

11.01.2019

# Statistical Methods in AI (CSE/ECE 471)



## Lecture-5: K-nearest neighbors classifier

Ravi Kiran

Center for Visual Information Technology (CVIT), IIIT Hyderabad




# Announcements

- A1 is due **20/1, 11.59 PM**
- **No tutorial this Saturday**



Felicity  
2019

## SUPERHERO FACTS

Wally West (  econd Flash ) stated he usually stays awake all night and takes micro sleeps during the day when people are blinking. Wish we could do that here at IIIT during lectures. :")

third

# Supervised Learning

```
graph TD; A[Supervised Learning] --> B[Classification]; A --> C[Regression]; A --> D[Reinforcement Learning];
```

Classification

Regression

Reinforcement  
Learning

# So far

- Decision Tree classifier

# Motivation

**Show me your  
friends and I  
will tell you  
who you are**

*labeled  
examples*

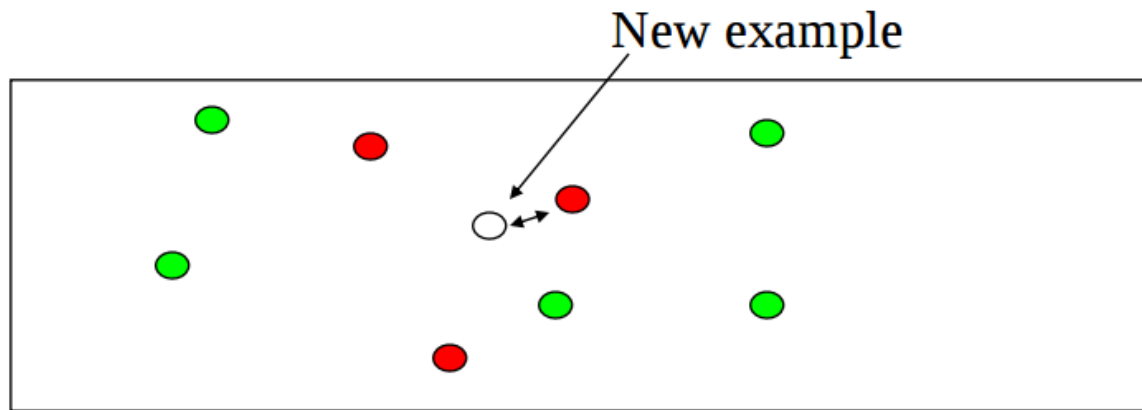
~~X~~

~~X~~

*→ label*

# Nearest neighbor classifier

- Given a new example  $\mathbf{x}$ , find the its closest training example  $\langle \mathbf{x}^i, y^i \rangle$  and predict  $y^i$



- How to measure distance – Euclidean (squared):

$$\|\mathbf{x} - \mathbf{x}^i\|^2 = \sum_j (x_j - x_j^i)^2$$

# 1-NN

## Algorithm:

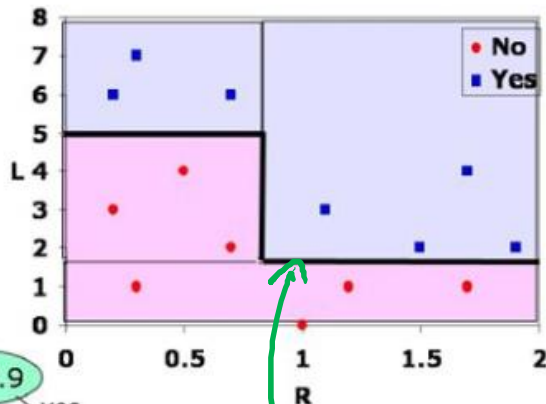
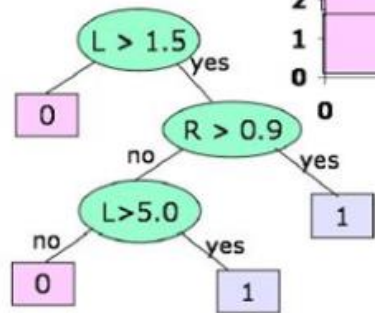
1. Find example  $(\mathbf{x}^*, t^*)$  (from the stored training set) closest to the test instance  $\mathbf{x}$ . That is:

$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\operatorname{argmin}} \quad \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

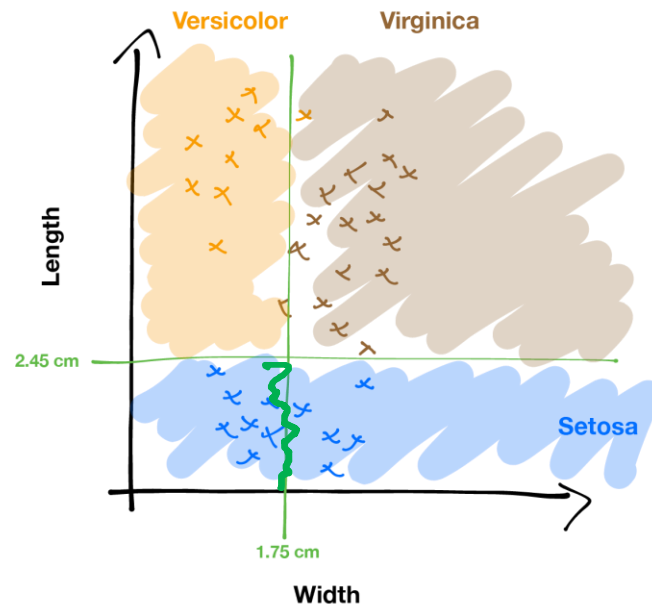
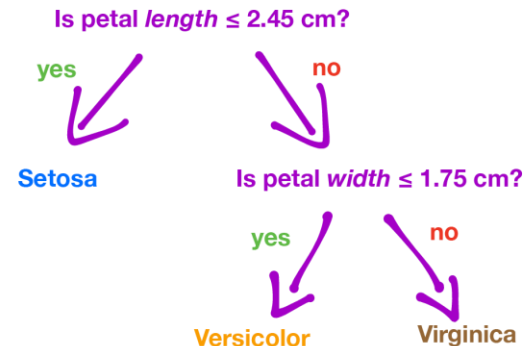
2. Output  $y = t^*$



