ENGR-E 533 “Deep Learning Systems”
Lecture 01: The Last Layer

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

New Machine Is Coming
- NVIDIA Volta

- http://www.nvidia.com/v100
- ISE provides a cluster with 16 Volta GPUs for this course
  - But, not now
  - We've ordered this way in advance, but it took some time for manufacturing, since it's a new architecture
  - About 6 times faster than K80
  - It will arrive by the end of this month
- In the mean time..
  - You can either locally install the libraries and play around
  - (Sorry for the delay)

The Last Layer?

- Function approximation

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

The output of the function $\in \mathbb{R}^{K\times 1}$ → $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x})$ ← 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 → $\hat{\boldsymbol{y}} = \mathcal{G}(\boldsymbol{x}\ ; \mathbb{W}\ )$ ← Parameters

The estimate of the mapping function

Error function of your choice

- The objective function? $\arg\min_{\mathcal{G}} \mathcal{D}(\boldsymbol{y}||\hat{\boldsymbol{y}})$
  - What's wrong with it?
- It's easier to work with a parametric function $\arg\min_{\mathbb{W}} \mathcal{D}\big(\boldsymbol{y}||\mathcal{G}(\boldsymbol{x};\mathbb{W})\big)$
- The actual optimization is done to reduce the sum of all error network

$$\arg\min_{\mathbb{W}} \sum_t \mathcal{D}\big(\boldsymbol{Y}_{:,t})||\mathcal{G}(\boldsymbol{X}_{:,t};\mathbb{W})\big)$$

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-vector mapping)
    - Vector-to-[bounded scalar] 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

Scalar-to-scalar Mapping

- Linear approximation

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

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

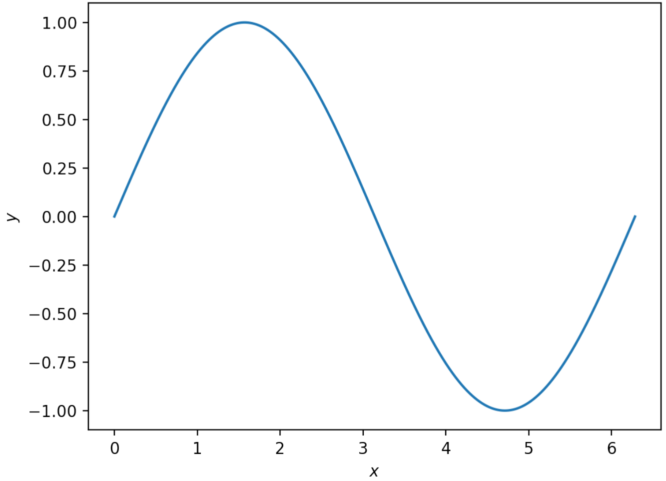
  - And we don't know it

- What happens if we try to approximate it linearly?

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

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

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



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

$$[\hat{y}_0\ \hat{y}_1\ \hat{y}_2\ \cdots\ \hat{y_{T-1}}] = [a_1\ a_0] \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{T-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$\Leftrightarrow \hat{\boldsymbol{y}}^\top = \boldsymbol{a}^\top \boldsymbol{X}$$

Scalar-to-scalar Mapping

- Linear approximation

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

- What are we going to do with this?

$$\arg\min_{\mathbb{W}=\boldsymbol{a}} \sum_t \mathcal{D}(\hat{y}_t || y_t) = \arg\min_{\boldsymbol{a}} \sum_t (\hat{y}_t - y_t)^2 = (\boldsymbol{y} - \hat{\boldsymbol{y}})^\top (\boldsymbol{y} - \hat{\boldsymbol{y}})$$

$$= \arg\min_{\boldsymbol{a}} (\boldsymbol{y}^\top - \boldsymbol{a}^\top \boldsymbol{X})(\boldsymbol{y}^\top - \boldsymbol{a}^\top \boldsymbol{X})^\top = \arg\min_{\boldsymbol{a}} (\boldsymbol{y}^\top - \boldsymbol{a}^\top \boldsymbol{X})(\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{a})$$

$$= \arg\min_{\boldsymbol{a}} (\boldsymbol{y}^\top \boldsymbol{y} + \boldsymbol{a}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{a} - 2\boldsymbol{y}^\top \boldsymbol{X}^\top \boldsymbol{a})$$

