Lecture 3:
Linear Classifiers

Reminder: Assignment 1

- http://web.eecs.umich.edu/~justincj/teaching/eecs498/assignment1.html
- Due **Sunday September 15, 11:59pm EST**
- We have written a **homework validation script** to check the format of your .zip file before you submit to Canvas:
- https://github.com/deepvision-class/tools#homework-validation
- This script ensures that your .zip and .ipynb files are properly structured; they do not check correctness
- It is **your responsibility** to make sure your submitted .zip file passes the validation script

Last time: Image Classification

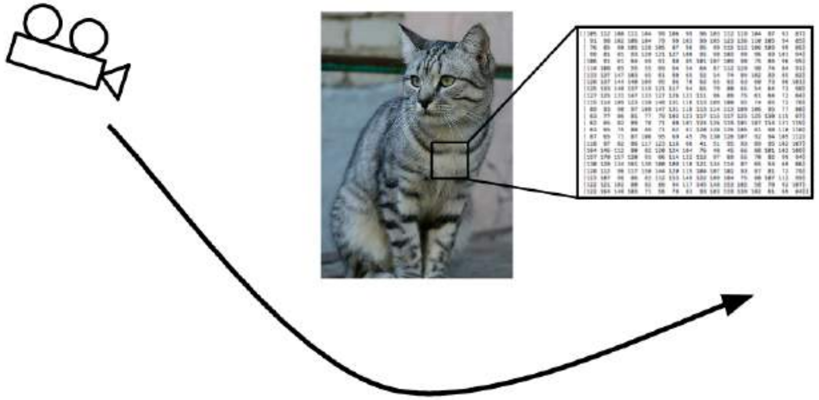
Input: image

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

cat
bird
deer
dog
truck

Last Time: Challenges of Recognition

Viewpoint



Illumination

Deformation

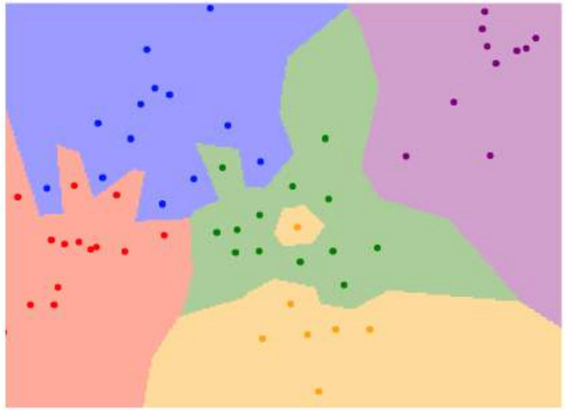
Occlusion

Clutter

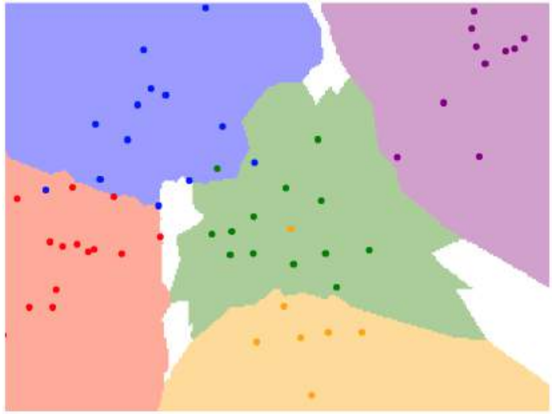
Intraclass Variation

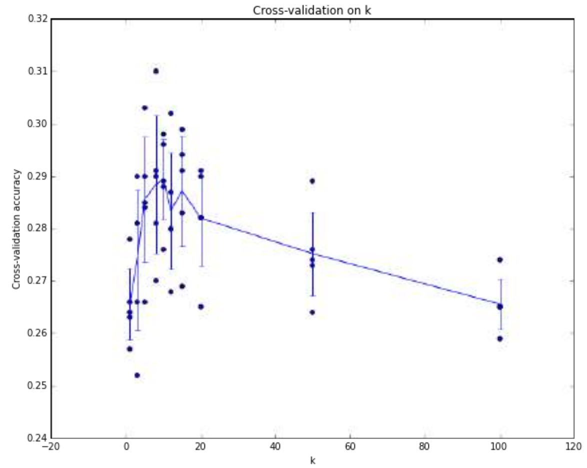
Last time: Data-Drive Approach, kNN

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)

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

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)

$f(x, W)$

10 numbers giving class scores

W

parameters or weights

Parametric Approach: Linear Classifier

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

(3072,)

(10,) (10, 3072)

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

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

Image

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

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

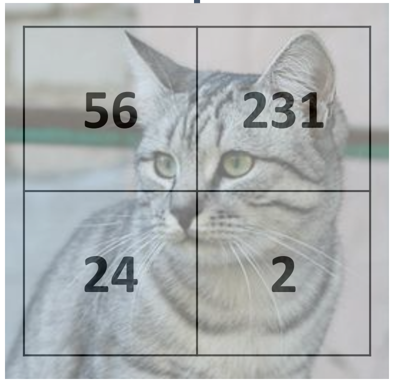
$\uparrow$

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)

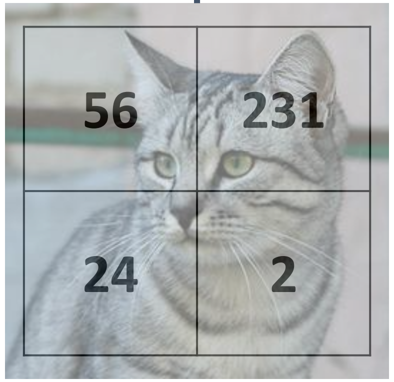
56
231
24
2

(4,)

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

Stretch pixels into column

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



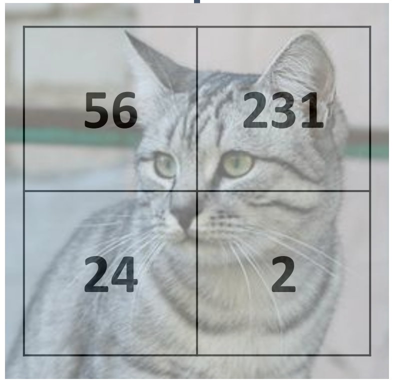
Input image
(2, 2)

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

Linear Classifier: Algebraic Viewpoint

Stretch pixels into column

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



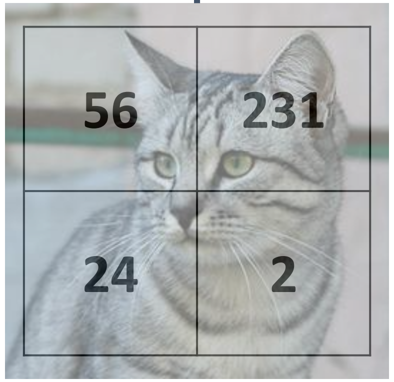
Input image
(2, 2)

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

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)

$$\underbrace{\left[\begin{array}{cccc|c} 0.2 & -0.5 & 0.1 & 2.0 & 1.1 \\ 1.5 & 1.3 & 2.1 & 0.0 & 3.2 \\ 0 & 0.25 & 0.2 & -0.3 & -1.2 \end{array}\right]}_{W\ (3,\,5)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \\ 1 \end{bmatrix}}_{(5,)} = \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{(3,)}$$

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

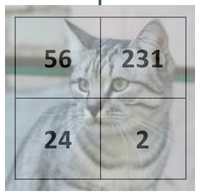
-48.4
218.9
31.0

Interpreting a Linear Classifier

Algebraic Viewpoint

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

Stretch pixels into column



Input image
(2, 2)

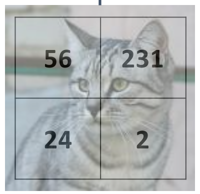
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W\ (3,\,4)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix}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

