^\top = \arg \min_{\mathbf{a}} (\mathbf{y}^\top - \mathbf{a}^\top \mathbf{X})(\mathbf{y} - \mathbf{X}^\top \mathbf{a}) \\ &= \arg \min_{\mathbf{a}} (\mathbf{y}^\top \mathbf{y} + \mathbf{a}^\top \mathbf{X} \mathbf{X}^\top \mathbf{a} - 2\mathbf{y}^\top \mathbf{X}^\top \mathbf{a})\end{aligned}$$

- Then what?

$$\begin{aligned}\frac{\partial \mathbf{y}^\top \mathbf{y} + \mathbf{a}^\top \mathbf{X} \mathbf{X}^\top \mathbf{a} - 2\mathbf{y}^\top \mathbf{X}^\top \mathbf{a}}{\partial \mathbf{a}} &= 2\mathbf{X} \mathbf{X}^\top \mathbf{a} - 2\mathbf{X} \mathbf{y}^\top = 0 \\ \mathbf{a} &= (\mathbf{X} \mathbf{X}^\top)^{-1} \mathbf{X} \mathbf{y}^\top\end{aligned}$$

- How can I be sure that this is the solution?
 - The objective function is quadratic
- We've got a closed form solution for this linear model
 - We don't need an iterative optimization algorithm

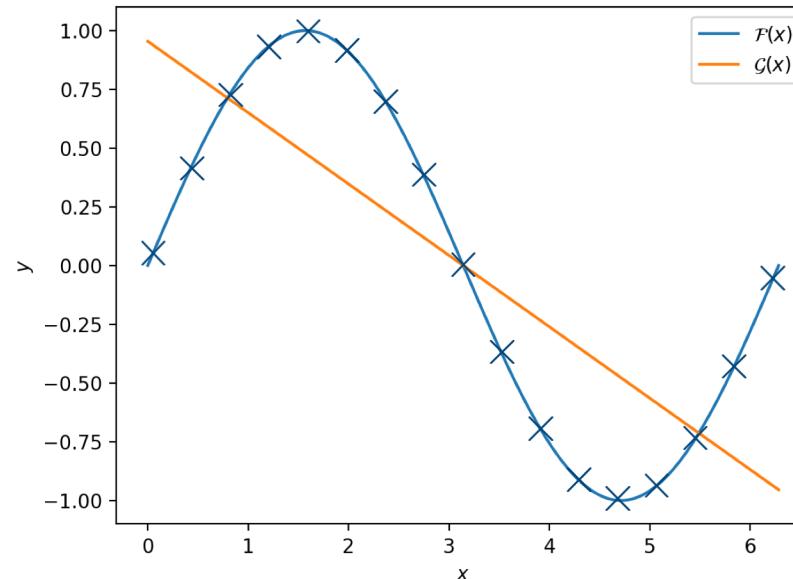


INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #1: scalar-to-scalar mapping, a linear approximation
 - Life is not that simple..



$$\mathcal{G}(x_t) = -0.30x_t + 0.95$$



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #2: scalar-to-scalar mapping using nonlinear feature transform

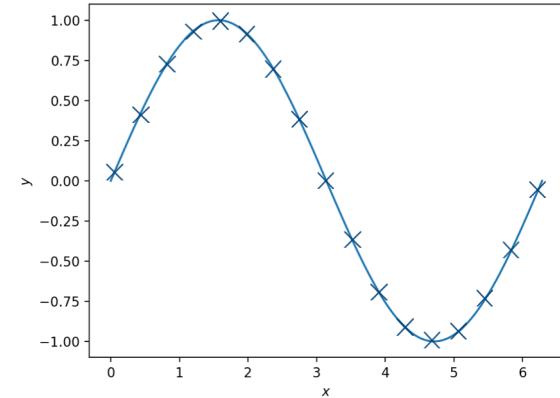
- It's meaningless to model a nonlinear function linearly
- Let's think of a nonlinear transformation of the input
 - So that the mapping between the features and the output can be linear
 - Any idea?
 - There are many different ways..
- I've got a feeling that this function looks like a cubic function
 - Because I was lucky to be able to visualize the data points
 - So, let's try a cubic transformation

$$[\hat{y}_0 \ \hat{y}_1 \ \hat{y}_2 \ \cdots \ \hat{y}_{T-1}] = [a_3 \ a_2 \ a_1 \ a_0] \begin{bmatrix} x_0^3 & x_1^3 & x_2^3 & \cdots & x_{T-1}^3 \\ x_0^2 & x_1^2 & x_2^2 & \cdots & x_{T-1}^2 \\ x_0 & x_1 & x_2 & \cdots & x_{T-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$
$$\Leftrightarrow \hat{\mathbf{y}}^\top = \mathbf{a}^\top \mathbf{X}$$

- How do we solve this?

