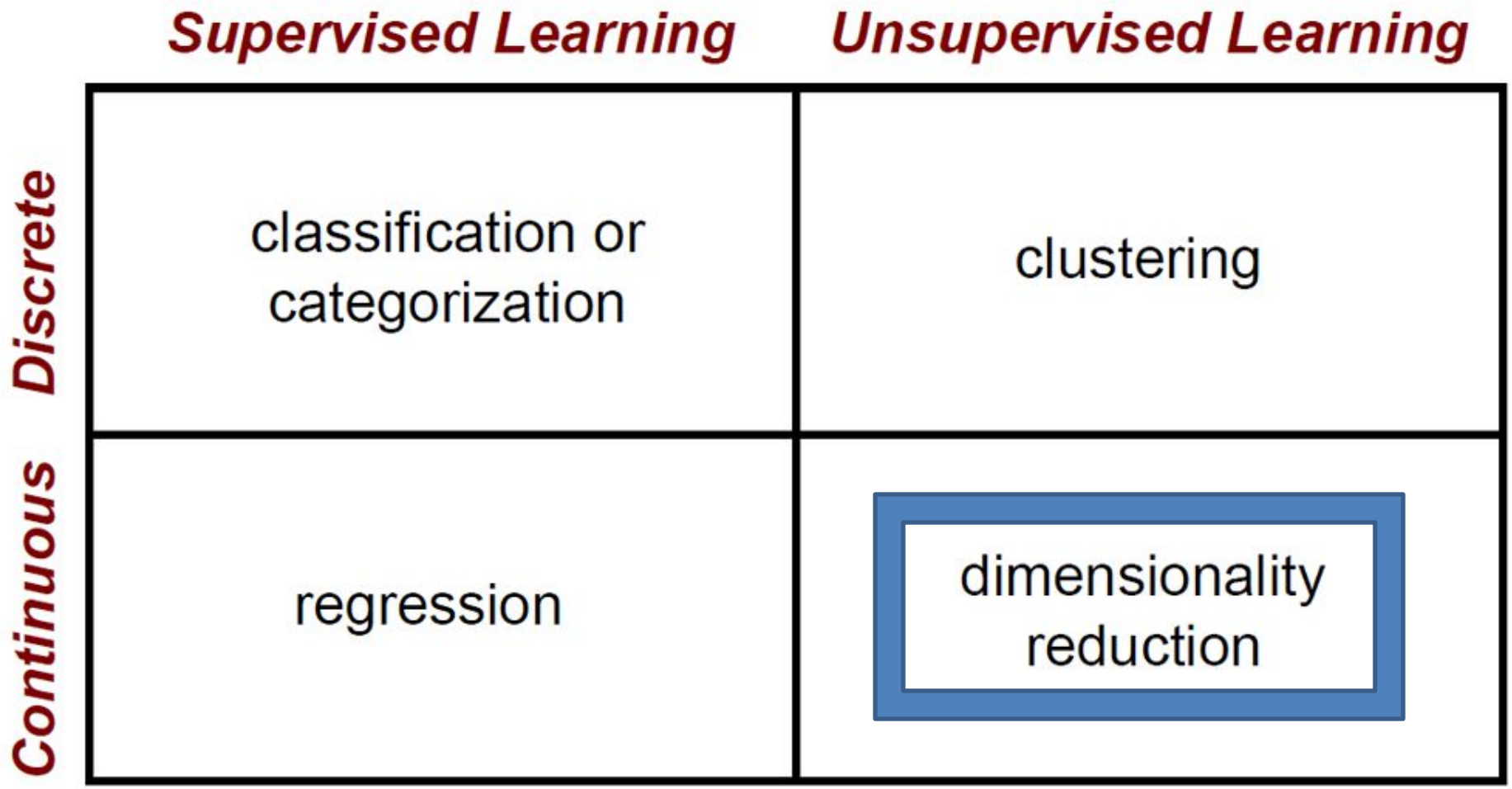
# Machine Learning Basics


Photo: CMU Machine Learning Department protests G20

# Machine Learning Problems

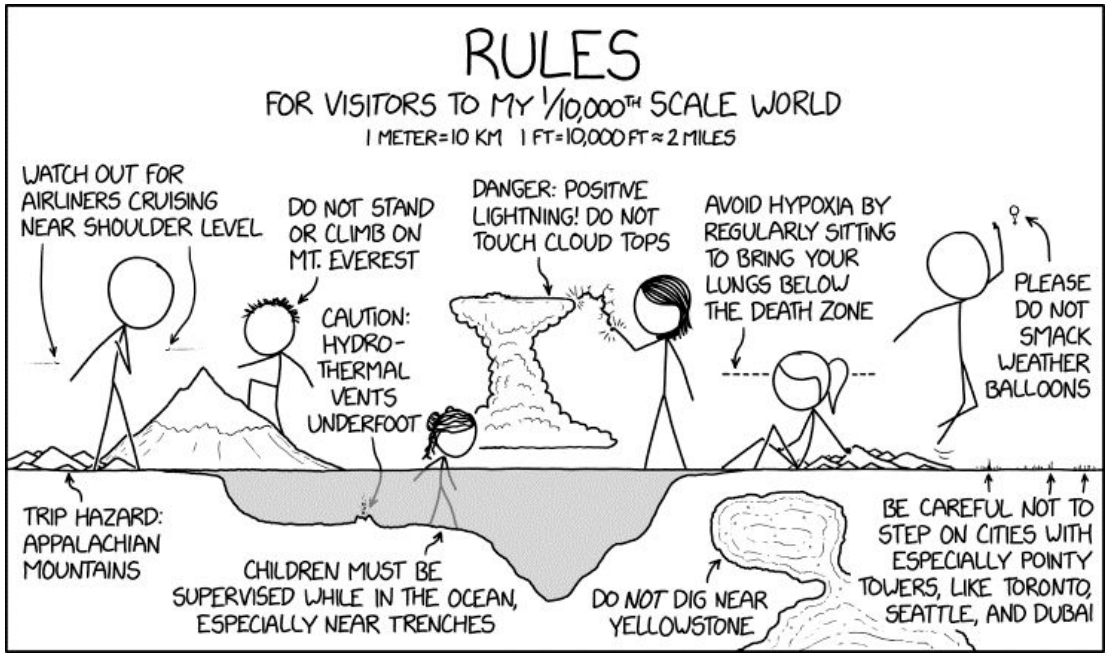


# Dimensionality Reduction


Simplest dimensionality reduction: drop a dimension

Credit: xkcd

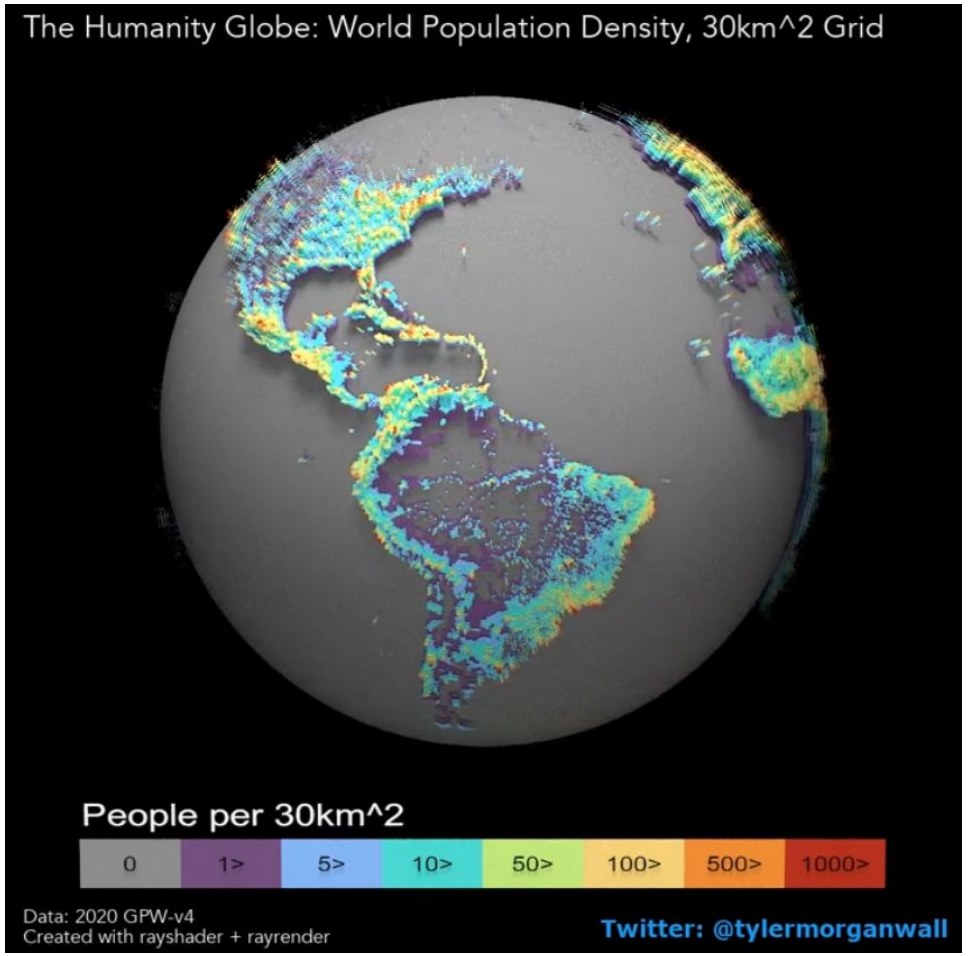
# Dimensionality Reduction


The Earth is pretty smooth

Credit: xkcd

# Non-linear Dimensionality Reduction

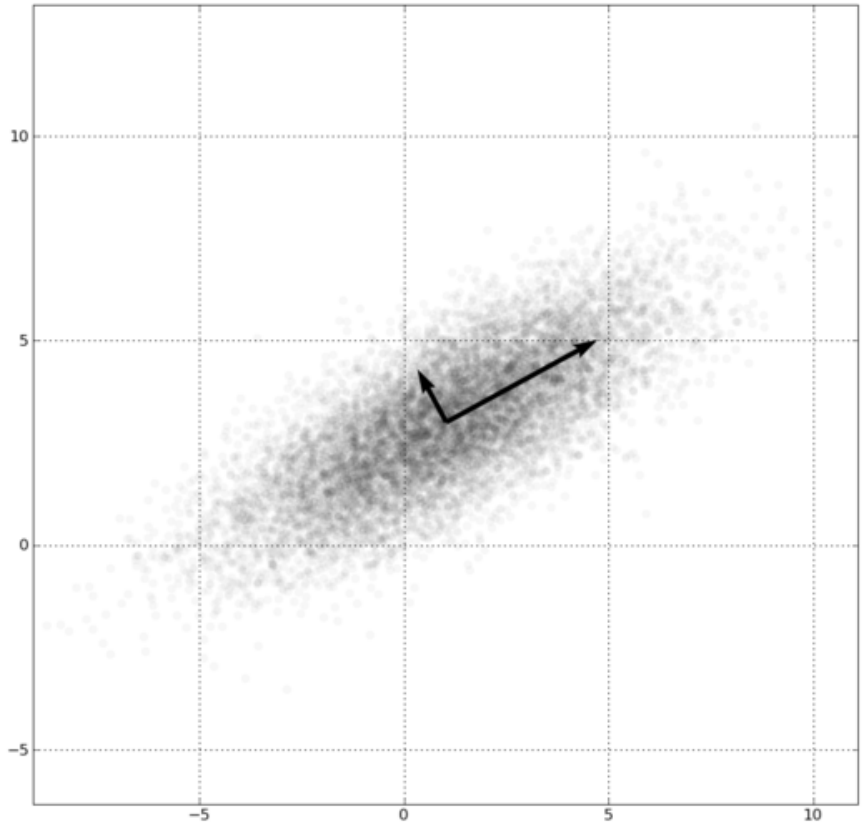

Credit: xkcd

# Dimensionality Reduction

- **PCA**, ICA, LLE, Isomap, *Autoencoder*

- PCA is the most important technique to know.  It takes advantage of correlations in data dimensions to produce the best possible lower dimensional representation based on linear projections (minimizes reconstruction error).

- Be wary of trying to assign meaning to the discovered bases.



# Eigenfaces for Recognition

**Matthew Turk and Alex Pentland**
Vision and Modeling Group
The Media Laboratory
Massachusetts Institute of Technology

Training data
16 256x256 images

**Figure 1. (b)** The average face $\Psi$.

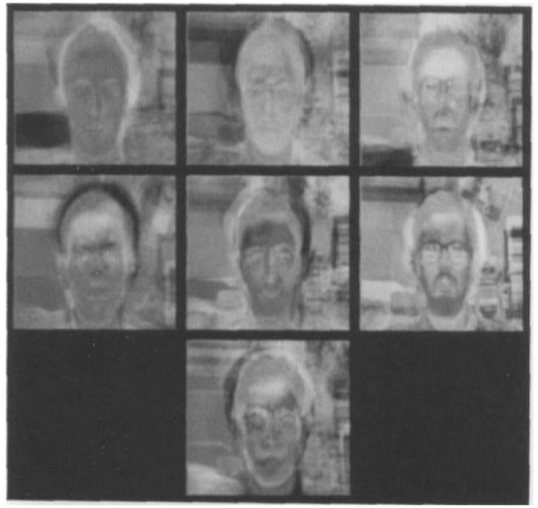
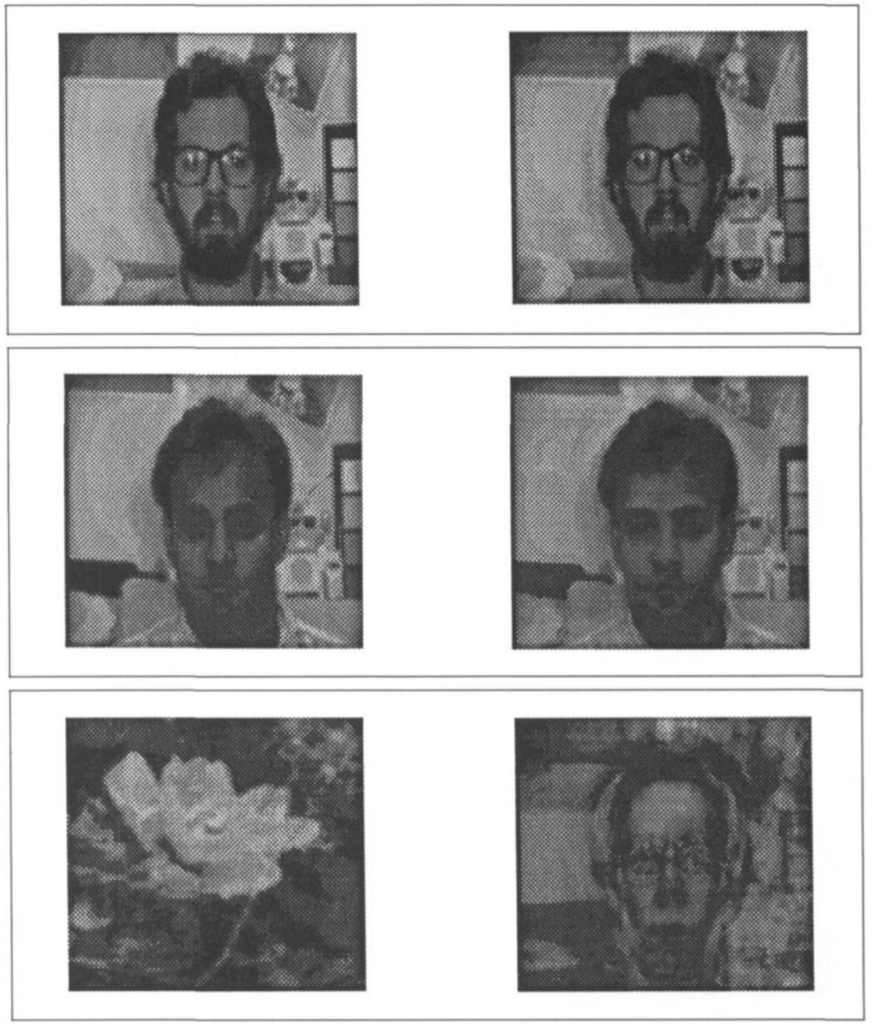


**Figure 2.** Seven of the eigenfaces calculated from the input images of Figure 1.

The "Eigenfaces"



Reconstruction of
in-domain and
out-of-domain images

# PCA as a data interpretation tool

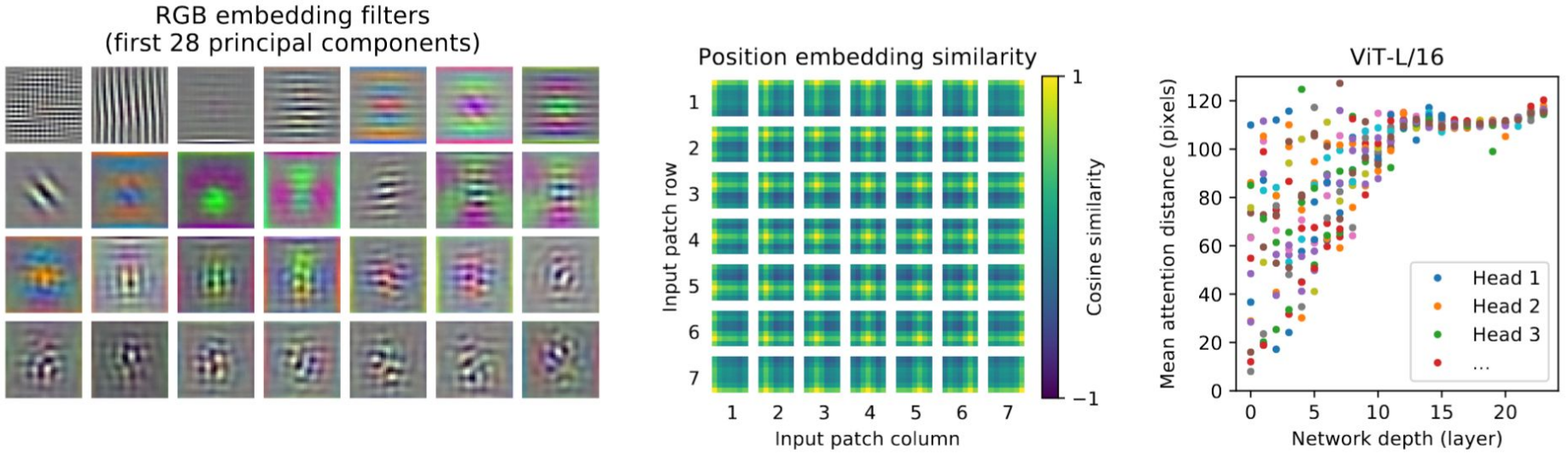
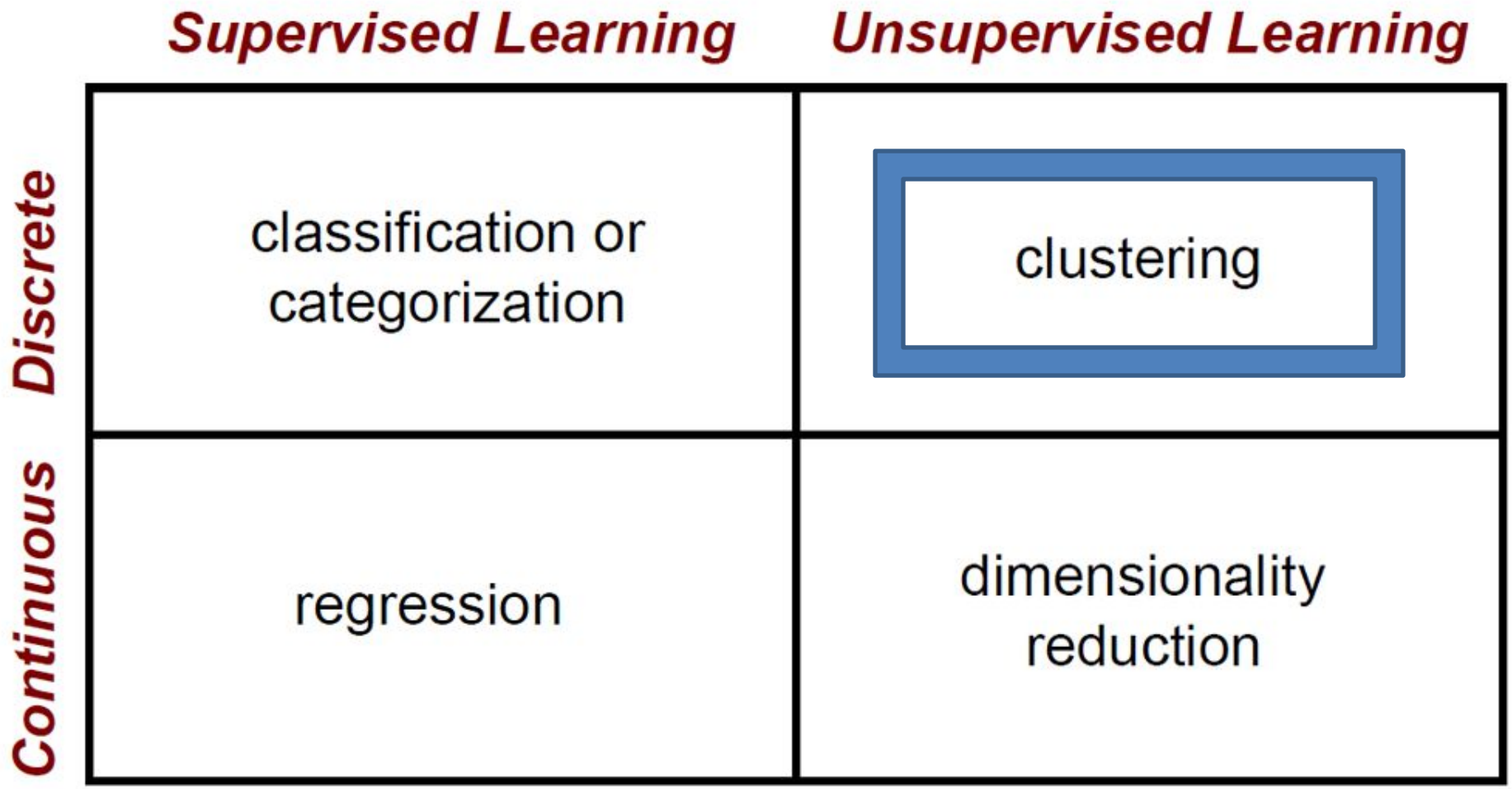
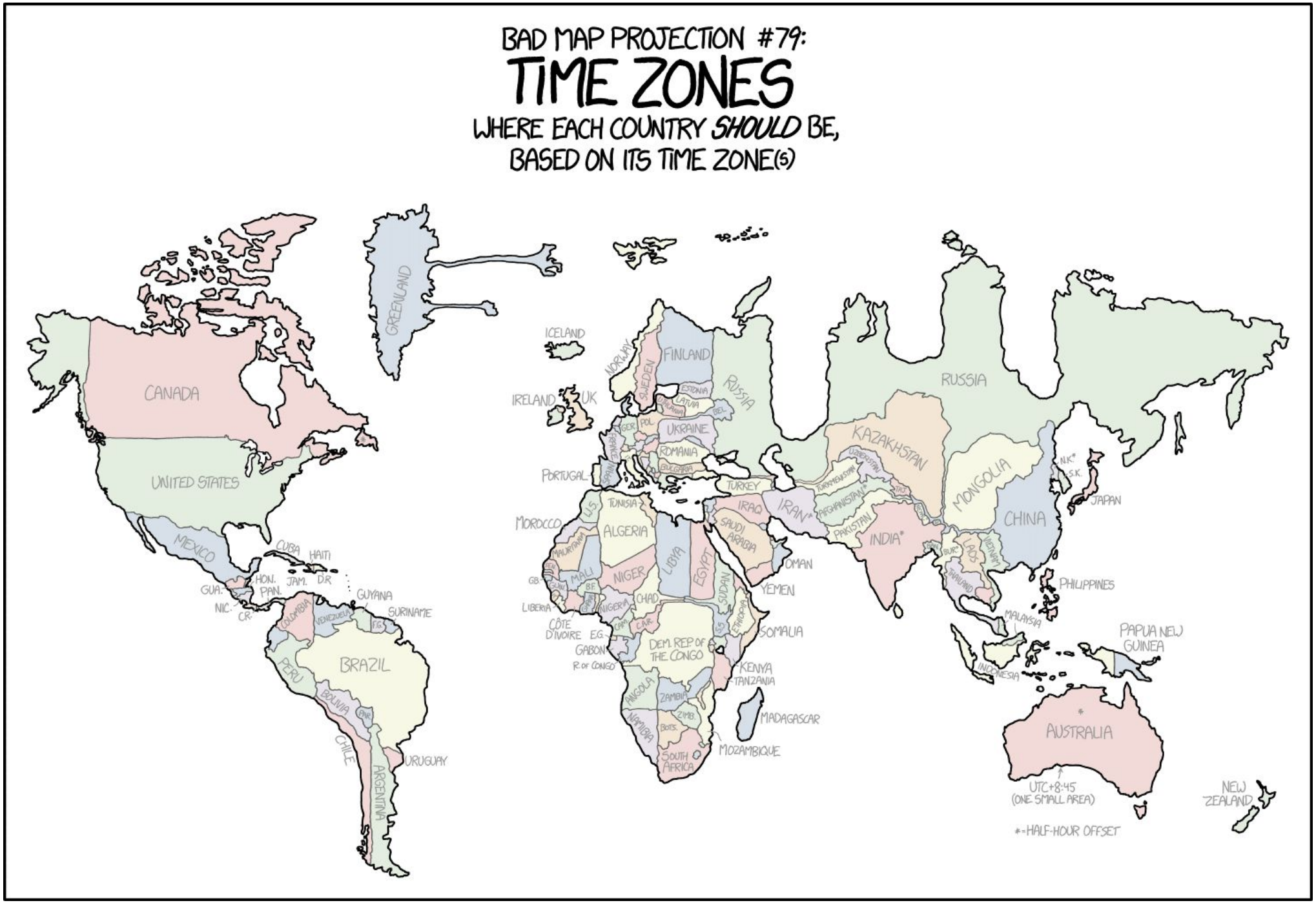



Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **Right:** Size of attended area by head and network depth. Each dot shows the mean attention distance across images for one of 16 heads at one layer. See Appendix D.6 for details.

