Introduction to K Nearest Neighbors

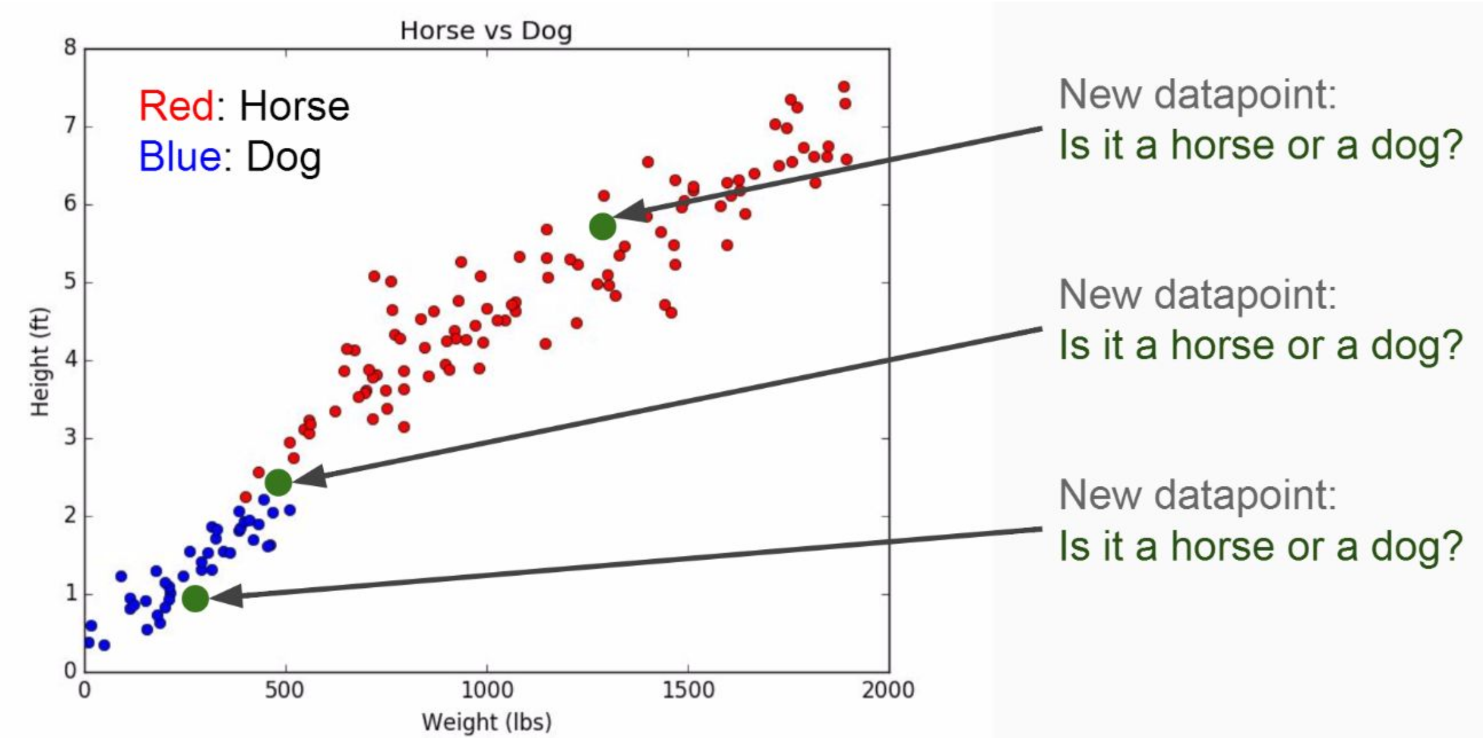
KNN

K Nearest Neighbors is a **classification** algorithm that operates on a very simple principle.

It is best shown through example!

Imagine we had some imaginary data on Dogs and Horses, with heights and weights.