- Same as before

$$\mathbf{a} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$$



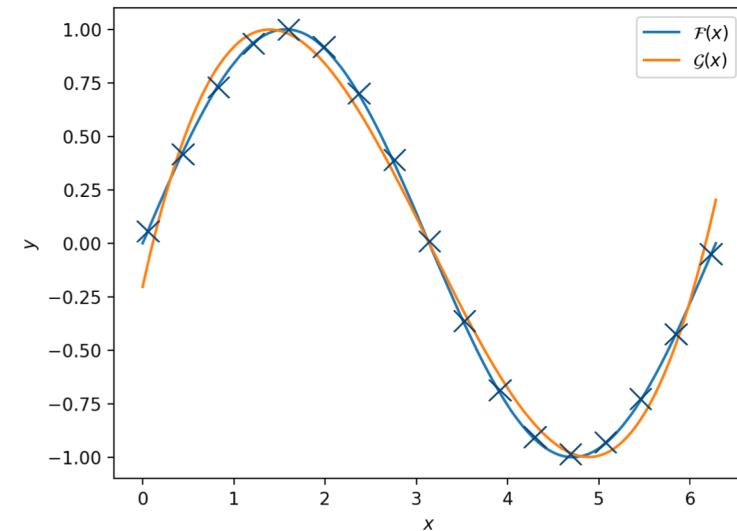
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #2: scalar-to-scalar mapping using nonlinear feature transform

- So far so good..
- Do you see any problem in this approach?
 - You should know the answer if you took MLSP
- You don't always know the mapping function
 - So, I was lucky to know that a cubic function will work
- You should ask me a question at this point
 - "Is there any systematic way that can model *any* function?"
 - But hold on that question for now
- In the procedure we just saw
 - We transformed the 1D data samples into 3D features
 - Then the mapping procedure became a linear function



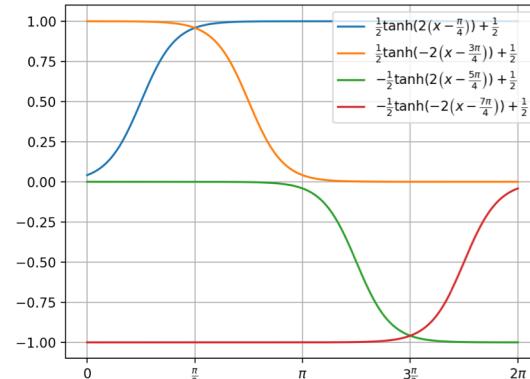
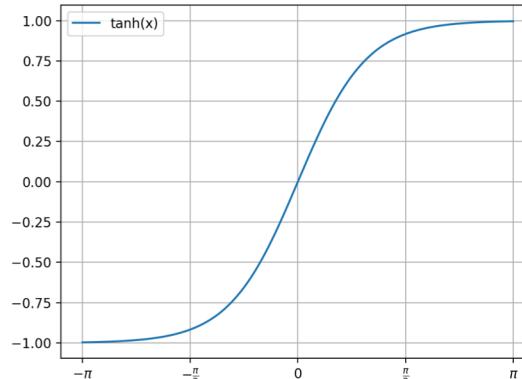
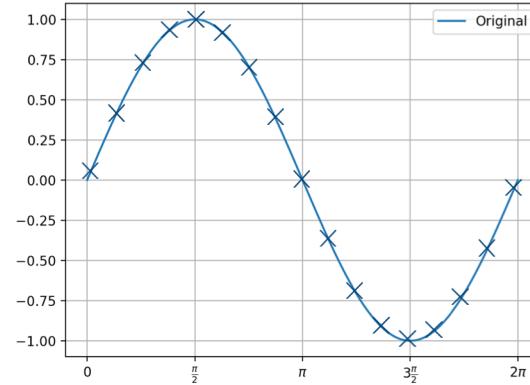
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #3: scalar-to-scalar mapping using *universal* nonlinear feature transform

- Another approach
- This time we're going to use another kind of transform
- Some observations
 - I see a hill and a valley
 - The hill: centered around $\frac{\pi}{2}$
 - The valley: centered around $\frac{3\pi}{2}$
- Let's create them
 - I will use sigmoid functions
 - I rescale and shift them around



