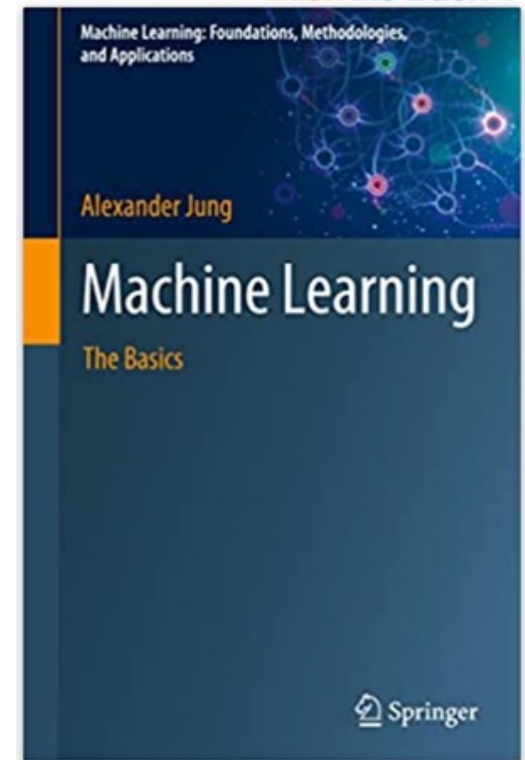# Regression

Alex(ander) Jung
Assistant Professor for Machine Learning
Department of Computer Science
Aalto University

Reading.

- Chapter 3.1-3.2 of AJ, "Machine Learning: The Basics", Springer, 2022.https://mlbook.cs.aalto.fi

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Learning Goals:

- know about notion of expected loss or risk

- know that average loss approximates risk

- know about empirical risk minimization

- know some regression methods

- know comp./stat. trade offs for diff. loss func.

What is ML About ?

fit models to data to make

predictions or forecasts !

Data. Model. Loss.

data: set of data points (x,y)

model: set of hypothesis maps h(.)

loss: quality measure L((x,y),h)

Data

	Year	m	d	Time	Time zone	Maximum temperature (degC)	Minimum temperature (degC)
0	2020	2	1	00:00	UTC	3.0	1.9
1	2020	2	2	00:00	UTC	4.9	2.4
2	2020	2	3	00:00	UTC	2.6	-0.4
3	2020	2	4	00:00	UTC	-0.2	-3.7
4	2020	2	5	00:00	UTC	2.5	-4.2
5	2020	2	6	00:00	UTC	2.4	-4.7
6	2020	2	7	00:00	UTC	1.2	-5.5
7	2020	2	8	00:00	UTC	2.7	0.2
8	2020	2	9	00:00	UTC	3.9	2.6

Data.

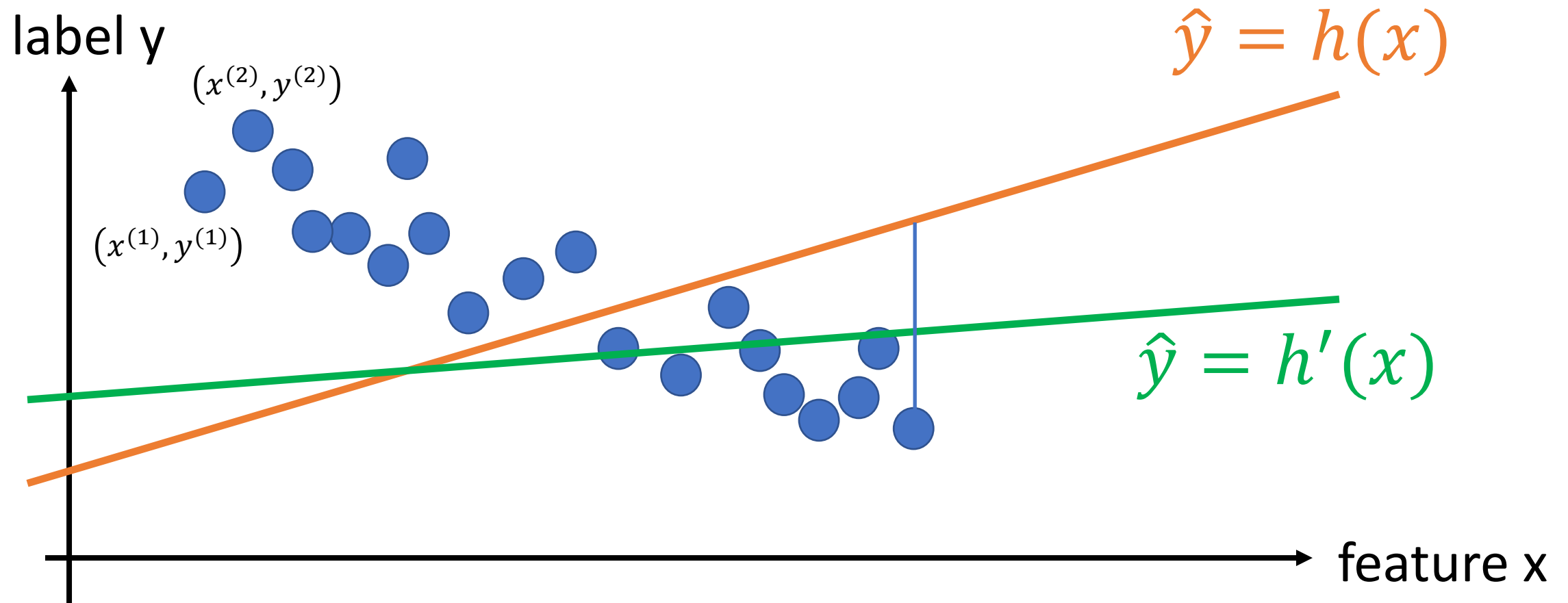$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}.$$

	Year	m	d	Time	Time zone	Maximum temperature (degC)	Minimum temperature (degC)
0	2020	2	1	00:00	UTC	3.0	1.9
1	2020	2	2	00:00	UTC	4.9	2.4
2	2020	2	3	00:00	UTC	2.6	-0.4
3	2020	2	4	00:00	UTC	-0.2	-3.7
4	2020	2	5	00:00	UTC	2.5	-4.2
5	2020	2	6	00:00	UTC	2.4	-4.7
6	2020	2	7	00:00	UTC	1.2	-5.5
7	2020	2	8	00:00	UTC	2.7	0.2
8	2020	2	9	00:00	UTC	3.9	2.6

stack feature vecs into matrix

$$\mathbf{X} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right)^T \in \mathbb{R}^{m \times n}$$
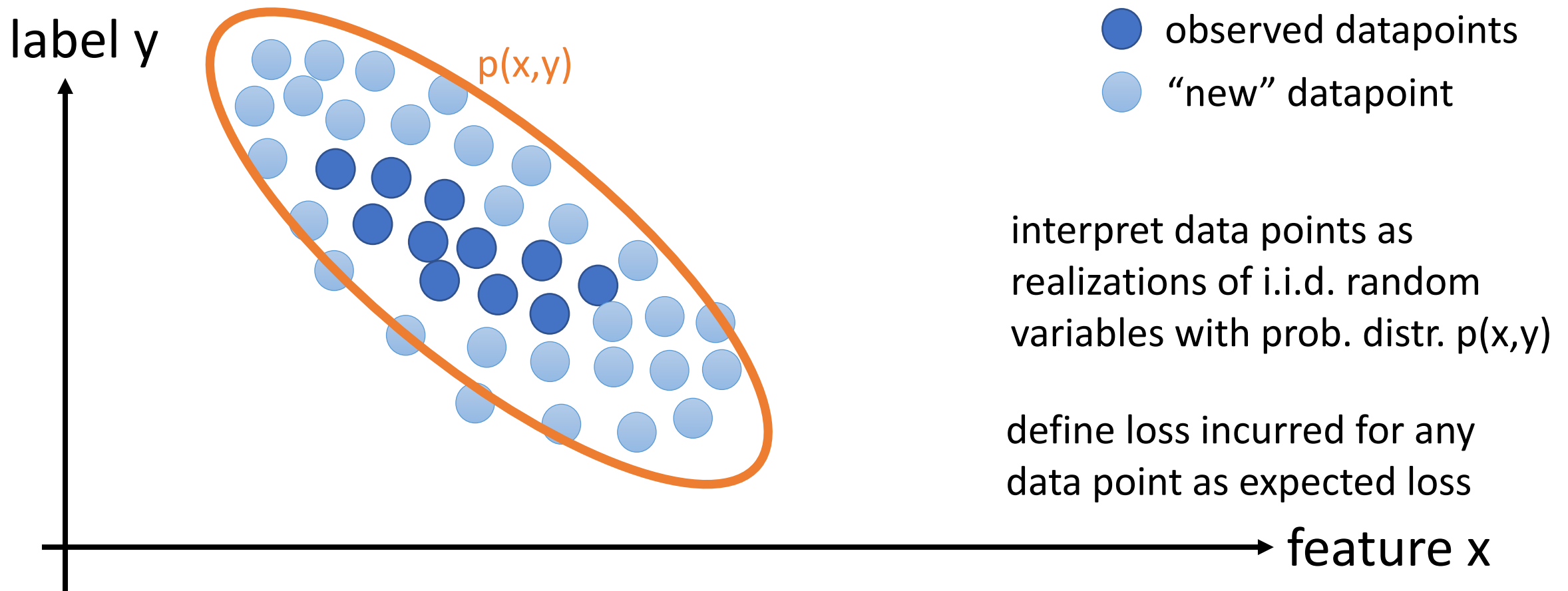
stack labels into vector

$$\mathbf{y} = (y^{(1)}, \ldots, y^{(m)})^T \in \mathbb{R}^m$$

label y

$(x^{(2)}, y^{(2)})$

$(x^{(1)}, y^{(1)})$

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

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

feature x

A. Jung - HCML Summer School'22

Machine Learning.

find hypothesis in model that incurs

smallest loss when predicting label of

any datapoint

What is Any Datapoint?

label y

p(x,y)

● observed datapoints

● "new" datapoint

interpret data points as realizations of i.i.d. random variables with prob. distr. p(x,y)

define loss incurred for any data point as expected loss

feature x

Expected Loss or Risk

$$\mathbb{E}\{L((\mathbf{x}, y), h)\} := \int_{\mathbf{x}, y} L((\mathbf{x}, y), h)\, dp(\mathbf{x}, y). \tag{2.14}$$

note: to compute this expectation
we need to know the probability distribution
p(x,y) of datapoints (x,y)

Empirical Risk

IDEA: approximate expected loss by average loss on some datapoints (training set)

$$\mathcal{D} = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(m)}, y^{(m)}\right) \right\}.$$

$$\mathbb{E}\left\{ L\left((\mathbf{x}, y), h\right) \right\} \approx (1/m) \sum_{i=1}^{m} L\left((\mathbf{x}^{(i)}, y^{(i)}), h\right) \text{ for sufficiently large sample size } m. \quad (2.17)$$

with the average loss or empirical risk

$$\widehat{L}(h|\mathcal{D}) = (1/m) \sum_{i=1}^{m} L\left((\mathbf{x}^{(i)}, y^{(i)}), h\right). \quad (2.16)$$

