Lazy Learning

Stephen G. Ware

CSCI 4525 / 5525

Eager vs. Lazy Learning

The machine learning techniques we have considered so far are called **eager learners** because they build a model of the function being learned. Training is expensive, but classification (applying the model) is cheap.

Lazy learners, on the other hand, do not build a model. Computation is deferred until classification, (making classification more expensive).

Citrus vs. Melon

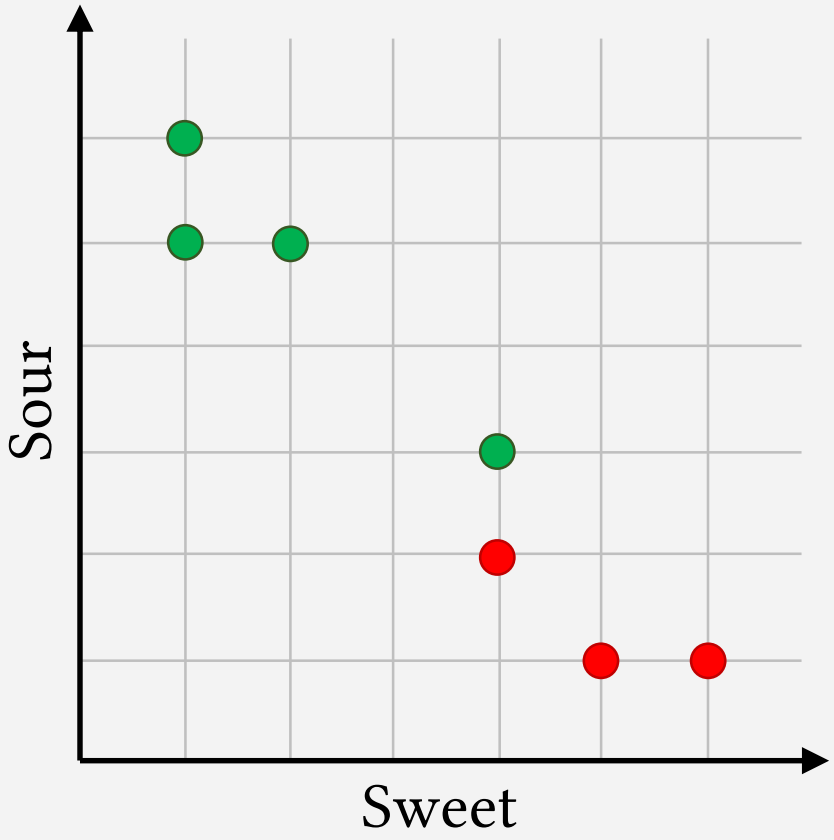
Say you have a number of judges taste different kinds of fruits (citrus and melons) and agree on values for how sweet and how sour they are.

Given an unknown fruit, your task is to classify it as citrus or melon based on its taste.

Fruit Data Set

Fruit	Sweet	Sour	Type
Lemon	1	5	Citrus
Lime	1	6	Citrus
Grapefruit	2	5	Citrus
Cantaloupe	4	2	Melon
Orange	4	3	Citrus
Honeydew	5	1	Melon
Watermelon	6	1	Melon

Fruit Data Set



Fruit	Sweet	Sour
Lemon	1	5
Lime	1	6
Grapefruit	2	5
Cantaloupe	4	2
Orange	4	3
Honeydew	5	1
Watermelon	6	1

k Nearest Neighbor

The **k-nearest-neighbor** (or KNN) method works by simply classifying unknown observations based on a majority vote of the k examples that are closest to it in metric space.

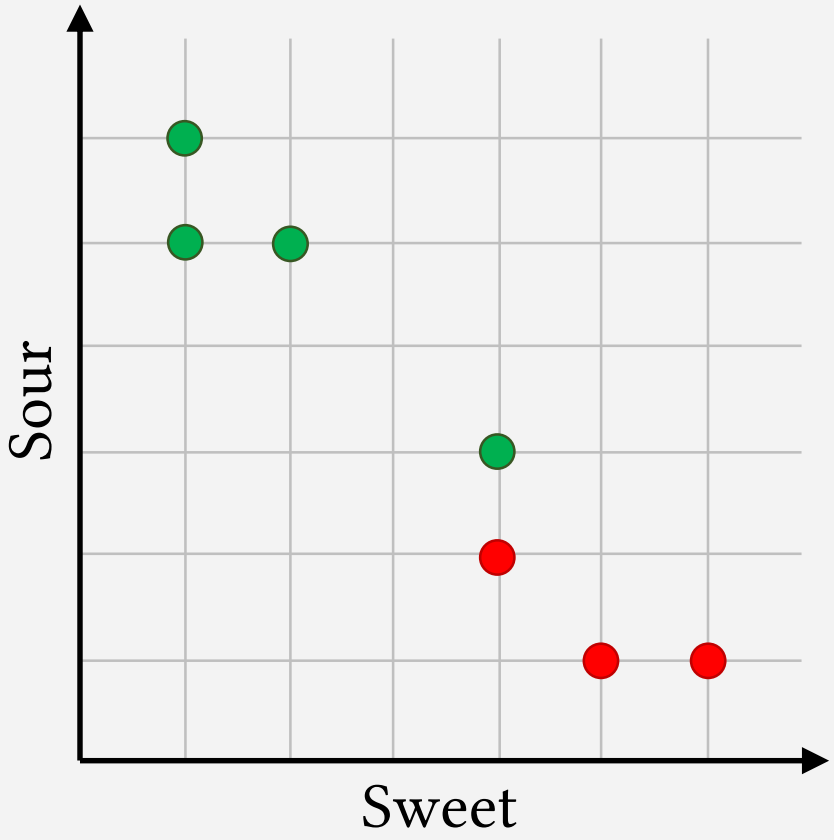
The number k is chosen arbitrarily, and is generally an odd number to avoid a tie in the votes.

k Nearest Neighbor

When performing KNN, we need to know the following information:

- How many dimensions n does the data have? (i.e. the number of attributes.)
- How many votes k will be taken?
- Which distance function will be used?