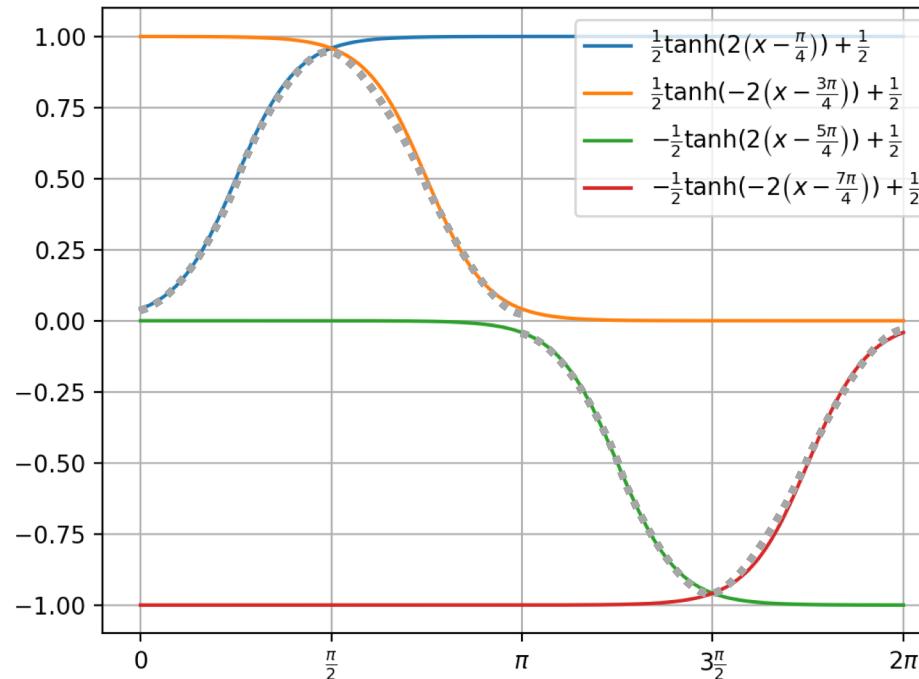
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #3: scalar-to-scalar mapping using *universal* nonlinear feature transform

- If we can somehow combine the “features,” we might recover the original sine function



□ How?



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

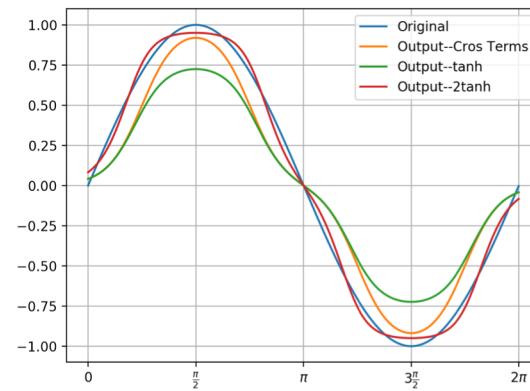
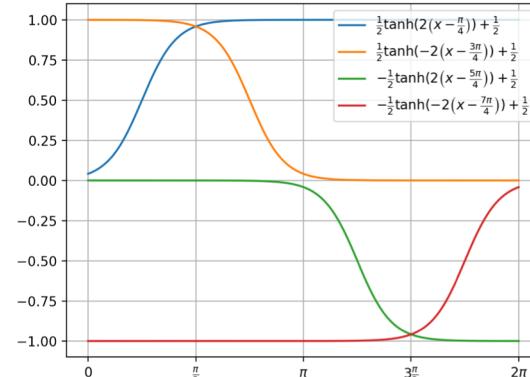
The Last Layer

- Case study #3: scalar-to-scalar mapping using *universal* nonlinear feature transform

- Let's define our features first

$$\begin{aligned} h_1 &= \frac{1}{2} \tanh\left(2\left(x - \frac{\pi}{4}\right)\right) + \frac{1}{2} \\ h_2 &= \frac{1}{2} \tanh\left(-2\left(x - \frac{3\pi}{4}\right)\right) + \frac{1}{2} \\ h_3 &= -\frac{1}{2} \tanh\left(2\left(x - \frac{5\pi}{4}\right)\right) + \frac{1}{2} \\ h_4 &= -\frac{1}{2} \tanh\left(-2\left(x - \frac{7\pi}{4}\right)\right) + \frac{1}{2} \end{aligned}$$

