Machine Learning for Networking
ML4N

Luca Vassio
Gabriele Ciravegna
Zhihao Wang
Tailai Song

The three components of ML

- Data
- Model
- Loss

Data

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

Data points characterized by features and label

– Features low-level properties

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

– Labels high-level properties (quantity of interest)

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

Model and loss

- GOAL of ML = learn to predict the label y of a data point from its features **x**
- ML model = learn a hypothesis $h \in \mathcal{H}$ $h : \mathcal{X} \to \mathcal{Y}$

  such that h(**x**) ≈ y
- Loss function: how to quantify/weight prediction error between y and h(**x**)

Empirical risk minimization

Learning goals

- Know about notion of **expected loss or risk**
- Know that **average loss approximates risk**
- Know about **empirical risk minimization**
- Know some **design choices in ERM**

Learn a hypothesis $h \in \mathcal{H}$ $h : \mathcal{X} \rightarrow \mathcal{Y}$

such that h(**x**) ≈ y for any data point (**x**,y)

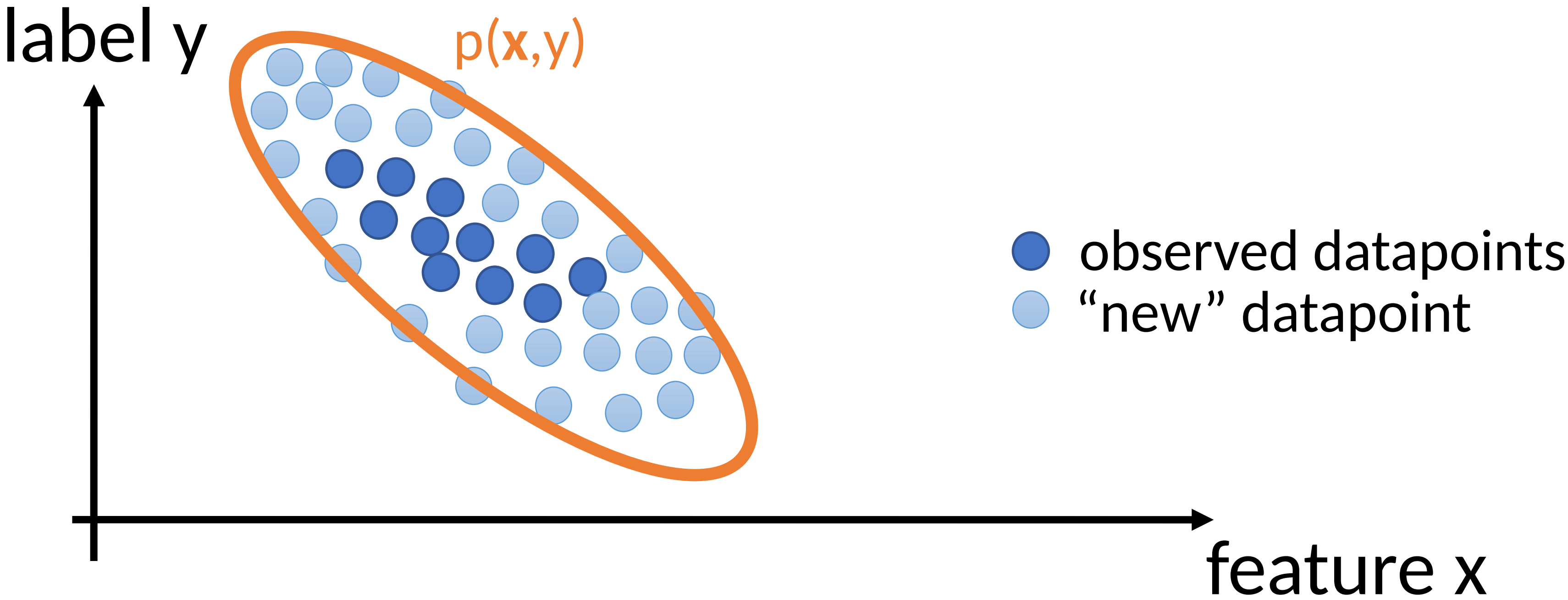
What exactly is "any data point"?

Data. Model. Loss.

- Data: set of data points (**x**,y)
- Model: set $\mathcal{H}$ of hypothesis maps h(.)
- Loss: quality measure L((**x**,y),h)

What is any data point?

- Interpret data points as realizations of i.i.d. random variables with probability distribution p(**x**,y)
- Define loss incurred for any data point as the **expected loss**, i.e., on average what will be the loss on the points of the distribution



Expected loss or risk

- Interpret data points as realizations of i.i.d. random variables with probability distribution p(**x**,y)
- Define loss incurred for any data point as the **expected loss**, i.e., on average what will be the loss on the points of the distribution
- Also called **expected risk** or **Bayes risk**

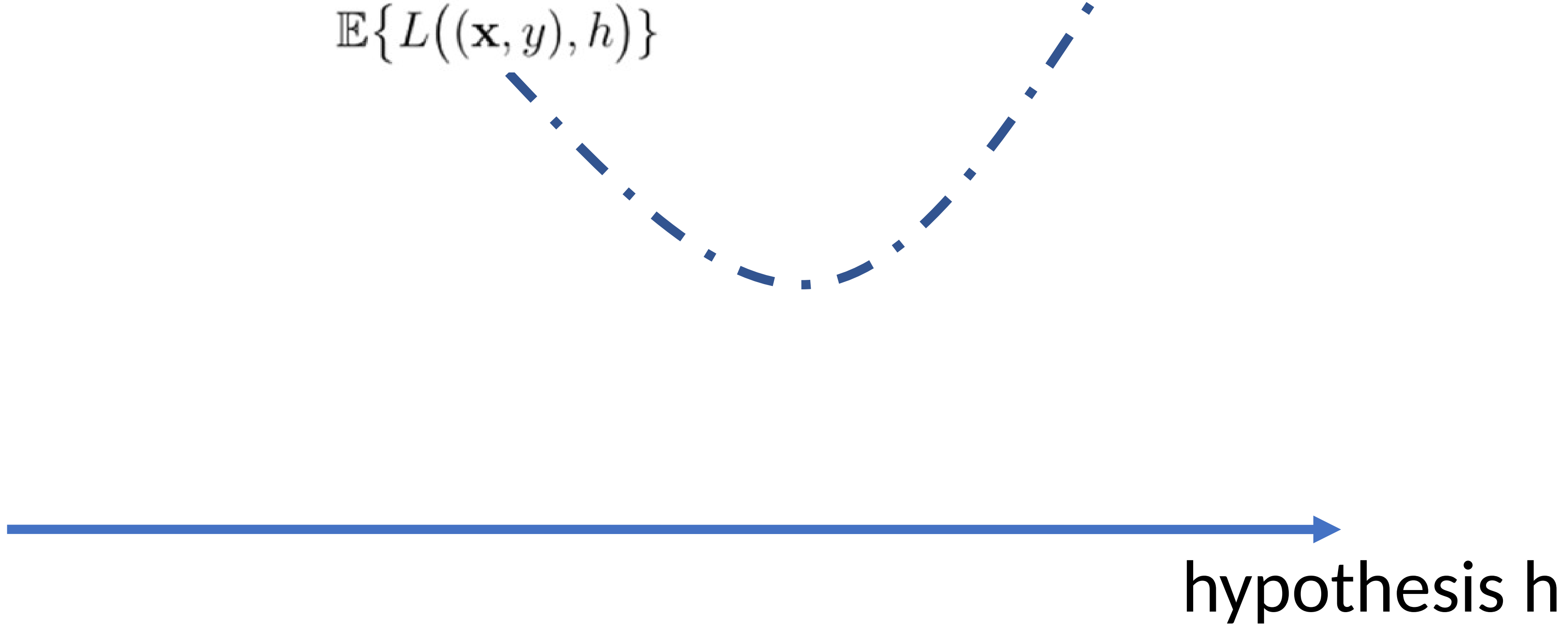
$$\mathbb{E}\{L((\mathbf{x},y),h)\} := \int_{\mathbf{x},y} L((\mathbf{x},y),h)\,dp(\mathbf{x},y).$$

To compute this expectation we need to know the probability distribution p(**x**,y) of data points (**x**,y)

Expected loss or risk

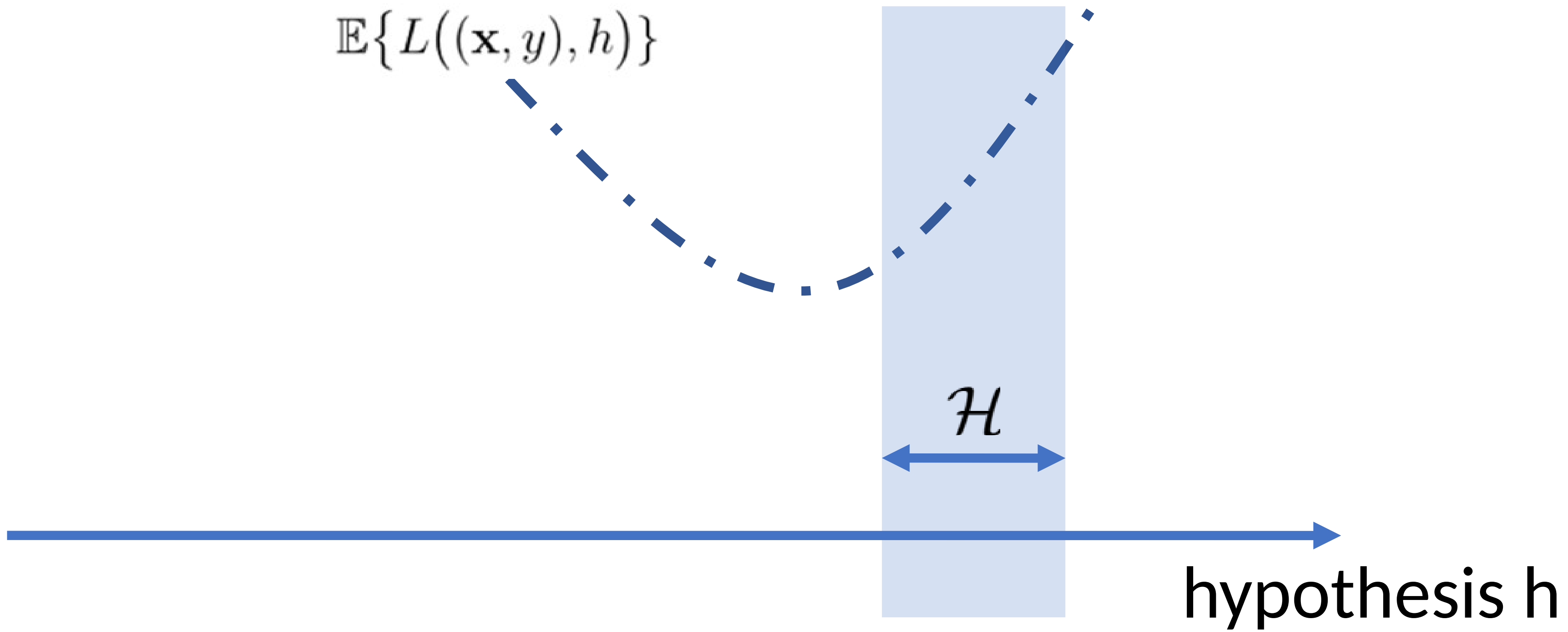
Expected loss to be minimized

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



Expected loss or risk

Expected loss to be minimized



Empirical risk

IDEA: approximate expected loss by average loss on data points (training set) = empirical risk

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

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

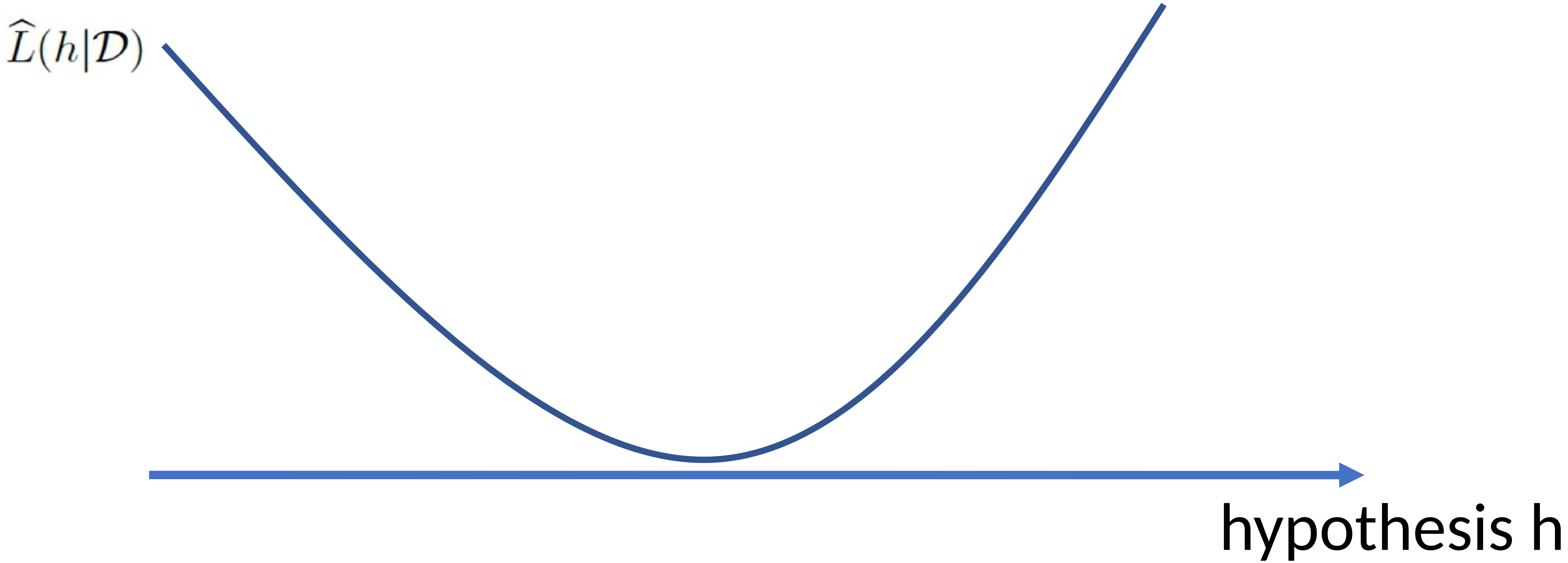
$$\mathbb{E}\{L((\mathbf{x}, y), h)\} \approx \widehat{L}(h|\mathcal{D}) \quad \text{for sufficiently large sample size } m.$$

