DeepRob

Lecture 2
Linear Classifiers

University of Michigan | Department of Robotics

Project 0

- Instructions and code available on the website
  - Here: deeprob.org/w24/projects/project0/

- **Due Thursday! January 18th, 11:59 PM EST**

- **Everyone granted 3 *total* late tokens for semester**
  - A penalty-free 24 hour extension

Project 0 Suggestions

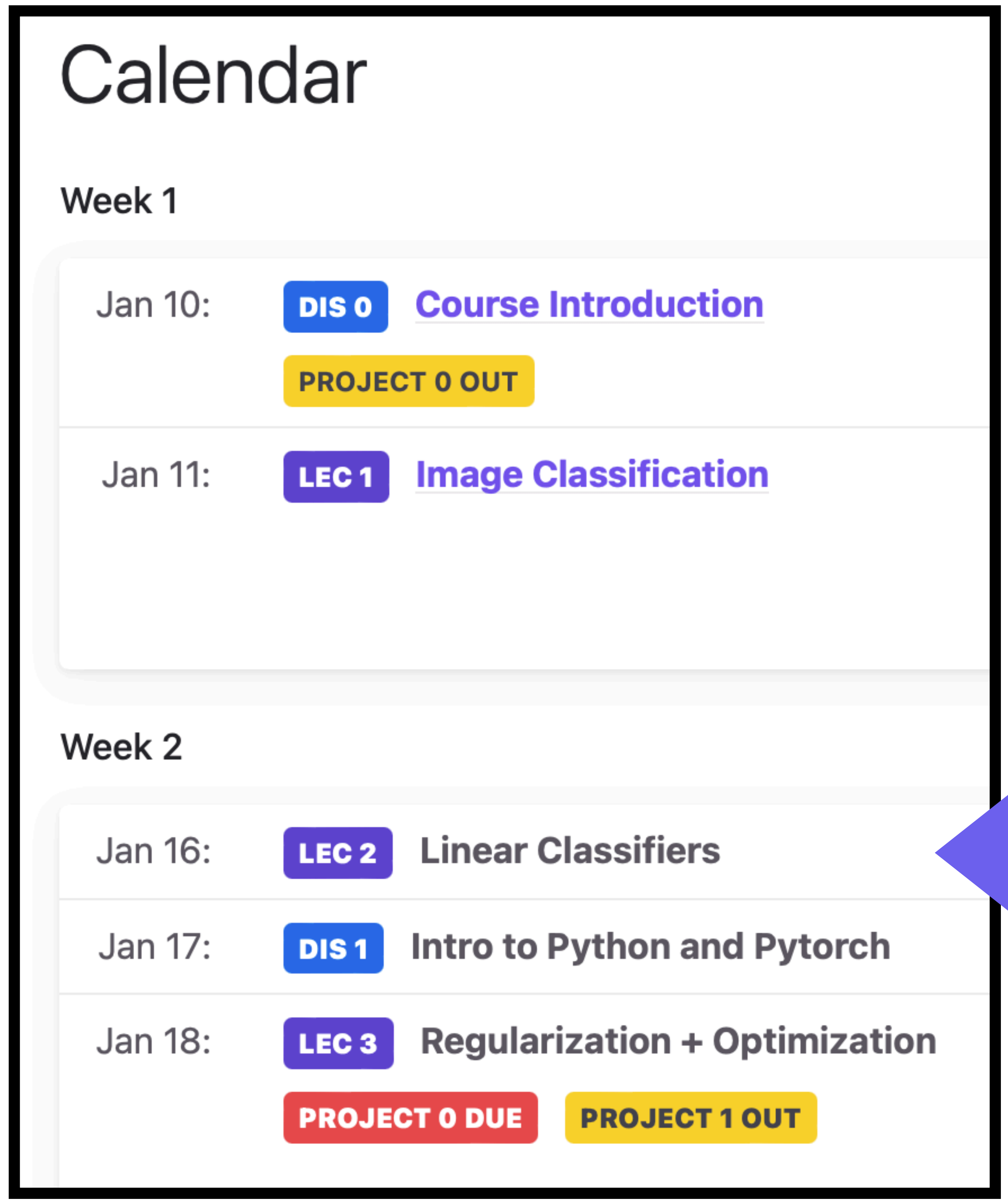
- If you choose to develop locally
  - **PyTorch Version 2.1.0**

- Ensure you save your notebook file before uploading submission

- Close any Colab notebooks not in use to avoid usage limits

Project 1 Upcoming

- Instructions and code will be available on the website before Thursday's lecture
- Classification using K-Nearest Neighbors and Linear Models



We're here!

Course Resources

- Everyone should have access to
  - [Course Website]
  - [Piazza]
  - [Gradescope]

- If not, please contact Anthony!

Enrollment

- Additional class permissions being issued
  - Both sections (498 & 599)
- *Room capacity is 74*
- **[If you are waitlisted and want to take the class, please email Xiaoxiao & Anthony!]**

Recap: Image Classification—A Core Computer Vision Task

Input: image

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

Chocolate Pretzels

Granola Bar

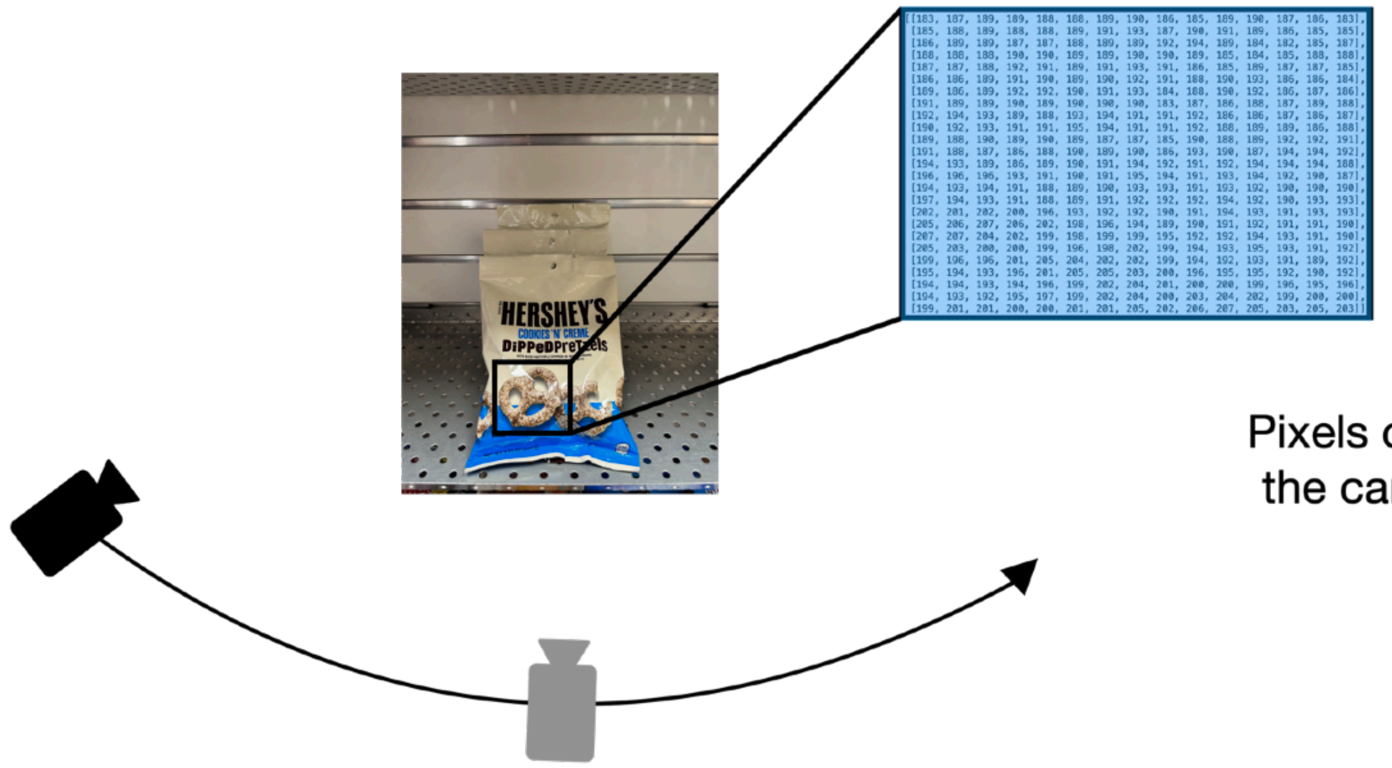
Potato Chips

Water Bottle

Popcorn

Image Classification Challenges

Viewpoint Variation & Semantic Gap



Pixels change when the camera moves

Illumination Changes

Milk Chocolate | White Chocolate | Cookies N' Creme | Peanut Butter | Ambiguous Category

Intraclass Variation

Recap: Machine Learning—Data-Driven Approach

1. Collect a dataset of images and labels

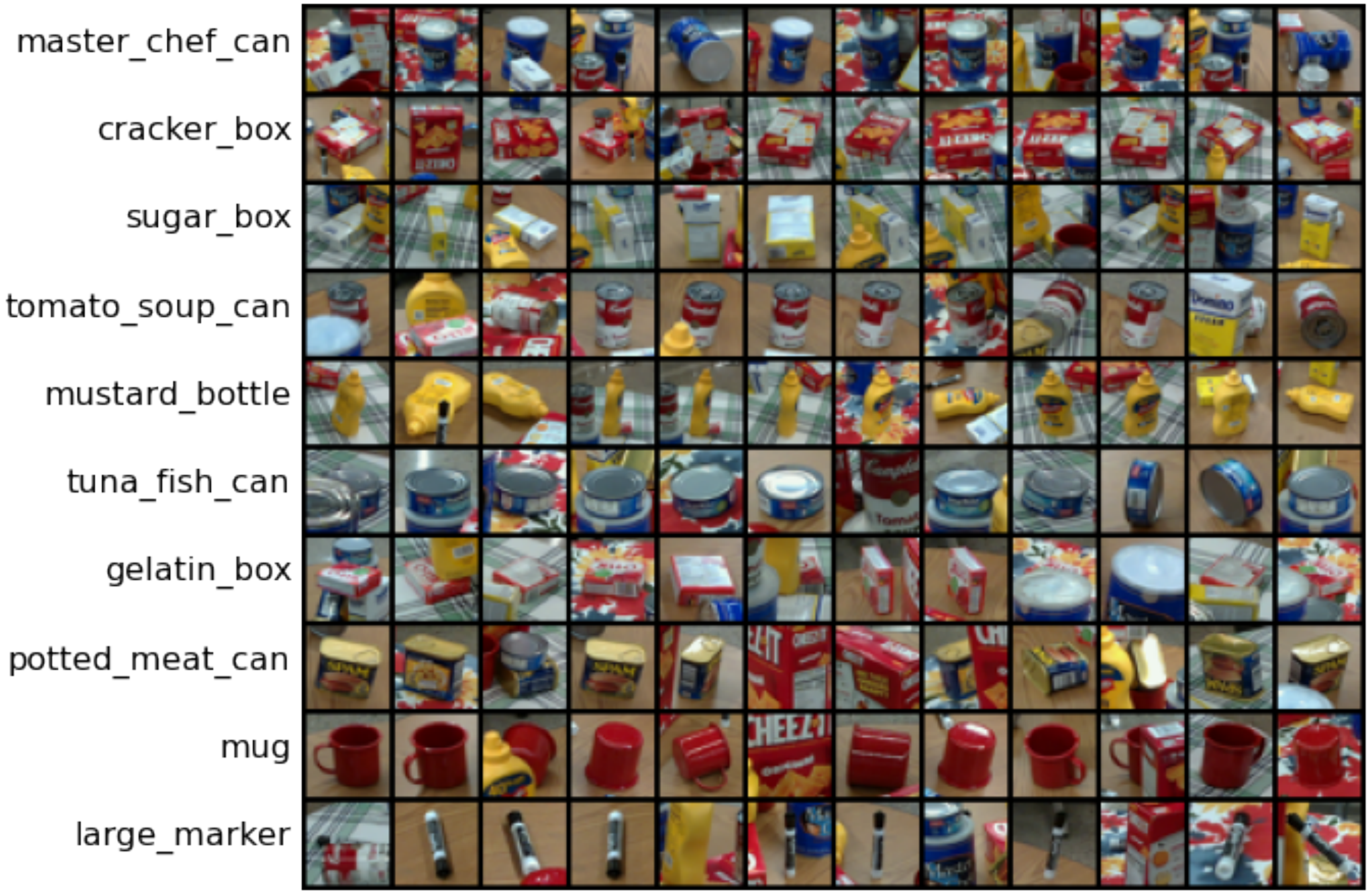
2. Use Machine Learning to train a classifier

3. Evaluate the classifier on new images

```python
def train(images, labels):
  # Machine learning!
  return model
```

```python
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training s



First Classifier—Nearest Neighbor

```
def train(images, labels):
  # Machine learning!
  return model
```

Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Predict the label of the most similar training image

Nearest Neighbor Classifier

```
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Q: With N examples how fast is training?

A: O(1)

Q: With N examples how fast is testing?

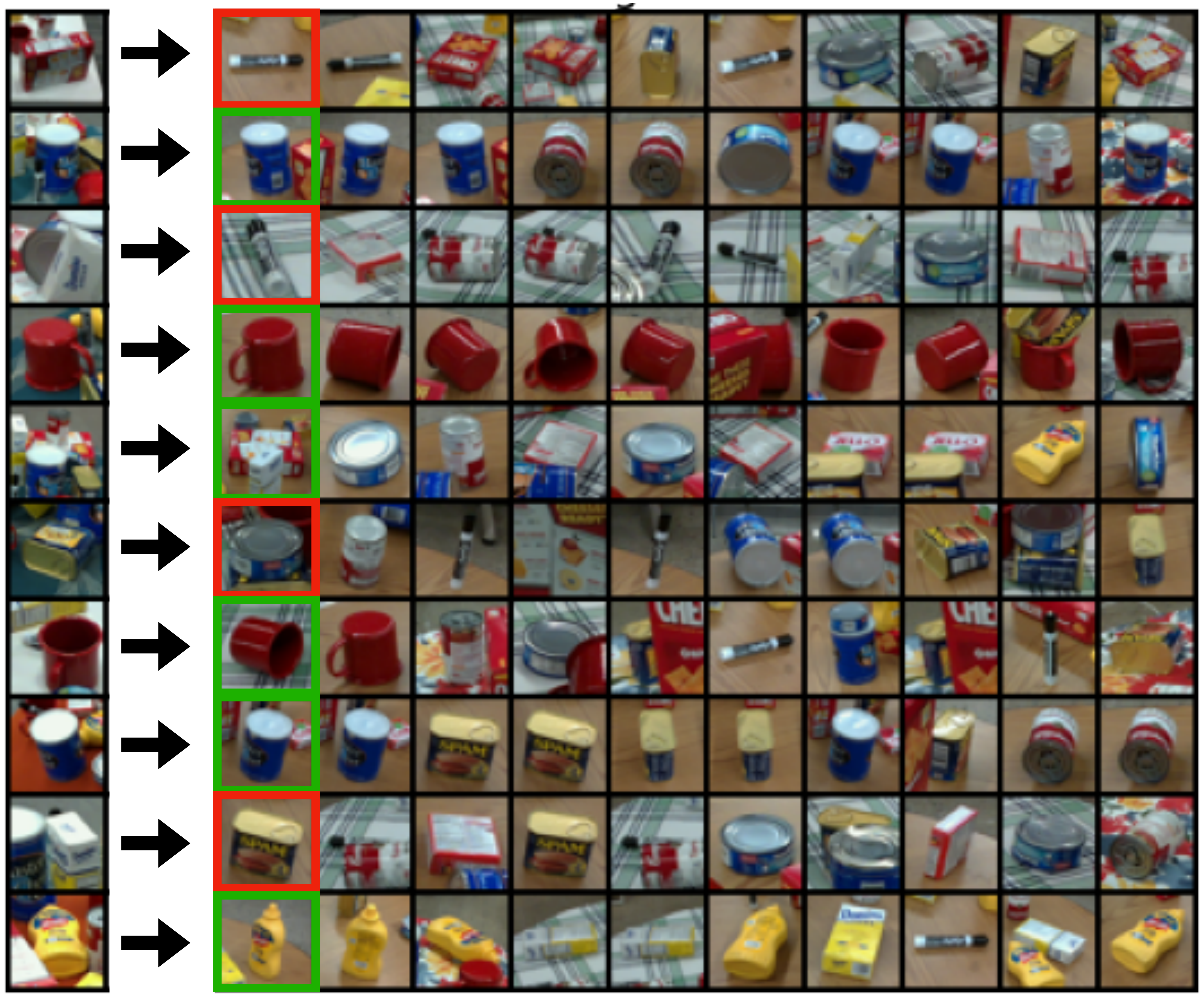
A: O(N)

This is a problem: we can train slow offline but need fast testing!

