CPSC429/529: Machine Learning

Lecture 2: Nearest Neighbor

Dongsheng Che
Computer Science Department
East Stroudsburg University

Nearest Neighbor

Similarity metrics

Similarity metrics

- A similarity metric measures the similarity between two instances according to a feature space
- One of the best known metrics is Euclidean distance which computes the length of the straight line between two points. Euclidean distance between two instances **a** and **b** in a *m*-dimensional feature space is defined as:

$$Euclidean(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{m} (\mathbf{a}[i] - \mathbf{b}[i])^2} \qquad (1)$$

Example

ID	Speed	Agility	Draft
1	2.50	6.00	No
2	3.75	8.00	No
3	2.25	5.50	No
4	3.25	8.25	No
5	2.75	7.50	No
6	4.50	5.00	No
7	3.50	5.25	No
8	3.00	3.25	No
9	4.00	4.00	No
10	4.25	3.75	No

ID	Speed	Agility	Draft
11	2.00	2.00	No
12	5.00	2.50	No
13	8.25	8.50	No
14	5.75	8.75	Yes
15	4.75	6.25	Yes
16	5.50	6.75	Yes
17	5.25	9.50	Yes
18	7.00	4.25	Yes
19	7.50	8.00	Yes
20	7.25	5.75	Yes

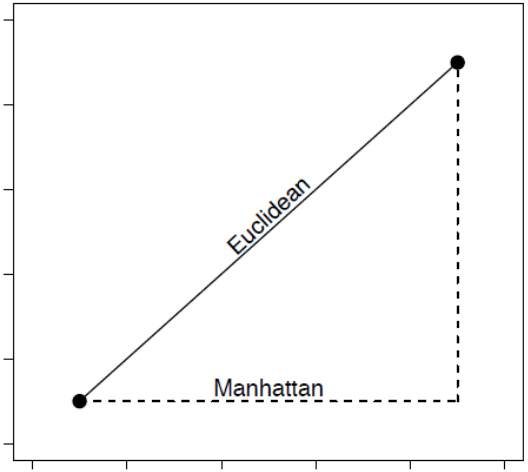
Example

The Euclidean distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75, AGILITY= 7.5) in Table 2 [25] is:

$$Euclidean(\langle 5.00, 2.50\rangle, \langle 2.75, 7.50\rangle) = \sqrt{(5.00-2.75)^2 + (2.50-7.50)^2}$$
$$= \sqrt{30.0625} = 5.4829$$

Manhattan distance

- Another distance measure is the Manhattan distance or taxi-cab distance.
- The Manhattan distance between two instances **a** and **b** in a feature space with *m* dimensions is:



$$Manhattan(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i]) \qquad (2)$$

Example

ID	Speed	Agility	Draft
1	2.50	6.00	No
2	3.75	8.00	No
3	2.25	5.50	No
4	3.25	8.25	No
5	2.75	7.50	No
6	4.50	5.00	No
7	3.50	5.25	No
8	3.00	3.25	No
9	4.00	4.00	No
10	4.25	3.75	No

ID	Speed	Agility	Draft
11	2.00	2.00	No
12	5.00	2.50	No
13	8.25	8.50	No
14	5.75	8.75	Yes
15	4.75	6.25	Yes
16	5.50	6.75	Yes
17	5.25	9.50	Yes
18	7.00	4.25	Yes
19	7.50	8.00	Yes
20	7.25	5.75	Yes

Example

The Manhattan distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75, AGILITY= 7.5) in Table 2 [25] is:

$$\begin{aligned} Manhattan(\langle 5.00, 2.50\rangle, \langle 2.75, 7.50\rangle) &= abs(5.00 - 2.75) + abs(2.5 - 7.5) \\ &= 2.25 + 5 = 7.25 \end{aligned}$$

Minkowski distance

- The Euclidean and Manhattan distances are special cases of Minkowski distance
- The Minkowski distance between two instances **a** and **b** in a feature space with m descriptive features is:

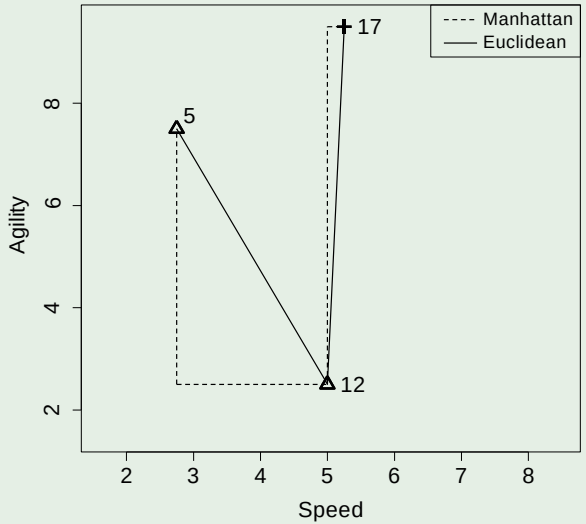
$$Minkowski(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i])^p \right)^{\frac{1}{p}} \quad (3)$$

where different values of the parameter p result in different distance metrics

- The Minkowski distance with $p = 1$ is the Manhattan distance and with $p = 2$ is the Euclidean distance.

