Lecture 2:

Image Classification with Linear Classifiers

Administrative: Assignment 1

Will be out **Wednesday 4/9, due 4/23 by 11:59 PM**

- K-Nearest Neighbor
- Linear classifiers: Softmax
- Two-layer neural network
- Image features
- Deep neural network and optimizers

Administrative: Course Project

Project proposal due 4/25 (Friday) 11:59 pm

Contact us on Ed, each project team will have a TA assigned to them for future questions

your assigned TA for initial guidance (Canvas -> People -> Groups)

Use the Google Form to find project partners (will be posted later today)

“Is X a valid project for 231n?” --- Ed private post / TA Office Hours

More info on the website

Administrative: Discussion Sections

This Friday 12:30 pm-1:20 pm, in person at NVIDIA Auditorium, remote on Zoom (recording will be made available)

Python / Numpy, Google Colab

Presenter: Emily Jin (TA) with Assistance from Matthew Jin (TA)

Syllabus

Deep Learning Basics

Data-driven approaches
Linear classification
K-Nearest Neighbor
Loss Functions
Optimization
Backpropagation
Multi-layer Perceptrons
Neural Networks
Activation Functions
Data Augmentation

Perceiving and Understanding the Visual World

Transfer Learning
Optimizers
Convolutions
PyTorch
RNNs / Attention / Transformers
Normalization Layers
Architecture Design
Video Understanding
Vision and Language
3D Vision
Object Detection and Segmentation

Reconstructing and Interacting with the Visual World

Style Transfer
Generative Models
Self-supervised Learning
Image Generation
Robotics and Embodied AI

Human-centered AI
Fairness & Ethics

Image Classification

A Core Task in Computer Vision

Today:

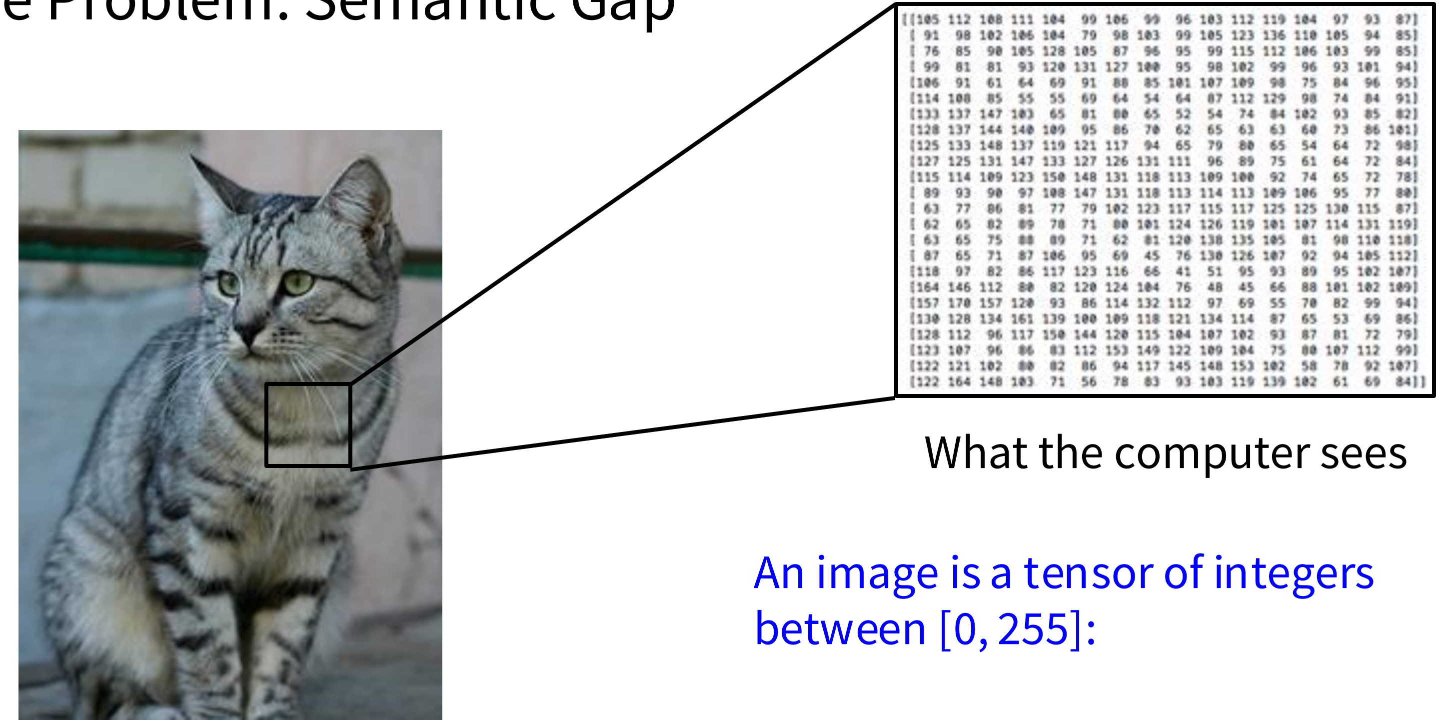
- The image classification task
- Two basic data-driven approaches to image classification
    - K-nearest neighbor and linear classifier

Image Classification: A core task in Computer Vision

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}

→ cat

The Problem: Semantic Gap

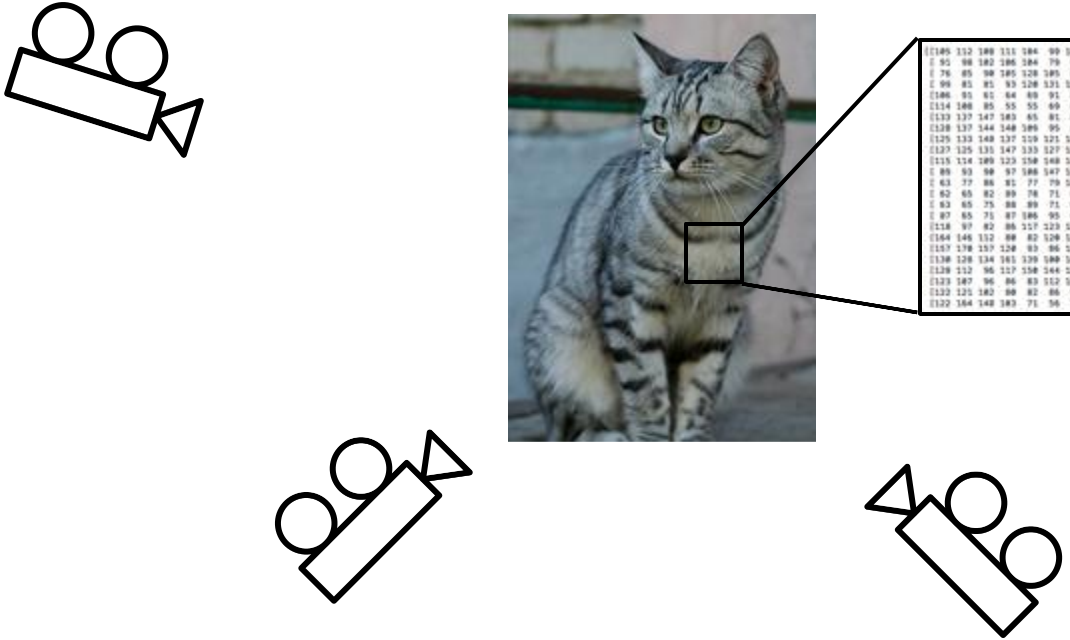


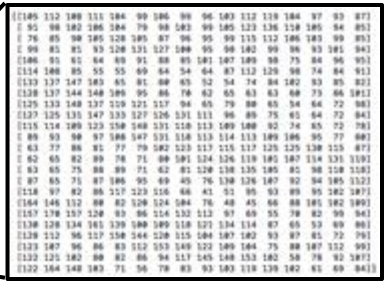
What the computer sees

An image is a tensor of integers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Viewpoint variation





All pixels change when the camera moves!

