K-Nearest Neighbor
Fall 2022

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.

Learning Objectives

- KNN Algorithms
- Hyper-parameter tuning
  - Train/dev/test
  - N-fold cross-validation
- Decision boundary
- Curse of dimensionality
- Practical concerns -- Data preprocessing

KNN algorithm

- Training examples are vectors $\mathbf{x}_i$ associated with a label y_i
- 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.

Inductive bias of KNN

Definition of inductive bias:

The set of assumptions that the learner uses to predict outputs of unseen inputs

- Label of point (data instance) is similar to the label of nearby points.

Example: Recognizing flowers

Types of Iris: setosa, versicolor, and virginica

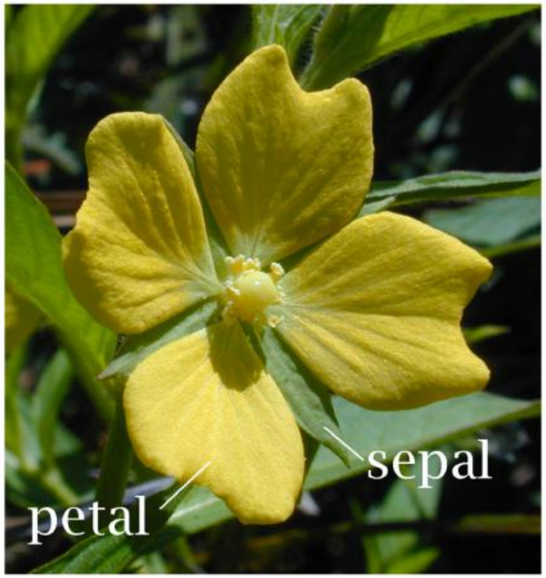
(A)

(B)

(C)