Empirical Risk Minimization (ERM)

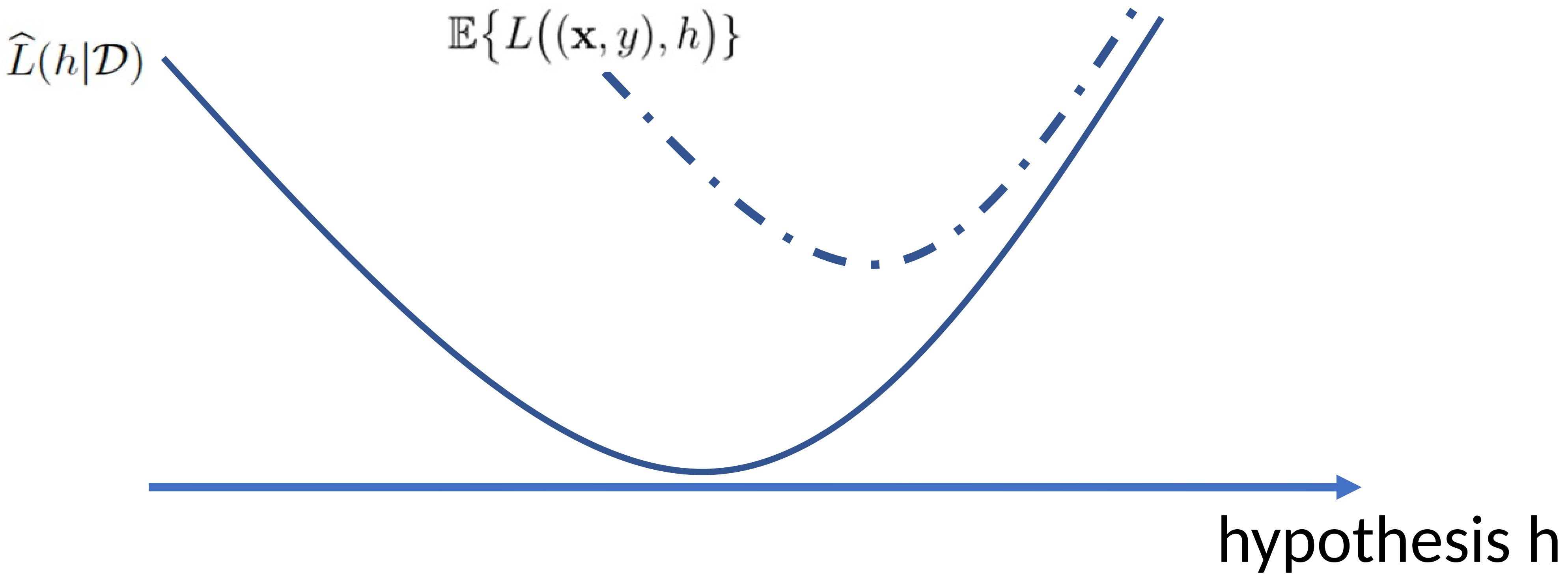
$$\hat{h} \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \widehat{L}(h|\mathcal{D}) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} (1/m) \sum_{i=1}^{m} L\big((\mathbf{x}^{(i)}, y^{(i)}), h\big).$$

- Any data points = training data points
- Learn **hypothesis** out of a hypothesis space or model that incurs minimum average **loss** when predicting **labels** of **training** datapoints based on their **features**

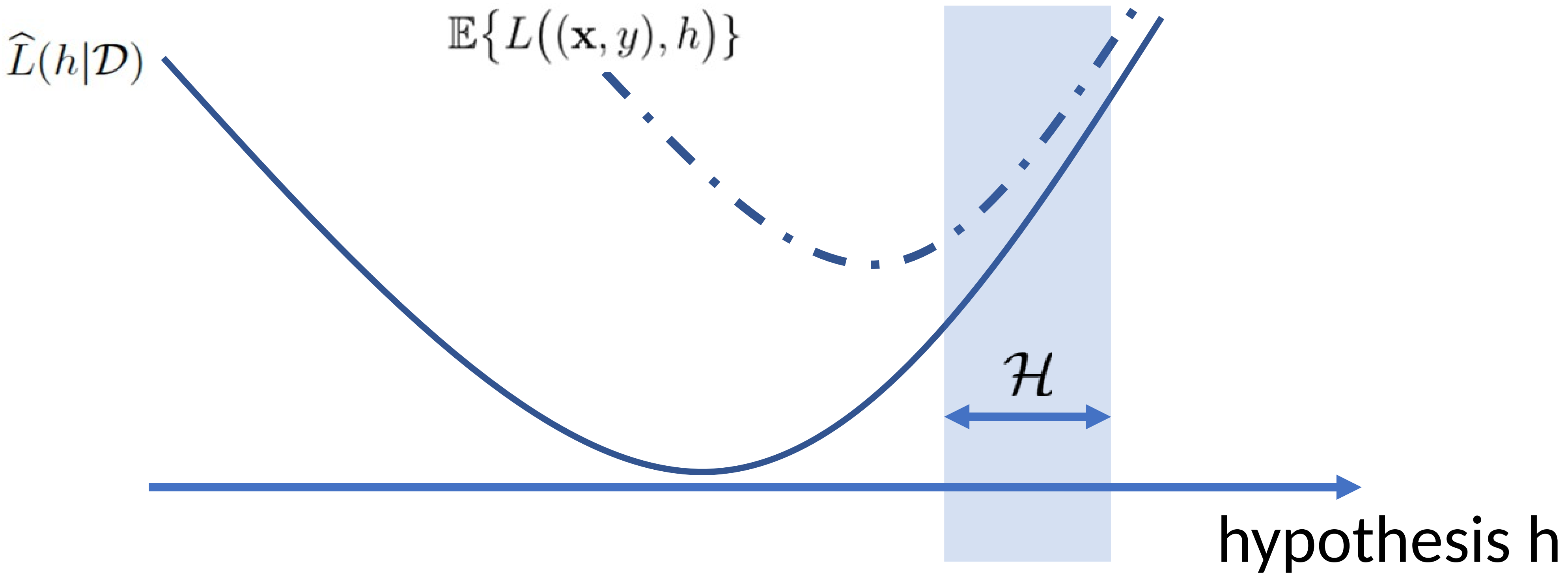
Empirical Risk Minimization



Empirical Risk Minimization



Empirical Risk Minimization



ERM for parametrized models

learnt (optimal) parameter vector

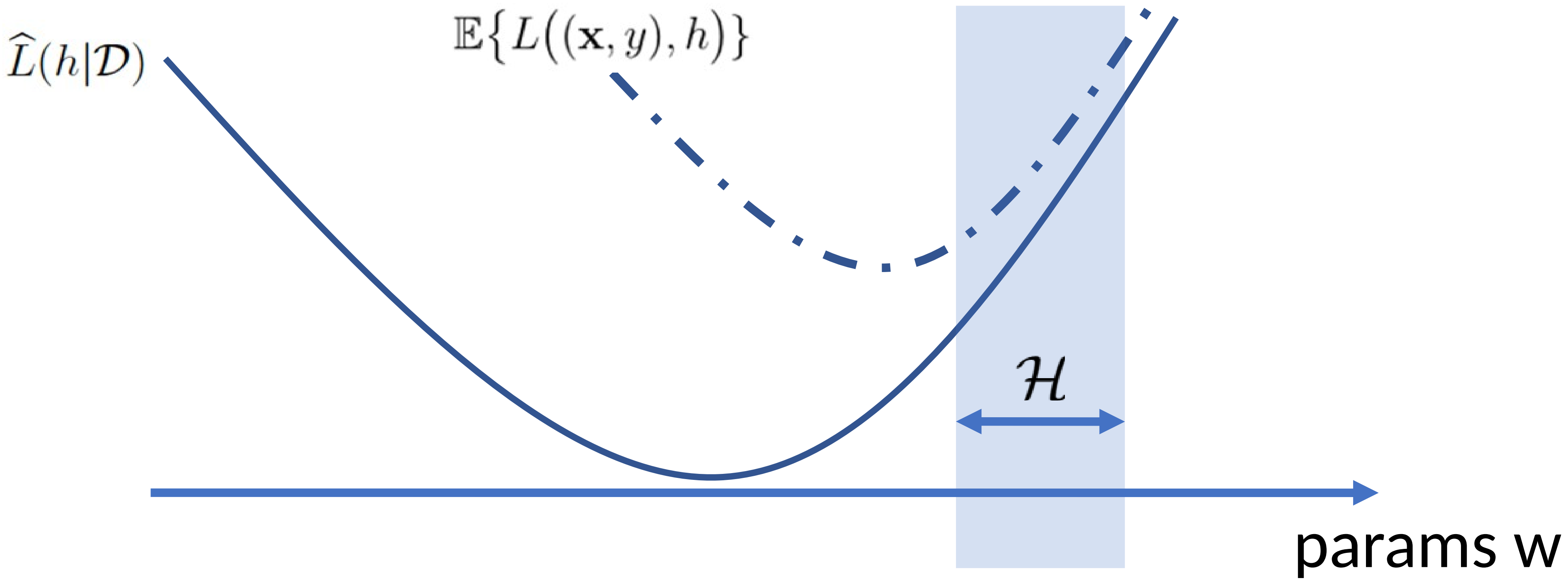
$$\widehat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w})$$

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

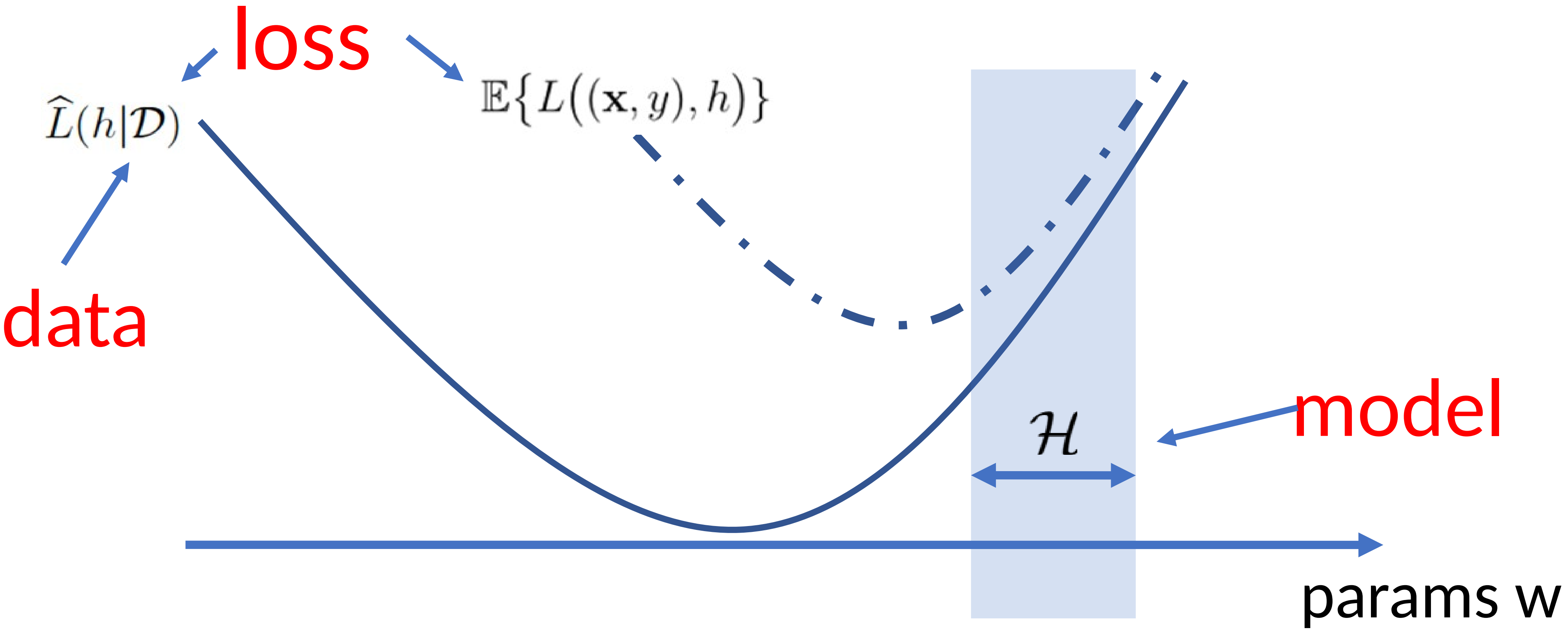
$$\text{with } f(\mathbf{w}) := \underbrace{(1/m) \sum_{i=1}^{m} L\Big((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}\Big)}_{\widehat{L}\left(h^{(\mathbf{w})} | \mathcal{D}\right)} .$$