# Recap: Decision Boundaries

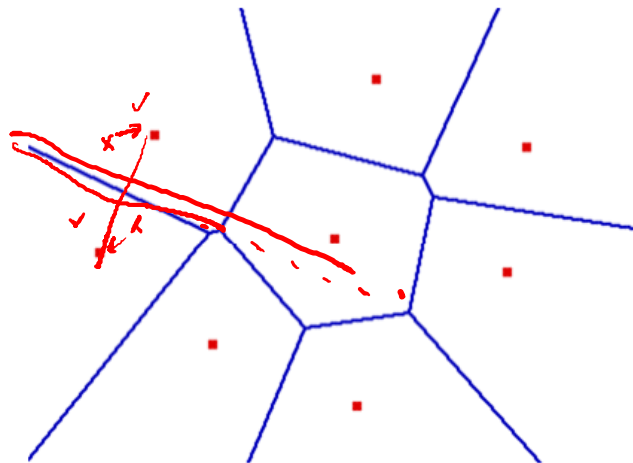


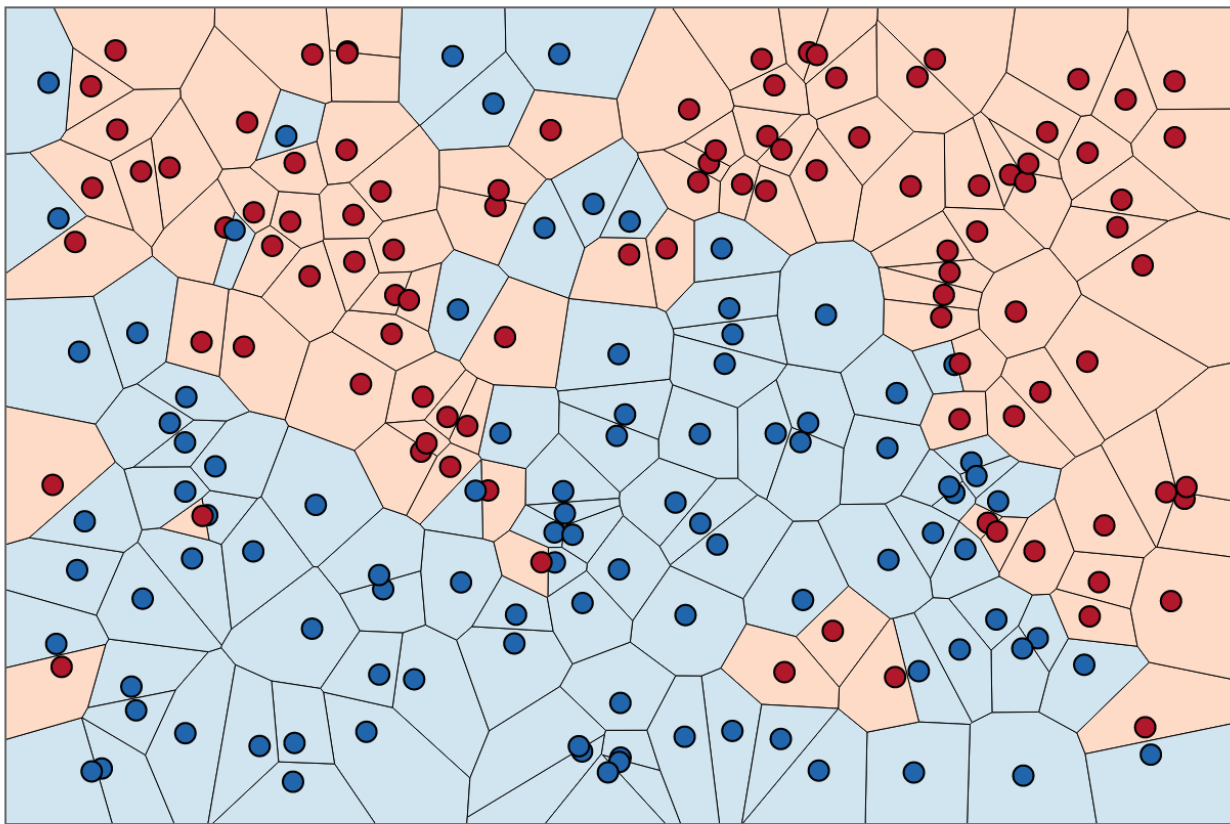
decision boundary

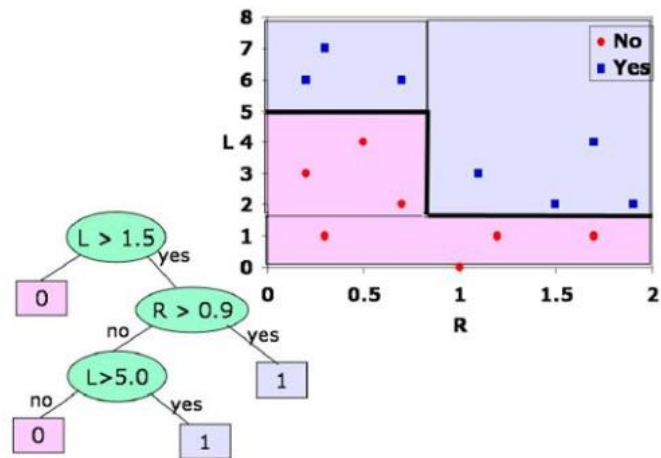
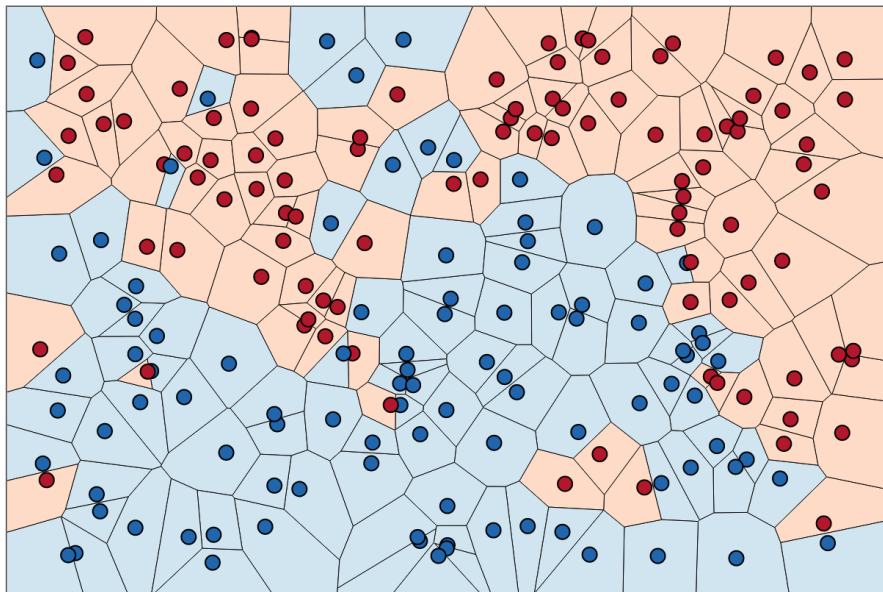
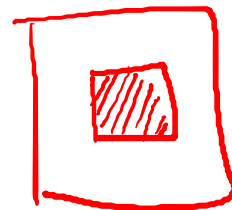


## Decision Boundaries: The Voronoi Diagram

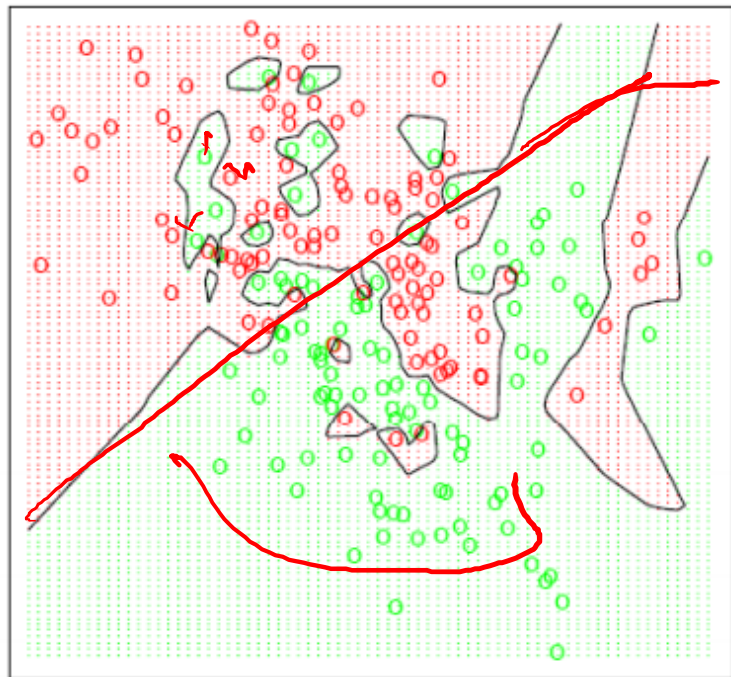
- Given a set of points, a **Voronoi diagram** describes the areas that are nearest to any given point.
- These areas can be viewed as zones of control.







