DINFK

Linear regression

Introduction to Machine Learning, Lecture 2

Fanny Yang

ETH zürich

Announcements

- Deadline for buddy matching: this **Friday 25.2. 23:59**
  - For today, the virtual attendees may take the time to brainstorm on their own
- Q&A session online **today 17:15 to 18:00**
  - may ask more administrative questions there
  - instructions on how to effectively use Moodle
  - find the link on the course website

Recap and plan for today

Last lecture: Different problems in machine learning

- supervised learning
- unsupervised learning

Today: Walking through the supervised learning pipeline with

- the simplest ML method: linear regression in 1d
- for the scenario of determining sales price of a house

Simplified diagram of supervised learning

Training Data

(x_1, y_1)
$\vdots$
(x_n, y_n)

attributes of patient

supervision: observed labels

ML method:
efficiently find rules that fit training data

ML scientists design

Some test input $\boldsymbol{x}$

Learnt Rules

Prediction $\boldsymbol{y}$

Simplified diagram of supervised learning

Training Data

(x_1, y_1)
$\vdots$
(x_n, y_n)

attributes of patient

supervision: observed labels

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

ML scientists design

Some test input x

Model $\hat{f}$

Prediction $\hat{f}(x)$

Machine learning pipeline

Training Data → ML method: efficiently find $\hat{f} \in F$ → Model $\hat{f}$

ML Scientists design

Function class $\mathbf{F}$

- linear functions

Training loss

- squared loss

Optimization

- closed-form

Example (house)

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 input-output examples, I can use supervised learning, it's a regression task!

Can't feed houses into computer!

Your house

Training Data

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

Model $\hat{f}$

Average price for your house in CHF

Step 0: find representation for house

- Determine how to represent houses in a digital fashion
- for example using a vector of attributes:
  e.g. size, # bathrooms, distance to public transport, years since construction, …
- Collect attributes and sales prices from other houses and your own

extract

House 1: attributes, price

…

House n: attributes, price

Your house: attributes

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

Simple linear regression in 1-d

(example with one attribute)

Step I: Collect data of other houses

extract

Input attribute $\boldsymbol{x}$: size in $\boldsymbol{m^2}$;

Output $\boldsymbol{y}$: sales price in CHF

$x_1 = 120, y_1 = 1.5\ mil$

$x_2 = 200 \ \ y_2 = 3\ mil$

$x_3 = 130 \ \ y_3 = 1\ mil$

$x_4 = 135, \ y_4 = 0.7\ mil$

$x_5 = 80, \ y_5 = 0.7\ mil$

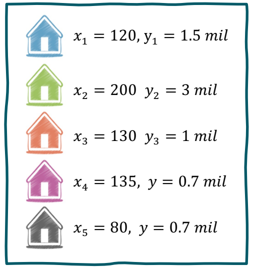
Disclaimer: these are artificial numbers in an imaginary city

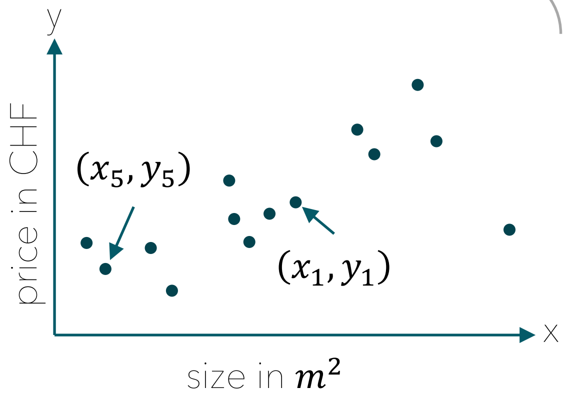
From training data to model $\hat{f}$

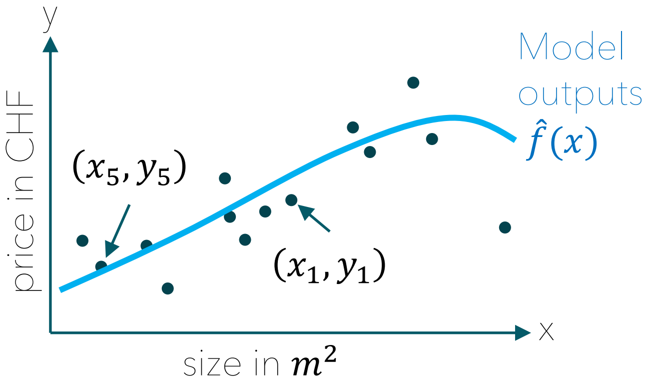
Training Data

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

Model $\hat{f}$







Step III: Prediction if we have $\hat{f}$

Training Data

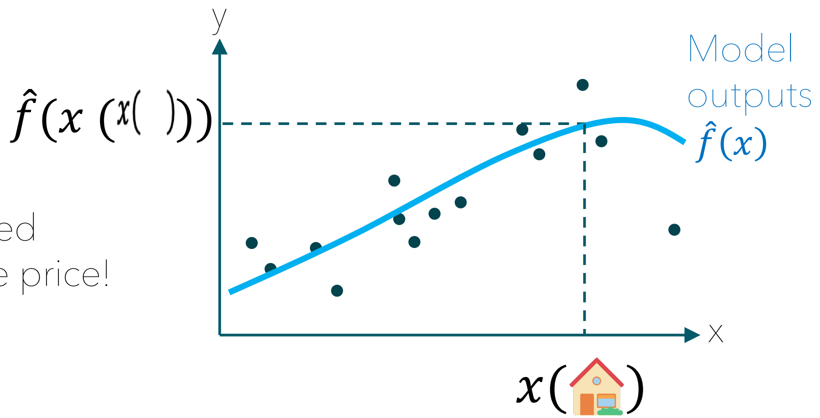
ML method:
efficiently find $\hat{f} \in F$
that fits training data

Your house: attributes

Model $\hat{f}$

Average price for
your house

Predicted
average price!



