AIN311

Fundamentals of Machine Learning

Lecture 2: Machine Learning by Examples, Nearest Neighbor Classifier

HACETTEPE UNIVERSITY COMPUTER VISION LAB

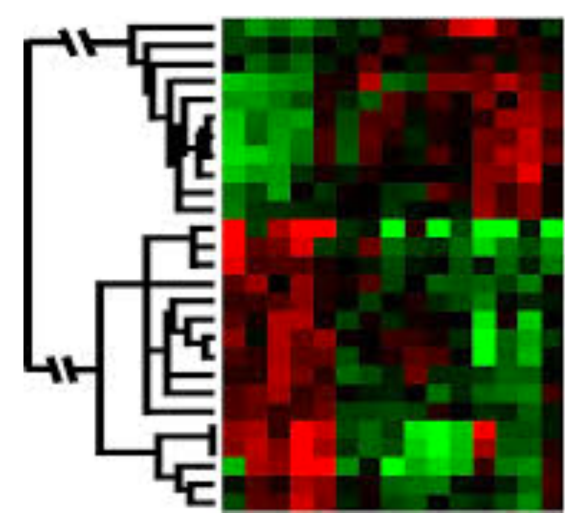
Erkut Erdem // Hacettepe University // Fall 2023

When Do We Use Machine Learning?

ML is used when:

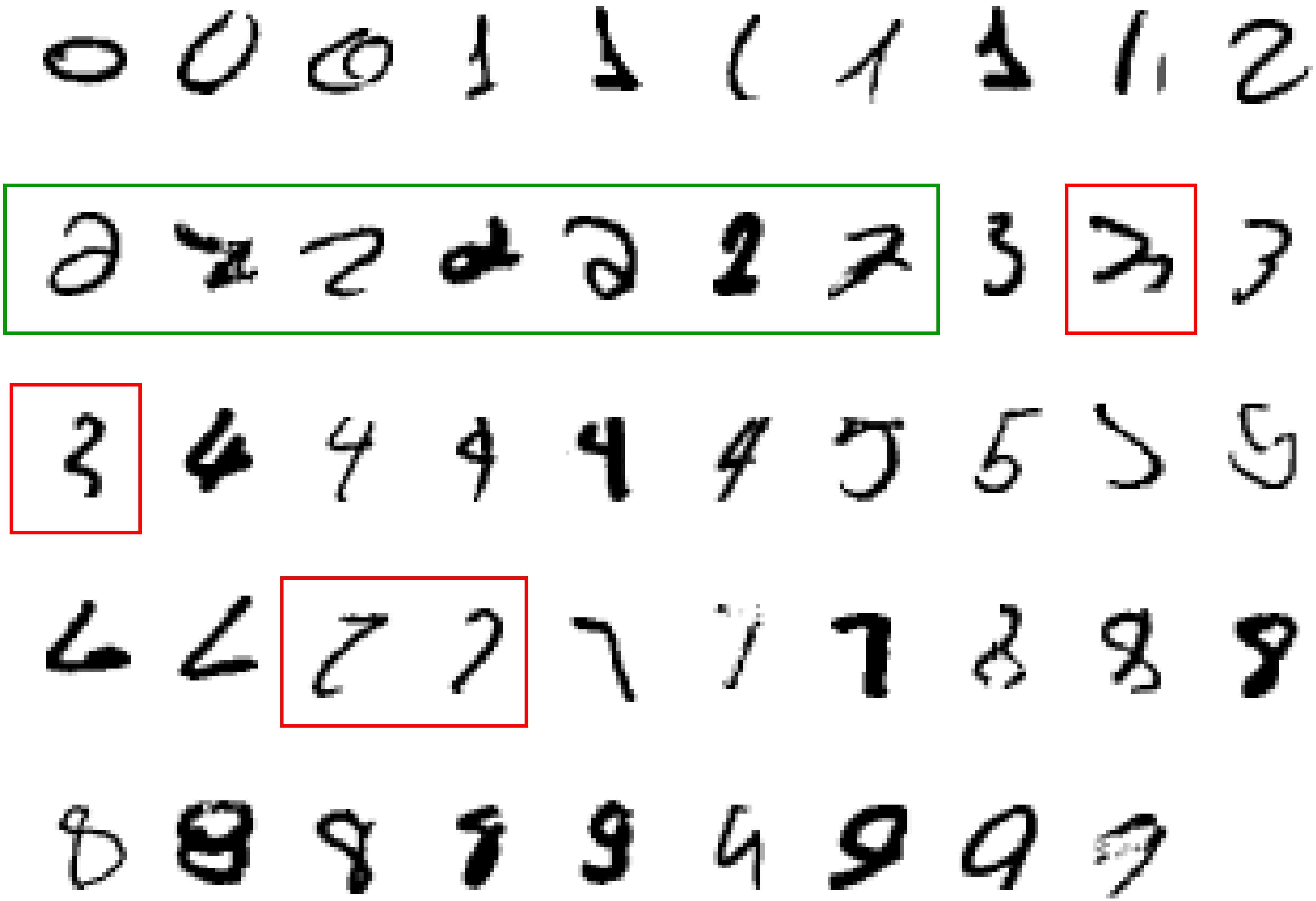
- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)



slide based on Ethem Alpaydin

A classic example of a task that requires machine learning: It is very hard to say what makes a 2



slide by Geoffrey Hinton

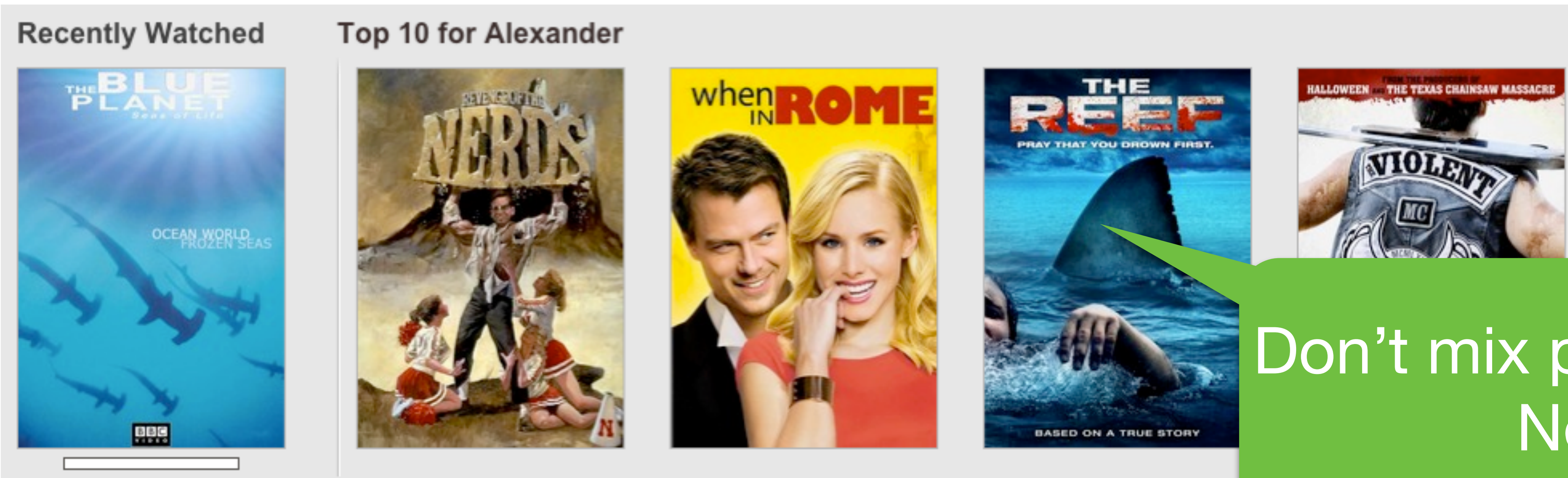
Machine Learning
(by examples)

Pose Estimation

KINECT SPORTS SEASON TWO

slide by Alex Smola

Collaborative Filtering



Don’t mix preferences on Netflix!

Amazon books

slide by Alex Smola

Collaborative Filtering

BUSINESS INSIDER

RETAIL

Amazon is being forced to review its website after it reportedly recommended shoppers buy items that can create explosives

Should be careful

Kate Taylor

Sep. 20, 2017, 11:51 AM 6,591

FACEBOOK LINKEDIN TWITTER EMAIL PRINT

Amazon is doing some self-examination after its website suggested customers purchase potentially dangerous groupings of products.

On Wednesday, Amazon told Reuters it was "reviewing its website" after the UK's Channel 4 News reported that the e-commerce giant's algorithm suggests that shoppers pair certain items with products that can be used to create homemade explosives.

This chemical compound's "frequently bought together" suggestions are the necessary ingredients to create a dangerous reaction. Amazon.com

Imitation Learning in Games

Avatar learns from your behavior

Black & White
Lionsgate Studios

slide by Alex Smola

Reinforcement Learning

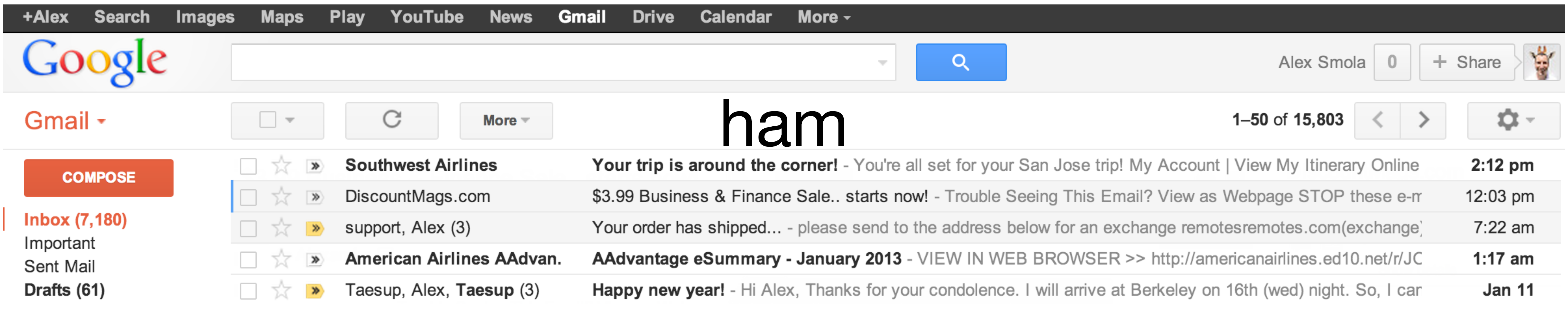
https://www.youtube.com/watch?v=IIeRKHsJBJ0

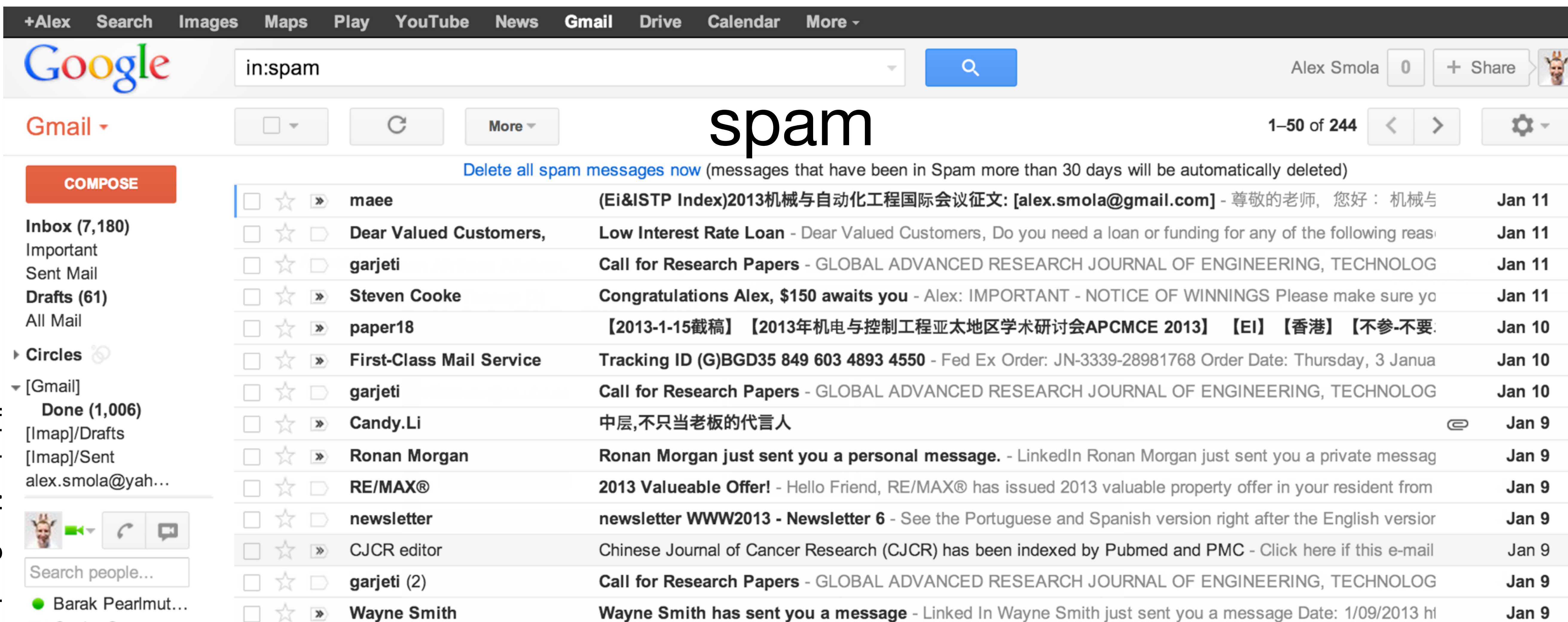
slide by Alex Smola

Reinforcement Learning

https://www.youtube.com/watch?v=5iZlrBqDYPM

Spam Filtering

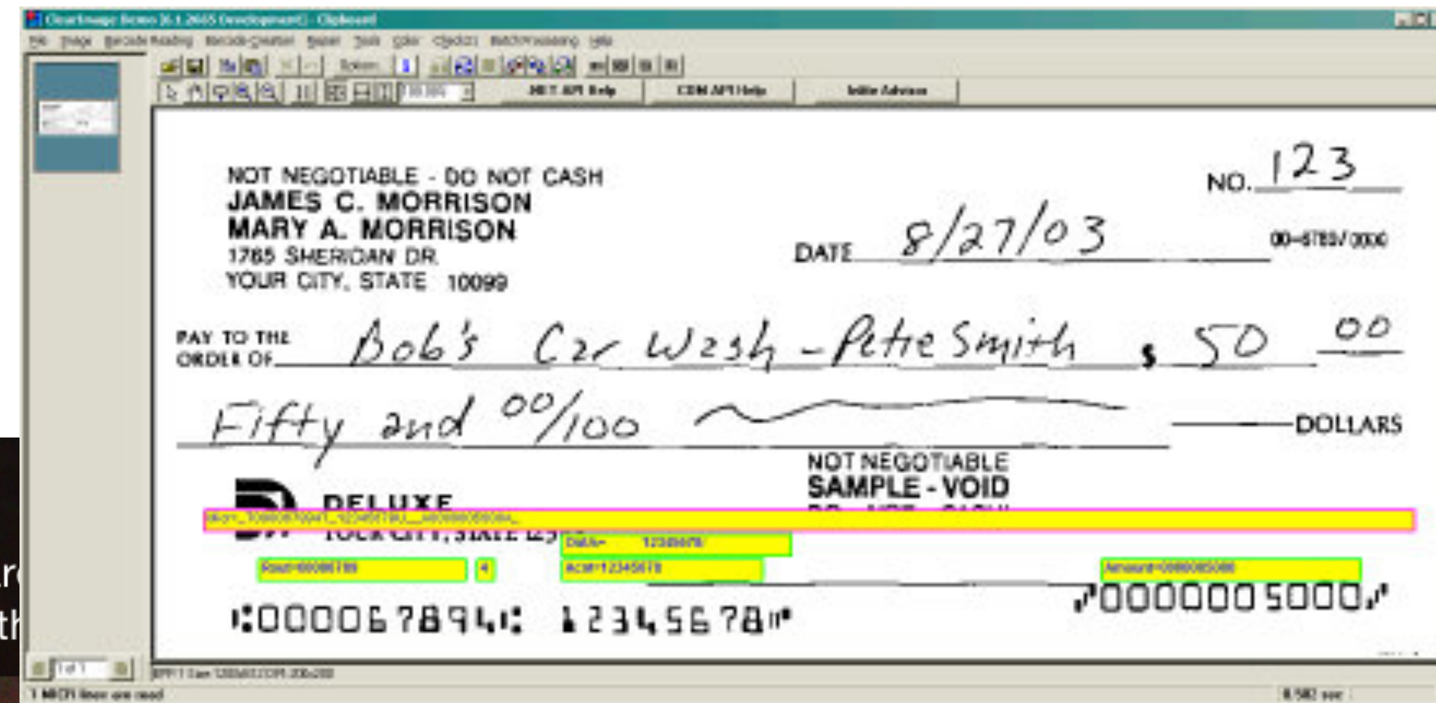


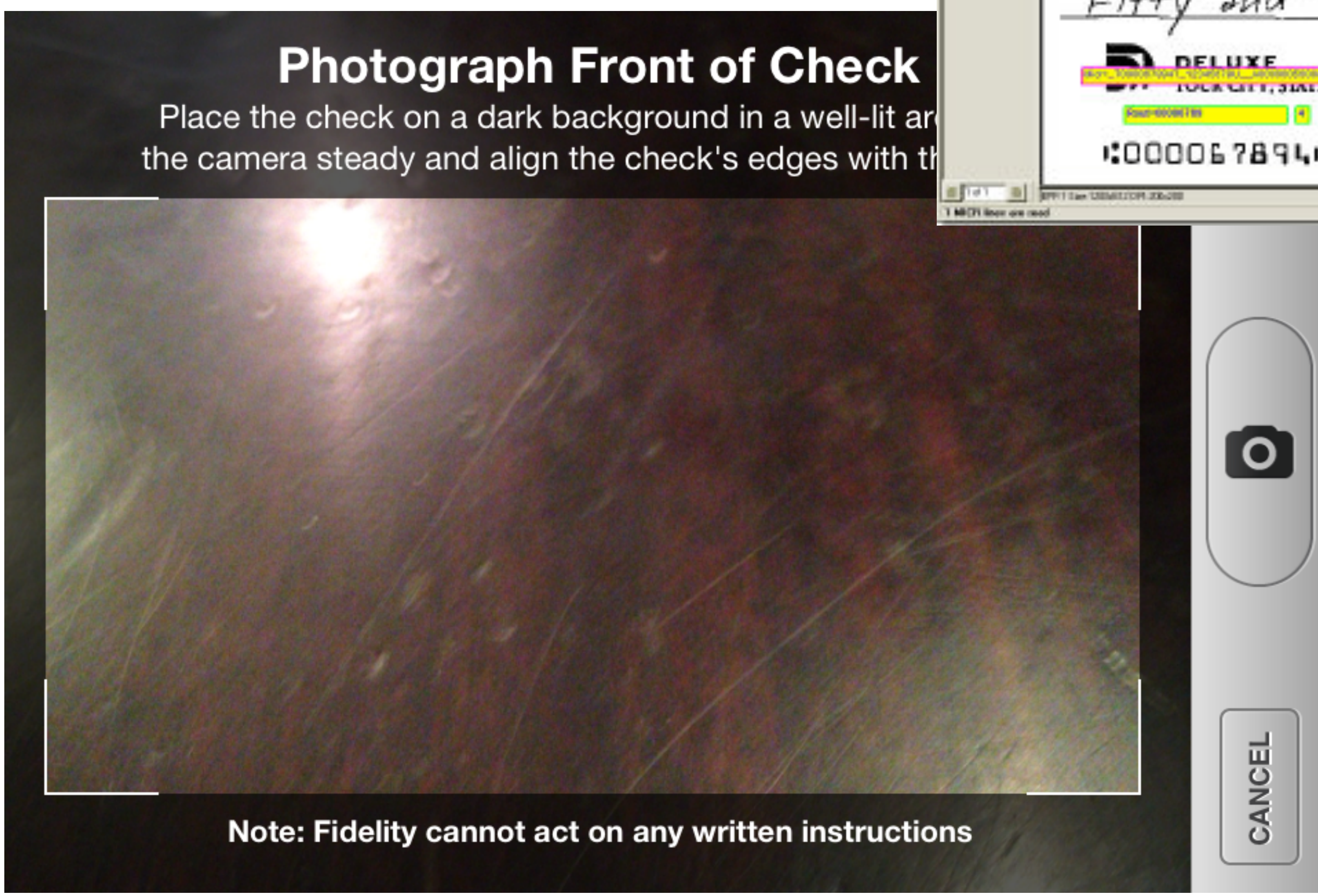


slide by Alex Smola

Cheque Reading

segment image





recognize handwriting

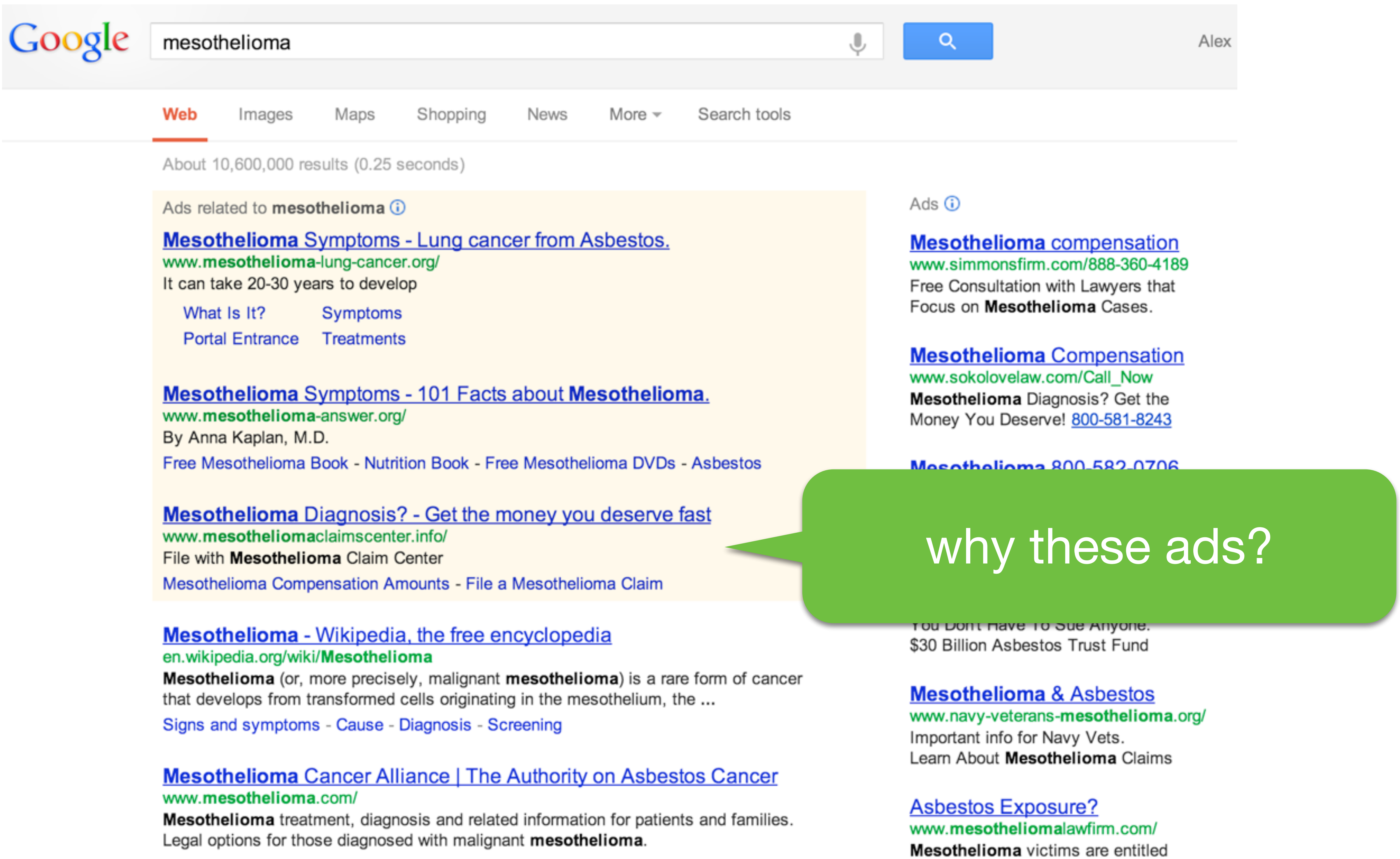
slide by Alex Smola

Image Layout

- Raw set of images from several cameras
- Joint layout based on image similarity

slide by Alex Smola

Search Ads



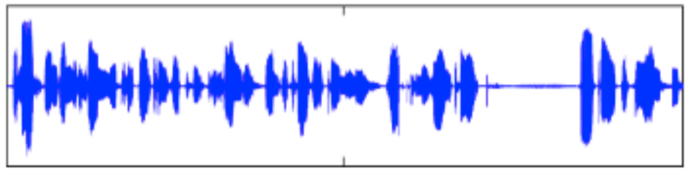
slide by Alex Smola

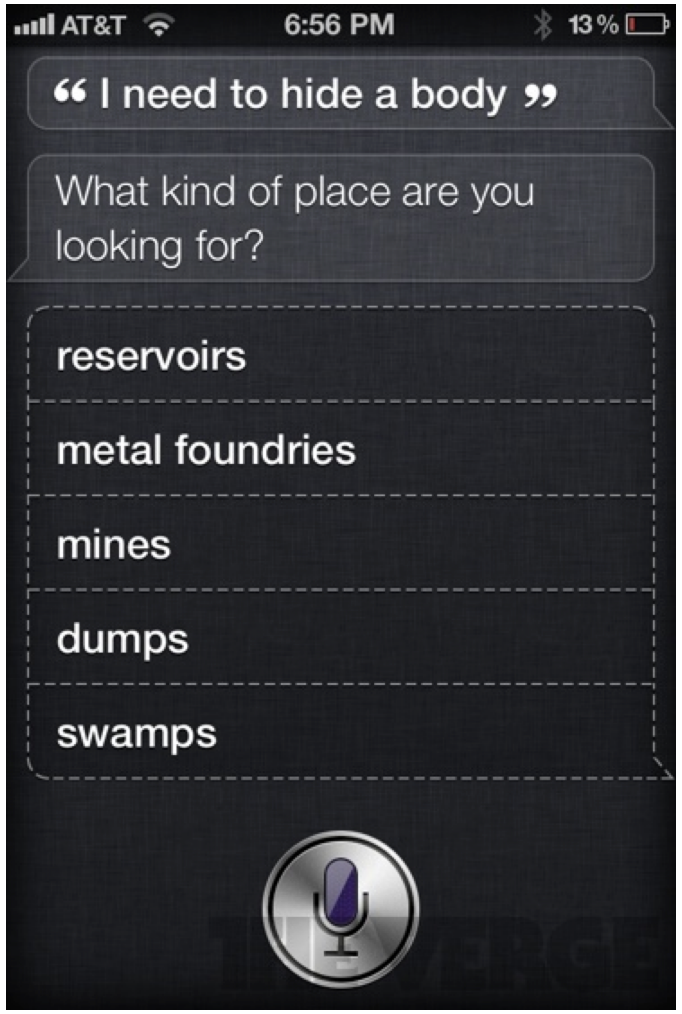
Self-Driving Cars

Image: https://medium.com/waymo/simulation-how-one-flashing-yellow-light-turns-into-thousands-of-hours-of-experience-a7a1cb475565