Challenges: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

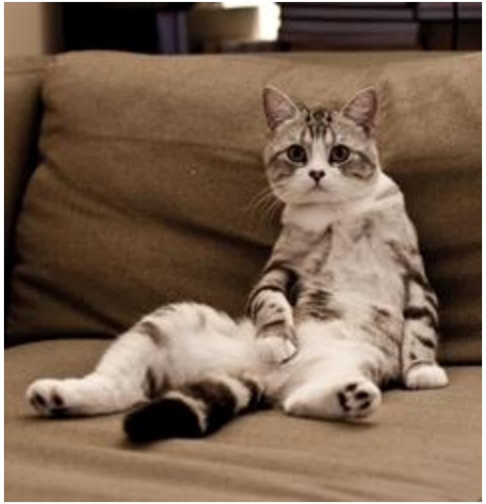
Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Challenges: Deformation



Challenges: Intraclass variation

This image is CC0 1.0 public domain

Challenges: Context

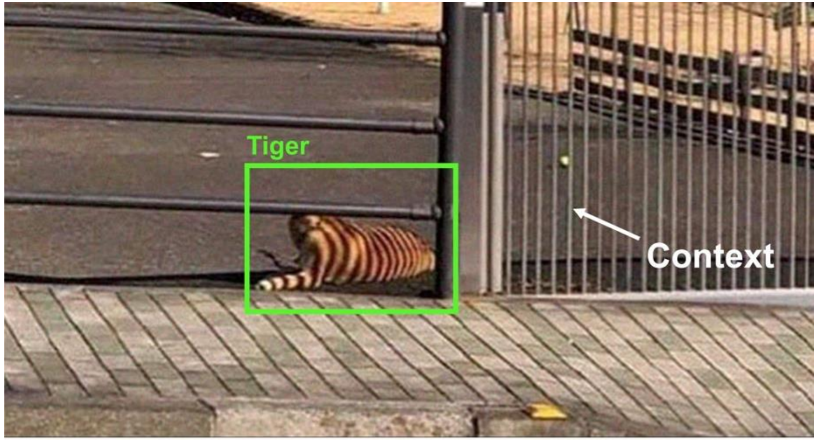


Image source: https://www.linkedin.com/posts/ralph-aboujaoude-diaz-40838313_technology-artificialintelligence-computervision-activity-6912446088364875776-h-Iq?utm_source=linkedin_share&utm_medium=member_desktop_web

Modern computer vision algorithms

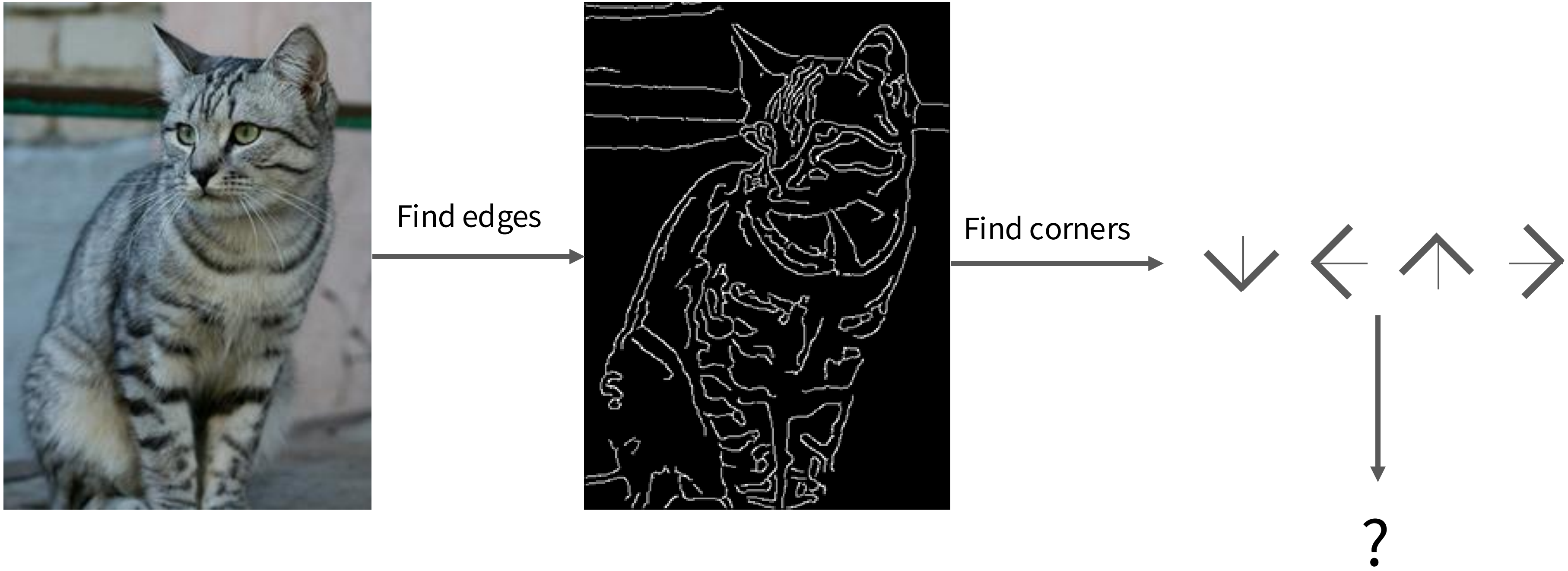
An image classifier

```
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.

Attempts have been made



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

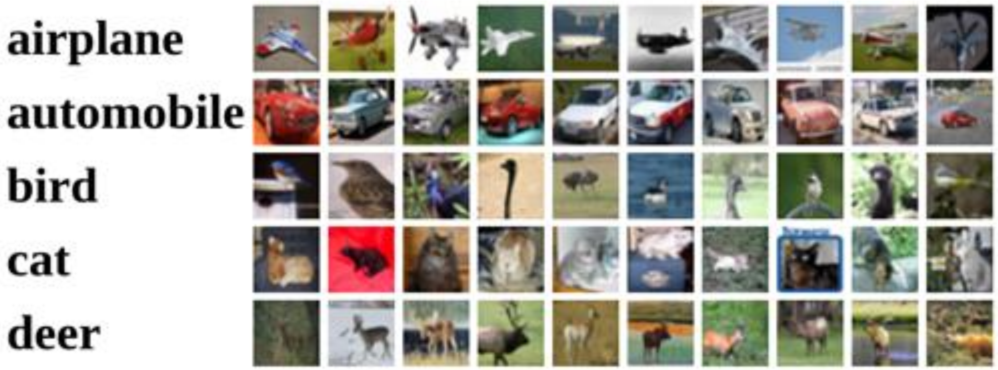
Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning algorithms 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



Nearest Neighbor Classifier

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

First classifier: Nearest Neighbor

deer bird plane cat car

Training data with labels

?

query data

Distance Metric | ,  | $\to \mathbb{R}$

Distance Metric to compare images

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

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

add → 456

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
```

```python
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
```

Nearest Neighbor classifier

Memorize training data

```python
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
```

Nearest Neighbor classifier

For each test image:
- Find closest train image
- Predict label of nearest image

```
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
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

Ans: Train O(1), predict O(N)

This is bad: we want classifiers that are fast at prediction; slow for training is ok