Interpreting a Linear Classifier

Algebraic Viewpoint

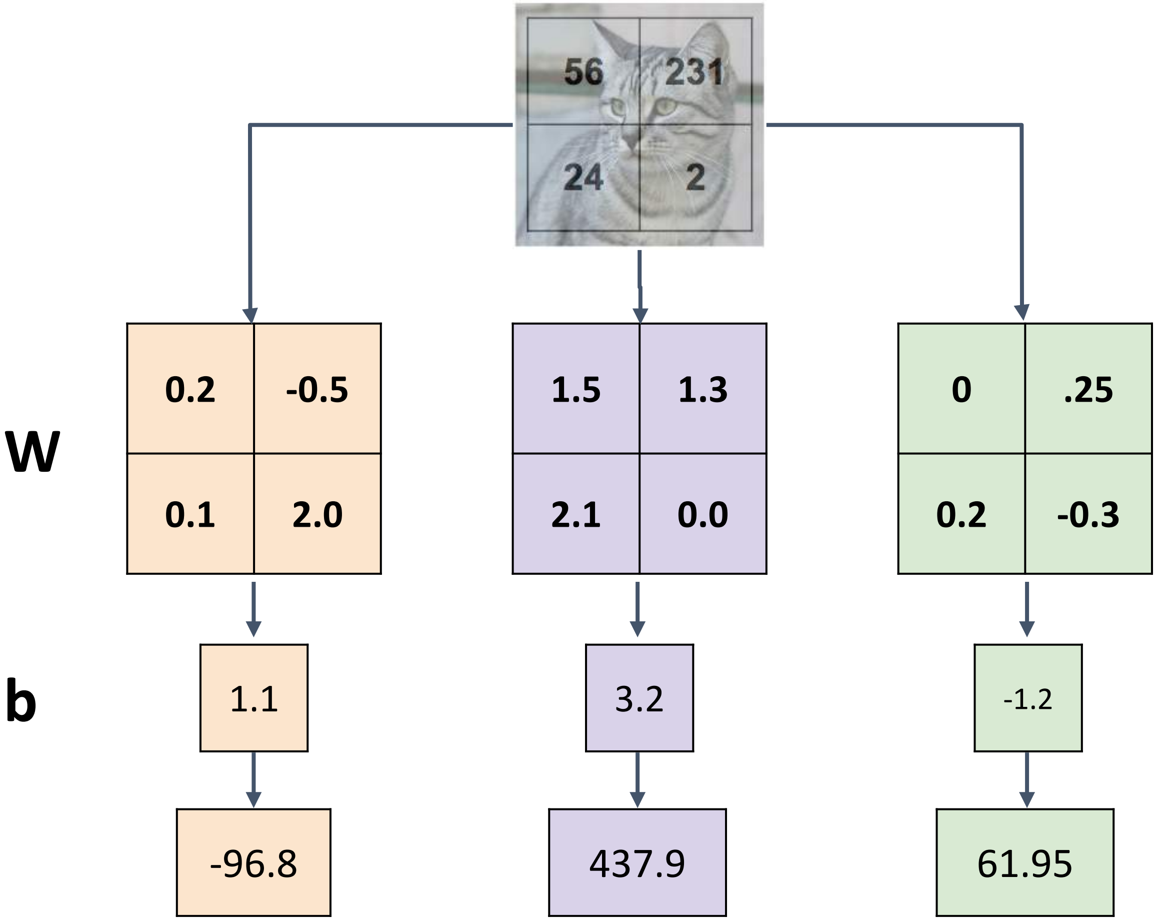
$$f(x,W) = Wx + b$$

Stretch pixels into column

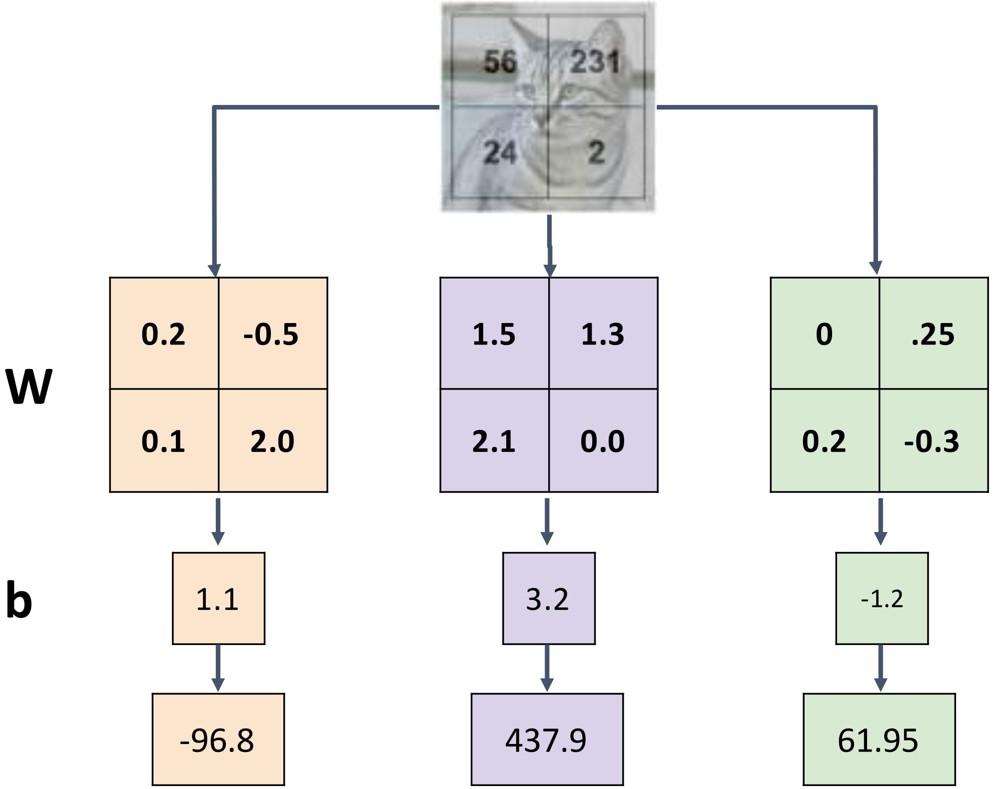


Input image
(2, 2)

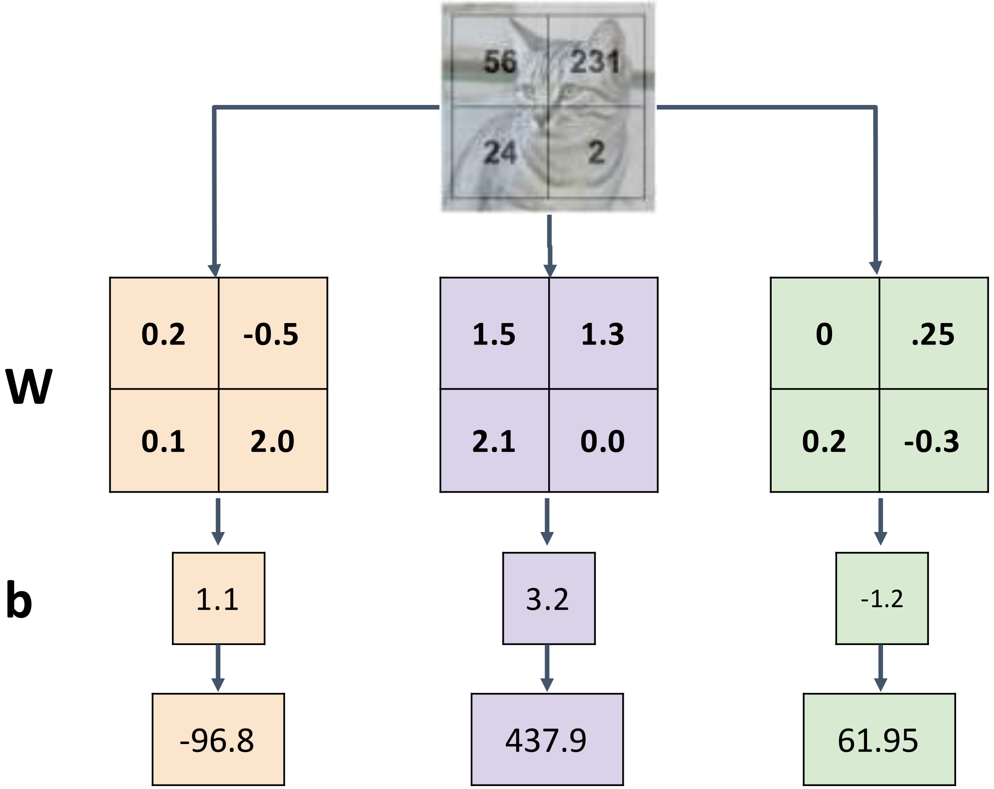
$$\underbrace{\begin{bmatrix}0.2 & -0.5 & 0.1 & 2.0\\ 1.5 & 1.3 & 2.1 & 0.0\\ 0 & 0.25 & 0.2 & -0.3\end{bmatrix}}_{W\ (3,4)} \underbrace{\begin{bmatrix}56\\ 231\\ 24\\ 2\end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix}1.1\\ 3.2\\ -1.2\end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix}-96.8\\ 437.9\\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