Speech Recognition

Given an audio waveform, robustly extract & recognize any spoken words



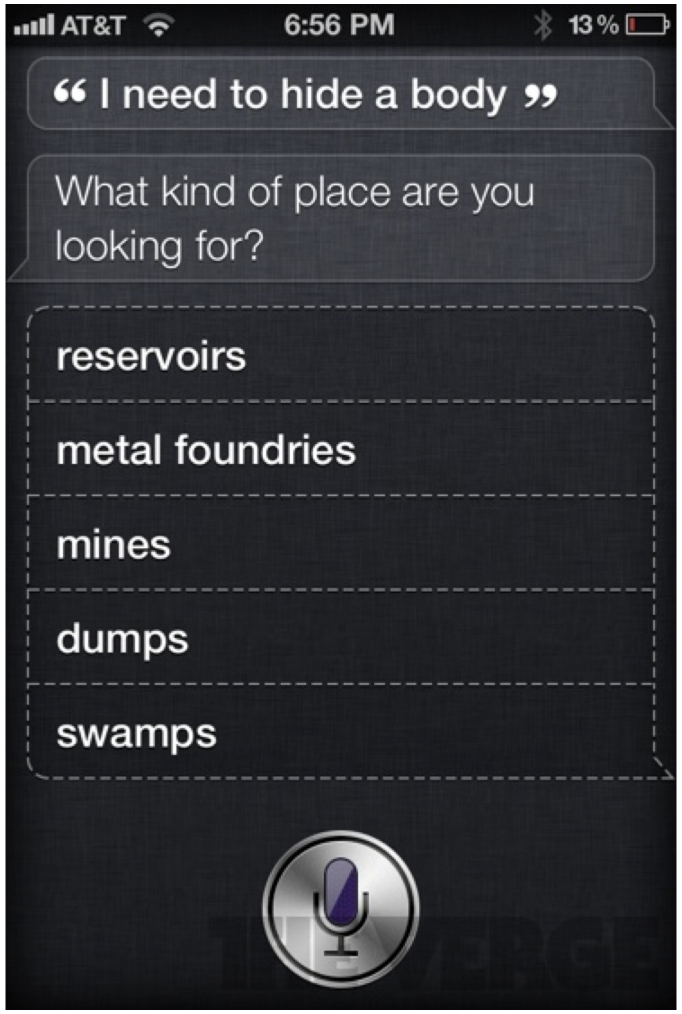


- Statistical models can be used to
  - Provide greater robustness to noise
  - Adapt to accent of different speakers
  - Learn from training

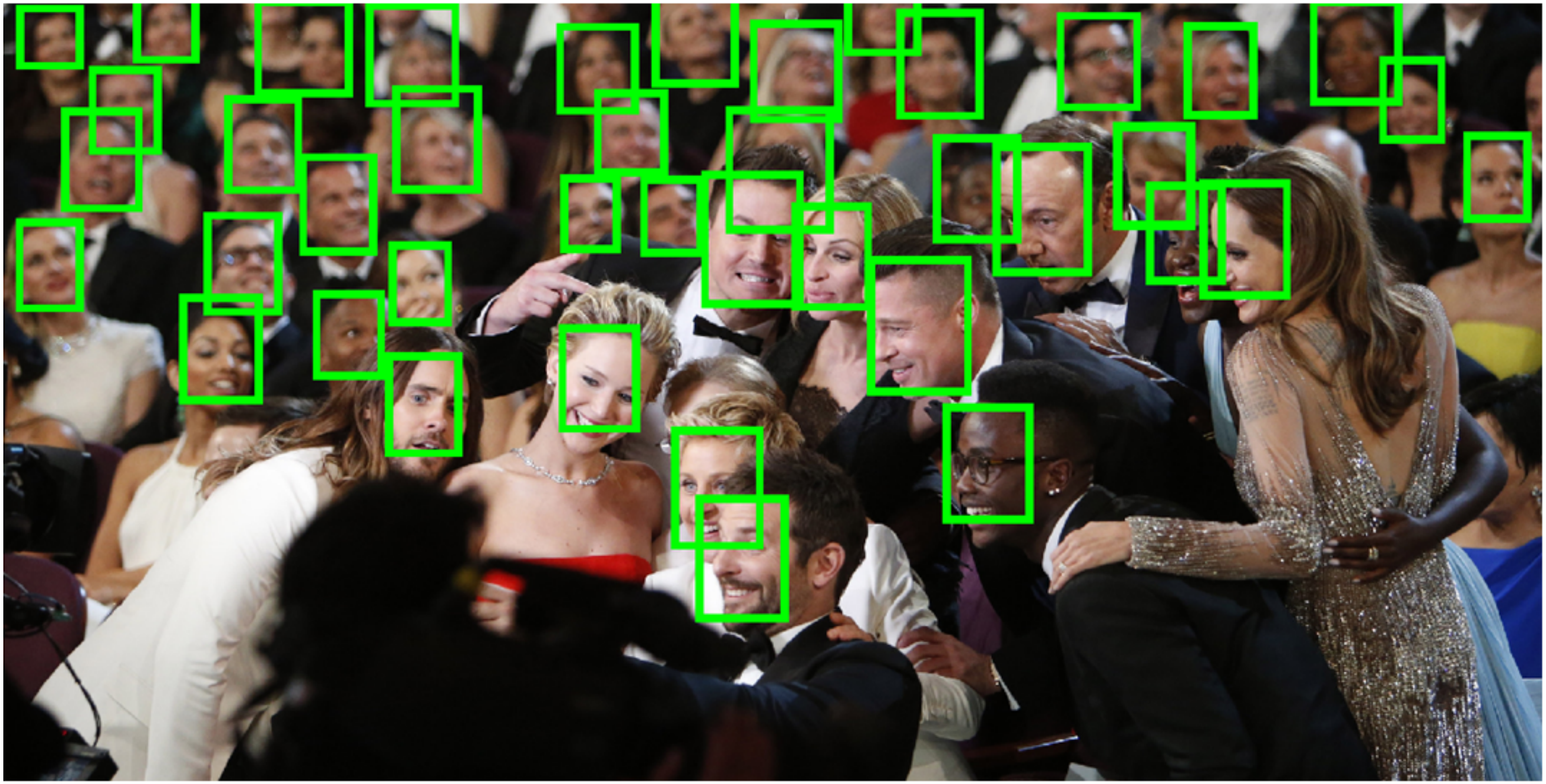
Natural Language Processing

I need to hide a body

noun, verb, preposition, …



Face Detection



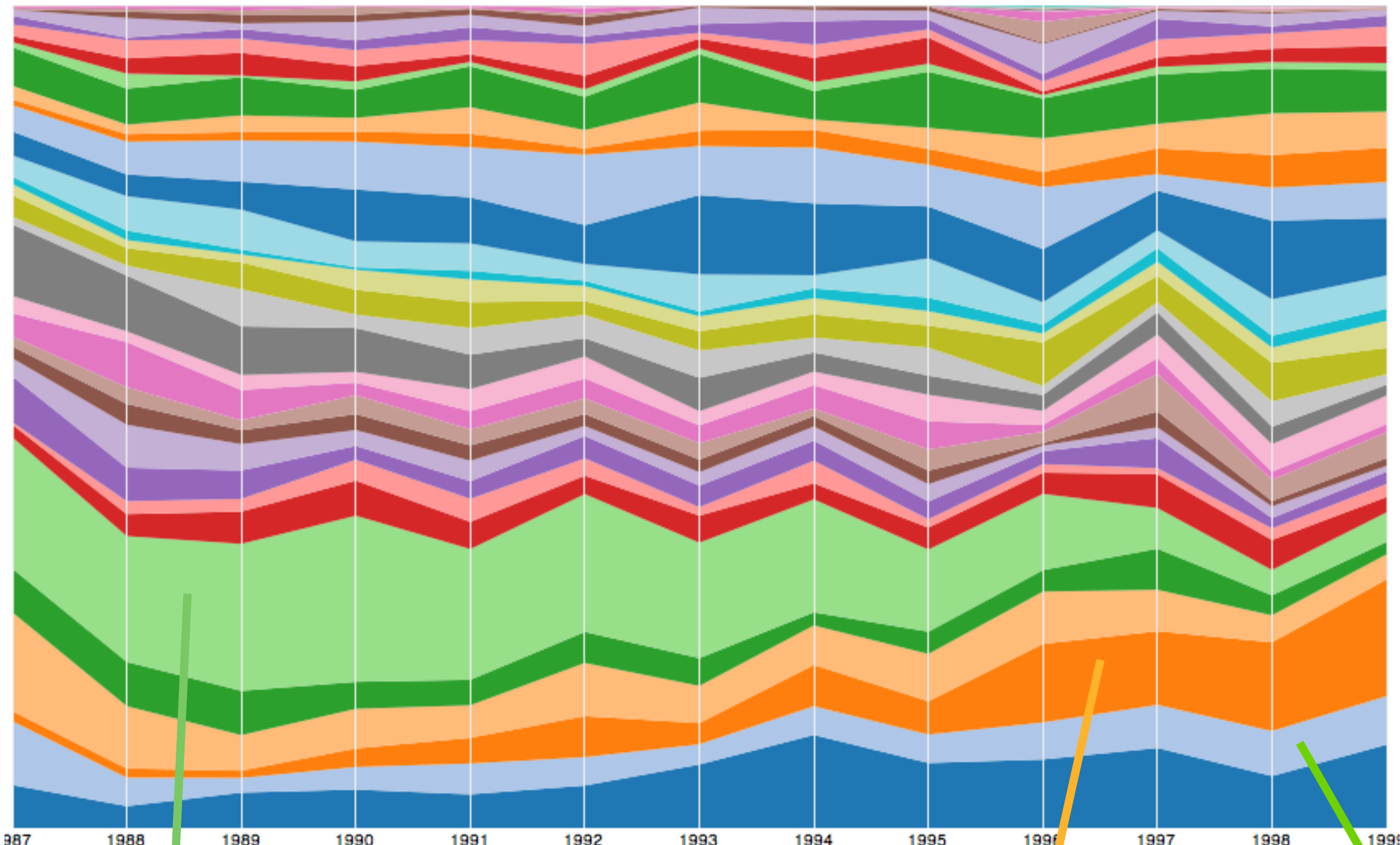
Yang et al., From Facial Parts Responses to Face Detection: A Deep Learning Approach, ICCV 2015

Scene Labeling via Deep Learning

[Farabet et al. ICML 2012, PAMI 2013]

slide by Eric Eaton

Topic Models of Text Documents



Neural Information Processing Systems Foundation

The New York Times

WIKIPEDIA
The Free Encyclopedia

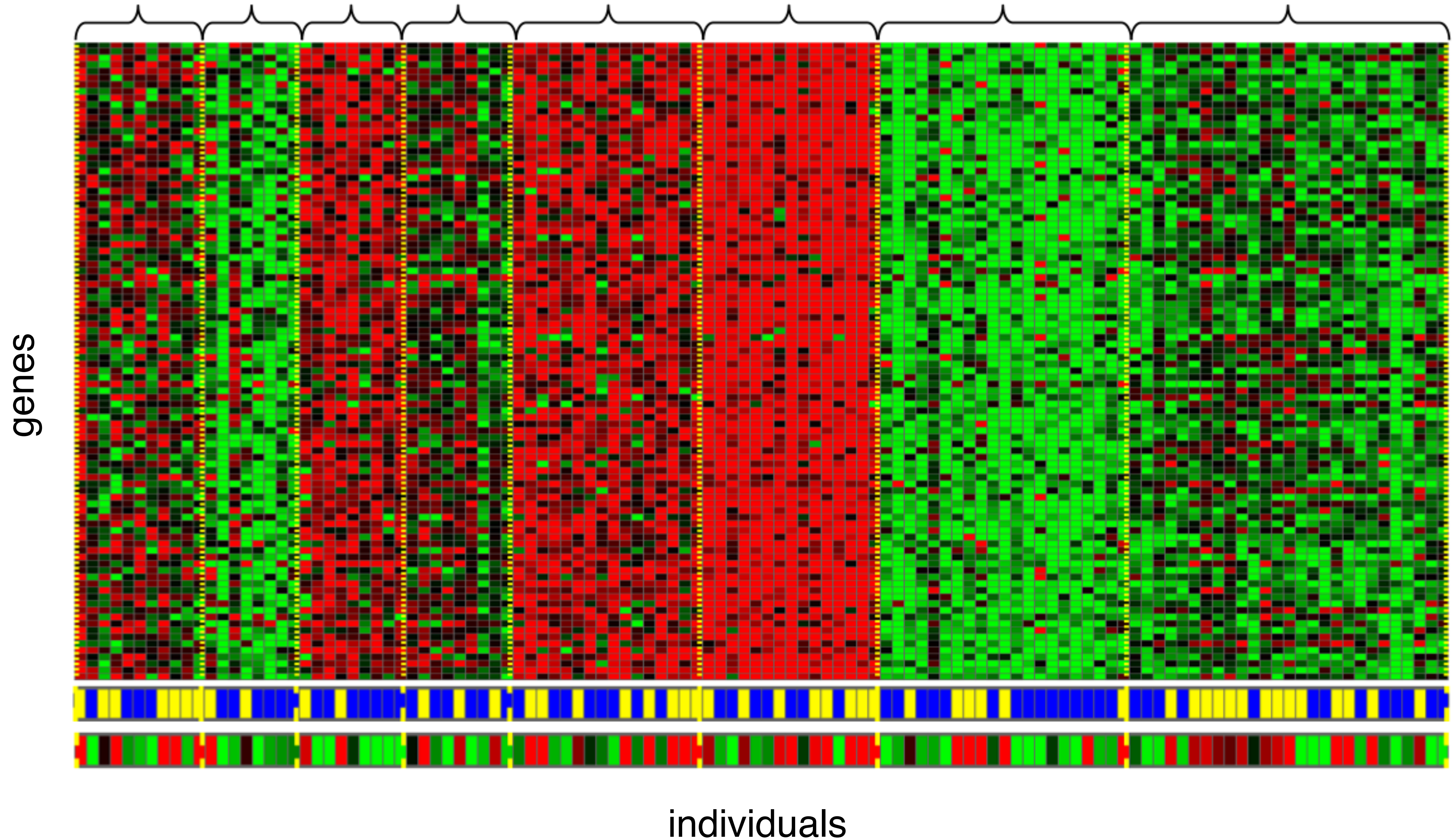
weight neural figure inputs error unit input layer network units weights training learning net hidden architecture set networks output number

estimation data density approach em probability model number set mixture gaussian posterior bayesian distribution figure parameters models log likelihood prior

cell responses field cells motion receptive visual tuning input model stimulus response neurons direction orientation stimuli spatial cortex cortical figure

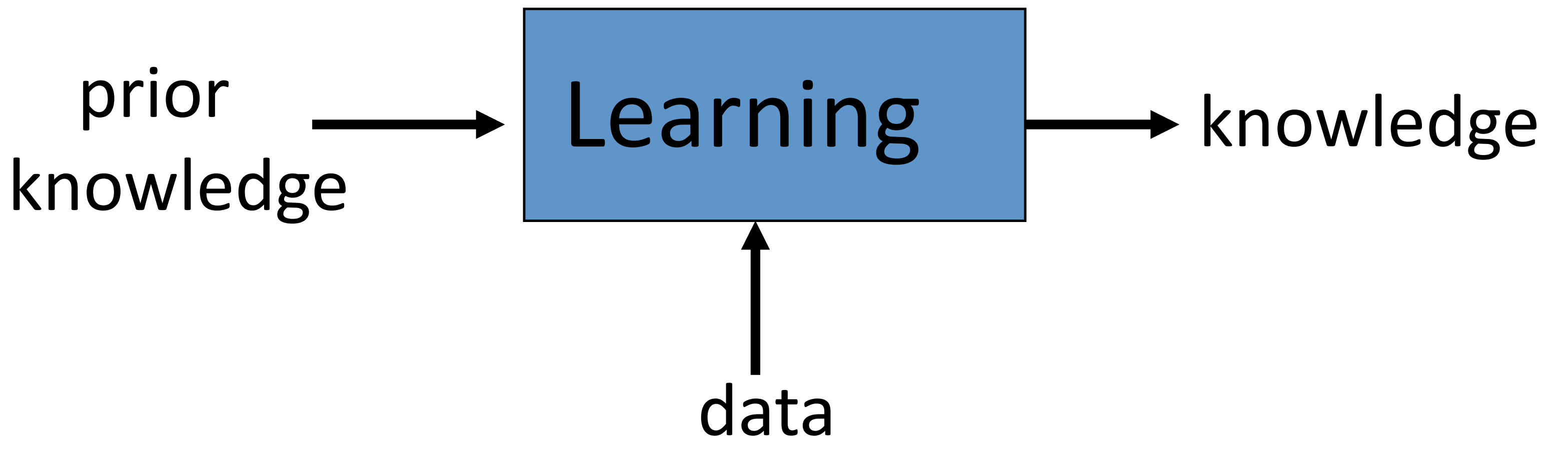
slide by Eric Sudderth

Genomics: group individuals by genetic similarity



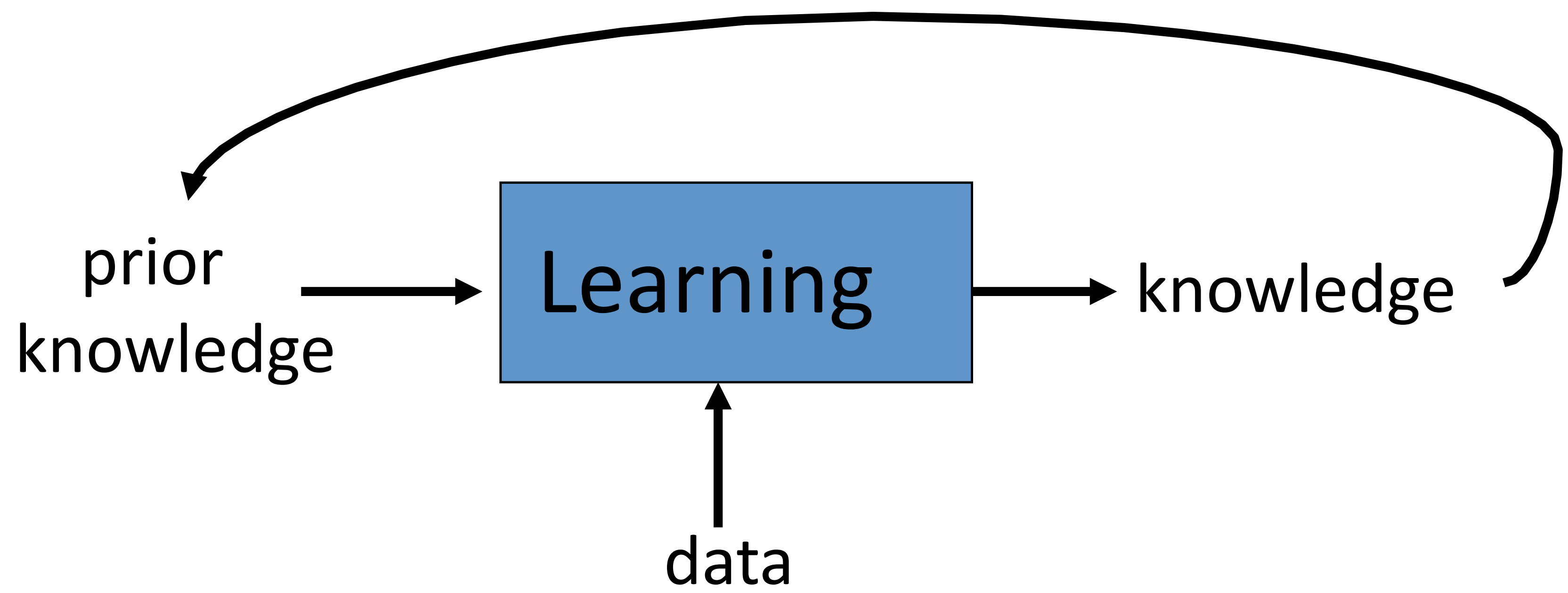
slide by Daphne Koller

Learning - revisited



slide by Stuart Russell

Learning - revisited



slide by Stuart Russell

Programming with Data

- Want adaptive robust and fault tolerant systems
- Rule-based implementation is (often)
  - difficult (for the programmer)
  - brittle (can miss many edge-cases)
  - becomes a nightmare to maintain explicitly
  - often doesn’t work too well (e.g. OCR)

- Usually easy to obtain examples of what we want
  IF x THEN DO y
- Collect many pairs (x_i, y_i)
- Estimate function f such that $f(x_i) = y_i$ (supervised learning)
- Detect patterns in data (unsupervised learning)

slide by Mehryar Mohri

Objectives of Machine Learning

- Algorithms: design of efficient, accurate, and general learning algorithms to
  - deal with large-scale problems.
  - make accurate predictions (unseen examples).
  - handle a variety of different learning problems.
- Theoretical questions:
  - what can be learned? Under what conditions?
  - what learning guarantees can be given?
  - what is the algorithmic complexity?

slide by Mehryar Mohri

Definitions and Terminology

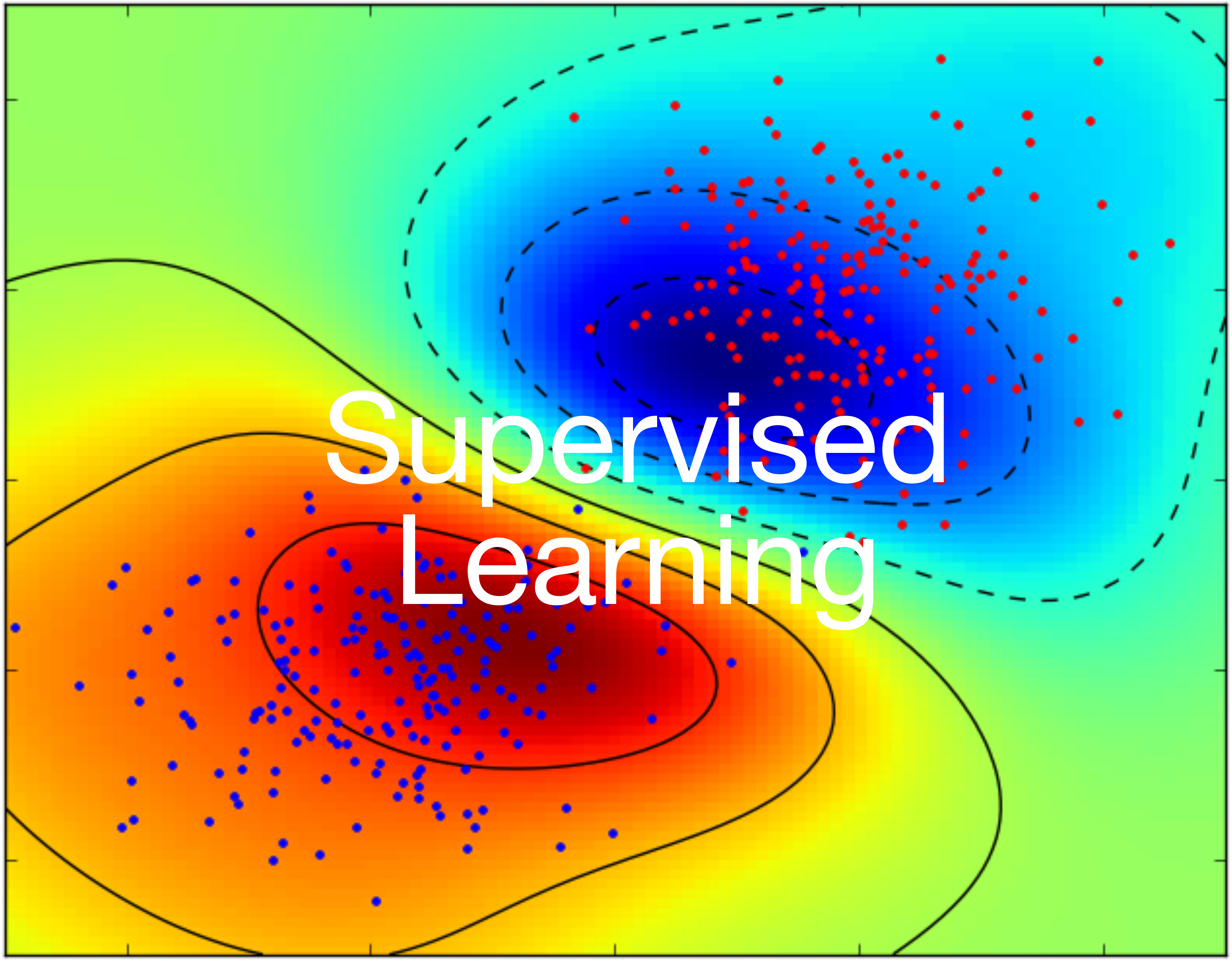
- Example: an object, instance of the data used.
- Features: the set of attributes, often represented as a vector, associated to an example (e.g., height and weight for gender prediction).
- Labels: in classification, category associated to an object (e.g., positive or negative in binary classification); in regression real value.
- Training data: data used for training learning algorithm (often labeled data).

slide by Mehryar Mohri

Definitions and Terminology (cont'd.)

