ECE5984 – Applications of Machine Learning
Lecture 11 – Similarity Measures

Creed Jones, PhD

Course update

- HW3 is posted
  - Due Tuesday, March 1
- Project I has been posted
  - Due Tuesday, March 22
- Quiz 3 this Thursday
  - Same timeframes as Quiz 2
  - Lectures 8-11
- Spring break in two weeks
  - I won't have office hours during Spring break

Today's Objectives

- Boosting and Bagging

- Similarity-based learning
- Feature space
- Distance measures
  - Euclidean
  - Manhattan
  - Minkowski
  - Mahalanobis
- Nearest-neighbor classification

BOOSTING AND BAGGING

Boosting works by iteratively creating models and adding them to the ensemble

- The iteration stops when a predefined number of models have been added.
- When we use **boosting** each new model added to the ensemble is biased to pay more attention to instances that previous models miss-classified.
- This is done by incrementally adapting the dataset used to train the models. To do this we use a **weighted dataset**

Weighted Dataset

- Each instance has an associated weight $\mathbf{w}_i \geq 0$,
- Initially set to $\frac{1}{n}$ where n is the number of instances in the dataset.
- After each model is added to the ensemble it is tested on the training data and the weights of the instances the model gets correct are decreased and the weights of the instances the model gets incorrect are increased.
- These weights are used as a distribution over which the dataset is sampled to created a replicated training set, where the replication of an instance is proportional to its weight.

During each training iteration the algorithm:

1. Induces a model and calculates the total error ϵ by summing the weights of the training instances for which the predictions made by the model are incorrect.
2. Increases the weights for the instances misclassified using:

$$w[i] \leftarrow w[i]\left(\frac{1}{2\epsilon}\right) \tag{5}$$

3. Decreases the weights for the instances correctly classified:

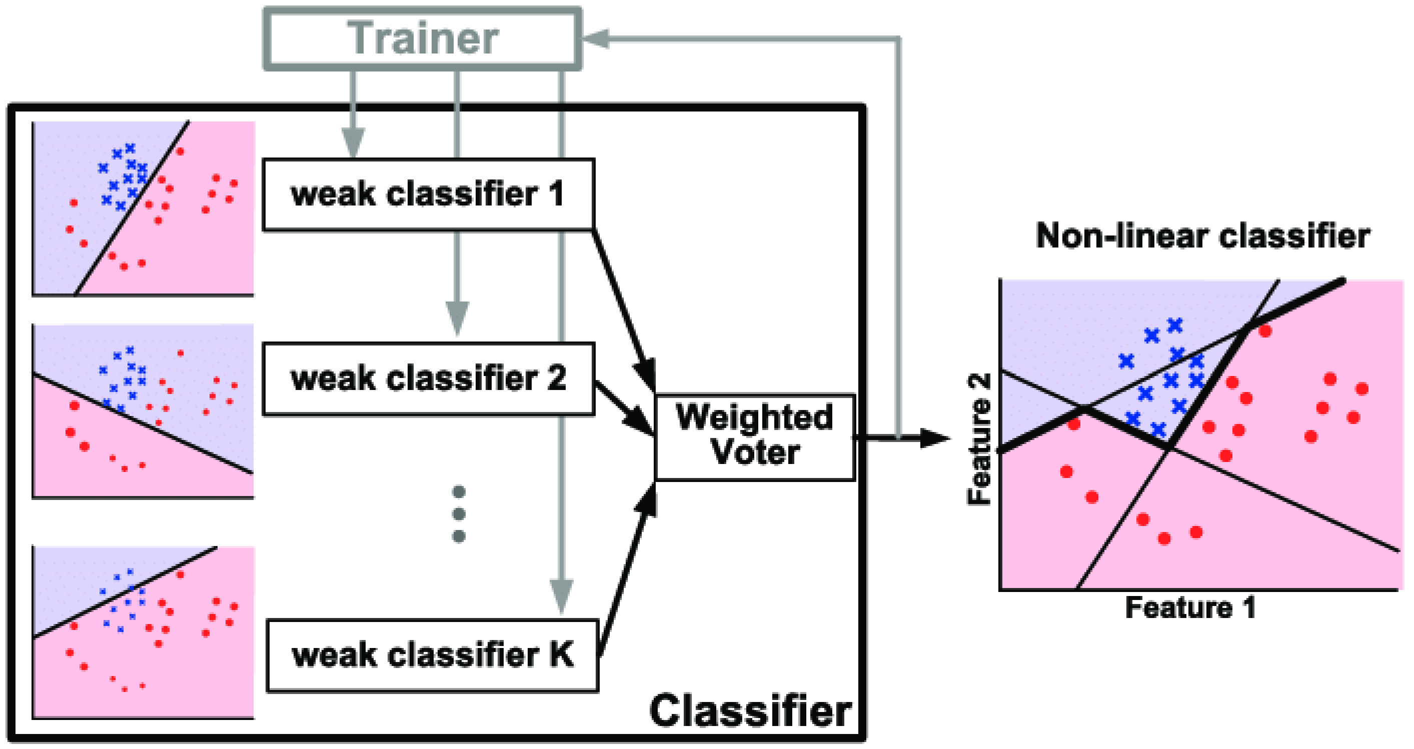
$$w[i] \leftarrow w[i]\left(\frac{1}{2(1-\epsilon)}\right) \tag{6}$$

4. Calculate a **confidence factor**, α, for the model such that

$$\alpha = \frac{1}{2} log_e\left(\frac{1-\epsilon}{\epsilon}\right) \tag{7}$$

- Once the set of models have been created the ensemble makes predictions using a weighted aggregate of the predictions made by the individual models.
- The weights used in this aggregation are simply the confidence factors associated with each model.

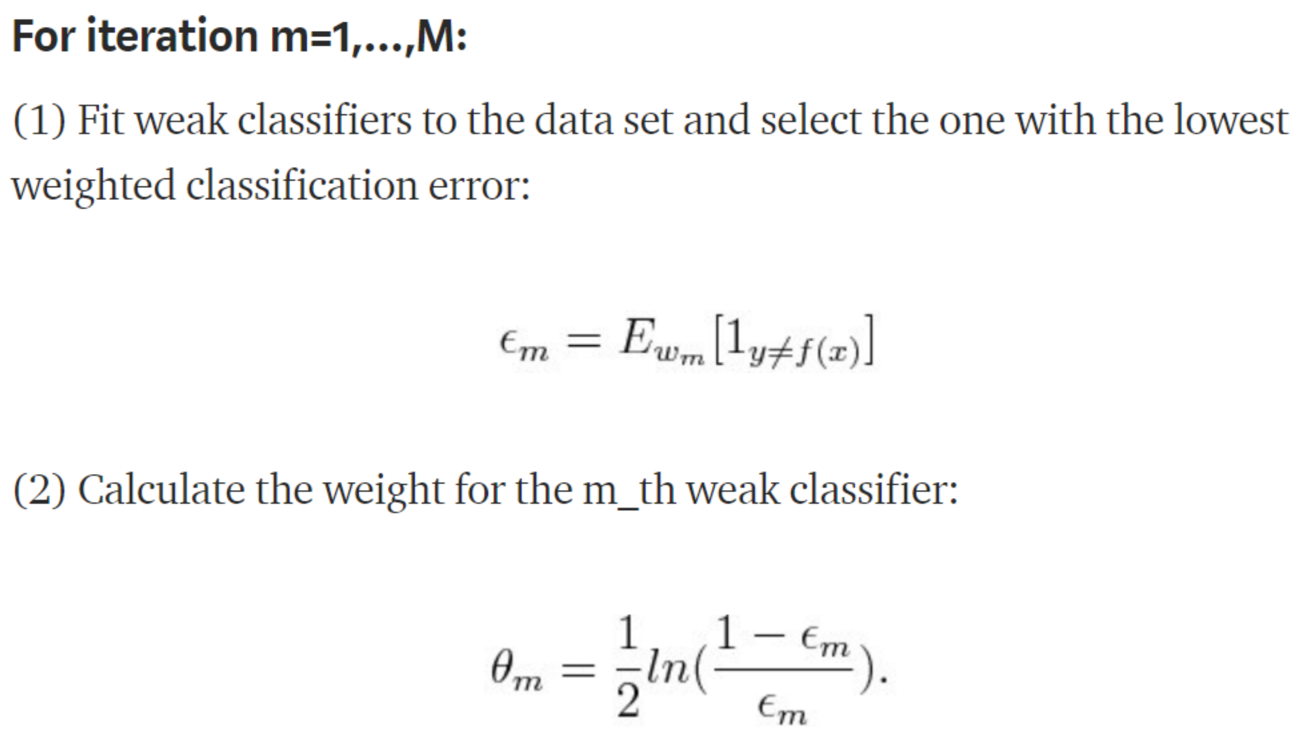
AdaBoost is a *meta-algorithm* for combining several "weak" classifiers into a more robust ensemble



- Outputs from several classifiers are weighted and summed
- The result does a better job of a complex task

- How are the classifiers weighted?
- Is there a difference in training?

