tai911s

José Quenum
PhD Computer Science

linear models

principle

from a dataset $(\mathrm{X}_{\mathit{I}}, \mathrm{Y}_{\mathit{I}})$ predict Y_{J} corresponding to input X_{J}

recap…

- step 0: **get**, **clean** and **understand** the dataset

- step 1:
  - decide which **function** to **learn** (or approximate)…
  - **select** the useful **features**

- step 2:
  - choose a training **algorithm**

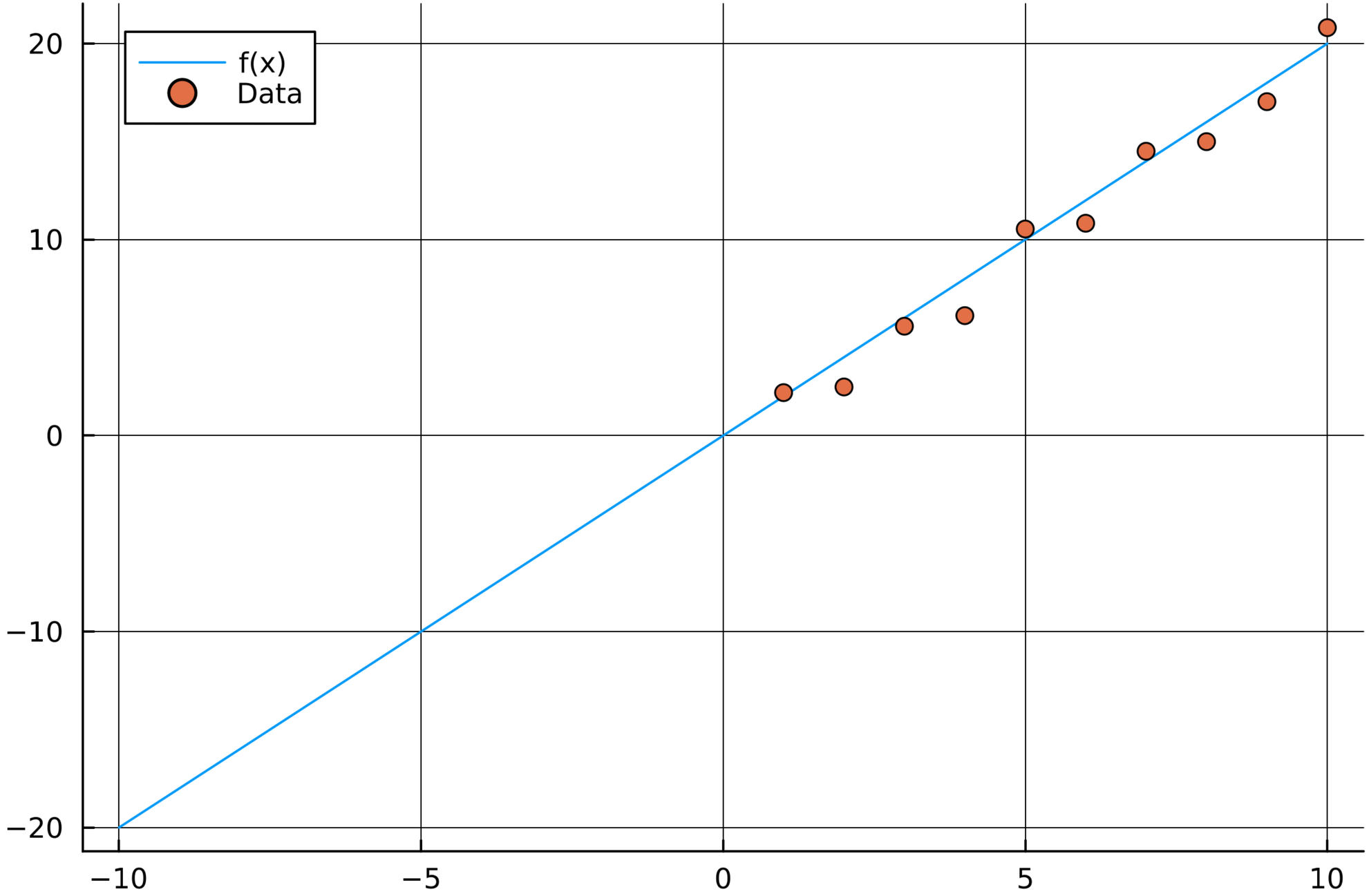
- step 3:
  - **train**, **test** and **evaluate** the model

tasks

- **regression**
  - **fit** a continuous **functional relation** between X_i and Y_i
  - e.g.,
    - weather forecasting, house price prediction

- **classification**
  - **find** out **decision boundaries** between the classes in a dataset
  - e.g.,
    - spam filter, hand-written character recognition

