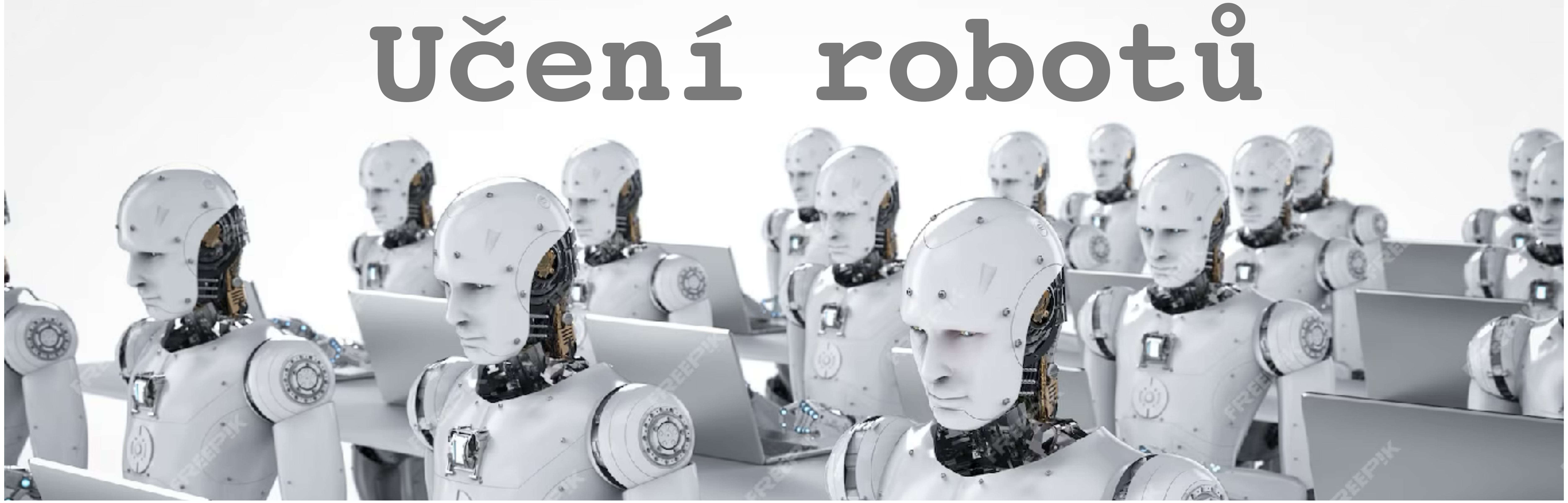


Learning 101

Engineering view on learning, issues, regression, classification.

Pre-requisites:

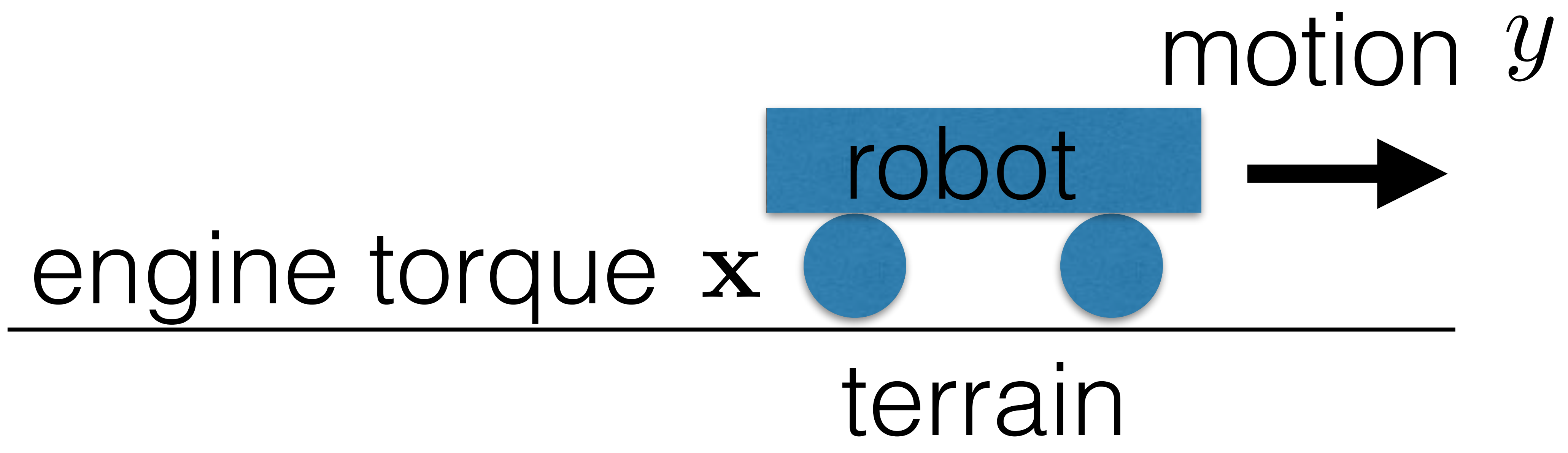
- just an elementary linear algebra

Karel Zimmermann
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Financováno Evropskou unií NextGenerationEU
Národní plán obnovy
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Motivation example: estimation of a motion model

- Which method will you use to build a motion model?

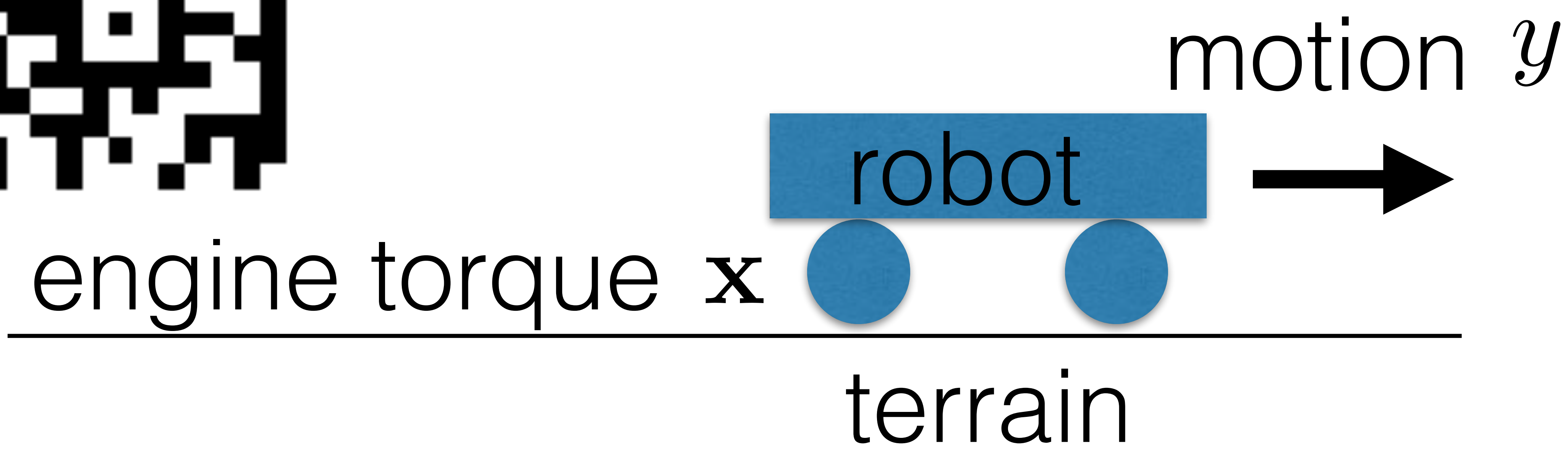


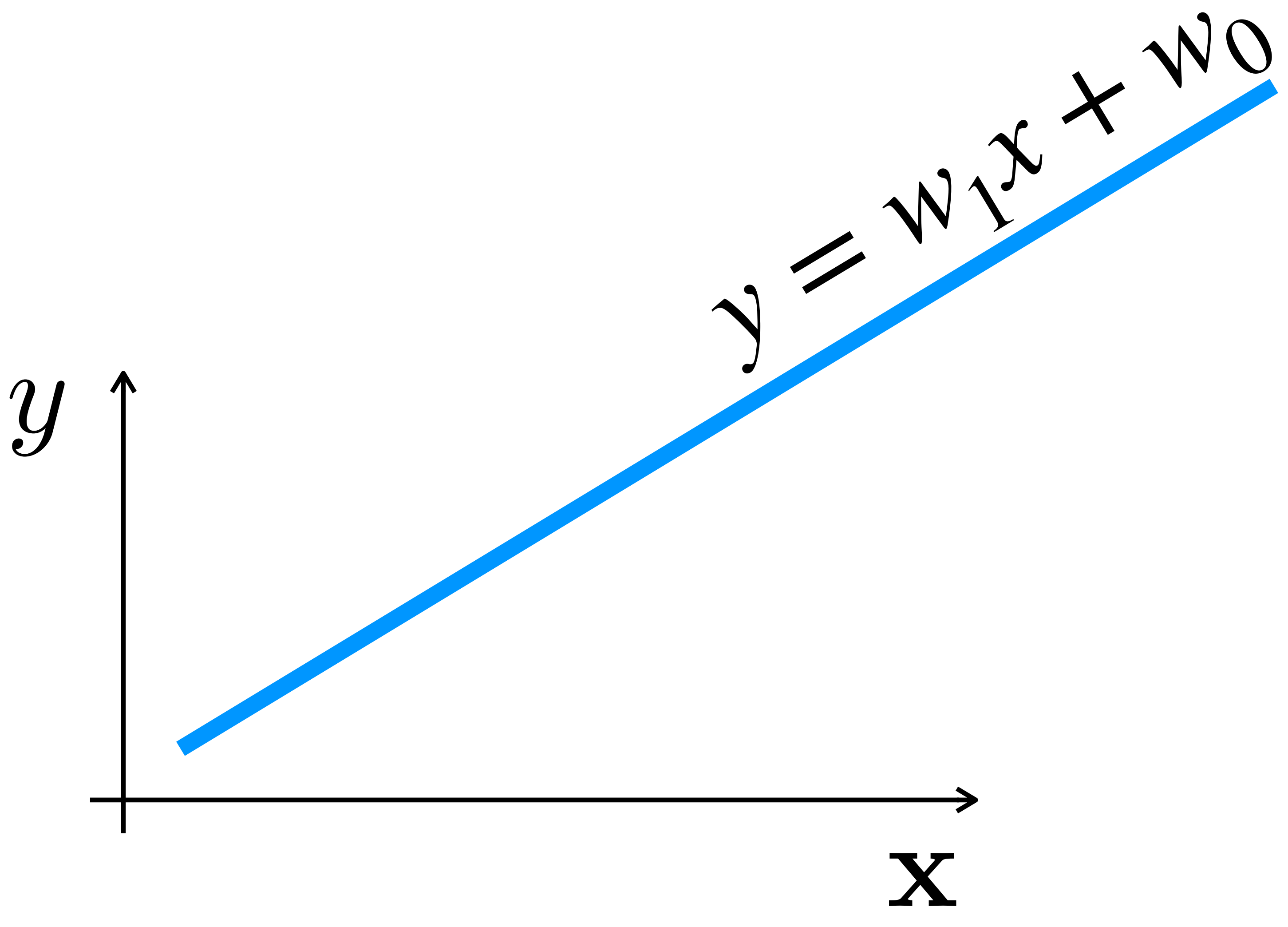
https://app.sli.do/event/ns8aHakuMNjuFcST9xf9rf

Motivation example: estimation of a motion model

- Which method will you use to build a motion model?
- Algorithm that maps x on y (or prob distr of y)

Do you know linear regression?

