Visual Classification 1: Intro and Linear Methods

CSE P576

Vitaly Ablavsky

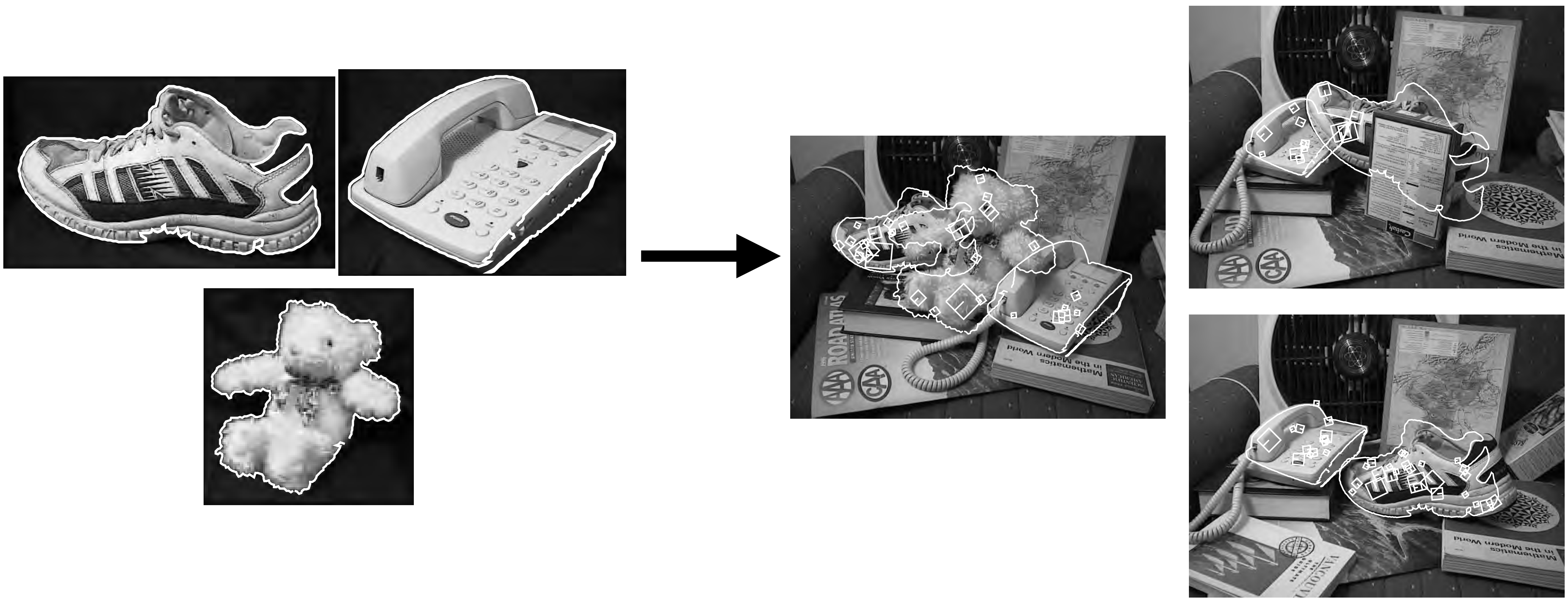
These slides were developed by Dr. Matthew Brown for CSEP576 Spring 2020 and adapted (slightly) for Fall 2021
credit → Matt
blame → Vitaly

Visual Classification 1

- Object recognition: instance, category
- Image classification vs object detection
- Linear classification, CIFAR10 case study
- 2-class, N-class, linear + softmax regression

Object Recognition

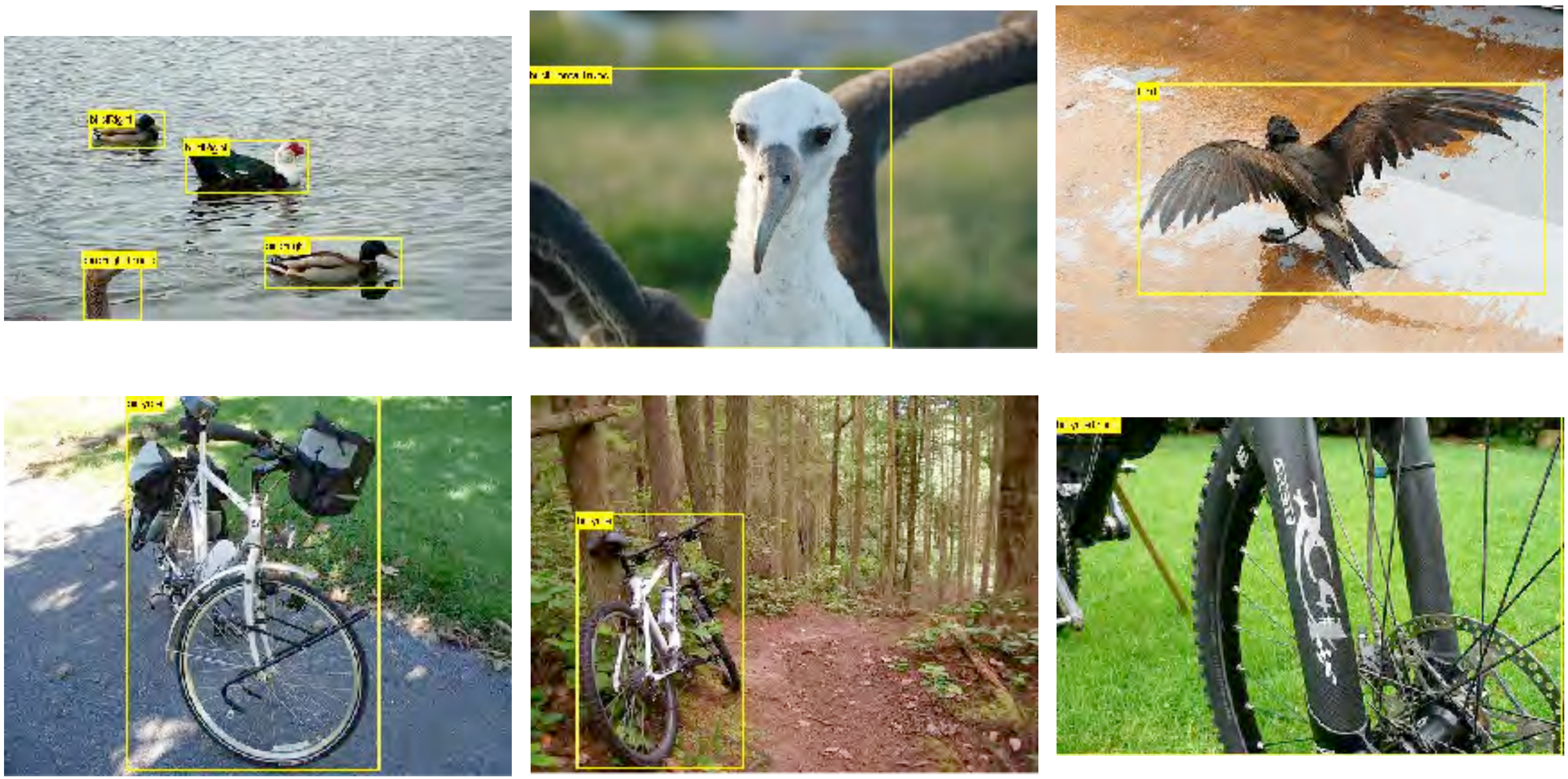
- Object recognition with SIFT features [Lowe 1999]



What is present? Where? What orientation?

Object Recognition

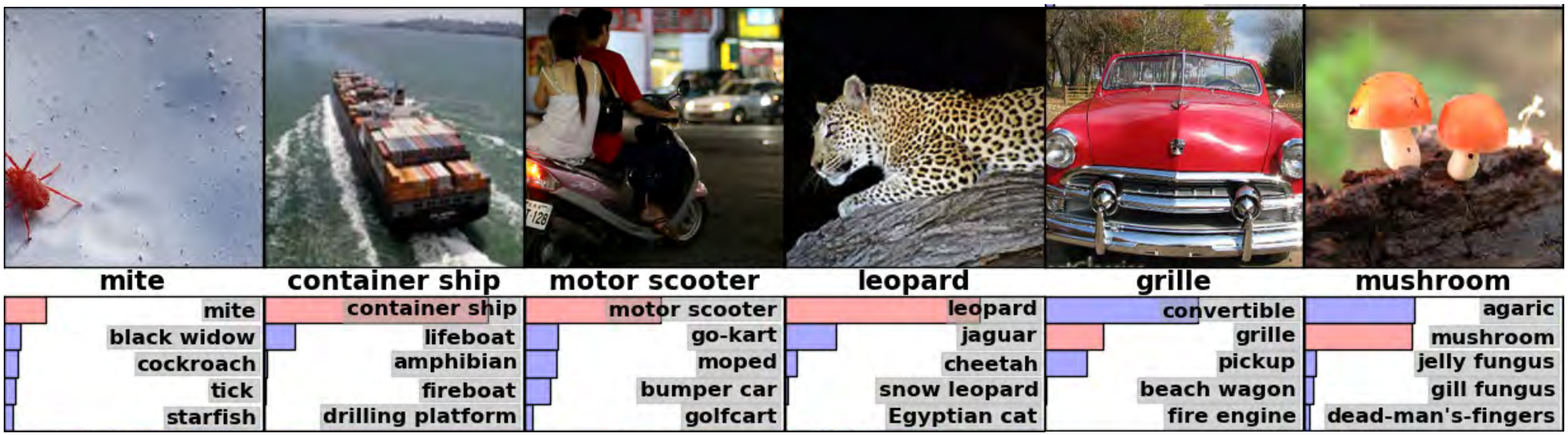
- PASCAL Visual Object Classes Challenges [2005-2012]



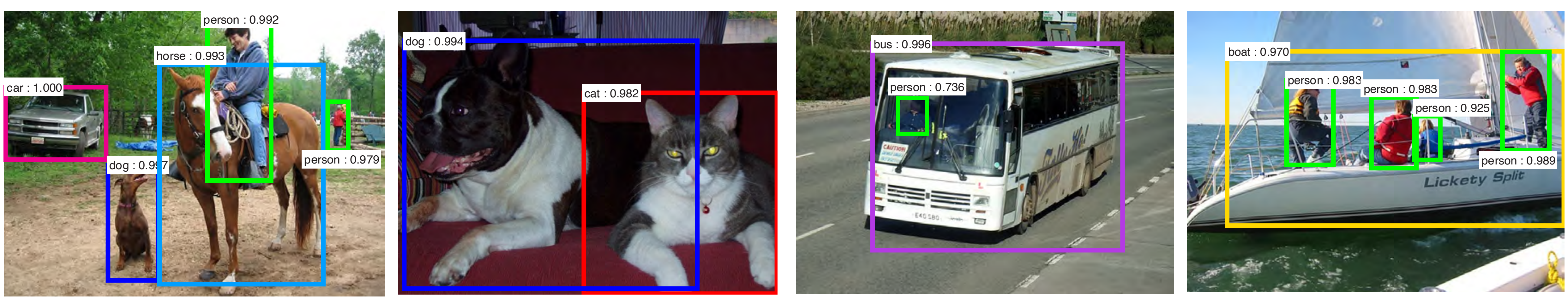
What is present? Where? What orientation?

Classification and Detection

- Classification: Label per image, e.g., ImageNet



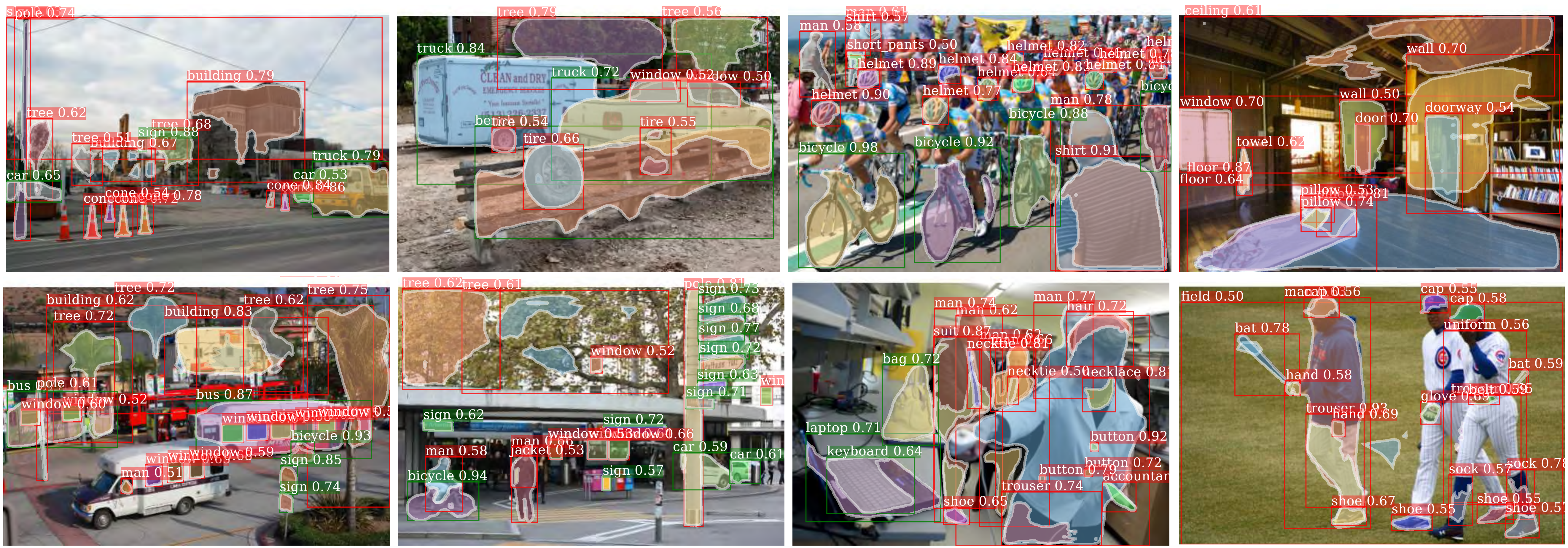
- Detection: Label per region, e.g., PASCAL VOC



[Krizhevsky et al 2011][Ren et al 2016]

Segmentation

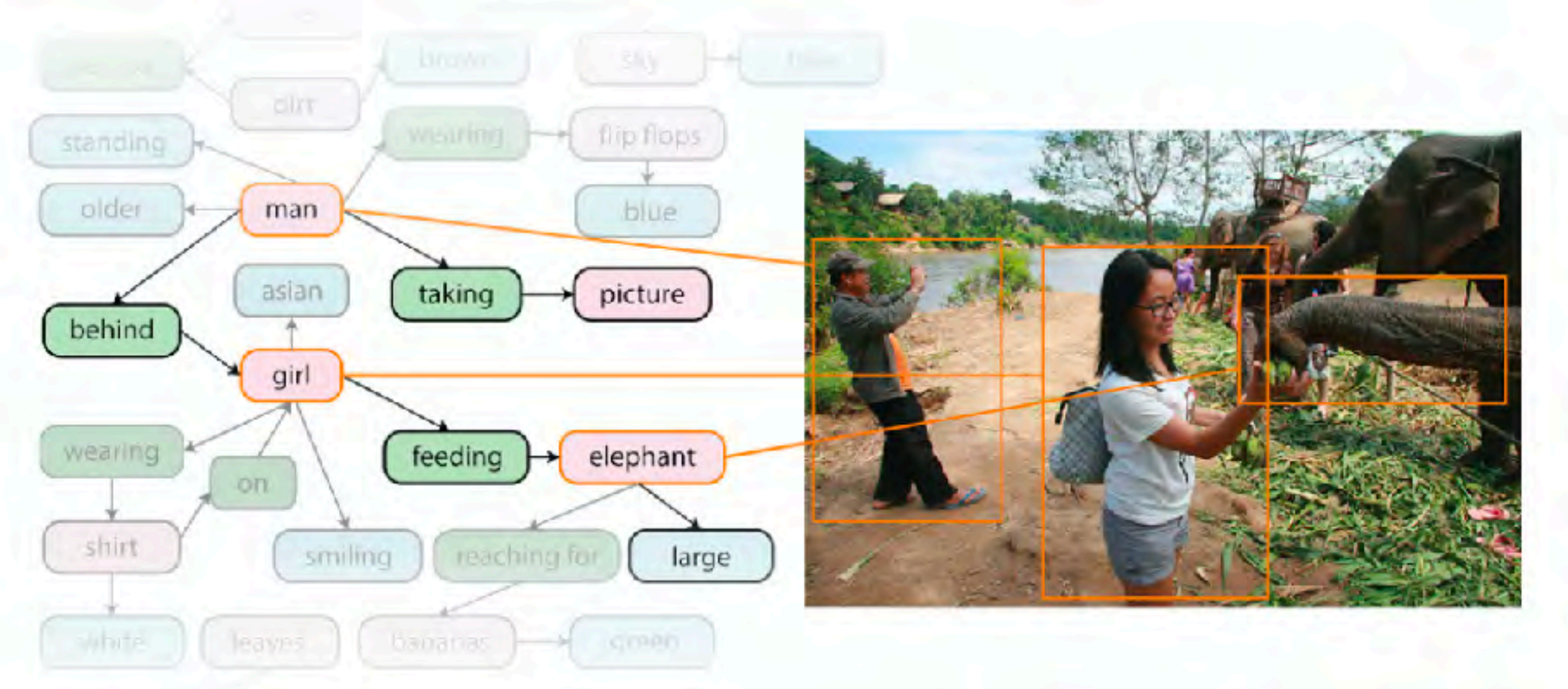
- Segmentation: Label per pixel, e.g., MS COCO



[Hu et al 2017]

Structured Image Understanding

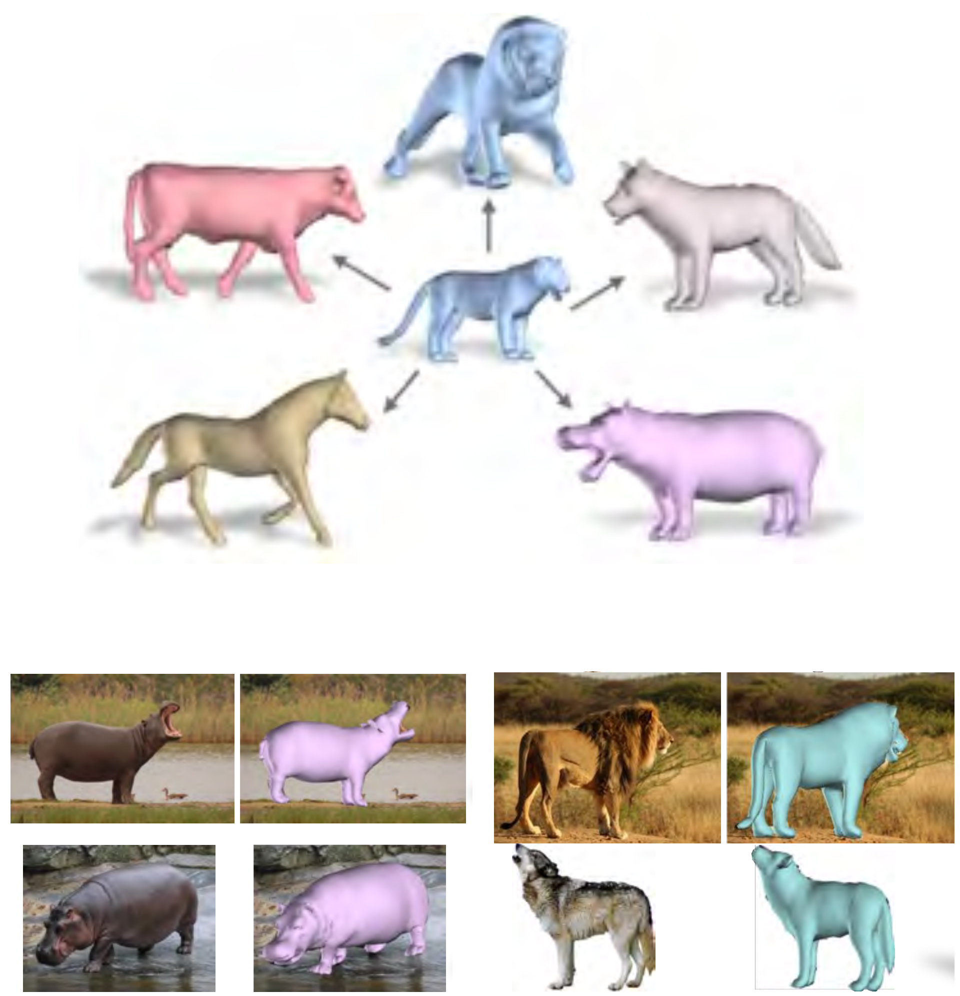
- "Girl feeding large elephant"
- "A man taking a picture behind girl"



visualgenome.org [Krishna et al 2017]

