Machine Learning for Classification

ML Foundation

What you will learn in this course

Low-Level Vision

Image Processing:
Photometric Image formation
Point Operations
Histogram Equalization
Linear & Non-Linear filters
Convolution & Cross correlation

Feature Detection:
Edge detection
Corner detection
Line detection

Feature Description:
SIFT
Feature Matching
Image Stitching
Panorama

Geometric Vision

Transformation and Camera Mode:
Geometric Image Formation
Pinhole Camera Model

Camera Calibration:
Estimating intrinsics
Estimating extrinsics

Visual Understanding

Object Detection & Tracking:
Single-stage detectors
Two-stage detectors
YOLO
Object tracking

Image Segmentation:
Traditional Image Segmentation
Learning-based Image Segmentation

Machine Learning for CV

Machine Learning for Classification:
k-NN
Linear Classifier

Deep Learning:
Optimization
Neural Network
CNN
CNN architectures:

Generative & Geometric Learning

Generative Models:
AE, VAE, GAN

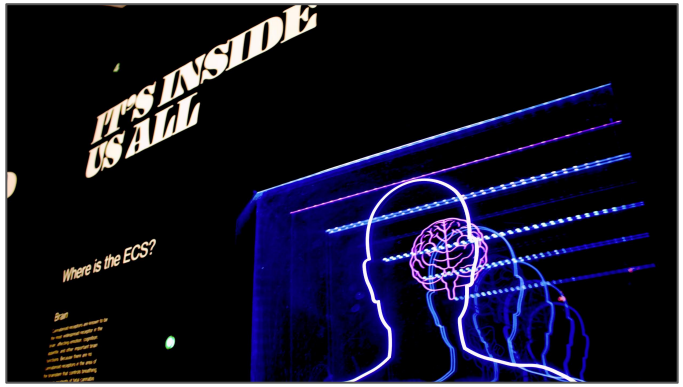
3D Vision and 3D Deep Learning:
3D shape representations
Geometric deep learning overview
3D Deep Learning models : pointnet

What we will learn today

- Classical ML Classifiers
  - k-NN
  - Linear Classifier

Intelligence and Learning Definitions

Intelligence



A mental capabilities, that involve the abilities:

- To reason
- Solve problems
- Learn **from experience**

Learning

Is the process of

- Acquiring new, or
- modifying/improving existing

Skills, abilities, knowledge **from experience**

Classical ML

Nearest Neighbor

Image Classification

Assign an image to a known class based on the objects present in the image

Input: image

Output: Assign image to one of fixed set of categories

Cat

Dog

Bird

Deer

Car

Cow

Pig

Image Classification

Problem: Semantic Gap


```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 1280 x 720 x 3(3 channels RGB)

Challenges

Illumination Variation

Viewpoint Variation

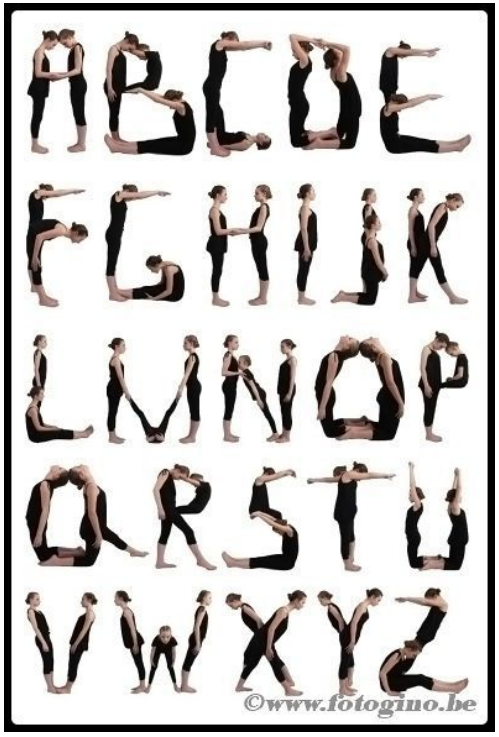
Intra-class Variation

Occlusion

Challenges

Scale Variation

Deformation



Background Clutter

Image Classification

Classifier

```
def classify_image(image):
  #some magic here
  return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Classifier

We could try

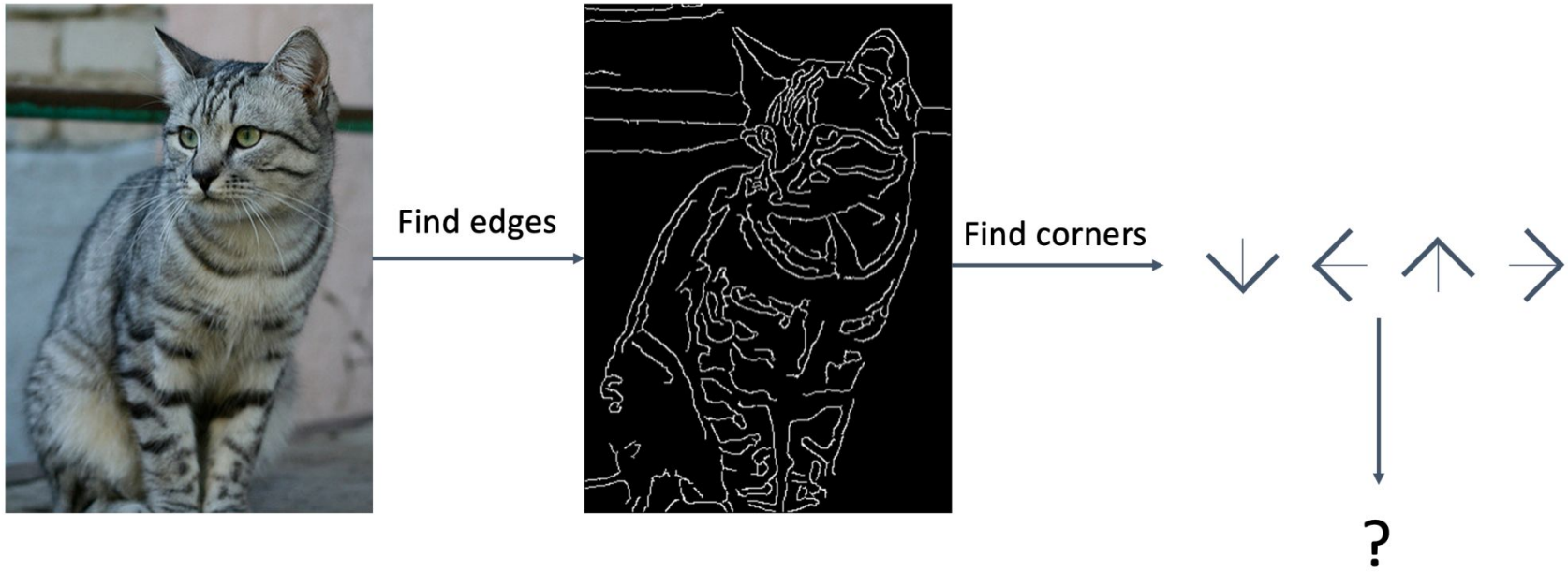


Image Classification

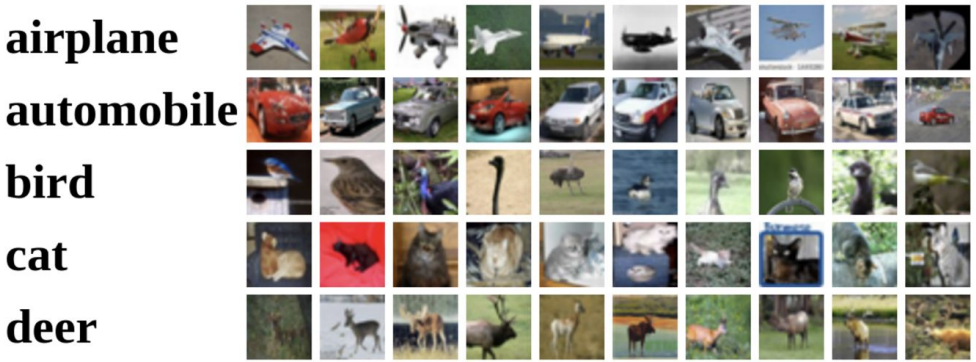
Classifier: Data-Driven Approach

1. Collect a dataset of **images and labels**
2. Use Machine Learning to **train** a classifier
3. **Evaluate** the classifier on new images

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

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

Example training set



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Datasets: MNIST

10 classes: Digits 0 to 9

28x28 **grayscale** images

50k training images

10k test images

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Datasets: CIFAR10

10 classes

50k training images (5k per class)

10k testing images (1k per class)

32x32 **RGB** images

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Datasets: CIFAR10

100 classes

50k training images (500 per class)

10k testing images (100 per class)

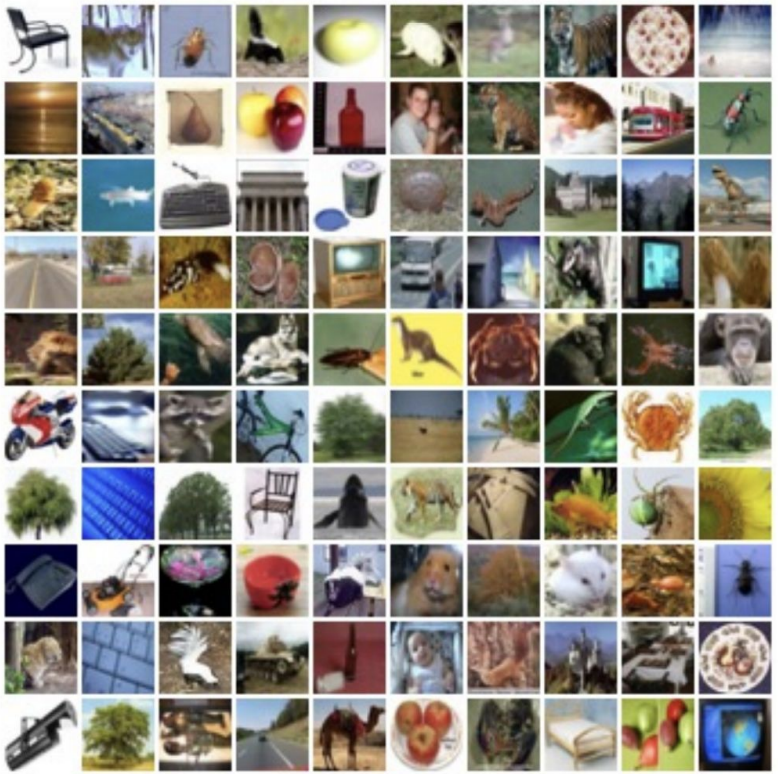
32x32 **RGB** images

20 superclasses with 5 classes each:

Aquatic mammals: beaver, dolphin,

otter, seal, whale

Trees: Maple, oak, palm, pine, willow



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Cherd

Image Classification

Datasets: ImageNet

1000 classes

~1.3M training images (~1.3K per class)

50K validation images (50 per class)

100K test images (100 per class)

Performance metric: Top 5 accuracy

Algorithm predicts 5 labels for each image; one of them needs to be right

test labels are secret! Images have variable size, but often resized to 256x256 for training There is also a 22k category version of ImageNet, but less commonly used

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

First classifier: Nearest Neighbor

Memorize all data and labels

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

Predict the label of the most similar training image

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor: Distance Metric to compare images

L1 (Manhattan) distance : $d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$

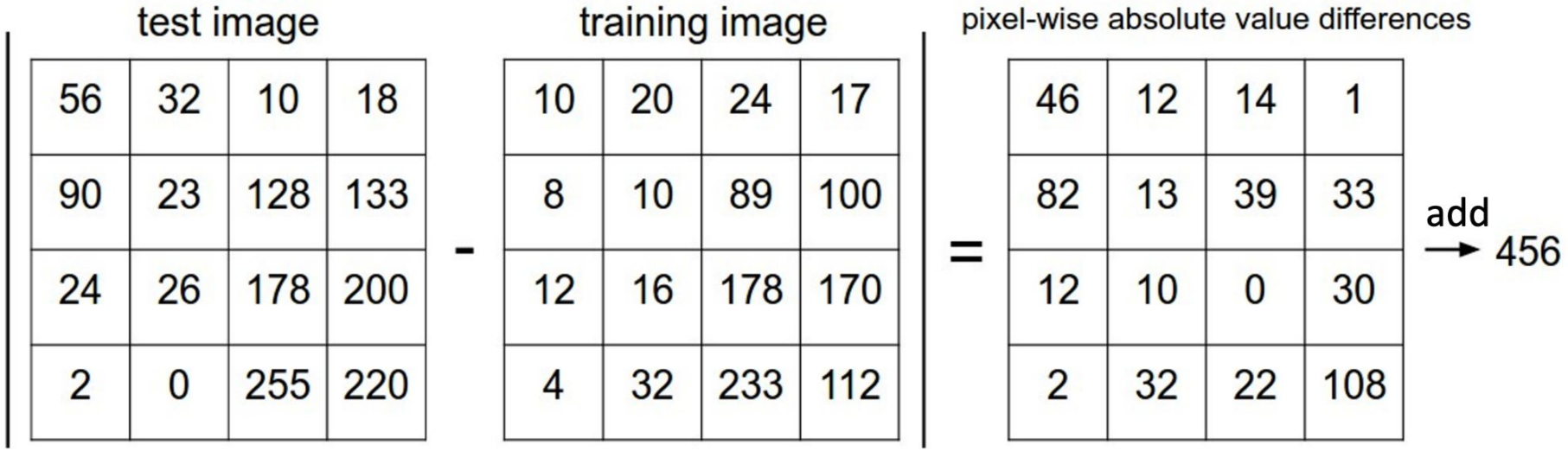


Image Classification

Nearest Neighbor Classifier

Memorize training data

For each test image:

- Find nearest training image
- Return label of nearest image

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor Classifier

Q: With N examples, how fast is training?

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor Classifier

Q: With N examples, how fast is training?

A: O(1) -> Constant time

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor Classifier

Q: With N examples, how fast is training?
A: O(1) -> Constant time

Q: With N examples, how fast is testing?

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor Classifier

Q: With N examples, how fast is training?
A: O(1)

Q: With N examples, how fast is testing?
A: O(N) - Linear Time

This is bad: We can afford slow training, but we need fast testing!

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor Classifier

There are many methods for fast / approximate nearest neighbors; e.g. see

https://github.com/facebookresearch/faiss

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

First classifier: Nearest Neighbor

What does this look like?

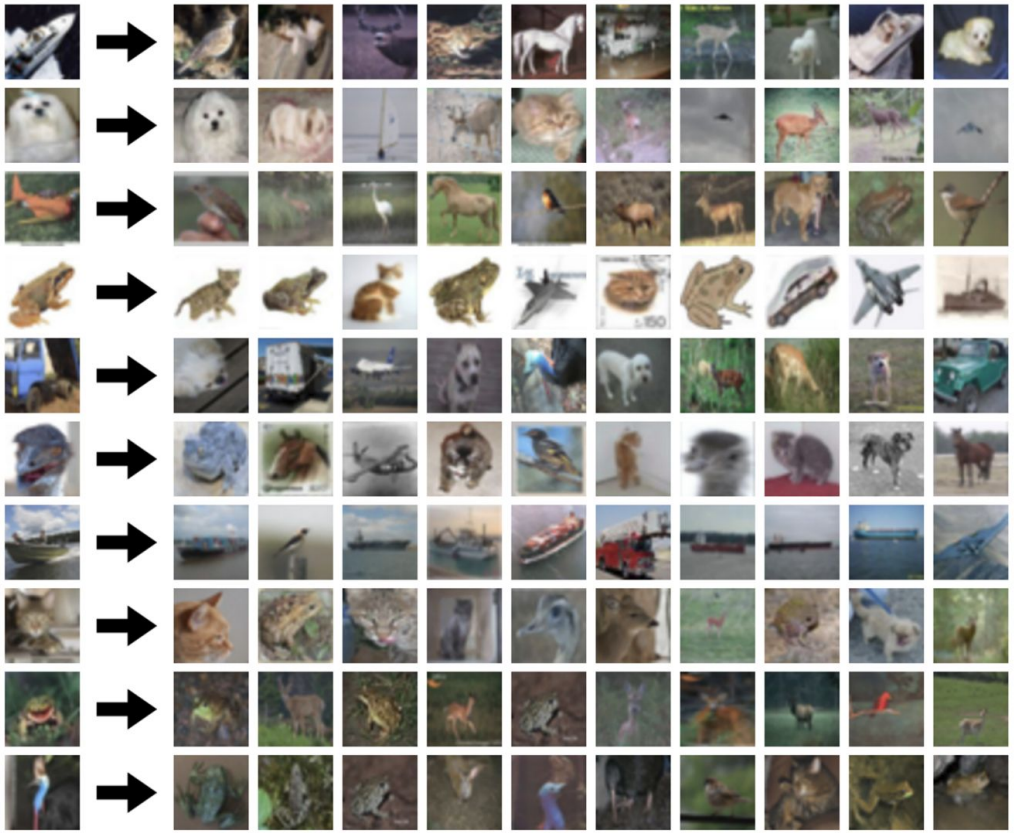


Image Classification

First classifier: Nearest Neighbor

What does this look like?