Iris dataset

- Features: the widths and lengths of sepal and petal



Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188

Understand dataset

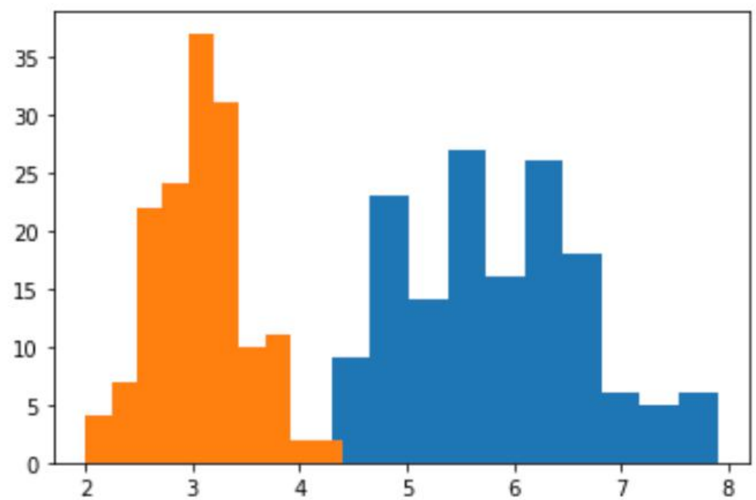
- https://bit.ly/CM146-KNN-IRIS

Fisher's *Iris* Data

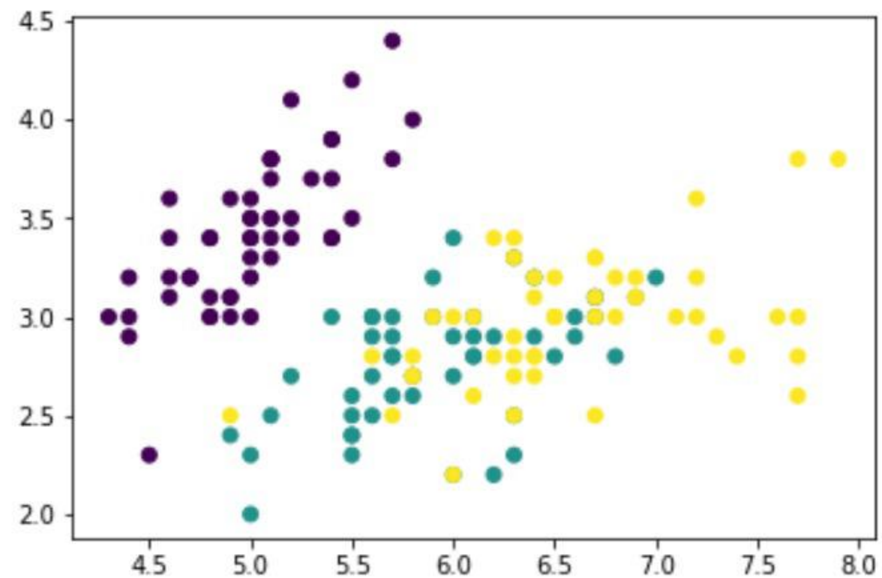
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*

```python
import matplotlib.pyplot as plt
import sklearn
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

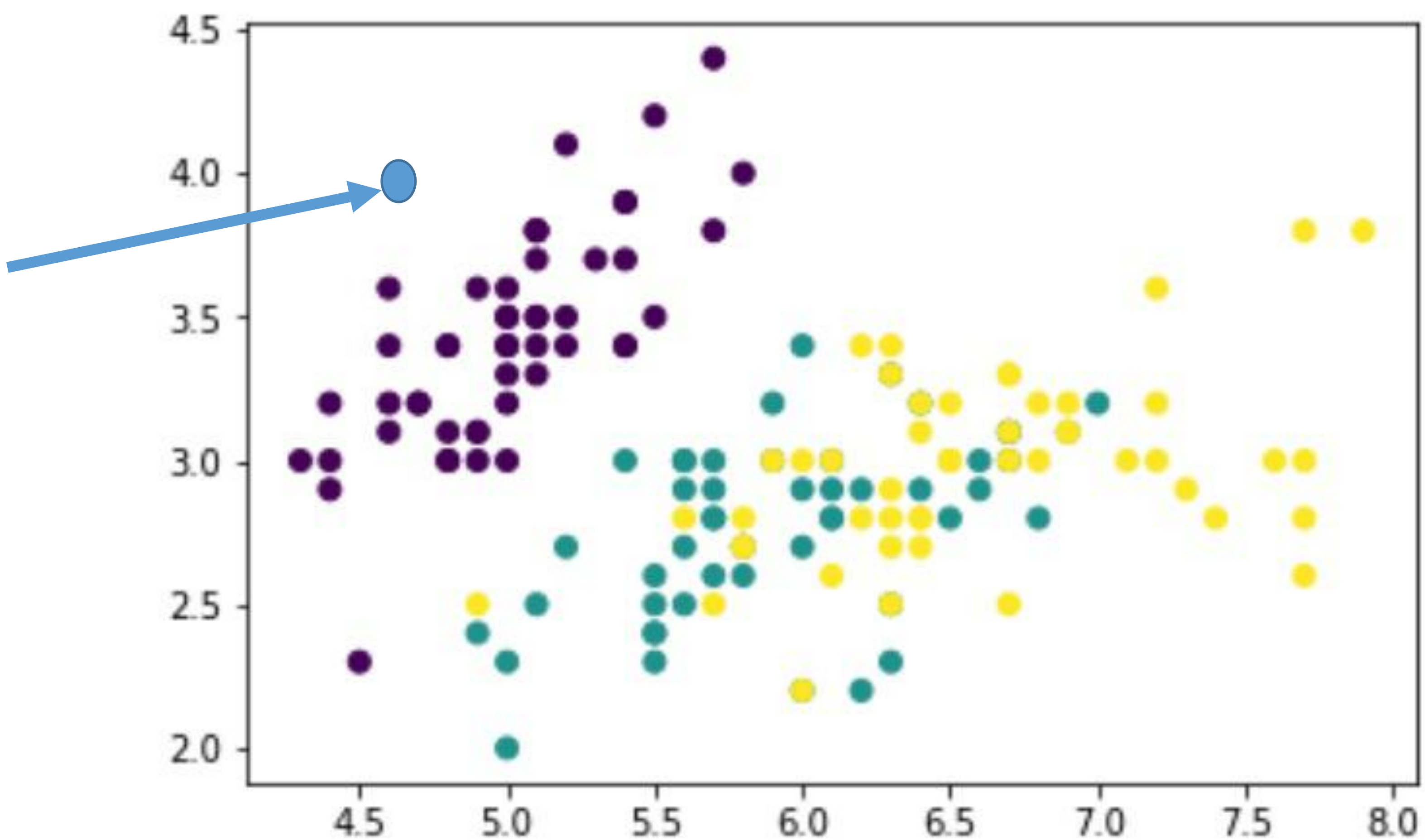
```python
plt.figure()
plt.hist(X[:, 0]);
plt.hist(X[:, 1]);
```