In AdaBoost, each classifier is weighted according to its total error; also, during training, certain samples receive higher weights (“boost”) if the error is high



For iteration m=1,…,M:

(1) Fit weak classifiers to the data set and select the one with the lowest weighted classification error:

$$\epsilon_m = E_{w_m}[1_{y \neq f(x)}]$$

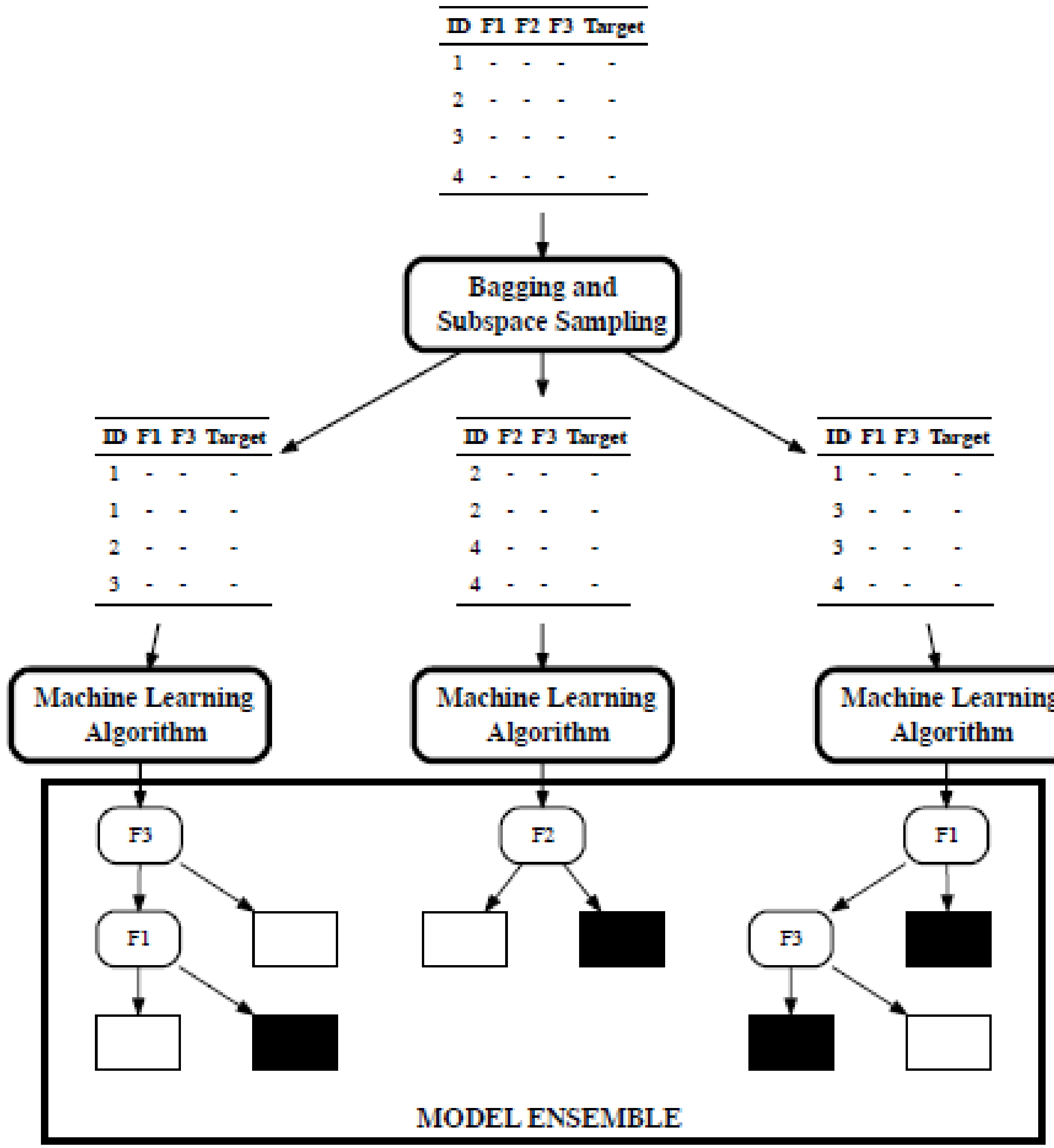
(2) Calculate the weight for the m_th weak classifier:

$$\theta_m = \frac{1}{2} ln(\frac{1-\epsilon_m}{\epsilon_m}).$$

Bagging results in a set of models trained on subsamples of the data

- When we use **bagging** (or **bootstrap aggregating**) each model in the ensemble is trained on a random sample of the dataset known as **bootstrap samples**.
- Each random sample is the same size as the dataset and **sampling with replacement** is used.
- Consequently, every bootstrap sample will be missing some of the instances from the dataset so each bootstrap sample will be different and this means that models trained on different bootstrap samples will also be different.
- When bagging is used with decision trees each bootstrap sample only uses a randomly selected subset of the descriptive features in the dataset. This is known as **subspace sampling**.
- The combination of bagging, subspace sampling, and decision trees is known as a **random forest** model.

The process of creating a model ensemble using bagging and subspace sampling



ID	F1	F2	F3	Target
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-

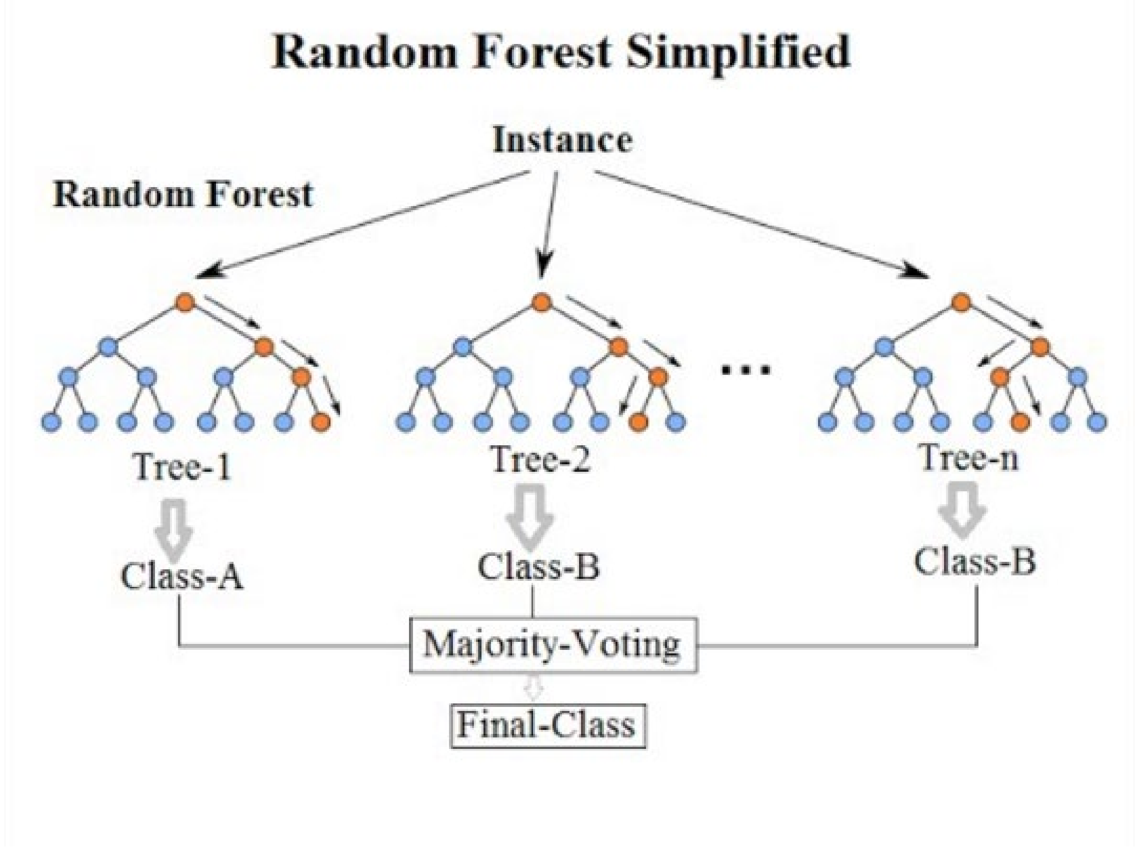
ID	F1	F3	Target
1	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-