Empirical Risk Minimization

$$\hat{h} \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \widehat{L}(h|\mathcal{D})$$

$$\overset{(2.16)}{=} \underset{h \in \mathcal{H}}{\operatorname{argmin}} (1/m) \sum_{i=1}^{m} L\big((\mathbf{x}^{(i)}, y^{(i)}), h\big).$$

Empirical Risk Minimization

$\widehat{L}(h|\mathcal{D})$

$\mathbb{E}\{L((\mathbf{x}, y), h)\}$

$\mathcal{H}$

hypothesis h
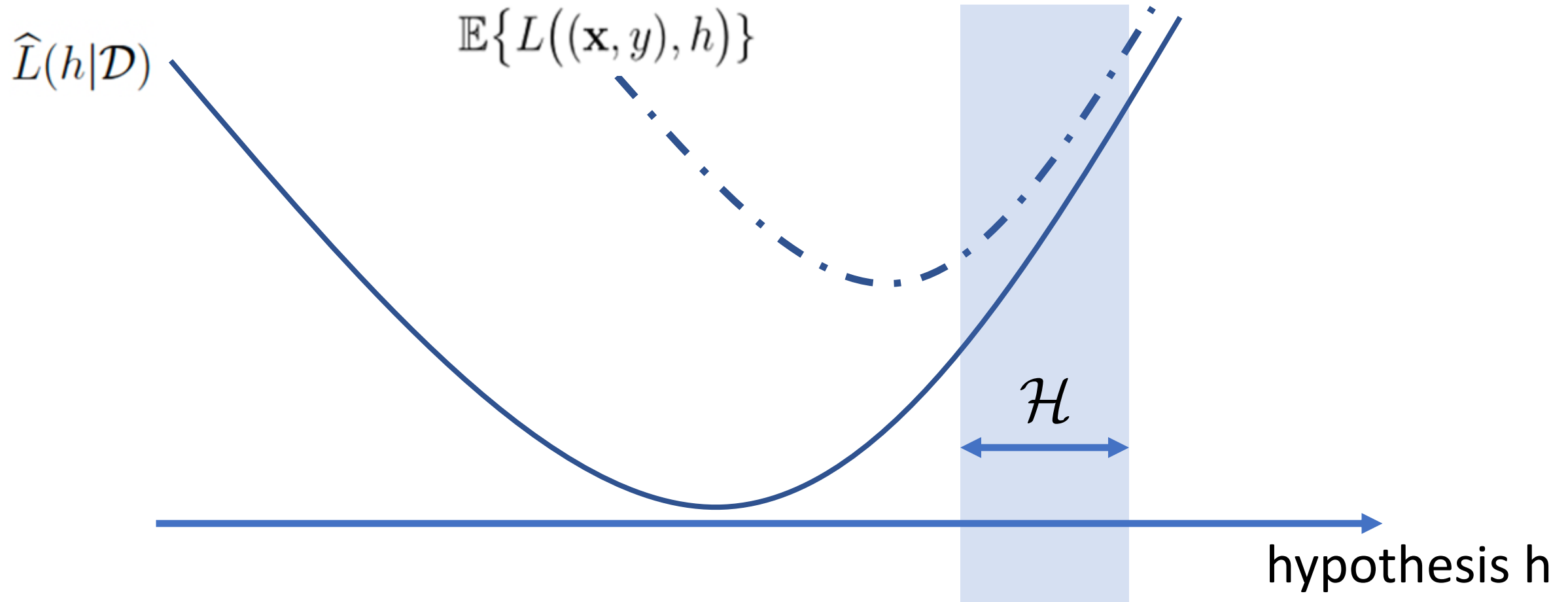
ERM for Parametrized Models

learnt (optimal) parameter vector

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} f(\mathbf{w})$$

loss incurred by h(.)
for i-th data point

$$\text{with } f(\mathbf{w}) := \underbrace{(1/m) \sum_{i=1}^{m} L\big((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}\big)}_{\widehat{L}\big(h^{(\mathbf{w})}|\mathcal{D}\big)}.$$

average loss or
empirical risk

ERM for Param. Models

$\widehat{L}(h|\mathcal{D})$

$\mathbb{E}\left\{ L\big((\mathbf{x}, y), h \big) \right\}$

$\mathcal{H}$

params w

Design Choices in ERM

loss

data

$\widehat{L}(h|\mathcal{D})$

$\mathbb{E}\{L((\mathbf{x}, y), h)\}$

model

$\mathcal{H}$

params w
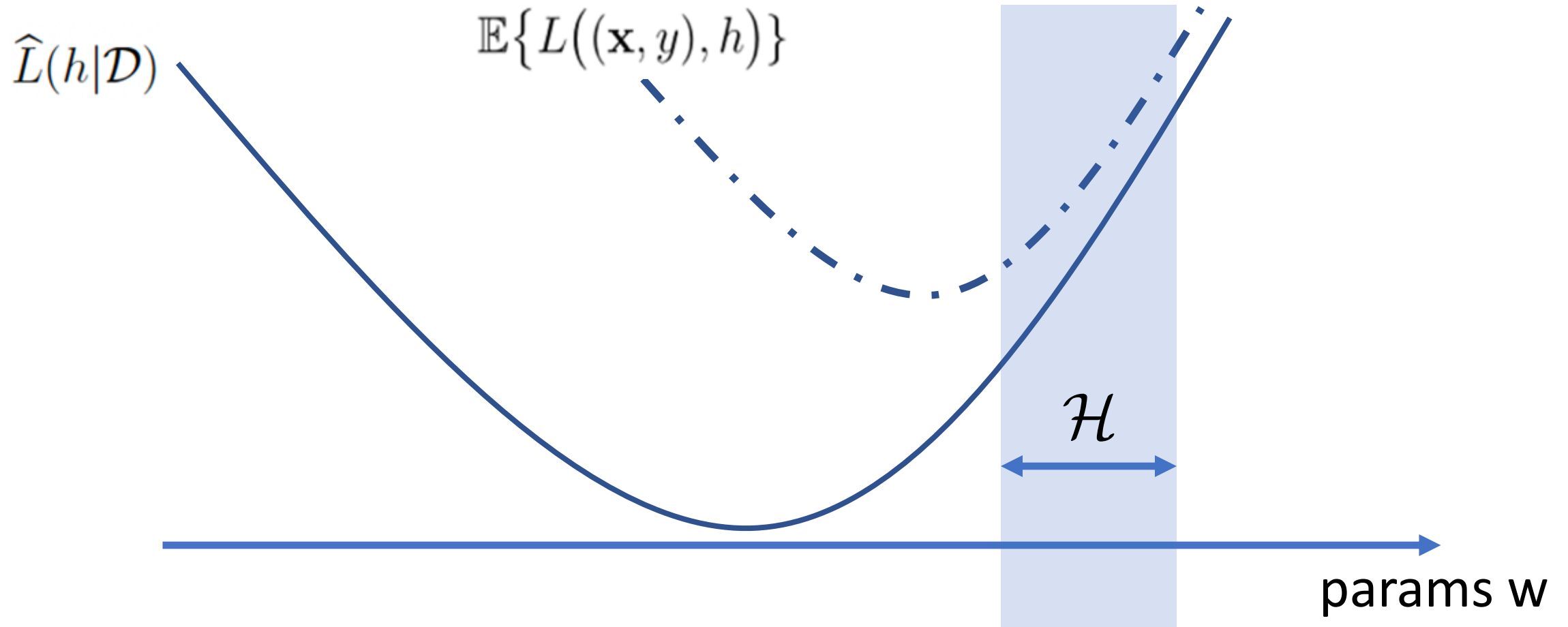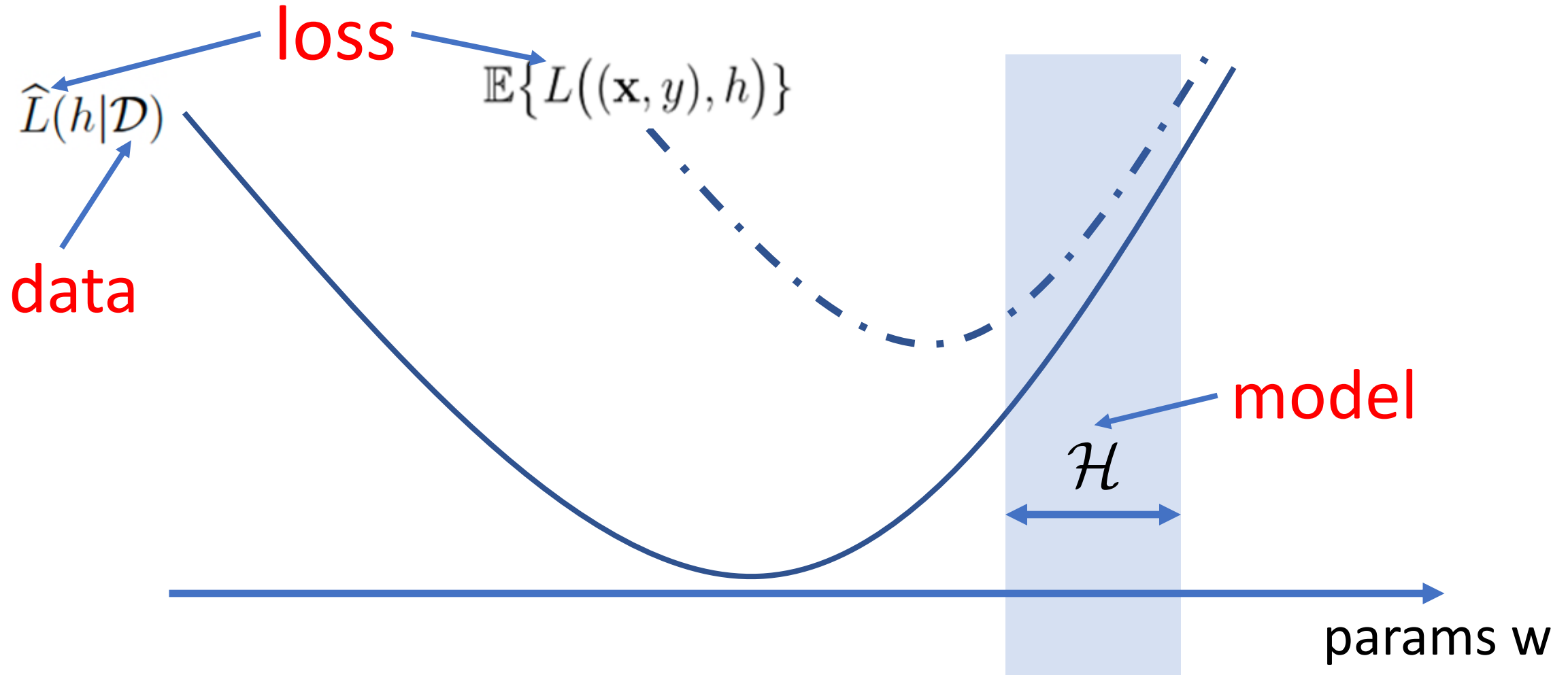
Design Choice: Model and Data

