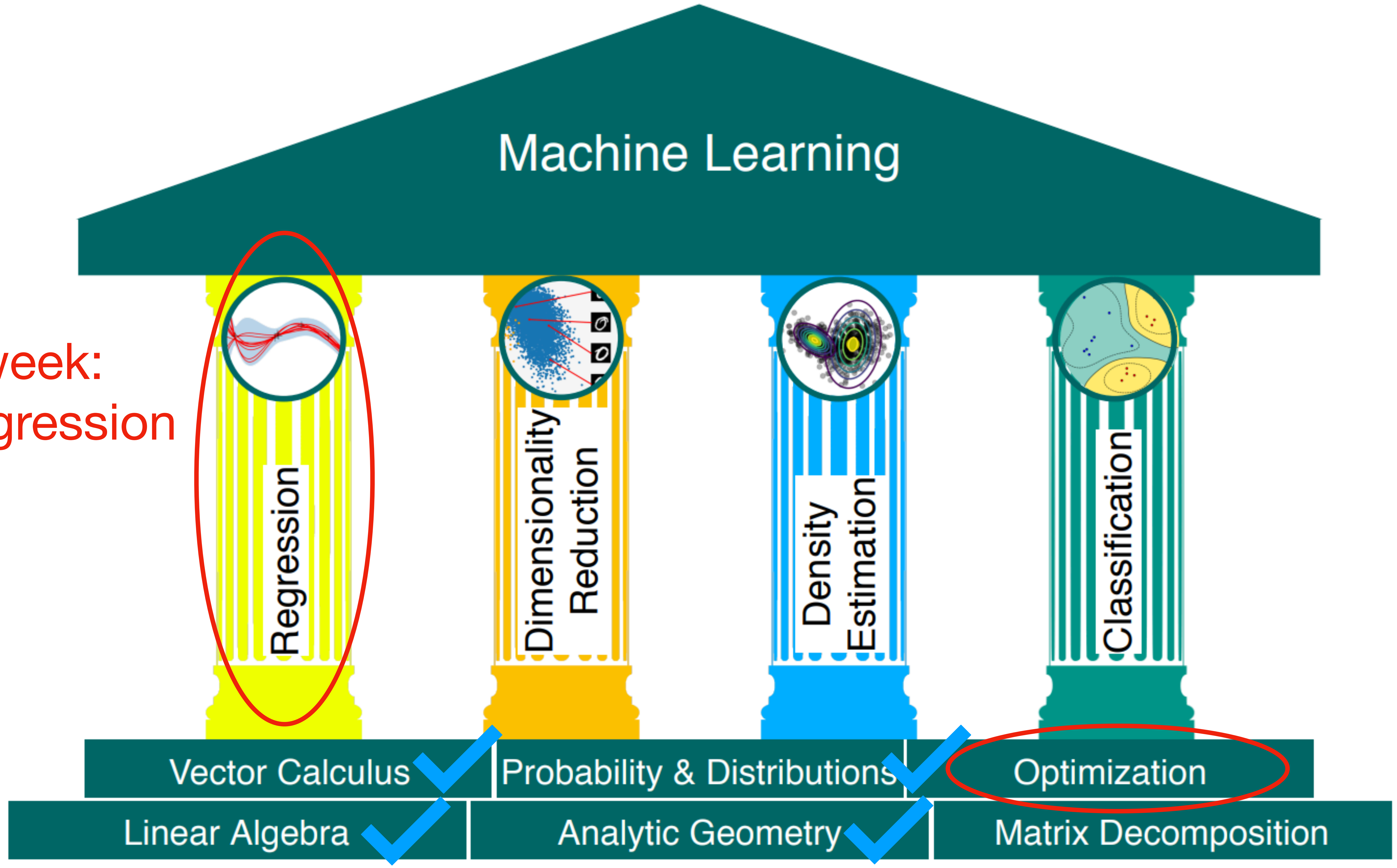
Linear regression

Week 7 - Introduction to ML / Thang Bui / ANU / 2023 S2

Housekeeping

- Assignment 1: Marks and feedback are now available
- Assignment 3 will be available this Wed
- Please use Ed!

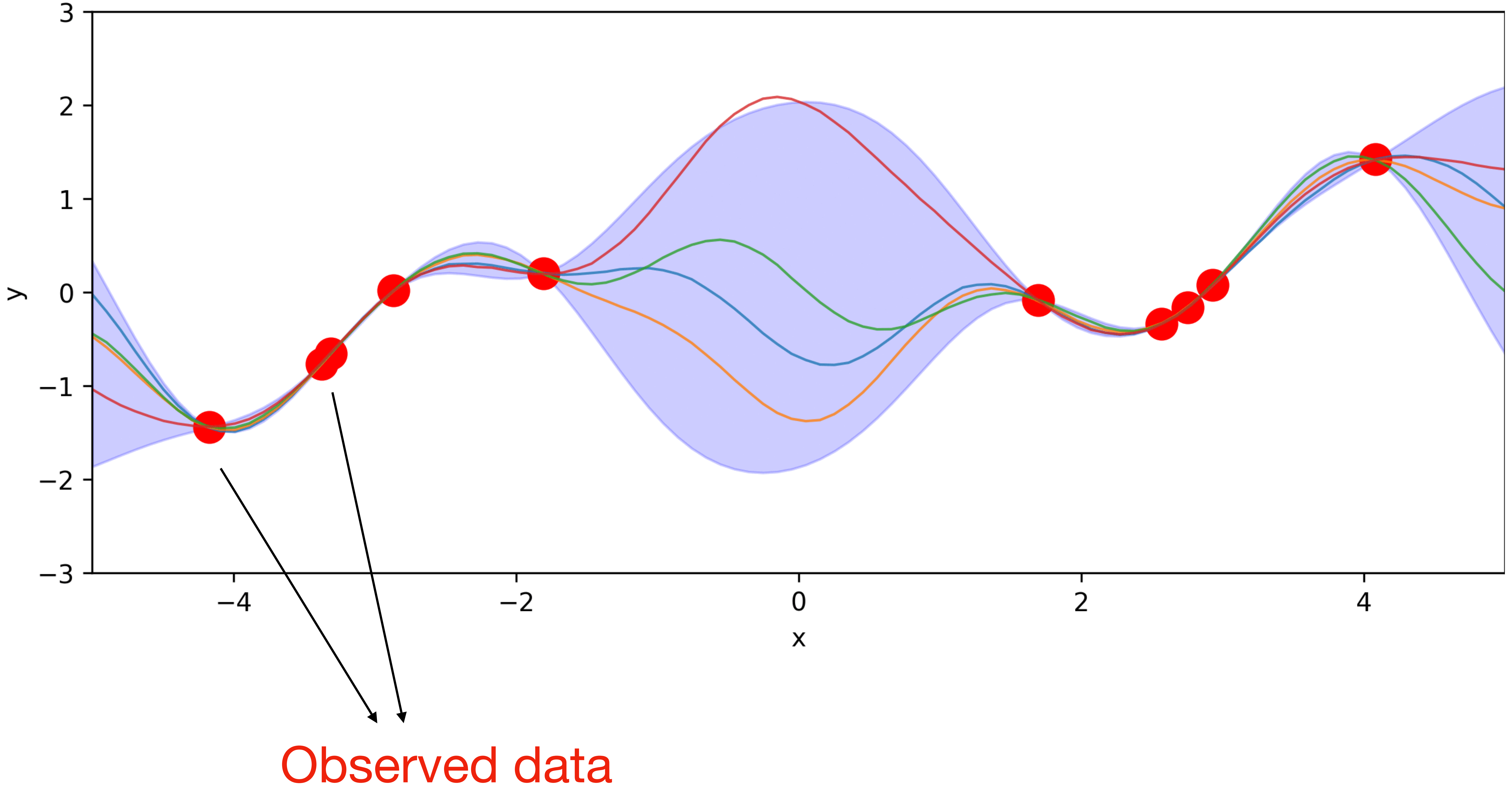
Foundations of ML



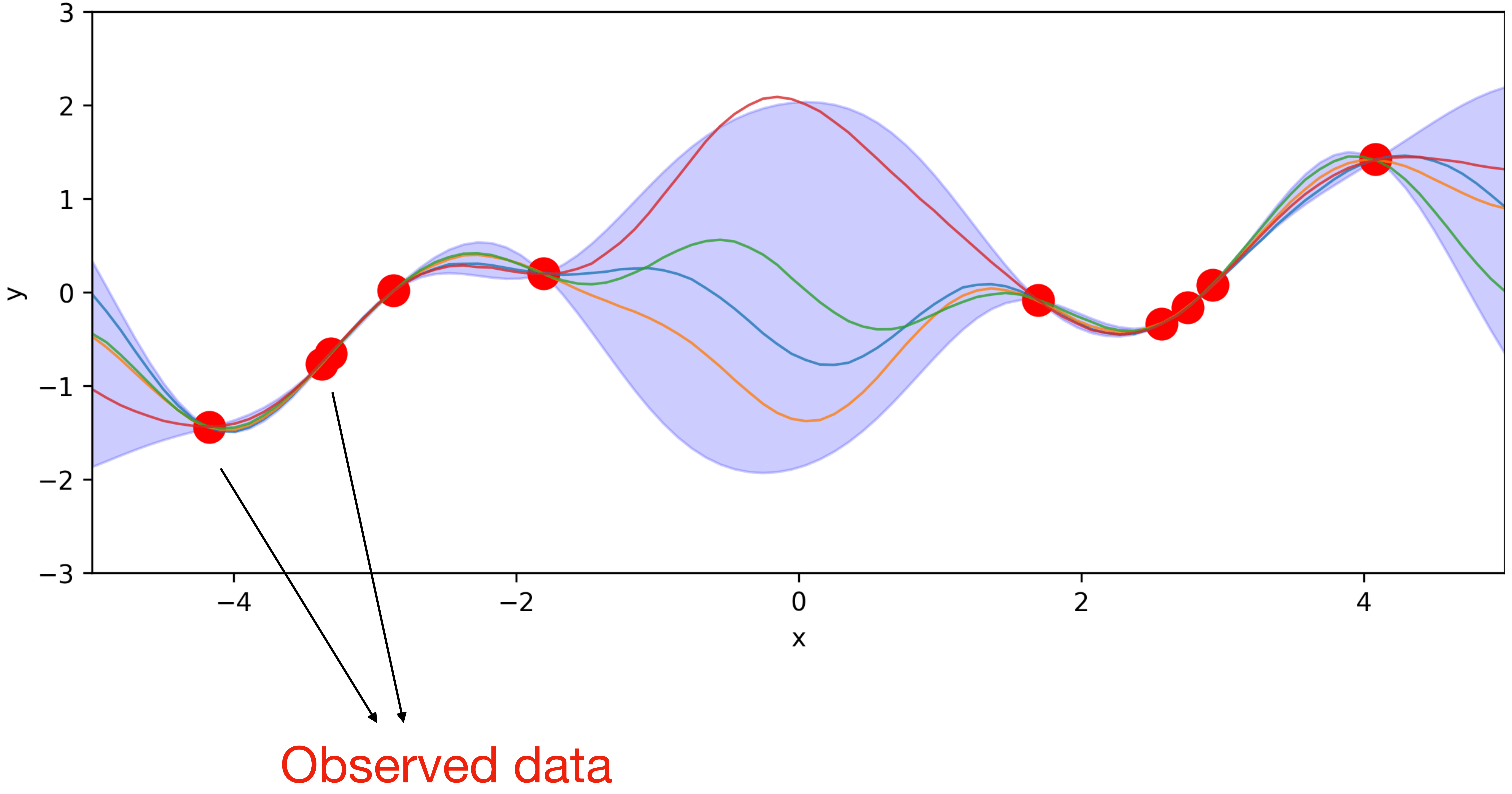
This week:
Linear regression

Source: Maths for ML Textbook

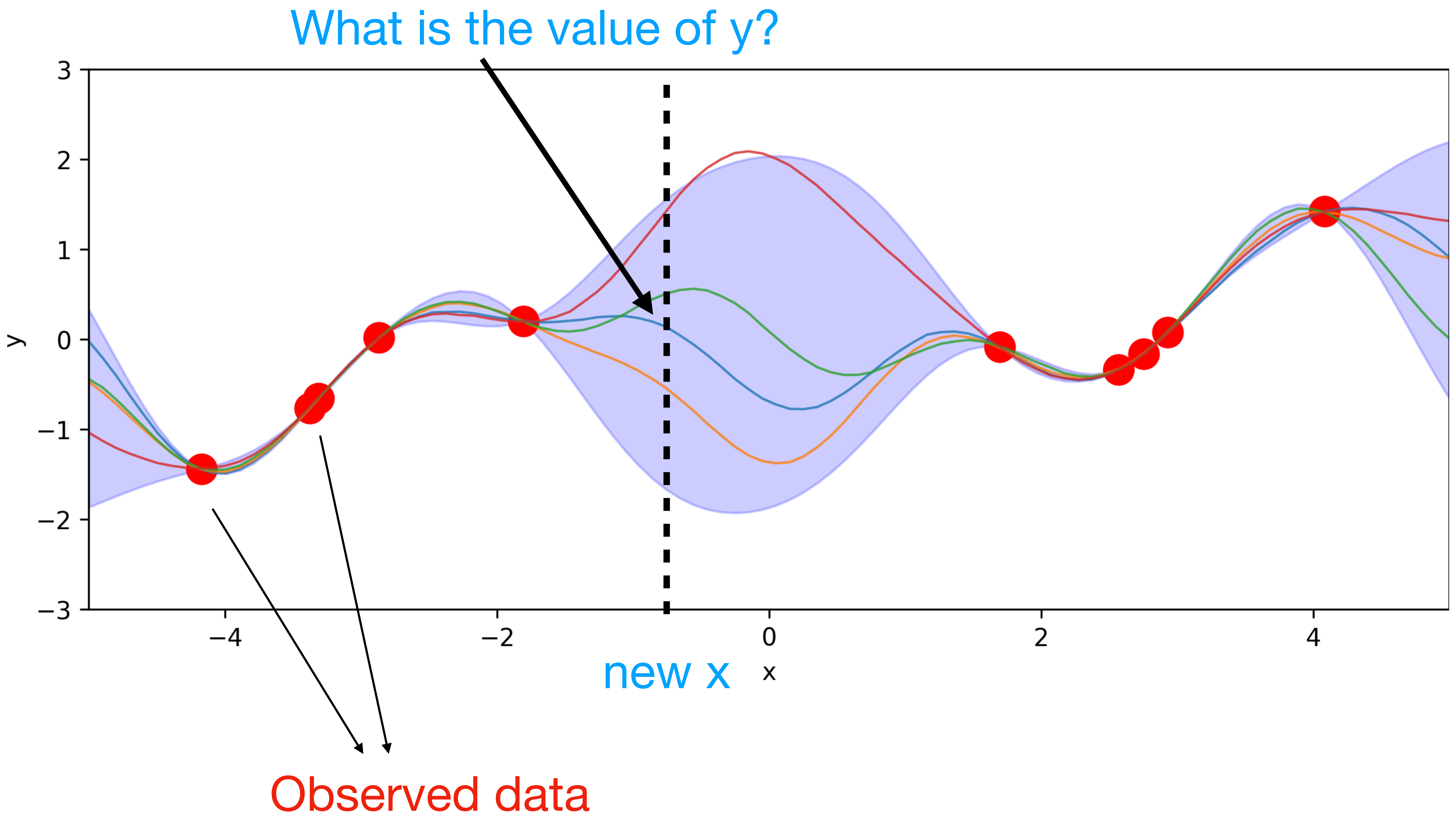
What is regression?



What is regression?



What is regression?