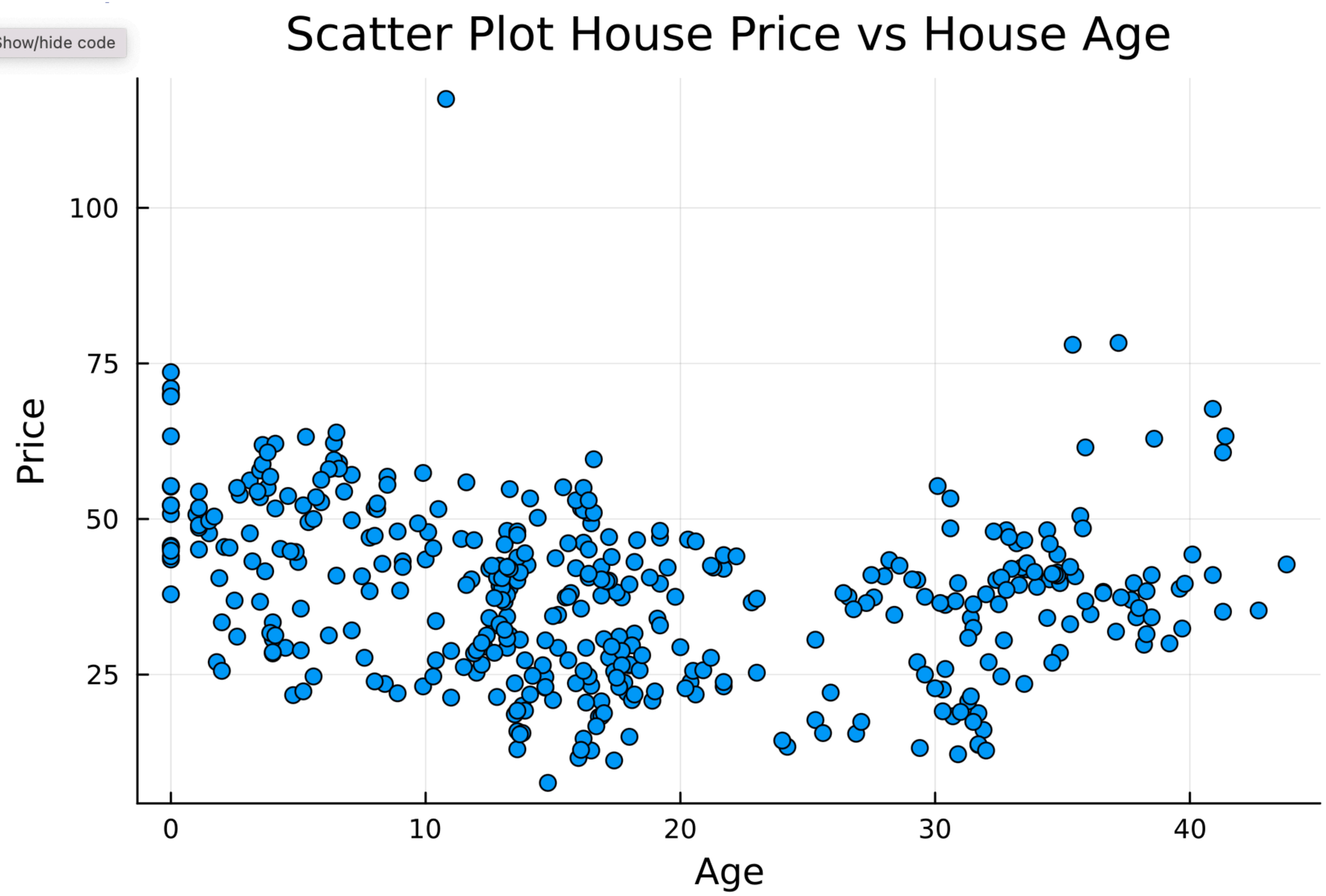
linear regression


f(x)
Data
20
10
0
−10
−20
−10
−5
0
5
10

- univariate (simple) linear regression
- multi-variate (multiple) linear regression

transaction_date	house_age	distance_to_train_st	convenience_stores	latitude	longitude	house_price
2012.9.2	32	84.8788	10	24.983	121.54	37.9
2013.5.8	19.5	306.595	9	24.9803	121.54	42.2

Show/hide code



$$\mathsf{h}(\mathrm{X}) = \mathrm{X} \times \Theta^{\top}$$

$$\mathsf{h}(\mathrm{X}) = \theta_0 + \theta_1 \mathrm{x}^1 + \theta_2 \mathrm{x}^2 + \ldots + \theta_\mathrm{n} \mathrm{x}^\mathrm{n}$$

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_\mathrm{n} \end{bmatrix}$$

$$\mathrm{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2^1 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_m^1 & x_m^2 & \cdots & x_m^n \end{bmatrix}$$

- residual difference between the hypothesis ($h(X_i)$) and the actual value (Y_i)
- loss function
  - mean square error
  - mean absolute error

$$\mathcal{L}(\mathrm{h}(\mathrm{X}), \mathrm{Y}) = \frac{1}{m} \sum_{i=1}^{m} (\mathrm{h}(\mathrm{X}^i) - \mathrm{Y}^i)^2$$

$$\mathcal{L}(\mathrm{h}(\mathrm{X}), \mathrm{Y}) = \frac{1}{m} \sum_{i=1}^{m} |\mathrm{h}(\mathrm{X}^i) - \mathrm{Y}^i|$$