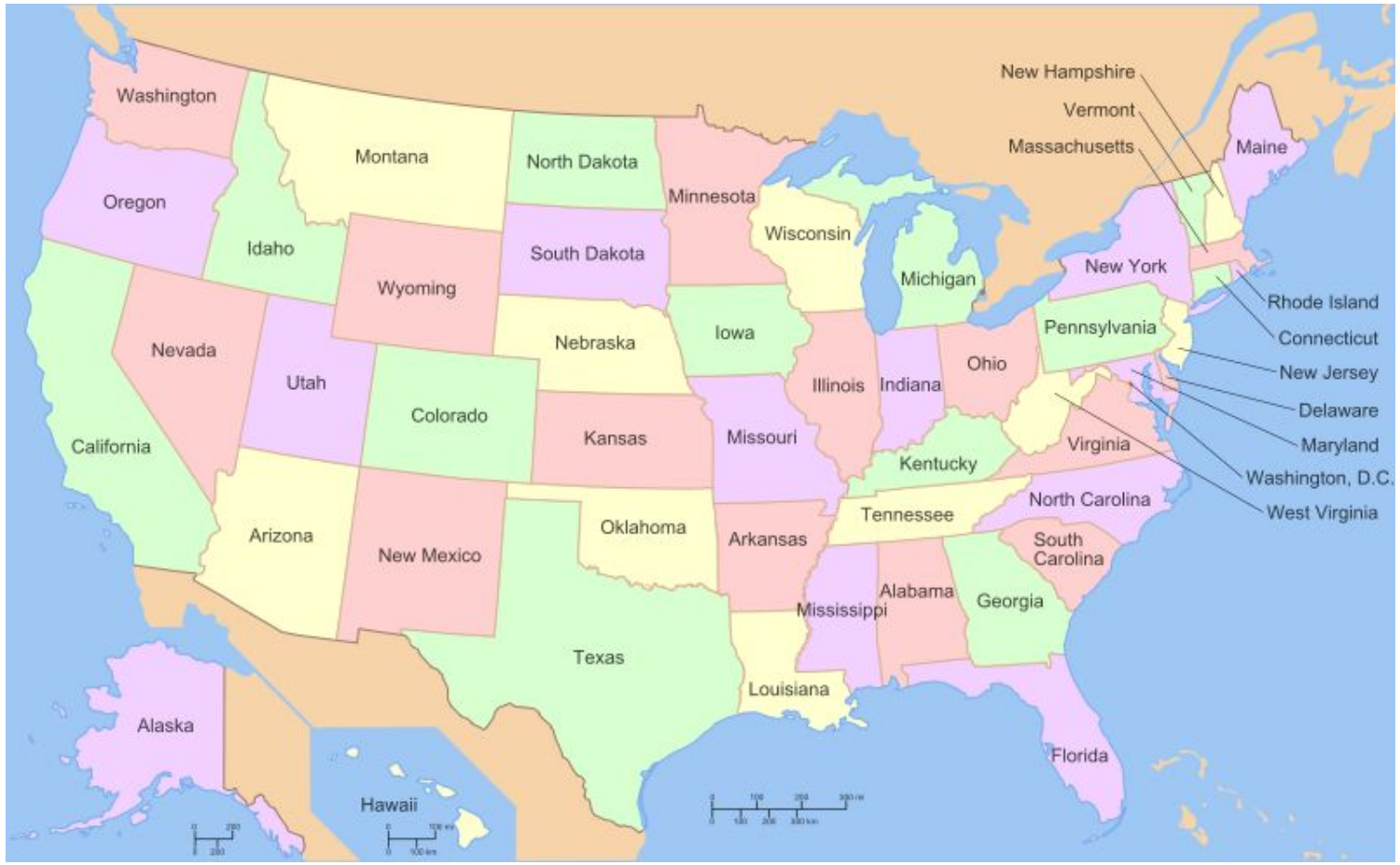
An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. ICLR 2021

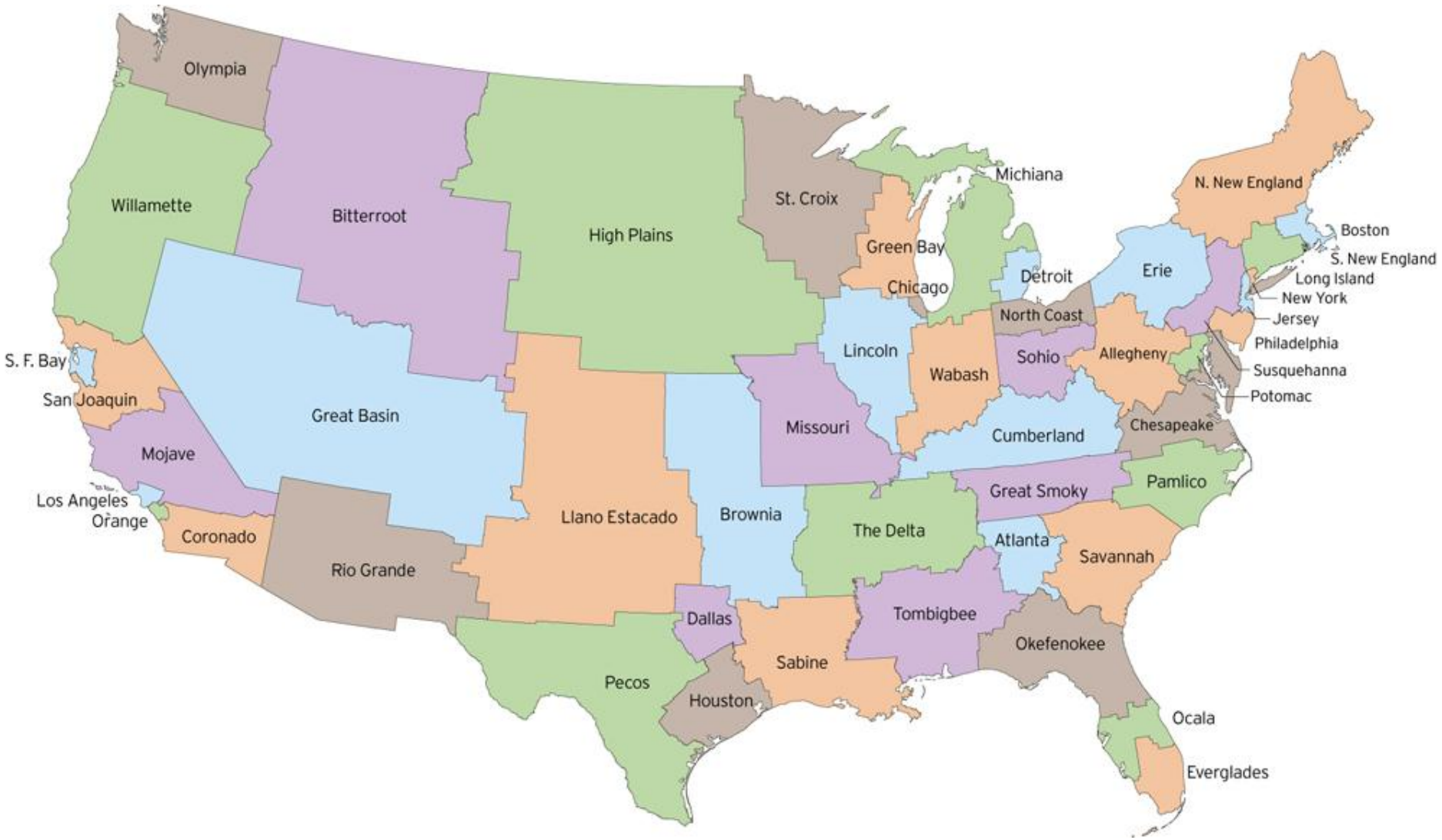
# Machine Learning Problems



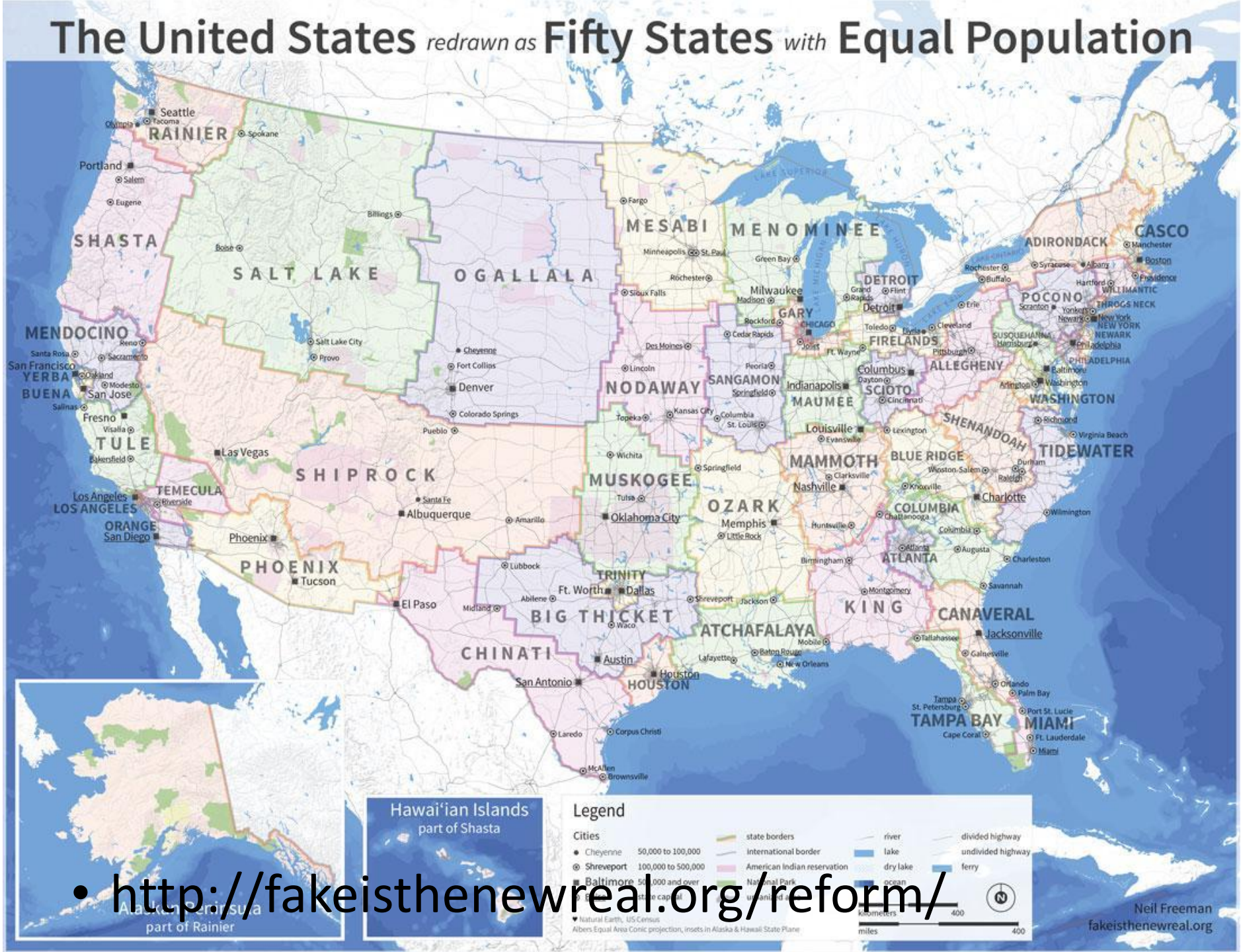

BAD MAP PROJECTION #79:
TIME ZONES
WHERE EACH COUNTRY SHOULD BE,
BASED ON ITS TIME ZONE(S)
GREENLAND
ICELAND
CANADA
UNITED STATES
MEXICO
CUBA
HAITI
HON.
JAM.
DR
PAN.
GUA.
NIC.
CR.
GUYANA
SURINAME
COLOMBIA
VENEZUELA
EC.
PERU
BRAZIL
BOLIVIA
PAR.
CHILE
ARGENTINA
URUGUAY
IRELAND
UK
NORWAY
SWEDEN
FINLAND
ESTONIA
LATVIA
BEL.
POL.
GER.
NETH.
UKRAINE
ROMANIA
BULGARIA
PORTUGAL
SPAIN
TURKEY
TUNISIA
U.S.
MOROCCO
ALGERIA
MAURITANIA
MALI
NIGER
LIBYA
EGYPT
SAUDI ARABIA
IRAQ
IRAN*
AFGHANISTAN*
PAKISTAN
INDIA*
RUSSIA
KAZAKHSTAN
UZBEKISTAN
TURKMENISTAN*
MONGOLIA
CHINA
N.K.*
S.K.*
JAPAN
OMAN
YEMEN
GB.
GUINEA
SENEGAL
BF.
NIGERIA
CHAD
SUDAN
ETHIOPIA
SOMALIA
LIBERIA
CÔTE D'IVOIRE
EG.
GABON
R. OF CONGO
CAMR.
CAR
DEM. REP OF THE CONGO
KENYA
TANZANIA
ANGOLA
ZAMBIA
ZIMB.
BOTS.
NAMIBIA
SOUTH AFRICA
MADAGASCAR
MOZAMBIQUE
BURMA
LAOS
VIETNAM
THAILAND
PHILIPPINES
MALAYSIA
INDONESIA
PAPUA NEW GUINEA
AUSTRALIA
UTC+8:45
(ONE SMALL AREA)
NEW ZEALAND
*=HALF-HOUR OFFSET

Credit: xkcd


Washington
Oregon
California
Nevada
Idaho
Montana
Wyoming
Utah
Arizona
Colorado
New Mexico
North Dakota
South Dakota
Nebraska
Kansas
Oklahoma
Texas
Minnesota
Iowa
Missouri
Arkansas
Louisiana
Wisconsin
Illinois
Michigan
Indiana
Ohio
Kentucky
Tennessee
Mississippi
Alabama
Georgia
Florida
South Carolina
North Carolina
Virginia
West Virginia
Pennsylvania
New York
Maryland
Delaware
New Jersey
Connecticut
Rhode Island
Massachusetts
Vermont
New Hampshire
Maine
Washington, D.C.
Alaska
Hawaii




- http://fakeisthenewreal.org/reform/




- http://fakeisthenewreal.org/reform/

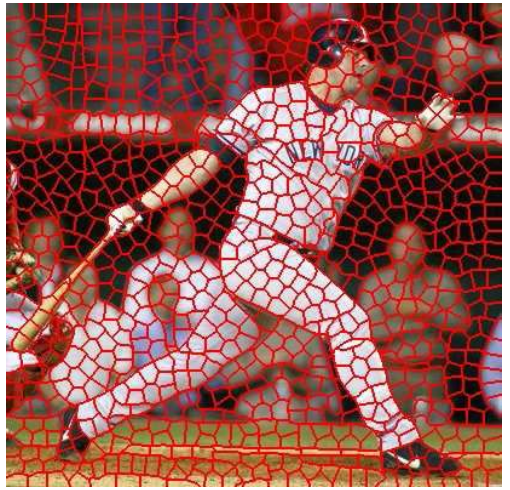
# Clustering example: image segmentation

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

# Segmentation for feature support or efficiency


50x50
Patch

[Felzenszwalb and Huttenlocher 2004]



[Shi and Malik 2001]

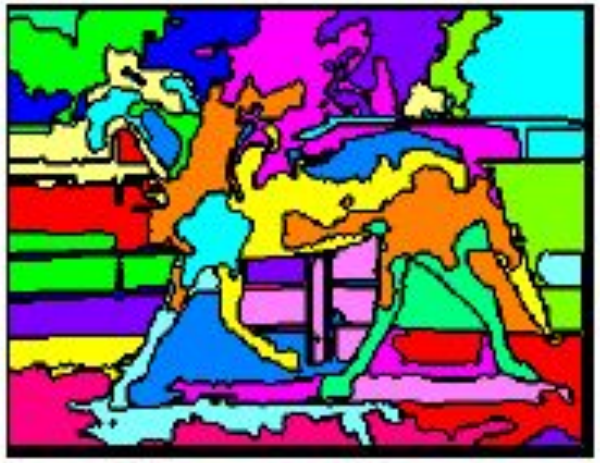
[Hoiem et al. 2005, Mori 2005]

Slide: Derek Hoiem

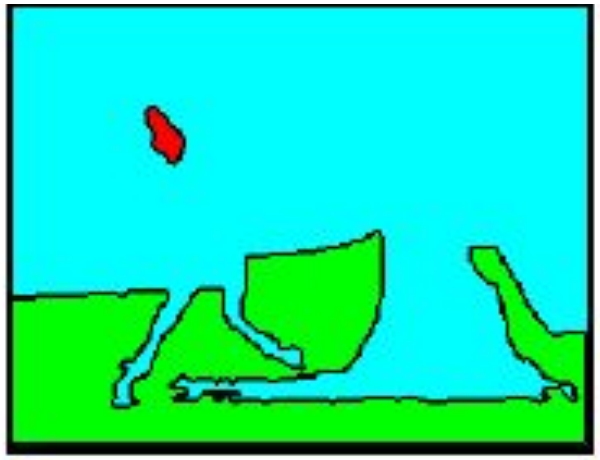
# Segmentation as a result

Rother et al. 2004

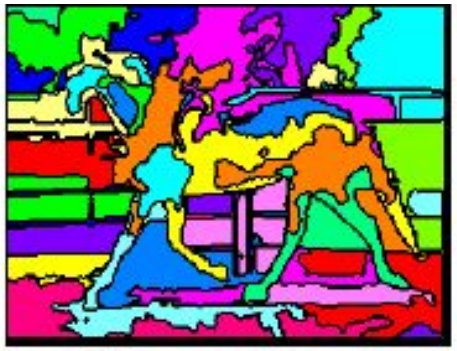
# Types of segmentations

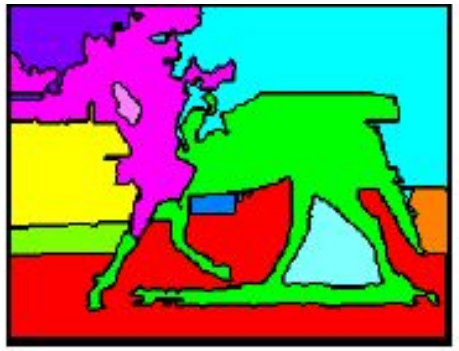


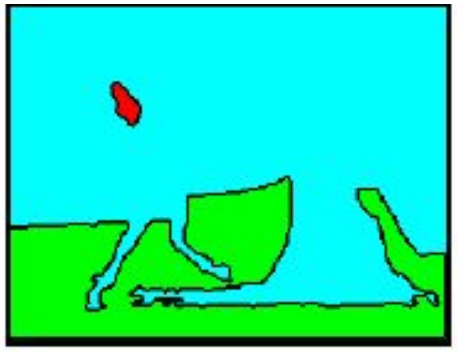
Oversegmentation



Undersegmentation



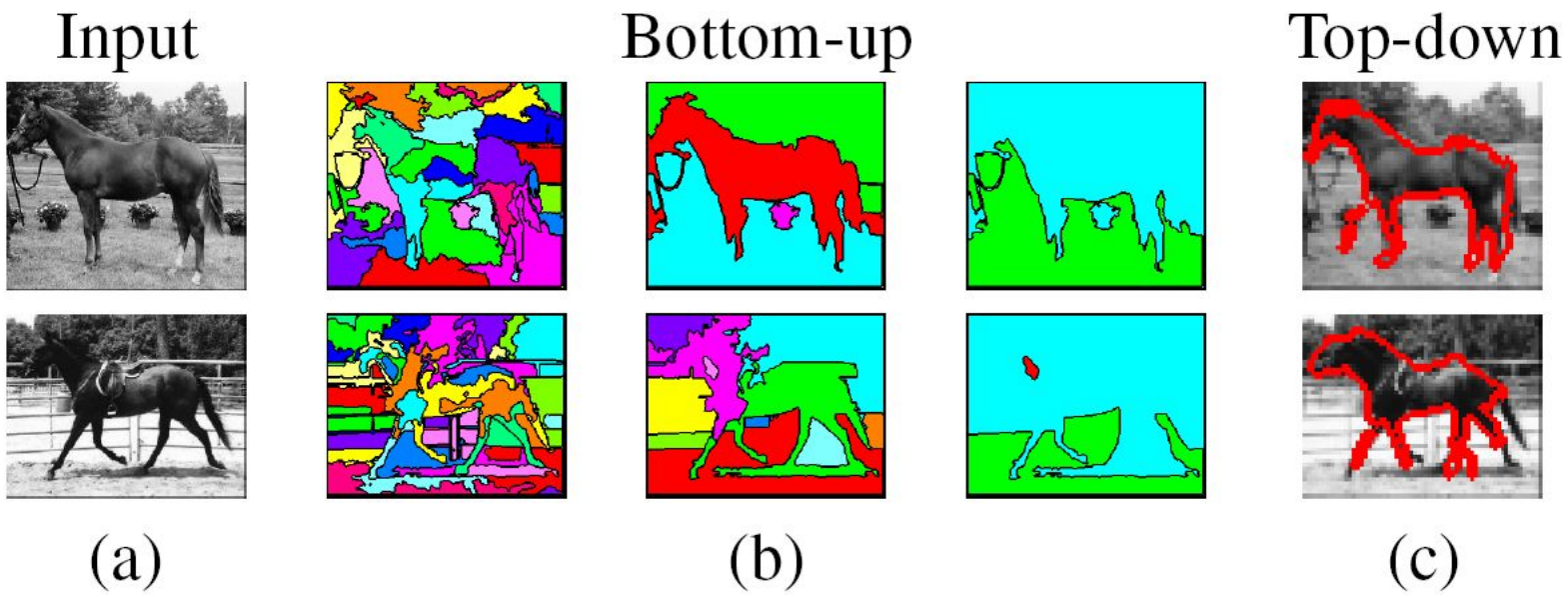




Multiple Segmentations

# Major processes for segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object




[Levin and Weiss 2006]

Clustering: 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?

Slide: Derek Hoiem

# 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

Slide: Derek Hoiem

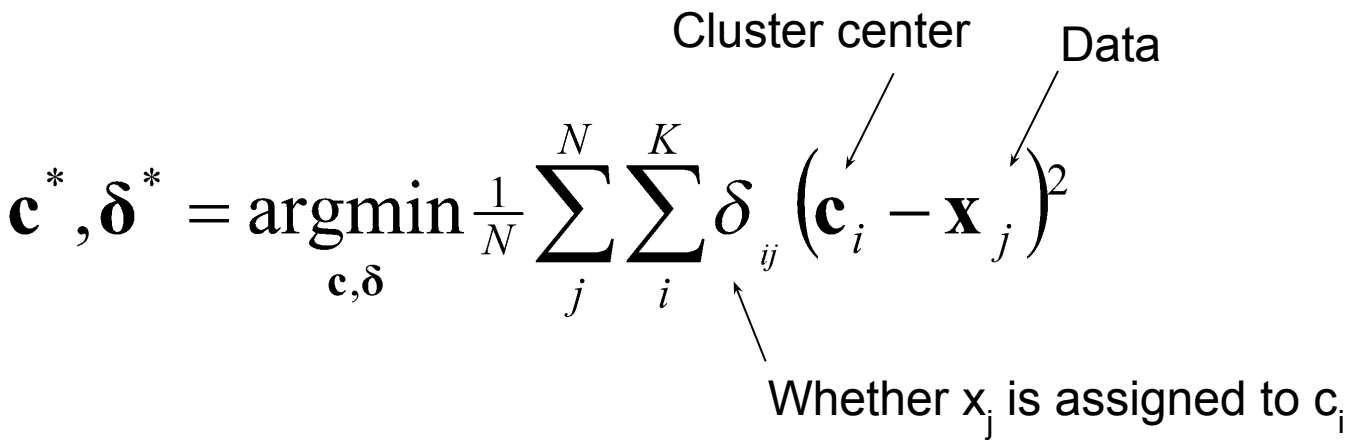
# How do we cluster?