- Then what?

$$\frac{\partial \boldsymbol{y}^\top \boldsymbol{y} + \boldsymbol{a}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{a} - 2\boldsymbol{y}^\top \boldsymbol{X}^\top \boldsymbol{a}}{\partial \boldsymbol{a}} = 2\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{a} - 2\boldsymbol{X}\boldsymbol{y}^\top = 0$$

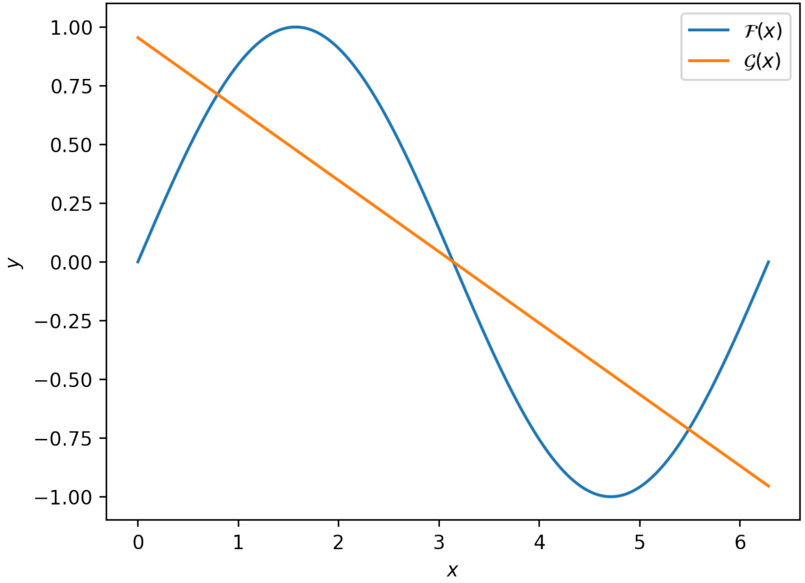
$$\boldsymbol{a} = (\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{X}\boldsymbol{y}^\top$$

  - 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

Scalar-to-scalar Mapping

- Linear approximation

- Life is not that simple..

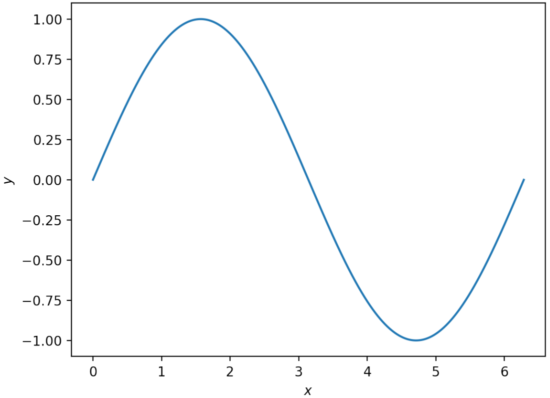


$$\mathcal{G}(x_t) = -0.30x_t + 0.95$$

Scalar-to-scalar Mapping

- Linear approximation with known 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



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

$$\Leftrightarrow \hat{\boldsymbol{y}}^\top = \boldsymbol{a}^\top \boldsymbol{X}$$

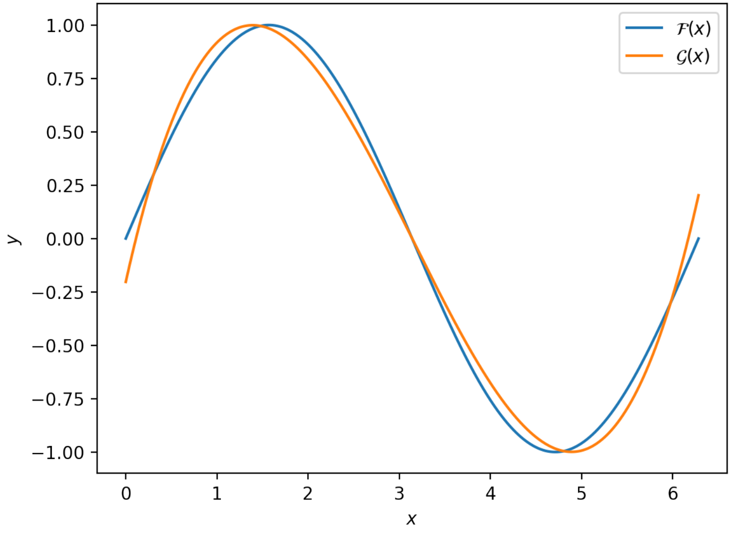
- How do we solve this?
  - Same as before