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

Training Data

ML method:
efficiently find $\hat{f} \in F$

Model $\hat{f}$

ML Scientists
chooses

Function class $\mathbf{F}$

Training loss

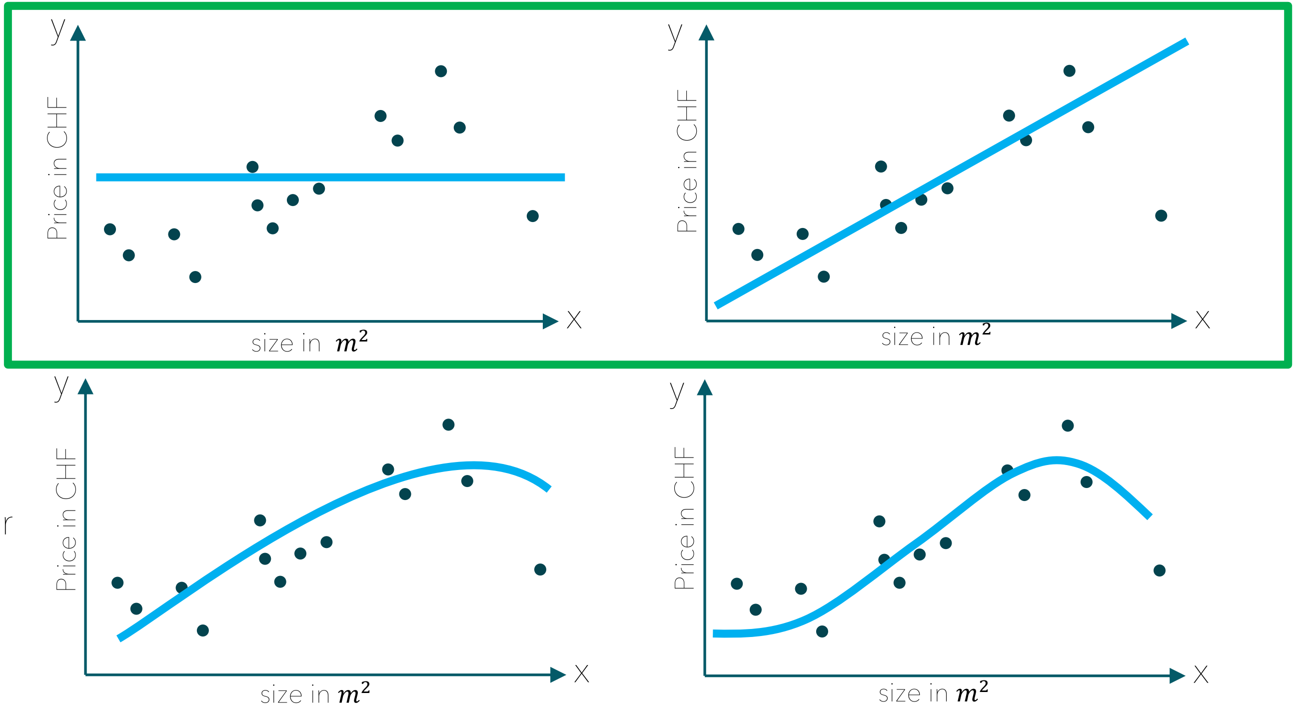
Optimization

Different function classes $\boldsymbol{F}$

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

constant
and linear

more generally
polynomial, nonlinear
(next time)



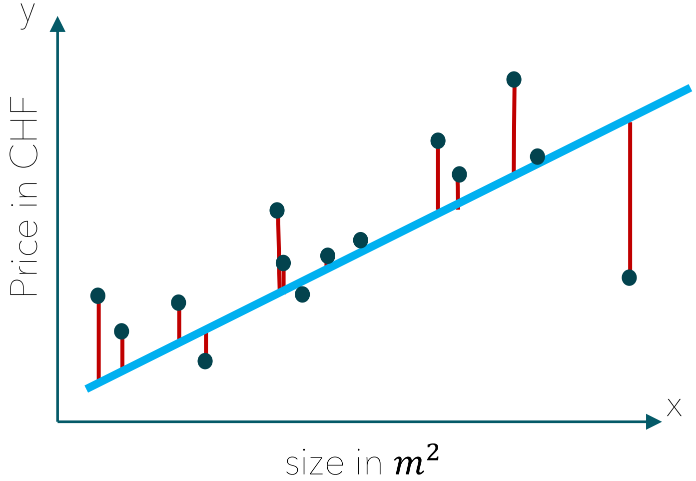
Different training losses

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?

If $f(x)$ is "close" to y for most 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) $max(y - f(x), 0)$

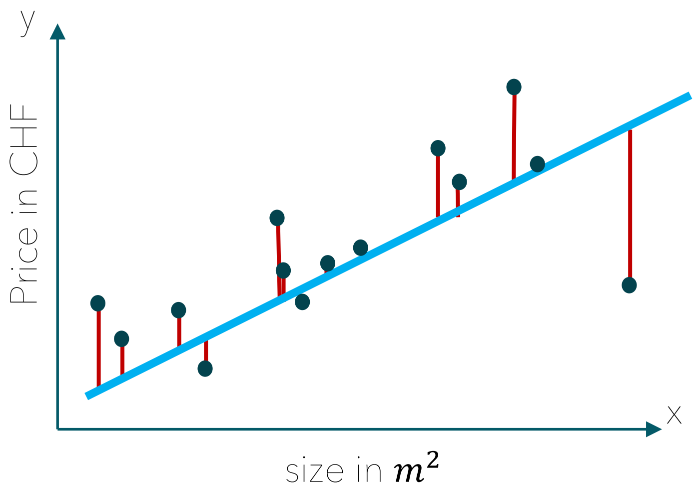
Different training losses

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?

