Lecture 3:

Hypothesis Space & KNN

Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Eric Eaton (UPenn), who assembled the original slides, Jessica Wu (Harvey Mudd), David Kauchak (Pomona), Dan Roth (Upenn), Sriram Sankararaman (UCLA), whose slides are also heavily used, and the many others who made their course materials freely available online.

Supervised Learning

Input

x$\in\mathcal{X}$

An item **x** drawn from an instance space $\mathcal{X}$

Learned Model

y = g(**x**)

Output

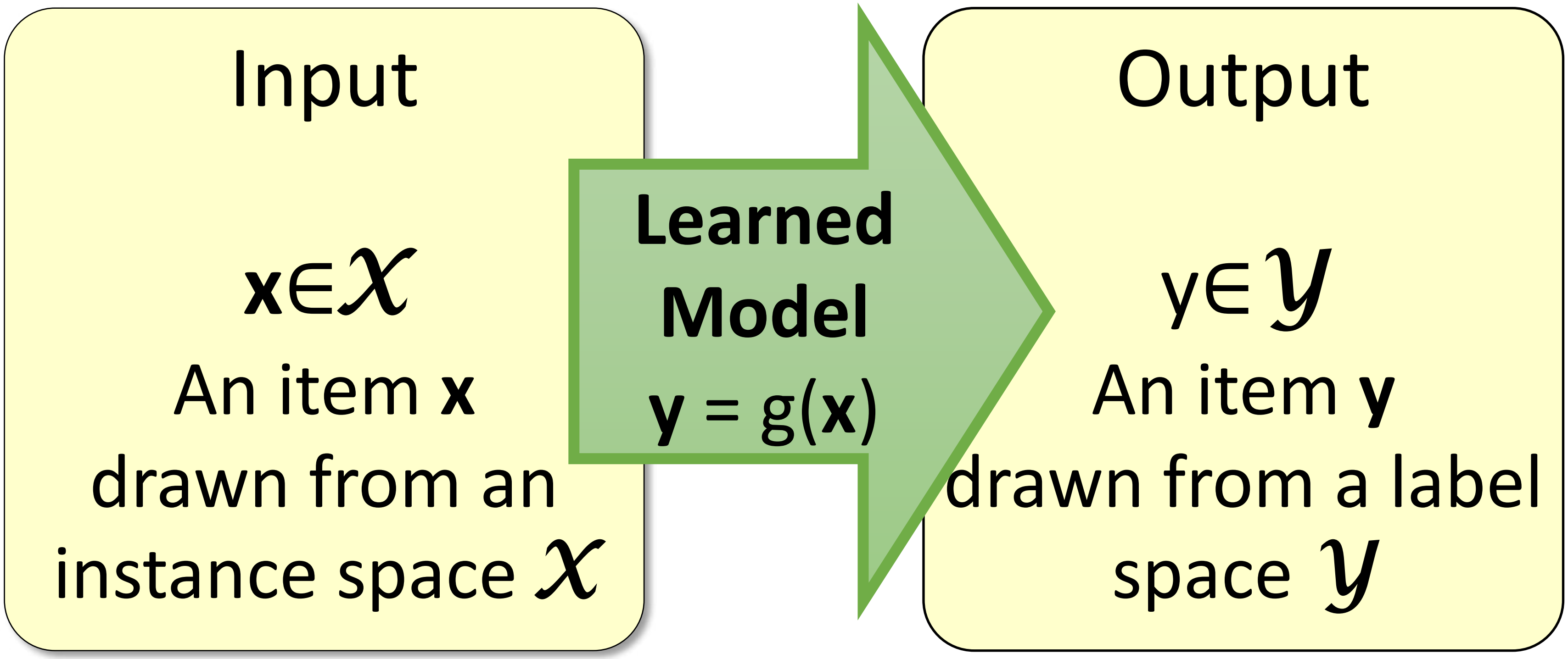
y$\in\mathcal{Y}$

An item **y** drawn from a label space $\mathcal{Y}$

Using supervised learning

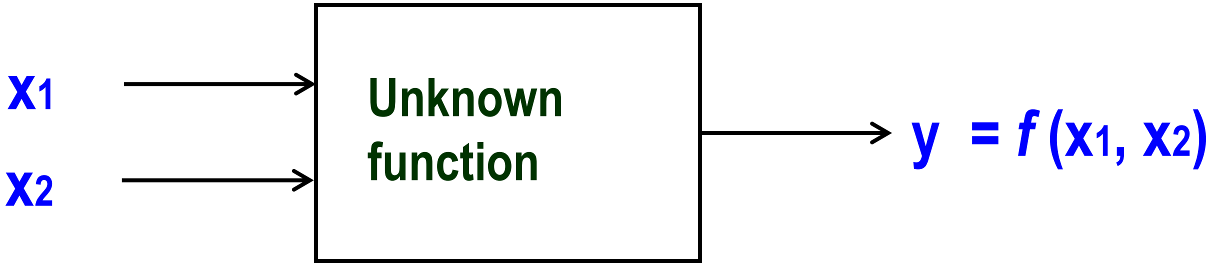
- What is our instance space?
  - Gloss: What kind of features are we using?
- What is our label space?
  - Gloss: What kind of learning task are we dealing with?
- What is our hypothesis space?
  - Gloss: What kind of functions (models) are we learning?
- What learning algorithm do we use?
  - Gloss: How do we learn the model from the labeled data?
- What is our loss function/evaluation metric?
  - Gloss: How do we measure success? What drives learning?

3. The model g(**x**)



- We need to choose what *kind* of model we want to learn

Boolean Function



x_1	x_2	y
0	0	0
0	1	0
1	0	1
1	1	1

Function 1

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

Function 2

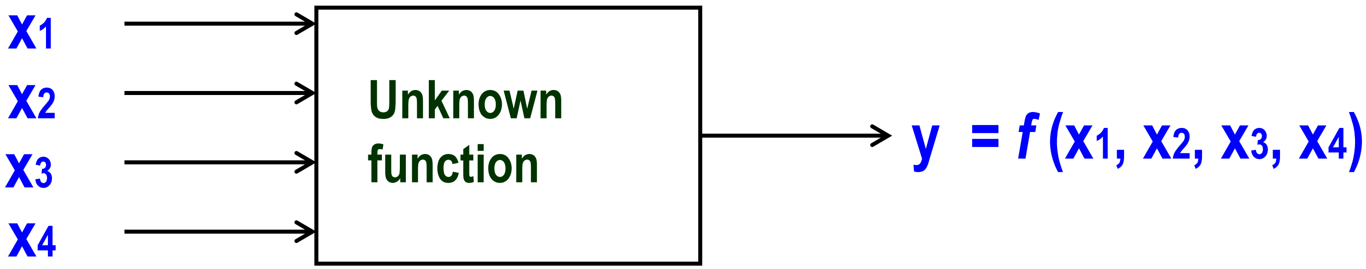
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	0

Function 3

…

Hypothesis Space

A Learning Problem



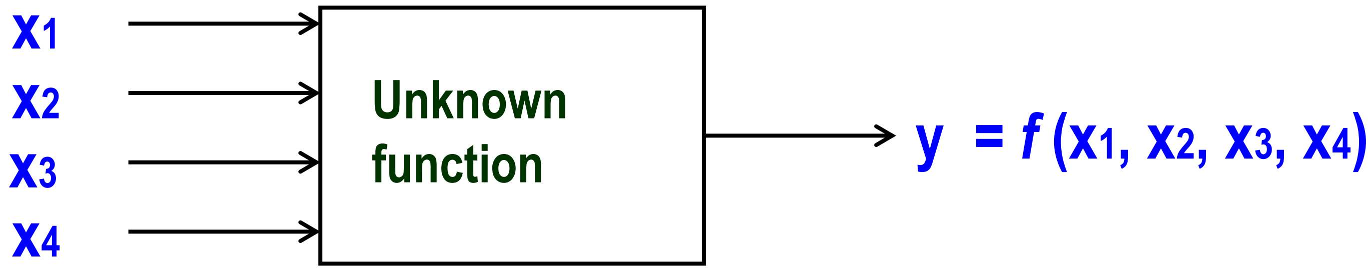
Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

A function g is consistent to a dataset $\mathrm{D} = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall\, i$

Discussion: A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

A function g is consistent to a dataset $\mathrm{D} = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall\, i$

How many possible functions over four features?

How many function is consistent to D on the left

Hypothesis Space

How many possible functions over four features?

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

Example x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	?
0	0	1	1	?
0	1	0	0	?
0	1	0	1	?
0	1	1	0	?
0	1	1	1	?
1	0	0	0	?
1	0	0	1	?
1	0	1	0	?
1	0	1	1	?
1	1	0	0	?
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Hypothesis Space

Complete Ignorance:
There are $2^{16} = 65536$ possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

After observing seven examples we still have 2^9 possibilities for f

Is Learning Possible?
which one is the most likely one?

Example	x_1	x_2	x_3	x_4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
	0	0	1	1	1
	0	1	0	0	0
	0	1	0	1	0
	0	1	1	0	0
	0	1	1	1	?
	1	0	0	0	?
	1	0	0	1	1
	1	0	1	0	?
	1	0	1	1	?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space

Complete Ignorance:
There are 2^{16} = 65536 possible functions over four input features.

We c
corre
poss

After
have 2^9 possibilities for f

Is Learning Possible?

Example	x_1	x_2	x_3	x_4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
					1
					0
					0
					0
					?
					?
					1
					?
					?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

- There are $|Y|^{|X|}$ possible functions f(**x**) from the instance space **X** to the label space **Y.**
- Learners typically consider only a *subset* of the functions from **X** to **Y**, called the hypothesis space **H** . **H** $\subseteq |Y|^{|X|}$

Hypothesis Space (2)

Simple Rules: **conjunctive rules**

of the form $y = x_i \wedge x_j \wedge \ldots \wedge x_k$

e.g., $y = x_2 \wedge x_3$

$y = x_1 \wedge x_2 \wedge X_4$

How large is the hypothesis space?

Hypothesis Space (2)

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Simple Rules: There are only 16 **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=1$		$x_2 \wedge x_3$	
x_1		$x_2 \wedge x_4$	
x_2		$x_3 \wedge x_4$	
x_3		$x_1 \wedge x_2 \wedge x_3$	
x_4		$x_1 \wedge x_2 \wedge x_4$	
$x_1 \wedge x_2$		$x_1 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_3$		$x_2 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_4$		$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	

Hypothesis Space (2)

1	0	0	1	0	**0**
2	0	1	0	0	**0**
3	0	0	1	1	**1**
4	1	0	0	1	**1**
5	0	1	1	0	**0**
6	1	1	0	0	**0**
7	0	1	0	1	**0**

Simple Rules: There are only 16 **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=c$		$x_2 \wedge x_3$	0011 1
x_1	1100 0	$x_2 \wedge x_4$	0011 1
x_2	0100 0	$x_3 \wedge x_4$	1001 1
x_3	0110 0	$x_1 \wedge x_2 \wedge x_3$	0011 1
x_4	0101 1	$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_2$	1100 0	$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_3$	0011 1	$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_4$	0011 1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

No simple rule explains the data. The same is true for **simple clauses**.