- Test data: data used for testing learning algorithm (unlabeled data).
- Unsupervised learning: no labeled data.
- Supervised learning: uses labeled data.
- Weakly or semi-supervised learning: intermediate scenarios.
- Reinforcement learning: rewards from sequence of action.

slide by Mehryar Mohri



slide by Alex Smola

Supervised Learning

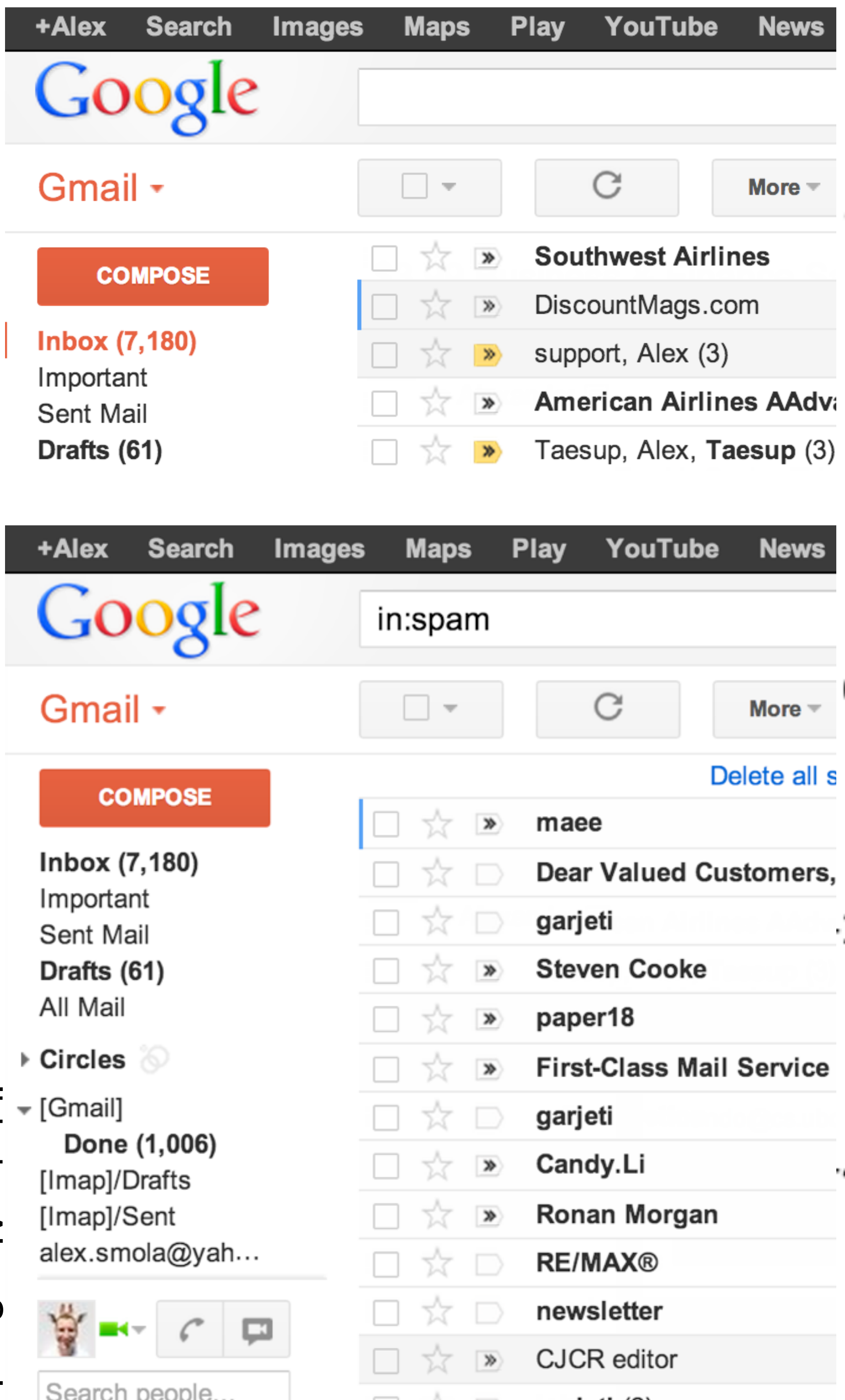
- **Binary classification**
  Given x find y in {-1, 1}
- **Multicategory classification**
  Given x find y in {1, ... k}
- **Regression**
  Given x find y in R (or R^d)
- **Sequence annotation**
  Given sequence x_1 ... x_l find y_1 ... y_l
- **Hierarchical Categorization (Ontology)**
  Given x find a point in the hierarchy of y (e.g. a tree)
- **Prediction**
  Given x_t and y_{t-1} ... y_1 find y_t

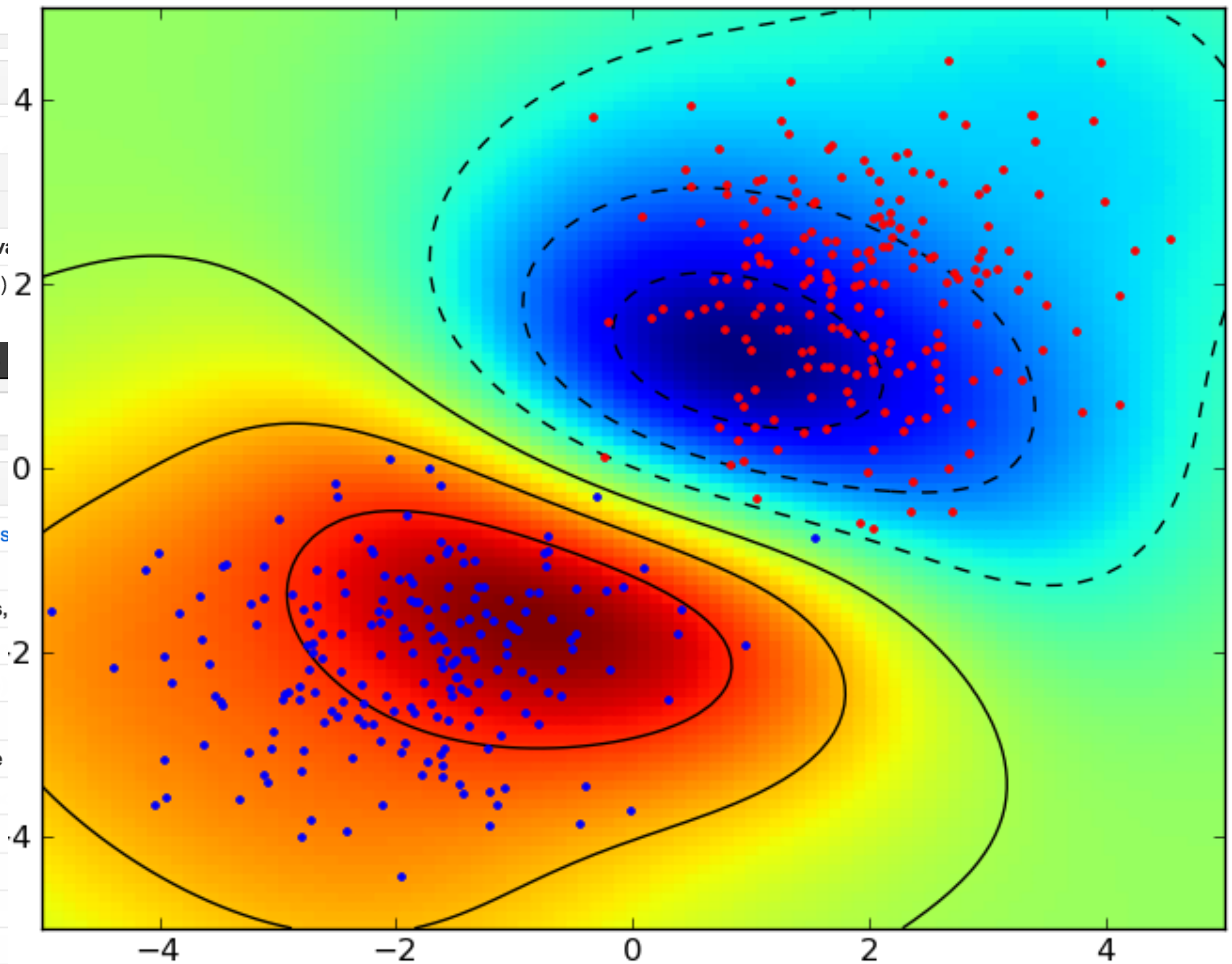
often with loss

$$l(y, f(x))$$

slide by Alex Smola

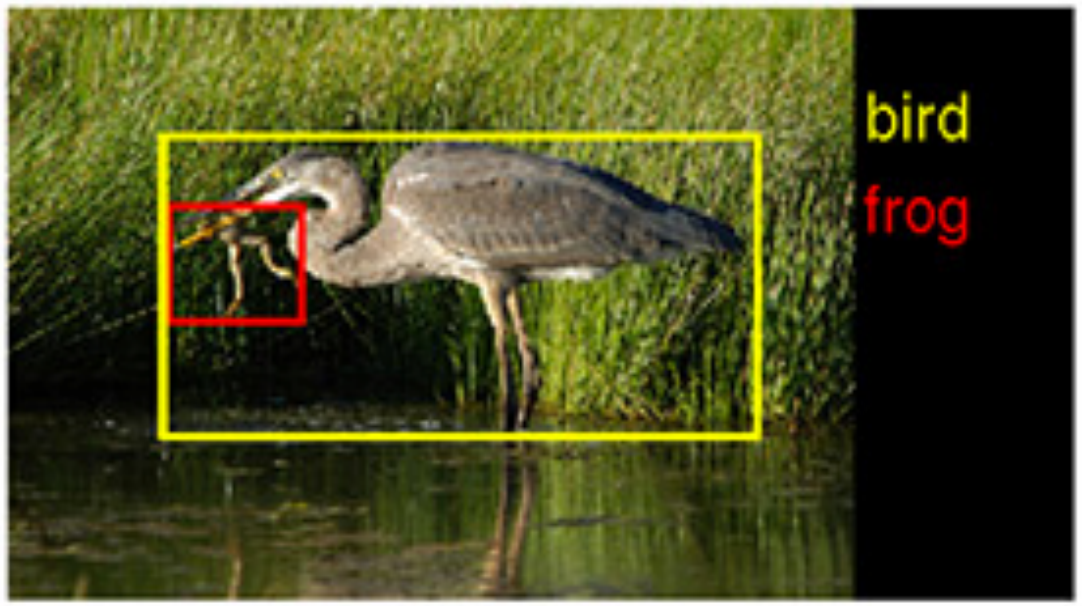
Binary Classification

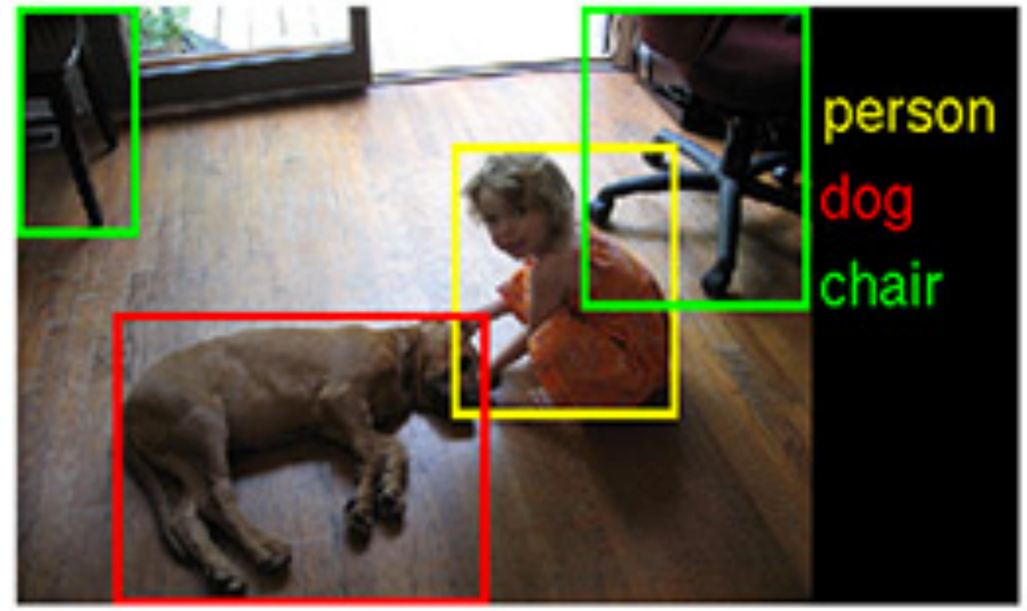


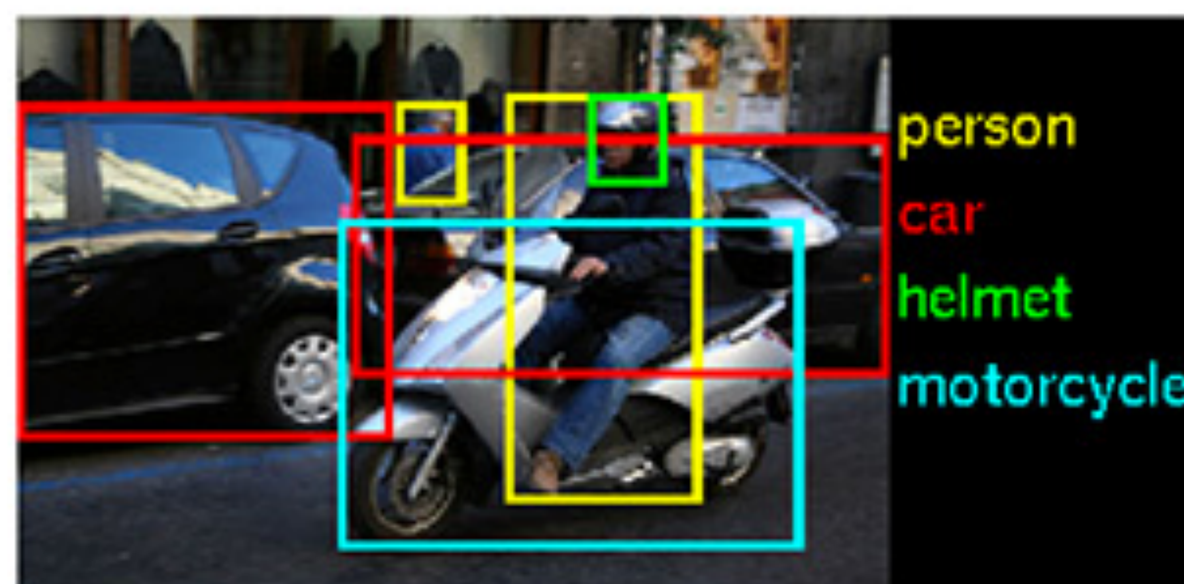


slide by Alex Smola

Multiclass Classification + Annotation

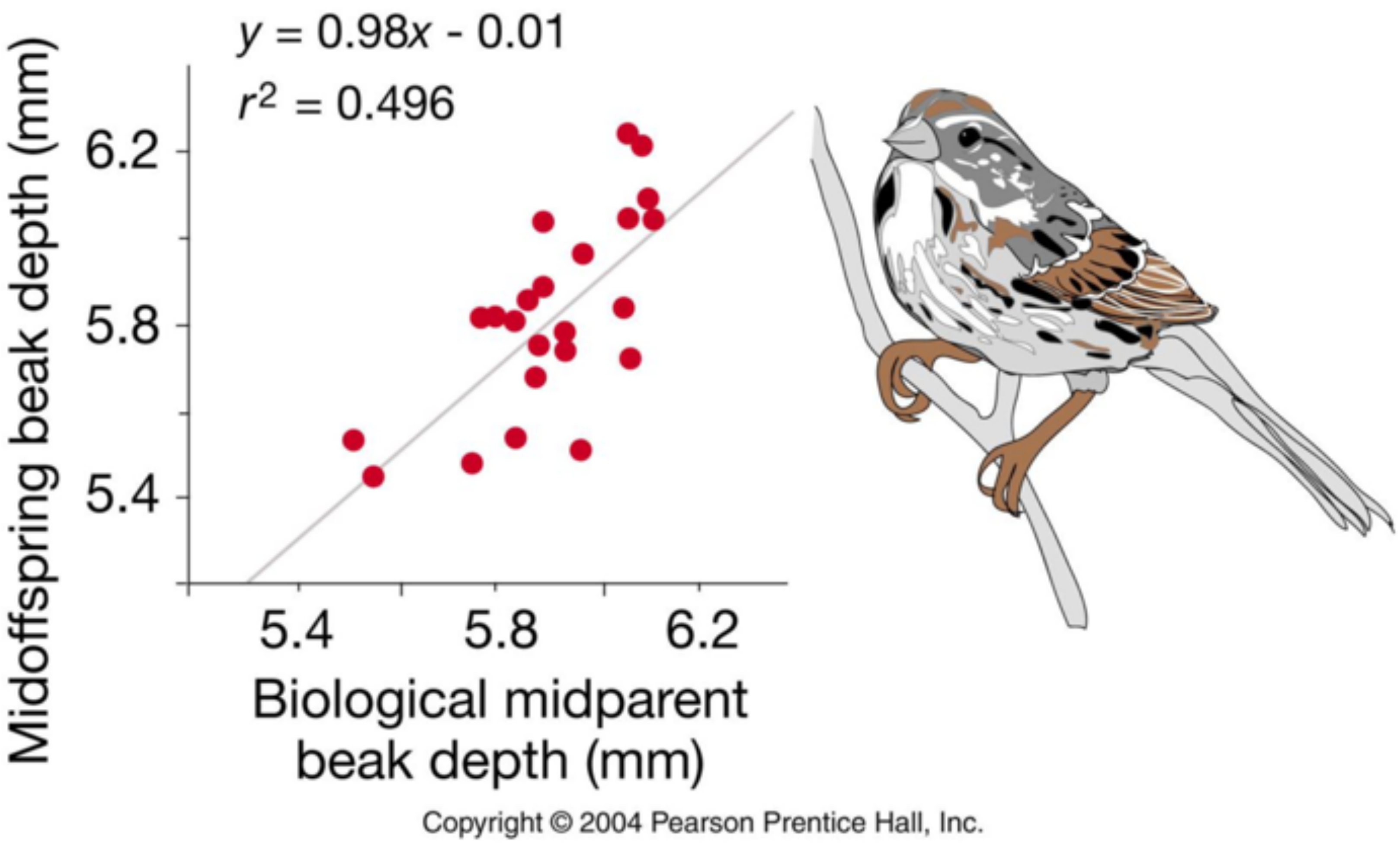




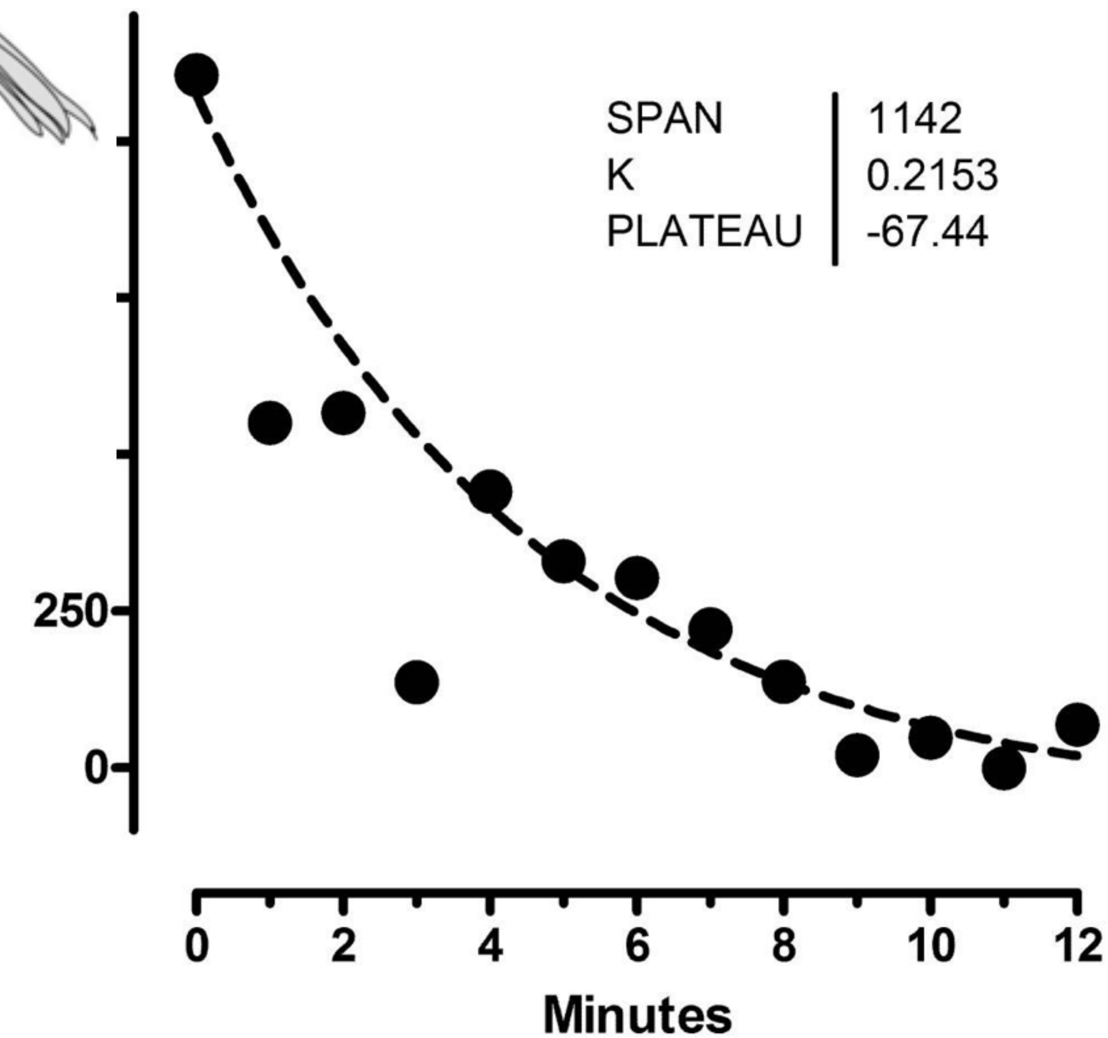


slide by Alex Smola

Regression



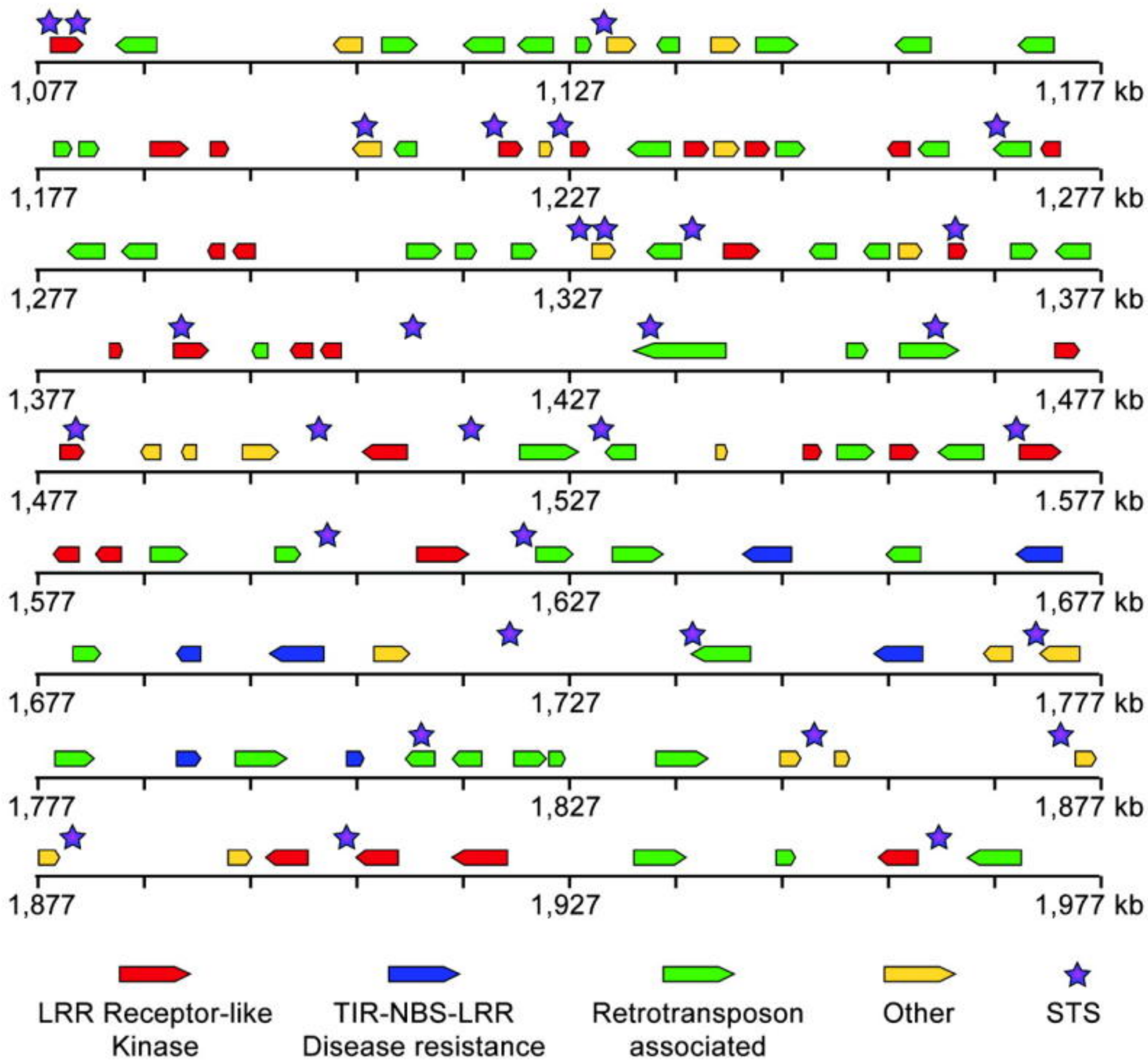
nonlinear



linear

slide by Alex Smola

Sequence Annotation



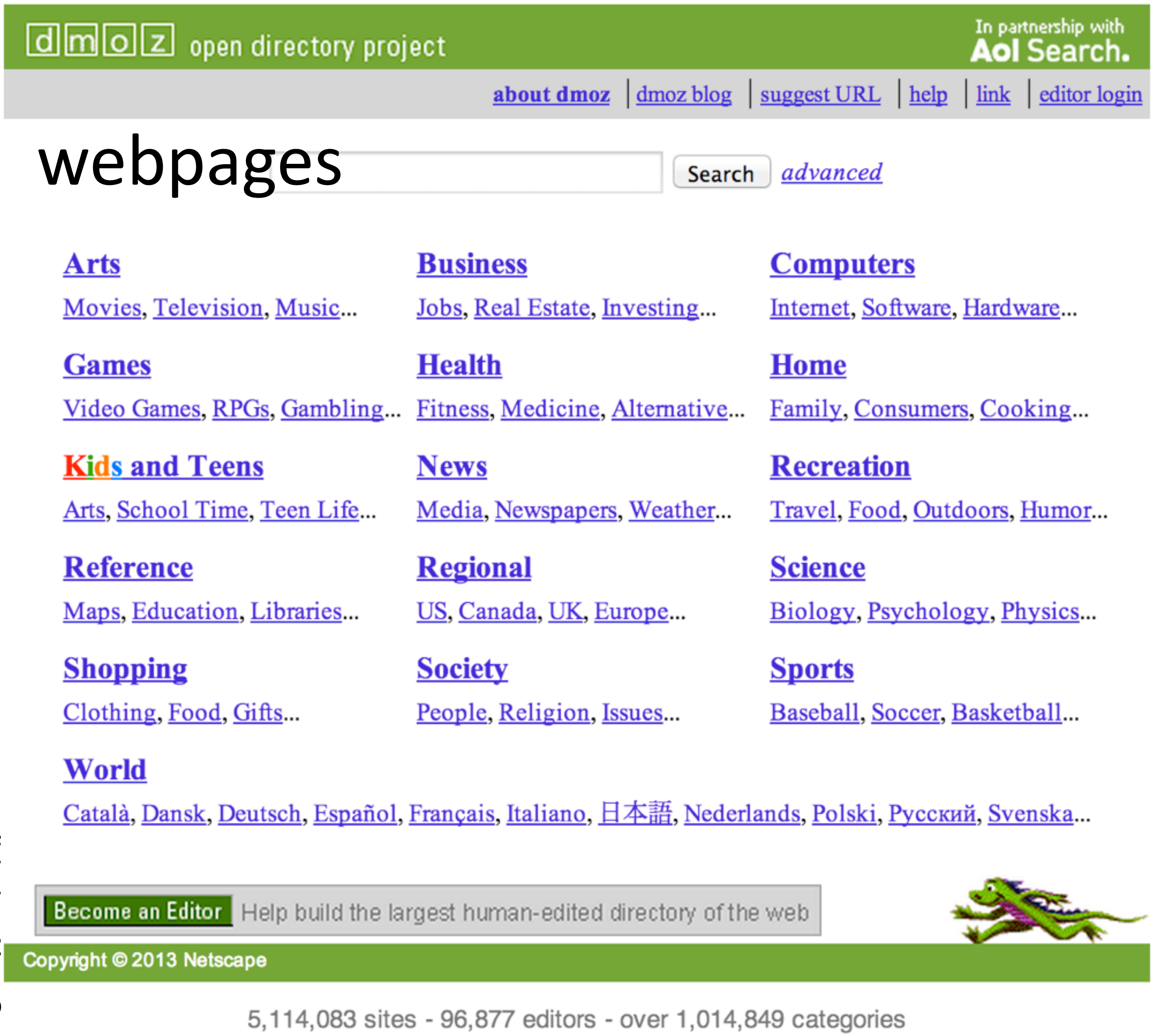
given sequence

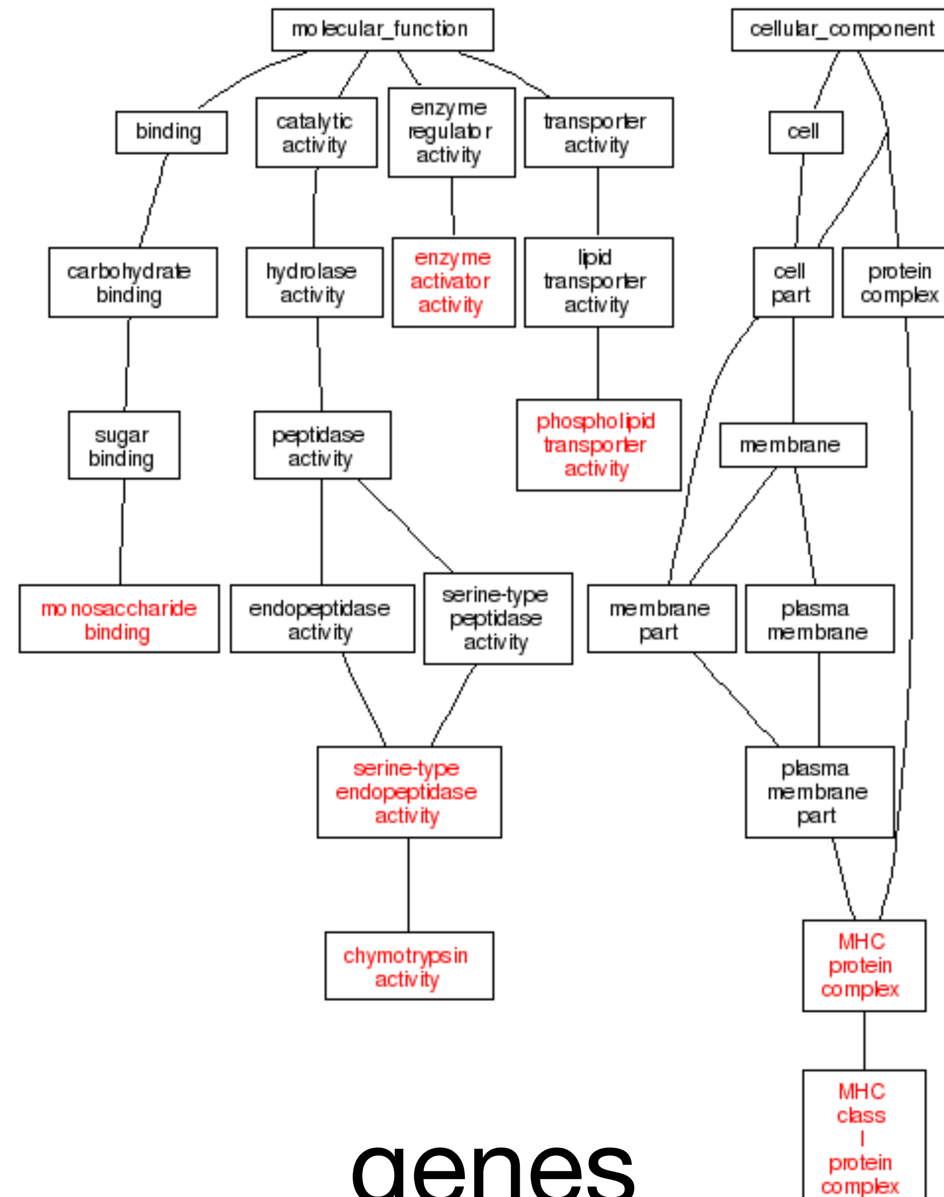
gene finding
speech recognition
activity segmentation
named entities

