# Lecture 5: K-Nearest Neighbor

## Winter 2018

Kai-Wei Chang

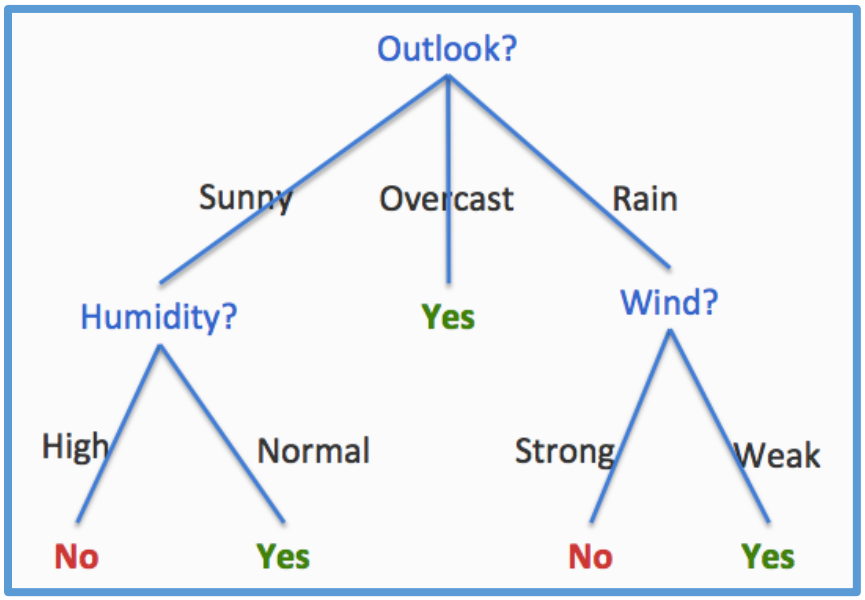
CS @ UCLA

kw+cm146@kwchang.net

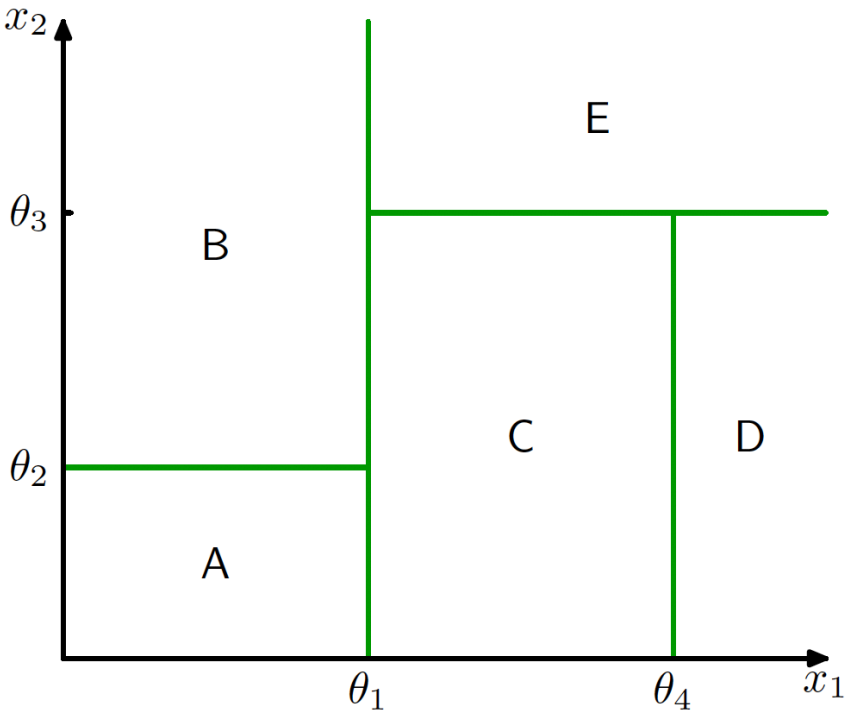
The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

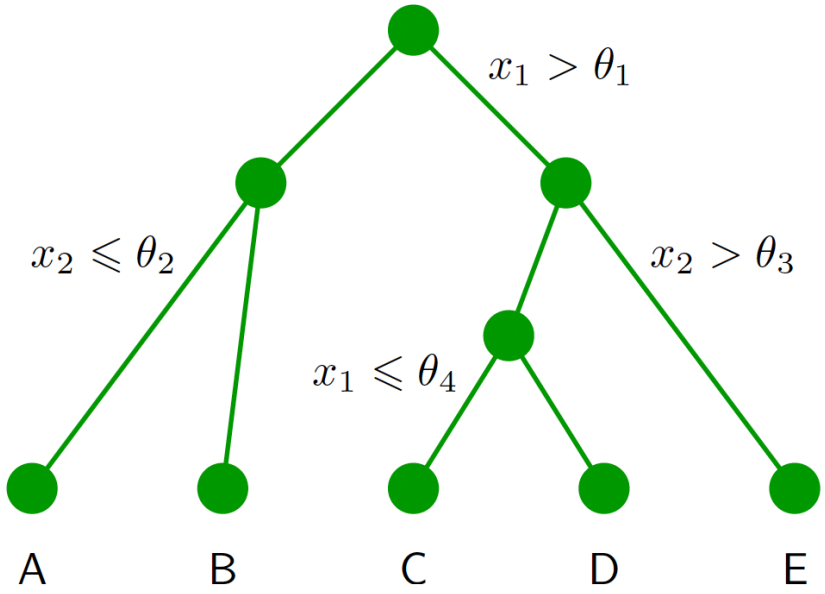
# Recap: Decision tree

- The attribute for splitting is picked based on the information gain



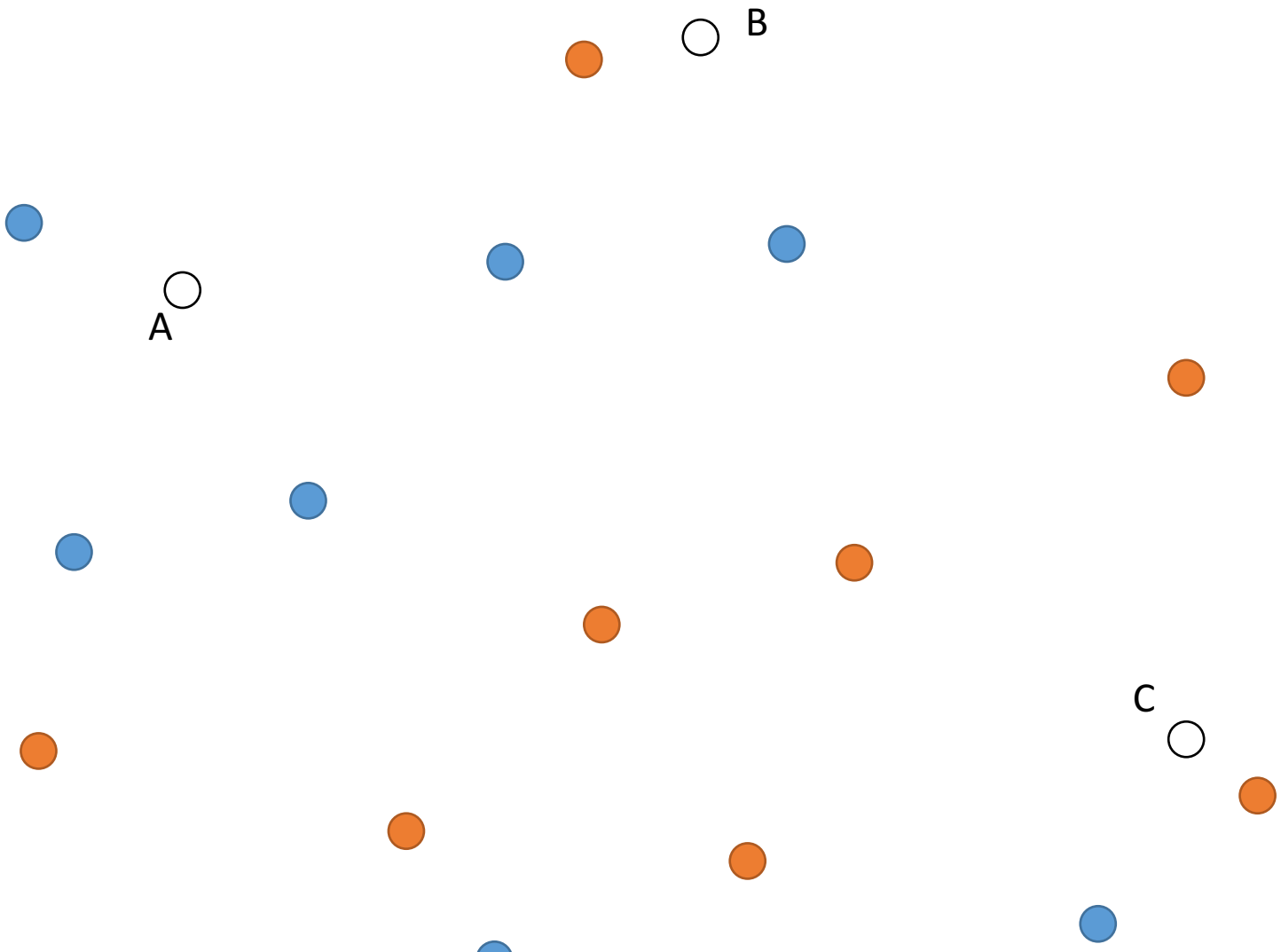
# A tree partitions the feature space




Other ways for splitting the input space?

