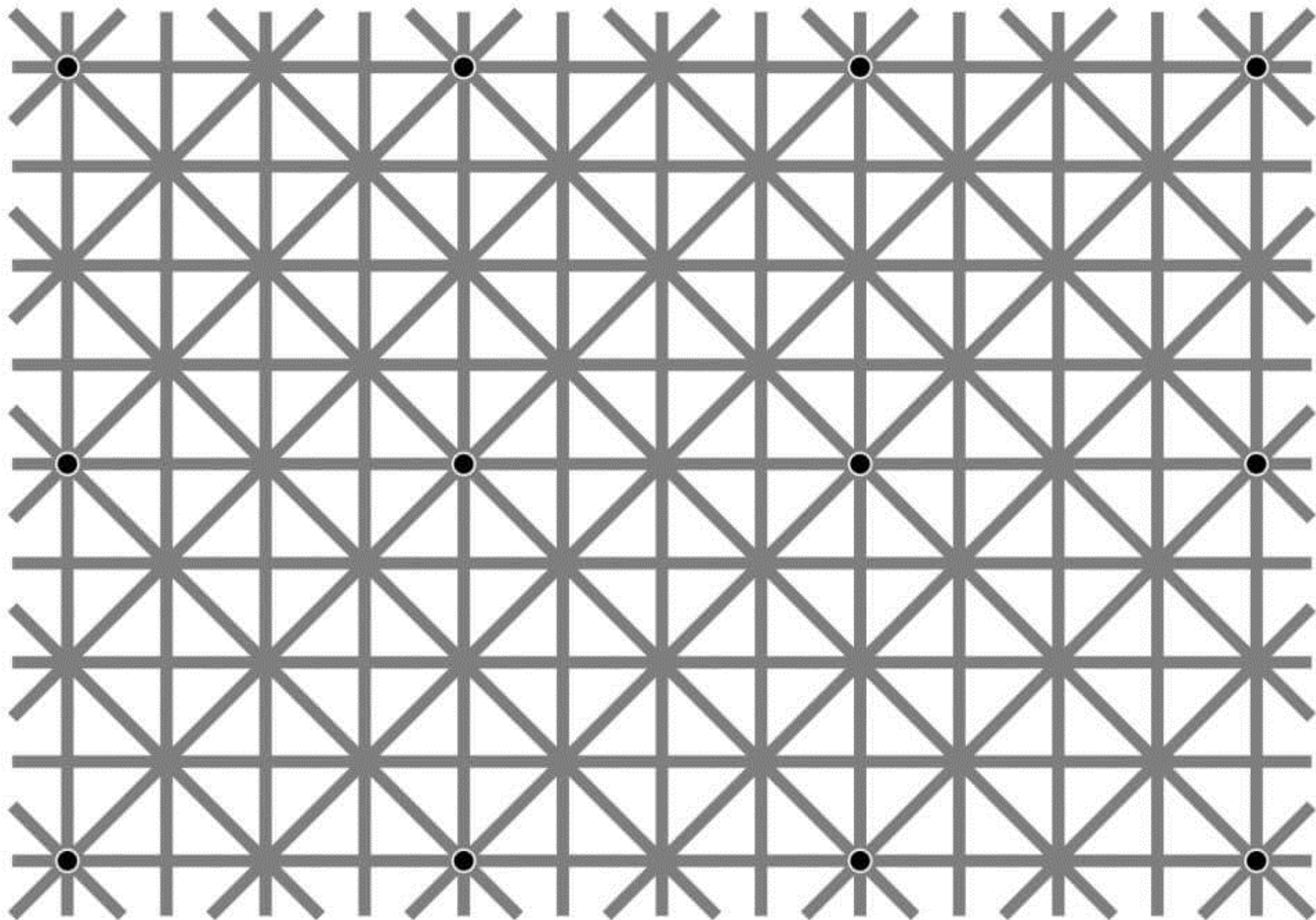




Ninio, J. and Stevens, K. A. (2000) Variations on the Hermann grid: an extinction illusion.
Perception, 29, 1209-1217.

Variations on the Hermann grid: an extinction illusion

Jacques Ninio

Laboratoire de Physique Statistique⁽¹⁾, École Normale Supérieure, 24 rue Lhomond,
75231 Paris cedex 05, France; e-mail: jacques.ninio@lps.ens.fr

Kent A Stevens

Department of Computer Science, Deschutes Hall, University of Oregon, Eugene, OR 97403, USA;
e-mail: kent@cs.uoregon.edu

Received 21 September 1999, in revised form 21 June 2000

Abstract. When the white disks in a scintillating grid are reduced in size, and outlined in black, they tend to disappear. One sees only a few of them at a time, in clusters which move erratically on the page. Where they are not seen, the grey alleys seem to be continuous, generating grey crossings that are not actually present. Some black sparkling can be seen at those crossings where no disk is seen. The illusion also works in reverse contrast.

The Hermann grid (Brewster 1844; Hermann 1870) is a robust illusion. It is classically presented as a two-dimensional array of black squares, separated by rectilinear alleys. It is thought to be caused by processes of local brightness computation in arrays of

From Wikipedia, the free encyclopedia

A **grid illusion** is any kind of [grid](#) that deceives a person's vision. The two most common types of grid illusions are the **Hermann grid illusion** and the **scintillating grid illusion**.

Hermann grid illusion [[edit](#)]

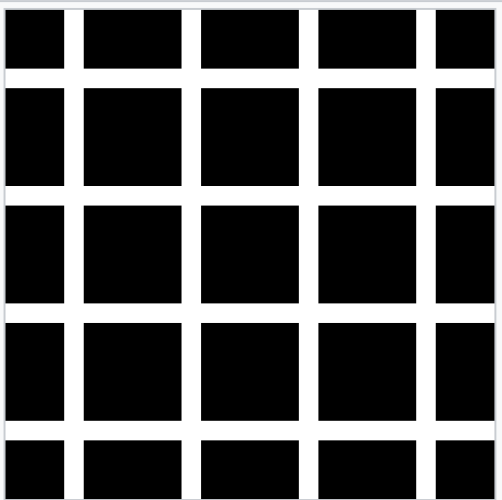
The **Hermann grid illusion** is an [optical illusion](#) reported by [Ludimar Hermann](#) in 1870.^[1] The illusion is characterized by "ghostlike" grey blobs perceived at the intersections of a white (or light-colored) grid on a black background. The grey blobs disappear when looking directly at an intersection.

Scintillating grid illusion [[edit](#)]

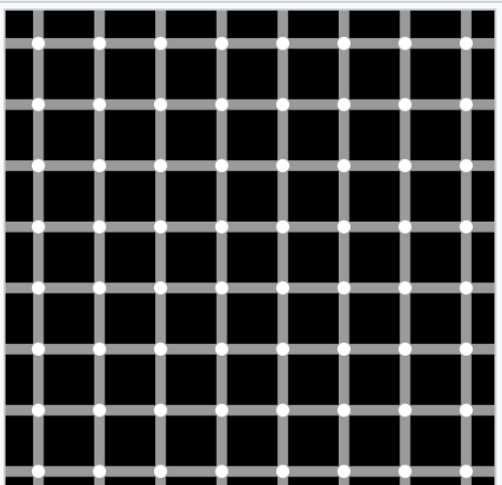
The **scintillating grid illusion** is an [optical illusion](#), discovered by E. and B. Lingelbach and M. Schrauf in 1994.^[2] It is often considered a variation of the Hermann grid illusion but possesses different properties.^{[2][3]}

It is constructed by superimposing white discs on the intersections of orthogonal gray bars on a black background. Dark dots seem to appear and disappear rapidly at random intersections, hence the label "scintillating". When a person keeps their eyes directly on a single intersection, the dark dot does not appear. The dark dots disappear if one is too close to or too far from the image.

Differences between the scintillating and Hermann grid illusions [[edit](#)]



An example of the Hermann grid illusion. Dark blobs appear at the intersections.



https://en.wikipedia.org/wiki/Grid_illusion

Machine Learning Crash Course



Computer Vision
James Hays

Photo: CMU Machine Learning
Department protests G20

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

Machine Learning Problems

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

Dimensionality Reduction



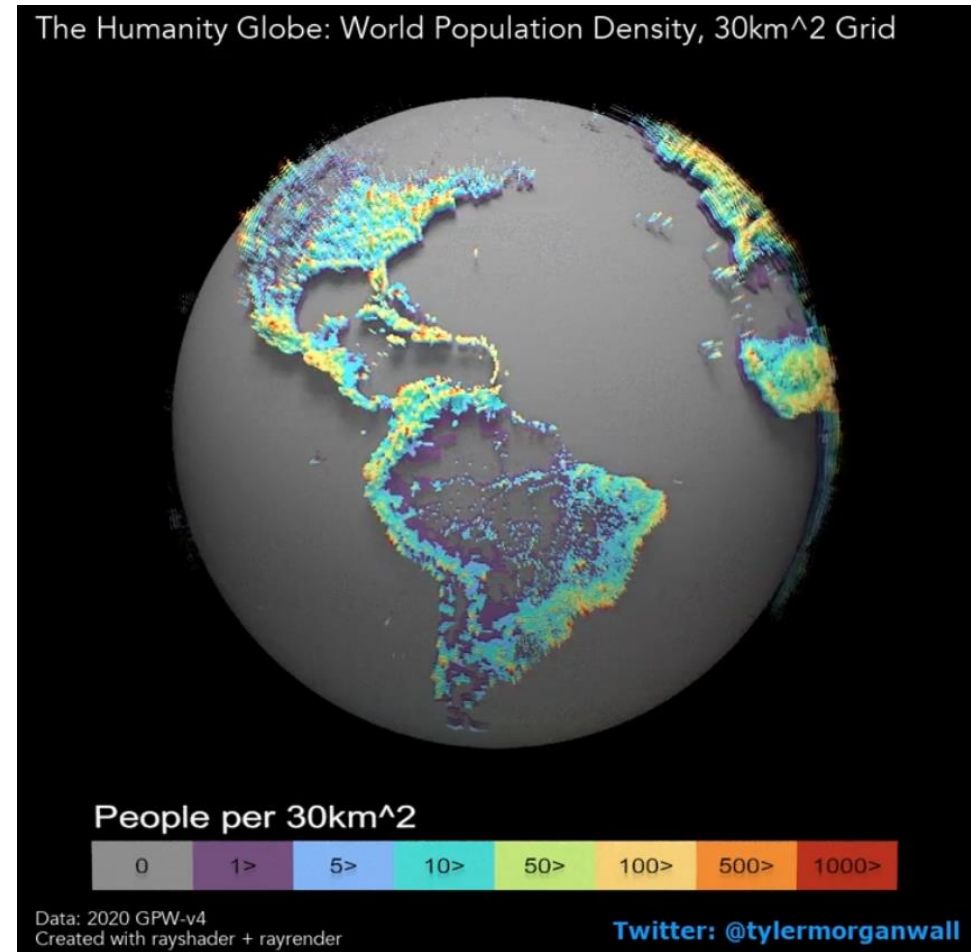
Simplest dimensionality reduction: drop a dimension

Dimensionality Reduction



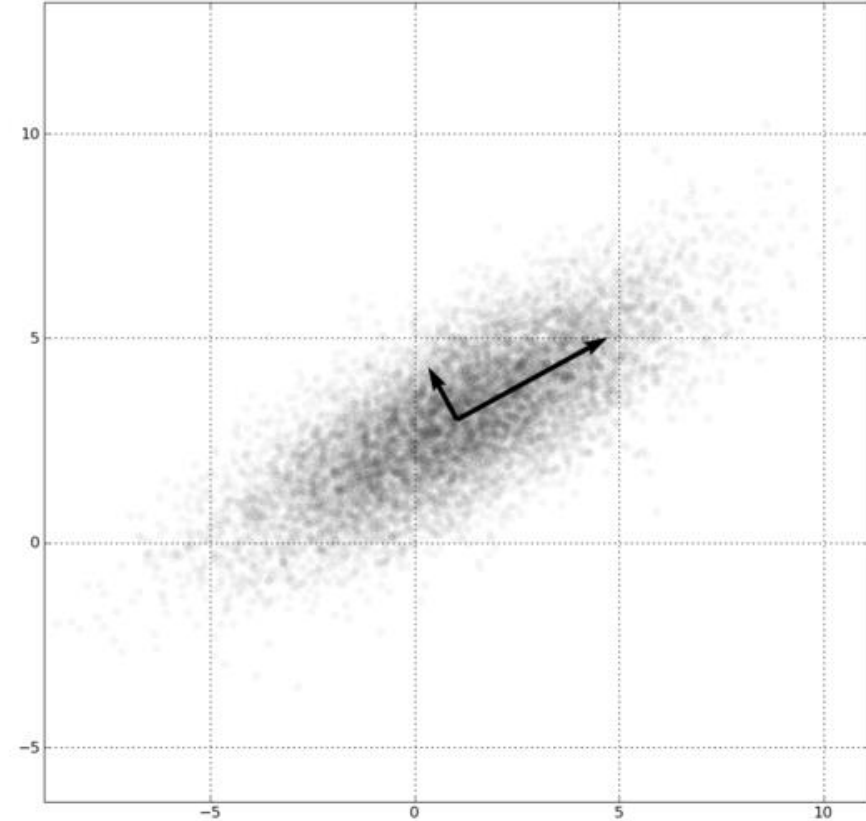
The Earth is pretty smooth

Non-linear Dimensionality Reduction



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

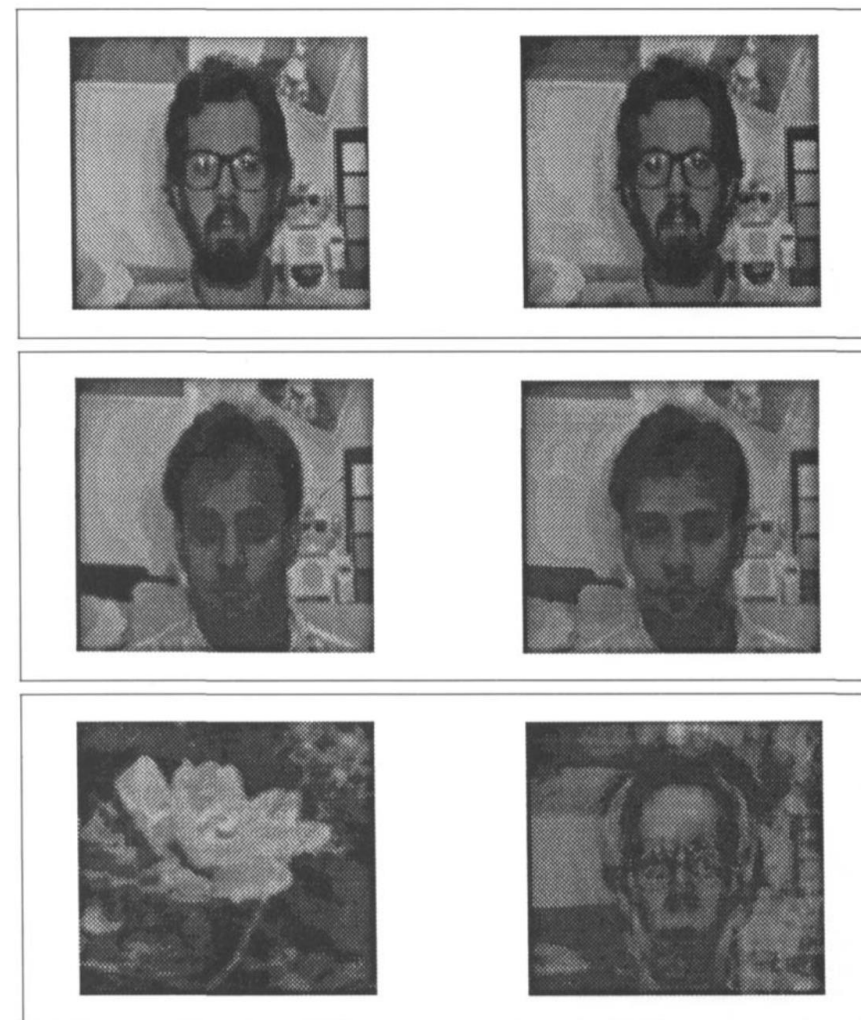


Figure 1. (b) The average face Ψ .