slide by Alex Smola

Ontology

webpages





genes

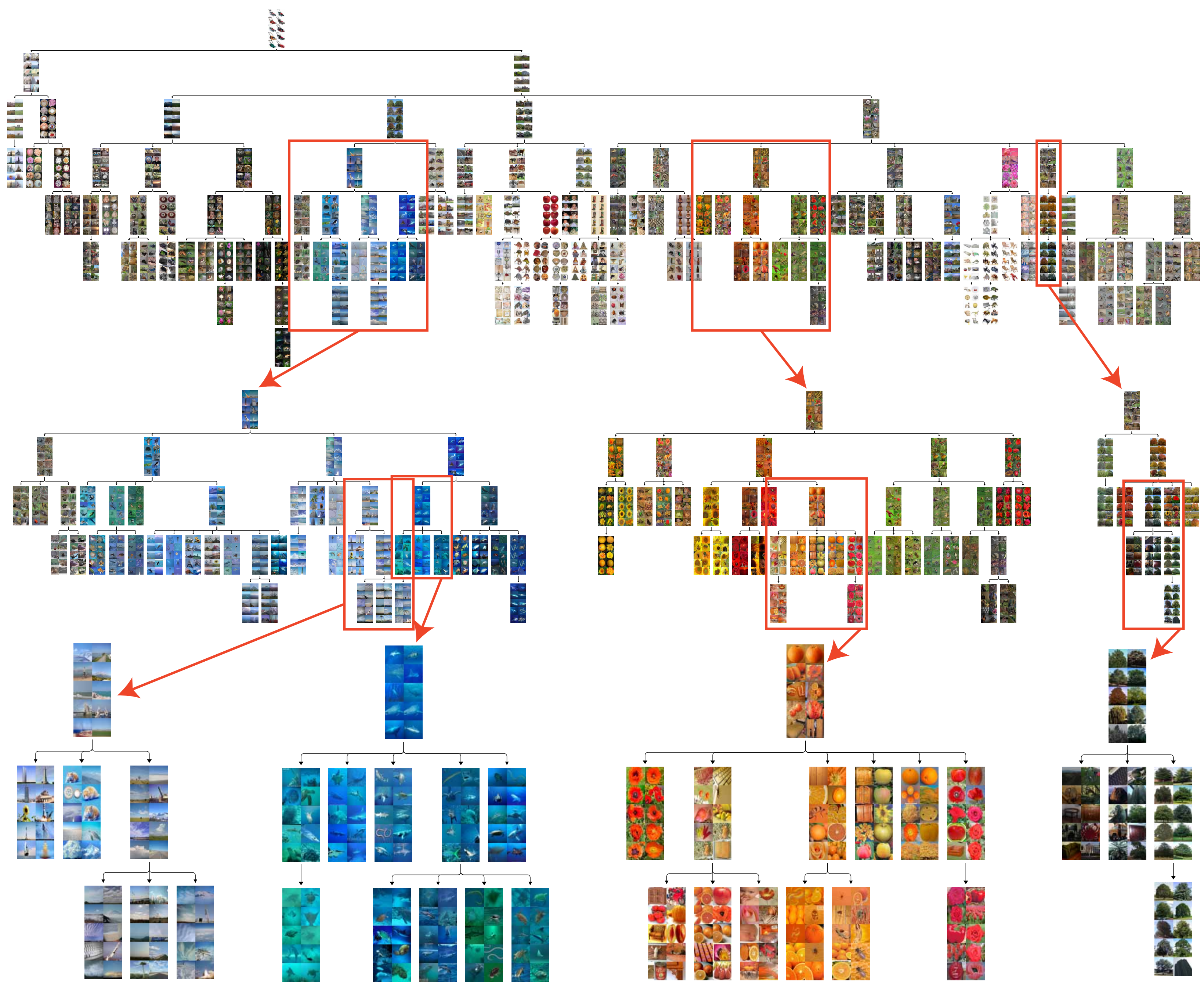
slide by Alex Smola

Prediction

slide by Alex Smola

tomorrow's stock price

Unsupervised Learning



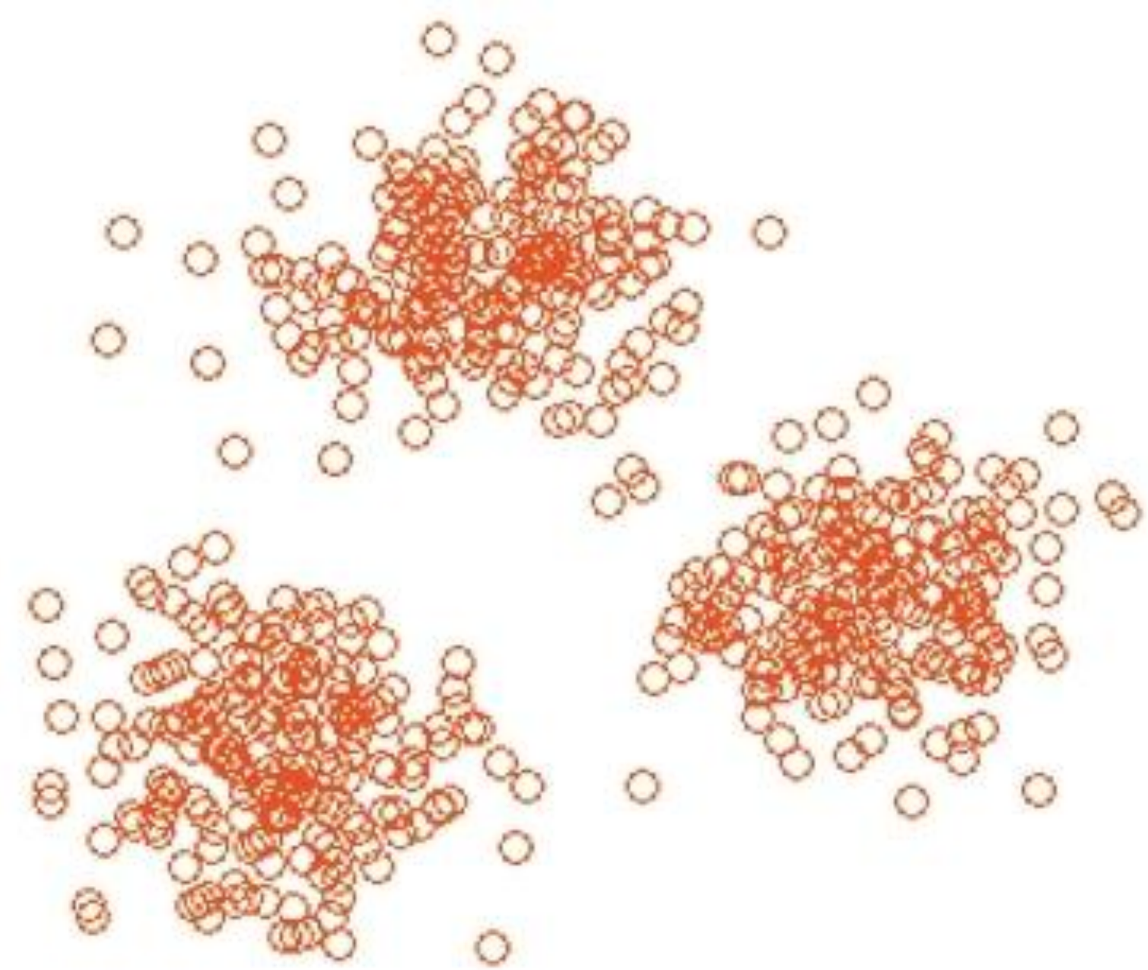
slide by Alex Smola

Unsupervised Learning

- Given data x, ask a good question ... about x or about model for x
- **Clustering**
  Find a set of prototypes representing the data
- **Principal Components**
  Find a subspace representing the data
- **Sequence Analysis**
  Find a latent causal sequence for observations
  - Sequence Segmentation
  - Hidden Markov Model (discrete state)
  - Kalman Filter (continuous state)
- **Hierarchical representations**
- **Independent components / dictionary learning**
  Find (small) set of factors for observation
- **Novelty detection**
  Find the odd one out

slide by Alex Smola

Clustering

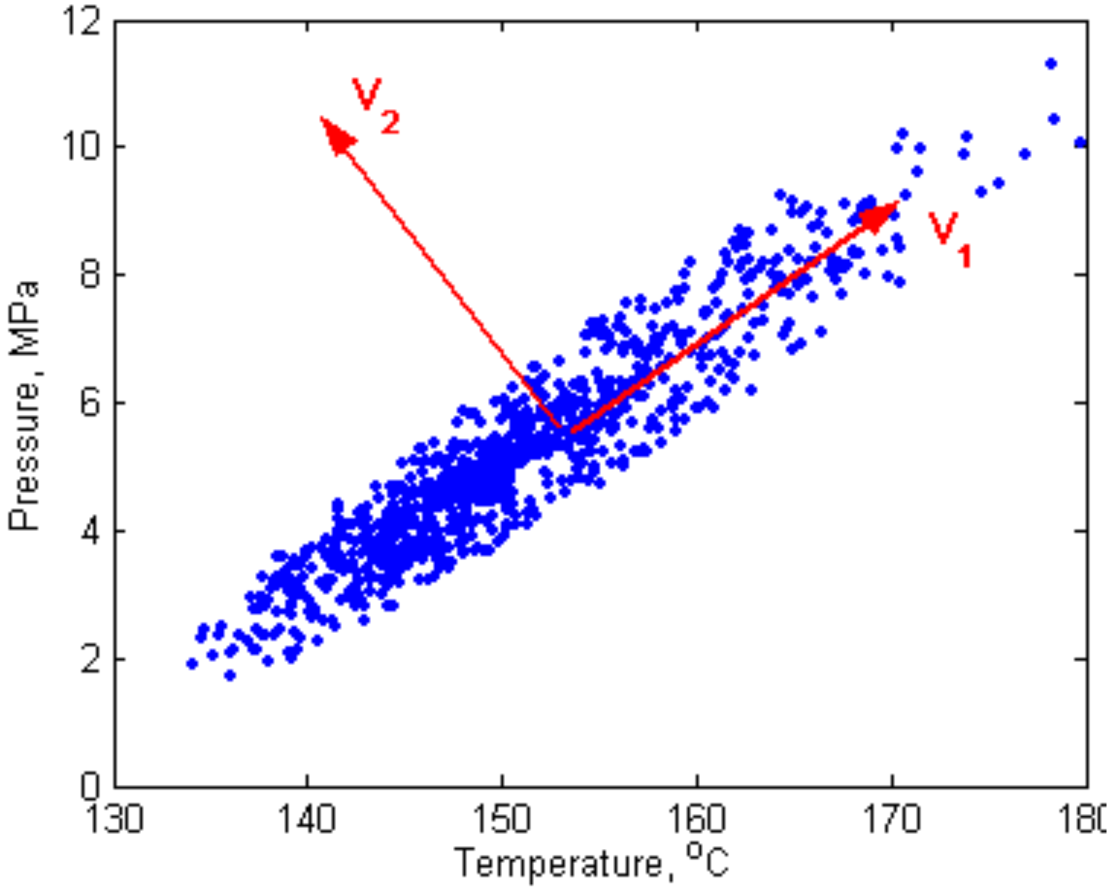


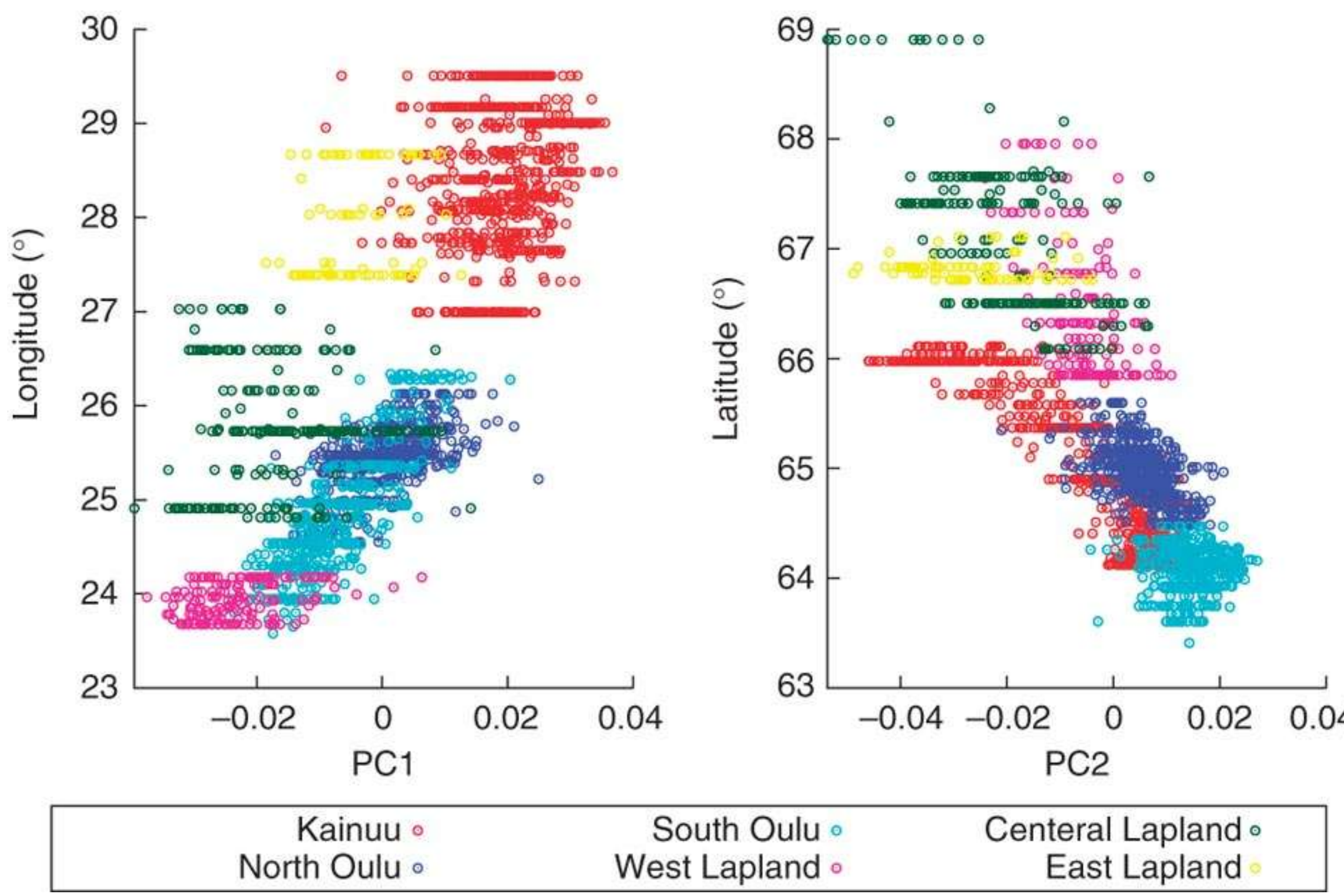
- Documents
- Users
- Webpages
- Diseases
- Pictures
- Vehicles

...

slide by Alex Smola

Principal Components

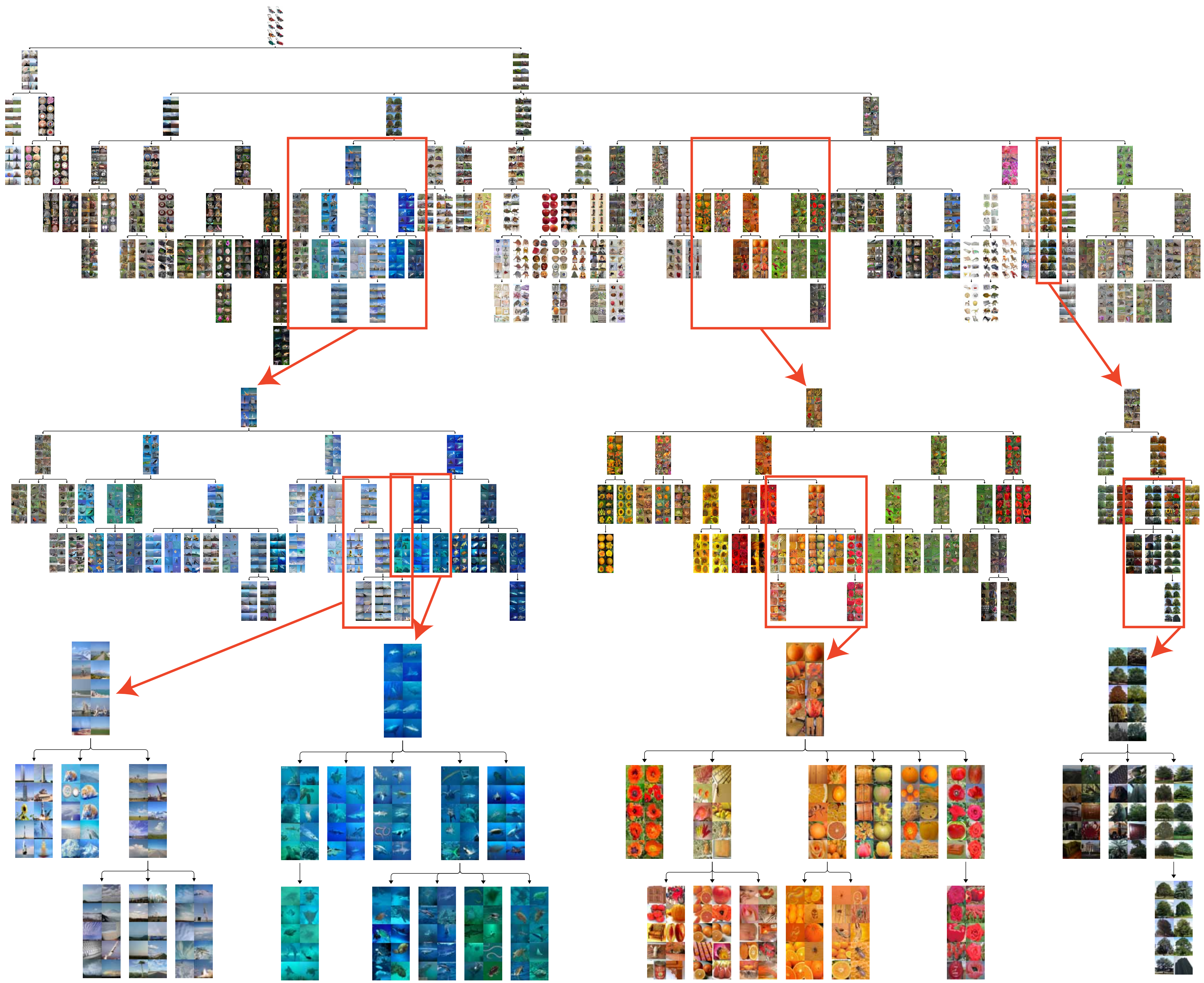




Variance component model to account for sample structure in genome-wide association studies, **Nature Genetics 2010**

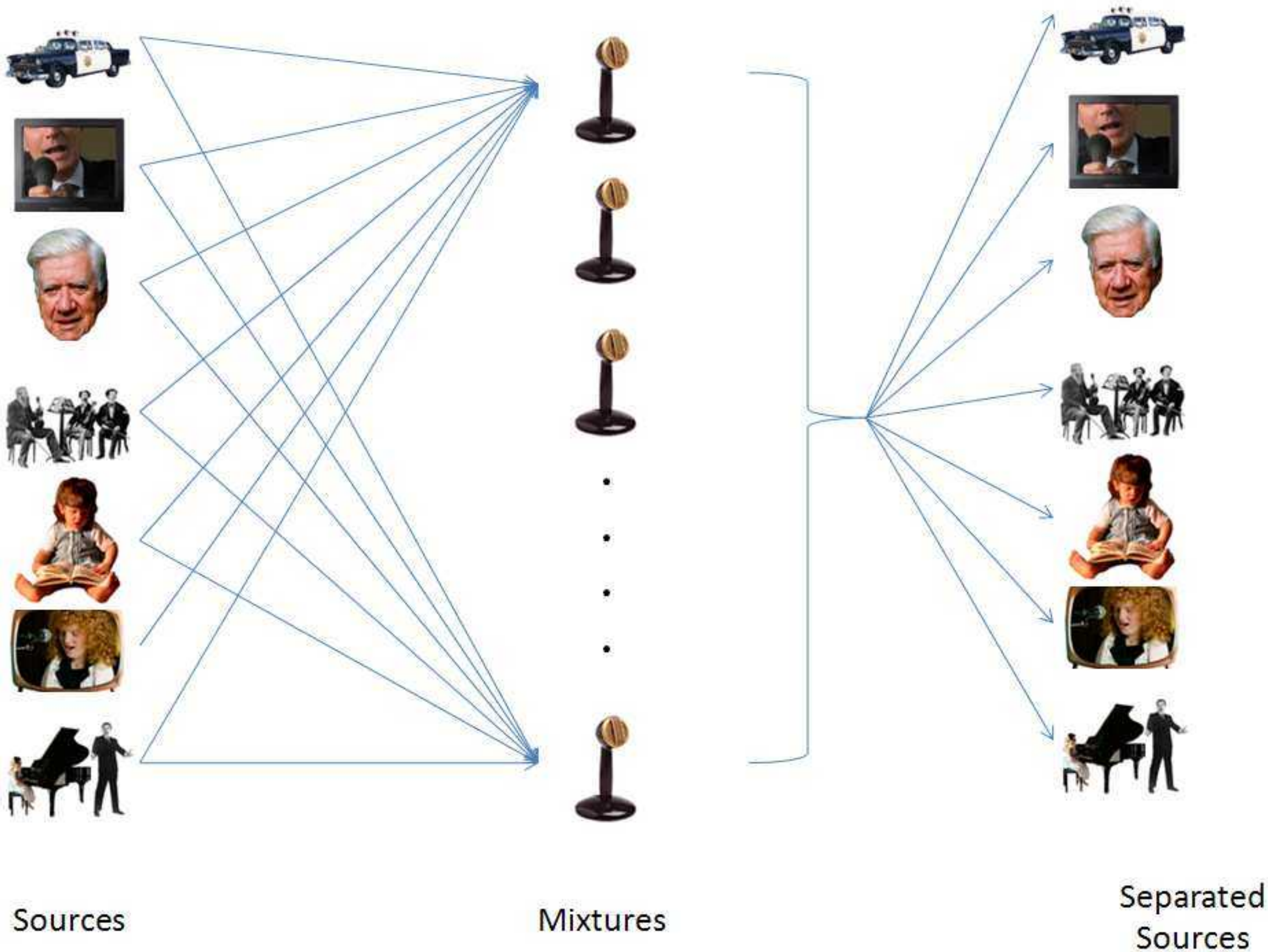
slide by Alex Smola

Hierarchical Grouping



slide by Alex Smola

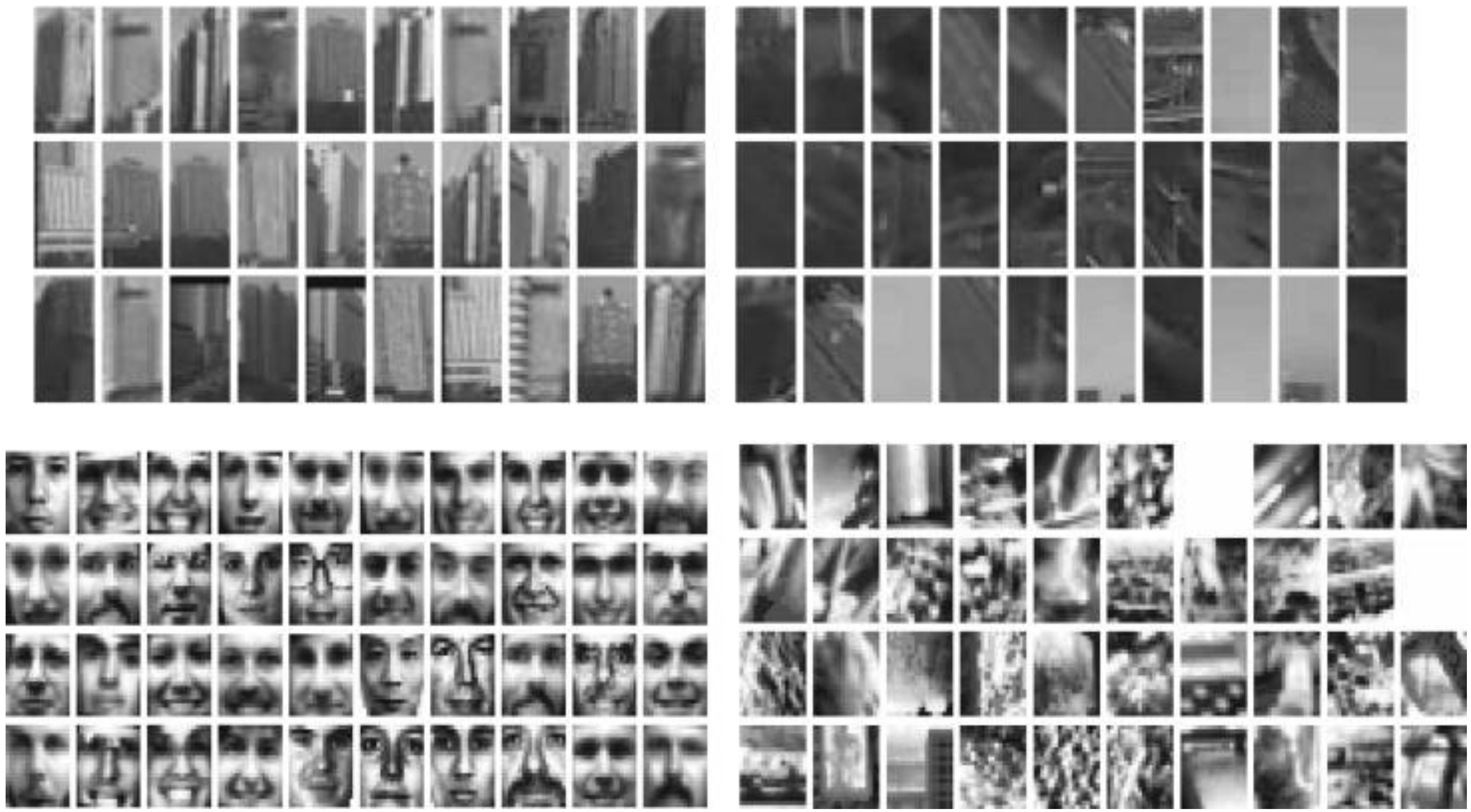
Independent Components



find them automatically

slide by Alex Smola

Novelty detection



typical atypical

Important challenges in ML

- How important is the actual learning algorithm and its tuning
- Simple versus complex algorithm
- Overfitting
- Model Selection
- Regularization

Your 1st Classifier: Nearest Neighbor Classifier

Concept Learning

- **Definition:** Acquire an operational definition of a general category of objects given *positive* and *negative* training examples.
- Also called *binary classification*, *binary supervised learning*

slide by Thorsten Joachims

Concept Learning Example

	correct (complete, partial, guessing)	**color** (yes, no)	**original** (yes, no)	**presentation** (clear, unclear, cryptic)	**binder** (yes, no)	**A+**
1	complete	yes	yes	clear	no	yes
2	complete	no	yes	clear	no	yes
3	partial	yes	no	unclear	no	no
4	complete	yes	yes	clear	yes	yes

- Instance Space X: Set of all possible objects describable by attributes (often called *features*).
- Concept c : Subset of objects from X (c is unknown).
- Target Function f : Characteristic function indicating membership in c based on attributes (i.e. *label*) (f is unknown).
- Training Data S : Set of instances labeled with target function.

slide by Thorsten Joachims

Concept Learning as Learning A Binary Function

- **Task**
  – Learn (to imitate) a function $f : X \rightarrow \{+1,-1\}$

- **Training Examples**
  – Learning algorithm is given the correct value of the function for particular inputs → training examples
  – An example is a pair (x, y), where x is the input and $y = f(x)$ is the output of the target function applied to x.