If $f(x)$ is "close" to y for most 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

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

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

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

overestimation is rewarded, not differentiable

Step II: The method choice for this lecture

Training Data

ML method:
efficiently find $\hat{f} \in F$

Model $\hat{f}$

ML Scientists chooses

Function class F	Training loss	Optimization
• linear function	• squared loss	• closed-form

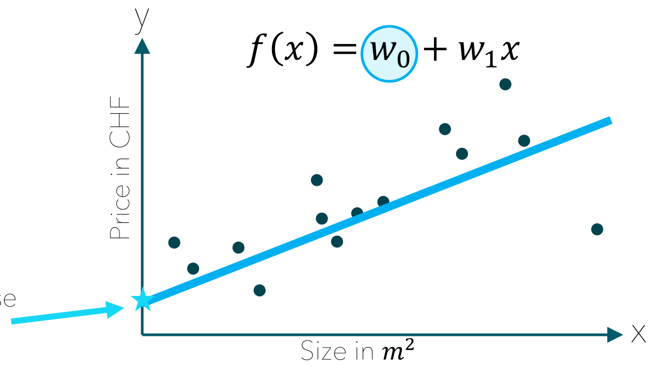
Function class: Linear functions

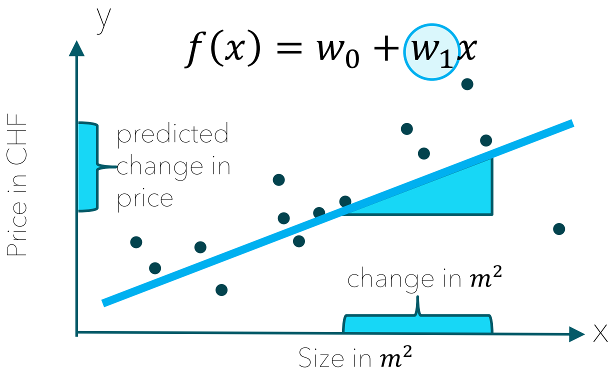
Function class $\boldsymbol{F}$	Loss function	Optimization
• linear function	• squared loss	• closed-form

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

We say, all functions in $\boldsymbol{F_{lin}}$ are *parameterized* by two scalars $\boldsymbol{w_0}$, $\boldsymbol{w_1}$

predicted
price for house
with 0 sqm





Training loss using the squared loss

Function class F

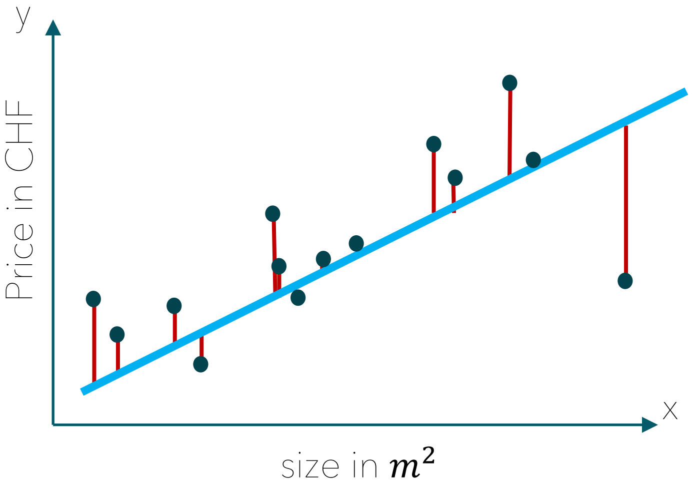
- linear function

Training loss

- squared loss

Optimization

- closed-form



What is a good fit of the training data?

We want to find the function $\hat{f}$ that minimizes the training loss (squared loss) on average over the training set:

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

$\hat{f}$ we call the output of our ML model – linear regression

Linear model $\hat{f}$ that minimizes training loss

- Goal: Find model $\hat{f}$ that has the smallest training loss

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

- Recall: all linear functions in $F_{lin} = \{ f : f(x) = w_0 + w_1 x \ \text{ for } w_0, w_1 \in \mathbb{R}\}$

  are parameterized by two scalars $\boldsymbol{w_0, w_1}$

- Since $\hat{f}$ is a function in F_{lin}, it has to be of the form $\hat{f}(x) = \widehat{w}_0 + \widehat{w}_1 x$ for two scalars w_0, w_1,

  → searching for a minimum in F_{lin} is the same as searching for scalars $\mathbf{w_0, w_1}$ that minimize

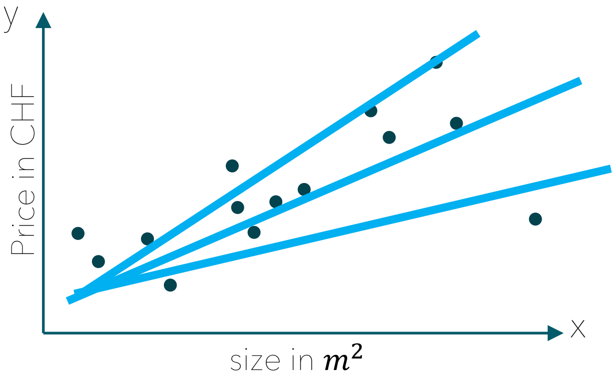
$$\widehat{w} \coloneqq (\widehat{w}_0, \widehat{w}_1) = \underset{w_0, w_1 \in \mathbb{R}}{\operatorname{argmin}} \ L(w_0, w_1) = \underset{w_0, w_1 \in \mathbb{R}}{\operatorname{argmin}} \ \frac{1}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2$$

