Lecture 2:
Machine Learning by Examples, Nearest Neighbor Classifier
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)
A classic example of a task that requires machine learning: It is very hard to say what makes a 2
Machine Learning
(by examples)
Pose Estimation
Collaborative Filtering

Don’t mix preferences on Netflix!

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

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
Reinforcement Learning

https://www.youtube.com/watch?v=LleRKHsJBJ0
Spam Filtering

Spam Filtering

ham

- Southwest Airlines
- DiscountMags.com
- support, Alex (3)
- American Airlines AAdvantage
- Taesup, Alex, Taesup (3)

Your trip is around the corner! - You're all set for your San Jose trip! My Account | View My Itinerary Online

$3.99 Business & Finance Sale.. starts now! - Trouble Seeing This Email? View as Webpage STOP these e-r

Your order has shipped... - please send to the address below for an exchange remotess.com(exchange)

AAAdvantage eSummary - January 2013 - VIEW IN WEB BROWSER >> http://americanairlines.ed10.net/rpJC

Happy new year! - Hi Alex, Thanks for your condolence. I will arrive at Berkeley on 16th (wed) night. So, I car

1:17 am

Jan 11

spam

- mae
- Dear Valued Customers,
- Steven Cooke
- paper18
- First-Class Mail Service
- garjeti
- Candy.Li
- Ronan Morgan
- RE/MAX®
- newsletter
- CJCR editor
- garjeti (2)
- Wayne Smith

(El&ISTP Index)2013机械与自动化工程国际会议征文: [alex.smola@gmail.com] - 尊敬的老师，您好：

Low Interest Rate Loan - Dear Valued Customers, Do you need a loan or funding for any of the following reasons:

Congratulations Alex, $150 awaits you - Alex: IMPORTANT - NOTICE OF WINNINGS Please make sure your


Tracking ID (G)BG035 849 603 4893 4550 - Fed Ex Order: JN-3339-28881768 Order Date: Thursday, 3 Janua

Call for Research Papers - GLOBAL ADVANCED RESEARCH JOURNAL OF ENGINEERING, TECHNOLOG

Call for Research Papers - GLOBAL ADVANCED RESEARCH JOURNAL OF ENGINEERING, TECHNOLOG

Call for Research Papers - GLOBAL ADVANCED RESEARCH JOURNAL OF ENGINEERING, TECHNOLOG

Jan 11

Jan 11

Jan 11

Jan 11

Jan 10

Jan 10

Jan 10

Jan 10

Jan 9

Jan 9

Jan 9

Jan 9

Jan 9
Cheque Reading

segment image

Photograph Front of Check
Place the check on a dark background in a well-lit area and keep the camera steady and align the check’s edges with the boundaries of the image.

Note: Fidelity cannot act on any written instructions

recognize handwriting
Image Layout

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

Why these ads?
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]
Topic Models of Text Documents

slide by Eric Sudderth
Genomics: group individuals by genetic similarity
Learning - revisited

prior knowledge → Learning → knowledge

data → Learning

slide by Stuart Russell

8/25/11
CS 194-10
Fall 2011, Stuart Russell
Learning - revisited

Learning

prior knowledge

data

knowledge

Crucial open problem: weak intermediate forms of knowledge that support future generalizations.
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)
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?
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).
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.
Supervised Learning
Supervised Learning

- **Binary classification**
  Given $x$ find $y$ in $\{-1, 1\}$

- **Multicategory classification**
  Given $x$ find $y$ in $\{1, \ldots, k\}$

- **Regression**
  Given $x$ find $y$ in $\mathbb{R}$ (or $\mathbb{R}^d$)

- **Sequence annotation**
  Given sequence $x_1 \ldots x_l$ find $y_1 \ldots y_l$

- **Hierarchical Categorization (Ontology)**
  Given $x$ find a point in the hierarchy of $y$ (e.g. a tree)

- **Prediction**
  Given $x_t$ and $y_{t-1} \ldots y_1$ find $y_t$

often with loss

$$l(y, f(x))$$
Binary Classification
Multiclass Classification + Annotation

- Bird and frog
- Person, dog, and chair
- Person, hammer, flower pot, power drill
- Person, car, helmet, motorcycle
Regression

\[ y = 0.98x - 0.01 \]

\[ r^2 = 0.496 \]