- 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

# Clustering for Summarization

Goal: cluster to minimize variance in data given clusters

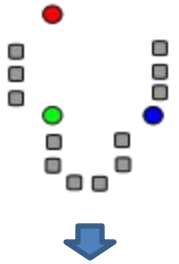
– Preserve information



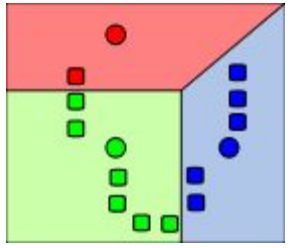

Slide: Derek Hoiem

# K-means algorithm

1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster

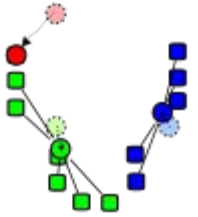
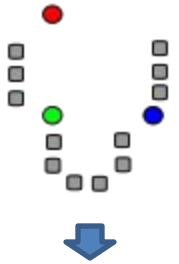


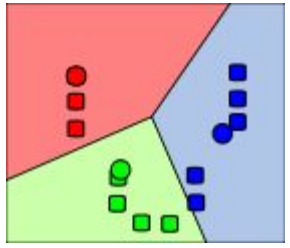
Illustration: http://en.wikipedia.org/wiki/K-means_clustering

# K-means algorithm

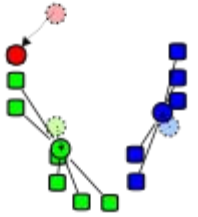
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2

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

# K-means

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

2. Assign each point to the closest center

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

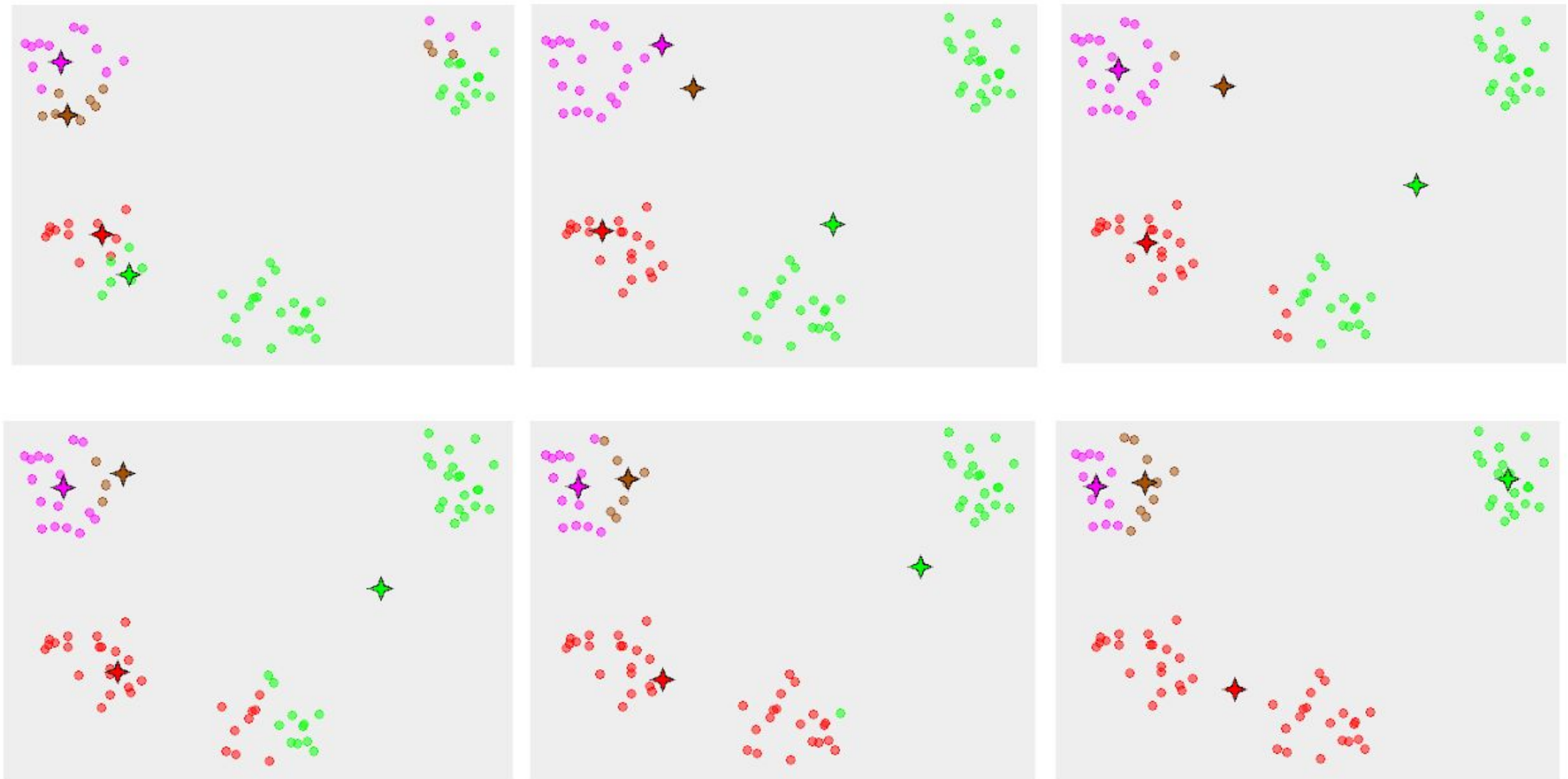
3. Update cluster centers as the mean of the points

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

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

Slide: Derek Hoiem

# K-means converges to a local minimum



# K-means: design choices

- Initialization
  - Randomly select K points as initial cluster center
  - Or greedily choose K points to minimize residual
- Distance measures
  - Traditionally Euclidean, could be others
- Optimization
  - Will converge to a *local minimum*
  - May want to perform multiple restarts

# K-means clustering using intensity or color

Image

Clusters on intensity

Clusters on color

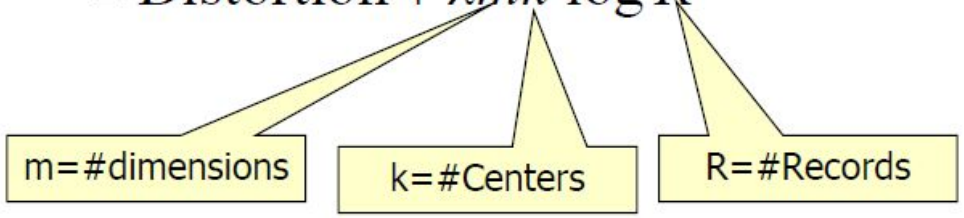
# How to choose the number of clusters?

- Minimum Description Length (MDL) principal for model comparison

- Minimize Schwarz Criterion
  - also called Bayes Information Criteria (BIC)

$$\text{Distortion} + \lambda\,(\#\text{parameters})\log R$$

$$= \text{Distortion} + \lambda mk \log R$$



# 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

Slide: Derek Hoiem

# How to choose the number of clusters?

- Validation set
  - Try different numbers of clusters and look at performance on some downstream task
    - When building dictionaries (discussed later), more clusters typically work better

Slide: Derek Hoiem

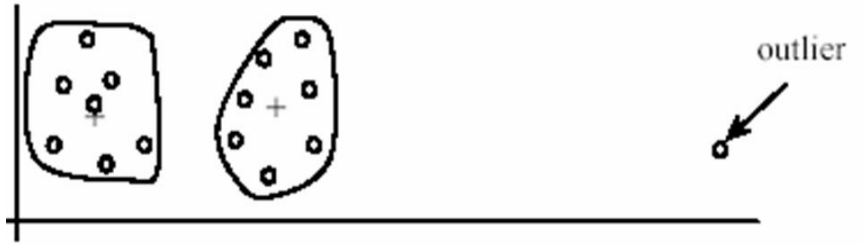
# K-means demos

General
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

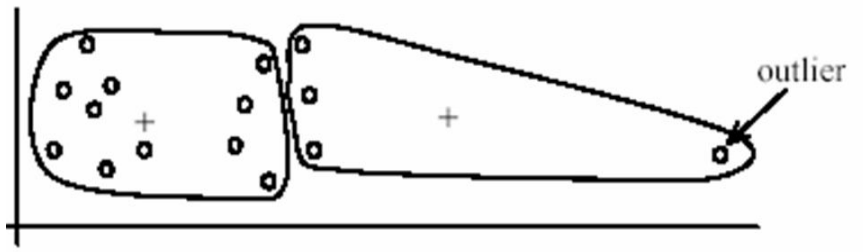
Color clustering
http://www.cs.washington.edu/research/imagedatabase/demo/kmcluster/

# 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
  - Rarely used for pixel segmentation



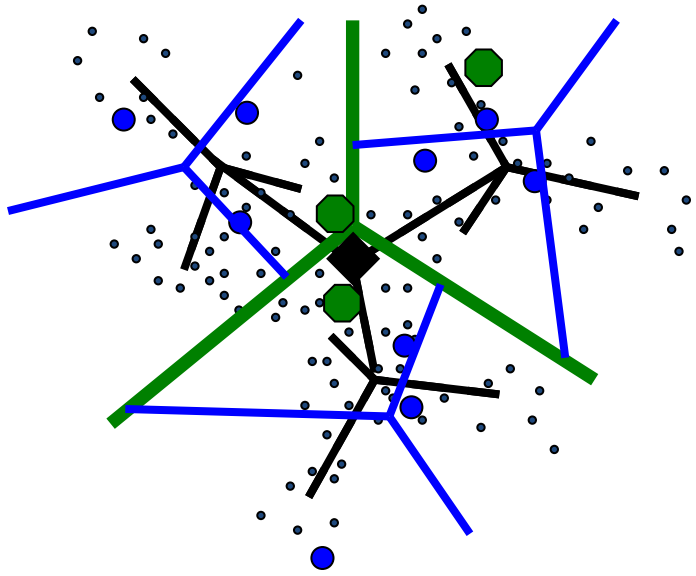

(B): Ideal clusters



# Building Visual Dictionaries

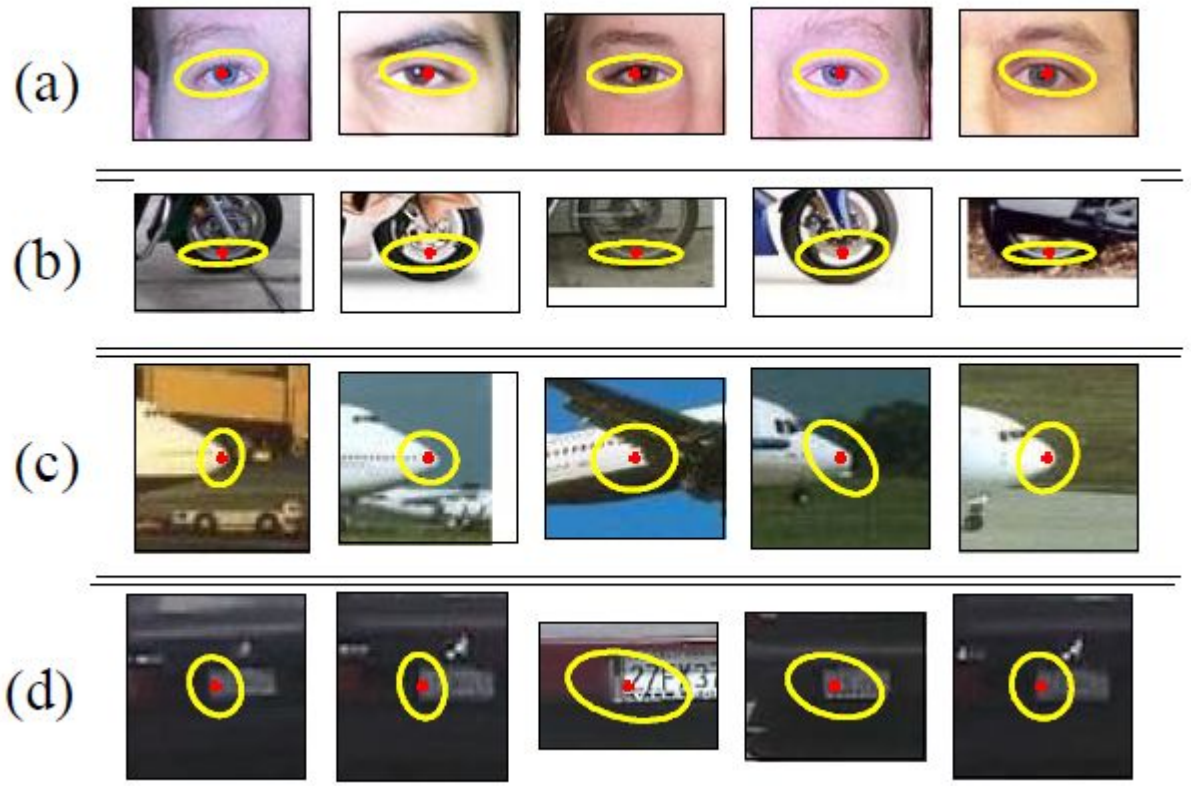
1. Sample patches from a database
   - E.g., 128 dimensional SIFT vectors

2. Cluster the patches
   - Cluster centers are the dictionary



3. Assign a codeword (number) to each new patch, according to the nearest cluster

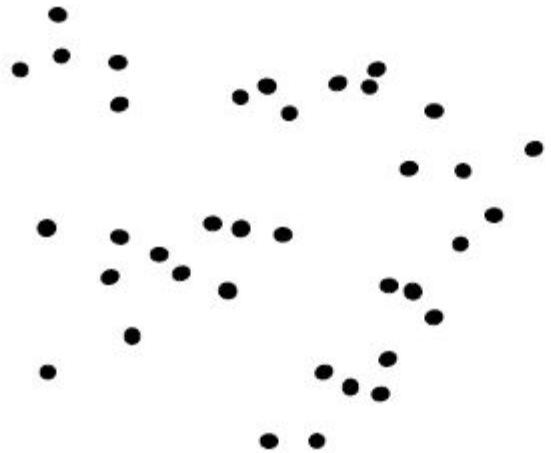
# Examples of learned codewords




Most likely codewords for 4 learned “topics”

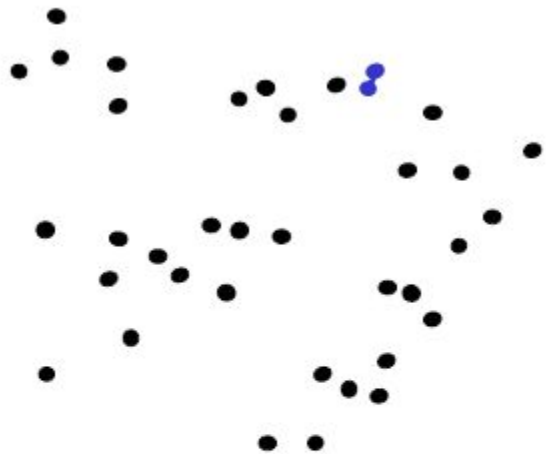
http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic05b.pdf Sivic et al. ICCV 2005

# Agglomerative clustering



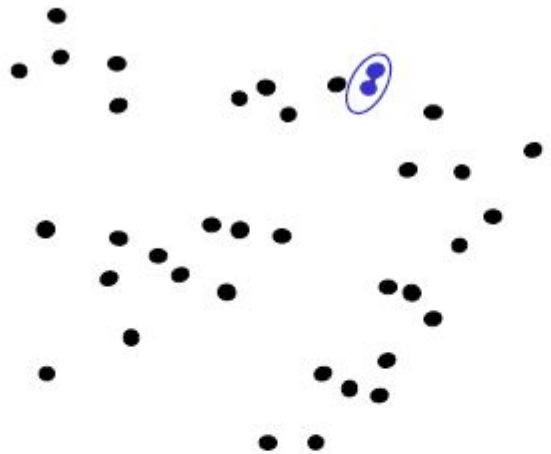
1. Say "Every point is its own cluster"

# Agglomerative clustering



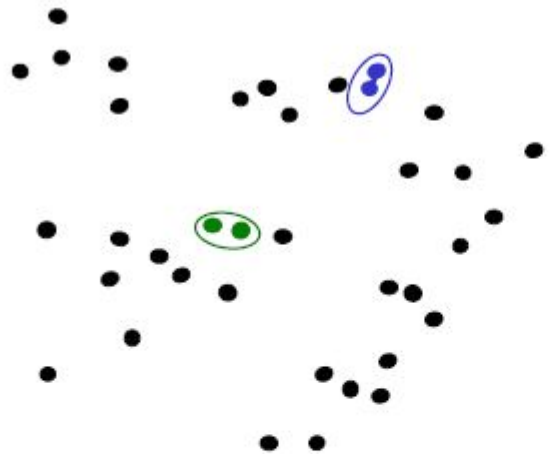
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

# Agglomerative clustering



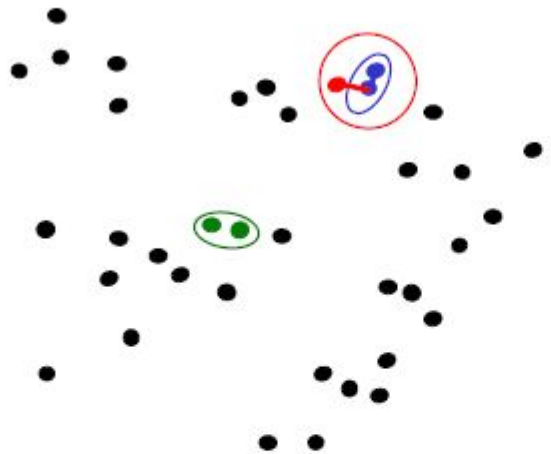
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

# Agglomerative clustering

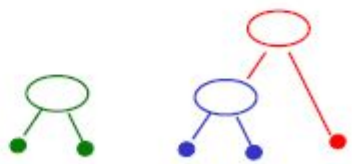


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

# Agglomerative clustering



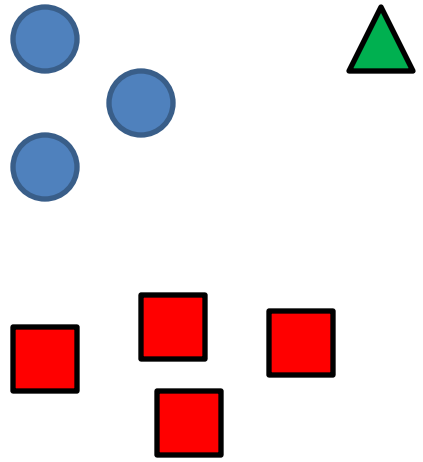
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat



# Agglomerative clustering

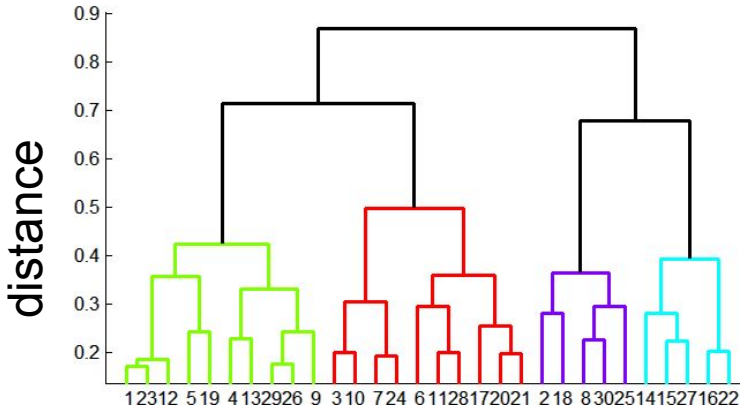
## How to define cluster similarity?

- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids



## How many clusters?

- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges



# Conclusions: Agglomerative Clustering

## Good

- Simple to implement, widespread application
- Clusters have adaptive shapes
- Provides a hierarchy of clusters

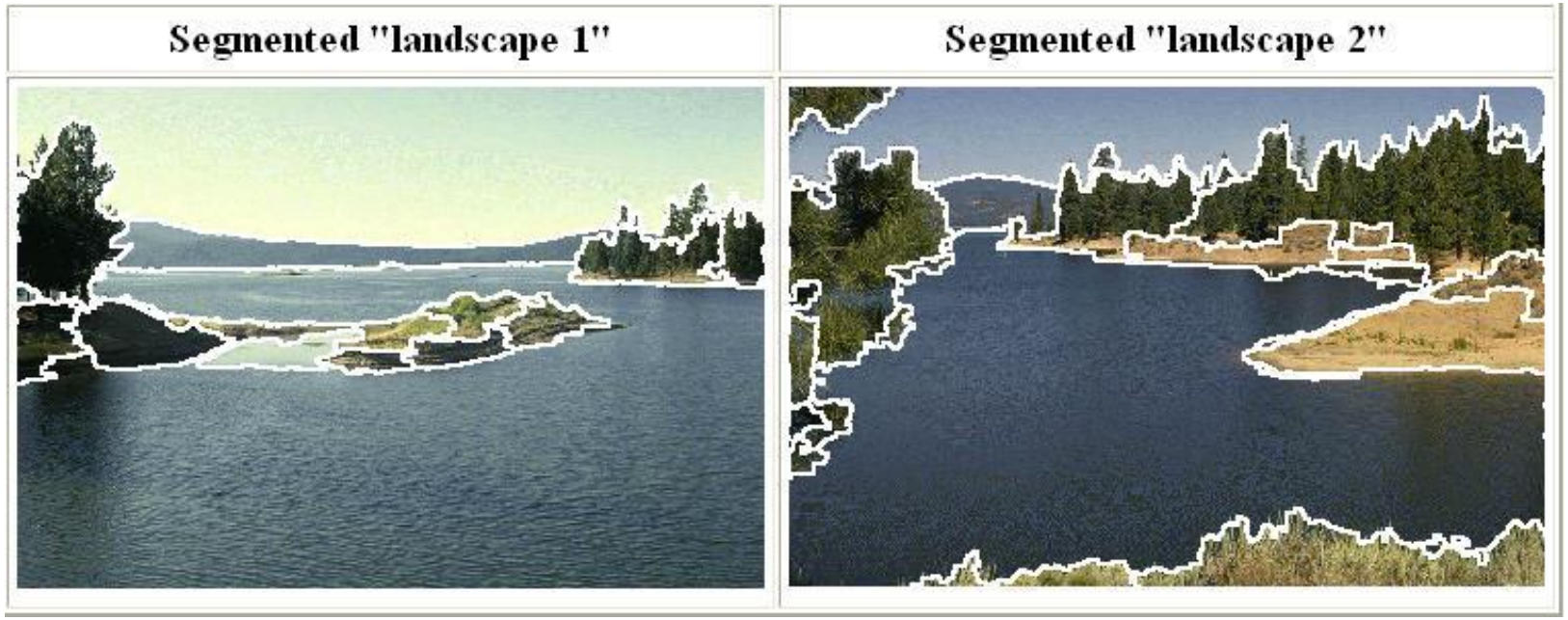
## Bad

- May have imbalanced clusters
- Still have to choose number of clusters or threshold
- Need to use an “ultrametric” to get a meaningful hierarchy

# Mean shift segmentation

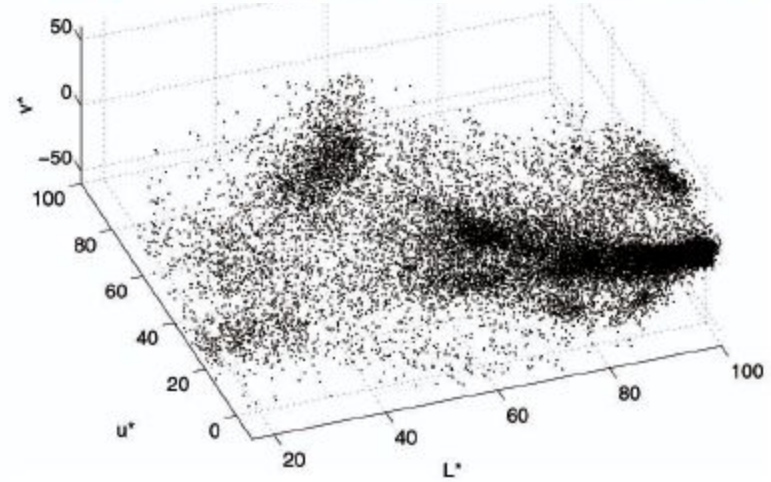
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