```python
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=Y);
```



```python
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=Y);
```



Example: using 2 features

- Training data

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- Test data

  petal width = 3 and sepal width = 6

- Let's use L1 (Manhattan) distance

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

Example: using 2 features

- Training data

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- Test data

  petal width = 3 and sepal width = 6

Category (y)	L1 Distance
setosa	1+1=2
versicolor	2+0=2
virginica	0+1=1

Hyper-Parameters & Design Choices

Issues in designing KNN algorithm (Modeling)

- Training examples are vectors $\mathbf{x}_i$ associated with a label y_i

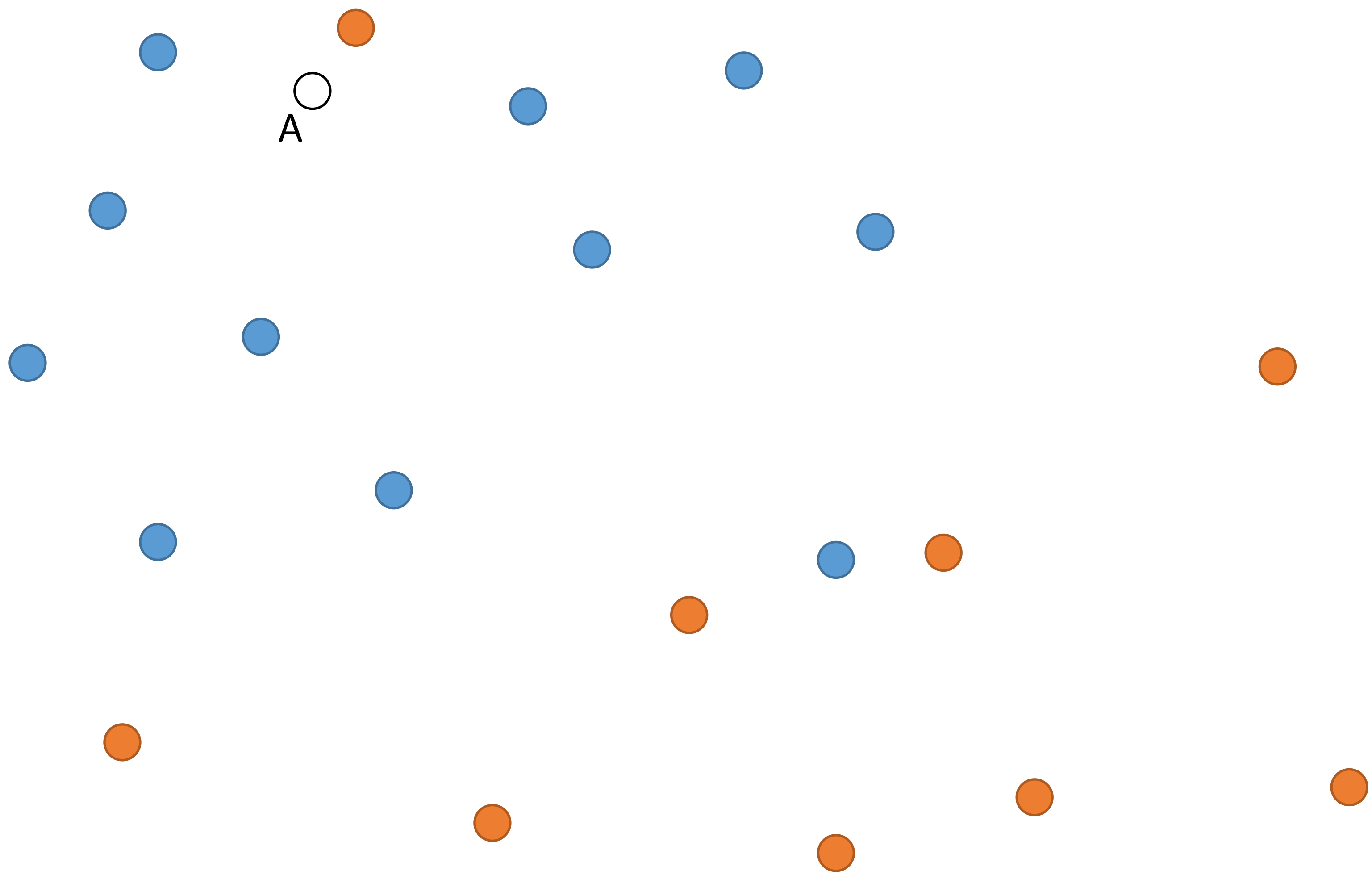
- Learning: Just store all the training examples

> How to choose k and the distance measure?

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

Hyper-Parameter K

What if K is too small?
What if K is too large?



Hyper-parameters in KNN

- Hyper-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.

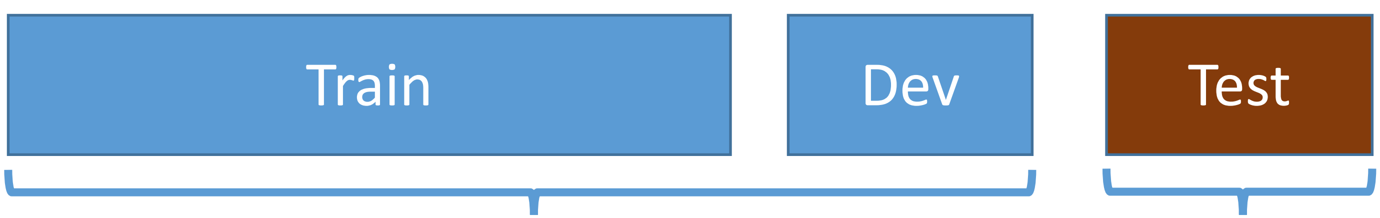
Train/Dev/Test splits

