Image Classification: K-NN and Linear Classifier

Xiaolong Wang

Coming Assignments

- The first assignment will be announced in This Thursday after the class

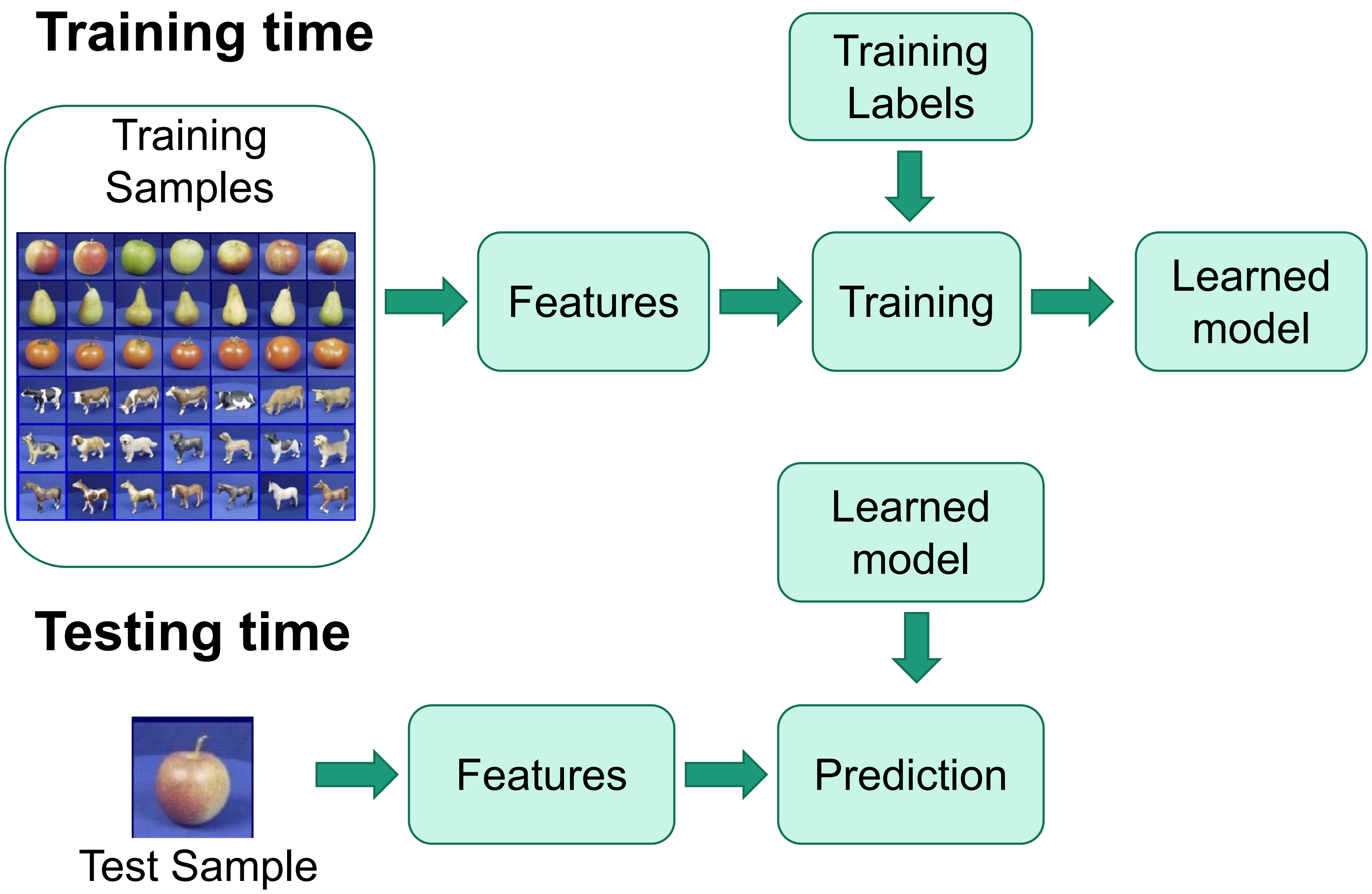
- There will be a tutorial on how to do/submit assignments This Friday, 4:00 - 5:00 pm

- We will use the compute resources in https://datahub.ucsd.edu/

Last class

- Overview of deep learning, applications on computer vision, NLP, robotics
- The concept and goal of learning

Last class


Training time
Training Samples
Training Labels
Features
Training
Learned model
Testing time
Learned model
Test Sample
Features
Prediction

Today: Two basic methods

- Nearest Neighbors
- Linear Classifier

Image Classification

An image is a 300 x 500 x 3 Tensor.

Each bit has value in the range [0, 255]

Images with different background

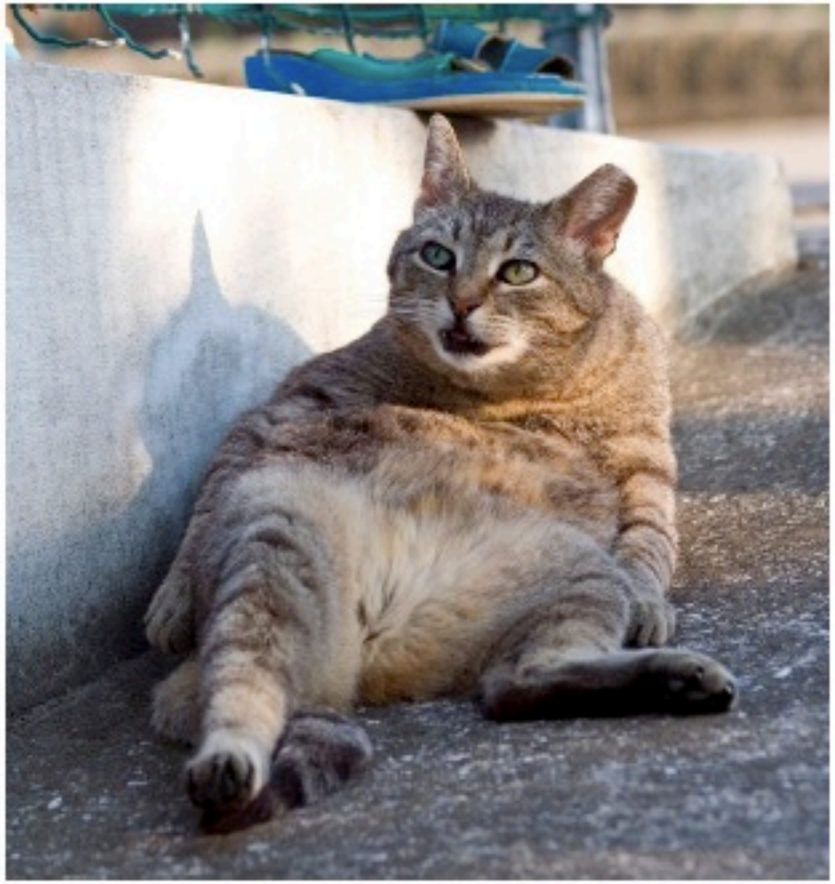
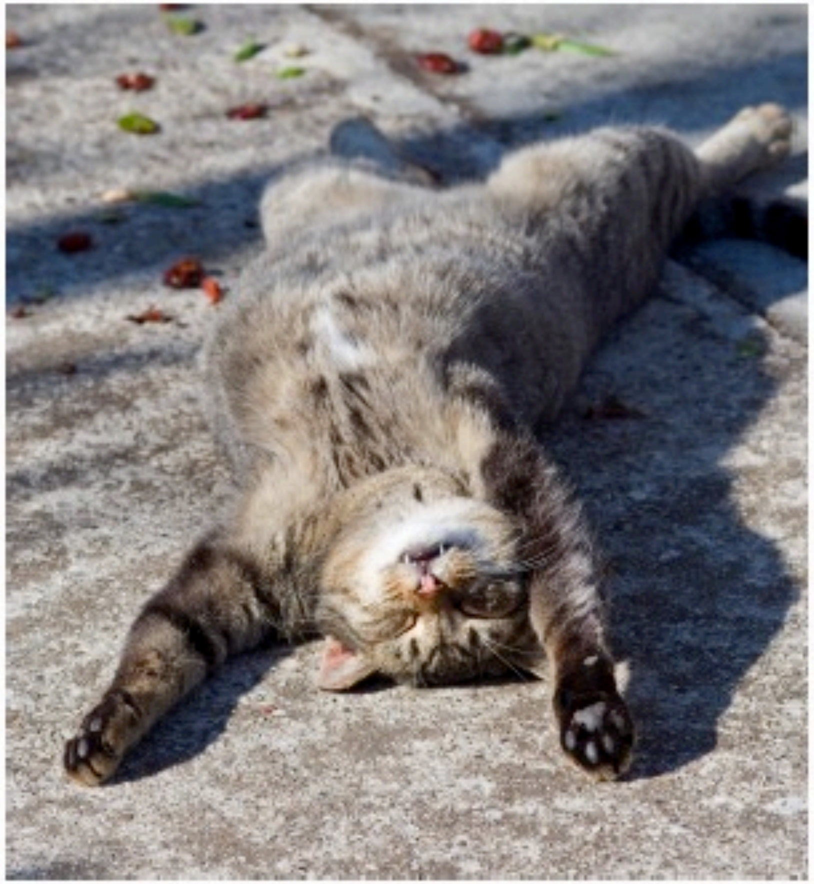
http://cs231n.stanford.edu/

