CS163: Deep Learning for Computer Vision

Lecture 3:
Linear Neural Networks

Courtesy of Prof. Justin Johnson at Umich and the team of Stanford CS231n for the part of the slides

Last time: Image Classification

Input: image

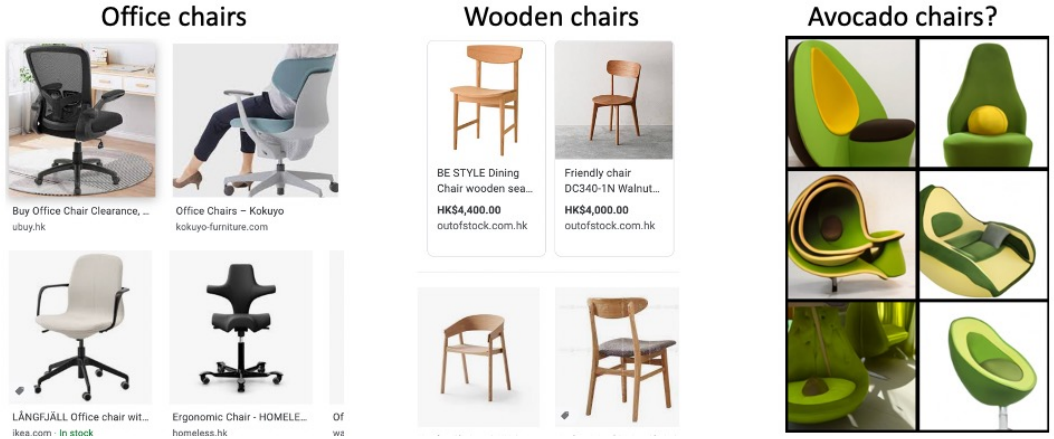
Output: Assign image to one of a fixed set of categories

chair
bed
sofa
table
cabinet

Last Time: Challenges of Visual Recognition

Viewpoint

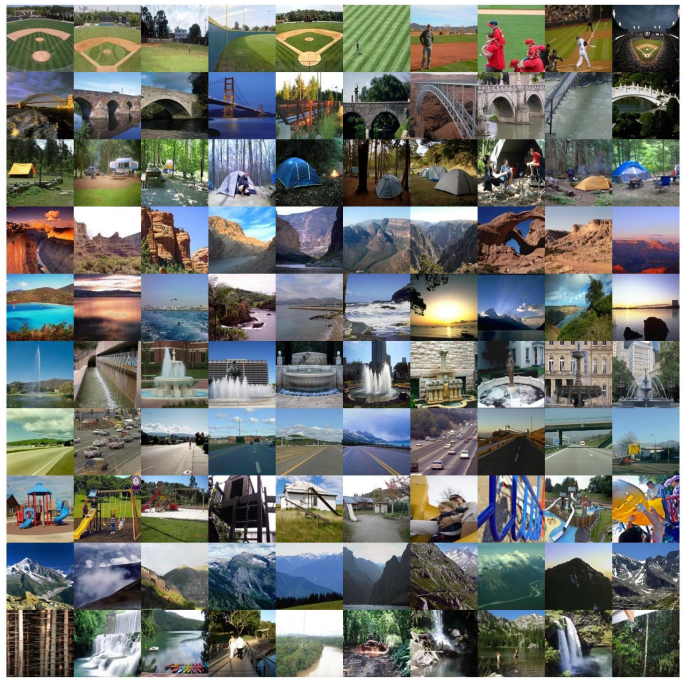
Intraclass variation and subcategories



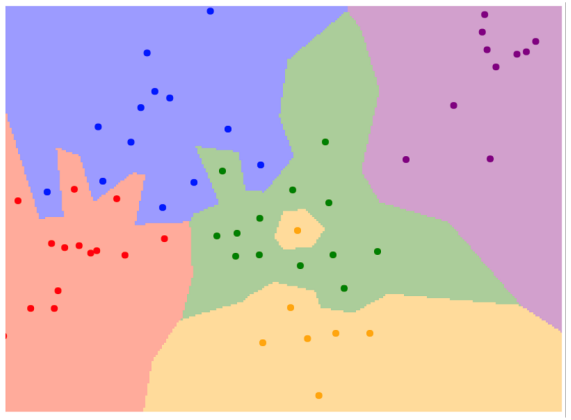
Domain changes

Occlusion & Clutter

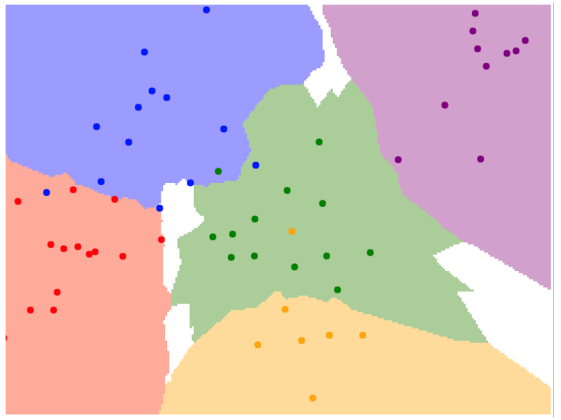
Last time: Data-Drive Approach, kNN

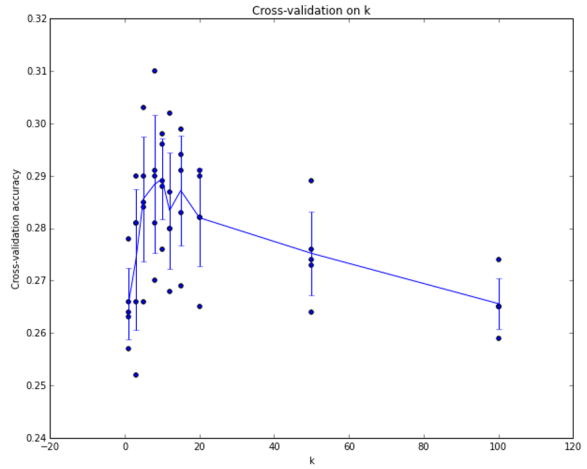


1-NN classifier



5-NN classifier





Today: Linear Neural Networks/Layers/Classifiers

Linear Layer is the building block of deep neural network

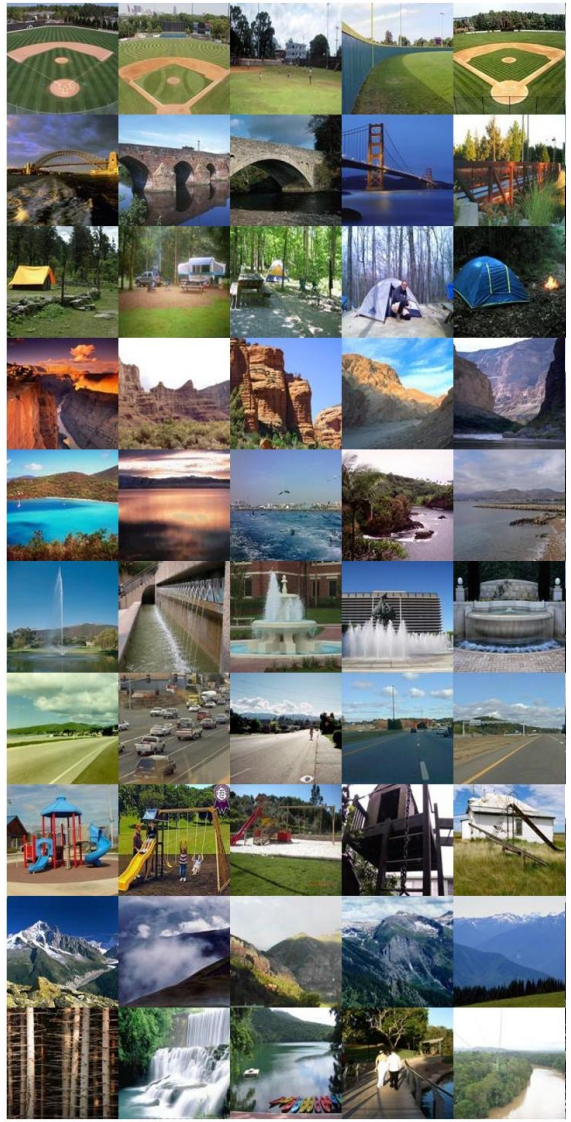
Linear layers

Linear layers

Credit: unknow

TinyPlaces: 20 scenes categories from Places

10 outdoor scenes



10 indoor scenes

20,000 training images
each image is **32x32x3**

2,000 test images.

Parametric Approach: Indoor/Outdoor Classification

Image

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

f(**x**,**W**)

1 number giving outdoor-ness score

W
parameters or weights

Parametric Approach: Scene Classification

Image

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

f(**x**,**W**)

W
parameters
or weights

20 numbers giving
class scores

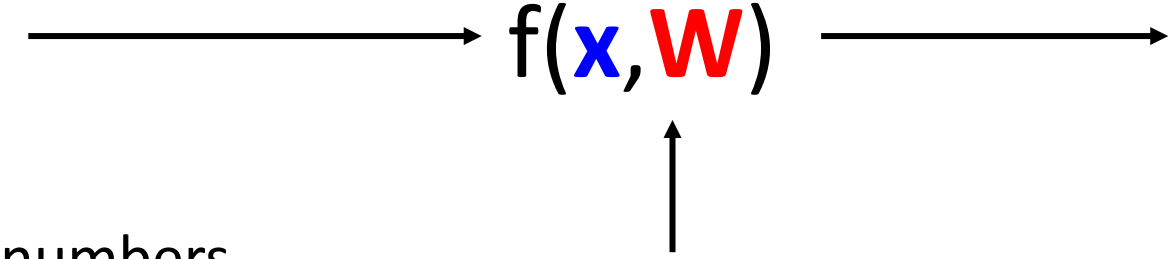
bathroom
bedroom
bookstore
classroom
dining_room
food_court
kitchen
lobby
living_room
Office
….

Parametric Approach: Linear Classifier

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

Image

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



20 numbers giving class scores

W

parameters or weights

Parametric Approach: Linear Classifier

(3072,)

$f(x,W) = Wx$

(20,) (20, 3072)

Image

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

f(**x**,**W**)

20 numbers giving class scores

W
parameters or weights

Parametric Approach: Linear Classifier

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

(3072,)

(20,) (20, 3072) (20,)

Image

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

f(**x**,**W**)

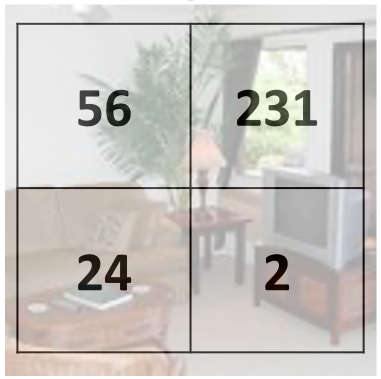
20 numbers giving class scores

W
parameters
or weights

Example for 2x2 image, outdoor-ness prediction

Stretch 4 pixels into a vector

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



Input grey image
(2, 2)

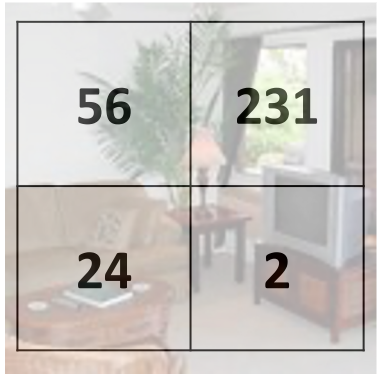
56
231
24
2

(4,)

Example for 2x2 image, outdoor-ness prediction

Stretch 4 pixels into a vector

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



Input image
(2, 2)

0.05	-0.02	0.13	-0.71

W (1, 4)

56
231
24
2

(4,)

\+ | 0.27 | = **0.15**

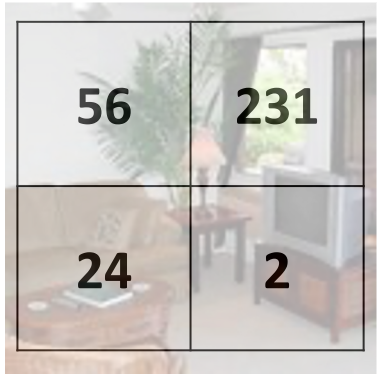
b (1,)

(1,)

Example for 2x2 image, 3 classes (livingroom/highway/mountain)

Stretch 4 pixels into a vector

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



Input image
(2, 2)

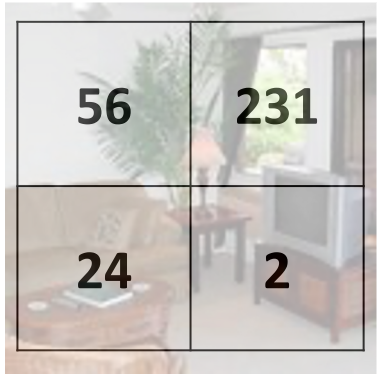
56
231
24
2

(4,)

Example for 2x2 image, 3 classes (livingroom/highway/mountain)

Stretch 4 pixels into a vector

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



Input image
(2, 2)

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W (3, 4)

56
231
24
2

(4,)

+

1.1
3.2
-1.2

b
(3,)

=

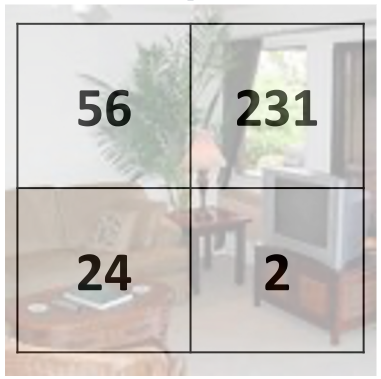
-96.8
437.9
61.95

(3,)

Linear Classifier: Linear Transformation

Stretch 4 pixels into a vector

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



Input image
(2, 2)

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W (3, 4)

56
231
24
2

(4,)

+

1.1
3.2
-1.2

b
(3,)

=

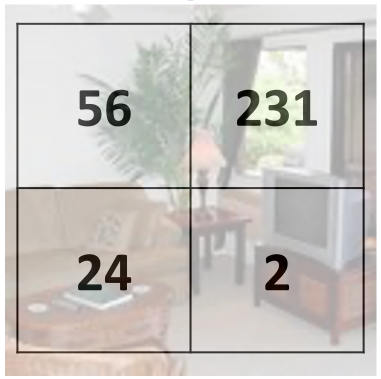
-96.8
437.9
61.95

(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!

