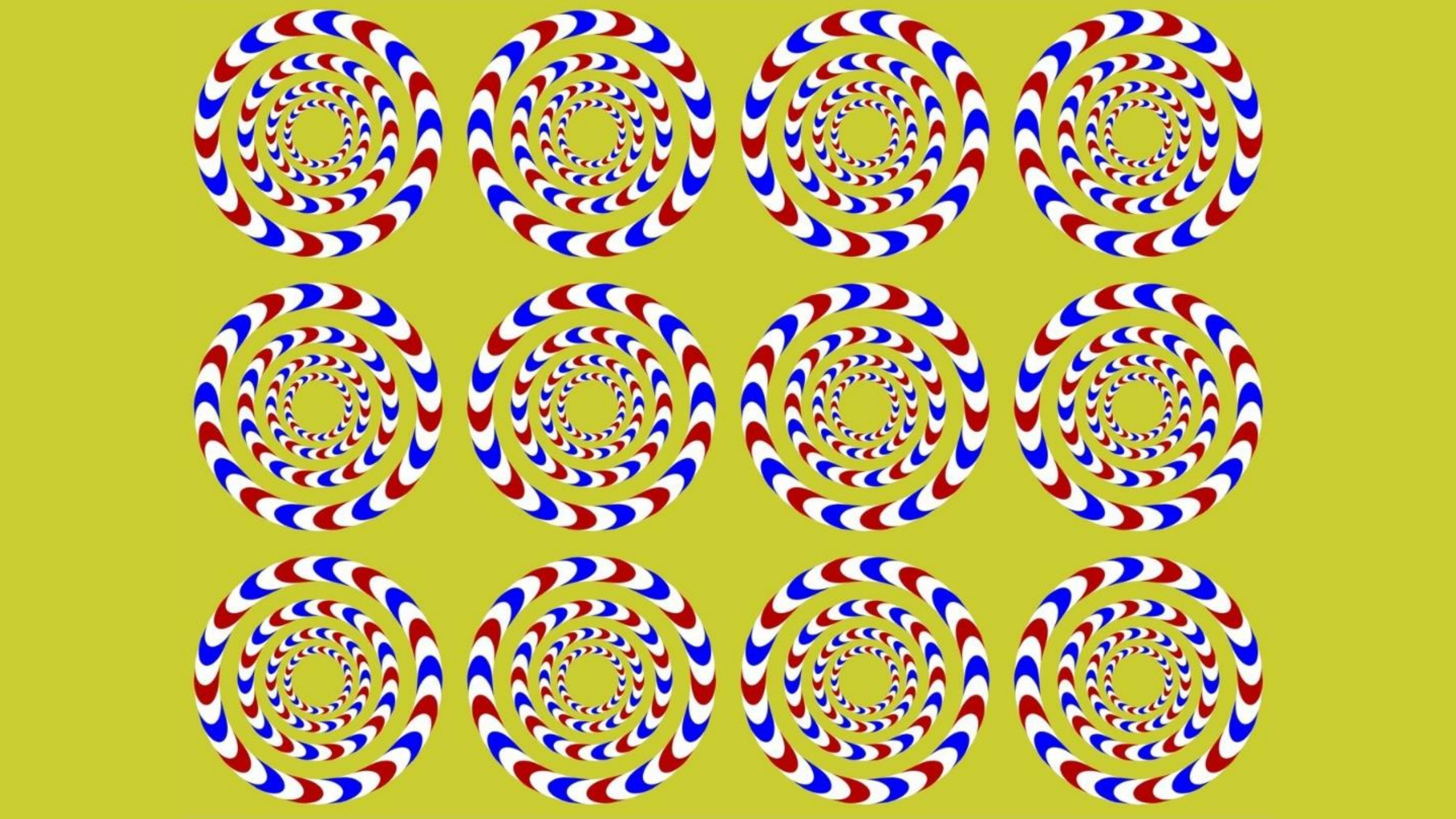


Machine Learning Crash Course

Photo: CMU Machine Learning Department protests G20

Computer Vision
James Hays

Slides: Isabelle Guyon, Erik Sudderth, Mark Johnson, Derek Hoiem

Machine Learning Problems

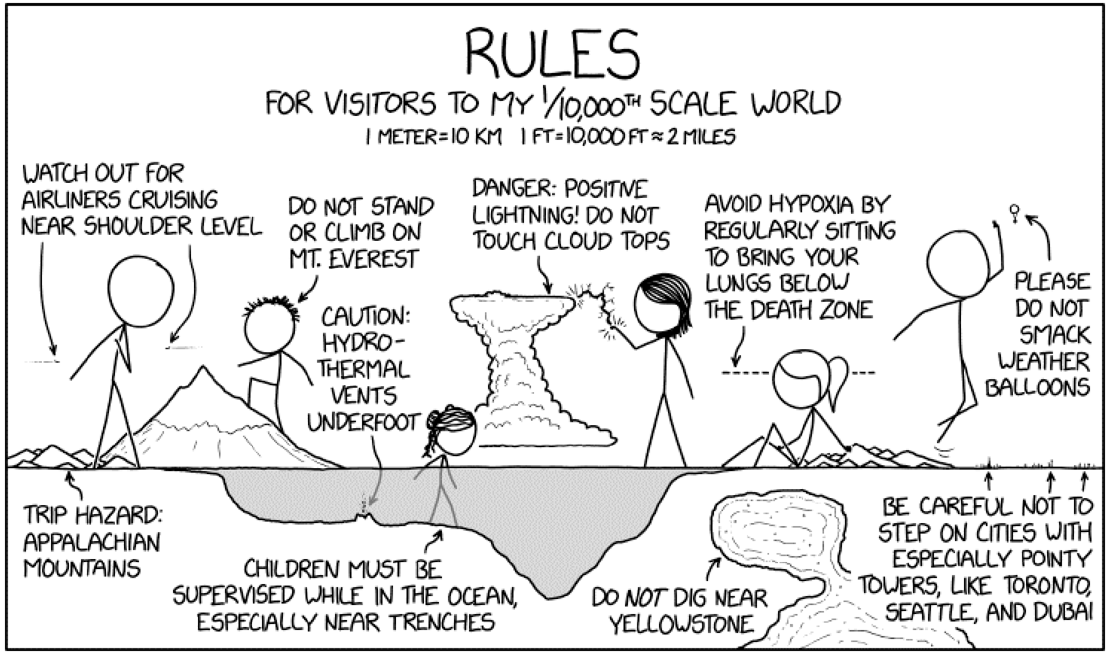
	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Dimensionality Reduction

Simplest dimensionality reduction: drop a dimension

Credit: xkcd

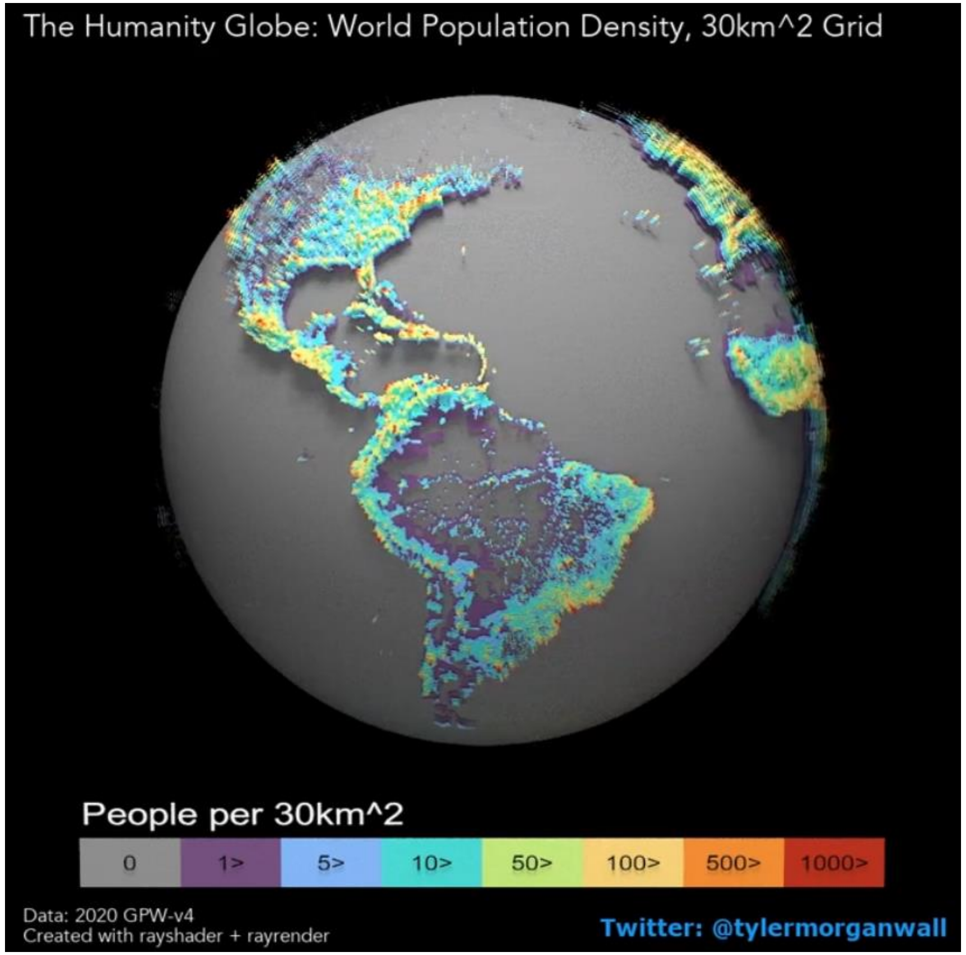
Dimensionality Reduction



The Earth is pretty smooth

Credit: xkcd

Non-linear Dimensionality Reduction



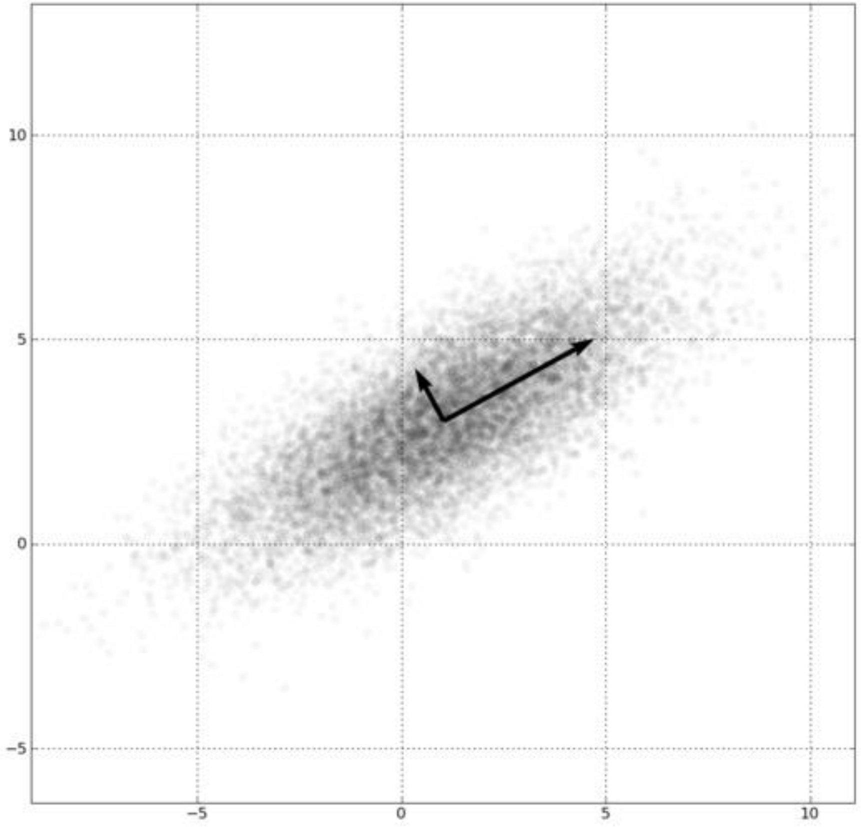
Credit: xkcd

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 Ψ.

