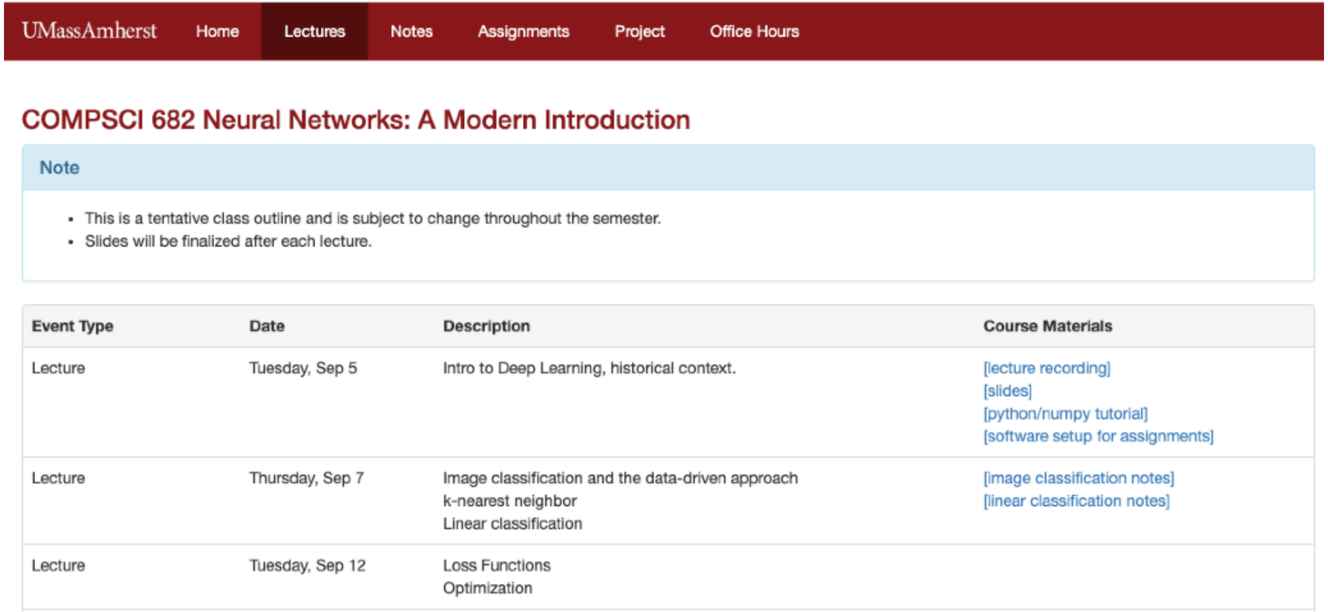
Lecture 2:
Nearest Neighbor and Linear Classification

Course web page

- https://cvl-umass.github.io/compsci682-fall-2023



COMPSCI 682 Neural Networks: A Modern Introduction

Note

- This is a tentative class outline and is subject to change throughout the semester.
- Slides will be finalized after each lecture.

Event Type	**Date**	**Description**	**Course Materials**
Lecture	Tuesday, Sep 5	Intro to Deep Learning, historical context.	[lecture recording] [slides] [python/numpy tutorial] [software setup for assignments]
Lecture	Thursday, Sep 7	Image classification and the data-driven approach k-nearest neighbor Linear classification	[image classification notes] [linear classification notes]
Lecture	Tuesday, Sep 12	Loss Functions Optimization	

Optional Discussion Sections

- Friday: 9:00-10:00 am, CS142
    - No discussion section this Friday
    - First one on Sept. 15 (Time & Location TBD)
- Will cover background topics such as:
    - Python techniques
        - slicing and broadcasting
        - Other parallelization techniques
    - Math techniques
        - Derivatives of vectors, matrices, etc.
        - Complex chain rule examples

Piazza

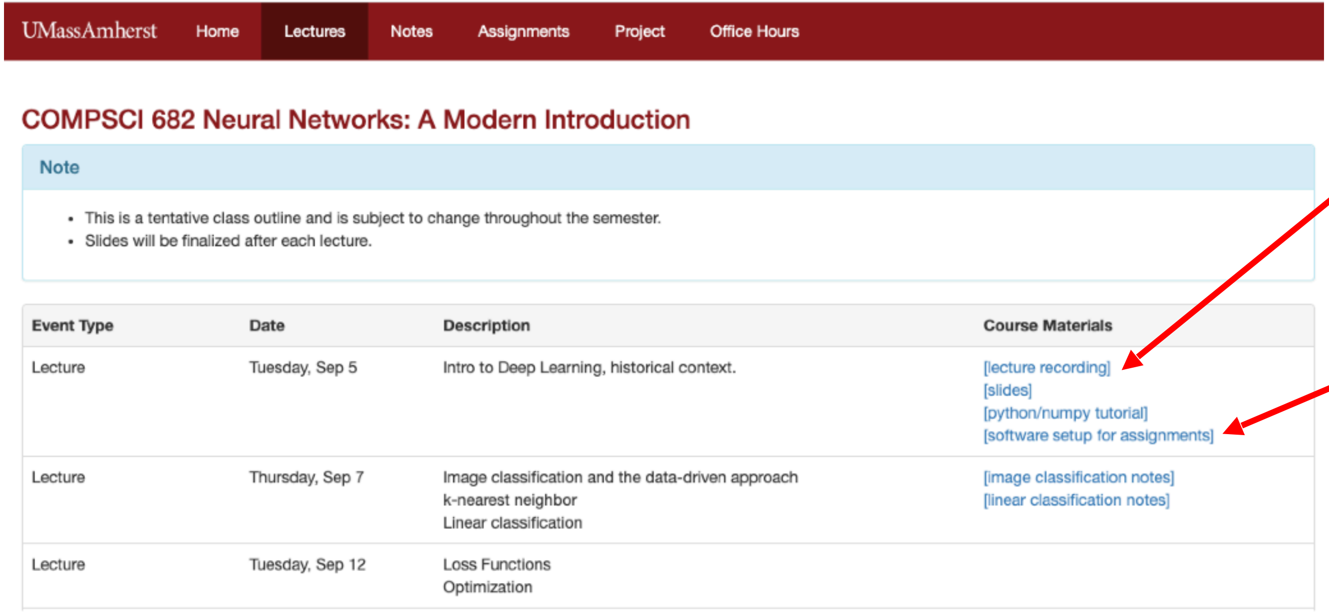
- Everyone needs to sign up for Piazza. This is how you get messages for the class.
- Please post most questions there, rather than sending email to me or the TAs.
- The TAs and I will answer questions posted to Piazza.