Fruit Data Set

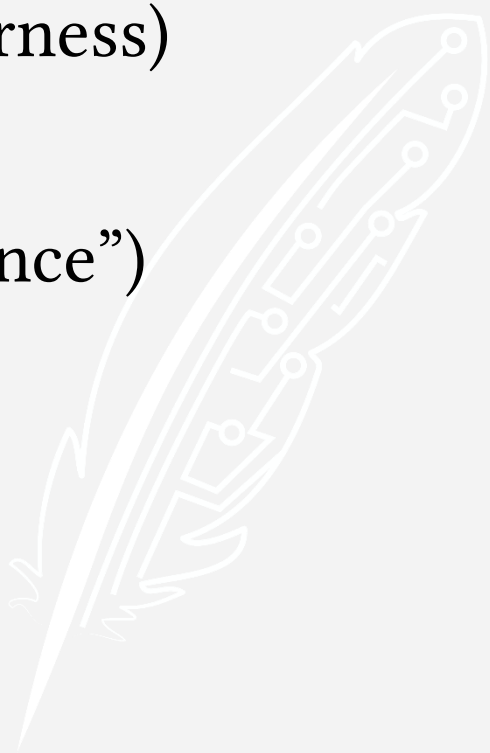


Fruit	Sweet	Sour
Lemon	1	5
Lime	1	6
Grapefruit	2	5
Cantaloupe	4	2
Orange	4	3
Honeydew	5	1
Watermelon	6	1

KNN Example

For the fruit classification example:

- Dimensions: $n = 2$ (sweetness and sourness)
- Votes: $k = 3$ (chosen arbitrarily)
- Distance: Euclidian ("straight line distance")



Euclidian Distance

Given two n-dimensional vectors x and y, the **Euclidian distance** (or straight line distance) is:

$$x = [x_1, x_2, \ldots, x_n] \quad \text{and} \quad y = [y_1, y_2, \ldots, y_n]$$

$$d(x, y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \cdots + (y_n - x_n)^2}$$

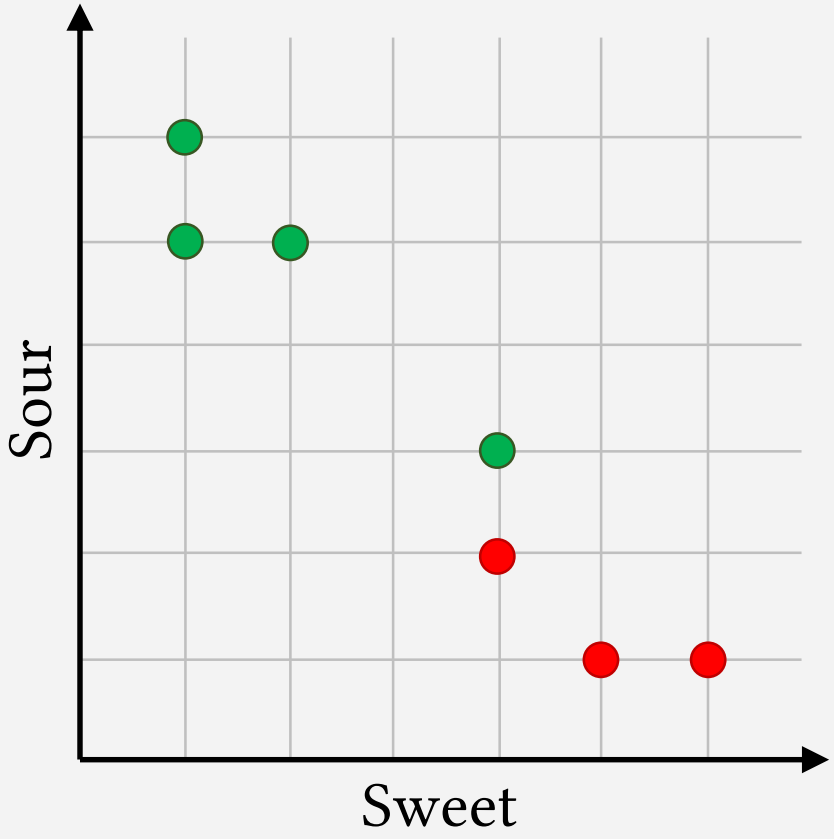
Euclidian Distance

Given two n-dimensional vectors x and y, the **Euclidian distance** (or straight line distance) is:

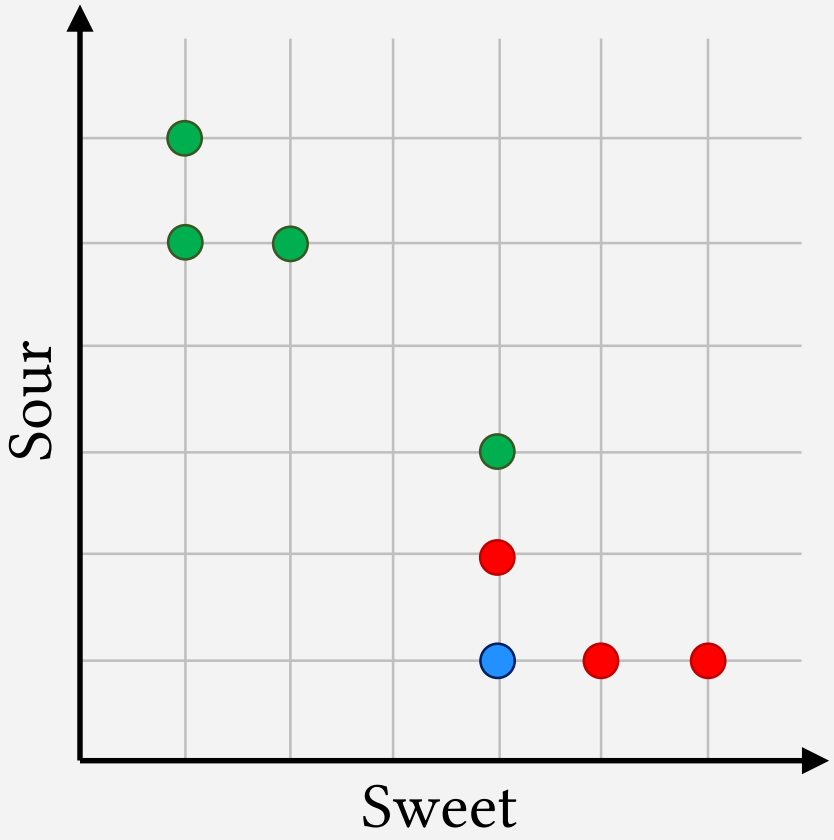
$$x = [x_1, x_2, \ldots, x_n] \quad \text{and} \quad y = [y_1, y_2, \ldots, y_n]$$