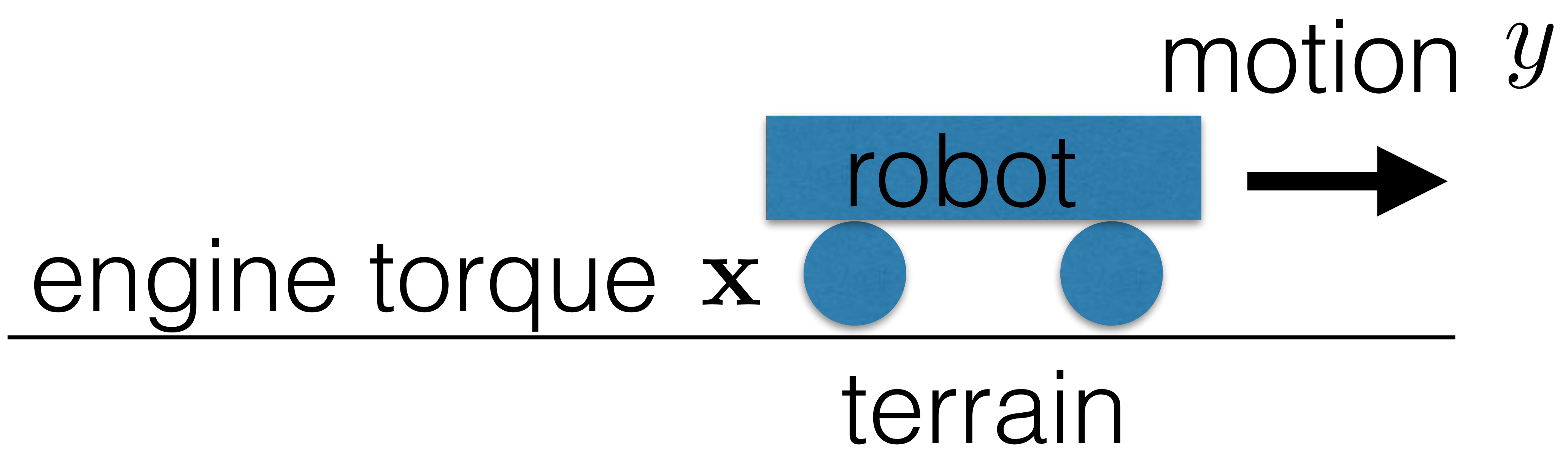


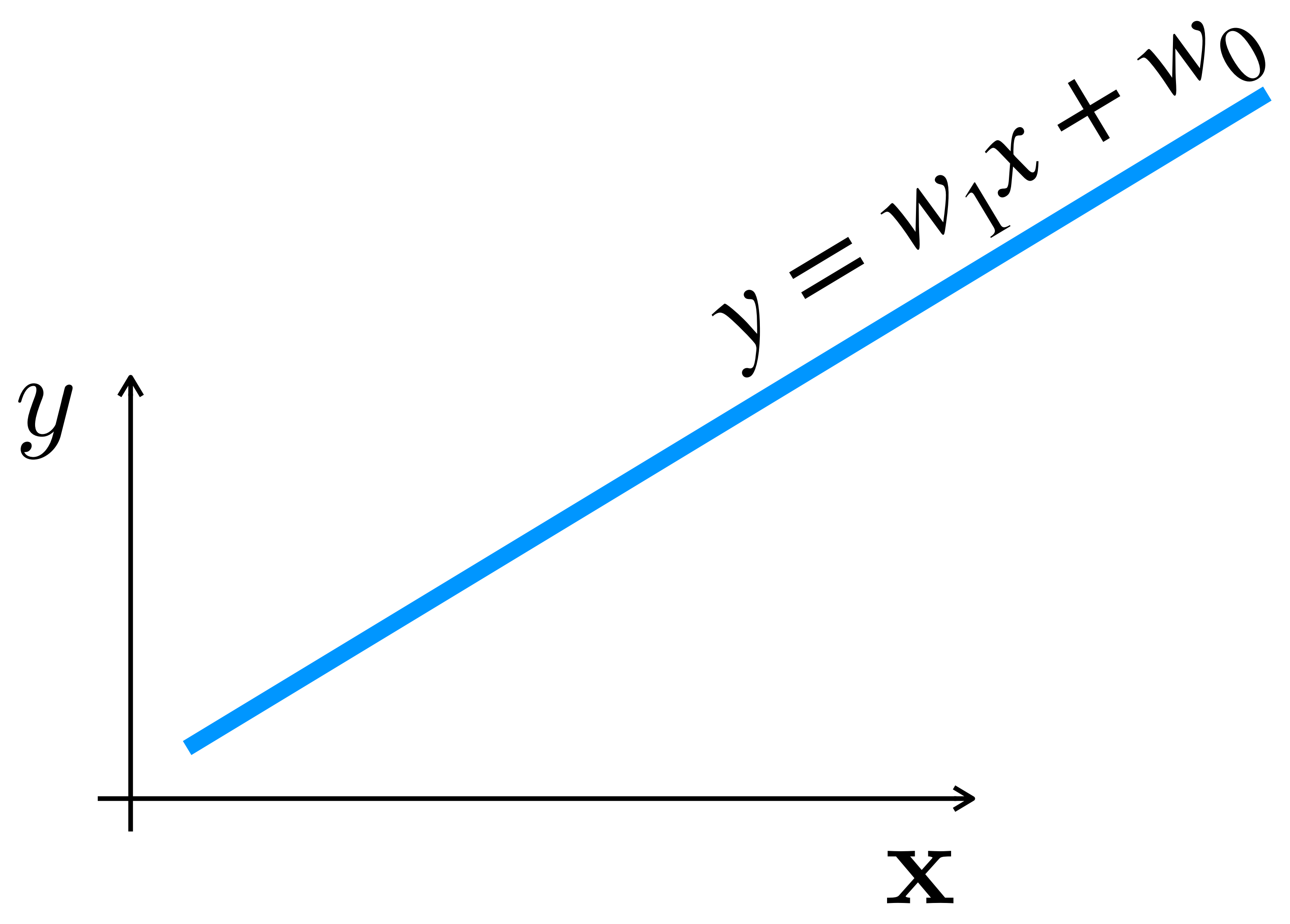


https://app.sli.do/event/ns8aHakuMNjuFcST9xf9rf

Motivation example: estimation of a motion model

- Which method will you use to build a motion model?
- Algorithm that maps x on y (or prob distr of y)
- This algorithm has some parameters => how to find them? => trn data+ loss+opt

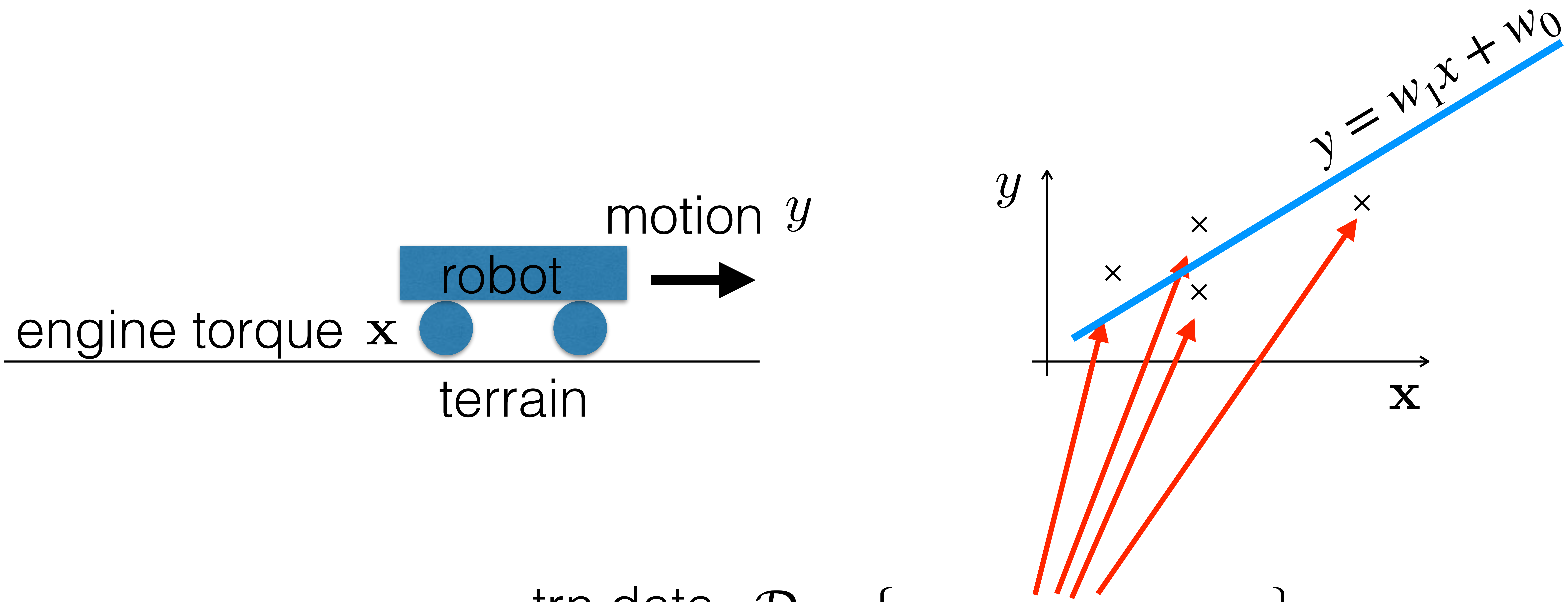




https://app.sli.do/event/ns8aHakuMNjuFcST9xf9rf

Motivation example: estimation of a motion model

- Which method will you use to build a motion model?
- Algorithm that maps x on y (or prob distr of y)
- This algorithm has some parameters => how to find them? =>   +loss+opt

