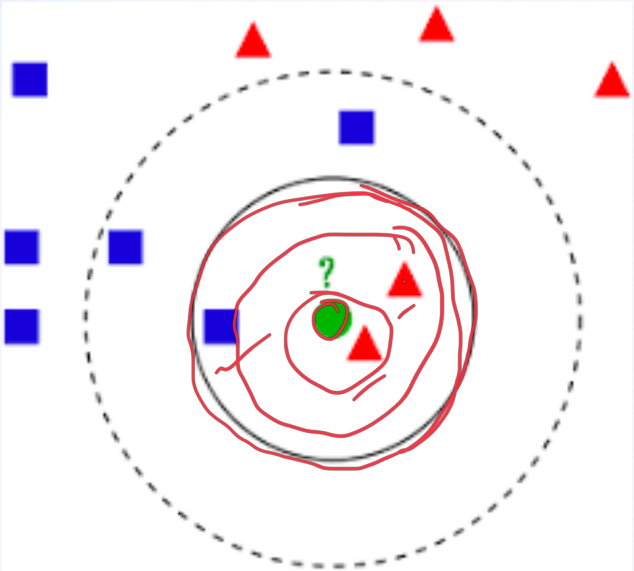
# Applied Machine Learning

## Lecture 9
## K-Nearest Neighbors (K-NN)

Ekarat Rattagan, Ph.D.

# K-Nearest Neighbor Algorithm

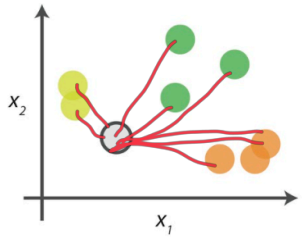
- Assumption: Similar Inputs have similar outputs
- To classify a new input vector x, examine the k-closet training data points to x and assign the object to the most frequently occurring class




Supervised learning
- Classification

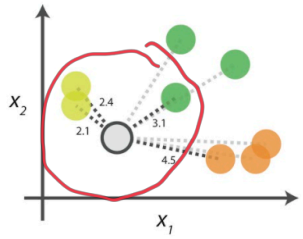
# kNN Algorithm

## 0. Look at the data




Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

## 1. Calculate distances

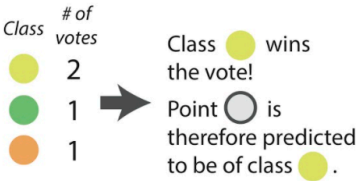



Start by calculating the distances between the grey point and all other points.

## 2. Find neighbours


Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

## 3. Vote on labels




Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

Diagram source: Cambridge Coding

## 1 Euclidian Distance

$$d(i,j) = \sqrt{\sum_{k=1}^{n}(x_{i,k} - x_{j,k})^2}$$

## 2 Manhattan Distance

$$d(i,j) = \sum_{k=1}^{n}|x_{i,k} - x_{j,k}|$$

## 3 Minkowski Distance

$$d(i,j) = \left(\sum_{k=1}^{n}|x_{i,k} - x_{j,k}|^3\right)^{1/3}$$

1, 2, 3, ...

q=1

q=2

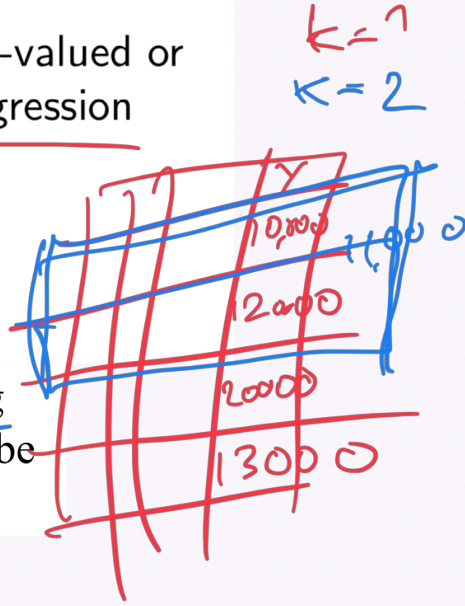
q=3

# Instance-based Learning

- Alternative to parametric models are non-parametric models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)
- Learning amounts to simply storing training data
- Test instances classified using similar training instances

- Instance-based methods are sometimes referred to as **lazy** learning methods because they delay processing until a new instance must be classified.

eager learning methods



# KNN Example

| | Food (3) | Chat (2) | Fast (2) | Price (3) | Bar (2) | BigTip |
|---|---|---|---|---|---|---|
| 1 | great | yes | yes | normal | no | yes |
| 2 | great | no | yes | normal | no | yes |
| 3 | mediocre | yes | no | high | no | no |
| 4 | great | yes | yes | normal | yes | yes |