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

Scalar-to-scalar Mapping

- Linear approximation with known nonlinear feature transform

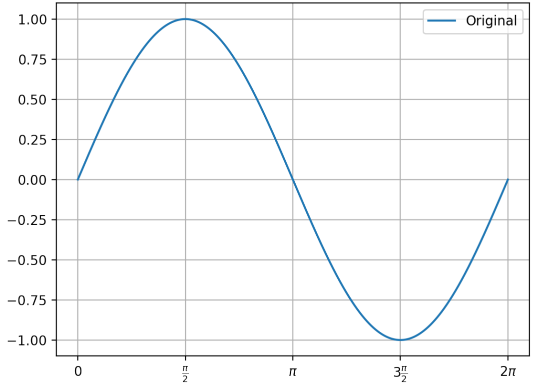
- So far so good..
- Do you see any problem in this approach?
  - You should know the answer if you took MLSP
- You don't always know the mapping function
  - So, I was lucky to know that a cubic function will work
- You should ask me a question at this point
  - "Is there any systematic way that can 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

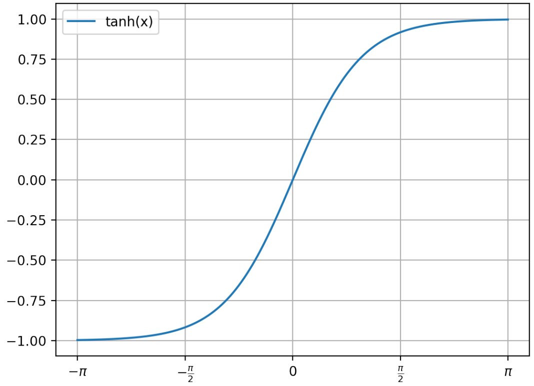


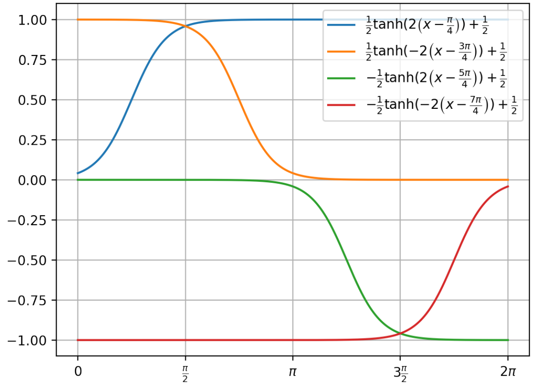
Scalar-to-scalar Mapping

- Linear approximation with 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