Minimization of the training loss: How to find $\hat{w}$

Function class F	Loss function	Optimization
• linear function	• squared loss	• closed-form

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 $\hat{w} = (\hat{w}_0, \hat{w}_1) = \underset{w_0, w_1 \in \mathbb{R}}{\operatorname{argmin}} L(w_0, w_1)$



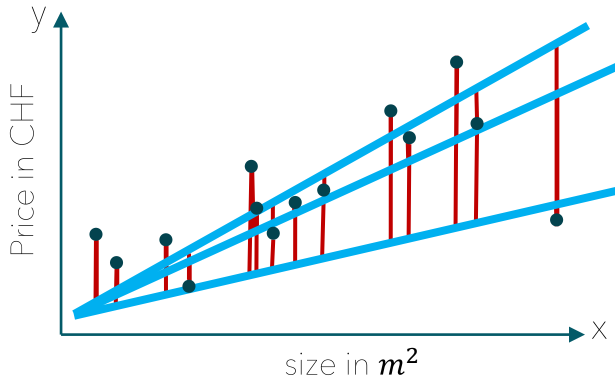
- Some candidate linear functions (left), but F_{lin} includes infinitely many candidates!
- We can compute minimizer w_0, w_1 analytically!

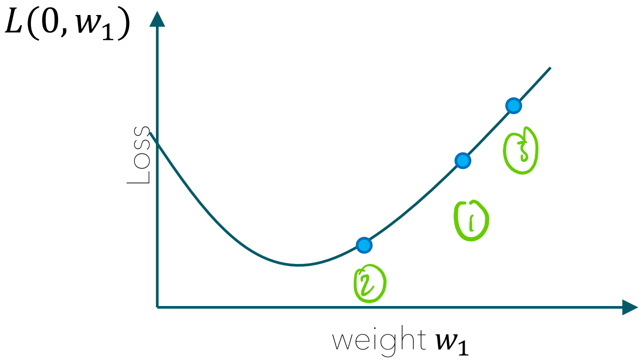
Simplifying the problem: Fixing $w_0 = 0$

Function class F	Loss function	Optimization
• linear function	• squared loss	• 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 $w_1 \in \mathbb{R}$



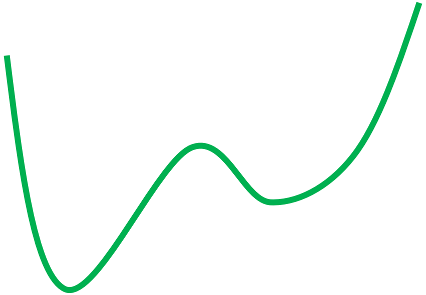


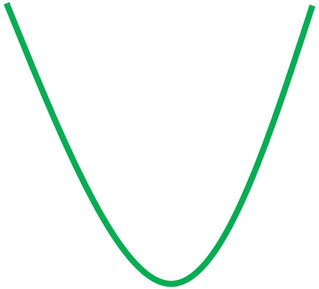
Analysis recap: Stationary points of 1-d functions

Function class F	Loss function	Optimization
• linear function	• squared loss	• closed-form

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

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





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

→ since $\mathrm{L}(0, \mathrm{w}_1)$ is a one-dimensional quadratic: there's one minimum where $\boxed{\mathrm{L}'(0, \widehat{\mathrm{w}}_1) = 0}$

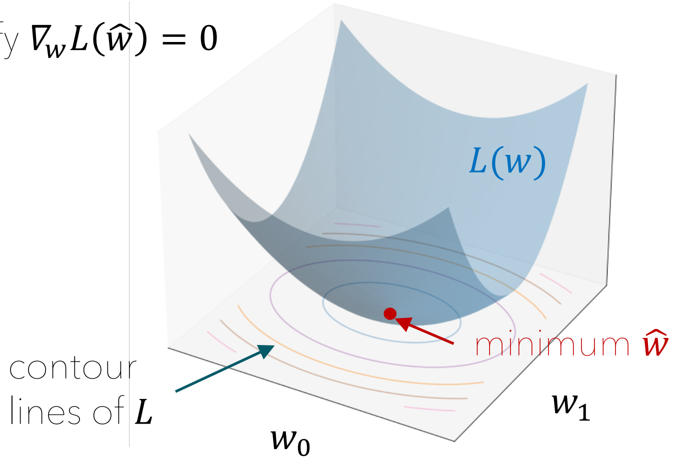
Solving the original problem: Finding the minimum $\hat{w}$

Function class F	Loss function	Optimization
• linear function	• squared loss	• 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. math recap, 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)



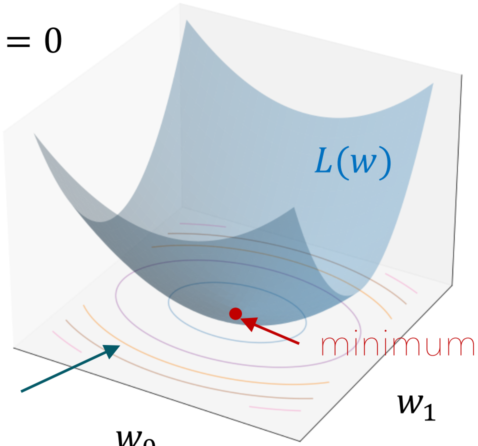
Solving the original problem: Finding the minimum $\hat{w}$