- Can you think of a way to form the sine function?
- $\hat{y} = h_1 h_2 - h_3 h_4$
- $\hat{y} = \tanh(h_1 + h_2 - 1) + \tanh(h_3 + h_4 + 1)$
- $\hat{y} = \tanh(2(h_1 + h_2 - 1)) + \tanh(2(h_3 + h_4 + 1))$
- We can combine a bunch of tanh functions to create valleys and hills!
- How do we find out those coefficients?
 - That's why we need optimization



INDIANA UNIVERSITY

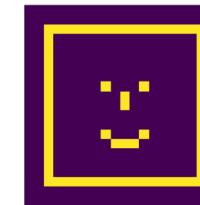
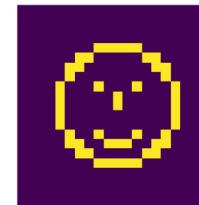
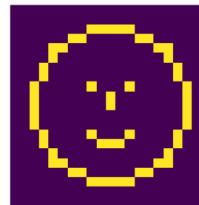
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

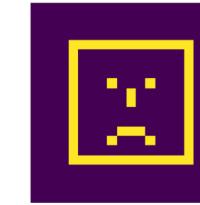
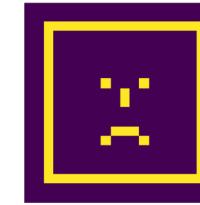
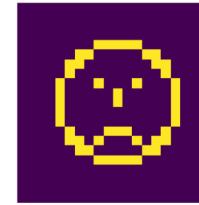
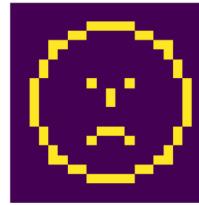
- Case study #4: vector-to-scalar mapping for binary classification

- Roughly speaking, classification can be seen as template matching
 - To see if there's a particular pattern in the example
- Can you figure out what kind of templates are there in the following images?

Class A



Class B



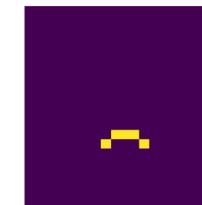
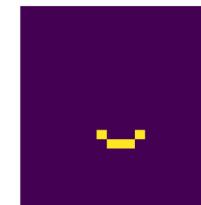
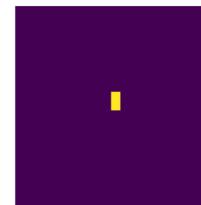
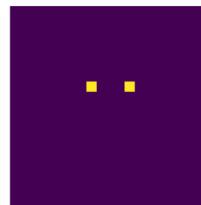
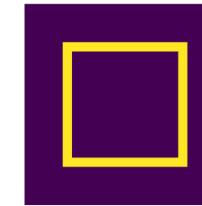
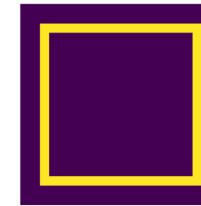
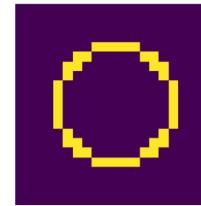
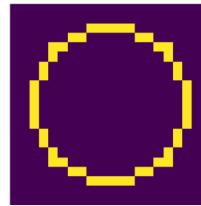
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #4: vector-to-scalar mapping for binary classification

- I feel that there are eight different templates (and features drawn from them)
 - Which one is important for the classification problem?



INDIANA UNIVERSITY

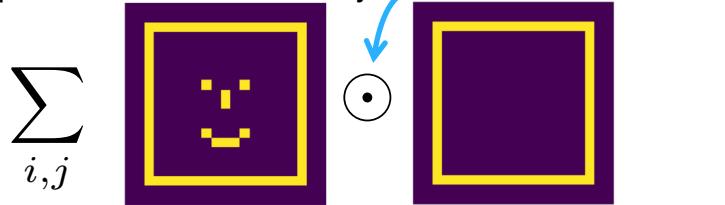
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #4: vector-to-scalar mapping for binary classification