Similarity metric: Number of matching attributes (k=2) Hamming distance

New examples:

– Example 1 (great, no, no, normal, no) Yes

→ most similar: number 2 (1 mismatch, 4 match) → yes

→Second most similar example: number 1 (2 mismatch, 3 match) → yes

# KNN Example

| | Food (3) | Chat (2) | Fast (2) | Price (3) | Bar (2) | BigTip |
|---|---|---|---|---|---|---|
| 1 | great | yes | yes | normal | no | yes |
| 2 | great | no | yes | normal | no | yes |
| 3 | mediocre | yes | no | high | no | no |
| 4 | great | yes | yes | normal | yes | yes |

Similarity metric: Number of matching attributes (k=2)

New examples:

- Example 1 (great, no, no, normal, no)

  → most similar: number 2 (1 mismatch, 4 match) → yes

  →Second most similar example: number 1 (2 mismatch, 3 match) → yes

- Example 2 (mediocre, yes, no, normal, no)

  → Most similar: number 3 (1 mismatch, 4 match) → no

  →Second most similar example: number 1 (2 mismatch, 3 match) → yes

# Ties

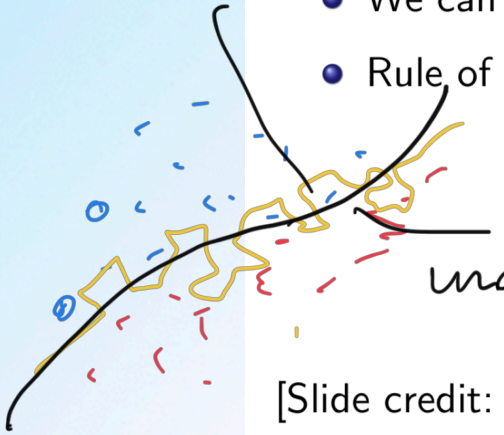
- ties may occur in a classification problem when $K > 1$
  - for binary classification: choose $K$ odd to avoid ties
  - for multi-class classification:
    - decrease the value of $K$ until the tie is broken
    - if that doesn't work, use the class given by a 1NN classifier

Credit: Mengye Ren

# K-Nearest Neighbors

How do we choose $k$?

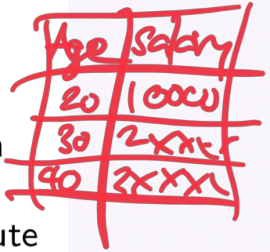
- Larger $k$ may lead to better performance
- But if we set $k$ too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use cross-validation to find $k$
- Rule of thumb is $k < sqrt(n)$, where $n$ is the number of training examples




[Slide credit: O. Veksler]

# Issues and Remedies (1/2)

- If some attributes (coordinates of **x**) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)




- Irrelevant, correlated attributes add noise to distance measure
  - eliminate some attributes

Credit: Zemel, Urtasun, Fidler (UofT)

# Issues and Remedies (2/2)

- **Expensive at test time:** To find one nearest neighbor of a query point $\mathbf{x}$, we must compute the distance to all N training examples. Complexity: $O(kdN)$ for kNN
  - Use subset of dimensions
  - Pre-sort training examples into fast data structures (e.g., kd-trees)
- **Storage Requirements:** Must store all training data
  - Remove redundant data (e.g., condensing)
- **High Dimensional Data:** "Curse of Dimensionality"

"As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially." Charles Isbell

[Slide credit: David Claus]

# Example: Digit Classification

- Decent performance when lots of data

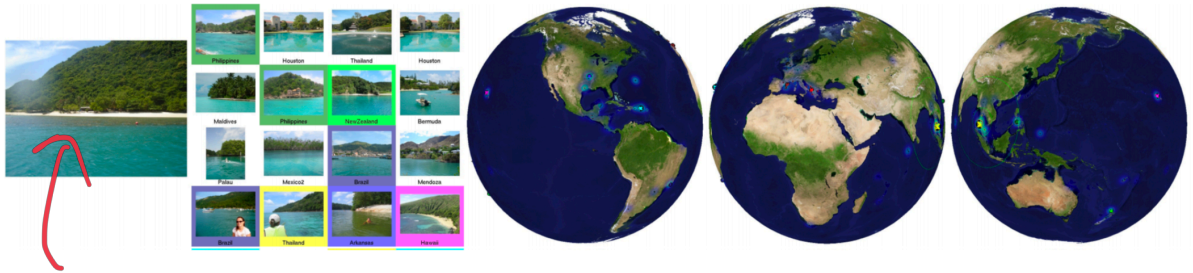
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

