



DINFK

Linear regression

Introduction to Machine Learning, Lecture 2

Fanny Yang

ETH zürich



Announcements

- Math recap this **Friday 20.2.**
- Python tutorial next **Monday 23.2.** including conda setup (briefly) & Numpy
- Demos (requiring gitlab) are separate from projects
- Mandatory project (different from homeworks!)
 - We allow groups of max. 3 students (including groups of 1 or 2)
 - Each person submits a video even if you're a group of 3 and explains the entire solution (not just the part that you did). The idea is that to make sure you understood the steps
 - Teams are registered per project

Recap and plan for today

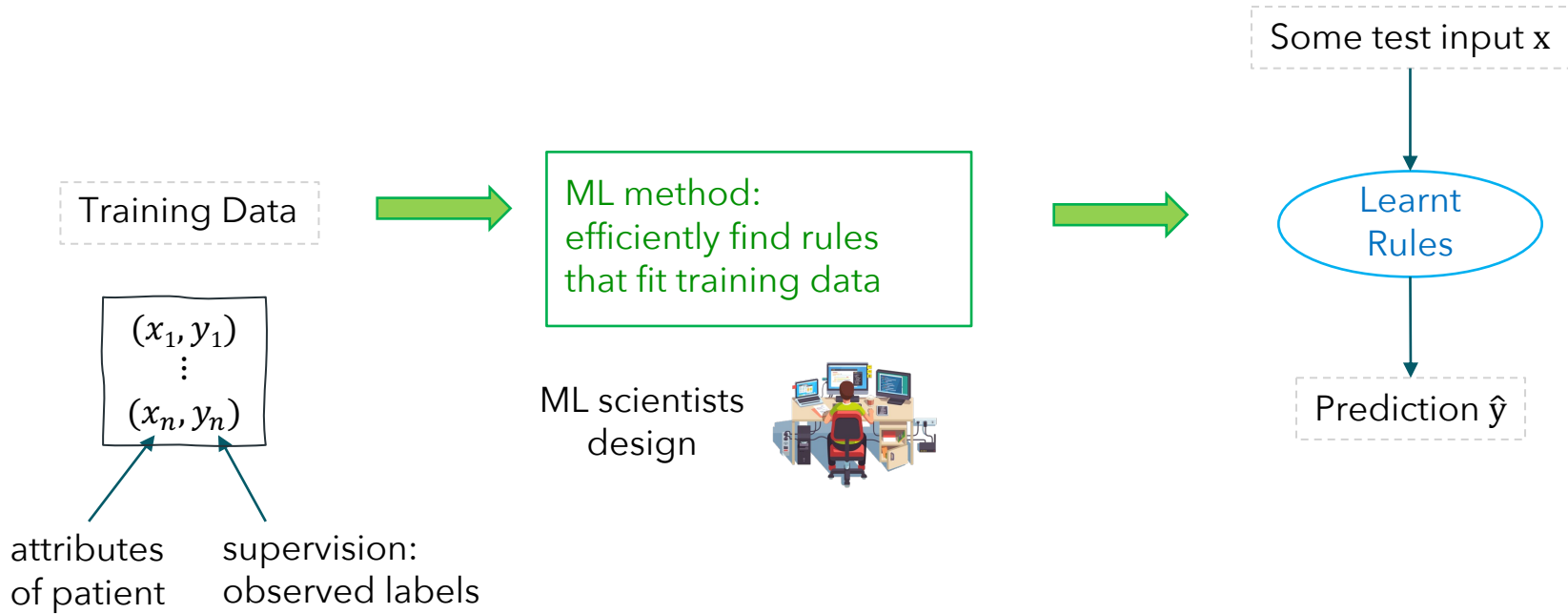
Last lecture: Different problems in machine learning

- supervised learning & unsupervised learning

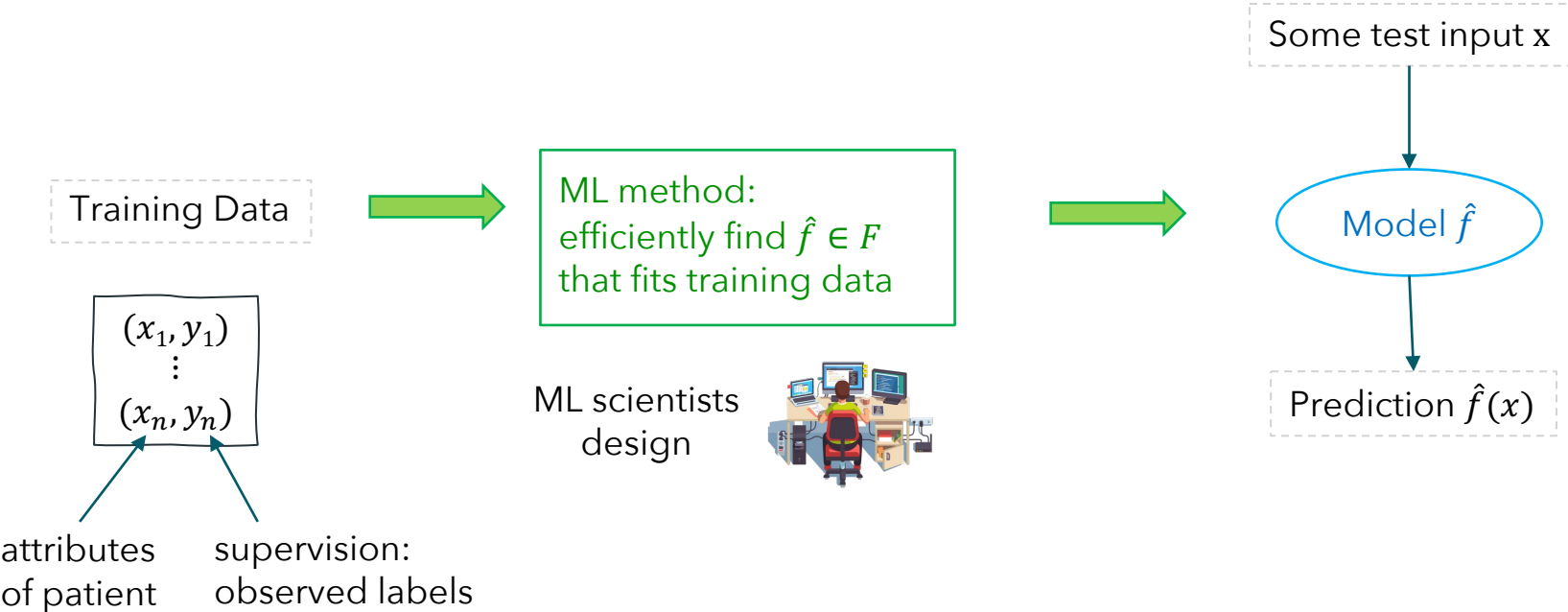
Today: Walking through the supervised learning pipeline with

- Example problem: Average housing price prediction
- the simplest ML method: solving linear regression in 1 dimension *and multiple*

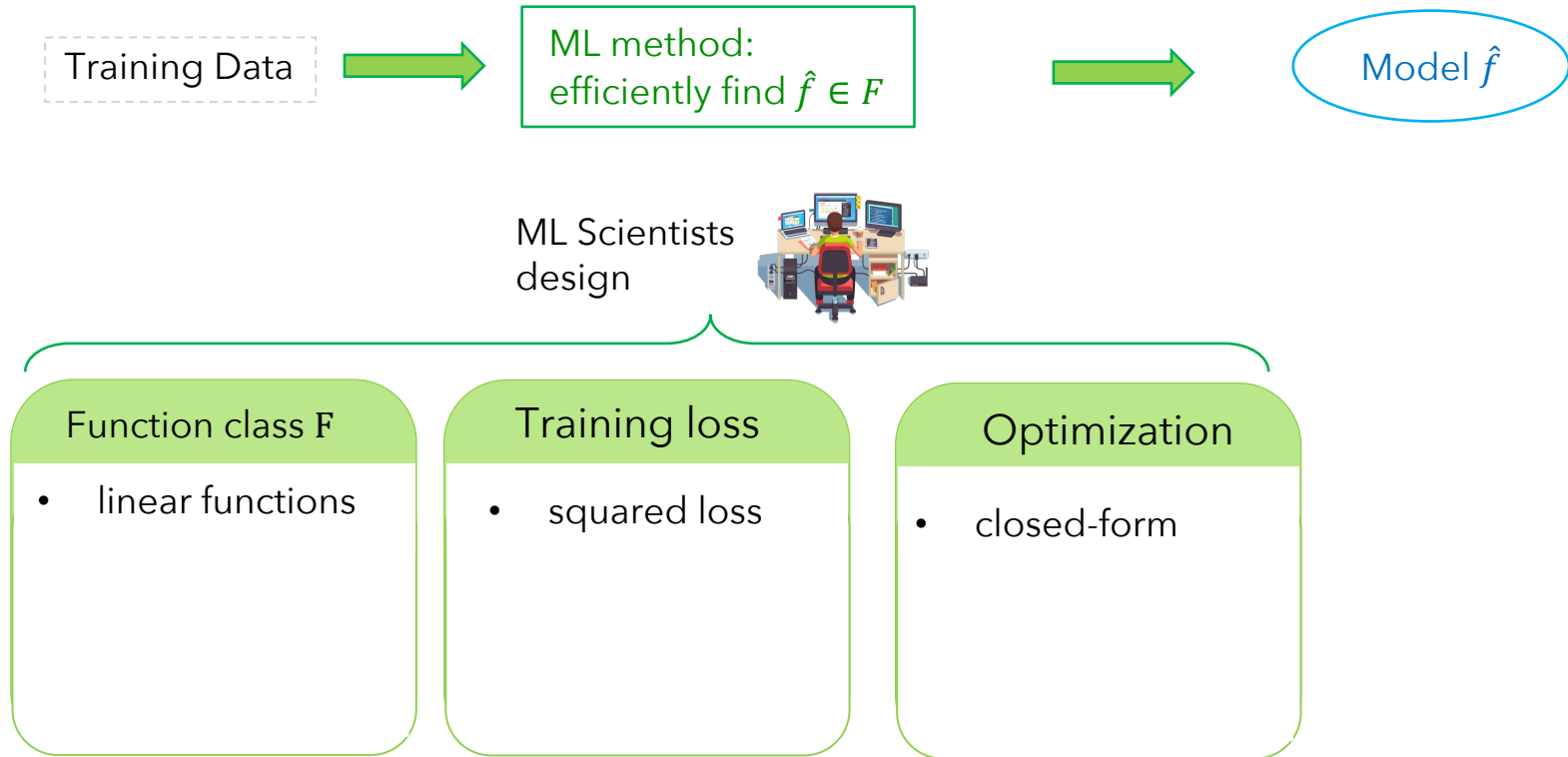
Simplified diagram of supervised learning



Simplified diagram of supervised learning



Machine learning pipeline



Example: Housing price prediction



I want to list my house for sale!
At which price should I sell it?