# Decision Boundaries

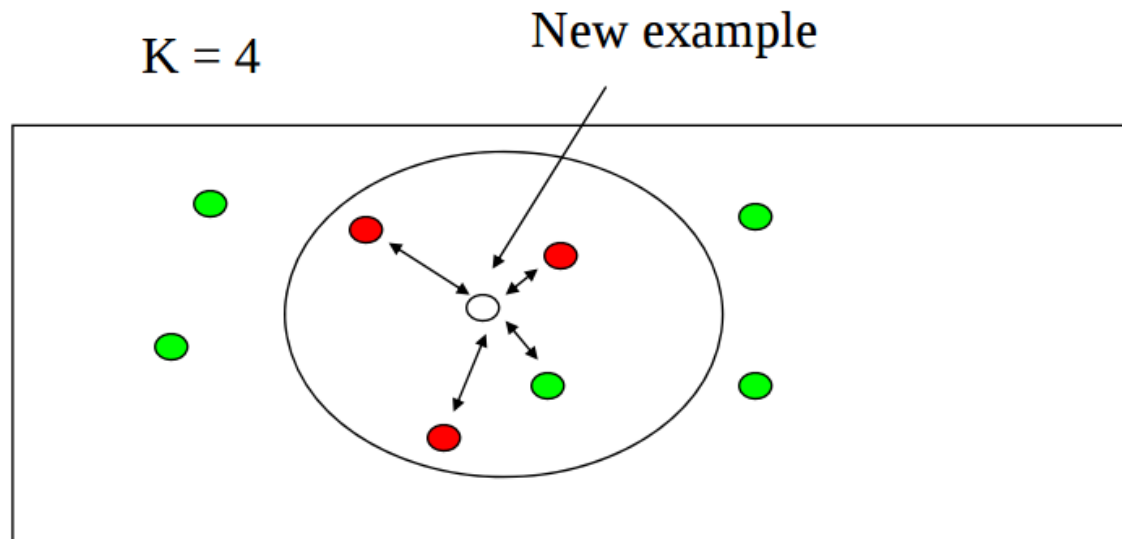


With large number of examples and possible noise in the labels, the decision boundary can become nasty!

We end up overfitting the data

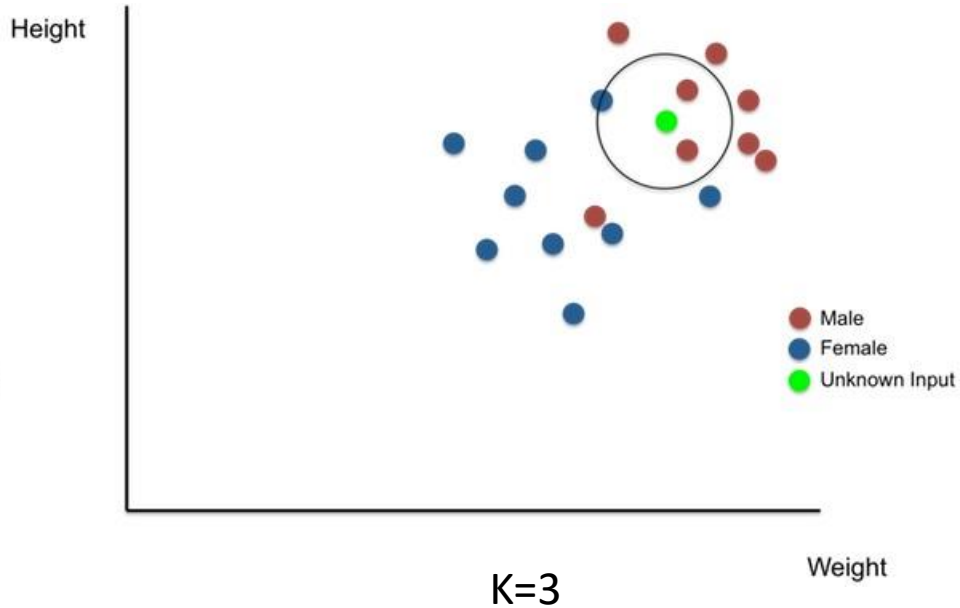
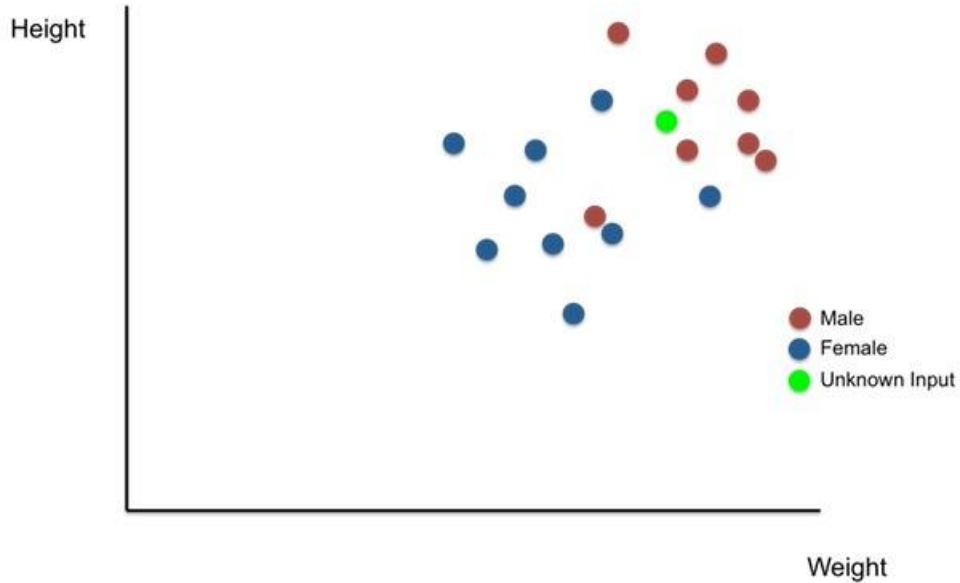
# K-Nearest Neighbor

Example:

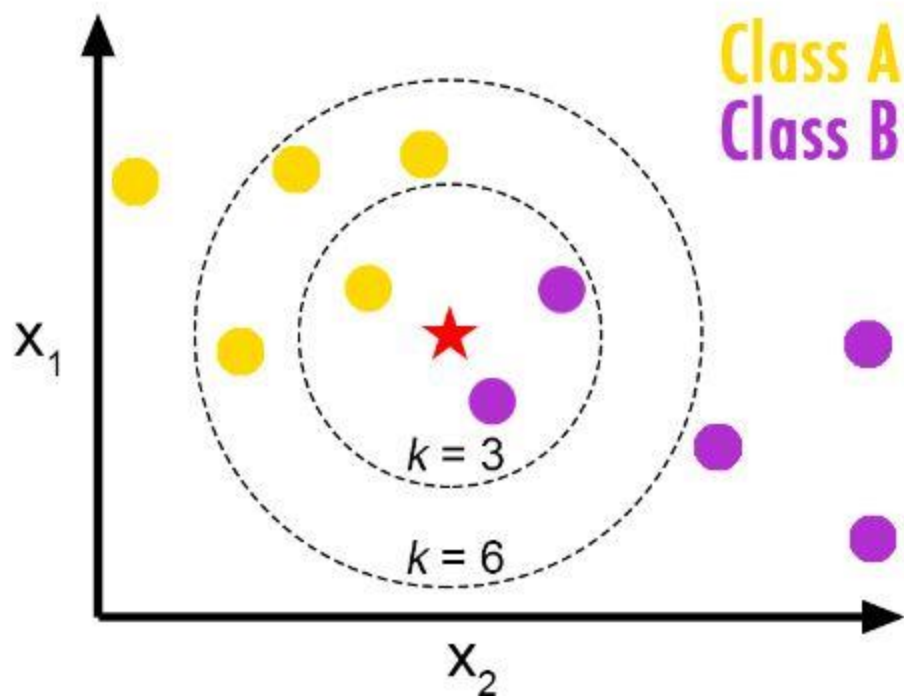


Find the  $k$  nearest neighbors and have them vote. Has a smoothing effect. This is especially good when there is noise in the class labels.