Images with occlusion

Images with illumination



Images with Deformation


Nearest Neighbor Classifier

Nearest Neighbor

Training set:

Mushroom

Dog

Ant

Cat

Car

Testing: Compute the distance between a test image and training images

|  ,  | $\rightarrow \mathbb{R}$

|  ,  | $\rightarrow \mathbb{R}$

Nearest Neighbor

- What metric? What representation?
- Metric, L1 distance:

$$d(x_1, x_2) = \sum_{h,w} \left| x_1^{h,w} - x_2^{h,w} \right|$$

$$\left| \overset{\text{test image}}{\begin{array}{|c|c|c|c|} \hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline \end{array}} - \overset{\text{training image}}{\begin{array}{|c|c|c|c|} \hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline \end{array}} \right| = \overset{\text{pixel-wise absolute value differences}}{\begin{array}{|c|c|c|c|} \hline 46 & 12 & 14 & 1 \\ \hline 82 & 13 & 39 & 33 \\ \hline 12 & 10 & 0 & 30 \\ \hline 2 & 32 & 22 & 108 \\ \hline \end{array}} \xrightarrow{\text{add}} 456$$

Recall Supervised Learning

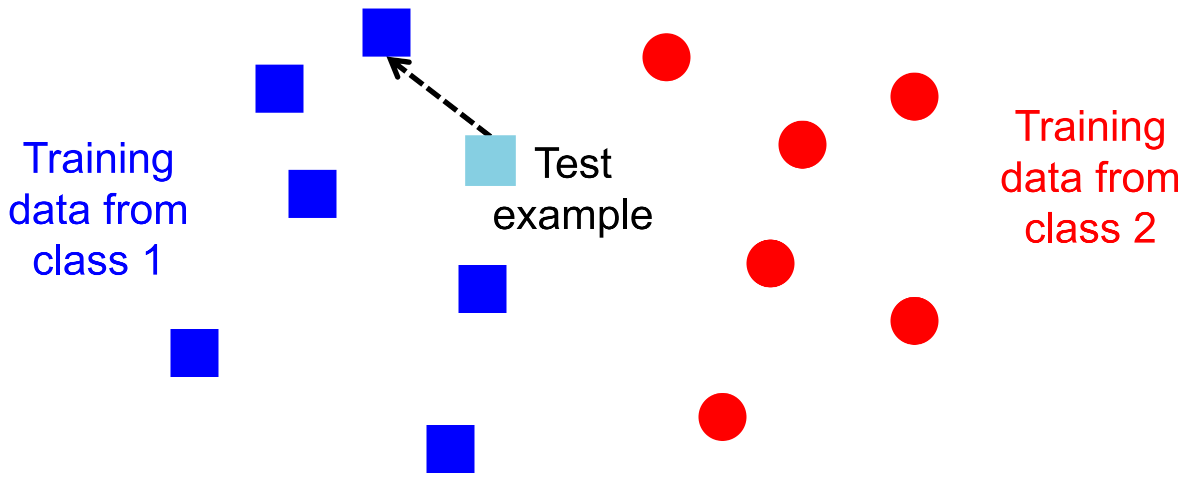
$$y = f(x)$$

output label — classifier — input image

- **Training** (or **learning**): given a *training set* of labeled examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$, train a predictor f

- **Testing** (or **inference**): apply predictor f to a new *test example* x and output the predicted value $y = f(x)$

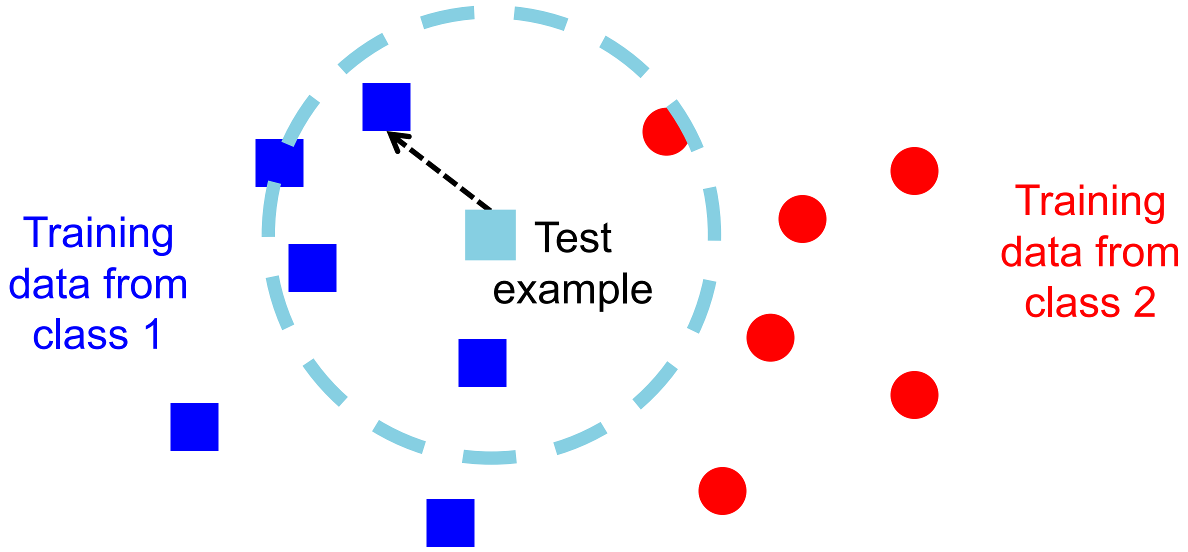
Nearest neighbor classifier



- $f(x)$ = the label of the closest example (computed via a distance metric)

