

Three Components of Machine Learning

Alex(ander) Jung

Assistant Professor for Machine Learning

Department of Computer Science

Aalto University

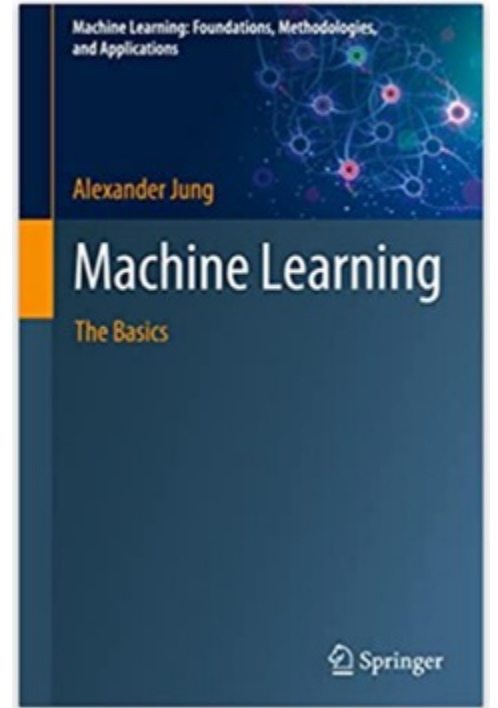
Learning Goals

- develop intuition for **how ML works**
- become familiar with concept of
 - **data** points (features, labels)
 - **model** (hypothesis space)
 - **loss function** (quality measure)

Reading.

- Chapter 1,2 of [MLBook]

AJ, “Machine Learning: The Basics”,
Springer, 2022.<https://mlbook.cs.aalto.fi>



NumPy

https://numpy.org/doc/stable/user/absolute_beginners.html

What is it all About ?


fit **model** to **data** to make **accurate**
predictions or forecasts !

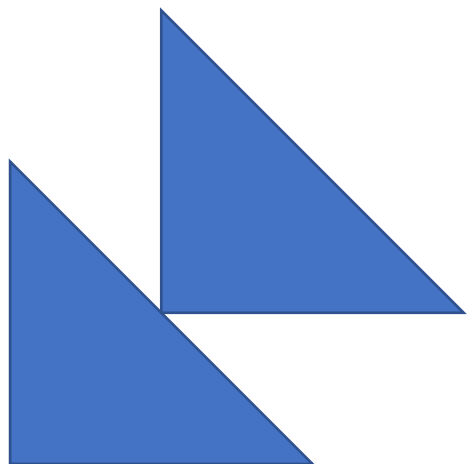
let's look at some learning
problems

1. element 2. 3. 4. 5. 6.

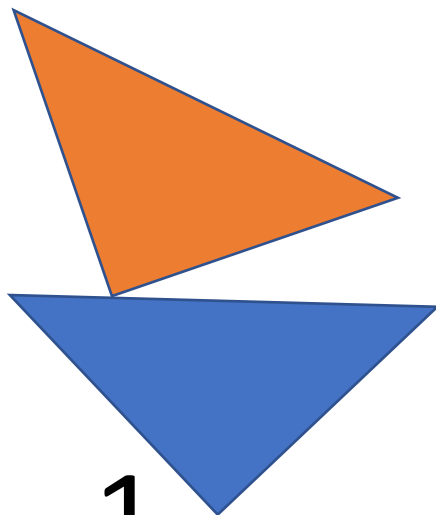
4, 5, 6, 7, 8, ?

“data point”

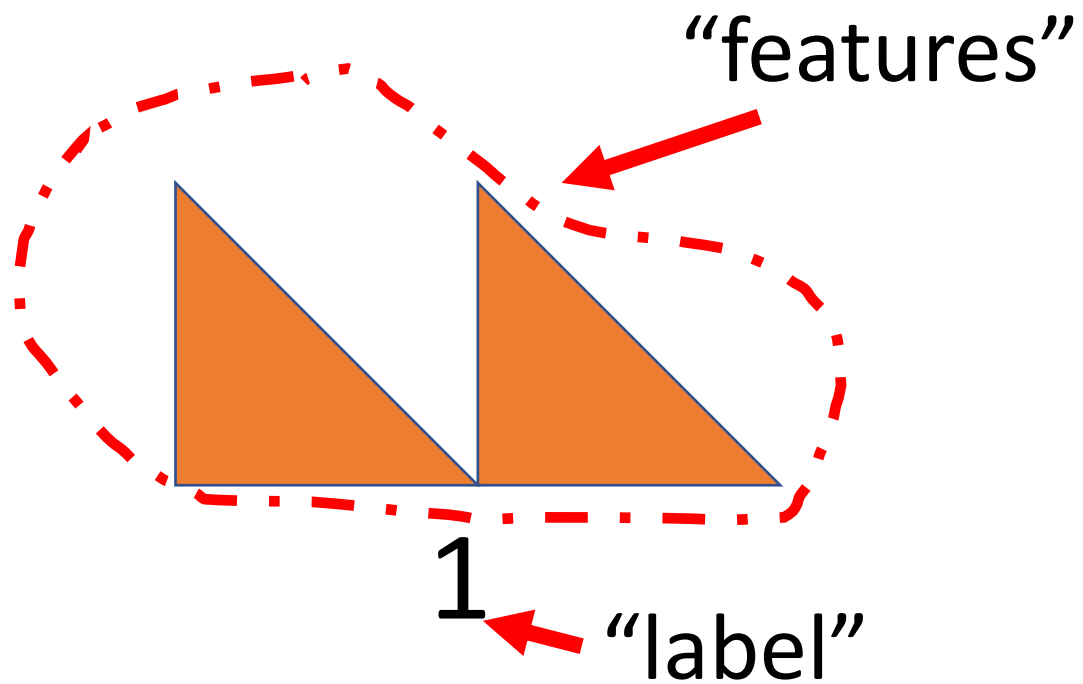




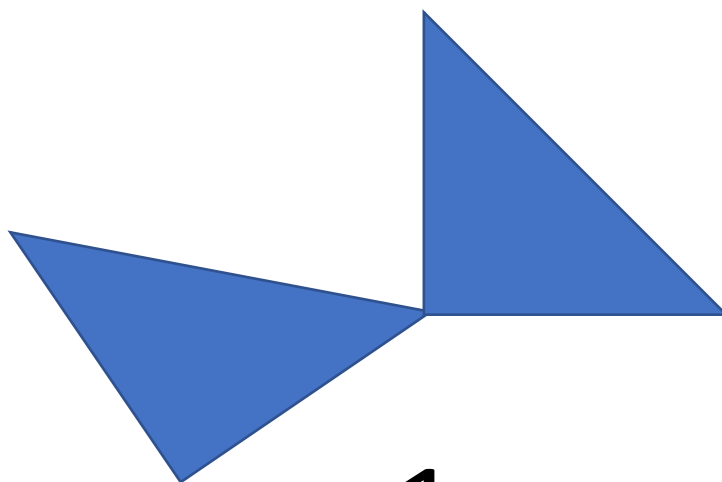
1



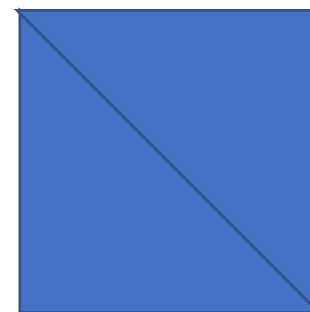
1



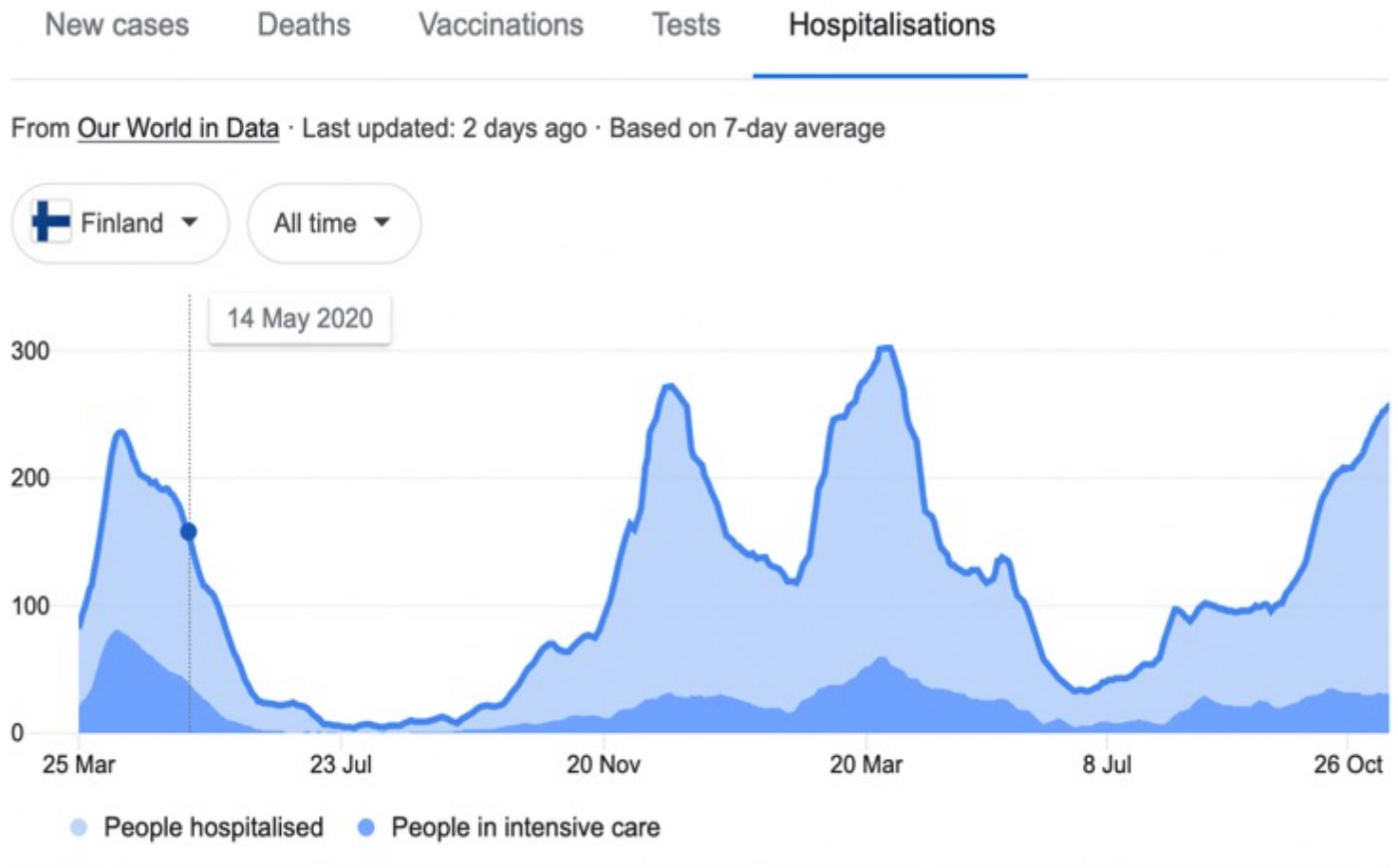
$1/2$



1



?



?





?

INNOVATION

How Artificial Intelligence Completed Beethoven's Unfinished Tenth Symphony

On October 9, the work will be performed in Bonn, Germany, and a recording will be released

Ahmed Elgammal, The Conversation

September 24, 2021

features (pixel RGB values)



“Cat”

“Dog”

“Cat”

?

← label →

<https://commons.wikimedia.org/>

“feature”



min tmp: -10
max tmp: -3



min tmp: -3
max tmp: 4



min tmp: 1
max tmp: 5

data point



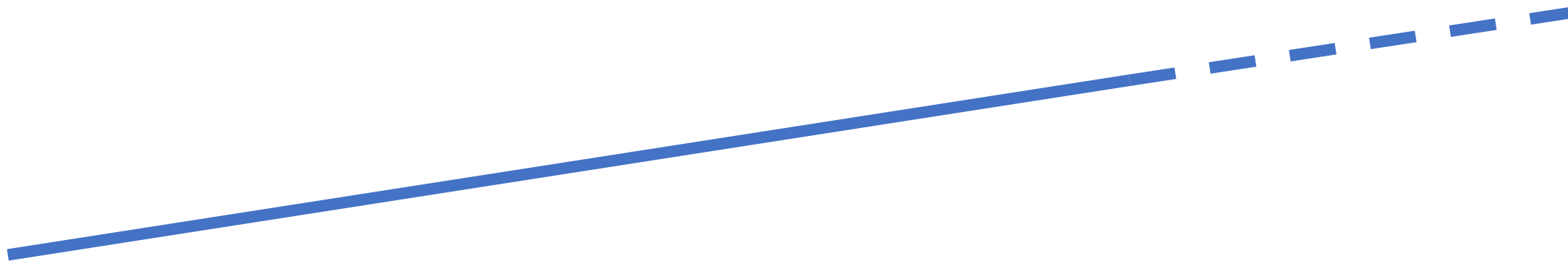
min tmp: -6
max tmp: ?

“label”

so, how does it work?

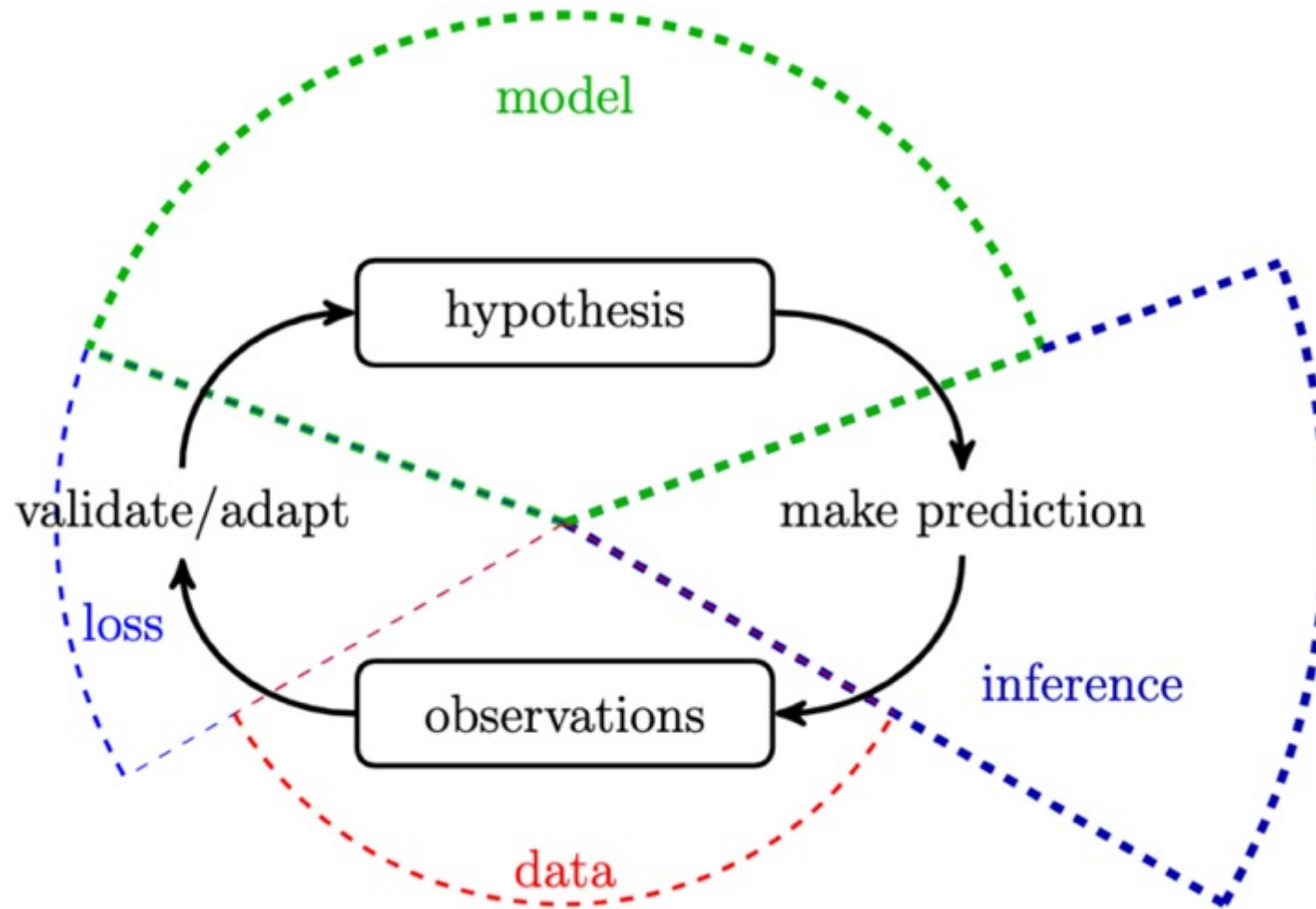
use hypothesis about data generation
to make predictions (forecasts)

4, 5, 6, 7, 8, ?