Example

Instance ID	Instance ID	Manhattan (Minkowski p=1)	Euclidean (Minkowski p=2)
12	5	7.25	5.4829
12	17	7.25	8.25



The Manhattan and Euclidean distances between instances $\mathbf{d}_{12}$ (SPEED= 5.00, AGILITY= 2.5) and $\mathbf{d}_5$ (SPEED= 2.75, AGILITY= 7.5) and between instances $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ (SPEED= 5.25, AGILITY= 9.5).

Nearest Neighbor

Similarity metrics

Nearest Neighbor

The Nearest Neighbor Algorithm

The Nearest Neighbor Algorithm

Require: set of training instances
Require: a query to be classified

1: Iterate across the instances in memory and find the instance that is shortest distance from the query position in the feature space.
2: Make a prediction for the query equal to the value of the target feature of the nearest neighbor.

A worked example

Table: The speed and agility ratings for 20 college athletes labelled with the decisions for whether they were drafted or not.

ID	Speed	Agility	Draft	ID	Speed	Agility	Draft
1	2.50	6.00	No	11	2.00	2.00	No
2	3.75	8.00	No	12	5.00	2.50	No
3	2.25	5.50	No	13	8.25	8.50	No
4	3.25	8.25	No	14	5.75	8.75	Yes
5	2.75	7.50	No	15	4.75	6.25	Yes
6	4.50	5.00	No	16	5.50	6.75	Yes
7	3.50	5.25	No	17	5.25	9.50	Yes
8	3.00	3.25	No	18	7.00	4.25	Yes
9	4.00	4.00	No	19	7.50	8.00	Yes
10	4.25	3.75	No	20	7.25	5.75	Yes

Example

- Should we draft an athlete with the following profile:

SPEED= 6.75, AGILITY= 3

Feature Space Plot

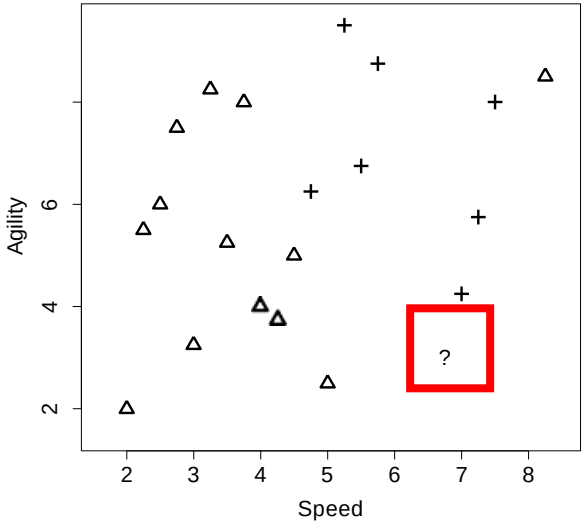


Figure: A feature space plot with the position in the feature space of the query represented by the ? marker. The triangles represent *'Non-draft'* instances and the crosses represent the *'Draft'* instances.

Nearest neighbor using Euclidean distance

Table: The distances (Dist.) between the query instance with SPEED = 6.75 and AGILITY = 3.00 and each instance in Table 2 [25].

ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27
12	5.00	2.50	no	1.82
10	4.25	3.75	no	2.61
20	7.25	5.75	yes	2.80
9	4.00	4.00	no	2.93
6	4.50	5.00	no	3.01
8	3.00	3.25	no	3.76
15	4.75	6.25	yes	3.82
7	3.50	5.25	no	3.95
16	5.50	6.75	yes	3.95

ID	SPEED	AGILITY	DRAFT	Dist.
11	2.00	2.00	no	4.85
19	7.50	8.00	yes	5.06
3	2.25	5.50	no	5.15
1	2.50	6.00	no	5.20
13	8.25	8.50	no	5.70
2	3.75	8.00	no	5.83
14	5.75	8.75	yes	5.84
5	2.75	7.50	no	6.02
4	3.25	8.25	no	6.31
17	5.25	9.50	yes	6.67

$$Euclidean(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{m} (\mathbf{a}[i] - \mathbf{b}[i])^2} \qquad (1)$$

