

Lecture 3:

Linear Classification

Last time: Image Classification

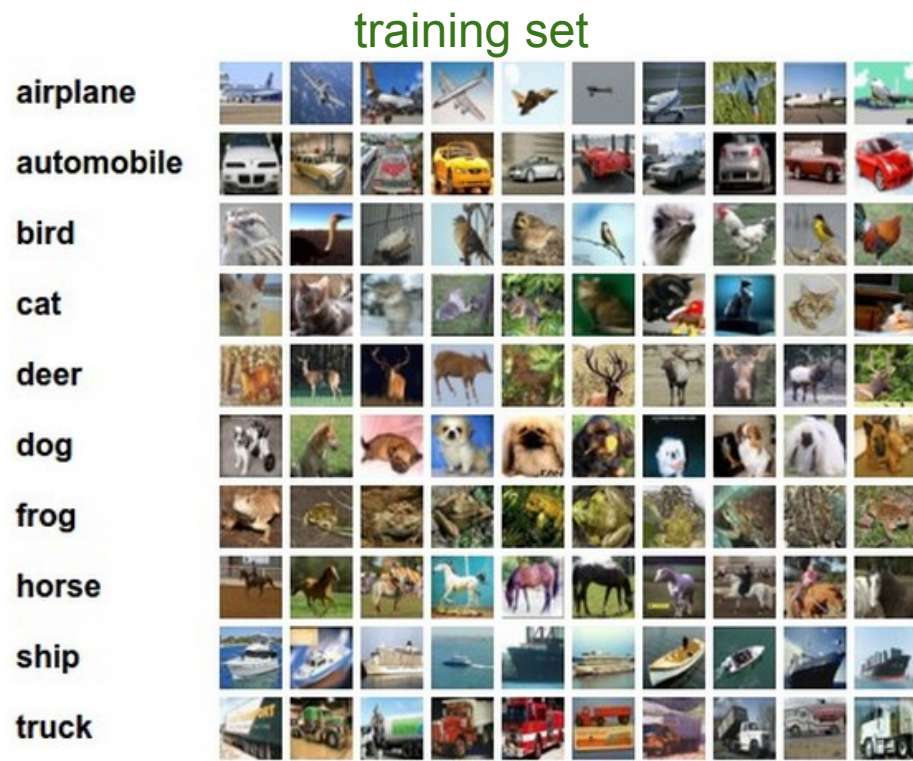


assume given set of discrete labels
{dog, cat, truck, plane, ...}

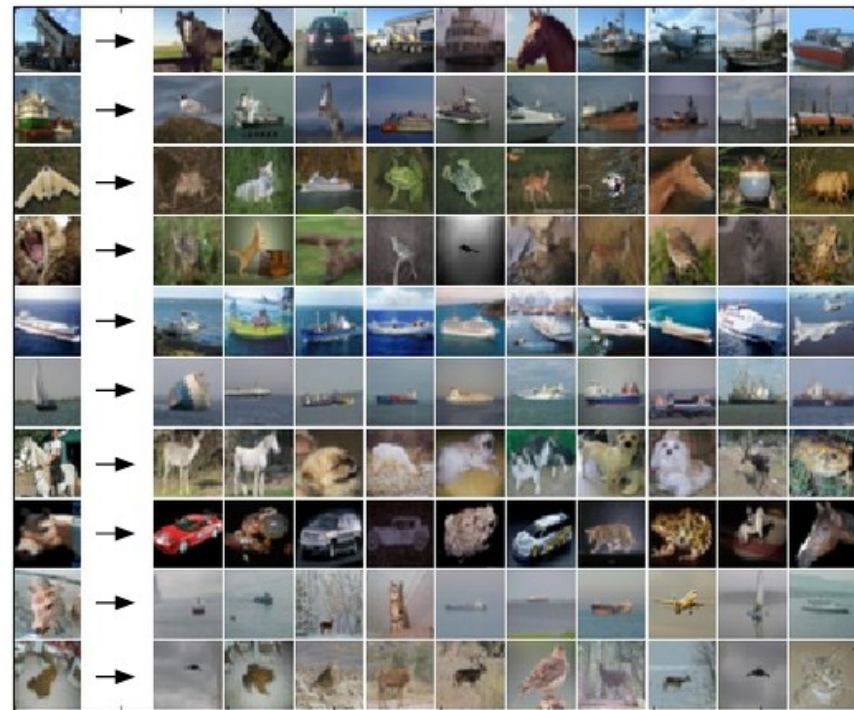


cat

k-Nearest Neighbor



test images



Linear Classification

1. define a **score function**



→ class scores

Linear Classification

1. define a **score function**

The diagram shows the equation $f(x_i, W, b) = Wx_i + b$ with several annotations. An arrow points from the text "data (image)" to the input x_i . An arrow points from the text "class scores" to the function f . An arrow points from the text "weights" to the weight matrix W . An arrow points from the text "bias vector" to the bias vector b . The text "parameters" is located below the weight matrix W and the bias vector b , indicating they are the learnable parameters of the model.

