# Lecture 3:
# Linear Classifiers

# Reminder: Assignment 1

- Due **Friday 1/14, 11:59pm EST**
- If you enroll late, you get a free extension for A1:
  - due_date = latest_day(original_due_date, your_enroll_date + 7 days)
- Make sure you submit the right .py file!
  - Make sure to **manually save** the .py file in Colab
  - After you download the .zip file, **check that the .py file is correct**

# Office Hours

- Check Google Calendar (link also on website): https://calendar.google.com/calendar/b/0?cid=dW1pY2guZWR1X2cxMXJnNnZxNmd2YWNqOWRhZDRxOHVvZHNvQGdyb3VwLmNhbGVuZGFyLmdvb2dsZS5jb20
- Office hours may shift a bit from week to week (especially mine) – check Google Calendar for up-to-date info
- We'll use Umich office hours queue system; find link in the description of each calendar event

# Last time: Image Classification

**Input**: image

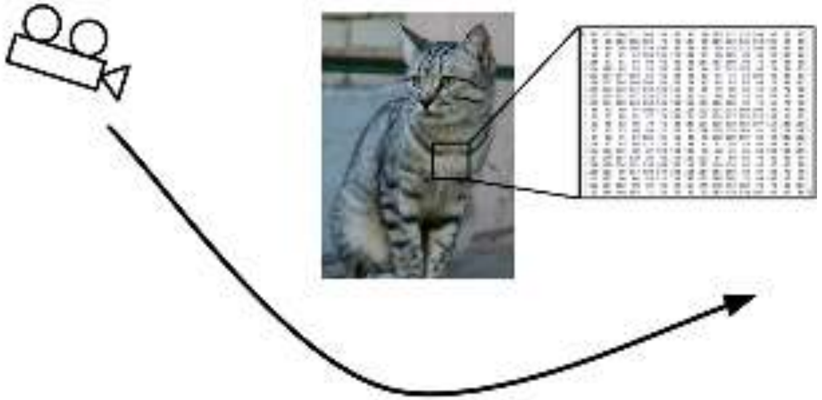


**Output**: Assign image to one of a fixed set of categories

cat
bird
deer
dog
truck

# Last Time: Challenges of Recognition

## Viewpoint



## Illumination



## Deformation



## Occlusion



## Clutter

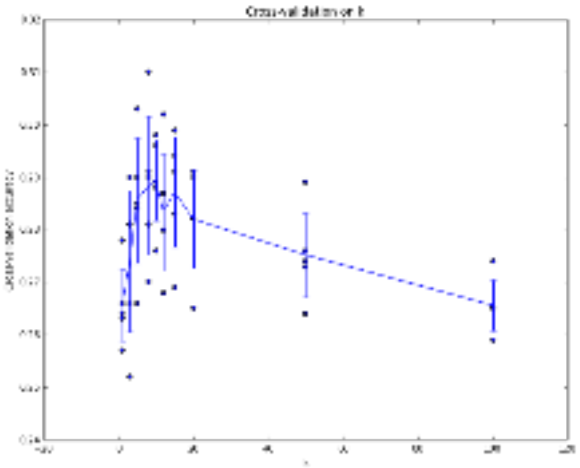


## Intraclass Variation

# Last time: Data-Drive Approach, kNN


1-NN classifier

5-NN classifier



| train | test |
|---|---|

| train | validation | test |
|---|---|---|

# Today: Linear Classifiers

# Neural Network

Linear classifiers

This image is CC0 1.0 public domain

# Recall CIFAR10


**50,000** training images
each image is **32x32x3**

**10,000** test images.

# Parametric Approach

Image

Array of **32x32x3** numbers (3072 numbers total)

$\longrightarrow$ f(**x**,**W**) $\longrightarrow$ **10** numbers giving class scores

$\uparrow$

**W**

parameters or weights

# Parametric Approach: Linear Classifier

$$f(x,W) = Wx$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W})$

**10** numbers giving
class scores

W
parameters
or weights

# Parametric Approach: Linear Classifier

$$\underbrace{\boxed{f(x,W)}}_{(10,)} = \underbrace{\boxed{W}}_{(10,\ 3072)}\overbrace{\boxed{x}}^{(3072,)}$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$\longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$ **10** numbers giving class scores

$\uparrow$

**W**
parameters
or weights

# Parametric Approach: Linear Classifier

(3072,)

$$\boxed{f(x,W)} = \boxed{W}\boxed{x} + \boxed{b}$$

(10,)

(10,) (10, 3072)

Image

Array of **32x32x3** numbers
(3072 numbers total)

$\longrightarrow$ f(**x**,**W**) $\longrightarrow$ **10** numbers giving class scores

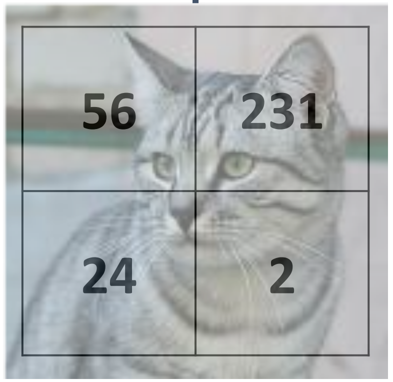
**W**

parameters
or weights

# Example for 2x2 image, 3 classes (cat/dog/ship)

Stretch pixels into column

$f(x,W) = Wx + b$




Input image
(2, 2)

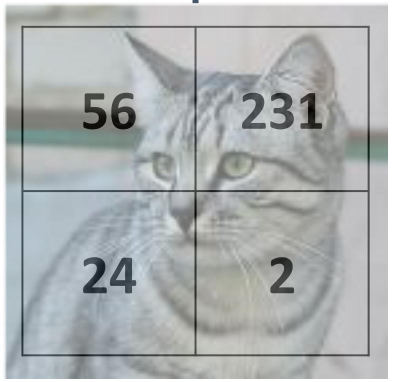
| |
|---|
| 56 |
| 231 |
| 24 |
| 2 |

(4,)

# Example for 2x2 image, 3 classes (cat/dog/ship)

Stretch pixels into column

$f(x,W) = Wx + b$



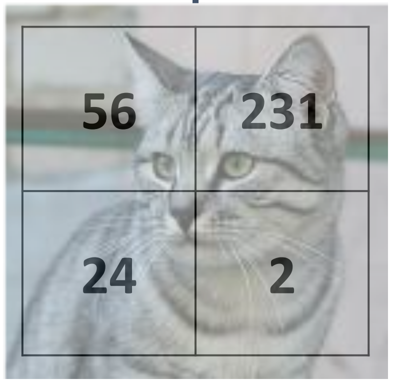

Input image
(2, 2)

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W\ (3,\,4)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{(3,)}$$

# Linear Classifier: Algebraic Viewpoint

Stretch pixels into column

$f(x,W) = Wx + b$



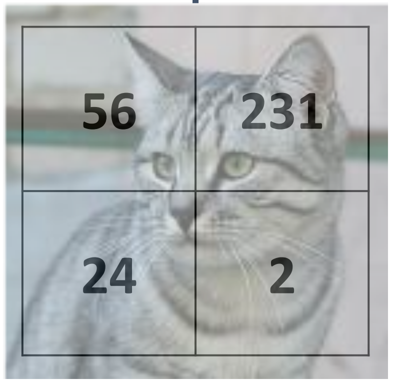

Input image
(2, 2)

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W\ (3,\,4)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{(3,)}$$

# Linear Classifier: Bias Trick

Add extra one to data vector; bias is absorbed into last column of weight matrix

Stretch pixels into column




Input image
(2, 2)

| | | | | |
|---|---|---|---|---|
| 0.2 | -0.5 | 0.1 | 2.0 | 1.1 |
| 1.5 | 1.3 | 2.1 | 0.0 | 3.2 |
| 0 | 0.25 | 0.2 | -0.3 | -1.2 |

W (3, 5)

| |
|---|
| 56 |
| 231 |
| 24 |
| 2 |
| 1 |

(5,)

=

| |
|---|
| -96.8 |
| 437.9 |
| 61.95 |

(3,)

# Linear Classifier: Predictions are Linear!

$$f(x, W) = Wx \quad \text{(ignore bias)}$$

$$f(cx, W) = W(cx) = c * f(x, W)$$

# Linear Classifier: Predictions are Linear!

$$f(x, W) = Wx \quad \text{(ignore bias)}$$

$$f(cx, W) = W(cx) = c * f(x, W)$$

Image

Scores

| |
|---|
| -96.8 |
| 437.8 |
| 62.0 |

0.5 * Image

0.5 * Scores

| |
|---|
| -48.4 |
| 218.9 |
| 31.0 |

# Interpreting a Linear Classifier

## Algebraic Viewpoint

$$f(x,W) = Wx + b$$

Stretch pixels into column


Input image
(2, 2)

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W\ (3,\ 4)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{(3,)}$$

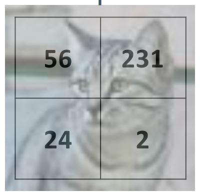
# Interpreting a Linear Classifier

Instead of stretching pixels into columns, we can equivalently stretch rows of W into images!

