Machine Learning Basics

Anupam Datta
CMU

Spring 2019

Image Classification

Image Classification

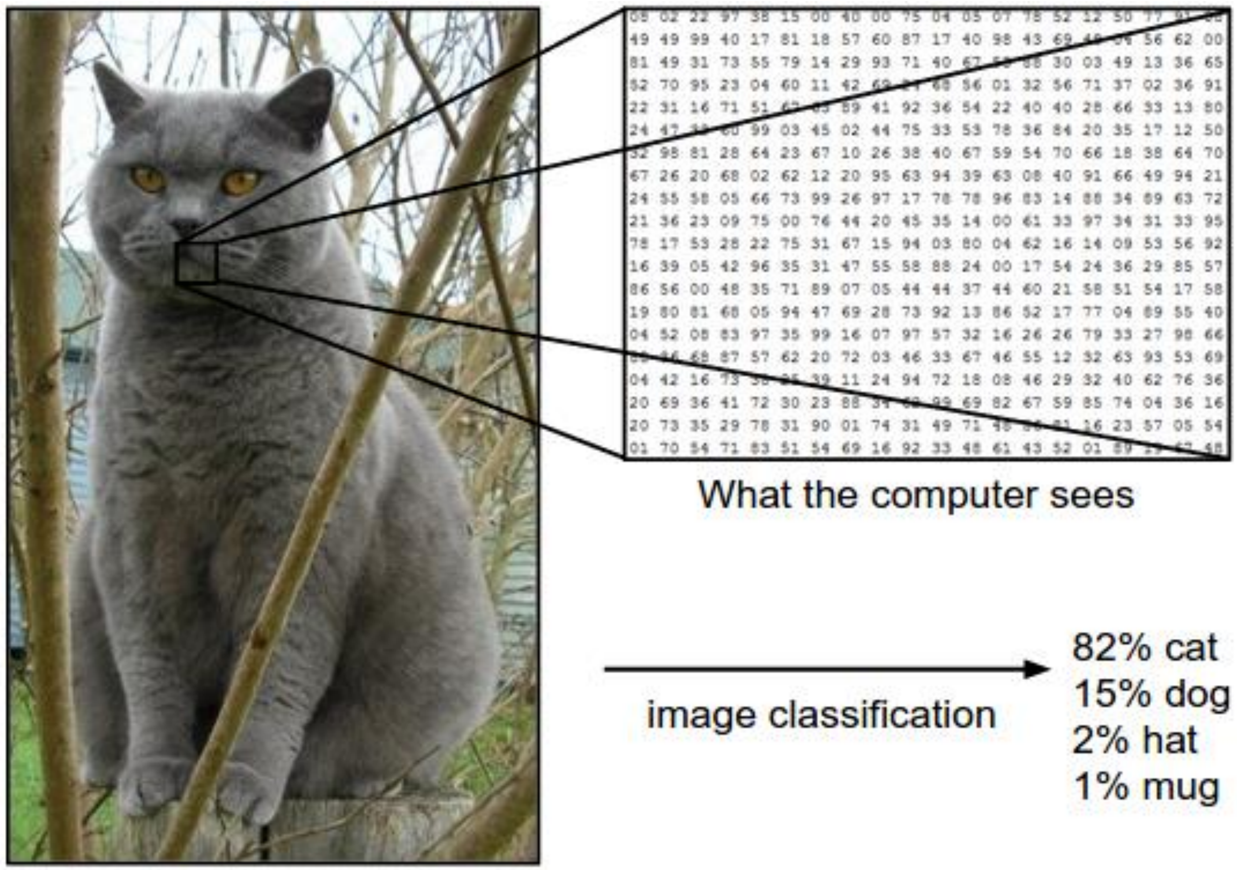
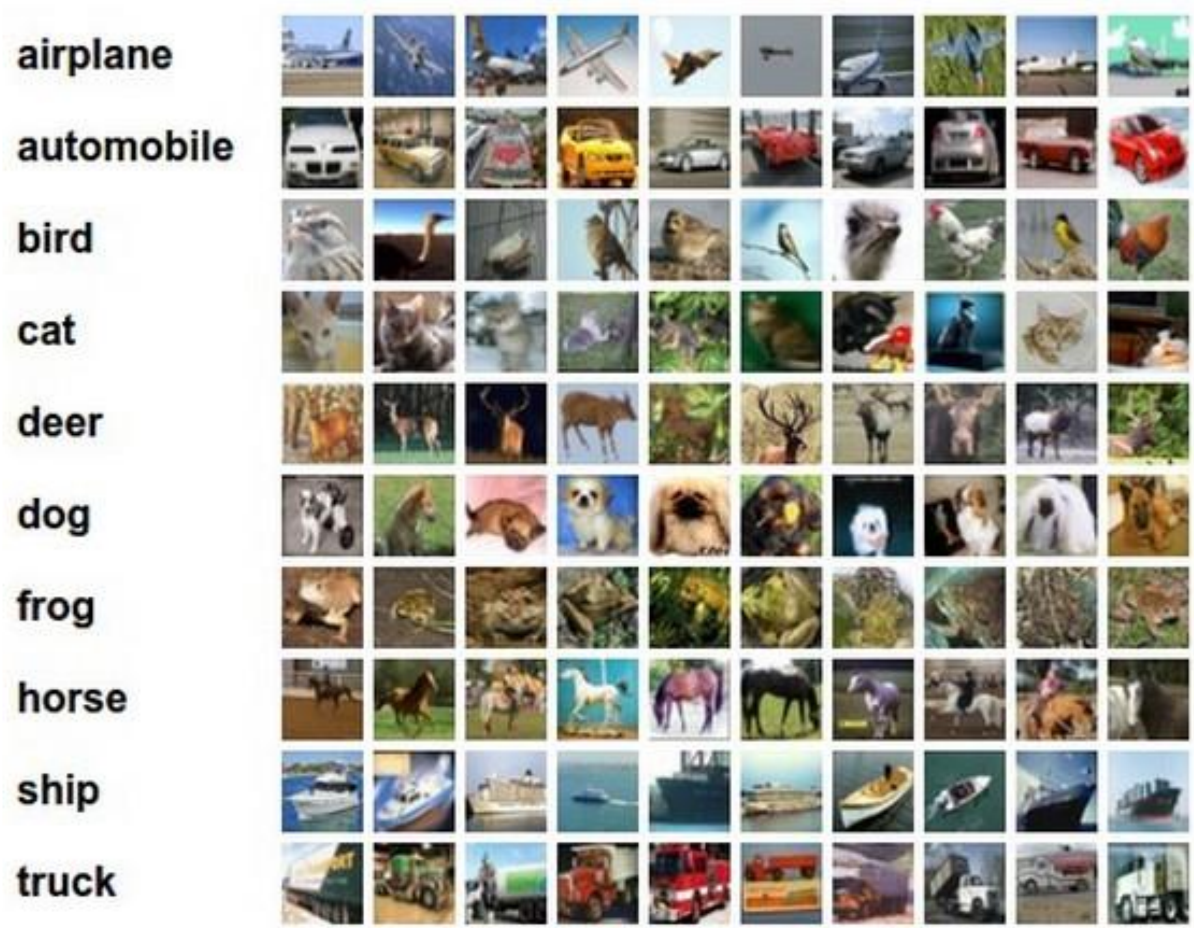