Euclidean(q, d18) = Sqrt((6.75-7.00)^2+(3.00-4.25)^2)=1.27

Nearest Neighbor

The Nearest Neighbor Algorithm

Nearest Neighbor

Handling Noisy Data

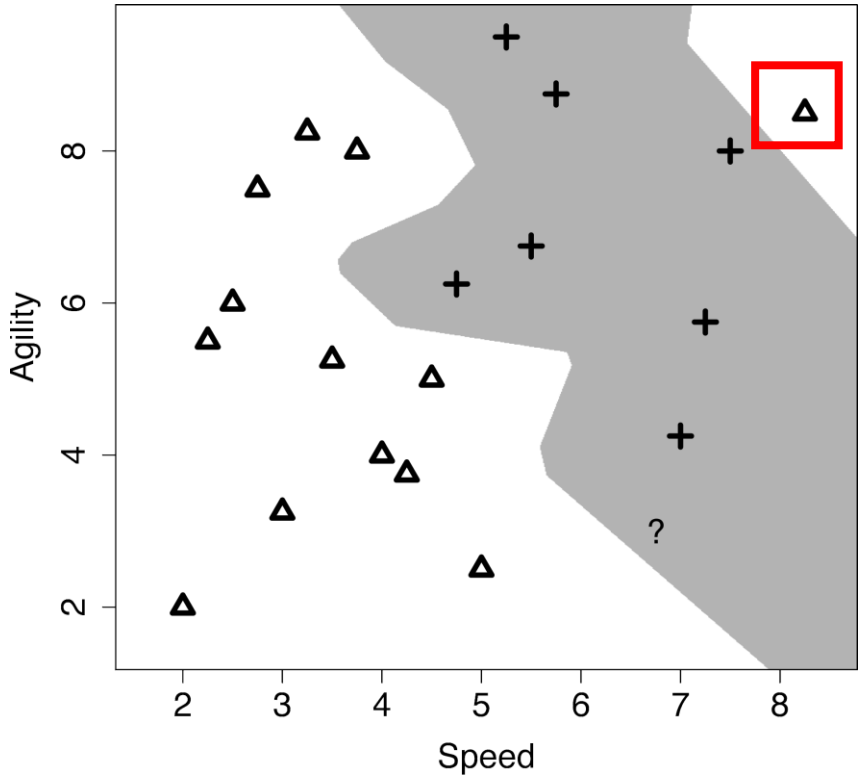


Figure: Is the instance at the top right of the diagram really *noise*?

k nearest neighbors

- The k nearest neighbors model predicts the target level with the majority vote from the set of k nearest neightbors to the query **q**:

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\text{argmax}} \sum_{i=1}^{k} \delta(t_i, l) \qquad (1)$$

Note:

- levels(t)- is a set of target values, e.g.,
  - Binary: {-1, +1}
  - Multi-class: {class1, class2, class3}
- Kronecker delta:

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

K-Nearest neighbor example

Table: The distances (Dist.) between the query instance with SPEED = 6.75 and AGILITY = 3.00 and each instance in Table 2 [25].

ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27
12	5.00	2.50	no	1.82
10	4.25	3.75	no	2.61
20	7.25	5.75	yes	2.80
9	4.00	4.00	no	2.93
6	4.50	5.00	no	3.01
8	3.00	3.25	no	3.76
15	4.75	6.25	yes	3.82
7	3.50	5.25	no	3.95
16	5.50	6.75	yes	3.95

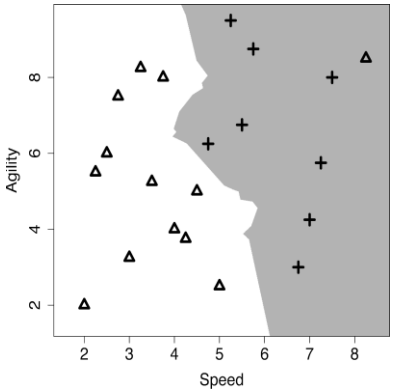
ID	SPEED	AGILITY	DRAFT	Dist.
11	2.00	2.00	no	4.85
19	7.50	8.00	yes	5.06
3	2.25	5.50	no	5.15
1	2.50	6.00	no	5.20
13	8.25	8.50	no	5.70
2	3.75	8.00	no	5.83
14	5.75	8.75	yes	5.84
5	2.75	7.50	no	6.02
4	3.25	8.25	no	6.31
17	5.25	9.50	yes	6.67

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\text{argmax}} \sum_{i=1}^{k} \delta(t_i, l)$$

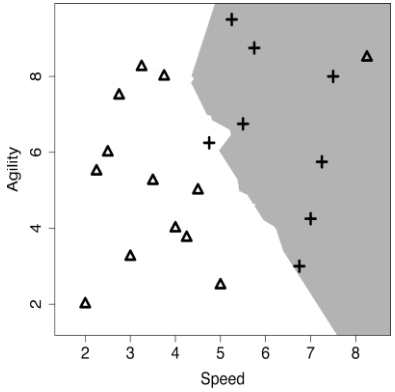
K=5

$M_k(q)$= majority(2 “yes”, 3 “no”) =“no”

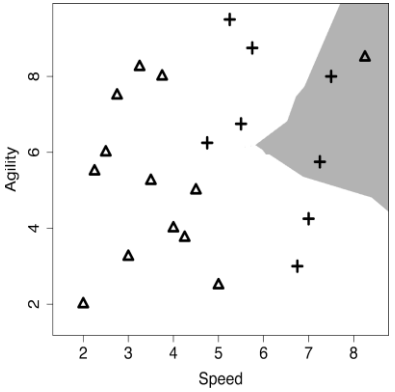
Decision boundaries with different K



K=3



K=5



K=15

Weighted K-Nearest neighbor

- In a distance weighted *k* nearest neighbor algorithm the contribution of each neighbor to the classification decision is weighted by the reciprocal of the squared distance between the neighbor **d** and the query **q**:

$$\frac{1}{dist(\mathbf{q}, \mathbf{d})^2} \tag{2}$$

- The weighted *k* nearest neighbor model is defined as:

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\text{argmax}} \sum_{i=1}^{k} \frac{1}{dist(\mathbf{q}, \mathbf{d_i})^2} \times \delta(t_i, l) \tag{3}$$

Weighted K-Nearest neighbor example

Table: The distances (Dist.) between the query instance with SPEED = 6.75 and AGILITY = 3.00 and each instance in Table 2 [25].

ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27
12	5.00	2.50	no	1.82
10	4.25	3.75	no	2.61
20	7.25	5.75	yes	2.80
9	4.00	4.00	no	2.93
6	4.50	5.00	no	3.01
8	3.00	3.25	no	3.76
15	4.75	6.25	yes	3.82
7	3.50	5.25	no	3.95
16	5.50	6.75	yes	3.95

ID	SPEED	AGILITY	DRAFT	Dist.
11	2.00	2.00	no	4.85
19	7.50	8.00	yes	5.06
3	2.25	5.50	no	5.15
1	2.50	6.00	no	5.20
13	8.25	8.50	no	5.70
2	3.75	8.00	no	5.83
14	5.75	8.75	yes	5.84
5	2.75	7.50	no	6.02
4	3.25	8.25	no	6.31
17	5.25	9.50	yes	6.67

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\text{argmax}} \sum_{i=1}^{k} \frac{1}{dist(\mathbf{q}, \mathbf{d_i})^2} \times \delta(t_i, l)$$

K=5,

$M_k(q)$= argmax(1/((1.27)^2)+1/((2.80)^2)“yes”,
1/(1.82)^2+1/(2.61)^2 +1/(2.93)^2 “no”)
= argmax(0.74755 “yes”, 0.5651 “no”) = “yes”

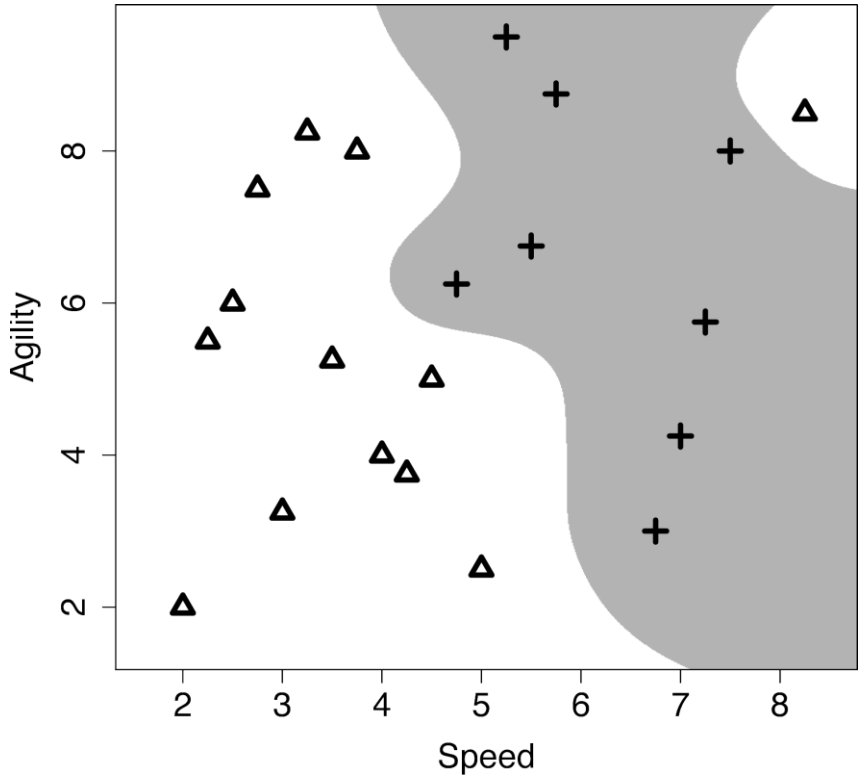


Figure: The weighted *k* nearest neighbor model decision boundary.

Nearest Neighbor

Handling Noisy Data

Nearest Neighbor

Data Normalization

Dataset and Query

Table: A dataset listing the salary and age information for customers and whether or not the purchased a pension plan .

ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes

- The marketing department wants to decide whether or not they should contact a customer with the following profile:

$$\langle \text{SALARY} = 56,000, \text{AGE} = 35 \rangle$$

Prediction without normalization

ID	Salary	Age	Purch.	Salary and Age Dist.	Salary and Age Neigh.	Salary Only Dist.	Salary Only Neigh.	Age Only Dist.	Age Only Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8

Feature Space

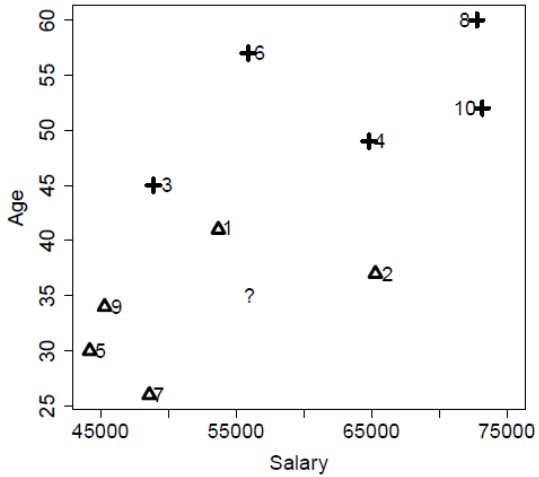


Figure: The salary and age feature space with the data in Table 1 [12] plotted. The instances are labelled their IDs, triangles represent the negative instances and crosses represent the positive instances. The location of the query $\langle \text{SALARY} = 56,000, \text{AGE} = 35 \rangle$ is indicated by the ?.