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

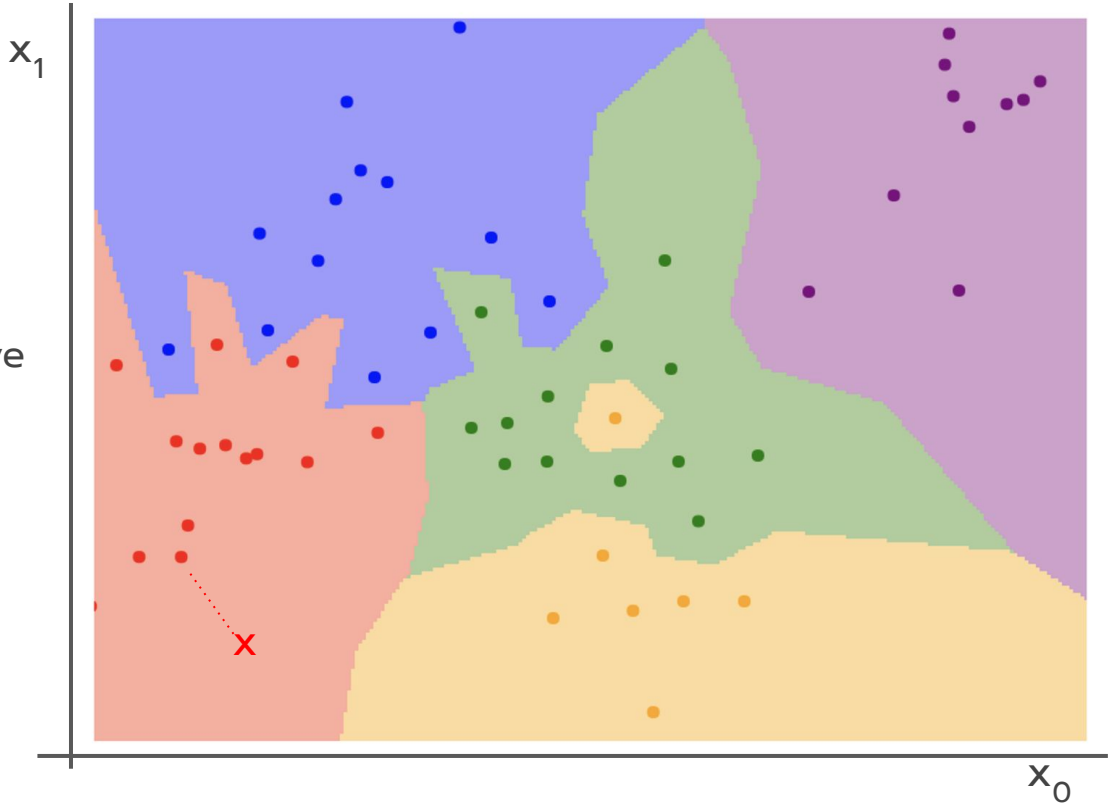
Image Classification

Nearest Neighbor Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels.

Background colors give the category a test point would be assigned.



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Cherdsak Kingkan

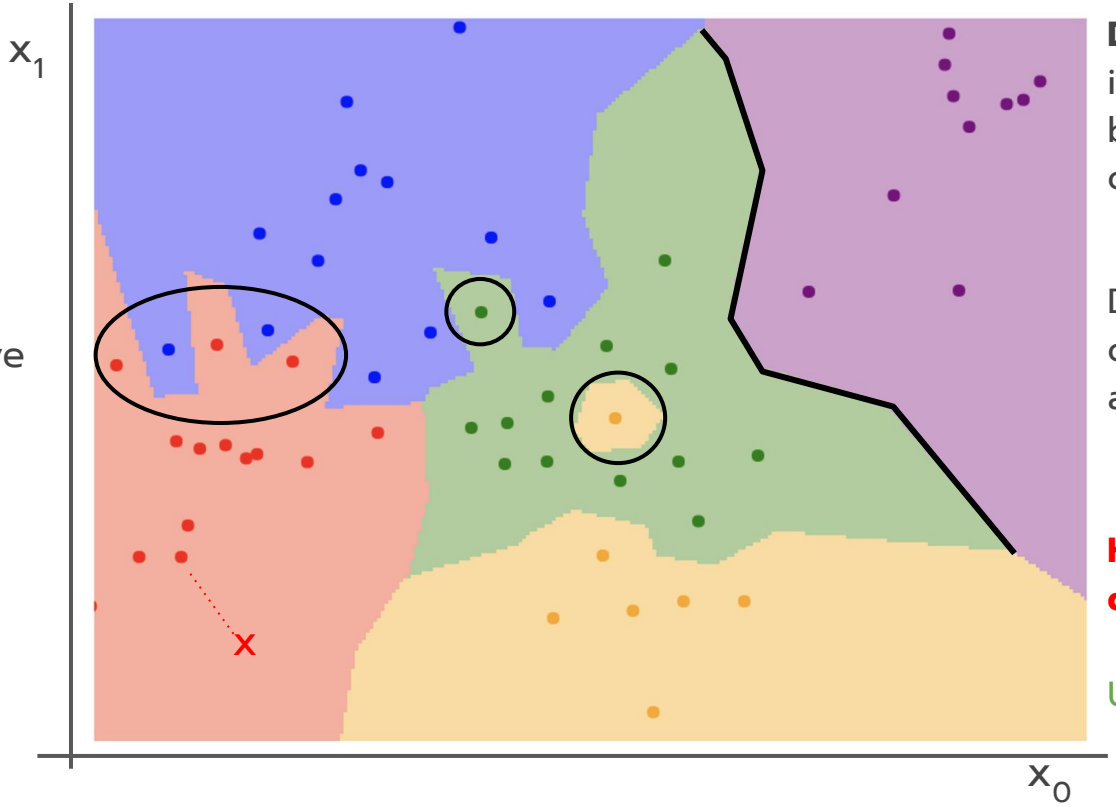
Image Classification

Nearest Neighbor Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels.

Background colors give the category a test point would be assigned.



Decision boundary is the boundary between two classification regions

Decision boundaries can be noisy; affected by outliers

How to smooth out decision boundaries?

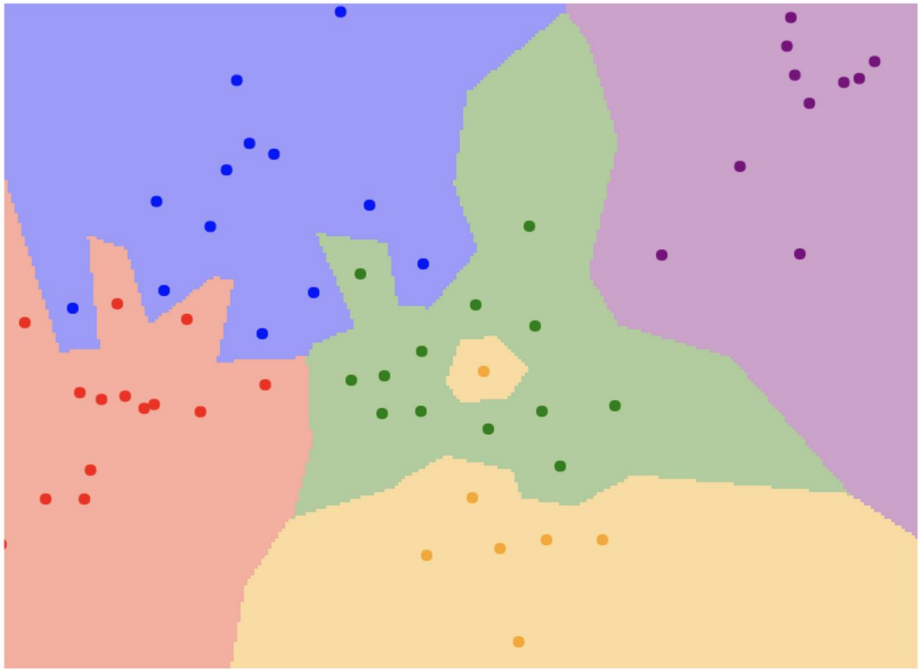
Use more neighbors!

Image Classification

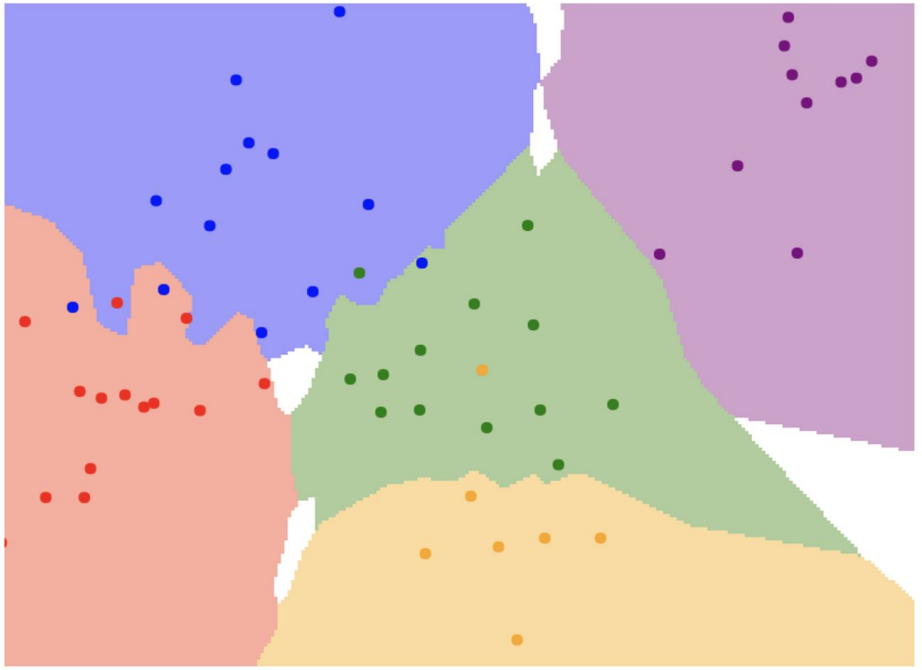
K-Nearest Neighbors

Instead of copying label from nearest neighbor, take **majority vote** from *K* closest points

K = 1



K = 3



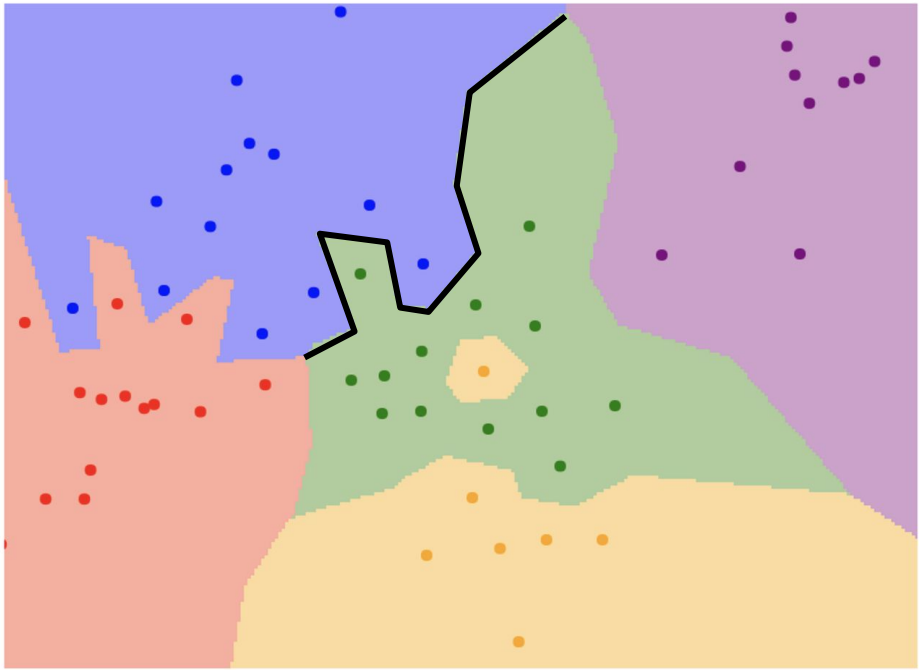
Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

K-Nearest Neighbors

Using more neighbors helps smooth out rough decision boundaries

K = 1



K = 3

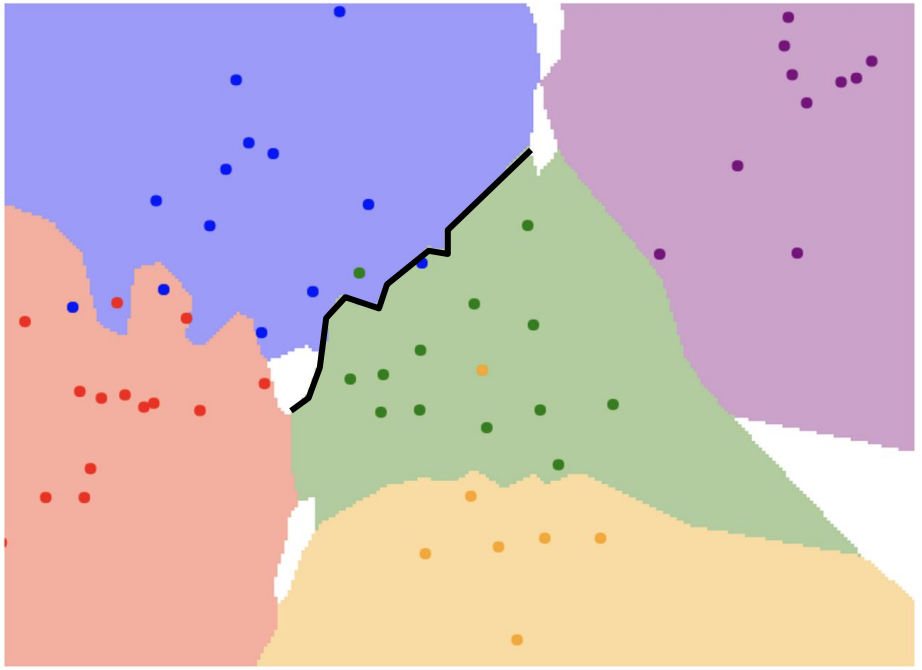
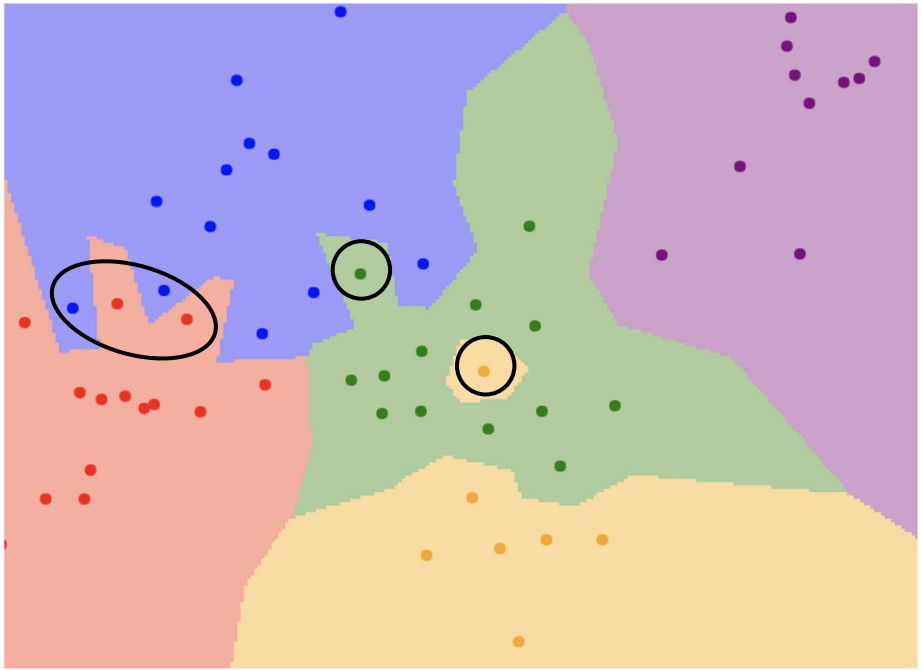


