

DS 6050 Deep Learning

1. Machine Learning Recap

Sheng Li

Associate Professor

School of Data Science

University of Virginia

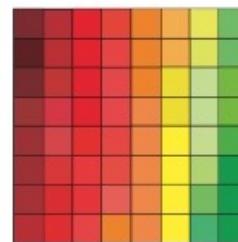
Matrices and Tensors

Tensors

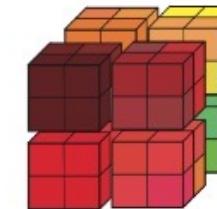
- What is a tensor?
 - Casual definitions:
 - A "tensor" is like a matrix but with an arbitrary number of dimensions. A 1-dimensional tensor is a vector. A 2-dimensional tensor is a matrix. And then you can have tensors with 3, 4, 5 or more dimensions.
 - A tensor is a container for numerical data.
 - An arbitrary dimensional generalization of matrices.
 - *note: dimension == axis



$$\mathbf{v} \in \mathbb{R}^{64}$$



$$\mathbf{X} \in \mathbb{R}^{8 \times 8}$$



$$\mathbf{X} \in \mathbb{R}^{4 \times 4 \times 4}$$

Scalars (0D Tensors)

- A tensor that contains only one number is called a scalar
 - or scalar tensor; or zero-dimensional tensor; or 0D tensor.
- We will write scalars in italics (e.g. $a, b, c, d, e, x, y, z, \alpha, \beta, \gamma, \varphi, \phi, \psi$)
- Scalars can be real-valued, complex-valued, etc. We will always specify what kind of number they are when we introduce them.
 - e.g. we might say “Let $s \in \mathbb{R}$ be the slope of the line,” while defining a real-valued scalar
 - or, we might say “Let $n \in \mathbb{N}$ be the number of units,” while defining a natural number scalar.

Vectors (1D Tensors)

- An array of numbers is called a vector, or 1D tensor.
- A vector that has n -entities/elements is called n -dimensional vector.
 - Don't get confused with nD Tensor!
- We will write vectors in bold (non-italic) typeface (e.g. **x, y, z, u, v, w, φ, ϕ, ψ**)
- The elements of a vector are identified by writing its name in italic with a subscript:

number of axis vs
number of rows/cols

$$\mathbf{x} = [x_1, x_2, \dots, x_n]$$

Matrices (2D Tensors)

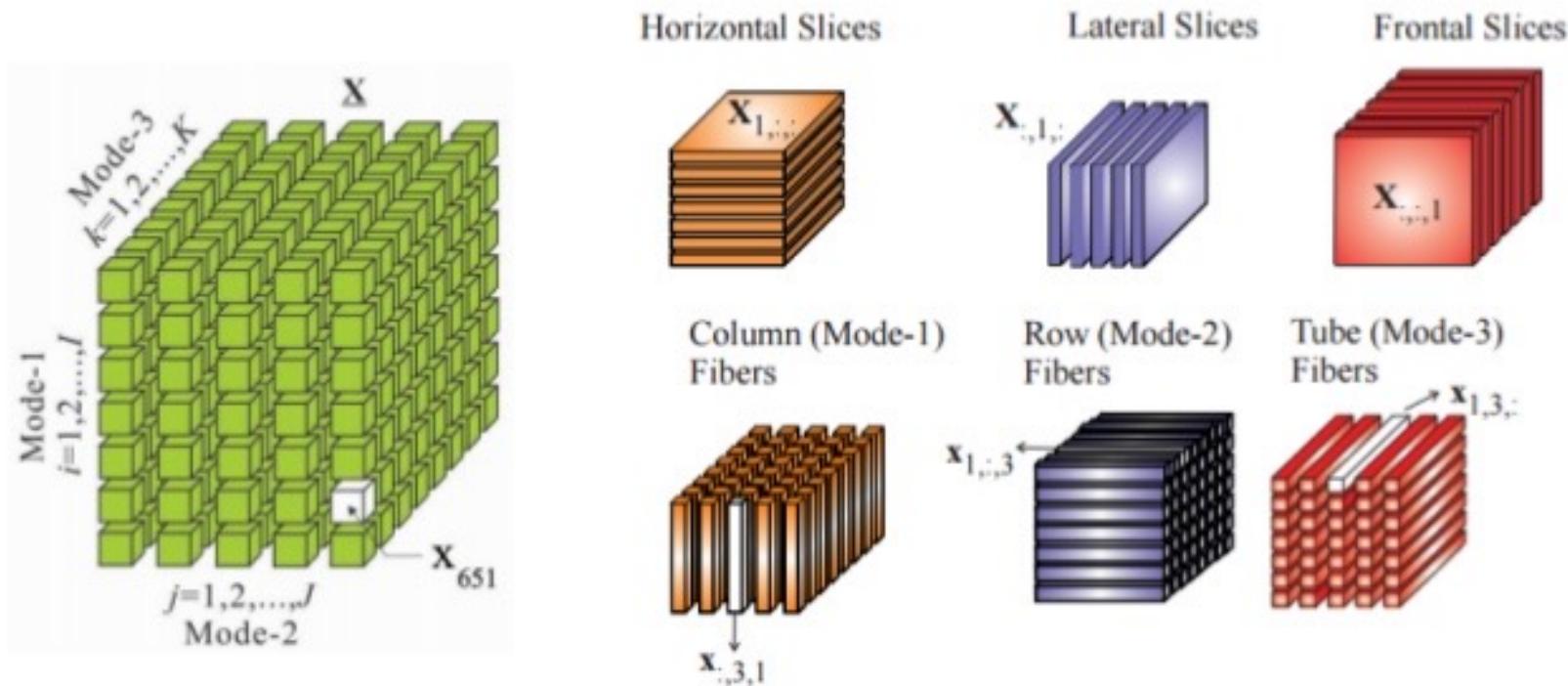
- An array of vectors is called a matrix, or 2D tensor.
 - Visually, a matrix is a rectangular grid of numbers.
 - A matrix has two axes—rows and columns.
- We will write matrices in bold uppercase letters (e.g. **A, B, C, X, Y, Z, Φ, Ψ, Ω**)
- The elements of a matrix are identified by writing its name in italic with two subscript indices:row, col

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}$$

- We can identify all the numbers with vertical coordinate i by writing ‘ $:$ ’ for the horizontal coordinate: $A_{i,:}$ (same goes for the horizontal cross-section)

3D Tensors and Higher

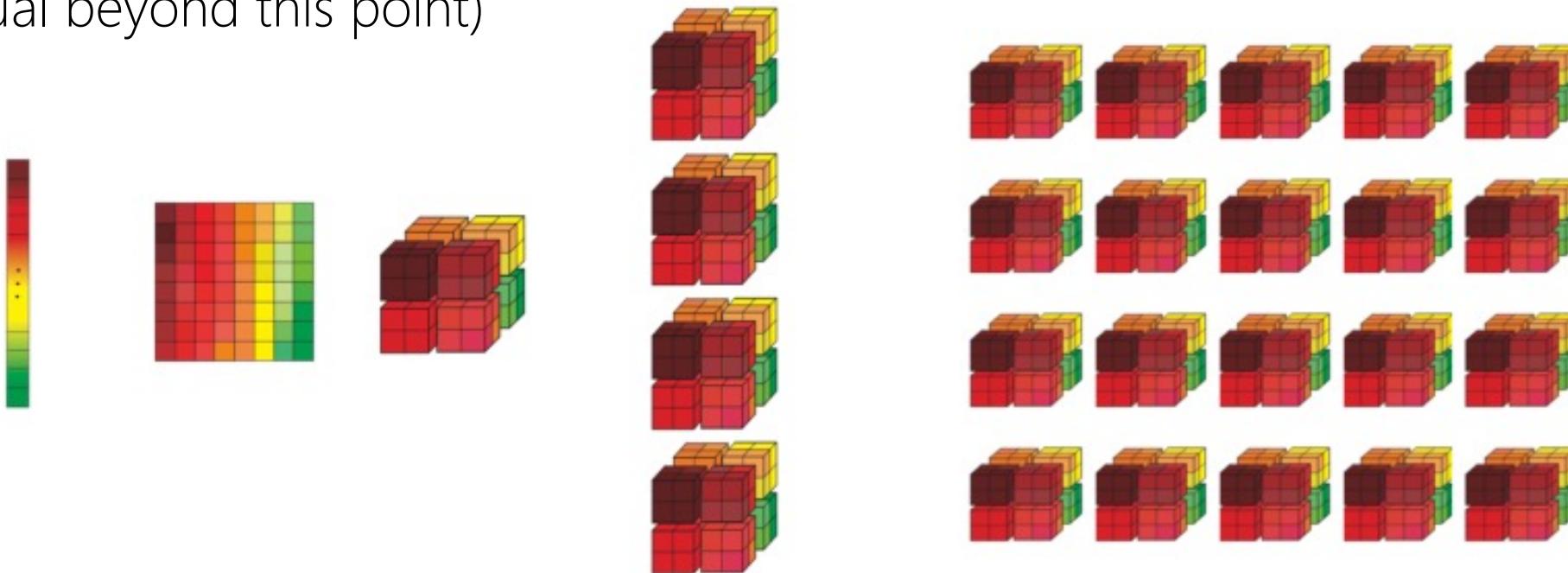
- A stack of matrices is called 3D tensor.
 - 3D tensor is like a cube of numbers.



$$\mathcal{X} \in \mathbb{R}^{7 \times 5 \times 8}$$

3D Tensors and Higher

- A stack of matrices is called 3D tensor.
 - 3D tensor is like a cube of numbers.
- A stack of 3D tensors along the fourth axis is called 4D tensor. (Things become virtual beyond this point)



ex) An Avocado Price Data is a Vector

Date	Average Price	Total Volume	Organic?	Region
12/18/2016	1.18	631117.18	conventional	Chicago



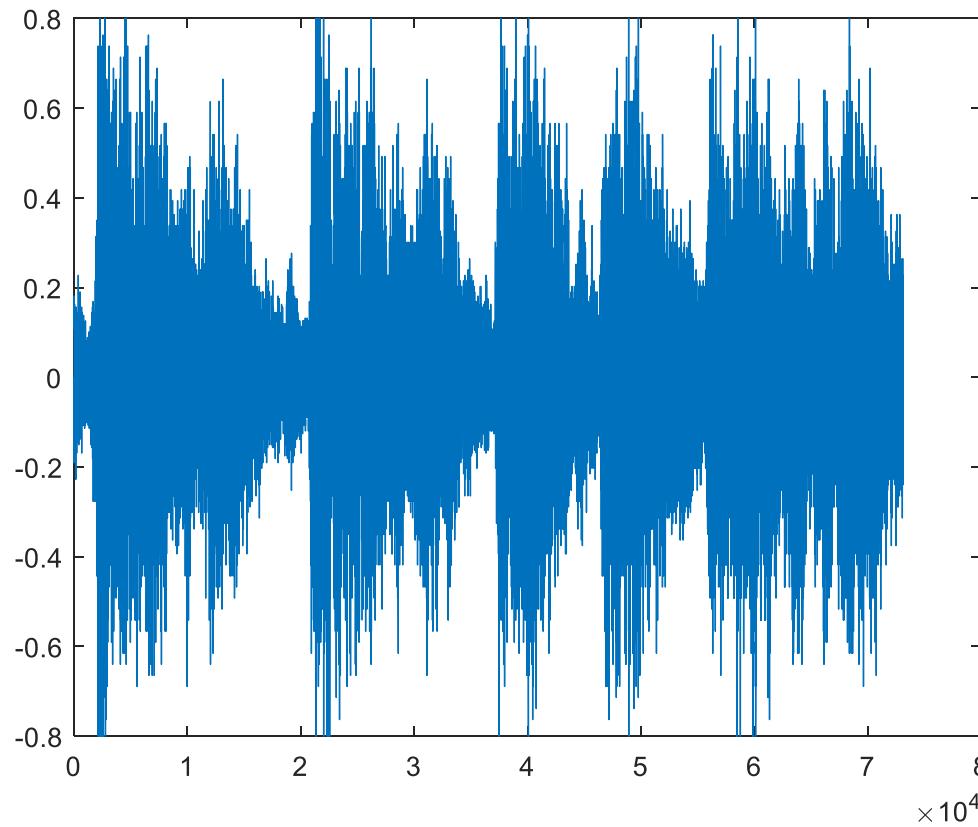
$$\mathbf{x} = \begin{bmatrix} 2016 \\ 12 \\ 18 \\ 1.18 \\ 631117.18 \\ 0 \\ 3 \end{bmatrix}$$

→ If organic 1, otherwise 0.
→ Region code.

