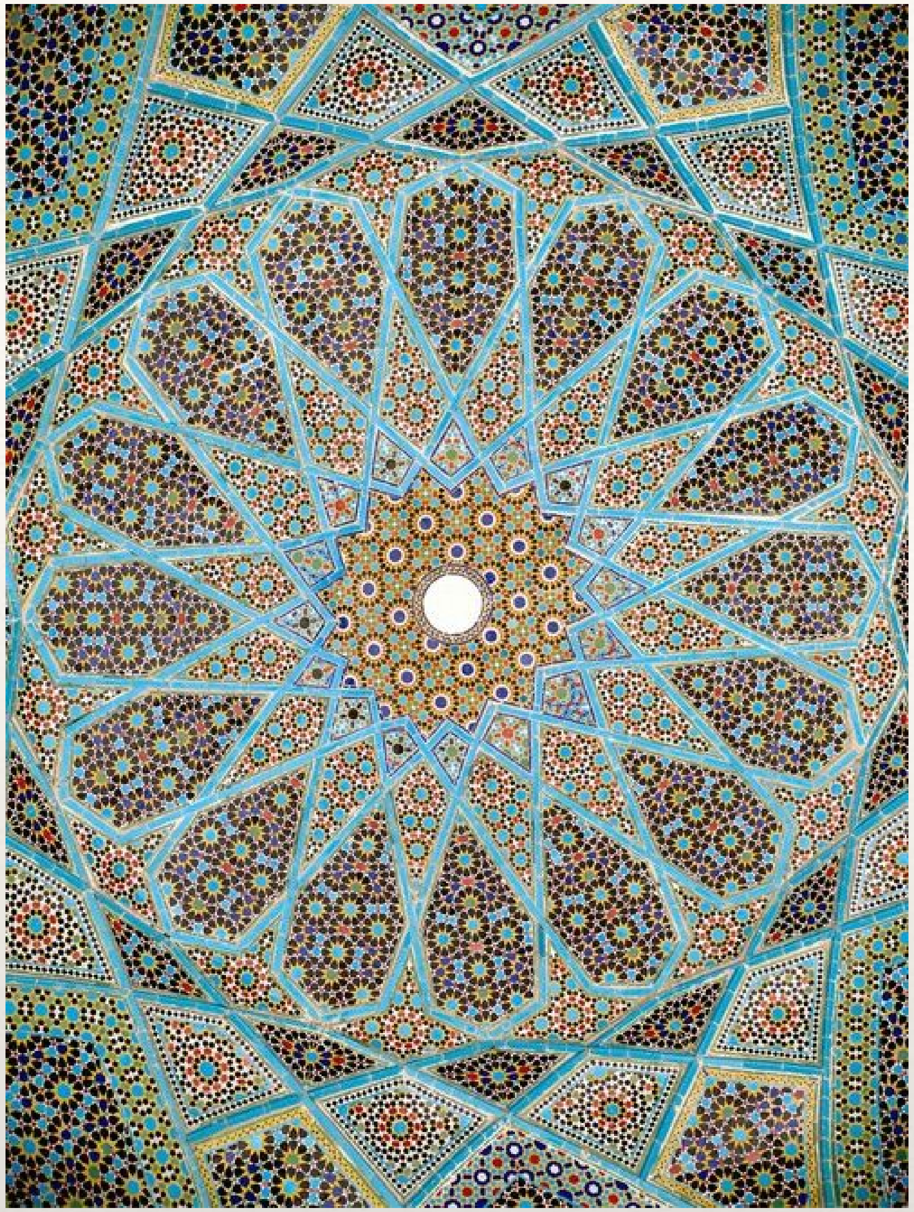
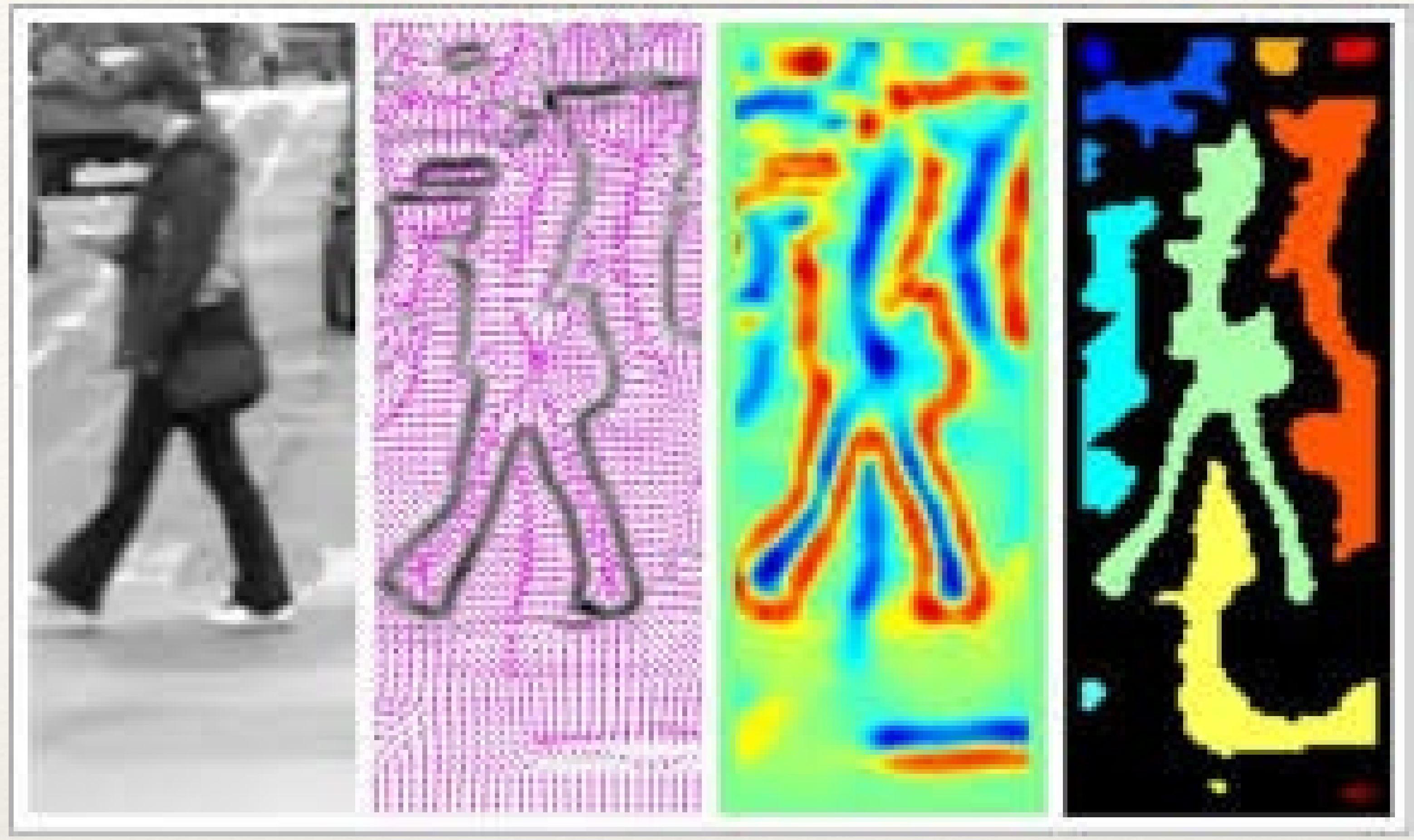
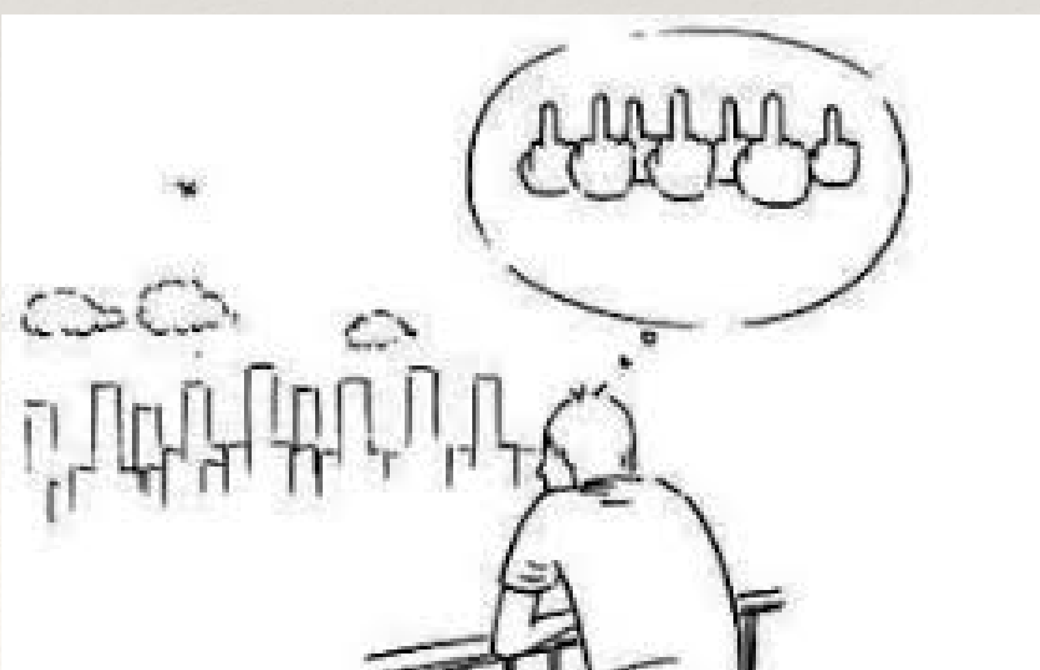
Computer Vision

