**Lecture 1 | Introduction to Convolutional Neural Networks for Visual Recognition**

What is this class?

* Computer vision
* Uses visual data

Computer vision

* Used in physics, biology, engineering, mathematics, computer science

History of computer vision

* Evolution’s big bang
  + Specific short term of time when # of species were ‘exploded’.
* Camera Obscura
  + Pin hole camera theory
  + Similar to early animals’ eye
* Hubel & Wiesel, 1959
  + To find neural response mechanism or visual recognition mechanism of mammals
  + Stick electrode in cat’s brain & checked what made neuron respond excited
  + Simple cells, responded in specific moving direction
* Block world
  + Reconstruct the structure
  + “The summer vision project”
* How can we recognize structure?
  + Generalized cylinder
  + Pictorial structure
* If object recognition is too hard, maybe we should first do object segmentation
  + Task of taking image and group the pixels into meaningful areas
  + **“Image Segmentation”**
* Face detection
  + AdaBoost to do real-time face recognition
* “SIFT” feature
  + SIFT & object recognition, David Lowe, 1999
* PASCAL visual object challenge (2006~2012)
  + 20 object categories
* ImageNet / ImageNet large scale visual recognition challenge
  + to recognize the objects / overcome the ML bottleneck of overfitting, ImageNet project is launched.
  + Error rate decreased, and made lower error(3.57%) than human(5%)

Goal of this course

* Learn about Convolutional Neural Network
* Focuses on image classification
* Object detection, image captioning

Difference of 1990’s and 2010’s

* # of transistors(speed of computation)
* # of labeled data

**Lecture 2 | Image Classification**

Image classification: a core task in computer vision

* Assume given set of discrete labels and recognize the object
* Image is just a big grid of numbers between [0,255] for 3 channels RGB
* Problem: semantic gap
  + Computer may recognize pixels, but don’t understand that the pixels are for one same object. We need ML to make computer learn this pattern and recognize the pattern
  + Pixel-level to semantic-level
* Challenges
  + Viewpoint variation: tilting camera
  + Illumination: different light condition
  + Deformation: cat’s various position
  + Occlusion: we can see only a part of cat
  + Background clutter: cat’s color may similar to environment
  + Intraclass variation: cat has various color and age(representation)
* Algorithm is unlike simple sorting numbers
* No obvious way to hard-code the algorithm for recognizing a cat, or other classes

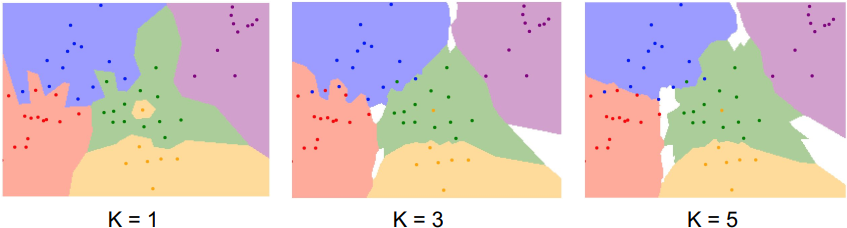
Attempts have been made

* Find edges and corners
  + Edge is one of important point to classify objects
    - Doesn’t work well: brittle and have to start all over again for other objects
* 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
  + Has two key modules: train, predict
  + First classifier: nearest neighbor
    - Train: memorize all data and labels
    - Predict: predict the label of the most similar training image
  + Example dataset: CIFAR10
    - 10 classes, 50k training images, 10k testing images

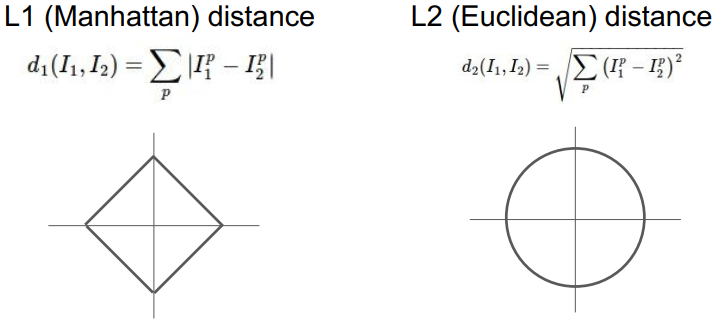
**K-nearest neighbors**

How to compare images? – Distance metric to compare images

* L1 distance (Manhattan)
  + Comparing single pixel of images
  + Add pixel-wise absolute value differences
* With N examples, how fast are training and prediction?
  + Train O(1), predict O(N)
  + Not good: we want fast at prediction. Slow for training is okay
* K-nearest neighbors
  + Instead of copying label from nearest neighbor, take majority vote from K closest points
  + If K is big, tends smother edges



* L2 distance (Euclidean)