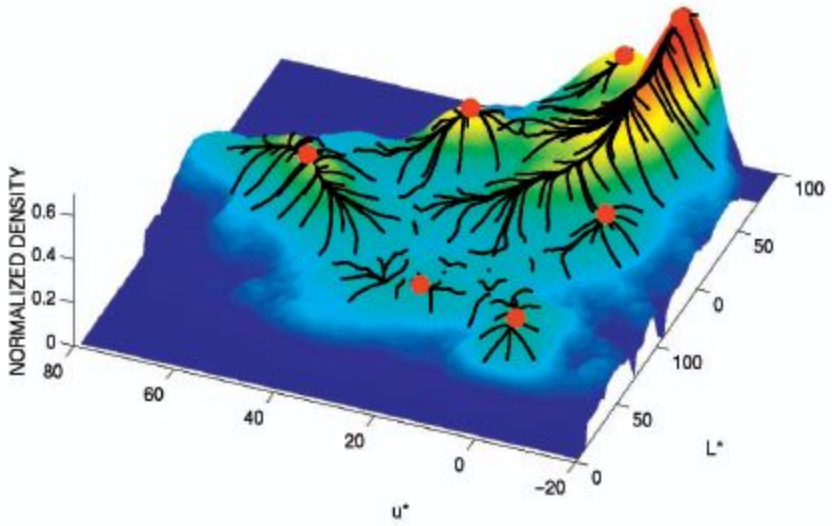
- Versatile technique for clustering-based segmentation

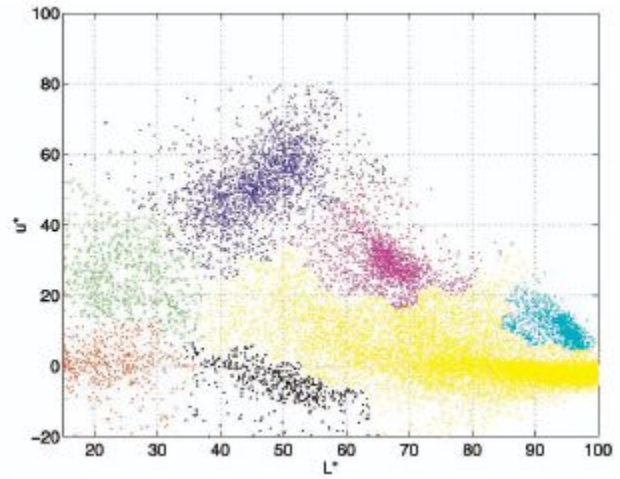


# Mean shift algorithm

- Try to find *modes* of this non-parametric density



# Kernel density estimation

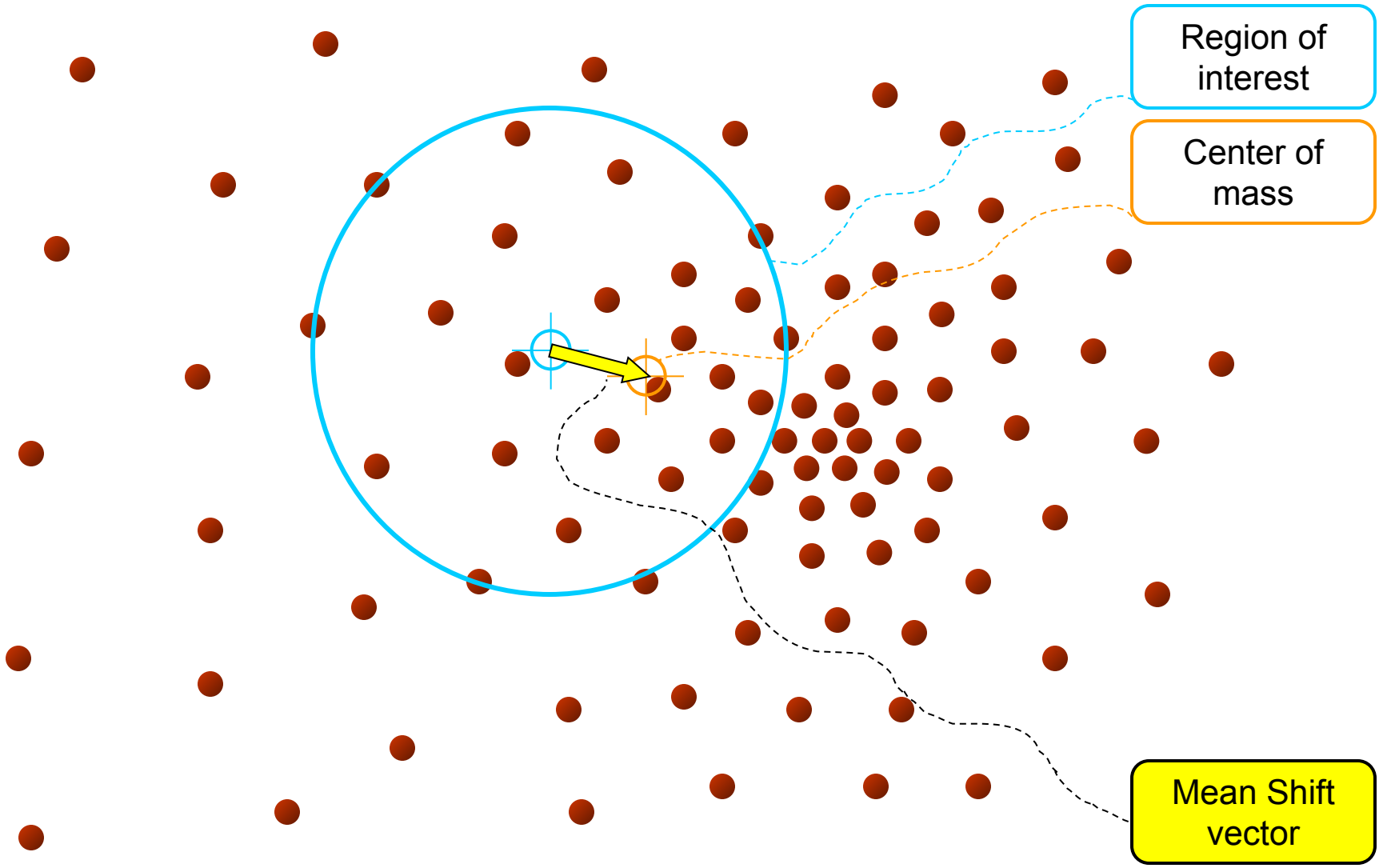
Kernel density estimation function

$$\widehat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Gaussian kernel

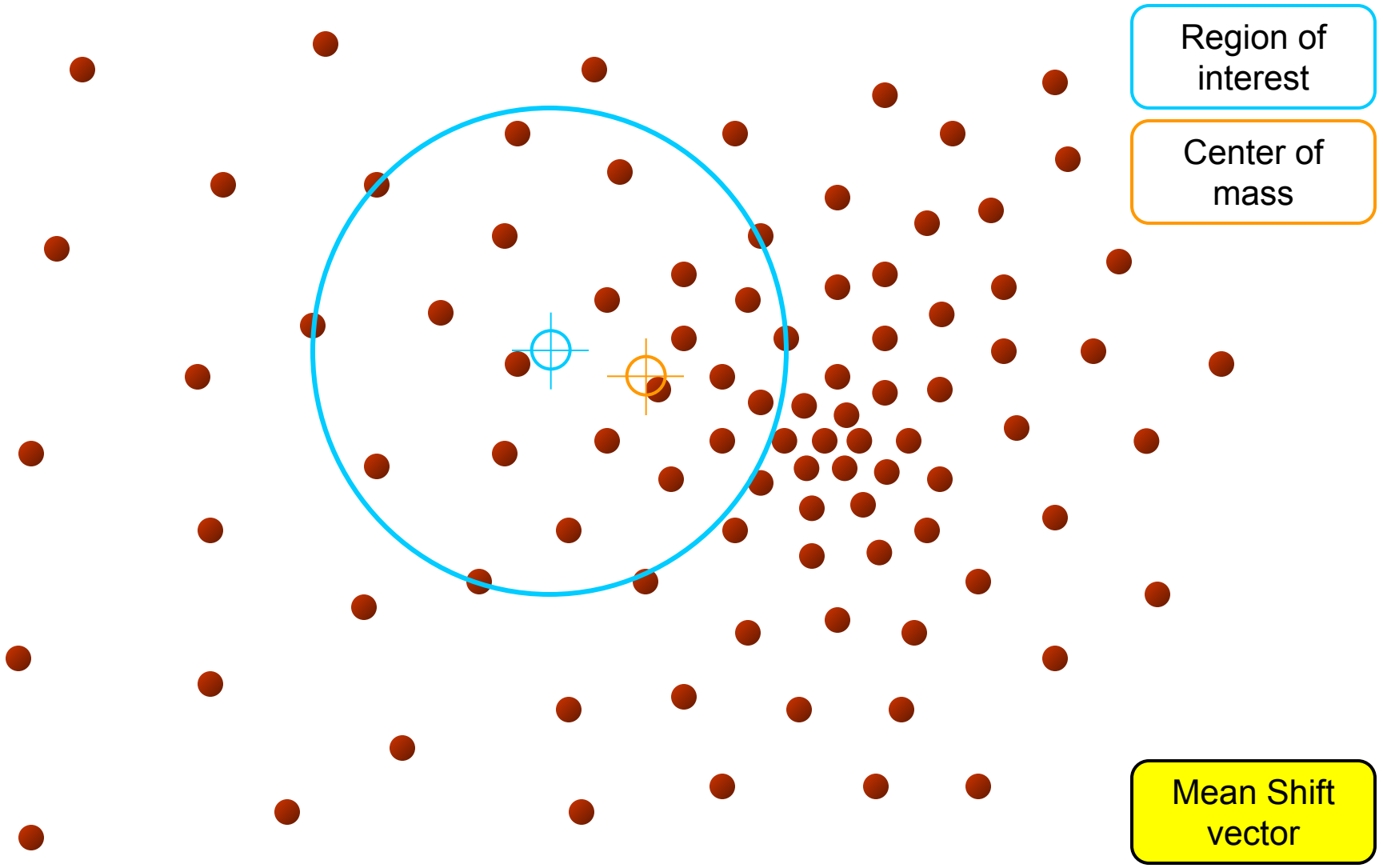
$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}.$$