# Motivation:
# How would you color the blank circles?



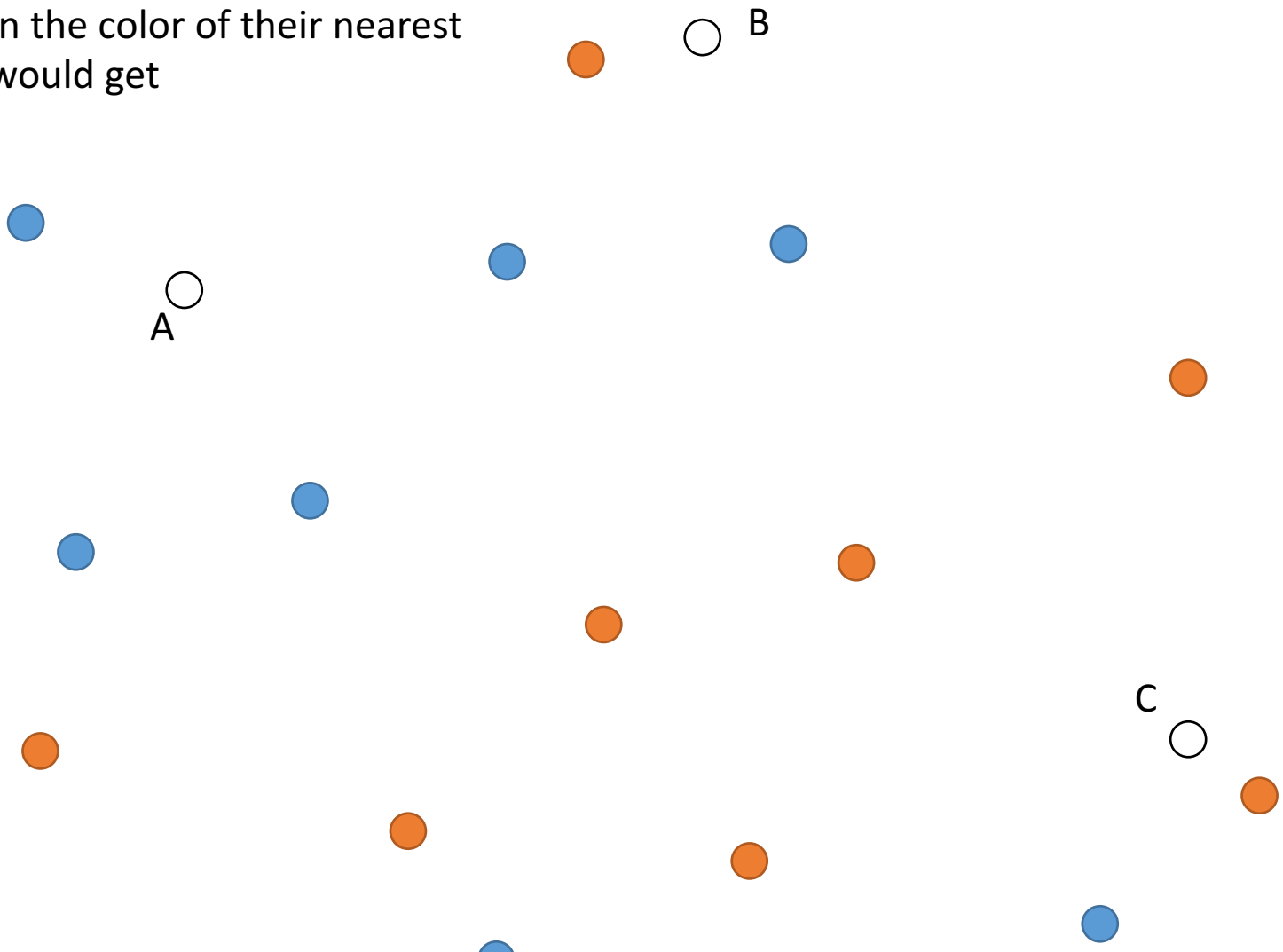
# How would you color the blank circles?