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

data (image)

class scores

"weights"

"bias vector"

"parameters"

Linear Classification

1. define a **score function**

(assume CIFAR-10 example so
32 x 32 x 3 images, 10 classes)

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

data (image)
[3072 x 1]

class scores

weights

bias vector

Linear Classification

1. define a **score function**

(assume CIFAR-10 example so
32 x 32 x 3 images, 10 classes)

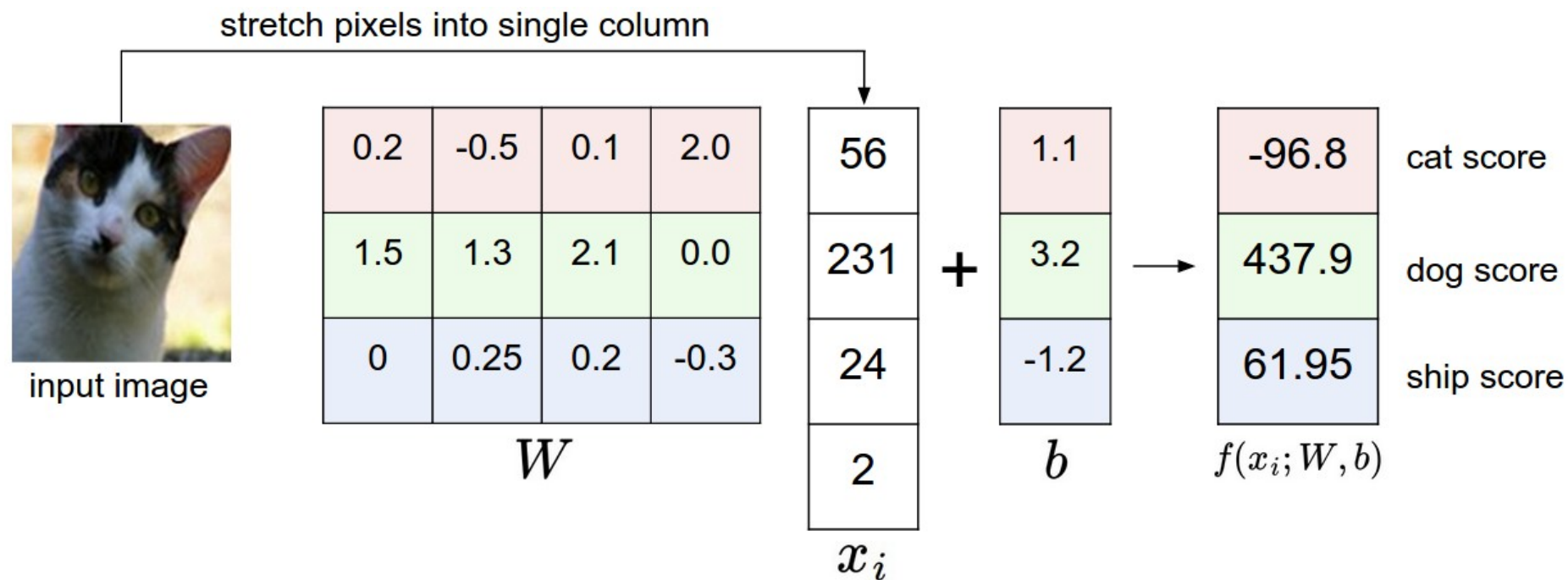
$$f(x_i, W, b) = Wx_i + b$$

The diagram illustrates the components of the linear classification score function $f(x_i, W, b) = Wx_i + b$. Arrows point from the following labels to their respective parts in the equation:

- data (image)** [3072 x 1] points to x_i .
- weights** [10 x 3072] points to W .
- bias vector** [10 x 1] points to b .
- class scores** [10 x 1] points to the entire function $f(x_i, W, b)$.