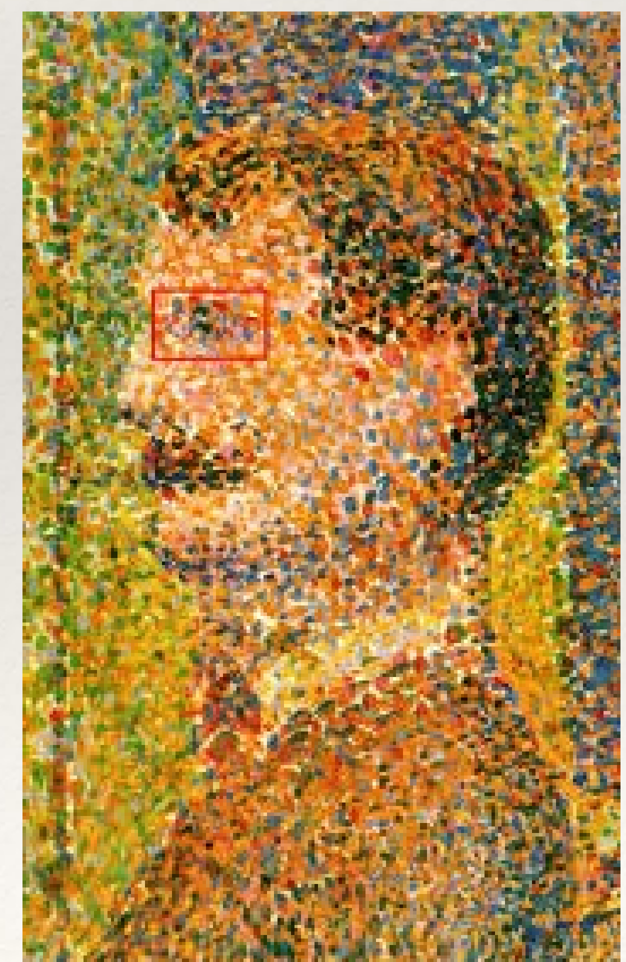
Machine learning for pattern recognition

Hansung Kim
h.kim@soton.ac.uk





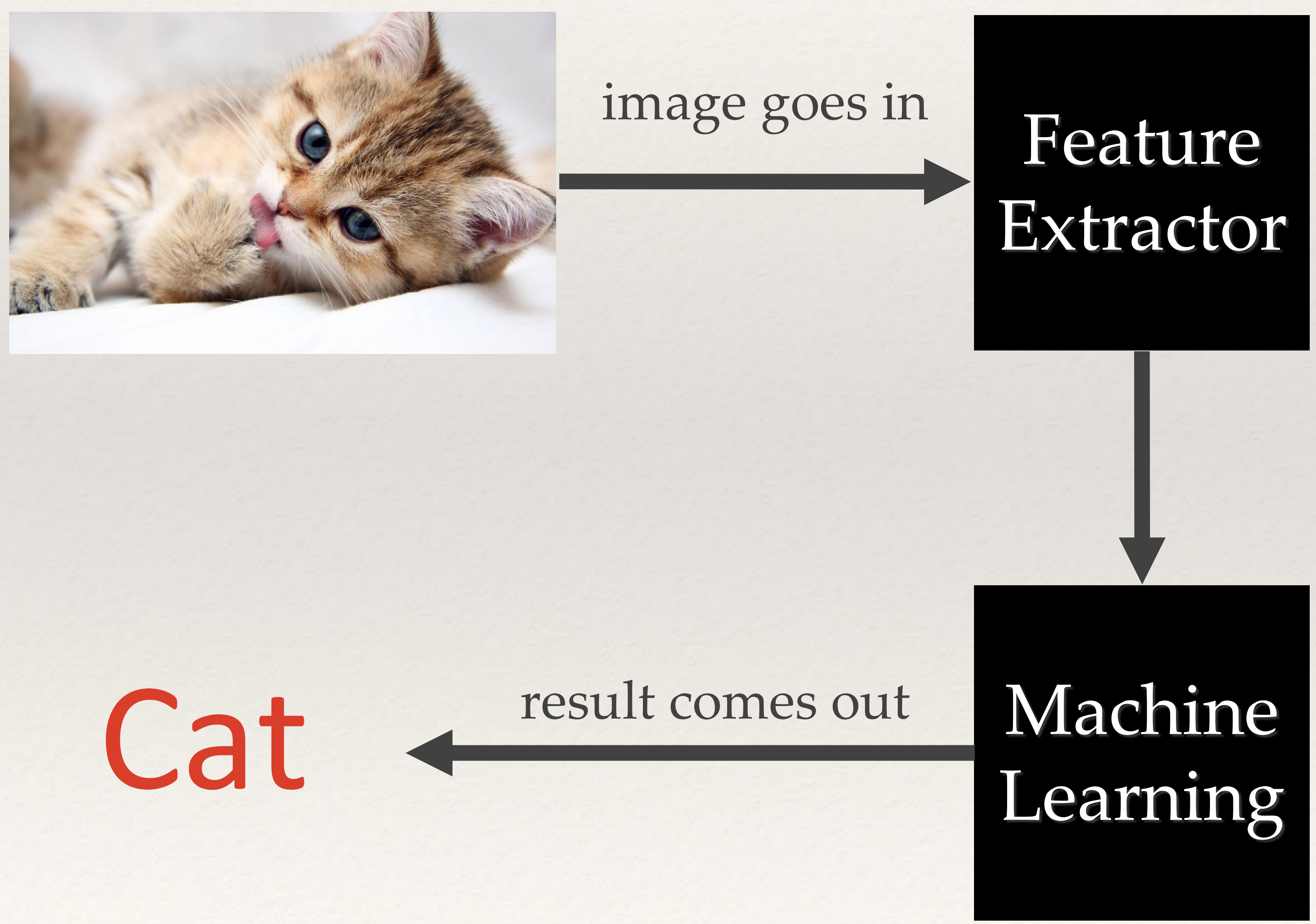




- Recognising patterns is a large part of computer vision
    - i.e. recognising text, people, objects, …
- Obviously there’s a lot of overlap with intelligent algorithms, machine learning and AI.
- This lecture will cover (recap?) some of the fundamentals of machine learning and introduce how you connect arrays of pixels to machine learning algorithms.

Feature spaces

Many computer vision applications involving machine learning take the following form:



Many computer vision applications involving machine learning take the following form:

this is where cool image processing happens

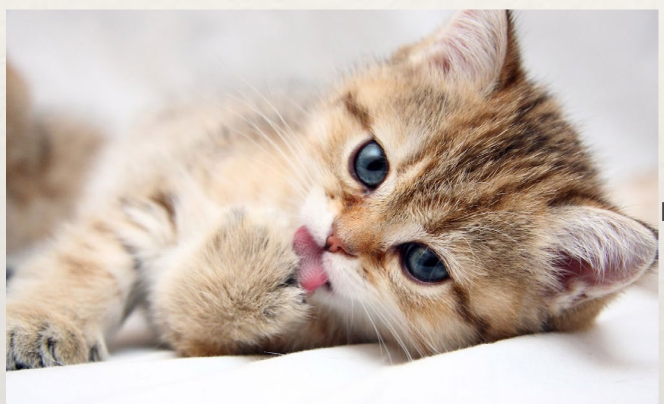


image goes in

Feature Extractor

Machine Learning

result comes out

Cat

Many computer vision applications involving machine learning take the following form:

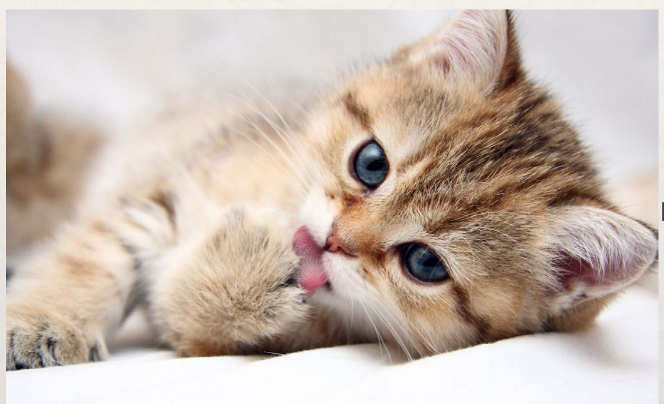


image goes in

Feature Extractor

feature extractors make feature vectors from images

Machine Learning

result comes out

Cat

Many computer vision applications involving machine learning take the following form:

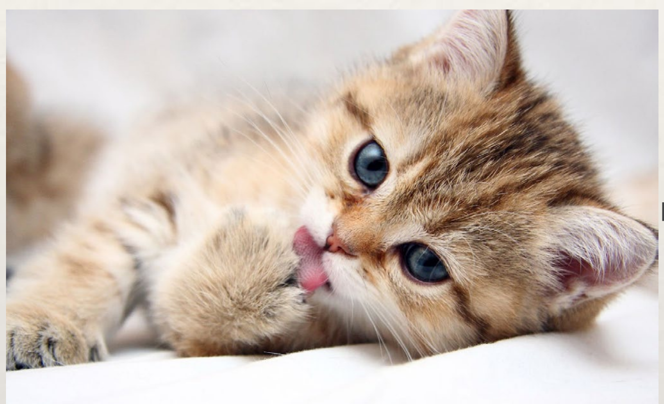


image goes in

Feature Extractor

Machine Learning

result comes out

Cat

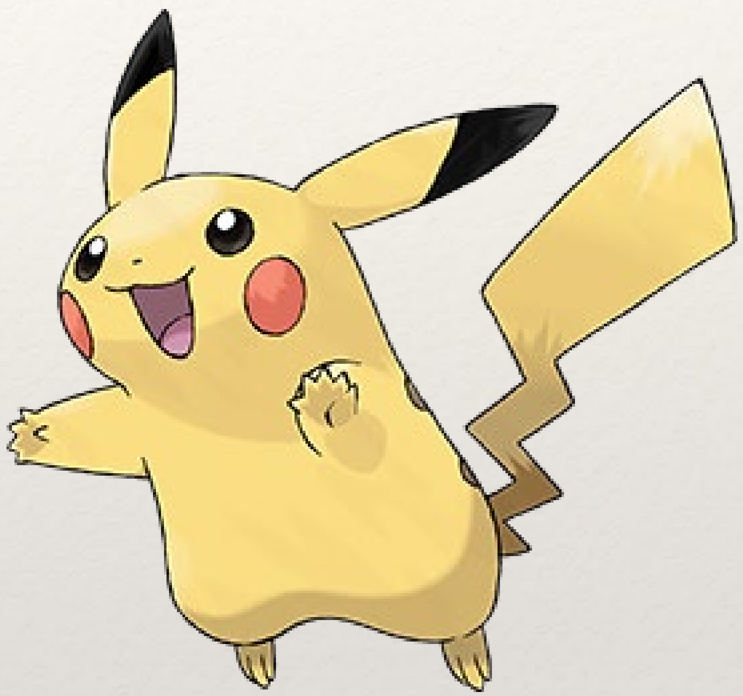
machine learning system uses feature vectors to make intelligent decisions

Key terminology

- **Feature vector**: a mathematical vector
  - just a list of (usually Real) numbers
  - has a fixed number of **elements** in it
    - The number of elements is the **dimensionality** of the vector
  - represents a **point** in a **feature space** or equally a **direction** in the feature space
  - the **dimensionality of a feature space** is the dimensionality of every vector within it
    - vectors of differing dimensionality can't exist in the same feature space

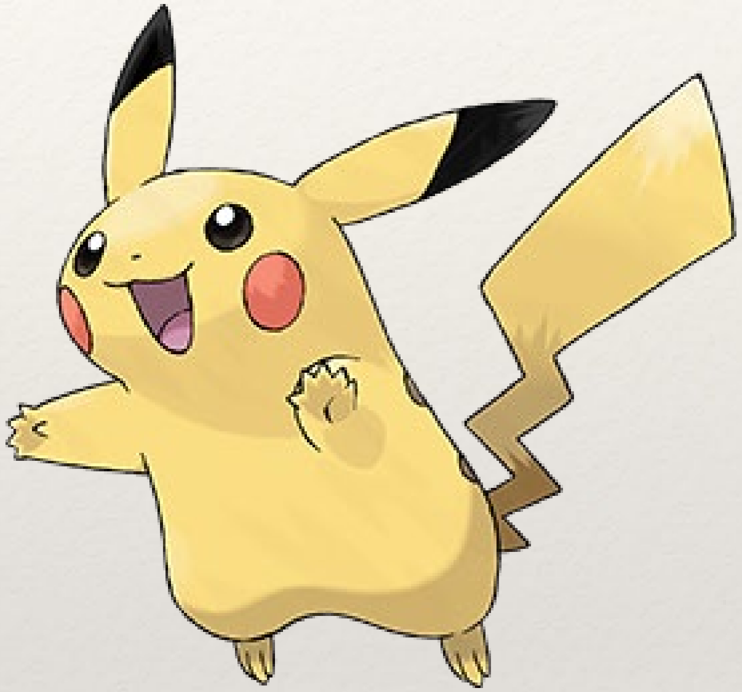
Simple feature vectors

- What kind of feature vectors can you extract from this Pikachu image?
  - No semantic interpretation!



Simple feature vectors

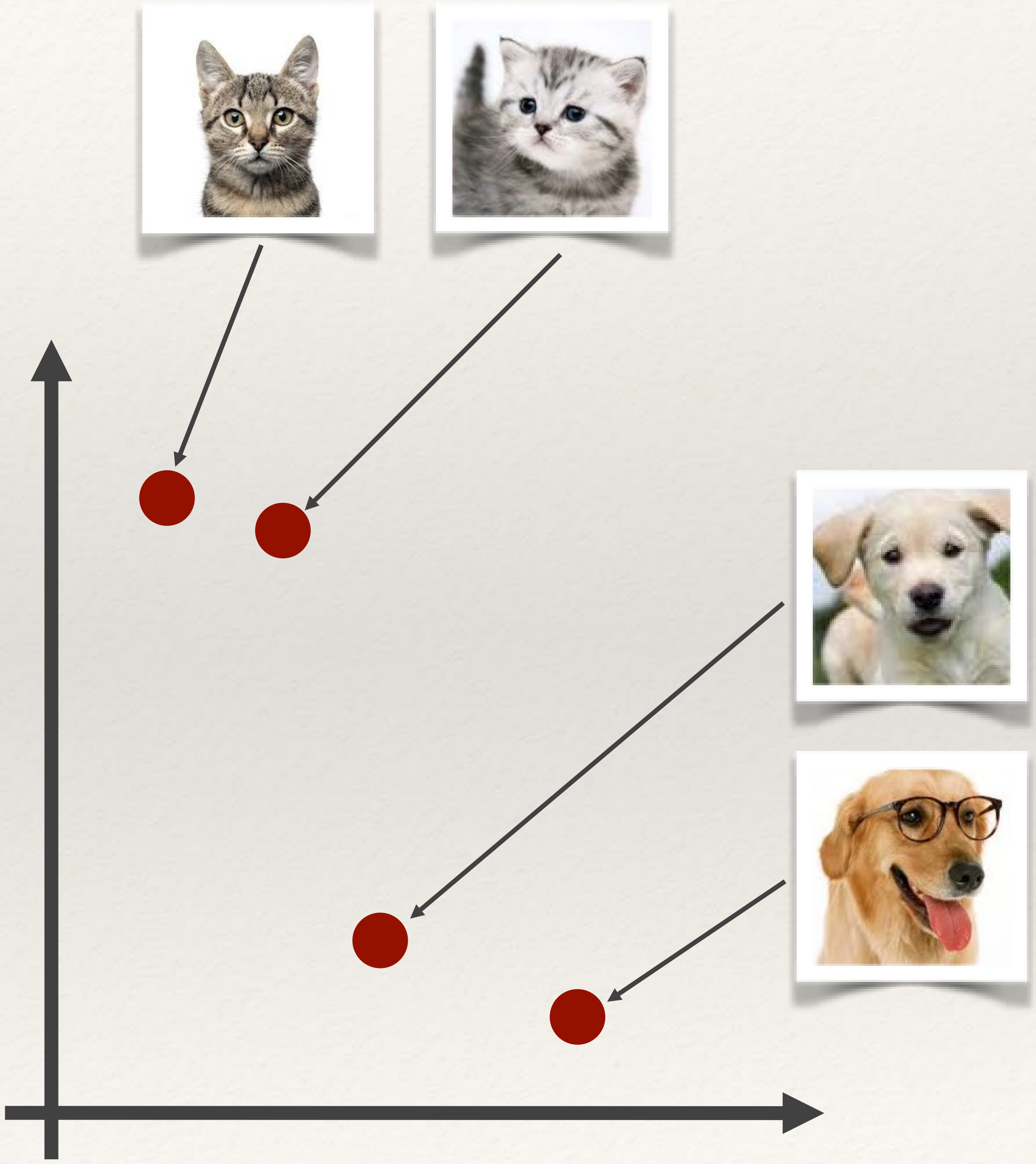
- Dimensions of the image
  - (256, 274, 3) (Height, width, channel)
- Colour mean
  - (51, 83, 95) (Blue, Green, Red)
- Mean and Standard Deviation
  - (51, 83, 95)(62.43, 101.17, 114.51)
- Colour histogram
  - 3 x 256 array or 256^3 vector