Prediction after range normalization

- We can adjust for this using normalization; the equation for range normalization is:

$$a_i' = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low \qquad (4)$$

	Normalized Dataset			Salary and Age		Salary Only		Age Only	
ID	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	0.3276	0.4412	No	0.1935	1	0.0793	2	0.17647	4
2	0.7276	0.3235	No	0.3260	2	0.3207	6	0.05882	2
3	0.1621	0.5588	Yes	0.3827	5	0.2448	3	0.29412	6
4	0.7103	0.6765	Yes	0.5115	7	0.3034	5	0.41176	7
5	0.0000	0.1176	No	0.4327	6	0.4069	8	0.14706	3
6	0.4034	0.9118	Yes	0.6471	8	0.0034	1	0.64706	9
7	0.1517	0.0000	No	0.3677	3	0.2552	4	0.26471	5
8	0.9862	1.0000	Yes	0.9361	10	0.5793	9	0.73529	10
9	0.0379	0.2353	No	0.3701	4	0.3690	7	0.02941	1
10	1.0000	0.7647	Yes	0.7757	9	0.5931	10	0.50000	8

A few notes on normalization

- You don’t need to do normalization on target column
- You also need to do normalization on test data before predicting on testing data
- Normalizing the data is an important thing to do for almost all machine learning algorithms, including nearest neighbor!

Nearest Neighbor

Data Normalization

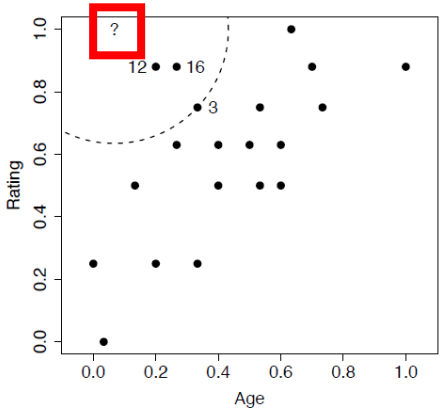
Nearest Neighbor

Predicting Continuous Targets (Regression)

k-nearest neighbor for regression

ID	Age	Rating	Price
1	0.0000	0.25	30.00
2	0.4000	0.63	40.00
3	0.3333	0.75	55.00
4	0.7000	0.88	550.00
5	0.4000	0.50	35.00
6	0.5000	0.63	45.00
7	0.5333	0.75	70.00
8	0.6000	0.50	85.00
9	0.6000	0.63	78.00
10	0.5333	0.50	75.00

ID	Age	Rating	Price
11	0.6333	1.00	500.00
12	0.2000	0.88	200.00
13	0.2667	0.63	65.00
14	0.7333	0.75	120.00
15	0.2000	0.25	12.00
16	0.2667	0.88	250.00
17	0.3333	0.25	18.00
18	1.0000	0.88	450.00
19	0.0333	0.00	10.00
20	0.1333	0.50	30.00



- Return the average value in the neighborhood:

$$\mathbb{M}_k(\mathbf{q}) = \frac{1}{k}\sum_{i=1}^{k} t_i \qquad (5)$$

- The model will return a price prediction that is the average price of the three neighbors:

$$\frac{200.00 + 250.00 + 55.00}{3} = 168.33$$

Weighted k-nearest neighbor for regression

$$\mathbb{M}_k(\mathbf{q}) = \frac{\sum_{i=1}^{k} \frac{1}{dist(\mathbf{q}, \mathbf{d}_i)^2} \times t_i}{\sum_{i=1}^{k} \frac{1}{dist(\mathbf{q}, \mathbf{d}_i)^2}}$$

ID	Price	Distance	Weight	Price × Weight
1	30.00	0.7530	1.7638	52.92
2	40.00	0.5017	3.9724	158.90
3	55.00	0.3655	7.4844	411.64
4	550.00	0.6456	2.3996	1319.78
5	35.00	0.6009	2.7692	96.92
6	45.00	0.5731	3.0450	137.03
7	70.00	0.5294	3.5679	249.75
8	85.00	0.7311	1.8711	159.04
9	78.00	0.6520	2.3526	183.50
10	75.00	0.6839	2.1378	160.33
11	500.00	0.5667	3.1142	1557.09
12	200.00	0.1828	29.9376	5987.53
13	65.00	0.4250	5.5363	359.86
14	120.00	0.7120	1.9726	236.71
15	12.00	0.7618	1.7233	20.68
16	250.00	0.2358	17.9775	4494.38
17	18.00	0.7960	1.5783	28.41
18	450.00	0.9417	1.1277	507.48
19	10.00	1.0006	0.9989	9.99
20	30.00	0.5044	3.9301	117.90
		Totals:	99.2604	16 249.85