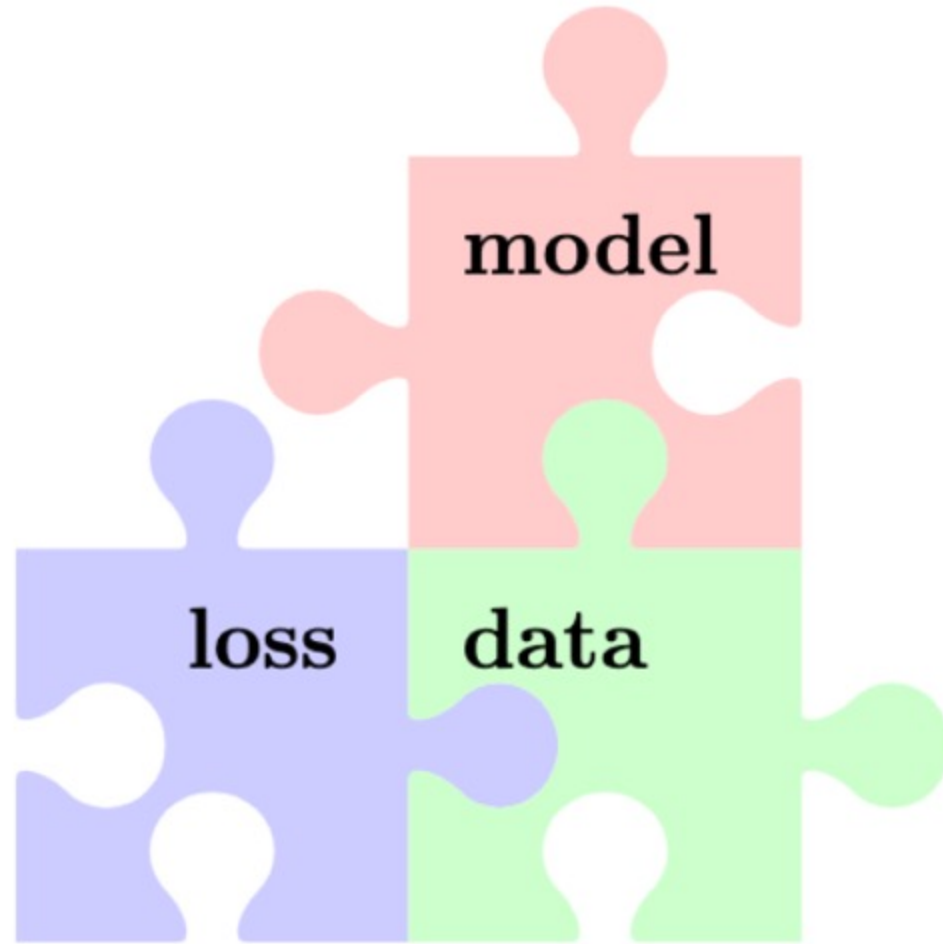


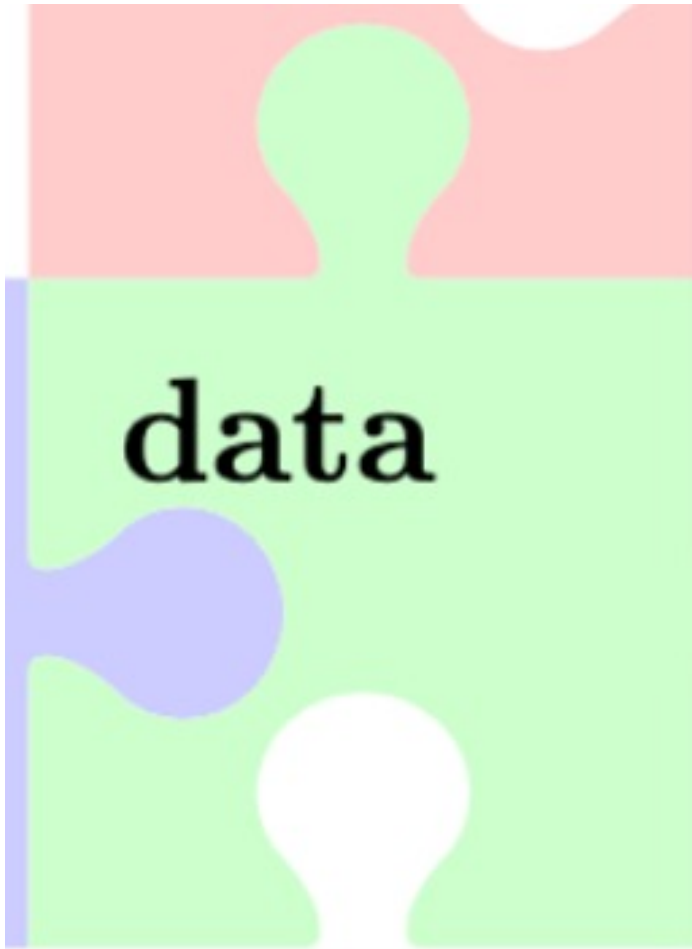
“hypothesis”

The Learning Cycle



Three Components of ML





“What I’m finding is that for a lot of problems, it’d be useful to shift our mindset toward **not just improving the code** but in a more systematic way of **improving the data**,” said Andrew Ng

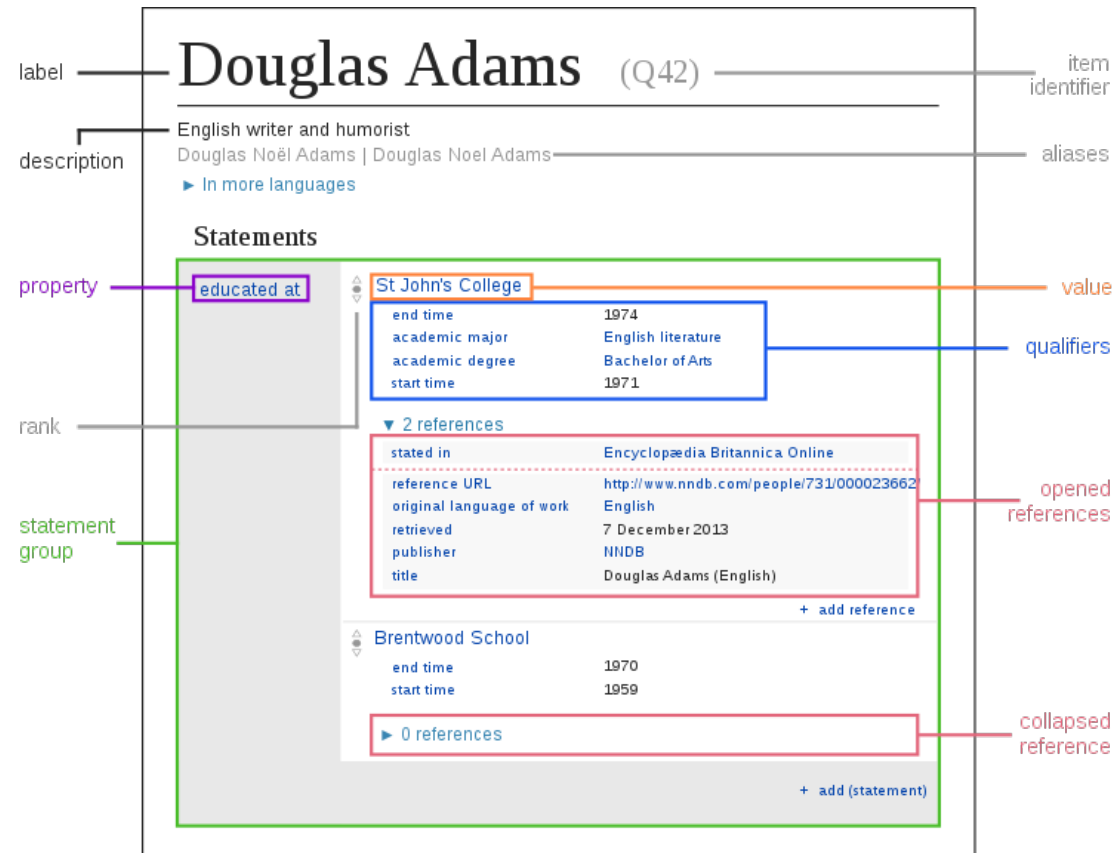
<https://read.deeplearning.ai/the-batch/issue-84/>

data
=
set of datapoints

What is a Datapoint?

some object that might carry relevant information

Datapoint = Some Item in Wikidata

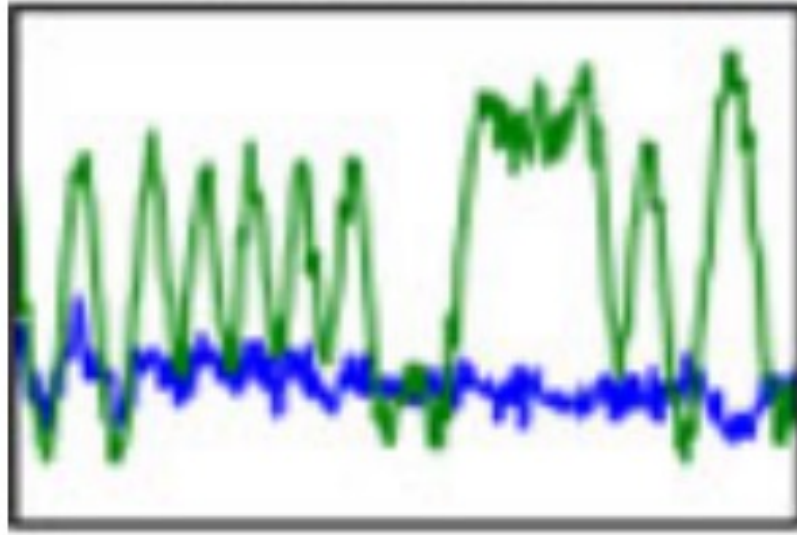


https://upload.wikimedia.org/wikipedia/commons/a/ae/Datamodel_in_Wikidata.svg

Datapoint = Some Period of Time

1.1.2020 01:00 - 2.1.2020 13:00

Datapoint = Some Wireless Signal

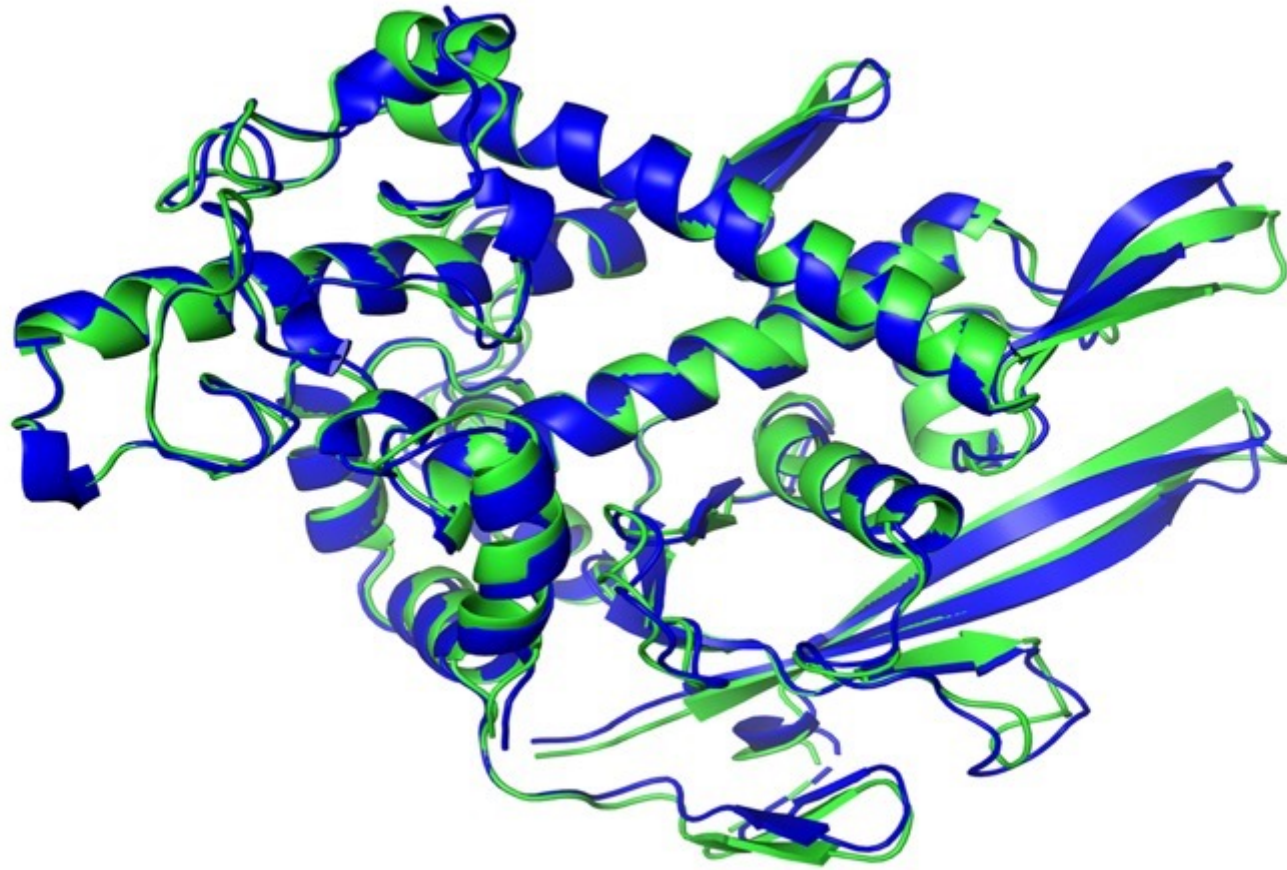


T. J. O'Shea, T. Roy and T. C. Clancy,
"Over-the-Air Deep Learning Based Radio Signal Classification,"
in *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168-179, Feb. 2018,
doi: 10.1109/JSTSP.2018.2797022.

Datapoint = Some Country



Datapoint = Some Protein



Datapoint = A Partial Differential Equation

$$\begin{aligned} \frac{\partial u}{\partial t}(t, x) + \frac{1}{2} \text{Tr} \left(\sigma \sigma^T(t, x) (\text{Hess}_x u)(t, x) \right) + \nabla u(t, x) \cdot \mu(t, x) \\ + f \left(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x) \right) = 0 \end{aligned} \quad [1]$$

RESEARCH ARTICLE



Solving high-dimensional partial differential equations using deep learning

 Jiequn Han, Arnulf Jentzen, and Weinan E

[+ See all authors and affiliations](#)

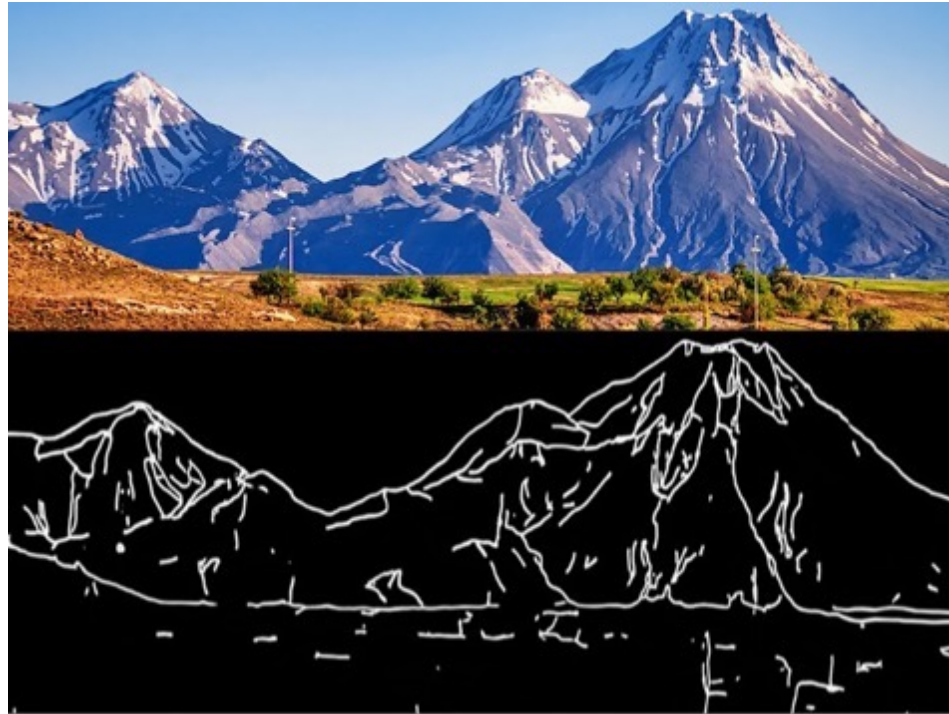
<https://www.pnas.org/content/115/34/8505/tab-article-info>

Datapoint = Some Bridge

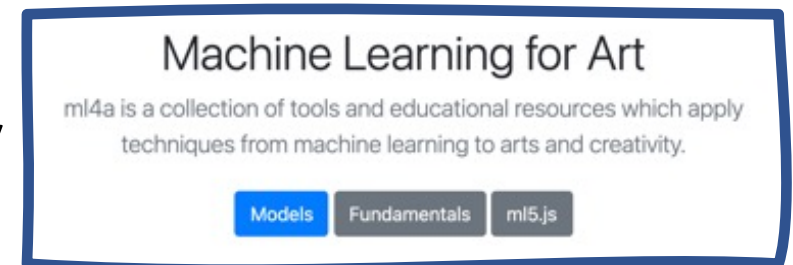


<https://commons.wikimedia.org/wiki/Category:Bridges>

Datapoint = Image Sketch



<https://ml4a.net/>



Features and Labels.

datapoint characterized by

- features: low-level properties; easy to measure/compute
- labels: high-level quantity of interest; difficult to measure/determine

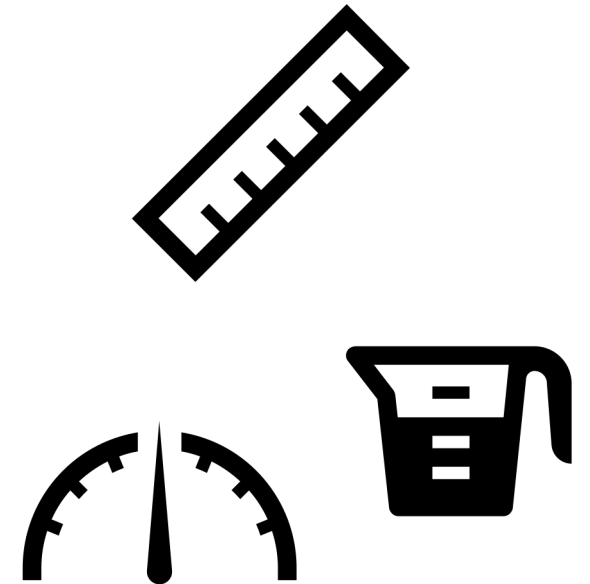
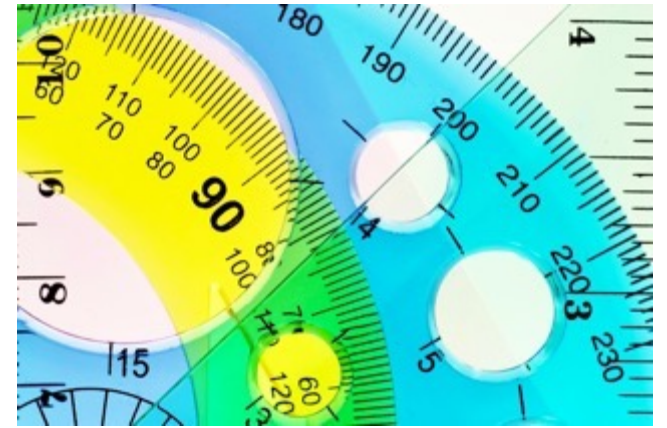
Numeric Features

we mainly use numeric features x_1, \dots, x_n to characterize a datapoint

stack features into **feature vector**

Python: use **numpy array** to store features

discuss feature learning methods later



Features of an Image.