Hypothesis Space (3)

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "y = 1 if and only if at least m of the following n variables are 1"

Notation: 2 variables from the set on the left. **Value**: Index of the counterexample.

variables	1-of	2-of	3-of	4-of
{X1}				
{X2}				
{X3}				
{X4}				
{X1,X2}				
{X1, X3}				
{X1, X4}				
{X2,X3}				

variables	1-of	2-of	3-of	4-of
{X2, X4}				
{X3, X4}				
{X1,X2, X3}				
{X1,X2, X4}				
{X1,X3,X4}				
{X2, X3,X4}				
{X1, X2, X3,X4}				

Hypothesis Space (3)

1	0	0	1	0	**0**
2	0	1	0	0	**0**
3	0	0	1	1	**1**
4	1	0	0	1	**1**
5	0	1	1	0	**0**
6	1	1	0	0	**0**
7	0	1	0	1	**0**

m-of-n rules: There are 32 possible rules of the form "y = 1 if and only if at least m of the following n variables are 1"

Notation: 2 variables from the set on the left. **Value**: Index of the counterexample.

variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	-	-	-
$\{x_2\}$	2	-	-	-
$\{x_3\}$	1	-	-	-
$\{x_4\}$	7	-	-	-
$\{x_1, x_2\}$	2	3	-	-
$\{x_1, x_3\}$	1	3	-	-
$\{x_1, x_4\}$	6	3	-	-
$\{x_2, x_3\}$	2	3	-	-

variables	1-of	2-of	3-of	4-of
$\{x_2, x_4\}$	2	3	-	-
$\{x_3, x_4\}$	4	4	-	-
$\{x_1, x_2, x_3\}$	1	3	3	-
$\{x_1, x_2, x_4\}$	2	3	3	-
$\{x_1, x_3, x_4\}$	1	***	3	-
$\{x_2, x_3, x_4\}$	1	5	3	-
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Found a consistent hypothesis.

Views of Learning

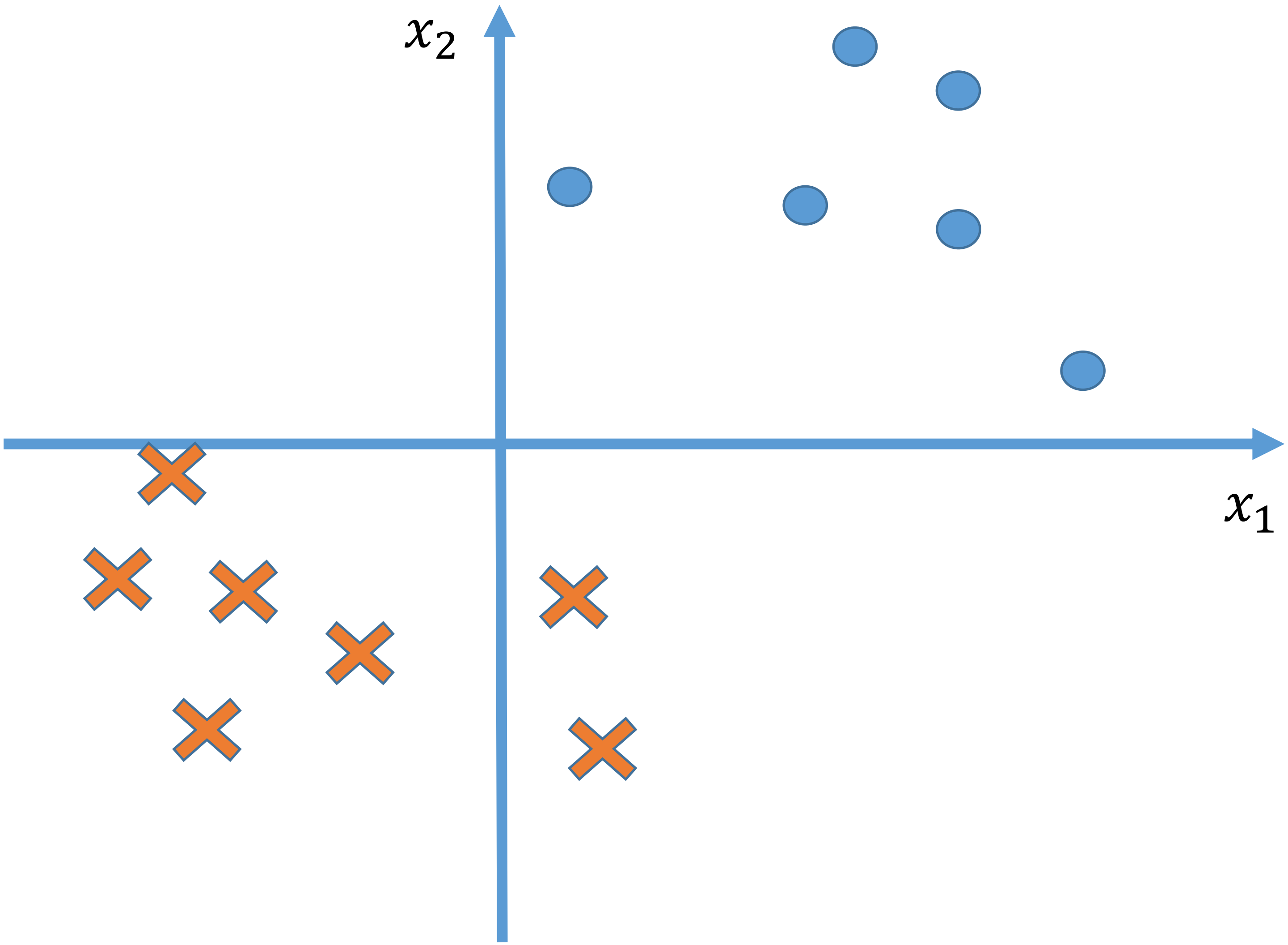
- Learning is the removal of our remaining uncertainty:
- Learning requires guessing a good hypothesis class
  - Start with a small class and enlarge it until it contains an hypothesis that fits the data.
- We could be wrong !
  - Our guess of the hypothesis space could be wrong
    - y=x4 $\wedge$ one-of (x1, x3) is also consistent

General strategies for Machine Learning

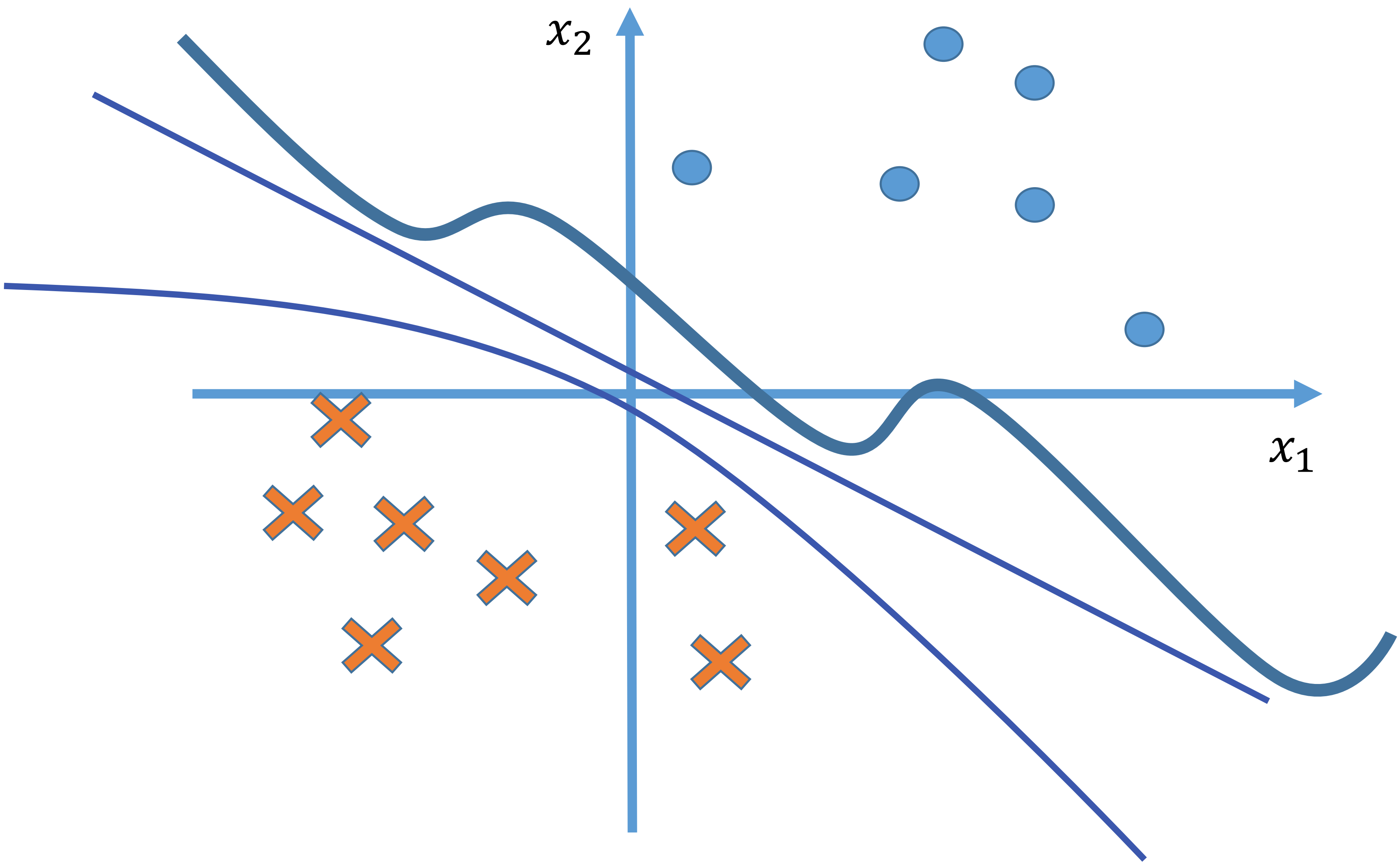
- Develop flexible hypothesis spaces:
  - Decision trees, neural networks, nested collections.
- Develop algorithms for finding the ”best” hypothesis in the hypothesis space, that fits the data
- And, hope that it will generalize well