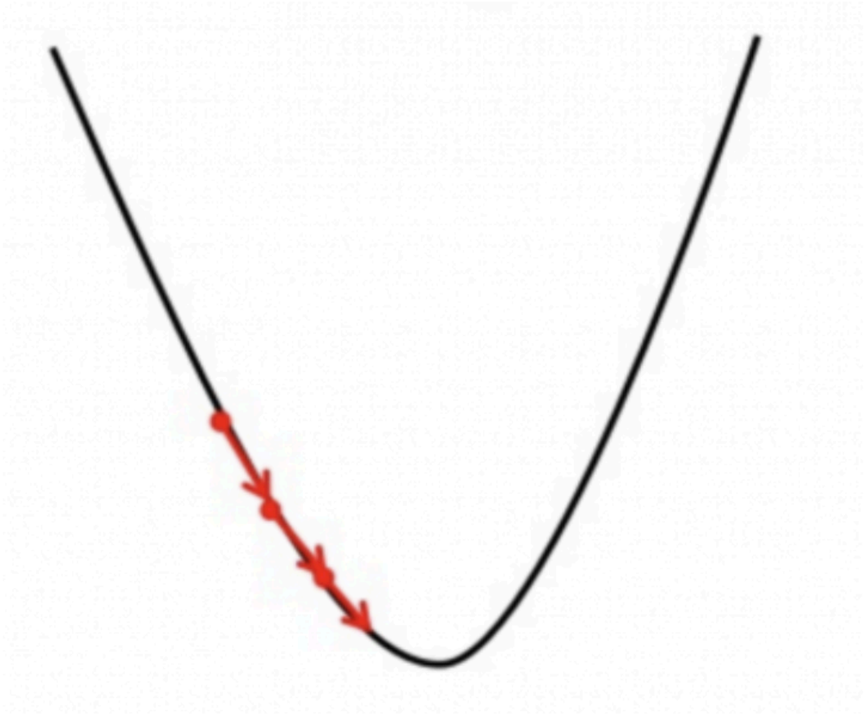
$$argmin \;\; \mathcal{L}(\mathsf{h}(\mathrm{X}), \mathrm{Y}) = \frac{1}{m} \sum_{i=1}^{m} (\mathrm{h}(\mathrm{X}^{i}) - \mathrm{Y}^{i})^{2}$$

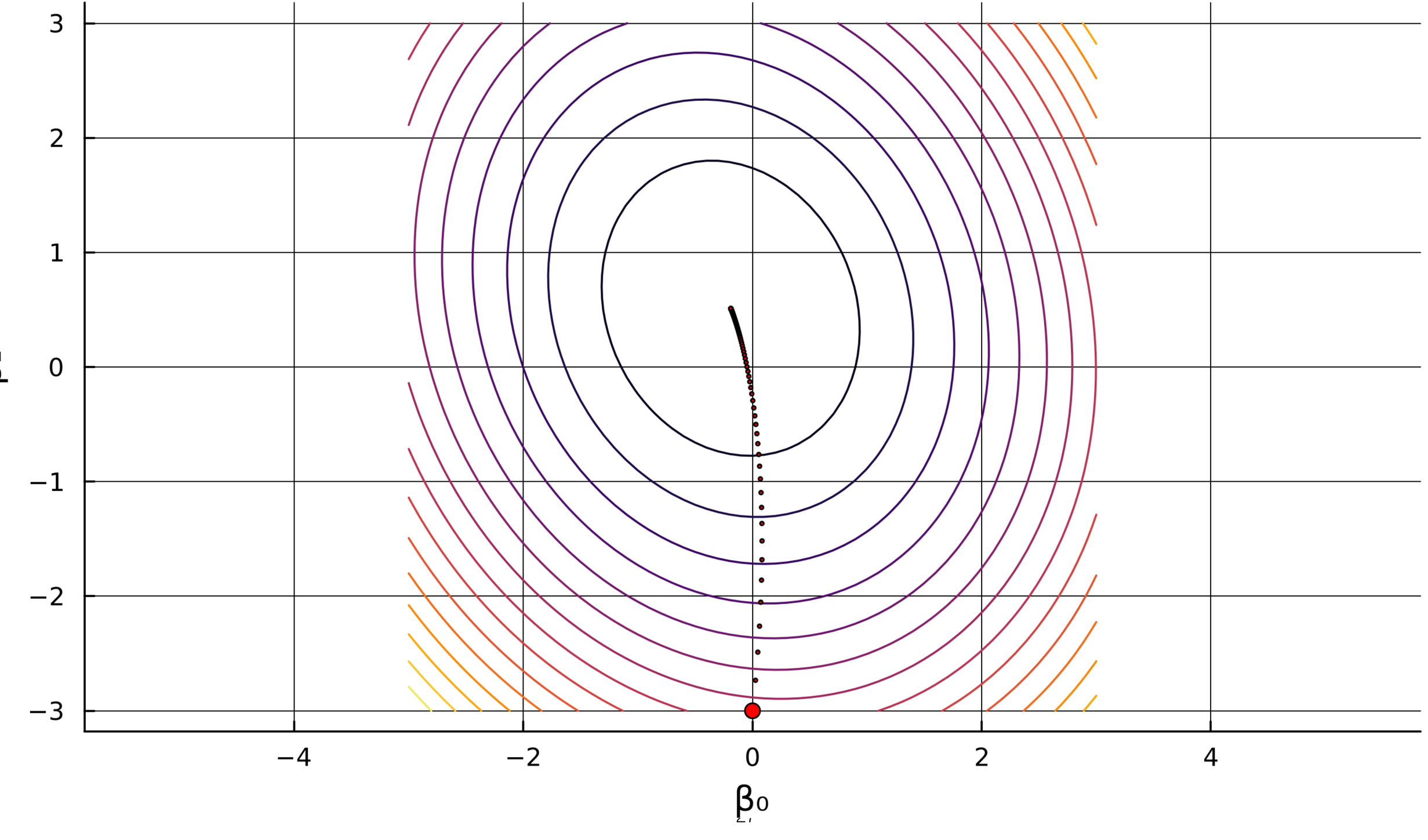
- vars in this optimisation fn?

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix}$$

optimisation method

- gradient descent
- stochastic gradient descent
- adam
- Etc.