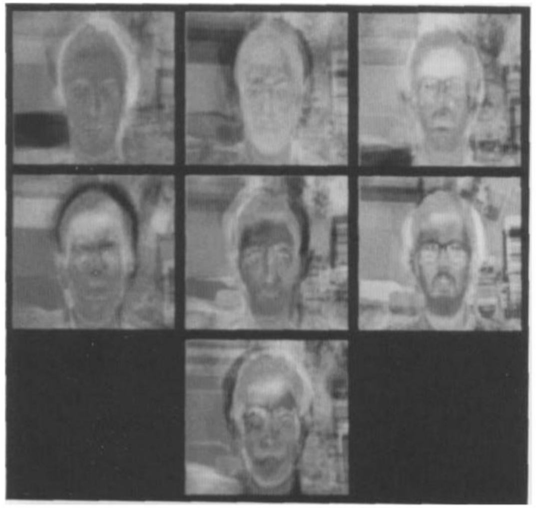
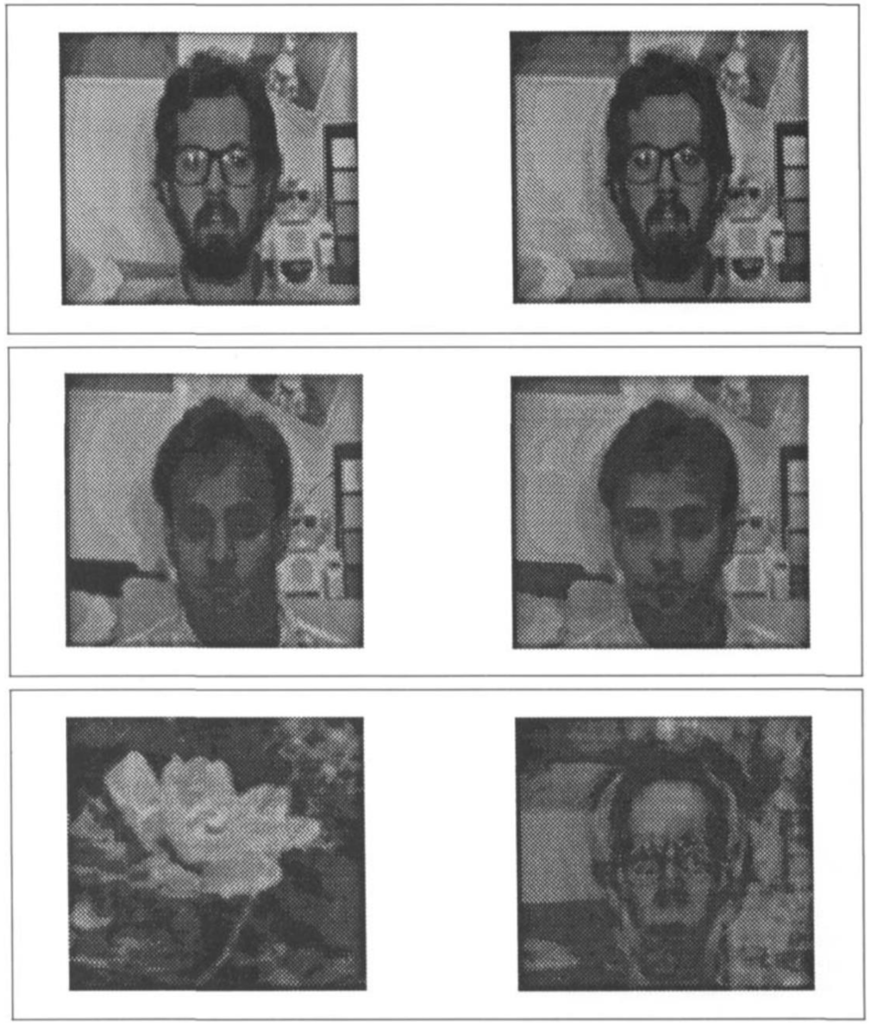


Figure 2. Seven of the eigenfaces calculated from the input images of Figure 1.

The “Eigenfaces”



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

PCA as a data interpretation tool

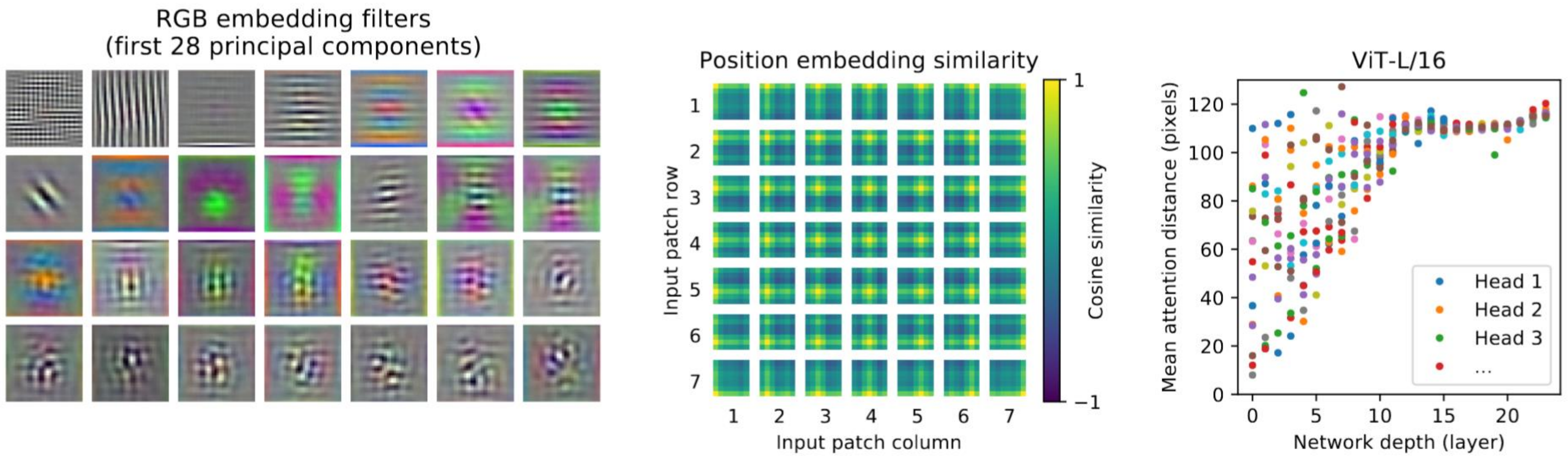
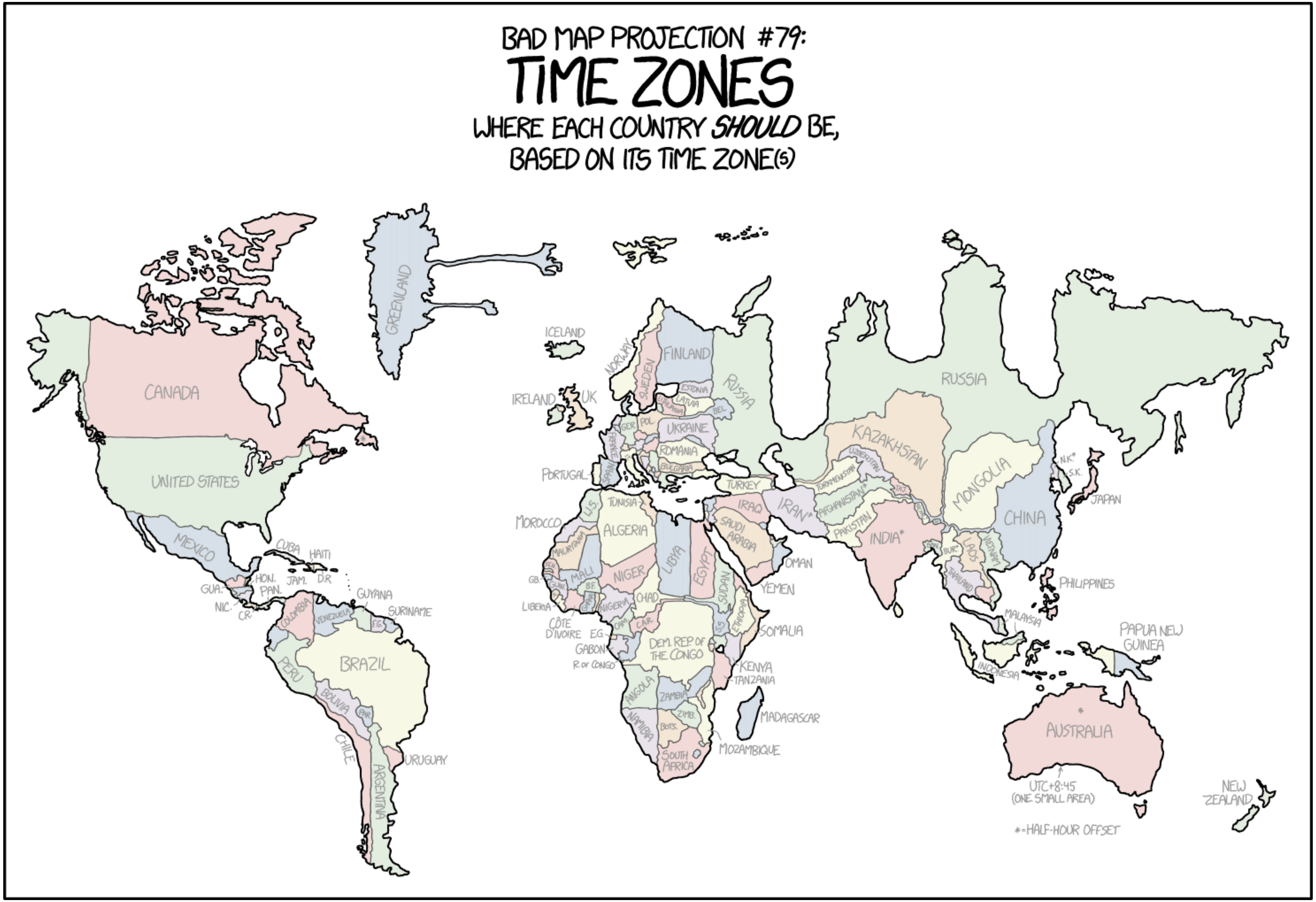


Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **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