## Algebraic Viewpoint

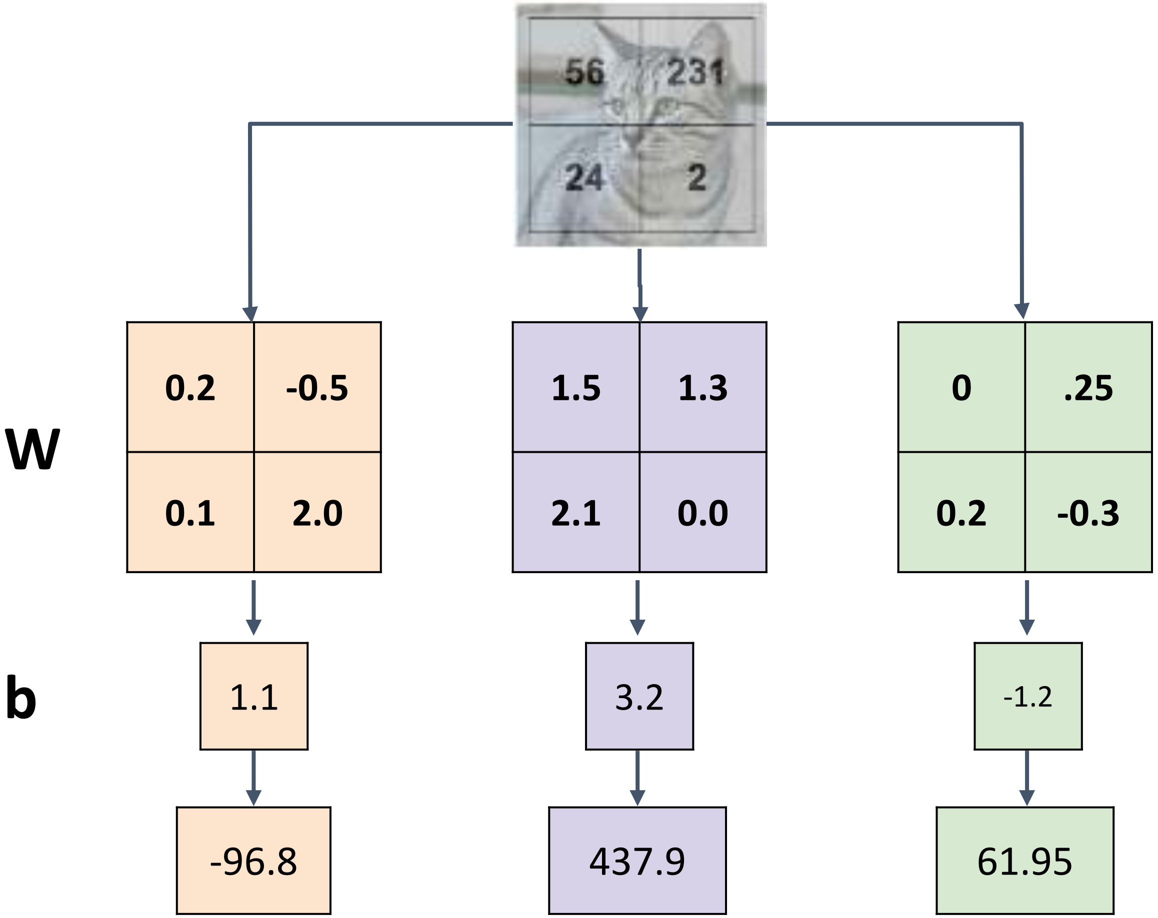
$$f(x,W) = Wx + b$$

Stretch pixels into column

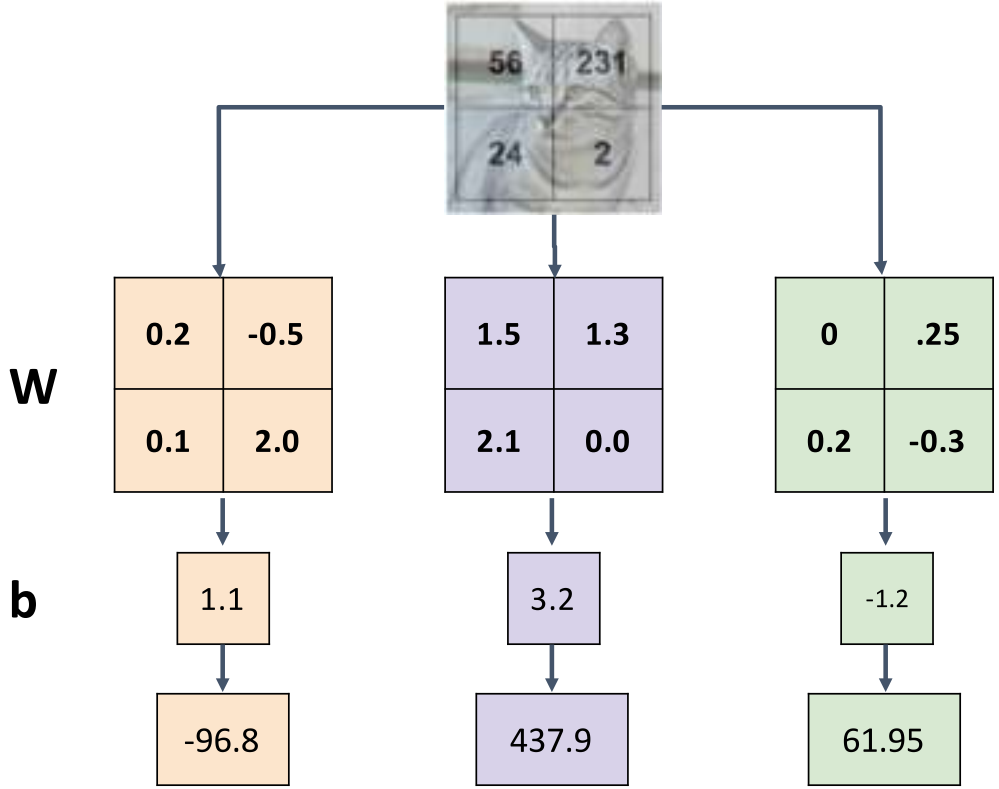



Input image
(2, 2)

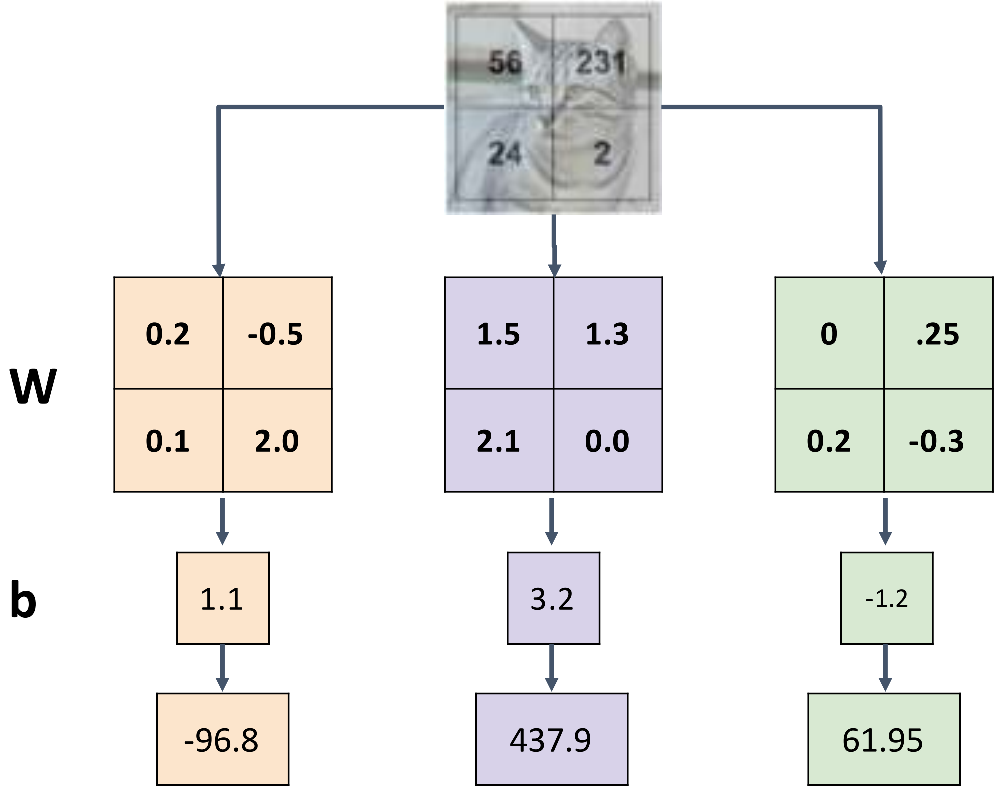
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W\ (3,\,4)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{(3,)}$$