average loss or
empirical risk

ERM for parametrized models



Design choices in ERM



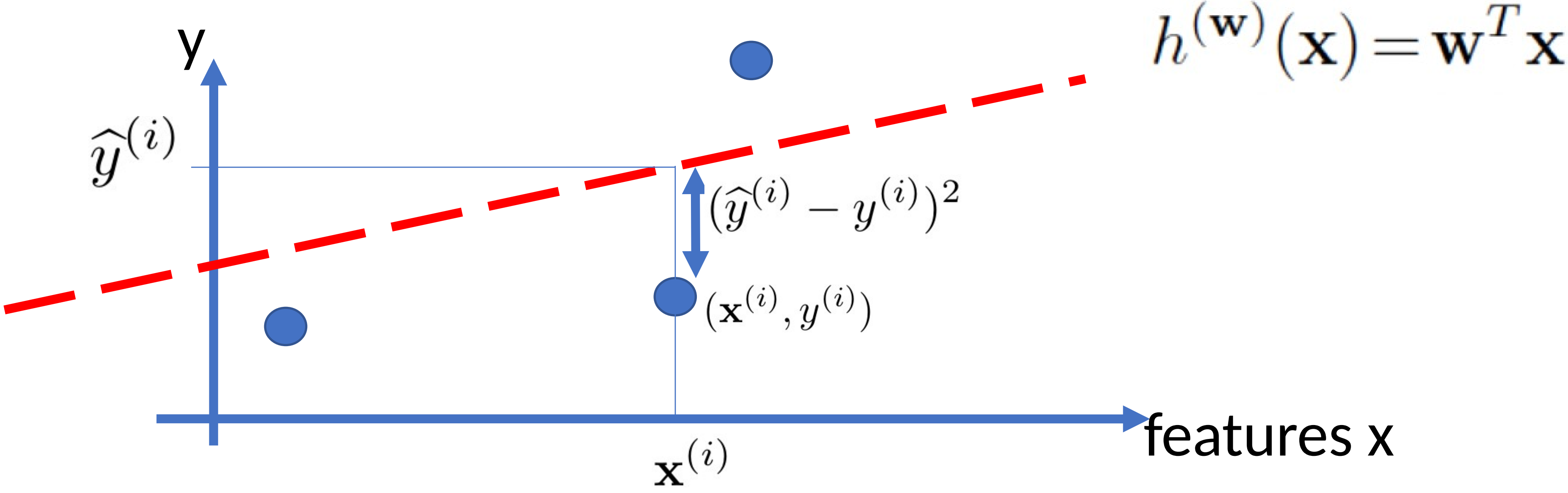
Learn a **hypothesis in model** that incurs in **smallest empirical risk (loss)** when predicting labels of **training data points**

ERM for regression

Linear Regression

- **Data**: Data points characterized by **numeric feature vector and numeric label**
- **Model:** model consists of **linear hypothesis maps**
- **Loss: squared error loss**

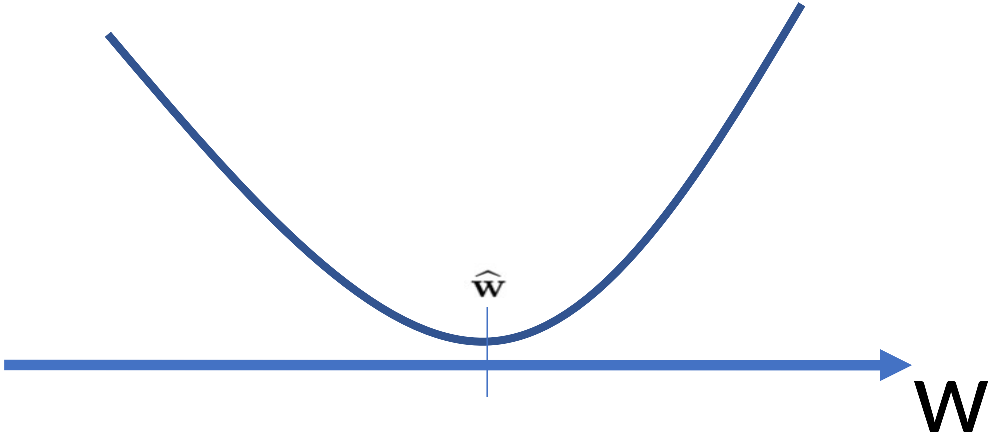
Linear Regression



Choose parameter/weight vector **w** to minimize average squared error loss

ERM for linear regression

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



ERM for linear regression in Python

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

`sklearn.linear_model`.LinearRegression

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

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

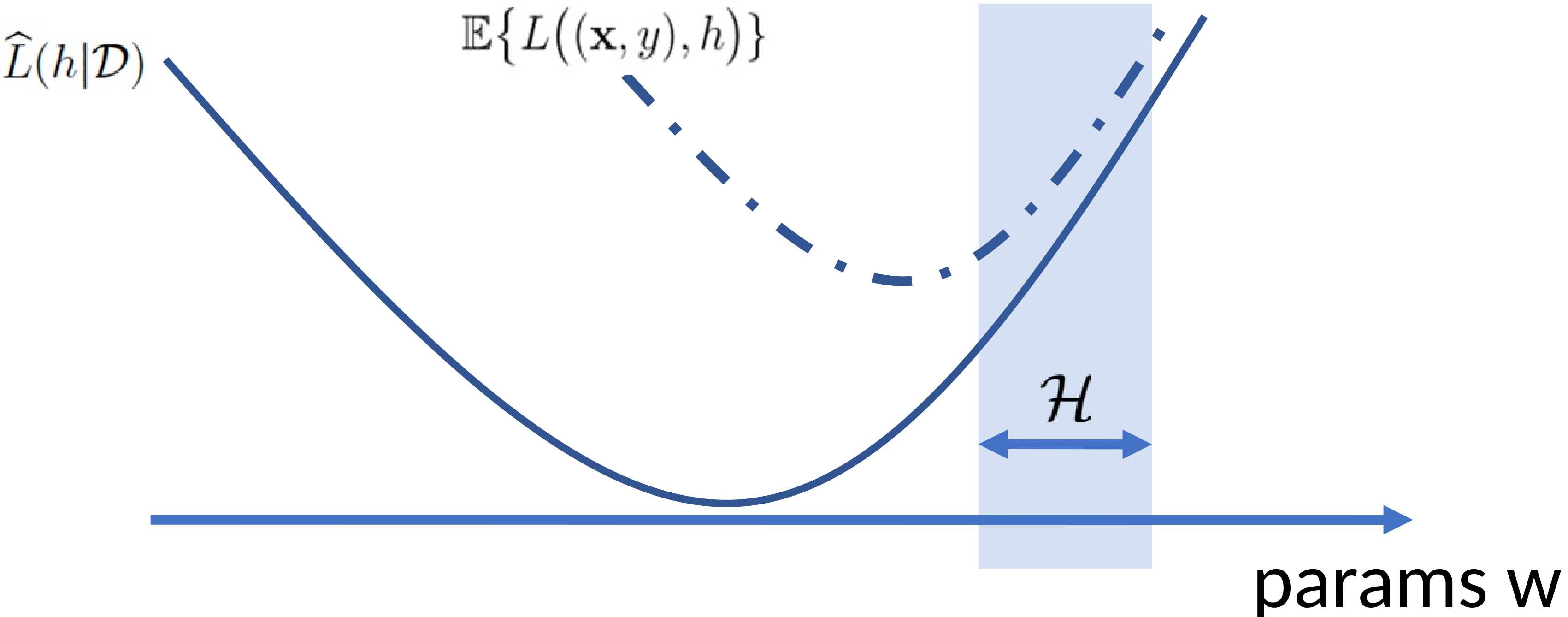
$$\mathbf{y} = (y^{(1)}, \ldots, y^{(m)})^T \in \mathbb{R}^m$$

ERM for linear regression in Python

```
# 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)
```

$\widehat{\mathbf{w}}$

$\widehat{L}\left(h^{(\mathbf{w})}|\mathcal{D}\right)$



Feature map + Linear model

linear map

feature map

$\phi(\cdot)$

single feature x

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

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

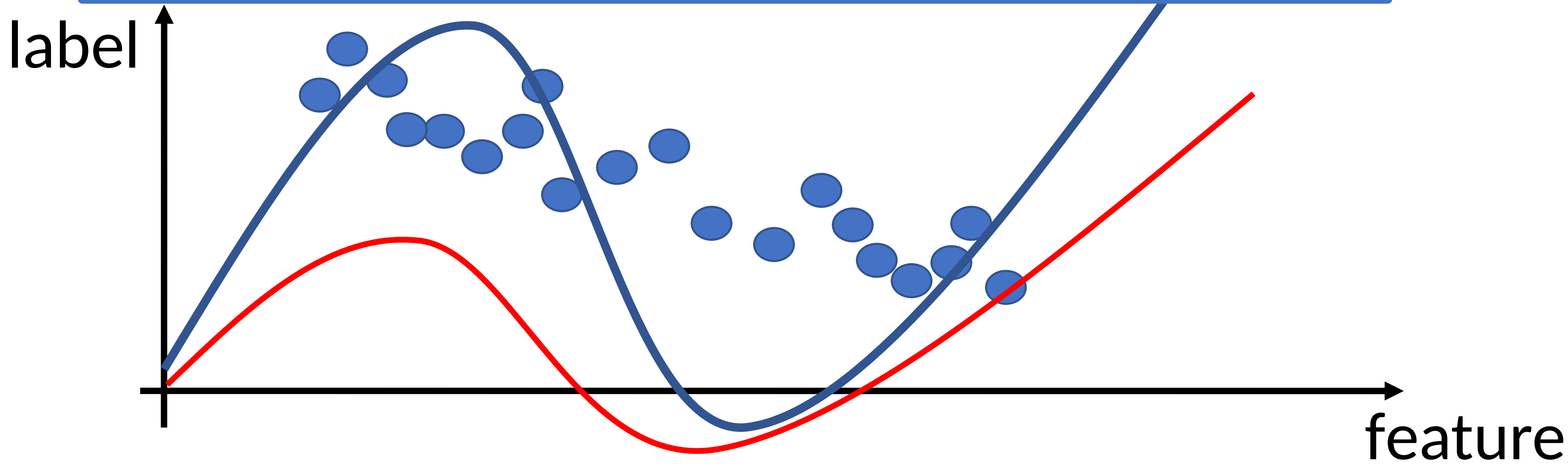
h(x)

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

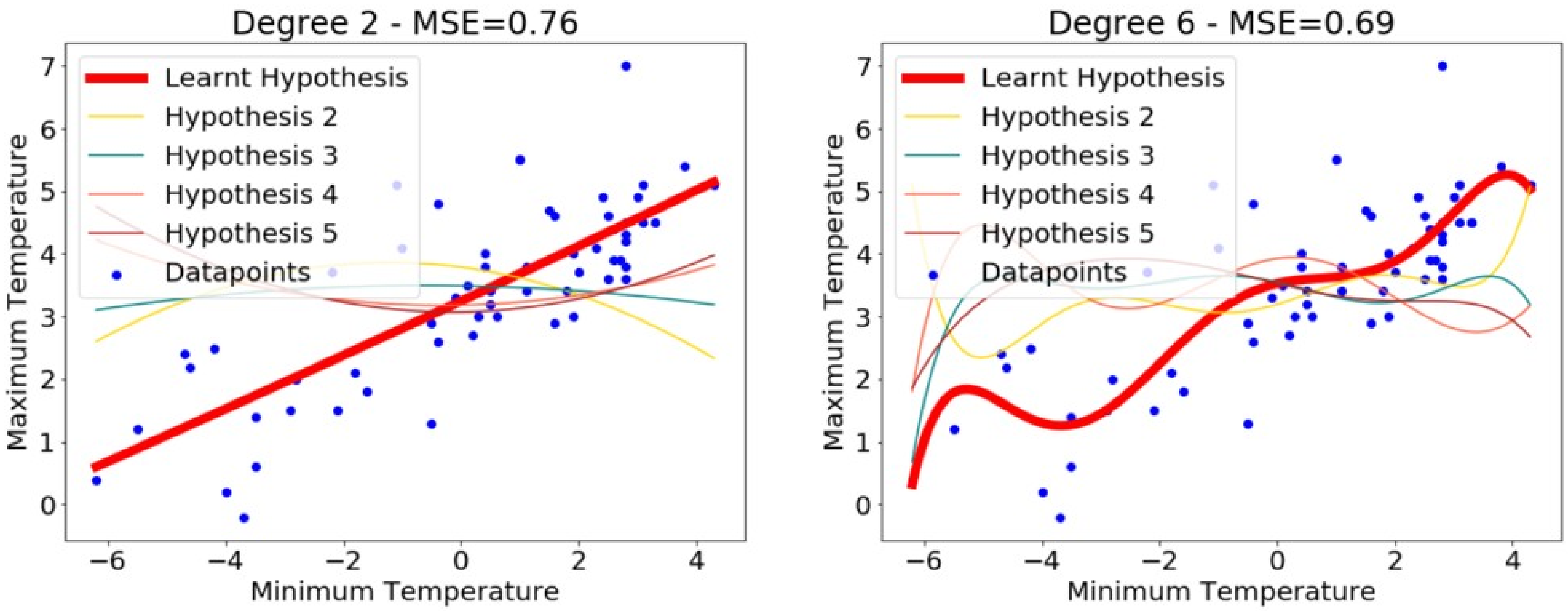
h(x) is linear in new features but non-linear in raw feature x