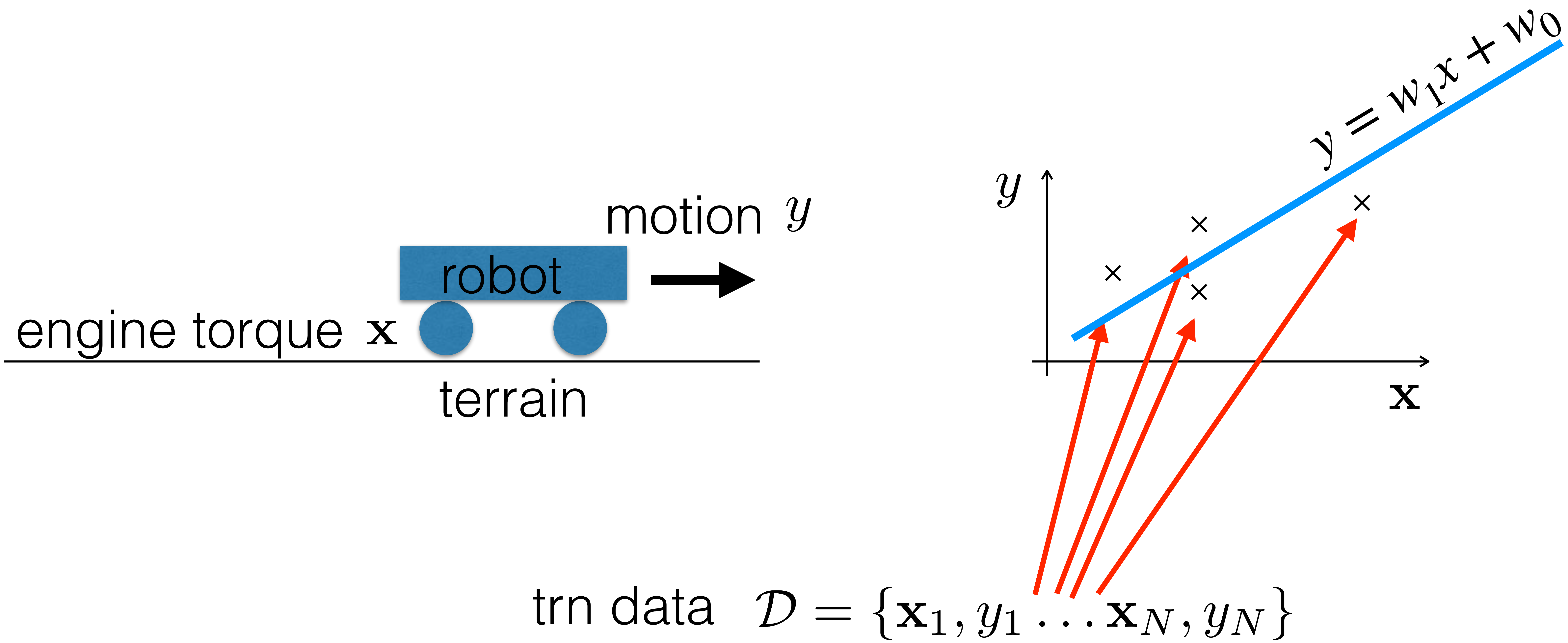


trn data $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

Motivation example: estimation of a motion model

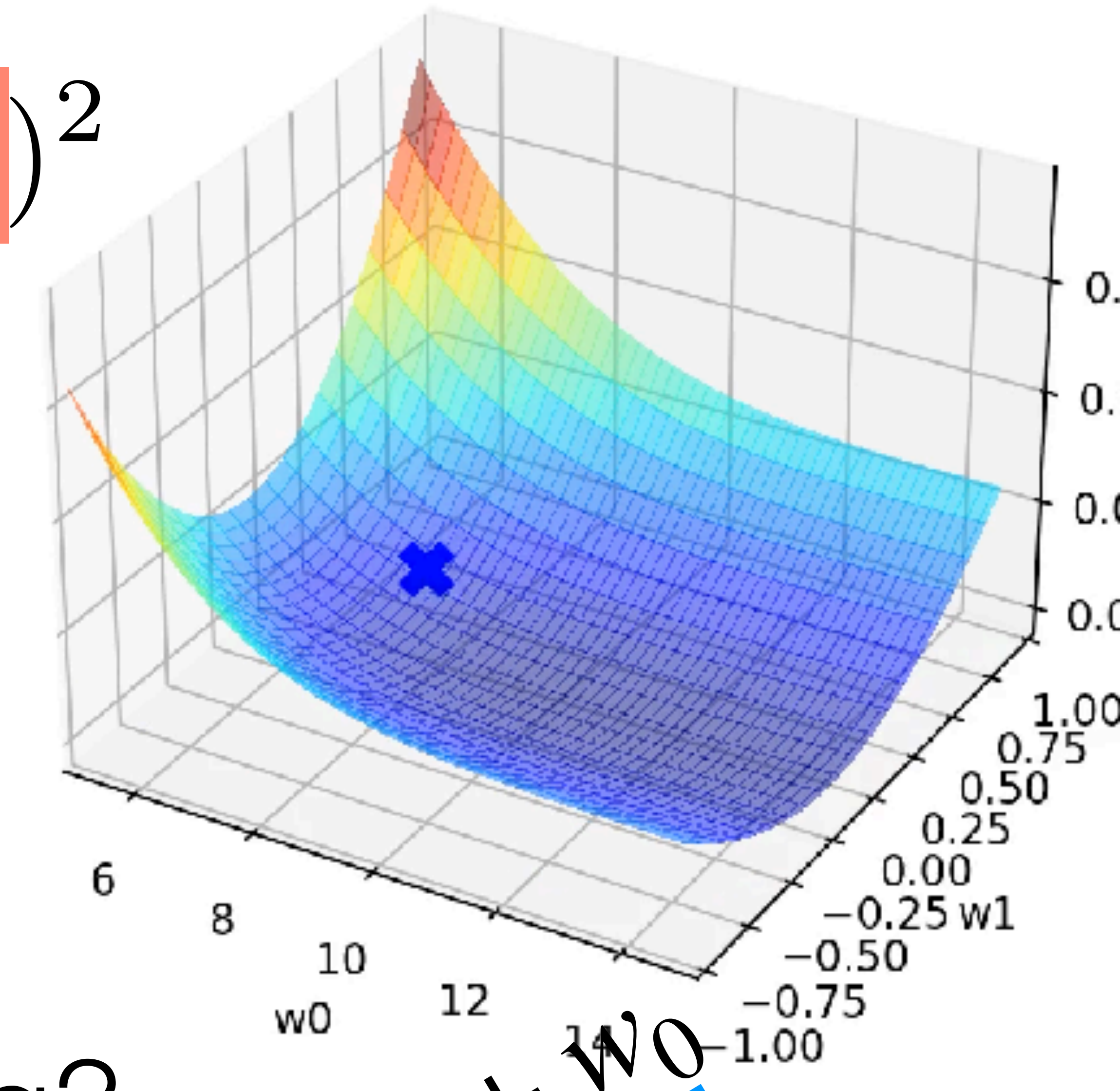
- Let's implement it! loss = **???**

opt = **???**



Motivation example: estimation of a motion model

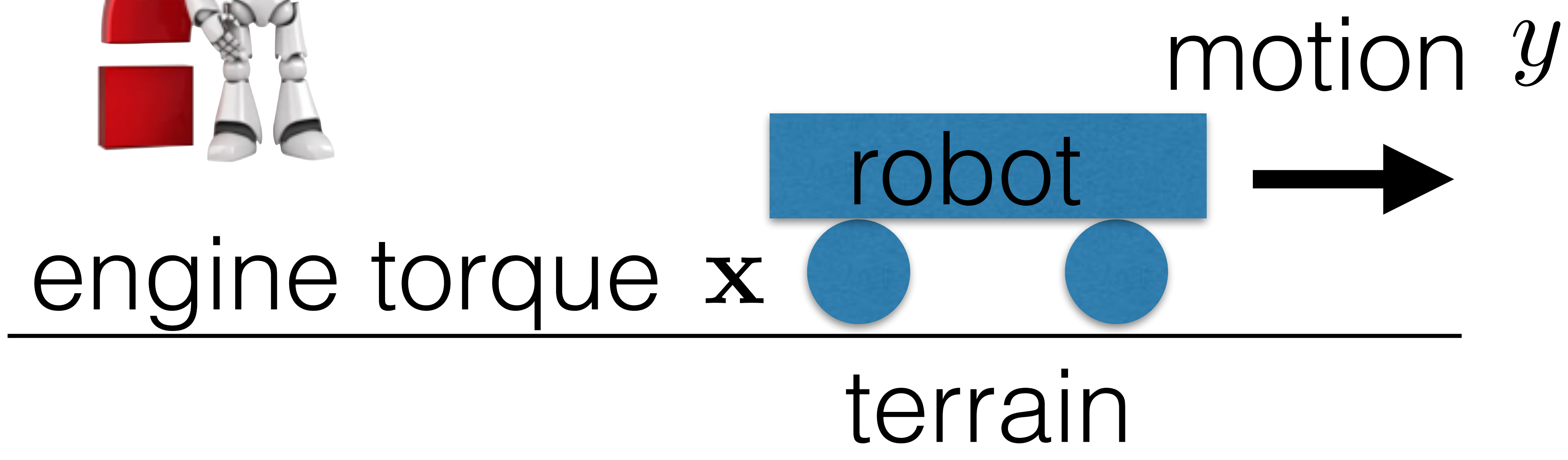
- Let's implement it! loss: $\arg\min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$

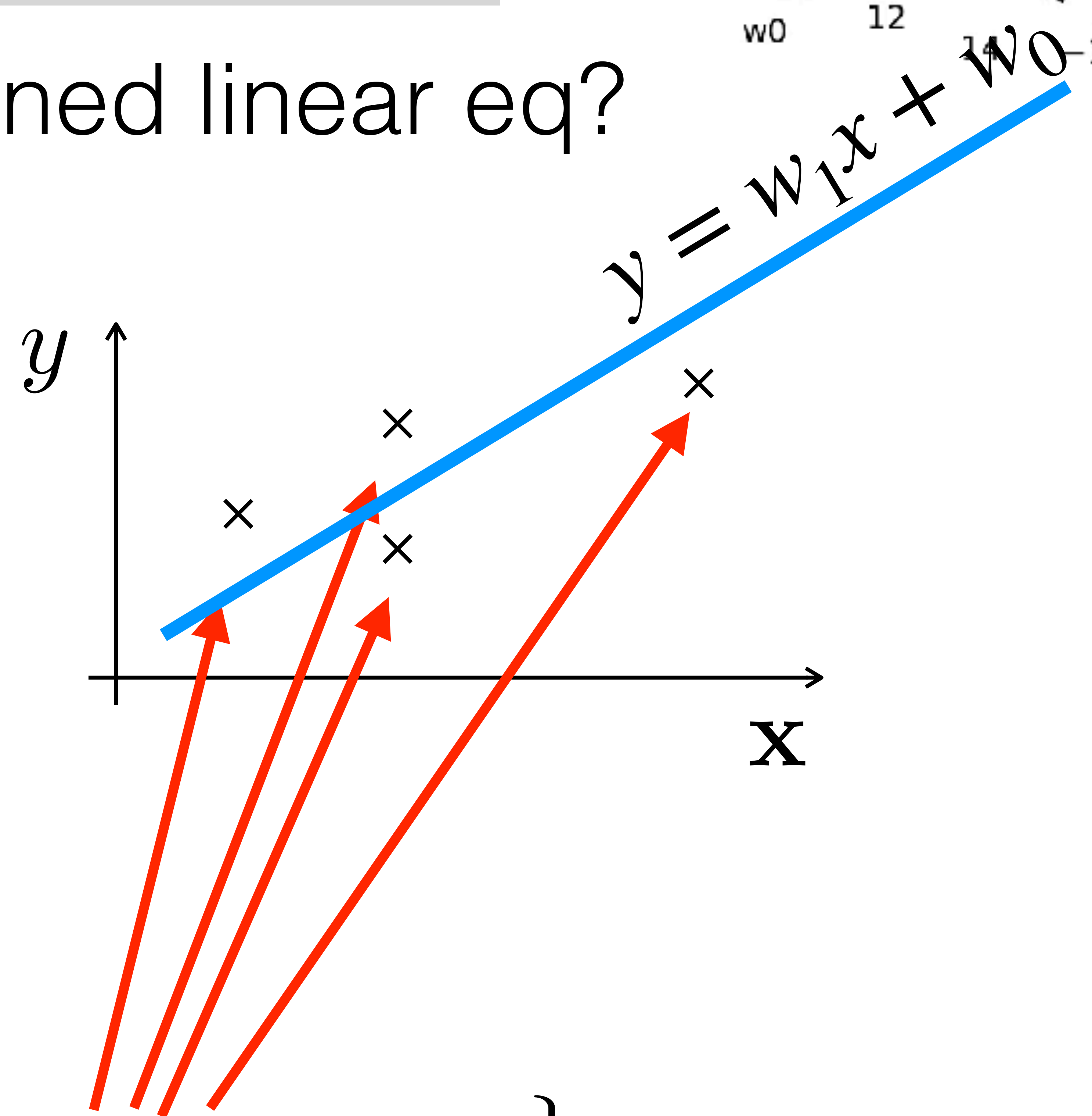


opt =

```
w = np.array([-2.0, 2.0])
for i in range(0, 10):
    loss = np.sum( (w[0] * x + w[1] - y)**2 )
    w = w - 0.1 * grad(loss, w)
```

- Which functions can be fitted through overdetermined linear eq?