K=20

Numerator = 16249.85

Denominator = 99.264

$M_k(q)$= 16249.85/ 99.264 =163.71

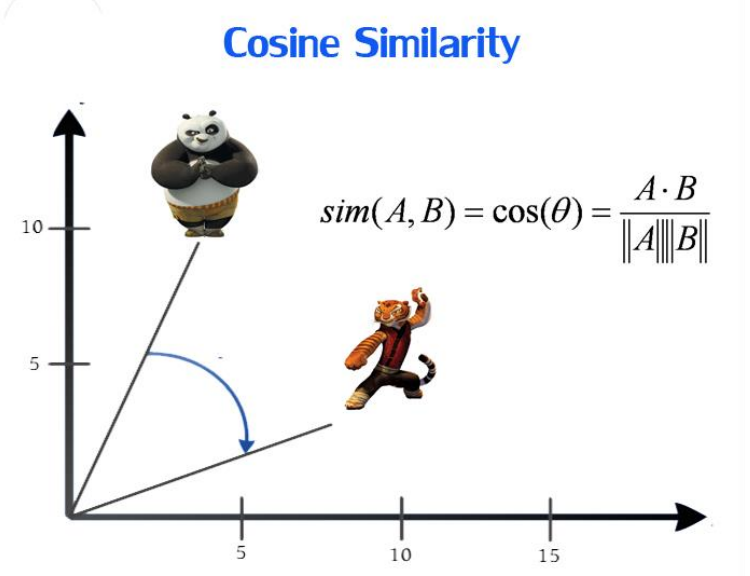
Nearest Neighbor

Predicting Continuous Targets (Regression)

Nearest Neighbor

Other Measures of Similarity

Cosine similarity



$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

https://dataaspirant.com/five-most-popular-similarity-measures-implementation-in-python/

Example using cosine similarity

```
doc_1 = "Data is the oil of the digital economy"
doc_2 = "Data is a new oil"

# Vector representation of the document
doc_1_vector = [1, 1, 1, 1, 0, 1, 1, 2]
doc_2_vector = [1, 0, 0, 1, 1, 0, 1, 0]
```

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

https://studymachinelearning.com/cosine-similarity-text-similarity-metric/

Cosine similarity calculation

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

$$= (1 * 1) + (1 * 0) + (1 * 0) + (1 * 1) + (0 * 1) + (1 * 0) + (1 * 1) + (2 * 0)$$

$$= 3$$

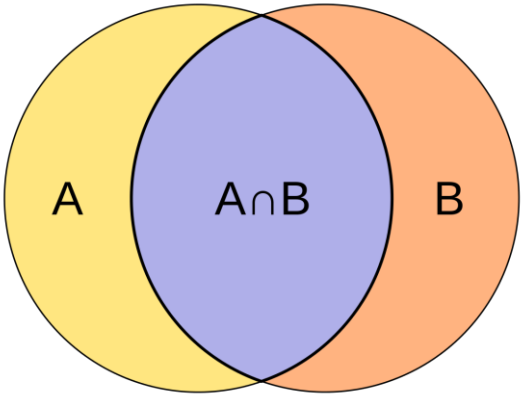
$$\sqrt{\sum_{i=1}^{n} A_i^2} = \sqrt{1 + 1 + 1 + 1 + 0 + 1 + 1 + 4} = \sqrt{10}$$

$$\sqrt{\sum_{i=1}^{n} B_i^2} = \sqrt{1 + 0 + 0 + 1 + 1 + 0 + 1 + 0} = \sqrt{4}$$

$$cosine\ similarity = cos\theta = \frac{A \cdot B}{|A||B|} = \frac{3}{\sqrt{10} * \sqrt{4}} = 0.4743$$

https://studymachinelearning.com/cosine-similarity-text-similarity-metric/

Jaccard Similarity



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Jaccard Similarity Example

Let's see the example about how to **Jaccard Similarity** work?

```
doc_1 = "Data is the new oil of the digital economy"
doc_2 = "Data is a new oil"
```

Let's get the set of unique words for each document.

```
words_doc1 = {'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'}
words_doc2 = {'data', 'is', 'a', 'new', 'oil'}
```

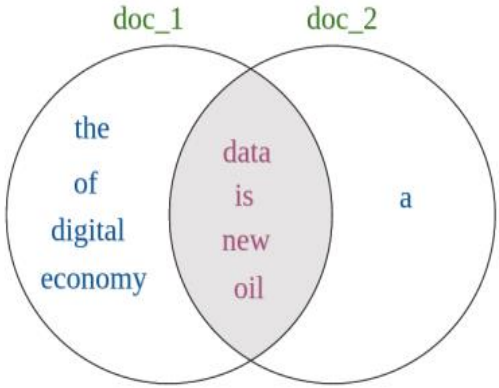

https://studymachinelearning.com/jaccard-similarity-text-similarity-metric-in-nlp/

Jaccard Similarity Calculation

$$J(doc_1, doc_2) = \frac{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cap \{'data', 'is', 'a', 'new', 'oil'\}}{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cup \{'data', 'is', 'a', 'new', 'oil'\}}$$

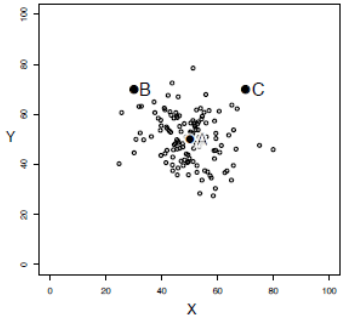
$$= \frac{\{'data', 'is', 'new', 'oil'\}}{\{'data', 'a', 'of', 'is', 'economy', 'the', 'new', 'digital', 'oil'\}}$$

$$= \frac{4}{9} = 0.444$$

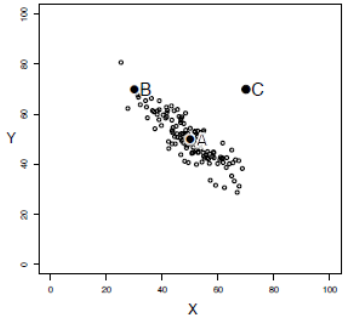


https://studymachinelearning.com/jaccard-similarity-text-similarity-metric-in-nlp/

