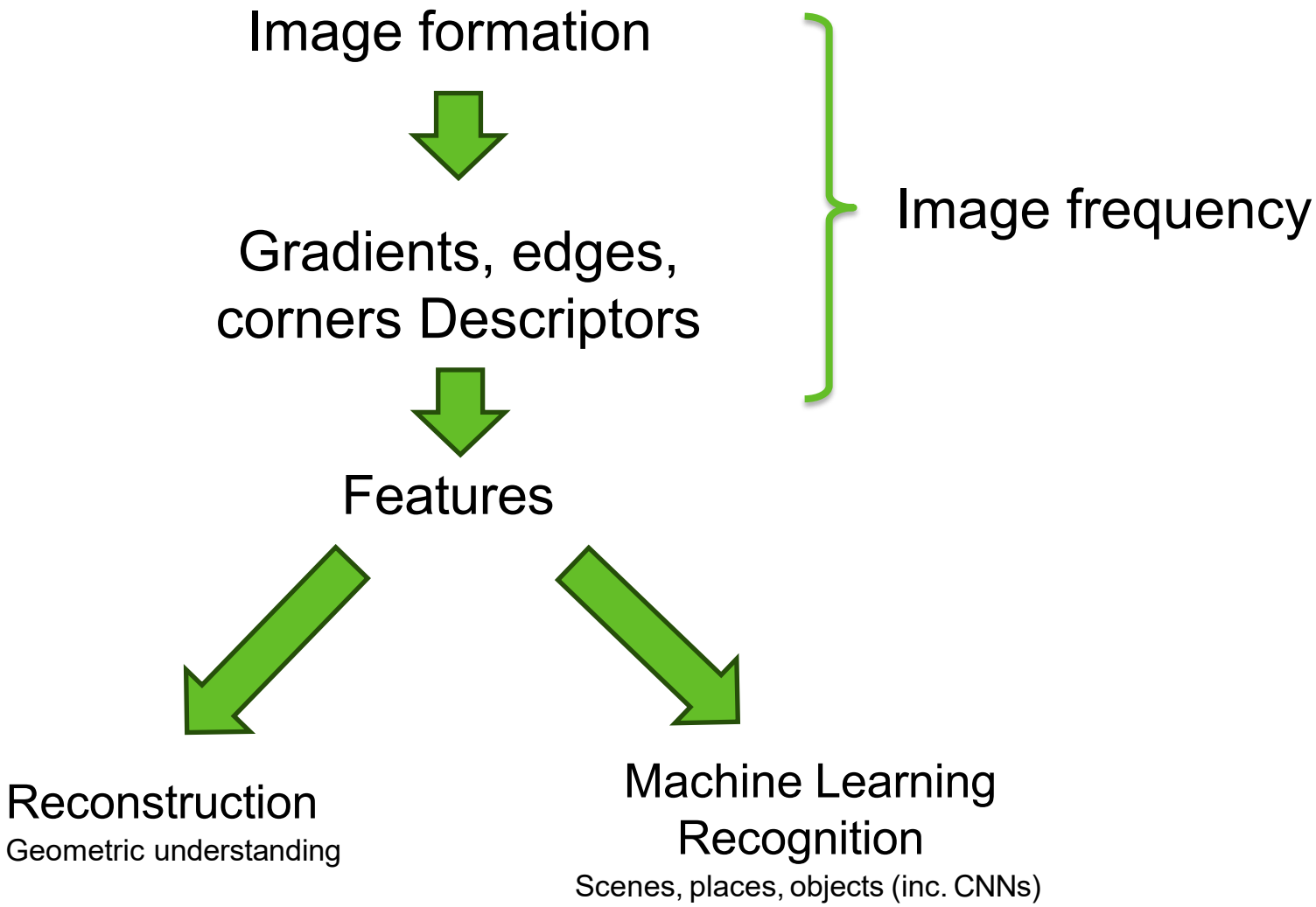
Introduction to Computer Vision

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lecture 14

Course Journey



Fundamental Equations of Computer Vision

1. Image Filtering $$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

2. Optical Flow $$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

3. Camera Geometry $$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R}\ \mathbf{t}\end{bmatrix}\mathbf{X} \qquad x^T F x' = 0$$

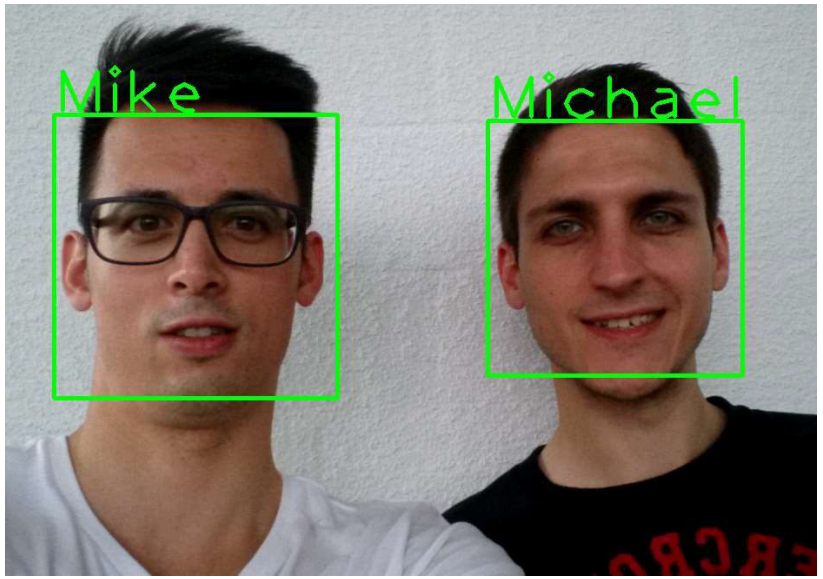
4. Machine Learning $$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2. \qquad y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

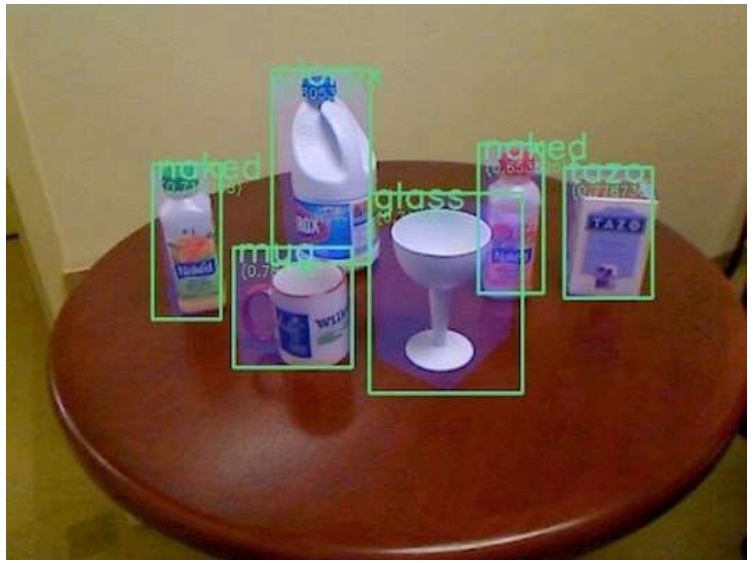
Machine Learning

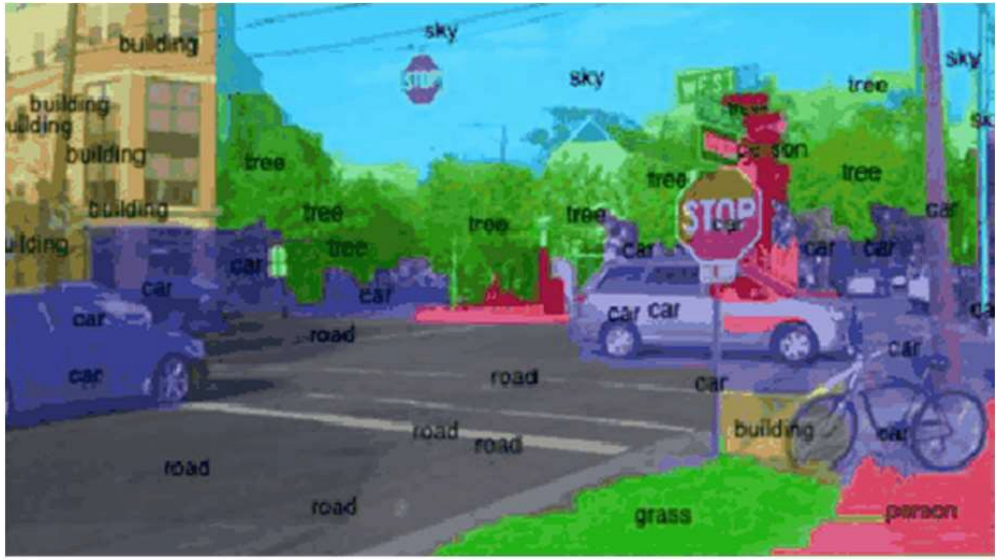
- This is a **brief** introduction!
- Take other courses for a more comprehensive introduction:
  - CSCI 470: Introduction to Machine Learning
  - SP TPS: Deep Learning

ML for Computer Vision

- Face Recognition
- Object Classification
- Scene Segmentation







Data, data, data!

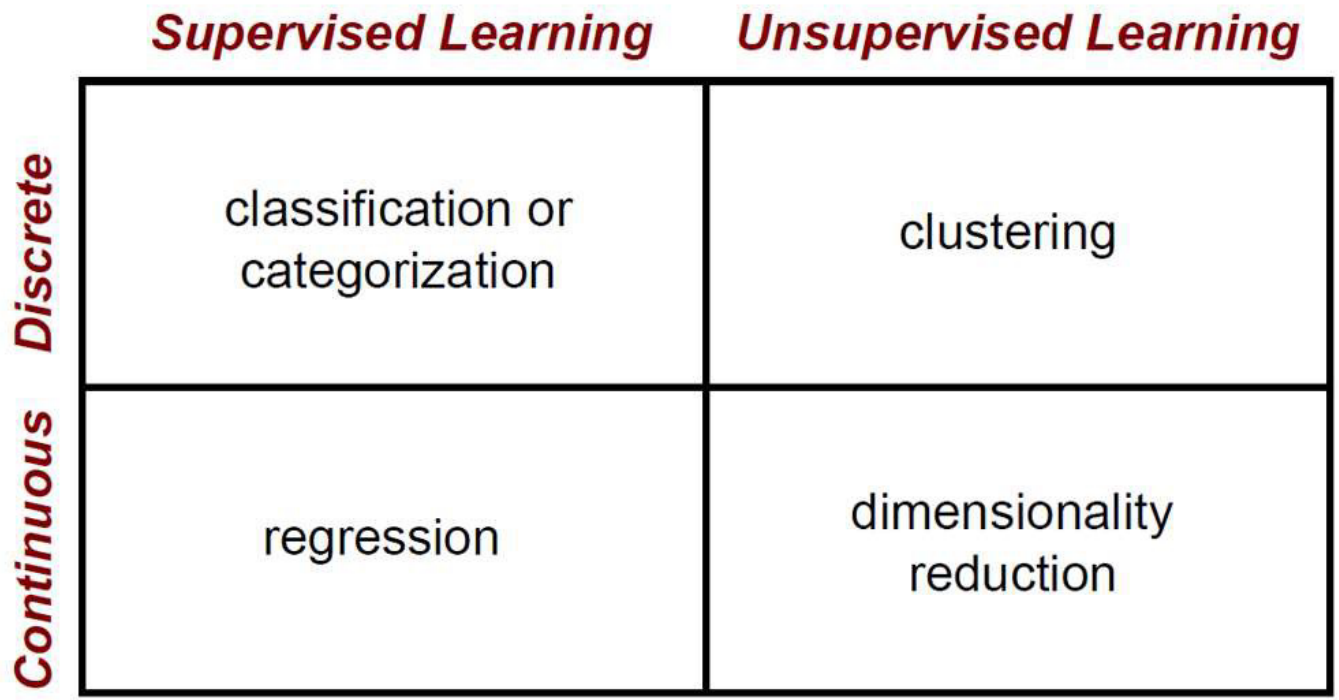
Peter Norvig

The Unreasonable Effectiveness of Data
(IEEE Intelligent Systems, 2009)

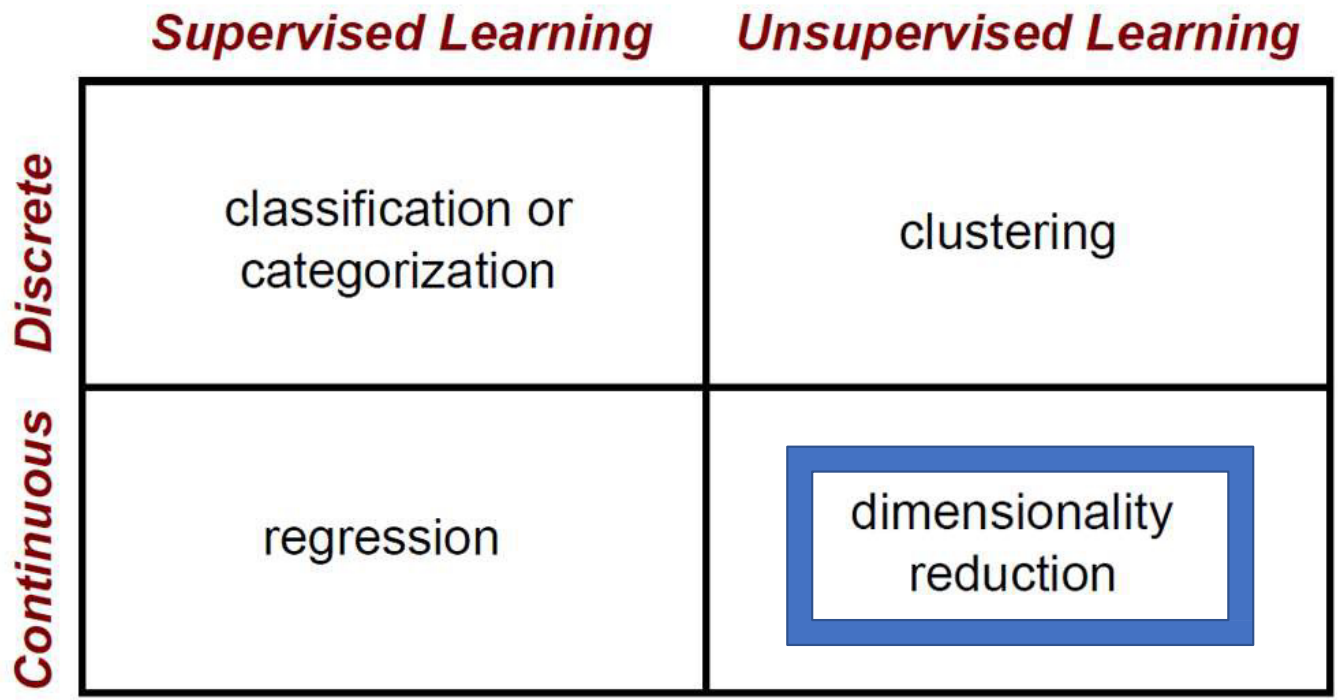
> *"... invariably, simple models and a lot of data trump more elaborate models based on less data"*