What does this look like?
Examples on the PROPS Dataset

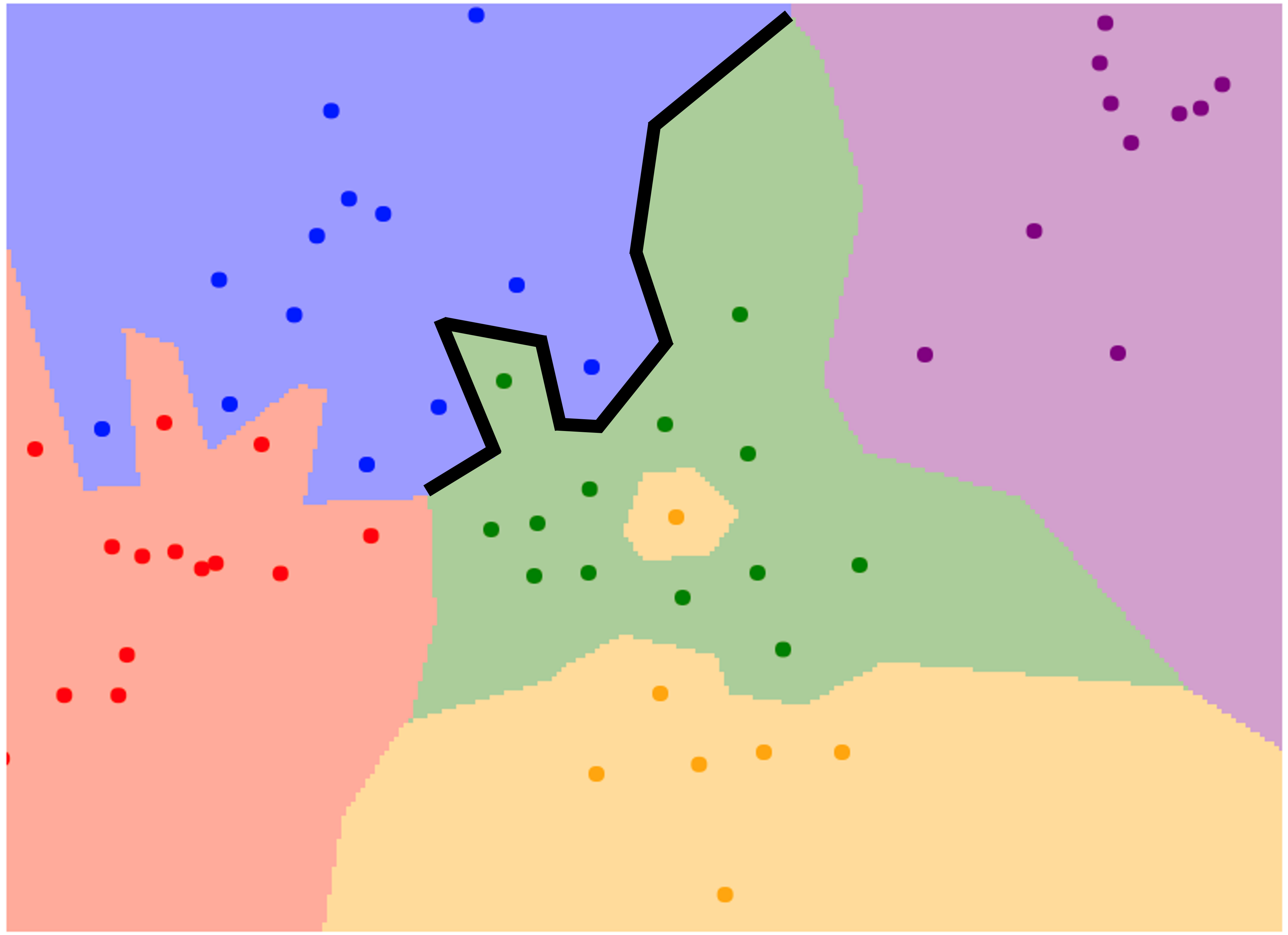
10 Nearest Neighbors from Training Set

Test Images
Unseen During Training

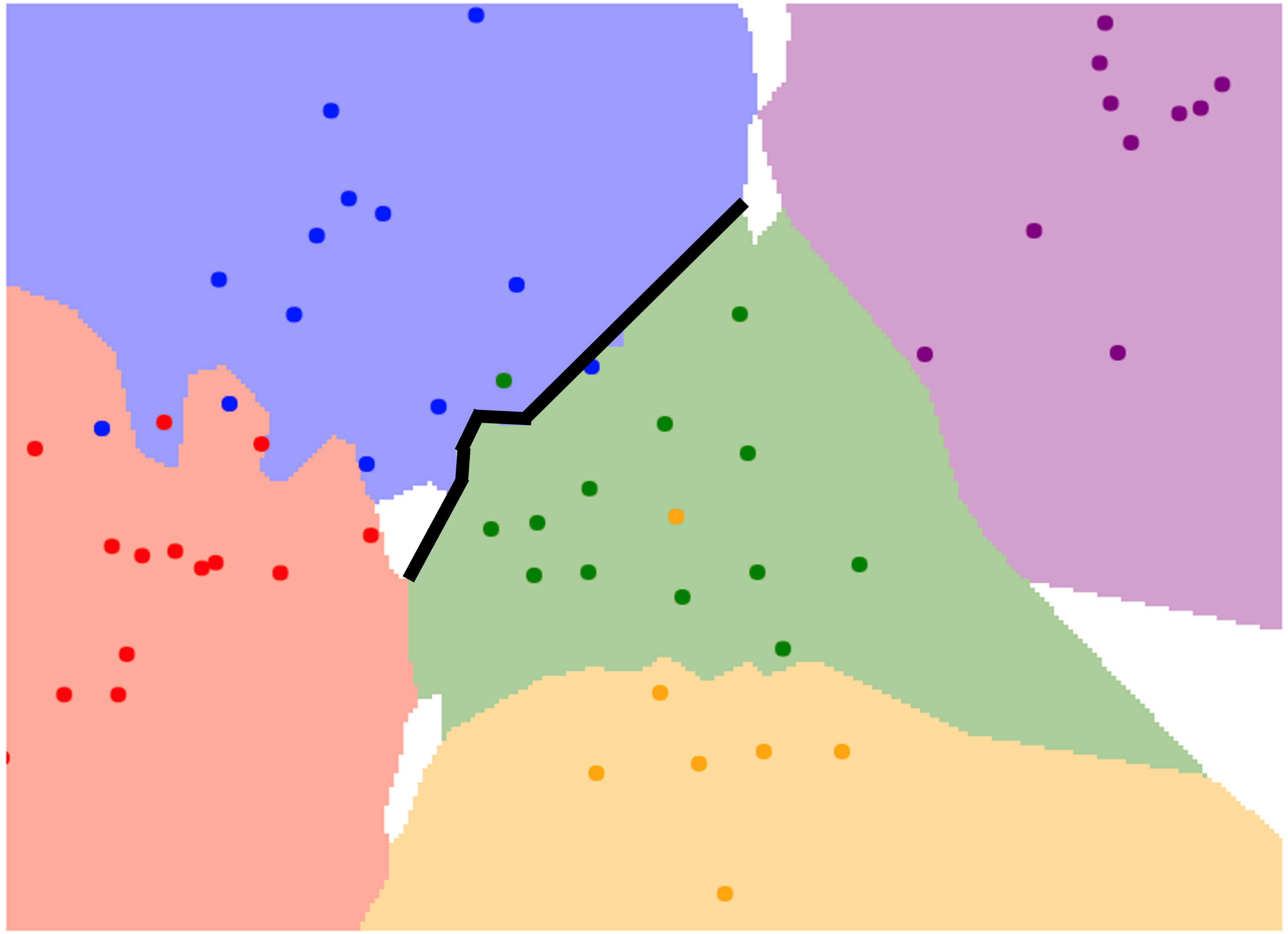


K-Nearest Neighbors Decision Boundaries

$K = 1$



$K = 3$



Using more neighbors helps smooth out rough decision boundaries

Hyperparameters

What is the best value of K to use?

What is the best **distance metric** to use?

Hyperparameters

What is the best value of K to use?

What is the best **distance metric** to use?

These are examples of **hyperparemeters:**
choices about our learning algorithm that we don't learn from the training data
Instead we set them at the start of the learning process

Hyperparameters

What is the best value of K to use?

What is the best **distance metric** to use?

These are examples of **hyperparameters:**
choices about our learning algorithm that we don't learn from the training data
Instead we set them at the start of the learning process

Very problem-dependent.
In general need to try them all and observe what works best for our data.

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

Your Dataset

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

train	test

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train	test

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train	test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Setting Hyperparameters

Your Dataset

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but (unfortunately) not used too frequently in deep learning

K-Nearest Neighbors with NN Features Works Well

Devlin et al., “Exploring Nearest Neighbor Approaches for Image Captioning”, 2015.

Summary of Image Classification and K-NN

In **image classification** we start with a training set of images and labels, and must predict labels for a test set

Image classification is challenging due to the **semantic gap**: we need invariance to occlusion, deformation, lighting, sensor variation, etc.

Image classification is a **building block** for other vision tasks

The **K-Nearest Neighbors** classifier predicts labels from nearest training samples

Distance metric and K are **hyperparameters**

Choose hyper parameters using the **validation set**; only run on the test set once at the very end!

Linear Classifiers

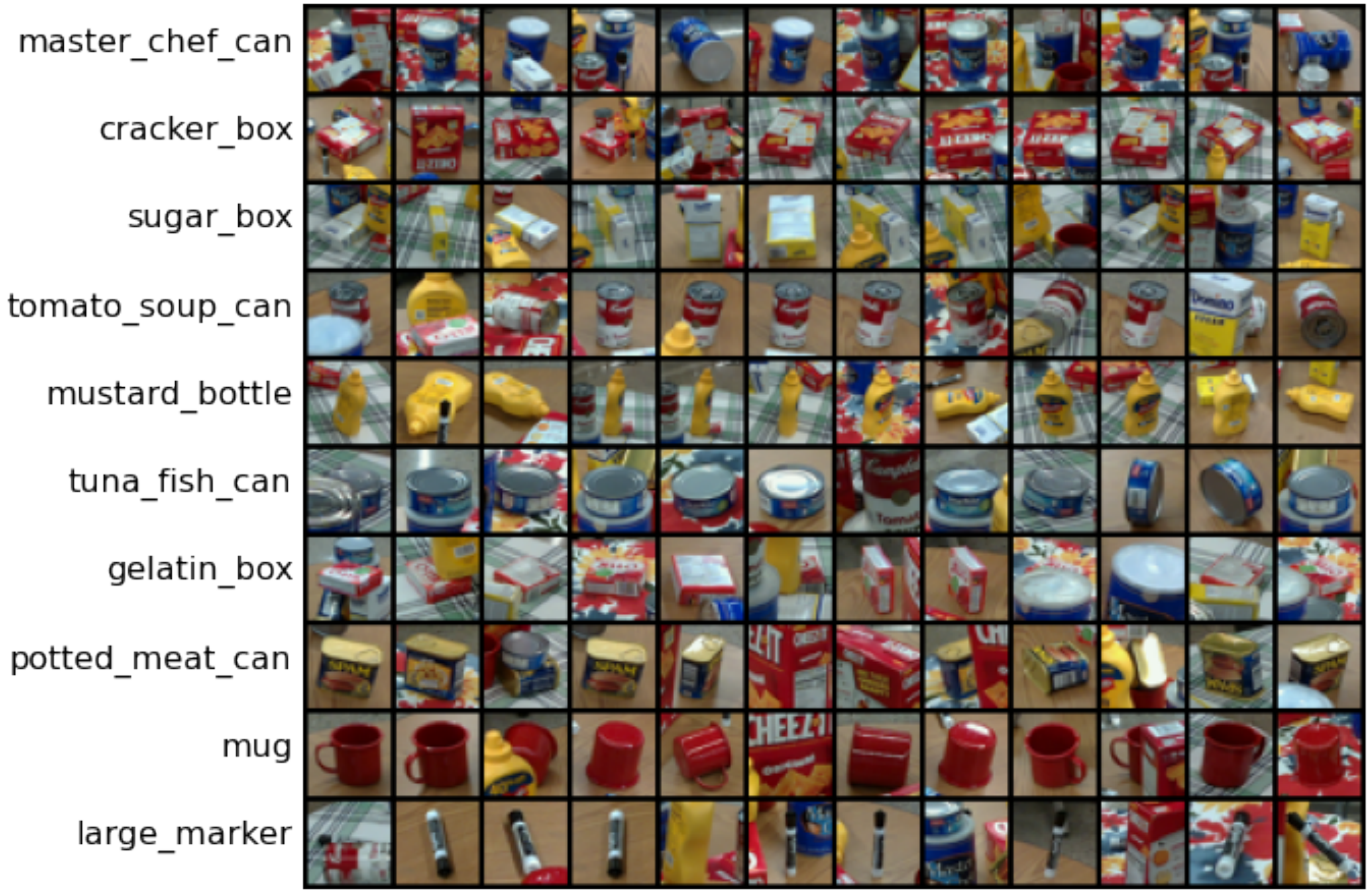
Building Block of Neural Networks

Linear classifiers

[This image](#) is [CC0 1.0](#) public domain

Recall PROPS

Progress **R**obot **O**bject **P**erception **S**amples **D**ataset



Chen et al., "ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception", IROS, 2022.

10 classes

32x32 RGB images

50k training images (5k per class)

10k test images (1k per class)

Parametric Approach

Image

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

f(x,W)

10 numbers giving
class scores

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

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

Image

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

→ f(**x**,**W**) → **10** numbers giving class scores

↑

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)

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

W

parameters
or weights

Example for 2x2 Image, 3 classes (crackers/mug/sugar)

Stretch pixels into column

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

Input image
(2, 2)

56
231
24
2

(4,)

Example for 2x2 Image, 3 classes (crackers/mug/sugar)

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)

$$\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}$$

W (3, 4)

(4,)

b
(3,)

(3,)

Linear Classifier—Bias Trick

Stretch pixels into column

Input image
(2, 2)

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

W (3, 5) (5,) (3,)

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

Linear Classifier—Predictions are Linear

$f(x, W) = Wx$ (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

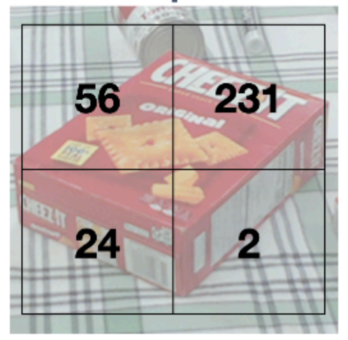
-48.4
218.9
31.0

Interpreting a Linear Classifier

Algebraic Viewpoint

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

Stretch pixels into column



Input image
(2, 2)

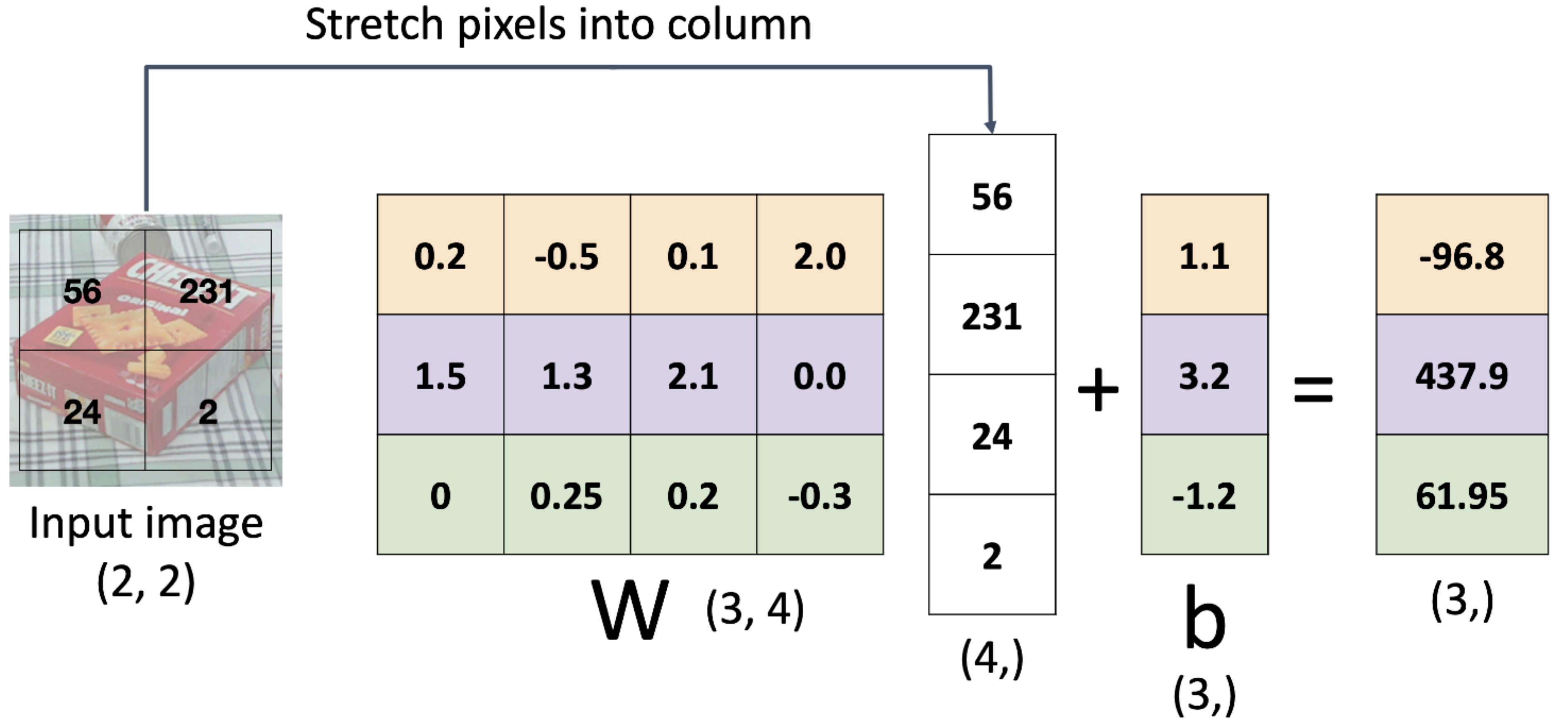
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W\ (3,\ 4)} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{(4,)} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b\ (3,)} = \underbrace{\begin{bmatrix}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