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

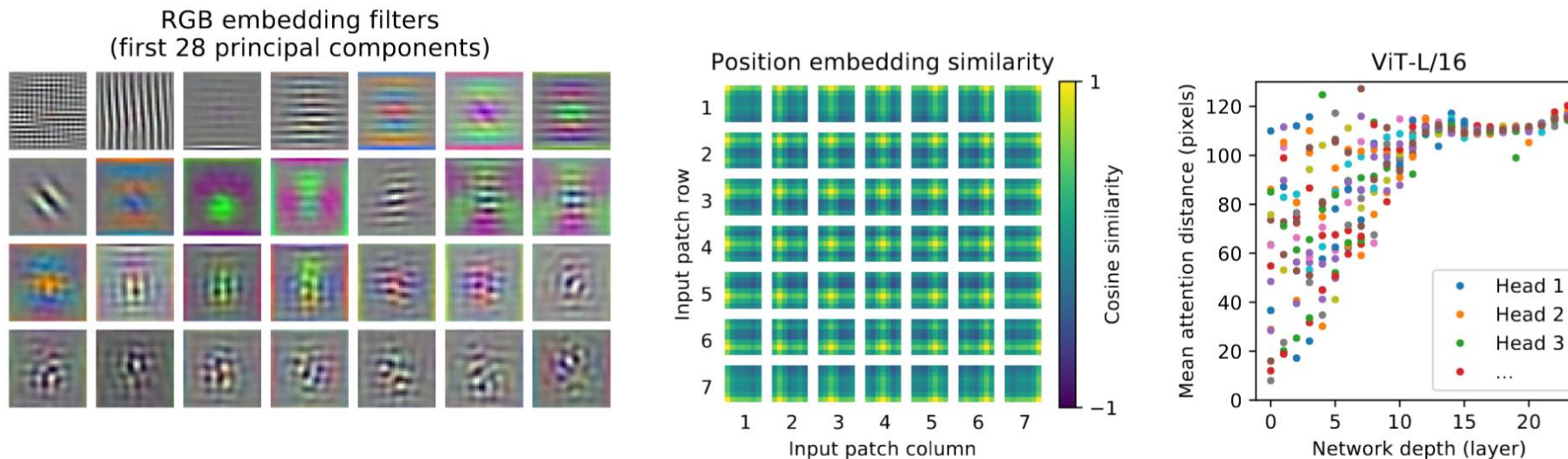
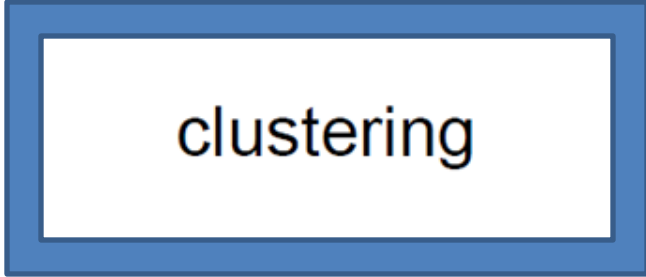


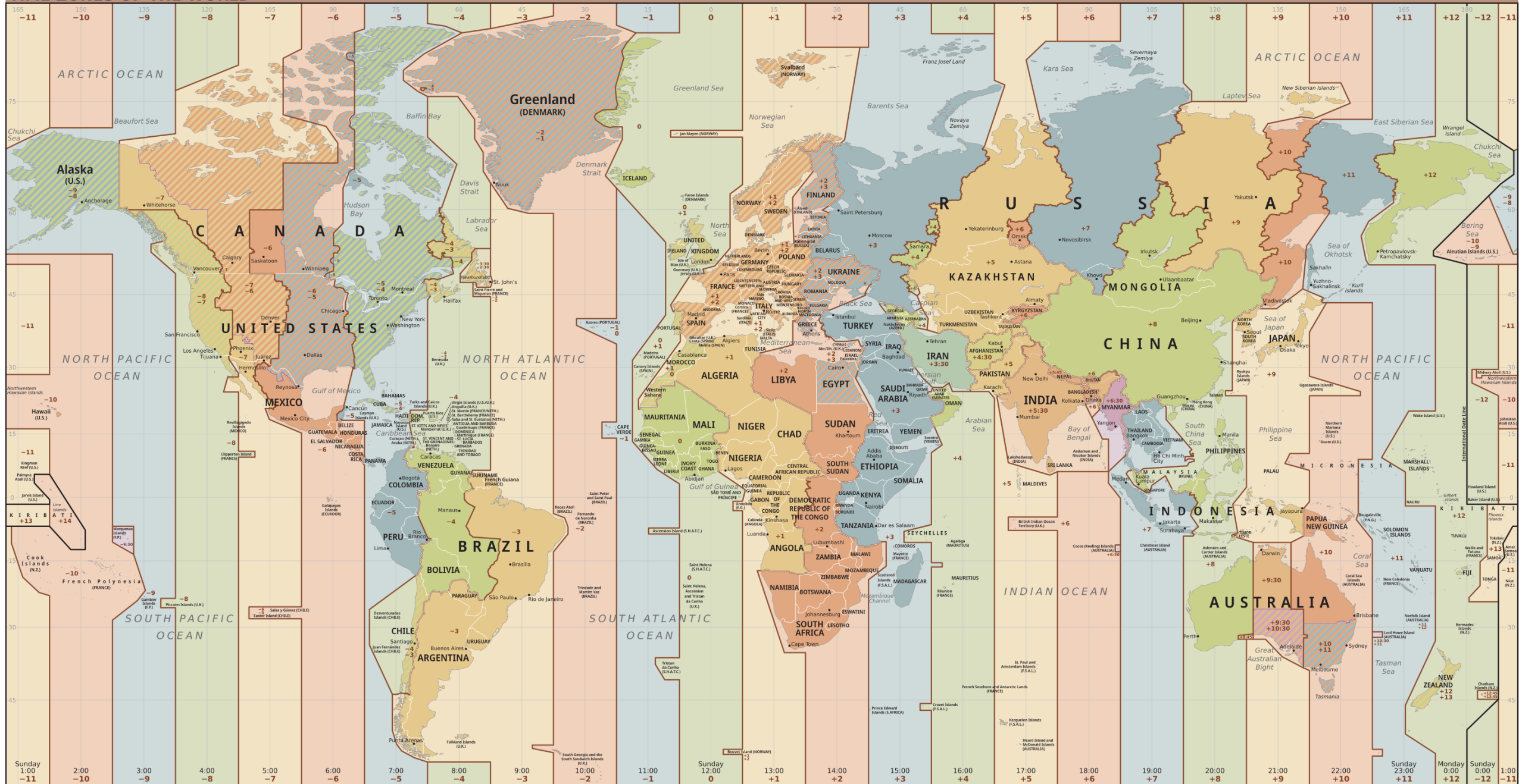
Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches.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

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	 clustering
<i>Continuous</i>	regression	dimensionality reduction

TIME ZONES OF THE WORLD



* These disputed areas observe UTC+3 de facto.

North America, except Greenland:
from second Sunday in March
to first Sunday in November

Chile:
from end of first Saturday in September
to end of first Saturday in April

In locations with two offsets, the second offset is used during the period listed below, and the first offset is used during the rest of the year.

Morocco:
from Sunday before Ramadan
to Sunday after Ramadan

Europe, Greenland and Lebanon:
from last Sunday in March
to last Sunday in October.

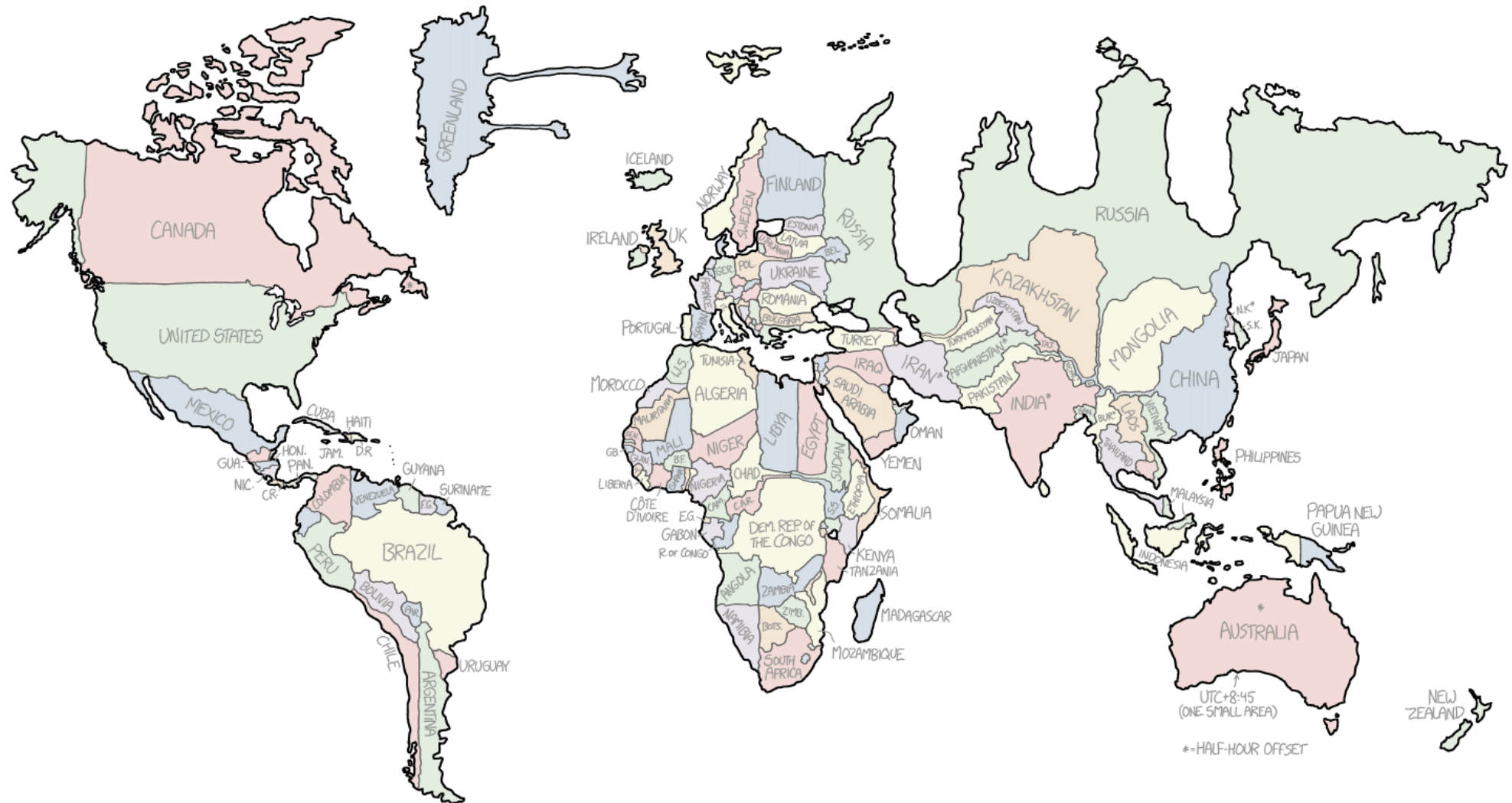
Israel:
from Friday before last Sunday in March
to last Sunday in October