Linear Classification

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

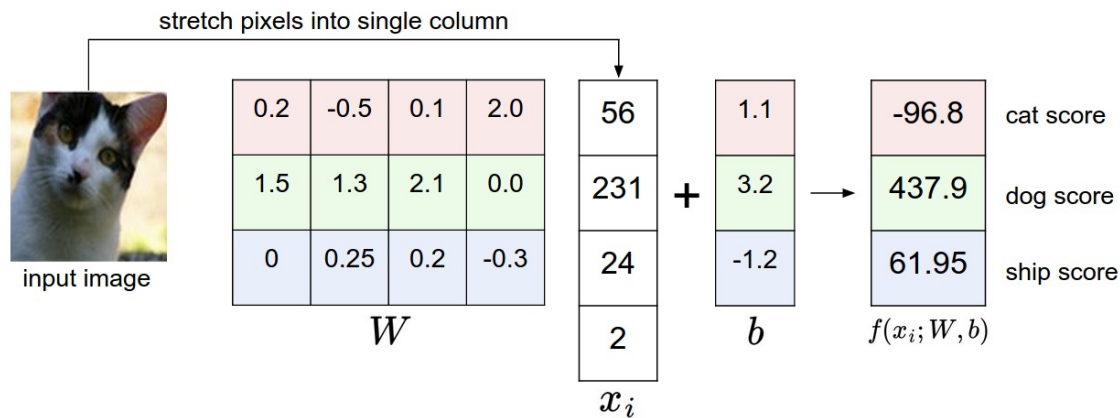


Interpreting a Linear Classifier

Question:

what can a linear classifier do?