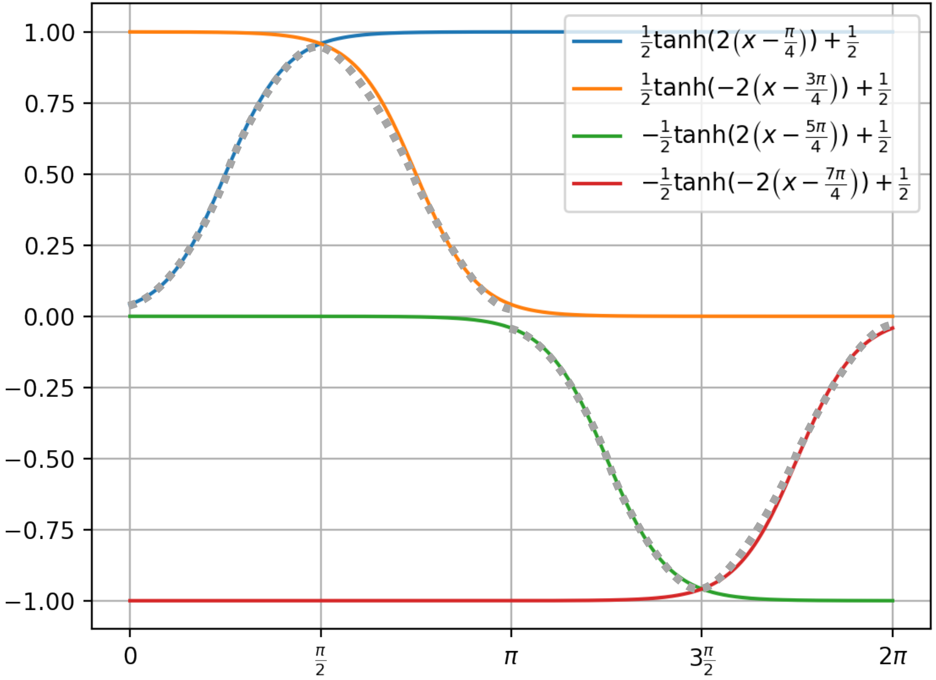




Scalar-to-scalar Mapping

- Linear approximation with universal nonlinear feature transform

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



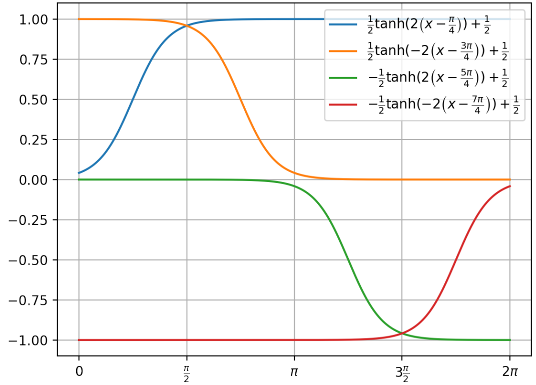
• How?

Scalar-to-scalar Mapping

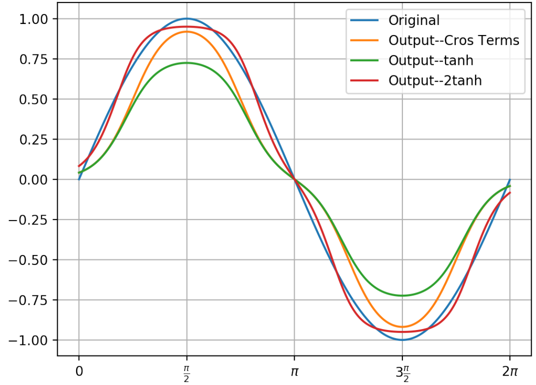
- Linear approximation with universal nonlinear feature transform

- Let's define our features first

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



- Can you think of a way to form the sine function?
- $\hat{y} = h_1 h_2 - h_3 h_4$
- $\hat{y} = \tanh\left(h_1 + h_2 - 1\right) + \tanh\left(h_3 + h_4 + 1\right)$
- $\hat{y} = \tanh\left(2(h_1 + h_2 - 1)\right) + \tanh\left(2(h_3 + h_4 + 1)\right)$
- 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