# k-nearest neighbor classifier



# k-nearest neighbor classifier



K is usually an odd number



# k-NN

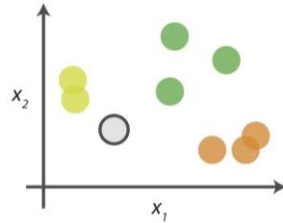
## Algorithm (kNN):

1. Find  $k$  examples  $\{\mathbf{x}^{(i)}, t^{(i)}\}$  closest to the test instance  $\mathbf{x}$
2. Classification output is majority class

$$y = \arg \max_{t^{(z)}} \sum_{r=1}^k \delta(t^{(z)}, t^{(r)})$$

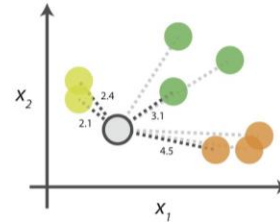
# k-NN algorithm in pictures

## 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

| Point Distance |              |
|----------------|--------------|
| ...            | 2.1 → 1st NN |
| ...            | 2.4 → 2nd NN |
| ...            | 3.1 → 3rd NN |
| ...            | 4.5 → 4th NN |

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.

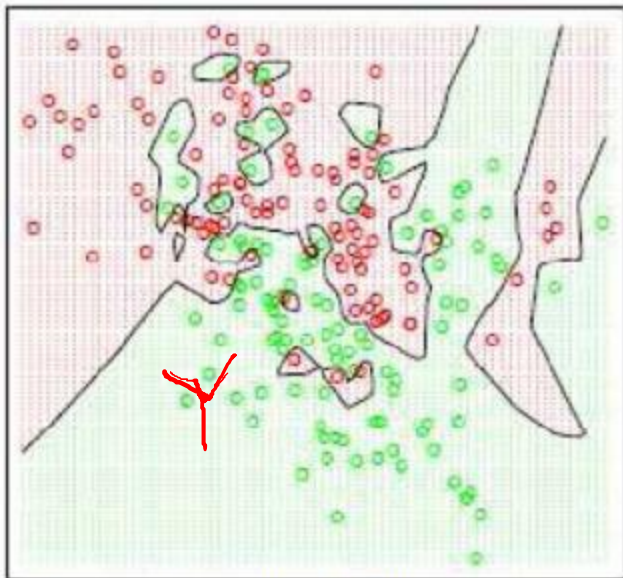
## 3. Vote on labels

| Class | # of votes |   |
|-------|------------|---|
|       | 2          | → Class  wins the vote!<br>Point  is therefore predicted to be of class . |
|       | 1          |   |
|       | 1          |   |

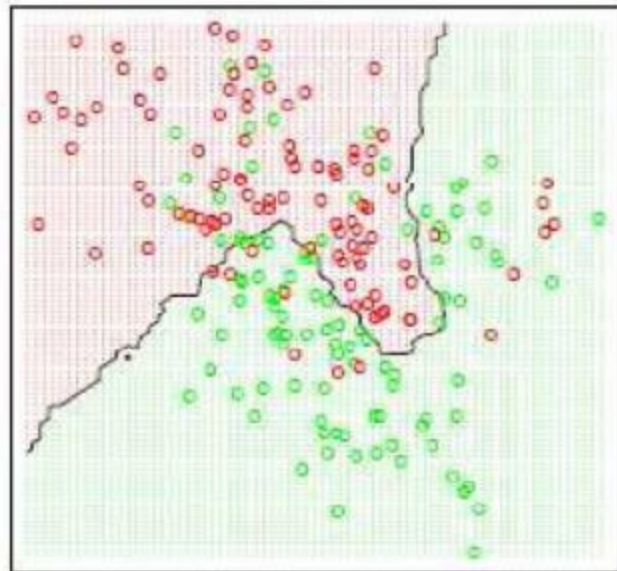
Vote on the predicted class labels based on the classes of the  $k$  nearest neighbours. Here, the labels were predicted based on the  $k=3$  nearest neighbours.

# Effect of K

K=1



K=15



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Larger k produces smoother boundary effect and can reduce the impact of class label noise.

$\left[ \begin{array}{c} \leftarrow d \rightarrow \end{array} \right]$

# Complexity of k-NN

$N$   
 $\mathbb{R}^d$

- Training

- Time:

$O(1)$

$O(1)$

- Space:

$O(Nd)$

$O(Nd)$

- Testing

- Time:

$O(Nd)$

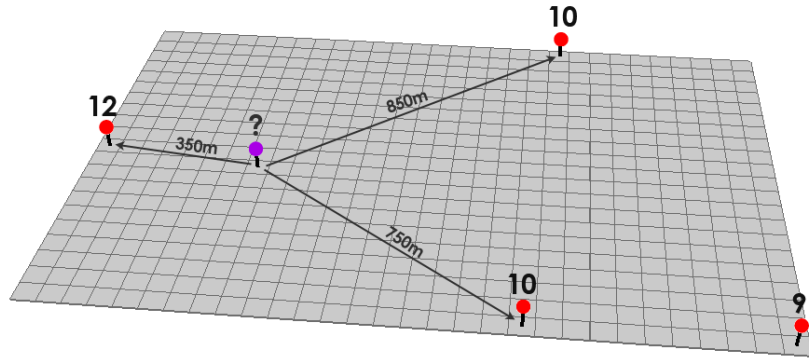
$O(Nd + k \log k)$

- Space:

$O(d)$

$O(kd) ?$

# Weighted k-NN

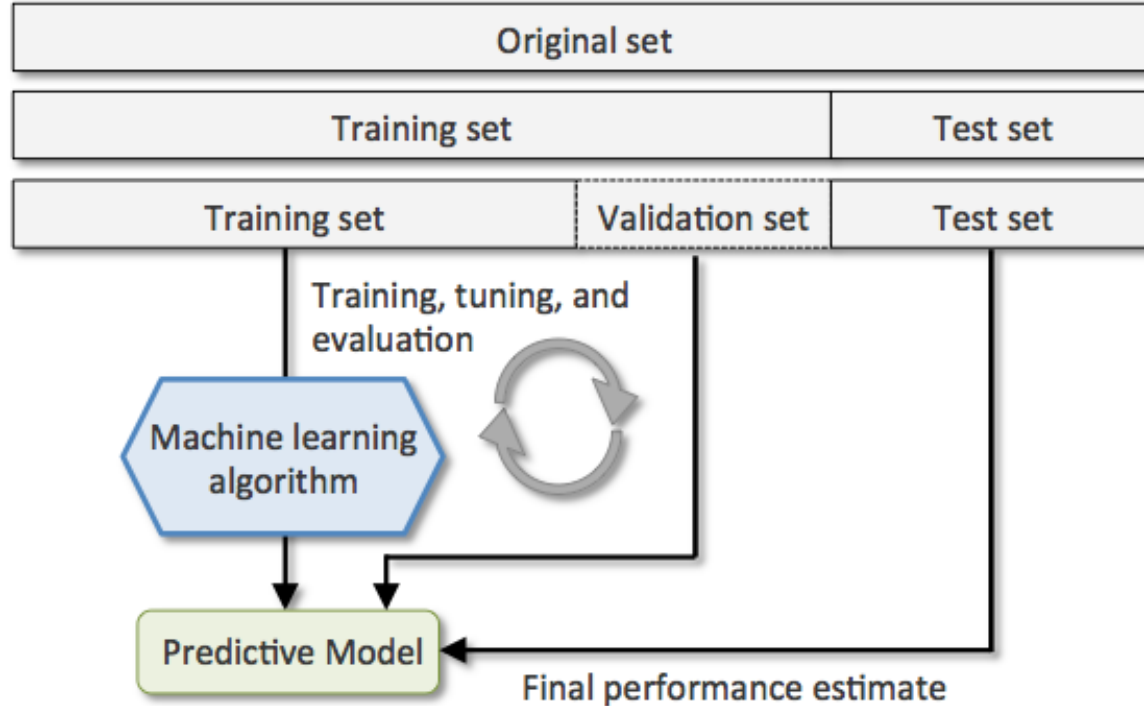


- Helps in case of class skew

# How to choose k ?

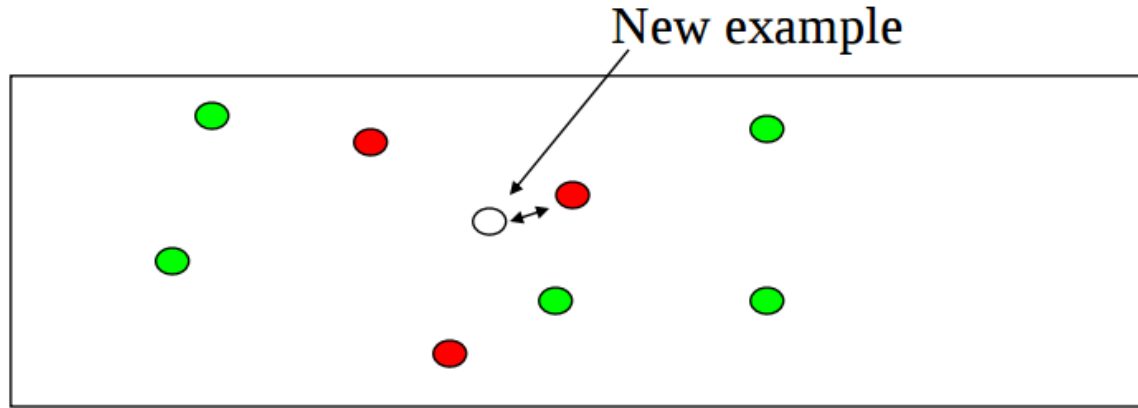


Rule of thumb:  $k < \sqrt{n}$ , where  $n$  is the number of training examples



# Nearest neighbor classifier

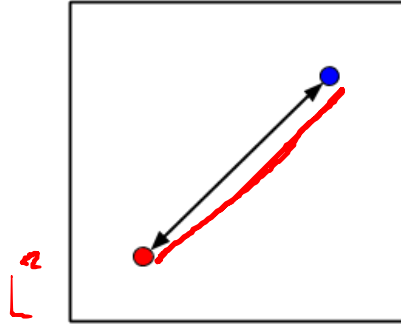
- Given a new example  $x$ , find the its closest training example  $\langle x^i, y^i \rangle$  and predict  $y^i$



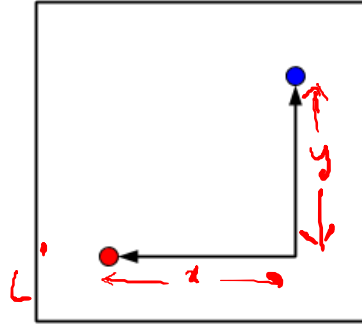
“Closest” →

# Distance measures

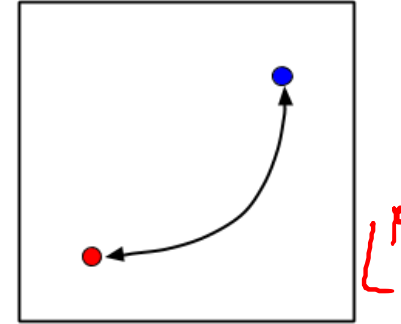
Euclidean



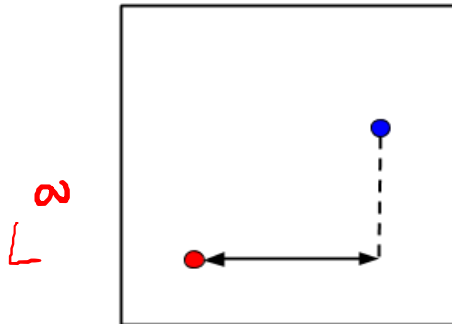
Manhattan



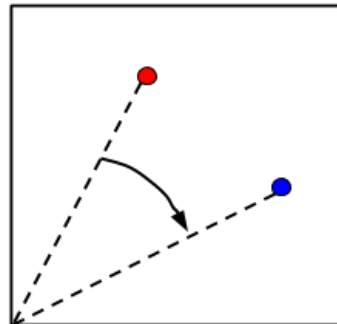
Minkowski



Chebychev



Cosine Similarity



Hamming



$[y_1, y_2, \dots, y_n]$   
 $[x_1, x_2, \dots, x_n]$   
 $m, n$



# Distance measures

Minkowski distance:

$$d_p = \|x_i - y_i\|_p = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$$

Chebyshev distance:

$$d_\infty = \|x_i - y_i\|_\infty = \max_{i=1..n} |x_i - y_i|$$

Cosine distance:

$$d_{cos} = 1 - \frac{\sum_{i=1}^n x_i * y_i}{\sqrt{\sum_{i=1}^n (x_i)^2} * \sqrt{\sum_{i=1}^n (y_i)^2}}$$

Mahalanobis  
distance:

$$d_M = \sqrt{\sum_{i=1}^n |x_i - y_i|^2 / s_i^2}$$

Euclidean distance:

$$d_2 = \|x_i - y_i\|_2 = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

Manhattan:

$$d_1 = \|x_i - y_i\|_1 = \sum_{i=1}^n |x_i - y_i|$$

Minkowski distance:

$$d_p = \|x_i - y_i\|_p = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$$

Chebyshev distance:

$$d_{\infty} = \|x_i - y_i\|_{\infty} = \max_{i=1..n} |x_i - y_i|$$

Cosine distance:

$$d_{cos} = 1 - \frac{\sum_{i=1}^n x_i * y_i}{\sqrt{\sum_{i=1}^n (x_i)^2} * \sqrt{\sum_{i=1}^n (y_i)^2}}$$

Distance based on  
Pearson correlation:

$$d_{corr} = 1 - \frac{\sum_{i=1}^n (x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} * \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Mahalanobis  
distance:

$$d_M = \sqrt{\sum_{i=1}^n |x_i - y_i|^2 / s_i^2}$$

Hellinger  
distance:

$$d_H = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^n |\sqrt{x_i} - \sqrt{y_i}|^2}$$

Euclidean distance:

$$d_2 = \|x_i - y_i\|_2 = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

Manhattan:

$$d_1 = \|x_i - y_i\|_1 = \sum_{i=1}^n |x_i - y_i|$$

Bray-Curtis  
distance:

$$d_{BC} = \sum_{i=1}^n |x_i - y_i| / \sum_{i=1}^n |x_i + y_i|$$

Canberra:

$$d_C = \sum_{i=1}^n |x_i - y_i| / (|x_i| + |y_i|)$$

# Properties and Issues with k-NN

- Non-parametric
- ‘Lazy’ learner (c.f. ‘eager’ learner in decision trees)
- Simple baseline (after 0-effort baselines)
- GOOD
  - No training
  - Learns highly non-linear decision boundaries
- BAD
  - Need to keep all training points around
  - Curse of dimensionality ! (suggested #dims < 20)

# Properties and Issues with k-NN

- If some attributes (coordinates of  $\mathbf{x}$ ) have larger **ranges**, they are treated as more important

$[0-1 \quad 2M-3M]$



# Properties and Issues with k-NN

- If some attributes (coordinates of  $\mathbf{x}$ ) 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:  $(x_j - m)/\sigma$ )

# Properties and Issues with k-NN

- If some attributes (coordinates of  $\mathbf{x}$ ) 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:  $(x_j - m)/\sigma$ )
  - ▶ be careful: sometimes scale matters