Interpreting an Linear Classifier

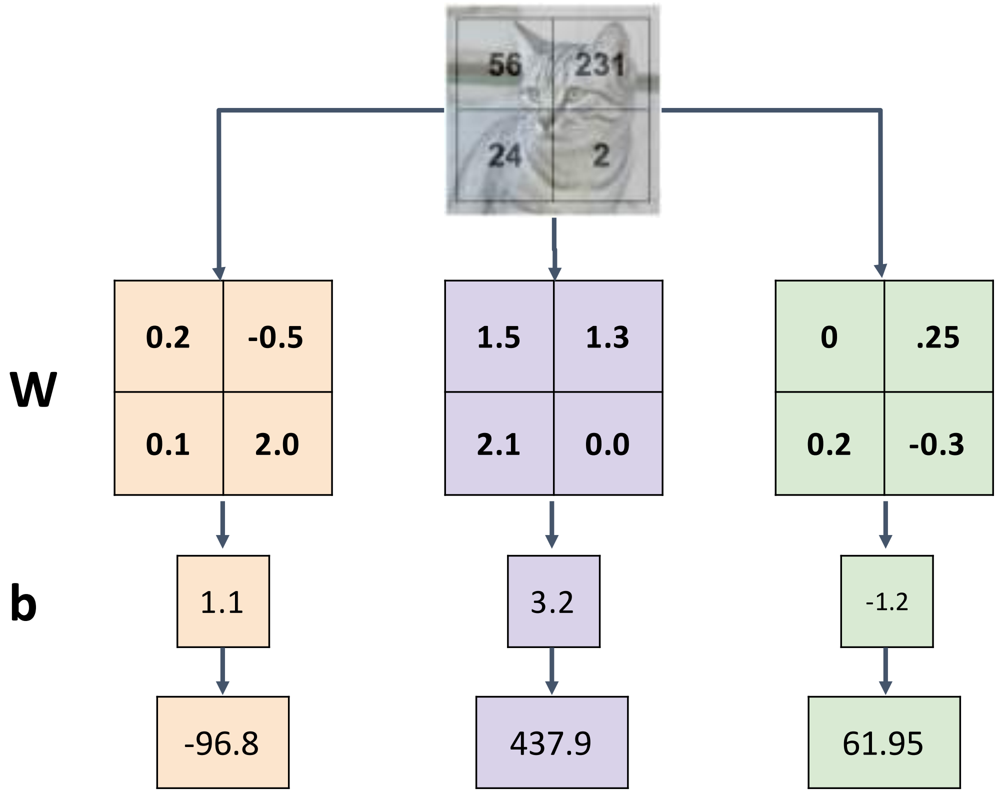


Interpreting an Linear Classifier: Visual Viewpoint



Interpreting an Linear Classifier: Visual Viewpoint

Linear classifier has one "template" per category

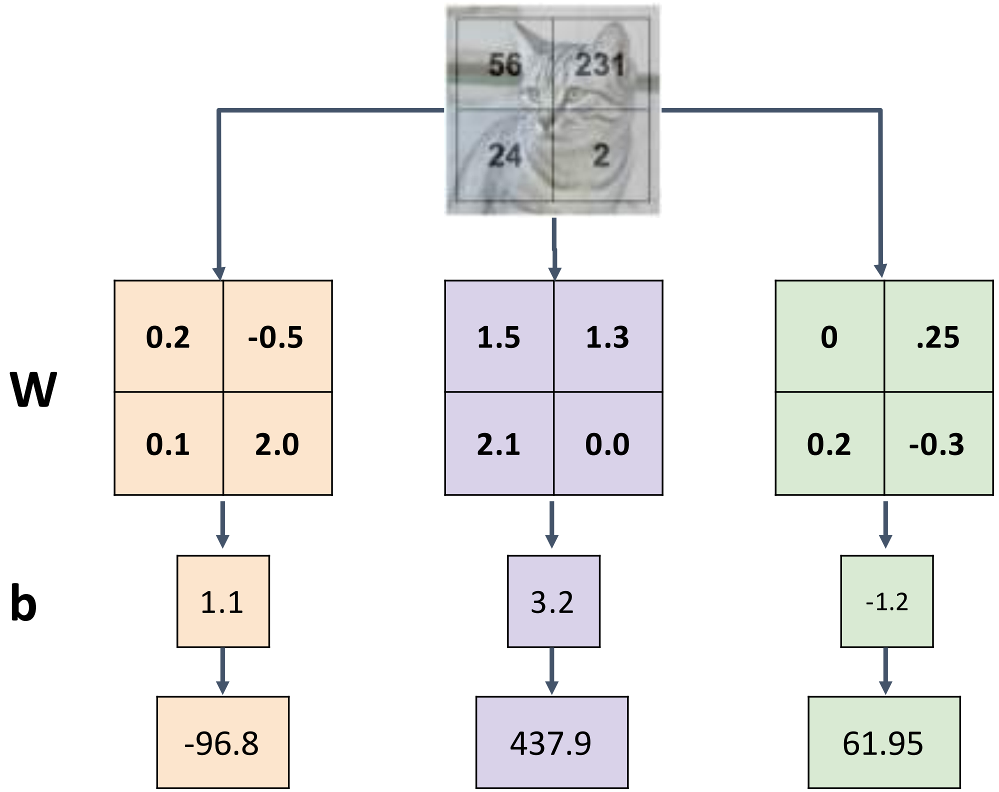


Interpreting an Linear Classifier: Visual Viewpoint

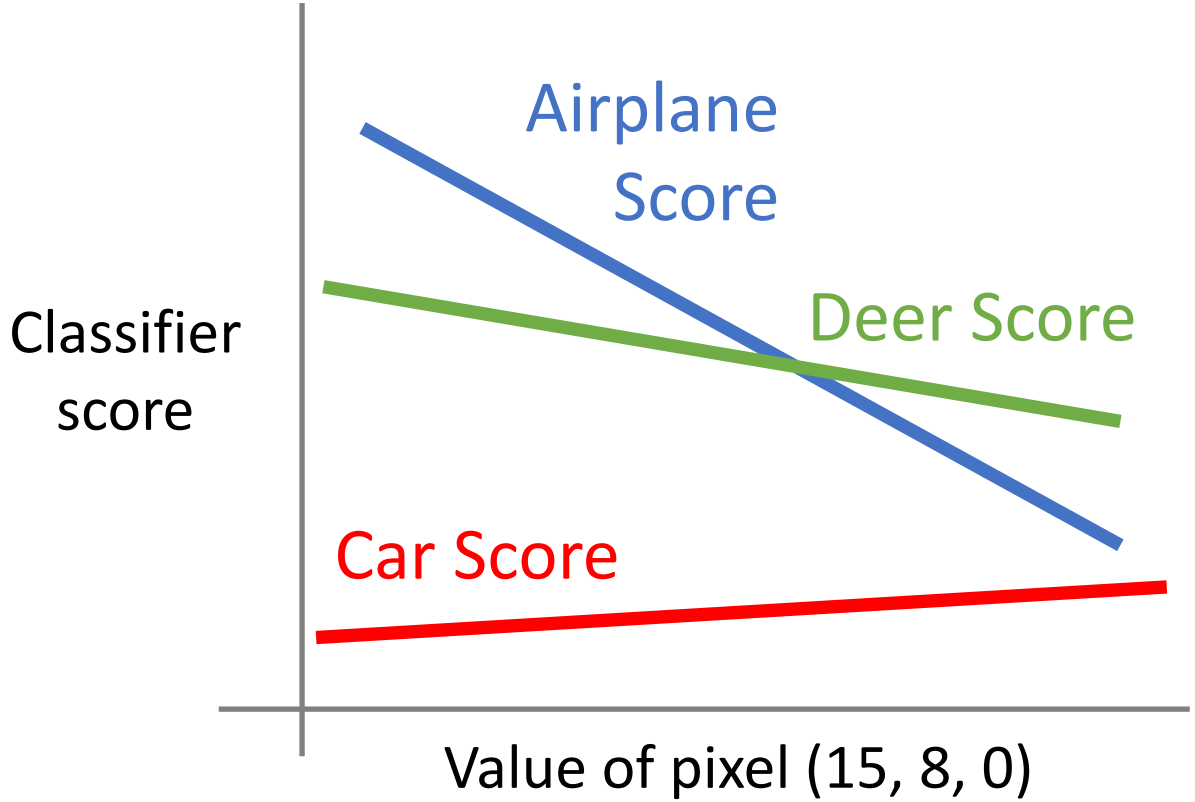
Linear classifier has one "template" per category

A single template cannot capture multiple modes of the data

e.g. horse template has 2 heads!



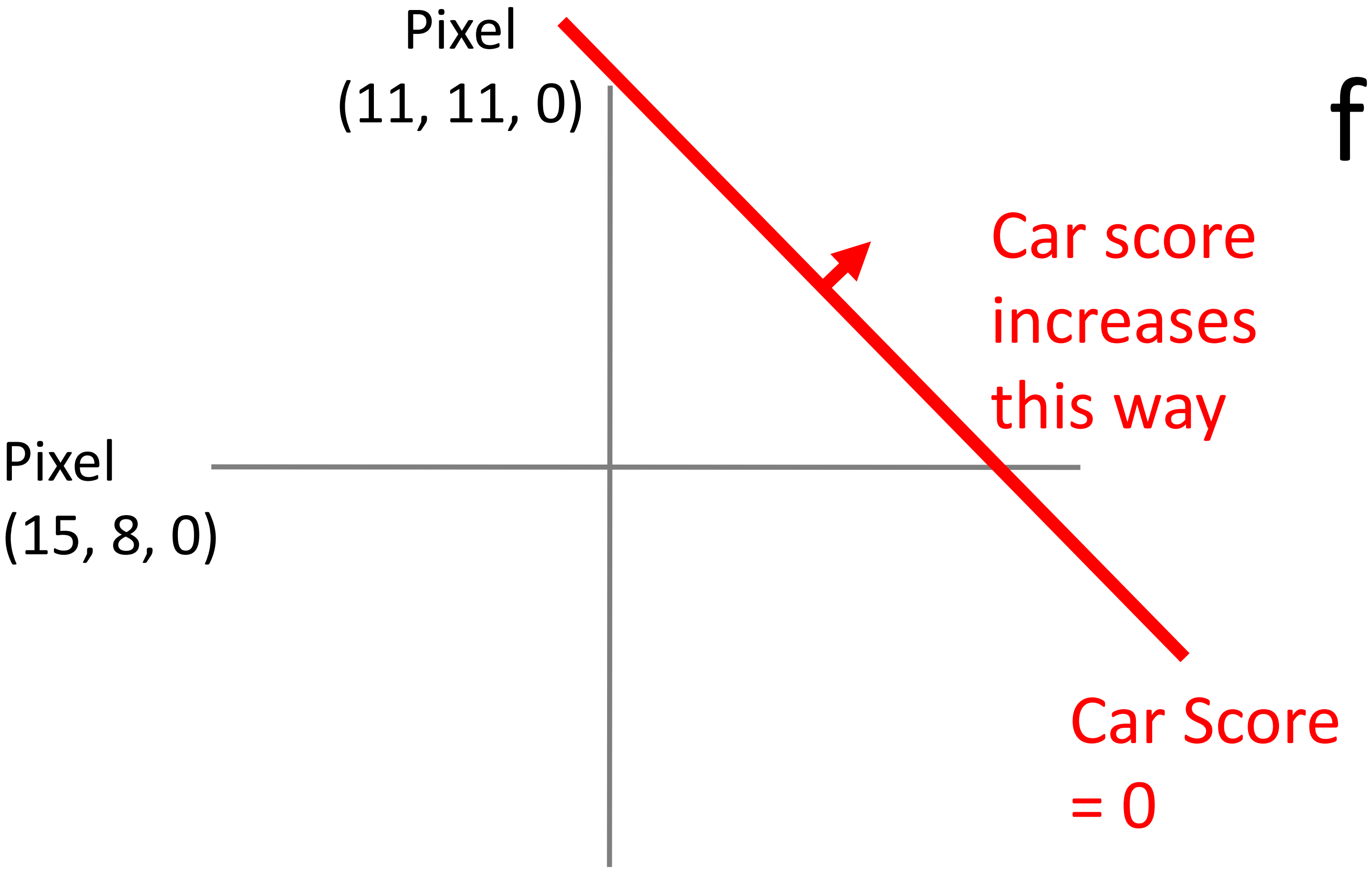
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

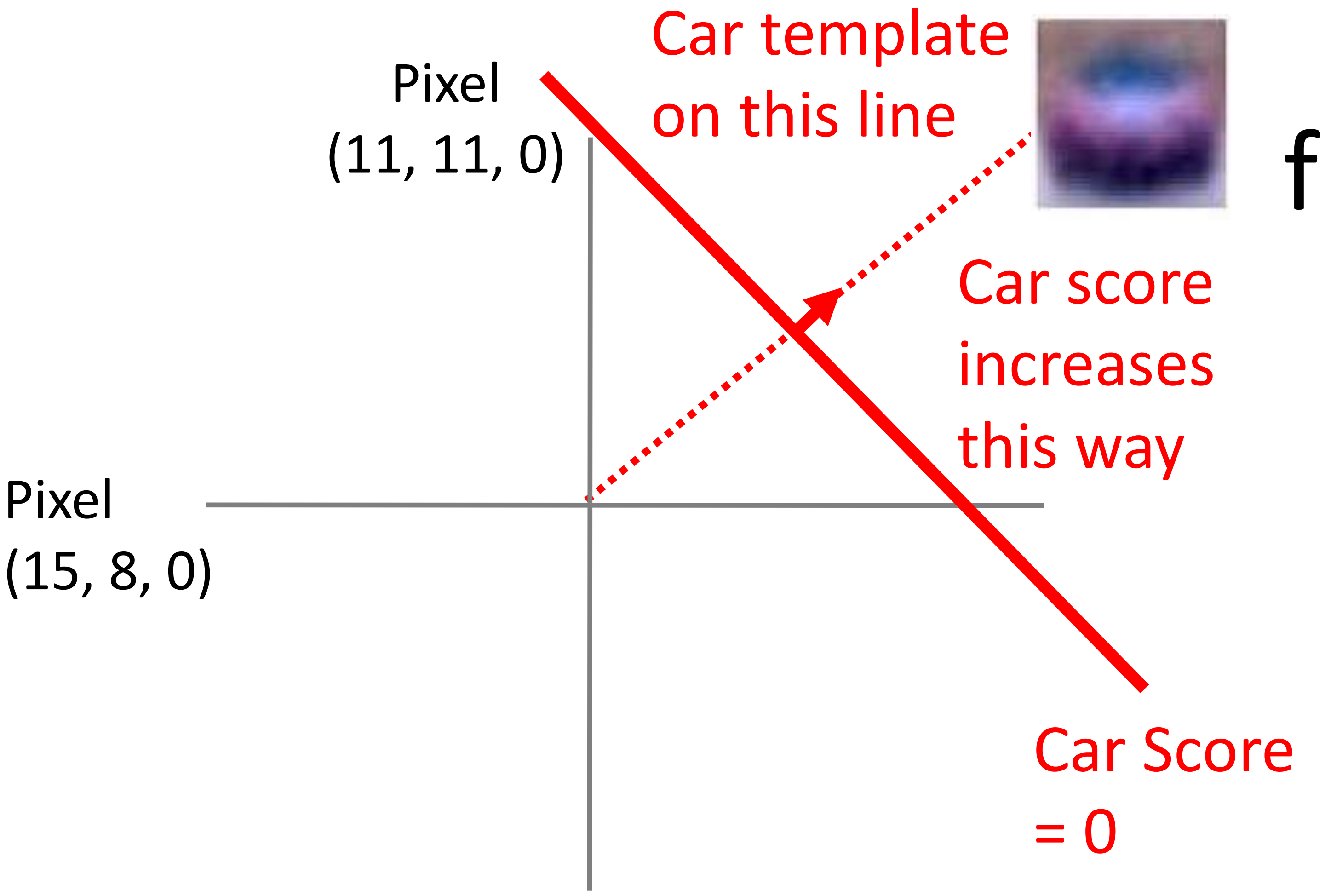
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

