

lec2

Lecture 2 notes on 02/11/2019 18:54

Last Updated: 26/12/2019 22:52

- Image Classification: A core task in CV
- An image is just a big grid of numbers between [0-255], and a computer only sees that matrix and it has to infer the content from it.
- **Semantic Gap** : The semantic idea of an object (cat in a image) and that of the pixel values that the computer sees
- Challenges:
 - Viewpoint variation: All pixels change when the camera moves!
 - Illumination: Different lighting conditions for the same kind of image content/subject.
 - Deformation: Different, varied poses, and various other transforms.
 - Occlusion: Only part of the object may be visible.
 - Background Clutter: Foreground object could actually look quite similar in appearance to the background.
 - Intraclass variation: Different visual appearances for the same object. Eg. Different ages, shape, breed of cats
- There is no obvious way to hard-code the algorithm for recognizing a cat, or other classes unlike in other problems like sorting, encryption etc. where we can write the steps to execute.
- But there have been attempts to write an algorithm using rules.
 - Image --> Find Edges --> Find Corners --> Infer the content based on relations of these elements of edges and corners.
 - Disadvantage: **Not generalizable**: The implementation is not flexible, and it is hardcoded for a single class.
 - Not scalable. Since we need to write different approaches for each class.
 - Solution: **Data-Driven Approach**

- Data-Driven Approach

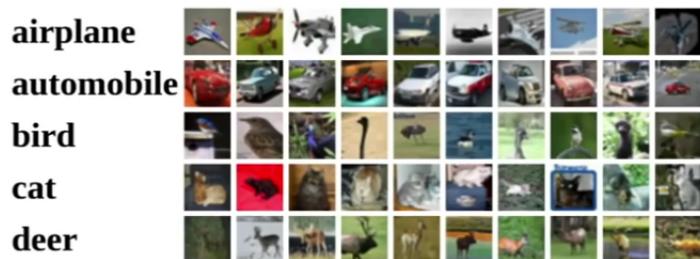
Data-Driven Approach

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

Example training set

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

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



- This Data-Driven approach is not restricted to deeplearning, **we are just utilizing the association of labels to the data.**
- Simplest approach for data-driven:
 - Nearest Neighbour

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

- To compare 2 images we need a metric

Distance Metric to compare images

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

test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
90	23	128	133	8	10	89	100	82	13	39	33
24	26	178	200	12	16	178	170	12	10	0	30
2	0	255	220	4	32	233	112	2	32	22	108