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

KNN Example



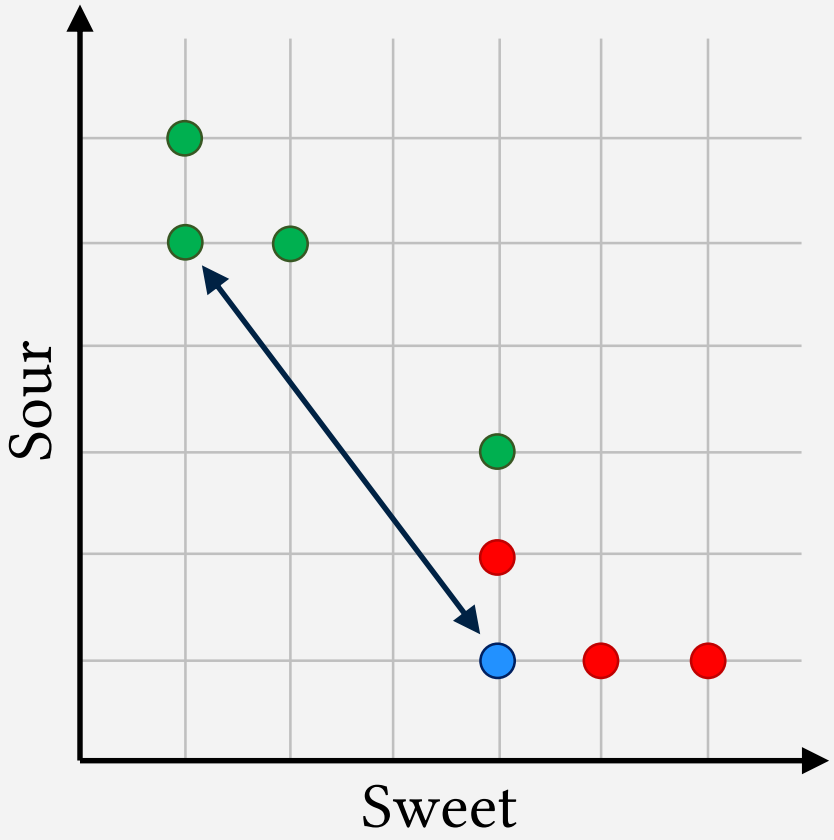
KNN Example



Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

KNN Example

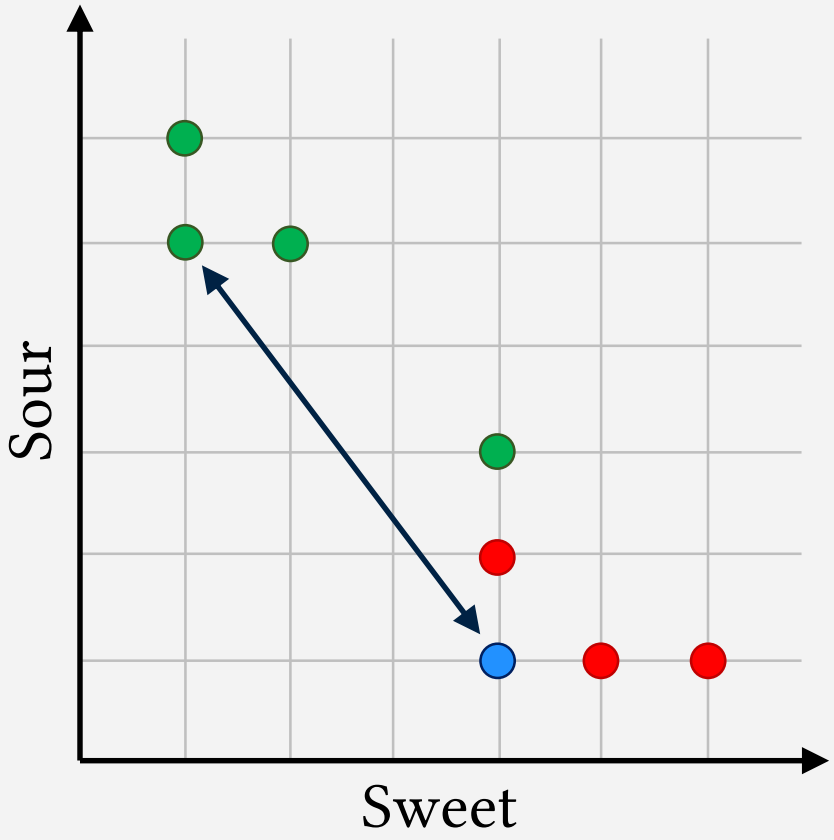


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$d([1,5],[4,1]) = ?$

KNN Example

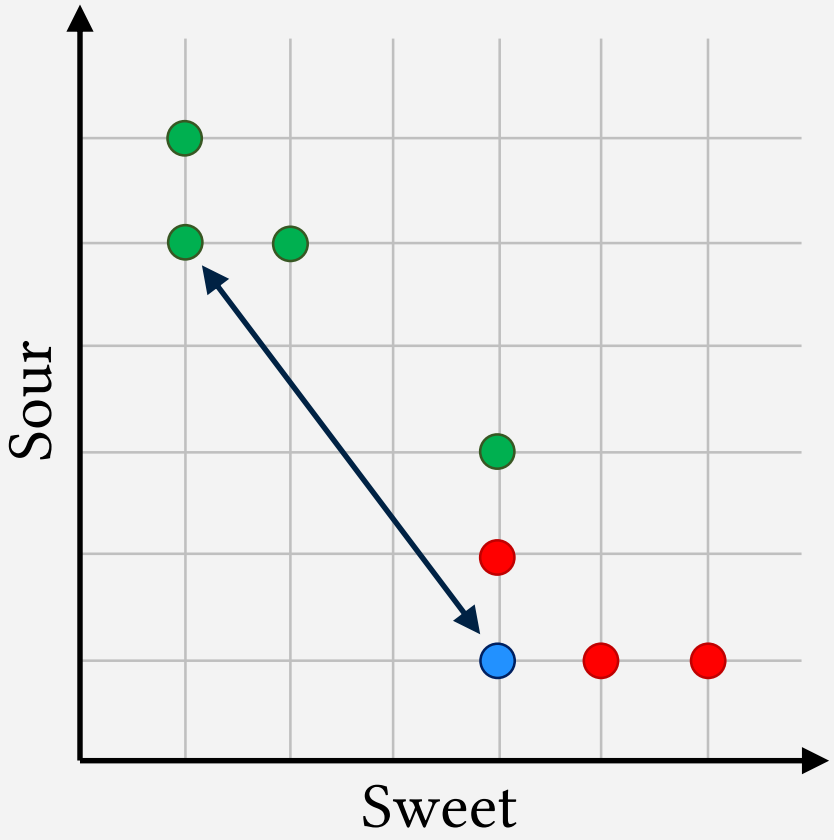


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$$d([1,5],[4,1]) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}$$