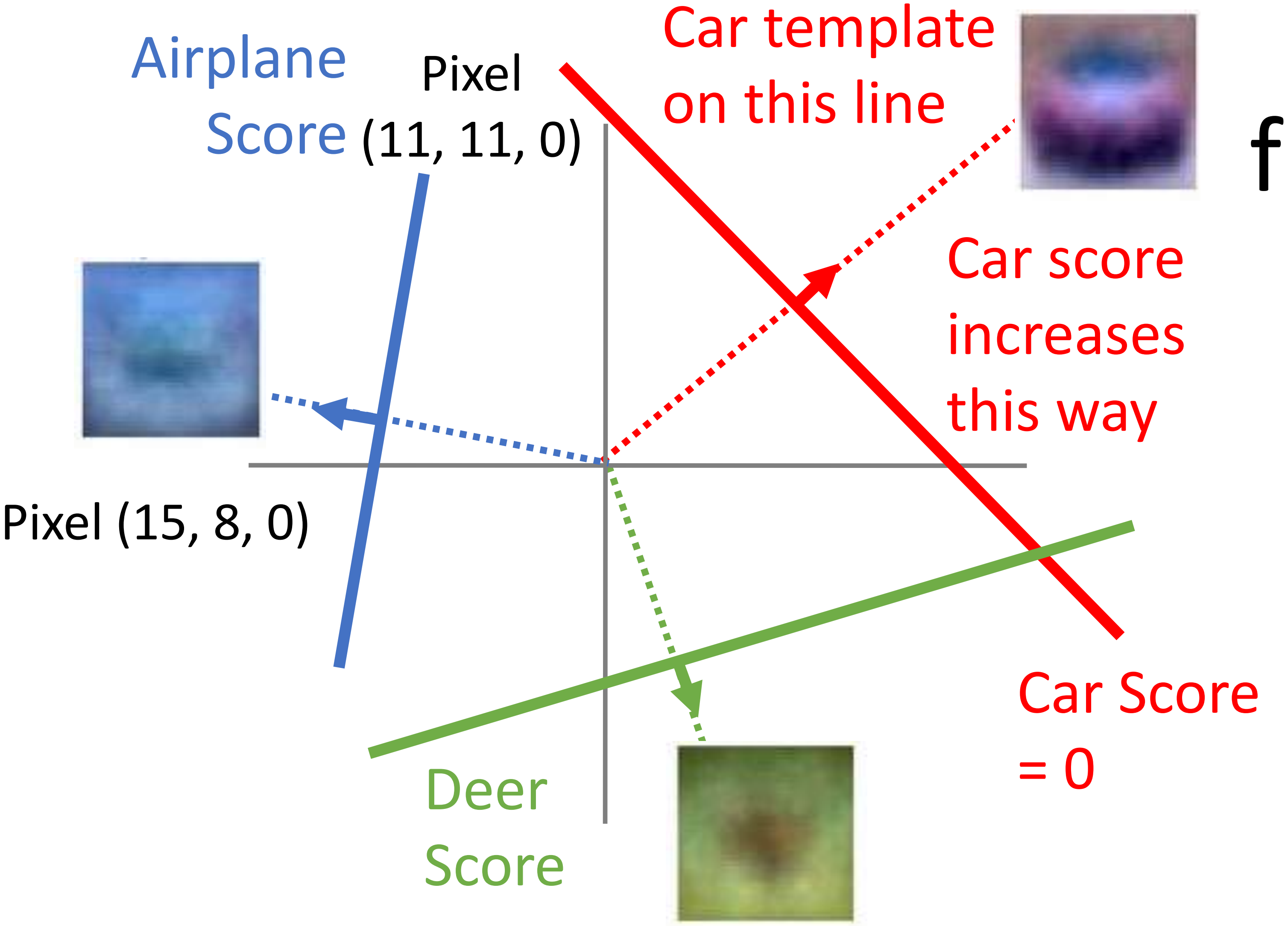
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

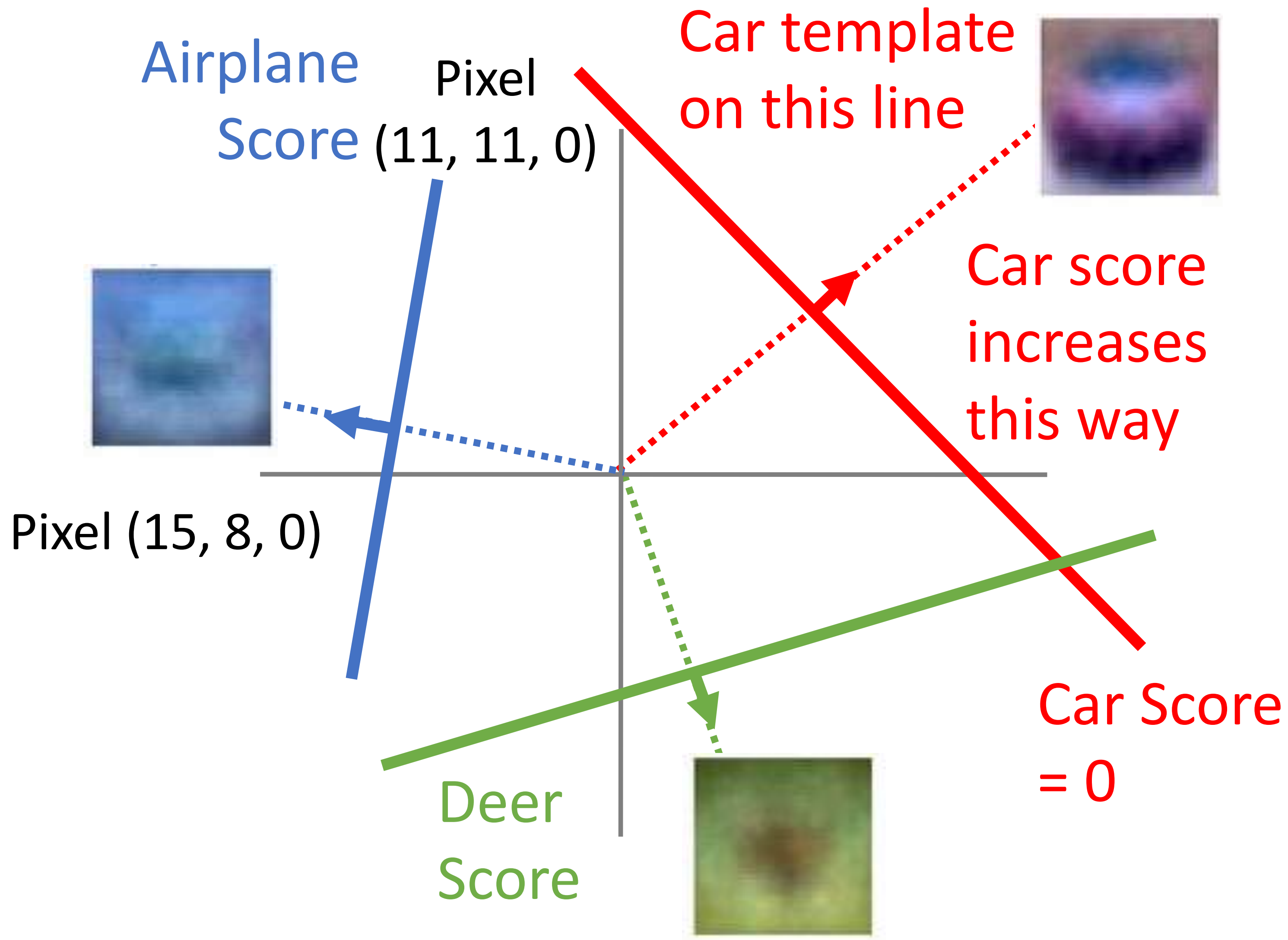
Interpreting a Linear Classifier: <u>Geometric Viewpoint</u>



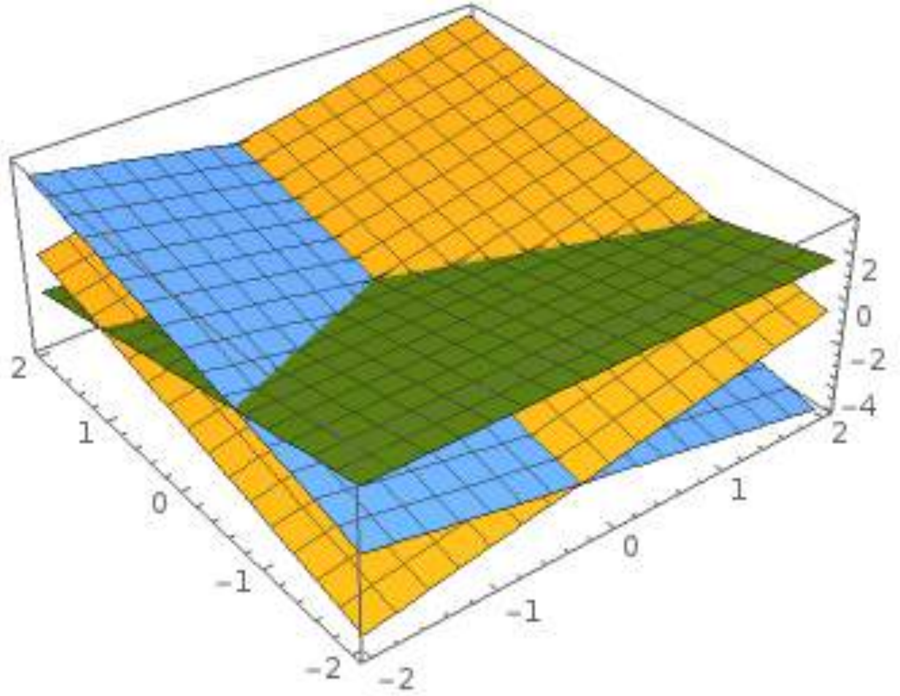
$$f(x,W) = Wx + b$$

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

Interpreting a Linear Classifier: Geometric Viewpoint



Hyperplanes carving up a high-dimensional space

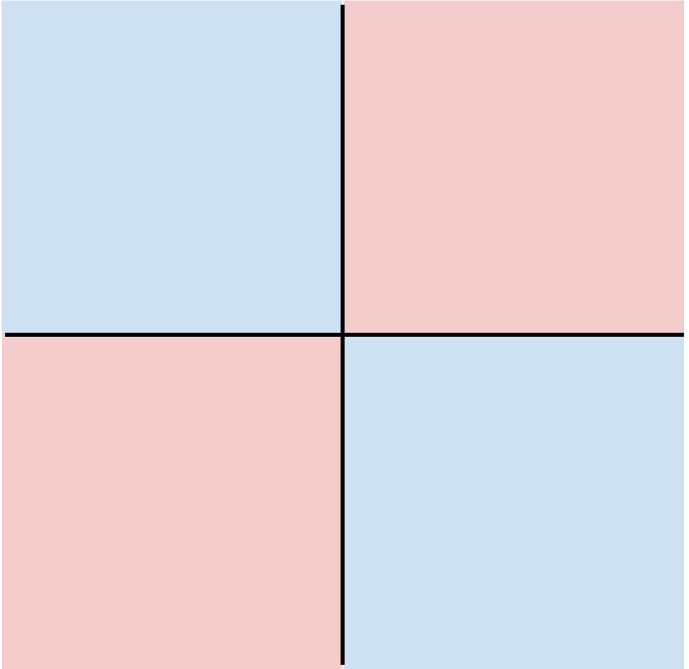


Plot created using Wolfram Cloud

Hard Cases for a Linear Classifier

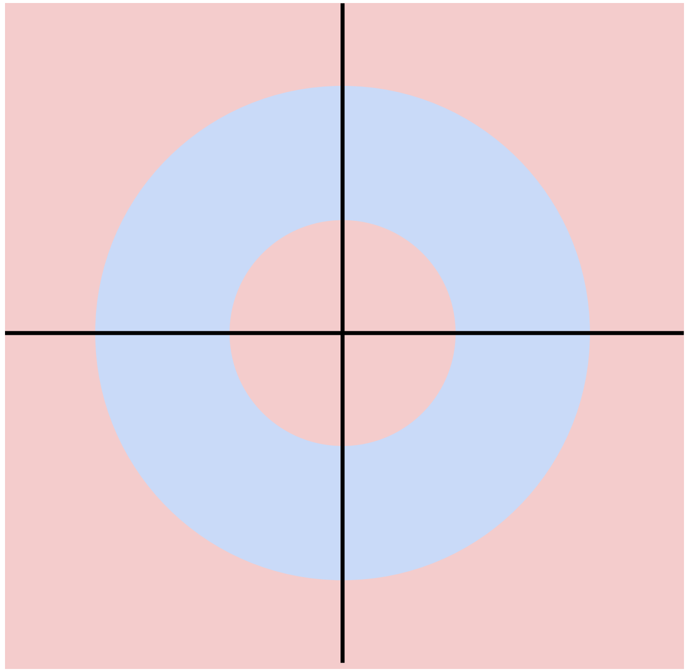
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