Distance and similarity

Distances in feature space

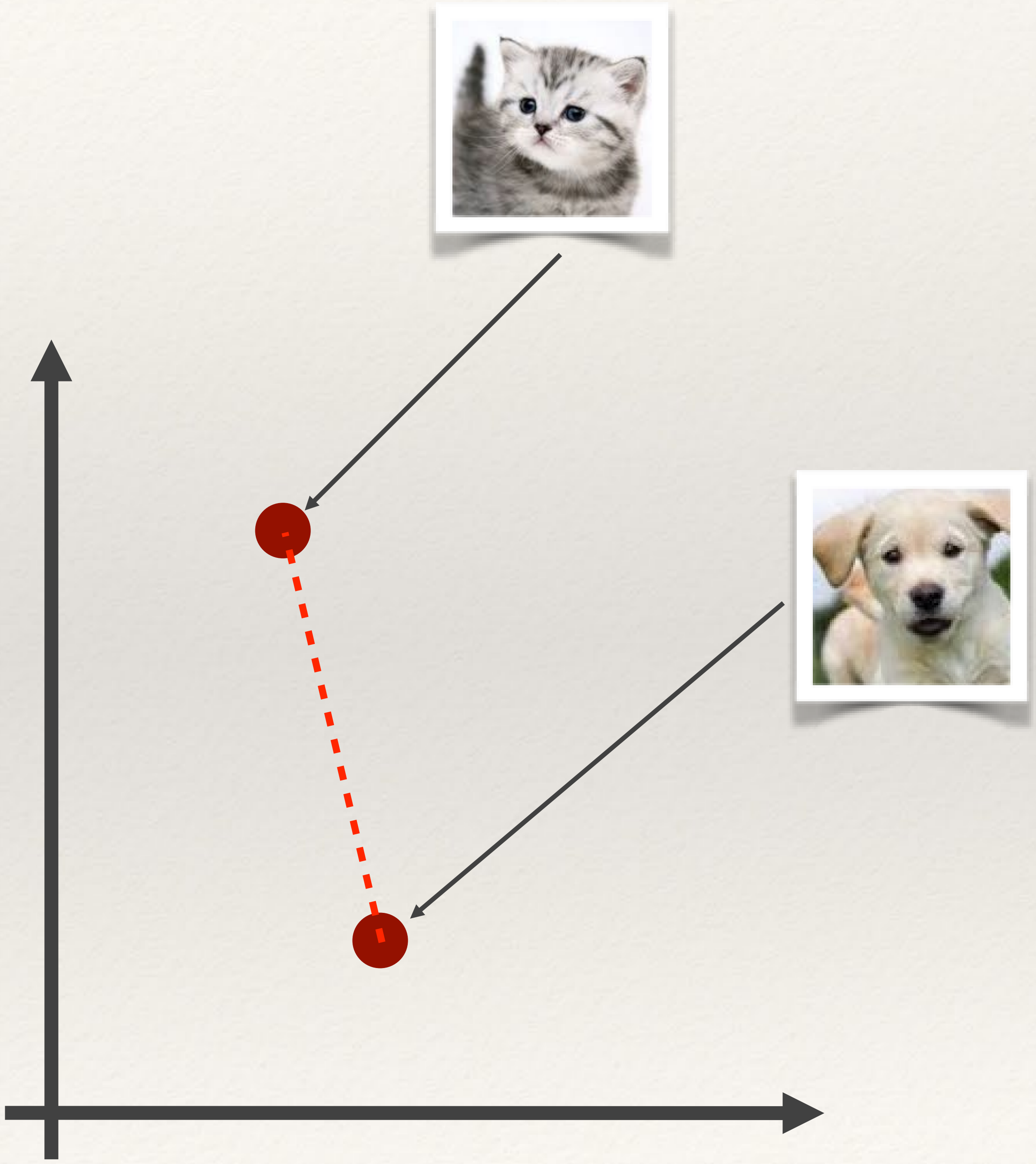
- Feature extractors are often defined so that they produce vectors that are *close* together for *similar* inputs
  - Closeness of two vectors can be computed in the feature space by measuring a distance between the vectors.



Euclidean distance (L2 distance)

- L2 distance is the most intuitive distance…
  - The straight-line distance between two points
  - Computed via an extension of Pythagoras theorem to *n* dimensions:

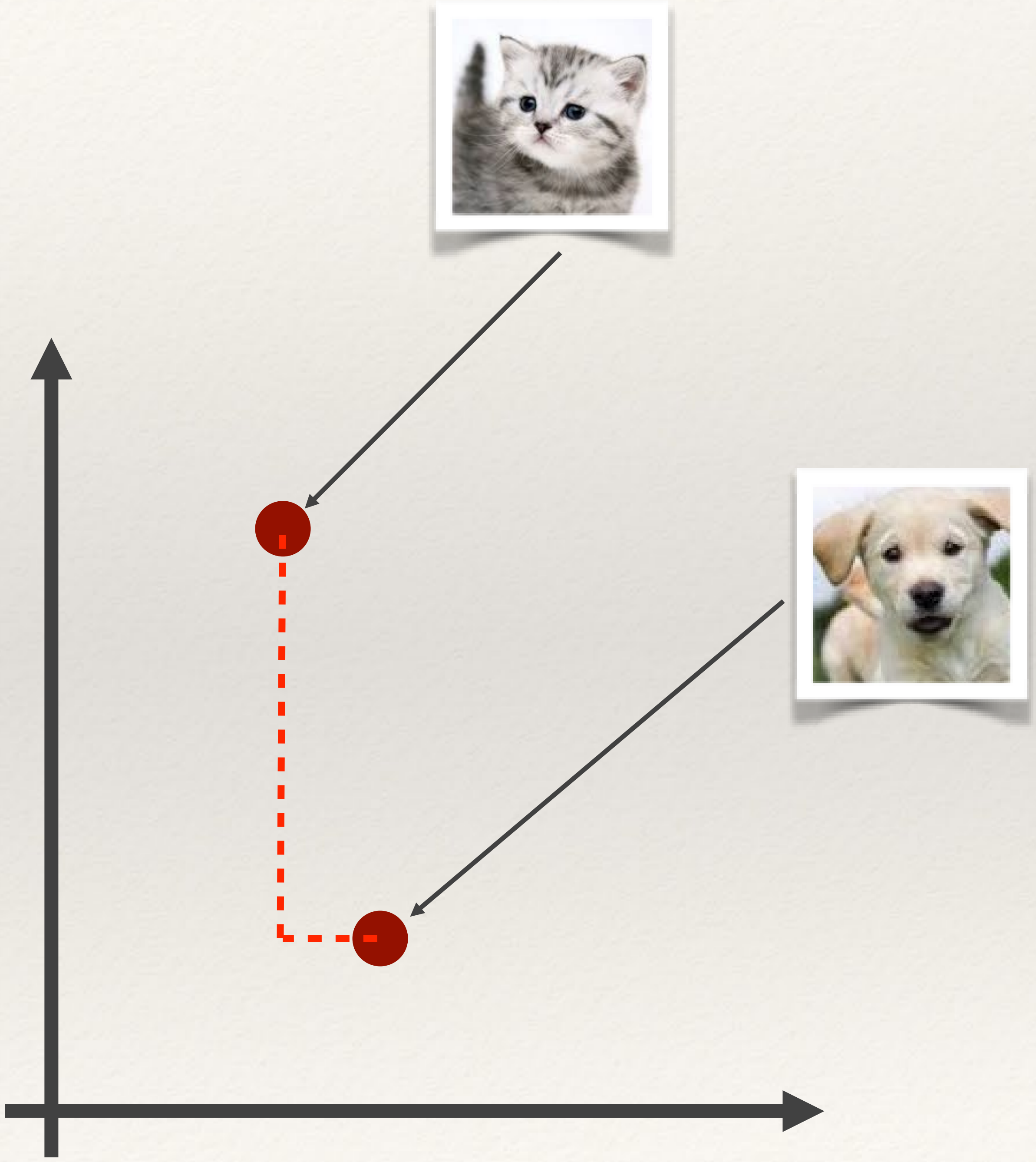
$$D_2(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} = \| p - q \| = \sqrt{(p-q)\cdot(p-q)}$$



L1 distance (aka Taxicab/Manhattan)

- L1 distance is computed along paths parallel to the axes of the space:

$$D_1(p,q) = \sum_{i=1}^{n} |p_i - q_i| = \|p-q\|_1$$

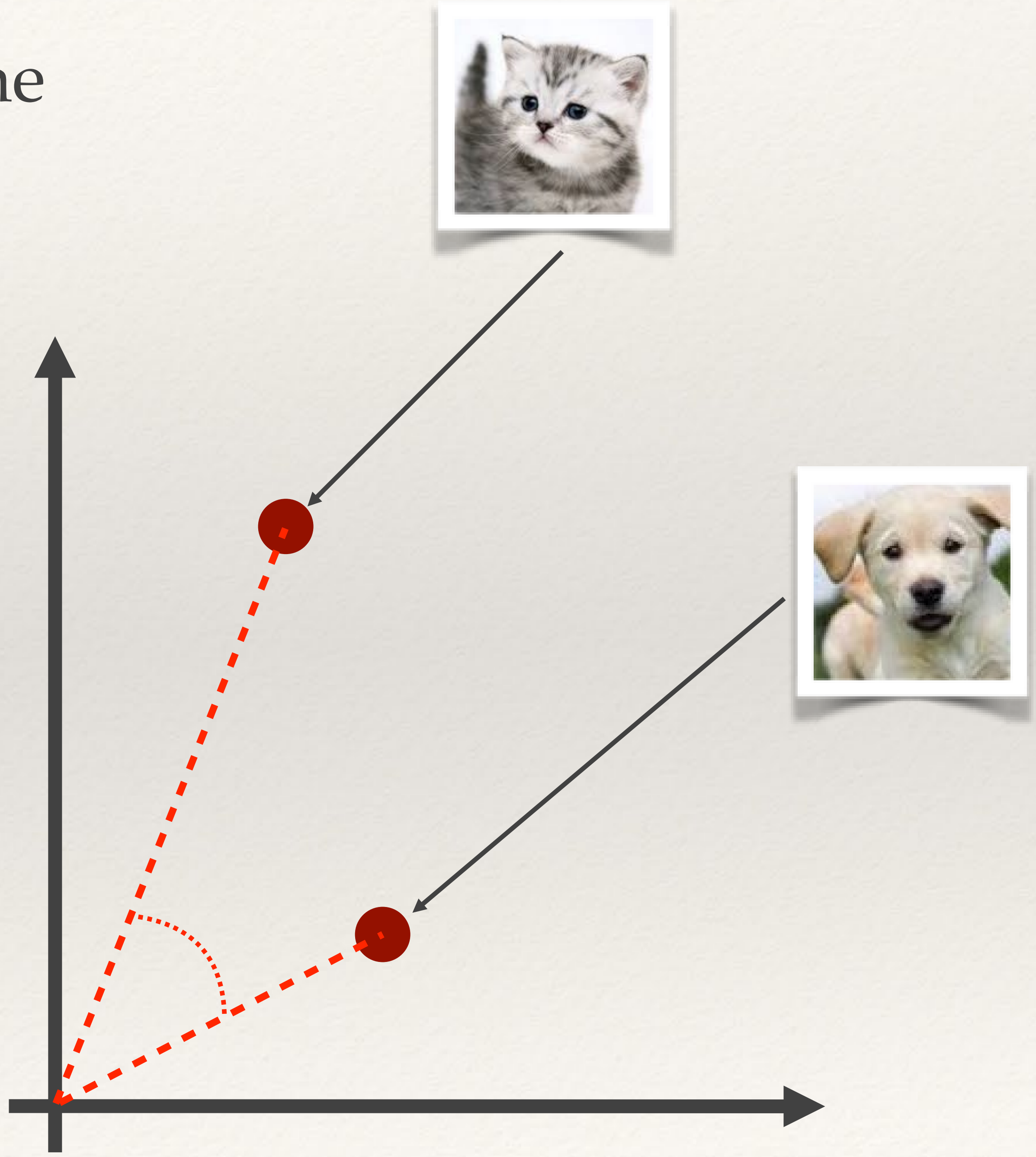


Cosine Similarity

- Cosine similarity measures the cosine of the angle between two vectors
  - **It is not a distance!**

$$cos(\theta) = \frac{p.q}{||p||\,||q||} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}$$

  - Useful if you don't care about the relative length of the vectors
  - Any example in our previous lectures?



Choosing good feature vector representations for machine-learning

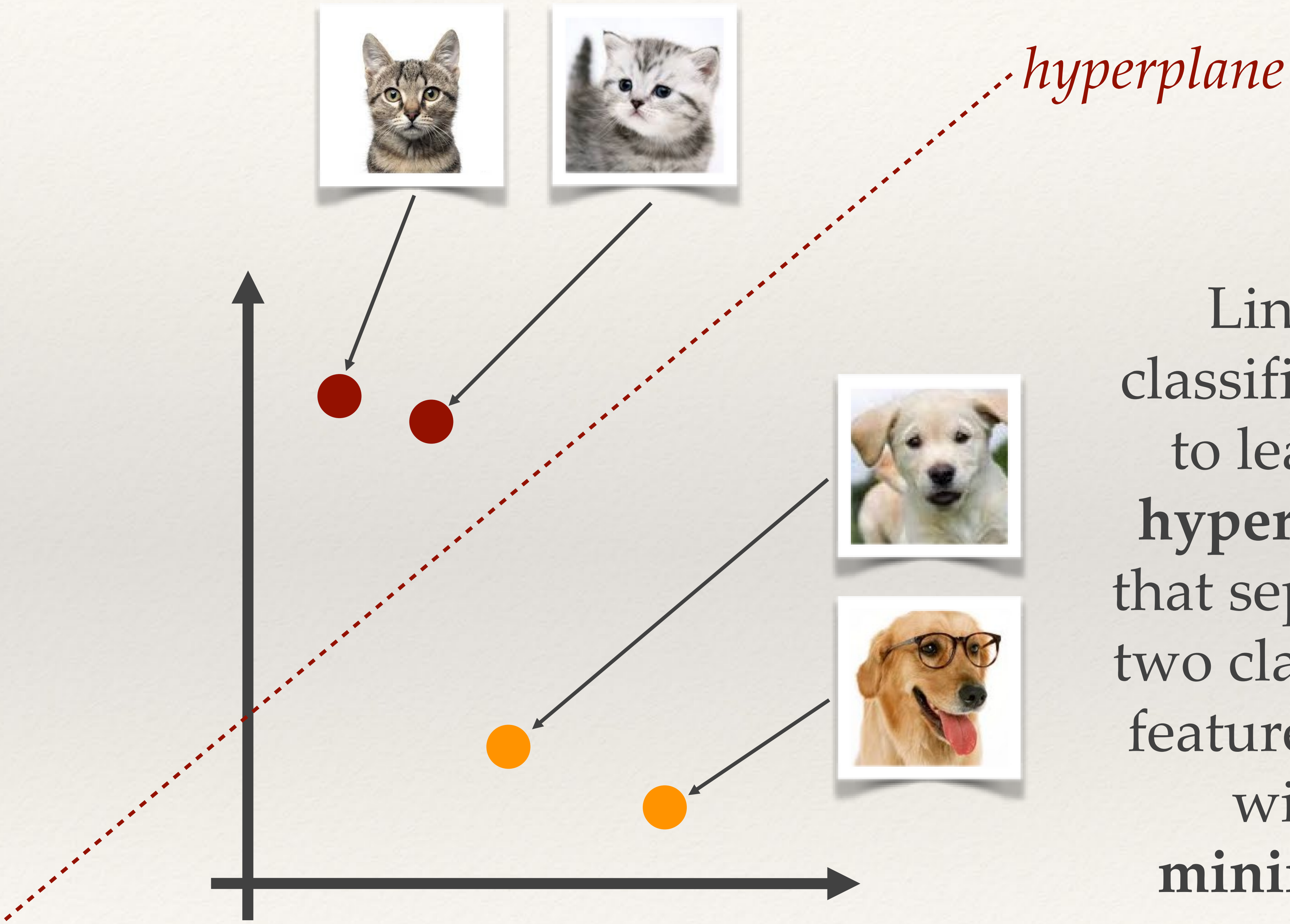
- Choose features which allow to distinguish objects or classes of interest
  - Similar within classes
  - Different between classes
- Keep number of features small
  - Machine-learning can get more difficult as dimensionality of featurespace gets large

Supervised Machine Learning:
Classification

- **Classification** is the process of assigning a **class label** to an object.
- A **supervised machine-learning algorithm** uses a set of pre-labelled *training data* to learn how to assign class labels to vectors (and the corresponding objects).
  - A **binary** classifier only has two classes
  - A **multiclass** classifier has many classes.

