

CS231n: Deep Learning for Computer Vision Week 1 Lecture Notes

**You can access the official lecture slide [here](#)*

Agenda

1. Image Classification Task Overview
2. Two basic data-driven approaches to image classification:
 - a. k-nearest neighbor
 - b. linear classifier

Image Classification Task Overview

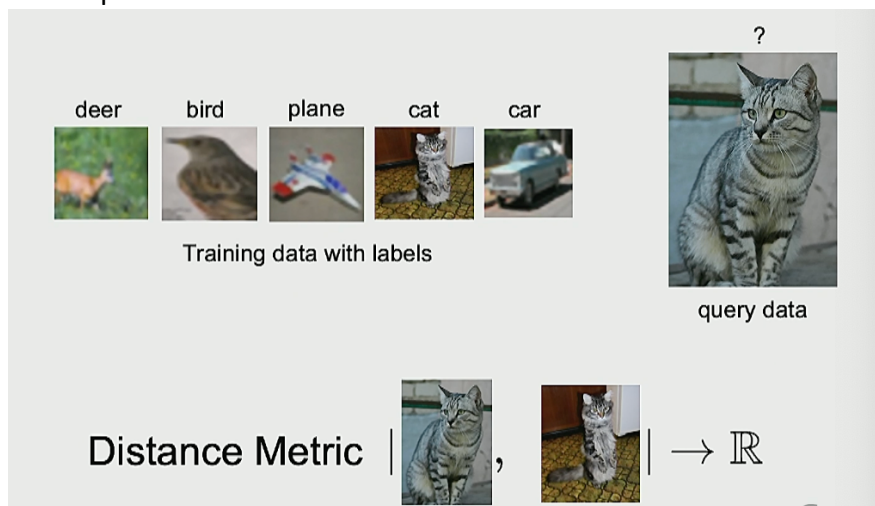
Challenges:

- Semantic gap between human perception and machine perception: 3 channels RGB is what the computer sees (a tensor of integers between 0 and 256)
- Background clutter: e.g. background shares similar colors
- Occlusion: e.g. cats hide in the bushes
- Intraclass variation: many cats together to distinguish each individual one
- Context: dog with stripes casted by door shadows mistaken as tiger

K-nearest Neighbor

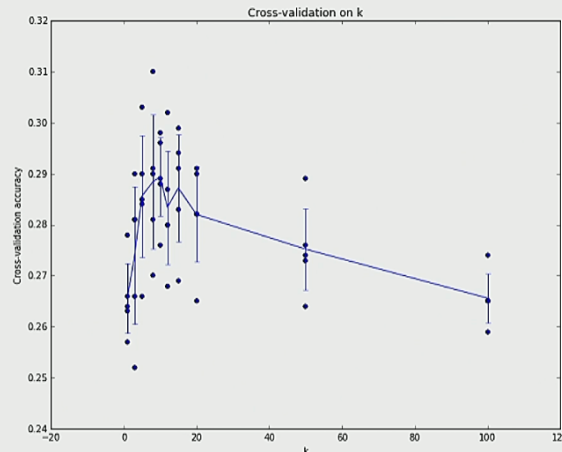
Modern computer vision algorithms: imagenet

- Data-driven approach with ML:
 - o Step 1: Collect a dataset of images and labels
 - o Step 2: Use machine learning algorithms to train a classifier
 - o Step 3: Evaluate the classifier on new images
- Nearest Neighbor Classifier
 - o Training function: memorize all data (training images) and labels, essentially building a look up table
 - o Prediction function: find a nearest neighbor image using the label from training sets as prediction



- L1 distance (Manhattan): most basic distance metric
- L2 distance (Euclidean): makes different assumptions about the data
- Hyperparameters:
 - What is the best value of k to use?
 - Highly dataset dependent
 - What is the best distance to use?
- How to set hyperparameters? (hint: train, validation, test split)
 - 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
 - Idea 3: split data into train, val; choose hyperparameters on val and evaluate on test
 - Idea 4: cross-validation: split data into folds, try each fold as validation and average the results
 - Useful for small datasets, but not used too frequently in deep learning
- CIFAR10: one of the largest computer vision data set
 - 10 classes
 - 50k training images
 - 10k test images