Shape + Tracking

- Other vision applications might need shape modelling (possibly deformable) and/or tracking in video



[Zuffi et al 2017]

[SMPL Loper et al 2015]

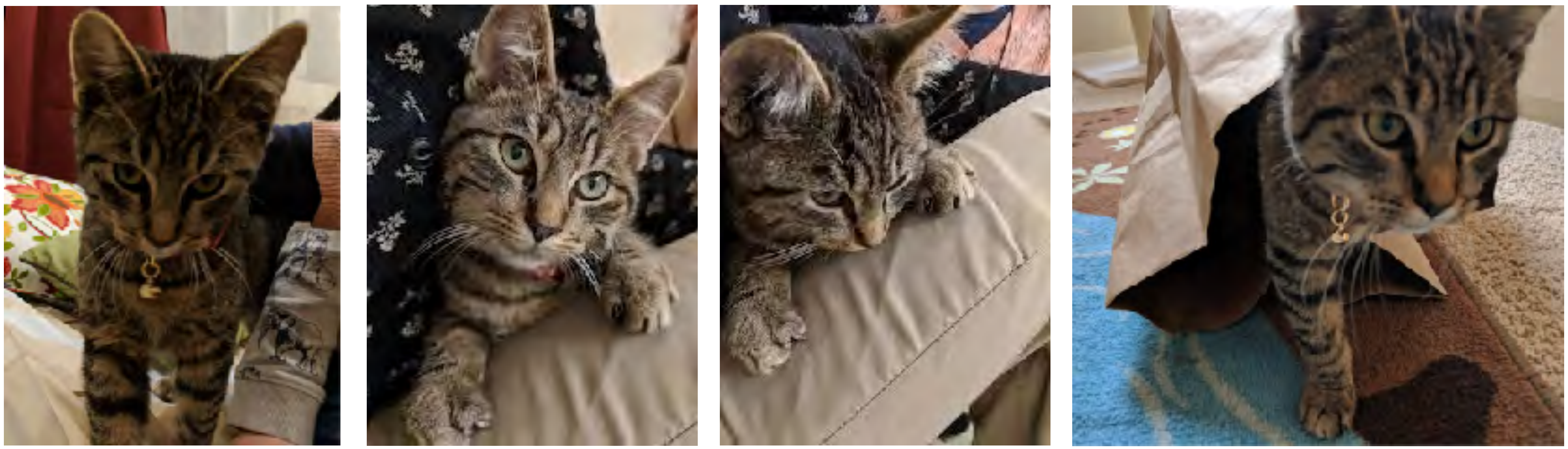
We'll focus on single image classification today

Classification: Instance vs Category

Instance of Aeroplane (Wright Flyer)

Category of Aeroplanes [Caltech 101]

Classification: Instance vs Category



Instance of a cat

Category of domestic cats

Taxonomy of Cats

↳ Mammals (Class Mammalia)

↳ Therians (Subclass Theria)

↳ Placental Mammals (Infraclass Placentalia)

↳ Ungulates, Carnivorans, and Allies (Superorder Laurasiatheria)

↳ Carnivorans (Order Carnivora)

↳ Felines (Family Felidae)

↳ Small Cats (Subfamily Felinae)

↳ Genus *Felis*

↳ Chinese Mountain Cat (*Felis bieti*)

↳ Domestic Cat (*Felis catus*)

↳ Jungle Cat (*Felis chaus*)

↳ African Wildcat (*Felis lybica*)

↳ Sand Cat (*Felis margarita*)

↳ Black-footed Cat (*Felis nigripes*)

↳ European Wildcat (*Felis silvestris*)

Bengal Tiger
[Omveer Choudhary]

Ocelot
[Jitze Couperus]

European Wildcat
[the wasp factory]

[inaturalist.org]

Taxonomy of Boats

[Deng et al 2009]

WordNet

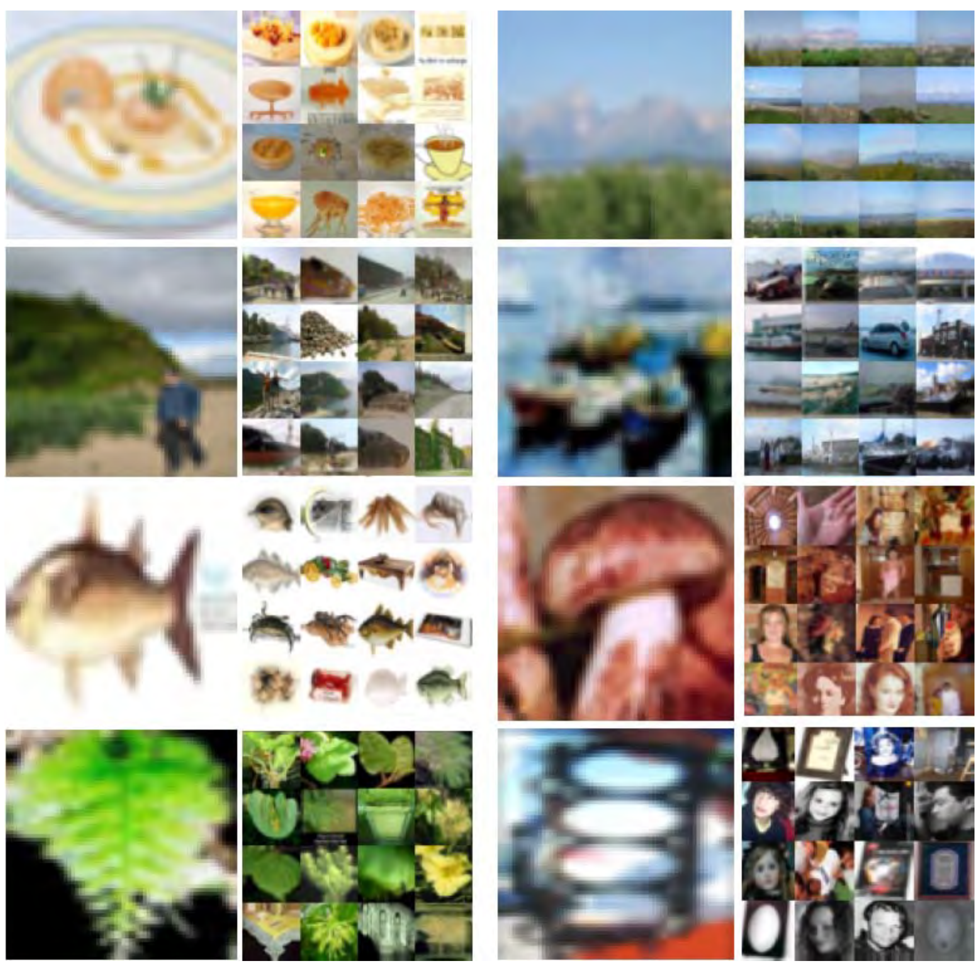
- We can use language to organise visual categories
- This is the approach taken in ImageNet [Deng et al 2009], which uses the WordNet lexical database [wordnet.princeton.edu]
- As in language, visual categories have complex relationships
- e.g., a "sail" is part of a "sailboat" which is a "watercraft"

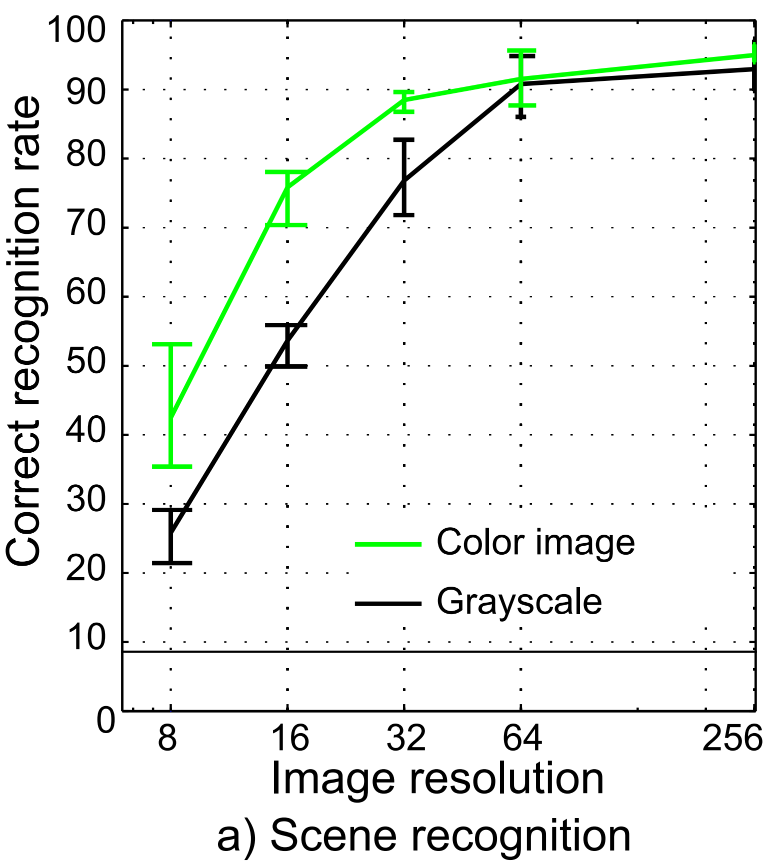
- S: (n) **sailboat**, sailing boat (a small sailing vessel; usually with a single mast)
  - *direct hyponym* / *full hyponym*
    - S: (n) catboat (a sailboat with a single mast set far forward)
    - S: (n) sharpie (a shallow-draft sailboat with a sharp prow, flat bottom, and triangular sail; formerly used along the northern Atlantic coast of the United States)
    - S: (n) trimaran (a fast sailboat with 3 parallel hulls)
  - *part meronym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) sailing vessel, sailing ship (a vessel that is powered by the wind; often having several masts)

If we call a "sailboat" a watercraft, is this wrong? What if we call it a "sail"?

Tiny Image Dataset

- Precursor to ImageNet and CIFAR10/100
- 80 million images collected via image search using 75,062 noun synsets from WordNet (labels are noisy)
- Very small images (32x32xRGB) used to minimise storage
- Note human performance is still quite good at this scale!

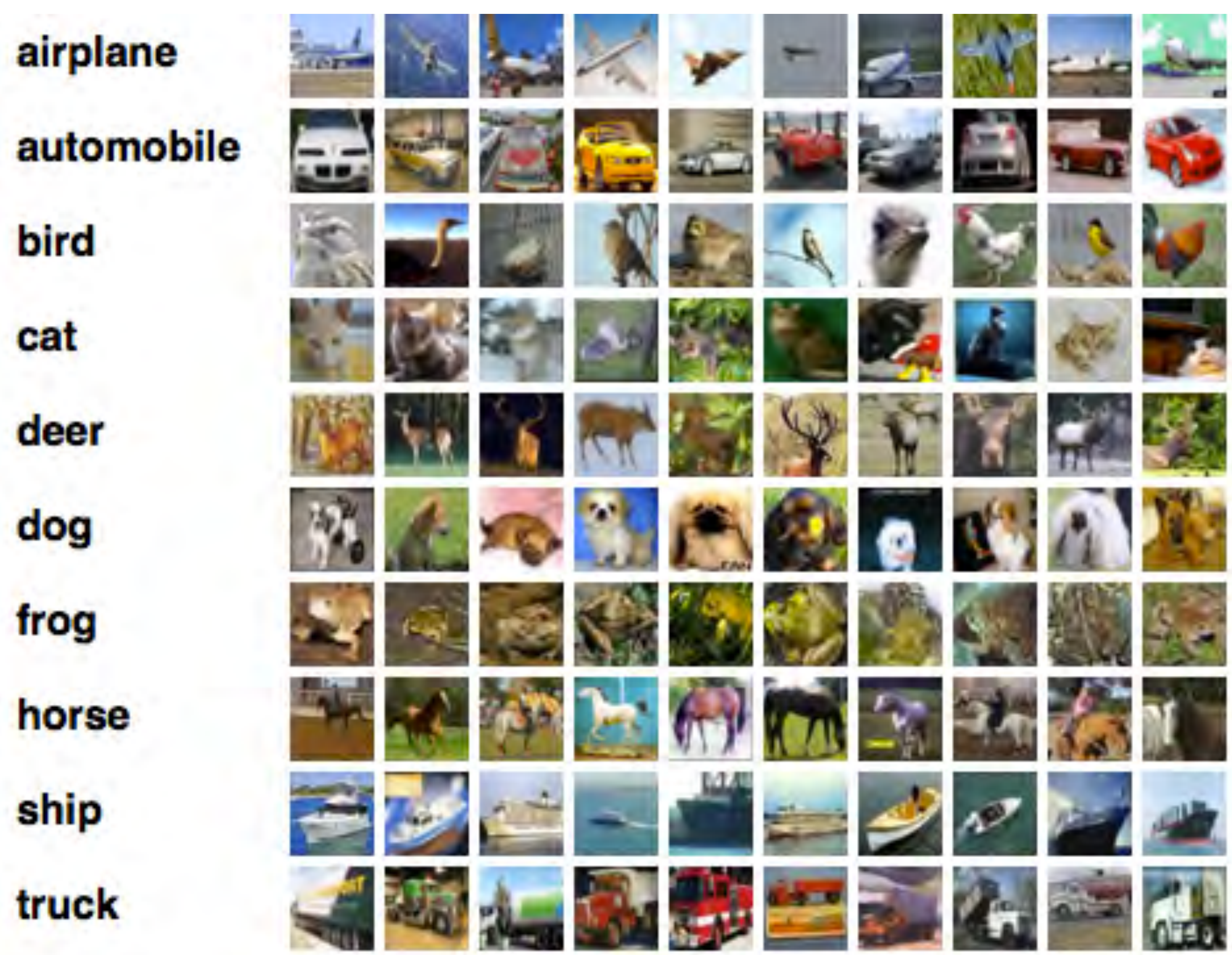




[Torralba Freeman Fergus 2008]

CIFAR10 Dataset

- Hand labelled set of 10 categories from Tiny Images dataset
- 60,000 32x32 images in 10 classes (50k train, 10k test)

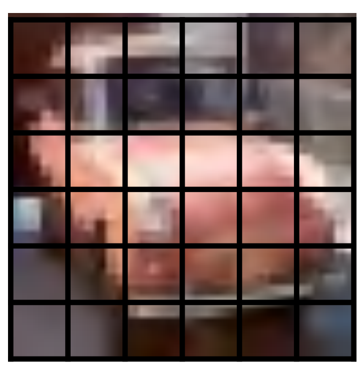


Good test set for visual recognition problems

CIFAR10 Classification

- Let's build an image classifier!

- Start by vectorizing the image data



32 x 32 x RGB (8 bit) image →

x = [65 102 33 57 54 …]

- x = 3072 element vector of 0-255
- Note this throws away spatial structure, we'll bring it back later when we look at feature extraction and CNNs