ID	F2	F3	Target
2	-	-	-
2	-	-	-
4	-	-	-
4	-	-	-

ID	F1	F3	Target
1	-	-	-
3	-	-	-
3	-	-	-
4	-	-	-

Bagging and Subspace Sampling
Machine Learning Algorithm
Machine Learning Algorithm
Machine Learning Algorithm
F3
F1
F2
F1
F3
MODEL ENSEMBLE

In a Random Forest, many trees are trained on subsets of the data, produced by sampling with replacement, and results are combined



- "This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets."
  - Wikipedia

There are some heuristic guidelines for use of ensembles of trees

- Which approach should we use? Bagging is simpler to implement and parallelize than boosting and, so, may be better with respect to ease of use and training time.
- Empirical results indicate:
  - boosted decision tree ensembles were the best performing model of those tested for datasets containing up to 4,000 descriptive features.
  - Random forest ensembles (based on bagging) performed better for datasets containing more that 4,000 features.

SIMILARITY-BASED LEARNING

We often classify new objects or experiences by comparing to (labeled) things we already know about

- "*if you are trying to classify something then you should search your memory to find things that are similar and label it with the same class as the most similar thing in your memory.*"

- Most often, similarity is considered to be the inverse of distance between two things
  - "*if you are trying to classify something then you should search your memory to find things that are similar and label it with the same class as the <u>least different</u> thing in your memory.*"

- As we have seen, there are a number of good distance measures available

You come to my house for lunch and I serve you this – what is it?

	?
Bread	Y
Meat	N
Cheese	N
Lettuce	N
Tomato	Y
Cucumber	N
Mayo	Y
Meat Patty	N
Temperature	Cold

	Sandwich	Burger	Pizza	Salad
Bread	Y	Kinda	Kinda	N
Meat	Y	N	Maybe	Maybe
Cheese	Maybe	Maybe	Y	Maybe
Lettuce	Maybe	Maybe	N	Y
Tomato	Maybe	Maybe	Maybe	Maybe
Cucumber	N	N	N	Maybe
Mayo	Maybe	Maybe	N	N
Meat Patty	N	Y	N	N
Temperature	Cold	Hot	Hot	Cold

Compare the attributes of this new dish to the attributes of what I have already seen – note we are using +, -, ? and "Kinda"

	?
Bread	Y
Meat	N
Cheese	N
Lettuce	N
Tomato	Y
Cucumber	N
Mayo	Y
Meat Patty	N
Temperature	Cold

	Sandwich	Burger	Pizza	Salad
Bread	+	Kinda	Kinda	-
Meat	-	+	?	?
Cheese	?	?	-	?
Lettuce	?	?	+	-
Tomato	?	?	?	?
Cucumber	+	+	+	?
Mayo	?	?	-	-
Meat Patty	+	-	+	+
Temperature	+	-	-	+

The best match is "Sandwich" as it has the most "+" (4, compared to 2-3 for others) and fewest "-" (1, compared to 2-3): It's the most *similar*

	?
Bread	Y
Meat	N
Cheese	N
Lettuce	N
Tomato	Y
Cucumber	N
Mayo	Y
Meat Patty	N
Temperature	Cold

	Sandwich	Burger	Pizza	Salad
Bread	+	Kinda	Kinda	-
Meat	-	+	?	?
Cheese	?	?	-	?
Lettuce	?	?	+	-
Tomato	?	?	?	?
Cucumber	+	+	+	?
Mayo	?	?	-	-
Meat Patty	+	-	+	+
Temperature	+	-	-	+

Considering each variable as a dimension, we can think of each sample as a location in an *N*-dimensional space called *feature space*

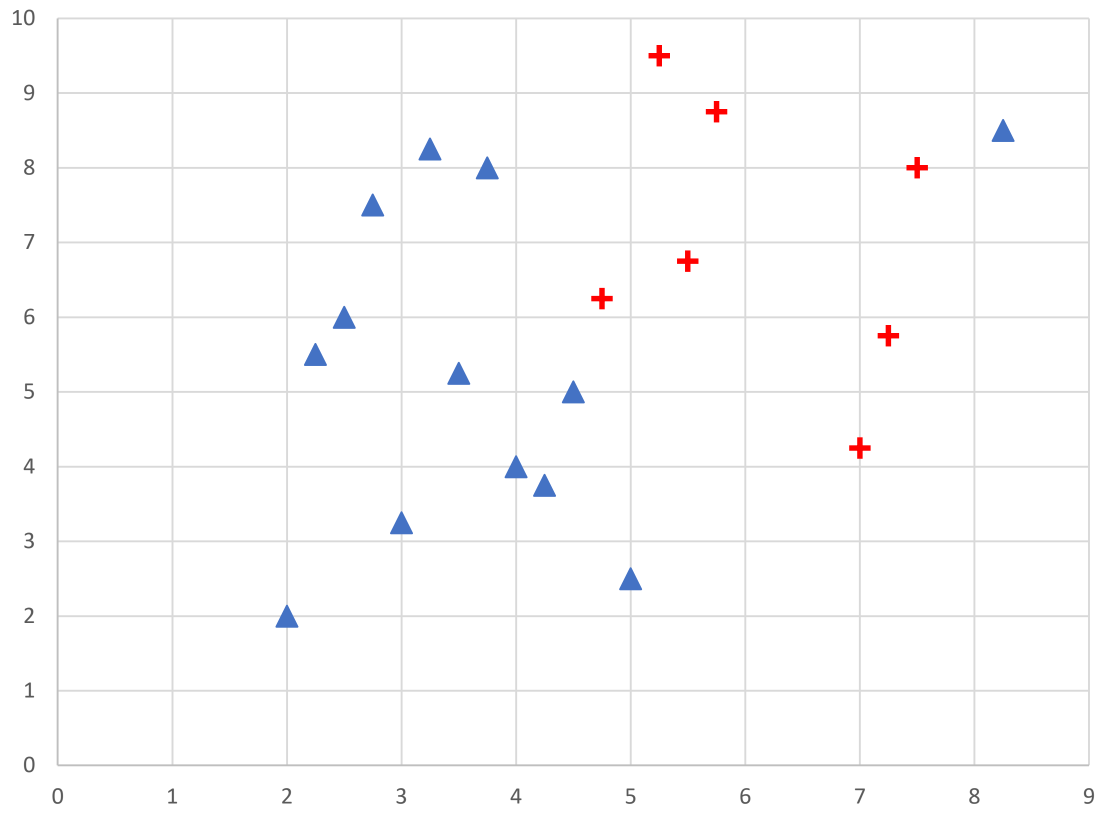
- A feature space is an abstract n-dimensional space that is created by taking each of the descriptive features in an ADT to be the axes of a reference space and each instance in the dataset is mapped to a point in the feature space based on the values of its descriptive features.