ex) An Avocado Price Dataset is a Matrix

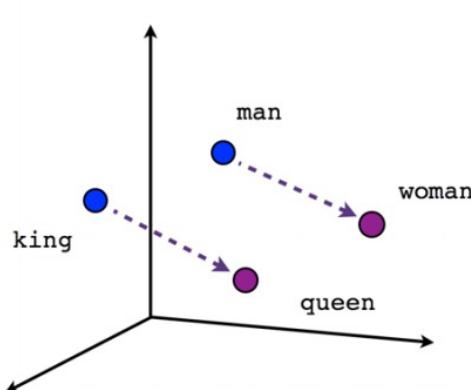
Date	Average Price	Total Volume	Organic?	Region
12/18/2016	1.18	631117.18	conventional	Chicago
12/11/2016	1.09	734954.21	conventional	Chicago
12/4/2016	1.27	624751.15	conventional	Chicago
11/27/2016	1.1	581638.75	conventional	Chicago
11/20/2016	1.13	694873.06	conventional	Chicago
11/13/2016	1.82	436710.91	conventional	Chicago
11/6/2016	2.07	376476.71	conventional	Chicago
10/30/2016	2.07	375213.57	conventional	Chicago
10/23/2016	1.84	431274.2	conventional	Chicago
10/16/2016	1.8	523682.5	conventional	Chicago
10/9/2016	1.82	515146.04	conventional	Chicago
10/2/2016	1.83	509215.79	conventional	Chicago
9/25/2016	1.8	553863.82	conventional	Chicago
9/18/2016	1.7	600389.56	conventional	Chicago

ex) A Mono Audio Signal is a Vector



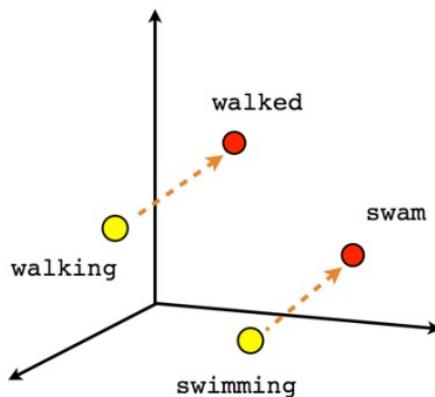
	73113x1 double
1	0
2	-0.0062
3	-0.0750
4	-0.0312
5	0.0062
6	0.0381
7	0.0189
8	-0.0250
9	-0.0312
10	-0.0750
11	-0.1258
12	-0.1443
13	-0.1812
14	-0.1905
15	-0.0750
16	-0.0127
17	-0.0381
18	-0.0750
19	0
20	0.0750
21	0.1258
22	0.1812
23	0.1505
24	0.1258
25	0.0812
26	0.0566
27	0
28	-0.0627
29	-0.0950
30	-0.0381

ex) A Word is a Vector.

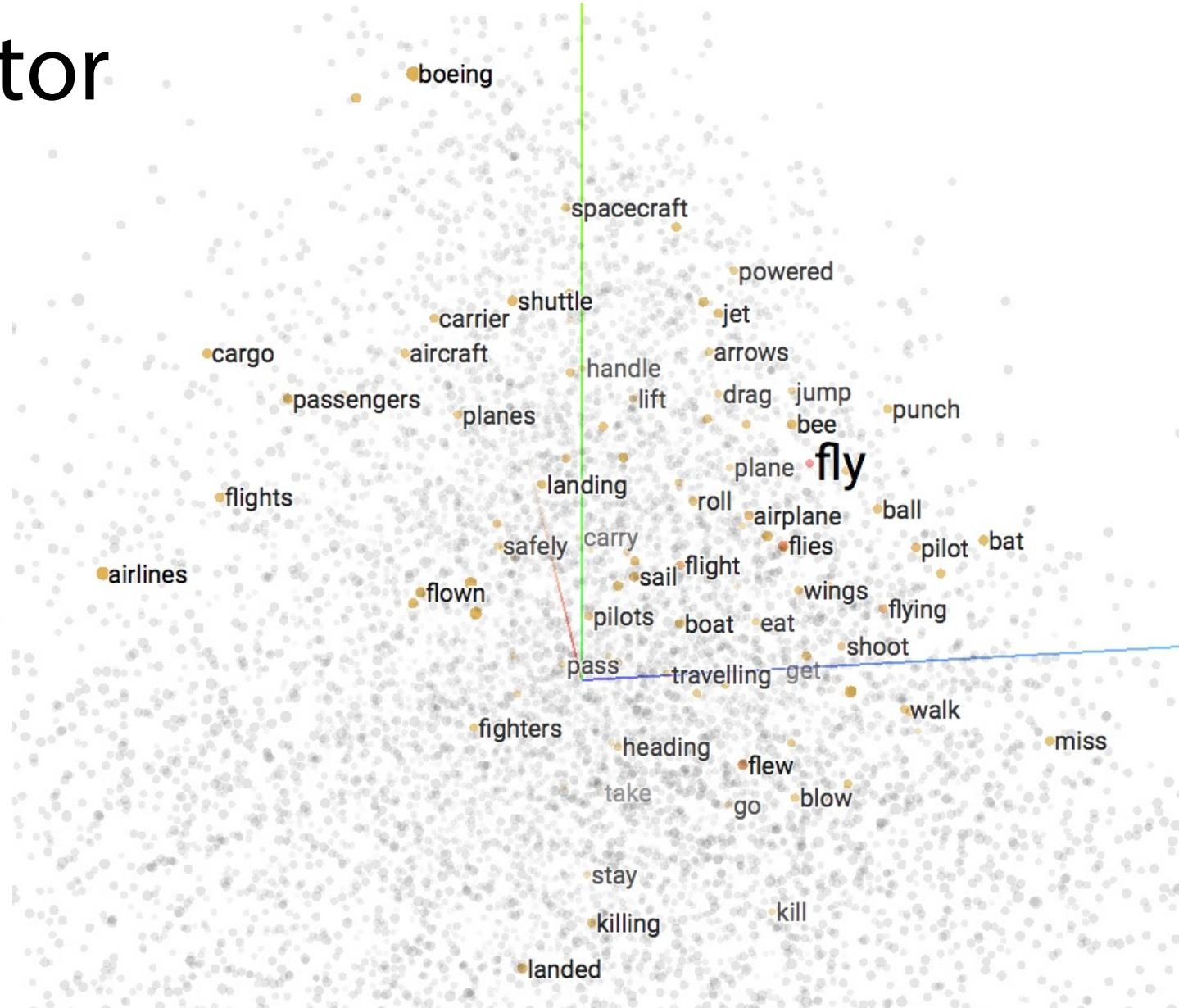


Male-Female

<https://www.tensorflow.org/tutorials/representation/word2vec>



Verb tense

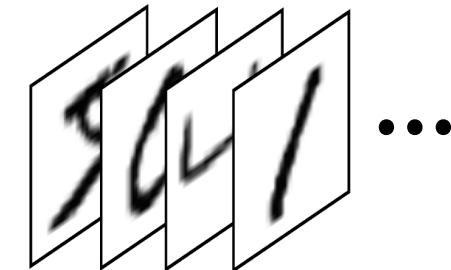
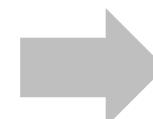


ex) A Gray Image is a Matrix

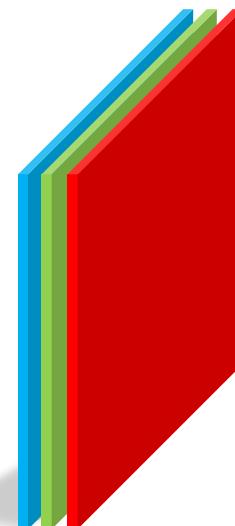


127	123	101	40	12	13	14	22	26	18	13	13	13	13	13	15
114	81	98	38	13	14	28	36	34	29	17	13	13	13	13	15
142	102	87	33	13	15	31	38	34	36	22	13	13	14	15	17
132	112	87	24	14	22	34	52	46	37	26	20	14	15	16	18
120	103	83	19	15	18	57	158	161	131	24	19	14	15	15	17
117	110	84	18	14	21	114	163	171	128	49	17	15	15	15	17
117	112	89	21	14	26	155	163	174	157	117	15	16	16	16	17
117	103	84	21	15	13	120	174	106	182	87	11	15	15	16	17
117	103	86	21	16	16	56	158	125	156	55	19	21	16	16	17
112	102	94	23	17	29	33	86	97	80	44	38	40	26	15	16
107	91	95	25	19	33	39	98	77	105	65	42	35	40	17	16
98	95	97	26	25	29	39	101	108	70	70	39	29	46	20	16
111	92	91	34	34	23	27	64	144	89	54	23	30	42	31	16
113	96	90	36	32	28	21	31	92	83	47	21	35	42	42	17
119	109	87	34	31	33	45	51	45	134	128	36	33	40	35	17
98	90	78	29	28	34	28	29	39	70	51	65	64	35	23	16

ex) MNIST Hand-written Digit Dataset = 3D



ex) A Color Image is a 3D Tensor



118	118	117	120	134	118	123	123	133	134	122	134
116	121	112	119	141	131	135	133	134	133	105	116
116	120	104	114	131	127	130	128	121	115	112	116
120	127	108	118	130	136	136	143	129	123	129	124
121	133	95	109	122	136	133	142	116	107	144	133
112	103	115	131	138	143	141	126	120	119	130	129
119	113	118	130	132	139	141	129	123	117	120	111
131	124	108	115	113	122	134	136	140	141	128	120
130	125	97	107	108	120	130	130	130	130	112	113
134	133	109	120	118	119	123	118	122	125	114	117
117	118	111	115	107	100	98	98	109	120	119	114
115	112	121	127	118	113	108	101	106	114	118	115
112	113	121	125	113	106	102	96	101	105	124	128
114	120	119	109	106	113	117	115	115	116	127	127
120	129	135	123	114	117	119	116	114	114	121	121
129	140	159	145	131	129	134	135	139	143	149	149
141	156	144	140	139	141	146	144	141	149	150	150
128	132	127	132	135	138	138	129	127	138	123	123



R

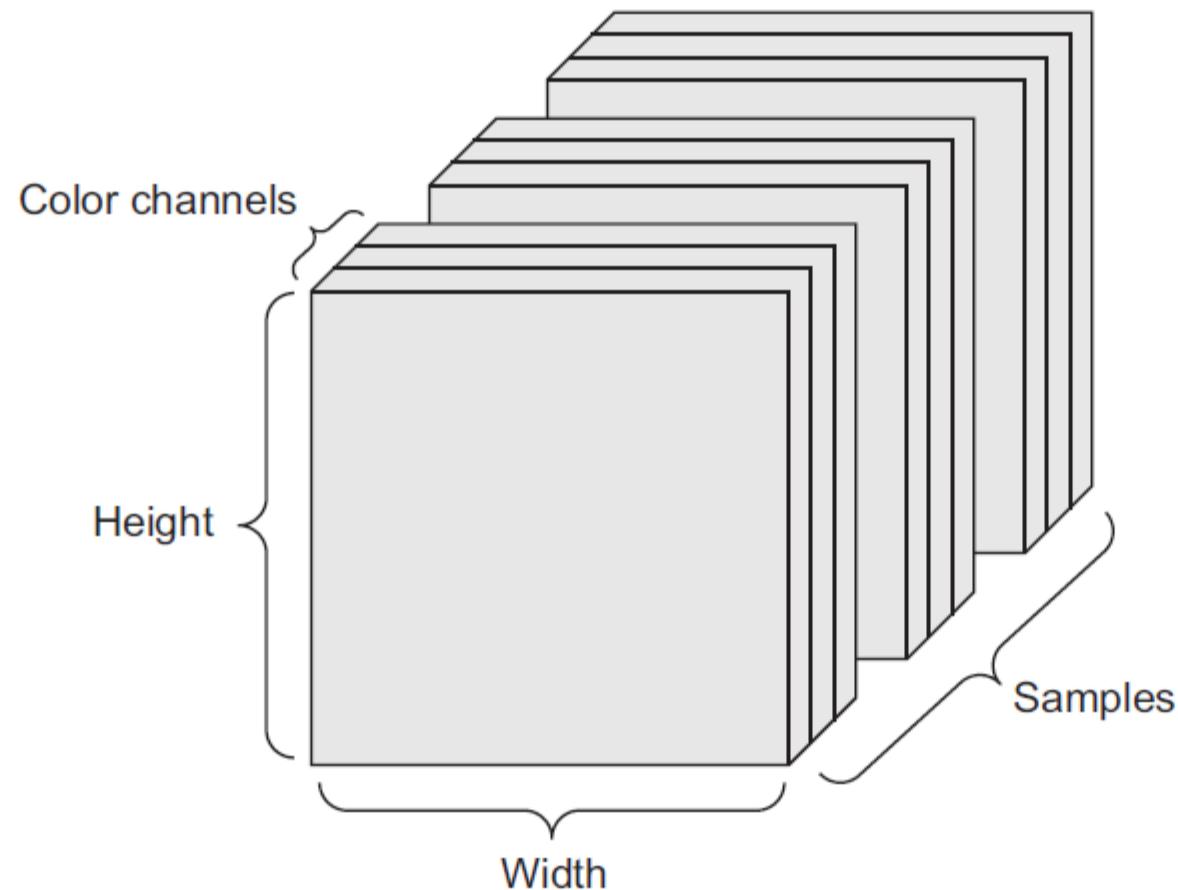


G



B

ex) Color Image Dataset is a 4D Tensor



ex) A Color Video is a 4D Tensor



Linear Algebra for ML

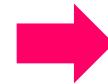
I assume you are comfortable with...

- Matrix Transpose
- Matrix Multiplication
- Identity and Inverse
- Linear Dependence and Span
- Norms
- Determinant
- Trace
- Eigendecomposition

Hadamard Product

look over LA notes
again ajfkldsjfkl

This guy!