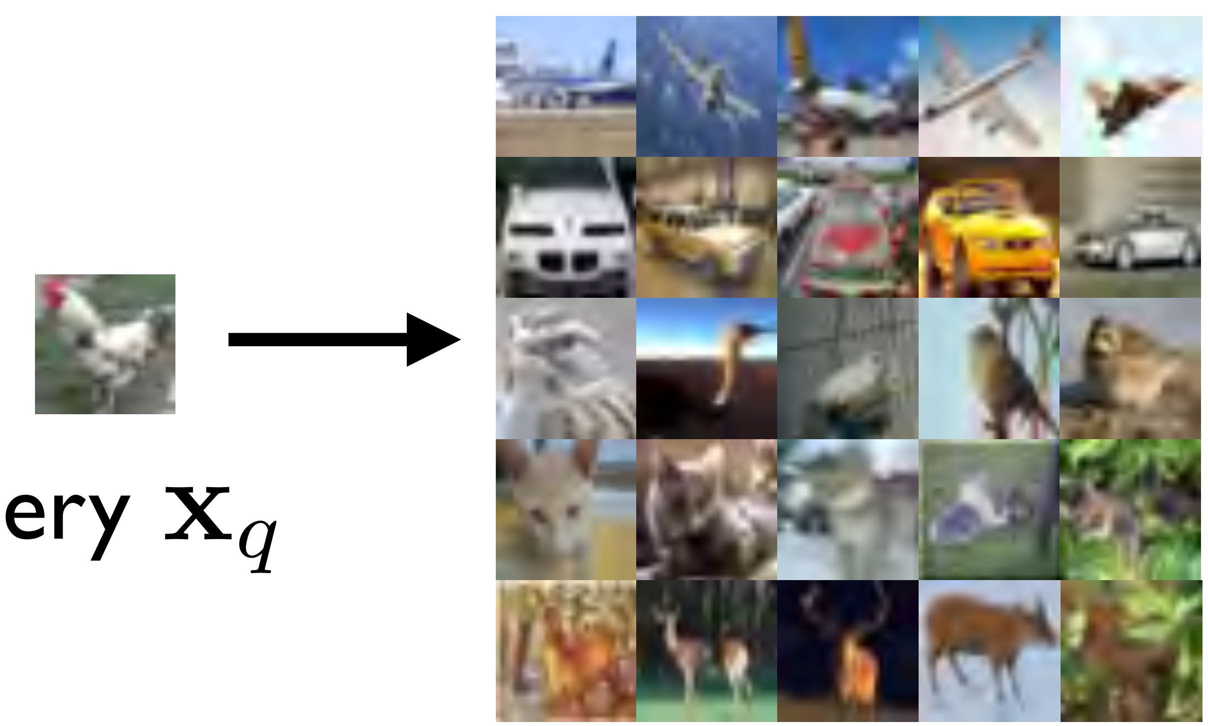
Nearest Neighbour Classification

- Find nearest neighbour in training set

$$i_{NN} = \arg\min_i |\mathbf{x}_q - \mathbf{x}_i|$$

- Assign class to class of the nearest neighbour

$$\hat{y}(\mathbf{x}_q) = y(\mathbf{x}_{i_{NN}})$$



1
2
3
4
5

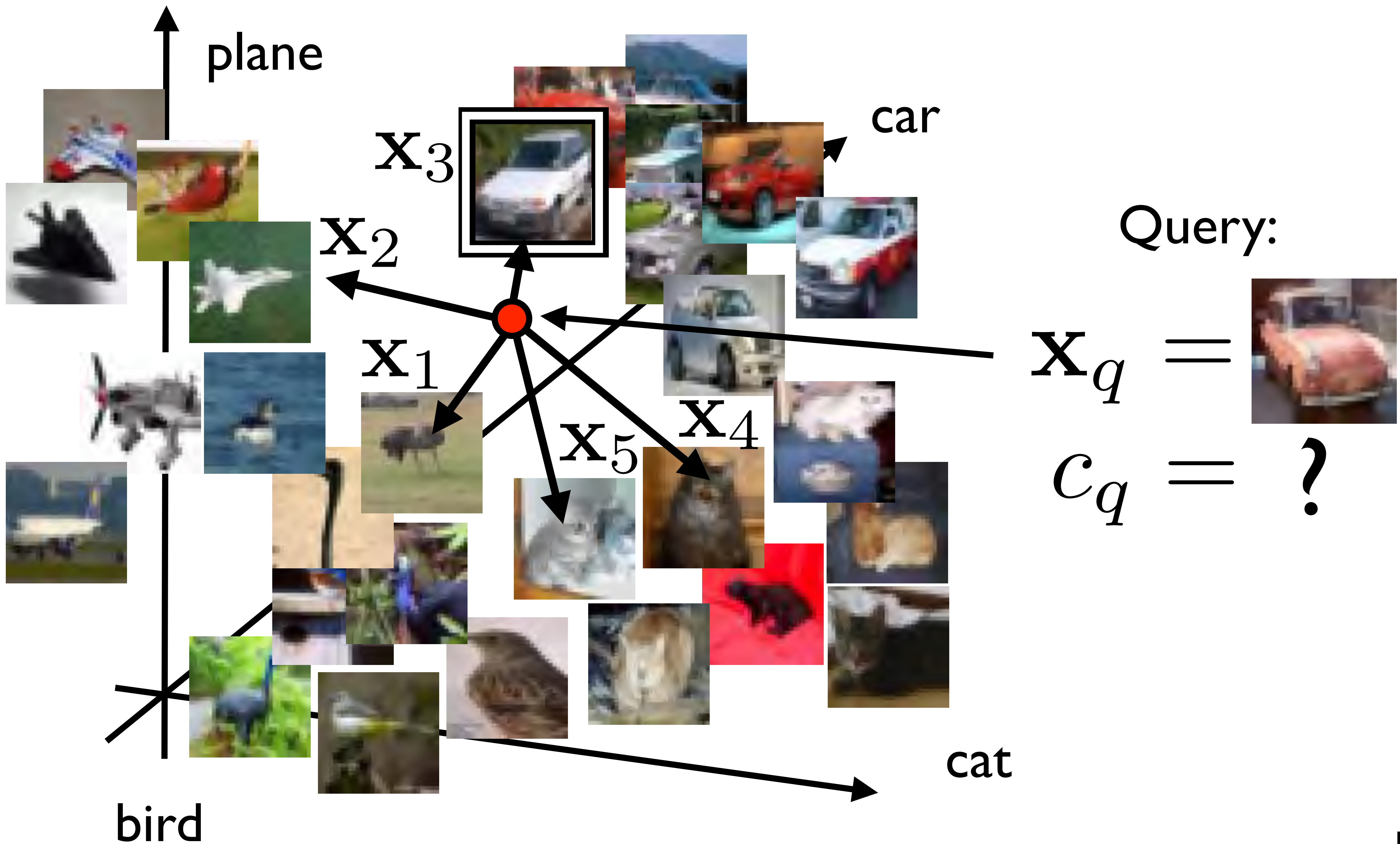
Result = 3

Query $\mathbf{x}_q$

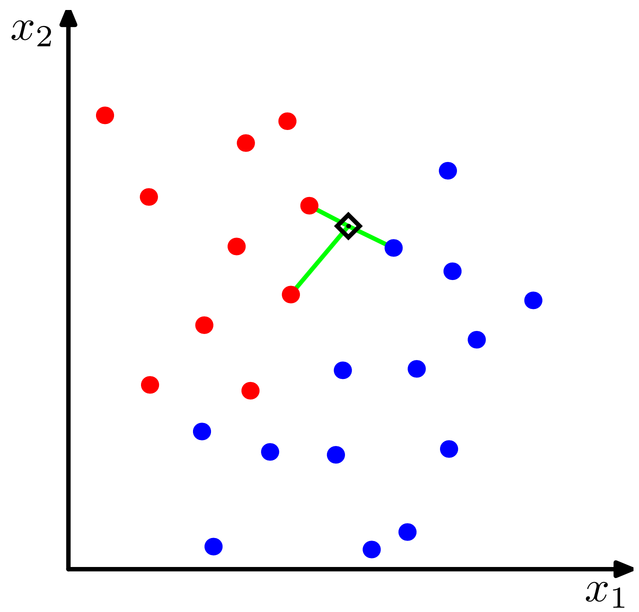
Calculate $|\mathbf{x}_q - \mathbf{x}_i|$ for all training data

Nearest Neighbour Classification

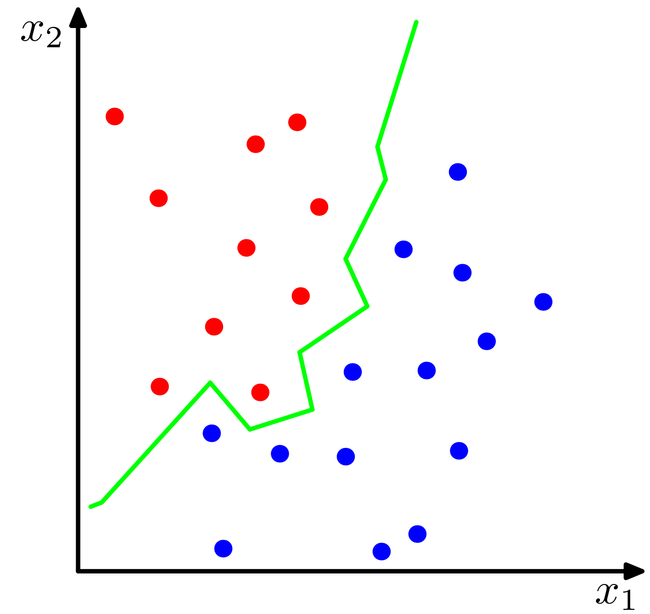
- We can view each image as a point in a high dimensional space



Nearest Neighbour Classifier



(a)

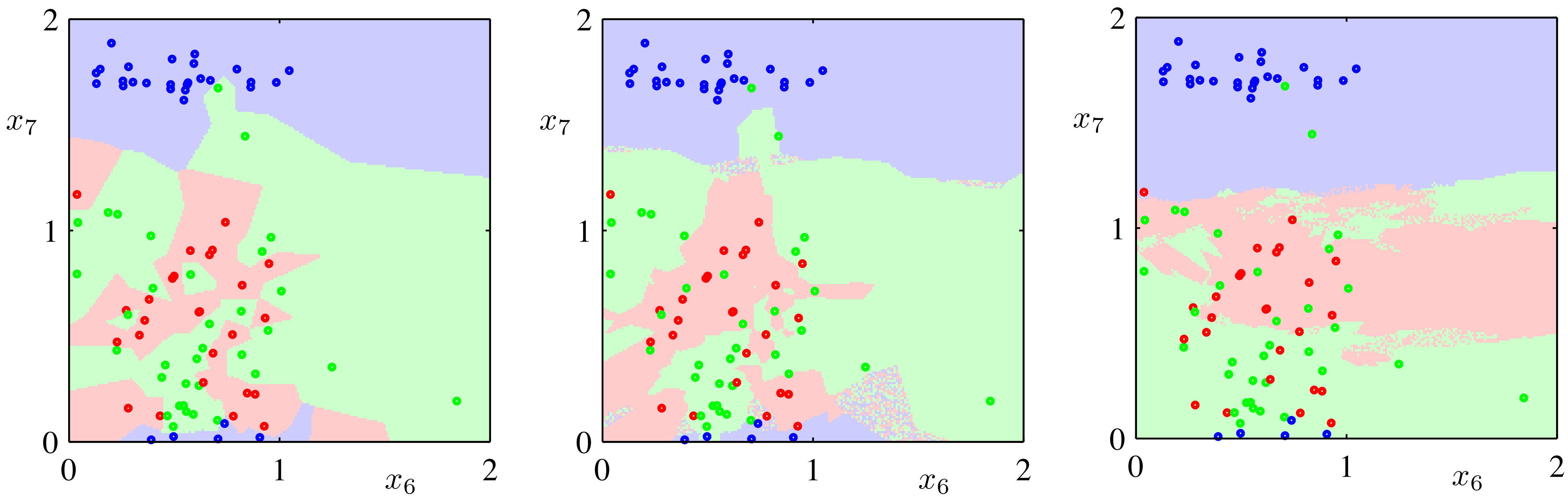


(b)

- What is the decision boundary for a nearest-neighbour classifier?

k-NN Classifier

- Identify k nearest neighbours of the query
- Assign class as most common class in set
- k-NN decision boundaries:



k = 1 k = 3 k = 31

Good performance depends on suitable choice of k

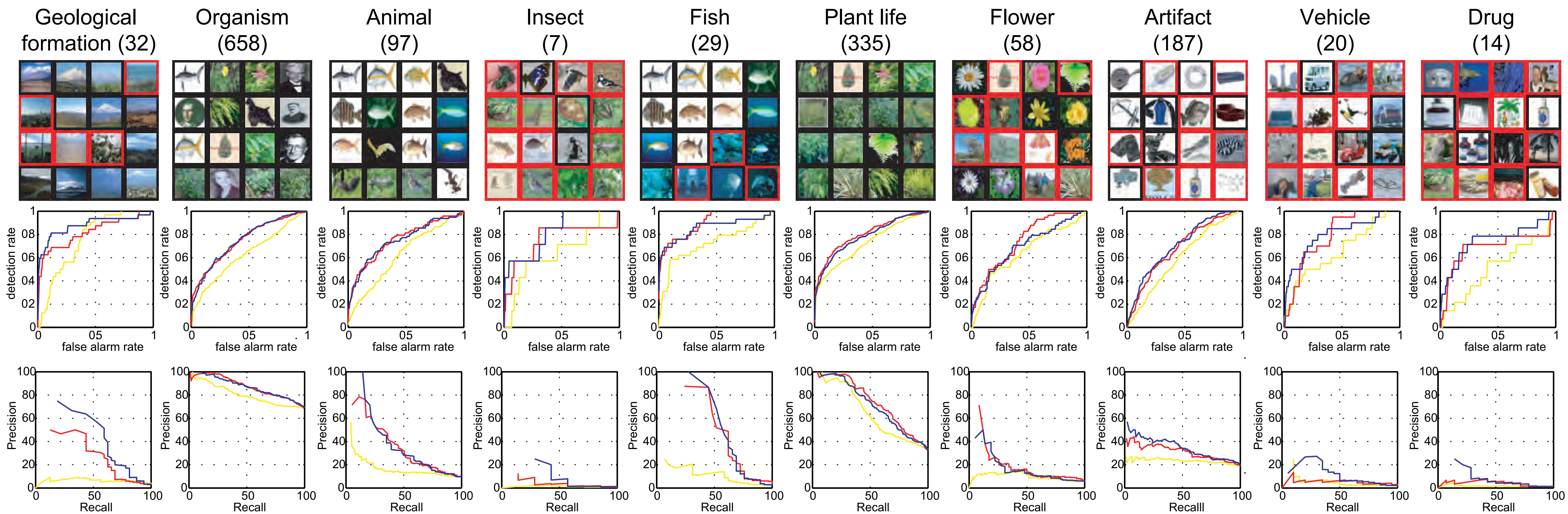
What do nearest neighbours look like with 80 million images?

[Torralba, Fergus, Freeman '08]

Query

Tiny Image Recognition

- Recognition performance (categories vary in semantic level)

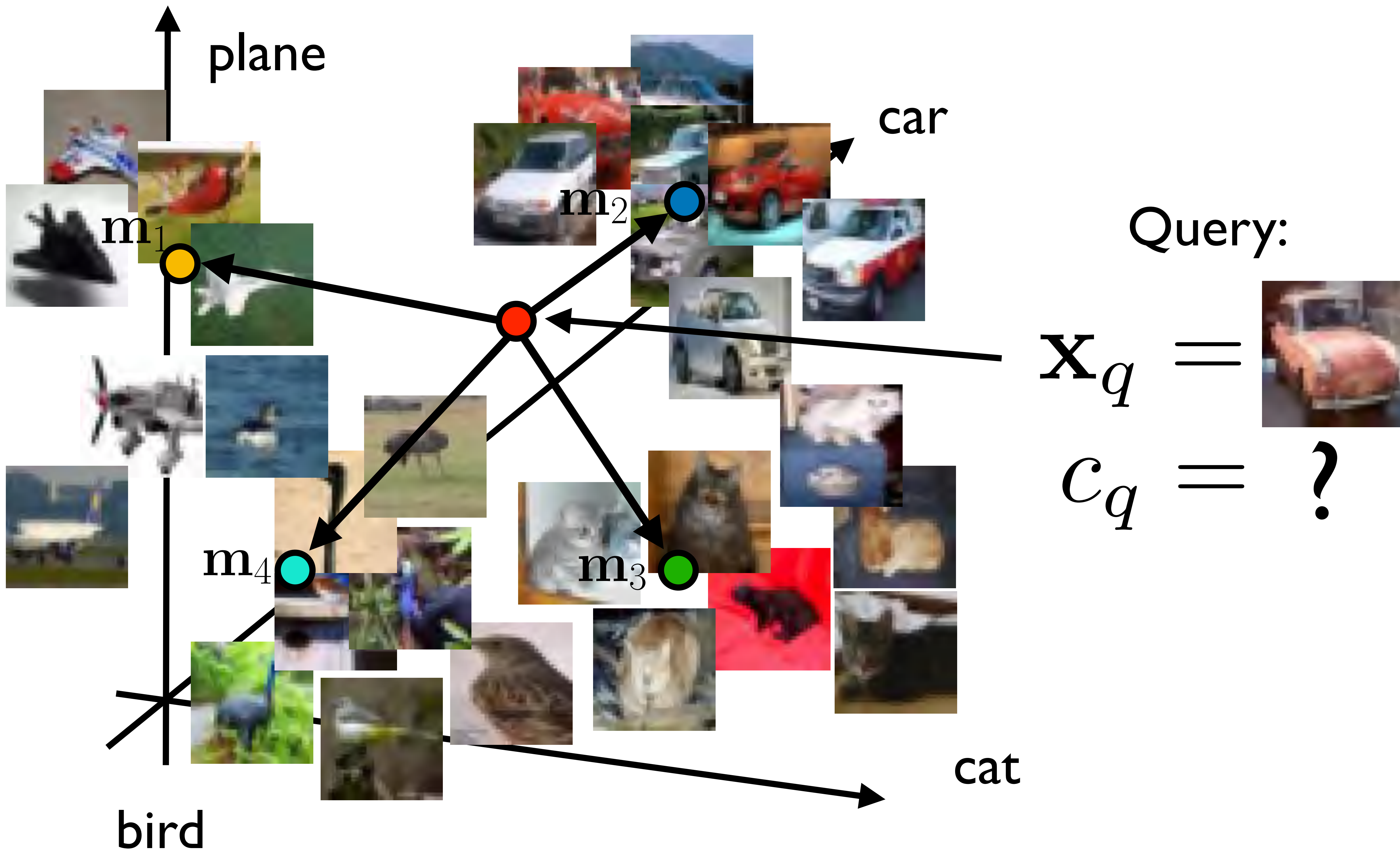


yellow = 7900, red = 790,000, blue = 79,000,000

Nearest neighbour becomes increasingly accurate as N increases, but do we need to store a dataset of 80 million images?

Nearest Mean Classification

- How about a single template per class



Nearest Mean Classification

- Find nearest mean and assign class

$$c_q = \arg\min_i |\mathbf{x}_q - \mathbf{m}_i|^2$$

- CIFAR 10 class means

- Can we do better?
- What is the best template for L2 matching?

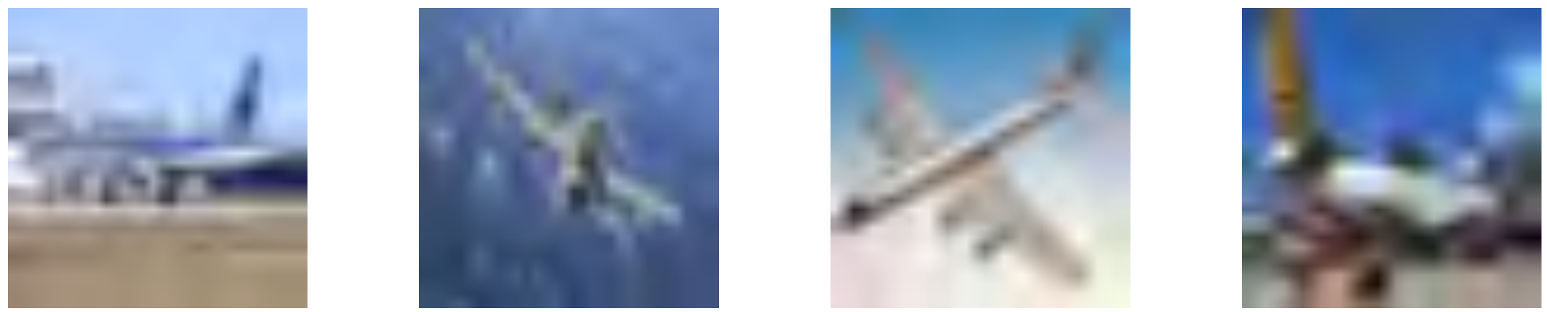
Linear Classification

- Linear classification, 2-class, N-class
- Regularization, softmax, cross entropy
- SGD, learning rate, momentum

Linear Classification

- Let’s start by using 2 classes, e.g., bird and plane
- Apply labels (y) to training set:

y = +1

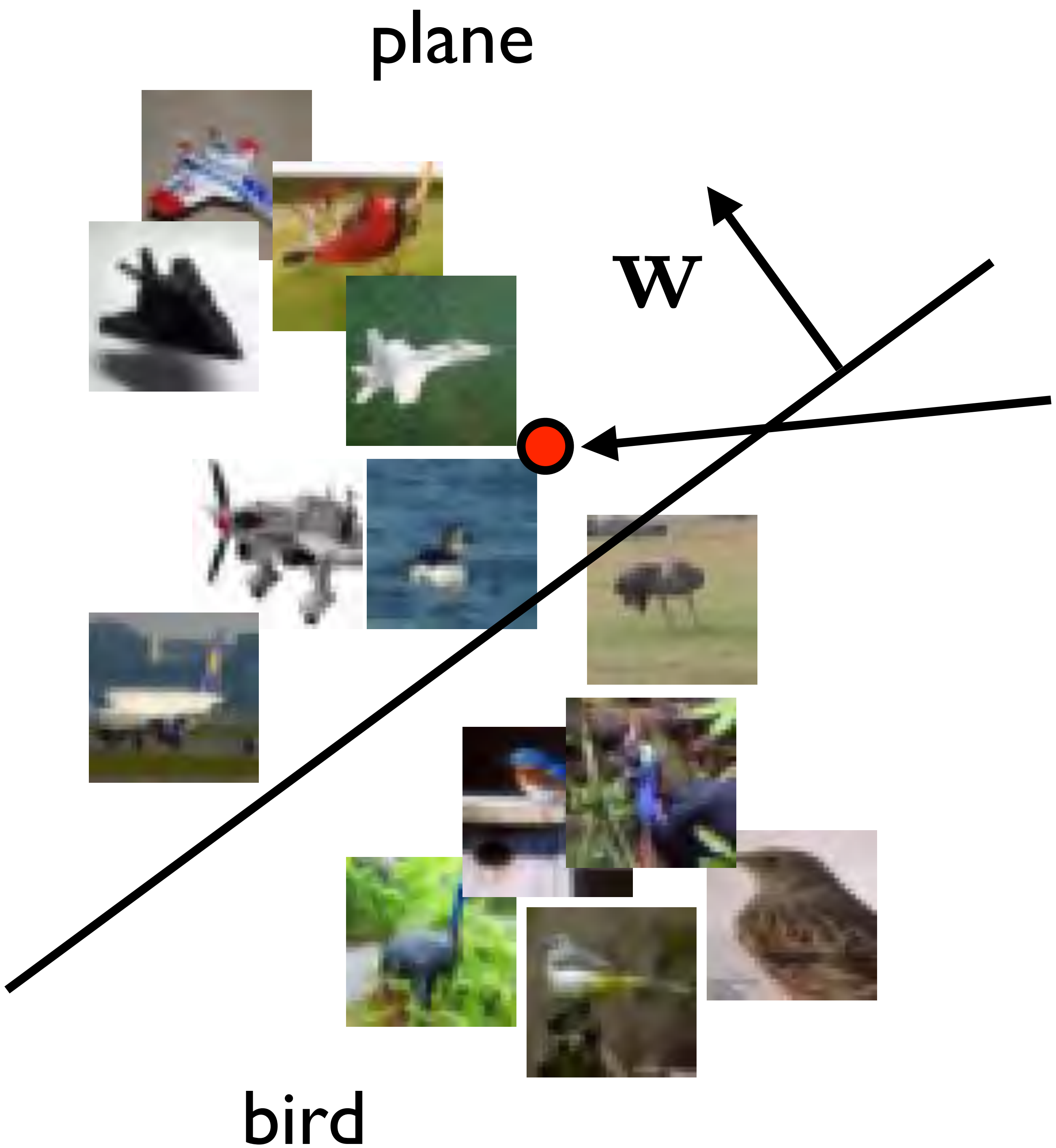


y = -1

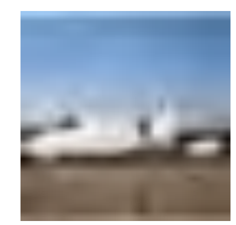
- Use a linear model to regress y from x