Polynomial regression

$$\mathcal{H}^{(n)}_{\mathrm{poly}} = \{h^{(\mathbf{w})} : \mathbb{R} \rightarrow \mathbb{R} : h^{(\mathbf{w})}(x) = \sum_{j=1}^{n} w_j x^{j-1},$$
$$\text{with some } \mathbf{w} = (w_1, \ldots, w_n)^T \in \mathbb{R}^n\}. \tag{3.4}$$



Polynomial regression



https://github.com/alexjungaalto/cs-c3240spring2022/blob/main/George_Demo_PolynomialRegression.ipynb

Feature map + Linear model

`sklearn.preprocessing`.PolynomialFeatures

`sklearn.linear_model`.LinearRegression

single feature x

feature map $\phi(\cdot)$

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

linear map

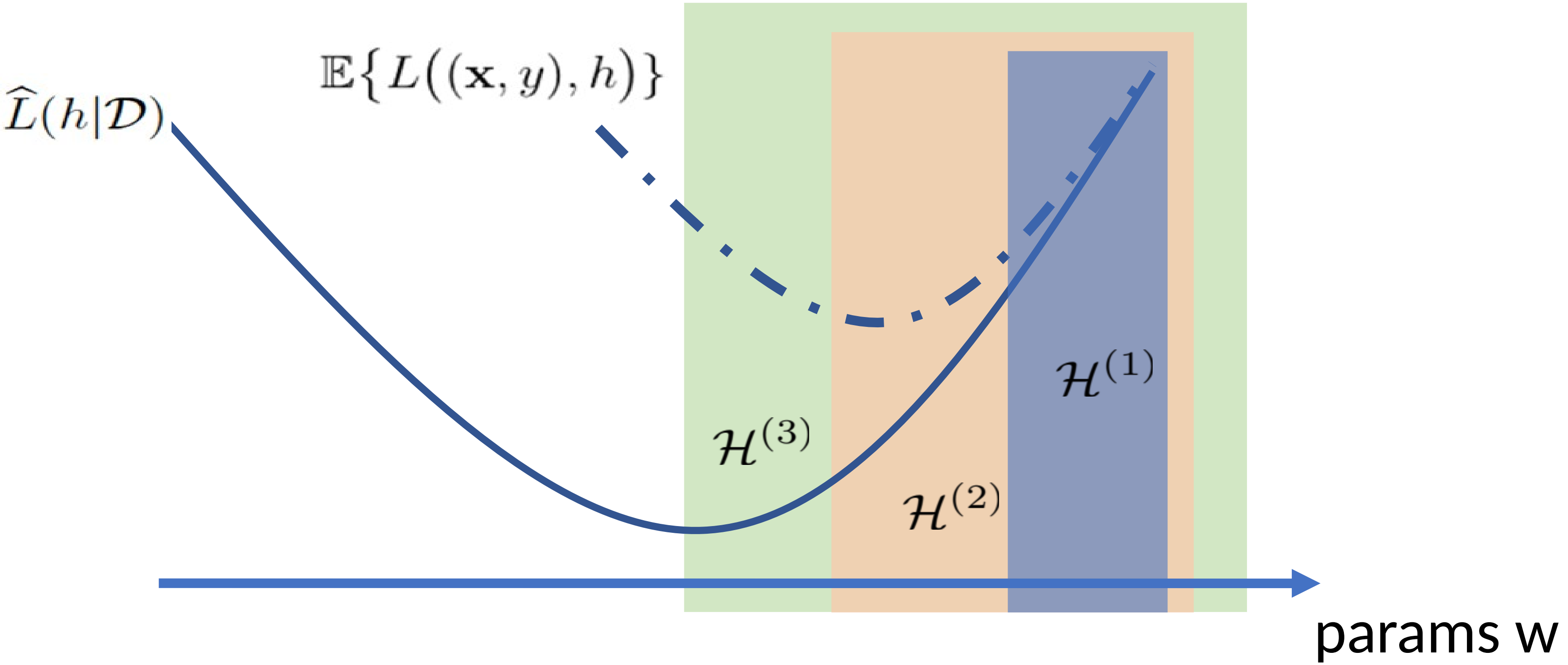
$$\mathbf{w}^T\mathbf{x} = \sum_{j=1}^{n} w_j x_j$$

h(x)

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

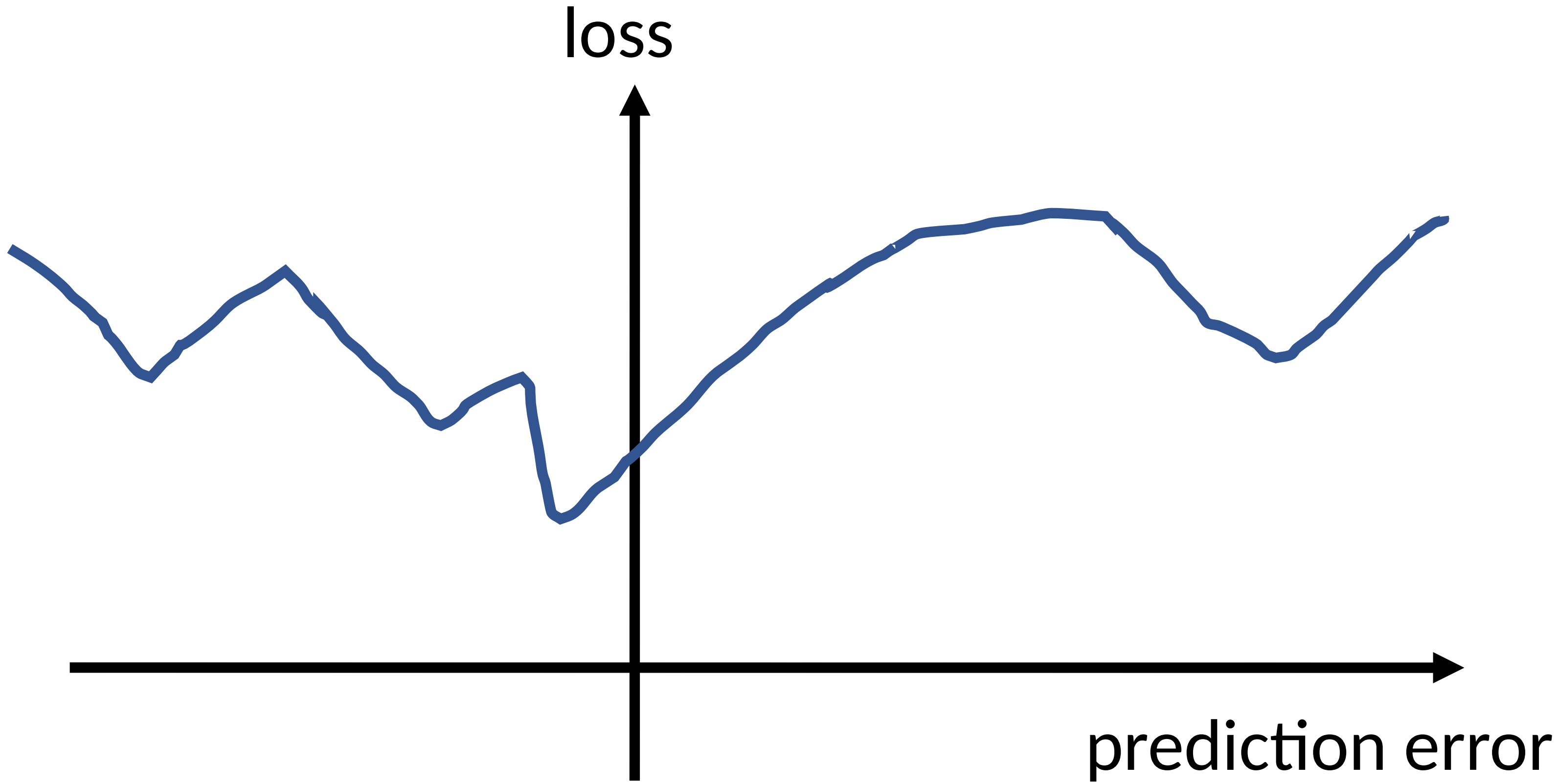
Polynomial regression = Linear regression with feature transformation

Which model is best?



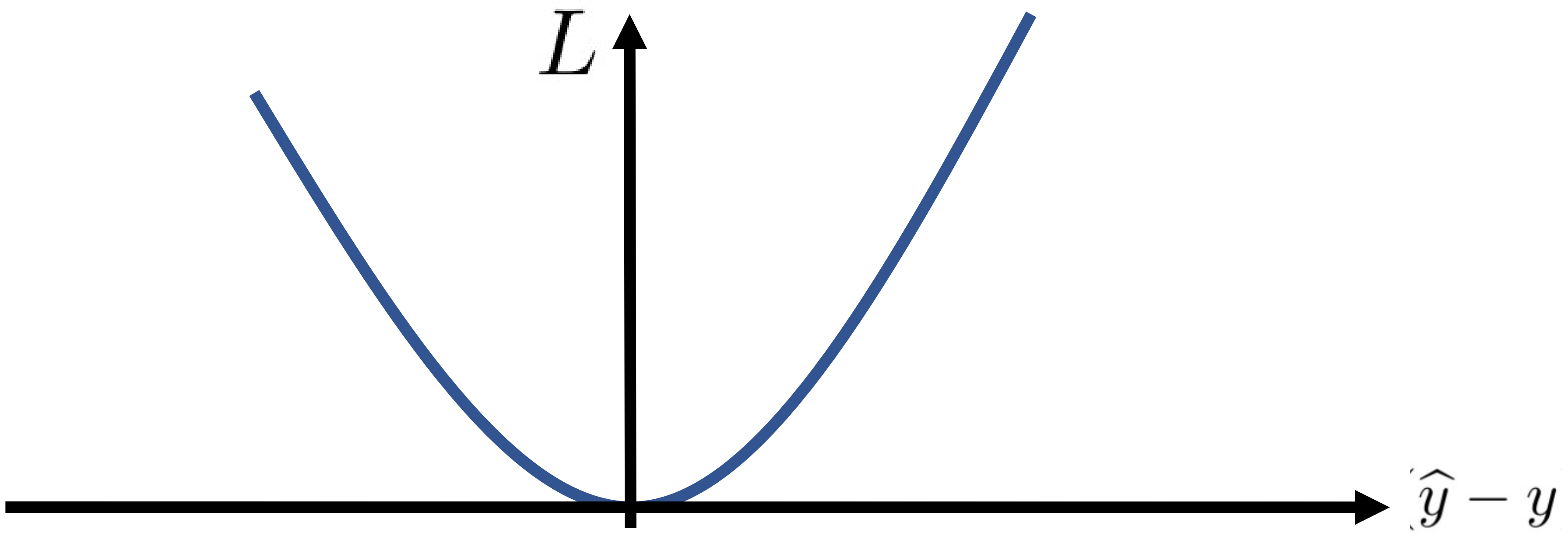
Measuring error via loss function

Loss function is also design choice !