- While we will often show examples in 2D or 3D feature spaces – because they are easier to draw and to understand, most useful feature spaces have much higher dimensionality
  - *N* of several thousands is not uncommon
  - We need to ensure that any techniques that we develop are applicable to feature spaces of arbitrary dimension

Feature-space representation of a simple data set. The X and Y axes are feature variables, while the binary target is represented by the color and symbol.

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

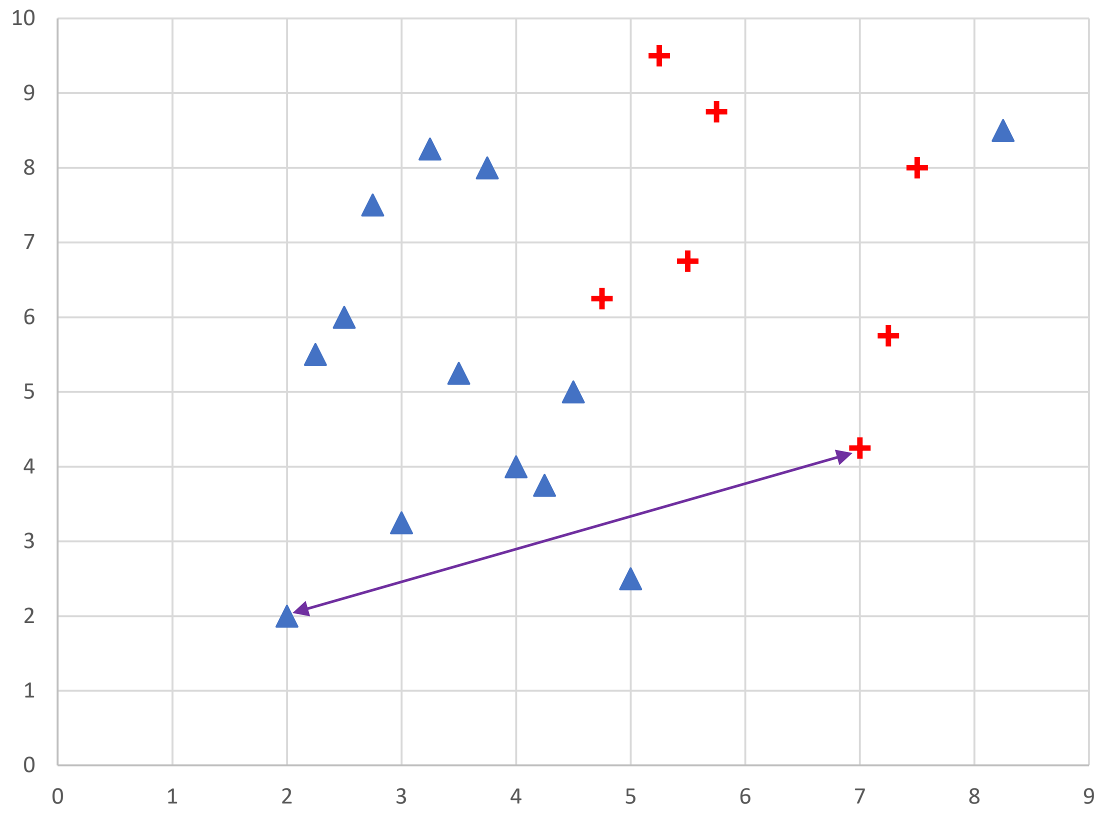


A distance metric measures the distance between two instances according to a feature space

- Example, using *Euclidean Distance,*

$$d = \sqrt{\sum (s_1 - s_2)^2 + (a_1 - a_2)^2}$$

$$= \sqrt{\sum (2-7)^2 + (2-4.25)^2} = 5.482$$



Distance measures

Mathematically, a distance measure (or *metric*) *m* must conform to the following four criteria, where $m(a, b)$ is a function that returns the distance between two instances a and b:

1. Non-negativity: $m(a, b) \geq 0$
2. Identity: $m(a, b) = 0 \iff a = b$
3. Symmetry: $m(a, b) = m(b, a)$
4. Triangular Inequality: $m(a, b) \leq m(a, c) + m(b, c)$

A very common distance measure is the Euclidean norm of the vector difference

- Remember that the Euclidean or L^2 norm is:

$$L^2 = \sqrt{\sum_{i=1}^{k} v_i^2}$$

- Applying this to the difference of two vector samples gives the Euclidean distance:

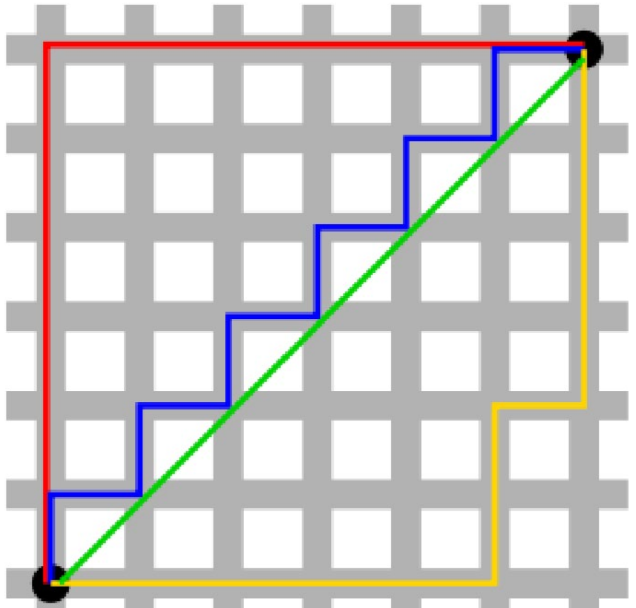
$$Euc(a,b) = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2}$$

A distance measure often used in discrete systems is the *Manhattan* or *city-block* distance

- The Manhattan distance is the sum of the magnitudes of the component-by-component differences:

$$Manh(a,b) = \sum_{i=1}^{k} |a_i - b_i|$$

- It's a measure of how many city blocks one would have to move to travel from one vector location to the other…



The Minkowski distance is a general equation, of which Euclidean and Manhattan are special cases

- The Minkowski distance allows us to specify the power used to compute the individual components' contributions:

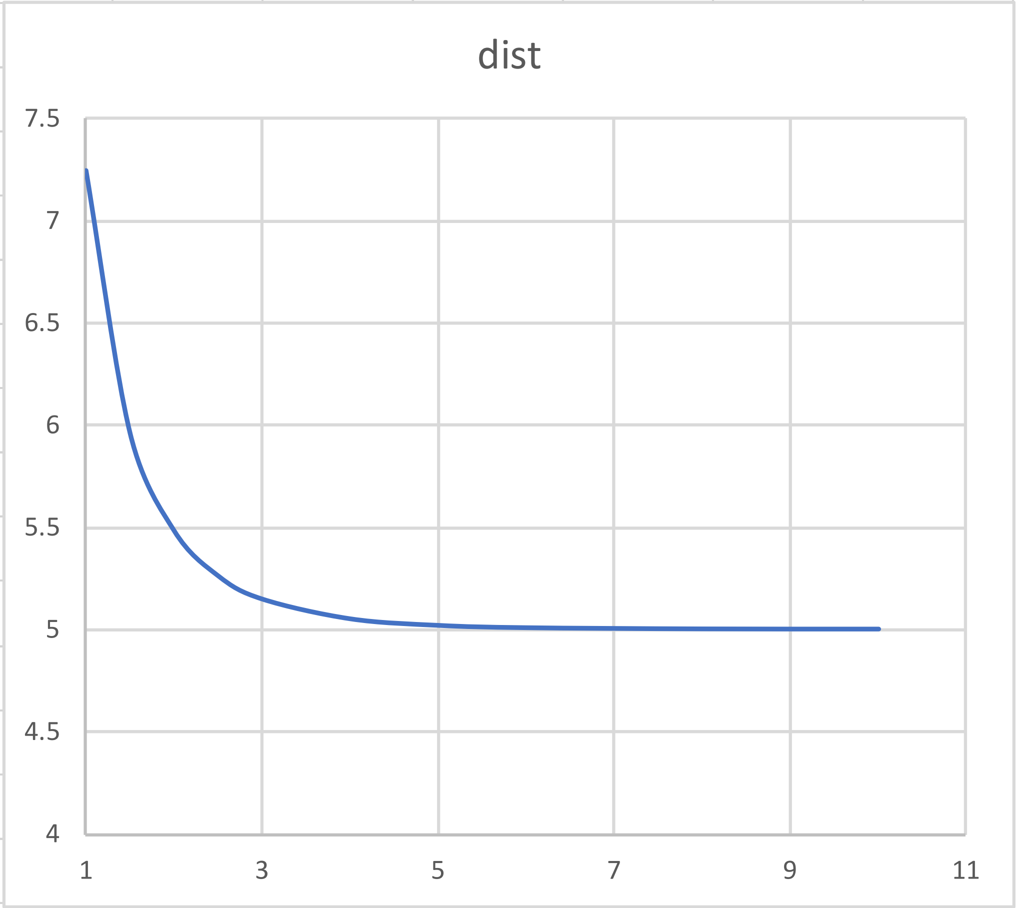
$$Mink_p(a,b) = \left(\sum_{i=1}^{k} |a_i - b_i|^p\right)^{\frac{1}{p}}$$

- $Mink_1(a,b)$ is the Manhattan distance
- $Mink_2(a,b)$ is the Euclidean distance

- "Larger values of p place more emphasis on large differences between feature values than smaller values of p… Consequently, the Euclidean distance (with p = 2) is more strongly influenced by a single large difference in one feature than the Manhattan distance (with p = 1)"

As p increases, the Minkowski distance approaches the L^{∞} norm of the distance vector

Pt1	2	2
Pt2	7	4.25
p	dist	dist
1	7.25	7.25
1.5	14.55534	5.9614
2	30.0625	5.482928
2.5	63.49545	5.261348
3	136.3906	5.147482
4	650.6289	5.050488
5	3182.665	5.018318
6	15754.75	5.006896
8	391281.8	5.00105
10	9768950	5.00017



The *Mahalanobis* distance is defined between a vector and a collection (or distribution) of vectors

$$Maha(\boldsymbol{z}, \boldsymbol{X}) = \sqrt{(\boldsymbol{z} - \boldsymbol{\mu})^T S^{-1} (\boldsymbol{z} - \boldsymbol{\mu})}$$

where the collection $\boldsymbol{X}$ has mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{S}$

The Mahalanobis distance can:

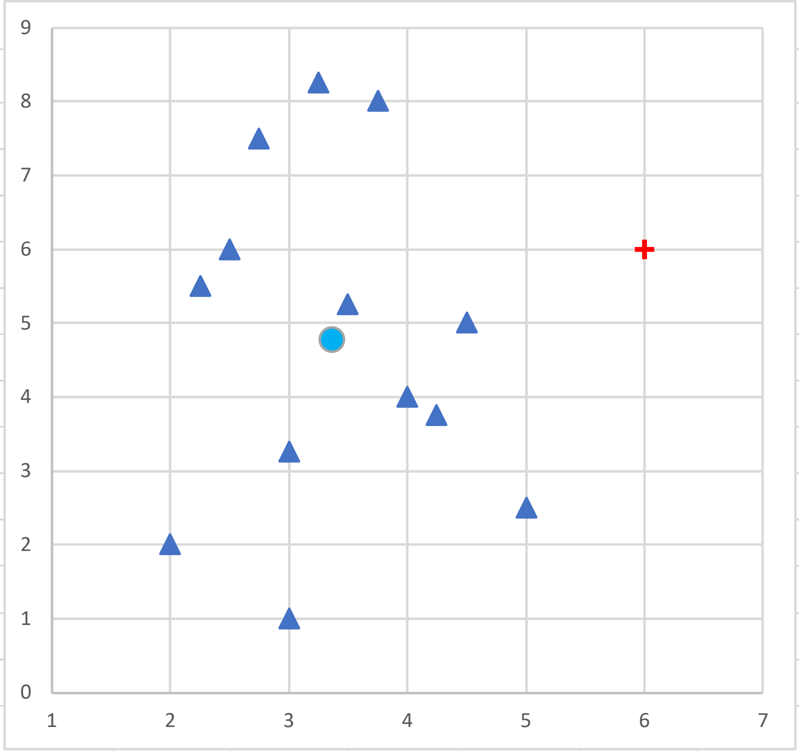
- measure distance relative to the observed variances of the data
- allow distance measurement to be insensitive to varying scales of the axes

We often use the Mahalanobis distance to measure difference between two vectors <u>where we know the distribution (or a sample population) for one of them</u>

- For example, when comparing an unlabeled sample to a collection of samples of one label, we may compare to the mean (or median?) vector of the collection

$$Maha(\boldsymbol{z},\boldsymbol{X}) = \sqrt{(\boldsymbol{z}-\boldsymbol{\mu})^T S^{-1} (\boldsymbol{z}-\boldsymbol{\mu})}$$

ID	X	Y
1	2.5	6
2	3.75	8
3	2.25	5.5
4	3.25	8.25
5	2.75	7.5
6	4.5	5
7	3.5	5.25
8	3	3.25
9	4	4
10	4.25	3.75
11	2	2
12	5	2.5
13	3	1
NEW	6	6



x-μx	y-μy
-0.865	1.231
0.385	3.231
-1.115	0.731
-0.115	3.481
-0.615	2.731
1.135	0.231
0.135	0.481
-0.365	-1.519
0.635	-0.769
0.885	-1.019
-1.365	-2.769
1.635	-2.269
-0.365	-3.769
0.000	0.000

	X	Y
μ=	3.365	4.769

$$S = \begin{bmatrix} 9.8269 & -1.7788 \\ -1.7788 & 63.3077 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} 0.1023 & 0.0029 \\ 0.0029 & 0.0159 \end{bmatrix}$$

$$\text{Maha} = \begin{bmatrix} 2.6346 & 1.2308 \end{bmatrix} * \begin{bmatrix} 0.1023 & 0.0029 \\ 0.0029 & 0.0159 \end{bmatrix} * \begin{bmatrix} 2.6346 \\ 1.2308 \end{bmatrix} = 0.8676$$

$$\begin{bmatrix} 0.2730 & 0.0271 \end{bmatrix} * \begin{bmatrix} 2.6346 \\ 1.2308 \end{bmatrix}$$

We can use the inverse of the distance between two vectors as a representation of their *similarity* or closeness in the feature space

- $Euc(a,b) = \sqrt{\sum_{i=1}^{k}(a_i - b_i)^2}$ $\quad SimEuc(a,b) = \frac{1}{\sqrt{\sum_{i=1}^{k}(a_i - b_i)^2}}$
- $Manh(a,b) = \sum_{i=1}^{k}|a_i - b_i|$ $\quad SimManh(a,b) = \frac{1}{\sqrt{\sum_{i=1}^{k}|a_i - b_i|}}$
- $Mink_p(a,b) = \left(\sum_{i=1}^{k}|a_i - b_i|^p\right)^{\frac{1}{p}}$ $\quad SimMink_p(a,b) = \frac{1}{\left(\sum_{i=1}^{k}|a_i - b_i|^p\right)^{\frac{1}{p}}}$
- $Maha(\boldsymbol{z},\boldsymbol{X}) = \sqrt{(\boldsymbol{z}-\boldsymbol{\mu})^T S^{-1}(\boldsymbol{z}-\boldsymbol{\mu})}$ $\quad SimMaha(\boldsymbol{z},\boldsymbol{X}) = \frac{1}{\sqrt{(\boldsymbol{z}-\boldsymbol{\mu})^T S^{-1}(\boldsymbol{z}-\boldsymbol{\mu})}}$