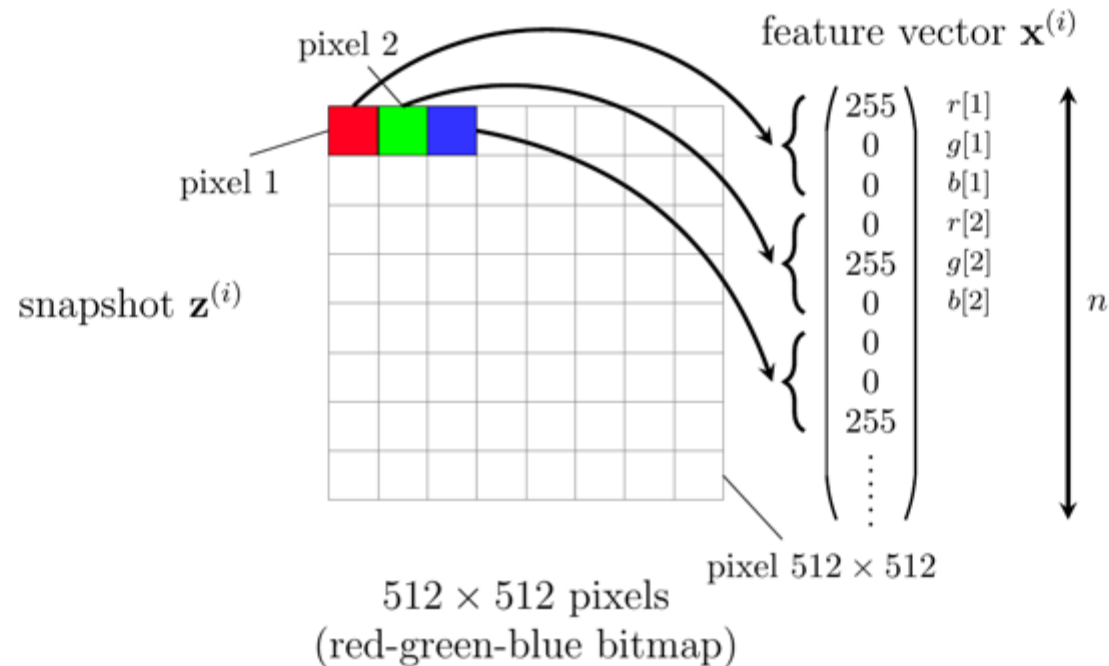


Figure 2.5: If the snapshot $\mathbf{z}^{(i)}$ is stored as a 512×512 RGB bitmap, we could use as features $\mathbf{x}^{(i)} \in \mathbb{R}^n$ the red-, green- and blue component of each pixel in the snapshot. The length of the feature vector would then be $n = 3 \times 512 \times 512 \approx 786000$.

Features of an Audio Recording.

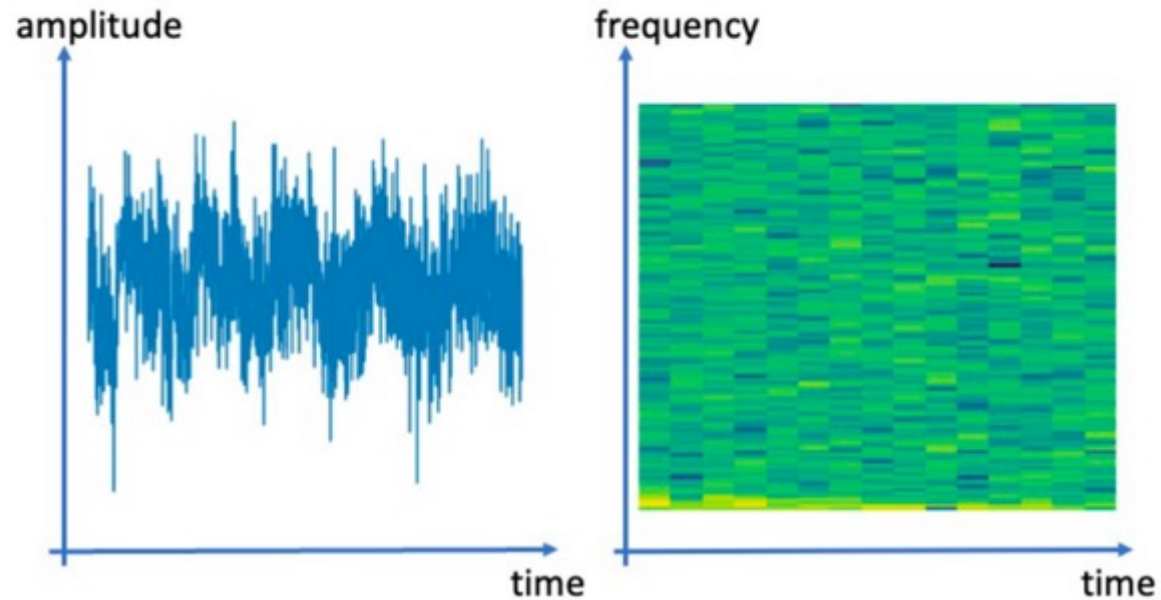


Figure 2.4: Two visualizations of a data point that represents an audio recording. The left figure shows a line plot of the audio signal amplitudes. The right figure shows a spectrogram of the audio recording.

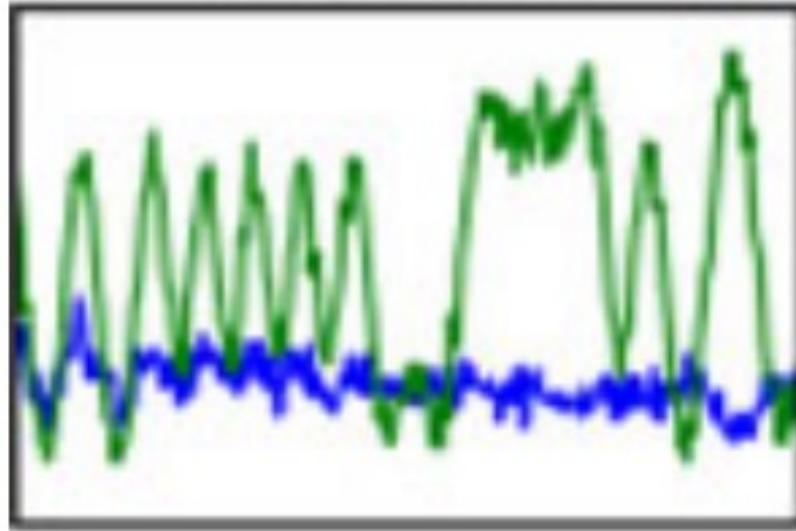
Datapoint = Period of Time

1.1.2020 00:00 - 1.1.2020 23:55

features: temperature recordings @ 01:00,
03:00, 05:00

label: temperature recording @ 23:00

Datapoint = Wireless Signal

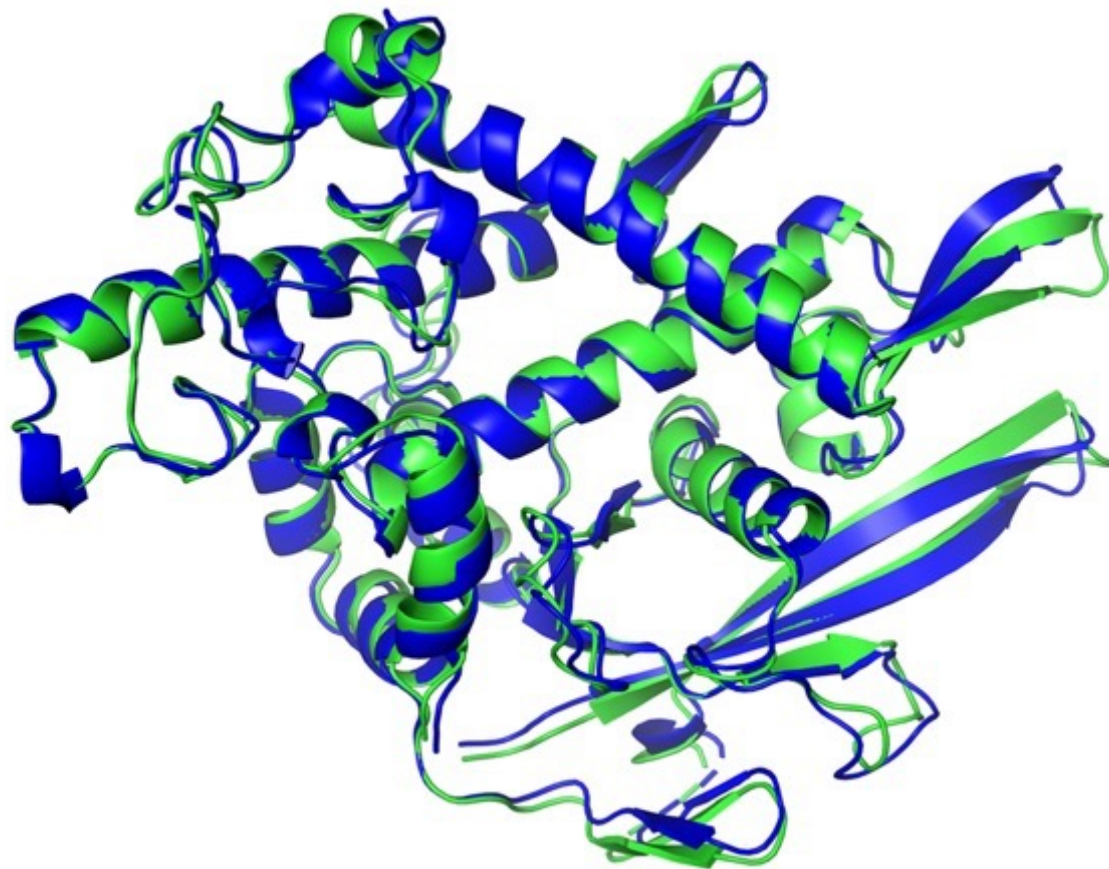


features:

label:

T. J. O'Shea, T. Roy and T. C. Clancy,
"Over-the-Air Deep Learning Based Radio Signal Classification,"
in *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168-179, Feb. 2018,
doi: 10.1109/JSTSP.2018.2797022.

Datapoint = A Protein



features:

label:

Datapoint = A Partial Differential Equation

$$\begin{aligned} \frac{\partial u}{\partial t}(t, x) + \frac{1}{2} \text{Tr} \left(\sigma \sigma^T(t, x) (\text{Hess}_x u)(t, x) \right) + \nabla u(t, x) \cdot \mu(t, x) \\ + f \left(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x) \right) = 0 \end{aligned} \quad [1]$$

features:

label:

Datapoint = A Bridge

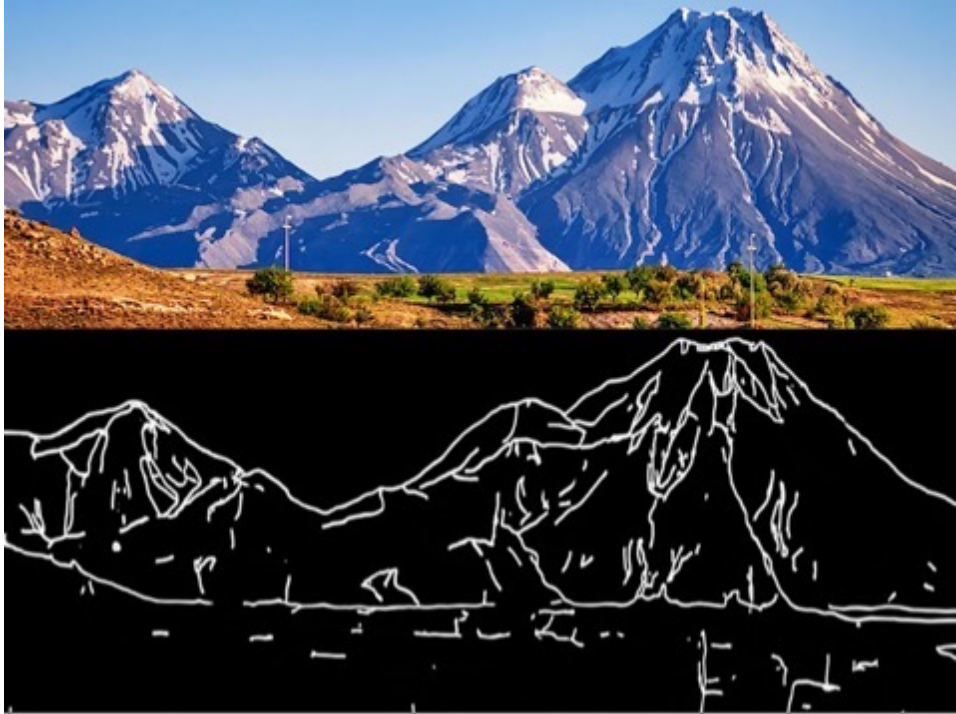


features:

label:

<https://commons.wikimedia.org/wiki/Category:Bridges>

Datapoint = Image Sketch



features:

label:

<https://ml4a.net/>

Datapoints, their Features and Labels are Design Choices!

raw data from FMI

<https://en.ilmatieteenlaitos.fi/download-observations>

	A	B	C	D	E	F	G	H	I
	Year	m	d	Time	precip	snow	airtmp	mintmp	maxtmp
1	2020	1	2	00:00	0,4	55	2,5	-2	4,5
2	2020	1	3	00:00	1,6	53	0,8	-0,8	4,6
3	2020	1	4	00:00	0,1	51	-5,8	-11,1	-0,7
4	2020	1	5	00:00	1,9	52	-13,5	-19,1	-4,6
5	2020	1	6	00:00	0,6	52	-2,4	-11,4	-1
6	2020	1	7	00:00	4,1	52	0,4	-2	1,3
7	2020	1	8	00:00	4,3	51	0,8	0,1	1,8
8	2020	1	9	00:00	-1	51	-0,6	-1,9	1,6
9	2020	1	10	00:00	-1	51	-6,2	-11	-1,4
10	2020	1	11	00:00	2,8	50	-4,8	-10,7	-2,1
11	2020	1	12	00:00	-1	53	-1,3	-3,5	0,9
12	2020	1	13	00:00	-1	53	-6,4	-12,9	-3,1
13	2020	1	14	00:00	9,7	52	-2,8	-9	-0,7
14	2020	1	15	00:00	-1	63	0,2	-0,7	0,6
15	2020	1	16	00:00	0,4	62	-3,9	-5,2	0,1
16	2020	1	17	00:00	2	62	-5,2	-8,4	-0,7

features

	A	B	C	D	E	F	G	H	I
1	Year	m	d	Time	precip	snow	airtmp	mintmp	maxtmp
2	2020	1	2	00:00	0,4	55	2,5	-2	4,5
3	2020	1	3	00:00	1,6	53	0,8	-0,8	4,6
4	2020	1	4	00:00	0,1	51	-5,8	-11,1	-0,7
5	2020	1	5	00:00	1,9	52	-13,5	-19,1	-4,6
6	2020	1	6	00:00	0,6	52	-2,4	-11,4	-1
7	2020	1	7	00:00	4,1	52	0,4	-2	1,3
8	2020	1	8	00:00	4,3	51	0,8	0,1	1,8
9	2020	1	9	00:00	-1	51	-0,6	-1,1	1,6
10	2020	1	10	00:00	1	51	-6,2	-11	-1,4
11	2020	1	11	00:00	2,8	50	-4,8	-10,7	-2,1
12	2020	1	12	00:00	-1	53	-1,3	-3,5	0,9
13	2020	1	13	00:00	-1	53	-6,4	-12,9	-3,1
14	2020	1	14	00:00	9,7	52	-2,8	-9	-0,7
15	2020	1	15	00:00	-1	63	0,2	-0,7	0,6
16	2020	1	16	00:00	0,4	62	-3,9	-5,2	0,1
17	2020	1	17	00:00	2	62	-5,2	-8,4	-0,7

label

data point

data point, features and label
are design choices!

```
newdataset= somedata[somedata['date'] == '2021-06-01'] ;  
print(newdataset)
```

	date	time	temperature
0	2021-06-01	00:00	6.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

Key Parameters of a Data Set

number n of features



	A	B	C	D	E	F	G	H	I
1	Year	m	d	Time	precip	snow	airtmp	mintmp	maxtmp
2	2020	1	2	00:00	0,4	55	2,5	-2	4,5
3	2020	1	3	00:00	1,6	53	0,8	-0,8	4,6
4	2020	1	4	00:00	0,1	51	-5,8	-11,1	-0,7
5	2020	1	5	00:00	1,9	52	-13,5	-19,1	-4,6
6	2020	1	6	00:00	0,6	52	-2,4	-11,4	-1
7	2020	1	7	00:00	4,1	52	0,4	-2	1,3
8	2020	1	8	00:00	4,3	51	0,8	0,1	1,8
9	2020	1	9	00:00	-1	51	-0,6	-1,9	1,6
10	2020	1	10	00:00	-1	51	-6,2	-11	-1,4
11	2020	1	11	00:00	2,8	50	-4,8	-10,7	-2,1
12	2020	1	12	00:00	-1	53	-1,3	-3,5	0,9
13	2020	1	13	00:00	-1	53	-6,4	-12,9	-3,1
14	2020	1	14	00:00	9,7	52	-2,8	-9	-0,7
15	2020	1	15	00:00	-1	63	0,2	-0,7	0,6
16	2020	1	16	00:00	0,4	62	-3,9	-5,2	0,1
17	2020	1	17	00:00	2	62	-5,2	-8,4	-0,7
18	2020	1	18	00:00	19,6	65	-4,6	-7,3	-4,2
19	2020	1	19	00:00	0,7	81	-4,4	-8,8	-2,7
20	2020	1	20	00:00	2,8	79	-1,8	-10,5	1,2