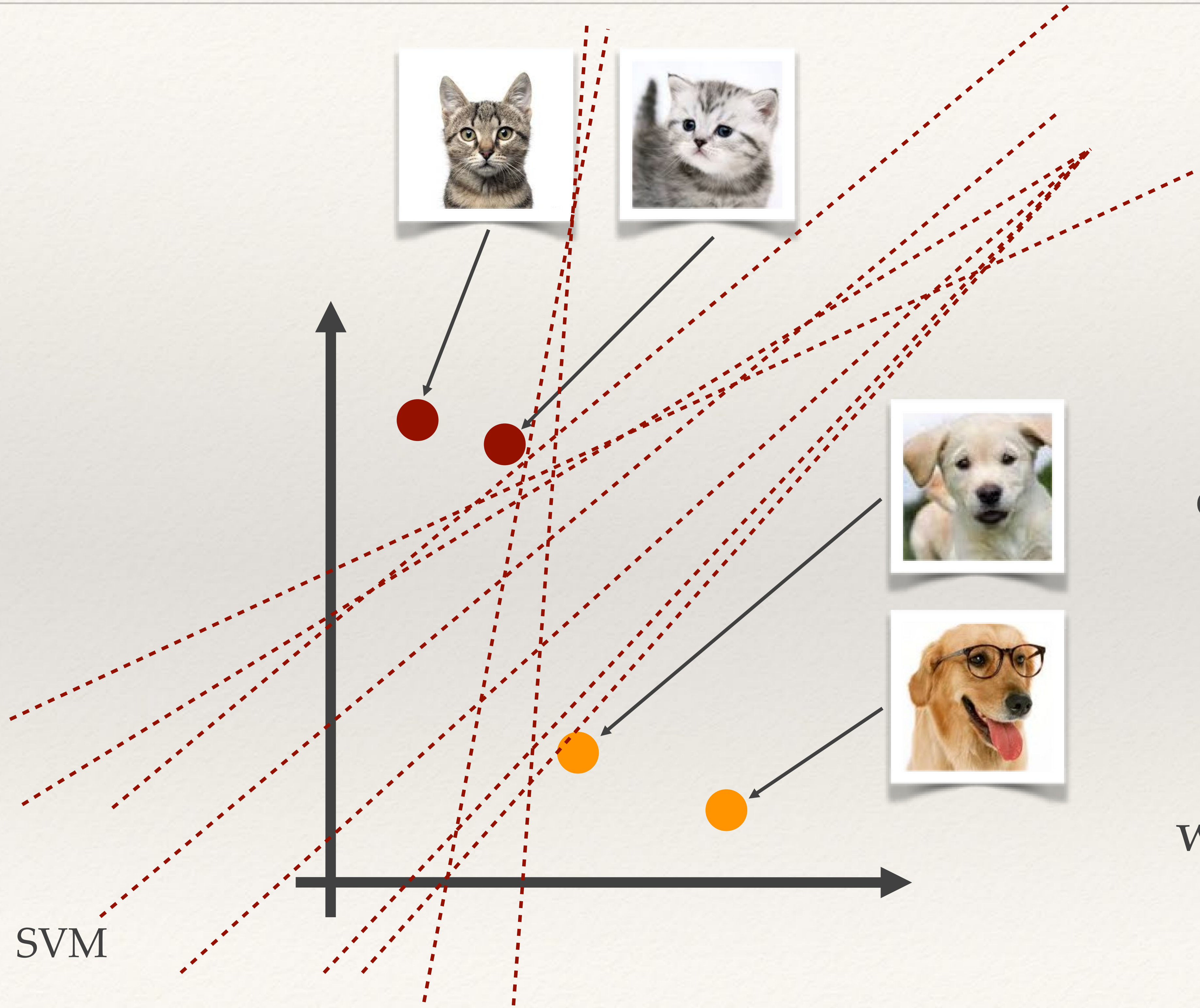
Cat or Dog?

Linear classifiers



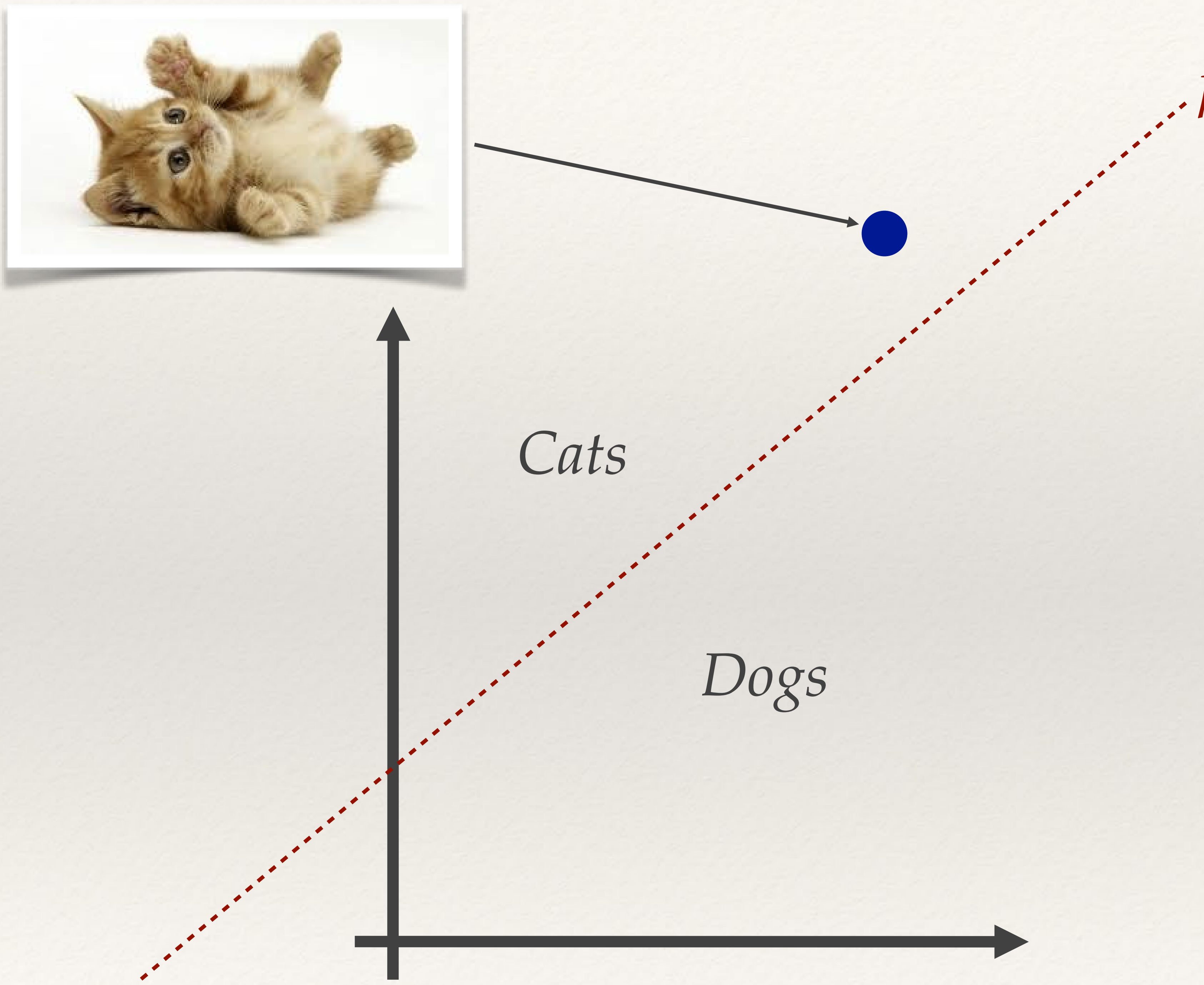
Linear classifiers try to learn a **hyperplane** that separates two classes in featurespace with **minimum error**

Linear classifiers



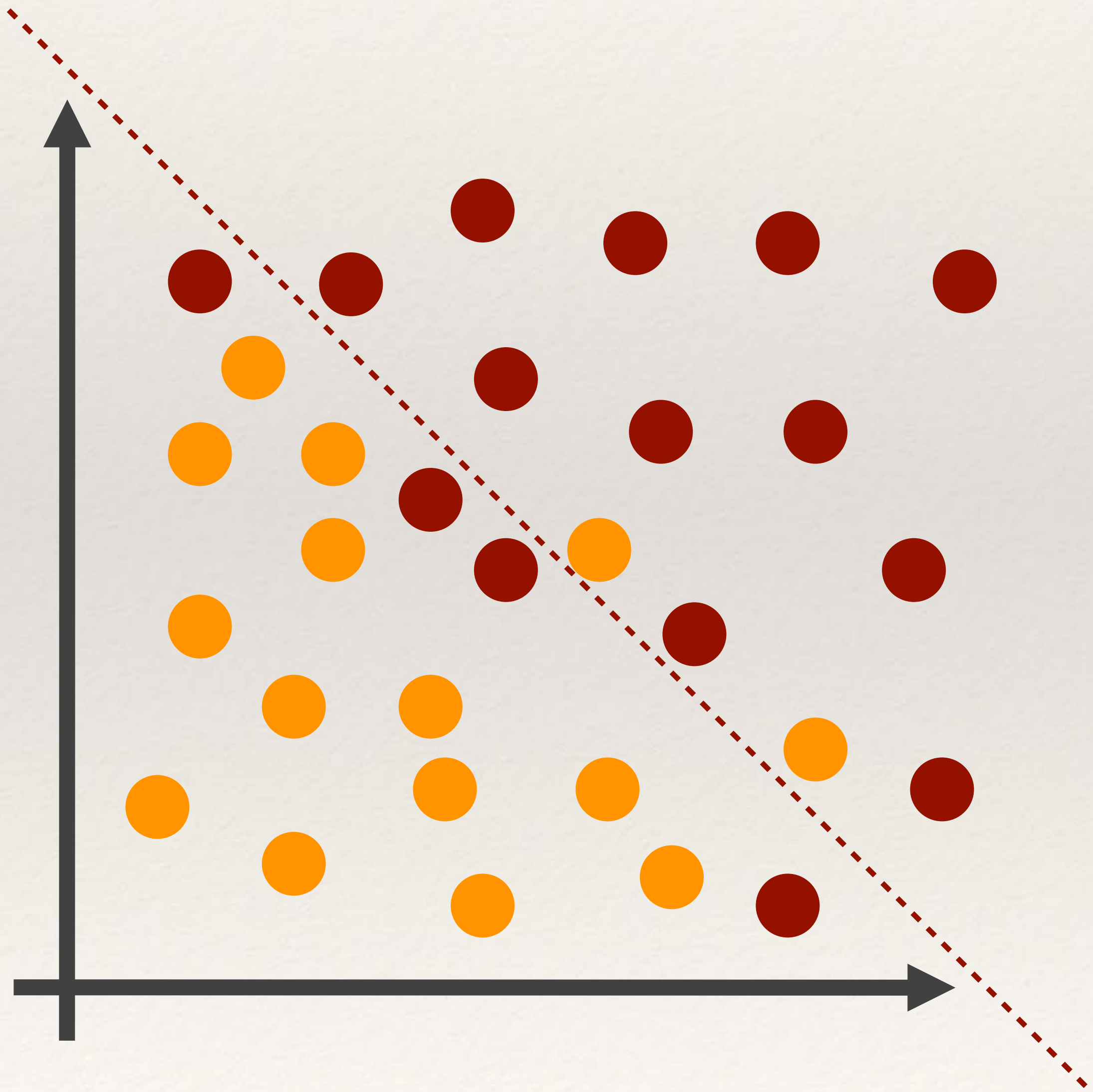
Lots of hyperplanes to choose from… different linear classification algorithms apply differing constraints when learning the classifier