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



airplane

automobile

bird

cat

deer

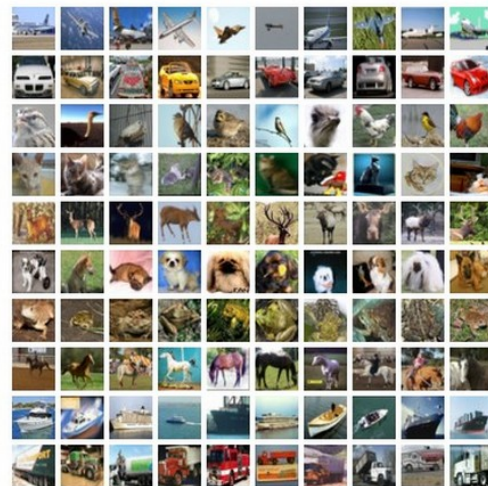
dog

frog

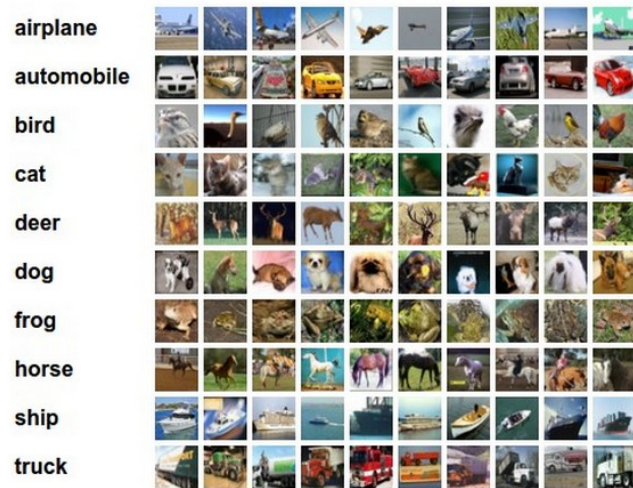
horse

ship

truck



Interpreting a Linear Classifier

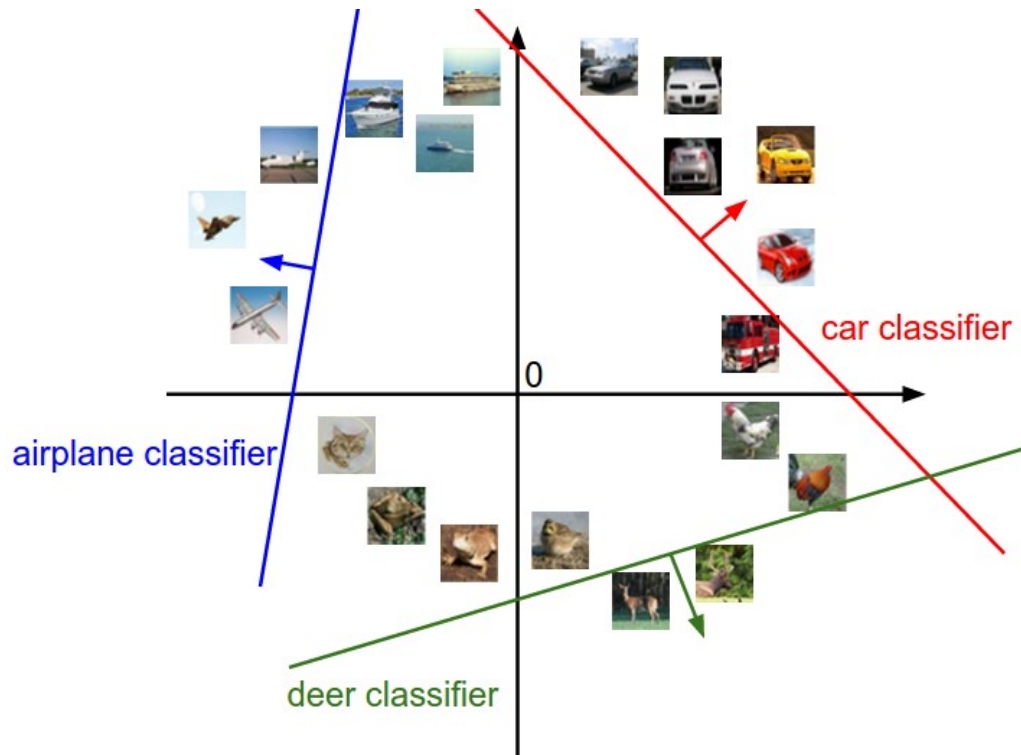


$$f(x_i, W, b) = Wx_i + b$$

Example training
classifiers on CIFAR-10:



Interpreting a Linear Classifier



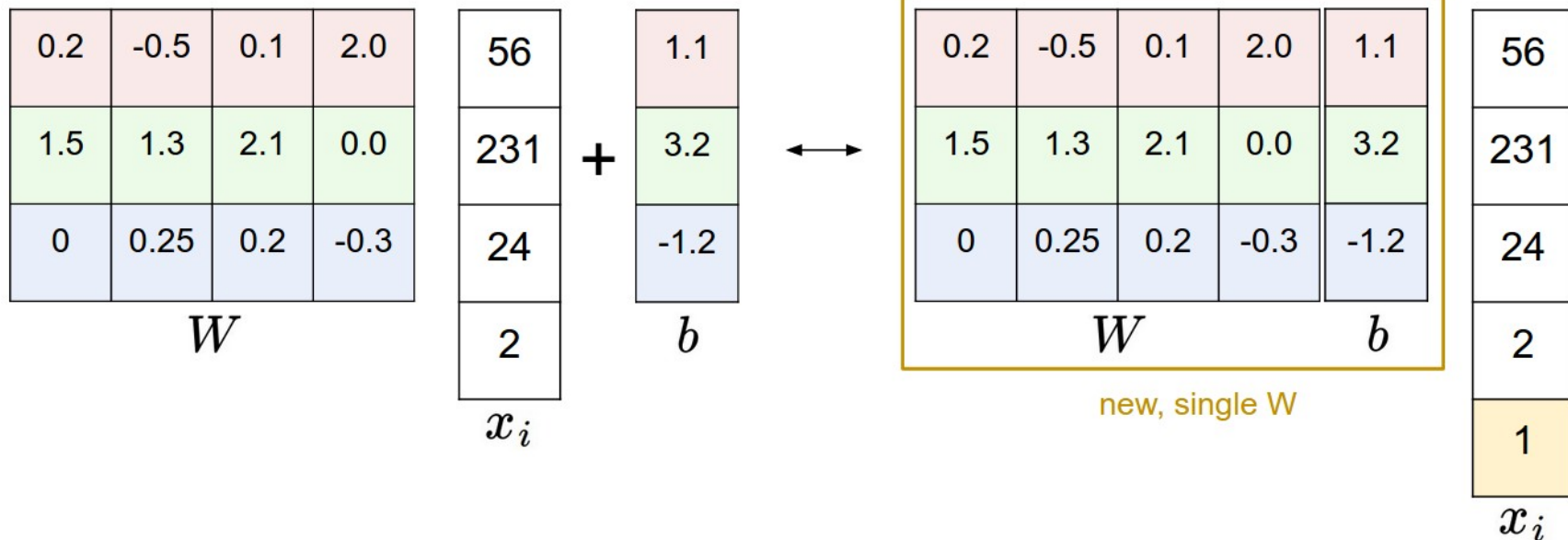
$$f(x_i, W, b) = Wx_i + b$$

Bias trick

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

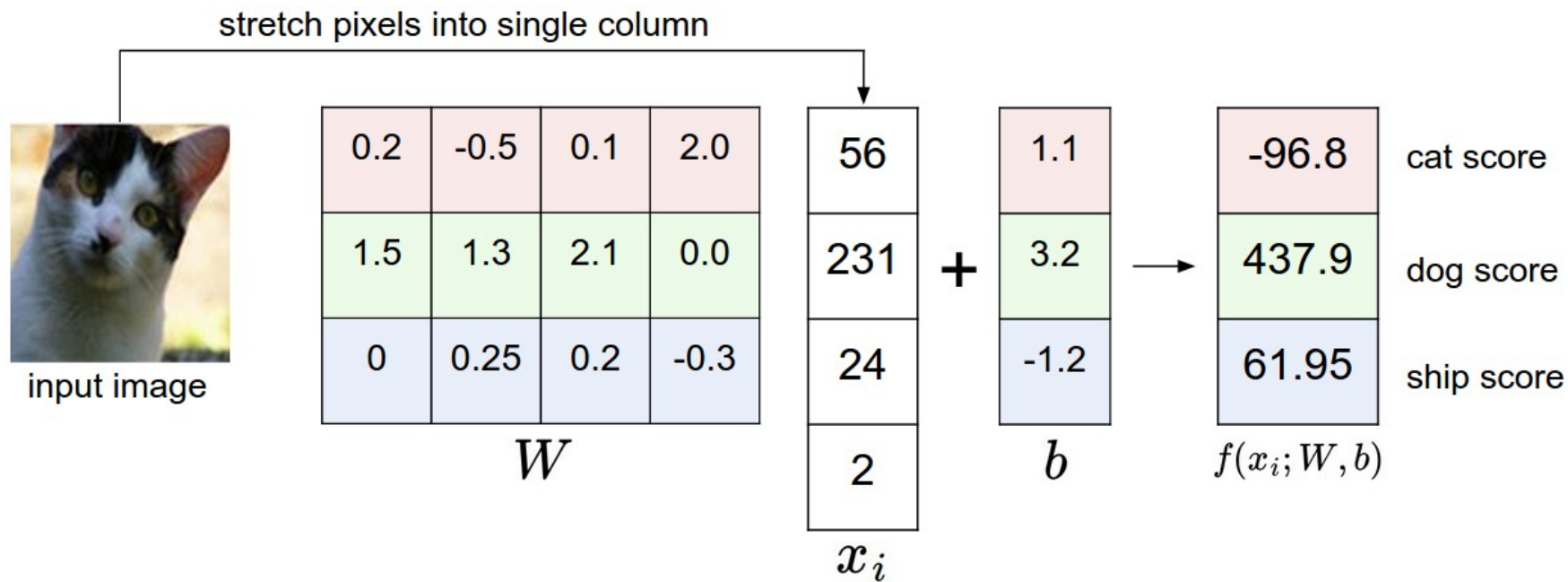


$$f(x_i, W) = Wx_i$$



So far:

We defined a (linear) **score function**: $f(x_i, W, b) = Wx_i + b$



2. Define a **loss function** (or cost function, or objective)

2. Define a **loss function** (or cost function, or objective)

- scores, label $\longrightarrow$ *loss*.
 $f(x_i, W)$ y_i L_i

2. Define a **loss function** (or cost function, or objective)

- scores, label $\longrightarrow$ loss.
 $f(x_i, W)$ y_i L_i

Example:

$$f(x_i, W) = [13, -7, 11]$$

$$y_i = 0 \longrightarrow$$


2. Define a **loss function** (or cost function, or objective)

$$\begin{array}{ccc} \text{scores, label} & \longrightarrow & \text{loss.} \\ f(x_i, W) & y_i & L_i \end{array}$$

Example:

$$f(x_i, W) = [13, -7, 11]$$
$$y_i = 0 \quad \longrightarrow \quad \uparrow$$

Question: if you were to assign a single number to how “unhappy” you are with these scores, what would you do?

2. Define a **loss function** (or cost function, or objective)
One (of many ways) to do it: **Multiclass SVM Loss**

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

2. Define a **loss function** (or cost function, or objective)
One (of many ways) to do it: **Multiclass SVM Loss**

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

(One possible generalization of Binary Support Vector Machine to multiple classes)

$$L_i = C \max(0, 1 - y_i w^T x_i) + R(W)$$

2. Define a **loss function** (or cost function, or objective)
One (of many ways) to do it: **Multiclass SVM Loss**

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

loss due to
example i

sum over all
incorrect labels

difference between the correct class
score and incorrect class score

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

loss due to
example i

sum over all
incorrect labels

difference between the correct class
score and incorrect class score



Example: $L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$



$$f(x_i, W) = [13, -7, 11]$$

e.g. 10

$$y_i = 0$$

loss = ?

Example: $L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$

$$f(x_i, W) = [13, -7, 11]$$

$$y_i = 0$$

e.g. 10

$$L_i = \max(0, -7 - 13 + 10) + \max(0, 11 - 13 + 10)$$

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

```
def L_i_vectorized(x, y, W):  
    scores = W.dot(x)  
    margins = np.maximum(0, scores - scores[y] + 1)  
    margins[y] = 0  
    loss_i = np.sum(margins)  
    return loss_i
```

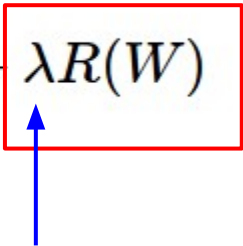
There is a bug with the objective...