Midoffspring beak depth (mm)

Biological midparent beak depth (mm)

Copyright © 2004 Pearson Prentice Hall, Inc.

linear

nonlinear

SPAN | 1142
K | 0.2153
PLATEAU | -67.44

Minutes
Sequence Annotation

given sequence

gene finding

speech recognition

activity segmentation

named entities

LRR Receptor-like Kinase
TIR-NBS-LRR Disease resistance
Retrotransposon associated
Other
STS
Ontology

webpages

Arts
Movies, Television, Music...

Games
Video Games, RPGs, Gambling...

Kids and Teens
Arts, School Time, Teen Life...

Reference
Maps, Education, Libraries...

Shopping
Clothing, Food, Gifts...

World
Català, Dansk, Deutsch, Español, Français, Italiano, 日本語, Nederlands, Polski, Русский, Svenska...

Business
Jobs, Real Estate, Investing...

Health
Fitness, Medicine, Alternative...

News
Media, Newspapers, Weather...

Regional
US, Canada, UK, Europe...

Society
People, Religion, Issues...

Sports
Baseball, Soccer, Basketball...

Computers
Internet, Software, Hardware...

Home
Family, Consumers, Cooking...

Recreation
Travel, Food, Outdoors, Humor...

Science
Biology, Psychology, Physics...

Become an Editor | Help build the largest human-edited directory of the web

Copyright © 2013 Netscape

5,114,083 sites - 96,877 editors - over 1,014,849 categories
Prediction

tomorrow’s stock price
Unsupervised Learning
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
Clustering

- Documents
- Users
- Webpages
- Diseases
- Pictures
- Vehicles
...
Principal Components

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

The text on the slide talks about hierarchical grouping in the context of document topics. It mentions that each document lives at a node and is described by a distribution over topics. In LDA, each document has a unique topic to be a distribution over words and each document to be considered a topic. The slide shows a tree structure learned from the 50,000 CIFAR-10 images, with subtrees and nodes that have at least 50 images. Detailed views of portions under the node's distribution are shown, with the ten images at each node being those with the highest probability.
Independent Components

find them automatically
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*
Concept Learning Example

- **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.

<table>
<thead>
<tr>
<th></th>
<th><strong>correct</strong> (complete, partial, guessing)</th>
<th><strong>color</strong> (yes, no)</th>
<th><strong>original</strong> (yes, no)</th>
<th><strong>presentation</strong> (clear, unclear, cryptic)</th>
<th><strong>binder</strong> (yes, no)</th>
<th><strong>A+</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>complete</td>
<td>yes</td>
<td>yes</td>
<td>clear</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>complete</td>
<td>no</td>
<td>yes</td>
<td>clear</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>partial</td>
<td>yes</td>
<td>no</td>
<td>unclear</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>complete</td>
<td>yes</td>
<td>yes</td>
<td>clear</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
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 $\rightarrow$ 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.
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 $\rightarrow$ 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.
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
Image Classification: a core task in Computer Vision

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

→ cat
The problem: semantic gap

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

E.g.

$300 \times 100 \times 3$

(3 for 3 color channels RGB)
Challenges: Viewpoint Variation
Challenges: Illumination
Challenges: Deformation
Challenges: Occlusion
Challenges: Background clutter
Challenges: Intraclass variation
An image classifier

```python
def predict(image):
    # ????
    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.
Attempts have been made
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
```
First classifier: **Nearest Neighbor Classifier**