3
2
1
0
−1
−2
−3
−4
−2
0
2
4
β0

$$\theta' = \theta - \alpha \bigtriangledown_{\theta} (\mathcal{L})$$

$$\theta'_{J} = \theta_{J} - \alpha \bigtriangledown_{\theta_{J}} (\mathcal{L})$$

$$\bigtriangledown_{\theta_{J}} (\mathcal{L}) = \frac{\partial \mathcal{L}_{J}}{\partial \theta_{J}}$$

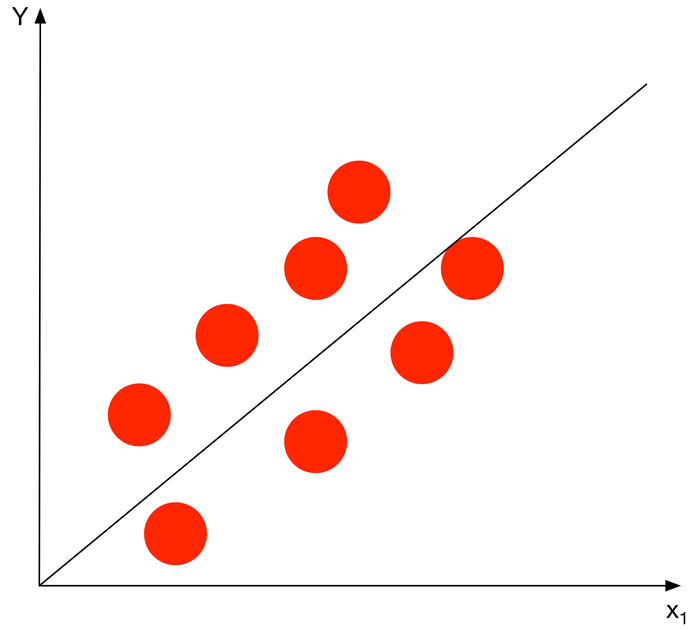
$$\theta_j = 0$$

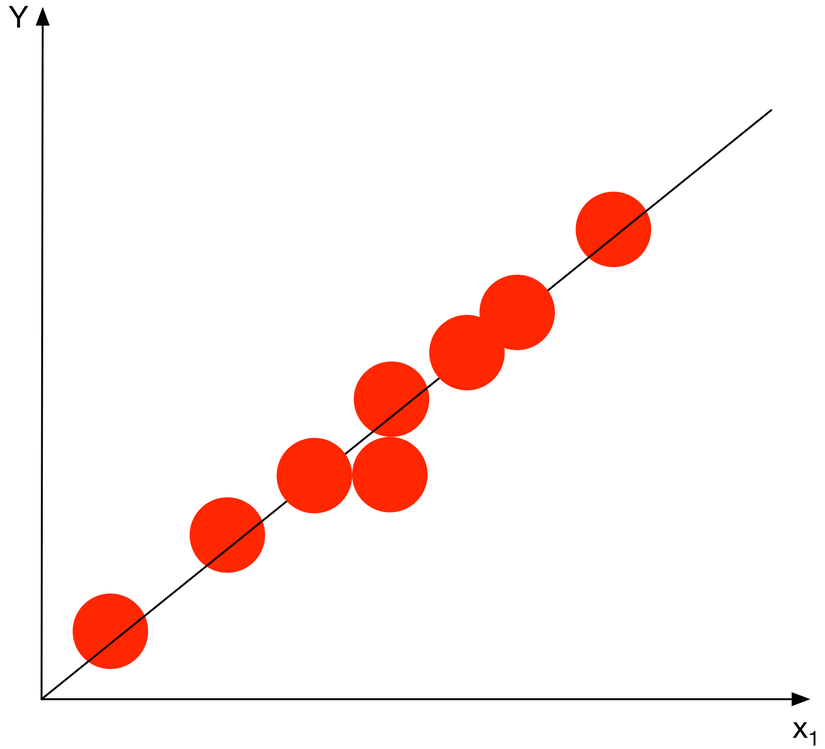
$$\theta'_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\mathrm{h}(\mathrm{X}^i) - \mathrm{Y}^i)\mathrm{X}^i_j$$