```python
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
```

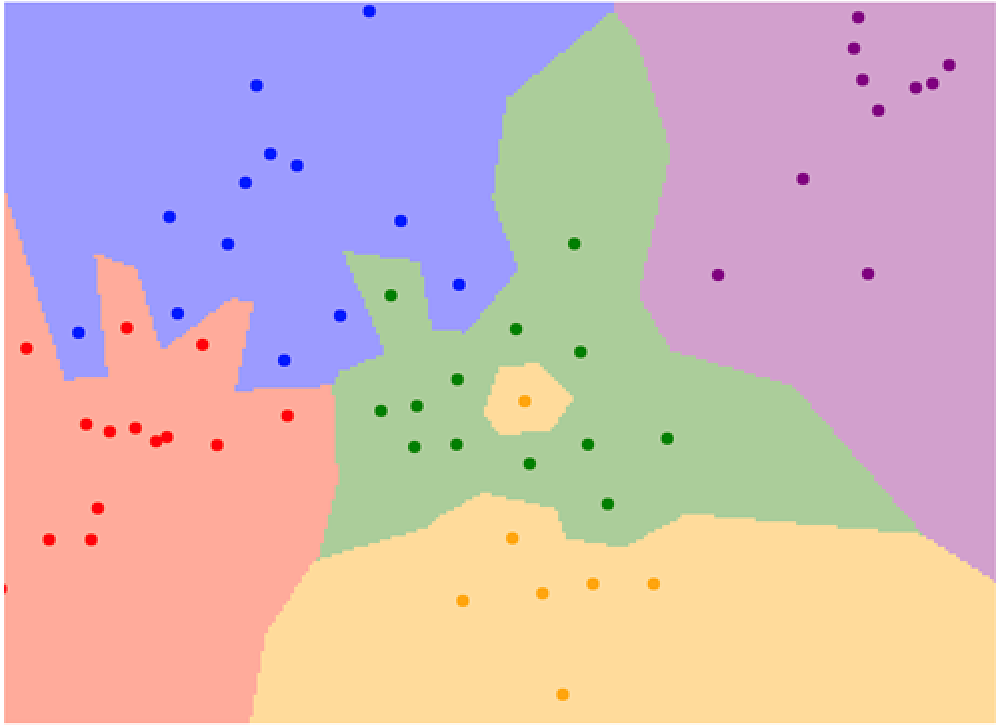
Nearest Neighbor classifier

Many methods exist for fast / approximate nearest neighbor (beyond the scope of 231N!)

A good implementation:
https://github.com/facebookresearch/faiss

Johnson et al, "Billion-scale similarity search with GPUs", arXiv 2017

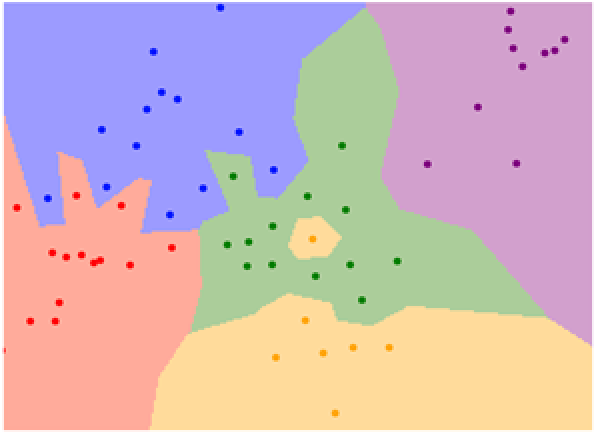
What does this look like?



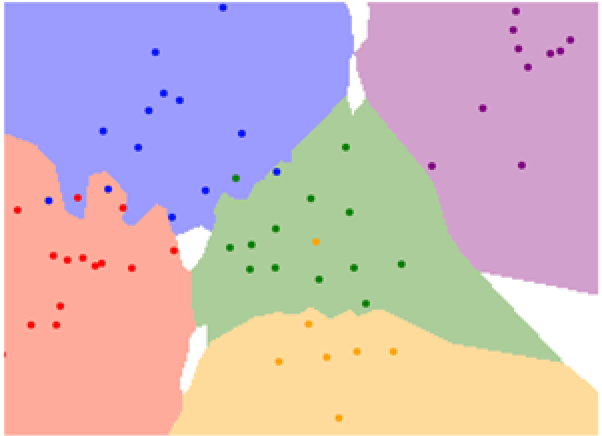
1-nearest neighbor

K-Nearest Neighbors

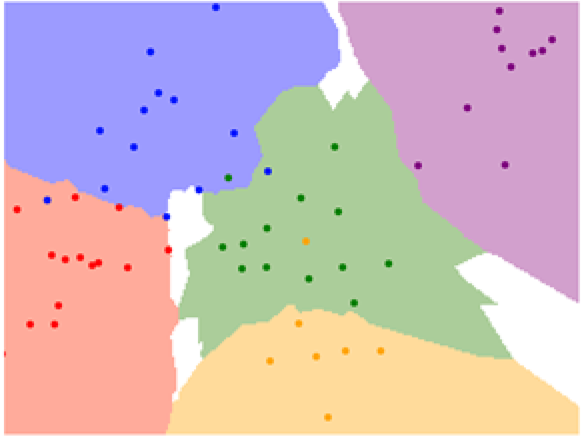
Instead of copying label from nearest neighbor, take majority vote from K closest points



K = 1



K = 3

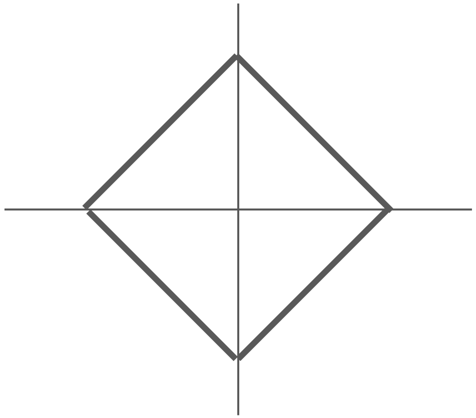


K = 5

K-Nearest Neighbors: Distance Metric

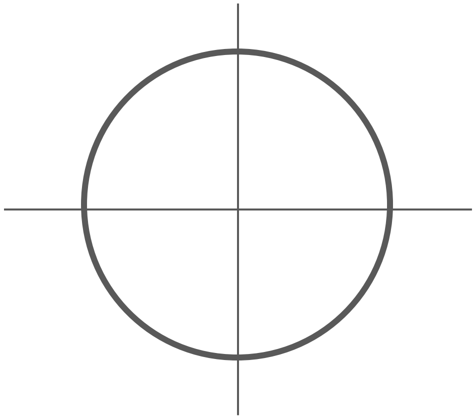
L1 (Manhattan) distance

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



L2 (Euclidean) distance

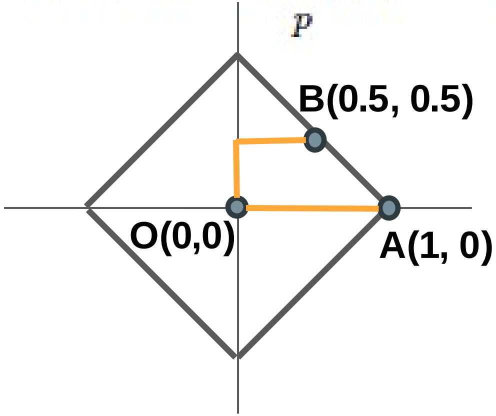
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



K-Nearest Neighbors: Distance Metric - Example

L1 Distance: Measures distance by moving along grid lines (like walking in a city with square blocks).

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



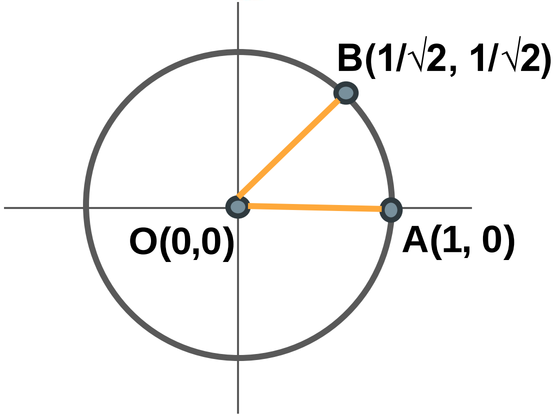
$d_1(O,A) = |0-1| + |0-0| = 1$

$d_1(O,B) = |0-0.5| + |0-0.5| = 0.5 + 0.5 = 1$

$d_1(O,A) = d_1(O,B) = 1$

L2 Distance: Measures the straight-line distance (as the crow flies).

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



$d_2(O,A) = \text{sqrt}((0-1)^2 + (0-0)^2) = \text{sqrt}(1^2) = 1$

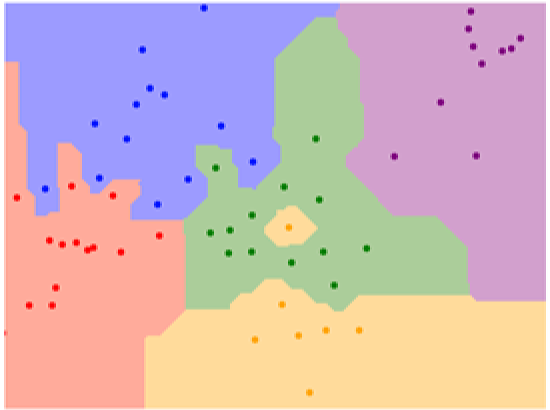
$d_2(O,B) = \text{sqrt}((0-1/\sqrt{2})^2 + (0-1/\sqrt{2})^2) = \text{sqrt}(1/2+1/2) = \text{sqrt}(1) = 1$

$d_2(O,A) = d_2(O,A) = 1$

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

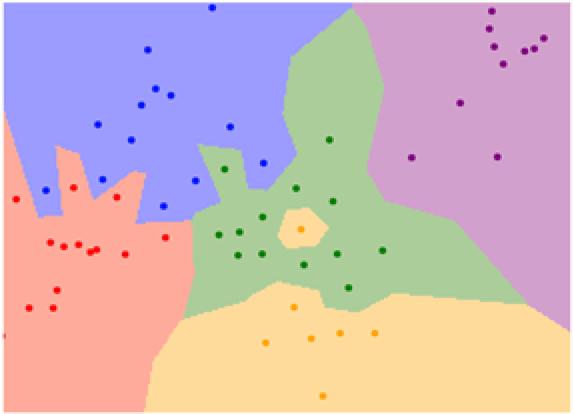
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

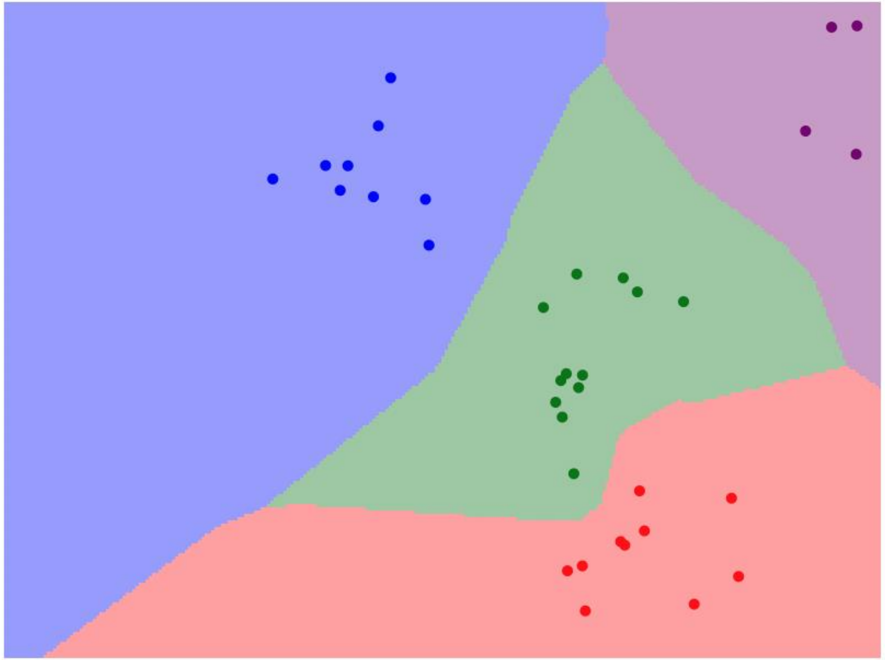
L2 (Euclidean) distance

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



K = 1

K-Nearest Neighbors: try it yourself!



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

Hyperparameters

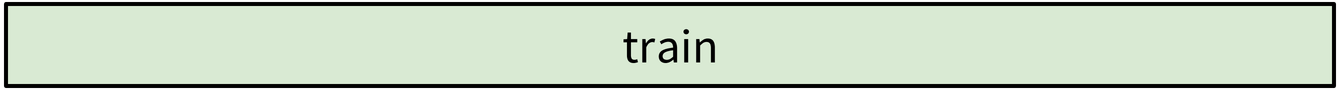
What is the best value of k to use?
What is the best distance to use?

These are hyperparameters: choices about the algorithms themselves.

Very problem/dataset-dependent.
Must try them all out and see what works best.

Setting Hyperparameters

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