Hypothesis Space -- Real-Value Features

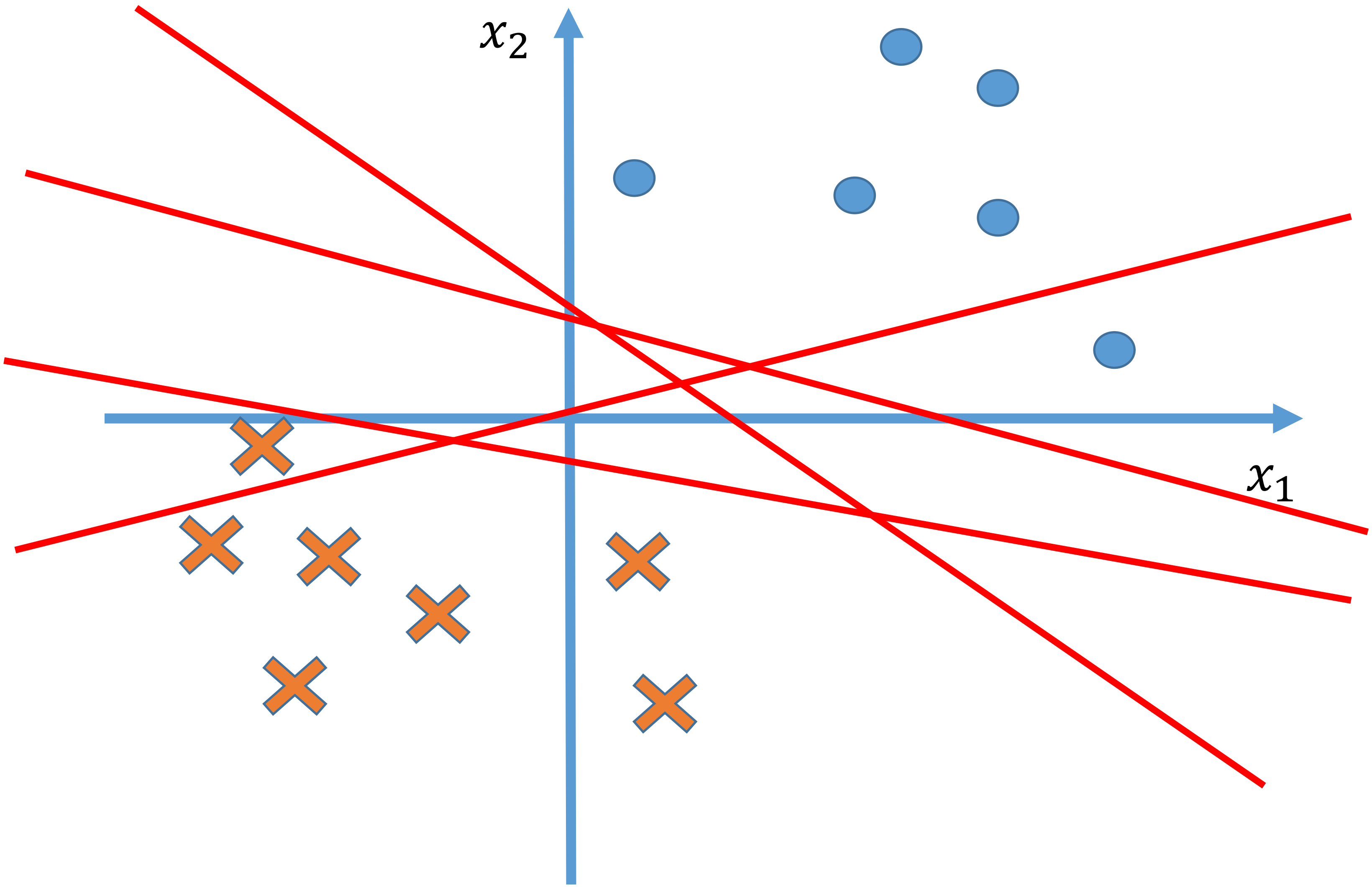
Example problem



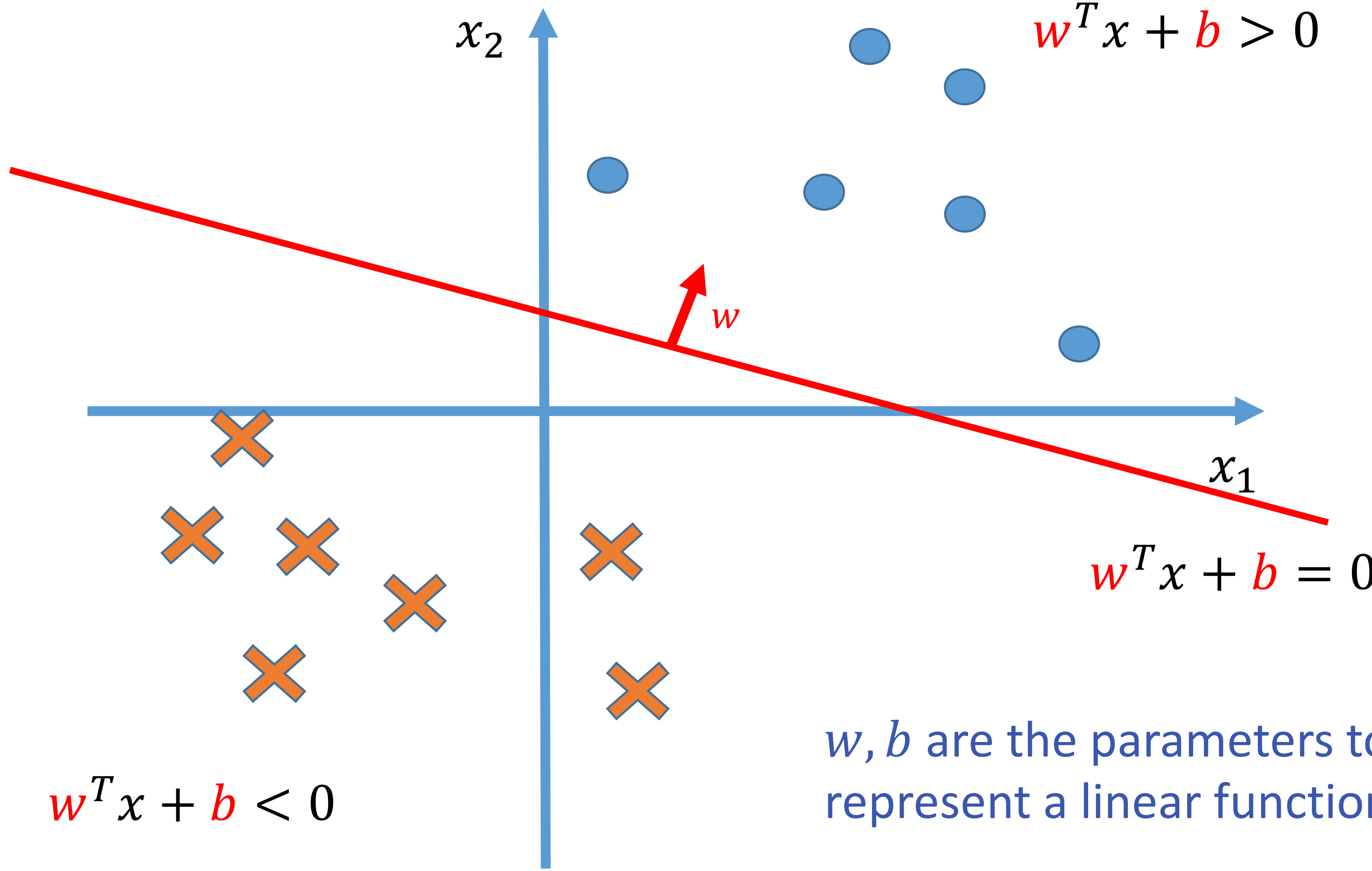
Hypothesis space:



Hypothesis space: linear model



Hypothesis space: linear model



w, b are the parameters to represent a linear function

How to learn?

How can we find a good model from the hypothesis space?

Recap: rule-out

m-of-n rules: There are 32 possible rules of the form "y = 1 if and only if at least m of the following n variables are 1"

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Notation: 2 variables from the set on the left. **Value**: Index of the counterexample.

variables	1-of	2-of	3-of	4-of
{X1}				
{X2}				
{X3}				
{X4}				
{X1,X2}				
{X1, X3}				
{X1, X4}				
{X2,X3}	2	3	-	-

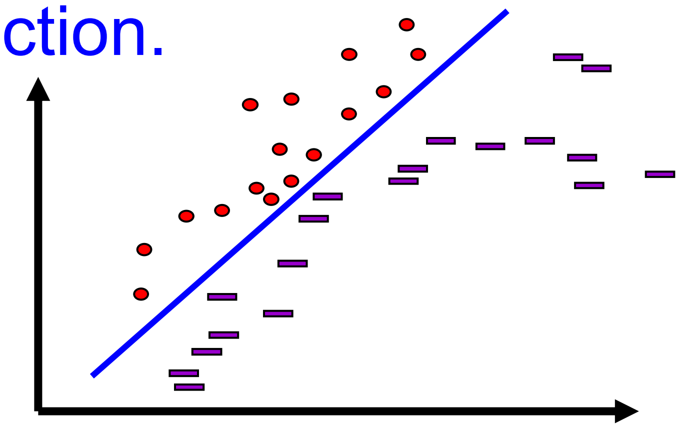
variables	1-of	2-of	3-of	4-of

Learning is the removal of our remaining uncertainty

Found a consistent hypothesis.

How about linear function?