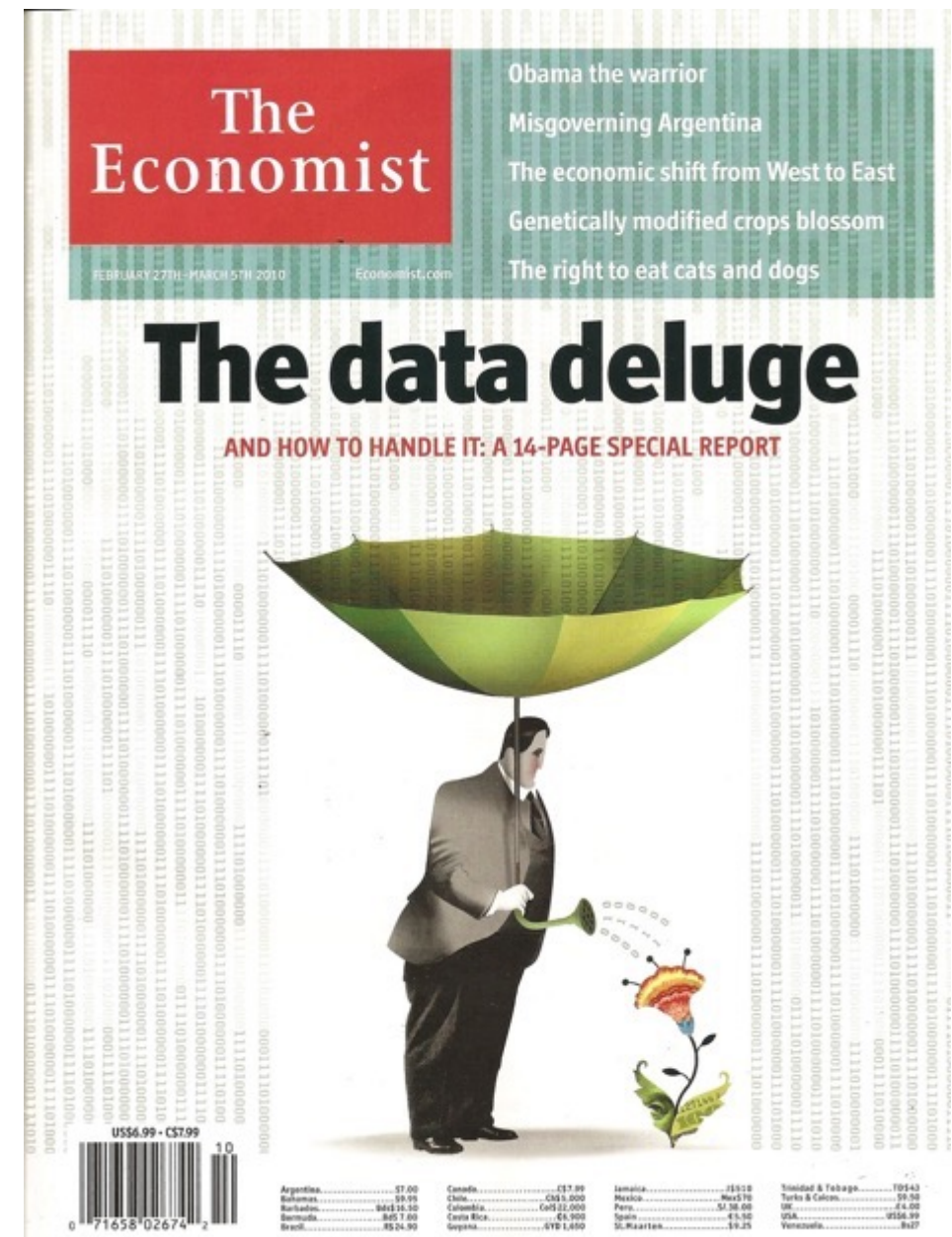
number m of
data points
“sample size”



Feature Deluge.

modern information
technology provides huge
number of raw features

- smartphones
- webcams
- social networks
- smart watch
-



use only most relevant features but not fewer.

missing relevant features bad for accuracy

using many irrelevant features wastes
computation and might result in overfitting


```
newdataset= somedata[somedata['date'] == '2021-06-01'] ;  
print(newdataset)
```

	date	time	temperature
0	2021-06-01	00:00	5.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

data point = some day at
FMI station

feature = nr of hourly observations

want to predict maximum daytime
temperature

missing relevant features bad for accuracy

```
newdataset= somedata[somedata['date'] == '2021-06-01'] ;  
print(newdataset)
```

	date	time	temperature
0	2021-06-01	00:00	6.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

data point = some day at
FMI station

feature = hourly temp. 00:00 –
15:00

want to predict temp at 16:00

using irrelevant features wastes comp. resources

Label is Design Choice!

YOU choose the label of a data point

by choosing/defining label you define
the ML problem or learning task !

Regression. Numeric Labels.

	date	time	temperature
0	2021-06-01	00:00	6.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

datapoint

“2021-06-01 at some FMI station”

label = tmp at 15:00

Binary Classification.

	date	time	temperature
0	2021-06-01	00:00	6.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

datapoint

“2021-06-01 at some FMI station”

label =

- “hot” if tmp at 15:00 > 10
- “cold” if ... ≤ 10

Multi-Class Classification

	date	time	temperature
0	2021-06-01	00:00	6.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

datapoint

“2021-06-01 at some FMI station”

label =

- “nice morning” if tmp at 15:00 < 10 and tmp at 10:00 > 10
- “nice noon” if tmp at 15:00 > 10 and tmp at 10:00 < 10
- “nice day” if tmp at 15:00 > 10 and tmp at 10:00 > 10

Multi-Label Problems – Multitask Learning

by choosing/defining label you define the ML task !

for same data, use different labels → multiple learning tasks

multi-label class. (special case of multi-task learning)

Multi-Label Regression.

	date	time	temperature
0	2021-06-01	00:00	6.2
1	2021-06-01	01:00	6.4
2	2021-06-01	02:00	6.4
3	2021-06-01	03:00	6.8
4	2021-06-01	04:00	7.1
5	2021-06-01	05:00	7.6
6	2021-06-01	06:00	7.5
7	2021-06-01	07:00	8.1
8	2021-06-01	08:00	10.3
9	2021-06-01	09:00	12.8
10	2021-06-01	10:00	15.0
11	2021-06-01	11:00	14.1
12	2021-06-01	12:00	16.5
13	2021-06-01	13:00	13.6
14	2021-06-01	14:00	14.2
15	2021-06-01	15:00	13.3
16	2021-06-01	16:00	14.5
17	2021-06-01	17:00	13.8

datapoint

“2021-06-01 at some FMI station”

label1 = tmp at 10:00

label2= tmp at 15:00

Multi-Label Classification.



$y_1 = 1$ or 0 if car present or not

$y_2 = 1$ or 0 if person present or not

$y_3 = 1$ or 0 if tree present or not

Features and Labels - Notation.

- consider m datapoints, indexed by $i = 1, \dots, m$
- i -th datapoint with features $x_1^{(i)}, \dots, x_n^{(i)}$ and label $y^{(i)}$
- stack features of i -th data point into feature vector

$$\mathbf{x}^{(i)} = \left(x_1^{(i)}, \dots, x_n^{(i)} \right)^T$$

- represent data by feature matrix and label vector !

Feature Matrix.

$$\mathbf{X} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)})^T = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Label Vector.

$$\mathbf{y} = (y_1, y_2, \dots, y_m)^T \in \mathbb{R}^m$$

NumPy Arrays

- feature matrix and label vector are numeric arrays
- Python library NumPy provides methods for num.arr.

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> #  $y = 1 * x_0 + 2 * x_1 + 3$ 
>>> y = np.dot(X, np.array([1, 2])) + 3
```

“Data = X and y ”

- represent data by feature mtx X and label vec. y