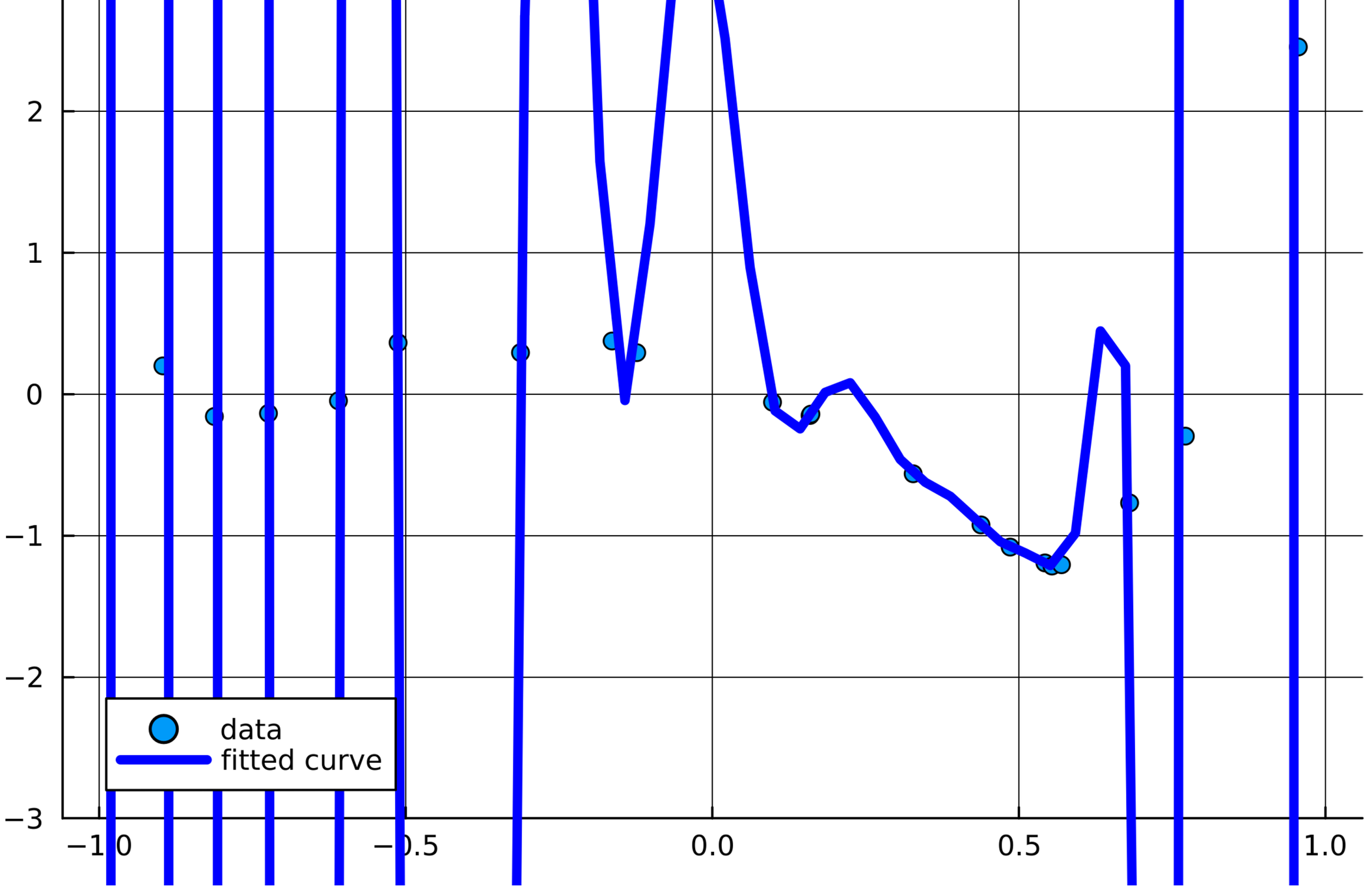
$$for\ t = 1..\mathsf{T}$$

but…

- overfitting
  - low bias and high variance
- underfitting
  - high bias and low variance


Y
x_1


Y
x_1


2
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
data
fitted curve

- regularisation
  - Lasso: absolute value of weight
  - Ridge: square weight

$$\mathrm{J}(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(\mathrm{h}(\mathrm{X}^{i}) - \mathrm{Y}^{i})^{2} + \frac{\lambda}{2n}\sum_{j=1}^{n}\theta_{j}^{2}$$

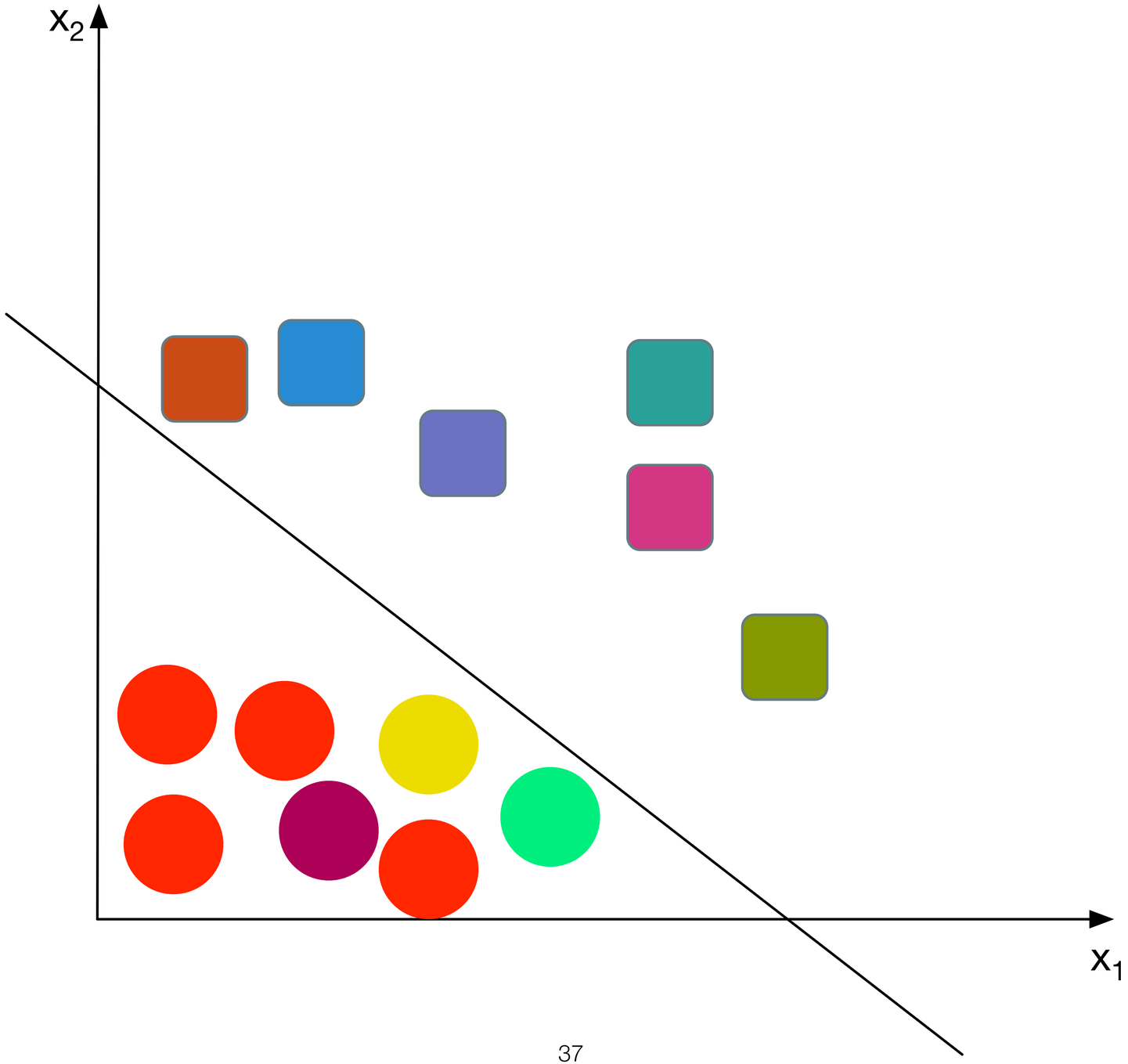
$$\theta'_{j} = \theta_{j} - \alpha[\frac{1}{m}\sum_{i=1}^{m}(\mathrm{h}(\mathrm{X}^{i}) - \mathrm{Y}^{i})\mathrm{X}_{j}^{i} + \frac{\lambda}{n}\theta_{j}]$$