If we based it on the color of their nearest neighbors, we would get

A: Blue

B: Red

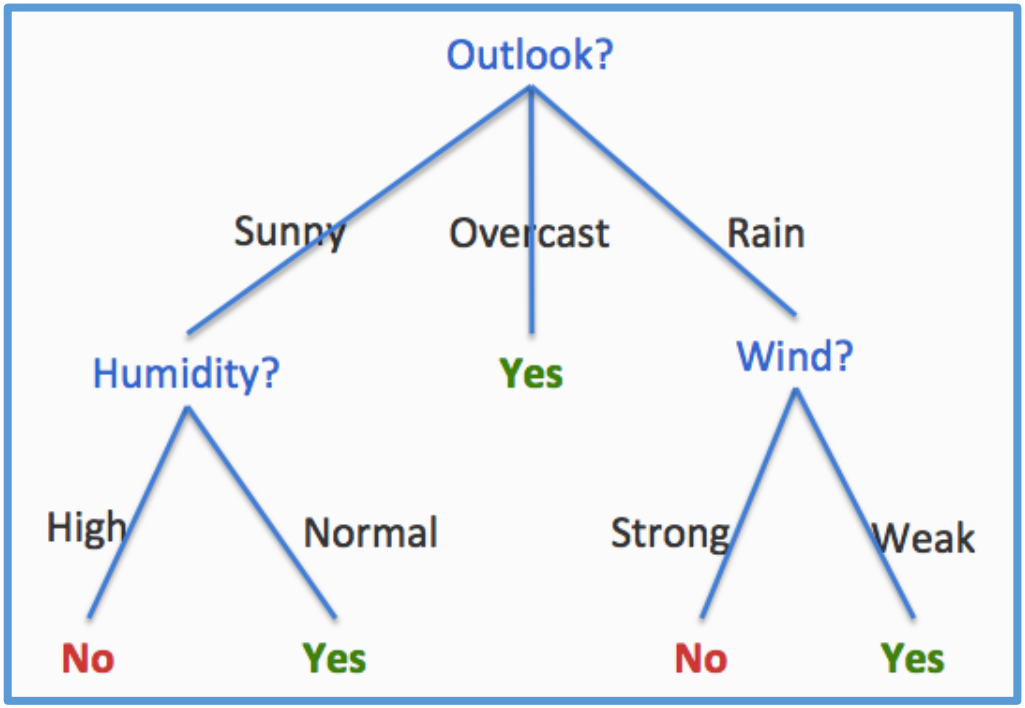
C: Red



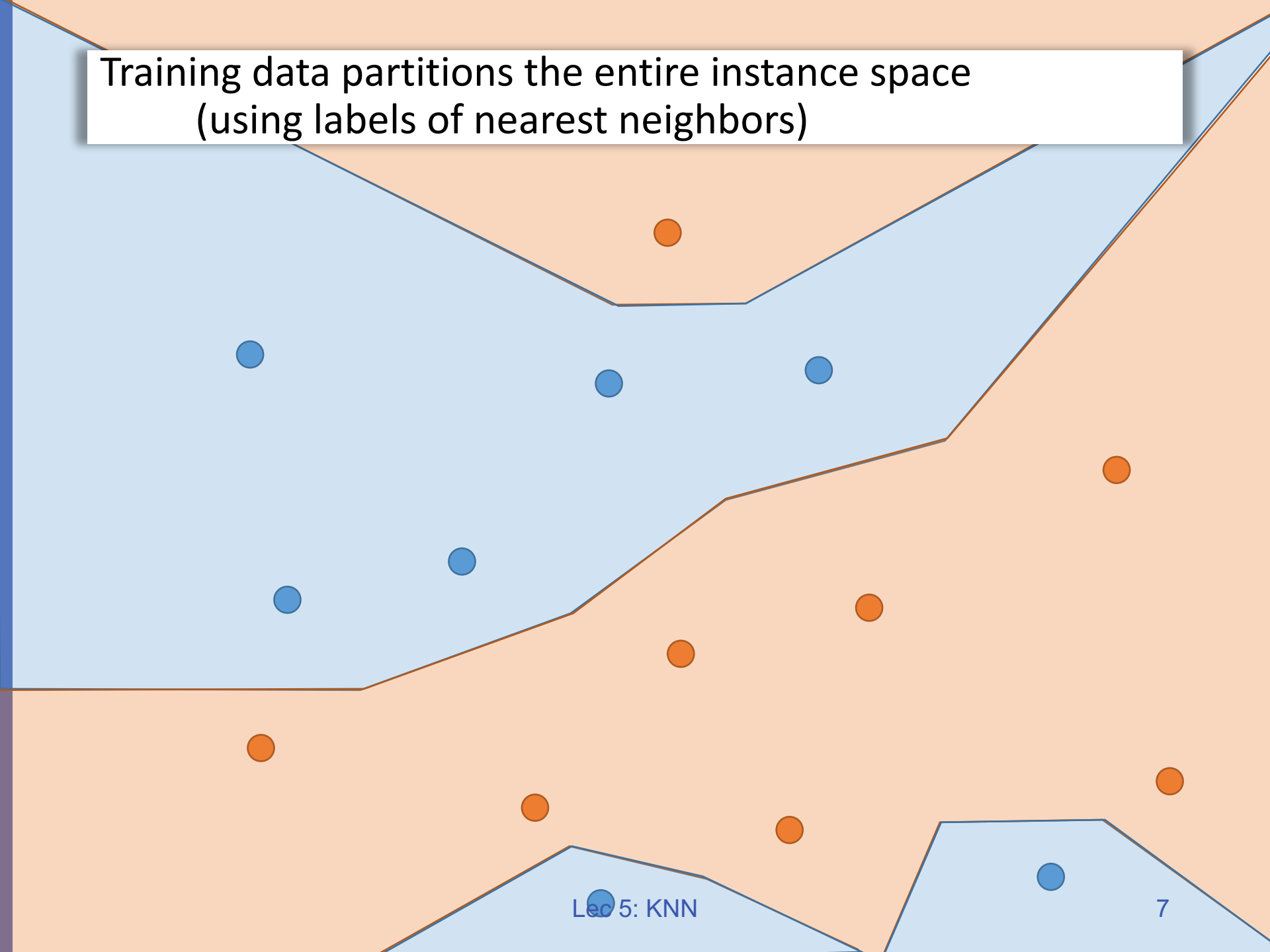
# Recap: Decision tree

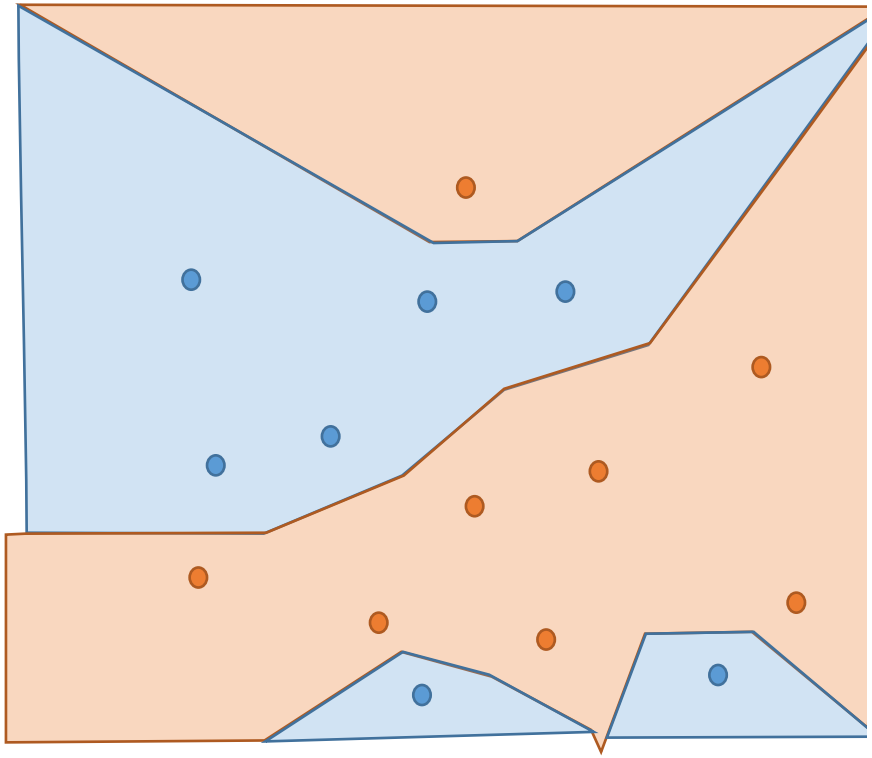
- Learning by divided the input space

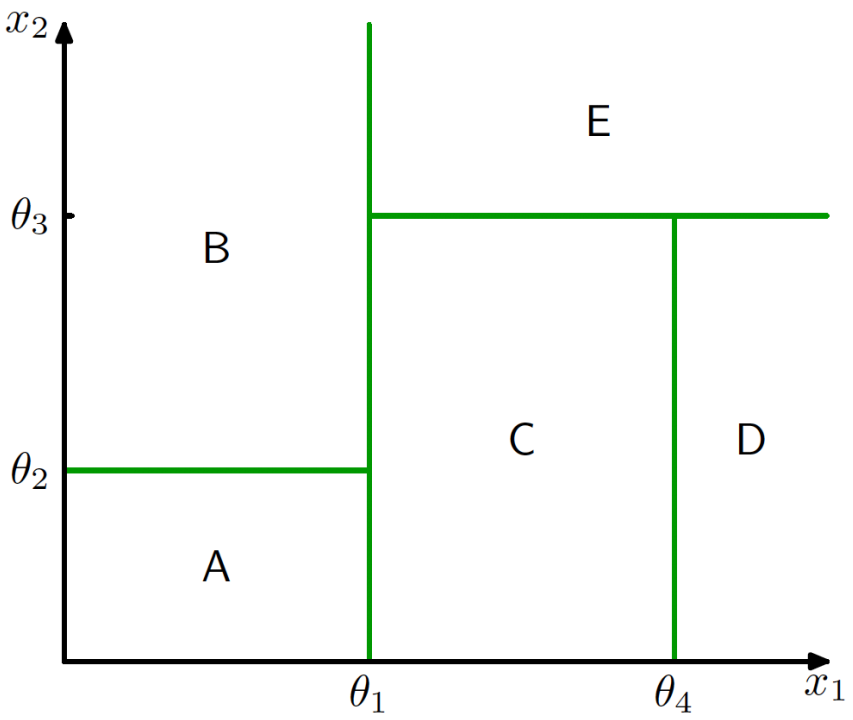
| | O | T | H | W | Play? |
|---|---|---|---|---|---|
| 1 | S | H | H | W | - |
| 2 | S | H | H | S | - |
| 3 | O | H | H | W | + |
| 4 | R | M | H | W | + |
| 5 | R | C | N | W | + |
| 6 | R | C | N | S | - |
| 7 | O | C | N | S | + |
| 8 | S | M | H | W | - |
| 9 | S | C | N | W | + |
| 10 | R | M | N | W | + |
| 11 | S | M | N | S | + |
| 12 | O | M | H | S | + |
| 13 | O | H | N | W | + |
| 14 | R | M | H | S | - |



Training data partitions the entire instance space
(using labels of nearest neighbors)





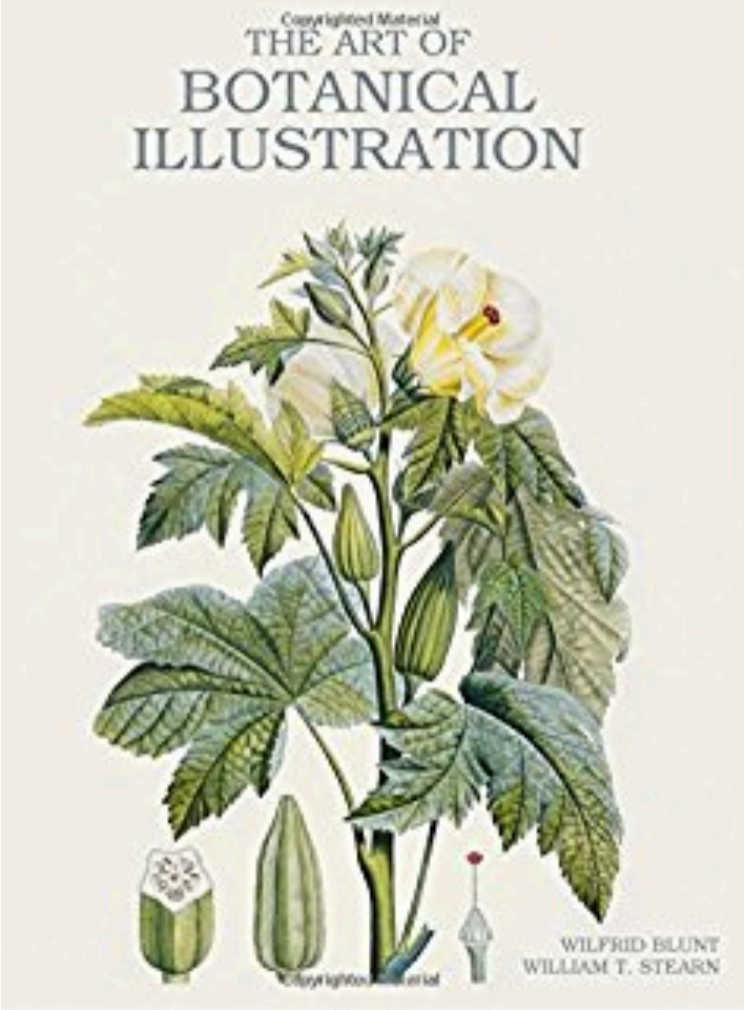



Questions:
How the boundaries are different in KNN and decision tree?

# Motivation: Learning from memorization Recognizing flowers

- What is this flower called?