Linear Regression

- datapoints with numeric features and label

- model consists of linear maps

- squared error loss

sklearn.linear_model.LinearRegression

```
class sklearn.linear_model.LinearRegression(*, fit_intercept=True, normalize='deprecated', copy_X=True, n_jobs=None, positive=False)                                    [source]
```

Linear Regression

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

y

$\hat{y}^{(i)}$

$(y^{(i)} - \hat{y}^{(i)})^2$

$(\boldsymbol{x}^{(i)}, y^{(i)})$

features x
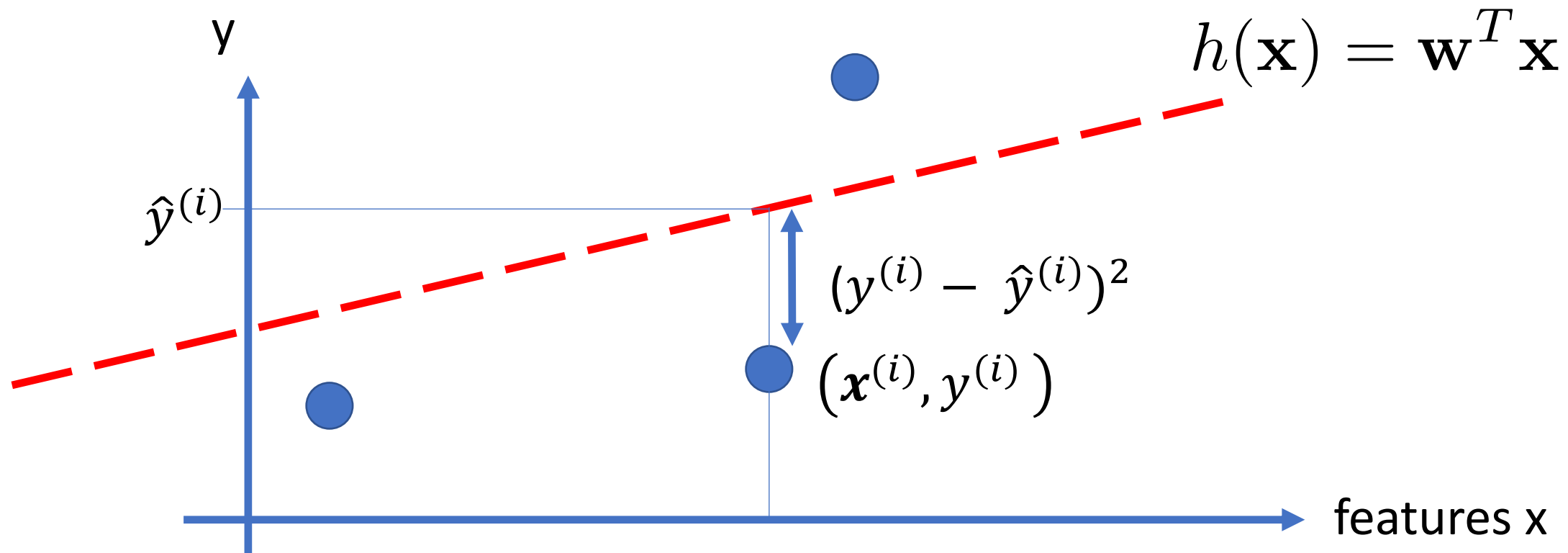
choose parameter/weight vector **w** to minimize average squared error loss

ERM for Linear Regression

$$\widehat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n}(1/m) \sum_{m=1}^{m} \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}\right)^2. \qquad (4.5)$$

A. Jung - HCML Summer School'22
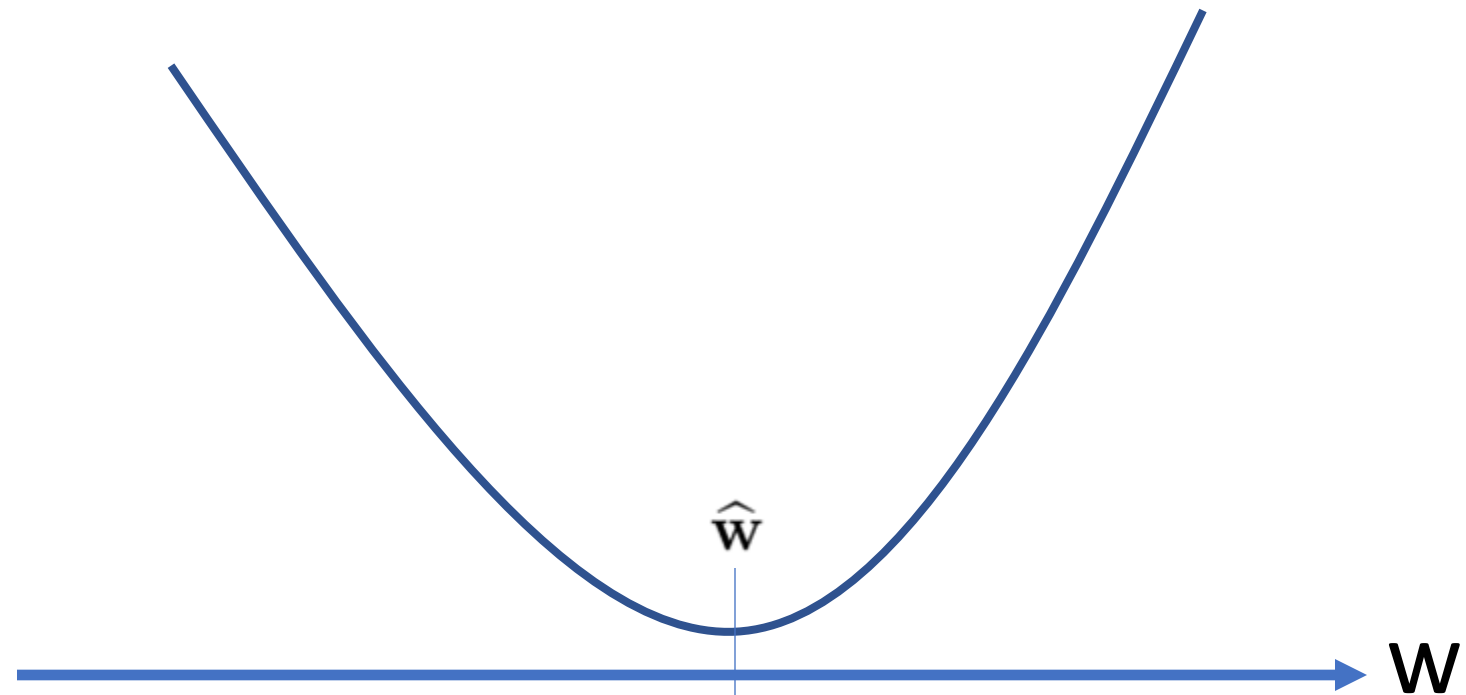
Linear Regression in Python

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} (1/m) \sum_{m=1}^{m} \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right)^2. \qquad (4.5)$$

```
In [81]:  # Create a linear regression model
          lr = LinearRegression()
          # Fit the model to our data in order to ge
          lr = lr.fit(features, labels)
```

$$\mathbf{X} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \right)^T \in \mathbb{R}^{m \times n} \qquad \mathbf{y} = \left(y^{(1)}, \ldots, y^{(m)} \right)^T \in \mathbb{R}^m$$

```
# create and train a linear model
lr = LinearRegression()
lr = lr.fit(X, y)
w_hat = lr.coef_
trainerr = mean_squared_error(lr.predict(X), y)
```
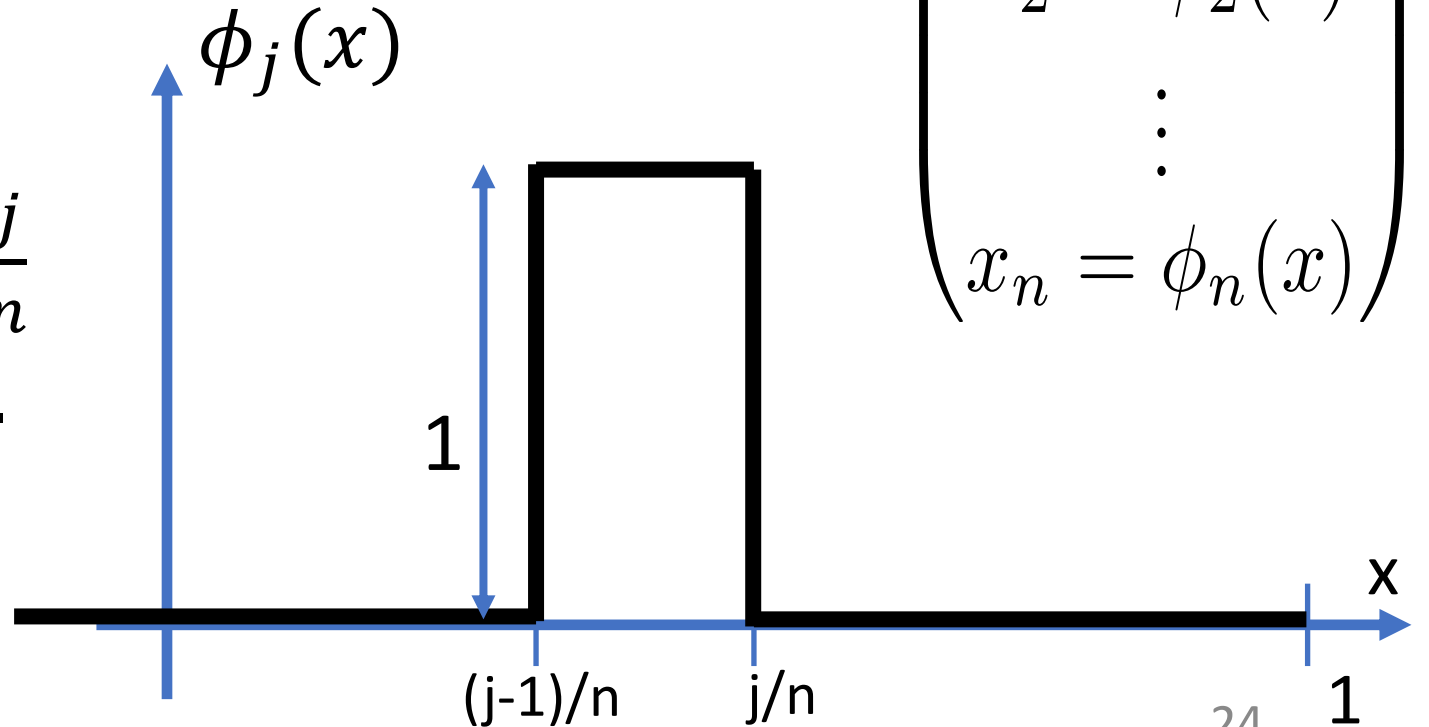
$$\widehat{L}(h|\mathcal{D})$$