```
In [5]: from sklearn.datasets import load_iris  
dataset = load_iris()  
X = dataset.data  
y = dataset.target
```



model



Statisticians, like artists, have the
bad habit of falling in love with their
models.

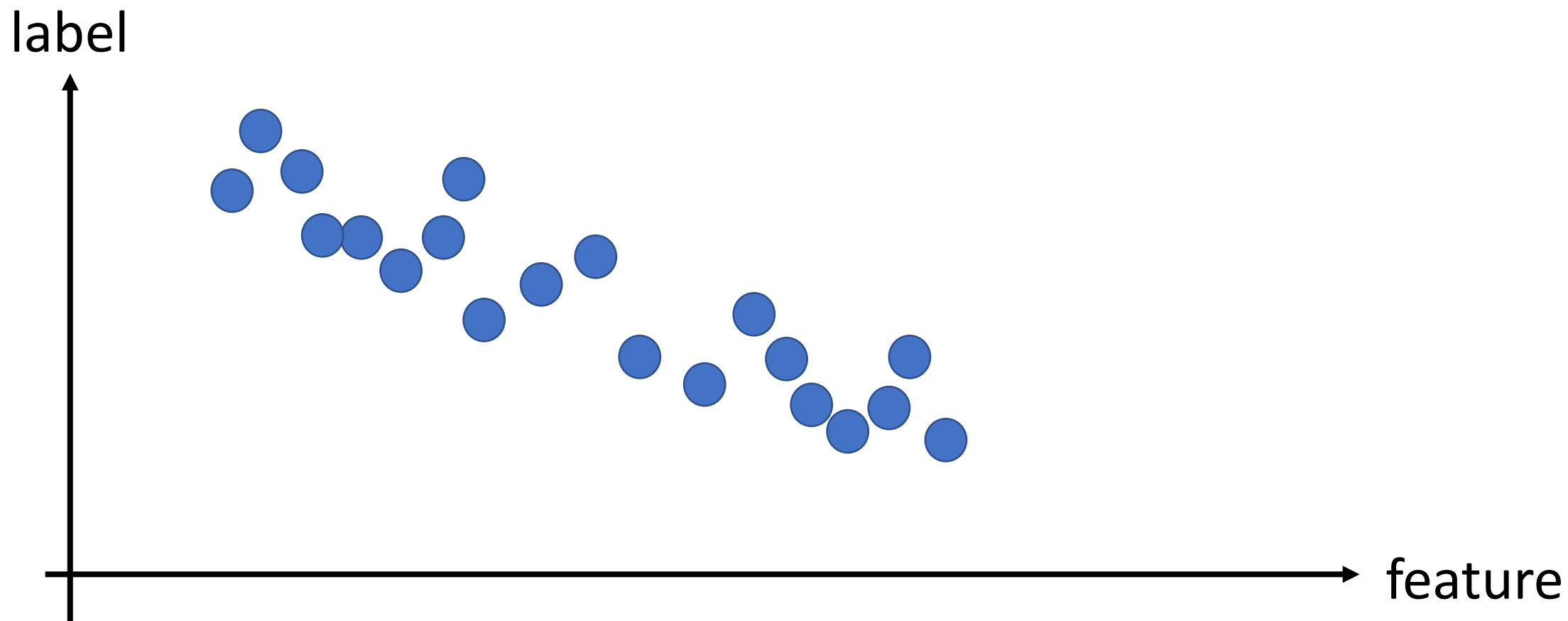
— *George E. P. Box* —

AZ QUOTES

Machine Learning.

“learn to **predict** the **label**
of a data point solely **from**
its features”

Scatterplot

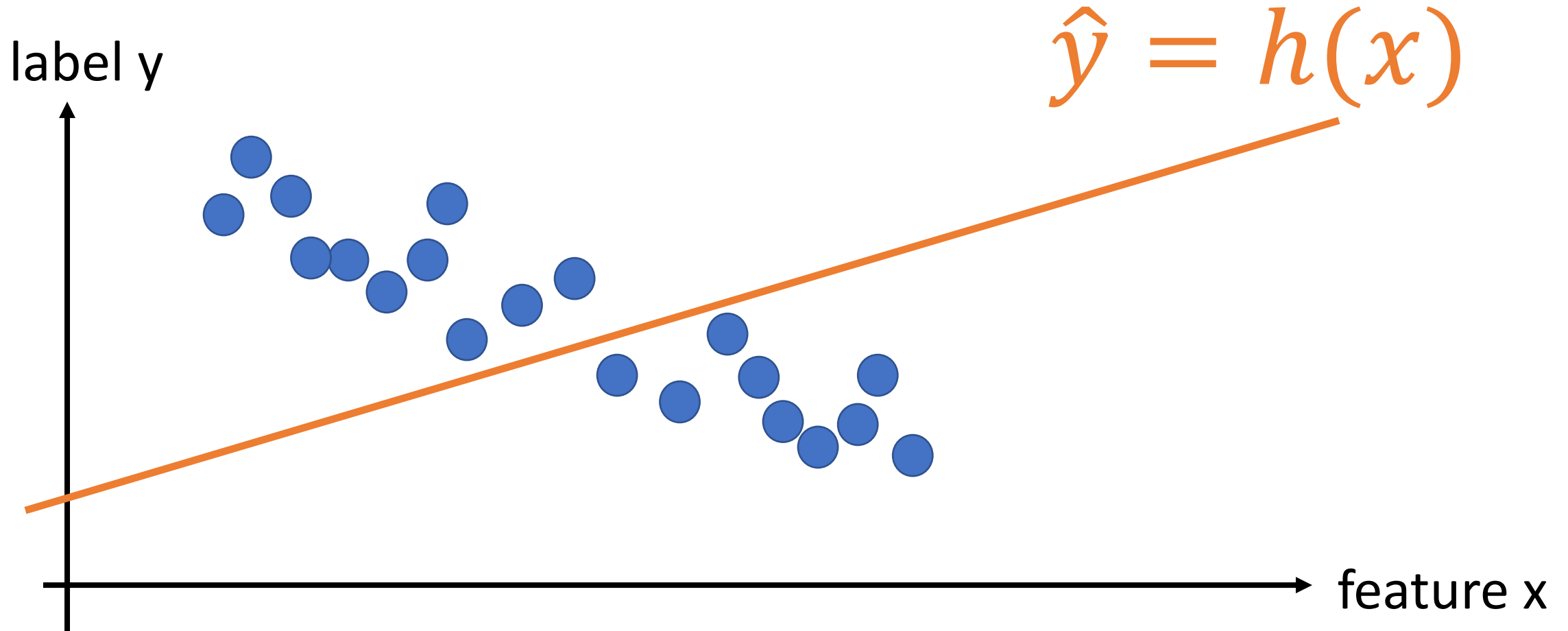


How to Predict?

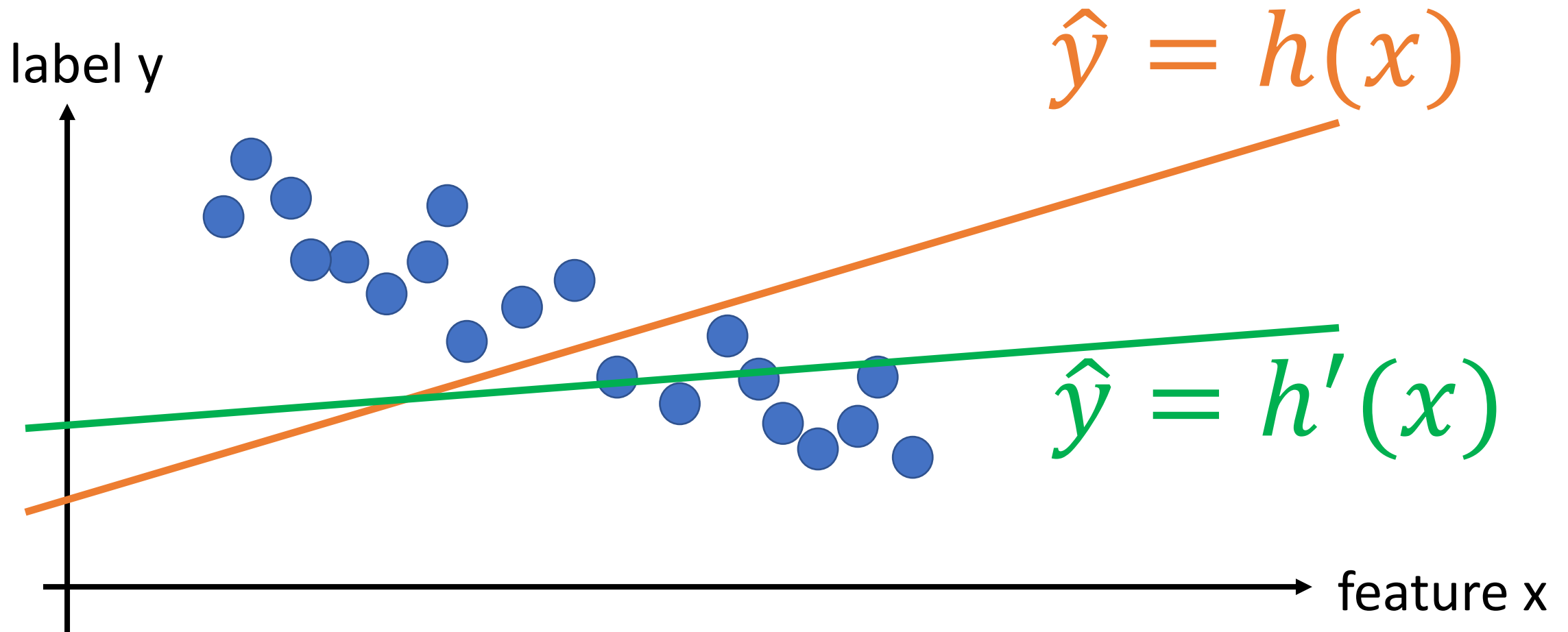
apply a hypothesis map h to features x ,

$$\hat{y} = h(x)$$

A Hypothesis.

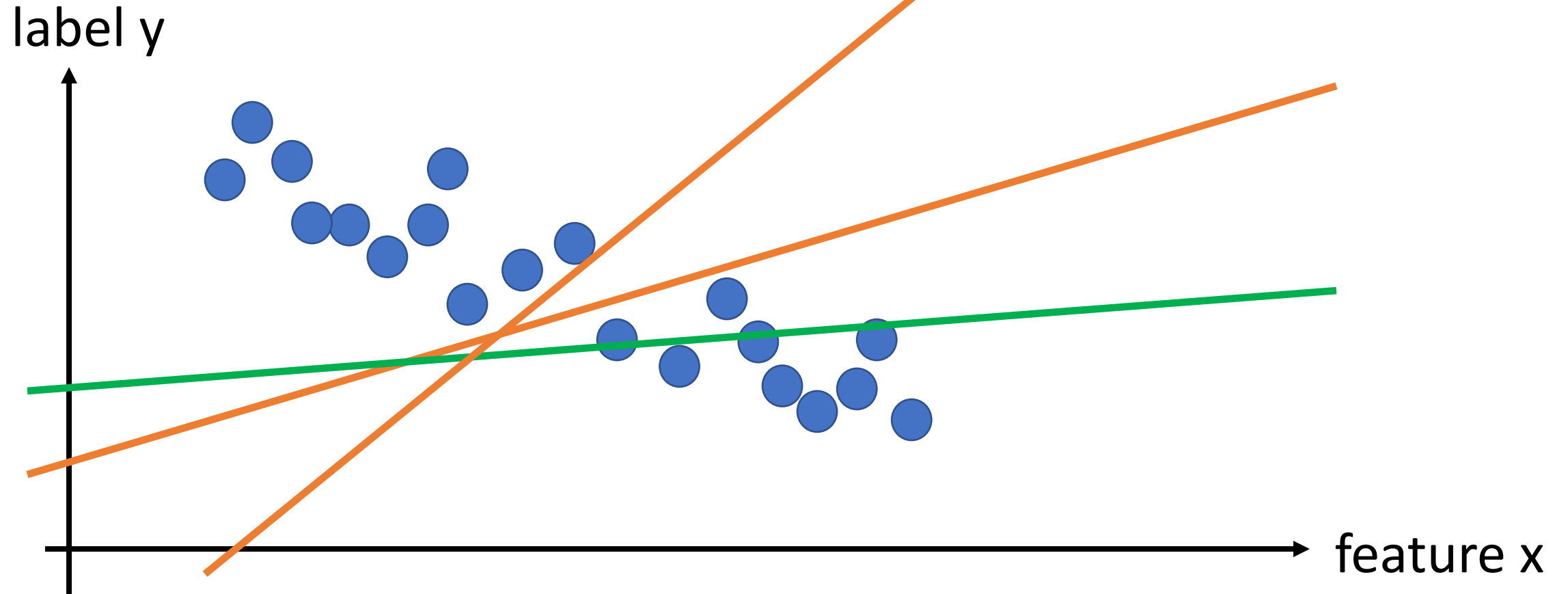


Model = Several Hypotheses.

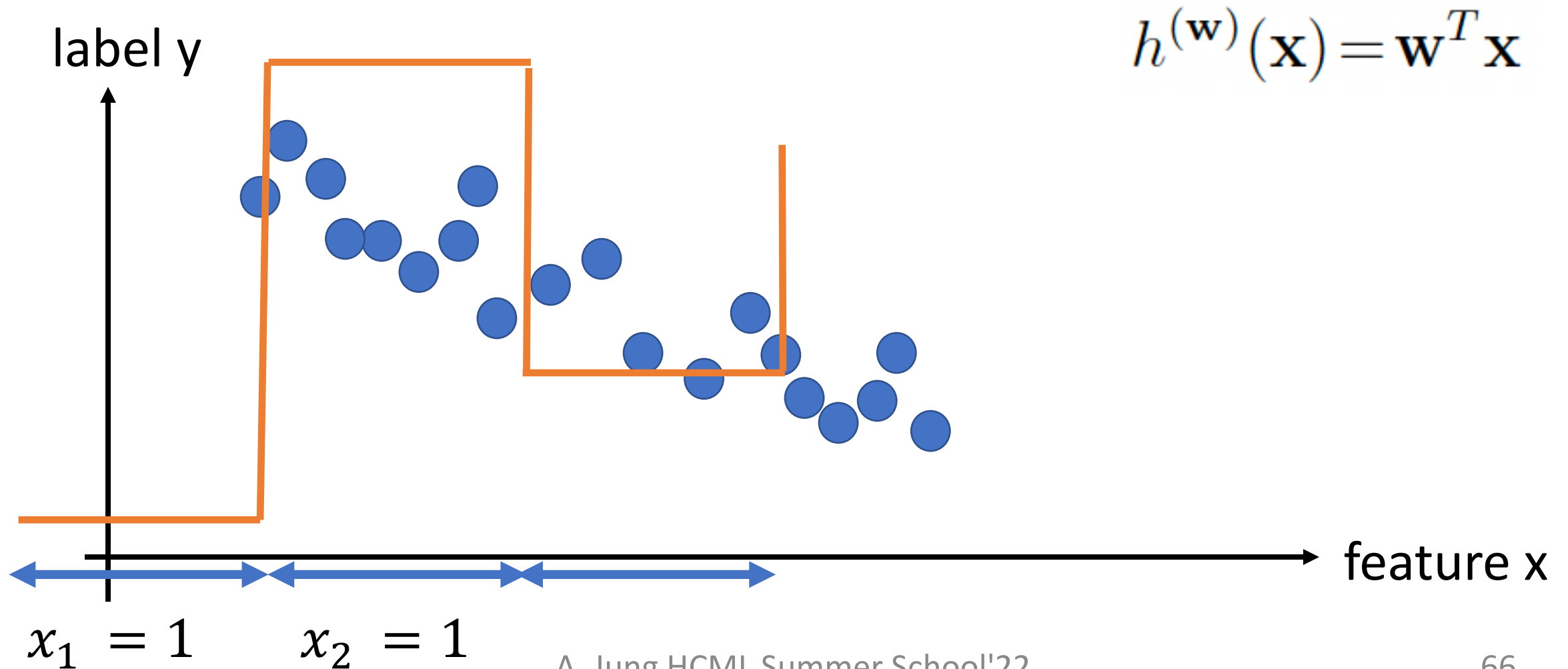


Linear Model

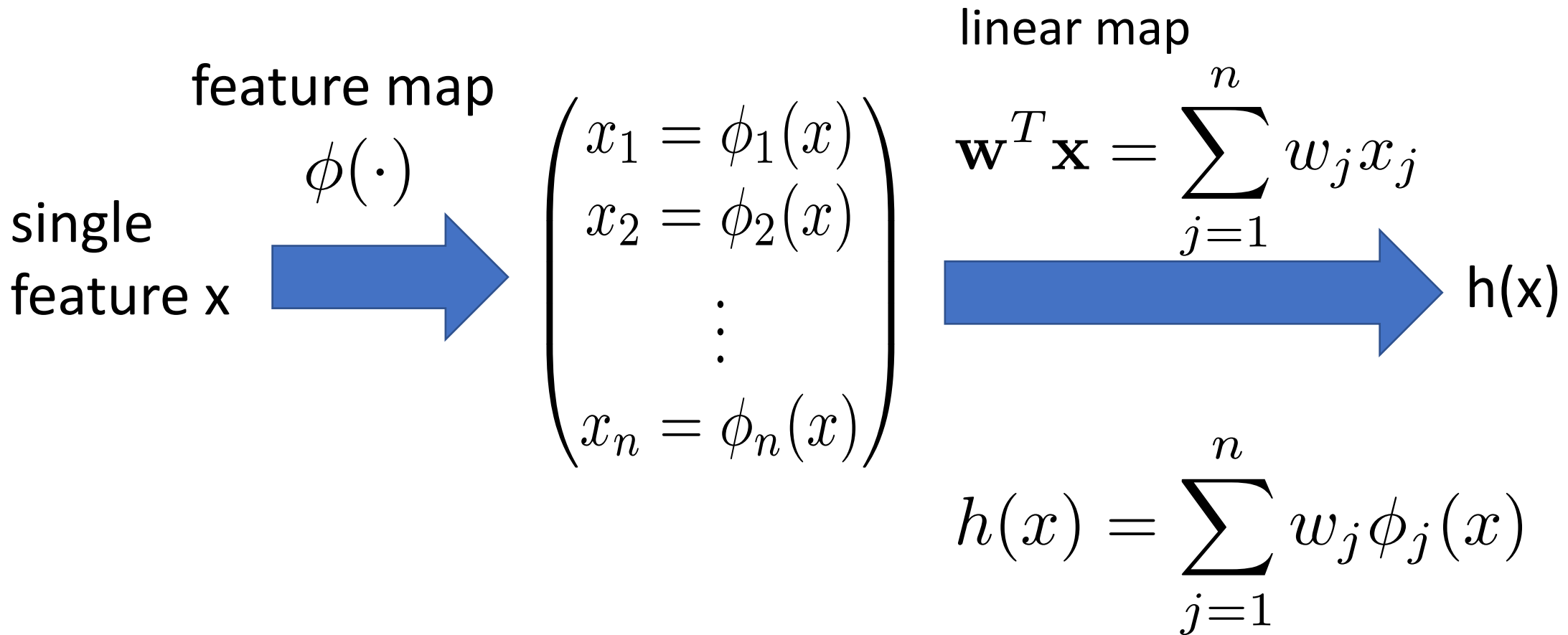
$$h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$



Linear Model is Versatile!

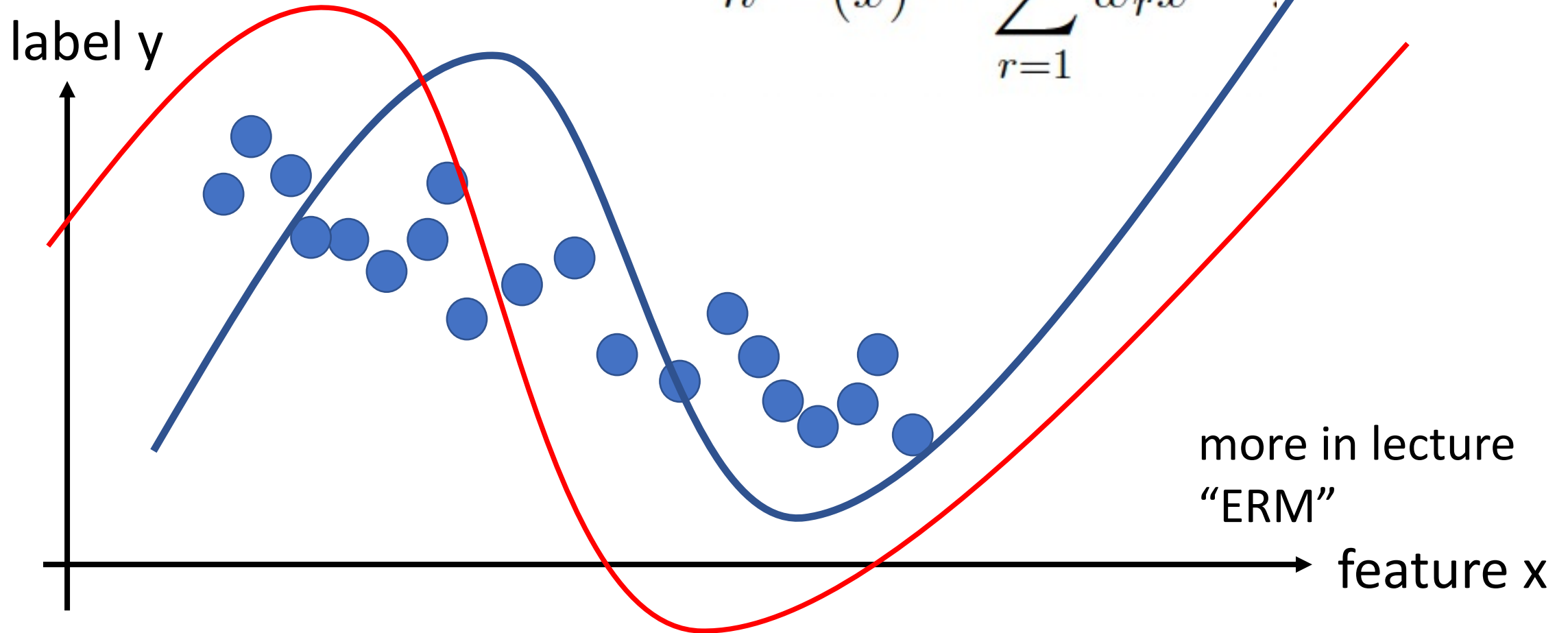


Linear + Feature Map

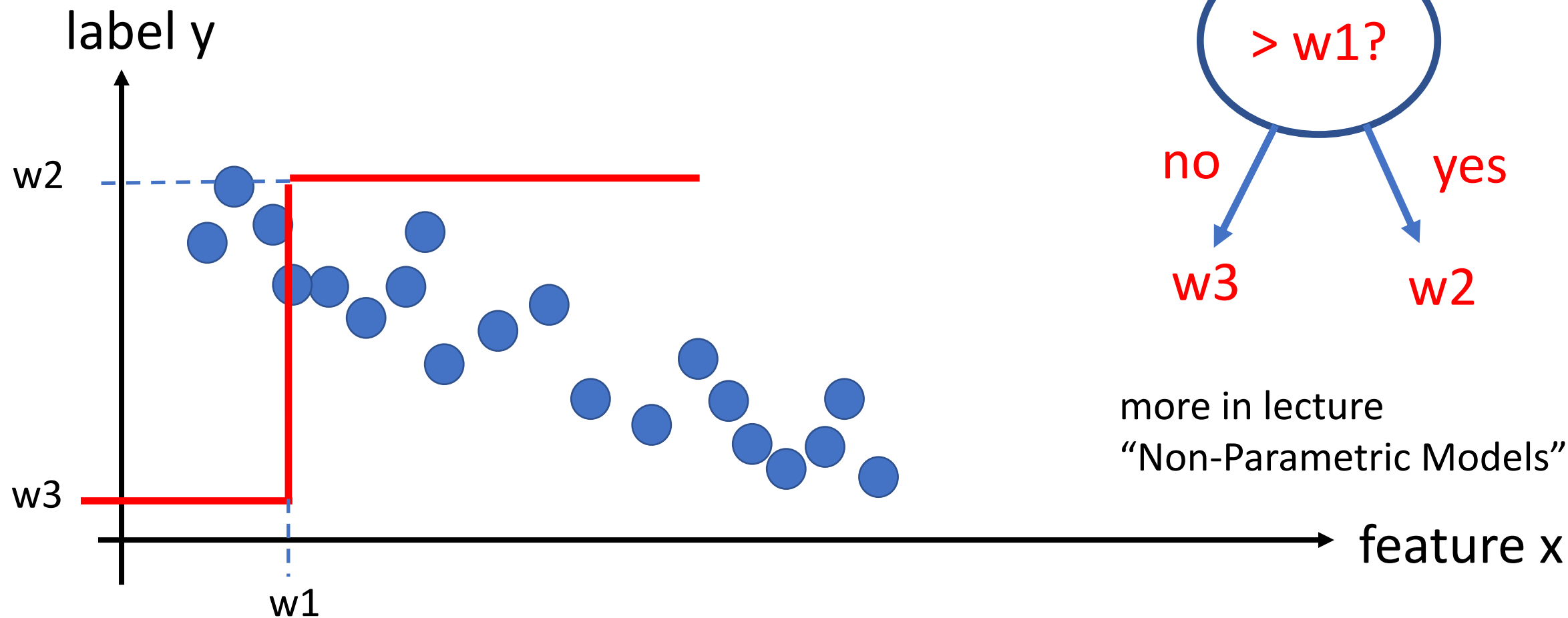


Polynomials

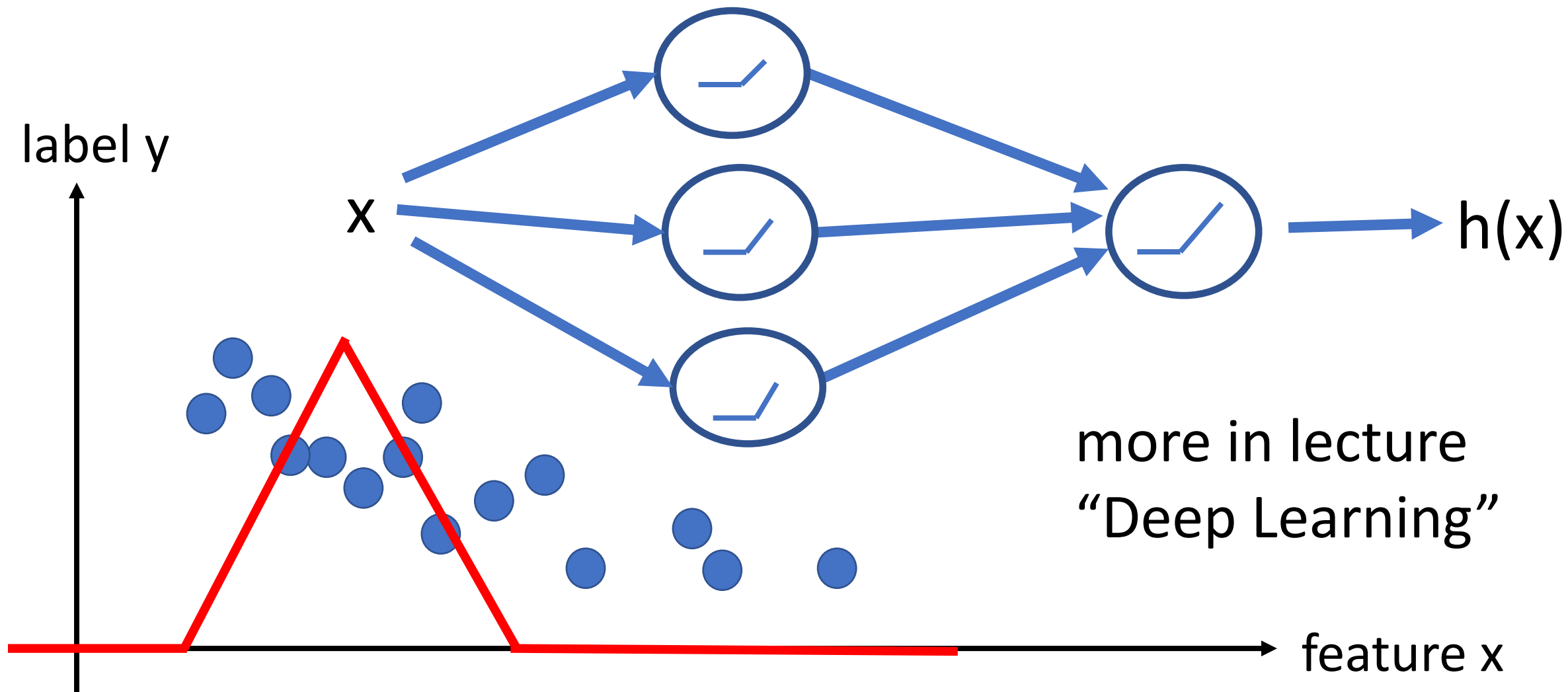
$$h^{(\mathbf{w})}(x) = \sum_{r=1}^n w_r x^{r-1}.$$



Decision Tree



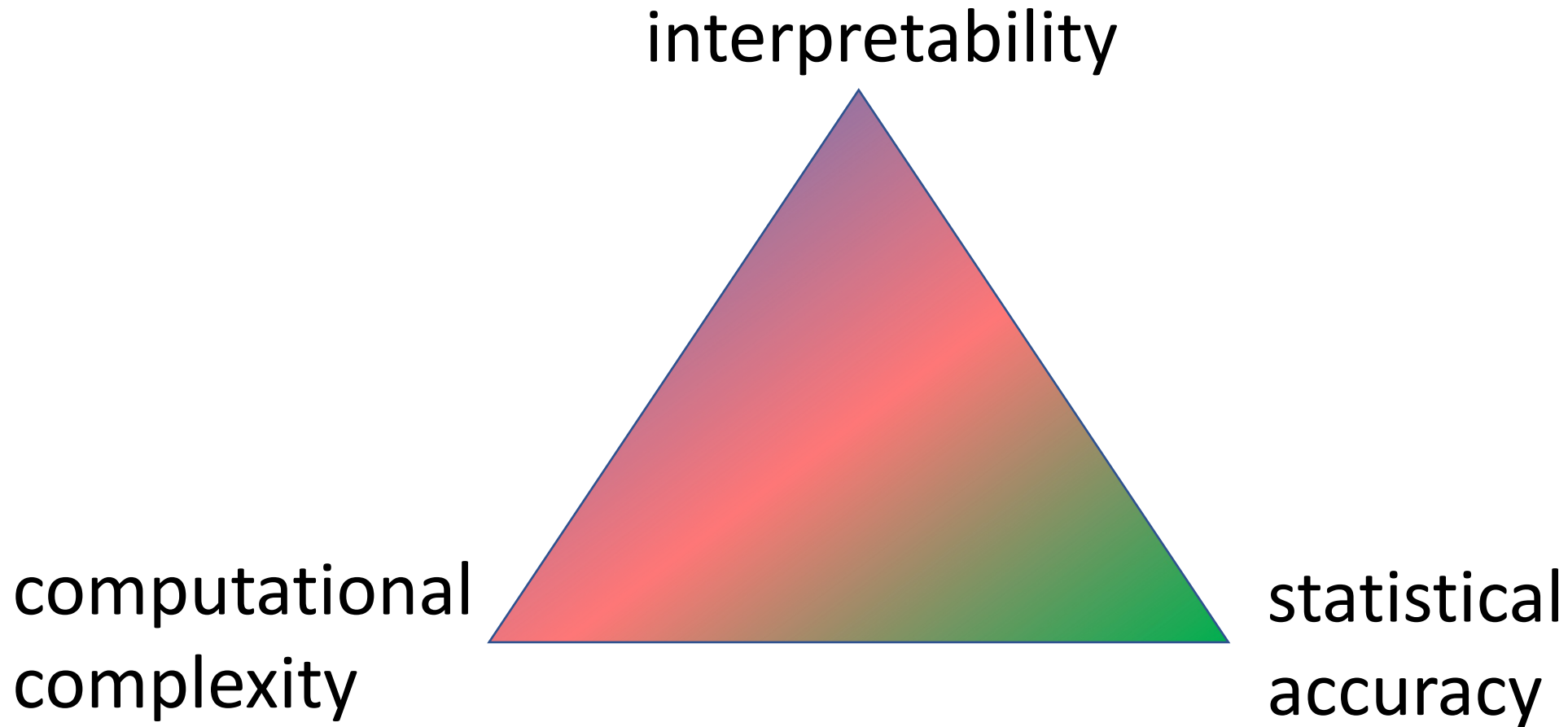
Artificial Neural Network



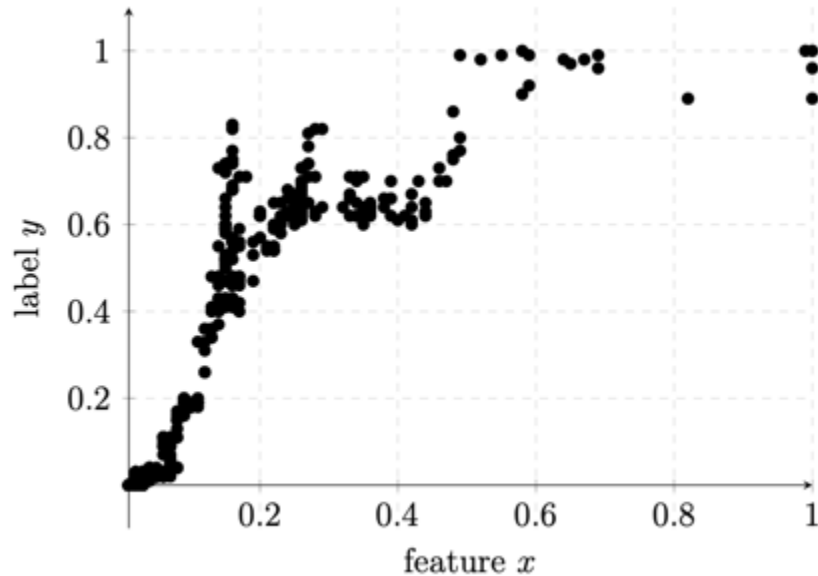
Which Model To Choose?

- large to offer a good hypothesis
- small to fit computational resources
- simple or interpretable

Design Choice: Model



Sufficiently Large



linear model might be too small for such data

there is no straight line that fits well the data points here