# Mean shift



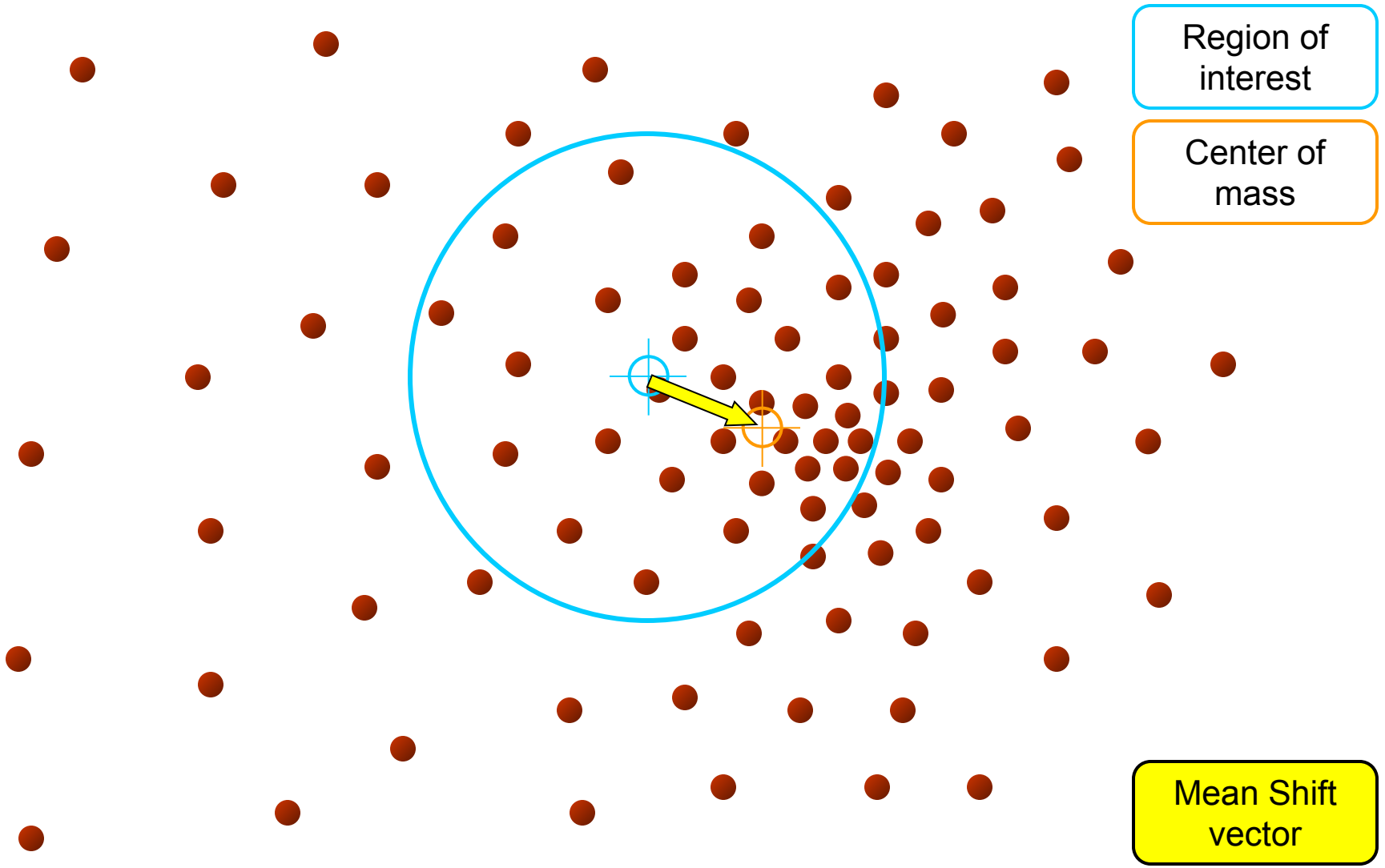

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



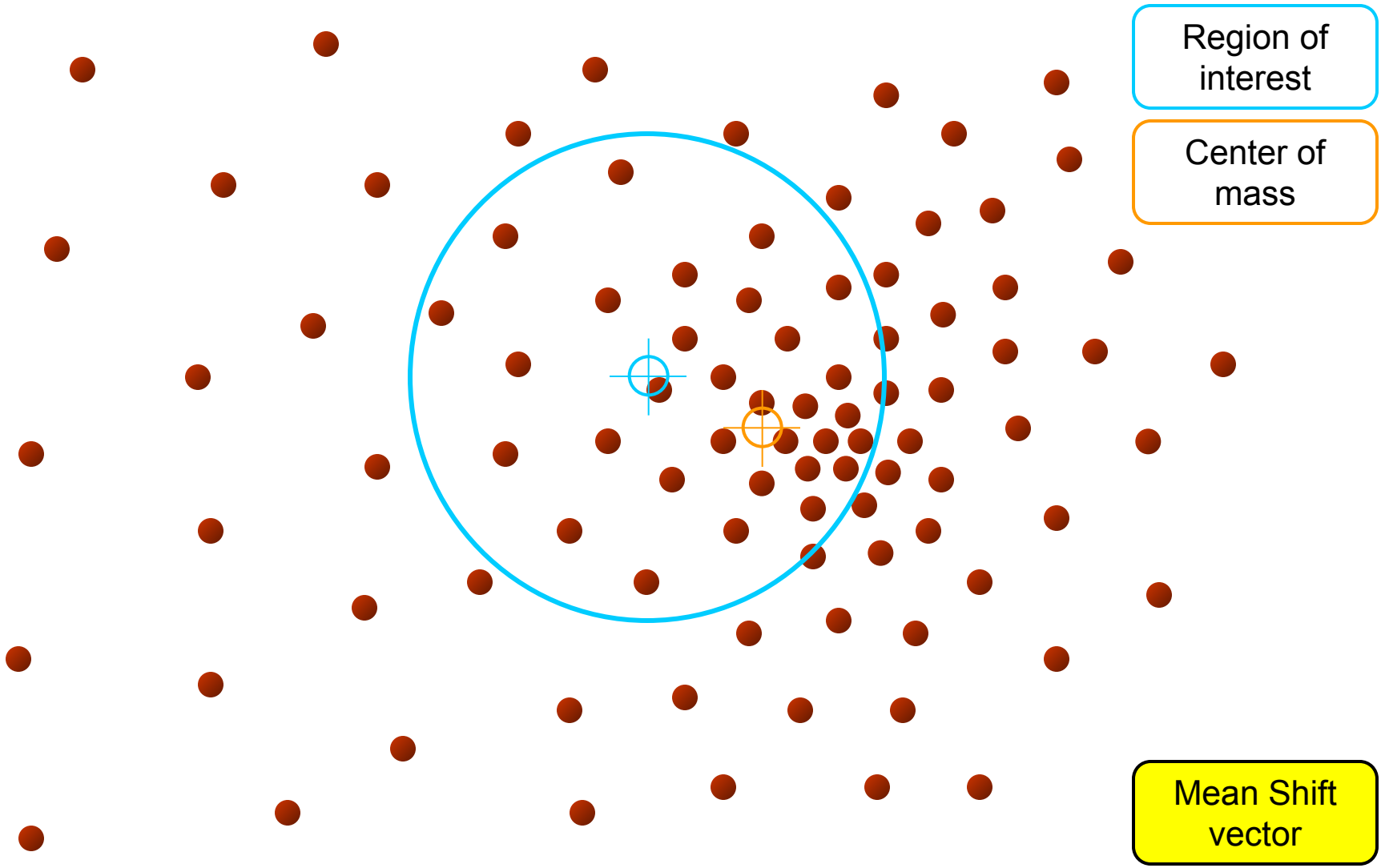

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



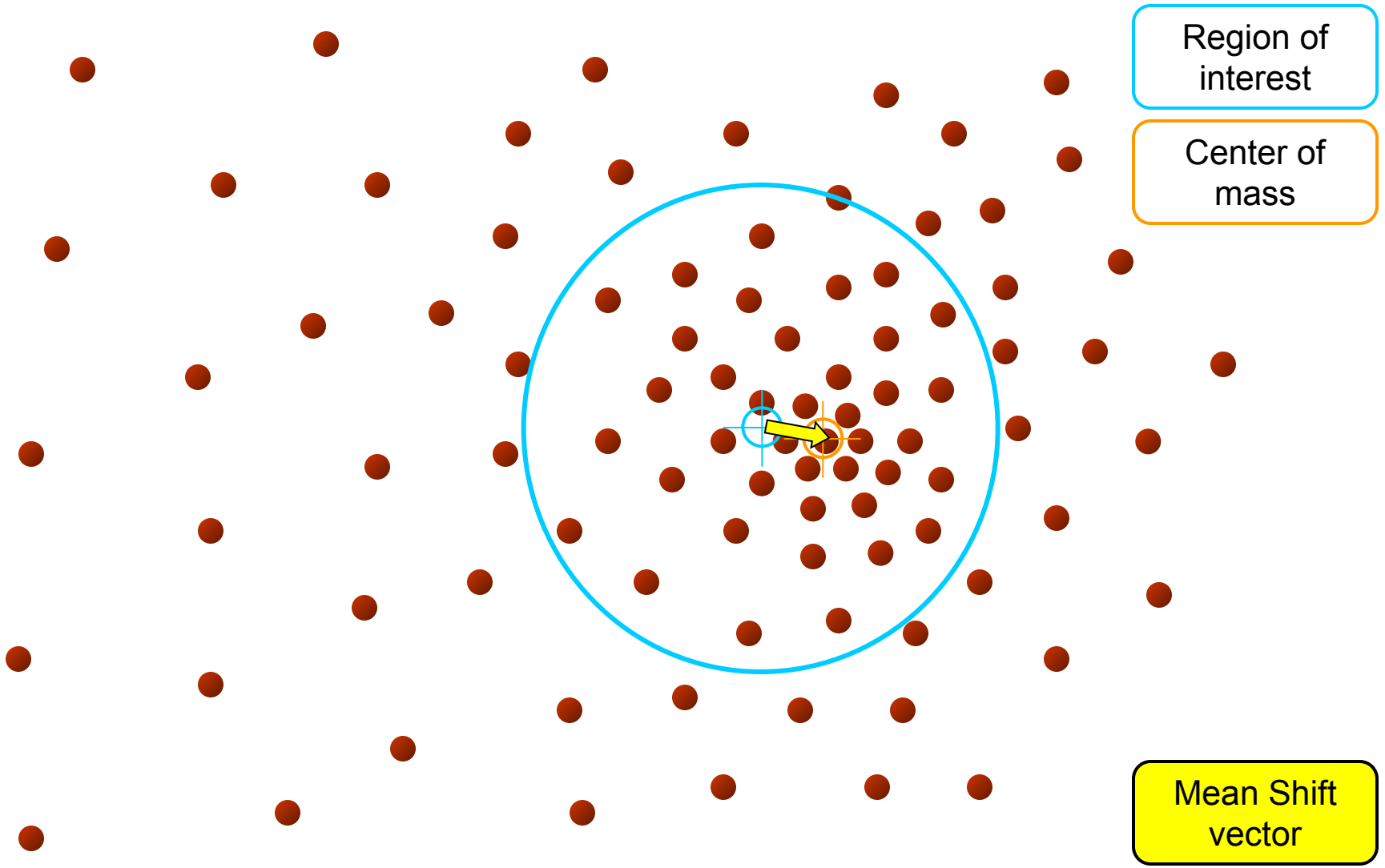

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



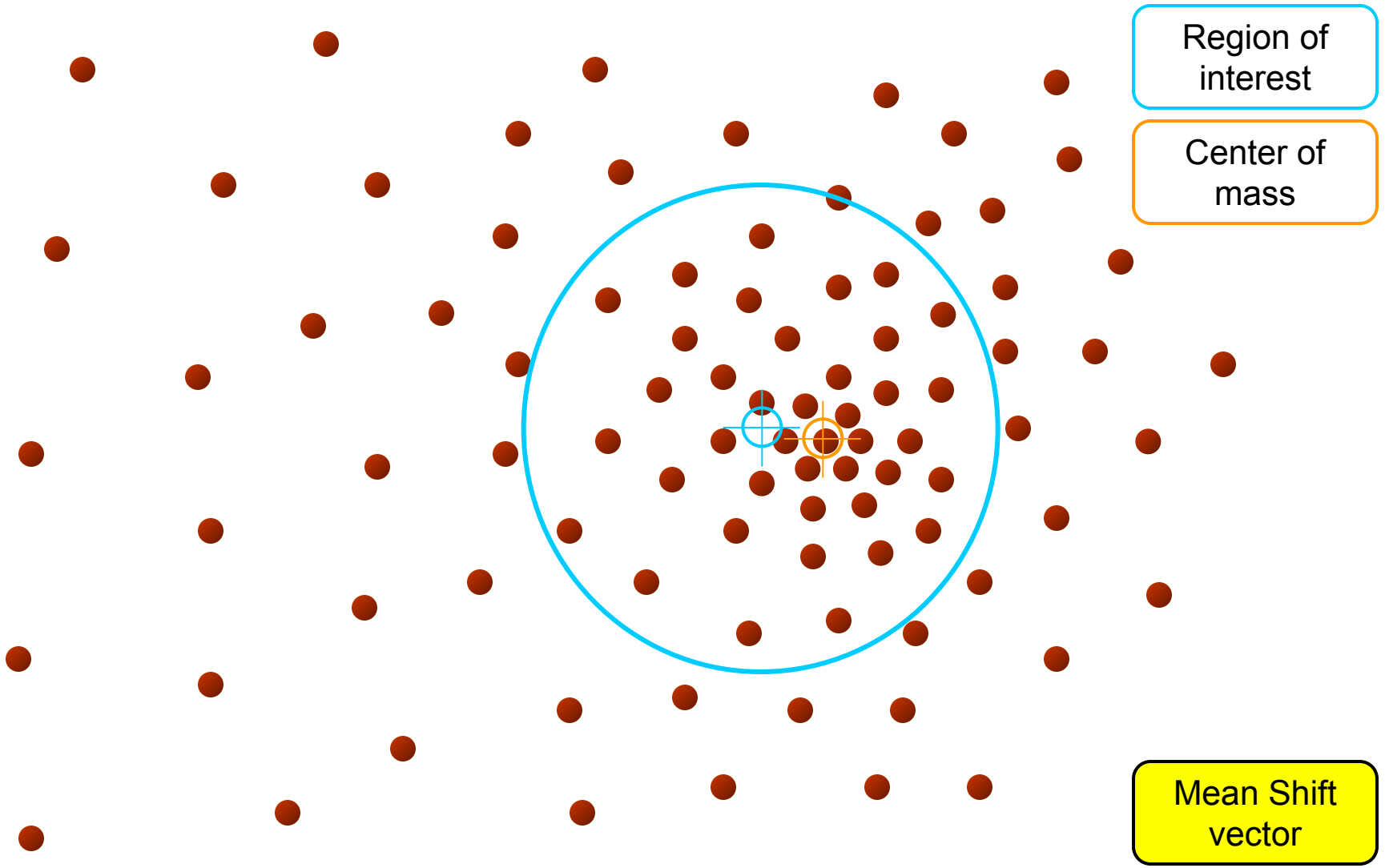

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



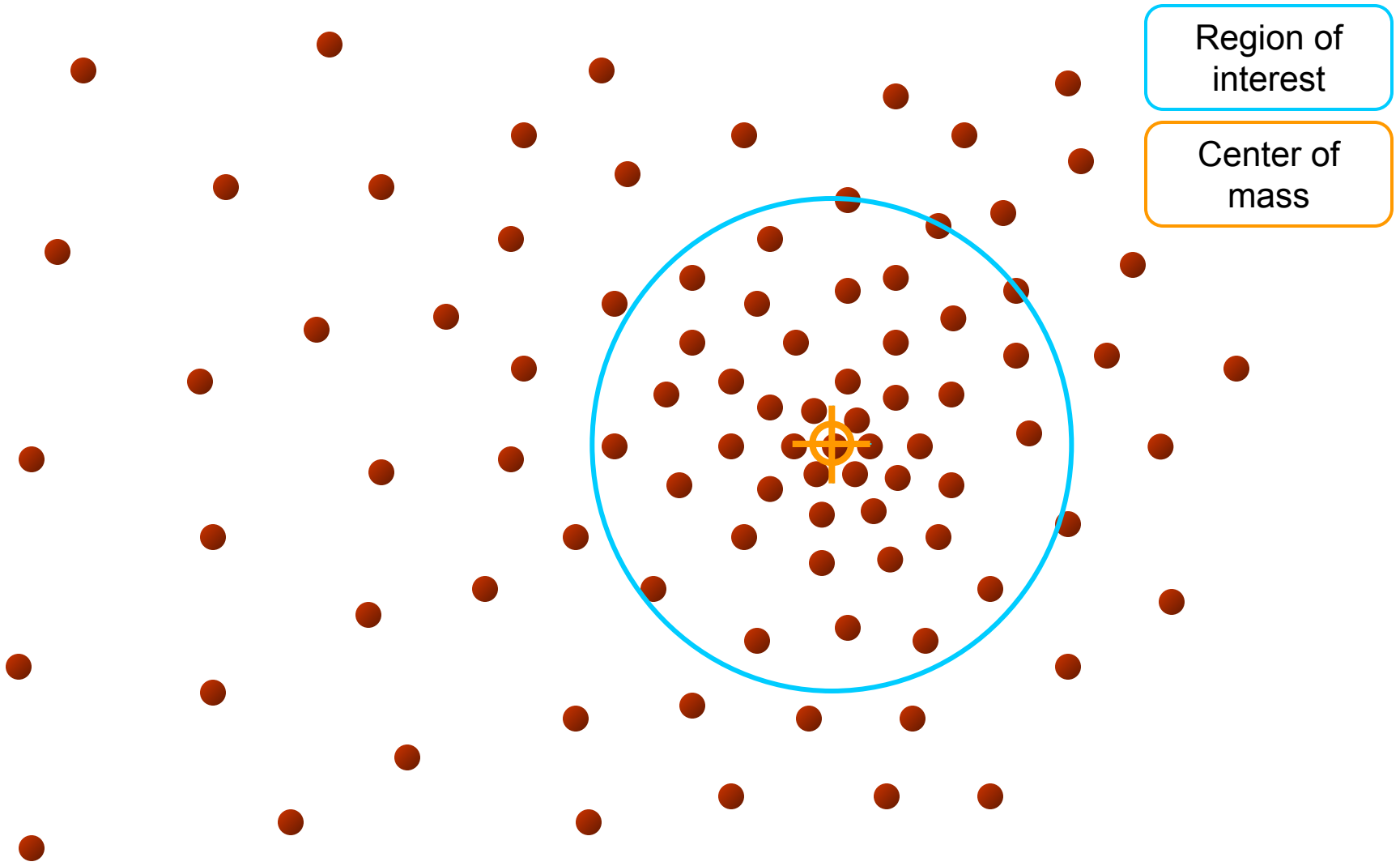

Slide by Y. Ukrainitz & B. Sarel

# Mean shift




Slide by Y. Ukrainitz & B. Sarel

# Mean shift




Slide by Y. Ukrainitz & B. Sarel

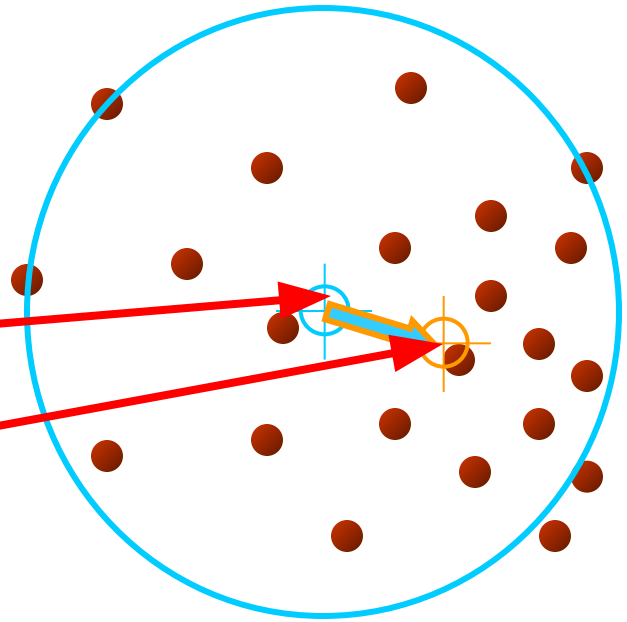
# Computing the Mean Shift

Simple Mean Shift procedure:

- Compute mean shift vector

- Translate the Kernel window by **m(x)**

$$\mathbf{m}(\mathbf{x}) = \left[ \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left( \frac{\left\| \mathbf{x} - \mathbf{x}_i \right\|^2}{h} \right)}{\sum_{i=1}^{n} g\left( \frac{\left\| \mathbf{x} - \mathbf{x}_i \right\|^2}{h} \right)} - \mathbf{x} \right]$$

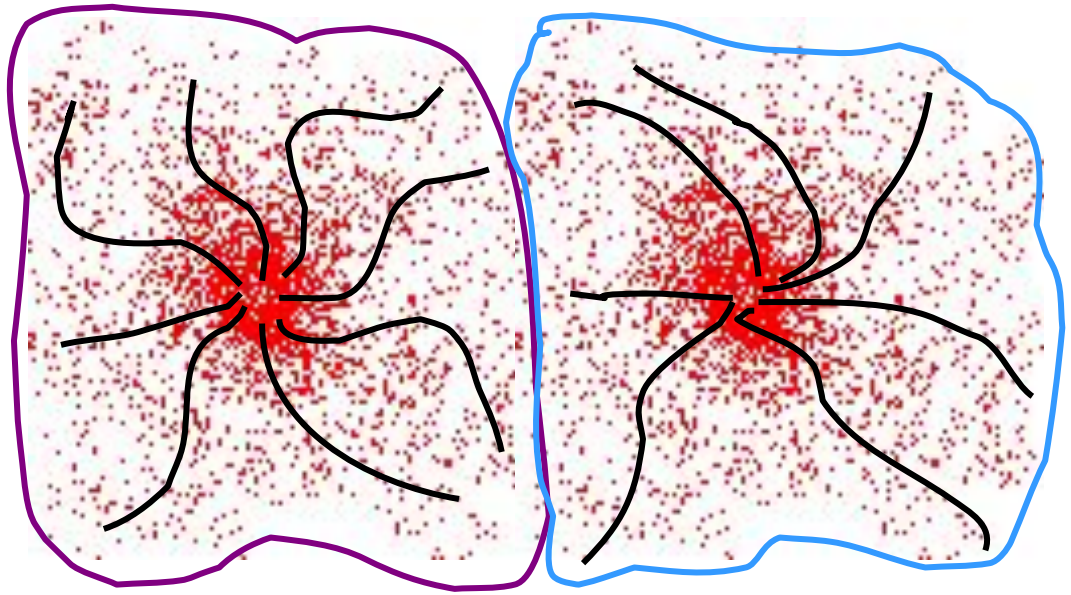


$$g(\mathbf{x}) = -k'(\mathbf{x})$$

Slide by Y. Ukrainitz & B. Sarel

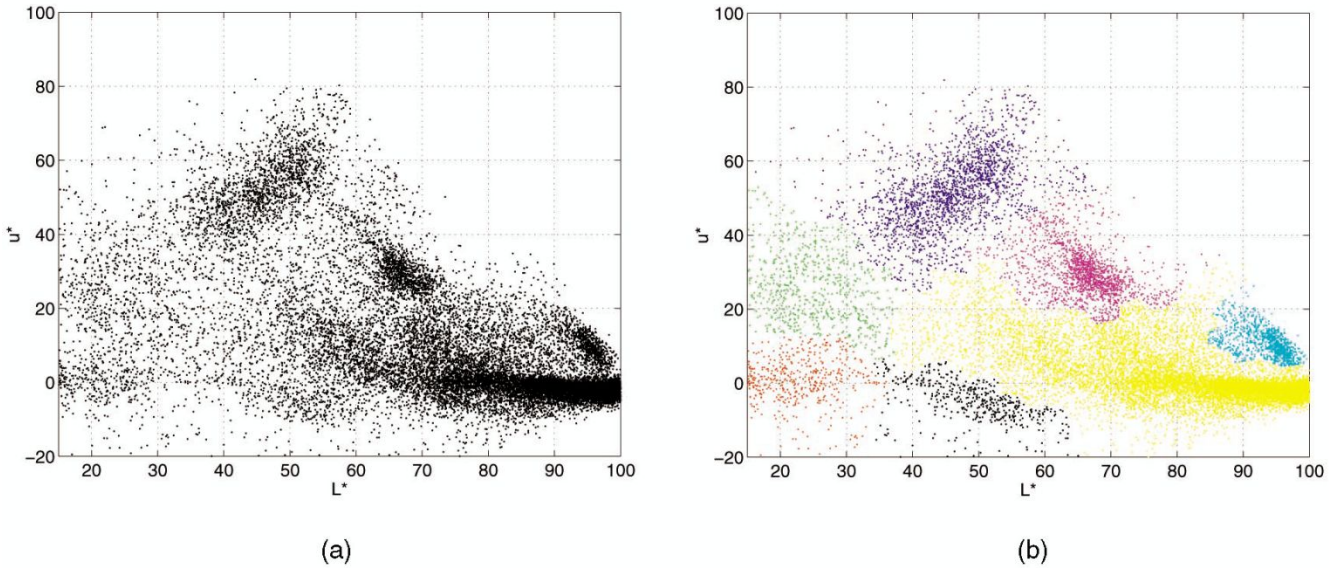
# Attraction basin

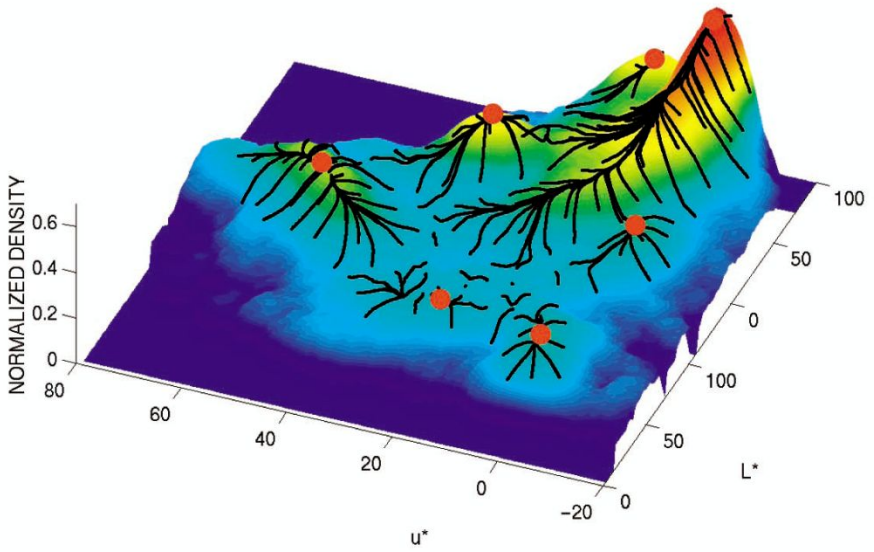
- Attraction basin: the region for which all trajectories lead to the same mode
- Cluster: all data points in the attraction basin of a mode



Slide by Y. Ukrainitz & B. Sarel

# Attraction basin



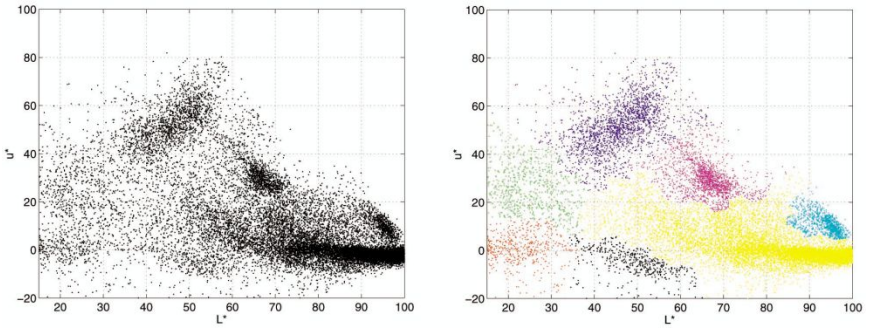



# Mean shift clustering

- The mean shift algorithm seeks *modes* of the given set of points
  1. Choose kernel and bandwidth
  2. For each point:
     a) Center a window on that point
     b) Compute the mean of the data in the search window
     c) Center the search window at the new mean location
     d) Repeat (b,c) until convergence
  3. Assign points that lead to nearby modes to the same cluster

# Segmentation by Mean Shift

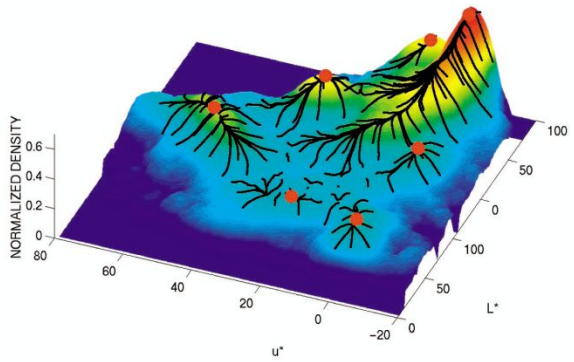
- Compute features for each pixel (color, gradients, texture, etc)
- Set kernel size for features $K_f$ and position $K_s$
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that are within width of $K_f$ and $K_s$




(a)



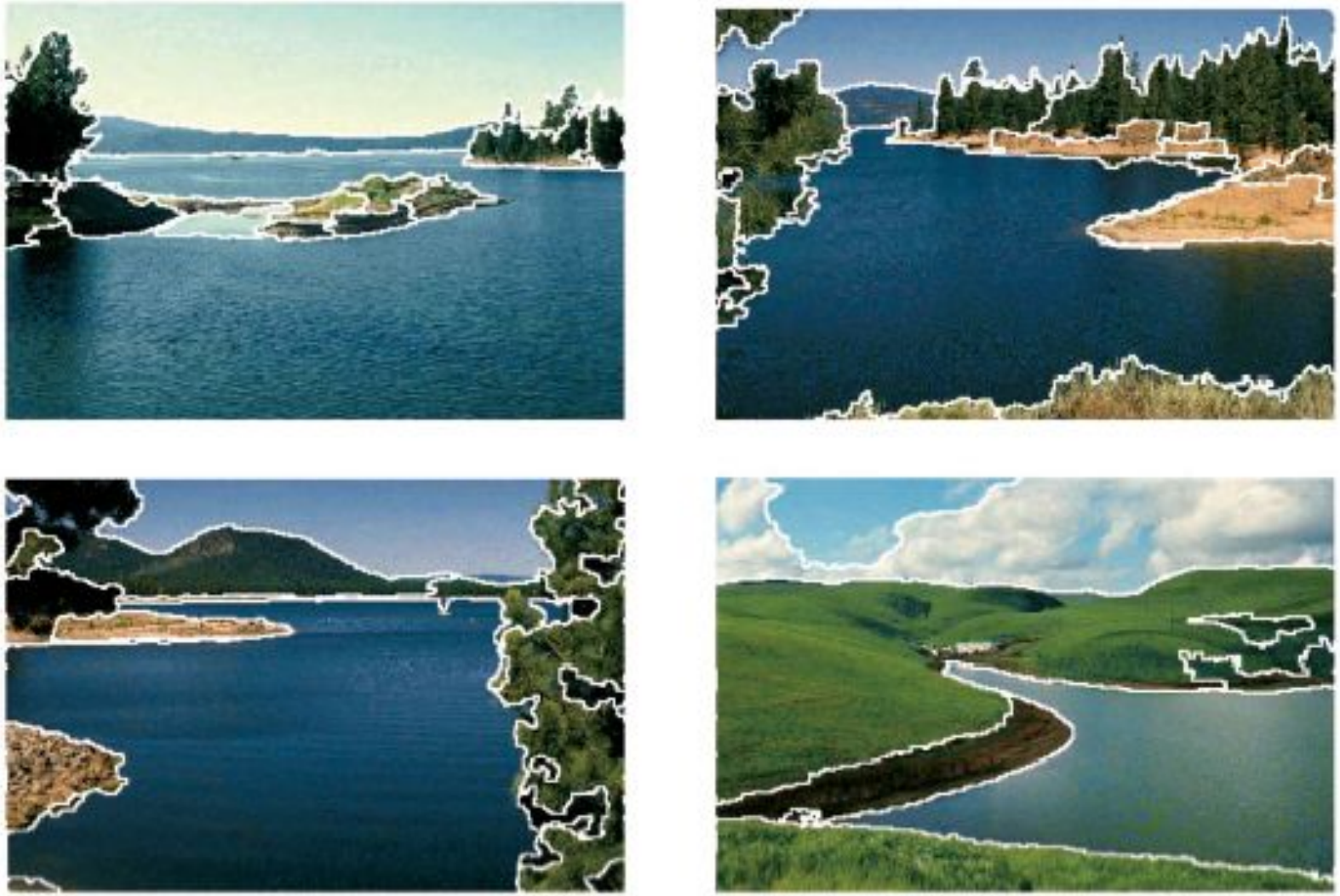

(b)




(c)

# Mean shift segmentation results

Comaniciu and Meer 2002



Comaniciu and Meer 2002

# Mean-shift: other issues

- Speedups
  - Binned estimation
  - Fast search of neighbors
  - Update each window in each iteration (faster convergence)

- Other tricks
  - Use kNN to determine window sizes adaptively

- Lots of theoretical support

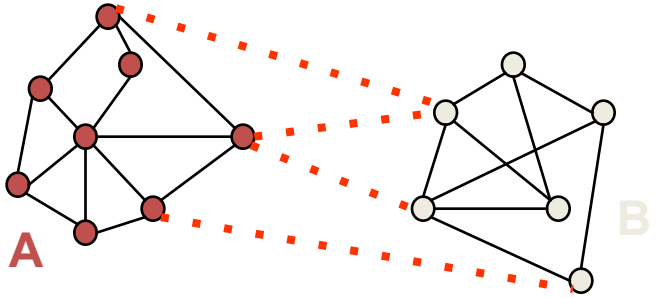
  D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

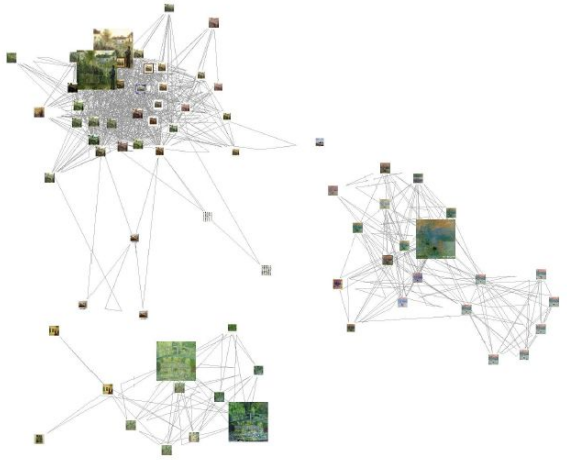
# Mean shift pros and cons

- Pros
  - Good general-practice segmentation
  - Flexible in number and shape of regions
  - Robust to outliers
- Cons
  - Have to choose kernel size in advance
  - Not suitable for high-dimensional features
- When to use it
  - Oversegmentatoin
  - Multiple segmentations
  - Tracking, clustering, filtering applications

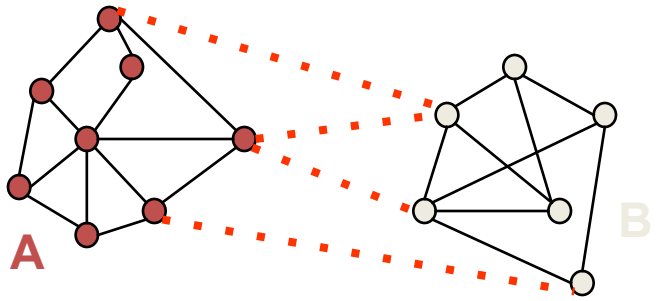
# Spectral clustering

Group points based on links in a graph



# Cuts in a graph




Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

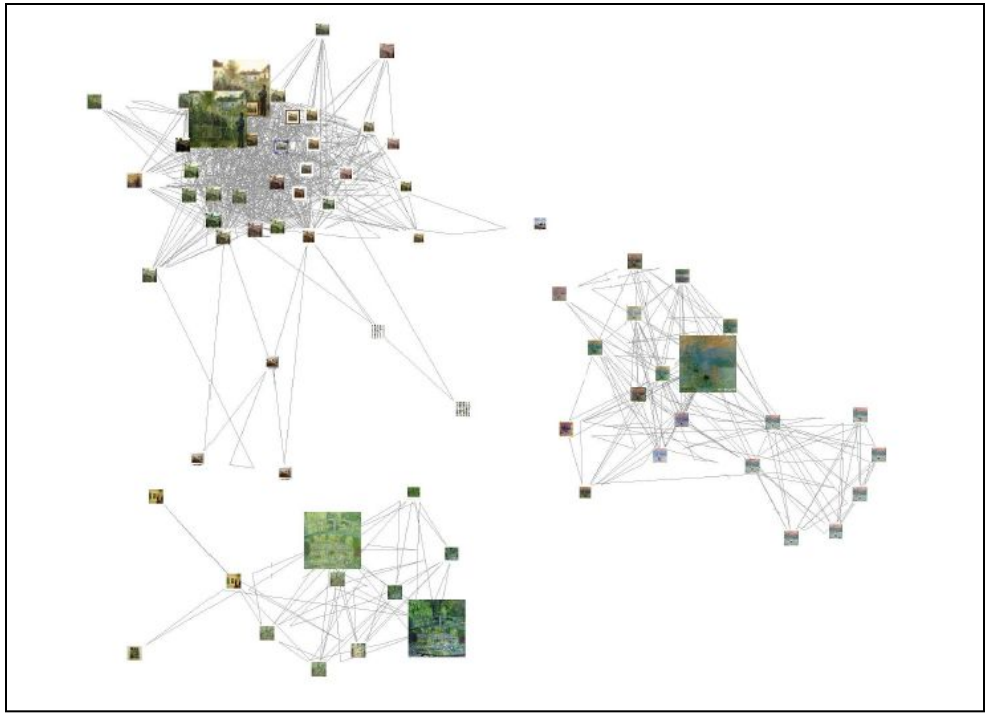
- volume(A) = sum of costs of all edges that touch A

Source: Seitz

# Normalized cuts for segmentation

# Visual PageRank

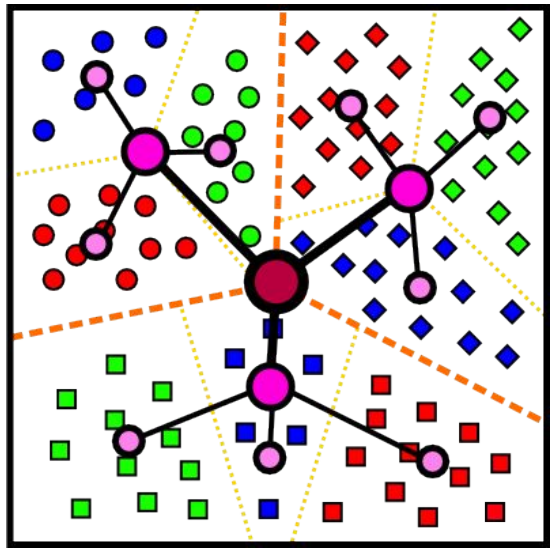
- Determining importance by random walk
  - What's the probability that you will randomly walk to a given node?
    - Create adjacency matrix based on visual similarity
    - Edge weights determine probability of transition