I am not in a rush nor am I
after maximum profit.



Find the average market
price for houses of my kind!



ML pipeline to find average house price

Last lecture I learned: I can use machine learning to do that. If I can find historical data of sold houses, I can use supervised learning. Since prices are continuous, it's a regression task!



Training Data



Can't feed houses into computer!

ML method:
efficiently find $\hat{f} \in F$
that fits training data

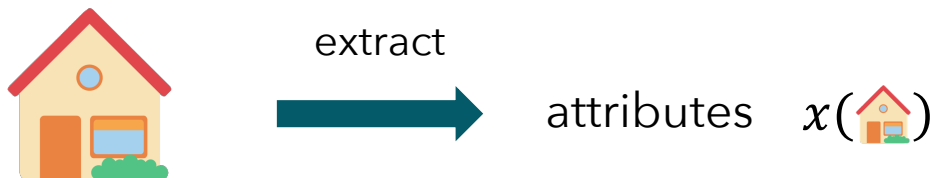


Your house

Model \hat{f}

Average price for
your house in CHF

Step 0: find numerical attributes to describe house




- Determine how to represent houses in using numerical values
- for example using **attributes** such as
size, # bathrooms, distance to public transport, years since construction ...

ML pipeline to find average house price

Step I: Collect

Training Data



House 1: attributes, price
...
House n: attributes, price

Step II: Learn

ML method:
efficiently find $\hat{f} \in F$
that fits training data

Step III: Predict

Your house: attributes

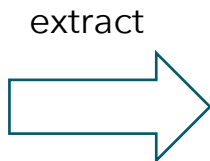
Model \hat{f}

Average price for
your house in CHF

Plan for today






- Linear regression in 1-d
 - Steps I (data collection), III (prediction with \hat{f}) in 1d
 - Step II (training method):
 - i. Class of linear functions
 - ii. Motivating the squared loss
 - iii. Finding minimizer of the loss via stationary points
 - Comment on other loss functions
- (Multiple) linear regression for general d

Step I: Collect data of other houses



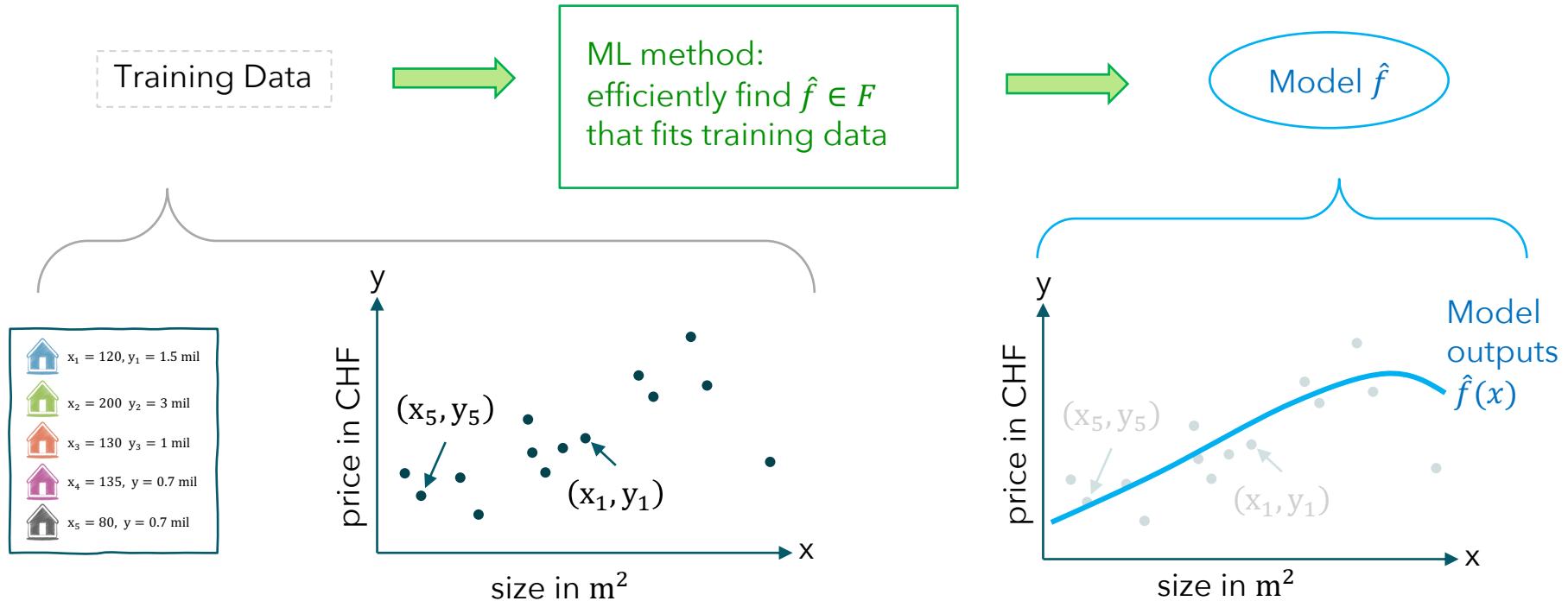
Input attribute x : size in m^2 ;

Output y : sales price in CHF

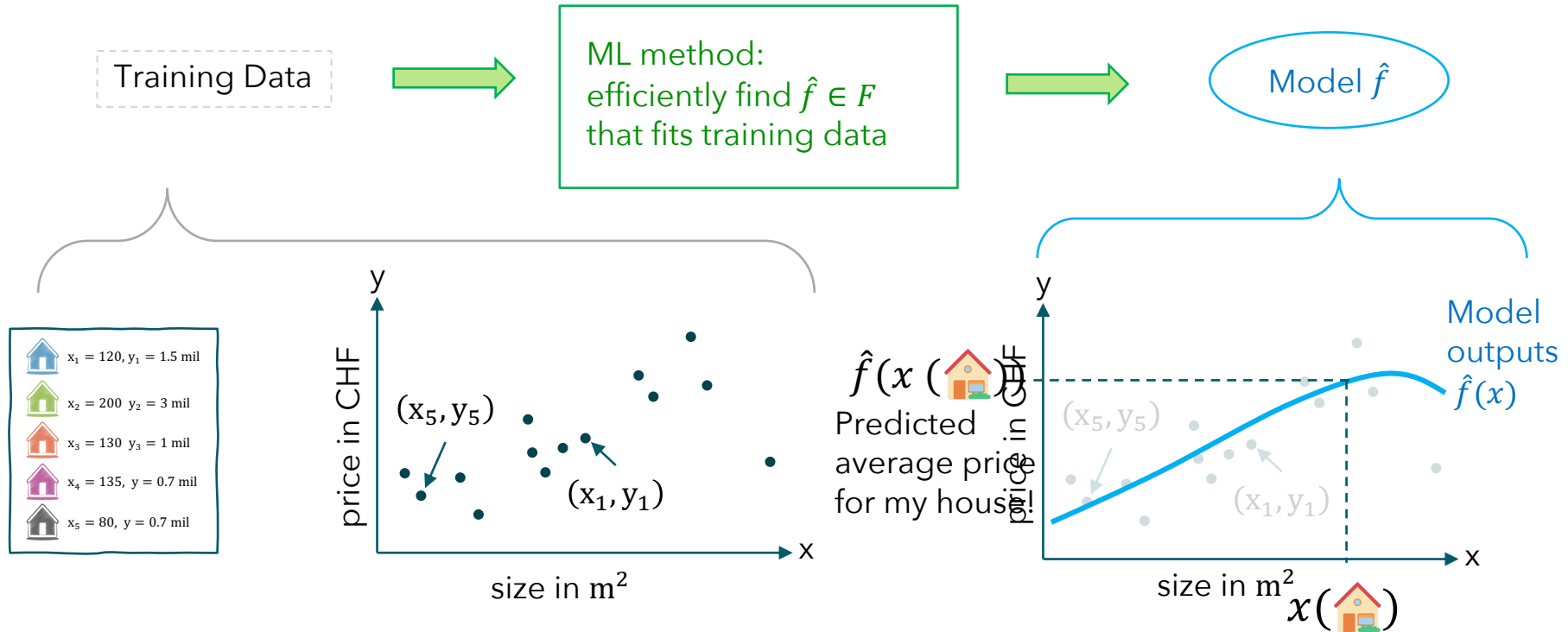
	$x_1 = 120, y_1 = 1.5 \text{ mil}$
	$x_2 = 200, y_2 = 3 \text{ mil}$
	$x_3 = 130, y_3 = 1 \text{ mil}$
	$x_4 = 135, y_4 = 0.7 \text{ mil}$
	$x_5 = 80, y_5 = 0.7 \text{ mil}$