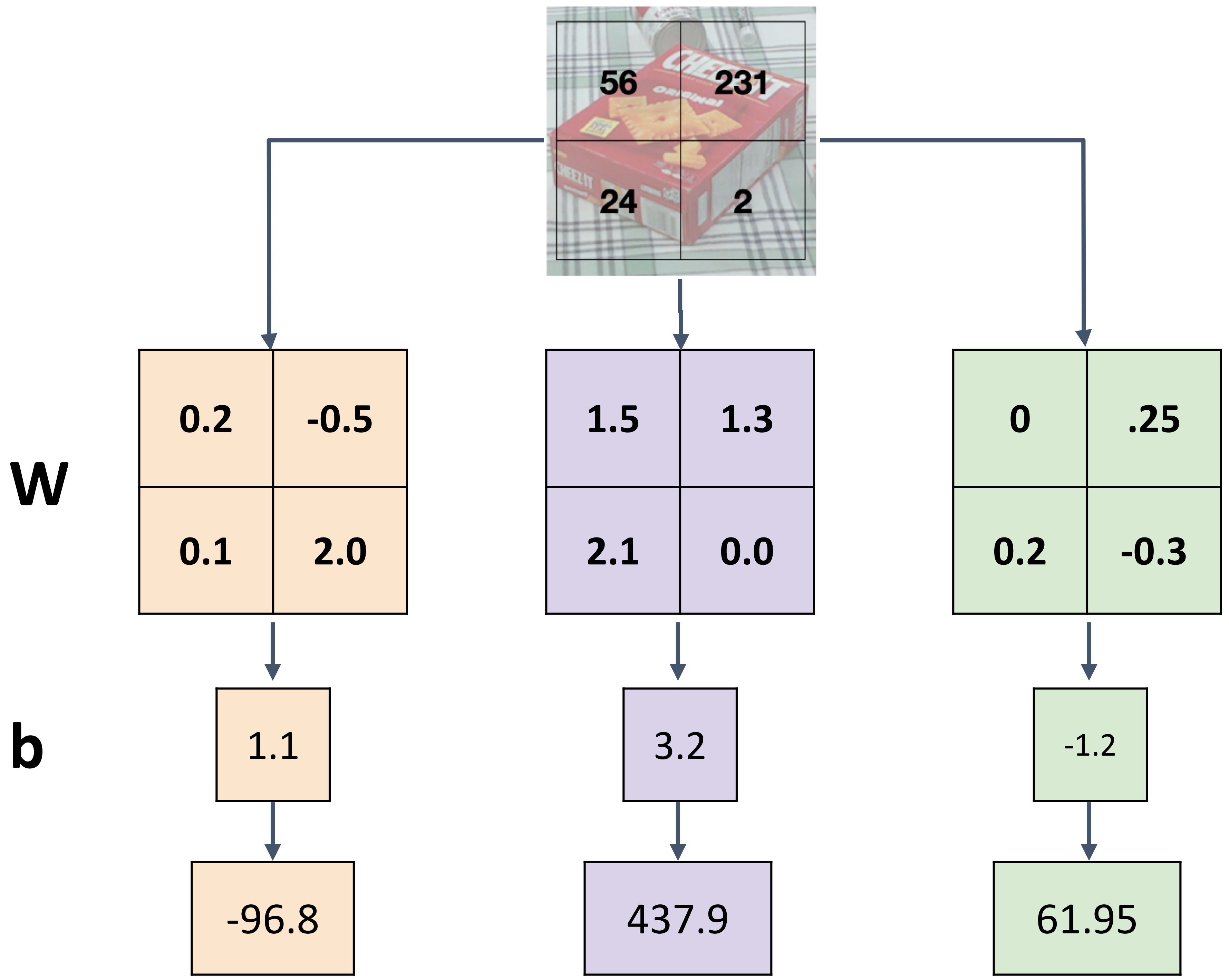
Interpreting a Linear Classifier

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

Algebraic Viewpoint

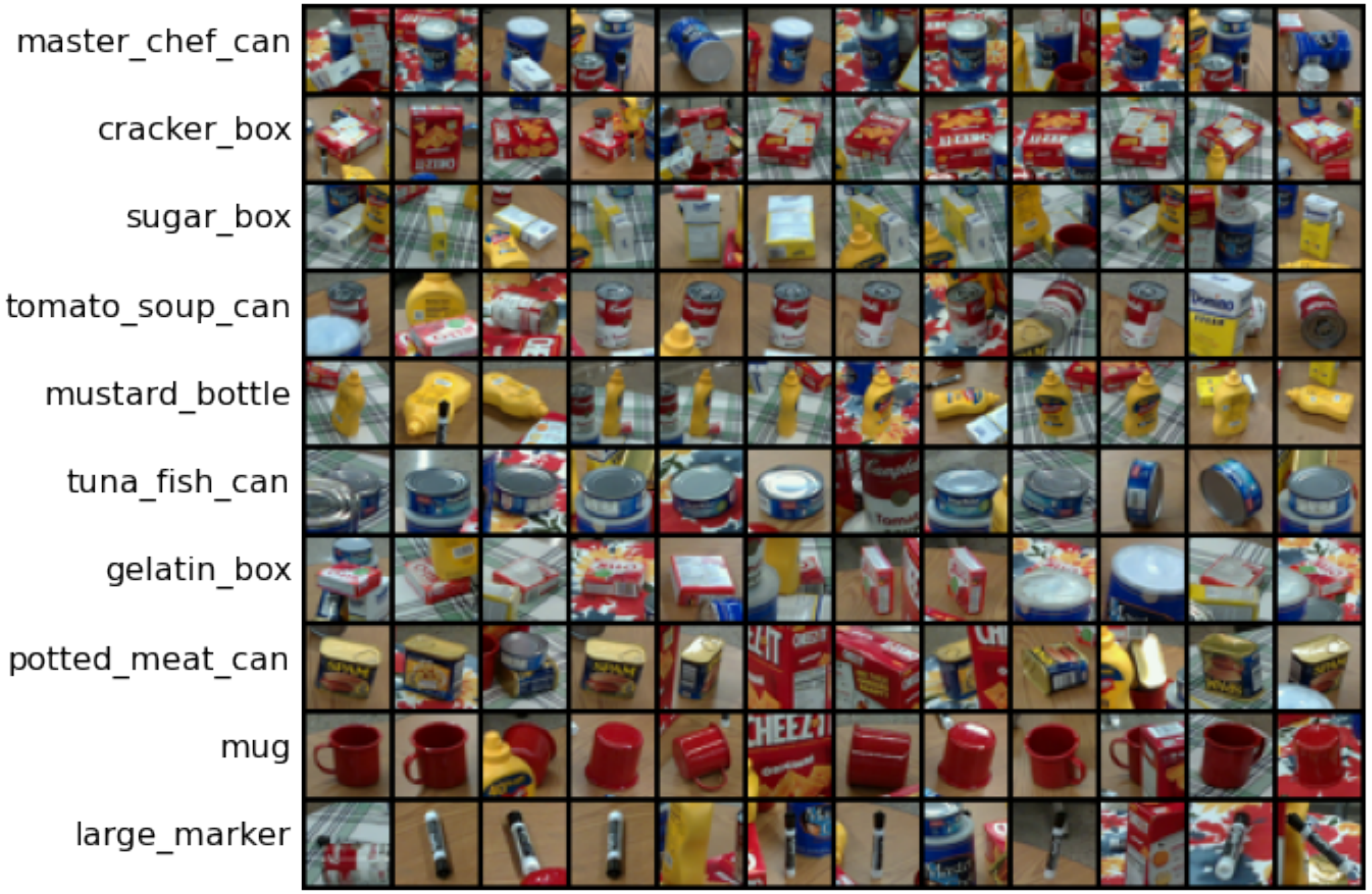
$$f(x,W) = Wx + b$$

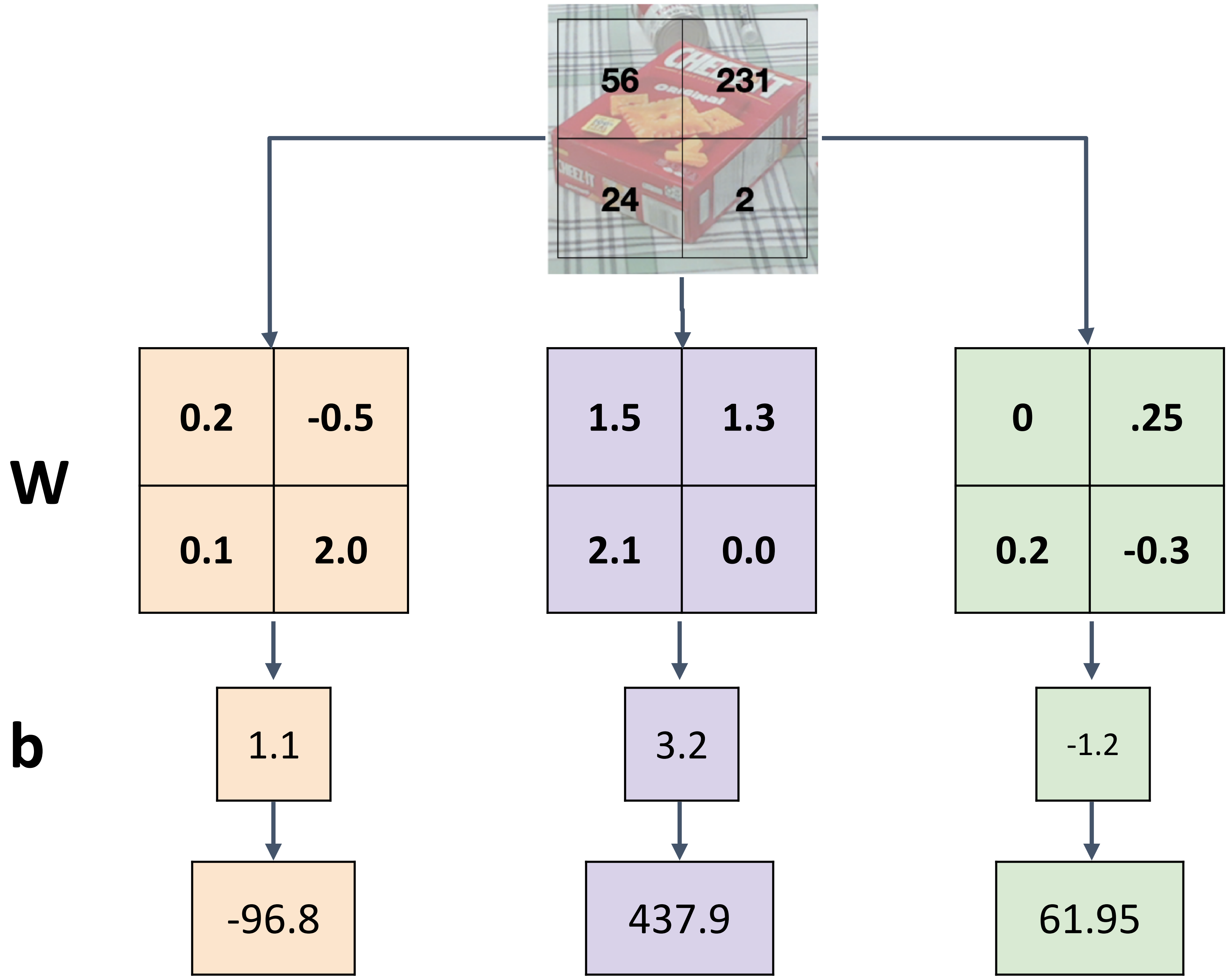




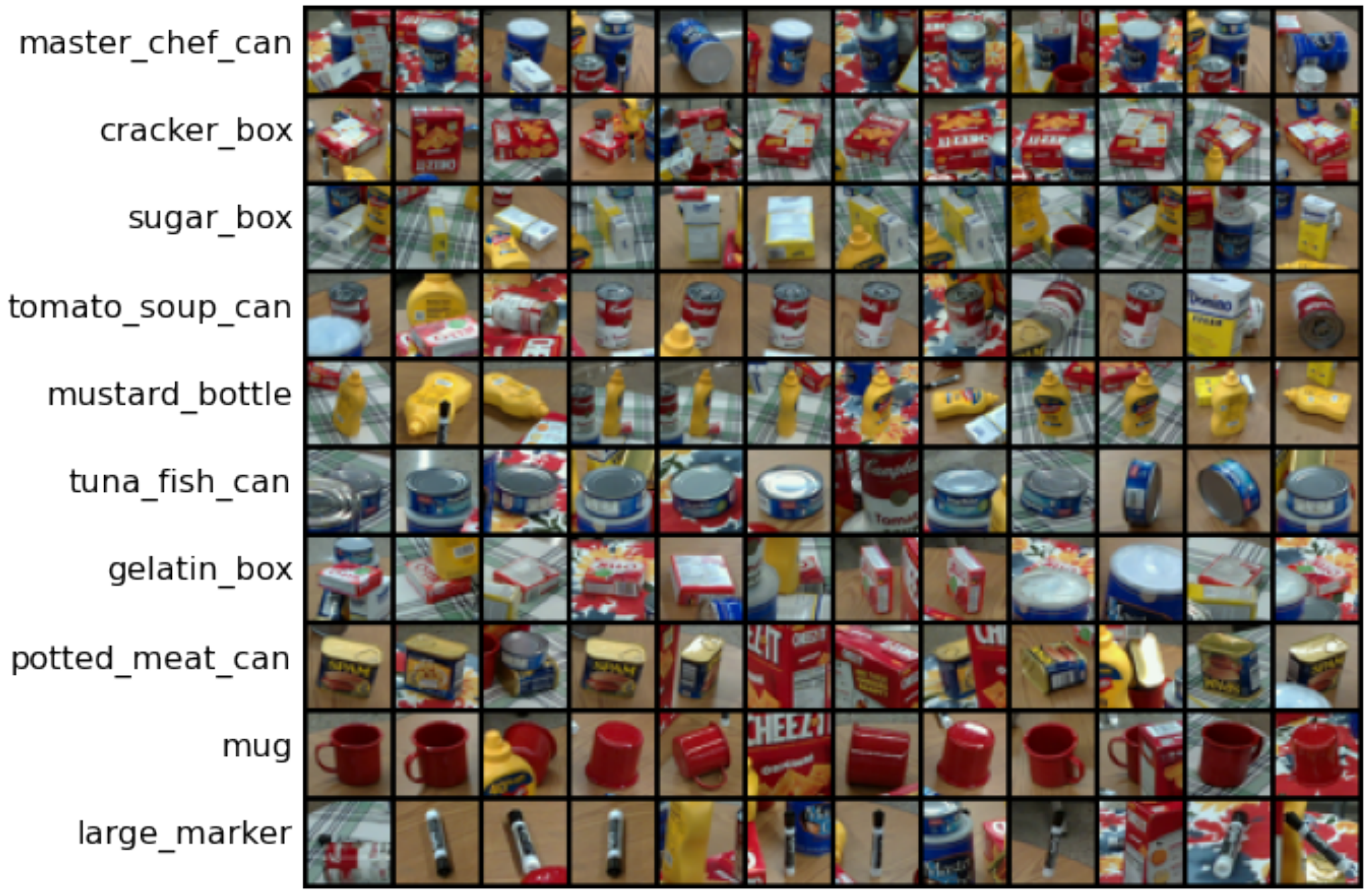
Interpreting a Linear Classifier

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

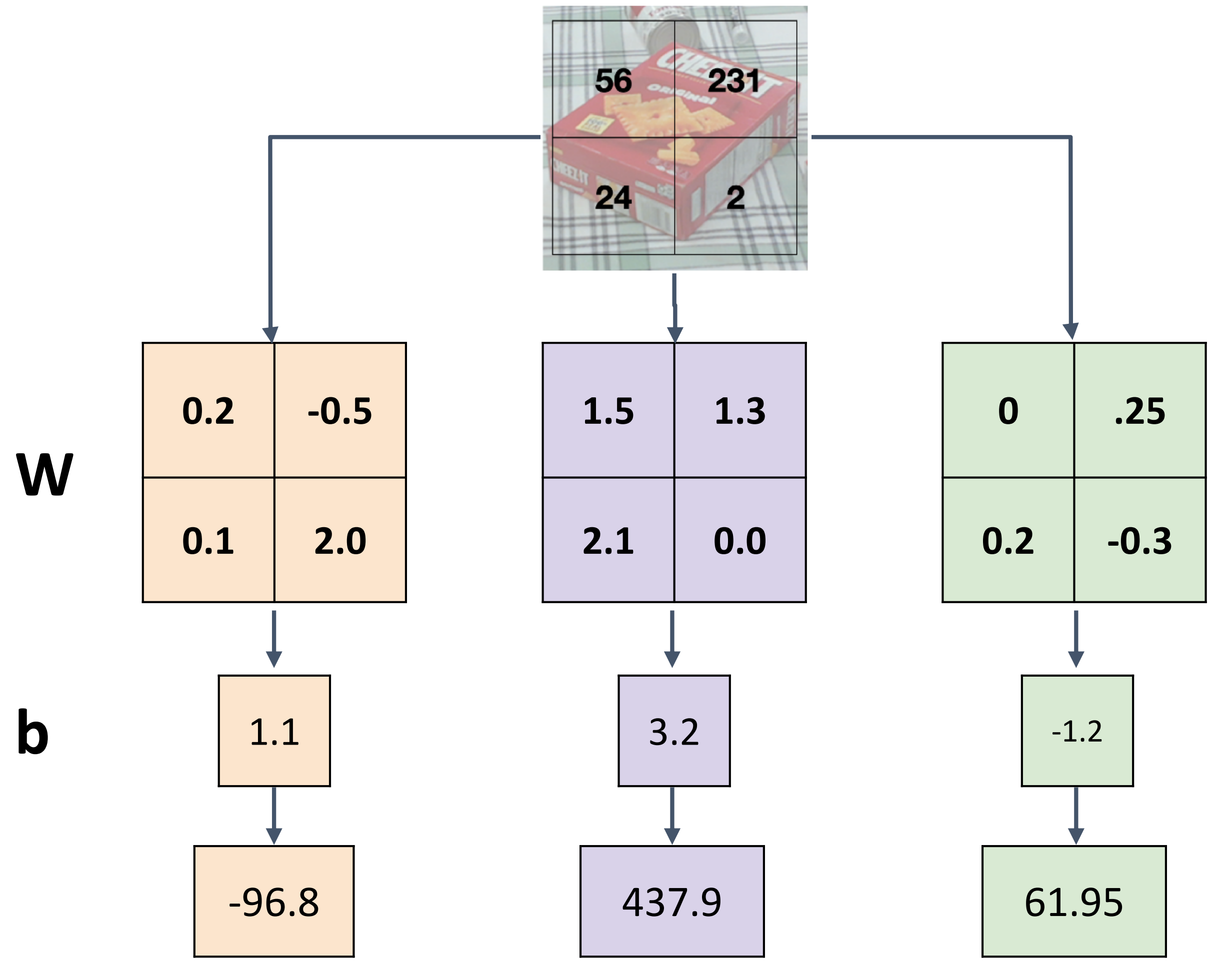


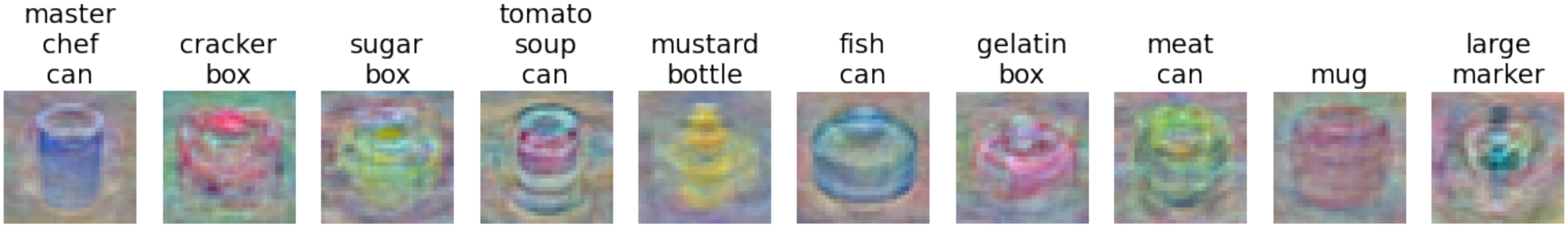


Interpreting a Linear Classifier



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

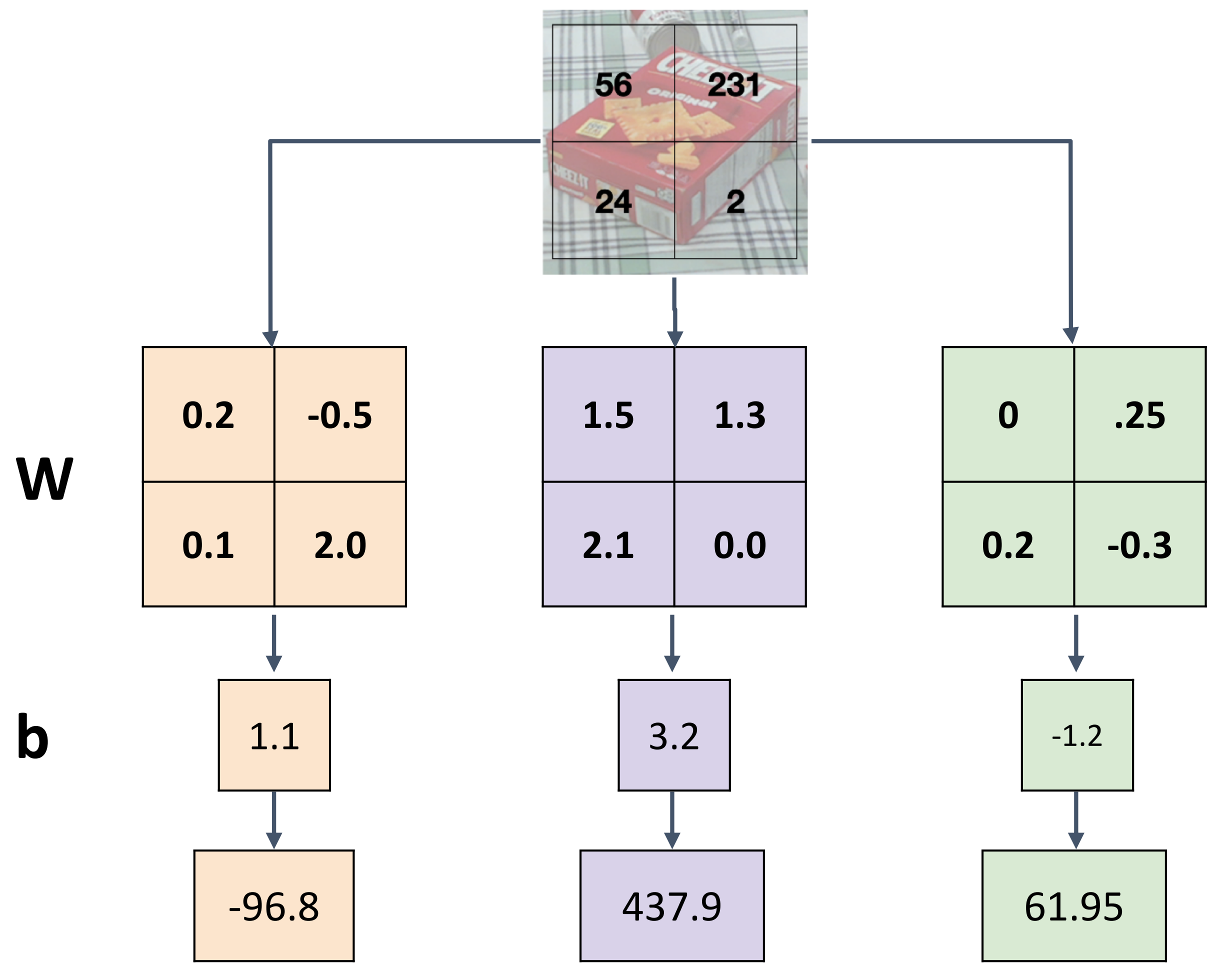


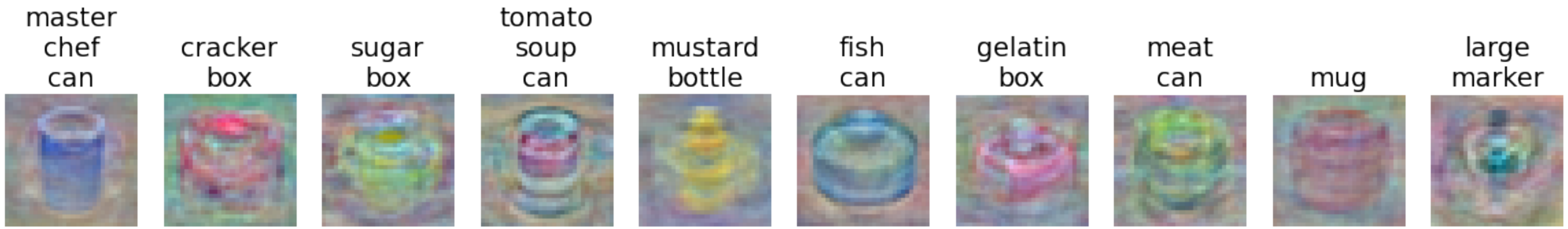


