Institute for Advanced Studies
in Basic Sciences
Gava Zang, Zanjan, Iran

Regression with Linear Models

By Dr. Parvin Razzaghi (p.razzaghi@iasbs.ac.ir)

Winter 2024

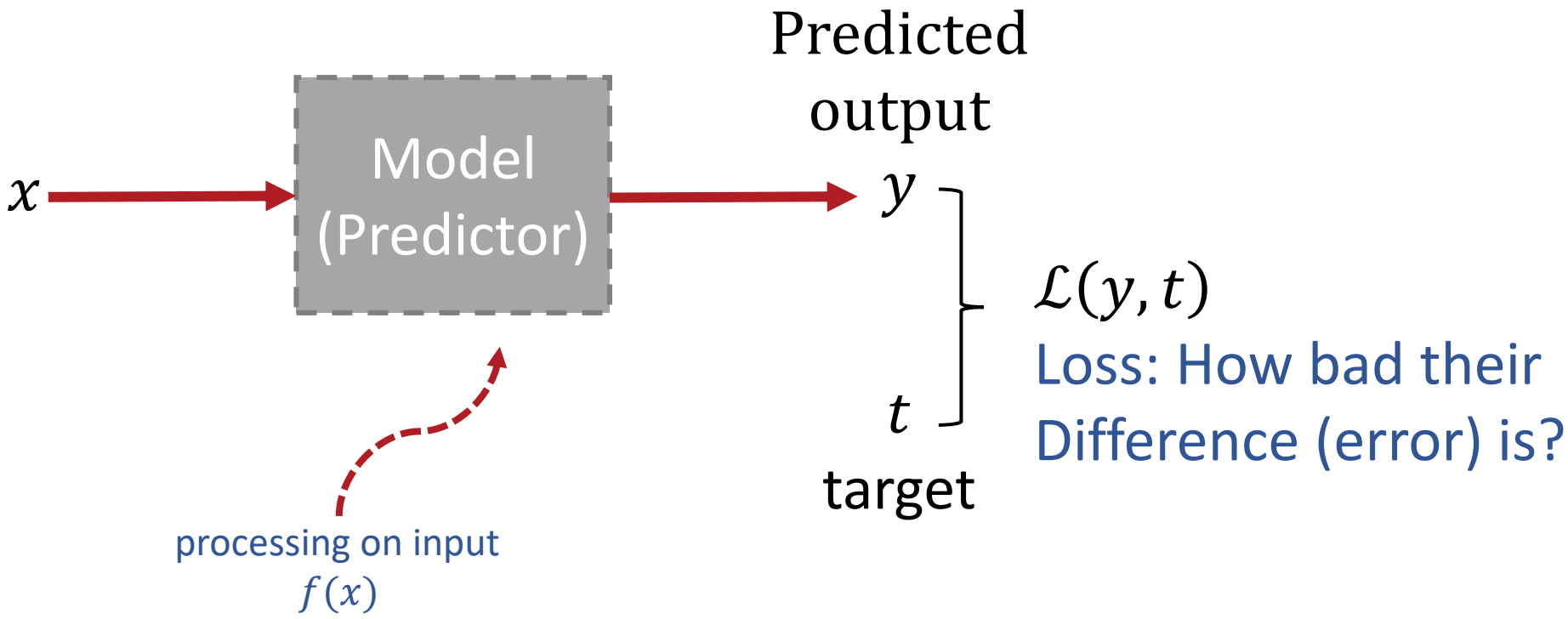
Slides credit

- Professor:
  - Dr. Parvin Razzaghi
- University:
  - University of Toronto and Vector Institute
- Original Slides:
  - Amir-massoud Farahman
- Acknowledgment:
  - Credit for slides goes to many members of the ML Group at the U of T, and beyond, including (recent past): Roger Grosse, Murat Erdogdu, Richard Zemel, Juan Felipe Carrasquilla, Emad Andrews. For any comments or suggestions, please contact: p.razzaghi@iasbs.ac.ir