Squared error loss

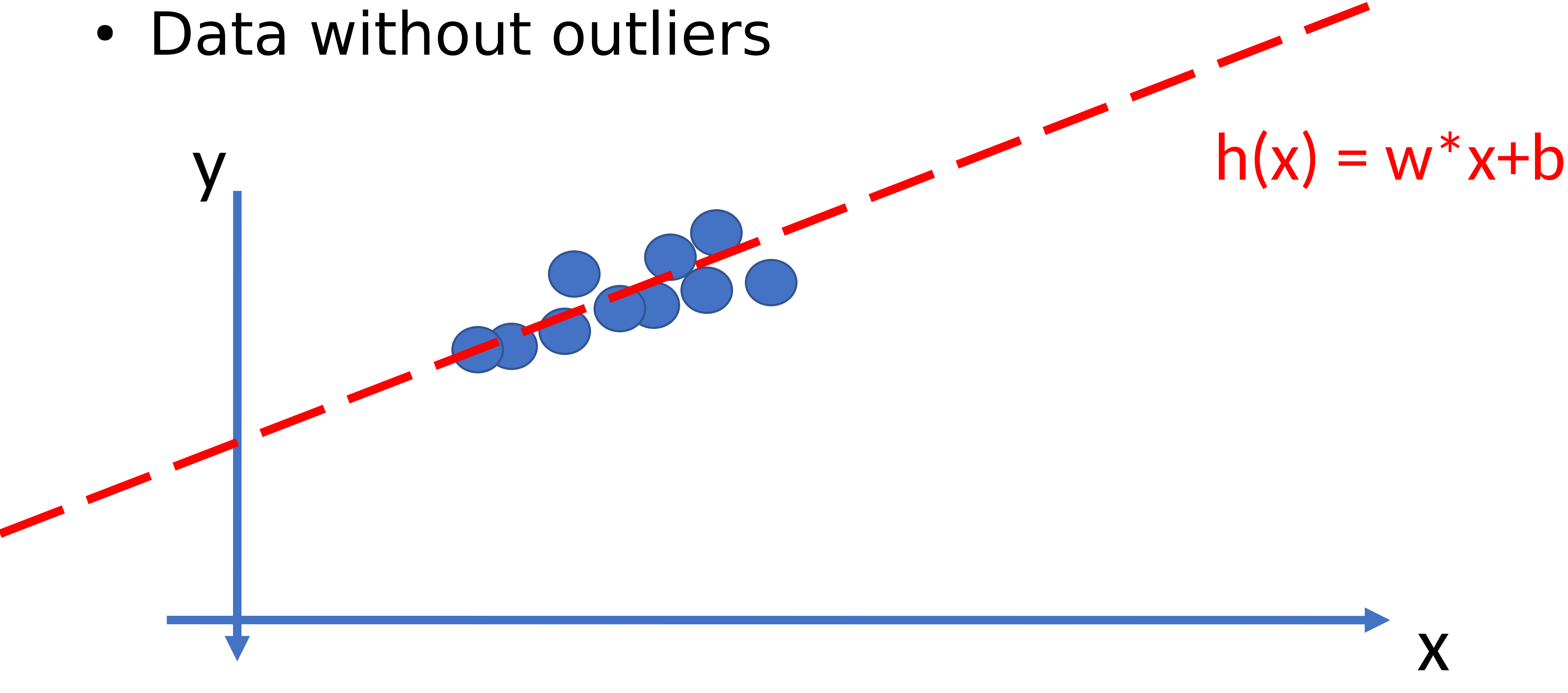
$$L := (\widehat{y} - y)^2$$



prediction error

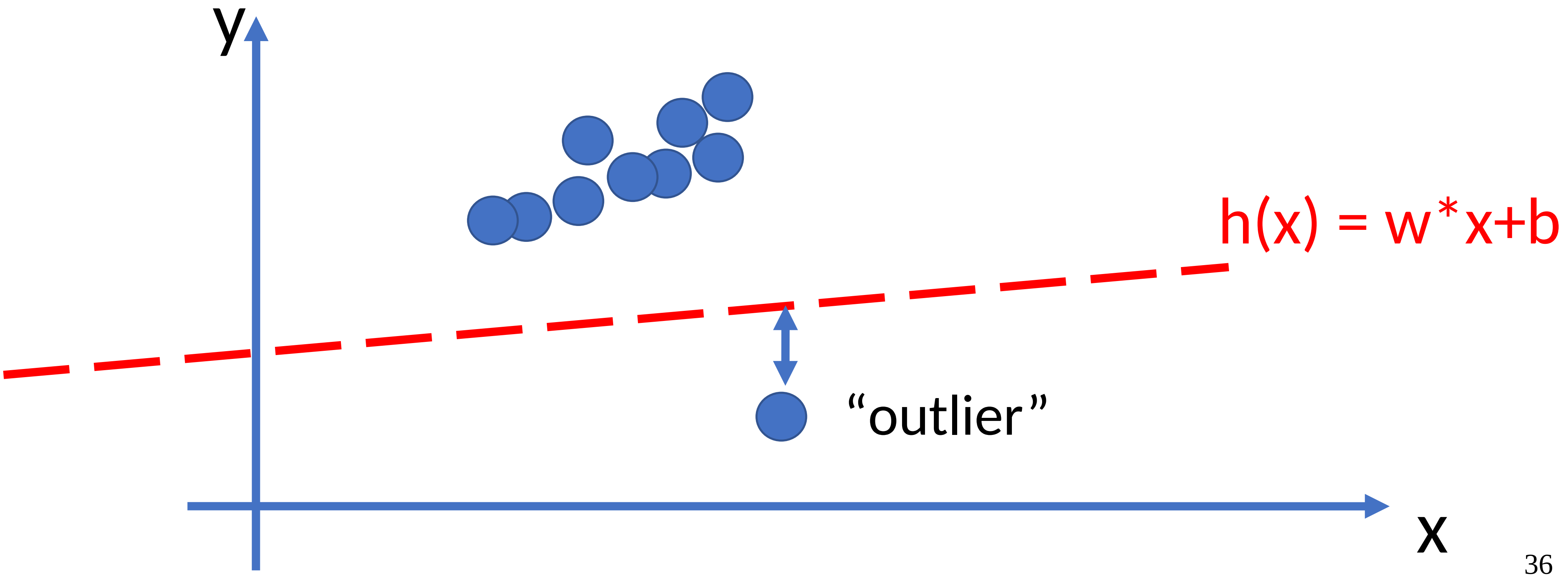
Squared error loss is sensitive to outliers

- Data without outliers



Squared error loss is sensitive to outliers

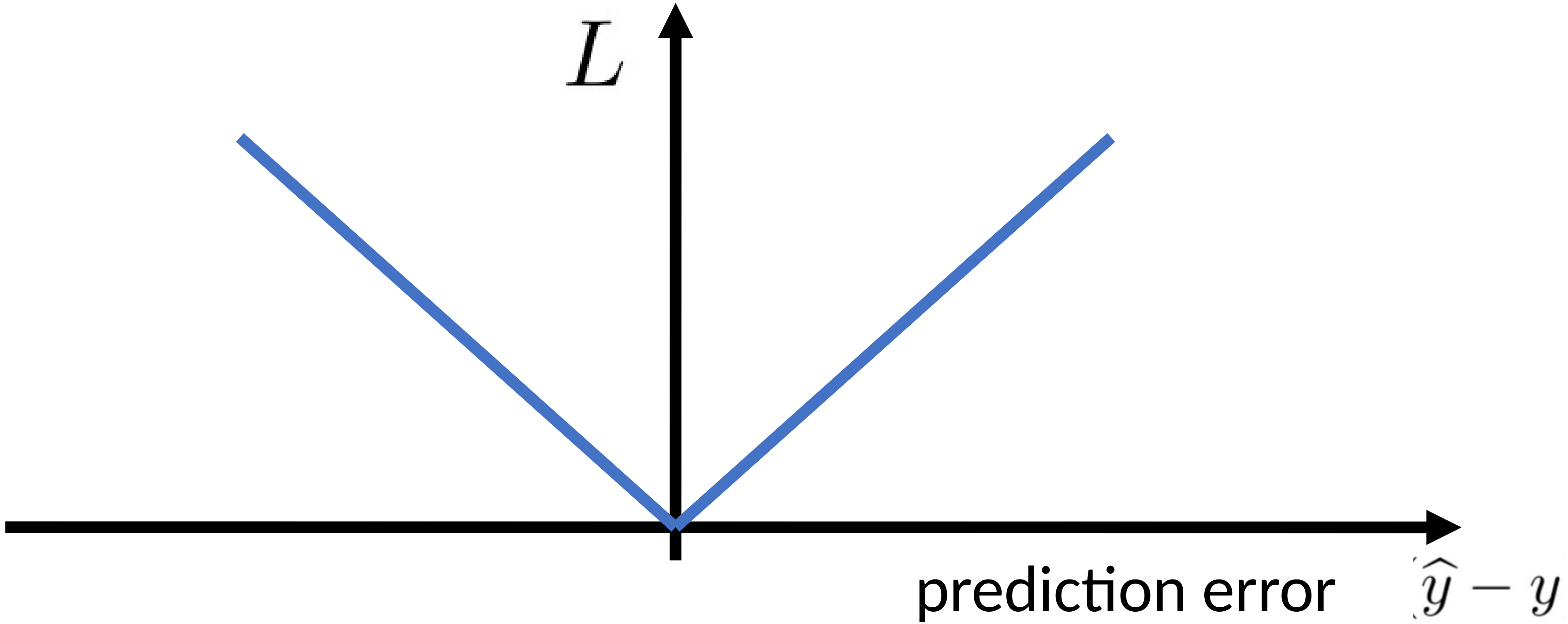
- Training set with single outlier



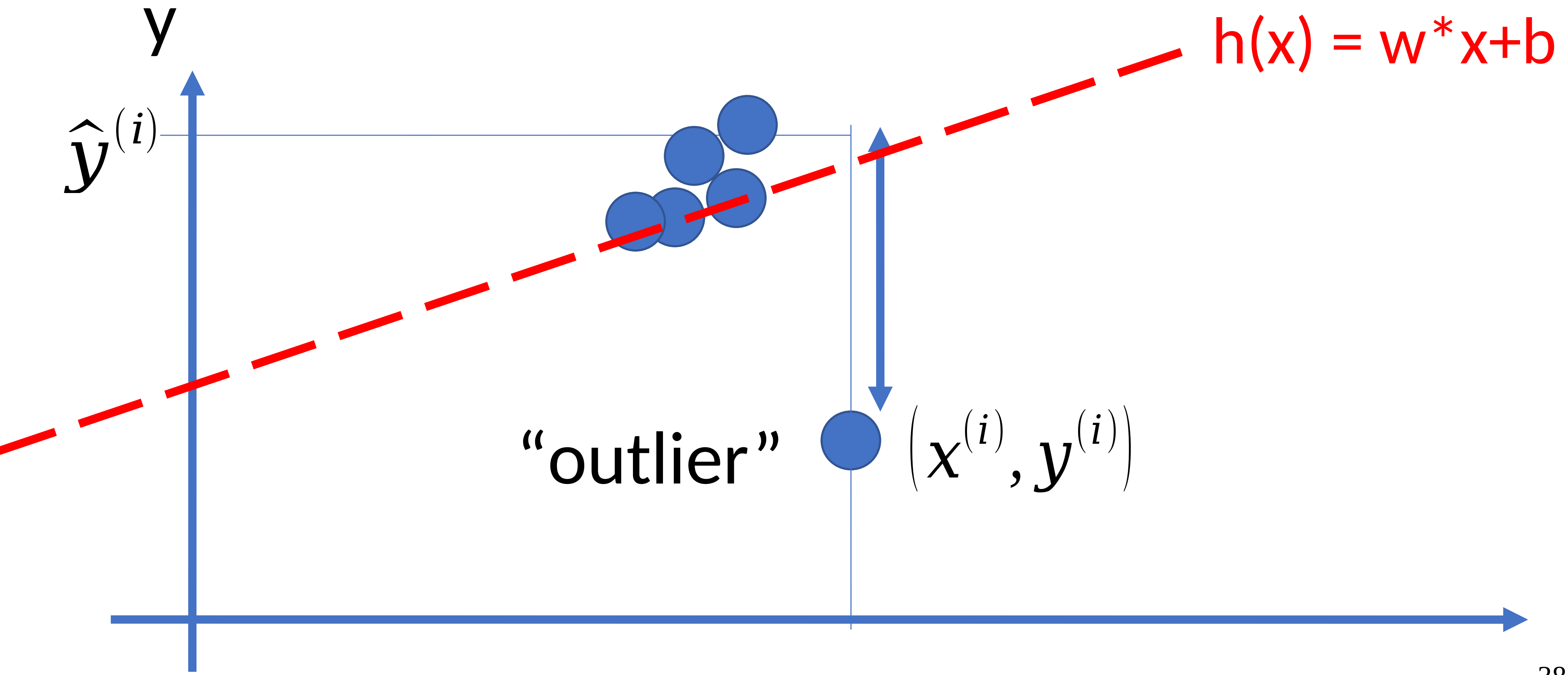
Minimize squared error loss forces predictor towards outlier

Absolute error loss

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



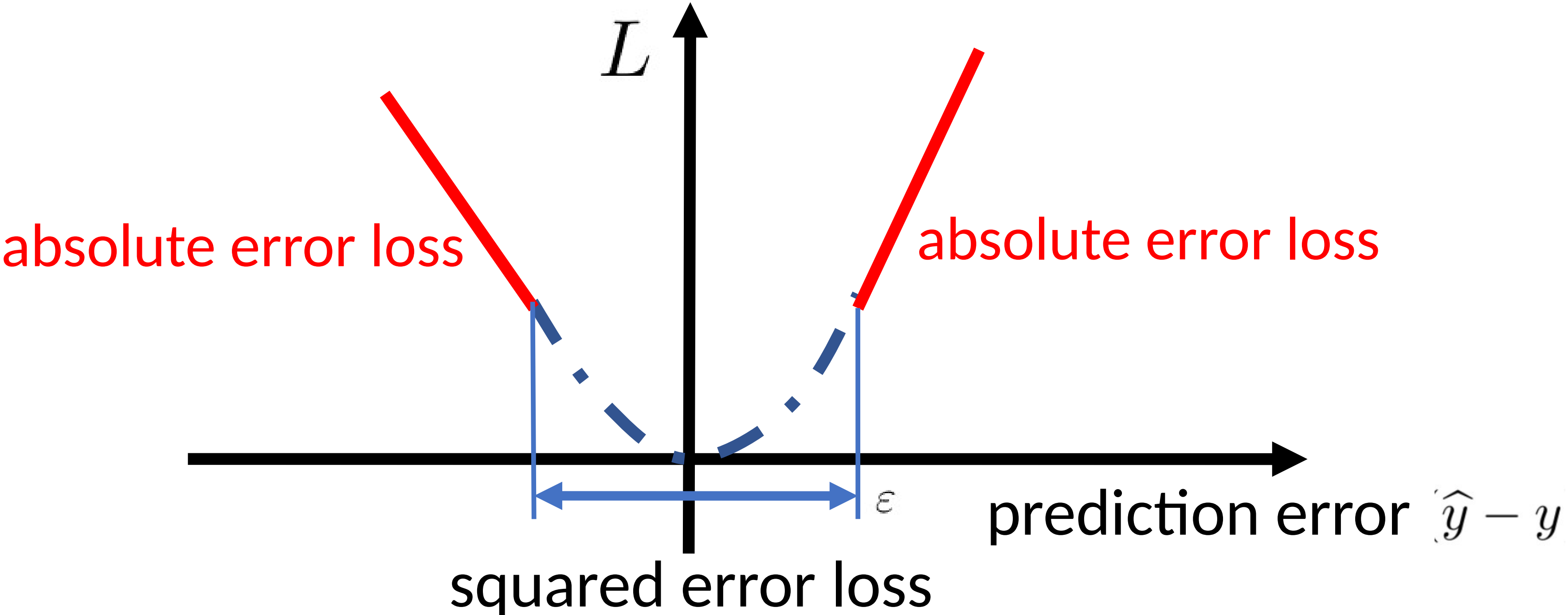
Absolute error loss is robust to outliers