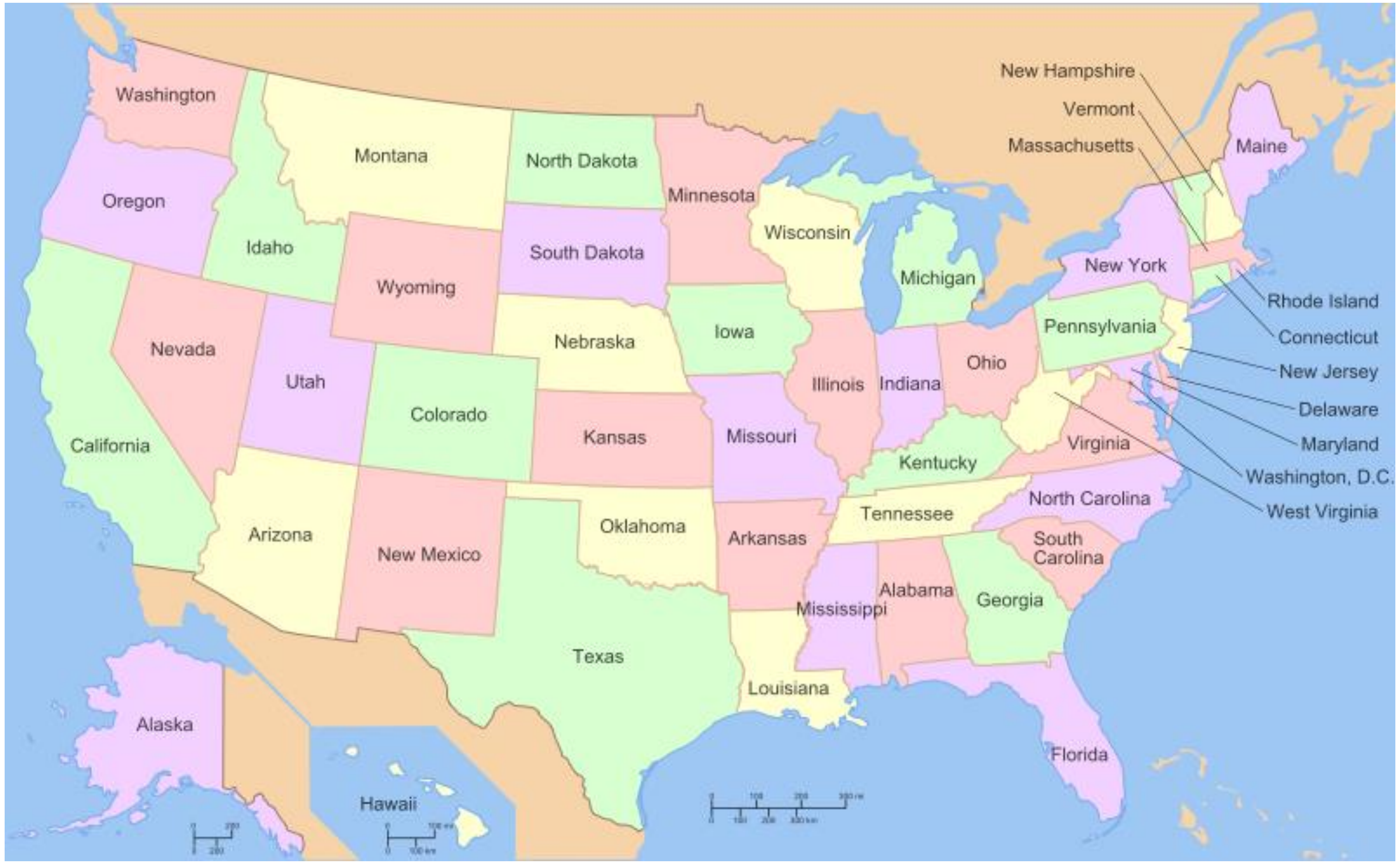
Machine Learning Problems

	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	$\boxed{\text{clustering}}$
Continuous	regression	dimensionality reduction

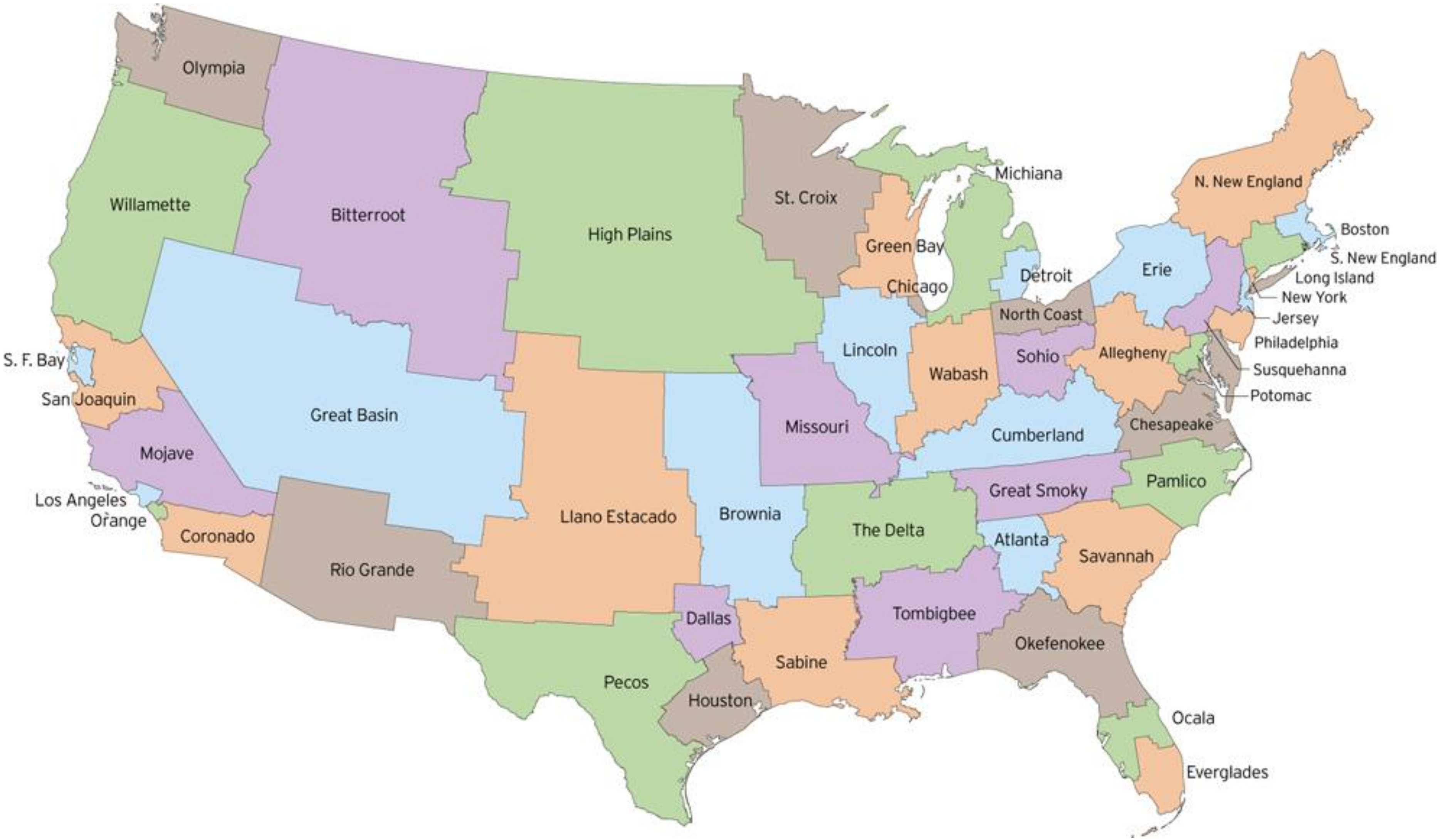


BAD MAP PROJECTION #79:
TIME ZONES
WHERE EACH COUNTRY SHOULD BE, BASED ON ITS TIME ZONE(S)
GREENLAND
ICELAND
NORWAY
SWEDEN
FINLAND
ESTONIA
LATVIA
BEL
RUSSIA
RUSSIA
IRELAND
UK
GER
POL.
UKRAINE
FRANCE
ROMANIA
BULGARIA
PORTUGAL
SPAIN
TURKEY
CANADA
UNITED STATES
MEXICO
CUBA
HAITI
HON.
PAN.
JAM.
D.R.
GUA.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

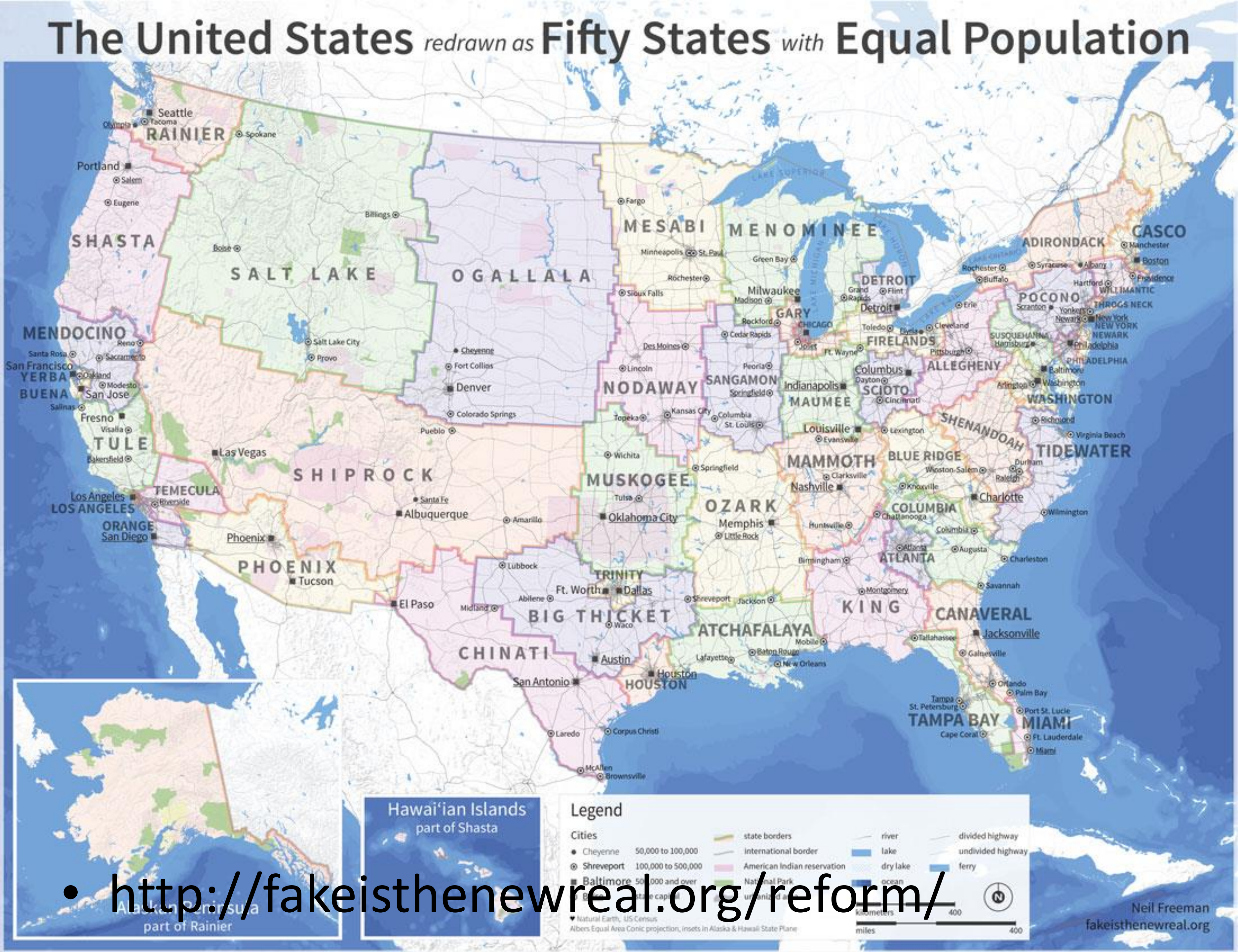
Credit: xkcd



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
Pennsylvania
New York
Maine
New Hampshire
Vermont
Massachusetts
Rhode Island
Connecticut
New Jersey
Delaware
Maryland
Washington, D.C.
West Virginia
Alaska
Hawaii



- http://fakeisthenewreal.org/reform/



- http://fakeisthenewreal.org/reform/

Clustering example: image segmentation

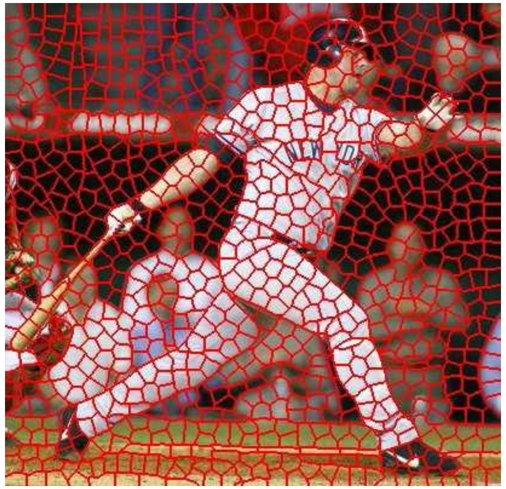
Goal: Break up the image into meaningful or perceptually similar regions

Segmentation for feature support or efficiency

50x50
Patch

50x50
Patch

[Felzenszwalb and Huttenlocher 2004]