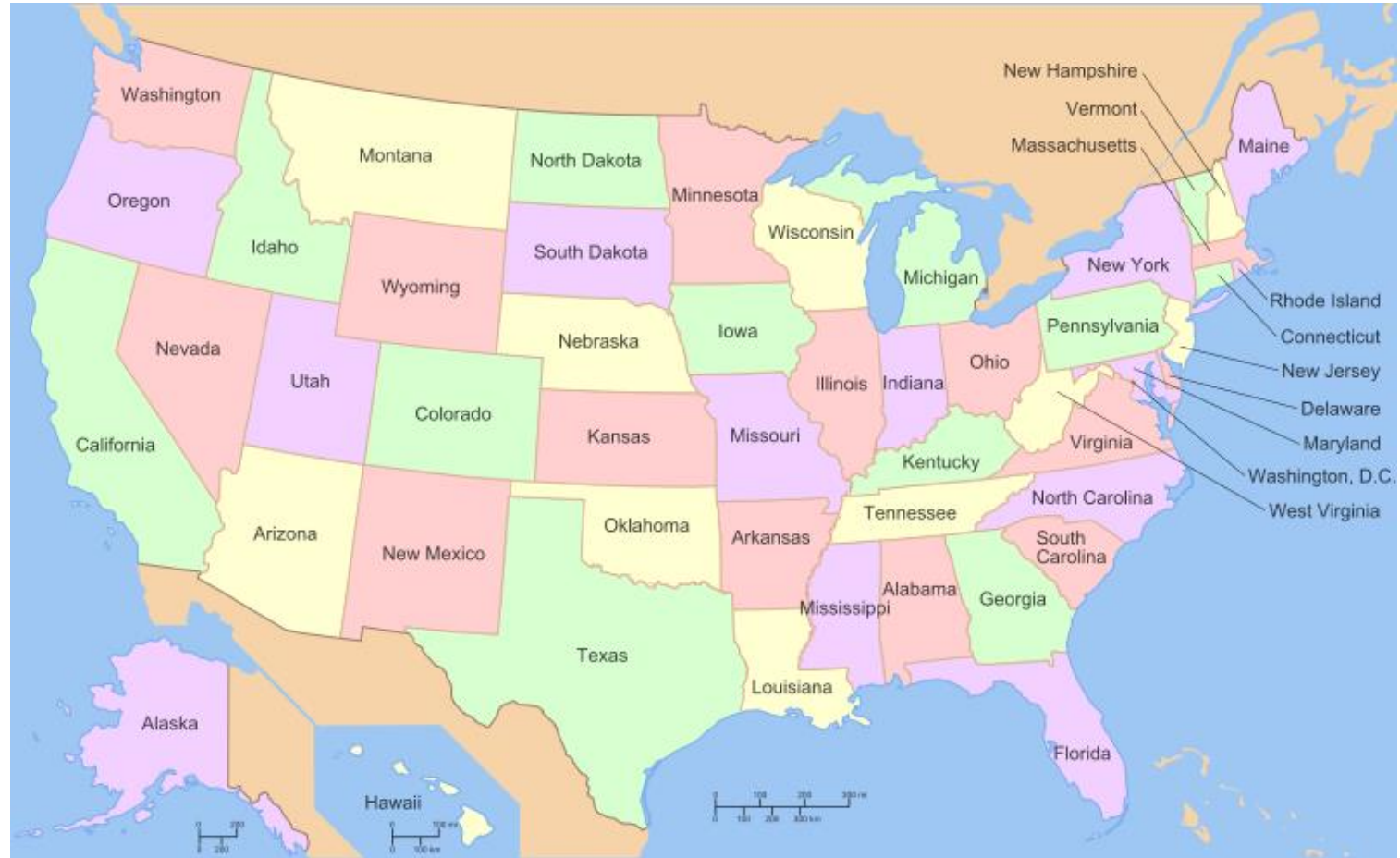
Palestine:
from second Saturday after Ramadan
to Saturday before last Sunday in October

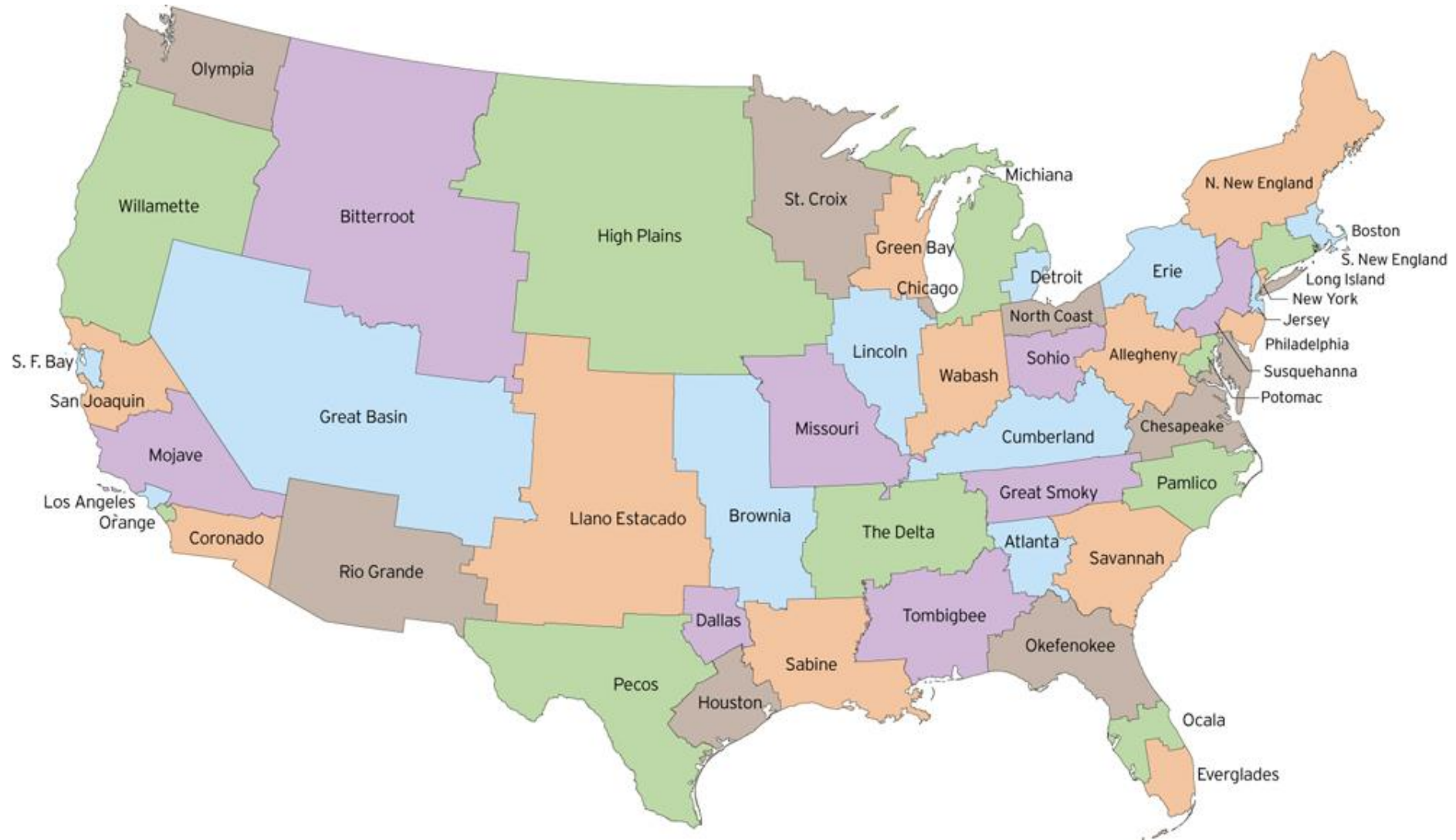
Australia:
from first Sunday in October
to first Sunday in April

New Zealand:
from last Sunday in September
to first Sunday in April

BAD MAP PROJECTION #79:
TIME ZONES
WHERE EACH COUNTRY *SHOULD* BE,
BASED ON ITS TIME ZONE(S)

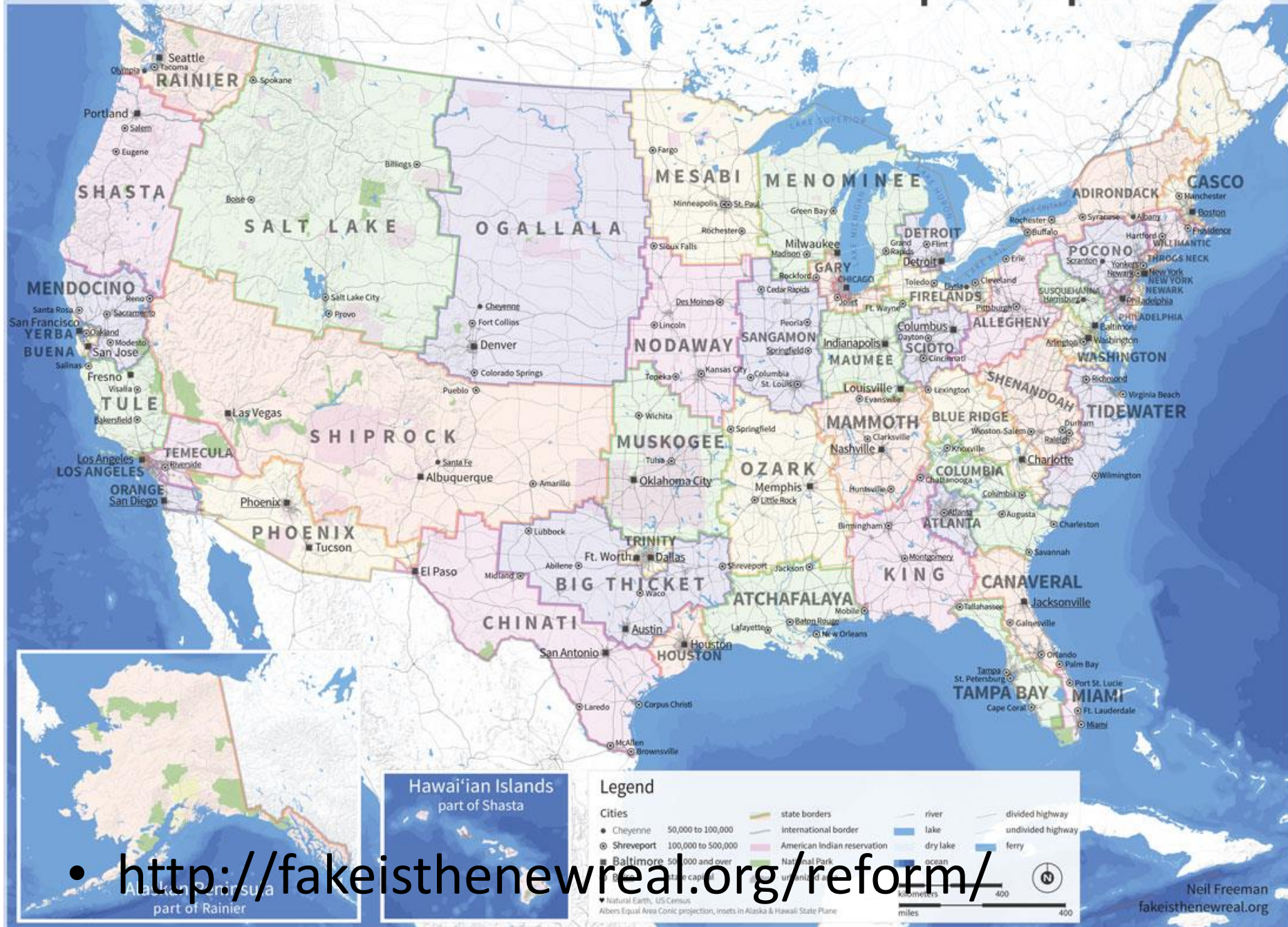






- <http://fakeisthenewreal.org/reform/>

The United States redrawn as Fifty States with Equal Population



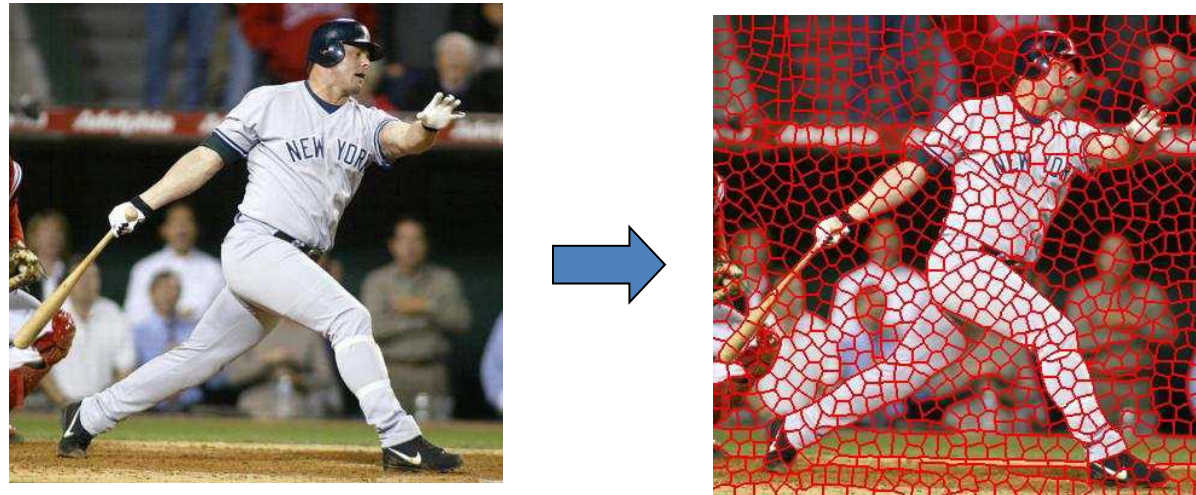
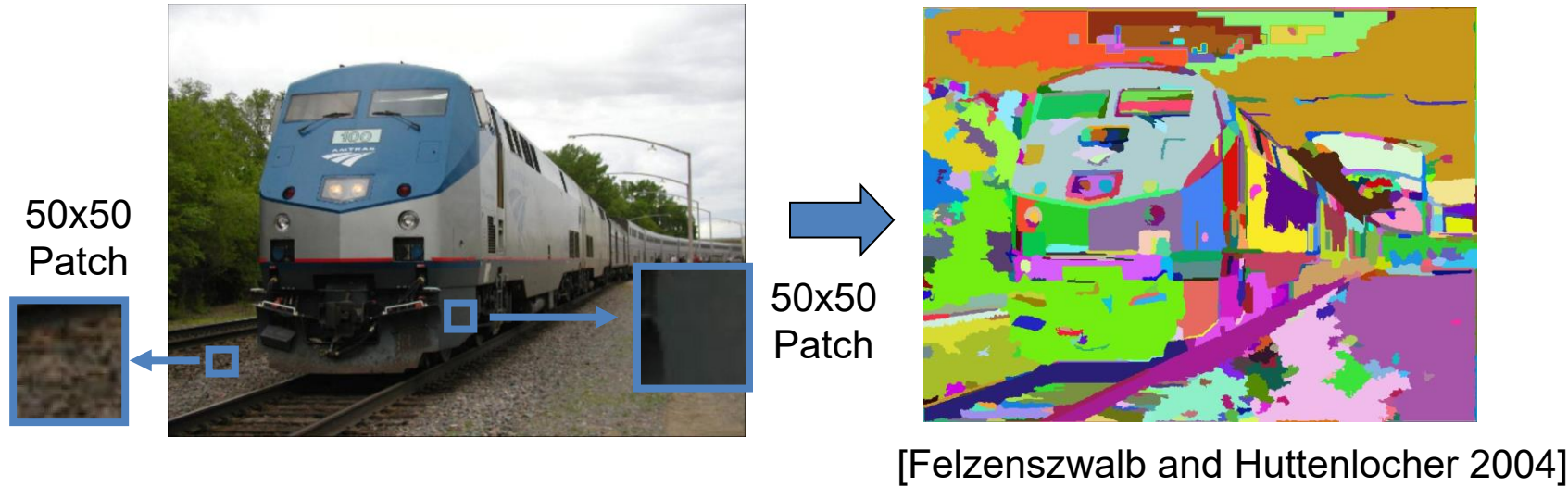
• <http://fakeisthenewreal.org/reform/>