Python: Up and running

- **We’re using Python 3.6+, not 2.7.**
- If you have not installed Python and tried a simple program yet, please do so as soon as possible.
- Try loading up the first Python Notebook for Homework 1. Even if you don’t have time to do the assignment yet, at least make sure the Python Notebook is working properly. This will make sure you don’t incur a major delay later.
- PYTHON Tutorial! See “Notes” tab of course web page. (Show Notes page).

Readings and Lecture Recordings

On Lectures tab of course webpage



COMPSCI 682 Neural Networks: A Modern Introduction

Note

- This is a tentative class outline and is subject to change throughout the semester.
- Slides will be finalized after each lecture.

Event Type	Date	Description	Course Materials
Lecture	Tuesday, Sep 5	Intro to Deep Learning, historical context.	[lecture recording] [slides] [python/numpy tutorial] [software setup for assignments]
Lecture	Thursday, Sep 7	Image classification and the data-driven approach k-nearest neighbor Linear classification	[image classification notes] [linear classification notes]
Lecture	Tuesday, Sep 12	Loss Functions Optimization	

Echo360

Readings

Back to classification...

Data-driven approach:

1. Collect a dataset of images and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

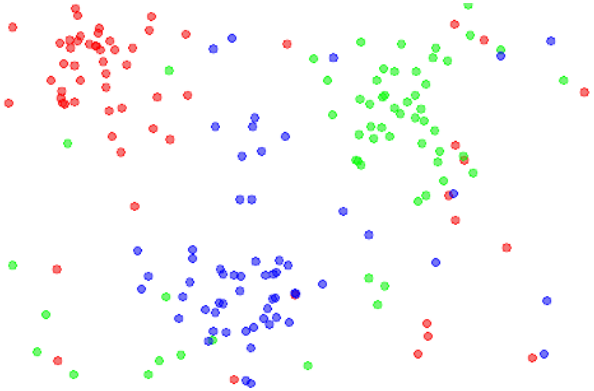
```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

Example training set

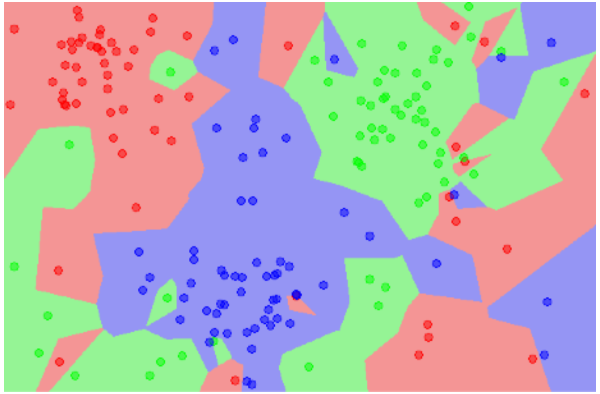
k-Nearest Neighbor

find the k nearest images, have them vote on the label

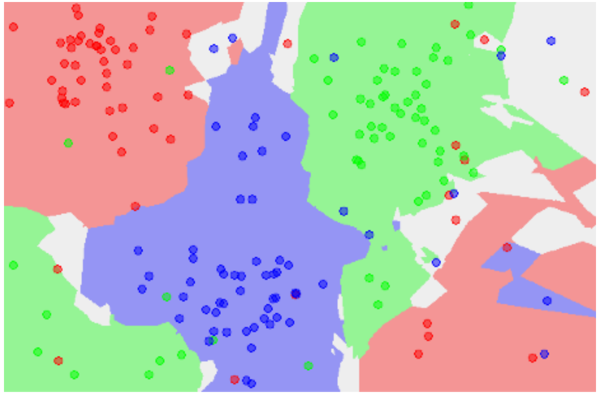
the data



NN classifier



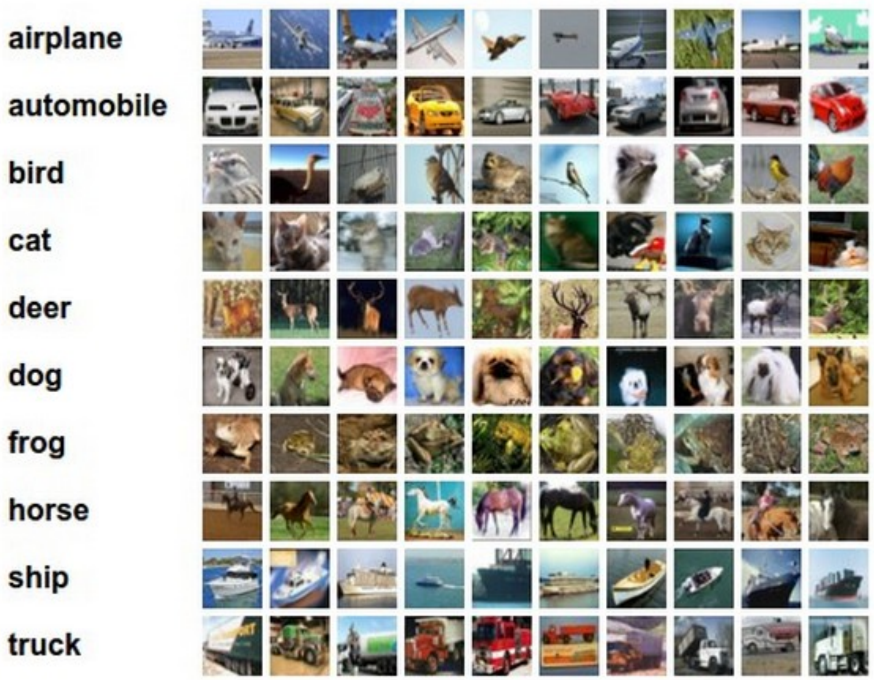
5-NN classifier

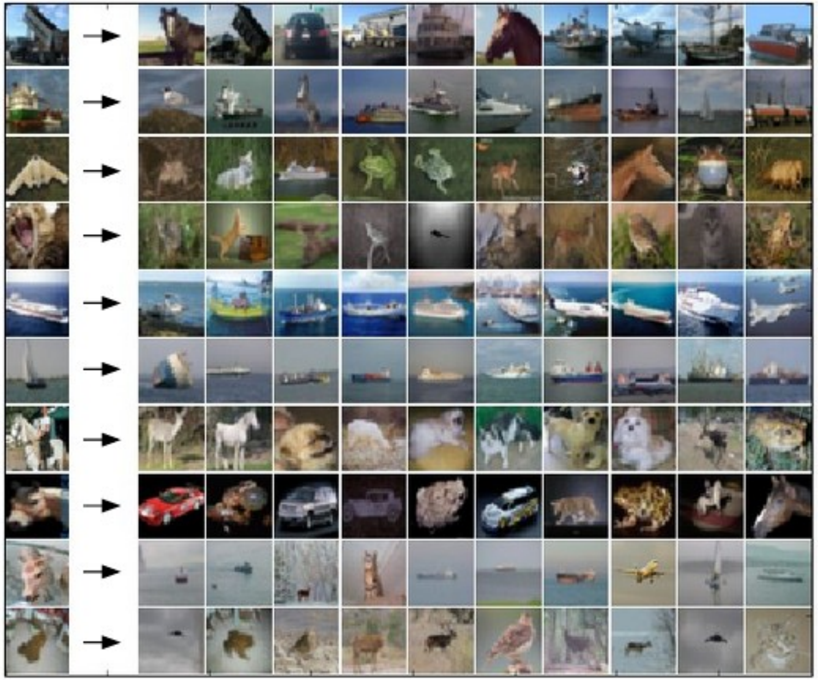


http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

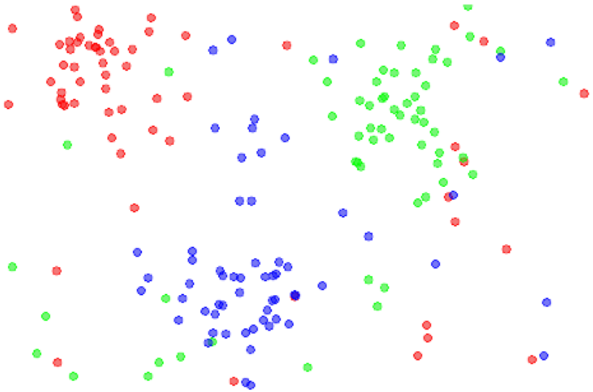
Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.

For every test image (first column), examples of nearest neighbors in rows

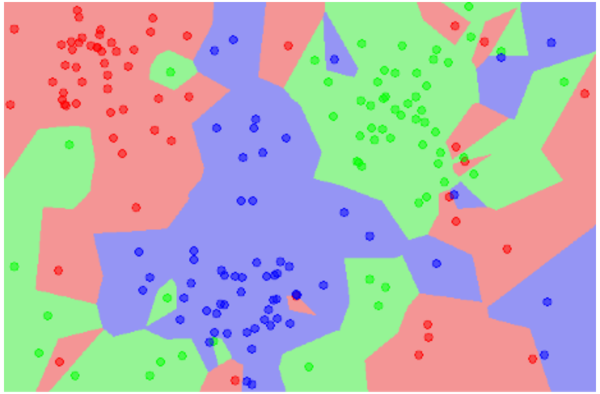




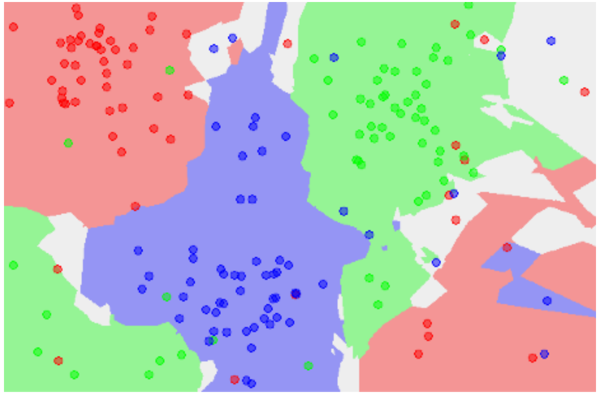
the data



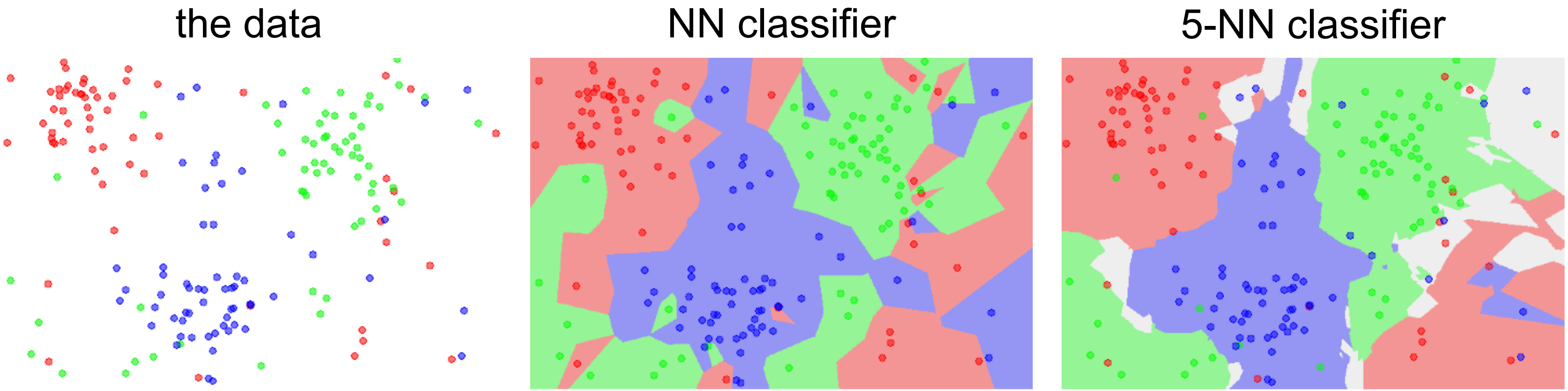
NN classifier



5-NN classifier