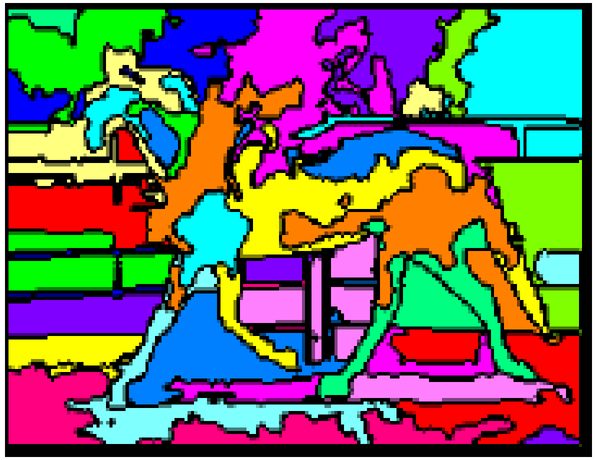


[Shi and Malik 2001]

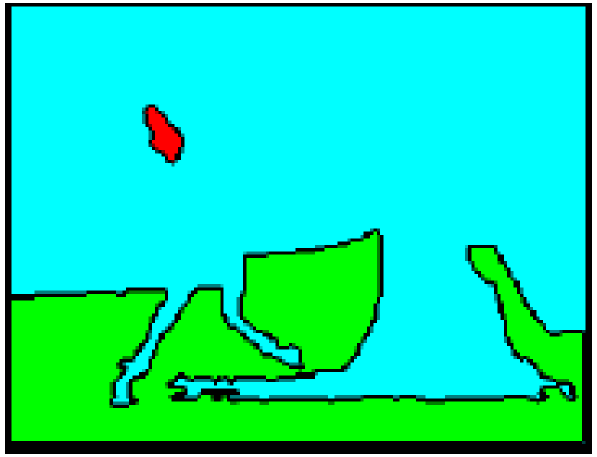
[Hoiem et al. 2005, Mori 2005]

Slide: Derek Hoiem

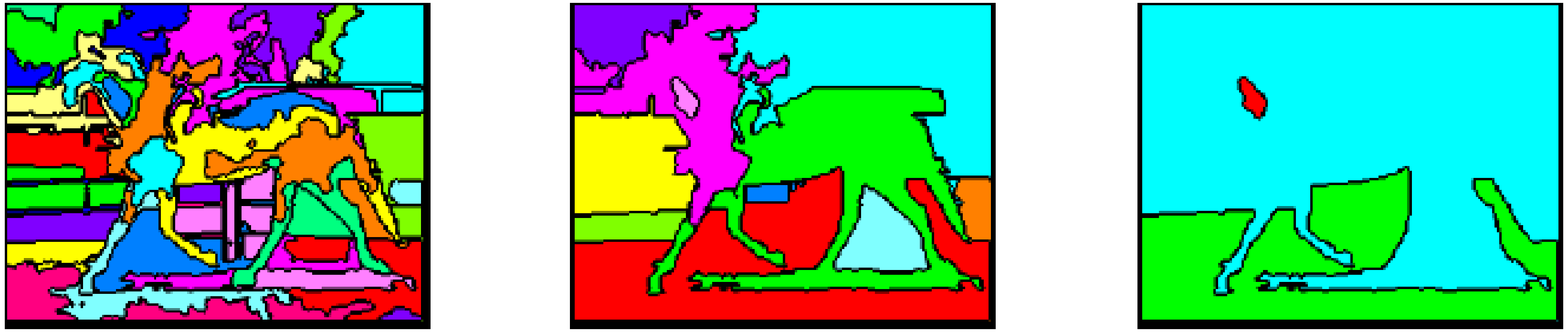
Types of segmentations



Oversegmentation



Undersegmentation



Multiple Segmentations

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

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

Cluster center

Data

$$\mathbf{c}^*, \boldsymbol{\delta}^* = \underset{\mathbf{c},\boldsymbol{\delta}}{\operatorname{argmin}} \tfrac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij} \left(\mathbf{c}_i - \mathbf{x}_j\right)^2$$

Whether x_j is assigned to c_i

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

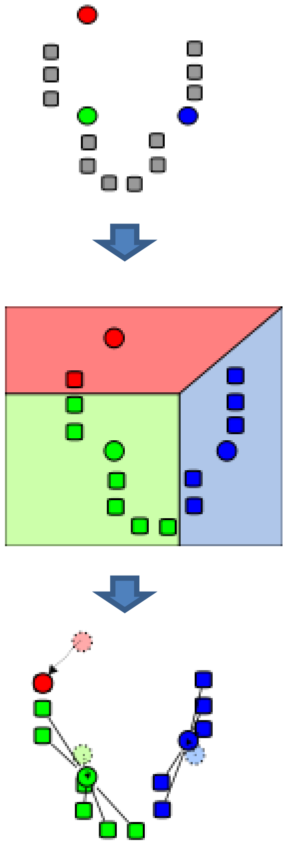


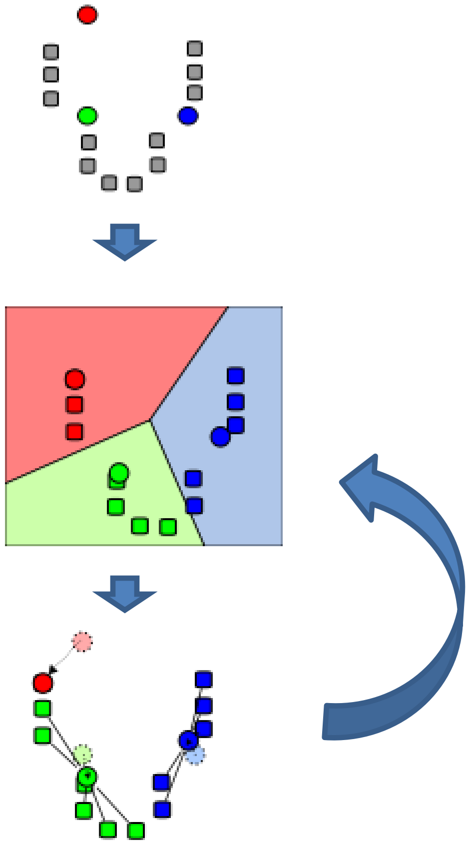
Illustration: http://en.wikipedia.org/wiki/K-means_clustering

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

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

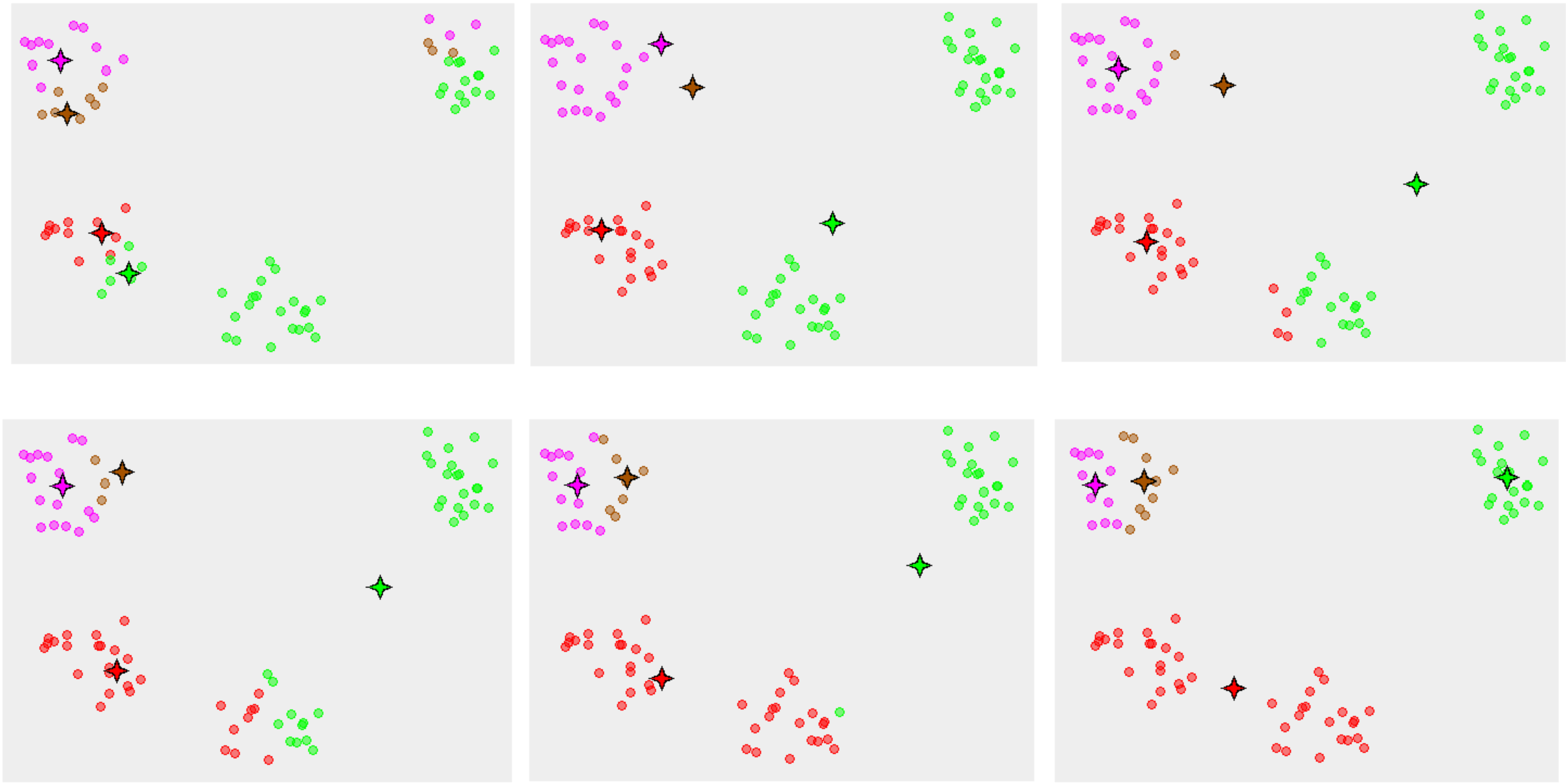
3. Update cluster centers as the mean of the points