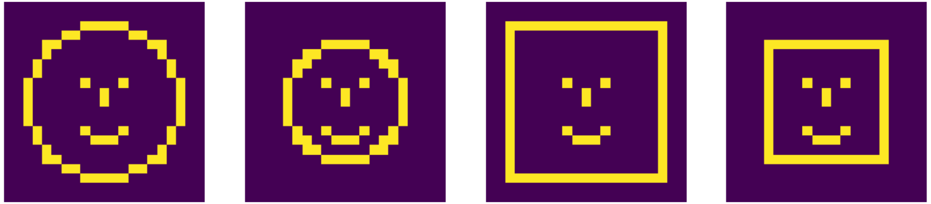


Vector-to-scalar Mapping

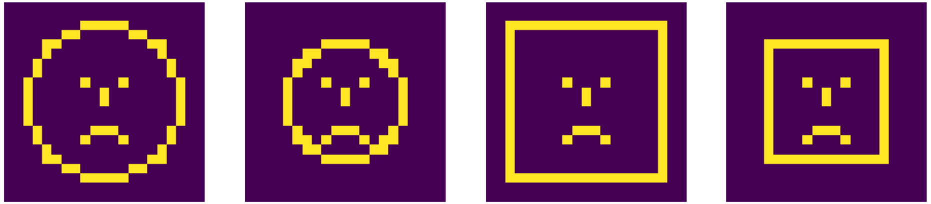
- Binary classification

- 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

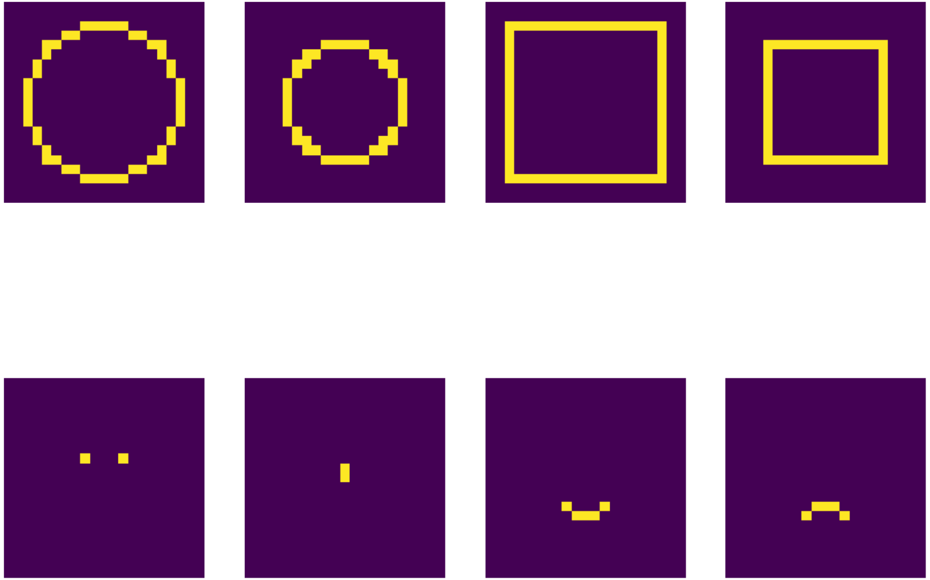


Class B



Vector-to-scalar Mapping
- Binary classification

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

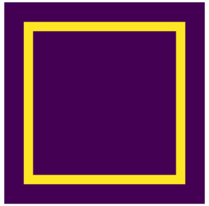


Vector-to-scalar Mapping

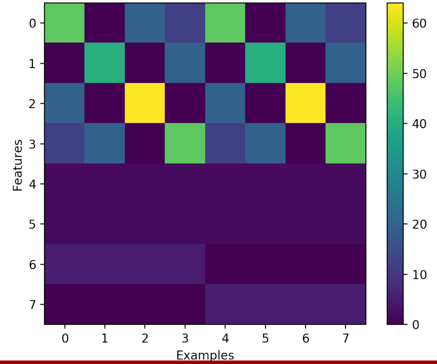
- Binary classification

- Let's check out which one is important
- We do dot product between the template and the image sample
  - Dot product between 2D arrays?

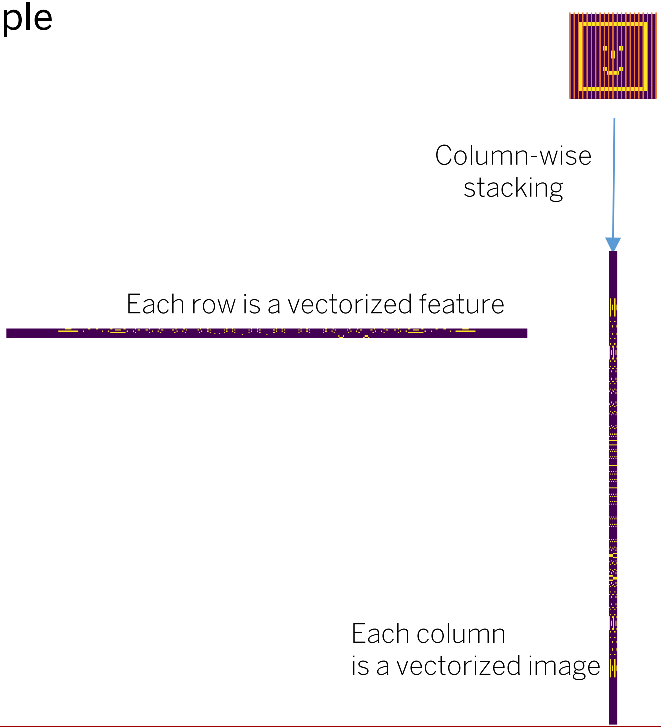
Hadamard product

$$\sum_{i,j}$$  $\odot$ 

  - Vectorize and inner product $\boldsymbol{H} = \boldsymbol{W}^\top \boldsymbol{X}$