# Interpreting an Linear Classifier

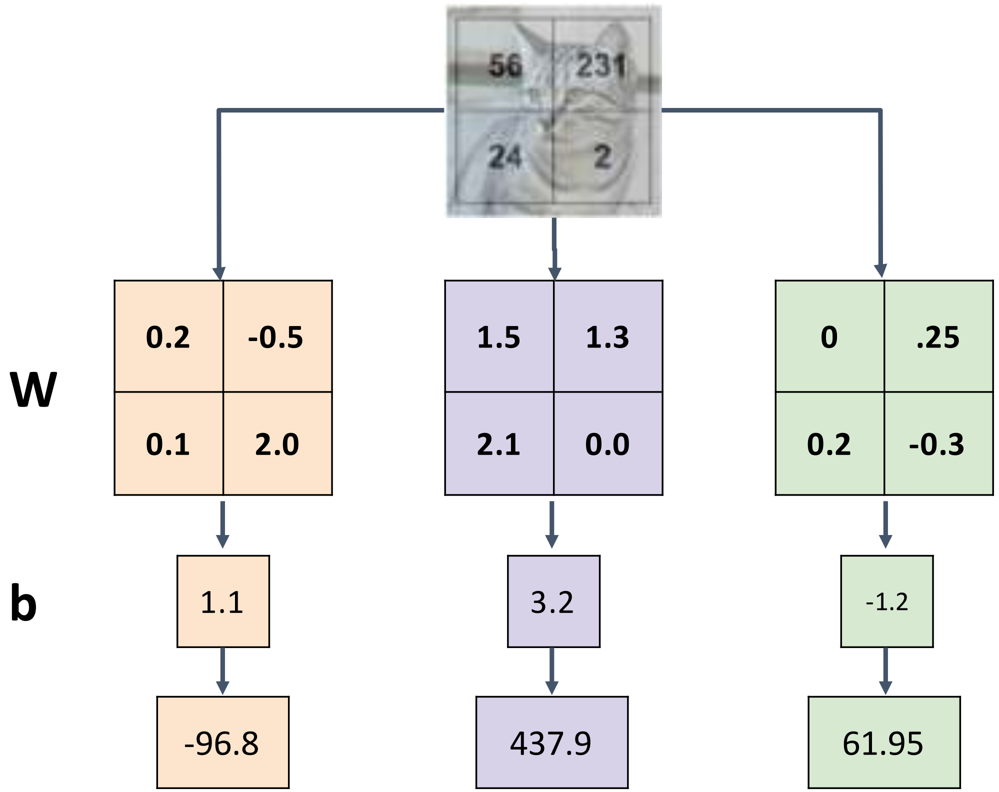



# Interpreting an Linear Classifier: Visual Viewpoint


# Interpreting an Linear Classifier: Visual Viewpoint

Linear classifier has one "template" per category

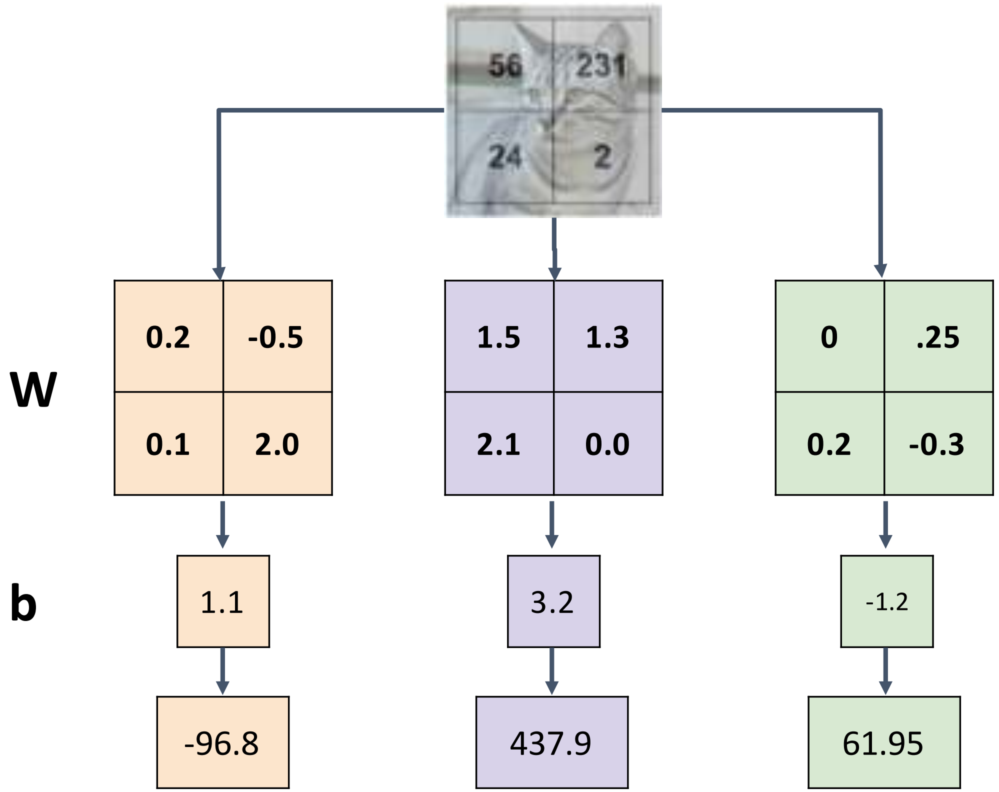



# Interpreting an Linear Classifier: Visual Viewpoint

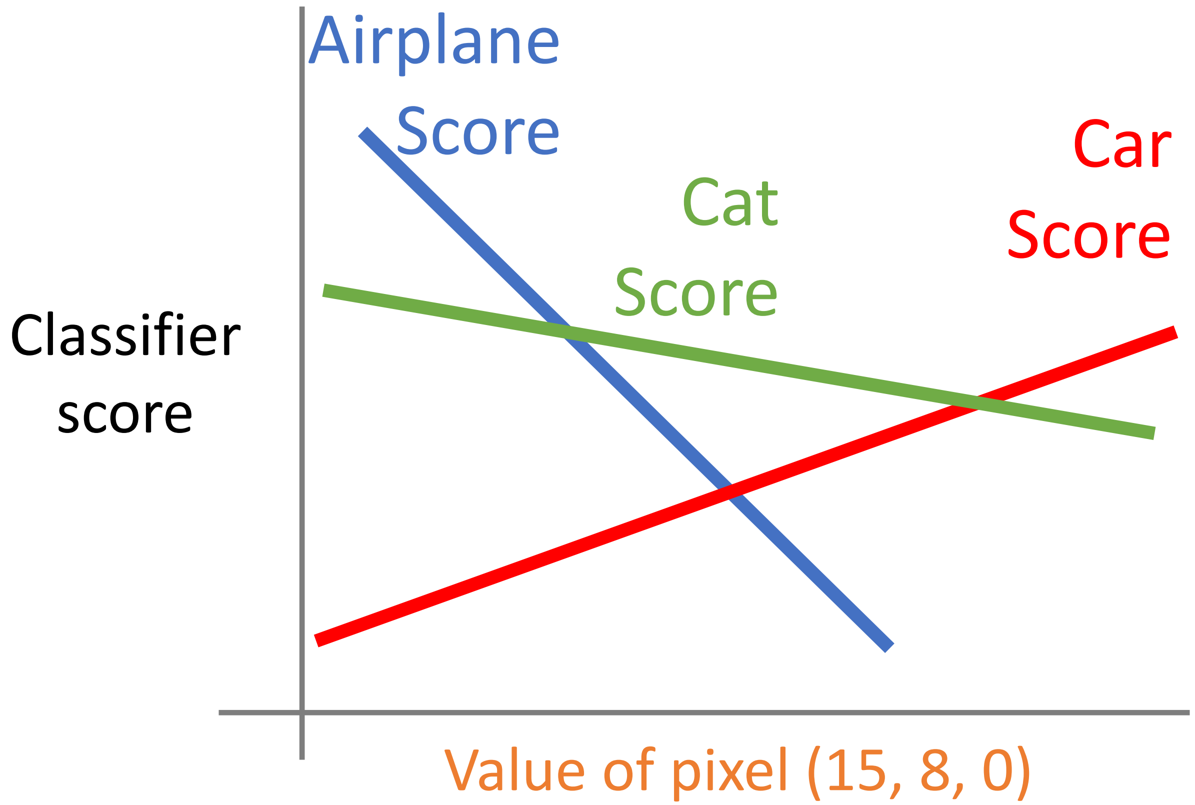
Linear classifier has one "template" per category

A single template cannot capture multiple modes of the data

e.g. horse template has 2 heads!



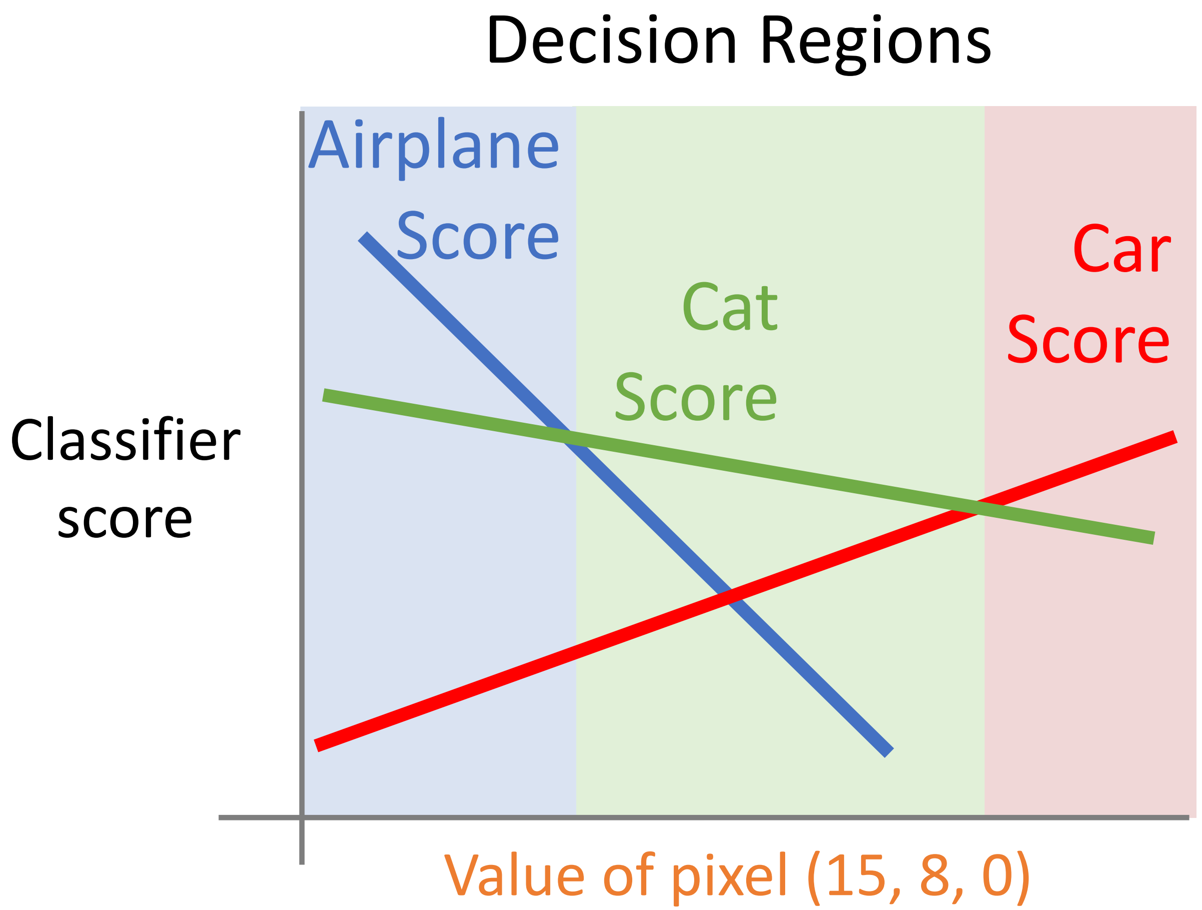

# Interpreting a Linear Classifier: Geometric Viewpoint




$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

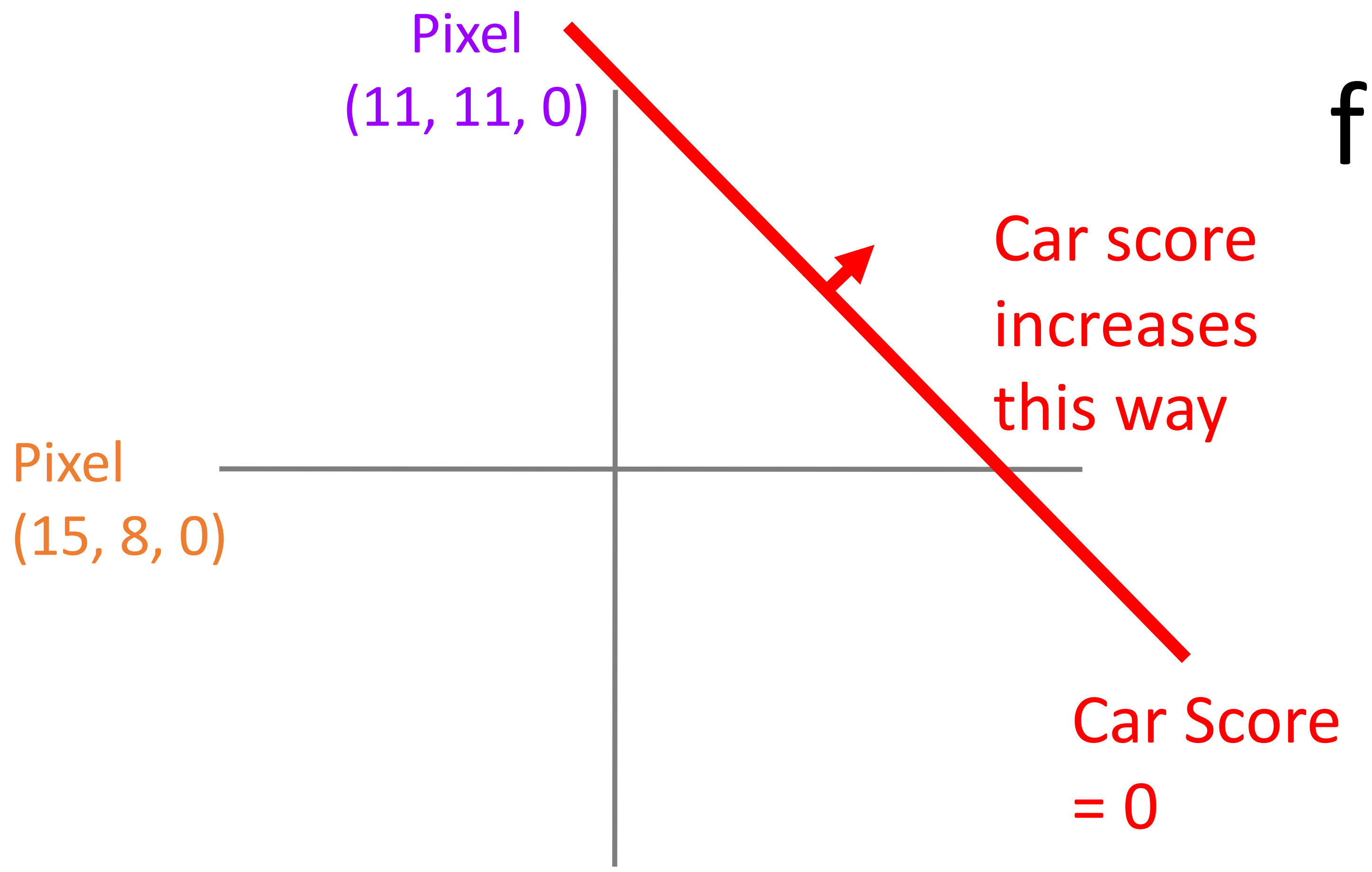
# Interpreting a Linear Classifier: Geometric Viewpoint