Distance measurement for mixed-class variable sets is usually done as if they were all scalars

- Real variables – distance calculation is natural
- Whole number variables (discrete case) – also natural
- Binary variables – only 0 and 1, the distance math works well
- Categorical variables – a problem. We <u>must</u> either have, or impose, a sensible order on the categoricals
  - {Small, Medium, Large}, Small is closer to Medium than it is to Large
  - Sometimes we assign inequally spaced numbers to the categories
    - Small = 8, Medium = 12, Large = 24
  - If we <u>cannot</u> impose a sensible order, then we must use one-hot encoding
    - With grouping for "Others" if necessary
- As an interesting aside, these distance metrics work fine for vectors of complex components

How do I do distance calculations when the examples contain ambiguous values like "Either"? Generally, include two prototypes, one with each possible value

	?
Bread	Y
Meat	N
Cheese	N
Lettuce	N
Tomato	Y
Cucumber	N
Mayo	Y
Meat Patty	N
Temperature	Cold

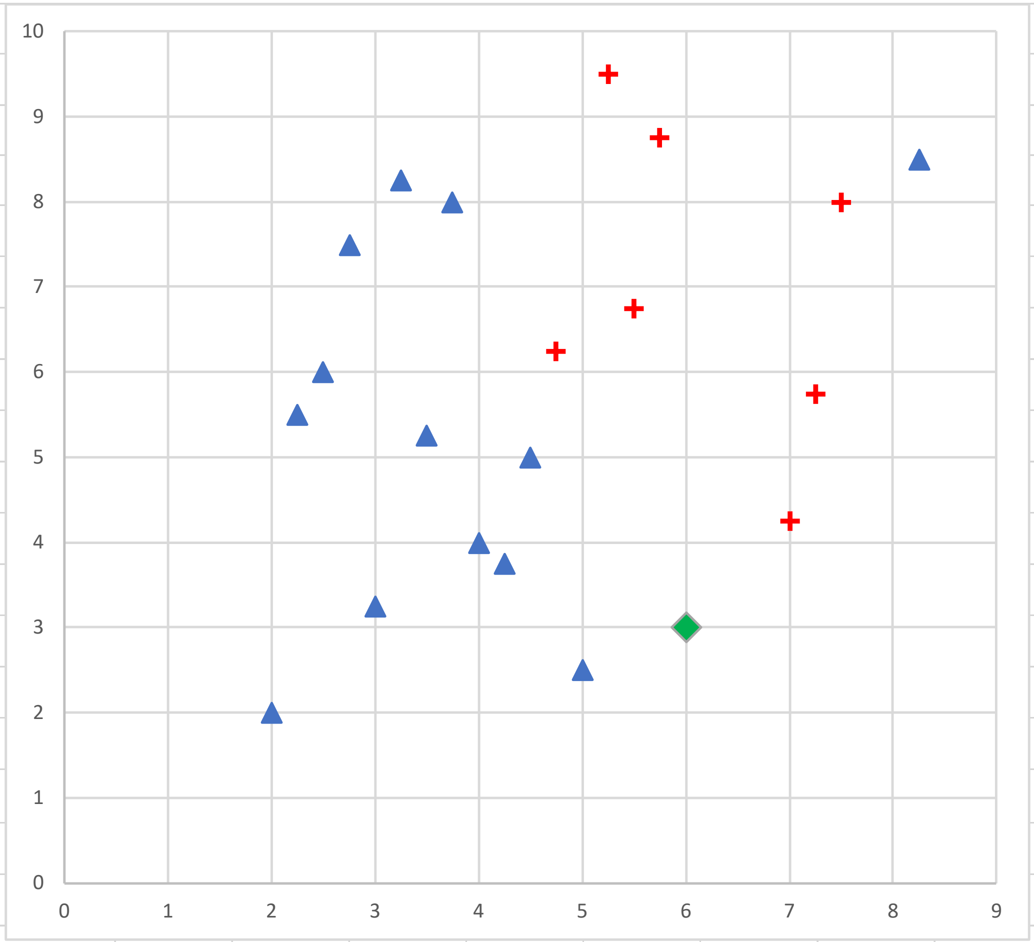
	Sandwich	Burger	Pizza	Salad
Bread	+	Kinda	Kinda	-
Meat	-	+	?	?
Cheese	?	?	-	?
Lettuce	?	?	+	-
Tomato	?	?	?	?
Cucumber	+	+	+	?
Mayo	?	?	-	-
Meat Patty	+	-	+	+
Temperature	+	-	-	+

NEAREST-NEIGHBOR CLASSIFICATION

We now have a set of distance measures that can be calculated on a pair of vectors in N-space; we can use distance-based methods to assign vectors to classes

- This is cluster-based classification
  - Assigning new vectors to existing classes defined by clusters of samples
- The simplest technique is the nearest-neighbor method
- Given a set of training instances and a new vector to be classified:
  - Iterate across the instances in memory and find the instance that is the shortest distance from the new vector in the feature space.
  - The vector is assigned the target value of the nearest neighbor.

ID	Speed	Agility	Draft
1	2.5	6	No
2	3.75	8	No
3	2.25	5.5	No
4	3.25	8.25	No
5	2.75	7.5	No
6	4.5	5	No
7	3.5	5.25	No
8	3	3.25	No
9	4	4	No
10	4.25	3.75	No
11	2	2	No
12	5	2.5	No
13	8.25	8.5	No
14	5.75	8.75	Yes
15	4.75	6.25	Yes
16	5.5	6.75	Yes
17	5.25	9.5	Yes
18	7	4.25	Yes
19	7.5	8	Yes
20	7.25	5.75	Yes



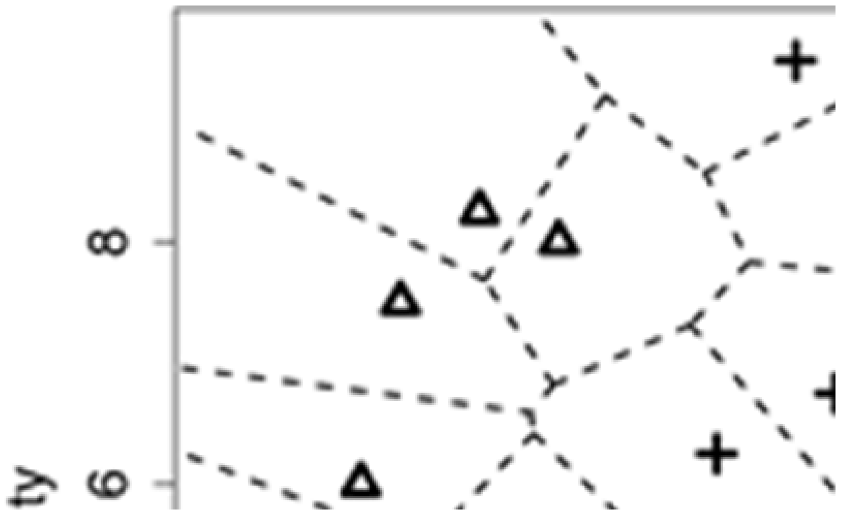
Speed	Agility	Draft
6	3	???