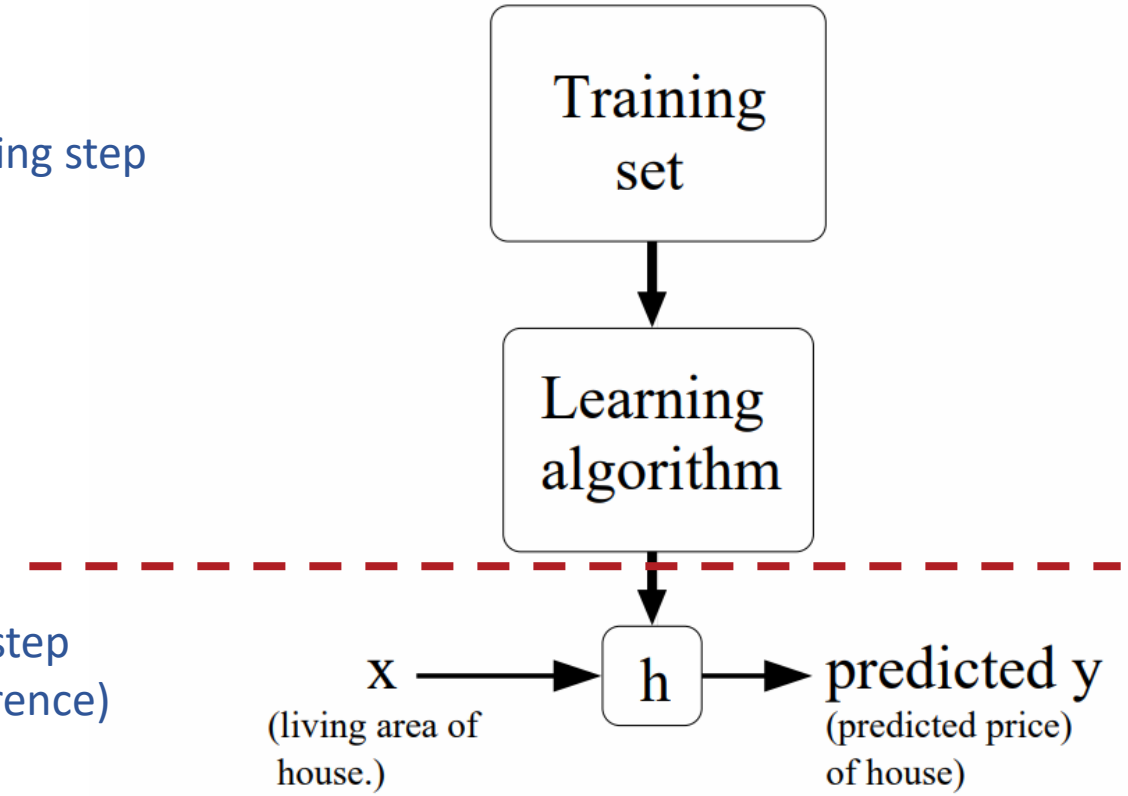
Table of Contents

- Modular Approach to ML
- Regression
  - Linear Regression
  - Basic Expansion
  - Regularization
  - Probabilistic Interpretation of the Squared Error
  - Bias-variance tradeoff
  - Bayesian linear regression

Modular Approach ML Algorithm Design



Training step



Modular Approach ML Algorithm Design

- we will take a more modular approach:
  - define the problem
  - provide a training dataset
  - choose a model describing the relationships between variables of interest
  - define a loss function quantifying how bad the fit to the data is
  - (possibly) choose a regularizer saying how much we prefer different candidate models (or explanations of data), before (prior to) seeing the data
  - fit the model that minimizes the loss function and satisfy the constraint/penalty imposed by the regularizer, possibly using an optimization algorithm
- Mixing and matching these modular components gives us a lot of new ML methods.