Image Classification

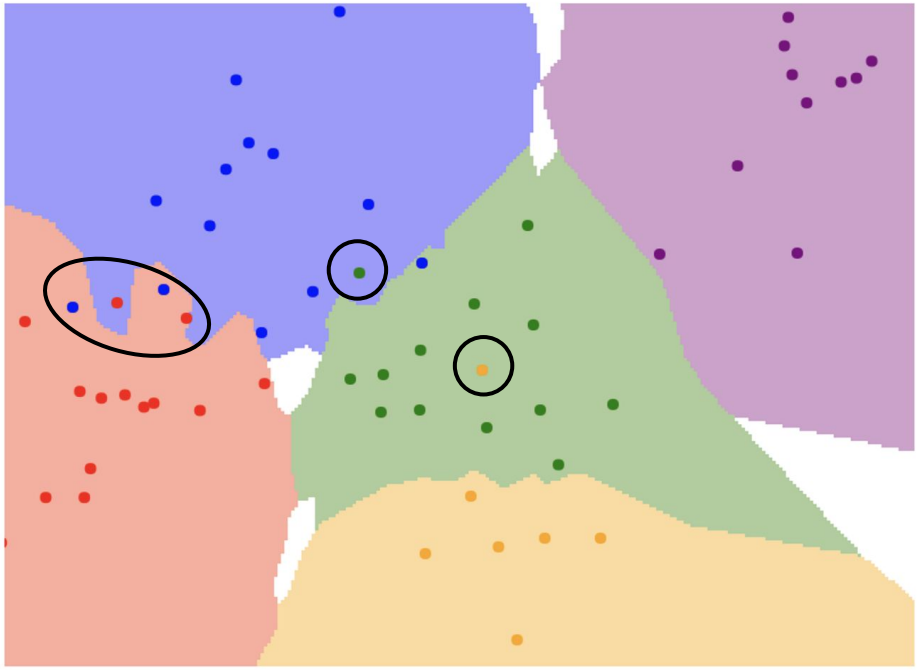
K-Nearest Neighbors

Using more neighbors helps reduce the effect of outliers

K = 1



K = 3



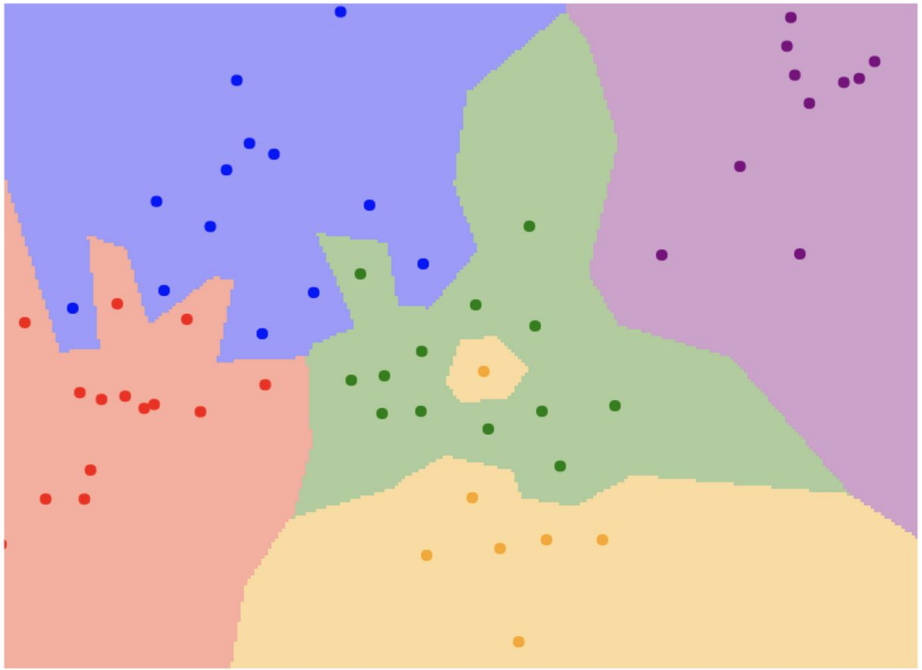
Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

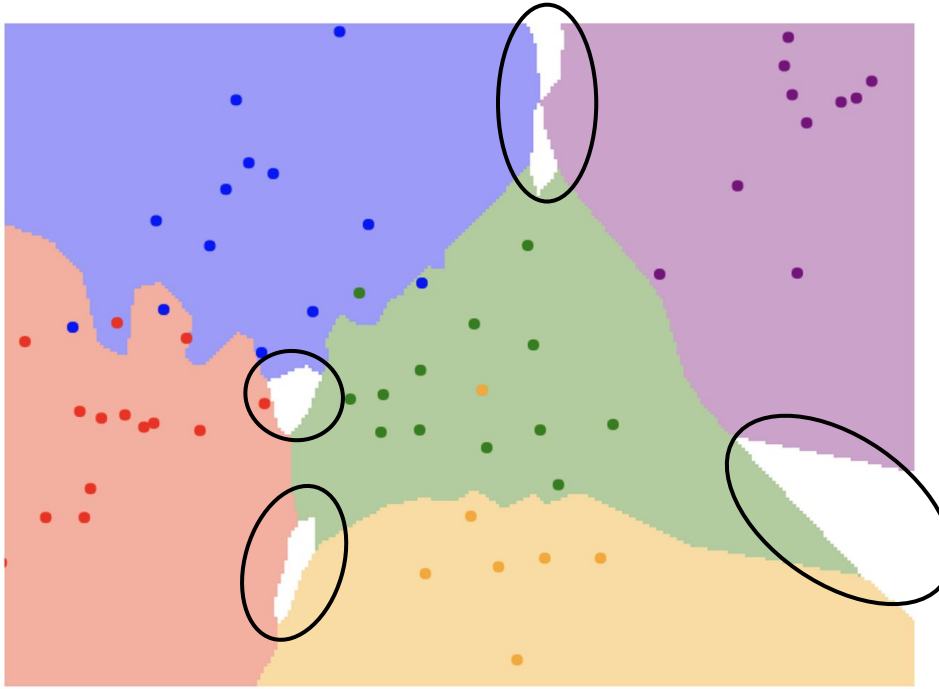
K-Nearest Neighbors

When K > 1 there can be ties between classes. Need to break somehow!

K = 1



K = 3



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance:

$$d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$$

L2 (Euclidean) distance:

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

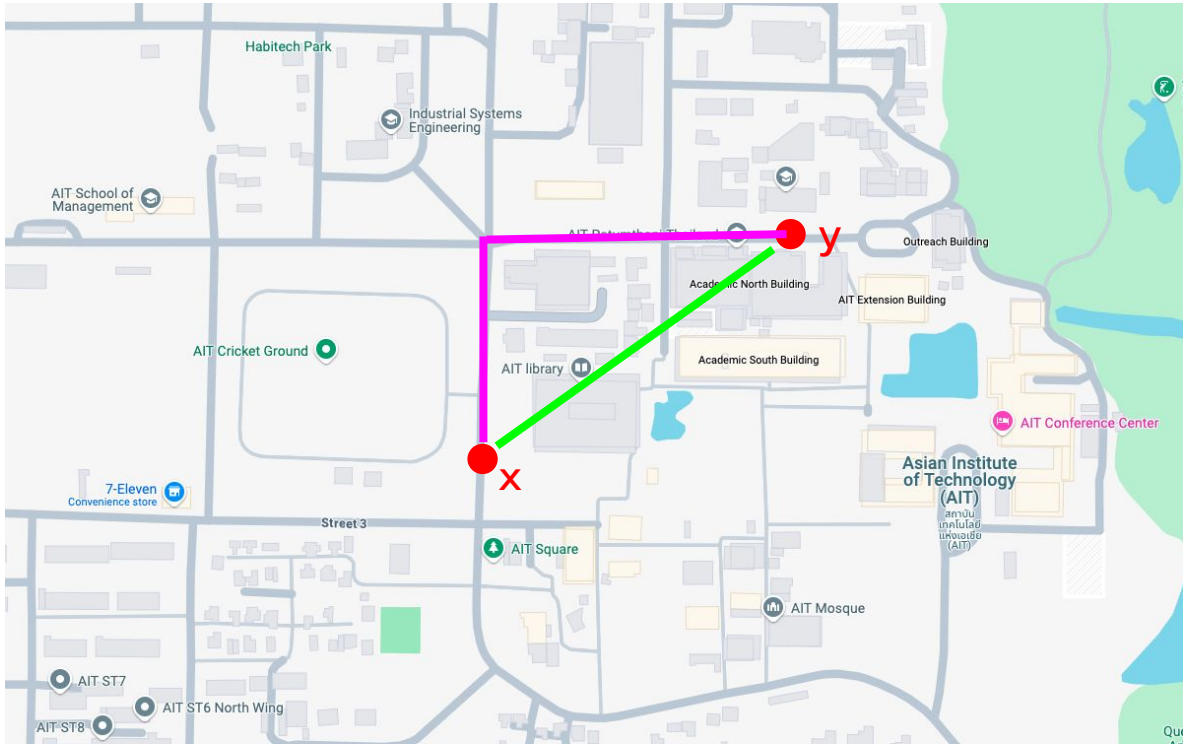


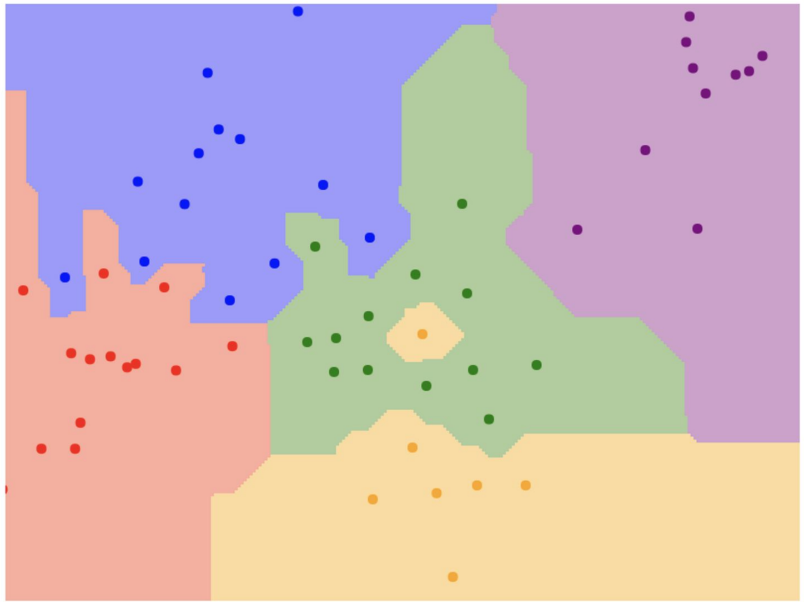
Image Classification

K-Nearest Neighbors: Distance Metric

With the right choice of distance metric, we can apply K-Nearest Neighbor to any type of data!

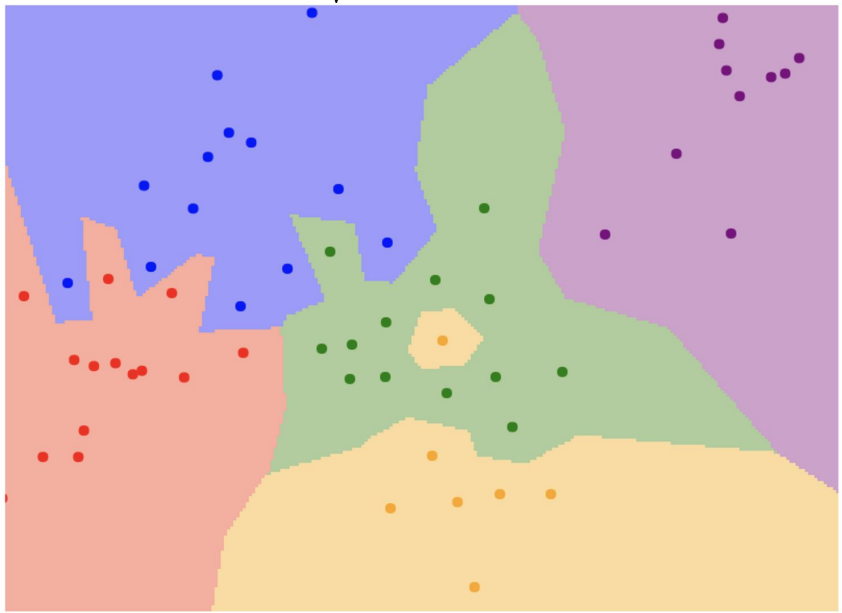
L1 (Manhattan) distance:

$$d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$$



L2 (Euclidean) distance:

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

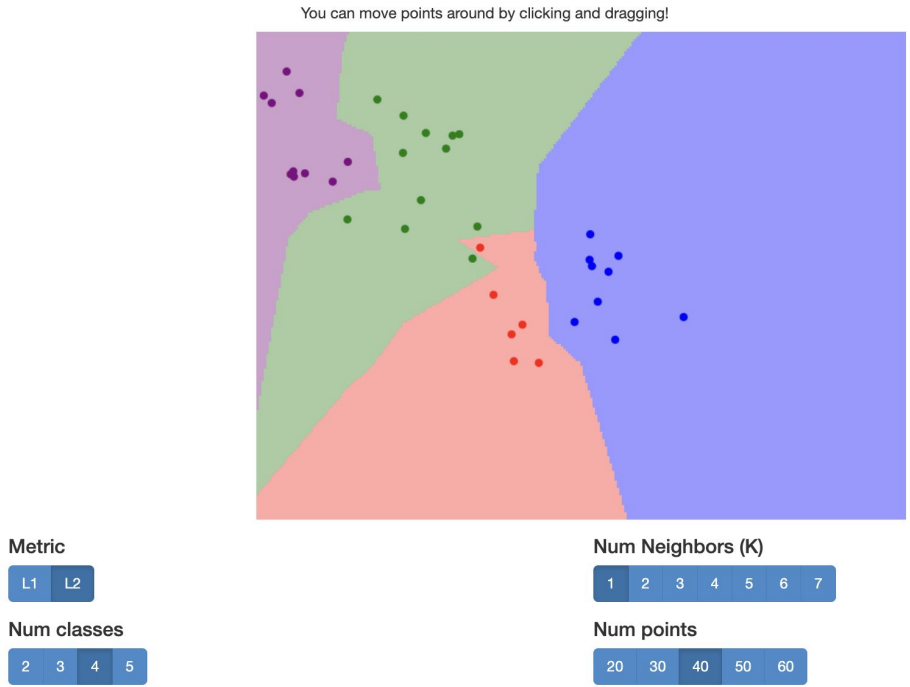


$K = 1$

Image Classification

K-Nearest Neighbors:

Demo:
http://vision.stanford.edu/teaching/cs231n-demos/knn/



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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 see what works best for our data / task.