An example

An example

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

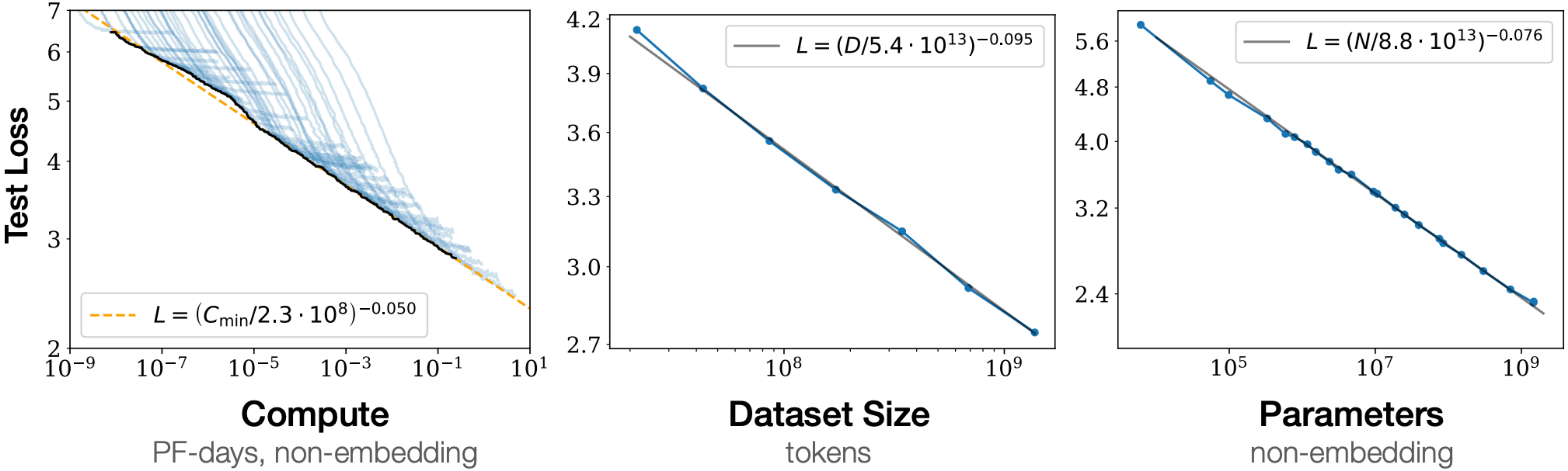


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

An example

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

How did they get these relationships?

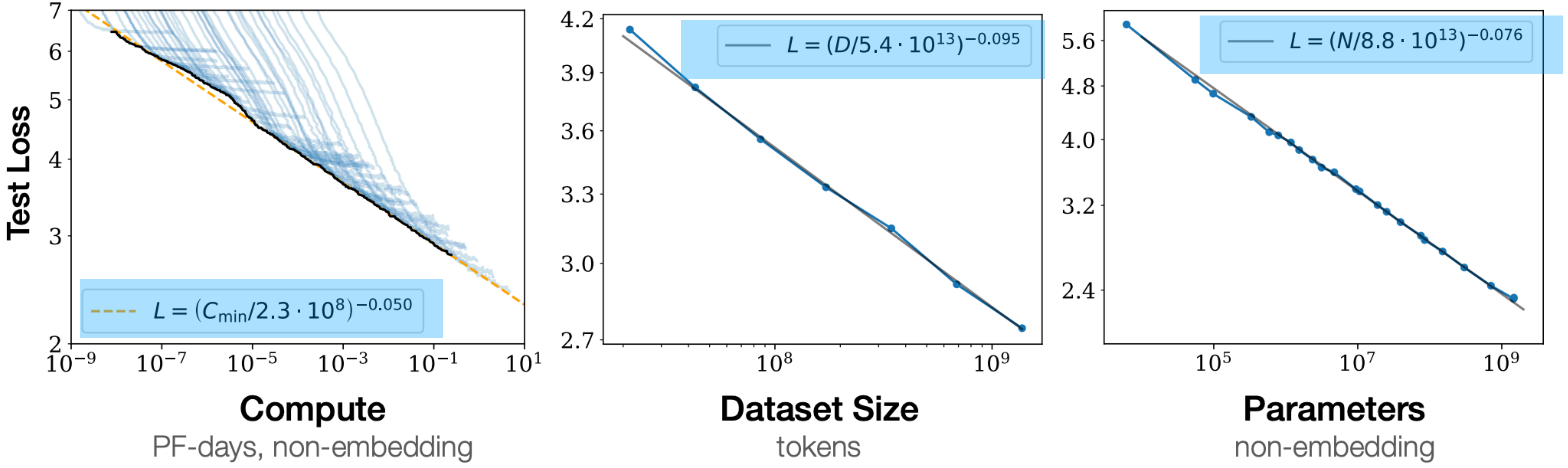


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

An example

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

How did they get these relationships?

Linear regression

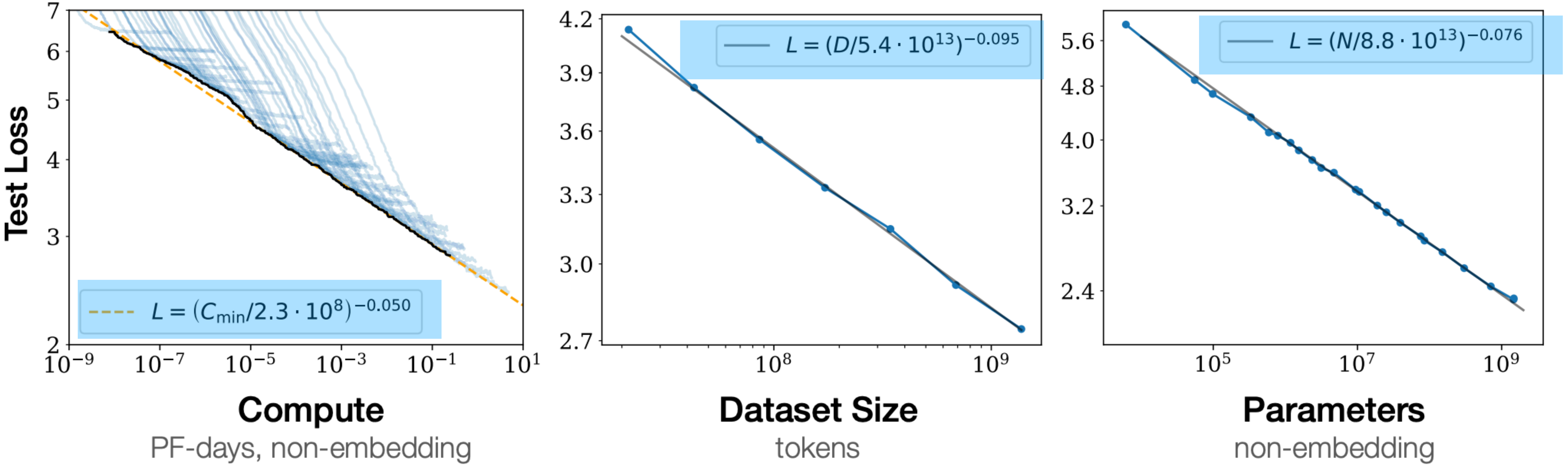
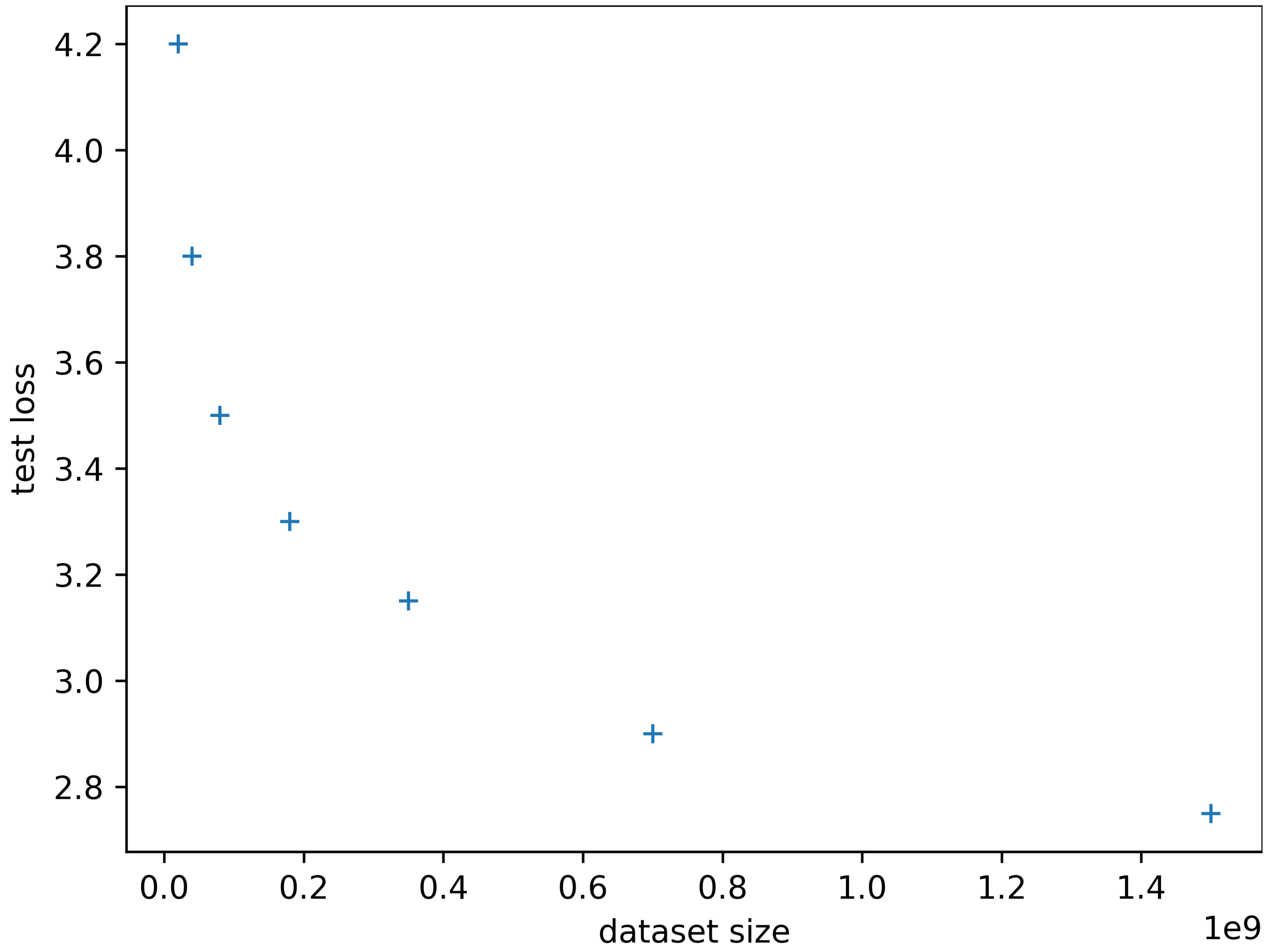


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

An example - Test loss vs data size [1]

Dataset	Test loss
2.0e+07	4.2
4.0e+07	3.8
8.0e+07	3.5
1.8e+08	3.3
3.5e+08	3.15
7.0e+08	2.90
1.5e+09	2.75

*

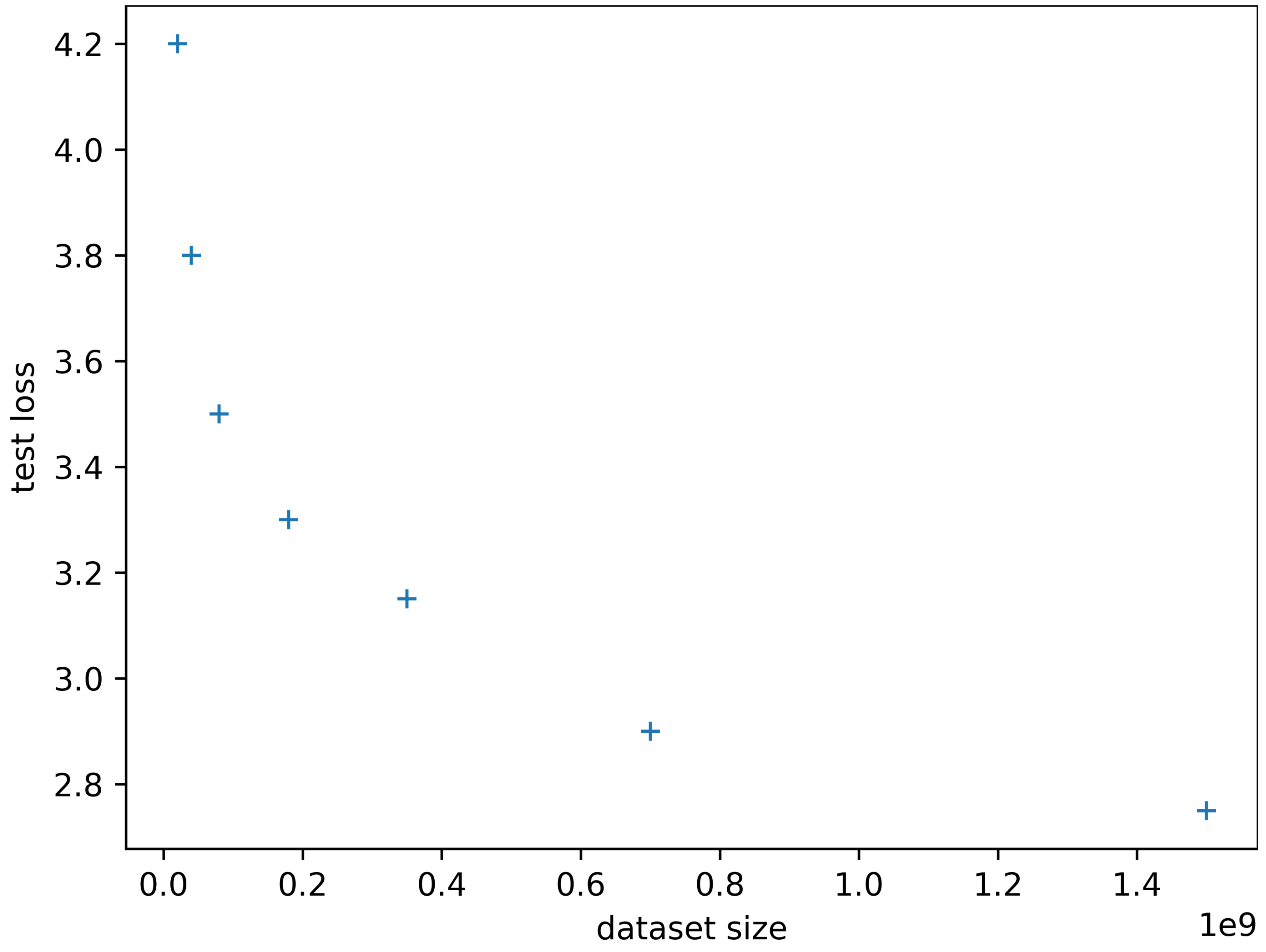


* I got these numbers by eyeballing the plot in the paper

An example - Test loss vs data size [1]