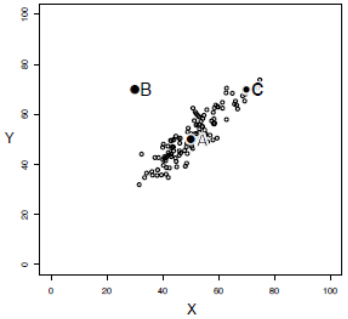
Mahalanobis Distance



(a)



(b)



(c)

Euclidean Distance $d_{L2}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ $d_{L2}(x,y) = \sqrt{(x-y)^T(x-y)}$

Mahalanobis Distance $d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$

S^{-1} is inverse matrix of S, where S is a covariance matrix

Variance/covariance/covariance matrix

Variance

Population mean is known

$$var(x) = \frac{\sum_i^n (x_i - \mu)^2}{N}$$

Population mean is unknown

$$var(x) = \frac{\sum_i^n (x_i - \overline{x})^2}{N - 1}$$

Covariance

Population mean is known

$$cov(x, y) = \frac{\sum_i^n (x_i - \mu) \cdot (y_i - \mu)}{N}$$

Population mean is unknown

$$cov(x) = \frac{\sum_i^n (x_i - \overline{x}) \cdot (y_i - \overline{y})}{N - 1}$$

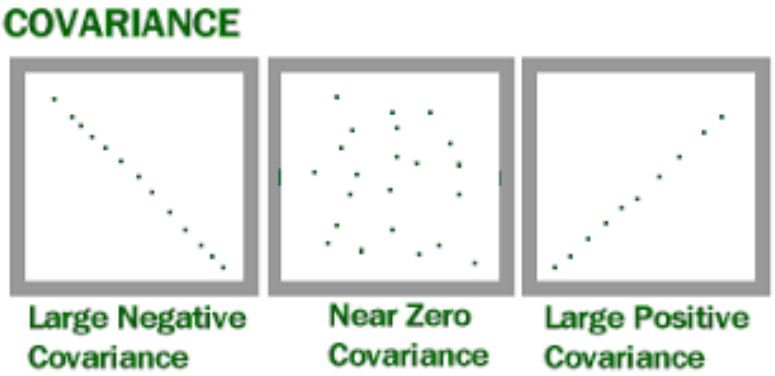
Covariance matrix

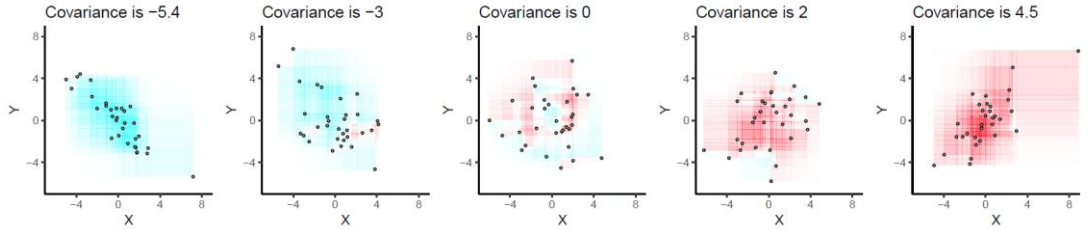
$$\begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} var(x) & cov(x, y) \\ cov(x, y) & var(y) \end{bmatrix} \end{array}$$

$$\begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{bmatrix} var(x) & cov(x, y) & cov(x, z) \\ cov(x, y) & var(y) & cov(y, z) \\ cov(x, z) & cov(y, z) & var(z) \end{bmatrix} \end{array}$$

https://towardsdatascience.com/5-things-you-should-know-about-covariance-26b12a0516f1

What does covariance value tell you?





S^{-1}: Inverse of Matrix S

2X2 Matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \qquad A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

3X3 Matrix $M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix},$

$$M^{-1} = \frac{1}{|M|} \begin{bmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{bmatrix}$$

$$|M| = aei + bfg + cdh - (ceg + bdi + afh)$$

A few notes on mahalanobis distance

- The mahalanobis distance uses covariance to scale distances so that distances along a direction where the dataset is spreadout a lot are scaled down and distances along directions where the dataset is tightly packed are scaled up.

- Similar to Euclidean distance, the mahalanobis distance squares the differences of the features.
- But it also rescales the differences (using the inverse covariance matrix) so that all the features have unit variance and the effects of covariance is removed.