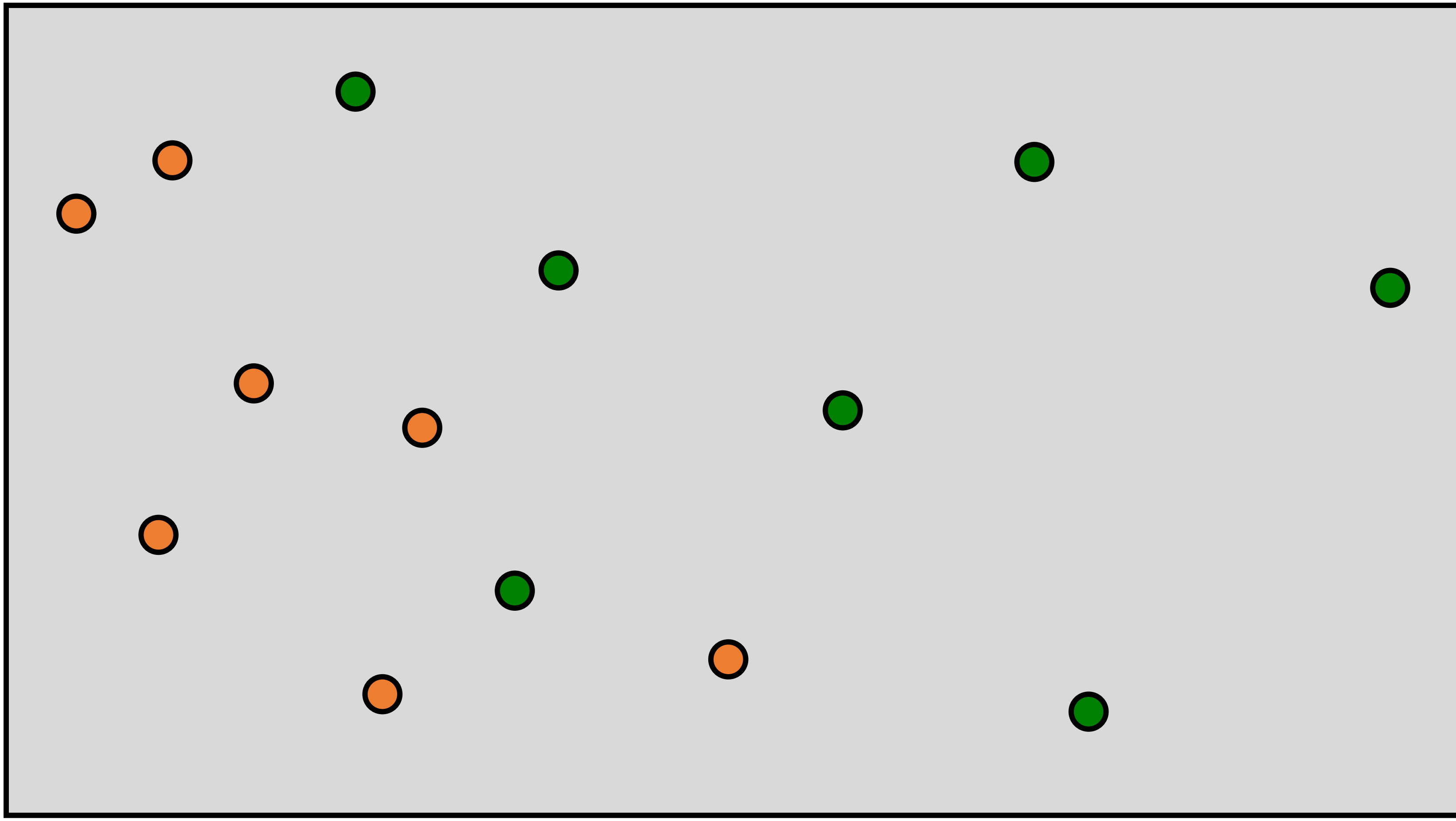
- Challenges
  - The hypothesis space contains infinite # functions
  - Several functions are consistent with the data
- A possibility: Local search
  - Start with a linear threshold function.
  - See how well you are doing.
  - Correct
  - Repeat until you converge.