KNN Example

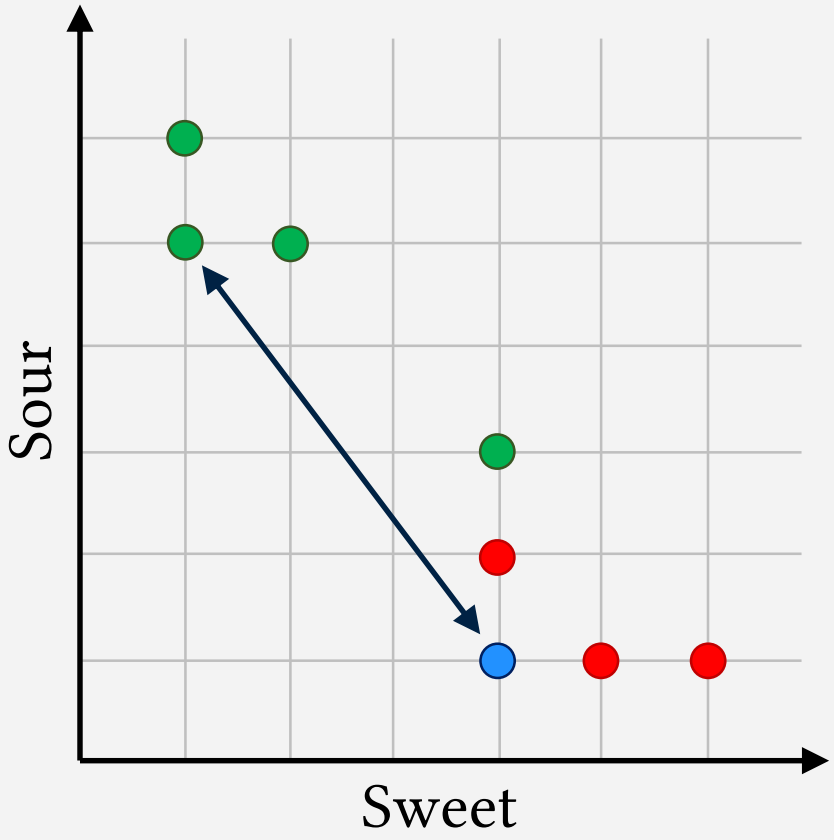


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$$d([1,5],[4,1]) = \sqrt{(4-1)^2 + (1-5)^2}$$

KNN Example

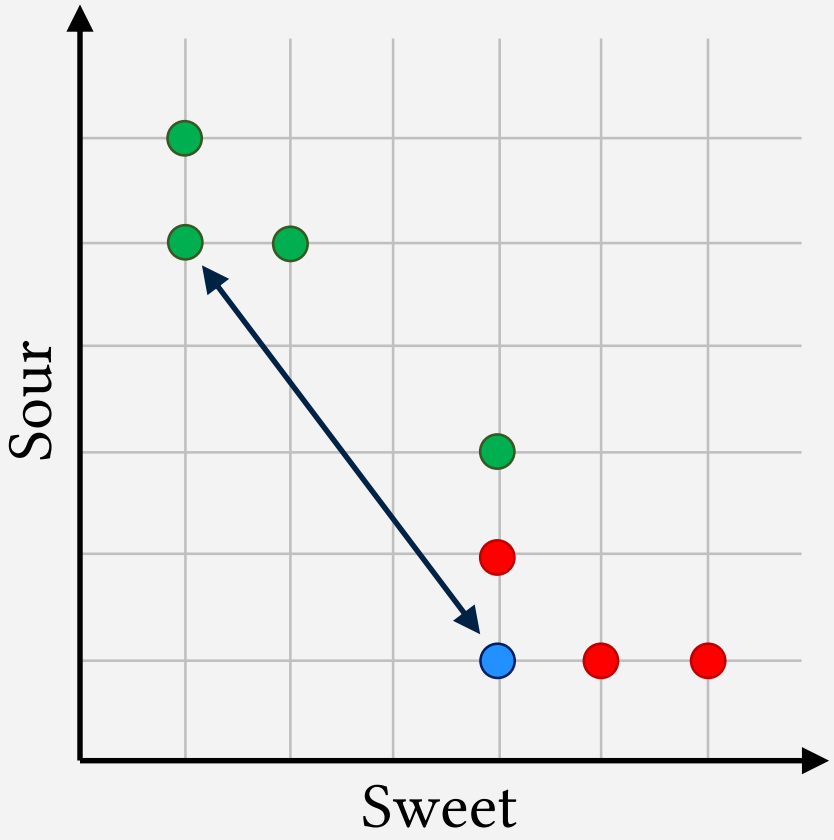


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$$d([1,5],[4,1]) = \sqrt{3^2 + -4^2}$$

KNN Example

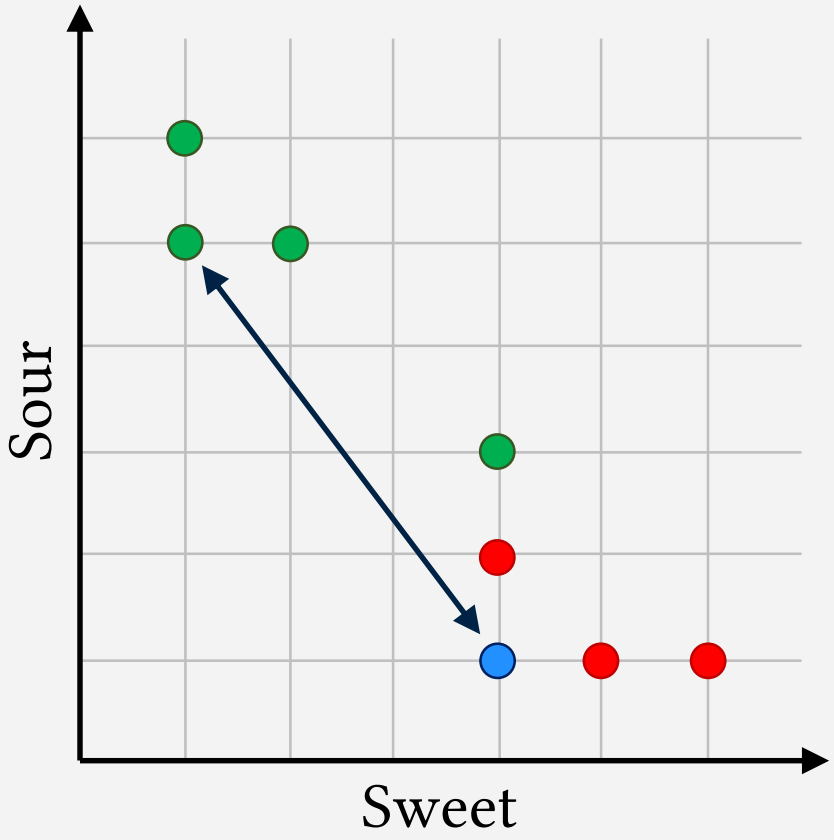


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$$d([1,5],[4,1]) = \sqrt{9+16}$$