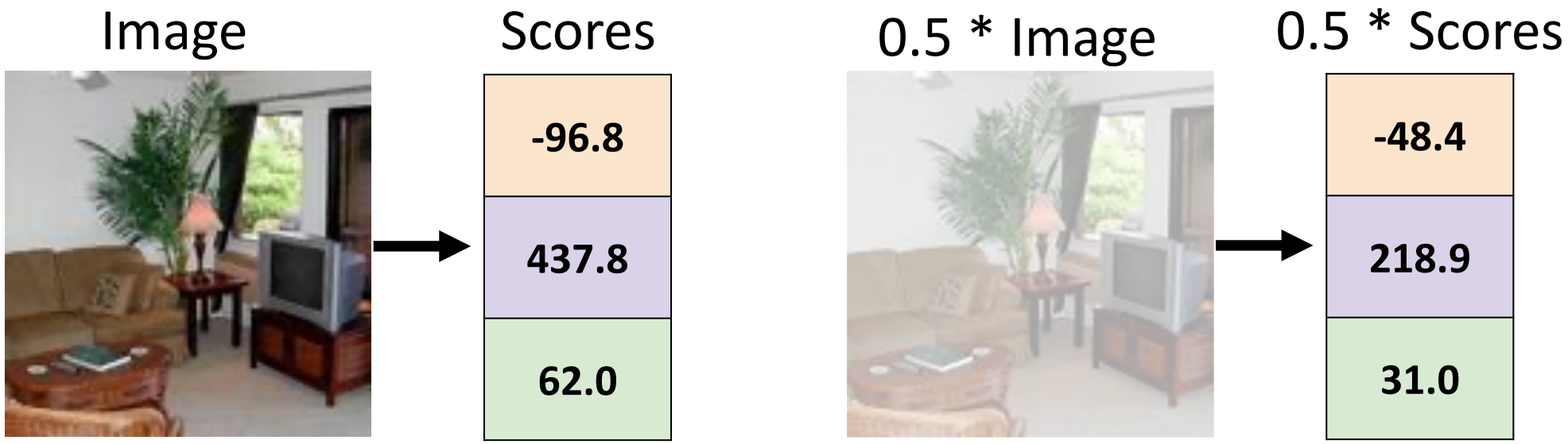
$$f(x, W) = Wx \quad \text{(ignore bias)}$$

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

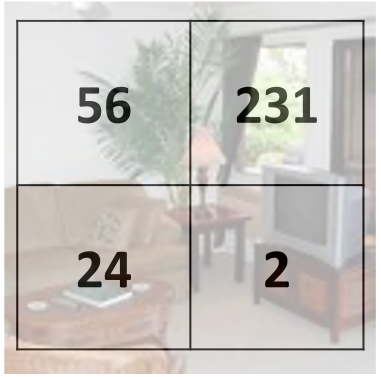
Linear Classifier: Predictions are Linear!

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

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

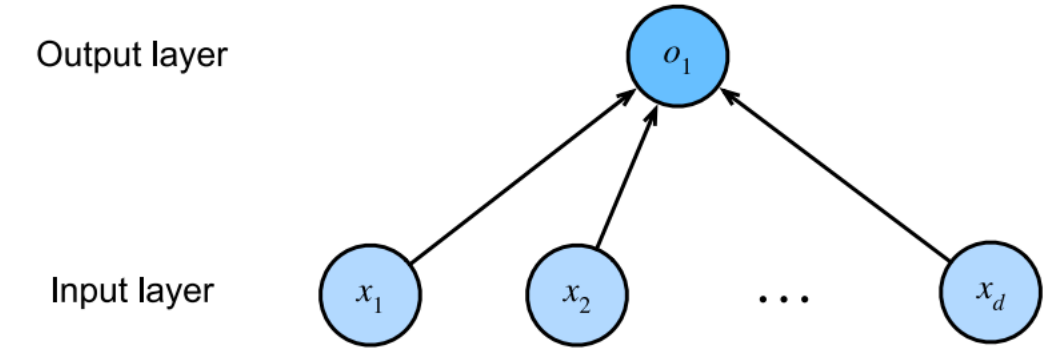


Linear Classifier is a single-layer neural network

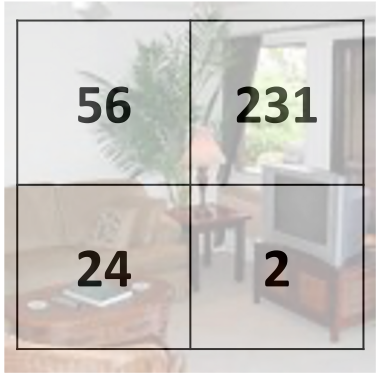


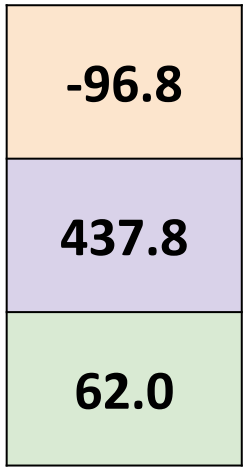
Outdoor-ness

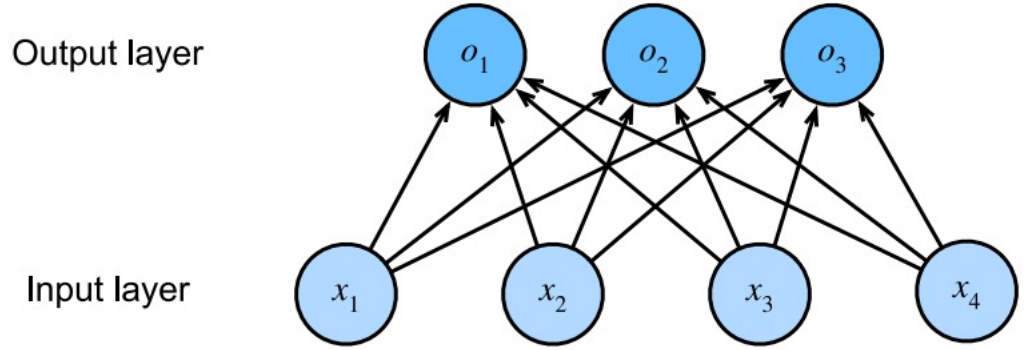
0.15



3 Scene Classification

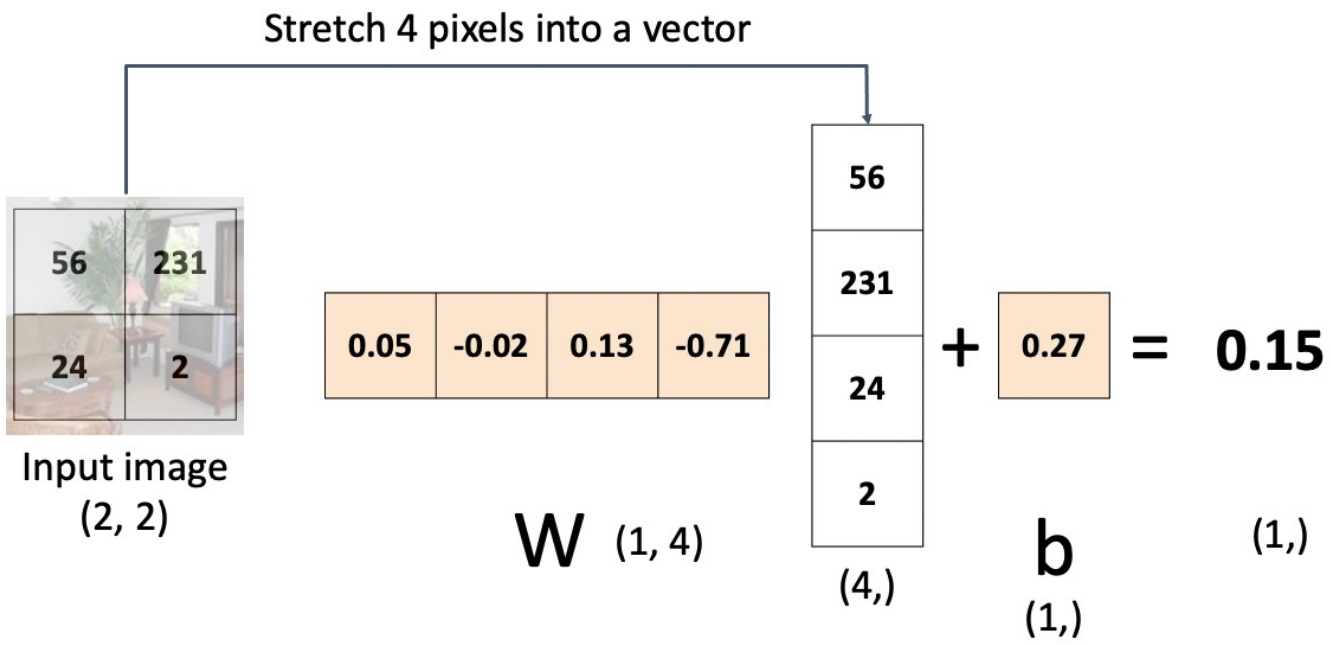






Interpreting a Linear Classifier

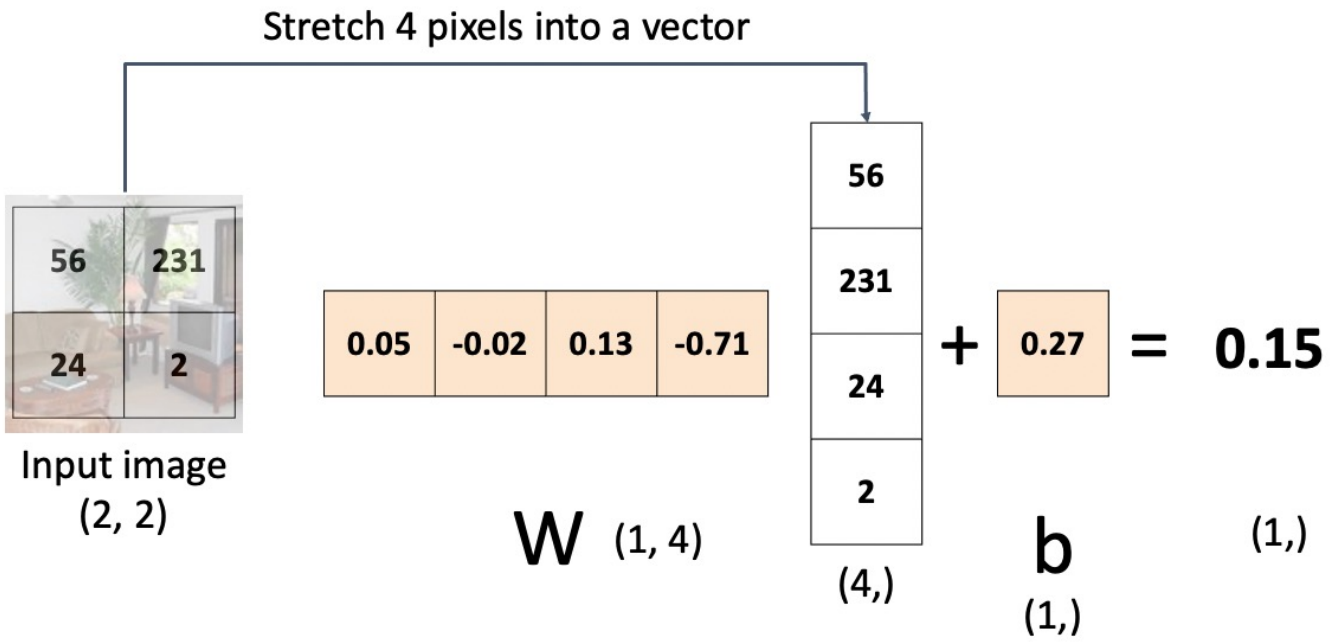
$$f(x,W) = Wx + b$$

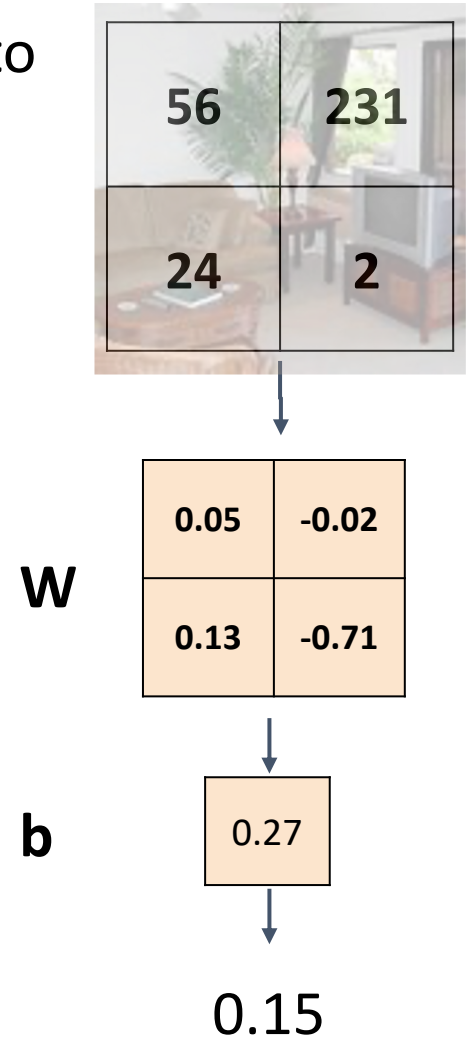


Interpreting a Linear Classifier

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

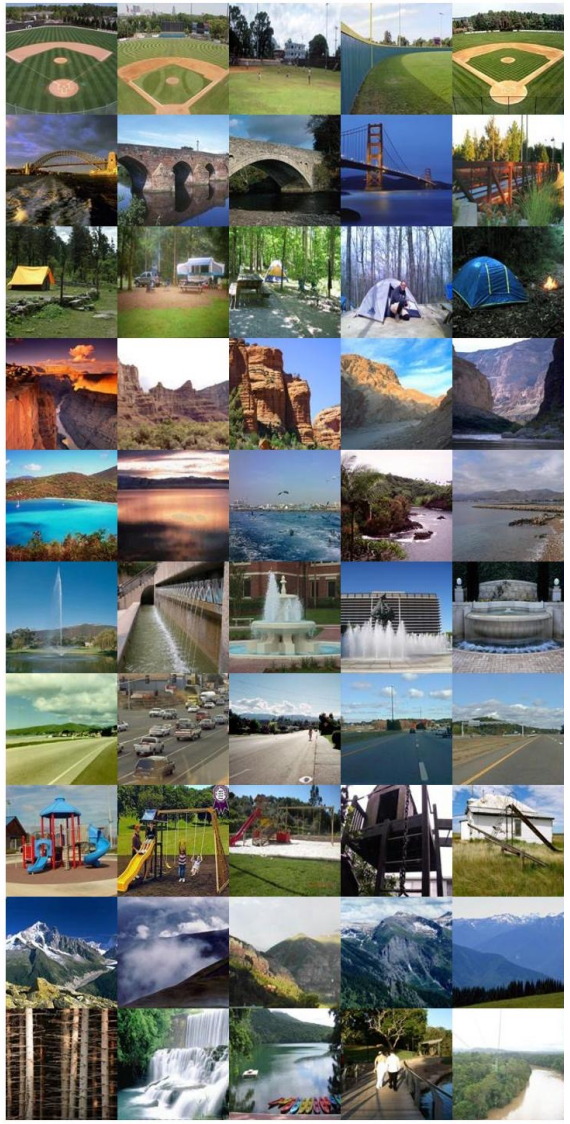
Instead of stretching pixels into a vector we can equivalently stretch rows of W into image!