Linear classifiers



To classify a new image, you just need to check what side of the hyperplane it is on

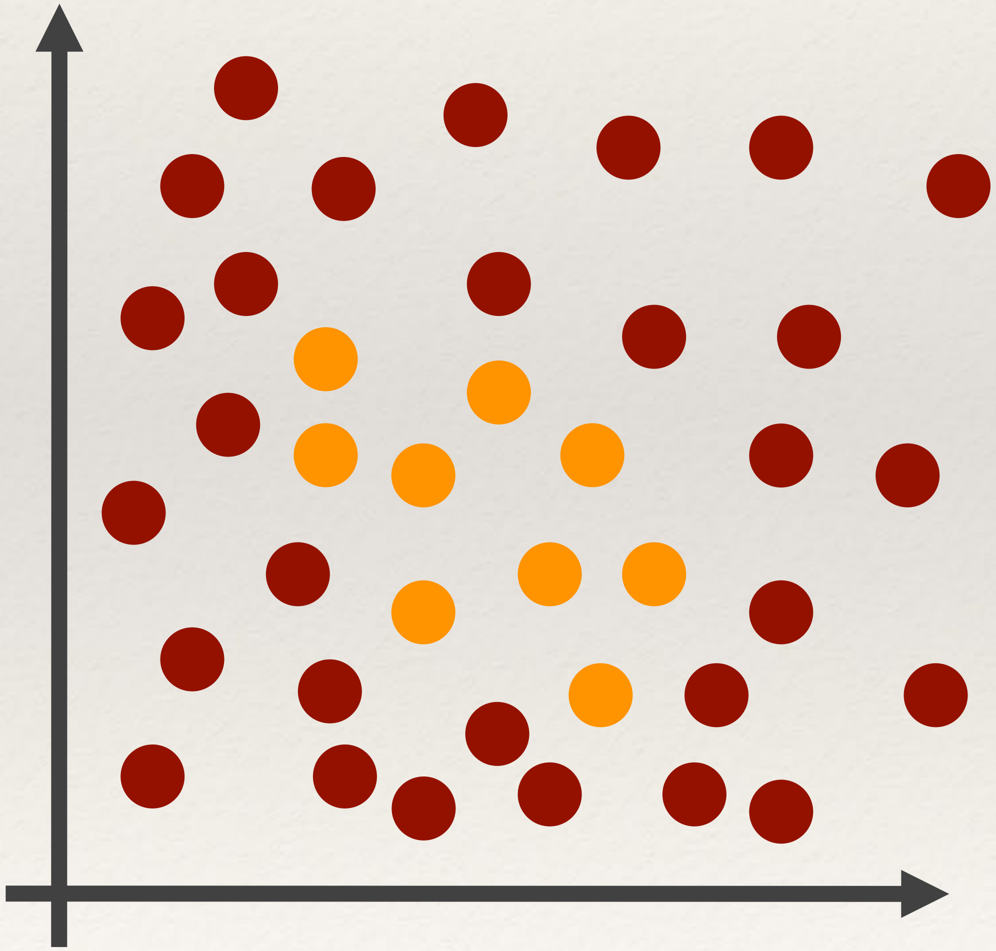
Non-linear binary classifiers



Linear classifiers work best when the data is linearly separable…

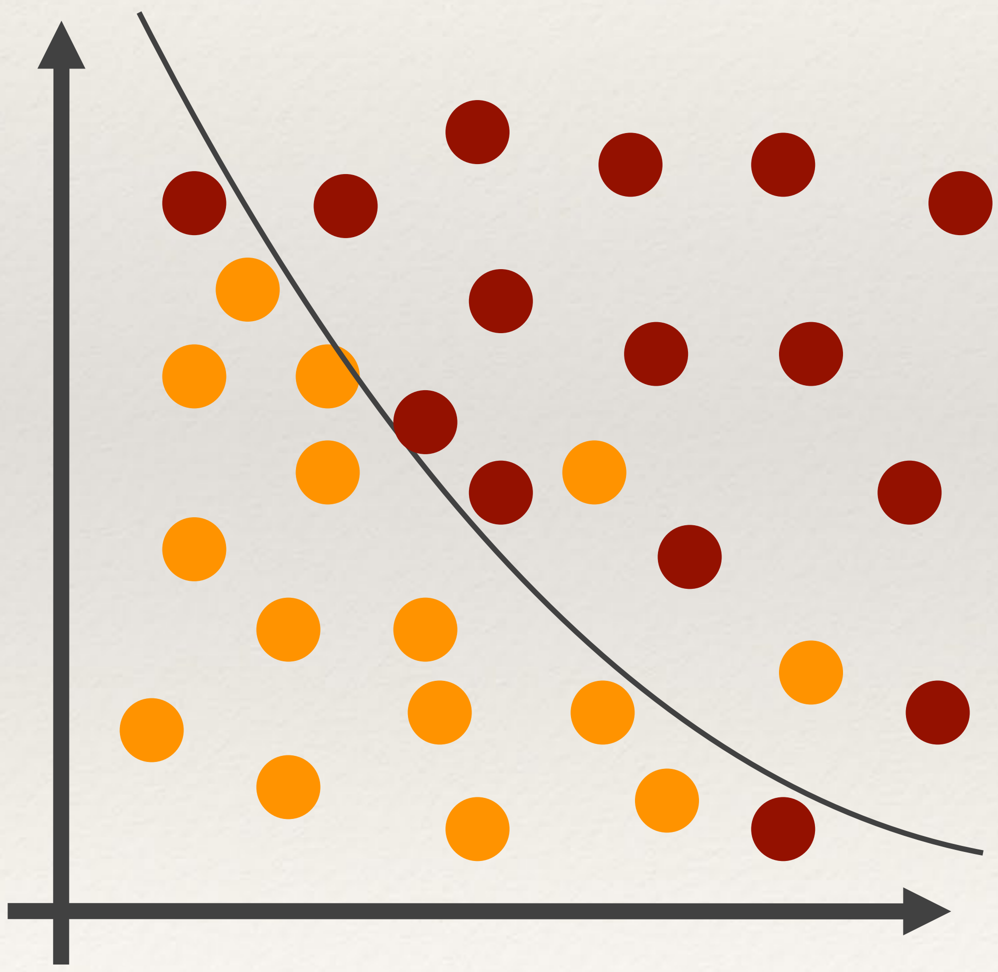
There can be outliers.
False Positive / False Negative
Which is more important?

Non-linear binary classifiers



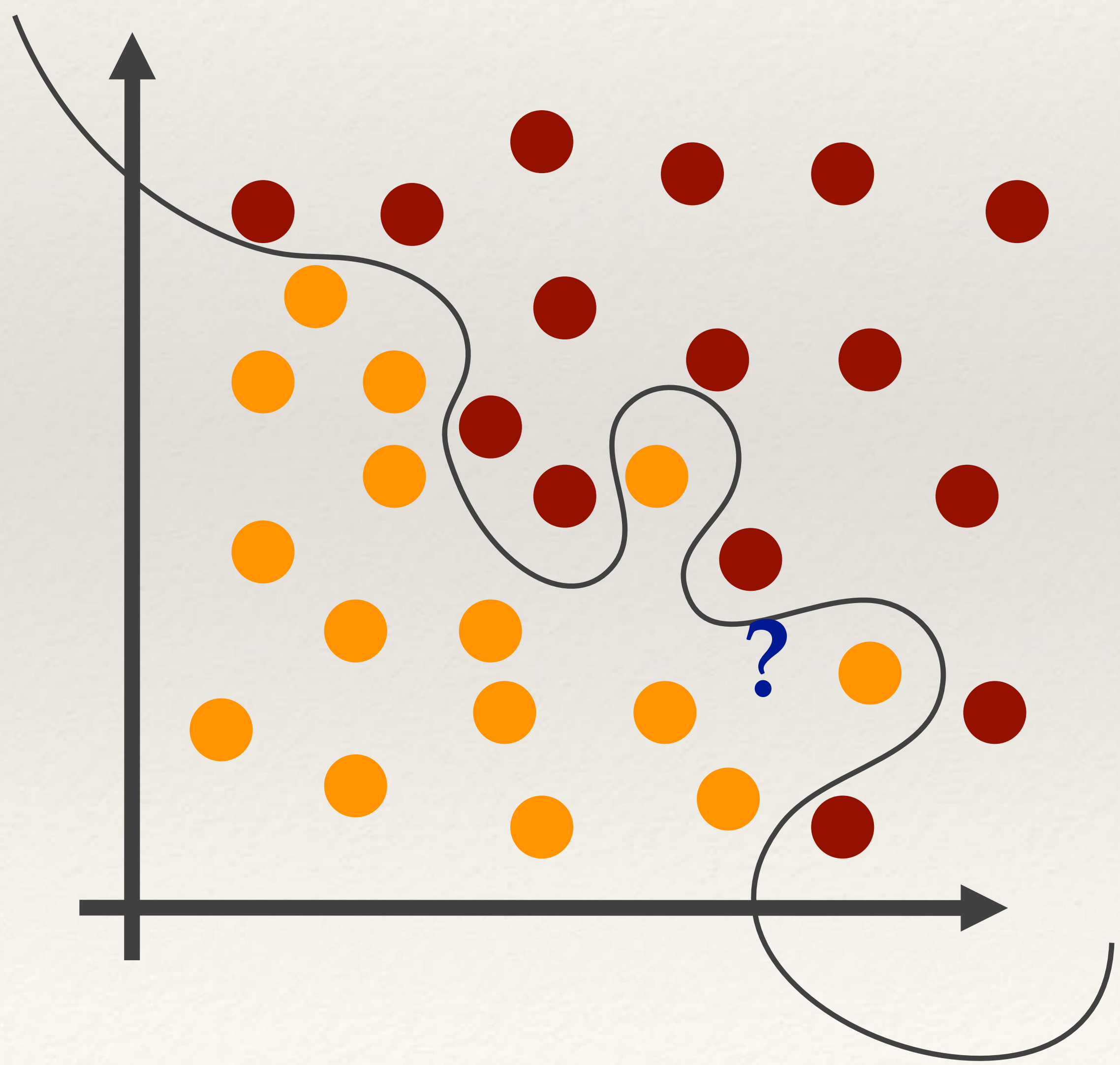
No hope for a linear classifier!