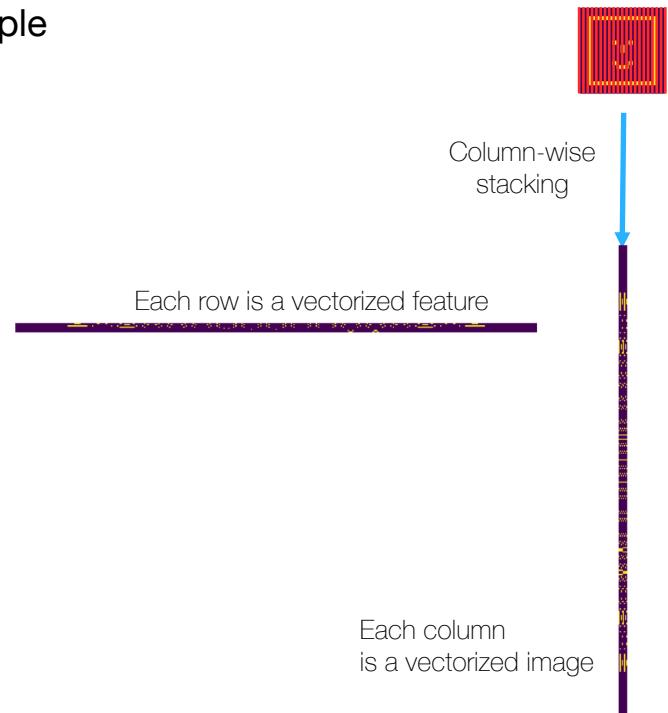
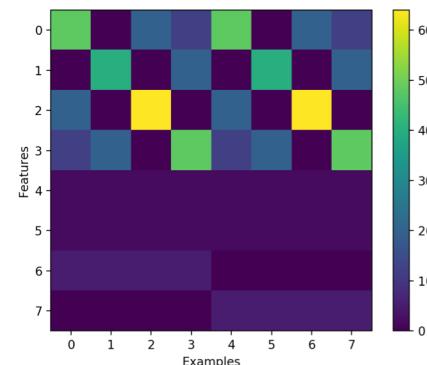
- Let's check out which one is important
- We do dot product between the template and the image sample

- Dot product between 2D arrays?



$$H = W^\top X$$

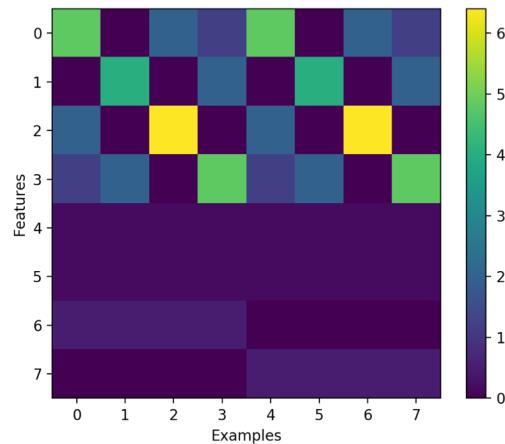
- Vectorize and inner product



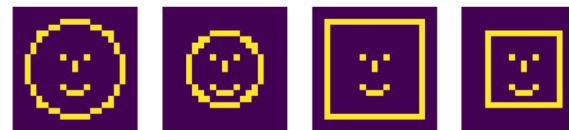
The Last Layer

- Case study #4: vector-to-scalar mapping for binary classification

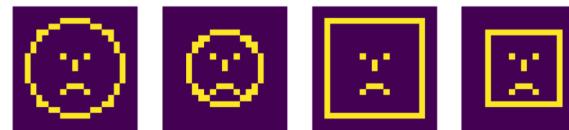
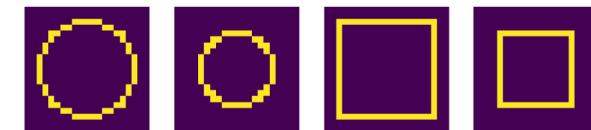
- What does this mean?



How many features were there originally?



Eight different templates form eight features



INDIANA UNIVERSITY

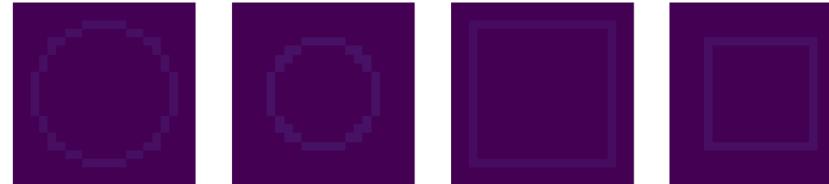
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #4: vector-to-scalar mapping for binary classification

- I finally have a feeling that 6th and 7th features are important
 - But their activations are not strong enough
 - Why? How do we improve it?
- I normalized the templates so that $W_{:,i}$ sums to one