- **Goal**
  – Find a function

$$h: X \rightarrow \{+1,-1\}$$

  that approximates

$$f: X \rightarrow \{+1,-1\}$$

  as well as possible.

slide by Thorsten Joachims

Supervised Learning

- **Task**
  – Learn (to imitate) a function $f : X \rightarrow Y$
- **Training Examples**
  – Learning algorithm is given the correct value of the function for particular inputs → training examples
  – An example is a pair $(x, f(x))$, where x is the input and $y=f(x)$ is the output of the target function applied to x.
- **Goal**
  – Find a function

$$h: X \rightarrow Y$$

  that approximates

$$f: X \rightarrow Y$$

  as well as possible.

slide by Thorsten Joachims

Supervised / Inductive Learning

- Given
  - examples of a function $(x, f(x))$

- Predict function $f(x)$ for new examples x
  - Discrete $f(x)$: Classification
  - Continuous $f(x)$: Regression
  - $f(x) = \text{Probability}(x)$: Probability estimation

slide by Thorsten Joachims

Image Classification: a core task in Computer Vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

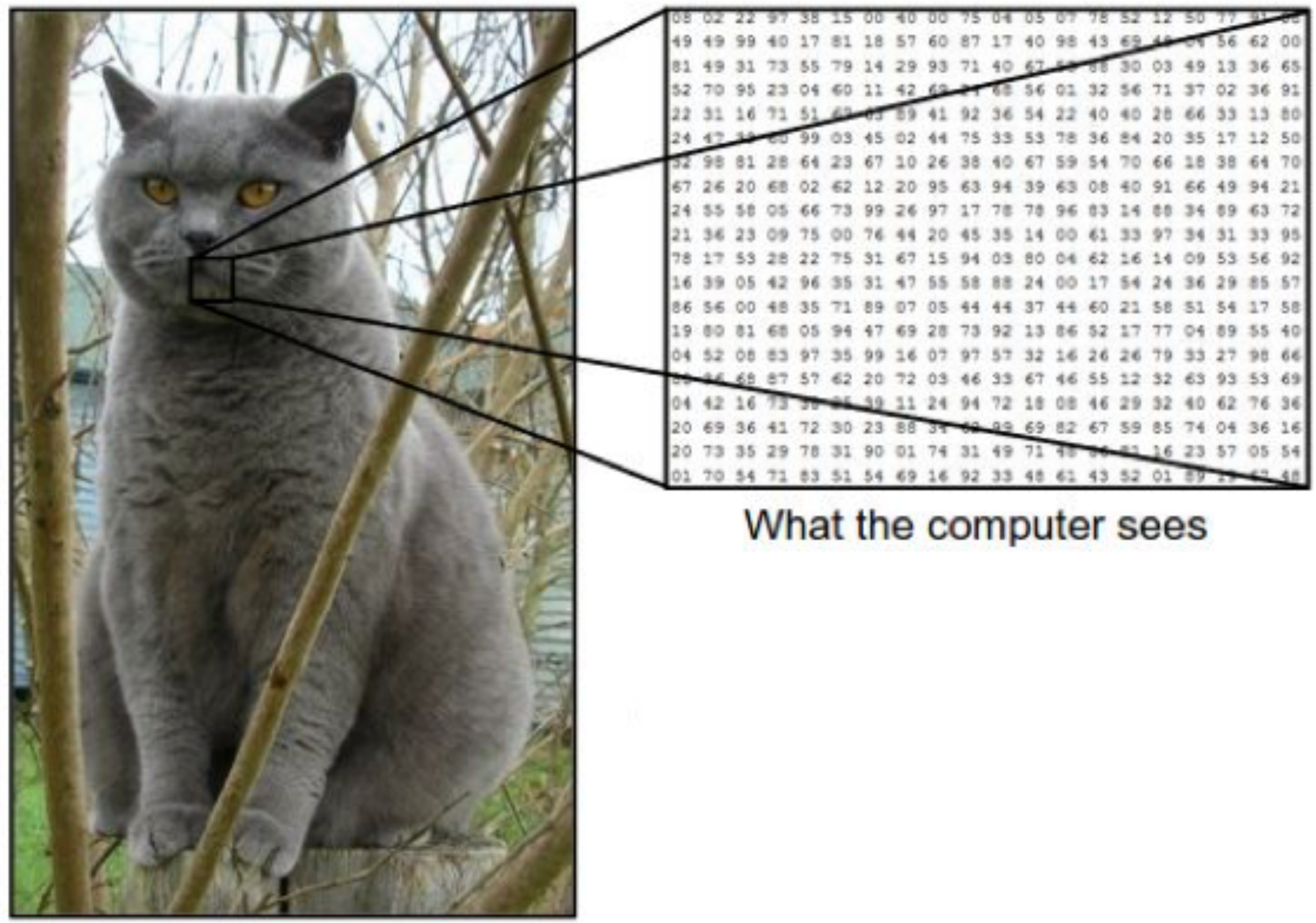
slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

The problem: semantic gap

Images are represented as 3D arrays of numbers, with integers between [0, 255].

E.g.
300 x 100 x 3

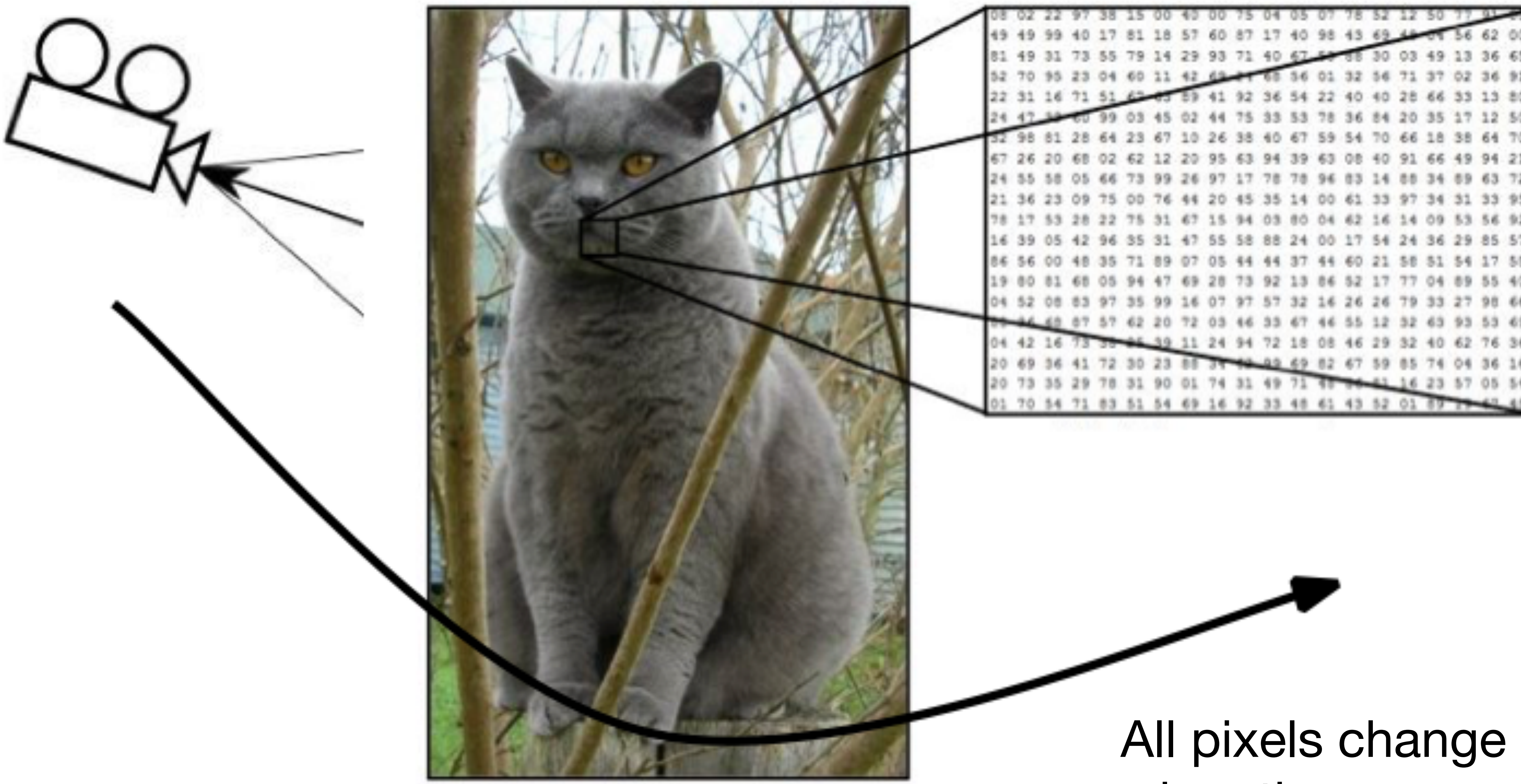
(3 for 3 color channels RGB)



What the computer sees

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Challenges: Viewpoint Variation



All pixels change when the camera moves!

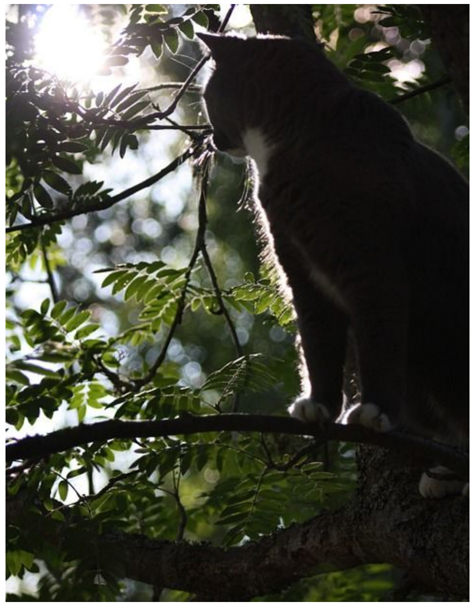
slide by Fei-Fei Li & Justin Johnson & Danfei Xu

Challenges: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

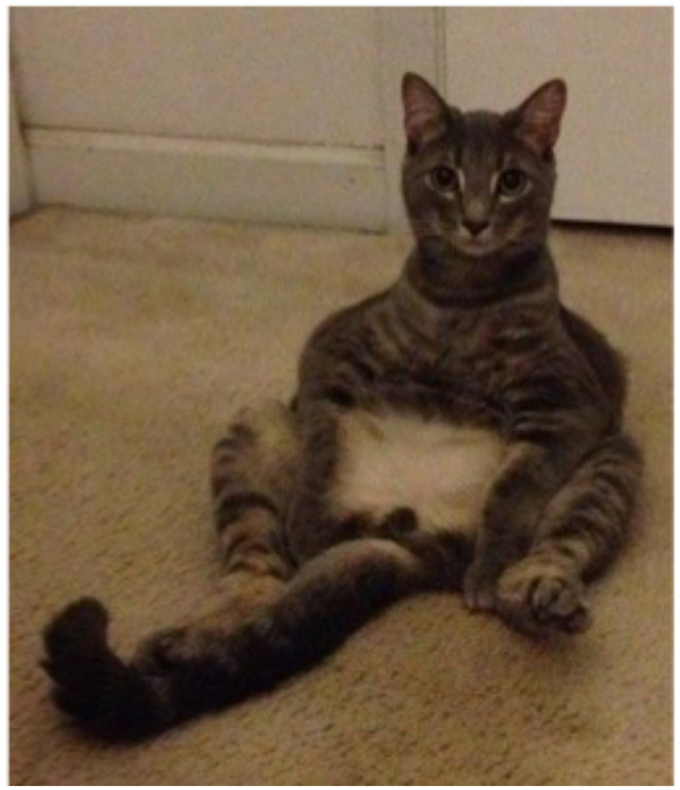
This image is CC0 1.0 public domain

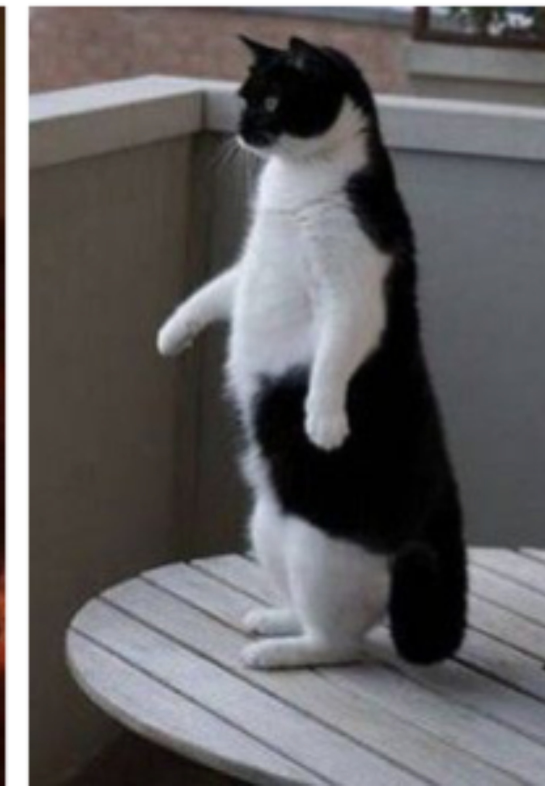


This image is CC0 1.0 public domain

slide by Fei-Fei Li & Justin Johnson & Danfei Xu

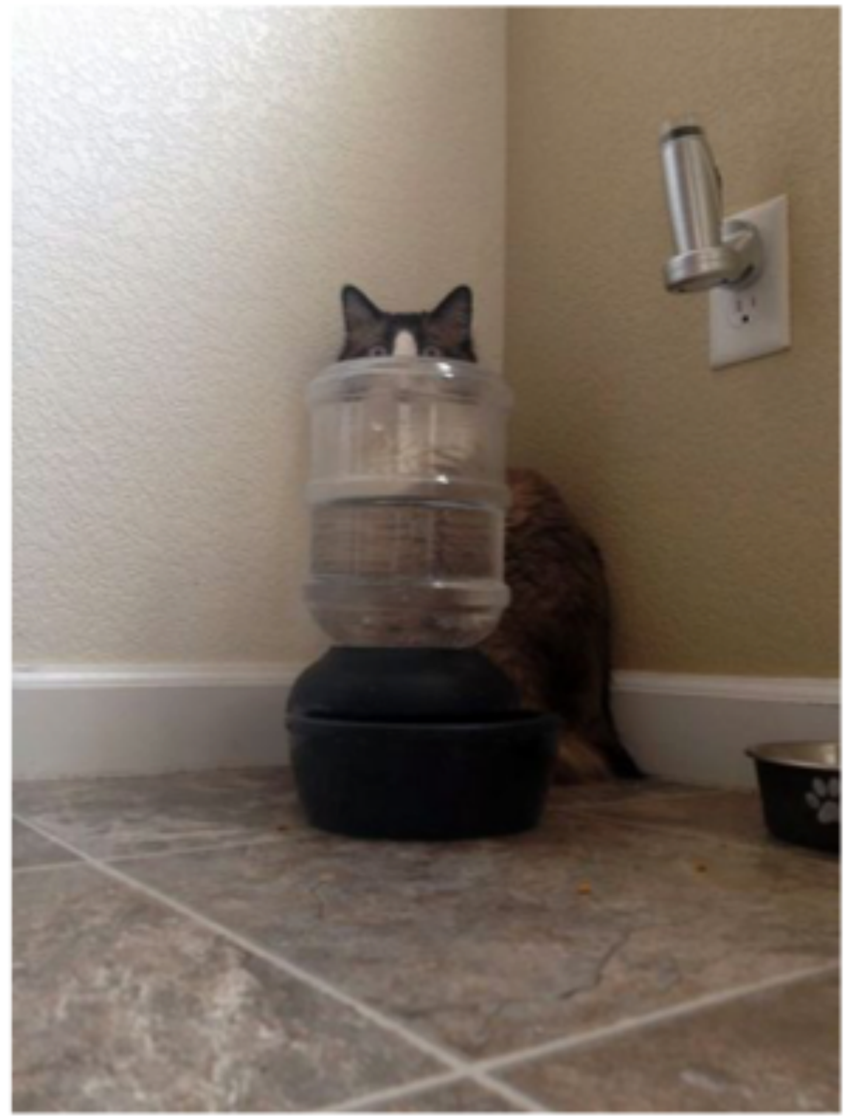
Challenges: Deformation



slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Challenges: Occlusion



slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Challenges: Background clutter

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Challenges: Intraclass variation

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

An image classifier

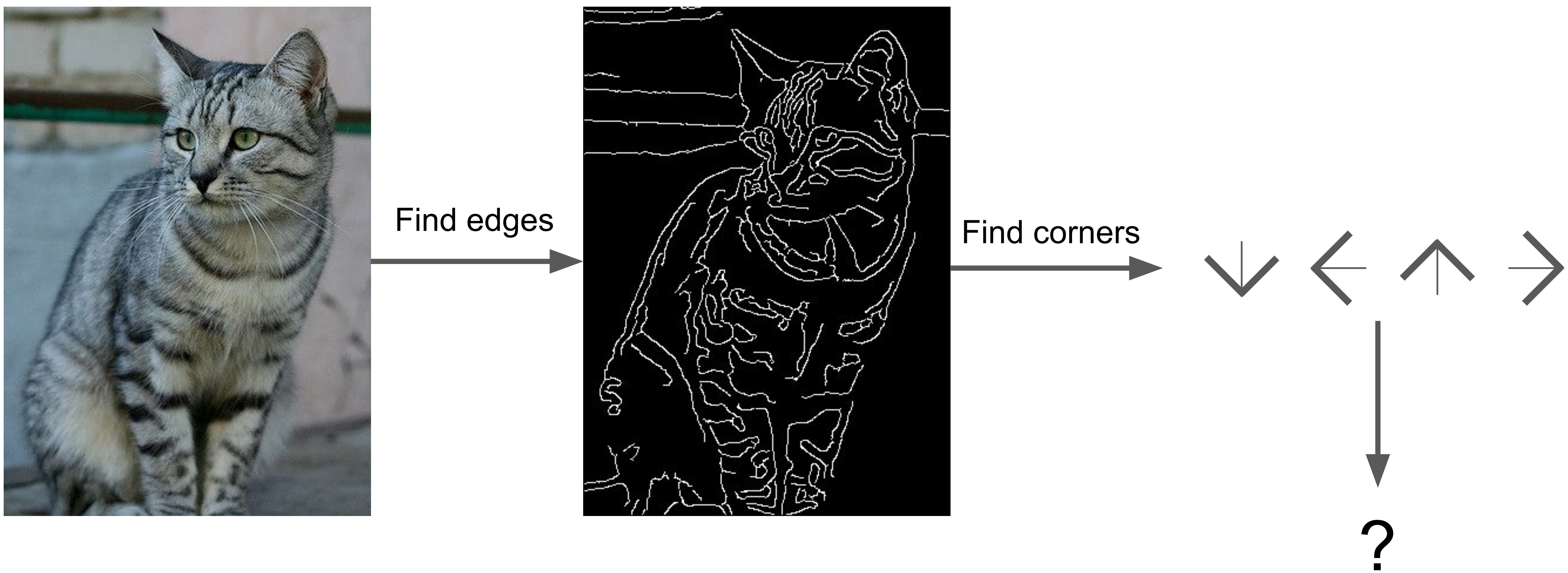
```
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike *e.g.* sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Attempts have been made



slide by Fei-Fei Li & Justin Johnson & Danfei Xu

Data-driven approach:

1. Collect a dataset of images and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

```python
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

Example training set

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

First classifier: **Nearest Neighbor Classifier**

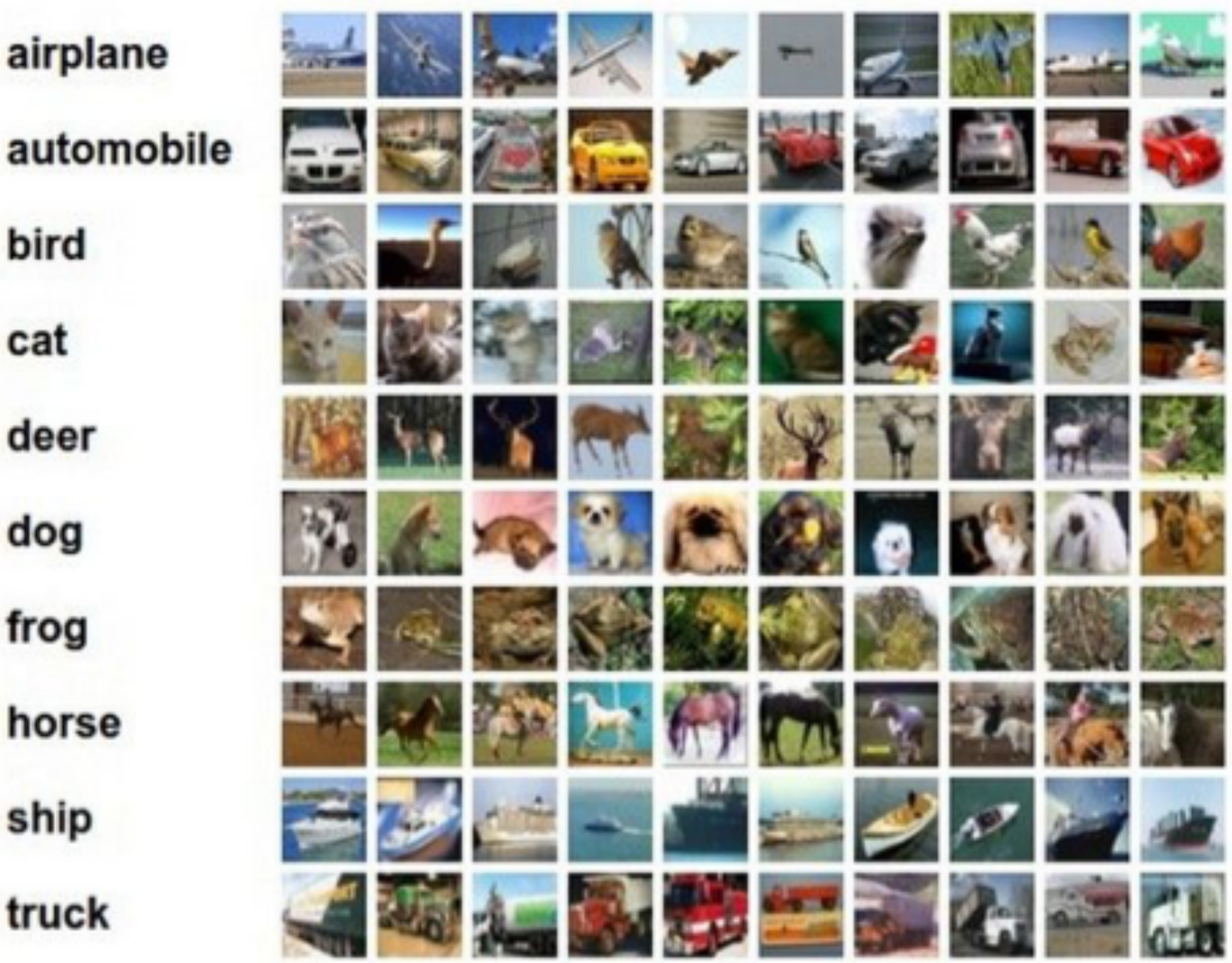
```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

Memorize all training images and their labels

Predict the label of the most similar training image

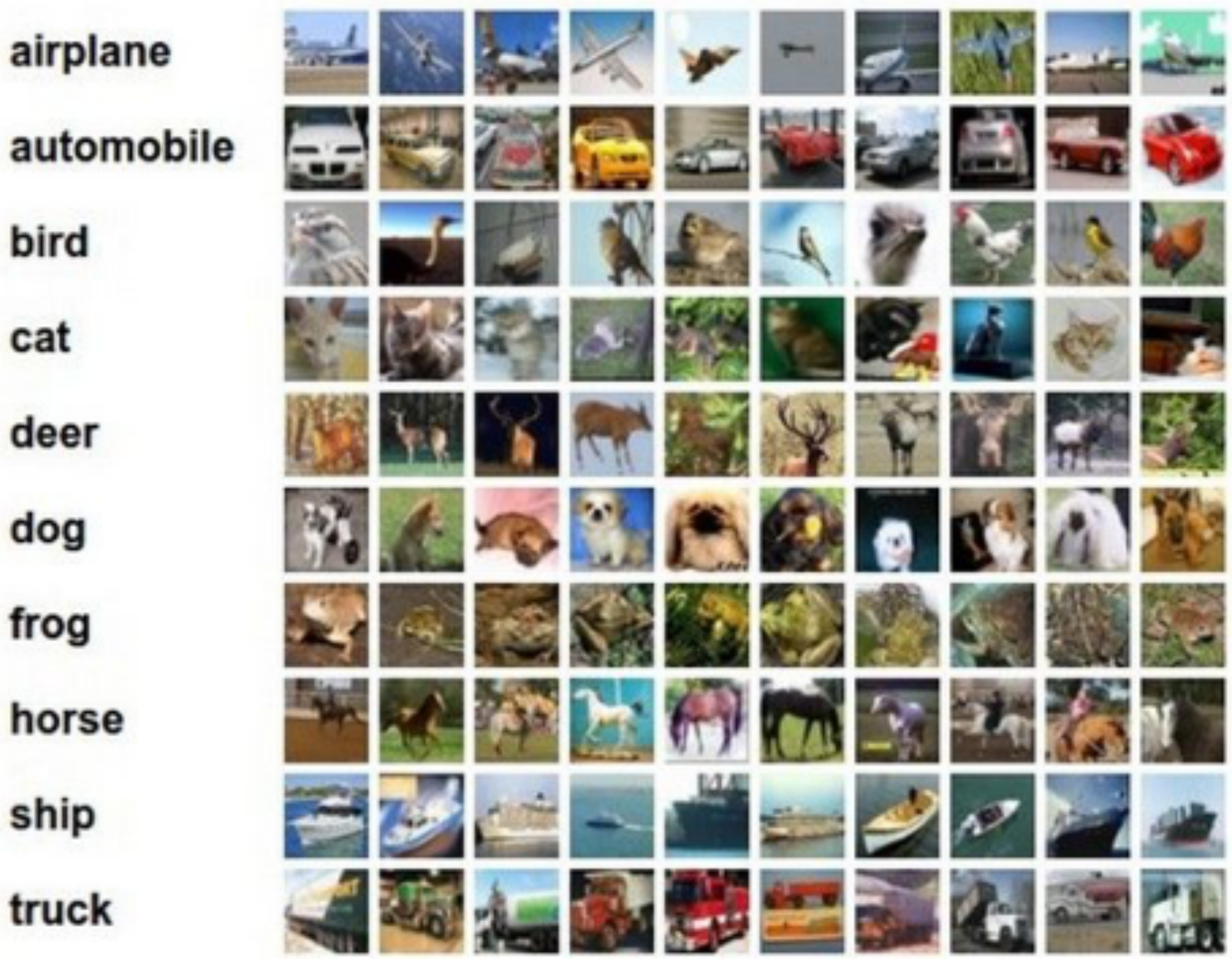
slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Example dataset: **CIFAR-10**
10 labels
50,000 training images, each image is tiny: 32x32
10,000 test images.



slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.



For every test image (first column), examples of nearest neighbors in rows

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

How do we compare the images? What is the **distance metric**?

L1 distance: $$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image — training image — pixel-wise absolute value differences

$$\left| \begin{array}{|c|c|c|c|} \hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline \end{array} - \begin{array}{|c|c|c|c|} \hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline \end{array} \right| = \begin{array}{|c|c|c|c|} \hline 46 & 12 & 14 & 1 \\ \hline 82 & 13 & 39 & 33 \\ \hline 12 & 10 & 0 & 30 \\ \hline 2 & 32 & 22 & 108 \\ \hline \end{array} \xrightarrow{\text{add}} 456$$

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Nearest Neighbor classifier

```
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

memorize training data

Nearest Neighbor classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

for every test image:
- find closest train image with L1 distance
- predict the label of nearest training image

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

slide by Fei-Fei Li & Justin Johnson & Danfei Xu

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

A: train O(1), predict O(N)

This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok.