- a.k.a. Element-wise matrix multiplication

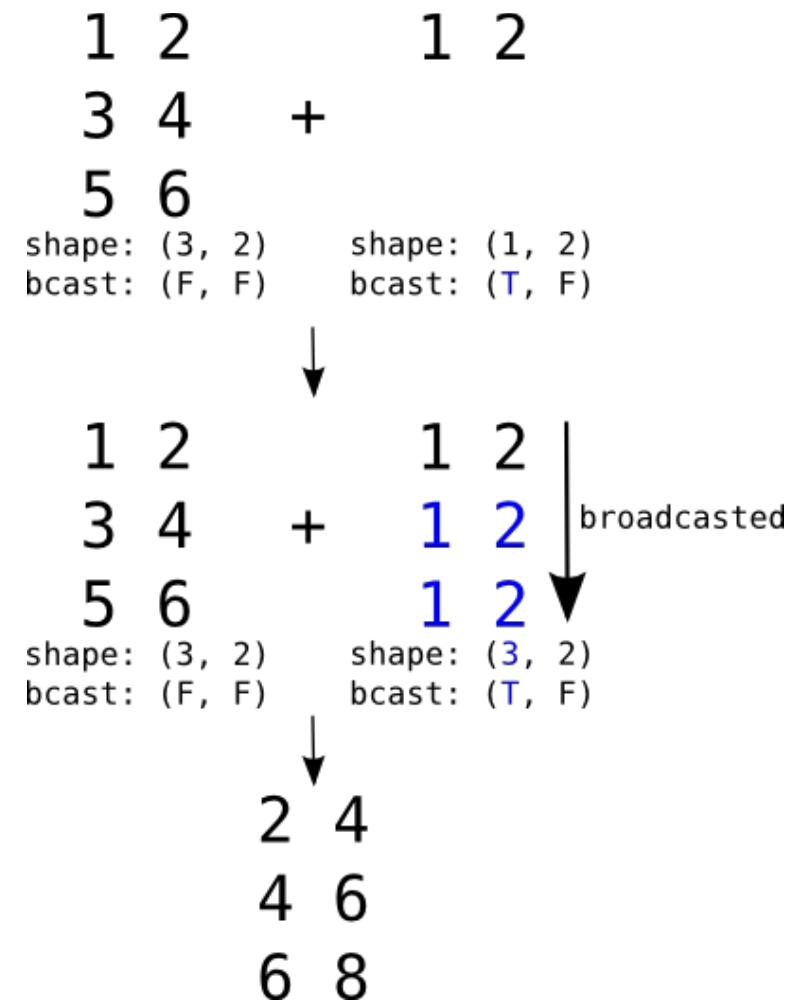
$$\mathbf{A} \odot \mathbf{B} := [a_{ij}b_{ij}]$$

- Hadamard product is commutative (unlike ordinary matmul)

$$\mathbf{A} \odot \mathbf{B} = \mathbf{B} \odot \mathbf{A}$$

Broadcasting

- A shorthand notation commonly used in ML.
- Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^m$, under broadcasting convention, the expression $\mathbf{B} = \mathbf{A} + \mathbf{x}$ is assumed to be equivalent to $b_{ij} = a_{ij} + x_j$.



Tensor Dot Product

- Most common, most useful tensor operation.
- Not to be confused with an element-wise (Hadamard) product.

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x} = \sum_i x_i y_i \text{ (vector dot)}$$

$$\mathbf{A} \cdot \mathbf{y} = \sum_j A_{i,j} y_j \text{ (matrix-vector dot)}$$

$$\mathbf{A} \cdot \mathbf{B} = \sum_j A_{i,j} B_{j,k} \text{ (matrix-matrix dot)}$$

$$\mathbf{X} \cdot \mathbf{Y} = \sum_l X_{i,j,k,l} Y_{l,m,n} \text{ (tensor-tensor dot)}$$

Moore-Penrose Pseudoinverse

- Consider a linear system: $\mathbf{XW} = \mathbf{Y}$, where $\mathbf{X} \in \mathbb{R}^{N \times d}$, $\mathbf{W} \in \mathbb{R}^{d \times 1}$, and $\mathbf{Y} \in \mathbb{R}^{N \times 1}$
- Find \mathbf{W} .
 - If $N = d$, $\exists \mathbf{X}$: ** if sq-matrix...
 - $\mathbf{W} = \mathbf{X}^{-1}\mathbf{Y}$
 - If $N > d$?
 - Least-square solution: $\min_{\mathbf{W}} \|\mathbf{Y} - \mathbf{XW}\|^2$
 - $$\begin{aligned} \mathcal{L}(\mathbf{W}) &= \|\mathbf{Y} - \mathbf{XW}\|^2 = (\mathbf{Y} - \mathbf{XW})'(\mathbf{Y} - \mathbf{XW}) \\ &= \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{XW} - (\mathbf{XW})'\mathbf{Y} + (\mathbf{XW})'(\mathbf{XW}) \quad \text{** } (\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T \\ &= \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{XW} - \mathbf{Y}'\mathbf{XW} + \mathbf{W}'\mathbf{X}'\mathbf{XW} \quad \text{(why?)} \end{aligned}$$
 - $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}(\mathbf{W}) = -2\mathbf{Y}'\mathbf{X} + 2\mathbf{W}'\mathbf{X}'\mathbf{X} \equiv \mathbf{0}$ (if you're not comfortable with vector calculus: <http://matrixcookbook.com/>)
 - $\mathbf{Y}'\mathbf{X} = \mathbf{W}'\mathbf{X}'\mathbf{X}$
 - $\mathbf{X}'\mathbf{XW} = \mathbf{X}'\mathbf{Y}$
 - $\mathbf{W} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ (if $(\mathbf{X}'\mathbf{X})$ non-singular)

Moore-Penrose Pseudoinverse:
$$\mathbf{X}^+ := (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$$

Linear Regression

$$\mathbf{x}^{(t)} \in \mathbb{R}^{1 \times d}, \quad 1 \leq t \leq N$$

$$r^{(t)} \in \mathbb{R}, \quad 1 \leq t \leq N$$



$$\mathbf{X} = \begin{bmatrix} \vdots \\ -\mathbf{x}^{(t)}- \\ \vdots \end{bmatrix} \in \mathbb{R}^{N \times d} \quad \mathbf{R} = \begin{bmatrix} \vdots \\ r^{(t)} \\ \vdots \end{bmatrix} \in \mathbb{R}^{N \times 1}$$

Linear Regression

Linear Model:

$$\mathbf{R} = \mathbf{X}\mathbf{W} + b$$

Model Parameters: $\mathbf{W} \in \mathbb{R}^{d \times 1}$, $b \in \mathbb{R}$

Broadcasting adding scalar to vector...

A "trick":

$$\mathbf{X} \rightarrow [\mathbf{X} \quad \mathbf{1}]$$

$$\boldsymbol{\Theta} := \begin{bmatrix} \mathbf{W} \\ b \end{bmatrix}$$

$$\mathbf{R} = \mathbf{X}\mathbf{W} + b \quad \rightarrow \quad \mathbf{R} = \mathbf{X}\boldsymbol{\Theta}$$

$$\boldsymbol{\Theta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{R}$$

Singular Value Decomposition (SVD)

- $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}'$ samples x attributes, N x d
 - $\Sigma \in \mathbb{R}^{N \times d}$: a diagonal matrix (singular values)
 - $\mathbf{U} \in \mathbb{R}^{N \times N}$: an orthogonal matrix (left singular vectors)
 - Eigenvectors of \mathbf{XX}'
 - $\mathbf{V} \in \mathbb{R}^{d \times d}$: an orthogonal matrix (right singular vectors)
 - Eigenvectors of $\mathbf{X}'\mathbf{X}$
- $\mathbf{X}^+ = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' = \mathbf{V}\Sigma^+\mathbf{U}'$
 - Σ^+ : pseudoinverse of a diagonal matrix is calculated as...?

$$\begin{array}{c} \text{M} \\ m \times n \end{array} = \begin{array}{c} \mathbf{U} \\ m \times m \end{array} \begin{array}{c} \Sigma \\ m \times n \end{array} \begin{array}{c} \mathbf{V}^* \\ n \times n \end{array}$$
$$\begin{array}{c} \mathbf{U} \\ m \times m \end{array} \quad \begin{array}{c} \mathbf{U}^* \\ m \times m \end{array} = \mathbf{I}_m$$
$$\begin{array}{c} \mathbf{V} \\ n \times n \end{array} \quad \begin{array}{c} \mathbf{V}^* \\ n \times n \end{array} = \mathbf{I}_n$$

Singular Value Decomposition (SVD)

- $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}'$

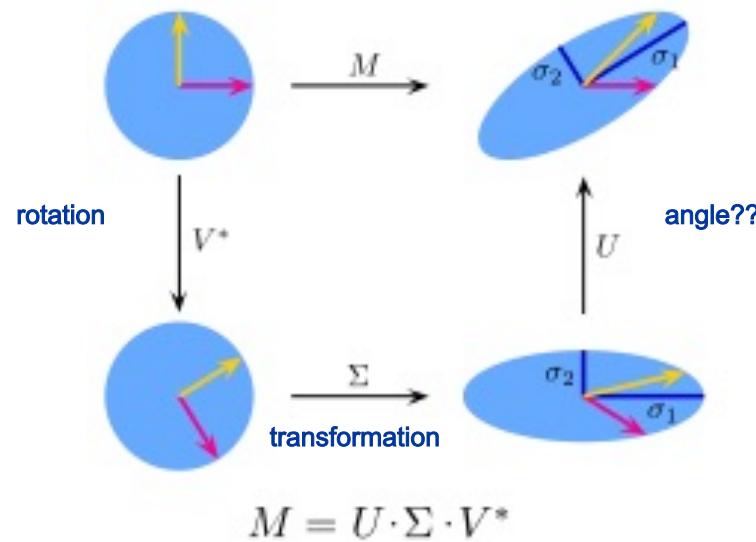


Image Classification

Why Image Classification?

- Images are more straightforward and intuitive subject to study deep networks
- Easy to generalize to other forms of data
- Major breakthroughs in deep learning research were in computer vision
- Image classification = core task in computer vision

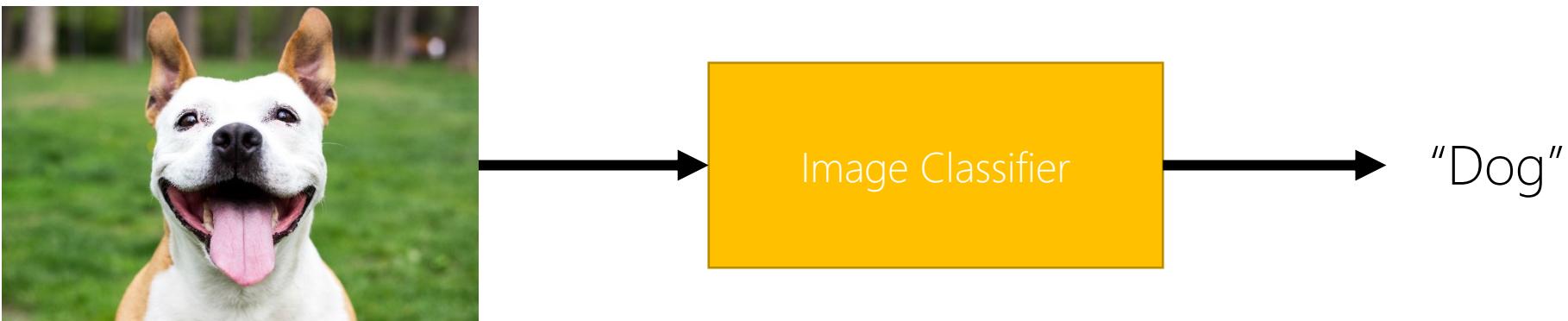


Image Classification is Challenging!

- Problem 1. Semantic Challenge



What we see

127	123	101	40	12	13	14	22	26	18	13	13	13	13	13	15
114	81	98	38	13	14	28	36	34	29	17	13	13	13	13	15
142	102	87	33	13	15	31	38	34	36	22	13	13	14	15	17
132	112	87	24	14	22	34	52	46	37	26	20	14	15	16	18
120	103	83	19	15	18	57	158	161	131	24	19	14	15	15	17
117	110	84	18	14	21	114	163	171	128	49	17	15	15	15	17
117	112	89	21	14	26	155	163	174	157	117	15	16	16	16	17
117	103	84	21	15	13	120	174	106	182	87	11	15	15	16	17
117	103	86	21	16	16	56	158	125	156	55	19	21	16	16	17
112	102	94	23	17	29	33	86	97	80	44	38	40	26	15	16
107	91	95	25	19	33	39	98	77	105	65	42	35	40	17	16
98	95	97	26	25	29	39	101	108	70	70	39	29	46	20	16
111	92	91	34	34	23	27	64	144	89	54	23	30	42	31	16
113	96	90	36	32	28	21	31	92	83	47	21	35	42	42	17
119	109	87	34	31	33	45	51	45	134	128	36	33	40	35	17
98	90	78	29	28	34	28	29	39	70	51	65	64	35	23	16

What computers see
= just a grid of numbers (e.g. 1024 x 768 x 3)

Image Classification is Challenging!