$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right]$$

L2 Regularization

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

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$


Regularization strength

L2 regularization: motivation

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

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

$$w_1^T x = w_2^T x = 1$$

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$



$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

Do we have to cross-validate both
 Δ and λ ?

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

$$\begin{aligned} & \max(0, \delta f + \Delta) \\ \rightarrow & \max(0, \delta f + \Delta_2) && \text{if } \Delta \rightarrow \Delta_2 \\ \rightarrow & \max(0, \frac{\Delta_2}{\Delta} \delta f + \Delta_2) && \text{then scale the weights by } \frac{\Delta_2}{\Delta} \\ = & \max(0, \frac{\Delta_2}{\Delta} (\delta f + \Delta)) \\ = & \frac{\Delta_2}{\Delta} \max(0, \delta f + \Delta) \end{aligned}$$

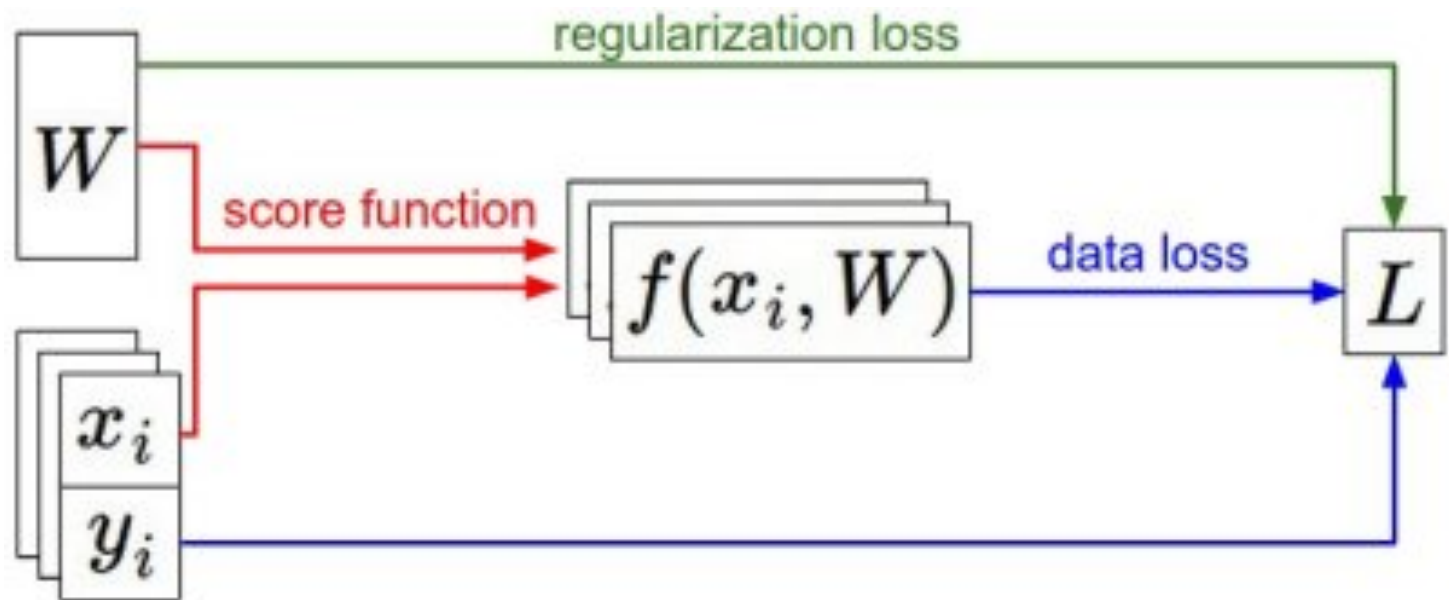
So far...

1. Score function

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

2. Loss function

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$



Softmax Classifier

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

score function
is the same

(extension of Logistic Regression to multiple classes)

Softmax Classifier

$f(x_i, W) = Wx_i$ score function
is the same

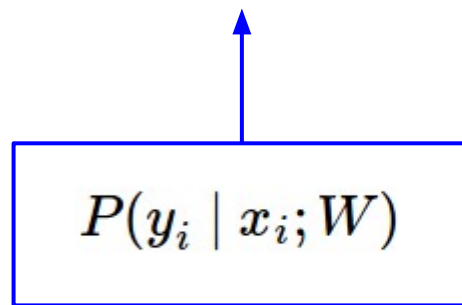
$$L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

Softmax Classifier

$f(x_i, W) = Wx_i$ score function
is the same

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

softmax function


$$P(y_i | x_i; W)$$

i.e. we're minimizing
the negative log
likelihood.