KNN Example

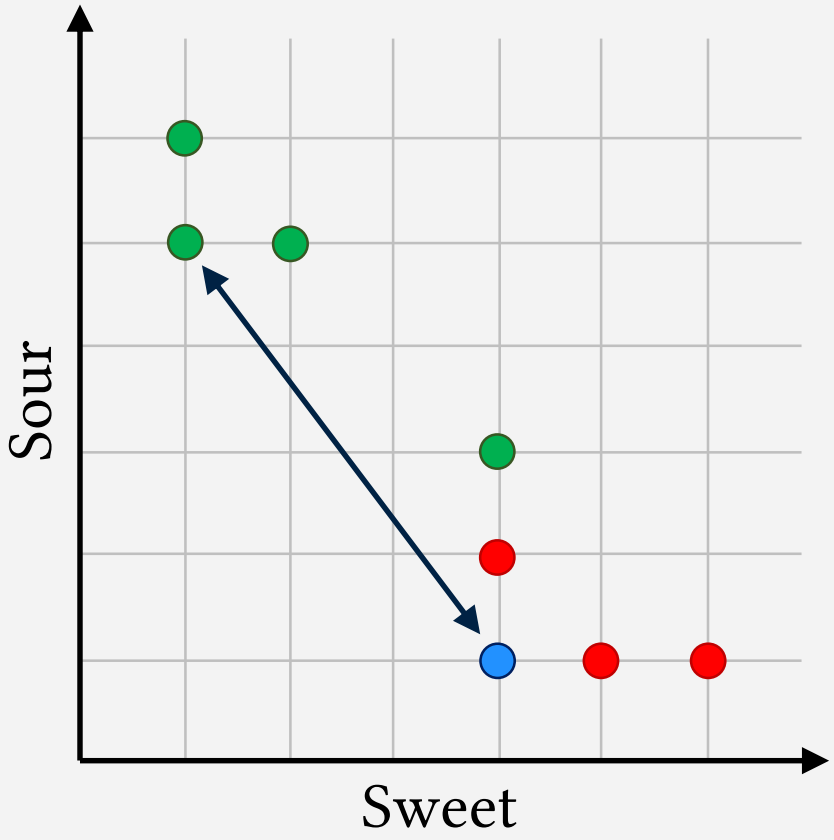


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$d([1,5],[4,1]) = \sqrt{25}$