- Problem 2. Varying Perspectives



Image Classification is Challenging!

- Problem 3. Illumination



<http://www.ifp.illinois.edu/~tchen5/vlfr.html>

Image Classification is Challenging!

- Problem 4. Occlusion

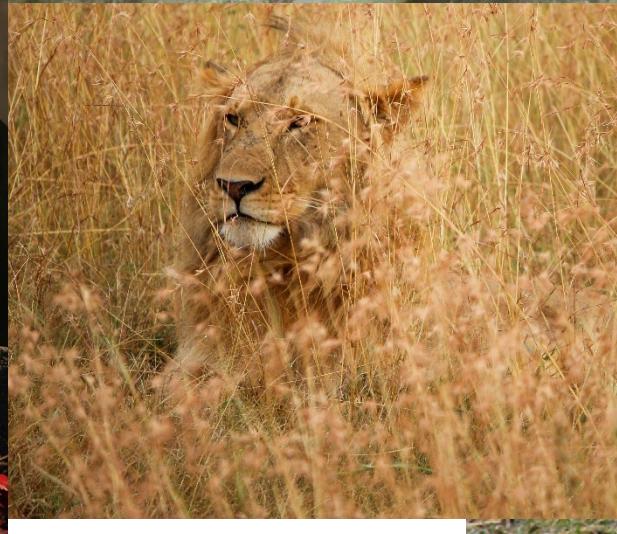


Image Classification is Challenging!

- Problem 5. Deformation, Shape & Posture Change



Image Classification is Challenging!

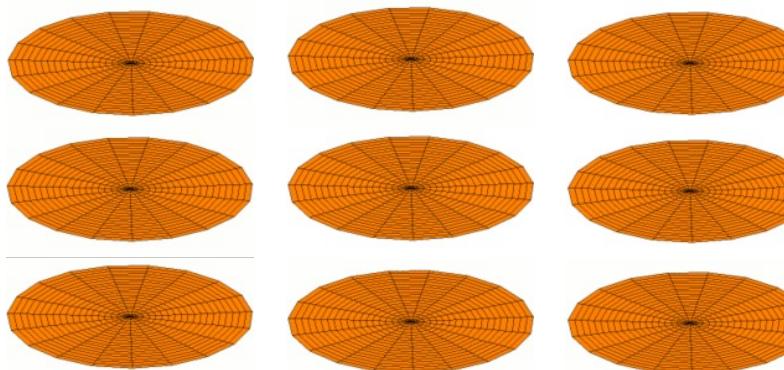
- Problem 6. Intraclass Variation



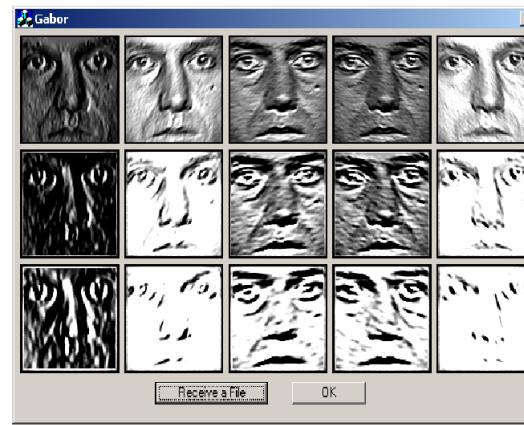
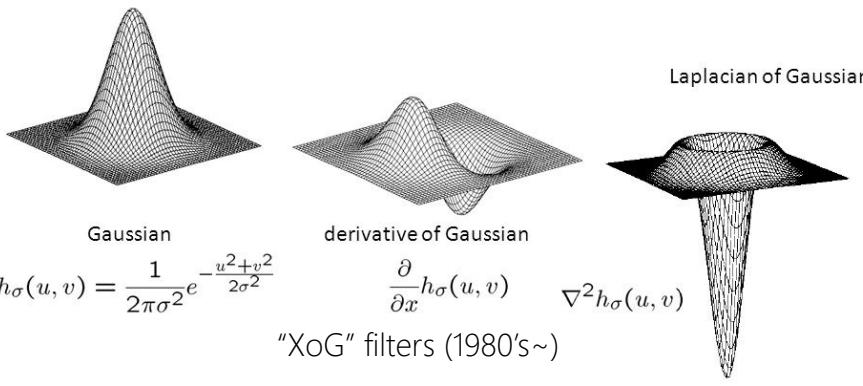
Arman, Alarm Clocks (Réveils), 1960, MCA Chicago

Conventional Approach (Model-Driven)

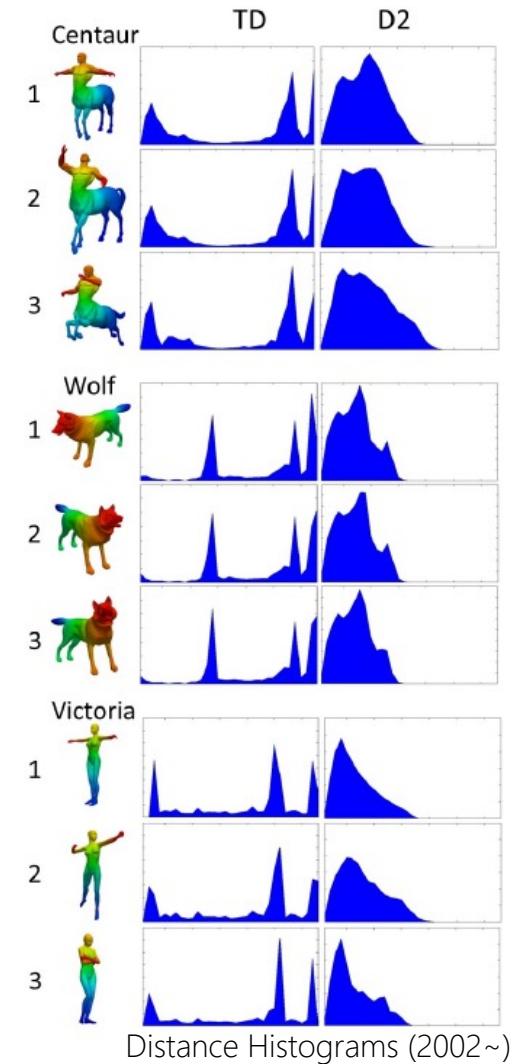
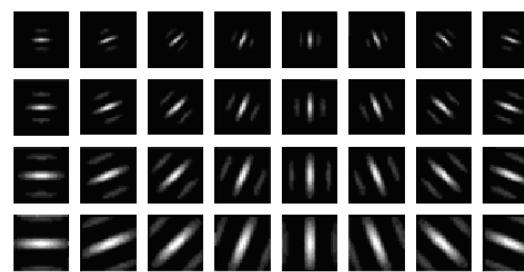
manually finding $f(x)$



Laplace-Beltrami Spectra (2000's~)



Gabor Filter (1998~)

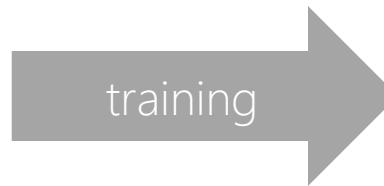


Machine Learning (Data-Driven)

don't need to
manually find $f(x)$,
instead use dataS

Data with Labels

airplane	
automobile	
bird	
cat	
deer	
dog	
frog	
horse	
ship	
truck	



Simplest ML Model: Linear Classifier

- Idea: Formulate the image classification problem as $y = \mathbf{Wx} + \mathbf{b}$ where \mathbf{x} is an image and y is the image label.

Simplest ML Model: Linear Classifier

- Idea: Formulate the image classification problem as $y = \mathbf{Wx} + \mathbf{b}$ where \mathbf{x} is an image and y is the image label.
- Our machine learning model f is then defined as $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$. \mathbf{W} and \mathbf{b} are called model parameters.

Simplest ML Model: Linear Classifier

- Idea: Formulate the image classification problem as $y = \mathbf{Wx} + \mathbf{b}$ where \mathbf{x} is an image and y is the image label.
- Our machine learning model f is then defined as $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$. \mathbf{W} and \mathbf{b} are called model parameters.
- Goal: Find \mathbf{W} and \mathbf{b} that minimize $|y - f(\mathbf{x} | \mathbf{W}, \mathbf{b})|^2$

Simplest ML Model: Linear Classifier

- Idea: Formulate the image classification problem as $y = \mathbf{Wx} + \mathbf{b}$ where \mathbf{x} is an image and y is the image label.
- Our machine learning model f is then defined as $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$. \mathbf{W} and \mathbf{b} are called model parameters.
- Goal: Find \mathbf{W} and \mathbf{b} that minimize $|y - f(\mathbf{x} | \mathbf{W}, \mathbf{b})|^2$
 - In English: I want the predicted label of my model $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$ to be as close as possible to the true label y and I'm going to accomplish that by tuning \mathbf{W} and \mathbf{b} .

Simplest ML Model: Linear Classifier

- Idea: Formulate the image classification problem as $y = \mathbf{Wx} + \mathbf{b}$ where \mathbf{x} is an image and y is the image label.
- Our machine learning model f is then defined as $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$. \mathbf{W} and \mathbf{b} are called model parameters.
- Goal: Find \mathbf{W} and \mathbf{b} that minimize $|y - f(\mathbf{x} | \mathbf{W}, \mathbf{b})|^2$
 - In English: I want the predicted label of my model $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$ to be as close as possible to the true label y and I'm going to accomplish that by tuning \mathbf{W} and \mathbf{b} .
 - Based on what, though?

Simplest ML Model: Linear Classifier

- Idea: Formulate the image classification problem as $y = \mathbf{W}\mathbf{x} + \mathbf{b}$ where \mathbf{x} is an image and y is the image label.
- Our machine learning model f is then defined as $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$. \mathbf{W} and \mathbf{b} are called model parameters.
- Goal: Find \mathbf{W} and \mathbf{b} that minimize $|y - f(\mathbf{x} | \mathbf{W}, \mathbf{b})|^2$
 - In English: I want the predicted label of my model $f(\mathbf{x} | \mathbf{W}, \mathbf{b})$ to be as close as possible to the true label y and I'm going to accomplish that by tuning \mathbf{W} and \mathbf{b} .
 - Based on what, though? Lots of labeled data! $(\mathbf{x}_i, y_i)_{i=1,\dots,N}$

Simplest ML Model: Linear Classifier

- Formulation:

$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N |y_i - f(\mathbf{x}_i \mid \mathbf{W}, \mathbf{b})|^2$$

Simplest ML Model: Linear Classifier

- Formulation:

$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N |y_i - f(\mathbf{x}_i \mid \mathbf{W}, \mathbf{b})|^2$$

- Solution?

Simplest ML Model: Linear Classifier

- Formulation:

$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N |y_i - f(\mathbf{x}_i \mid \mathbf{W}, \mathbf{b})|^2$$

- Solution?
 - Linear least squares/Moore-Penrose Pseudoinverse
 - Iterative optimization
 - 1st order/gradient-based methods
 - 2nd order/Hessian-based/(quasi)Newton methods
 - ...
 - ...

Simplest ML Model: Linear Classifier

- Formulation:

$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N |y_i - f(\mathbf{x}_i \mid \mathbf{W}, \mathbf{b})|^2$$

- Solution?
 - Linear least squares/Moore-Penrose Pseudoinverse
 - Iterative optimization
 - 1st order/gradient-based methods
 - 2nd order/Hessian-based/(quasi)Newton methods
 - ...
 - ...
 - We are not going to worry about this now. (We'll re-visit this later.)

Linear Classifier: Step-by-Step

1. Collect a bunch of data $(\mathbf{x}_i, y_i)_{i=1,\dots,N}$
2. Split them into two sets: training set and test set
3. Solve $\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^{N_{\text{train}}} |y_i - f(\mathbf{x}_i \mid \mathbf{W}, \mathbf{b})|^2$
4. See if $y - f(\mathbf{x} \mid \mathbf{W}, \mathbf{b})$ holds true for the test set
 - If it doesn't: sit and cry (or build a better model than the linear classifier)
 - If it does: profit! the best step^^

Example: Cat/Dog Classifier

- The cats and dogs dataset: <https://www.microsoft.com/en-us/download/details.aspx?id=54765>
- 12,500 cat images & 12,500 dog images from internet



Example: Cat/Dog Classifier

$$(\text{ (image of a cat) }, -1)$$
$$(\text{ (image of a cat) }, -1)$$
$$(\text{ (image of a cat) }, -1)$$

...

$$(\text{ (image of a dog) }, 1)$$
$$(\text{ (image of four puppies) }, 1)$$
$$(\text{ (image of two dogs) }, 1)$$

...