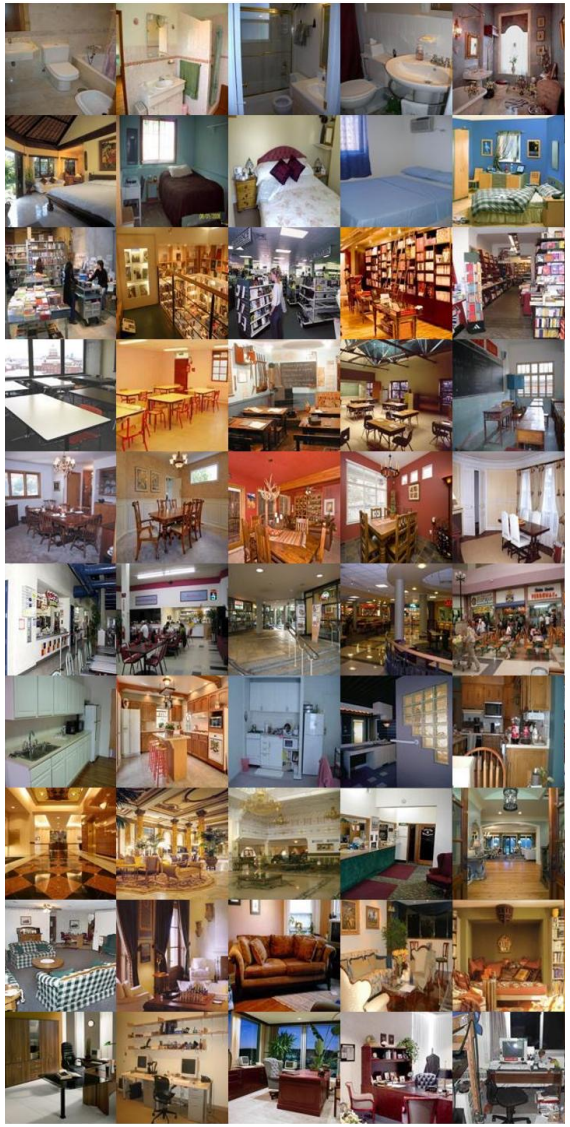


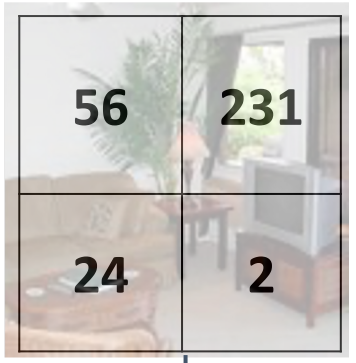
Interpreting a Linear Classifier

10 outdoor scenes



10 indoor scenes





W

0.05	-0.02
0.13	-0.71

b

0.27

0.15

Average images for outdoor and indoor

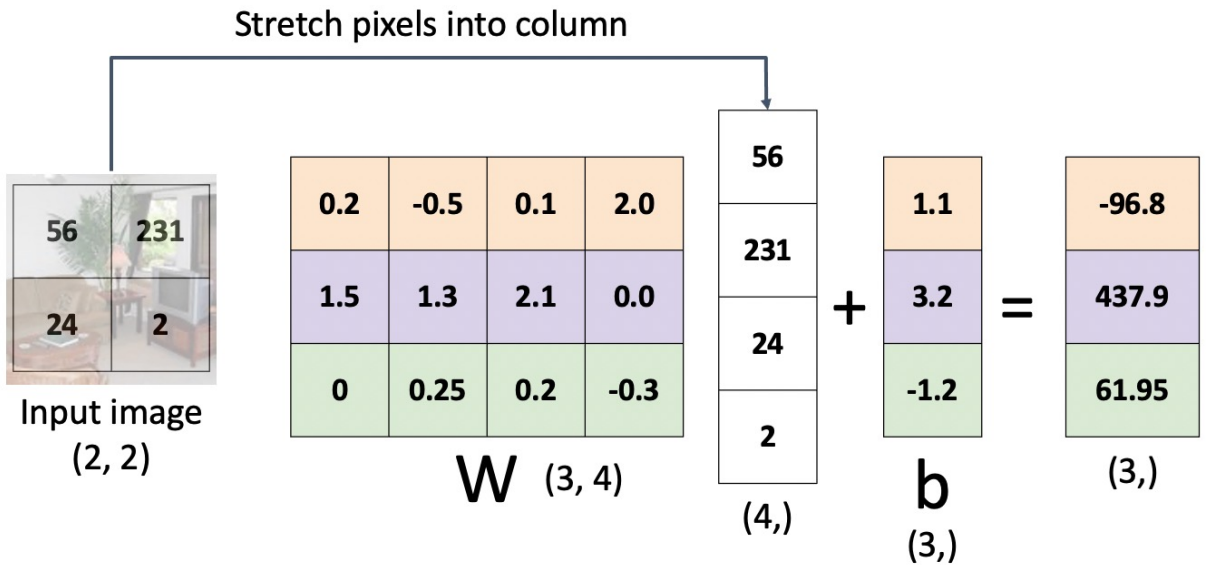
More blue at the top

More dark at the bottom

Interpreting a Linear Classifier

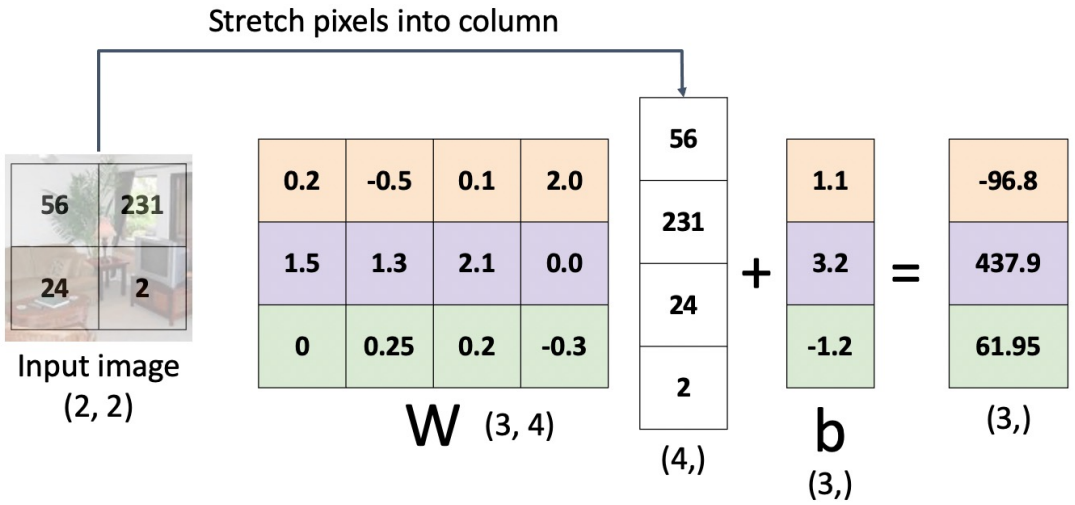
Similar to scene classifier

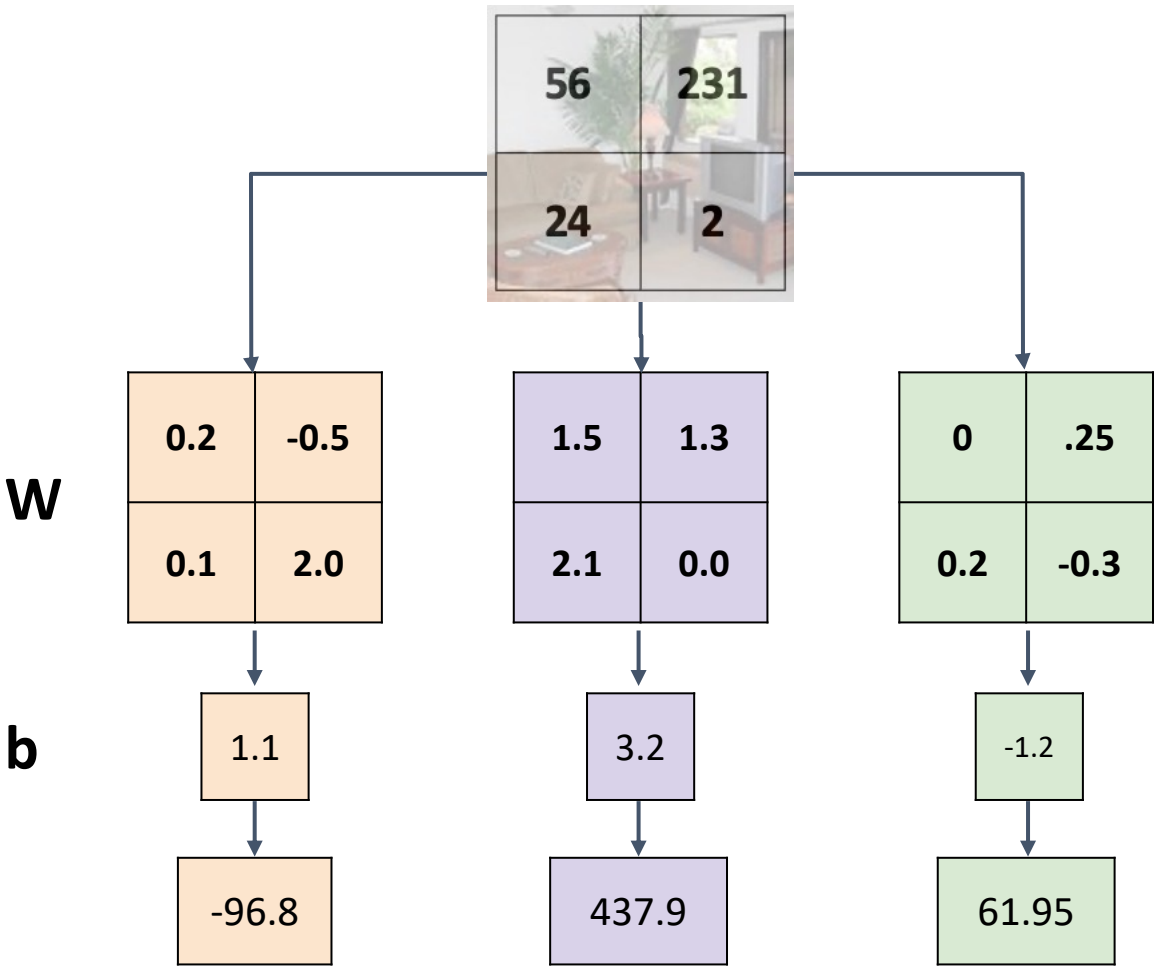
$$f(x,W) = Wx + b$$



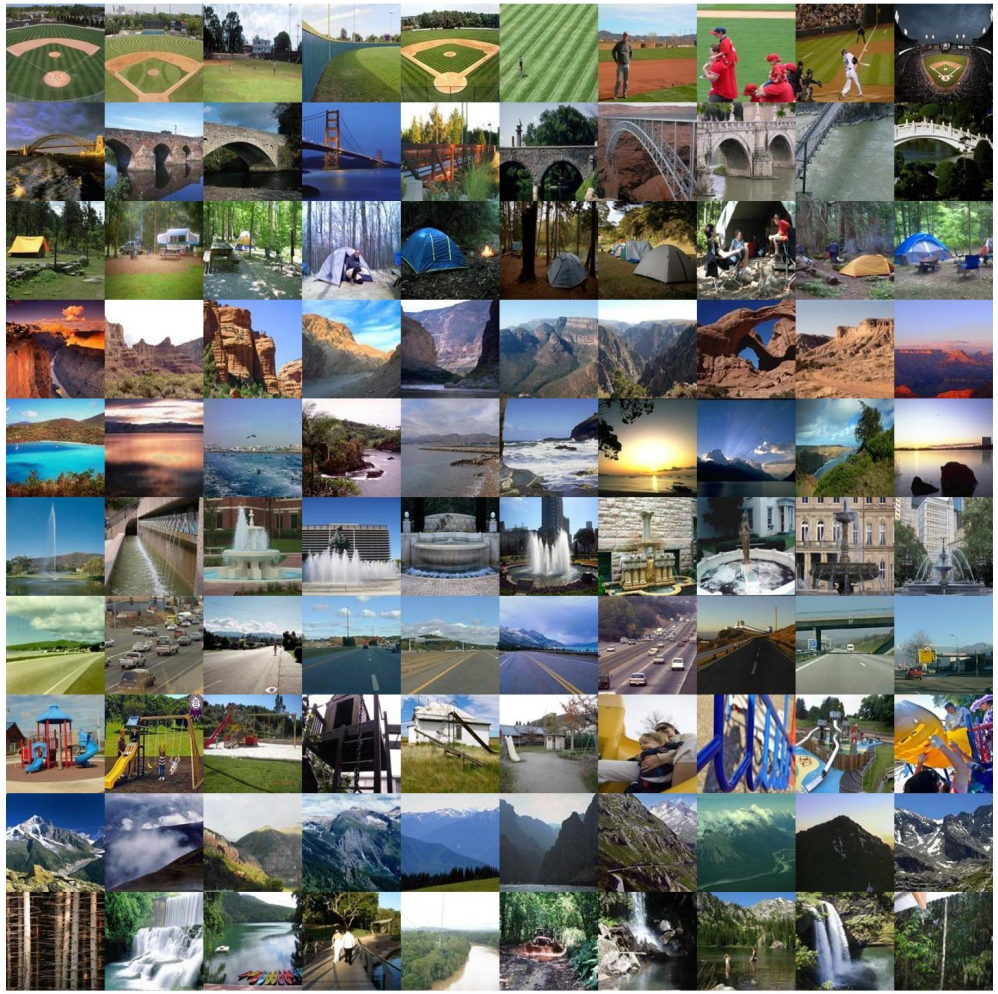
Interpreting a Linear Classifier

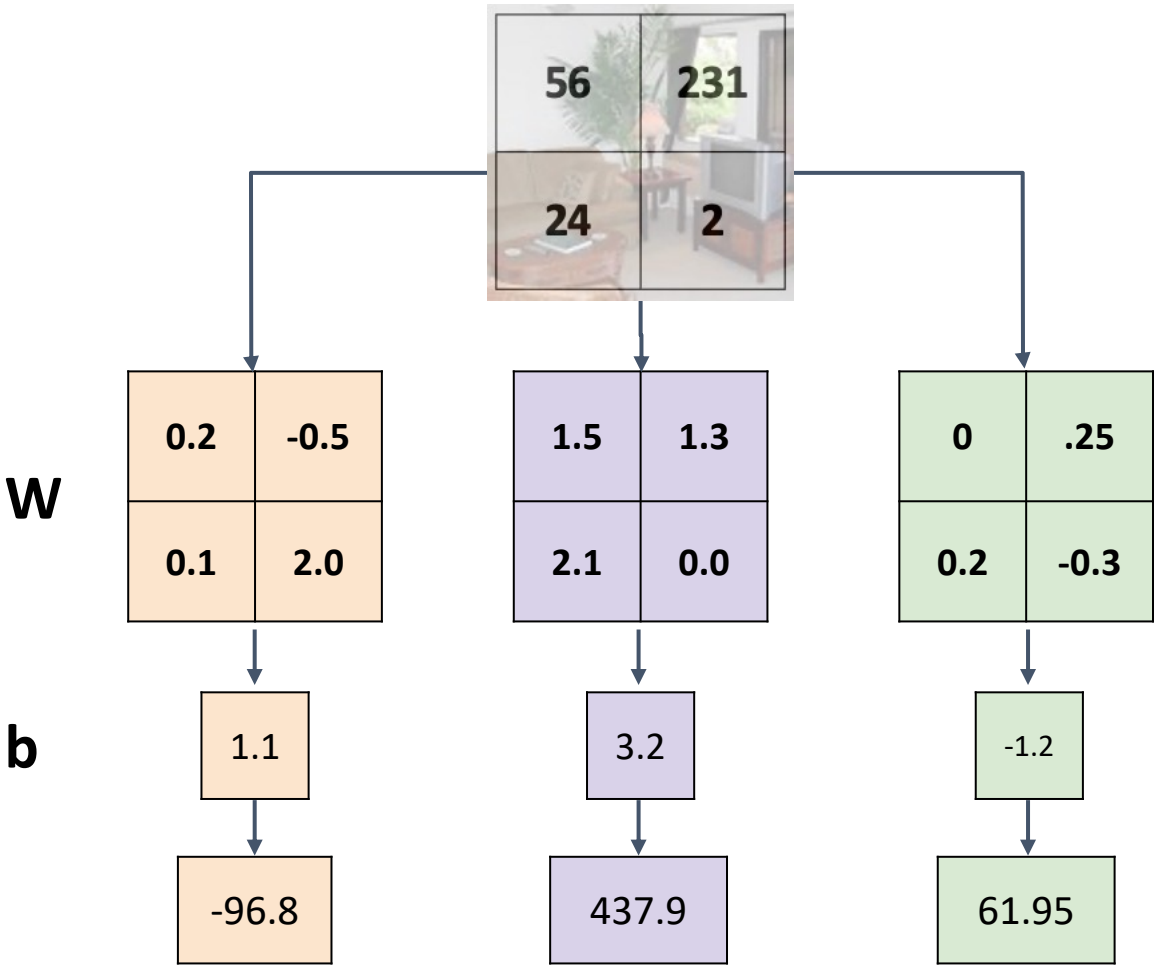
$$f(x,W) = Wx + b$$



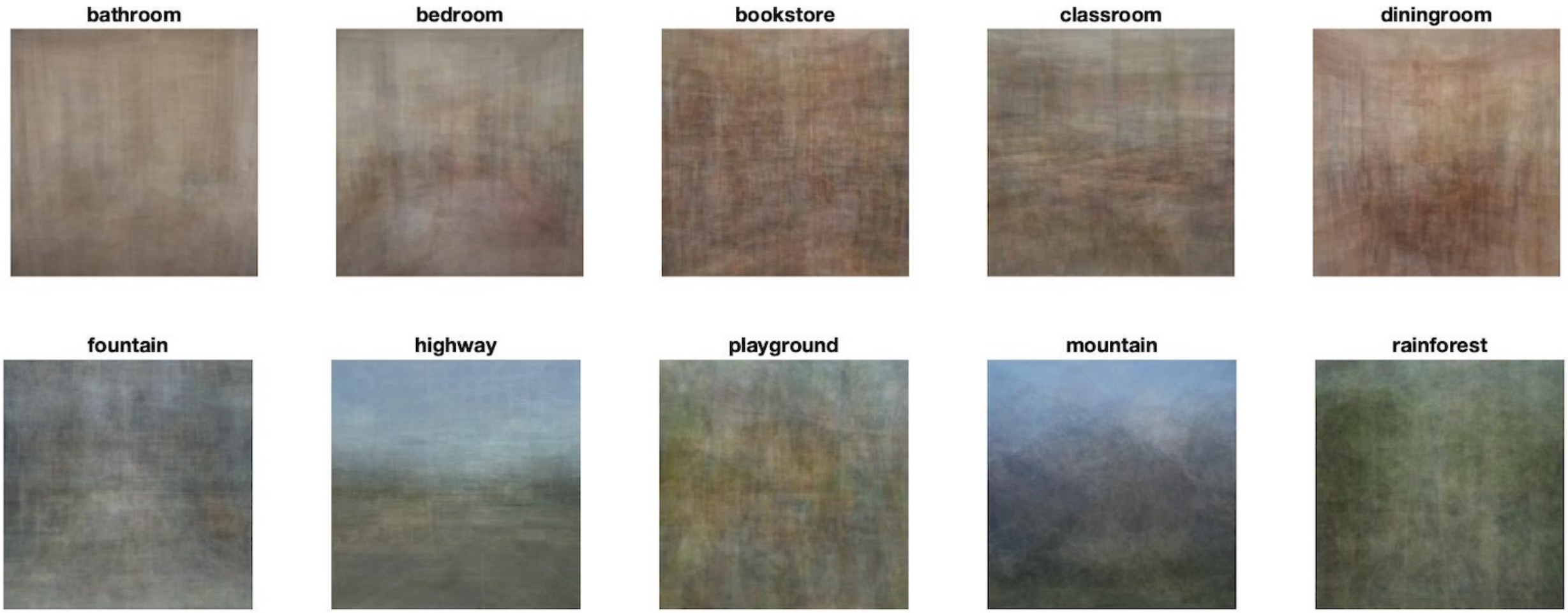


Interpreting a Linear Classifier





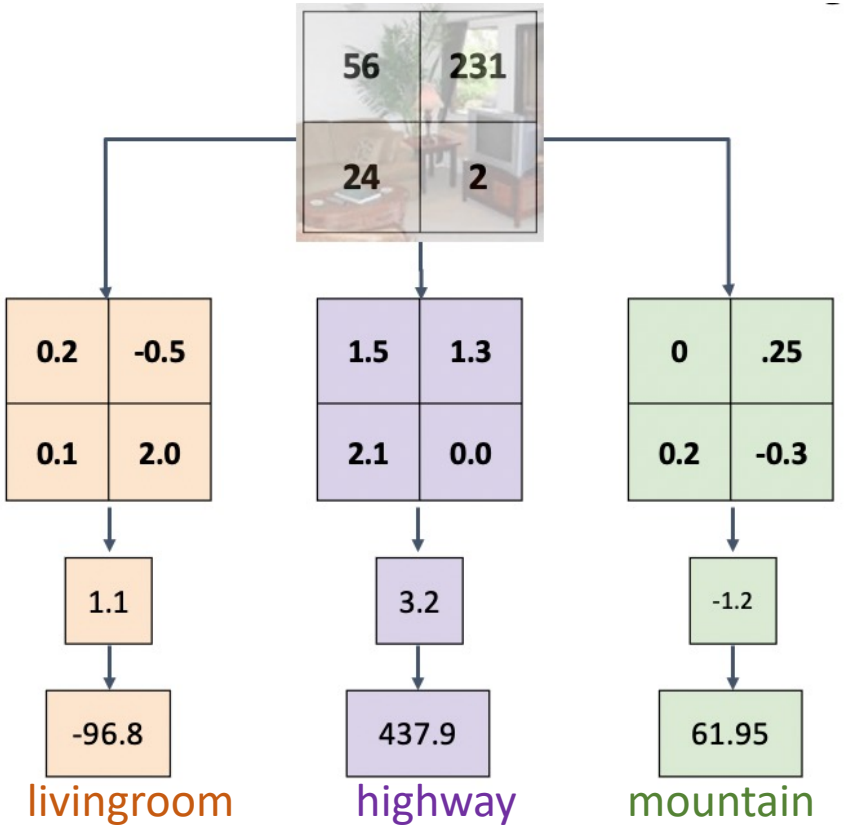
Average Images for Categories of TinyPlaces



Interpreting a Linear Classifier