$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

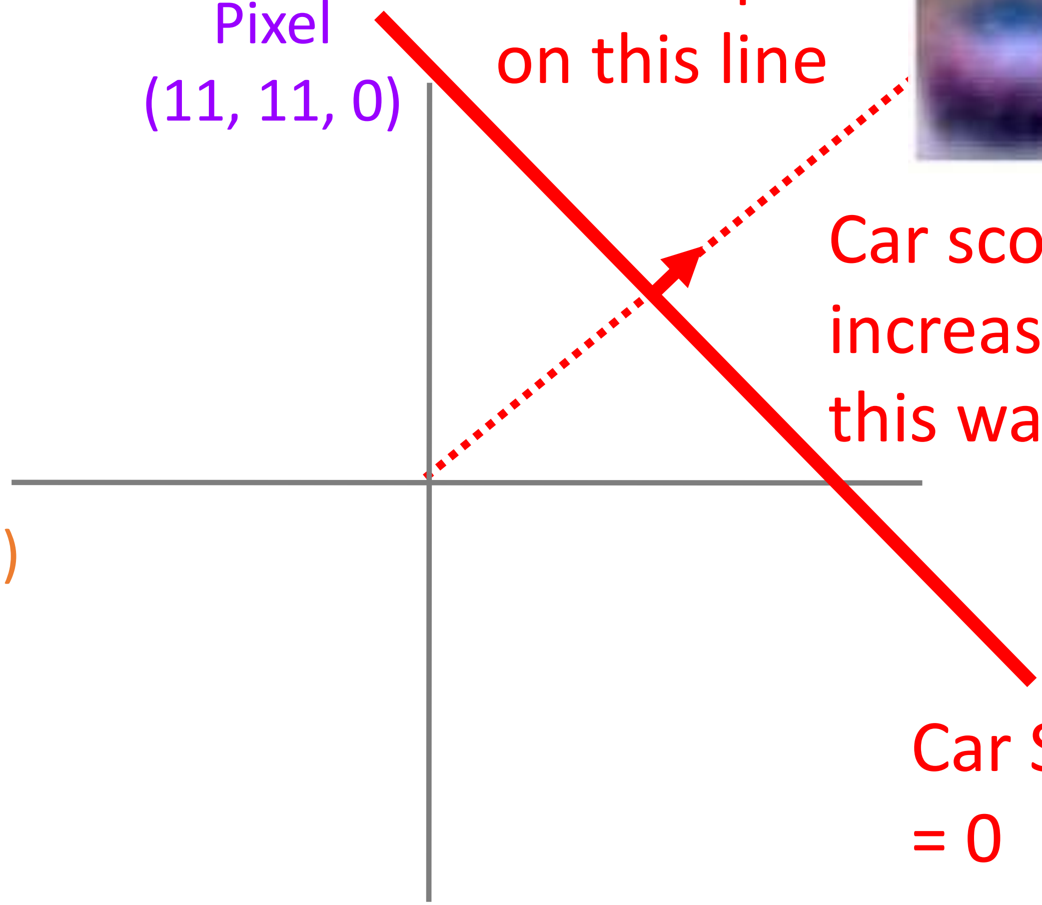
# Interpreting a Linear Classifier: Geometric Viewpoint




$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

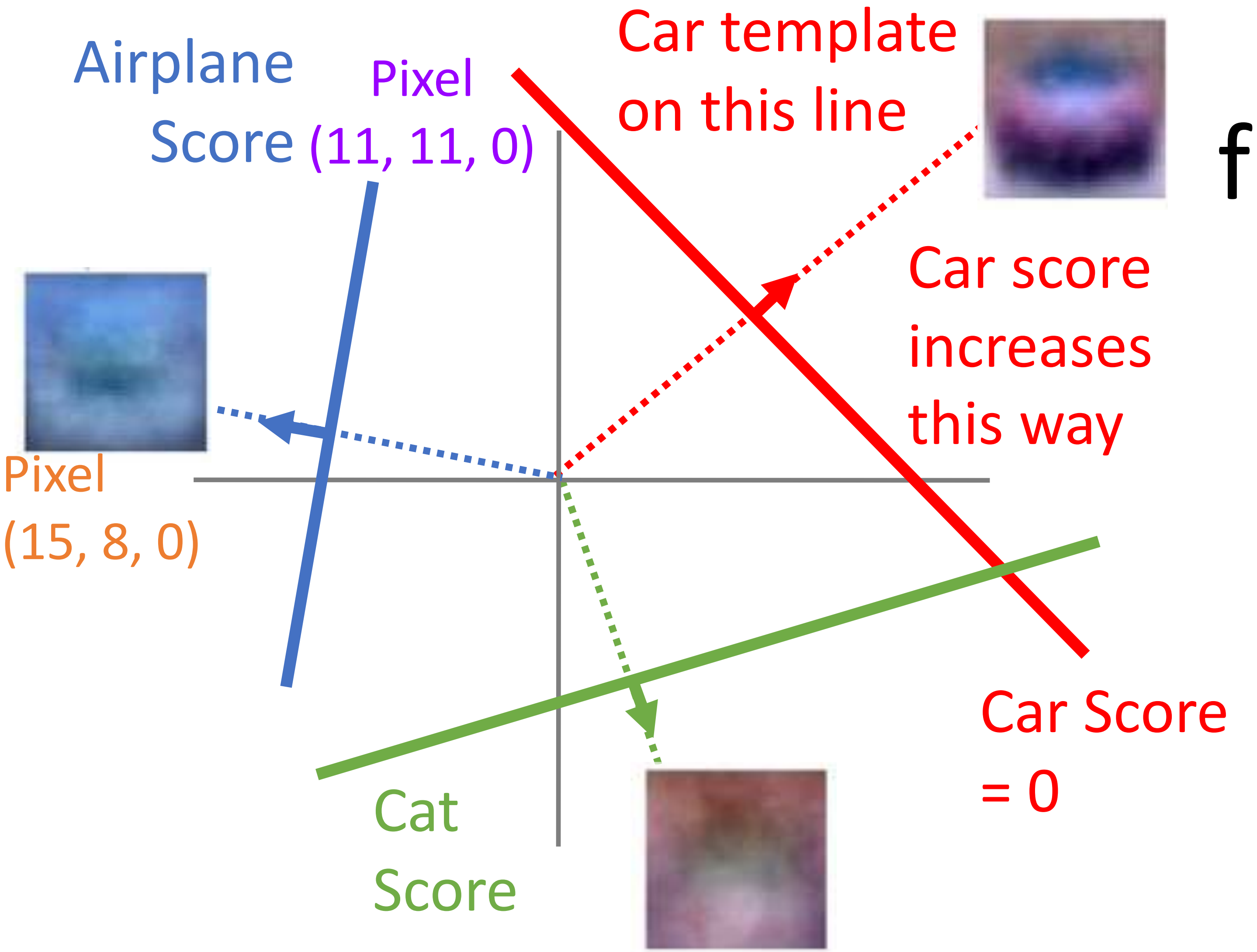
# Interpreting a Linear Classifier: Geometric Viewpoint