- Split your data into Train/Dev/Test
  - Only use Train and Dev for developing models. Why?
- Train: Data for training models
- Dev (a.k.a. validation set): find the best parameters by evaluating models on dev

Practice exam

- Test: report the performance

Final exam



Labeled data

Reveal labels only in test time

Recipe of train/dev/test

- For each possible value of the hyper-parameter (e.g., M = 1,2, 3,…,10)
  - Train a model using D^{TRAIN}
  - Evaluate the performance on D^{DEV}
- Choose the model with the best performance on D^{DEV}
- Evaluate the final model on D^{TEST}

Recipe of train/dev/test

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

Tradeoff between Train v.s. Dev Size

- Consider having 120 data points; 20 data points are reserved for testing
  - What is the best way to split the remainder?
  - (A) # instances: Train: 95 Dev: 5 ?
  - (B) # instances: Train: 60 Dev: 40 ?

Trade off in Train/Dev splits

- Large Train, small Dev
  (e.g., #train = 95, #dev = 5)

  Result on dev is not representive

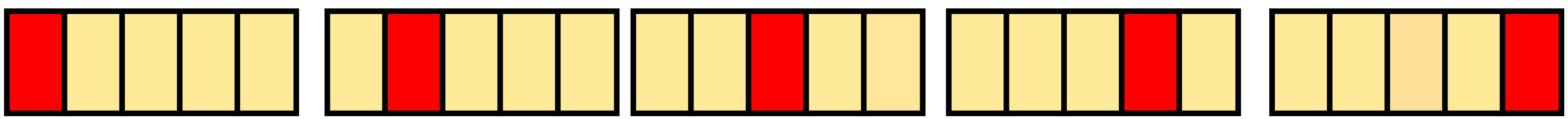
- Small Train, Large Dev
  (e.g., #train = 60, #dev = 40)

  No enough data to train a model

