Crash Course in Machine Learning

Dr. Randy C. Hoover

Department of Electrical and Computer Engineering
South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

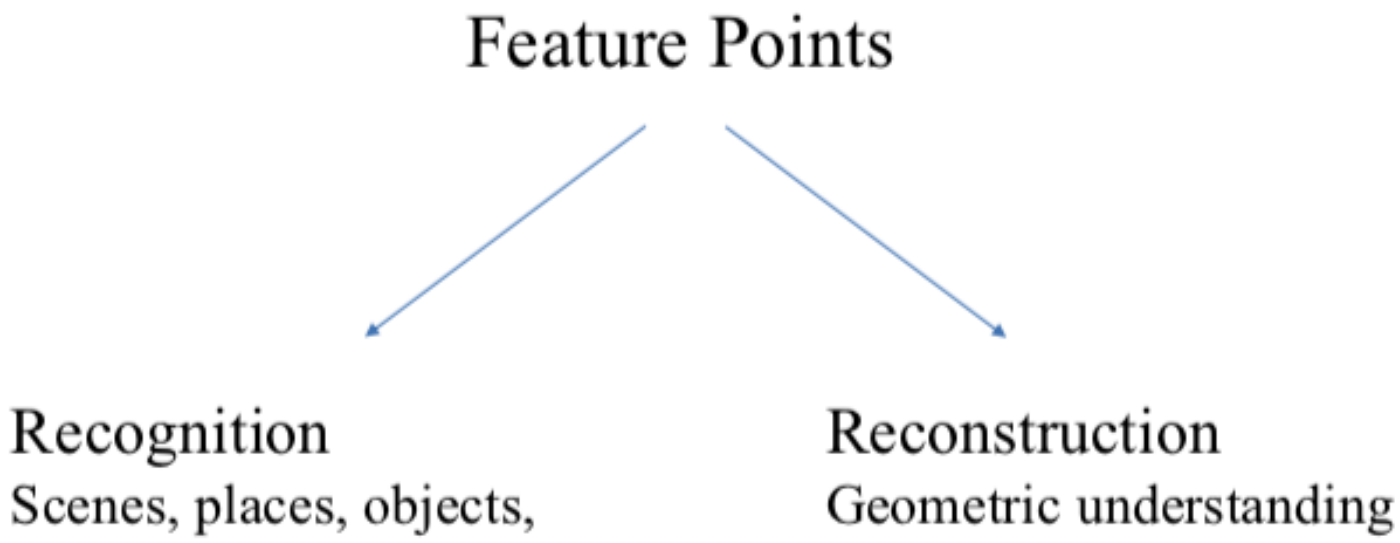
April 2, 2019

Features Check - What now?

- To this point:
  - Everything we have done is feature based!
  - This was a common trend for several decades
  - However, we can also do things based upon appearance (as we will soon see)
  - Good for us since this is complementary to feature-based approaches!
- **So where do we go from here???**

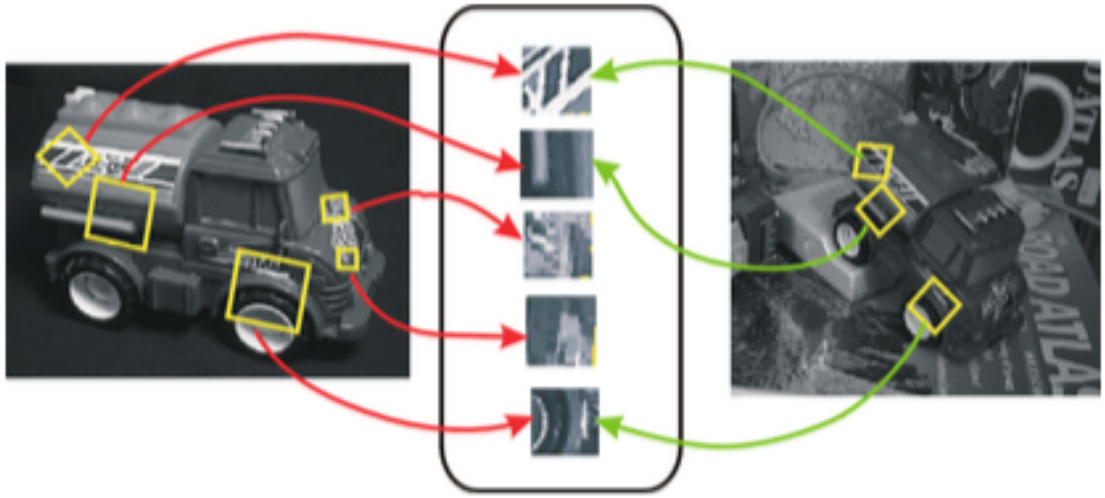
Features Check - What now?

- To this point:
  - Everything we have done is feature based!
  - This was a common trend for several decades
  - However, we can also do things based upon appearance (as we will soon see)
  - Good for us since this is complementary to feature-based approaches!
- **So where do we go from here???**



What we have done with features!

- Panoramic Stitching/Image Alignment!
- Can we use features to "understand" scenes?
- Are there other techniques for scene understanding?

What about object/scene recognition?

- Generally need machine learning techniques to "learn/describe" the visual world.

Scene recognition
- City/forest/factory/...

Find pedestrians

Amusing Satirical Aside

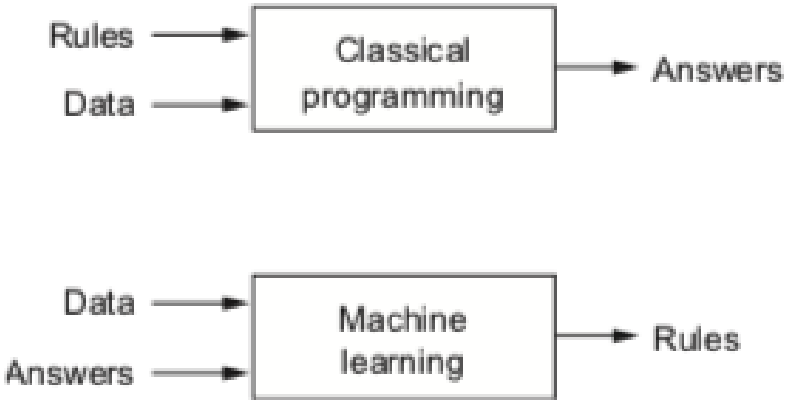
- CMU Machine Learning Department "protests" the G20! (Flickr)

Machine Learning (ML) as a Tool

- In this course we use ML as a tool and will NOT dive into the mathematical rigor (mostly)
- If you're interested in this stuff - the CSC department may be offering a ML course soon - per popular request!

What is Machine Learning?

- Learn from and make predictions on data:

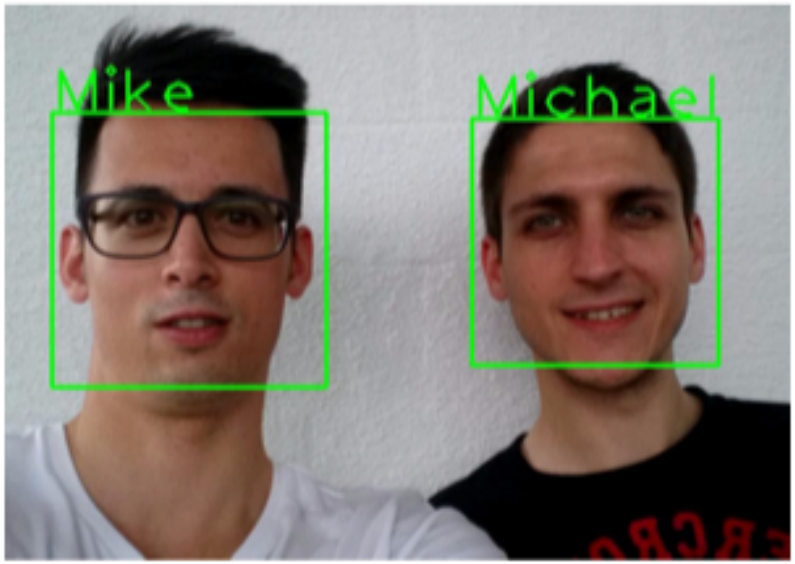


- One of the greatest exports from computing to other fields (and believe me when I say it's everywhere these days!)
- One of the fastest growing and hottest research areas in the world!

ML is applied in MANY different domains!

- In this course we're interested in ML for Computer Vision!

- Face Recognition
- Object Classification
- Scene Segmentation

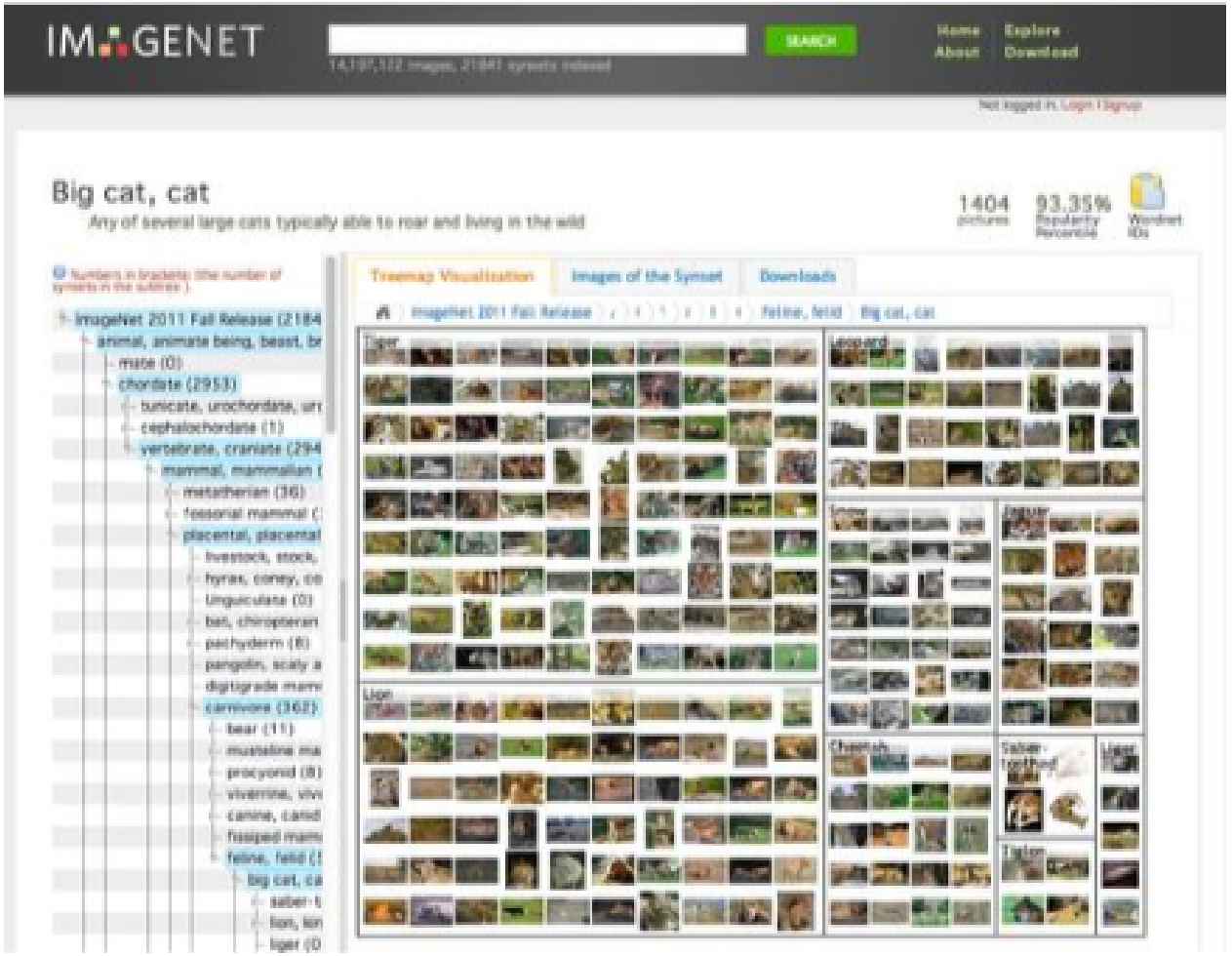


How exactly can machines "learn"?

- Lots and lots and lots and lots and lots of Data!
- 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"

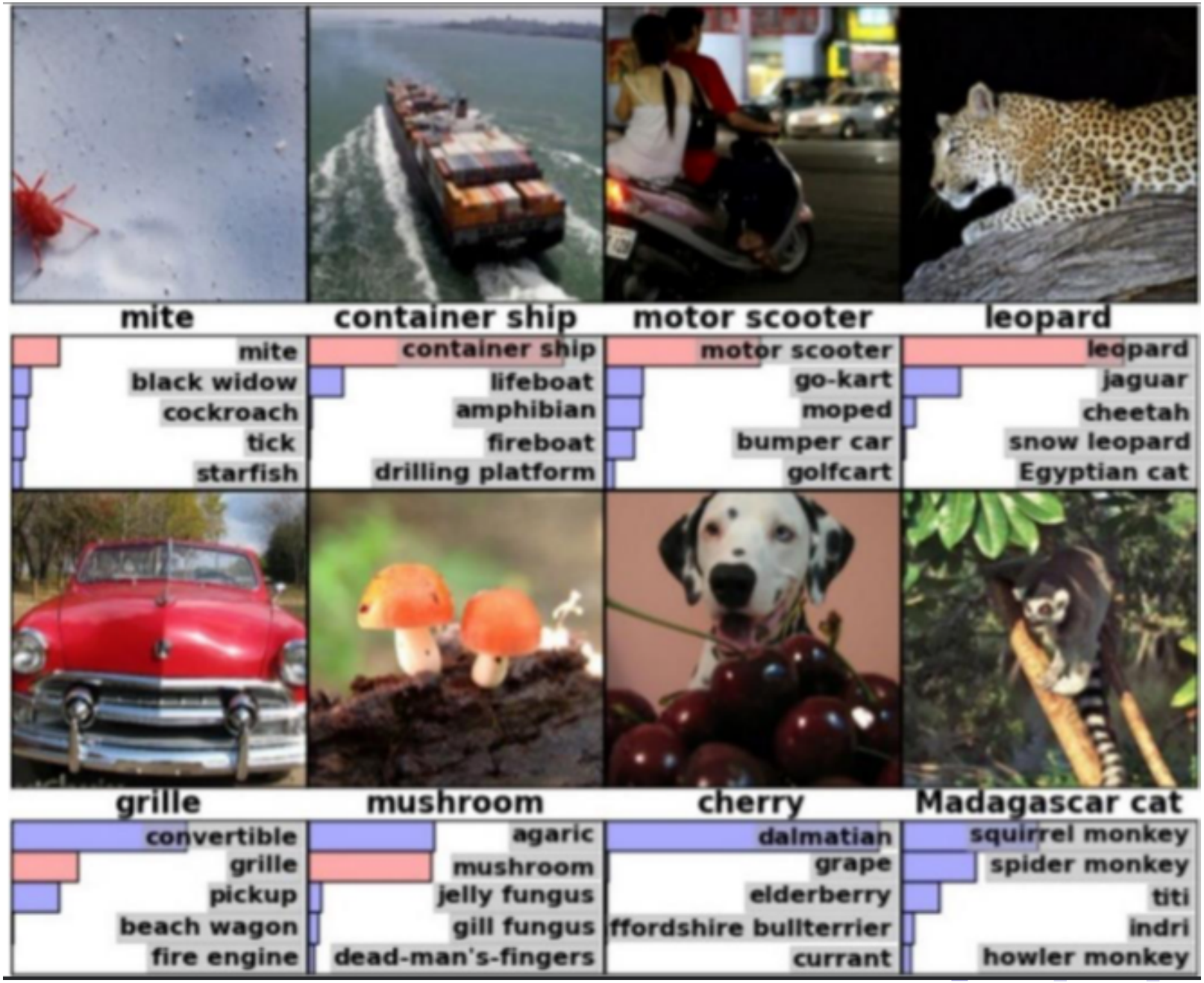
Classic Benchmark Challenge - ImageNet

- Images for each category of WordNet
- 1000 classes
- 1.2mil images
- 100k test
- Top 5 error



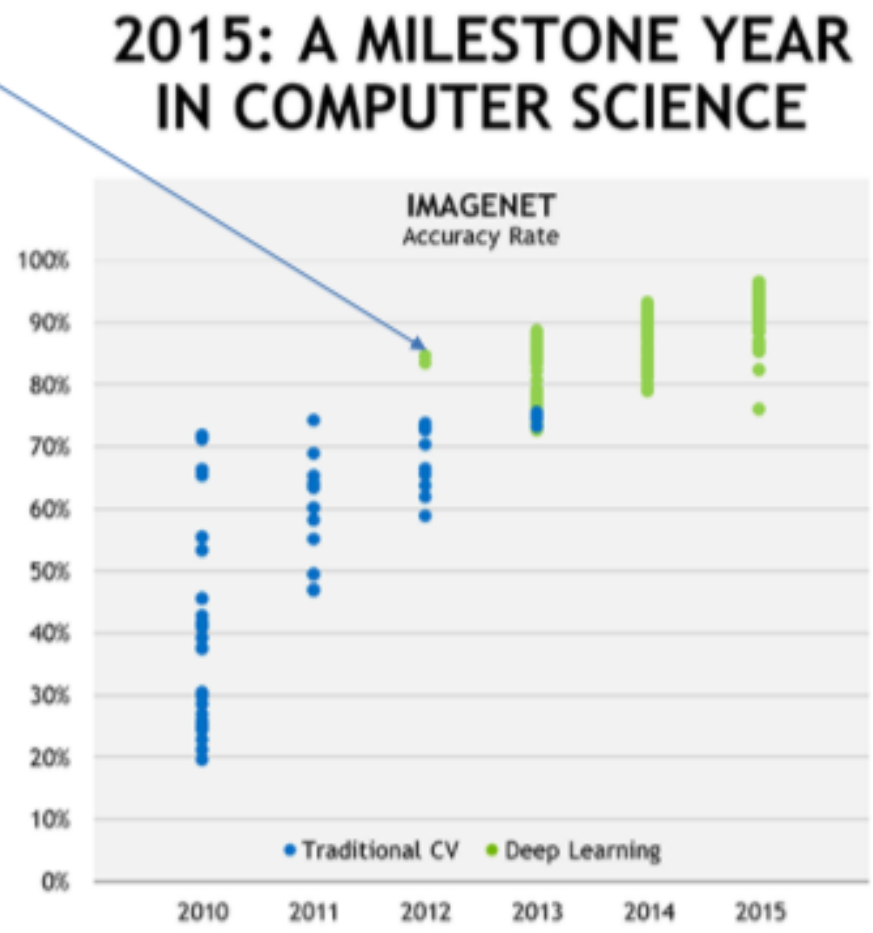
ImageNet = Lots and Lots and ... and Lots of Data

Example output (classification problem)



ImageNet Competition

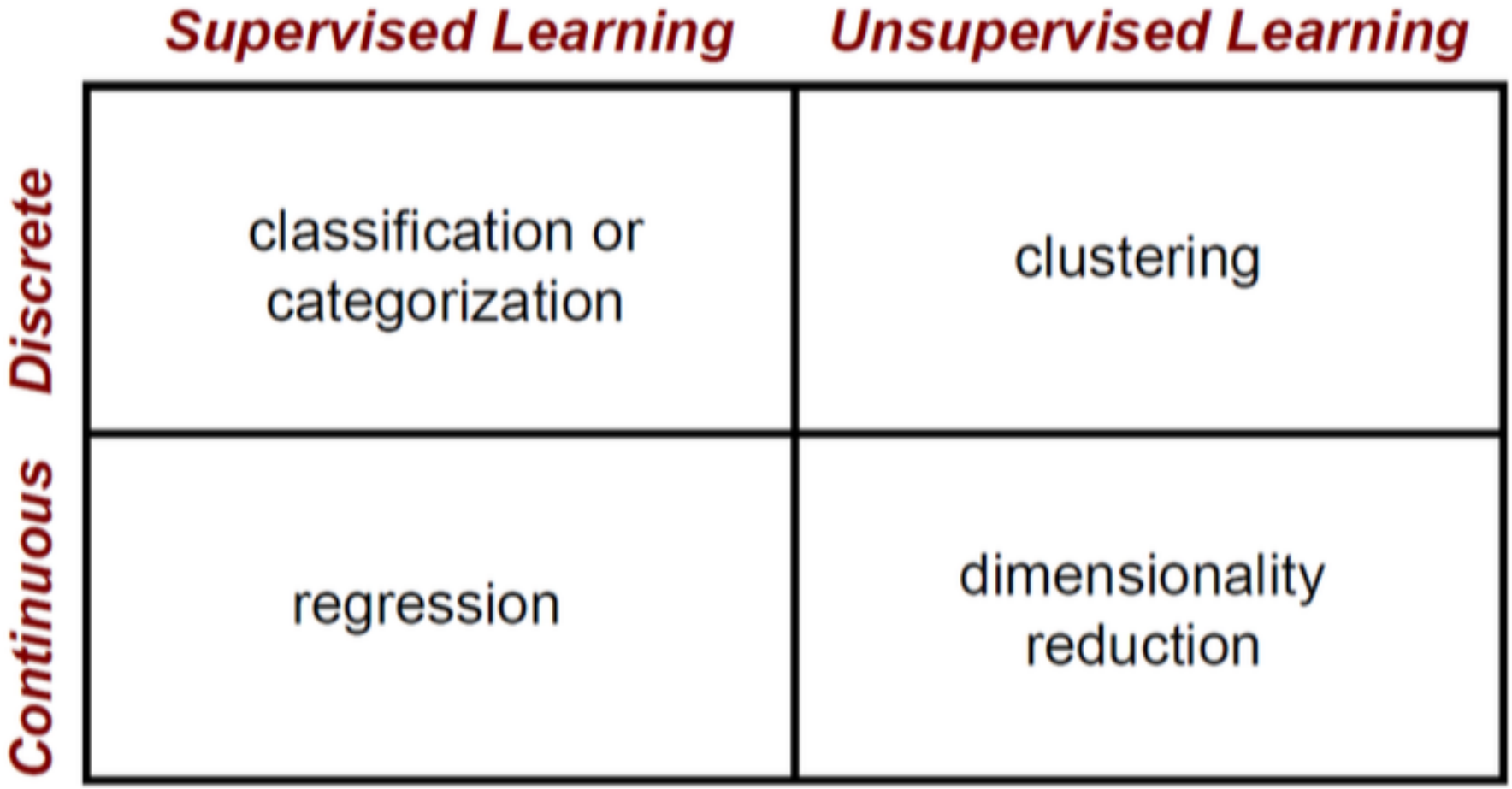
- Krizhevsky, 2012
- Google, Microsoft 2015
  - Beat the best human score in the ImageNet challenge.