$$for\ j > 0$$

$$\theta'_{0} = \theta_{0} - \frac{\alpha}{m}\sum_{i=1}^{m}(\mathrm{h}(\mathrm{X}^{i}) - \mathrm{Y}^{i})\mathrm{X}_{j}^{i}$$

- metrics
  - mean square error

Logistic regression


x_2
x_1

- binomial classification
- multinomial classification

binomial classification

$$y = \begin{cases} 0 \\ 1 \end{cases}$$

$$z = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \ldots + \theta_n x^n$$

$$z = X \times \Theta$$

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix}$$

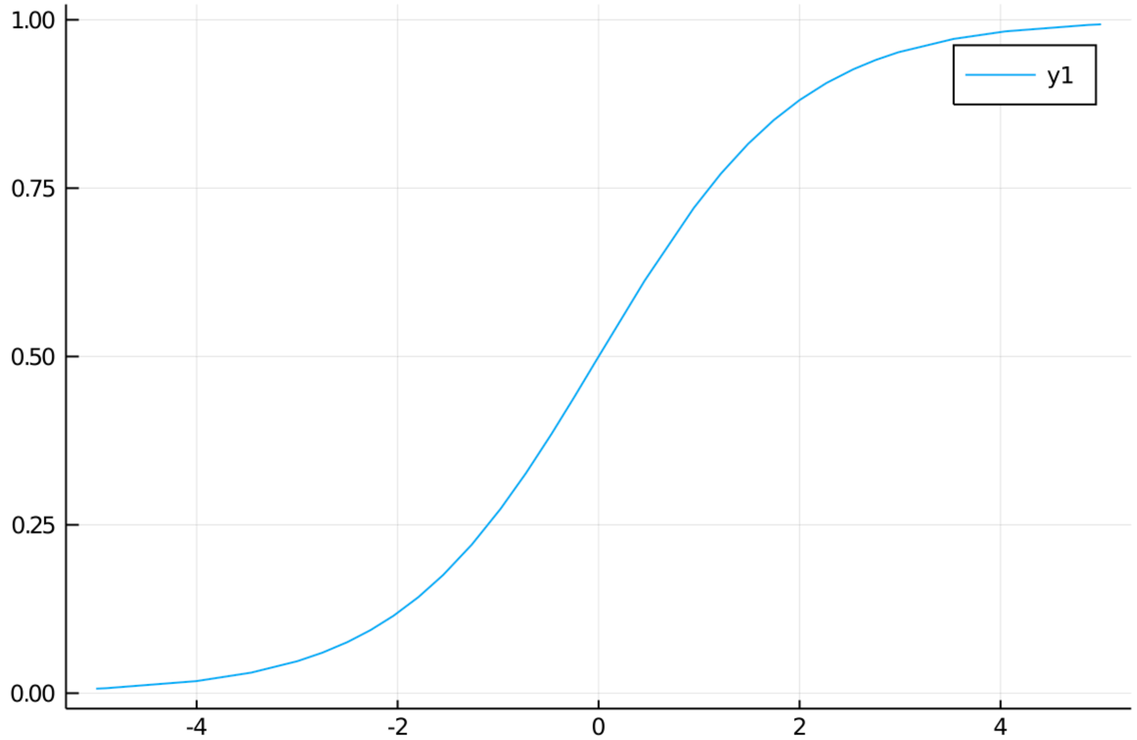
$$X = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2^1 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_m^1 & x_m^2 & \cdots & x_m^n \end{bmatrix}$$