\tilde{W}

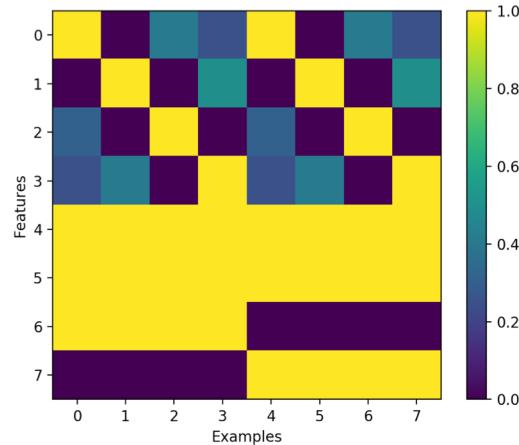


INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

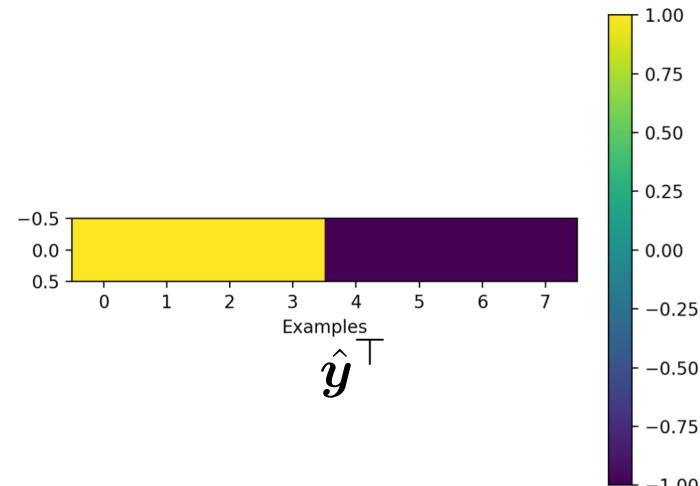
- Case study #4: vector-to-scalar mapping for binary classification



$$H = \tilde{W}^\top X$$

- Then, how do I make a decision?
 - I'll figure out the way to focus only on those important ones
- One way: scalar output $w^{(2)} = [0, 0, 0, 0, 0, 0, 1, -1]^\top$

$$\hat{y}^\top = w^{(2)\top} H$$



INDIANA UNIVERSITY

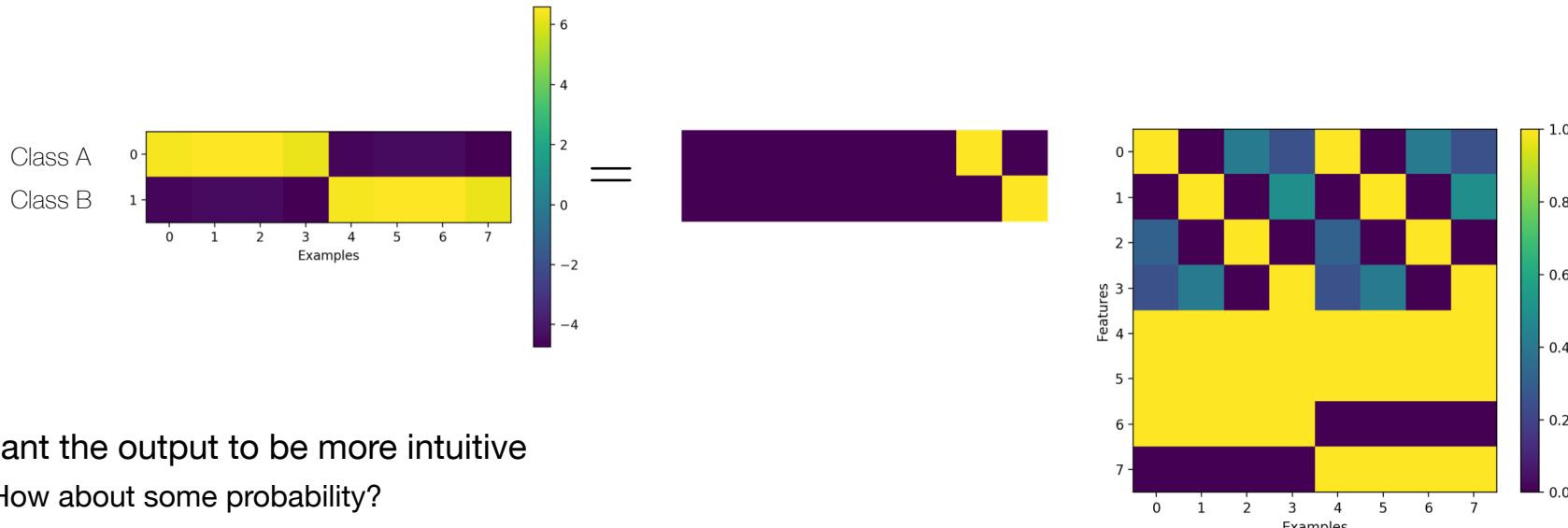
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #5: vector-to-vector mapping for multiclass classification

- As for the binary classification there's another way to encode the output

$$\mathbf{W}^{(2)} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & 10 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 10 \end{bmatrix}$$
$$\hat{\mathbf{Y}} = \mathbf{W}^{(2)} \mathbf{H}$$



- I want the output to be more intuitive
 - How about some probability?