KNN Example

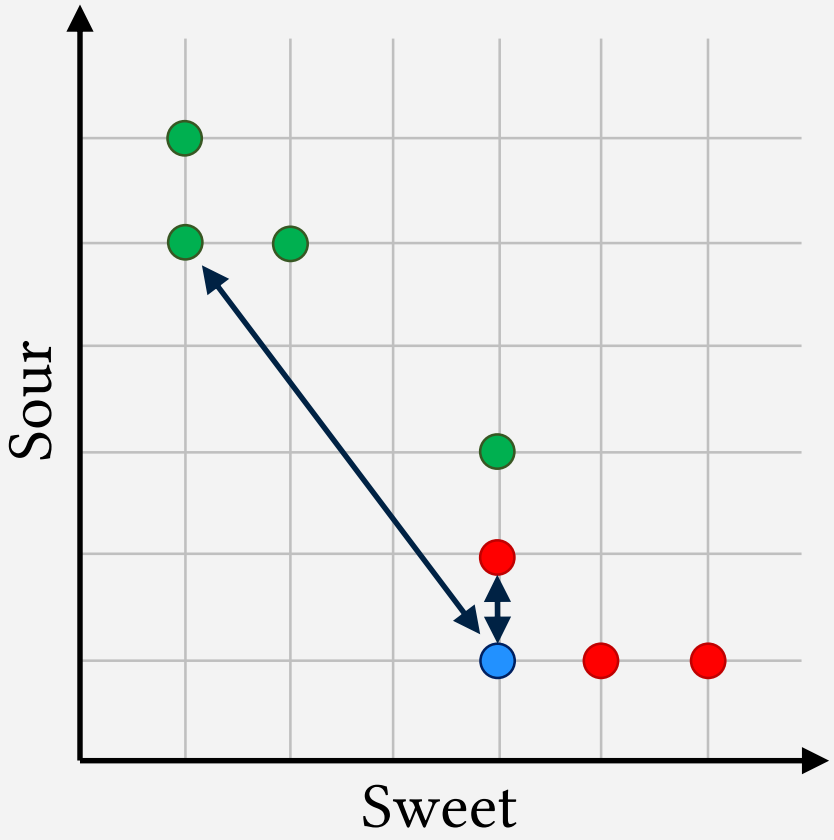


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$d([1,5],[4,1]) = 5$

KNN Example

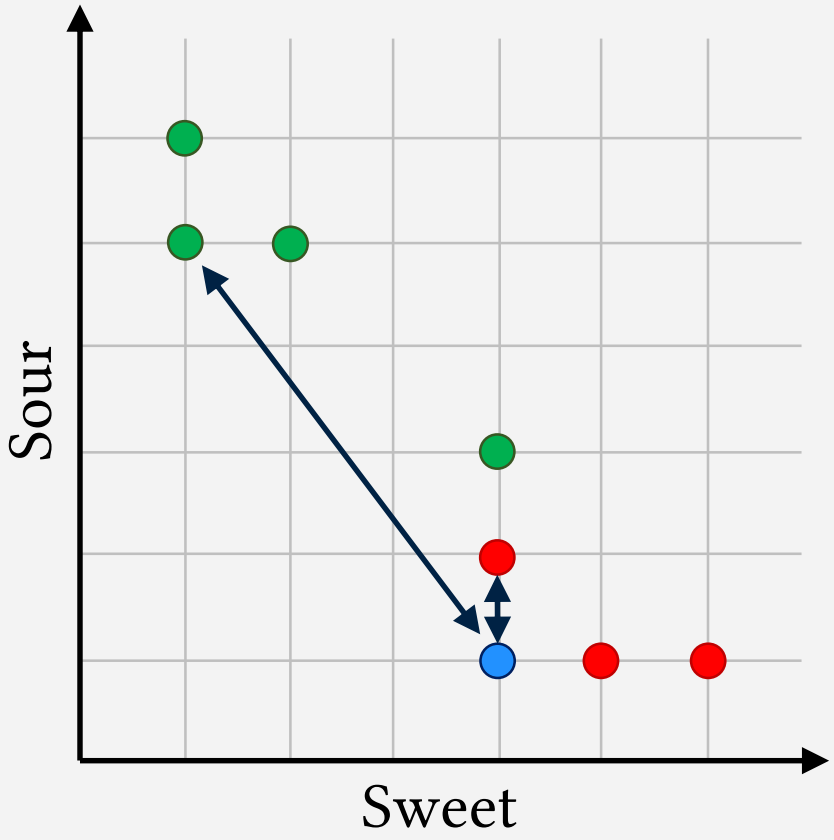


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$d([4,2],[4,1]) = ?$

KNN Example

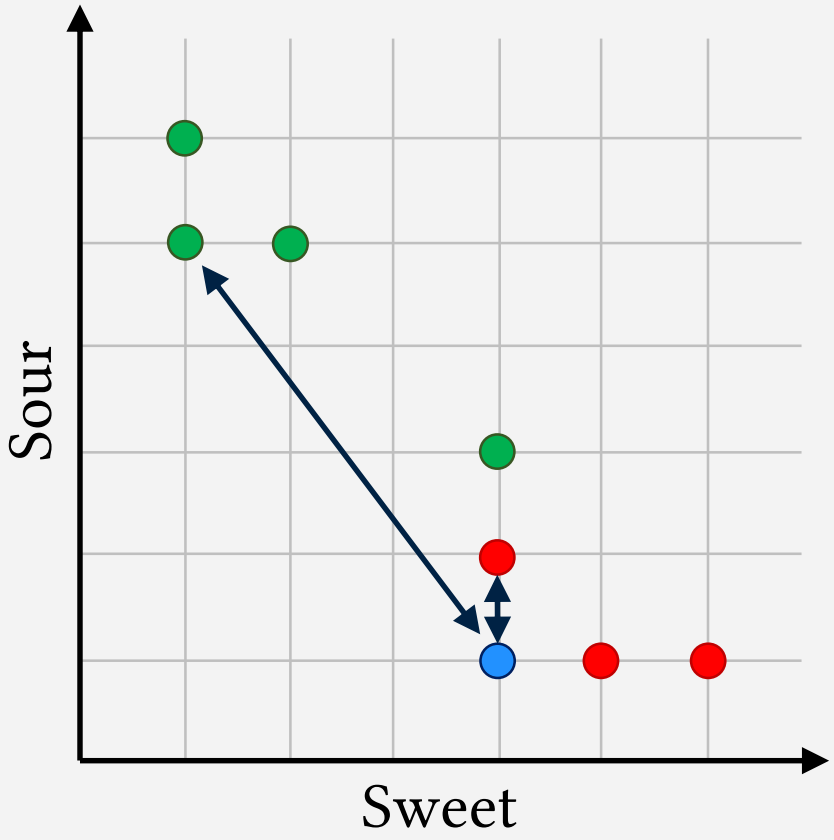


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$$d([4,2],[4,1]) = \sqrt{(4-4)^2 + (1-2)^2}$$