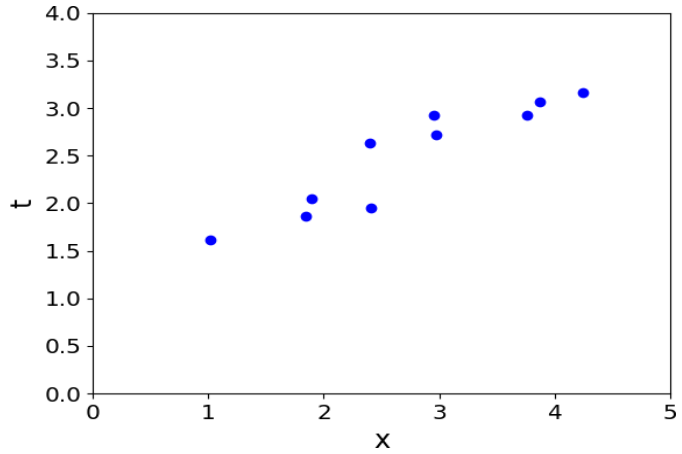
Define the problem & provide a dataset

The Supervised Learning Setup

- Features
  - Living area
  - Number of bedrooms
  - …
- Target variable
  - Price

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

The Supervised Learning Setup



Recall that in supervised learning:

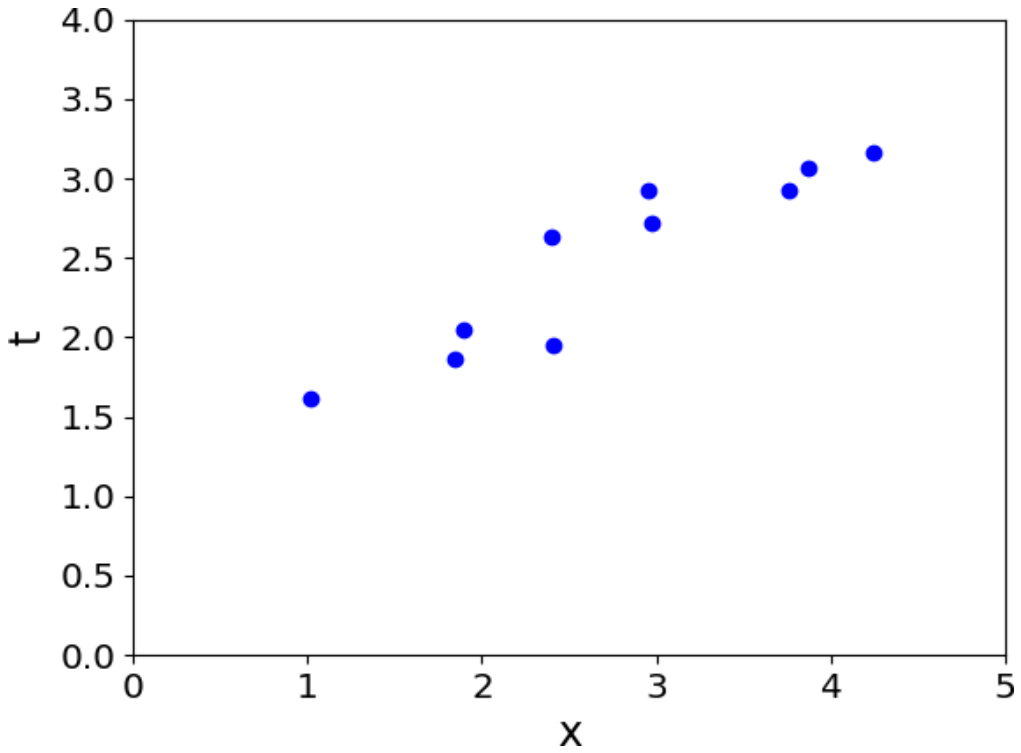
- There is a target $t \in T$ (also called response, outcome, output, class)
- There are features $\boldsymbol{x} \in X$ (also called inputs or covariates)

  The goal is to learn a function $f : X \rightarrow T$ such that

$$t \approx y = f(x),$$

based on given data $D = \{(\boldsymbol{x}^{(i)}, t^{(i)}) \; for \; i = 1, 2, \ldots, N\}$.

choose a model describing the relationships between variables of interest?