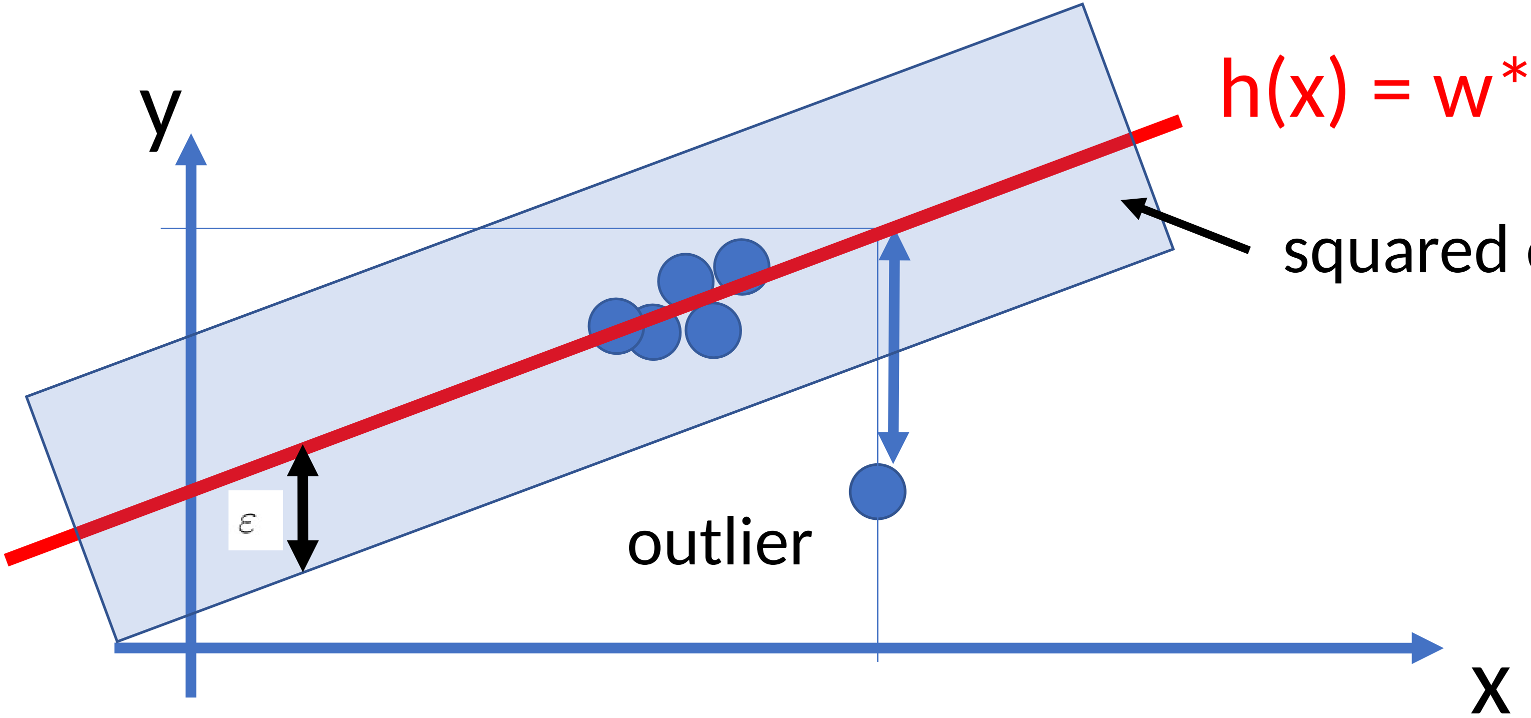
Absolute error "tolerates" few outliers

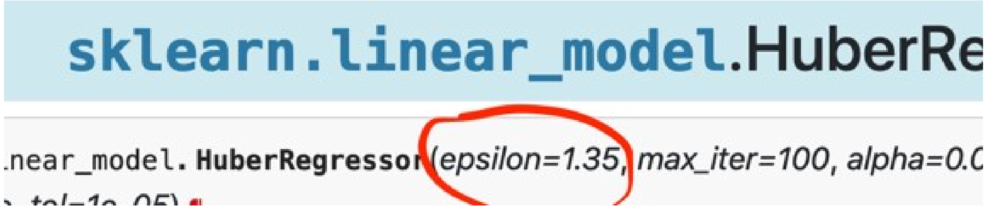
Huber loss

$$L\left((\mathbf{x}, y), h\right) = \begin{cases} (1/2)(y - h(\mathbf{x}))^2 & \text{for } |y - h(\mathbf{x})| \leq \varepsilon \\ \varepsilon(|y - h(\mathbf{x})| - \varepsilon/2) & \text{else.} \end{cases}$$



Linear predictor with Huber Loss




sklearn.linear_model.HuberRe

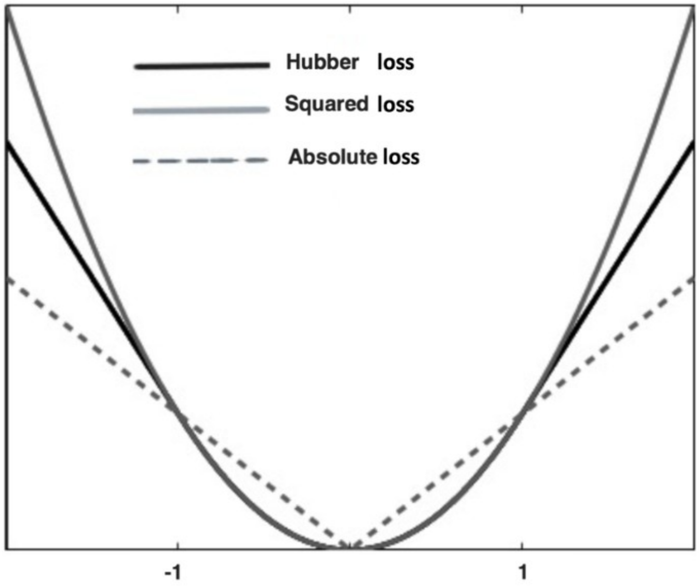
.near_model. HuberRegressor(epsilon=1.35, max_iter=100, alpha=0.0

Loss comparison

	Differentiable	Robust to outliers	Insensitive to noise
Absolute Loss	No	Yes	No
Squared Loss	Yes	No	Yes
Huber Loss	Yes	Yes	Yes

All of them are convex functions

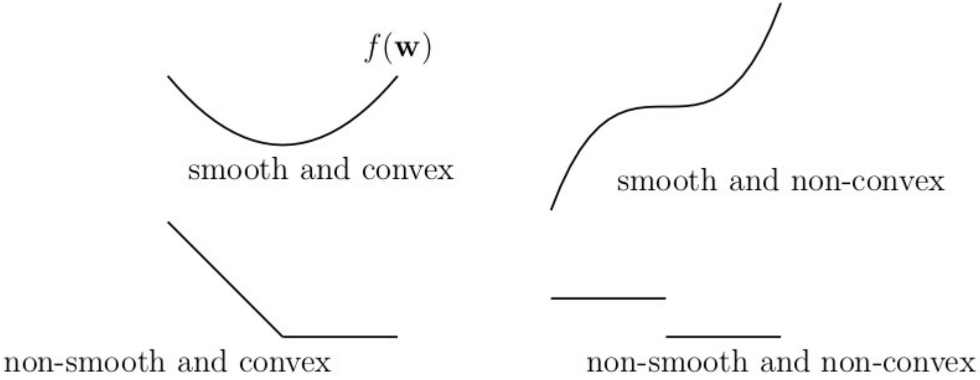
Insensitive to noise: deviations of the samples $y^{(i)}$ that are very close to the prediction $h(x^{(i)})$ have a lower effect on the loss



Loss comparison

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^n}{\operatorname{argmin}} f(\mathbf{w}) \qquad \text{with } f(\mathbf{w}) := \underbrace{(1/m)\sum_{i=1}^{m} L\big((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}\big)}_{\widehat{L}\left(h^{(\mathbf{w})}|\mathcal{D}\right)}.$$

- Non-convex objective functions are more difficult to minimize
- Non-differentiable objective functions are more difficult to minimize



ERM for classification

Regression vs. classification

- Regression
  - Numeric labels
  - Loss functions obtained from distance between numbers
- Classification
  - Categorical discrete-valued labels
    - If only 2 categories (e.g., y=-1 vs. y=1): binary classification
    - If more than 2 categories: multi-class classification
  - Loss functions obtained from “confidence” measures

Logistic regression

- **Data points** with numeric features -- same as in linear regression
- **Model** = space of linear maps -- same as in linear regression
- Logistic **loss** -- different from linear regression

Logistic regression

- linear hypothesis h(x) =$w^t x$
- sign of h(x) used for label prediction
  - h(x) > 0 means sign(h(x))= $\widehat{y}$ =1
  - h(x) < 0 means sign(h(x))= $\widehat{y}$ =-1
- |h(x)| used as confidence measure
  - h(x) = 100000 > 0 means very confident in $\widehat{y}$ =1
  - h(x) = -100000 < 0 very confident in $\widehat{y}$ = -1

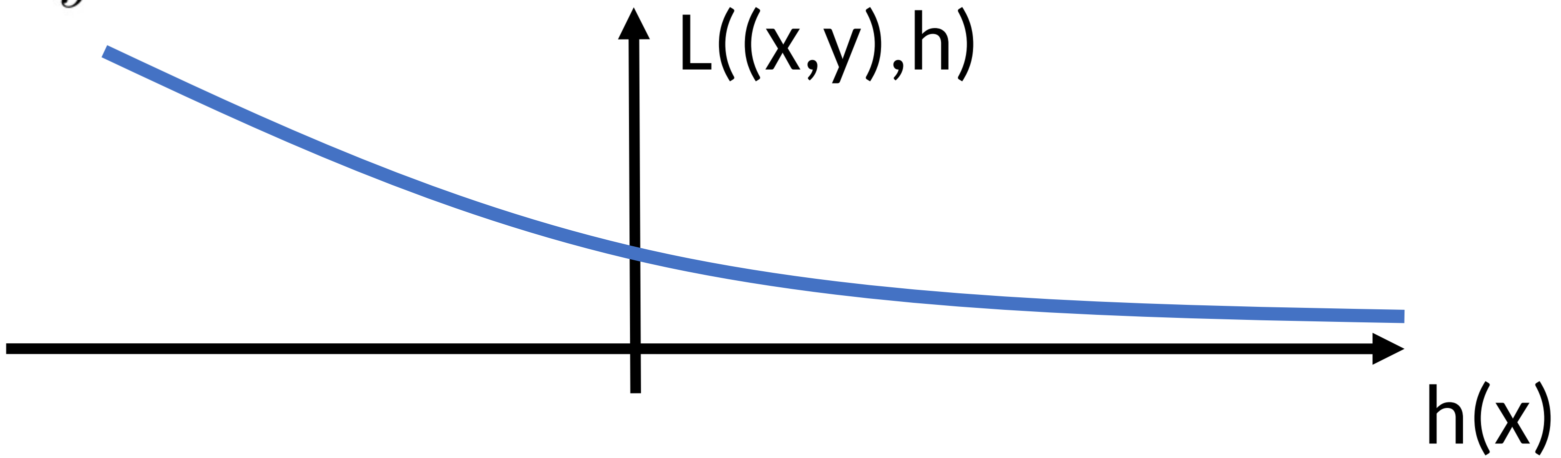
Logistics loss

formula when using -1 and 1 as label values

$$L((\mathbf{x}, y), h) := \log(1 + \exp(-yh(\mathbf{x}))).$$

differentiable and convex as function of h(x) and, in turn, of weight w for linear $h(x) = w^t x$

if $y = 1$



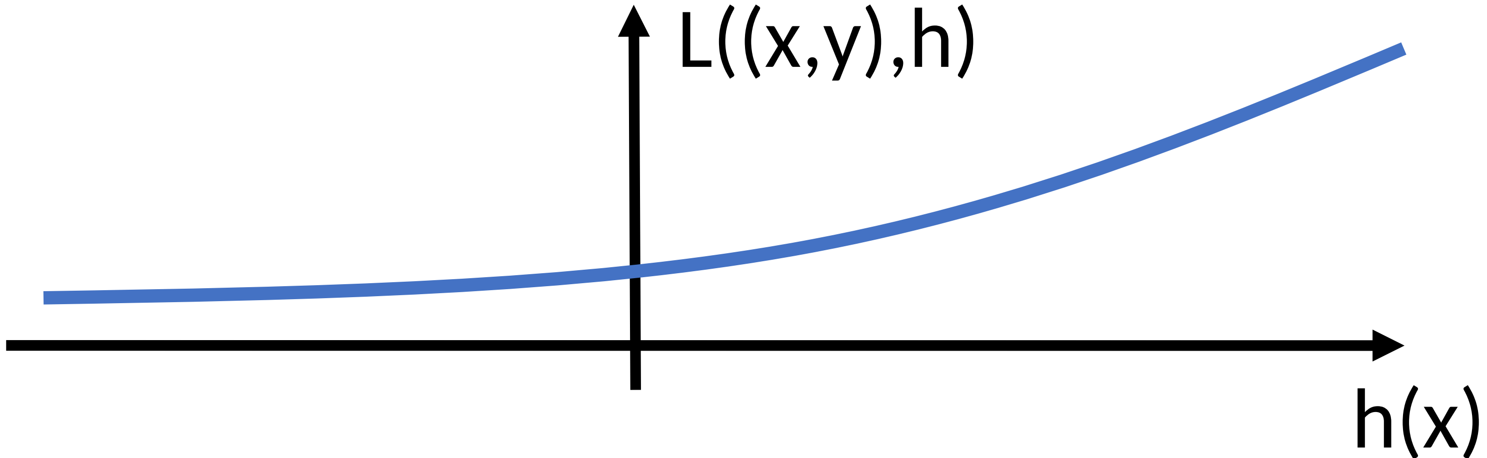
Logistics loss

formula when using -1 and 1 as label values

$$L((\mathbf{x}, y), h) := \log(1 + \exp(-yh(\mathbf{x}))).$$

differentiable and convex as function of h(x) and, in turn, of weight w for linear $h(x) = w^t x$