Image classification pipeline

- **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

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

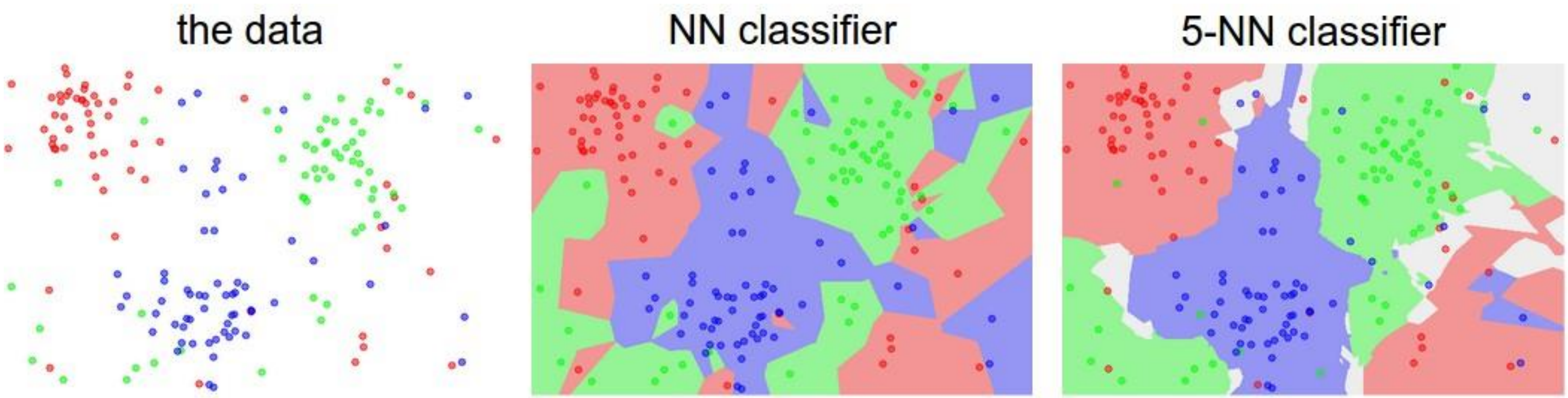
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