2-class Linear Classification

- Separating hyperplane, projection to a line defined by **w**



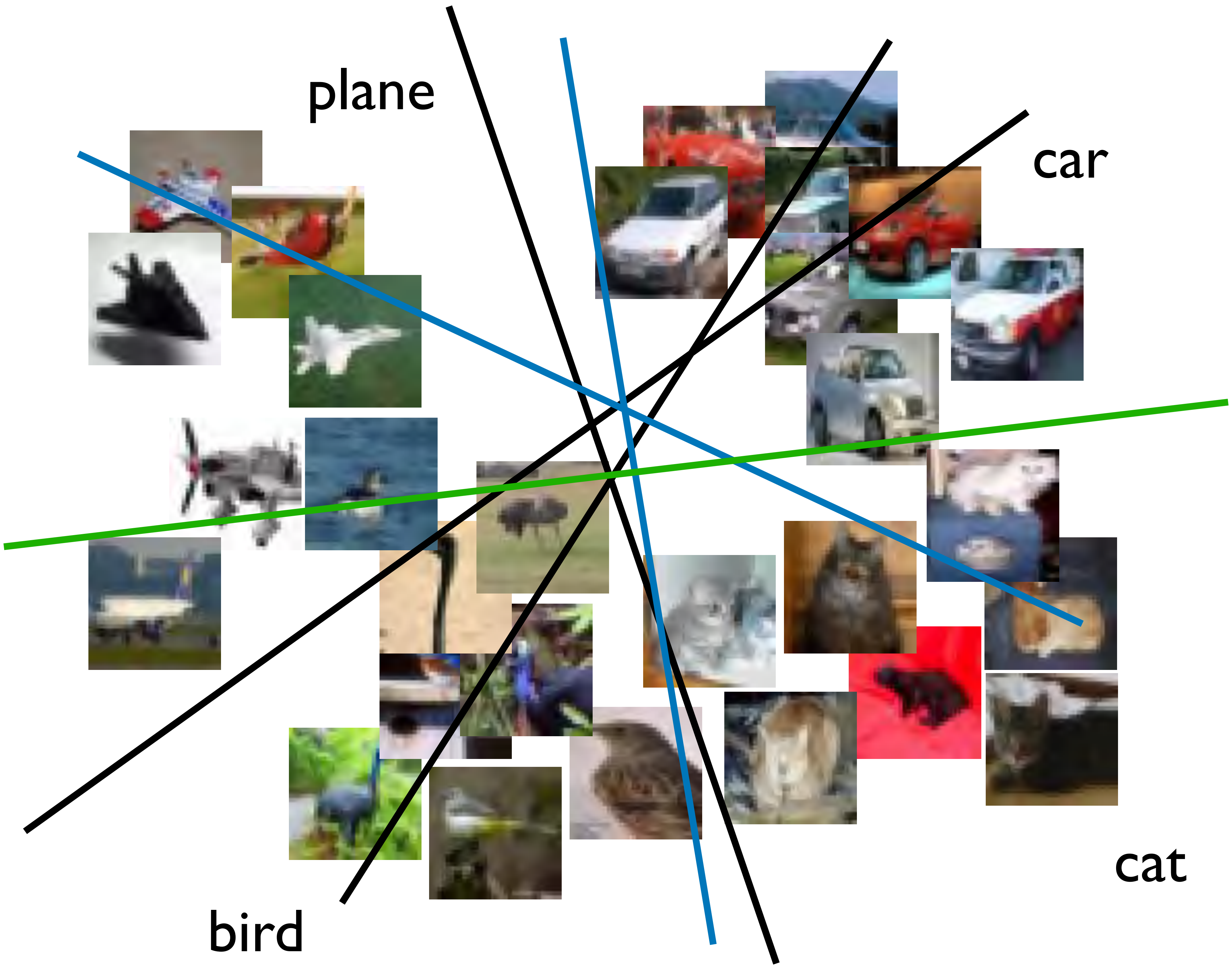
Query:

$\mathbf{x}_q =$ 

$$\hat{y} = \text{sign } h = \text{sign } \mathbf{w}^T \mathbf{x}_q$$

N-class Linear Classification

- We could construct $O(n^2)$ 1 vs 1 classifiers



N-class Linear Classification

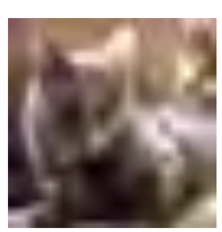
- We could regress directly to integer class id, y = {0,1,2,3…9}

One-Hot Regression

- A better solution is to regress to one-hot targets = 1 vs all classifiers

$$\begin{bmatrix} & & \\ & \mathbf{W}^T & \\ & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \cdots \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \cdots \end{bmatrix}$$

class 2 = 'automobile'



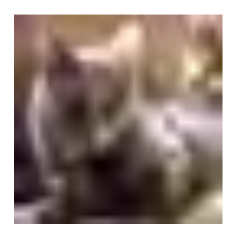
$$\begin{bmatrix} & & \\ & \mathbf{W}^T & \\ & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \cdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \cdots \end{bmatrix}$$

class 4 = 'cat'

One-Hot Regression

- Stack into matrix form

$$\begin{bmatrix} & \mathbf{W}^T & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \dots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \dots \end{bmatrix} \dots = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \dots \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \dots \end{bmatrix} \dots$$

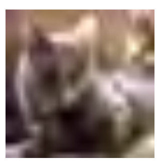



class 2 = 'automobile'

class 4 = 'cat'

One-Hot Regression

- Notation changed to transposed matrix/vector

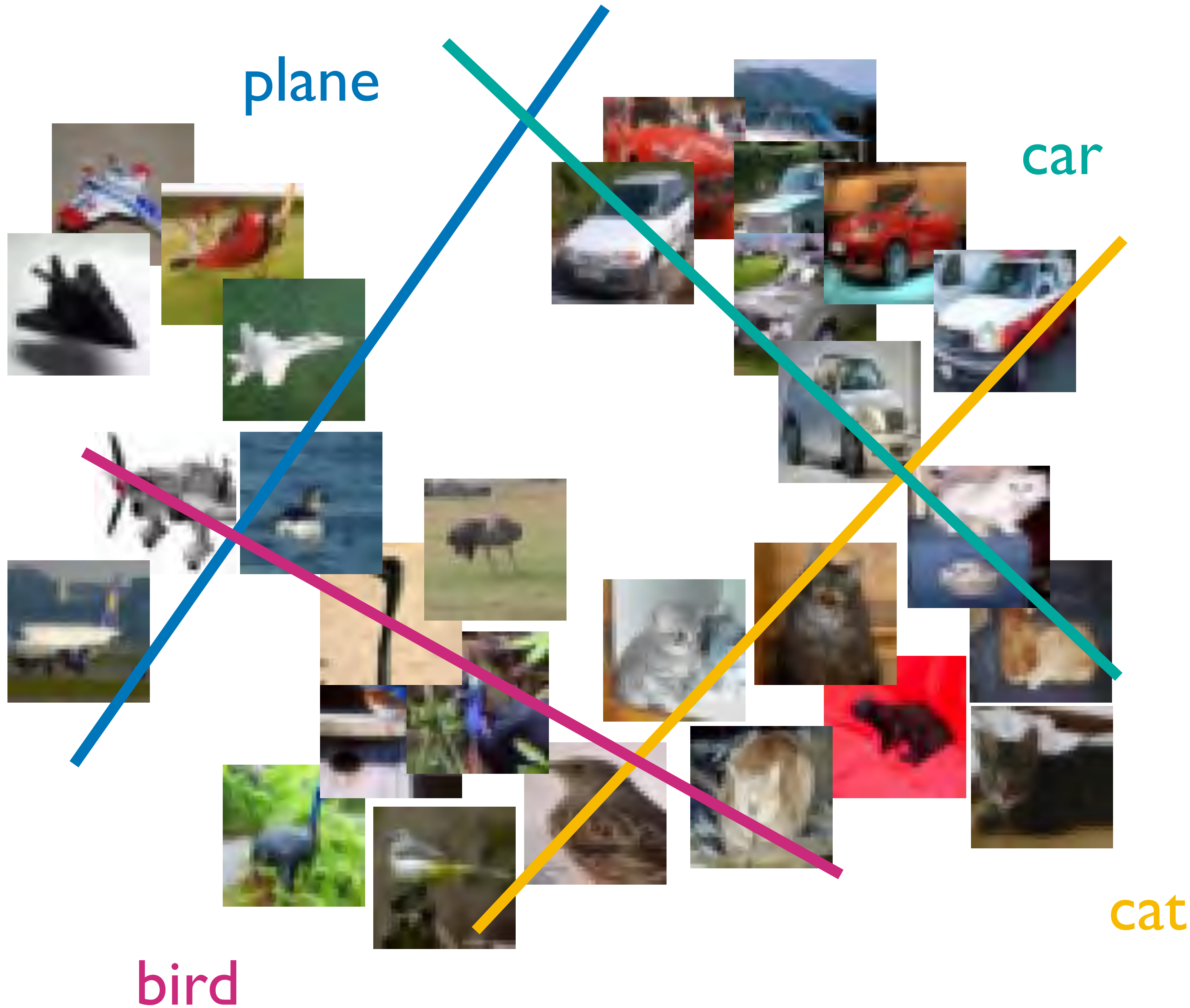
$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots \\ x_{21} & x_{22} & x_{23} & \dots \\ x_{31} & x_{32} & x_{33} & \dots \\ & \dots & & \end{bmatrix} \begin{bmatrix} & & \\ & \mathbf{W} & \\ & & \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ & .. & .. & & \end{bmatrix} \begin{array}{l} \text{auto} \\ \text{cat} \\ \end{array}$$

$$\mathbf{XW} = \mathbf{T}$$

- Solve regression problem by Least Squares

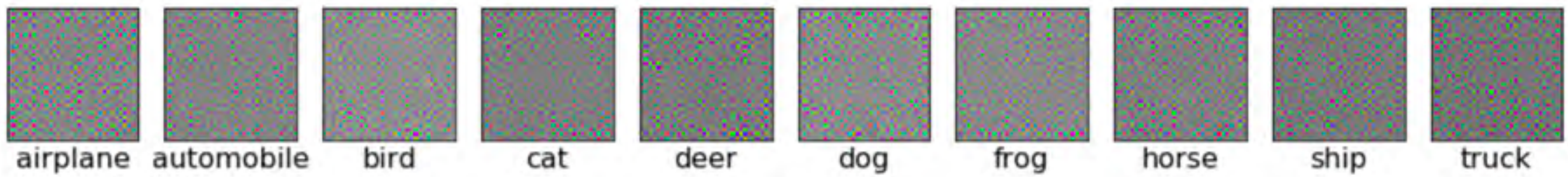
N-class Linear Classification

- One hot regression = 1 vs all classifiers



One-Hot Regression

- Visualise class templates for the least squares solution

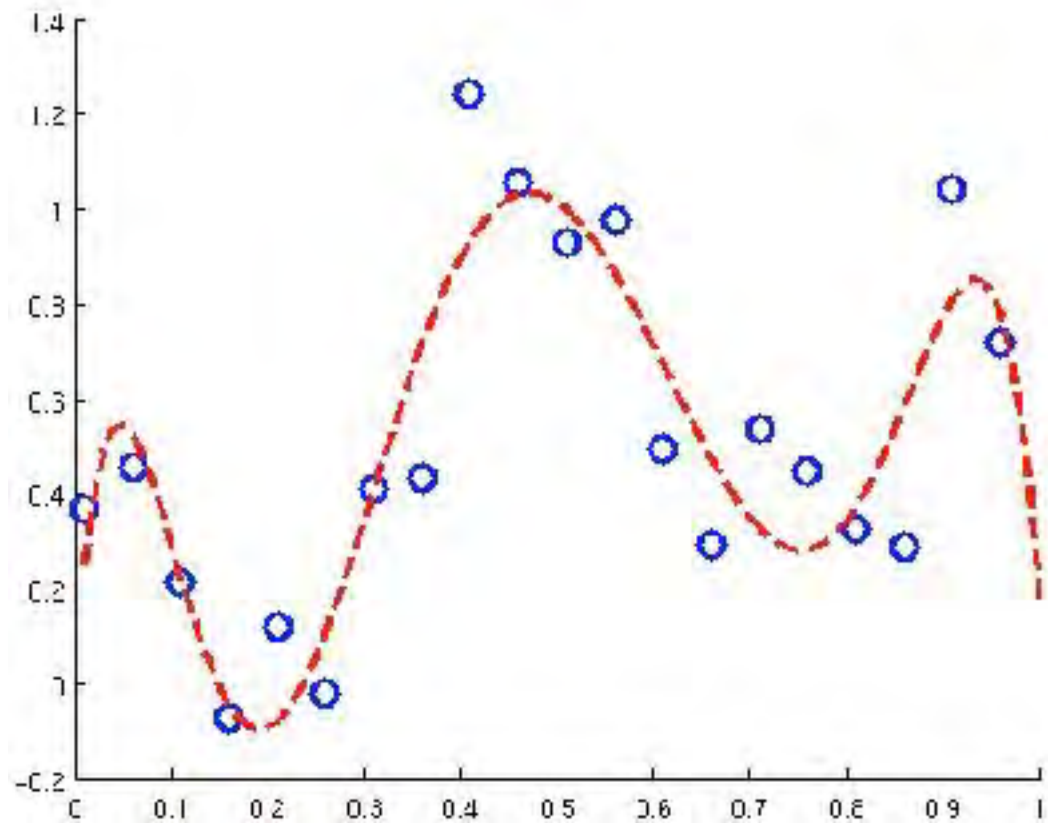


- Classifier accuracy = 35% (not bad, c.f., nearest mean = 27%)

What is happening here?

Polynomial Fitting

- Consider fitting a polynomial to some data by linear regression



Polynomial Fitting

- Multiple data points (y_i, x_i)

$$y_1 = a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3$$
$$y_2 = a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3$$
$$y_3 = a_0 + a_1 x_3 + a_2 x_3^2 + a_3 x_3^3$$
$$\ldots$$

- In matrix form

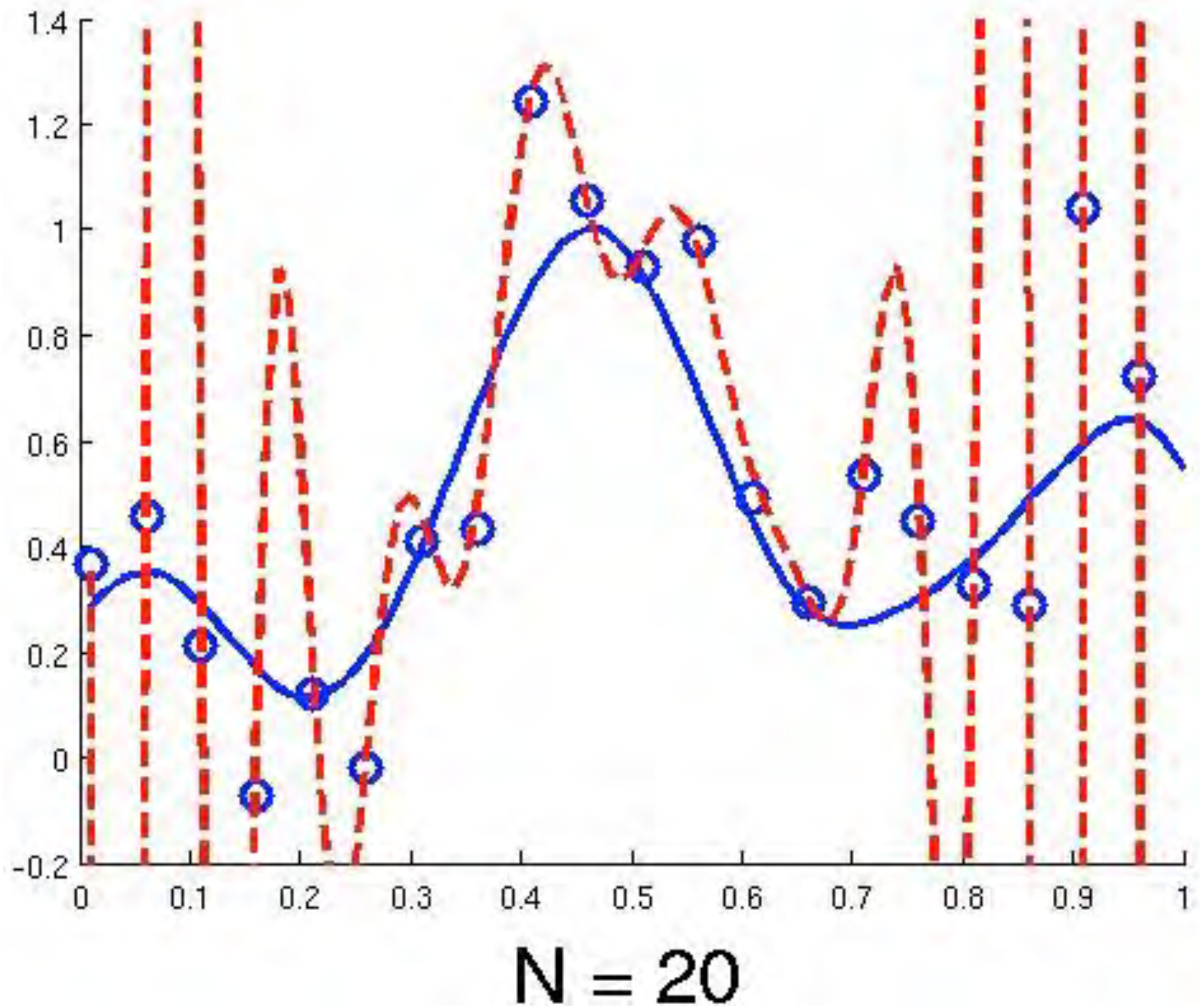
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \ldots \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ & \ldots & \ldots & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$$\mathbf{y} = \mathbf{M}\mathbf{a}$$

- Solve linear system by Gaussian elimination (if square) or Least Squares (if overconstrained)