KNN


Horse vs Dog
Red: Horse
Blue: Dog
Height (ft)
Weight (lbs)
0
1
2
3
4
5
6
7
8
0
500
1000
1500
2000
New datapoint:
Is it a horse or a dog?
New datapoint:
Is it a horse or a dog?
New datapoint:
Is it a horse or a dog?

KNN

Training Algorithm:

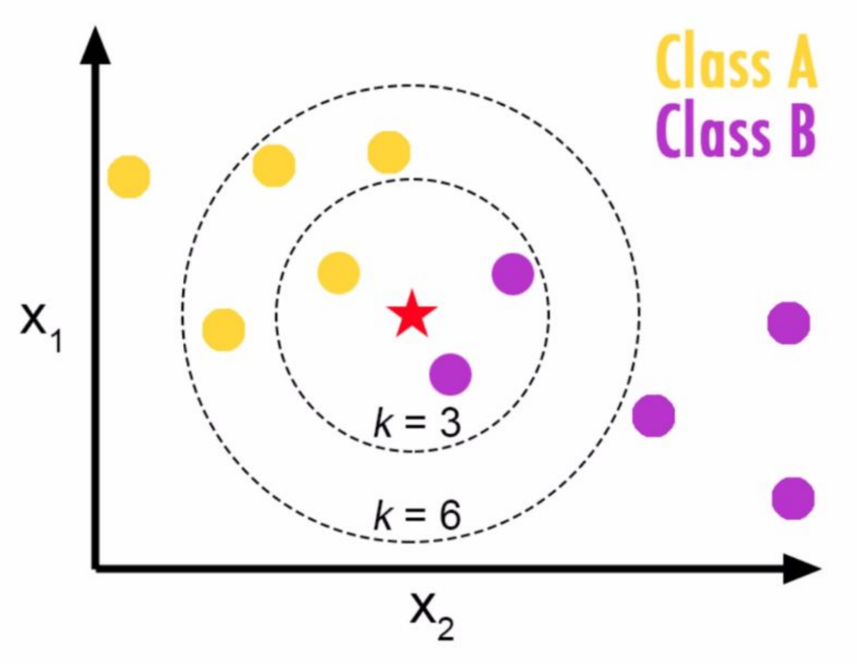
1. Store all the Data

Prediction Algorithm:

1. Calculate the distance from x to all points in your data
2. Sort the points in your data by increasing distance from x
3. Predict the majority label of the “k” closest points