$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

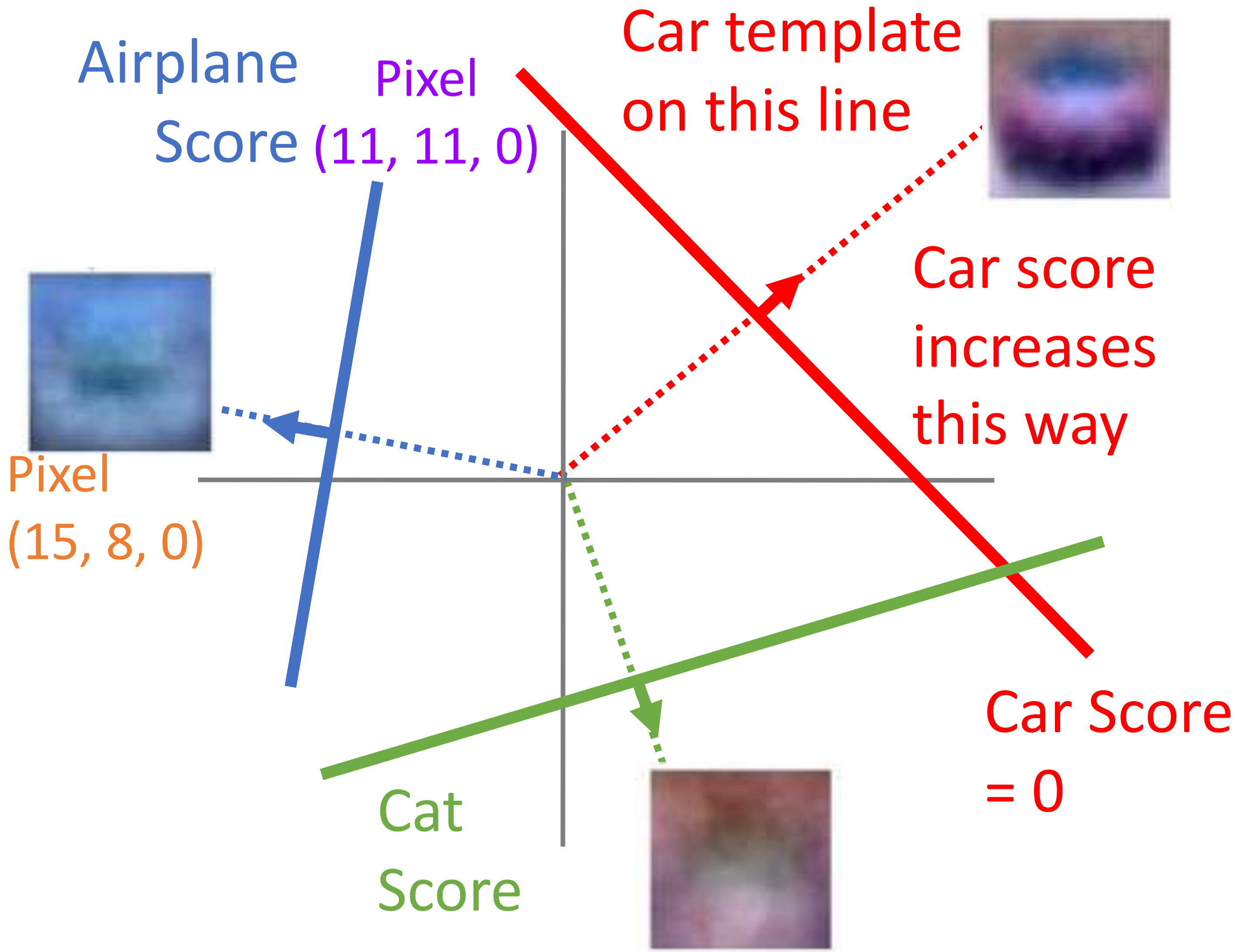
# Interpreting a Linear Classifier: Geometric Viewpoint



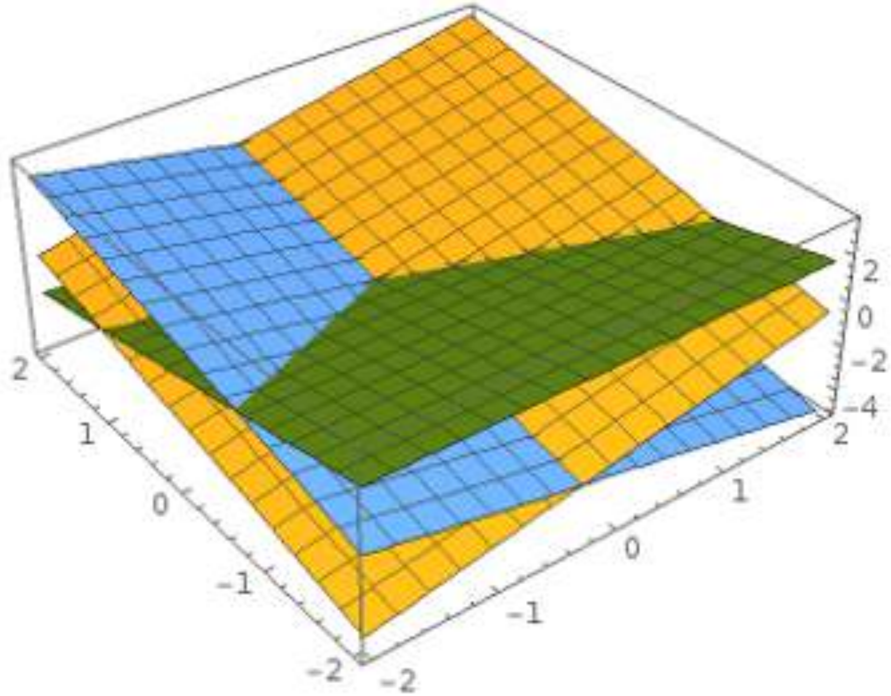

$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

# Interpreting a Linear Classifier: Geometric Viewpoint




Hyperplanes carving up a high-dimensional space

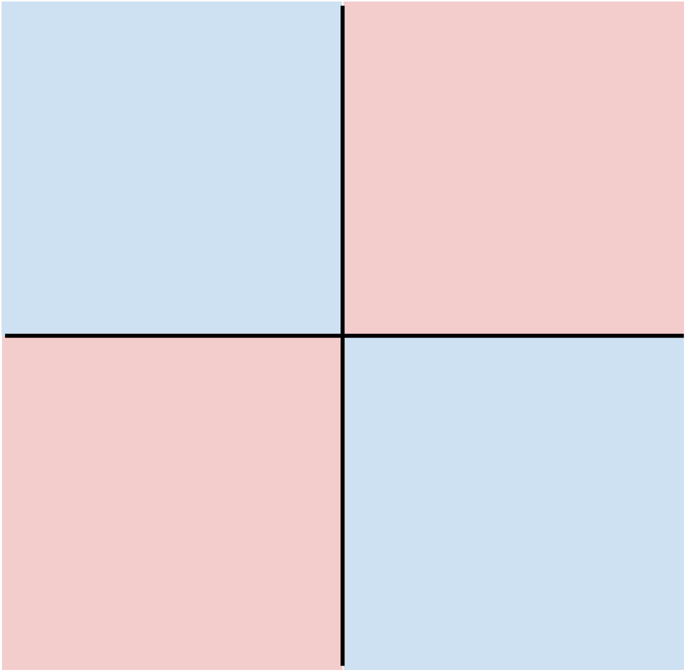



Plot created using [Wolfram Cloud]