Q: what is the accuracy of the nearest neighbor classifier on the training data, when using the Euclidean distance?



the data

NN classifier

5-NN classifier

Q2: what is the accuracy of the **k-**nearest neighbor classifier on the training data?

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

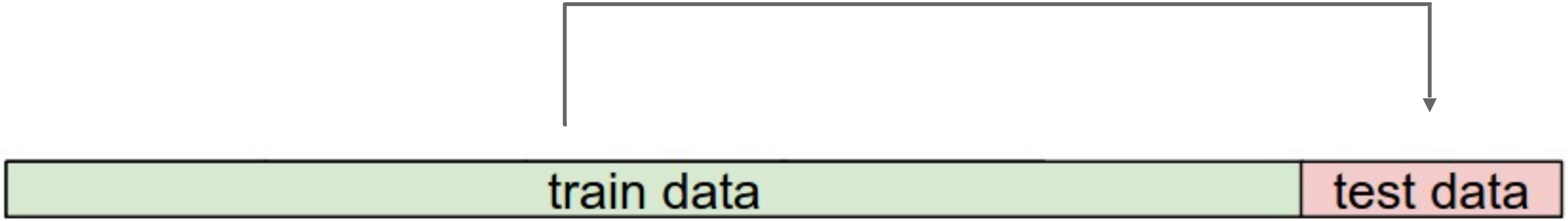
i.e. how do we set the **hyperparameters**?

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

i.e. how do we set the **hyperparameters**?

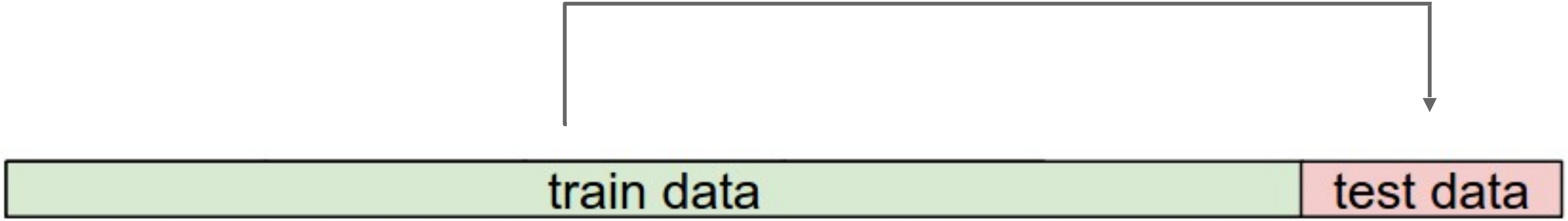
Very problem-dependent.
Must try them all out and see what works best.

Trying out what hyperparameters work best on test set.



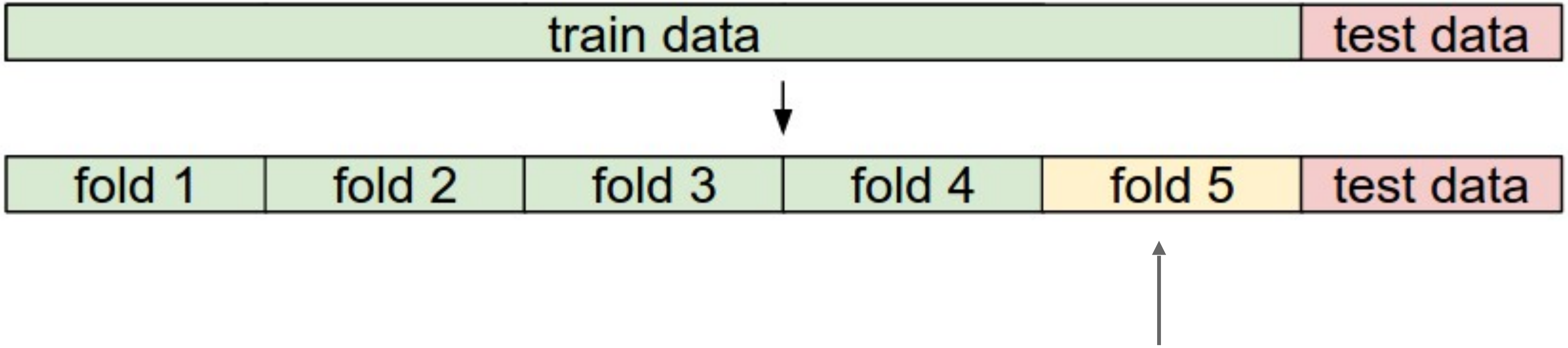
Trying out what hyperparameters work best on test set:

Very bad idea. The test set is a proxy for the generalization performance!
Use only **VERY SPARINGLY,** at the end.



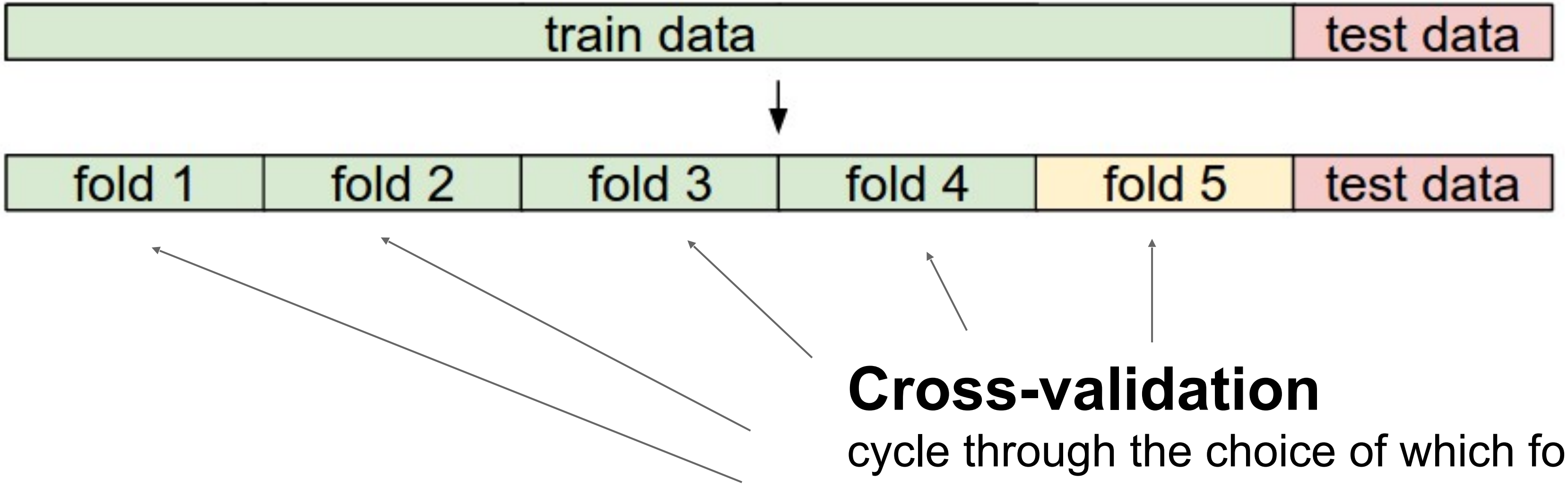
train data

test data



Validation data

use to tune hyperparameters

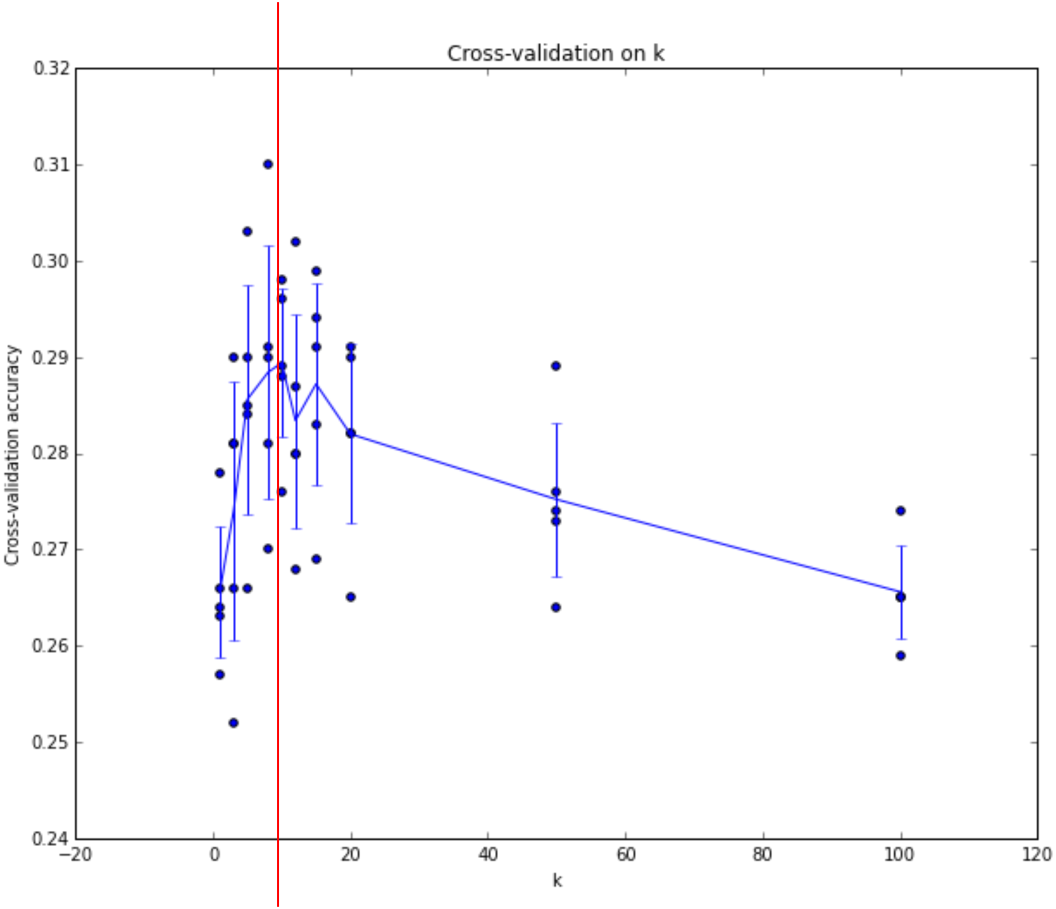


train data | test data

fold 1 | fold 2 | fold 3 | fold 4 | fold 5 | test data