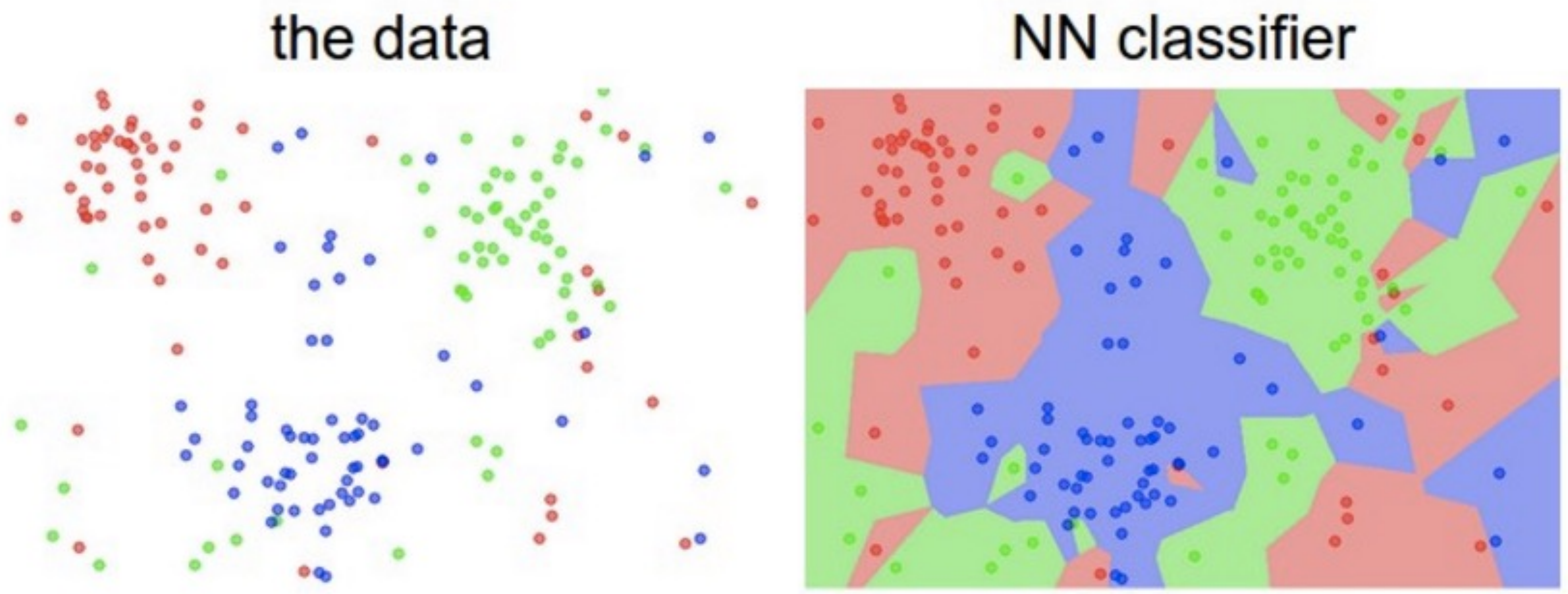
- Store all the training data, search all data each test time given a test example

K-nearest neighbor classifier



- 1 example is sometimes not enough.
- K-NN, K=5: Find closest 5 examples instead of 1. Follow the label of the majority in the NN examples.

K-nearest neighbor classifier

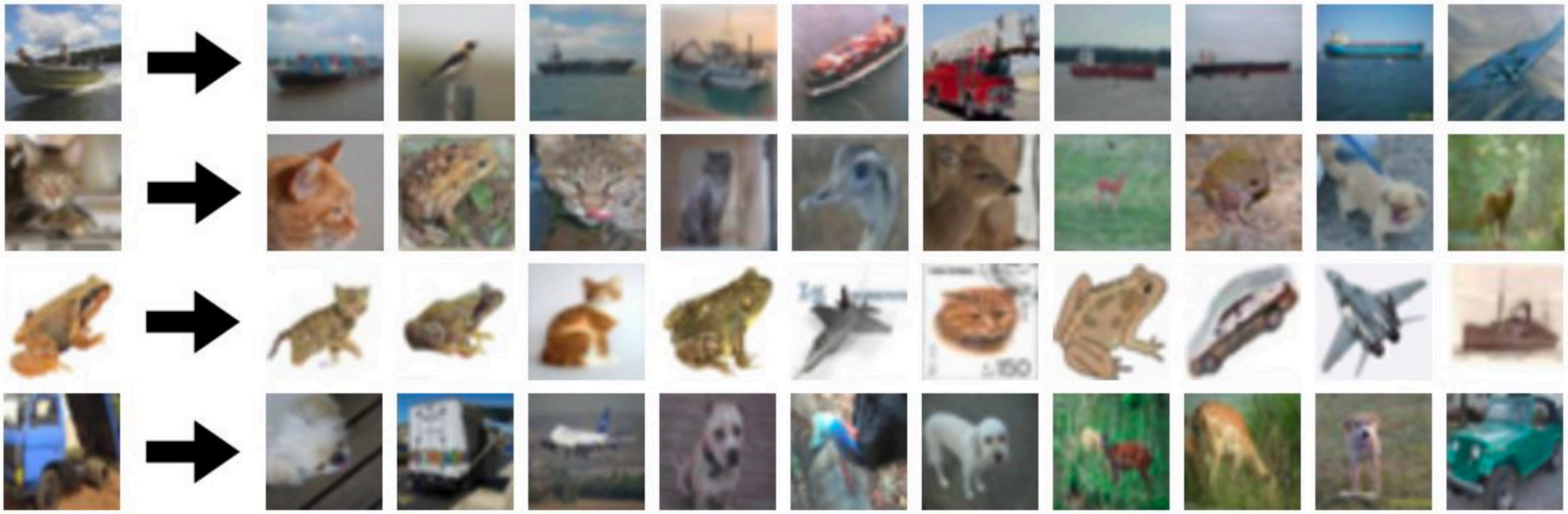


Larger K gives cleaner boundary between classes

Larger K is more robust to outliers

Credit: Andrej Karpathy, http://cs231n.github.io/classification/

K-NN examples (K=10), based on pixel-wise difference



The algorithm

- Extract the features of each image in the training data, and record the corresponding labels

- Given a test image, extract the feature, and compute the distance between the test image and the whole training dataset