# Hard Cases for a Linear Classifier

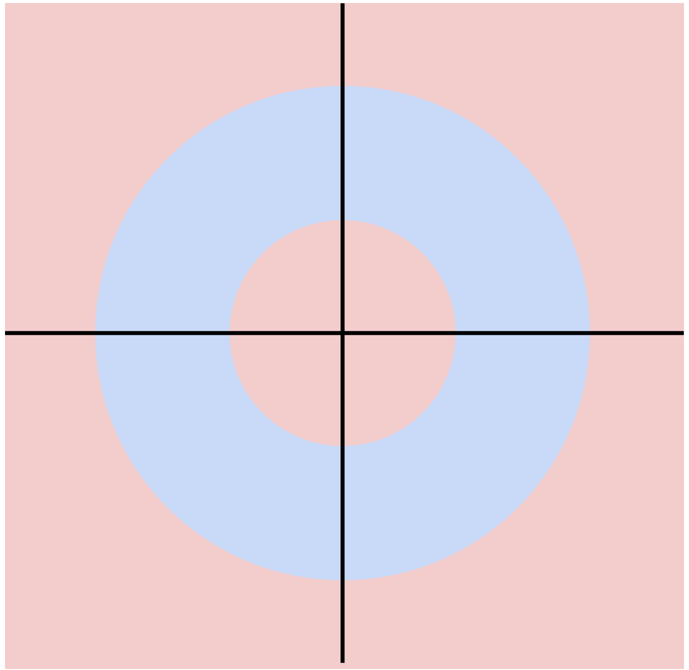
**Class 1**:
First and third quadrants

**Class 2**:
Second and fourth quadrants



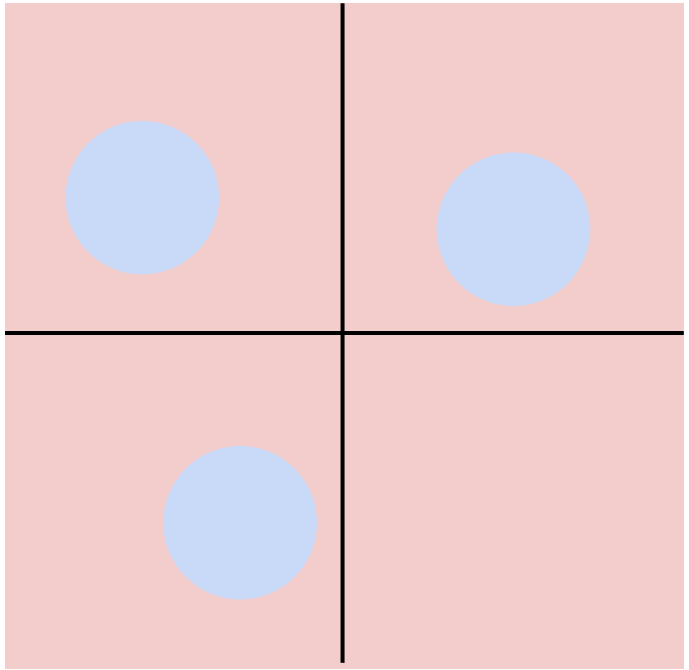
**Class 1**:
1 <= L2 norm <= 2

**Class 2**:
Everything else



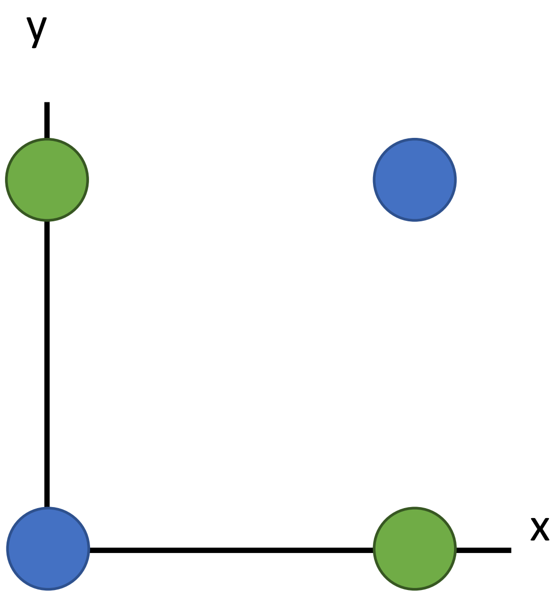
**Class 1**:
Three modes

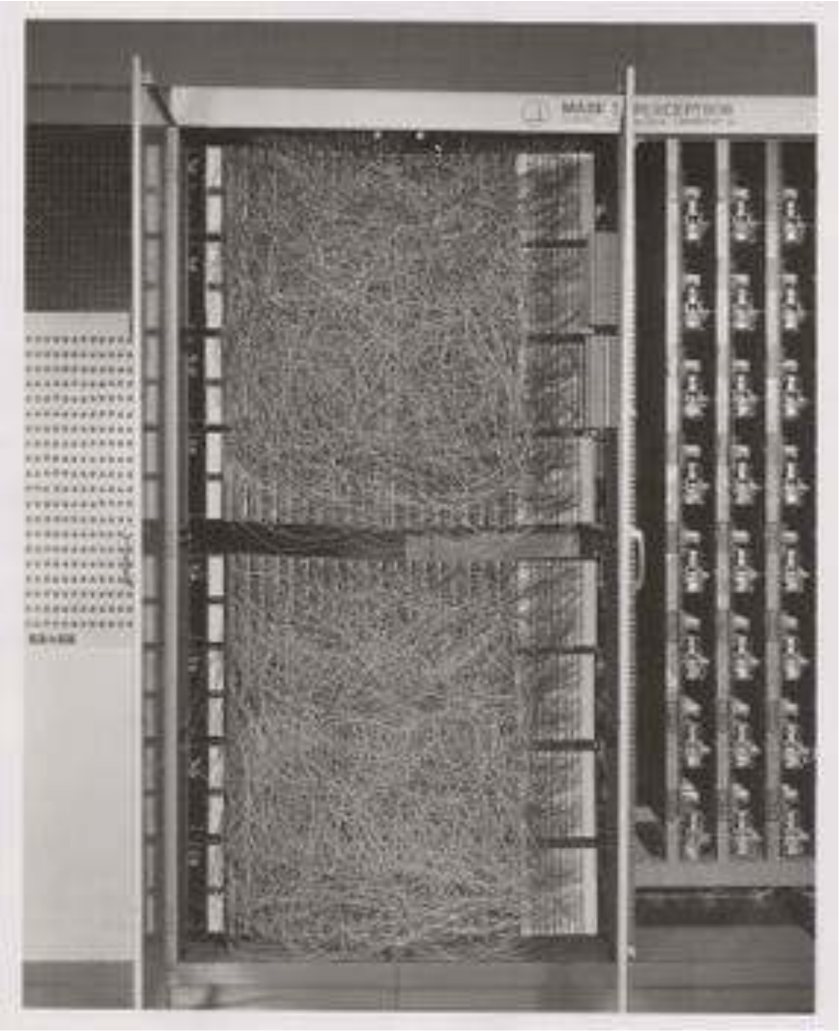
**Class 2**:
Everything else



# Recall: Perceptron couldn’t learn XOR

| X | Y | F(x,y) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

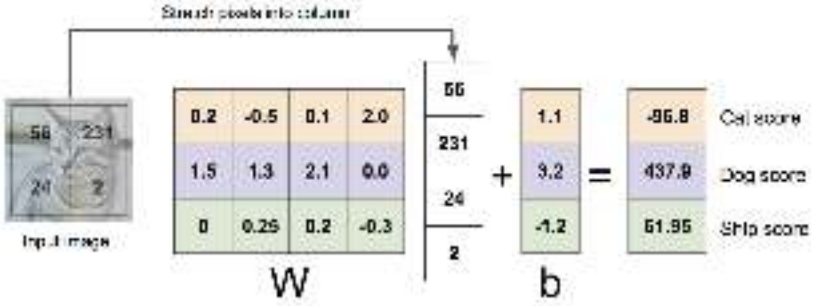


# Linear Classifier: Three Viewpoints

## Algebraic Viewpoint

$f(x,W) = Wx$

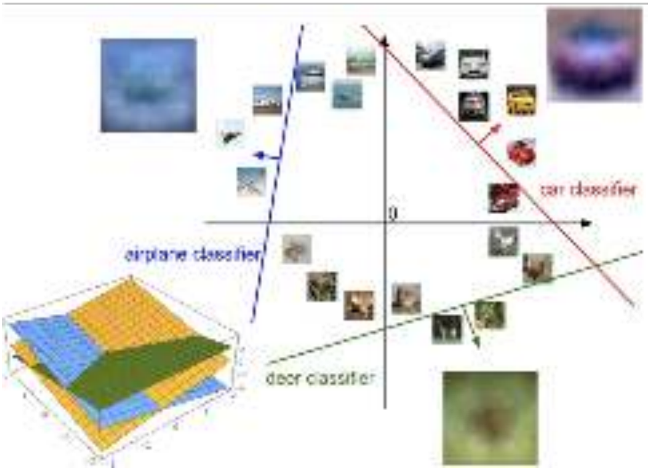



## Visual Viewpoint

One template per class


## Geometric Viewpoint

Hyperplanes cutting up space



# So Far: Defined a linear score function

$f(x,W) = Wx + b$



| | | | |
|---|---|---|---|
| airplane | -3.45 | -0.51 | 3.42 |
| automobile | -8.87 | **6.04** | 4.64 |
| bird | 0.09 | 5.31 | 2.65 |
| cat | **2.9** | -4.22 | 5.1 |
| deer | 4.48 | -4.19 | 2.64 |
| dog | 8.02 | 3.58 | 5.55 |
| frog | 3.78 | 4.49 | **-4.34** |
| horse | 1.06 | -4.37 | -1.5 |
| ship | -0.36 | -2.09 | -4.79 |
| truck | -0.72 | -2.93 | 6.14 |

Given a W, we can compute class scores for an image x.

But how can we actually choose a good W?

# Choosing a good W

$f(x,W) = Wx + b$

| | | | |
|---|---|---|---|
| airplane | -3.45 | -0.51 | 3.42 |
| automobile | -8.87 | **6.04** | 4.64 |
| bird | 0.09 | 5.31 | 2.65 |
| cat | **2.9** | -4.22 | 5.1 |
| deer | 4.48 | -4.19 | 2.64 |
| dog | 8.02 | 3.58 | 5.55 |
| frog | 3.78 | 4.49 | **-4.34** |
| horse | 1.06 | -4.37 | -1.5 |
| ship | -0.36 | -2.09 | -4.79 |
| truck | -0.72 | -2.93 | 6.14 |

TODO:

1. Use a **loss function** to quantify how good a value of W is

2. Find a W that minimizes the loss function (**optimization**)

# Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

# Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

# Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where $x_i$ is image and
$y_i$ is (integer) label

# Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where $x_i$ is image and
$y_i$ is (integer) label

Loss for a single example is

$$L_i(f(x_i, W), y_i)$$

# Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where $x_i$ is image and
$y_i$ is (integer) label

Loss for a single example is

$$L_i(f(x_i, W), y_i)$$

Loss for the dataset is average of per-example losses:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ Softmax function

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}}$$

Softmax function

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

Unnormalized log-probabilities / logits

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}}$$ Softmax function

Probabilities must be >= 0

| | Unnormalized log-probabilities / logits | $\xrightarrow{\text{exp}}$ | unnormalized probabilities |
|---|---|---|---|
| cat | **3.2** | | **24.5** |
| car | 5.1 | | 164.0 |
| frog | -1.7 | | 0.18 |

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ Softmax function

Probabilities must be >= 0

Probabilities must sum to 1

| | Unnormalized log-probabilities / logits | $\xrightarrow{\text{exp}}$ | unnormalized probabilities | $\xrightarrow{\text{normalize}}$ | probabilities |
|---|---|---|---|---|---|
| cat | **3.2** | | **24.5** | | **0.13** |
| car | 5.1 | | 164.0 | | 0.87 |
| frog | -1.7 | | 0.18 | | 0.00 |

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ Softmax function

$$L_i = -\log P(Y = y_i \mid X = x_i)$$

Probabilities must be >= 0

Probabilities must sum to 1

| | Unnormalized log-probabilities / logits | | unnormalized probabilities | | probabilities |
|---|---|---|---|---|---|
| cat | **3.2** | exp → | **24.5** | normalize → | **0.13** |
| car | 5.1 | | 164.0 | | 0.87 |
| frog | -1.7 | | 0.18 | | 0.00 |

$L_i$ = -log(0.13)
= **2.04**

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ Softmax function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i \mid X = x_i)$$

| | Unnormalized log-probabilities / logits | | unnormalized probabilities | | probabilities |
|---|---|---|---|---|---|
| cat | **3.2** | $\xrightarrow{\text{exp}}$ | **24.5** | $\xrightarrow{\text{normalize}}$ | **0.13** |
| car | 5.1 | | 164.0 | | 0.87 |
| frog | -1.7 | | 0.18 | | 0.00 |

$L_i = -\log(0.13)$
$= \mathbf{2.04}$

**Maximum Likelihood Estimation**
Choose weights to maximize the likelihood of the observed data
(See EECS 445 or EECS 545)

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}}$$ Softmax function

Probabilities must be >= 0

Probabilities must sum to 1

$$\boxed{L_i = -\log P(Y = y_i \mid X = x_i)}$$

| | Unnormalized log-probabilities / logits | | unnormalized probabilities | | probabilities | | Correct probs |
|---|---|---|---|---|---|---|---|
| cat | **3.2** | exp → | **24.5** | normalize → | **0.13** | → Compare ← | **1.00** |
| car | 5.1 | | 164.0 | | 0.87 | | 0.00 |
| frog | -1.7 | | 0.18 | | 0.00 | | 0.00 |

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$s = f(x_i; W)$

$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$ Softmax function

$L_i = -\log P(Y = y_i \mid X = x_i)$

Probabilities must be >= 0

Probabilities must sum to 1

| | Unnormalized log-probabilities / logits | | unnormalized probabilities | | probabilities | | Correct probs |
|---|---|---|---|---|---|---|---|
| cat | **3.2** | exp → | **24.5** | normalize → | **0.13** | → Compare ← | **1.00** |
| car | 5.1 | | 164.0 | | 0.87 | | 0.00 |
| frog | -1.7 | | 0.18 | | 0.00 | | 0.00 |

*Kullback–Leibler divergence*

$D_{KL}(P \,||\, Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ Softmax function

$$L_i = -\log P(Y = y_i \mid X = x_i)$$

Probabilities must be >= 0

Probabilities must sum to 1

| | Unnormalized log-probabilities / logits | $\xrightarrow{\text{exp}}$ | unnormalized probabilities | $\xrightarrow{\text{normalize}}$ | probabilities | Compare | Correct probs |
|---|---|---|---|---|---|---|---|
| cat | **3.2** | | **24.5** | | **0.13** | → Compare ← | **1.00** |
| car | 5.1 | | 164.0 | | 0.87 | *Cross Entropy* | 0.00 |
| frog | -1.7 | | 0.18 | | 0.00 | | 0.00 |

$$H(P, Q) = H(P) + D_{KL}(P \,||\, Q)$$

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$

Softmax function

Maximize probability of correct class

$$L_i = -\log P(Y = y_i \mid X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}}$$

Softmax function

Maximize probability of correct class

$$\boxed{L_i = -\log P(Y = y_i \mid X = x_i)}$$

Putting it all together:

$$\boxed{L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)}$$

**Q:** What is the min / max possible loss $L_i$?

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}}$$

Softmax function

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

Maximize probability of correct class

$$\boxed{L_i = -\log P(Y = y_i \mid X = x_i)}$$

Putting it all together:

$$\boxed{L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)}$$

**Q:** What is the min / max possible loss $L_i$?

**A**: Min 0, max +infinity

# Cross-Entropy Loss (Multinomial Logistic Regression)

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$

Softmax function

Maximize probability of correct class

$$L_i = -\log P(Y = y_i \mid X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