NVIDIA

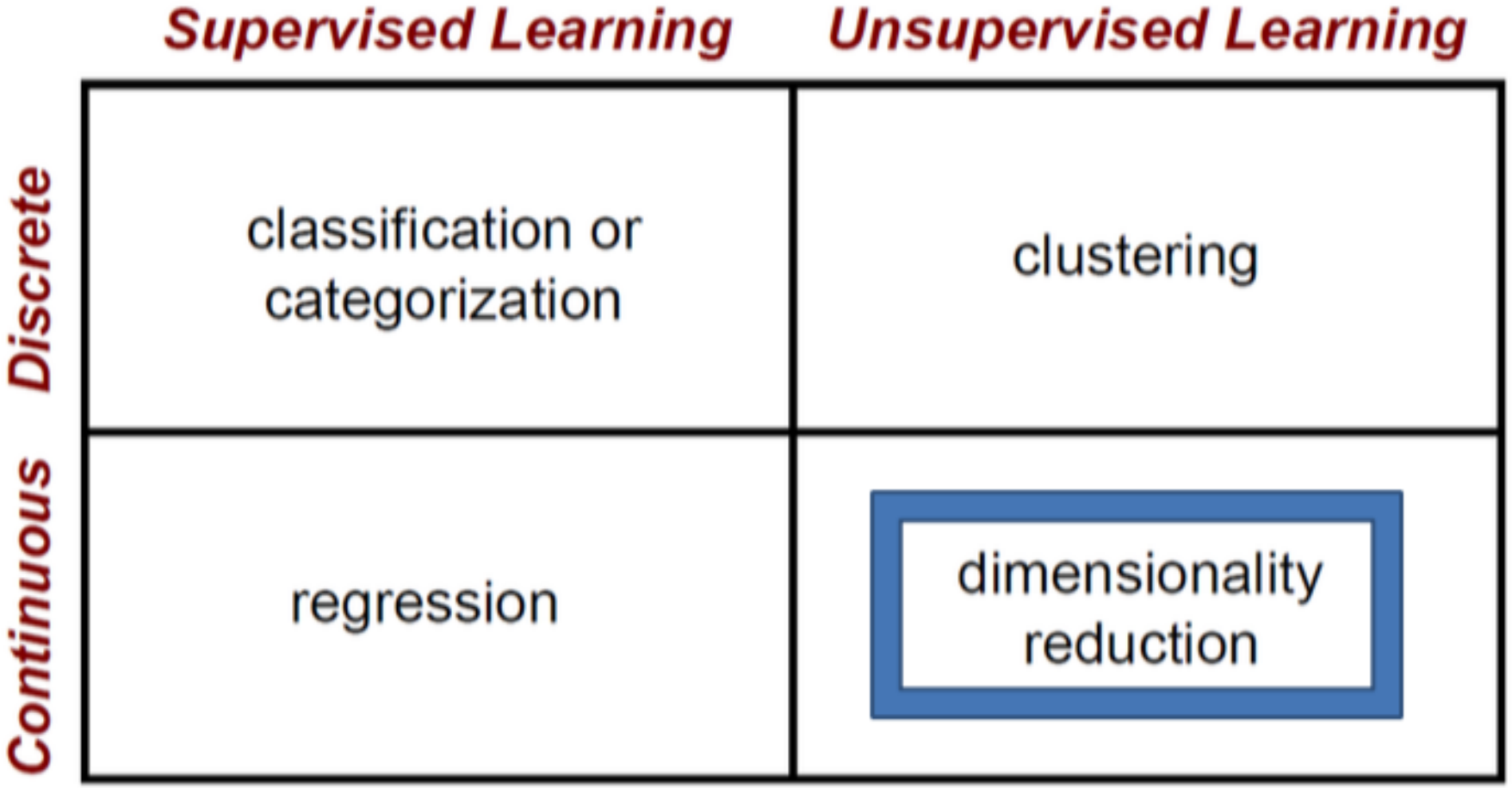
How can ML solve these types of problems? (or at a minimum help?)

- Four broad categories:



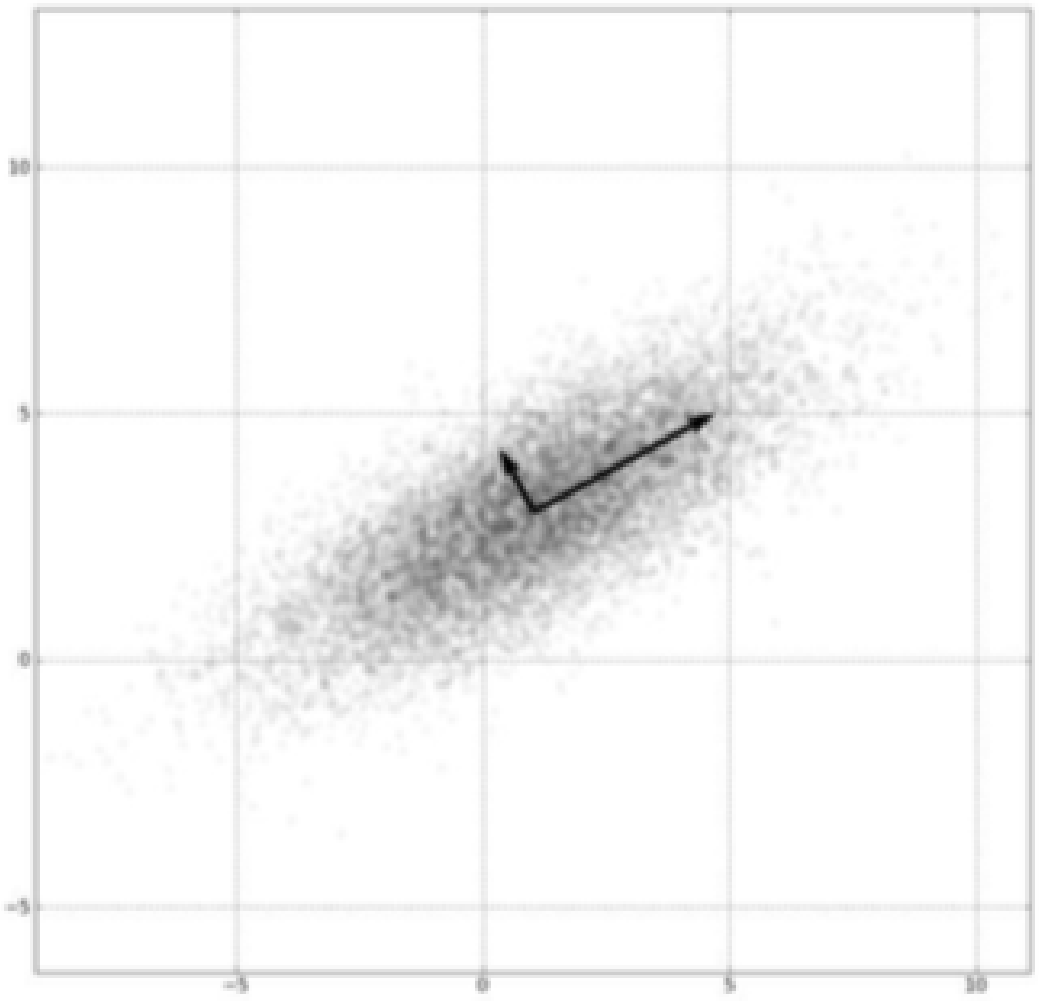
How can ML solve these types of problems? (or at a minimum help?)

- Dimensionality reduction (Closely related to Project 4 - more to come!)



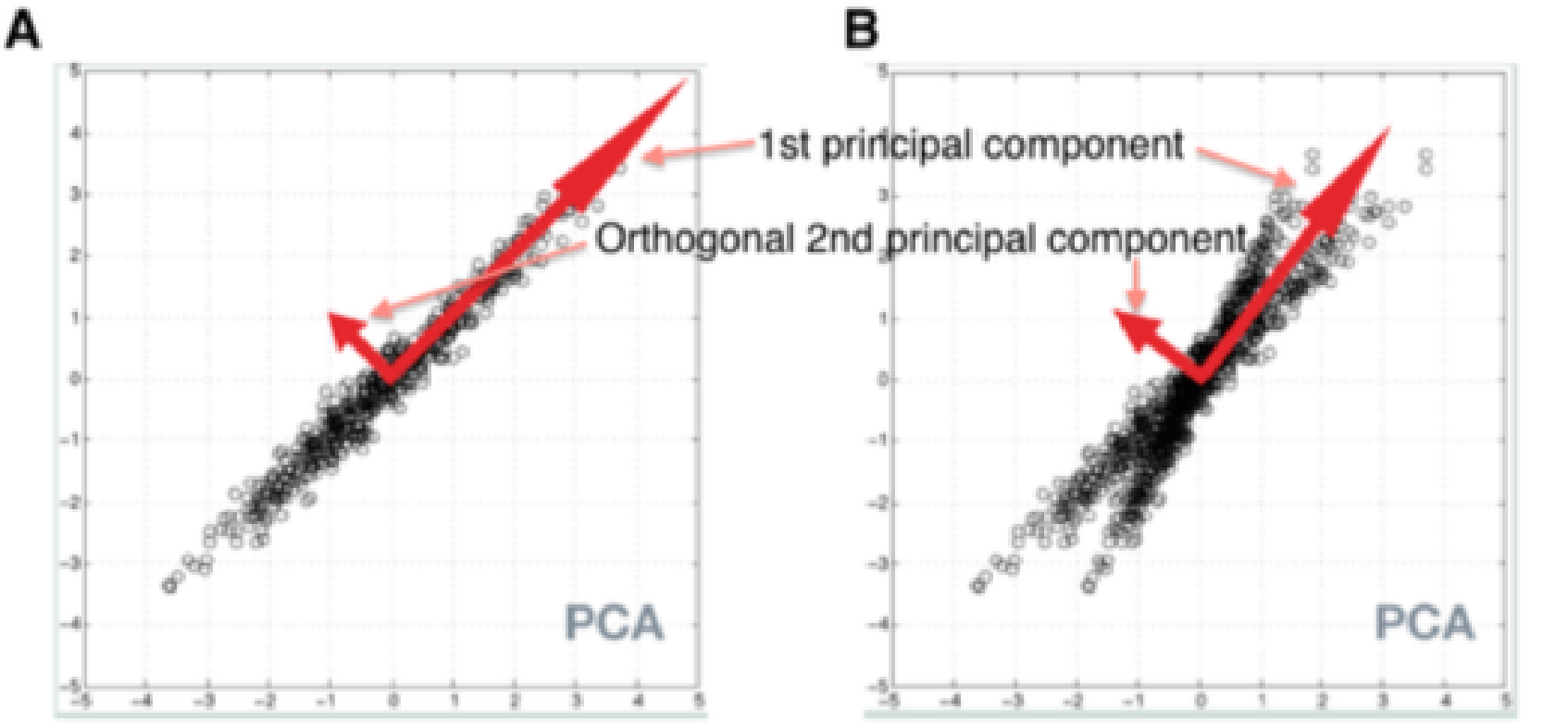
Many different approaches to Dimensionality Reduction

- **PCA**, ICA, LLE, ISOMAP, etc.
- Principal Component Analysis (PCA)
  - Arguably the most common approach (at least to start with)
  - Idea: Find a reduced dimensional subspace that maximizes the total variance in the data.
  - PCA finds an orthogonal subspace that finds correlation in the data: *best possible* low-dimensional subspace when using linear projections.



PCA

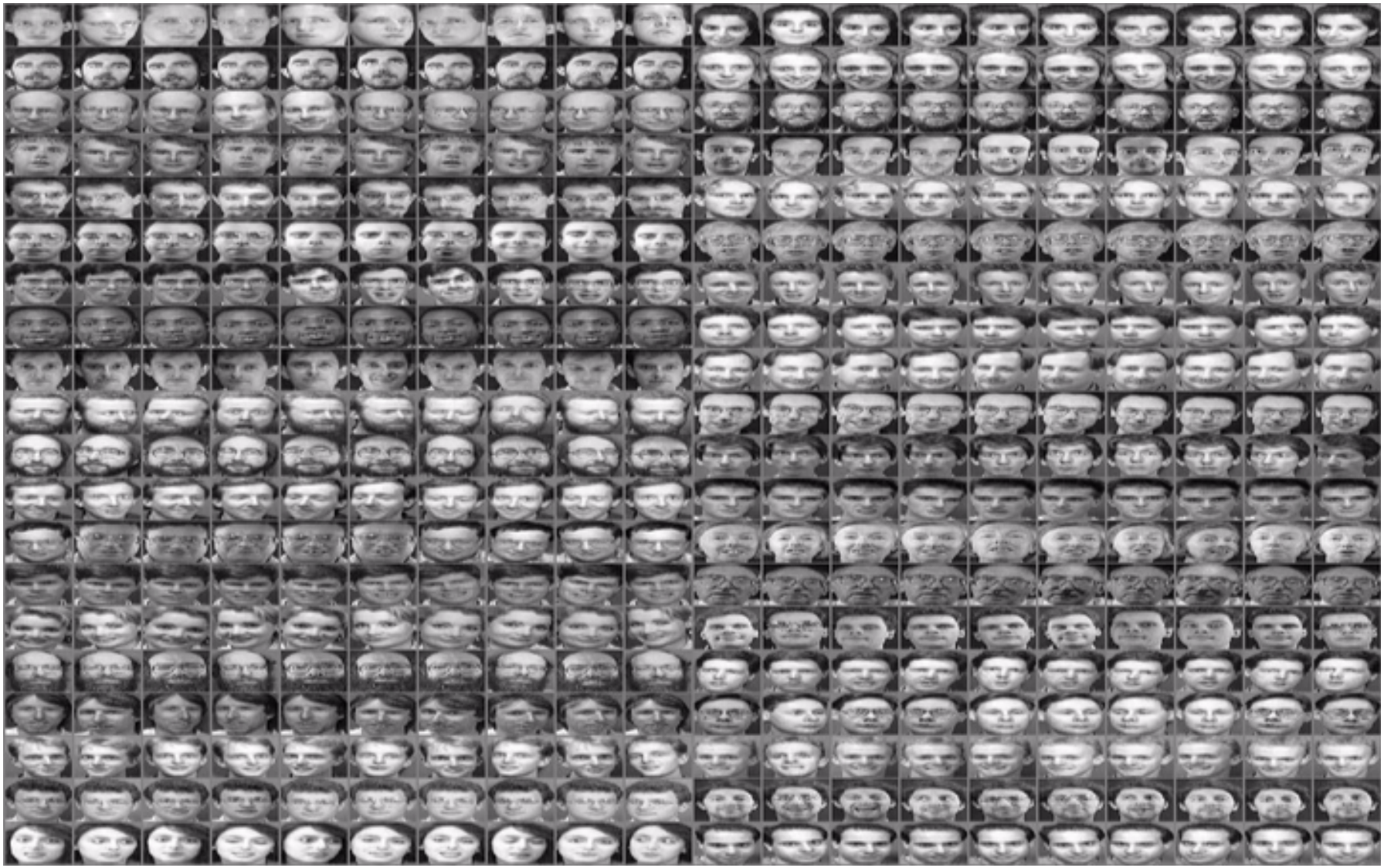
- We will dive into this in great detail in the next project!



(Figure adapted from C. Beckmann, Oxford FMRIB)

Application to Imaging: Eigenfaces for Recognition

- AT&T Face Database: 10 images of 40 different subjects.



Eigenfaces

Mean face

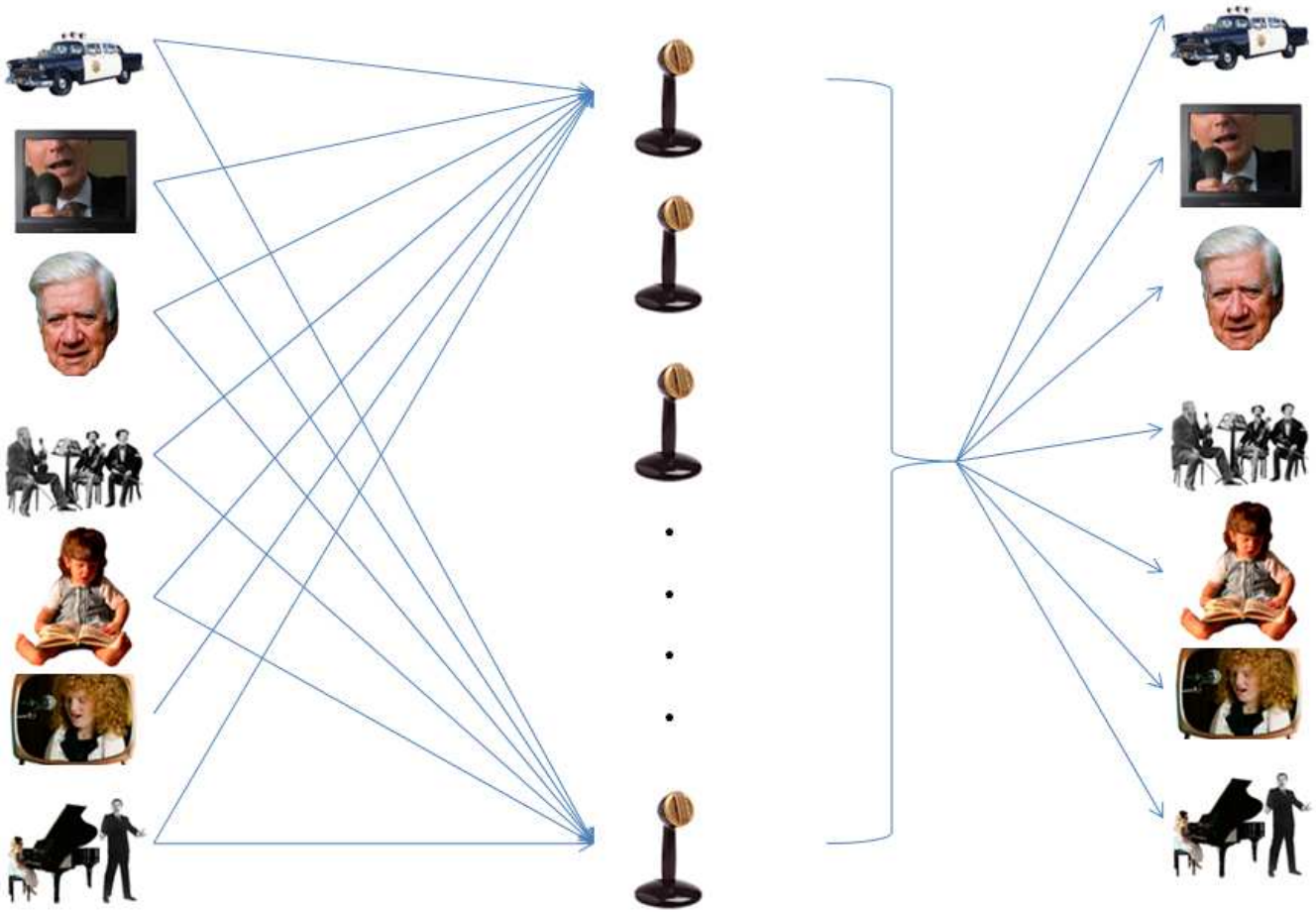
Basis of variance (eigenvectors)

M. Turk; A. Pentland (1991). "Face recognition using eigenfaces" (PDF).
Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–591.