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #5: vector-to-vector mapping for multiclass classification

- Logistic regression

- We want the prediction to be probability vectors

$$P(\text{Class A} | \mathbf{X}_{:,t}) \neq \hat{Y}_{0,t} = \mathbf{W}_{0,:}^{(2)} \mathbf{H}_{:,t}$$

Why not? Can be negative

$$P(\text{Class B} | \mathbf{X}_{:,t}) \neq \hat{Y}_{1,t} = \mathbf{W}_{1,:}^{(2)} \mathbf{H}_{:,t}$$

$$P(\text{Class A} | \mathbf{X}_{:,t}) \neq \exp(\hat{Y}_{0,t}) = \exp(\mathbf{W}_{0,:}^{(2)} \mathbf{H}_{:,t})$$

$$P(\text{Class B} | \mathbf{X}_{:,t}) \neq \exp(\hat{Y}_{1,t}) = \exp(\mathbf{W}_{1,:}^{(2)} \mathbf{H}_{:,t})$$

$$P(\text{Class A} | \mathbf{X}_{:,t}) = \frac{\exp(\hat{Y}_{0,t})}{\exp(\hat{Y}_{0,t}) + \exp(\hat{Y}_{1,t})}$$

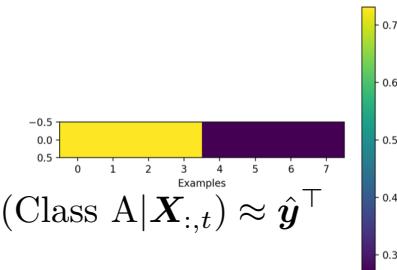
$$P(\text{Class B} | \mathbf{X}_{:,t}) = \frac{\exp(\hat{Y}_{1,t})}{\exp(\hat{Y}_{0,t}) + \exp(\hat{Y}_{1,t})}$$

- $\mathbf{W}^{(2)}$ is overparameterized and we can simplify it

$$\frac{\exp(\hat{Y}_{0,t})}{\exp(\hat{Y}_{0,t}) + \exp(\hat{Y}_{1,t})} = \frac{1}{1 + \exp(\hat{Y}_{1,t} - \hat{Y}_{0,t})} = \frac{1}{1 + \exp((\mathbf{W}_{1,:}^{(2)} - \mathbf{W}_{0,:}^{(2)})^\top \mathbf{H}_{:,t})} = \frac{1}{1 + \exp(\mathbf{w}^{(2)\top} \mathbf{H}_{:,t})} \quad \leftarrow \text{Logistic Function}$$

$$\frac{\exp(\hat{Y}_{1,t})}{\exp(\hat{Y}_{0,t}) + \exp(\hat{Y}_{1,t})} = \frac{\exp(\hat{Y}_{1,t} - \hat{Y}_{0,t})}{1 + \exp(\hat{Y}_{1,t} - \hat{Y}_{0,t})} = \frac{\exp((\mathbf{W}_{1,:}^{(2)} - \mathbf{W}_{0,:}^{(2)})^\top \mathbf{H}_{:,t})}{1 + \exp((\mathbf{W}_{1,:}^{(2)} - \mathbf{W}_{0,:}^{(2)})^\top \mathbf{H}_{:,t})} = \frac{\exp(\mathbf{w}^{(2)\top} \mathbf{H}_{:,t})}{1 + \exp(\mathbf{w}^{(2)\top} \mathbf{H}_{:,t})}$$

$$\mathbf{w}^{(2)} = \mathbf{W}_{1,:}^{(2)} - \mathbf{W}_{0,:}^{(2)}$$



- This gives us a hint about multiclass classification



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

The Last Layer

- Case study #5: vector-to-vector mapping for multiclass classification

- Softmax regression

- Again, we want the output to be a probabilistic vector

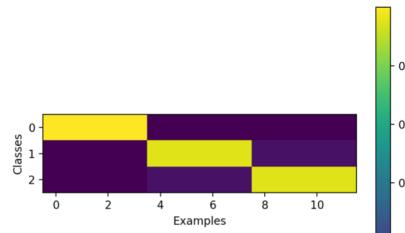
$$\mathbf{Z}_{c,t} = \mathbf{W}_{c,:}^{(2)} \mathbf{H}_{:,t}$$

$$P(\text{Class } \#0 | \mathbf{X}_{:,t}) = \frac{\exp(\mathbf{Z}_{0,t})}{\sum_{c=0}^{C-1} \exp(\mathbf{Z}_{c,t})}$$

$$P(\text{Class } \#1 | \mathbf{X}_{:,t}) = \frac{\exp(\mathbf{Z}_{1,t})}{\sum_{c=0}^{C-1} \exp(\mathbf{Z}_{c,t})}$$