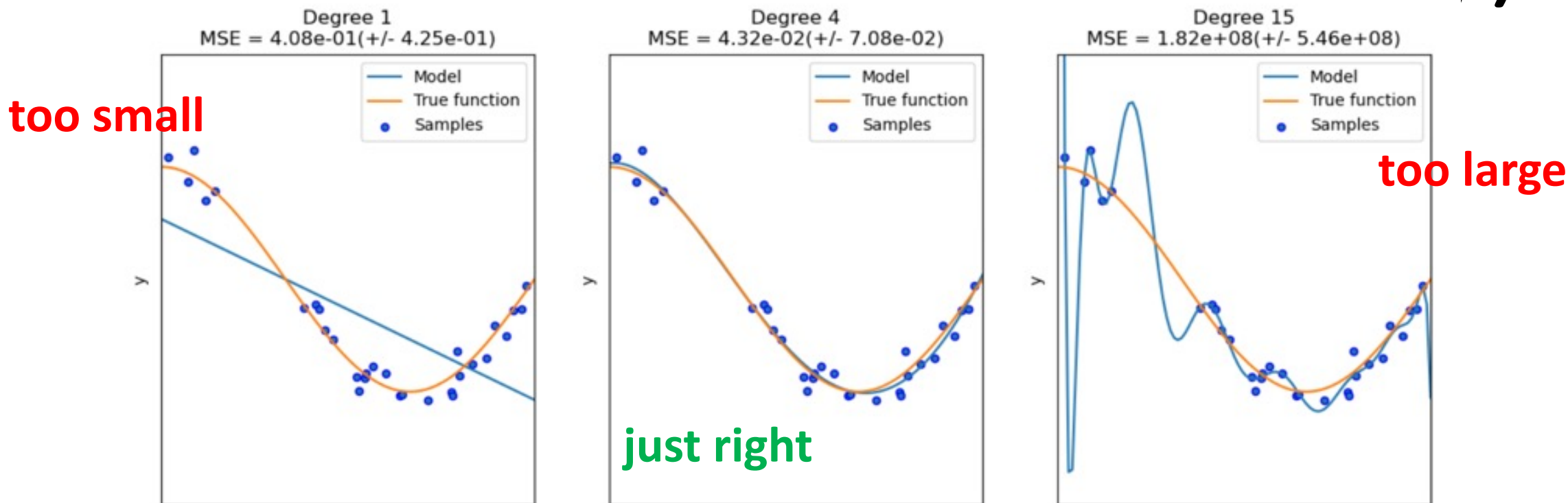
need larger models that also contain non-linear maps

more on large (non-linear) models in Lectures
“Deep Learning” and “Non-Parametric Models”

Sufficiently Small (Statistically)

- large models will always offer a hypothesis that fits training data (nearly) perfectly
- model fits well training data but does poor job outside the training data
- more on overfitting in Lecture “Model Val/Sel”

Sufficiently Small (Statistically)



source: https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Alex' rule of thumb:

training set (much) larger than # model parameters

Sufficiently Small (Comput.)

- consider linear model using n features
- fit linear model on $m > n$ datapoints
- need to invert “ n by n ” matrix ! [Sec. 4.3, MLBook]

Sufficiently Simple (Comput.)

- hypothesis maps $h(x)$ should be easy to evaluate
- recent MSc thesis on “Predicting Gas Valve Position”

need to compute $h(x)$ **in real-time** (while engine is running!)

Datapoint = A Cow

Features:

- Quantity of milk produced per day or over time
- Temperature of the milk

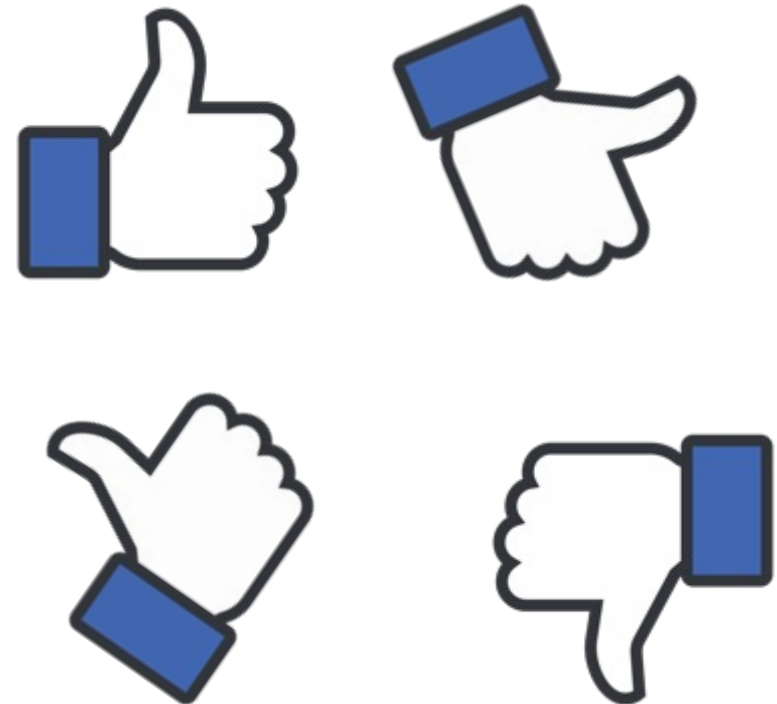


Labels:

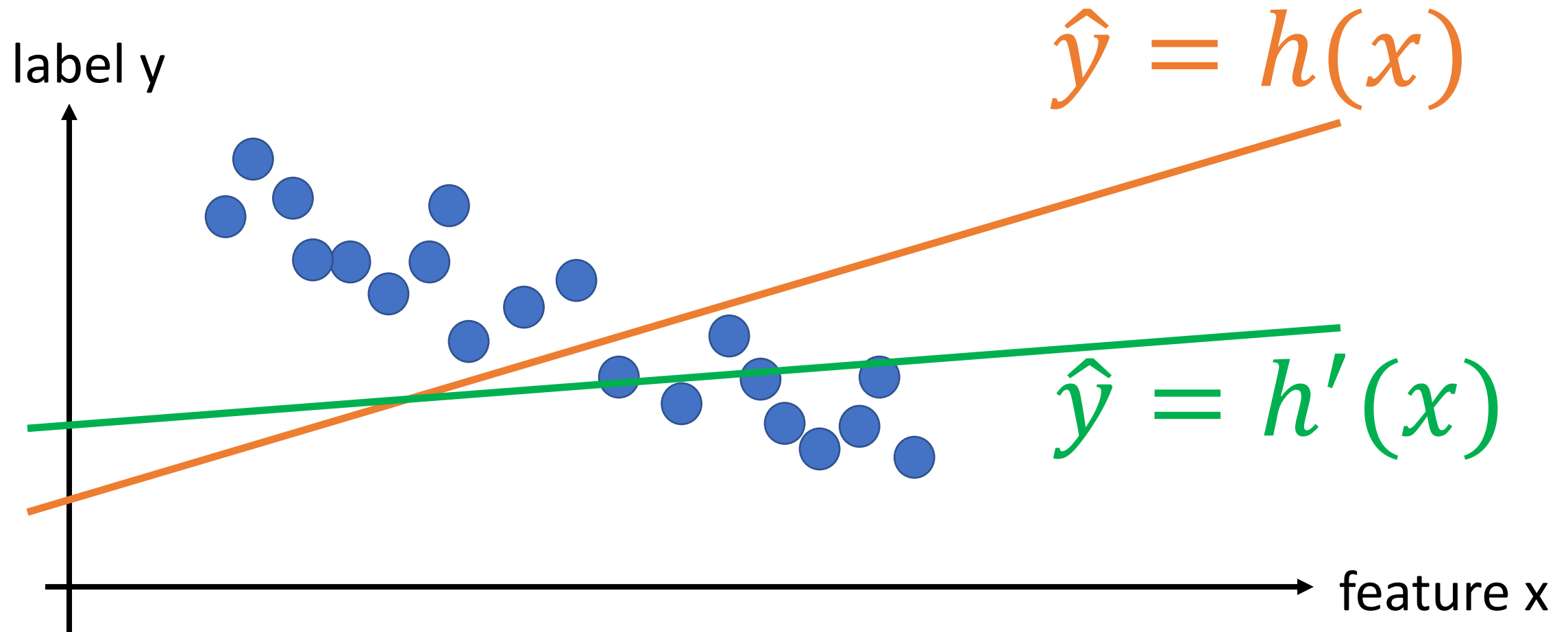
- Is the cow sick or not.
- Is it past its peak production time or not.
- Is it of species X or Y.

by Carlos Santos

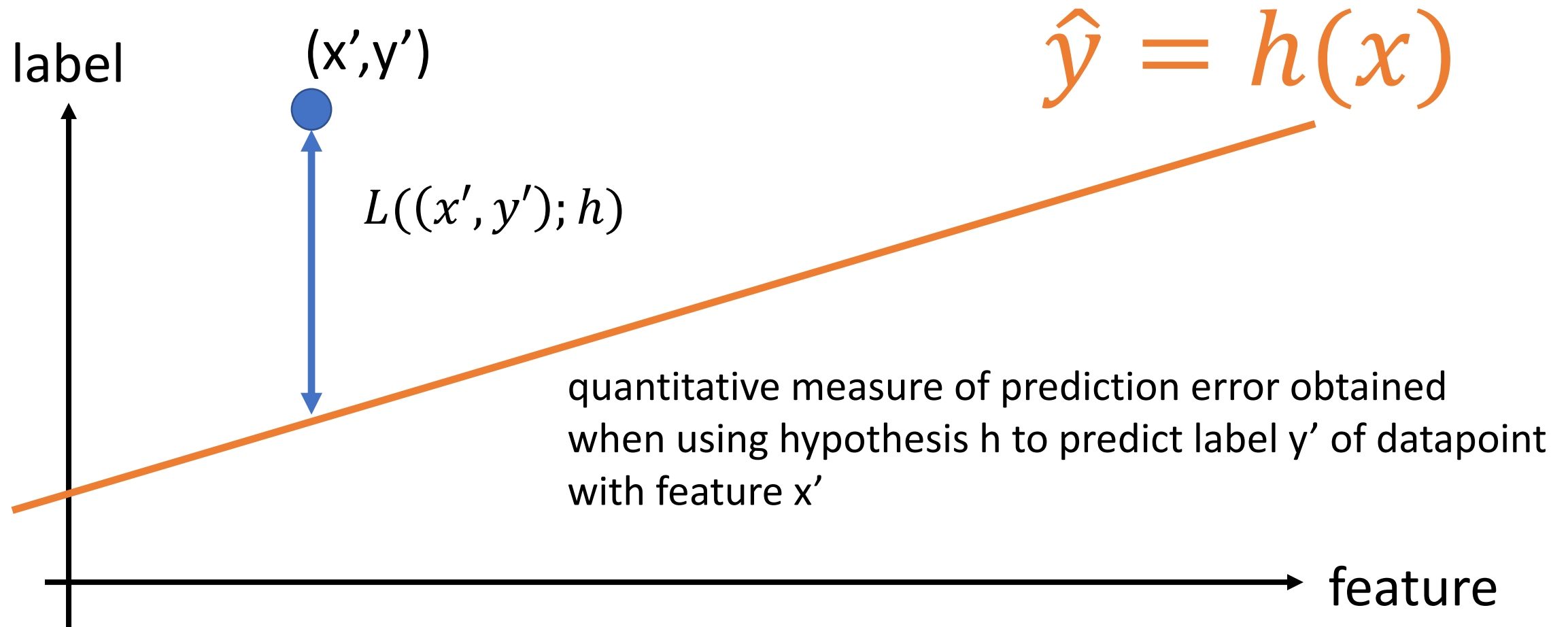
A. Jung HCML Summer School'22



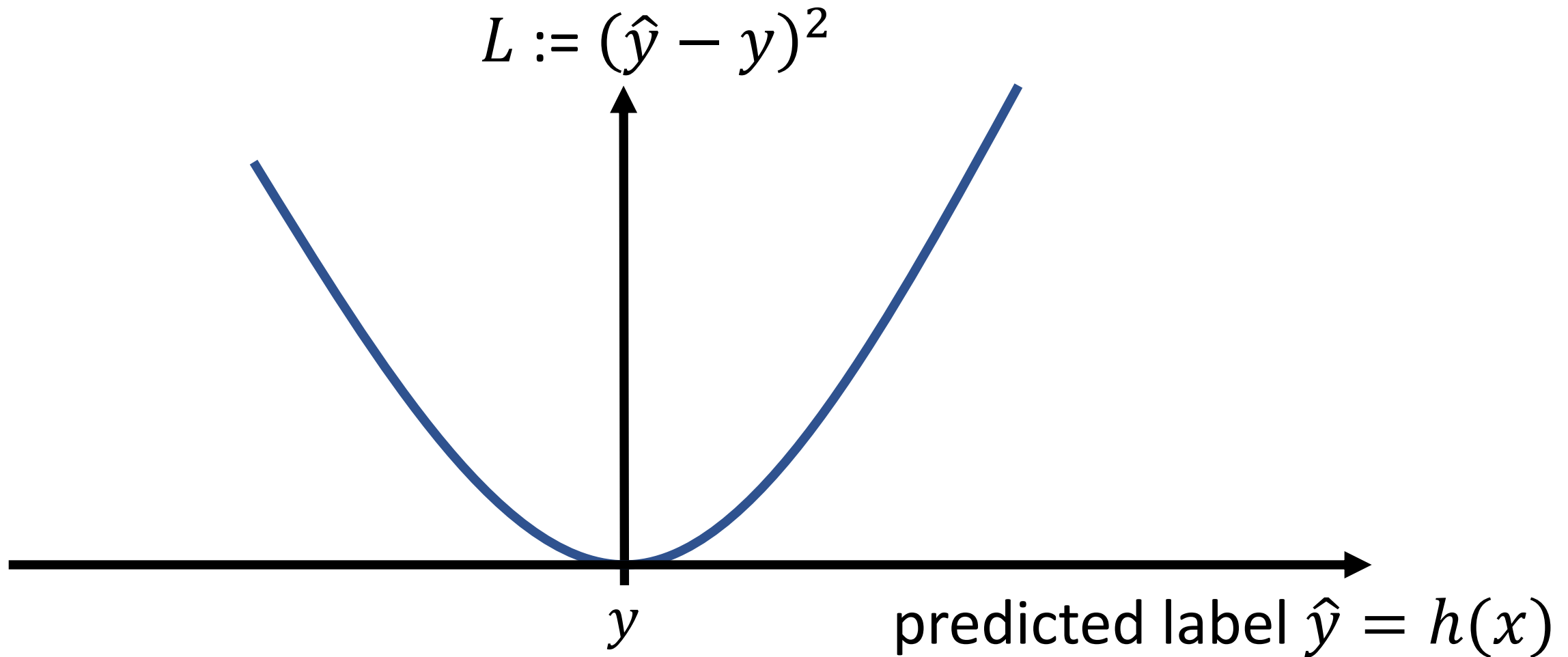
Which Hypothesis is Better?



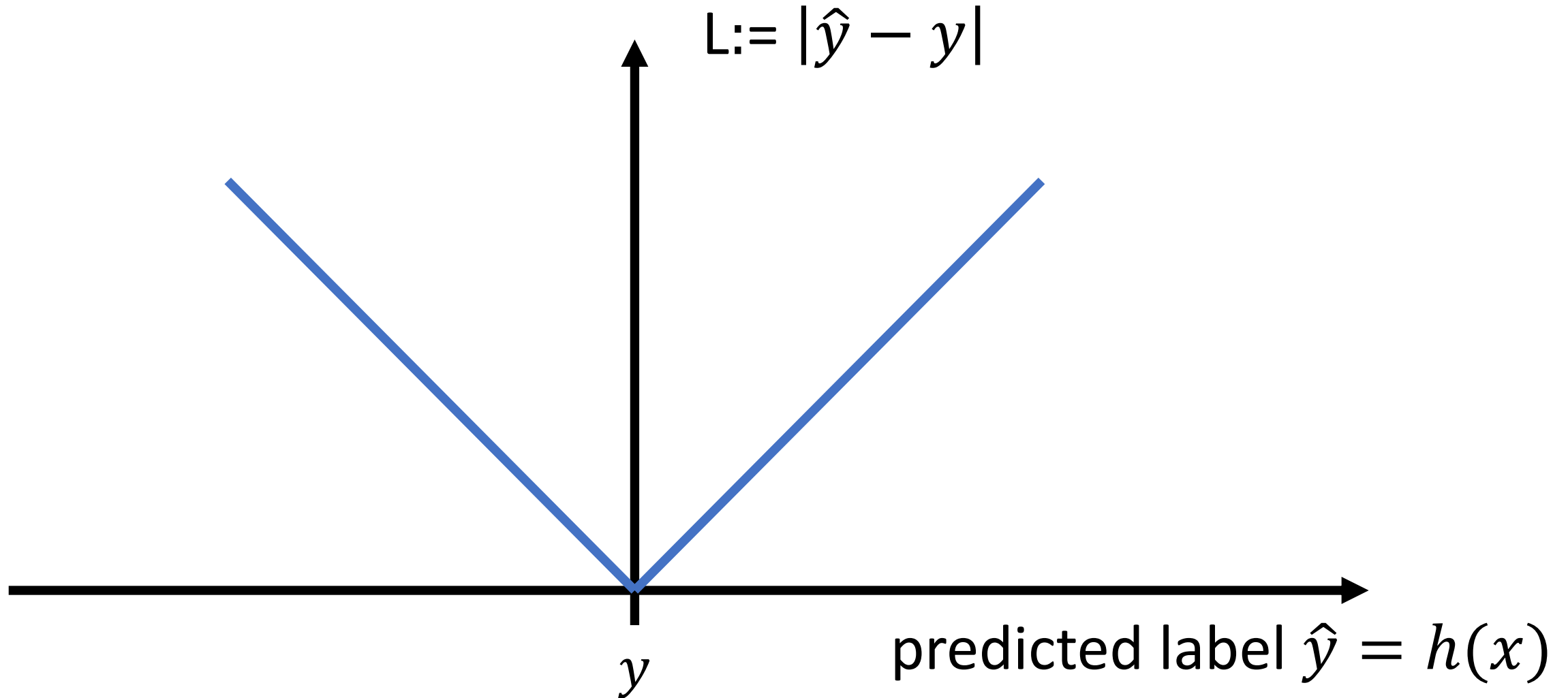