- = add → 456

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 20

Stanford University, April 3, 2017

- Algorithm for Nearest Neighbour

```
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?

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

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

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 26

Stanford University, April 3, 2017

- Disadvantage of the above approach is that, it takes a long time to predict which is not useful in real-life scenarios. We would expect to train a model for long time but need a low prediction time.

- The decision boundary would look something like this:

What does this look like?



The color of a region represents the class label and the points represent the examples belonging/classified to that class.

- Disadvantage:

- Wrong decision boundary in some cases since the decision boundary is calculated based on how near the examples are.

- Solution:

- Use K-Nearest Neighbour

- K-Nearest Neighbour

K-Nearest Neighbors

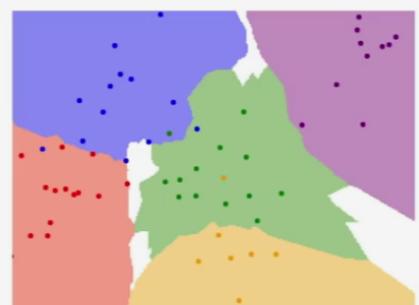
Instead of copying label from nearest neighbor,
take **majority vote** from K closest points



$K = 1$



$K = 3$



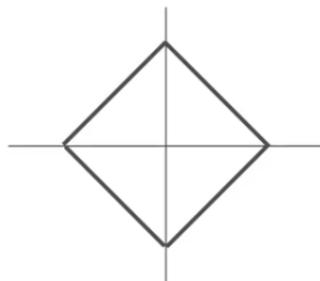
$K = 5$

- creates smoother boundary regions
- The white region shows where there is no majority. For this case, we can perform a random choice of neighbour.
- Distance Metrics:

K-Nearest Neighbors: Distance Metric

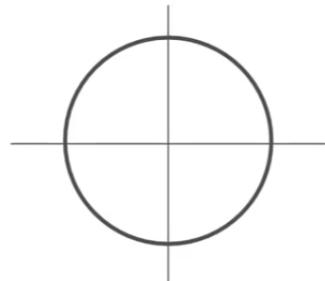
L1 (Manhattan) distance

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



L2 (Euclidean) distance

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



- If the individual entries of your vector has significance based on their position in the vector, then L1 is the wiser choice compared to L2, because L2 distance is invariant to position of entries in a vector

Hyperparameters

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

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

These are **hyperparameters**: choices about the algorithm that we set rather than learn

Very problem-dependent.

Must try them all out and see what works best.

Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

Your Dataset

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

BAD: No idea how algorithm will perform on new data

train

test

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

Better!

train

validation

test

Stanford

University

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 40

April, 2017

Setting Hyperparameters

Your Dataset

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

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

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

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

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

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 41

Stanford

University

April, 2017

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

- Very slow at test time
- Distance metrics on pixels are not informative

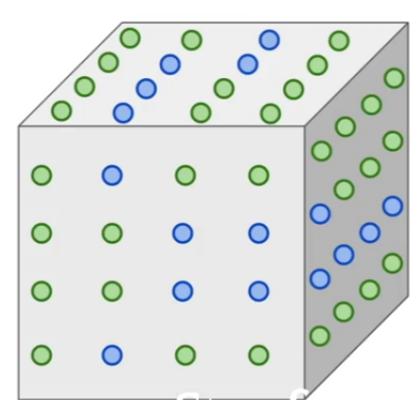
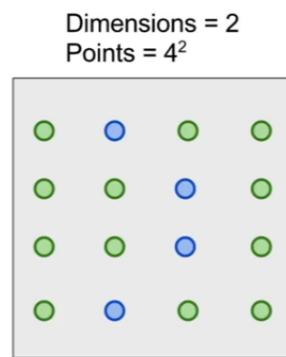
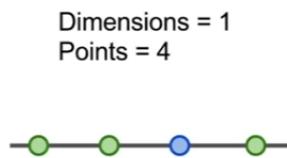


Original image is
CC0 public domain

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

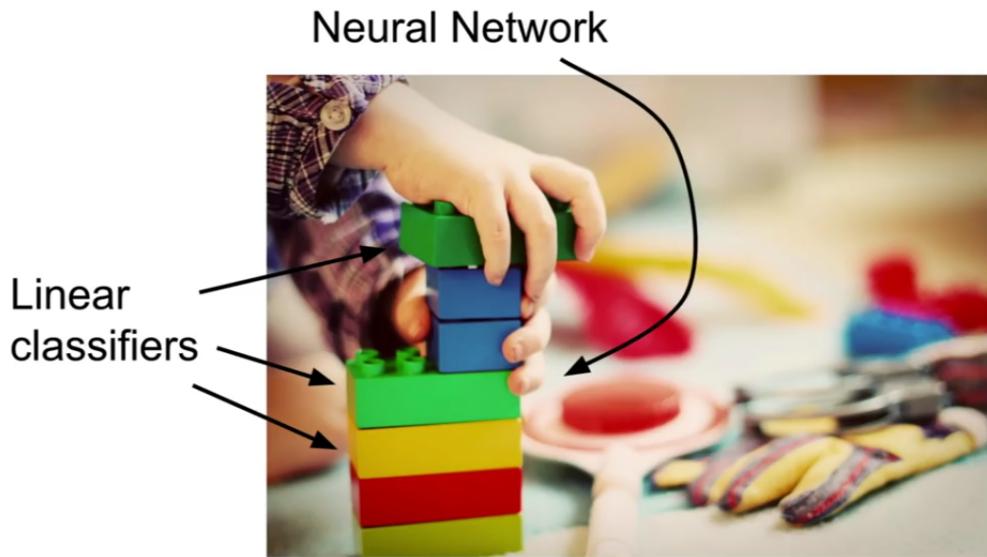
k-Nearest Neighbor on images **never used**.

- Curse of dimensionality



- Linear Classifiers

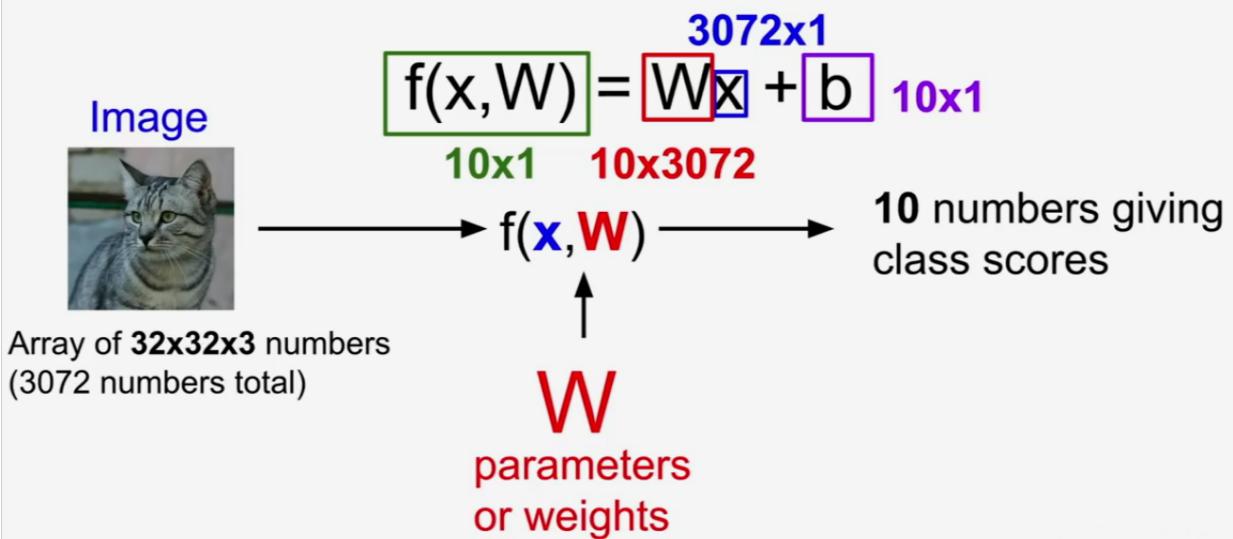
- One notion of Linear Classifiers in NN is:



This image is CC0 1.0 public domain

- These deep NN are like LEGO Blocks, we can use many such modules to solve the problem. Eg. Image Captioning: Uses CNN block to understand the image, and RNN to generate the caption.
- Parametric Approach:**

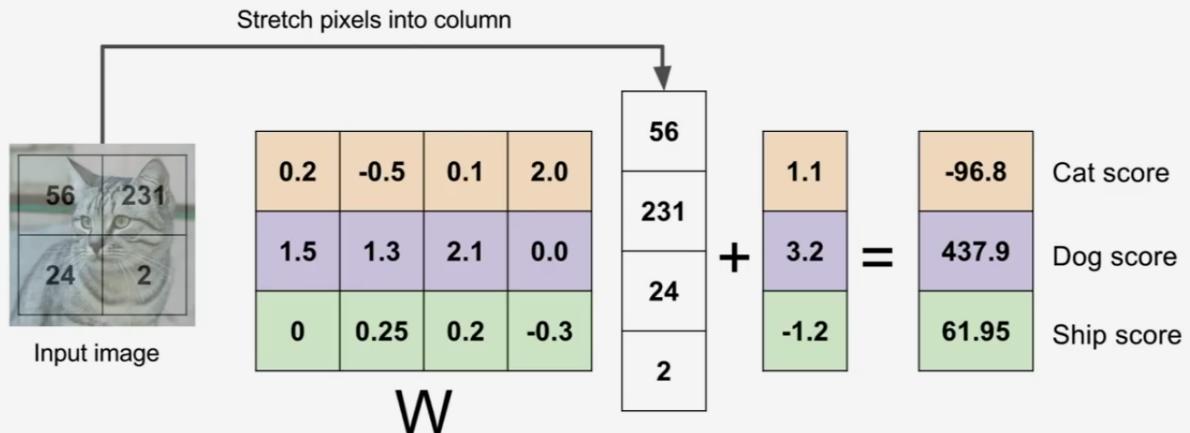
Parametric Approach: Linear Classifier



- The crux is to define the function f which better defines our training data.
- In Linear Classifier, the function $f = Wx$, i.e. it is just a multiplication operation of parameters with the input data.**
- The bias term does not interact with the training data.**
- The bias term acts as some sort of data independant preferences for some classes over another. So if our dataset(Cats vs. Dogs) has more cats than dogs then the bias**

term corresponding to the cat will be relatively higher than for the dog