Softmax Classifier

$f(x_i, W) = Wx_i$ score function
is the same

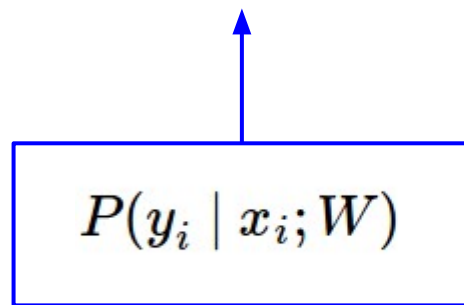
$$L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

softmax function

Softmax Classifier

$f(x_i, W) = Wx_i$ score function
is the same

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



A blue box containing the expression $P(y_i | x_i; W)$. A blue arrow points upwards from the top center of the box to the denominator of the fraction in the equation $L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$ above it.

$$P(y_i | x_i; W)$$

i.e. we're minimizing
the negative log
likelihood.

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

matrix multiply + bias offset

0.01	-0.05	0.1	0.05
0.7	0.2	0.05	0.16
0.0	-0.45	-0.2	0.03

W

-15
22
-44
56

x_i

y_i

+

0.0
0.2
-0.3

b

2

hinge loss (SVM)

-2.85
0.86
0.28

$$\begin{aligned} &\max(0, -2.85 - 0.28 + 1) + \\ &\max(0, 0.86 - 0.28 + 1) \\ &= \\ &\mathbf{1.58} \end{aligned}$$

cross-entropy loss (Softmax)

-2.85
0.86
0.28

$\exp$

0.058
2.36
1.32

$\xrightarrow{\text{normalize}}$
(to sum to one)

0.016
0.631
0.353

$$\begin{aligned} &-\log(0.353) \\ &= \\ &\mathbf{0.452} \end{aligned}$$

Softmax vs. SVM

- Interpreting the probabilities from the Softmax

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

Softmax vs. SVM

- Interpreting the probabilities from the Softmax

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

suppose the weights W were only half as large

(we use a higher regularization strength)

Softmax vs. SVM

- Interpreting the probabilities from the Softmax

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

suppose the weights W were only half as large:

$$[0.5, -1, 0] \rightarrow [e^{0.5}, e^{-1}, e^0] = [1.65, 0.37, 1] \rightarrow [0.55, 0.12, 0.33]$$

Softmax vs. SVM

- Interpreting the probabilities from the Softmax

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

suppose the weights W were only half as large:

$$[0.5, -1, 0] \rightarrow [e^{0.5}, e^{-1}, e^0] = [1.65, 0.37, 1] \rightarrow [0.55, 0.12, 0.33]$$

What happens in the limit, as the regularization strength goes to infinity?

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + 1)$$

Softmax vs. SVM

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

scores:

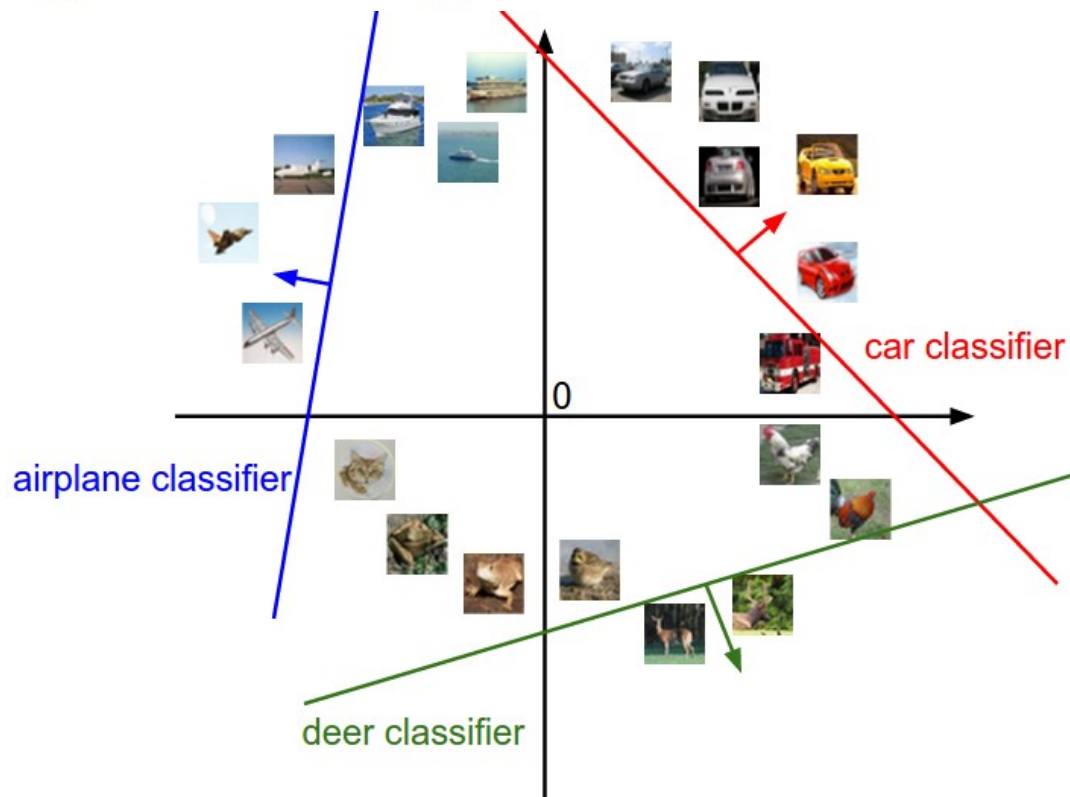
[10, -2, 3]