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

L2 norm: $d_2(I_1, I_2) = \sqrt[2]{\Sigma_p (I_1^p - I_2^p)^2}.$

K-Nearest Neighbor 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

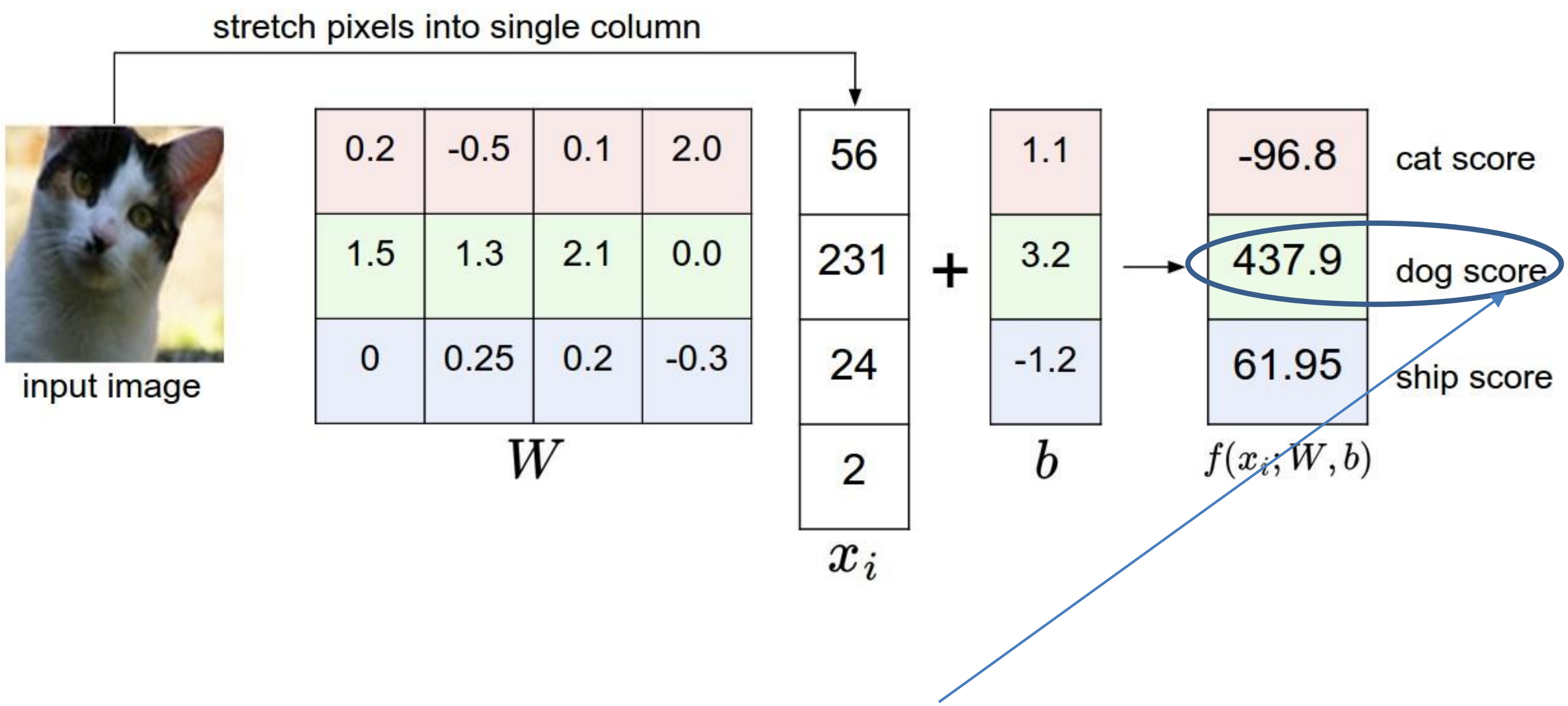
Linear score function

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

- x_i input image
- W weights
- b bias