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

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

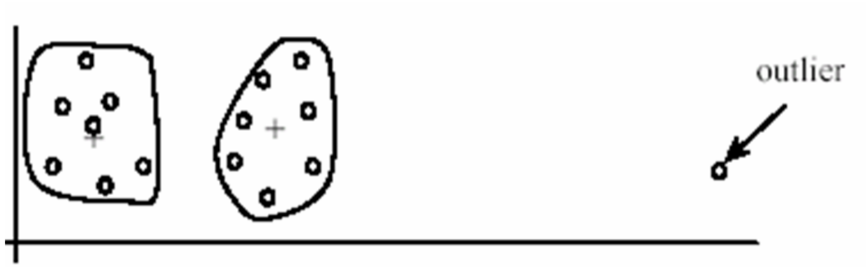
Image

Clusters on intensity

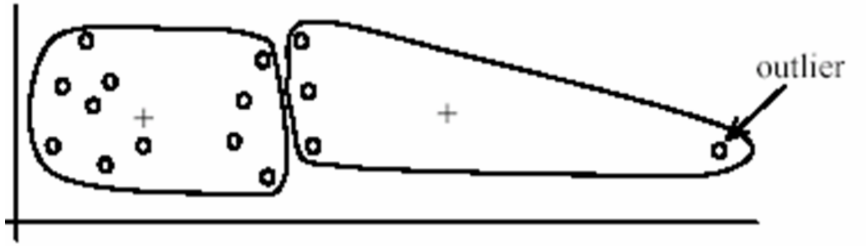
Clusters on color

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



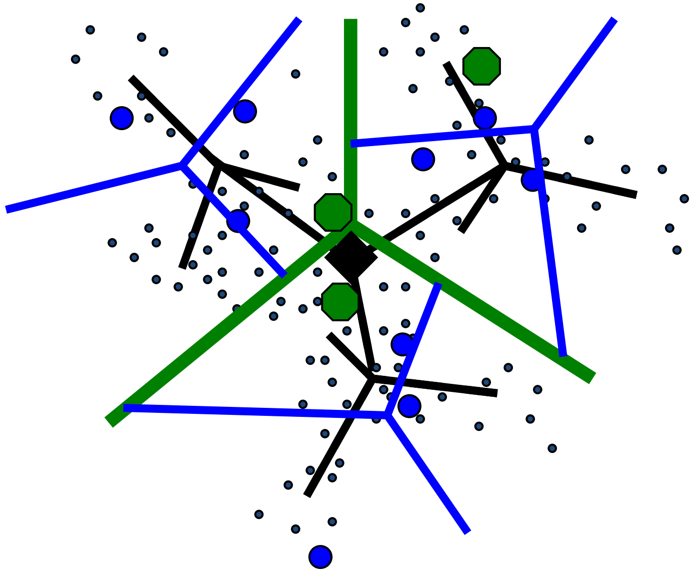
(B): Ideal clusters



Building Visual Dictionaries

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

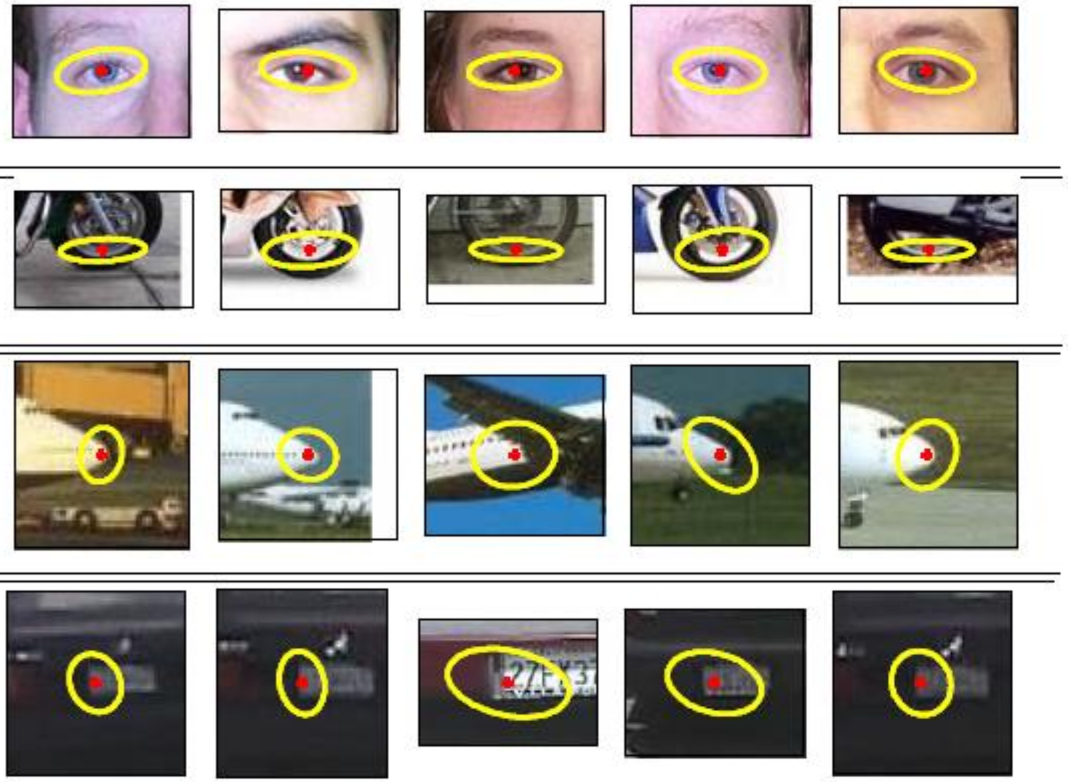
2. Cluster the patches
   – Cluster centers are the dictionary



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

Examples of learned codewords

(a)
(b)
(c)
(d)

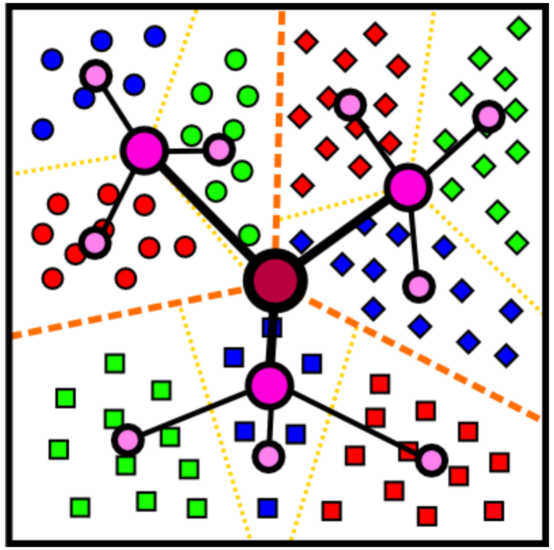


Most likely codewords for 4 learned "topics"

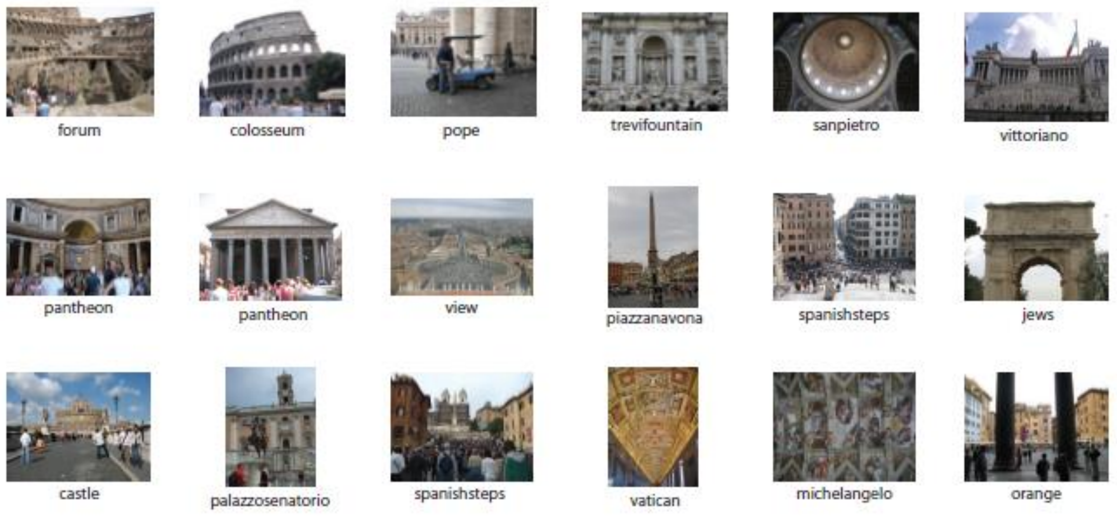
http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic05b.pdf Sivic et al. ICCV 2005

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/

Machine Learning Problems

	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

The machine learning framework

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

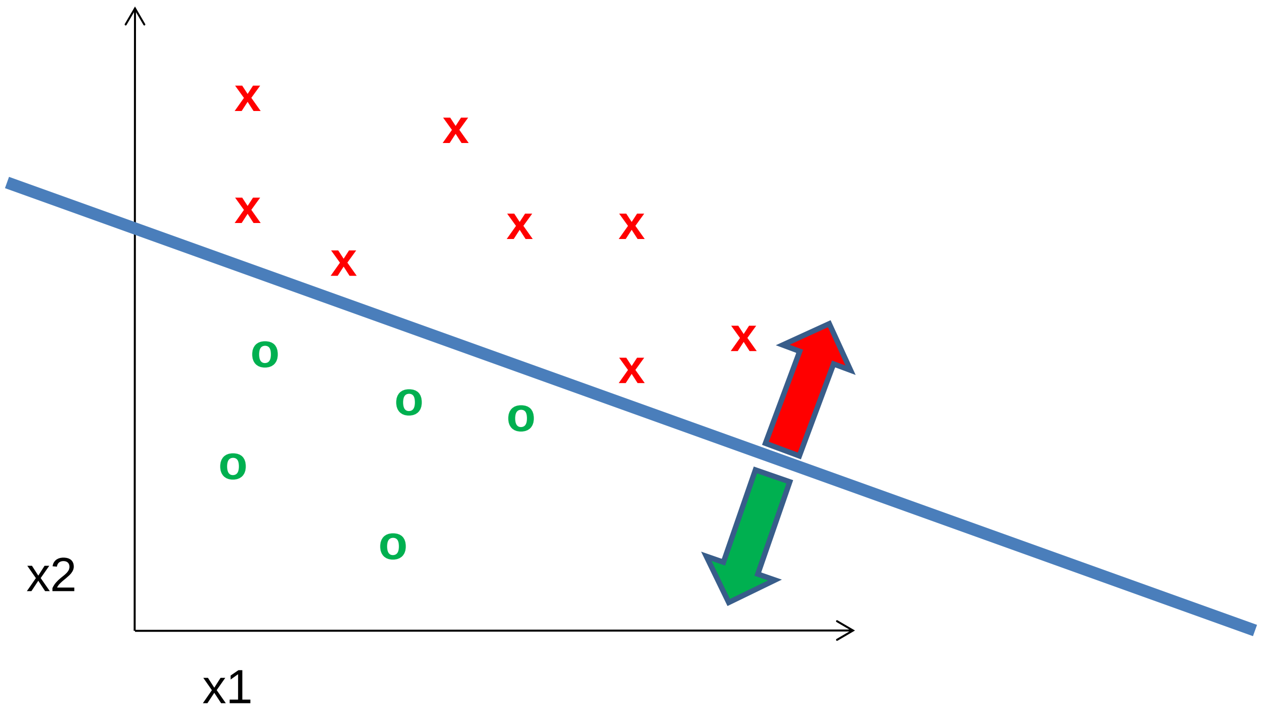
f() = “apple”

f() = “tomato”

f() = “cow”

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)

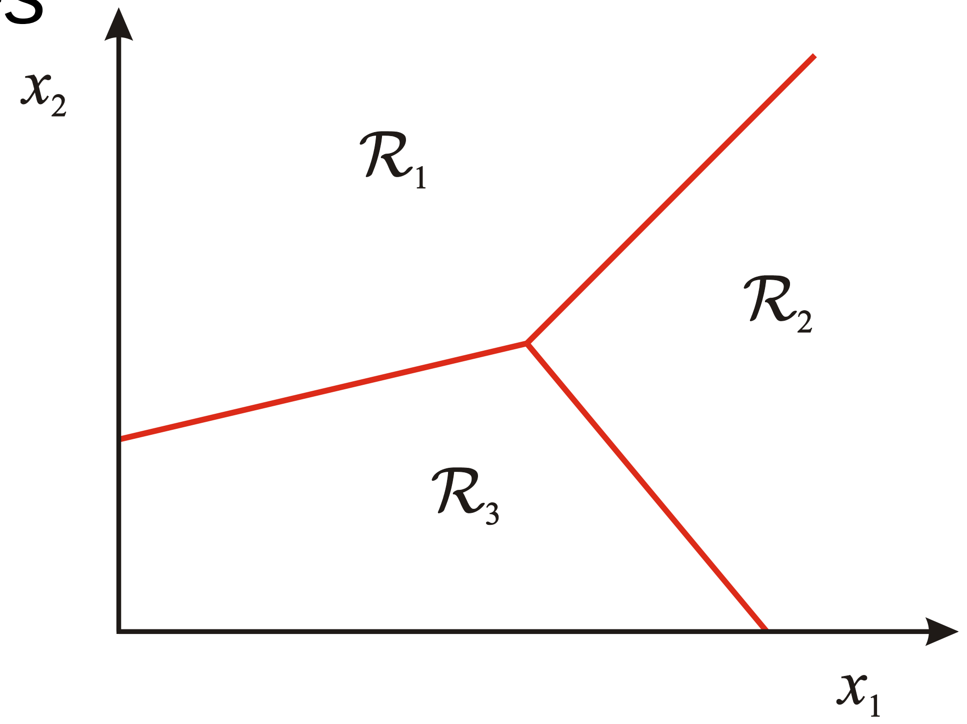
- How well does a learned model generalize from the data it was trained on to a new test set?

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.