Clustering example: image segmentation

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



Segmentation for feature support or efficiency (I.e. Superpixels)



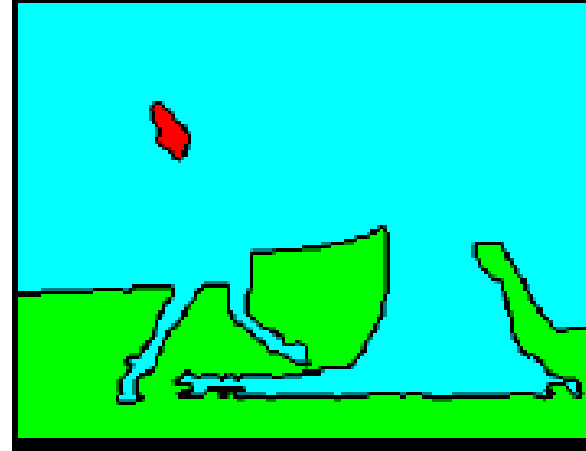
[Hoiem et al. 2005, Mori 2005]

Slide: Derek Hoiem

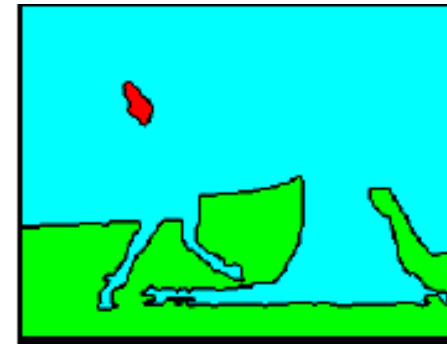
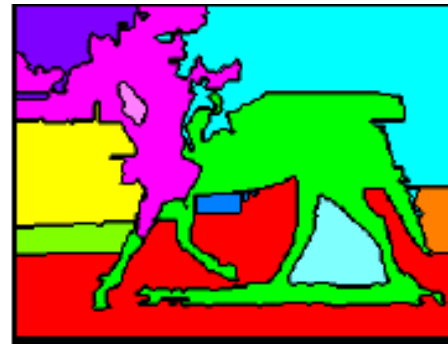
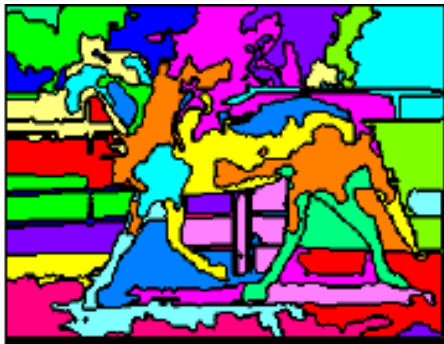
Types of segmentations



Oversegmentation

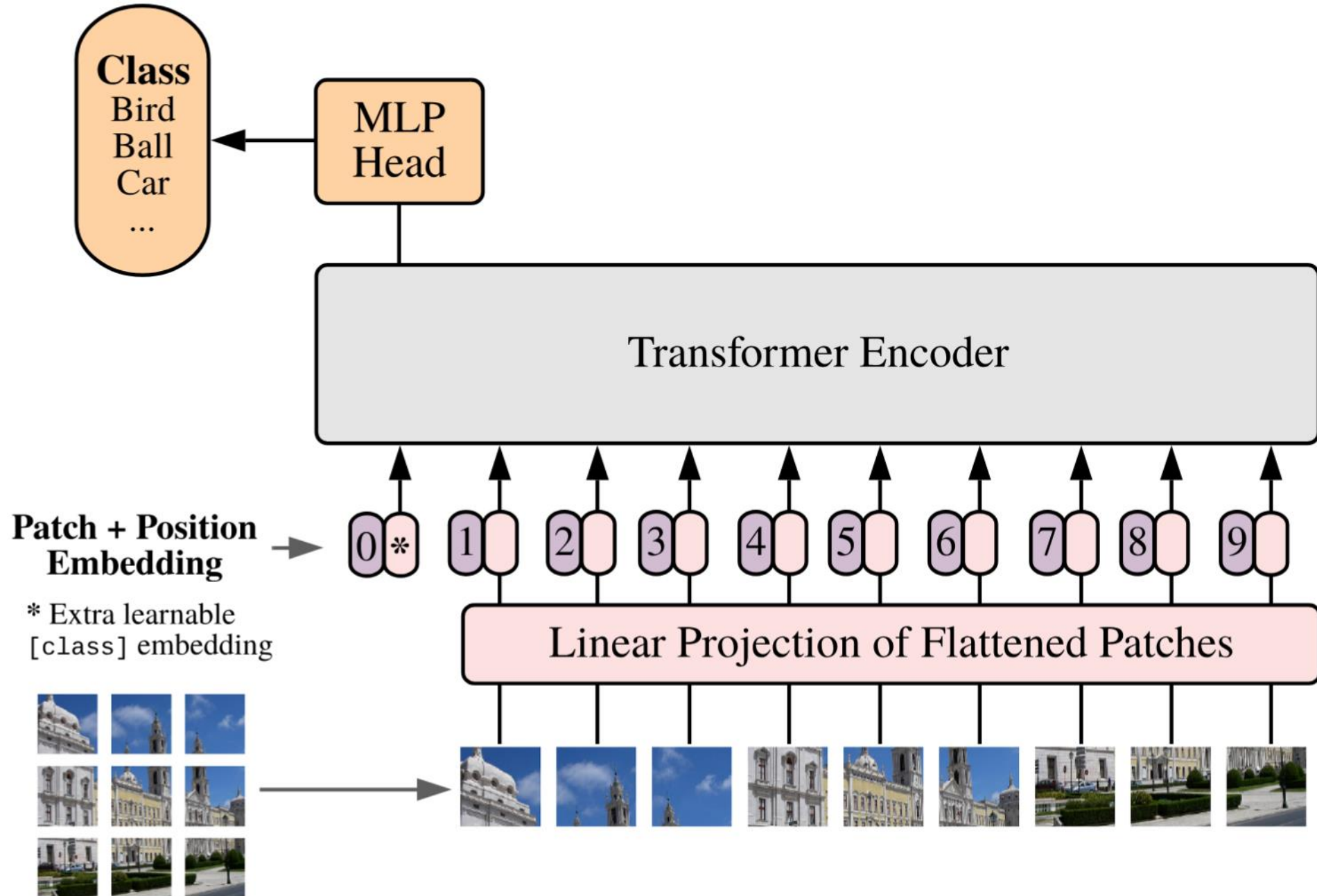


Undersegmentation



Multiple Segmentations

Vision Transformer (ViT)



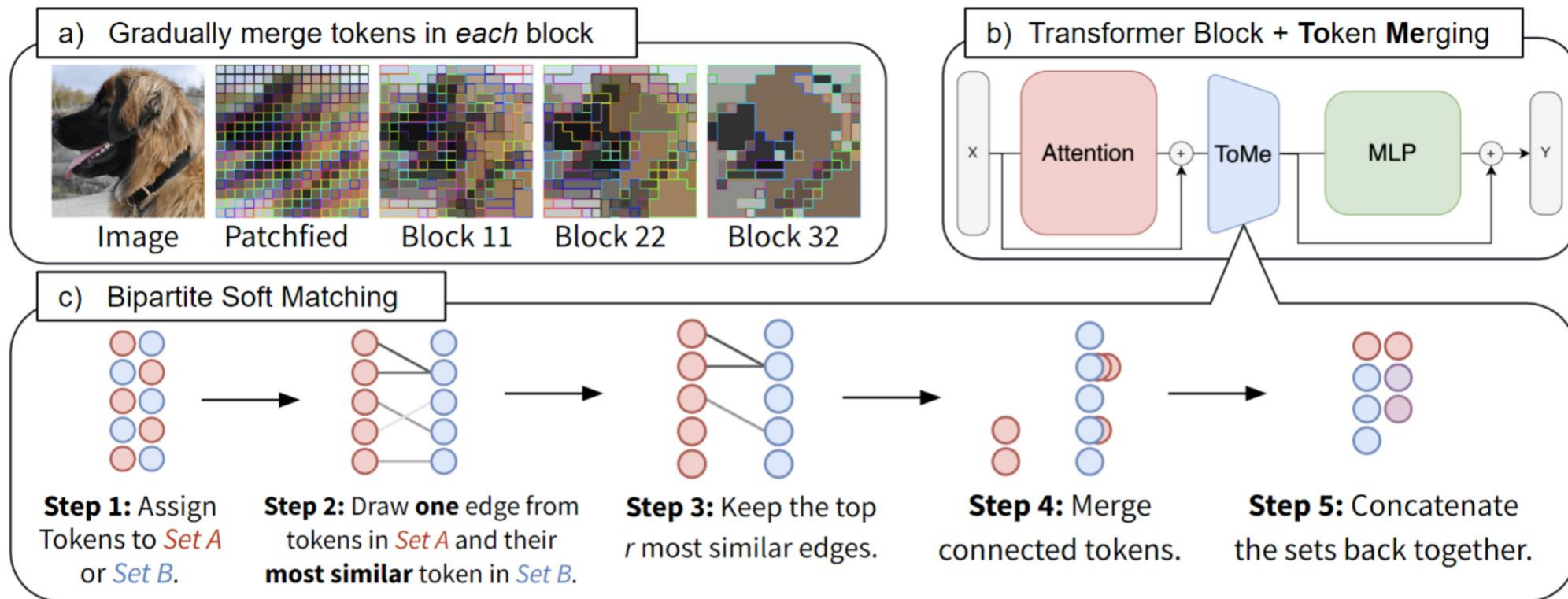


Figure 1: **Token Merging.** (a) With ToMe, similar patches are merged in each transformer block: for example, the dog’s fur is merged into a single token. (b) ToMe is simple and can be inserted inside the standard transformer block. (c) Our fast merging algorithm, see Appendix **D** for implementation.

Token Merging: Your ViT But Faster

[Daniel Bolya](#), [Cheng-Yang Fu](#), [Xiaoliang Dai](#), [Peizhao Zhang](#), [Christoph Feichtenhofer](#), [Judy Hoffman](#)

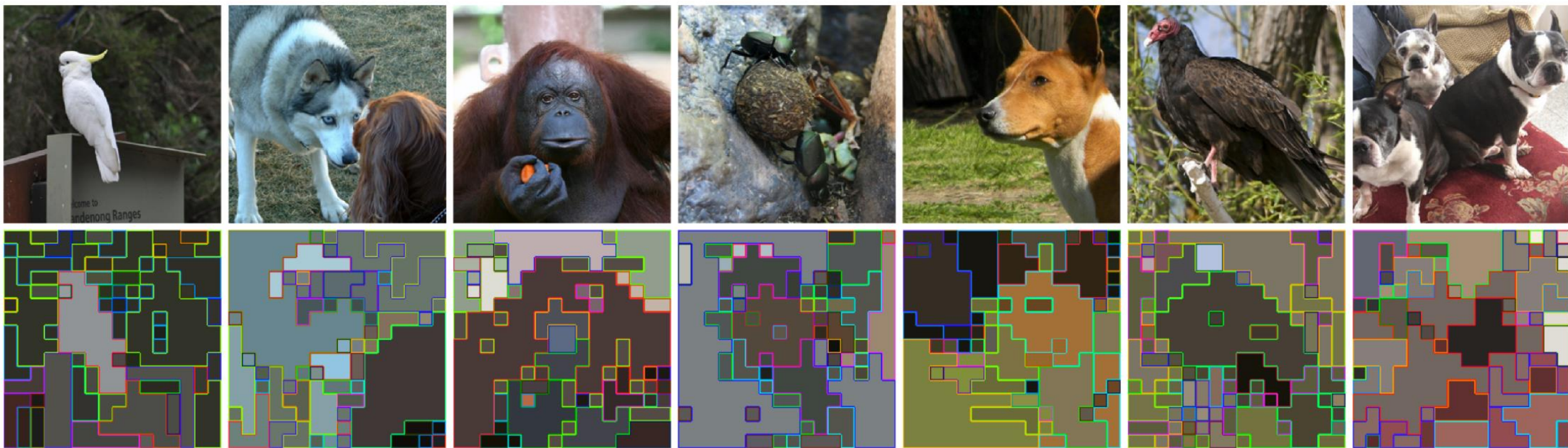


Figure 4: **Image Visualizations.** Results of merging on ImageNet-1k val using a $\text{ViT-H}_{r_7 \rightarrow}^{\text{MAE}}$ model trained with ToMe. Patches with the same inner and border color are merged together. Unlike pruning, ToMe can merge similar parts of the image whether they're in the foreground or background.

Token Merging: Your ViT But Faster

[Daniel Bolya](#), [Cheng-Yang Fu](#), [Xiaoliang Dai](#), [Peizhao Zhang](#), [Christoph Feichtenhofer](#), [Judy Hoffman](#)

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?

