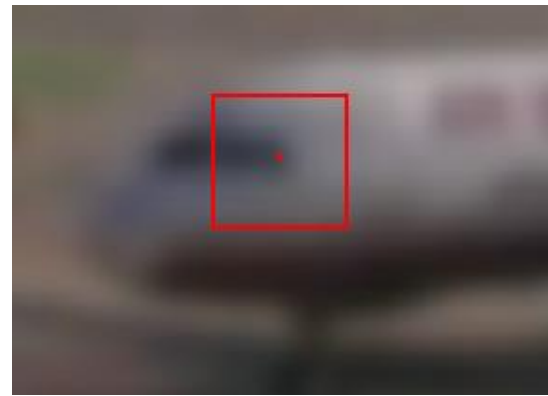


Recap: Optical Flow

Brightness constancy constraint equation

$$I_x u + I_y v + I_t = 0$$

- What do the static image gradients have to do with motion estimation?



If I told you
 I_t is -5
 I_x is 2
 I_y is 1

What was
the pixel
shift (u,v) ?

Recap: Conditions for solvability

Optimal (u, v) satisfies Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $A^T b$

When is this solvable? I.e., what are good points to track?

- $A^T A$ should be invertible
- $A^T A$ should not be too small due to noise
 - eigenvalues λ_1 and λ_2 of $A^T A$ should not be too small
- $A^T A$ should be well-conditioned
 - λ_1 / λ_2 should not be too large ($\lambda_1 =$ larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

Machine Learning Crash Course



Photo: CMU Machine Learning
Department protests G20

Computer Vision
James Hays

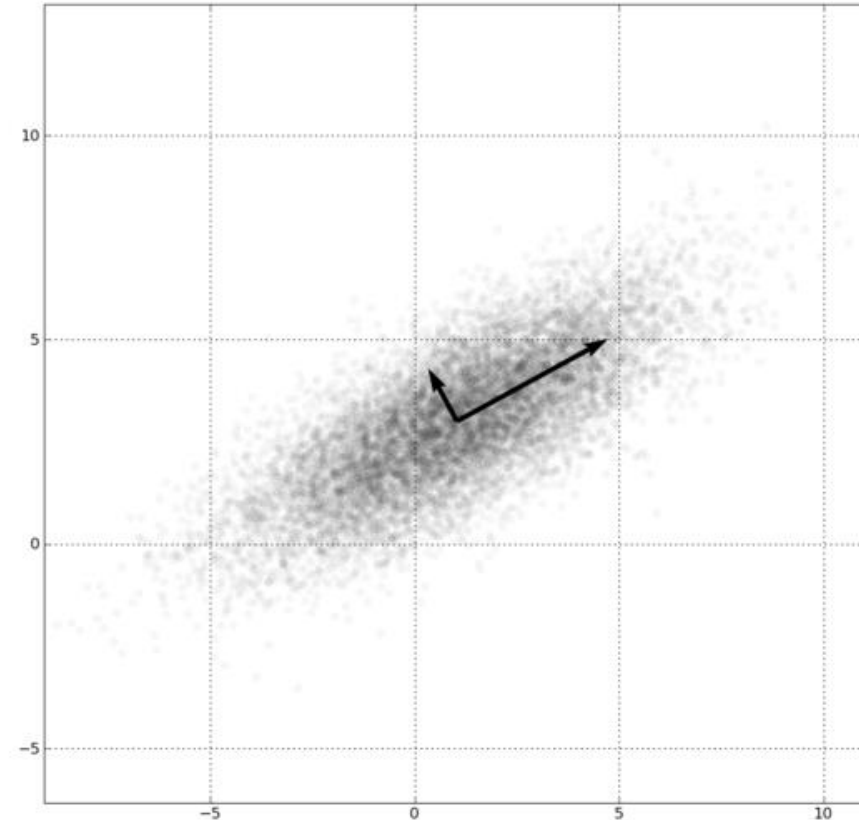
Slides: Isabelle Guyon,
Erik Sudderth,
Mark Johnson,
Derek Hoiem

Machine Learning Problems

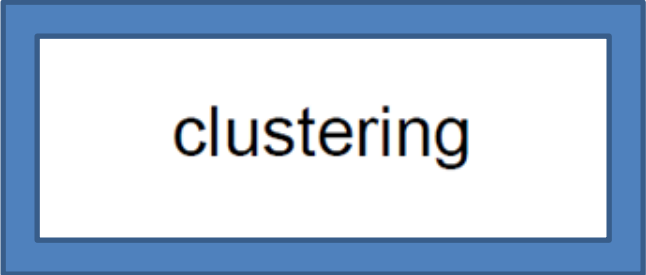
	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