Linear classifier has one “template” per category, then compute the correlation.

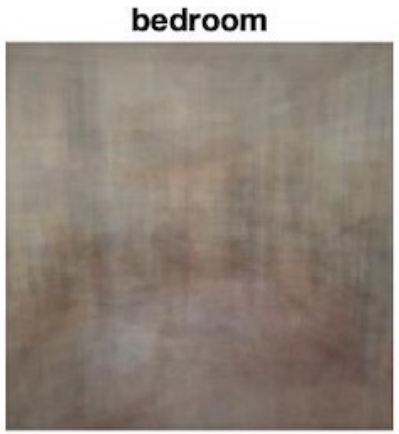
Drawback: A single template cannot capture multiple modes of the data



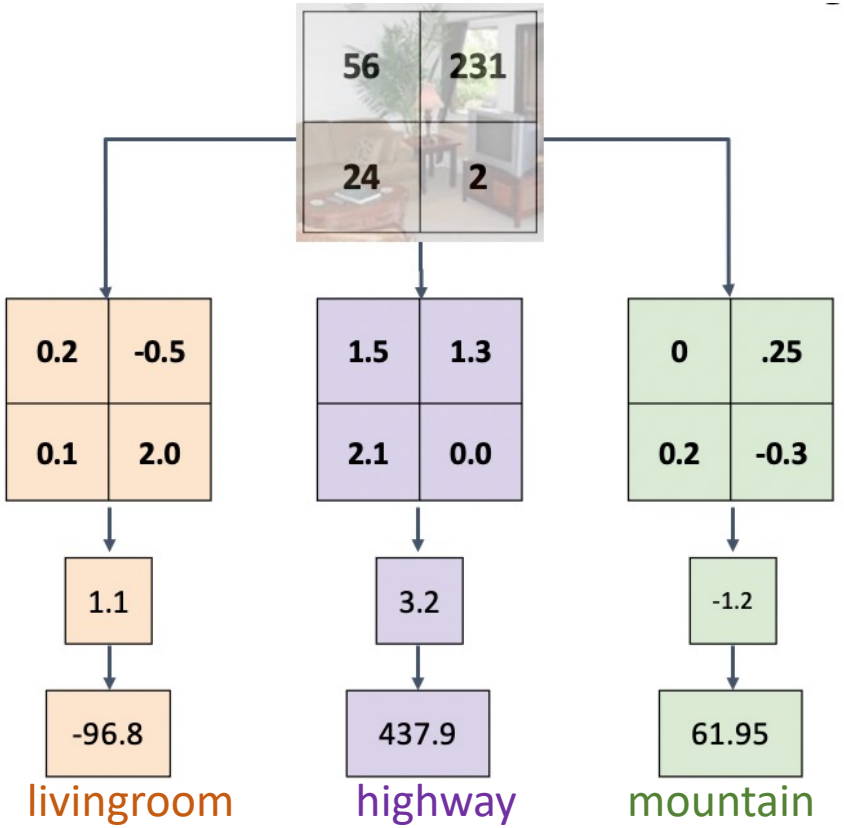
Understanding the Linear Classifier

Drawback: A single template cannot capture multiple modes of the data

Potential improvement: go beyond pixel template



Pose of the bed



Average Image to explore regularities in data

- Side project from Prof. Antonio Torralba (my former PhD advisor)

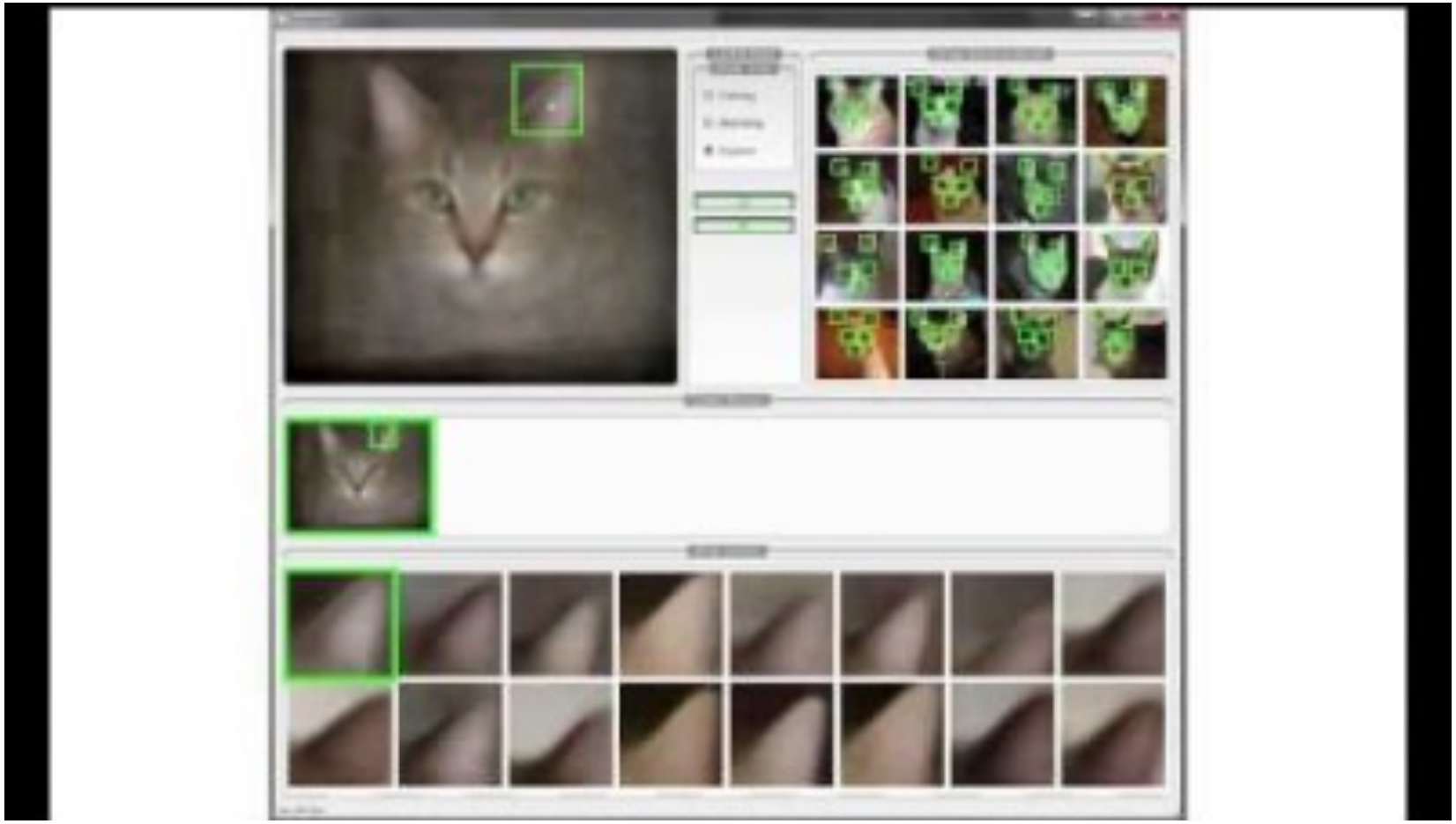
People in Valladolid, Spain

Sailboats in Charles river (spring). Pictures aligned on a sailboat on the right.

https://people.csail.mit.edu/torralba/gallery/

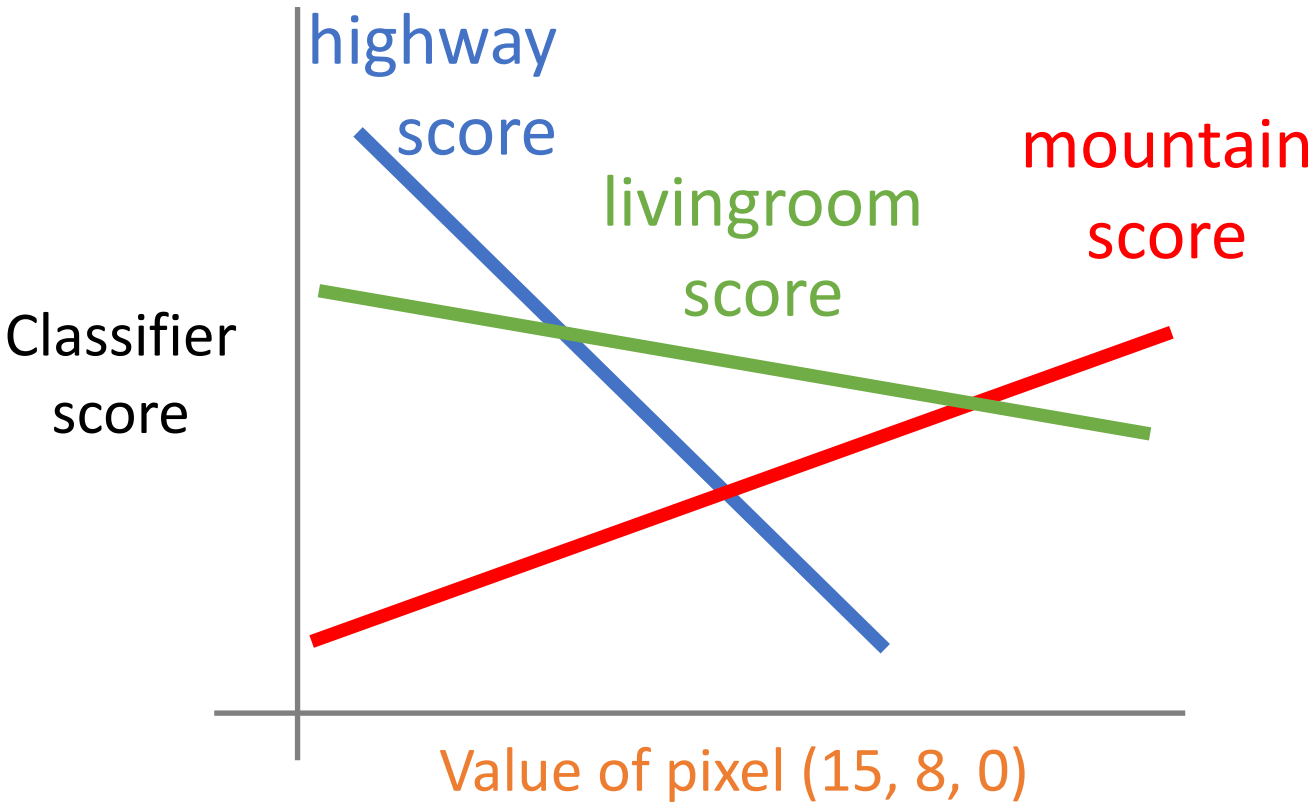
Average Image to explore regularities in data