Function class F	Loss function	Optimization
• linear function	• squared loss	• 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. math recap, a global minimum $\hat{w}$ must satisfy $\nabla_w L(\hat{w}) = 0$
- Recall the training loss $L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2$
- And hence the minimum $\hat{w} = (\hat{w}_0, \hat{w}_1)$ satisfies

$$\nabla_w L(\hat{w}_0, \hat{w}_1) = \begin{pmatrix} -\frac{2}{n} \sum_{i=1}^{n} (y_i - \hat{w}_0 - \hat{w}_1 x_i) \\ -\frac{2}{n} \sum_{i=1}^{n} (y_i - \hat{w}_0 - \hat{w}_1 x_i)\, x_i \end{pmatrix} = 0$$



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

A minimum $\hat{w}$ must satisfy $\begin{pmatrix} -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{w}_0 - \hat{w}_1 x_i) \\ -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{w}_0 - \hat{w}_1 x_i)\, x_i \end{pmatrix} = \mathbf{0}$. First think about how many solutions exist to this equation. How many (global) minima $\hat{w}$ exist?

(A) One

(B) Multiple

(C) None

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

Solution: (D) Regarding number of solutions:

- *brute force: deriving closed-form solutions (see homework)*
- *or: matrix vector notation (we see later today)*
- *or: system of two linear equations for two parameters → when does it have a unique solution?*

Further, all solutions are global minima, we'll discuss next week in more detail, why …

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) by deriving a closed-form solution
- know what a training loss is and minimize the training loss (with square loss) over 1-d linear functions by minimizing over the scalars that parameterize the function

Side comment: other losses

MORE IN TUTORIAL...

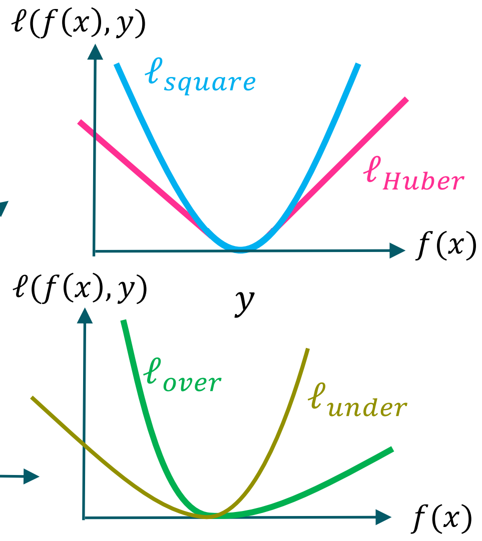
Function class F	Loss function	Optimization
• linear function	• different choices	• closed-form

Squared loss

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

Instead might want:

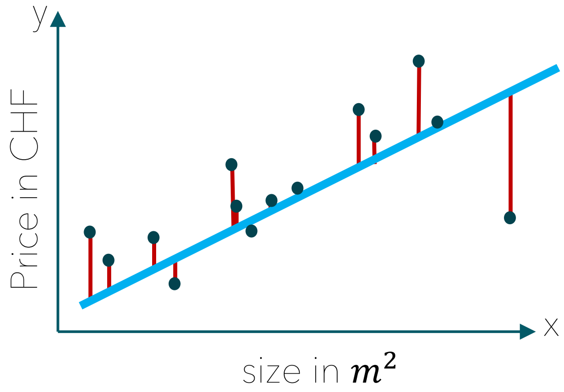
- ignore outliers (ones with very large penalty) → Huber loss
- weigh over- and underestimation differently → asymmetric losses



Minimizer of Huber loss

Function class F	Loss function	Optimization
• linear function	• Huber loss	• closed-form

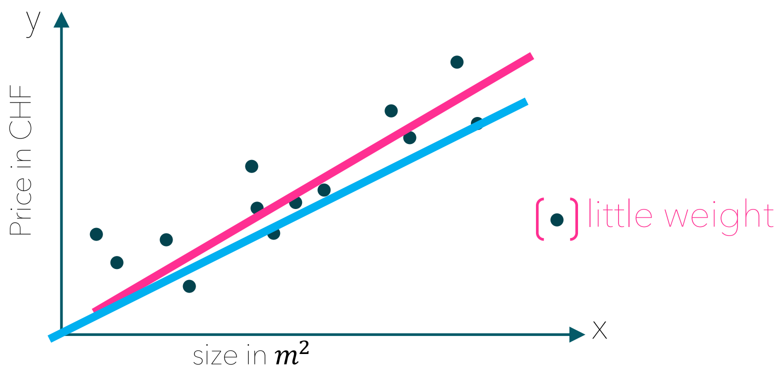
Minimizer of training loss with square loss ℓ_{square}



far points

have very little weight

Minimizer of training loss Huber loss ℓ_{Huber}

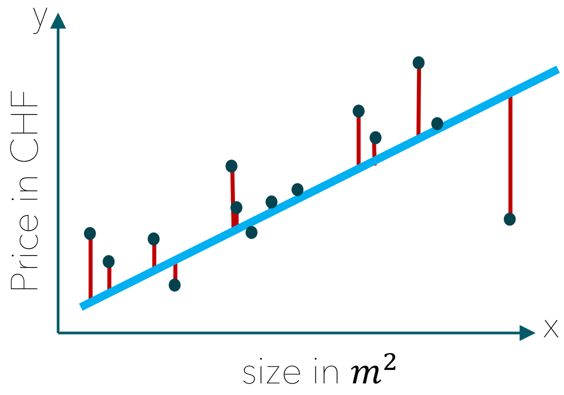


Perhaps better choice if there are outliers in your training data not representing current market