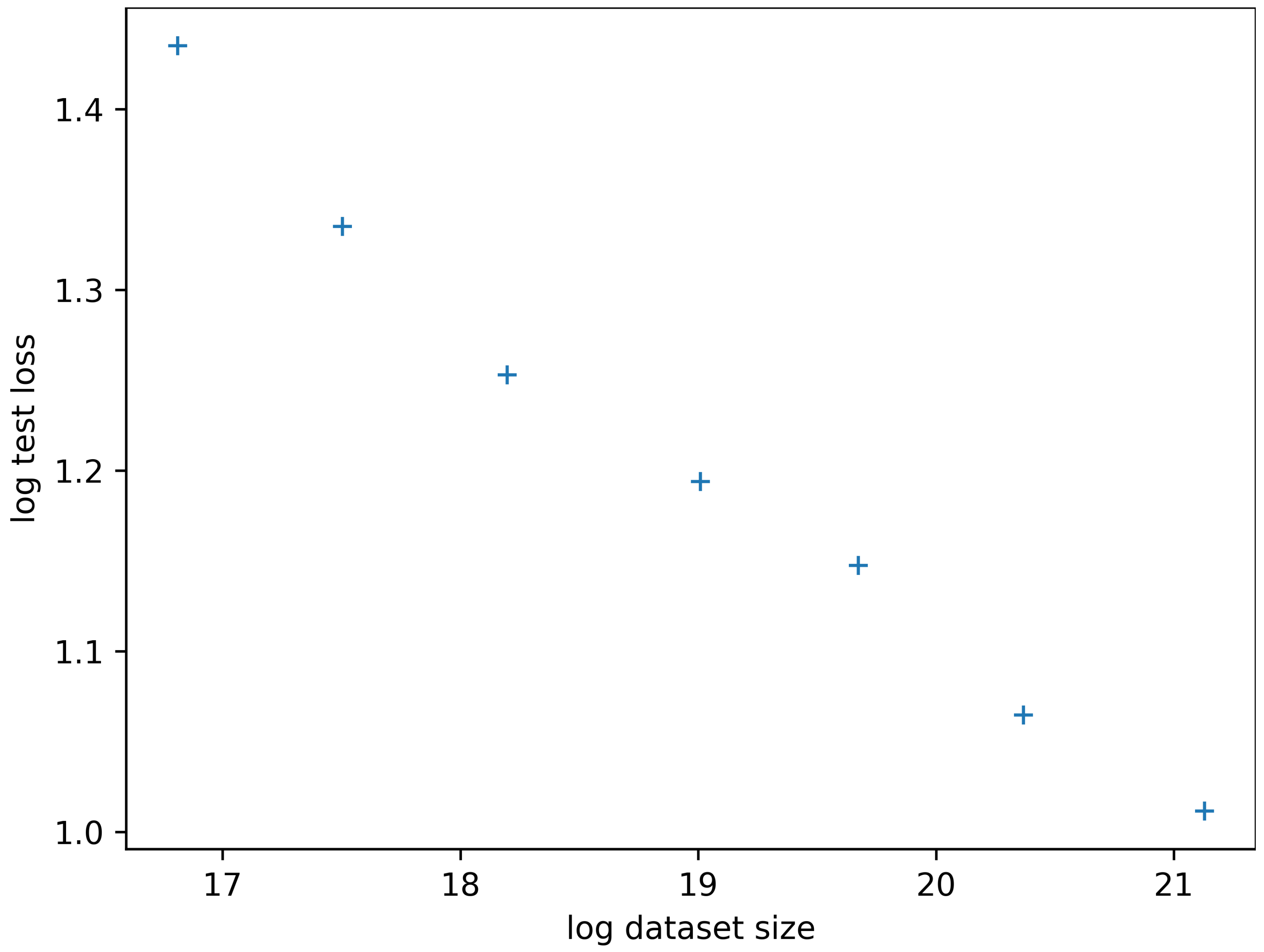
Dataset	Test loss
2.0e+07	4.2
4.0e+07	3.8
8.0e+07	3.5
1.8e+08	3.3
3.5e+08	3.15
7.0e+08	2.90
1.5e+09	2.75

*



Log transform

log(Dataset size)	log(Test loss)
16.8	1.44
17.5	1.34
18.2	1.25
19.0	1.19
19.7	1.14
20.4	1.06
21.1	1.01



* I got these numbers by eyeballing the plot in the paper

An example - Test loss vs data size [2]

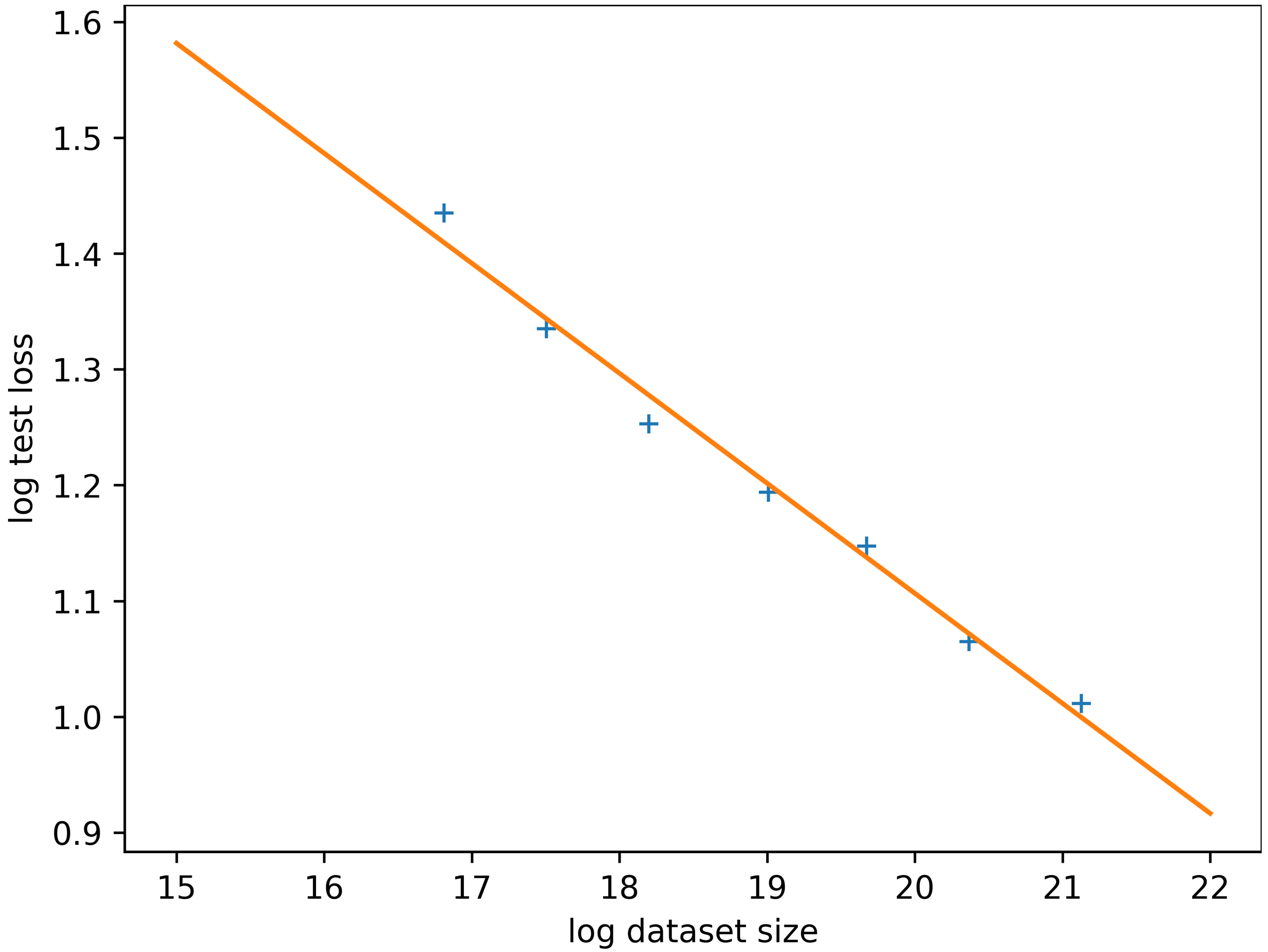
Let x = log(Dataset size), y = log(test loss). Assume $y \approx f(x) = ax + b$

Question: find a and b

Step 1: Write down an **objective function** *or* **goodness of fit**, which tell us how **good** the current a and b are

Step 2: **Optimise** this objective function

An example - Test loss vs data size [3]

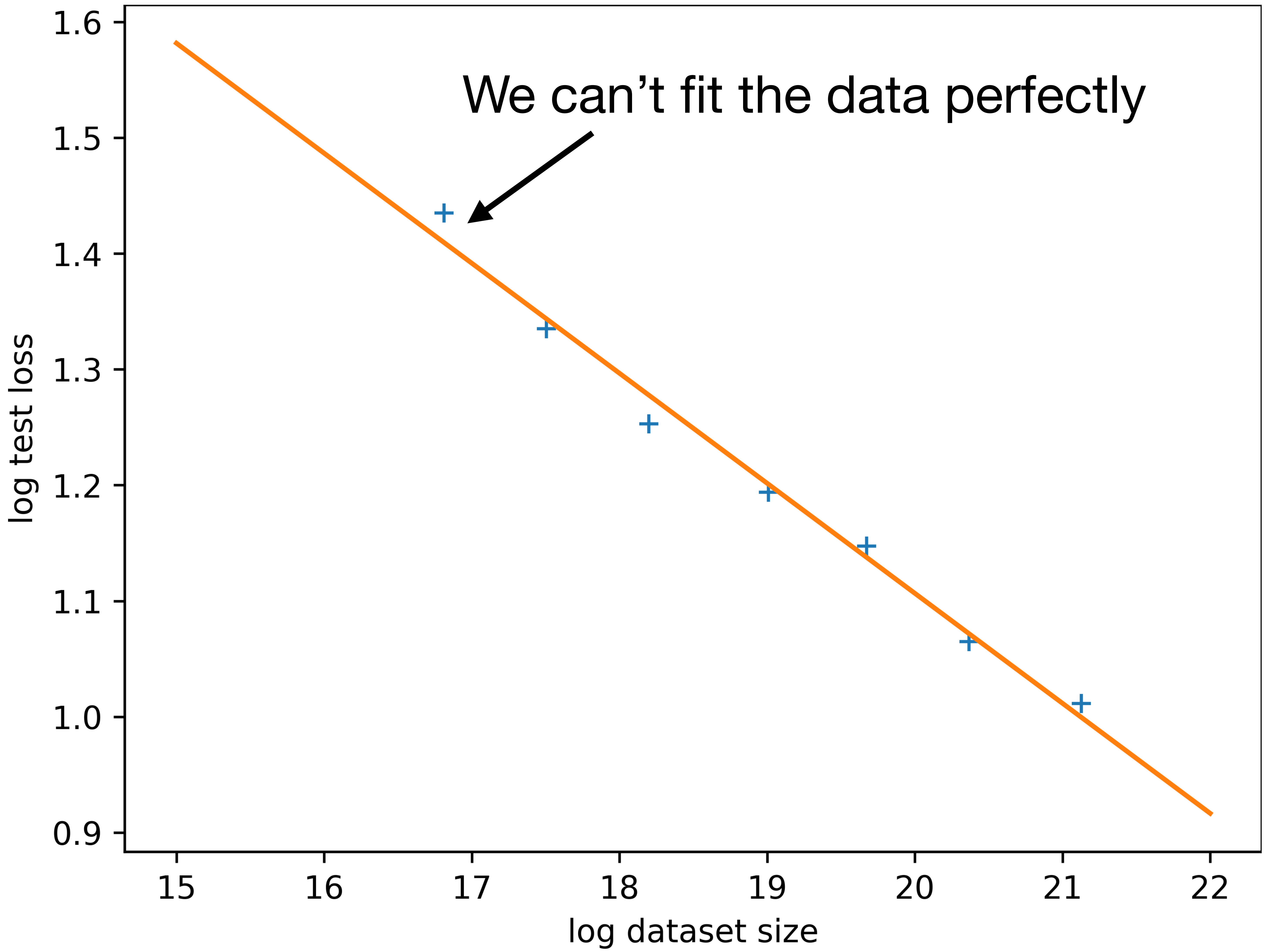



```
Sx = np.sum(x)
Sy = np.sum(y)
Sxy = np.sum(x * y)
Sx2 = np.sum(x**2)
N = x.shape[0]

a = (N * Sxy - Sx * Sy) / (N * Sx2 - Sx**2)
b = (Sy - a*Sx) / N

xpred = np.linspace(15, 22, 100)
fpred = a * xpred + b
```

$$y = -0.095x + 3$$

$$L = \left(\frac{D}{5 \times 10^{13}}\right)^{-0.095}$$

An example - Test loss vs data size [3]



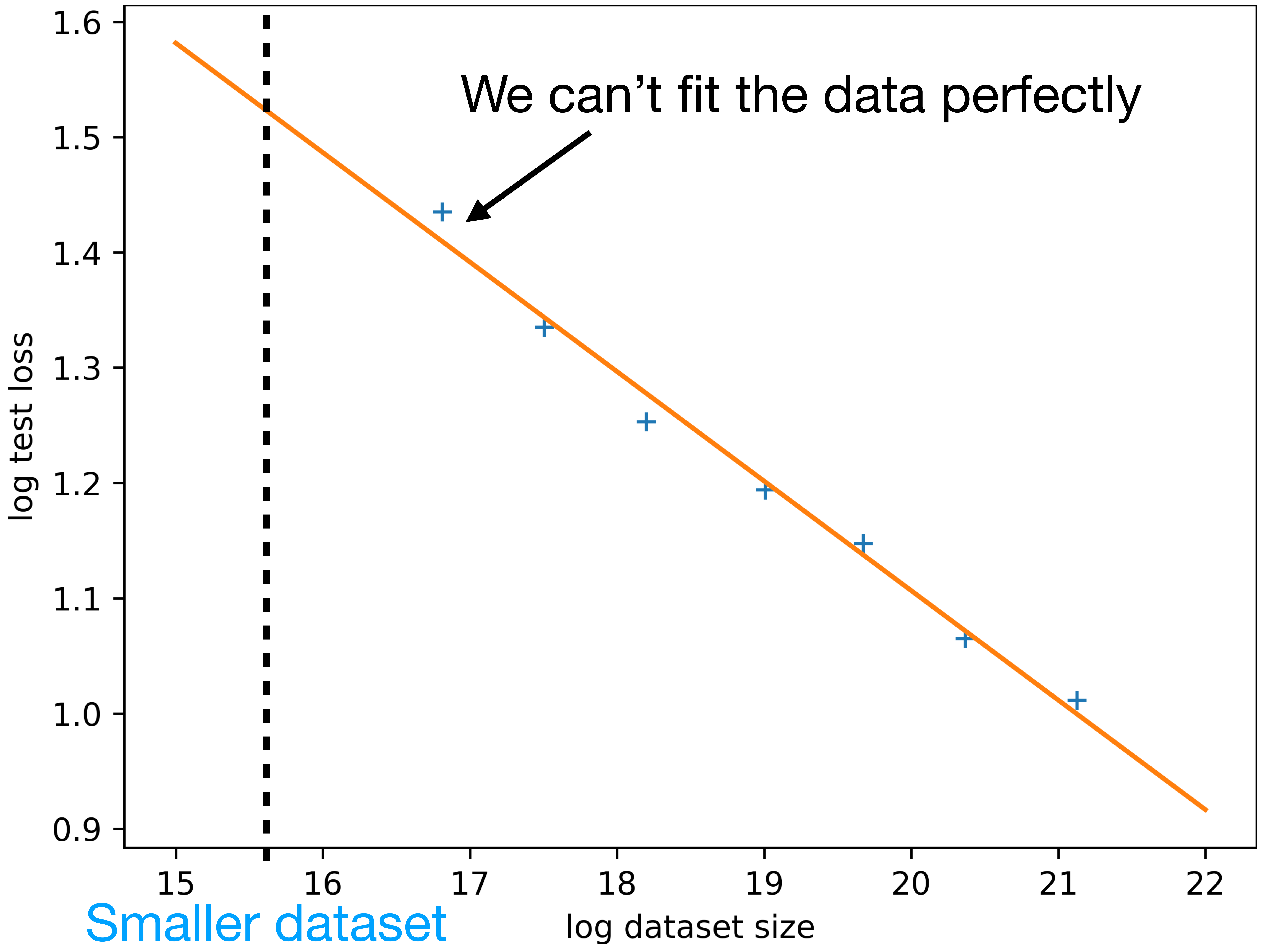

```
Sx = np.sum(x)
Sy = np.sum(y)
Sxy = np.sum(x * y)
Sx2 = np.sum(x**2)
N = x.shape[0]

a = (N * Sxy - Sx * Sy) / (N * Sx2 - Sx**2)
b = (Sy - a*Sx) / N

xpred = np.linspace(15, 22, 100)
fpred = a * xpred + b
```

$$y = -0.095x + 3$$

$$L = \left(\frac{D}{5 \times 10^{13}}\right)^{-0.095}$$

An example - Test loss vs data size [3]

What is the test performance?