$$\mathbb{E}\{L((\mathbf{x}, y), h)\}$$

$$\mathcal{H}$$

params w
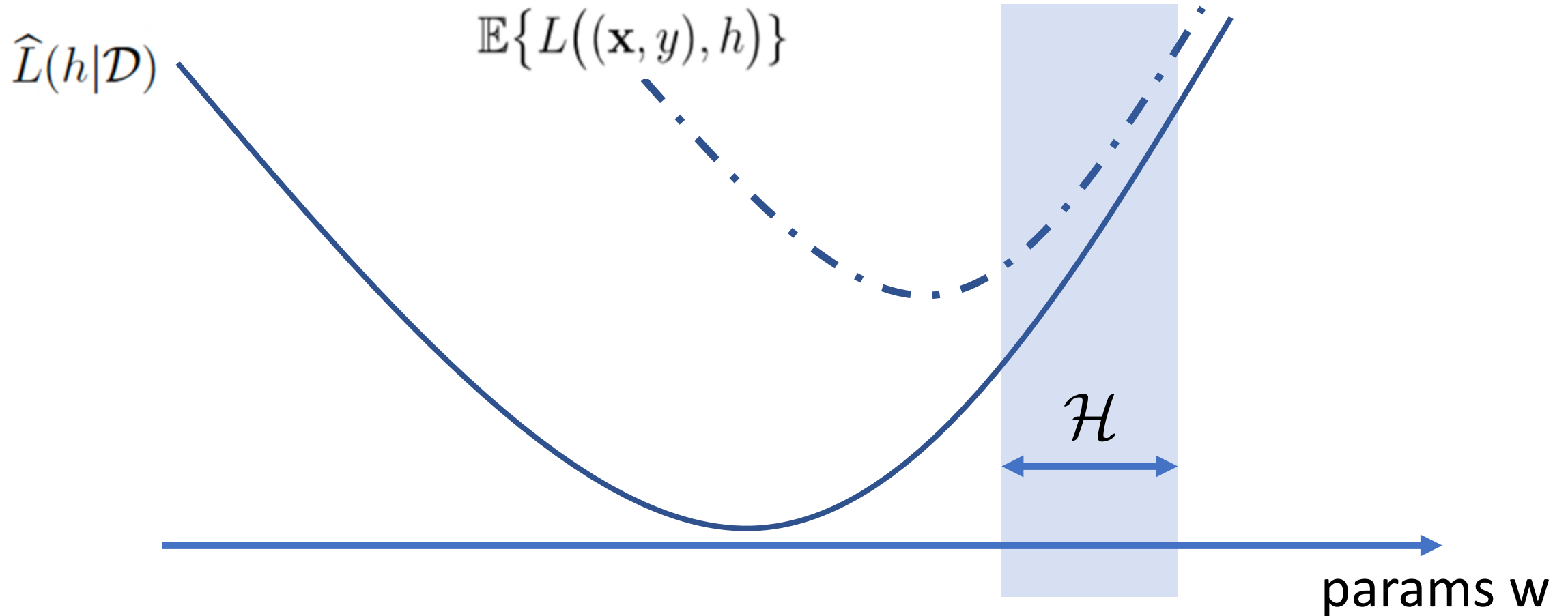
Upgrade Linear Model with new Features !

- consider data points with single numeric feature x

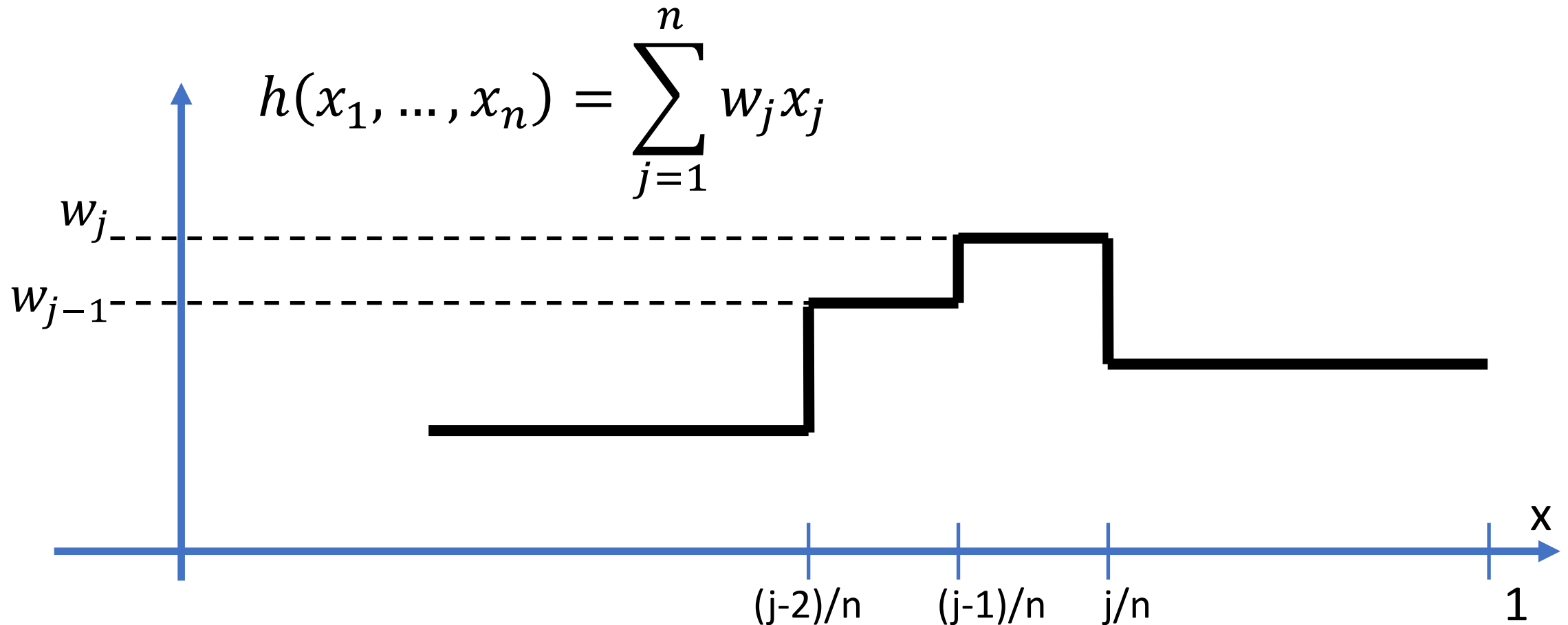- construct new features $x_1, \ldots, x_n$

$$\begin{pmatrix} x_1 = \phi_1(x) \\ x_2 = \phi_2(x) \\ \vdots \\ x_n = \phi_n(x) \end{pmatrix}$$

- $x_j = \begin{cases} 1 \ for \ \ \frac{j-1}{n} \leq x \leq \frac{j}{n} \\ 0 \ for \ all \ other \ x. \end{cases}$

$\phi_j(x)$

1

(j-1)/n   j/n

x

1

You Can Do Anything with Linear Predictors!

- h(x) is linear in new features but non-linear in raw feature x!
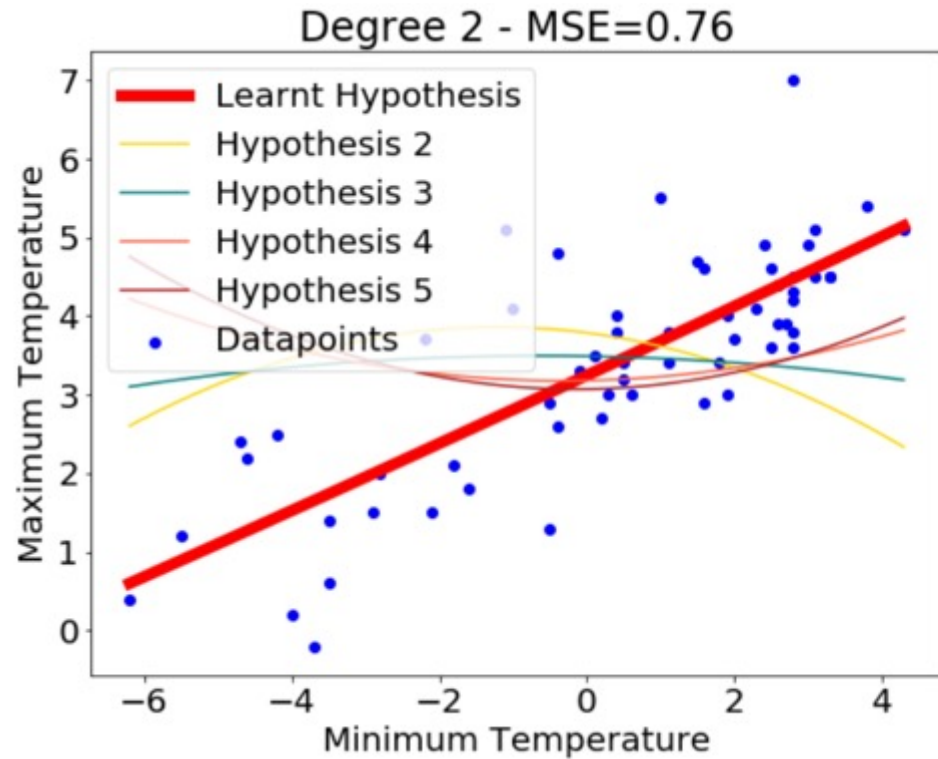
$$h(x_1, \ldots, x_n) = \sum_{j=1}^{n} w_j x_j$$

Polynomial Regression

$$\mathcal{H}_{\text{poly}}^{(n)} = \{h^{(\mathbf{w})} : \mathbb{R} \to \mathbb{R} : h^{(\mathbf{w})}(x) = \sum_{j=1}^{n} w_j x^{j-1},$$

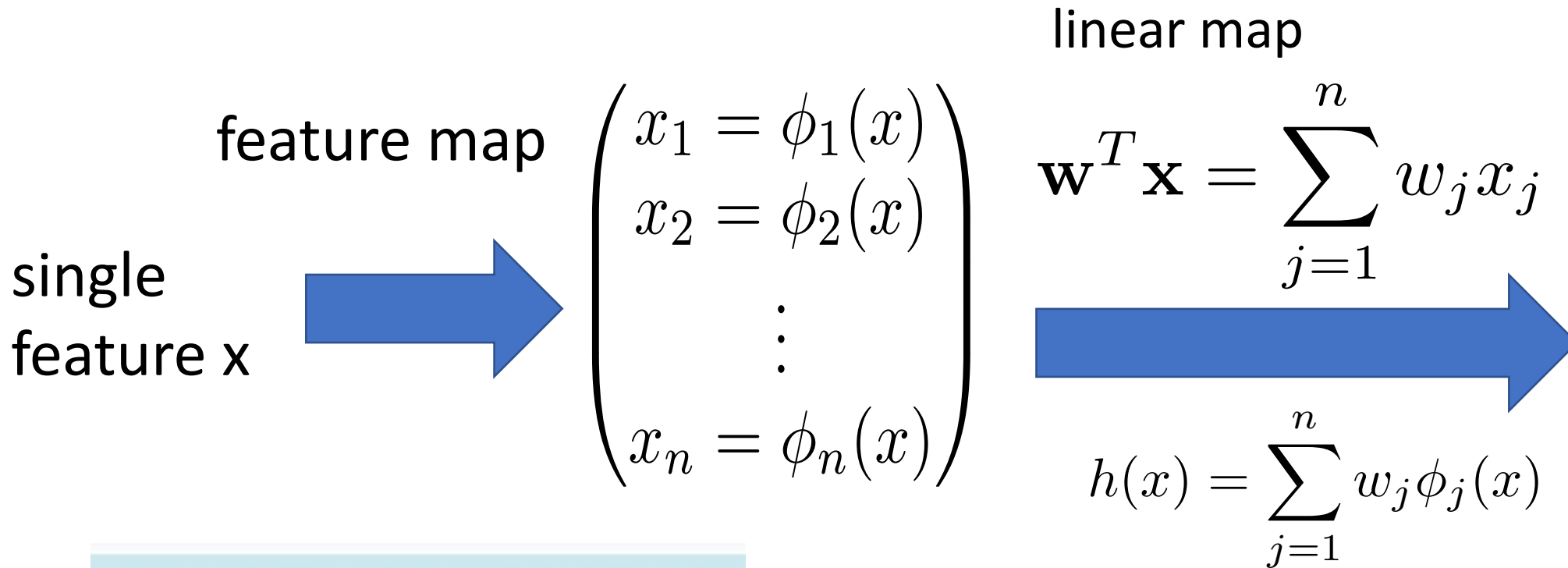$$\text{with some } \mathbf{w} = (w_1, \ldots, w_n)^T \in \mathbb{R}^n\}. \tag{3.4}$$