A Loss Function



The Squared Error Loss

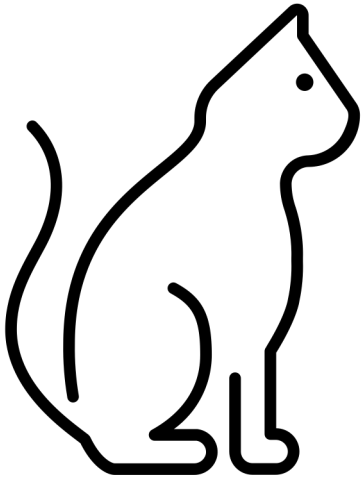


The Absolute Error Loss



Loss Functions for Binary Classification

label $y = \text{"cat"}$



features $x = \text{pixels}$

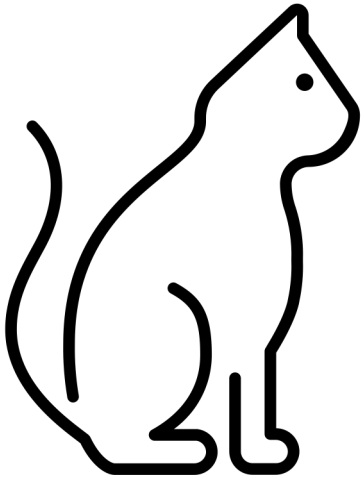


$h(x) = \text{"dog"}$

Loss = 100

Loss Functions for Binary Classification

label $y = \text{"cat"}$



features $x = \text{pixels}$



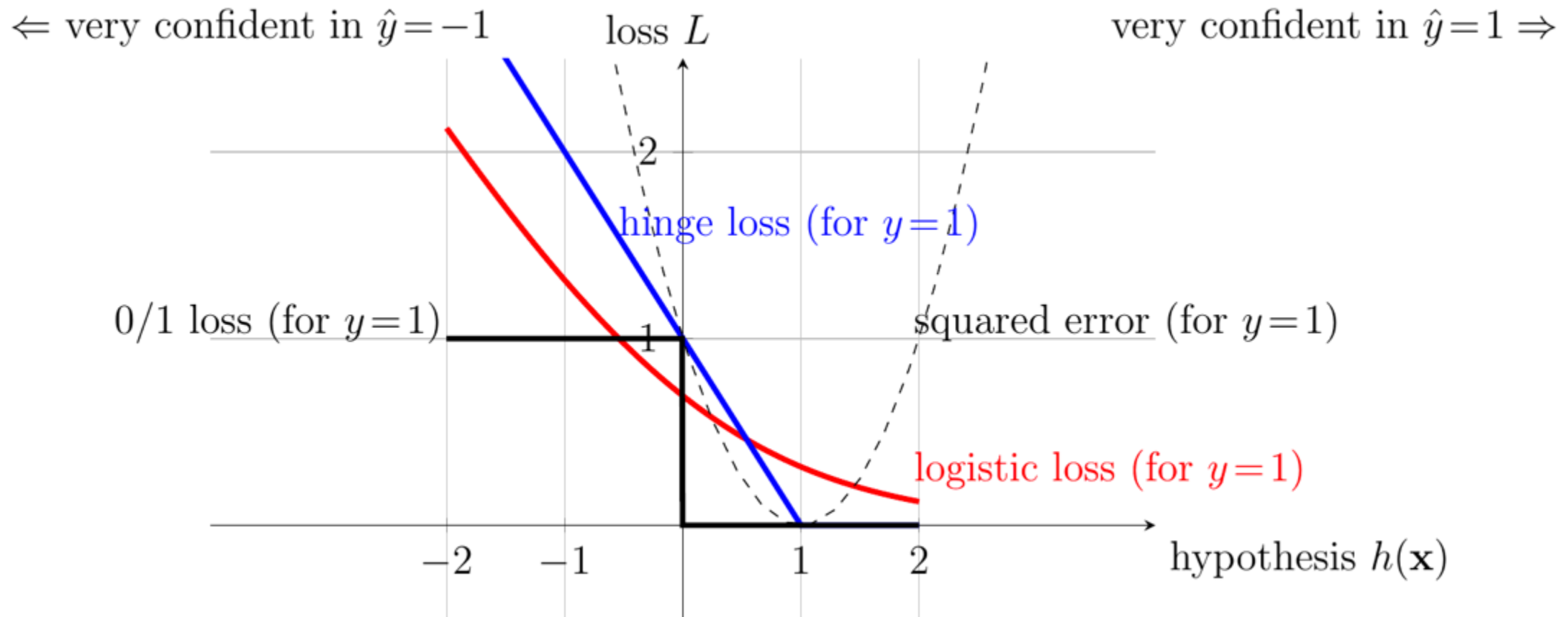
$h(x) = \text{"cat"}$

Loss = 0

Classifiers

- consider label values either “cat” or “dog”
- features vector x = pixels values
- can we use linear hypothesis maps $h(x)$?
- YES!
- use sign $h(x)$ to classify: $h(x) > 0 \rightarrow$ “dog”
- use $|h(x)|$ as confidence measure

Loss Functions for Binary Classification



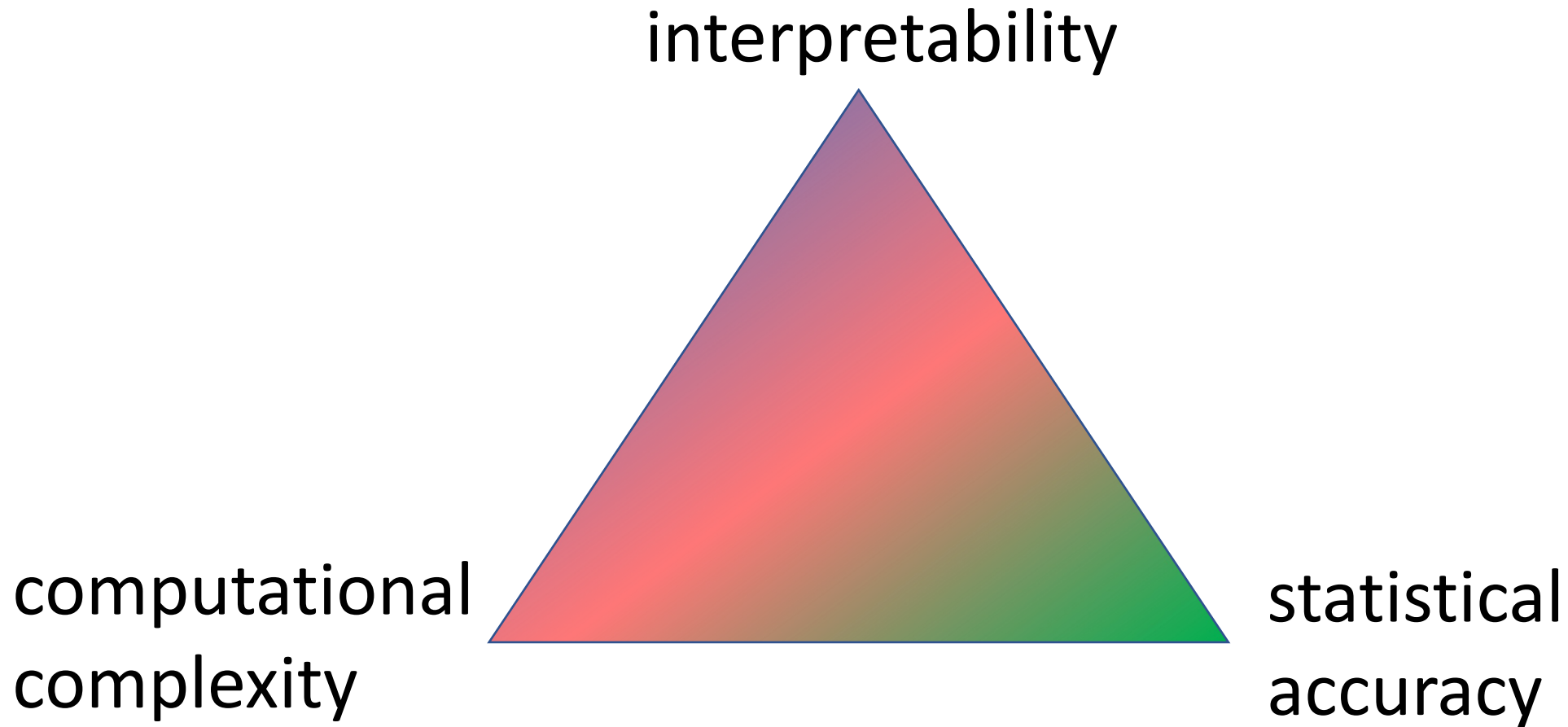
more on this in lecture “Classification”

Which Loss Function ?

- **statistical** aspects (should favour “reasonable” hypothesis)
- **computational** aspects (must be able to minimize them)
- **interpretation** (what does $\log\text{-loss} = -3$ mean ?)

.....choosing a suitable loss function is often non-trivial !

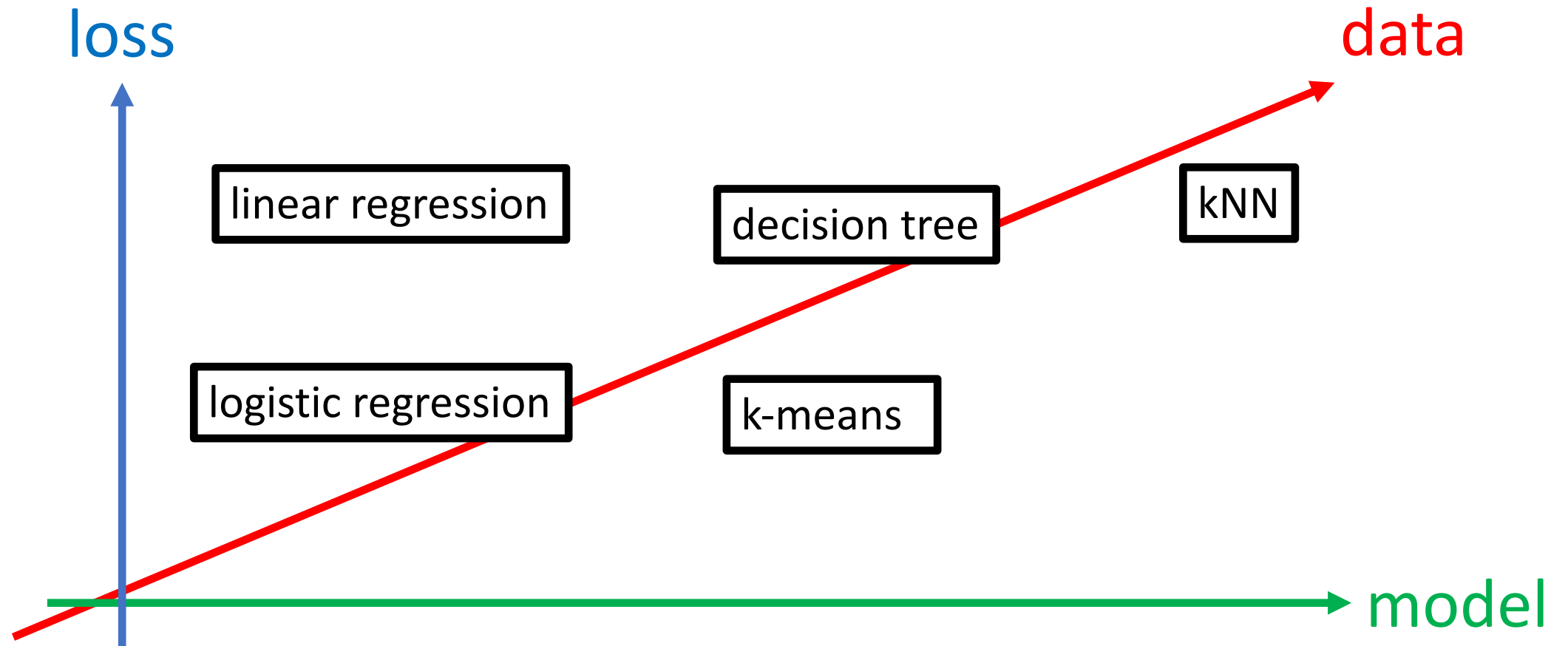
Design Choice: Loss



Main Components of ML

- data
- model
- loss

Landscape of ML Methods



ML Method: Linear Regression

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> #  $y = 1 * x_0 + 2 * x_1 + 3$ 
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

data



model, loss



ML Method: Decision Tree Classifier