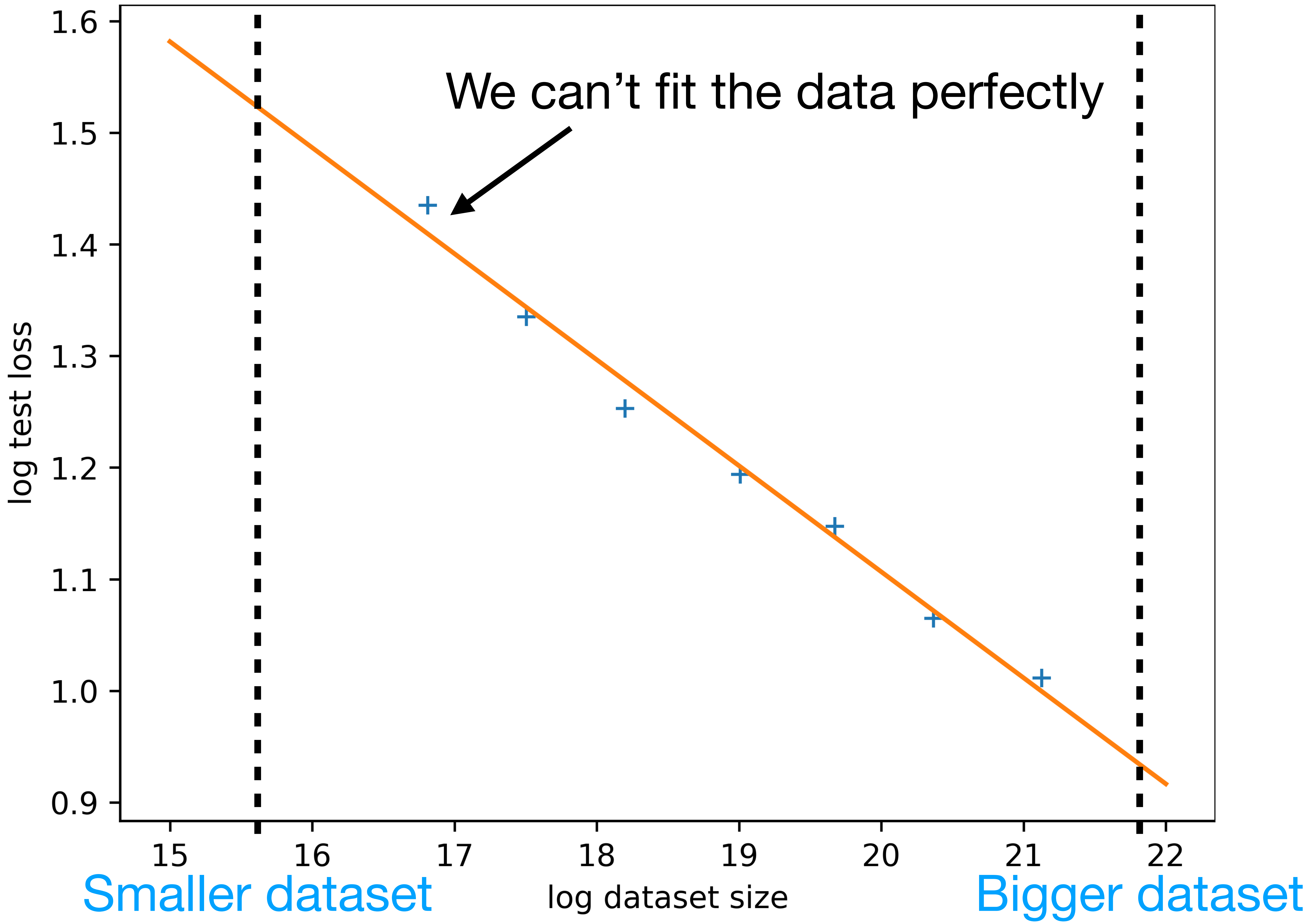

```
Sx = np.sum(x)
Sy = np.sum(y)
Sxy = np.sum(x * y)
Sx2 = np.sum(x**2)
N = x.shape[0]

a = (N * Sxy - Sx * Sy) / (N * Sx2 - Sx**2)
b = (Sy - a*Sx) / N

xpred = np.linspace(15, 22, 100)
fpred = a * xpred + b
```

$$y = -0.095x + 3$$

$$L = \left(\frac{D}{5 \times 10^{13}}\right)^{-0.095}$$

An example - Test loss vs data size [3]

What is the test performance? What is the test performance?




```python
Sx = np.sum(x)
Sy = np.sum(y)
Sxy = np.sum(x * y)
Sx2 = np.sum(x**2)
N = x.shape[0]

a = (N * Sxy - Sx * Sy) / (N * Sx2 - Sx**2)
b = (Sy - a*Sx) / N

xpred = np.linspace(15, 22, 100)
fpred = a * xpred + b
```

$$y = -0.095x + 3$$

$$L = \left(\frac{D}{5 \times 10^{13}}\right)^{-0.095}$$

Overview

Formalise the problem, extend to *multiple* input dimensions, aka *vectorise*

How to handle non-linear *features*

How to control *overfitting* by *regularisation*

Discuss *equivalent* views: least squares = maximum likelihood, regularised least square = maximum a-posteriori (MAP). Why do we care -> *Bayesian* linear regression!

Numerical *issues*, computational *complexity,* and *workarounds*

When to use *numerical optimisation* instead, and how

Data and linear assumption

Data and linear assumption

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Each row is a data point

Each column is a feature dimension

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

One scalar per data point

Data and linear assumption

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Each row is a data point

Each column is a feature dimension

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

One scalar per data point

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Data and linear assumption

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Each row is a data point

Each column is a feature dimension

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

One scalar per data point

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Test time: given a new input $\mathbf{x}^*$, prediction = $f(\mathbf{x}^*) = \theta^\intercal \mathbf{x}^*$

Data and linear assumption

Training data N input, output pairs $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Each row is a data point

Each column is a feature dimension

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

One scalar per data point

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Test time: given a new input $\mathbf{x}^*$, prediction = $f(\mathbf{x}^*) = \theta^\intercal \mathbf{x}^*$

Question: where is the bias/intercept in this formulation?

Objective function

Objective function

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \approx \quad \begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Objective function

Training data N input, output pairs $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \approx \quad \begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Desideratum: y is well approximated by $f_\theta(\mathbf{x}) = \theta^\intercal \mathbf{x}$

Objective function

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \approx \quad \begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Desideratum: y is well approximated by $f_\theta(\mathbf{x}) = \theta^\intercal \mathbf{x}$

Objective function measures the approximation quality. Several options (all smaller is better):

- Raw difference, $y - f_\theta(\mathbf{x})$. What can go wrong?
- Absolute difference, $|y - f_\theta(\mathbf{x})|$. What is the problem here?
- Squared difference, $(y - f_\theta(\mathbf{x}))^2$. Any issue here?

Objective function

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \approx \quad \begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Desideratum: y is well approximated by $f_\theta(\mathbf{x}) = \theta^\intercal \mathbf{x}$

Objective function measures the approximation quality. Several options (all smaller is better):

- Raw difference, $y - f_\theta(\mathbf{x})$. What can go wrong?
- Absolute difference, $|y - f_\theta(\mathbf{x})|$. What is the problem here?
- Squared difference, $(y - f_\theta(\mathbf{x}))^2$. Any issue here?

We will use the squared difference, also called L2 loss or squared loss or squared error.

Least squares for linear regression

Loss function: $L(\theta) = \frac{1}{N}\sum_{n=1}^{N}(y_n - f_\theta(\mathbf{x}_n))^2 = \frac{1}{N}||\mathbf{y} - X\theta||_2^2 = \frac{1}{N}(\mathbf{y} - X\theta)^\intercal(\mathbf{y} - X\theta)$

Least squares for linear regression

Loss function: $L(\theta) = \frac{1}{N}\sum_{n=1}^{N}(y_n - f_\theta(\mathbf{x}_n))^2 = \frac{1}{N}||\mathbf{y} - X\theta||_2^2 = \frac{1}{N}(\mathbf{y} - X\theta)^\intercal(\mathbf{y} - X\theta)$

We want to find θ that minimises the loss function. Closed-form analytic solution!

$$\theta = (X^\intercal X)^{-1}X^\intercal\mathbf{y}$$

Let's derive this!

Overview

Formalise the problem, extend to *multiple* input dimensions, aka *vectorise*

How to handle non-linear *features*

How to control *overfitting* by *regularisation*

Discuss *equivalent* views: least squares = maximum likelihood, regularised least square = maximum a-posteriori (MAP). Why do we care -> *Bayesian* linear regression!

Numerical *issues*, computational *complexity,* and *workarounds*

When to use *numerical optimisation* instead, and how

Linear regression with features

So far, we have discussed linear regression which fits straight lines to data. Fortunately, "linear regression" only refers to "**linear** in the **parameters**".

We can perform an arbitrary nonlinear transformation $\phi(\mathbf{x})$ of the inputs $\mathbf{x}$ and then linearly combine the components. That is, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d \phi(x)_d = \theta^\intercal \phi(\mathbf{x}), \theta \in \mathbb{R}^D$

Linear regression with features

So far, we have discussed linear regression which fits straight lines to data. Fortunately, "linear regression" only refers to "**linear** in the **parameters**".

We can perform an arbitrary nonlinear transformation $\phi(\mathbf{x})$ of the inputs $\mathbf{x}$ and then linearly combine the components. That is, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d \phi(x)_d = \theta^\intercal \phi(\mathbf{x}), \theta \in \mathbb{R}^D$

Loss function: $L(\theta) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f_\theta(\mathbf{x}_n))^2 = \frac{1}{N} \|\mathbf{y} - \Phi\theta\|_2^2 = \frac{1}{N} (\mathbf{y} - \Phi\theta)^\intercal (\mathbf{y} - \Phi\theta)$

Each row is a data point

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_D(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_D(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_D(x_N) \end{bmatrix}$$

Each col corresponds to a feature mapping

Linear regression with features

So far, we have discussed linear regression which fits straight lines to data. Fortunately, “linear regression” only refers to “**linear** in the **parameters**”.

We can perform an arbitrary nonlinear transformation $\phi(\mathbf{x})$ of the inputs $\mathbf{x}$ and then linearly combine the components. That is, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d \phi(x)_d = \theta^\intercal \phi(\mathbf{x}), \theta \in \mathbb{R}^D$

Loss function: $L(\theta) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f_\theta(\mathbf{x}_n))^2 = \frac{1}{N} \|\mathbf{y} - \Phi\theta\|_2^2 = \frac{1}{N} (\mathbf{y} - \Phi\theta)^\intercal (\mathbf{y} - \Phi\theta)$

Each row is a data point

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_D(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_D(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_D(x_N) \end{bmatrix}$$

Each col corresponds to a feature mapping

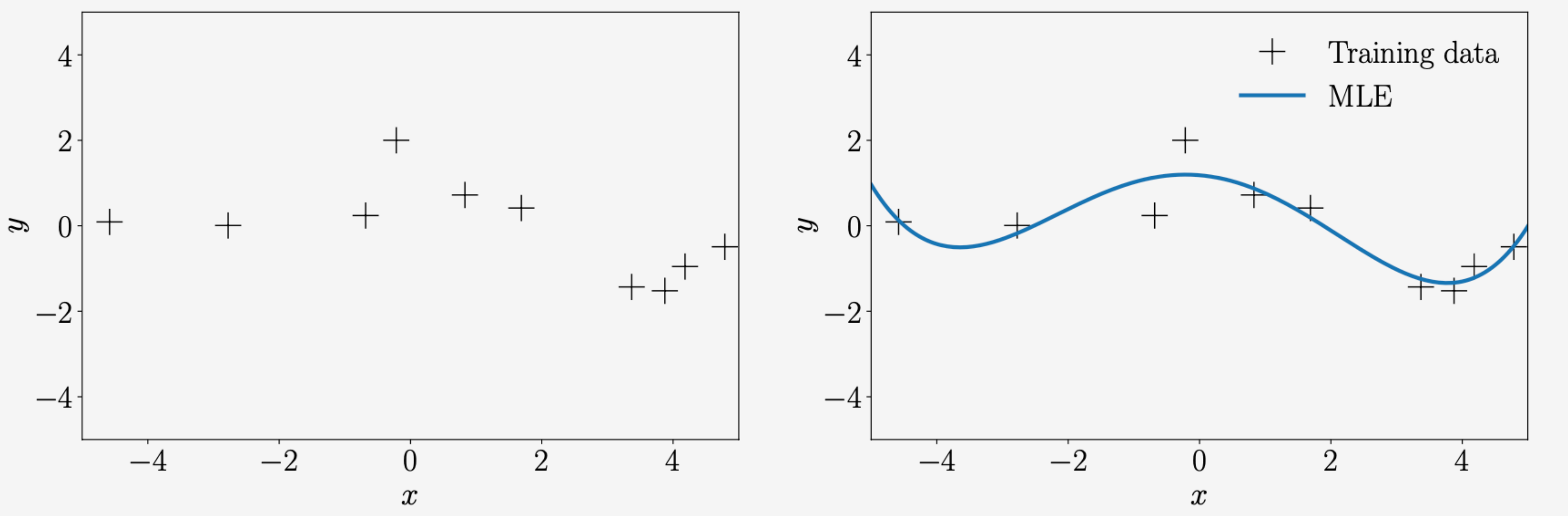
Closed-form analytic solution: $\theta = (\Phi^\intercal \Phi)^{-1} \Phi^\intercal \mathbf{y}$

Linear regression with features - examples

Features = anything you want the underlying function to encode, e.g. square, cubic, sin…

Example: second-order polynomial regression $\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$

Example: fourth-order polynomial regression, features = $1, x, x^2, x^3, x^4$



(a) Regression dataset.

(b) Polynomial of degree 4 determined by maximum likelihood estimation.