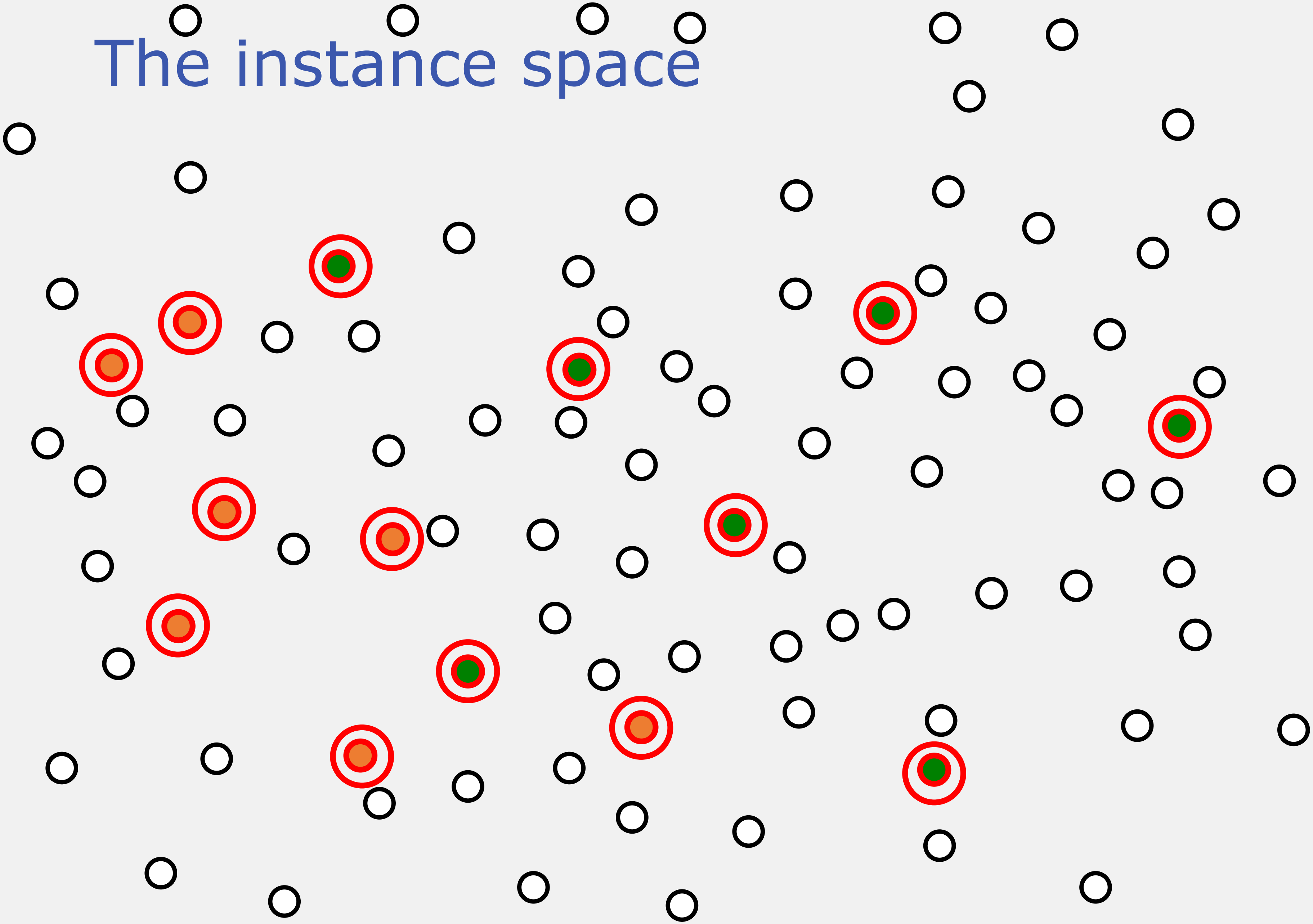
- Optimize a function with calculus

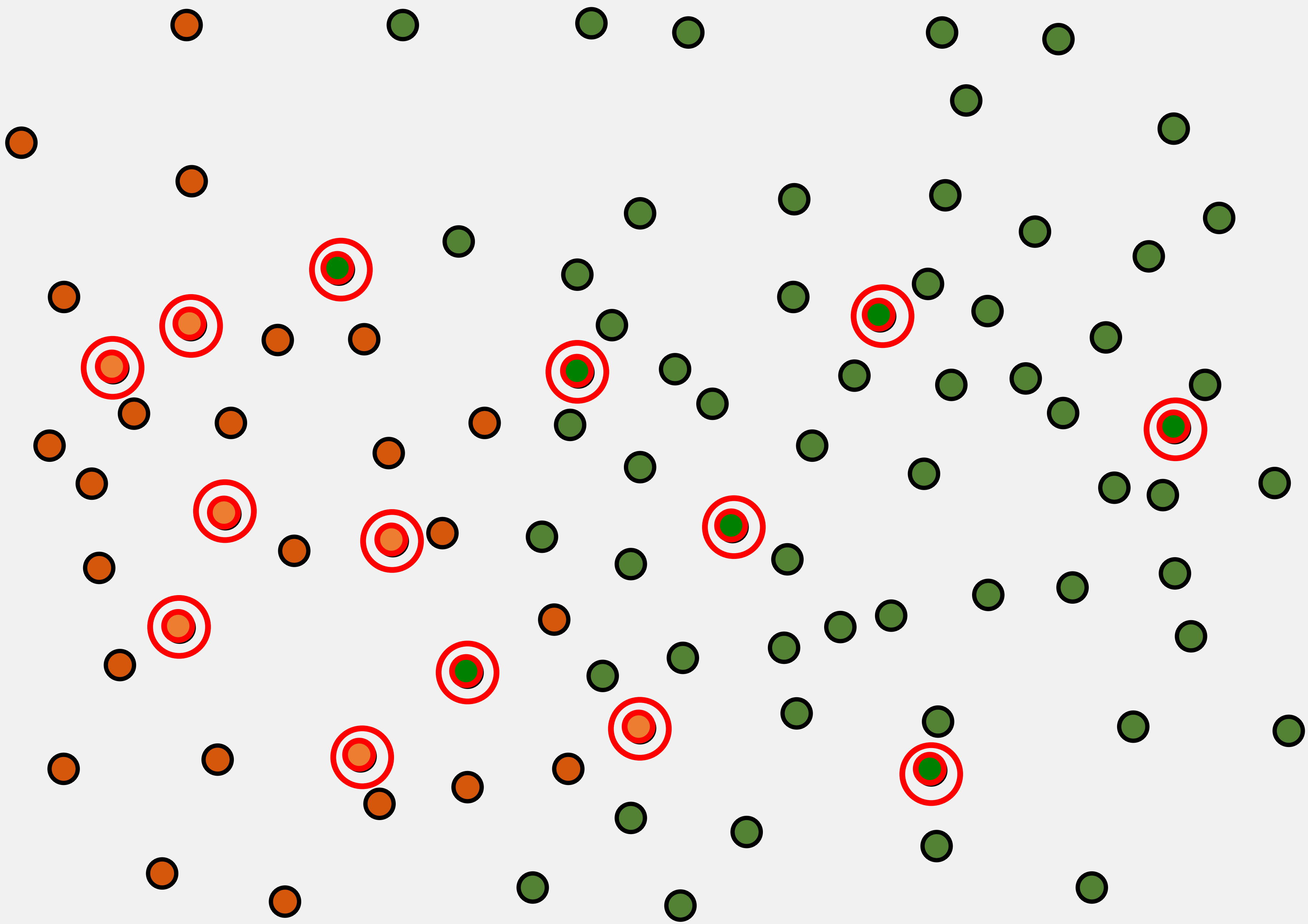
Choose Hypothesis Space

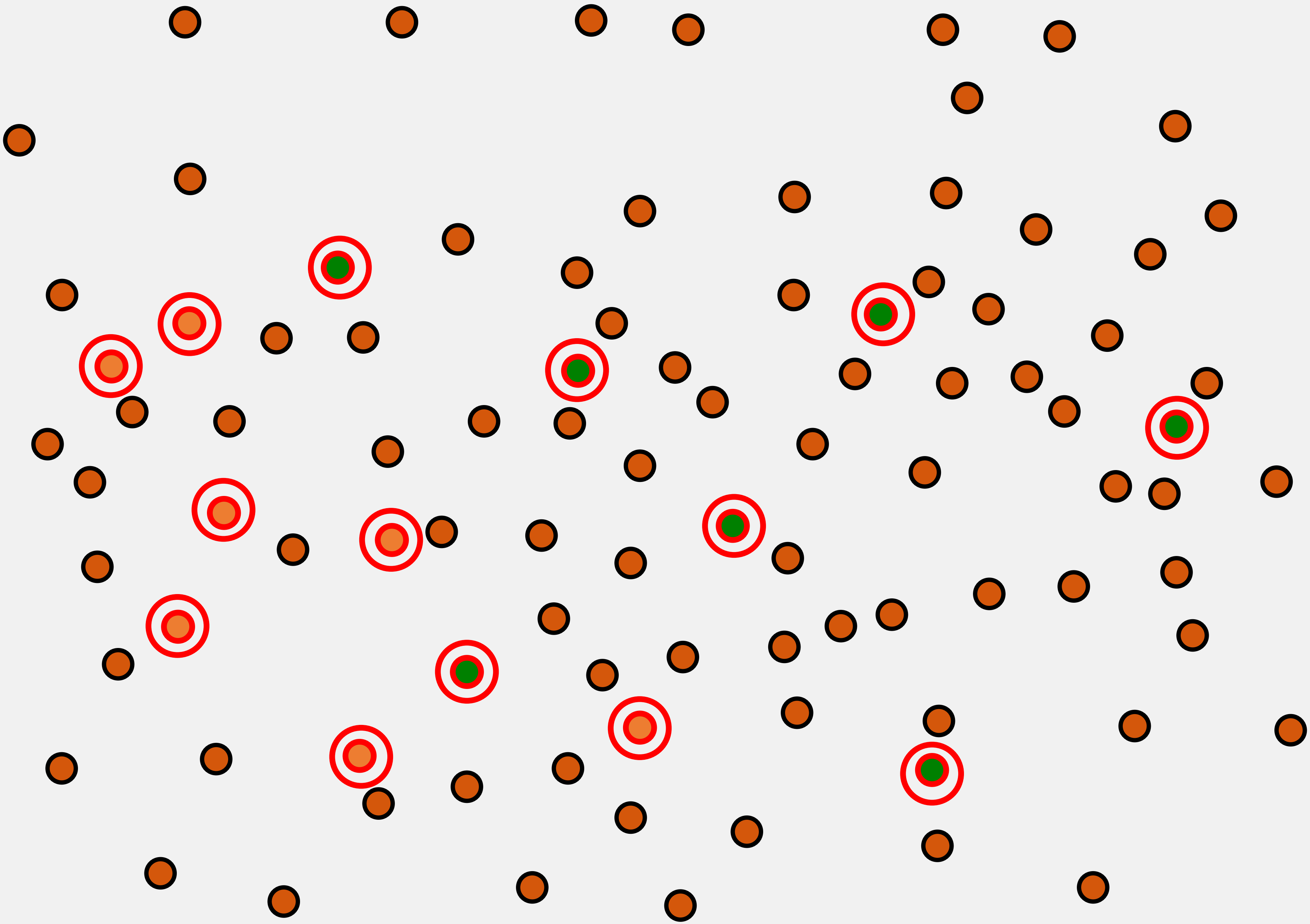
Our training data



The instance space

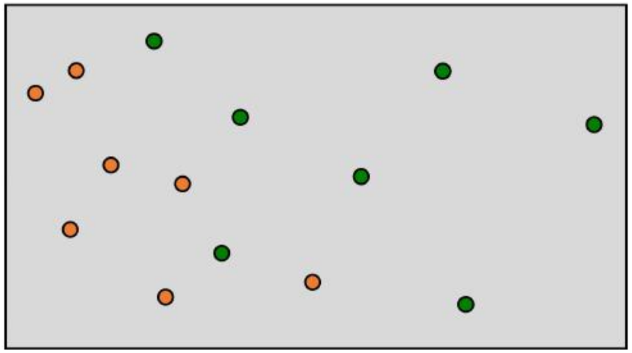




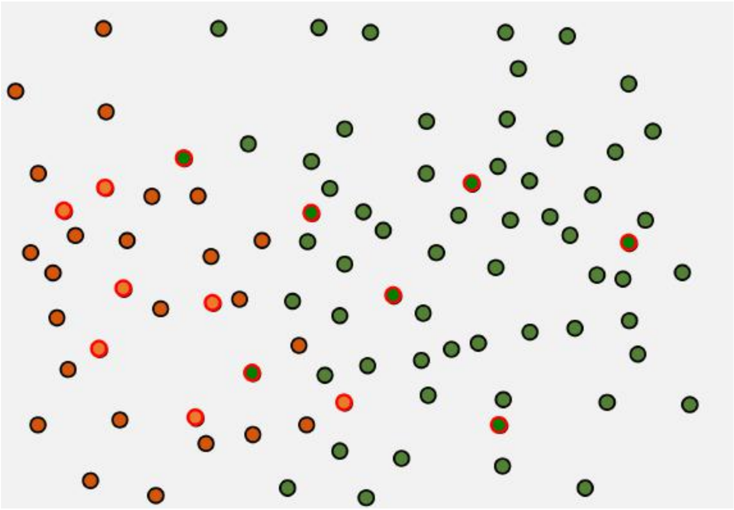


Which one is more likely?

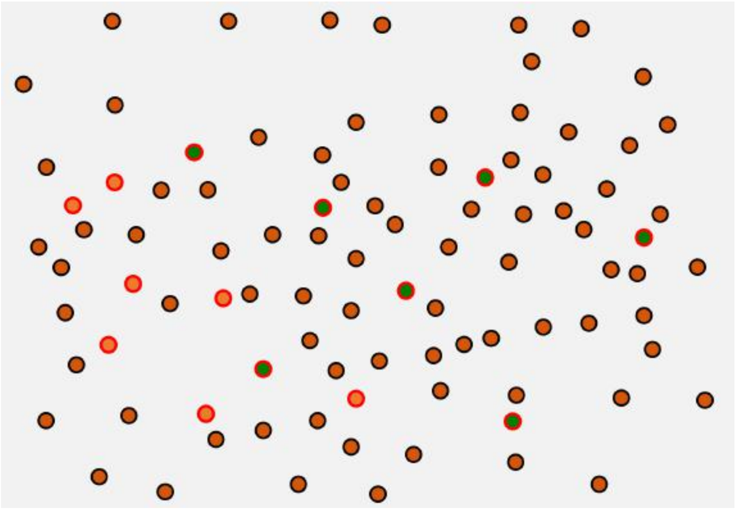
Training set



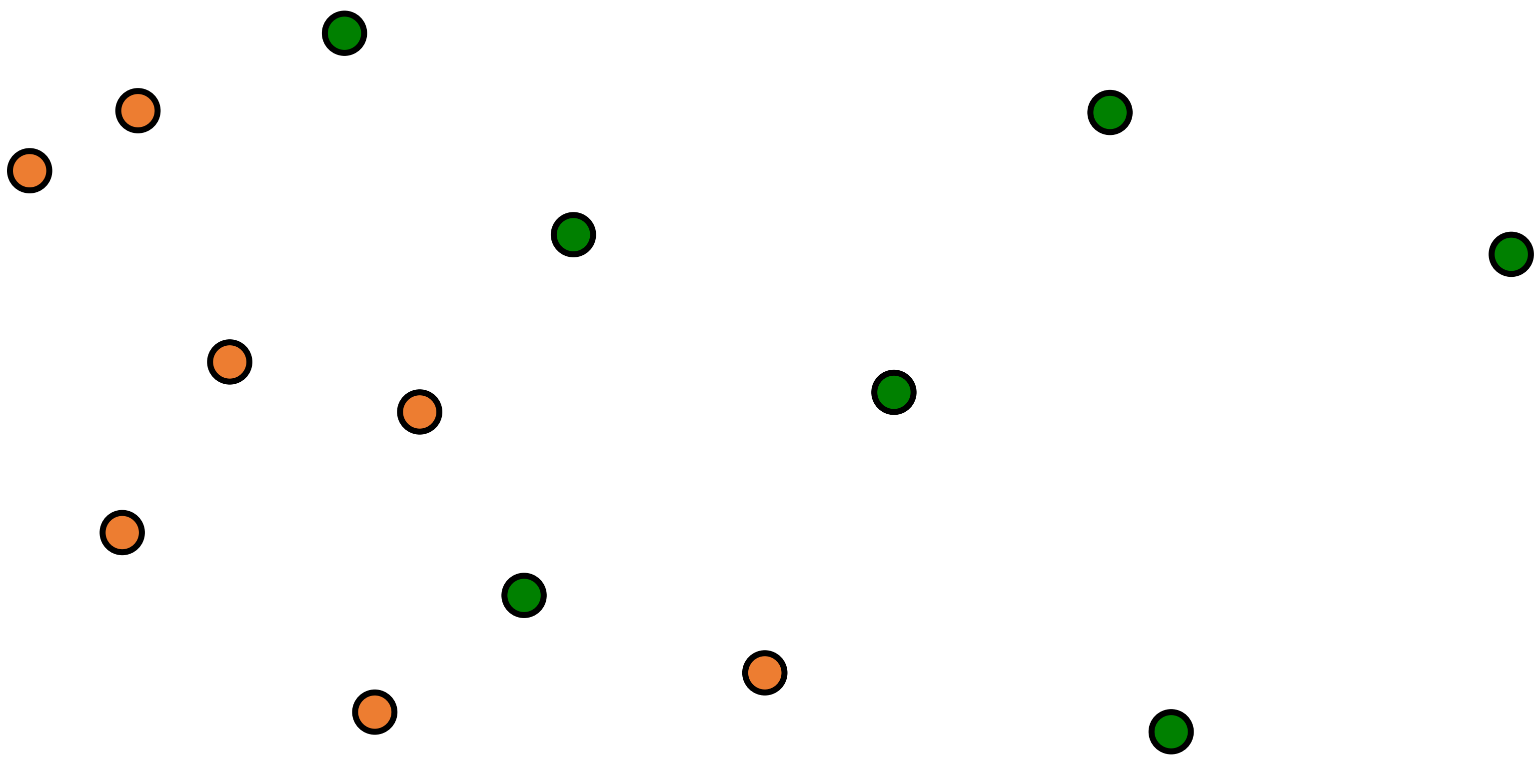
Data distribution A



Data distribution B

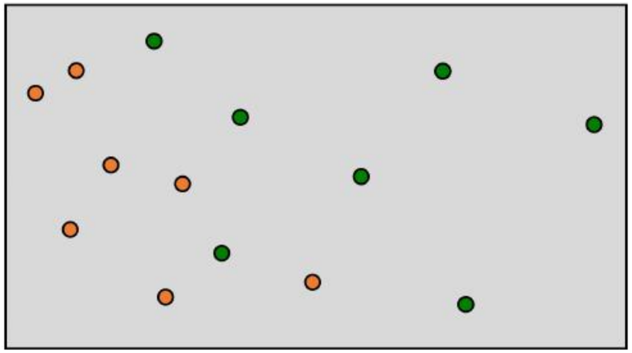


Our training data

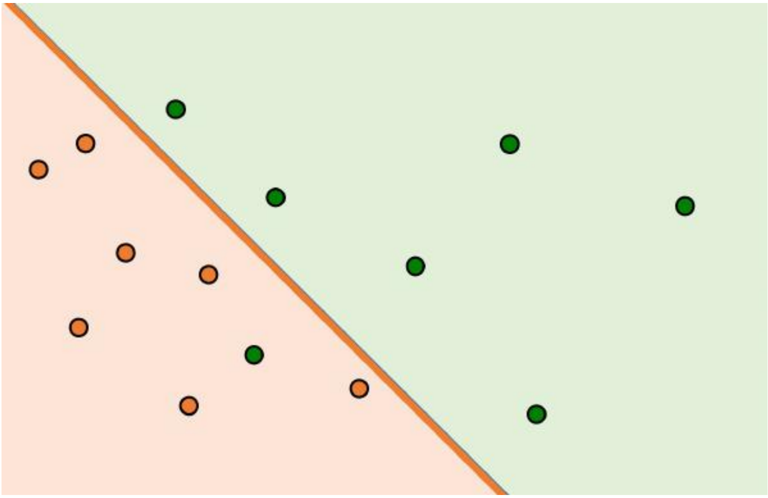


Which one is more likely to be a good hypothesis?