Minimizer of asymmetric loss

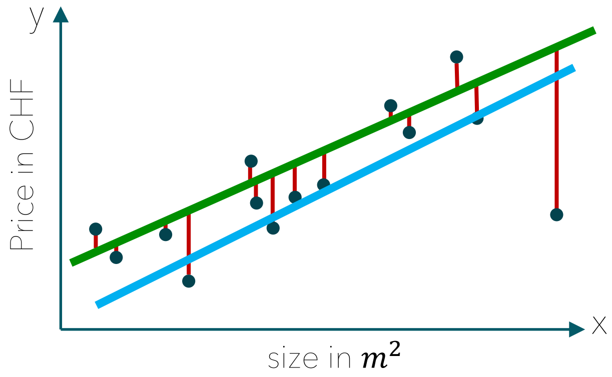
Function class F	Loss function	Optimization
• linear function	• asymmetric regression loss	• closed-form

Minimum of square loss ℓ_{square}



More weight on overestimation

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

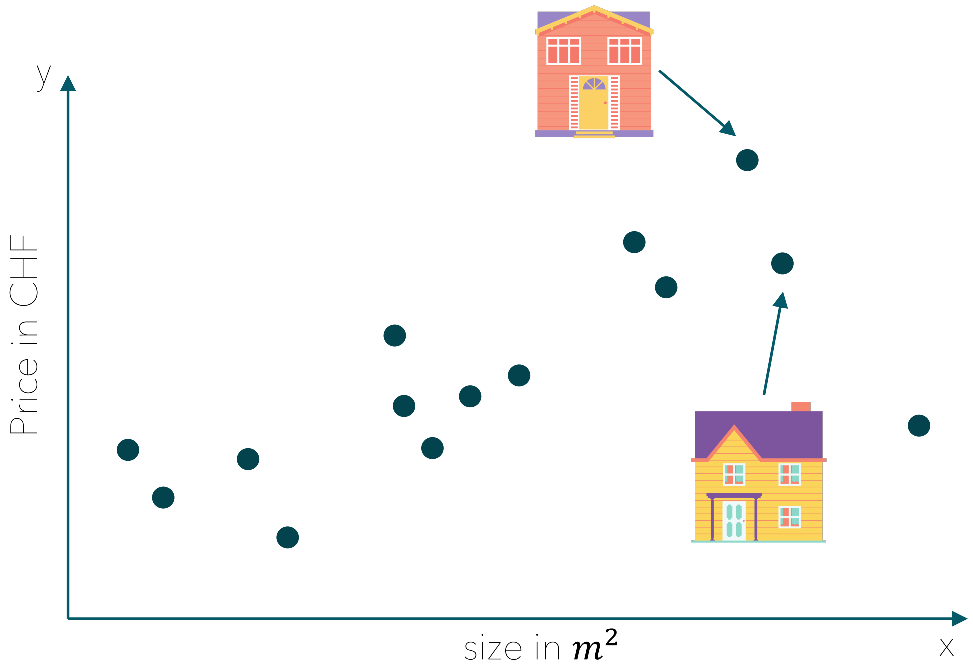


Perhaps better choice if you care more about maximizing profit than how fast you can sell

Multiple regression

(Linear least squares)

More inputs available



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 data 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

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

Output: y: sales price in CHF

extract

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

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

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

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

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

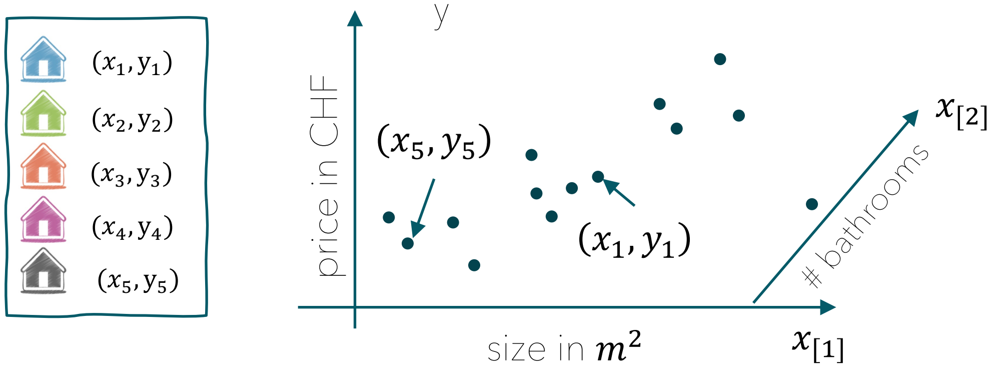
notation:
x_i - input attributes of sample i
$x_{[i]}$ - i-th attribute

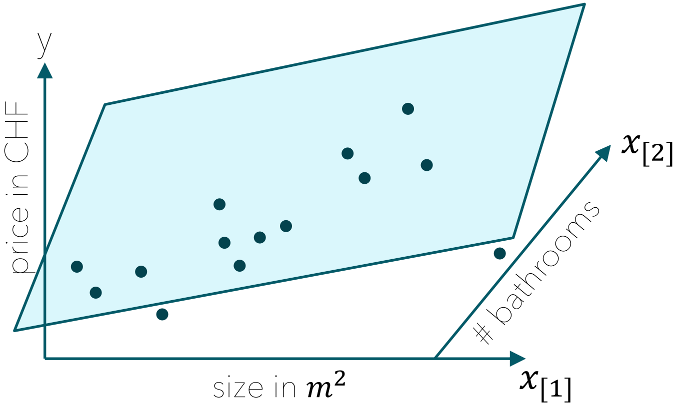
Disclaimer: these are artificial numbers in an imaginary city