# Look up at botanical illustration books




# Motivation: Learning from memorization Recognizing flowers

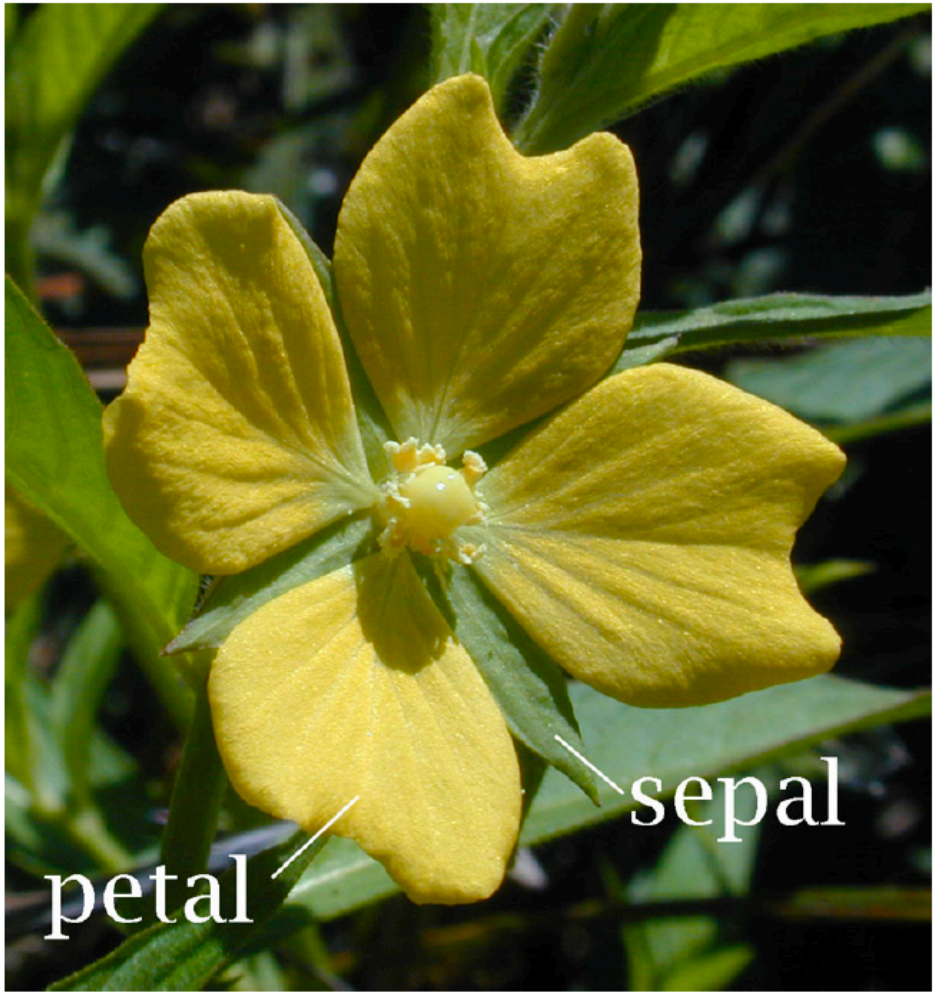
**Types of Iris: setosa, versicolor, and virginica**

# Motivation: Learning from memorization
# Recognizing flowers

**Types of Iris: setosa, versicolor, and virginica**

# Measuring the properties of the flowers

- Features: the widths and lengths of sepal and petal



# Snapshot of Iris data

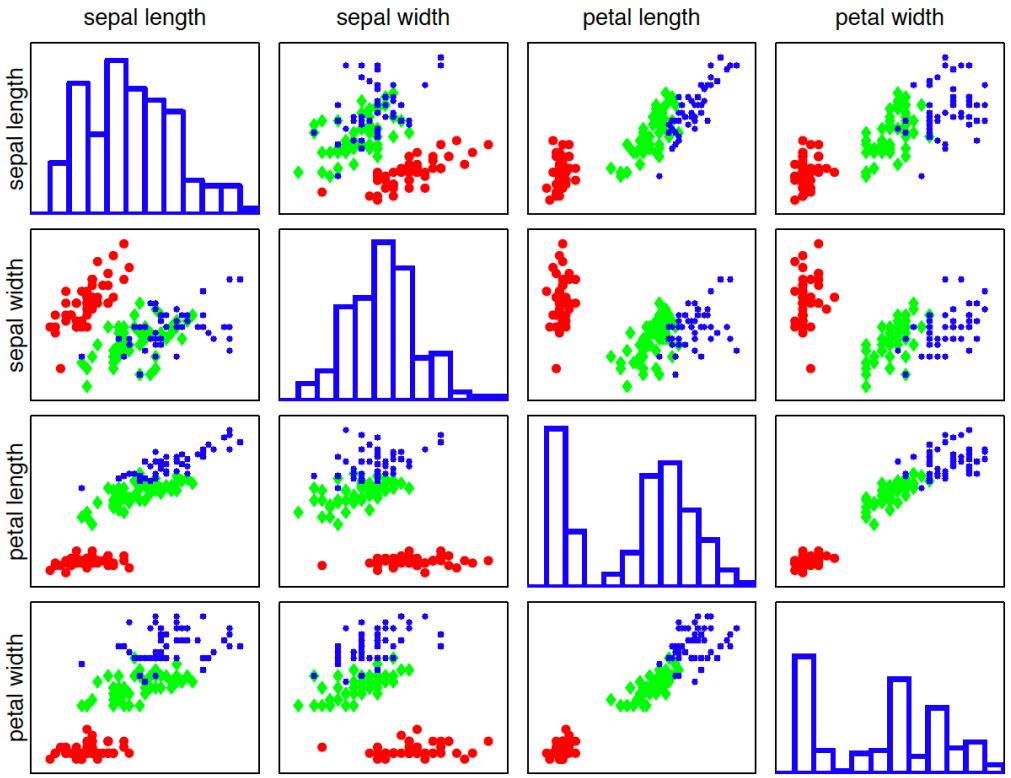
- Existing  Iris data
- 4 features
- 3 classes

**Fisher's *Iris* Data**

| Sepal length | Sepal width | Petal length | Petal width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.0 | 1.4 | 0.2 | *I. setosa* |
| 4.7 | 3.2 | 1.3 | 0.2 | *I. setosa* |
| 4.6 | 3.1 | 1.5 | 0.2 | *I. setosa* |
| 5.0 | 3.6 | 1.4 | 0.2 | *I. setosa* |
| 5.4 | 3.9 | 1.7 | 0.4 | *I. setosa* |
| 4.6 | 3.4 | 1.4 | 0.3 | *I. setosa* |
| 5.0 | 3.4 | 1.5 | 0.2 | *I. setosa* |
| 4.4 | 2.9 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.1 | 1.5 | 0.1 | *I. setosa* |

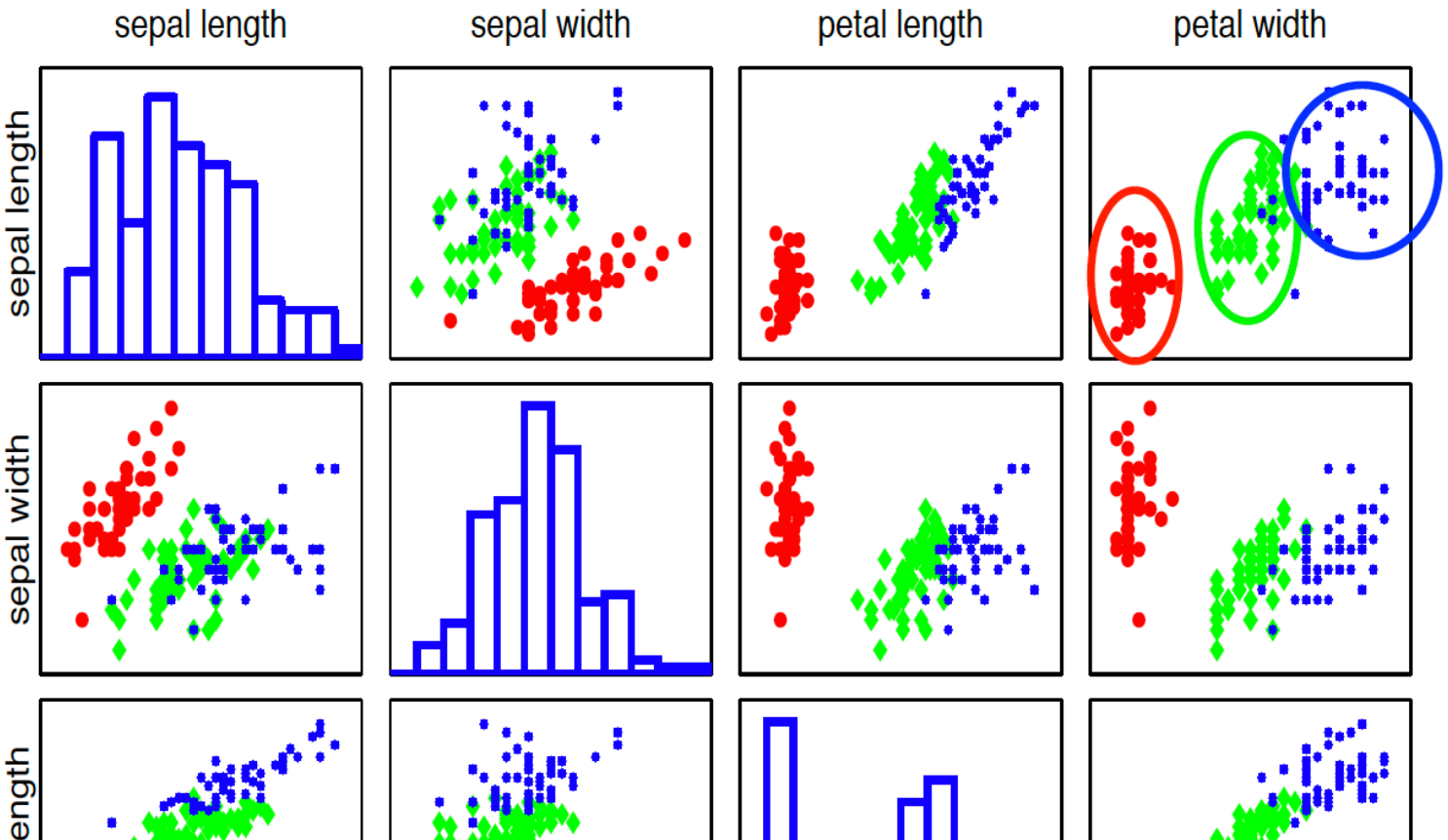
# Pairwise scatter plots of 131 flower specimens

Each colored point is an instance ( flower specimen ): setosa, versicolor, virginica