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
t
0
1
2
3
4
5
x

Regression with Linear Models

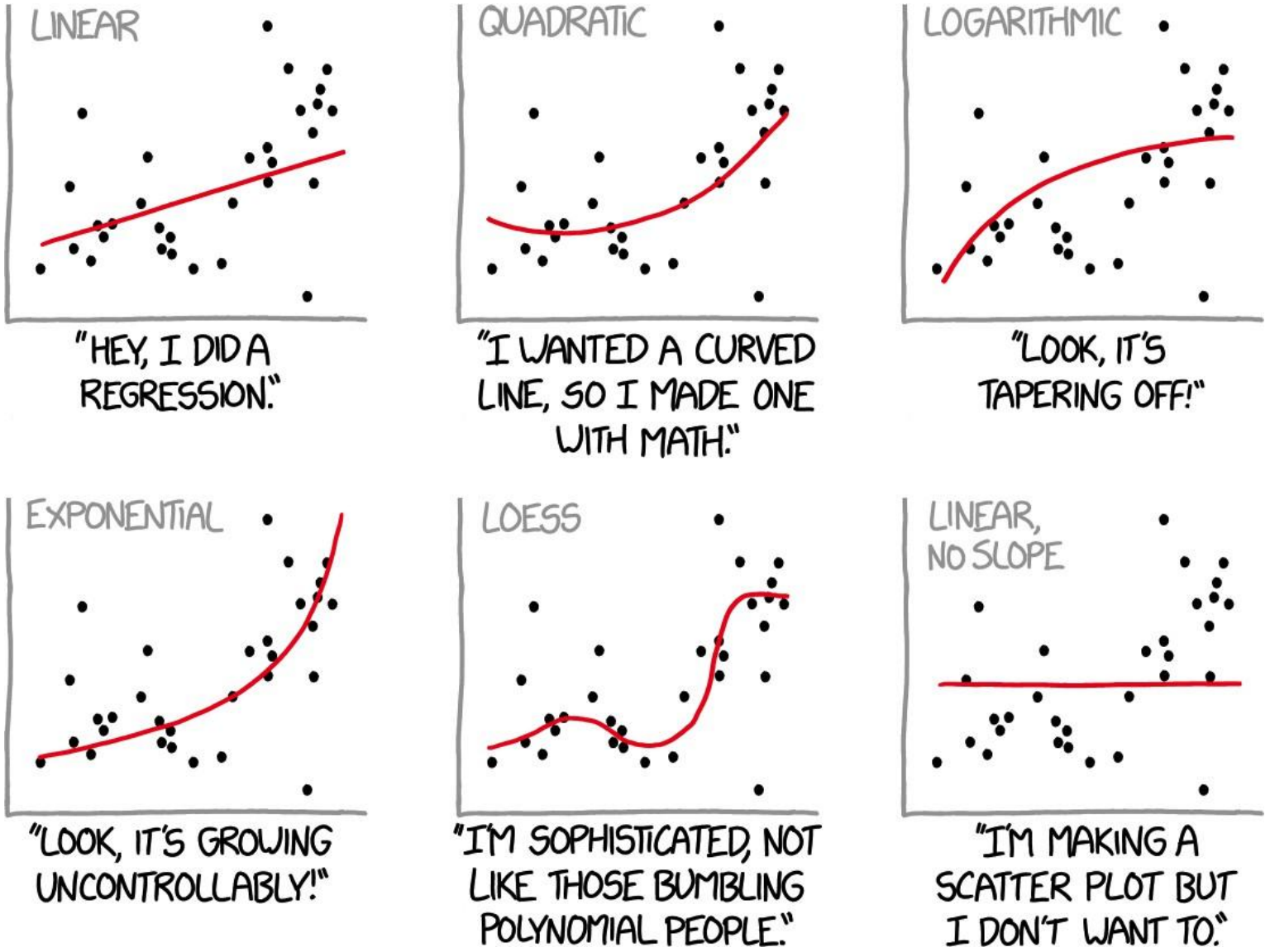


Image credit: xkcd (cropped)

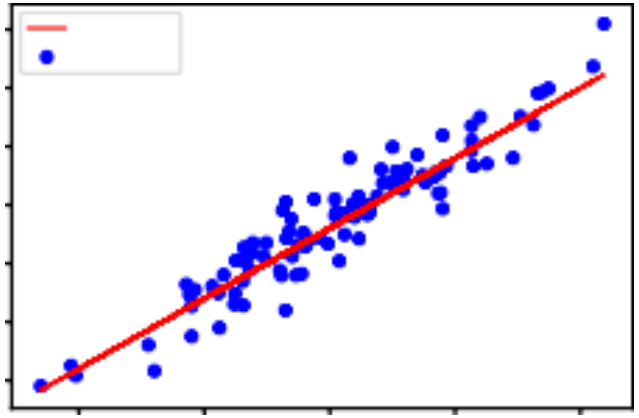
Linear Regression – Model

- Model: In linear regression, we use linear functions of the inputs $\boldsymbol{x} = (x_1, \ldots, x_D)$ to make predictions y of the target value t:

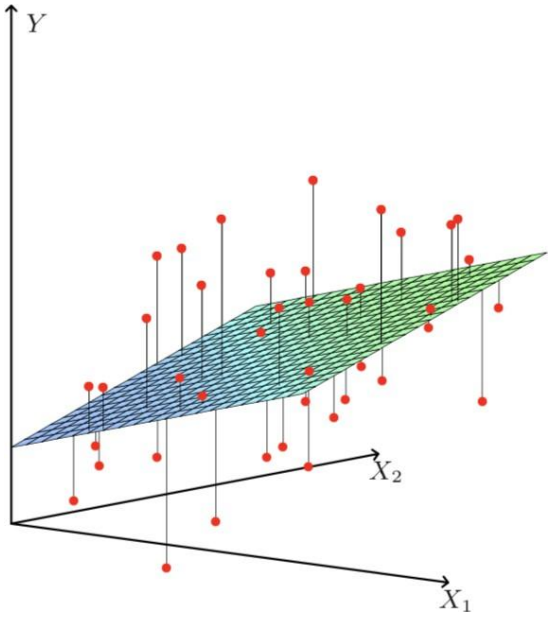
$$y = f(x) = \sum_j w_j x_j + b$$

  - y is the prediction
  - $\boldsymbol{w}$ is the weights
  - b is the bias (or intercept) (do not confuse with the bias-variance tradeoff in the next lecture)
- $\boldsymbol{w}$ and b together are the parameters
- We hope that our prediction is close to the target: $y \approx t$.

What is Linear? 1 Feature vs. D Features



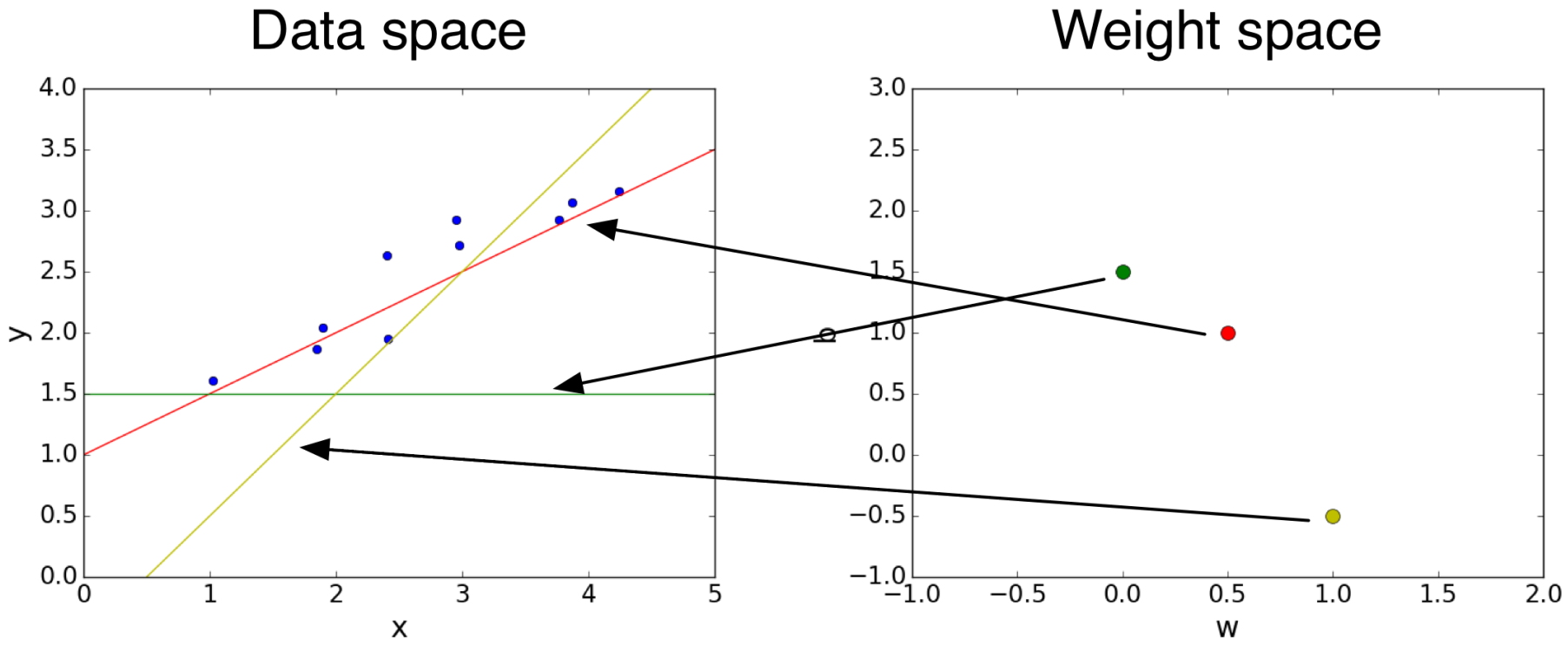
- If we have only 1 feature:
  $y = wx + b$ where $w, x, b \in R$
- y is linear in x