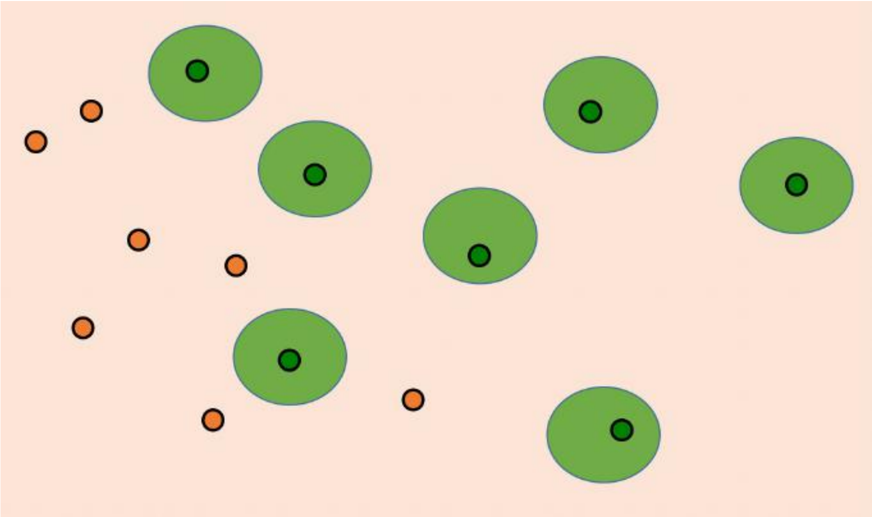
Training set



Hypothesis A

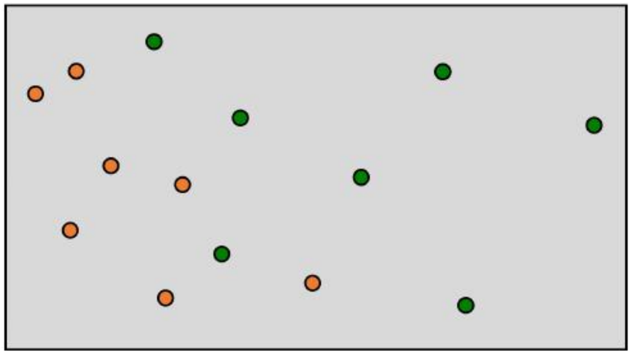


Hypothesis B

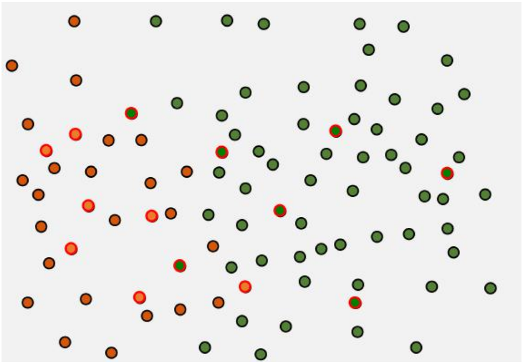


Which one is more likely to be a good hypothesis?

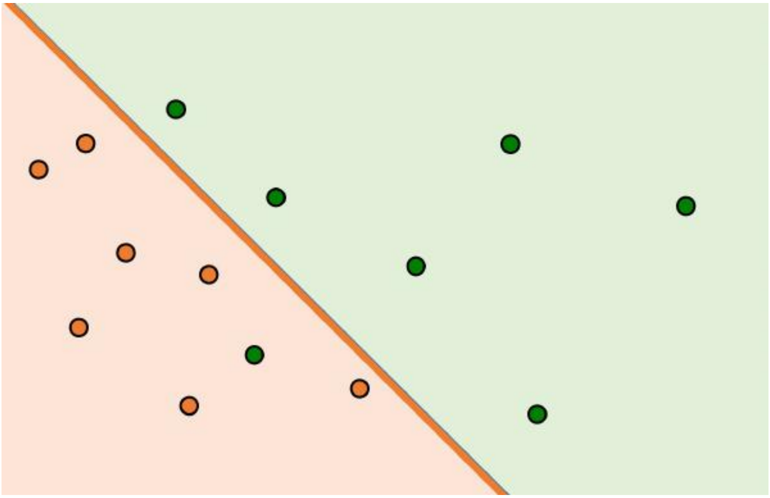
Training set



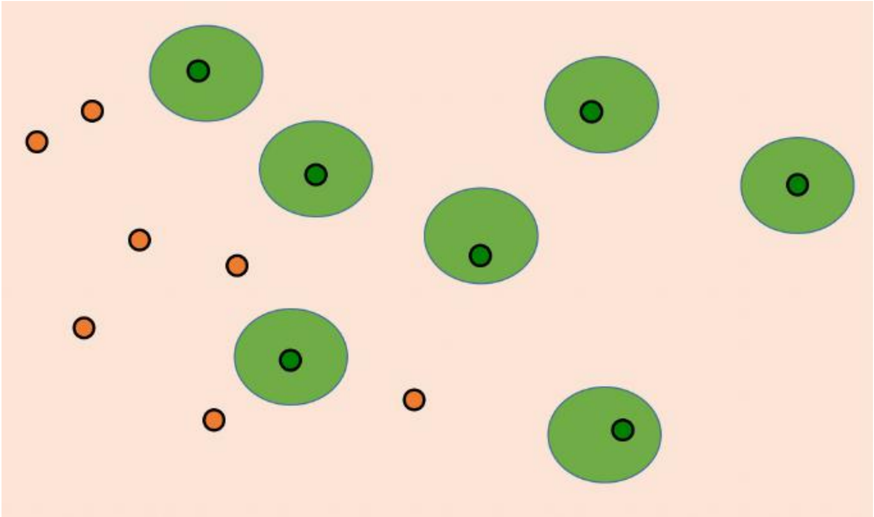
(Likely) Data Distribution



Hypothesis A

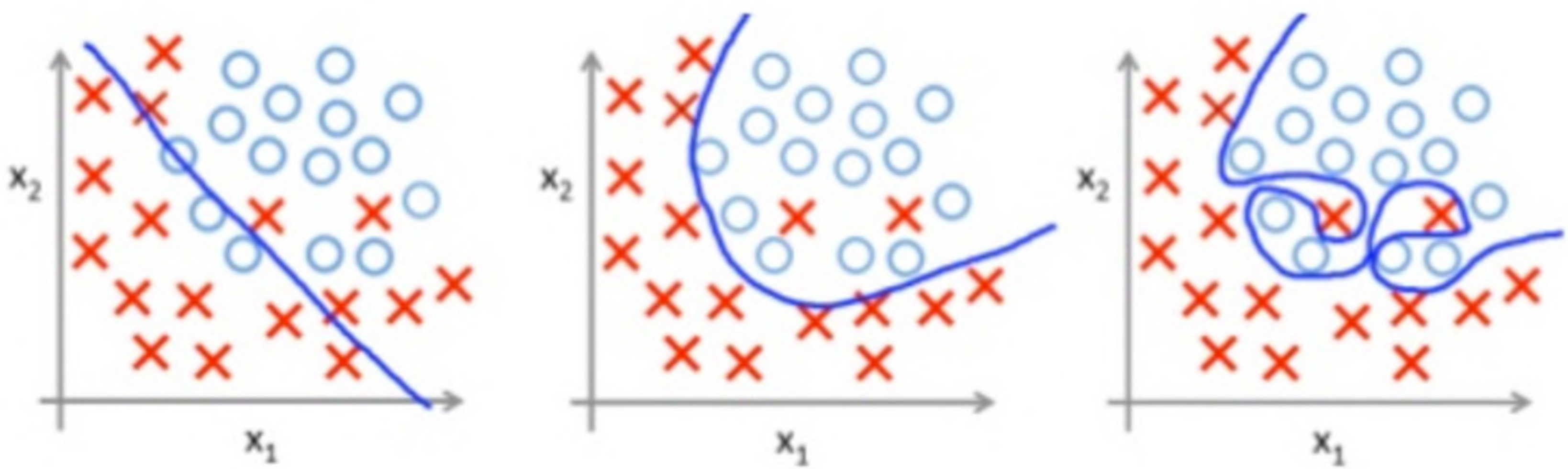


Hypothesis B



Under-fitting and over-fitting

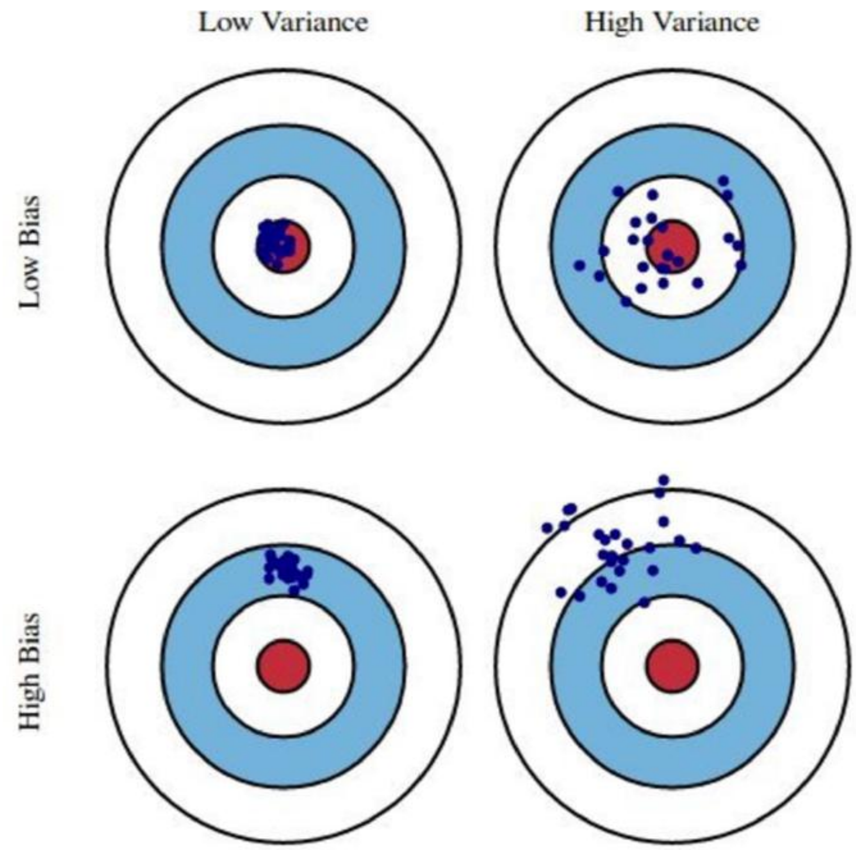
❖ Which classifier (blue line) is the best one?