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

$$\mathbf{c}^*, \boldsymbol{\delta}^* = \operatorname{argmin}_{\mathbf{c}, \boldsymbol{\delta}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} (\mathbf{c}_i - \mathbf{x}_j)^2$$

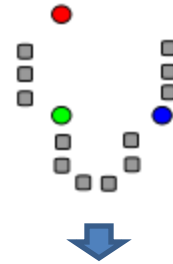
Cluster center

Data

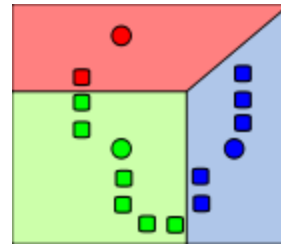
Whether $\mathbf{x}_j$ is assigned to $\mathbf{c}_i$

K-means algorithm

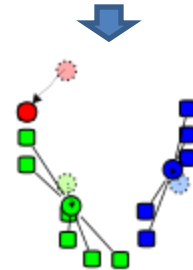
1. Randomly select K centers



2. Assign each point to nearest center

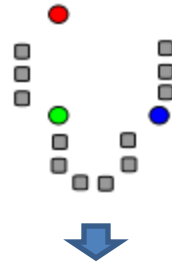


3. Compute new center (mean) for each cluster



K-means algorithm

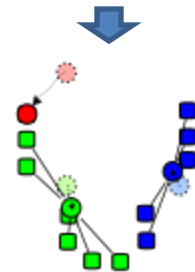
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2

K-means

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

2. Assign each point to the closest center

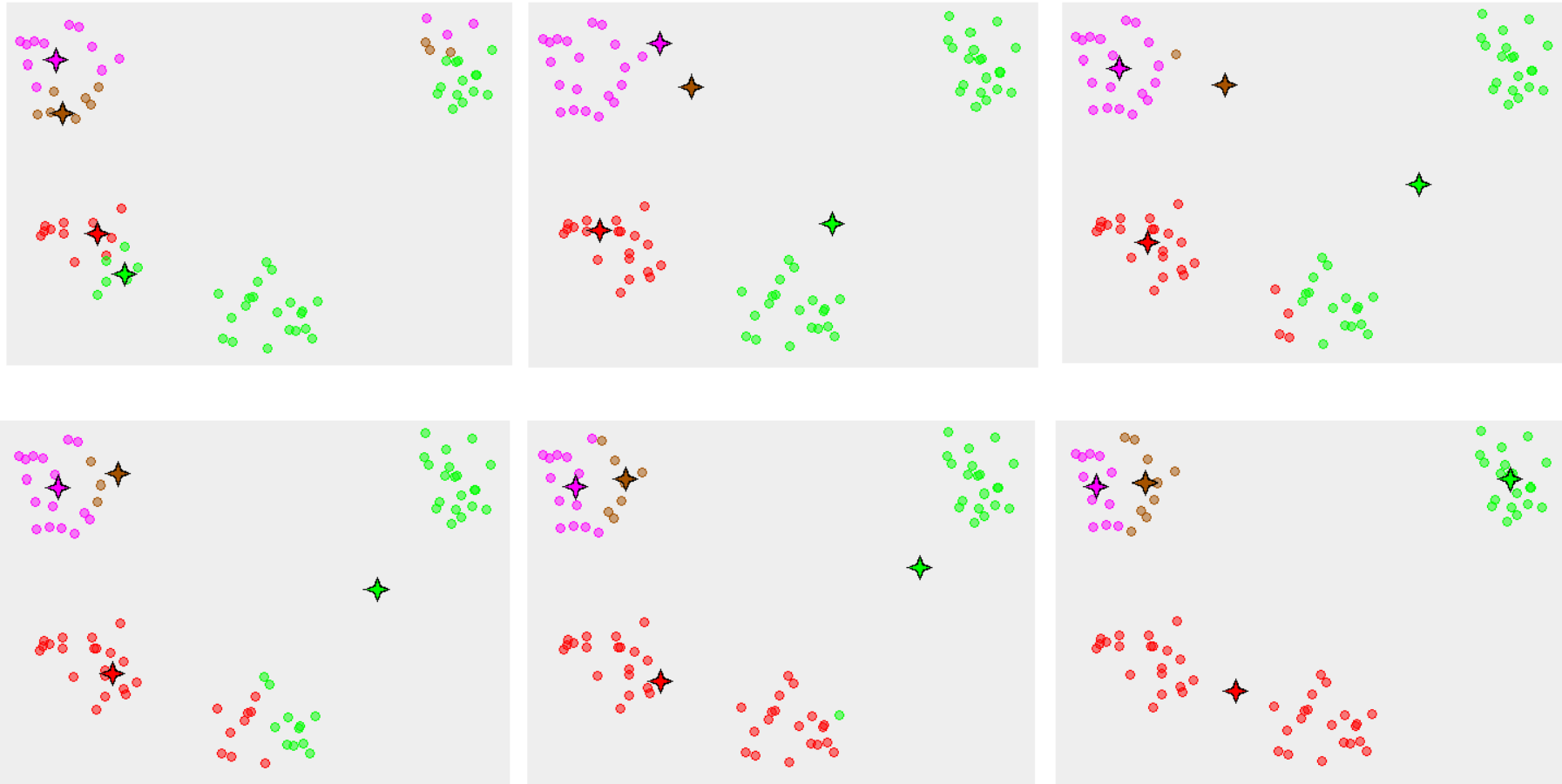
$$\delta^t = \underset{\delta}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i^{t-1} - \mathbf{x}_j \right)^2$$

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 ($t=t+1$)

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

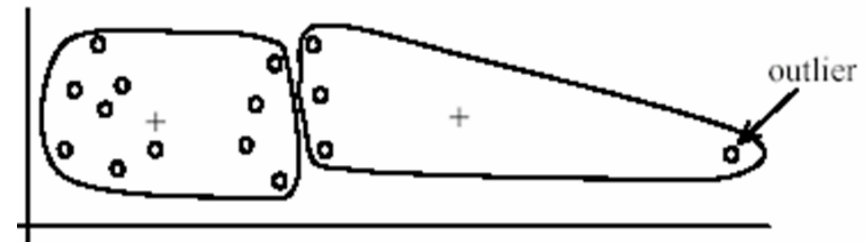


K-Means pros and cons

- 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

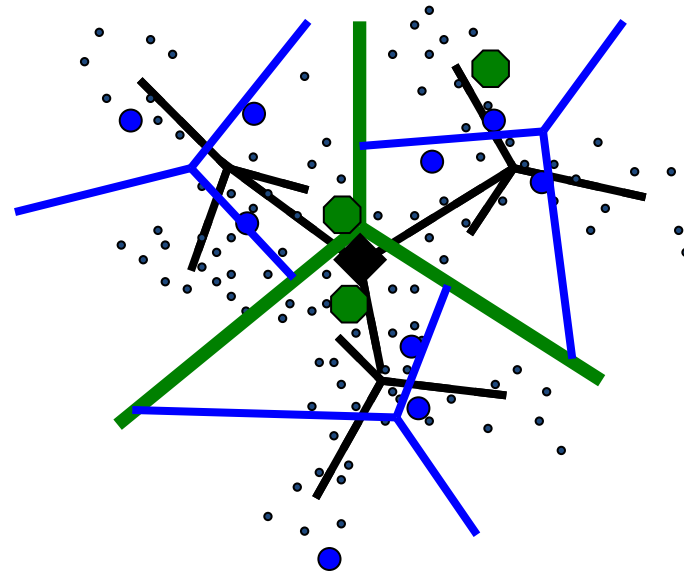


Building Visual Dictionaries

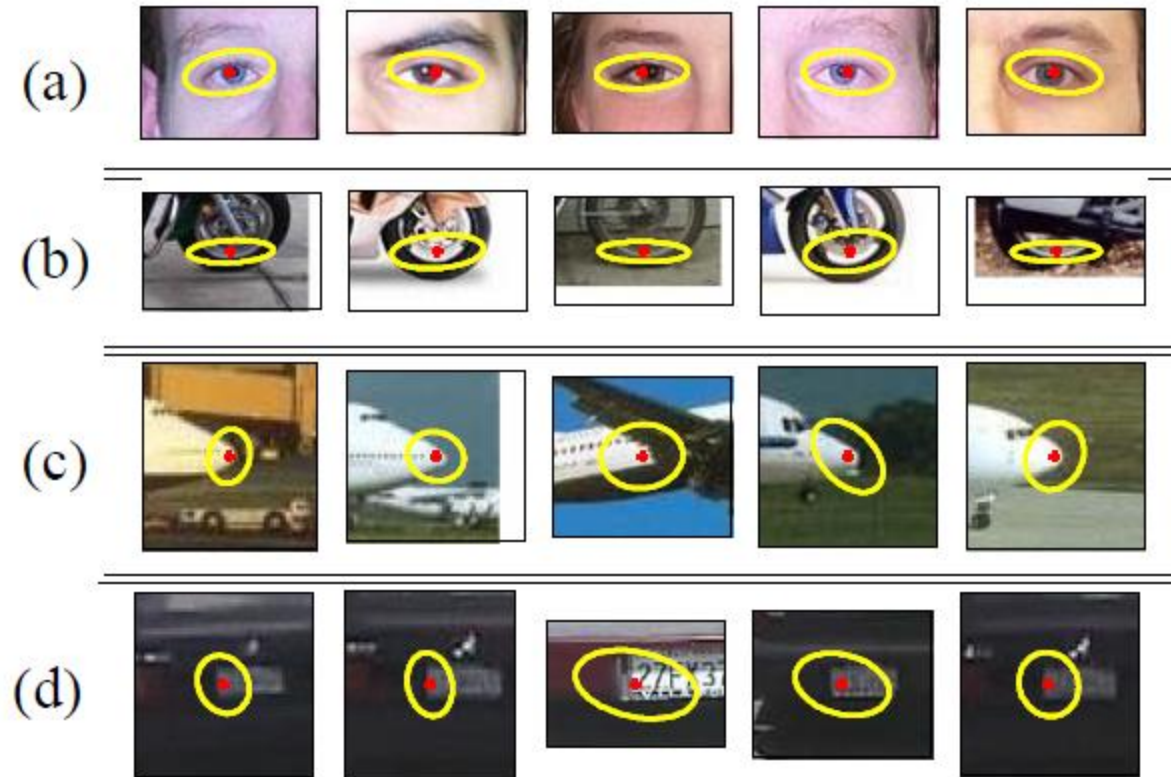
1. Sample patches from a database
 - E.g., 128 dimensional SIFT vectors



2. Cluster the patches
 - Cluster centers are the dictionary
3. Assign a codeword (number) to each new patch, according to the nearest cluster



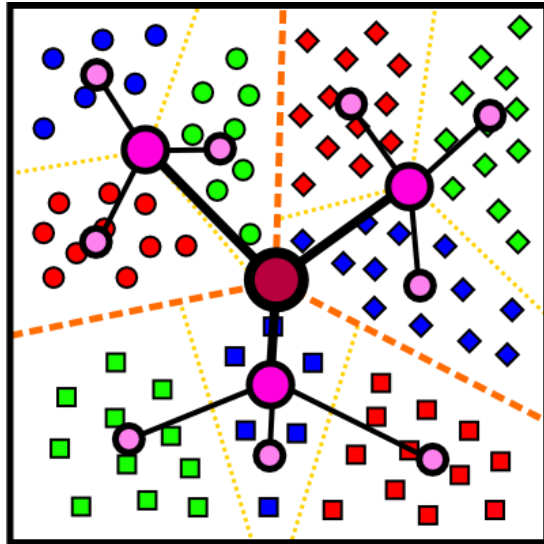
Examples of learned codewords



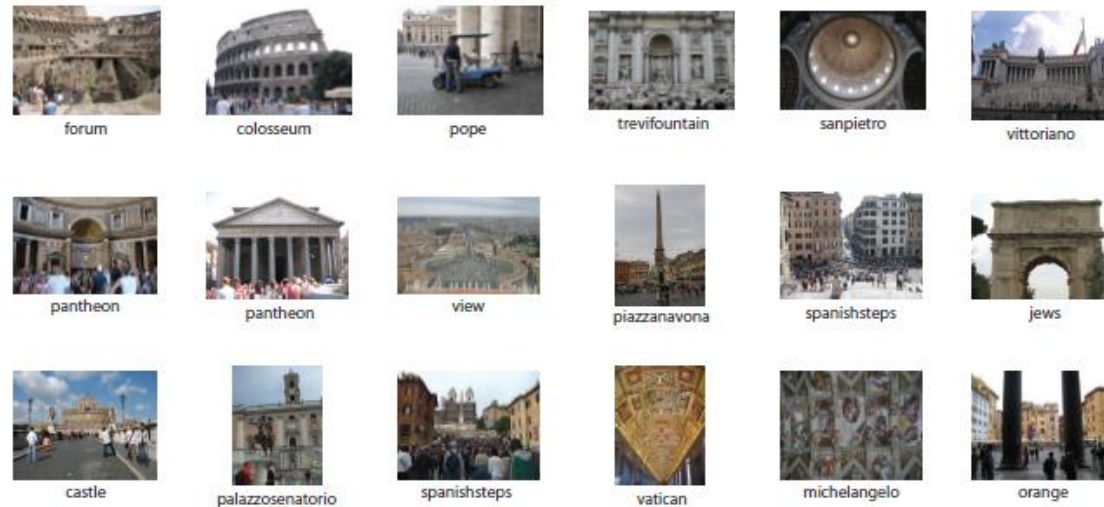
Most likely codewords for 4 learned “topics”

Which algorithm to try first?