[10, 9, 9]

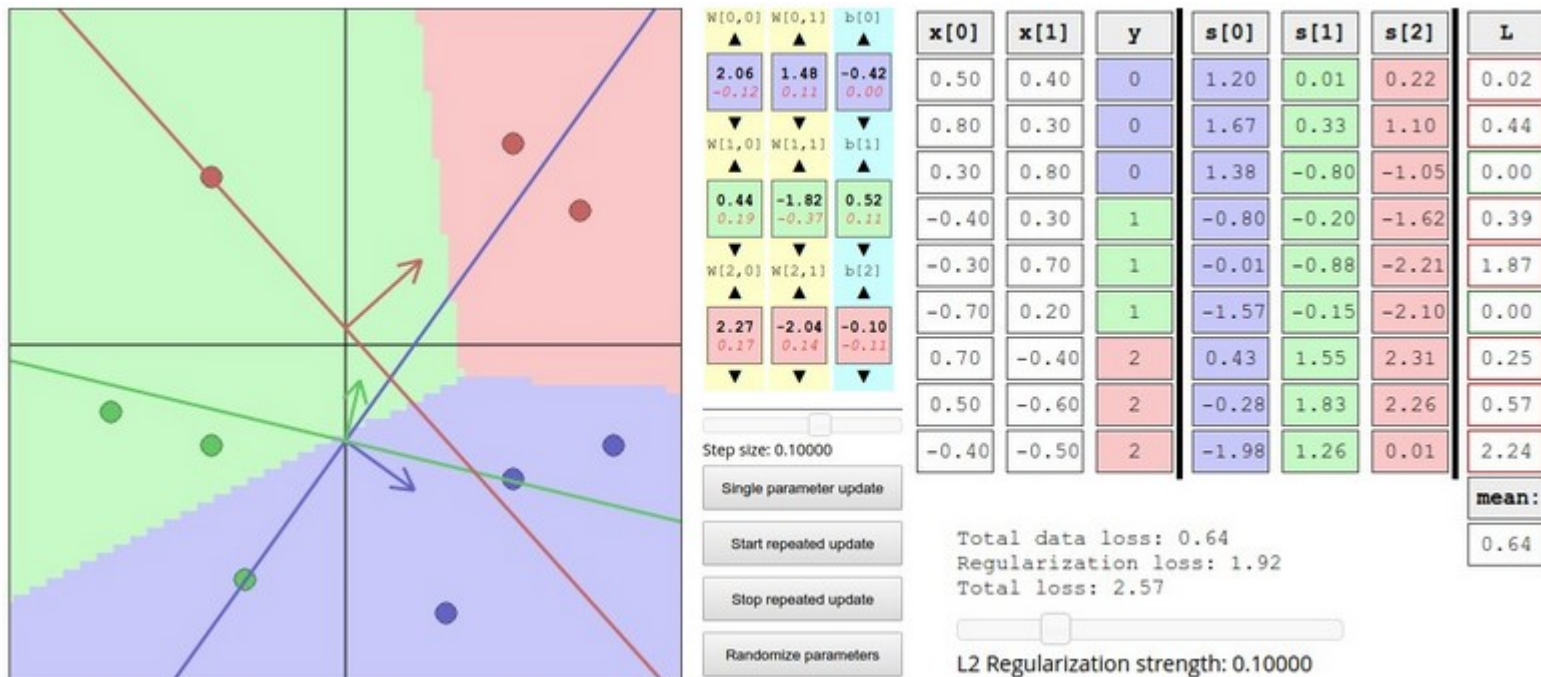
$y_i = 0$ [10, -100, -100]

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + 1)$$

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



Interactive Web Demo time....



<http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/>

Summary

- We introduced a **parametric approach** to image classification
- We defined a **score function** (linear map)
- We defined a **loss function** (SVM / Softmax)

One problem remains: How to find W, b ?

Next class: Optimization, Backpropagation

Find the W, b that minimizes the loss function.