Polynomial Regression
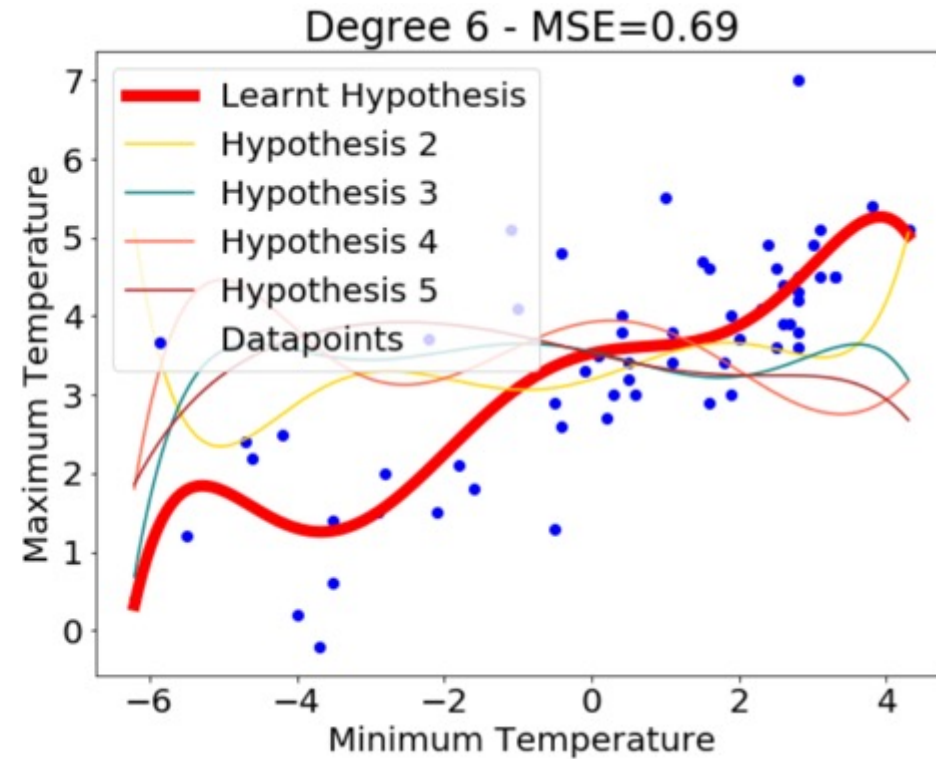
from notebook
https://github.com/alexjungaalto/cs-c3240spring2022/blob/main/George_Demo_PolynomialRegression.ipynb

Polynomial Regression=
Lin. Reg. with Feature Transform.

feature map

single
feature x

linear map

$$\begin{pmatrix} x_1 = \phi_1(x) \\ x_2 = \phi_2(x) \\ \vdots \\ x_n = \phi_n(x) \end{pmatrix}$$

$$\mathbf{w}^T\mathbf{x} = \sum_{j=1}^{n} w_j x_j$$

$$h(x) = \sum_{j=1}^{n} w_j \phi_j(x)$$

sklearn.preprocessing.PolynomialFeatures

.preprocessing.PolynomialFeatures(degree=2, *, interaction_only=False, include_bias=True

sklearn.linear_model.LinearRegression

class sklearn.linear_model.LinearRegression(*, fit_intercept=True, normalize='deprecated', copy_X=True, n_
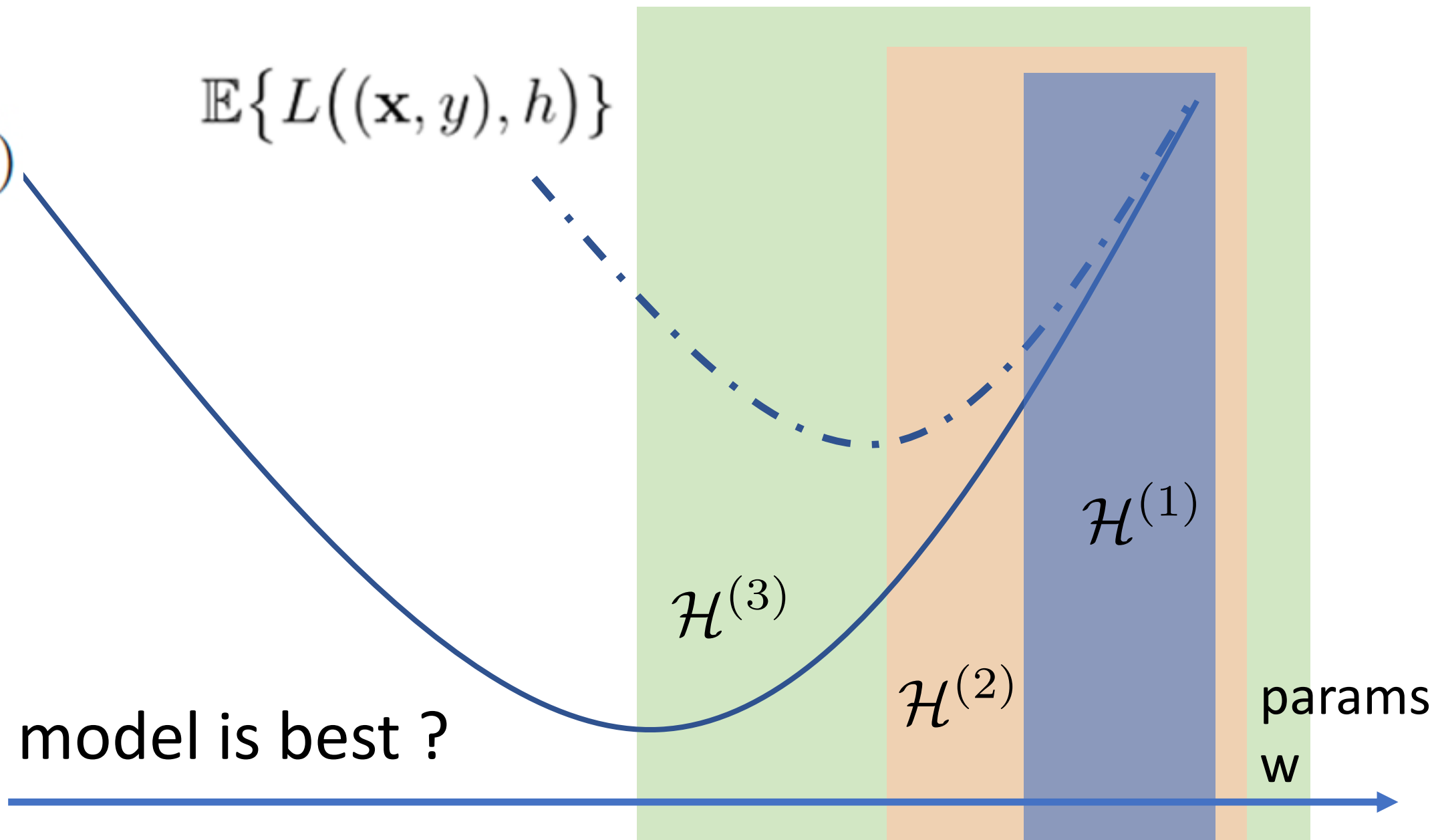positive=False)

Polynomial Features

we can use anything as features that can be computed or measured easily !

	Date	Max temp	Min temp		(Min temp)^2
0	2020-2-1	3.0	1.9		3.61
1	2020-2-2	4.9	2.4		5.76
2	2020-2-3	2.6	-0.4		0.16
3	2020-2-4	-0.2	-3.7		13.69
4	2020-2-5	2.5	-4.2		17.64

$\widehat{L}(h|\mathcal{D})$

$\mathbb{E}\{L((\mathbf{x}, y), h)\}$

$\mathcal{H}^{(1)}$

$\mathcal{H}^{(3)}$

$\mathcal{H}^{(2)}$

params w

which model is best ?

Design Choice: Loss Function

Measuring Error Size via Loss Functions

loss L

loss function is design choice !