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Hyperparameters

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

Your dataset

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Hyperparameters

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

BAD: $K = 1$ always works perfectly on training data

Your dataset

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Hyperparameters

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

BAD: $K = 1$ always works perfectly on training data

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Hyperparameters

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

BAD: $K = 1$ always works perfectly on training data

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

Because once you use the test set to set the values of your hyperparameters, the test set is no longer unseen data. And you no longer have any estimate how your model is going to perform when you deploy it in the wild.

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Hyperparameters

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

BAD: $K = 1$ always works perfectly on training data

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

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

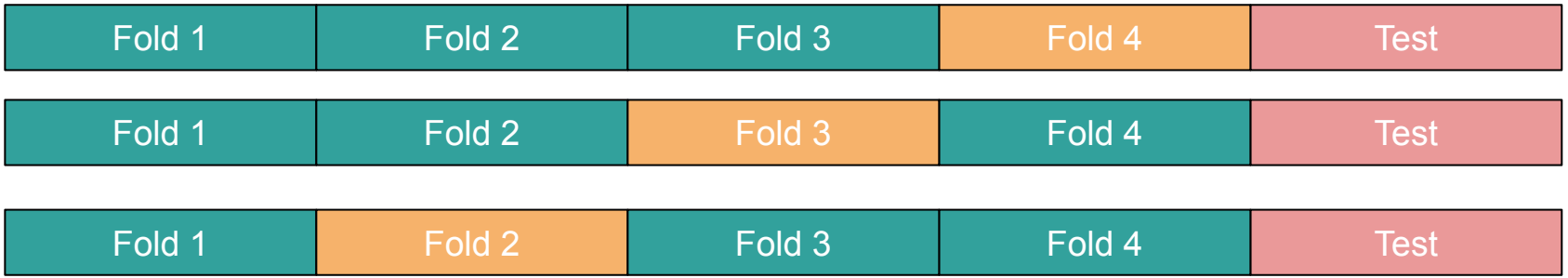
Better!

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Hyperparameters

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



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

Image Classification

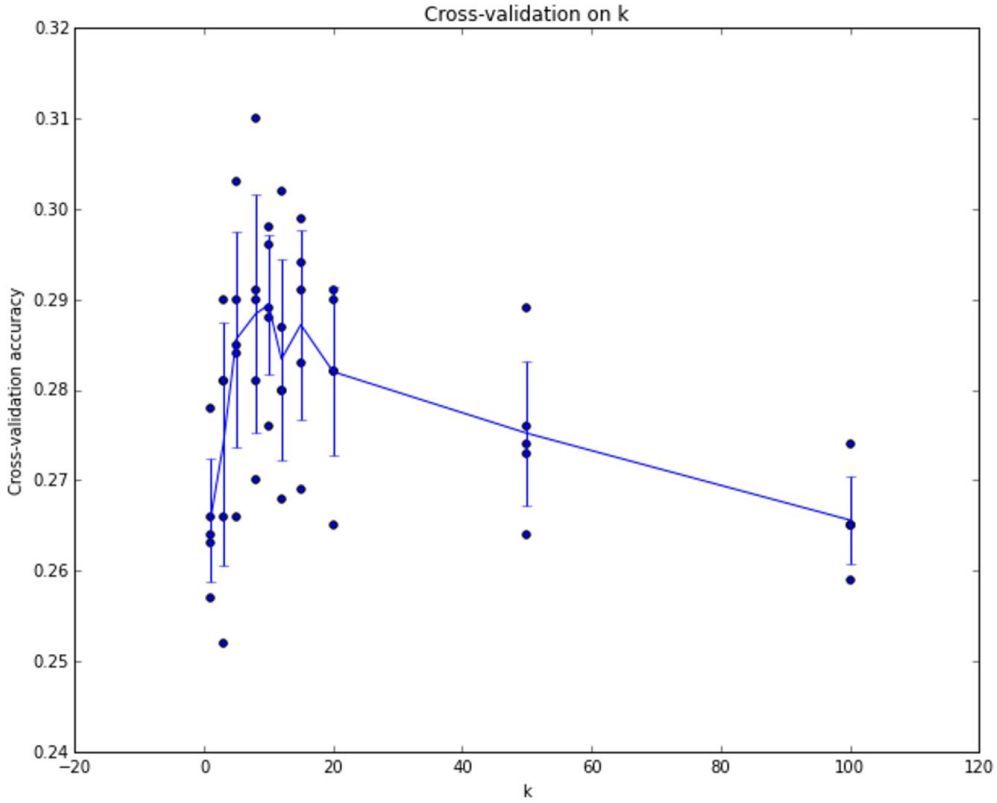
Hyperparameters

Example of 5-fold cross-validation for the value of k.

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

(Seems that k ~ 7 works best for this data)

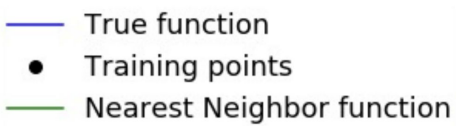


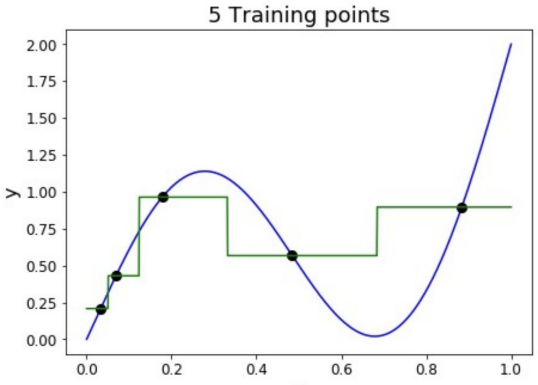
Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

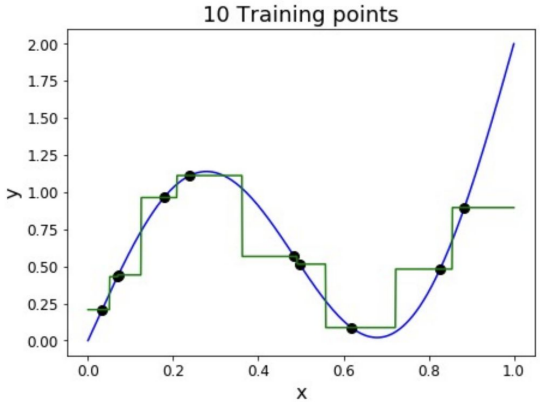
Image Classification

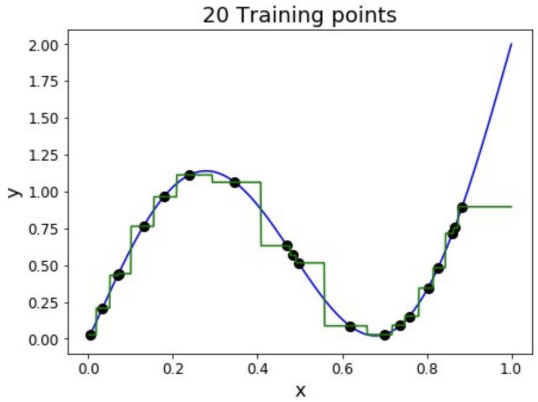
K-Nearest Neighbor: Universal Approximation

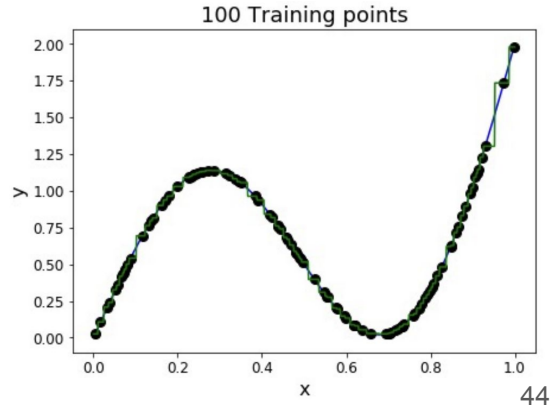
As the number of training samples goes to infinity, nearest neighbor can represent any(*) function!











Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

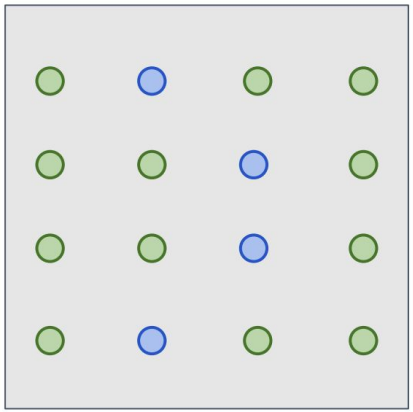
Image Classification

Problem: Curse of Dimensionality

Curse of dimensionality: For uniform coverage of space, number of training points needed grows exponentially with dimension.

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2



Dimensions = 3
Points = 4^3

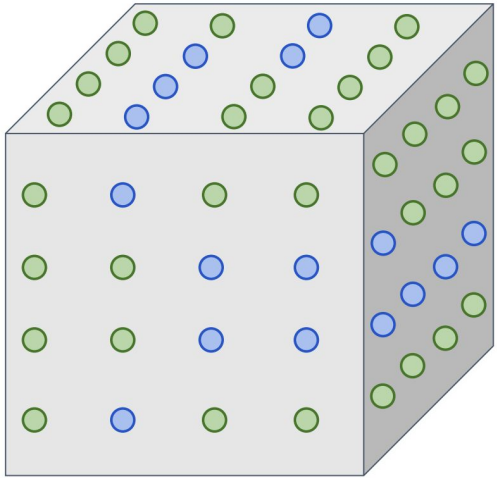


Image Classification

Problem: Curse of Dimensionality

Curse of dimensionality: For uniform coverage of space, number of training points needed grows exponentially with dimension.

Number of possible
32x32 binary images:

$$2^{32x32} \approx 10^{308}$$

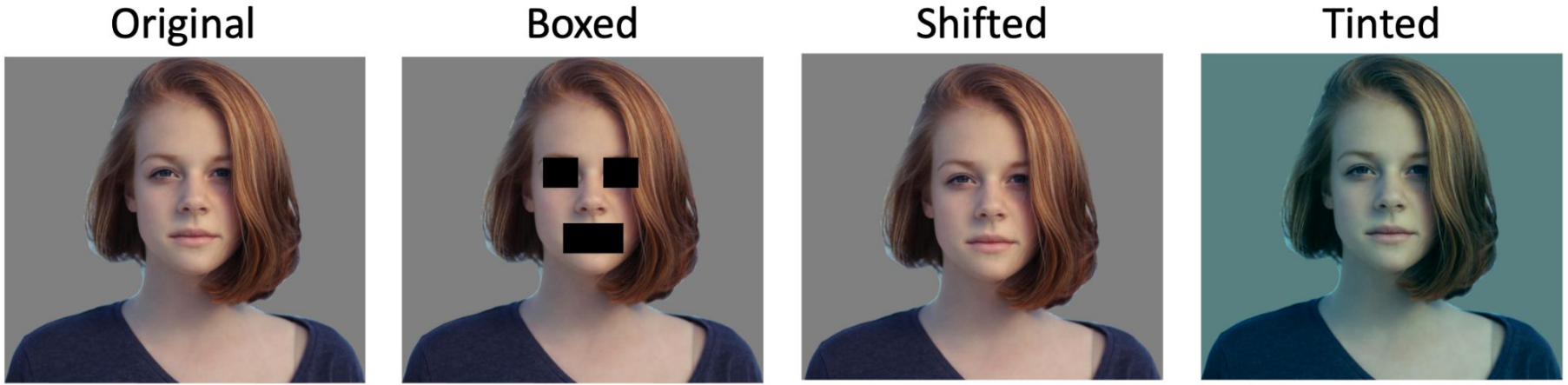
Number of elementary particles in the visible universe:

$$\approx 10^{97}$$

Image Classification

K-Nearest Neighbor on raw pixels is seldom used

- Very slow at test time
- Very data-hungry. It is difficult to get enough data to cover the space of all images.
- Distance metrics on pixels are not informative



(all 3 images have same L2 distance to the one on the left)

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Nearest Neighbor with ConvNet features works well!

Devlin et al, "Exploring Nearest Neighbor Approaches for Image Captioning", 2015

Classical ML

Linear Classifier

Image Classification

Neural Network

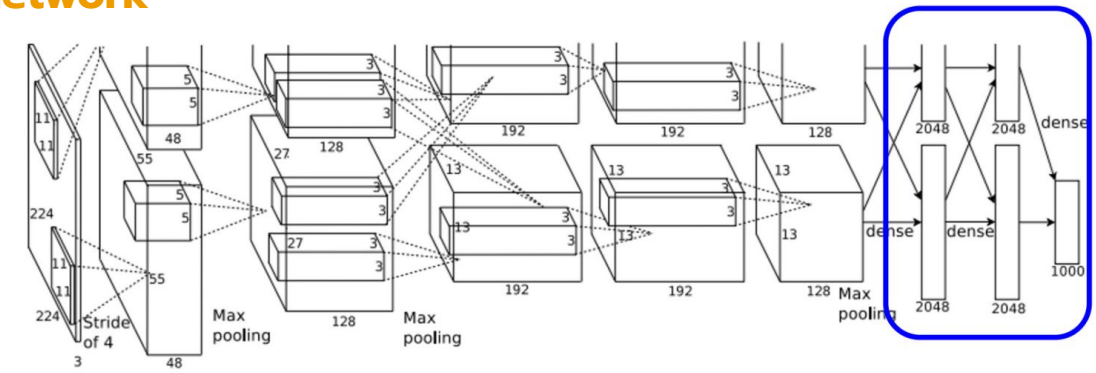
Linear classifiers

This image is CC0 1.0 public domain

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

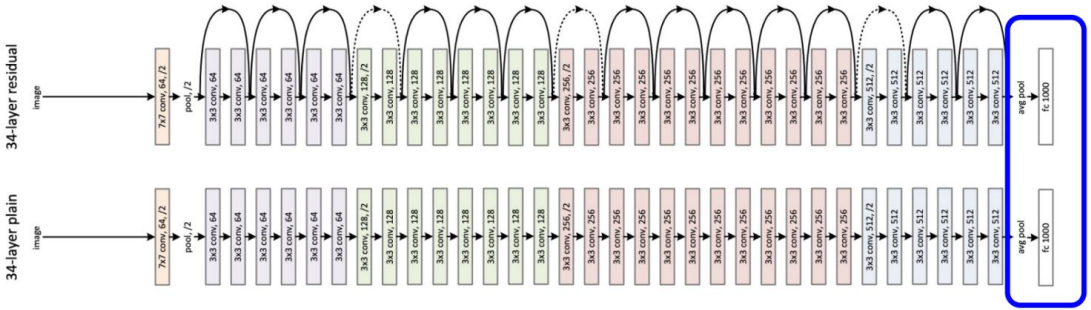
Image Classification

Neural Network



[Krizhevsky et al. 2012]

Linear layers



[He et al. 2015]

Image Classification

CIFAR 10

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

10,000 test images.

How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

Step 3: Load training samples and feed into a model

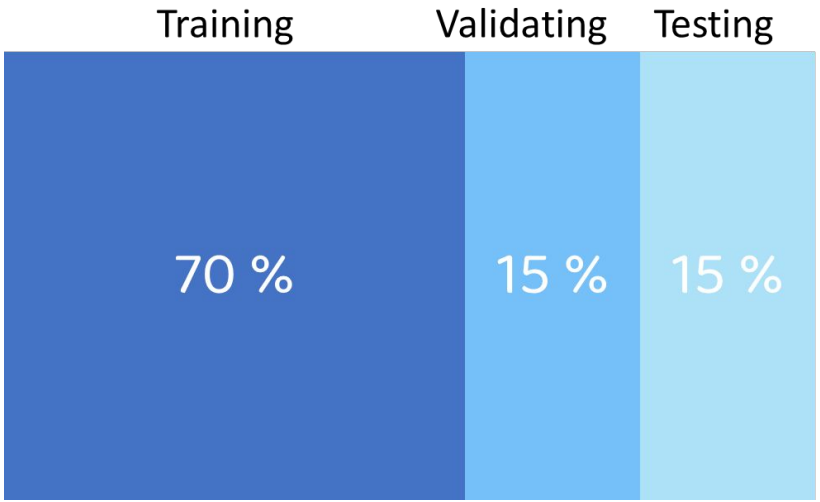
Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset

- Split data into 3 datasets: training, validating, and testing
- Data Augmentation and Normalization



How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset

How to Train a Machine Learning Model

Epoch: # of loops that the entire dataset was seen by the model.