(A) (B) (C)

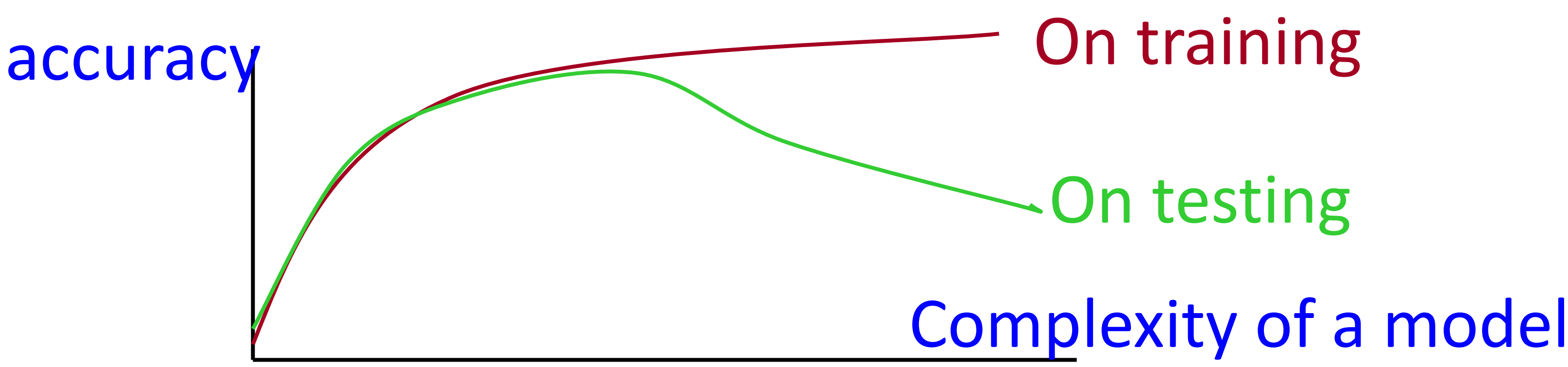
Bias V.S. Variance

- Remember, training data are subsamples drawn from the true distribution
- Exam strategy:
  - Study every chapter well
    - A+: Low var & bias
  - Study only a few chapters
    - A+? B? C? Low bias; High var
  - Study every chapter roughly
    - B+: Low var; high bias
  - Go to sleep
    - B ~D: High var, high bias



Overfitting the Data

- A classifier perform perfectly on the training data may not lead to the best generalization performance.
  - There may be noise in the training data
  - The algorithm might be making decisions based on very little data



Prevent overfitting

- Using a less-expressive model
  - E.g., linear model
- Adding regularization
  - Promote simper models
- Data perturbation (add noise in training)
  - Can be done algorithmically (e.g., dropout)
- Stop the optimization process earlier
  - Sounds bad in theory; but works in practice

More discussion in later lectures

K-Nearest Neighbor

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 many others who made their course material freely available online.

Motivation: spam mail

hi Spam x

marina <marina0279@static.vnpt.vn> Jan 16 (7 days ago)

to kw

This message has a from address in static.vnpt.vn but has failed static.vnpt.vn's required tests for authentication. Learn more

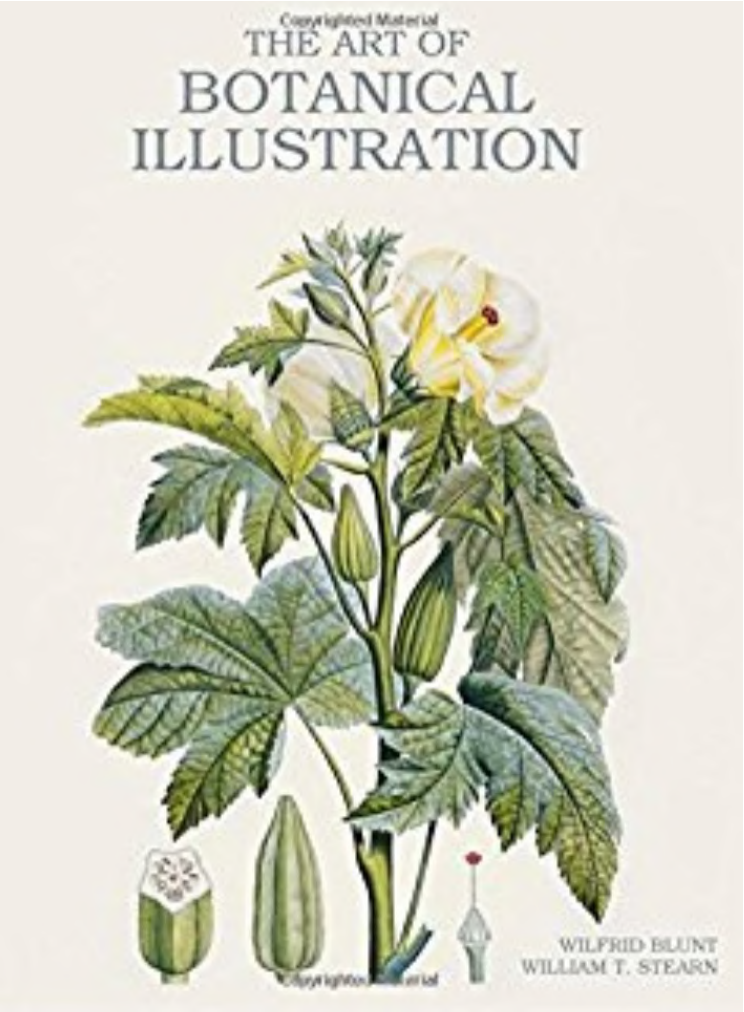
You seem like my type and I would like to know you more! Write me if you are interested, here is my email denisavafursula@rambler.ru and, if you want, I will send some of my photos.
Hugs, marina

Victoria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email vanesammrenate@	Jan 17
Oksana	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elenarzpilse@rambl	Jan 16
Oksana	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email venhelgarjxq@raml	**Jan 16**
Victoria	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elipetrayk@rambler	**Jan 16**
Natasha	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email lucietmsabine@ram	**Jan 16**
Daria	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email rokcmingrid@rambl	**Jan 16**
Katya	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email michaelavat62@rar	**Jan 16**
Yulia	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elisa4inge@ramble	**Jan 16**
Veronika	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email venlsajutta@ramble	**Jan 16**
Tanya	**hi** - You seem like my type and I would like to know you more! Write me if you are interested, here is my email ellairxt6t@rambler.r	**Jan 16**

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

(A)

(B)

(C)

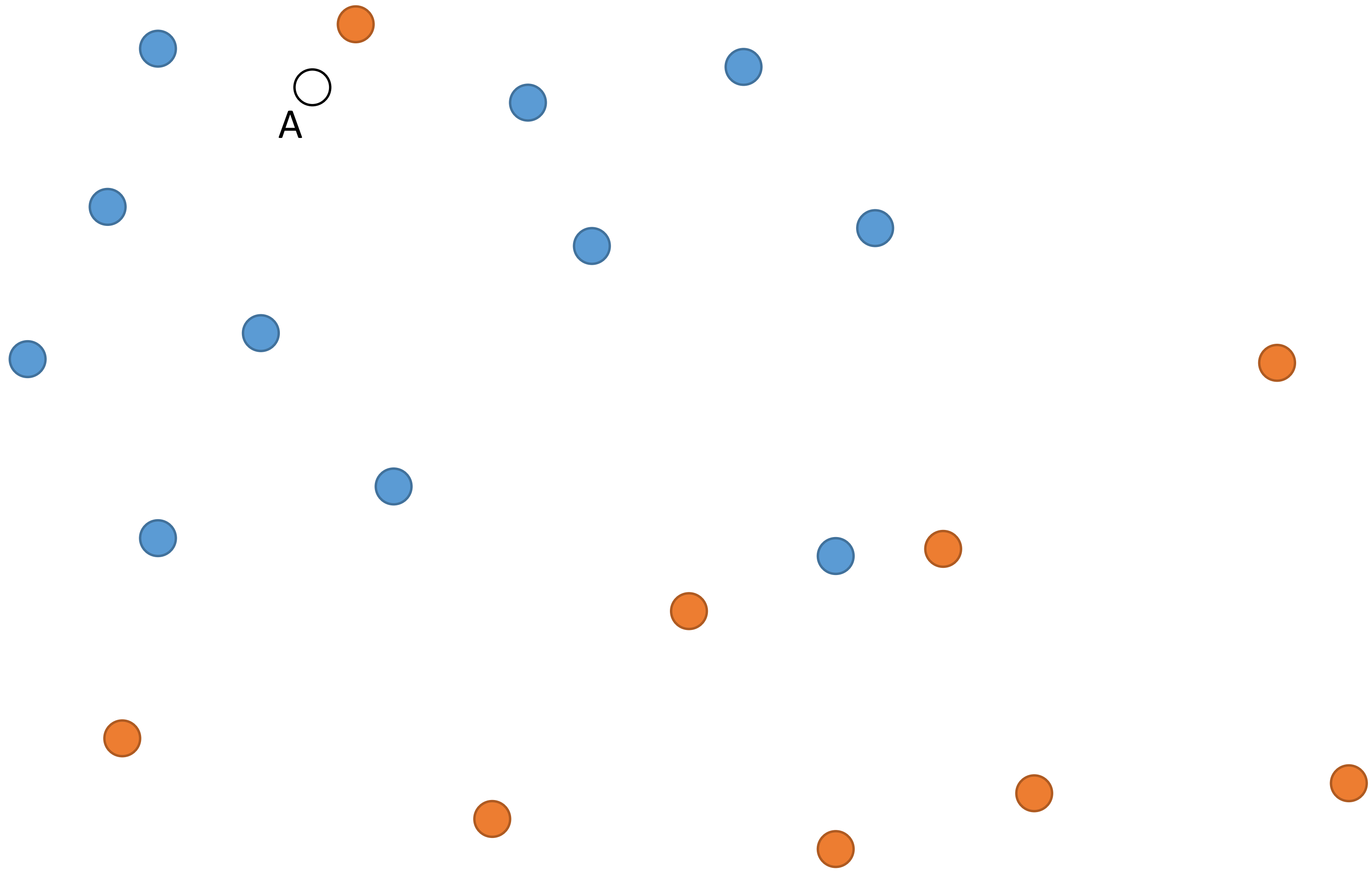
What you will learn in this lecture

- k-NN algorithm
- Distance between in the vector space
- Parameter tuning
- Decision boundary
- Curse of the Dimensionality (Theoretical Limitation)

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 y_i associated with $\mathbf{x}_i$

Example:
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

How to define distance?