Aside: Approximate Nearest Neighbor

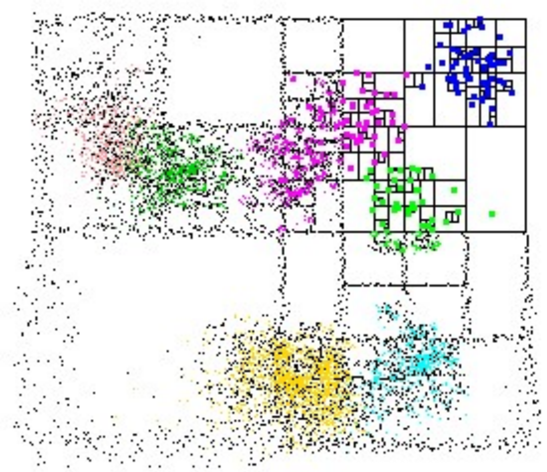
find approximate nearest neighbors quickly

ANN: A Library for Approximate Nearest Neighbor Searching

David M. Mount and Sunil Arya

Version 1.1.2

Release Date: Jan 27, 2010



What is ANN?

ANN is a library written in C++, which supports data structures and algorithms for both exact and approximate nearest neighbor searching in arbitrarily high dimensions.

In the nearest neighbor problem a set of data points in d-dimensional space is given. These points are preprocessed into a data structure, so that given any query point q, the nearest or generally k nearest points of P to q can be reported efficiently. The distance between two points can be defined in many ways. ANN assumes that distances are measured using any class of distance functions called Minkowski metrics. These include the well known Euclidean distance, Manhattan distance, and max distance.

Based on our own experience, ANN performs quite efficiently for point sets ranging in size from thousands to hundreds of thousands, and in dimensions as high as 20. (For applications in significantly higher dimensions, the results are rather spotty, but you might try it anyway.)

The library implements a number of different data structures, based on kd-trees and box-decomposition trees, and employs a couple of different search strategies.

The library also comes with test programs for measuring the quality of performance of ANN on any particular data sets, as well as programs for visualizing the structure of the geometric data structures.

FLANN - Fast Library for Approximate Nearest Neighbors

- Home
- News
- Publications
- Download
- Changelog
- Repository

What is FLANN?

FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It contains a collection of algorithms we found to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.

FLANN is written in C++ and contains bindings for the following languages: C, MATLAB and Python.

News

- (14 December 2012) Version 1.8.0 is out bringing incremental addition/removal of points to/from indexes
- (20 December 2011) Version 1.7.0 is out bringing two new index types and several other improvements.
- You can find binary installers for FLANN on the Point Cloud Library project page. Thanks to the PCL developers!
- Mac OS X users can install flann though MacPorts (thanks to Mark Moll for maintaining the Portfile)
- New release introducing an easier way to use custom distances, kd-tree implementation optimized for low dimensionality search and experimental MPI support
- New release introducing new C++ templated API, thread-safe search, save/load of indexes and more.
- The FLANN license was changed from LGPL to BSD.

How fast is it?

In our experiments we have found FLANN to be about one order of magnitude faster on many datasets (in query time), than previously available approximate nearest neighbor search software.

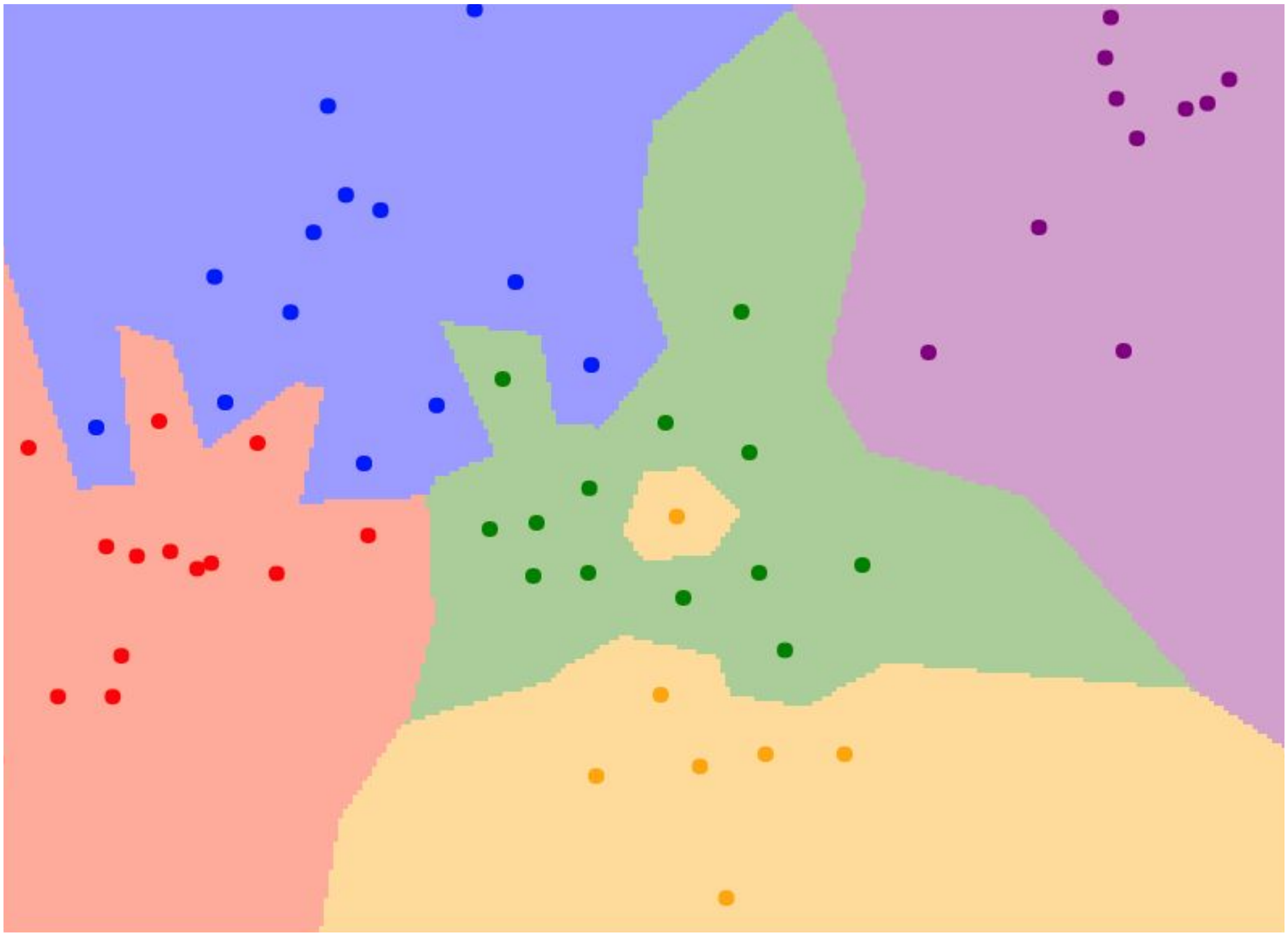
Publications

More information and experimental results can be found in the following papers:

- Marius Muja and David G. Lowe: **"Scalable Nearest Neighbor Algorithms for High Dimensional Data"**. Pattern Analysis and Machine Intelligence (PAMI), Vol. 36, 2014. [PDF] [BibTeX]
- Marius Muja and David G. Lowe: **"Fast Matching of Binary Features"**. Conference on Computer and Robot Vision (CRV) 2012. [PDF] [BibTeX]
- Marius Muja and David G. Lowe, **"Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration"**, in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009 [PDF] [BibTeX]

slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

What does Nearest Neighbor classifier look like?

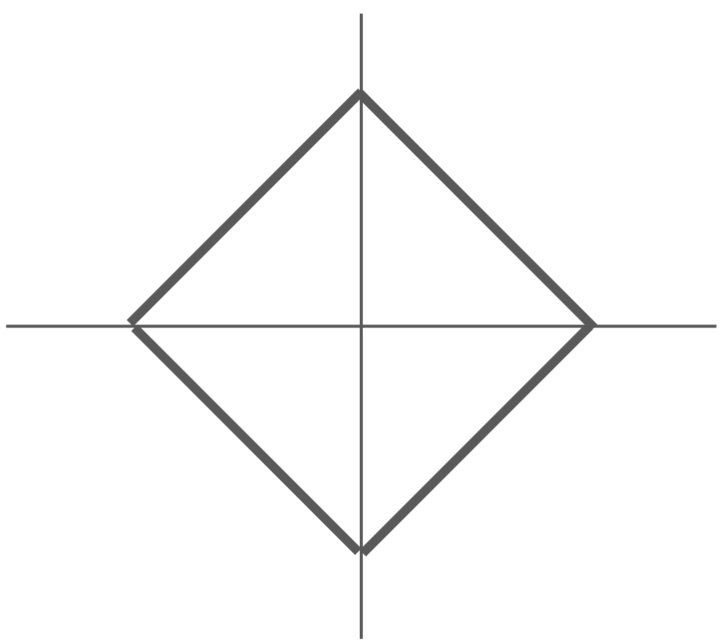


slide by Fei-Fei Li & Justin Johnson & Danfei Xu

The choice of distance is a **hyperparameter**
common choices:

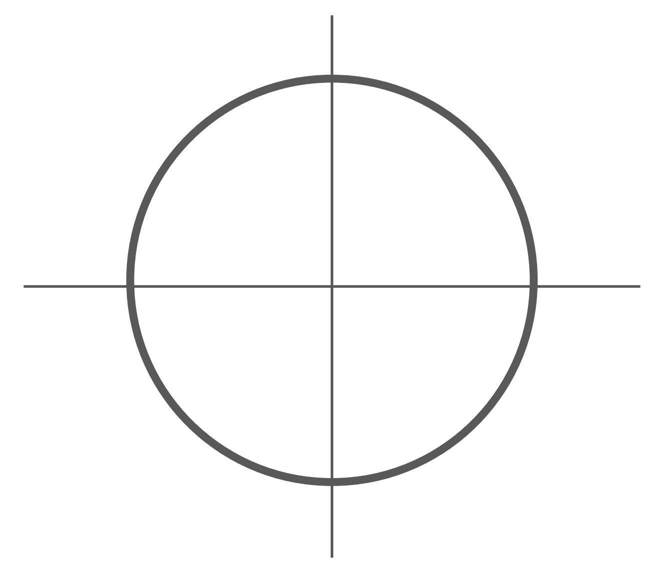
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

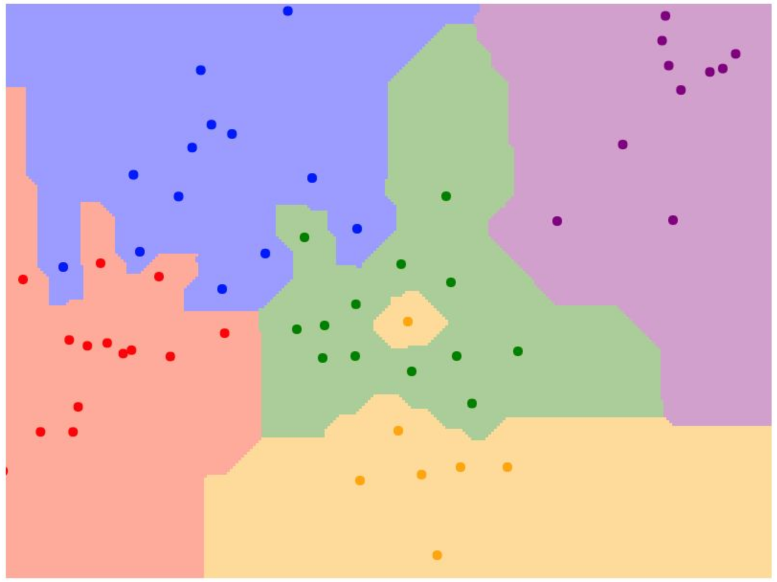


slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

The choice of distance is a **hyperparameter**
common choices:

L1 (Manhattan) distance

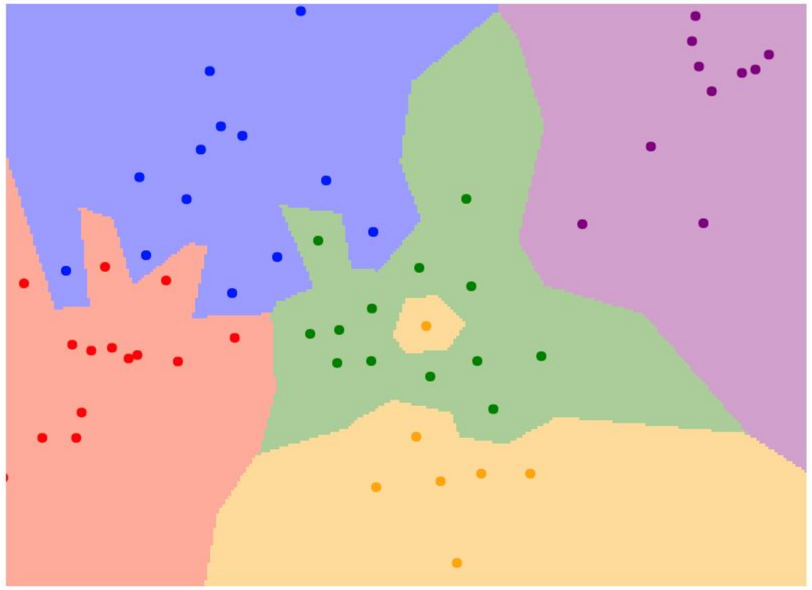
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

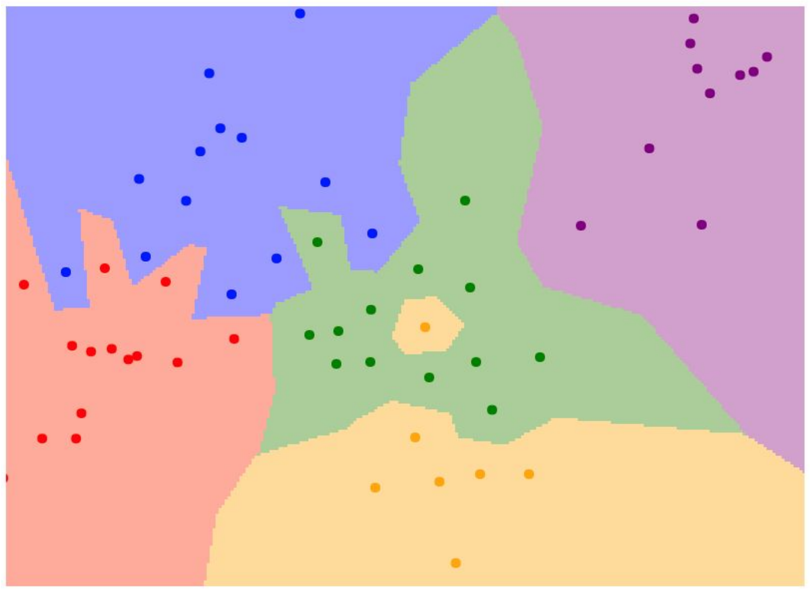


K = 1

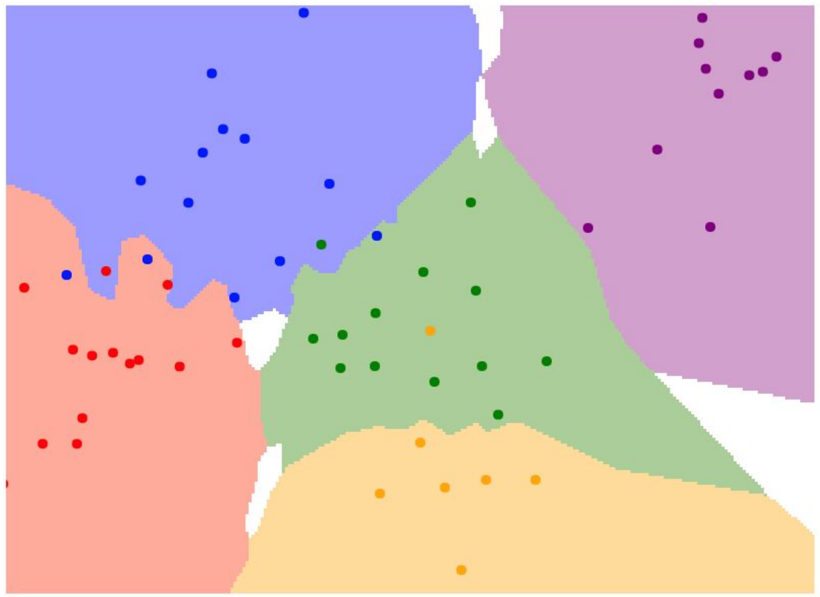
slide by Fei-Fei Li & Justin Johnson & Danfei Xu

k-Nearest Neighbor

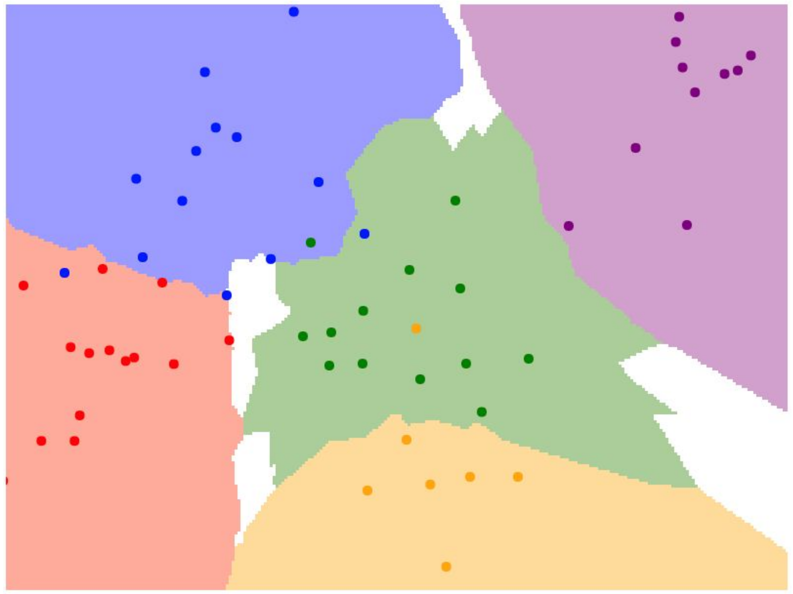
Instead of copying label from nearest neighbor, take **majority vote** from K closest points



K = 1



K = 3



K = 5

slide by Fei-Fei Li & Justin Johnson & Danfei Xu

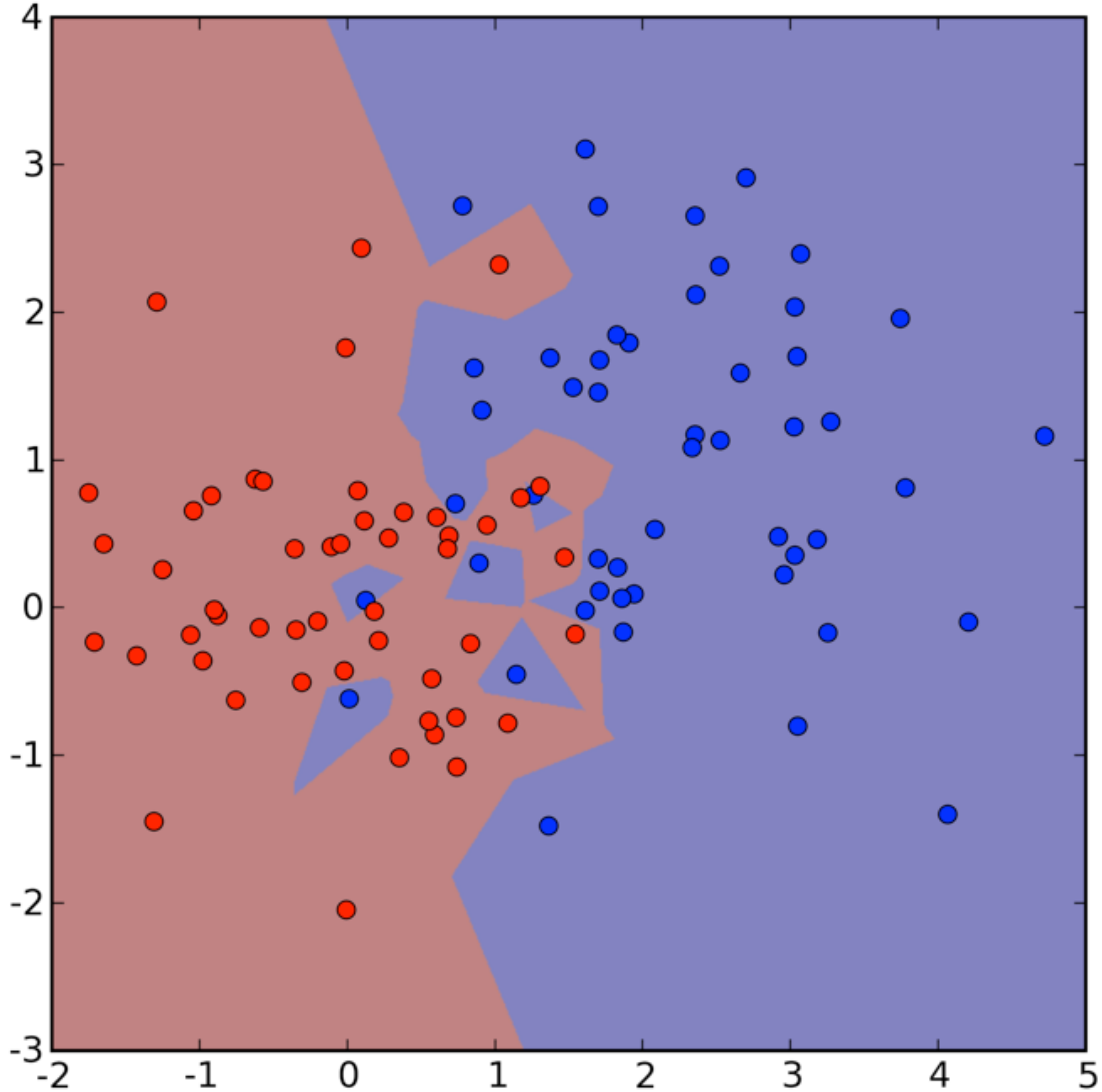
K-Nearest Neighbor (kNN)

- Given: Training data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$
  - Attribute vectors: $x_i \in X$
  - Labels: $y_i \in Y$
- Parameter:
  - Similarity function: $K : X \times X \rightarrow R$
  - Number of nearest neighbors to consider: k
- Prediction rule
  - New example x'
  - K-nearest neighbors: k train examples with largest $K(x_i, x')$

$$h(\vec{x}') = \arg\max_{y \in Y} \left\{ \sum_{i \in knn(\vec{x}')} 1_{[y_i = y]} \right\}$$