Cross-validation

cycle through the choice of which fold is the validation fold, average results.



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)

k-Nearest Neighbor on *raw* images is **never used.**

- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive

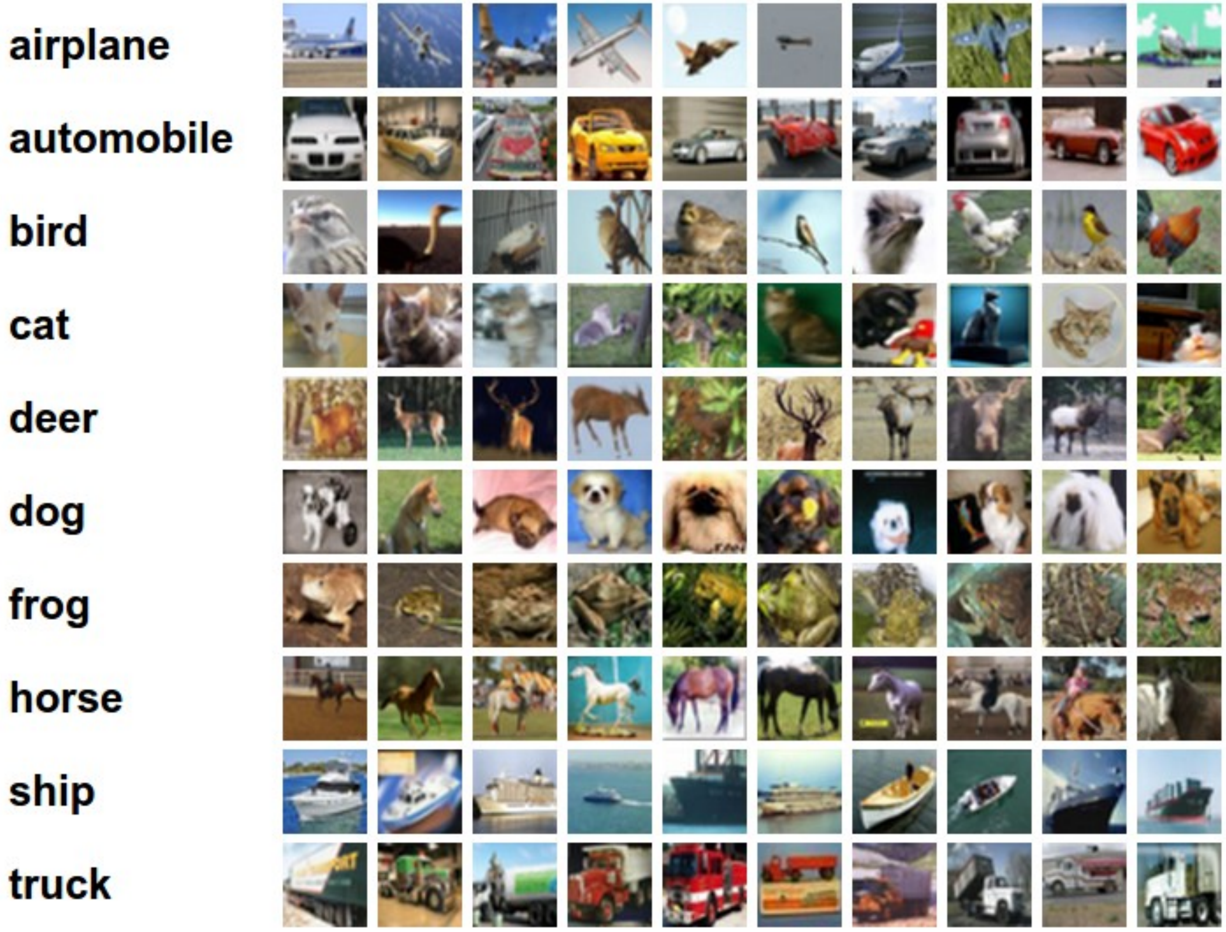
(all 3 images have same L2 distance to the one on the left)

Before moving on

- K-NN: the Rodney Dangerfield of classifiers

- Convergence of K-NN to the Bayes error rate.
- Universality of K-NN.

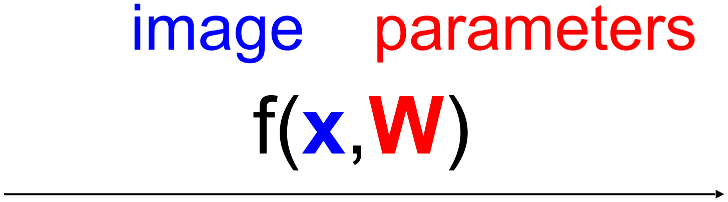
Linear Classification



Example dataset: **CIFAR-10**
10 labels
50,000 training images
each image is **32x32x3**
10,000 test images.

Parametric approach

[32x32x3]
array of numbers 0...1
(3072 numbers total)



10 numbers,
indicating class
scores

Parametric approach: **Linear classifier**

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

[32x32x3]
array of numbers 0...1

10 numbers,
indicating class
scores

Parametric approach: **Linear classifier**

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

3072x1

10x1

10x3072