KNN Example

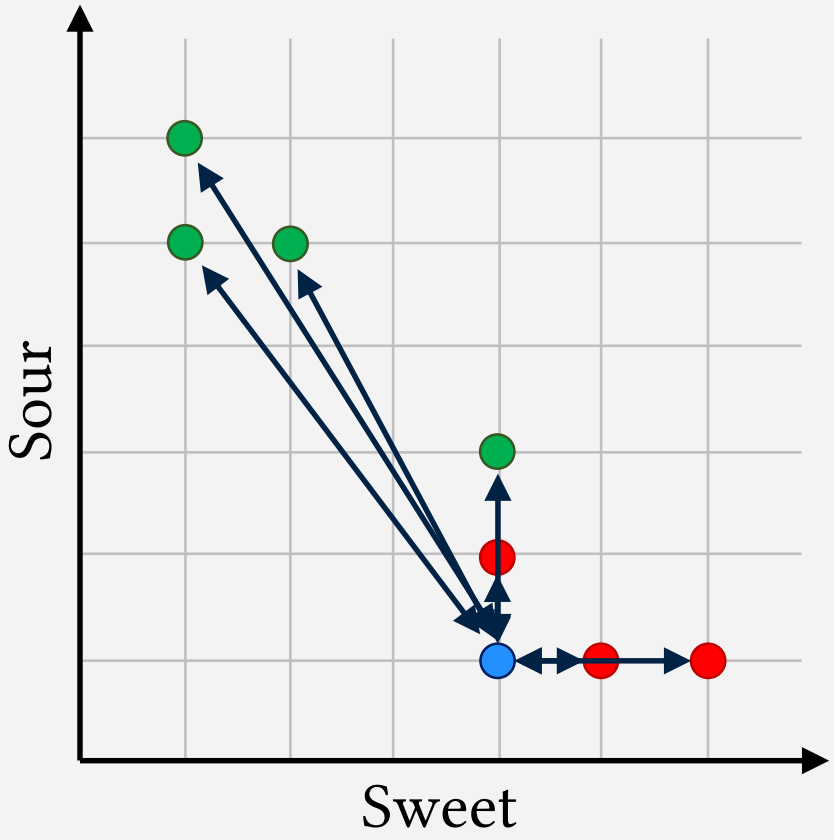


Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

$d([4,2],[4,1]) = 1$

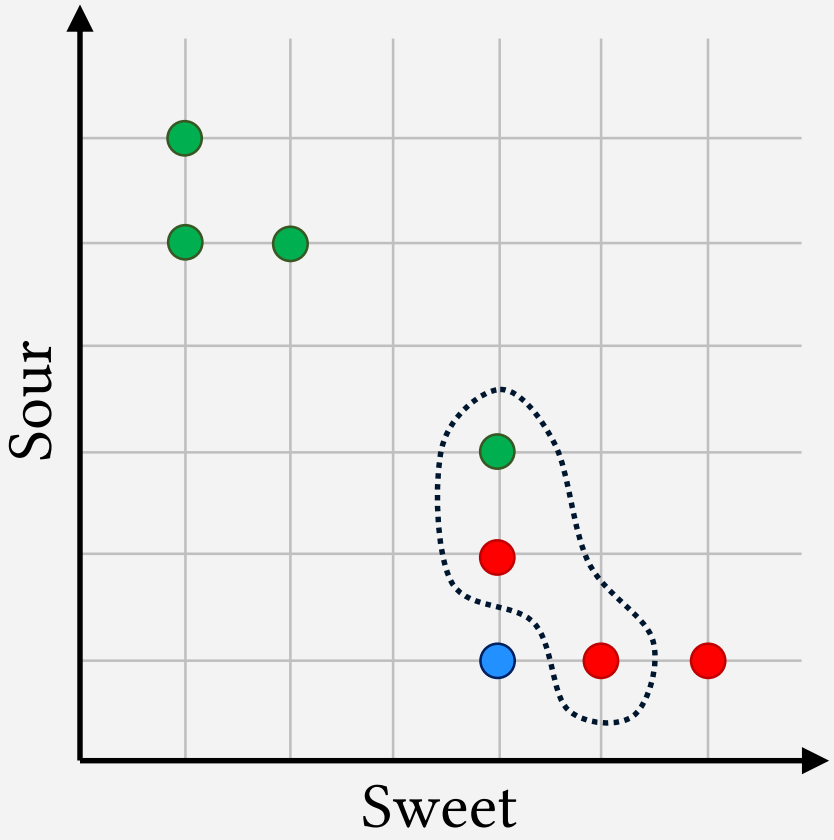
KNN Example



Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

KNN Example



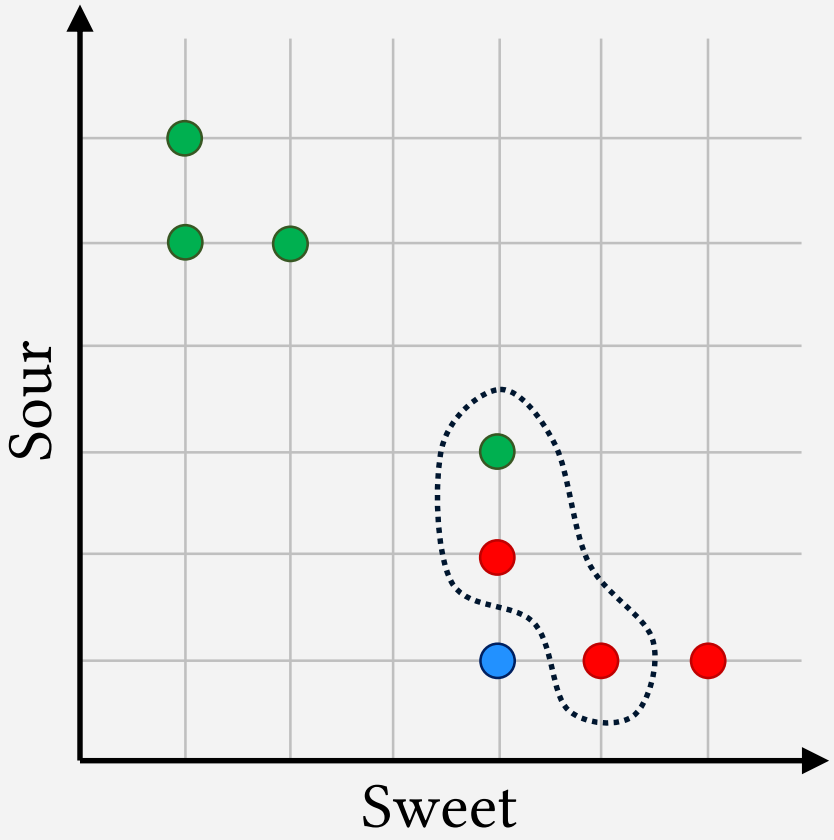
Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

The 3 nearest neighbors:

- Cantaloupe ($d = 1$)
- Honeydew ($d = 1$)
- Orange ($d = 2$)

KNN Example



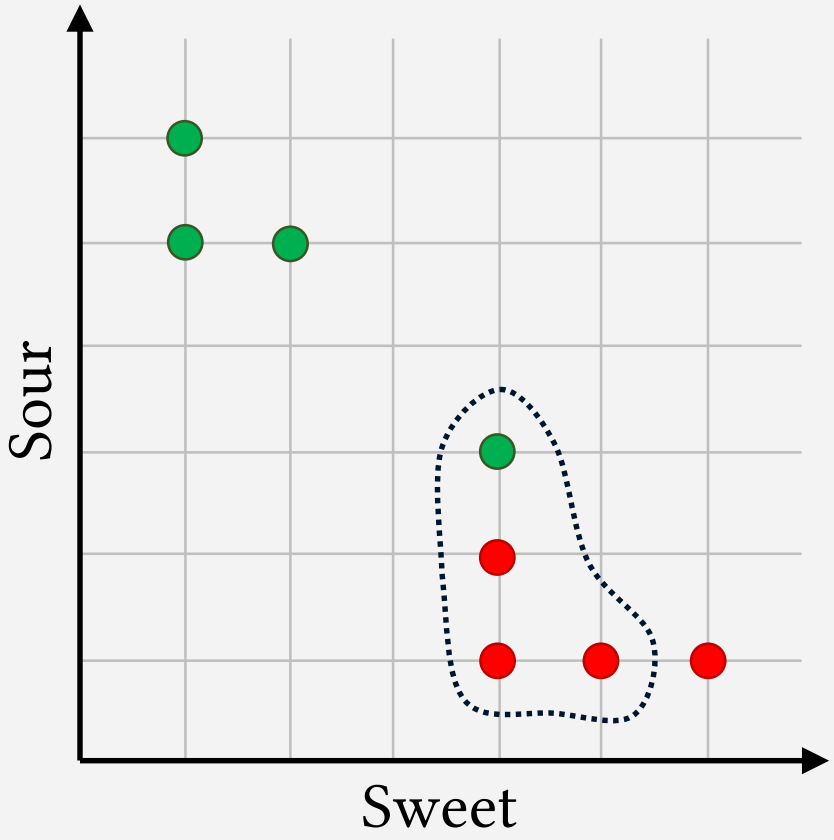
Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

The 3 nearest neighbors:

- Cantaloupe (Melon)
- Honeydew (Melon)
- Orange (Citrus)

KNN Example



Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

By majority vote (2 for melon and 1 for citrus), we classify a Santa Claus Melon as a melon.

Distance Metrics

In addition to Euclidian distance, we might use:

- **Hamming Distance** for comparing strings of text.
- **Pearson Correlation Coefficient** for strings of numbers

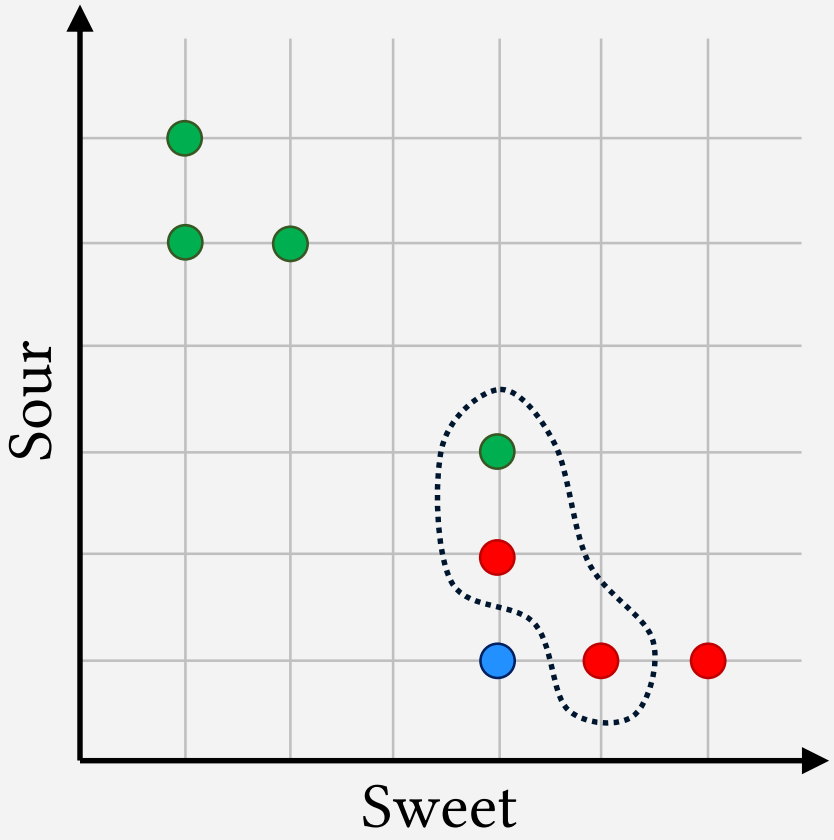
KNN Problems

Problem: If training data is highly skewed (e.g. we have many examples of citrus but only a few examples of melons), KNN may err in favor of the majority class label.