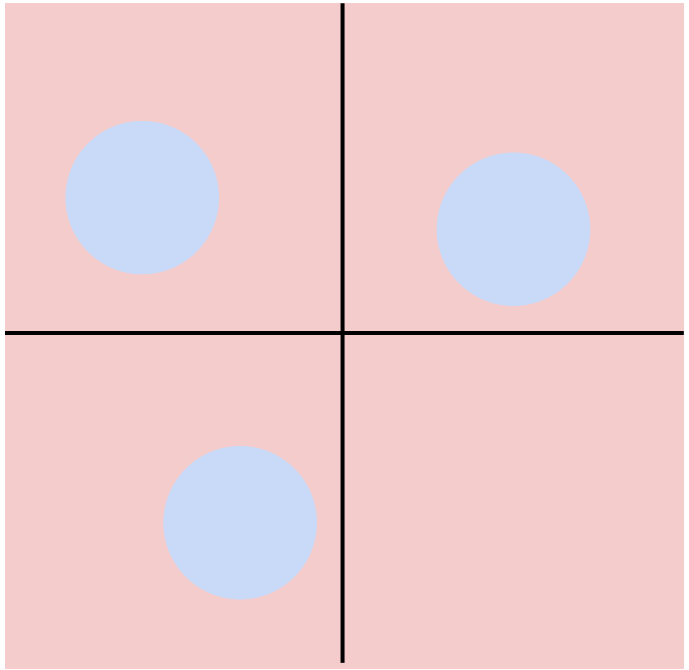
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