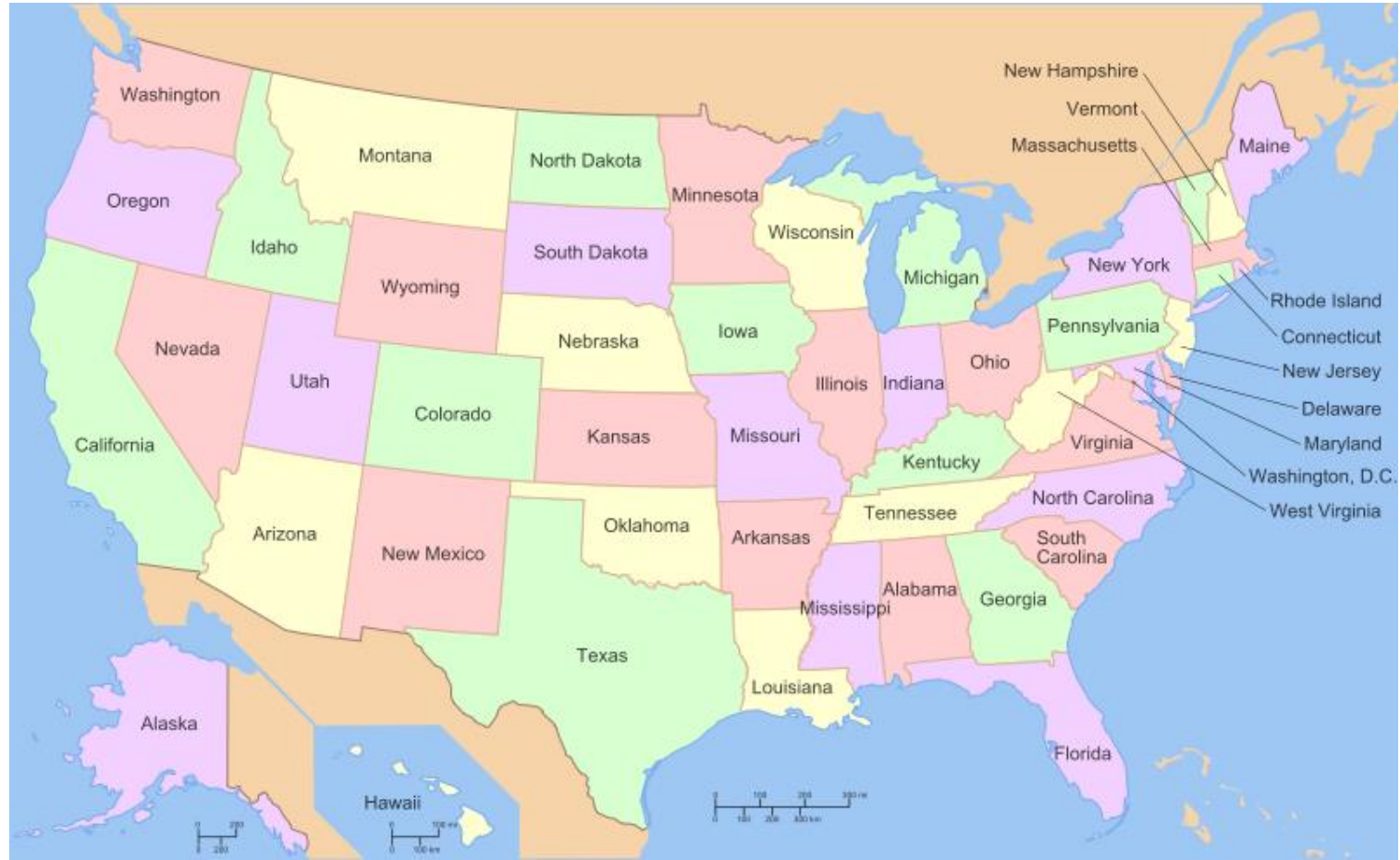
Dimensionality Reduction

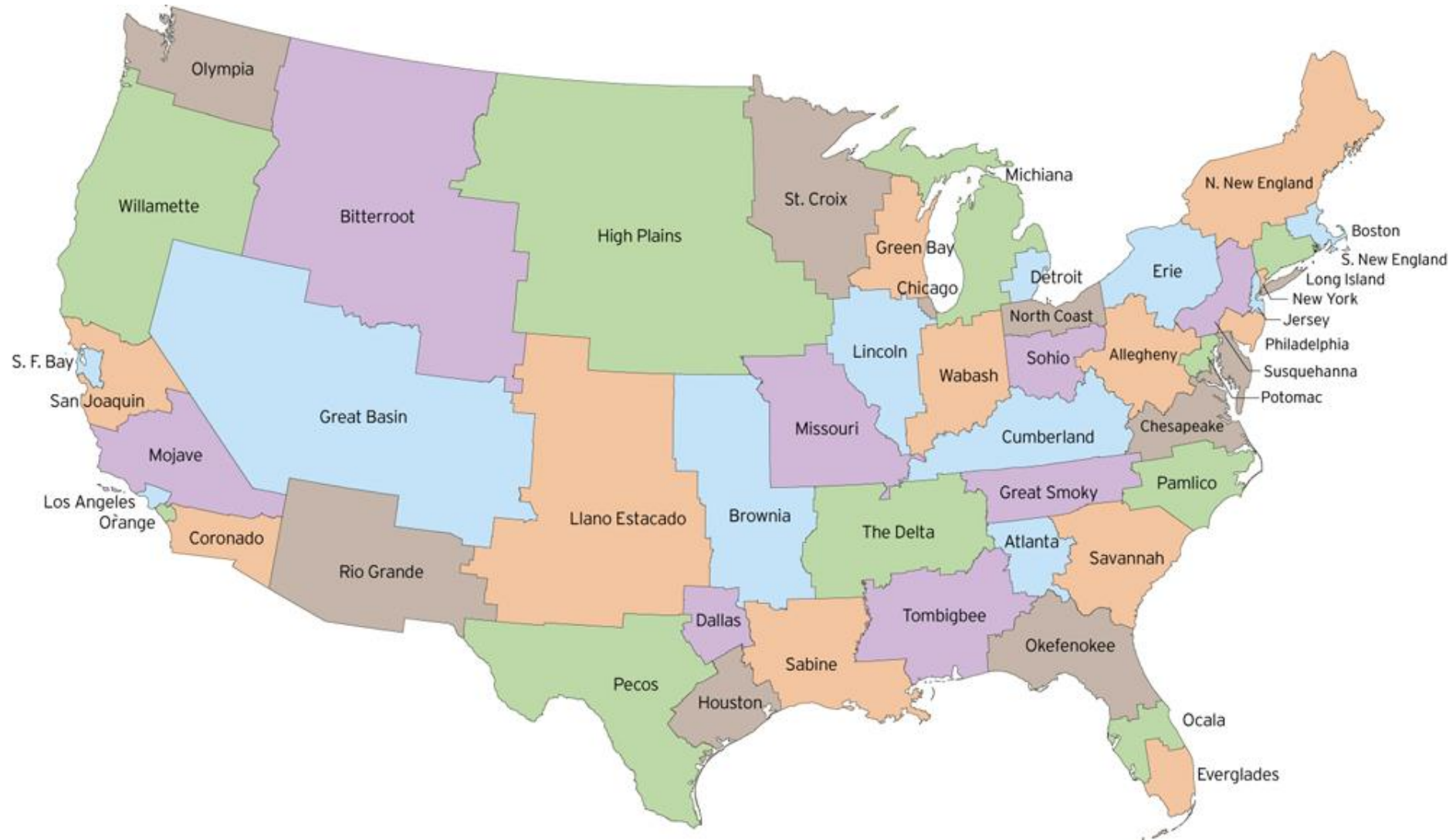
- **PCA, ICA, LLE, Isomap, Autoencoder**
- PCA is the most important technique to know. It takes advantage of correlations in data dimensions to produce the best possible lower dimensional representation based on linear projections (minimizes reconstruction error).
- PCA should be used for dimensionality reduction, not for discovering patterns or making predictions. Don't try to assign semantic meaning to the bases.



Machine Learning Problems

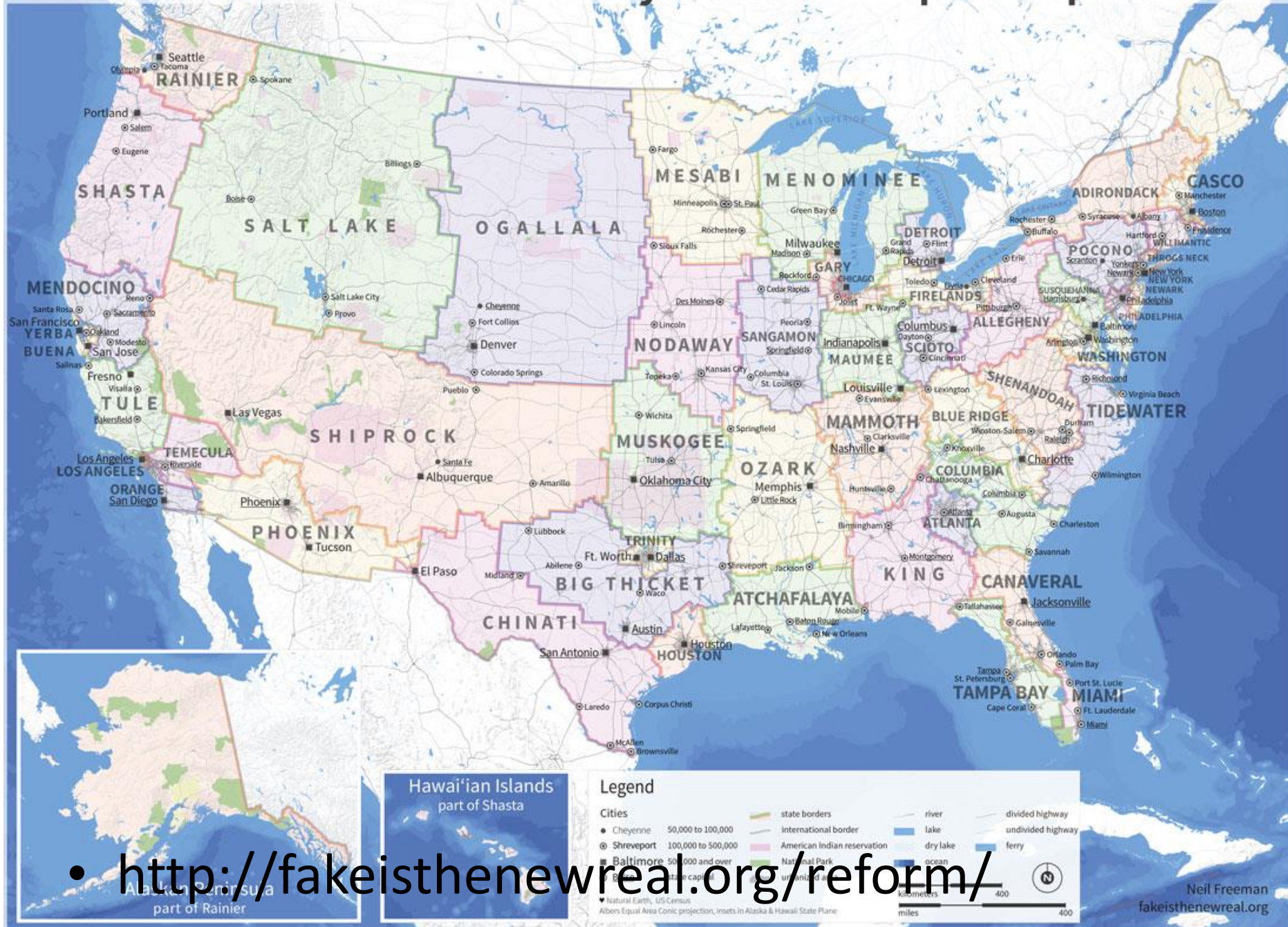
	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	 clustering
<i>Continuous</i>	regression	dimensionality reduction





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

The United States redrawn as Fifty States with Equal Population



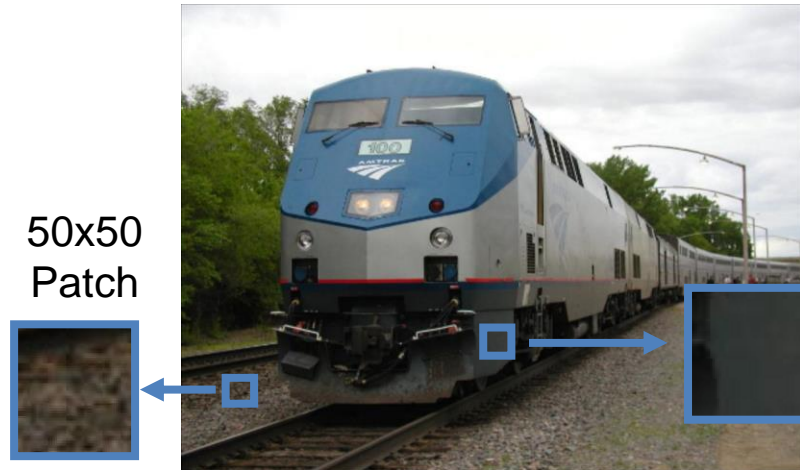
• <http://fakeisthenewreal.org/reform/>

Clustering example: image segmentation

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



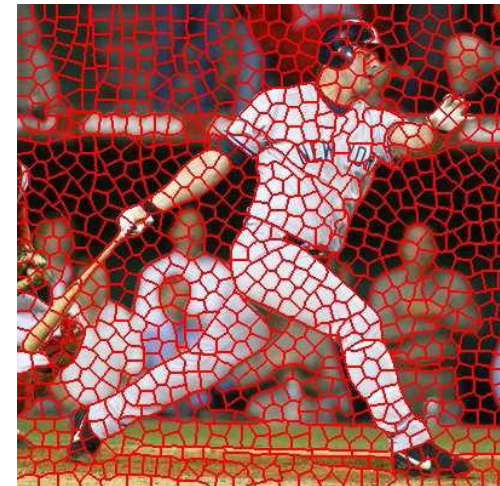
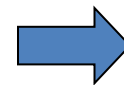
Segmentation for feature support or efficiency