Norvig is a former Director of Research @ Google

Machine Learning Problems



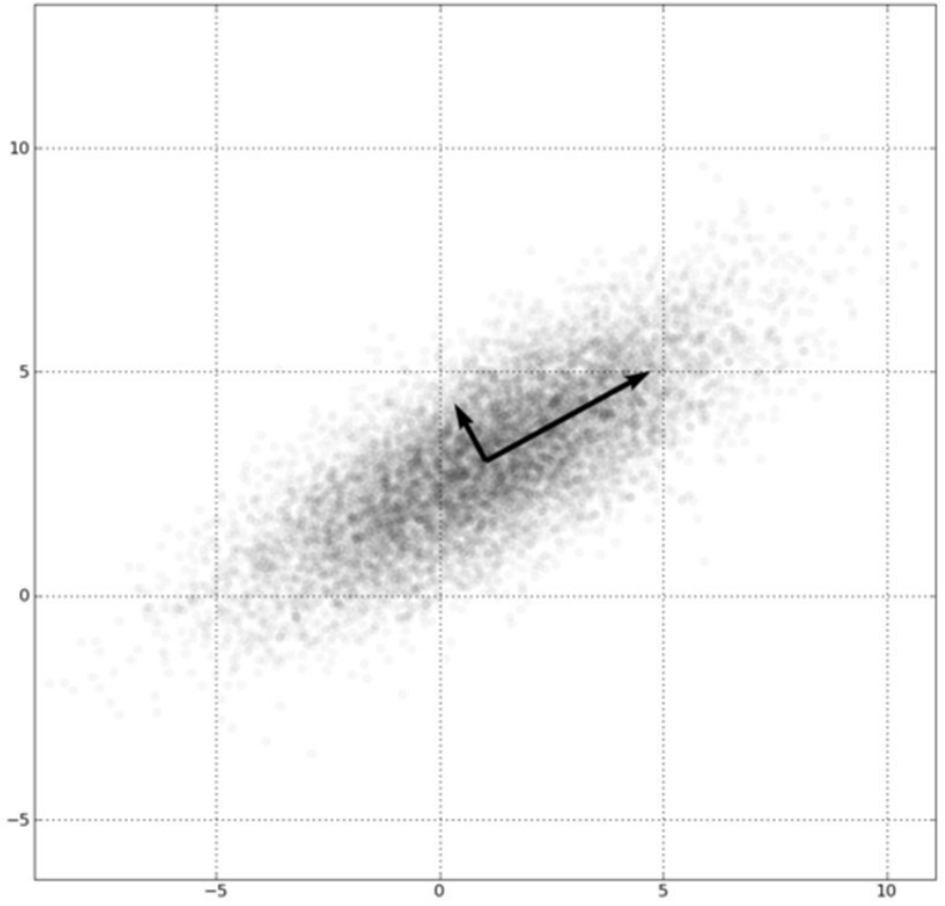
Machine Learning Problems



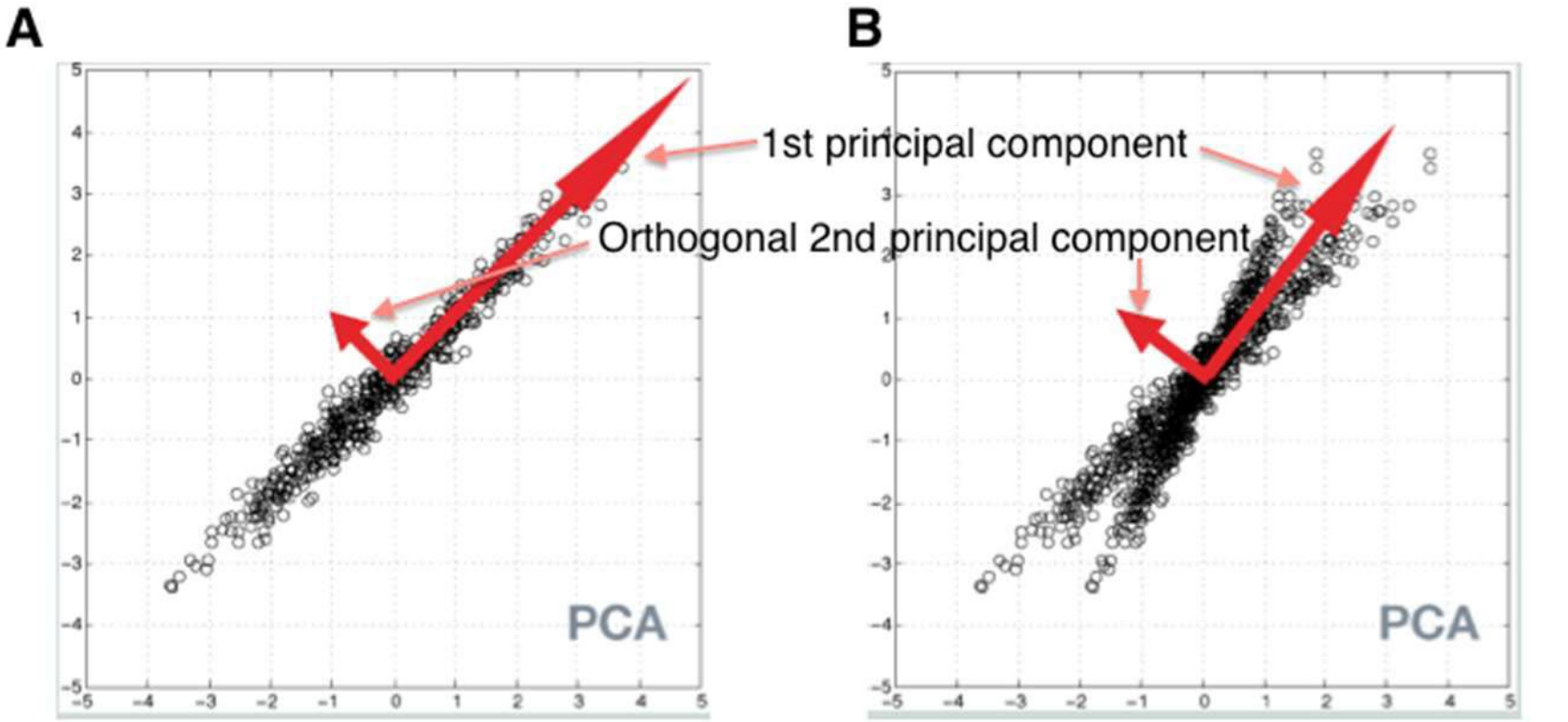
Dimensionality Reduction

- **PCA** (others include ICA, Isomap)

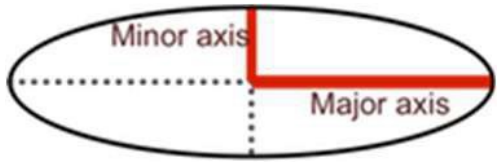
- **Principal Component Analysis (PCA)**
  - Creates a basis where the axes represent the dimensions of variance, from high to low.
  - Finds correlations in data dimensions to produce best possible lower-dimensional representation based on linear projections.
  - Subtract mean of data, compute eigen decomposition.
  - Eigenvectors are directions of variance; eigenvalues are magnitudes of variance.



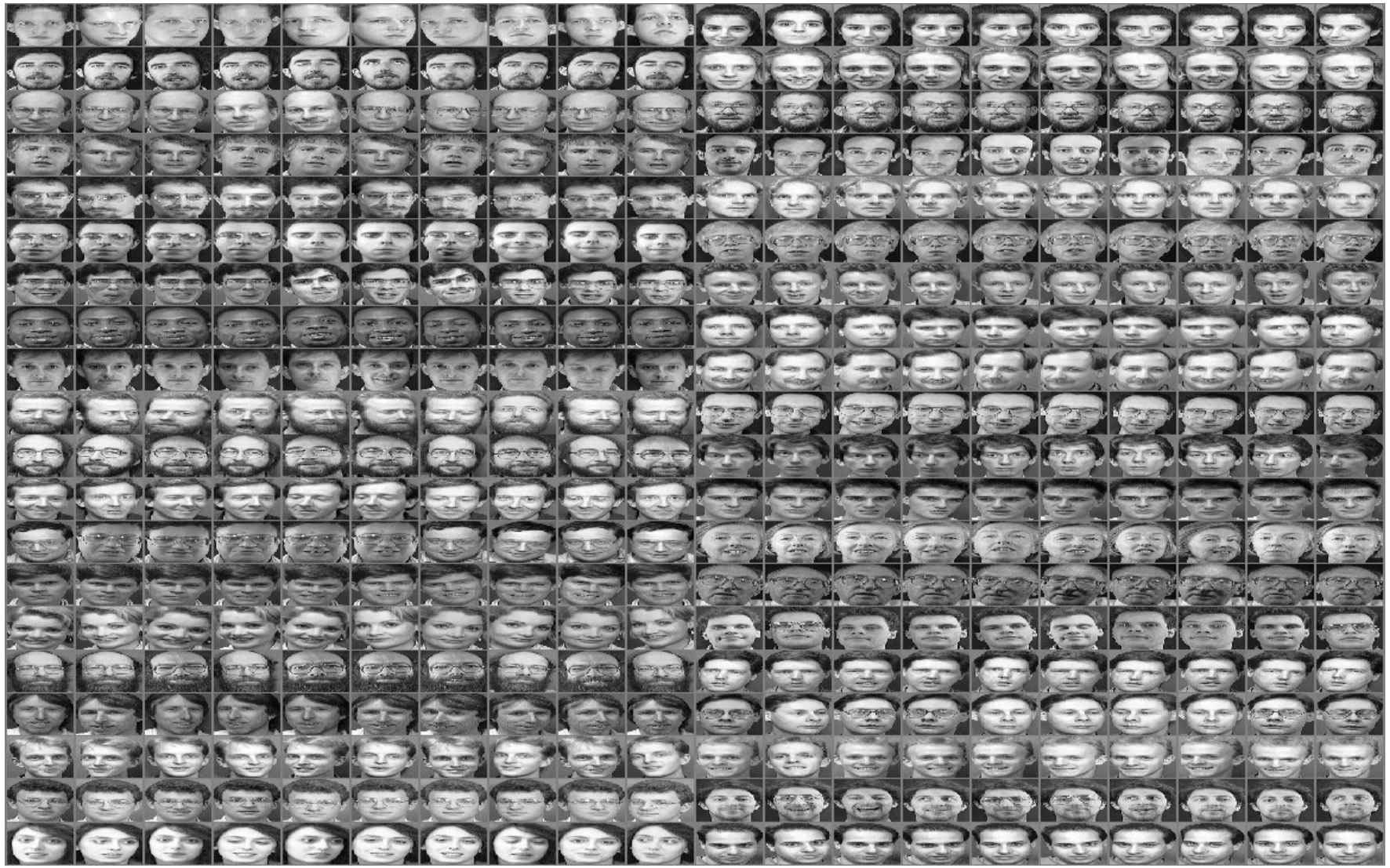
PCA



(Figure adapted from C. Beckmann, Oxford FMRIB)



Eigenfaces

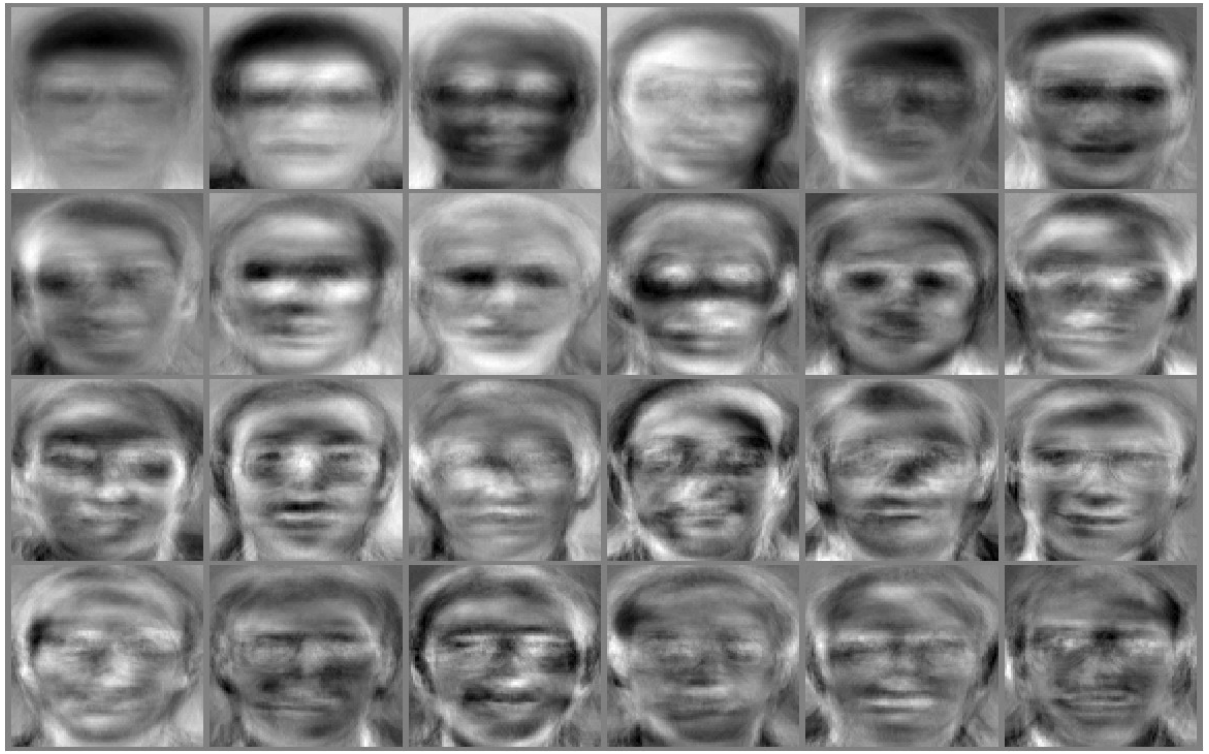


The ATT face database (formerly the ORL database), 10 pictures of 40 subjects each

Eigenfaces

Useful in face recognition

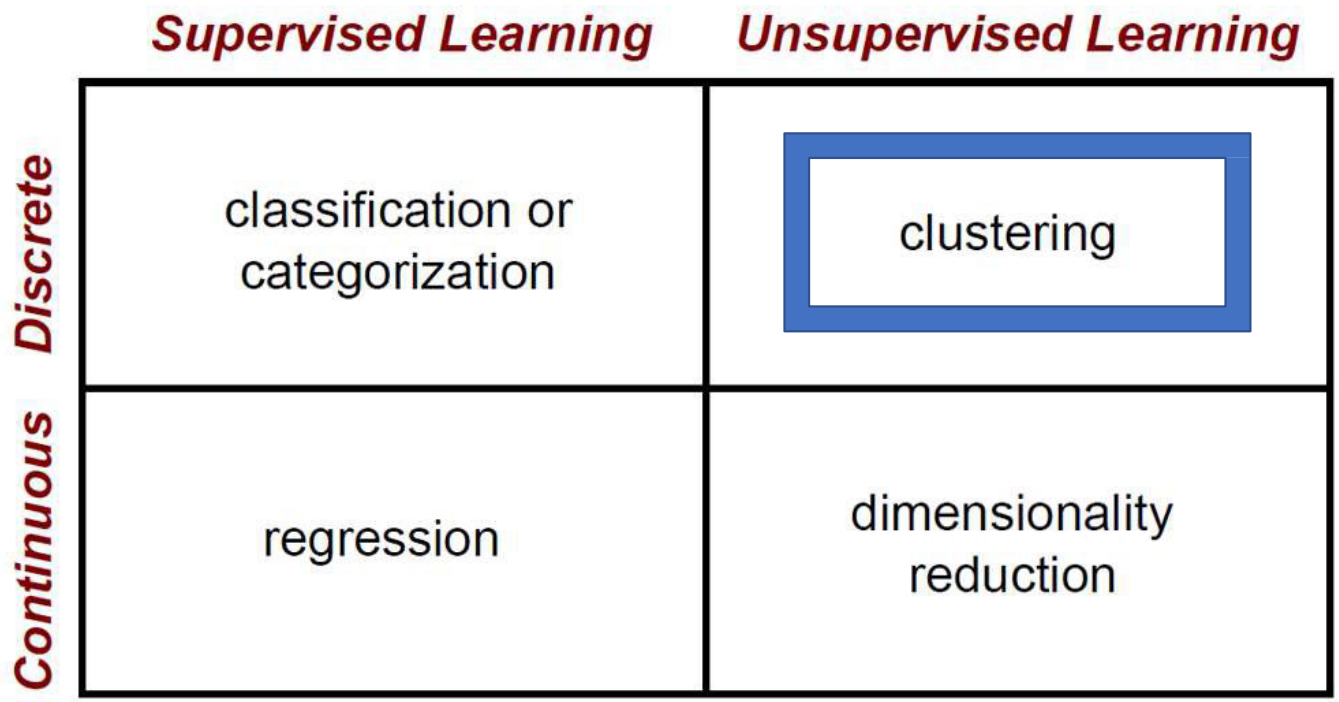
Mean face



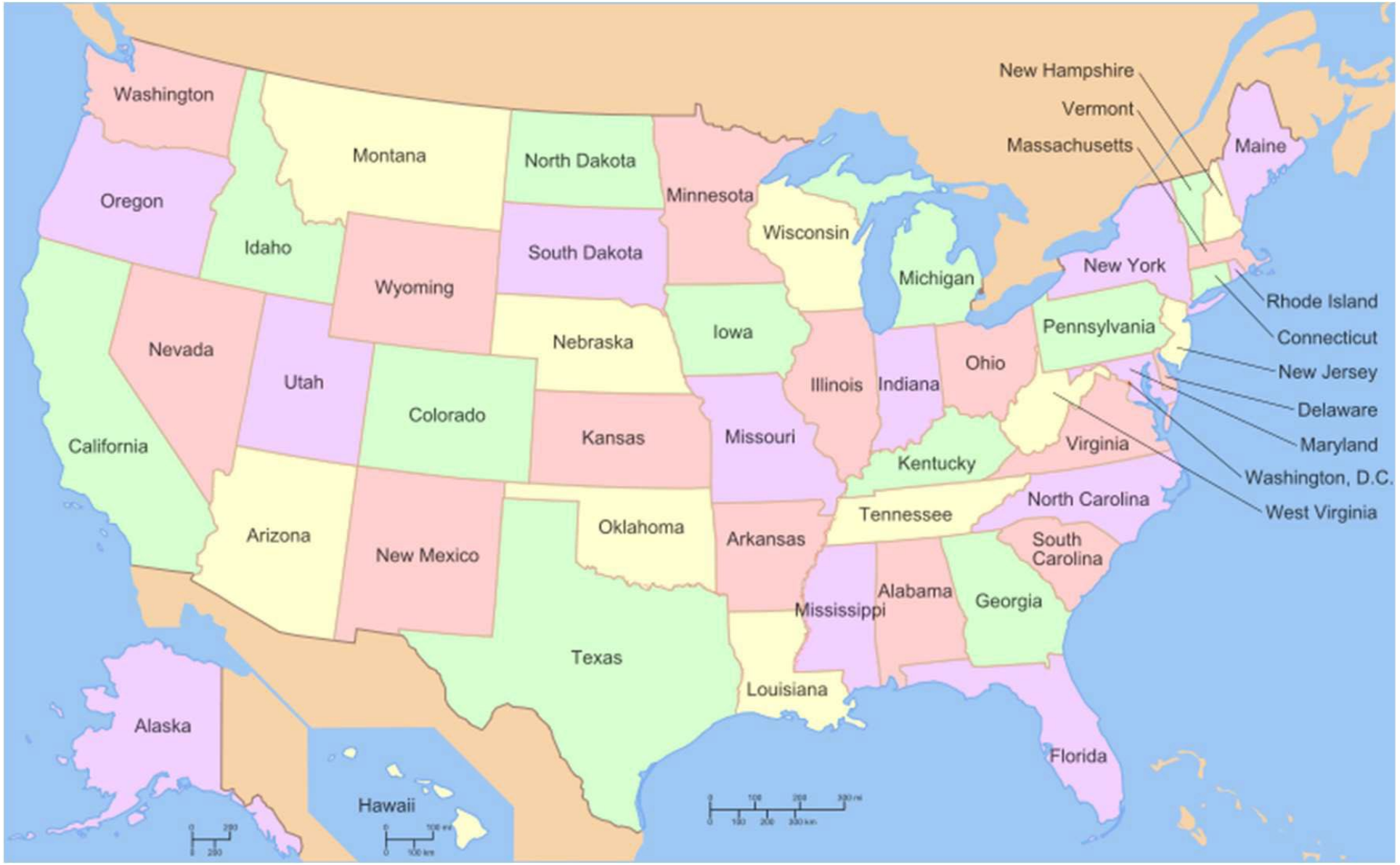
Basis of variance (eigenvectors)