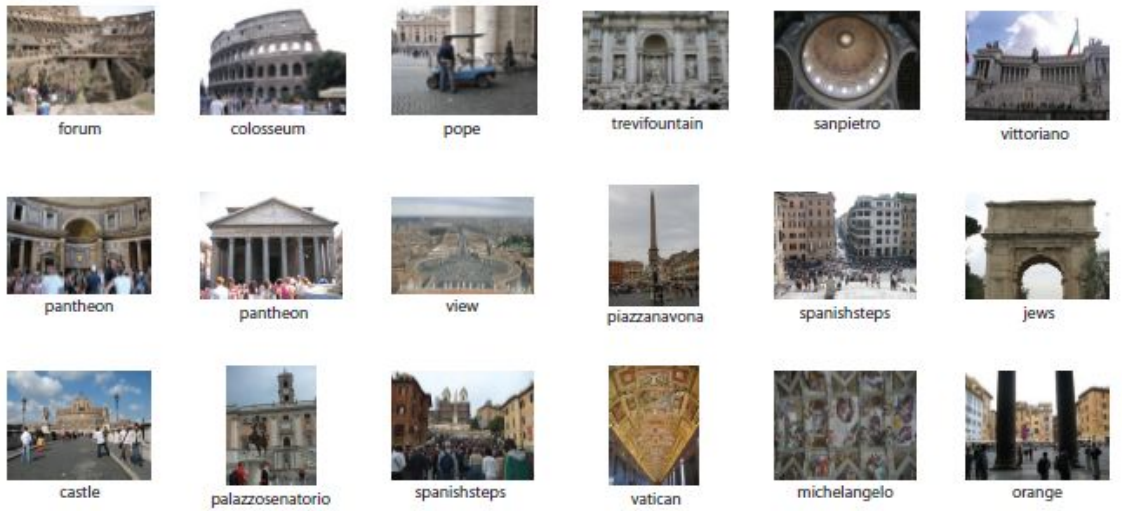
Jing Baluja 2008

# Which algorithm to try first?

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



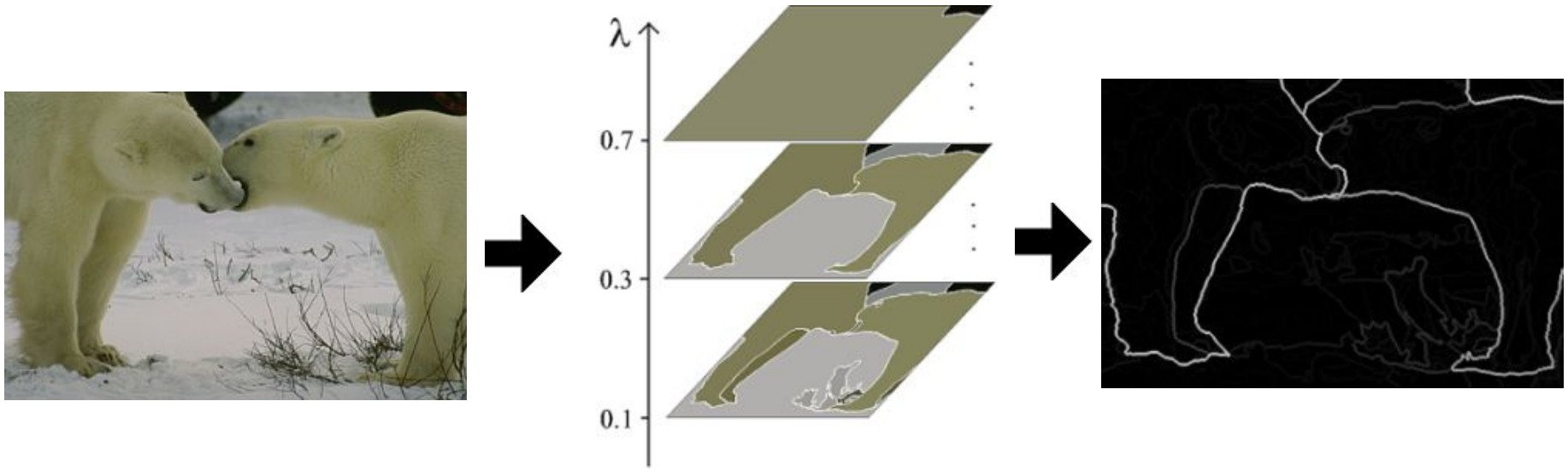
Quantization for computing histograms




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

# Which algorithm to use?

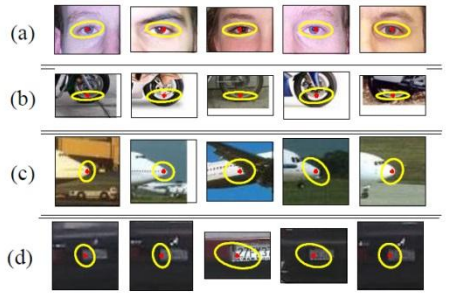
- Image segmentation: agglomerative clustering
  - More flexible with distance measures (e.g., can be based on boundary prediction)
  - Adapts better to specific data
  - Hierarchy can be useful




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

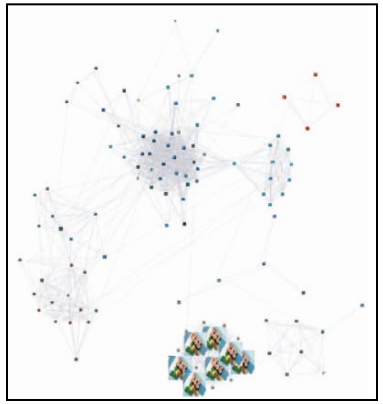
# Things to remember

- K-means useful for summarization, building dictionaries of patches, general clustering




- Agglomerative clustering useful for segmentation, general clustering

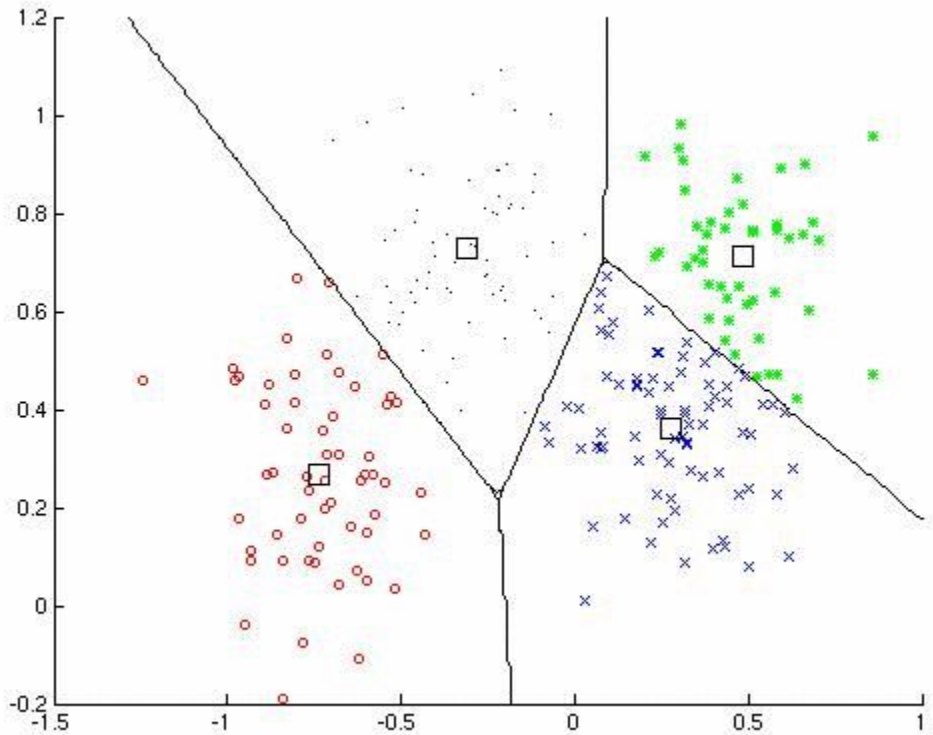
- Spectral clustering useful for determining relevance, summarization, segmentation



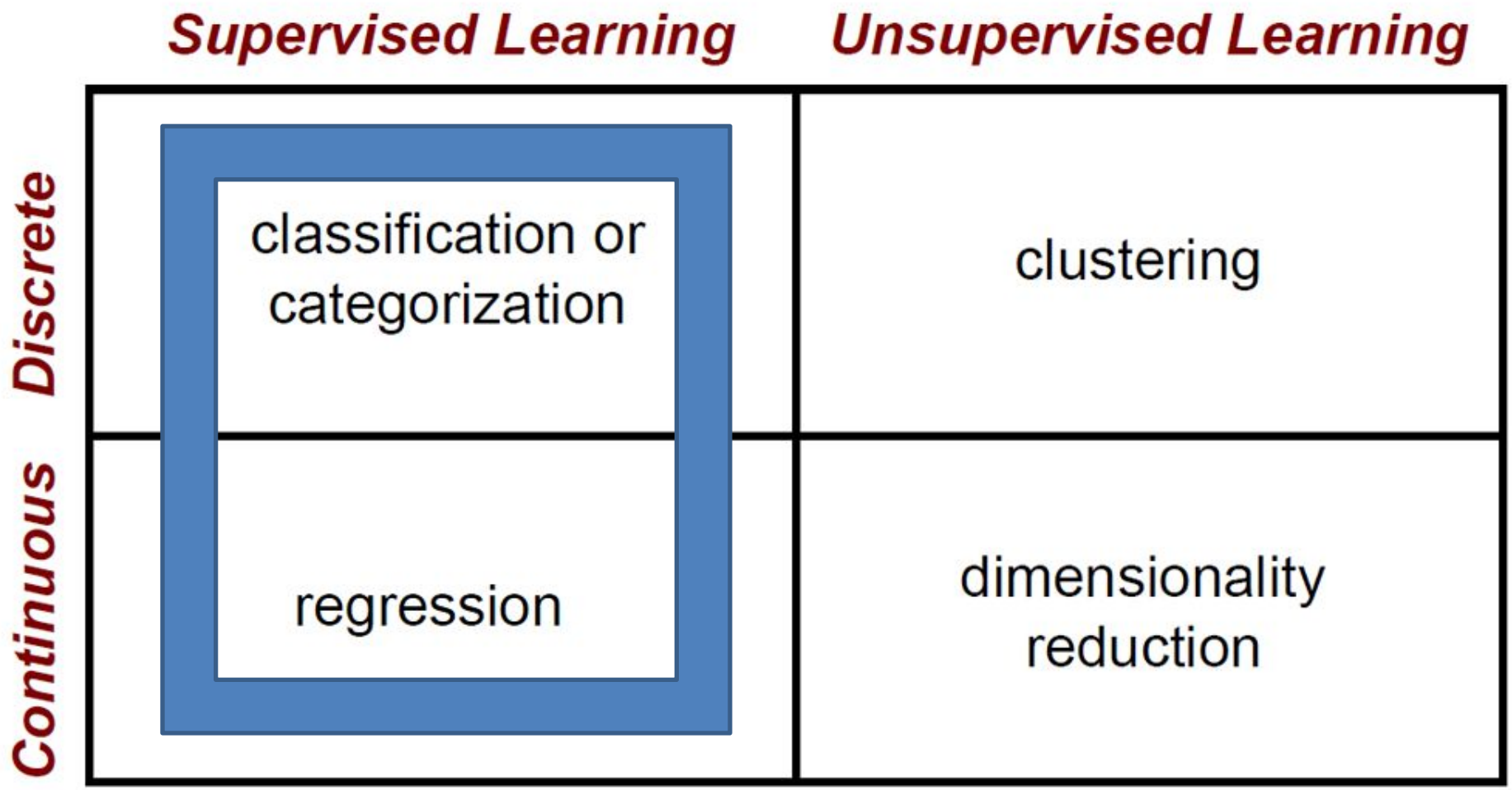
# Clustering

Key algorithm

- K-means



# Machine Learning Problems



# The machine learning framework

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

f(  ) “apple”

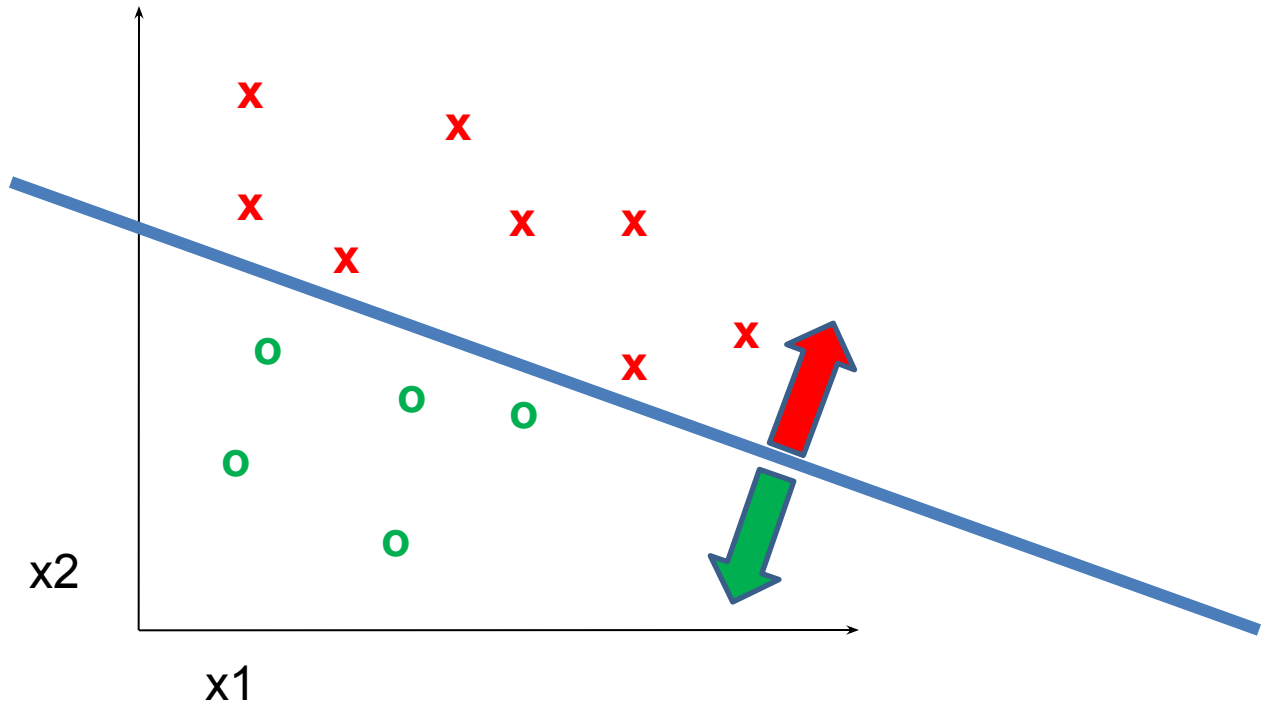
f(  ) “tomato”

f(  ) “cow”

Slide credit: L. Lazebnik

# Learning a classifier

Given some set of features with corresponding labels, learn a function to predict the labels from the features



# 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?

Slide credit: L. Lazebnik

# Very brief tour of some classifiers

- **K-nearest neighbor**
- **SVM**
- Boosted Decision Trees
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- RBMs
- Deep Convolutional Network
- Attentional models or “Transformers”
- Etc.

# Generative vs. Discriminative Classifiers

## Generative Models

- Represent both the data and the labels
- Often, makes use of conditional independence and priors
- Examples
  - Naïve Bayes classifier
  - Bayesian network

- Models of data may apply to future prediction problems

## Discriminative Models

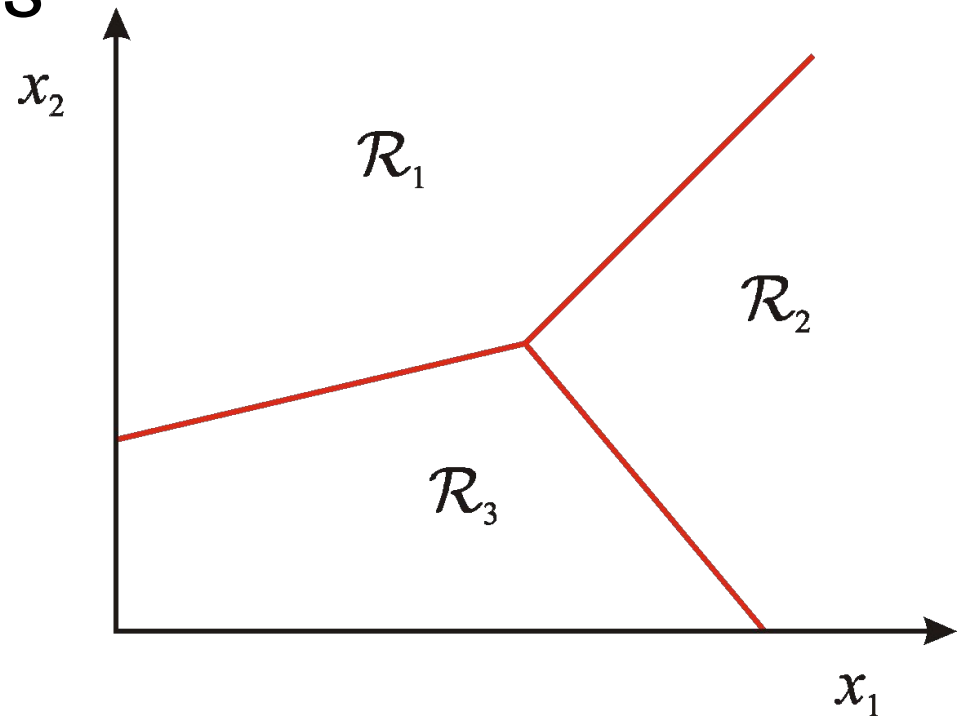
- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

- Often easier to predict a label from the data than to model the data

Slide credit: D. Hoiem

# Classification

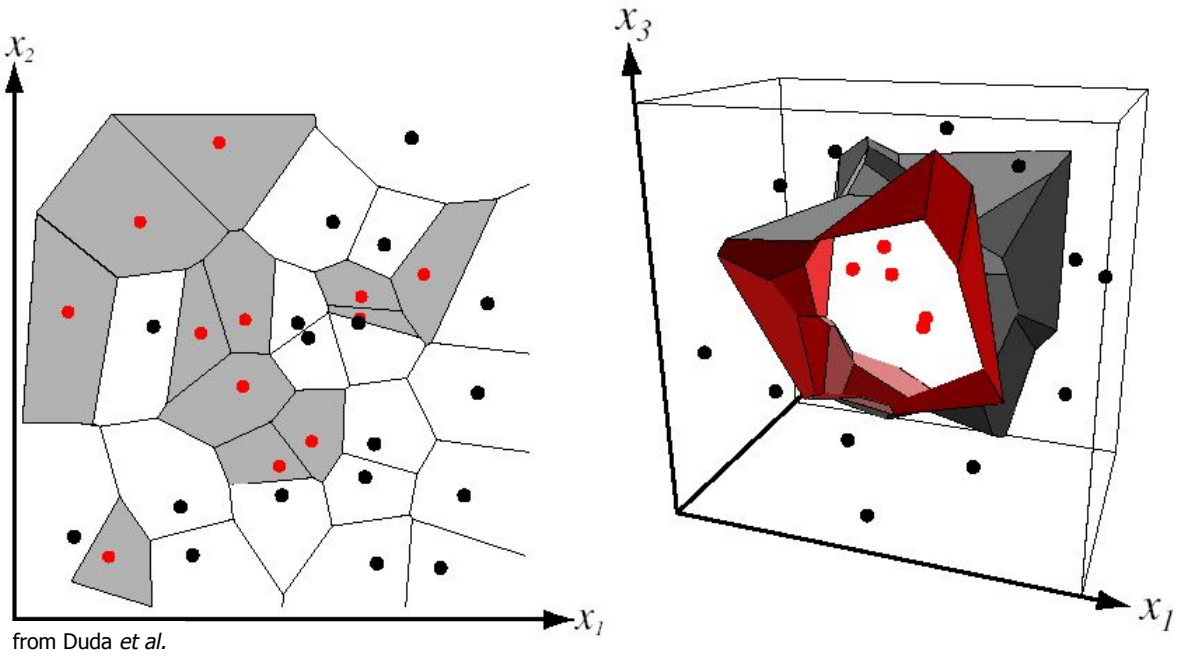
- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*




Slide credit: L. Lazebnik

# Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point

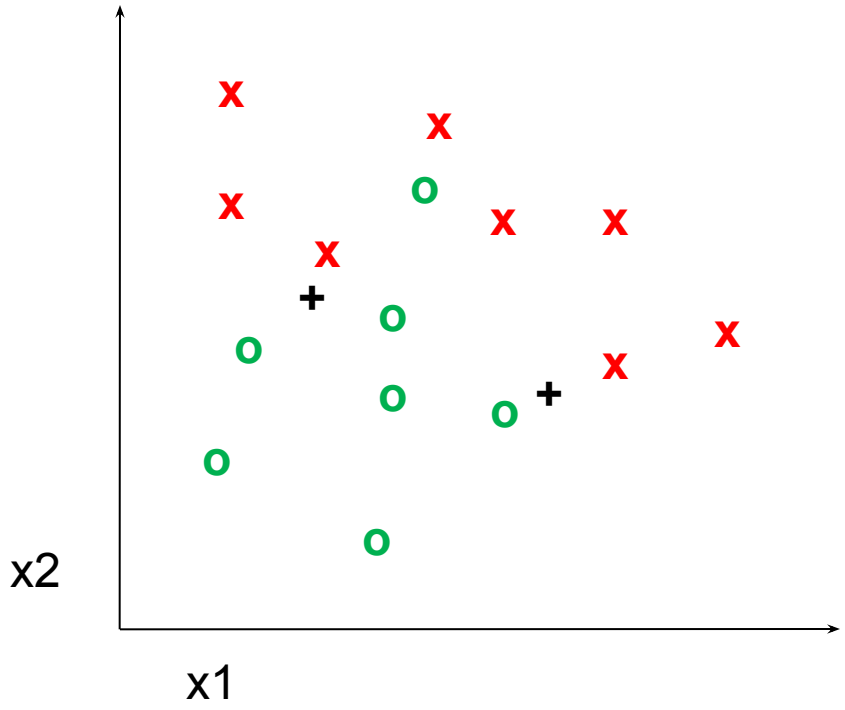



from Duda *et al.*

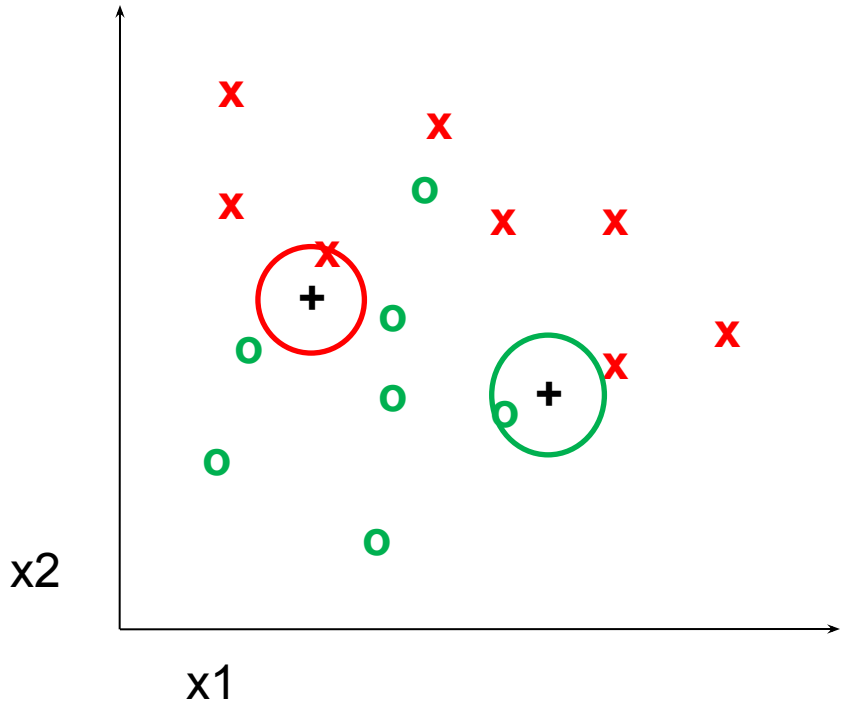
Voronoi partitioning of feature space
for two-category 2D and 3D data