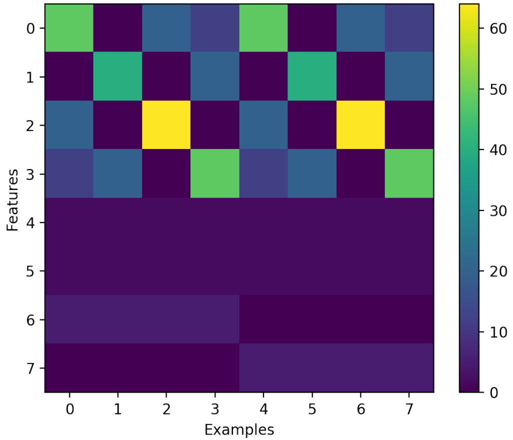
$=$



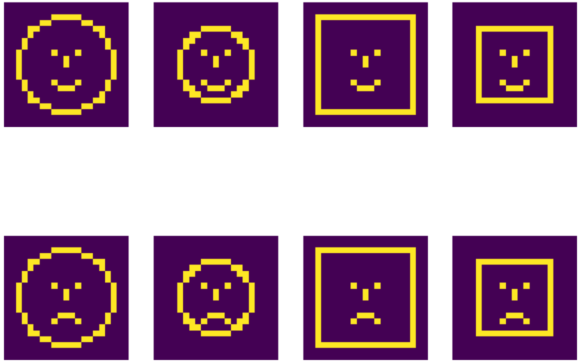
Vector-to-scalar Mapping

- Binary classification

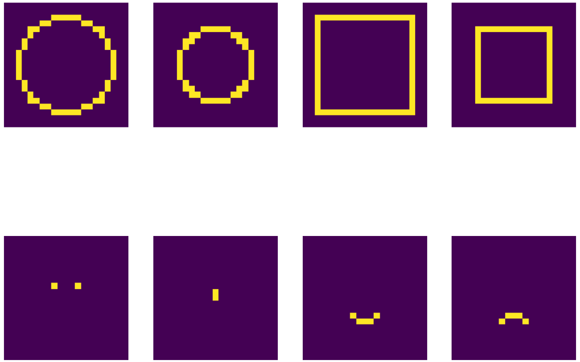
- What does this mean?



How many features were there originally?



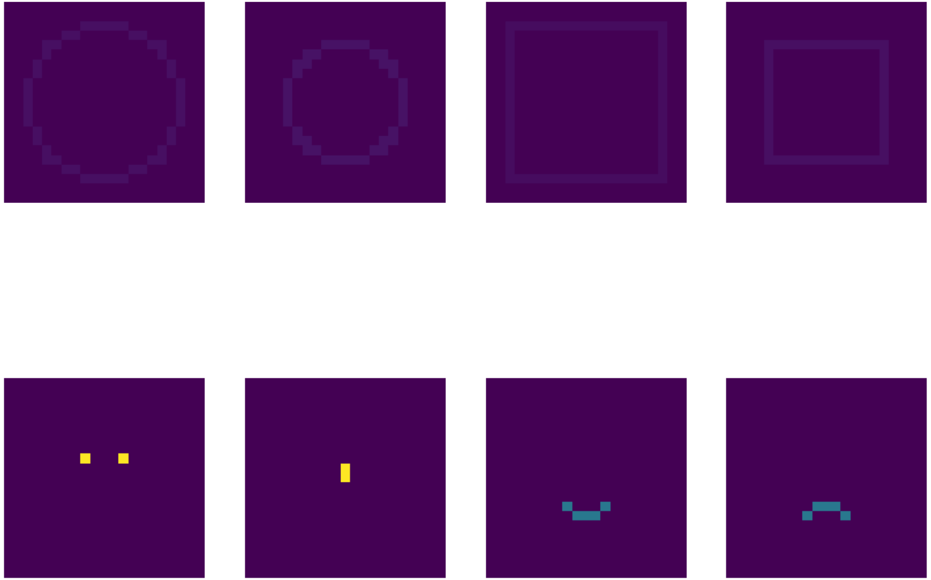
Eight different templates form eight features