Batch size: # of samples the is being fed into the model at a time.

Iteration (step): # of steps the model takes to see the entire dataset.

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset

We want to train this model

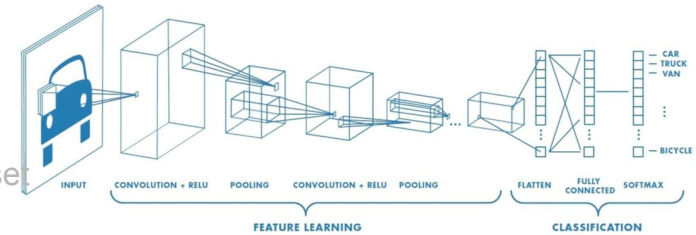
- 50 epoch
- Feed 5 images at a time
- The model will take 20 Iterations (steps) to complete 1 epoch.

Dataset

img_1
img_2
img_3
img_4
img_5
Batch 1

.
.
.
.
.

img_96
img_97
img_98
img_99
img_100
Batch 20



How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

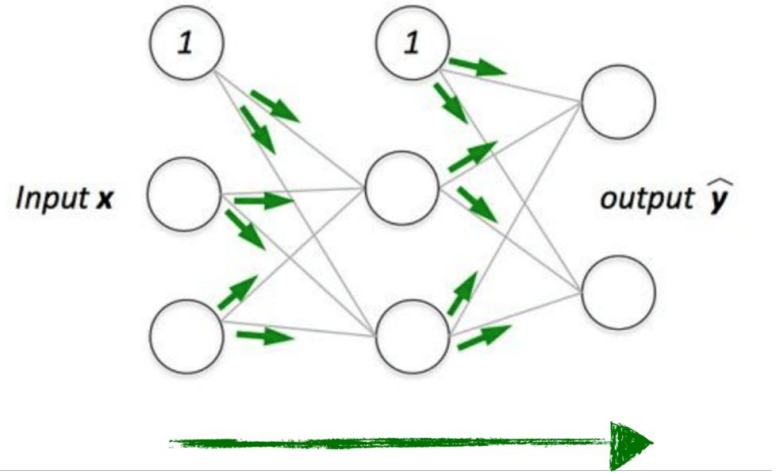
Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset



How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset

Regression problems	Classification problems
Mean Squared Error	Cross Entropy Loss

$$MSE = \frac{1}{N}\sum(t_i - s_i)^2$$

Prediction

Ground Truth

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Classes

Prediction

Ground Truth {0,1}

How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

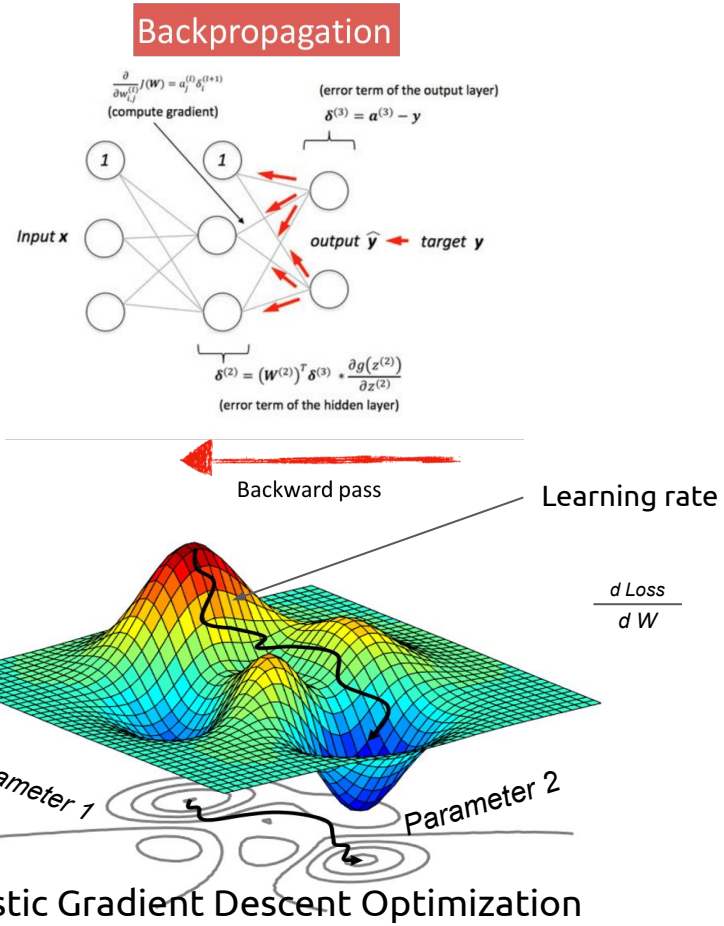
Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset



How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset

Loop until num. Of **steps** reaches.

How to Train a Machine Learning Model

Step 1: Prepare dataset

Step 2: Model's weights(parameters are randomly initialize)

Step 3: Load training samples and feed into a model

Step 4: A model make a prediction

Step 5: Compute loss (cost) function

Step 6: Parameters update (optimization)

Step 7: Validate the model using the validating dataset

Loop until num. Of **epoch** reaches.

Image Classification

Parametric Approach

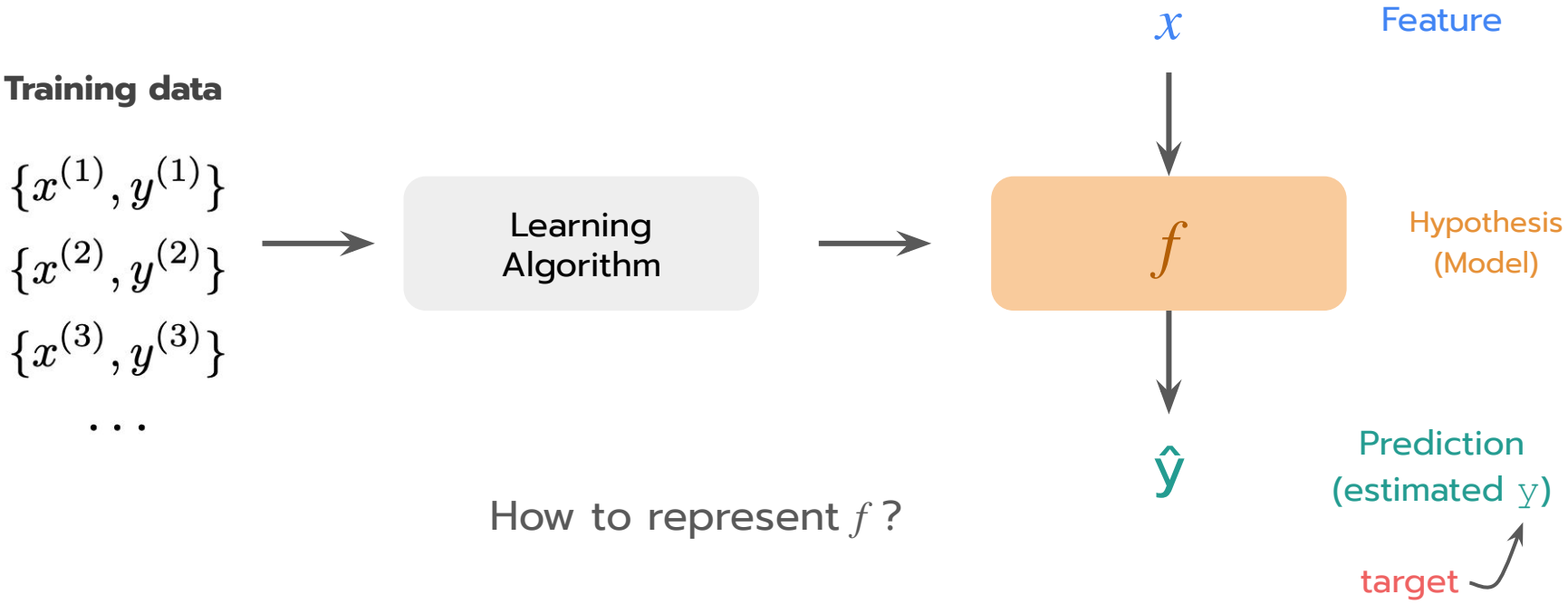


Image Classification

Parametric Approach

Image

Array of 32x32x3 numbers
(3072 numbers total)

$f(x, W)$

W

(learnable) Parameters
or weight

10 numbers giving
class scores

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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

(learnable) Parameters or weight

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Parametric Approach: Linear Classifier

Image

Array of 32x32x3 numbers
(3072 numbers total)

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

(3072,)

(10,)

(10, 3072)

$f(x, W)$

10 numbers giving class scores

W

(learnable) Parameters or weight

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Parametric Approach: Linear Classifier

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

(3072,) (10,) (10, 3072) (10,)

Image

Array of 32x32x3 numbers
(3072 numbers total)

$f(x, W)$

10 numbers giving class scores

W

(learnable) Parameters or weight

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Parametric Approach: Linear Classifier