Polynomial Fitting

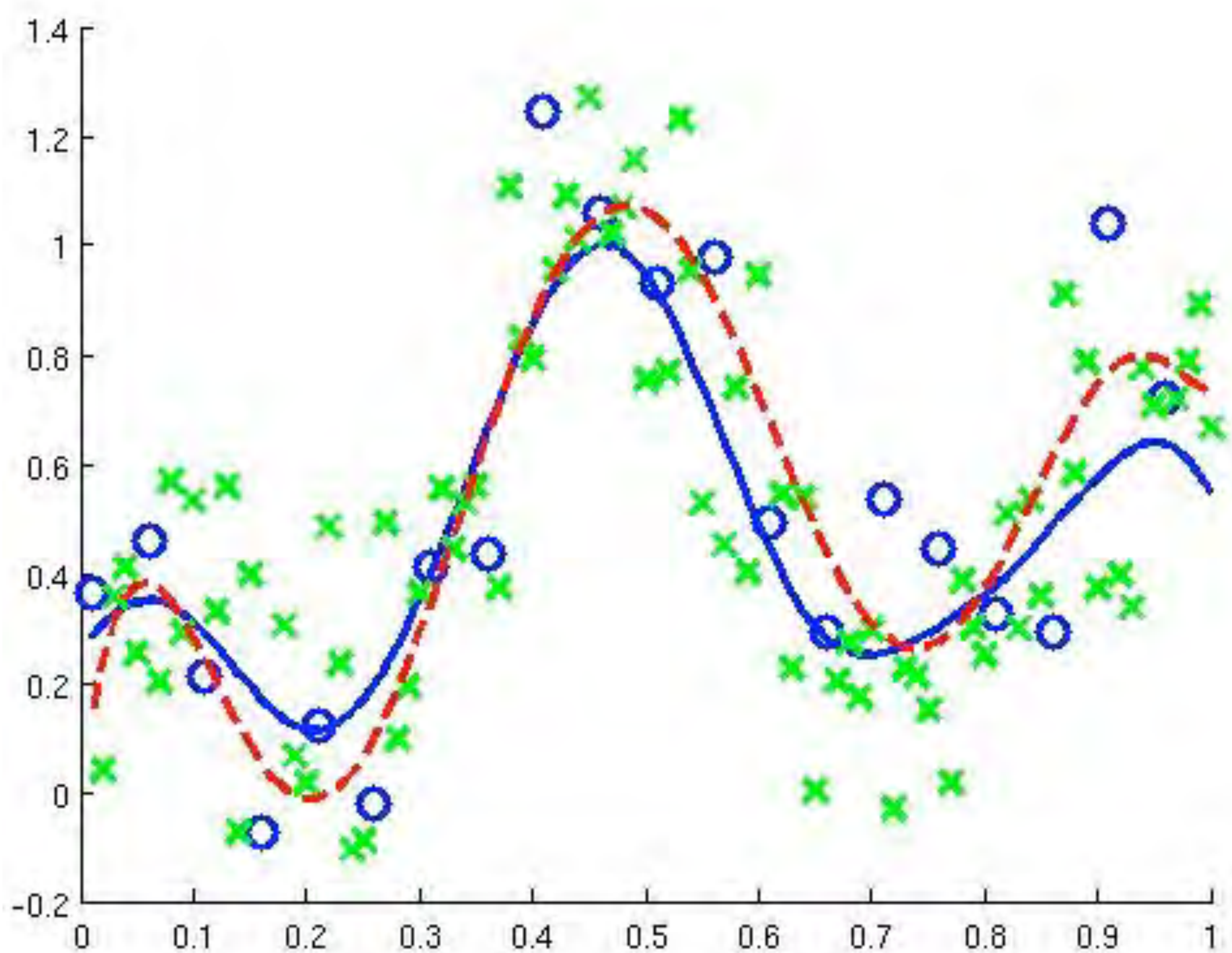
- Fit Nth order polynomial by least squares



- Overfitting

Cross Validation

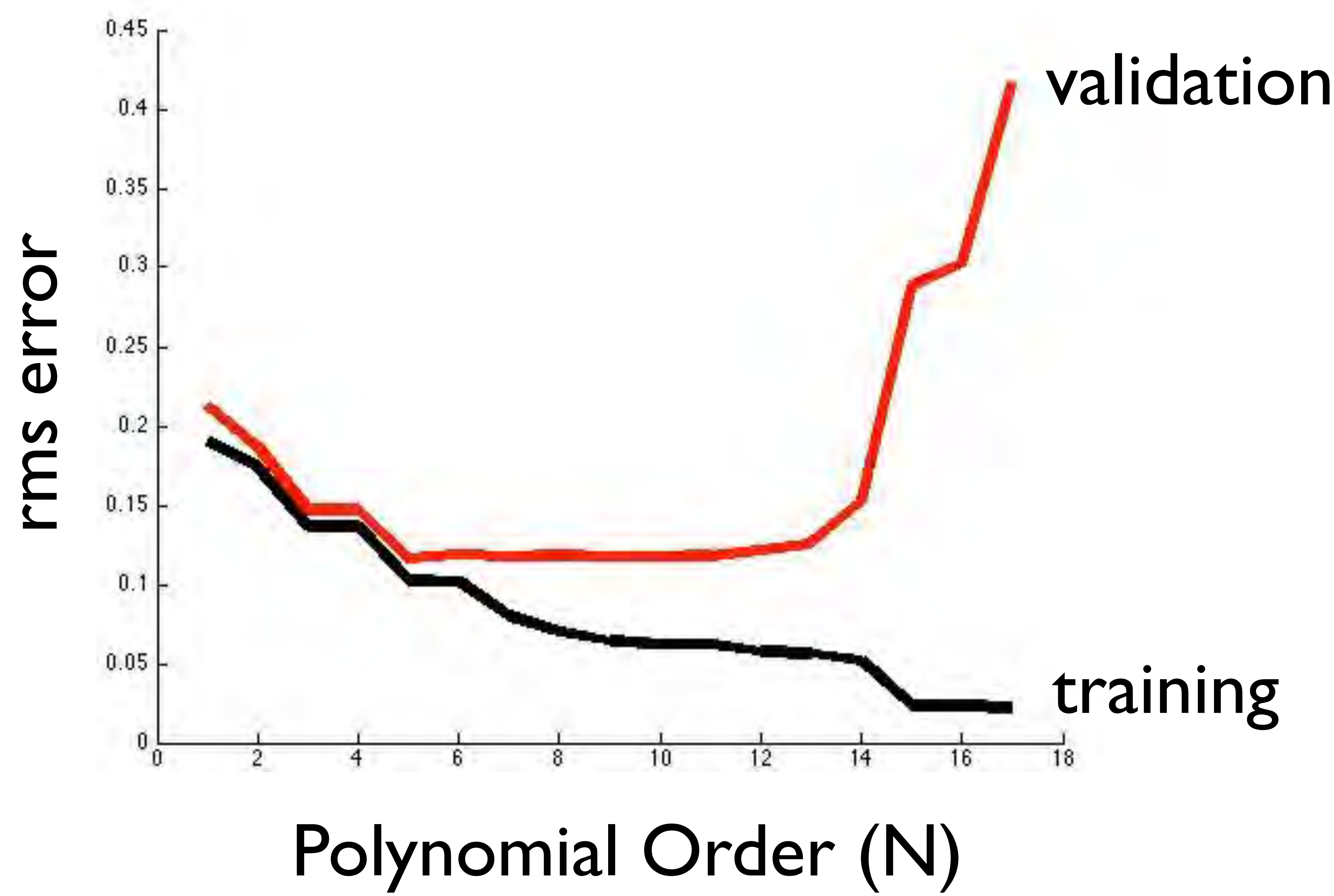
- Fit the model to a subset of data, and evaluate the fit on a held out **validation set**



- Calculate rms error $e_{rms} = \left(\frac{1}{N} \sum_i (y_i - \hat{y}_i)^2 \right)^{\frac{1}{2}}$

Cross Validation

- Training error always decreases, but validation error has a minimum for the best model order



Polynomial Fitting

- For large N, coefficients become HUGE!

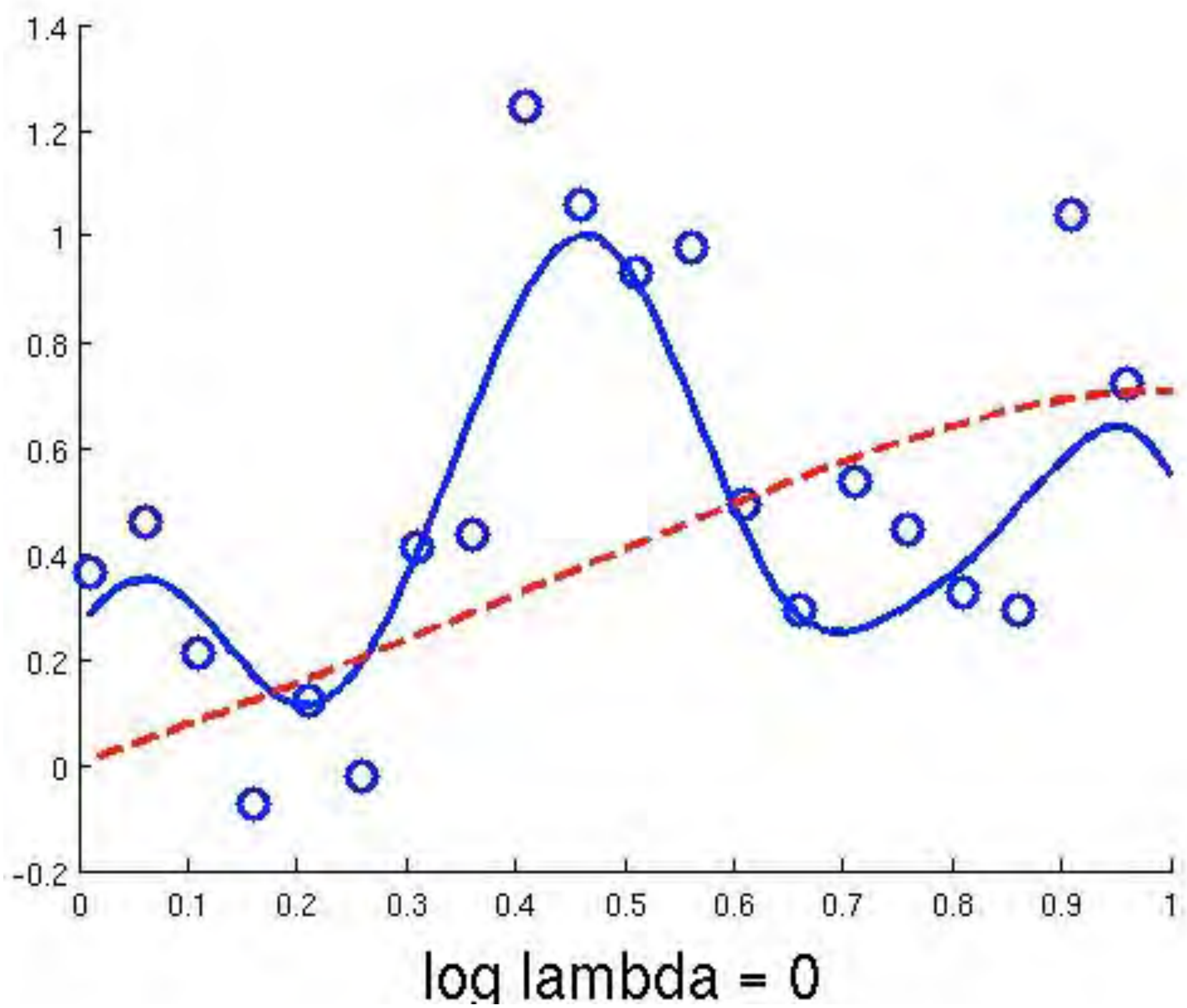
	N=1	N=2	N=4	N=10
a_0	0.90	2.03	-2.88	48.50
a_1		-1.54	29.76	-1294.90
a_2			-57.43	14891.41
a_3			31.86	-95161.10
a_4				367736.84
a_5				-885436.68
a_6				1331063.41
a_7				-1212056.89
a_8				610930.32
a_9				-130727.39

Regularization

- L2 penalty on polynomial coefficients

Regularized Linear Regression

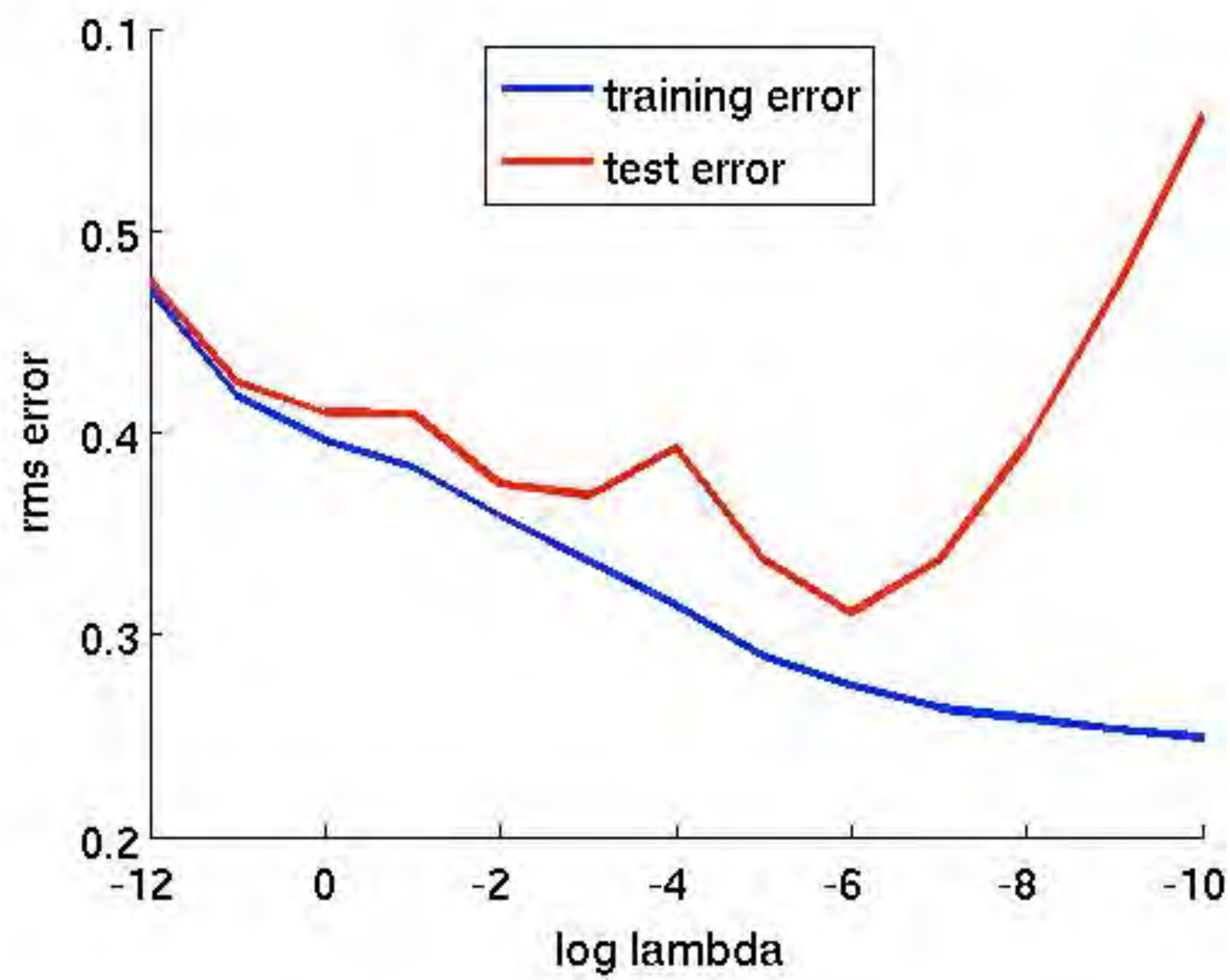
- 10th order polynomial, prior on the coefficients weight λ



- Over-smoothing...

Under/Overfitting

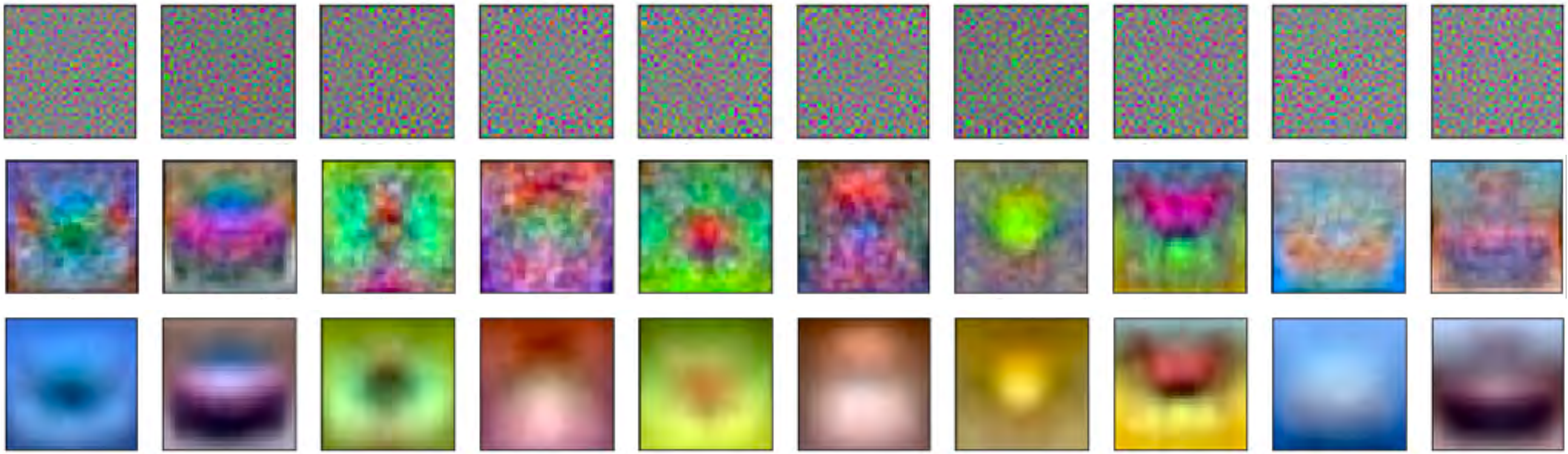
- Test error vs lambda



- Training error always decreases as lambda is reduced
- Test error reaches a minimum, then increases ⇒ overfitting

Regularized Classification

- Add regularization to CIFAR10 linear classifier



- Row 1 = overfitting, Row 3 = oversmoothing?