Multiple regression

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

Training Data → ML method: efficiently find $\hat{f} \in F$ → Model $\hat{f}$





Function class: Linear with $\boldsymbol{d}$ linear features

Function class $\boldsymbol{F}$

- linear function

Loss function

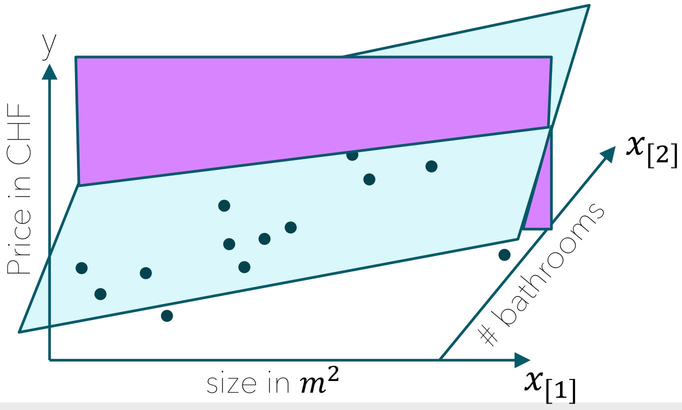
- squared loss

Optimization

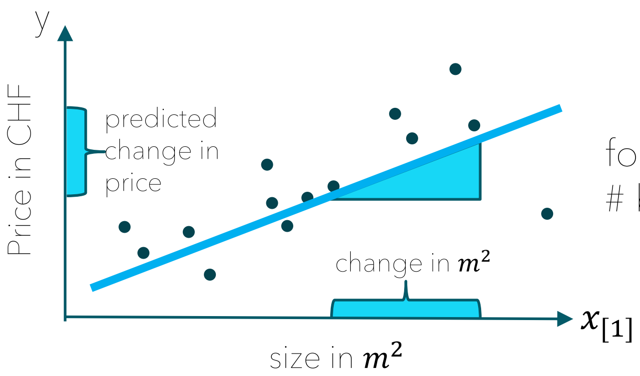
- closed-form

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

Visualization for d = 2: $f(x) = w_0 + w_1 x_{[1]} + w_2 \boxed{x_{[2]}}$ fix



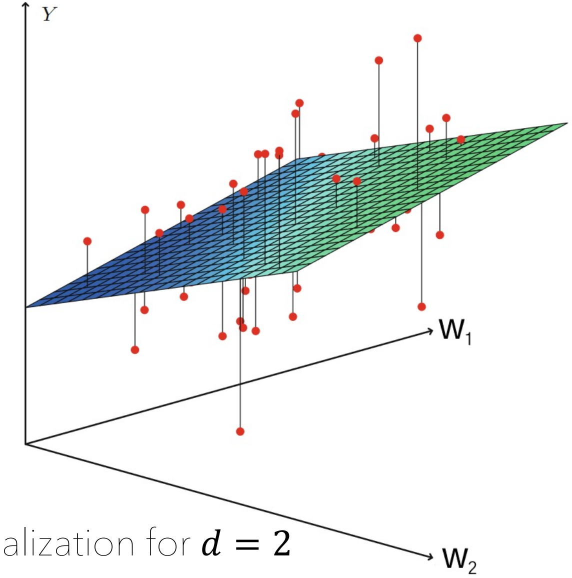
along the
"cut"



for a fixed
bathrooms

Training loss using the squared loss

Function class F	Loss function	Optimization
• linear function	• squared loss	• closed-form



Visualization for $d = 2$

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

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

equivalent to minimizing over vector w

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

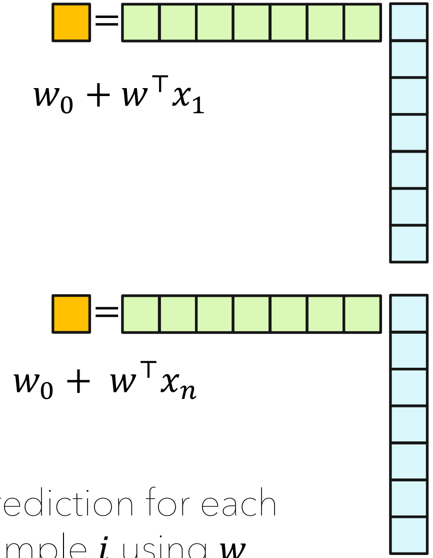
notation:
x_i - input attributes of sample i
$x_{[i]}$ - i-th attribute
w_i - i-th element of vector w

Training loss in matrix vector notation

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

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

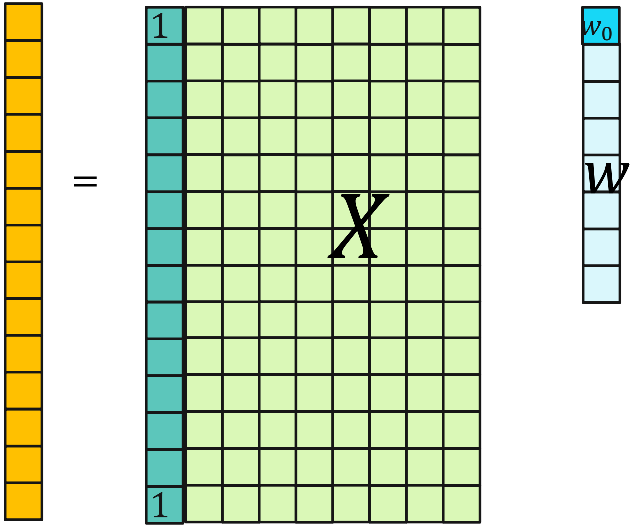
$\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^d$
is the all-ones vector