Setting Hyperparameters

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

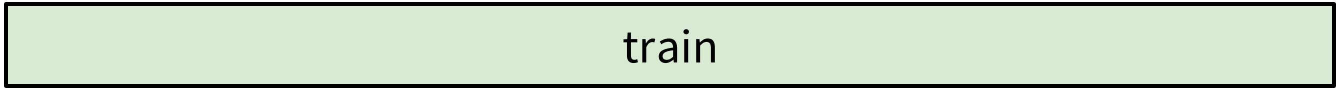
BAD: K = 1 always works perfectly on training data

train

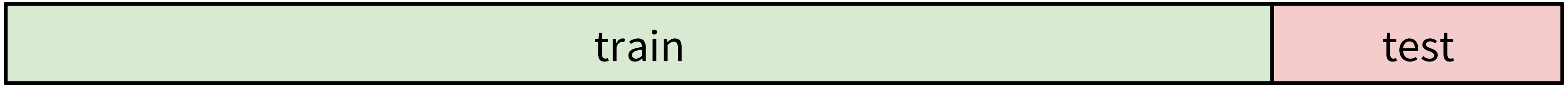
Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data



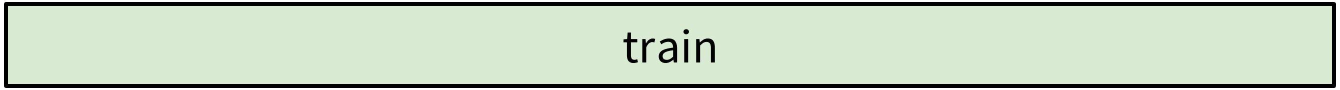
Idea #2: choose hyperparameters that work best on test data



Setting Hyperparameters

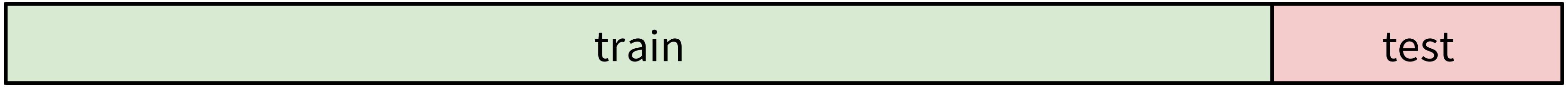
Idea #1: Choose hyperparameters that work best on the training data

BAD: K = 1 always works perfectly on training data



Idea #2: choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data



Never do this!

Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

train

Idea #2: 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; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Setting Hyperparameters

train

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

test

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

Example Dataset: CIFAR10

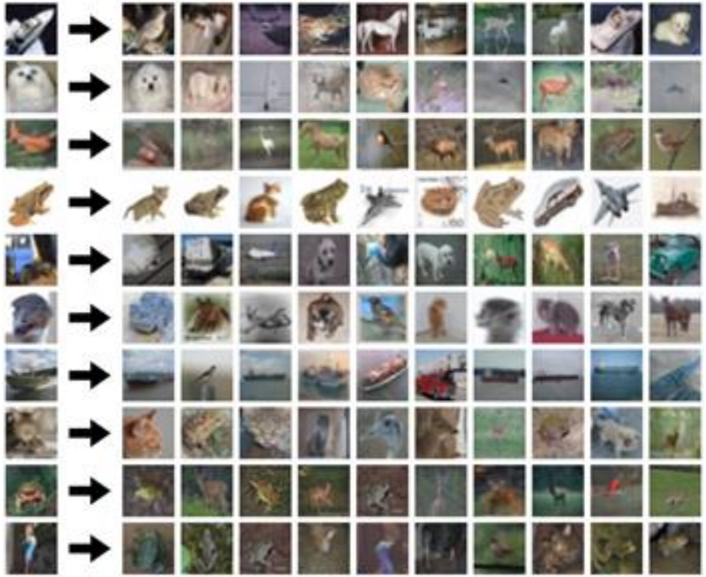
10 classes
50,000 training images
10,000 testing images

Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Example Dataset: CIFAR10

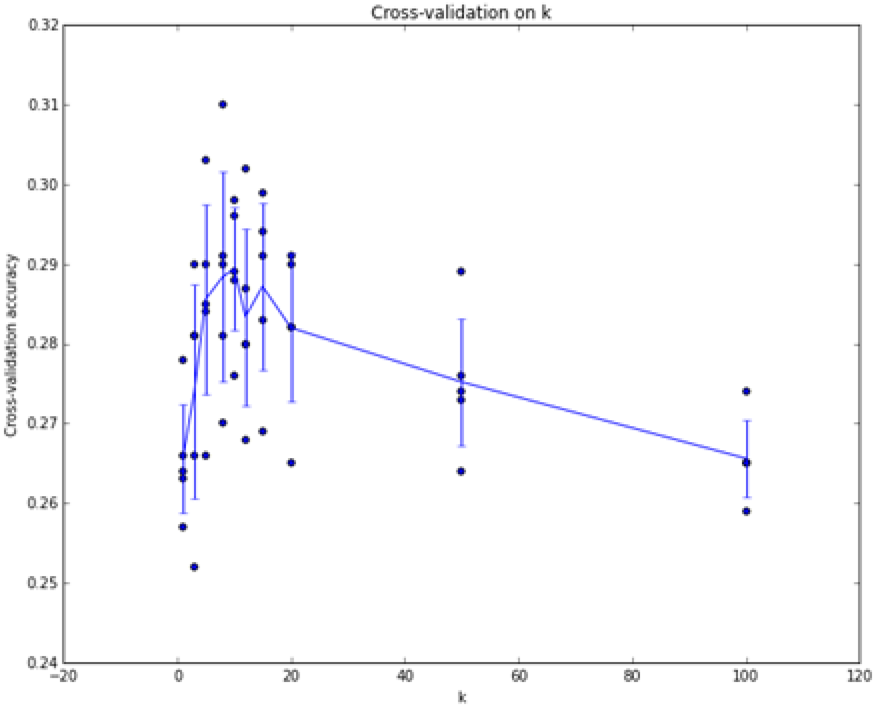
10 classes
50,000 training images
10,000 testing images

Test images and nearest neighbors



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Setting Hyperparameters



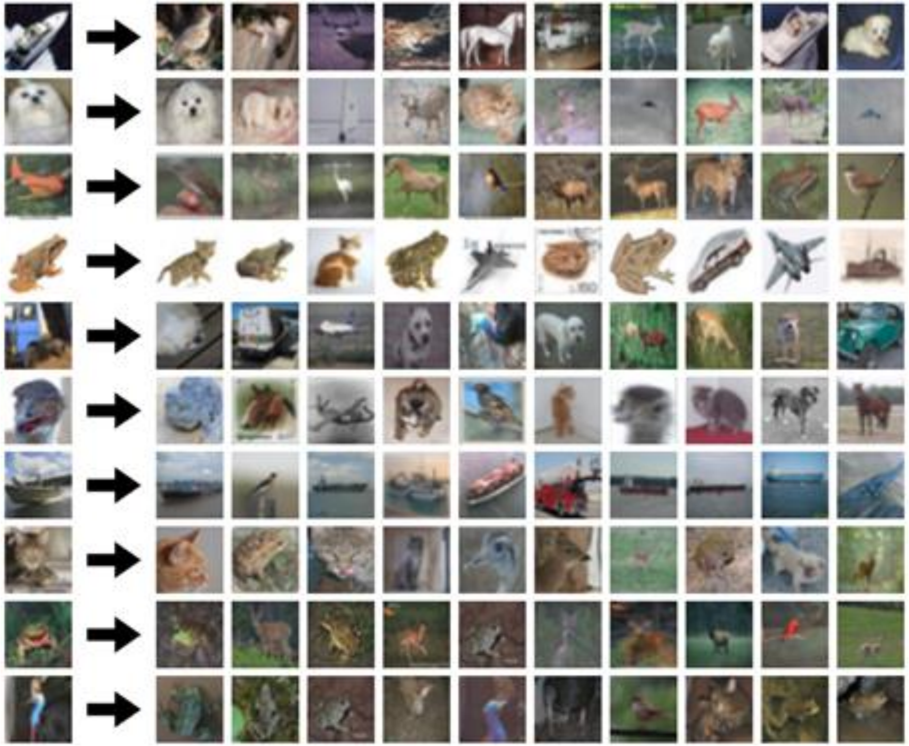
Example of
5-fold cross-validation
for the value of k.

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

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

What does this look like?



What does this look like?

k-Nearest Neighbor with pixel distance never used.

- Distance metrics on pixels are not informative

Original

Occluded

Shifted (1 pixel)

Tinted

(All three images on the right have the same pixel distances to the one on the left)

K-Nearest Neighbors: Summary

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

The K-Nearest Neighbors classifier predicts labels based on the **K nearest training examples**

Distance metric and K are hyperparameters

Choose hyperparameters using the validation set

Only run on the test set once at the very end!

Linear Classifier

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(x,W) → 10 numbers giving class scores

↑

W
parameters
or weights

Parametric Approach: Linear Classifier

3072x1