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



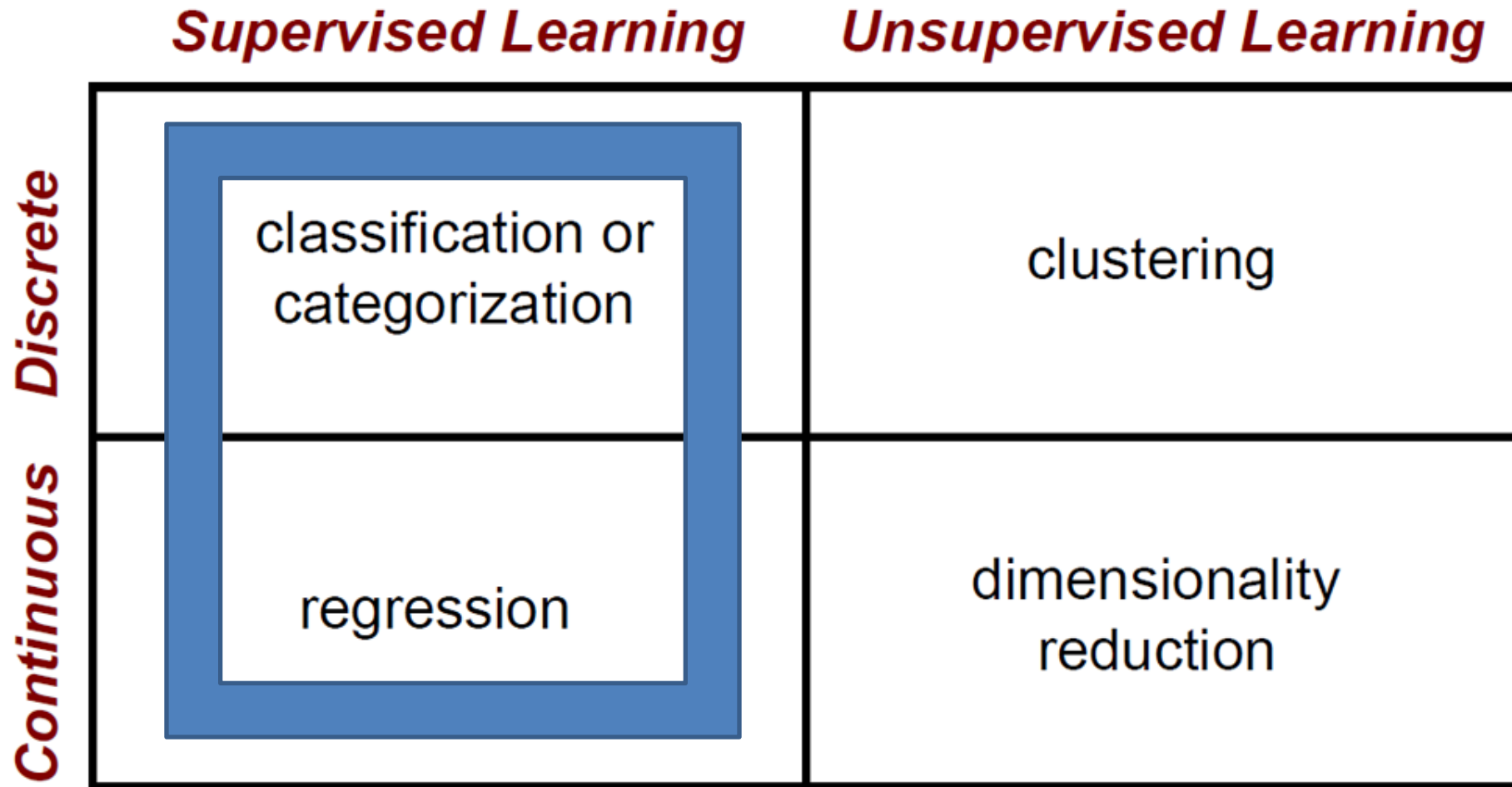
Quantization for
computing histograms



Summary of 20,000 photos of Rome using
“greedy k-means”

<http://grail.cs.washington.edu/projects/canonview/>

Machine Learning Problems



The machine learning framework

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

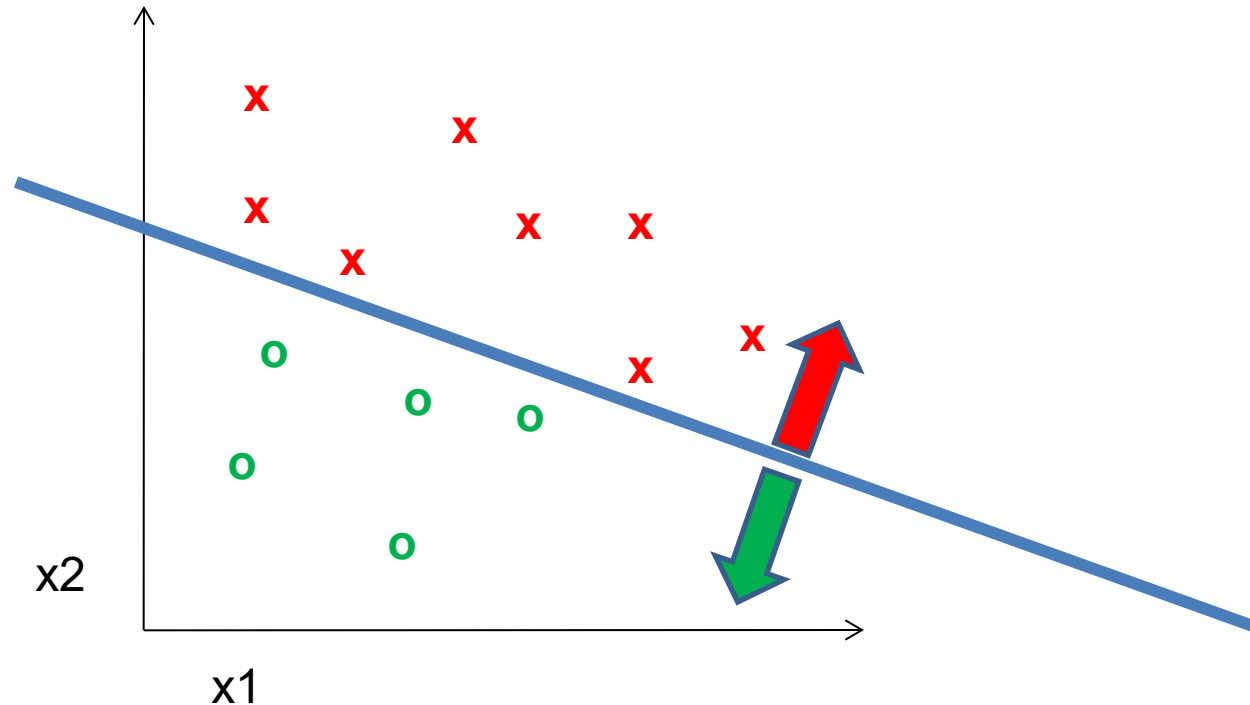
$$f(\text{apple image}) = \text{"apple"}$$

$$f(\text{tomato image}) = \text{"tomato"}$$

$$f(\text{cow image}) = \text{"cow"}$$

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)

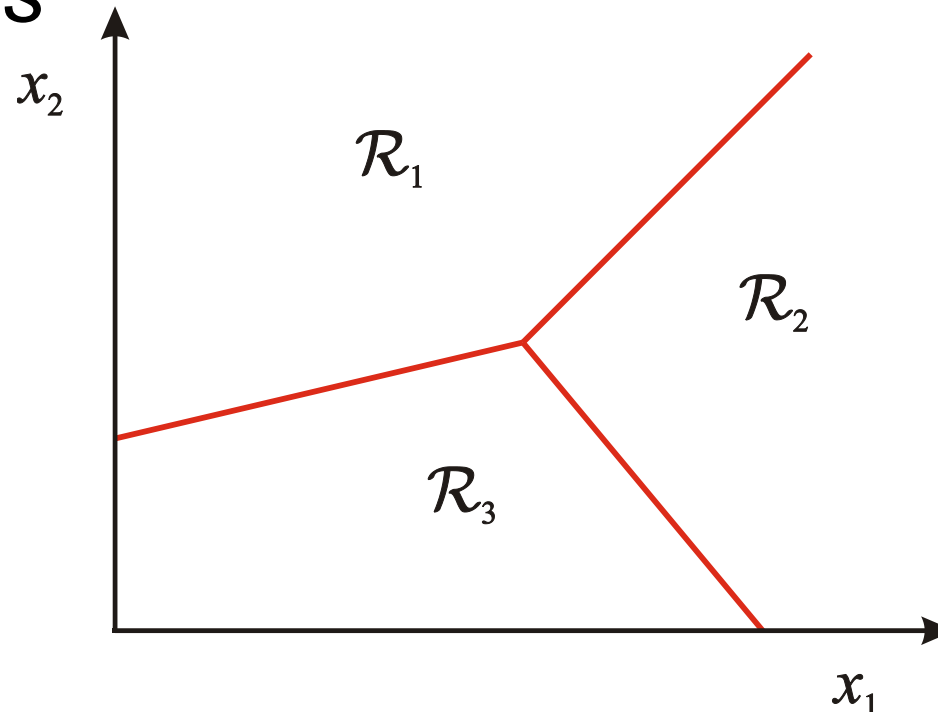
- How well does a learned model generalize from the data it was trained on to a new test set?

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.