Non-Linear Optimisation

- With a linear predictor and L2 loss, we have a closed form solution for model weights W
- How about this (non-linear) function

$$\mathbf{h} = \mathbf{W}_2 \max(0, \mathbf{W}_1 \mathbf{x})$$

- Previously (e.g., bundle adjustment), we locally linearised the error function and iteratively solved linear problems

$$e = \sum_i |\mathbf{h}_i - \mathbf{t}_i|^2 \approx |\mathbf{J}\Delta\mathbf{W} + \mathbf{r}|^2$$

$$\Delta\mathbf{W} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}$$

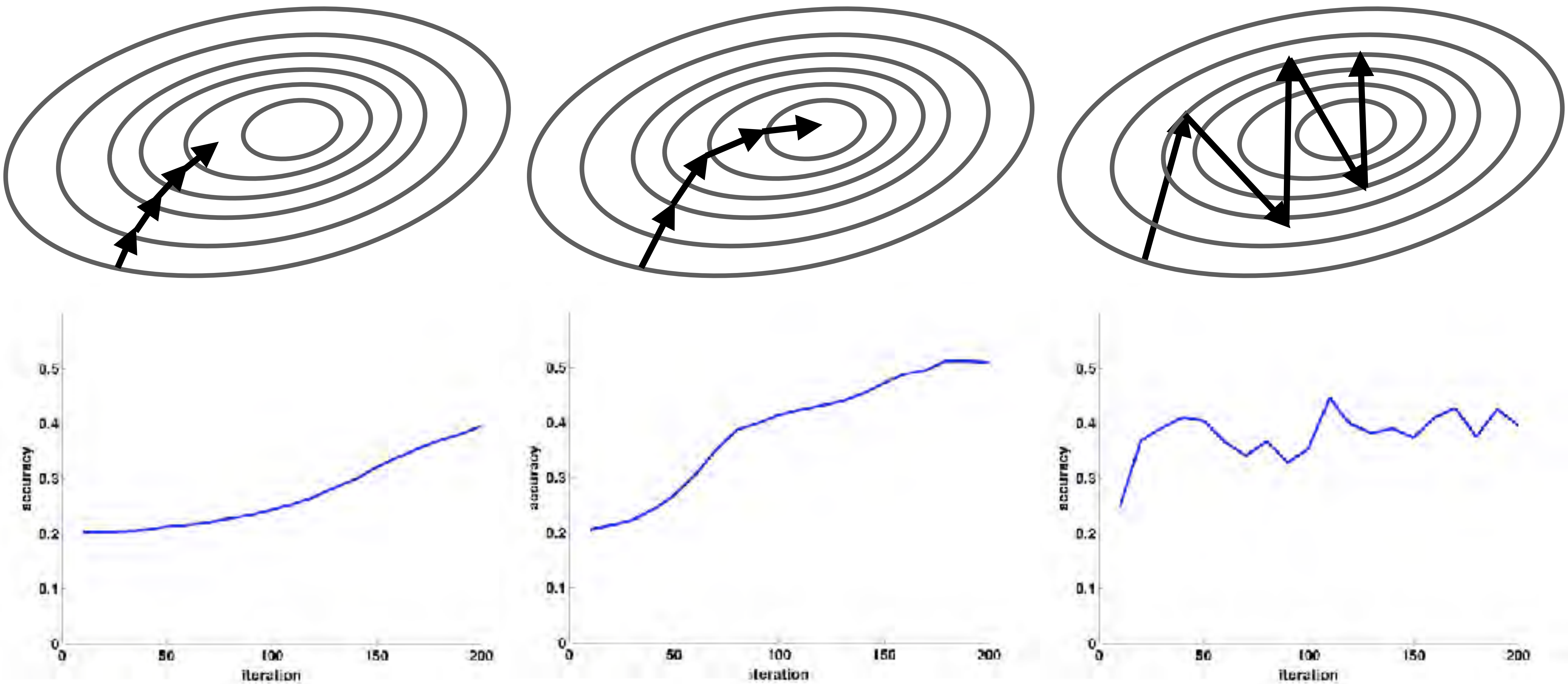
Does this look like a promising approach?

Gradient Descent

- Let's try 1st order optimization instead
- Even though we can solve our Linear L2 model in closed form, we'll try it out with gradient descent
- In stochastic gradient descent (SGD), we select a random batch of data, compute the gradient, and take a step
- L2 loss for a single example x

Learning Rate

- Controls the size of the gradient descent step



$\alpha = 0.02$ $\alpha = 0.05$ $\alpha = 0.2$

Too slow Too fast

Loss and Activation Functions

Softmax + Logistic Outputs

- Linear regression to one-hot targets is a bit strange..
- Output could be very large, and scores >>1 are penalised even for the correct class, ditto scores << 1 for incorrect
- How about restricting output scores to 0-1?

Softmax + Cross Entropy

- What is the gradient of the softmax linear classifier?
- We could use L2 loss, but we'll use cross entropy instead
- This has a sound motivation — it is a measure of the difference between probability distributions
- It also leads to a simple update rule

Linear + Softmax Regression

- We found the following gradient descent update rule

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha(\underbrace{\mathbf{h}}_{\text{prediction}} - \underbrace{\mathbf{t}}_{\text{targets}})\underbrace{\mathbf{x}^T}_{\text{data}}$$

- This applies to:

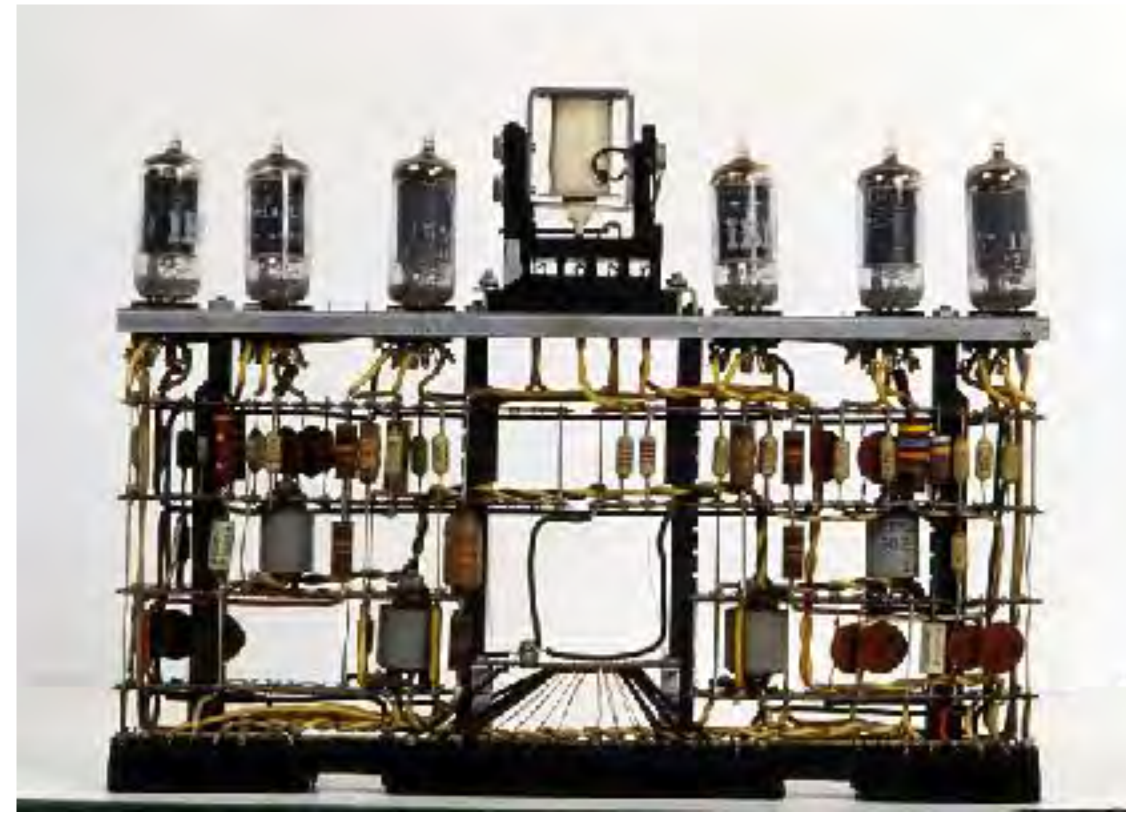
Linear regression	$\mathbf{h} = \mathbf{W}^T\mathbf{x}$	L2 loss
Softmax regression	$\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x})$	cross-entropy loss

- The same update rule with a binary prediction function

$$\mathbf{h} = \mathbb{1}_{\max}(\mathbf{W}^T\mathbf{x})$$

implements the multiclass Perceptron learning rule

History of the Perceptron



[I.B.M. Italia]

- This machine (IBM 704) was used by Frank Rosenblatt to implement the perceptron in 1958
- Based on his statements, the New York Times reported it as: "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

2-class Perceptron Classifier

- Classification function is

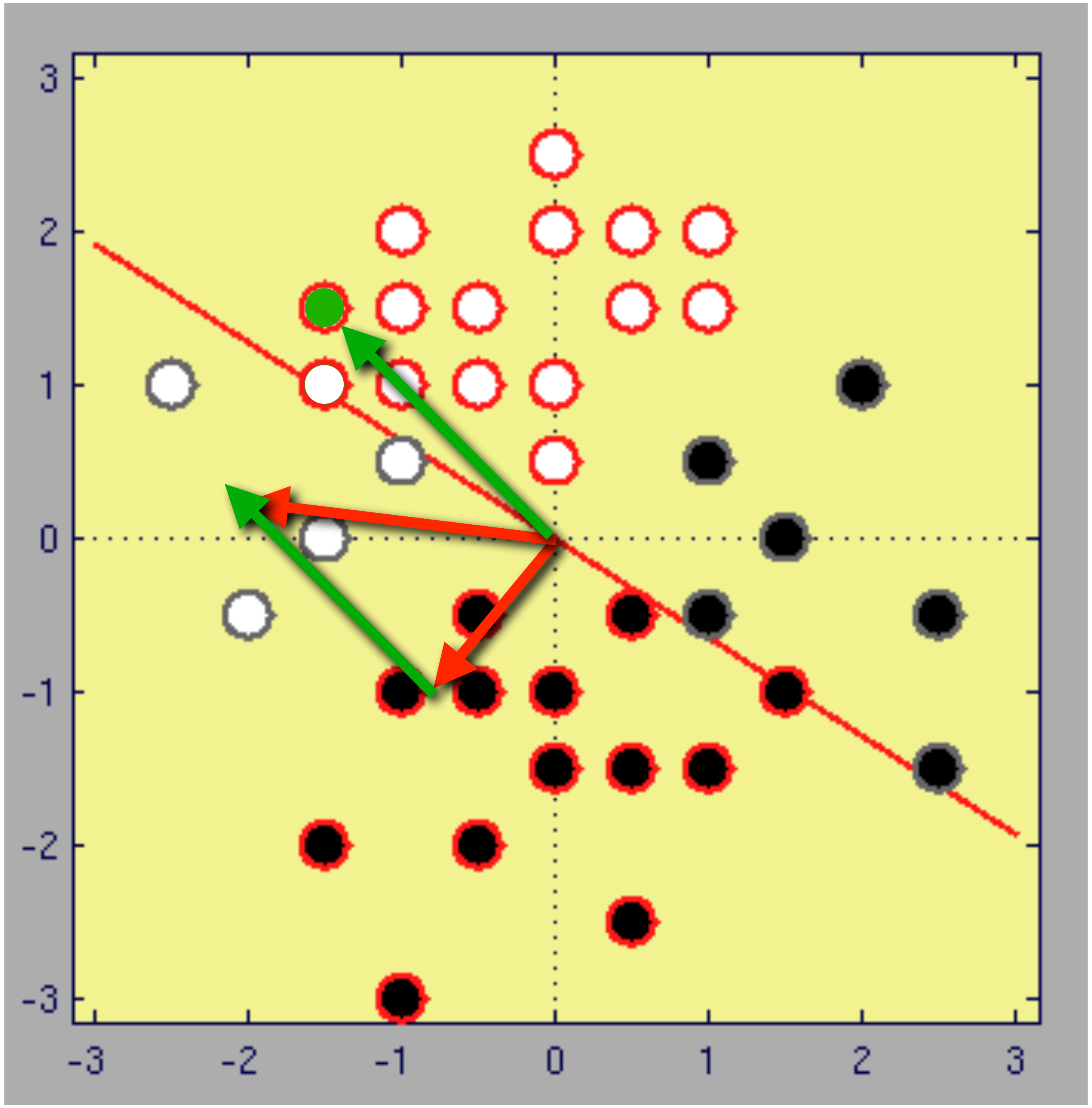
$$\hat{y} = \mathrm{sign}(\mathbf{w}^T \mathbf{x})$$

- Linear function of the data (x) followed by 0/1 activation

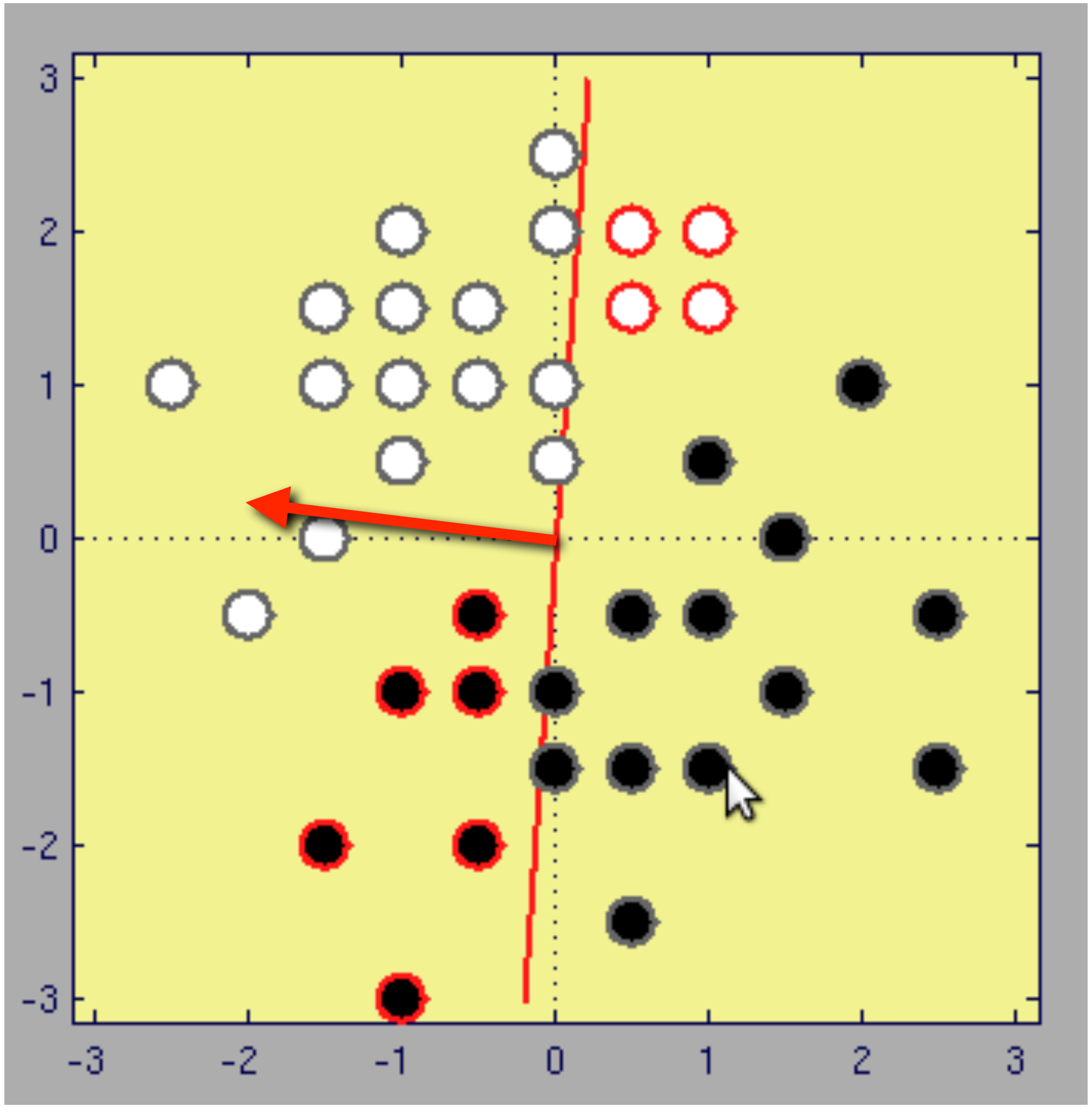
- Update rule: present data x
  - if correctly classified, do nothing
  - if incorrectly classified, update the weight vector