prediction error $\hat{y} - y$

The Squared Error Loss

$$L := (\hat{y} - y)^2$$

prediction error $\hat{y} - y$
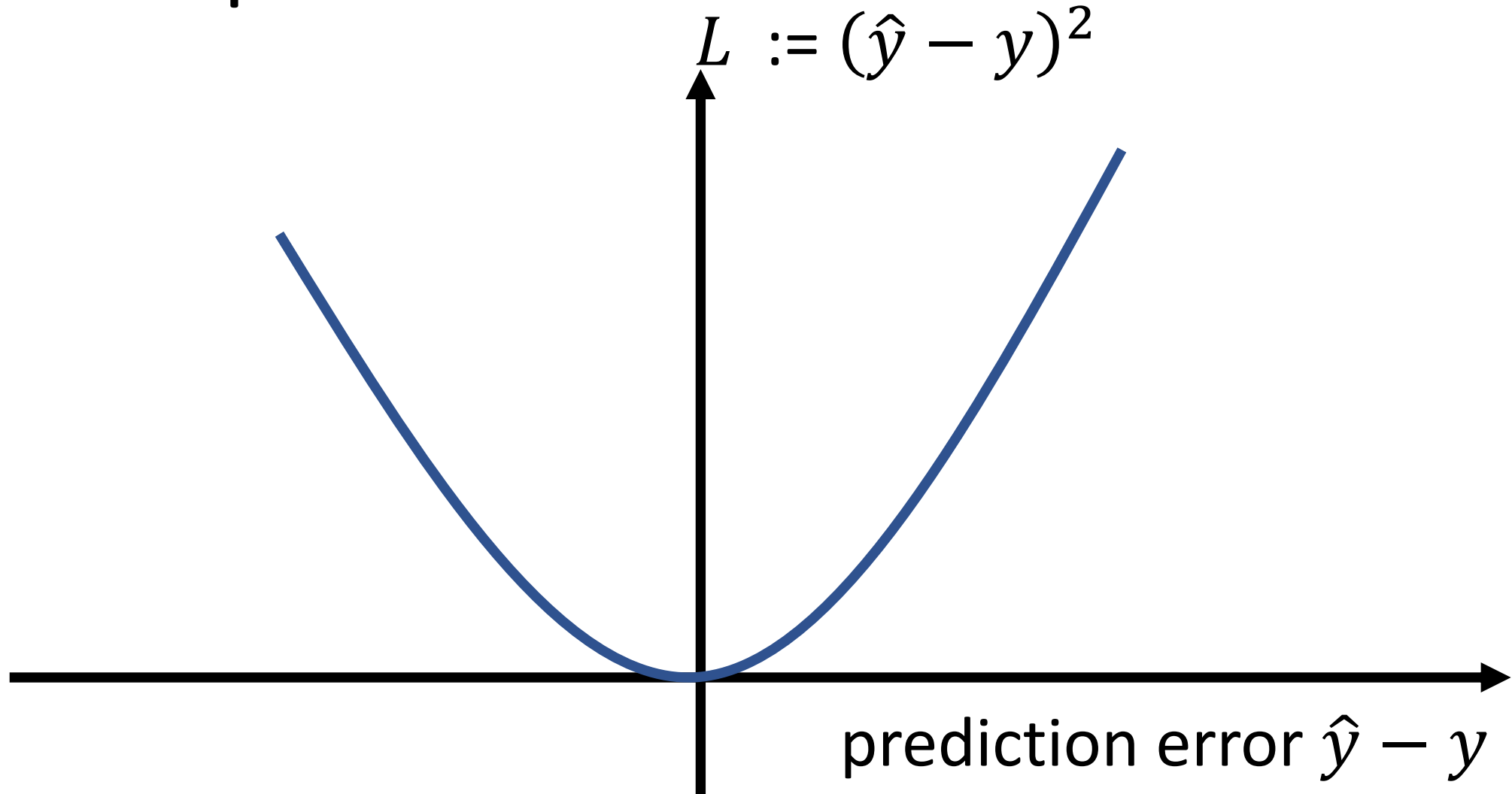
Squared Error Loss Sensitive to Outliers

$h(x) = w*x+b$

$\hat{y}^{(i)}$

$y^{(i)} - \hat{y}^{(i)}$

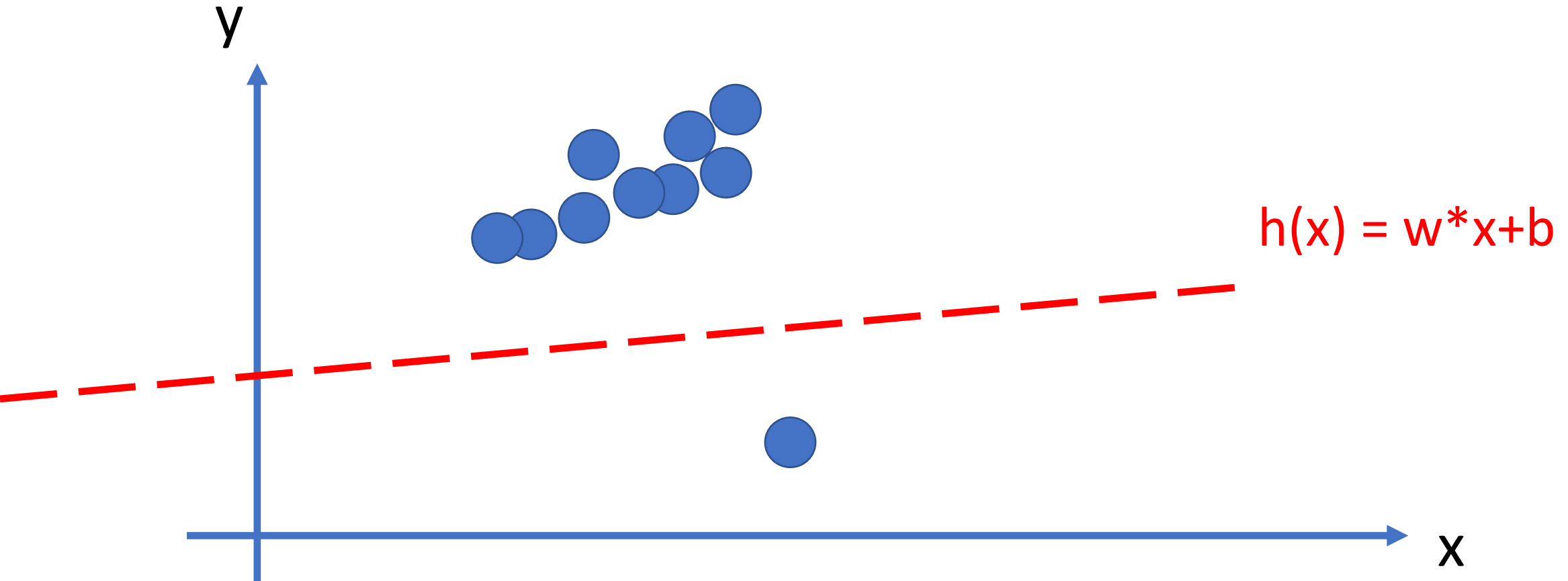"outlier" $\left(x^{(i)}, y^{(i)}\right)$
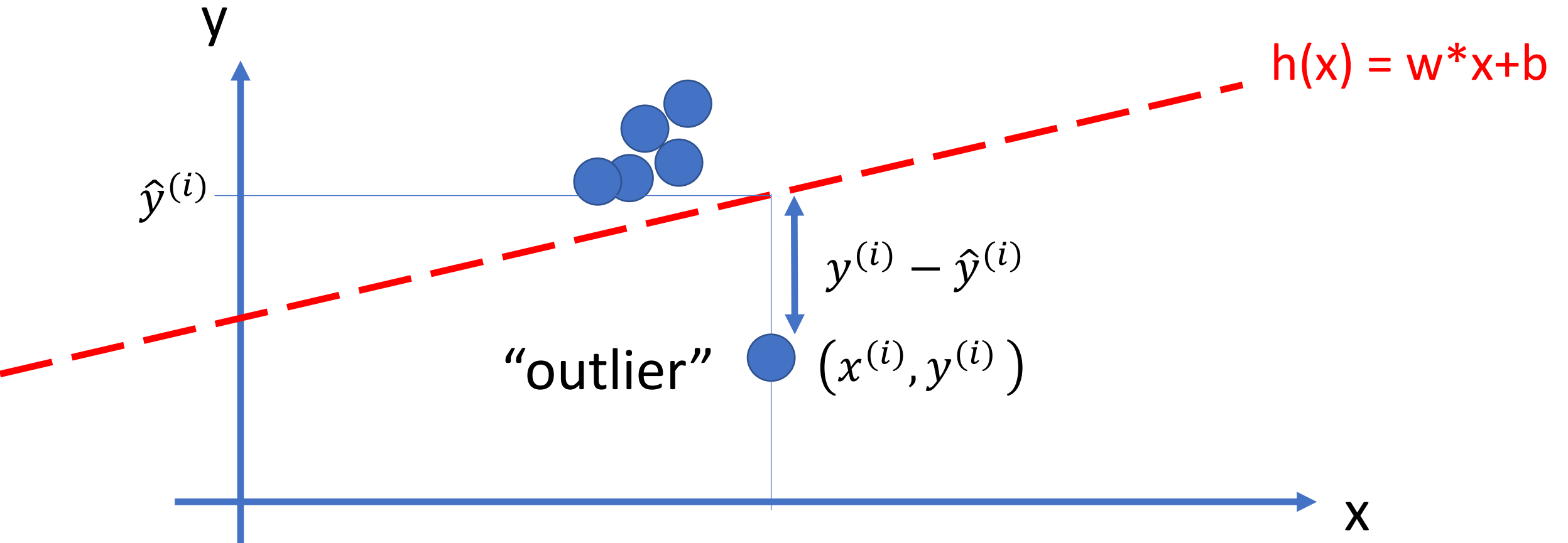
min. squared error loss forces predictor towards outlier