Learning goal:
Learn weights and bias that minimize loss

Using score function



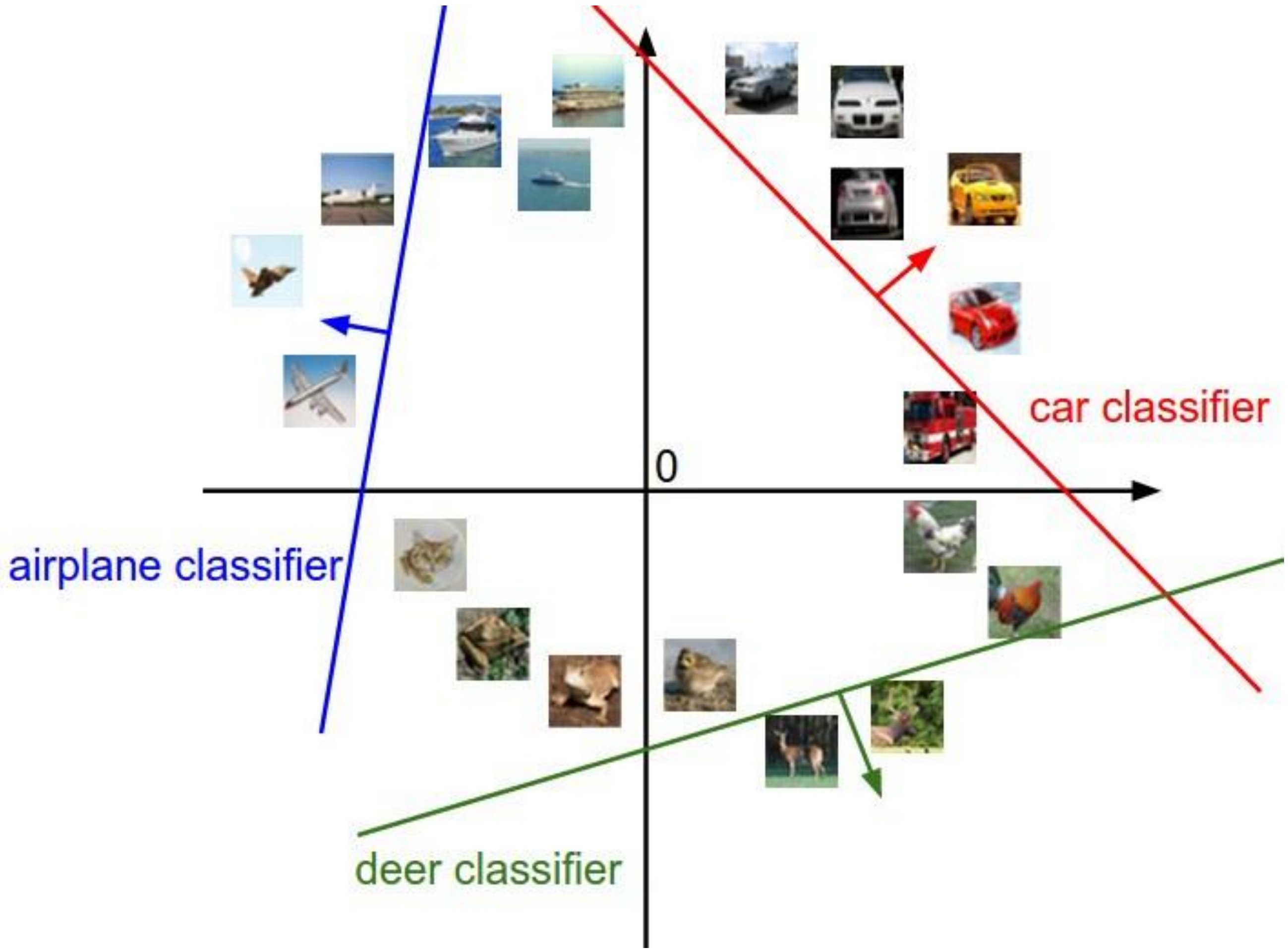
Predict class with highest score

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

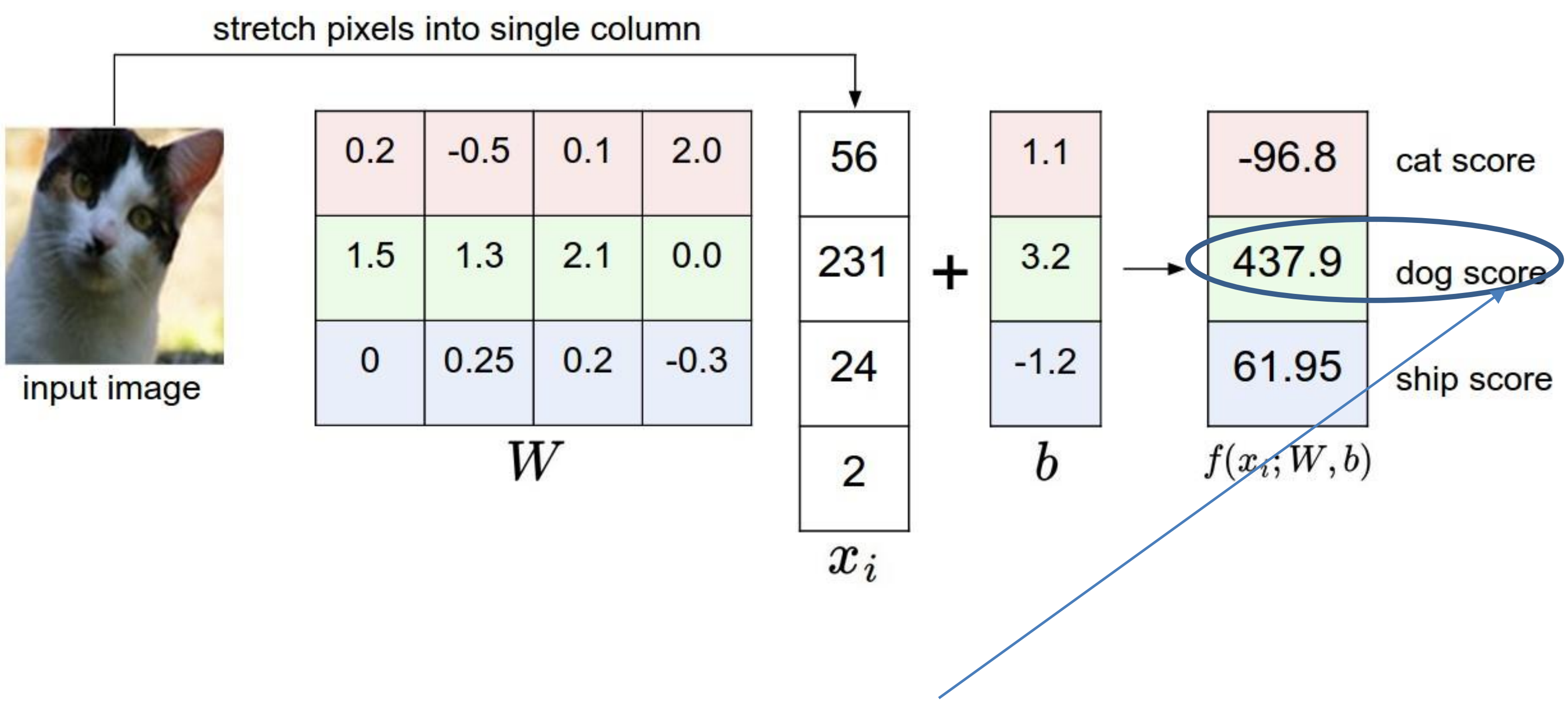


Linear classifiers as template matching

- 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.