$$\mathbf{w}_{n+1} = \mathbf{w}_n + y_i \mathbf{x}_i$$

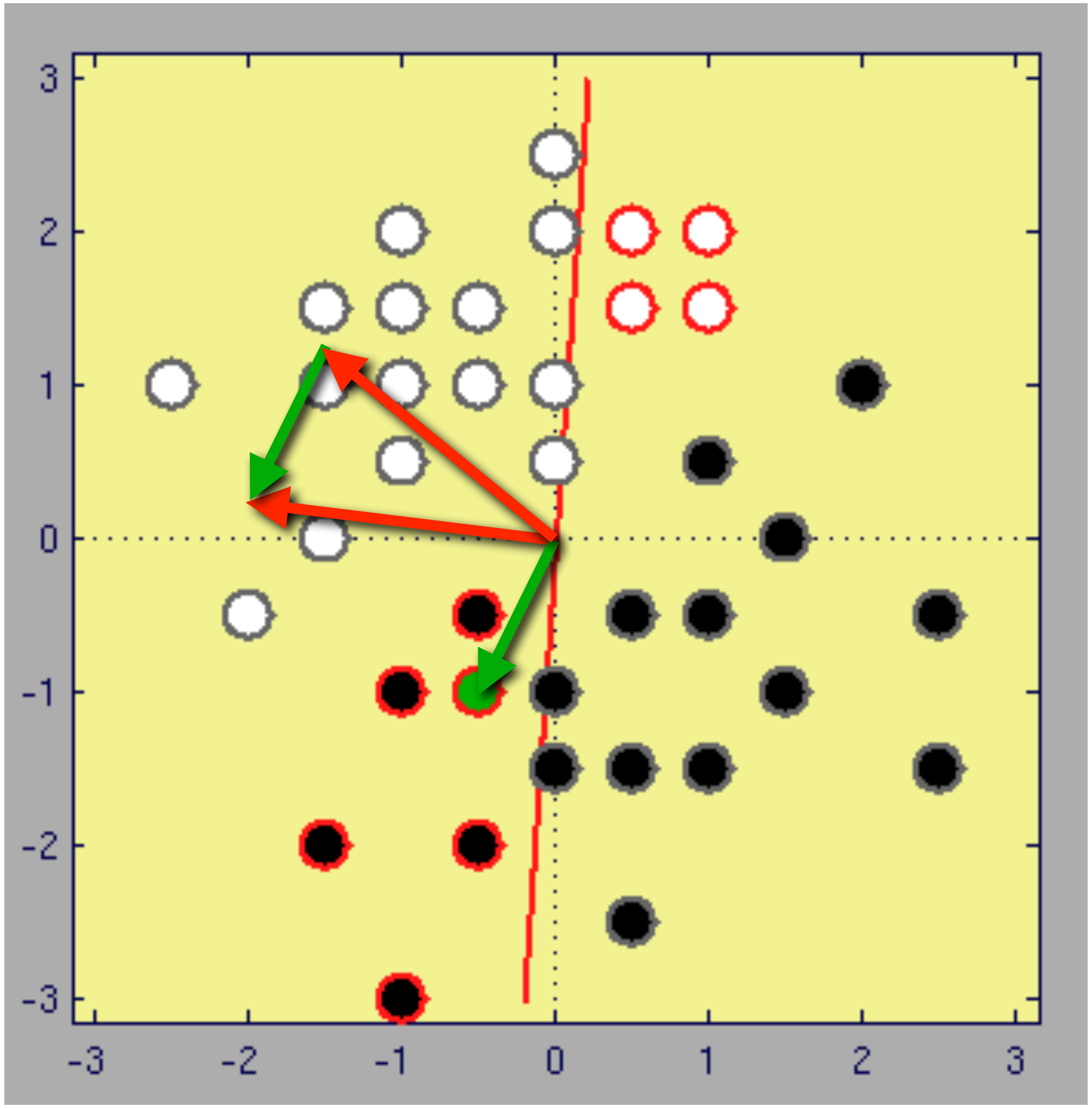
Example of Perceptron Learning



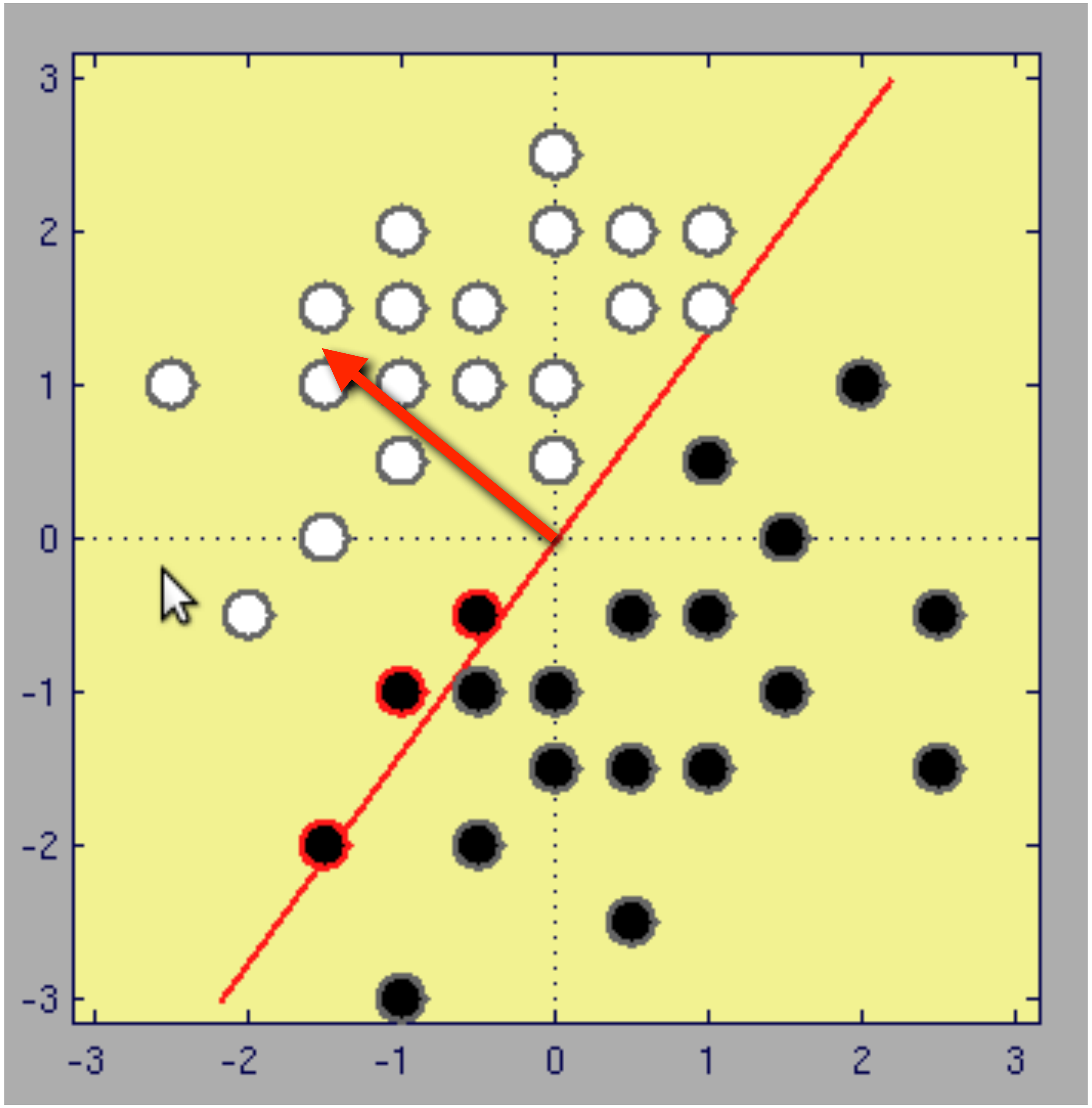
Example of Perceptron Learning



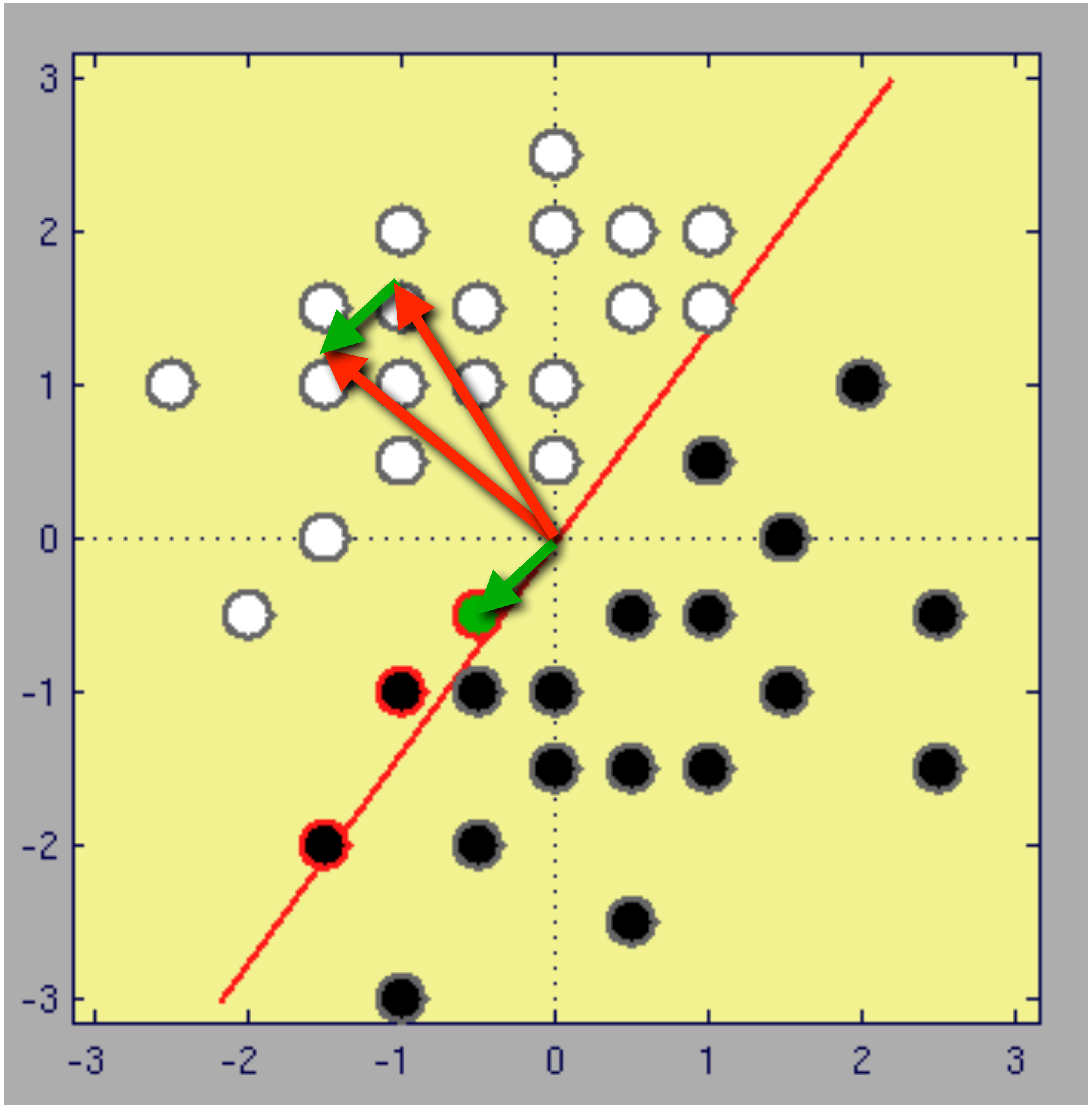
Example of Perceptron Learning



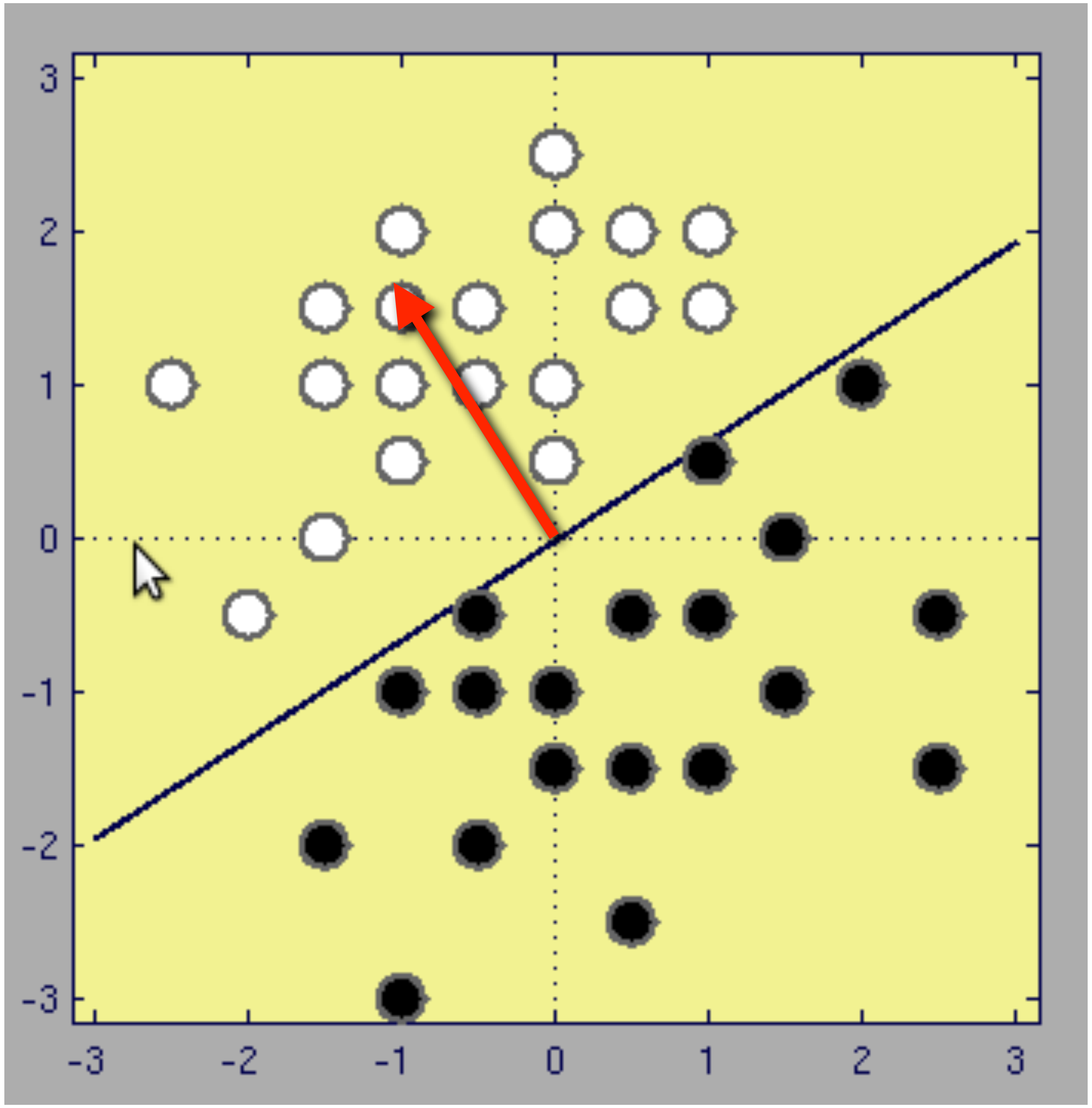
Example of Perceptron Learning



Example of Perceptron Learning

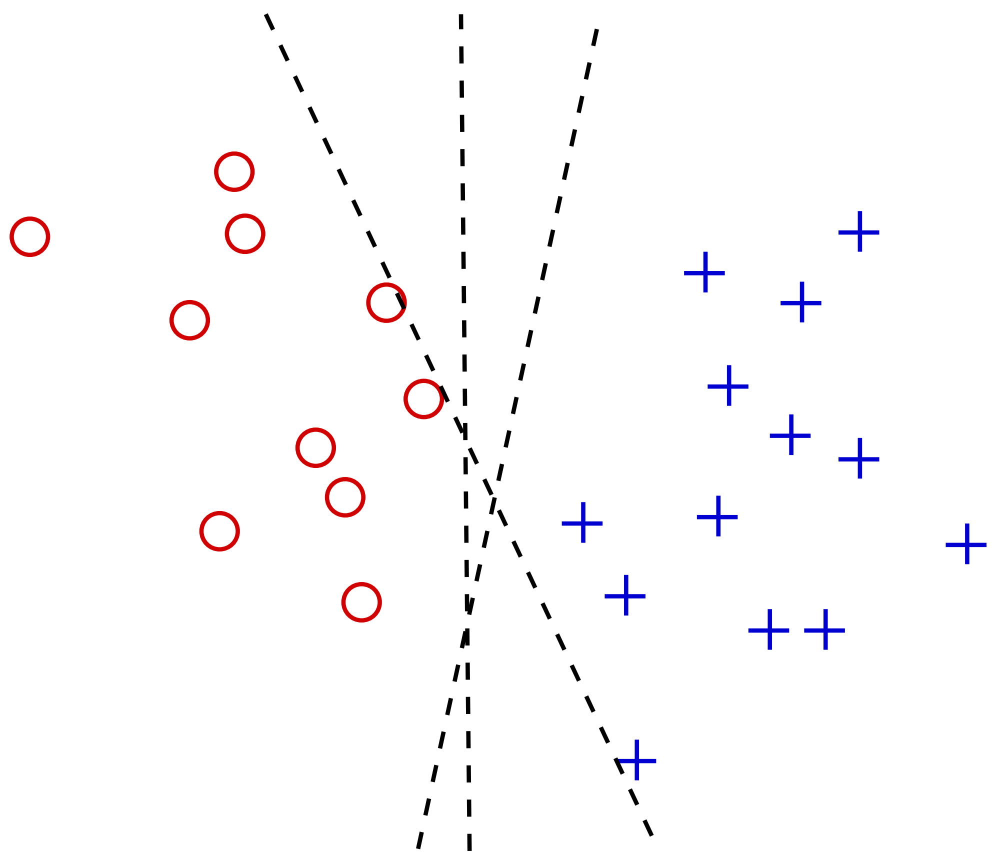


Example of Perceptron Learning

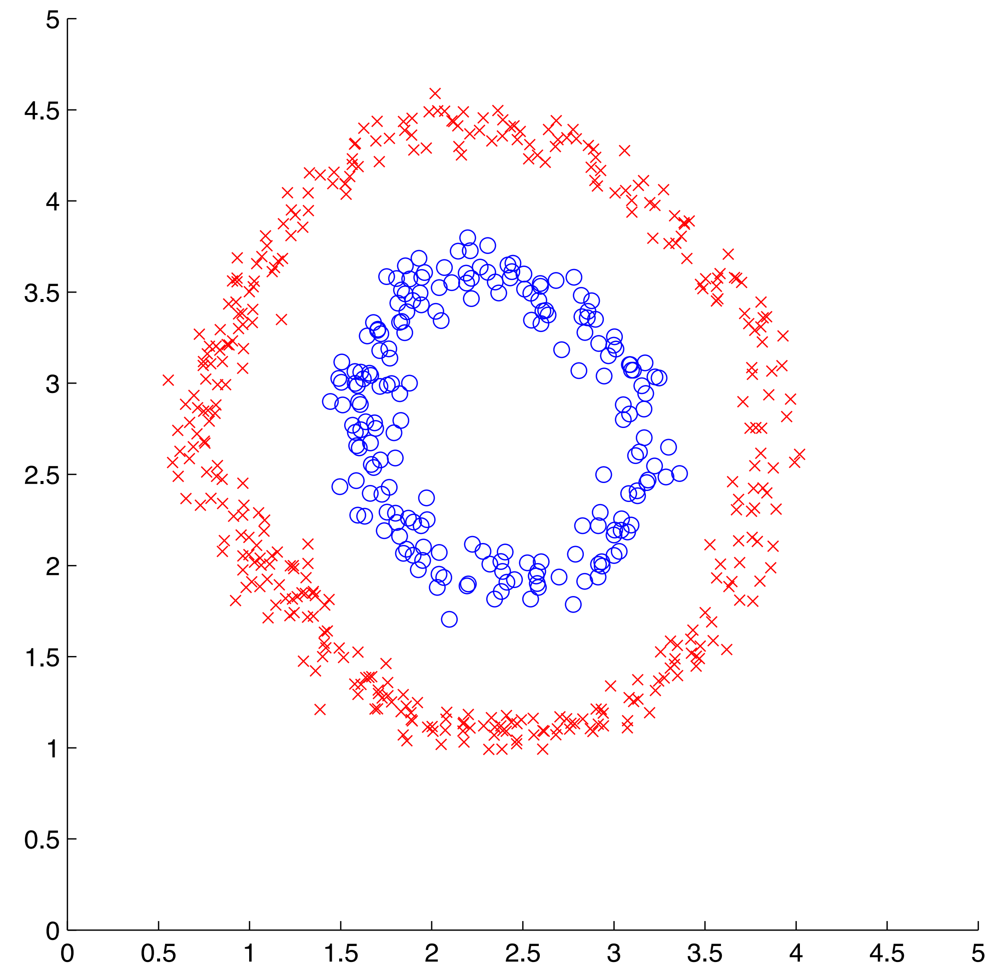


Perceptron Limitations

- Perceptrons + linear + softmax regressors are limited to data that are linearly separable, e.g.,



Linearly separable

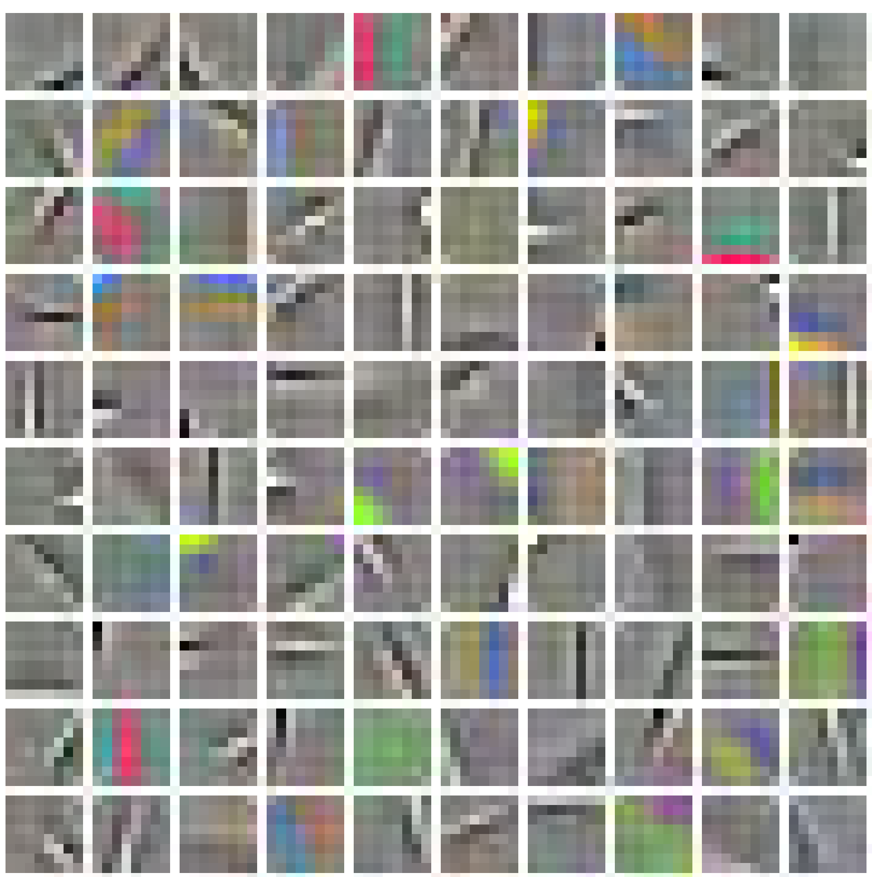


Not linearly separable

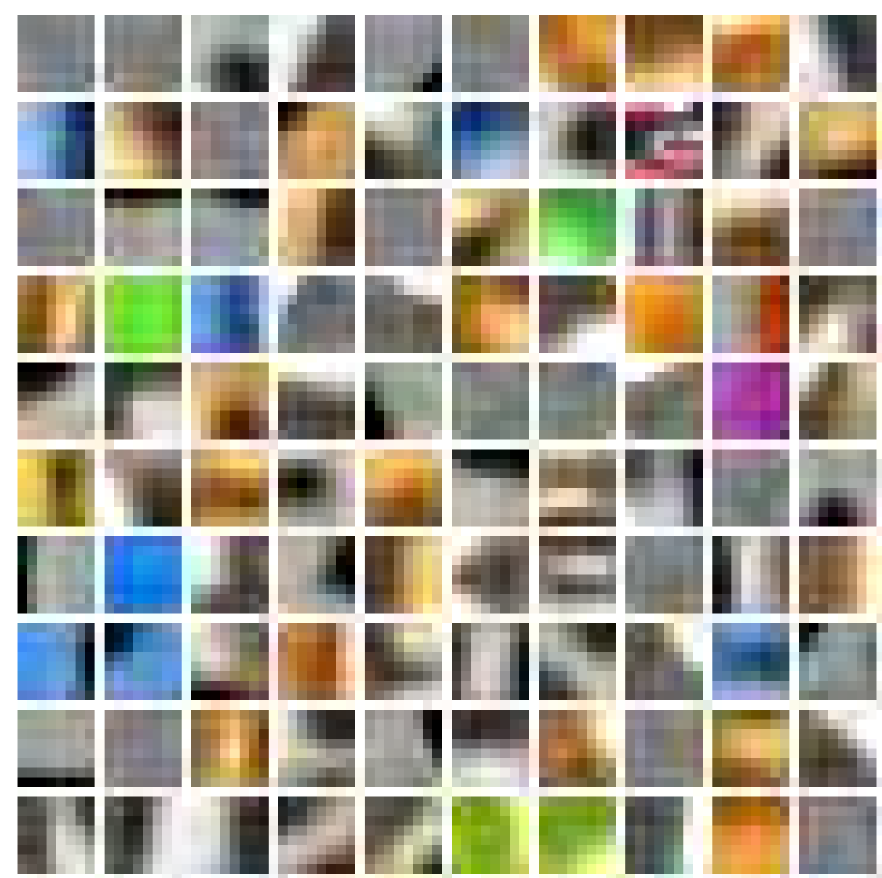
How could we transform the RHS to be linearly separable?