- Select the top-K Nearest Neighbors and obtain their corresponding labels

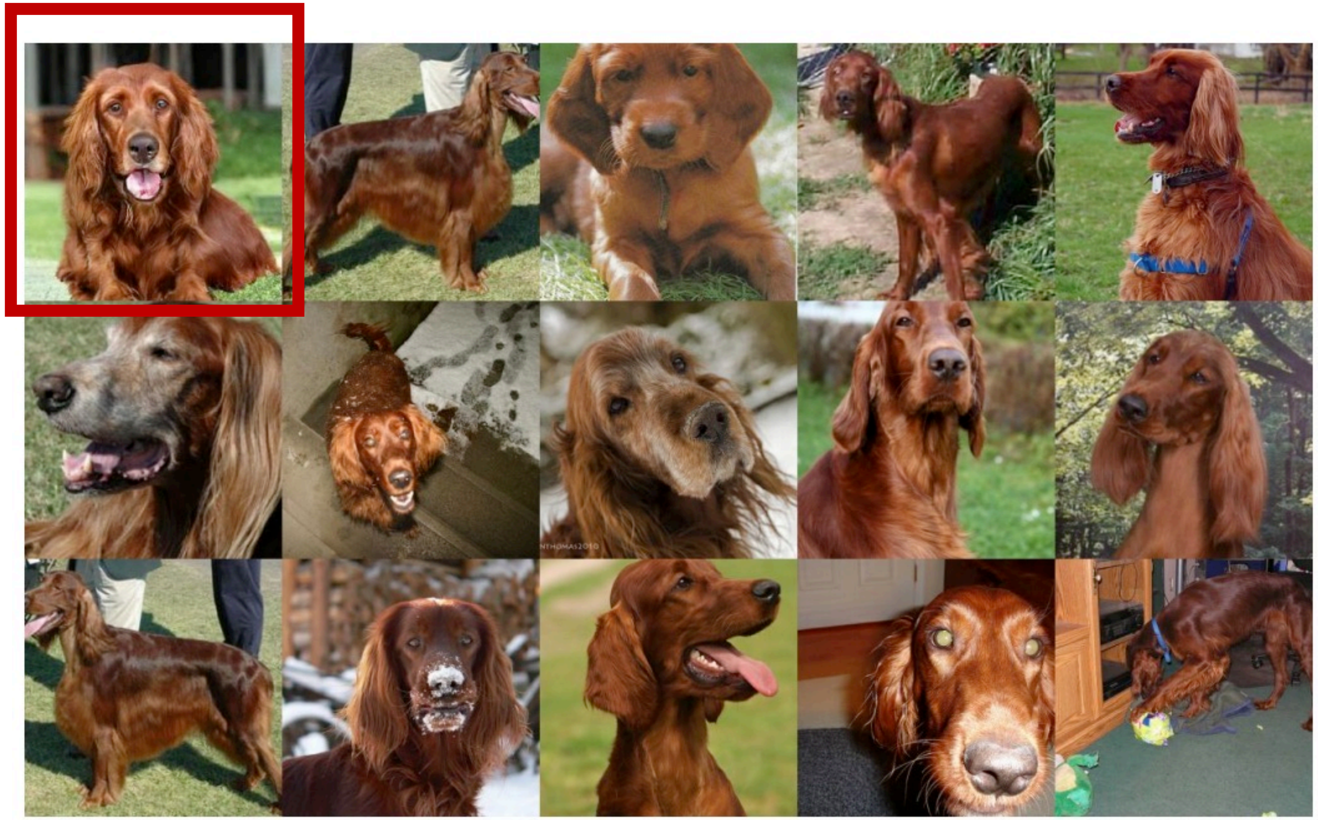
- The test image is classified as the majority class in the K-NN examples

Tunning Hyperparameters

- What is the best K to use?
- What is a good distance metric?
  - L1 distance: $d(x_1, x_2) = \sum_{h,w} \left| x_1^{h,w} - x_2^{h,w} \right|$
  - L2 distance: $d(x_1, x_2) = \sum_{h,w} \left|\left| x_1^{h,w} - x_2^{h,w} \right|\right|_2^2$

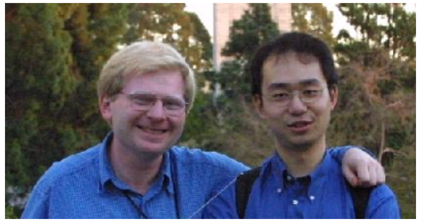
Nearest Neighbor is a great way for visualization neural network