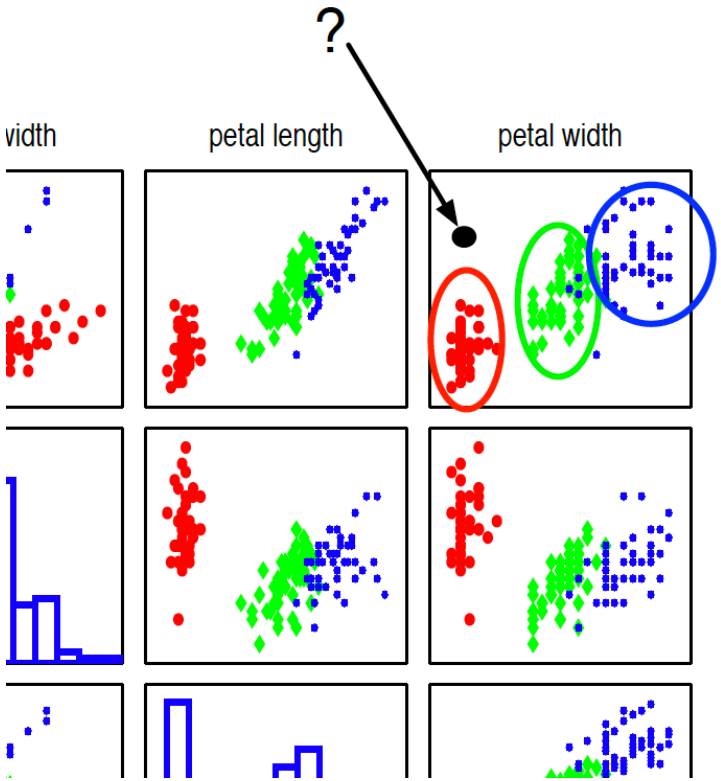
# Different types seem well-clustered and separable

- By even just two features



# Labeling an unknown flower

- Closer to the red cluster
  => label




Each colored point is an instance ( flower specimen ): setosa, versicolor, virginica

# What you will learn in this lecture

- What is the k-NN algorithm
- How to define distance between two points in the vector space
- Recap: parameter tuning
- Advantage/disadvantage of KNN
- Decision boundary of KNN
- The curse of the dimensionality
  - Why/when KNN may fail

# Nearest Neighbors: The basic version

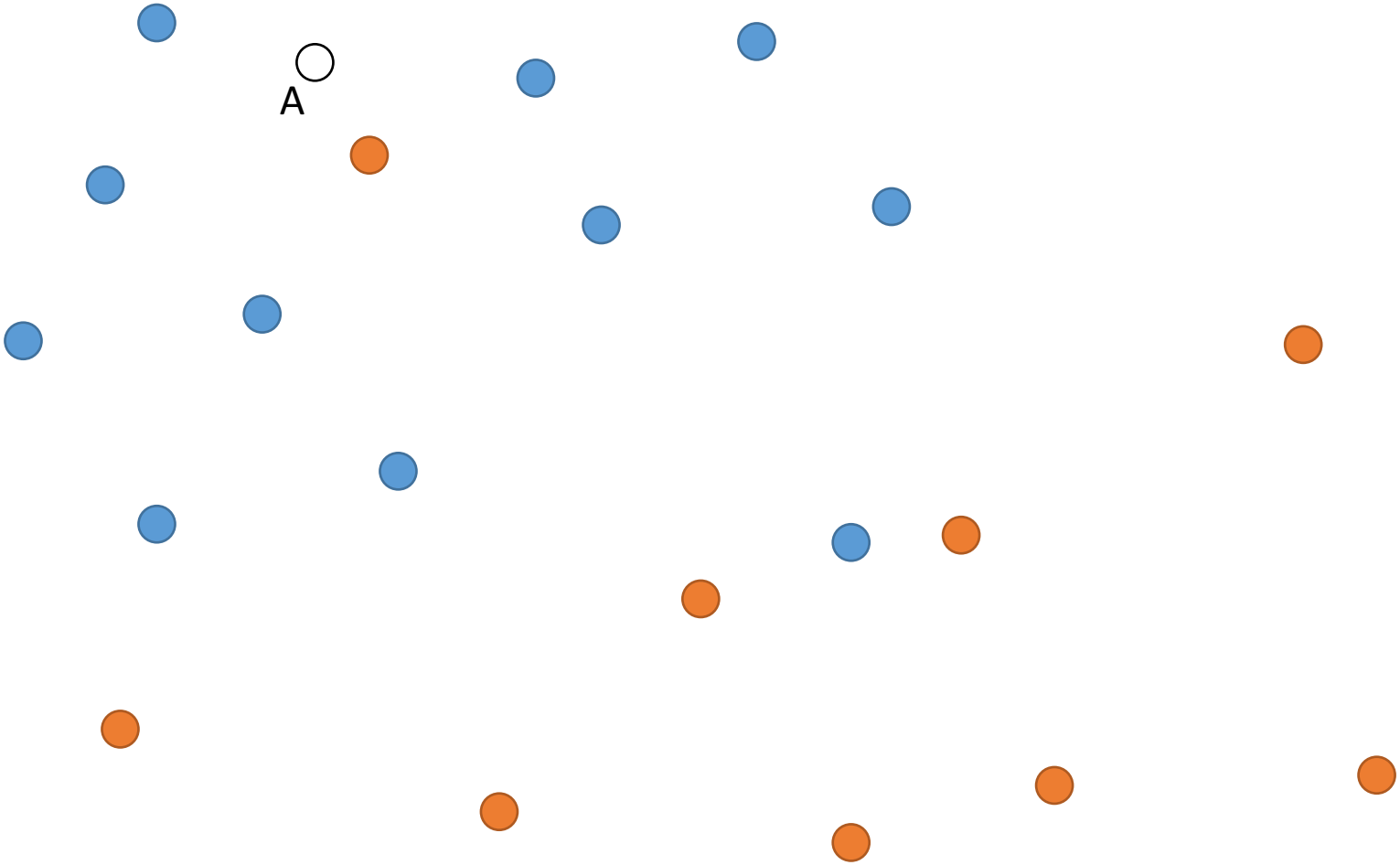
- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

- Prediction: for a new example $\mathbf{x}$
  - Find the training example $\mathbf{x}_i$ that is *closest* to $\mathbf{x}$
  - Predict the label of $\mathbf{x}$ to the label associated with $\mathbf{x}_i$

May not be a good idea

# Counterexample:
# How would you color the blank circles?



# K-Nearest Neighbors

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

- Prediction for a new example $\mathbf{x}$
  - Find the k *closest* training examples to $\mathbf{x}$
  - Construct the label of $\mathbf{x}$ using these k points.

# Issues in designing KNN algorithm (Modeling)

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM
- Learning: Just store all the training examples
- Prediction for a new example **x**
  - Find the k *closest* training examples to **x**
  - Construct the label of **x** using these k points.

How to choose k?

How to define distance?

How to aggregate the information and make the prediction?

# Issues in designing KNN algorithm (computation/algorithm)

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

How to store data?

- Prediction for a new example **x**
  - Find the k *closest* training examples to **x**
  - Construct the la[illegible] of **x** using these k points.

How to find the closest points?

# Issues in designing KNN algorithm (computation/algorithm)

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

How to store data?

- Prediction for a new example **x**
  - Find the k *closest* training examples to **x**
  - Construct the

How to find the closest points?

This is an important research topic, but I will not cover it in this class.
Reference: e.g. K-d tree (https://en.wikipedia.org/wiki/K-d_tree)

# Issues in designing KNN algorithm (Modeling)

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM
- Learning: Just store all the training examples
- Prediction for a new example **x**
  - Find the k *closest* training examples to **x**
  - Construct the label of **x** using these k points.

How to aggregate the information and make the prediction?

# K-Nearest Neighbors

- Prediction for a new example **x**
  - Find the k *closest* training examples to **x**
  - Construct the label of **x** using these k points.
    - For classification: Every neighbor votes on the label.
      Predict the most frequent label among the neighbors.
    - For regression:
      Predict the mean value

Q: other alternatives?

# Issues in designing KNN algorithm (Modeling)

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

How to define distance?

- Prediction for a new example **x**
  - Find the k *closest* training examples to **x**
  - Construct the label of **x** using these k points.

# Distance between instances

- In general, a good place to inject knowledge about the domain
- Behavior of this approach can depend on this
- How do we measure distances between instances?