M.Turk; A. Pentland (1991). *"Face recognition using eigenfaces" (PDF)*

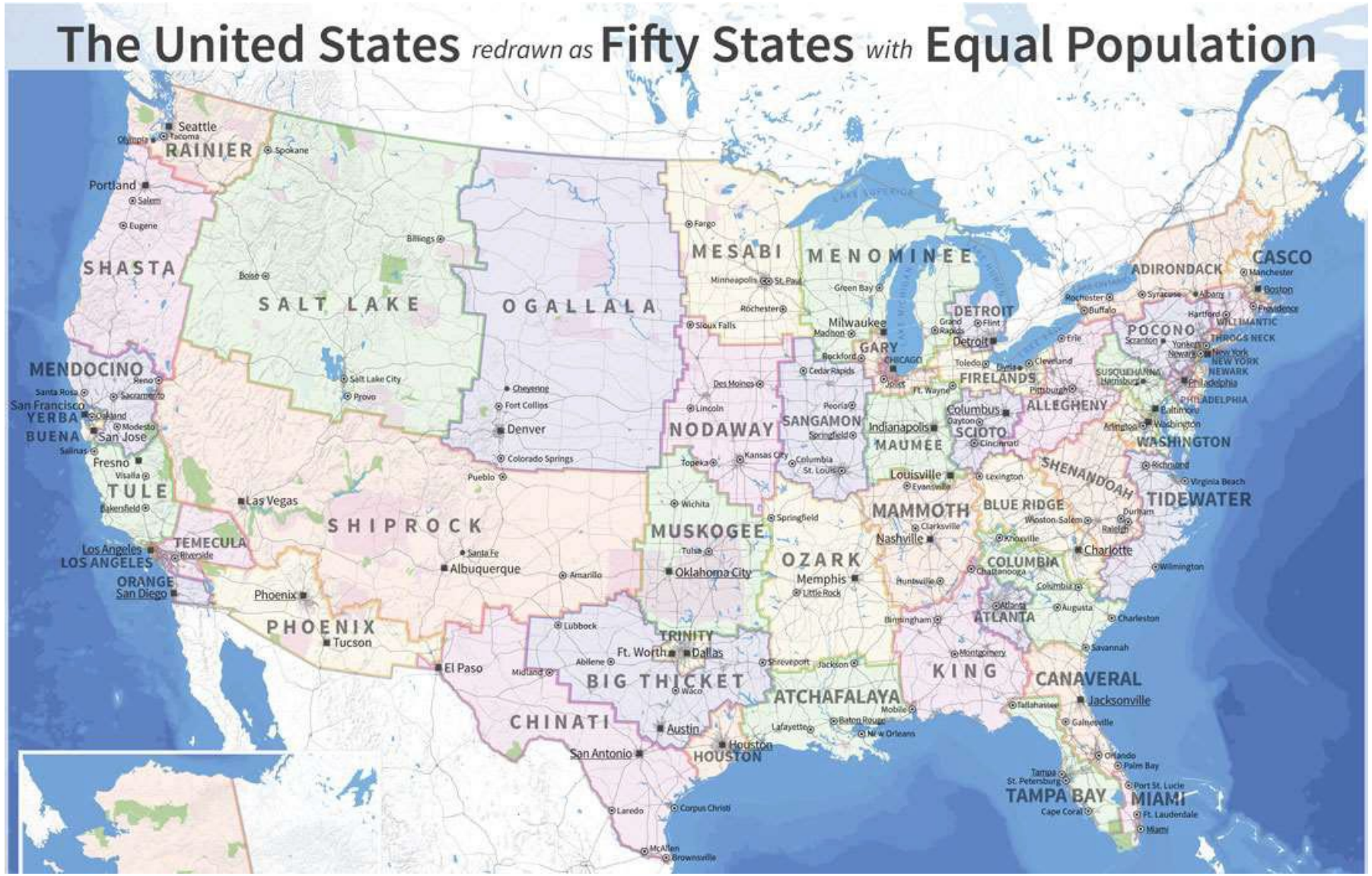
Machine Learning Problems



Clustering Example



Clustering Example

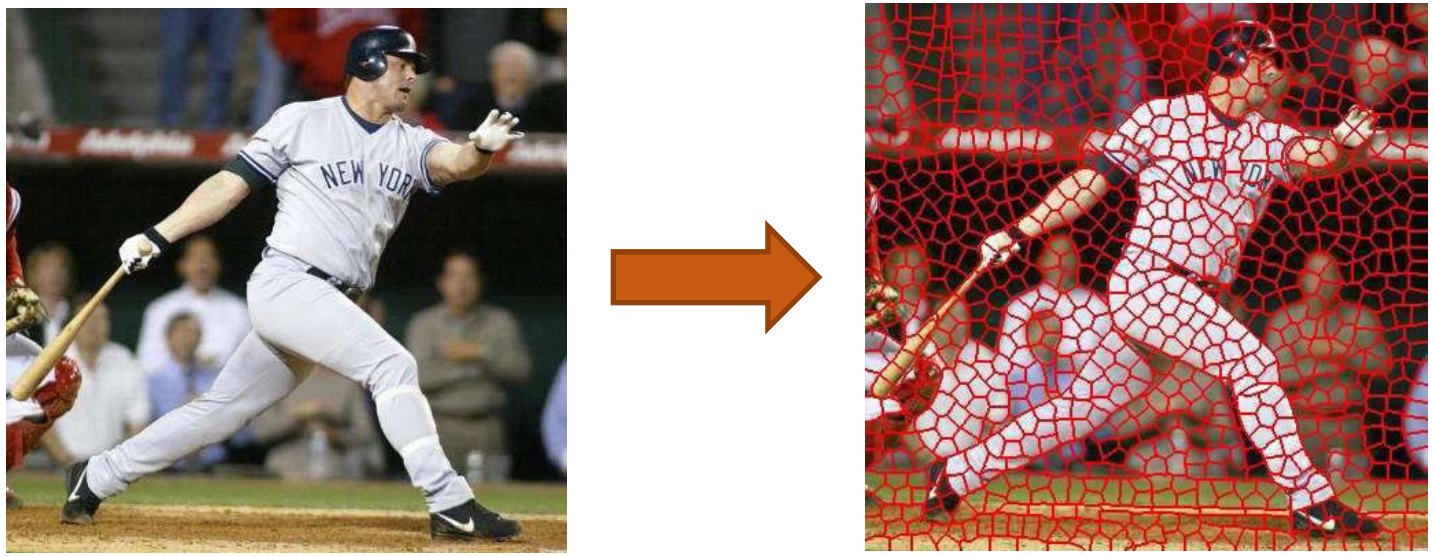


Clustering: Color Image Segmentation

Goal: Break up the image into meaningful or perceptually similar regions

Segmentation for Efficiency or Better Features

Efficient graph-based image segmentation [Felzenszwalb and Huttenlocher 2004]



Superpixel [Shi and Malik 2001]

Why do we cluster?

- **Summarizing data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster number
- **Counting**
  - Histograms of texture, color, SIFT vectors
- **Segmentation**
  - Separate the image into different regions
- **Prediction**
  - Images in the same cluster may have the same labels

Challenges

Group together similar 'points' and represent them with a single token.

Key Challenges:

1) What makes two points/images/patches similar?
2) How do we compute an overall grouping from pairwise similarities?

Clustering Techniques

- **K-means**
  - –Iteratively re-assign points to the nearest cluster center
- **Agglomerative clustering**
  - –Start with each point as its own cluster and iteratively merge the closest clusters
- **Mean-shift clustering**
  - –Estimate modes of pdf
- **Spectral clustering**
  - –Split the nodes in a graph based on assigned links with similarity weights

K-means algorithm

Algorithm

1. Randomly select K centers
2. Assign each point to nearest center
3. Compute new center (mean) for each cluster

Repeat steps 2-3

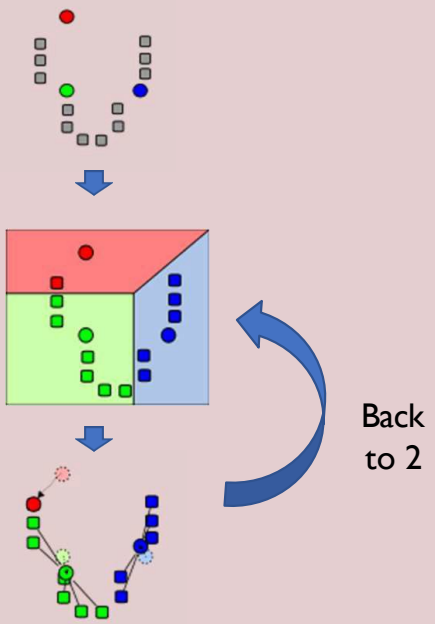


Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means

Algorithm

1. Initialize cluster centers: $\mathbf{c}^0$; t=0

2. Assign each point to the closest center

$$\boldsymbol{\delta}^t = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i^{t-1} - \mathbf{x}_j \right)^2$$

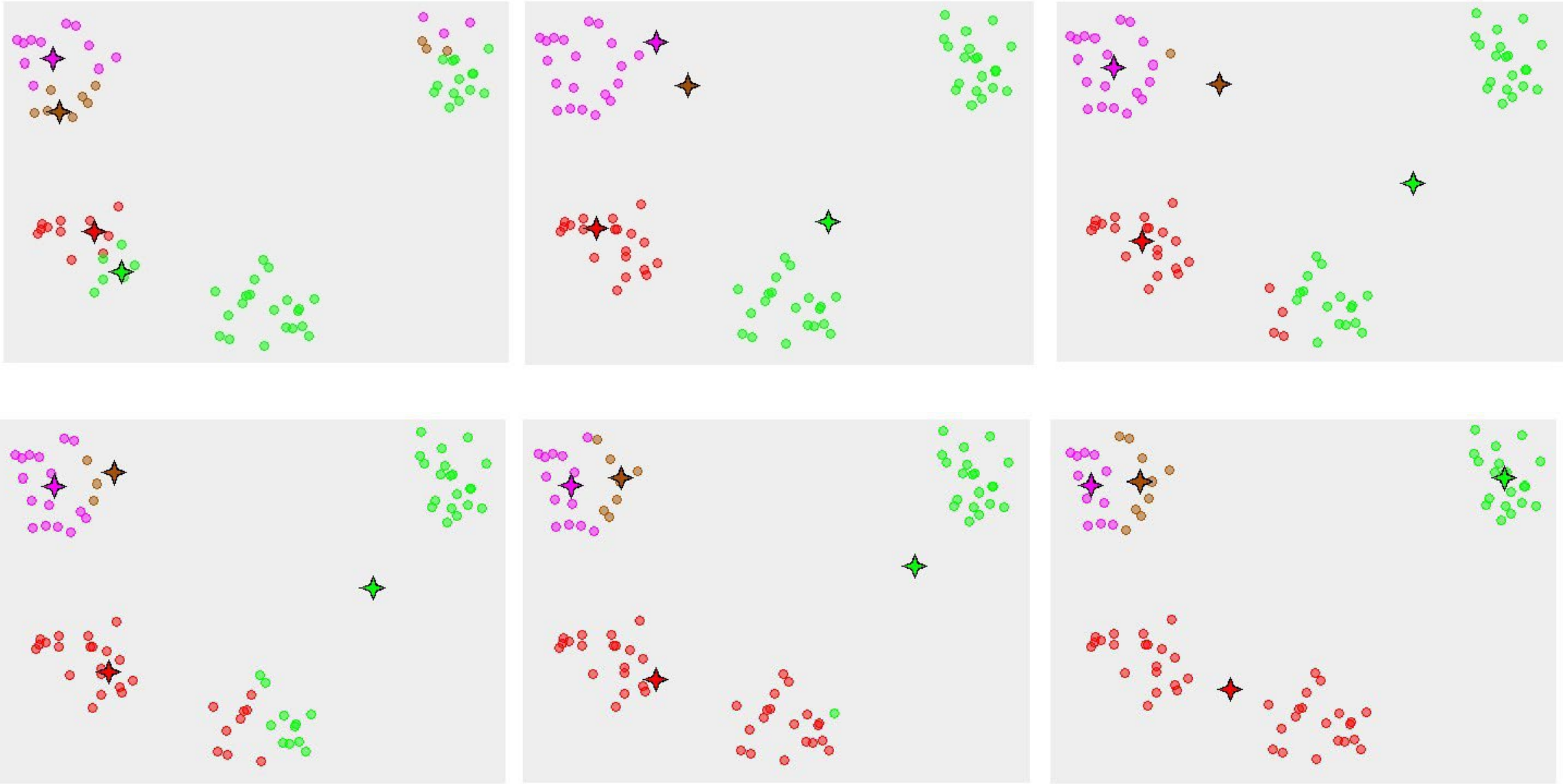
3. Update cluster centers as the mean of the points

$$\mathbf{c}^t = \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t \mathbf{x}_j$$

4. Repeat 2-3 until no points are re-assigned (t=t+1)

Derek Hoiem

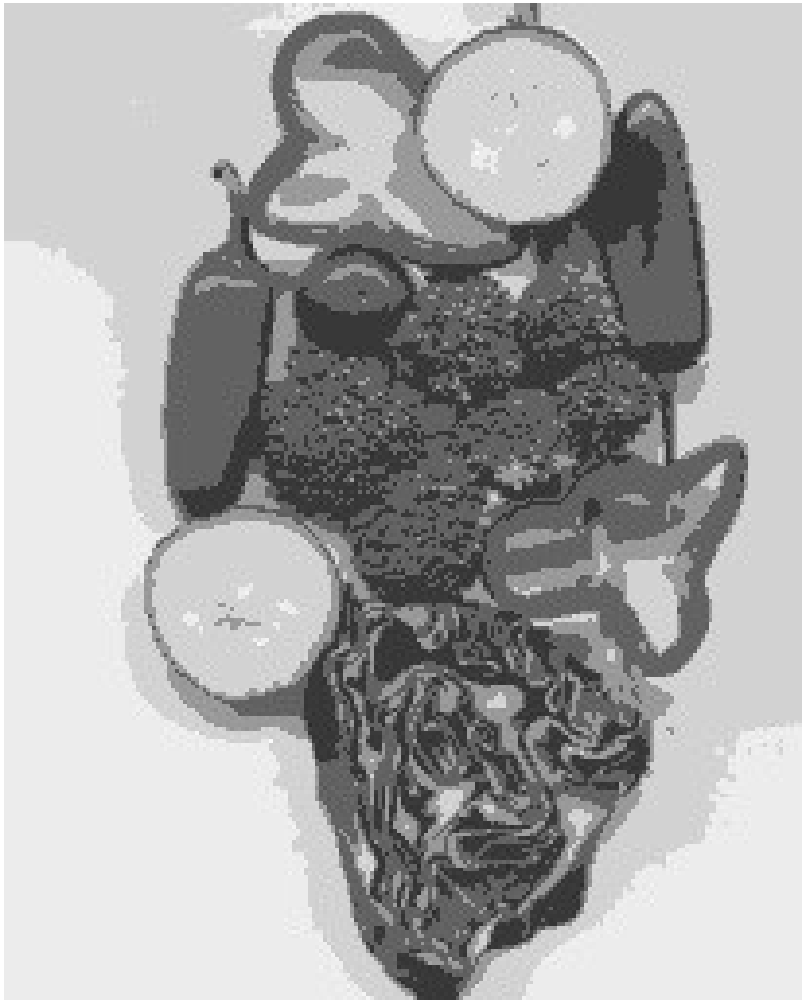
K-means Convergence



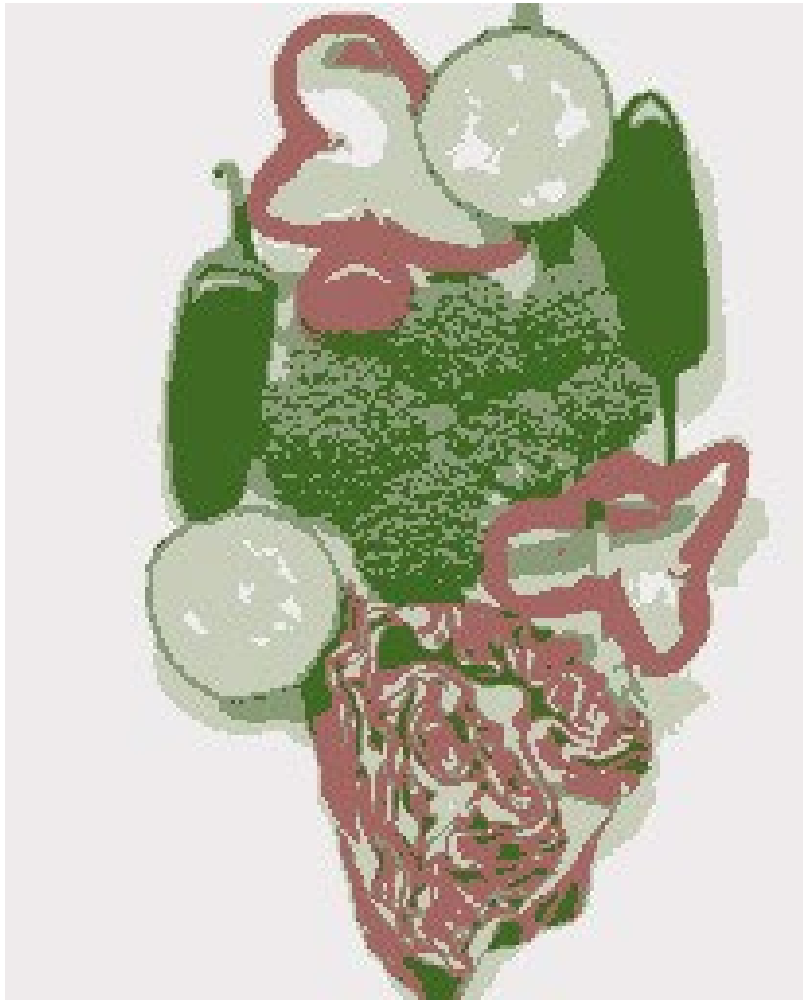
K-means Clustering with Pixel Values

Image

Clusters on intensity



Clusters on color



Additional Slide: How to initialize the clusters?

k-means++ initialization

- Make the initial cluster centers span the space

1. Choose one center uniformly at random from all data points.
2. For each point x, compute the distance $D(x)$ between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Proceed using standard *k*-means clustering.

Wikipedia / Arthur and Vassilvitskii

Additional Slide: How to evaluate clusters?

- Generative
  - How well are points reconstructed from the clusters?