Train Linear Model on "Clean Data"
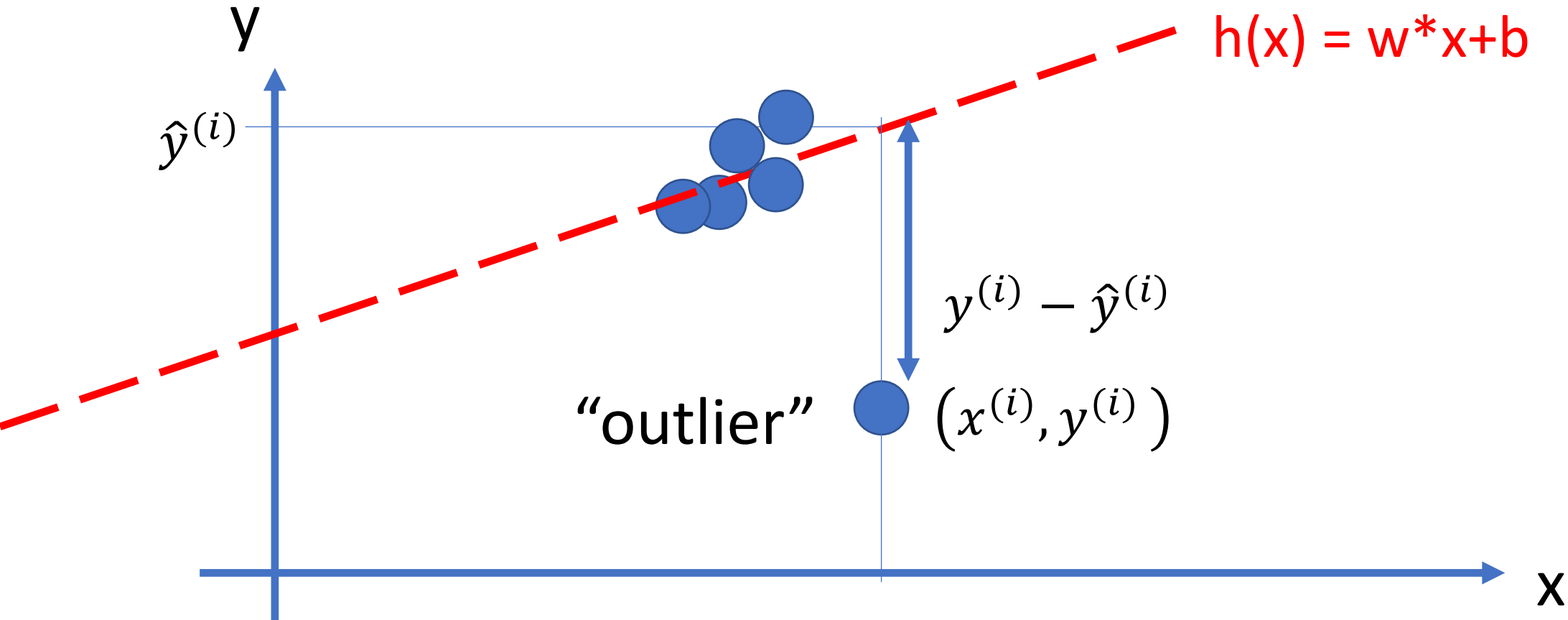
$h(x) = w*x+b$
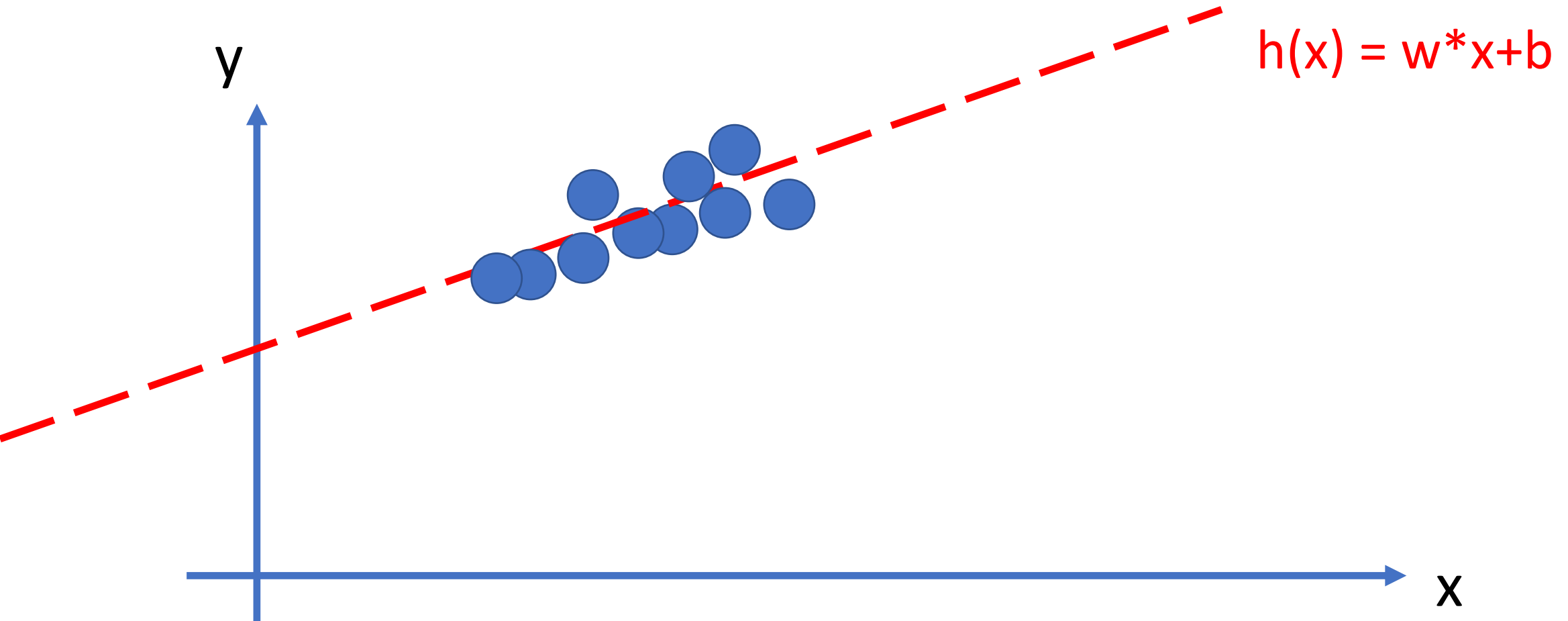
Training Set with a SINGLE OUTLIER !

$h(x) = w*x+b$

How to make learning robust against presence of few outliers in training set ?

A. Jung - HCML Summer School'22

The Absolute Error Loss

$$L := |\hat{y} - y|$$

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

prediction error $\hat{y} - y$
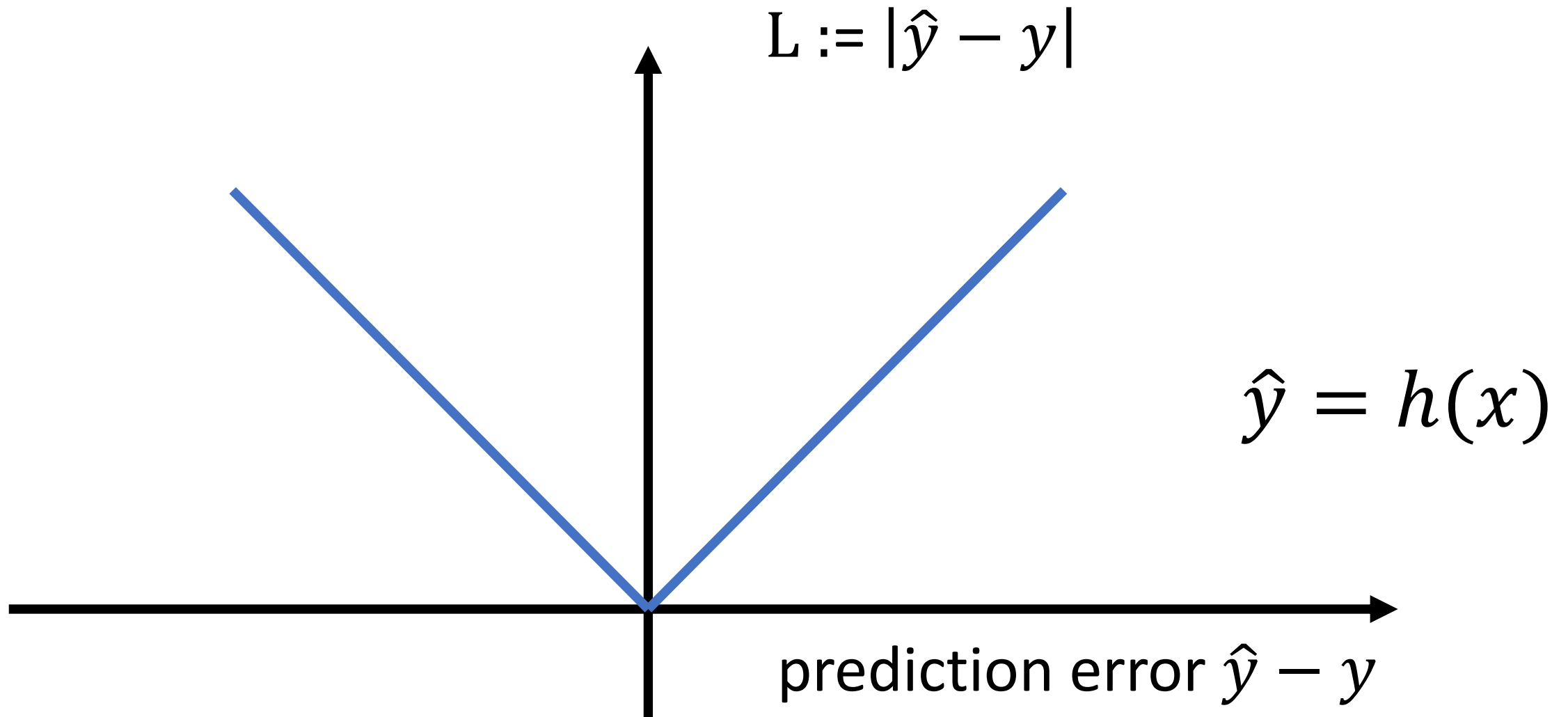
Absolute Error Loss Robust to Outliers

y

$h(x) = w*x+b$

$\hat{y}^{(i)}$

$y^{(i)} - \hat{y}^{(i)}$

"outlier" $\left(x^{(i)}, y^{(i)}\right)$

x

absolute error "tolerates" few outliers

Huber Loss

absolute error loss

prediction error $\hat{y} - y$
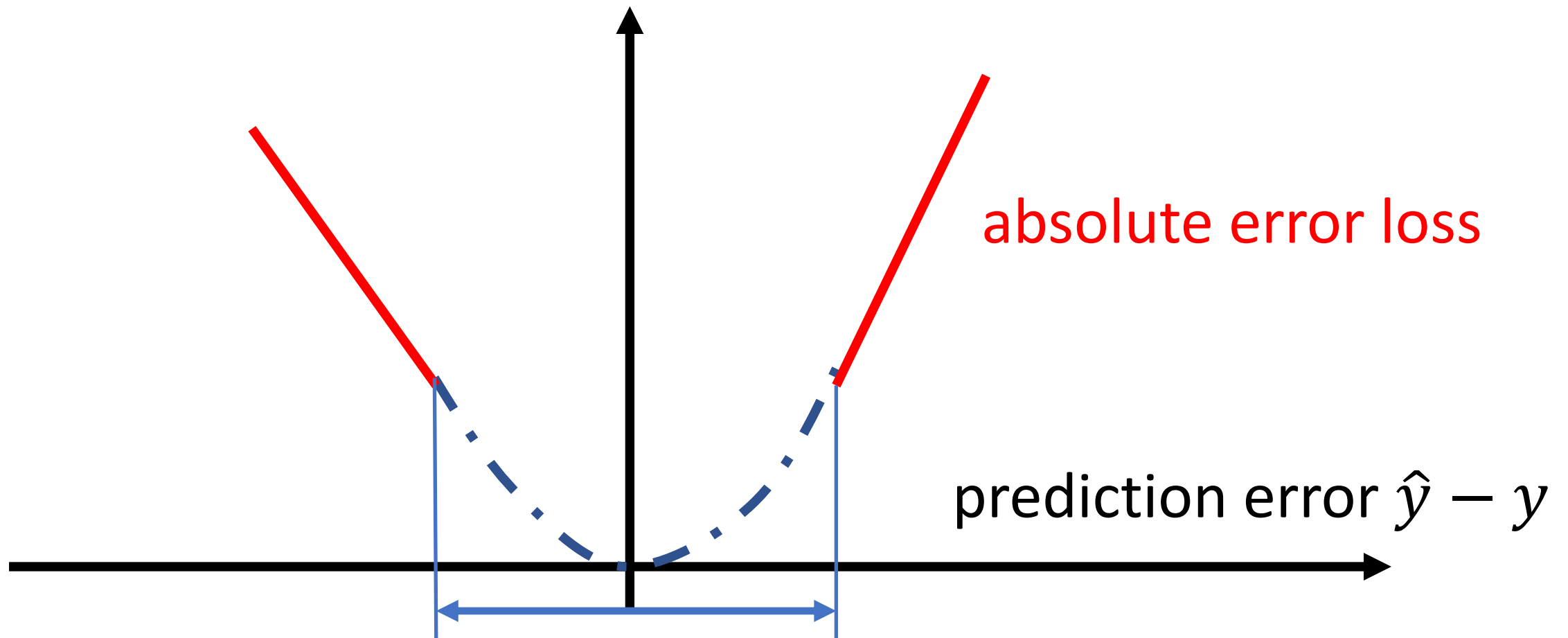
squared error loss