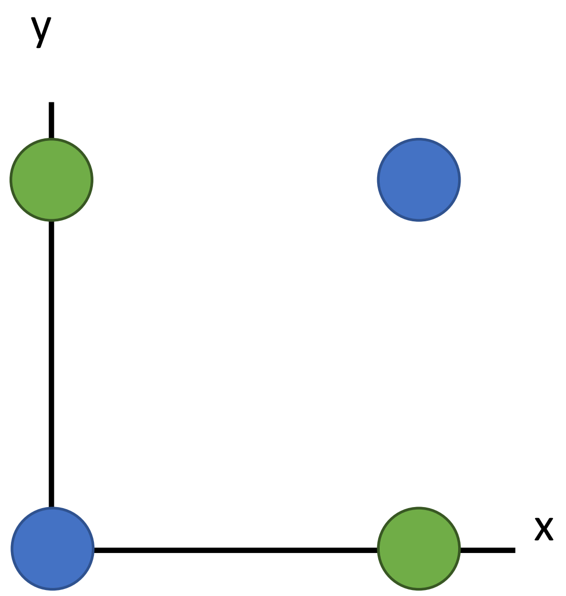
Class 1:
Three modes

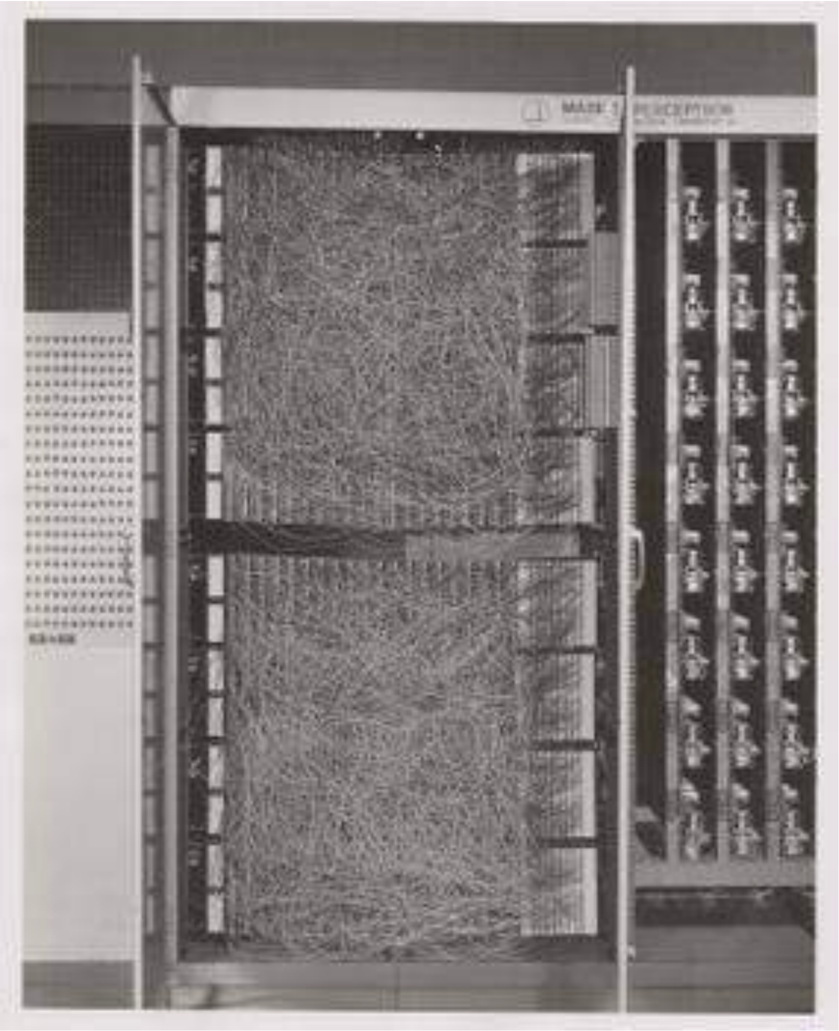
Class 2:
Everything else



Recall: Perceptron couldn't learn XOR

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

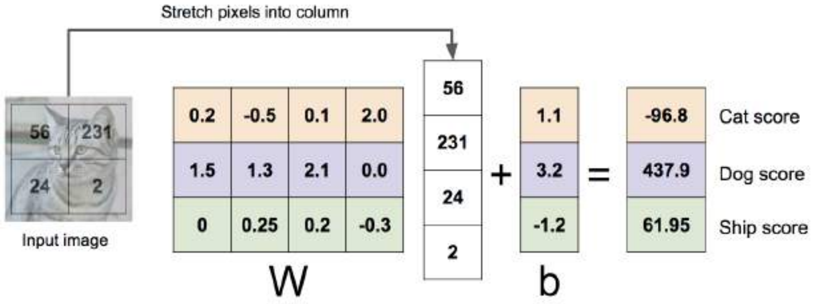




Linear Classifier: Three Viewpoints

Algebraic Viewpoint

$f(x,W) = Wx$

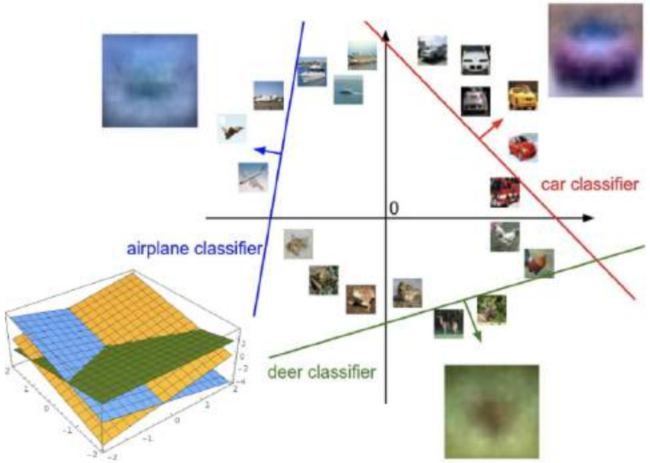


Visual Viewpoint

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

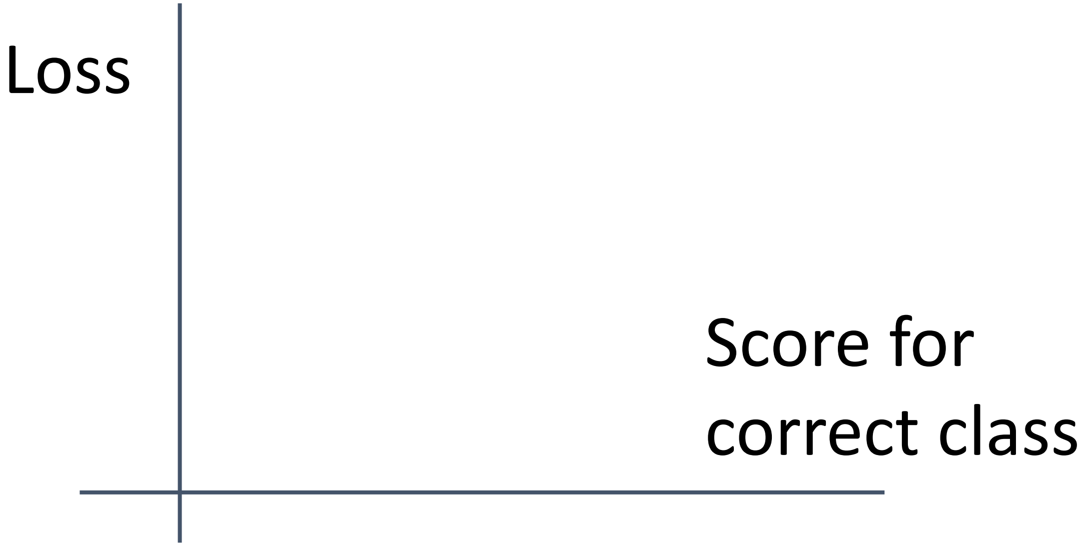
$$L_i(f(x_i, W), y_i)$$

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

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

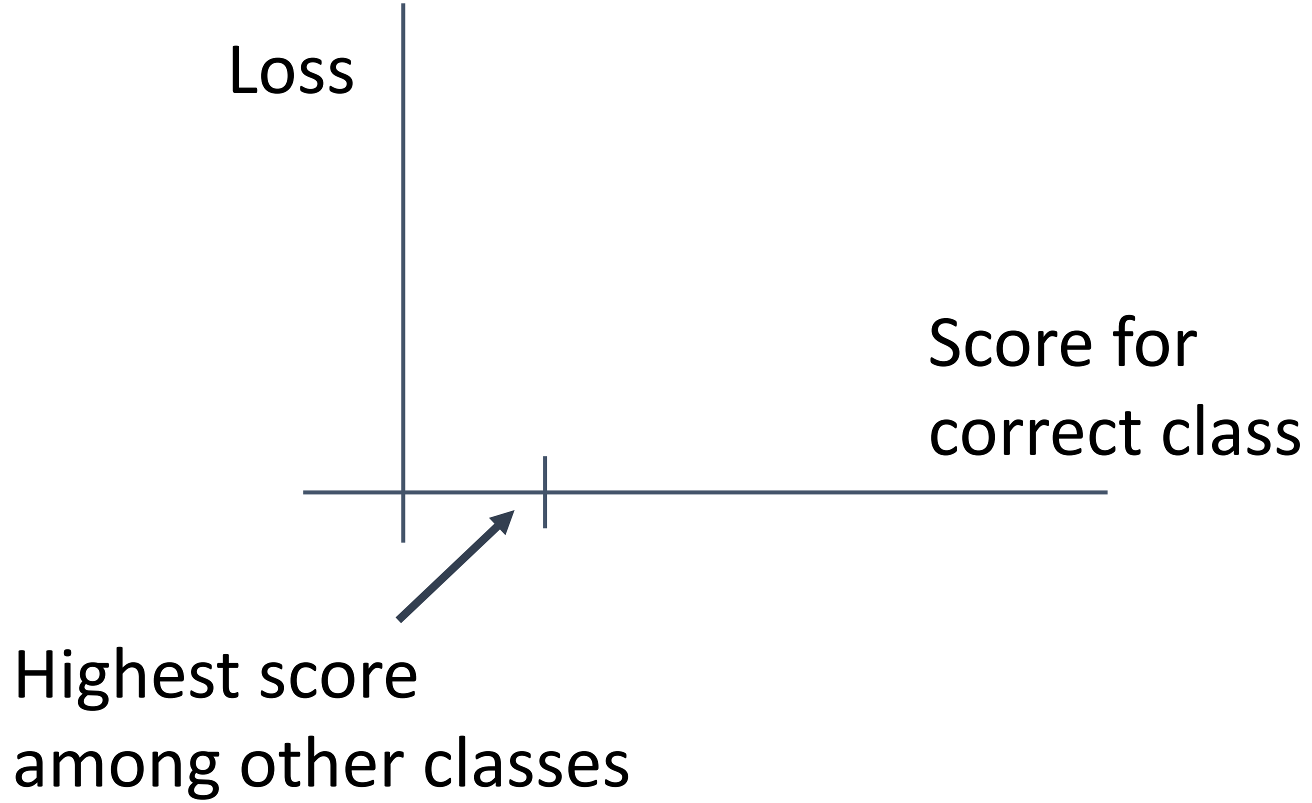
Multiclass SVM Loss

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



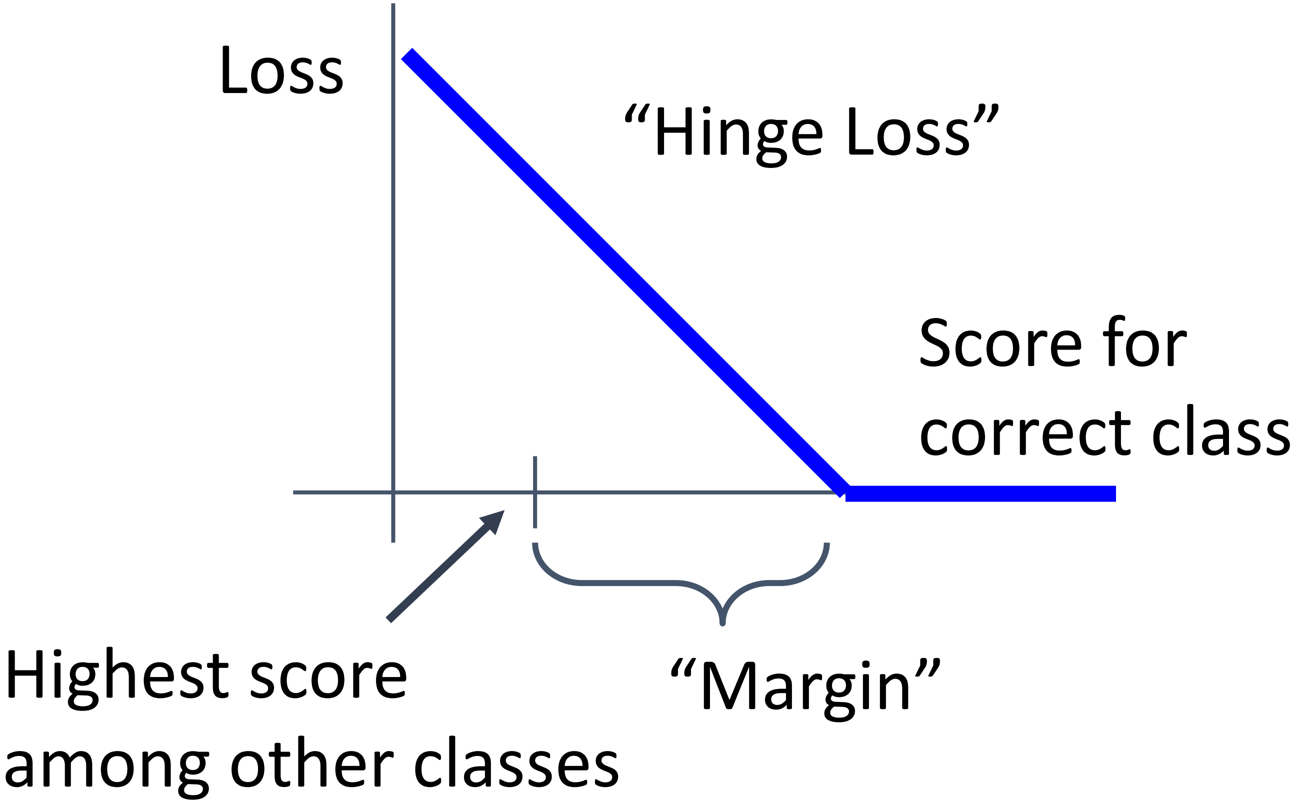
Multiclass SVM Loss

”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

w^2

$2w$

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

64

8

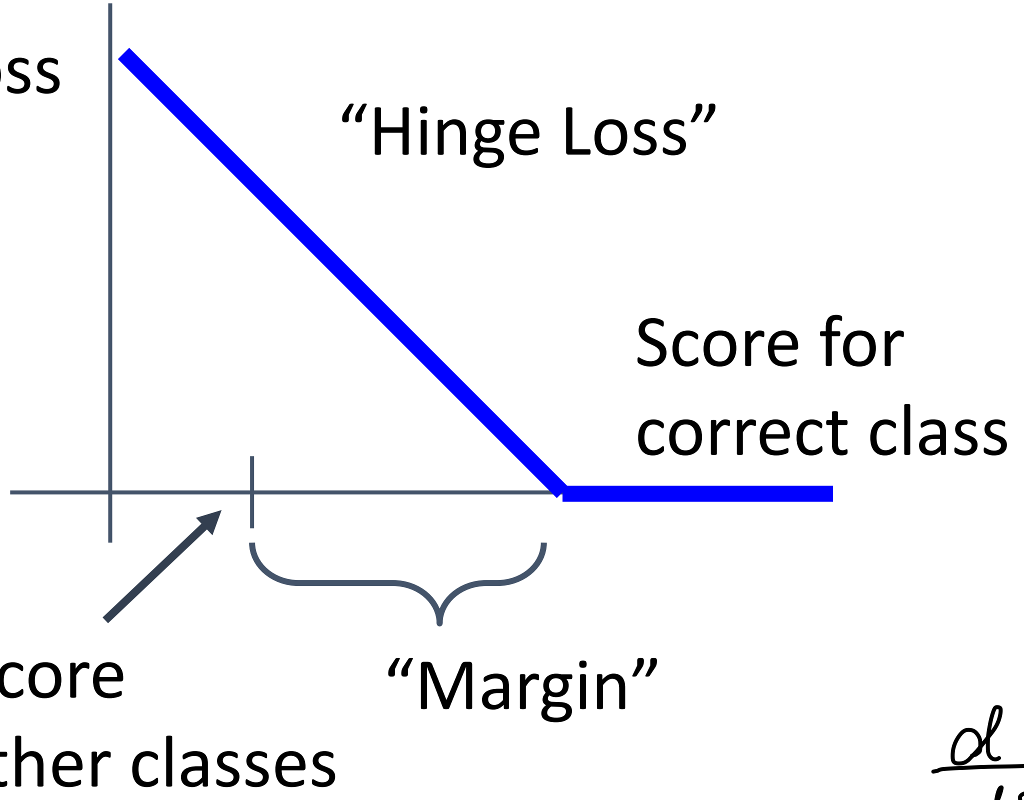
32

56

48

49

32



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:

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

$$\boxed{S_j - S_{y_i} + 1}$$

$$w_j \cdot x - w_{y_i} x + 1$$

$$\frac{dL_i^j}{dw_j} = x \qquad \frac{dL_i^j}{dw_{y_i}} = -x$$

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:

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

$N \times C$

$\frac{dL_i^j}{dw_j} = x$

$W : D \times C$

$\overset{D \times N}{X} \cdot \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$ 1 3 -2

x_1 x_1 0 $-x$ 0
x_2 x_2

Multiclass SVM Loss

$N \times C$

1
0
-1

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	$\boxed{2.2}$
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?

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

Multiclass SVM Loss

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

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Q: Suppose we found some W with L = 0. Is it unique?

Multiclass SVM Loss

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

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Q: Suppose we found some W with L = 0. Is it unique?

No! 2W is also has L = 0!

Multiclass SVM Loss

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

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

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

Original W:

= 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

Using 2W instead:

= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0
= 0

Multiclass SVM Loss

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

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



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

How should we choose between W and 2W if they both perform the same on the training data?