Interpreting a Linear Classifier—Visual Viewpoint

Linear classifier has one "template" per category

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





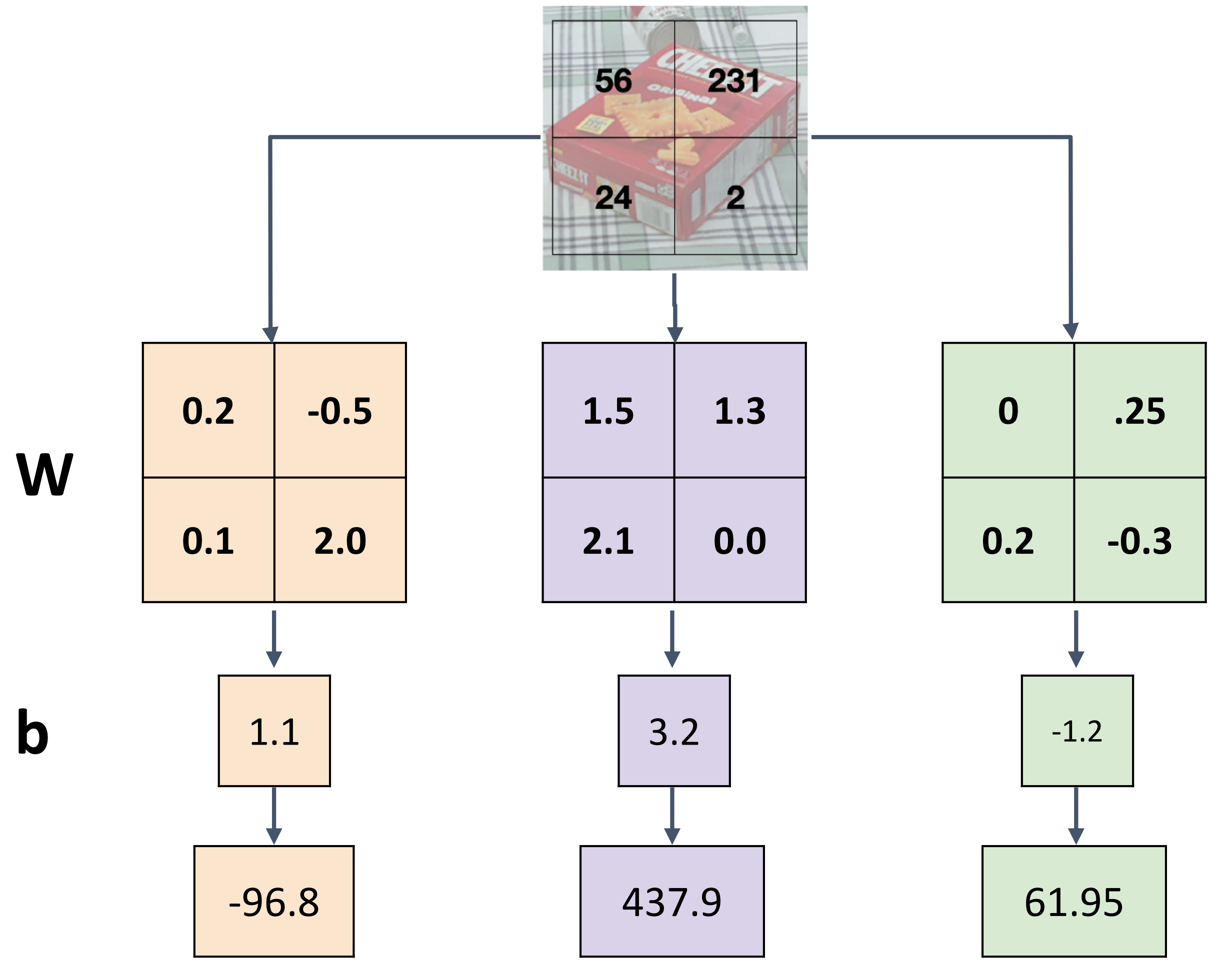
Interpreting a Linear Classifier—Visual Viewpoint

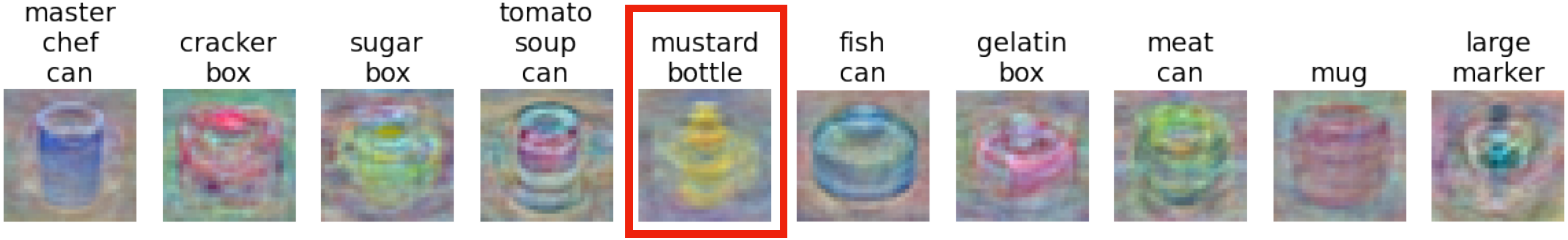
Instead of stretching pixels into columns, we can equivalently stretch rows of W into images!

Linear classifier has one "template" per category

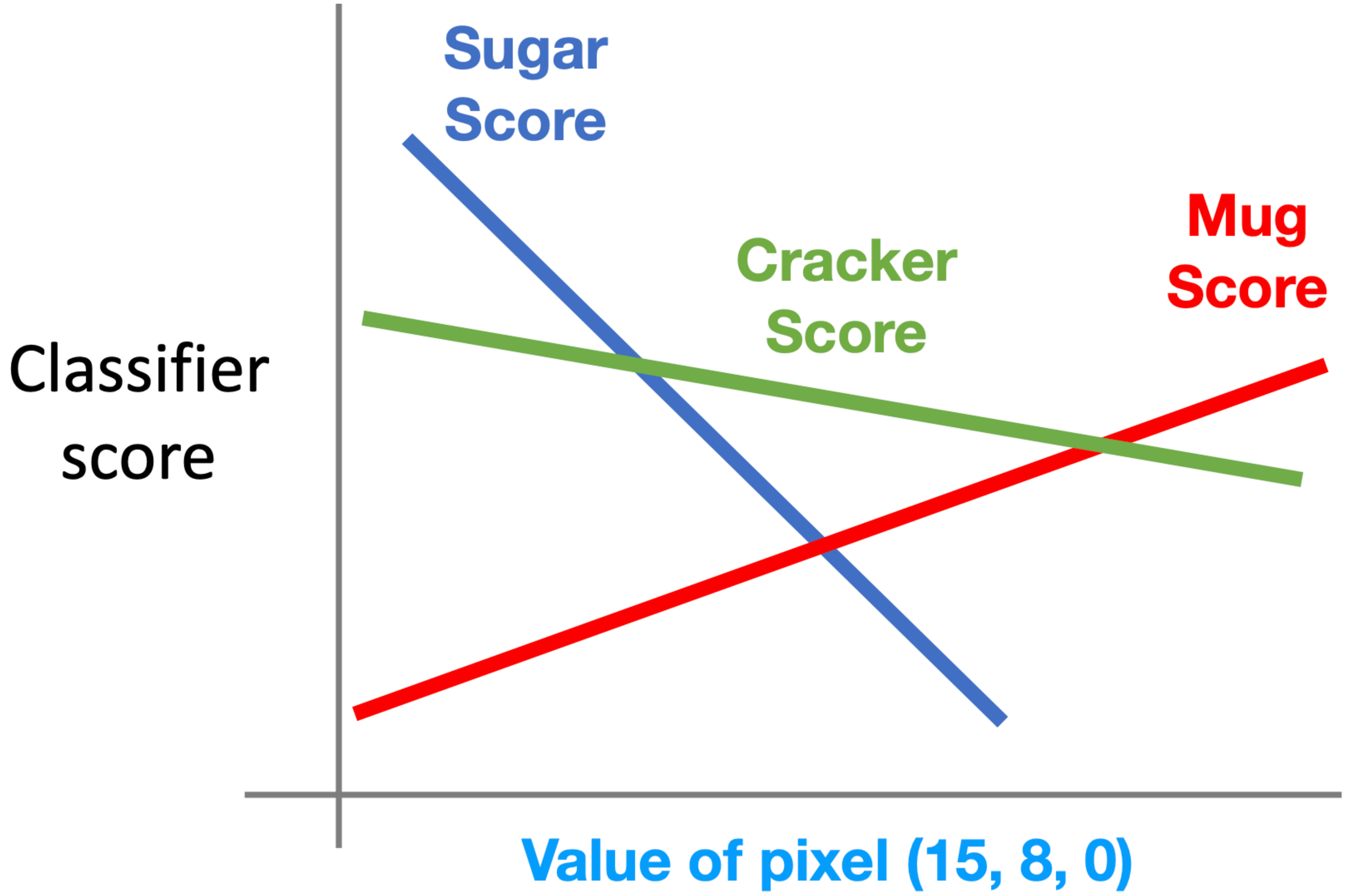
A single template cannot capture multiple modes of the data

e.g. mustard bottles can rotate





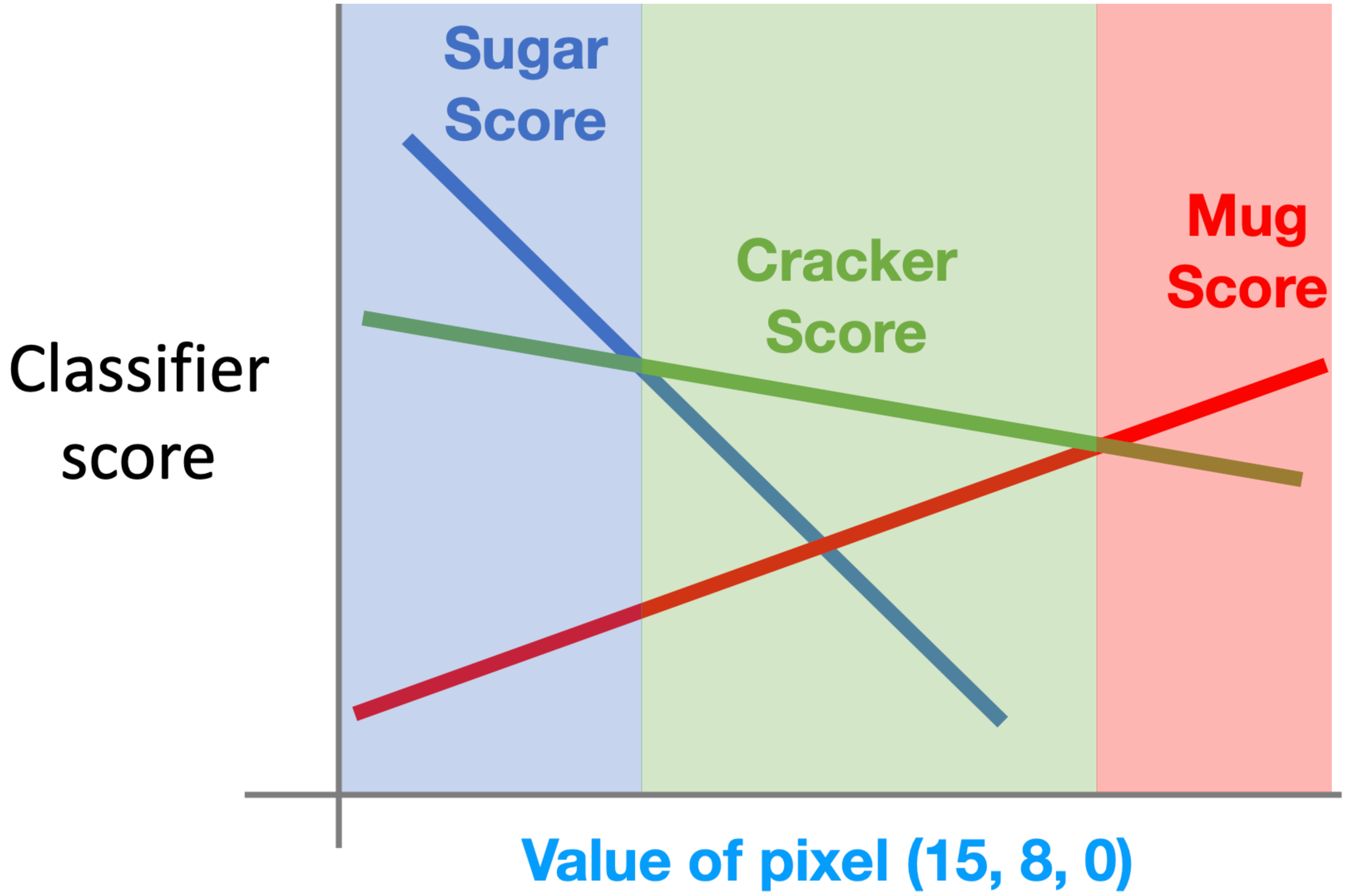
Interpreting a Linear Classifier—Geometric Viewpoint