# Distance between instances

Numeric features, represented as n dimensional vectors

# Distance between instances

## Numeric features, represented as n dimensional vectors

- Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

# Distance between instances

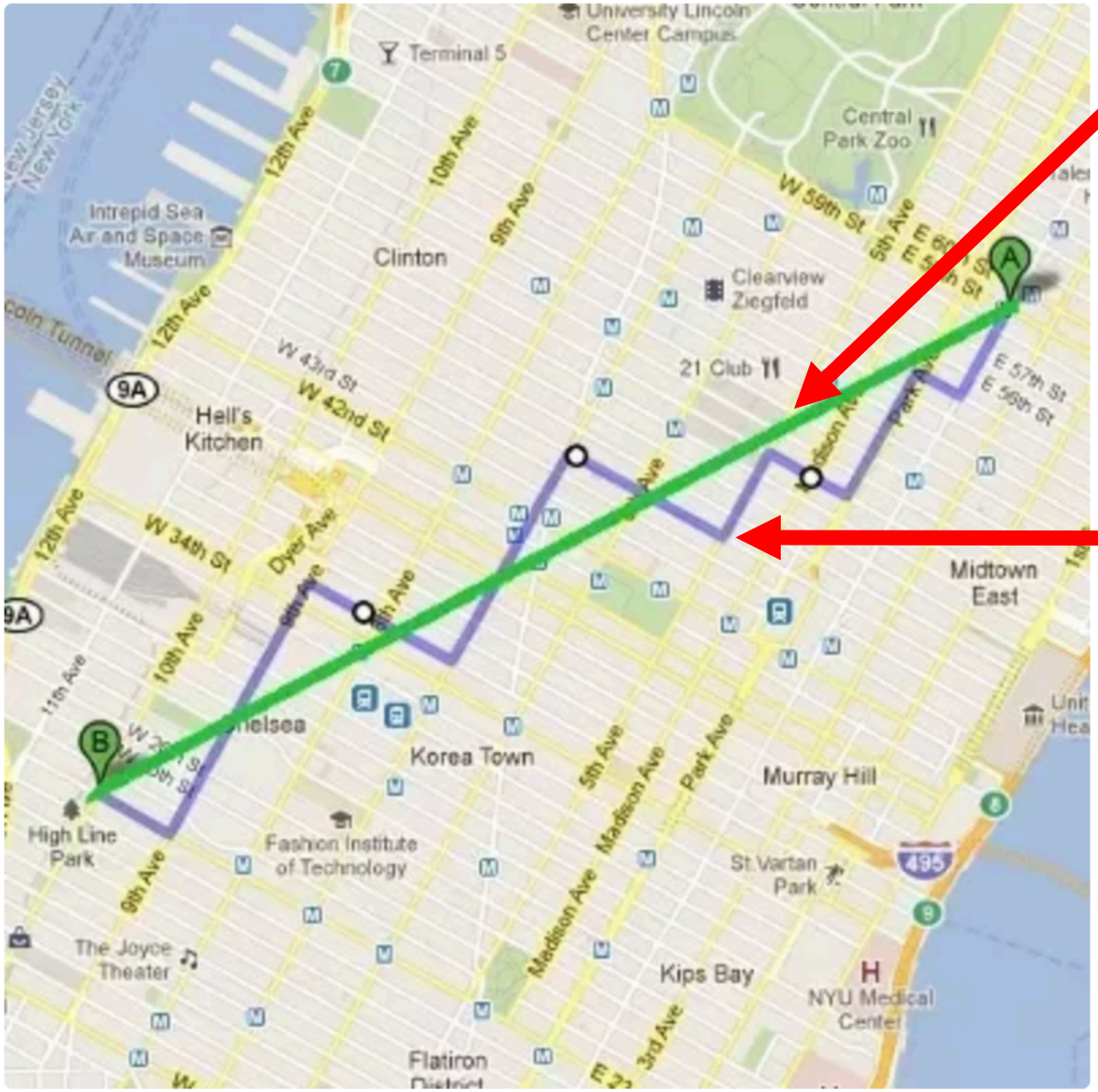
## Numeric features, represented as n dimensional vectors

- Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- Manhattan distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$


Euclidean distance
Manhattan distance
Terminal 5
Central Park Zoo
Intrepid Sea Air and Space Museum
Clinton
Clearview Ziegfeld
21 Club
Hell's Kitchen
Midtown East
Chelsea
Korea Town
Murray Hill
High Line Park
Fashion Institute of Technology
St Vartan Park
The Joyce Theater
Kips Bay
NYU Medical Center
Flatiron
12th Ave
10th Ave
9th Ave
8th Ave
6th Ave
5th Ave
Madison Ave
Park Ave
W 59th St
W 43rd St
W 42nd St
W 34th St
E 57th St
E 59th St
Dyer Ave
9A
495

# Distance between instances

## Numeric features, represented as n dimensional vectors

- Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- Manhattan distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

- $L_p$-norm
  - Euclidean = $L_2$
  - Manhattan = $L_1$

$$||\mathbf{x}_1 - \mathbf{x}_2||_p = \left( \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

# Distance between instances

What about symbolic/categorical features?

# Distance between instances

## Symbolic/categorical features

Most common distance is the *Hamming distance*

- Number of bits that are different
- Or: Number of features that have a different value
- Example:

  $\mathbf{X}_1$: {Shape=Triangle, Color=Red, Location=Left, Orientation=Up}

  $\mathbf{X}_2$: {Shape=Triangle, Color=Blue, Location=Left, Orientation=Down}

  Hamming distance = 2

# Issues in designing KNN algorithm (Modeling)

- Training examples are vectors $\mathbf{x}_i$ associated with a label $y_i$
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM
- Learning: Just store all the training examples

How to choose k and the distance measure?

- Prediction fo ew example **x**
  - Find the k *closest* training examples to **x**
  - Construct the label of **x** using these k points.

# Hyper-parameters in KNN

- Parameters:
  - Choosing K (# nearest neighbors)
  - Distance measurement (e.g., p in the $L_p$-norm)

$$||\mathbf{x}_1 - \mathbf{x}_2||_p = \left( \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

- Those are not specified by the algorithm itself
  - Require empirical studies
  - The best parameter set is task/dataset-specific.

# Recap: Turning parameters by using a validation set

**Training data (set)**

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{N}}, y_{\text{N}})\}$
- They are used for learning $h(\cdot)$

**Test (evaluation) data**

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{M}}, y_{\text{M}})\}$
- They are used for assessing how well $h(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

**Development (or validation) data**

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{L}}, y_{\text{L}})\}$
- They are used to optimize hyperparameter(s).

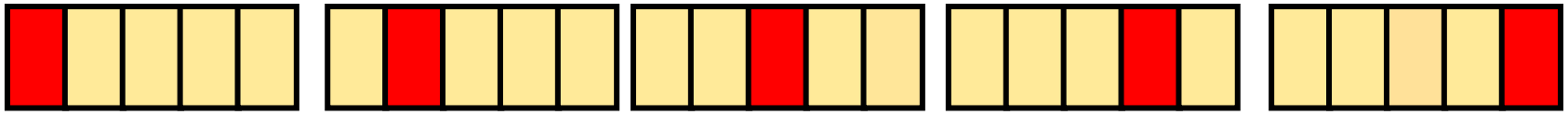
Training data, validation and test data should *not* overlap!

# Recap: Recipe of train/dev/test

- For each possible value of the hyper-parameter (e.g., K = 1,2, 3,…,10)
  - Train a model using $D^{TRAIN}$
  - Evaluate the performance on $D^{DEV}$
- Choose the model parameter with the best performance on $D^{DEV}$
- (optional) Re-train the model on $D^{TRAIN} \cup D^{DEV}$ with the best parameter set
- Evaluate the model on $D^{TEST}$

# N-fold cross validation

- Split data into N equal-sized parts



- Train and test N different classifiers
- Report average accuracy and standard deviation of the accuracy
- If N == datasize, we called it leave-one-out cross validation

# Finding parameters based on cross validation

- Given $D^{TRAIN}$ and $D^{TEST}$, for each possible value of the hyper-parameter (e.g., K = 1,2, 3,…,10)
  - Conduct cross validation on $D^{TRAIN}$ with parameter K
- Choose the model parameter with the best cross validation performance
- Re-train the model on $D^{TRAIN}$ with the best parameter set
- Evaluate the model on $D^{TEST}$

# Put together (K=1, distance= Euclidean)

- Training: Store the entire training set.
- Test:

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathsf{nn}(\boldsymbol{x})}$$

where $\mathsf{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$, i.e., the index to one of the training instances

$$\mathsf{nn}(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \arg\min_{n \in [\mathsf{N}]} \sum_{d=1}^{\mathsf{D}} (x_d - x_{nd})^2$$

$$y = h(\boldsymbol{x}) = y_{\mathsf{nn}(\boldsymbol{x})}$$

# Other tricks

- In practice, use an odd K. Why?
  - To break ties
- Neighbors' labels could be weighted by their distance
  - Related to Kernel method
- Feature normalization could be important. Why?
  - Different features could have different scales (weight, height, etc); but the distance weights them equally
  - Often, good idea to center the features to make them zero mean and unit standard deviation.

# Example: Feature normalization

- Classify Iris with two features

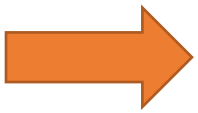
**Training data**

| ID (n) | petal width ($x_1$) | sepal length ($x_2$) | category ($y$) |
|---|---|---|---|
| 1 | 0.2 | 5.1 | setosa |
| 2 | 1.4 | 7.0 | versicolor |
| 3 | 2.5 | 6.7 | virginica |

- Flower with unknown category

petal width $= 1.8$ and sepal width $= 6.4$

Calculating distance $= \sqrt{(x_1 - x_{n1})^2 + (x_2 - x_{n2})^2}$



| ID | distance |
|---|---|
| 1 | 1.75 |
| 2 | 0.72 |
| 3 | 0.76 |

# Example: Feature normalization

- Change units of $x_2$ from cm to mm

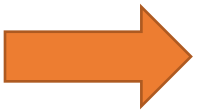
**Training data**

| ID (n) | petal width ($x_1$) | sepal length ($x_2$) | category ($y$) |
|---|---|---|---|
| 1 | 0.2 | 51 | setosa |
| 2 | 1.4 | 70 | versicolor |
| 3 | 2.5 | 67 | virginica |

- Flower with unknown category

petal width $= 1.8$ and sepal width $= 64$

Calculating distance $= \sqrt{(x_1 - x_{n1})^2 + (x_2 - x_{n2})^2}$



| ID | distance |
|---|---|
| 1 | 13 |
| 2 | 6 |
| 3 | 3 |

# Preprocess data

- Normalize data to have zero mean and unit standard deviation in each dimension

$$\bar{x}_d = \frac{1}{\mathsf{N}} \sum_n x_{nd}, \qquad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

# Where are we?

- What is the k-NN algorithm
- How to define distance between two points in the vector space
- Recap: parameter tuning
- Decision boundary of KNN
- Advantage/disadvantage of KNN
- The curse of the dimensionality
  - Why/when KNN may fail