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

parameters, or "weights"

Parametric approach: **Linear classifier**

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

3072x1

10x1

10x1

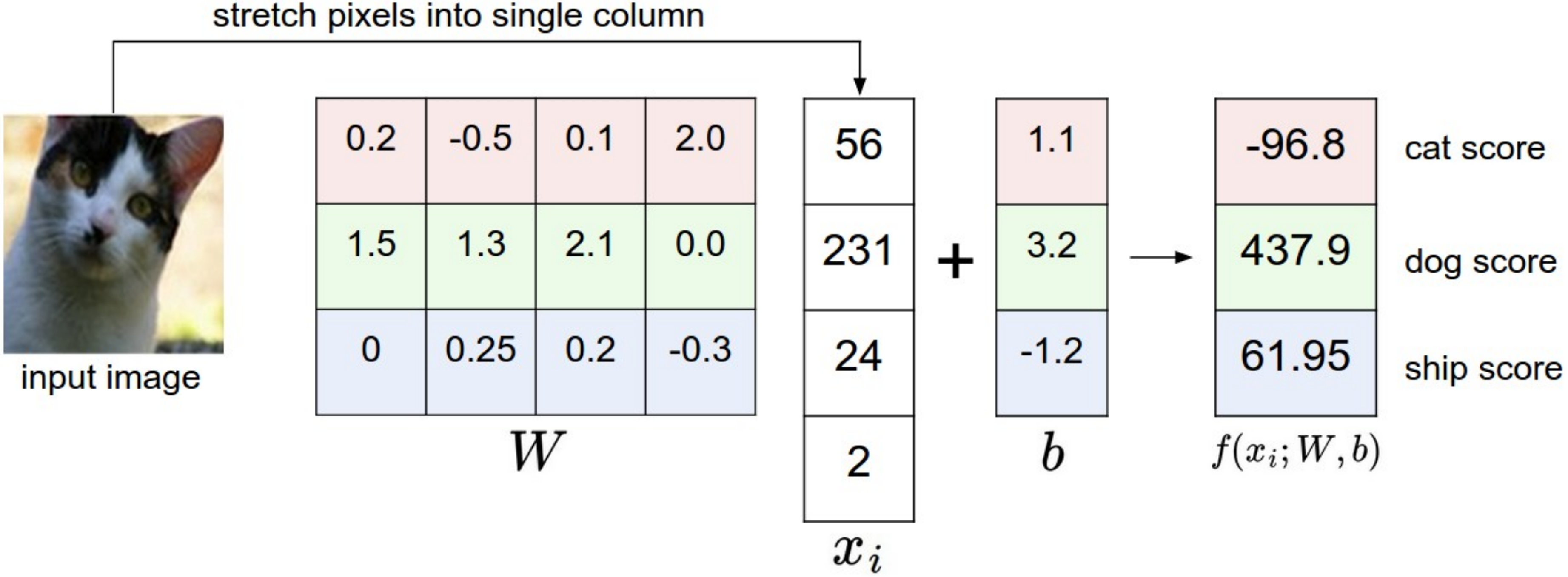
10x3072

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

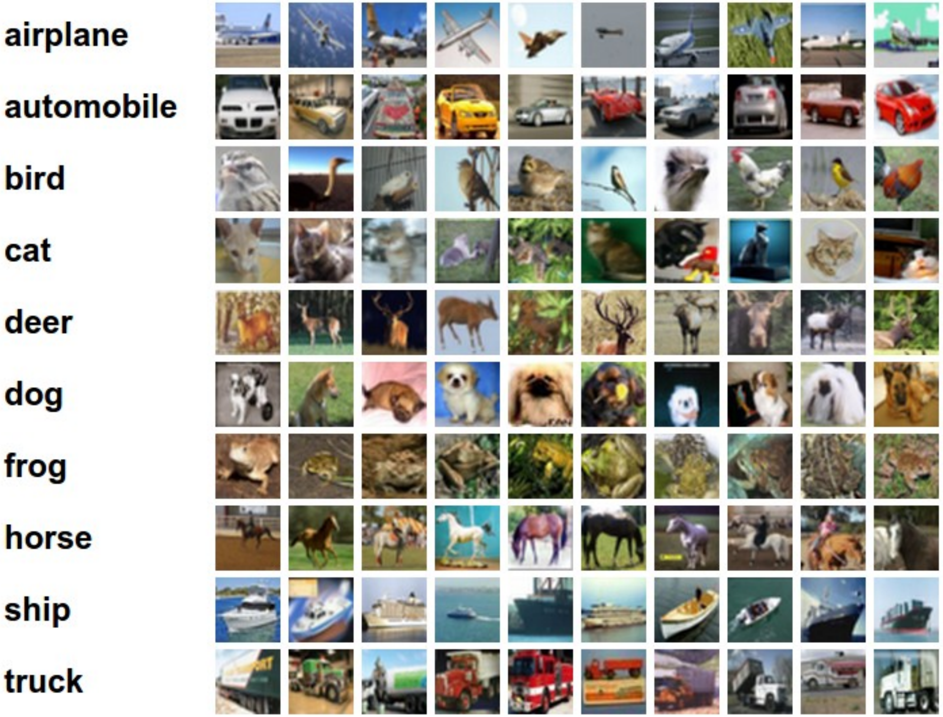
parameters, or “weights”

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



$$\underset{W}{\begin{bmatrix}0.2 & -0.5 & 0.1 & 2.0\\ 1.5 & 1.3 & 2.1 & 0.0\\ 0 & 0.25 & 0.2 & -0.3\end{bmatrix}}\underset{x_i}{\begin{bmatrix}56\\ 231\\ 24\\ 2\end{bmatrix}}+\underset{b}{\begin{bmatrix}1.1\\ 3.2\\ -1.2\end{bmatrix}}\rightarrow\underset{f(x_i;W,b)}{\begin{bmatrix}-96.8\\ 437.9\\ 61.95\end{bmatrix}}\begin{array}{l}\text{cat score}\\ \text{dog score}\\ \text{ship score}\end{array}$$

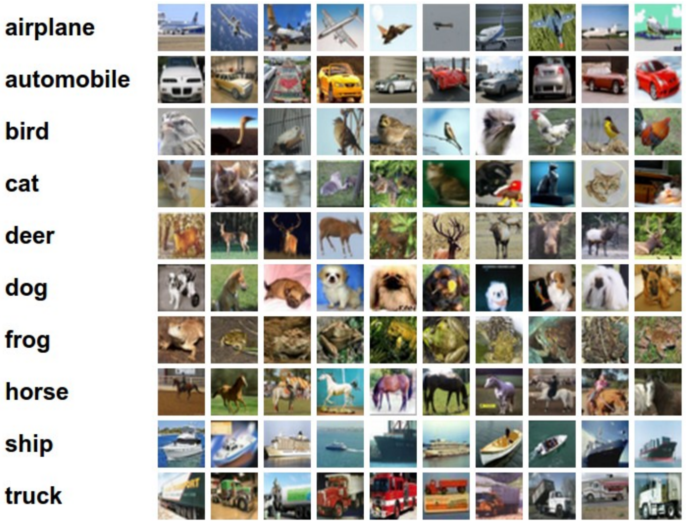
Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$

Q: what does the linear classifier do, in English?

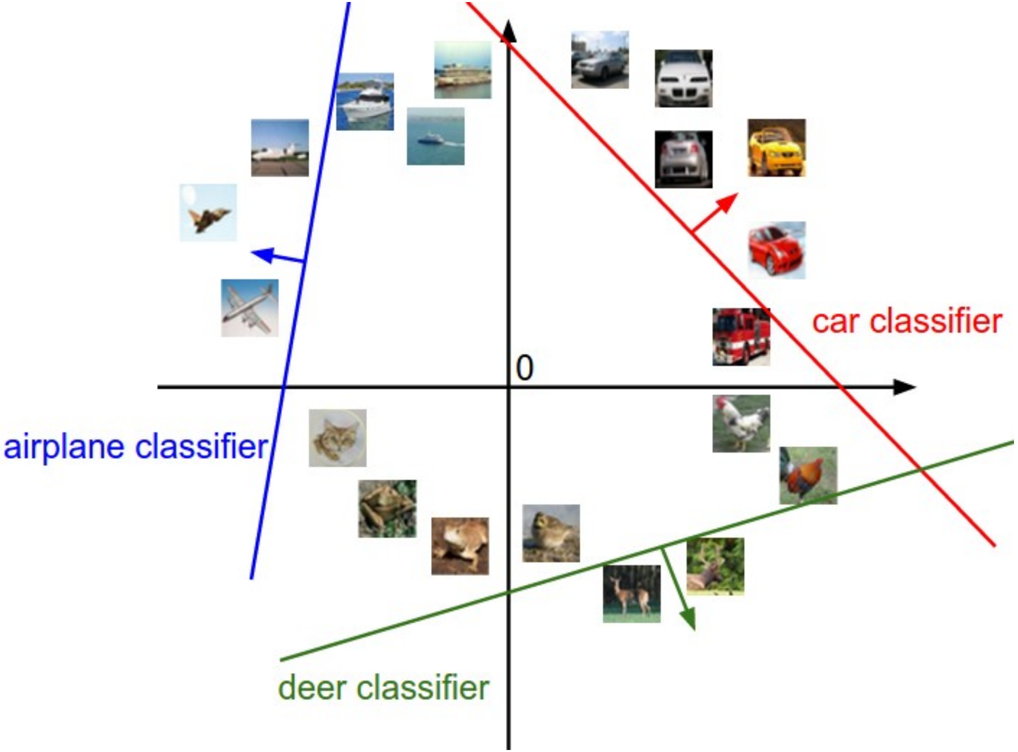
Interpreting a Linear Classifier (poll)