Disclaimer: these are artificial numbers in an imaginary city

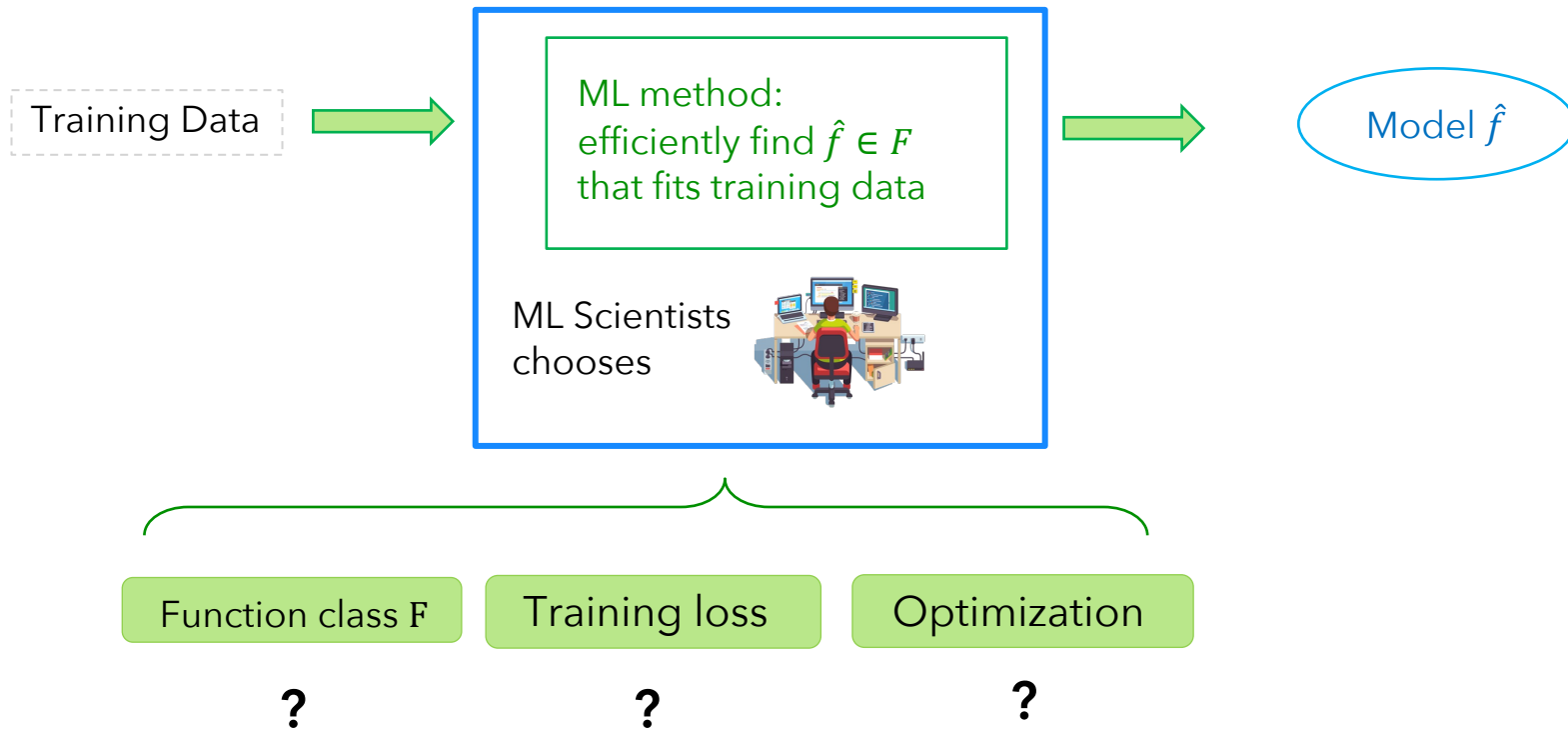
Step I → III: From training data to predicting with \hat{f}



Step I → III: From training data to predicting with \hat{f}



Step II: A method to obtain a model \hat{f}



Plan for today

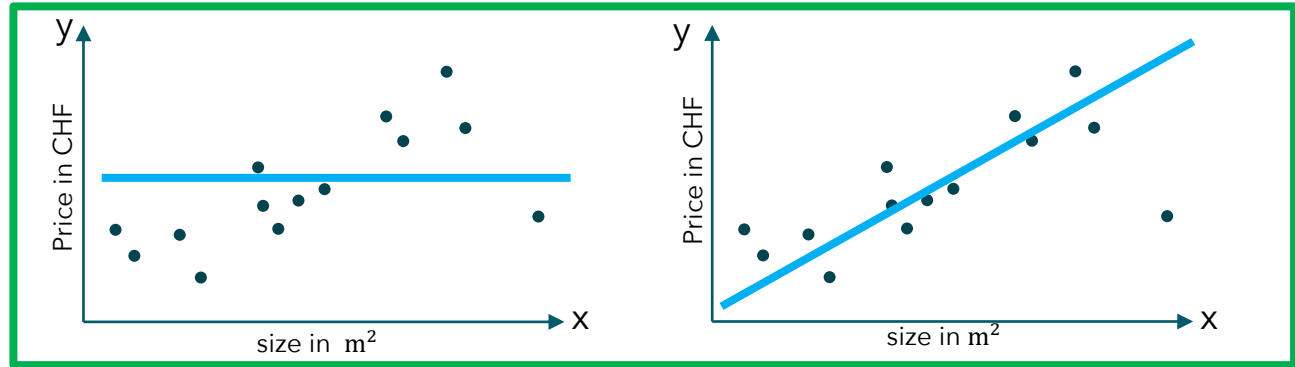
- Linear regression in 1-d
 - Steps I (data collection), III (prediction with \hat{f}) in 1d
 - Step II (training method):
 - i. Class of linear functions
 - ii. Motivating the squared loss
 - iii. Finding minimizer of the loss via stationary points
 - Comment on other loss functions
- (Multiple) linear regression for general d

Different function classes F

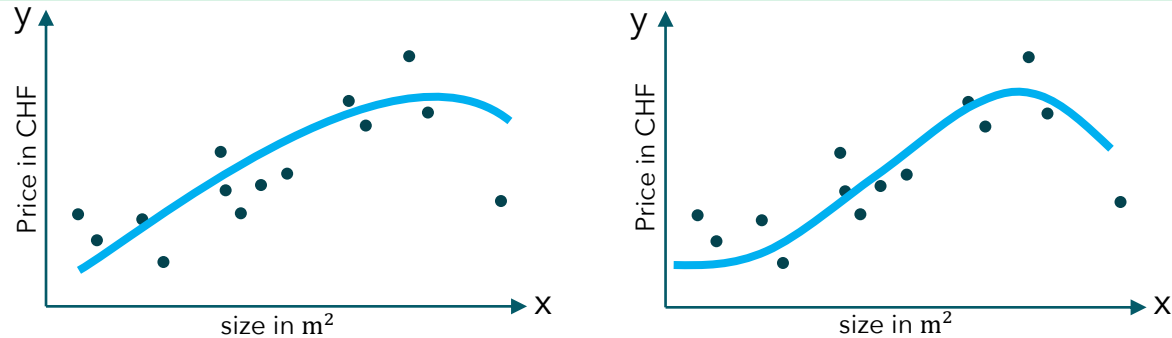
Question: With what kind of functions can we fit 1-dimensional data?

this lecture

constant
and linear



more generally
polynomial, nonlinear
(next time)



Function class: Linear functions

Function class F

- linear function

Loss function

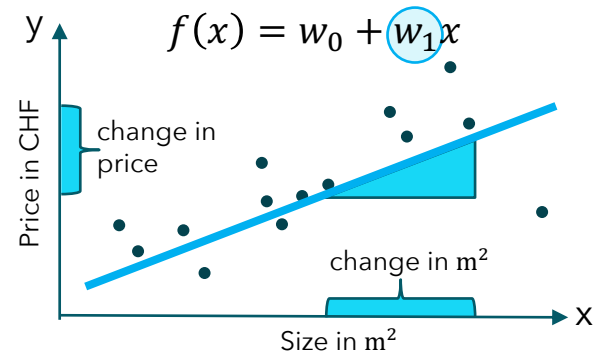
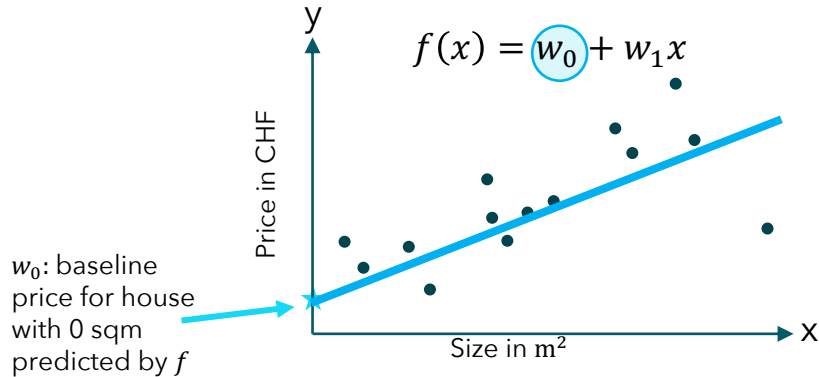
- squared loss

Optimization

- closed-form

Class/set of all linear functions is the set $F_{lin} = \{f: f(x) = w_0 + w_1x \text{ for } w_0, w_1 \in \mathbb{R}\}$

We say, all functions in F_{lin} are **parameterized** by two scalars w_0, w_1 (also called *weights*)

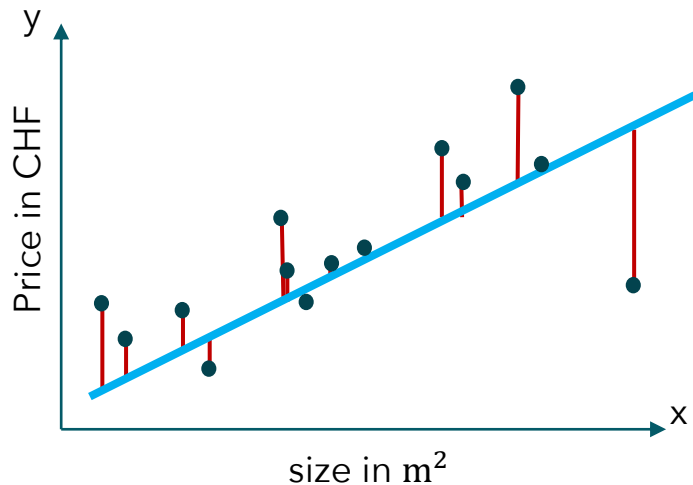


Different training losses

pointwise loss function ℓ ,
representing "closeness"
between $f(x)$ and y for a point (x, y)

Question: What is a good fit of the training data?

- When $f(x)$ is "close" to y on many points, that is, it has low training loss $L(f) := \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$
- Let's think about suitable loss functions, representing some notion of distance



Which loss $\ell(f(x), y)$ would you choose?

Discuss with neighbor for two minutes
and answer on eduApp.