Overview

Formalise the problem, extend to *multiple* input dimensions, aka *vectorise*

How to handle non-linear *features*

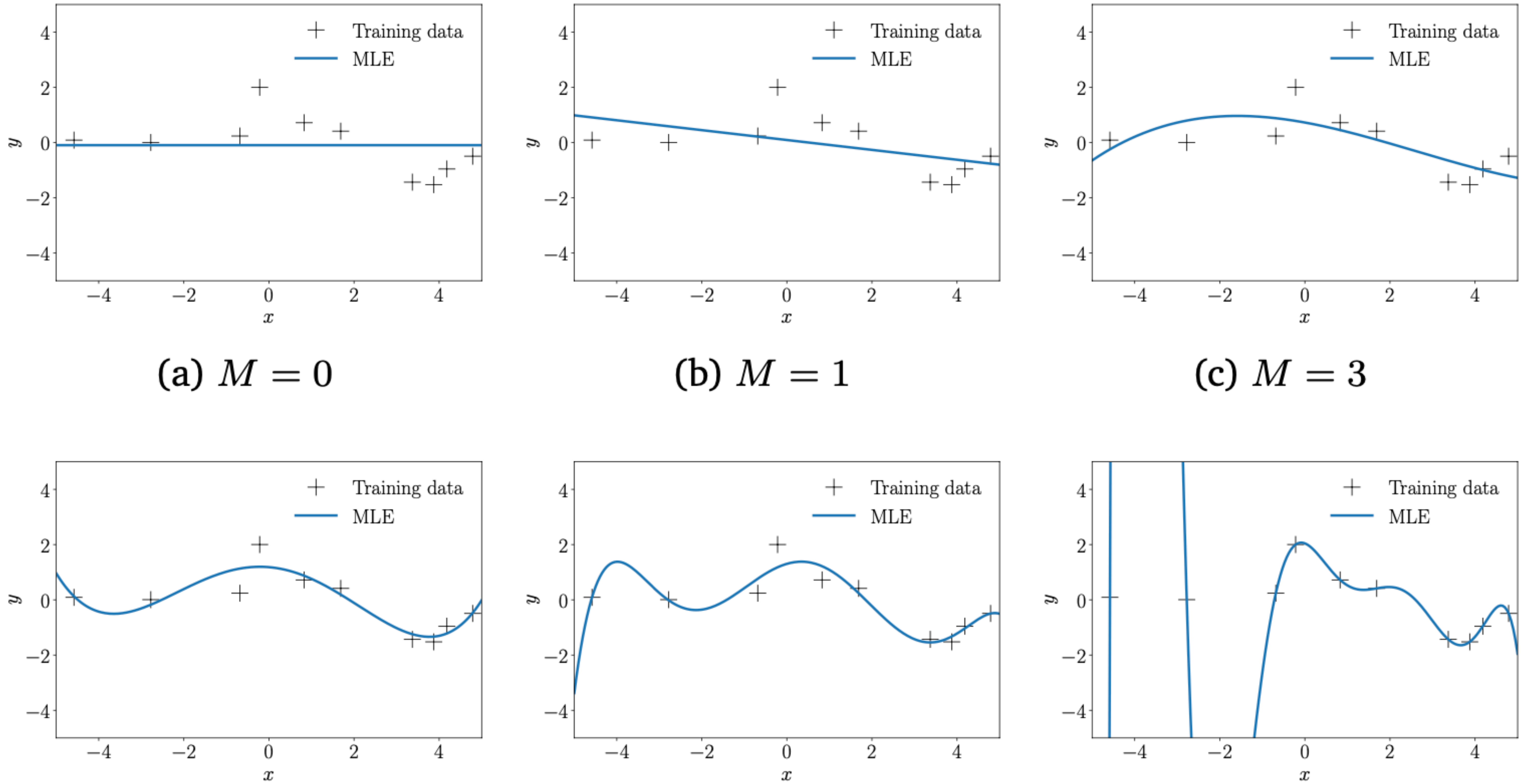
How to control *overfitting* by *regularisation*

Discuss *equivalent* views: least squares = maximum likelihood, regularised least square = maximum a-posteriori (MAP). Why do we care -> *Bayesian* linear regression!

Numerical *issues*, computational *complexity,* and *workarounds*

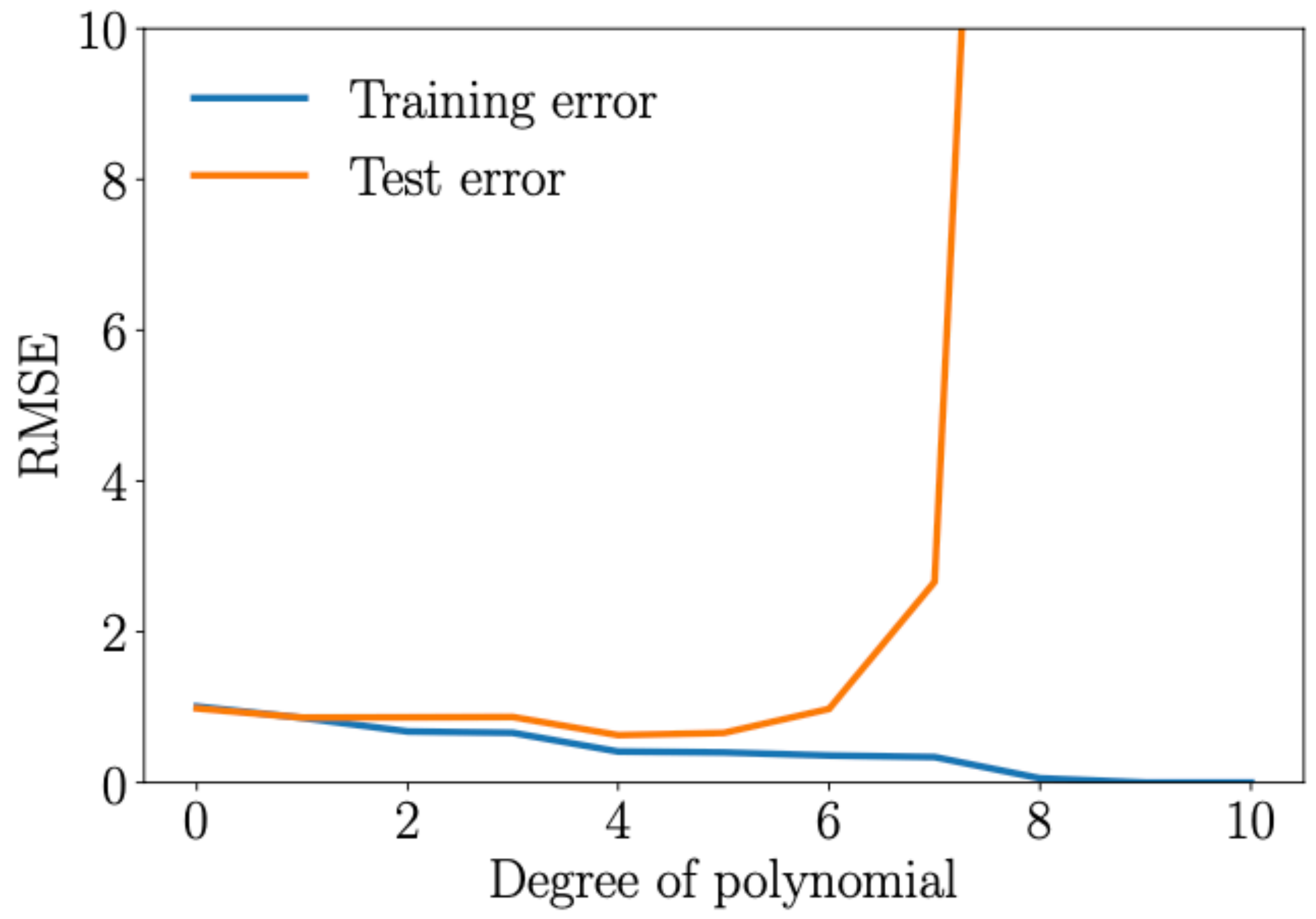
When to use *numerical optimisation* instead, and how

Polynomial regression example - underfitting and overfitting

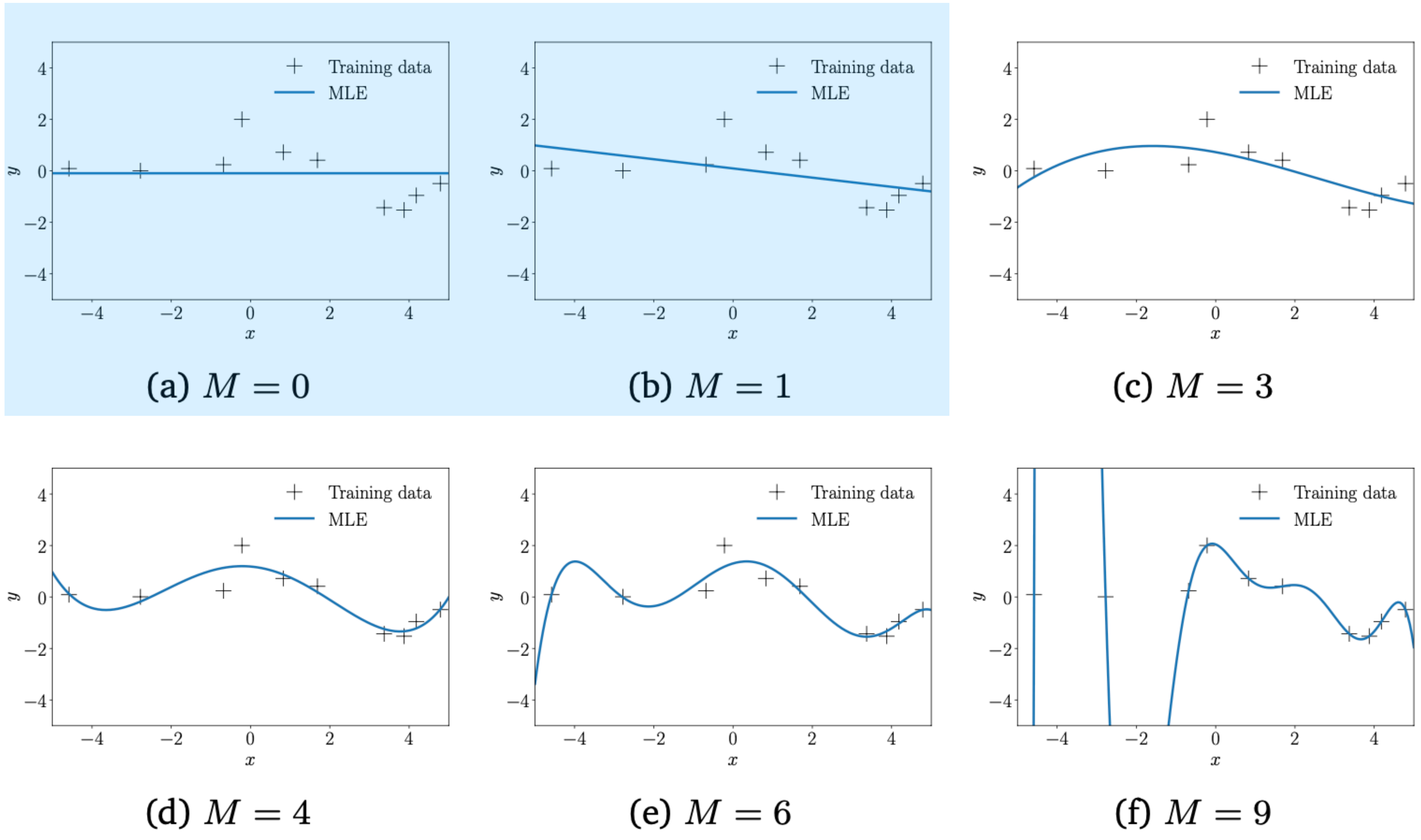


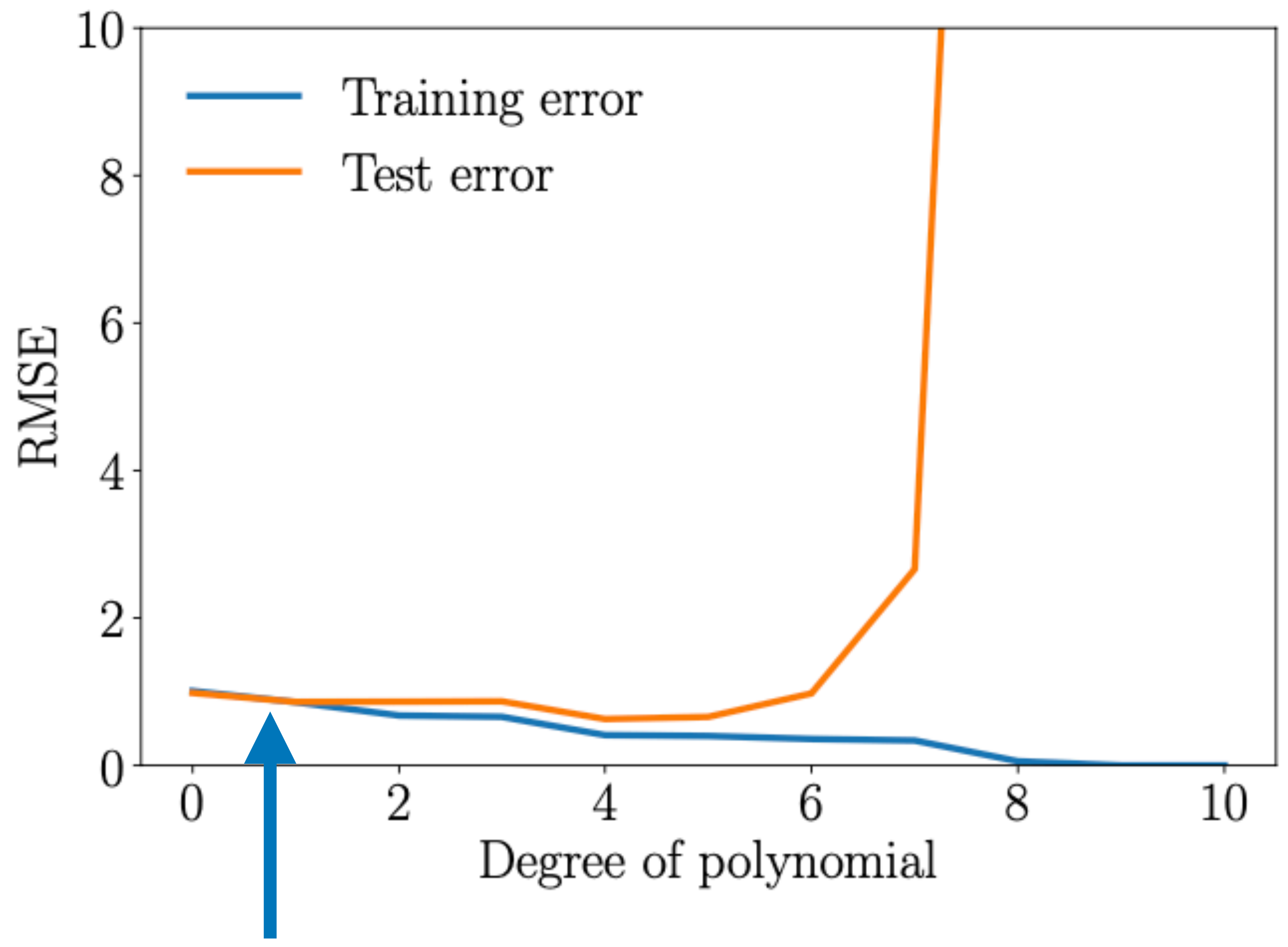
(a) $M = 0$ (b) $M = 1$ (c) $M = 3$

(d) $M = 4$ (e) $M = 6$ (f) $M = 9$



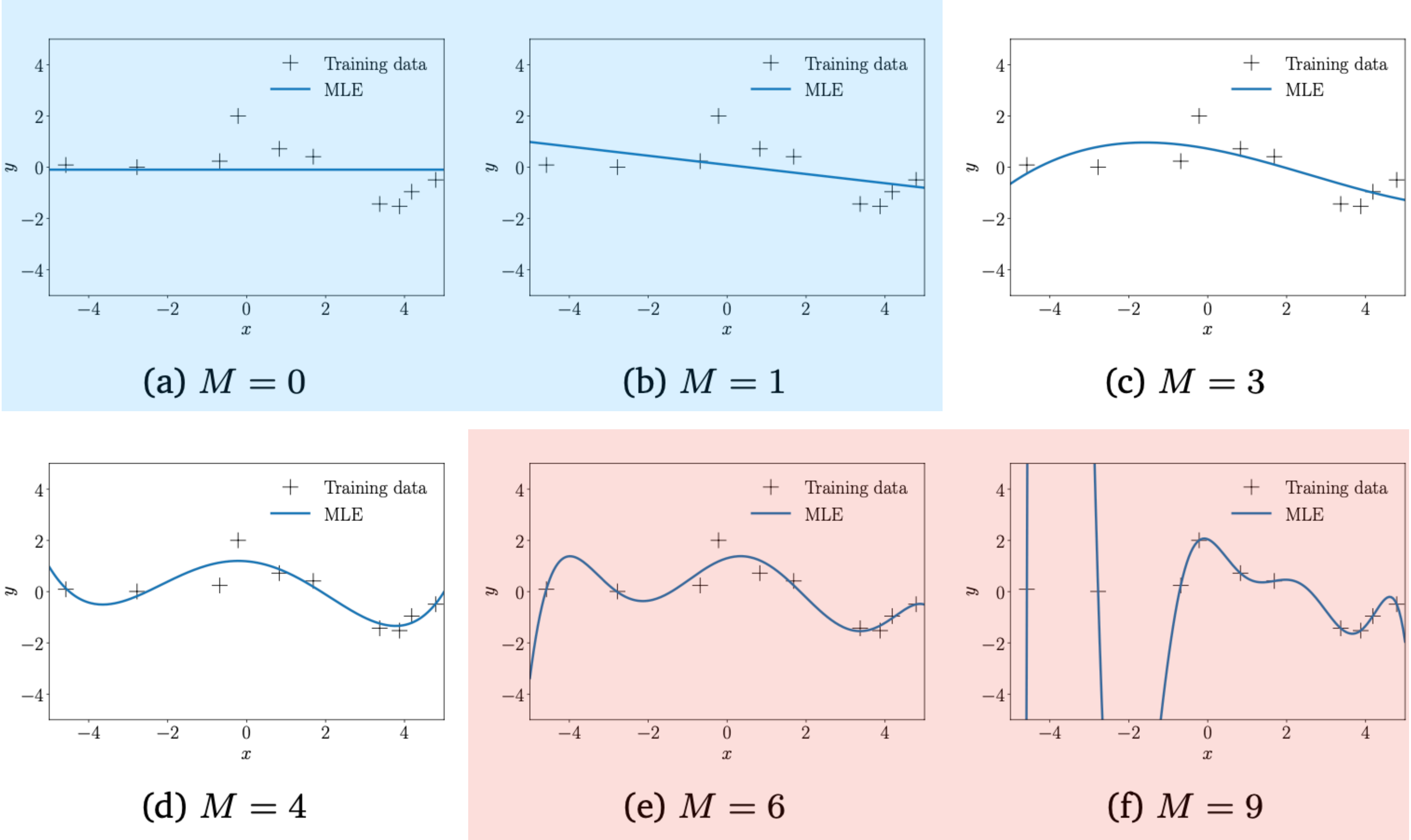
Polynomial regression example - underfitting and overfitting