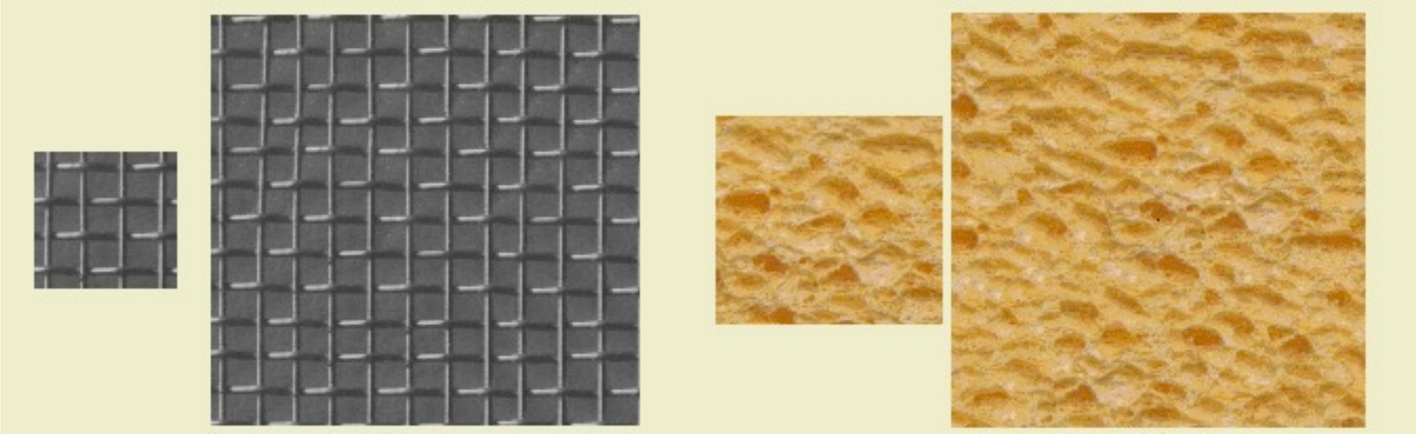
Query

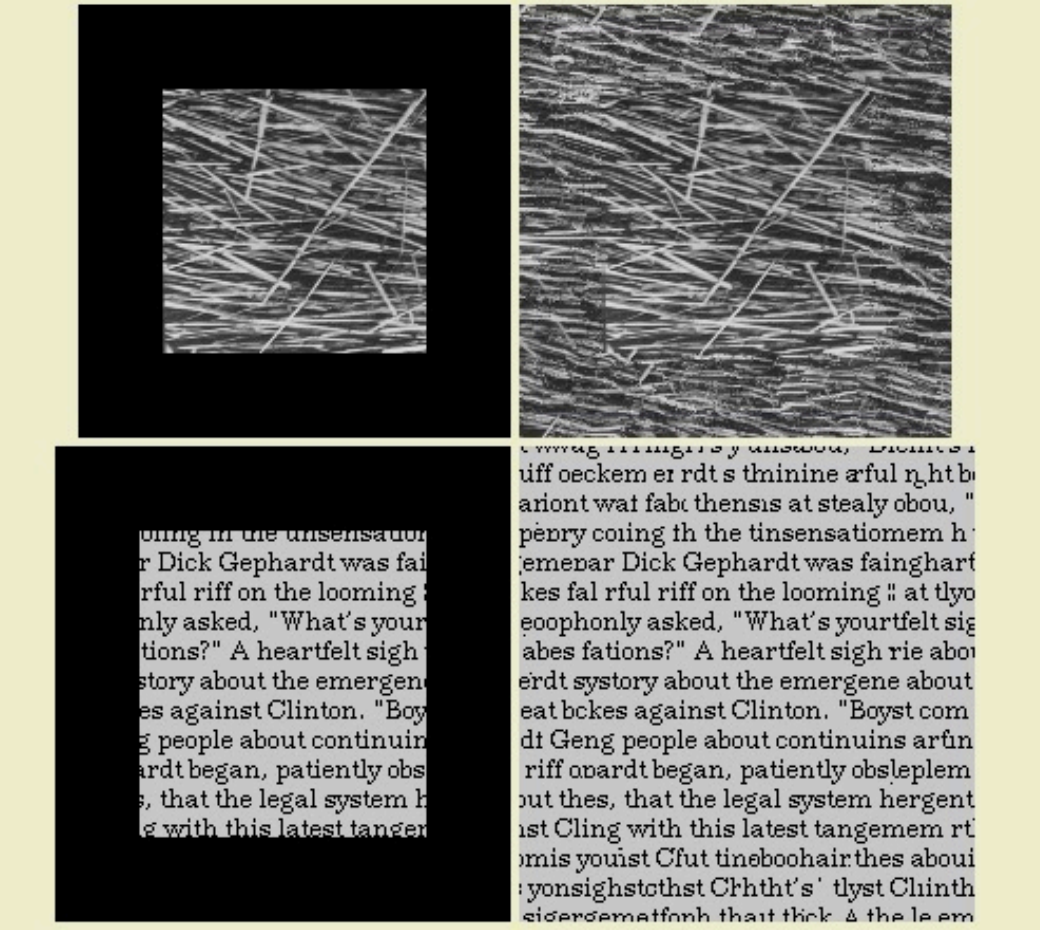


GANs (Brock et al., 2019)

Texture Synthesis







https://people.eecs.berkeley.edu/~efros/research/NPS/alg.html

``Texture Synthesis by Non-parametric Sampling''
Alexei A. Efros and Thomas K. Leung, ICCV 1999.

Goods and Bads of Nearest Neighbor

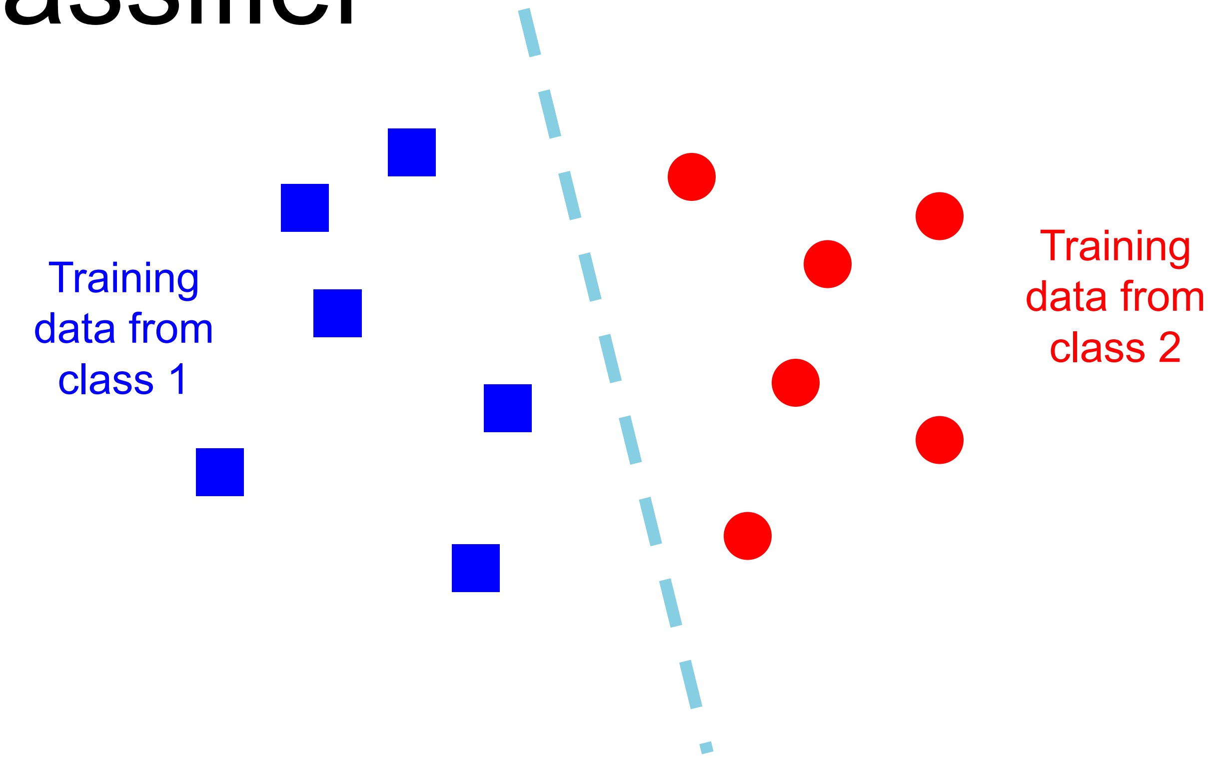
- Good:
  - Do not require training
  - Simple and robust to outliers

- Bad:
  - Storage: needs to store the whole dataset
  - Time: needs to go over each training data point, inference time grows linearly as the training data increases

- Can we *compress* the training samples to a set of weights?

Linear Classifier

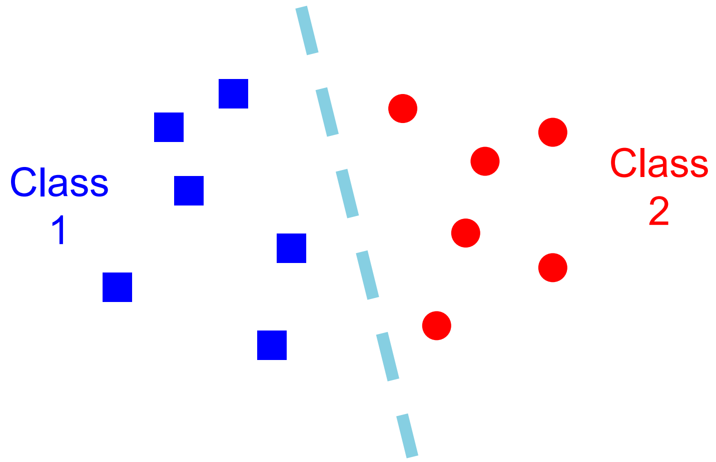
Linear Classifier



- Goal: Learn a d-dimentional vector of parameters $W \in \mathbb{R}^d$, given a set of d-dimentional data