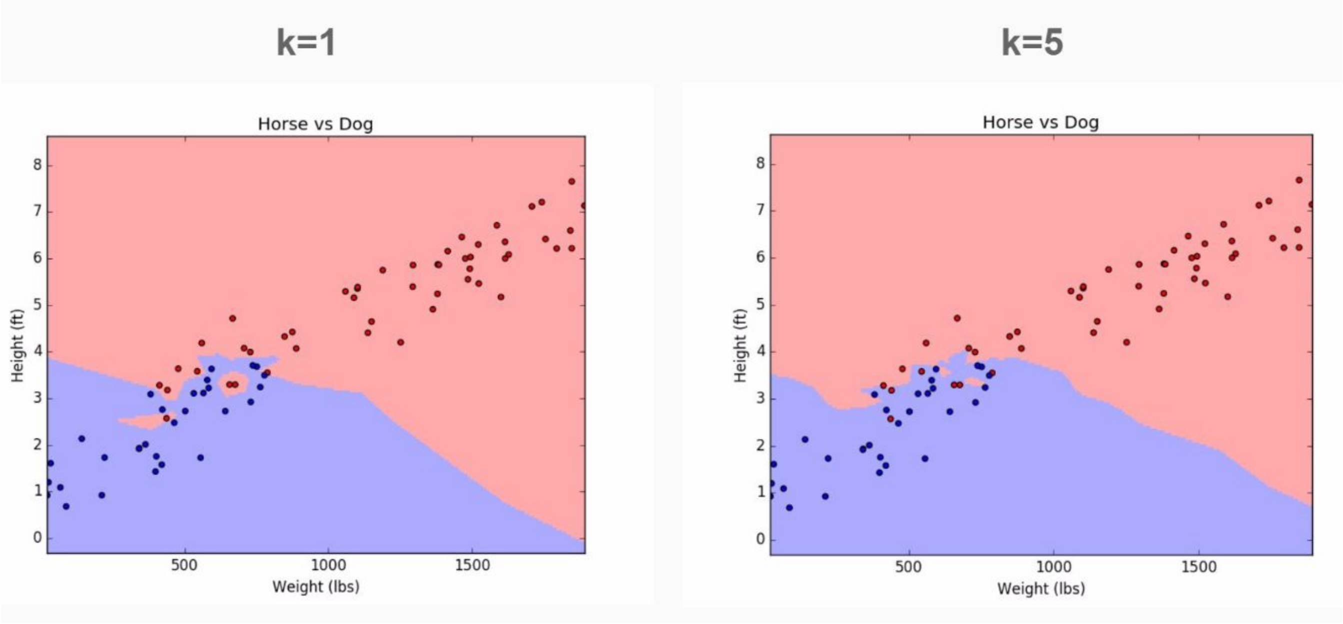
KNN

Choosing a K will affect what class a new point is assigned to:



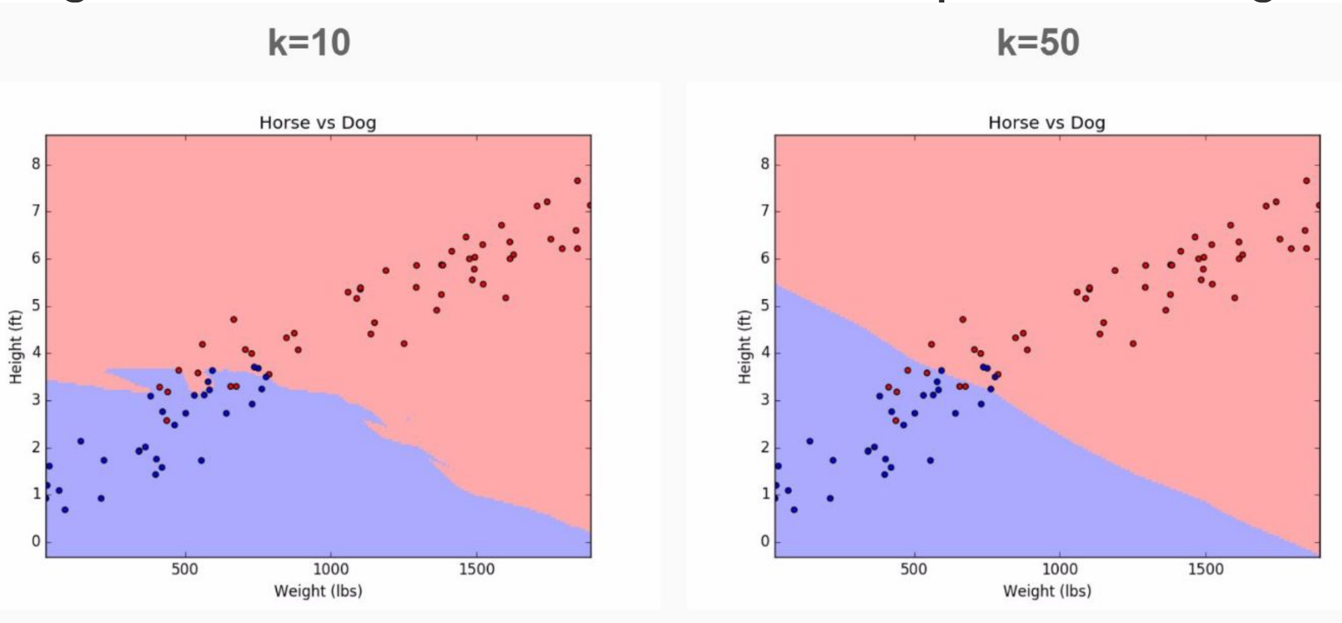
KNN

Choosing a K will affect what class a new point is assigned to:



KNN

Choosing a K will affect what class a new point is assigned to:



KNN

Pros

- Very simple
- Training is trivial
- Works with any number of classes
- Easy to add more data
- Few parameters
  - K
  - Distance Metric

KNN

Cons

- High Prediction Cost (worse for large data sets)
- Not good with high dimensional data
- Categorical Features don’t work well

MACHINE LEARNING

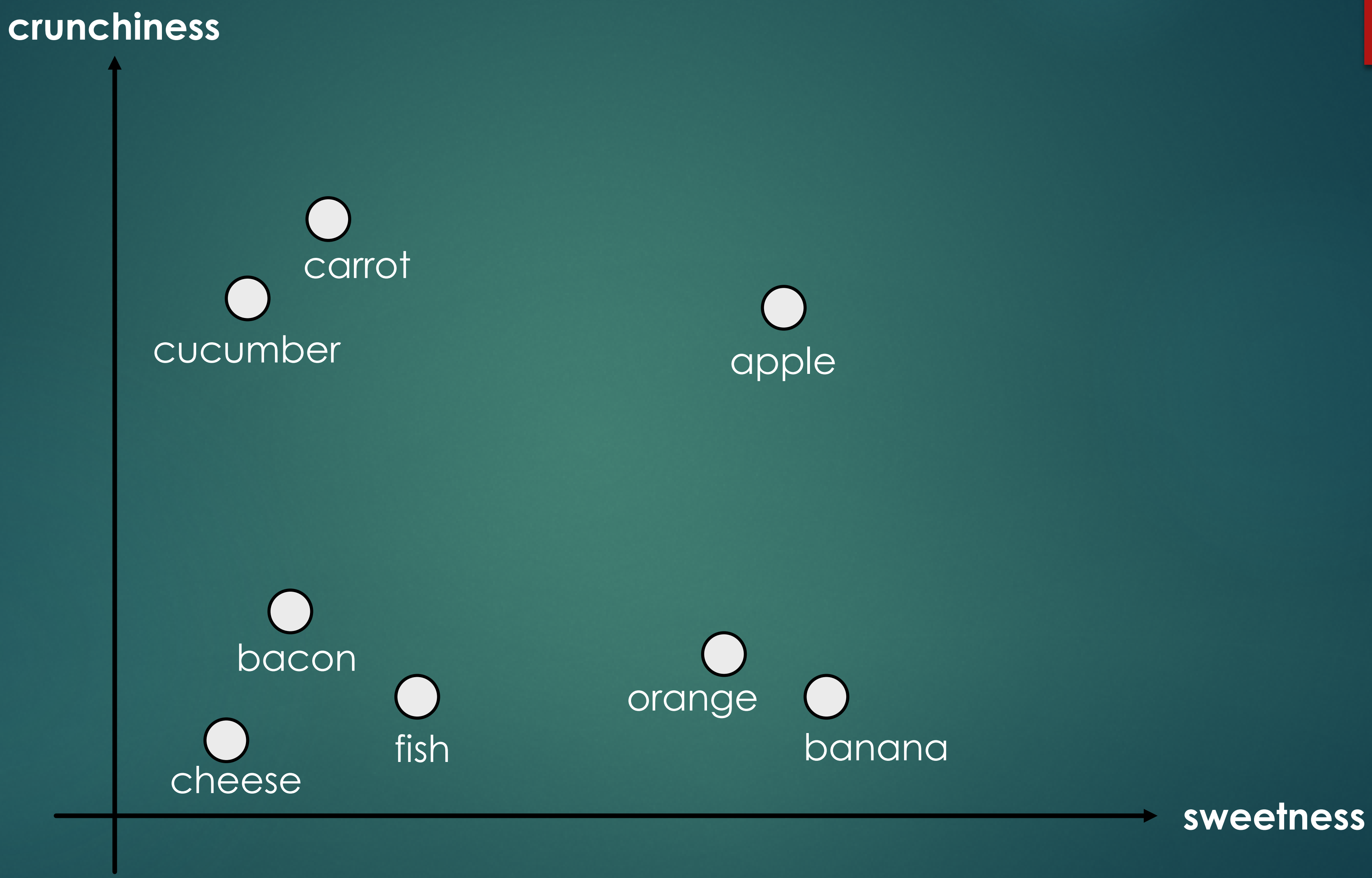