- AverageExplorer from Zhu, Lee, Efros. SIGGRAPH'14



https://www.cs.cmu.edu/~junyanz/projects/averageExplorer/

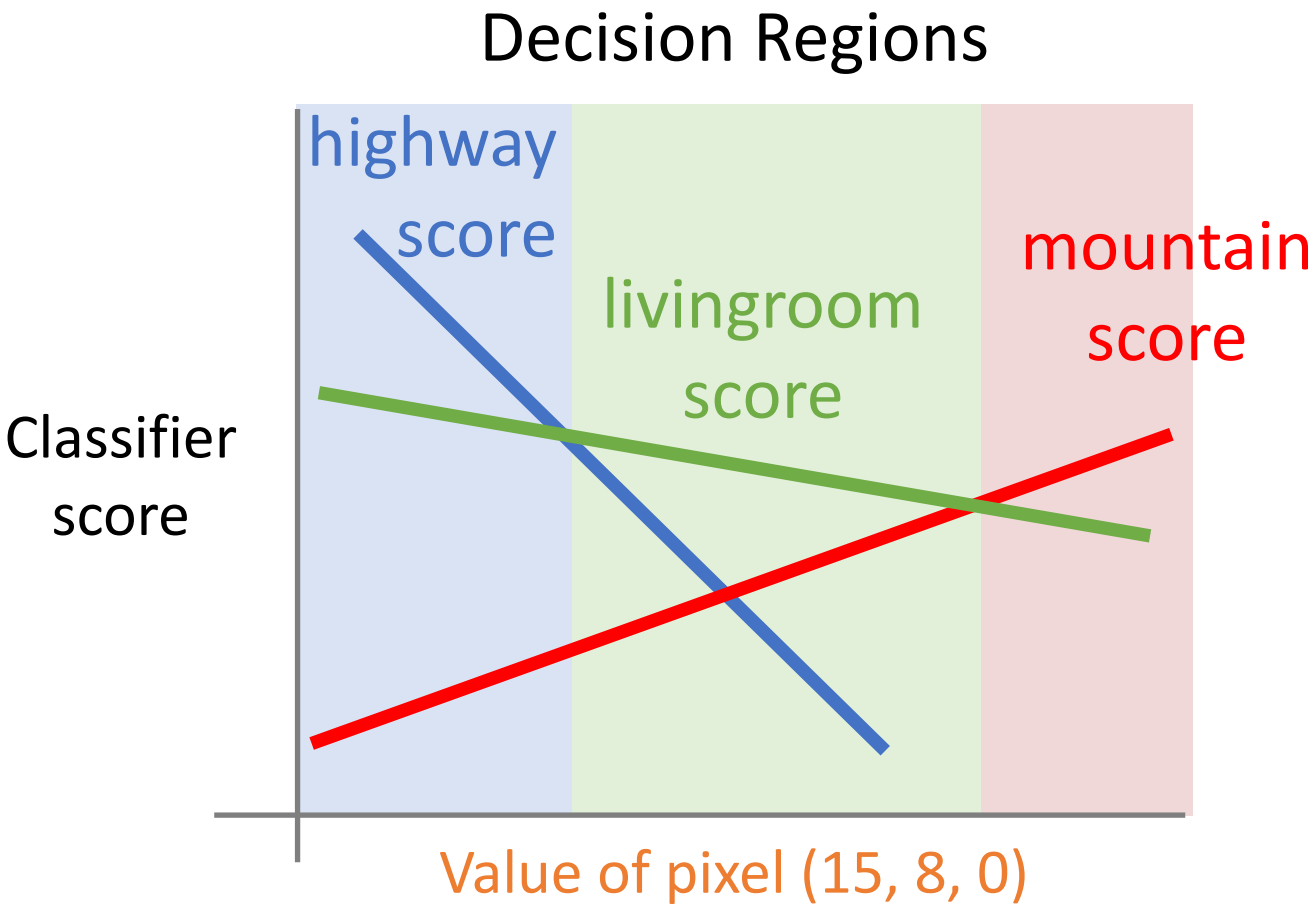
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

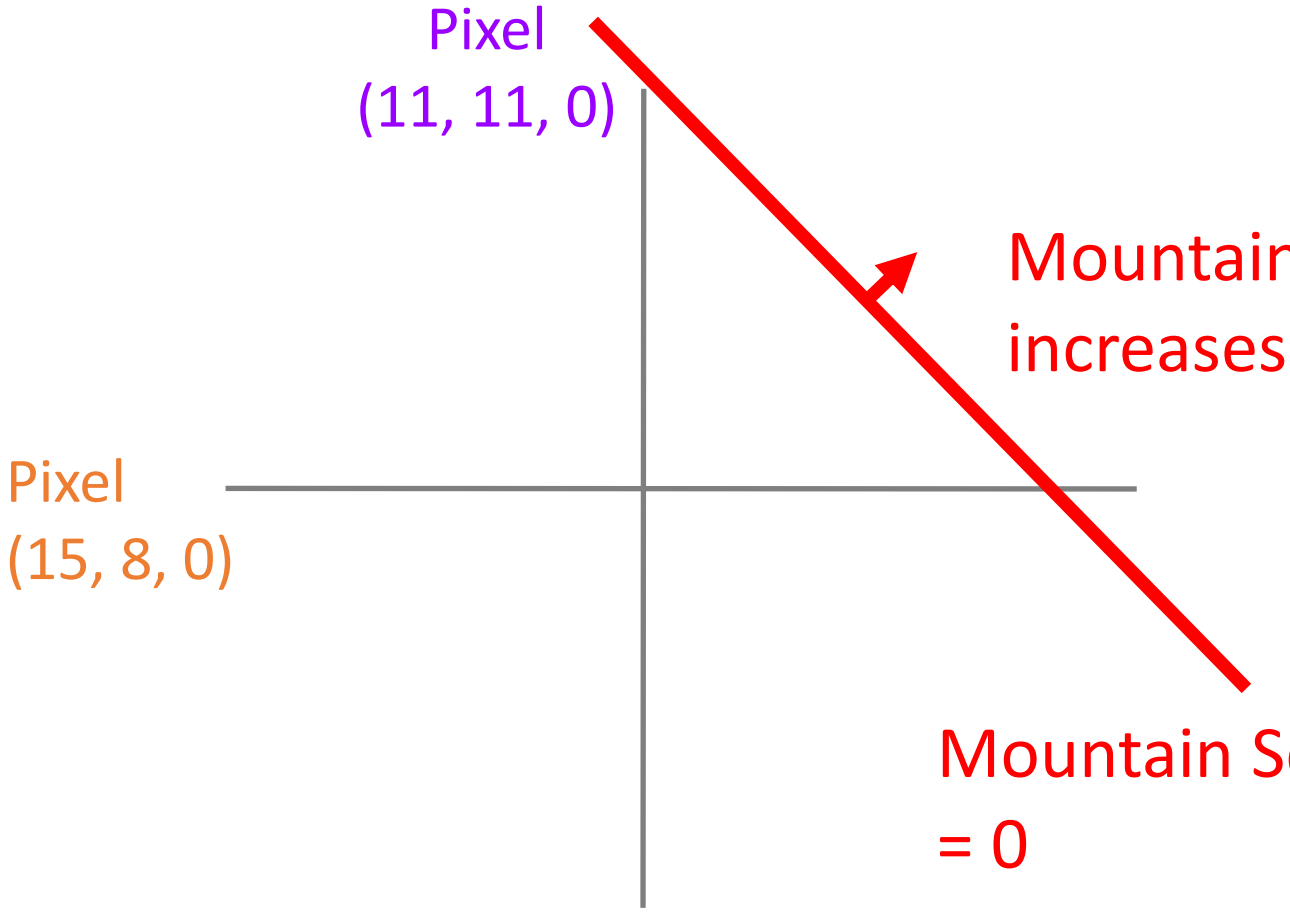
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

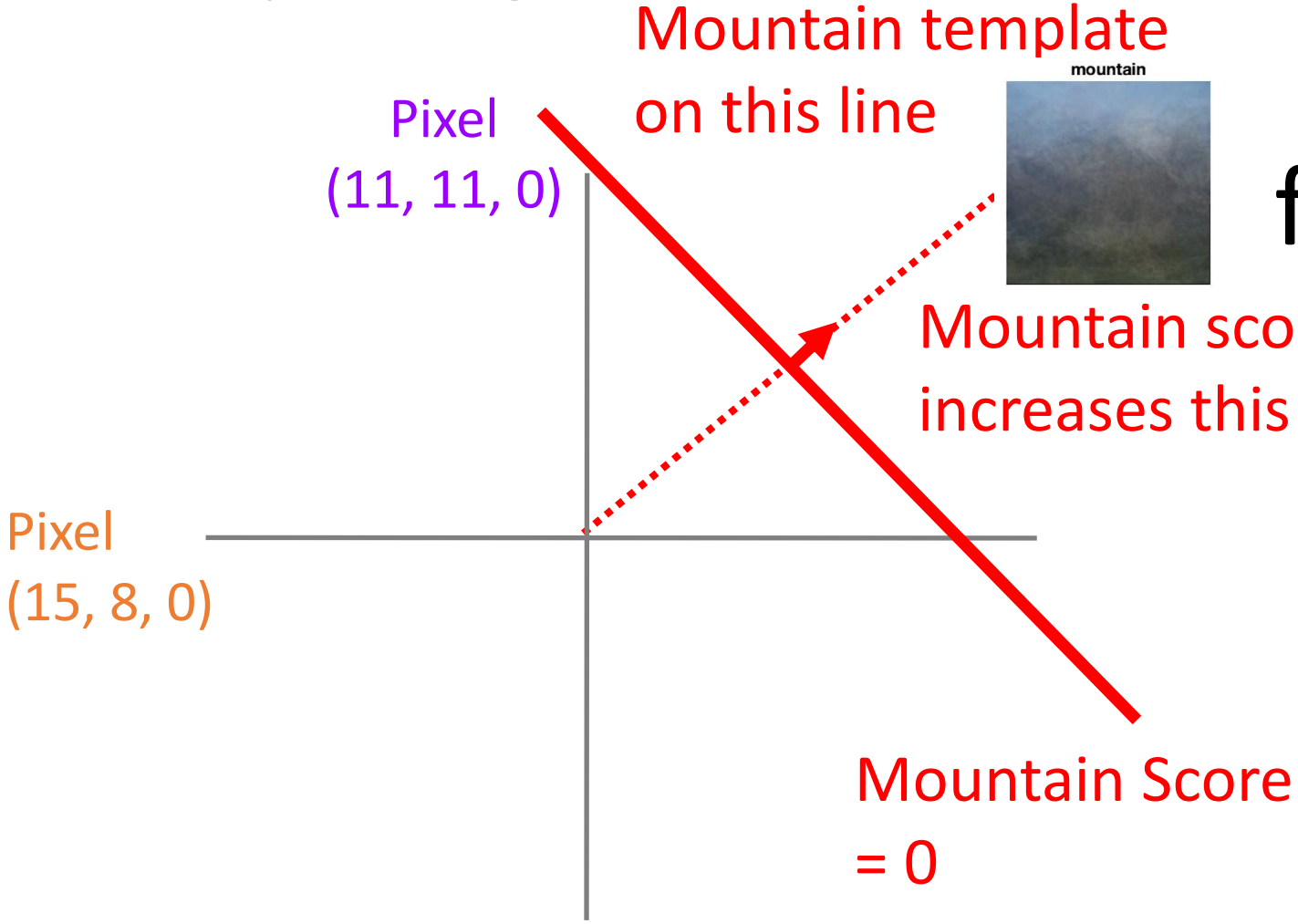
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

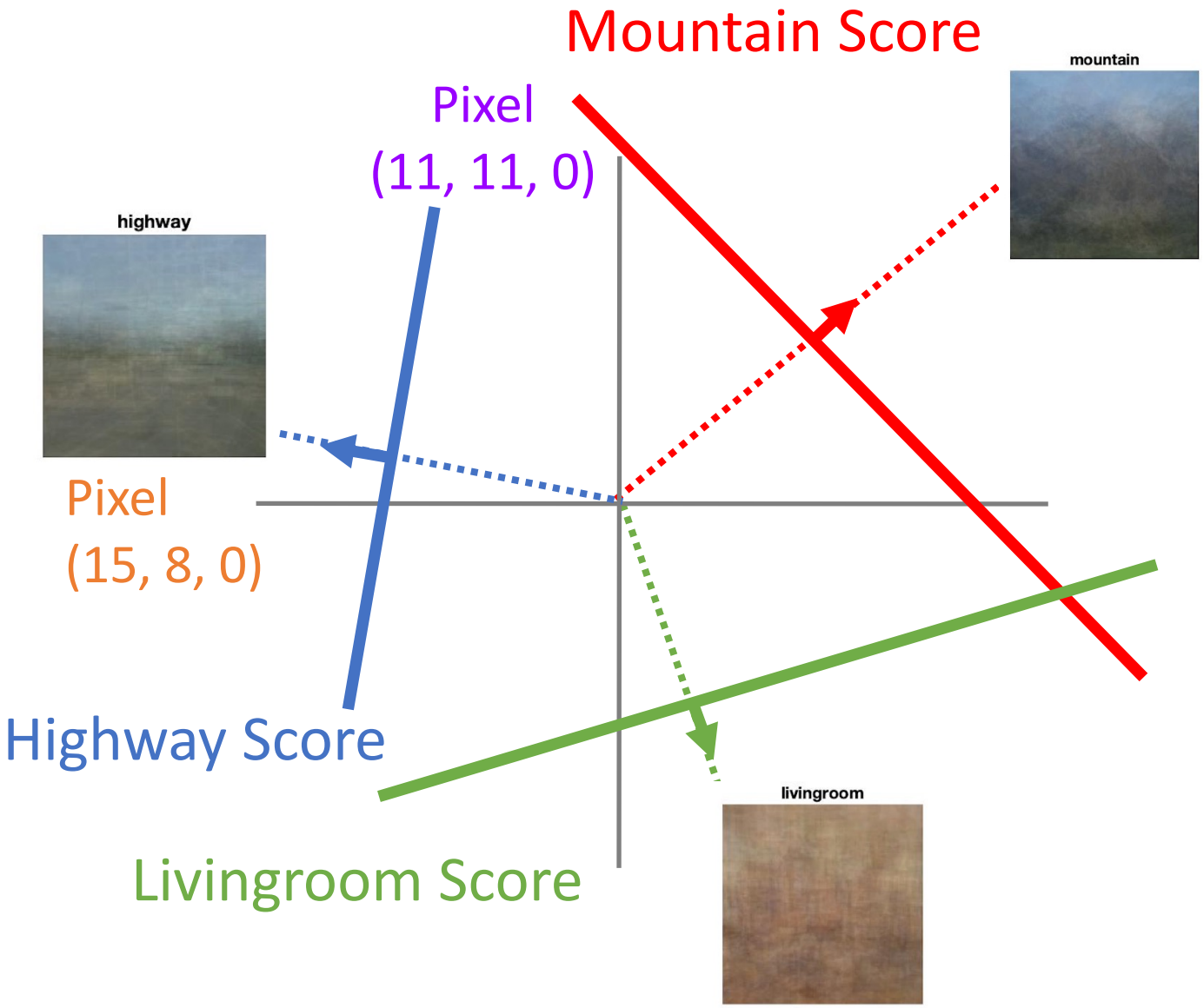
Interpreting a Linear Classifier: Geometric Viewpoint