Classification

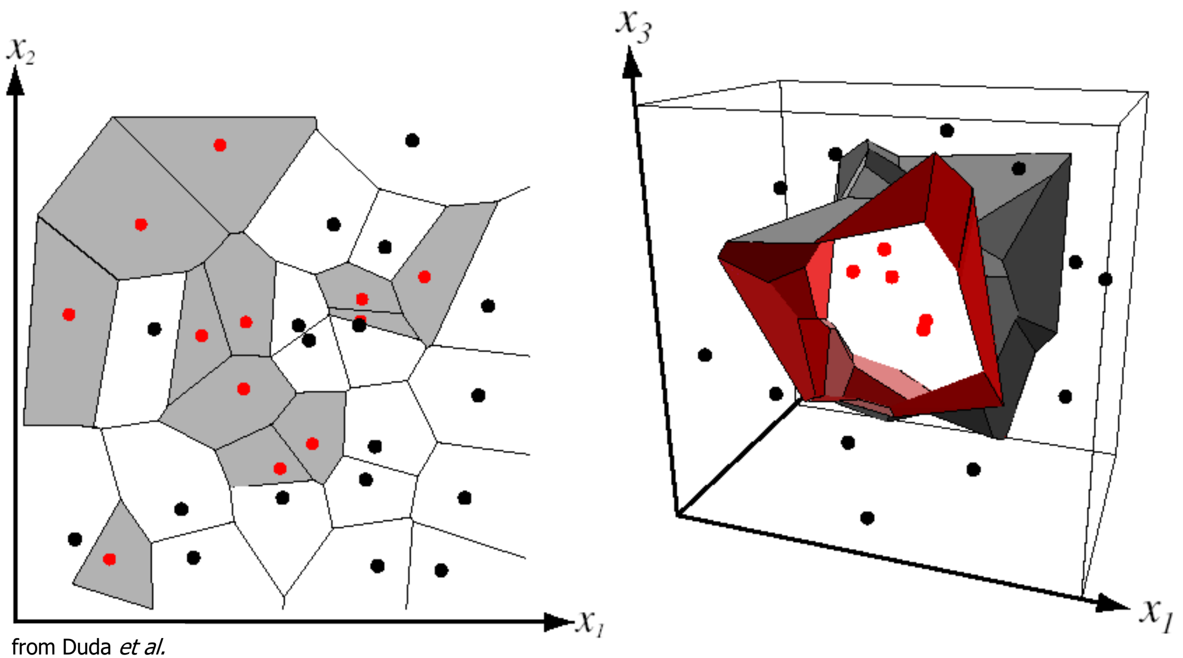
- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Slide credit: L. Lazebnik

Nearest Neighbor Classifier

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

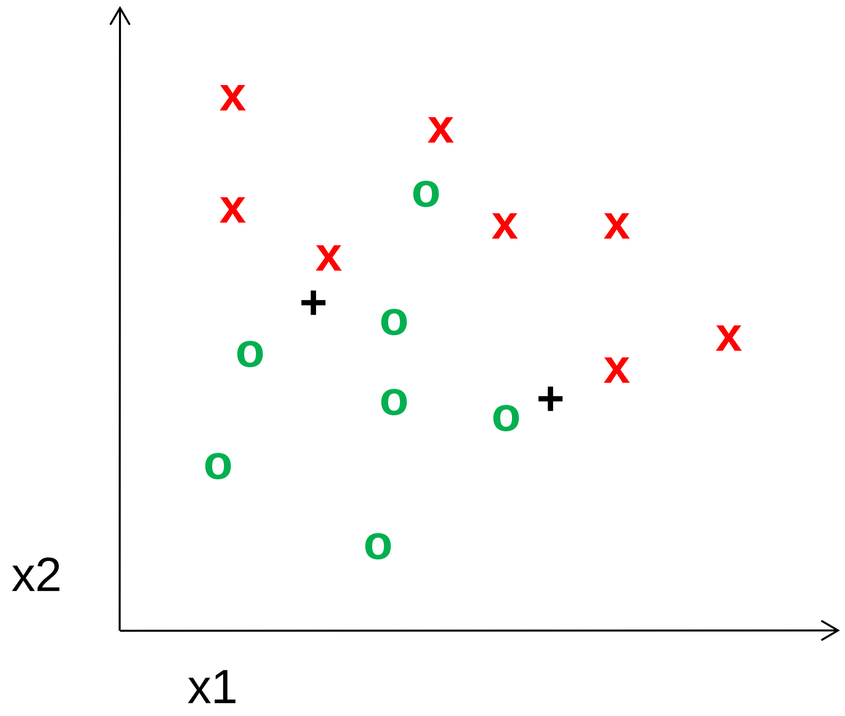


from Duda *et al.*

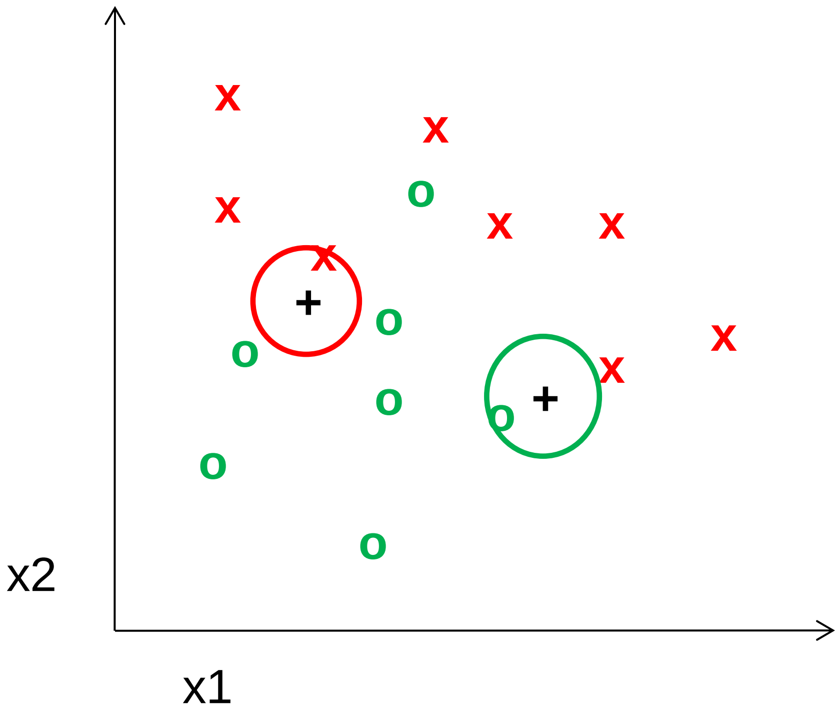
Voronoi partitioning of feature space for two-category 2D and 3D data