Example: Cat/Dog Classifier



width = 400

height = 500

channels = 3

Example: Cat/Dog Classifier



width = 400
height = 500
channels = 3

Example: Cat/Dog Classifier



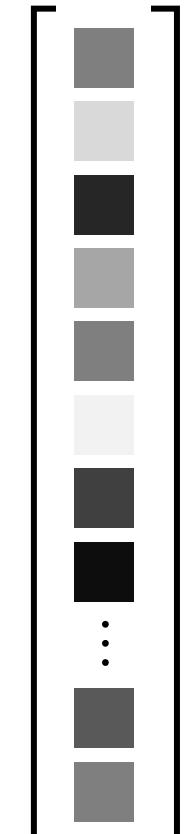
width = 400
height = 500
channels = 3

gray + resize



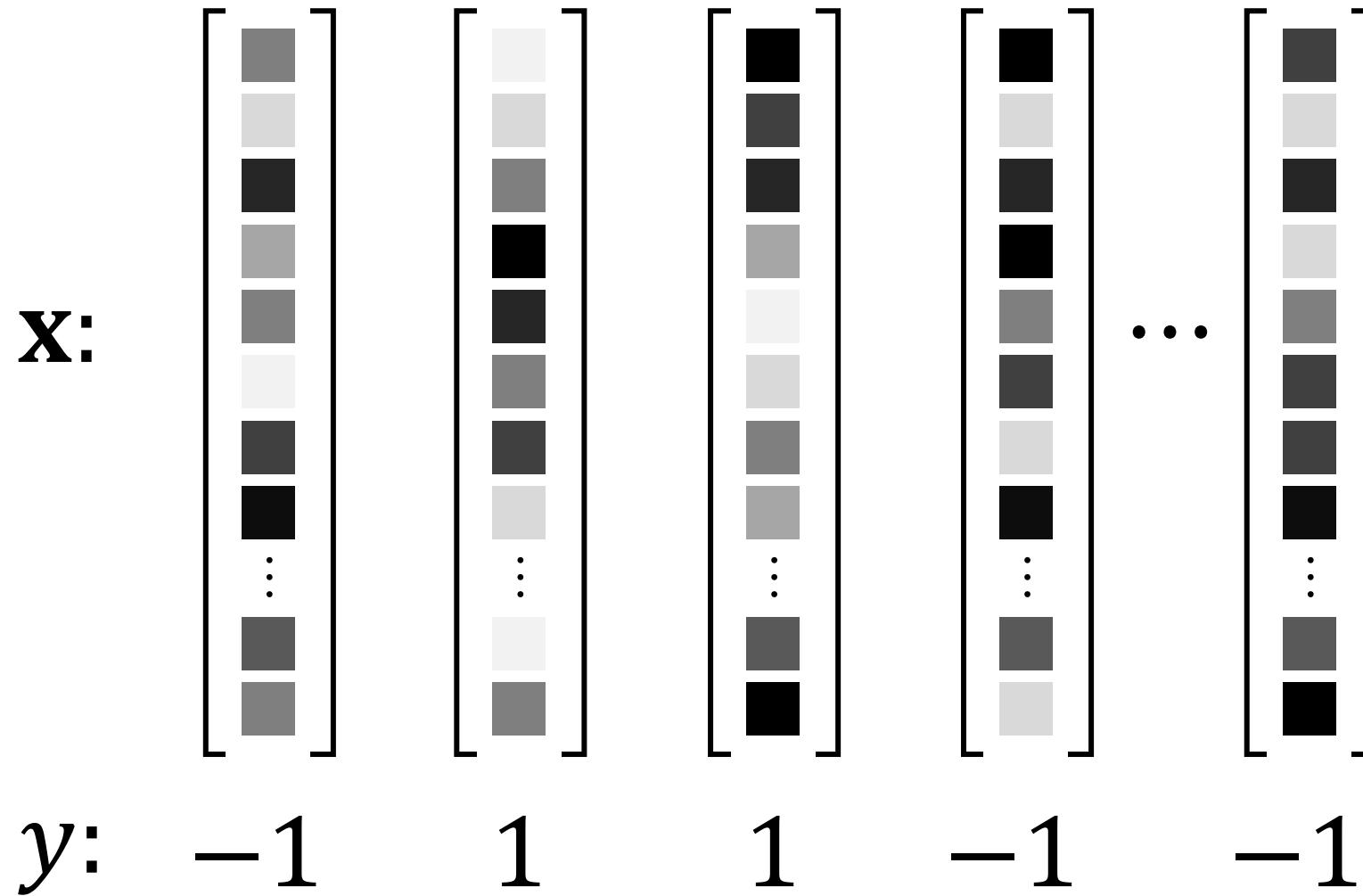
64 x 64

flatten

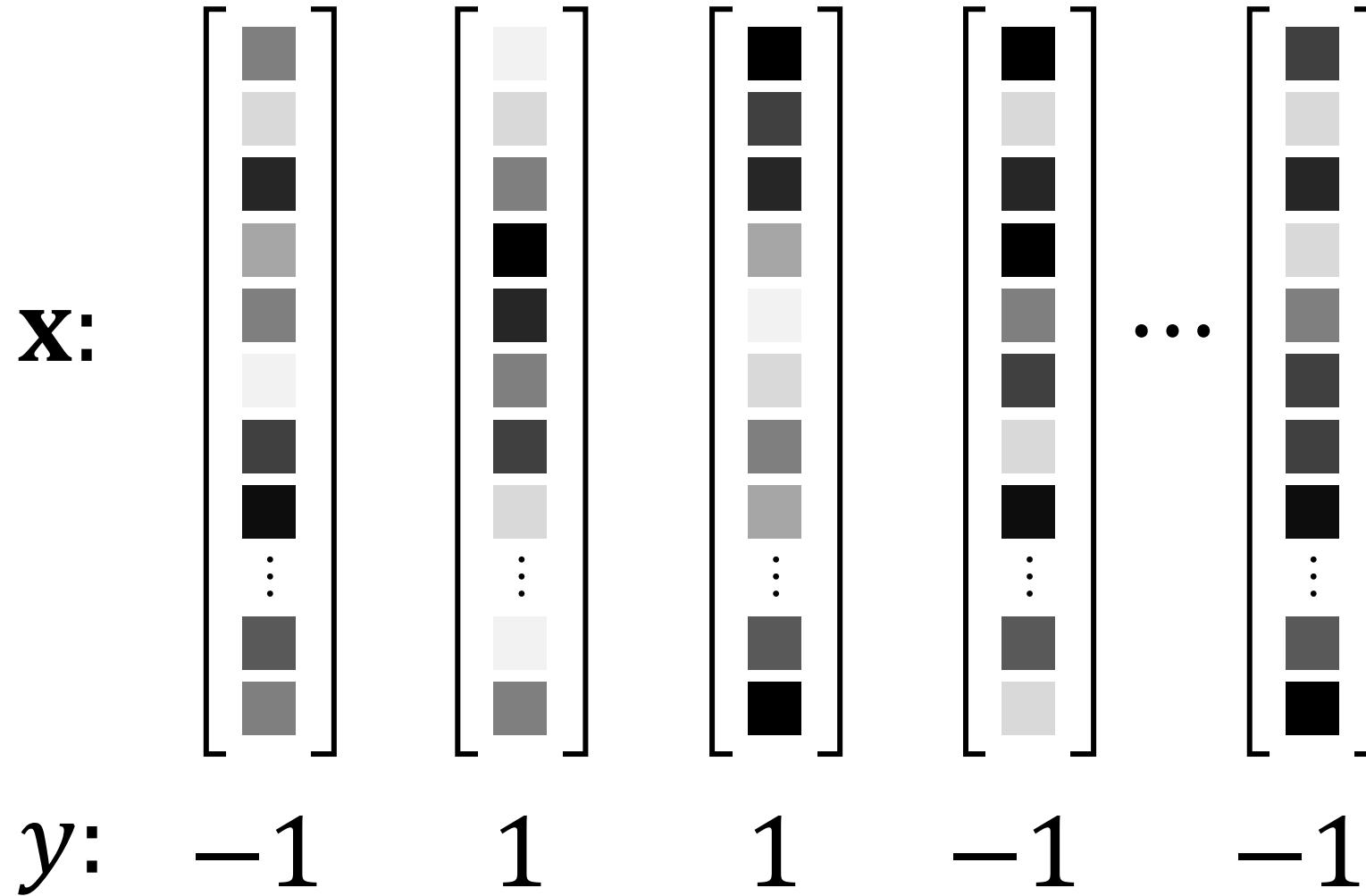


4096 x 1

Example: Cat/Dog Classifier



Example: Cat/Dog Classifier



$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N |\mathbf{W}\mathbf{x}_i + \mathbf{b} - y_i|^2$$

x_i is a col vector

W is a row vector *1 x 4096* / *1 x 1* *b is scalar*

Example: Cat/Dog Classifier

X:	$\begin{bmatrix} \dots \\ \vdots \\ \dots \end{bmatrix}$	$\begin{bmatrix} \dots \\ \vdots \\ \dots \end{bmatrix}$	$\begin{bmatrix} \dots \\ \vdots \\ \dots \end{bmatrix}$	\dots	$\begin{bmatrix} \dots \\ \vdots \\ \dots \end{bmatrix}$
y:	-1	1	1	-1	-1

$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N |\mathbf{W}\mathbf{x}_i + \mathbf{b} - y_i|^2$$



$$\mathbf{W}^*, \mathbf{b}^*$$

Example: Cat/Dog Classifier

(Online)



Query

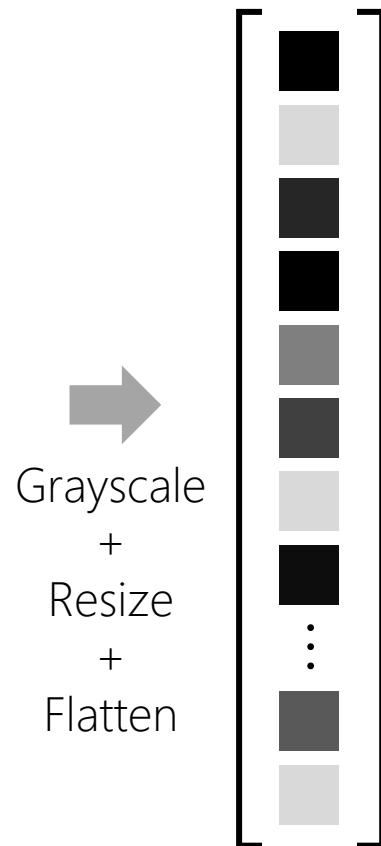
(*Never seen by the model)

Example: Cat/Dog Classifier

(Online)



Query

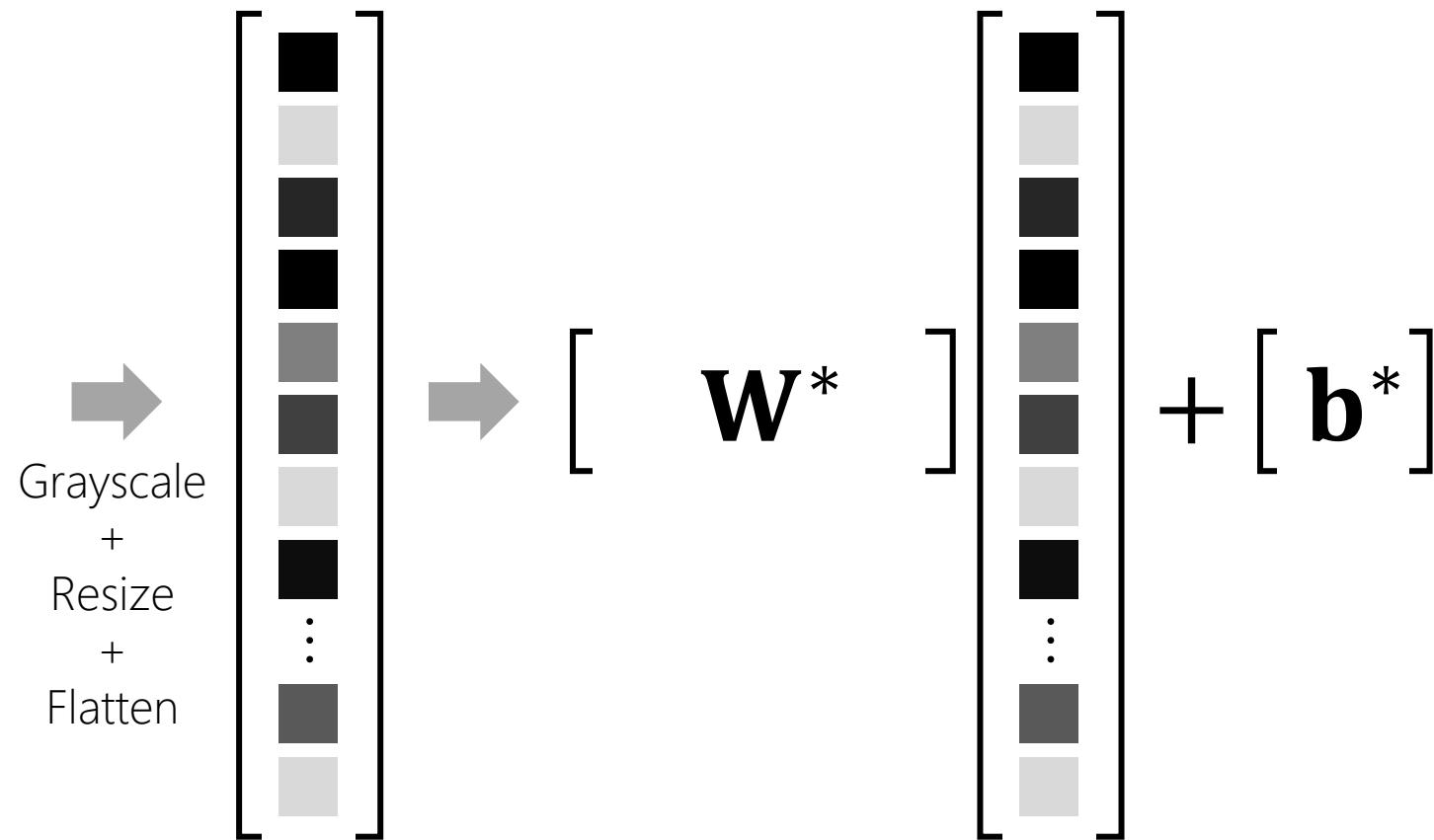


Example: Cat/Dog Classifier

(Online)



Query

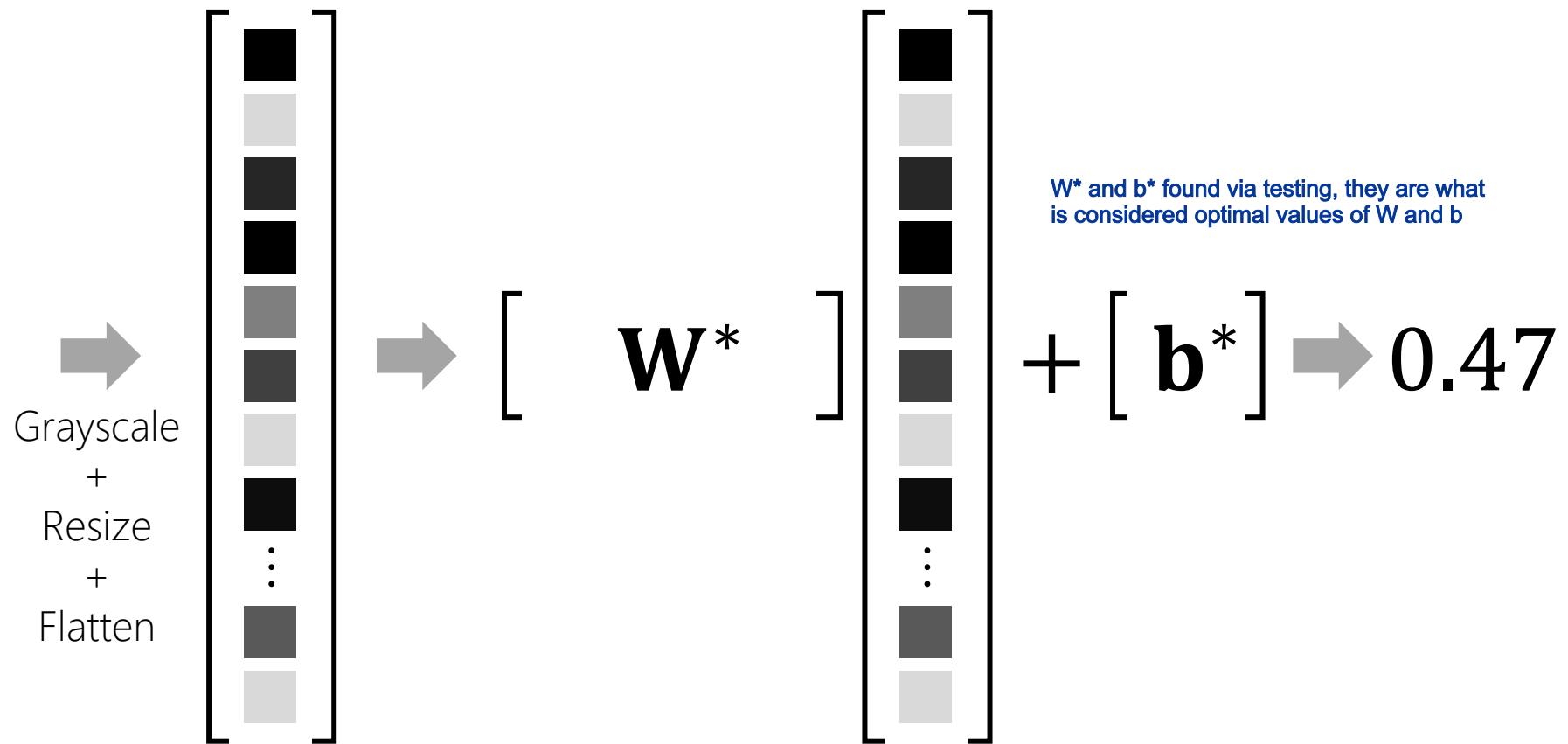


Example: Cat/Dog Classifier

(Online)

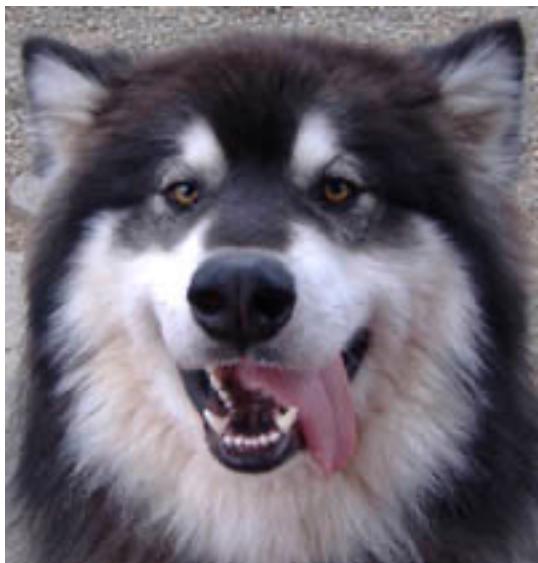


Query

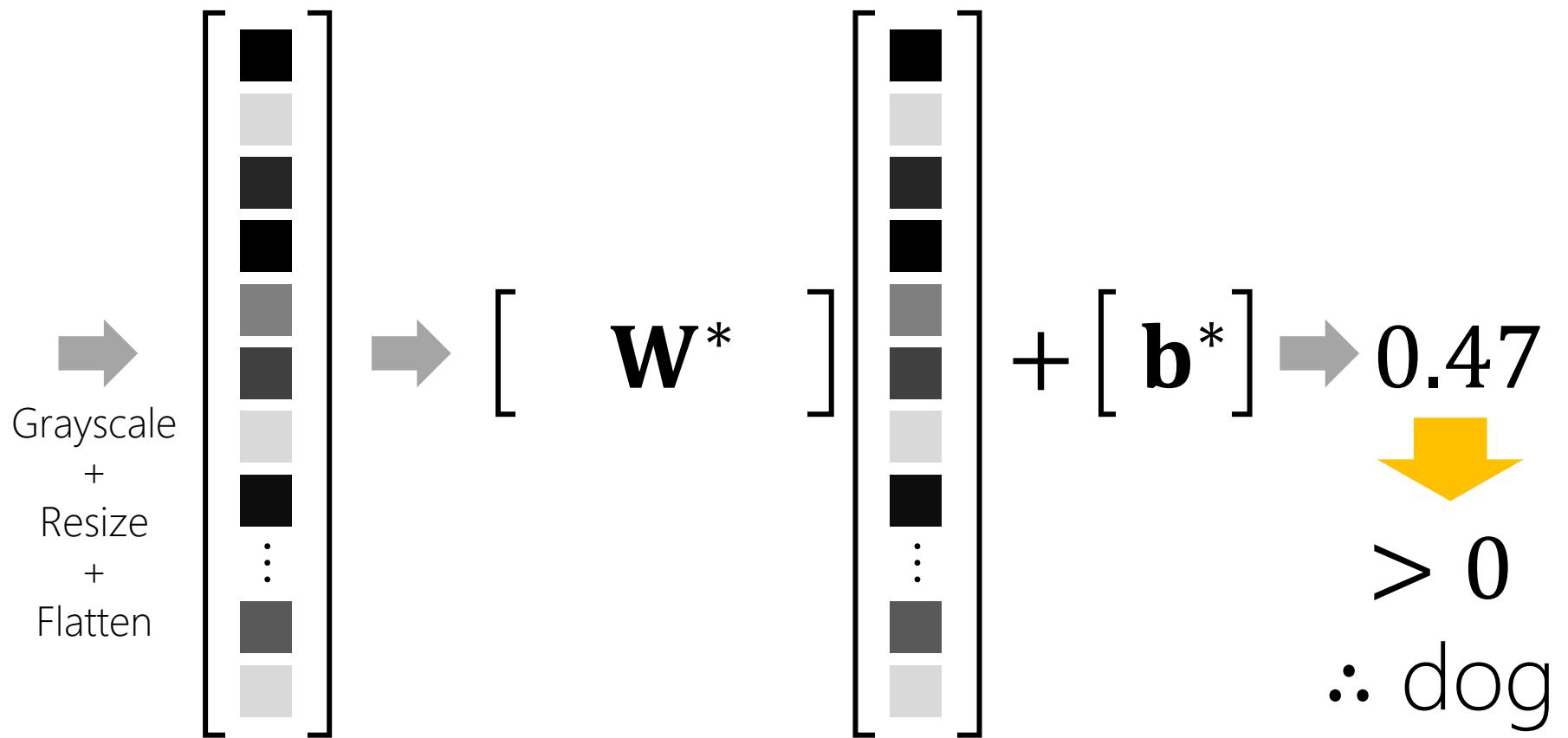


Example: Cat/Dog Classifier

(Online)



Query

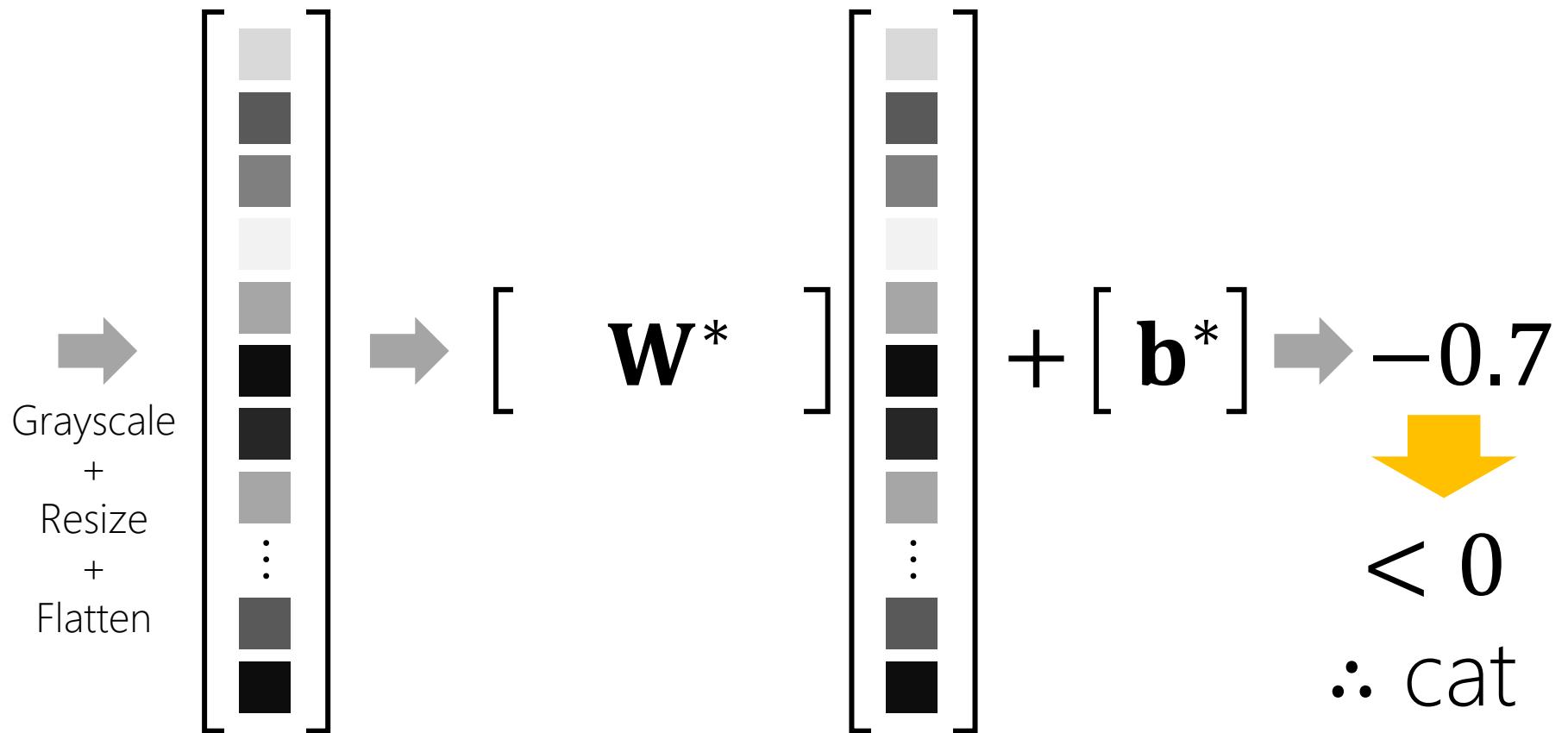


Example: Cat/Dog Classifier

(Online)



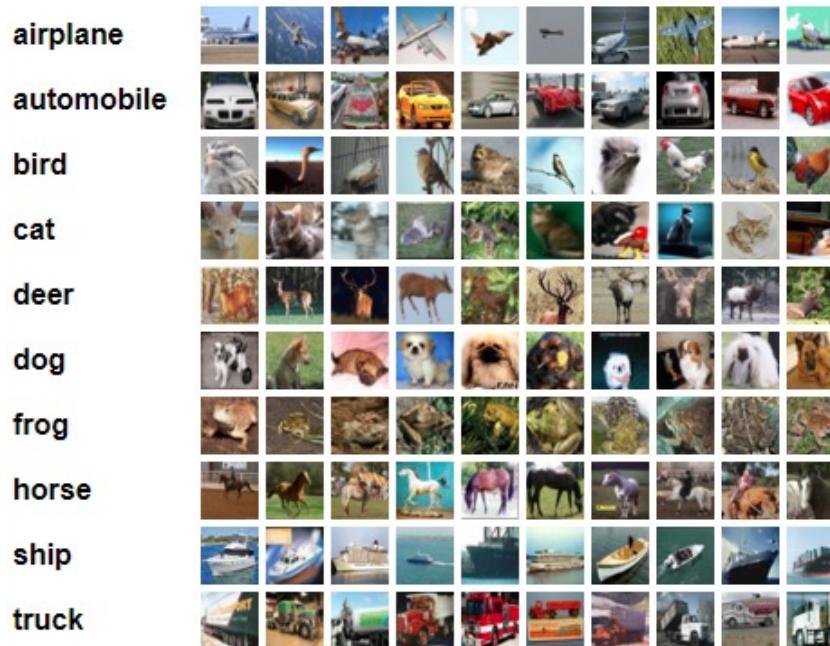
Query



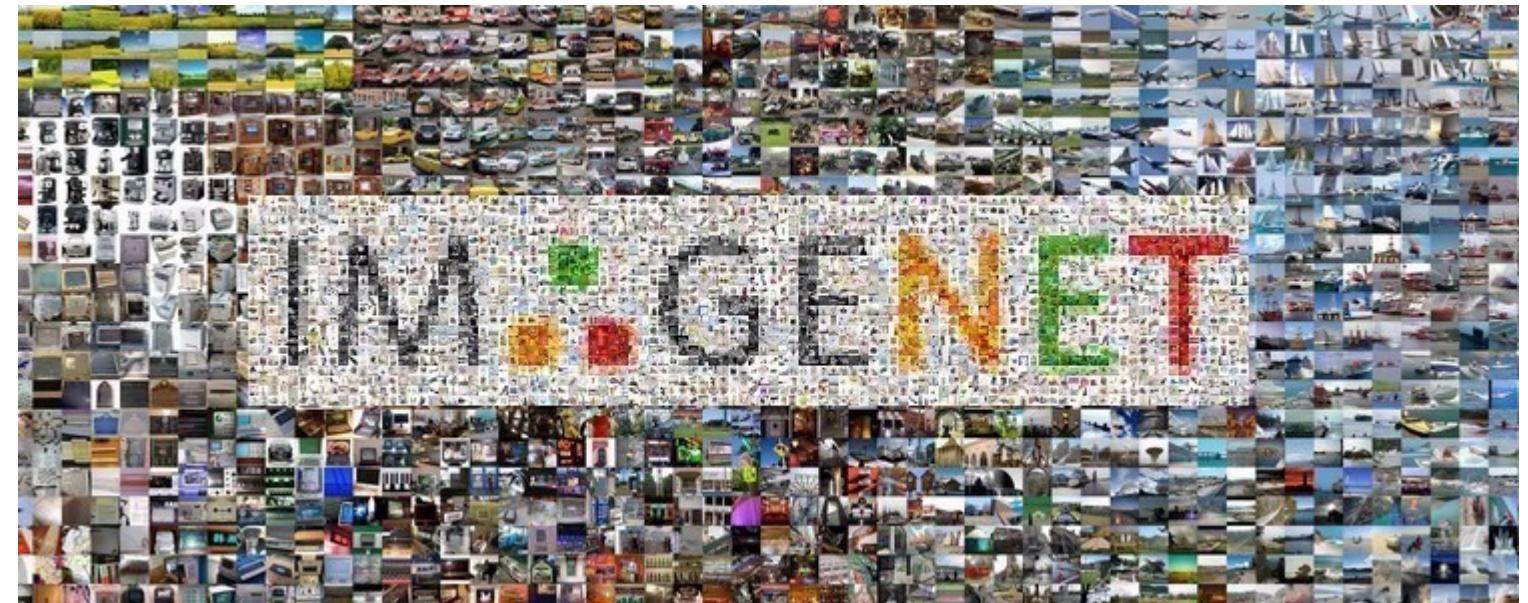
Example: 10-way Classifier (CIFAR-10 Dataset)

- What if there were more than two classes (e.g. CIFAR, ImageNet)?

CIFAR-10: 60,000 images in 10 classes



ImageNet: 14,197,122 images in 21,841 classes
ILSVRC: 1.4M images in 1,000 classes



Probability Mass Function

$$\begin{array}{l} \text{airplane} \\ \text{automobile} \\ \text{bird} \\ \text{cat} \\ \text{deer} \\ \text{dog} \\ \text{frog} \\ \text{horse} \\ \text{ship} \\ \text{truck} \end{array} \rightarrow \begin{bmatrix} y \\ P_{\text{airplane}} \\ P_{\text{automobile}} \\ P_{\text{bird}} \\ P_{\text{cat}} \\ P_{\text{deer}} \\ P_{\text{dog}} \\ P_{\text{frog}} \\ P_{\text{horse}} \\ P_{\text{ship}} \\ P_{\text{truck}} \end{bmatrix}$$

$0 \leq P_i \leq 1, \forall i$

$\sum_i P_i = 1$

“One-hot” Encoding



True PMF

y

1	airplane
0	automobile
0	bird
0	cat
0	deer
0	dog
0	frog
0	horse
0	ship
0	truck

“One-hot” Encoding



an ideal vector, where model is 100% confident that the picture is of an airplane

True PMF

y

1	airplane
0	automobile
0	bird
0	cat
0	deer
0	dog
0	frog
0	horse
0	ship
0	truck

what it actually looks like, the model isn't 100% confident on the image classification

What ML model predicts

f

0.7
0.1
0.05
0
0
0
0
0
0.15
0



Cross-entropy

$$-\sum_i y_i \log(f_i)$$

In English: How similar are the two distributions?
(We will re-visit this later to learn how and why.)

True PMF

y

$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

What ML model predicts

f

$\begin{bmatrix} 0.7 \\ 0.1 \\ 0.05 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.15 \\ 0 \end{bmatrix}$

Example: 10-way Classifier (CIFAR-10 Dataset)



$$\mathbf{x}_1 = \begin{matrix} \text{airplane} \\ \text{automobile} \\ \text{bird} \\ \text{cat} \\ \text{deer} \\ \text{dog} \\ \text{frog} \\ \text{horse} \\ \text{ship} \\ \text{truck} \end{matrix} \dots \quad y_1 = [1 \quad 0 \quad 0]$$



$$\mathbf{x}_2 = \begin{matrix} \text{airplane} \\ \text{automobile} \\ \text{bird} \\ \text{cat} \\ \text{deer} \\ \text{dog} \\ \text{frog} \\ \text{horse} \\ \text{ship} \\ \text{truck} \end{matrix} \dots \quad y_2 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]$$

⋮

⋮

⋮

Example: 10-way Classifier (CIFAR-10 Dataset)



$$\mathbf{x}_1 = \begin{array}{ccccccccccccc} \text{airplane} & \text{automobile} & \text{bird} & \text{cat} & \text{deer} & \text{dog} & \text{frog} & \text{horse} & \text{ship} & \text{truck} \\ [1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0] \end{array}$$



$$\text{Minimize } - \sum_i y_i \log(f(\mathbf{x}_i | \mathbf{W}, \mathbf{b})) \rightarrow \mathbf{W}^*, \mathbf{b}^*$$

$$\mathbf{x}_2 = \begin{array}{ccccccccccccc} \text{airplane} & \text{automobile} & \text{bird} & \text{cat} & \text{deer} & \text{dog} & \text{frog} & \text{horse} & \text{ship} & \text{truck} \\ [0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0] \end{array}$$

⋮

⋮

⋮

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)



Query

$$\begin{bmatrix} \text{10 x 4096} \\ \mathbf{W}^* \end{bmatrix} \begin{bmatrix} \text{4096 x 1} \\ \vdots \end{bmatrix} + \begin{bmatrix} \text{10 x 1} \\ \mathbf{b}^* \end{bmatrix}$$

The diagram illustrates the computation of a 10-way classification score. It shows the multiplication of a weight matrix \mathbf{W}^* (dimensions 10 x 4096) by a query vector (dimensions 4096 x 1). The result is then added to a bias vector \mathbf{b}^* (dimensions 10 x 1). The vertical ellipsis between the weight matrix and the addition operator indicates that there are 10 such rows, one for each class.

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)



Query

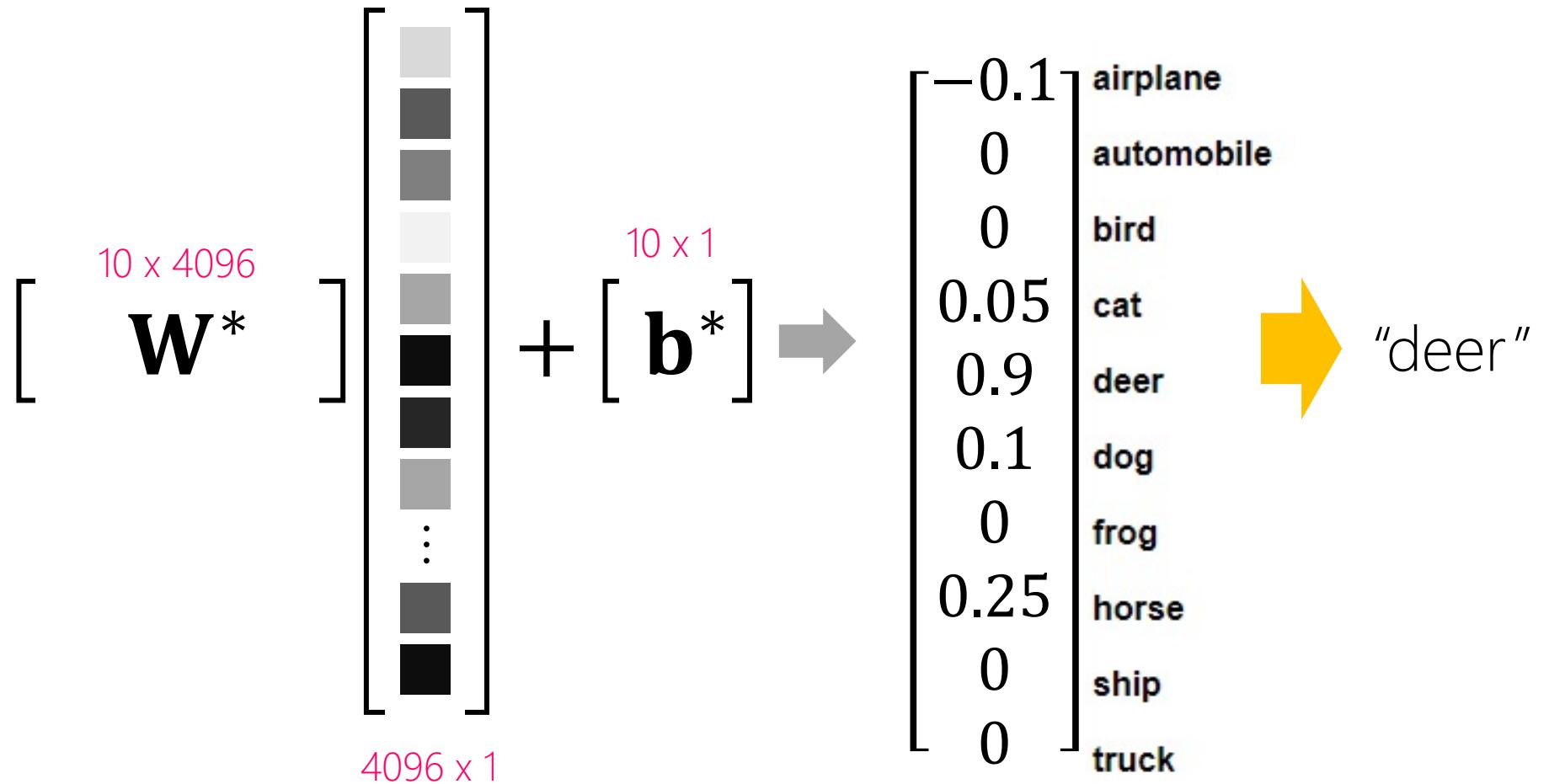
$$\begin{bmatrix} \text{10 x 4096} \\ \mathbf{W}^* \end{bmatrix} \begin{bmatrix} \vdots \\ \text{4096 x 1} \end{bmatrix} + \begin{bmatrix} \text{10 x 1} \\ \mathbf{b}^* \end{bmatrix} \rightarrow \begin{bmatrix} -0.1 \\ 0 \\ 0 \\ 0.05 \\ 0.9 \\ 0.1 \\ 0 \\ 0.25 \\ 0 \\ 0 \end{bmatrix}$$

The diagram illustrates the computation of a 10-way classification output. On the left, a query image of a deer is shown. To its right is a matrix equation representing the forward pass of a neural network layer. The first term is a matrix \mathbf{W}^* of size 10 x 4096 multiplied by a vector of size 4096 x 1. The second term is a matrix \mathbf{b}^* of size 10 x 1. The result of this addition is a vector of size 10 x 1 containing the classification scores for each of the ten categories. The categories and their corresponding scores are listed on the right, with the category names in red and the scores in black. The scores range from -0.1 for 'airplane' to 0.9 for 'deer'.