**Q:** If all scores are small random values, what is the loss?

# Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

| | |
|---|---|
| cat | **3.2** |
| car | 5.1 |
| frog | -1.7 |

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}}$$

Softmax function

Maximize probability of correct class

$$\boxed{L_i = -\log P(Y = y_i \mid X = x_i)}$$

Putting it all together:

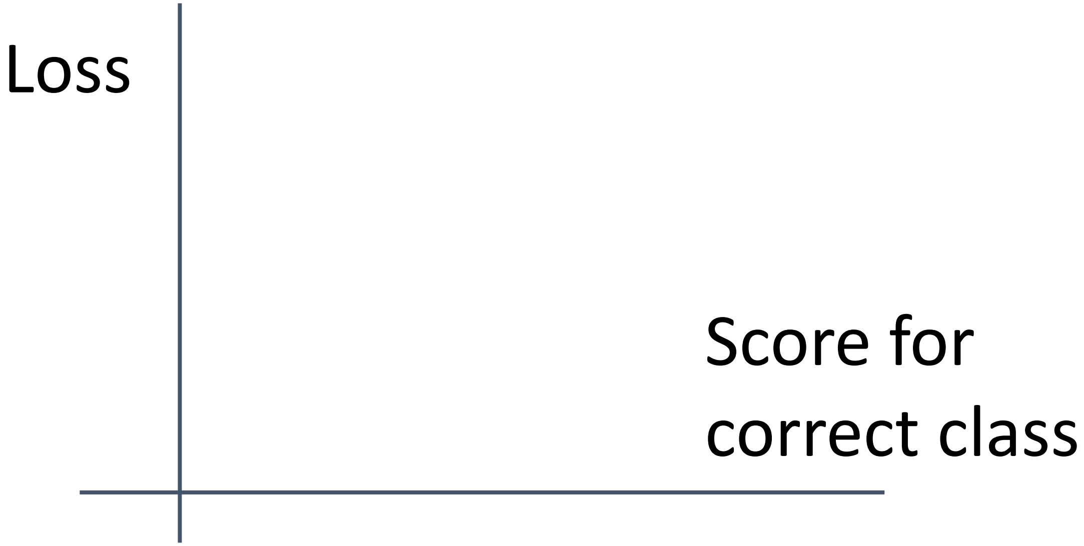
$$\boxed{L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)}$$

**Q:** If all scores are small random values, what is the loss?

**A**: -log(1/C)
log(10) ≈ 2.3

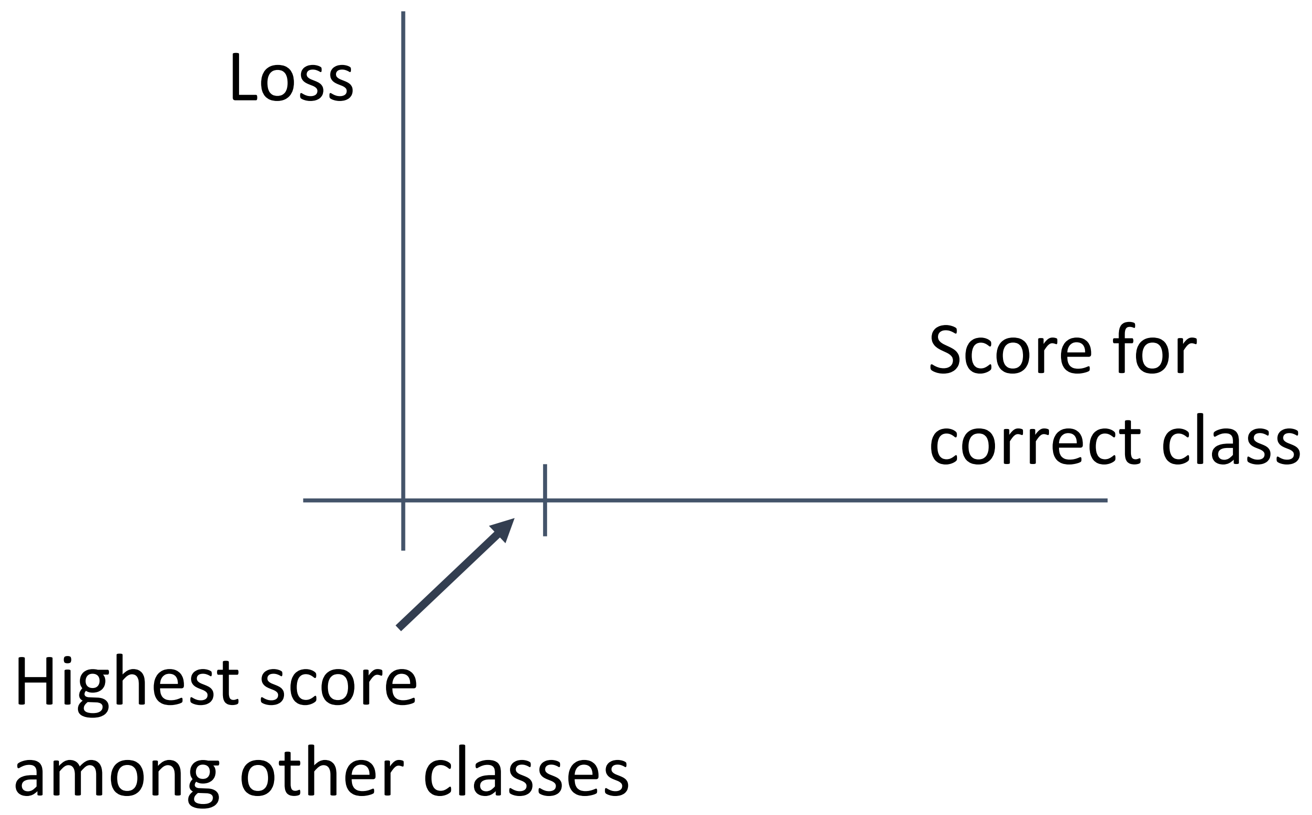
# Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"



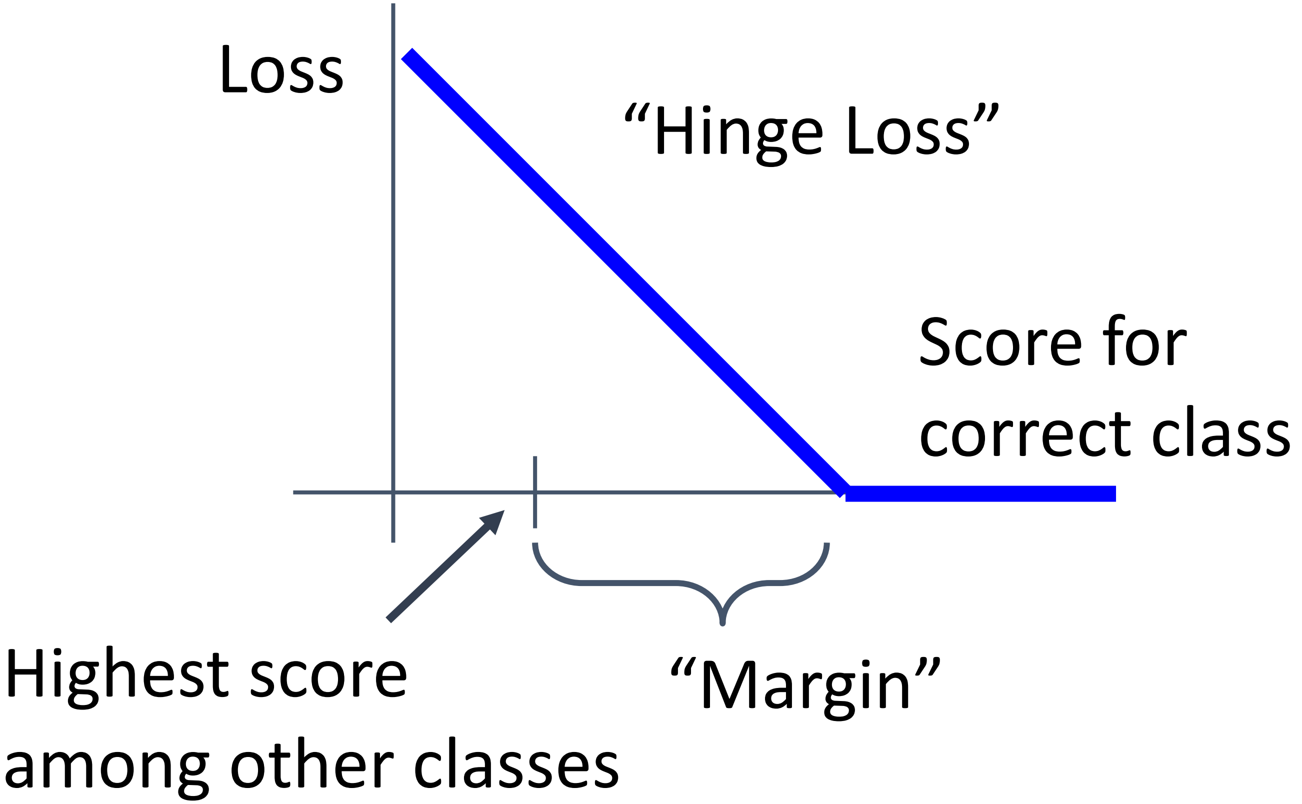
# Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"