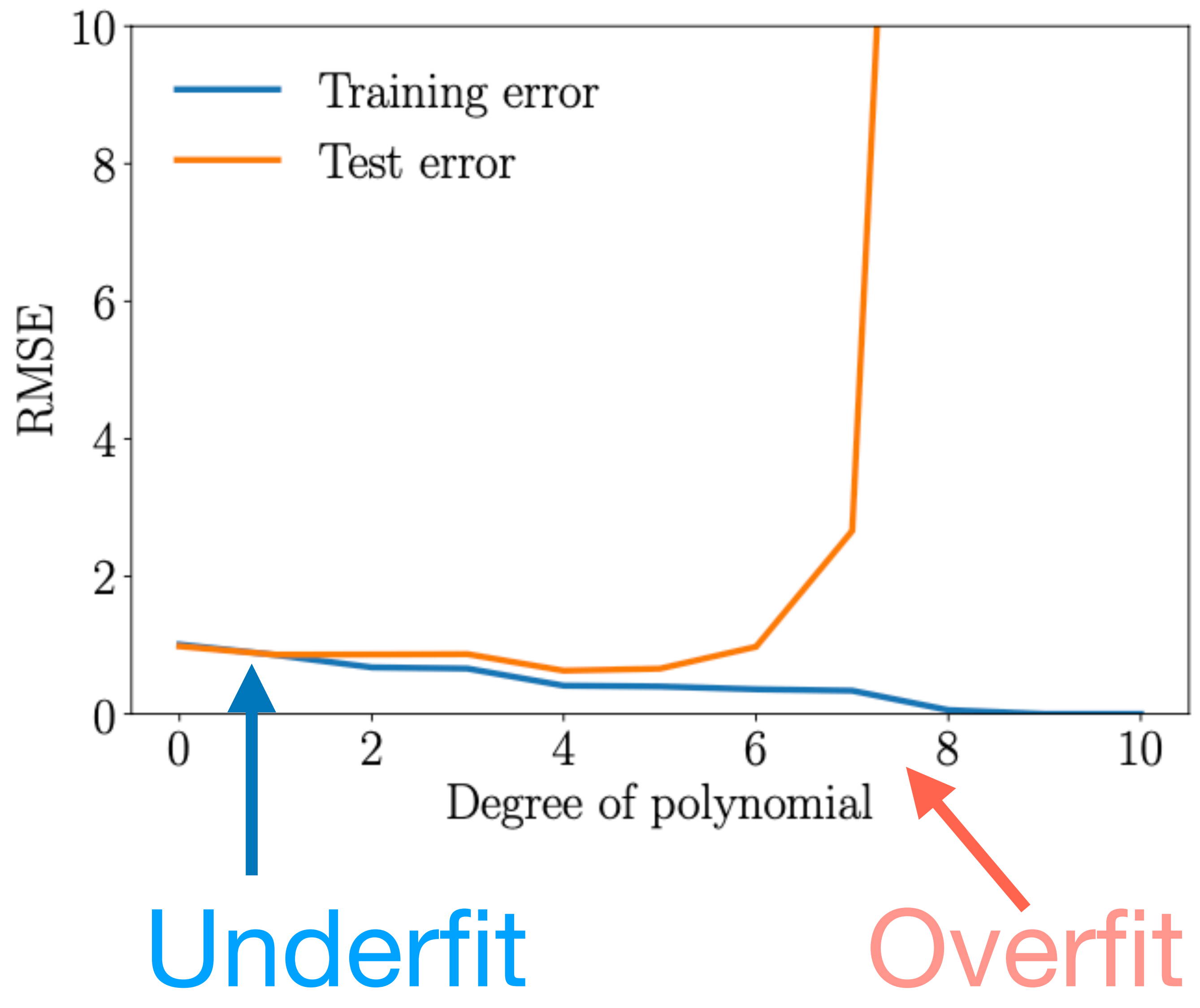
Underfit

Polynomial regression example - underfitting and overfitting

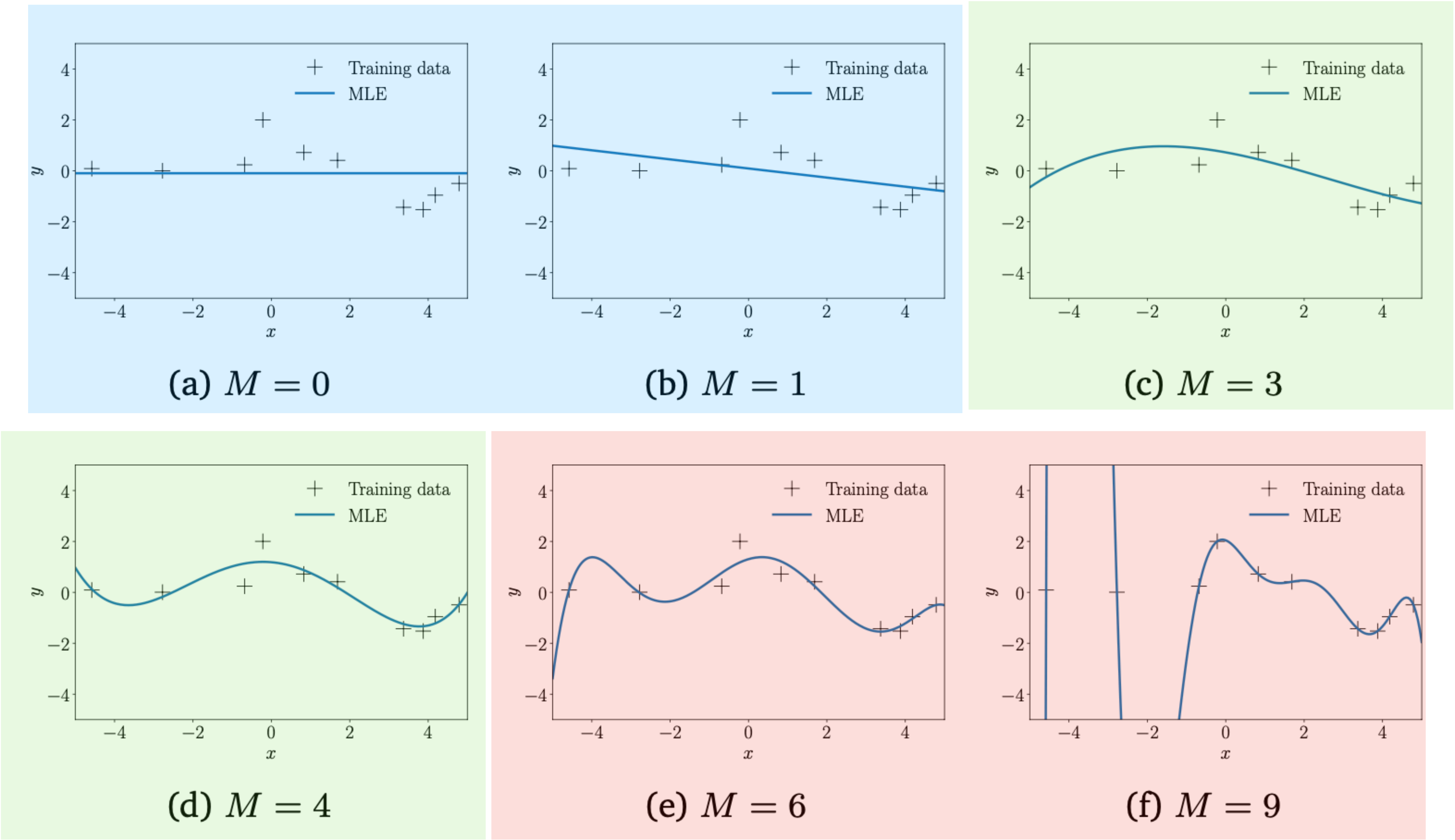


(a) $M = 0$ (b) $M = 1$ (c) $M = 3$

(d) $M = 4$ (e) $M = 6$ (f) $M = 9$

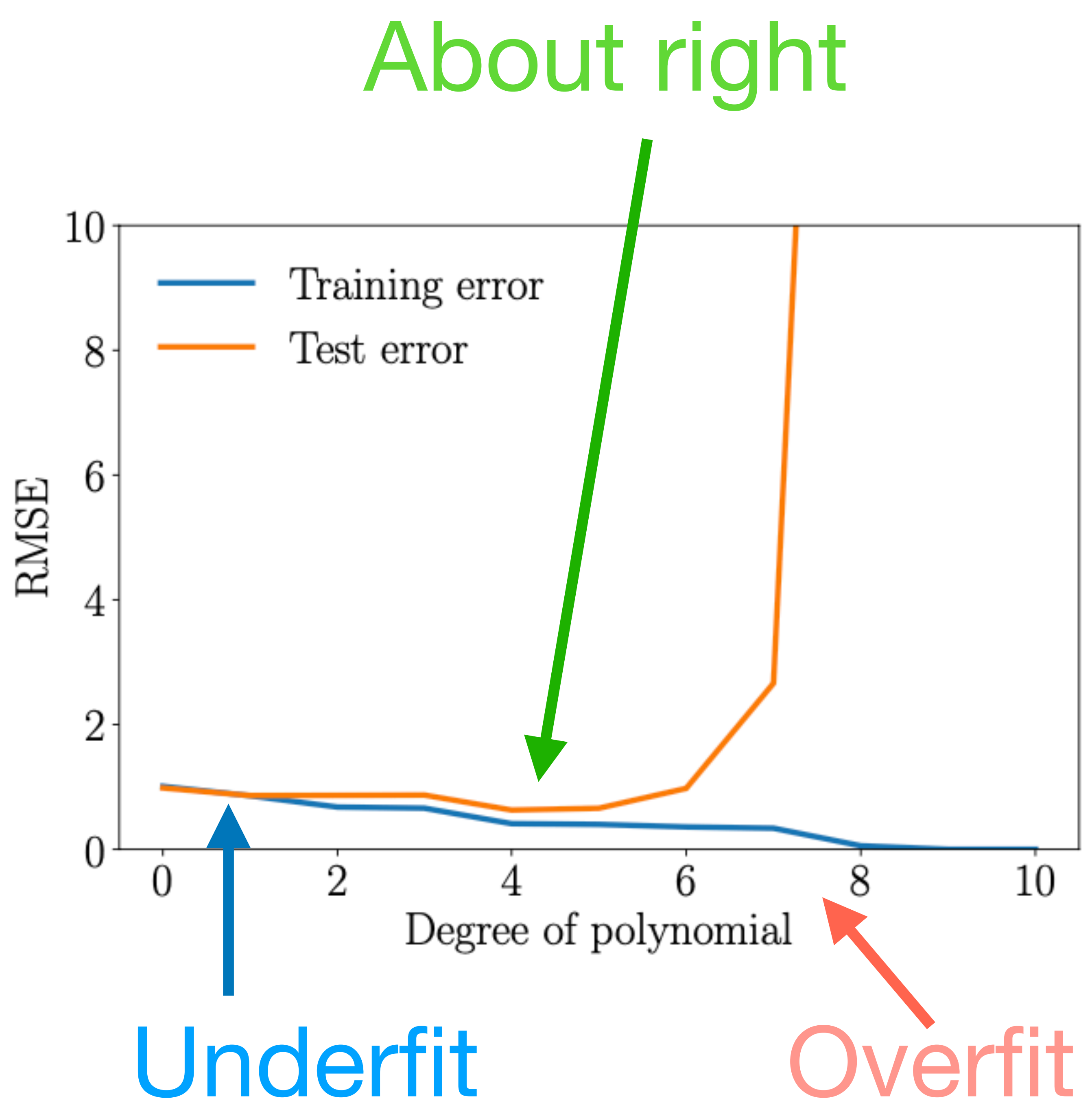


Polynomial regression example - underfitting and overfitting



(a) $M = 0$ (b) $M = 1$ (c) $M = 3$

(d) $M = 4$ (e) $M = 6$ (f) $M = 9$



Regularised least squares

Overfitting occurs because the model is too **complex** (θ has too many large entries), while there are too limited training sample.

We want to *penalise* the amplitude of parameters by **regularisation.**

Regularised least squares = least squares + regularisation

$$L_\lambda(\theta) = \underbrace{\frac{1}{N}\sum_{n=1}^{N}(y_n - f_\theta(\mathbf{x}_n))^2}_{\text{Data-fit}} + \underbrace{\lambda \|\theta\|_p^p}_{\text{Regulariser}}$$

Data-fit
Lower = better fit

Regulariser
Lower = simpler model

Hyperparameter

We can use any p-norm $\|\cdot\|_p$. Smaller p leads to sparser solutions, i.e., many parameter values $\theta_d = 0$. We will use p = 2 for this lecture (course).

Regularised least squares - analytic solution

Loss function: $L(\theta) = \dfrac{1}{N}||\mathbf{y} - X\theta||_2^2 + \ \lambda||\theta||_2^2 = \dfrac{1}{N}(\mathbf{y} - X\theta)^\intercal(\mathbf{y} - X\theta) + \lambda||\theta||_2^2$

Regularised least squares - analytic solution

Loss function: $L(\theta) = \frac{1}{N}||\mathbf{y} - X\theta||_2^2 + \ \lambda||\theta||_2^2 = \frac{1}{N}(\mathbf{y} - X\theta)^\intercal(\mathbf{y} - X\theta) + \lambda||\theta||_2^2$

We want to find θ that minimises the loss function. Closed-form analytic solution!

$$\theta = (X^\intercal X + N\lambda \mathrm{I})^{-1}X^\intercal \mathbf{y}$$

Let’s derive this!

Overview

Formalise the problem, extend to *multiple* input dimensions, aka *vectorise*

How to handle non-linear *features*

How to control *overfitting* by *regularisation*

Discuss *equivalent* views: least squares = maximum likelihood, regularised least square = maximum a-posteriori (MAP). Why do we care -> *Bayesian* linear regression!