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

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

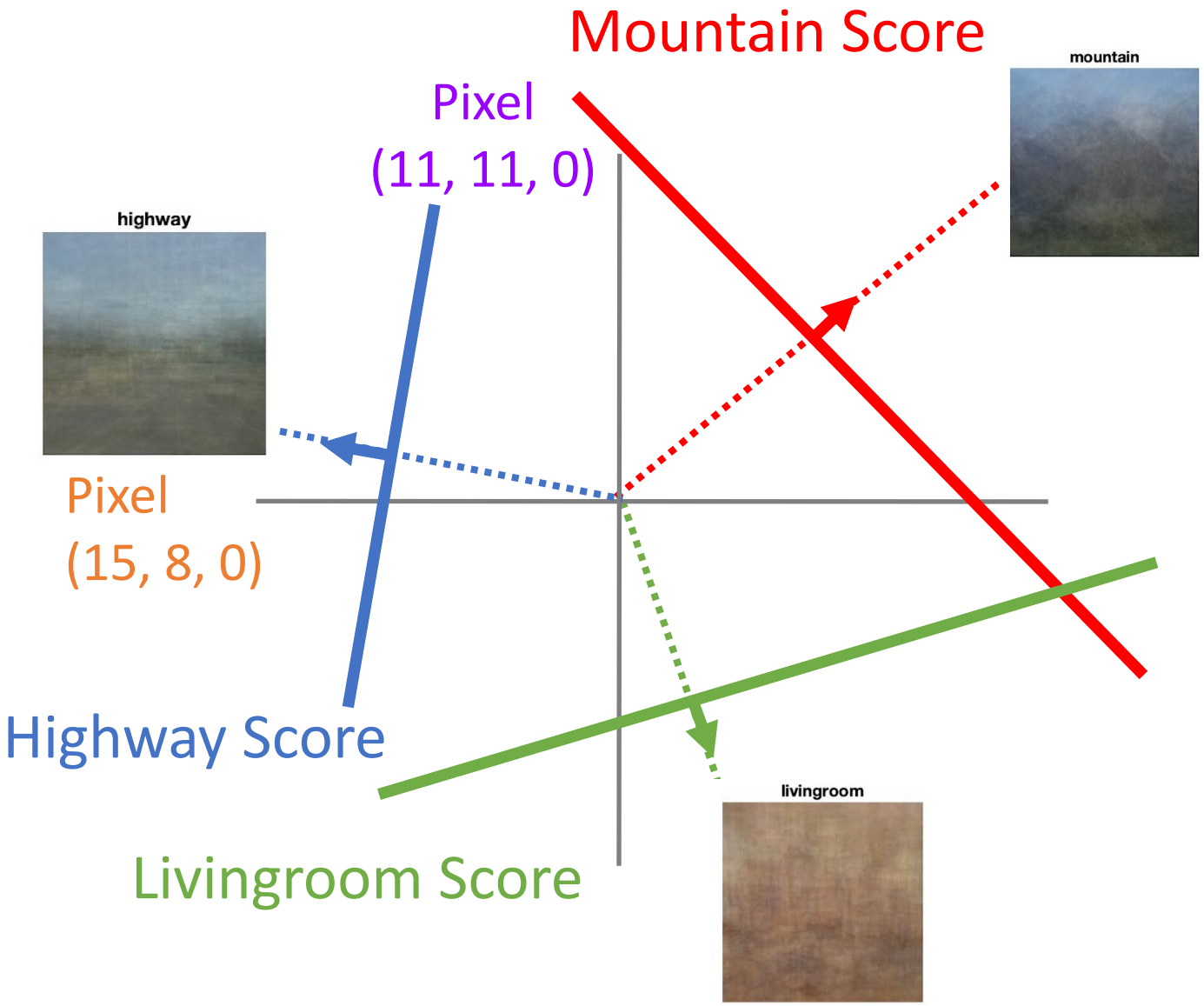
Interpreting a Linear Classifier: Geometric Viewpoint



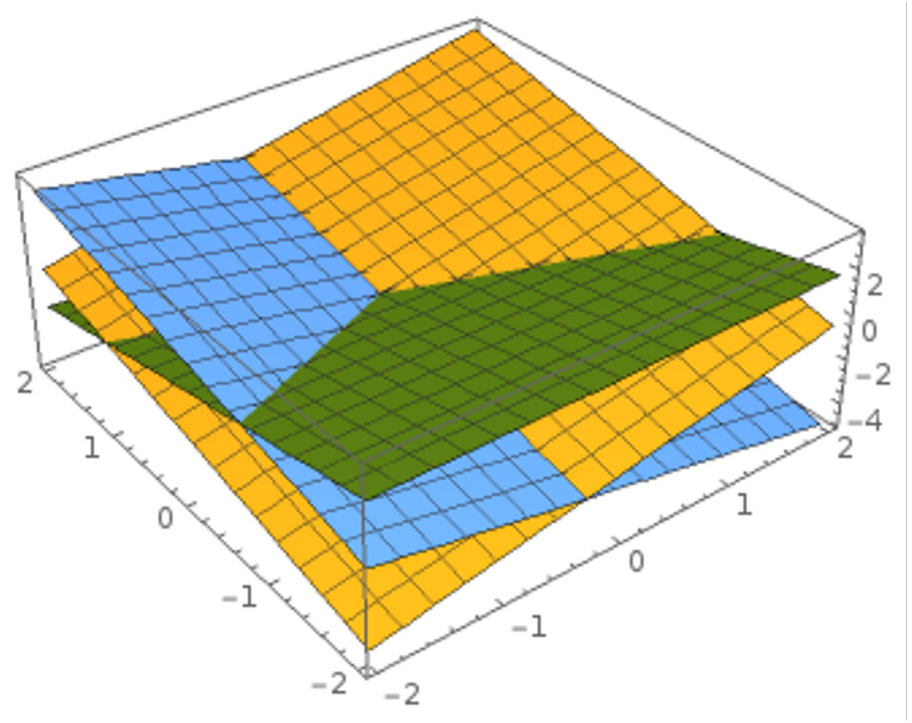
$$f(x,W) = Wx + b$$

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

Interpreting a Linear Classifier: Geometric Viewpoint



Hyperplanes carving up a high-dimensional space

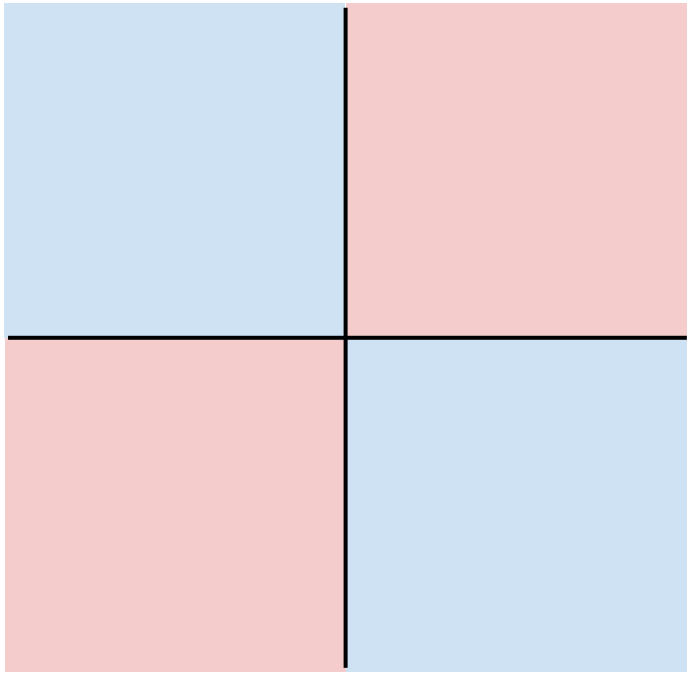


Plot created using Wolfram Cloud

Hard Cases for a Linear Classifier

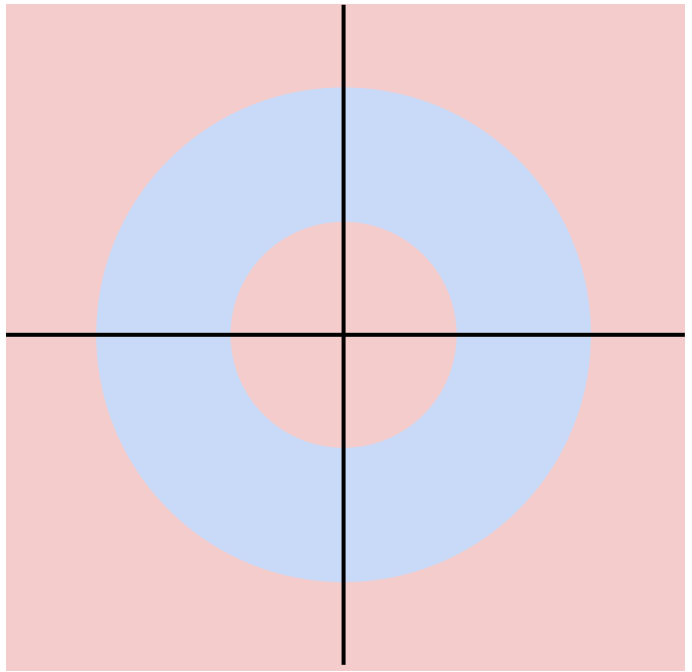
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



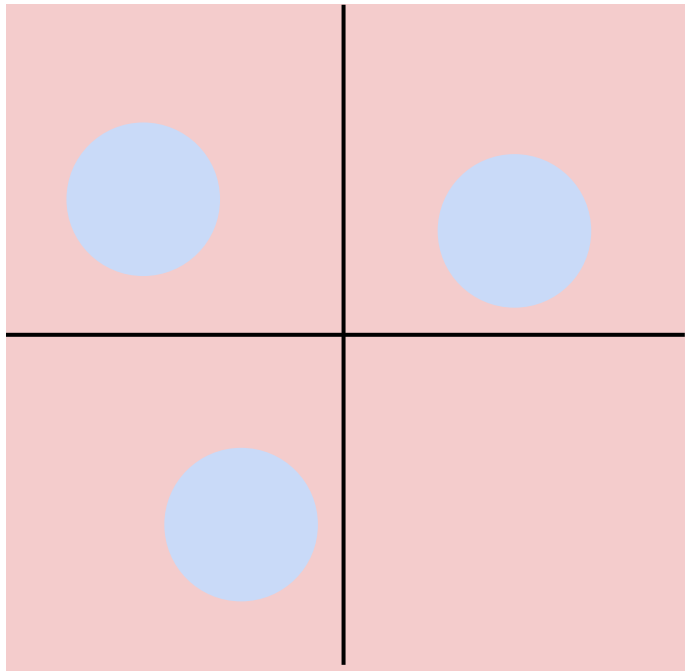
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