(a) $|f(x) - y|$ (b) $f(x) - y$

(c) $(f(x) - y)^2$ (d) $\max(y - f(x), 0)$

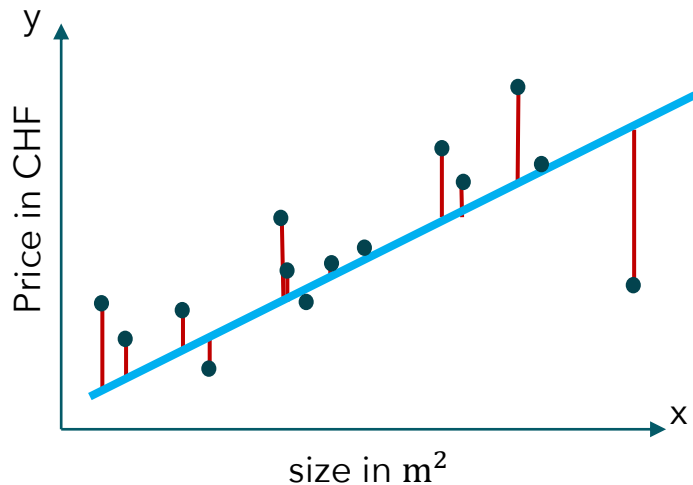


Different training losses

pointwise loss function ℓ ,
representing "closeness"
between $f(x)$ and y for a point (x, y)

Question: What is a good fit of the training data?

- When $f(x)$ is "close" to y on many points, that is, it has low training loss $L(f) := \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$
- Let's think about suitable loss functions, representing some notion of distance



not differentiable at $f(x) = y$
no closed-form

$f(x) = -10^5$ for all x
would have low loss!

(a) $|f(x) - y|$ (b) $f(x) - y$

(c) $(f(x) - y)^2$ (d) $\max(y - f(x), 0)$

$f(x) = 10^{100}$ would have 0 loss, not differentiable

Training loss using the squared loss

Function class F

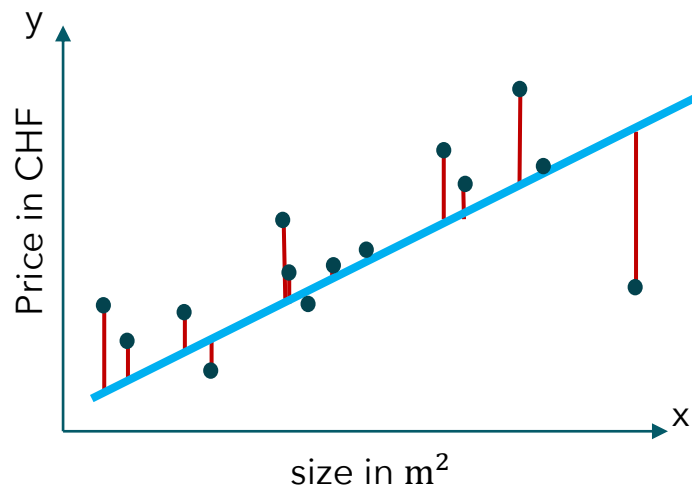
- linear function

Training loss

- squared loss

Optimization

- closed-form

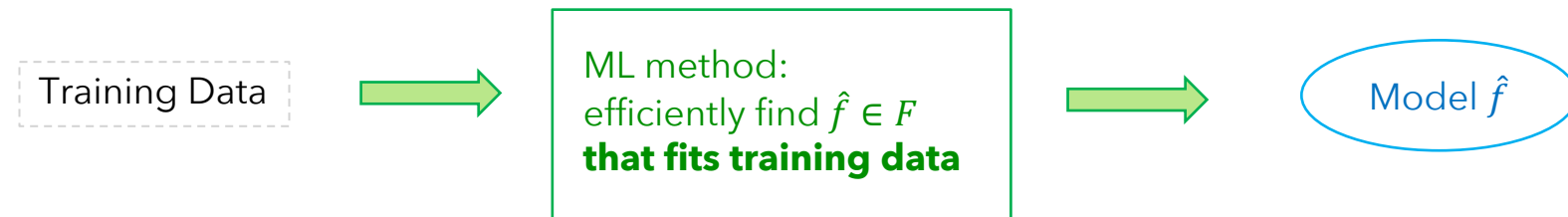


Remember: Our final “learnt rule” is the function \hat{f} that minimizes the training loss (squared loss on average):

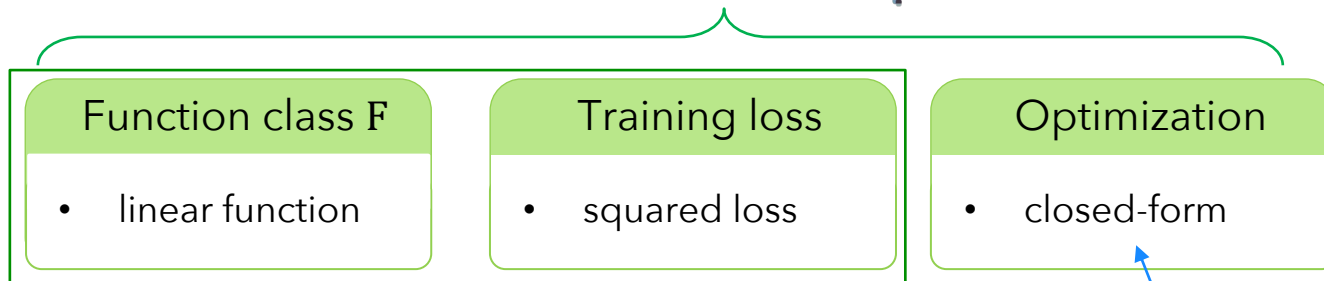
$$\hat{f} = \operatorname{argmin}_{f \in F_{lin}} L(f) = \operatorname{argmin}_{f \in F_{lin}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

We call \hat{f} the solution of the ML method **linear regression**

Step II: The method choice for this lecture



ML Scientists chooses



in short called **linear regression / linear least-squares**

today

Plan for today

- Linear regression in 1-d
 - Steps I (data collection), III (prediction with \hat{f}) in 1d
 - Step II (training method):
 - i. Class of linear functions
 - ii. Motivating the squared loss
 - iii. Finding minimizer of the loss via stationary points
 - Comment on other loss functions
- (Multiple) linear regression for general d

Linear model \hat{f} that minimizes training loss

- **Goal:** Find model \hat{f} in $F_{lin} = \{f: f(x) = w_0 + w_1x \text{ for } w_0, w_1 \in \mathbb{R}\}$ that has the smallest training loss

$$\hat{f} = \operatorname{argmin}_{f \in F_{lin}} L(f) = \operatorname{argmin}_{f \in F_{lin}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

Linear model \hat{f} that minimizes training loss

- **Goal:** Find model \hat{f} in $F_{lin} = \{f: f(x) = w_0 + w_1x \text{ for } w_0, w_1 \in \mathbb{R}\}$ that has the smallest training loss

$$\hat{f} = \operatorname{argmin}_{f \in F_{lin}} L(f) = \operatorname{argmin}_{f \in F_{lin}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

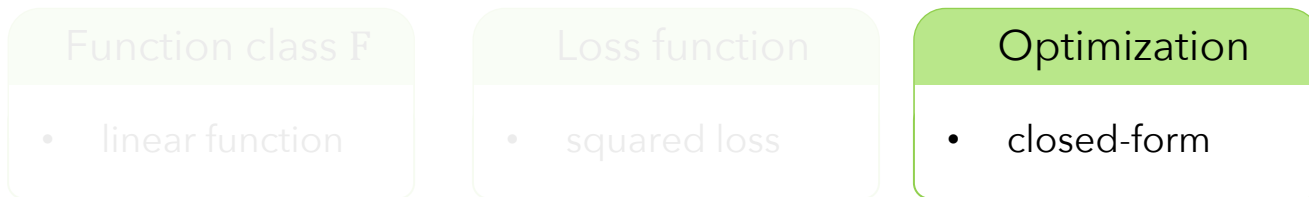
- Can write any function in F_{lin} as $f(x) = w_0 + w_1x$ for some parameters $w_0, w_1 \in \mathbb{R}$

→ searching for a minimum in F_{lin} is the same as **searching for parameters w_0, w_1** that minimize

$$\hat{w} := (\hat{w}_0, \hat{w}_1) = \operatorname{argmin}_{w_0, w_1 \in \mathbb{R}} L(w_0, w_1) = \operatorname{argmin}_{w_0, w_1 \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1x_i)^2$$

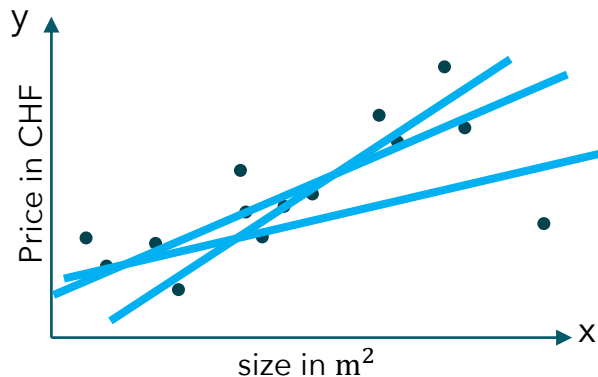
- Can then output the function $\hat{f}(x) = \hat{w}_0 + \hat{w}_1x$

Minimizing the training loss: How to find \hat{w}



Effective training loss for linear functions $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$

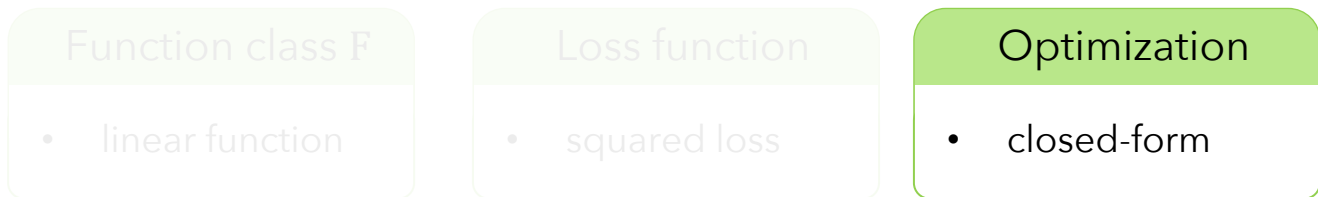
We would like to find training loss minimizer $\hat{w} = (\hat{w}_0, \hat{w}_1) = \underset{w_0, w_1 \in \mathbb{R}}{\operatorname{argmin}} L(w_0, w_1)$



- Could try out some candidates w (blue lines left: for each w we plot $f(x) = w_0 + w_1 x$)
- \mathcal{F}_{lin} includes infinitely many candidates, can't try all
- We can find minimizer \hat{w} efficiently!