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

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

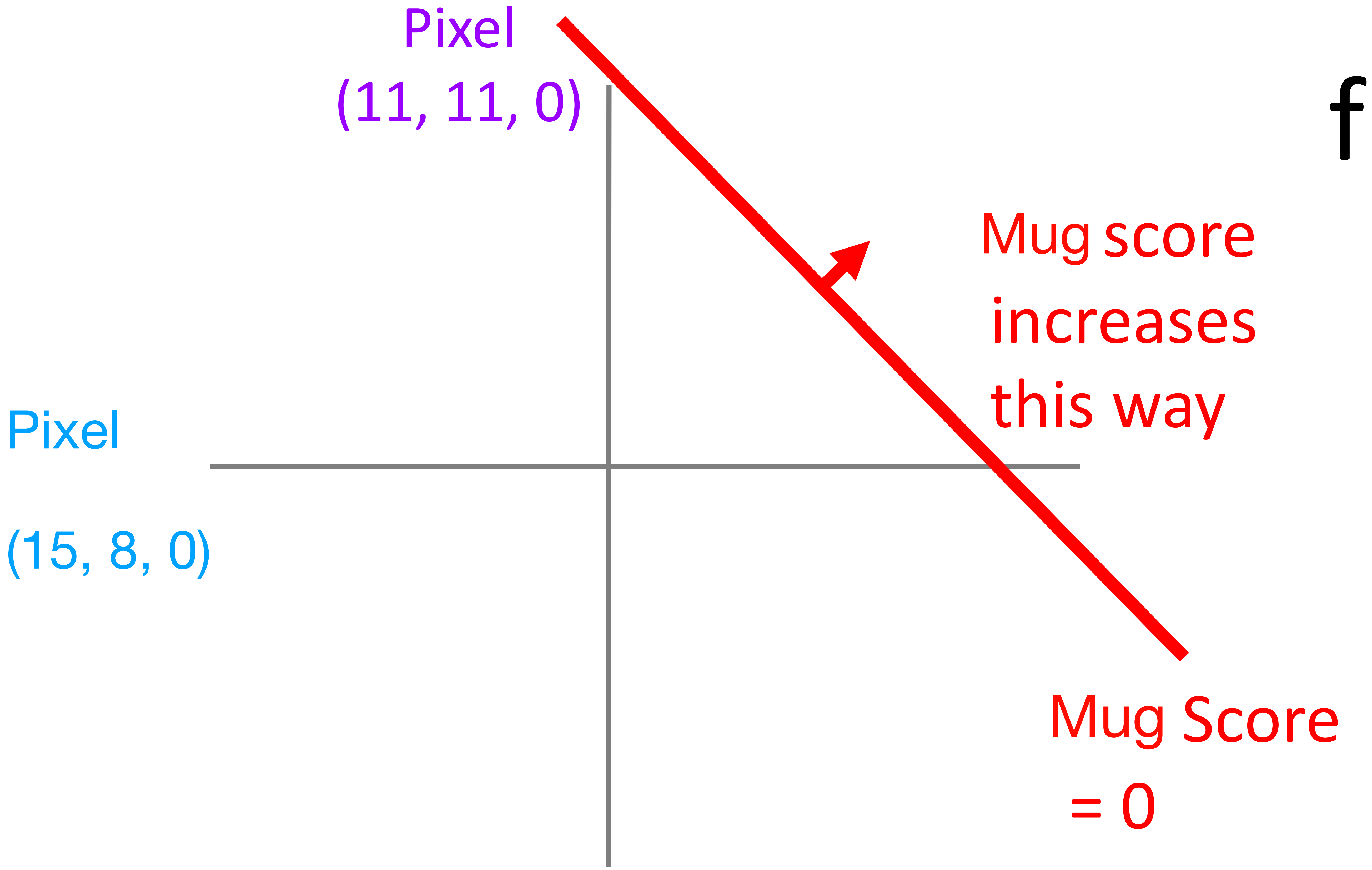
Interpreting a Linear Classifier—Geometric Viewpoint



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

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

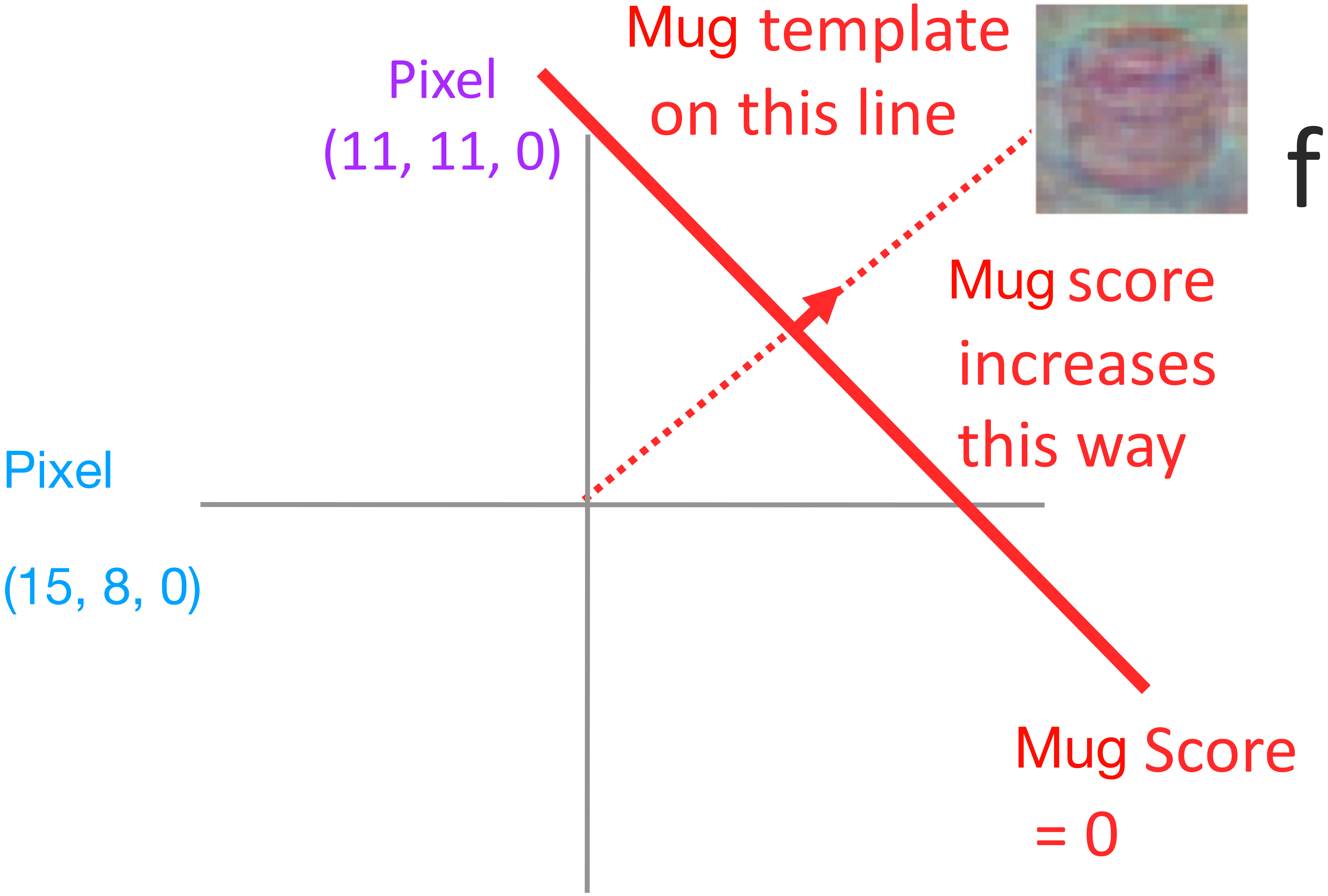
Interpreting a Linear Classifier—Geometric Viewpoint



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

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

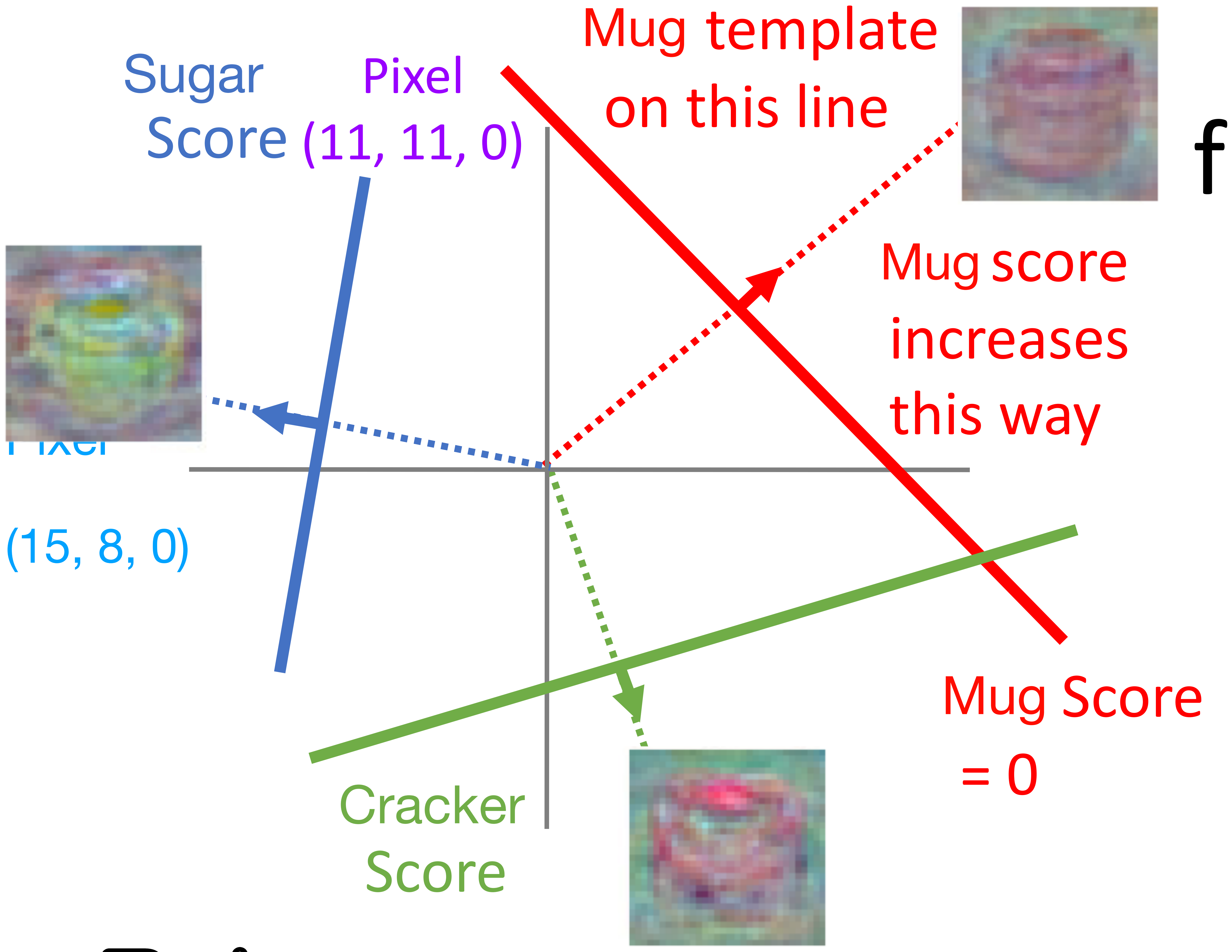
Interpreting a Linear Classifier—Geometric Viewpoint



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

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

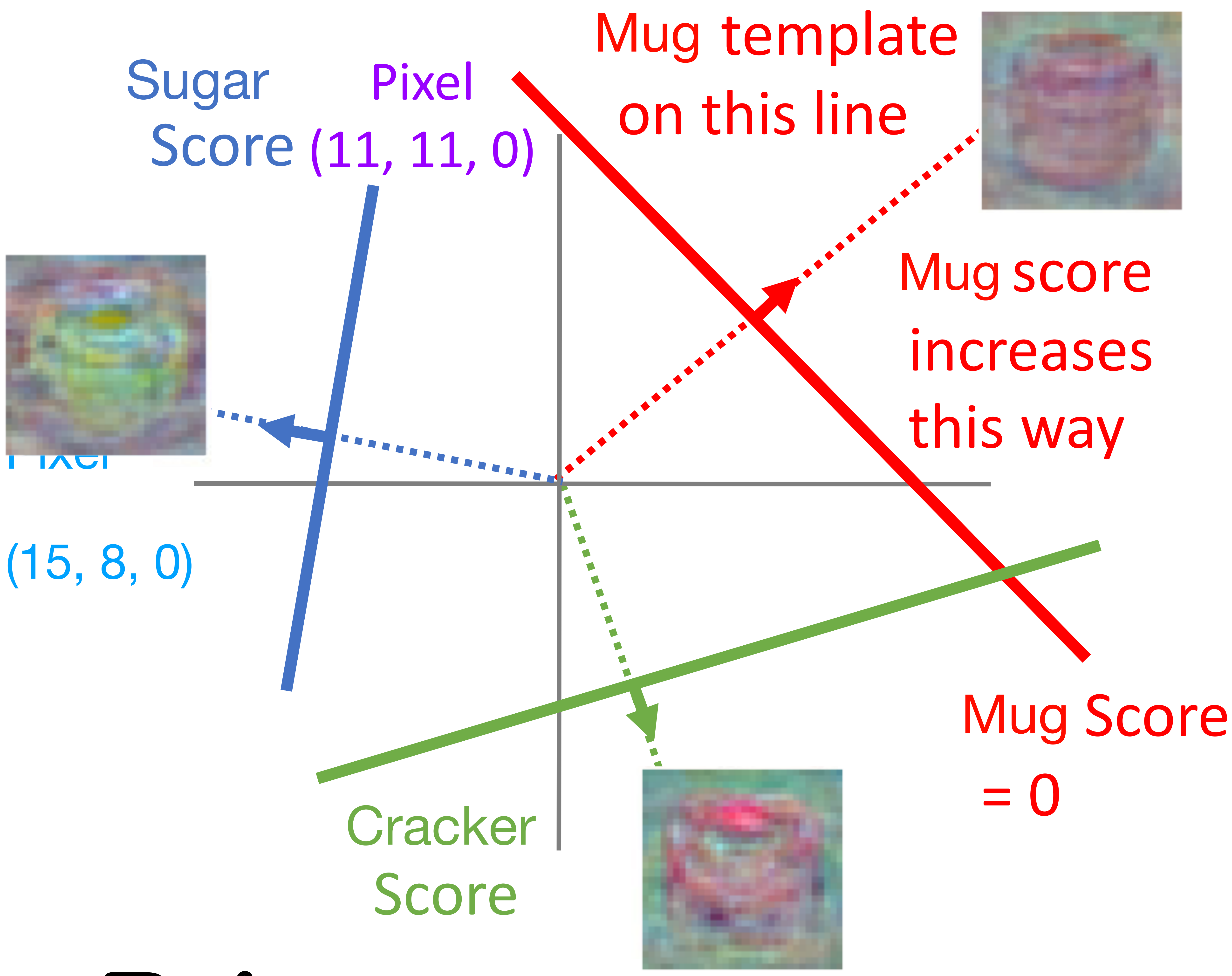
Interpreting a Linear Classifier—Geometric Viewpoint