N-fold cross validation

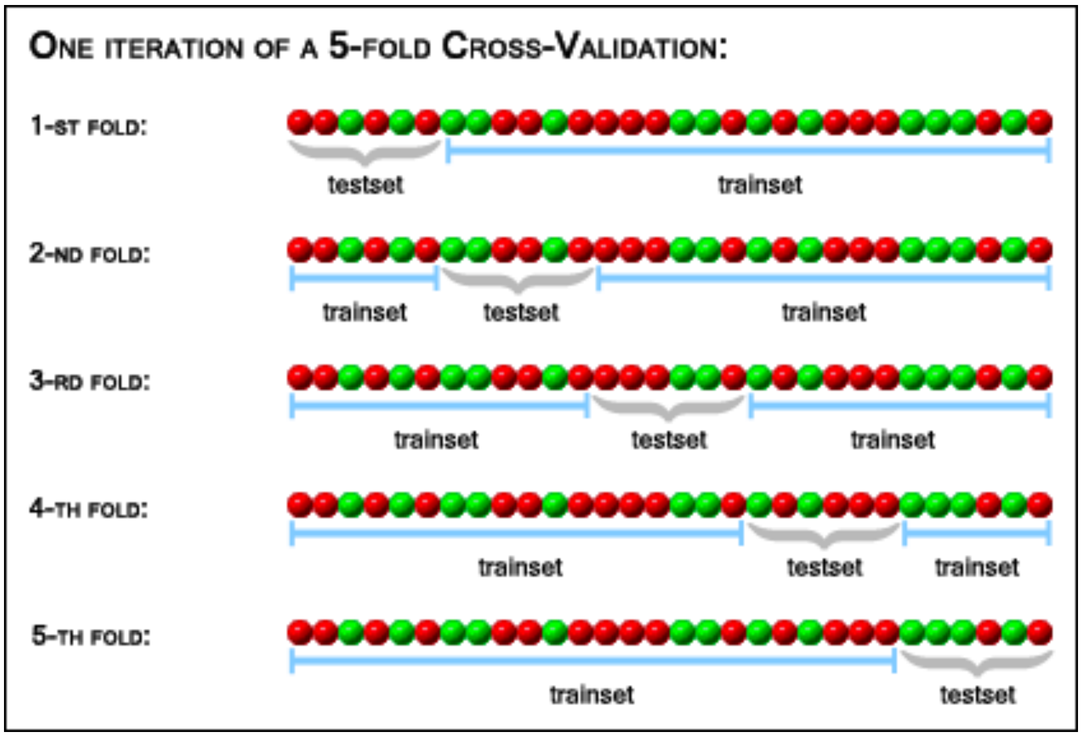
- Instead of a single training-dev split:

- Split data into N equal-sized parts



- Train and test N different classifiers
- Report average accuracy and standard deviation of the accuracy

Example



https://genome.tugraz.at/proclassify/help/pages/XV.html

Parameter 1	Parameter 2
Accuracy: 100%	Accuracy: 100%
Accuracy: 50%	Accuracy: 100%
Accuracy: 50%	Accuracy: 100%
Accuracy: 100%	Accuracy: 100%
Accuracy: 100%	Accuracy: 50%
Avg Accuracy: 80%	Avg Accuracy: 90%

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
- (Optional) Re-train the model on D^{TRAIN} with the best parameter set
- Evaluate the model on D^{TEST}

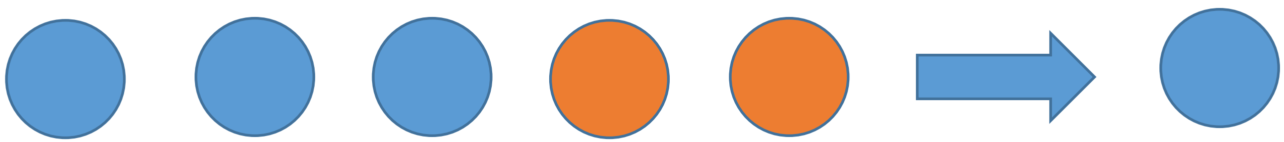
Issues in designing KNN algorithm (Modeling)

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

How to aggregate the information and make the prediction?

Construct the label of **x** using these k points

- Majority vote



To break ties, it's better to use odd K

- Weighted vote
  - Weight by their distances; this is related to kernel methods (will talk about this later)

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 $\mathbf{x}$