Prediction for each sample i using w

aggregate into one vector

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



Minimizing the training loss

- For simplicity let's set $w_0 = 0$ and minimize over w

- Training loss $L(0, w) = \frac{1}{n}||y - Xw||^2 = \frac{1}{n}||y||^2 - \frac{2}{n}y^\top Xw + \frac{1}{n}w^\top X^\top Xw$

  with gradient $\nabla_w L(0, w) = \frac{2}{n}(X^\top Xw - X^\top y)$ and Hessian $D^2 L(0, w) = \frac{2}{\mathrm{n}} X^\top X$

- We'll now look at two ways to find the minimum
  - *stationary point condition (gradient = 0)*
  - *geometric argument (orthogonal projection)*

Optimal solution: via stationary point condition

Function class F	Loss function	Optimization
• linear function	• squared loss	• closed-form

- Again: Minimum $\hat{w} = \mathbf{argmin}_w \, L(0, w)$ must be a stationary point, i.e. satisfying $\nabla_w L(0, \hat{w}) = 0$
- All stationary points of this quadratic loss $\frac{1}{n}||y - Xw||^2$ are minima, because the Hessian $\frac{2}{\mathrm{n}} X^\top X$ is positive semi-definite (psd) *(see math recap Sec 1.6., 2.5 and details next week)*
- $\nabla_w L(0, w) = \frac{2}{n}(X^\top X w - X^\top y)$ together with stationary point condition $\nabla_w L(0, \hat{w}) = 0$ yields

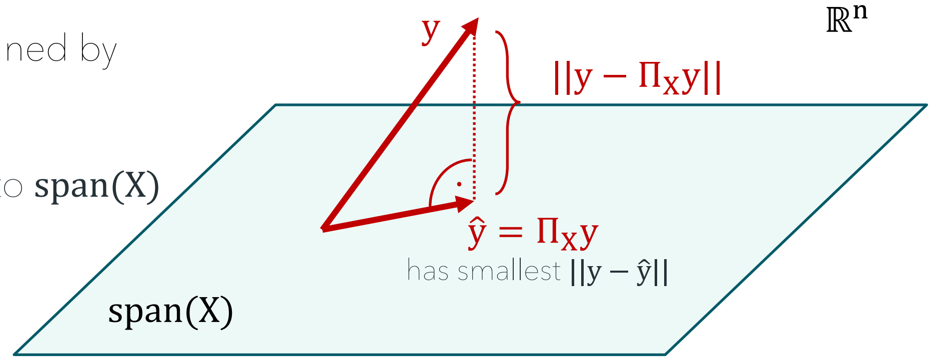
$$\rightarrow X^\top y = X^\top X \, \hat{w} \rightarrow \hat{w} = (X^\top X)^{-1} X^\top y$$

Optimal solution: via geometric argument

Function class F	Loss function	Optimization
• linear function	• squared loss	• closed-form

$$\hat{w} = \operatorname{argmin}_w \left\| \; {}^{n}\boxed{y} \; - \; {}^{n}\overset{d}{\boxed{X}} \; \boxed{w}^{d} \; \right\|$$

- set of all possible Xw: $\mathbf{span}(X)$ (subspace spanned by columns of X, *math recap)*
- let Π_X be the orthogonal projection matrix onto $\mathbf{span}(X)$
- then, the closest point to y on $\mathbf{span}(X)$ is $\Pi_X y$ (math recap Section 1.4)



Optimal solution: via geometric argument

We now derive the expression for the vector $\hat{w}$ such that $X\hat{w} = \Pi_X y$

We know that because $X\hat{w}$ is an orthogonal projection

- the residual $y - X\hat{w}$ orthogonal to all $v \in span(X) \iff (y - X\hat{w})^\top X w = 0$ for all $\mathbf{w} \in \mathbb{R}^d$

Optimal solution: via geometric argument

We now derive the expression for the vector $\widehat{\boldsymbol{w}}$ such that $\boldsymbol{X}\widehat{\boldsymbol{w}} = \Pi_X \boldsymbol{y}$ (see also Exerc. 1.6. in recap notes)

We know that because $\boldsymbol{X}\widehat{\boldsymbol{w}}$ is an orthogonal projection

- the residual $\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{w}}$ orthogonal to all $\boldsymbol{v} \in span(X) \iff (y - X\widehat{w})^\top X w = 0$ for all $\mathbf{w} \in \mathbb{R}^d$
- hence we require $X^\top(y - X\widehat{w}) = 0 \iff X^\top y = X^\top X\, \widehat{w}$ (called normal equations!)

→ this yields the unique solution $\widehat{w} = (X^\top X)^{-1} X^\top y$ if $X^\top X$ is invertible

Question for you until next week (Answers discussed next week):

How many minima $\widehat{w}$ does $\frac{1}{n}||y - Xw||^2$ have? Argue using the matrix $X^\top X$

(A) One (B) Multiple (C) None (D) Depends on $\{(x_i, y_i)\}_{i=1}^n$

What you can do now

On a high level:

- for inputs with multiple attributes
- teach a machine how to use training data to output a prediction rule/model

  (that can be used for prediction of a new point)
- by deriving a closed-form solution for the best linear fit

  (in the square loss sense) on the training data (training loss minimzer)

References / acknowledgements

- We will provide lecture notes that explain the derivations in more detail
- Other reference: ISLR Chapter 3