- Prediction: $f(x) = W_1 x_1 + W_2 x_2 + \ \ldots + W_d x_d = Wx$

Linear Classifier

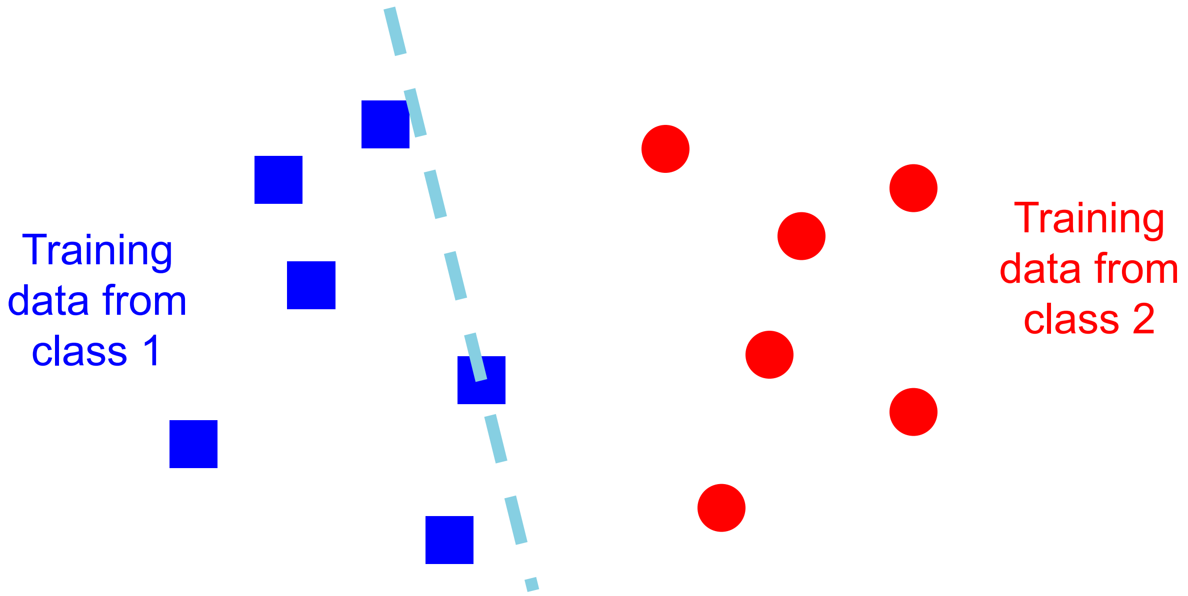


- Prediction: $f(x) = W_1 x_1 + W_2 x_2 + \ldots + W_d x_d = Wx$

- If $f(x) > 0$, x belongs to class 1, if $f(x) < 0$, x belongs to class 2.

- See W as the compression of the whole training dataset, and we only need to compute 1 multiplication for obtaining the label.

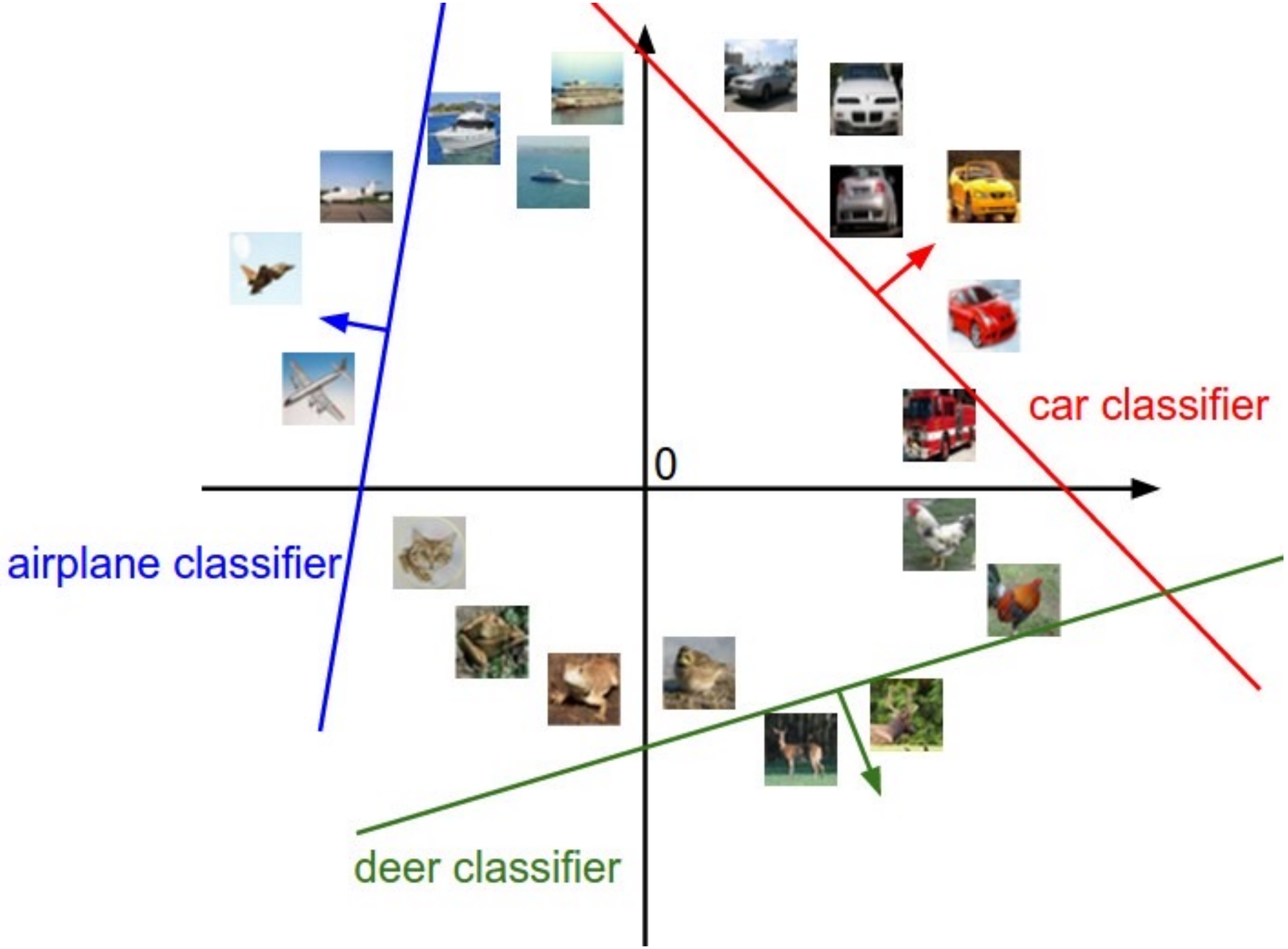
Linear Classifier: adding bias



- Prediction: $f(x) = W_1x_1 + W_2x_2 + \ ... + W_dx_d + b = Wx + b$
- $b \in \mathbb{R}^1$, b is only a 1-dimentional digit for 2-class classification