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

10x1 10x3072

Image

Array of 32x32x3 numbers
(3072 numbers total)

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

↑

W
parameters
or weights

Parametric Approach: Linear Classifier

3072x1

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

10x1 10x3072 10x1

Image

Array of 32x32x3 numbers
(3072 numbers total)

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

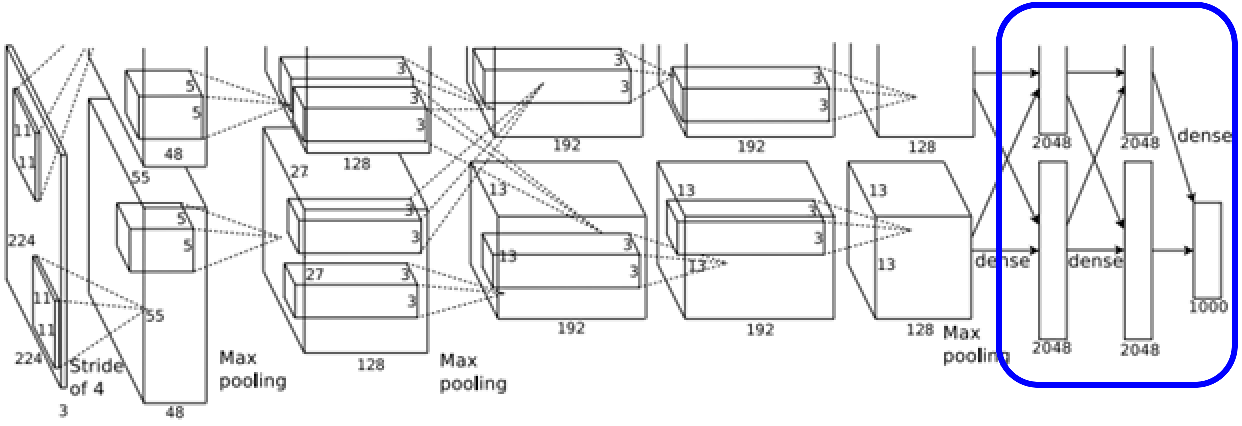
↑

W
parameters
or weights

Neural Network

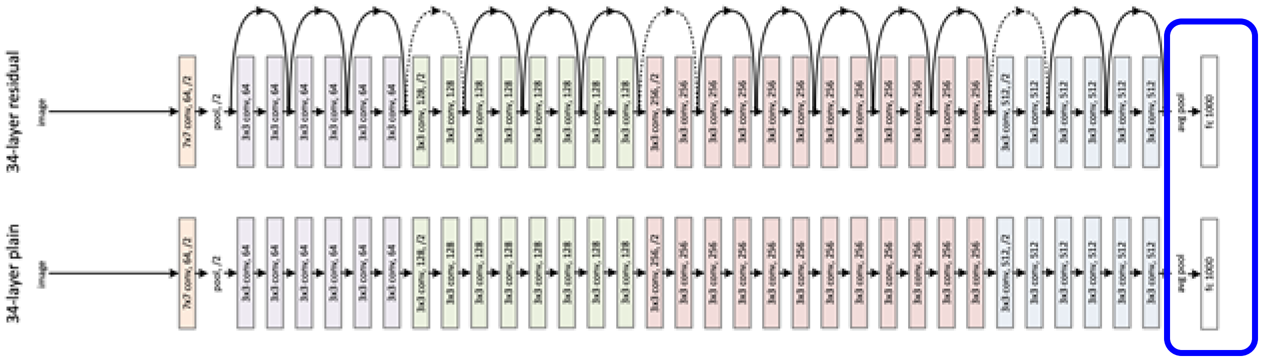
Linear classifiers

This image is CC0 1.0 public domain



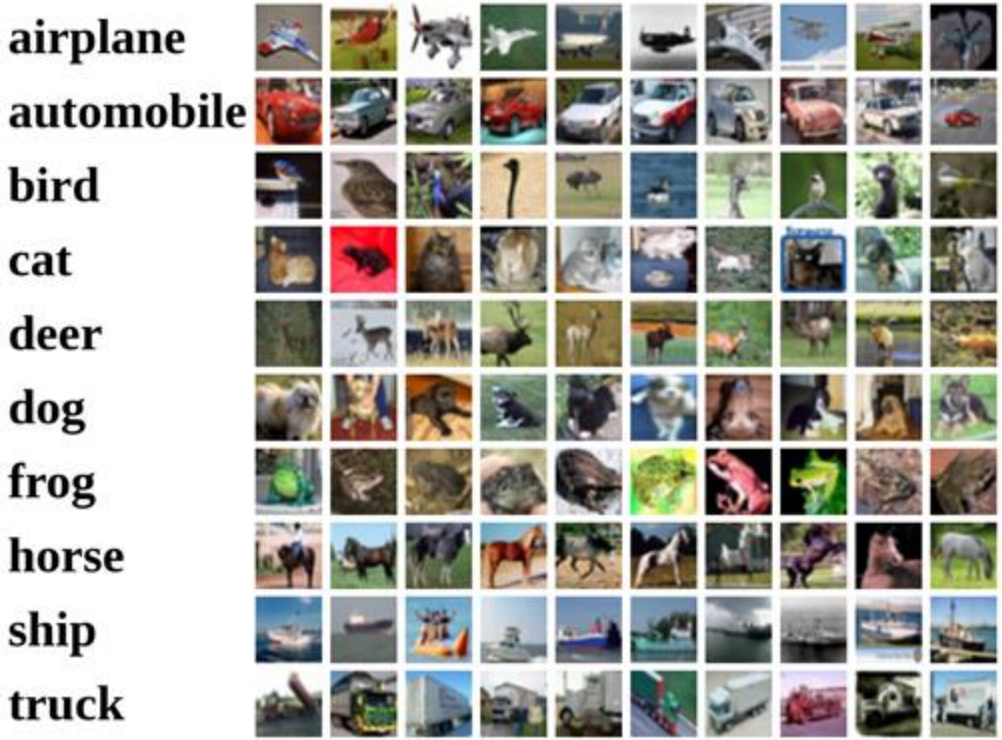
[Krizhevsky et al. 2012]

Linear layers



[He et al. 2015]

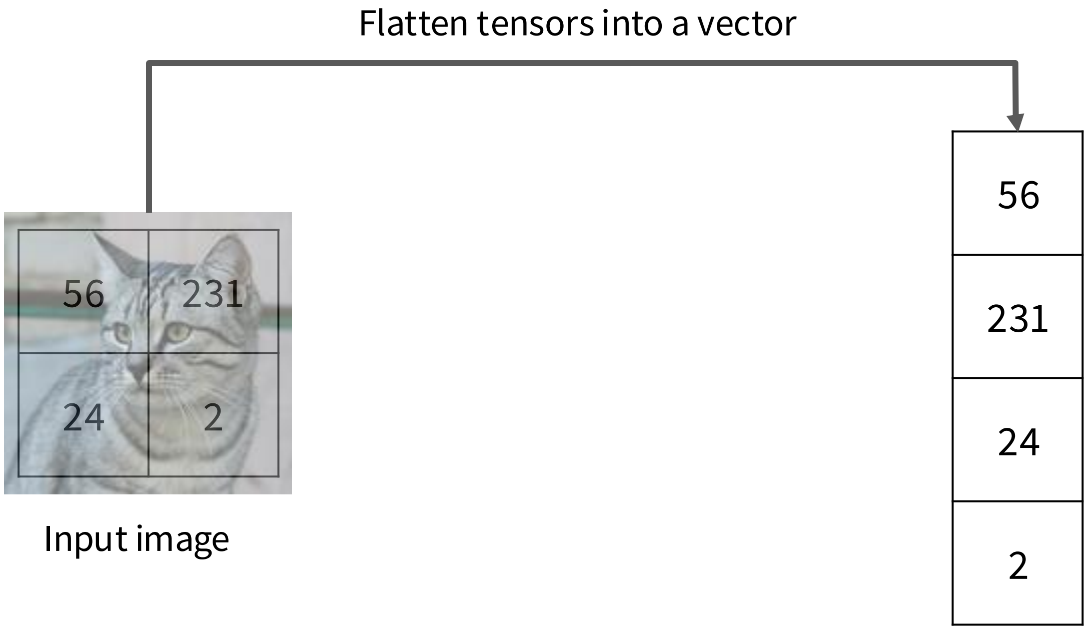
Recall CIFAR10



50,000 training images
each image is 32x32x3

10,000 test images.

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

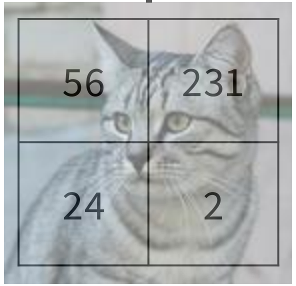


Input image

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Algebraic Viewpoint

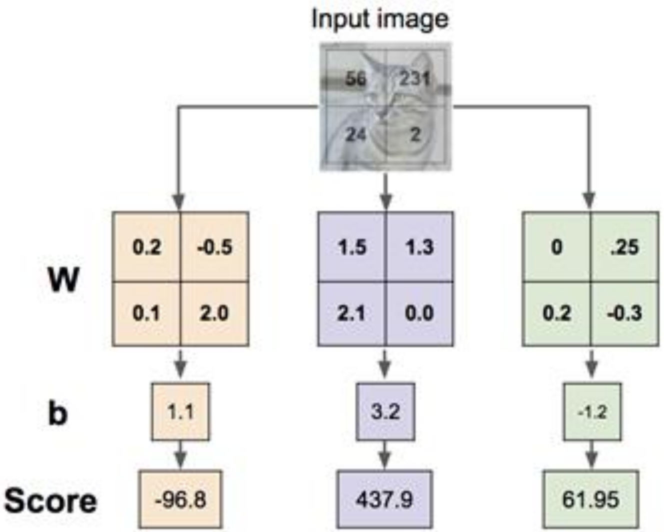
Flatten tensors into a vector



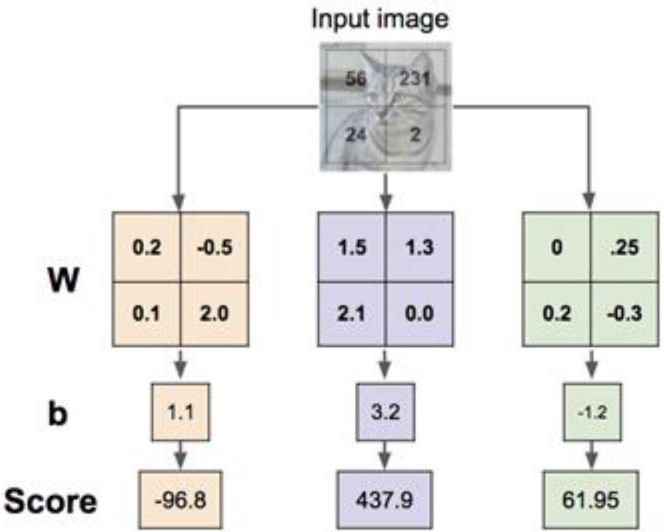
Input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{matrix} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{matrix}$$

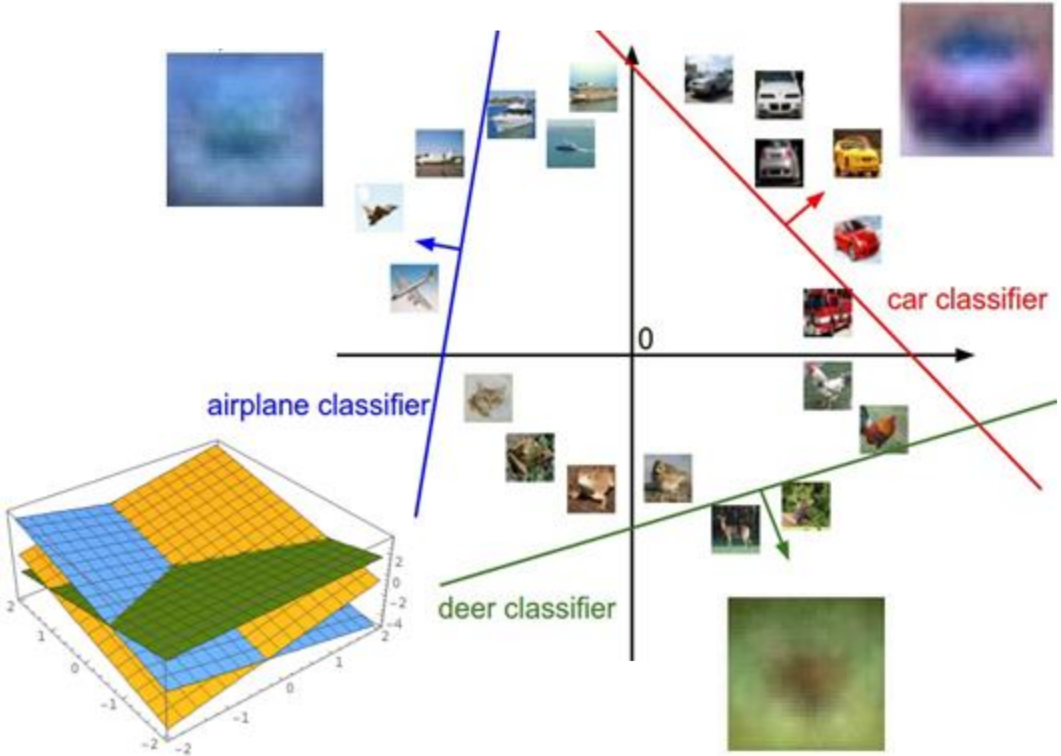
Interpreting a Linear Classifier



Interpreting a Linear Classifier: Visual Viewpoint



Interpreting a Linear Classifier: Geometric Viewpoint



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

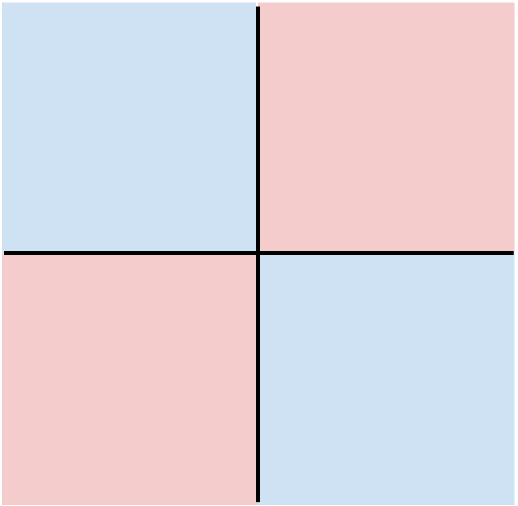
Array of 32x32x3 numbers
(3072 numbers total)

Plot created using Wolfram Cloud

Hard cases for a linear classifier

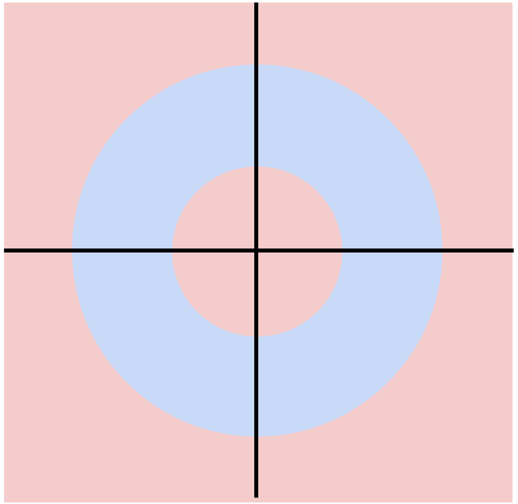
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



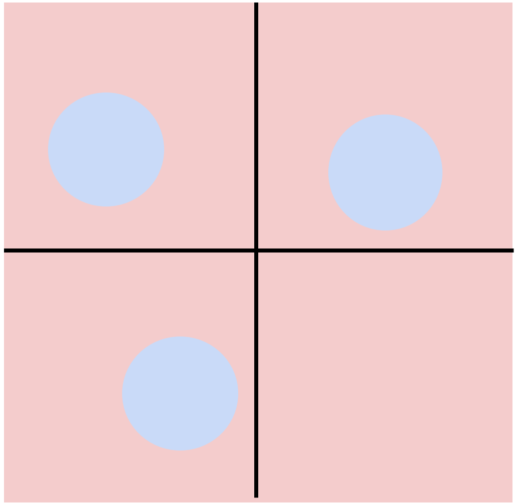
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Linear Classifier – 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

1. Define a loss function that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. (optimization)

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

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

A loss function tells how good our current classifier is

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

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 is (integer) label

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 is (integer) label

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