# Properties and Issues with k-NN

- If some attributes (coordinates of  $\mathbf{x}$ ) 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:  $(x_j - m)/\sigma$ )
  - ▶ be careful: sometimes scale matters
- **Irrelevant, correlated** attributes add noise to distance measure
  - ▶ eliminate some attributes
  - ▶ or vary and possibly adapt weight of attributes
- **Non-metric** attributes (symbols)
  - ▶ Hamming distance

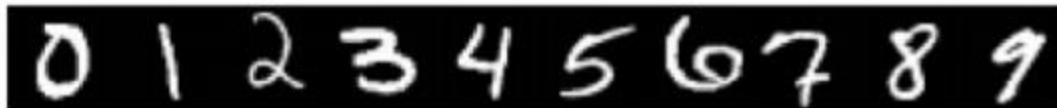
# Properties and Issues with k-NN

- If some attributes (coordinates of  $\mathbf{x}$ ) 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:  $(x_j - m)/\sigma$ )
  - ▶ be careful: sometimes scale matters
- **Irrelevant, correlated** attributes add noise to distance measure
  - ▶ eliminate some attributes
  - ▶ or vary and possibly adapt weight of attributes
- **Non-metric** attributes (symbols)
  - ▶ Hamming distance



# Some use cases for k-NN

- 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                 |

# Some use cases for k-NN

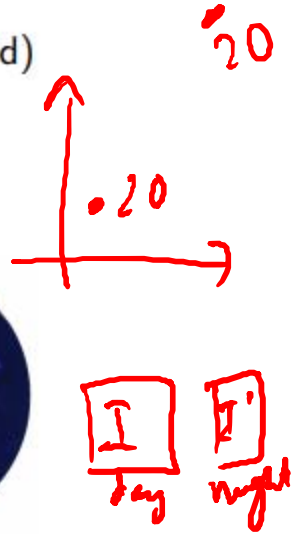
- Problem: Where (e.g., which country or GPS location) was this picture taken?



[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/>]

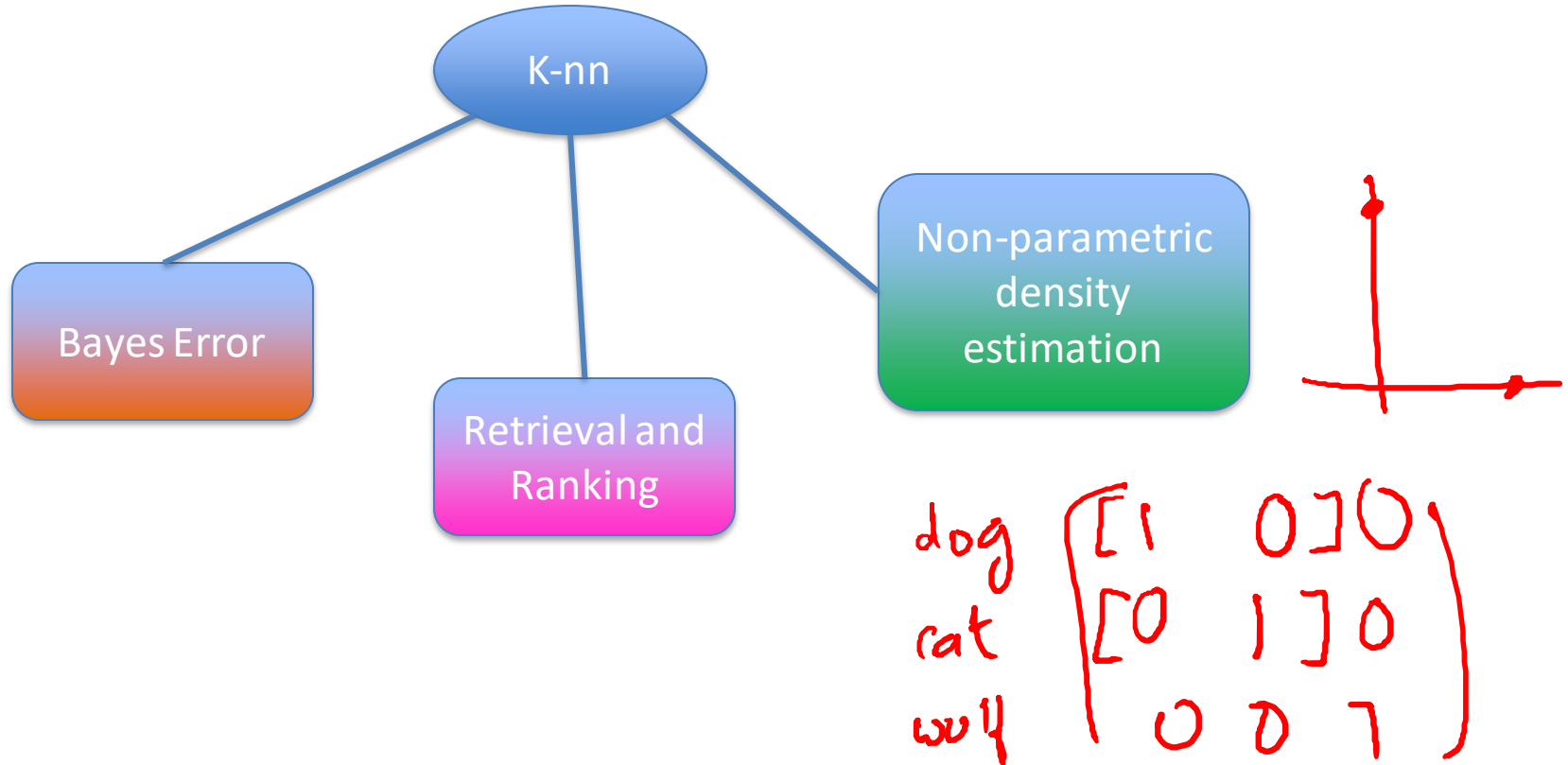
# Some use cases for k-NN

- 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/>]

# Related topics



$$\text{PROBABILITY} = \frac{\text{EVENT}}{\text{OUTCOMES}}$$



# Data – a probability-based perspective

- The basis for Statistical Learning Theory



Then we observe candies drawn from some bag: ● ● ● ● ● ● ● ● ● ●

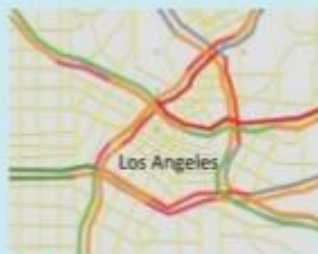
- Domain described by random variables (r.v.)
  - $X = \{\text{apple, grape}\}$
  - $b_i \in [1,5]$
- **Data = Instantiation of some or all r.v.'s in the domain**



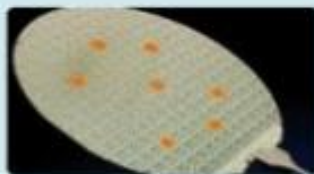
## Uncertainty arises from many sources

### Process Uncertainty

Processes contain  
"randomness"



Uncertain travel times



Semiconductor yield

### Data Uncertainty

Data input is uncertain



GPS Uncertainty



Testimony



{Paris Airport}

Ambiguity



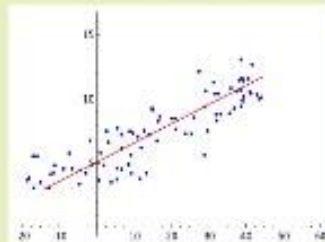
Contaminated?  
Rumors

{John Smith, Dallas}  
{John Smith, Kansas}

Conflicting Data

### Model Uncertainty

All modeling is approximate



Fitting a curve to data



Forecasting a hurricane  
([www.noaa.gov](http://www.noaa.gov))