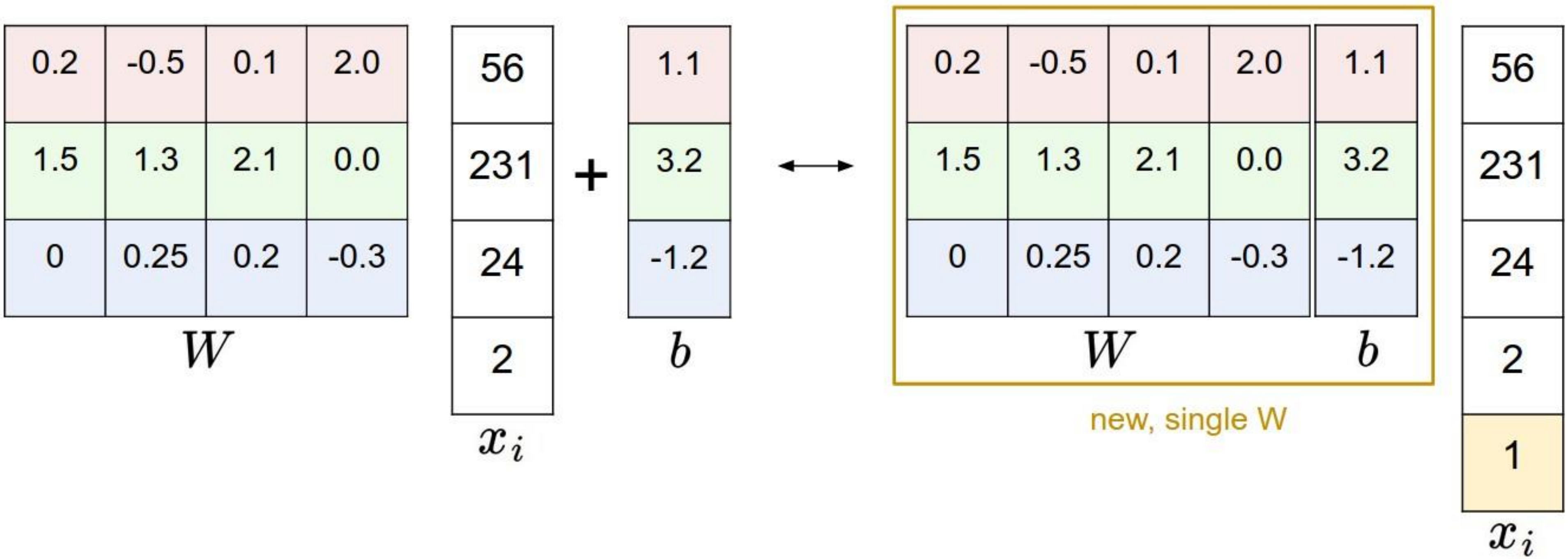
Template matching example



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

Bias trick

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



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 Δ

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

$$s = [13, -7, 11] \qquad True\ class: y_i = 0 \qquad \Delta = 10$$

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

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

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

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

$$L2\ penalty\ of\, w_1 = 1.0$$

$$L2\ penalty\ of\, w_2 = 0.25$$

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

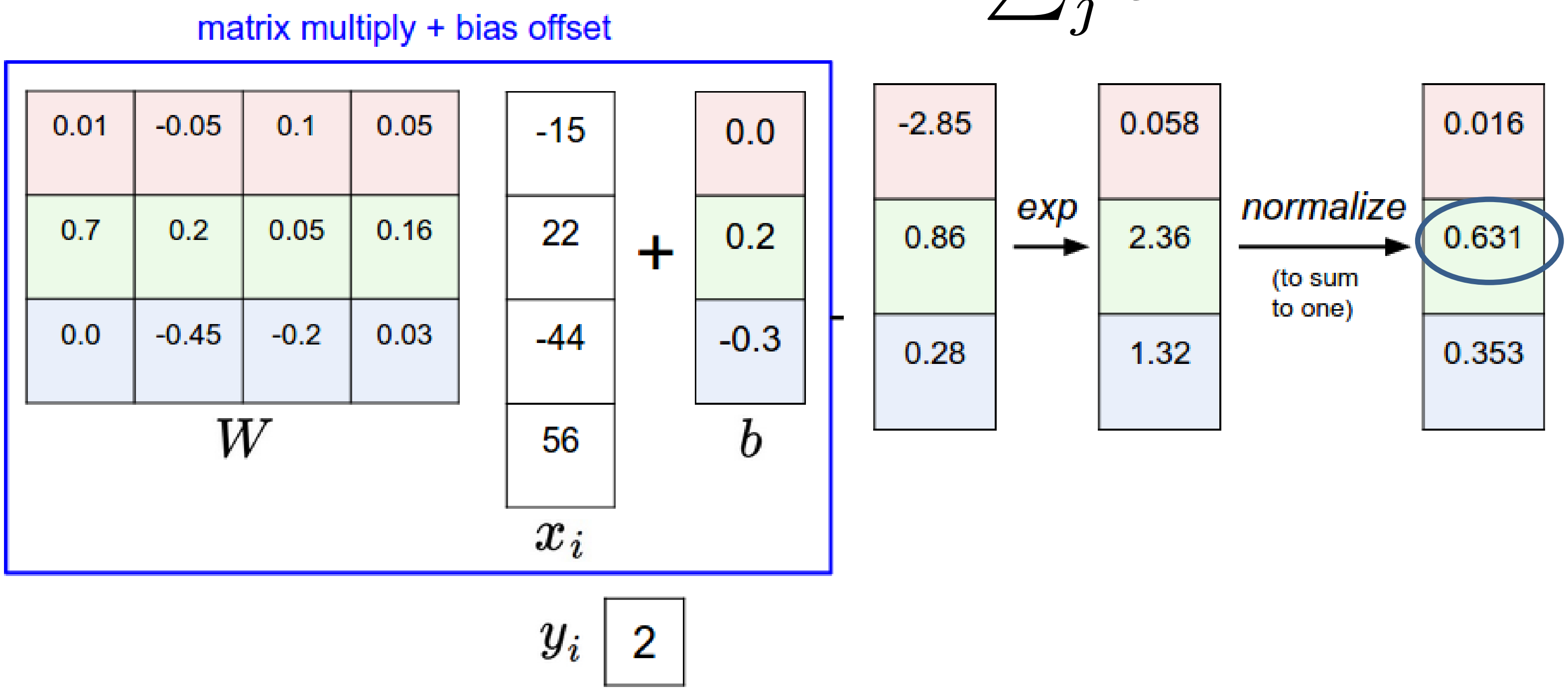
Two loss functions

- Multiclass Support Vector Machine loss

- Softmax classifier (multiclass logistic regression)
  - Cross-entropy loss function