Numerical *issues*, computational *complexity,* and *workarounds*

When to use *numerical optimisation* instead, and how

The probabilistic perspective

Week 6

Bayes' rule: $P(\theta \mid \mathscr{D}) = \dfrac{P(\theta, \mathscr{D})}{P(\mathscr{D})} = \dfrac{P(\theta)\, P(\mathscr{D} \mid \theta)}{P(\mathscr{D})}$

$$\textbf{Posterior} \text{ (belief about } \theta \text{ after knowing } \mathscr{D}) = \frac{\textbf{Prior} \text{ (prior belief about } \theta) \times \textbf{Likelihood} \text{ (likelihood of } \theta \text{ given } \mathscr{D})}{\textbf{Marginal likelihood or evidence}}$$

Posterior
belief about θ after knowing $\mathscr{D}$

Prior
prior belief about θ

Likelihood
likelihood of θ given $\mathscr{D}$

Marginal likelihood
or evidence

likelihood averaged over all potential θ

The probabilistic perspective

Week 6

Bayes' rule: $P(\theta \mid \mathscr{D}) = \dfrac{P(\theta, \mathscr{D})}{P(\mathscr{D})} = \dfrac{P(\theta)\, P(\mathscr{D} \mid \theta)}{P(\mathscr{D})}$

Posterior
belief about θ after knowing $\mathscr{D}$

=

Prior
prior belief about θ

×

Likelihood
likelihood of θ given $\mathscr{D}$

Marginal likelihood
or evidence

likelihood averaged over all potential θ

- Maximum likelihood (ML/MLE), $\operatorname{argmax}_\theta \, p(\mathscr{D} \mid \theta)$ is equiv. empirical loss minimisation (ERM)
- Maximum a-posteriori (MAP), $\operatorname{argmax}_\theta \, p(\mathscr{D} \mid \theta) p(\theta)$ is equiv. regularised ERM
- Exact/approximate Bayesian inference

Linear regression - maximum likelihood

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Linear regression - maximum likelihood

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Likelihood: $p(\mathscr{D} \,|\, \theta) = \prod_n \mathscr{N}(y_n; f(x_n), \sigma^2) = \prod_n \mathscr{N}(y_n; x_n^\intercal \theta, \sigma^2)$

Factorise across data points

Mean = linear mapping

Measurement noise
Constant across data points

Linear regression - maximum likelihood

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Likelihood: $p(\mathscr{D} \mid \theta) = \prod_n \mathcal{N}(y_n; f(x_n), \sigma^2) = \prod_n \mathcal{N}(y_n; x_n^\intercal \theta, \sigma^2)$

Factorise across data points

Mean = linear mapping

Measurement noise
Constant across data points

Maximum likelihood, $\operatorname{argmax}_\theta\, p(\mathscr{D} \mid \theta)$ **is equiv. to least squares**

Linear regression - Maximum a posteriori

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Linear regression - Maximum a posteriori

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Likelihood: $p(\mathscr{D} \,|\, \theta) = \prod_n \mathscr{N}(y_n; f(x_n), \sigma^2) = \prod_n \mathscr{N}(y_n; x_n^\intercal \theta, \sigma^2)$

Prior: $p(\theta) = \mathscr{N}(\theta; \mathbf{0}, \sigma_o^2 \mathrm{I}) = \prod_d \mathscr{N}(\theta_d; 0, \sigma_o^2)$

Factorise across dimensions

Zero mean

Same variance for all dimensions

Linear regression - Maximum a posteriori

Training data N input, output pairs $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}$

Assumptions:

- Underlying function is **linear**, $f_\theta(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^\intercal \mathbf{x}, \theta \in \mathbb{R}^D$
- Due to measurement noise, observed y is a noisy version of $f(\mathbf{x})$

$$\begin{bmatrix} f_\theta(\mathbf{x}_1) \\ f_\theta(\mathbf{x}_2) \\ \vdots \\ f_\theta(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \theta^\intercal \mathbf{x}_1 \\ \theta^\intercal \mathbf{x}_2 \\ \vdots \\ \theta^\intercal \mathbf{x}_N \end{bmatrix} = X\theta$$

Likelihood: $p(\mathscr{D} \mid \theta) = \prod_n \mathcal{N}(y_n; f(x_n), \sigma^2) = \prod_n \mathcal{N}(y_n; x_n^\intercal \theta, \sigma^2)$

Prior: $p(\theta) = \mathcal{N}(\theta; \mathbf{0}, \sigma_o^2 \mathrm{I}) = \prod_d \mathcal{N}(\theta_d; 0, \sigma_o^2)$

Factorise across dimensions   Zero mean   Same variance for all dimensions

Maximum a posteriori (MAP), $\mathrm{argmax}_\theta \; p(\mathscr{D} \mid \theta) p(\theta)$ **is equiv. to regularised least squares**

Linear regression - Exact Bayesian inference

Week 6

Bayes' rule: $P(\theta \mid \mathscr{D}) = \dfrac{P(\theta, \mathscr{D})}{P(\mathscr{D})} = \dfrac{P(\theta)\, P(\mathscr{D} \mid \theta)}{P(\mathscr{D})}$

$$\underset{\text{belief about } \theta \text{ after knowing } \mathscr{D}}{\textbf{Posterior}} = \frac{\underset{\text{prior belief about } \theta}{\textbf{Prior}} \times \underset{\text{likelihood of } \theta \text{ given } \mathscr{D}}{\textbf{Likelihood}}}{\underset{\text{or evidence}}{\textbf{Marginal likelihood}}}$$

likelihood averaged over all potential θ

Linear regression - Exact Bayesian inference

Week 6

Bayes' rule: $P(\theta \mid \mathcal{D}) = \dfrac{P(\theta, \mathcal{D})}{P(\mathcal{D})} = \dfrac{P(\theta)\, P(\mathcal{D} \mid \theta)}{P(\mathcal{D})}$

Posterior
belief about θ after knowing $\mathcal{D}$ $=$ $\dfrac{\textbf{Prior}\text{ prior belief about }\theta \times \textbf{Likelihood}\text{ likelihood of }\theta\text{ given }\mathcal{D}}{\textbf{Marginal likelihood}\text{ or evidence}}$

likelihood averaged over all potential θ

Likelihood: $p(\mathcal{D} \mid \theta) = \mathcal{N}(\mathbf{y}; \mathbf{X}\theta, \sigma^2 \mathrm{I}_N)$

Prior: $p(\theta) = \mathcal{N}(\theta; \mathbf{0}, \sigma_o^2 \mathrm{I})$

Linear regression - Exact Bayesian inference

Week 6

Bayes' rule: $P(\theta \mid \mathscr{D}) = \dfrac{P(\theta, \mathscr{D})}{P(\mathscr{D})} = \dfrac{P(\theta)\, P(\mathscr{D} \mid \theta)}{P(\mathscr{D})}$

$$\dfrac{\textbf{Prior: } \text{prior belief about } \theta \;\times\; \textbf{Likelihood: } \text{likelihood of } \theta \text{ given } \mathscr{D}}{\textbf{Marginal likelihood or evidence}}$$

Posterior
belief about θ after knowing $\mathscr{D}$

=

Prior
prior belief about θ

×

Likelihood
likelihood of θ given $\mathscr{D}$

Marginal likelihood
or evidence

likelihood averaged over all potential θ

Likelihood: $p(\mathscr{D} \mid \theta) = \mathcal{N}(\mathbf{y}; \mathbf{X}\theta, \sigma^2 \mathrm{I}_N)$

Prior: $p(\theta) = \mathcal{N}(\theta; \mathbf{0}, \sigma_o^2 \mathrm{I})$

Posterior: $p(\theta \mid \mathscr{D}) = \mathcal{N}(\theta; \mu, \Sigma)$

$$\mu = \Sigma \sigma^{-2} \mathbf{X}^\intercal \mathbf{y}$$

Mean = MAP

$$\Sigma = (\sigma^{-2} \mathbf{X}^\intercal \mathbf{X} + \sigma_o^{-2} \mathrm{I}_D)^{-1}$$

Linear regression - Exact Bayesian inference

Week 6

Bayes' rule: $P(\theta \mid \mathscr{D}) = \dfrac{P(\theta, \mathscr{D})}{P(\mathscr{D})} = \dfrac{P(\theta)\, P(\mathscr{D} \mid \theta)}{P(\mathscr{D})}$

Posterior
belief about θ after knowing $\mathscr{D}$

=

Prior
prior belief about θ

×

Likelihood
likelihood of θ given $\mathscr{D}$

Marginal likelihood
or evidence

likelihood averaged over all potential θ

Likelihood: $p(\mathscr{D} \mid \theta) = \mathcal{N}(\mathbf{y}; \mathbf{X}\theta, \sigma^2 \mathrm{I}_N)$

Prior: $p(\theta) = \mathcal{N}(\theta; \mathbf{0}, \sigma_o^2 \mathrm{I})$

Posterior: $p(\theta \mid \mathscr{D}) = \mathcal{N}(\theta; \mu, \Sigma)$

$$\mu = \Sigma \sigma^{-2} \mathbf{X}^\intercal \mathbf{y}$$

Mean = MAP

$$\Sigma = (\sigma^{-2} \mathbf{X}^\intercal \mathbf{X} + \sigma_o^{-2} \mathrm{I}_D)^{-1}$$

Marginal likelihood: $p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \sigma_o^2 \mathbf{X}\mathbf{X}^\intercal + \sigma^2 \mathrm{I}_N)$

Linear regression - Exact Bayesian inference

Week 6

Bayes' rule: $P(\theta \mid \mathscr{D}) = \dfrac{P(\theta, \mathscr{D})}{P(\mathscr{D})} = \dfrac{P(\theta)\, P(\mathscr{D} \mid \theta)}{P(\mathscr{D})}$

Posterior
belief about θ after knowing $\mathscr{D}$

=

Prior
prior belief about θ

$\times$

Likelihood
likelihood of θ given $\mathscr{D}$

Marginal likelihood
or evidence

likelihood averaged over all potential θ

Likelihood: $p(\mathscr{D} \mid \theta) = \mathcal{N}(\mathbf{y}; \mathbf{X}\theta, \sigma^2 \mathrm{I}_N)$

Prior: $p(\theta) = \mathcal{N}(\theta; \mathbf{0}, \sigma_o^2 \mathrm{I})$

Posterior: $p(\theta \mid \mathscr{D}) = \mathcal{N}(\theta; \mu, \Sigma)$

$\mu = \Sigma \sigma^{-2} \mathbf{X}^\intercal \mathbf{y}$

Mean = MAP

$\Sigma = (\sigma^{-2} \mathbf{X}^\intercal \mathbf{X} + \sigma_o^{-2} \mathrm{I}_D)^{-1}$

Marginal likelihood: $p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \sigma_o^2 \mathbf{X}\mathbf{X}^\intercal + \sigma^2 \mathrm{I}_N)$

Assignment 3: use the marginal likelihood to pick noise variance

Linear regression - Prediction

Point estimate

$\theta_{\mathrm{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y}$ or $\theta_{\mathrm{MAP}} = (X^\intercal X + N\lambda \mathrm{I})^{-1} X^\intercal \mathbf{y}$

$f(\mathbf{x}^*) = \theta_{\mathrm{ML}}^\intercal \mathbf{x}^*$ or $\theta_{\mathrm{MAP}}^\intercal \mathbf{x}^*$

Exact posterior $p(\theta \,|\, \mathscr{D}) = \mathcal{N}(\theta; \mu, \Sigma)$

$p(f(\mathbf{x}^*) \,|\, \mathbf{x}^*, \mathscr{D}) = \mathcal{N}(f(\mathbf{x}^*); \mu^\intercal \mathbf{x}^*, \mathbf{x}^{*\intercal} \Sigma \mathbf{x}^*)$

Linear regression - Prediction

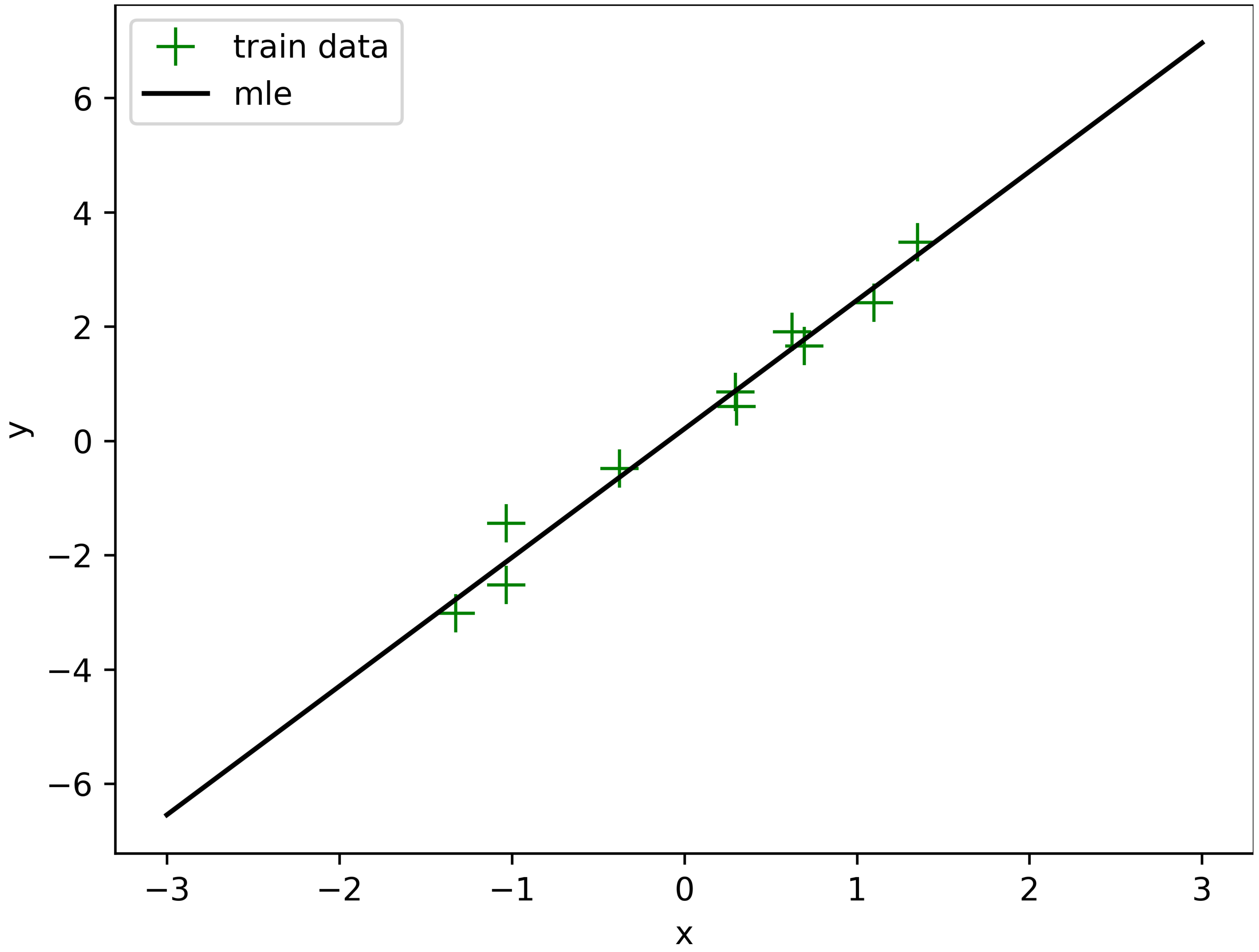
Point estimate

$\theta_{\mathrm{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y}$ or $\theta_{\mathrm{MAP}} = (X^\intercal X + N\lambda \mathbf{I})^{-1} X^\intercal \mathbf{y}$

$f(\mathbf{x}^*) = \theta_{\mathrm{ML}}^\intercal \mathbf{x}^*$ or $\theta_{\mathrm{MAP}}^\intercal \mathbf{x}^*$

Exact posterior $p(\theta \,|\, \mathscr{D}) = \mathscr{N}(\theta; \mu, \Sigma)$

$p(f(\mathbf{x}^*) \,|\, \mathbf{x}^*, \mathscr{D}) = \mathscr{N}(f(\mathbf{x}^*); \mu^\intercal \mathbf{x}^*, \mathbf{x}^{*\intercal} \Sigma \mathbf{x}^*)$