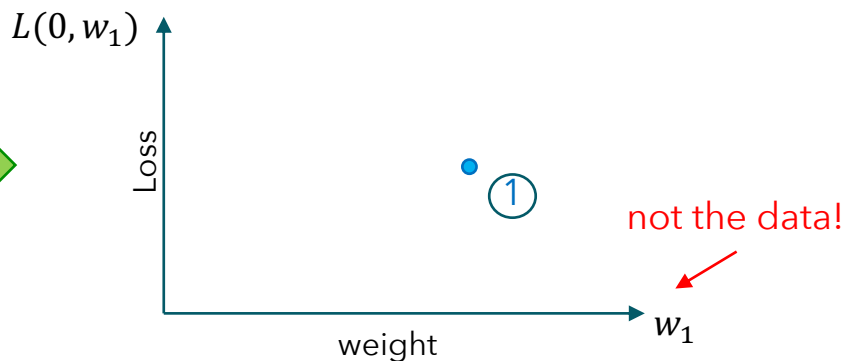
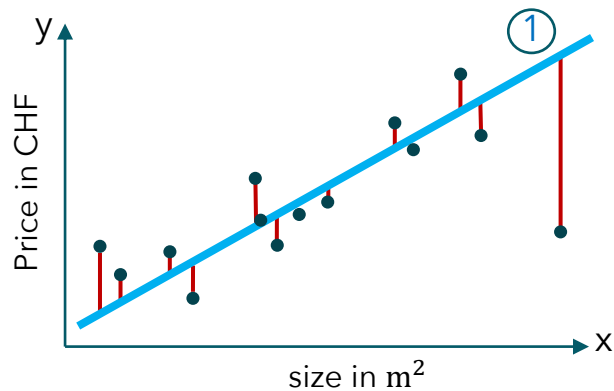
Next: characterizing the minimizer

Simplifying the problem: Fixing $w_0 = 0$



The training loss for linear functions $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$ is a function of w_0, w_1

For simplification, we first fix $w_0 = 0$, and **only optimize** $L(0, w_1)$ over $w_1 \in \mathbb{R}$

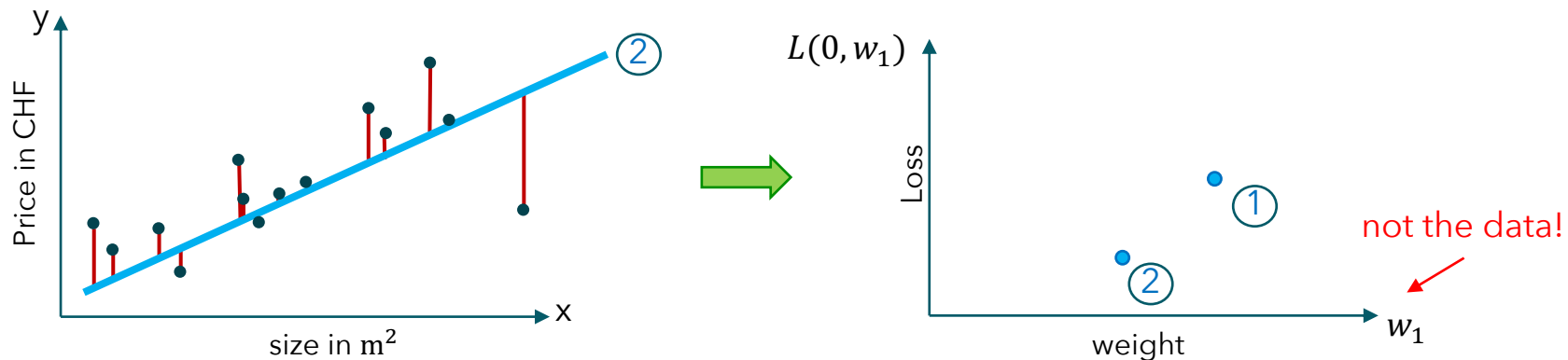


Simplifying the problem: Fixing $w_0 = 0$

Function class F	Loss function	Optimization
<ul style="list-style-type: none">linear function	<ul style="list-style-type: none">squared loss	<ul style="list-style-type: none">closed-form

The training loss for linear functions $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$ is a function of w_0, w_1

For simplification, we first fix $w_0 = 0$, and **only optimize** $L(0, w_1)$ over $w_1 \in \mathbb{R}$

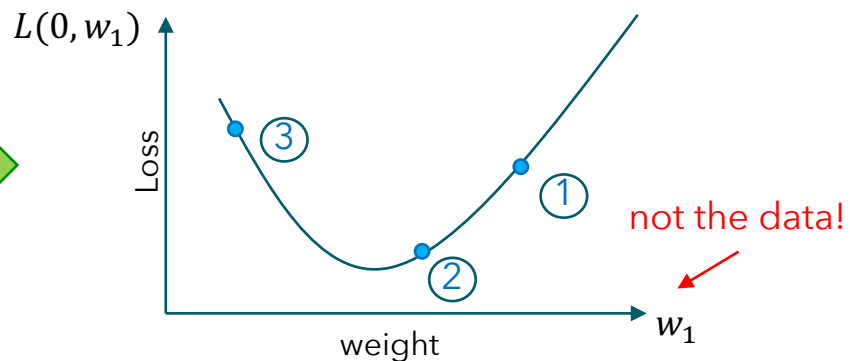
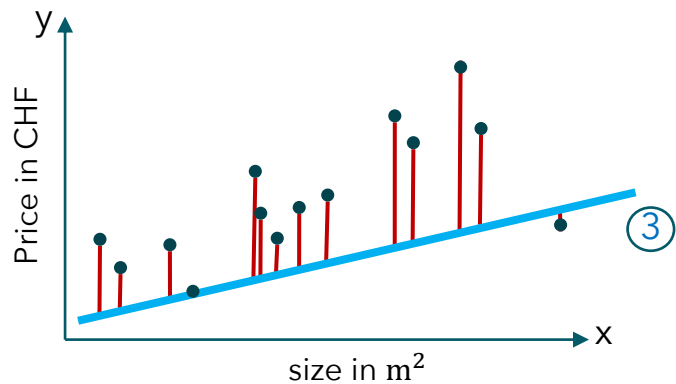


Simplifying the problem: Fixing $w_0 = 0$

Function class F	Loss function	Optimization
<ul style="list-style-type: none">linear function	<ul style="list-style-type: none">squared loss	<ul style="list-style-type: none">closed-form

The training loss for linear functions $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$ is a function of w_0, w_1

For simplification, we first fix $w_0 = 0$, and only optimize $L(0, w_1)$ over $w_1 \in \mathbb{R}$

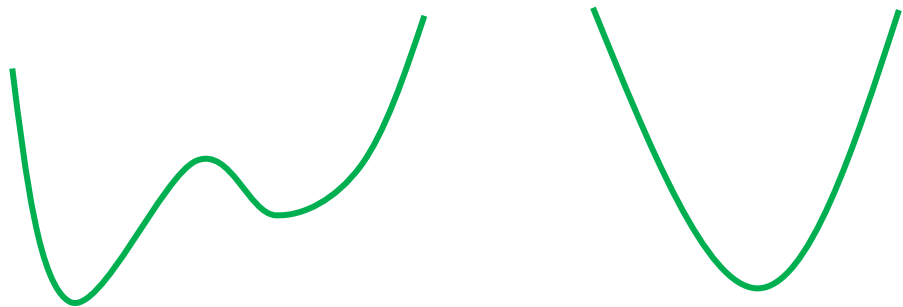


Analysis recap: Stationary points of 1-d functions

Function class F	Loss function	Optimization
<ul style="list-style-type: none">linear function	<ul style="list-style-type: none">squared loss	<ul style="list-style-type: none">closed-form

For a general 1-d function $g(w)$, where's the minimum $\hat{w} = \operatorname{argmin}_{w \in \mathbb{R}} g(w)$?

Here are two example functions. Find all stationary points, local and global minima.



derivative of $L(0, w_1)$
wrt w_1 evaluated at \hat{w}_1

→ since $L(0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_1 x_i)^2$ is a 1-d "upward" quadratic: there's one minimum where $L'(0, \hat{w}_1) = 0$
(unless all $x_i = 0$)

Back in 2-d: Finding the minimum $\hat{w} = (\hat{w}_0, \hat{w}_1)$

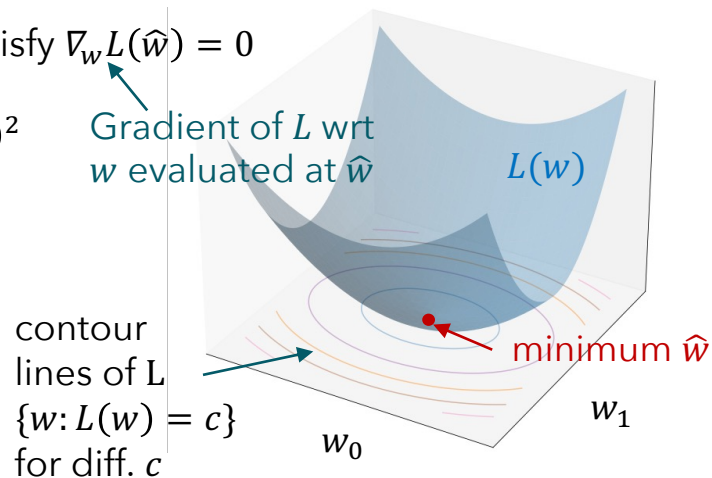
Function class F	Loss function	Optimization
<ul style="list-style-type: none">linear function	<ul style="list-style-type: none">squared loss	<ul style="list-style-type: none">closed-form

More generally: let w_0 again be variable, i.e. want to find $\hat{w} = (\hat{w}_0, \hat{w}_1) = \underset{w_0, w_1 \in \mathbb{R}}{\operatorname{argmin}} L(w_0, w_1)$

- By Theorem 2.3. Chapter I, a global minimum \hat{w} must satisfy $\nabla_w L(\hat{w}) = 0$
- Recall the training loss $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$



In the next five minutes: Turn to your neighbor and derive precise conditions on \hat{w} as a function of the sample points $\{(x_i, y_i)\}_{i=1}^n$ (calculate the gradient first)



How many minima \hat{w} ? - Linear Algebra refresher



A minimum \hat{w} must satisfy $\nabla_w L(\hat{w}) = \begin{pmatrix} -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{w}_0 - \hat{w}_1 x_i) \\ -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{w}_0 - \hat{w}_1 x_i) x_i \end{pmatrix} = 0$ (*). First think about how many stationary points there are. How many (global) minima \hat{w} does the function $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$ have?