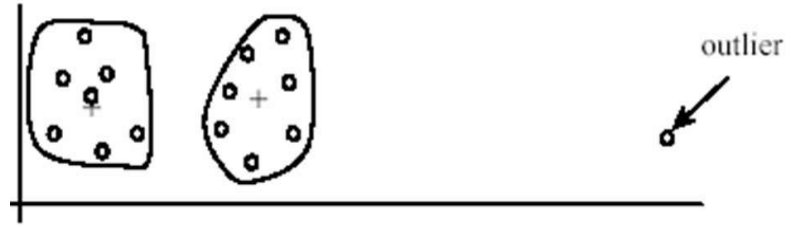
- Discriminative
  - How well do the clusters correspond to labels?
    - Purity
  - Note: unsupervised clustering does not aim to be discriminative

K-means: Design Choices

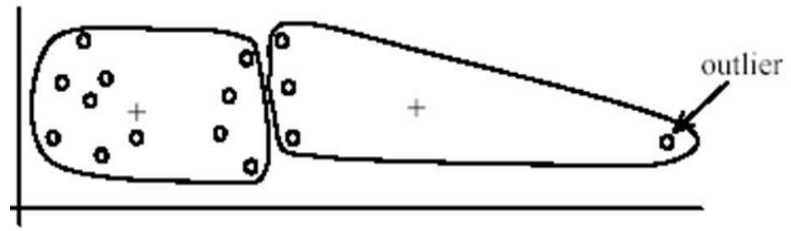
- **Initialization**
  - Randomly select K points as initial cluster center
  - Or greedily choose K points to minimize residual
  - Pick K given known application and expected performance; test on known data
- **Distance** measures
  - Traditionally Euclidean, could be others
- **Optimization**
  - Will converge to a *local minimum*
  - May want to perform multiple restarts

K-Means pros and cons

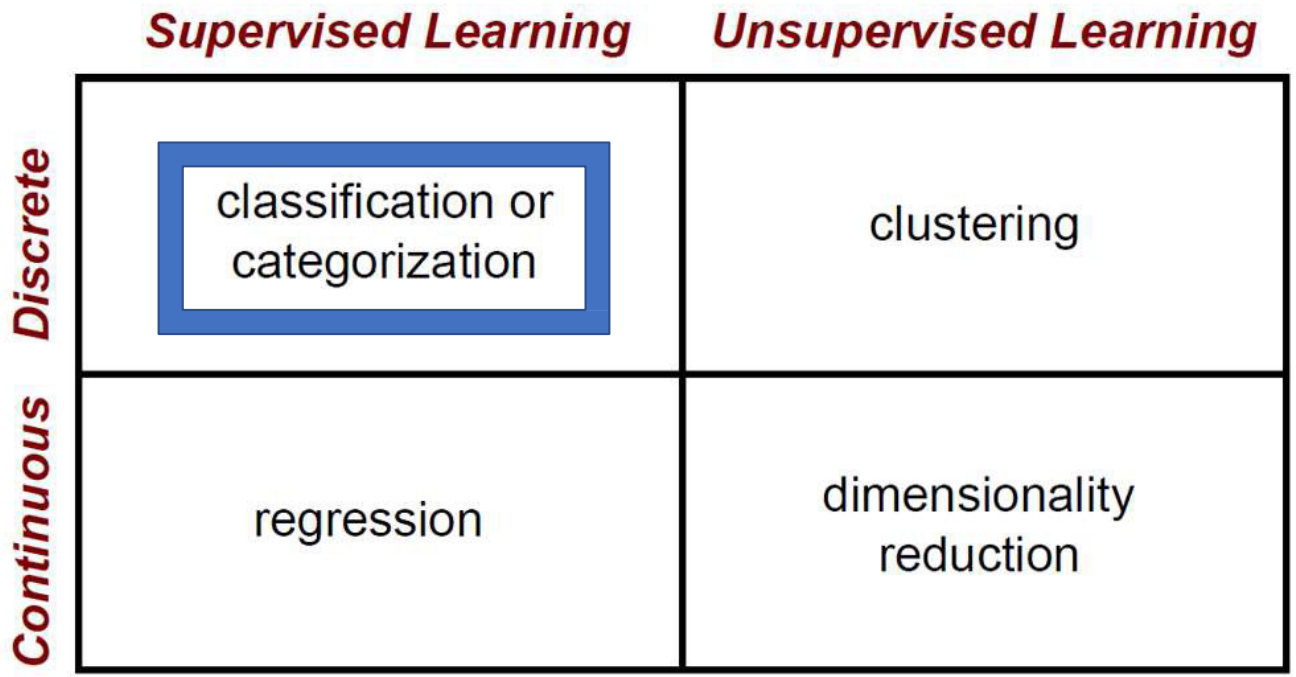
- **Pros**
  - Finds cluster centers that minimize conditional variance (good representation of data)
  - Simple and fast*
  - Easy to implement
- **Cons**
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)
  - *Can be slow: each iteration is O(KNd) for N d-dimensional points
- **Usage**
  - Cluster features to build visual dictionaries



(B): Ideal clusters



Machine Learning Problems

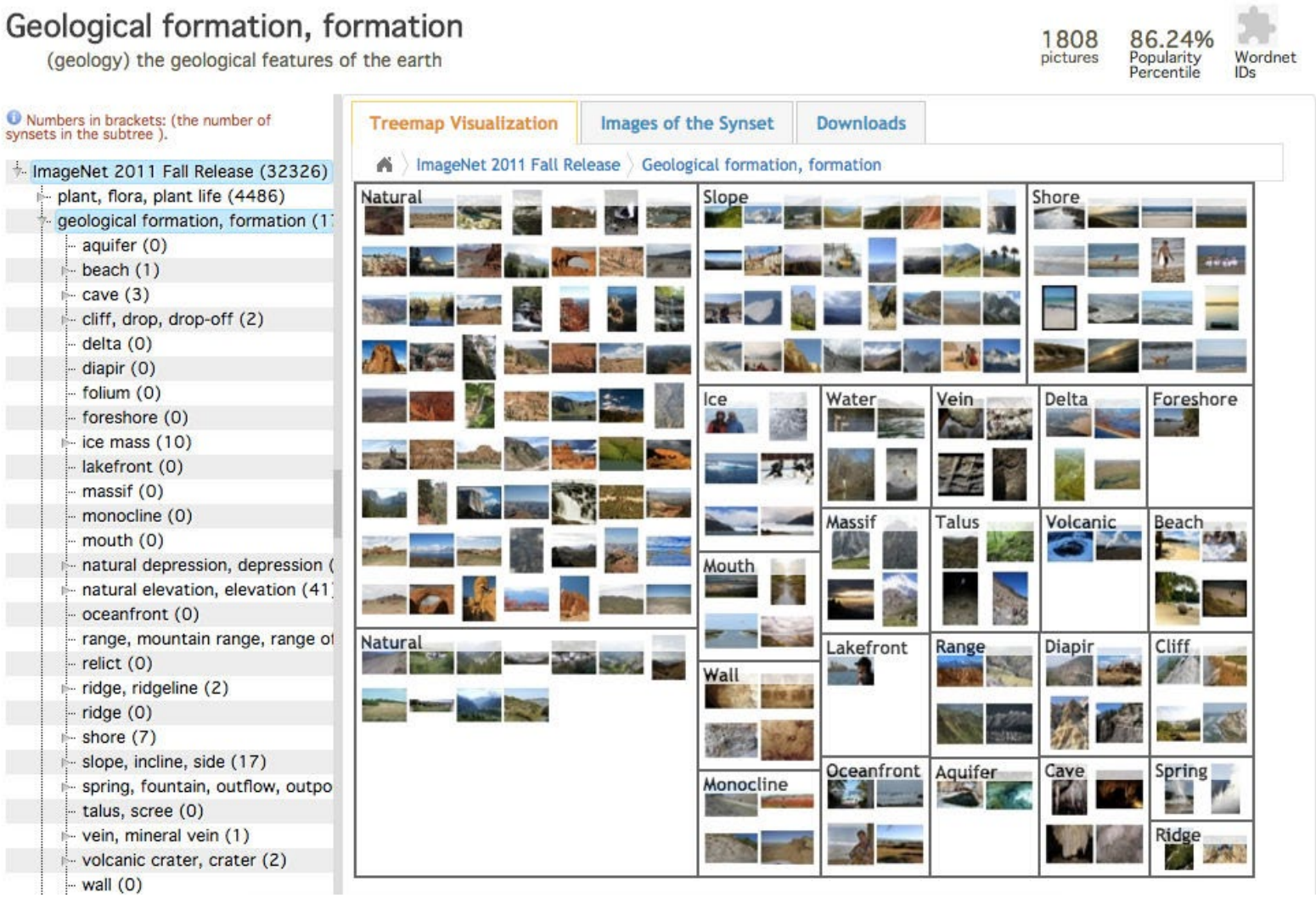


ImageNet

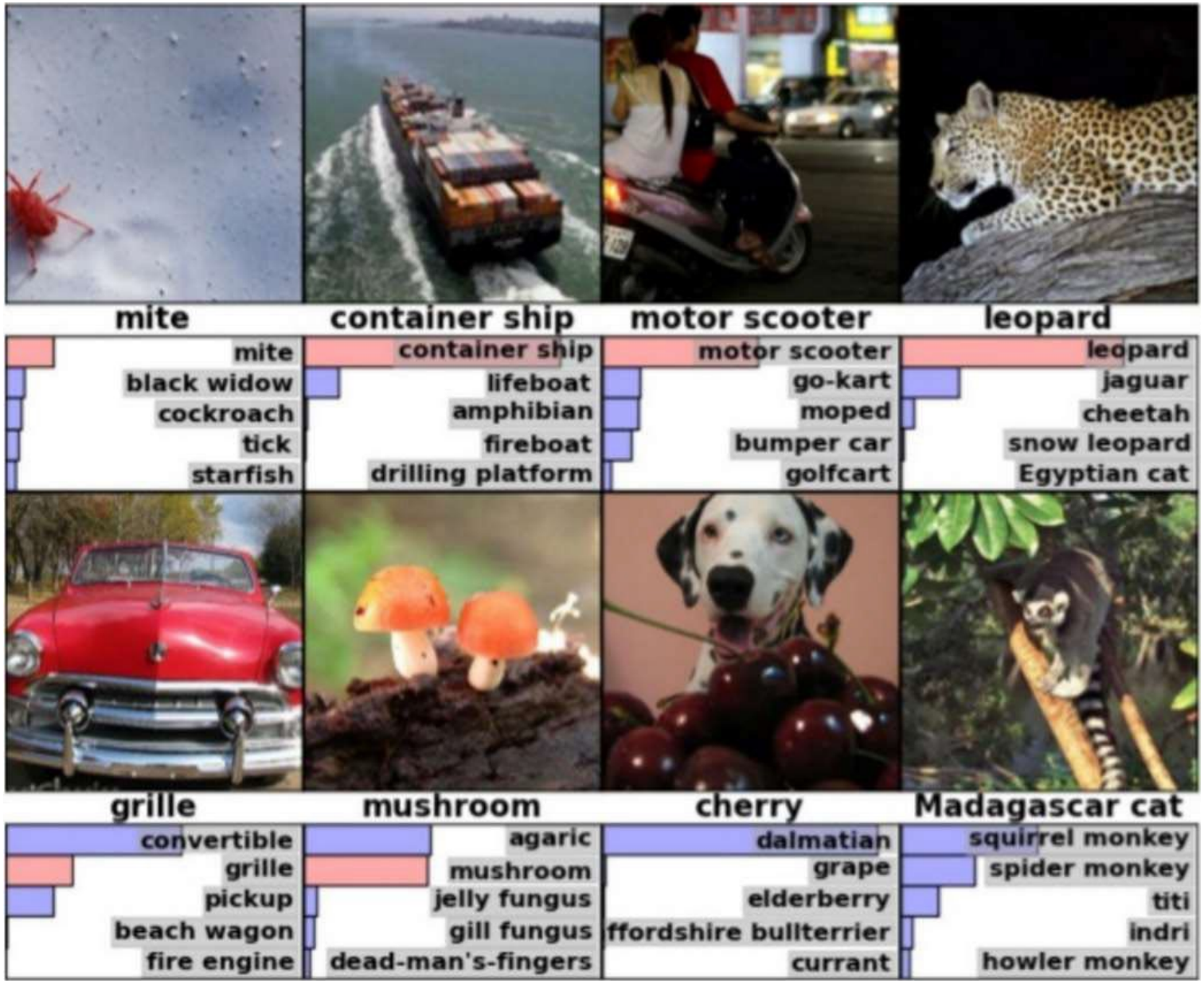
- Images for each category of WordNet
- 1000 classes
  - Manually labeled by humans
- 1.2mil images
- 100k test