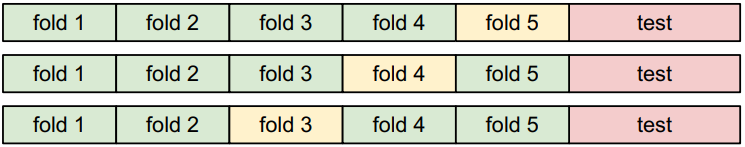
* L1 and L2 distance
  + L1 depends on choice of coordinate system
  + If we rotate the coordinate frame, L1 distance changes but L2 doesn’t.
  + If the input features or individual entries in vector has some important meaning for task, L1 distance is a natural fit
  + If the input vector is a generic vector in some space and don’t know which of the different elements does actually mean, L2 distance is a natural fit.
* K-nearest neighbors with L1/L2 distance
  + Boundaries changes slightly for each method.

Hyperparameters

* What is the best value of **k** to use?
* What is the best **distance** to use?
* Hyperparameter: k and distance, choices about the algorithm that we set rather than learn
* Problem dependent, must try them all out and see what works best

Setting hyperparameters

* Idea 1: choose hyperparameters that works best on the data
  + Very bad idea: k=1 always works perfectly on training data
  + If the k gets bigger, it might cause us to misclassify some of the training data.
  + But it leads to better performance on the data which were not in training data
  + Ultimately, we really care about how our classifier or method performs on unseen data out of training data.
* Idea 2: split data into train and test, choose hyperparameters that work best on test data
  + Seems more reasonable, but also a terrible idea.
  + No idea how algorithm will perform on new data
* Idea 3: split data into train, validation, test and choose hyperparameters on val and evaluate on test
  + Try many hyperparameters at the training set and evaluate on validation set. Choose hyperparameters that best for the validation set, and run once on the test set.
  + Better method
* Idea 4: cross-validation; split data into folds, try each fold as validation and average the results



* + Change validation sets for each fold
  + Useful for small datasets, but not used too frequently in deep learning

k-nearest neighbor on images never used

* Very slow at test time
* Distance metrics on pixels are not informative
* Curse of dimensionality: needs more dots for bigger dimensions(grows exponentialy)

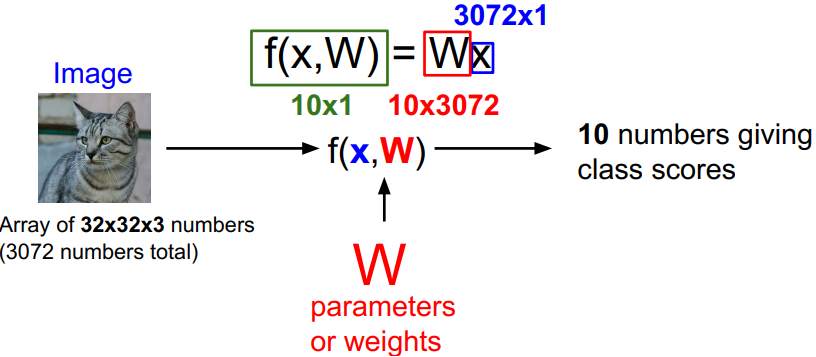
k-nearest neighbors: summary

* 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 nearest training examples
* Distance metric and k are hyperparameters
* Choose hyperparameters using the validation set; only run on test once at the very end

**Linear classification**

Parametric approach

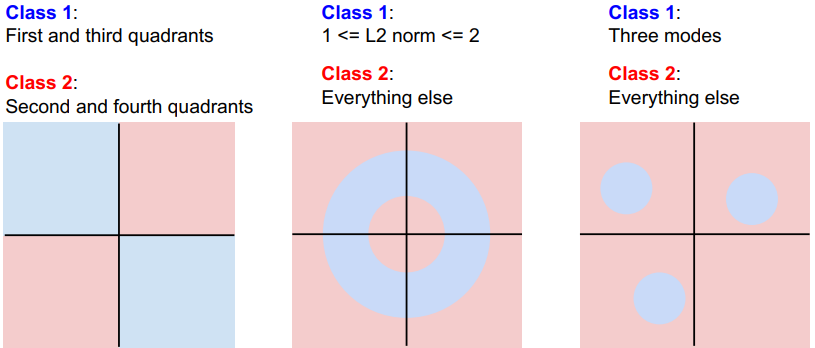
* Recall CIFAR10; 50k different image with 32\*32\*3
* Input data: image, array of 32\*32\*3 numbers (3072 numbers total)
* Make function of input data and parameters or weights(W)
* Output: 10 numbers giving class scores



* Simple function: f(x, W) = Wx +b (b: bias)

Interpreting a linear classifier

* Make a single linear line of classifying objects
* If object is over the line, then it is classified with that group
* Hard cases for s linear classifier



* + Hard to make a linear line to separate into two groups
  + Parity problem separating odd from even/multimodal situations