K-NEAREST NEIGHBORS ALGORITHM

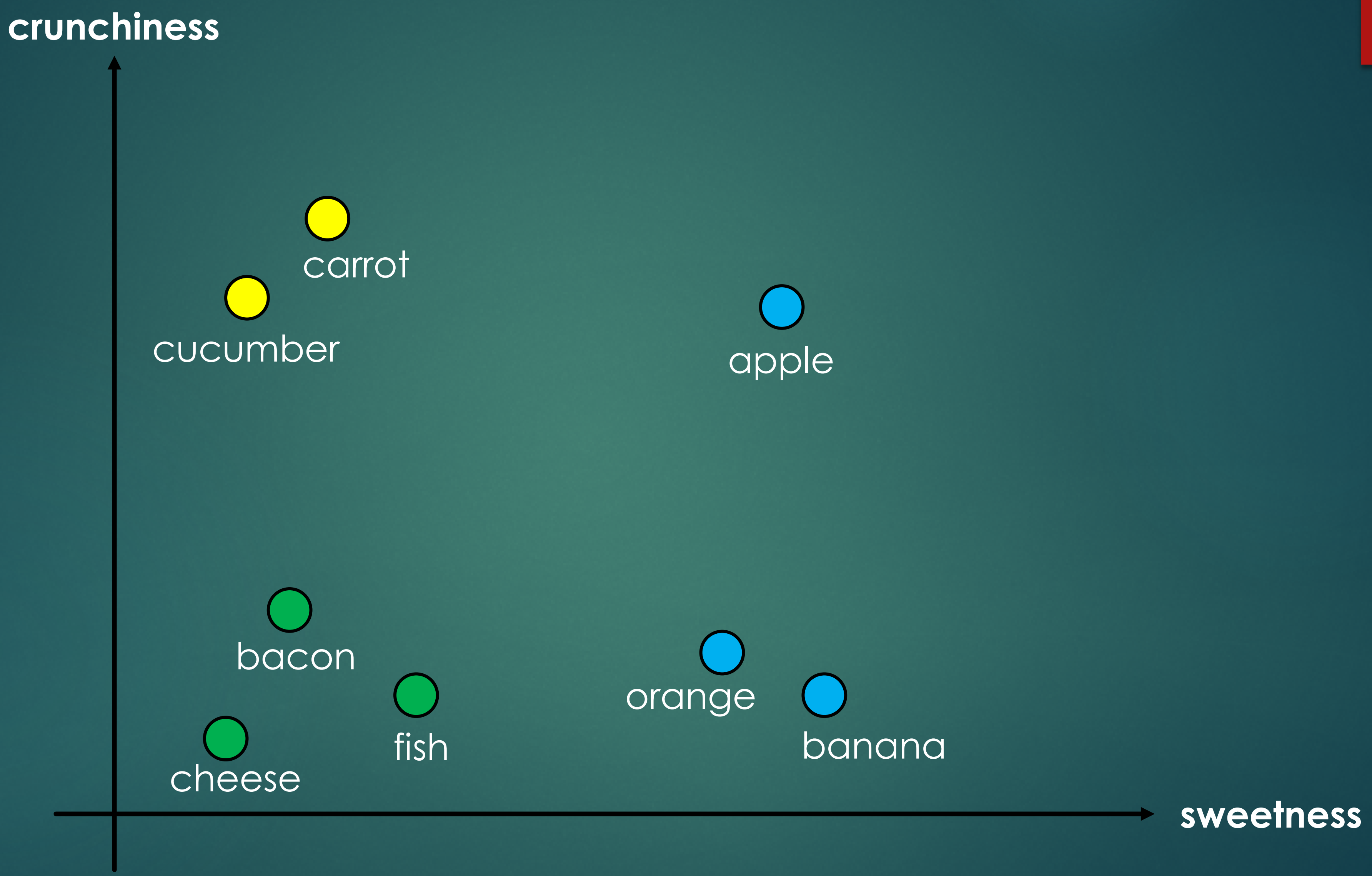
K-nearest neighbors classifier

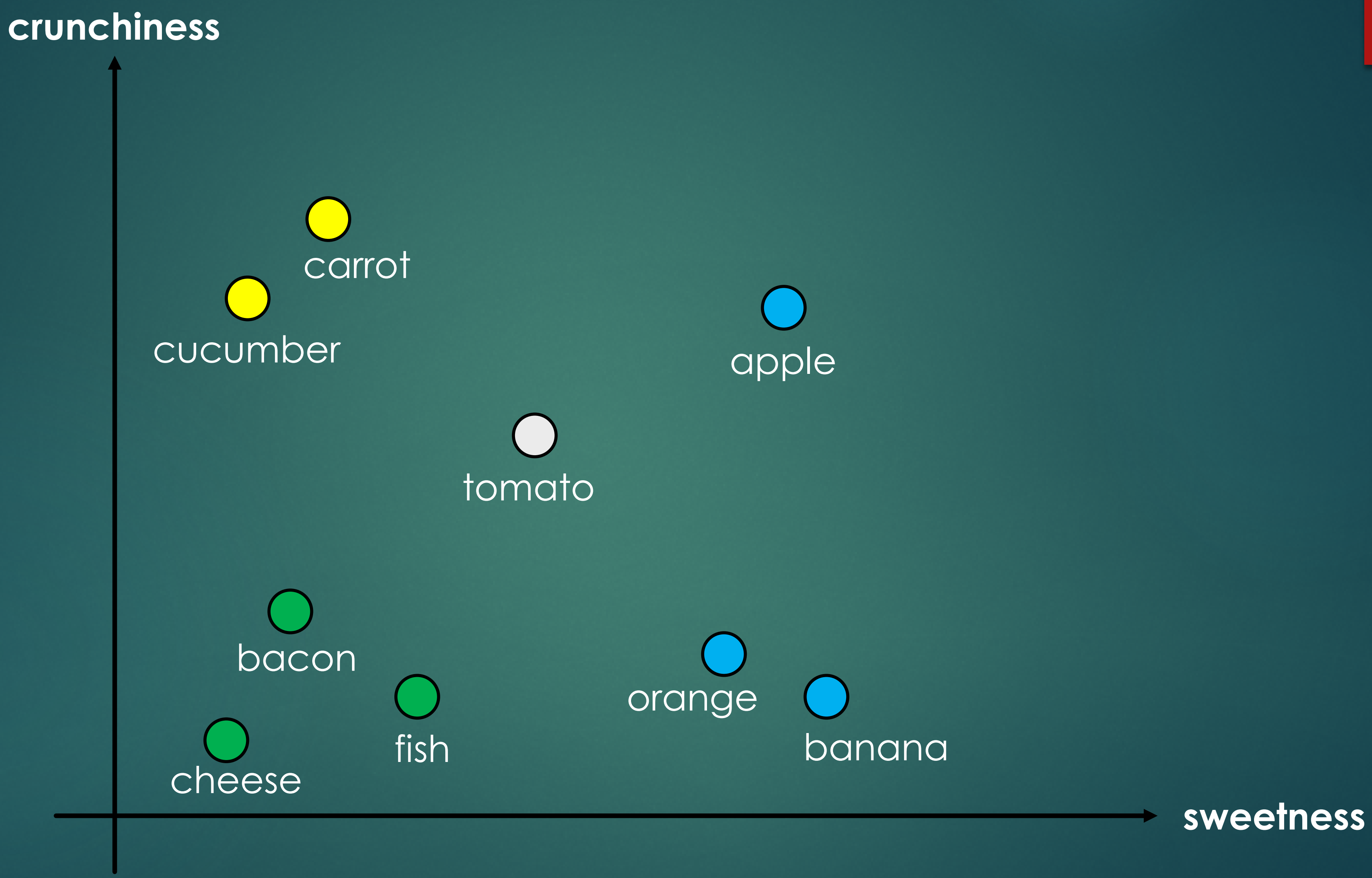
- K-nearest neighbors classifiers can classify examples by assigning them the class of the most similar labeled examples
- Very simple **BUT** extremely powerful algorithm !!!
- **kNN** is well suited for classification tasks where the relationship between the features are very complex and hard to understand
- We have a training dataset → examples that are classified into several categories
- We have a new example: (with the same number of features as the training data) → **kNN** algorithm identifies **k** elements in the training dataset that are the „nearest" in similarity
- The unlabeled test example is assigned to the class of the majority of the **k** nearest neighbors