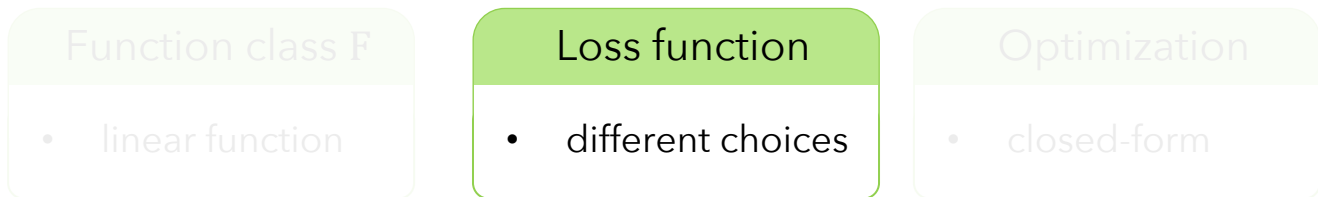
(A) One

(B) Multiple

(C) None

(D) Depends on $\{(x_i, y_i)\}_{i=1}^n$

Side comment: other losses

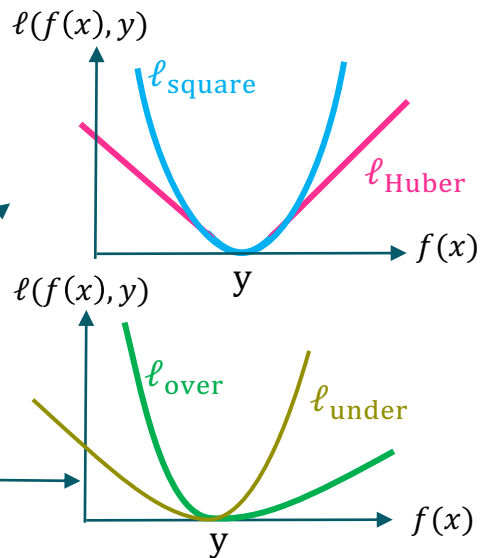


So far considered the squared loss

- weighs over- and underestimation the same
- Cost grows quadratically (large errors hugely penalized)

Instead might want it to...

- ignore outliers (ones with very large penalty) → Huber loss
- weigh over- and underestimation differently → asymmetric losses



Minimizer of Huber loss

Function class \mathcal{F}

- linear function

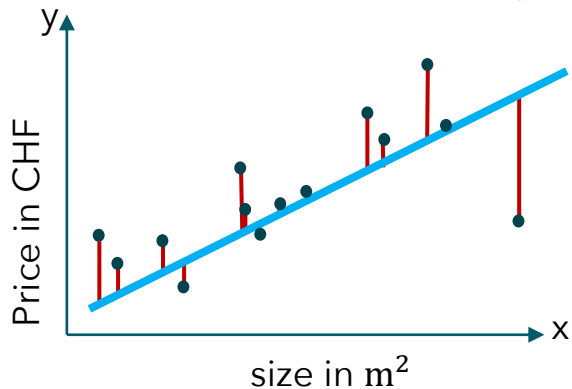
Loss function


- Huber loss

Optimization

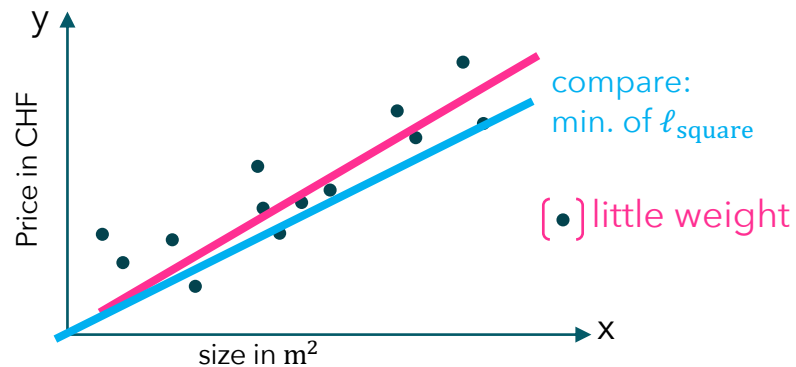
- closed-form

Minimizer of training loss with square loss ℓ_{square}



far points

 have very little weight

Minimizer of training loss Huber loss ℓ_{Huber}



Perhaps better choice if there are outliers in your training data not representing current market

Minimizer of asymmetric loss

Function class F

- linear function

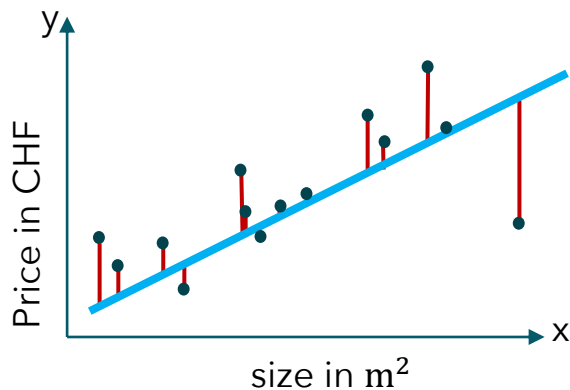
Loss function

- asymmetric regression loss

Optimization

- closed-form

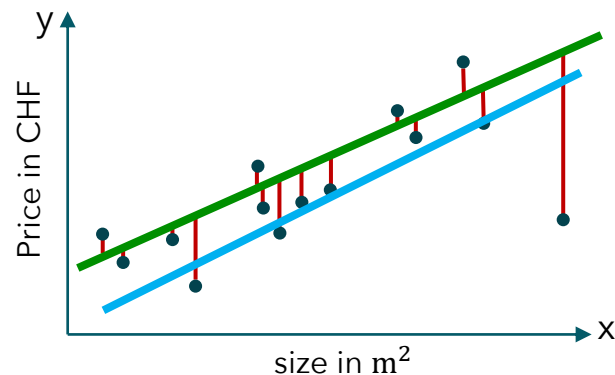
Minimum of square loss ℓ_{square}



overestimation
penalized less



Minimum of loss that places
less weight on overestimation ℓ_{over}

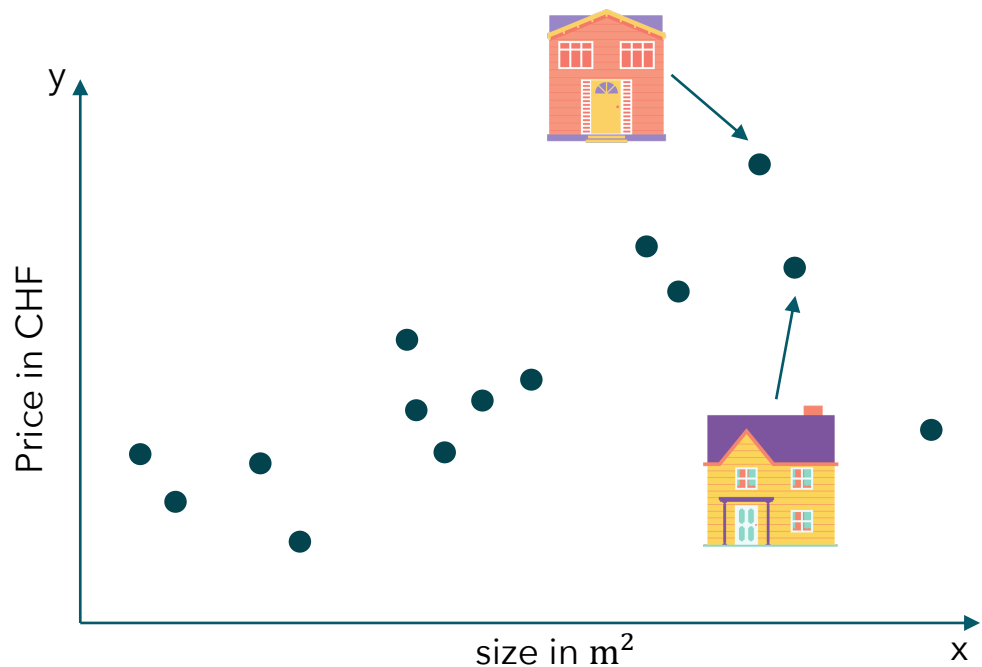


Perhaps better choice if you care more about maximizing profit than how fast you can sell

Plan for today

- Linear regression in 1-d
 - Steps I (data collection), III (prediction) in 1d
 - Step II (training method)
 - Comment on other loss functions
- (Multiple) linear regression for general d
 - Steps I (data collection), III (prediction) in multi-d
 - Step II (training method):
 - i. Class of linear functions and squared loss in multi-d
 - ii. Squared loss in standard matrix-vector notation
 - iii. Finding minimizer of the loss

Step I: More attributes available to describe house



These two houses have similar size, but very different price?

→ this 1d model completely ignores other attributes of the house!

For example, they might differ in

- number of bathrooms,
- distance to train station,
- construction year etc.