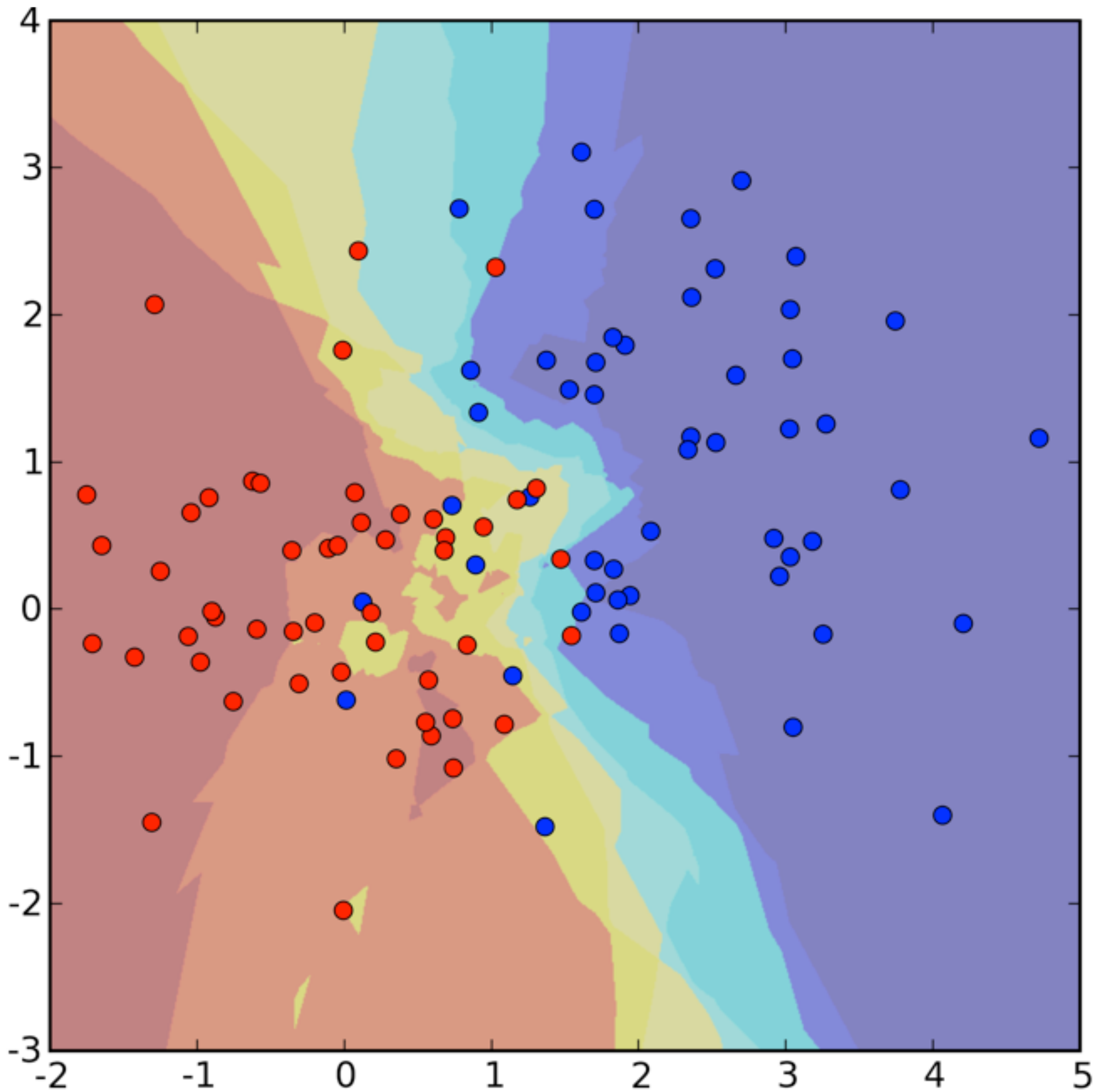
slide by Thorsten Joachims

1-Nearest Neighbor



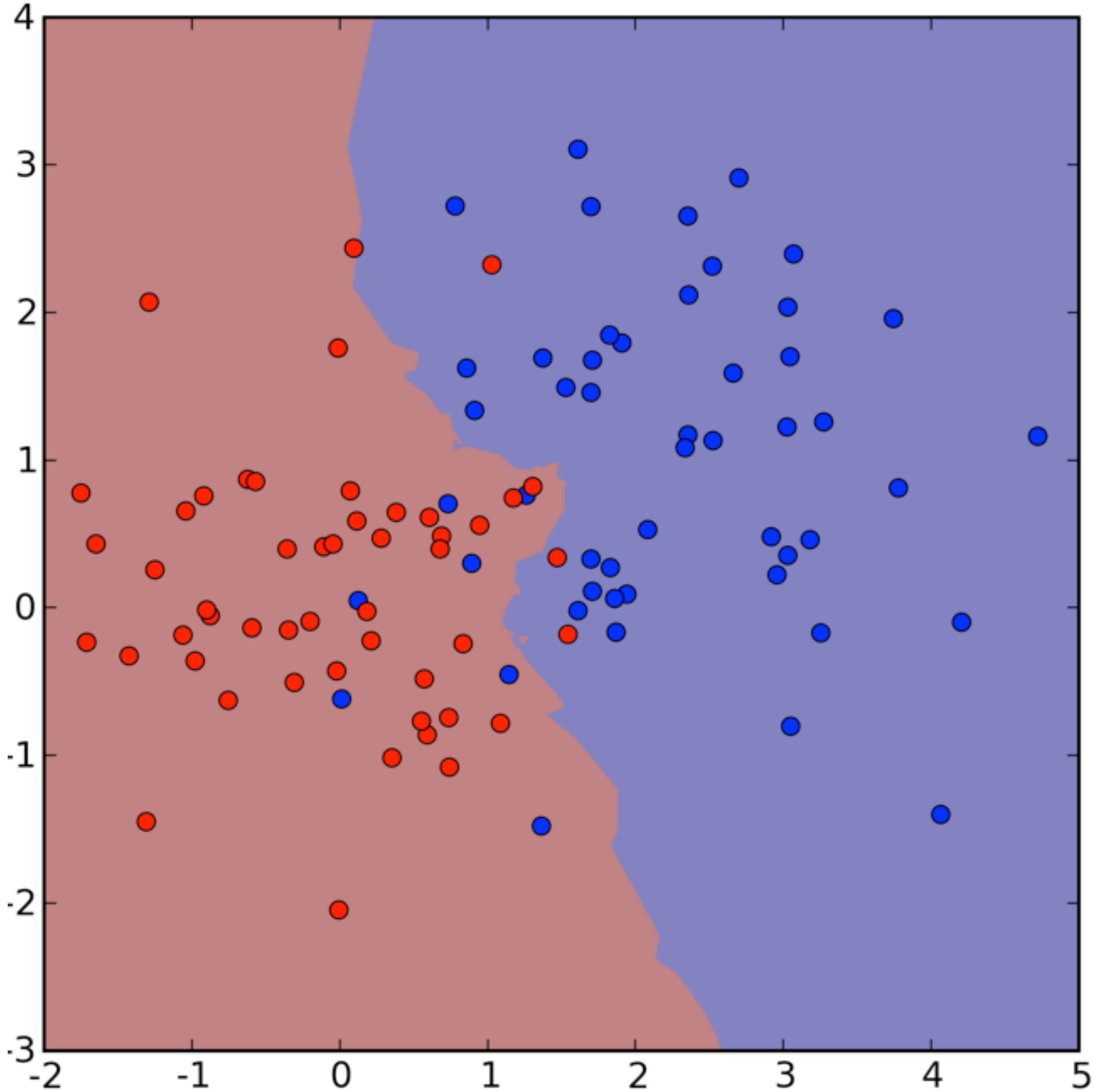
slide by Thorsten Joachims

4-Nearest Neighbors



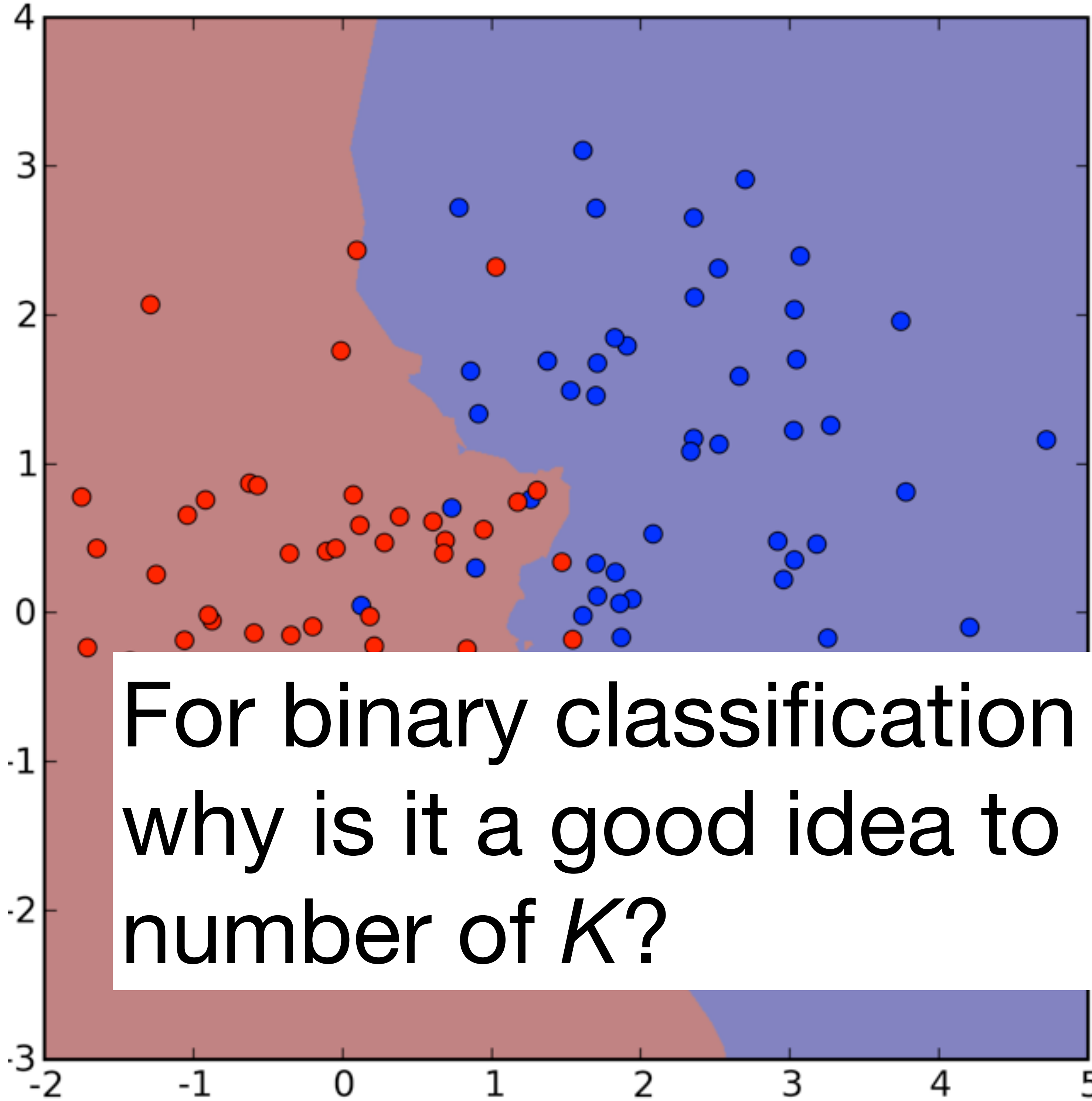
slide by Thorsten Joachims

4-Nearest Neighbors Sign



slide by Thorsten Joachims

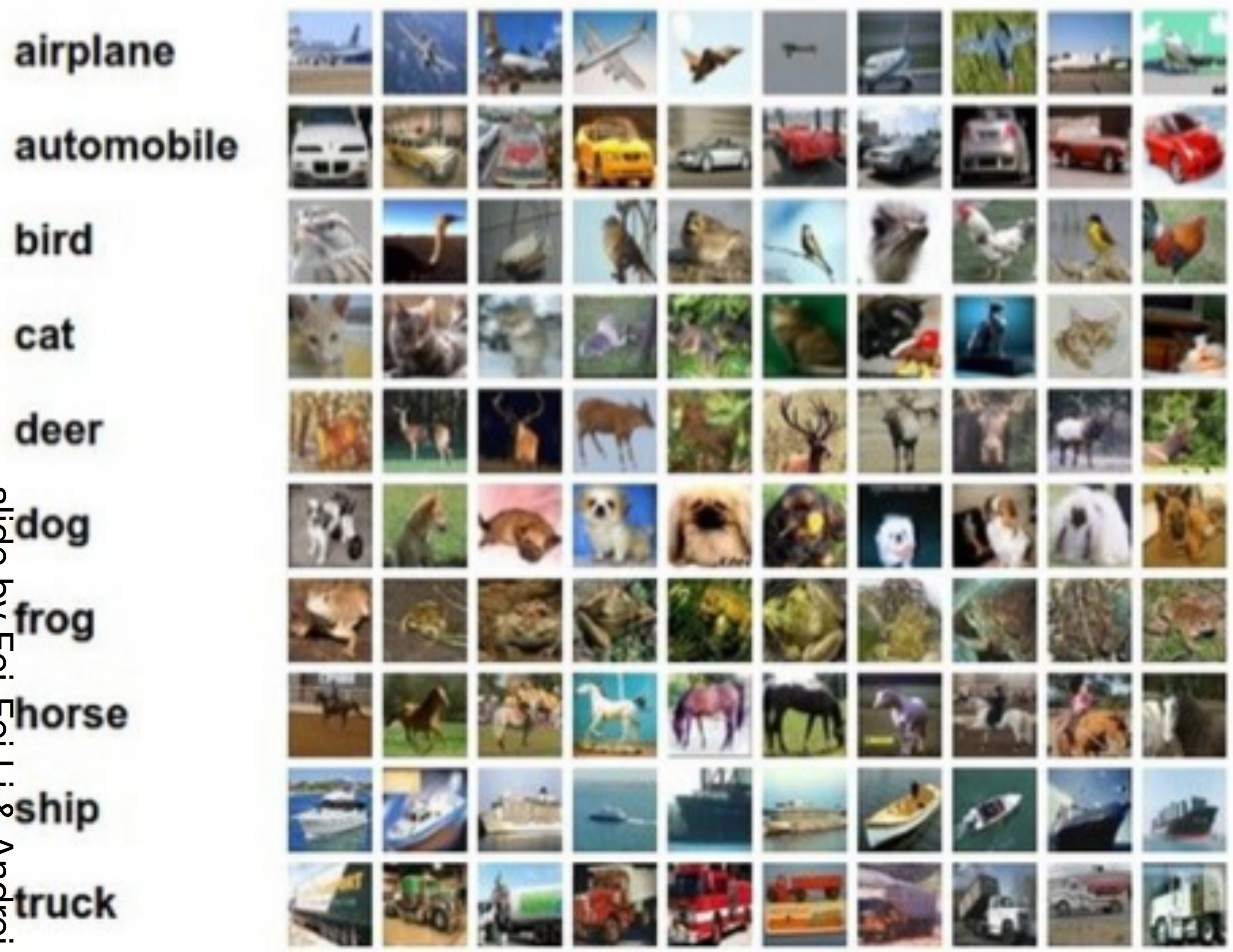
4-Nearest Neighbors Sign



For binary classification problems, why is it a good idea to use an odd number of *K*?

slide by Thorsten Joachims

Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.



For every test image (first column), examples of nearest neighbors in rows

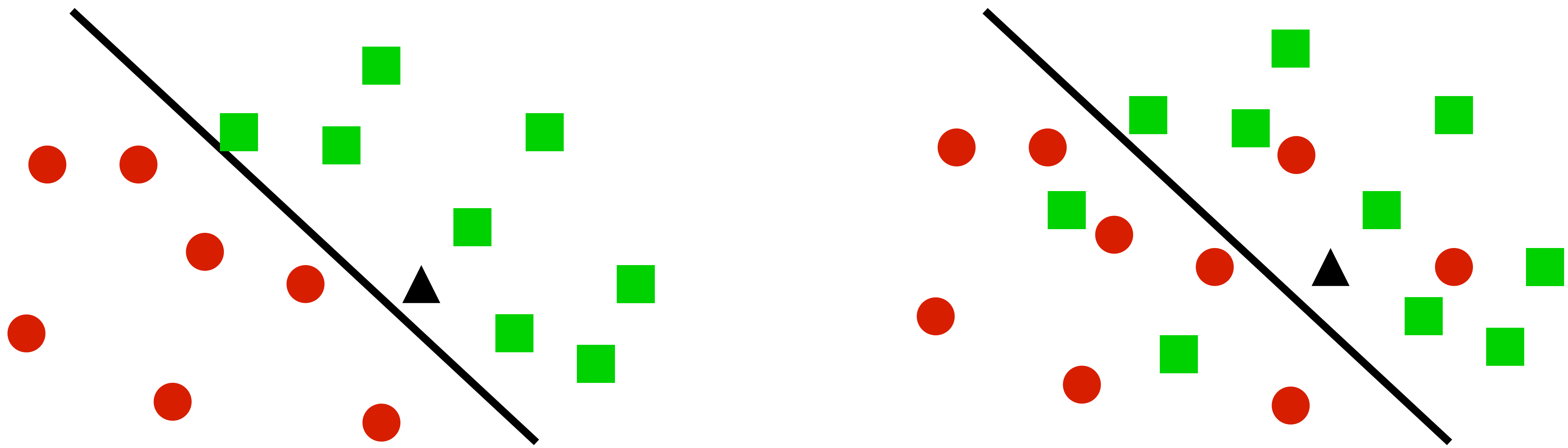
slide by Fei-Fei Li & Andrej Karpathy & Justin Johnson

What is the best **distance** to use?
What is the best value of **k** to use?

i.e. how do we set the **hyperparameters**?

We will talk about this later!

If we get more data



- 1 Nearest Neighbor
  - Converges to perfect solution if clear separation
  - Twice the minimal error rate $2p(1\text{-}p)$ for noisy problems
- k-Nearest Neighbor
  - Converges to perfect solution if clear separation (but needs more data)
  - Converges to minimal error $\min(p, 1\text{-}p)$ for noisy problems if k increases

Demo

http://vision.stanford.edu/teaching/cs231n-demos/knn/

Weighted K-Nearest Neighbor

- Given: Training data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$
  - Attribute vectors: $x_i \in X$
  - Target attribute $y_i \in Y$
- Parameter:
  - Similarity function: $K : X \times X \to R$
  - Number of nearest neighbors to consider: k
- Prediction rule
  - New example x'
  - K-nearest neighbors: k train examples with largest $K(x_i, x')$

$$h(\vec{x}') = \arg\max_{y \in Y} \left\{ \sum_{i \in knn(\vec{x}')} 1_{[y_i = y]} K(\vec{x}_i, \vec{x}') \right\}$$

More Nearest Neighbors in Visual Data

Where in the World? [Hays & Efros, CVPR 2008]

A nearest neighbor recognition example

slide by James Hays

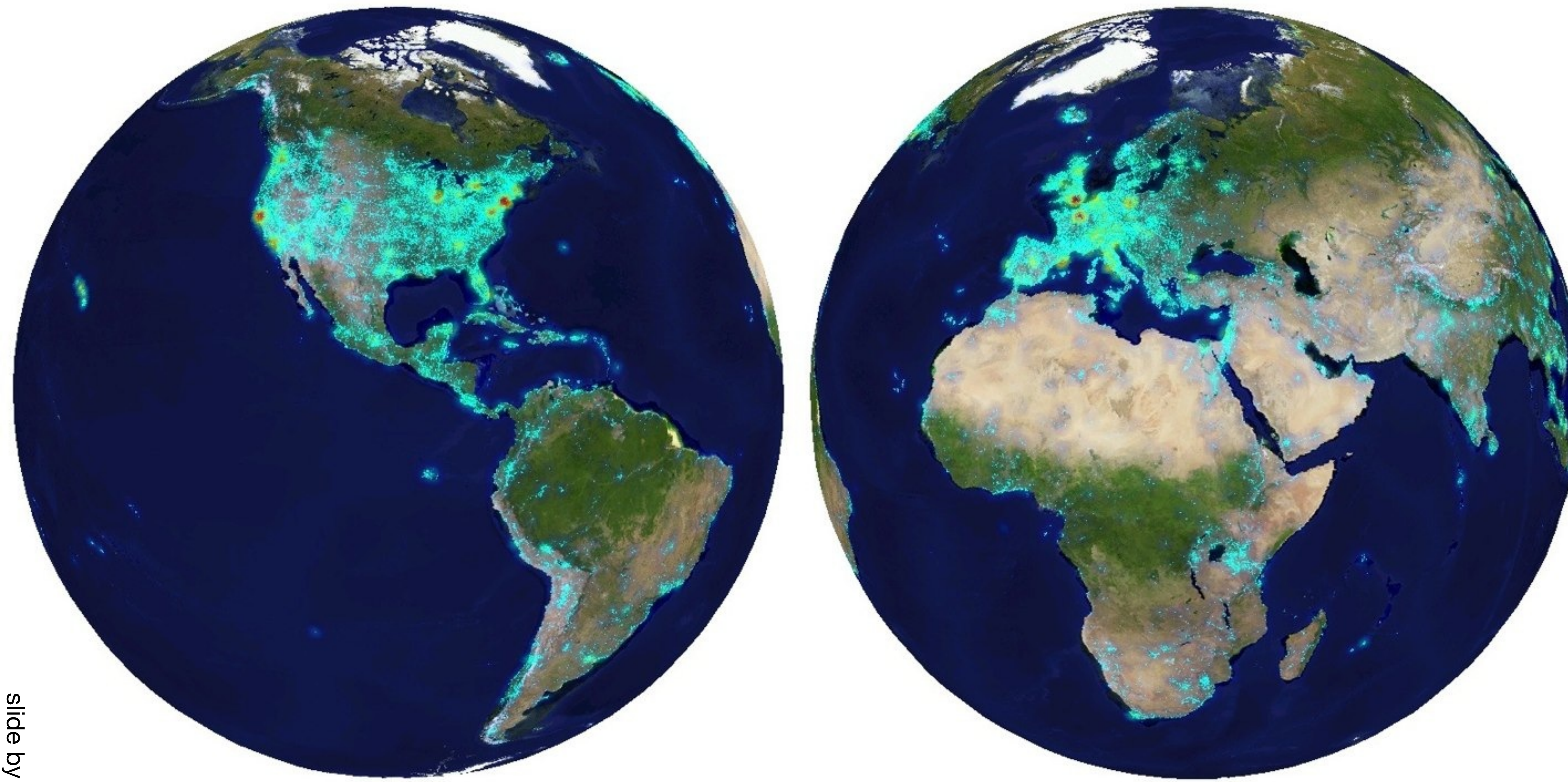
Where in the World? [Hays & Efros, CVPR 2008]

slide by James Hays

Where in the World? [Hays & Efros, CVPR 2008]

slide by James Hays

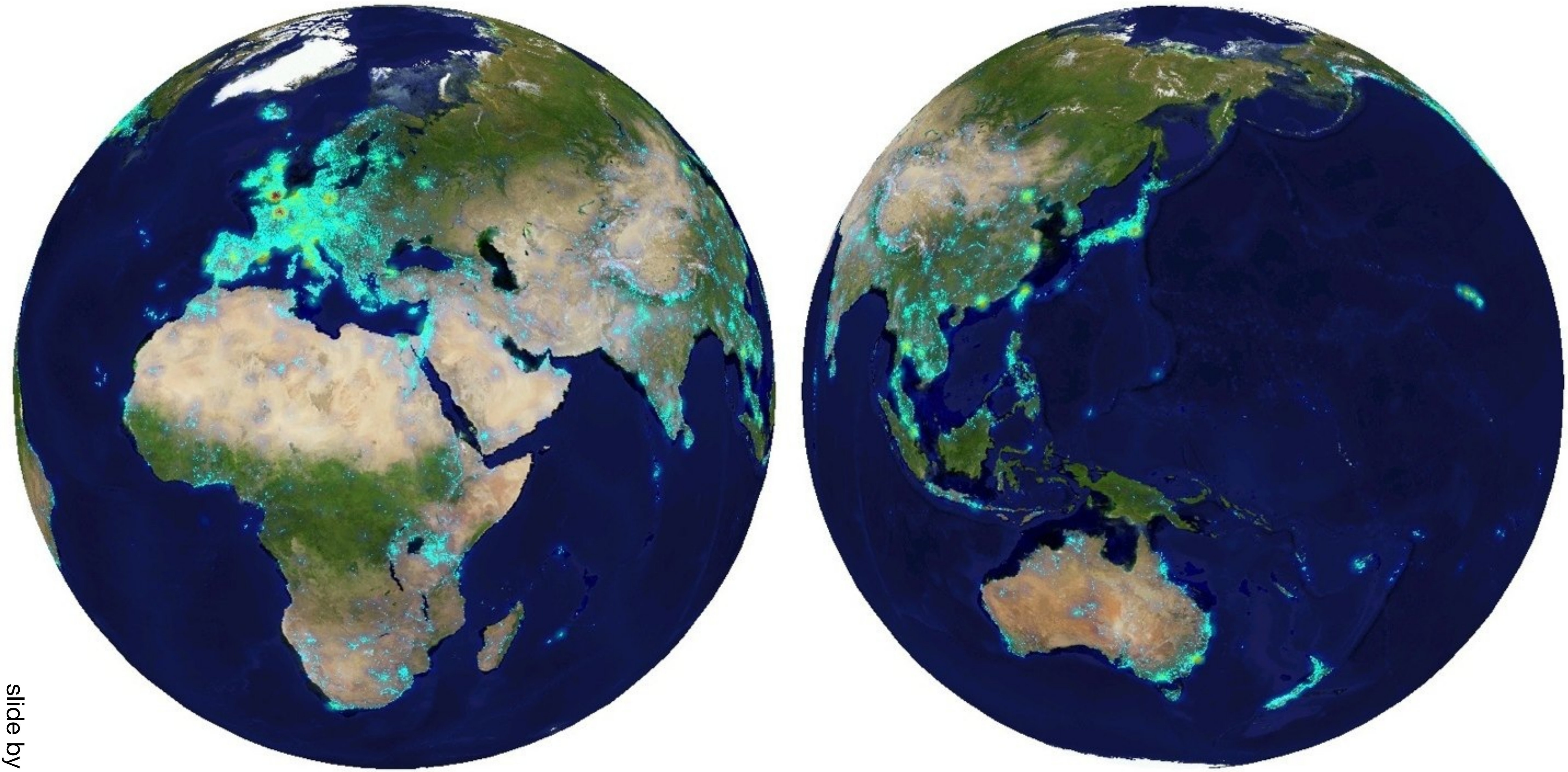
6+ million geotagged photos by 109,788 photographers