Source: D. Lowe

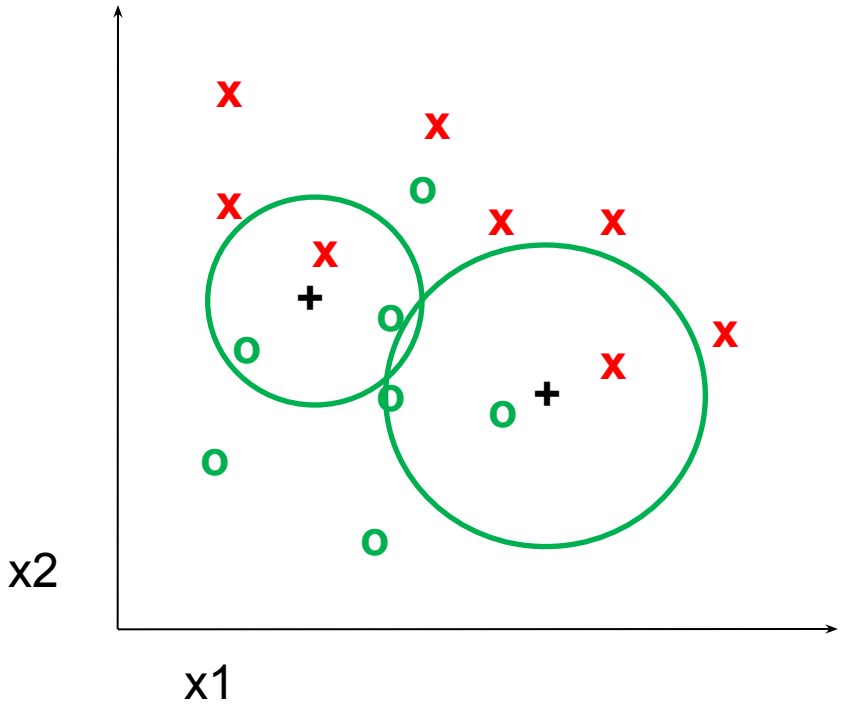
# K-nearest neighbor



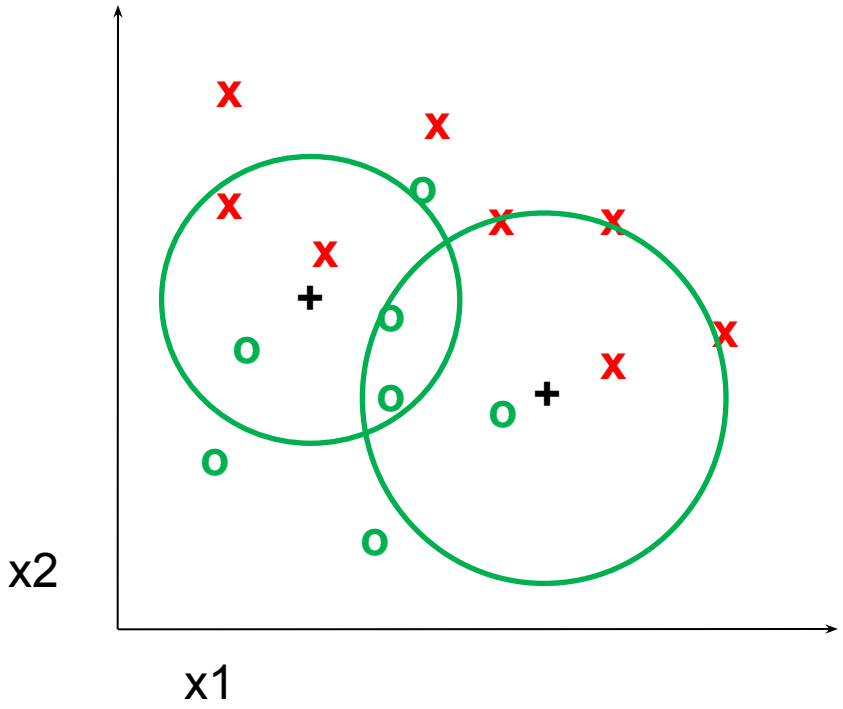
# 1-nearest neighbor



# 3-nearest neighbor



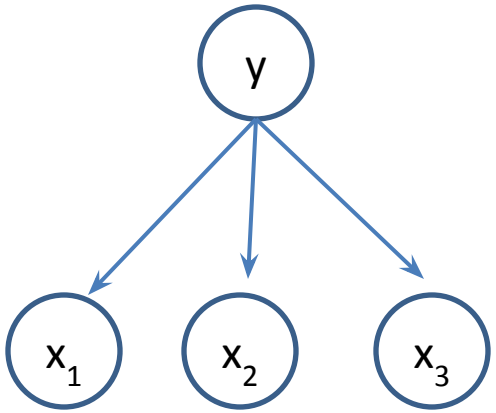
# 5-nearest neighbor



# Using K-NN

- Simple to implement and interpret, a good classifier to try first

# Naïve Bayes

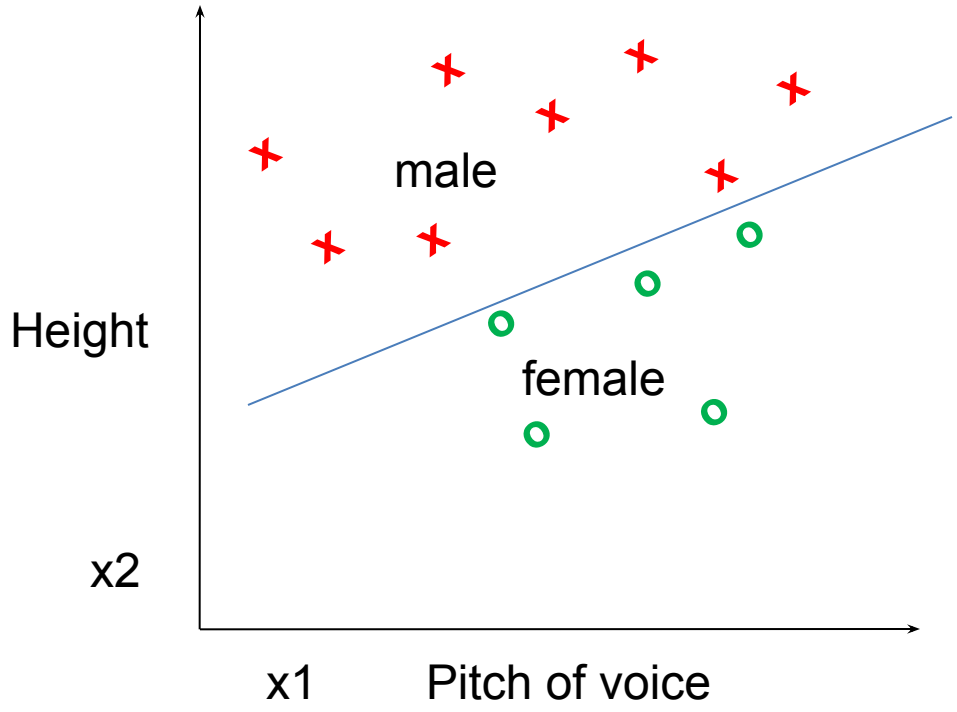


# Using Naïve Bayes

- Simple thing to try for categorical data
- Very fast to train/test

# Classifiers: Logistic Regression

Maximize likelihood of label given data, assuming a log-linear model



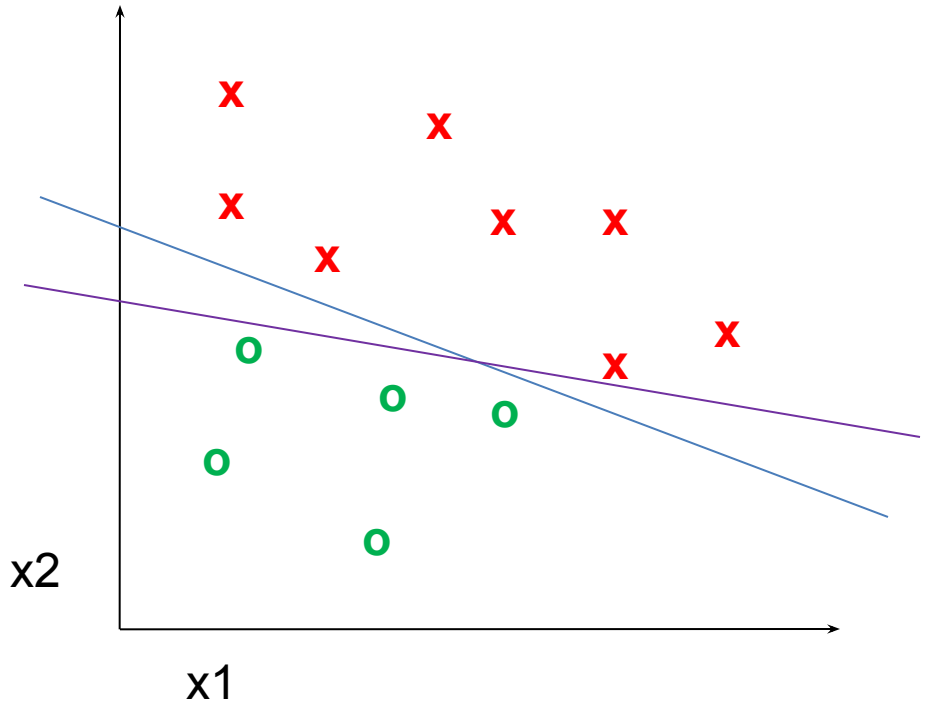

$$\log \frac{P(x_1, x_2 \mid y = 1)}{P(x_1, x_2 \mid y = -1)} = \mathbf{w}^T \mathbf{x}$$

$$P(y = 1 \mid x_1, x_2) = 1 / \left(1 + \exp\left(-\mathbf{w}^T \mathbf{x}\right)\right)$$

# Using Logistic Regression

- Quick, simple classifier (try it first)
- Outputs a probabilistic label confidence
- Use L2 or L1 regularization
  - L1 does feature selection and is robust to irrelevant features but slower to train

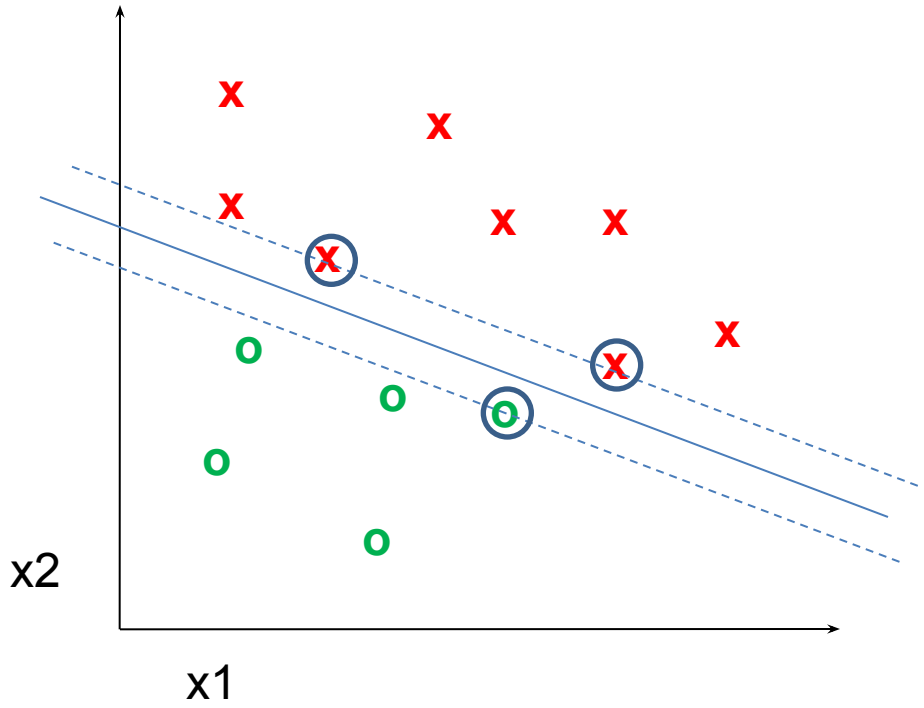
# Classifiers: Linear SVM




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

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

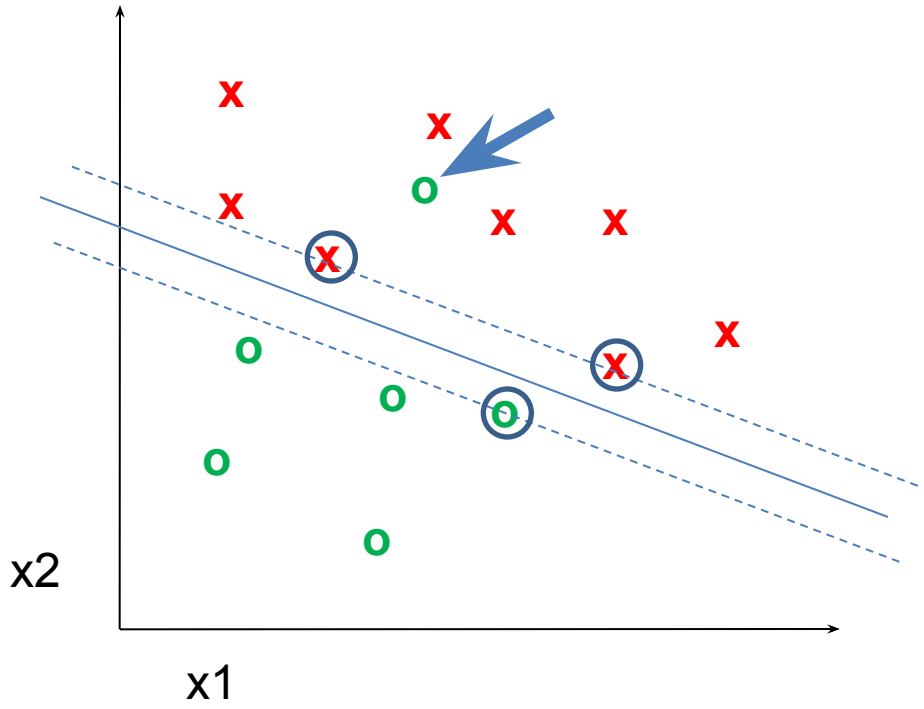
# Classifiers: Linear SVM




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

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

# Classifiers: Linear SVM

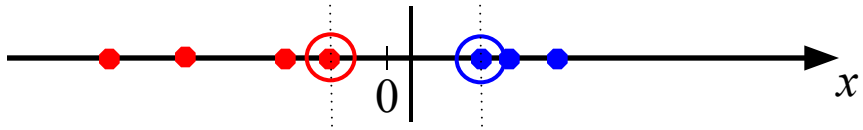



- Find a *linear function* to separate the classes:

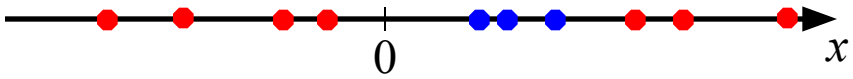
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Nonlinear SVMs

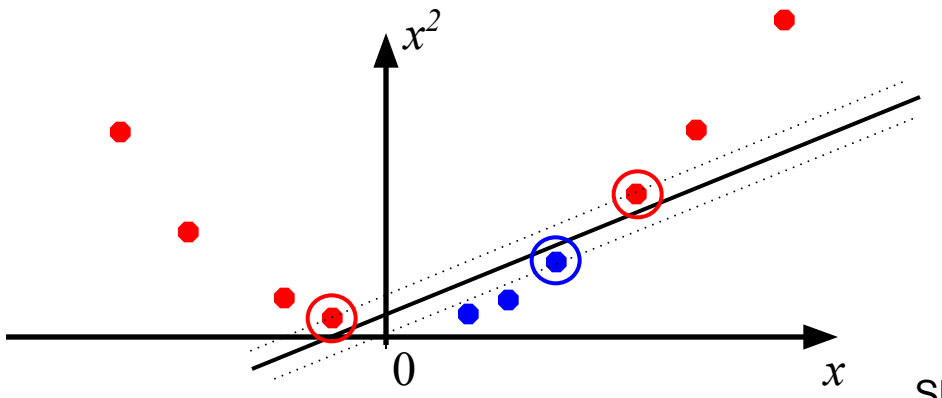
- Datasets that are linearly separable work out great:




- But what if the dataset is just too hard?



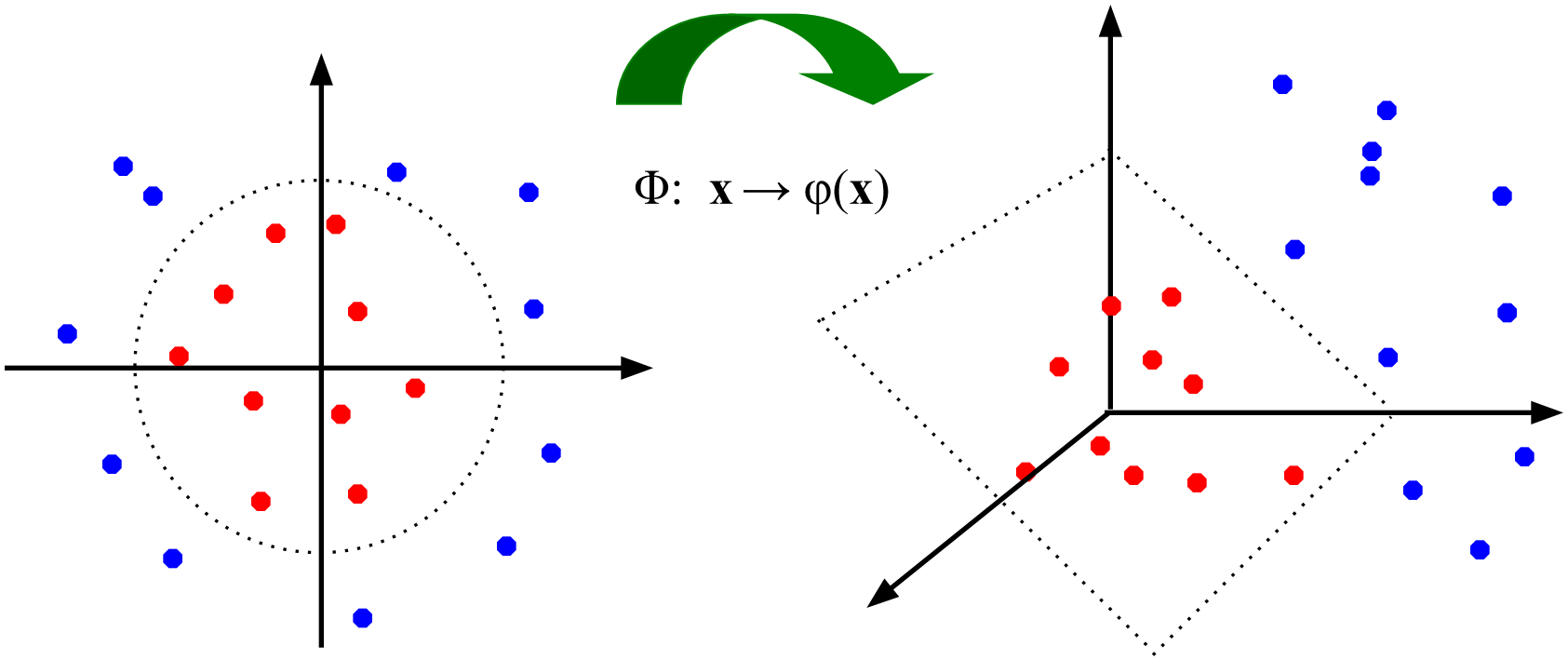

- We can map it to a higher-dimensional space:




Slide credit: Andrew Moore

# Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:




Slide credit: Andrew Moore

# Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

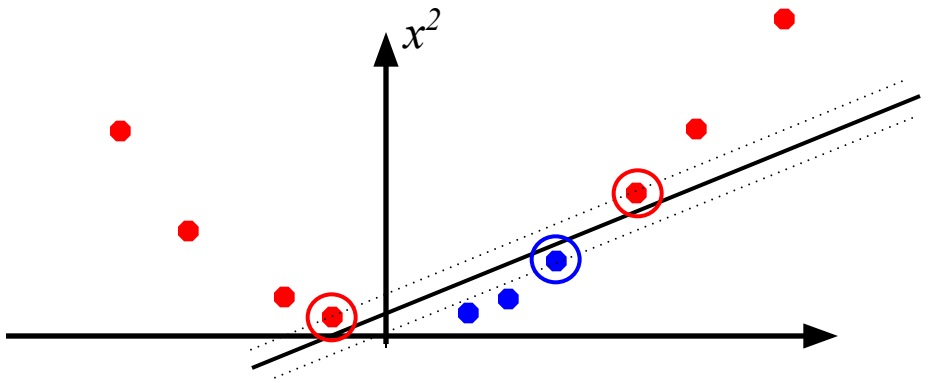
  (to be valid, the kernel function must satisfy *Mercer's condition*)
- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}) + b = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

# Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$




$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

# State of the art in 2007: Kernels for bags of features

- Histogram intersection kernel:

$$I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Generalized Gaussian kernel:

$$K(h_1, h_2) = \exp\left( -\frac{1}{A} D(h_1, h_2)^2 \right)$$

- *D* can be (inverse) L1 distance, Euclidean distance, $\chi^2$ distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study, IJCV 2007

# Summary: SVMs for image classification

1. Pick an image representation (e.g. histogram of quantized sift features)
2. Pick a kernel function for that representation
3. Compute the matrix of kernel values between every pair of training examples
4. Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights
5. At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

Slide credit: L. Lazebnik

# What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
  - Traning: 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

Slide credit: L. Lazebnik

# SVMs: Pros and cons

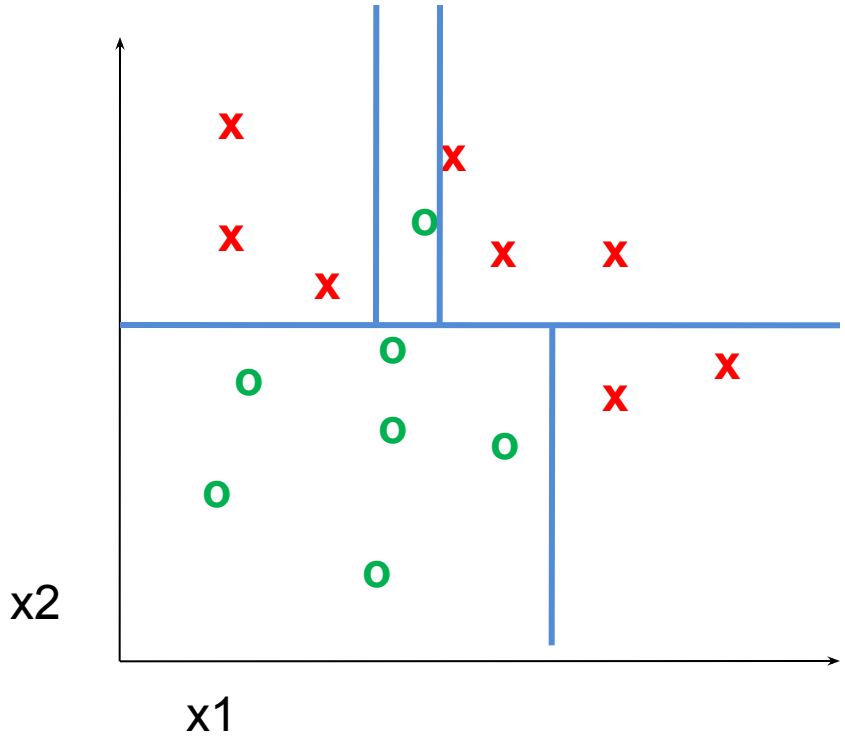
- Pros
  - Linear SVMs are surprisingly accurate, while being lightweight and interpretable
  - Non-linear, kernel-based SVMs are very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes
- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples. Quadratic memory consumption.
    - Learning can take a very long time for large-scale problems

# Summary: Classifiers

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, $\chi^2$ distance, quadratic distance, histogram intersection
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Kernel functions: histogram intersection, generalized Gaussian, pyramid match
  - Multi-class
- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees, …

Slide credit: L. Lazebnik

# Classifiers: Decision Trees



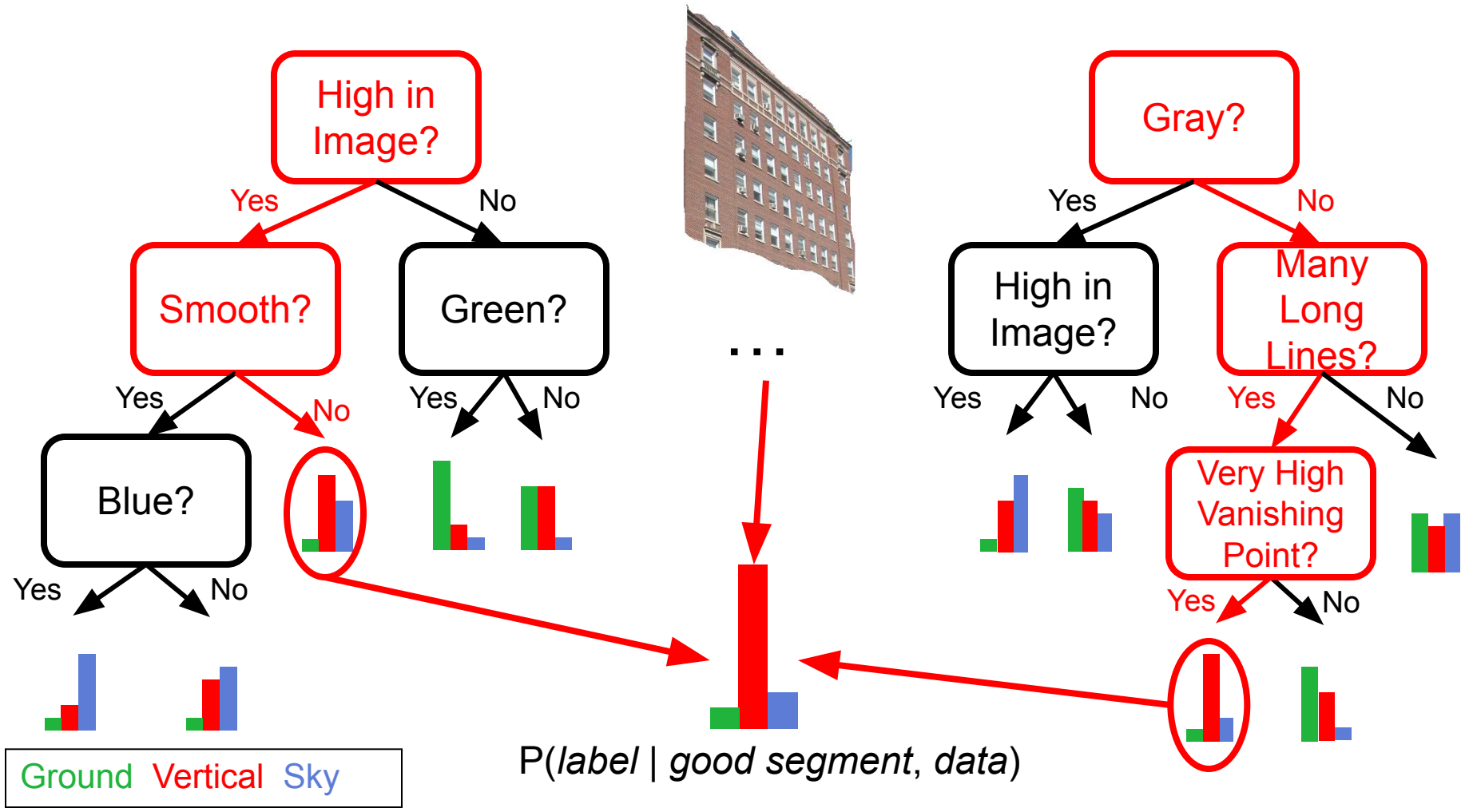
# Ensemble Methods: Boosting

**Discrete AdaBoost(Freund & Schapire 1996*b*)**

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

figure from Friedman et al. 2000

# Boosted Decision Trees




[Collins et al. 2002]

# Using Boosted Decision Trees

- Flexible: can deal with both continuous and categorical variables
- How to control bias/variance trade-off
  - Size of trees
  - Number of trees
- Boosting trees often works best with a small number of well-designed features
- Boosting “stubs” can give a fast classifier

# Ideals for a classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: takes advantage of the structure of the problem
- Regularization: good priors on the parameters
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for labels that maximize objective function for a test example

# Two ways to think about classifiers

1. What is the objective? What are the parameters? How are the parameters learned? How is the learning regularized? How is inference performed?