50x50 Patch



[Felzenszwalb and Huttenlocher 2004]



[Shi and Malik 2001]

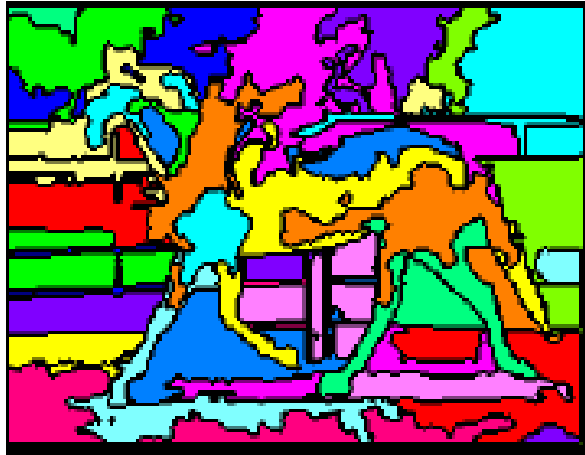
[Hoiem et al. 2005, Mori 2005]

Slide: Derek Hoiem

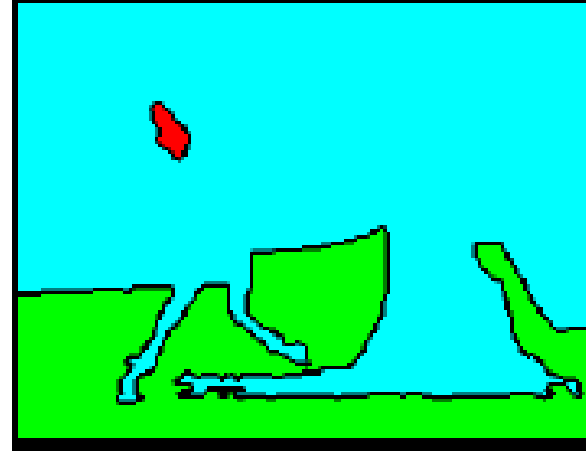
Segmentation as a result



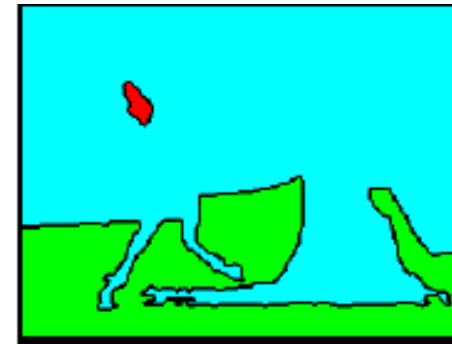
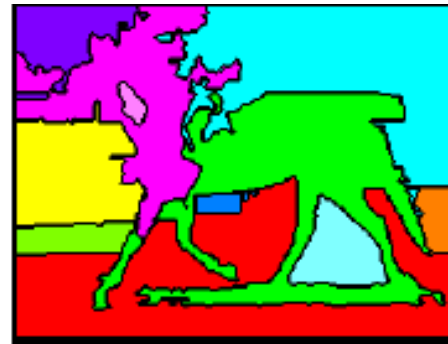
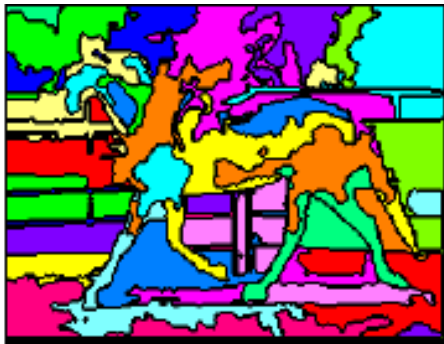
Types of segmentations



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?

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

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

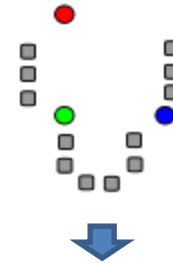
Cluster center

Data

Whether x_j is assigned to c_i

K-means algorithm

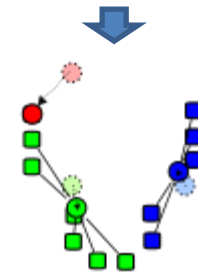
1. Randomly select K centers



2. Assign each point to nearest center

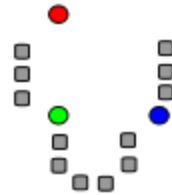


3. Compute new center (mean) for each cluster



K-means algorithm

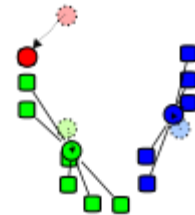
1. Randomly select K centers



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

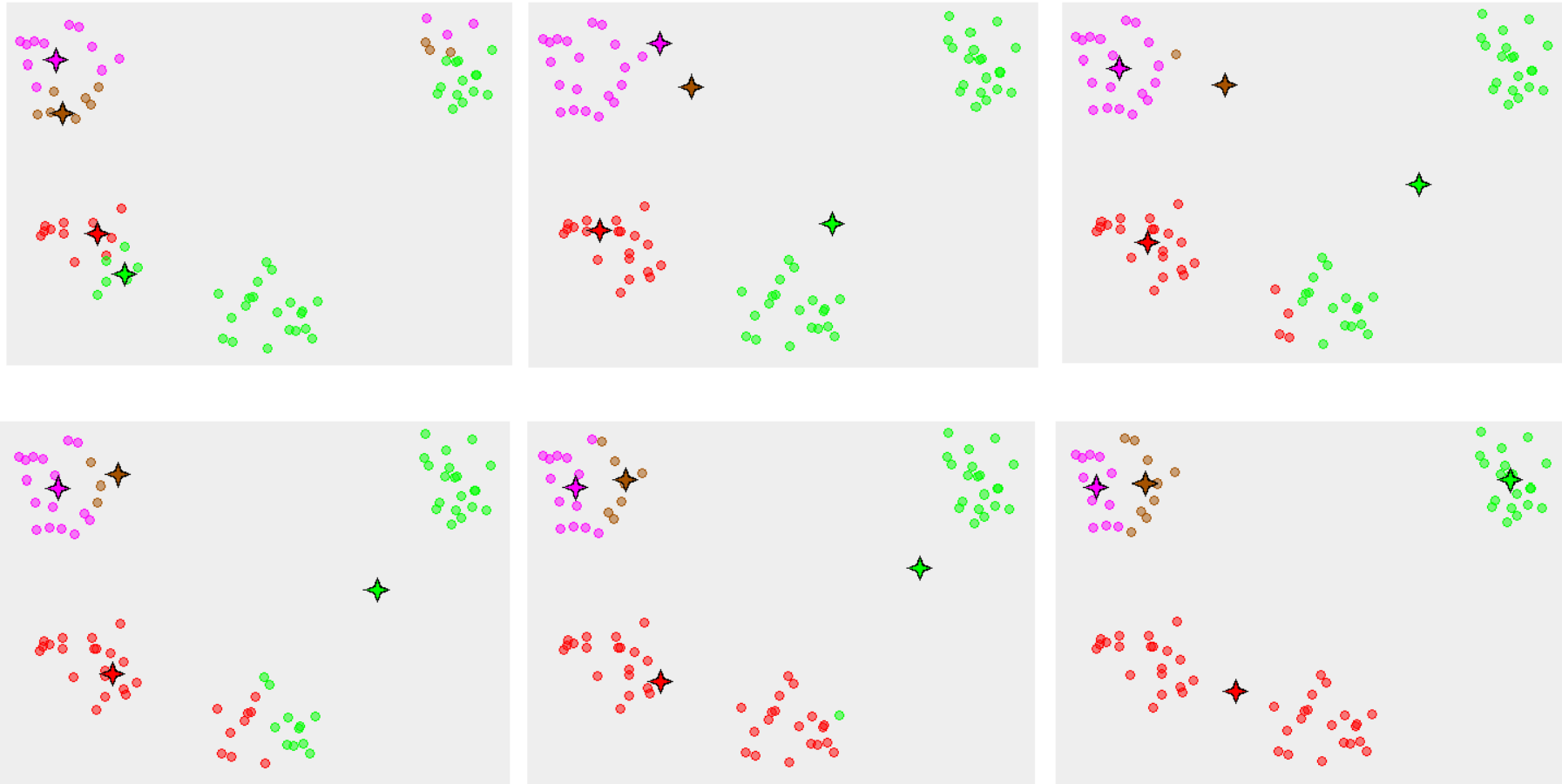
$$\delta^t = \underset{\delta}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i^{t-1} - \mathbf{x}_j \right)^2$$