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

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

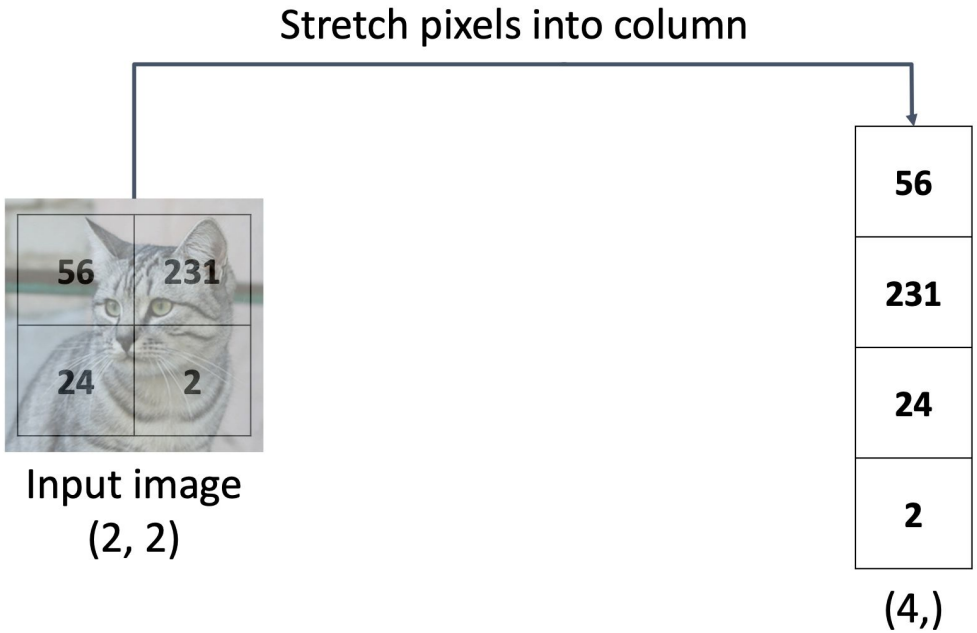


Image Classification

Parametric Approach: Linear Classifier

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

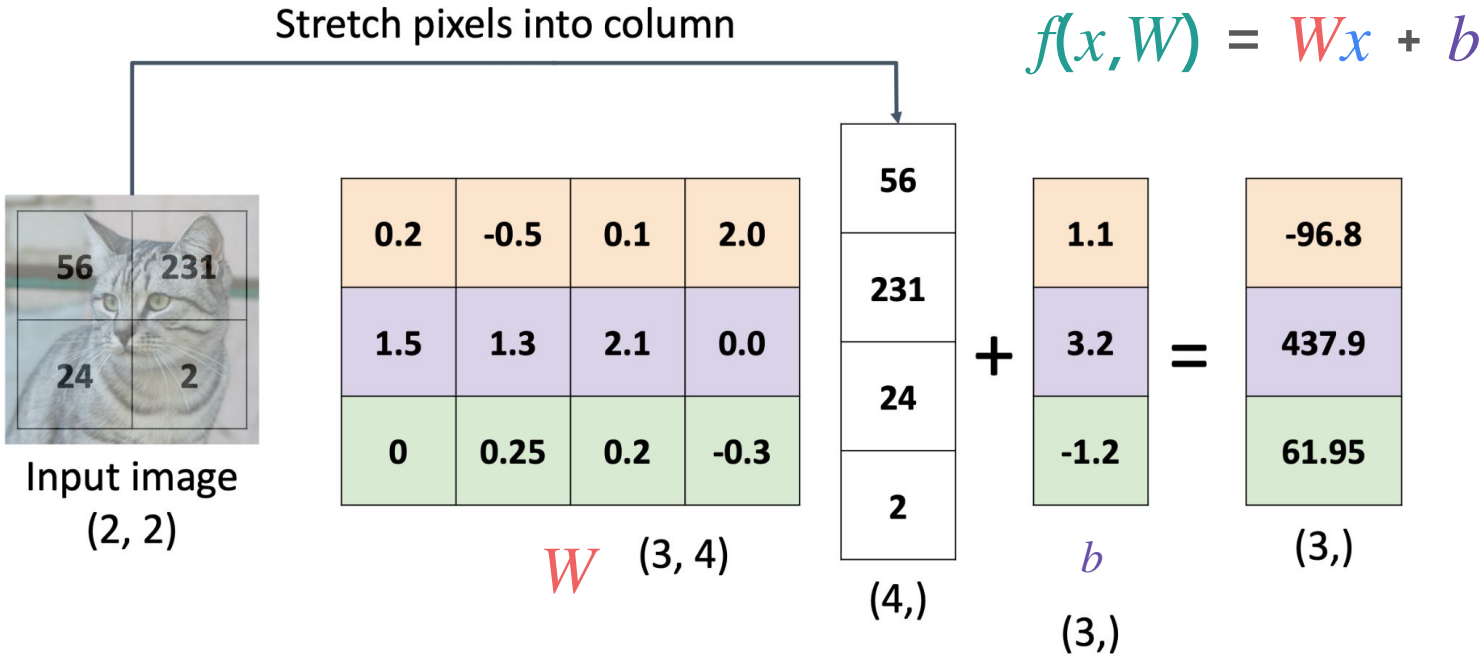


Image Classification

Interpreting a Linear Classifier: Algebraic Viewpoint

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

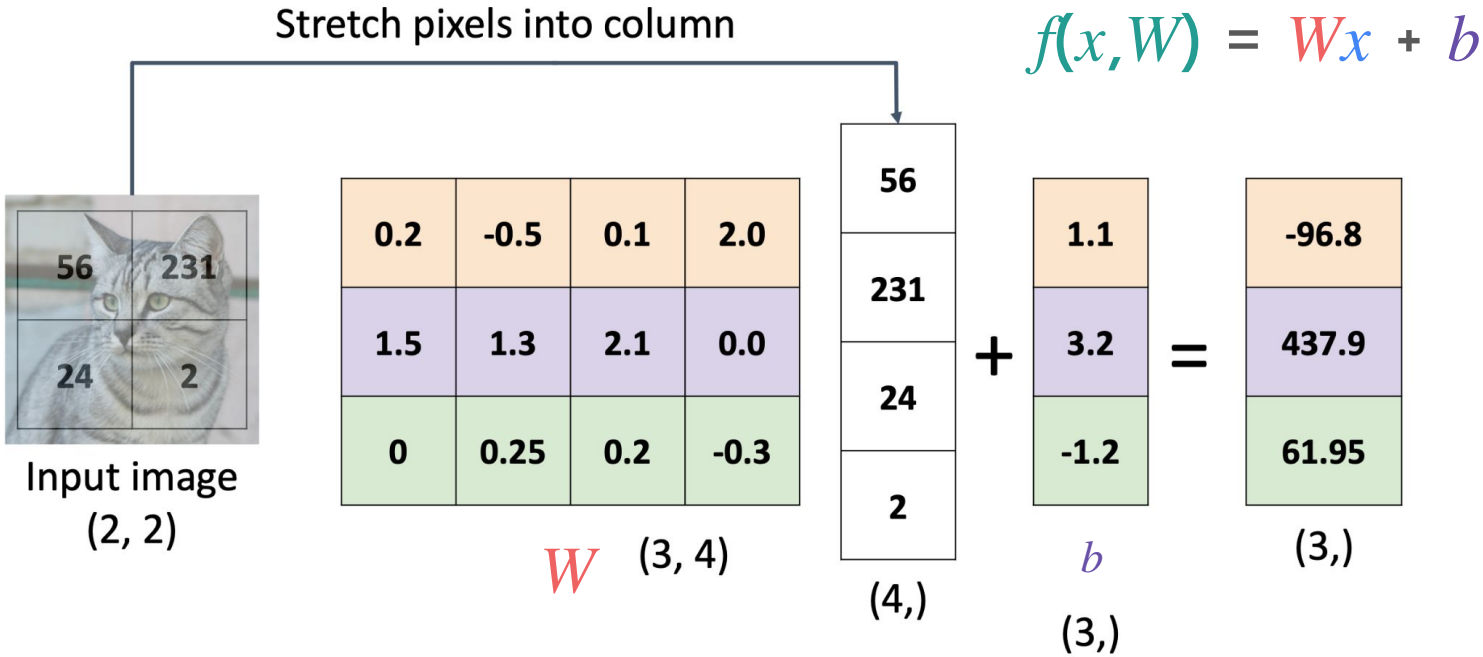


Image Classification

Linear Classifier: Bias Trick

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

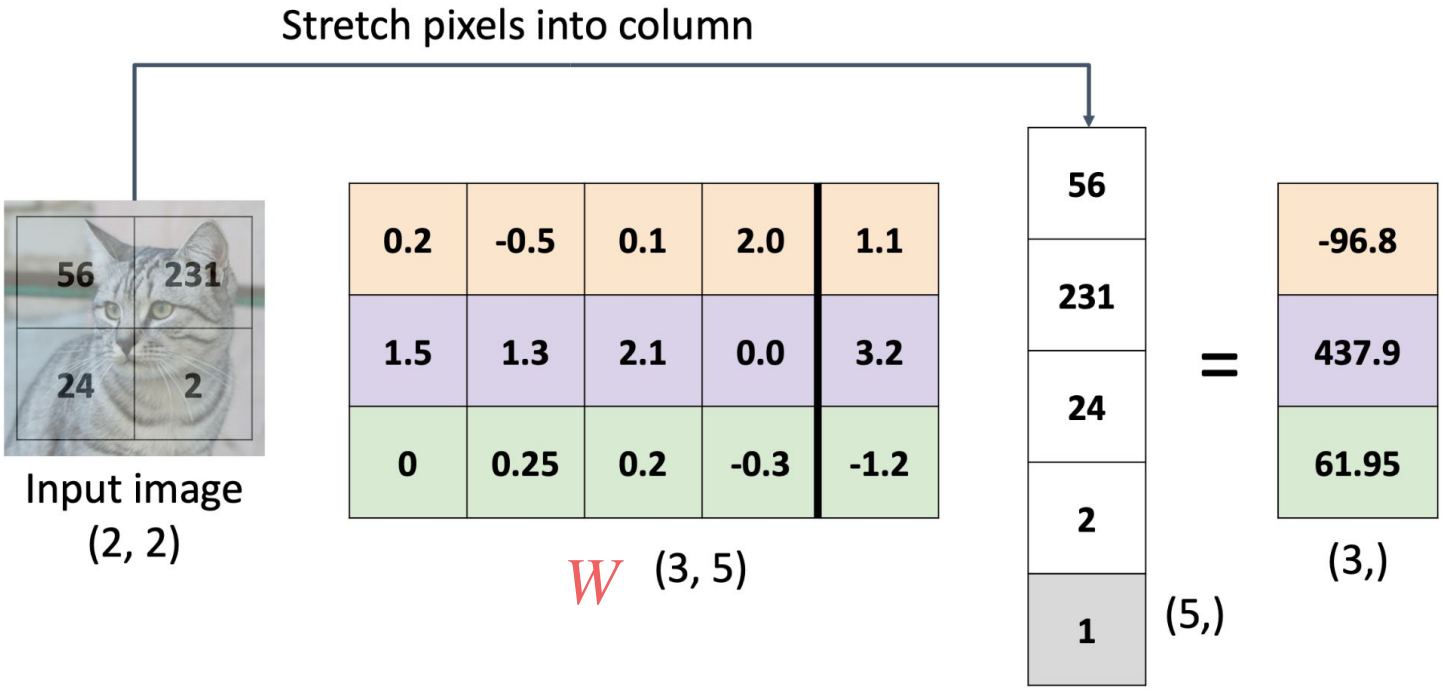
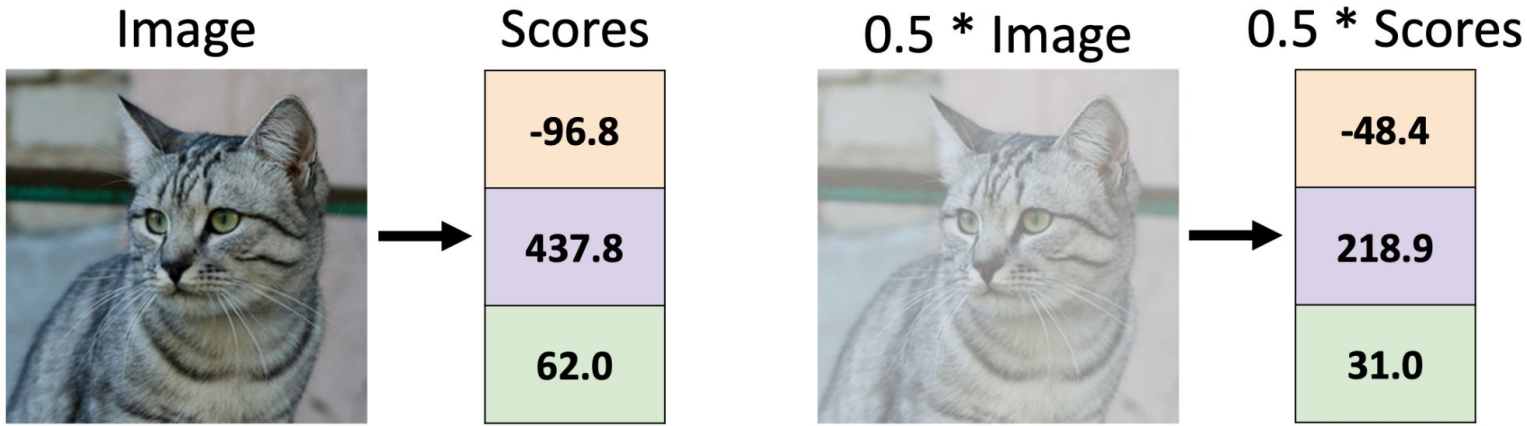


Image Classification

Linear Classifier: Predictions are Linear!

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

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



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

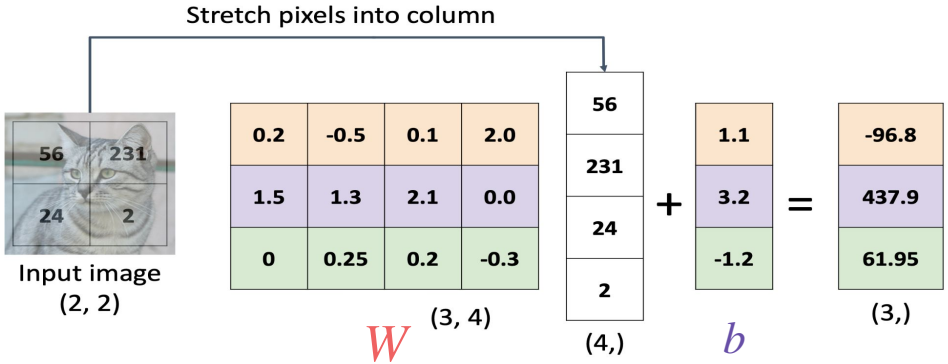
Image Classification

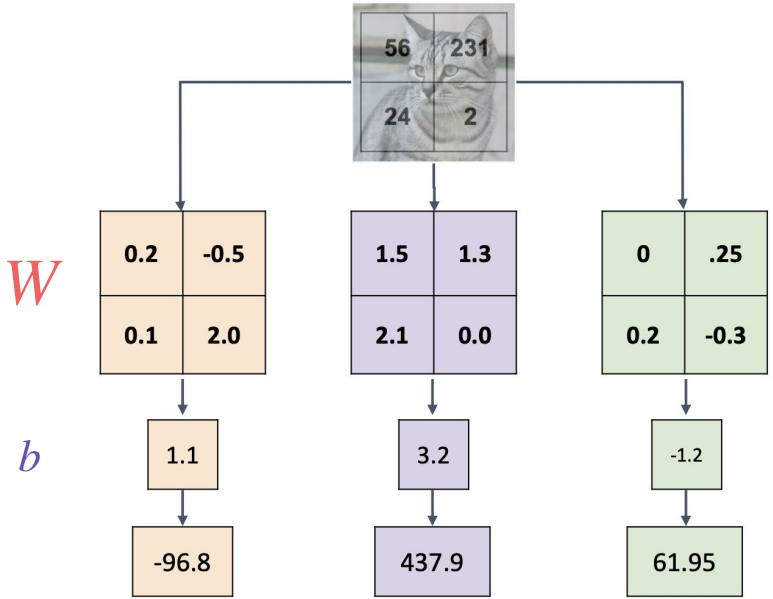
Interpreting a Linear Classifier

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

Algebraic Viewpoint

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



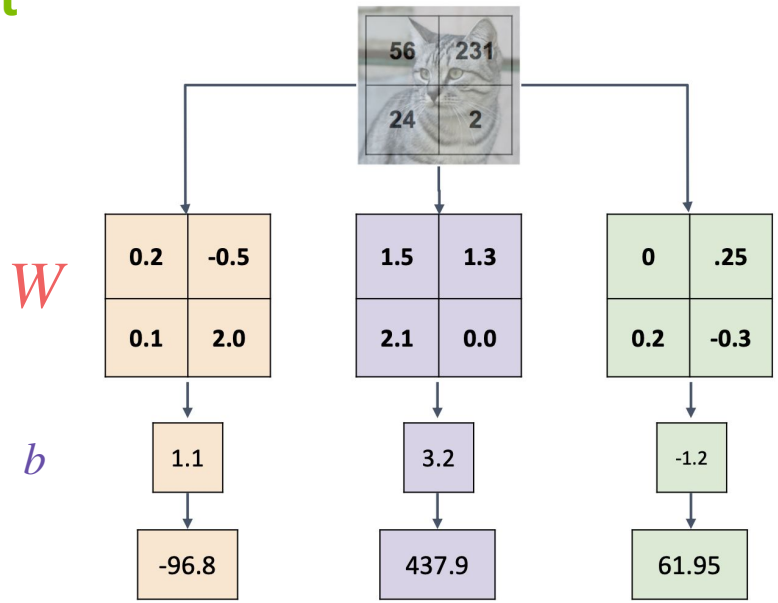


Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Interpreting a Linear Classifier: Visual Viewpoint

We can try to **visualize** the matrices as images.

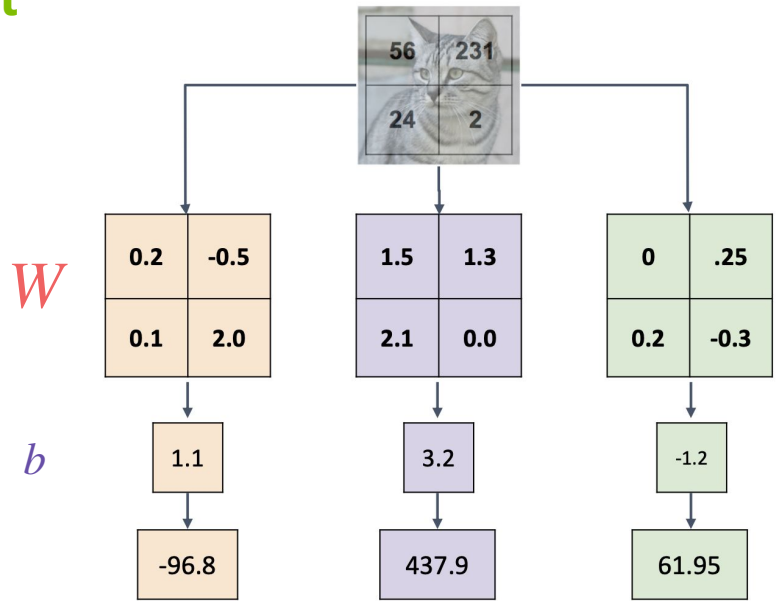


Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Interpreting a Linear Classifier: Visual Viewpoint

Linear classifier has one **"template"** per category



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Interpreting a Linear Classifier: Visual Viewpoint

Linear classifier has one **"template"** per category

A single template cannot capture multiple modes of the data

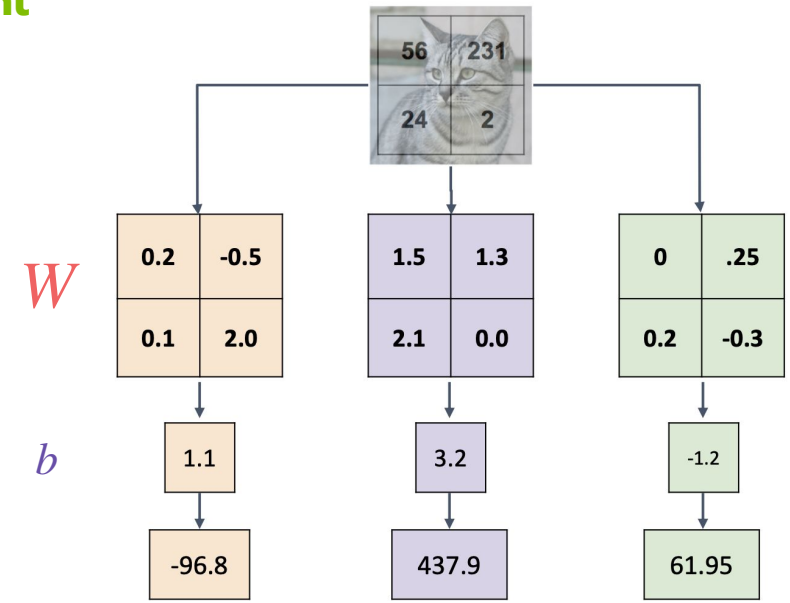


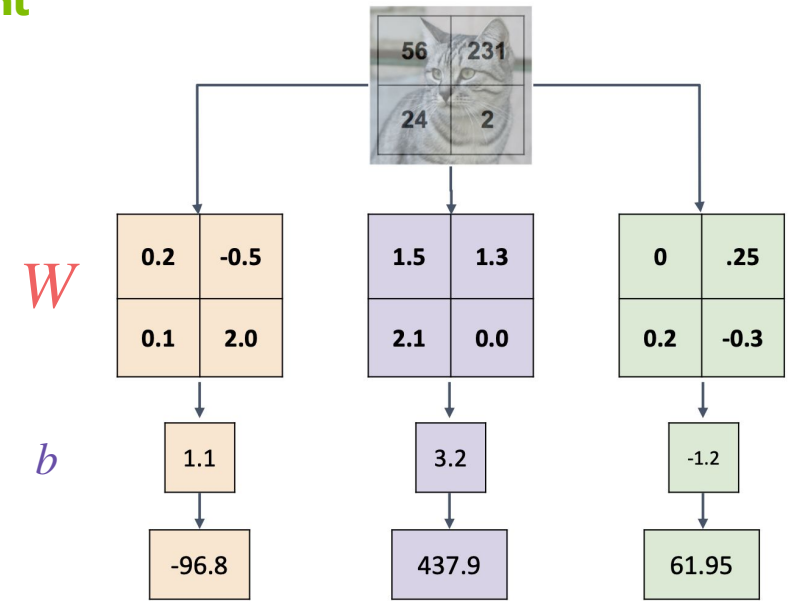
Image Classification

Interpreting a 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!