Regularization: Beyond Training Error

$$\underbrace{L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

Regularization: Beyond Training Error

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{} + \underbrace{\lambda R(W)}_{}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization: Beyond Training Error

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

λ = regularization strength (hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization: Beyond Training Error

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

λ = regularization strength (hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Cutout, Mixup, Stochastic depth, etc…

Regularization: Beyond Training Error

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

λ = regularization strength (hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Purpose of Regularization:

- Express preferences in among models beyond ”minimize training error”
- Avoid **overfitting**: Prefer simple models that generalize better
- Improve optimization by adding curvature

Regularization: Expressing Preferences

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

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

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

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

L2 Regularization

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

Regularization: Expressing Preferences

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

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

$$w_2 = \boxed{[0.25, 0.25, 0.25, 0.25]}$$

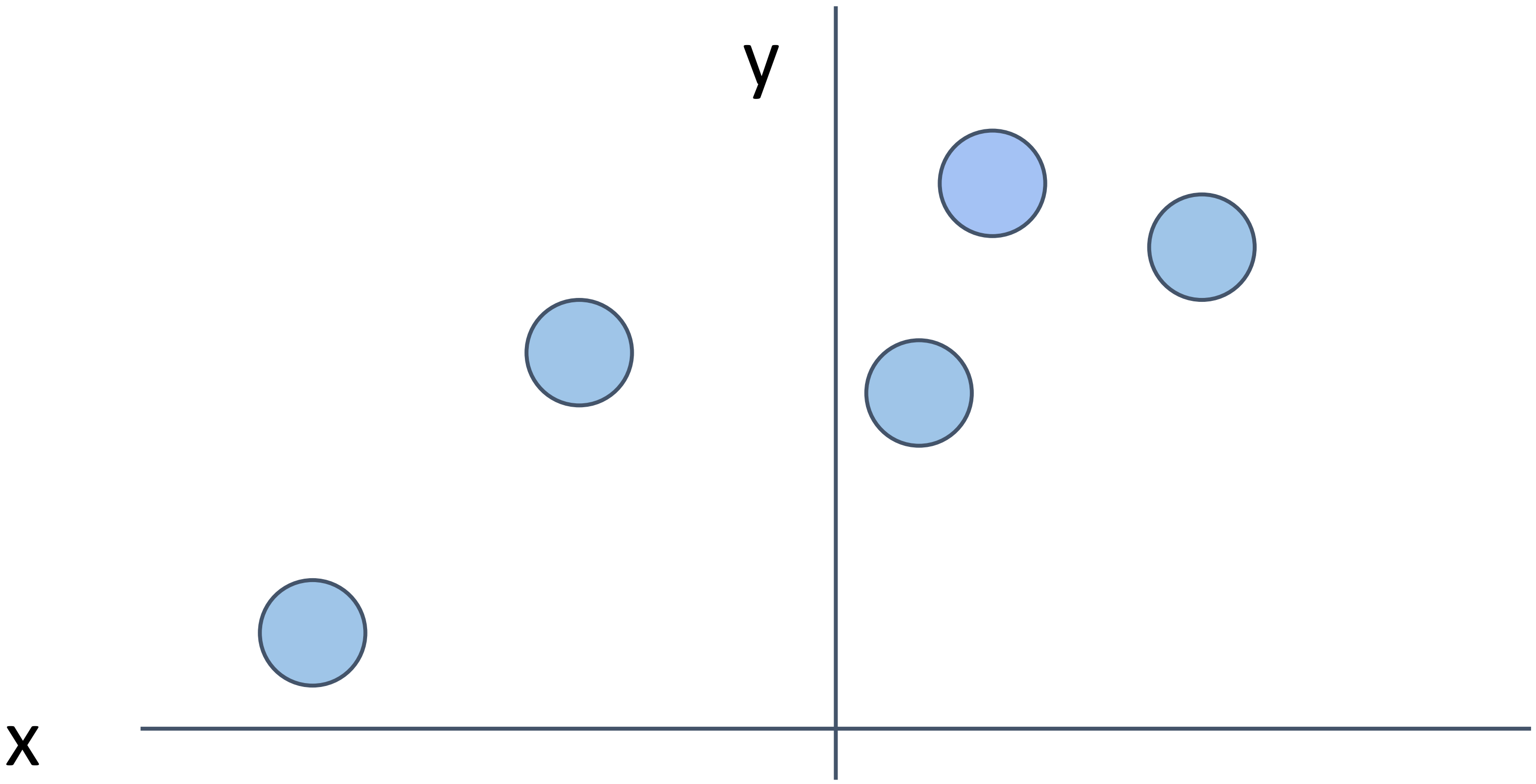
$$w_1^T x = w_2^T x = 1$$

L2 Regularization

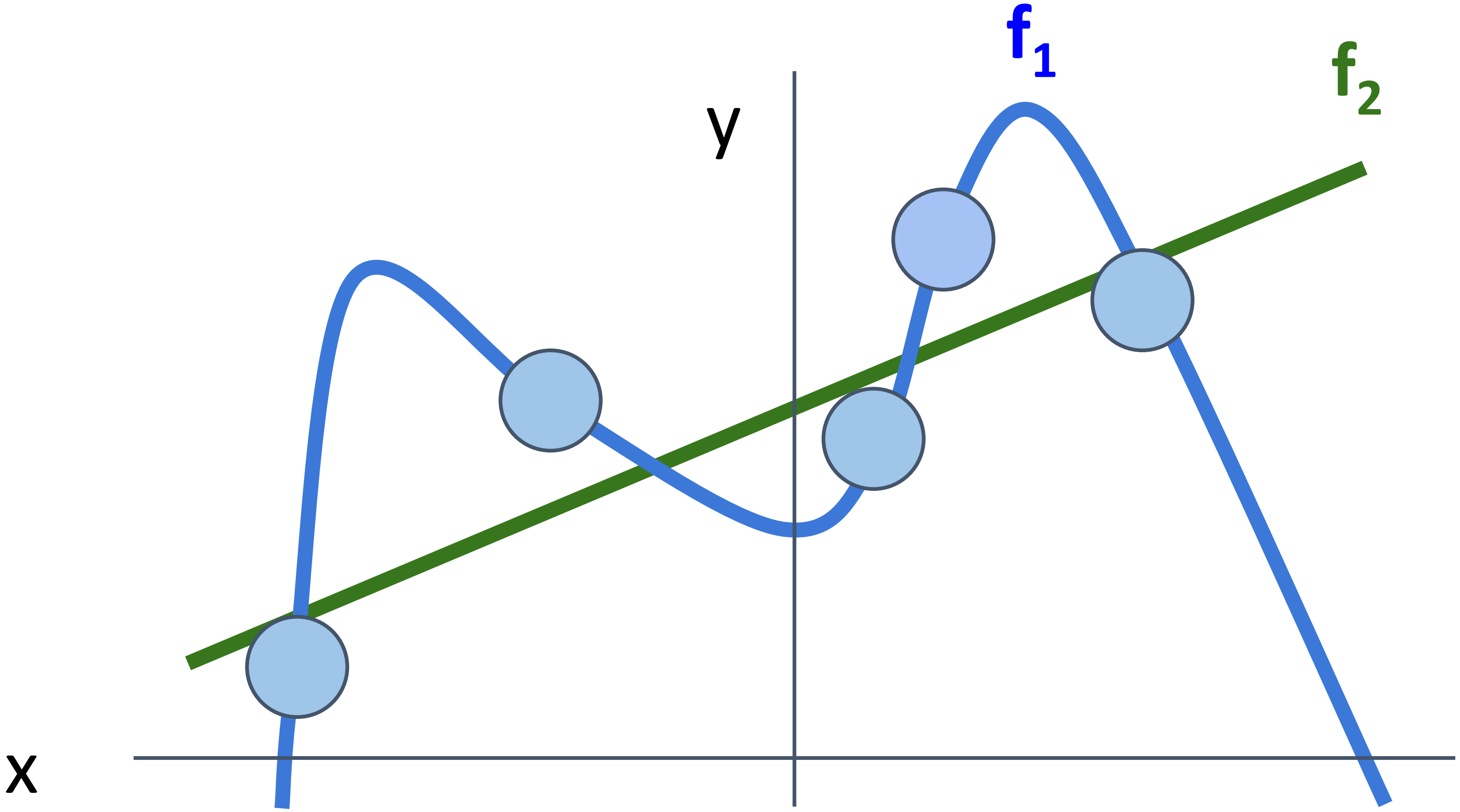
$$R(W) = \sum_k \sum_l W_{k,l}^2$$

L2 regularization likes to "spread out" the weights

Regularization: Prefer Simpler Models

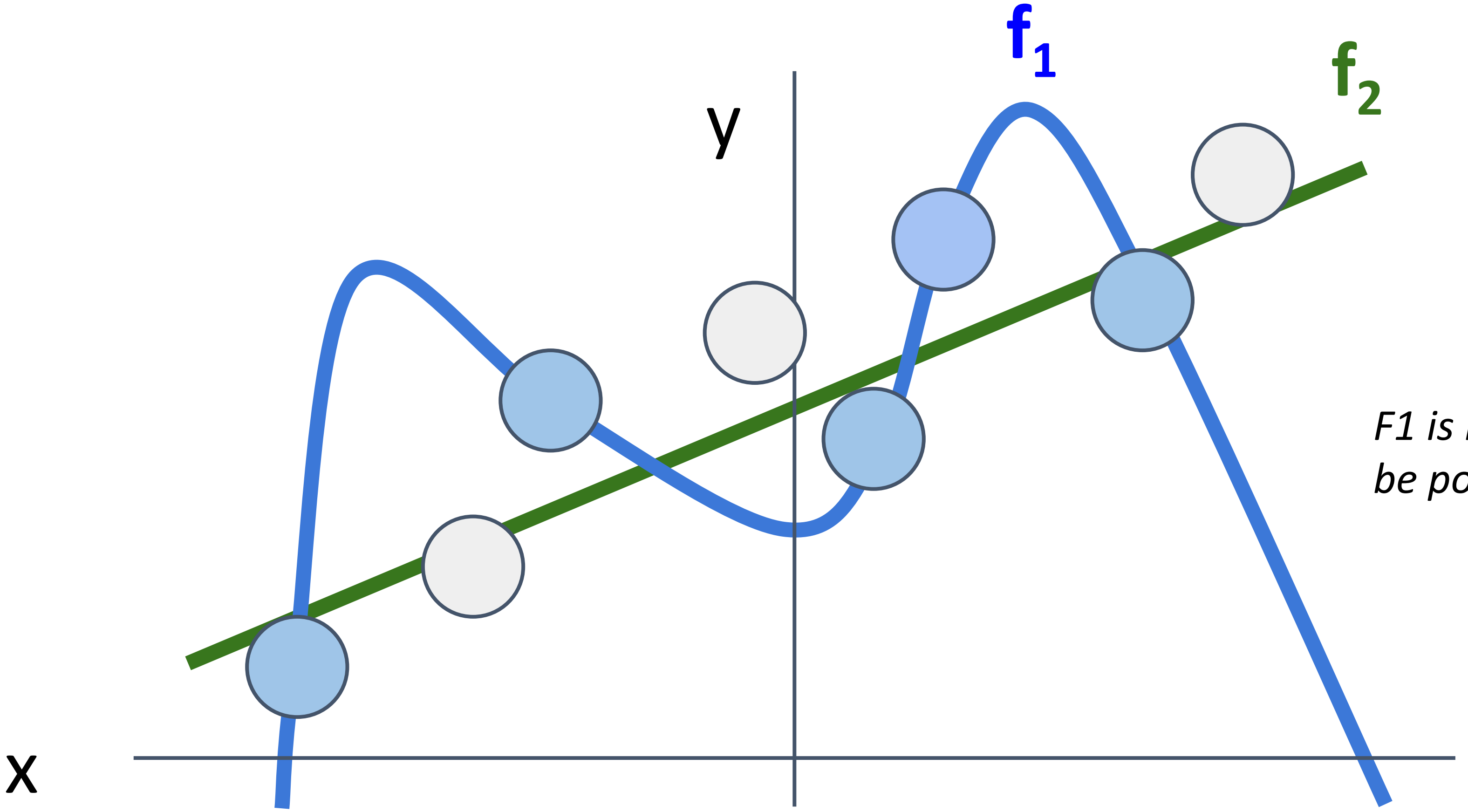


Regularization: Prefer Simpler Models



The model f_1 fits the training data perfectly
The model f_2 has training error, but is simpler