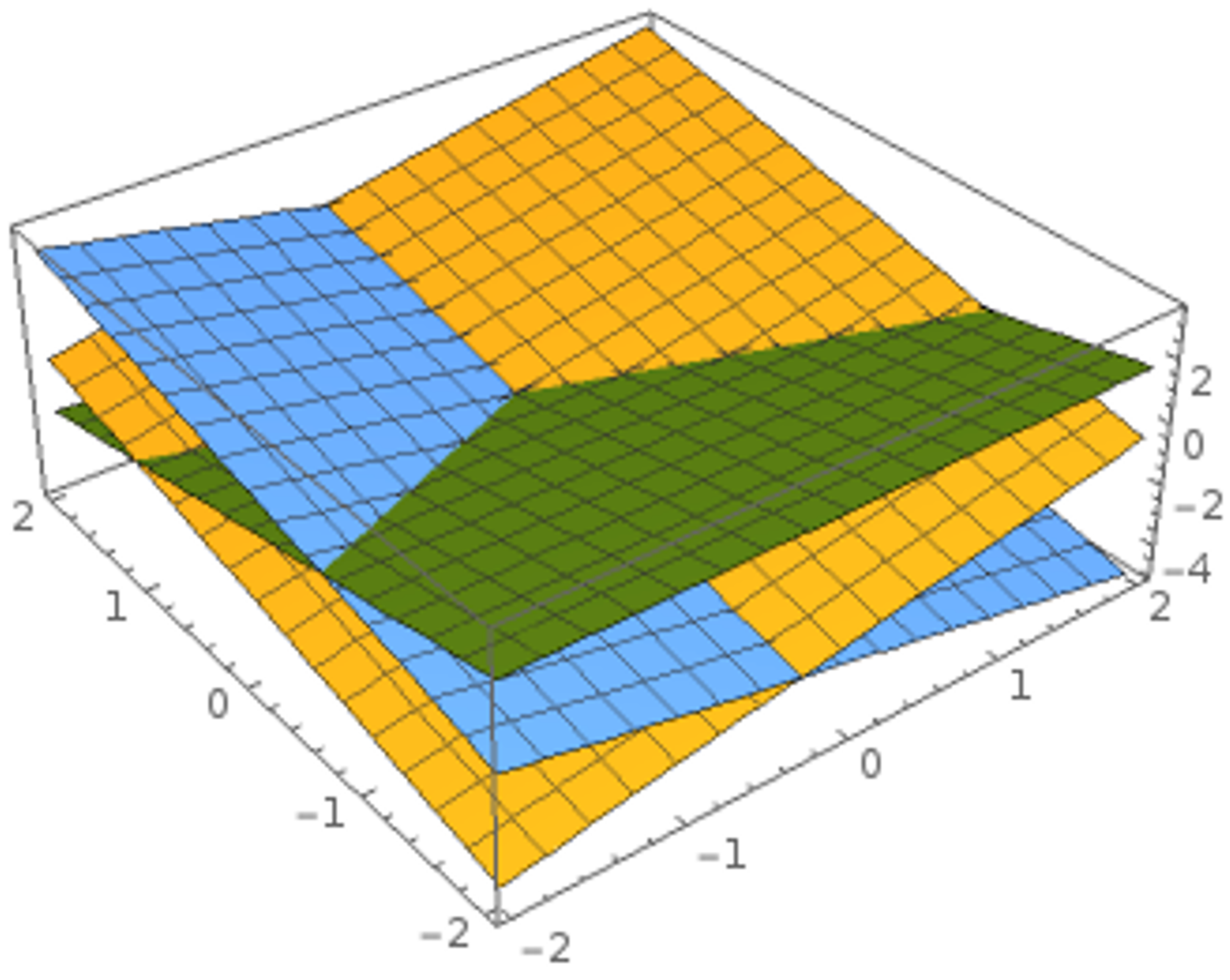
$$f(x,W) = Wx + b$$

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

Interpreting a Linear Classifier—Geometric Viewpoint



Hyperplanes carving up a high-dimensional space

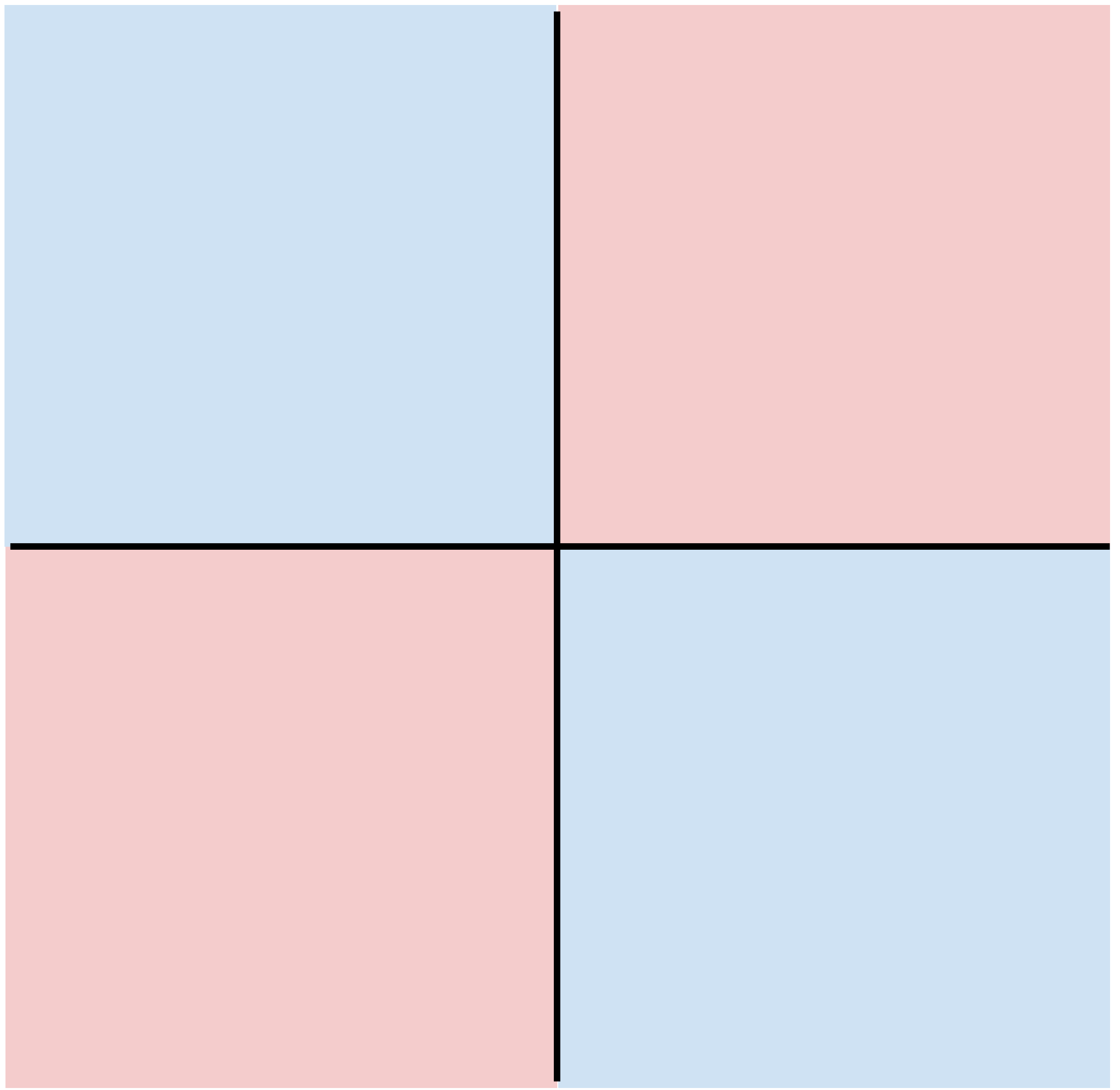


Plot created using [Wolfram Cloud]

Hard Cases for a Linear Classifier

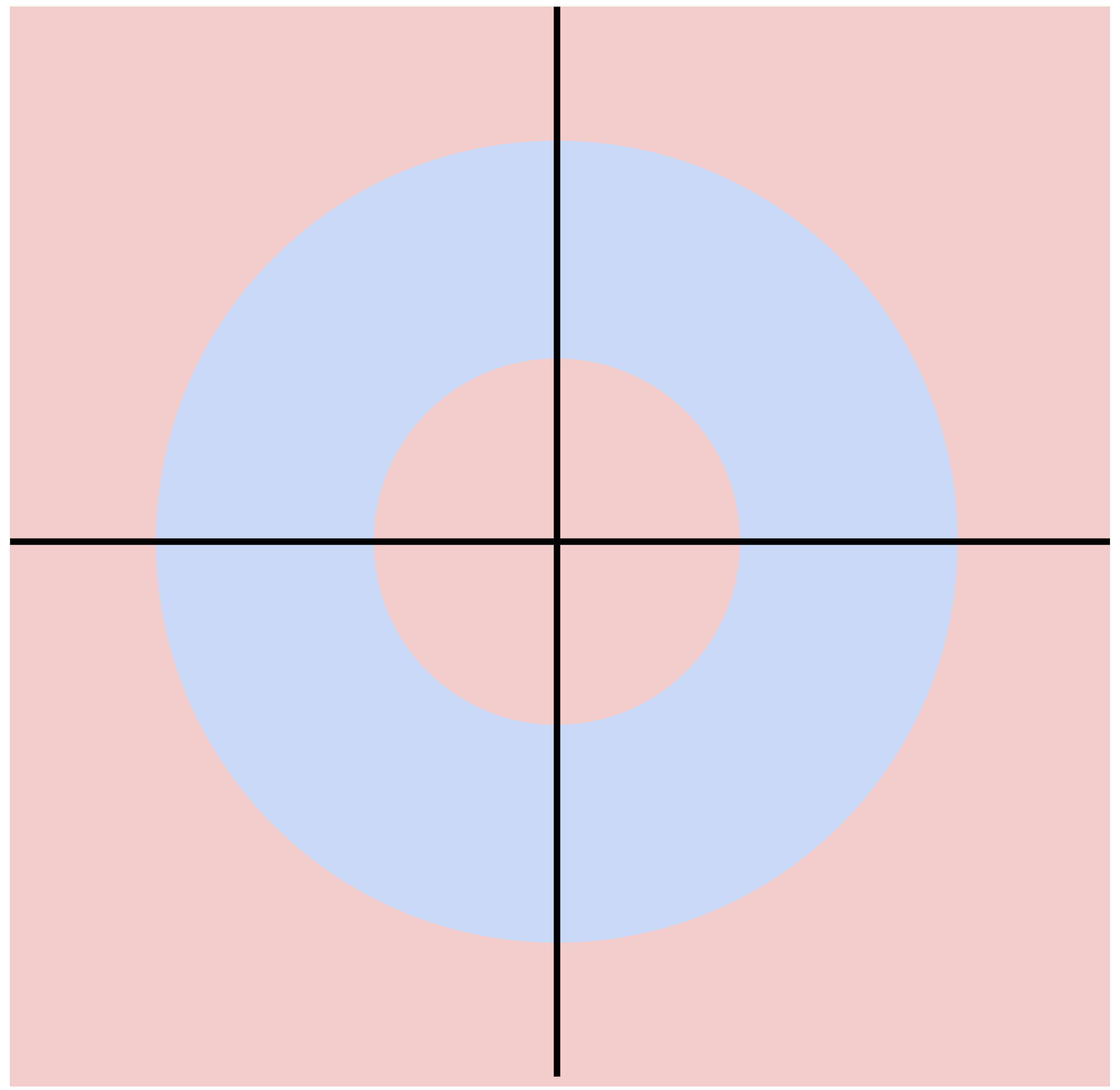
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



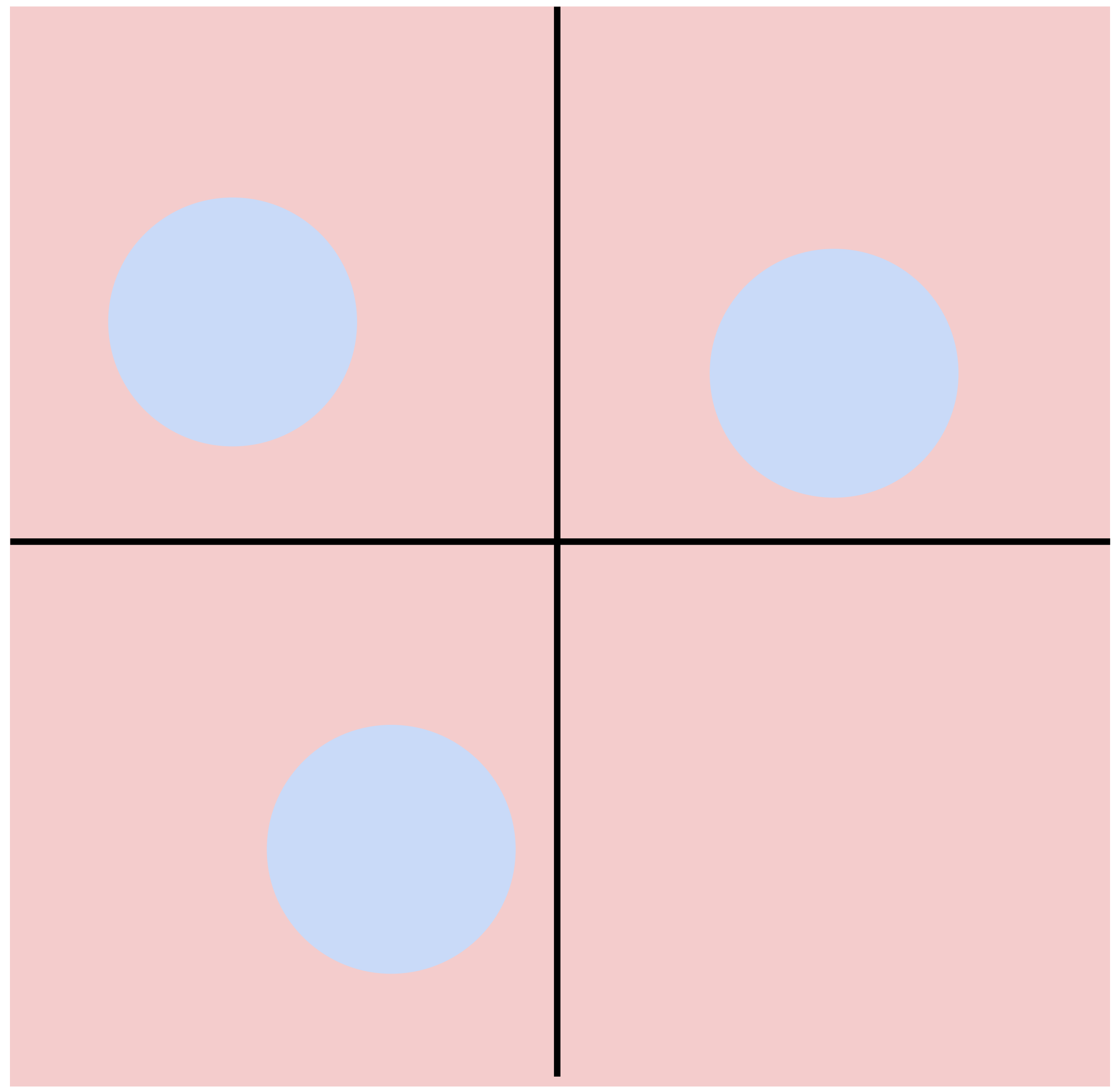
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Linear Classifier—Three Viewpoints

Algebraic Viewpoint

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

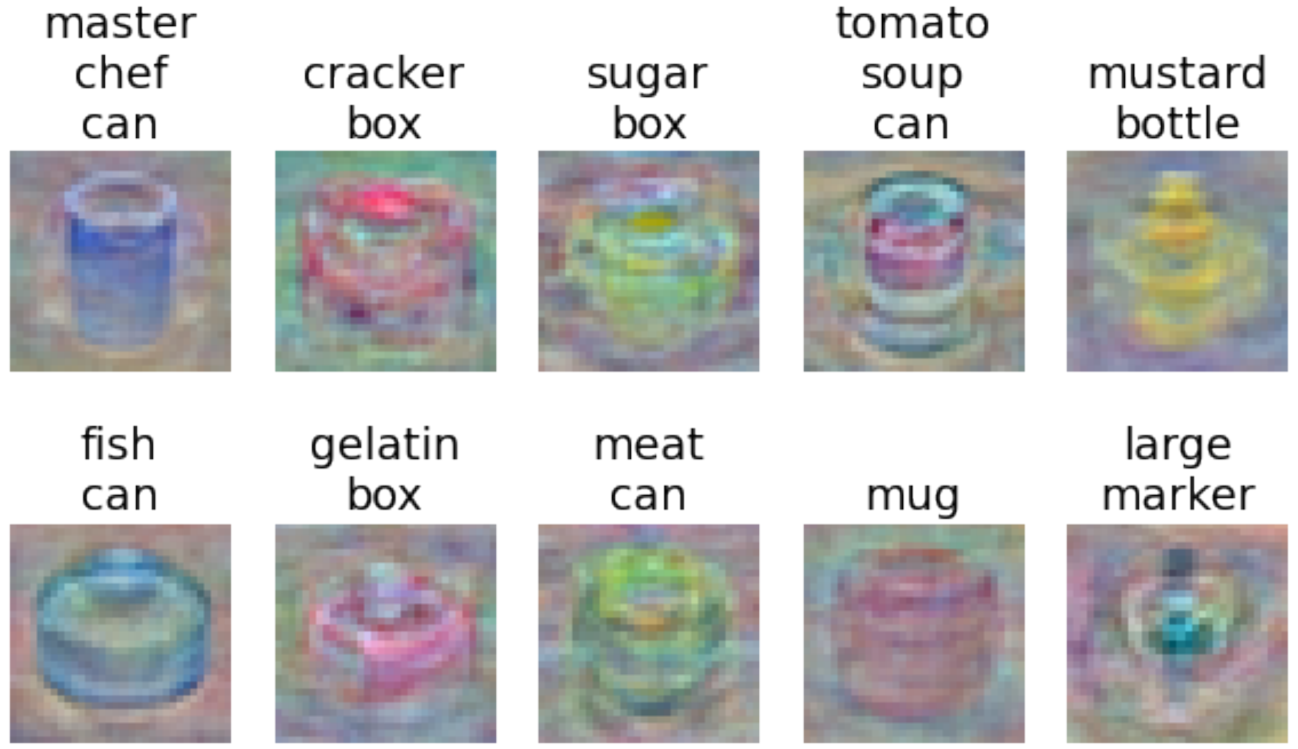
Stretch pixels into column

Input image
(2, 2)

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

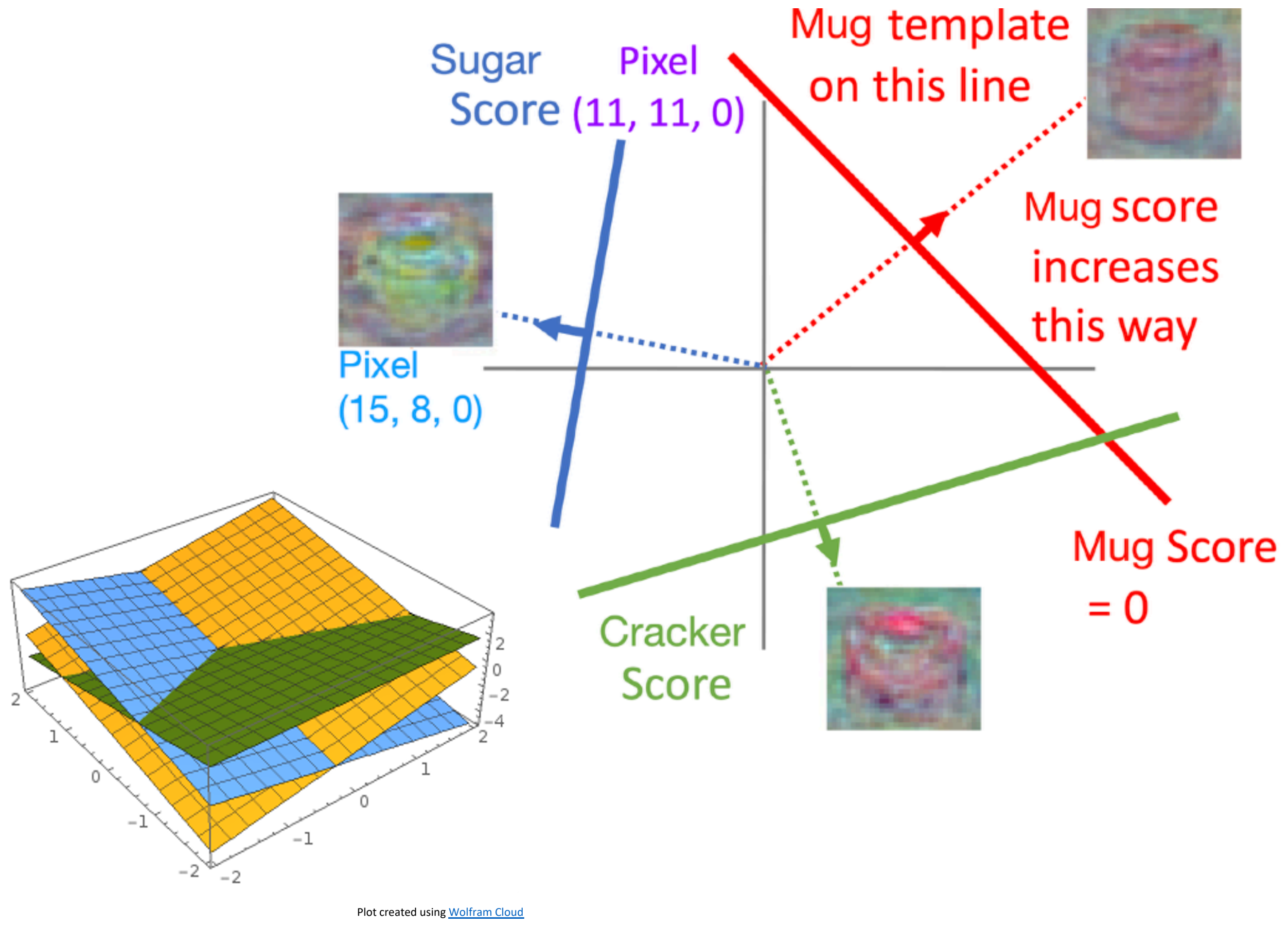
Visual Viewpoint

One template
per class



Geometric Viewpoint

Hyperplanes
cutting up space



So far—Defined a Score Function


master chef can	-3.45	-0.51	3.42
mug	-8.87	**6.04**	4.64
tomato soup can	0.09	5.31	2.65
cracker box	**2.9**	-4.22	5.1
mustard bottle	4.48	-4.19	2.64
tuna fish can	8.02	3.58	5.55
sugar box	3.78	4.49	**-4.34**
gelatin box	1.06	-4.37	-1.5
potted meat can	-0.36	-2.09	-4.79
large marker	-0.72	-2.93	6.14

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

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

But how can we actually choose a good W?