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

Softmax classifier

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

cat	3.2
car	5.1
frog	-1.7

Softmax Classifier (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

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

cat	3.2	exp →	24.5
car	5.1		164.0
frog	-1.7		0.18

unnormalized probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

		exp		normalize	
cat	3.2		24.5		0.13
car	5.1	→	164.0	→	0.87
frog	-1.7		0.18		0.00

unnormalized probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

Softmax Classifier (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		unnormalized probabilities		probabilities	
			Probabilities must be >= 0		Probabilities must sum to 1	
cat	3.2	exp →	24.5	normalize →	0.13	→ $L_i = -\log(0.13) = 2.04$
car	5.1		164.0		0.87	
frog	-1.7		0.18		0.00	

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

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

→ $L_i = -\log(0.13)$
$= 2.04$

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data
(See CS 229 for details)

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

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

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

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

Kullback–Leibler divergence

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

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

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

probabilities → compare ← Correct probs

Cross Entropy

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

Softmax Classifier (Multinomial Logistic Regression)

cat	3.2
car	5.1
frog	-1.7

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Maximize probability of correct class

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

Putting it all together:

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

Softmax Classifier (Multinomial Logistic Regression)

cat	3.2
car	5.1
frog	-1.7

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Maximize probability of correct class

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

Putting it all together:

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

Q1: What is the min/max possible softmax loss L_i?

Q2: At initialization all s_j will be approximately equal; what is the softmax loss L_i, assuming C classes?

Softmax Classifier (Multinomial Logistic Regression)

cat	3.2
car	5.1