R.P.W. Duin

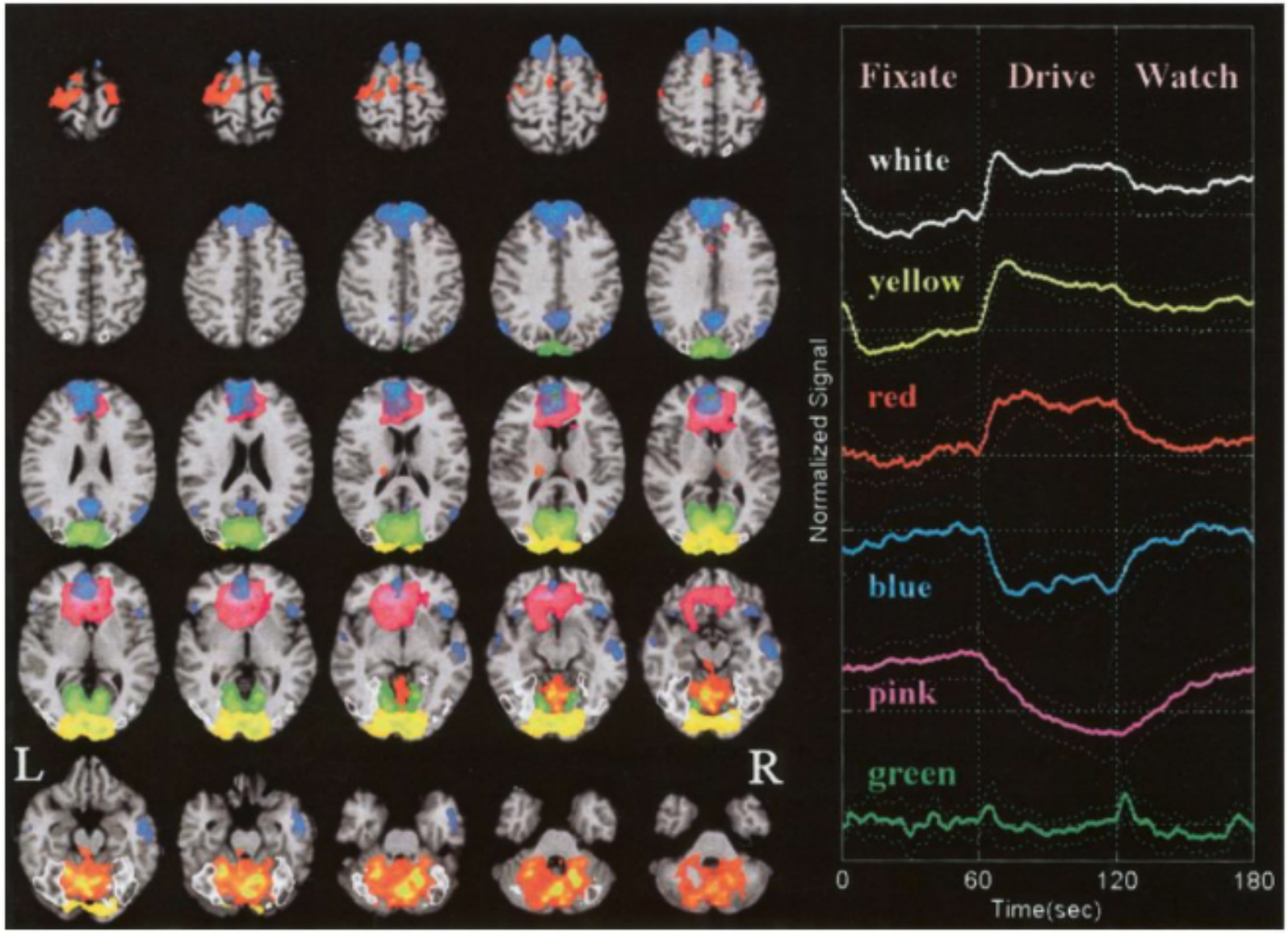
ICA - Blind Source Seperation

- Classically used to solve the "cocktail" party problem!



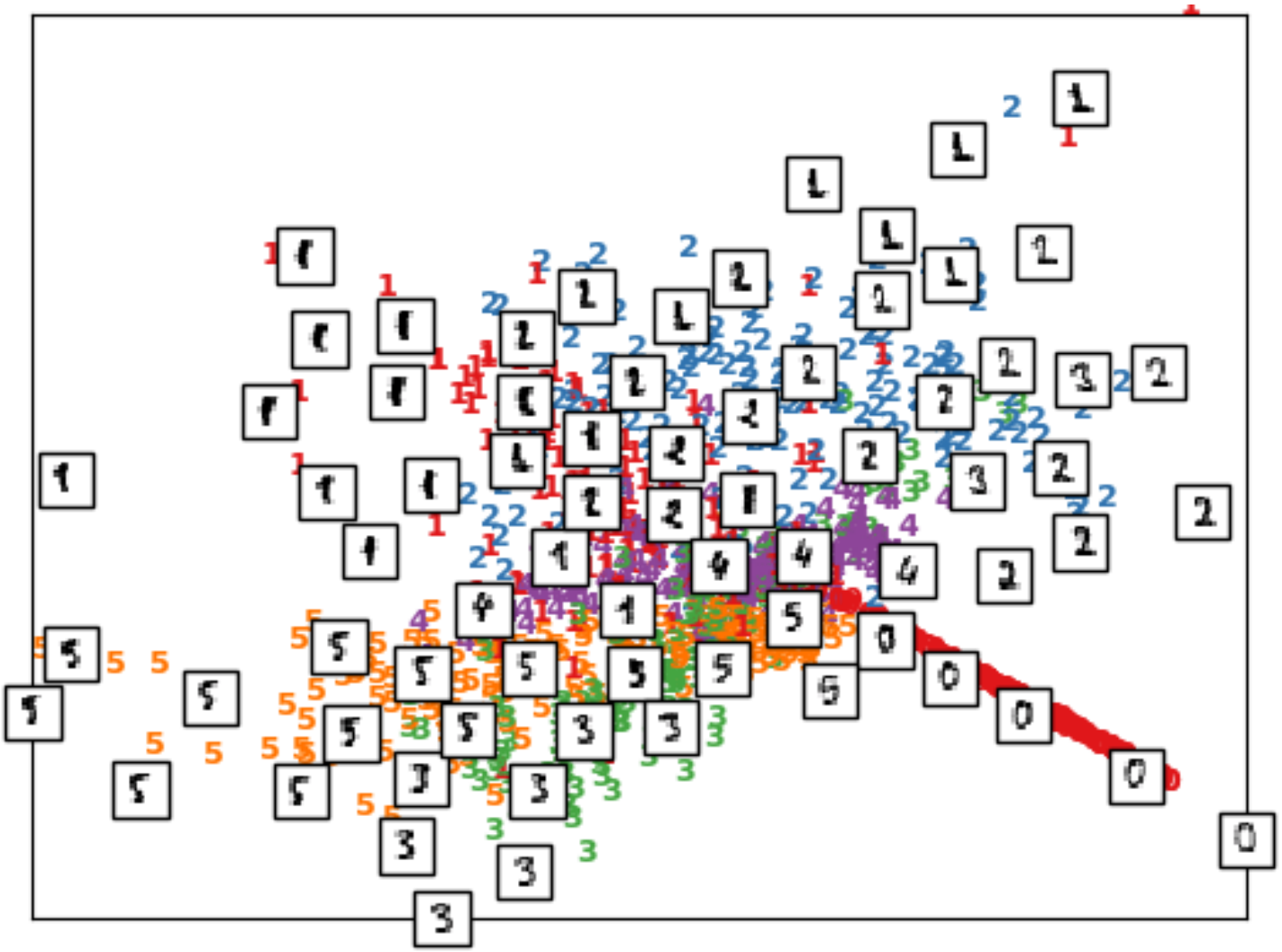
ICA - Blind Source Seperation

- Classically used to solve the "cocktail" party problem!



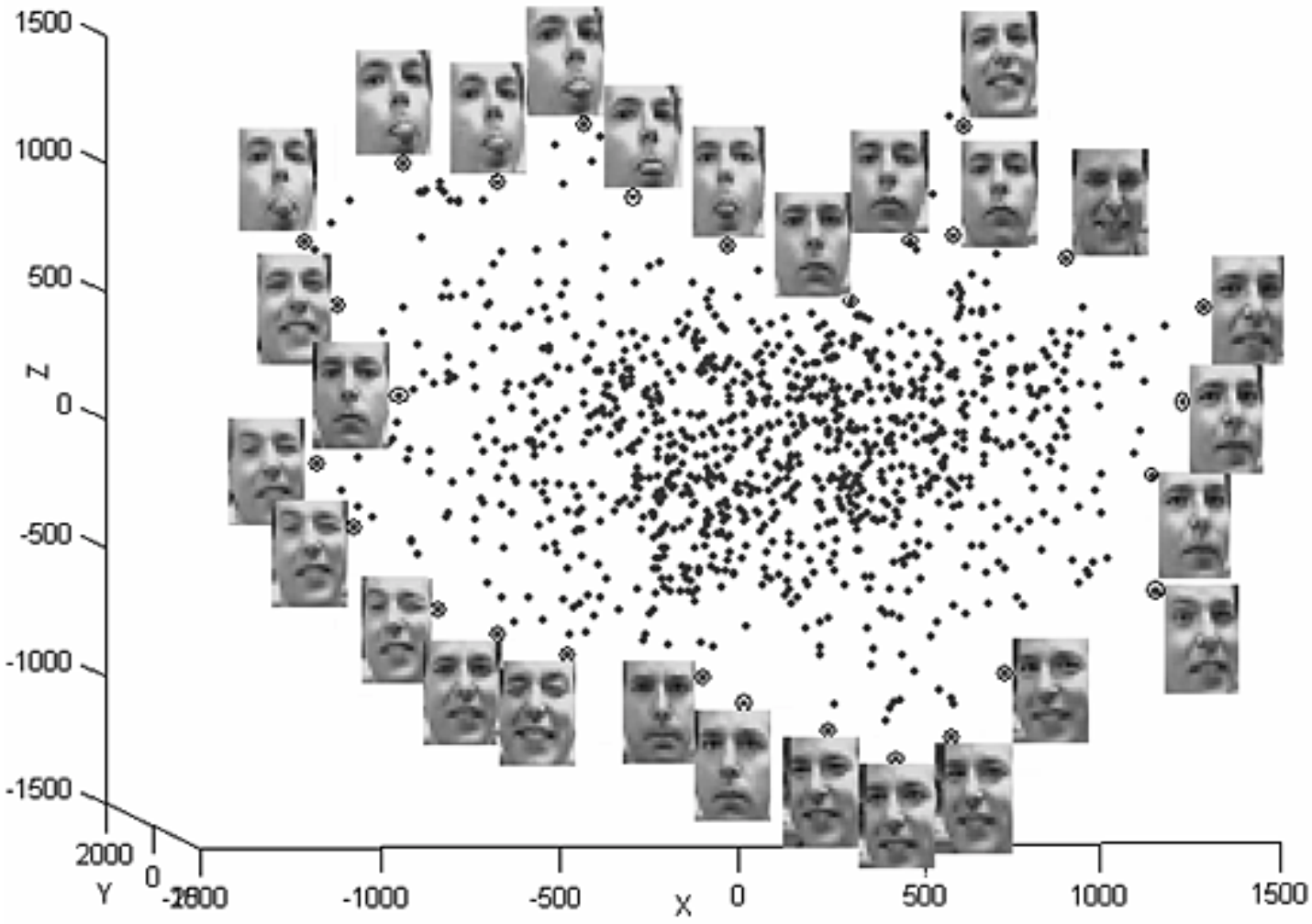
Local Linear Embedding (LLE)

- Think global - Fit local (Manifold learning approach)
- Looks at local linear planes in the data for "learning" manifold structure



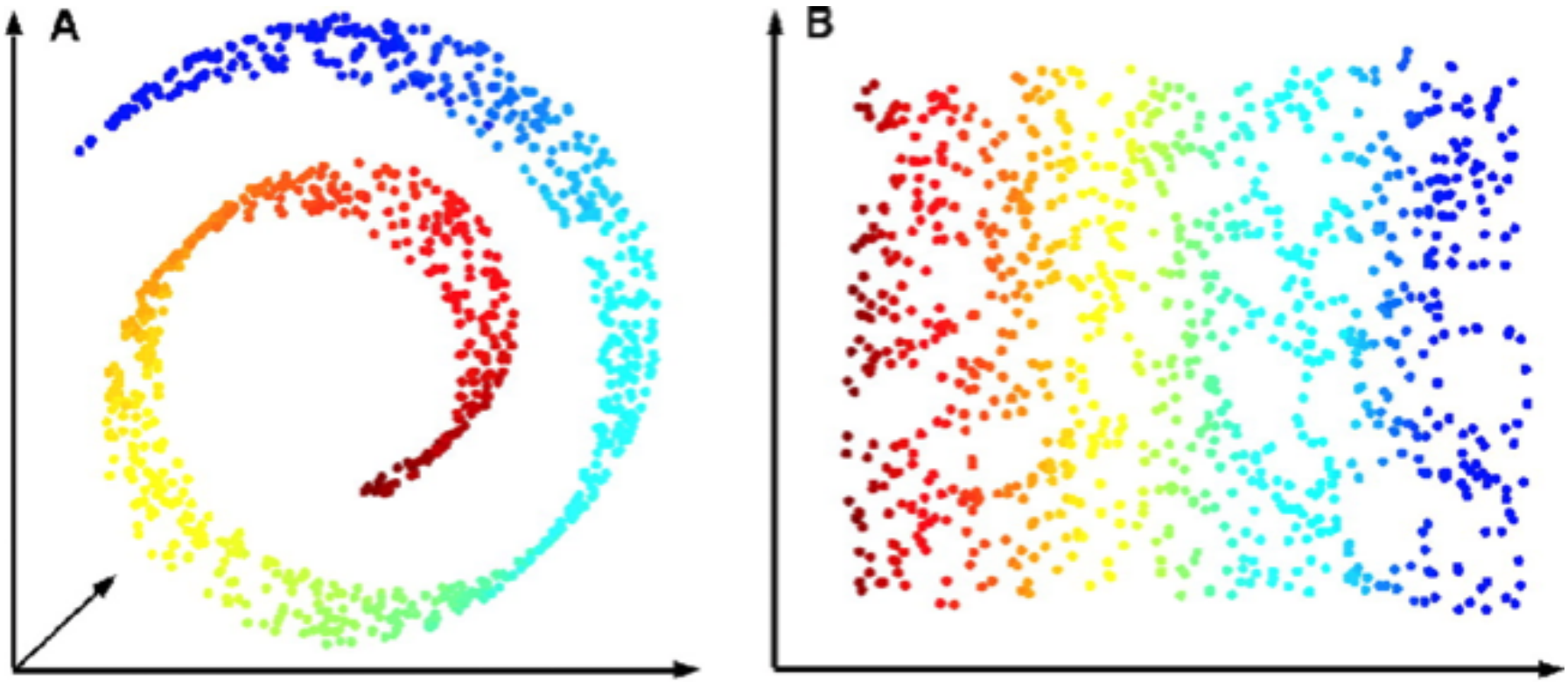
Local Linear Embedding (LLE)

- Think global - Fit local (Manifold learning approach)
- Looks at local linear planes in the data for "learning" manifold structure



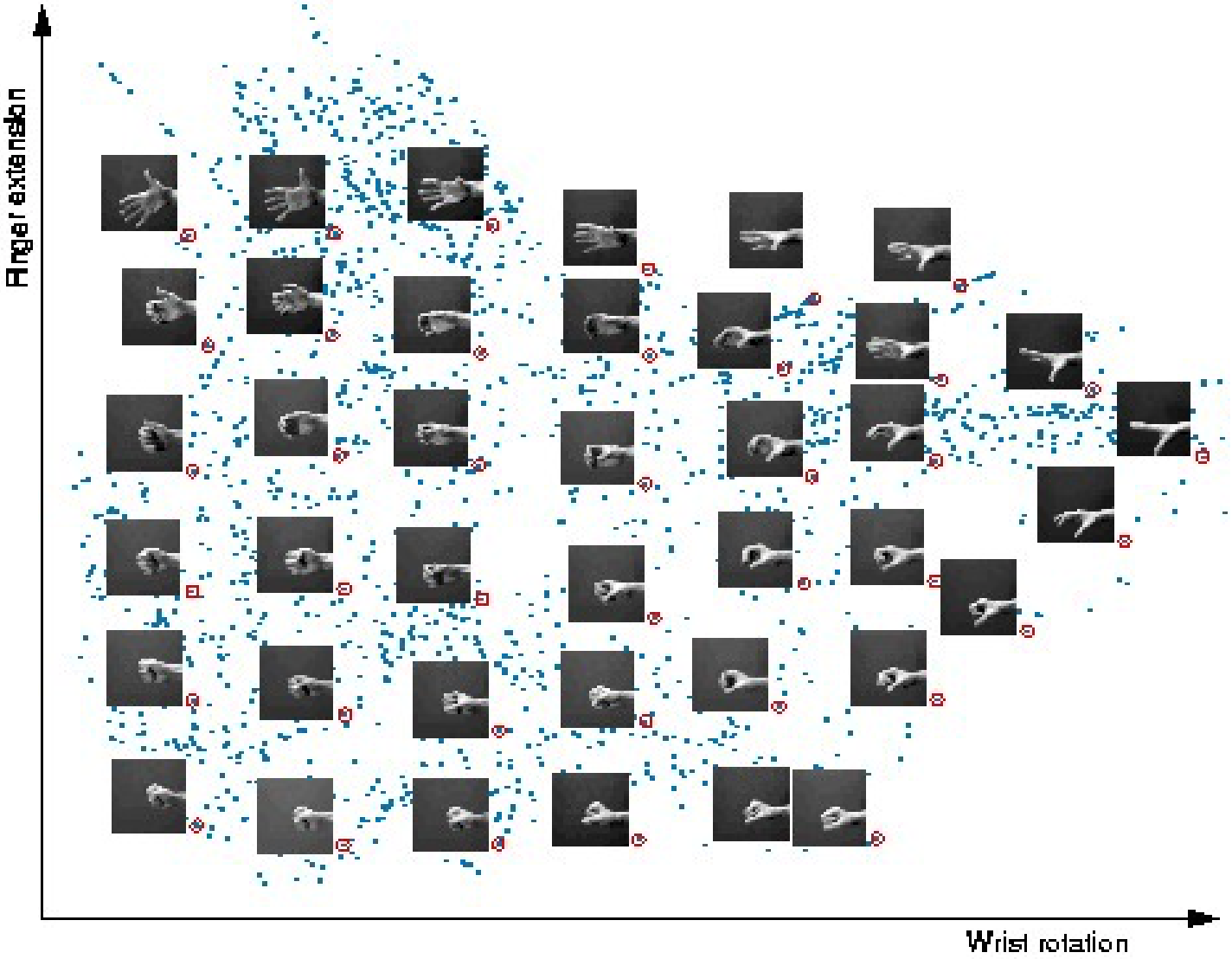
Isometric Mapping (ISOMAP)

- Maintain local isometries in the data (geodesic distance as opposed to Euclidean)



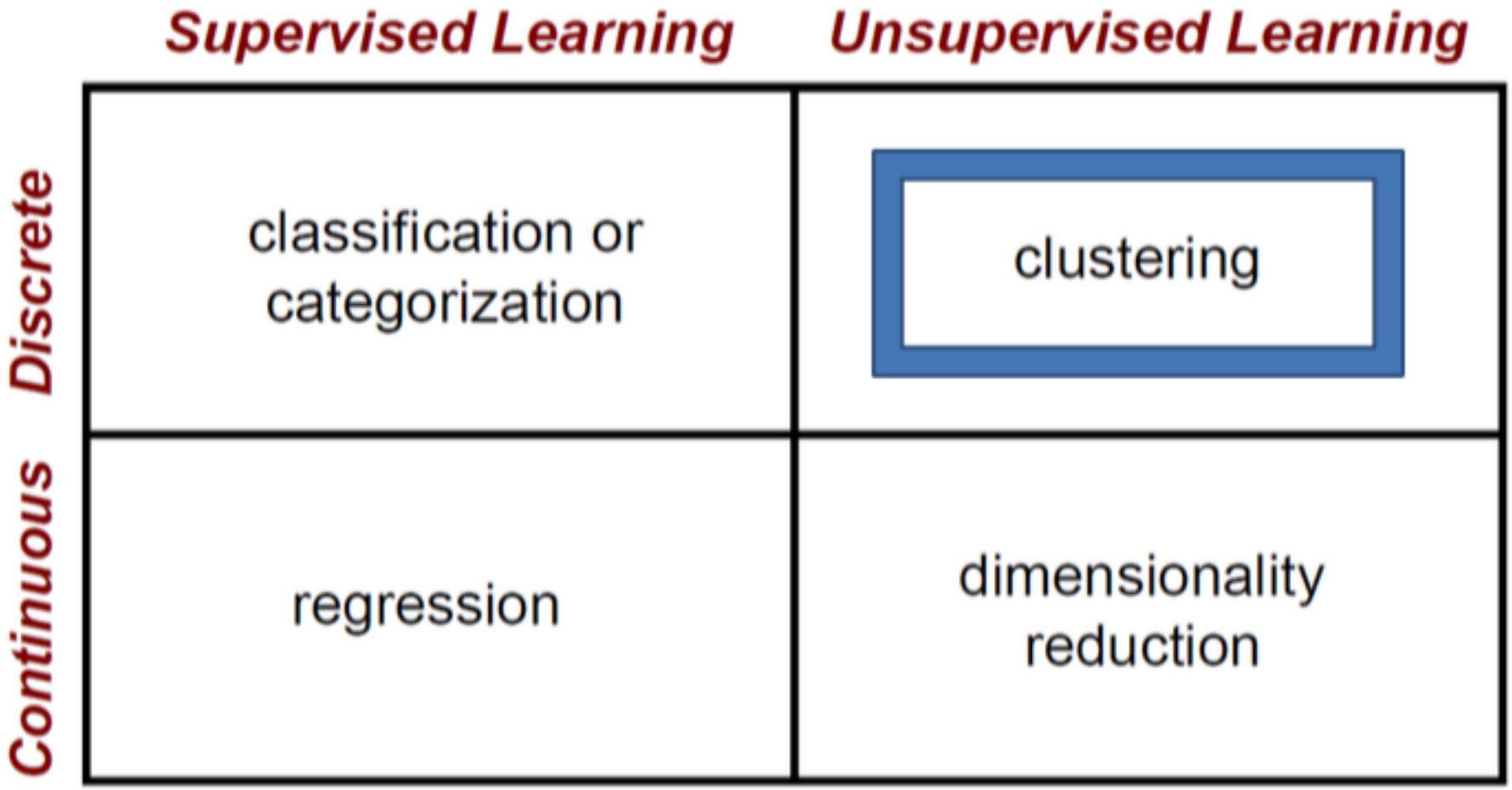
Isometric Mapping (ISOMAP)