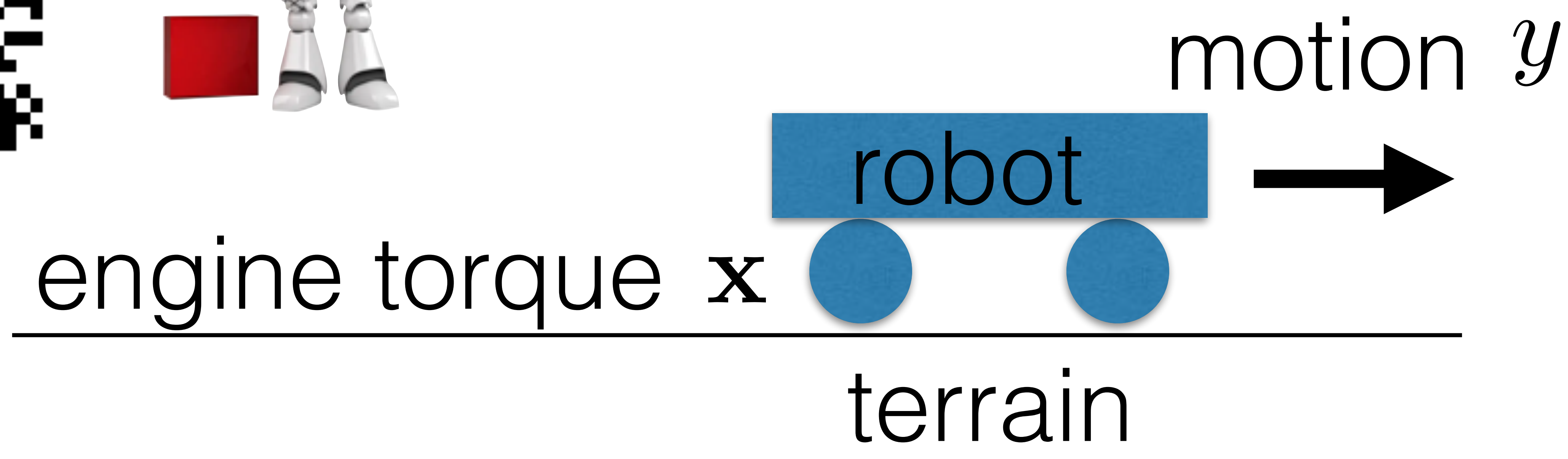
trn data $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

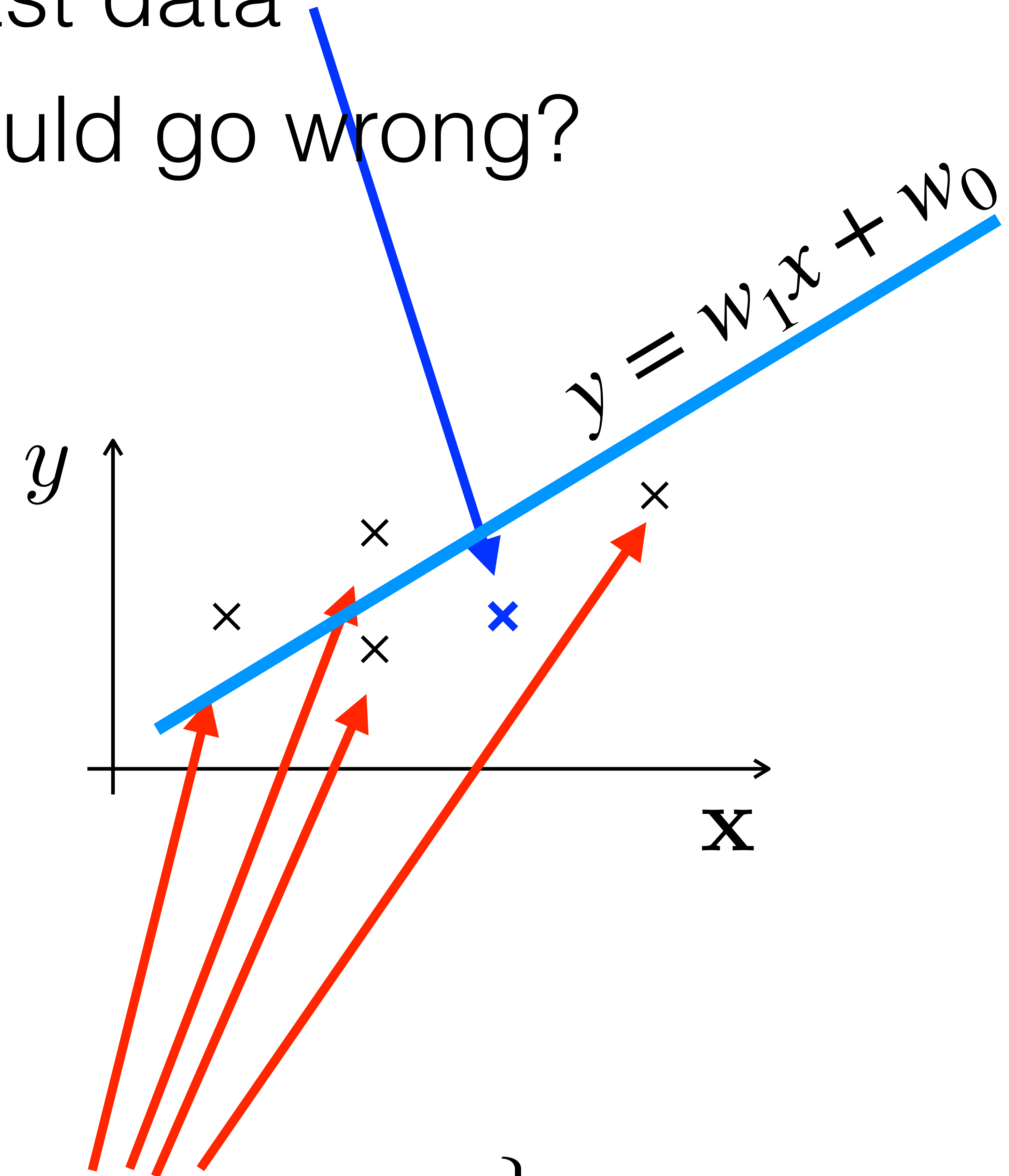
Motivation example: estimation of a motion model

- What do I need to build a motion model?
- Algorithm that maps x on y (or prob distr of y)
- This algorithm has some parameters => how to find them? => loss+trn data+opt

SOLVED

- How to decide that the algorithm works well? => tst data
- What if the algorithm does not work well? What could go wrong?

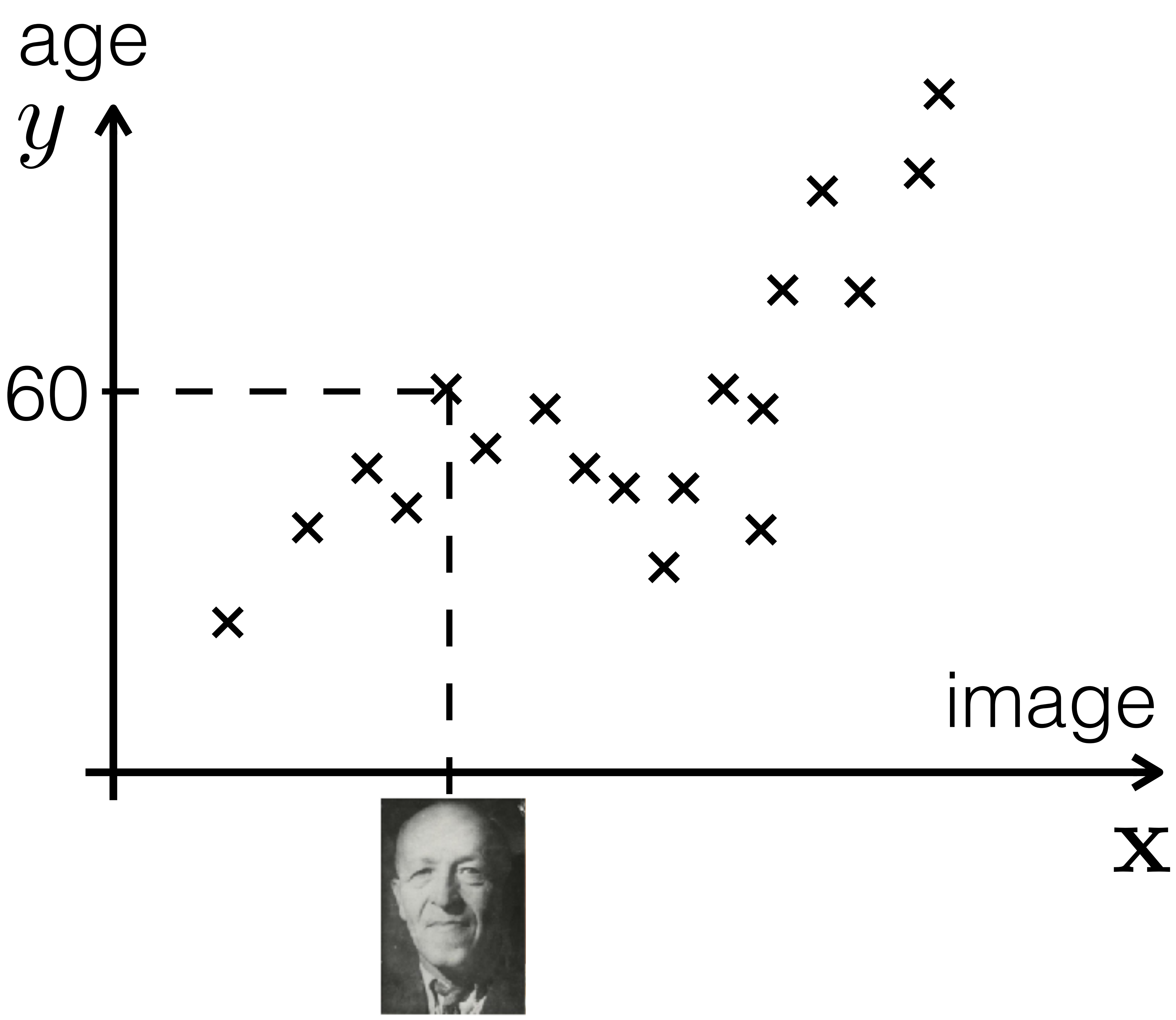




trn data $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

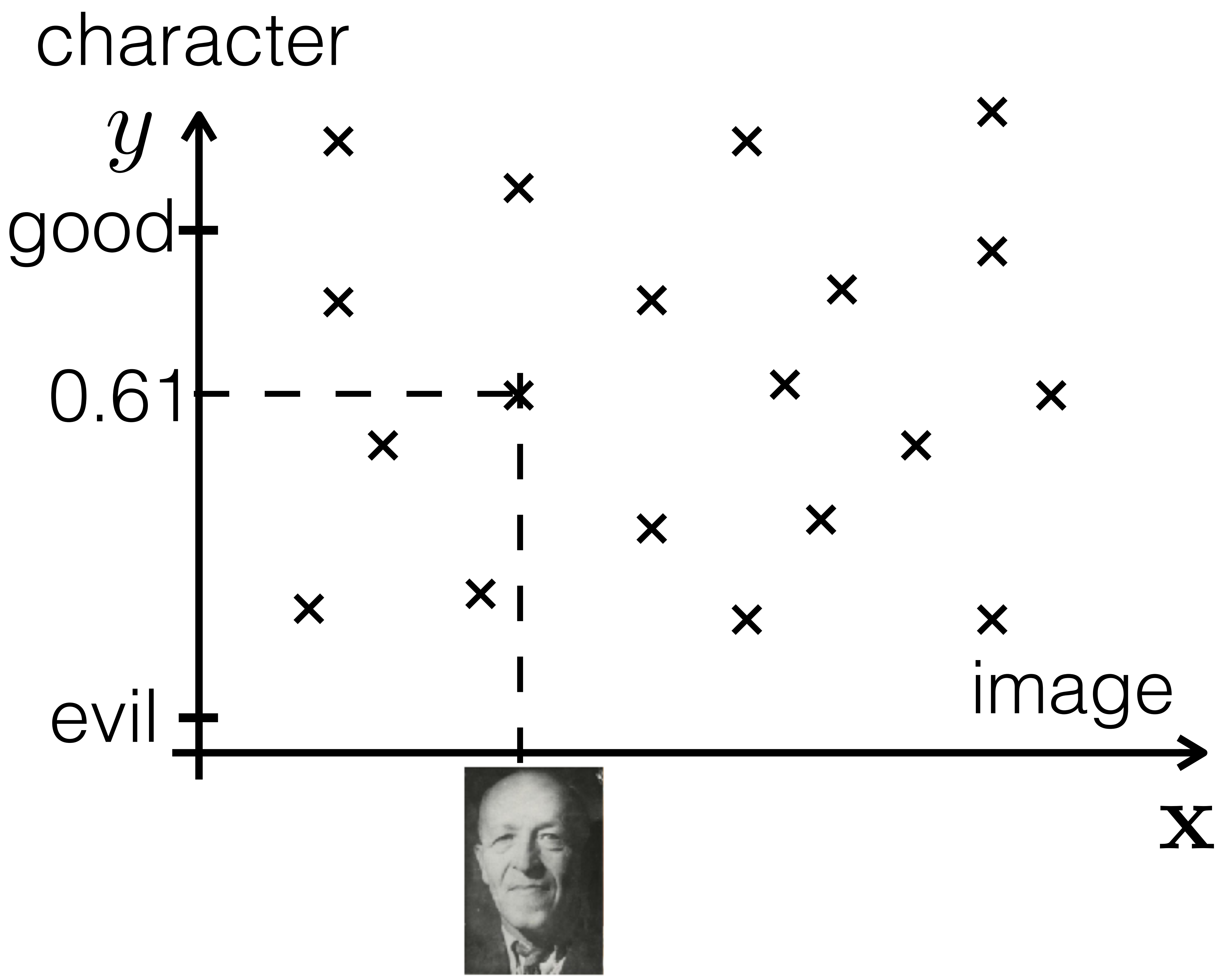
What can go wrong: **inputs x does not allow to predict y**