ingredients	sweetness	crunchiness	type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
cheese	1	1	protein

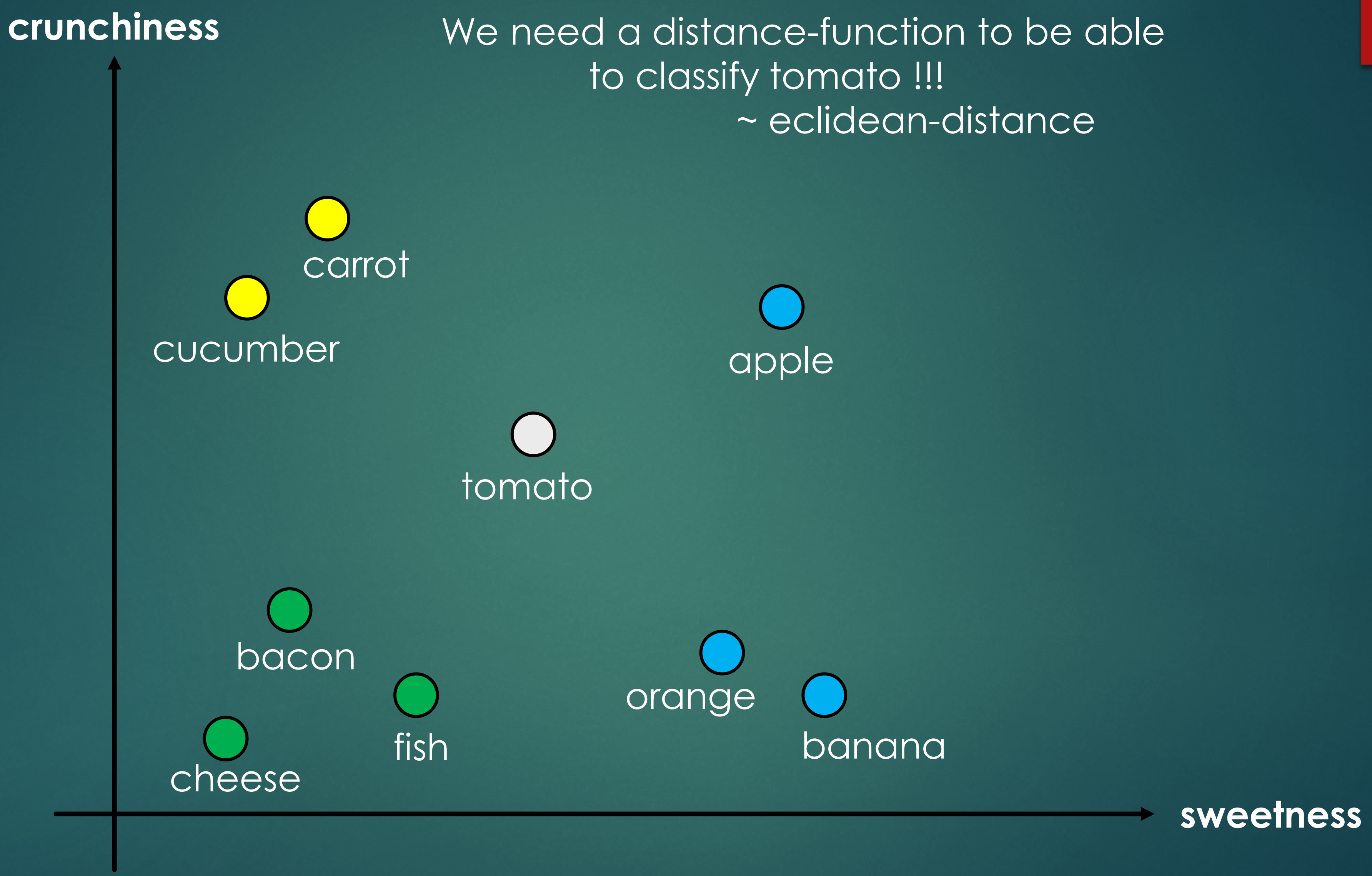
ingredients	sweetness	crunchiness	type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
cheese	1	1	protein
tomato	6	4	???