Euclidean Distance $d_{L2}(x,y) = \sqrt{(x-y)^T(x-y)}$

Mahalanobis Distance $d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$

Mahalanobis distance contours

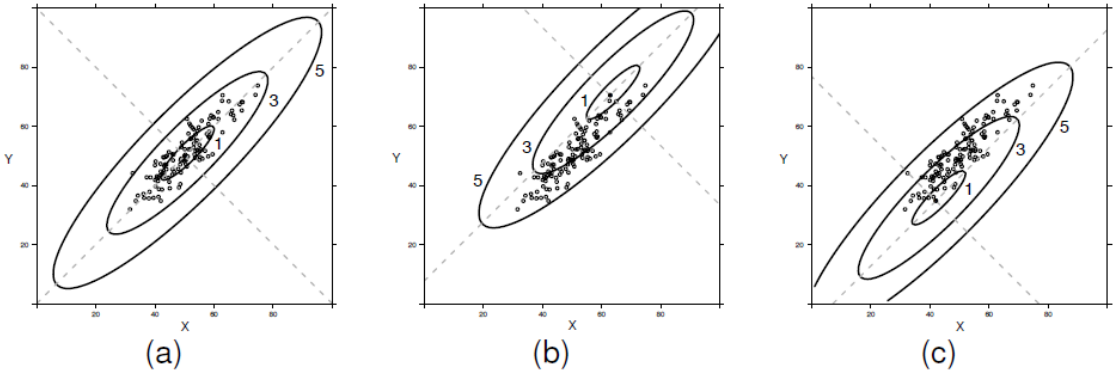
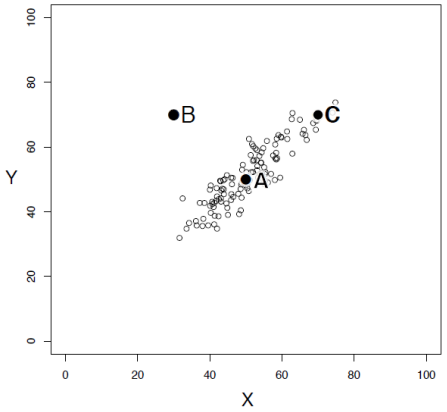
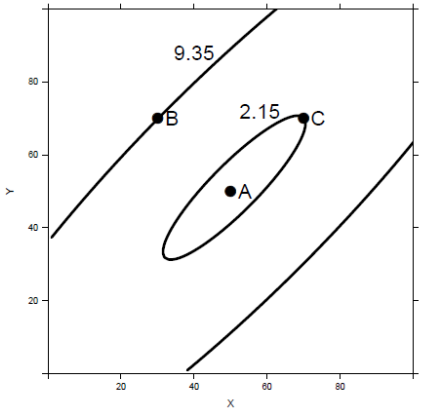


Figure: The coordinate systems defined by the mahalanobis metric using the co-variance matrix for the dataset in Figure 9(c) [46] using three different origins: (a) $(50, 50)$, (b) $(63, 71)$, (c) $(42, 35)$. The ellipses in each figure plot the 1, 3, and 5 unit distances contours from each origin.

Mahalanobis distance calculation





Mahalanobis Distance

$$d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

A (50,50)
B (30,70)
C (70,70)

$$Mahalanobis(A,B) = \sqrt{[50-30, 50-70] \times \begin{bmatrix} 0.059 & -0.521 \\ -0.521 & 0.0578 \end{bmatrix} \times \begin{bmatrix} 50-30 \\ 50-70 \end{bmatrix}} = 9.4049$$

$$Mahalanobis(A,C) = \sqrt{[50-70, 50-70] \times \begin{bmatrix} 0.059 & -0.521 \\ -0.521 & 0.0578 \end{bmatrix} \times \begin{bmatrix} 50-70 \\ 50-70 \end{bmatrix}} = 2.2540$$

Nearest Neighbor

Other Measures of Similarity

Nearest Neighbor

K-nearest Neighbor
In Scikit-Learn

Loading Iris dataset

In [2]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Note: Original dataset has four features: SepalLength[cm], SepalWidth[cm],
#       PetalLength[cm], PetalWidth[cm]. We only pick last two features only
iris = load_iris()
X, y = iris.data[:, 2:], iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=123,
                                                    shuffle=True)
```

Train K-NN

In [5]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier(n_neighbors=3)
knn_clf.fit(X_train, y_train)
```

Out[5]:
```
KNeighborsClassifier(n_neighbors=3)
```

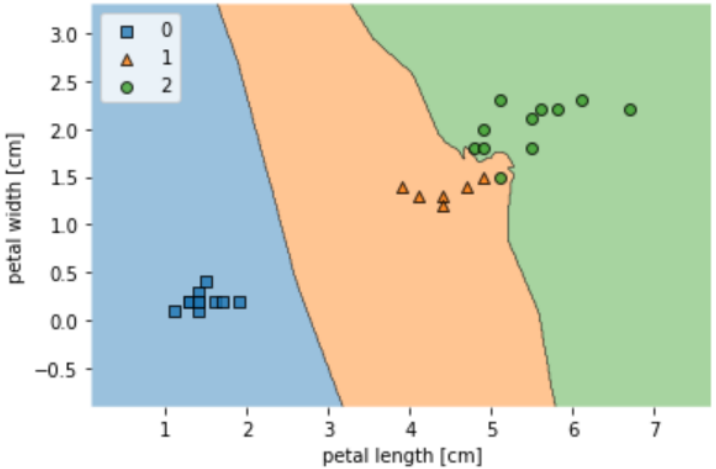
In [6]:
```
y_pred = knn_clf.predict(X_test)
```

In [7]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)*100
print(f'Test set accuracy: {accuracy:.2f}%')
```

```
Test set accuracy: 96.67%
```

Visualize Decision Boundary

In [10]:
```
# note: install "conda install -c conda-forge mlxtend"
from mlxtend.plotting import plot_decision_regions

plot_decision_regions(X_test, y_test, knn_clf)
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.show()
```



Nearest Neighbor

K-nearest Neighbor
In Scikit-Learn