Non-linear binary classifiers



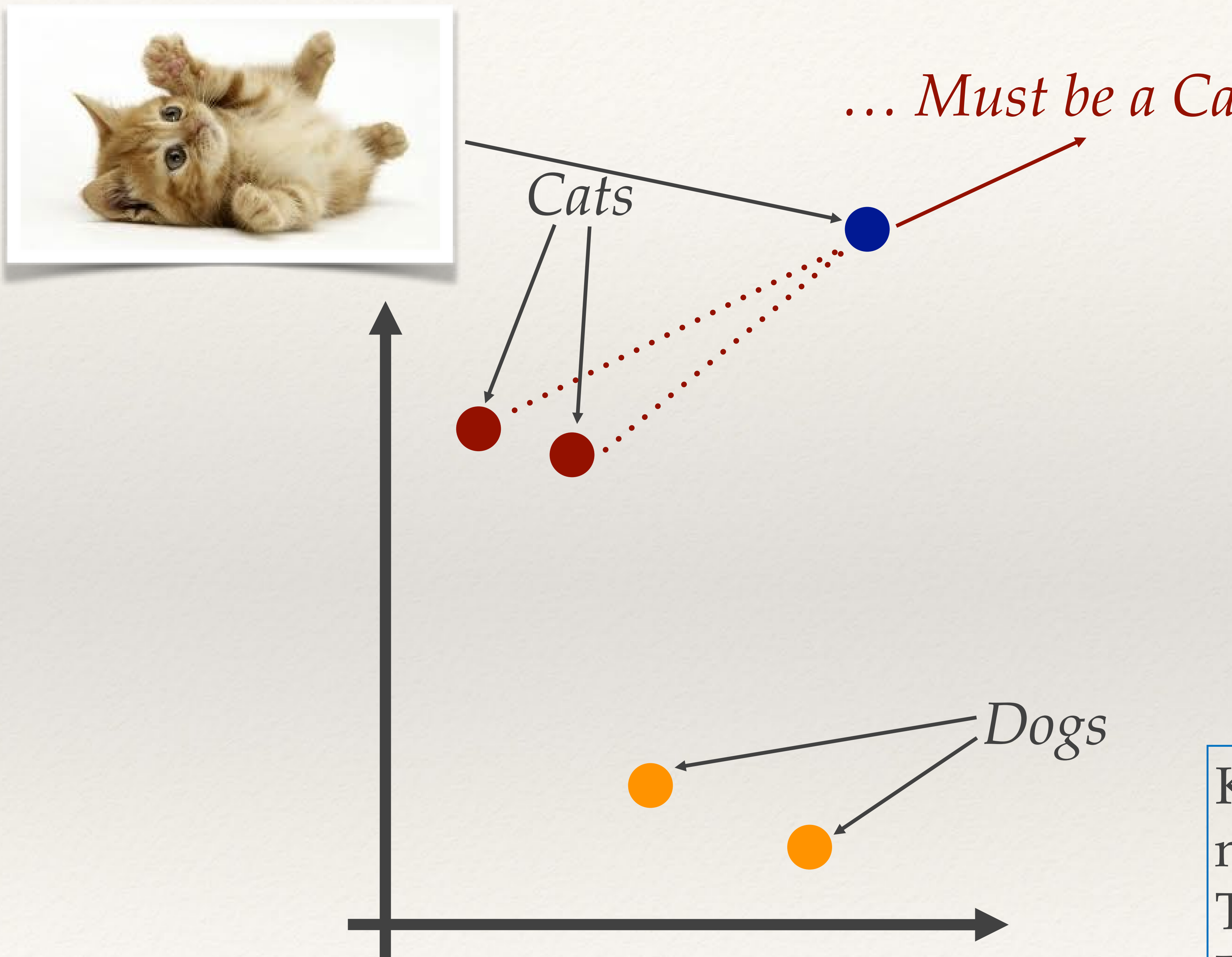
Non-linear binary classifiers, such as **Kernel Support Vector Machines** learn non-linear decision boundaries

Non-linear binary classifiers



Have to be careful… you might lose generality by overfitting

Multiclass classifiers: KNN



Assign class of unknown point based on majority class of *closest K* neighbours in feature space

K: even number is not a good idea.
Tie breaker:
Random or Nearest

KNN Classification Demo

- KNN interactive demo by Stanford Vision Lab
  - vision.stanford.edu/teaching/cs231n-demos/knn/

KNN Problems

- Computationally expensive if there are:
  - Lots of training examples
  - Many dimensions

Multiclass linear classifiers

- A linear classifier is by definition binary
  - So, how can we solve multiclass problems with linear classifiers?
    - One versus All (OvA)/One versus Rest (OvR)
      - one classifier per class
    - One versus One (OvO)
      - $K\,(K-1)\,/\,2$ classifiers
    - Check the confidences (distances) and choose the highest one.

Unsupervised Machine Learning:
Clustering

- Clustering aims to group data without any prior knowledge of what the groups should look like or contain.
- In terms of feature vectors, items with similar vectors should be grouped together by a clustering operation.
- Some clustering operations create overlapping groups; for now we're only interested in disjoint clustering methods that assign an item to a single group.

K-Means Clustering

- K-Means is a classic featurespace clustering algorithm for grouping data into *K* groups with each group represented by a *centroid*:
  - `The value of K is chosen`
  - `K initial cluster centres are chosen`
  - `Then the following process is performed iteratively until the centroids don’t move between iterations:`
    - `Each point is assigned to its closest centroid`
    - `The centroid is recomputed as the mean of all the points assigned to it. If the centroid has no points assigned it is randomly re-initialised to a new point.`
  - `The final clusters are created by assigning all points to their nearest centroid.`

K-Means Clustering Demo

- by Middle East Technical University
  - http://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

Summary (1)

- Machine learning
  - Standard way of training the pattern recognition system
  - Feature extraction
    - Transforms raw data into feature vectors of some fixed number of elements
  - Distance and similarity measures
    - Feature vectors can be compared by measuring distance
    - L1 / L2 distance, Cosine similarity (relative direction)

Summary (2)

- Supervised learning: classification
  - Use pre-labelled training data to learn how to assign class labels to vectors
  - A linear classifier tries to learn a hyperplane that separates the feature space in two (binary classifier)
  - Common linear classifier - Support Vector Machine (SVM)
  - Multiclass supervised classification - KNN
- Unsupervised machine learning: clustering
  - Learns to group data without prior knowledge of what the groups should look like
  - K-Means algorithm – iterative clustering represented by centroids
  - K-Means always converges, but not necessarily to the most optimal solution

Further reading and exercises

- **Further reading**
  - Mark's book (third edition) p.424-429 covers K-nearest-neighbours and some other approaches.
  - Wikipedia has good entries for:
    - SVM: https://en.wikipedia.org/wiki/Support-vector_machine
    - KNN: http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
    - K-Means: http://en.wikipedia.org/wiki/K-means_clustering
- **Practical exercises with OpenCV**
  - OpenIMAJ tutorials for K-means and KNN
  - SVM: https://docs.opencv.org/4.5.5/d1/d73/tutorial_introduction_to_svm.html
  - KNN: https://docs.opencv.org/4.5.5/d5/d26/tutorial_py_knn_understanding.html
  - K-Means: https://docs.opencv.org/4.5.5/d1/d5c/tutorial_py_kmeans_opencv.html

* *Acknowledgements:* Based on earlier Computer Vision lecture slides by Dr. Jon Hare.