- Top 5 error

- https://www.image-net.org/

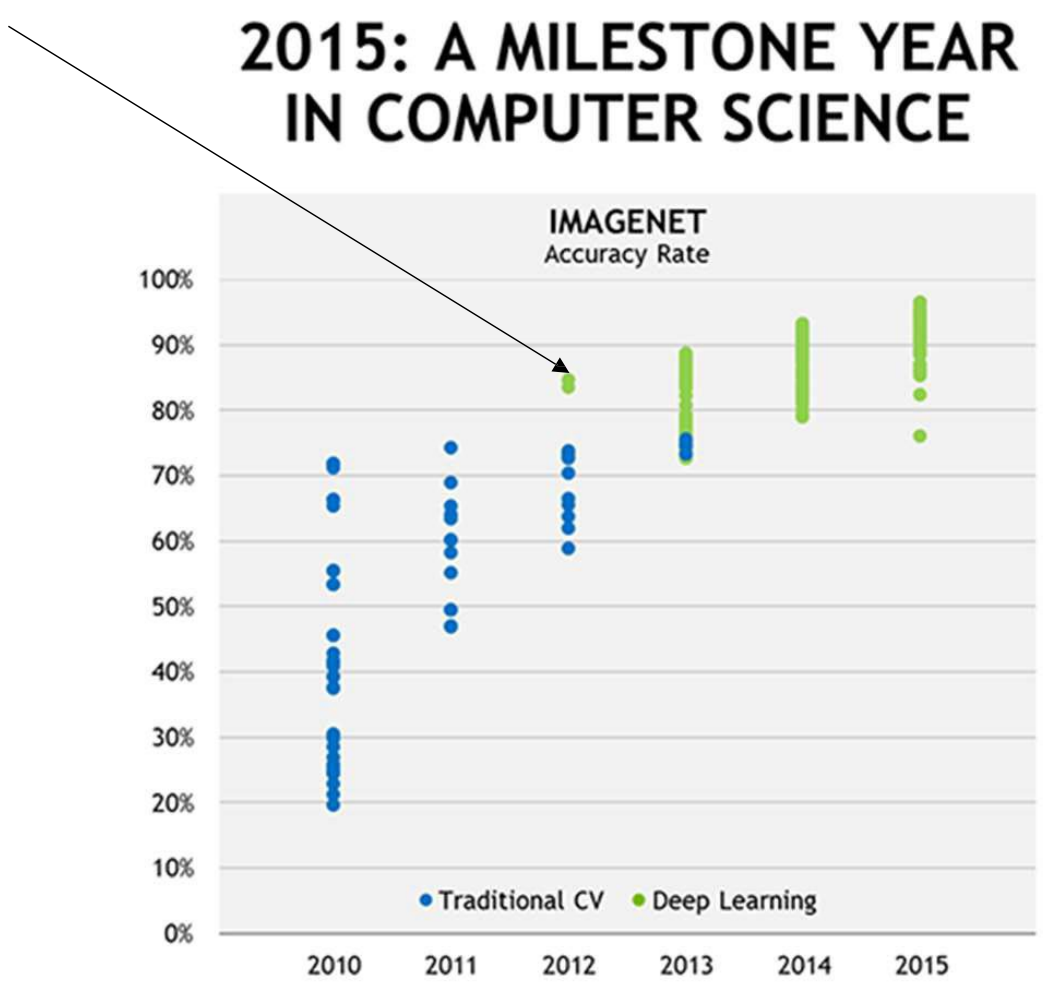


Top 5 error

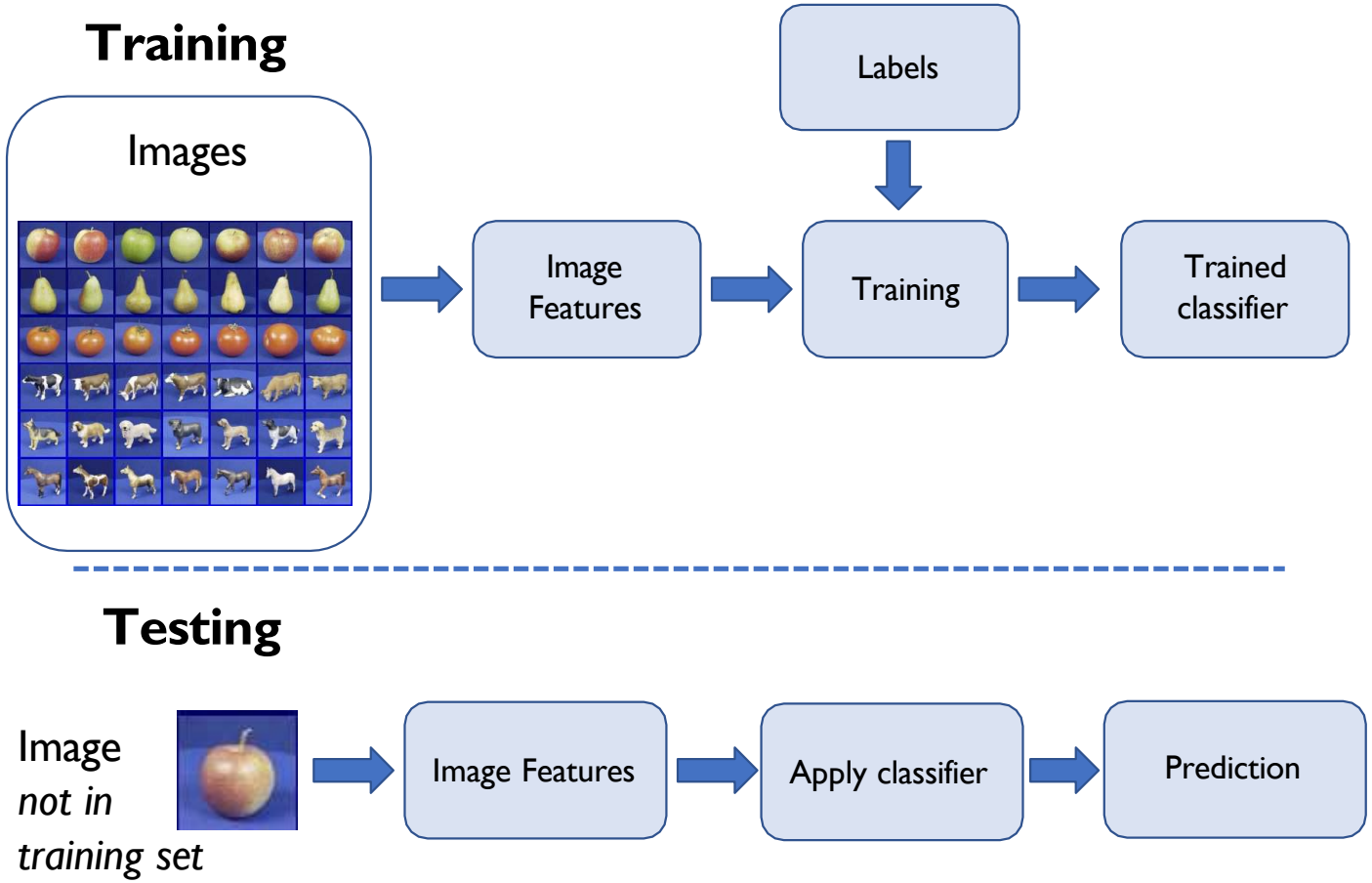

mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

ImageNet Competition

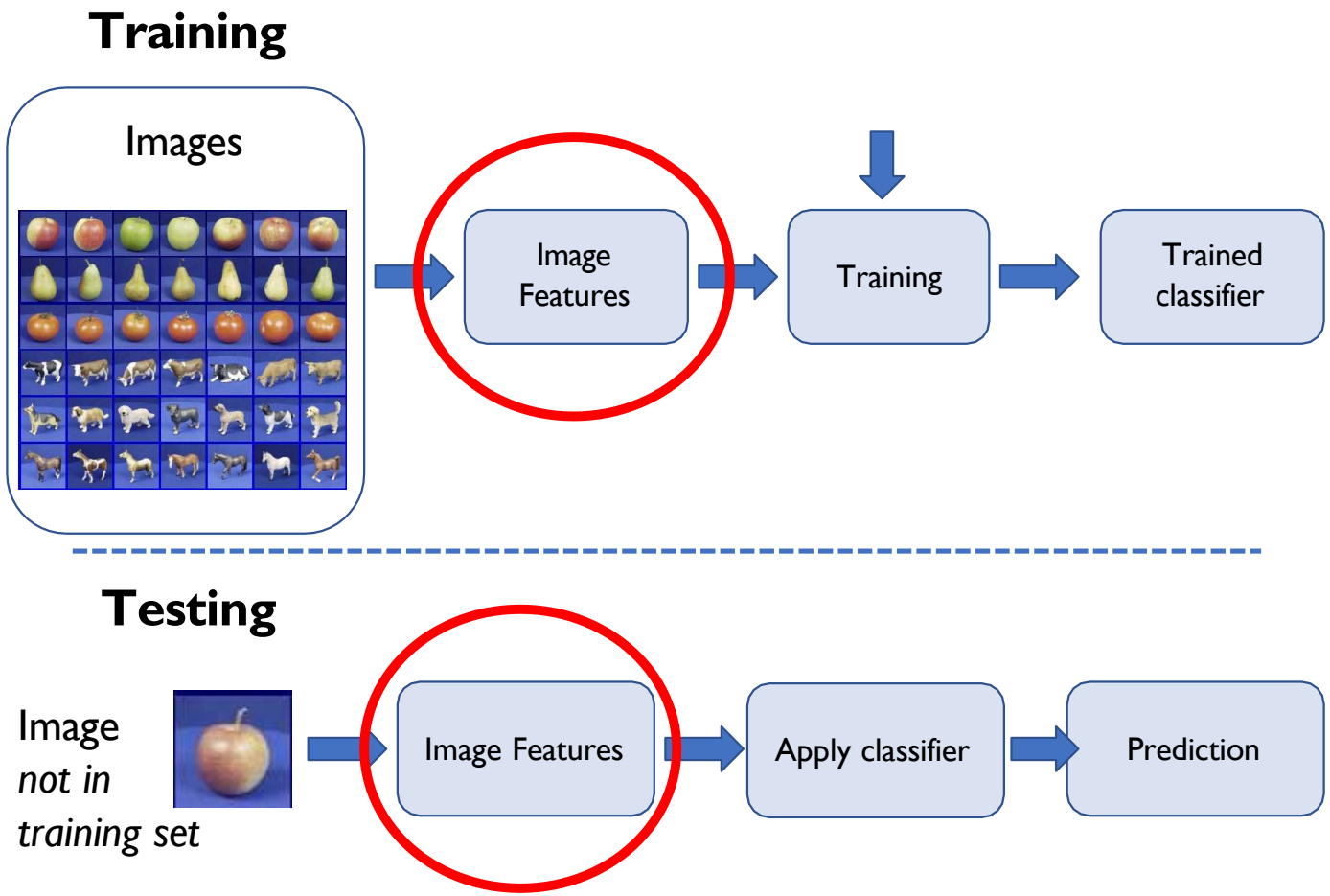
- Krizhevsky, 2012
- Google, Microsoft 2015
  - Beat the best human score in the ImageNet challenge.



Training & Testing

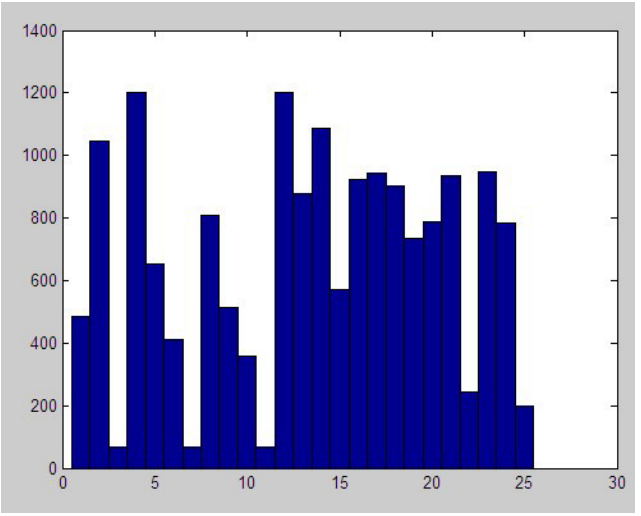


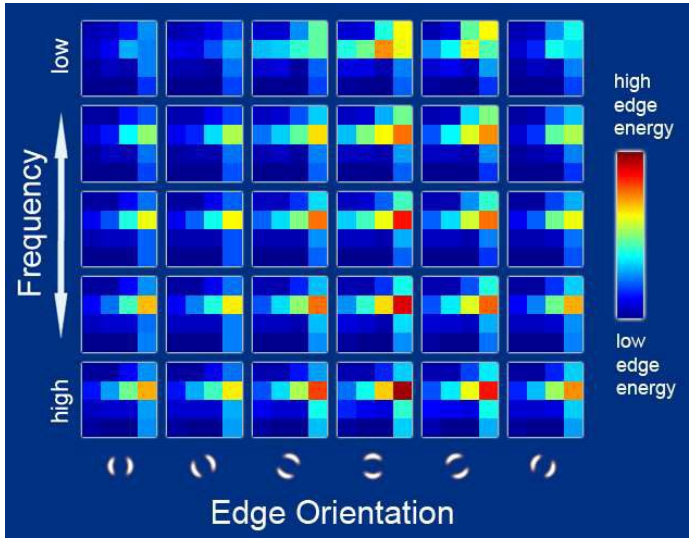
Features


Training
Images
Image Features
Training
Trained classifier
Testing
Image *not in training set*
Image Features
Apply classifier
Prediction

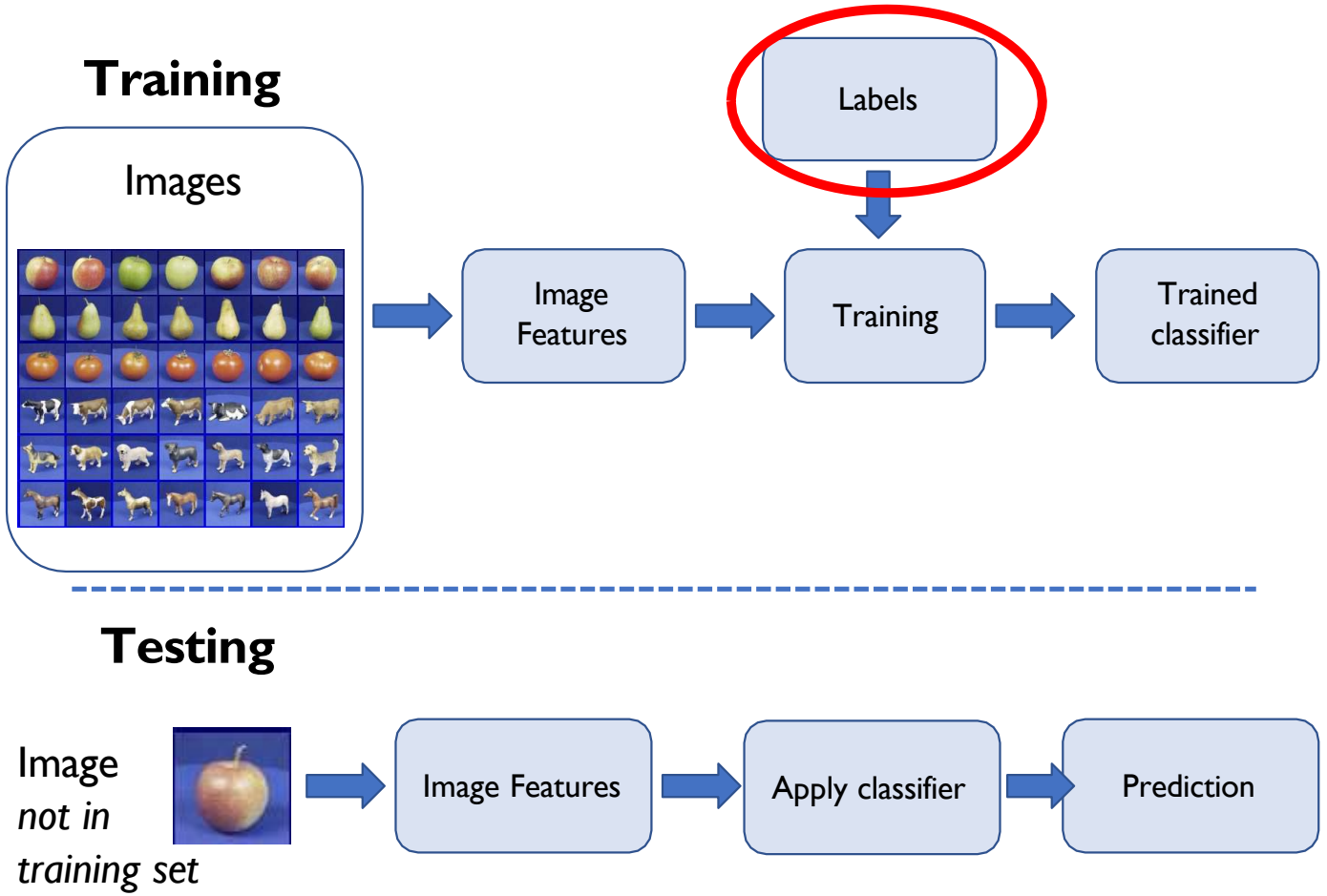
Features

- Raw pixels
- Histograms
- Templates
- SIFT descriptors
  - GIST
  - ORB
  - HOG….





Labels


Training
Images
Labels
Image Features
Training
Trained classifier
Testing
Image *not in training set*
Image Features
Apply classifier
Prediction

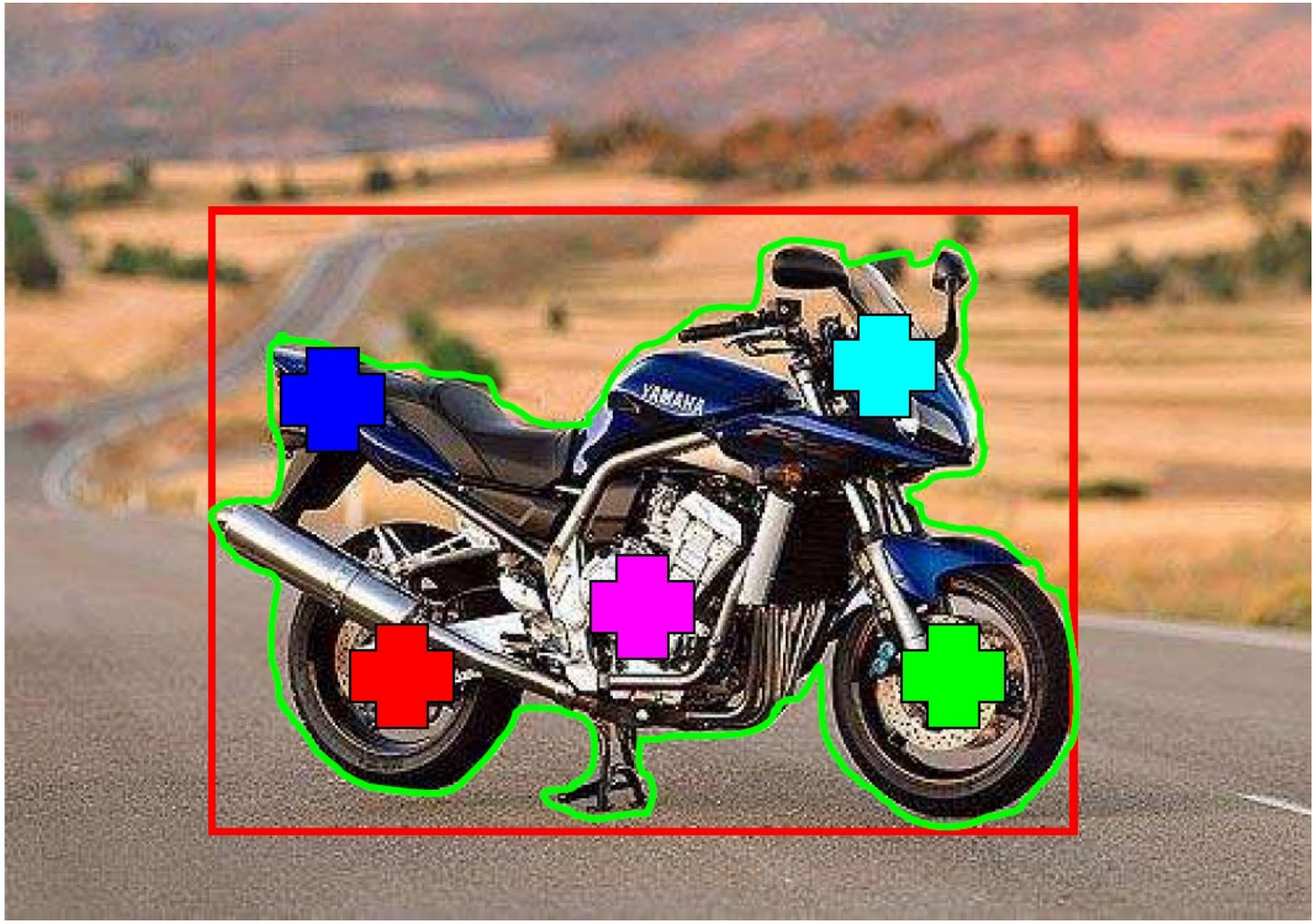
Label Types

What are all the possible supervision ('label') *types* to consider?