crunchiness
carrot
cucumber
apple
bacon
orange
fish
banana
cheese
sweetness


crunchiness
carrot
cucumber
apple
bacon
fish
cheese
orange
banana
sweetness


crunchiness
carrot
cucumber
apple
tomato
bacon
orange
banana
fish
cheese
sweetness

We need a distance-function to be able to classify tomato !!!
~ eclidean-distance



Euclidean-distance

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

ingredients	sweetness	crunchiness	type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
cheese	1	1	protein
tomato	6	4	???

$$\text{dist(tomato,carrot)} = \sqrt{(6-7)^2+(4-10)^2} = 6.083$$

dist(tomato,carrot) = $\sqrt{(6-7)^2+(4-10)^2} = 6.083$

dist(tomato,apple) = $\sqrt{(6-10)^2+(4-9)^2} = 6.403$

dist(tomato,bacon) = $\sqrt{(6-1)^2+(4-4)^2} = 5$

dist(tomato,banana) = $\sqrt{(6-10)^2+(4-1)^2} = 5$

dist(tomato,cheese) = $\sqrt{(6-1)^2+(4-1)^2} = 5.83$

dist(tomato,carrot) = $\sqrt{(6-7)^2+(4-10)^2} = 6.083$

dist(tomato,apple) = $\sqrt{(6-10)^2+(4-9)^2} = 6.403$

dist(tomato,bacon) = $\sqrt{(6-1)^2+(4-4)^2} = 5$

dist(tomato,banana) = $\sqrt{(6-10)^2+(4-1)^2} = 5$

dist(tomato,cheese) = $\sqrt{(6-1)^2+(4-1)^2} = 5.83$

k=1 we consider the smallest distance: bacon and banana

k=2 we consider the 2 smallest distances: bacon and banana
50%-50% that tomato is a fruit or a protein

k=3 we consider the 3 smallest distances: bacon, banana and cheese
So tomato appears to be a protein !!!

Choosing k values

- Deciding how many neighbors to use for kNN → determines how well the model will generalize and work on other dataset
- **k** is small → noisy data or outliers have a huge impact on our classifier ... this is called „underfitting”
- **k** is large → the classifier has the tendency to predict the majority class regardless of which neighbors are nearest ... this is called „overfitting”

Lazy learning

- Lazy learners does not learn anything !!!
- We just store the training data: training is very fast (because there is no training at all) BUT making the prediction is rather slow
- **WE DO NOT BUILD A MODEL !!!**
- This is a non-parametric learning: no parameters are to be learned about the data

Applications

- Optical character recognition + facial recognition (images and videos)
- Recommender systems: whether a person will enjoy a movie or not
- Identifying patterns in genetic data