| | Test Error Rate (%) |
|---|---|
| Linear classifier (1-layer NN) | 12.0 |
| K-nearest-neighbors, Euclidean | 5.0 |
| K-nearest-neighbors, Euclidean, deskewed | 2.4 |
| K-NN, Tangent Distance, 16x16 | 1.1 |
| K-NN, shape context matching | 0.67 |
| 1000 RBF + linear classifier | 3.6 |
| SVM deg 4 polynomial | 1.1 |
| 2-layer NN, 300 hidden units | 4.7 |
| 2-layer NN, 300 HU, [deskewing] | 1.6 |
| LeNet-5, [distortions] | 0.8 |
| Boosted LeNet-4, [distortions] | 0.7 |

https://www.kaggle.com/cdeotte/mnist-perfect-100-using-knn

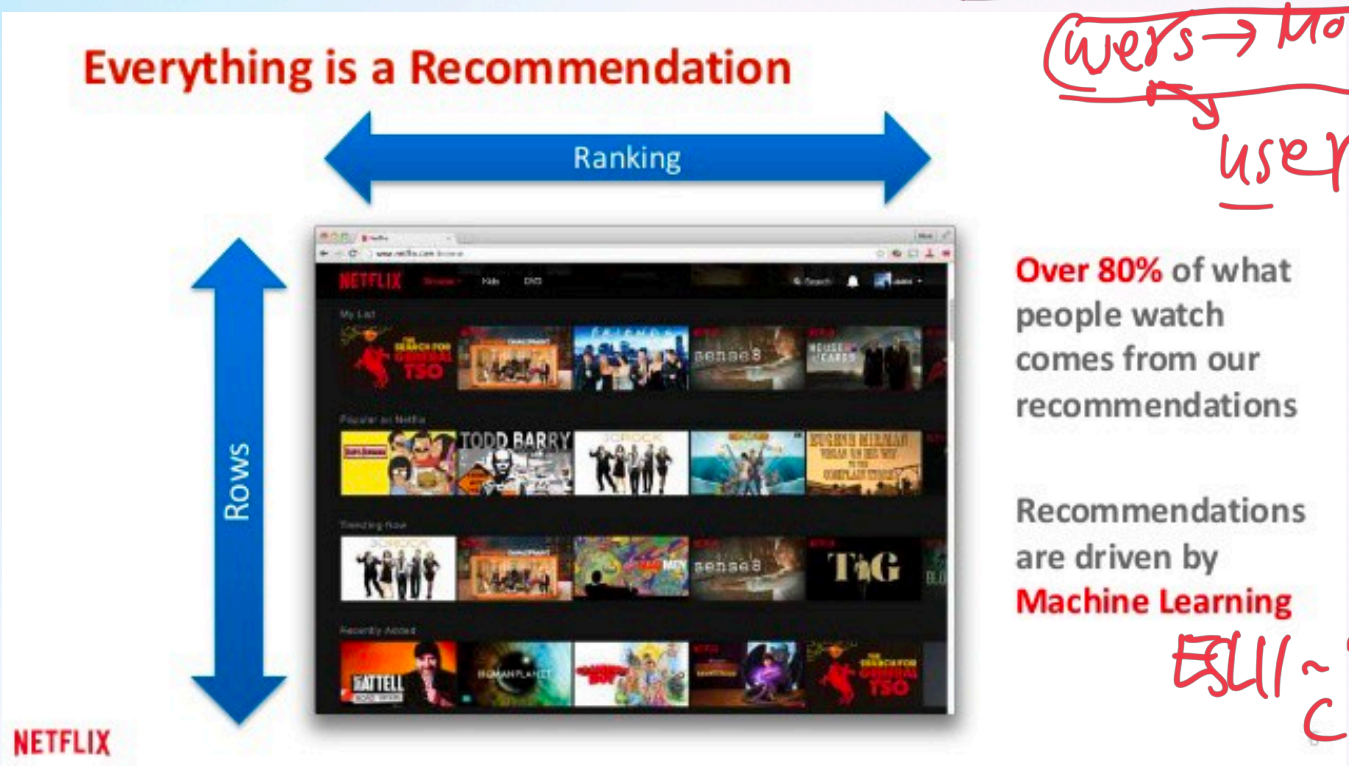
# Example: Where on Earth is this Photo from?

- Problem: Where (eg, which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with gps info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN (large $k$ better, they use $k = 120$)!



[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

# Example: Use of KNN for the Netflix Prize




https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea

http://cs229.stanford.edu/proj2006/HongTsamis-KNNForNetflix.pdf