# Data: a probabilistic perspective

## Output

|    | DBAName           | AKAName   | Address          | City           | State | Zip          |
|----|-------------------|-----------|------------------|----------------|-------|--------------|
| t1 | John Veliotis Sr. | Johnnyo's | 3465 S Morgan ST | <b>Chicago</b> | IL    | <b>60608</b> |
| t2 | John Veliotis Sr. | Johnnyo's | 3465 S Morgan ST | Chicago        | IL    | <b>60609</b> |
| t3 | John Veliotis Sr. | Johnnyo's | 3465 S Morgan ST | Chicago        | IL    | <b>60609</b> |
| t4 | <b>Johnnyo's</b>  | Johnnyo's | 3465 S Morgan ST | <b>Cicago</b>  | IL    | 60608        |

Conflicts

Does not obey data distribution

Conflict



### Proposed Cleaned Dataset

|    | DBAName                  | Address          | City           | State | Zip          |
|----|--------------------------|------------------|----------------|-------|--------------|
| t1 | John Veliotis Sr.        | 3465 S Morgan ST | Chicago        | IL    | 60608        |
| t2 | John Veliotis Sr.        | 3465 S Morgan ST | Chicago        | IL    | <b>60608</b> |
| t3 | John Veliotis Sr.        | 3465 S Morgan ST | Chicago        | IL    | <b>60608</b> |
| t4 | <b>John Veliotis Sr.</b> | 3465 S Morgan ST | <b>Chicago</b> | IL    | 60608        |

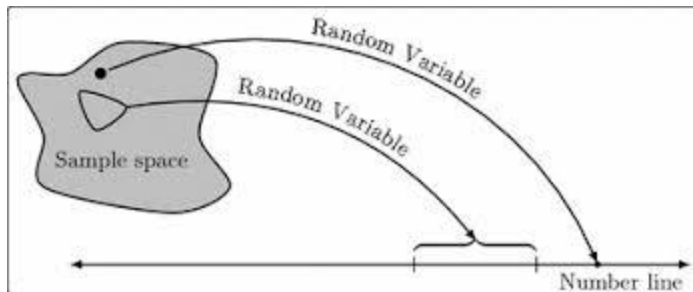
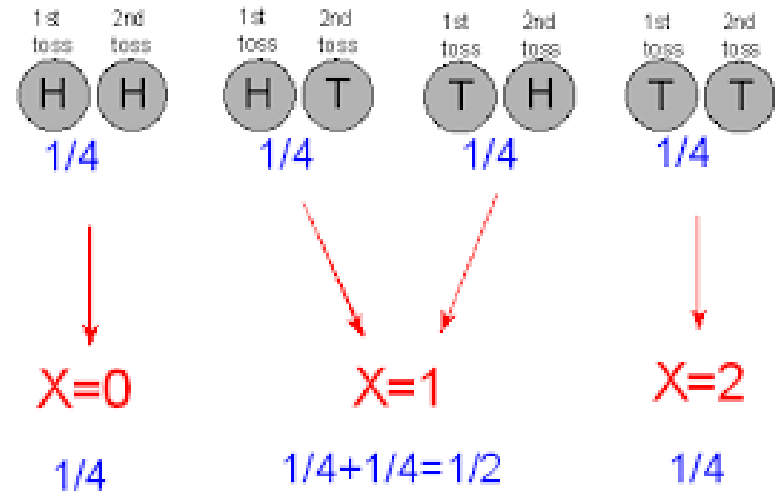
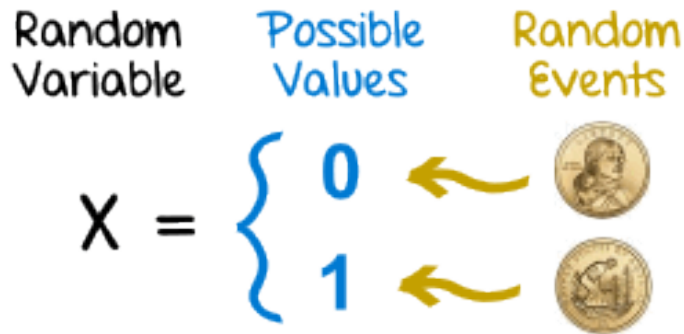
### Marginal Distribution of Cell Assignments

| Cell       | Possible Values   | Probability |
|------------|-------------------|-------------|
| t2.Zip     | 60608             | 0.84        |
|            | 60609             | 0.16        |
| t4.City    | Chicago           | 0.95        |
|            | Cicago            | 0.05        |
| t4.DBAName | John Veliotis Sr. | 0.99        |
|            | Johnnyo's         | 0.01        |



# Random Variables

R.V. = A numerical value from a random experiment



# Random variables

- A **discrete random variable** can assume a countable number of values.
  - Number of steps to the top of the Eiffel Tower\*



# Random variables

- A **discrete random variable** can assume a countable number of values.
  - Number of steps to the top of the Eiffel Tower\*
- A **continuous random variable** can assume any value along a given interval of a number line.
  - The time a tourist stays at the top once s/he gets there



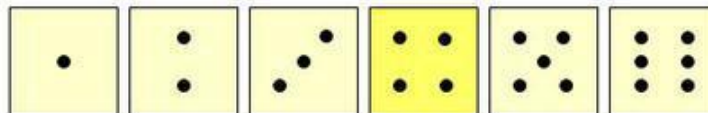
\*Believe it or not, the answer ranges from 1,652 to 1,789. See [Great Buildings](#)



# Discrete Random Variables

- Can only take on a countable number of values

Examples:



- Roll a die twice**

**Let  $X$  be the number of times 4 comes up**  
(then  $X$  could be 0, 1, or 2 times)

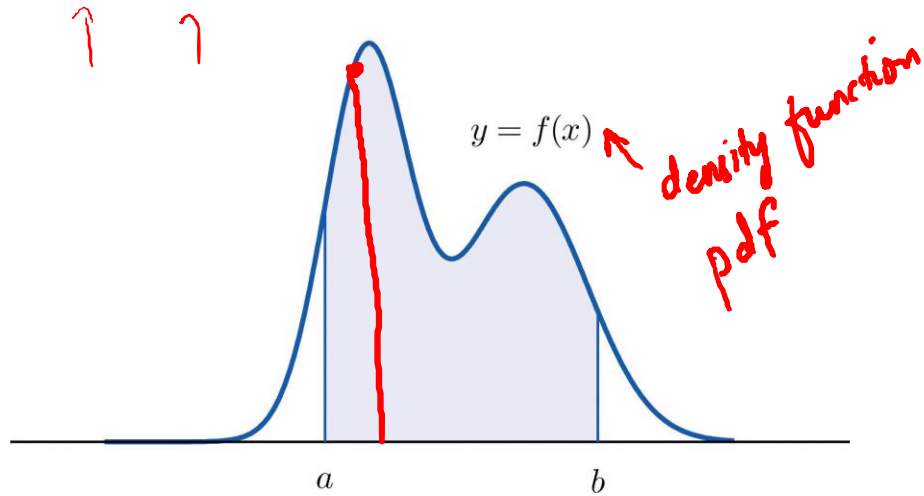
- Toss a coin 5 times.**

**Let  $X$  be the number of heads**  
(then  $X = 0, 1, 2, 3, 4, \text{ or } 5$ )

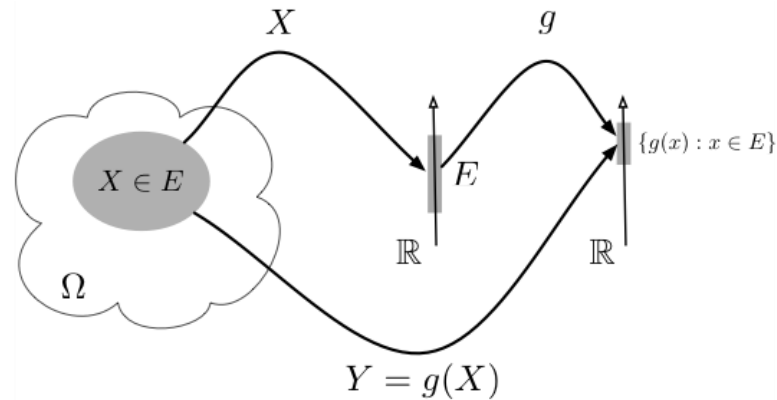


# Continuous random variable

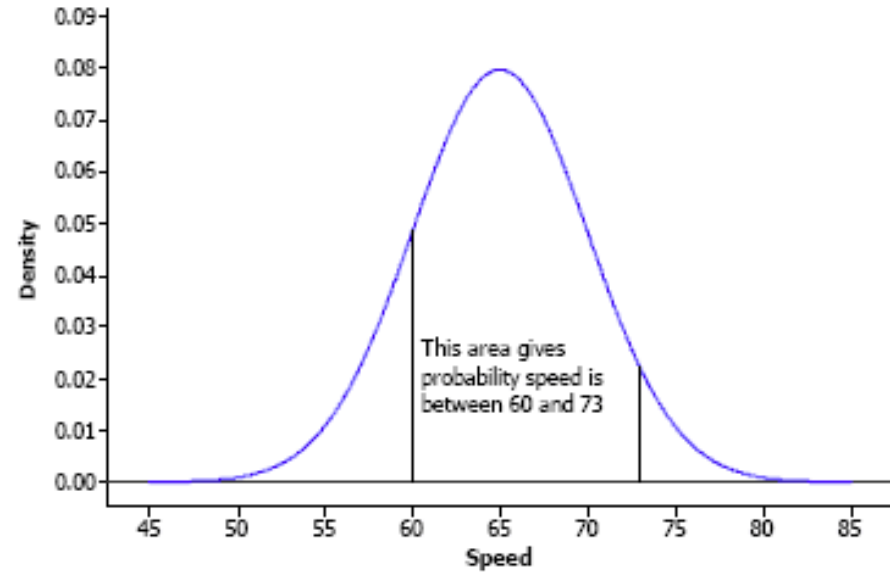
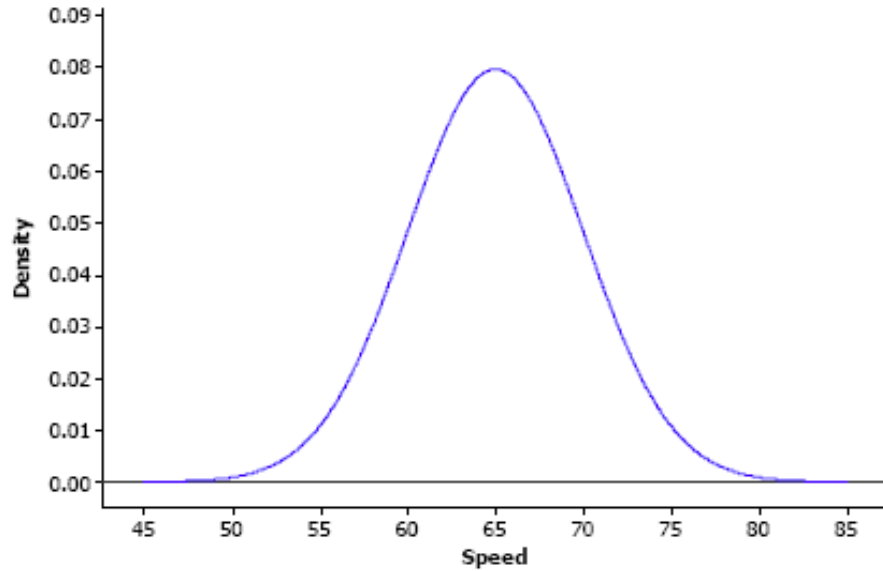
$P(a < X < b) = \text{area of shaded region}$



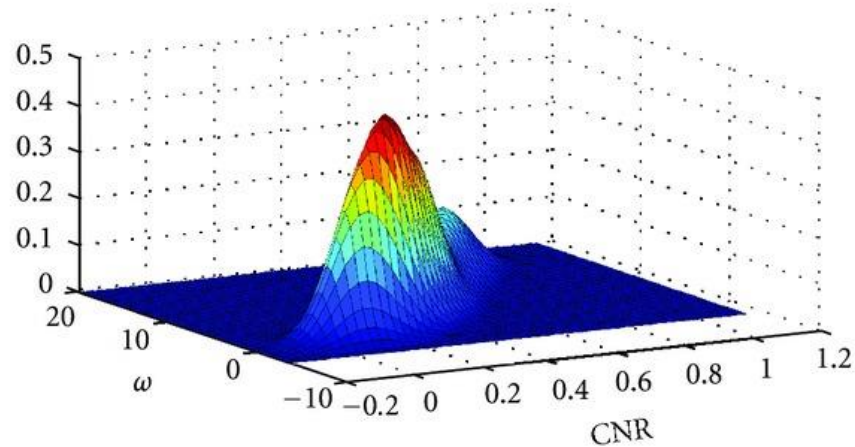
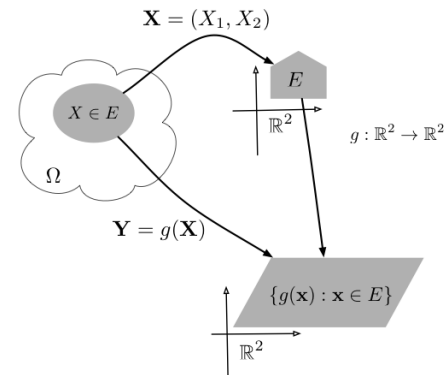
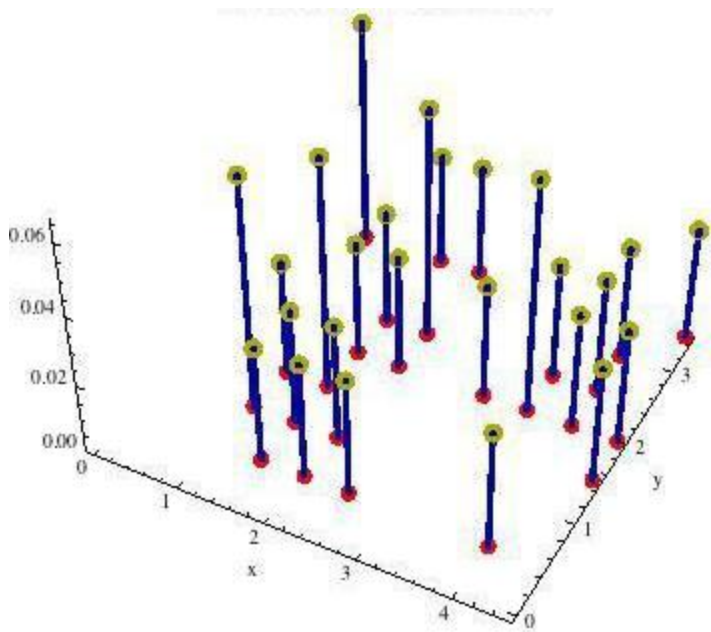
$$P(Y = 0.2) = 0$$



# Continuous random variable



# Random vectors



# References and Reading

- [https://en.wikipedia.org/wiki/K-nearest\\_neighbors\\_algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)
- Euclidean v/s Cosine distance
  - Code example: <https://cmry.github.io/notes/euclidean-v-cosine>
  - <https://stackoverflow.com/a/53175061>
  - <https://www.quora.com/Why-cosine-is-better-than-Euclidean-in-high-dimensional-data-as-in-text-documents>