  - Find the k *closest* training examples to $\mathbf{x}$
  - Construct the label of $\mathbf{x}$ 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

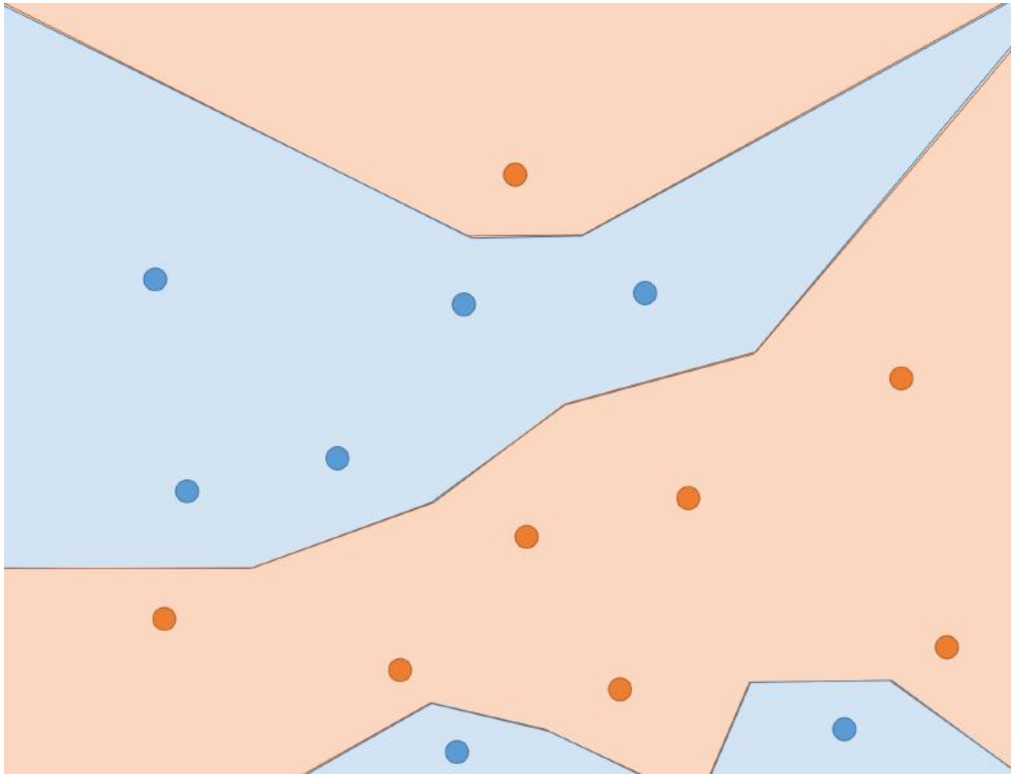
How to store data?

- Prediction for a new example $\mathbf{x}$
  - Find the k *closest* training examples to $\mathbf{x}$
  - 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)

Decision Boundary



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

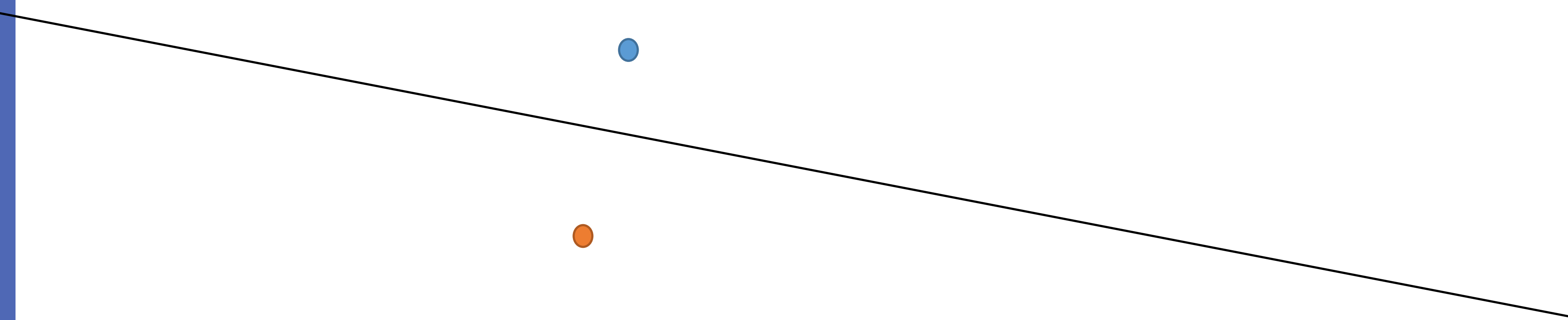
Given a training set what is the implicit function that is being computed

Exercise

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

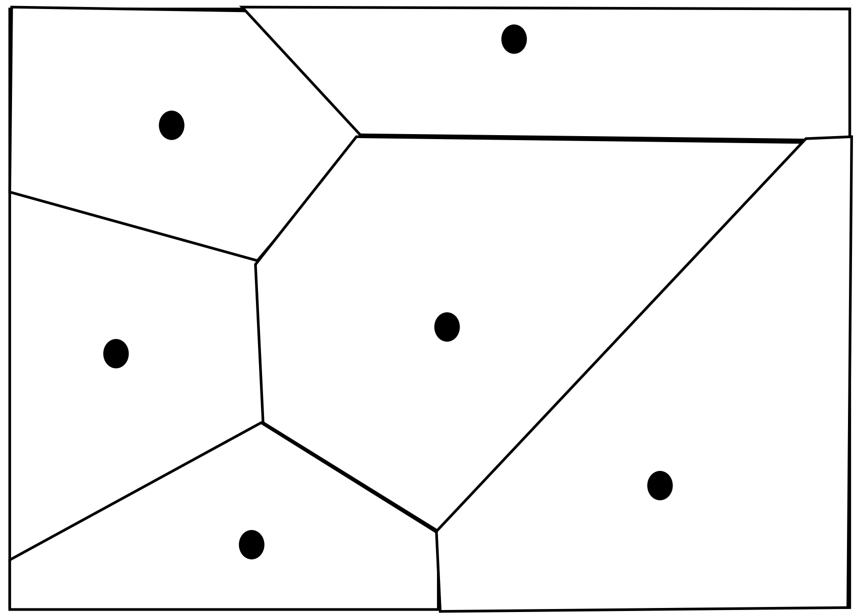
Exercise

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



- A line bisecting the two points

The Voronoi Diagram (w/ Euclidean distance)

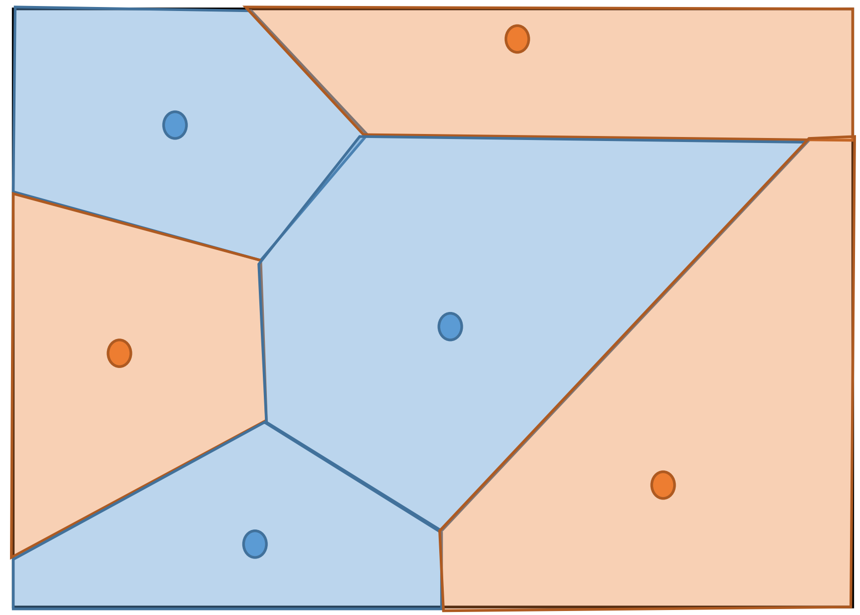


For any point **x** in a training set S,
the Voronoi Cell of **x** is a polytope consisting of all points closer to **x** than any other points in S

The Voronoi diagram is the union of all Voronoi cells

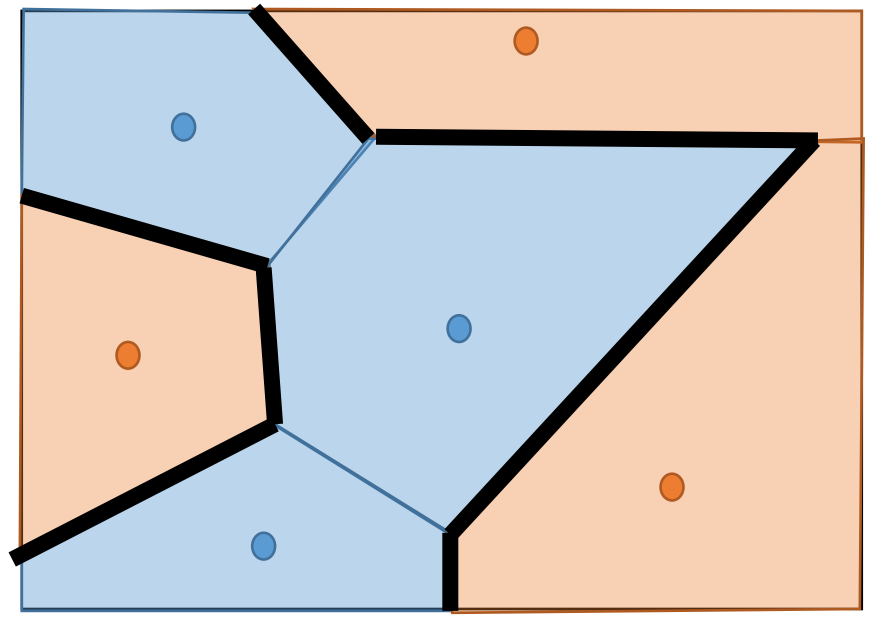
- Covers the entire space

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