predicting person's age from face image

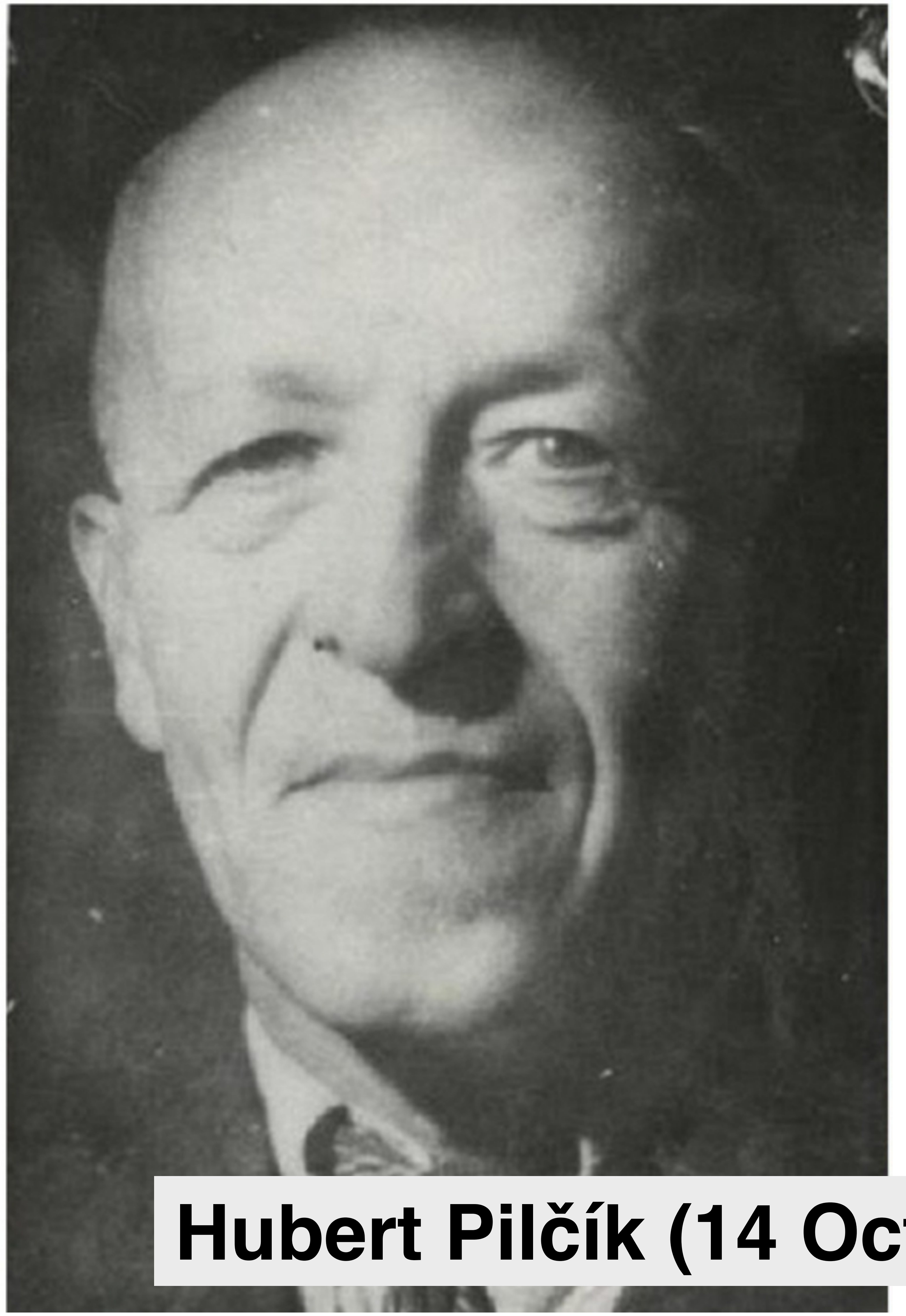


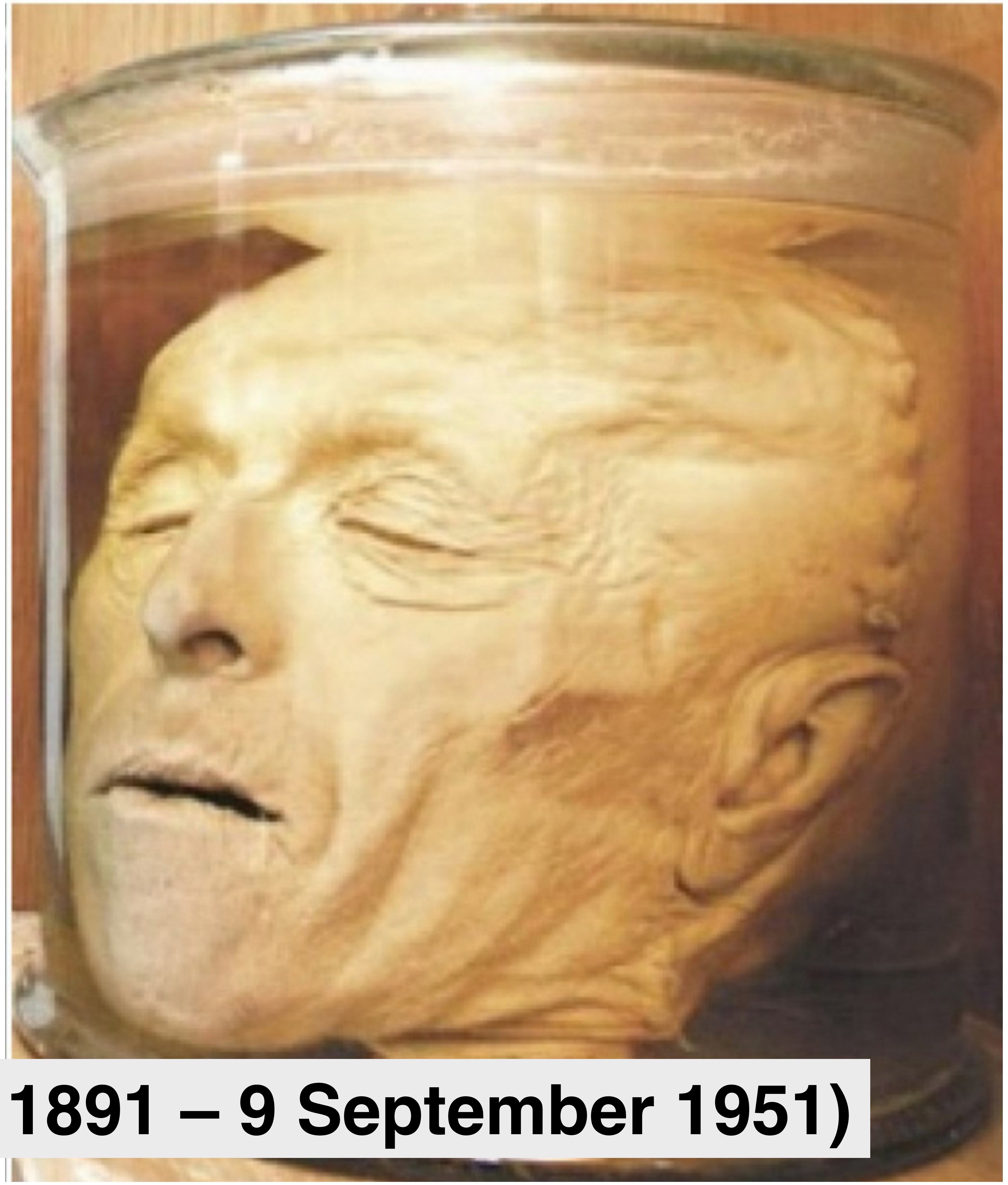
vs

predicting human character from face pictures



What can go wrong: **inputs x does not allow to predict y**





Hubert Pilčík (14 October 1891 – 9 September 1951)

A Deep Neural Network Model to Predict Criminality Using Image Processing
https://medium.com/@CoalitionForCriticalTechnology/abolish-the-techtoprisonpipeline-9b5b14366b16

What can go wrong: **inputs x does not allow to predict y**

- Nazi V-2
- US Apollo program

- Kalman filter

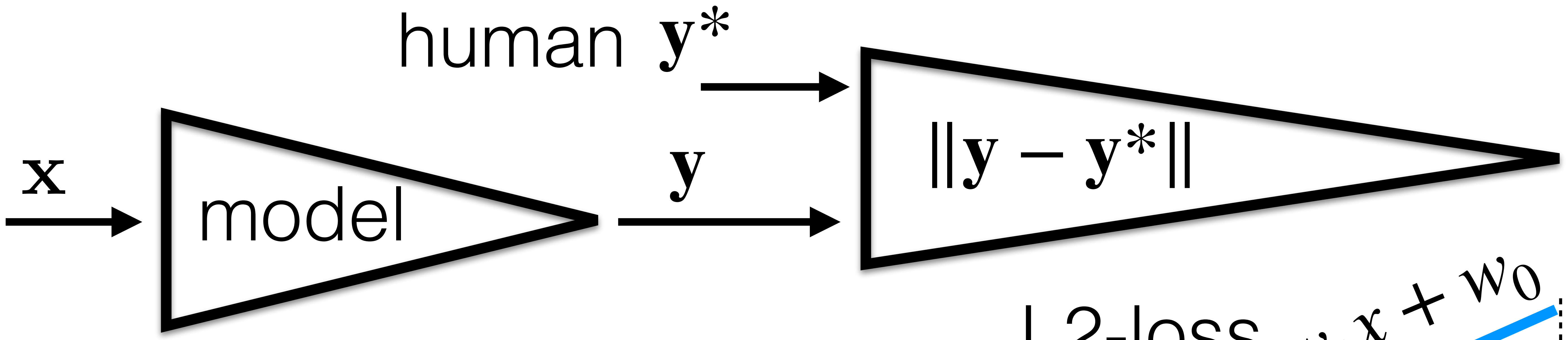
What can go wrong: **trn/tst data distribution mismatch**

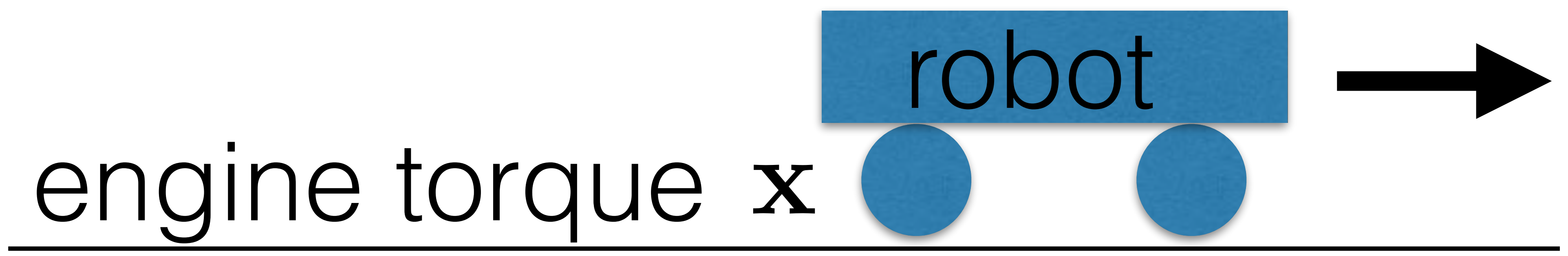
- Day/night, summer/winter, USA/China, young/old people, American/Indian English