Label Types

Images in the training set must be annotated with the "correct answer" that the model is expected to produce

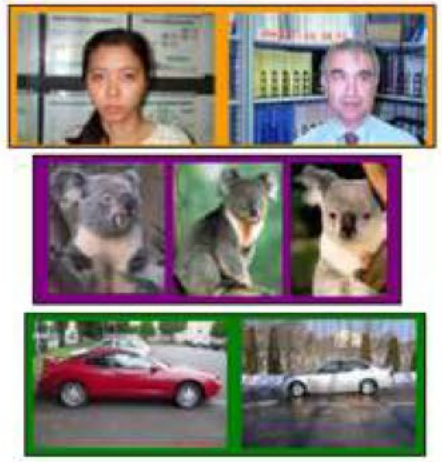
Contains a motorbike

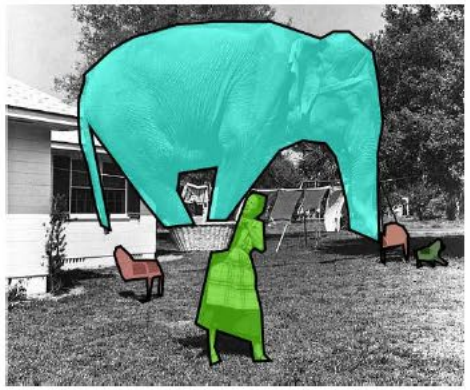


Spectrum of Supervision

Less

More





Unsupervised

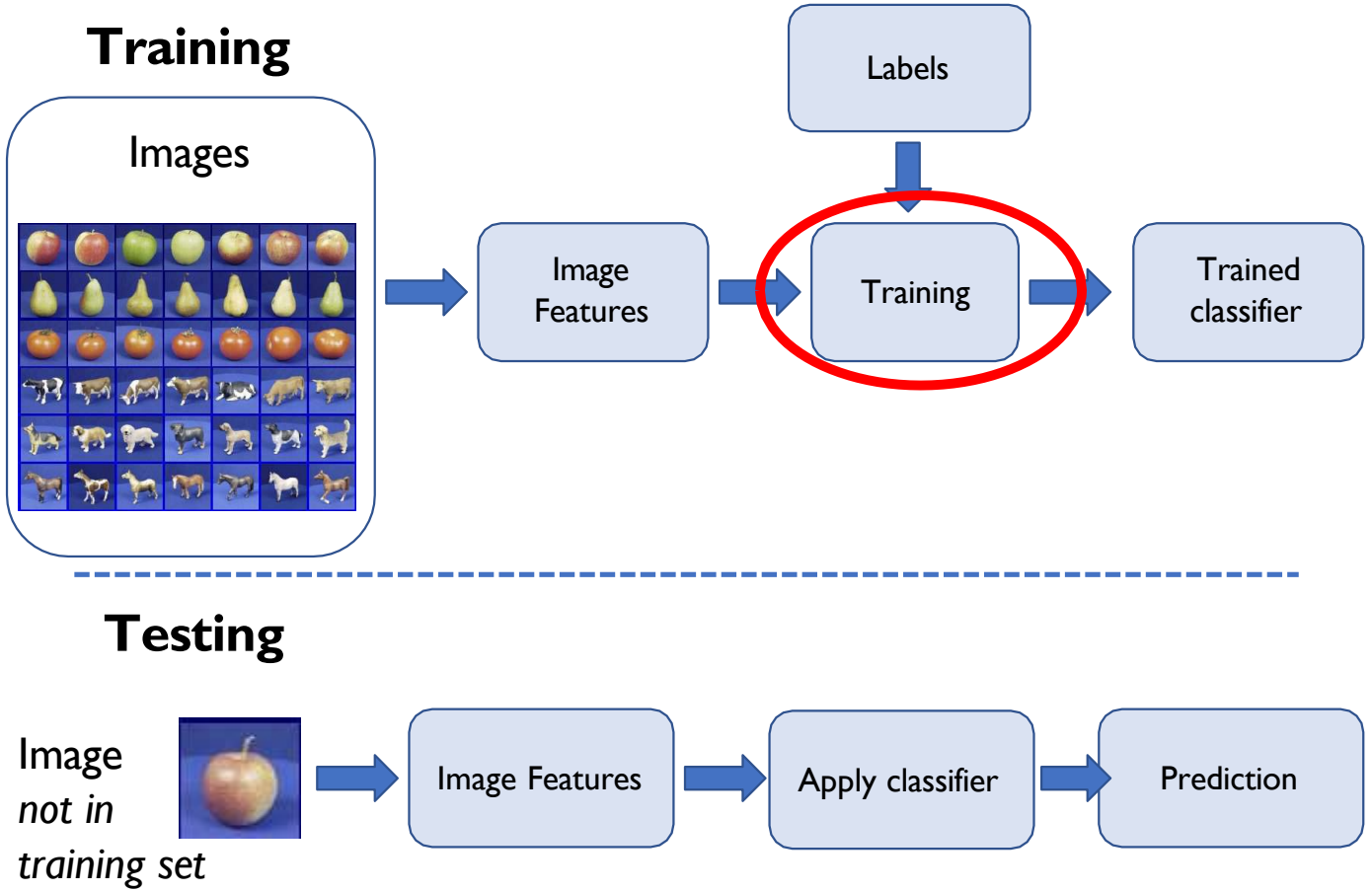
"Weakly" supervised

Fully supervised

Fuzzy; definition depends on task

'Semi-supervised': small partial labeling

Training


Training
Images
Labels
Image Features
Training
Trained classifier
Testing
Image *not in training set*
Image Features
Apply classifier
Prediction

The (Supervised) Machine Learning Framework

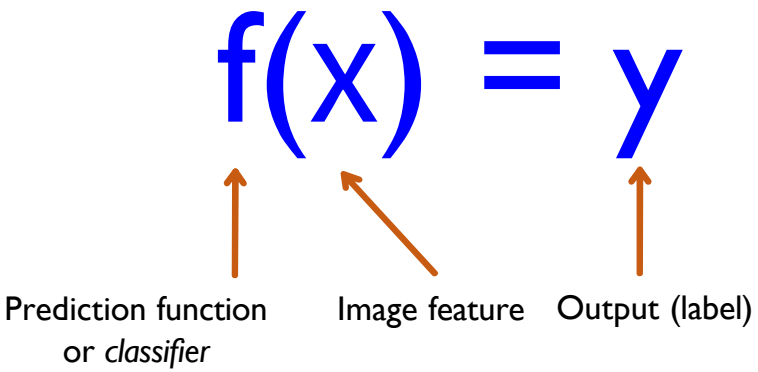
Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

The (Supervised) Machine Learning Framework



Training: Given a *training set* of labeled examples:

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

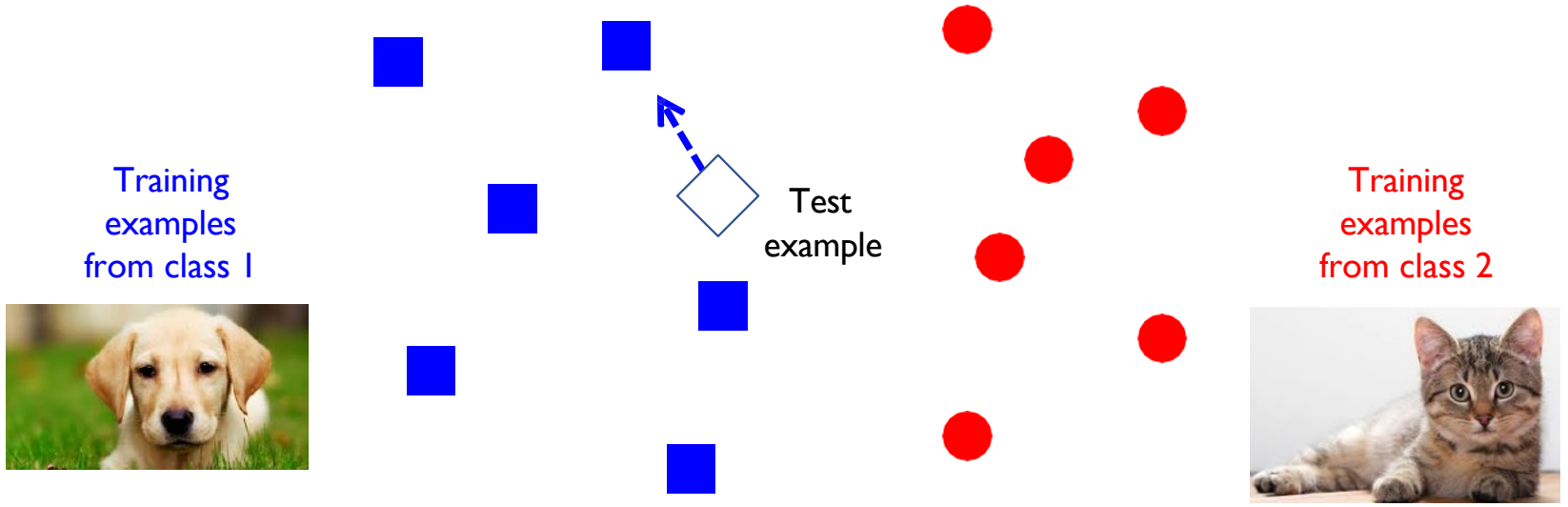
Estimate the prediction function f by minimizing the prediction error on the training set.

Testing: Apply f to an unseen *test example* $\mathbf{x}_u$ and output the predicted value $y_u = f(\mathbf{x}_u)$ to *classify* $\mathbf{x}_u$.

Classifiers: Nearest Neighbor

f(**x**) = label of the training example nearest to **x**

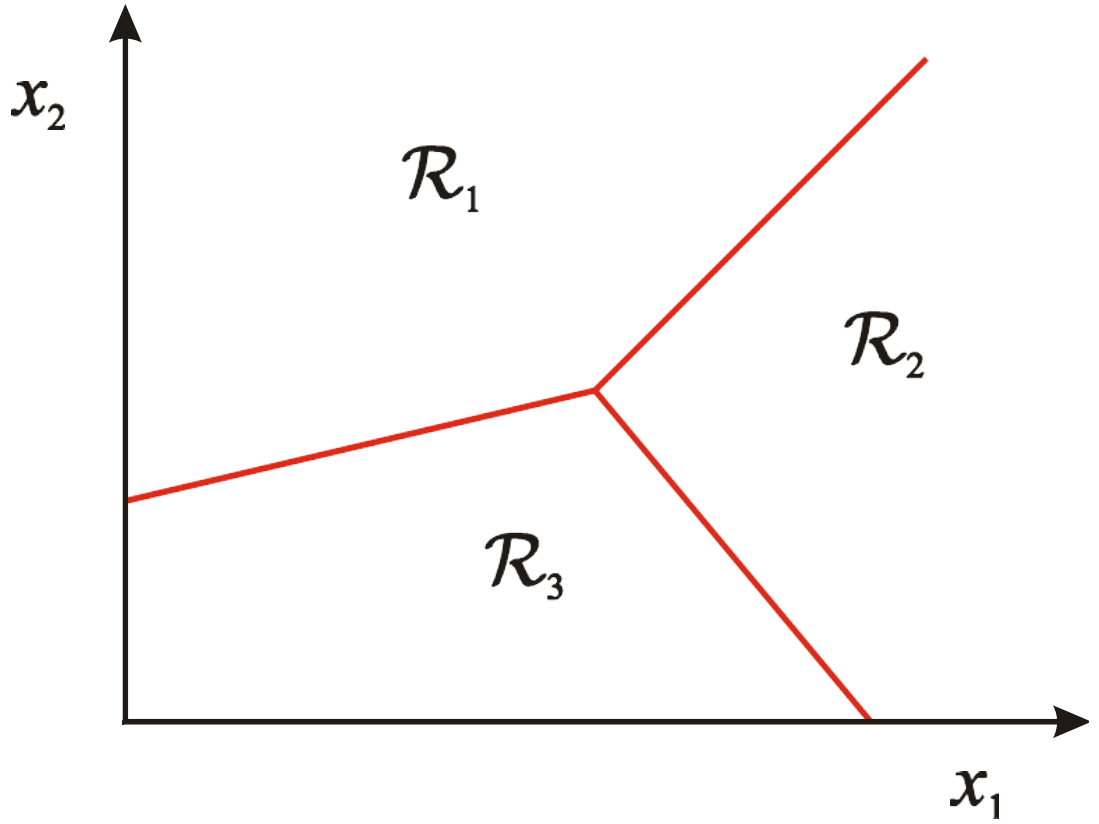
- All we need is a distance function for our inputs
- No training required!



What does the decision boundary look like?

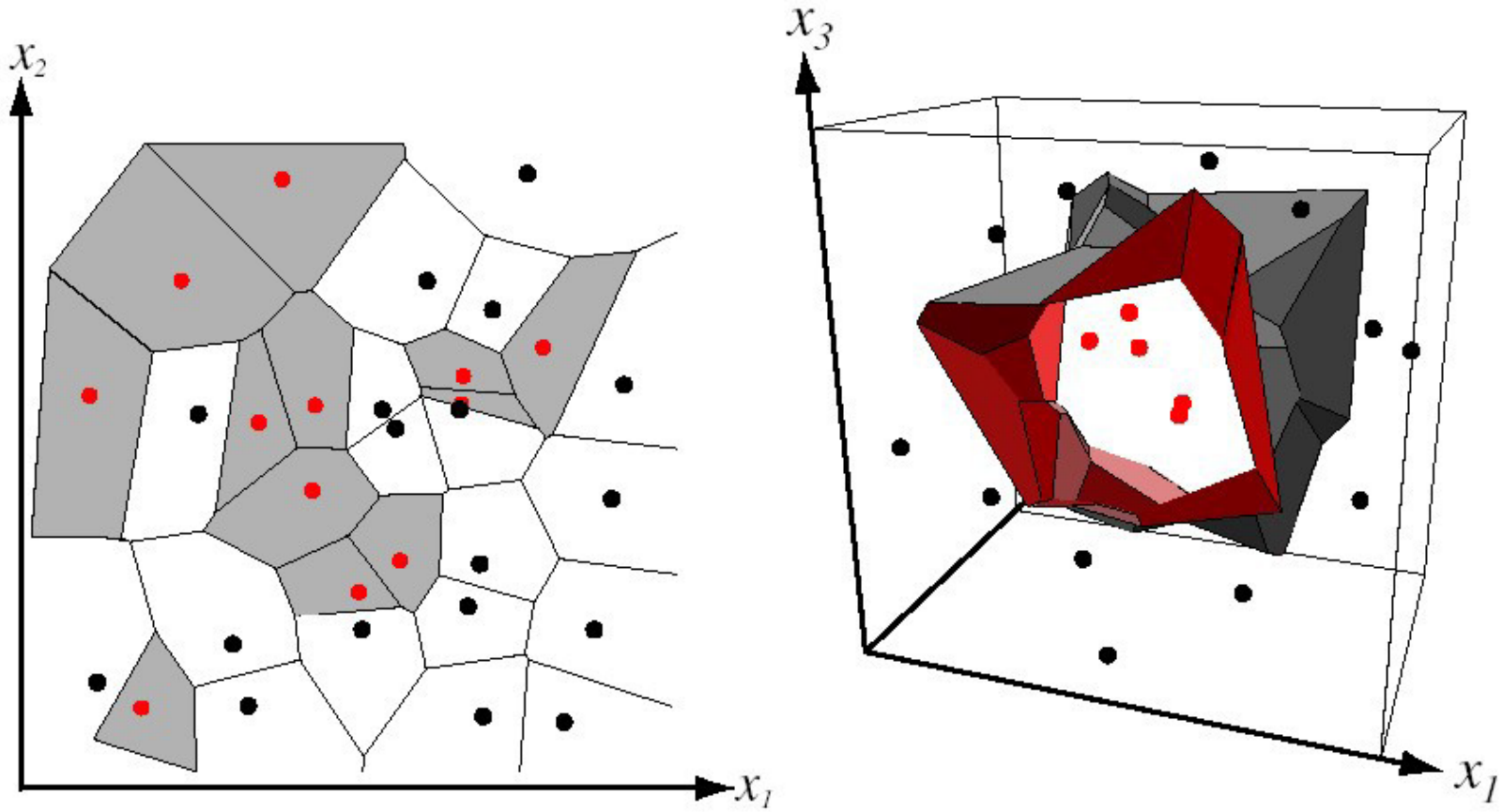
Classification

- Assign **x** to one of two (or more) classes.
- A decision rule divides input space into ***decision regions*** separated by ***decision boundaries*** – literally boundaries in the space of the features.



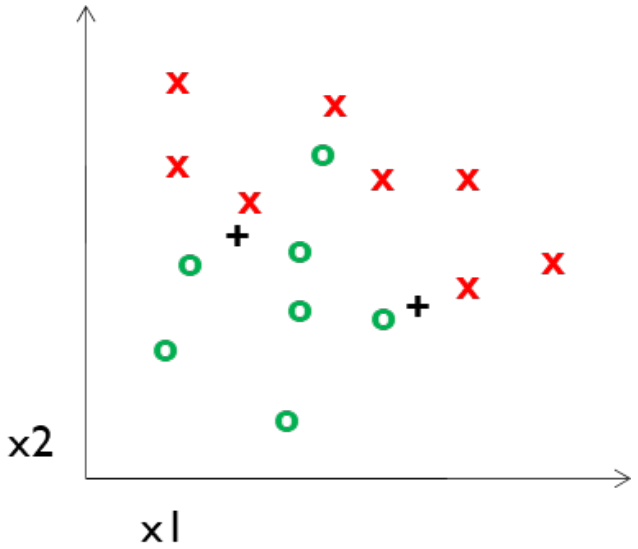
Decision boundary for Nearest Neighbor Classifier

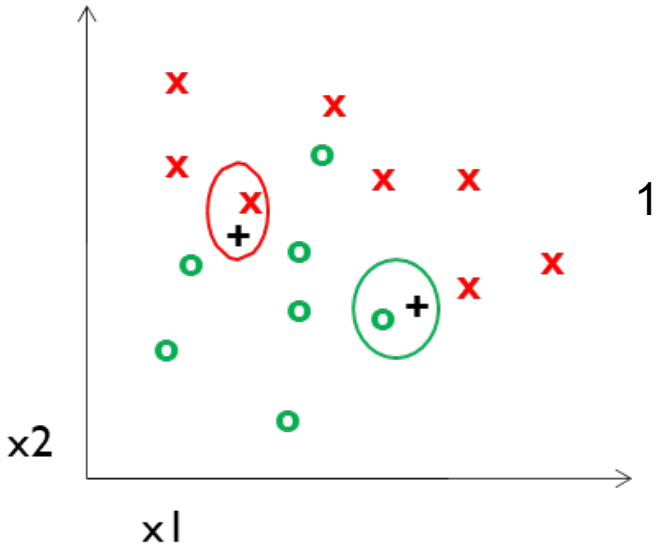
Divides input space into *decision regions* separated by *decision boundaries* – *Voronoi.*