CIFAR10 Feature Extraction

- So far, we used RGB pixels as the input to our classifier
- Feature extraction can improve results by a lot
- e.g., Coates et al. achieve 79.6% accuracy on CIFAR10 with a features based on k-means of whitened image patches



k-means, whitened

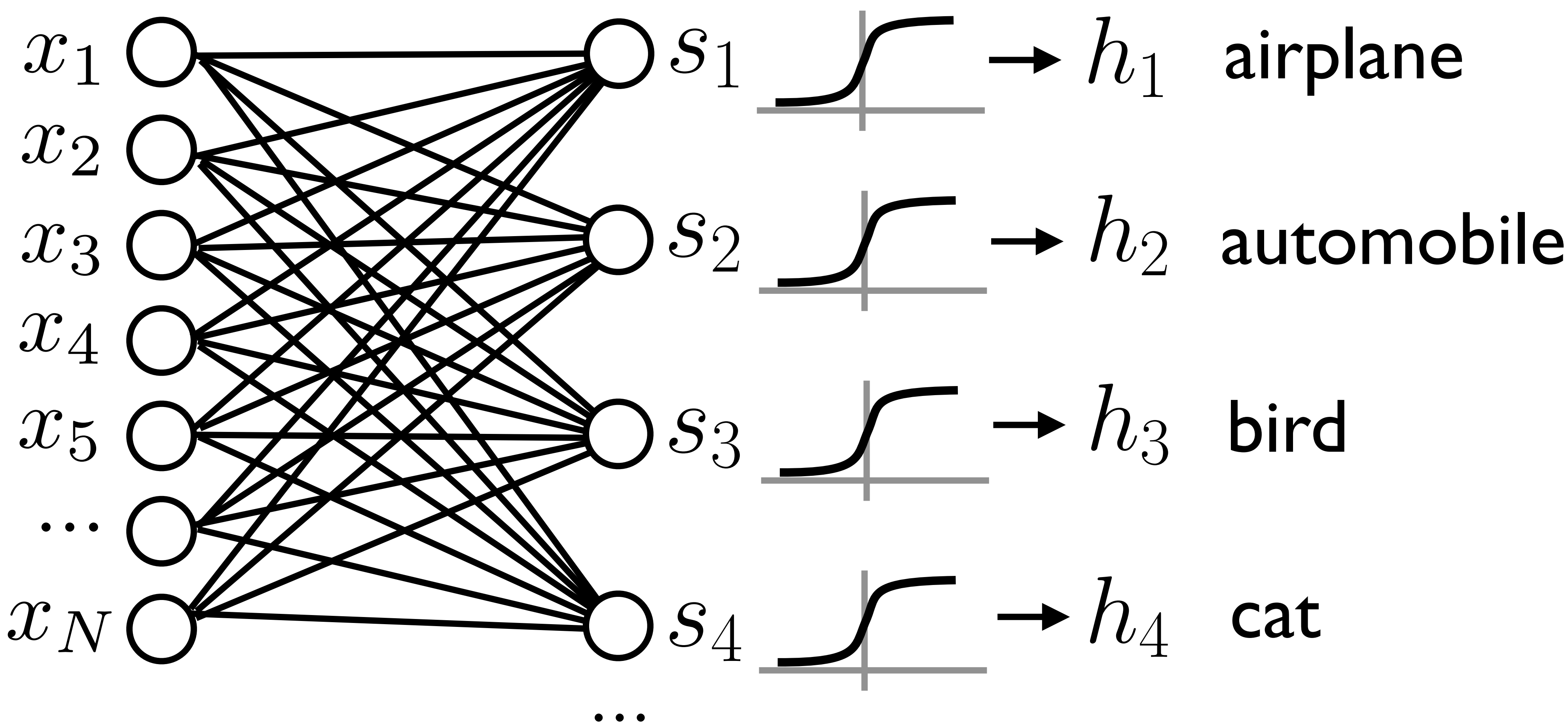


k-means, raw RGB

[Coates et al. 2011]

Linear = Fully Connected Layer

- Note that our linear matrix multiplication classifier is equivalent to a fully connected layer in a neural network

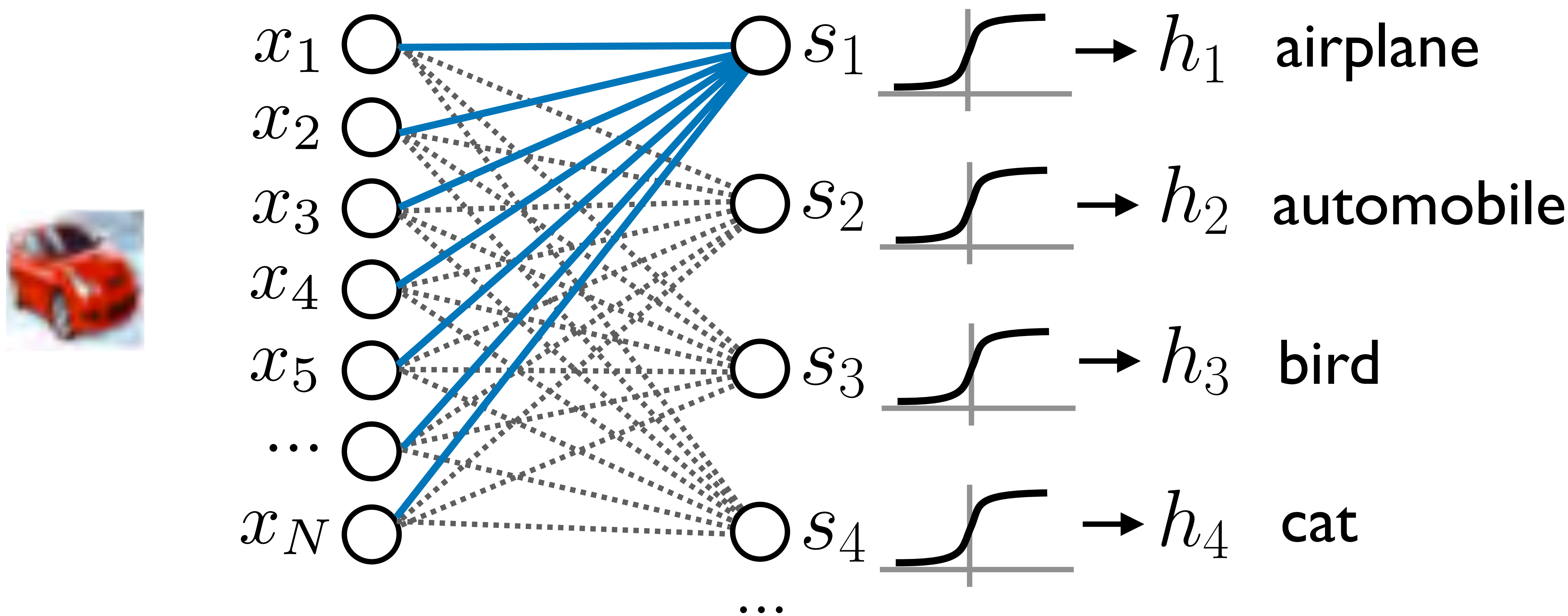


- Typically, we'll also add a bias term b

$$\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

Linear = Fully Connected Layer

- Note that our linear matrix multiplication classifier is equivalent to a fully connected layer in a neural network

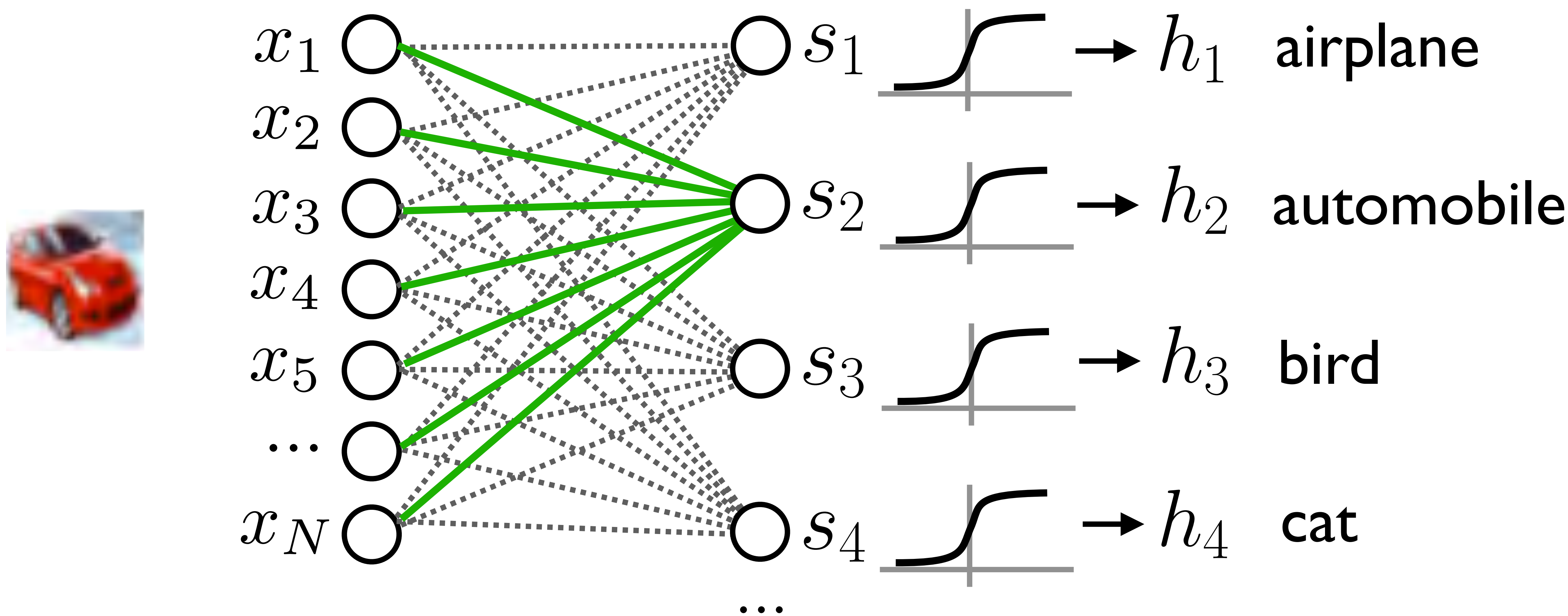


- Typically, we'll also add a bias term b

$$\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

Linear = Fully Connected Layer

- Note that our linear matrix multiplication classifier is equivalent to a fully connected layer in a neural network

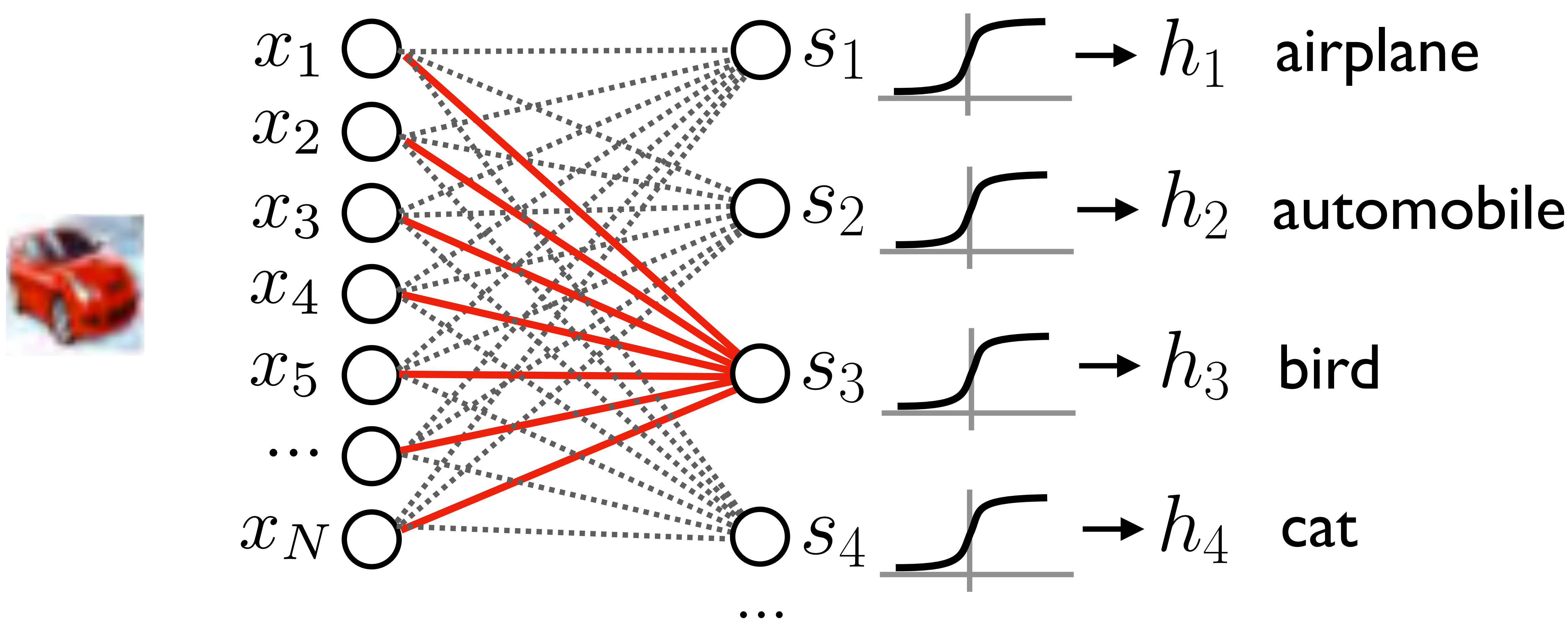


- Typically, we'll also add a bias term b

$$\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

Linear = Fully Connected Layer

- Note that our linear matrix multiplication classifier is equivalent to a fully connected layer in a neural network

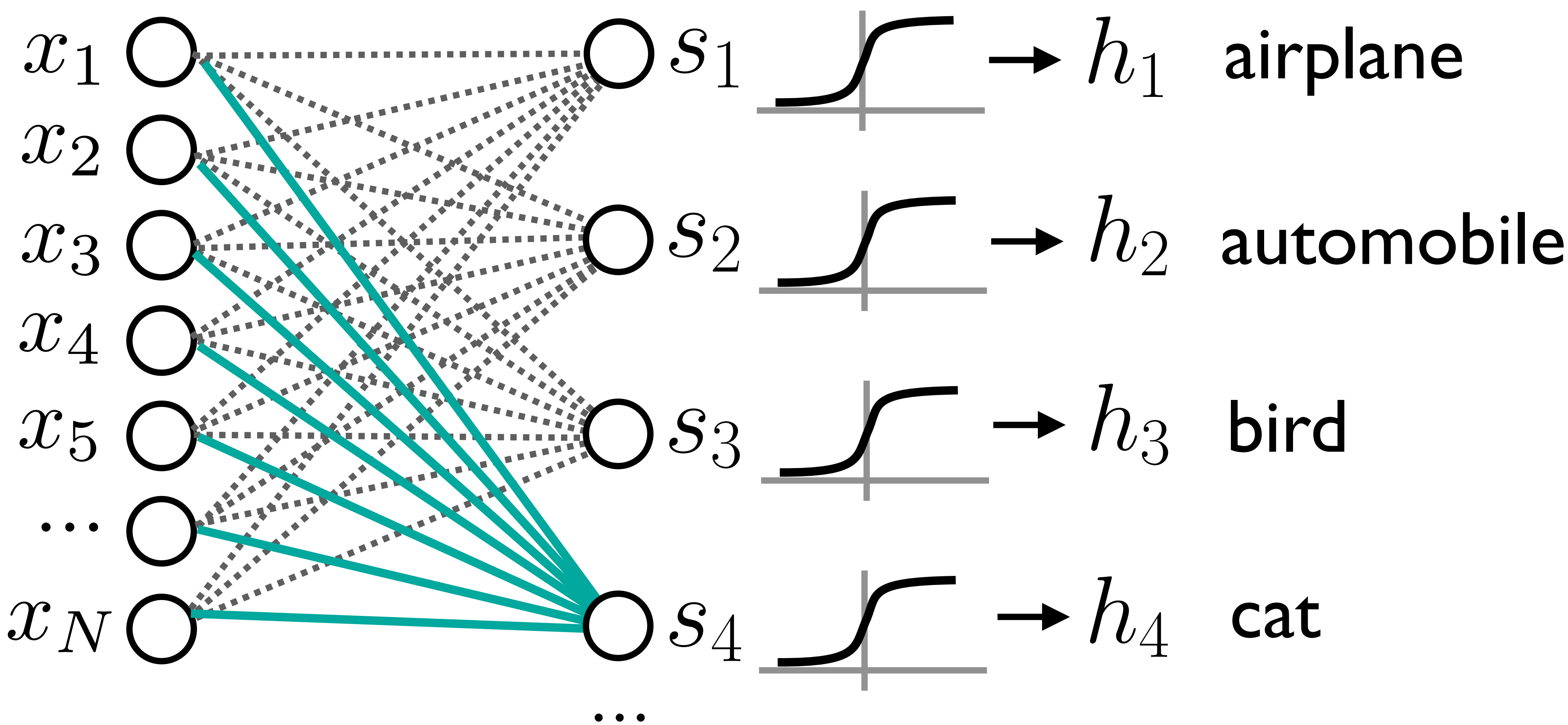


- Typically, we'll also add a bias term b

$$\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

Linear = Fully Connected Layer

- Note that our linear matrix multiplication classifier is equivalent to a fully connected layer in a neural network



- Typically, we'll also add a bias term b

$$\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

Next Lecture

- Visual Classification 2: Fundamentals + Pre-deep learning