- If we have D features:
  $y = \boldsymbol{w}^T \boldsymbol{x} + b$ where $\boldsymbol{w}, \boldsymbol{x} \in R^D, b \in R$
- y is linear in $\boldsymbol{x}$

Relation between the prediction y and inputs **x** is linear in both cases.

Weight Space vs. Data Space



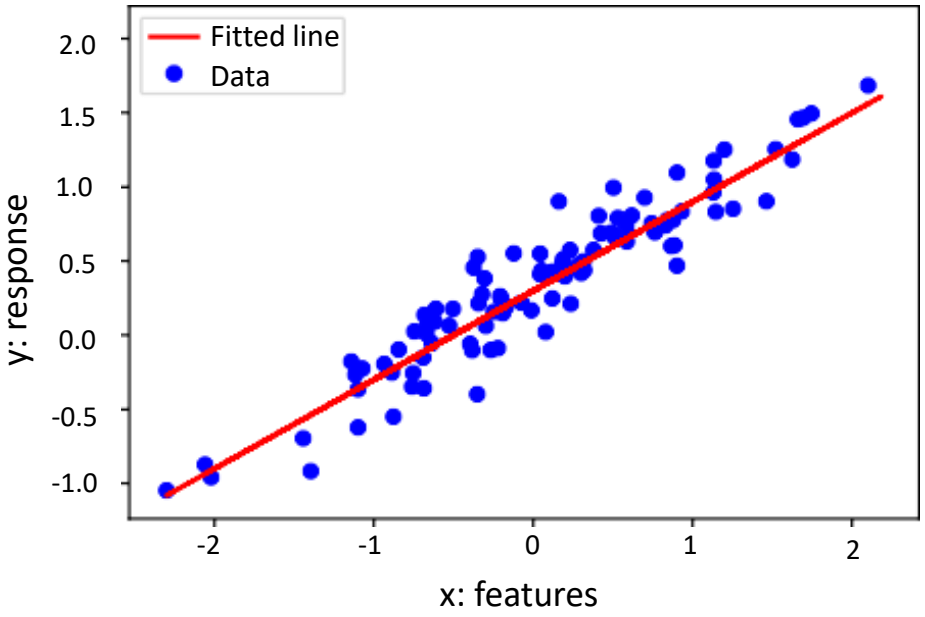
Recall that:

$$y = f(x) = \sum_j w_j x_j + b$$

Linear Regression

We have a dataset $D = \{(x^{(i)}, t^{(i)})\}_{i=1}^{N}$ where:

- $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_D^{(i)})^T \in \mathbb{R}^D$ are the inputs, e.g., age, height
- $t^{(i)} \in \mathbb{R}$ is the target or response, e.g., income
- predict $t^{(i)}$ with a linear function of $x^{(i)}$:



- $t^{(i)} \approx y^{(i)} = \boldsymbol{w}^T \boldsymbol{x}^{(i)} + b$
- Find the "best" line (w, b).