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

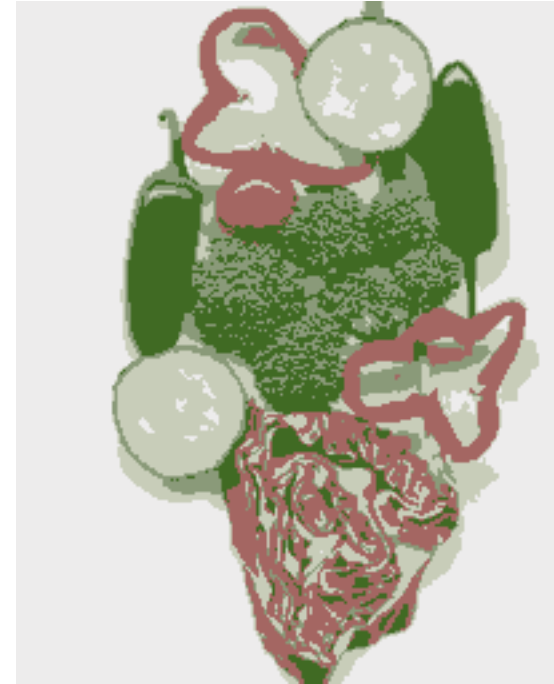
Image



Clusters on intensity



Clusters on color



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

How to choose the number of clusters?

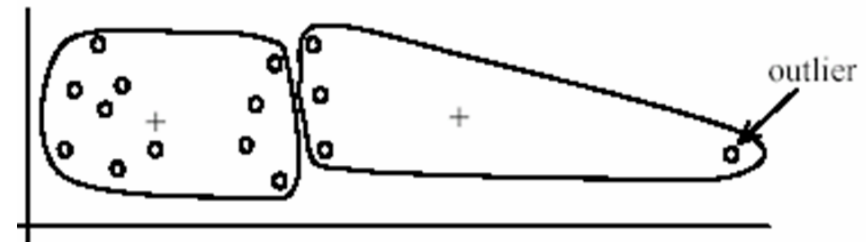
- Validation set
 - Try different numbers of clusters and look at performance
 - When building dictionaries (discussed later), more clusters typically work better

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



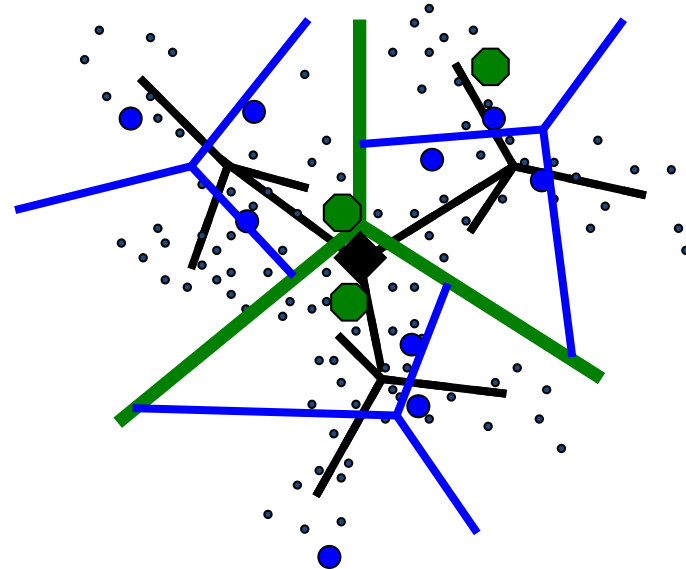
Building Visual Dictionaries

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

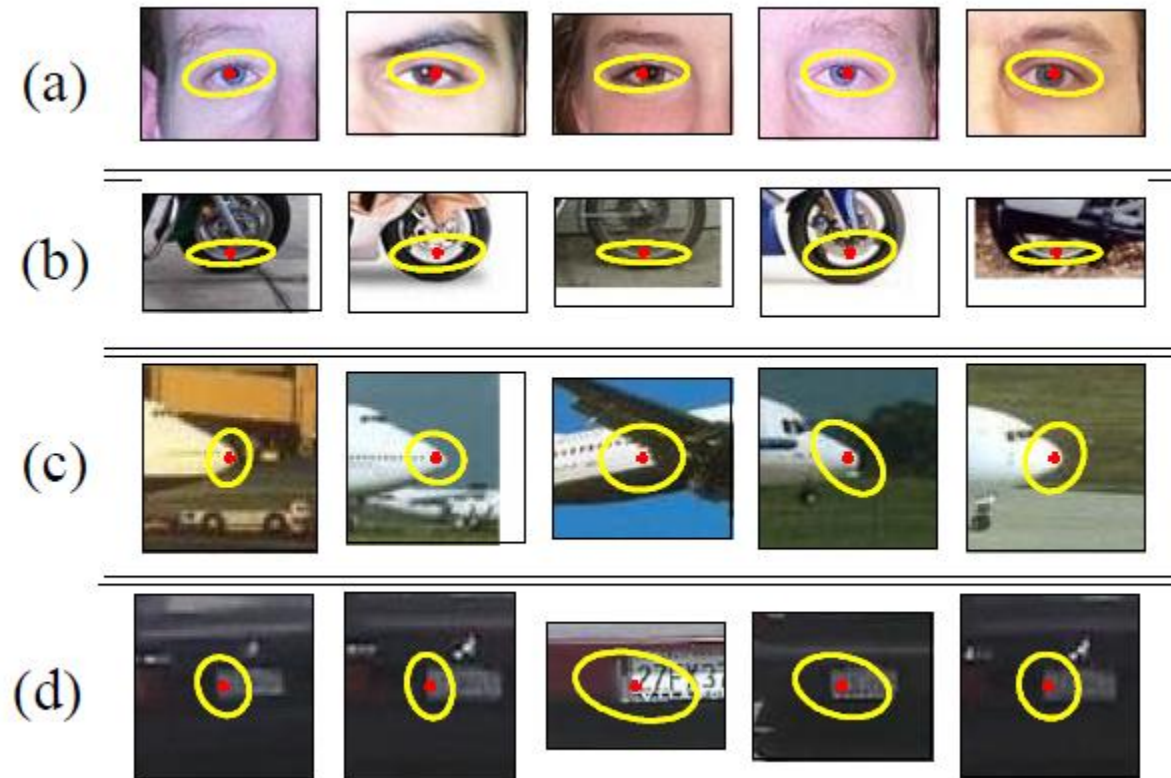


2. Cluster the patches
 - Cluster centers are the dictionary

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



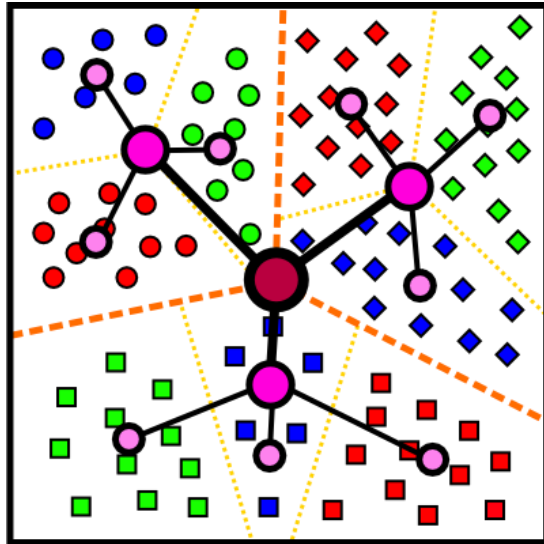
Examples of learned codewords



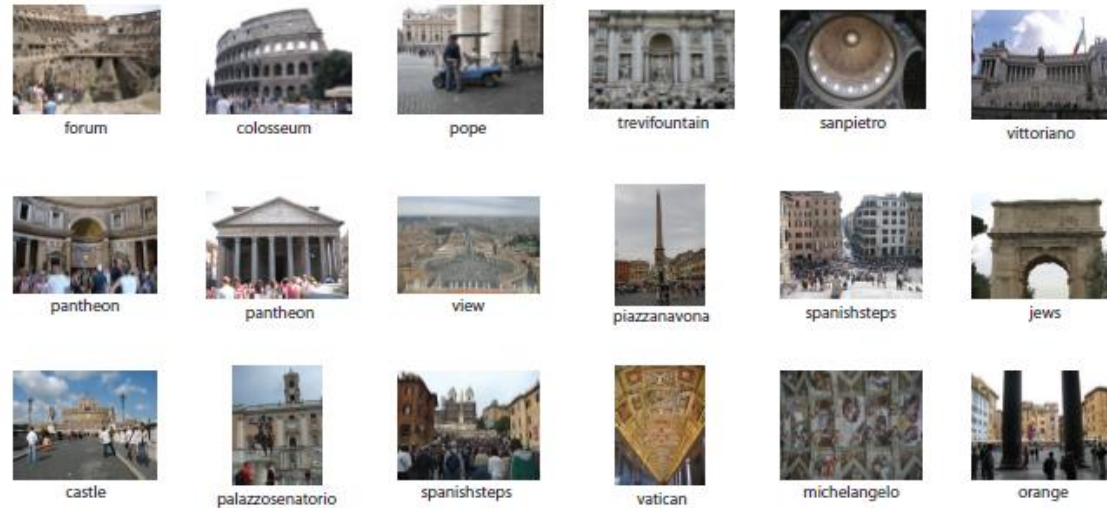
Most likely codewords for 4 learned “topics”

Which algorithm to try first?