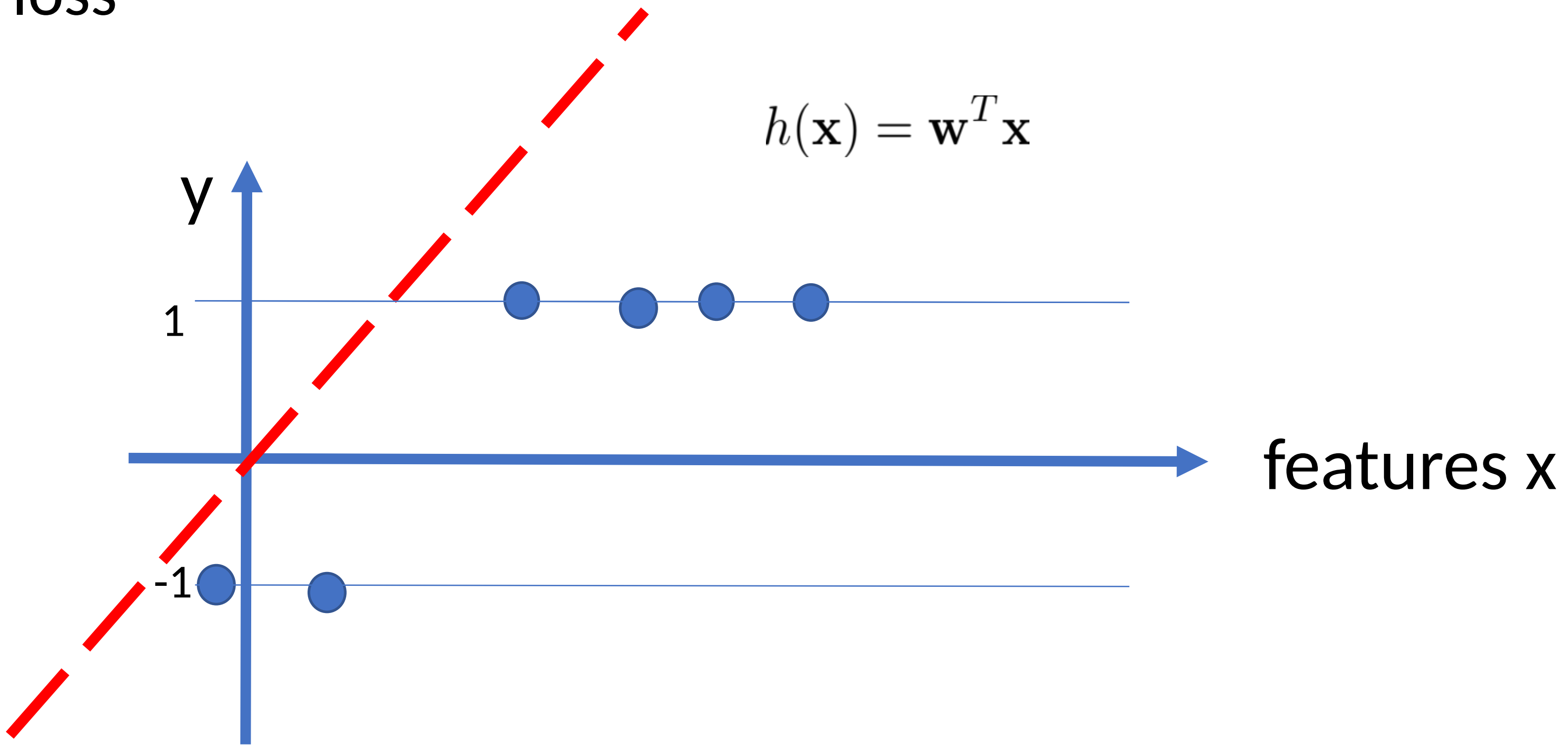
if $y = -1$



Why not Squared Loss?

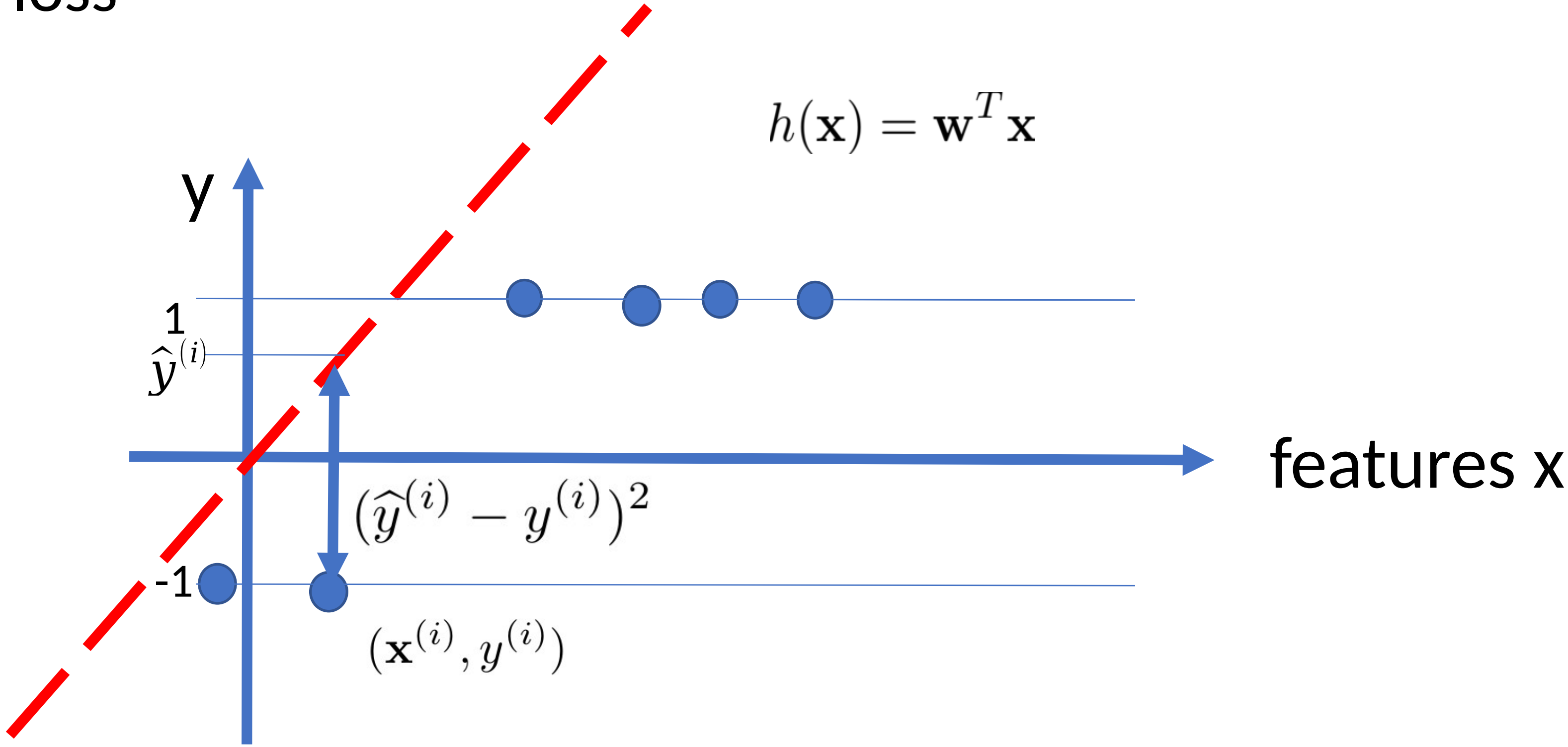
Choose parameter/weight vector **w** to minimize average squared error loss

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



Why not Squared Loss?

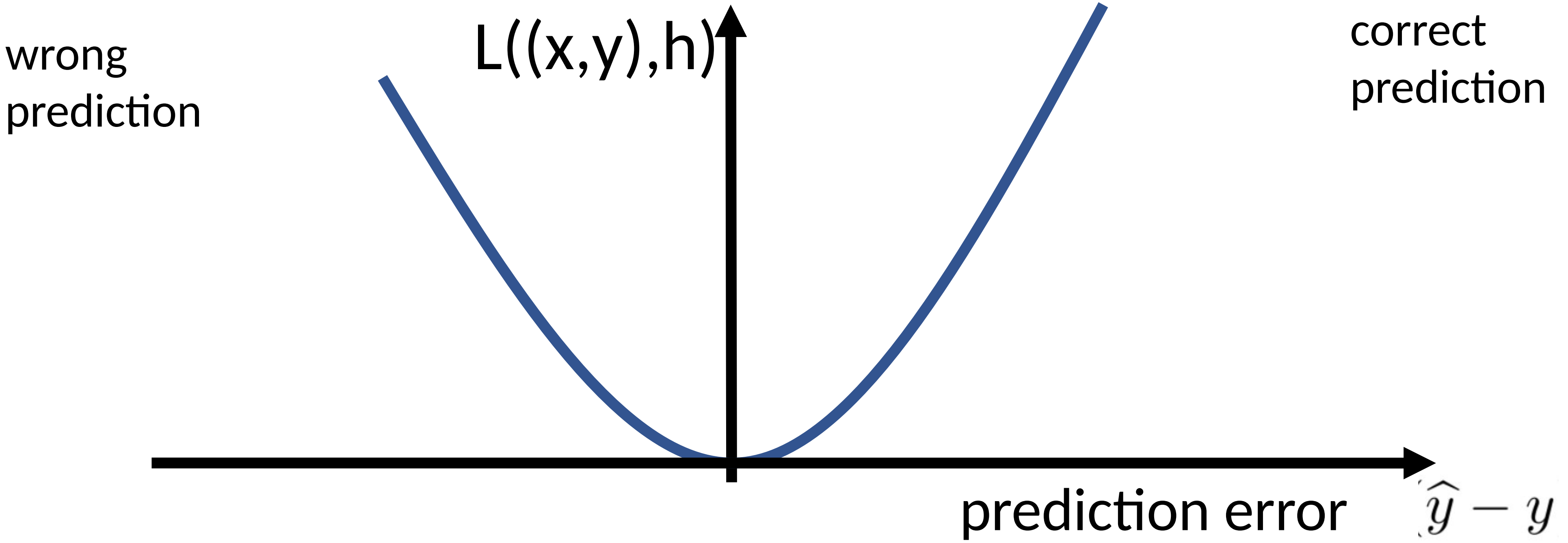
Choose parameter/weight vector **w** to minimize average squared error loss