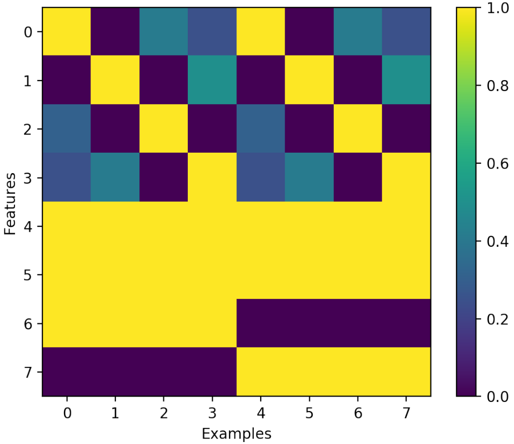
Vector-to-scalar Mapping
- Binary classification

- 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 $\boldsymbol{W}_{:,i}$ sums to one

$$\tilde{W}$$



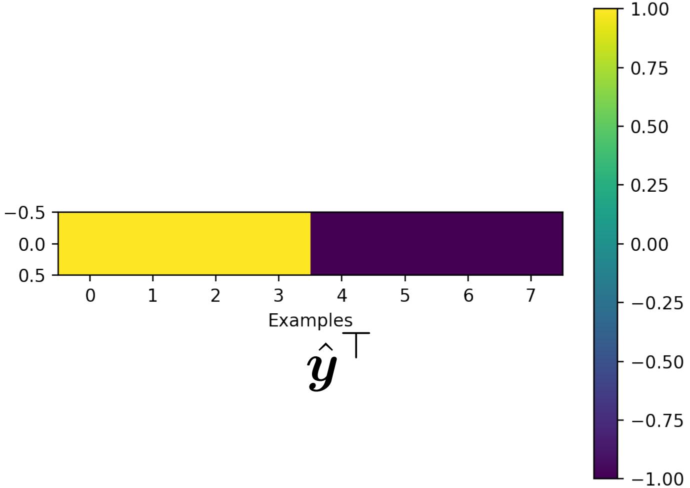
Vector-to-scalar Mapping
- Binary classification