- Maintain local isometries in the data (geodesic distance as opposed to Euclidean)



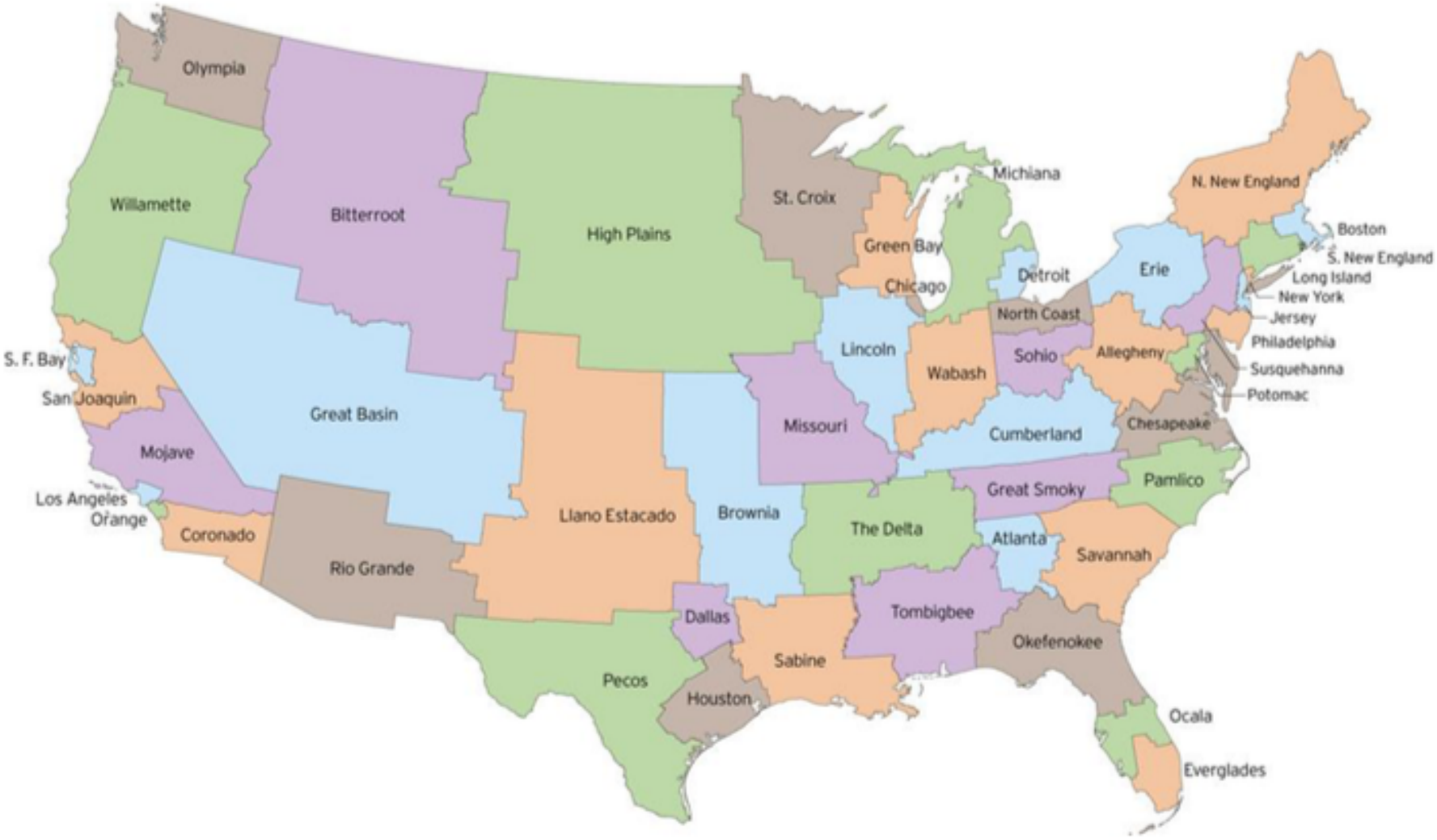
How can ML solve these types of problems? (or at a minimum help?)

- Data clustering



Example 1 - Mapping

Example 1 - Mapping



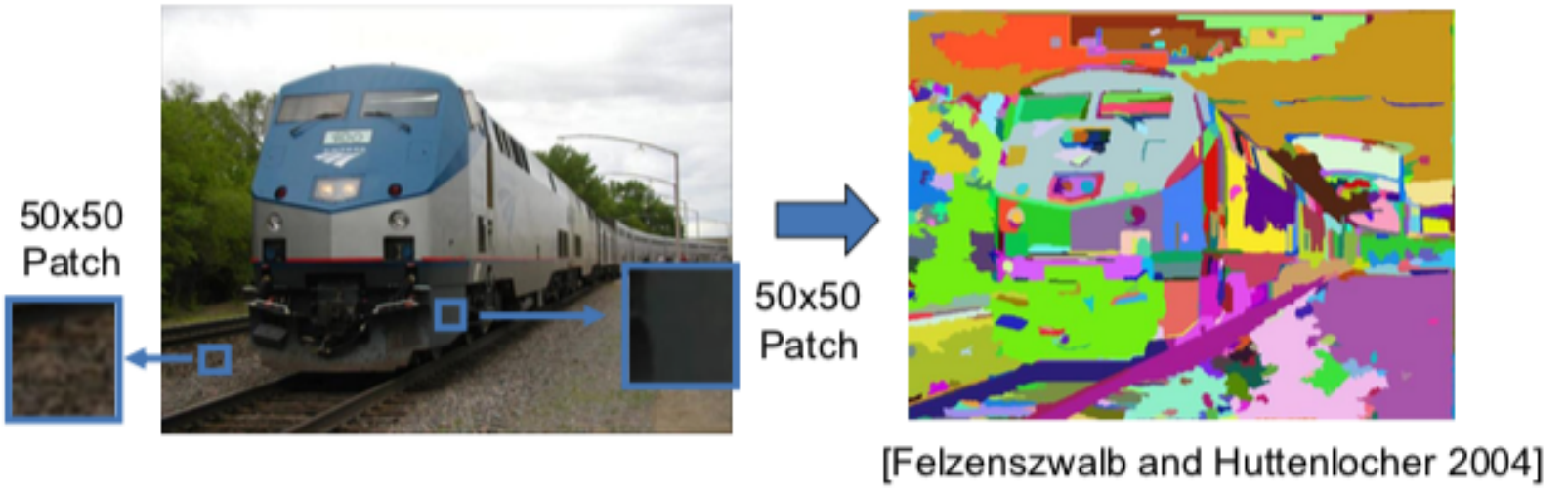
Example 1 - Mapping

Example 2 - Segmentation

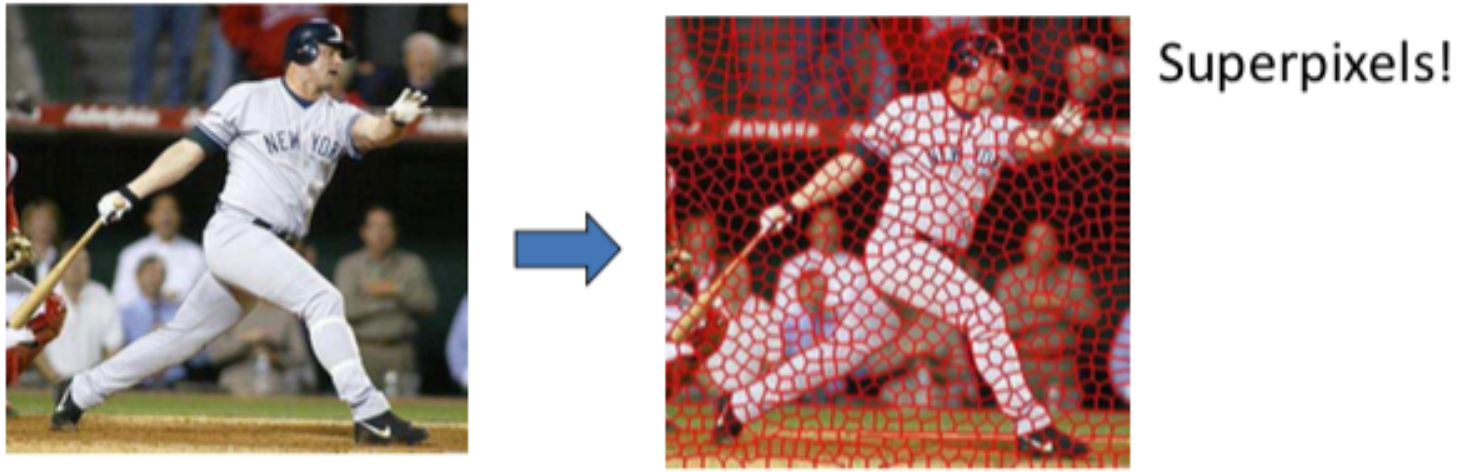
- Break up image into meaningful/similar regions.

Example 2 - Segmentation

- Can help feature support/region support.



[Felzenszwalb and Huttenlocher 2004]



[Hoiem et al. 2005, Mori 2005]

[Shi and Malik 2001]

Derek Hoiem

Example 2 - Segmentation

- Clean cropping (GrabCut: Rother *et al.*, 2004)




Clustering: Basic idea

- 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: Why do we Cluster?

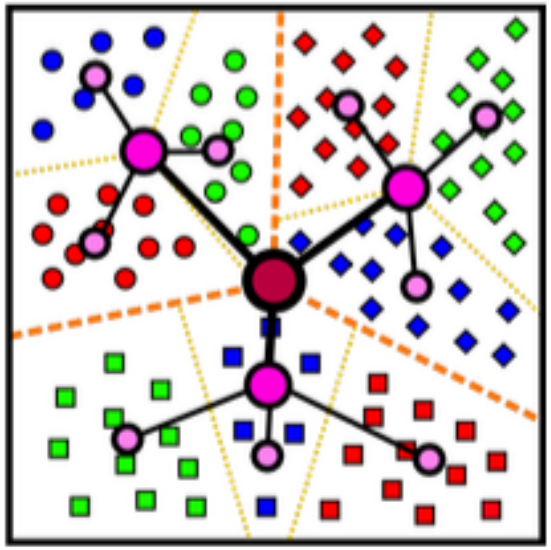
- Summarize/Visualize Data:
  1. Look at large amounts of data
  2. Patch-based compression or denoising
  3. Represent a large continuous vector with the cluster number
- Counting
  1. Histograms of texture, color, SIFT vectors
- Segmentation
  1. Separate the image into different regions
- Prediction/Classification
  1. Images in the same cluster may have the same labels

How do we cluster?

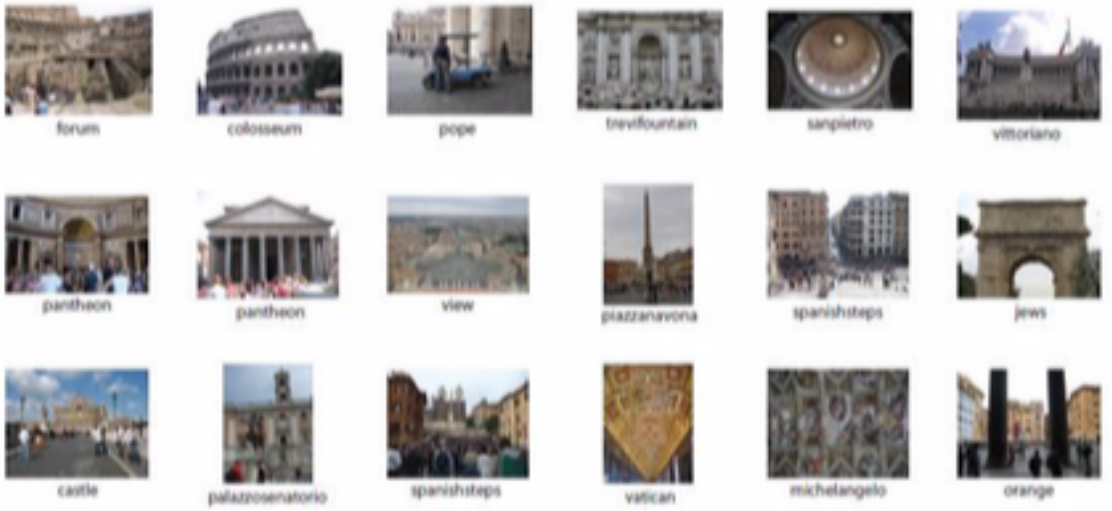
- K-means
  1. Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  1. Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  1. Estimate modes of pdf
- Spectral clustering
  1. Split the nodes in a graph based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

Applications for each: K-Means

- Quantization/Summarization: K-means
  1. Aims to preserve variance of original data
  2. Can easily assign new point to a cluster



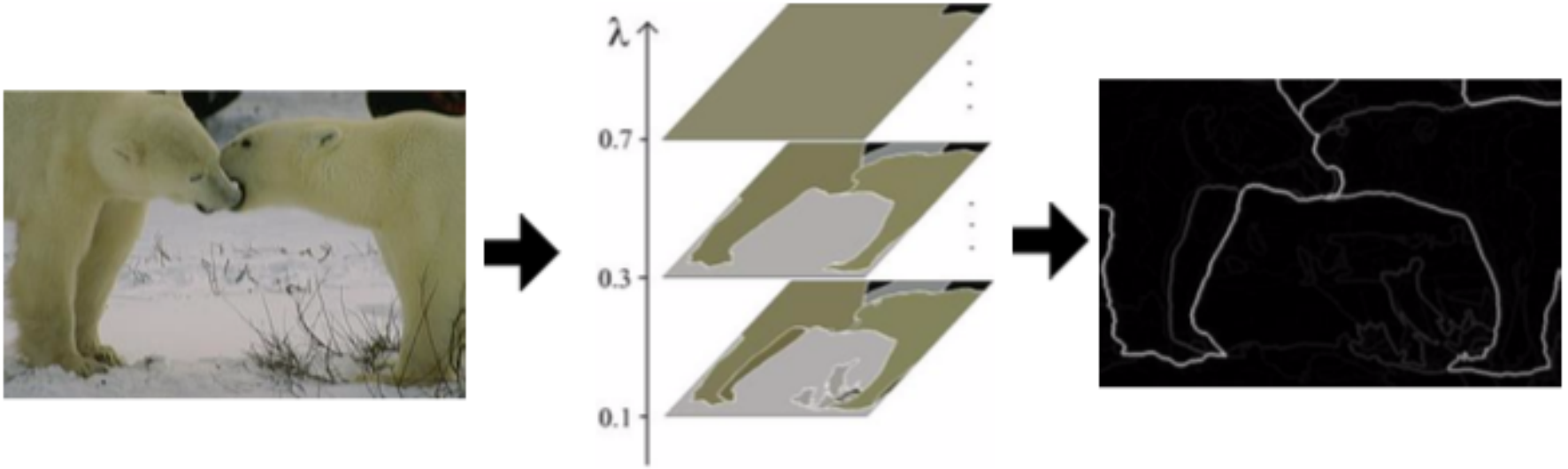
Quantization for computing histograms



Summary of 20,000 photos of Rome using "greedy k-means"
http://grail.cs.washington.edu/projects/canonview/

Applications for each: Agglomerative

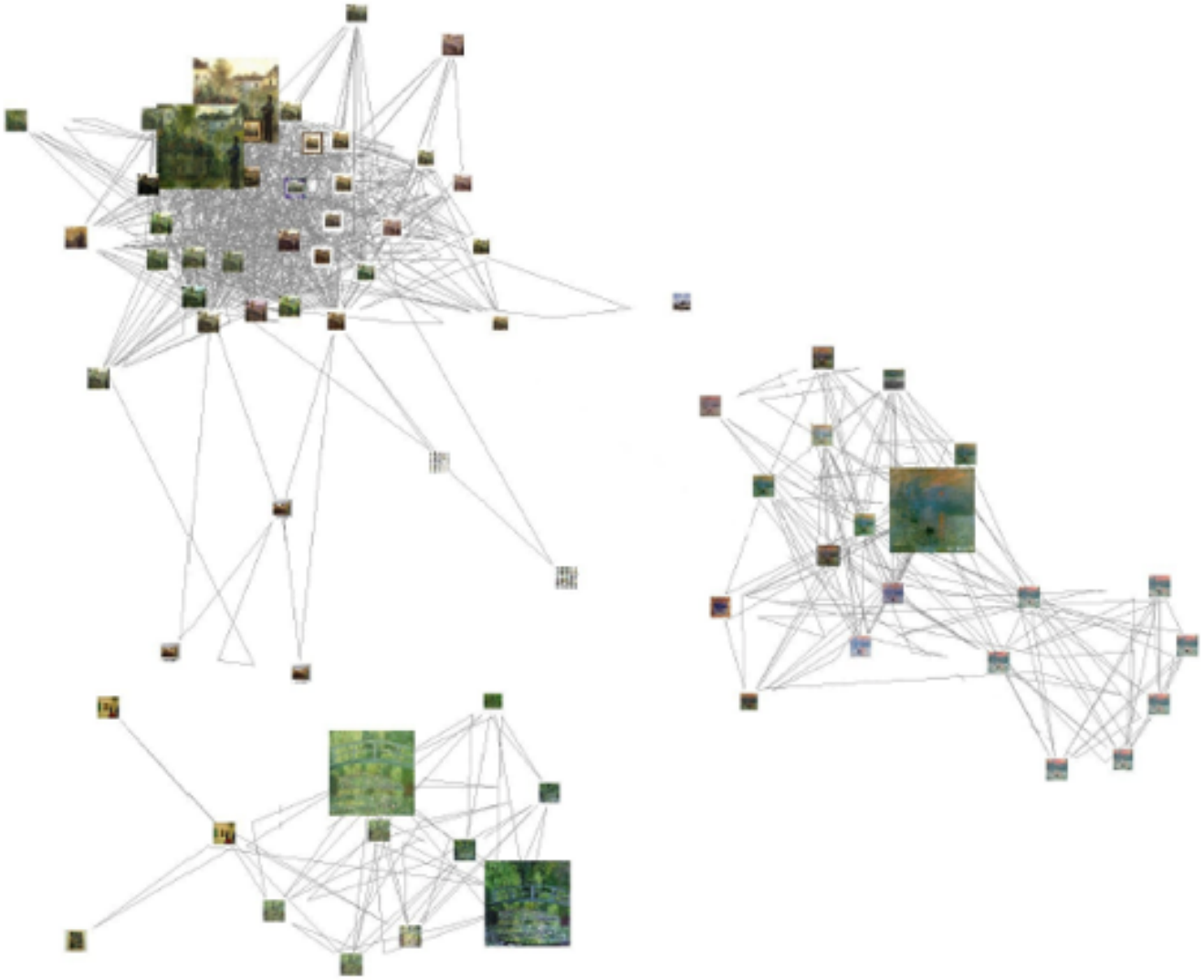
- Image segmentation: Agglomerative clustering
  1. More flexible with distance measures (e.g., can be based on boundary prediction)
  2. Adapts better to specific data
  3. Hierarchy can be useful



http://www.cs.berkeley.edu/~arbelaez/UCM.html

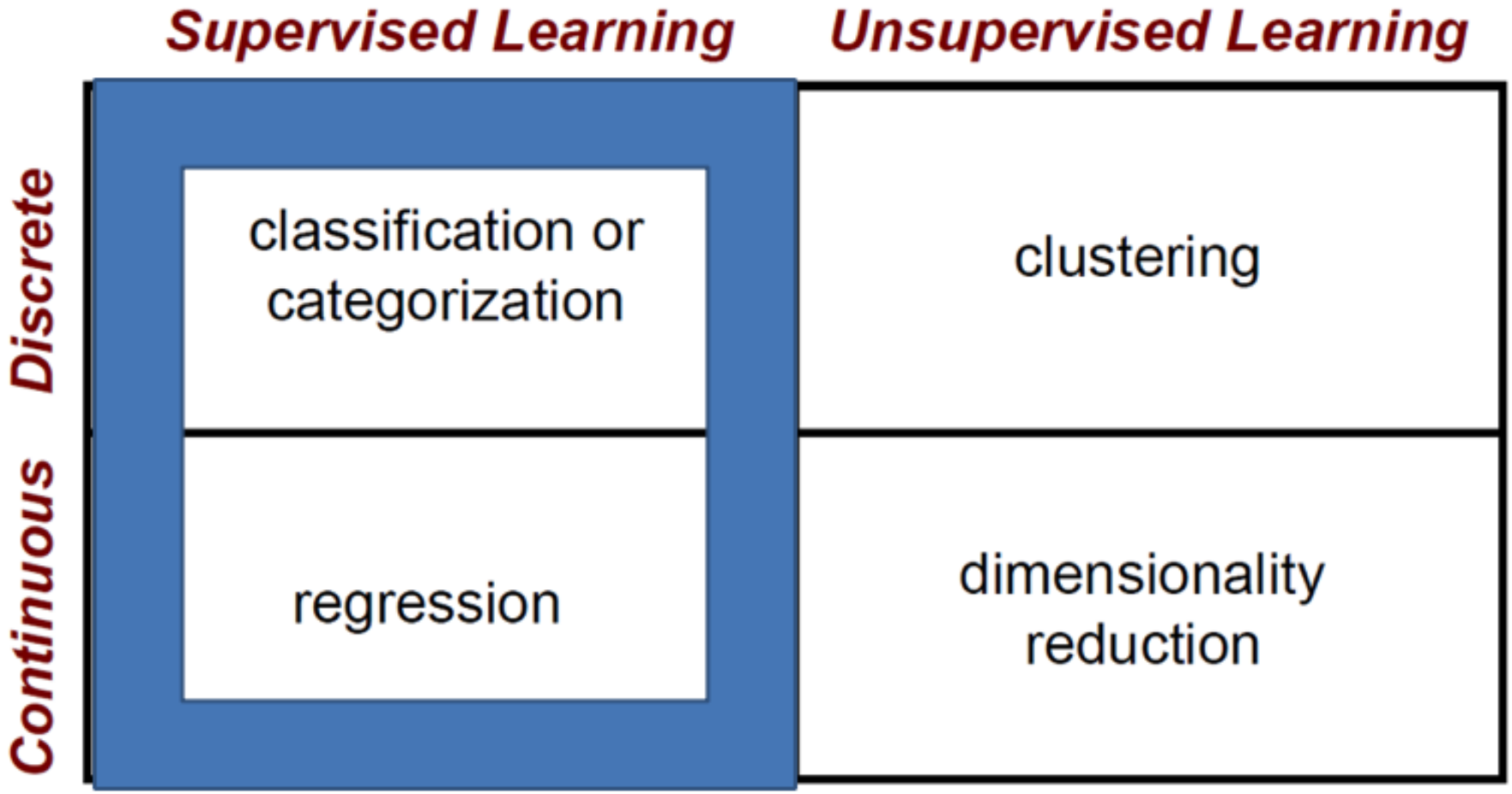
Applications for each: Spectral

- Image segmentation: Spectral clustering
  1. Useful for determining relevance of one object to another
  2. Segmentation