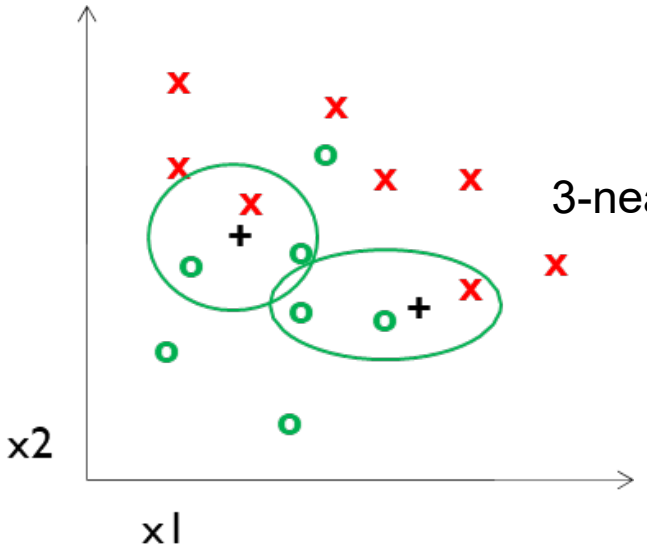
Voronoi partitioning of feature space for two-category 2D and 3D data

k-Nearest Neighbor

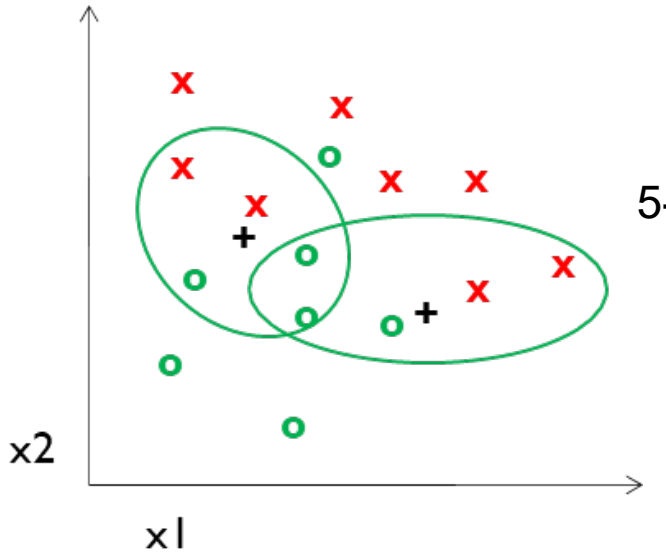




1-nearest

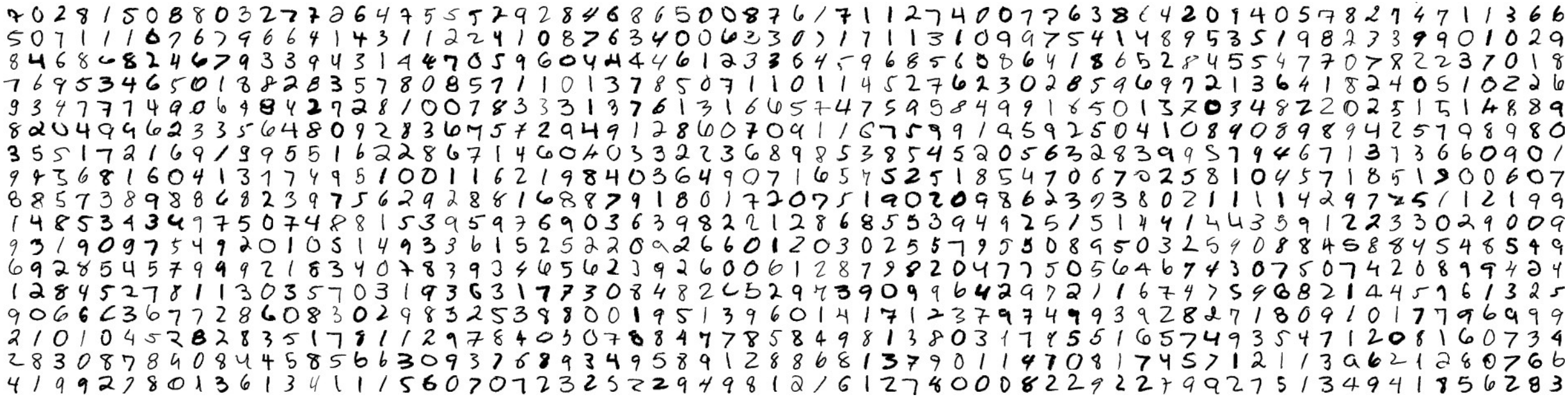


3-nearest



5-nearest

Nearest Neighbor is competitive



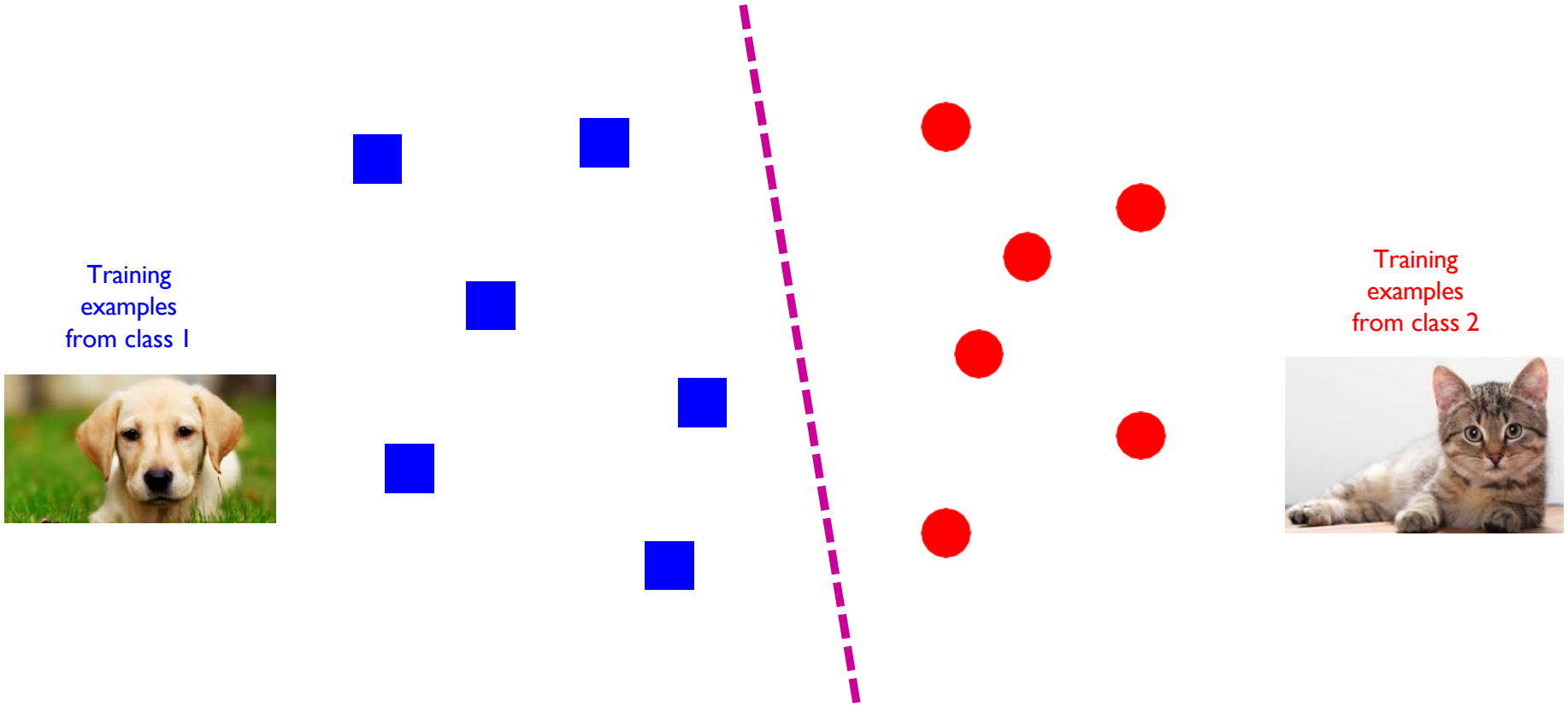
MNIST Digit Recognition

- Handwritten digits
- 28x28 pixel images: d = 784
- 60,000 training samples
- 10,000 test samples

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Classifiers: Linear

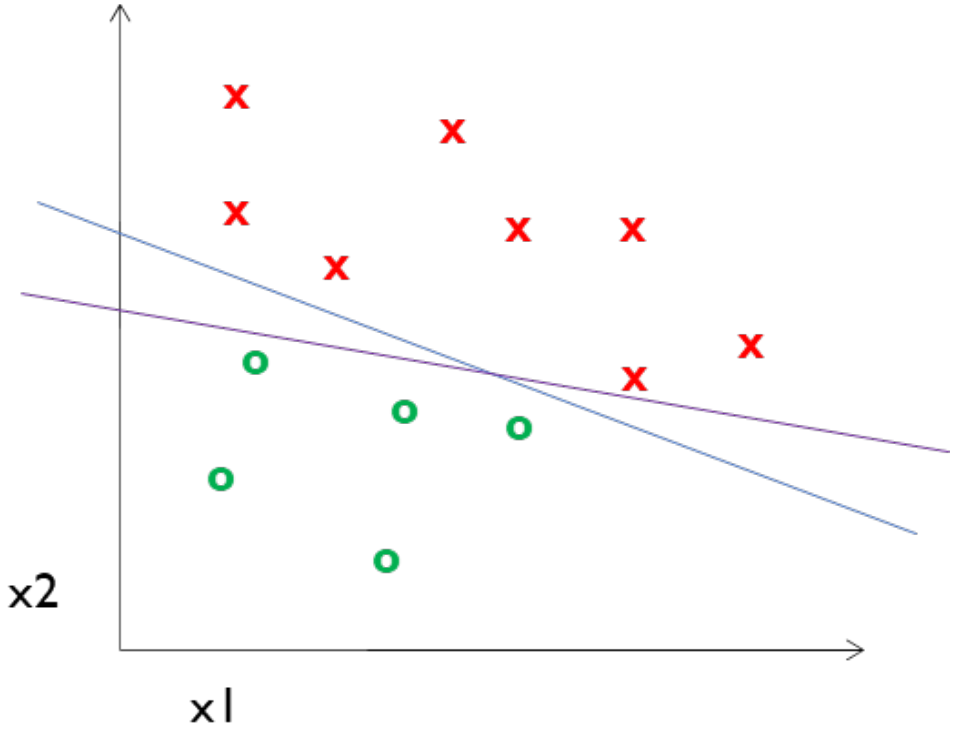
Find a ***linear function*** to separate the classes



Classifiers: Linear Support Vector Machine (SVM)

Find a *linear function*
to separate the classes:

$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$



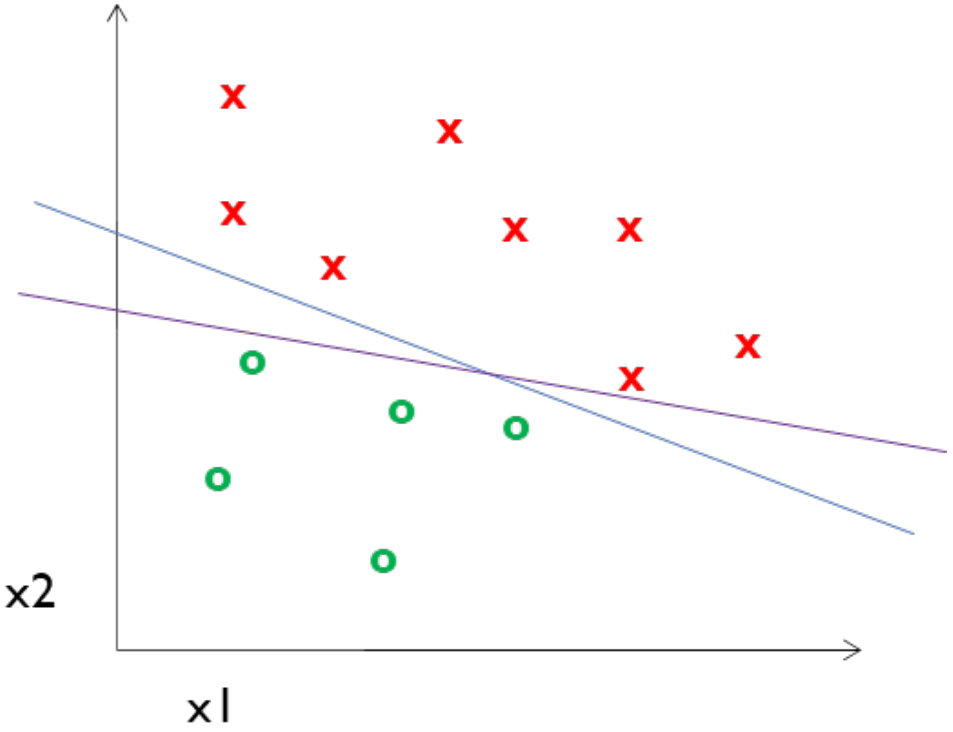
Classifiers: Linear SVM

Find a *linear function* to separate the classes:

$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$

How?

X = all data points



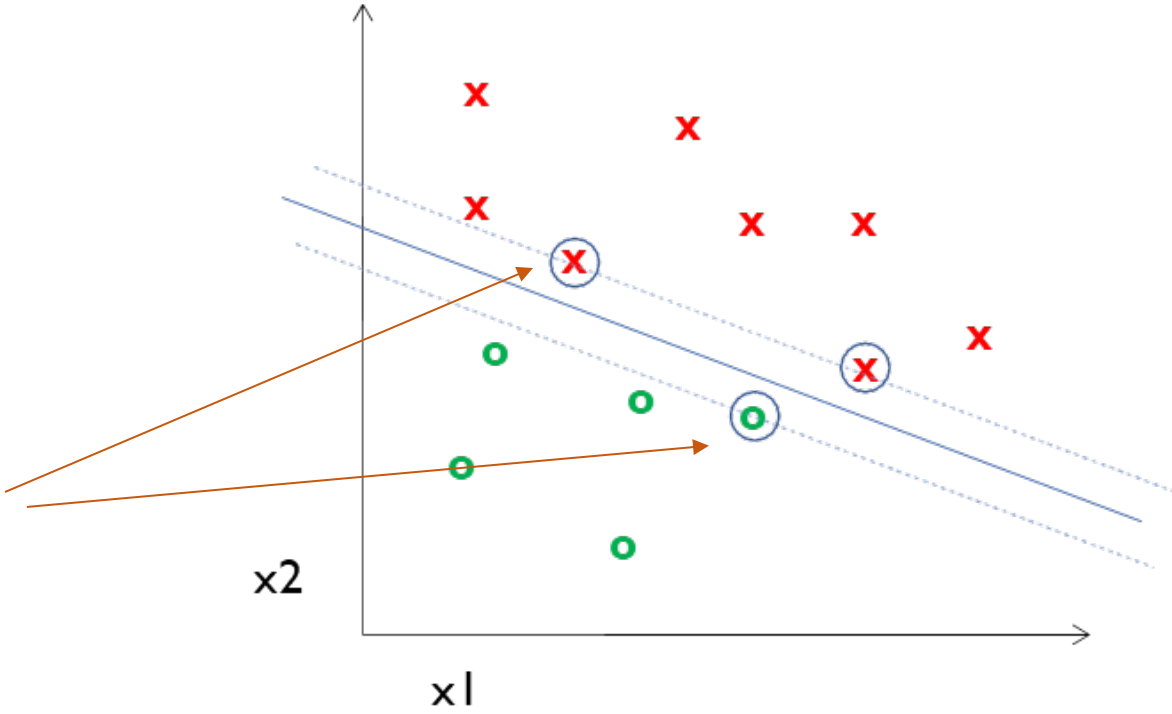
Define *hyperplane* **tX-b = 0**, where **t** is tangent to hyperplane.

Minimize ||**t**|| s.t. **tX-b** produces correct label for all **X**

Classifiers: Linear SVM

Find a *linear function* to separate the classes:

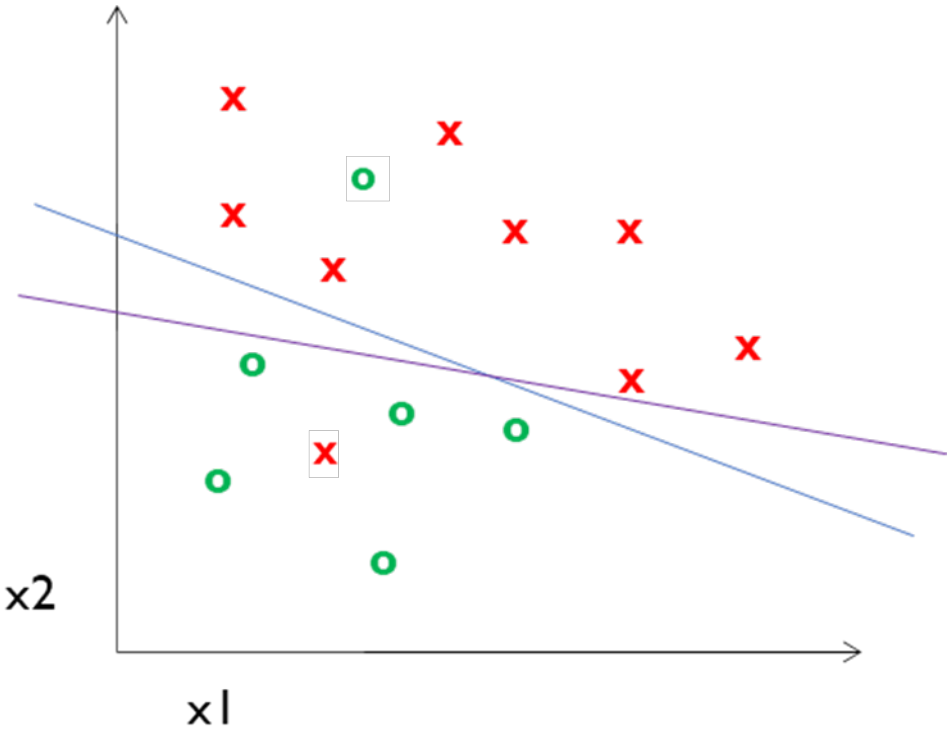
$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$



Classifiers: Linear SVM

Find a *linear function* to separate the classes:

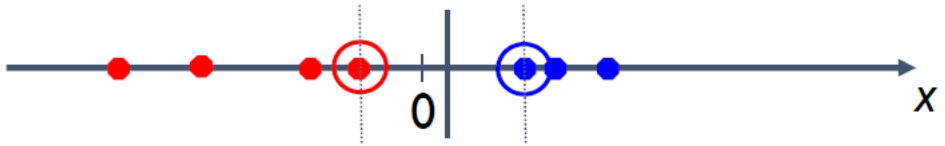
$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$



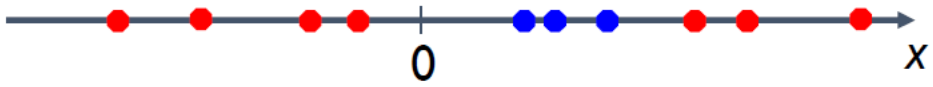
What if my data are not linearly separable?