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



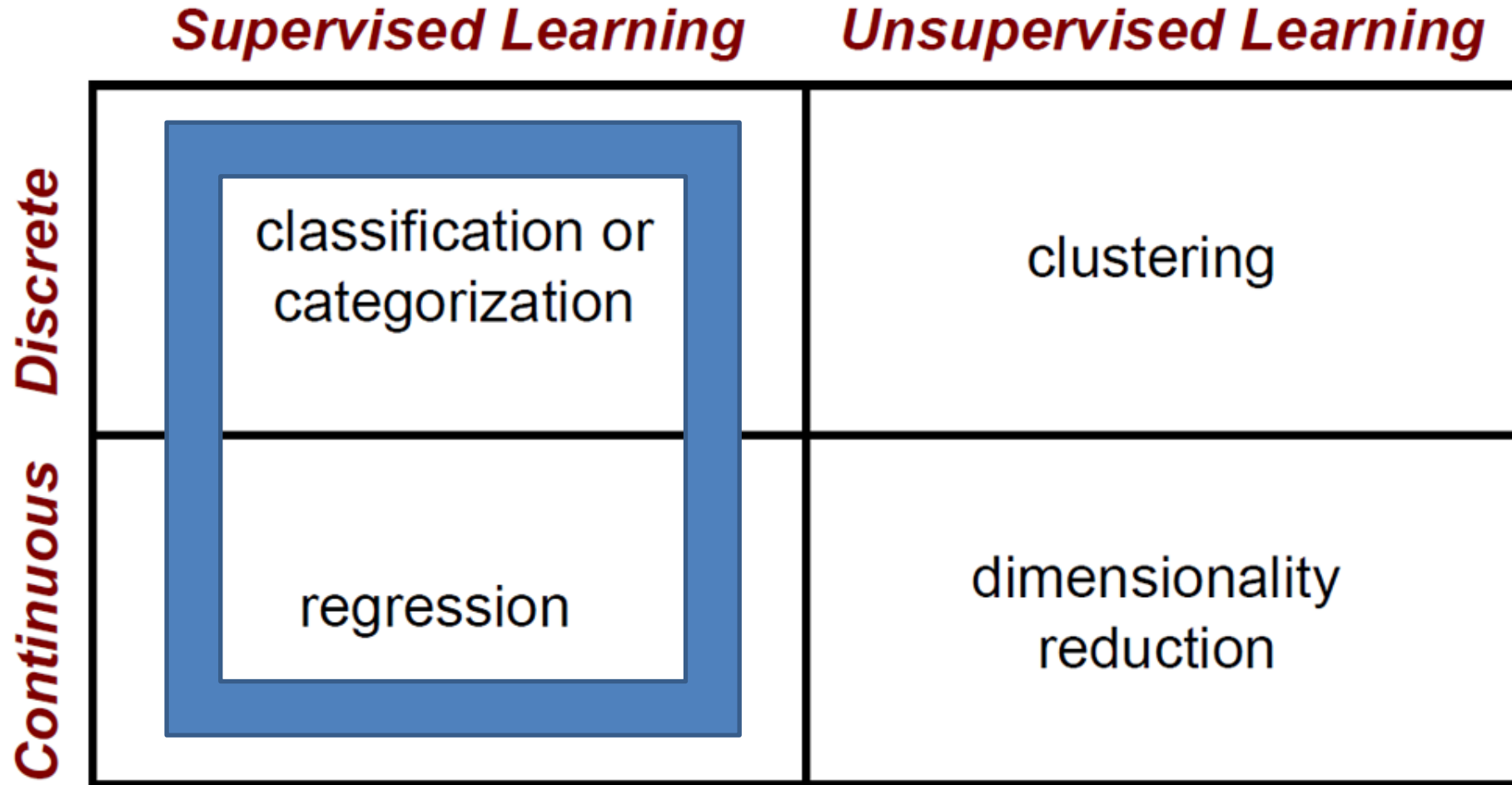
Quantization for
computing histograms



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:

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

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

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

The machine learning framework

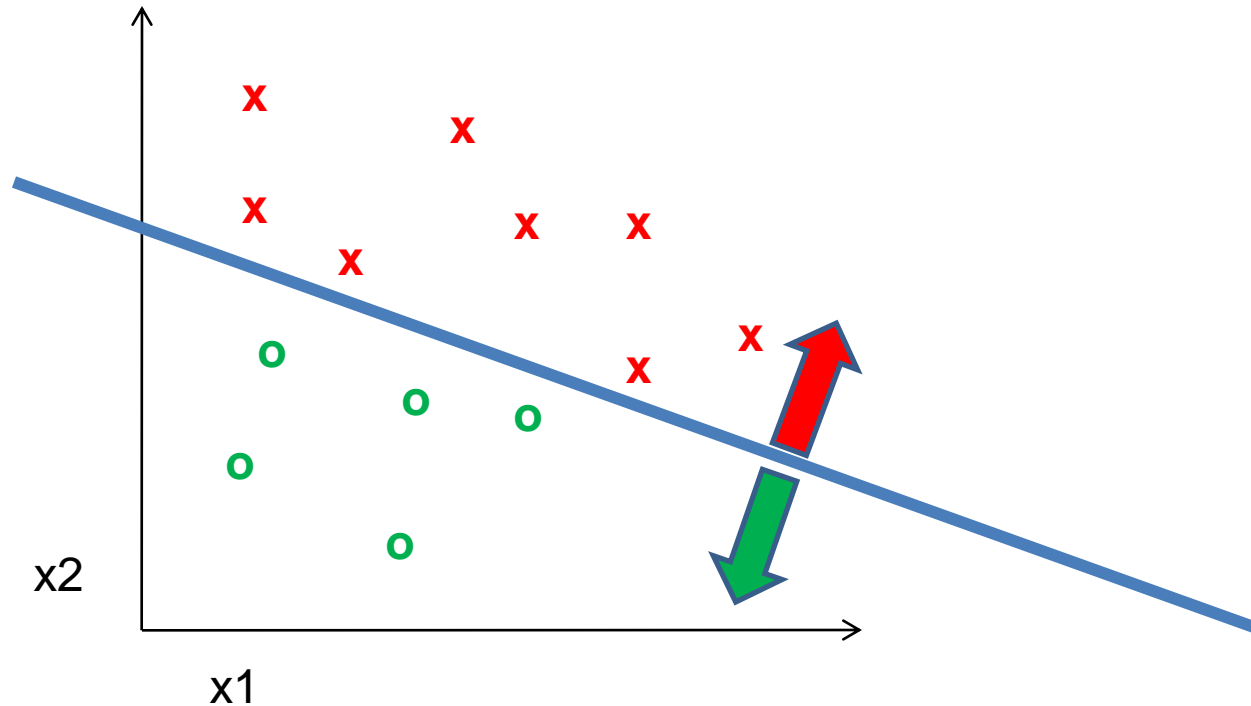
$$y = f(\mathbf{x})$$

output prediction function Image feature

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* \mathbf{x} and output the predicted value $y = f(\mathbf{x})$