Step I: Collect more attributes of other houses

$x[1]$ = size (in m^2),
 $x[2]$ = # of bathrooms
 $x[3]$ = distance to train (in km),
 $x[4]$ = years since construction



extract
→

notation:
 x_i - input attributes of sample i
 $x[i]$ - i -th attribute
in rare cases:
 $x_j[i]$ - i -th attribute of sample j

Input: attribute vector $x = (x[1], \dots, x[d])$,

Output: y : sales price in CHF



$x_1 = (120, 2, 0.5, 1), y_1 = 1.5 \text{ mio}$



$x_2 = (200, 3, 0.5, 3), y_2 = 3 \text{ mio}$



$x_3 = (130, 2, 5, 1), y_3 = 1 \text{ mio}$



$x_4 = (135, 1, 5, 1), y_4 = 0.7 \text{ mio}$

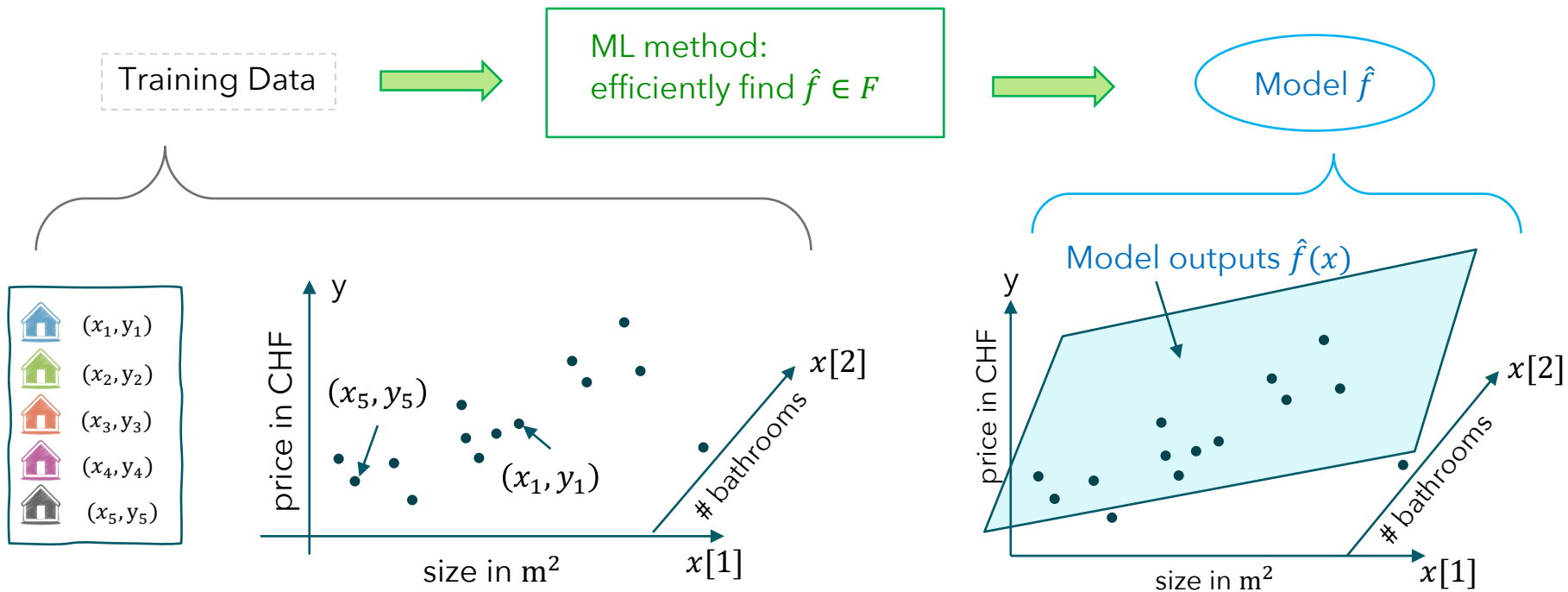


$x_5 = (80, 4, 1, 0.5), y_5 = 0.7 \text{ mio}$

Disclaimer: these are artificial numbers in an imaginary city

Step I → Step III: Multiple regression

(All visualizations are for two attributes, i.e. $d = 2$, the formulas are more general $d > 1$)



Function class: Linear with d linear features

Function class F

- linear function

Loss function

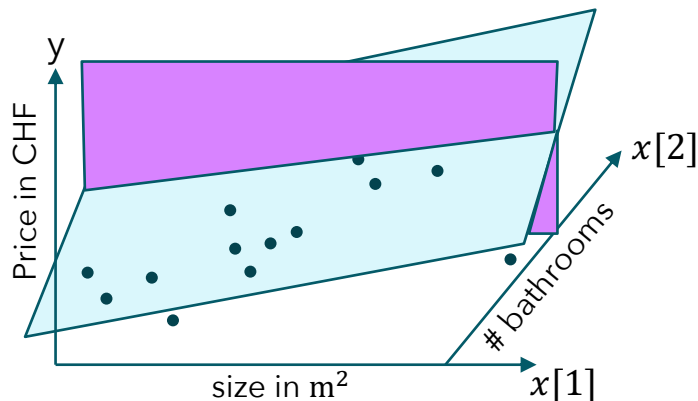
- squared loss

Optimization

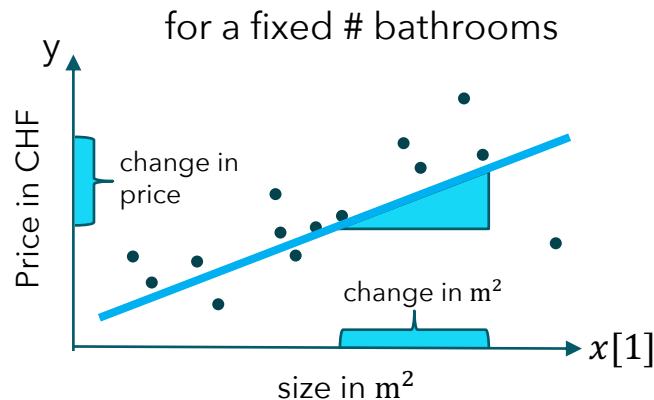
- closed-form

Class/set of all linear functions $F_{lin} = \{f: f(x) = w_0 + \sum_{j=1}^d w_j x[j] = w_0 + w^T x \text{ for } w = (w_1, \dots, w_d) \in \mathbb{R}^d\}$

Visualization for $d = 2$: $f(x) = w_0 + w_1 x[1] + w_2 x[2]$



along the
"purple cut"



Loss function: squared loss

Function class F

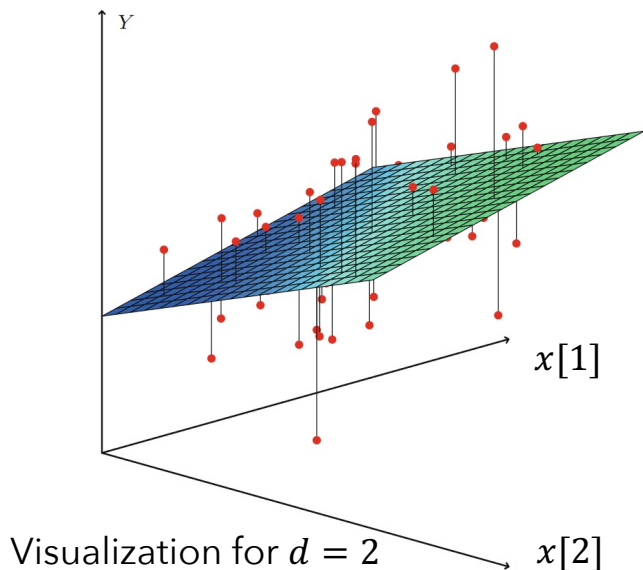
- linear function

Loss function

- squared loss

Optimization

- closed-form



Analogous to 1-d case: learned model \hat{f} minimizes training loss

$$\hat{f} = \operatorname{argmin}_{f \in F_{lin}} L(f) = \operatorname{argmin}_{f \in F_{lin}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

since $\hat{f} \in F_{lin}$ must be of the form $\hat{f}(x) = \hat{w}_0 + \hat{w}^T x$ it is equivalent to minimizing over parameter vector w and scalar w_0

$$\hat{w} = \operatorname{argmin}_{w_0 \in \mathbb{R}, w \in \mathbb{R}^d} L(w_0, w) = \operatorname{argmin}_{w_0 \in \mathbb{R}, w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (y_i - w_0 - w^T x_i)^2$$

Training loss in matrix vector notation

notation:

x_i - input attributes of sample i

$x[i]$ - i -th attribute

w_i - i -th element of vector w

We now see how we can rewrite the training loss L in matrix vector notation

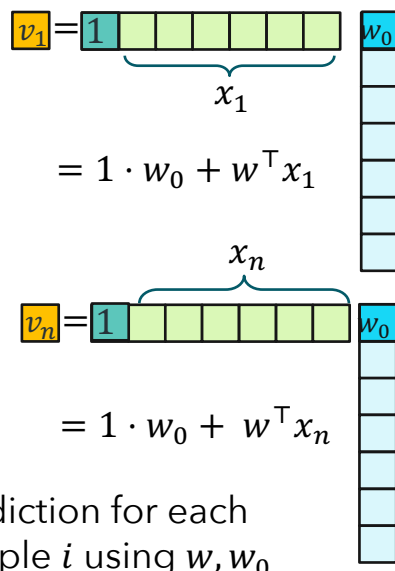
$$L(w_0, w) = \frac{1}{n} \sum_{i=1}^n (y_i - (w_0 + w^\top x_i))^2 = \frac{1}{n} \|y - (\mathbf{1}w_0 + Xw)\|^2$$

Training loss in matrix vector notation

notation:
 x_i - input attributes of sample i
 $x[i]$ - i -th attribute
 w_i - i -th element of vector w

We now see how we can rewrite the training loss L in matrix vector notation

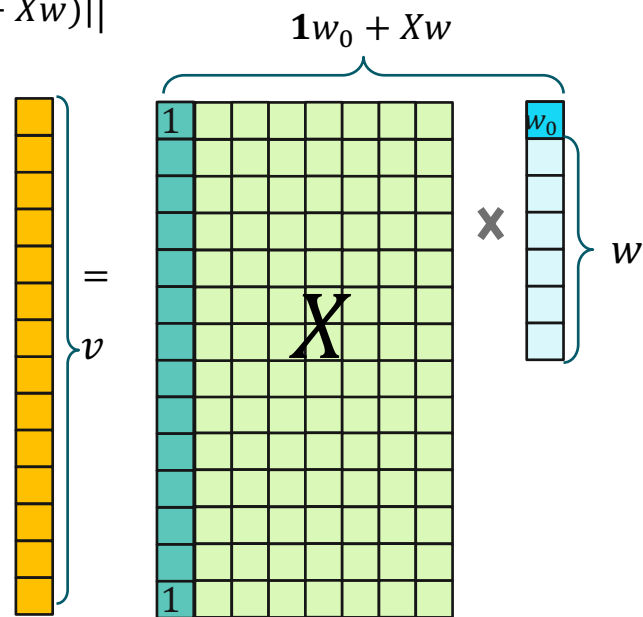
$$L(w_0, w) = \frac{1}{n} \sum_{i=1}^n (y_i - \underbrace{(w_0 + w^T x_i)}_{v_i})^2 = \frac{1}{n} \|y - (\mathbf{1}w_0 + Xw)\|^2$$