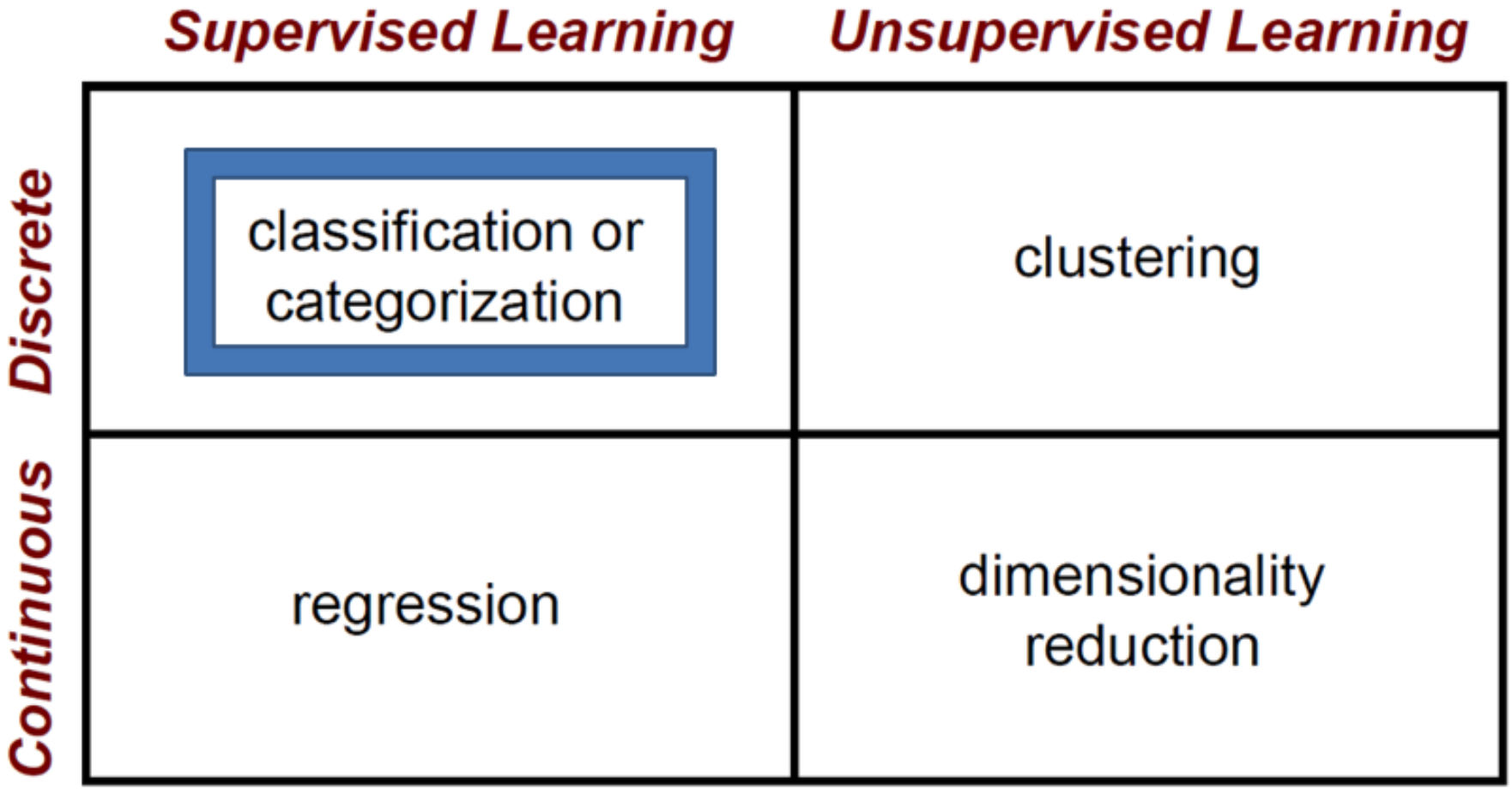
This is all unsupervised - what about supervised?

- What's the difference????



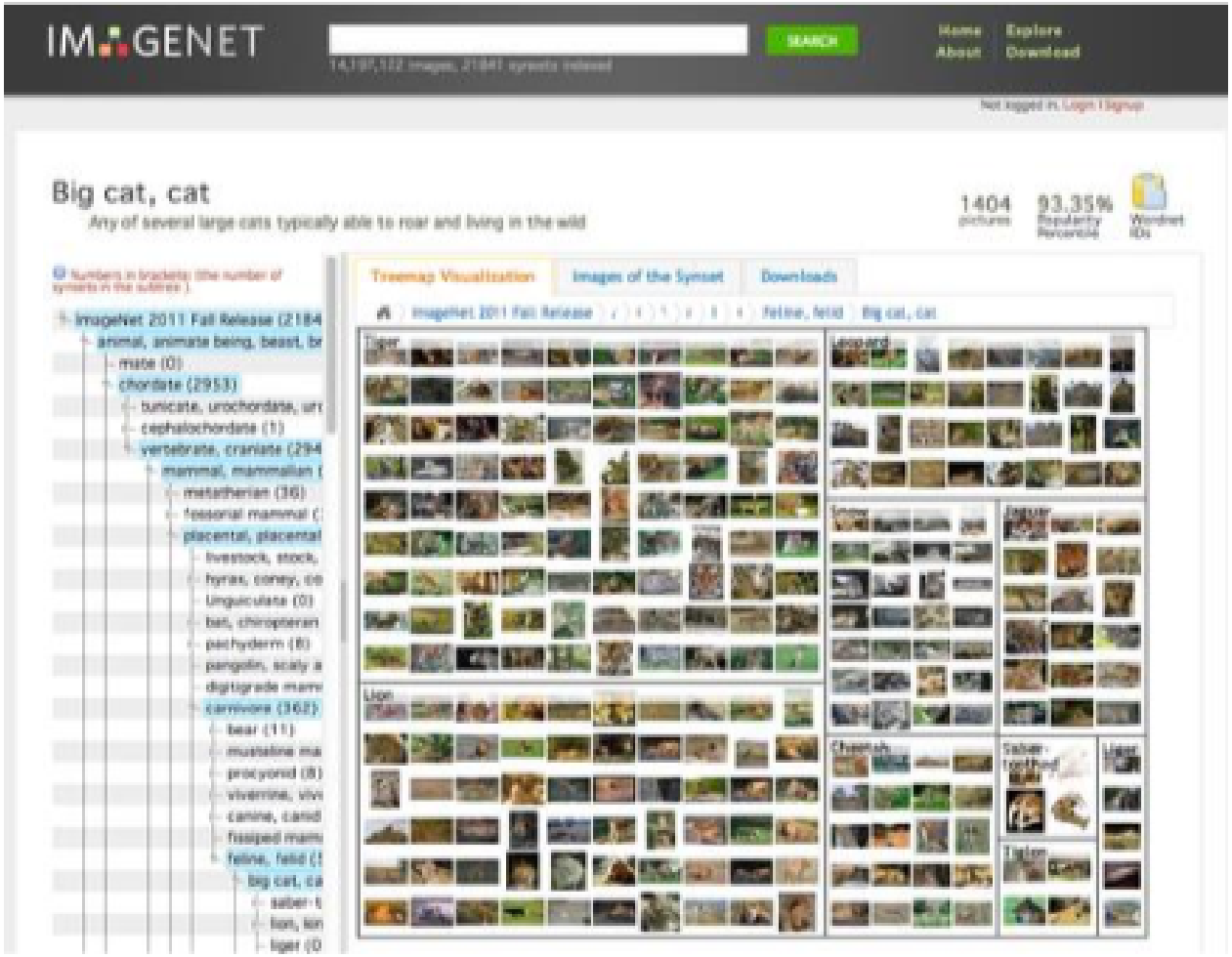
Let's have a look at classification

- Need some "additional" information: classes, labels, etc.



Recall: ImageNet

- Images for each category of WordNet
- 1000 classes
- 1.2mil images
- 100k test
- Top 5 error



Data Separation: Train, Validate, Test

- Need to separate the data into different categories for each class

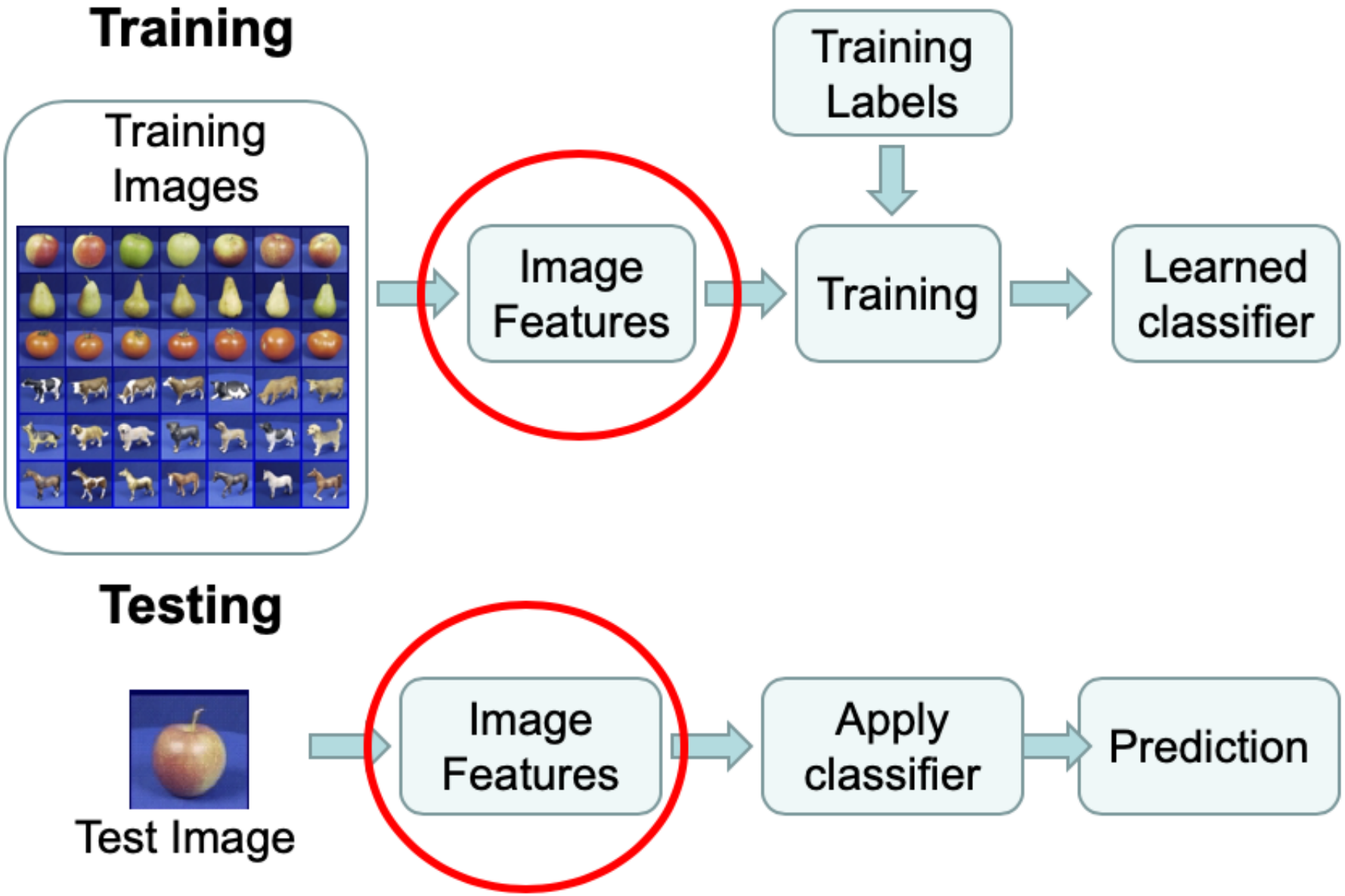
- Train classifier

- Measure error
- Tune model hyperparameters

- Secret labels
- Measure error

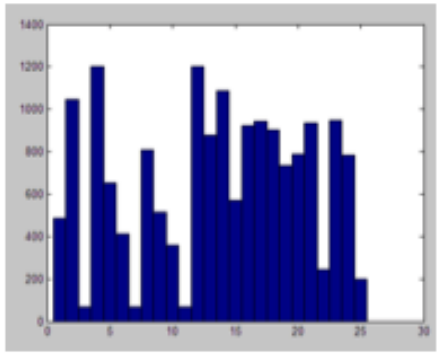
Random train/validate splits = cross validation

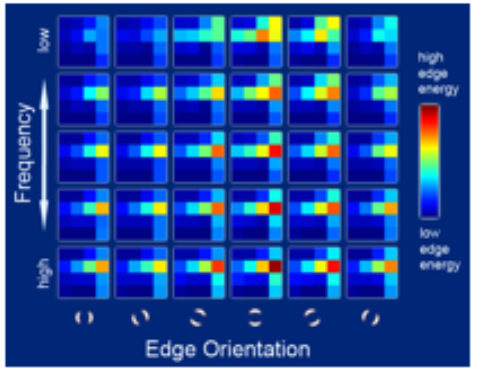
Classification Architecture: In General



What features???

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

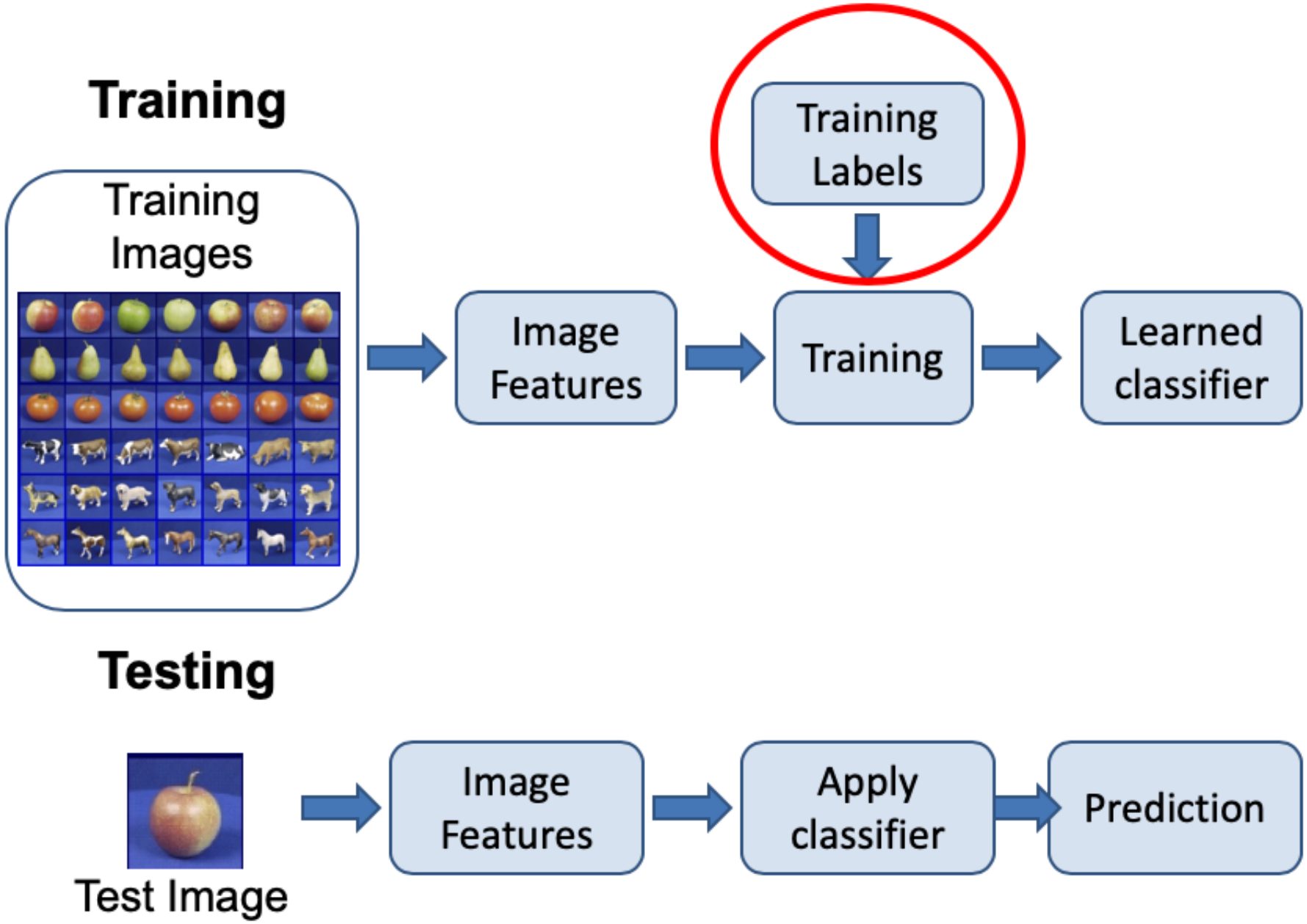




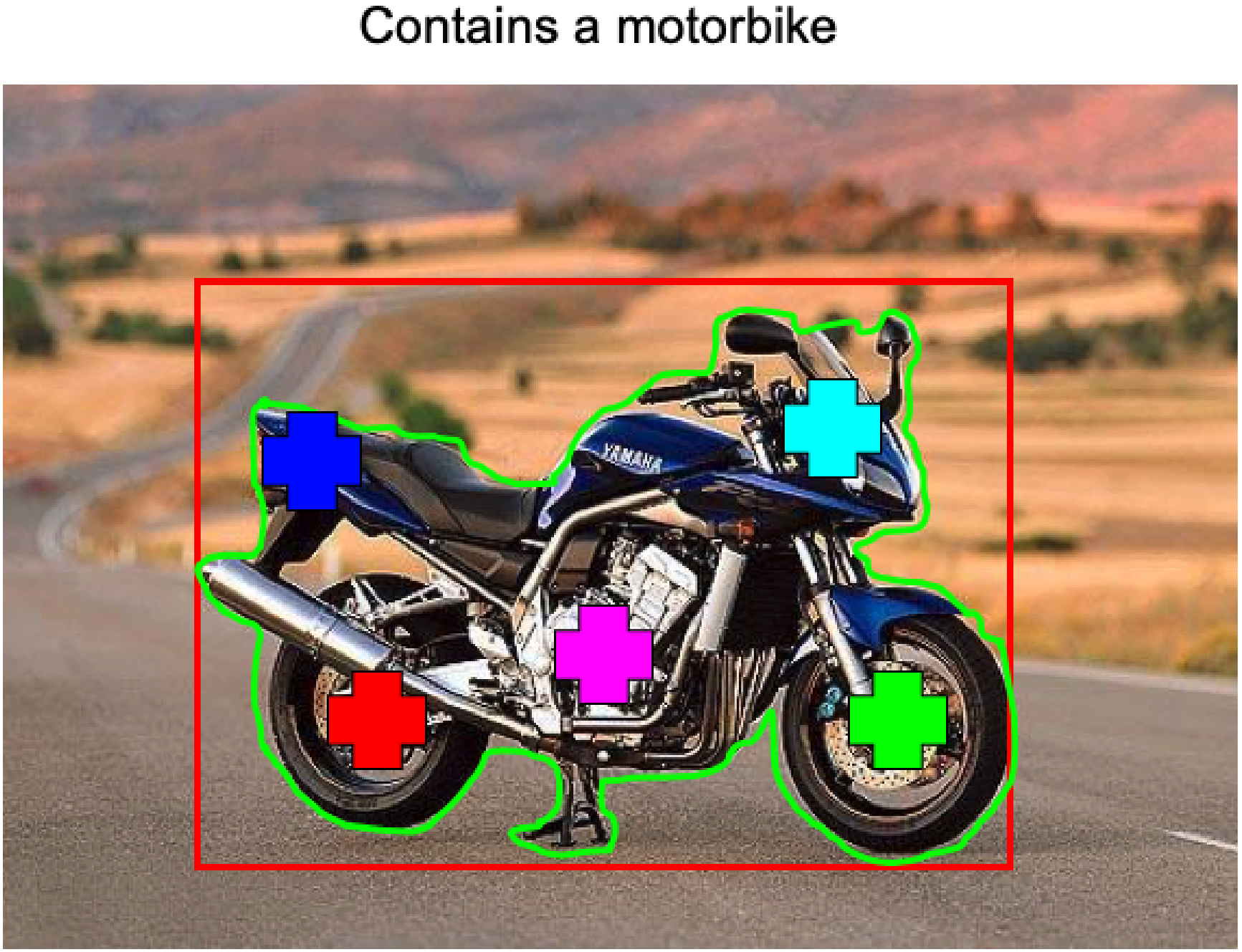
Data Representation for Classification

- Coverage
  - Ensure that all relevant info is captured
- Concision
  - Minimize number of features without sacrificing coverage
- Directness
  - Ideal features are independently useful for prediction

How do we "Label" classes?