So far—Choosing a Good W


master chef can	-3.45	-0.51	3.42
mug	-8.87	**6.04**	4.64
tomato soup can	0.09	5.31	2.65
cracker box	**2.9**	-4.22	5.1
mustard bottle	4.48	-4.19	2.64
tuna fish can	8.02	3.58	5.55
sugar box	3.78	4.49	**-4.34**
gelatin box	1.06	-4.37	-1.5
potted meat can	-0.36	-2.09	-4.79
large marker	-0.72	-2.93	6.14

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

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** measures 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** measures 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** measures 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 an image and

y_i is a (discrete) label

Loss Function

A **loss function** measures 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 an image and
y_i is a (discrete) label

Loss for a single example is

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

Loss Function

A **loss function** measures 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 an image and
y_i is a (discrete) 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**

cracker	**3.2**
mug	5.1
sugar	-1.7

Cross-Entropy Loss
Multinomial Logistic Regression

cracker	**3.2**
mug	5.1
sugar	-1.7

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

Cross-Entropy Loss
Multinomial Logistic Regression

cracker	**3.2**
mug	5.1
sugar	-1.7

Unnormalized log-probabilities (logits)

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

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

Probabilities must be >=0

	Unnormalized log-probabilities (logits)	$\xrightarrow{\exp(\cdot)}$	Unnormalized probabilities
cracker	**3.2**		**24.5**
mug	5.1		164.0
sugar	-1.7		0.18

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

	Unnormalized log-probabilities (logits)		Unnormalized probabilities (Probabilities must be >=0)		Probabilities (Probabilities must sum to 1)
cracker	**3.2**	exp(·) →	**24.5**	normalize →	**0.13**
mug	5.1		164.0		0.87
sugar	-1.7		0.18		0.00

Cross-Entropy Loss
Multinomial Logistic Regression

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

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

Softmax function

Unnormalized log-probabilities (logits)	$\xrightarrow{\exp(\cdot)}$	Unnormalized probabilities	$\xrightarrow{\text{normalize}}$	Probabilities
3.2		**24.5**		**0.13**
5.1		164.0		0.87
-1.7		0.18		0.00

Probabilities must be >=0

Probabilities must sum to 1

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

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

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

	Unnormalized log-probabilities (logits)	$\xrightarrow{\exp(\cdot)}$	Unnormalized probabilities (Probabilities must be >=0)	$\xrightarrow{\text{normalize}}$	Probabilities (Probabilities must sum to 1)
cracker	**3.2**		**24.5**		**0.13**
mug	5.1		164.0		0.87
sugar	-1.7		0.18		0.00

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

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

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**

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

Softmax function

	Unnormalized log-probabilities (logits)	$\xrightarrow{\exp(\cdot)}$	Unnormalized probabilities (Probabilities must be >=0)	$\xrightarrow{\text{normalize}}$	Probabilities (Probabilities must sum to 1)	$\rightarrow$ compare $\leftarrow$	Correct probabilities
cracker	**3.2**		**24.5**		**0.13**		**1.00**
mug	5.1		164.0		0.87		0.00
sugar	-1.7		0.18		0.00		0.00

Cross-Entropy Loss
Multinomial Logistic Regression

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

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

	Unnormalized log-probabilities (logits)	exp(·) →	Unnormalized probabilities (Probabilities must be >=0)	normalize →	Probabilities (Probabilities must sum to 1)	→ compare ←	Correct probabilities
cracker	**3.2**		**24.5**		**0.13**		**1.00**
mug	5.1		164.0		0.87		0.00
sugar	-1.7		0.18		0.00		0.00

Kullback-Leibler divergence

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

Cross-Entropy Loss
Multinomial Logistic Regression

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

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

	Unnormalized log-probabilities (logits)	$\exp(\cdot)$ →	Unnormalized probabilities (Probabilities must be >=0)	normalize →	Probabilities (Probabilities must sum to 1)	compare	Correct probabilities
cracker	**3.2**		**24.5**		**0.13**	→ compare ←	**1.00**
mug	5.1		164.0		0.87		0.00
sugar	-1.7		0.18		0.00		0.00

Cross Entropy

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

Cross-Entropy Loss
Multinomial Logistic Regression

cracker	**3.2**
mug	5.1
sugar	-1.7

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

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

cracker	**3.2**
mug	5.1
sugar	-1.7

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

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: What is the min / max possible loss L_i?

Cross-Entropy Loss
Multinomial Logistic Regression

cracker	**3.2**
mug	5.1
sugar	-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: What is the min / max possible loss L_i?

A: Min: 0, Max: $+\infty$

Cross-Entropy Loss
Multinomial Logistic Regression

cracker	**3.2**
mug	5.1
sugar	-1.7

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

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

cracker	**3.2**
mug	5.1
sugar	-1.7

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

Maximize probability of correct class

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

Putting it all together

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

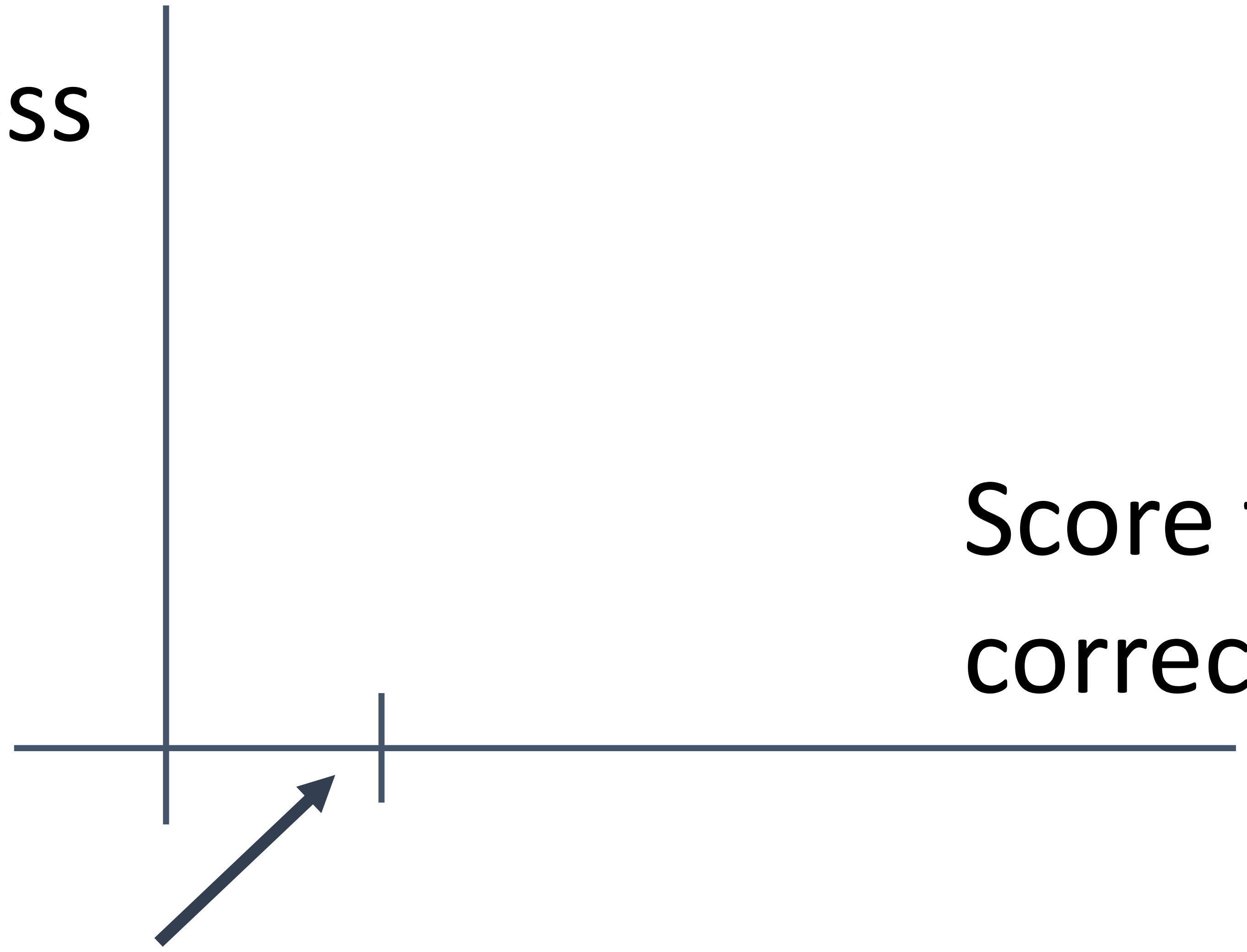
Q: If all scores are small random values, what is the loss?

A: $-\log(\frac{1}{C})$

$$\log(\frac{1}{10}) \approx 2.3$$

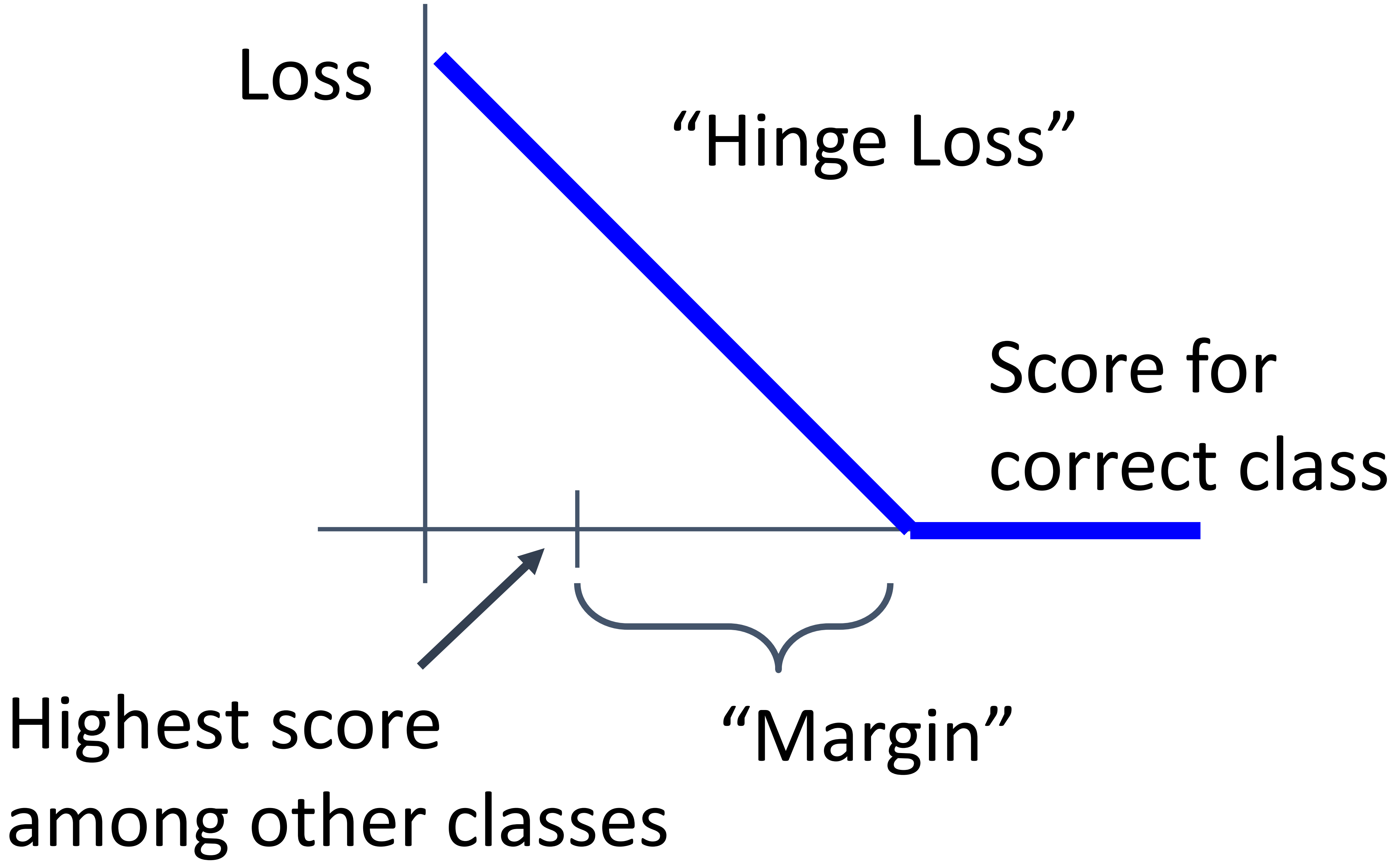
Multiclass SVM Loss

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



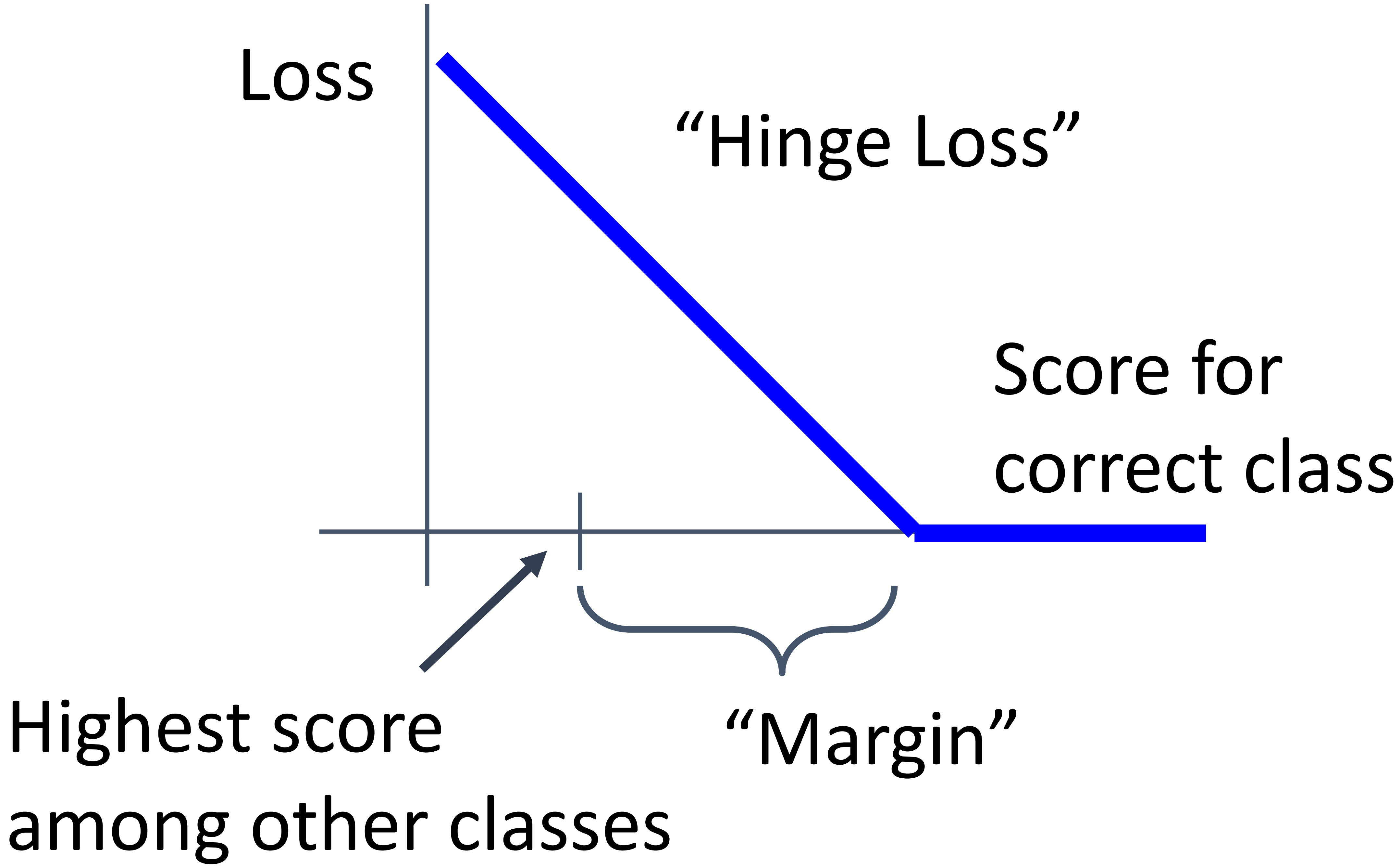
Multiclass SVM Loss

”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”



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

cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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

cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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


cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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

cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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:

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

= 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


cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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

cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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 mug image change a bit?

Multiclass SVM Loss


cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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


cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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


cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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


cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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

cracker	**3.2**	1.3	2.2
mug	5.1	**4.9**	2.5
sugar	-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 $\boxed{y_i = 0}$

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

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

Cross-Entropy vs SVM Loss

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

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

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $\boxed{y_i = 0}$

Q: What happens to each loss if I 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 $\boxed{y_i = 0}$

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

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

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

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?

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 $\boxed{y_i = 0}$

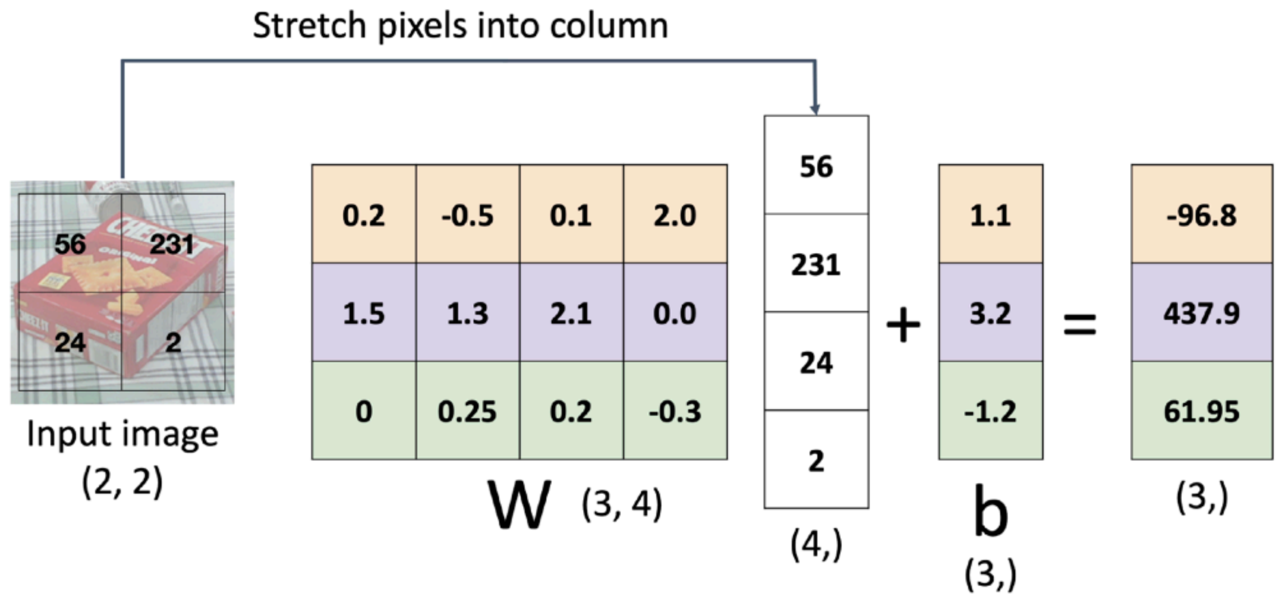
Q: What happens to each loss if I double the score of the correct class from 10 to 20?