Remember all training images and their labels

Predict the label of the most similar training image

```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```

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

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

<table>
<thead>
<tr>
<th>test image</th>
<th>training image</th>
<th>pixel-wise absolute value differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>56 32 10 18</td>
<td>10 20 24 17</td>
<td>46 12 14 1</td>
</tr>
<tr>
<td>90 23 128 133</td>
<td>8 10 89 100</td>
<td>82 13 39 33</td>
</tr>
<tr>
<td>24 26 178 200</td>
<td>12 16 178 170</td>
<td>12 10 0 30</td>
</tr>
<tr>
<td>2 0 255 220</td>
<td>4 32 233 112</td>
<td>2 32 22 108</td>
</tr>
</tbody>
</table>
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-dimensional 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
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
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-dimensional 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 range(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 nearest train image with L1 distance
- predict the label of nearest training image

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 """
        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]
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        for i in xrange(num_test):
            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

Q: how does the classification speed depend on the size of the training data?
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-dimensional 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]
        # let's 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

Q: how does the classification speed depend on the size of the training data? linearly :(
Aside: Approximate Nearest Neighbor
find approximate nearest neighbors quickly

ANN: A Library for Approximate Nearest Neighbor Searching
David M. Mount and Suneil 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 personal 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

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 through MacPorts (thanks to Mark Moll for maintaining the Porttle)
- 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:
- Matthias 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]
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**

\[ d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2} \]
k-Nearest Neighbor

find the k nearest images, have them vote on the label

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: \( K : X \times X \rightarrow \mathbb{R} \)
  - Number of nearest neighbors to consider: \( k \)

- Prediction rule
  - New example \( x' \)
  - K-nearest neighbors: \( k \) train examples with largest \( K(x_i, x') \)

\[
h(x') = \arg \max_{y \in Y} \left\{ \sum_{i \in \text{knn}(x')} 1[y_i = y] \right\}
\]
1-Nearest Neighbor
4-Nearest Neighbors
4-Nearest Neighbors Sign
For binary classification problems, why is it a good idea to use an odd number of $K$?
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.
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 $2p(1-p)$ for noisy problems

- k-Nearest Neighbor
  - Converges to perfect solution if clear separation (but needs more data)
  - Converges to minimal error $\min(p, 1-p)$ for noisy problems if $k$ increases
Demo
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: \(K : X \times X \rightarrow \mathbb{R}\)
  – Number of nearest neighbors to consider: \(k\)

• Prediction rule
  – New example \(x'\)
  – K-nearest neighbors: \(k\) train examples with largest \(K(x_i, x')\)

\[
h(x') = \arg \max_{y \in Y} \left\{ \sum_{i \in \text{knn}(x')} 1[y_i = y] K(x'_i, x') \right\}
\]
More Nearest Neighbors in Visual Data
A nearest neighbor recognition example
Where in the World? [Hays & Efros, CVPR 2008]
Where in the World? [Hays & Efros, CVPR 2008]
6+ million geotagged photos by 109,788 photographers

Annotated by Flickr users
6+ million geotagged photos by 109,788 photographers

Annotated by Flickr users
Scene Matches

Madrid

england

France

Paris

Croatia

heidelberg

Macau

Malta

Cairo

Italy

Italy

Italy

Latvia

europe

Barcelona

Austria
Scene Matches
The Importance of Data

![Graph showing the relationship between database size and percentage of geolocations within 200km. The graph compares the first nearest neighbor scene match with chance-random scenes.](image-url)
Scene Completion  [Hays & Efros, SIGGRAPH07]

Original

Input

Scene Matches

Output
Context Matching
Graph cut + Poisson blending
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 \( y_i \in \mathbb{R} \)

• Parameter:
  – Similarity function: \( K: X \times X \to \mathbb{R} \)
  – Number of nearest neighbors to consider: \( k \)

• Prediction rule
  – New example \( x' \)
  – K-nearest neighbors: \( k \) train examples with largest \( K(x_i,x') \)

\[
h(x') = \frac{\sum_{i \in \text{knn}(x')} y_i K(x_i, x')}{\sum_{i \in \text{knn}(x')} K(x_i, x')}
\]
Collaborative Filtering

<table>
<thead>
<tr>
<th>Rating Matrix</th>
<th>m₁</th>
<th>m₂</th>
<th>m₃</th>
<th>m₄</th>
<th>m₅</th>
<th>m₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u₂</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>u₃</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>?</td>
<td>1</td>
<td>4</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Recently Watched
1. MILLIONAIRE BOYS
2. THE LAST ENEMY
3. GEORGE CLOONEY
4. MI-5

Top 10 for Thorsten
1. LOVE THE BEAST
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
Next Class:
Linear Regression and Least Squares