Source: D. Lowe

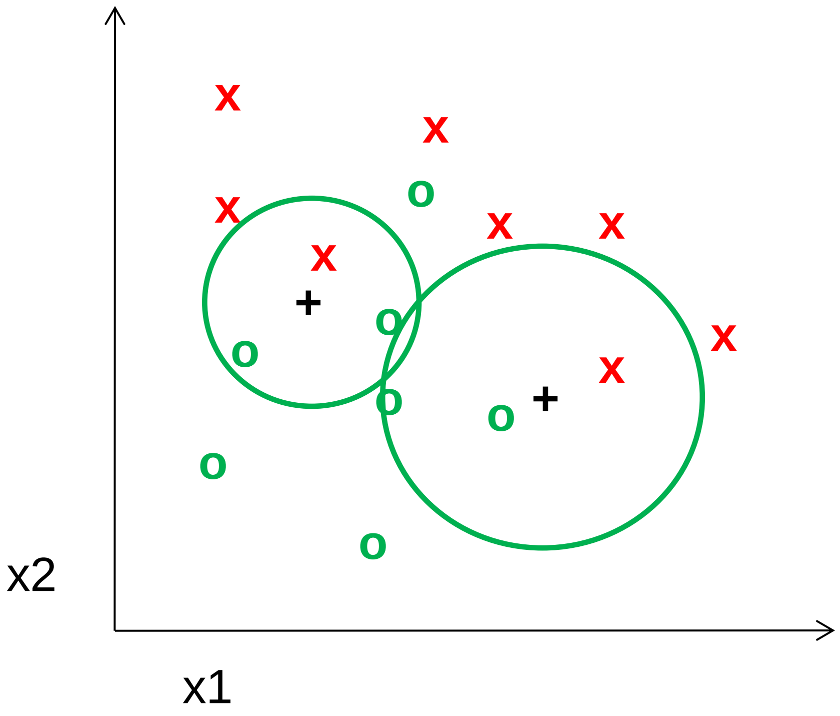
K-nearest neighbor



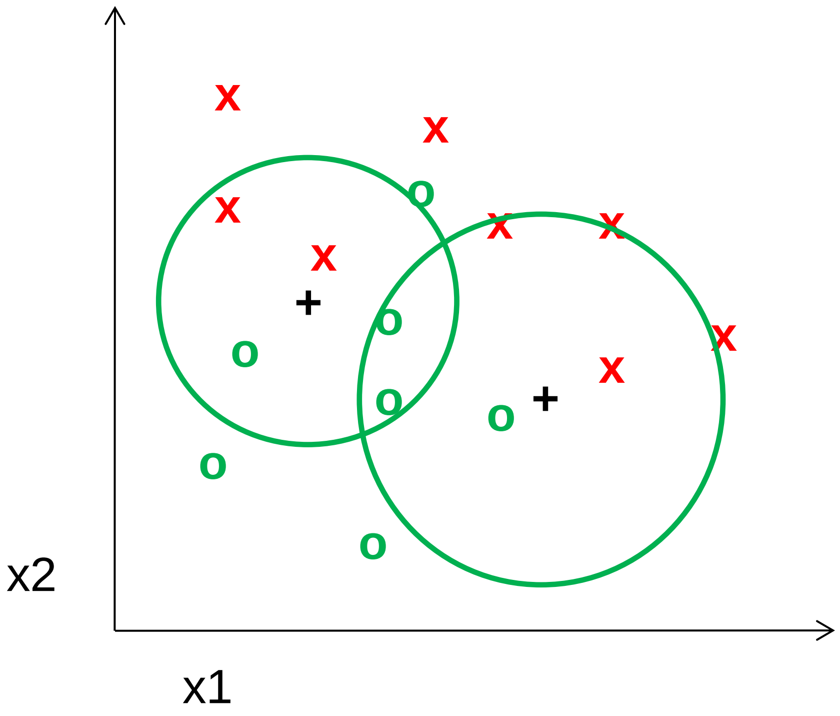
1-nearest neighbor



3-nearest neighbor



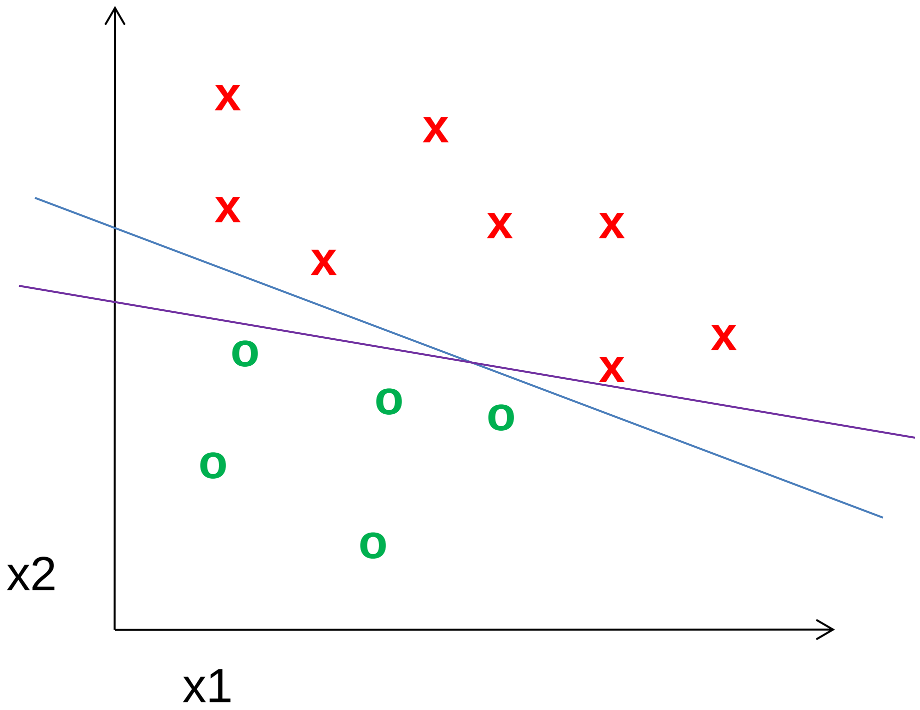
5-nearest neighbor



Using K-NN

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

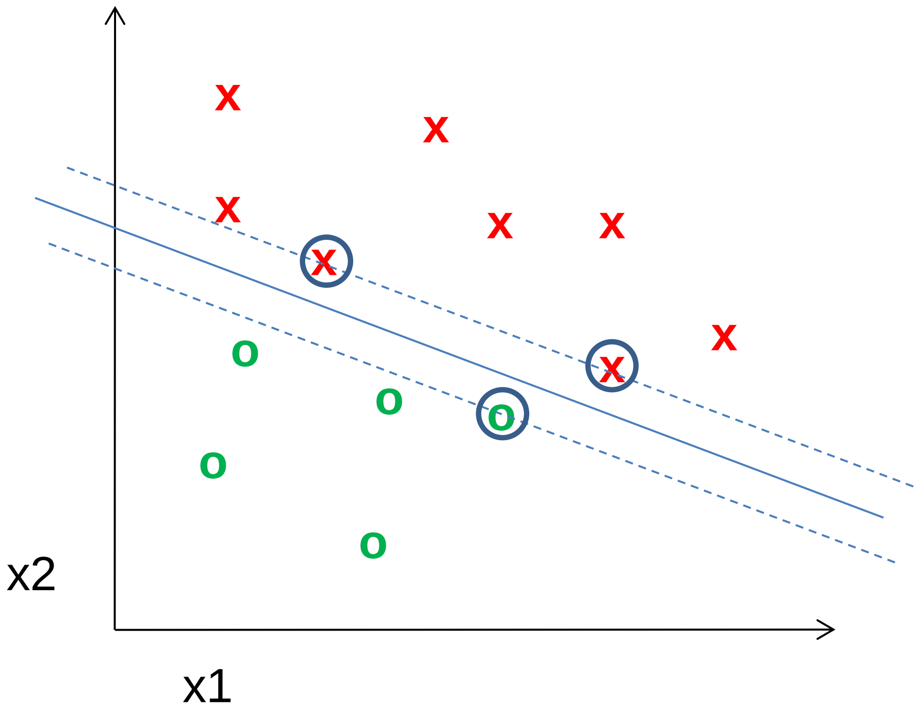
Classifiers: Linear SVM



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

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

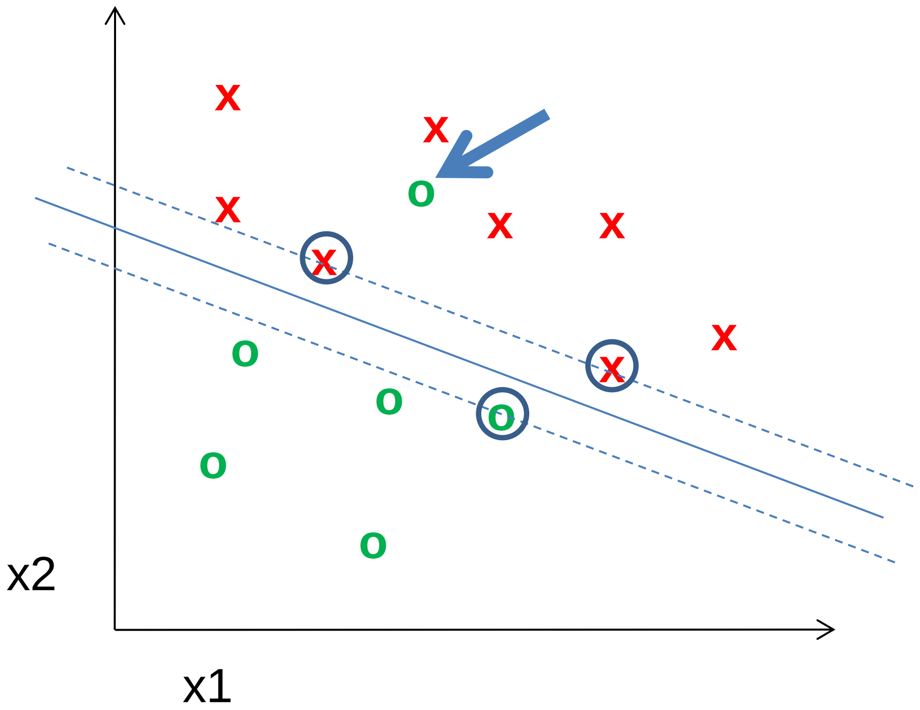
Classifiers: Linear SVM



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

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

Classifiers: Linear SVM

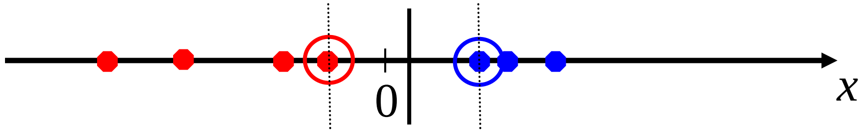


- Find a *linear function* to separate the classes:

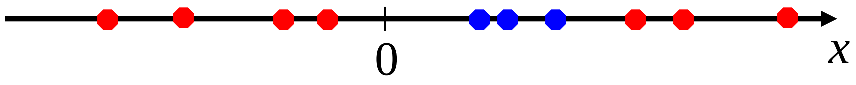
$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

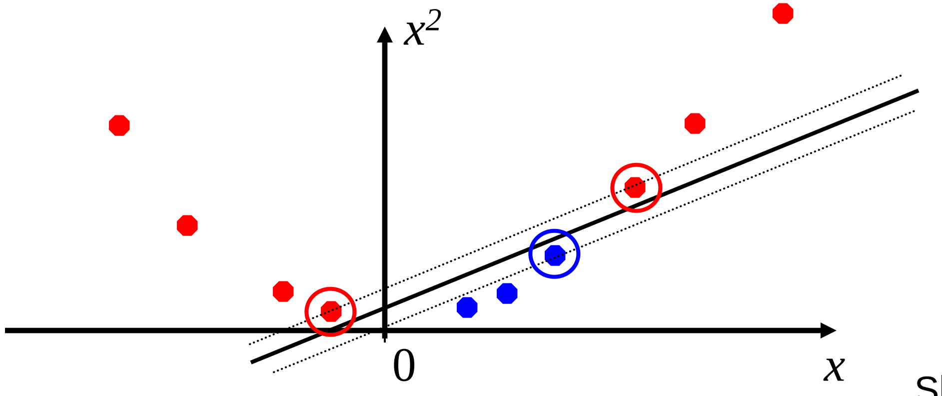
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



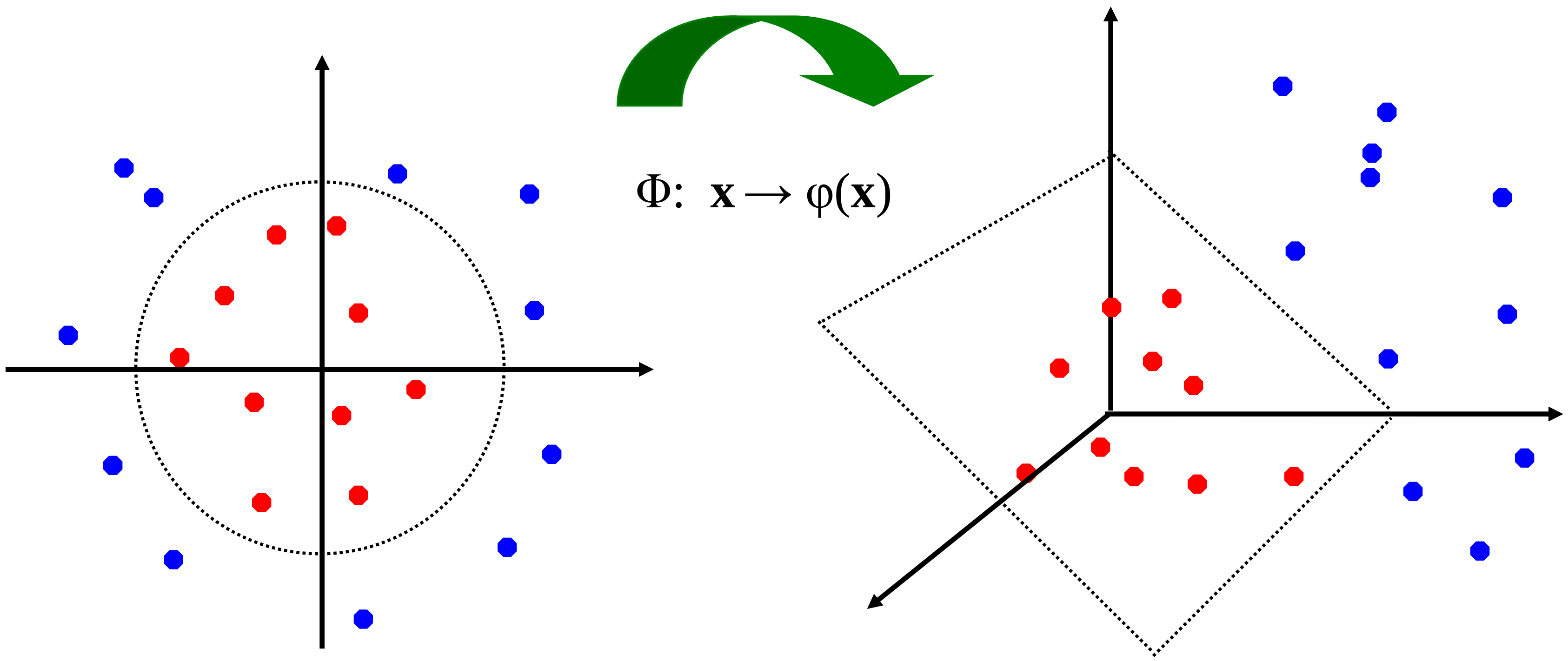
- We can map it to a higher-dimensional space:



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

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

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)

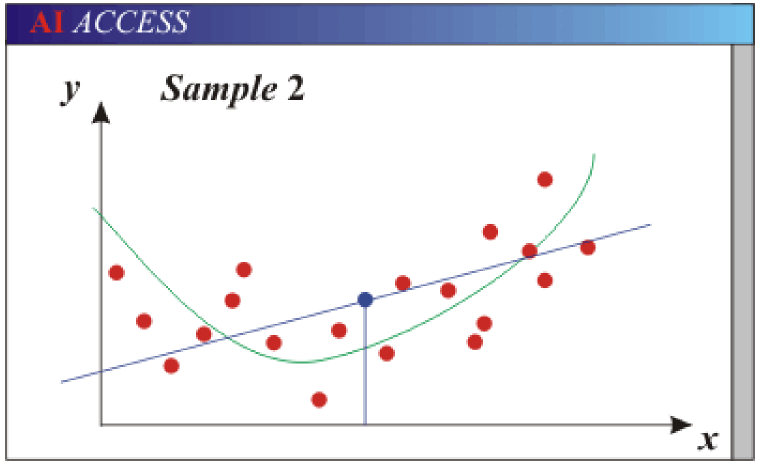
- How well does a learned model generalize from the data it was trained on to a new test set?