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

Recap—Three Ways to Interpret Linear Classifiers

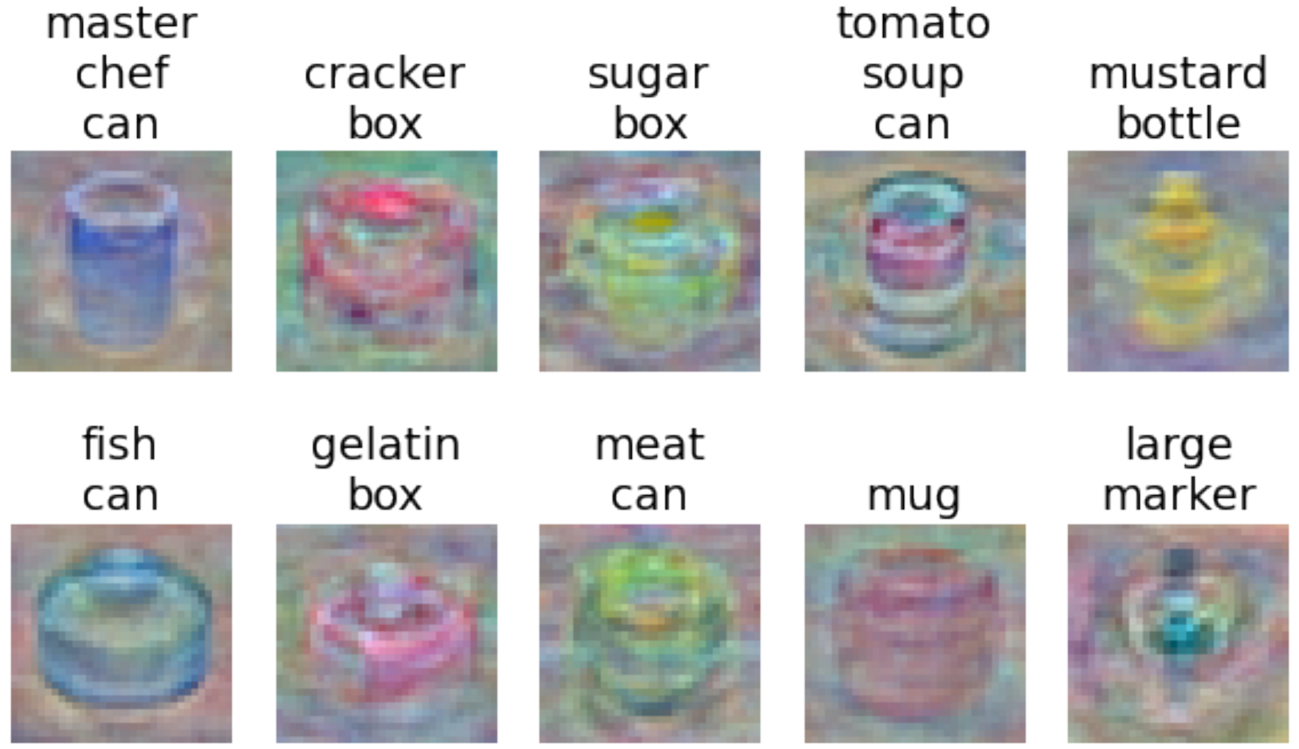
Algebraic Viewpoint

$f(x,W) = Wx$



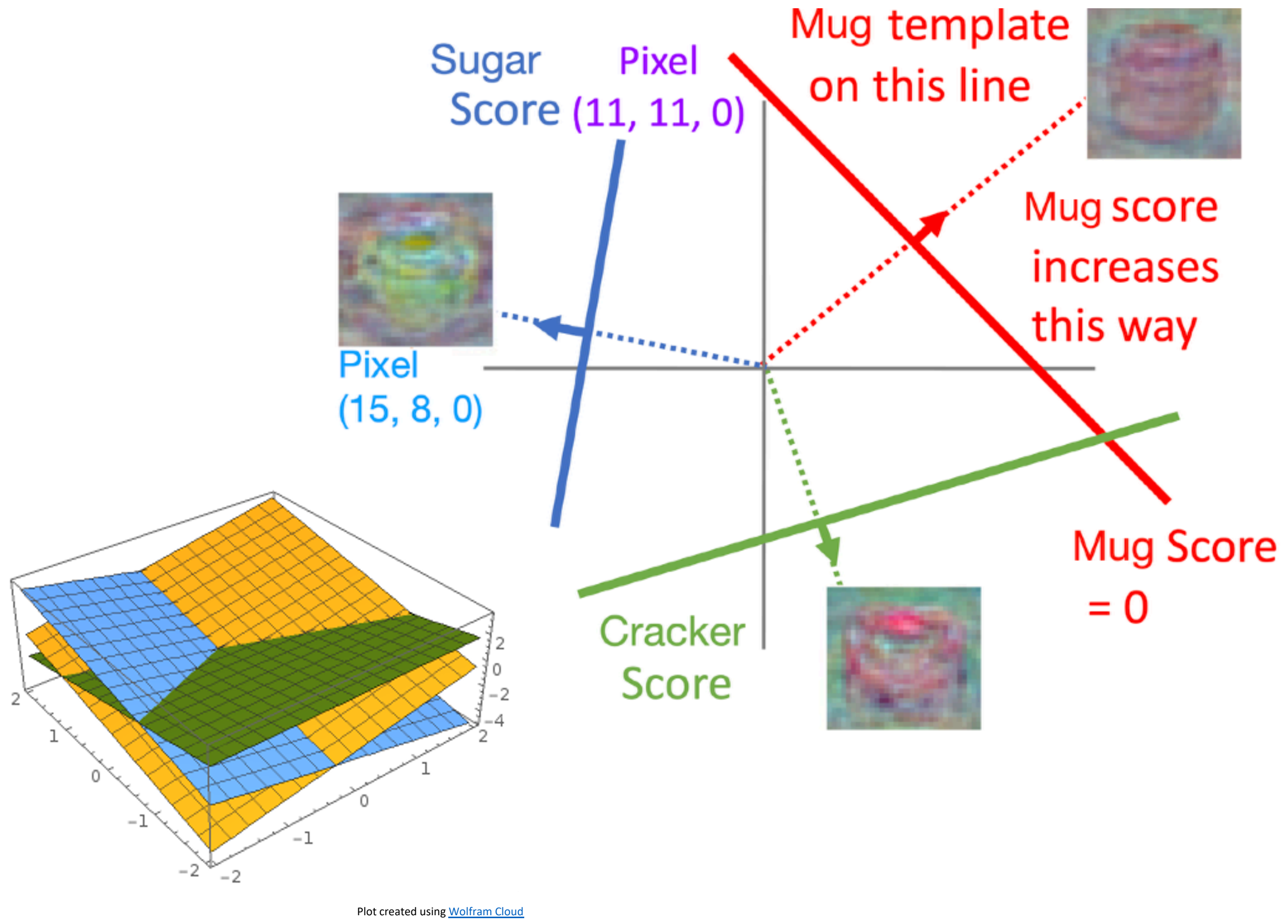
Visual Viewpoint

One template
per class



Geometric Viewpoint

Hyperplanes
cutting up space



Recap—Loss Functions Quantify Preferences

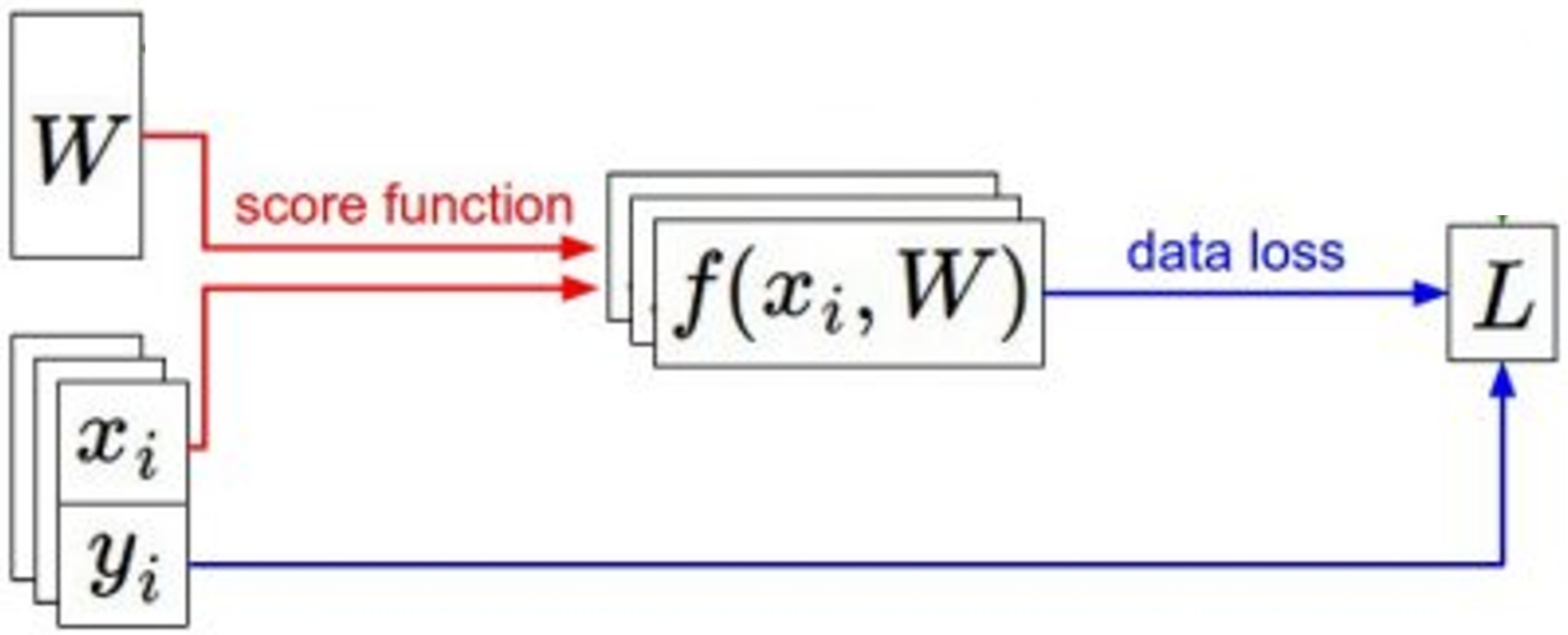
- We have some dataset of (x, y)
- We have a **score function:**
- We have a **loss function**:

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

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**:

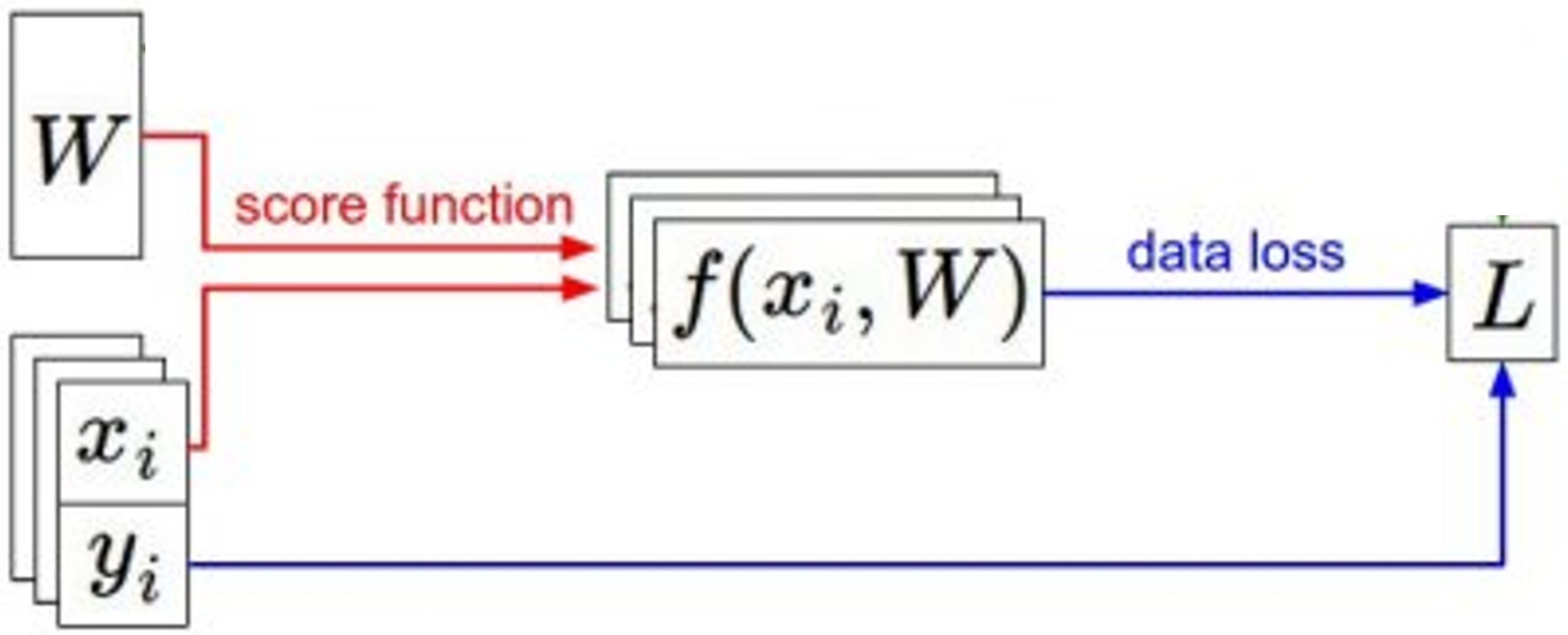
Q: How do we find the best W,b?

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

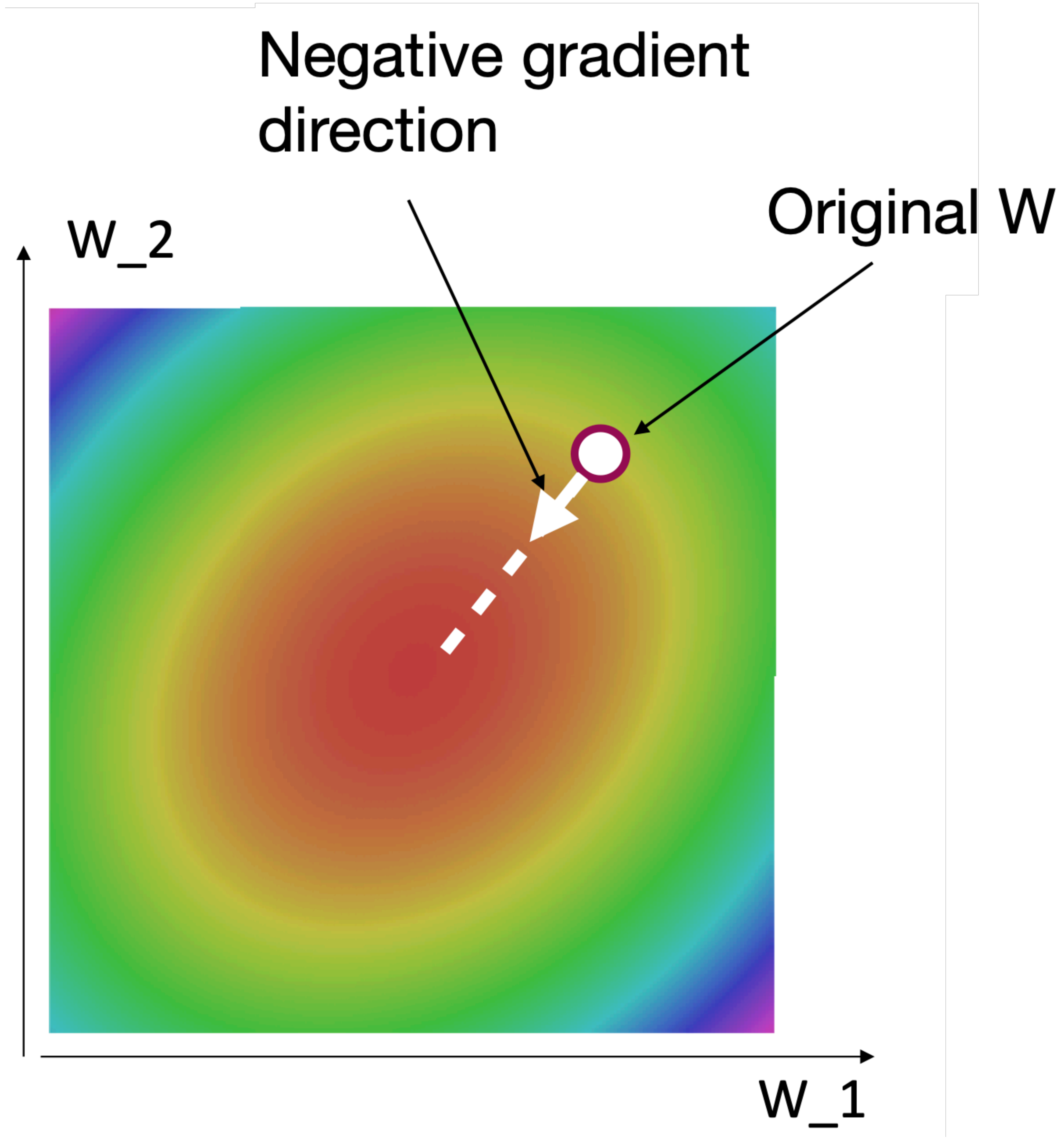
Linear classifier

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



DeepRob

Lecture 2
Linear Classifiers
University of Michigan | Department of Robotics