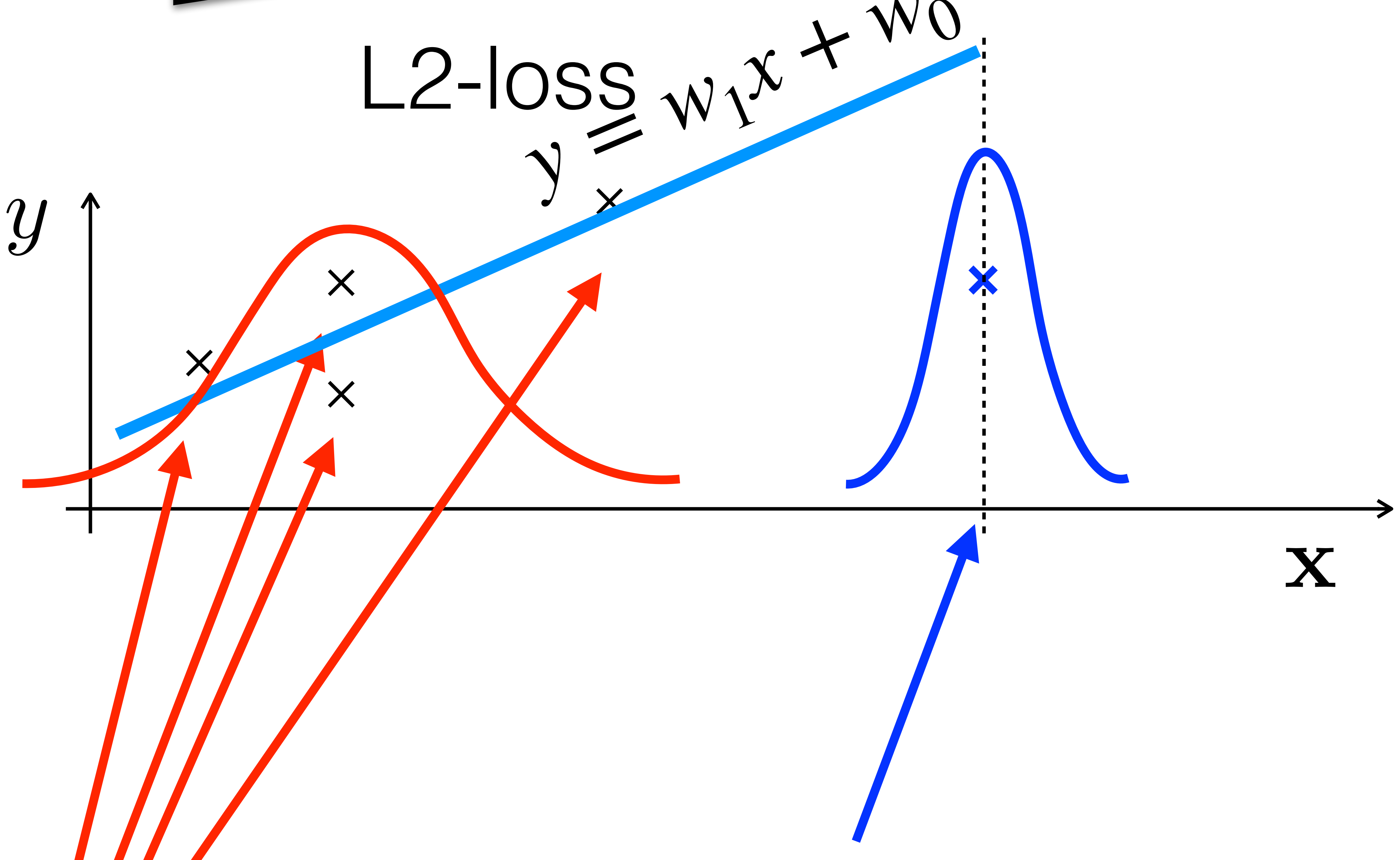
[NVidia, CVPR, 2016]

Does this suffer from trn/tst distribution mismatch?

Statistical consistency







trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$ tst data:

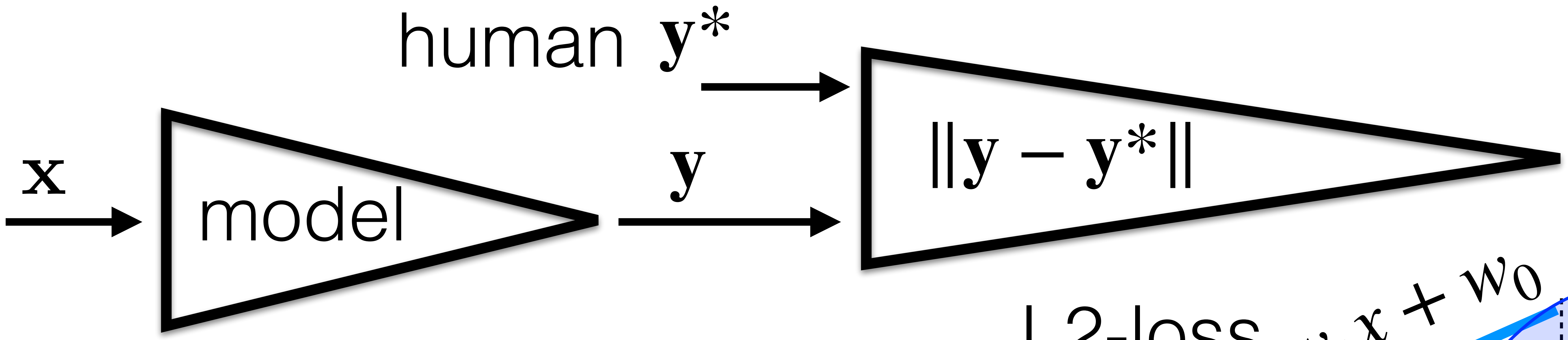
What can go wrong: **trn/tst data distribution mismatch**

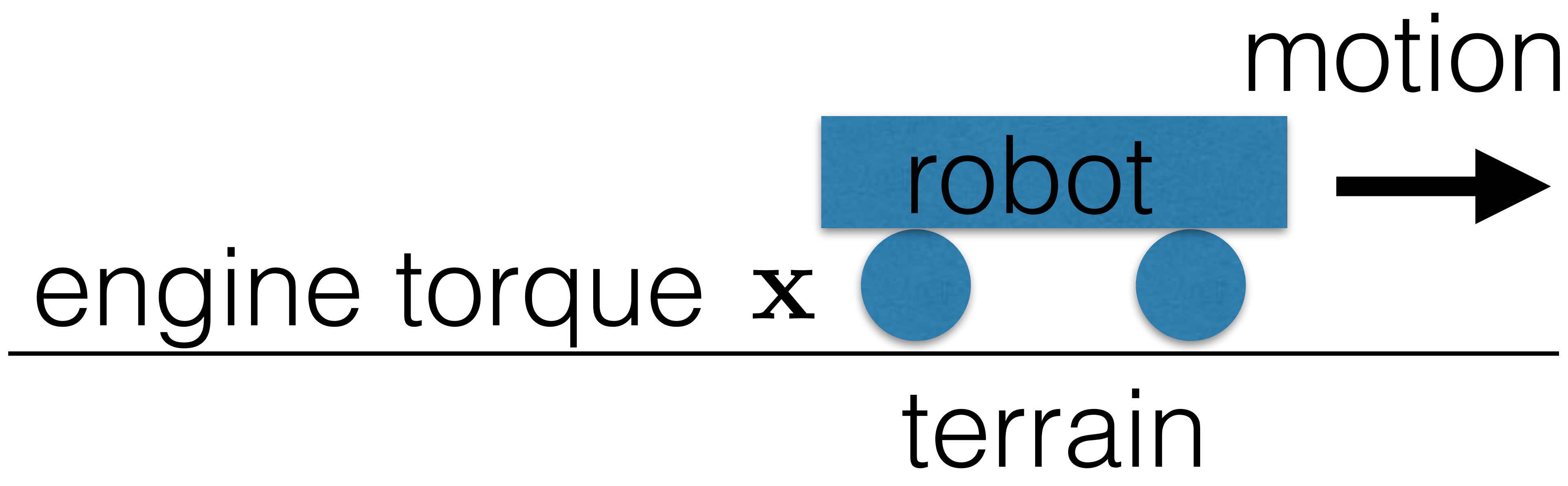
- Day/night, summer/winter, USA/China, young/old people, American/Indian English

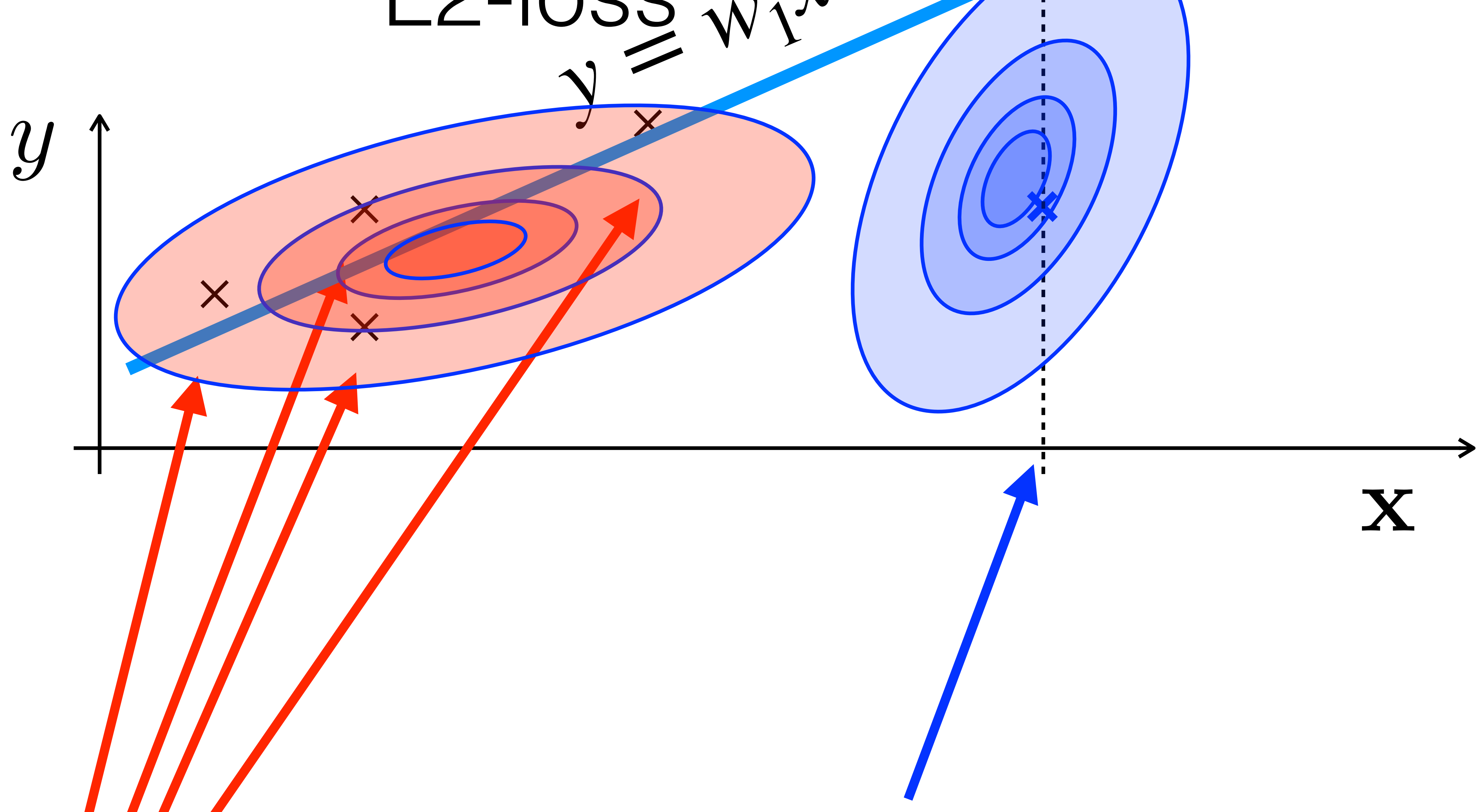
[NVidia, CVPR, 2016]

Does this suffer from trn/tst distribution mismatch?