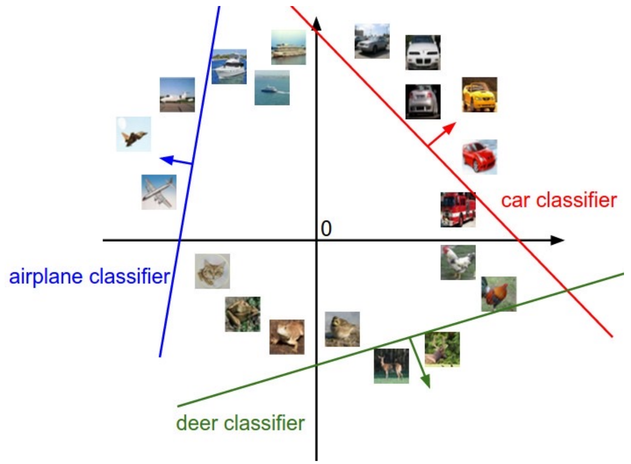
Linear Classifier: Multiple Class

- 1 plane is not enough
- Multiple planes



Source: Andrej Karpathy, http://cs231n.github.io/linear-classify/

Linear Classifier: Multiple Class

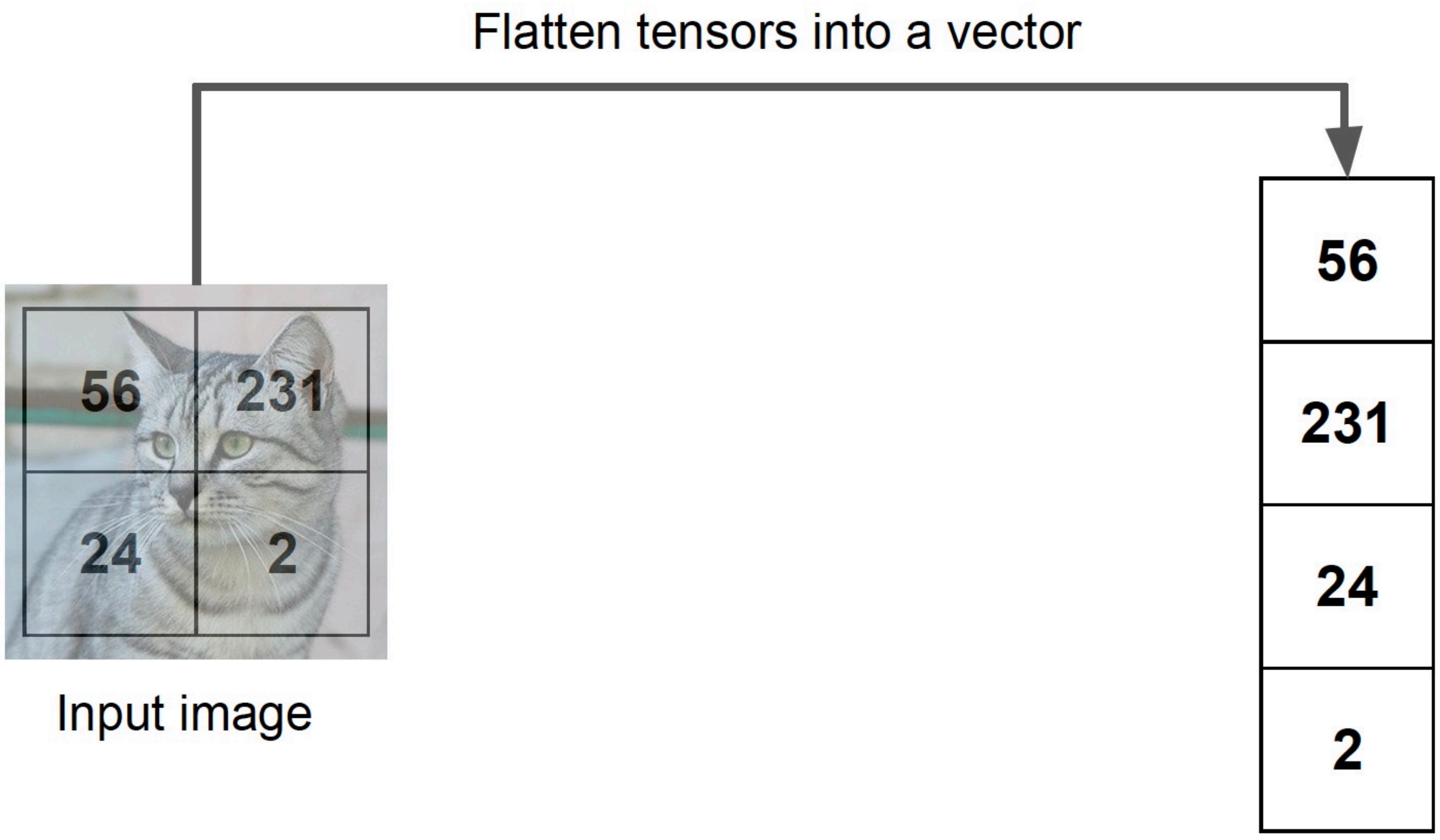


- Instead of learning one vector of weights, we will need to learn one vector of weights for each category:

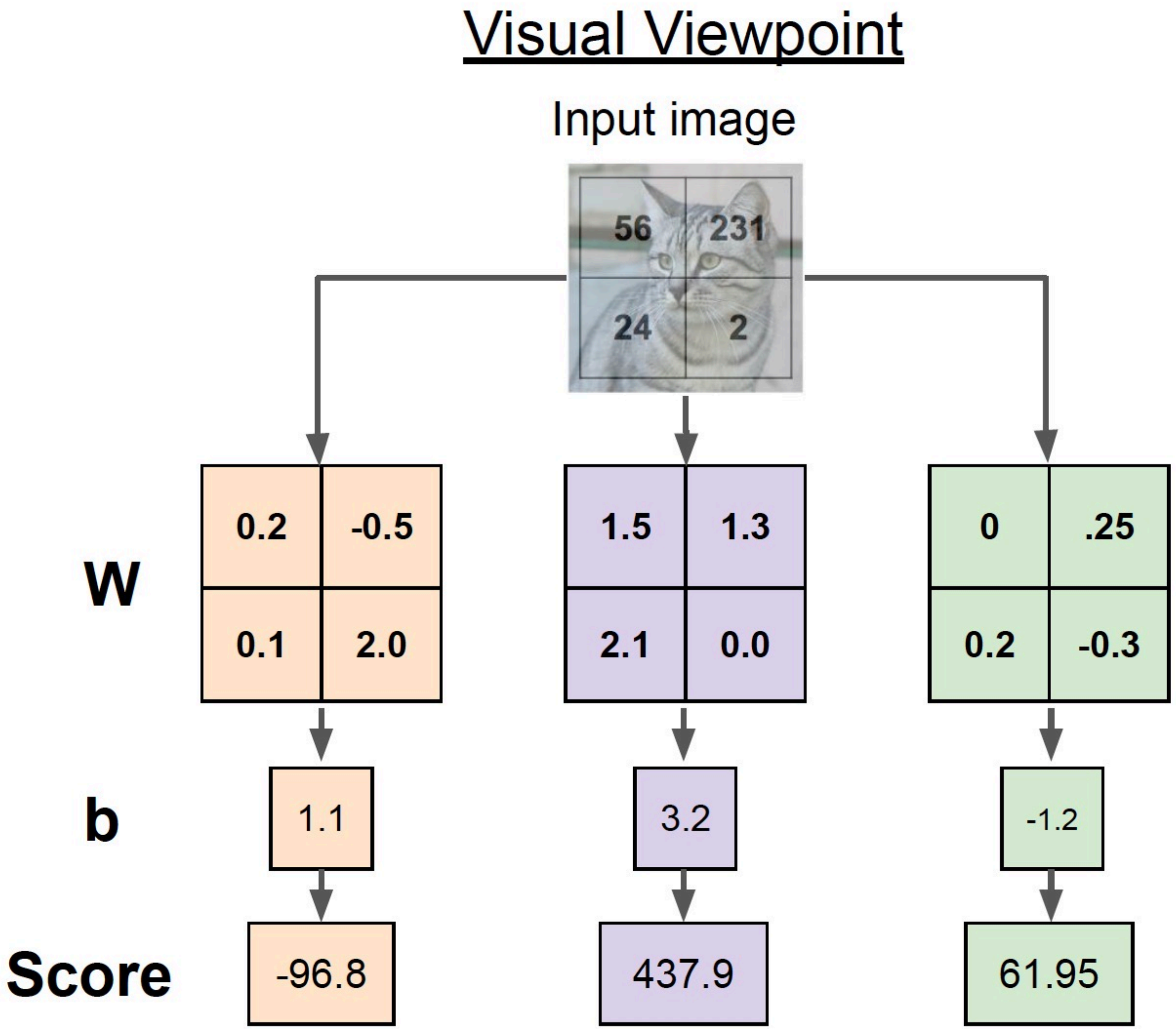
  - A dog classifier: $f_1(x) = W^1x + b^1$
  - A cat classifier: $f_2(x) = W^2x + b^2$
  - A ship classifier: $f_3(x) = W^3x + b^3$