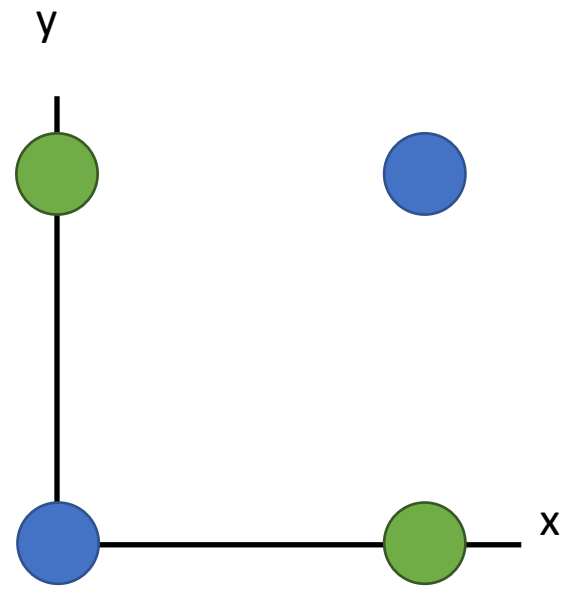
Class 1:
Three modes

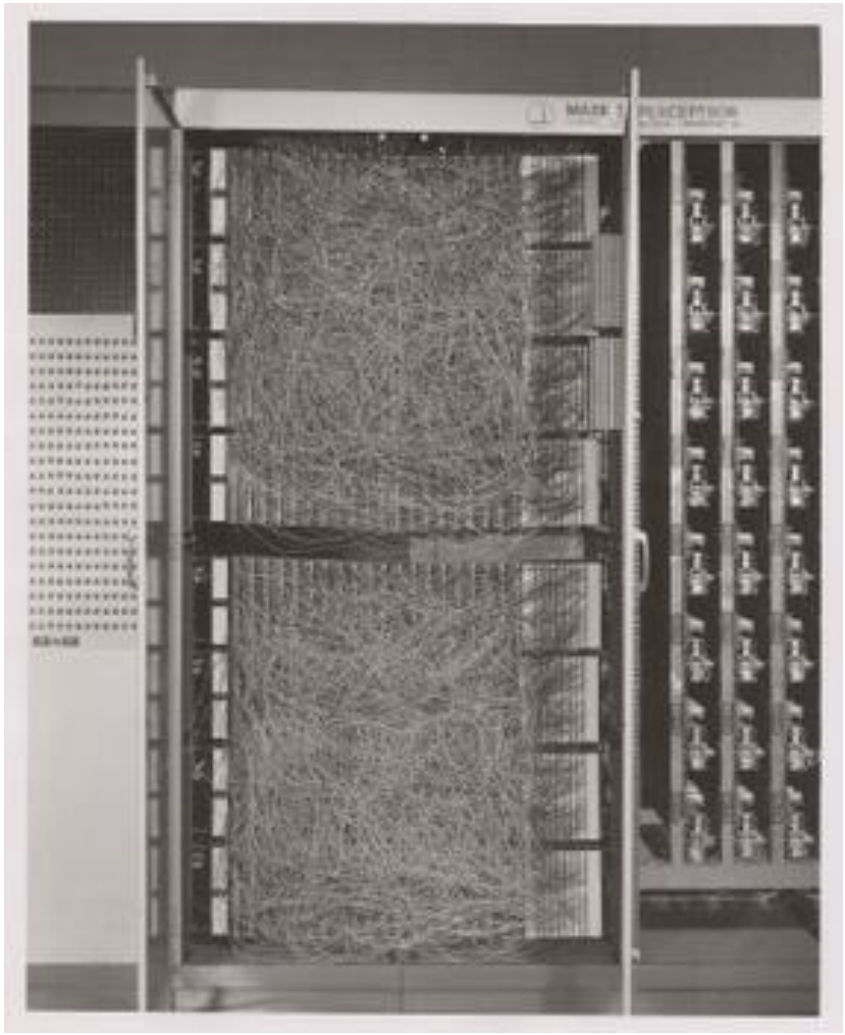
Class 2:
Everything else



Recall: Perceptron couldn’t learn XOR

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





A live demo on training classifier

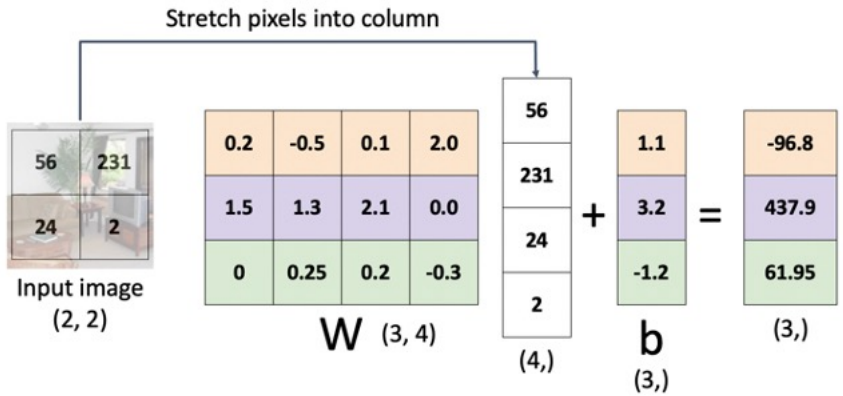
- https://playground.tensorflow.org/

Linear NN/Classifier: Three Viewpoints

Equation Viewpoint

f(x,W) = Wx

f(x,W) = Wx + b

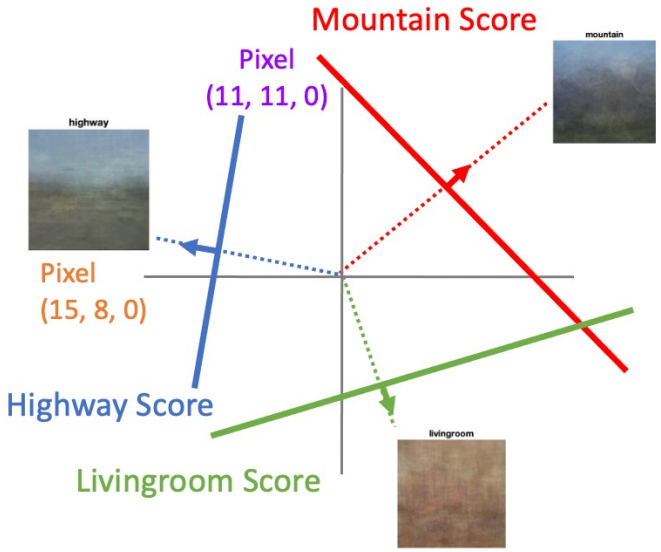


Visualization Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space



So Far: Defined a linear score function

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

Wrong prediction1

bathroom	-1.09	1.42	**2.33**
bedroom	0.03	0.29	-0.89
bookstore	0.55	0.20	0.10
classroom	1.10	1.59	-0.54
dining_room	1.54	-0.80	0.30
food_court	0.09	0.70	-0.60
kitchen	-1.49	0.84	0.49
lobby	-0.74	-0.24	0.74
living_room	**2.01**	0.22	1.71
office	2.35	-1.17	-0.19
baseball_field	-0.62	-1.15	-2.14
bridge	0.75	0.10	-0.84
campsite	-0.19	0.72	1.35
canyon	0.89	2.59	-1.07
coast	-0.76	-0.67	0.96
fountain	-1.40	**2.76**	0.12
highway	-1.42	-0.08	1.44
playground	0.49	-1.93	-1.96
mountain	-0.18	-0.44	-0.20
rainforest	-0.20	-1.79	-1.21

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

But how can we have a good W?

Choosing a good W

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

bathroom	-1.09	1.42	**2.33**
bedroom	0.03	0.29	-0.89
bookstore	0.55	0.20	0.10
classroom	1.10	1.59	-0.54
dining_room	1.54	-0.80	0.30
food_court	0.09	0.70	-0.60
kitchen	-1.49	0.84	0.49
lobby	-0.74	-0.24	0.74
living_room	**2.01**	0.22	1.71
office	2.35	-1.17	-0.19
baseball_field	-0.62	-1.15	-2.14
bridge	0.75	0.10	-0.84
campsite	-0.19	0.72	1.35
canyon	0.89	2.59	-1.07
coast	-0.76	-0.67	0.96
fountain	-1.40	**2.76**	0.12
highway	-1.42	-0.08	1.44
playground	0.49	-1.93	-1.96
mountain	-0.18	-0.44	-0.20
rainforest	-0.20	-1.79	-1.21

Objective:

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

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

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

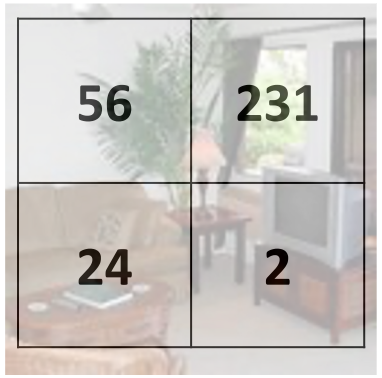
Loss for a single example is

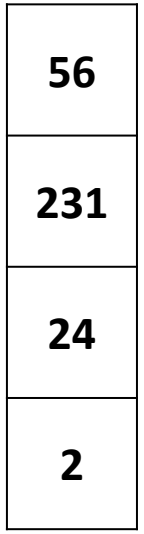
$$L(f(x_i, W), y_i)$$

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

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

Linear Regression Loss

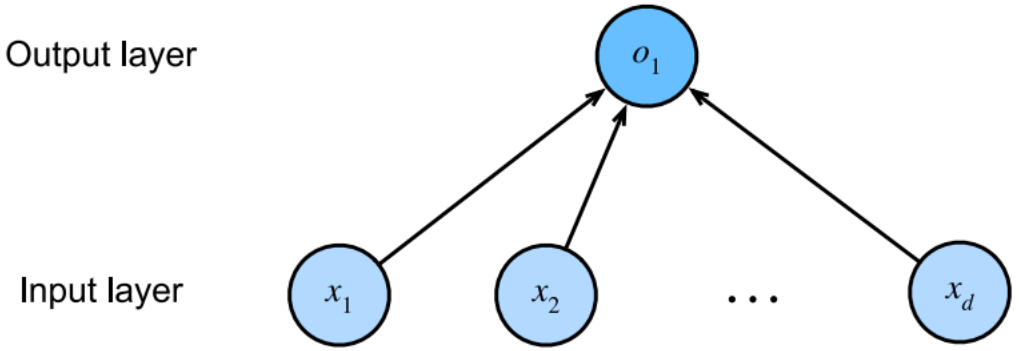


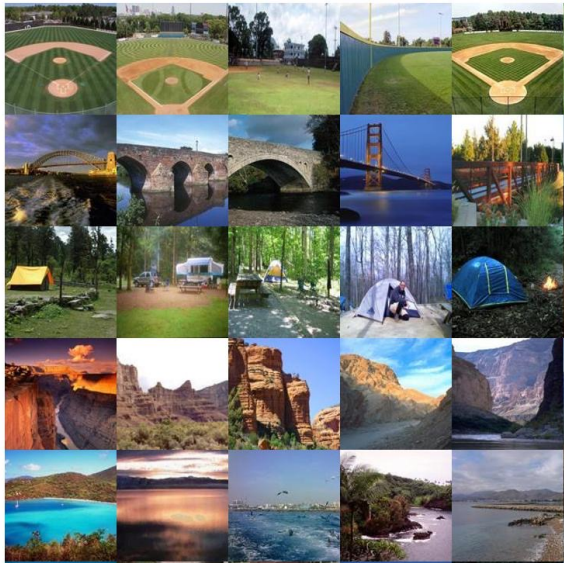


Outdoor-ness

0.15

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



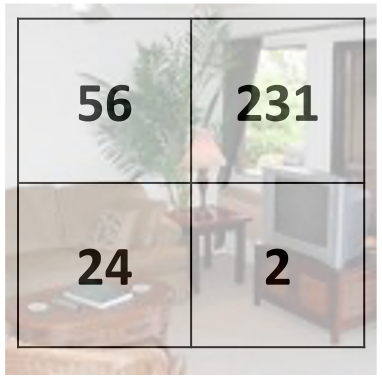


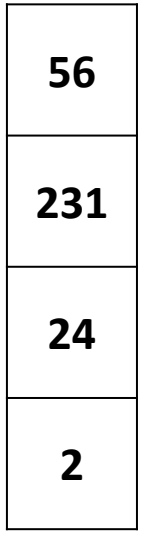
Outdoor images

Indoor images

- Set the output for indoor images as 0 and for outdoor images as 1
- Categorical label is turned into a scalar value

Linear Regression Loss

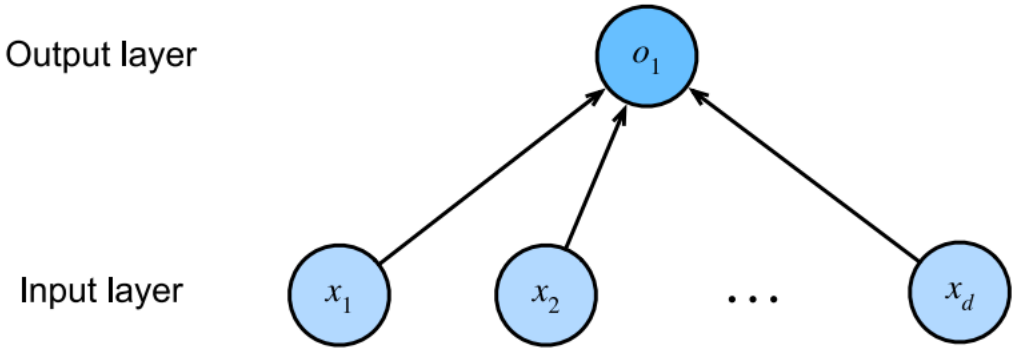




Outdoor-ness

0.15

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

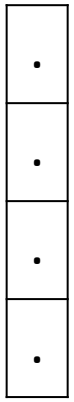


Squared loss for one sample: $L_i(W) = \frac{1}{2}(f(x_i, W) - y_i)^2$

Squared loss for all the training samples: $L(W) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(f(x_i, W) - y_i)^2$

Learning objective: $W^* = \underset{W}{\operatorname{argmin}}\, L(W)$

Linear Regression Loss



Model output

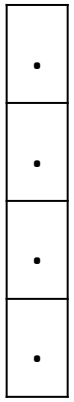
→ **0.15**

Loss

$\frac{1}{2}(0.15 - 1)^2$

Ground truth

1



→ **0.05**

$\frac{1}{2}(0.05 - 0)^2$

0

Analytic Solution for Linear Regression Loss

- You will have a practice in the Assignment 1 for outdoor and indoor classification with a linear regression loss

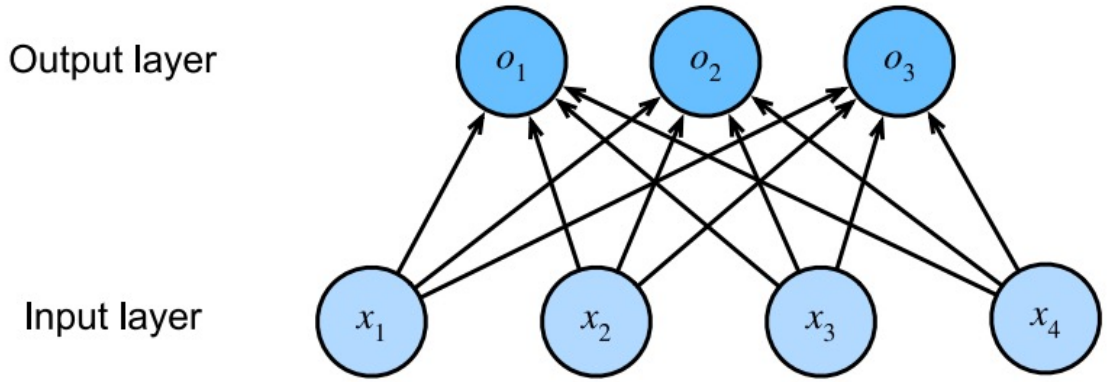
$$L(W) = \frac{1}{2n}\sum_{i=1}^{n}(\boldsymbol{y_i} - \boldsymbol{W x_i})^2$$

$$\boldsymbol{W^* = \left(X^T X\right)^{-1} X^T y}, \textbf{ where } \boldsymbol{X} \textbf{ is the batch of all the data}$$

In which situation this close-form solution cannot be used directly?

Cross-Entropy Loss (Multinomial Logistic Regression)

3 scene classification



Bridge **3.2**

Mountain 5.1

Coast -1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

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

Bridge	**3.2**
Mountain	5.1
Coast	-1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

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

$$s = f(x_i; W) \qquad P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$

Softmax function

Bridge	**3.2**
Mountain	5.1
Coast	-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

Bridge	**3.2**
Mountain	5.1
Coast	-1.7

Unnormalized log-probabilities / logits

Cross-Entropy Loss (Multinomial Logistic Regression)

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

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

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

Softmax function

Probabilities must be >= 0

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities
Bridge	**3.2**	→	**24.5**
Mountain	5.1	→	164.0
Coast	-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

Bridge	**3.2**	**24.5**	**0.13**
Mountain	5.1	164.0	0.87
Coast	-1.7	0.18	0.00

exp

normalize

Unnormalized log-probabilities / logits

unnormalized probabilities

probabilities

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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1
Bridge	**3.2**		**24.5**		**0.13**
Mountain	5.1		164.0		0.87
Coast	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

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

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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1
Bridge	**3.2**		**24.5**		**0.13**
Mountain	5.1		164.0		0.87
Coast	-1.7		0.18		0.00
	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities

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

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data

Cross-Entropy Loss (Multinomial Logistic Regression)

One-hot encoding for the categorical label of each sample




Bridge	1	1	0	0
Mountain	0	0	1	0
Coast	0	0	0	1

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	exp	unnormalized probabilities	normalize	probabilities	Compare	Correct probs
Bridge	**3.2**		**24.5**		**0.13**		**1.00**
Mountain	5.1		164.0		0.87		0.00
Coast	-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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1	Compare	Correct probs
Bridge	**3.2**	→	**24.5**	→	**0.13**	→	**1.00**
Mountain	5.1		164.0		0.87		0.00
Coast	-1.7		0.18		0.00		0.00

Kullback–Leibler divergence

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

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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1	Compare	Correct probs
Bridge	**3.2**	→	**24.5**	→	**0.13**	→	**1.00**
Mountain	5.1		164.0		0.87		0.00
Coast	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(P, Q) = -\sum p(x) \log q(x)$$
$$H(P) + D_{KL}(P \,||\, Q)$$

Then H(P)=0, and D_{KL} (P||Q) will have just one term for the correct class

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

Bridge **3.2**

Mountain 5.1

Coast -1.7

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**

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

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

Softmax function

Maximize probability of correct class

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

Putting it all together:

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

Bridge	**3.2**
Mountain	5.1
Coast	-1.7

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

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

Maximize probability of correct class

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

Putting it all together:

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

Bridge	**3.2**
Mountain	5.1
Coast	-1.7

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**

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

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

Softmax function

Maximize probability of correct class

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

Putting it all together:

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

Bridge	**3.2**
Mountain	5.1
Coast	-1.7

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

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

Maximize probability of correct class

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

Putting it all together:

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

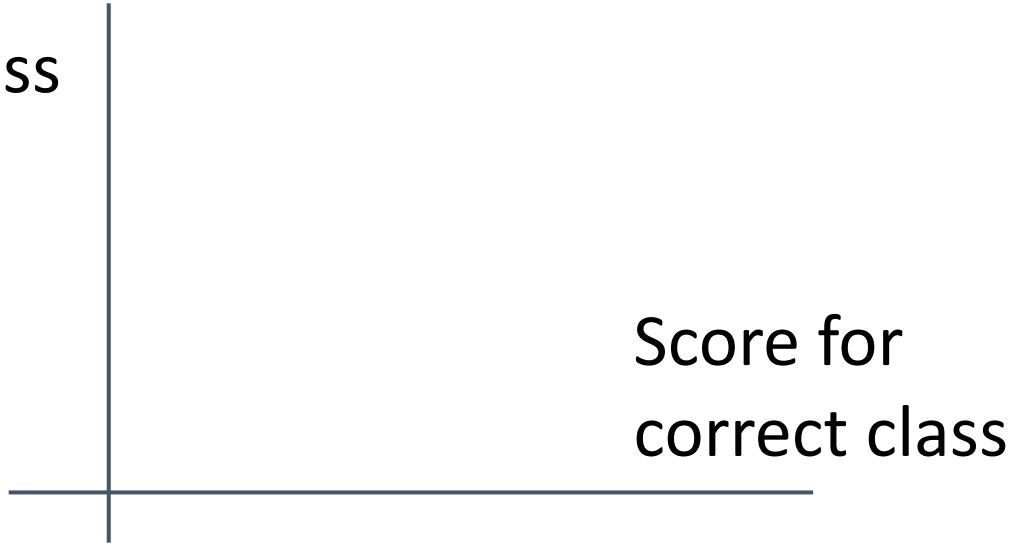
Bridge	**3.2**
Mountain	5.1
Coast	-1.7

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

A: -log(1/C)
log(10) ≈ 2.3

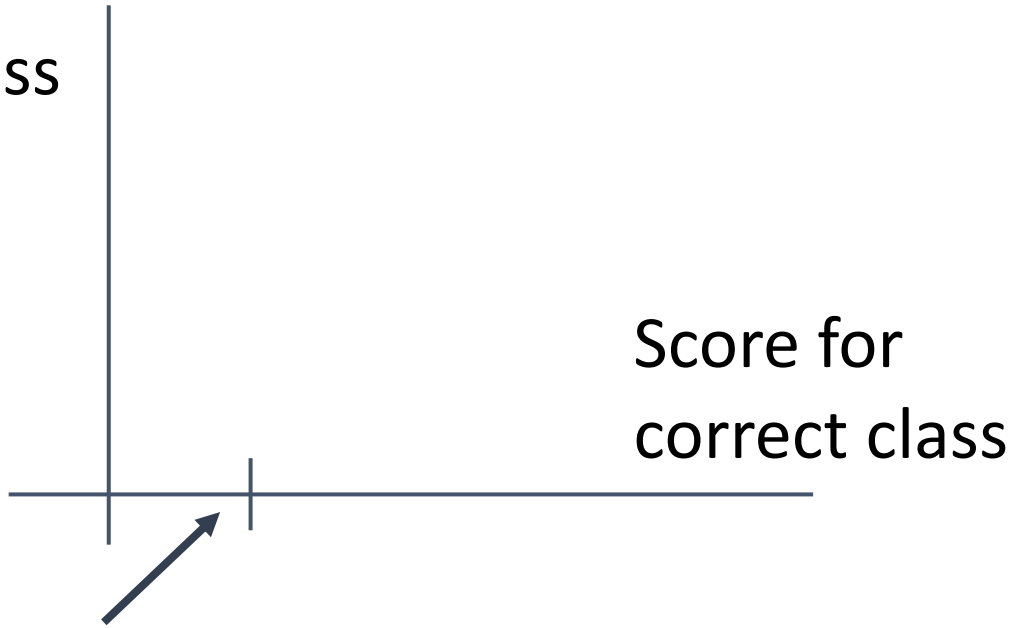
Multiclass SVM Loss

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



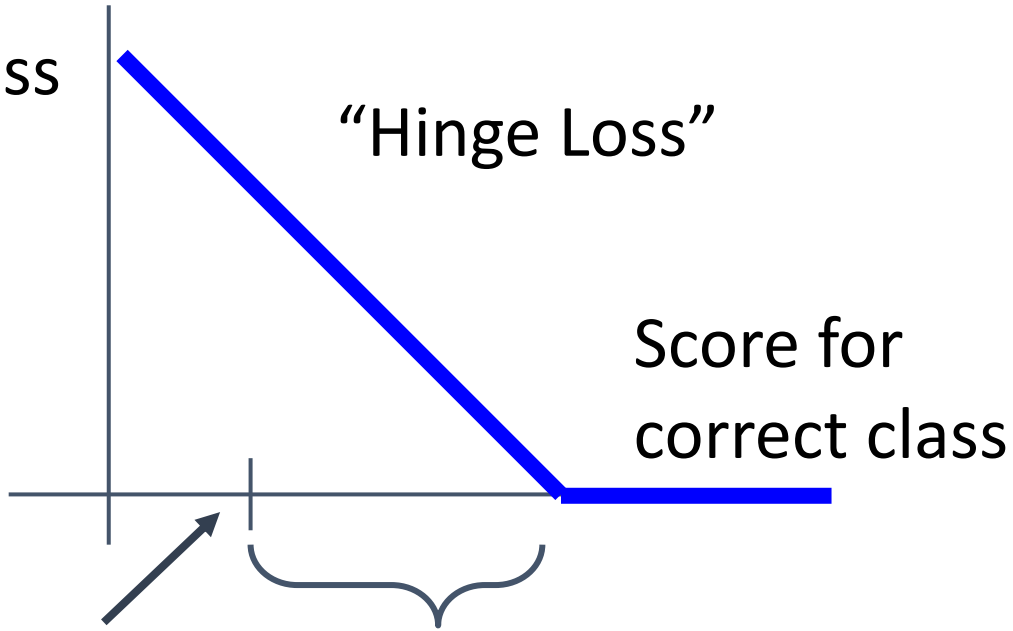
Multiclass SVM Loss

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



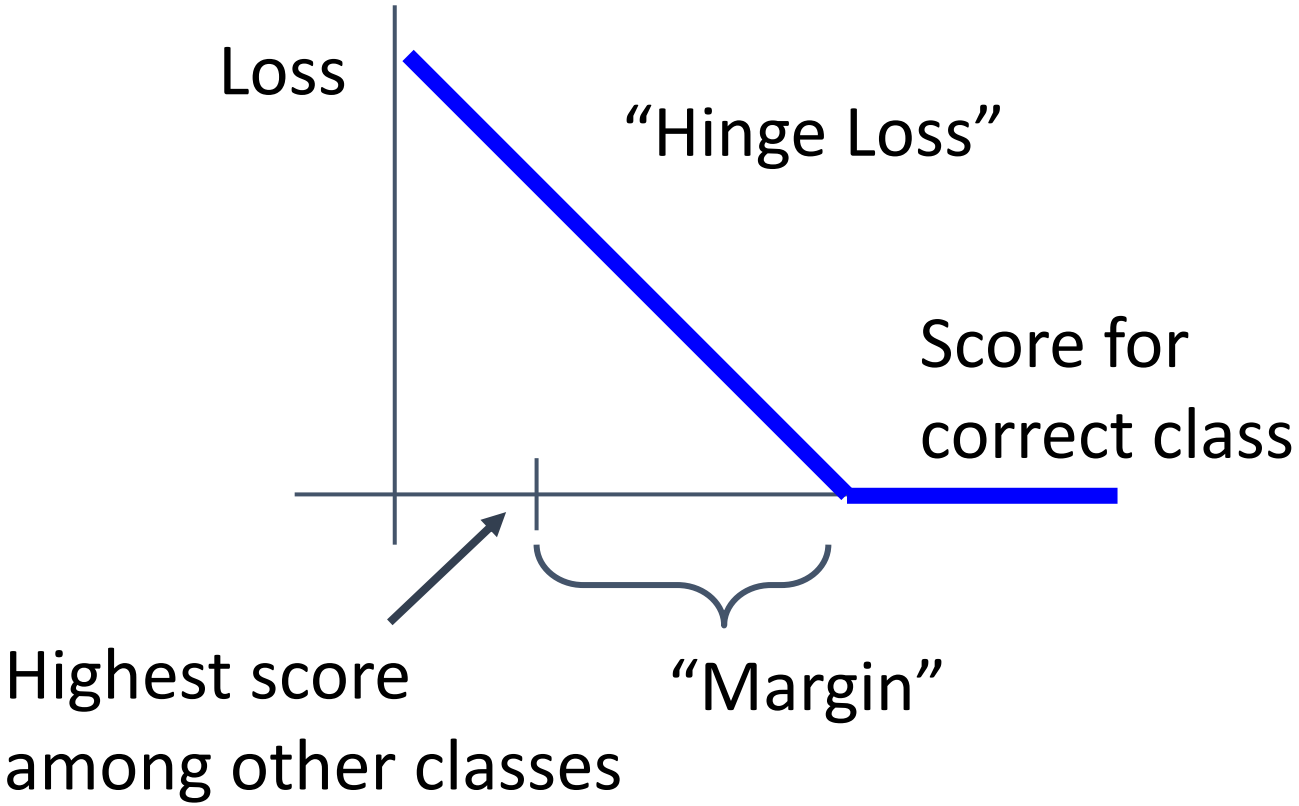
Multiclass SVM Loss

”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



Bridge	**3.2**	1.3	2.2
Mountain	5.1	**4.9**	2.5
Coast	-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



Bridge	**3.2**	1.3	2.2
Mountain	5.1	**4.9**	2.5
Coast	-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



Bridge	**3.2**	1.3	2.2
Mountain	5.1	**4.9**	2.5
Coast	-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



Bridge	**3.2**	1.3	2.2
Mountain	5.1	**4.9**	2.5
Coast	-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



Bridge	**3.2**	1.3	2.2
Mountain	5.1	**4.9**	2.5
Coast	-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

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?

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 $y_i = 0$

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

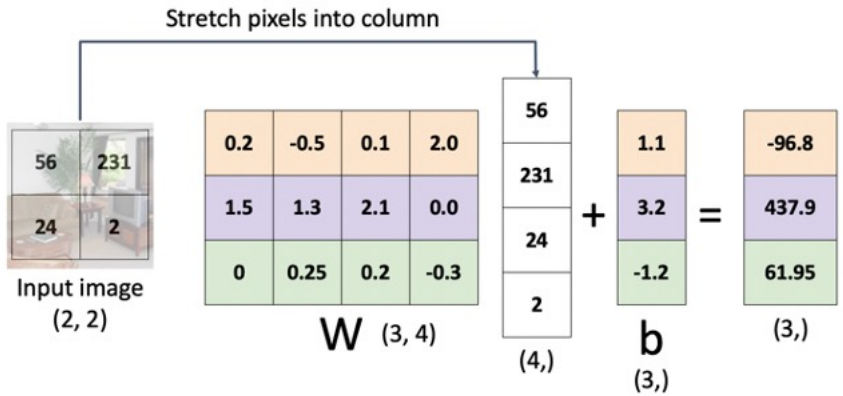
A: Cross-entropy loss will decrease, SVM loss still 0

Recap: Linear NN/Classifier: Three Viewpoints

Equation Viewpoint

f(x,W) = Wx

f(x,W) = Wx + b

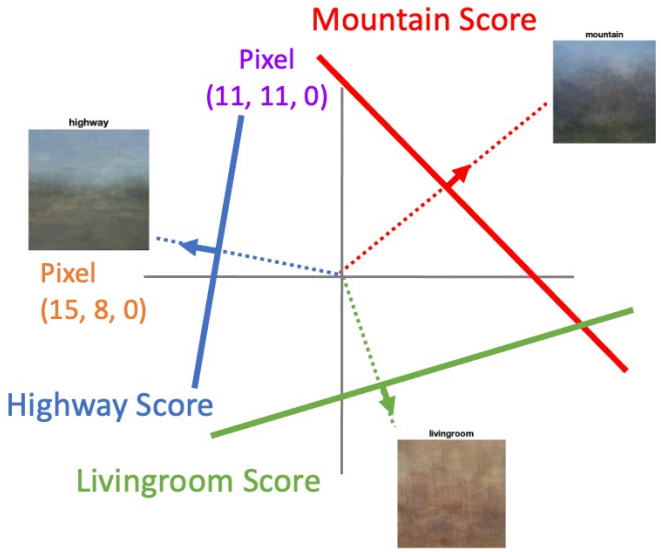


Visualization Viewpoint

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**:

Linear classifier

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

Linear regression loss:

$$L_i(W) = \frac{1}{2}(s_i - y_i)^2$$

Analytical solution:

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Softmax loss:

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

SVM loss:

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

But after we compute the loss,
how do we find the best W?

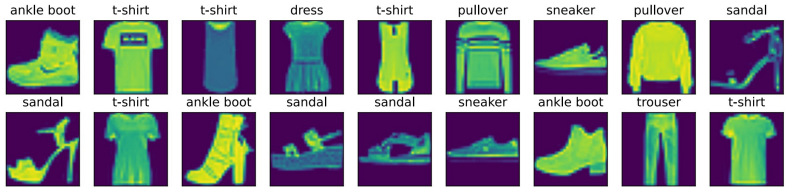
Reading and Practice

- Textbook: D2L: https://d2l.ai/d2l-en-pytorch.pdf
- Go through Chapter 3. Linear Neural Networks
- Go through the notebooks for that chapter: https://github.com/d2l-ai/d2l-pytorch-colab/tree/master/chapter_linear-classification

```
def show_images(imgs, num_rows, num_cols, titles=None, scale=1.5):  #@save
    """Plot a list of images."""
    figsize = (num_cols * scale, num_rows * scale)
    _, axes = d2l.plt.subplots(num_rows, num_cols, figsize=figsize)
    axes = axes.flatten()
    for i, (ax, img) in enumerate(zip(axes, imgs)):
        if torch.is_tensor(img):
            # Tensor Image
            ax.imshow(img.numpy())
        else:
            # PIL Image
            ax.imshow(img)
        ax.axes.get_xaxis().set_visible(False)
        ax.axes.get_yaxis().set_visible(False)
        if titles:
            ax.set_title(titles[i])
    return axes
```

Here are the images and their corresponding labels (in text) for the first few examples in the training dataset.

```
X, y = next(iter(data.DataLoader(mnist_train, batch_size=18)))
show_images(X.reshape(18, 28, 28), 2, 9, titles=get_fashion_mnist_labels(y));
```

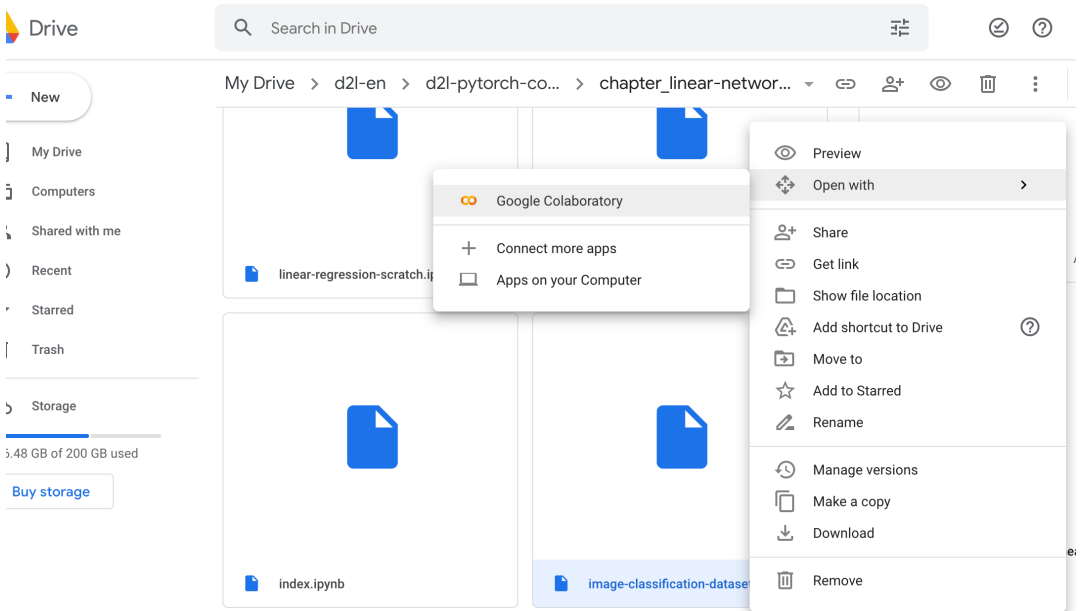


Two ways of running D2L Jupyter notebook

- Run on local machine/desktop/labtop
- https://d2l.ai/chapter_installation/index.html

Two ways of running D2L Jupyter notebook

- Run one Google Colab:
  - Download https://github.com/d2l-ai/d2l-pytorch-colab
  - Upload the notebook files to your Google Drive
  - In Google Drive d2l-pytorch-colab folder, click the notebook, open with Google Colab



Next time:
Optimization Algorithms