Statistical consistency



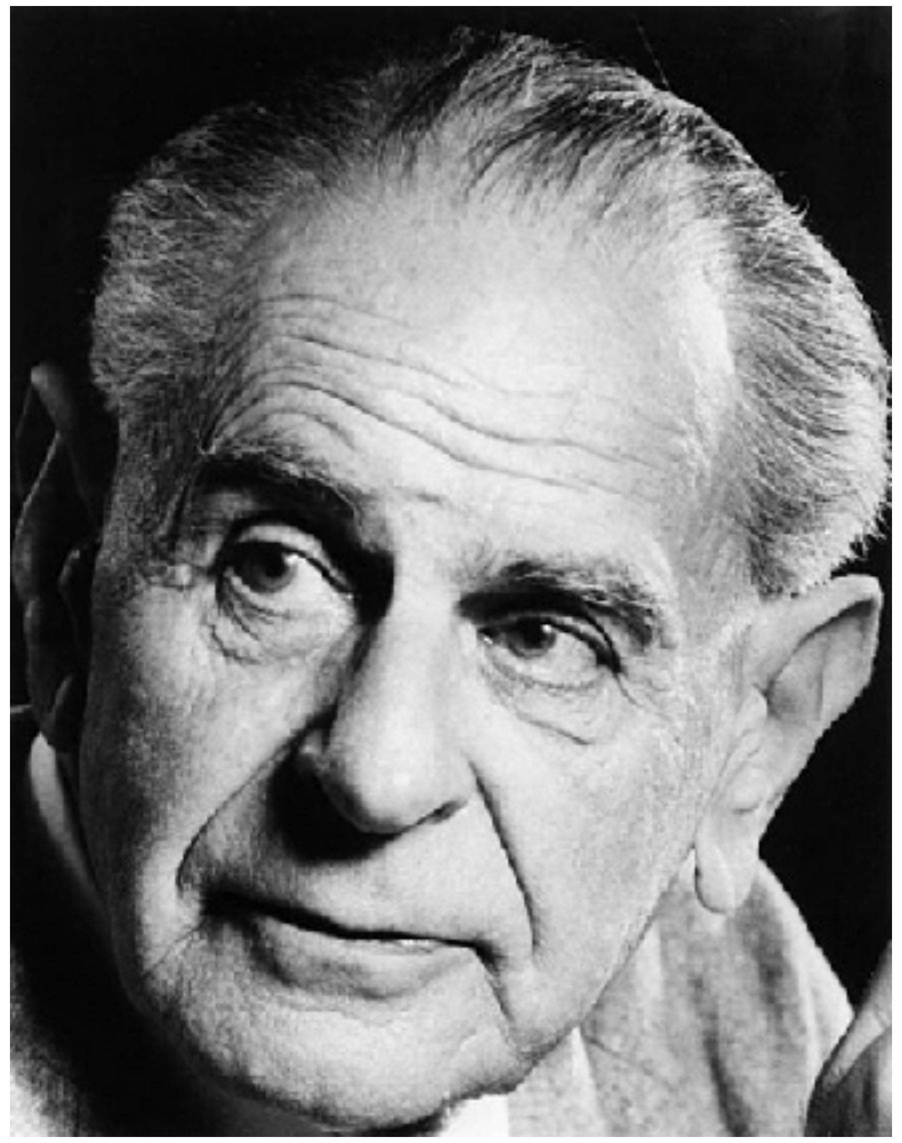




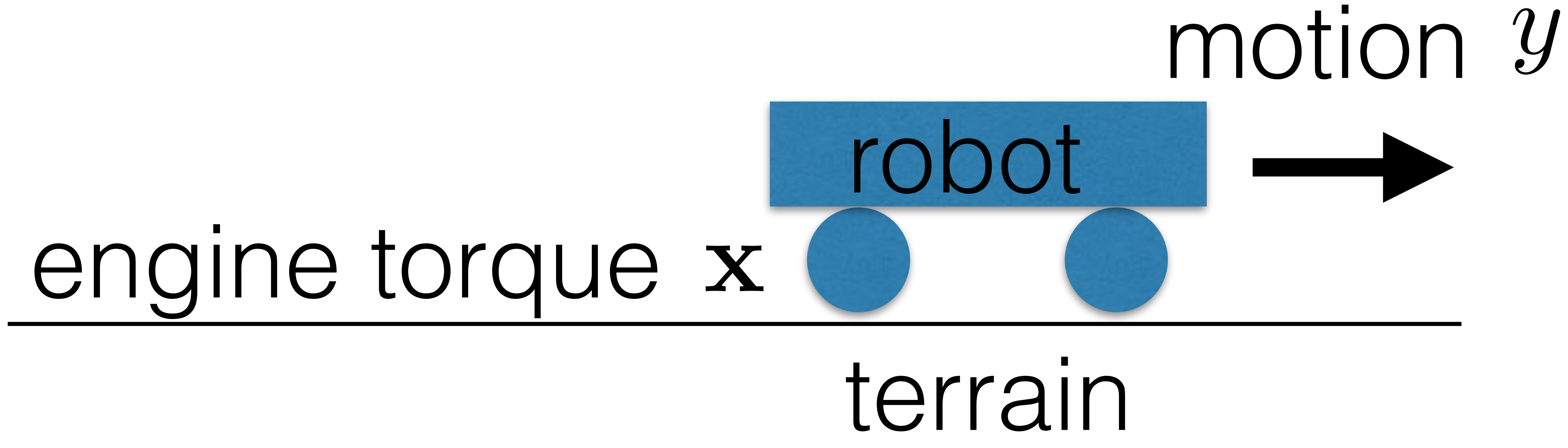
trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

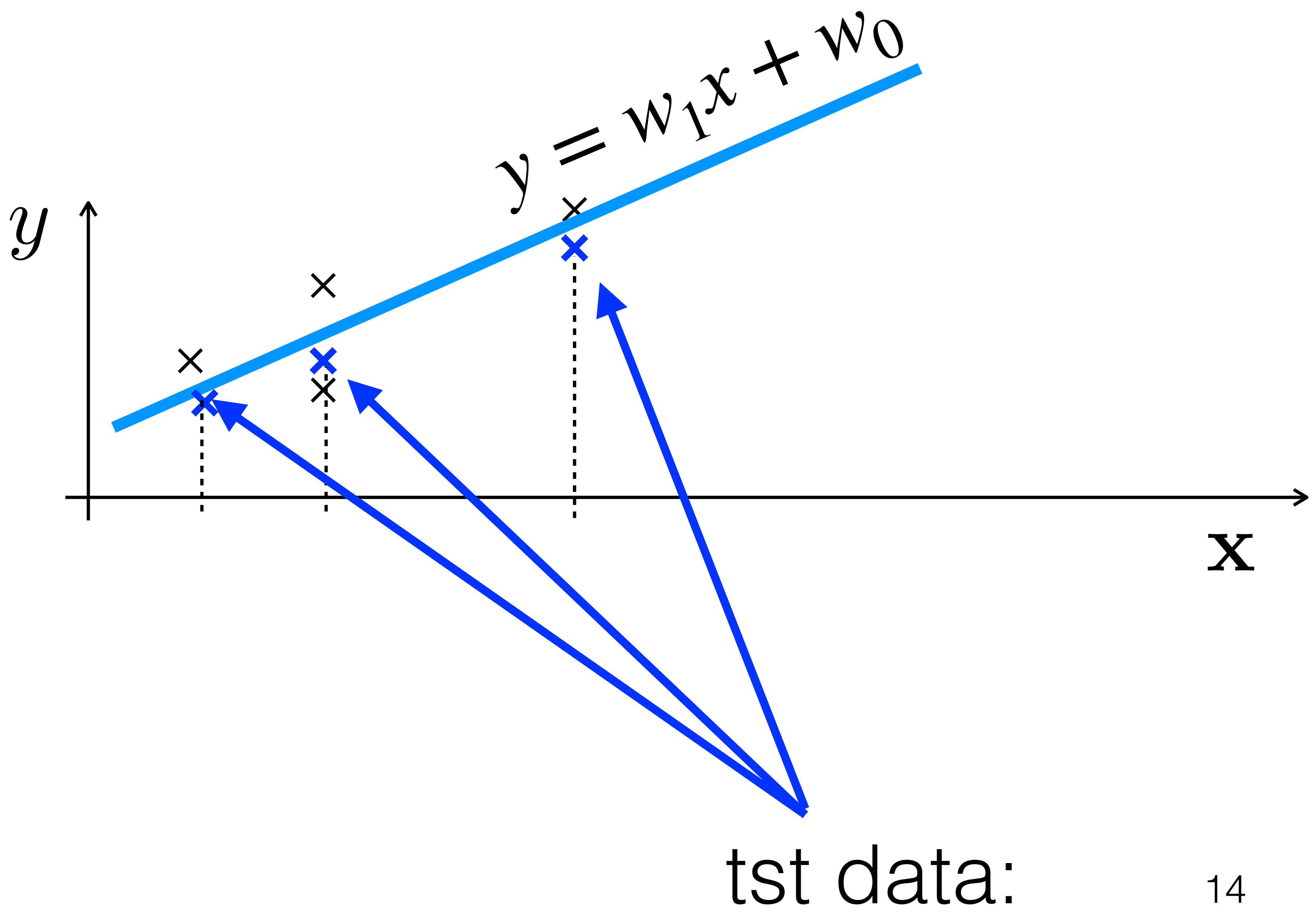
tst data:

What can go wrong: **trn/tst data are too similar**

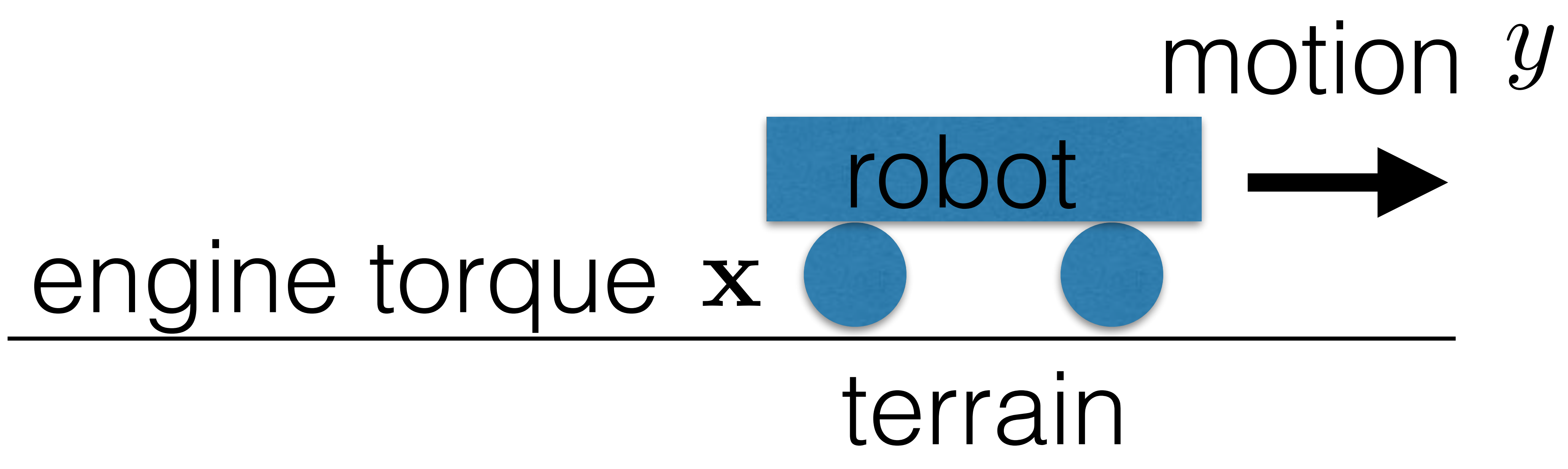


Humans subconsciously selects testing data that
that are consistent with their proposed solution.