frog	-1.7

Want to interpret raw classifier scores as probabilities

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

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

Softmax Function

Maximize probability of correct class

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

Putting it all together:

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

Q2: At initialization all s will be approximately equal; what is the loss?

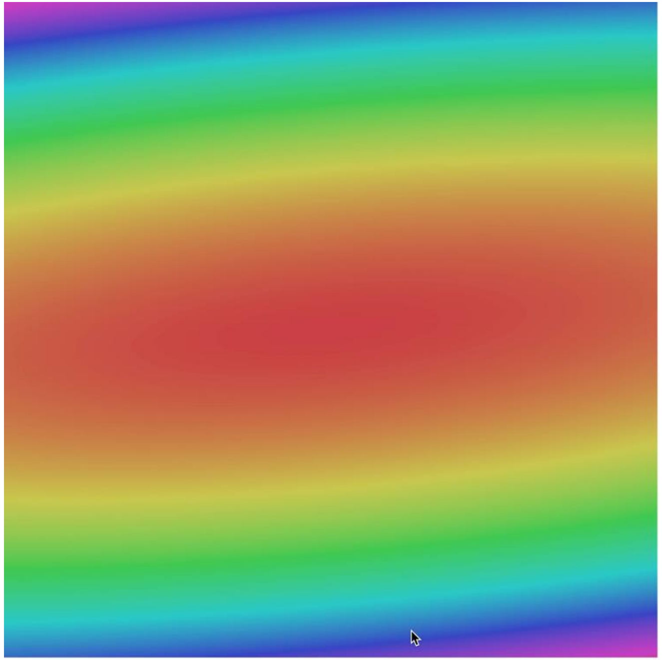
A: -log(1/C) = log(C),

If C = 10, then L_i = log(10) ≈ 2.3

Coming up:

- Regularization
- Optimization

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



Reading Assignment – 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

Multiclass SVM loss:

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:

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

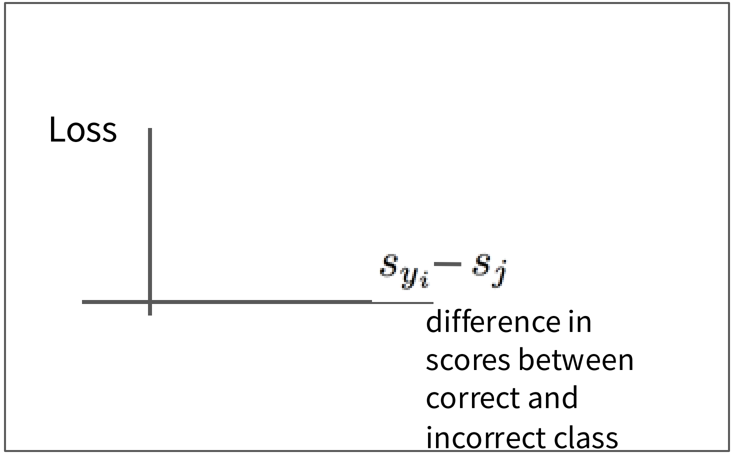
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

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:



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

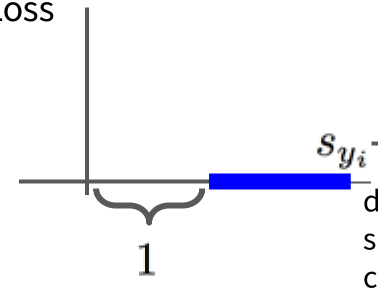
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

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:



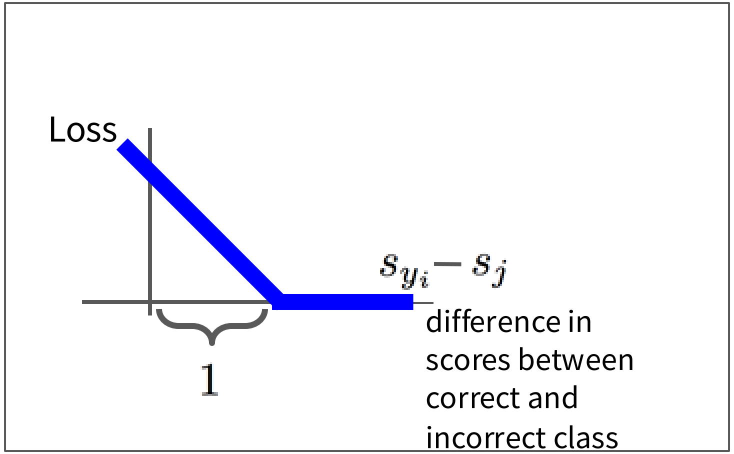
$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

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:



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

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

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

Multiclass SVM loss:

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:

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

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		

Multiclass SVM loss:

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, 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

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	

Multiclass SVM loss:

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

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

Multiclass SVM loss:

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, 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

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

Multiclass SVM loss:

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:

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

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