ID	Dist	Draft	
1	4.610	No	
2	5.483	No	
3	4.507	No	
4	5.927	No	
5	5.551	No	
6	2.500	No	
7	3.363	No	
8	3.010	No	
9	2.236	No	
10	1.904	No	
11	4.123	No	
12	**1.118**	**No**	←
13	5.942	No	
14	5.755	Yes	
15	3.482	Yes	
16	3.783	Yes	
17	6.543	Yes	
18	1.601	Yes	
19	5.220	Yes	
20	3.021	Yes	

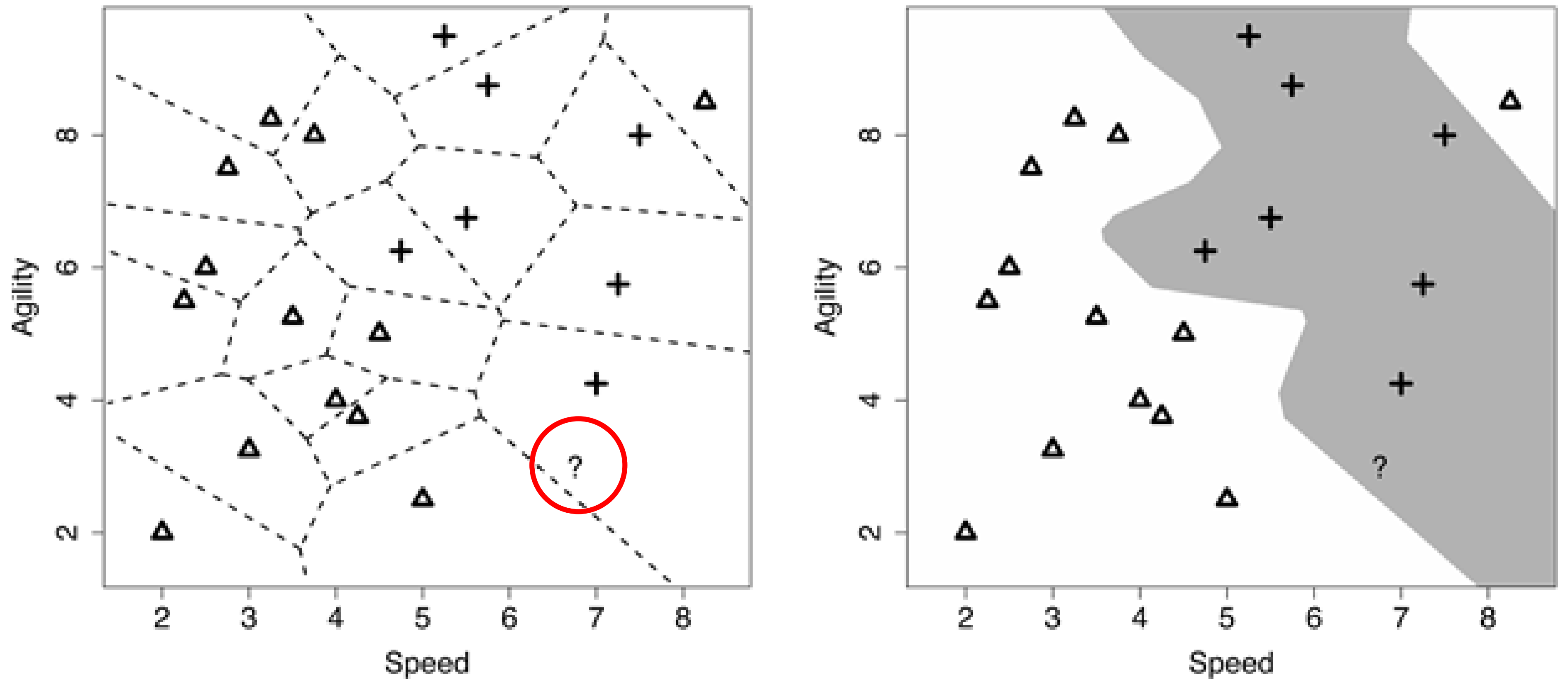
Voronoi tessellation is the division of feature space into regions defined by which existing sample is closest to the point in space

- The Voronoi regions cut space up into areas that share the same nearest neighbor
- New vectors in a given Voronoi region will receive the target value of the neighbor defining that region
- The boundaries form a *Voronoi diagram*

- Voronoi tessellation of hyperspace is one of the fundamental problems in the theoretical study of algorithms and computing

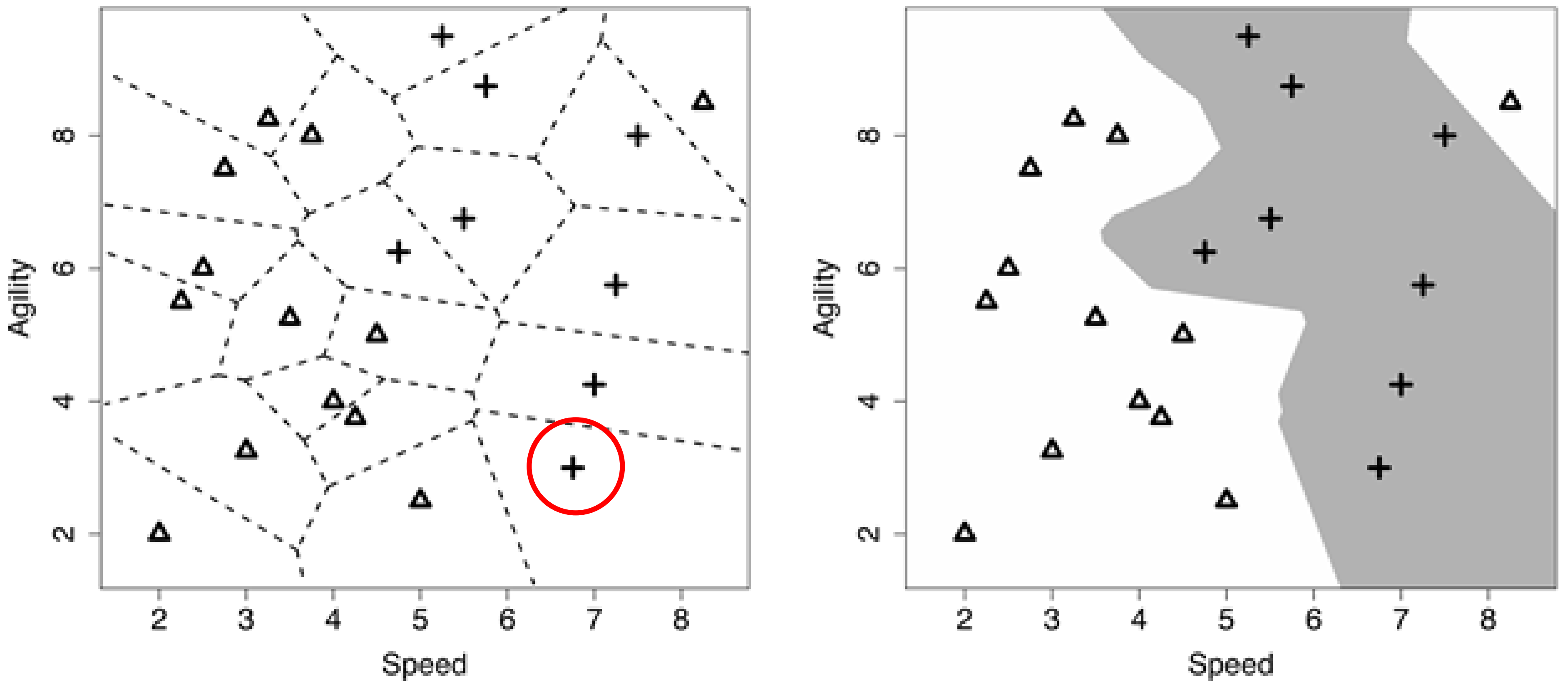


"hyperspace" is the term I use for n-dimensional space, especially where *n* is large



(a) Voronoi tessellation (b) Decision boundary (k = 1)

Figure: (a) The Voronoi tessellation of the feature space for the dataset in Table 2 [25] with the position of the query represented by the ? marker; (b) the decision boundary created by aggregating the neighboring Voronoi regions that belong to the same target level.