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

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

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



$\hat{\boldsymbol{y}}^\top$

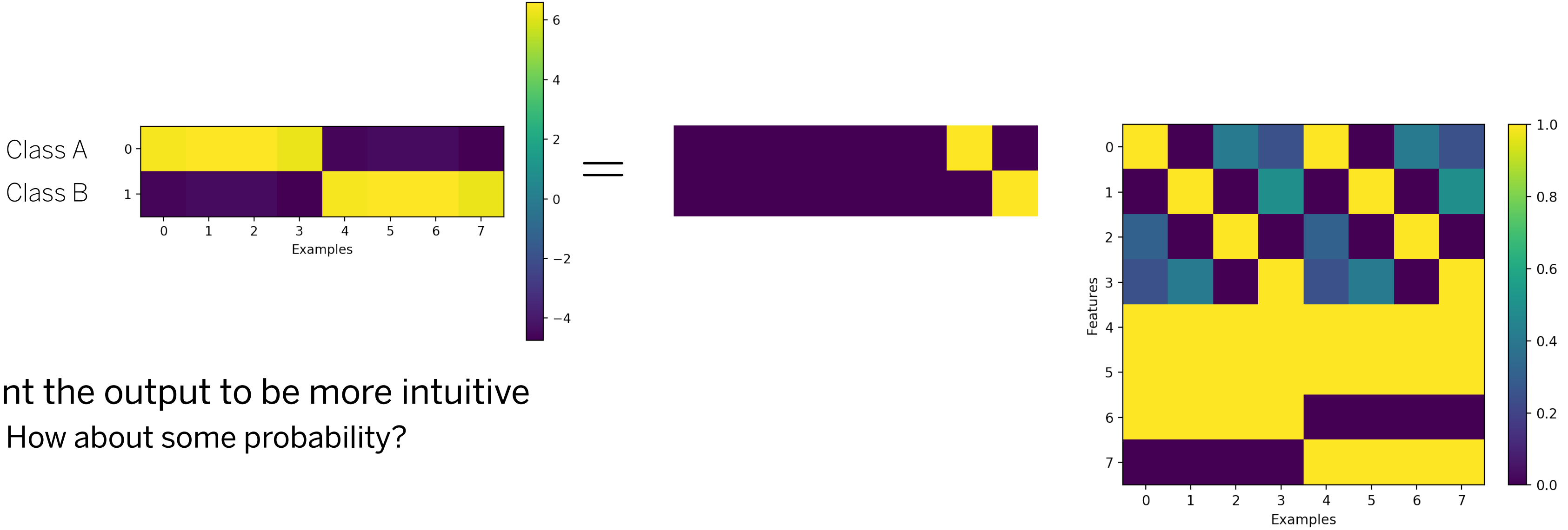
Vector-to-Vector Mapping

- Multiclass classification

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

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & 10 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 10 \end{bmatrix}$$

$$\hat{\boldsymbol{Y}} = \boldsymbol{W}^{(2)}\boldsymbol{H}$$



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

Vector-to-Vector Mapping

- Multiclass classification

- Logistic regression
  - We want the prediction to be probability vectors

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

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

Why not? Can be negative

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

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

Why not? Can be larger than one

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

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

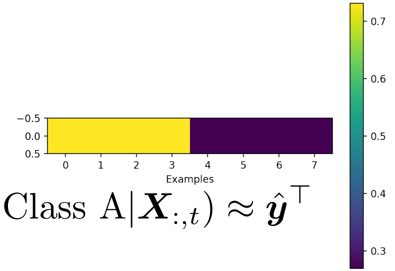
- $\boldsymbol{W}^{(2)}$ is overparameterized and we can simplify it

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

Logistic Function

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

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



$$P(\text{Class A}|\boldsymbol{X}_{:,t}) \approx \hat{\boldsymbol{y}}^\top$$

- This gives us a hint about multiclass classification

Vector-to-Vector Mapping

- Multiclass classification

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

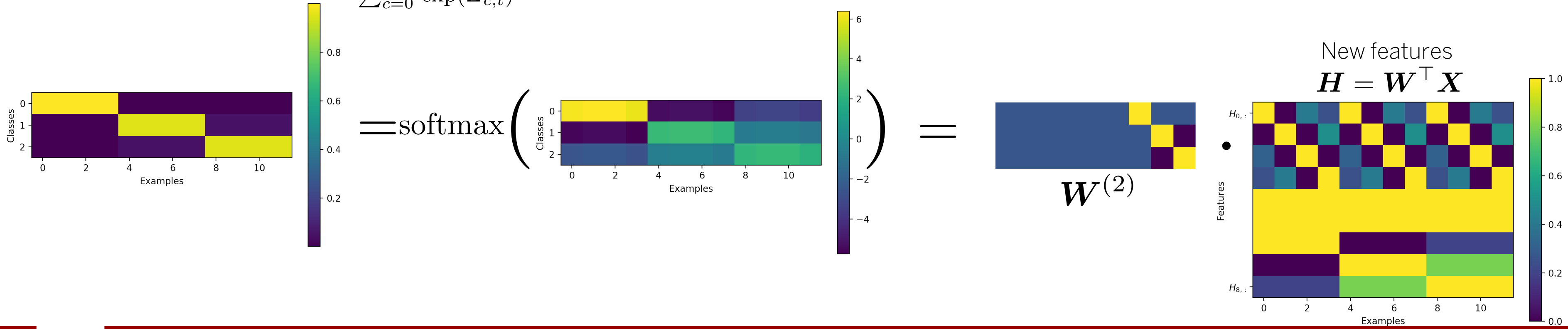
$$\boldsymbol{Z}_{c,t} = \boldsymbol{W}^{(2)}_{c,:}\boldsymbol{H}_{:,t}$$

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

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

$$\cdots$$

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



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

Thank You!

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