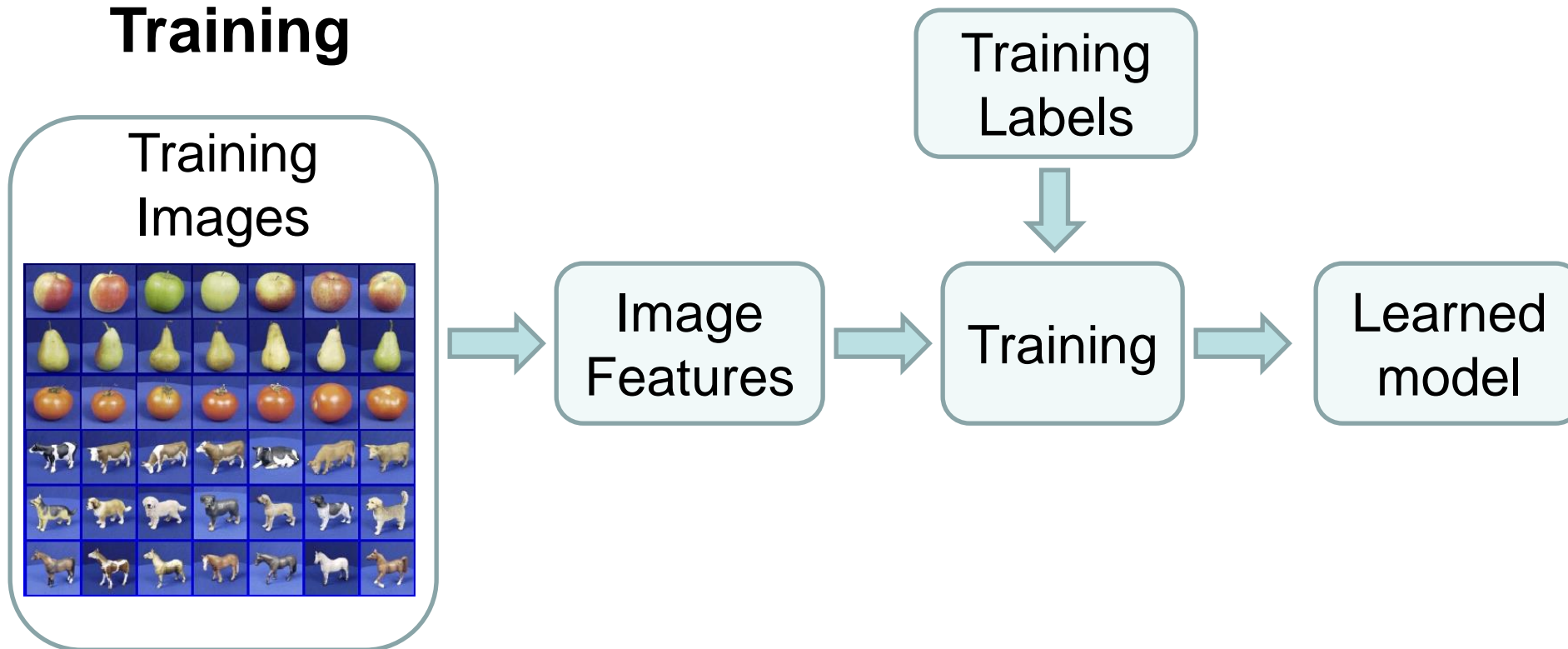
Learning a classifier

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

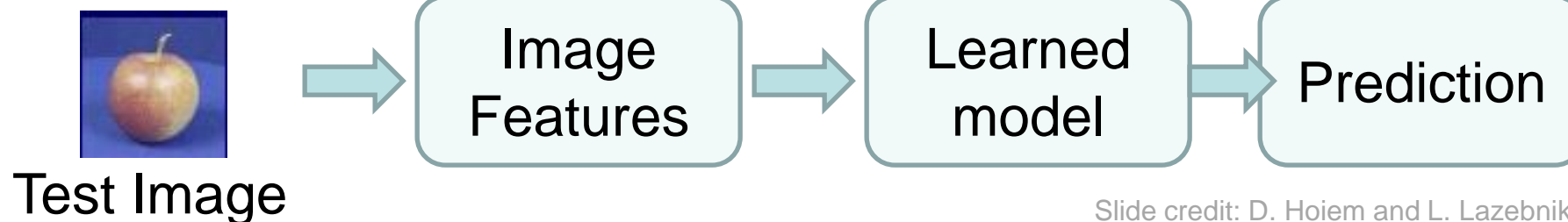


Steps

Training

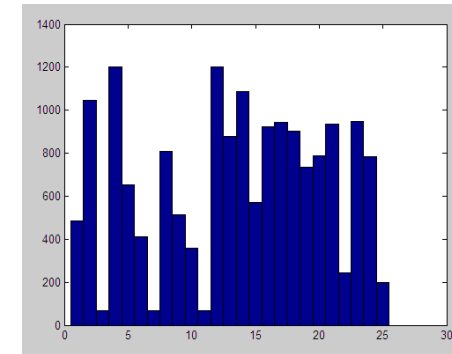


Testing



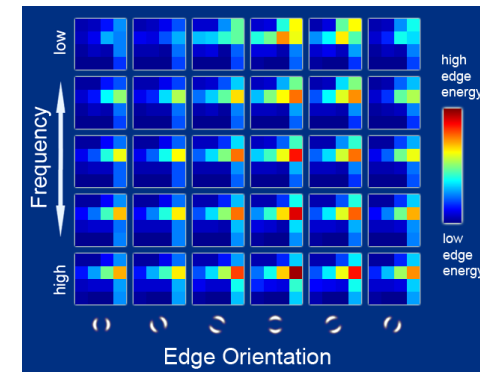
Features

- Raw pixels



- Histograms

- GIST descriptors



- ...

One way to think about it...

- Training labels dictate that two examples are the same or different, in some sense
- Features and distance measures define visual similarity
- Classifiers try to learn weights or parameters for features and distance measures so that visual similarity predicts label similarity

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Deep Convolutional Network
- Etc.

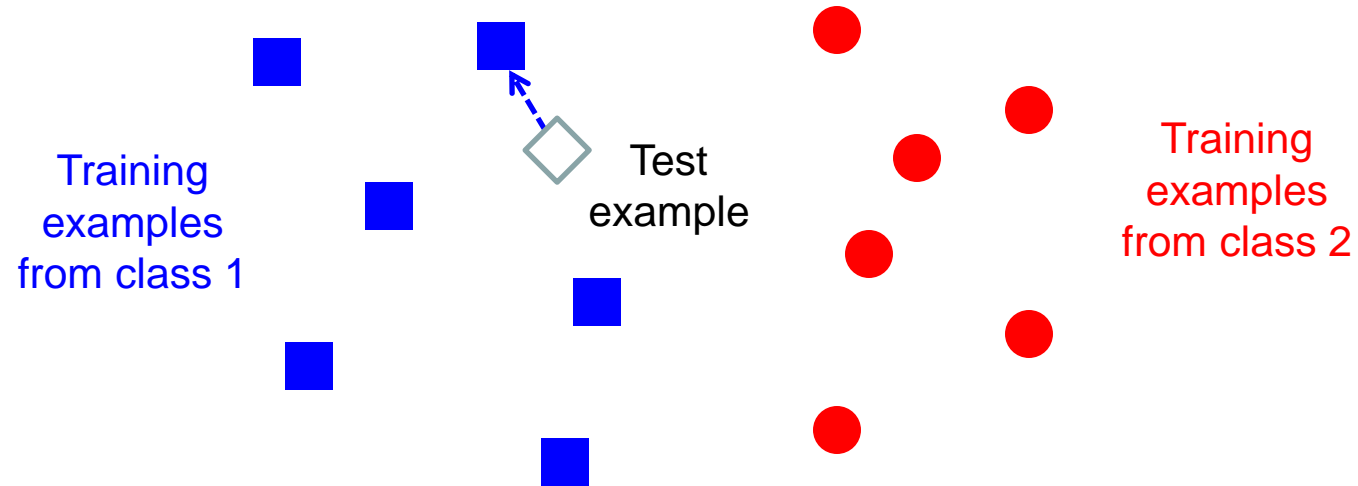
Which is the best one?

Claim:

The decision to *use* machine learning is more important than the choice of a *particular* learning method.

*Deep learning seems to be an exception to this, at the moment, probably because it is learning the feature representation.