```
[11]: from sklearn.datasets import load_iris
      from sklearn.tree import DecisionTreeClassifier

      iris = load_iris()

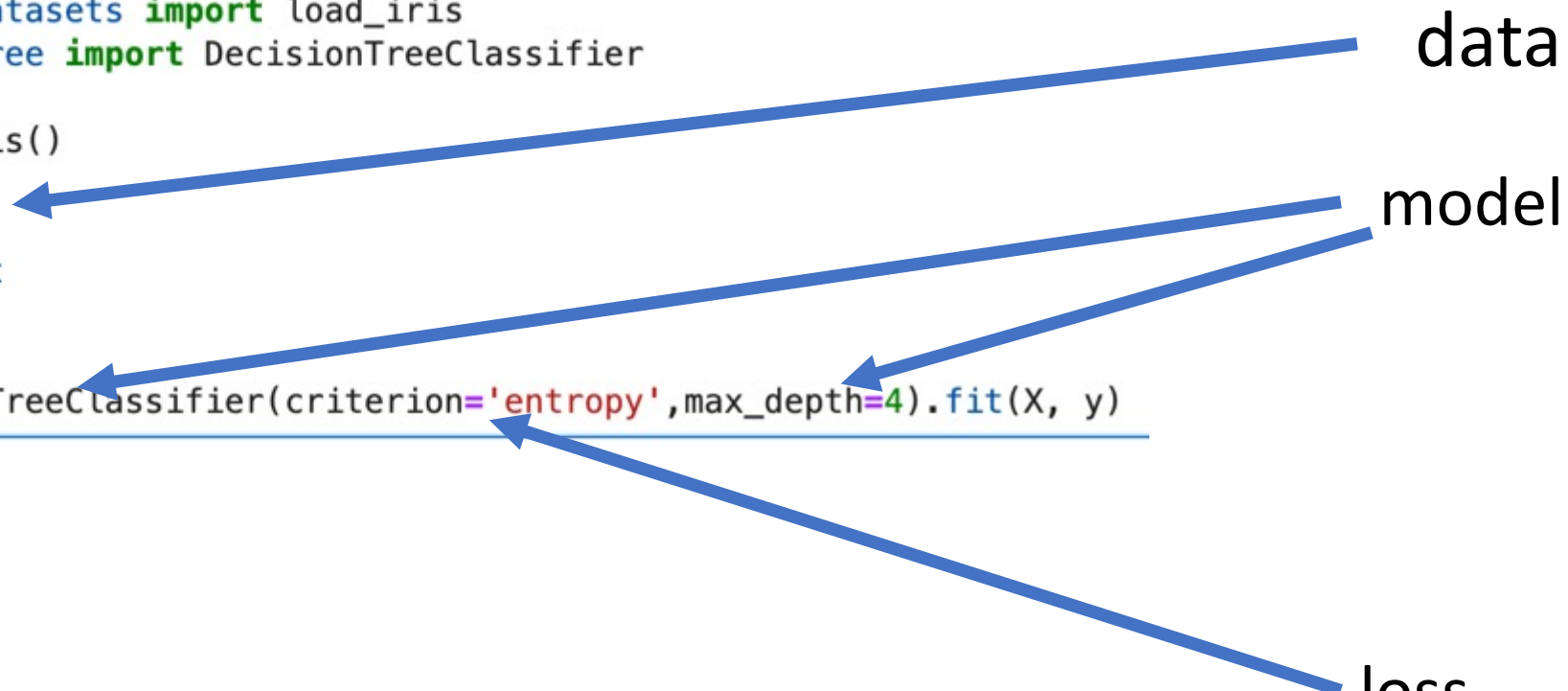
      X = iris.data
      y = iris.target

      # Train
      clf = DecisionTreeClassifier(criterion='entropy', max_depth=4).fit(X, y)
```

data

model

loss



ML Method: Deep Learning

```
# -*- coding: utf-8 -*-
```

```
import torch
```

```
import math
```

```
# Create Tensors to hold input and outputs.
```

```
x = torch.linspace(-math.pi, math.pi, 2000)
```

```
y = torch.sin(x)
```

```
# Prepare the input tensor (x, x^2, x^3).
```

```
p = torch.tensor([1, 2, 3])
```

```
xx = x.unsqueeze(-1).pow(p)
```

```
# Use the nn package to define our model and loss function.
```

```
model = torch.nn.Sequential(
```

```
    torch.nn.Linear(3, 1),
```

```
    torch.nn.Flatten(0, 1)
```

```
)
```

```
loss_fn = torch.nn.MSELoss(reduction='sum')
```

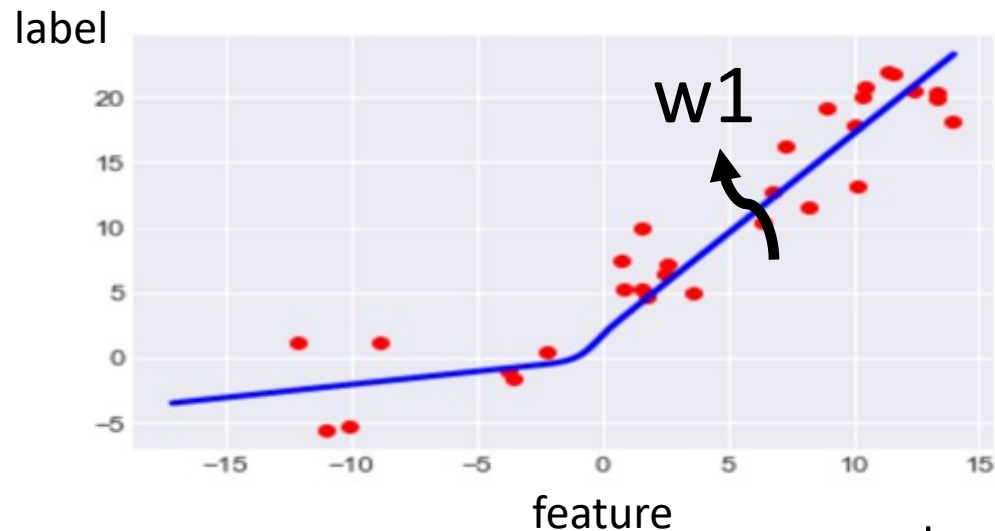
data

model

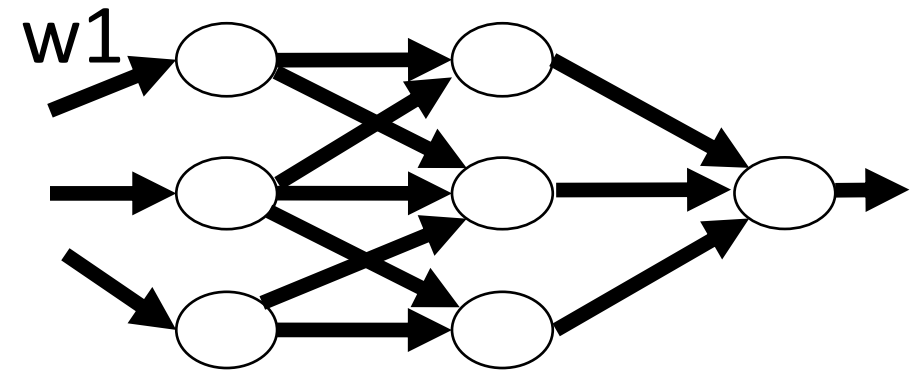
loss

Three Views on Machine Learning

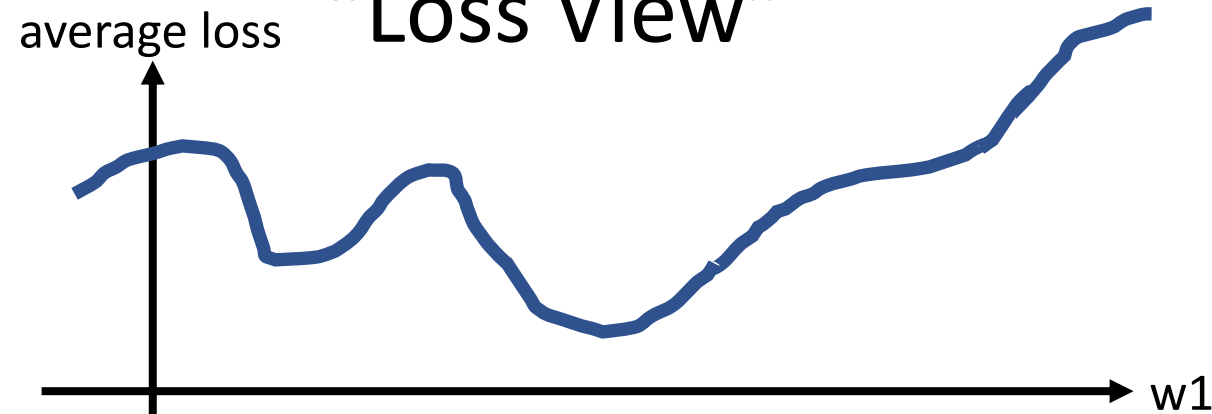
“Data View”



“Model View”



“Loss View”



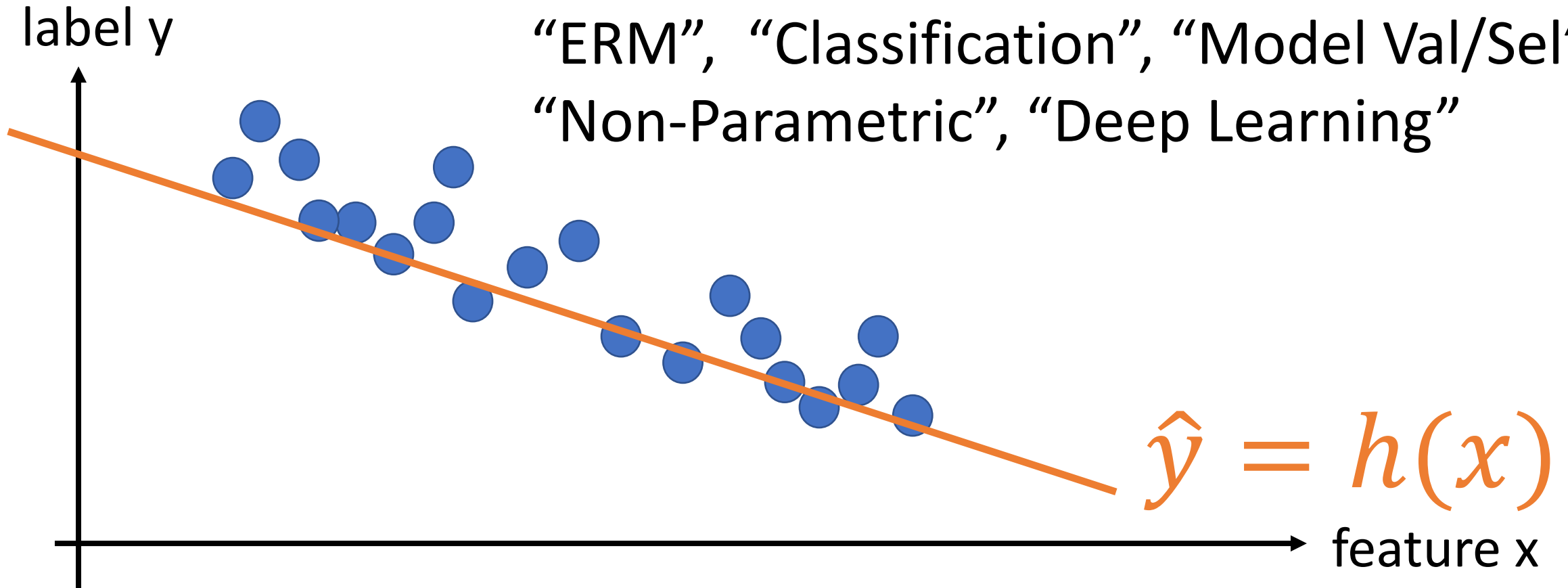
Three Main Flavours of ML

- supervised ML (use labeled data to imitate teacher)
- unsupervised ML (no labeled data needed)
- reinforcement learning (learn while collecting data)

Supervised Learning

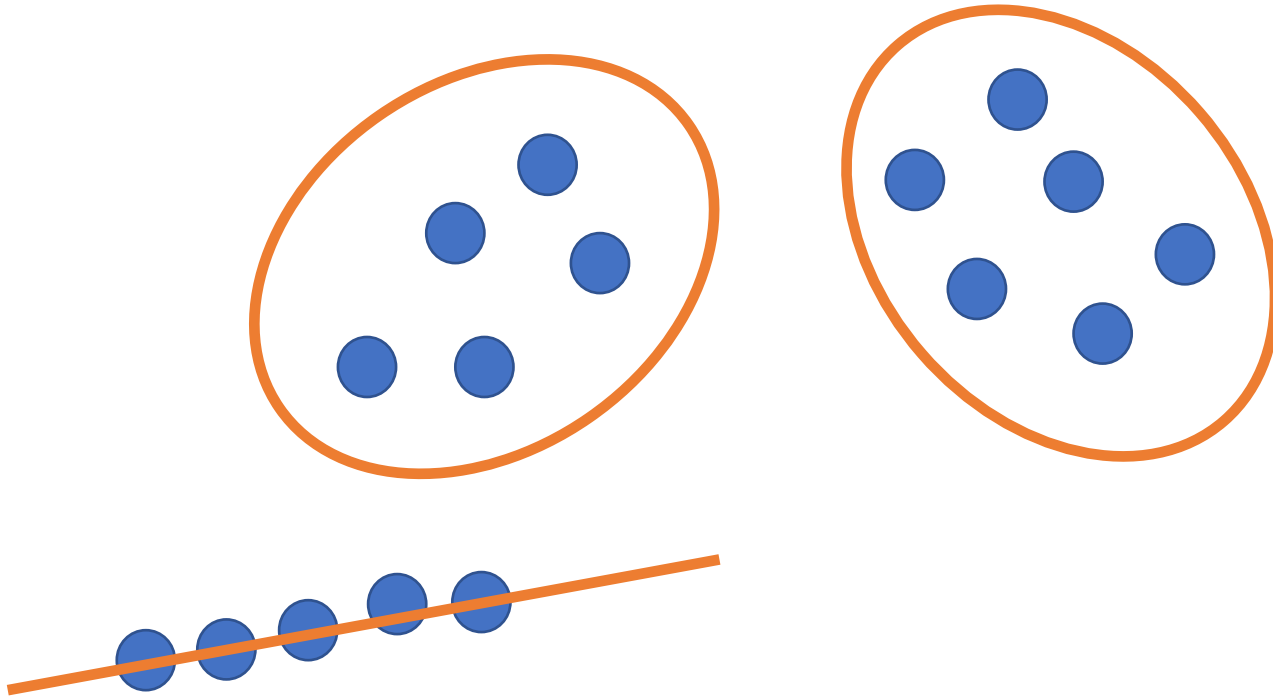
more on this in lecture:

“ERM”, “Classification”, “Model Val/Sel”,
“Non-Parametric”, “Deep Learning”



Unsupervised Learning

more on this in Lecture
“Clustering”,
“Feature Learning”



label of datapoint = cluster assignment or
nearby subspace

Reinforcement Learning

features = on-board
camera video

label = “optimal steering
direction”



not covered in this school !

Wrap Up

- **data points** characterized by features and label
- features \approx low-level properties
- labels \approx high-level properties (quantity of interest)
- GOAL of ML: learn a hypothesis map $h(\cdot)$ such that $h(x) \approx y$
- ML **model** = comp. tractable subset of possible maps $h(\cdot)$
- ML quantifies prediction error $y-h(x)$ with a **loss function**

Next Lecture: Regression

GOAL of ML: Learn hypothesis $h(\cdot)$ such that $y \approx h(x)$ for any data point (x,y) .

what exactly is “any data point” ?