Annotated by Flickr users

slide by James Hays

6+ million geotagged photos by 109,788 photographers



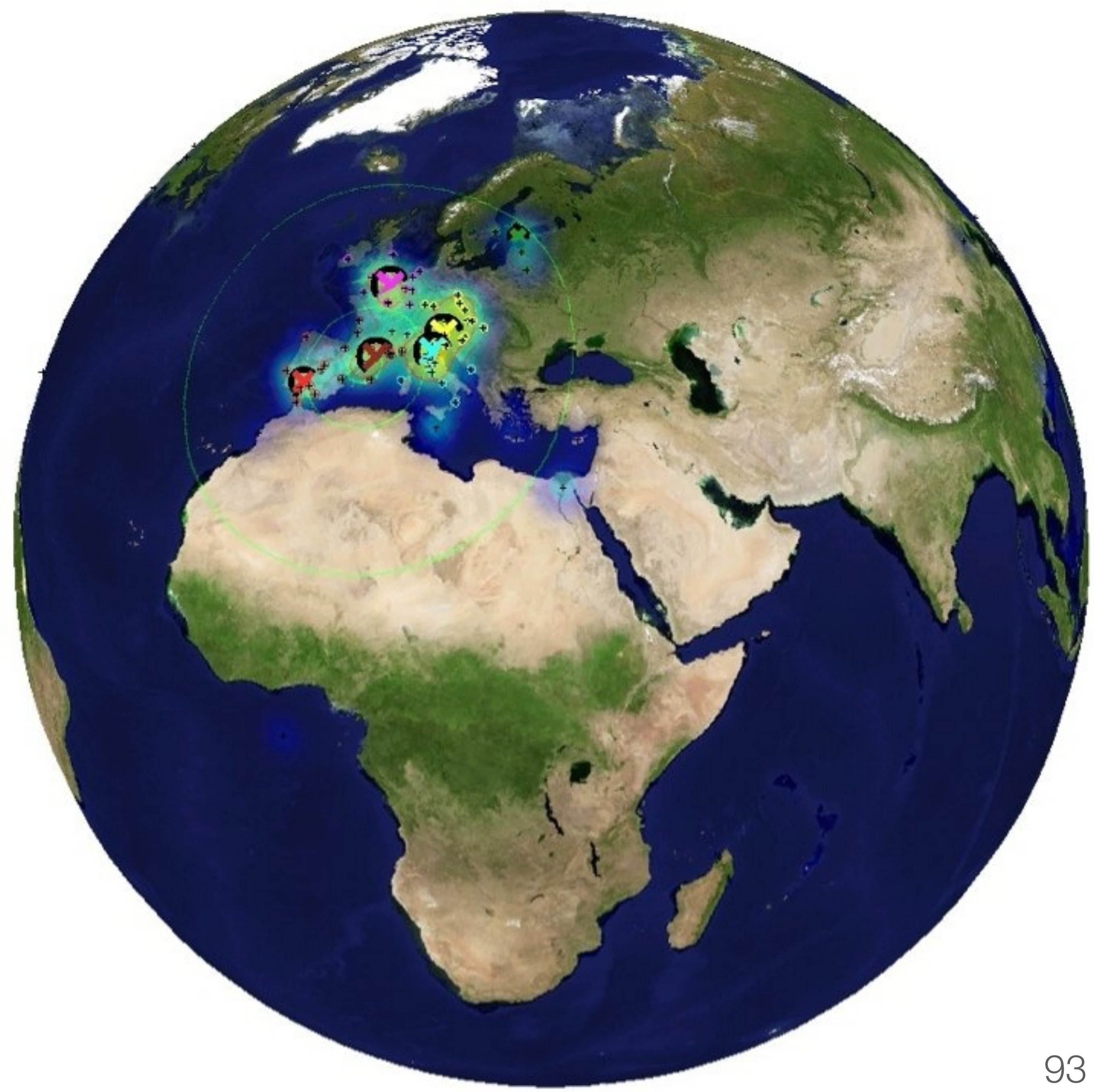
Annotated by Flickr users

slide by James Hays

slide by James Hays

Scene Matches

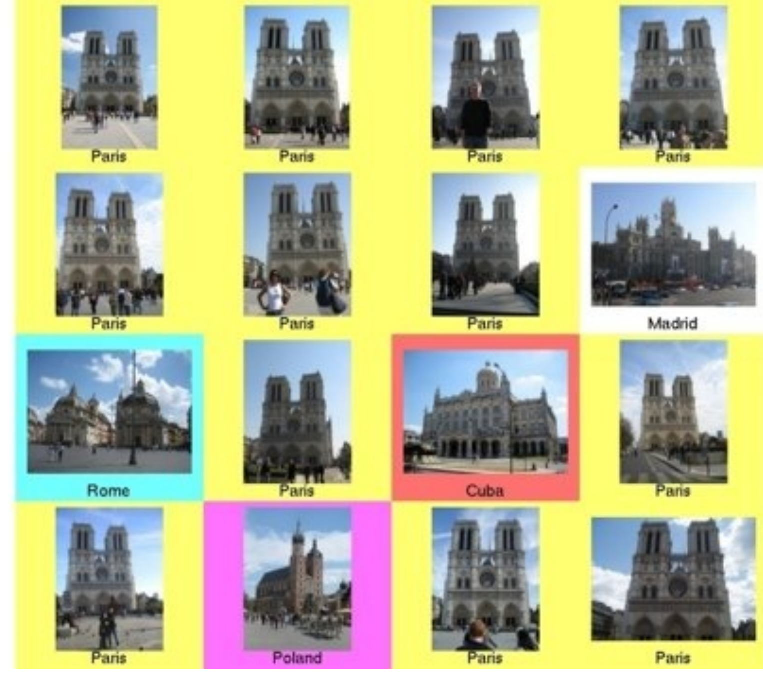
slide by James Hays

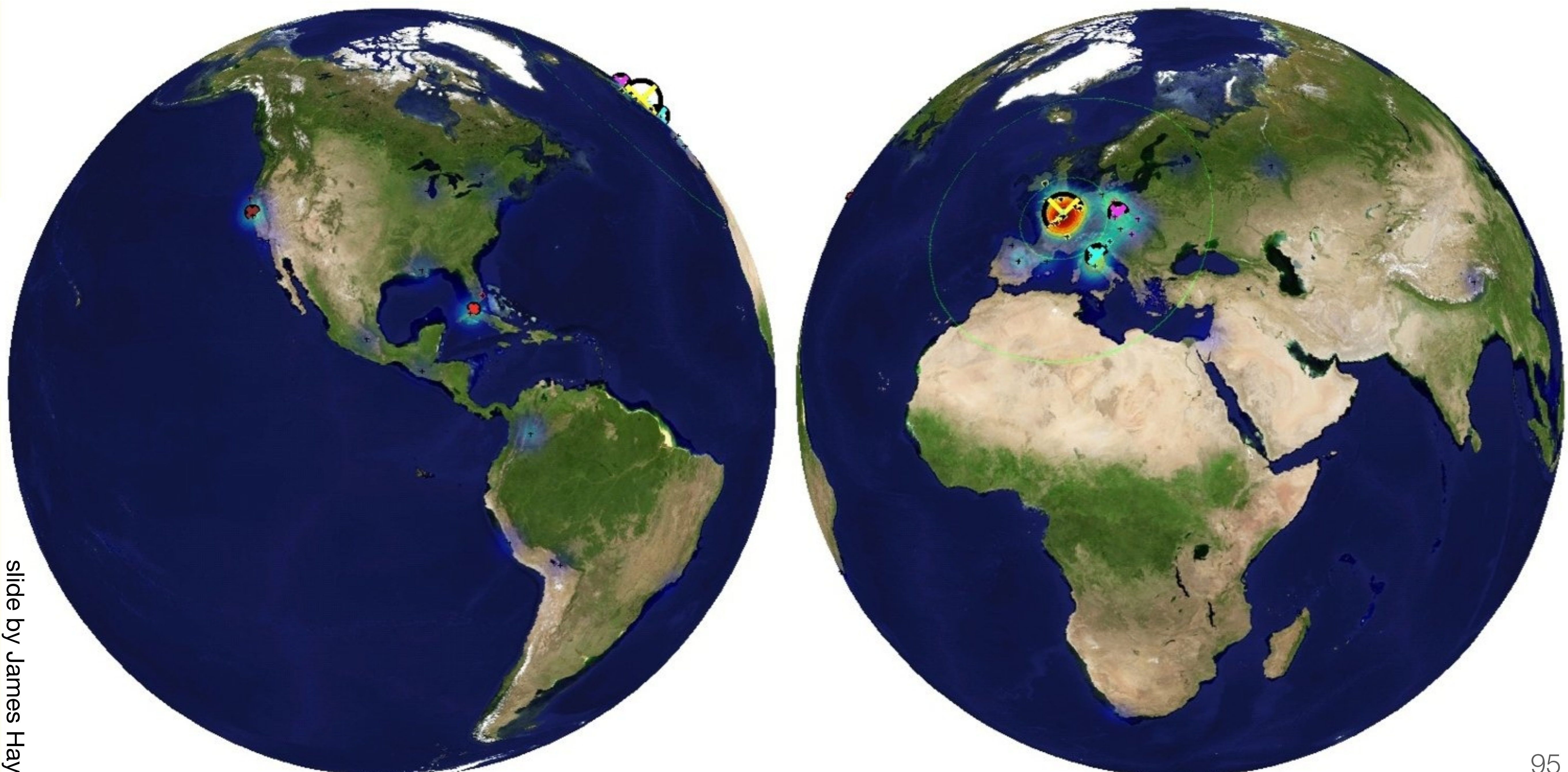


slide by James Hays

Scene Matches

slide by James Hays





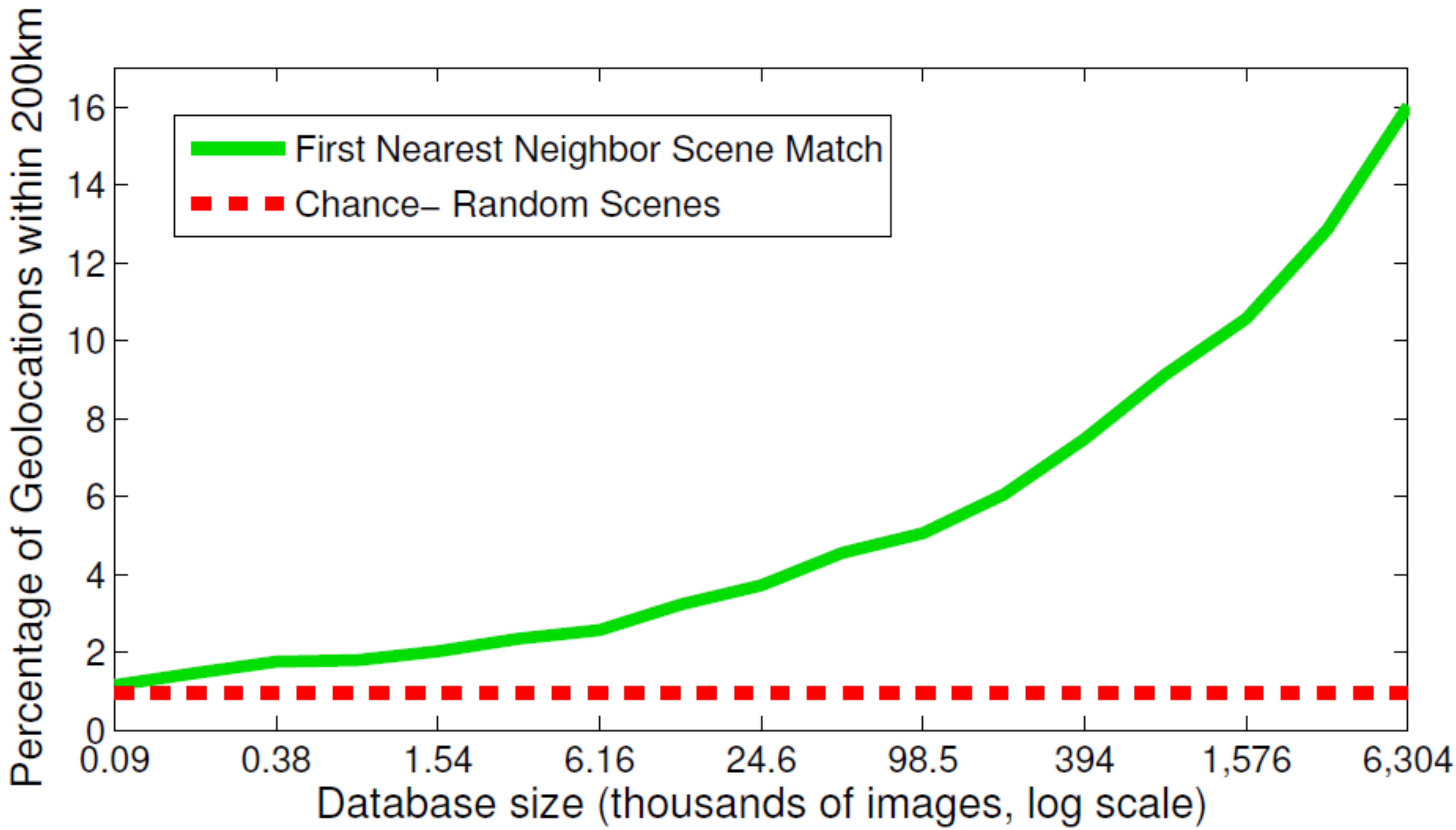
slide by James Hays

Scene Matches

slide by James Hays

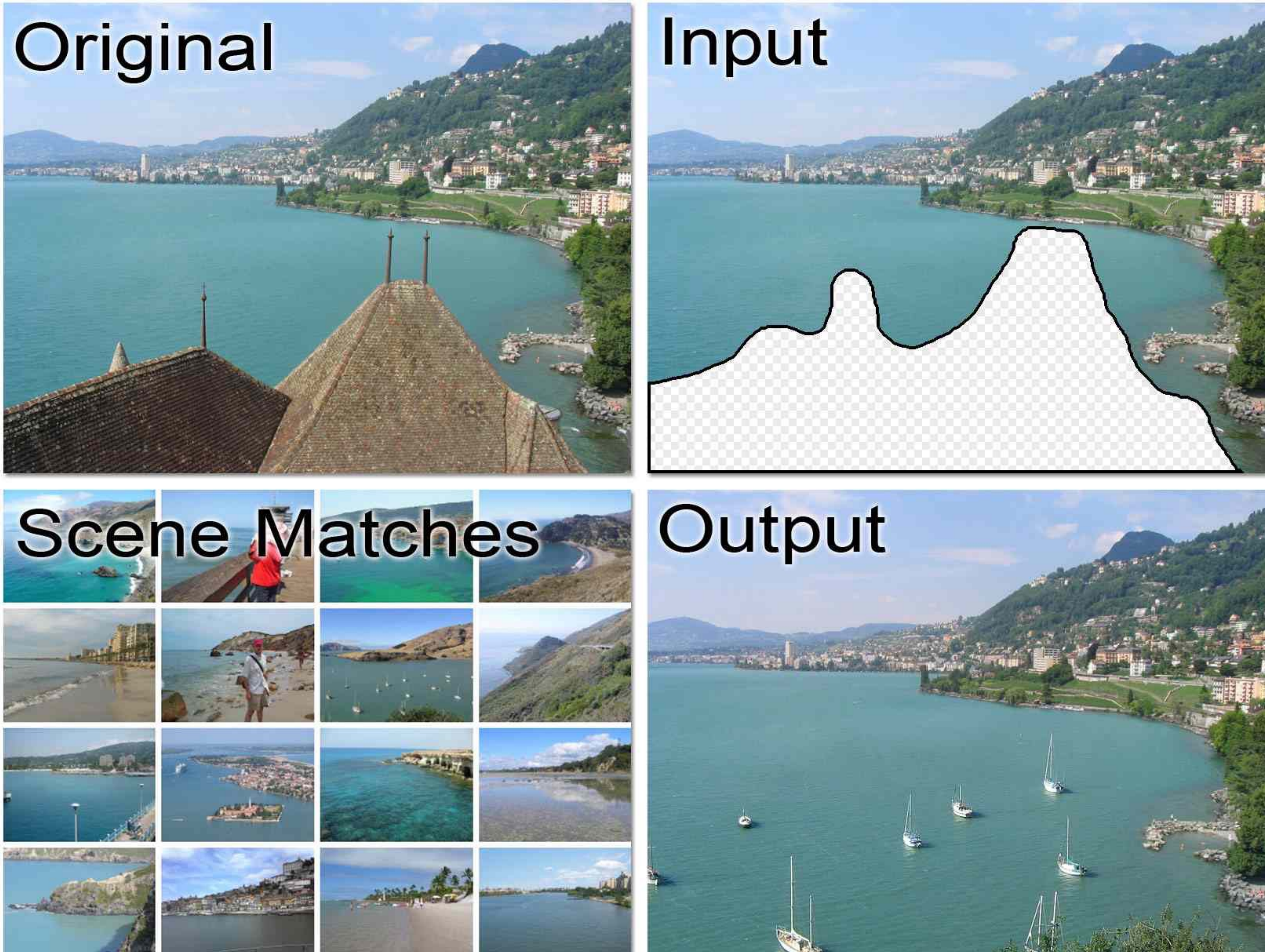
slide by James Hays

The Importance of Data

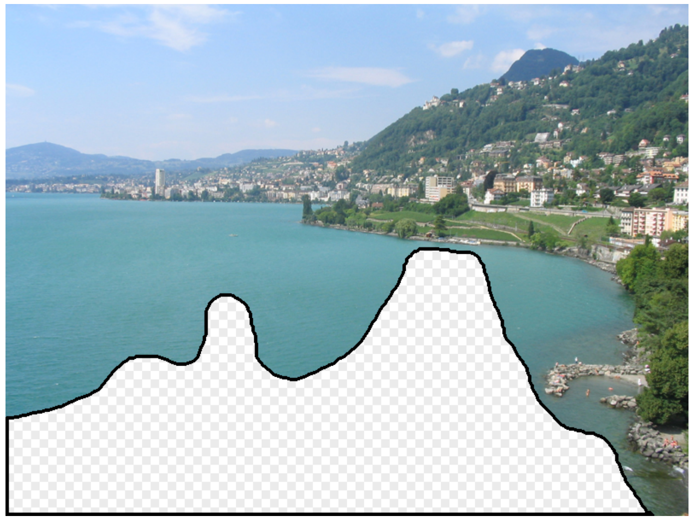


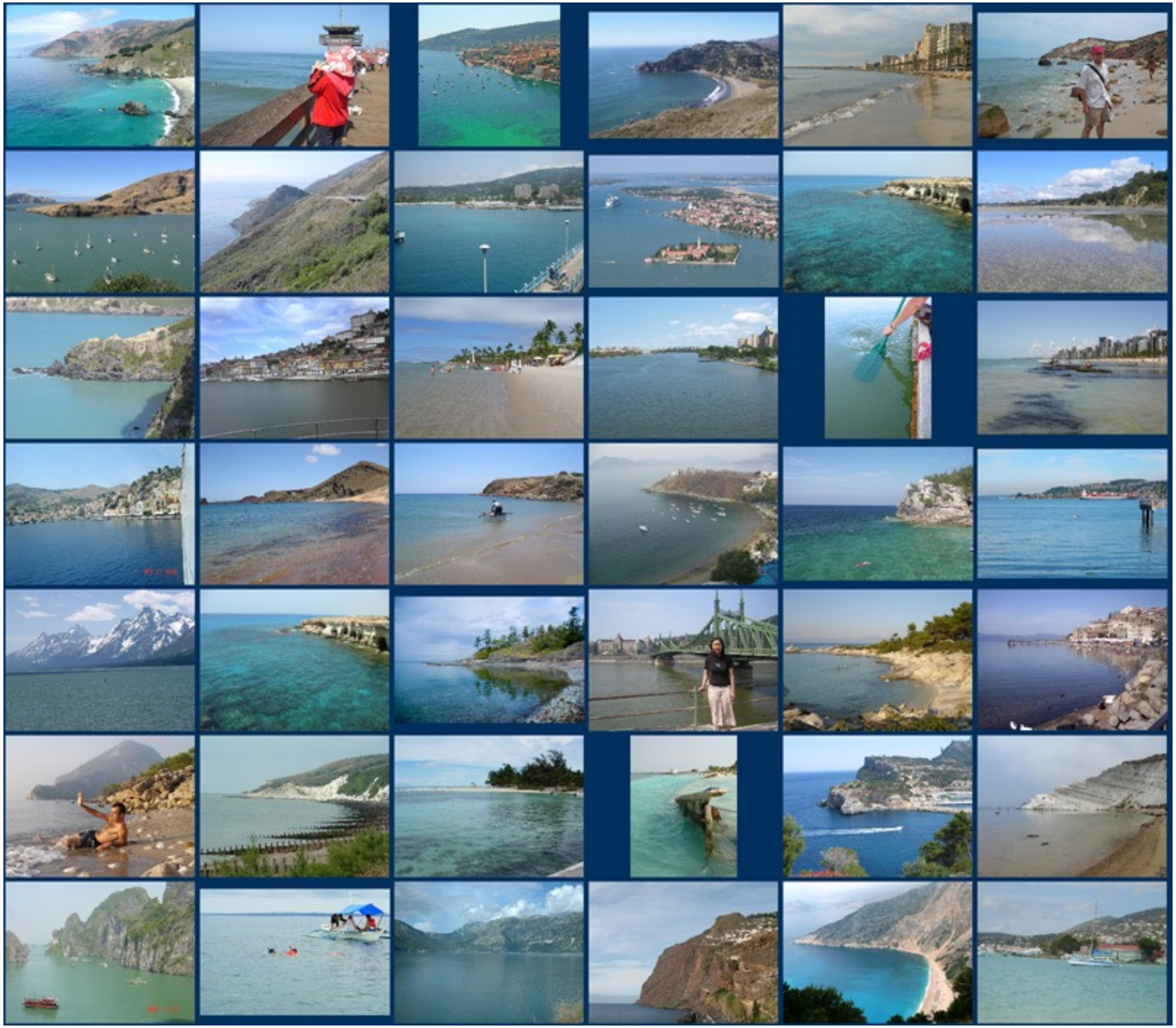
slide by James Hays

Scene Completion [Hays & Efros, SIGGRAPH07]



slide by James Hays



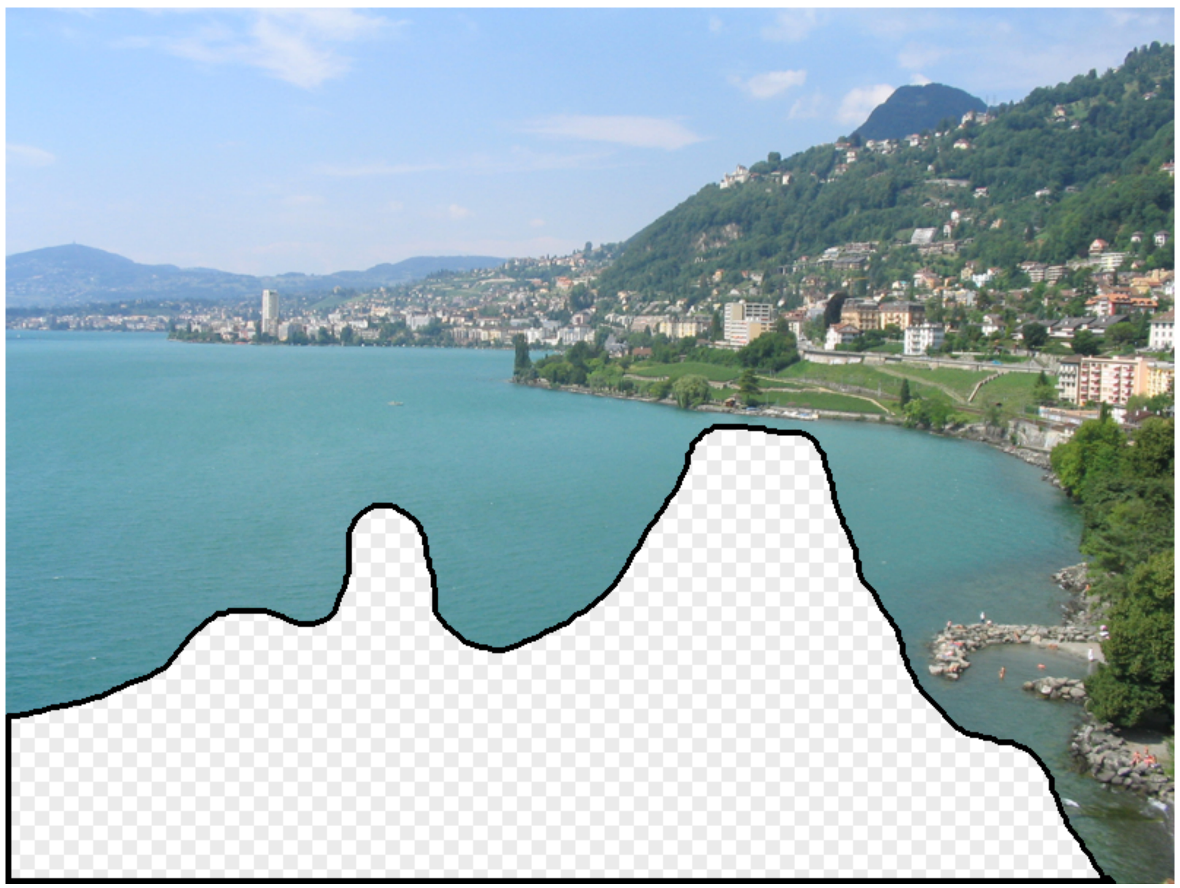


… 200 total

Hays and Efros, SIGGRAPH 2007

slide by James Hays

Context Matching



slide by James Hays

Hays and Efros, SIGGRAPH 2007


Graph cut + Poisson blending
slide by James Hays
Hays and Efros, SIGGRAPH 2007

slide by James Hays

Hays and Efros, SIGGRAPH 2007

slide by James Hays

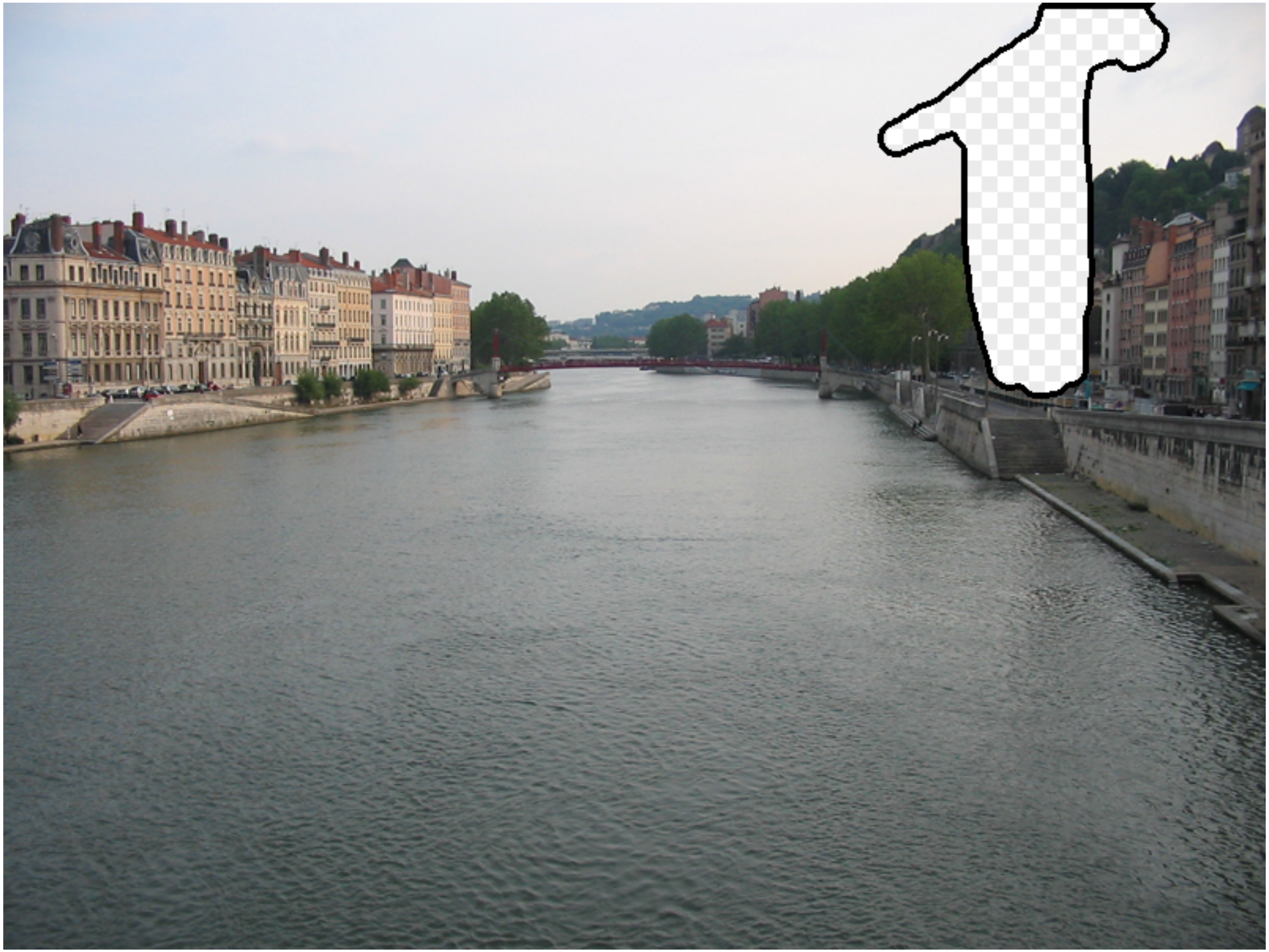
Hays and Efros, SIGGRAPH 2007

slide by James Hays

Hays and Efros, SIGGRAPH 2007

slide by James Hays

Hays and Efros, SIGGRAPH 2007



Hays and Efros, SIGGRAPH 2007

slide by James Hays

slide by James Hays

Hays and Efros, SIGGRAPH 2007

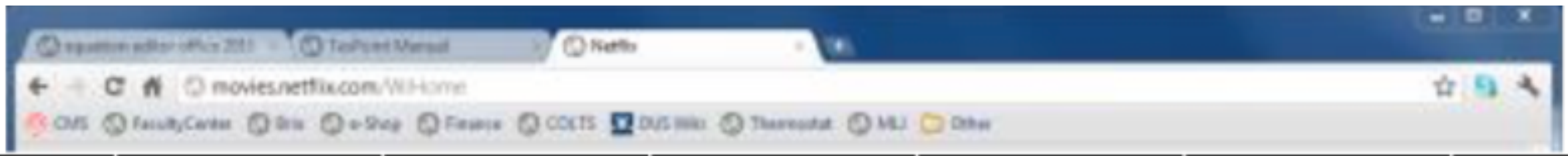
Weighted K-NN for Regression

- Given: Training data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$
  – Attribute vectors: $x_i \in X$
  – Target attribute $y_i \in \mathcal{R}$
- Parameter:
  – Similarity function: $K : X \times X \rightarrow \mathcal{R}$
  – Number of nearest neighbors to consider: k
- Prediction rule
  – New example x'
  – K-nearest neighbors: k train examples with largest $K(x_i,x')$

$$h(\vec{x}') = \frac{\sum_{i \in knn(\vec{x}')} y_i K(\vec{x}_i, \vec{x}')}{\sum_{i \in knn(\vec{x}')} K(\vec{x}_i, \vec{x}')}$$

slide by Thorsten Joachims

Collaborative Filtering



Rating Matrix	m_1	m_2	m_3	m_4	m_5	m_6
u_1		1	5		3	5
u_2		5	1	1	3	1
u_3		2	4		1	5
u	?	1	4	?	?	?

slide by Thorsten Joachims

Overview of Nearest Neighbors

- Very simple method
- Retain all training data
  - Can be slow in testing
  - Finding NN in high dimensions is slow
- Metrics are very important
- Good baseline

slide by Rob Fergus

Next Class:

Kernel Regression,
Distance Metrics,
Curse of Dimensionality