$$f(x_i, W, b) = Wx_i + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

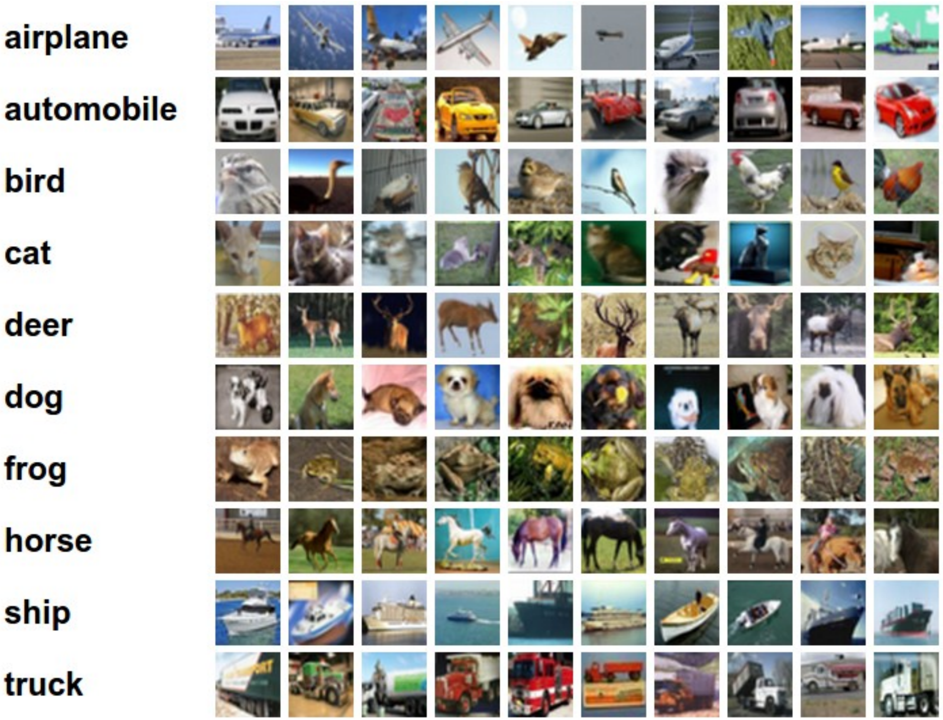
Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$

Q2: what would be a very hard set of classes for a linear classifier to distinguish?

So far: We defined a (linear) **score function**: $f(x_i, W, b) = Wx_i + b$

really ***affine***

Example class scores for 3 images, with a random 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

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

Coming up:

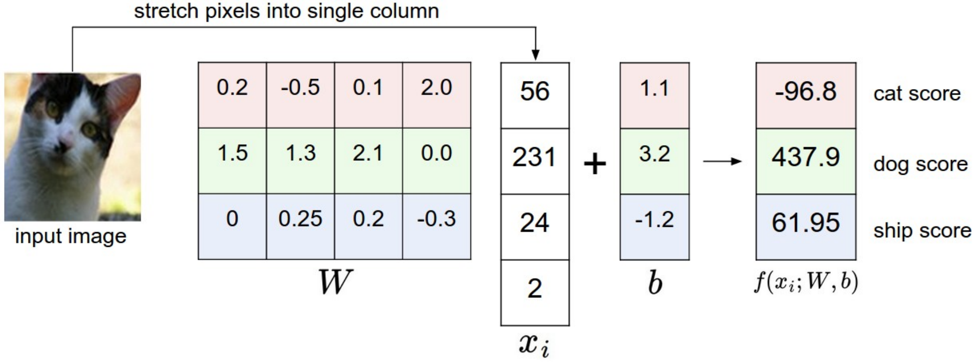
- Loss function (quantifying what it means to have a “good” W)
- Optimization (start with random W and find a W that minimizes the loss)
- Neural nets! (tweak the functional form of f)

Summary so far ... Linear classifier

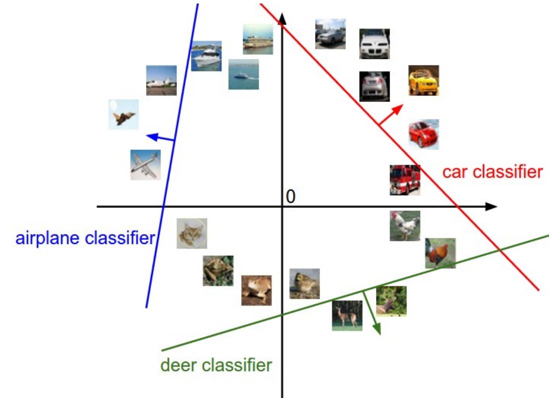
[32x32x3]
array of numbers 0...1
(3072 numbers total)

image parameters
f(**x**,**W**)

10 numbers, indicating class scores



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



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

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

1. 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 $f(x,W) = Wx$ are:

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

and the full training loss is the mean over all examples in the training data:

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

L = (2.9 + 0 + 12.9)/3
= **5.3**

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

Q: what if the sum was instead over all classes?
(including j = y_i)

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

Q2: what if we used a mean instead of a sum here?

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

Q3: 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 $f(x, W) = Wx$ are:

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

Q4: what is the min/max possible loss?

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

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: $s_i = f(x_i, W)$

the SVM loss has the form:

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

Q5: usually at initialization W are small numbers, so all s ~= 0. What is the loss?