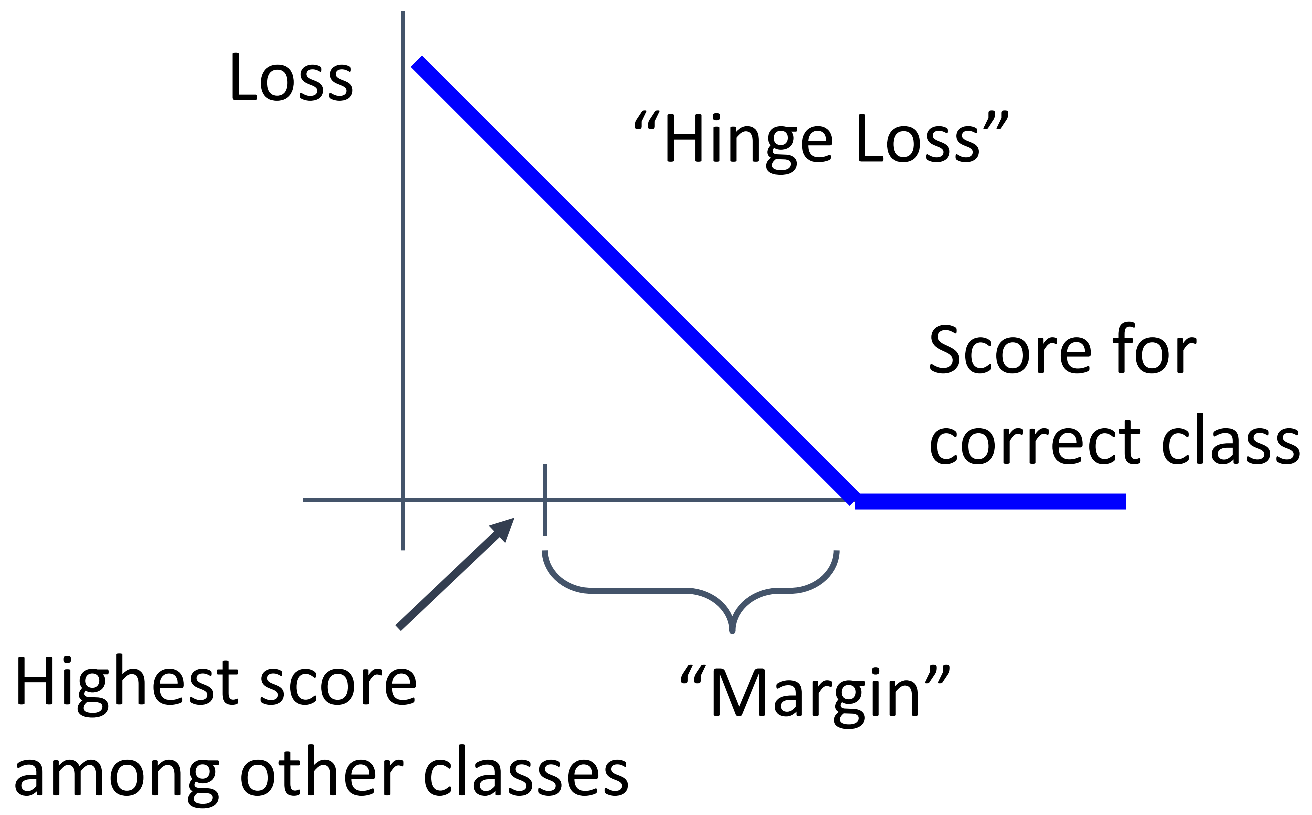
# Multiclass SVM Loss

”The score of the correct class should be higher than all the other scores”



# Multiclass SVM Loss

”The score of the correct class should be higher than all the other scores”




Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | | |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$= \max(0, 5.1 - 3.2 + 1)$
$+ \max(0, -1.7 - 3.2 + 1)$
$= \max(0, 2.9) + \max(0, -3.9)$
$= 2.9 + 0$
$= 2.9$

# Multiclass SVM Loss



| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

# Multiclass SVM Loss



| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Loss over the dataset is:

L = (2.9 + 0.0 + 12.9) / 3
= 5.27

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Q**: What happens to the loss if the scores for the car image change a bit?

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Q2**: What are the min and max possible loss?

# Multiclass SVM Loss



| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Q3**: If all the scores were random, what loss would we expect?

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Q4**: What would happen if the sum were over all classes? (including $i = y_i$)

# Multiclass SVM Loss

| |  |  |  |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Q5**: What if the loss used a mean instead of a sum?

# Multiclass SVM Loss

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |
| Loss | 2.9 | 0 | 12.9 |

Given an example $(x_i, y_i)$
($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Q6**: What if we used this loss instead?

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

# Cross-Entropy vs SVM Loss

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

**Q**: What is cross-entropy loss?

What is SVM loss?

# Cross-Entropy vs SVM Loss

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $y_i = 0$

**Q**: What is cross-entropy loss?
What is SVM loss?

**A**: Cross-entropy loss > 0
SVM loss = 0

# Cross-Entropy vs SVM Loss

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $y_i = 0$

**Q**: What happens to each loss if I slightly change the scores of the last datapoint?

# Cross-Entropy vs SVM Loss

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $y_i = 0$

**Q**: What happens to each loss if I slightly change the scores of the last datapoint?

**A**: Cross-entropy loss will change; SVM loss will stay the same

# Cross-Entropy vs SVM Loss

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $y_i = 0$

**Q**: What happens to each loss if I double the score of the correct class from 10 to 20?

# Cross-Entropy vs SVM Loss

$$L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

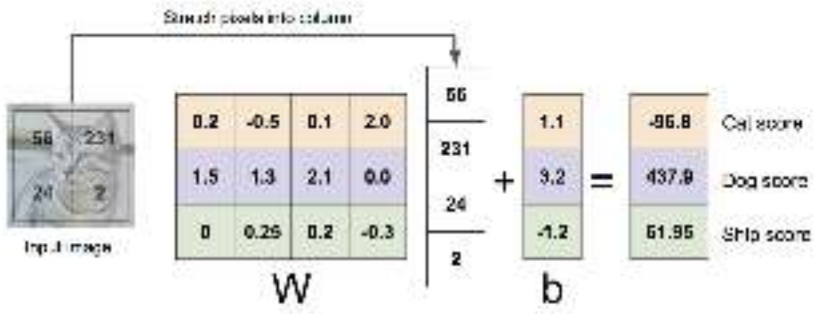
**Q**: What happens to each loss if I double the score of the correct class from 10 to 20?

**A**: Cross-entropy loss will decrease, SVM loss still 0

# Recap: Three ways to think about linear classifiers

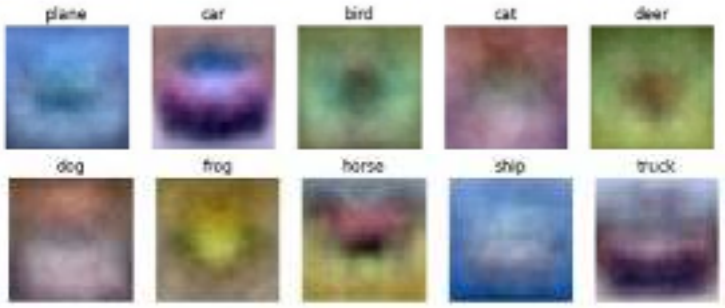
## Algebraic Viewpoint

f(x,W) = Wx



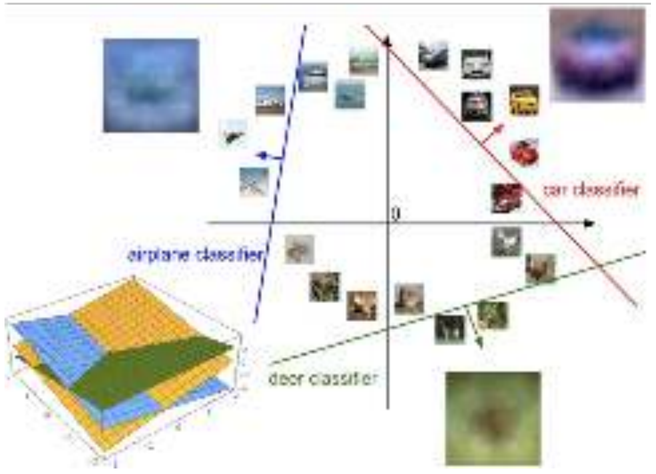

## Visual Viewpoint

One template per class




## Geometric Viewpoint

Hyperplanes cutting up space



# Recap: Loss Functions quantify preferences

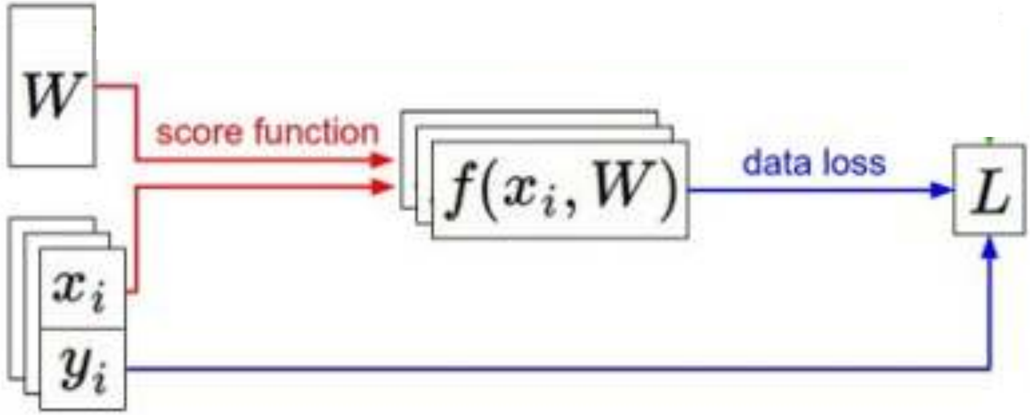
- We have some dataset of (x, y)
- We have a **score function:**
- We have a **loss function**:

$$s = f(x; W, b) = Wx + b$$

Linear classifier

Softmax: $L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$

SVM: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$



# Recap: Loss Functions quantify preferences

- We have some dataset of (x, y)
- We have a **score function:**
- We have a **loss function**:

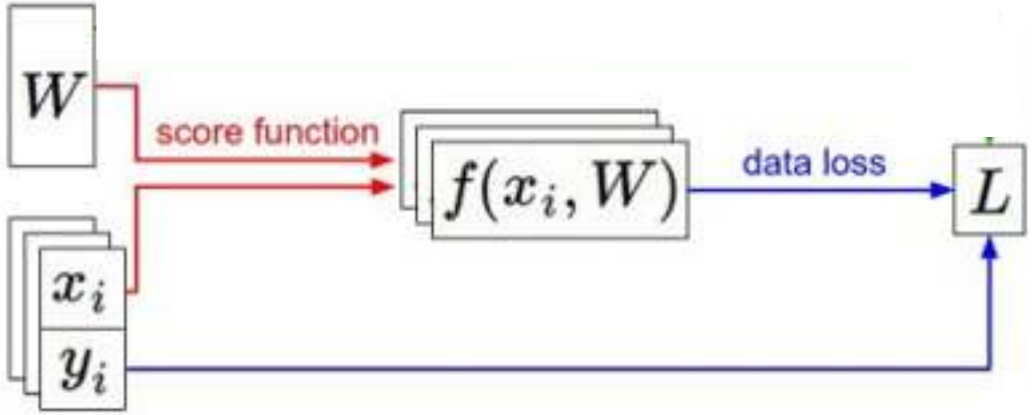
**Q**: How do we find the best W, b?

$$s = f(x; W, b) = Wx + b$$

Linear classifier

Softmax: $L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)$

SVM: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$



# Next time: Regularization + Optimization