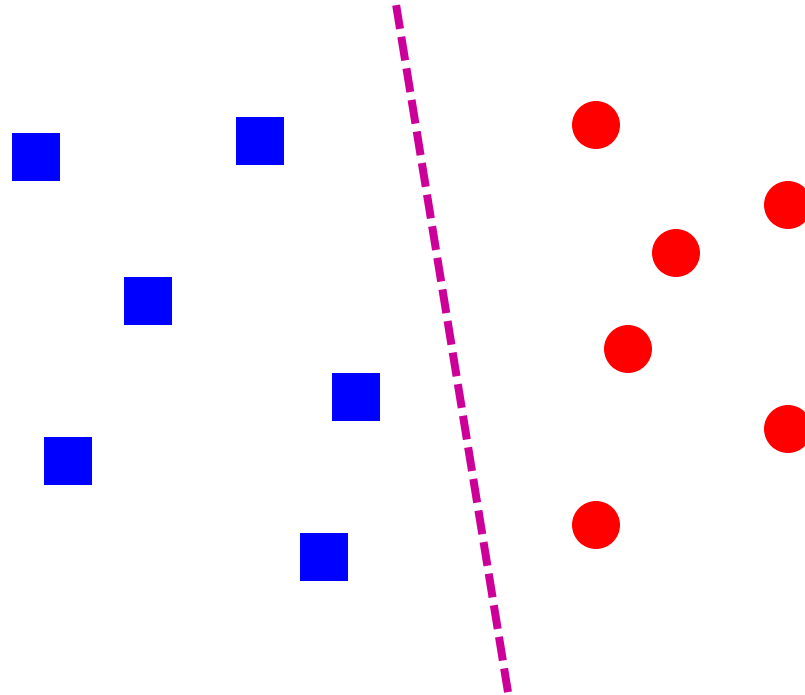
Classifiers: Nearest neighbor



$f(\mathbf{x}) = \text{label of the training example nearest to } \mathbf{x}$

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

Classifiers: Linear



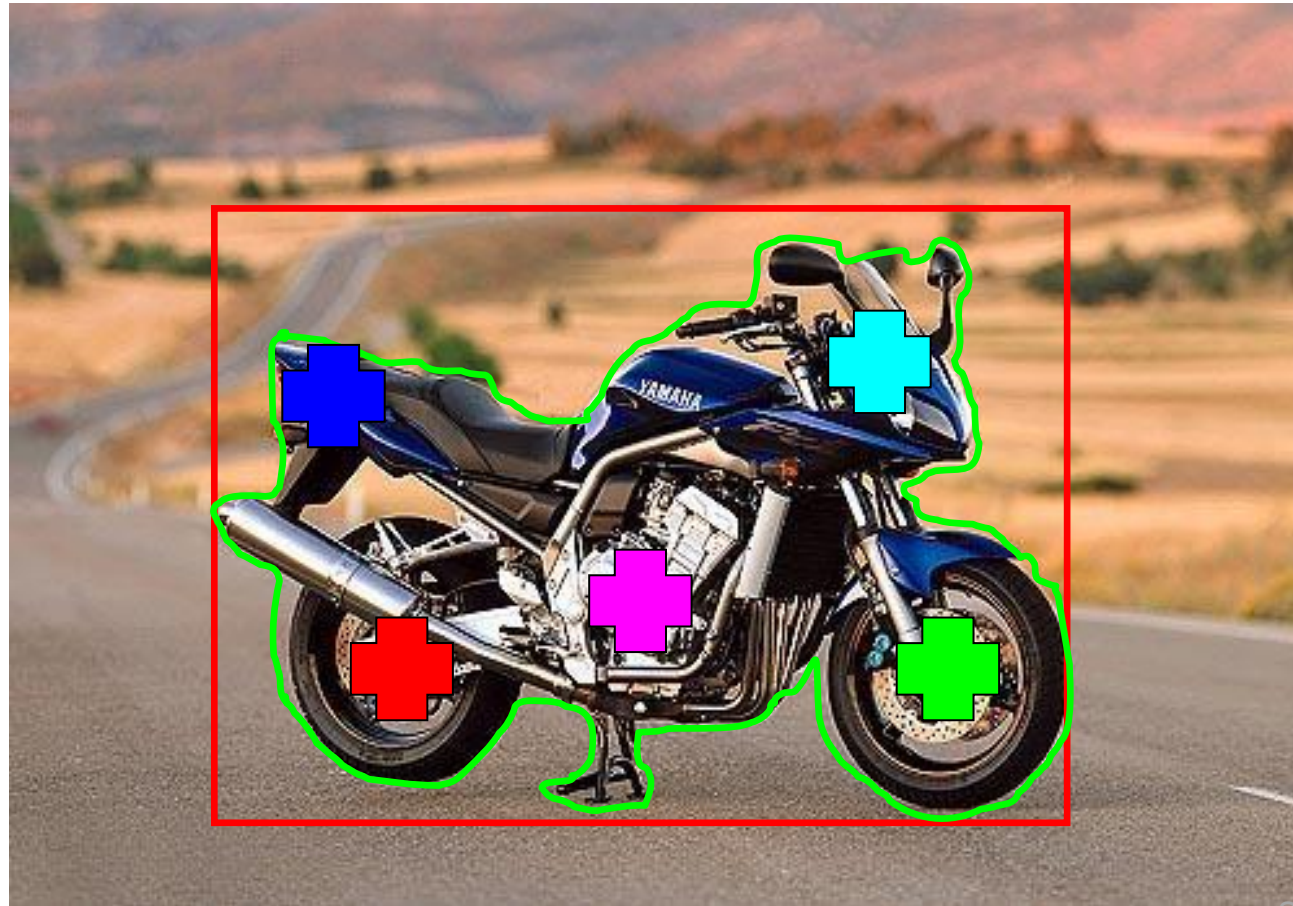
- Find a *linear function* to separate the classes:

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

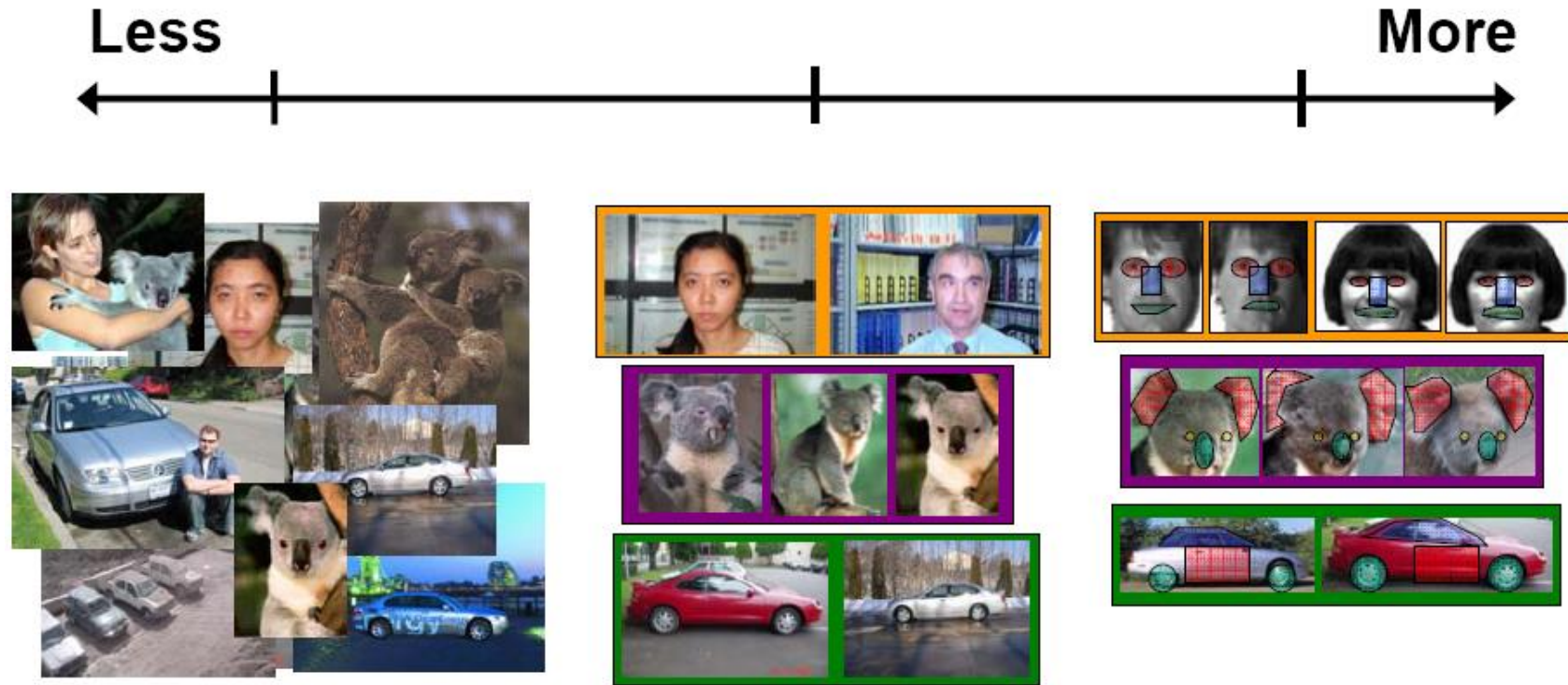
Recognition task and supervision

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

Contains a motorbike



Spectrum of supervision



Unsupervised

"Weakly" supervised

Fully supervised

Definition depends on task

Generalization



Training set (labels known)



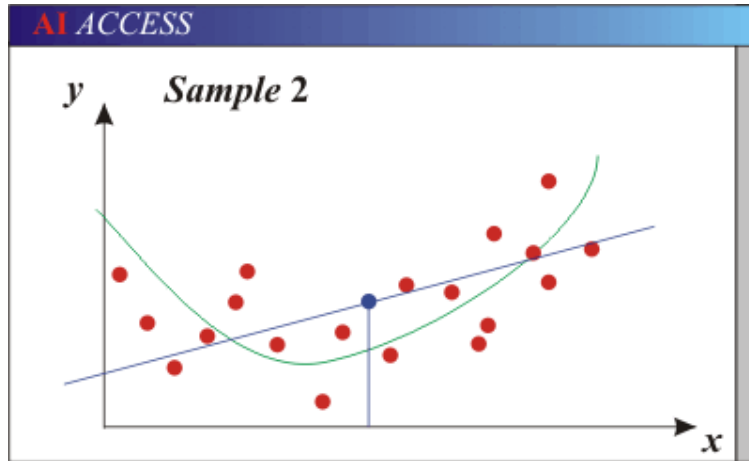
Test set (labels unknown)

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

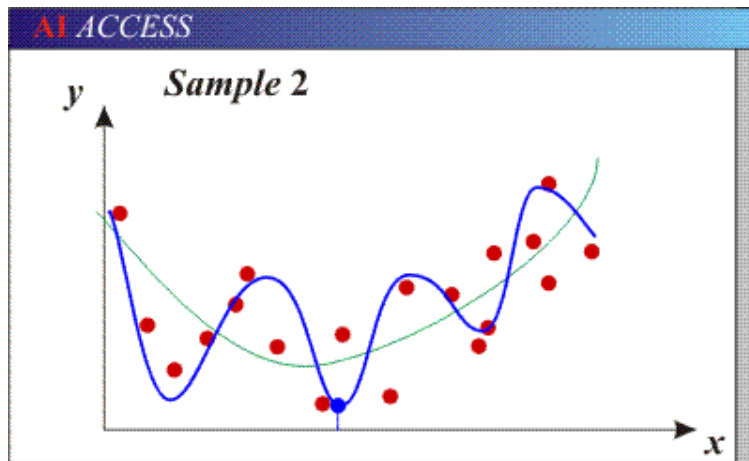
Generalization

- 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.
- **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