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

cat	1.3
car	4.9
frog	2.0
Losses:	0

Multiclass SVM loss:

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

Q1: What happens to loss if car scores decrease by 0.5 for this training example?

Q2: what is the min/max possible SVM loss L_i?

Q3: At initialization W is small so all $s \approx 0$. What is the loss L_i, assuming N examples and C classes?

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

Multiclass SVM loss:

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:

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

Q4: What if the sum was over all classes? (including j = y_i)

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

Multiclass SVM loss:

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:

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

Q5: What if we used mean instead of sum?

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

Multiclass SVM loss:

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:

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

Q6: What if we used

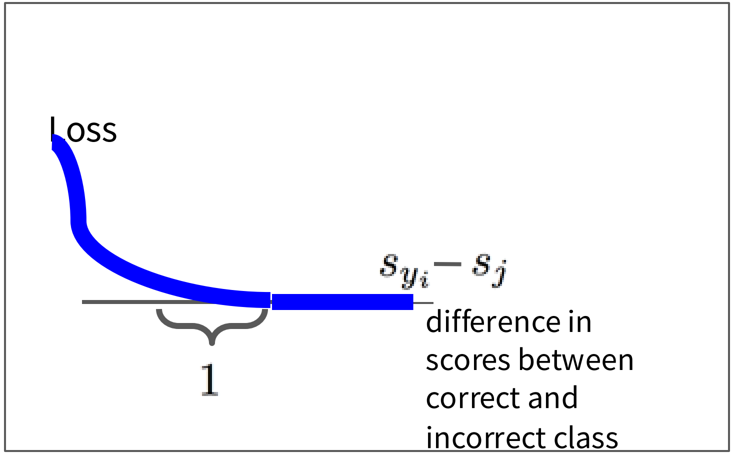
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

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

Multiclass SVM loss:



Q6: What if we used

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

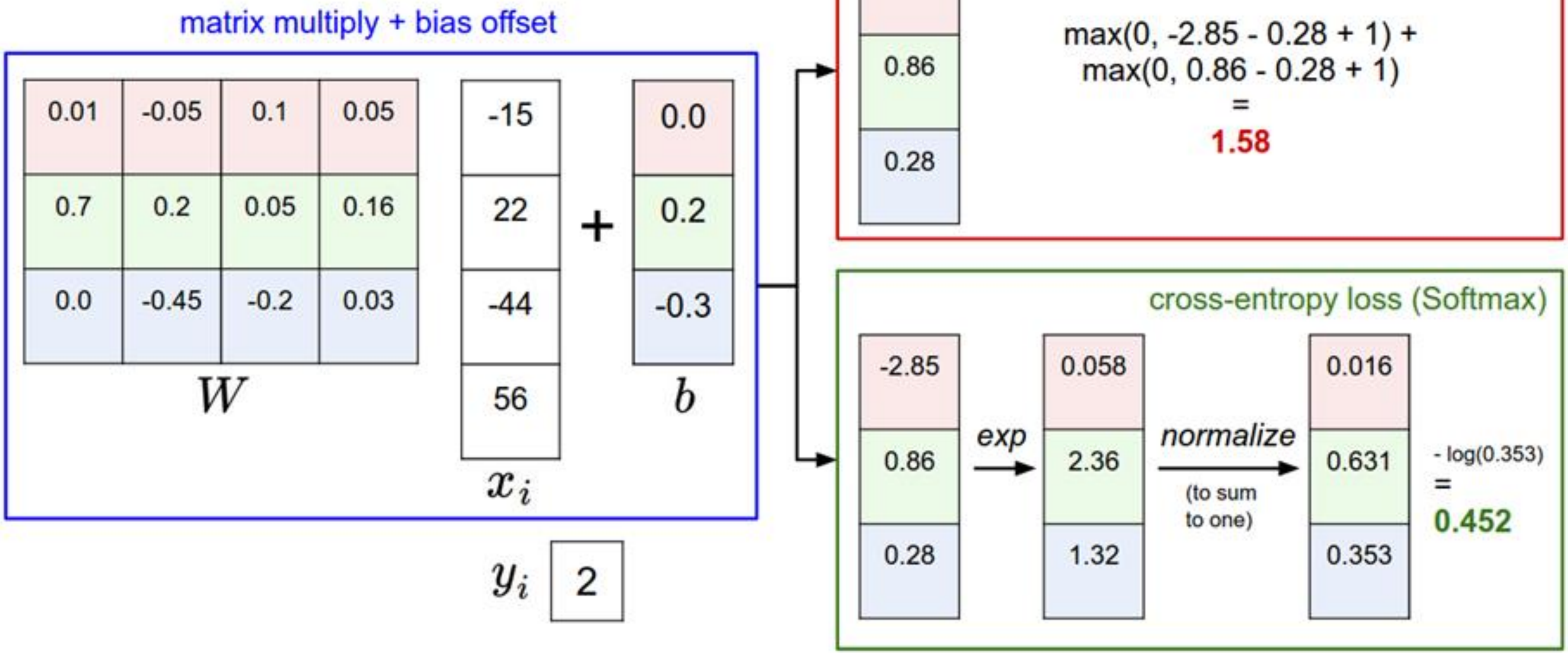
Multiclass SVM Loss: Example code

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

```
def L_i_vectorized(x, y, W):
  scores = W.dot(x)
  margins = np.maximum(0, scores - scores[y] + 1)
  margins[y] = 0
  loss_i = np.sum(margins)
  return loss_i
```

First calculate scores
Then calculate the margins $s_j - s_{yi} + 1$
only sum j is not y_i, so when j = y_i, set to zero.
sum across all j

Softmax vs. SVM



Softmax vs. SVM

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

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

Softmax vs. SVM

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

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

assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and $y_i = 0$

Q: What is the softmax loss and the SVM loss?

Softmax vs. SVM

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

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

assume scores:
[20, -2, 3]
[20, 9, 9]
[20, -100, -100]
and $y_i = 0$

Q: What is the softmax loss and the SVM loss if I double the correct class score from 10 -> 20?