$\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^n$
 is the all-ones vector

aggregate into one vector

$$\begin{pmatrix} w_0 + w^T x_1 \\ \vdots \\ w_0 + w^T x_n \end{pmatrix}$$



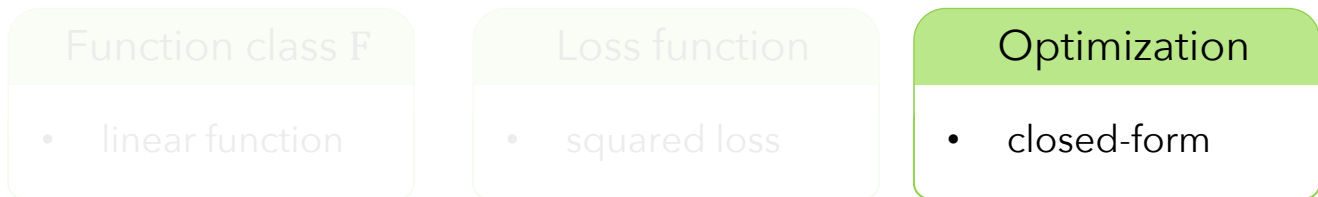
Plan for today

- Linear regression in 1-d
 - Steps I (data collection), III (prediction) in 1d
 - Step II (training method)
 - Comment on other loss functions
- (Multiple) linear regression for general d
 - Steps I (data collection), III (prediction) in multi-d
 - Step II (training method):
 - i. Class of linear functions in multi-d
 - ii. Squared loss in standard matrix-vector notation
 - iii. Finding minimizer of the loss

Minimizing the multi-d training loss

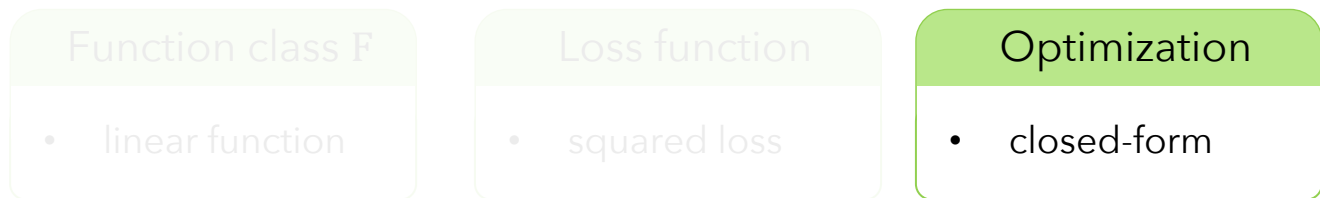
- Remember we want to find $\hat{w} = \underset{w_0 \in \mathbb{R}, w \in \mathbb{R}^d}{\operatorname{argmin}} L(w_0, w) = \underset{w_0 \in \mathbb{R}, w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \|y - (\mathbf{1}w_0 + Xw)\|^2$
- For simplicity let's again set $w_0 = 0$ and minimize over $w \rightarrow \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} L(0, w) = \frac{1}{n} \|y - Xw\|^2$.
- We'll now look at two ways to find a minimizer / solve the optimization problem:
 1. stationary point condition (find vector \hat{w} that have zero gradient)
 2. geometric argument (orthogonal projection)

Optimal solution: via stationary point condition



- Again: Minimum $\hat{w} \in \operatorname{argmin}_w L(0, w)$ must be a **stationary point**, i.e. satisfying $\nabla_w L(0, \hat{w}) = 0$
- Taking derivative of $L(w) = \frac{1}{n} \|y\|^2 - \frac{2}{n} y^\top Xw + \frac{1}{n} w^\top X^\top Xw$ yields the gradient

Optimal solution: via stationary point condition



- Again: Minimum $\hat{w} \in \operatorname{argmin}_w L(0, w)$ must be a **stationary point**, i.e. satisfying $\nabla_w L(0, \hat{w}) = 0$
- Taking derivative of $L(w) = \frac{1}{n} \|y\|^2 - \frac{2}{n} y^\top Xw + \frac{1}{n} w^\top X^\top Xw$ yields the gradient $\nabla_w L(0, w) = \frac{2}{n} (X^\top Xw - X^\top y)$ and Hessian $D^2 L(0, w) = \frac{2}{n} X^\top X$
- Therefore all minima must satisfy $X^\top y = X^\top X \hat{w}$ (normal equations)
- All stationary points of this quadratic loss $\frac{1}{n} \|y - Xw\|^2$ are minima, because the Hessian $\frac{2}{n} X^\top X$ is positive semi-definite (psd) (see Ex. 2.6. in the notes, details more next week)

Optimal solution: via geometric argument (for $d \leq n$)

Function class \mathcal{F}

- linear function

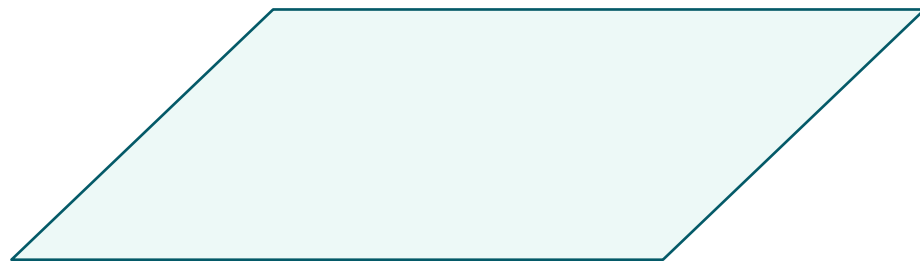
Loss function

- squared loss

Optimization

- closed-form

$$\hat{w} = \operatorname{argmin}_w \left\| \begin{matrix} n \\ y \end{matrix} - \begin{matrix} d \\ n \\ X \end{matrix} \begin{matrix} d \\ w \end{matrix} \right\|^2$$

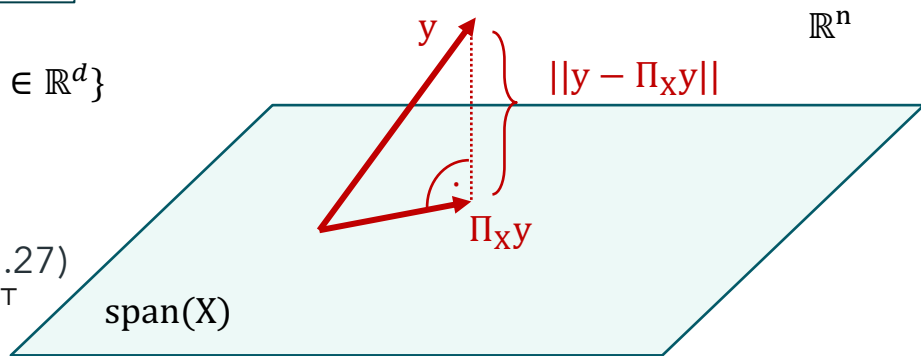


Optimal solution: via geometric argument (for $d \leq n$)

Function class F	Loss function	Optimization
<ul style="list-style-type: none"> linear function 	<ul style="list-style-type: none"> squared loss 	<ul style="list-style-type: none"> closed-form

$$\hat{w} = \operatorname{argmin}_w \left\| \begin{matrix} n \\ y \end{matrix} - \begin{matrix} n & d \\ X & \end{matrix} \begin{matrix} d \\ w \end{matrix} \right\|^2$$

- set of all possible Xw : $\operatorname{span}(X) = \{v \in \mathbb{R}^n : v = Xw, w \in \mathbb{R}^d\}$ (subspace spanned by d columns of X , Chapter I) with dimension $\leq n$
- $\Pi_X y$: the unique closest point to y on $\operatorname{span}(X)$ is the orthogonal projection of y onto $\operatorname{span}(X)$ (Ch. I, Ex. 1.27)
- In Chapter I, Def 1.24 we see that $\Pi_X = X(X^T X)^{-1} X^T$



Optimal solution: via geometric argument

Note that by the definition of the projection, there's always \hat{w} such that $X\hat{w} = \Pi_X y$ (no matter which X)

Which \hat{w} satisfy the condition $X\hat{w} = \Pi_X y$? (see math recap Exercise 1.26)

- We know that because $X\hat{w}$ is an orthogonal projection onto $\text{span}(X) = \{Xw : w \in R^d\}$,
the following has to be satisfied: $y - X\hat{w} \perp Xw$ for all $w \rightarrow \underbrace{(y - X\hat{w})^T X}_{\text{let's call it } u^T} w = \langle u, w \rangle = 0$ for all $w \in R^d$
- In general: a vector that is orthogonal to all vectors, can only be the zero vector!
that is, we require $u = X^T y - X^T X \hat{w} = 0$ and hence $X^T y = X^T X \hat{w}$

Optimal solution

- We saw two different ways to derive conditions $X^T y = X^T X \hat{w}$

on the minima of the loss $\frac{1}{n} \|y - Xw\|^2$ (called normal equations!)

- If $X^T X$ is invertible, this yields the unique solution $\hat{w} = (X^T X)^{-1} X^T y$

- Exercise: How to find minimizer when $w_0 \neq 0$? Hint: define new matrix $\tilde{X} = (\mathbf{1} \ X)$

- Think about the following (answer in Sec. 4.2.3.): In general ...



How many minima \hat{w} does $\frac{1}{n} \|y - Xw\|^2$ have? What is the minimum value?

How do these answers depend on n, d ? Argue using the matrix $X^T X$

What you can do now

- high-level: teach a machine how to use training data to output a prediction rule/model (that can be used for prediction of a new point)
- know what a training loss is
- linear regression with multiple attributes including
 - deriving a closed-form solution for the best linear fit (in the square loss sense) on the training data (training loss minimizer)

References

- Lecture notes Chapters I, Chapter II.4
- Other reference: ISLR Chapter 3