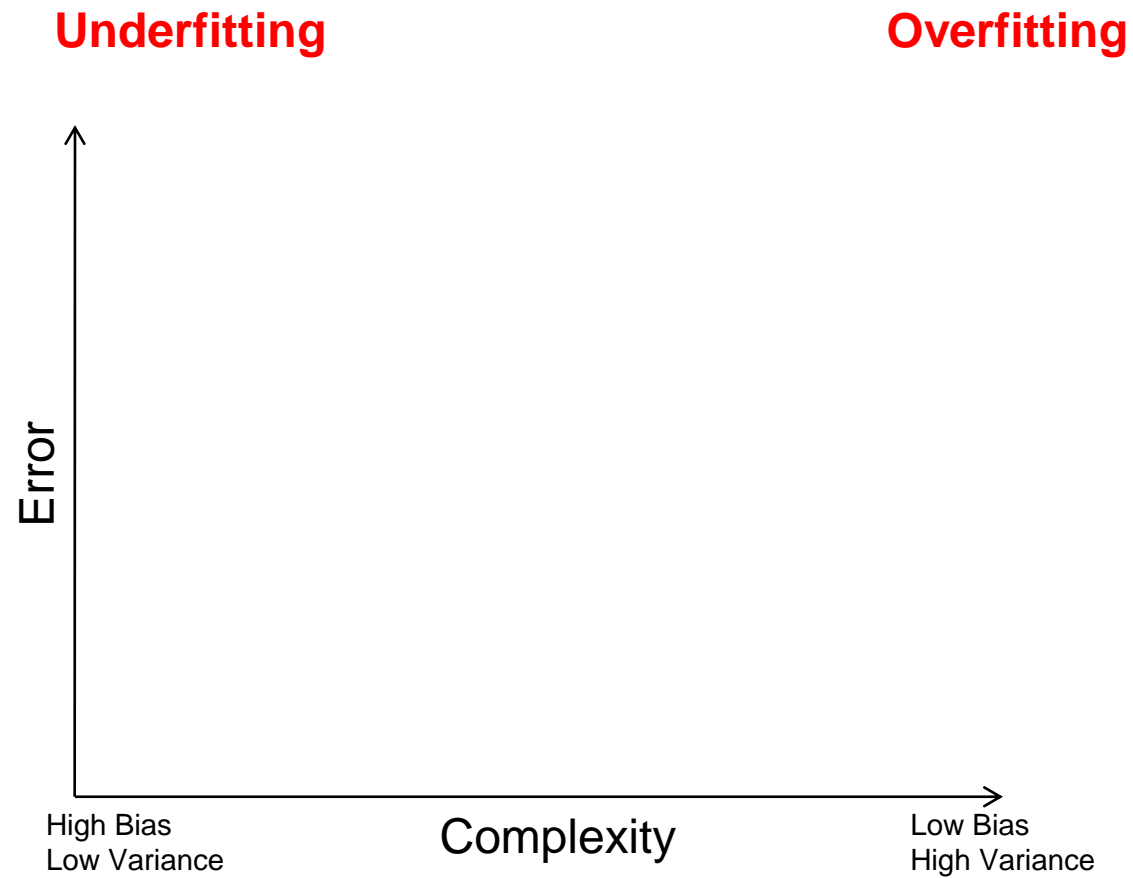


- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

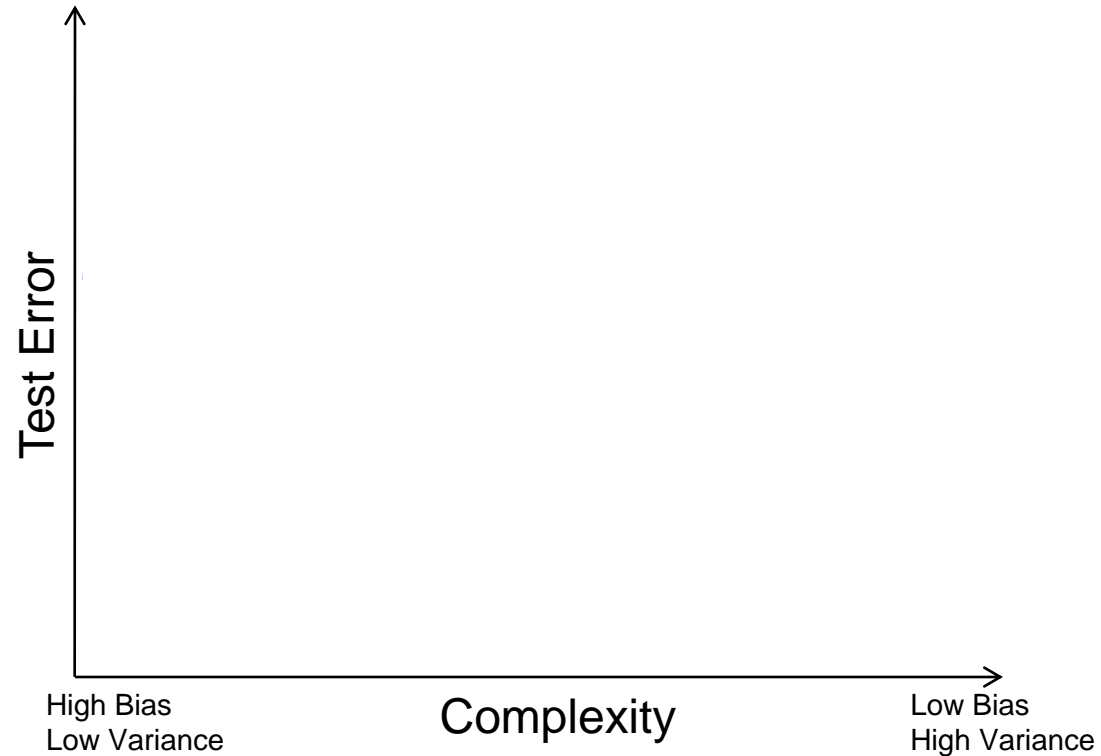


- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Bias-variance tradeoff

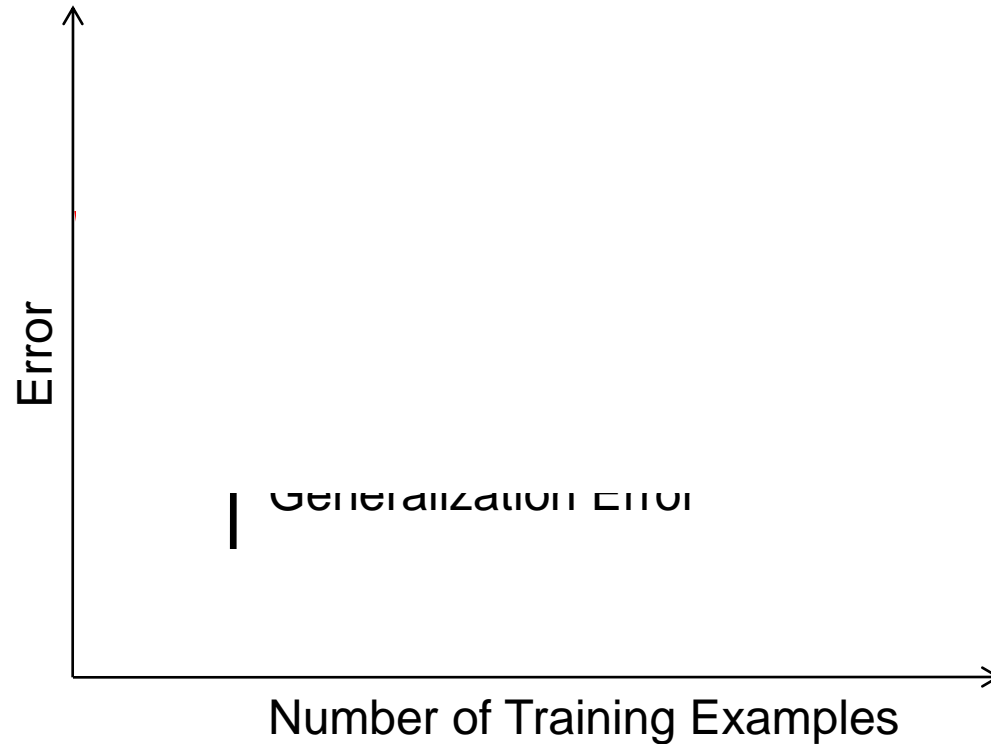


Bias-variance tradeoff



Effect of Training Size

Fixed prediction model



Remember...

- No classifier is inherently better than any other: you need to make assumptions to generalize
- Three kinds of error
 - Inherent: unavoidable
 - Bias: due to over-simplifications
 - Variance: due to inability to perfectly estimate parameters from limited data



- 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)

To be continued...