- 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.

Distance between instances

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

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

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

Euclidean distance
Manhattan distance
Clinton
Hell's Kitchen
Korea Town
Murray Hill
Midtown East
Kips Bay
High Line Park
Fashion Institute of Technology
Central Park Zoo
Clearview Ziegfeld
21 Club
Intrepid Sea Air and Space Museum
The Joyce Theater
St. Vartan Park
NYU Medical Center
Terminal 5
W 59th St
W 42nd St
W 43rd St
W 34th St
E 57th St
E 56th St
5th Ave
Madison Ave
Park Ave
3rd Ave
9th Ave
10th Ave
11th Ave
12th Ave
Dyer Ave
A
B

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

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

P > 0

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

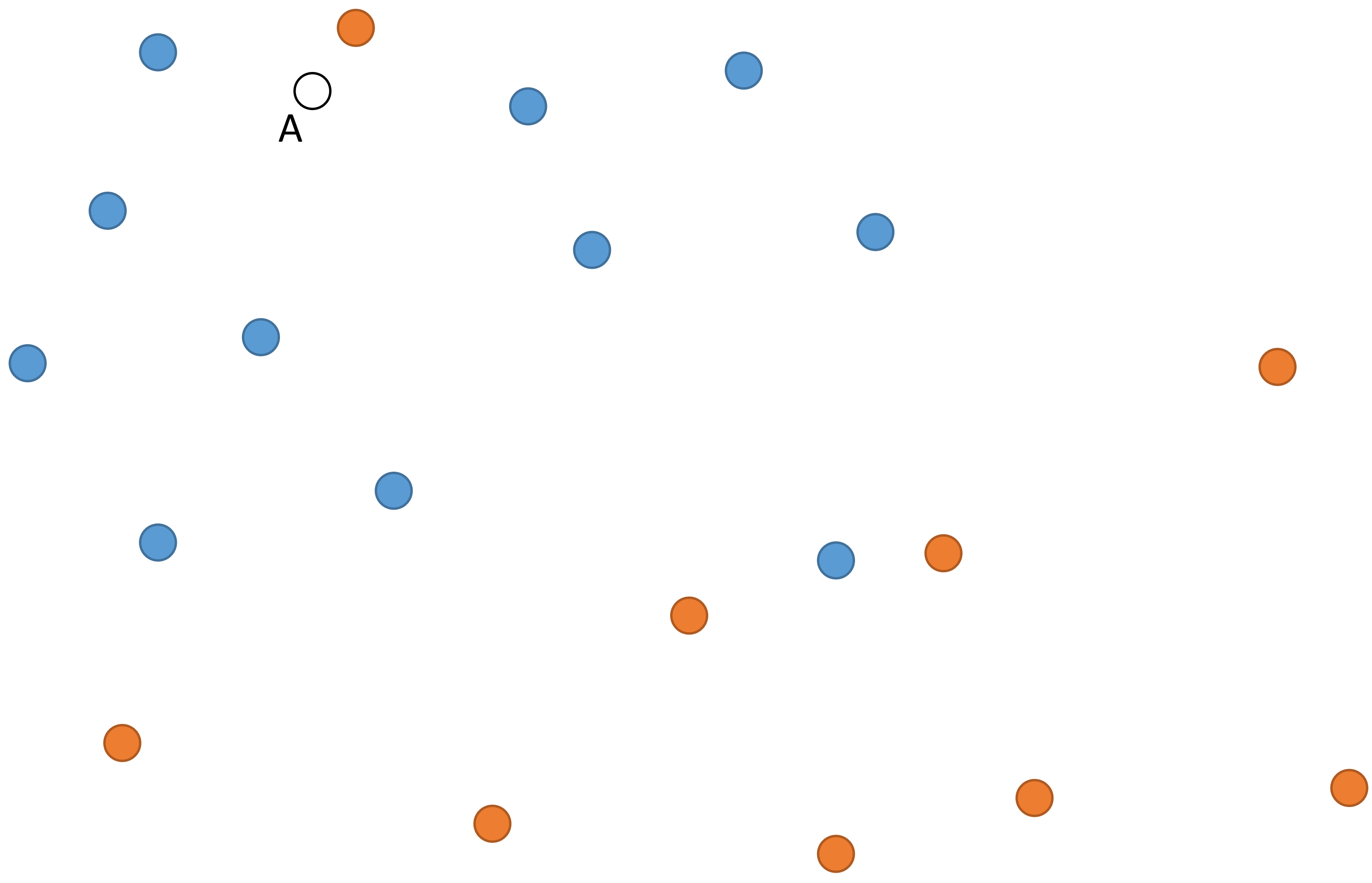
- Learning: Just store all the training examples

How to choose k and the distance measure?

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

Parameter K

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



Hyper-parameters in KNN

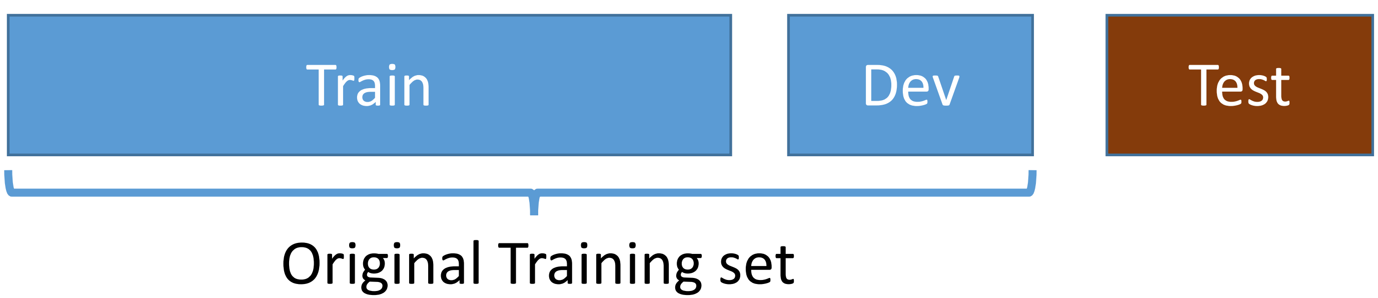
- (Hyper-)Parameters:
  - Choosing K (# nearest neighbors)
  - Distance measurement (e.g., p in the L_p-norm)

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

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

Train/Dev/Test splits

- You are not allowed to look at Text in parameter turning. Why?
- Split your training data into two sets:
  - Train: Training data (often 80-90%)
  - Dev: Development data (10-20%)
- Use Dev set (a.k.a. validation set) to find the best parameters



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 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}

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 represented

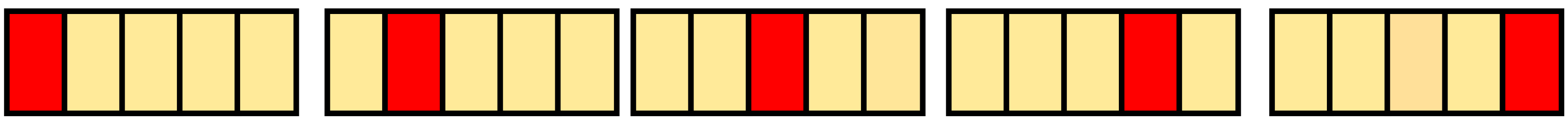
- Small Train, Large Dev
  (e.g., #train = 60, #dev = 40)

  No enough data to train a model

N-fold cross validation

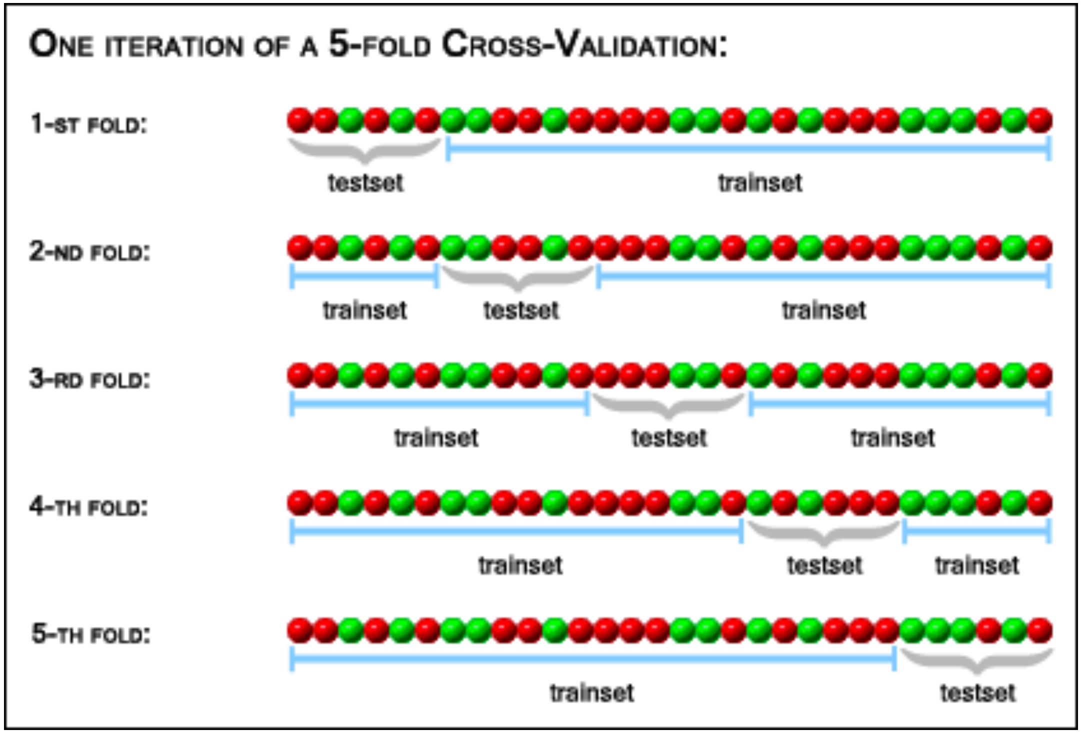
- Instead of a single training-dev split:

- 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?

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?

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

Neighbors' labels could be weighted by their distance

Related to Kernel method

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)