Linear regression - Prediction

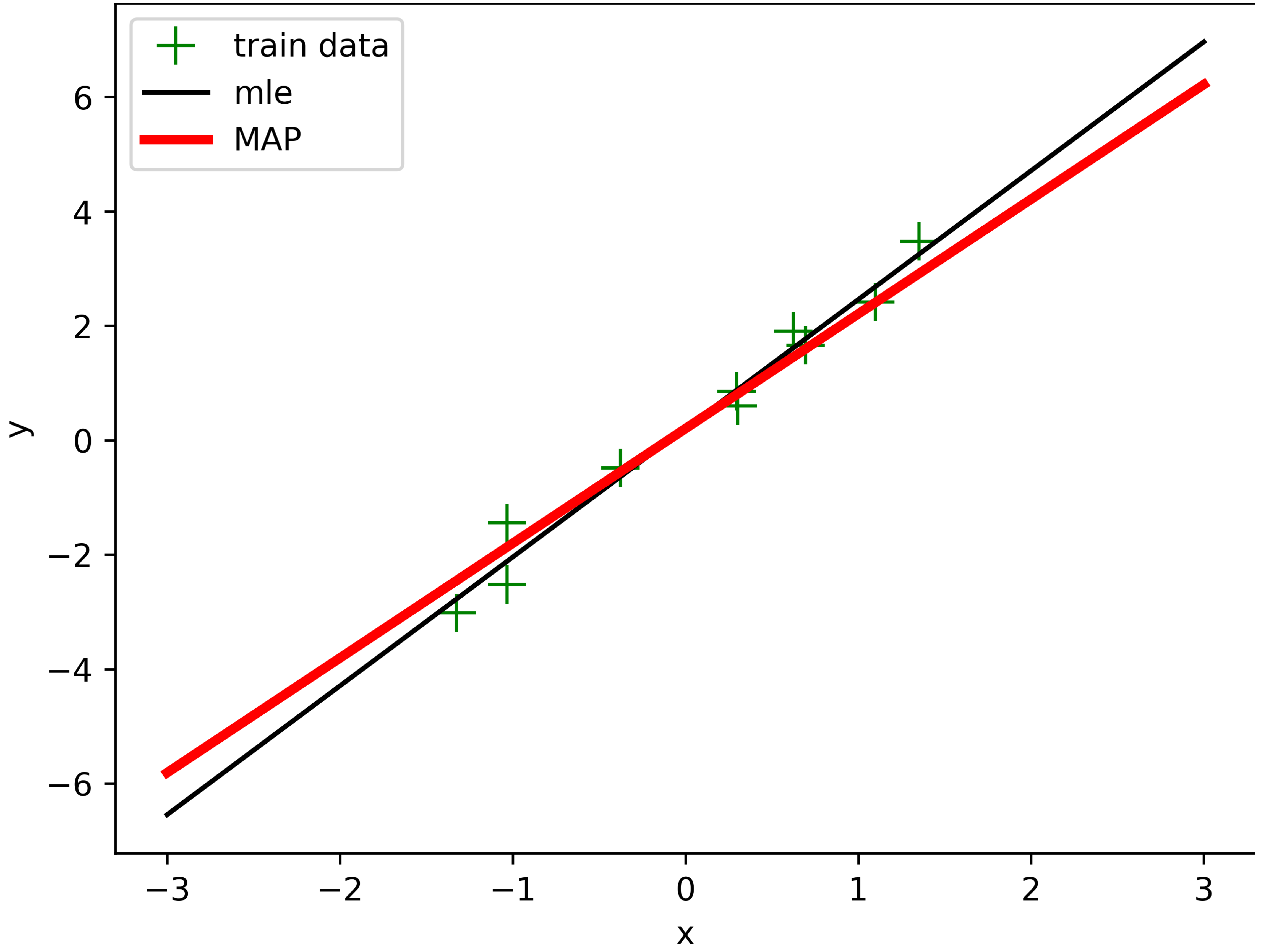
Point estimate

$\theta_{\mathrm{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y}$ or $\theta_{\mathrm{MAP}} = (X^\intercal X + N\lambda \mathrm{I})^{-1} X^\intercal \mathbf{y}$

$f(\mathbf{x}^*) = \theta_{\mathrm{ML}}^\intercal \mathbf{x}^*$ or $\theta_{\mathrm{MAP}}^\intercal \mathbf{x}^*$

Exact posterior $p(\theta \,|\, \mathscr{D}) = \mathscr{N}(\theta; \mu, \Sigma)$

$p(f(\mathbf{x}^*) \,|\, \mathbf{x}^*, \mathscr{D}) = \mathscr{N}(f(\mathbf{x}^*); \mu^\intercal \mathbf{x}^*, \mathbf{x}^{*\intercal} \Sigma \mathbf{x}^*)$



Linear regression - Prediction

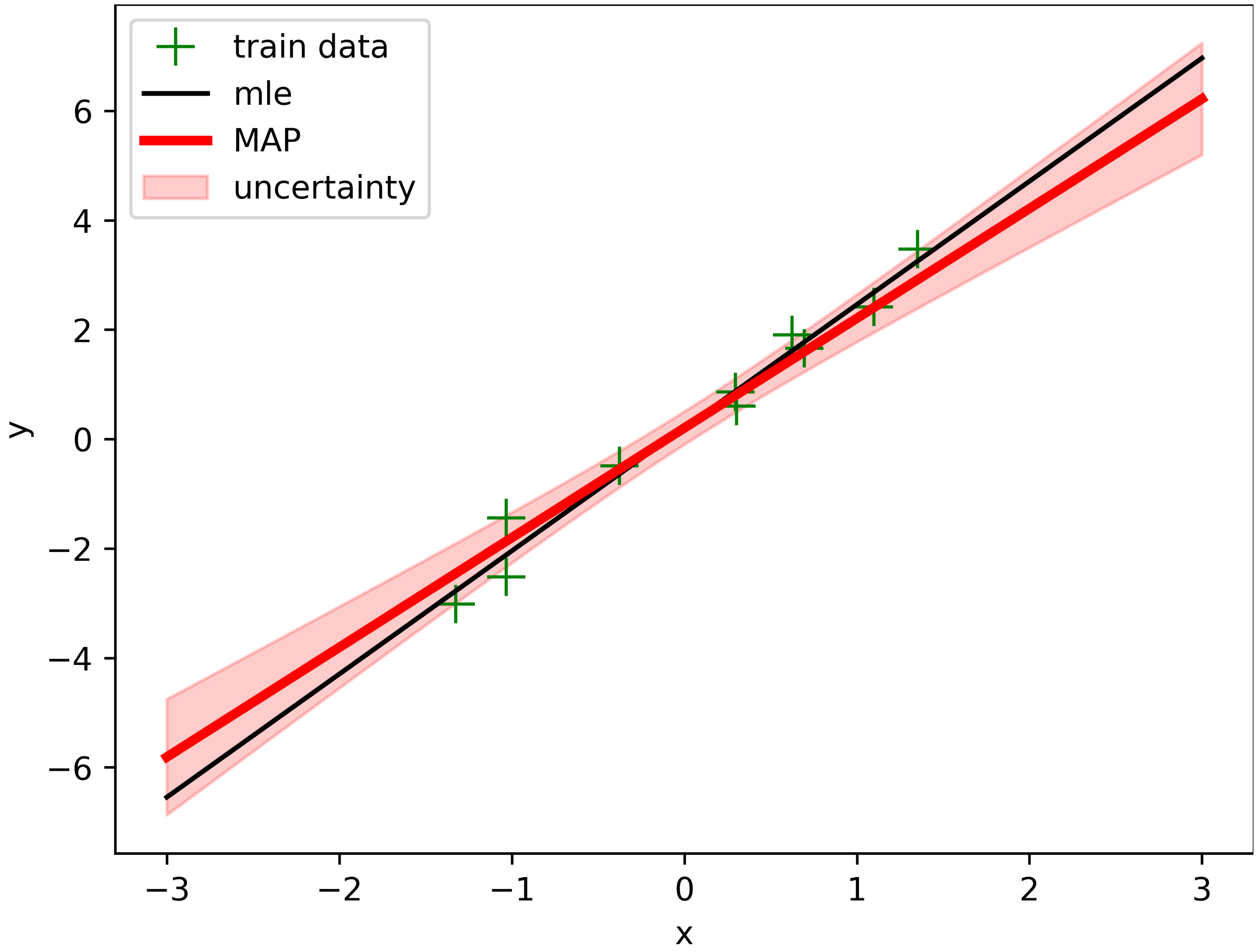
Point estimate

$$\theta_{\mathrm{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y} \text{ or } \theta_{\mathrm{MAP}} = (X^\intercal X + N\lambda \mathrm{I})^{-1} X^\intercal \mathbf{y}$$

$$f(\mathbf{x}^*) = \theta_{\mathrm{ML}}^\intercal \mathbf{x}^* \quad \text{or} \quad \theta_{\mathrm{MAP}}^\intercal \mathbf{x}^*$$

Exact posterior $p(\theta \mid \mathscr{D}) = \mathscr{N}(\theta; \mu, \Sigma)$

$$p(f(\mathbf{x}^*) \mid \mathbf{x}^*, \mathscr{D}) = \mathscr{N}(f(\mathbf{x}^*); \mu^\intercal \mathbf{x}^*, \mathbf{x}^{*\intercal} \Sigma \mathbf{x}^*)$$



Linear regression - Prediction

Point estimate

$$\theta_{\text{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y} \text{ or } \theta_{\text{MAP}} = (X^\intercal X + N\lambda \mathrm{I})^{-1} X^\intercal \mathbf{y}$$

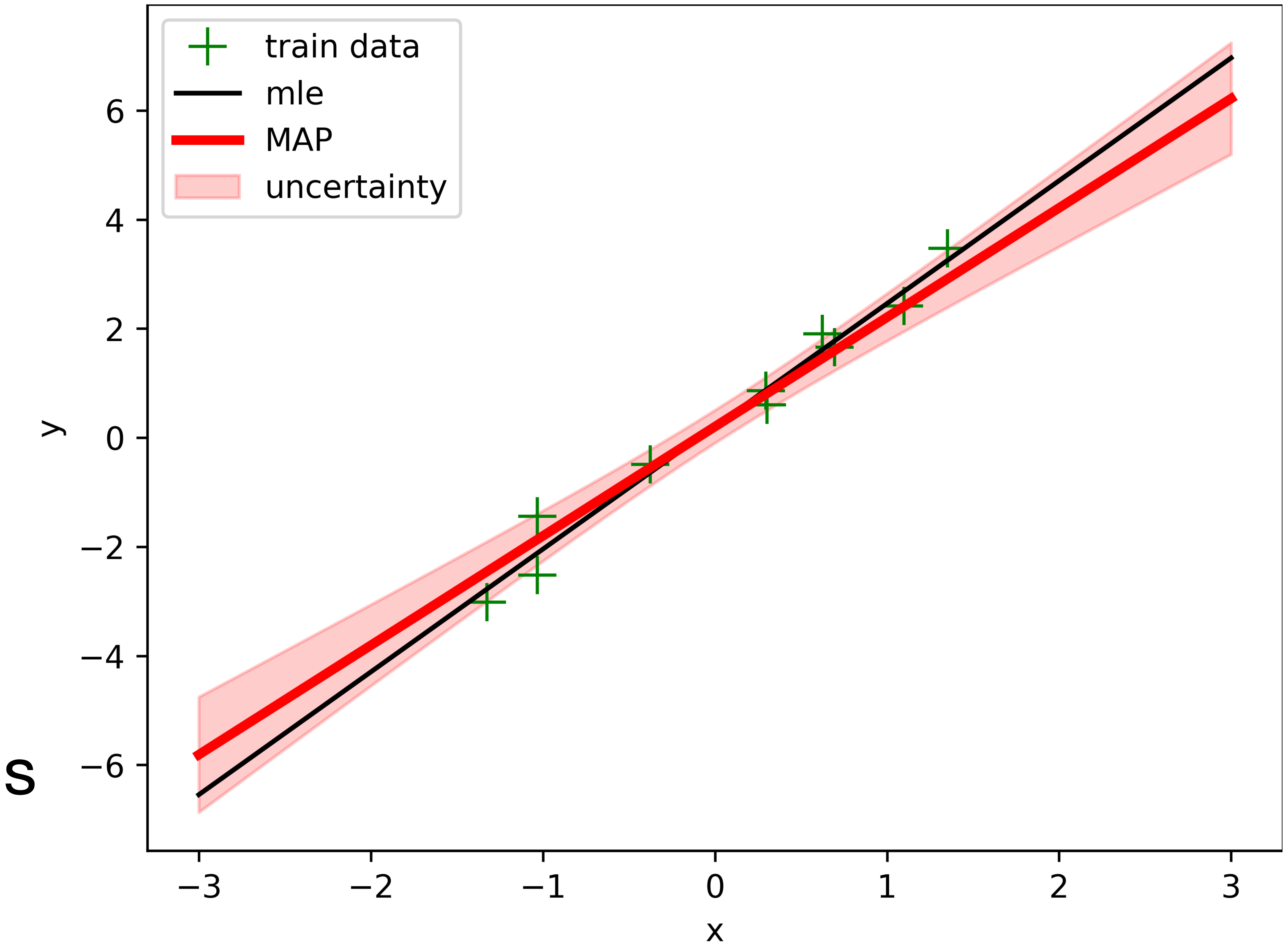
$$f(\mathbf{x}^*) = \theta_{\text{ML}}^\intercal \mathbf{x}^* \quad \text{or} \quad \theta_{\text{MAP}}^\intercal \mathbf{x}^*$$

Exact posterior $p(\theta \,|\, \mathscr{D}) = \mathcal{N}(\theta; \mu, \Sigma)$

$$p(f(\mathbf{x}^*) \,|\, \mathbf{x}^*, \mathscr{D}) = \mathcal{N}(f(\mathbf{x}^*); \mu^\intercal \mathbf{x}^*, \mathbf{x}^{*\intercal} \Sigma \mathbf{x}^*)$$

Nice things about the Bayesian perspective:

- MLE and MAP as special cases
- Capture all plausible solutions*
- Be explicit about the assumptions:
  - Gaussian independent measurement noise
  - Gaussian prior over parameters
- Can be adapted to handle other priors/likelihoods



*Model mis-specification is an issue

Overview

Formalise the problem, extend to *multiple* input dimensions, aka *vectorise*

How to handle non-linear *features*

How to control *overfitting* by *regularisation*

Discuss *equivalent* views: least squares = maximum likelihood, regularised least square = maximum a-posteriori (MAP). Why do we care -> *Bayesian* linear regression!

Numerical *issues*, computational *complexity,* and *workarounds*

When to use *numerical optimisation* instead, and how

Linear regression - Potential issues

Point estimate $\theta_{\mathrm{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y}$ or $\theta_{\mathrm{MAP}} = (X^\intercal X + N\lambda \mathrm{I})^{-1} X^\intercal \mathbf{y}$

We have assumed this is invertible. But this is not guaranteed! So use this instead! Why?

Linear regression - Potential issues

Point estimate $\theta_{\mathrm{ML}} = (X^\intercal X)^{-1} X^\intercal \mathbf{y}$ or $\theta_{\mathrm{MAP}} = (X^\intercal X + N\lambda \mathbf{I})^{-1} X^\intercal \mathbf{y}$

We have assumed this is invertible. But this is not guaranteed! So use this instead! Why?

Computational complexity $\mathcal{O}(ND^2 + ND + D^3) = \mathcal{O}(ND^2 + D^3)$. Can be large for large D.

- use matrix inversion lemma, or,
- use numerical optimisation instead

Overview

Formalise the problem, extend to *multiple* input dimensions, aka *vectorise*

How to handle non-linear *features*

How to control *overfitting* by *regularisation*

Discuss *equivalent* views: least squares = maximum likelihood, regularised least square = maximum a-posteriori (MAP). Why do we care -> *Bayesian* linear regression!

Numerical *issues*, computational *complexity,* and *workarounds*

When to use *numerical optimisation* instead, and how