2. How is the data modeled? How is similarity defined? What is the shape of the boundary?

Slide credit: D. Hoiem

# Comparison

assuming x in {0 1}

| | Learning Objective | Training | Inference |
|---|---|---|---|
| Naïve Bayes | $\text{maximize} \sum_i \left[ \sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$ | $\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$ | $\theta_1^T \mathbf{x} + \theta_0^T (1 - \mathbf{x}) > 0$ <br> where $\theta_{1j} = \log \frac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$, <br> $\theta_{0j} = \log \frac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}$ |
| Logistic Regression | $\text{maximize} \sum_i \log(P(y_i \mid \mathbf{x}, \boldsymbol{\theta})) + \lambda \lVert \boldsymbol{\theta} \rVert$ <br> where $P(y_i \mid \mathbf{x}, \boldsymbol{\theta}) = 1 / (1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}))$ | Gradient ascent | $\boldsymbol{\theta}^T \mathbf{x} > 0$ |
| Linear SVM | $\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \lVert \boldsymbol{\theta} \rVert$ <br> such that $y_i \boldsymbol{\theta}^T \mathbf{x} \geq 1 - \xi_i \quad \forall i$ | Linear programming | $\boldsymbol{\theta}^T \mathbf{x} > 0$ |
| Kernelized SVM | complicated to write | Quadratic programming | $\sum_i y_i \alpha_i K(\hat{\mathbf{x}}_i, \mathbf{x}) > 0$ |
| Nearest Neighbor | most similar features □ same label | Record data | $y_i$ <br> where $i = \underset{i}{\text{argmin}} \; K(\hat{\mathbf{x}}_i, \mathbf{x})$ |

Slide credit: D. Hoiem

# Very brief tour of some classifiers

- **K-nearest neighbor**
- **SVM**
- Boosted Decision Trees
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- RBMs
- Deep Convolutional Network
- Attentional models or “Transformers”
- Etc.

# 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?

Slide credit: L. Lazebnik

# Generalization

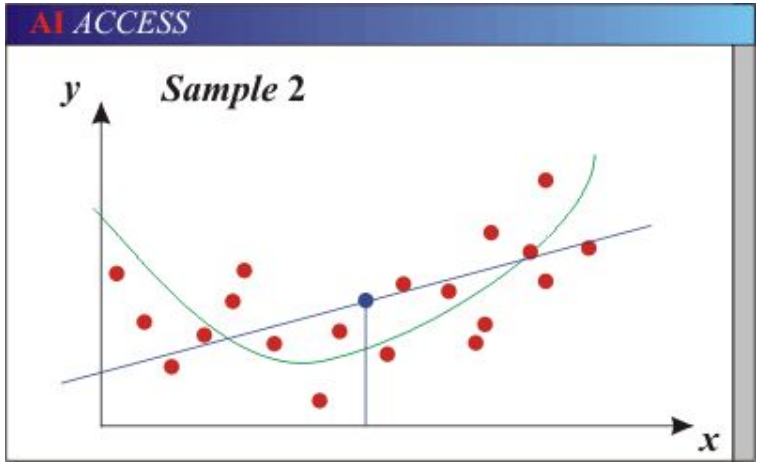
- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model. “Bias” sounds negative. “Regularization” sounds nicer.
  - **Variance:** how much models estimated from different training sets differ from each other. Typical of more “expressive” models.
- **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
- **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

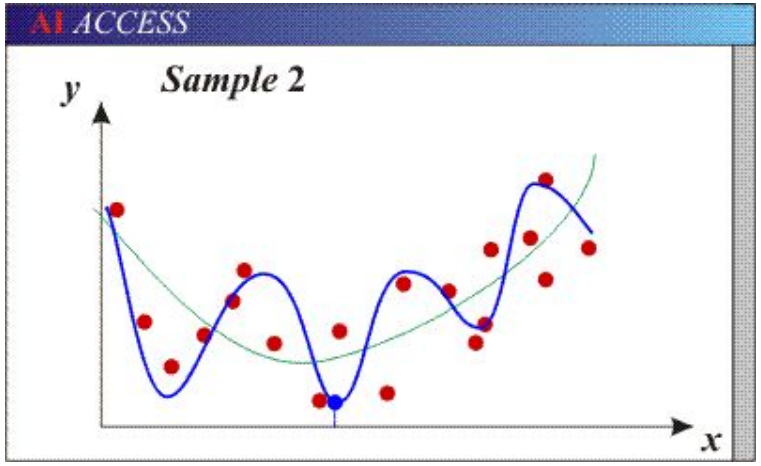
Slide credit: L. Lazebnik

# No Free Lunch Theorem


Slide credit: D. Hoiem

# Bias-Variance Trade-off




- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Slide credit: D. Hoiem

# Bias-Variance Trade-off

$$E(MSE) = noise^2 + bias^2 + variance$$

Unavoidable error

Error due to incorrect assumptions

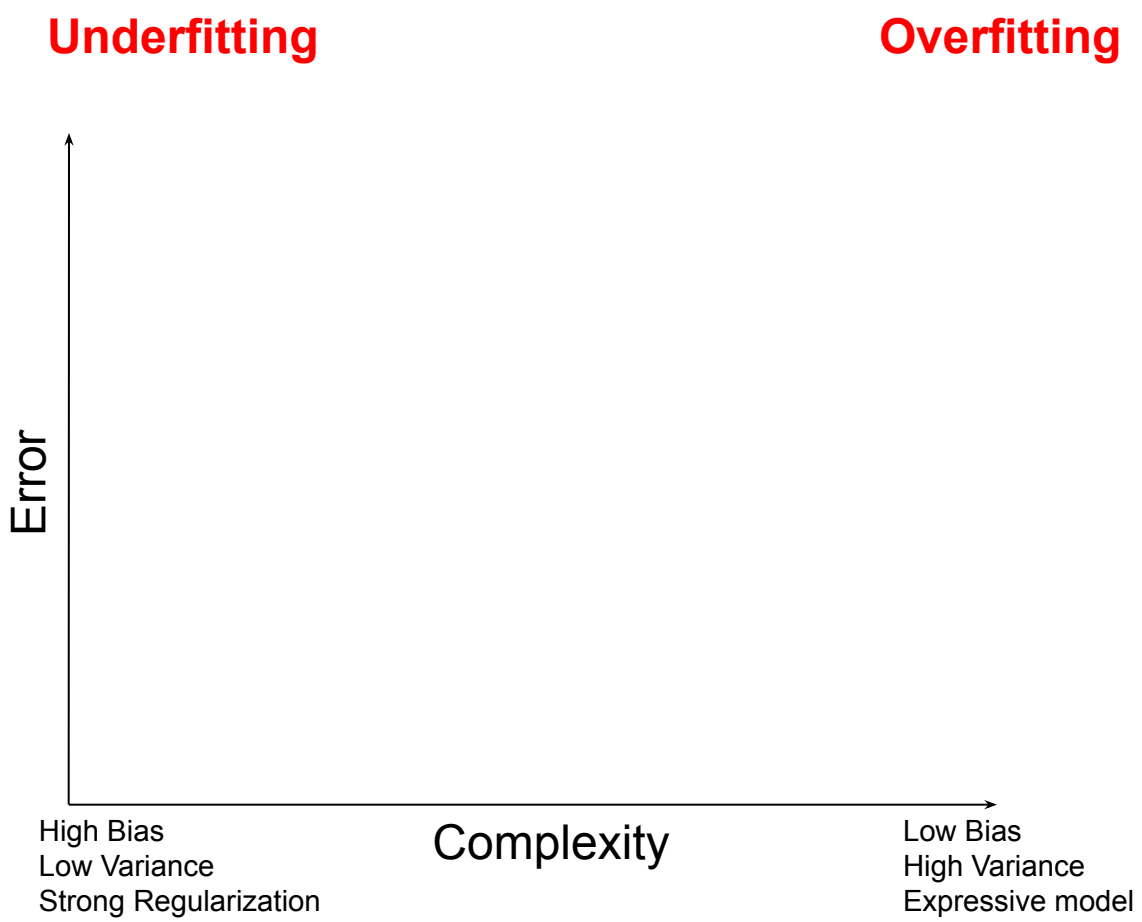
Error due to variance of training samples

See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

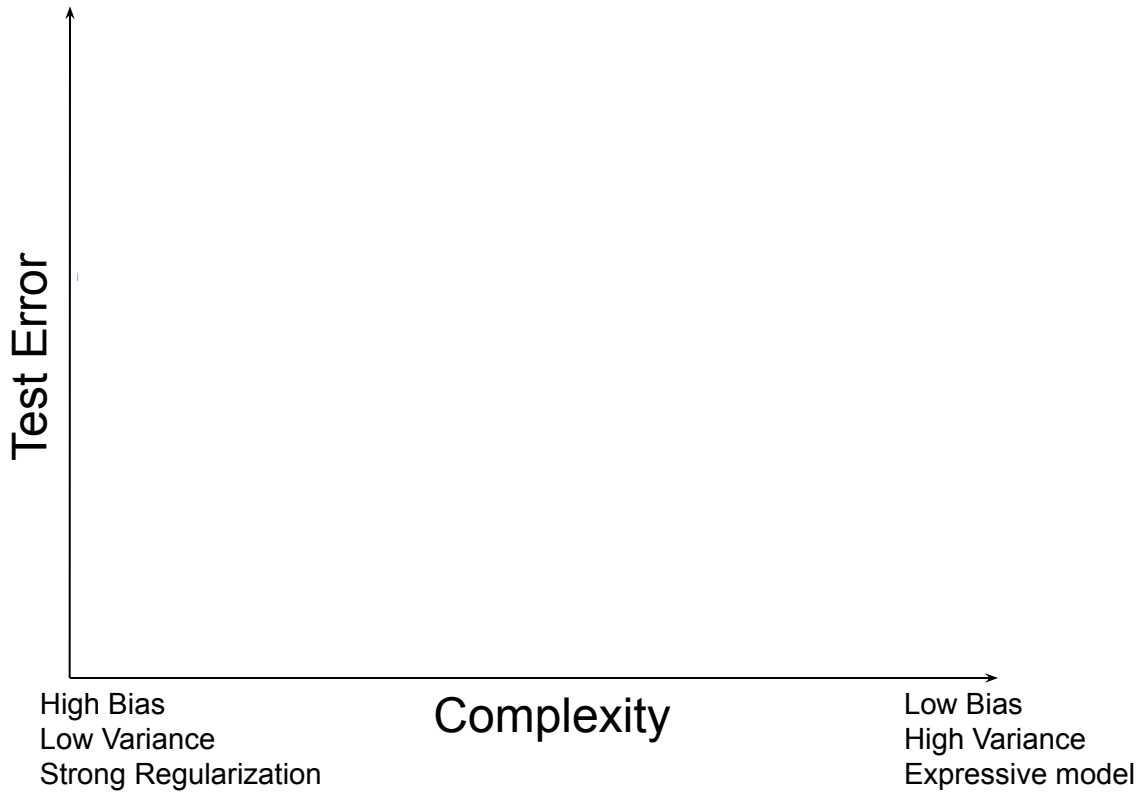
Slide credit: D. Hoiem

# Bias-variance tradeoff



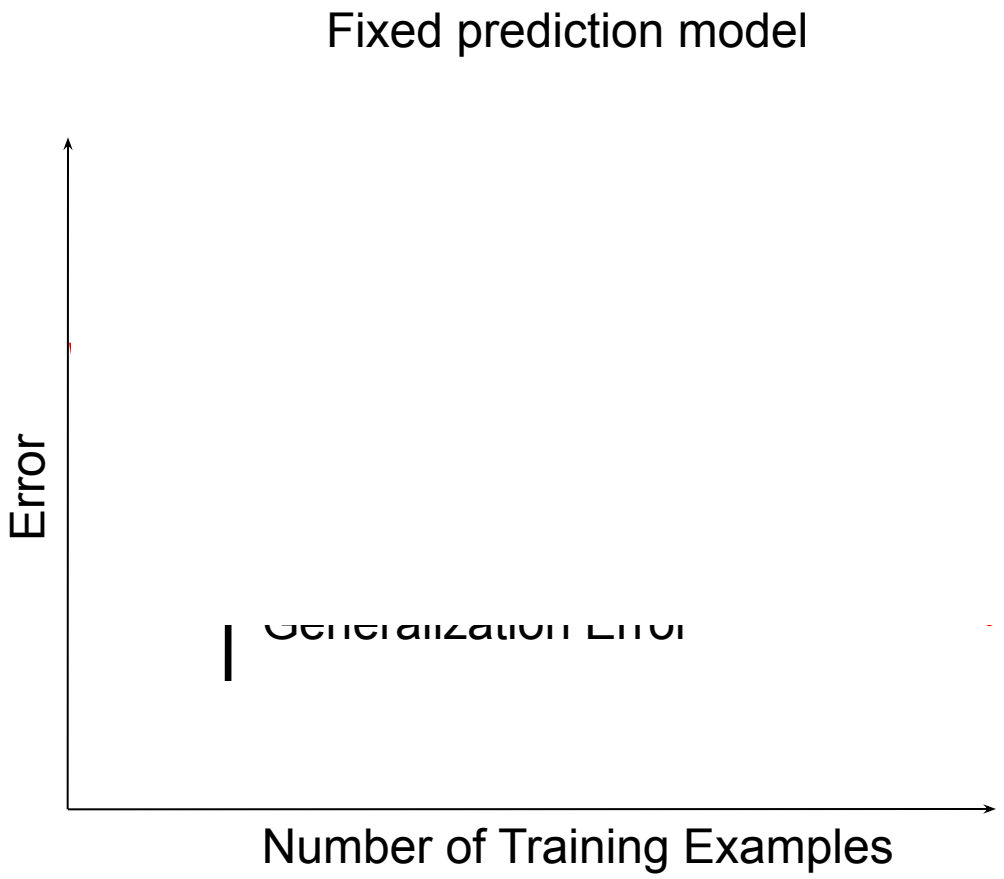

Slide credit: D. Hoiem

# Bias-variance tradeoff




Slide credit: D. Hoiem

# Effect of Training Size




Slide credit: D. Hoiem

# The perfect classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: makes assumptions that fit the problem
- Regularization: right level of regularization for amount of training data
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for objective function in evaluation

Slide credit: D. Hoiem

# Remember…

- No classifier is inherently better than any other: you need to make assumptions to generalize

- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications / regularization
  - Variance: due to inability to perfectly estimate parameters from limited data


Slide credit: D. Hoiem

- How to reduce variance?
  - Choose a simpler classifier
  - Regularize the parameters
  - Get more training data
- How to reduce bias?
  - Choose a more complex, more expressive classifier
  - Remove regularization
  - (These might not be safe to do unless you get more training data)

Slide credit: D. Hoiem

# What to remember about classifiers

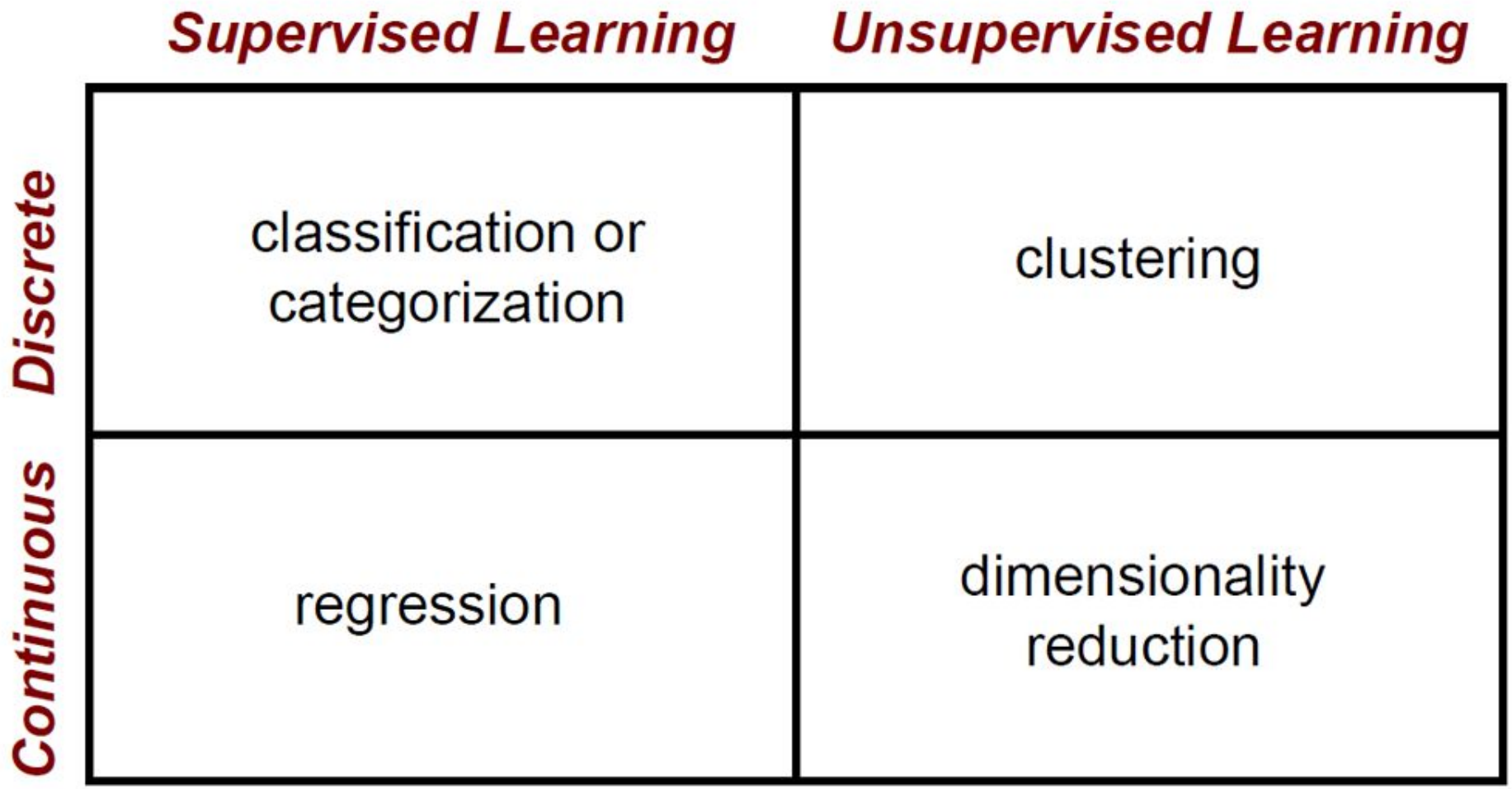
- No free lunch: machine learning algorithms are tools, not dogmas
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Slide credit: D. Hoiem

# Machine Learning Considerations

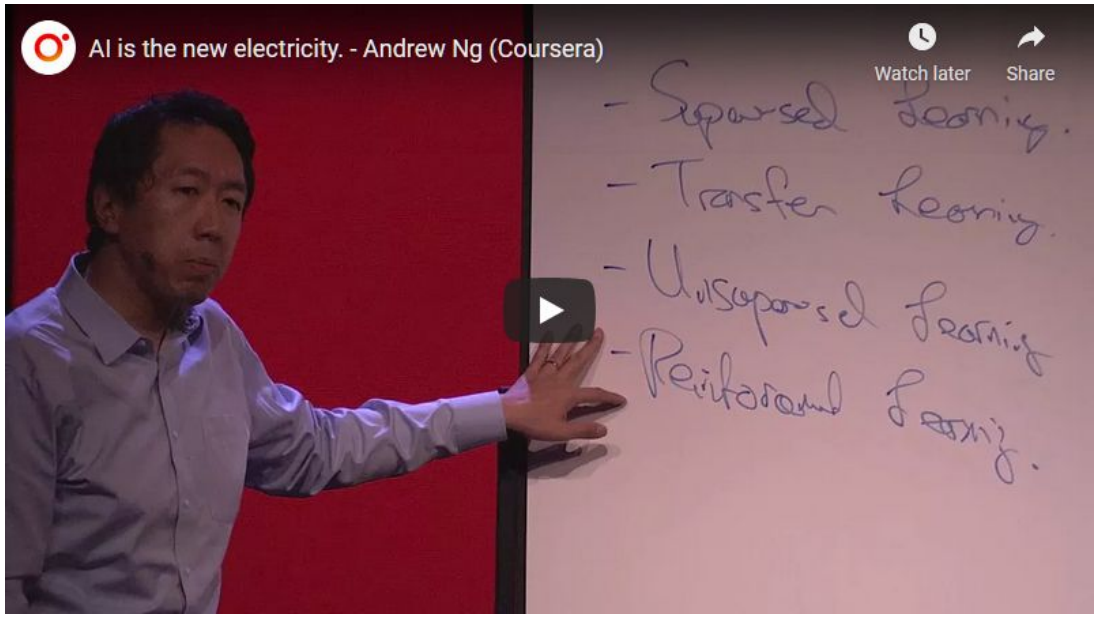
- 3 important design decisions:
  1) What data do I use?
  2) How do I represent my data (what feature)?
  3) What classifier / regressor / machine learning tool do I use?
- These are in decreasing order of importance
- Deep learning addresses 2 and 3 simultaneously (and blurs the boundary between them).
- You can take the representation from deep learning and use it with any classifier.

# Machine Learning Problems



- Andrew Ng's ranking of machine learning impact
  1. Supervised Learning
  2. Transfer Learning
  3. Unsupervised Learning (I prefer "self-supervised" learning)
  4. Reinforcement Learning

James thinks 2 and 3 might have switched ranks.



# Usage in recent computer vision papers

- “PCA” 3,610
- “K-means” 2,950
- “ResNet” 14,900
- “ViT” 5,540
- “Reinforcement learning” 3,320
- “Self-supervised” 11,300
- “Unsupervised” 18,400

site:https://openaccess.thecvf.com “search term” seems to search ICCV, CVPR, and WACV papers

# Some Machine Learning References

- General
  - Tom Mitchell, *Machine Learning*, McGraw Hill, 1997
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
- Adaboost
  - Friedman, Hastie, and Tibshirani, “Additive logistic regression: a statistical view of boosting”, Annals of Statistics, 2000
- SVMs
  - http://www.support-vector.net/icml-tutorial.pdf

Slide credit: D. Hoiem