Recognition $\rightarrow$ Data Association



Level of Supervision May Vary

'Semi-supervised': small partial labeling

Need to select "Good" training images

Good training example?

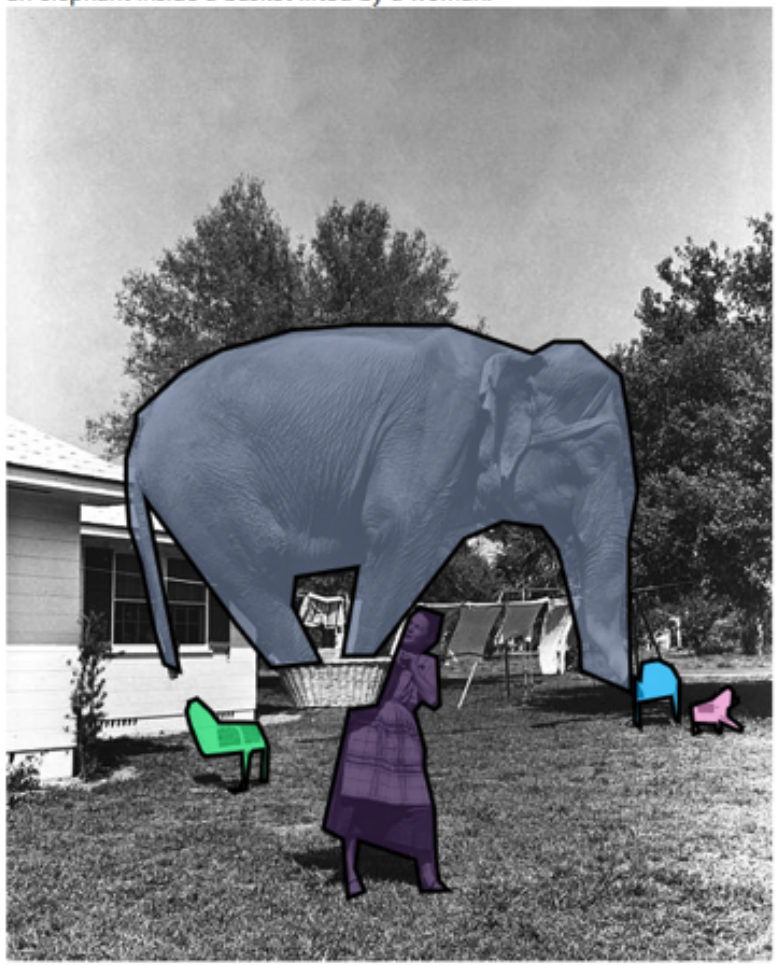
Need to select "Good" labels (MS-COCO)

http://mscoco.org/explore/?id=134918

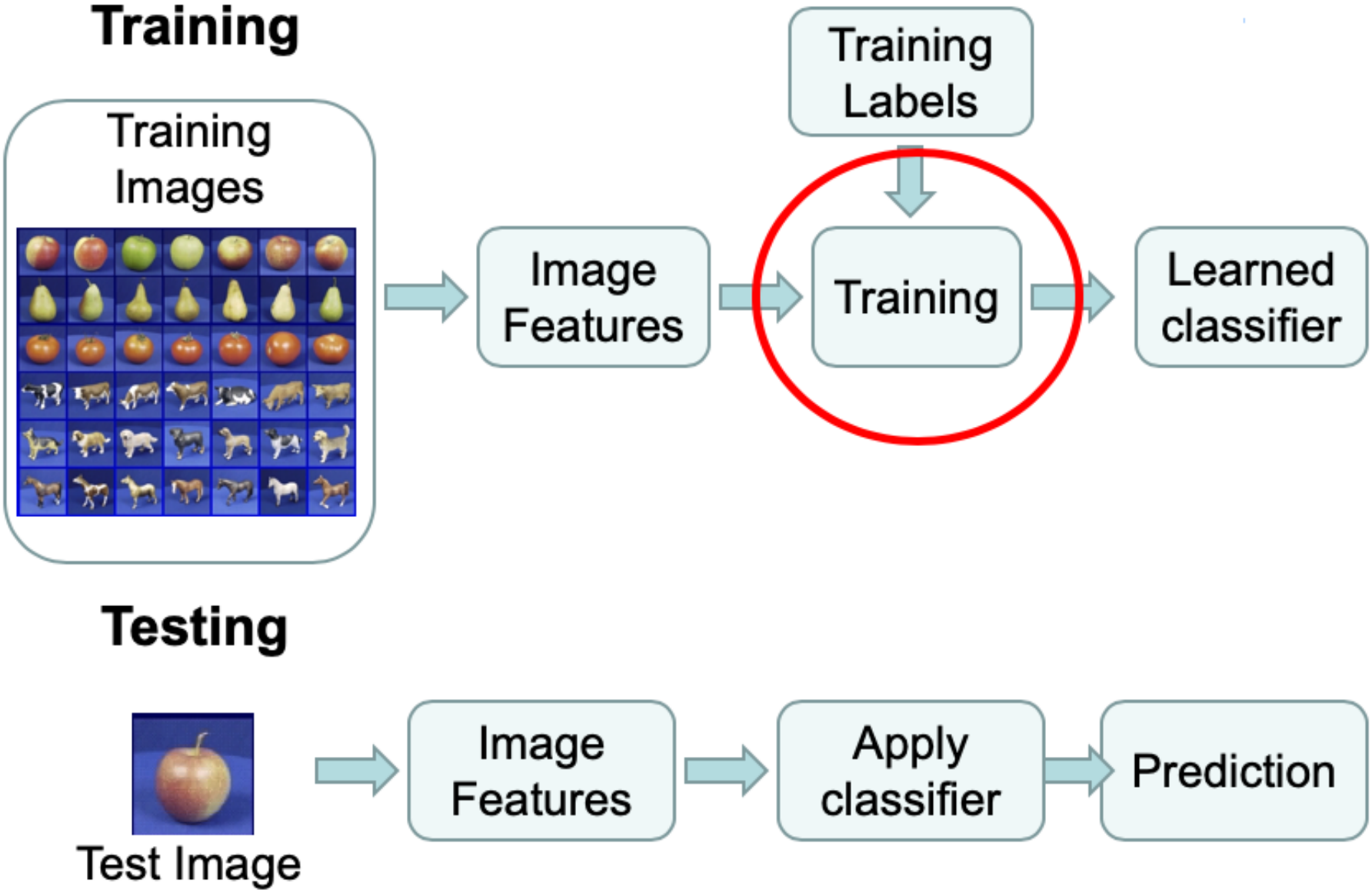
Good labels?

URL

an elephant standing on top of a basket being held by a woman.
a woman standing holding a basket with an elephant in it.
a lady holding an elephant in a small basket.
a lady holds an elephant in a basket.
an elephant inside a basket lifted by a woman.



ML Framework - Training



ML Framework

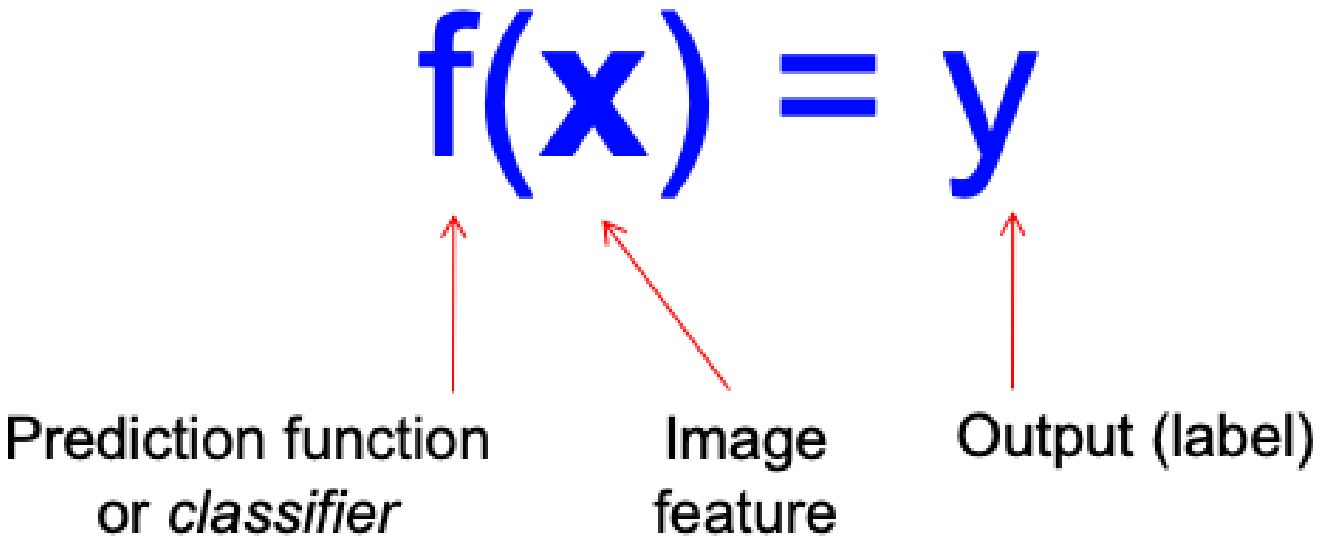
- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

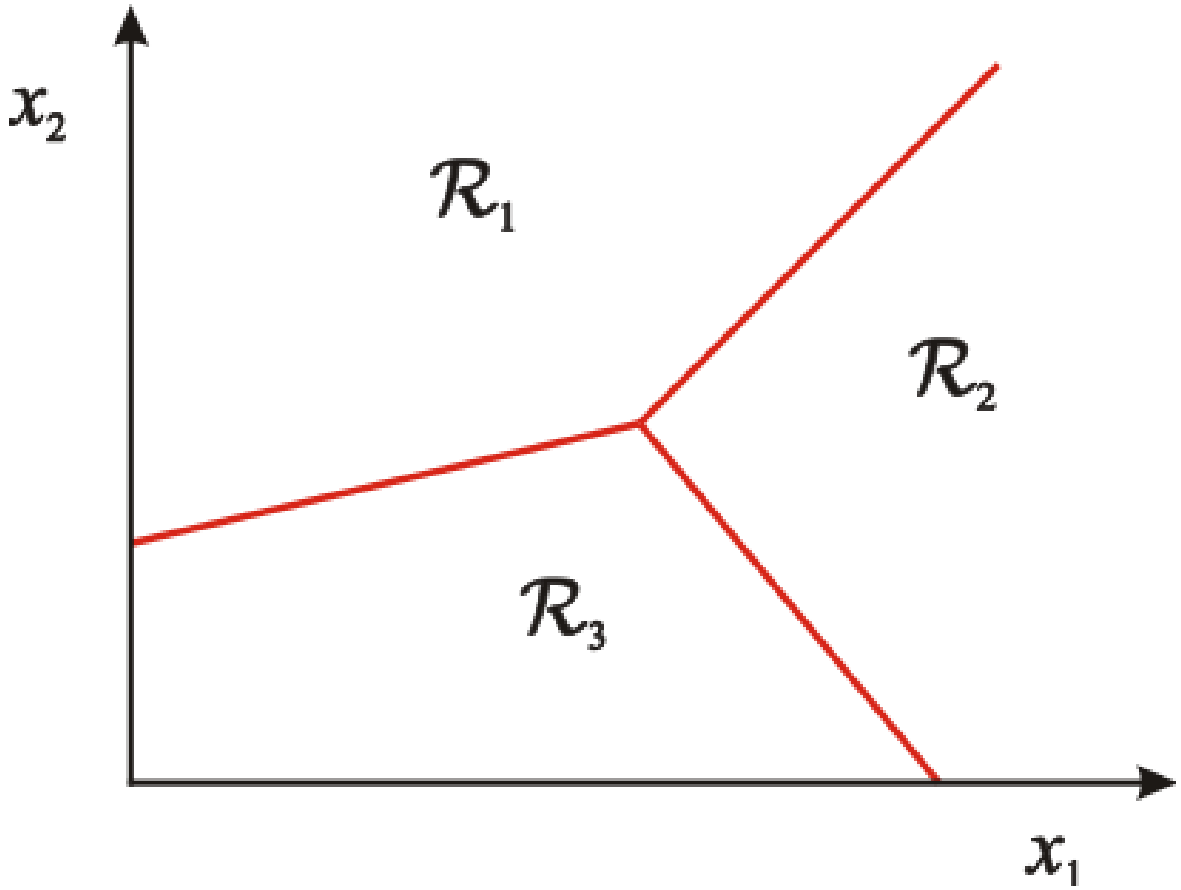
ML Framework



- **Training:** Given a *training set* of labeled samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  Estimate the prediction function $f(\cdot)$ by minimizing the prediction error on the training set.
- **Testing:**. Apply $f(\cdot)$ to some unseen *test sample* x_t and output the prediction $y_t = f(x_t)$ to classify x_t.

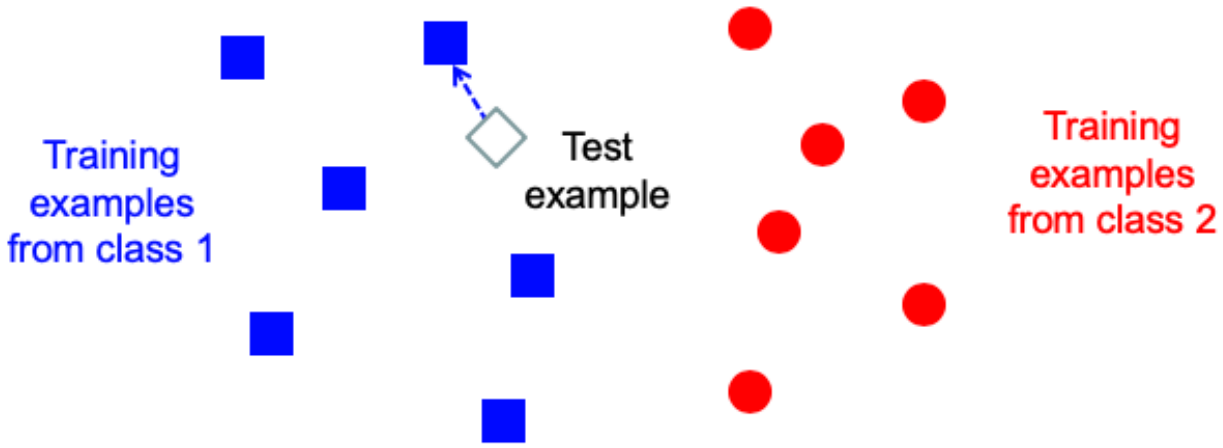
Classification

- Assign x_t to one (or more) classes.
- A decision rule divides input space into decision regions separated by decision boundaries.



Nearest Neighbor Search

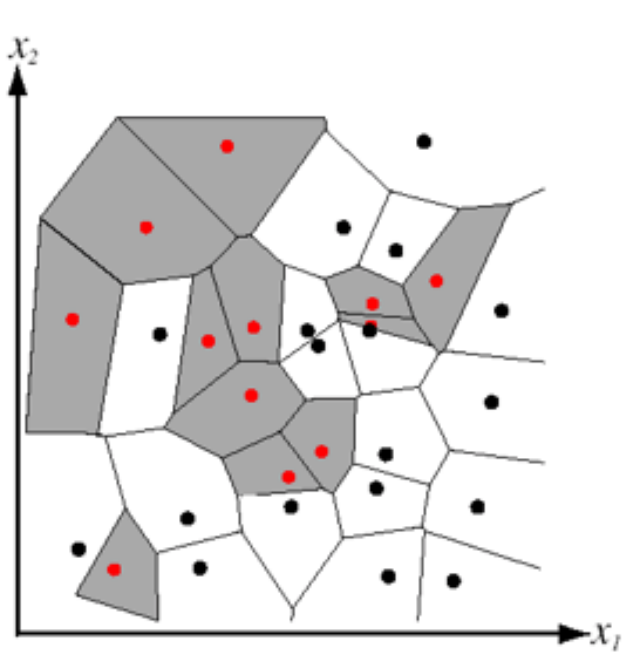
- Can range from simple to complex!
- Classify x_t as $y_t = f(x_t)$ that is "closest to" y_t in the predictor space.