Solution: Weight each neighbor's vote by its distance from the query.

Specifically, weight each vote by $\frac{1}{d}$ where d is the distance between the neighbor and query.

KNN Example



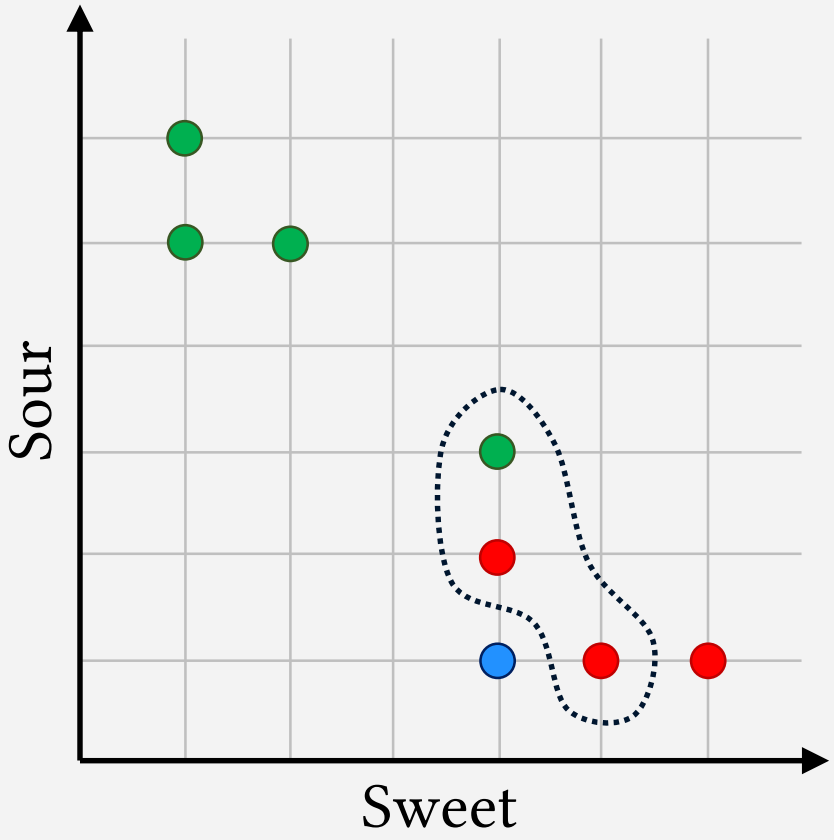
Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

The 3 nearest neighbors:

- Cantaloupe ($d = 1$ Melon)
- Honeydew ($d = 1$ Melon)
- Orange ($d = 2$ Citrus)

KNN Example



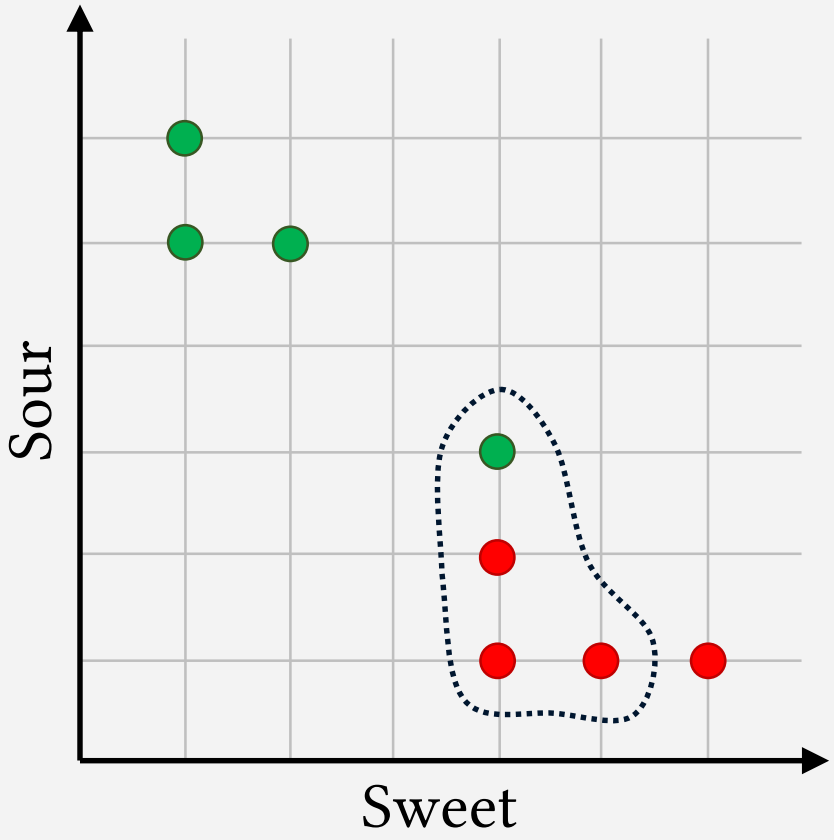
Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

The 3 nearest neighbors:

- 1/1 vote for Melon
- 1/1 vote for Melon
- 1/2 vote for Citrus

KNN Example



Unknown observation:

- Santa Claus Melon
  sweetness = 4
  sourness = 1

By majority vote (2 for melon and ½ for citrus), we classify a Santa Claus Melon as a melon.

KNN Problems

Problem: Because most distance metrics consider all attributes, KNN is very sensitive to irrelevant information.

For example, if we included the day of the week on which the taste test occurred in our data set, this would influence distance calculations even though it has no relevance to the class label.

Contrast this weakness to techniques like decision tree induction, which can exclude irrelevant attributes.

Lazy Learning Tradeoffs

- Positive: No training! No time spent training and no memory taken up by a model.
- Positive: Incorporating new training data is easy, because there is no model to rebuild.
- Negative: Every time we want to classify something, we need to perform distance calculations for every* example in the training database. This makes classification expensive (relative to most eager learning techniques) both in terms of time and memory.

*There are ways to avoid testing every example.