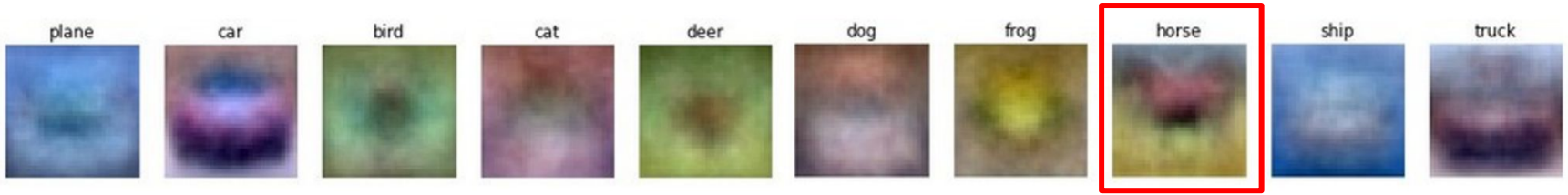
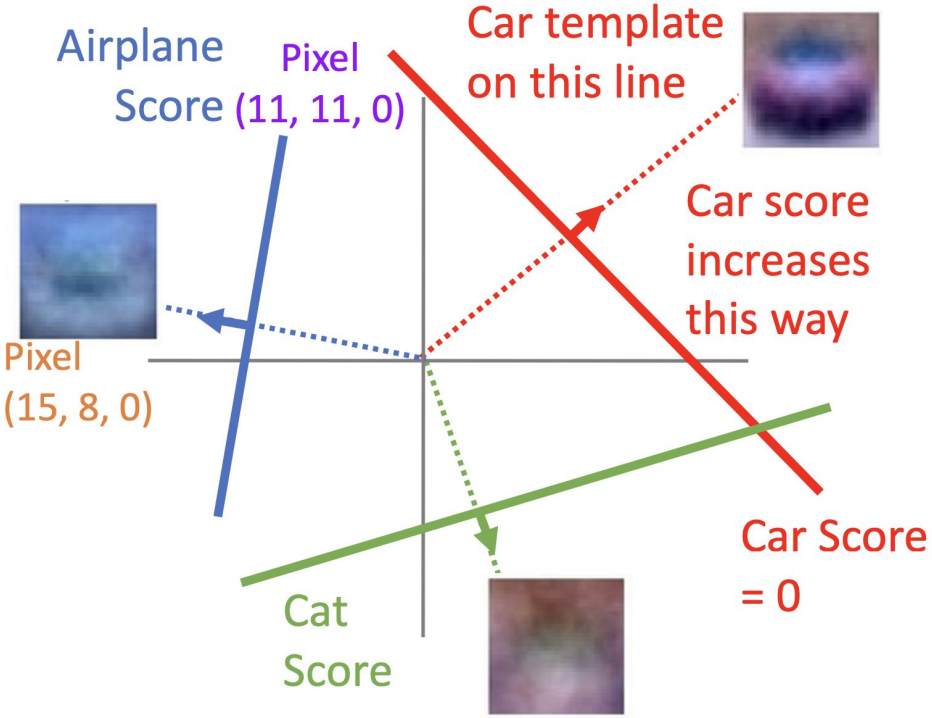


Image Classification

Interpreting a Linear Classifier: Geometric Viewpoint



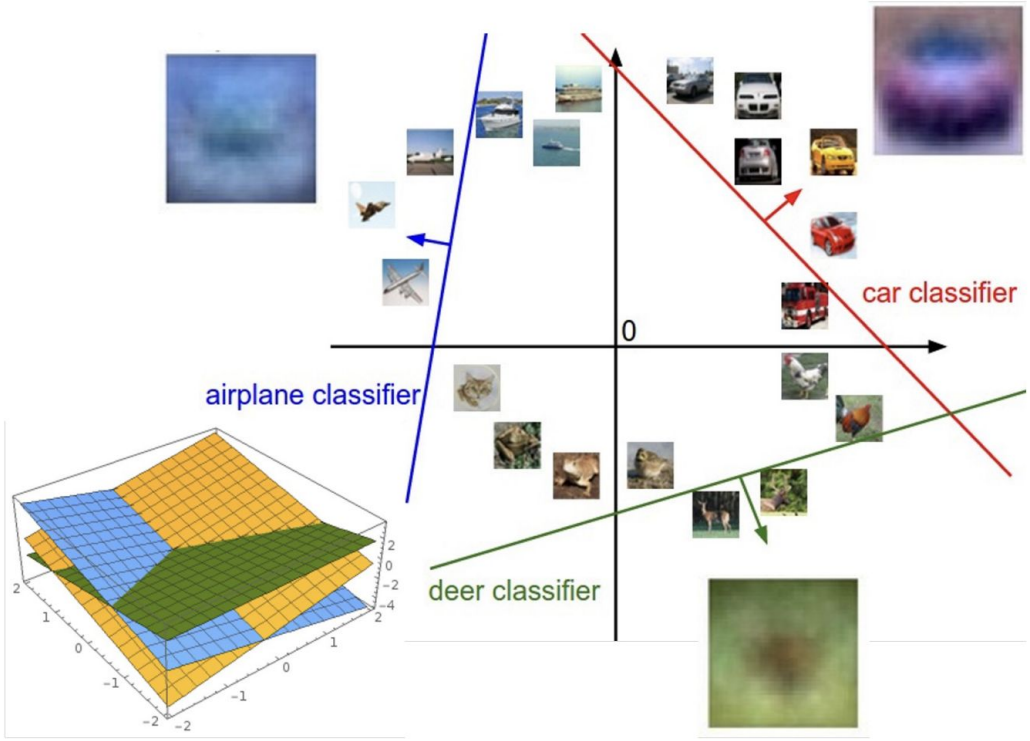
$$f(x, W) = Wx + b$$

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Interpreting a Linear Classifier: Geometric Viewpoint



Plot created using Wolfram Cloud

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

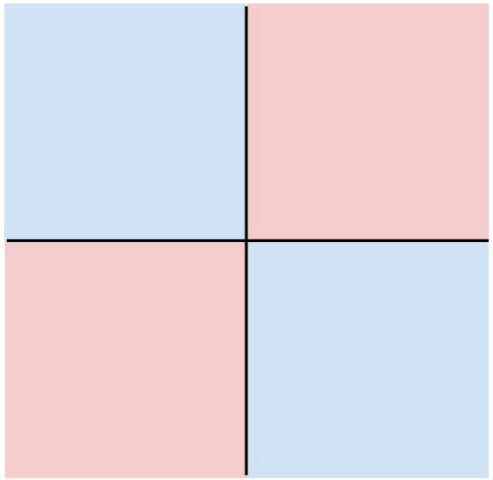
Array of **32x32x3** numbers
(3072 numbers total)

Image Classification

Hard Cases for a Linear Classifier

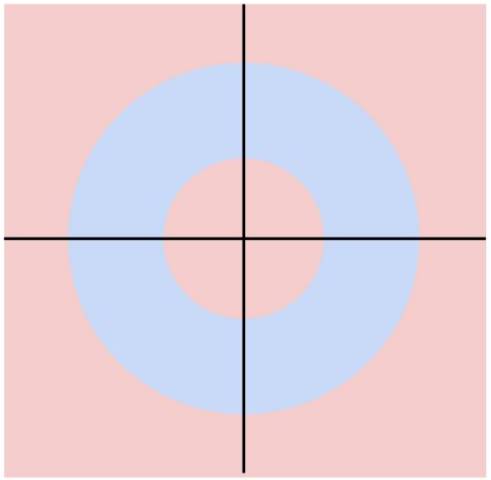
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else

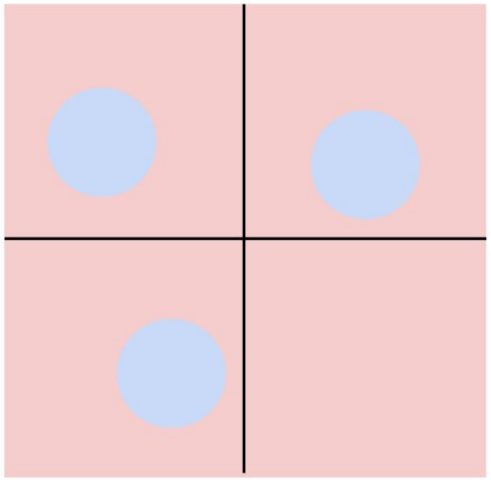
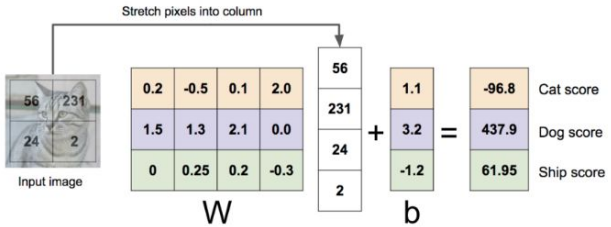


Image Classification

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

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

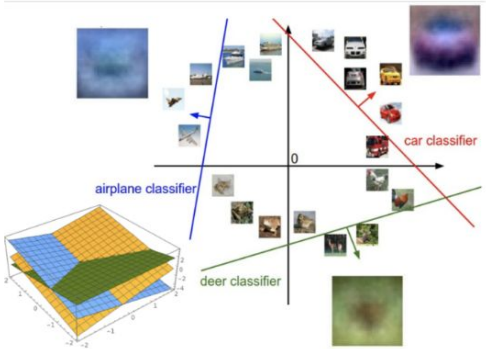


Visual Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space



Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: Score Function

So Far: Defined a linear **score function**

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

But how can we actually **choose a good** W ?

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: choose a good W

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

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**)

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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:

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

Image Classification

How to represent class labels?

Given a dataset of examples

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

Where x_i is image and
y_i is (integer) label

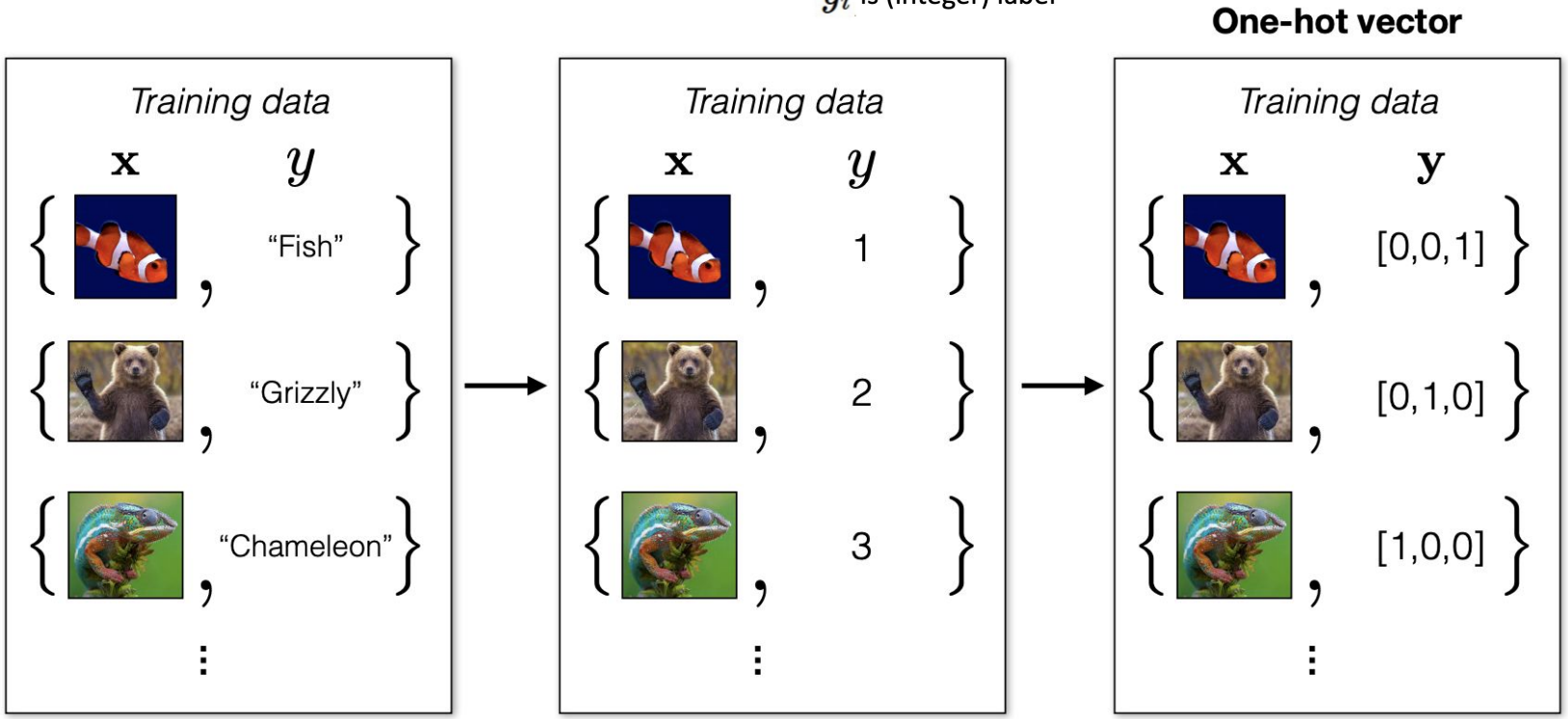


Image Classification

Linear Classifier:

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier:

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



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

A loss function tells how good our current classifier is

Given a dataset of examples

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

Where x_i is image and
y_i (integer) label

Loss over the dataset is a average of loss over examples:

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

Image Classification

Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



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)
where x_i s the image and
where y_i s the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Image Classification

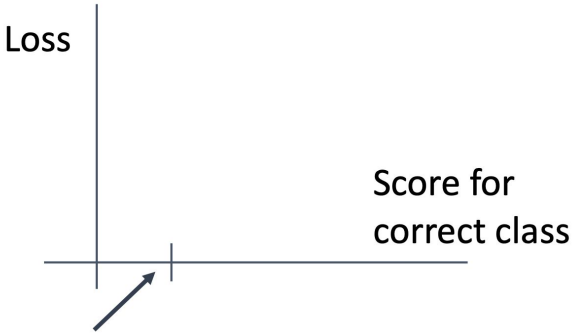
Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



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

Interpreting Multiclass SVM loss:



$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

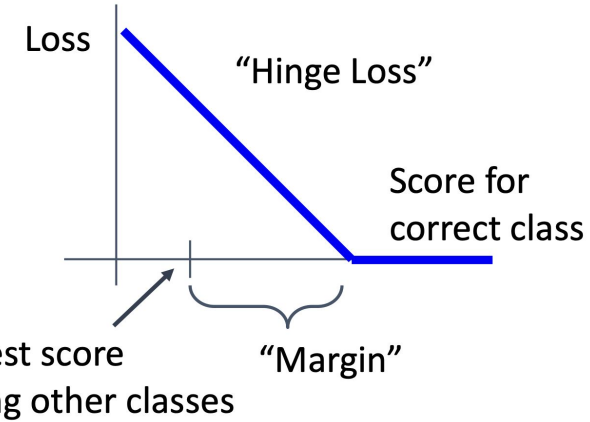
Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



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

Interpreting Multiclass SVM loss:



$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9		

Given an example (x_i, y_i)
where x_i s the image and
where y_i s the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Given an example (x_i, y_i)
where x_i s the image and
where y_i s the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: Multiclass SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Given an example (x_i, y_i)
where x_i s the image and
where y_i s the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Loss over full dataset is average:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= 5.27

Image Classification

Linear Classifier: 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?

Image Classification

Linear Classifier: 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!

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Linear Classifier: Multiclass SVM loss

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



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

Image Classification

Linear Classifier: Multiclass SVM loss

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



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

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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...