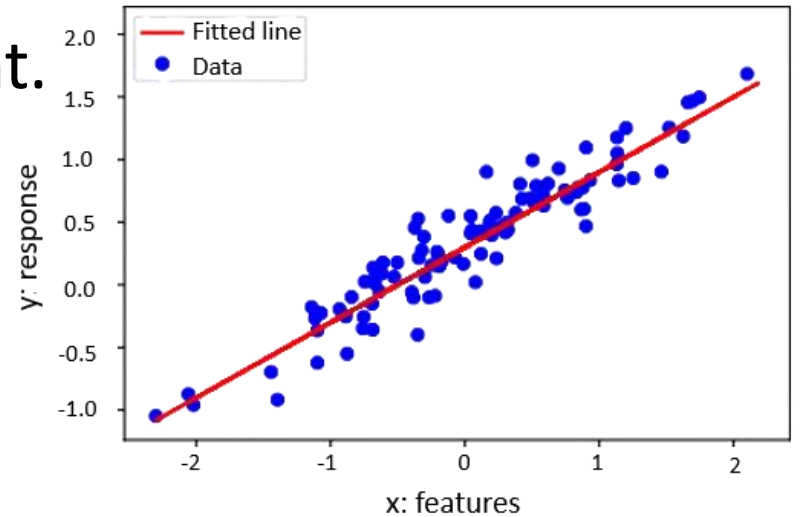
- Q: How should we find the best line?

Linear Regression – Loss Function

- How to quantify the quality of the fit to data?
- A loss function $\mathcal{L}(y, t)$ defines how bad it is if, for some input $\boldsymbol{x}$, the algorithm predicts y, but the target is actually t.
- Squared error loss function:

$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$$

- $y - t$ is the residual, and we want to make its magnitude small
- The $\frac{1}{2}$ factor is just to make the calculations convenient.



Linear Regression – Loss Function

- Cost function: loss function averaged over all training examples

$$\mathcal{J}(\boldsymbol{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(y^{(i)}, t^{(i)}\right)$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(y^{(i)} - t^{(i)}\right)^2$$

$$= \frac{1}{2N}\sum_{i=1}^{N}(\boldsymbol{w}^T x^{(i)} + b - t^{(i)})^2$$

Linear Regression – Loss Function

- To find the best fit, we find a model (parameterized by its weights $\boldsymbol{w}$ and b) that minimizes the cost:

$$\underset{(\boldsymbol{w}, b)}{\text{minimize}}\, \mathcal{J}(\boldsymbol{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(y^{(i)}, t^{(i)}\right)$$

- The terminology is not universal. Some might call “loss” pointwise loss and the “cost function” the empirical loss or average loss.

Vector Notation

- We can organize all the training examples into a design matrix **X** with one row per training example, and all the targets into the target vector **t**.

$$X = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ x^{(3)T} \end{bmatrix} = \begin{bmatrix} 8 & 0 & 3 & 0 \\ 6 & -1 & 5 & 3 \\ 2 & 5 & -2 & 8 \end{bmatrix}$$

One feature across all training examples

One training example (vector)

- Computing the predictions for the whole dataset:

$$\boldsymbol{X}\boldsymbol{w} + b\mathbf{1} = \begin{bmatrix} w^T x^{(1)} + b \\ \vdots \\ w^T x^{(N)} + b \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix} = y$$

Vectorization

- Computing the squared error cost across the whole dataset:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + b\mathbf{1}$$

$$\mathcal{J} = \frac{1}{2N} \|y - t\|^2$$

- Note that sometimes we may use $\mathcal{J} = \frac{1}{2N} \|y - t\|^2$, without $\frac{1}{N}$ normalizer. That would correspond to the sum of losses, and not the average loss. That does not matter as the minimizer does not depend on N .

Vectorization

- We can also add a column of 1s to the design matrix, combine the bias and the weights, and conveniently write

$$X = \begin{bmatrix} 1 & \left[x^{(1)}\right]^T \\ 1 & \left[x^{(2)}\right]^T \\ 1 & \vdots \end{bmatrix} \in \mathbb{R}^{N \times D+1} \text{ and } w = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \end{bmatrix} \in \mathbb{R}^{D+1}$$

Then, our predictions reduce to $\boldsymbol{y} = \boldsymbol{Xw}$.

Any questions?