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 $k \approx 7$ works best for this data)

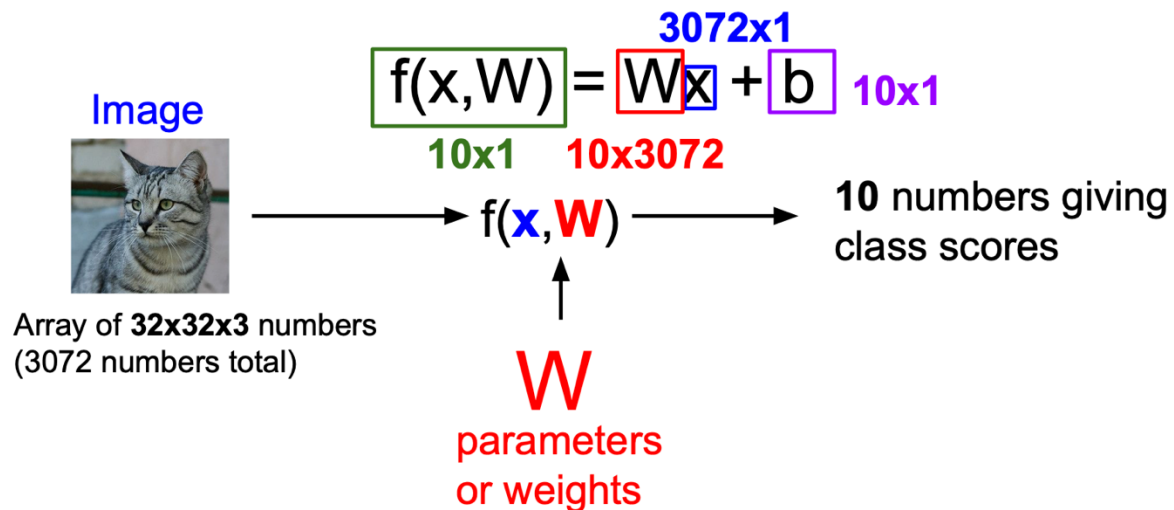
- Summary
 - In image classification we start with training set of images and labels, and must predict labels on the test set
 - The k -nearest neighbors classifier predicts labels based on the K nearest training examples
 - Distance metric and K are hyperparameters
 - Choose hyperparameters using the validation set
 - Only run once on the test dataset at the very end!

Linear Classifier

Unlike nearest neighbor which is rarely used in deep learning models, this is a very important building block for deep learning neural network.

- Parametric approach: class score does not have any probabilistic component, the higher the score, the more likely it's correct

Parametric Approach: Linear Classifier



- Choose a good W :
 - o Define a loss function that quantifies our unhappiness with the scores across the training data
 - Higher loss: unhappy with the classifier, not performing well
 - Loss function tells how good our current classifier is
 - Loss over the dataset is the average of loss over examples
 - o Come up with a way of efficiently finding the parameters that minimize the loss function
 - o Multiclass SVM loss
 - Pairwise comparison between the correct class score and every other classes




the SVM loss has the form:

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

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

- How it works:

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: $s = f(x_i, W)$

the SVM loss has the form:

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

$$= \max(0, 2.2 - (-3.1) + 1) + \max(0, 2.5 - (-3.1) + 1)$$

$$= \max(0, 6.3) + \max(0, 6.6)$$

$$= 6.3 + 6.6$$

$$= 12.9$$

Stanford

Softmax Classifier

Multinomial logistic regression: interpret raw classifier scores as probabilities

- Probabilities must be ≥ 0
- Probabilities must sum to 1
- Cross entropy

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

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

Softmax Function

Coming up

- Regularization
- Optimization