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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

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

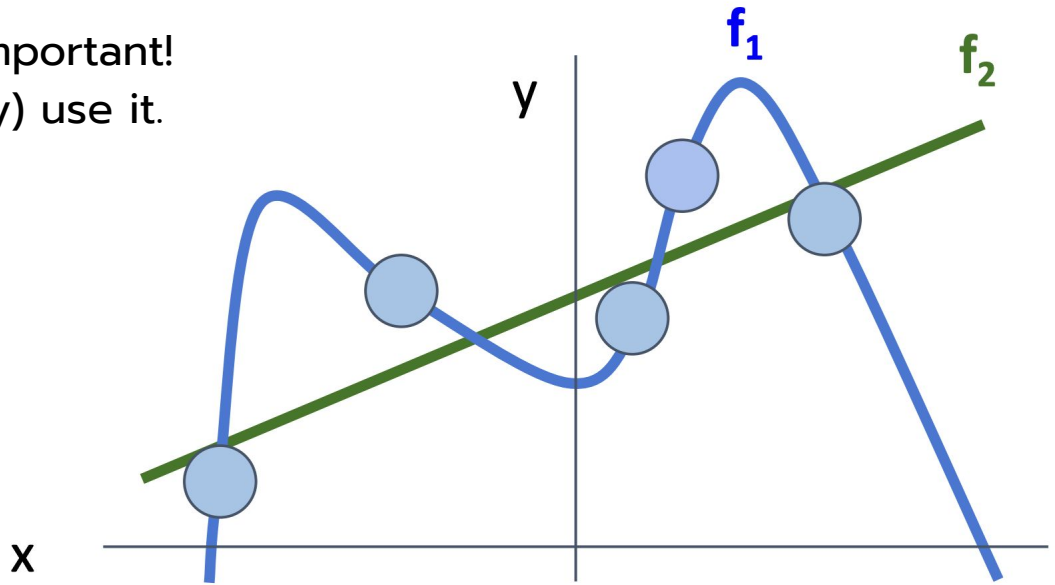
L2 regularization likes to "spread out" the weights

Image Classification

Regularization: Prefer Simpler Models

f_1 is not a linear model; could be polynomial regression, etc.

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



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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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

Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

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

cat	**3.2**
car	5.1
frog	-1.7

Unnormalized log-probabilities / logits

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

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

Softmax function

Image Classification

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0 unnormalized probabilities	normalize	Probabilities must sum to 1 probabilities
cat	3.2		24.5		0.13
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

Image Classification

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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1	probabilities
cat	3.2		24.5		0.13	
car	5.1		164.0		0.87	
frog	-1.7		0.18		0.00	

unnormalized probabilities

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

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

Image Classification

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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1	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
			unnormalized probabilities		probabilities		

Kullback–Leibler divergence

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

Image Classification

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

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

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1		Compare		Correct probs
cat	3.2	→	24.5	→	0.13	→		←	1.00
car	5.1		164.0		0.87		Cross Entropy		0.00
frog	-1.7		0.18		0.00		$H(P,Q) = H(p) + D_{KL}(P\|Q)$		0.00
			unnormalized probabilities		probabilities				

Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

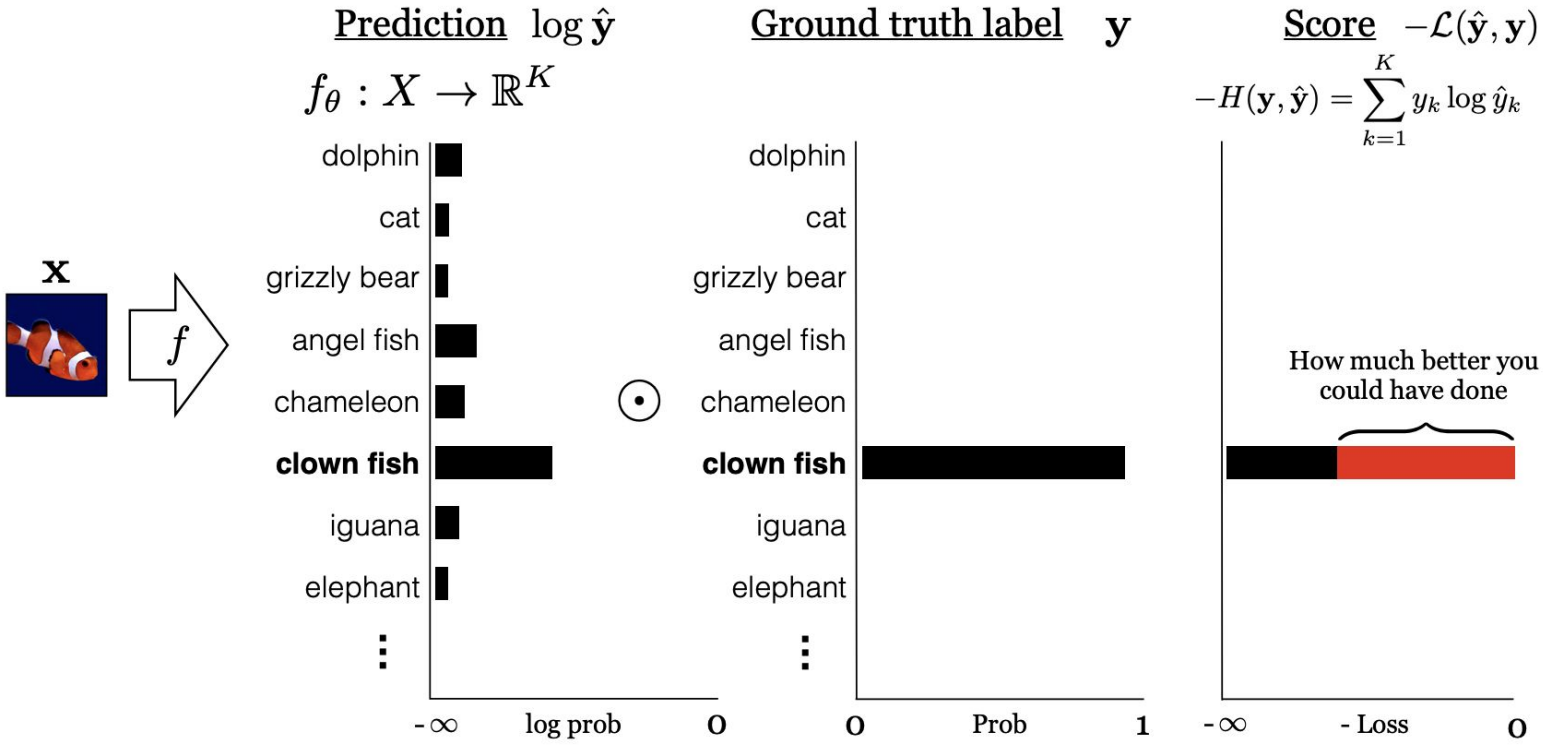


Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

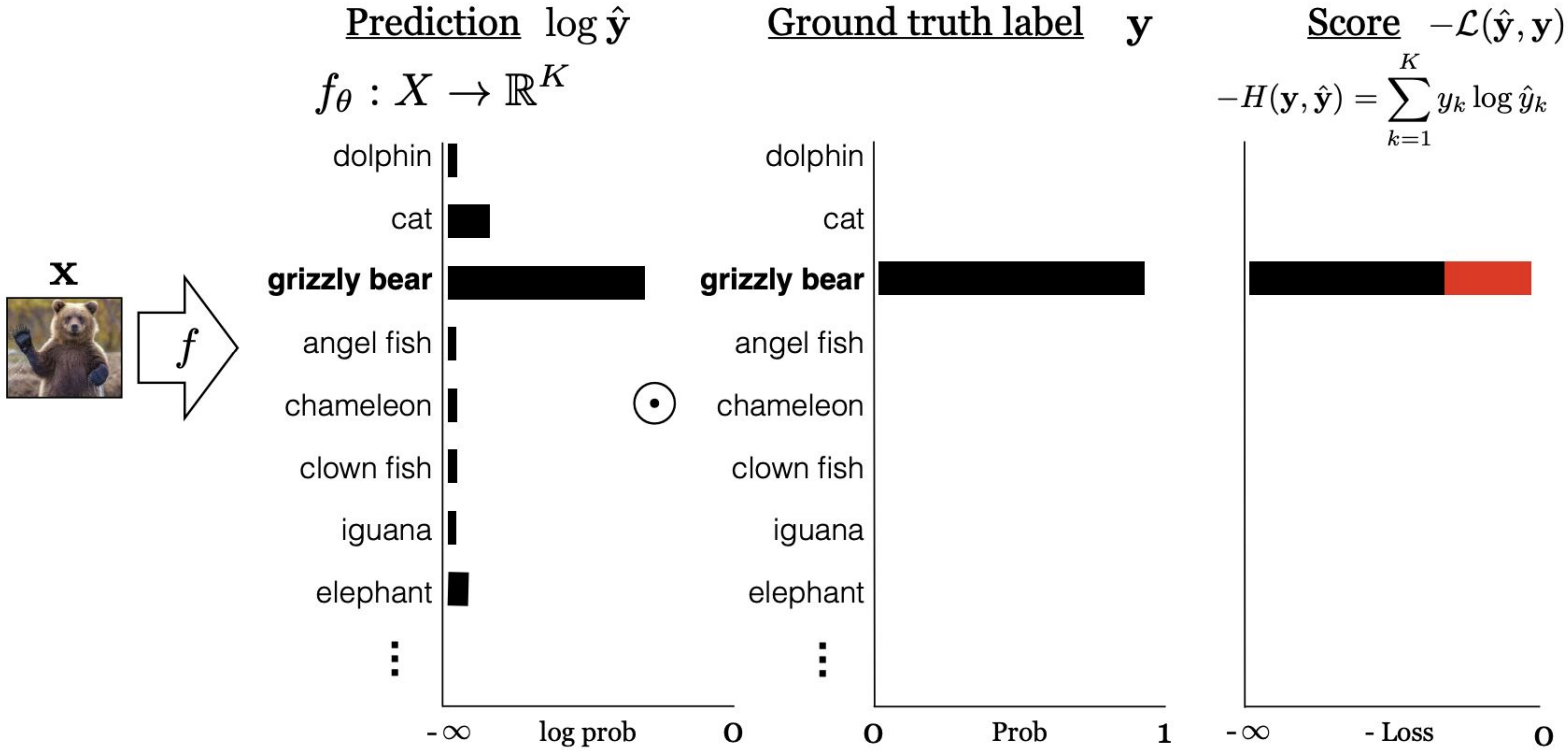


Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

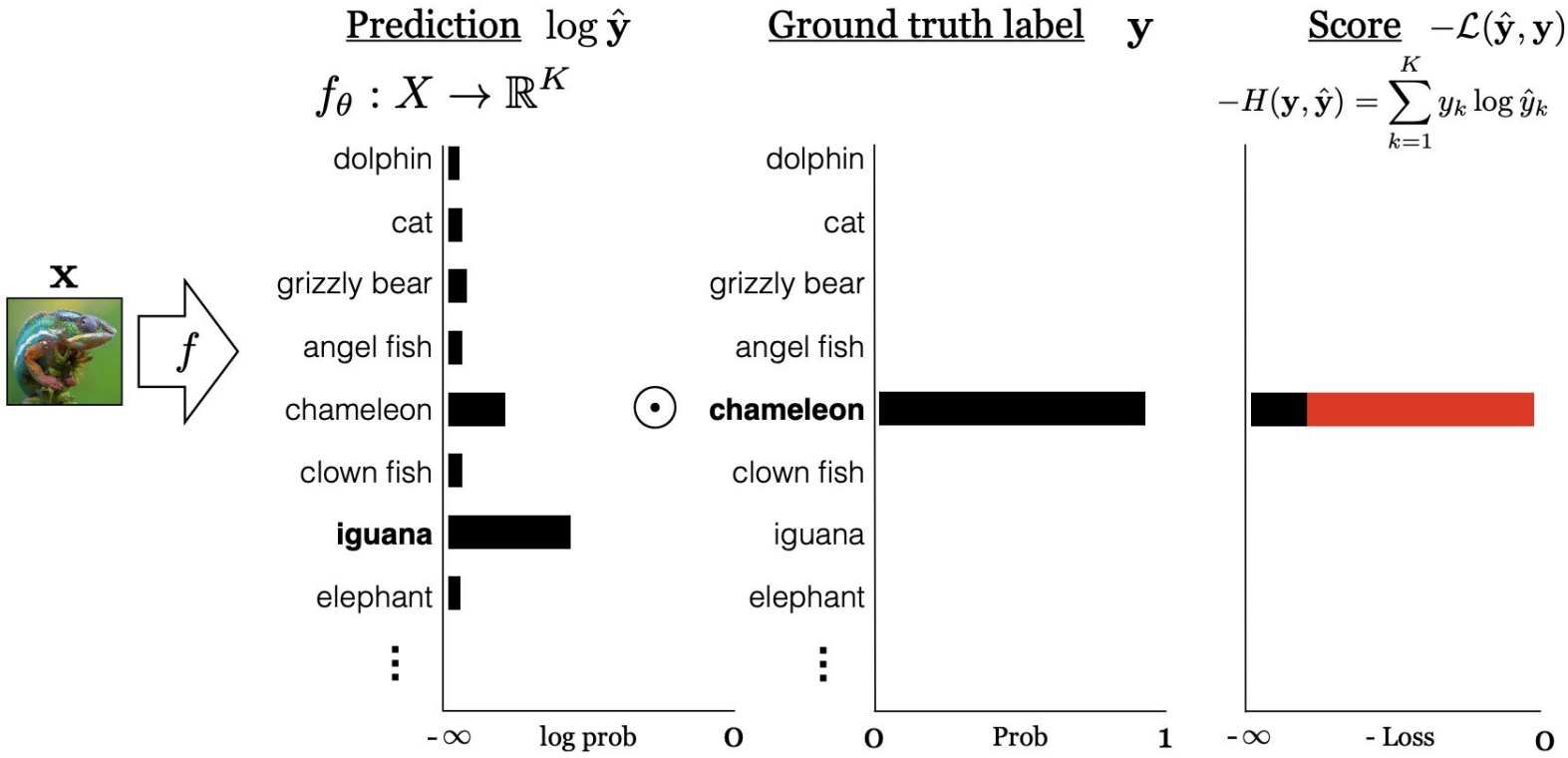


Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

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

cat **3.2**

car 5.1

frog -1.7

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

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

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?

Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

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

cat **3.2**

car 5.1

frog -1.7

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

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

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 +ve infinity

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

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

cat **3.2**

car 5.1

frog -1.7

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

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

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?

Image Classification

Cross-Entropy Loss (Multinomial Logistic Regression)

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

cat **3.2**

car 5.1

frog -1.7

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

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

Softmax function

Maximize probability of correct class

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

Putting it all together:

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

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

Uniform distribution over all classes

A: -log(C)

log(10) ≈ 2.3

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Cross-Entropy vs SVM Loss

Cross-Entropy	SVM Loss
$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$	$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and y = 0

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

Image Classification

Cross-Entropy vs SVM Loss

Cross-Entropy	SVM Loss
$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$	$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and y = 0

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

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

Image Classification

Cross-Entropy vs SVM Loss

Cross-Entropy	SVM Loss
$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$	$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and y = 0

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

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Cross-Entropy vs SVM Loss

Cross-Entropy	SVM Loss
$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$	$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and y = 0

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

A: cross-entropy loss will change

(it will try to push the correct (to +ve inf) and incorrect (to -ve inf) class away from each other)

SVM loss will stay the same.

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

Cross-Entropy vs SVM Loss

Cross-Entropy	SVM Loss
$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$	$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Assume scores:
[20, -2, 3]
[20, 9, 9]
[20, -100, -100]
and y = 0

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

Image Classification

Cross-Entropy vs SVM Loss

Cross-Entropy	SVM Loss
$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$	$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Assume scores:
[20, -2, 3]
[20, 9, 9]
[20, -100, -100]
and y = 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.

Slide adopted from: Justin Johnson, EECS 498-007 / 598-005: Deep Learning for Computer Vision, University of Michigan, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Image Classification

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

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