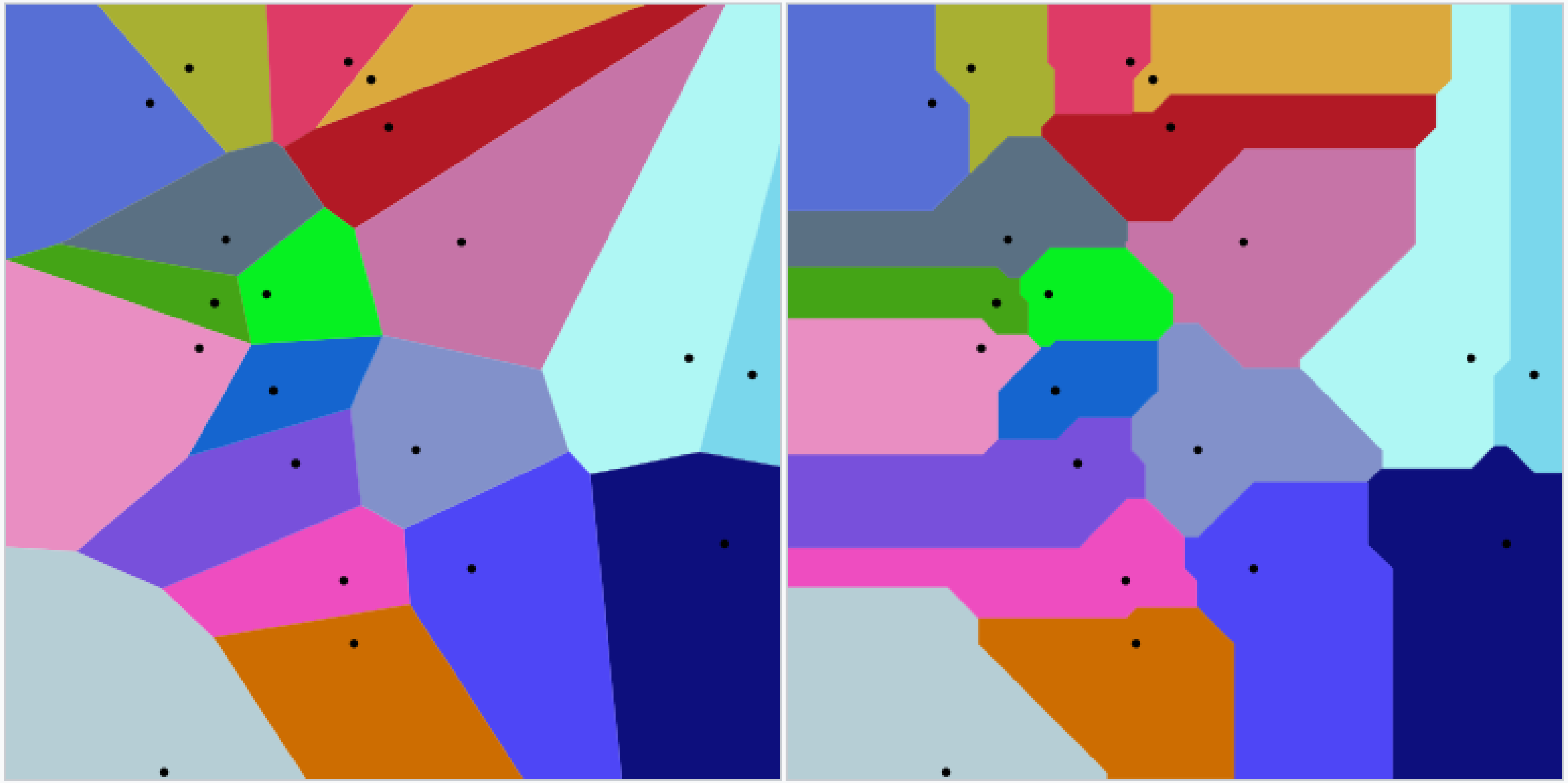


(a) Voronoi tessellation (b) Decision boundary ($k = 1$)

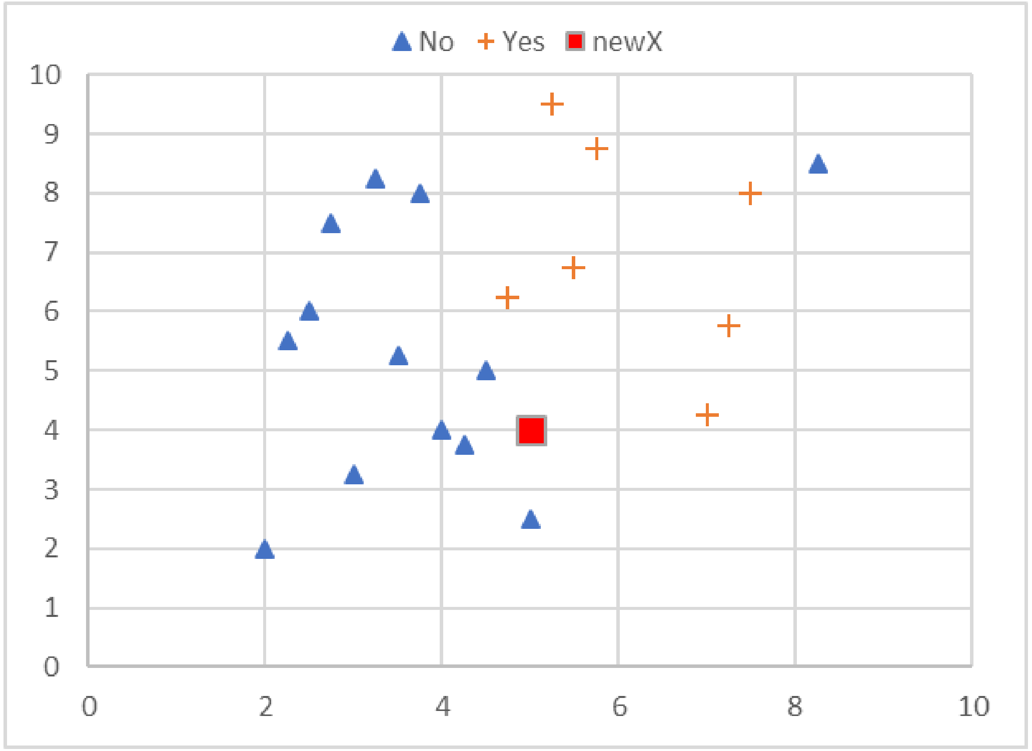
Figure: (a) The Voronoi tessellation of the feature space when the dataset has been updated to include the query instance; (b) the updated decision boundary reflecting the addition of the query instance in the training set.

The distance metric being used <u>does</u> matter…

Voronoi diagrams of 20 points under two different metrics



```
# -*- coding: utf-8 -*-
"""
nearestNeighbor.py
Created on Thu Feb 20 17:41:33 2020
@author: crjones4
"""

from sklearn import tree
import pandas as pd
import numpy.linalg as linalg

def distance(a, b):
    # Euclidean distance – norm of the difference vector
    return linalg.norm(a-b)

pathName = "C:\\Data\\"
dataFrame = pd.read_excel(pathName + 'DraftData.xlsx',
sheet_name='rawData')

X = dataFrame.drop(["ID", "Draft"], axis=1)
y = dataFrame.Draft
newX = [5, 4]

minDist = 100000
count = 0
for row in X.iterrows():        # NOTE! slow & only for use on small ADS
    samp = row[1]               # pull data point from list
    dist = distance(newX, samp)
    if (dist < minDist):        # find smallest distance
        minDist = dist
        minRow = row
        minTarget = y[count]
    count = count + 1
print("Min at", minRow)
print(minDist, minTarget)       # identify with most similar point
# could add newX to data here, if desired
```



```
('Min at', (9L, Speed 4.25  Agility 3.75
Name: 9, dtype: float64))
(0.7905694150420949, u'No')
```

Today's Objectives

- Boosting and Bagging

- Similarity-based learning
- Feature space
- Distance measures
  - Euclidean
  - Manhattan
  - Minkowski
  - Mahalanobis
- Nearest-neighbor classification