- **sigmoid** function

$$\mathrm{h(x)} = \frac{1}{1 + \exp^{-\mathrm{z(x)}}}$$

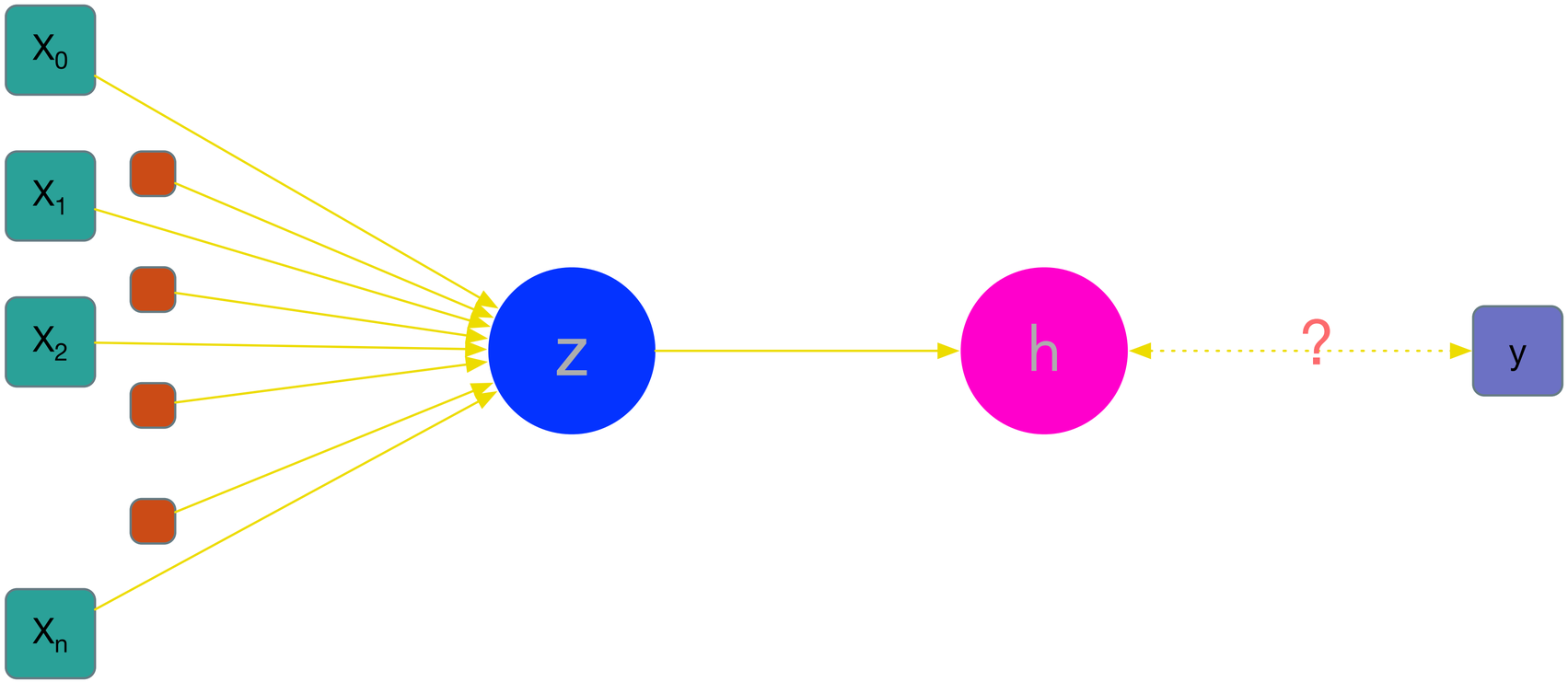
$$\mathrm{z} = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \ldots + \theta_n x^n$$

$$\mathrm{h(x)} = p(\mathrm{y} = 1 | \mathrm{x}; \theta)$$



$$\mathrm{h(x)} = p(\mathrm{y} = 1 | \mathrm{x}; \theta)$$

$$\begin{cases} \mathrm{h(x)} >= 0.5 \implies \mathrm{y} = 1 \\ \mathrm{h(x)} < 0.5 \implies \mathrm{y} = 0 \end{cases}$$


x_0
x_1
x_2
x_n
z
h
?
y

$$\mathcal{L}(\mathrm{h}(\mathrm{x}), \mathrm{y}) = \begin{cases} -\log(\mathrm{h}(\mathrm{x})) & \text{if } \mathrm{y} = 1 \\ -\log(1 - \mathrm{h}(\mathrm{x})) & \text{if } \mathrm{y} = 0 \end{cases}$$

$$\mathcal{L}(\mathrm{h}(\mathrm{x}), \mathrm{y}) = -\mathrm{y} \log(\mathrm{h}(\mathrm{x})) - (1 - \mathrm{y}) \log(1 - \mathrm{h}(\mathrm{x}))$$

$$\mathsf{J}(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} (-\mathrm{y}^{i} \log(\mathrm{h}(\mathrm{x}^{i})) - (1 - \mathrm{y}^{i}) \log(1 - \mathrm{h}(\mathrm{x}^{i}))) + \frac{\lambda}{2n} \sum_{j=1}^{n} \theta_{j}^{2}$$

$$argmin \quad \frac{1}{2m}\sum_{i=1}^{m}(-\mathrm{y}^{i}\log(\mathrm{h}(\mathrm{x}^{i}))-(1-\mathrm{y}^{i})\log(1-\mathrm{h}(\mathrm{x}^{i})))+\frac{\lambda}{2n}\sum_{j=1}^{n}\theta_{j}^{2}$$

$$\theta'_{j}=\theta_{j}-\alpha[\frac{1}{m}\sum_{i=1}^{m}(\mathrm{h}(\mathrm{X}^{i})-\mathrm{Y}^{i})\mathrm{X}_{j}^{i}+\frac{\lambda}{n}\theta_{j}]$$

$$for\ j>0$$

$$\theta'_{0}=\theta_{0}-\frac{\alpha}{m}\sum_{i=1}^{m}(\mathrm{h}(\mathrm{X}^{i})-\mathrm{Y}^{i})\mathrm{X}_{j}^{i}$$