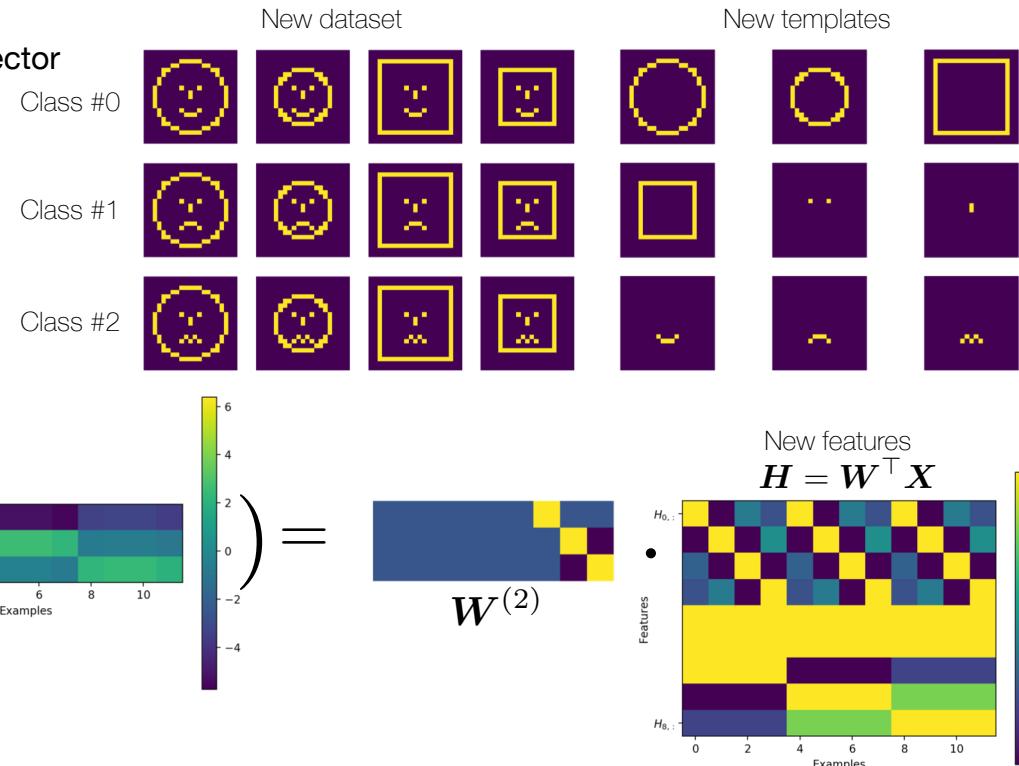
...

$$P(\text{Class } \#C-1 | \mathbf{X}_{:,t}) = \frac{\exp(\mathbf{Z}_{C-1,t})}{\sum_{c=0}^{C-1} \exp(\mathbf{Z}_{c,t})}$$



$$= \text{softmax} \left(\begin{array}{c|cccccc} & \text{Examples} \\ \hline \text{Classes} & 0 & 1 & 2 & 0 & 1 & 2 \\ \hline 0 & 0.8 & 0.0 & 0.0 & 0.8 & 0.0 & 0.0 \\ 1 & 0.0 & 0.8 & 0.0 & 0.0 & 0.8 & 0.0 \\ 2 & 0.0 & 0.0 & 0.8 & 0.0 & 0.0 & 0.8 \end{array} \right) =$$

$$\mathbf{W}^{(2)}$$



The Last Layer

- Take-home messages

- If you know good features things are easier
 - For regression, you can combine hills and valleys to approximate any funky shape
 - For classification, good features can be created from template matching
 - If you know a good set of templates for the classification
- Some details
 - Sigmoid functions are useful and versatile to create hills and valleys
 - Softmax and logistic functions are useful for the posterior-probability-like output variables
- Potential questions
 - How do we find out those features?
 - Note that I manually found them for this lecture



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING