

Security and Fairness of Deep Learning

# Machine Learning Basics

Anupam Datta  
CMU

Spring 2018

# Image Classification

# Image Classification



05	02	21	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	21
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	45	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	57	03	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	44	69	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	02	59	41	92	36	54	22	40	40	28	66	33	13	80
24	47	33	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
52	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
55	16	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	35	95	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	49	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	34	81	16	23	57	05	54
01	70	84	71	83	51	54	69	16	92	33	48	61	43	52	01	89	27	47	48

What the computer sees

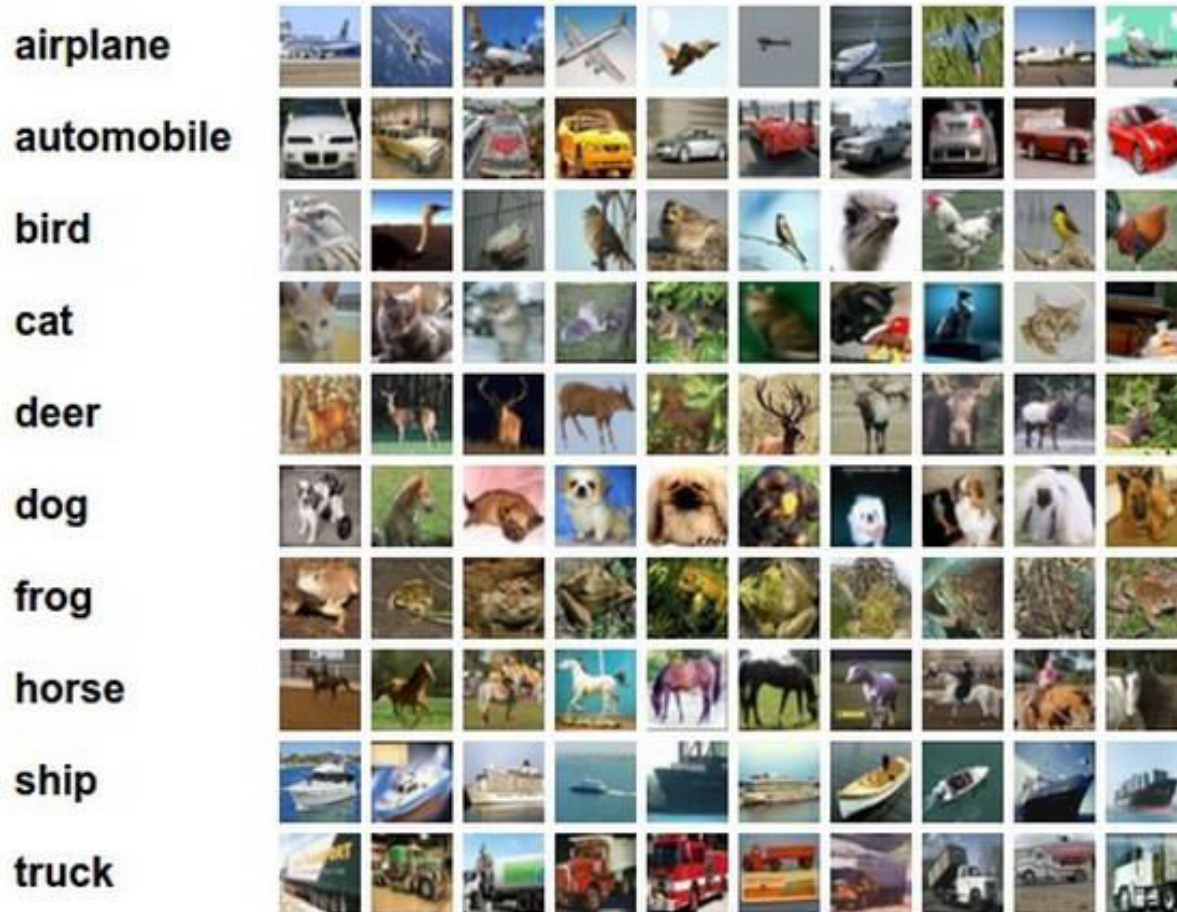
image classification →

- 82% cat
- 15% dog
- 2% hat
- 1% mug

# Image classification pipeline

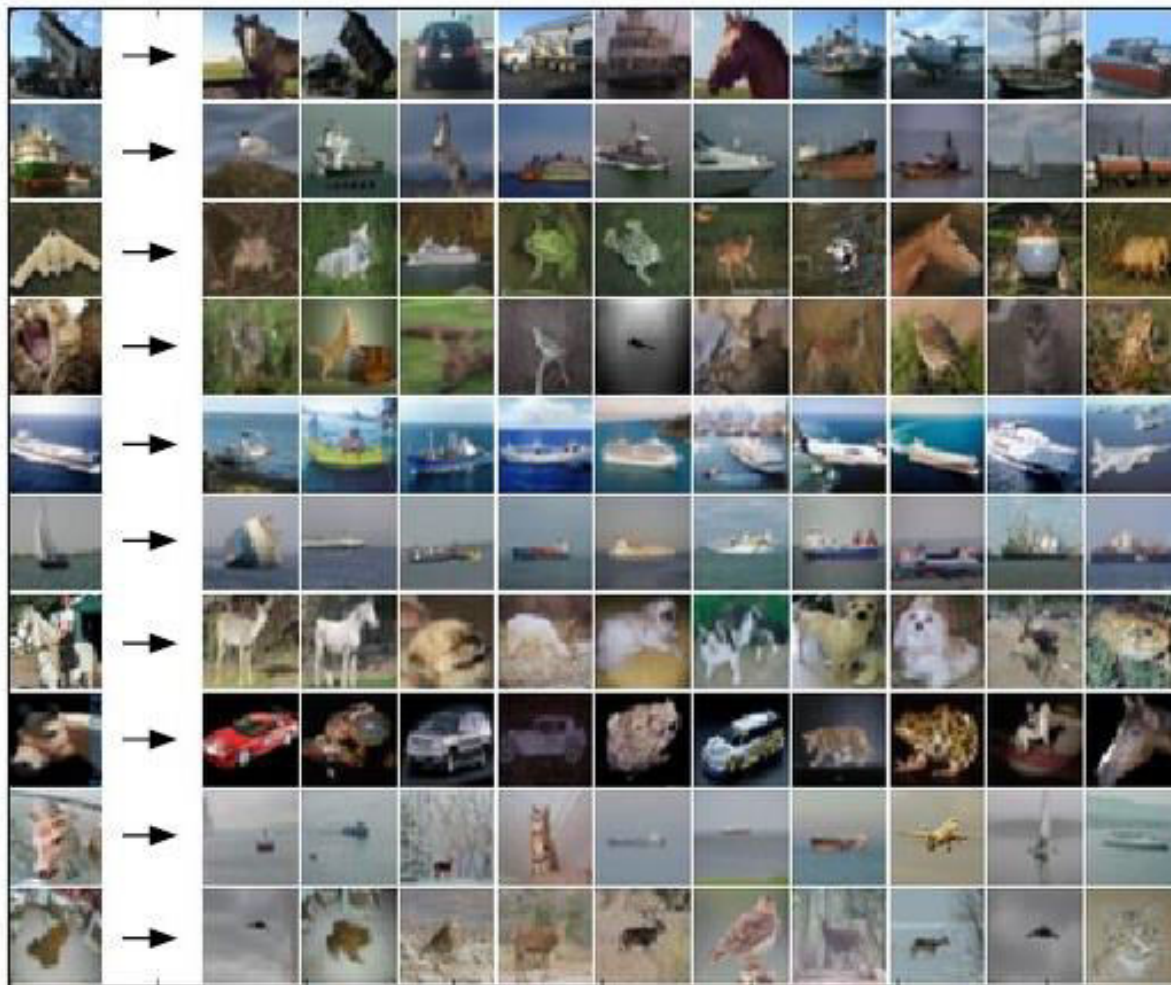
- **Input:** A training set of  $N$  images, each labeled with one of  $K$  different classes.
- **Learning:** Use training set to learn classifier (model) that predicts what class input images belong to.
- **Evaluation:** Evaluate quality of classifier by asking it to predict labels for a new set of images that it has never seen before.

# CIFAR-10 dataset



- 60,000 tiny images that are 32 pixels high and wide.
- Each image is labeled with one of 10 classes

# Nearest Neighbor Classification



The top 10 nearest neighbors in the training set according to “pixel-wise difference”.

# Pixel-wise difference

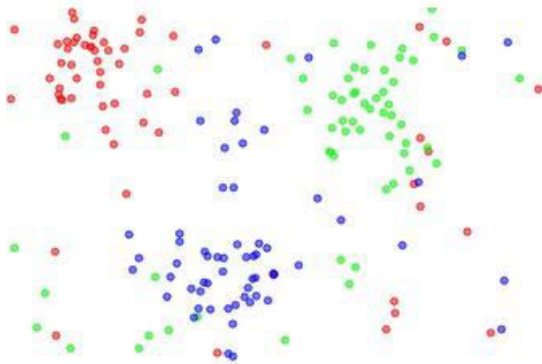
test image		training image		pixel-wise absolute value differences				
56	32	10	18	46	12	14	1	→ 456
90	23	128	133	82	13	39	33	
24	26	178	200	12	10	0	30	
2	0	255	220	2	32	22	108	

L1 norm:  $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

L2 norm:  $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$

# K-Nearest Neighbor Classifier

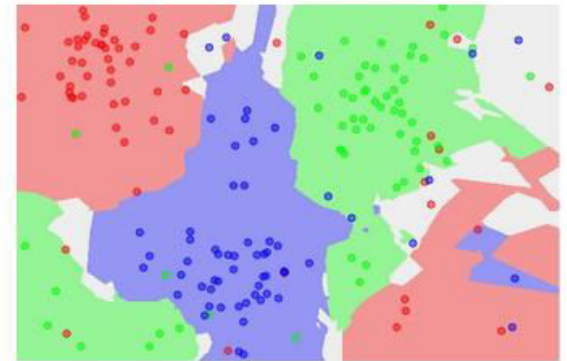
the data



NN classifier



5-NN classifier





# Disadvantages of k-NN

- The classifier must *remember* all of the training data and store it for future comparisons with the test data. This is space inefficient because datasets may easily be gigabytes in size.
- Classifying a test image is expensive since it requires a comparison to all training images.

# Linear Classification

# Toward neural networks

- Logistic regression model
  - A one-layer neural network
- Training a logistic regression model
  - Introduction to gradient descent
- These techniques generalize to deep networks

# Linear model

- Score function
  - Maps raw data to class scores
- Loss function
  - Measures how well predicted classes agree with ground truth labels
- Learning
  - Find parameters of score function that minimize loss function

# Linear score function

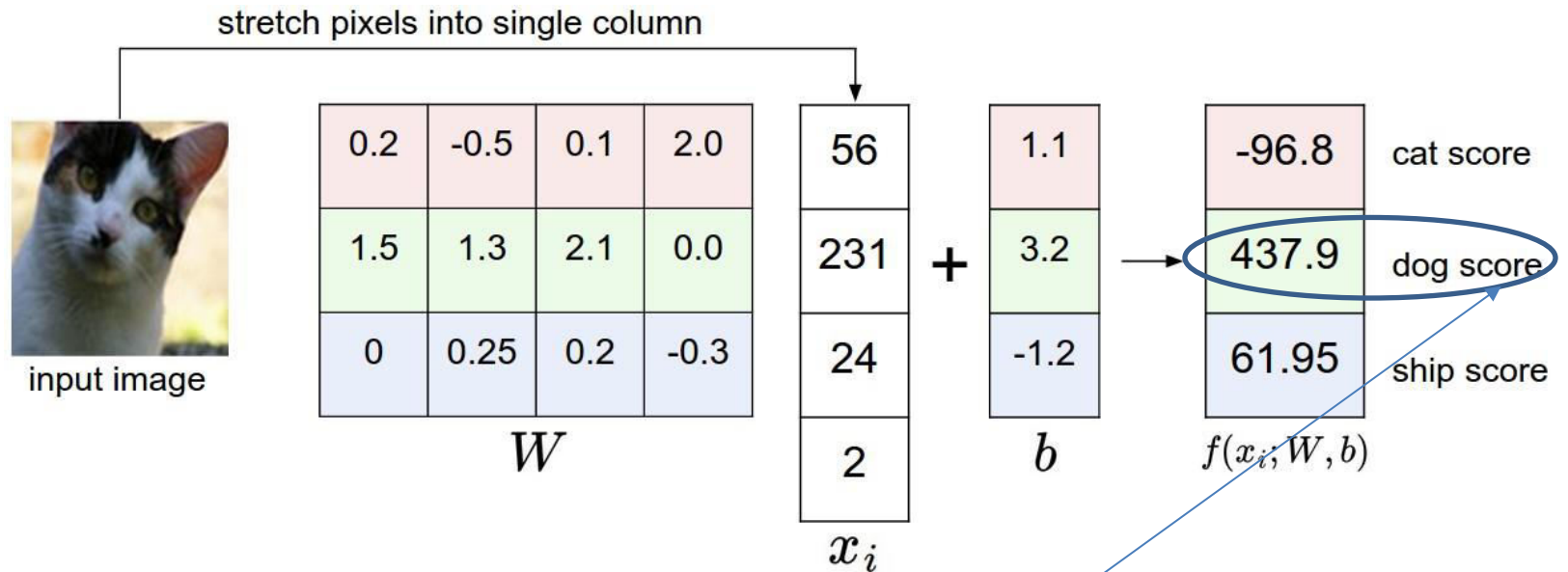
$$f(x_i, W, b) = Wx_i + b$$

- $x_i$  input image
- $W$  weights
- $b$  bias

Learning goal:

Learn weights and bias that minimize loss

# Using score function

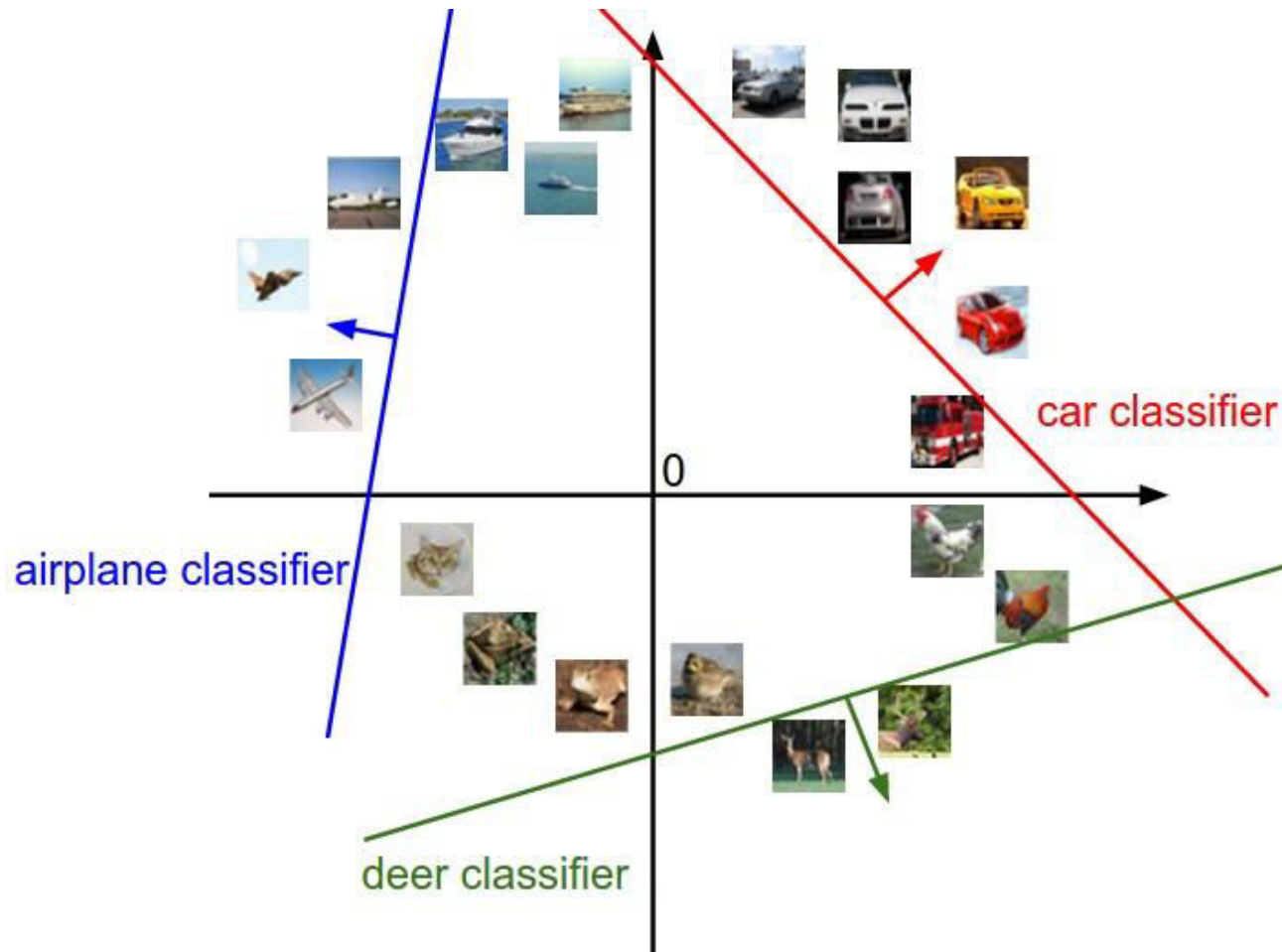


Predict class with highest score

# Addresses disadvantages of k-NN

- The classifier does not need to remember all of the training data and store it for future comparisons with the test data. It only needs the weights and bias.
- Classifying a test image is inexpensive since it just involves tensor multiplication. It does not require a comparison to all training images.

# Linear classifiers as hyperplanes

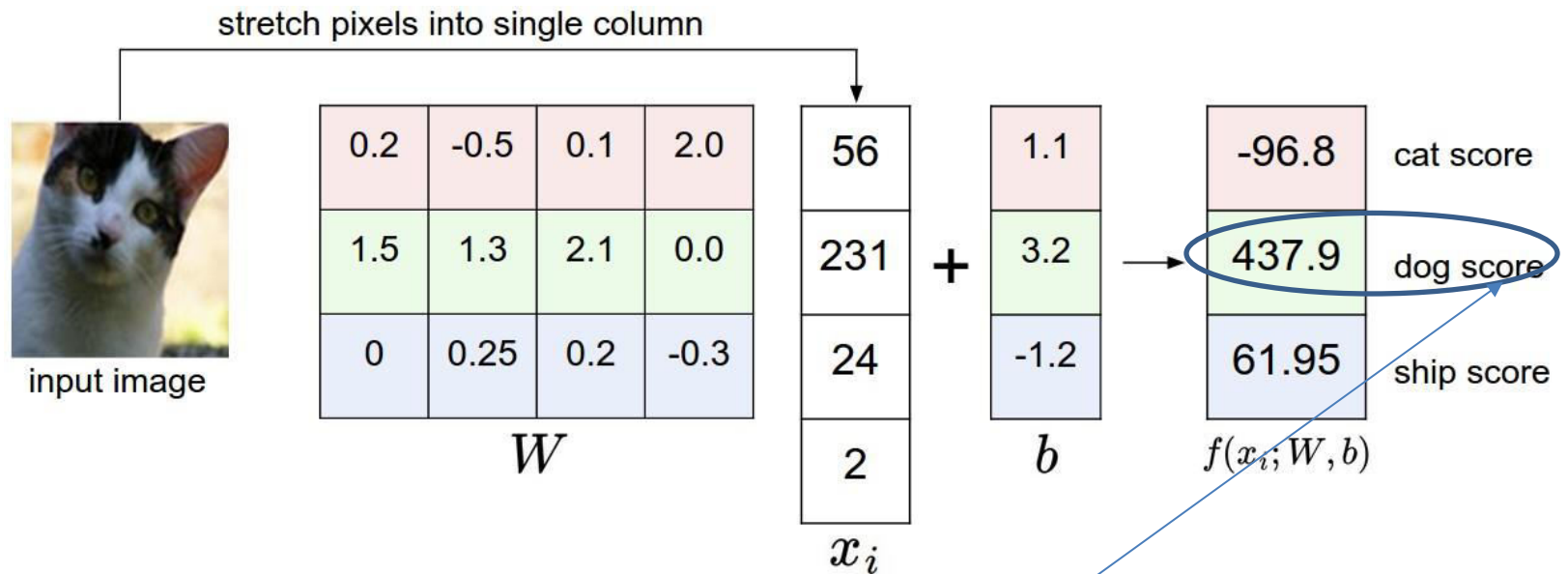




# Linear classifiers as template matching

- Each row of the weight matrix is a template for a class
- The score of each class for an image is obtained by comparing each template with the image using an *inner product* (or *dot product*) one by one to find the one that “fits” best.

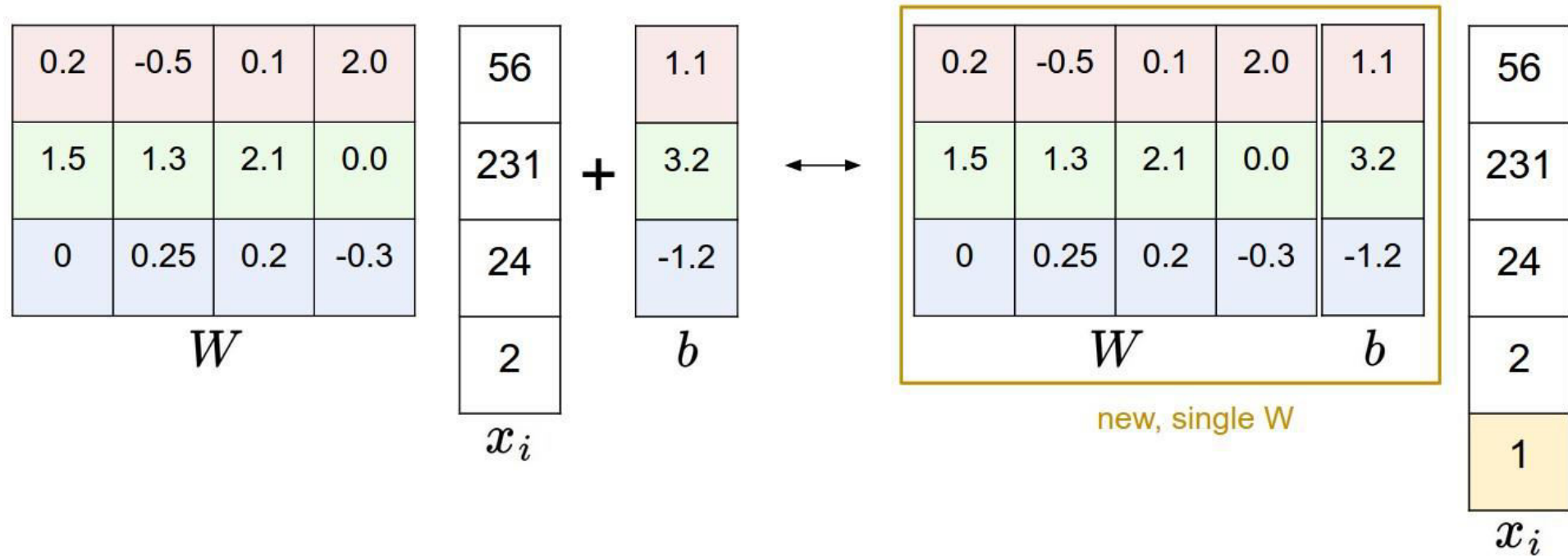
# Template matching example



Predict class with highest score  
(i.e., best template match)

# Bias trick

$$f(x_i, W) = Wx_i$$



# Linear model

- Score function
  - Maps raw data to class scores
- Loss function
  - Measures how well predicted classes agree with ground truth labels
- Learning
  - Find parameters of score function that minimize loss function

# Two loss functions

- Multiclass Support Vector Machine loss (SVM loss)
- Softmax classifier (multiclass logistic regression)
  - Cross-entropy loss function

# SVM loss idea

The SVM loss is set up so that the SVM “wants” the correct class for each image to have a score higher than the incorrect classes by some fixed margin  $\Delta$



# Scores

- Score vector

$$s = f(x_i, W)$$

- Score for j-th class

$$s_j = f(x_i, W)_j$$

# SVM loss for i-th training example

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$





# Example

$$s = [13, -7, 11] \quad \text{True class : } y_i = 0 \quad \Delta = 10$$

$$\begin{aligned} L_i &= \max(0, -7 - 13 + 10) + \max(0, 11 - 13 + 10) \\ &= 0 + 8 \end{aligned}$$



# An Issue

- Suppose  $\Delta = 10$
- If the difference in scores between a correct class and a nearest incorrect class is at least 15 for all examples, then multiplying all elements of  $W$  by 2 would make the new difference 30.
- $\lambda W$  where  $\lambda > 1$  also gives zero loss if  $W$  gives zero loss

# Regularization

- Add a regularization penalty to the loss function

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

# Multiclass SVM loss

$$L = \underbrace{\frac{1}{N} \sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Final classifier encouraged to take into account all input dimensions to small amounts rather than a few input dimensions very strongly

# Multiclass SVM loss

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} [\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta)] + \lambda \sum_k \sum_l W_{k,l}^2$$

# Example

$$x = [1, 1, 1, 1] \quad w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$L2 \text{ penalty of } w_1 = 1.0$$

$$L2 \text{ penalty of } w_2 = 0.25$$

Final classifier encouraged to take into account all input dimensions to small amounts rather than a few input dimensions very strongly

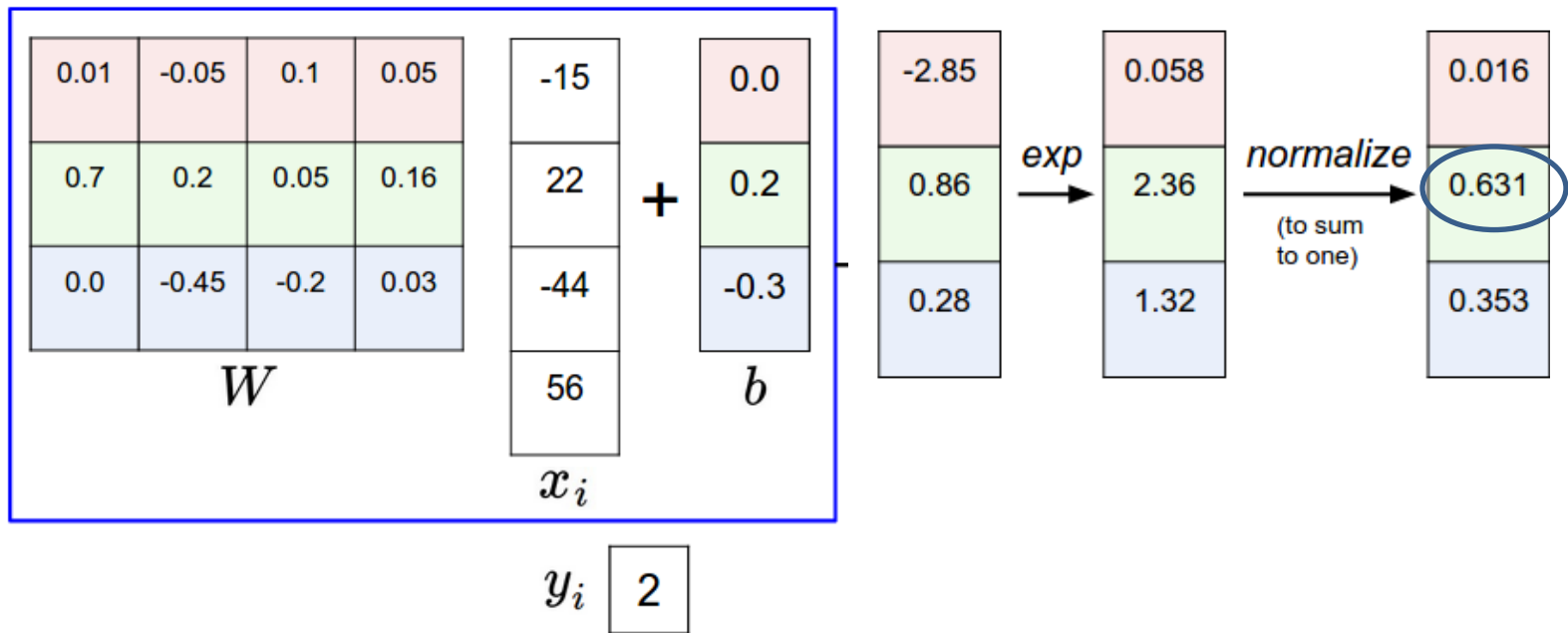
# Two loss functions

- Multiclass Support Vector Machine loss
- Softmax classifier (multiclass logistic regression)
  - Cross-entropy loss function

# Softmax classifier (multiclass logistic regression)

$$P(y_i | x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

matrix multiply + bias offset



Pick class with highest probability



# Logistic function

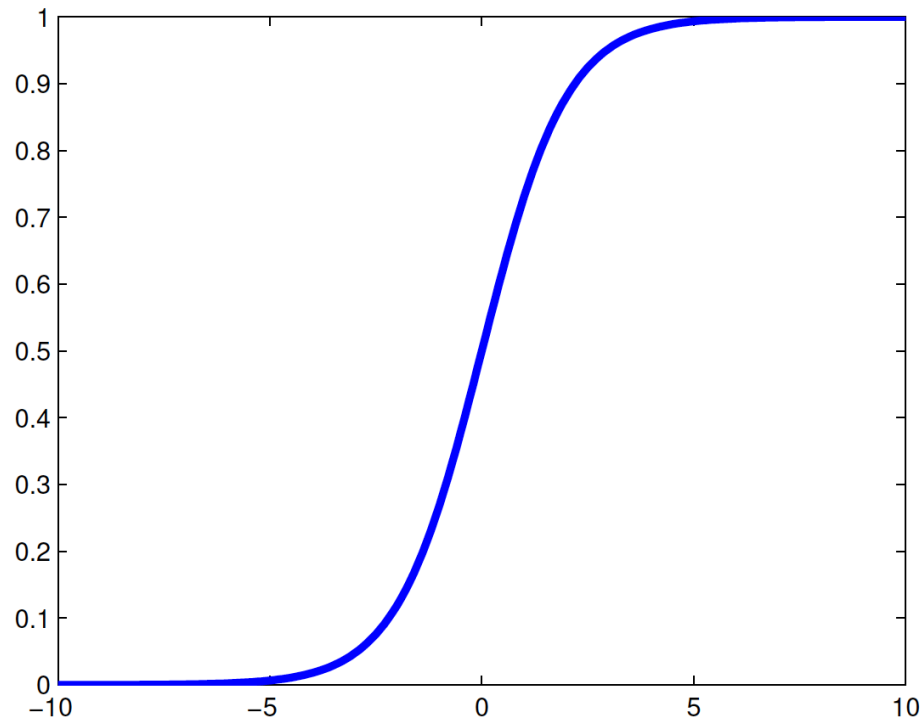


Figure 1.19(a) from Murphy

# Logistic regression example

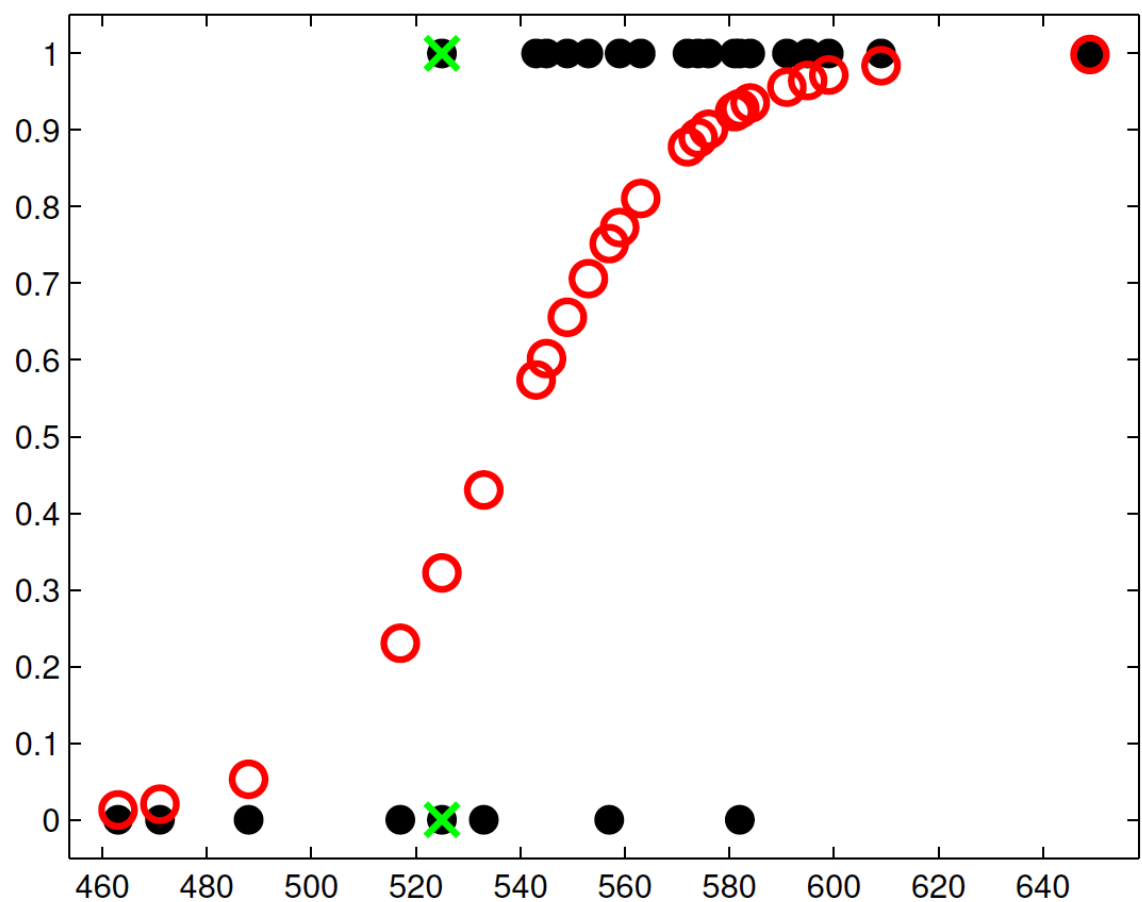
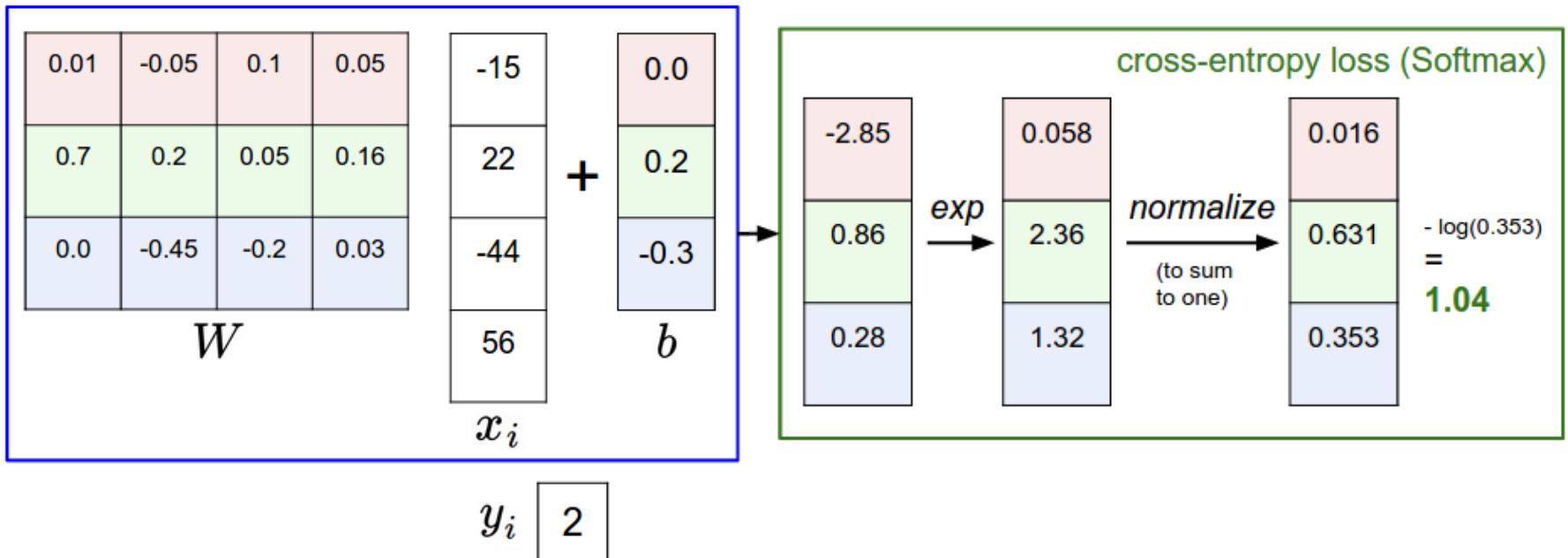


Figure 1.19(b) from Murphy

# Cross-entropy loss

$$L_i = -\log \left( \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

matrix multiply + bias offset



Full loss for the dataset is the mean of  $L_i$  over all training examples plus a regularization term

# Interpreting cross-entropy loss

The cross-entropy objective *wants* the predicted distribution to have all of its mass on the correct answer.

# Information theory motivation for cross-entropy loss

*Cross-entropy* between a true distribution  $p$  and an estimated distribution  $q$

$$H(p, q) = - \sum_x p(x) \log q(x)$$

# Information theory motivation for cross-entropy loss

The Softmax classifier is minimizing the cross-entropy between the estimated class probabilities

$$( q = e^{f_{y_i}} / \sum_j e^{f_j} ) \text{ and}$$

the “true” distribution, which in this interpretation is the distribution where all probability mass is on the correct class

(  $p = [0, \dots, 1, \dots, 0]$  contains a single 1 in the  $y_i$  position)

# Quiz

$$H(p, p) = ?$$

# Quiz

$$H(p, p) = 0$$



# Learning task

- Find parameters of the model that minimize loss
- Looking ahead: Stochastic gradient descent

# Looking ahead: linear algebra

- Images represented as tensors (3D arrays)
- Operations on these tensors used to train models
- Review basics of linear algebra
  - Chapter 2 of Deep Learning textbook
  - Will review briefly in class

# Looking ahead: multivariate calculus

- Optimization of functions over tensors used to train models
- Involves basics of multivariate calculus
  - Gradients, Hessian
- Will review briefly in class

# Acknowledgment

- Based on material from Stanford CS231n  
<http://cs231n.github.io/>