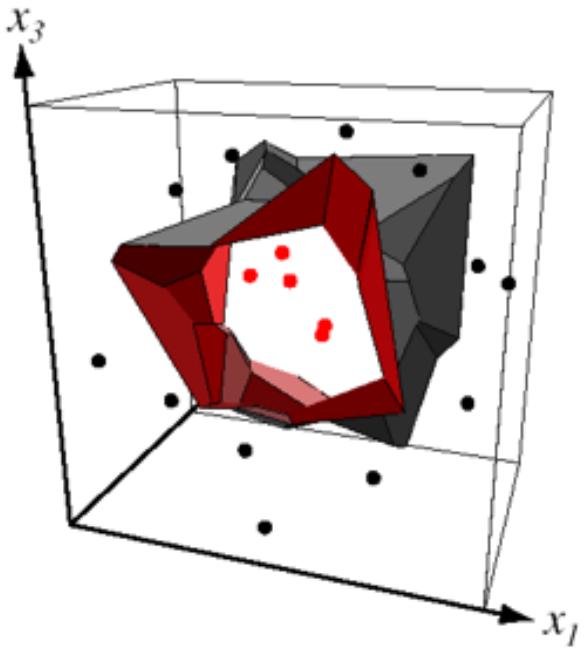
- Can visualize decision boundary via Voronoi diagram

Decision Boundary for NN Classifier

- Divides input space into decision regions separated by decision boundaries – Voronoi



from Duda *et al.*

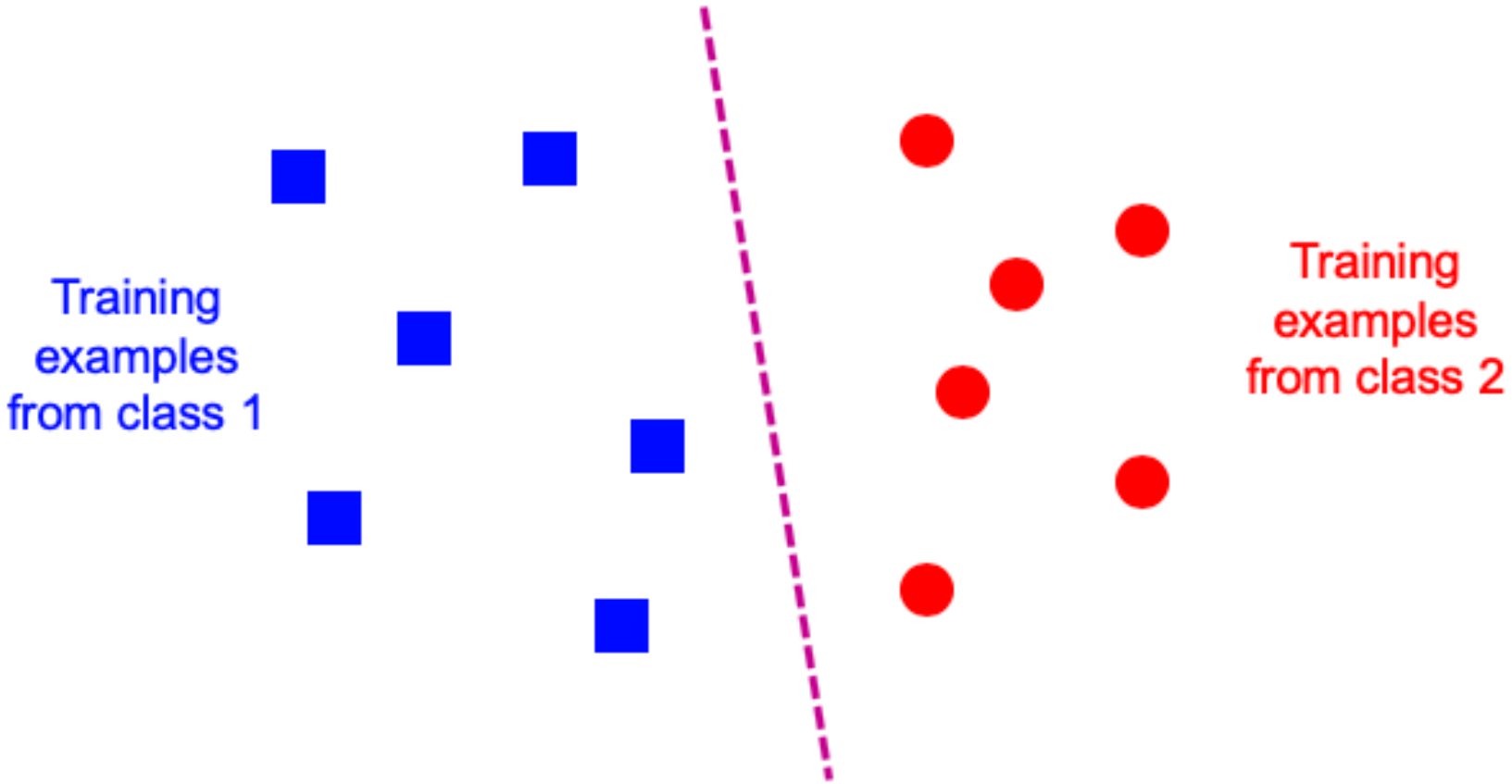


Voronoi partitioning of feature space for two-category 2D and 3D data

Source: D. Lowe

Examples: How to design other "Classifiers"

- Linear:
  - Find a linear function to separate the classes (PCA often works well here)



Examples: How to design other "Classifiers"

- Naive Bayes (more when we discuss Variational Autoencoders):
  - Define the conditional probability model over classes C_k

$$p(C_k|x_1, x_2, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

where Z is the evidence defined by

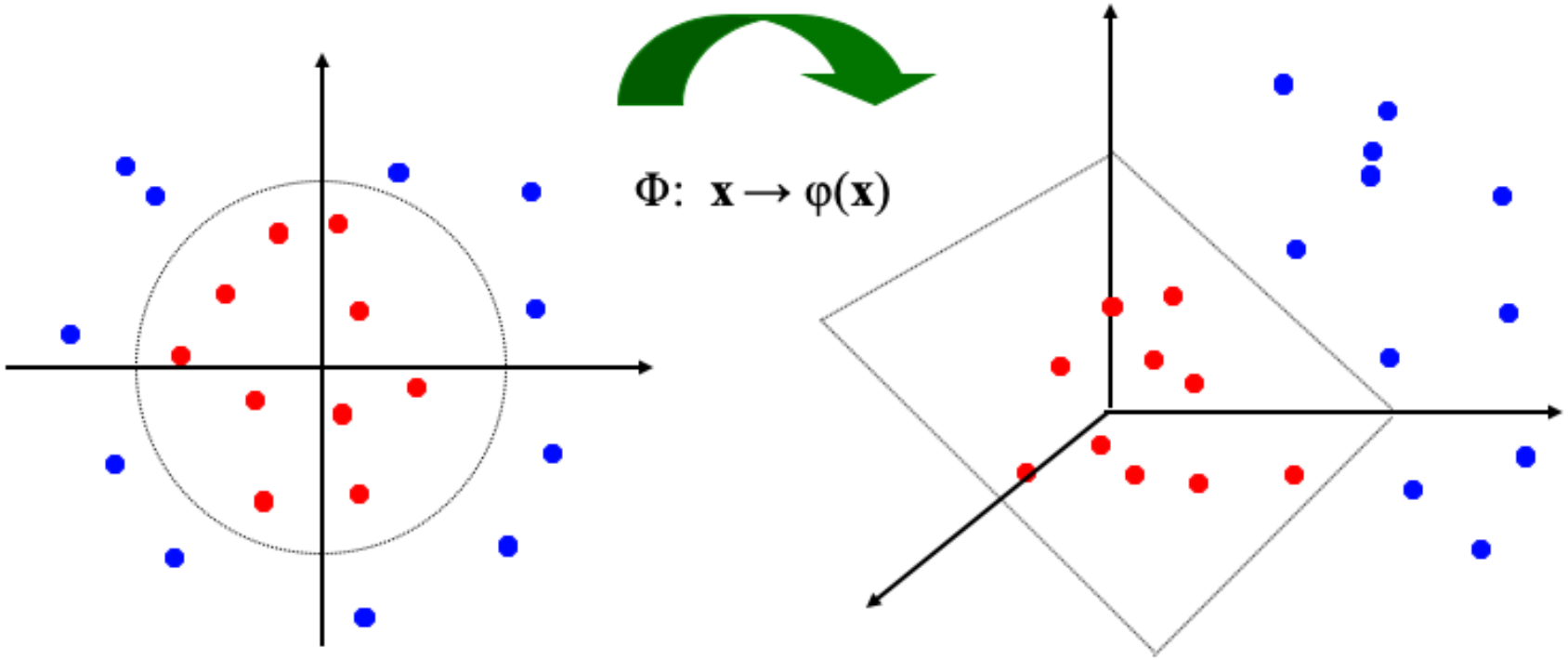
$$Z = p(\mathbf{x}) = \sum_k p(C_k) p(\mathbf{x}|C_k)$$

  - The classifier is then given by:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

Examples: How to design other "Classifiers"

- Nonlinear Support Vector Machine (SVM) (can also use Linear SVM on some data)
  - Map the original input space to some higher-dimensional feature space where the training set is separable



Examples: How to design other "Classifiers"

- Nonlinear Support Vector Machine (SVM) (can also use Linear SVM on some data)
  - So called *kernel trick* - Define a kernel function s.t.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

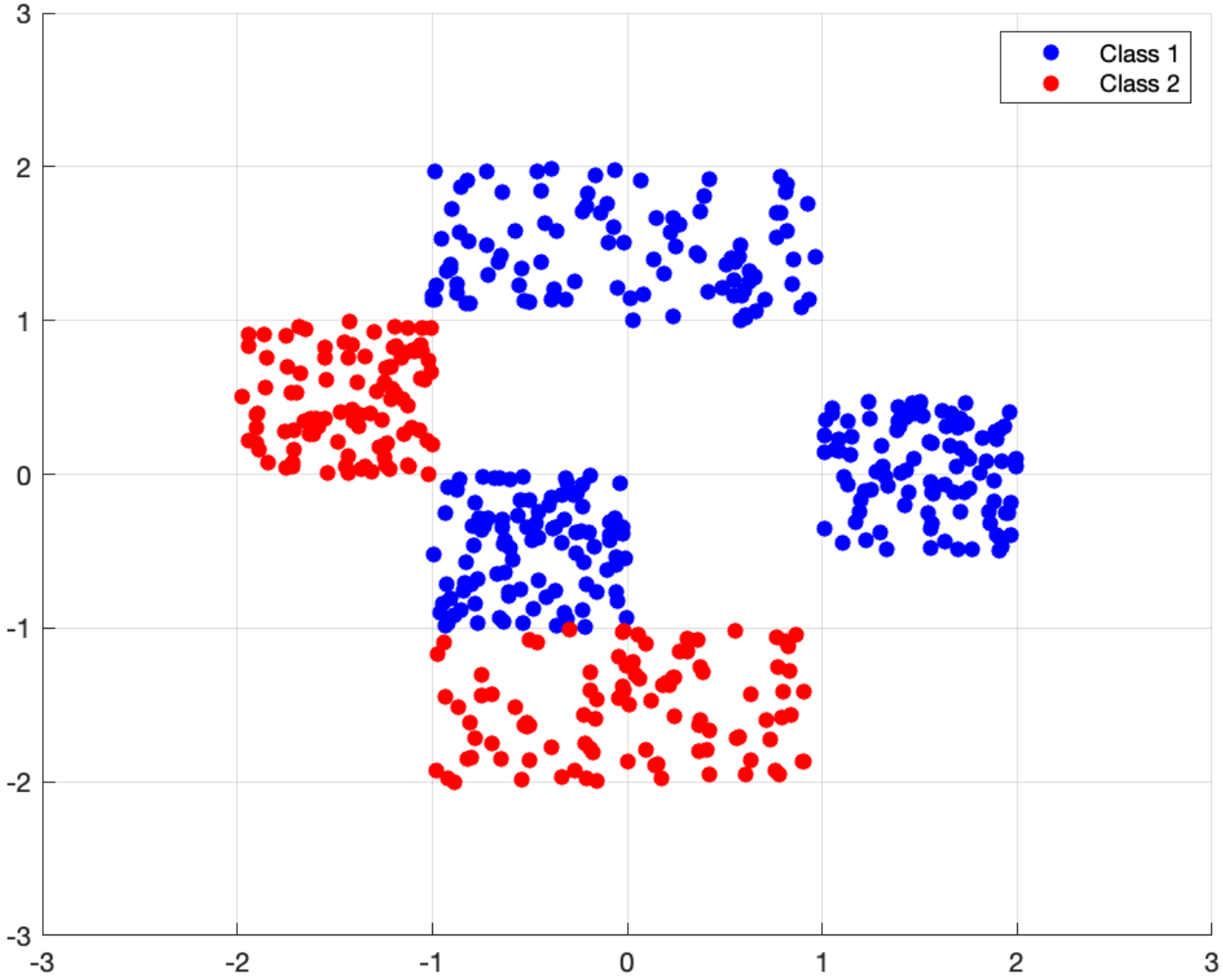
  - Produces a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \phi(x_i) \cdot \phi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(x_i, \mathbf{x}) + b$$

  - Generally use Radial-Basis functions for kernel

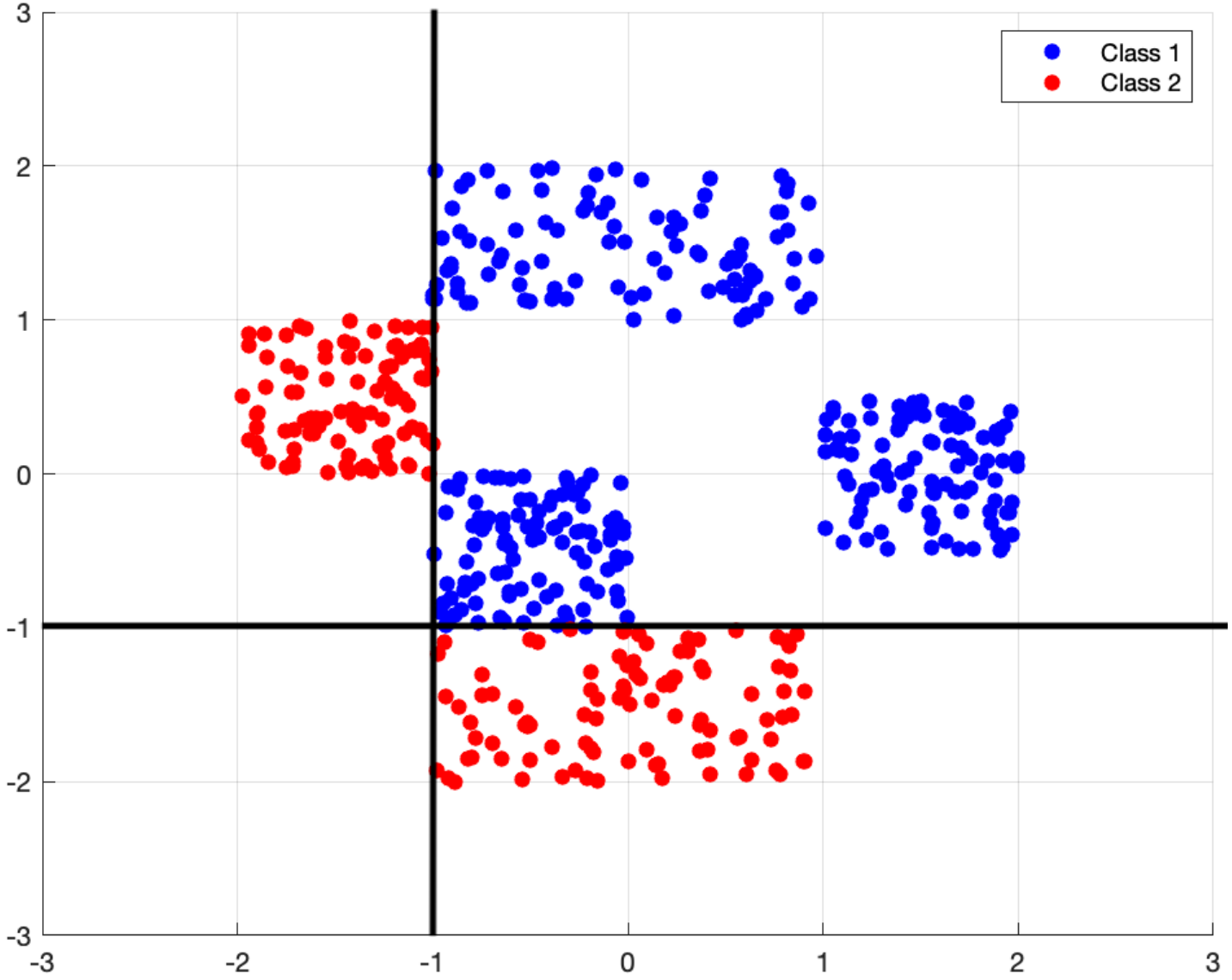
Examples: How to design other "Classifiers"

- Concrete example:



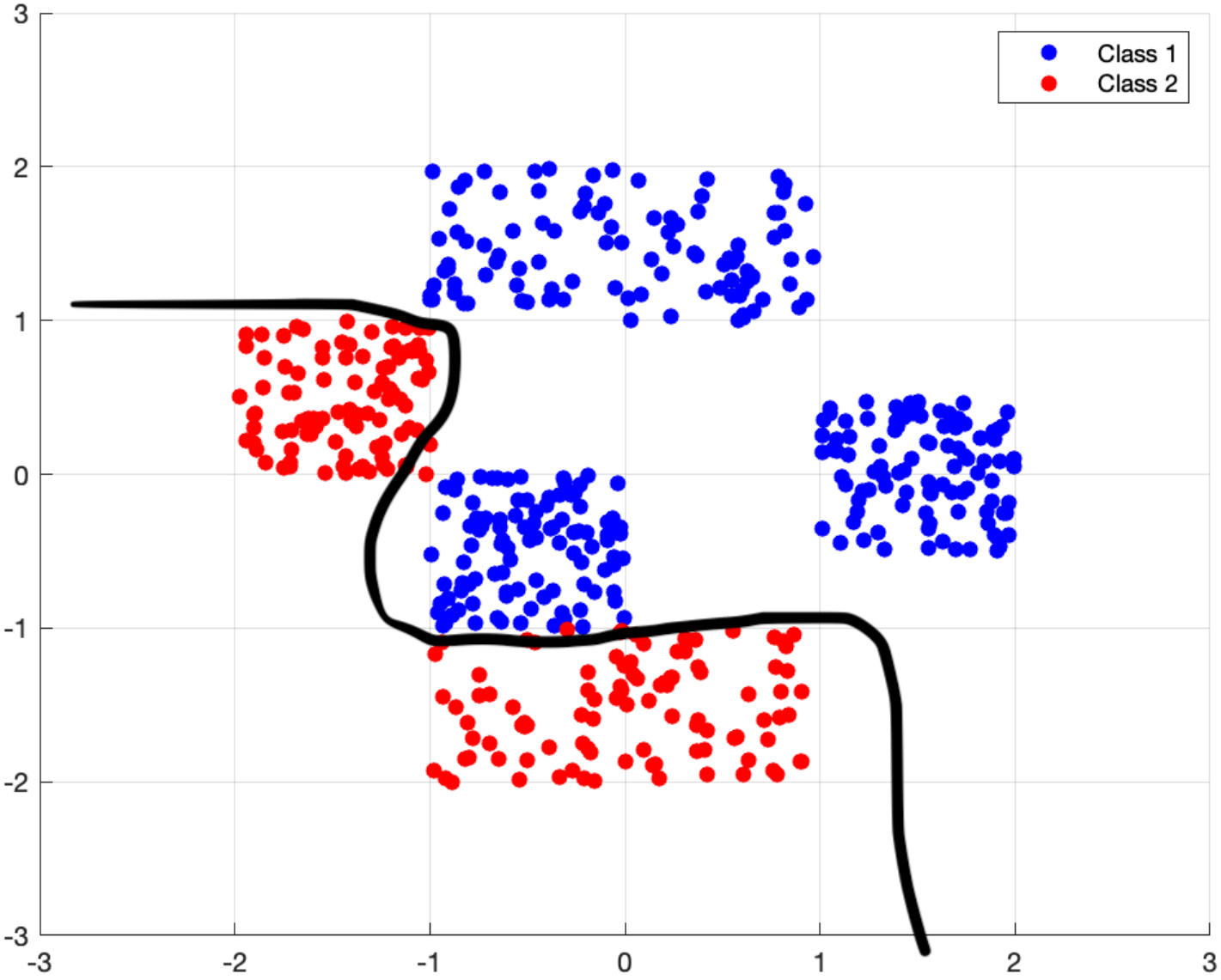
Examples: How to design other "Classifiers"

- Concrete example:



Examples: How to design other "Classifiers"

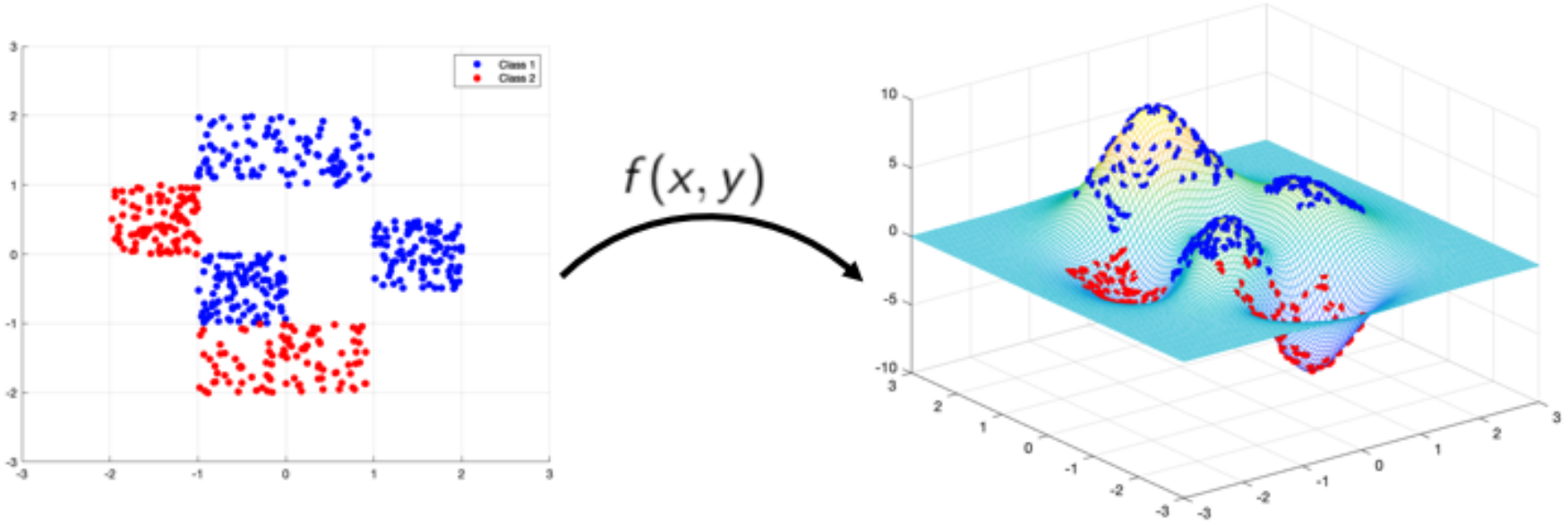
- Concrete example:



Examples: How to design other "Classifiers"