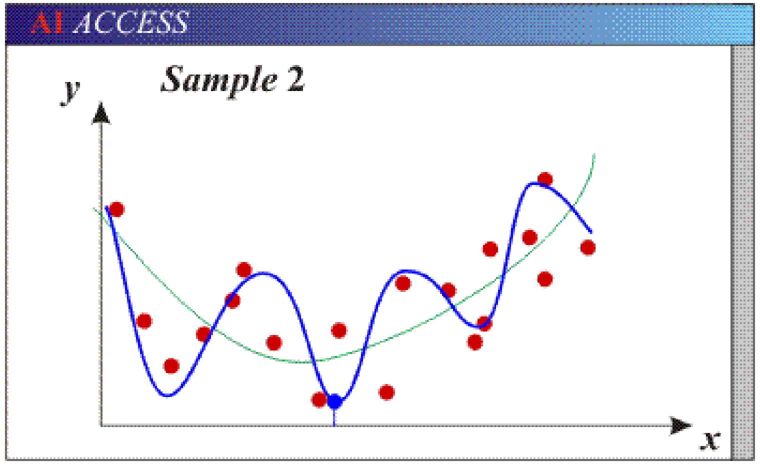
Generalization

- 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

Bias-Variance Trade-off



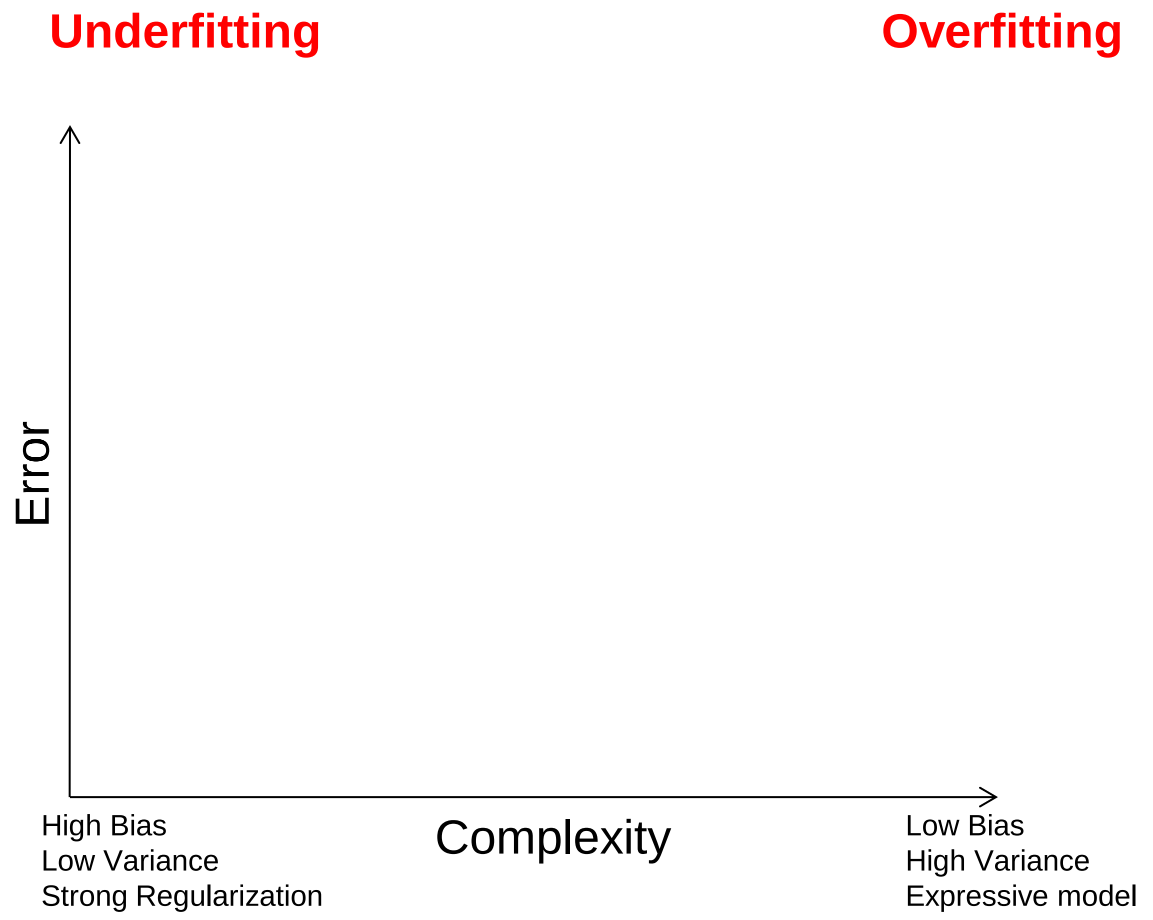


- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

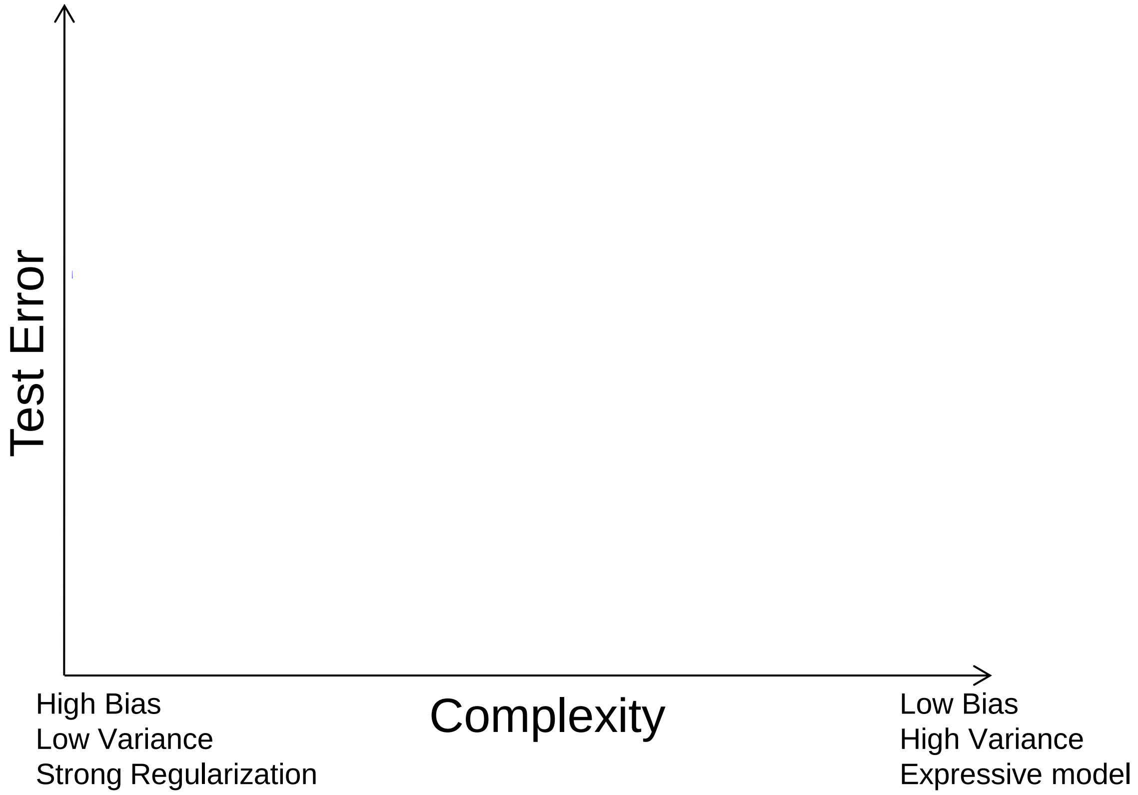
Slide credit: D. Hoiem

Bias-variance tradeoff



Slide credit: D. Hoiem

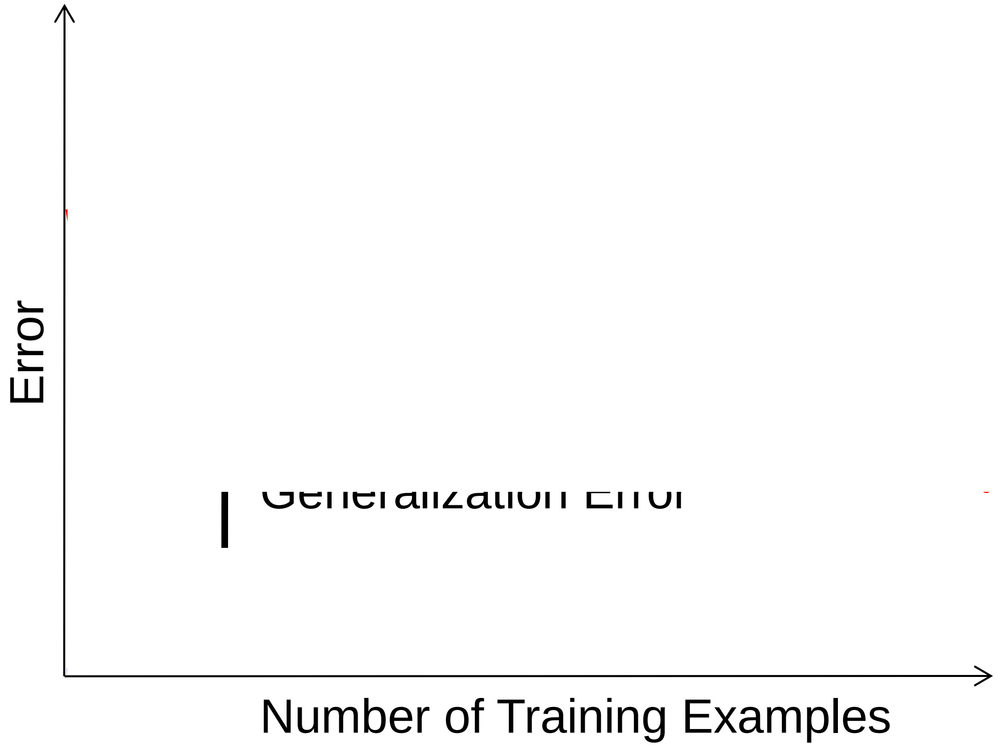
Bias-variance tradeoff



Slide credit: D. Hoiem

Effect of Training Size

Fixed prediction model



Remember…

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

- Error sources
  - Bias: due to over-simplifications / regularization
  - Variance: due to inability to perfectly estimate parameters from limited data

- How to reduce variance (expressiveness)?
  - Choose a simpler classifier
  - Regularize the parameters
  - Get more training data
- How to reduce bias (regularization)?
  - 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 expressive classifiers with more training data (bias-variance tradeoff)

Slide credit: D. Hoiem

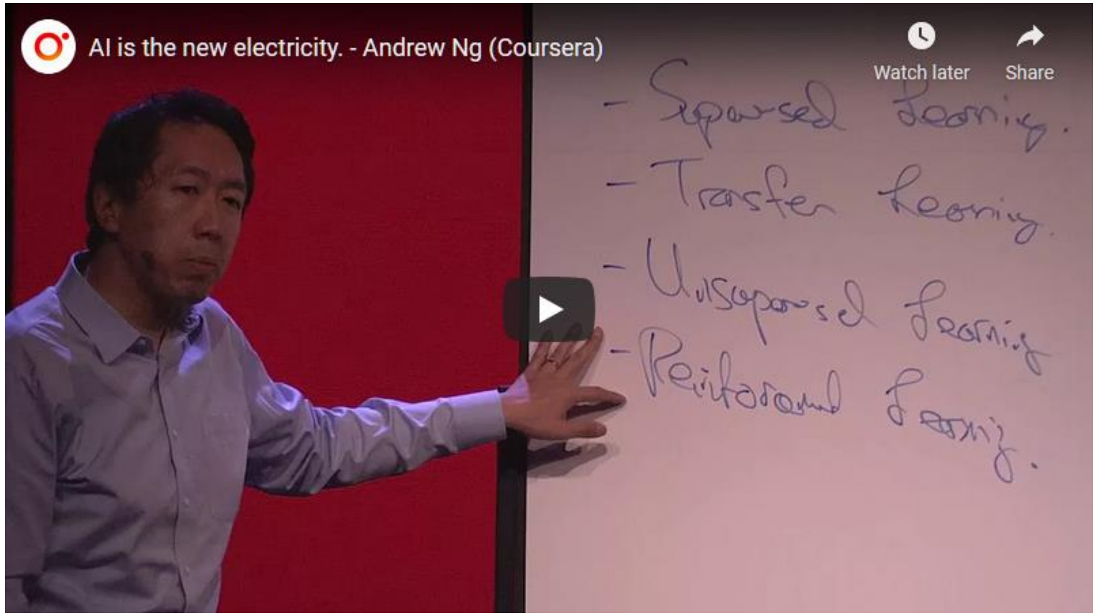
Machine Learning Considerations

- 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

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

- 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