Softmax classifier (multiclass logistic regression)

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



Pick class with highest probability

Logistic function

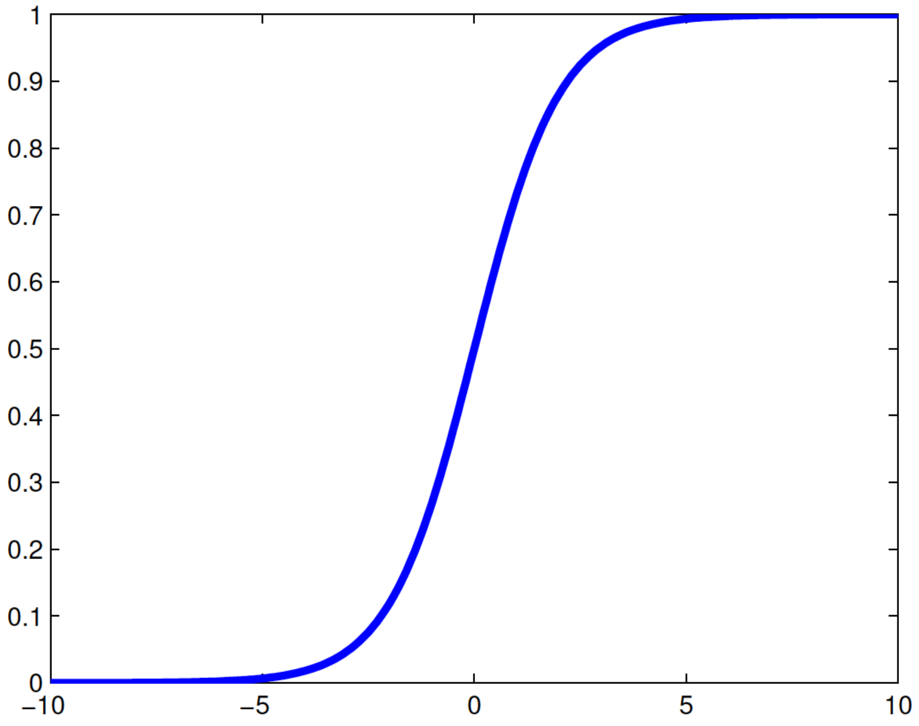


Figure 1.19(a) from Murphy

Logistic regression example

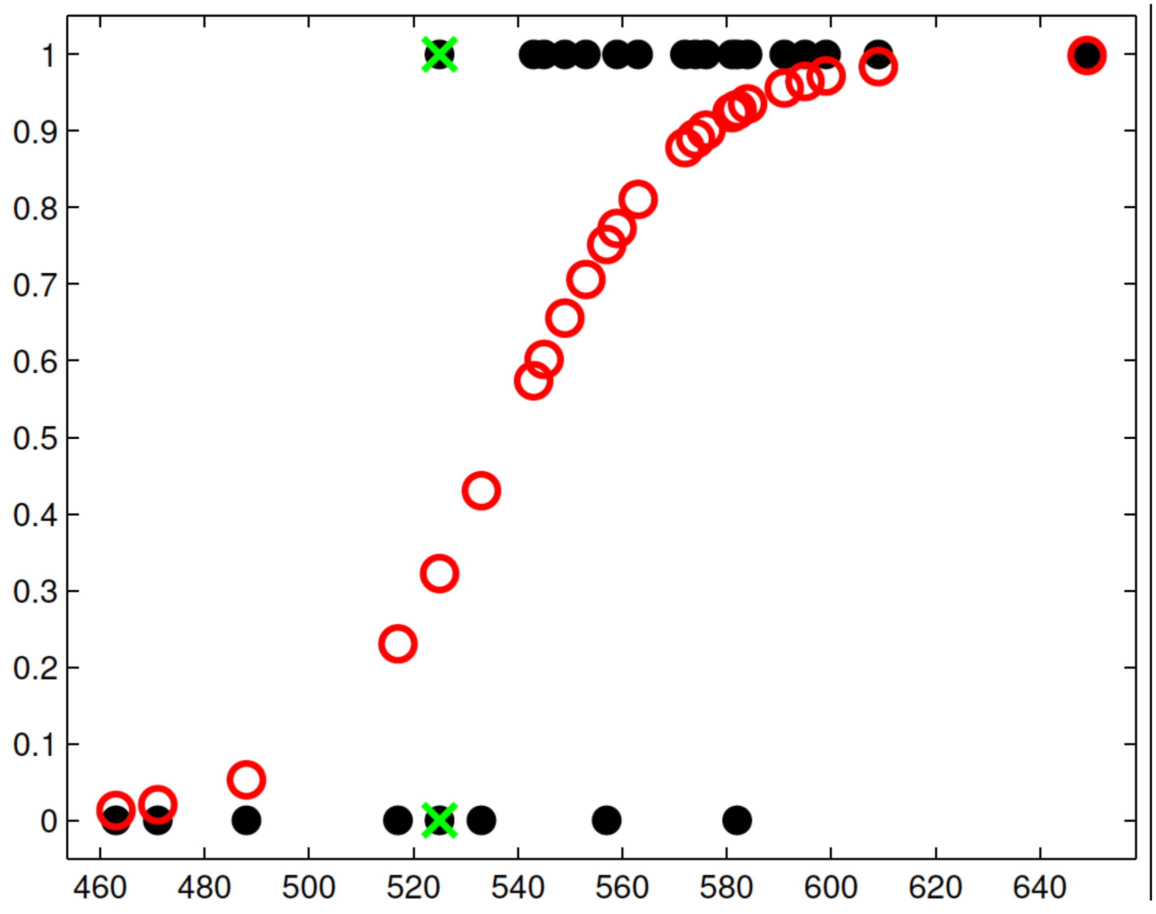
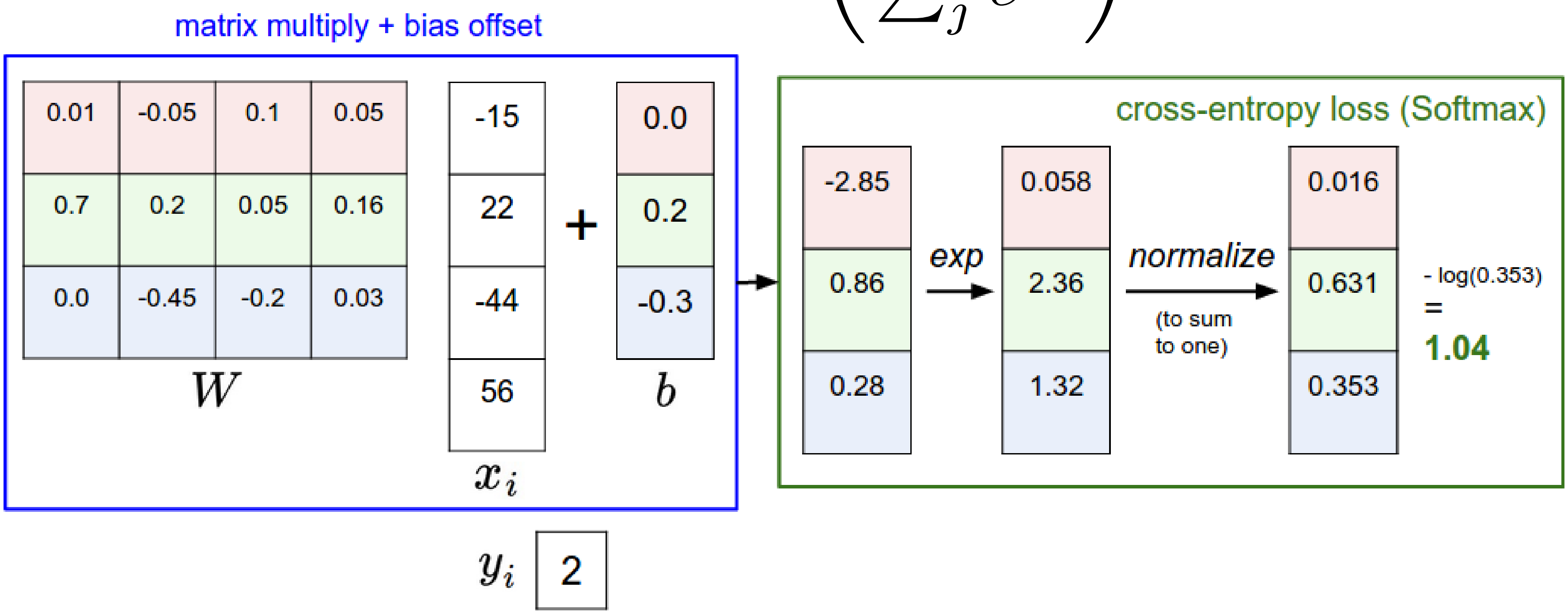


Figure 1.19(b) from Murphy

Cross-entropy loss

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



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

the “true” distribution, which in this interpretation is the distribution where all probability mass is on the correct class
($p = [0, \ldots 1, \ldots, 0]$ 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/