- Select the class with the max classification score

Example: Represent an image with 4 pixels



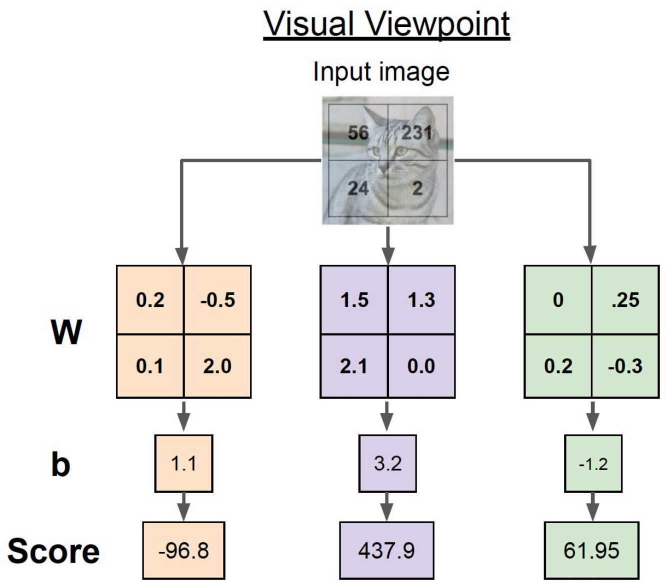
Example: Represent an image with 4 pixels



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

$$x \in \mathbb{R}^{3072}\ (32{\times}32{\times}3)$$
$$W \in \mathbb{R}^{3072}$$
$$b \in \mathbb{R}^{1}$$

Example: Represent an image with 4 pixels



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

$$x \in \mathbb{R}^{3072} \ (32\times32\times3)$$
$$W \in \mathbb{R}^{3072}$$
$$b \in \mathbb{R}^{1}$$

Visualizing W in 10 different classes:

Training the Linear Classifier

- Linear regression
- Logistic regression (next class)

Training with Linear Regression

- Given the training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, drawn from distribution D.

- Find predictor $f(x)$ so that it performs well on test (unseen) data drawn from the same distribution D.

- Potential problem: What if the data is not taken from the same distribution D?

How to evaluate ”performs well”?

- Define an expected loss as,

$$\mathbb{E}_{(x,y)\backsim D}[l(f, x, y)]$$

- To approximate the loss using N examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$,

$$\frac{1}{N}\sum_{i=1}^{N} l(f, x_i, y_i)$$

Linear Regression

- Loss: Using L2 distance:

$$l(f, x_i, y_i) = (f(x_i) - y_i)^2 = (Wx_i + b - y_i)^2$$

- Average through all the examples

$$\frac{1}{N}\sum_{i=1}^{N}(Wx_i + b - y_i)^2$$

Linear Regression

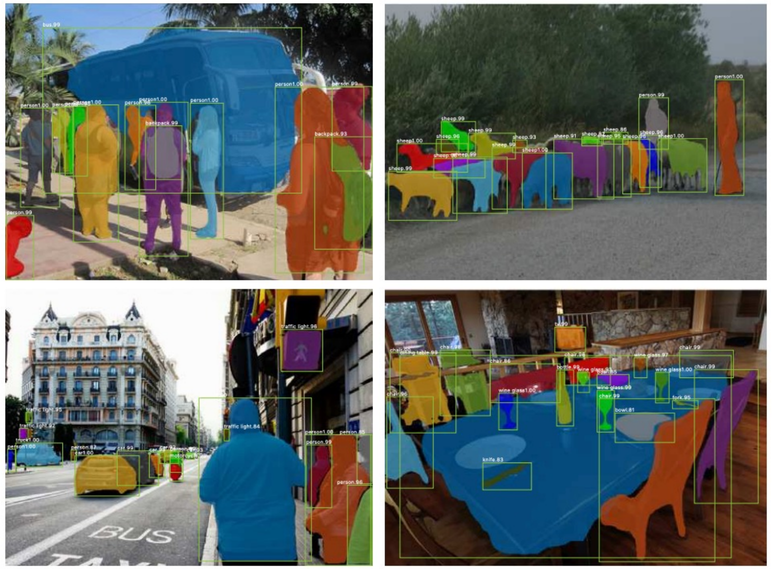
$$\frac{1}{N}\sum_{i=1}^{N}(Wx_i + b - y_i)^2$$

- In two-class classification: $y \in \{-1,1\}$. However, there is no regulation to constrain the output range.
- In multiple-class case, for each class we perform two-class classification: $y \in \{-1,1\}$.
- Not convenient for classification

What problem can be solved using linear regression

- Predicting a continuous number instead of category ID

- Predicting bounding box location, human pose location.



What problem can be solved using linear regression

- Video Prediction, Colorization

What I have not talked about yet

Optimization of linear classifier using the loss function (next class)

Compare K-NN and Linear classifier

- Do not need training
- Time consuming in test time
- Non-parametric, explicitly search through data
- More robust to outliers, using larger K

- Need training
- Time efficient in test time
- Parametric, use parameters to ”memorize” the dataset
- Can be sensitive to outliers

In this class

- K-nearest neighbor classifier
- Linear classifier
- Training linear classifier with linear regression (loss funnction)

Next class

- More on linear classifier
- Loss function
- Optimization of linear classifier
- Regularization