https://erikbern.com/2015/10/20/nearest-neighbors-and-vector-models-epilogue-curse-of-dimensionality.html

The Curse of Dimensionality

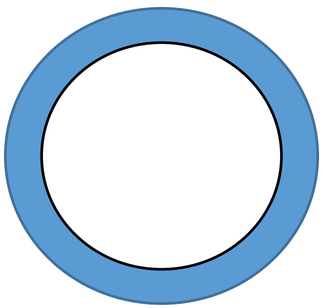
Example : What fraction of the volume of a unit sphere lies between radius 1 - ϵ and radius 1?

volume

radius

$$V = \pi R^2$$

In two dimensions



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

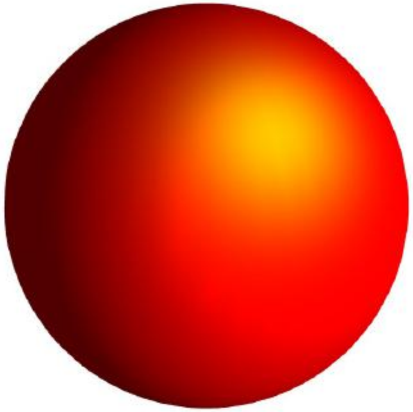
$$\frac{\pi \cdot 1^2 - \pi(1-\epsilon)^2}{\pi \cdot 1^2} = 1 - (1-\epsilon)^2$$

The Curse of Dimensionality

Example : What fraction of the volume of a unit sphere lies between radius 1 - ϵ and radius $1?\text{Type equation here.}$

In three dimensions

$$V = \frac{4\pi}{3} R^3$$



$$\frac{\frac{4\pi}{3} 1^3 - \frac{4\pi}{3} (1-\epsilon)^3}{\frac{4\pi}{3} 1^3} = 1 - (1-\epsilon)^3$$

The Curse of Dimensionality

Example : What fraction of the volume of a unit sphere lies between radius 1 - ϵ 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 $$1 - (1-\epsilon)^d$$

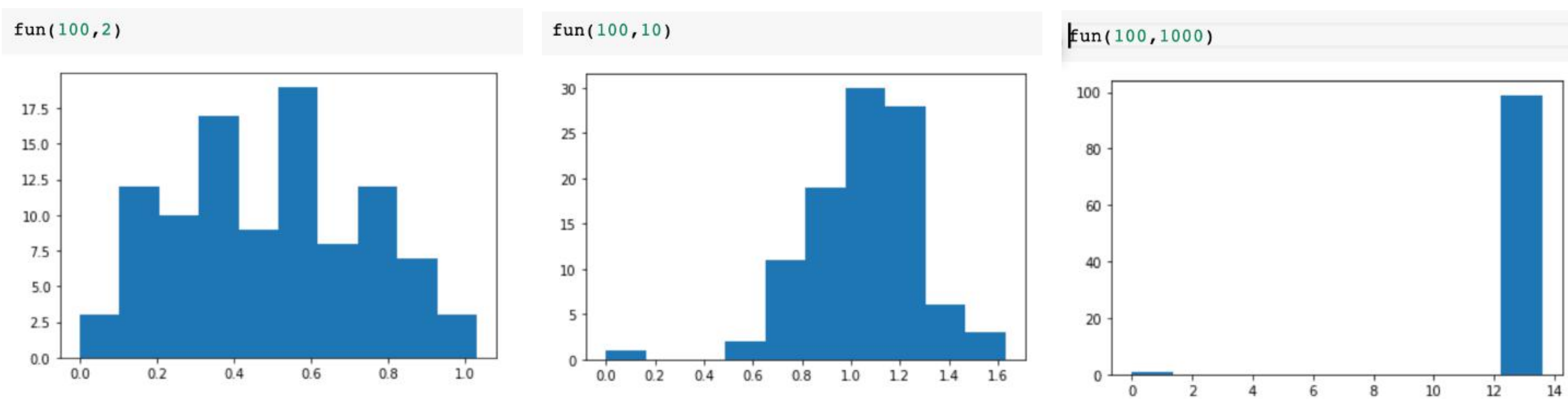
When d is large this fraction goes to 1!

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

Demo

- https://bit.ly/CM146-KNN-dim