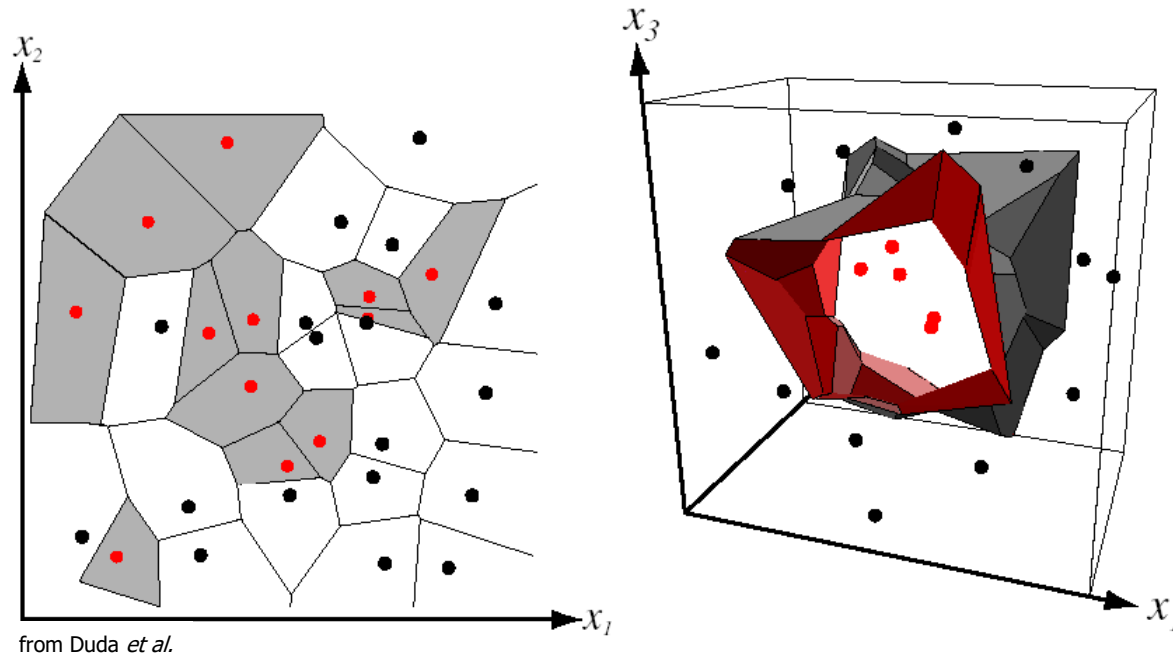
Classification

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



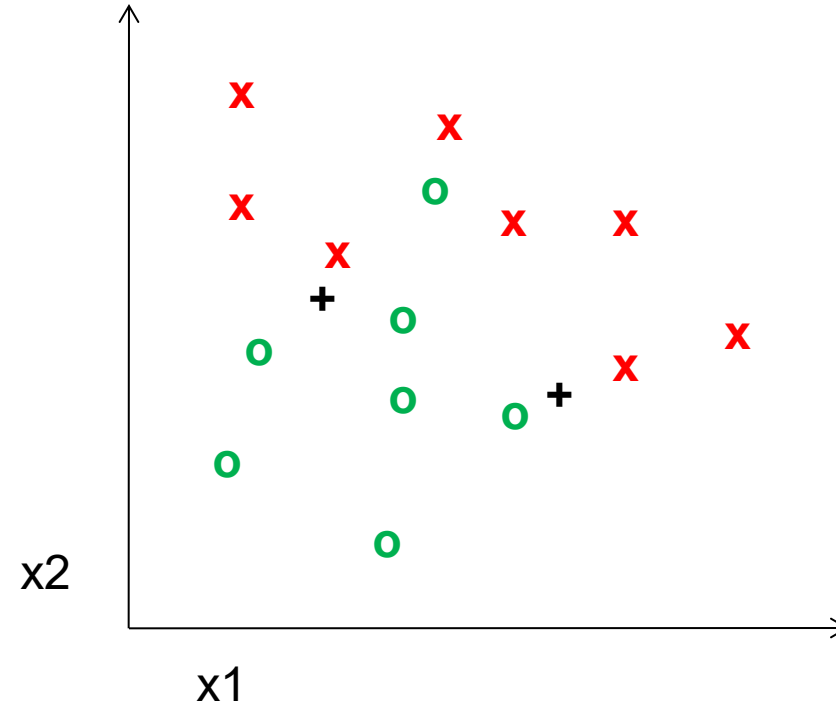
Nearest Neighbor Classifier

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

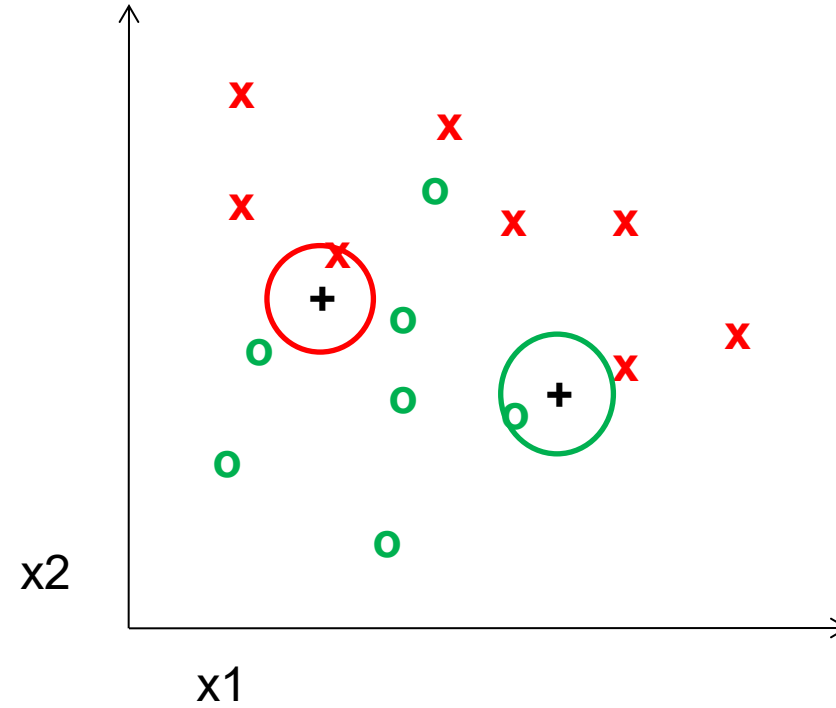


Voronoi partitioning of feature space
for two-category 2D and 3D data

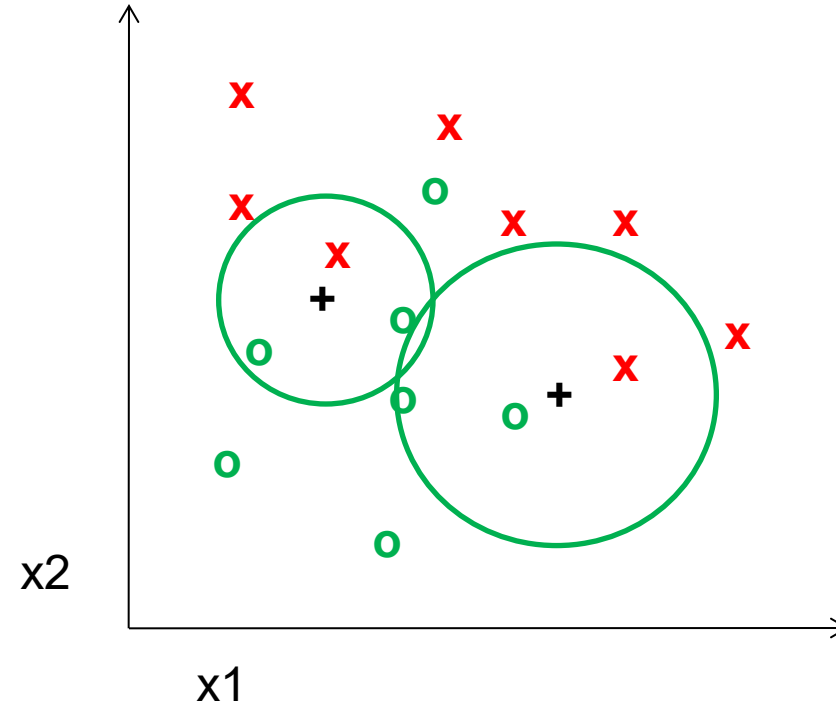
K-nearest neighbor



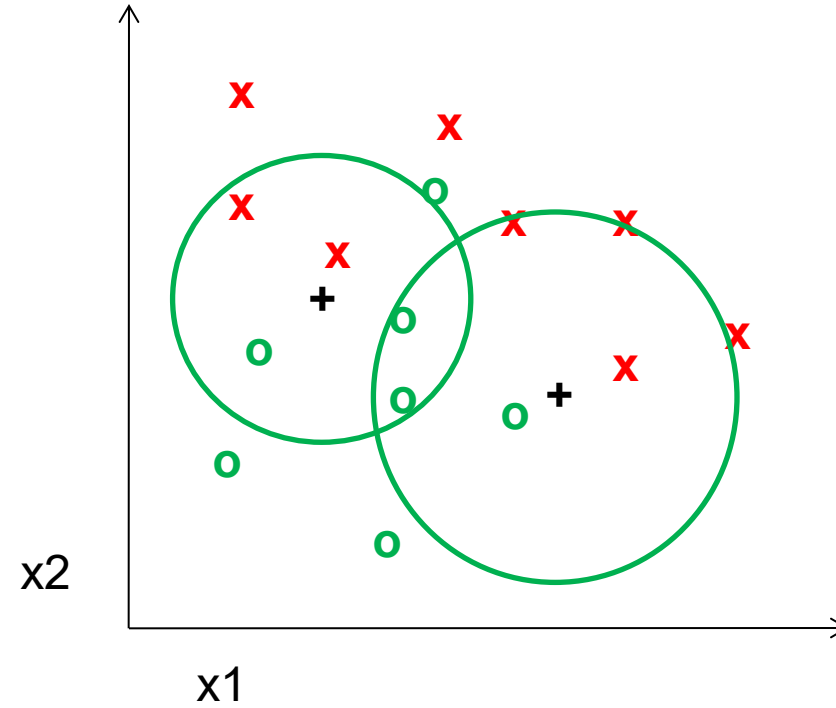
1-nearest neighbor



3-nearest neighbor



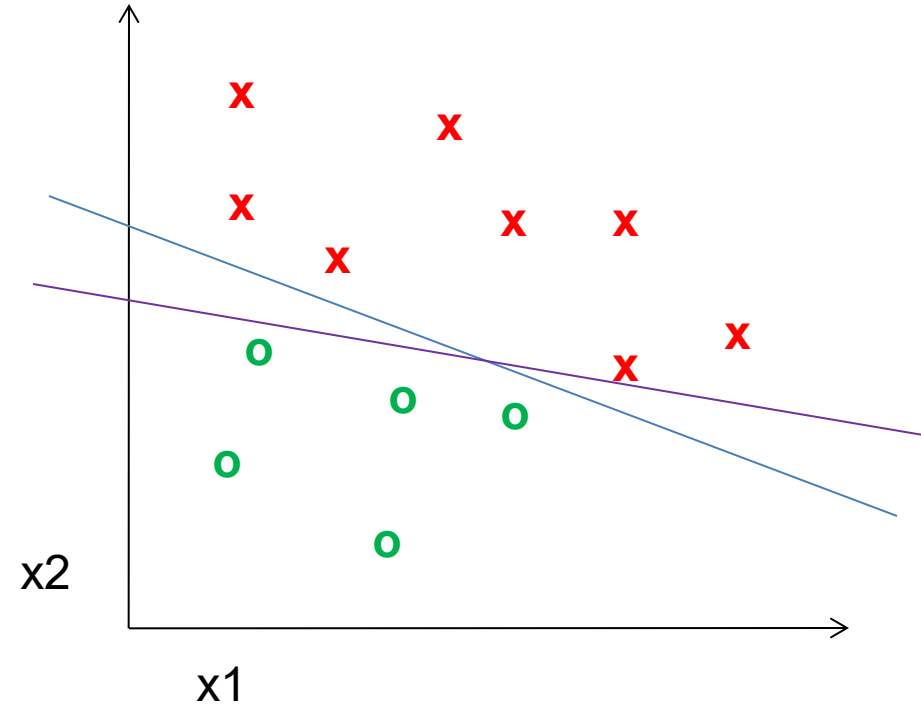
5-nearest neighbor



Using K-NN

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

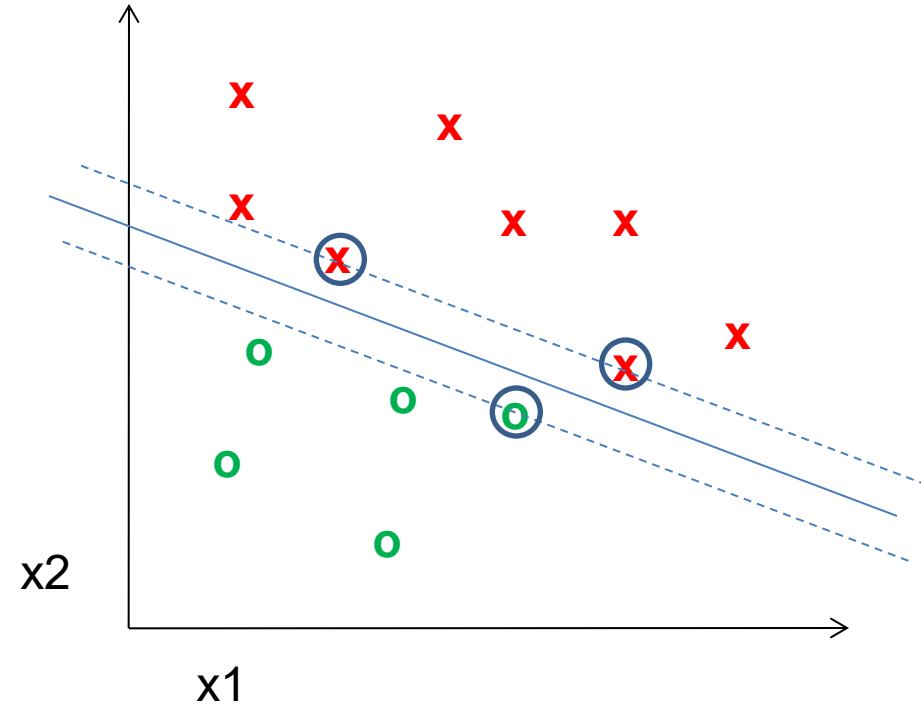
Classifiers: Linear SVM



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

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

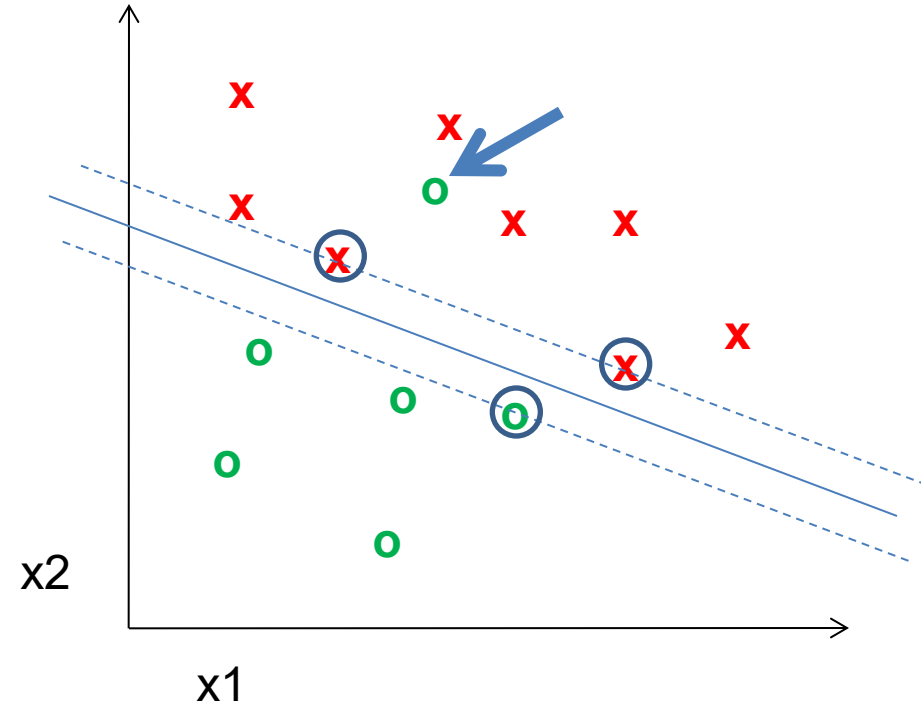
Classifiers: Linear SVM



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

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

Classifiers: Linear SVM

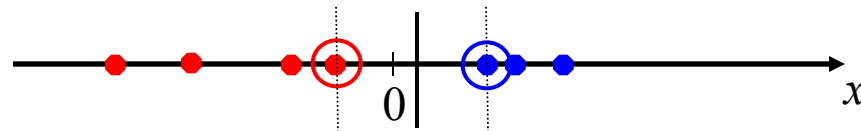


- Find a *linear function* to separate the classes:

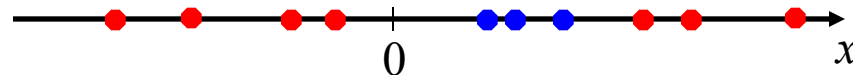
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

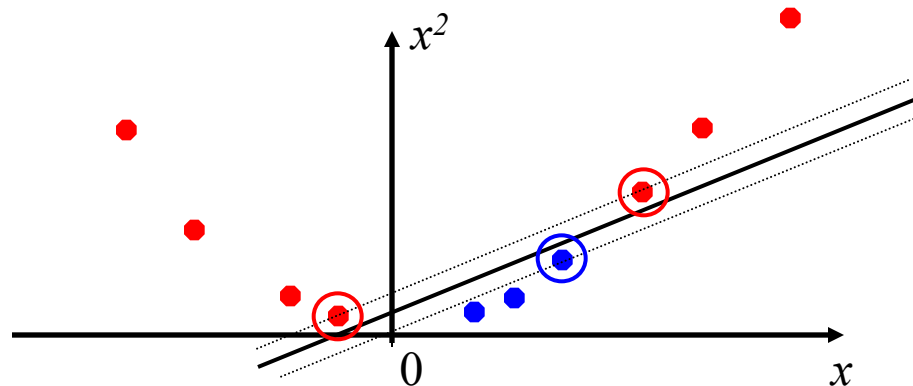
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

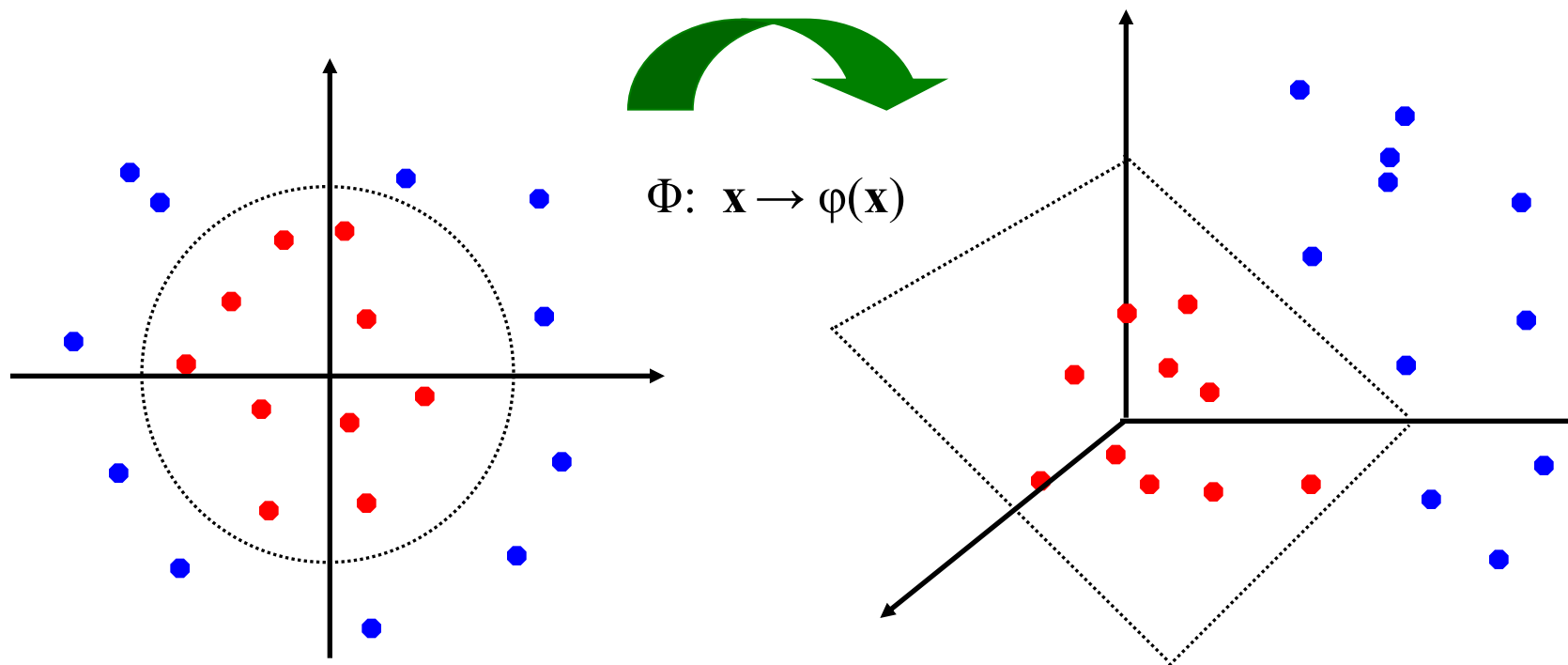


- We can map it to a higher-dimensional space:



Nonlinear SVMs

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



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:

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

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)



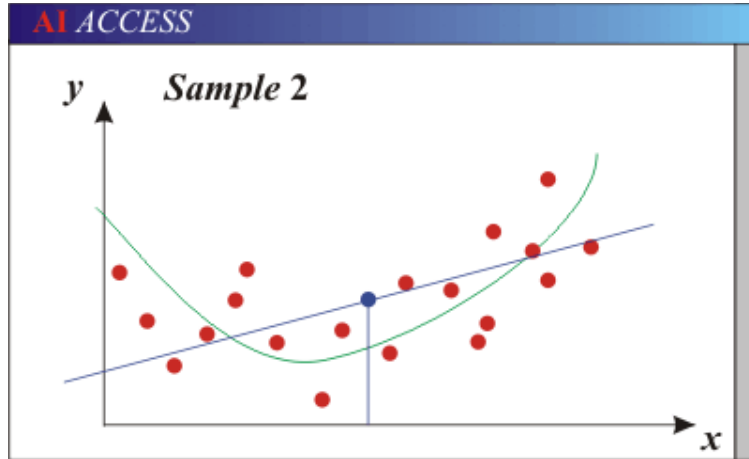
Test set (labels unknown)

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

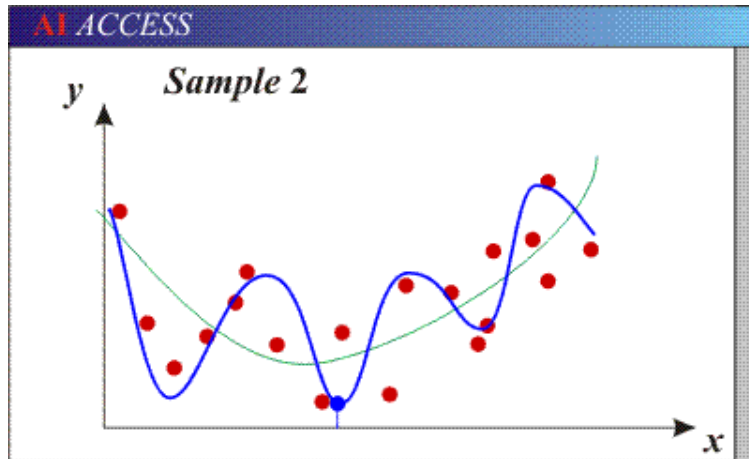
Generalization

- 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

Bias-Variance Trade-off

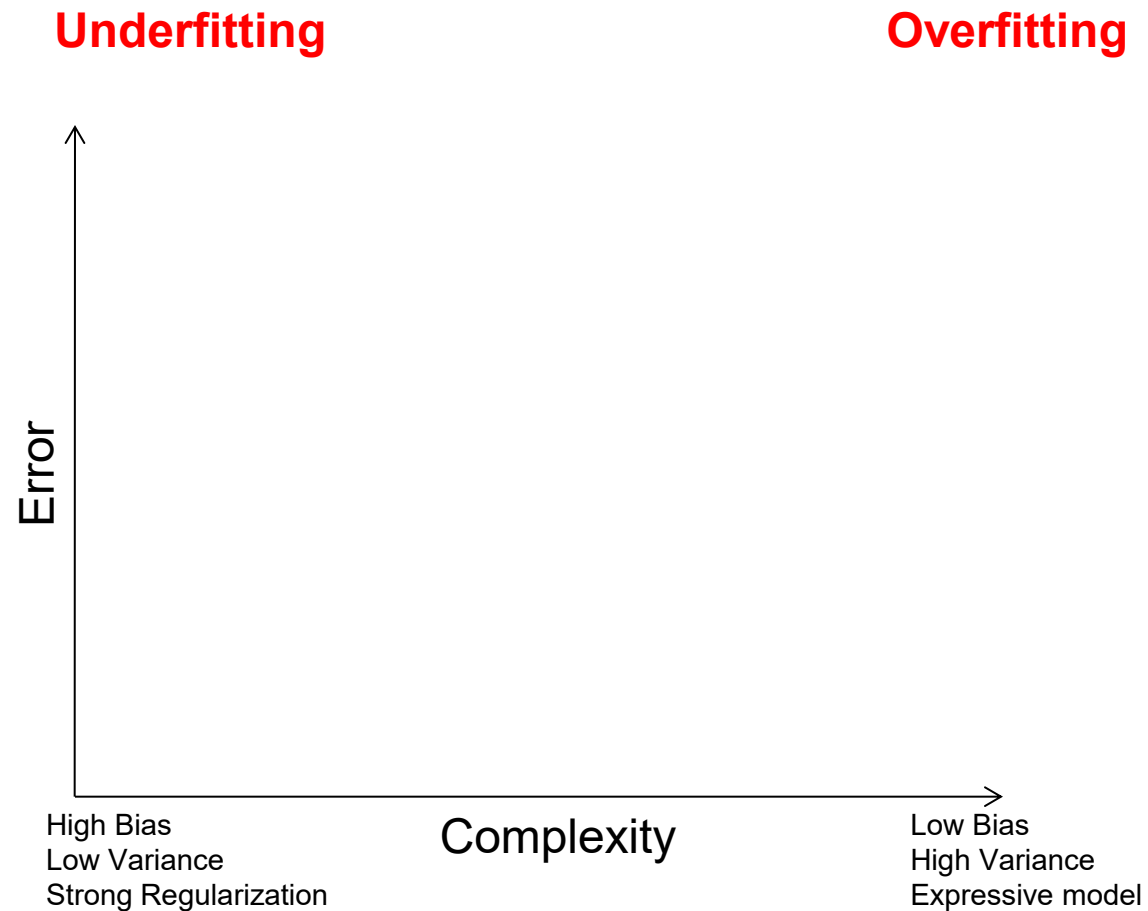


- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

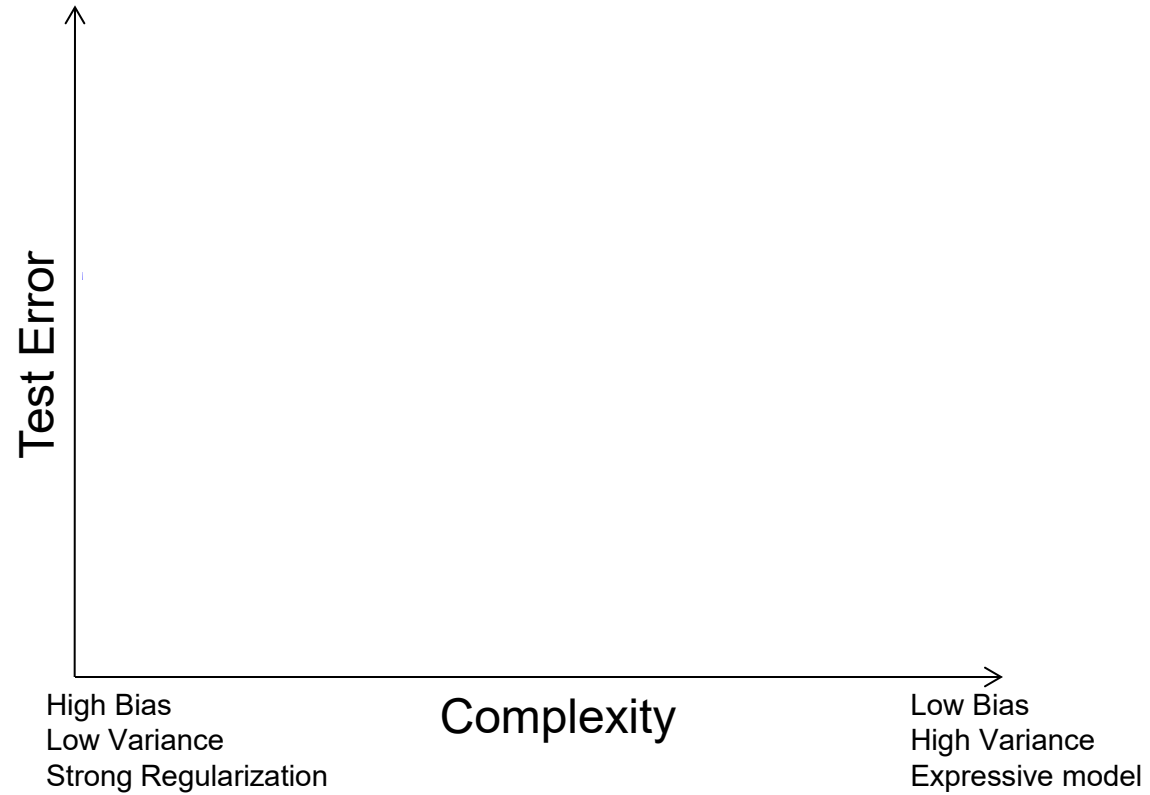


- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Bias-variance tradeoff

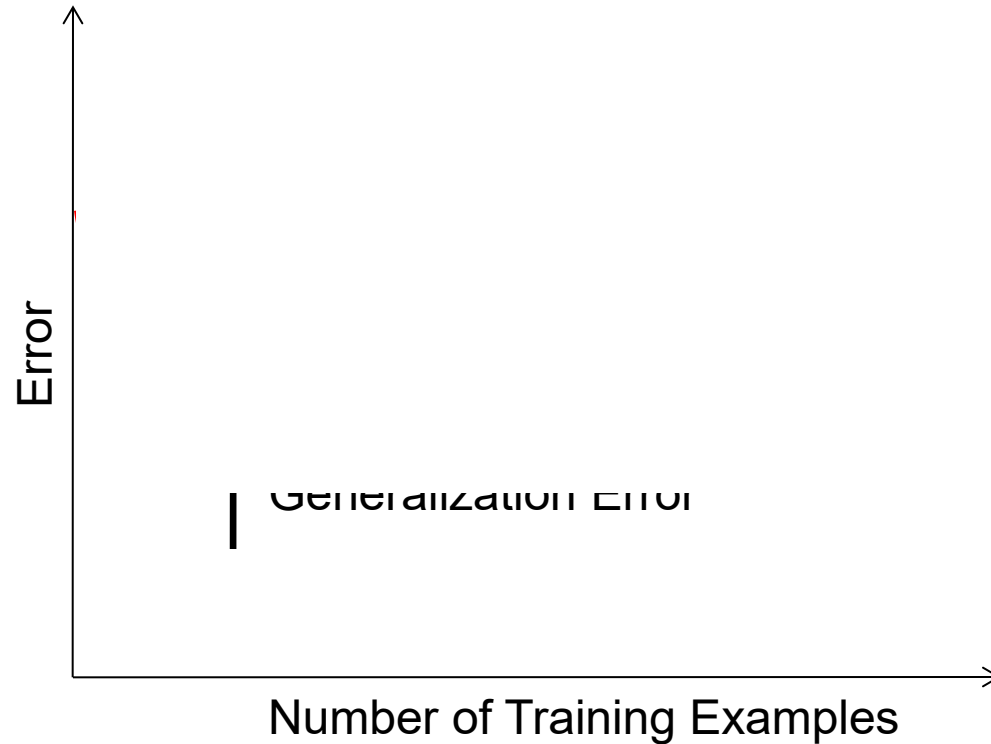


Bias-variance tradeoff



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)

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)

Machine Learning Considerations

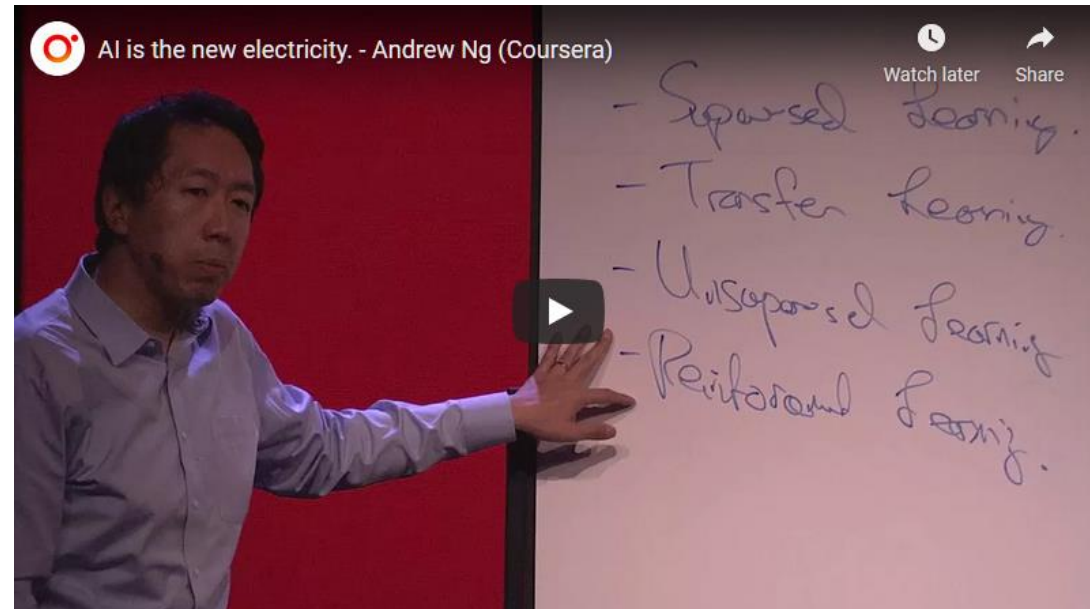
- 3 important design decisions:
 - 1) What data do I use?
 - 2) How do I represent my data (what feature)?
 - 3) What classifier / regressor / machine learning tool do I use?
- These are in decreasing order of importance
- Deep learning addresses 2 and 3 simultaneously (and blurs the boundary between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

Machine Learning Problems

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	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