- metrics
  - confusion matrix

	h = 0	h = 1
y = 0	true negative (TN)	false positive (FP)
y = 1	false negative (FN)	true positive (TP)

- accuracy
  - overall, how often is the classifier correct?
- precision
  - when it predicts “yes”, how often is it correct?

$$\textbf{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TN} + \mathrm{FP} + \mathrm{TP} + \mathrm{FN}}$$

$$\textbf{precision} = \frac{\mathrm{TP}}{\mathrm{FP} + \mathrm{TP}}$$

- error rate
  - overall, how often is it wrong?
- true positive rate (recall)
  - when it is actually “yes”, how often does it predict “yes”?

$$\textbf{error rate} = 1 - \textbf{accuracy}$$

$$\textbf{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- true negative rate (specificity)
  - when it is actually “no”, how often does it predict “no”
- and more…
  - e.g., F-score

$$\textbf{specifity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

multinomial classification

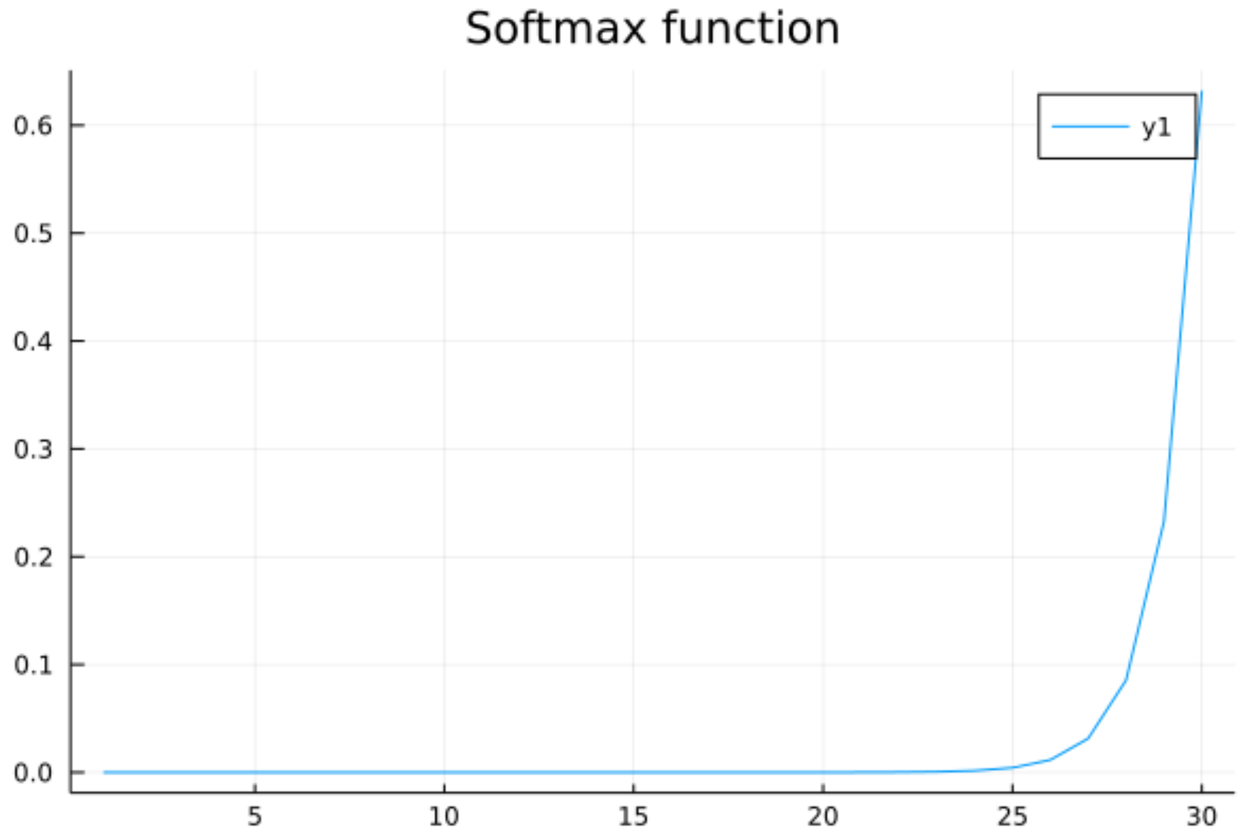
- first strategy

- use the training set for multiple separate binomial classifications
    - e.g., for a 3-class classifier c1, c2, c3
        - c1 vs c2 + c3; c2 vs c1 + c3 and c3 vs c1 + c2
- train the logistic classifier for each class to predict the probability of the outcome in a given class
- pick the class that maximises its probabilty

- second strategy

- use the **softmax** function instead of the sigmoid

$$\mathrm{h(x)} = \frac{\exp(\mathrm{Z}_i(X))}{\sum_{j=1}^{k} \exp \mathrm{Z}_j(\mathrm{X})} \; 1 \leq i \leq k$$


Softmax function
y1
0.6
0.5
0.4
0.3
0.2
0.1
0.0
5
10
15
20
25
30

in short…

- linear regression
- logistic regression
- regularisation