Additional Slide: Nonlinear SVMs

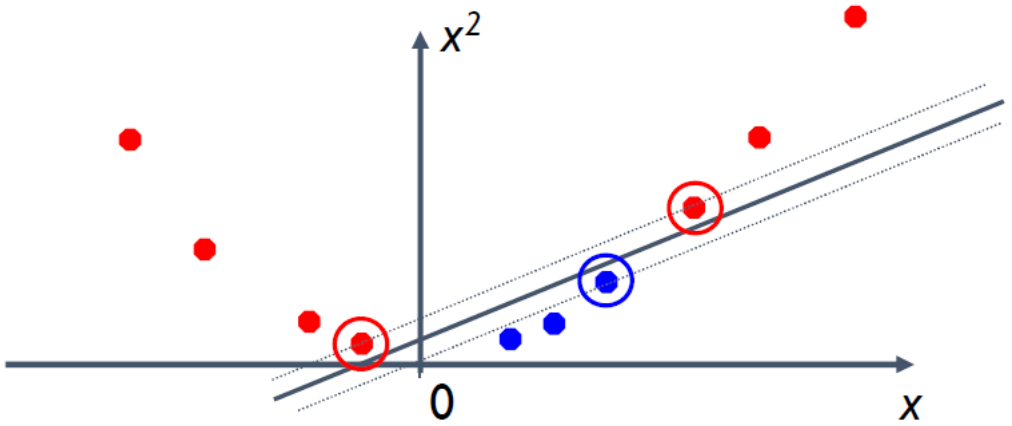
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

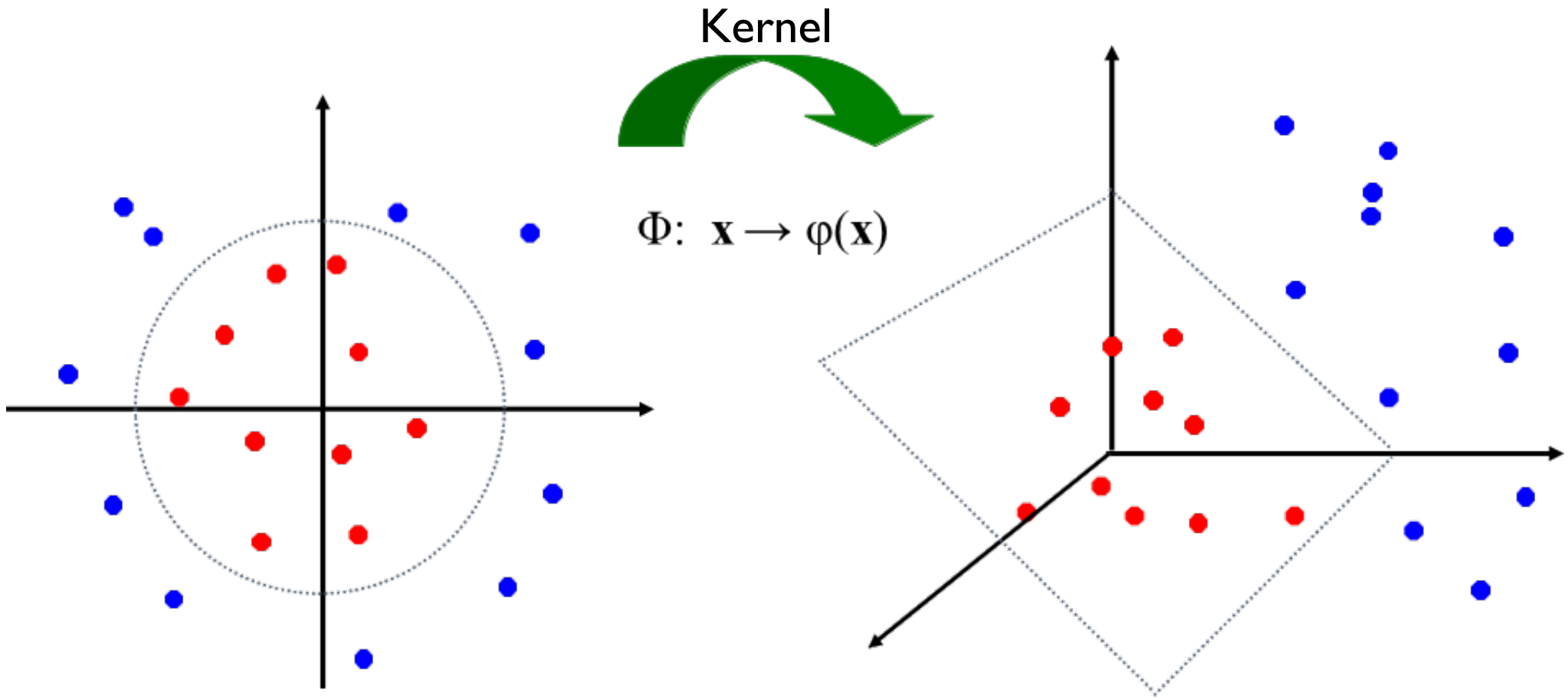


- We can map it to a higher-dimensional space:



Nonlinear SVMs

Map the original input space to some higher-dimensional feature space where the training set is separable:



What about multi-class SVMs?

- Unfortunately, there is no "definitive" multi-class SVM.
- In practice, we combine multiple two-class SVMs
- One vs. others
  - Training: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages:
    https://scikit-learn.org/1.5/modules/svm.html
    https://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Dataset Split

Training Images

- Train classifier

Validation Images

- Measure error
- Tune 'hyperparameters'

Testing Images

- Secret labels
- Measure error

Random train/validate splits = cross validation

Generalization

Training set (labels known)

Test set (labels unknown)

How well does a learned model generalize from the data it was trained on to a new test set?

Generalization Error

Bias:

- Difference between the expected (or average) prediction of our model and the correct value.
- Error due to inaccurate assumptions/simplifications

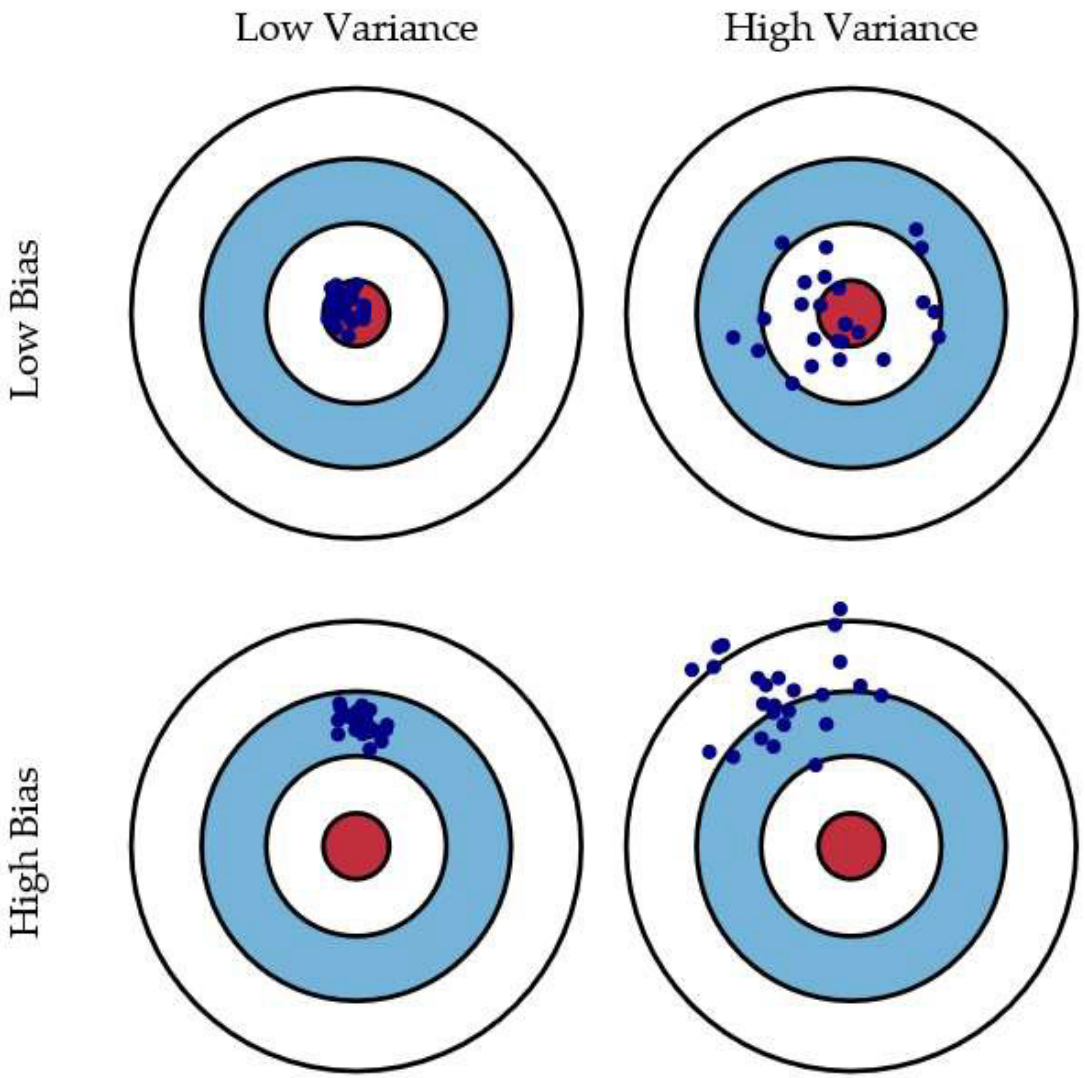
Variance:

- Amount that the estimate of the target function will change if different training data was used.

Bias/variance trade-off

Bias = accuracy

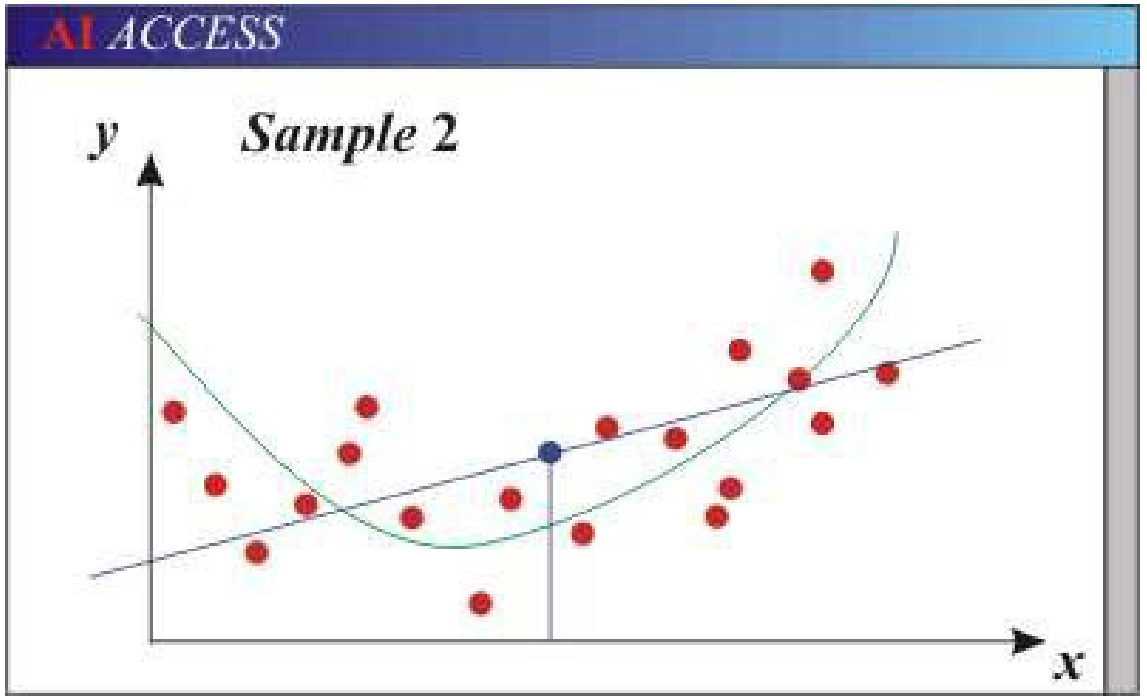
Variance = precision



Generalization Error Effects

Underfitting: model is too "simple" to represent all the relevant class characteristics

- High bias (few degrees of freedom) and low to high variance
- High training error and high test error

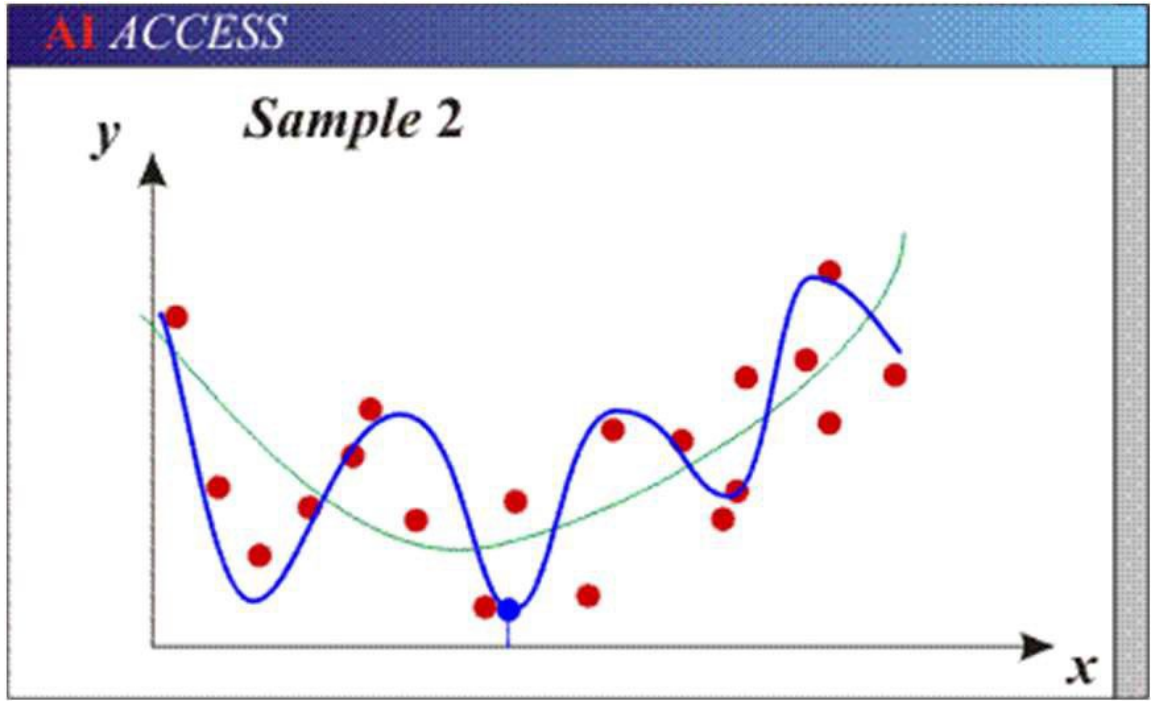


Green line = true data-generating function without noise
Blue line = data model which underfits

Generalization Error Effects

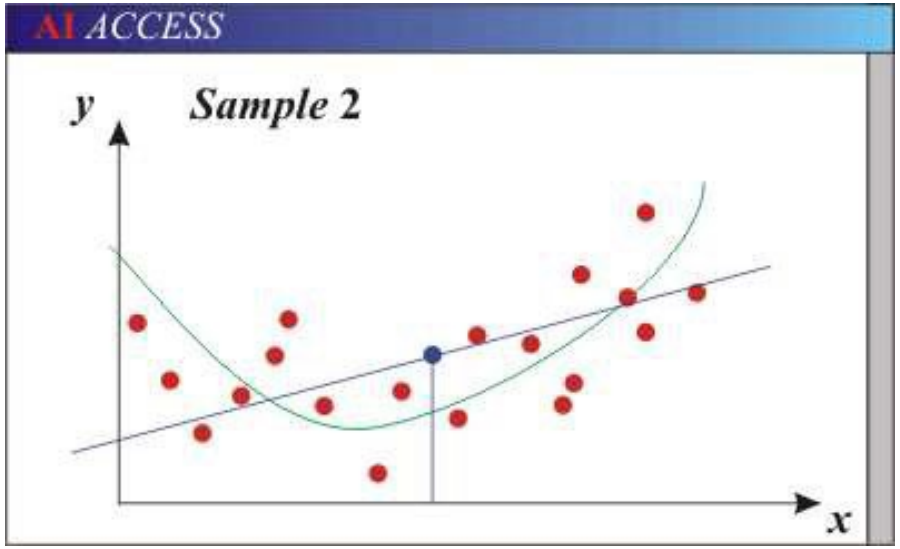
Overfitting: model is too "complex" and fits irrelevant characteristics (noise) in the data

- Low bias (many degrees of freedom) and high variance
- Low training error and high test error



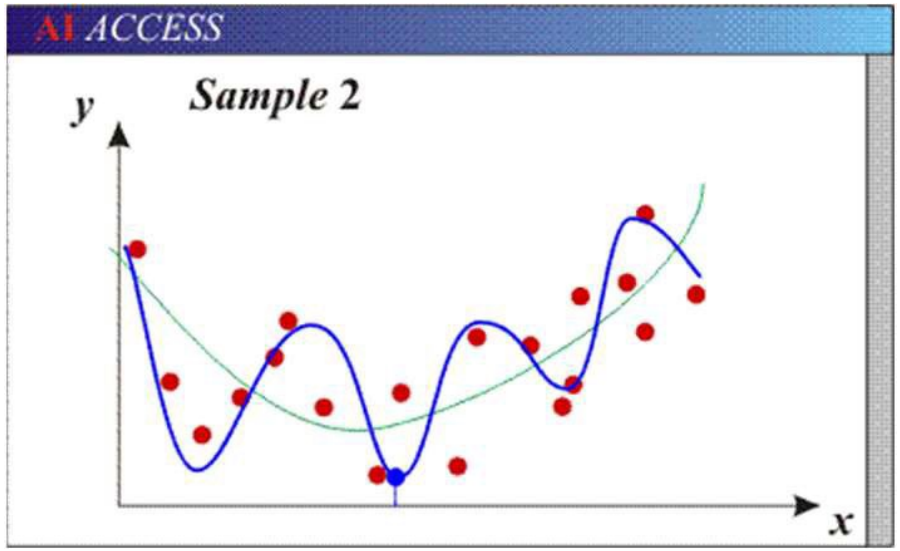
Green line = true data-generating function without noise
Blue line = data model which overfits

Bias-Variance Trade-off



Models with too **few parameters** are inaccurate because of a large bias

- Not enough flexibility!
- Too many assumptions/constraints



Models with too **many parameters** are inaccurate because of a large variance

- Too much sensitivity to the sample
- Slightly different data -> very different function

Many classifiers to choose from…

- K-nearest neighbor
- SVM
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- Restricted Boltzmann Machines
- Neural networks
- Deep Convolutional Network
- …

→ Cat

What to remember about classifiers

- No free lunch: Machine learning algorithms are tools
- Try **simple** classifiers first
- Better to have **smart features** and simple classifiers than simple features and smart classifiers
- Use **increasingly** powerful classifiers with more training data (bias-variance tradeoff)