- Suppose we're "allowed" to increase the dimension by 1

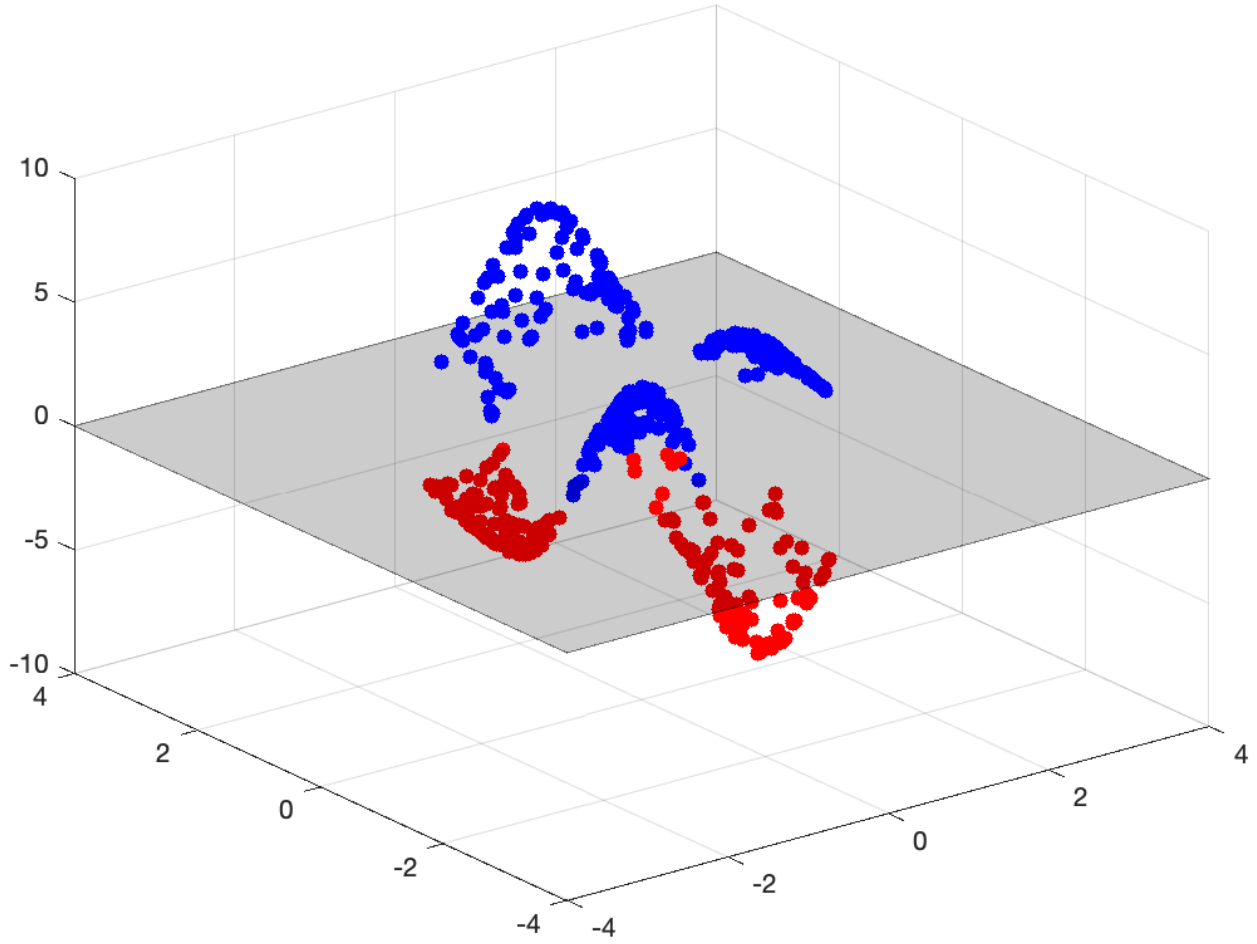
$$f(x,y) = 3(1-x)^2 e^{(-(x^2)-(y+1)^2)} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{(-x^2-y^2)} - \frac{1}{3} e^{(-(x+1)^2-y^2)}$$



Examples: How to design other "Classifiers"

- Suppose we're "allowed" to increase the dimension by 1

$$f(x,y) = 3(1-x)^2 e^{(-(x^2)-(y+1)^2)} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{(-x^2-y^2)} - \frac{1}{3} e^{(-(x+1)^2-y^2)}$$



Multi-Class SVMs?

- As you might imagine, as the number of classes increases so does the complexity of "finding" a nonlinear mapping to do what we want:
  - We can " sort of " claim this is what neural networks attempt to accomplish
- Another approach is attempt to combine multiple two-class SVMs
  1. One vs. the other classes:
     - Training: learn an SVM for each class vs. the others
     - Testing: apply each SVM to test the input sample and assign to it the class of the SVM that returns the highest decision value
  2. One class vs. one class:
     - Training: learn an SVM for each pair of classes
     - Testing: each learned SVM "votes" for a class to assign to the test example

Classifiers in general (feature-based)

- **Features** and **distance measures**
  - define visual similarity
- **Training labels**
  - dictate that examples are the same or different
- **Classifiers**
  - learn weights (or parameters) of features and distance measures so that visual similarity predicts label similarity

Generalization

Training set (labels known)

Test set (labels unknown)

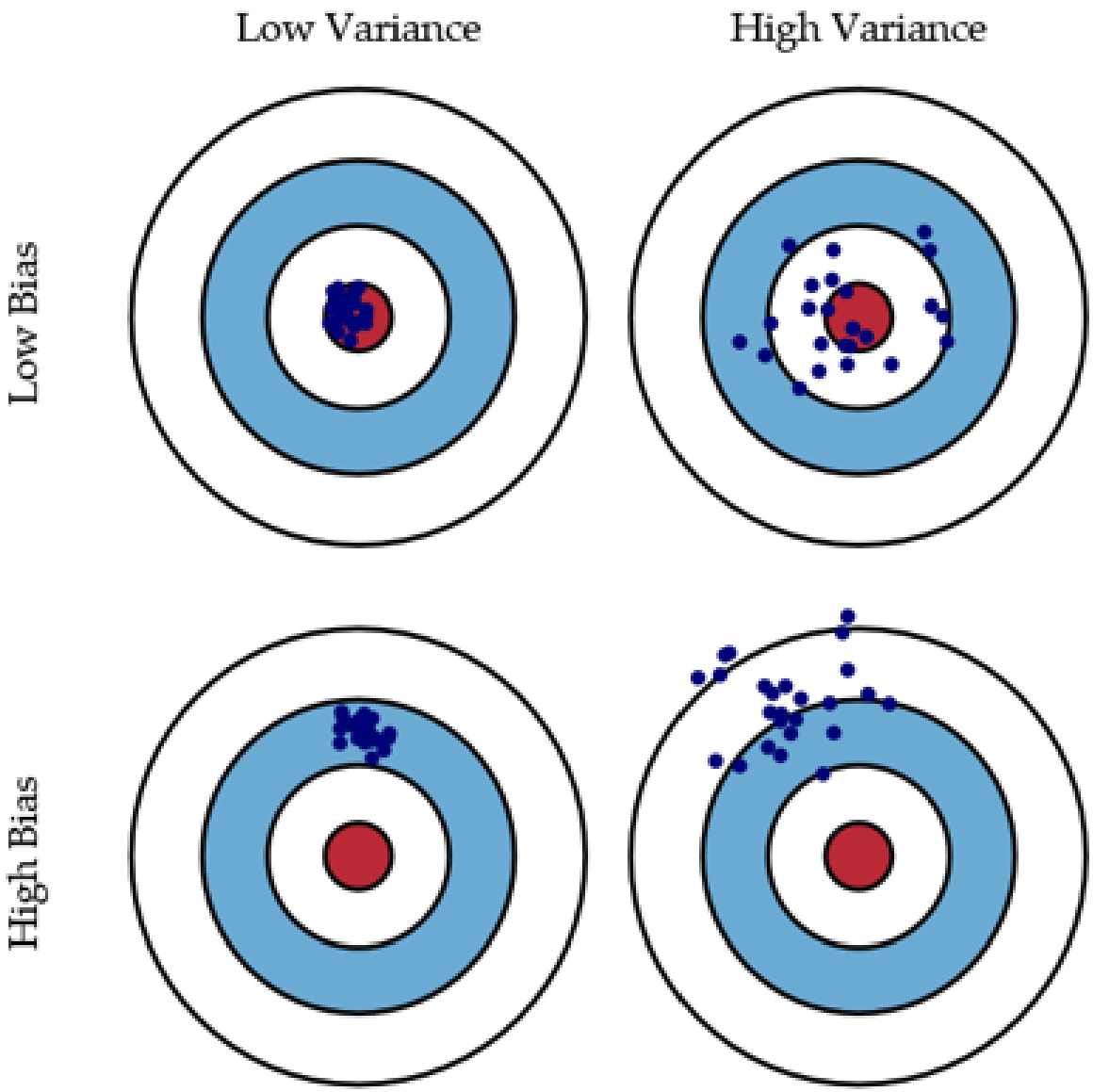
- How well does a learned model generalize from the data it was trained on to a new test set?

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

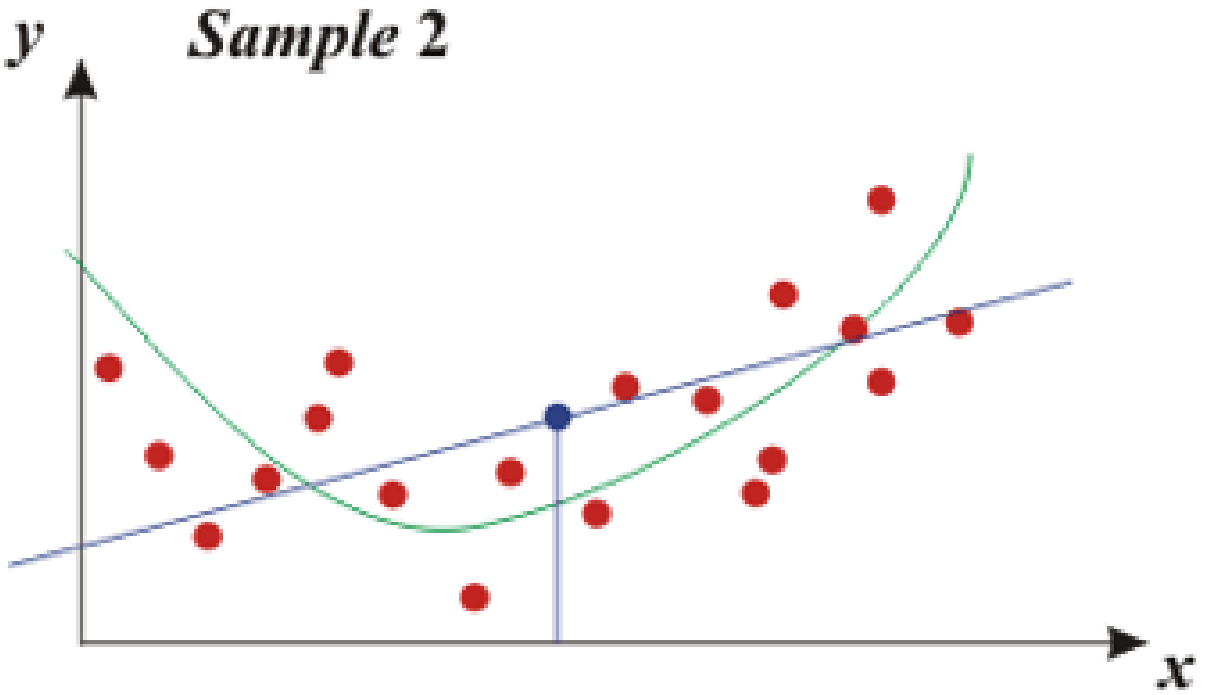


Bias = accuracy

Variance = precision

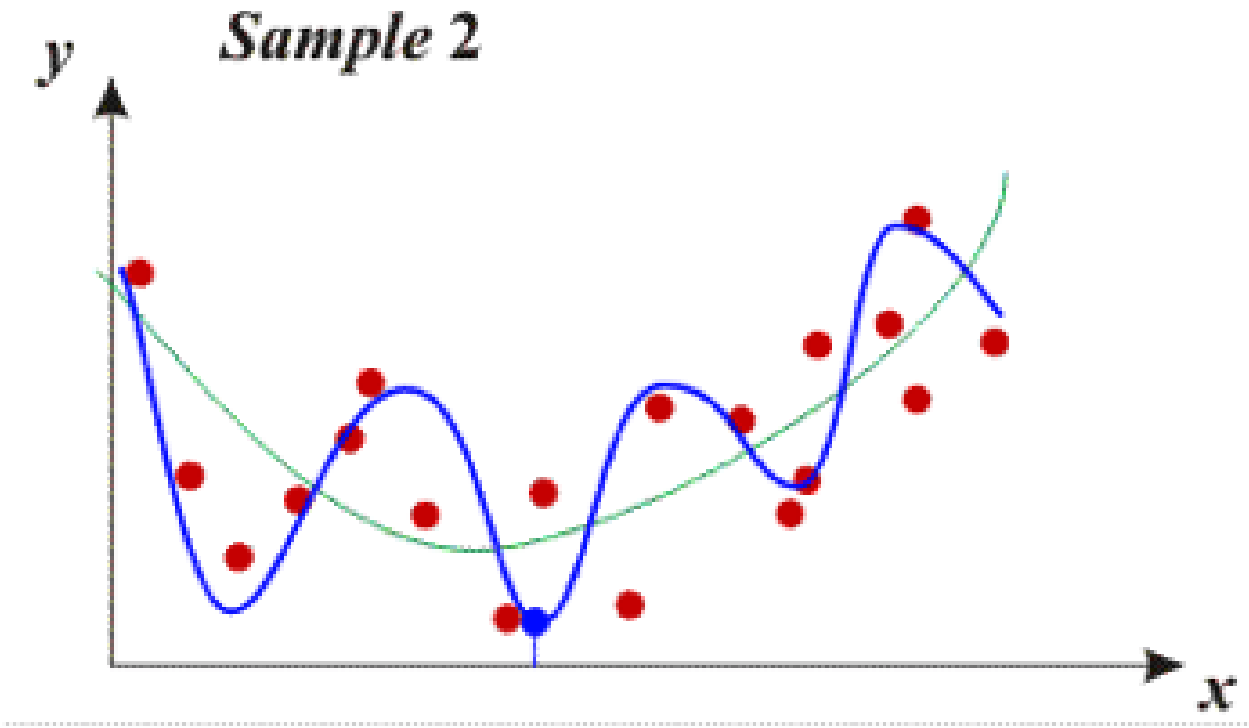
Generalization: Error effects

- **Underfitting:** model is too "simple" to represent all the relevant class characteristics
  - High bias (few degrees of freedom) and low variance
  - High training error and high test error

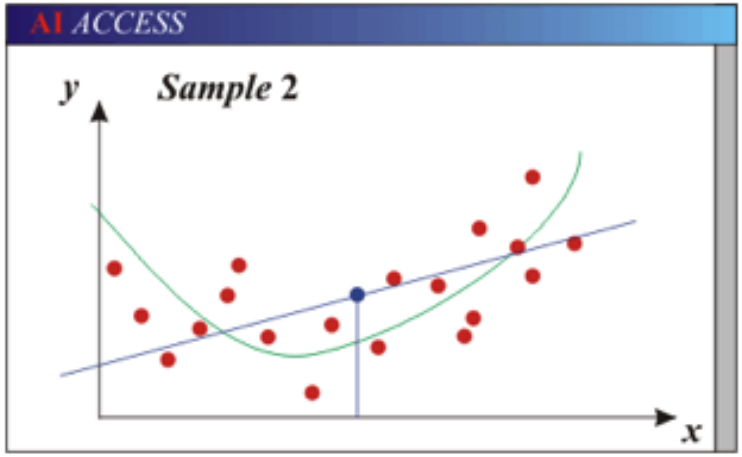


Generalization: Error effects

- **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

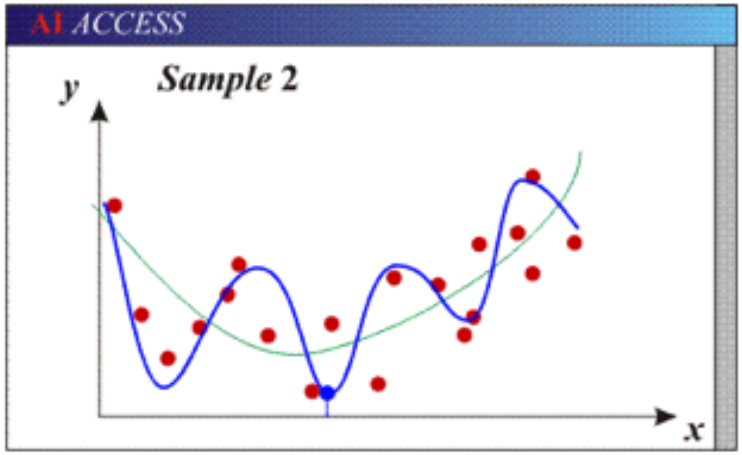


Generalization: Error effects



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

- Not enough flexibility!
- Too many assumptions



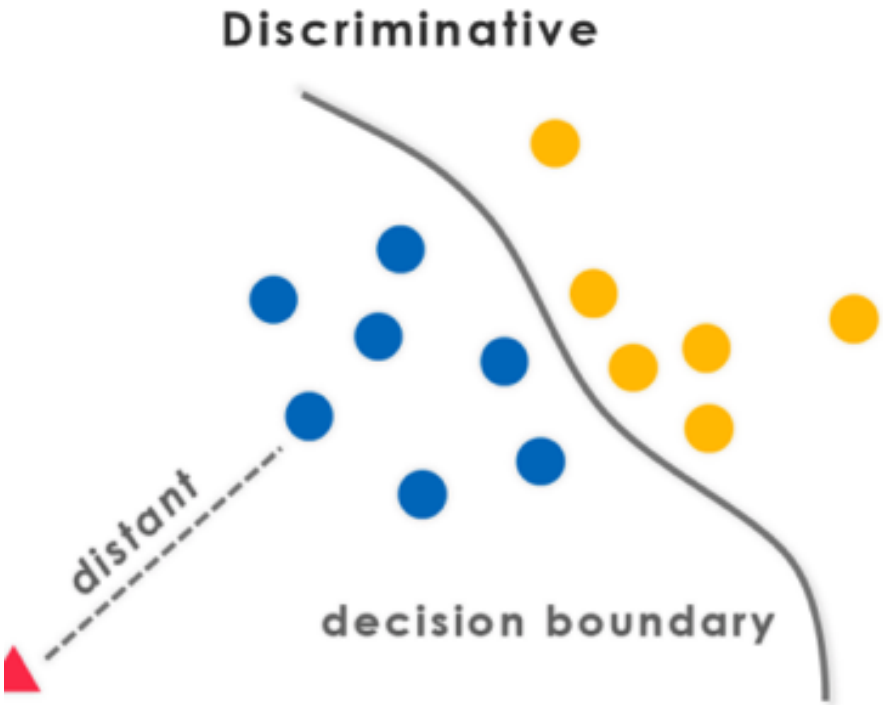
Models with too many parameters are inaccurate because of a large variance.

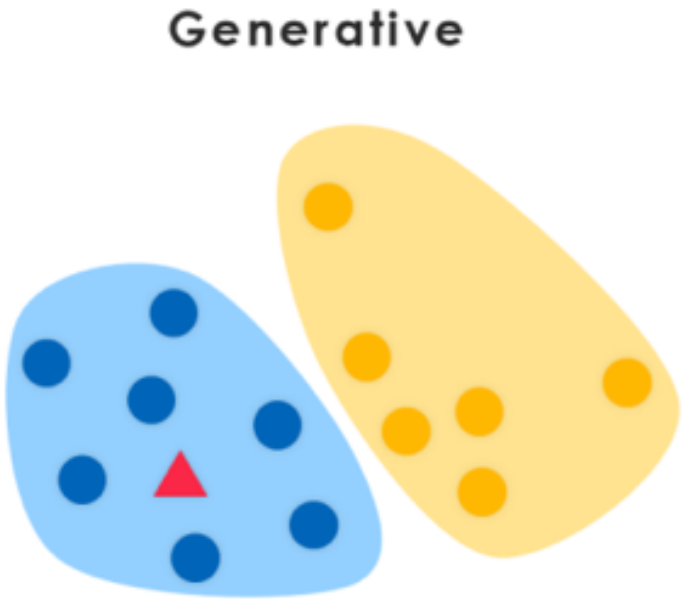
- Too much sensitivity to the sample.
- Slightly different data -> very different function.

Classifiers: Generative vs. Discriminative

- Discriminative Models
  - Learn to directly predict the labels from the data
  - Often, assume a simple boundary (e.g., linear)
  - Examples:
    - Logistic regression
    - SVM
    - Boosted decision trees
  - Often easier to predict a label from the data than to model the data
- Generative Models
  - Represent both the data and the labels
  - Often, makes use of conditional independence and priors
  - Examples:
    - Naive Bayes classifier
    - Bayesian network
  - Models of data may apply to future prediction problems

Classifiers: Generative vs. Discriminative





"Learn the data boundary"

"Represent the data + boundary"

Bayesian methods:
Condition model on
data probabilistically

Classifiers

- We will really " dig in " to linear classifiers over the next couple weeks (PCA & LDA)
- What to keep in mind:
  - No free lunch: machine learning algorithms are tools, not dogmas
  - 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)