Category	Score
airplane	-0.1
automobile	0
bird	0
cat	0.05
deer	0.9
dog	0.1
frog	0
horse	0.25
ship	0
truck	0

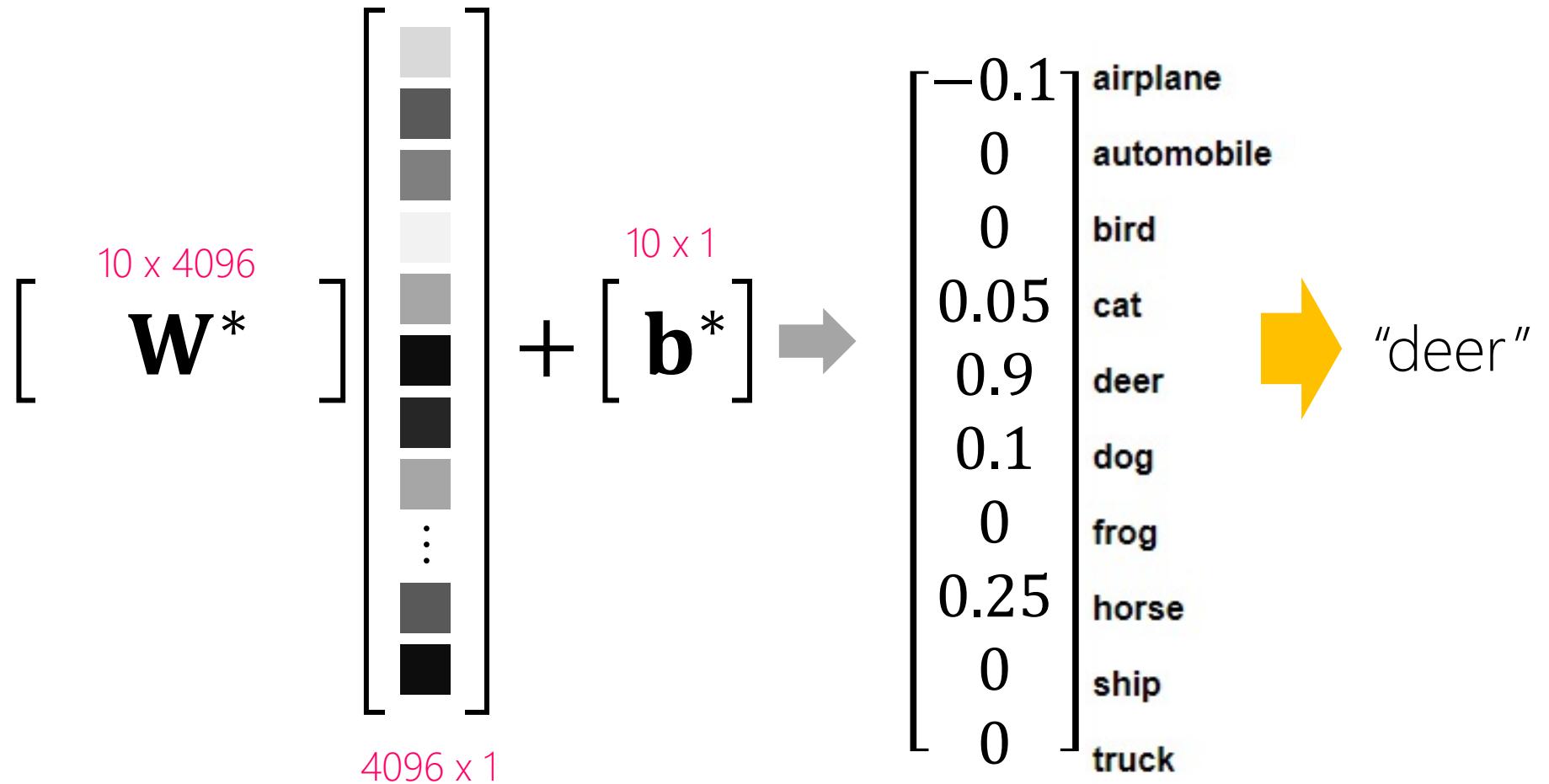
Example: 10-way Classifier (CIFAR-10 Dataset)

(online)



Example: 10-way Classifier (CIFAR-10 Dataset)

(online)

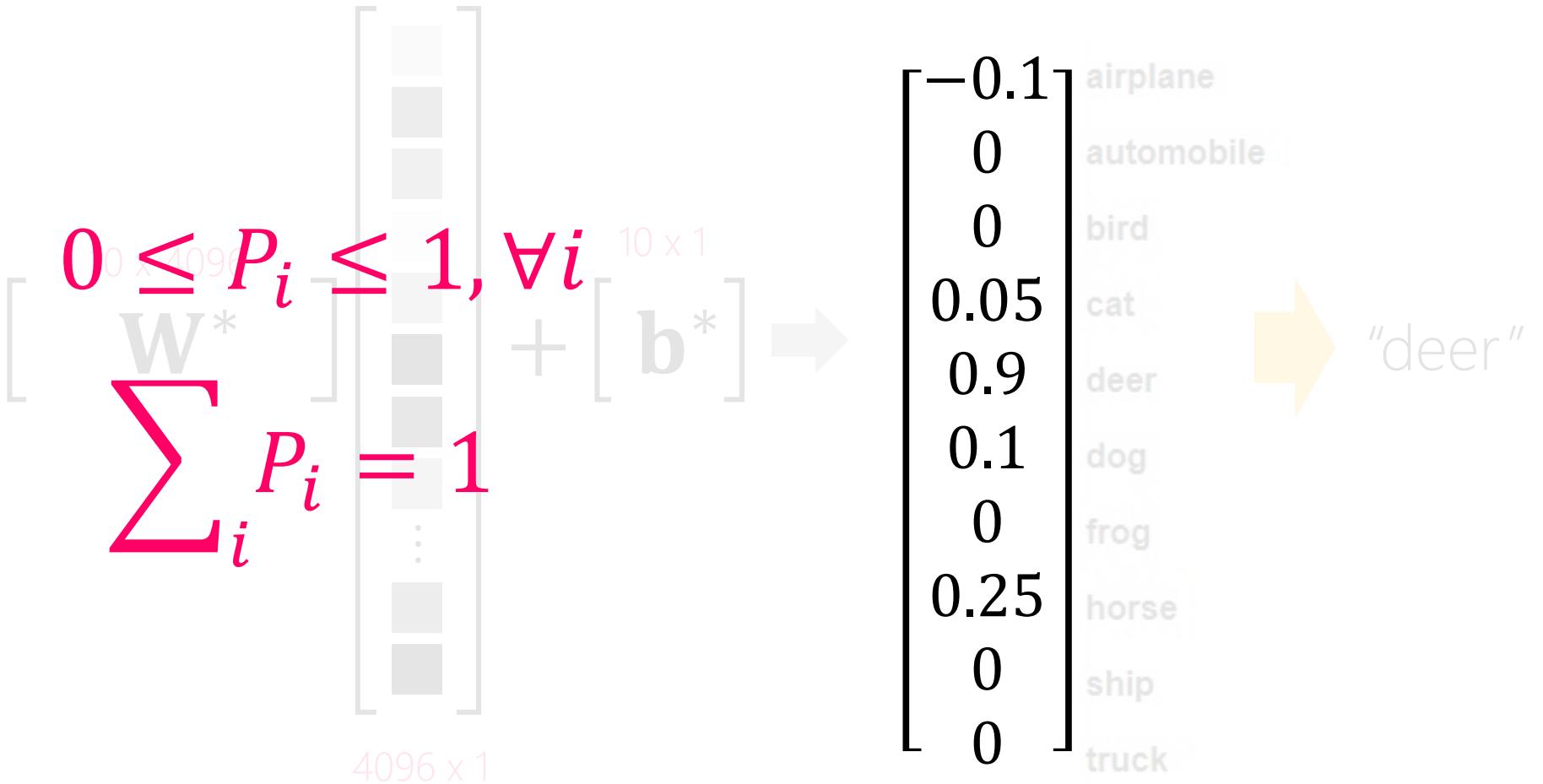


Query

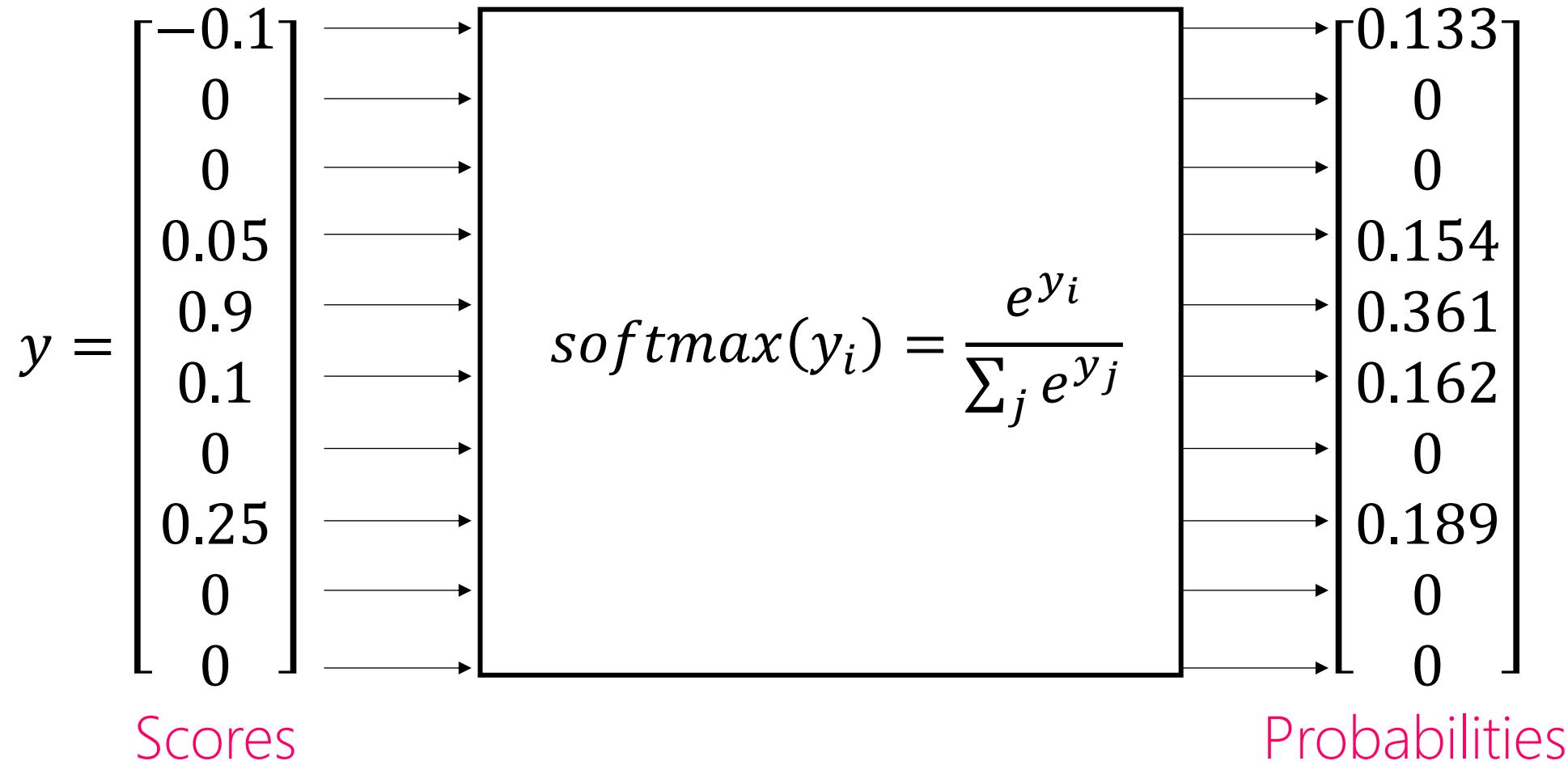
Wait a second, anything wrong?

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)



Softmax!



n-way Softmax Classifier: Step-by-Step

1. Collect a bunch of data $(\mathbf{x}_i, y_i)_{i=1,\dots,N}$
2. Encode y_i in one-hot encoding
3. Split the data into two sets: training set and test set
4. Solve $\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^{N_{\text{train}}} -y_i \log(f_i)$ where $f_i = \text{softmax}(\mathbf{W}\mathbf{x}_i + \mathbf{b})$
5. Test the trained model $f(\mathbf{x} | \mathbf{W}^*, \mathbf{b}^*) = \text{softmax}(\mathbf{W}^*\mathbf{x} + \mathbf{b}^*)$ on the test set