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



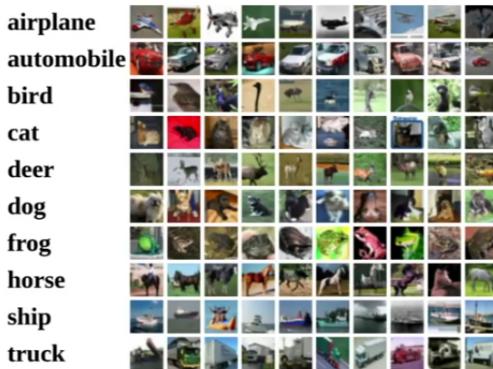
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 55

Stanford University, April, 2017

- So this Weight matrix(\mathbf{W}) acts as a template matrix where each row corresponds to a template for the corresponding class.
- We can take each row of \mathbf{W} and then reshape it to form an image to check what the classifier is learning for the corresponding class.

Interpreting a Linear Classifier



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

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



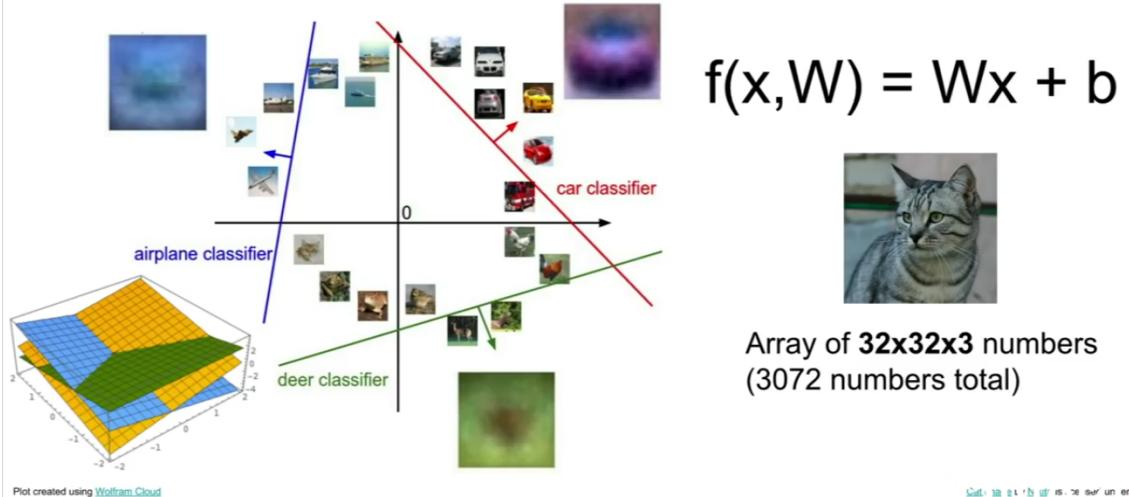
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 57

Stanford University, April, 2017

- We can notice in the second row that for the 'car' class the template is looking for a red blob with a black top blob in the image to classify it as car.
- Disadv:
 - There is only 1 template for each class.** Eg. For the car class it is looking for a "red" blob with a "black" blob to classify it as "car".
 - Solution: **Neural Networks**. Since they don't have the restriction of learning just 1 template per category.

Interpreting a Linear Classifier



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 58

Stanford University April 6, 2017

- During training the Linear Classifier will try to find the best fitting line to separate the given class with the rest of the classes.

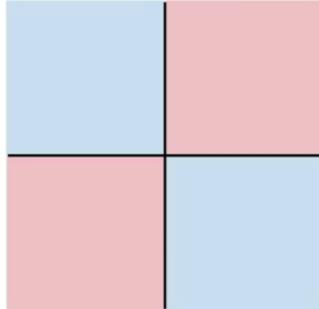
Hard cases for a linear classifier

Class 1:

number of pixels > 0 odd

Class 2:

number of pixels > 0 even

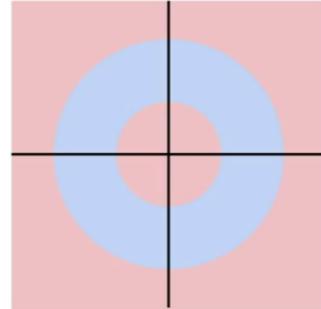


Class 1:

$1 \leq L2 \text{ norm} \leq 2$

Class 2:

Everything else

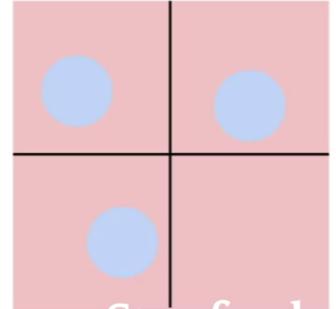


Class 1:

Three modes

Class 2:

Everything else



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 2 - 59

Stanford University April 6, 2017

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

Coming up:

- Loss function
- Optimization
- ConvNets!

(quantifying what it means to have a “good” W)

(start with random W and find a W that minimizes the loss)

(tweak the functional form of f)