Fitting Linear Predictor with Huber Loss

$h(x) = w*x+b$

$\hat{y}^{(i)}$

squared error loss

$y^{(i)} - \hat{y}^{(i)}$

"outlier" $(x^{(i)}, y^{(i)})$

y

x

```
sklearn.linear_model.HuberRe
.near_model. HuberRegressor(epsilon=1.35, max_iter=100, alpha=0.0
tol=1e-05)
```

Train Linear Model on "Clean Data"

$h(x) = w*x+b$

Training Set with a SINGLE OUTLIER !

y

$h(x) = w*x+b$

x
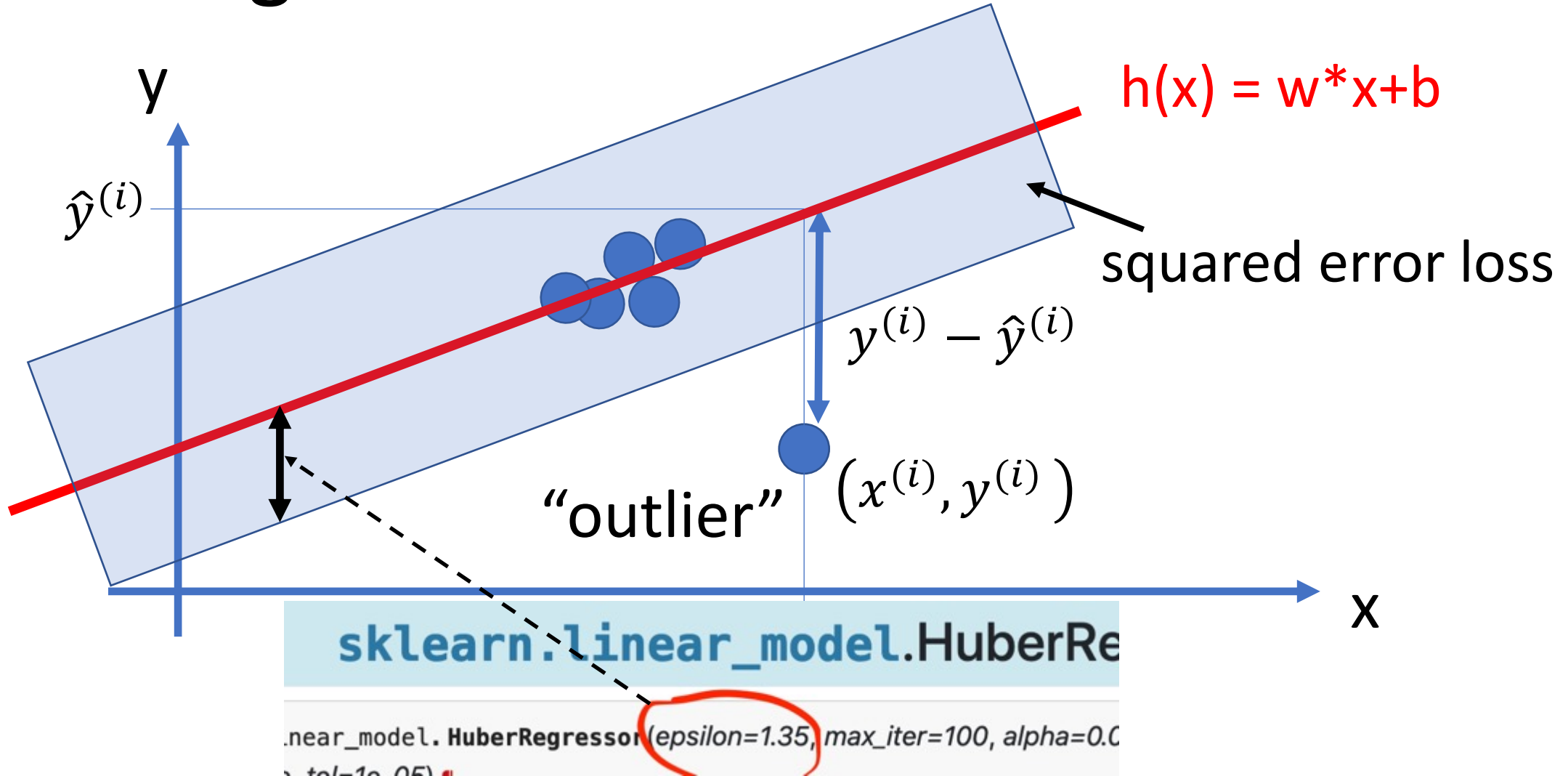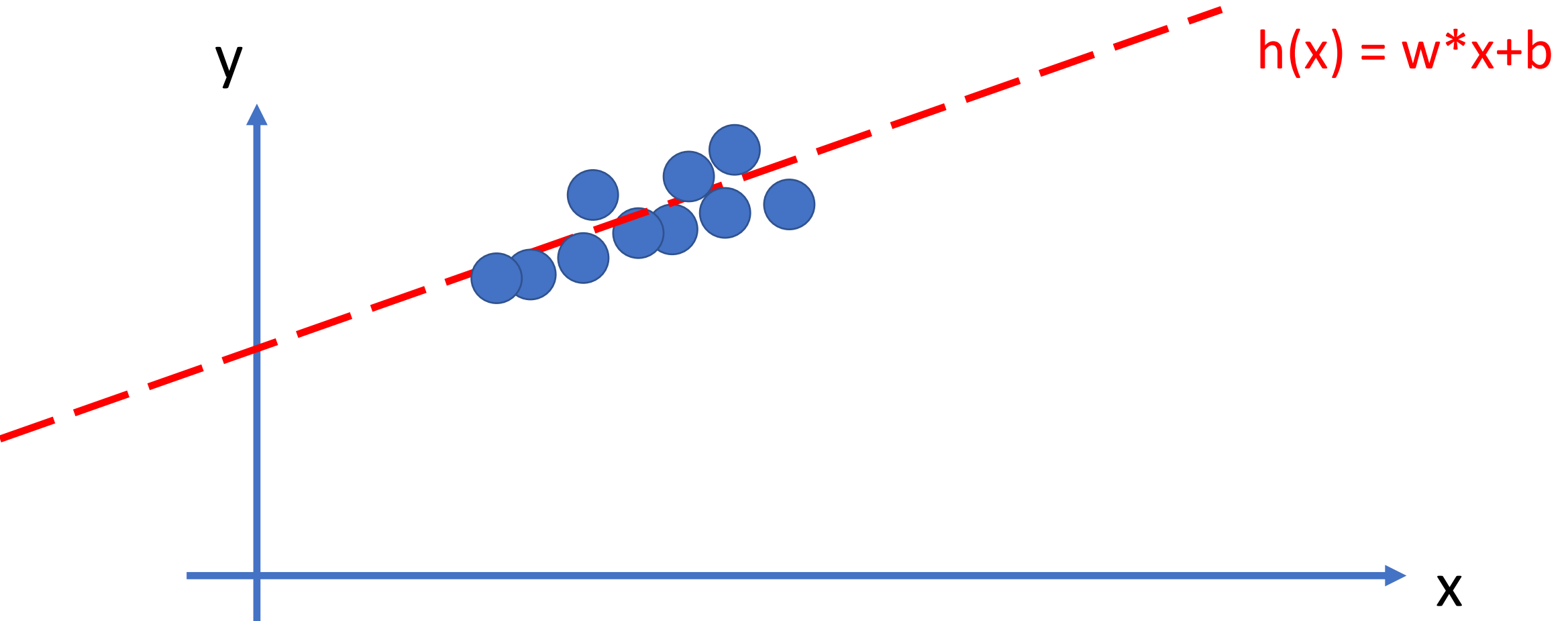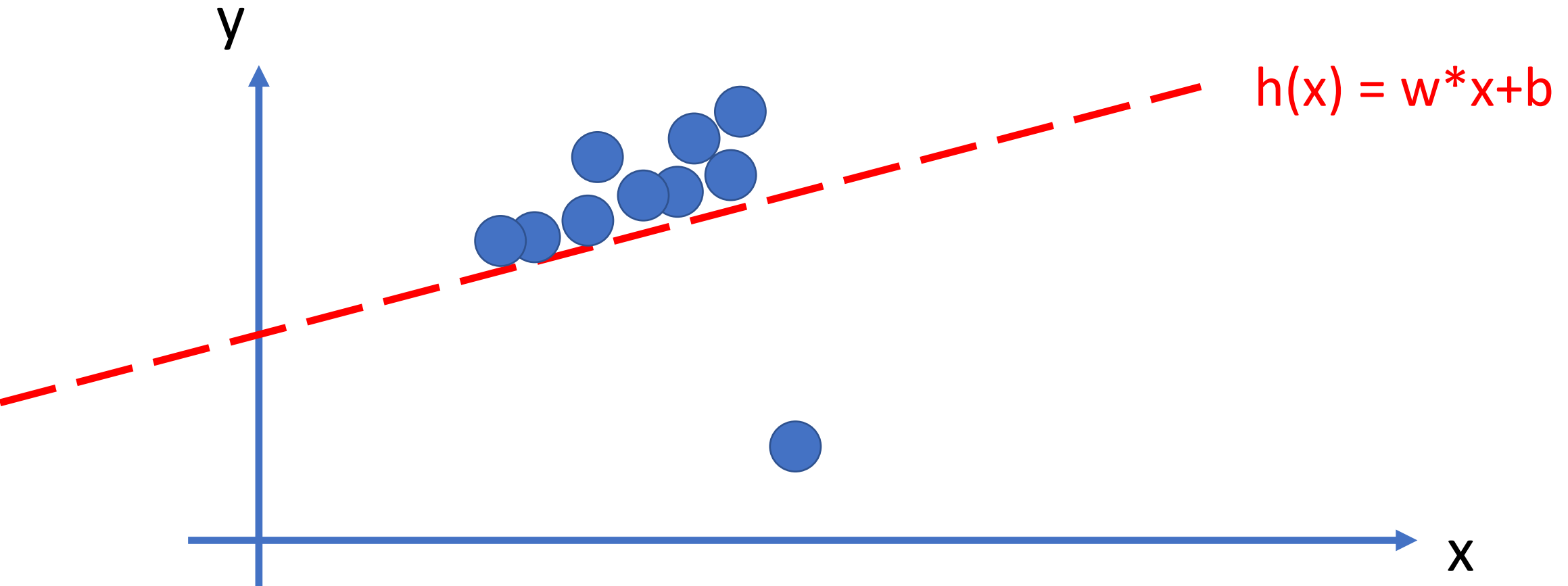
Huber vs. Squared Error Loss

Squared Error

- cvx and diff.able
- minimized via simple gradient descent
- sensitive to outliers

Huber

- cvx and non-diff.
- requires more advanced opt. methods
- robust against outliers

Summary

- ultimate quality measure: expected loss or risk ("iid" asspt!)

- approximate risk by average loss (empirical risk)

- regression methods are instances of ERM

- three design choices of ERM: data, model and loss

- ERM can fail if empirical risk deviates from risk (e.g., due to outliers)

What's Next ?

- tmrw, lecture "Classification"

- like regression, class. methods are instances of ERM

- class. use different label values and loss functions