Squared error loss

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

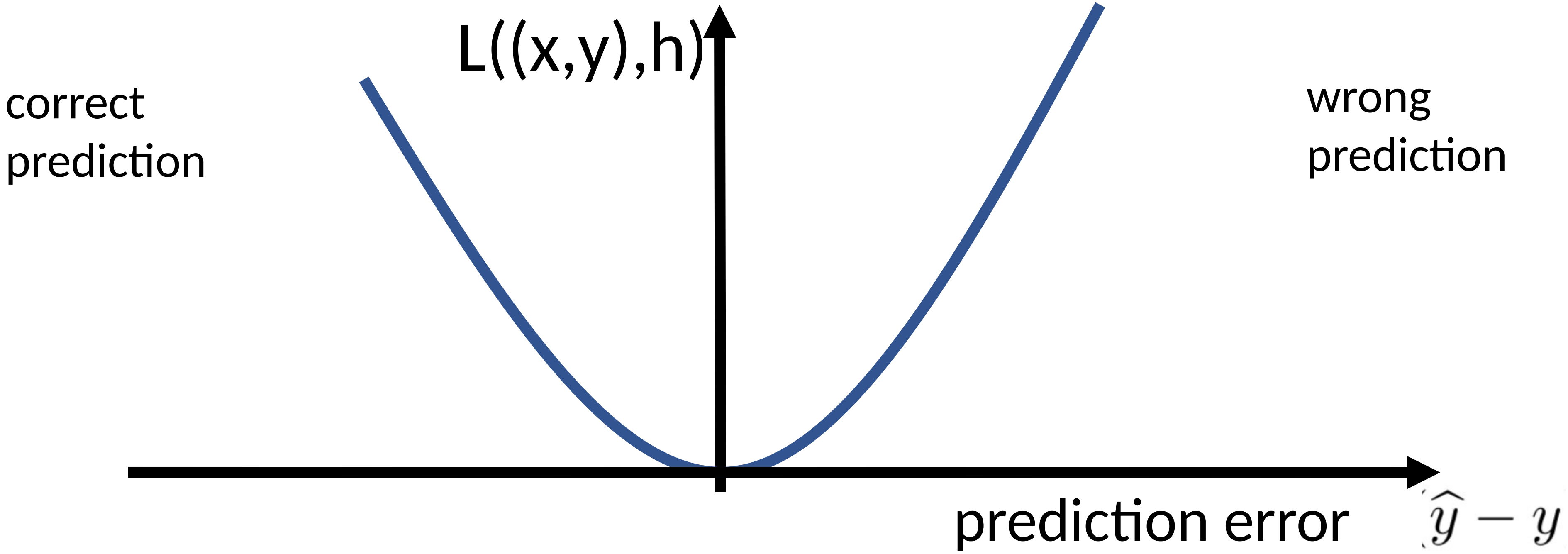
$$\text{if } y = 1$$



Squared error loss

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

if $y = -1$



Logistic regression – decision boundary in 2D

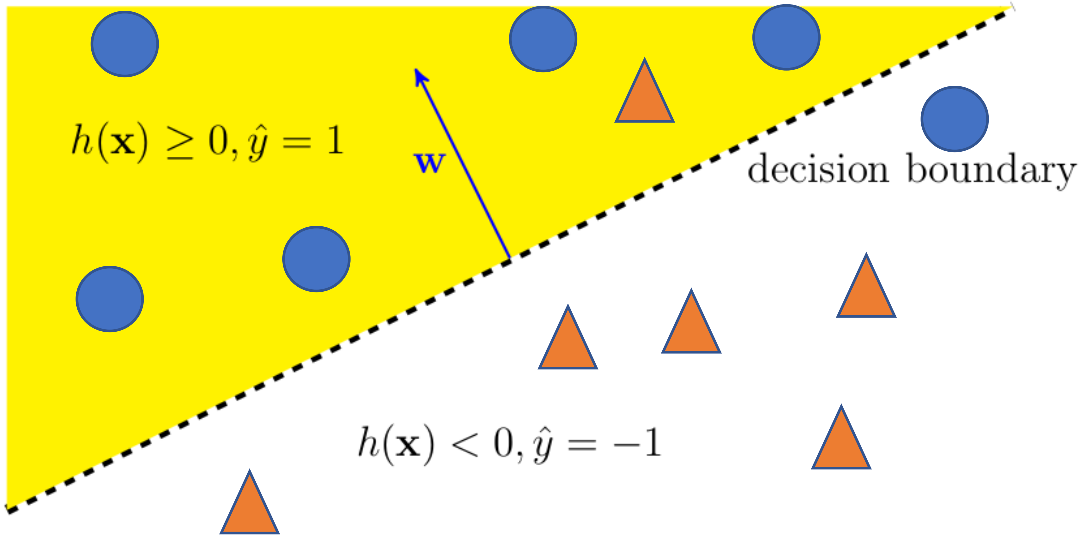


Figure 2.9: A hypothesis $h : \mathcal{X} \to \mathcal{Y}$ for a binary classification problem, with label space $\mathcal{Y} = \{-1, 1\}$ and feature space $\mathcal{X} = \mathbb{R}^2$, can be represented conveniently via the decision boundary (dashed line) which separates all feature vectors $\mathbf{x}$ with $h(\mathbf{x}) \geq 0$ from the region of feature vectors with $h(\mathbf{x}) < 0$. If the decision boundary is a hyperplane $\{\mathbf{x} : \mathbf{w}^T\mathbf{x} = b\}$ (with normal vector $\mathbf{w} \in \mathbb{R}^n$), we refer to the map h as a linear classifier.

Logistic regression in python

sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.LogisticRegression`(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

Logistic regression: probabilistic interpretation

- interpret label of data point as realization of binary RV with probability

$$p(y=1;\mathbf{w}) = 1/(1+\exp(-\mathbf{w}^T\mathbf{x}))$$

$$\overset{h^{(\mathbf{w})}(\mathbf{x})=\mathbf{w}^T\mathbf{x}}{=} 1/(1+\exp(-h^{(\mathbf{w})}(\mathbf{x}))))$$

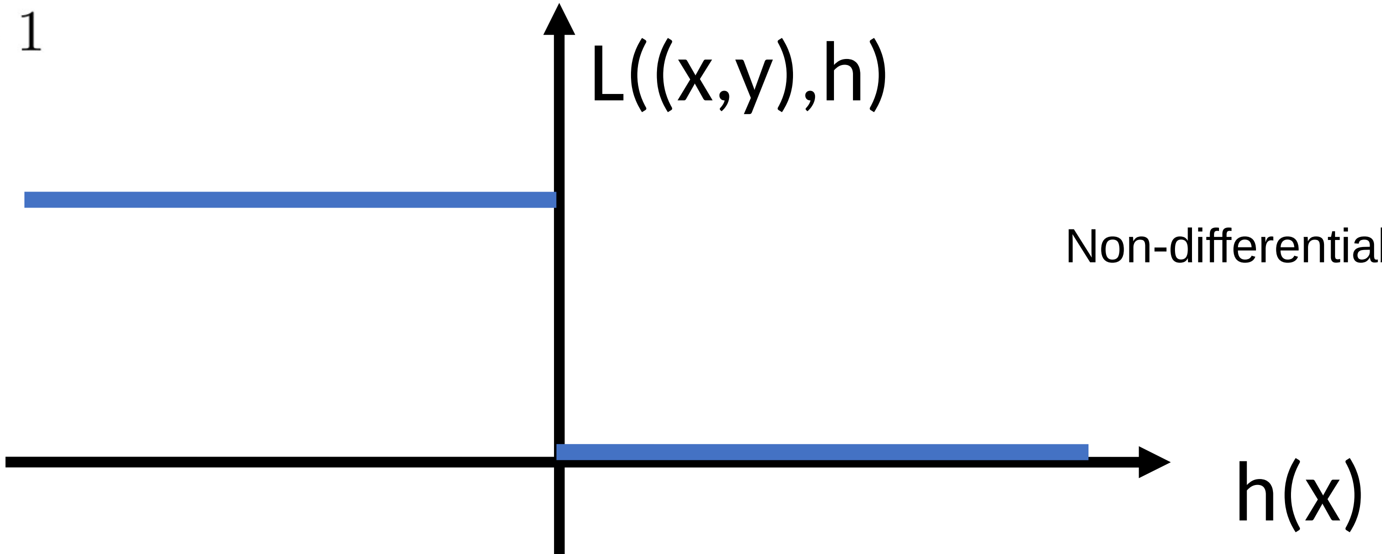
- Maximum likelihood estimation for **w** equivalent to logistic regression - see Sec. 3.6 of MLBook

Losses in classification

- 0/1 loss

$$L\left((\mathbf{x}, y), h\right) := \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{else,} \end{cases} \quad \text{with } \hat{y} = 1 \text{ for } h(\mathbf{x}) \geq 0, \text{ and } \hat{y} = -1 \text{ for } h(\mathbf{x}) < 0$$

if $y = 1$



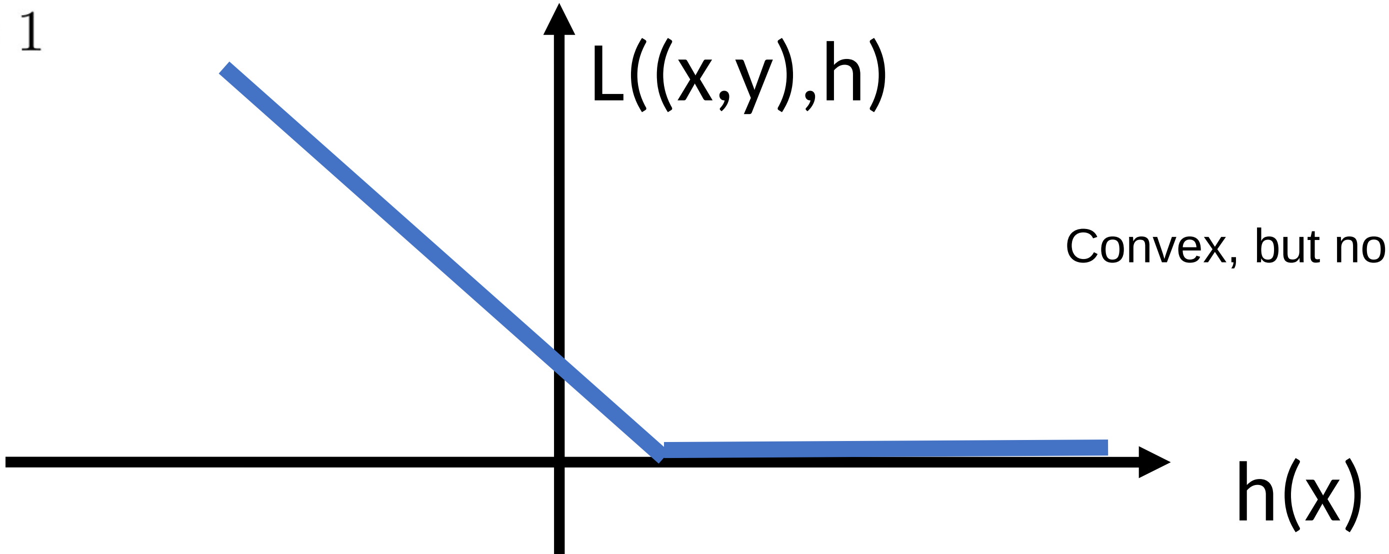
Non-differentiable and non-convex

Losses in classification

- Hinge loss

$$L\left((\mathbf{x}, y), h\right) := \max\{0, 1 - yh(\mathbf{x})\}.$$

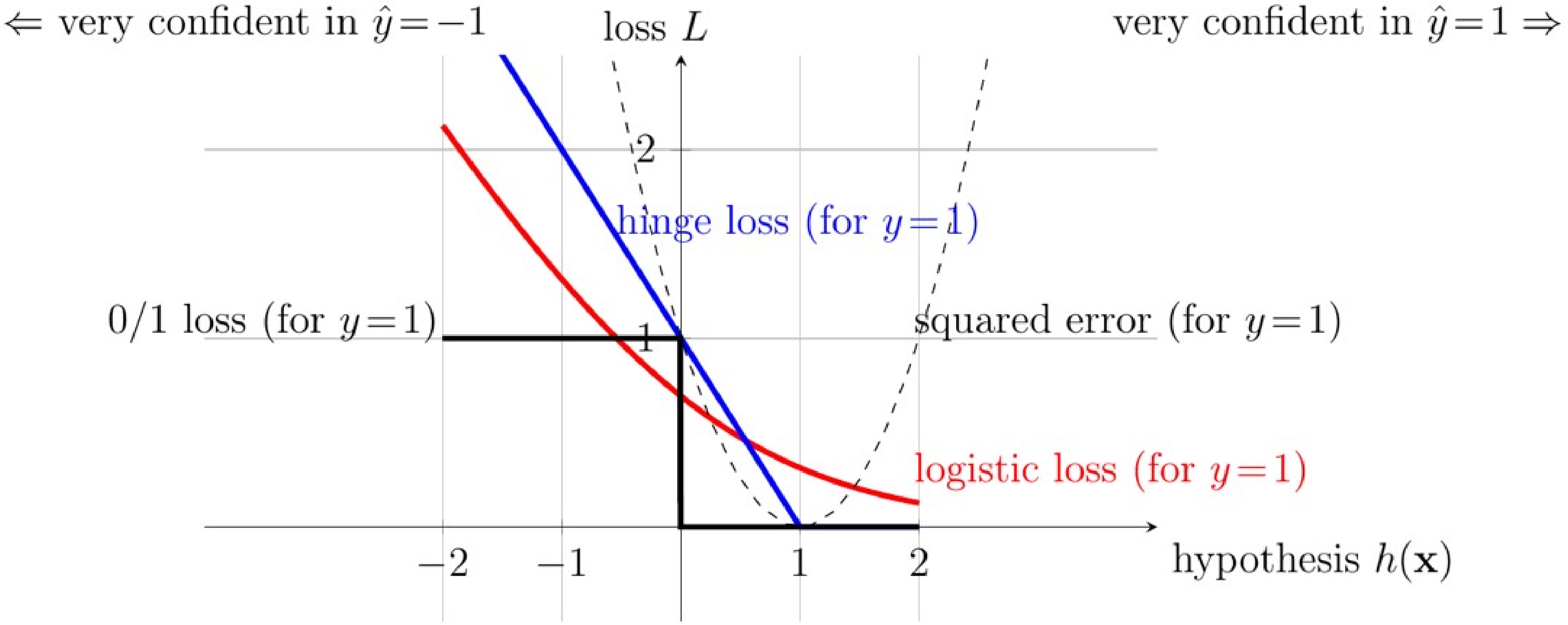
if $y = 1$



Convex, but non differentiable

Losses in classification

if $y = 1$



Summary

- **Approximate risk by average loss (empirical risk) on training data**
- ERM can fail if empirical risk deviates from risk
- Many ML methods are instances of ERM
- Three design choices of ERM: data, model and loss
- Examples of ERM regression and classification: linear models and derived ones
- In classification loss is obtained from confidence measures

Any questions?

Self-assessment quiz

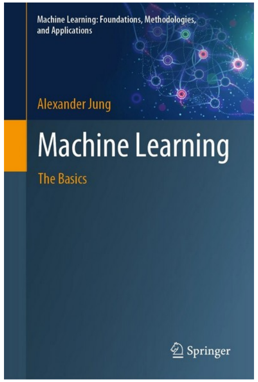
- Write the expected risk minimization problem for linear regression of a single feature $x \in \mathbb{R}$, if data p(x,y) follows a multivariate normal distribution with mean 0 and covariance matrix $S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, with squared error loss

- Write the empirical risk minimization problem for linear regression of the following data, with squared error loss

Object id	Feature 1	Feature 2	Label
a	0	10	2
b	-1	9	0.5
c	0	0	0
d	98	99	100

References: readings

- Chapter 2-3-4



Slide acknowledgments

- Alexander Jung – Aalto University