Regularization: Prefer Simpler Models

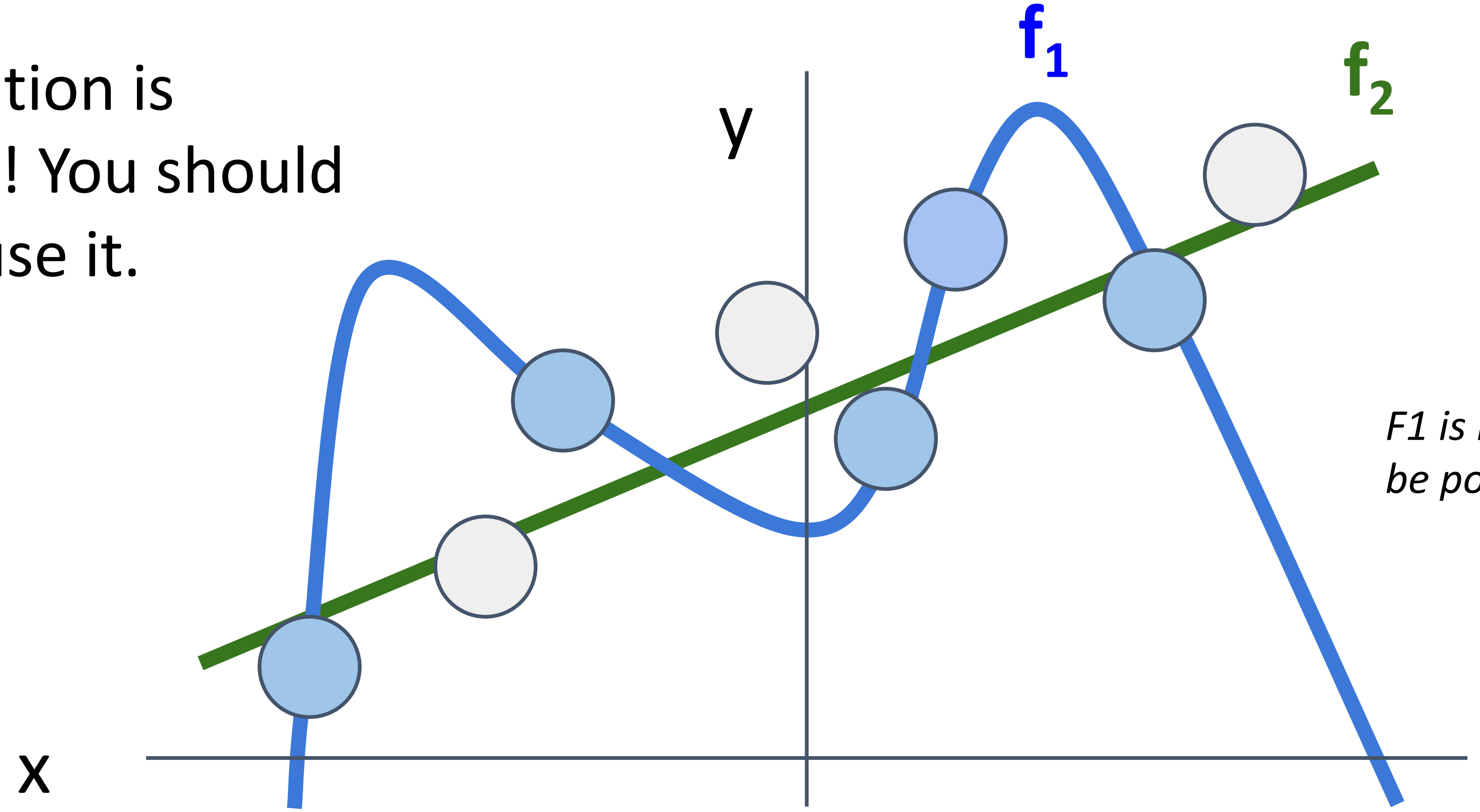


F1 is not a linear model; could be polynomial regression, etc

Regularization pushes against fitting the data *too* well so we don't fit noise in the data

Regularization: Prefer Simpler Models

Regularization is important! You should (usually) use it.



F1 is not a linear model; could be polynomial regression, etc

Regularization pushes against fitting the data *too* well so we don't fit noise in the data

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**

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

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ 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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax function

Probabilities must be >= 0

	Unnormalized log-probabilities / logits		unnormalized probabilities
cat	**3.2**	exp →	**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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax function

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp $\rightarrow$	unnormalized probabilities	normalize $\rightarrow$	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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax function

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

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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax function

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

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

$L_i = -\log(0.13)$

$= \mathbf{2.04}$

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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax function

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

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities	→ Compare ←	Correct probs
cat	**3.2**		**24.5**		**0.13**		**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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax function

Probabilities must be >= 0

Probabilities must sum to 1

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

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities	→ Compare ←	Correct probs
cat	**3.2**		**24.5**		**0.13**		**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**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax function

Probabilities must be >= 0

Probabilities must sum to 1

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

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities		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)

cat	**3.2**
car	5.1
frog	-1.7

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

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

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax function

Maximize probability of correct class

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

Putting it all together:

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

Cross-Entropy Loss (Multinomial Logistic Regression)

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

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

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Maximize probability of correct class

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

Putting it all together:

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

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

Cross-Entropy Loss (Multinomial Logistic Regression)

cat	**3.2**
car	5.1
frog	-1.7

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax function

Maximize probability of correct class

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

Putting it all together:

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

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

A: Min 0, max +infinity

Cross-Entropy Loss (Multinomial Logistic Regression)

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

cat	**3.2**
car	5.1
frog	-1.7

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

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax function

Maximize probability of correct class

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

Putting it all together:

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

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**

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

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Maximize probability of correct class

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

Putting it all together:

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

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

A: -log(C)

log(10) ≈ 2.3

Cross-Entropy vs SVM Loss

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

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

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

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

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

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?

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

Cross-Entropy vs SVM Loss

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

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

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

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

Cross-Entropy vs SVM Loss

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

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

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

Q: What happens to each loss if I 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

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

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

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?

Cross-Entropy vs SVM Loss

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

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

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

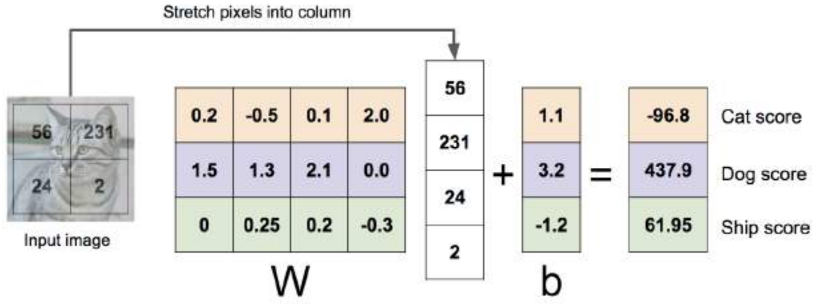
Q: What happens to each loss if I double the score of the correct class from 10 to 20?

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

Recap: Three ways to think about linear classifiers

Algebraic Viewpoint

$f(x,W) = Wx$

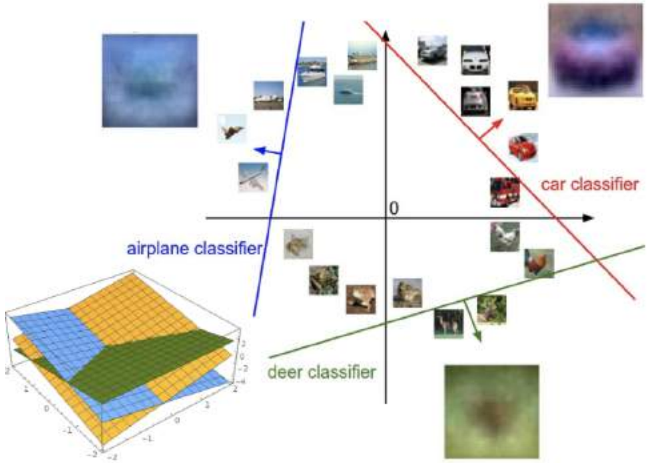


Visual Viewpoint

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**:

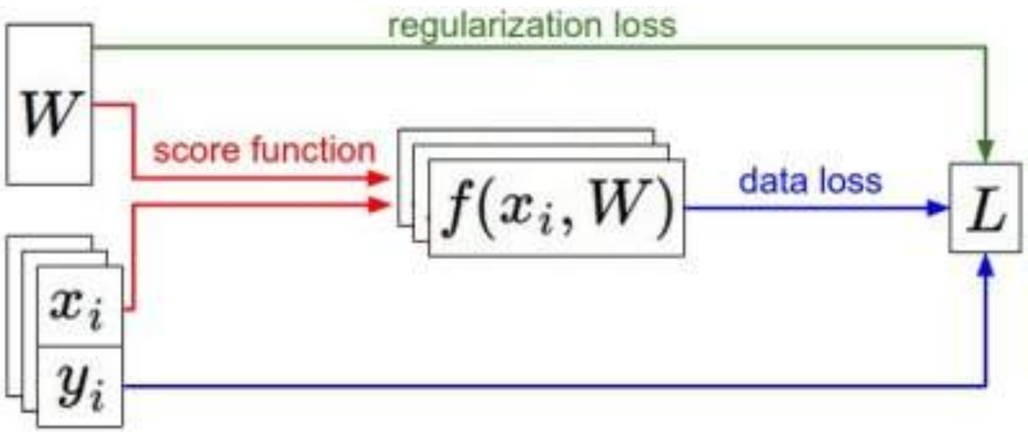
$$s = f(x; W) = Wx$$

Linear classifier

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

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

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)$$ Full loss



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?

$$s = f(x; W) = Wx$$

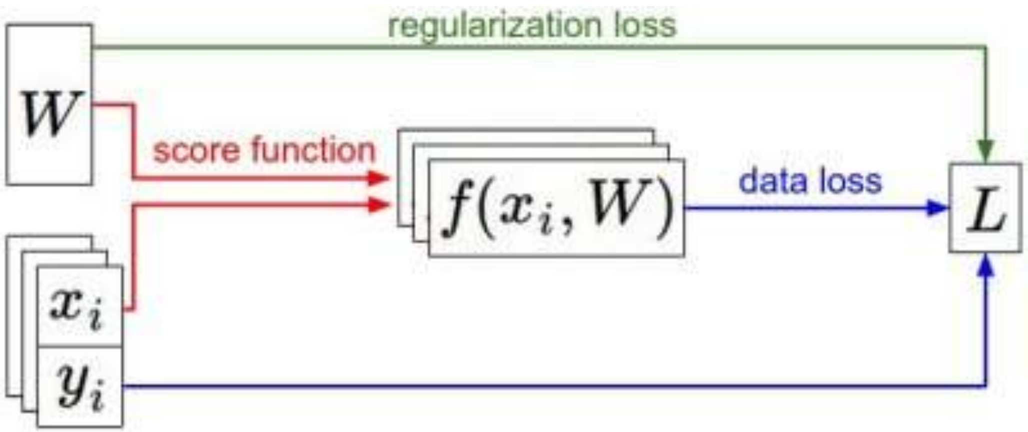
Linear classifier

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

SVM

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

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)$$ Full loss



Next time: Optimization