```
def fun(num, dim):
  X =  np.random.rand(num, dim)
  a = [np.linalg.norm(X[0,:]-X[i,:]) for i in range(num)]
  plt.hist(a)
```



The Curse of Dimensionality

- Most of the points in high dimensional spaces are far away from the origin!
  - 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 away from each other!

Dealing with the curse of dimensionality

- Most "*real-world*" data is not uniformly distributed in the high dimensional space
  - Capturing the *underlying dimensionality* of the space -- dimensionality reduction

Data Preprocessing

Image from https://twitter.com/cat_cooking/status/858502449149218816

Example: using 2 features

- Training data (length in cm)

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- Test data
  petal width = 3 and sepal width = 6
- Let's use L1 (Manhattan) distance

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

Example: using 2 features

- Training data (length in cm)

ID	Petal Width	Sepal Length	Category (y)
1	4	5	setosa
2	1	6	versicolor
3	3	5	virginica

- Test data

  petal width = 3 and sepal width = 6

Category (y)	L1 Distance
setosa	1+1=2
versicolor	2+0=2
virginica	0+1=1

Example: using 2 features

1cm=10mm

- Training data

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	4	50	setosa
2	1	60	versicolor
3	3	50	virginica

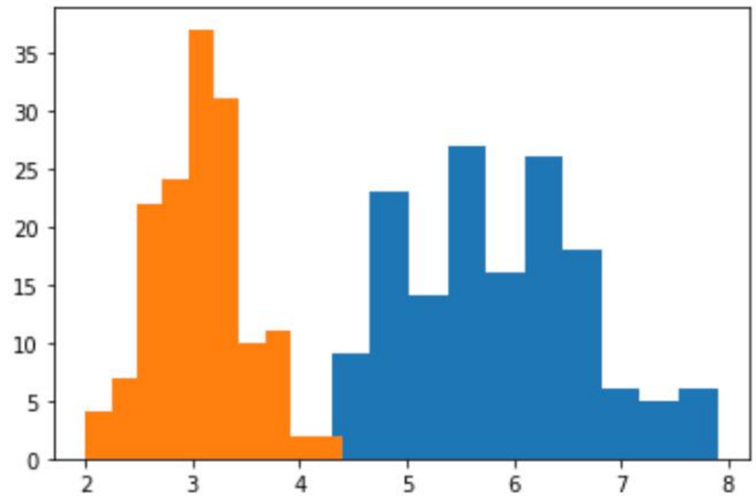
- Test data

  petal width = 3 and sepal width = 60

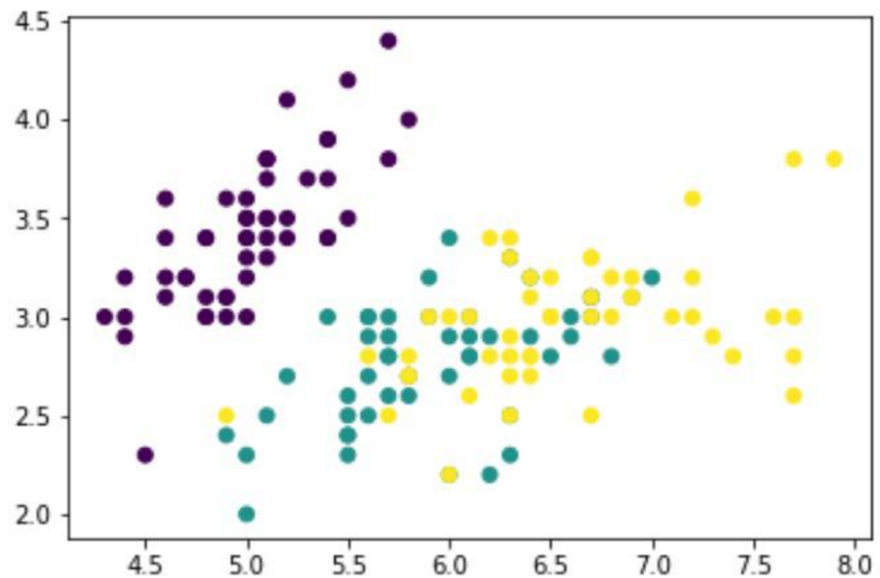
Category (y)	L1 Distance
setosa	1+10=11
versicolor	2+0=2
virginica	0+10=10

```
import matplotlib.pyplot as plt
import sklearn
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

```
plt.figure()
plt.hist(X[:, 0]);
plt.hist(X[:, 1]);
```




```
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=Y);
```



Preprocess data

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

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

- Scale the feature accordingly

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

Example: using 2 features

- Training data

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	4	50	setosa
2	1	60	versicolor
3	3	50	virginica
Mean	2.67	53.3	
Std	1.53	5.77	

Example: using 2 features

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	4	50	setosa
2	1	60	versicolor
3	3	50	virginica
Mean	2.67	53.3	
Std	1.53	5.77	

ID	Petal Width (cm)	Sepal Length (mm)	Category (y)
1	0.87	-0.57	setosa
2	-1.09	1.15	versicolor
3	0.22	-0.57	virginica

Test data (after pre-processing):

petal width = 0.22 and sepal width = 1.15

Exercise

1) Draw the decision boundary of 1-NN
2) Draw the decision boundary of 3-NN