# The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

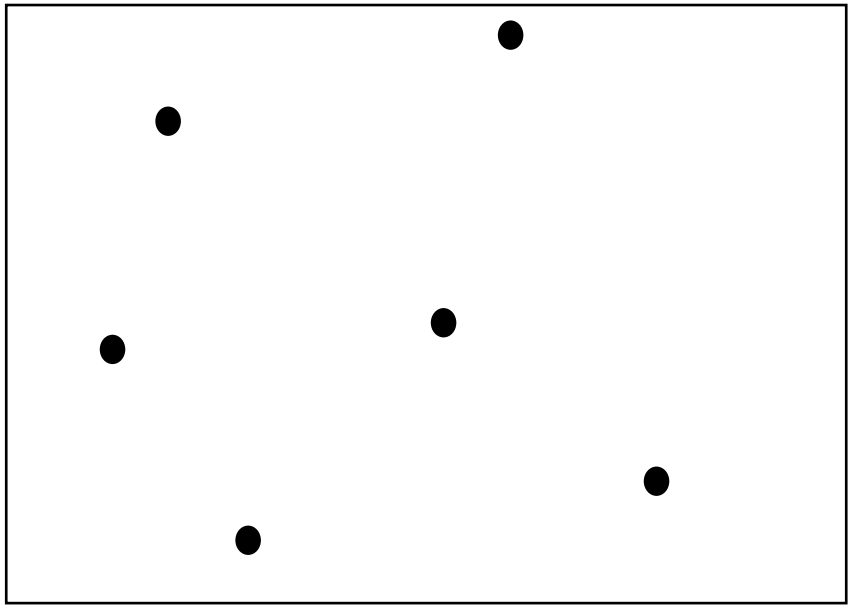
# The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

- **No,** it never forms an explicit hypothesis

But we can still ask: Given a training set what is the implicit function that is being computed

# The Voronoi Diagram

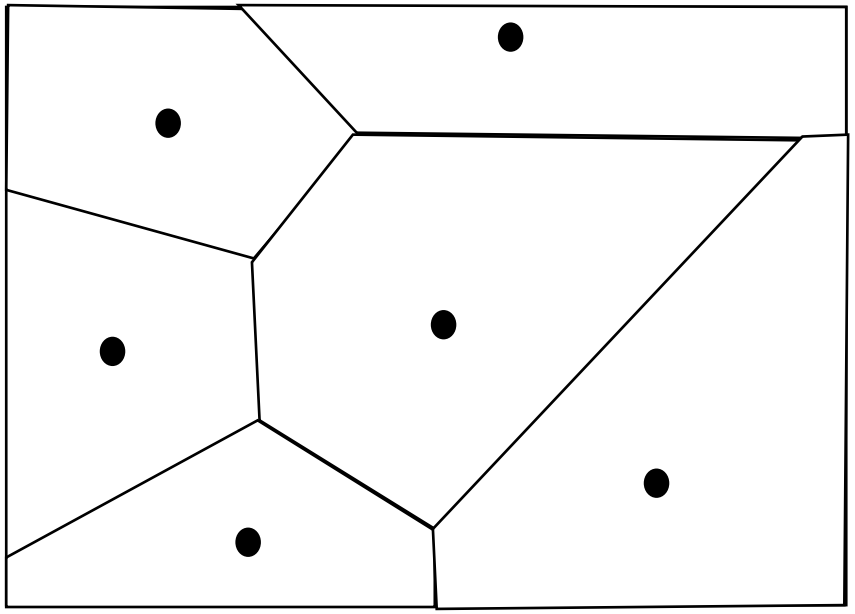


For any point **x** in a training set S,
the Voronoi Cell of **x** is a polytope consisting of all points closer to **x** than any other points in S

The Voronoi diagram is the union of all Voronoi cells

- Covers the entire space

# The Voronoi Diagram (w/ Euclidean distance)

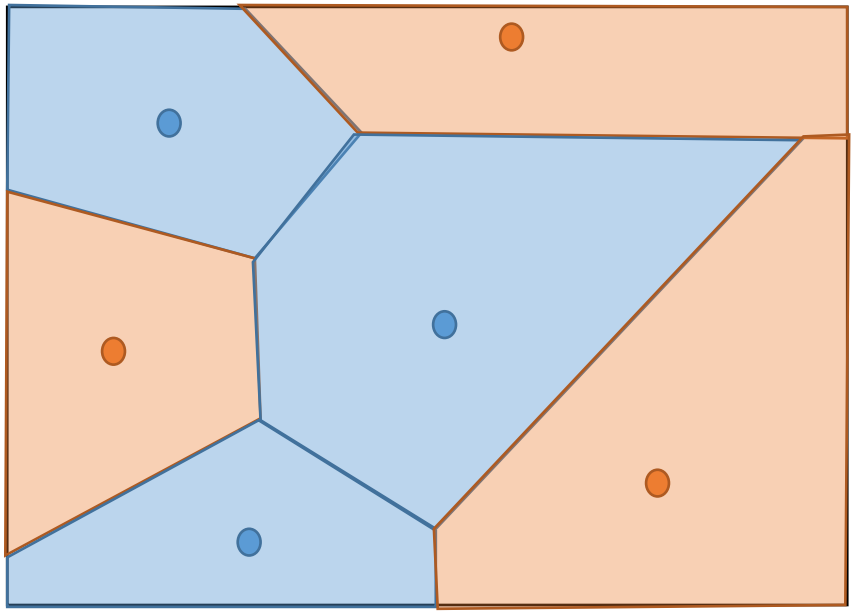


For any point **x** in a training set S,
the Voronoi Cell of **x** is a polytope consisting of all points closer to **x** than any other points in S

The Voronoi diagram is the union of all Voronoi cells

- Covers the entire space

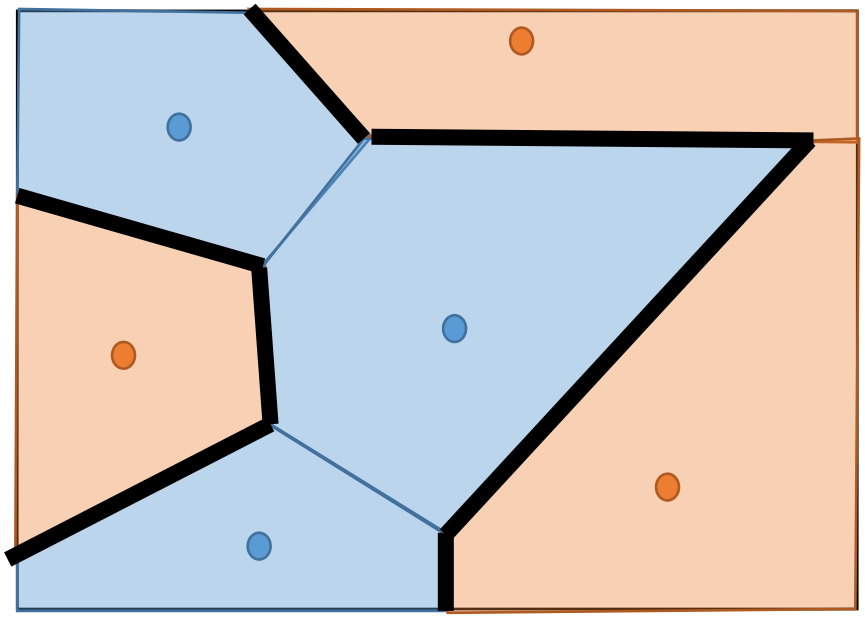
# Voronoi diagrams of training examples



Voronoi diagrams colored with the output label

Picture uses Euclidean distance with 1-nearest neighbor.

# Decision boundary of 1-NN



What about K-nearest neighbors?

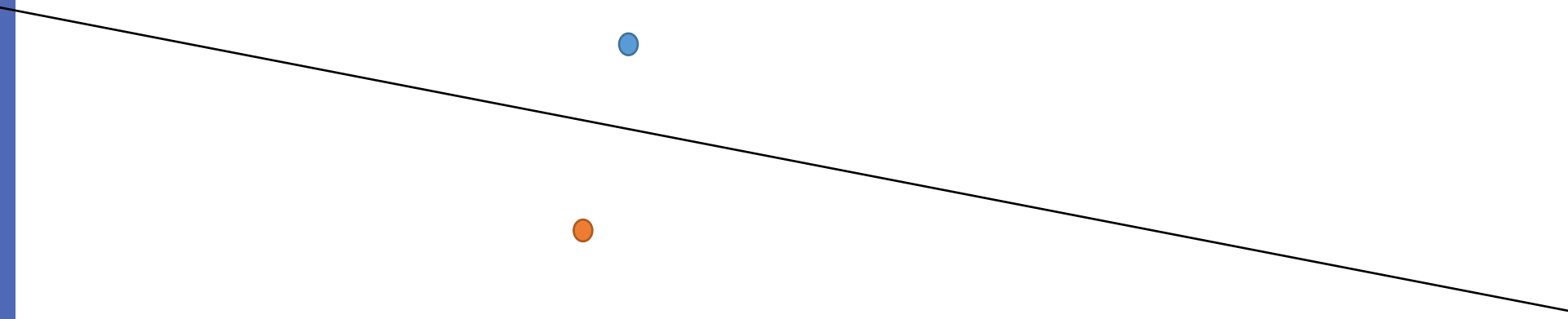
Also partitions the space, but much more complex decision boundary

# Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?

# Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



- A line bisecting the two points

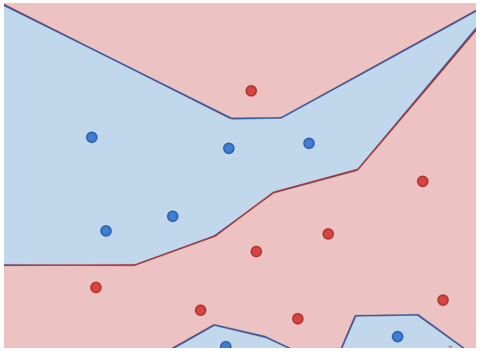
# Instance based learning

- A class of learning methods
  - Learning: Storing examples with labels
  - Prediction: When presented a new example, classify the labels using *similar* stored examples

- K-nearest neighbors algorithm is an example of this class of methods
- Most of the computation is performed only at prediction time
  - Open book v.s. closed book exams

# Advantages

- Training is *very fast*
  - Just adding labeled instances to a list
  - More complex indexing methods can be used, which slow down learning slightly to make prediction faster



- Can learn very complex functions
- We always have the training data
  - For other learning algorithms, after training, we don't store the data anymore. What if we want to do something with it later…

# Disadvantages

- Needs a lot of storage
- Prediction can be slow!
  - Naïvely: O(dN) for N training examples in d dimensions
  - More data will make it slower
- Nearest neighbors are fooled by irrelevant attributes
  - Important and subtle
- Curse of dimensionality
  - Things can go wrong in the high-dimensional spaces
  - What might be intuitive for 2 or 3 dimensions do not always apply to high dimensional spaces

# Of course, irrelevant attributes will hurt

Suppose we have 1000 dimensional feature vectors

- But only 10 features are relevant
- Distances will be dominated by the large number of irrelevant features

# Of course, irrelevant attributes will hurt

Suppose we have 1000 dimensional feature vectors

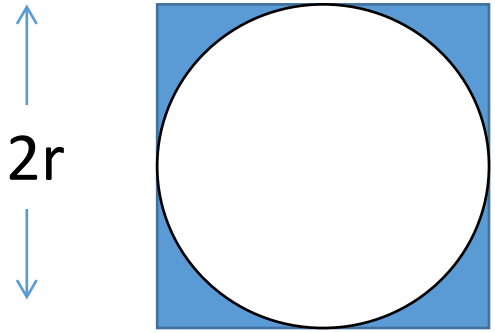
- But only 10 features are relevant
- Distances will be dominated by the large number of irrelevant features

*But even with only relevant attributes, high dimensional spaces behave in odd ways*

# The Curse of Dimensionality

*Example 1*: What fraction of the points in a cube lie outside the sphere inscribed in it?

In two dimensions

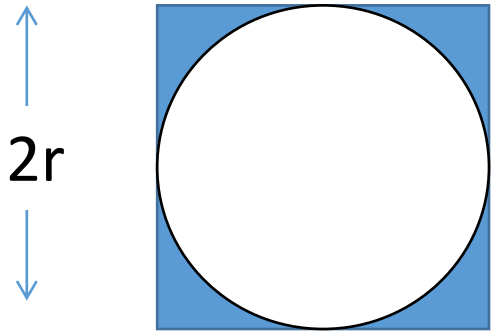



What fraction of the square (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

# The Curse of Dimensionality

*Example 1*: What fraction of the points in a cube lie outside the sphere inscribed in it?

In two dimensions



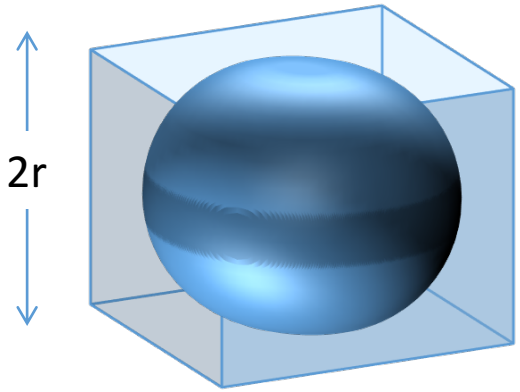

What fraction of the square (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

$$1 - \frac{\pi r^2}{4r^2} = 1 - \frac{\pi}{4}$$

# The Curse of Dimensionality

*Example 1*: What fraction of the points in a cube lie outside the sphere inscribed in it?

In three dimensions




What fraction of the square (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

$$1 - \frac{\frac{4}{3}\pi r^3}{8r^3} = 1 - \frac{\pi}{6}$$

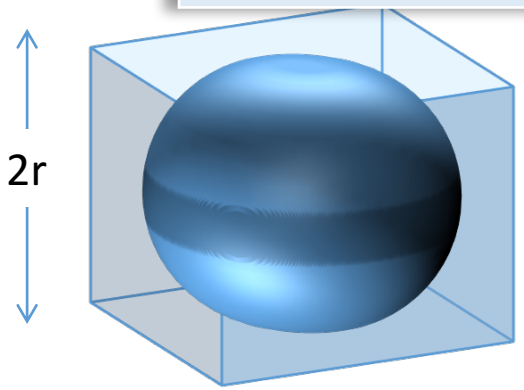
General form:
https://en.wikipedia.org/wiki/Volume_of_an_n-ball

# The Curse of Dimensionality

*Example 1*: What fraction of the points in a cube lie outside the sphere inscribed in it?

As the dimensionality increases, this fraction approaches 1!!

In thre



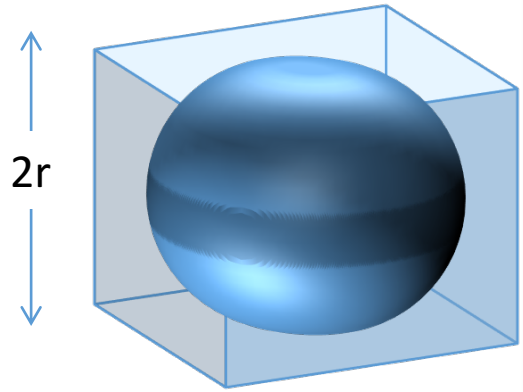

are (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

$$1 - \frac{\frac{4}{3}\pi r^3}{8r^3} = 1 - \frac{\pi}{6}$$

# The Curse of Dimensionality

*Example 1*: W
outside the sp

In three dime




As the dimensionality increases, this fraction approaches 1!!

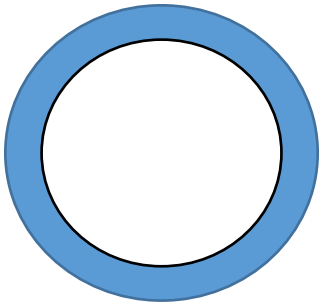
Indication:

In high dimensions, most of the volume of the cube is far away from the center!

# The Curse of Dimensionality

*Example 2*: What fraction of the volume of a unit sphere lies between radius 1 - $\epsilon$ and radius 1?

In two dimensions



What fraction of the area of the circle is in the blue region?

$$\frac{\pi \cdot 1^2 - \pi(1-\epsilon)^2}{\pi \cdot 1^2} = 1 - (1-\epsilon)^2$$

# The Curse of Dimensionality

*Example 2*: What fraction of the volume of a unit sphere lies between radius 1 - $\epsilon$ and radius 1?

In two dimensions $\frac{\pi \cdot 1^2 - \pi(1-\epsilon)^2}{\pi \cdot 1^2} = 1 - (1-\epsilon)^2$

In d dimensions $1 - (1-\epsilon)^d$

When d is large this fraction goes to 1!

In high dimensions, most of the volume of the sphere is far away from the center!

# The Curse of Dimensionality

- Most of the points in high dimensional spaces are far away from the origin!
  - In 2 or 3 dimensions, most points are near the center
  - Need more data to "fill up the space"
- Bad news for k-NN in high dimensional spaces
  - Even if most/all features are relevant, in high dimensional spaces, most points are equally far from each other!
  - "Neighborhood" becomes very large

# Dealing with the curse of dimensionality

- Most "*real-world*" data is not uniformly distributed in the high dimensional space
  - Different ways of capturing the *underlying dimensionality* of the space -- dimensionality reduction
- Feature selection, an art
  - Select features, (e.g., by information gain)
  - Try out different feature sets of different sizes and pick a good set based on a validation set
- Prior knowledge or preferences about the hypotheses can also help

# What you will learn in this lecture

- What is the k-NN algorithm
- How to define distance between two points in the vector space
- Recap: parameter tuning
- Advantage/disadvantage of KNN
- Decision boundary of KNN
- The curse of the dimensionality
  - Why/when KNN may fail

# Exercises

1. What will happen when you choose K to the number of training examples?

2. Suppose you want to build a nearest neighbors classifier to predict whether a beverage is a coffee or a tea using two features: the volume of the liquid (in milliliters) and the caffeine content (in grams). You collect the following data:

| Volume (ml) | Caffeine (g) | Label |
|---|---|---|
| 238 | 0.026 | Tea |
| 100 | 0.011 | Tea |
| 120 | 0.040 | Coffee |
| 237 | 0.095 | Coffee |

What is the label for a test point with Volume = 120, Caffeine = 0.013?

Why might this be incorrect?

How would you fix the problem?

# Exercises

1. What will happen when you choose K to the number of training examples?

   *The label will always be the most common label in the training data*

2. Suppose you want to build a nearest neighbors classifier to predict whether a beverage is a coffee or a tea using two features: the volume of the liquid (in milliliters) and the caffeine content (in grams). You collect the following data:

| Volume (ml) | Caffeine (g) | Label |
|---|---|---|
| 238 | 0.026 | Tea |
| 100 | 0.011 | Tea |
| 120 | 0.040 | Coffee |
| 237 | 0.095 | Coffee |

What is the label for a test point with Volume = 120, Caffeine = 0.013?

*Coffee*

Why might this be incorrect?

*Because Volume will dominate the distance*

How would you fix the problem?

*Rescale the features. Maybe to zero mean, unit variance*