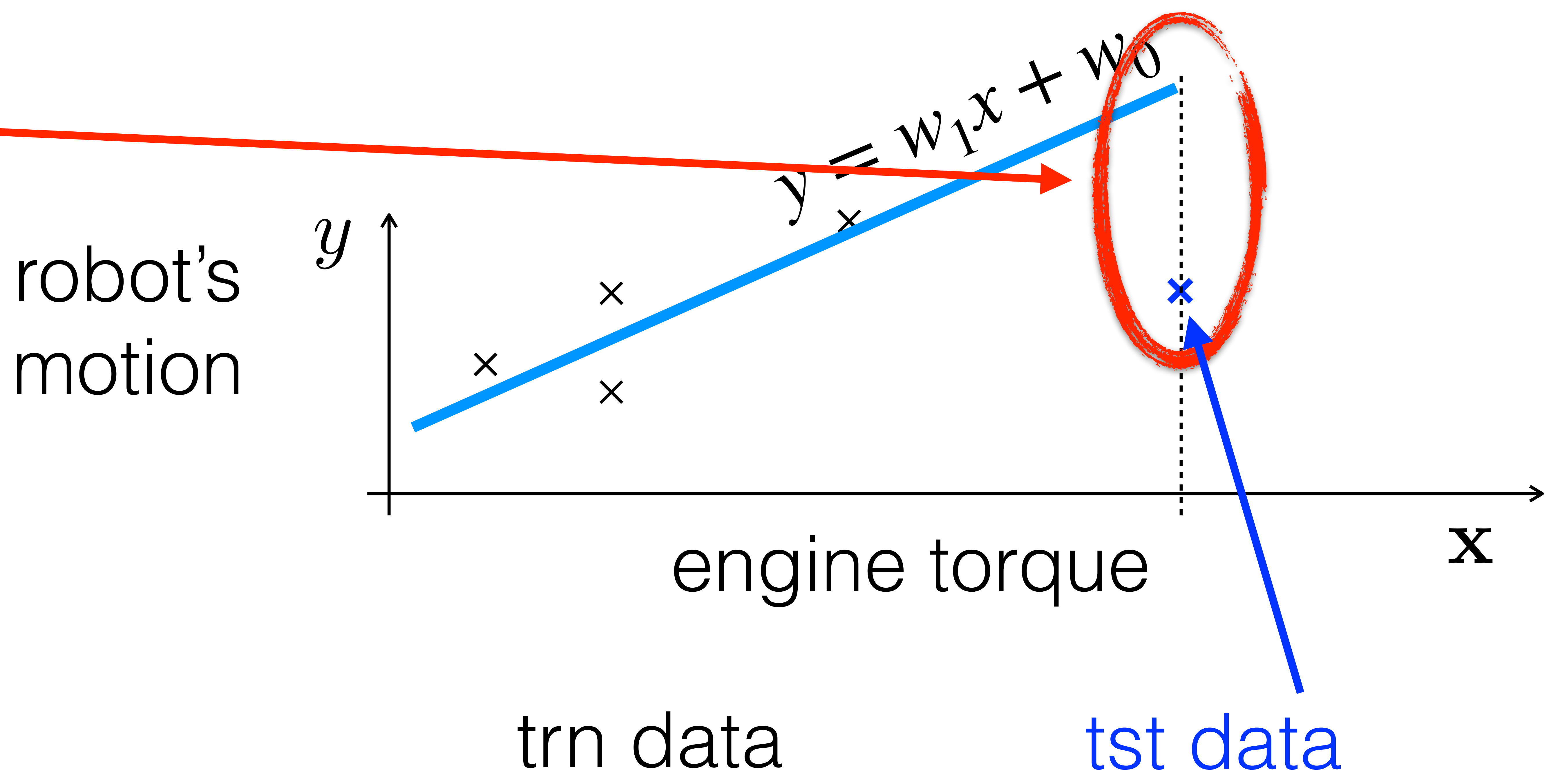
What can go wrong: **inappropriate model**



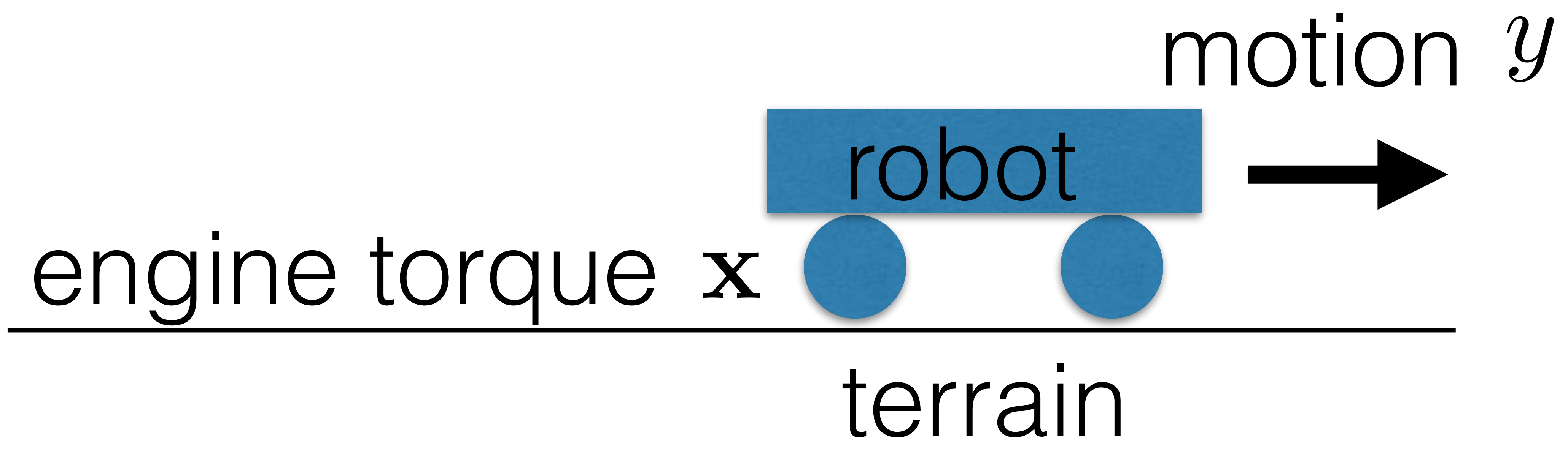
linear function => underfitting

Underfitting:

- bad generalization due to oversimiplified model



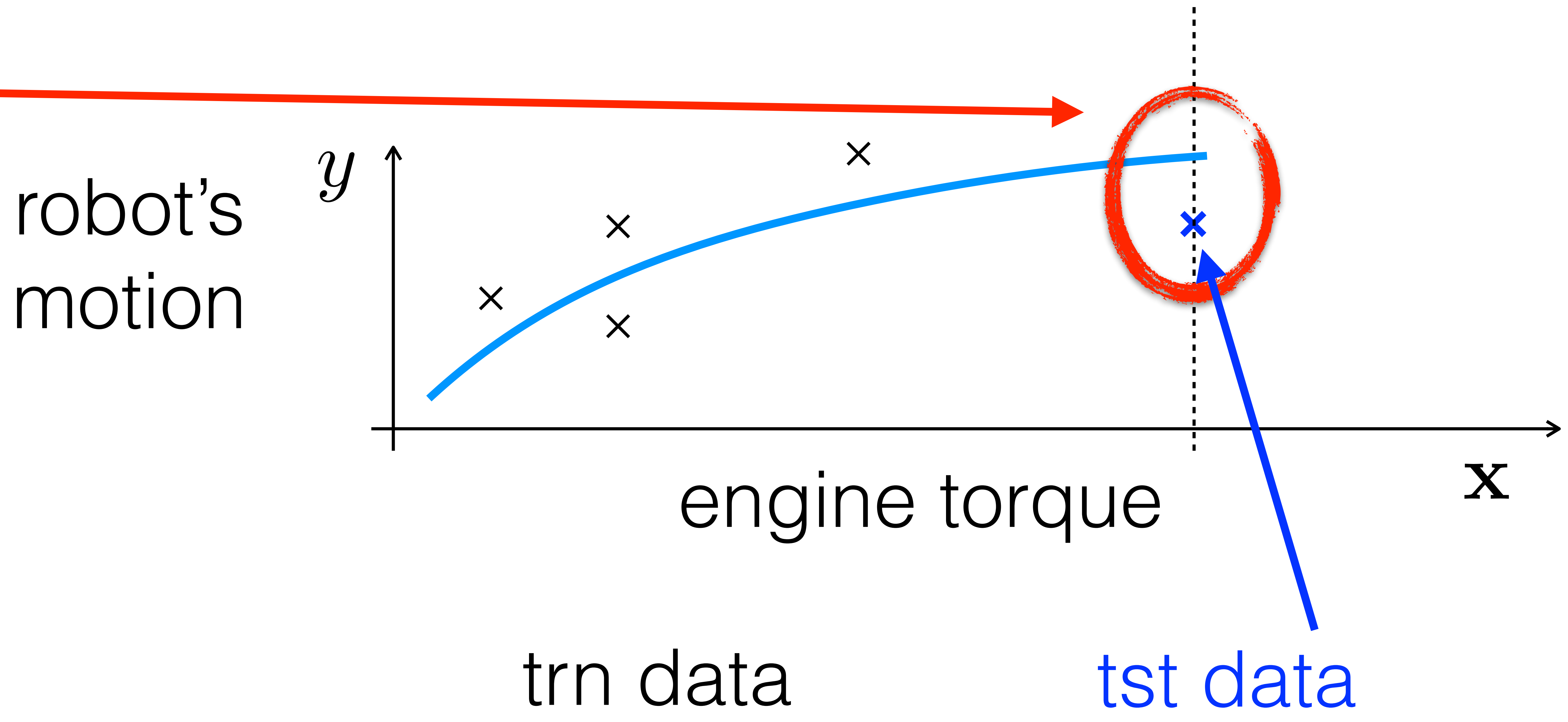
What can go wrong: **inappropriate model**



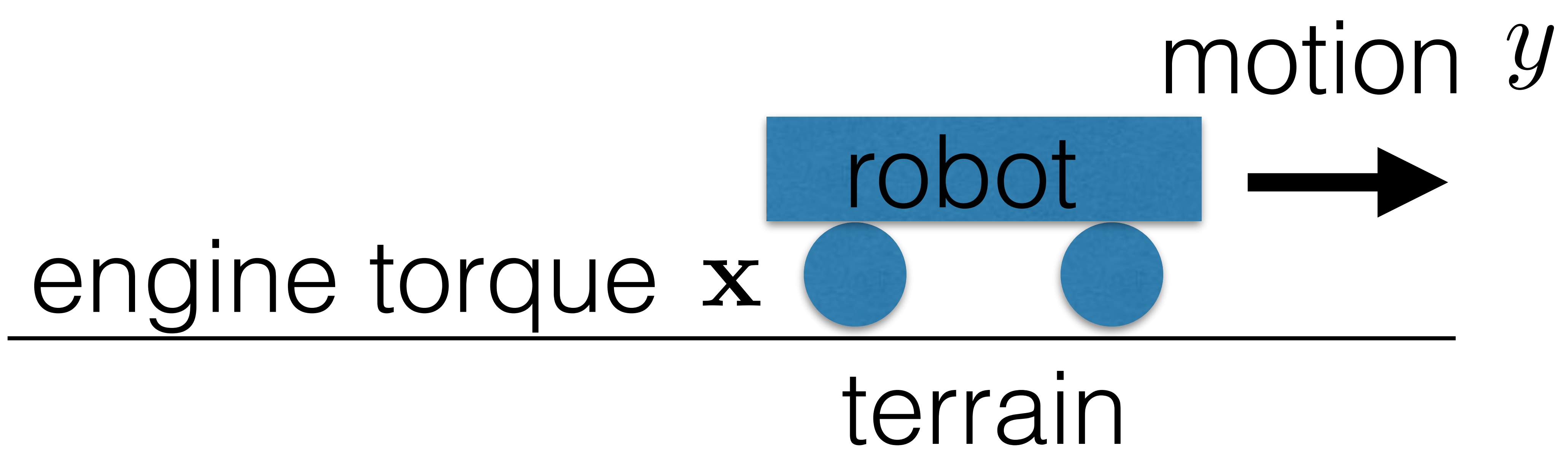
log function => good fit

Good model provides:

- good generalization
  (less sensitive to trn/tst mismatch)



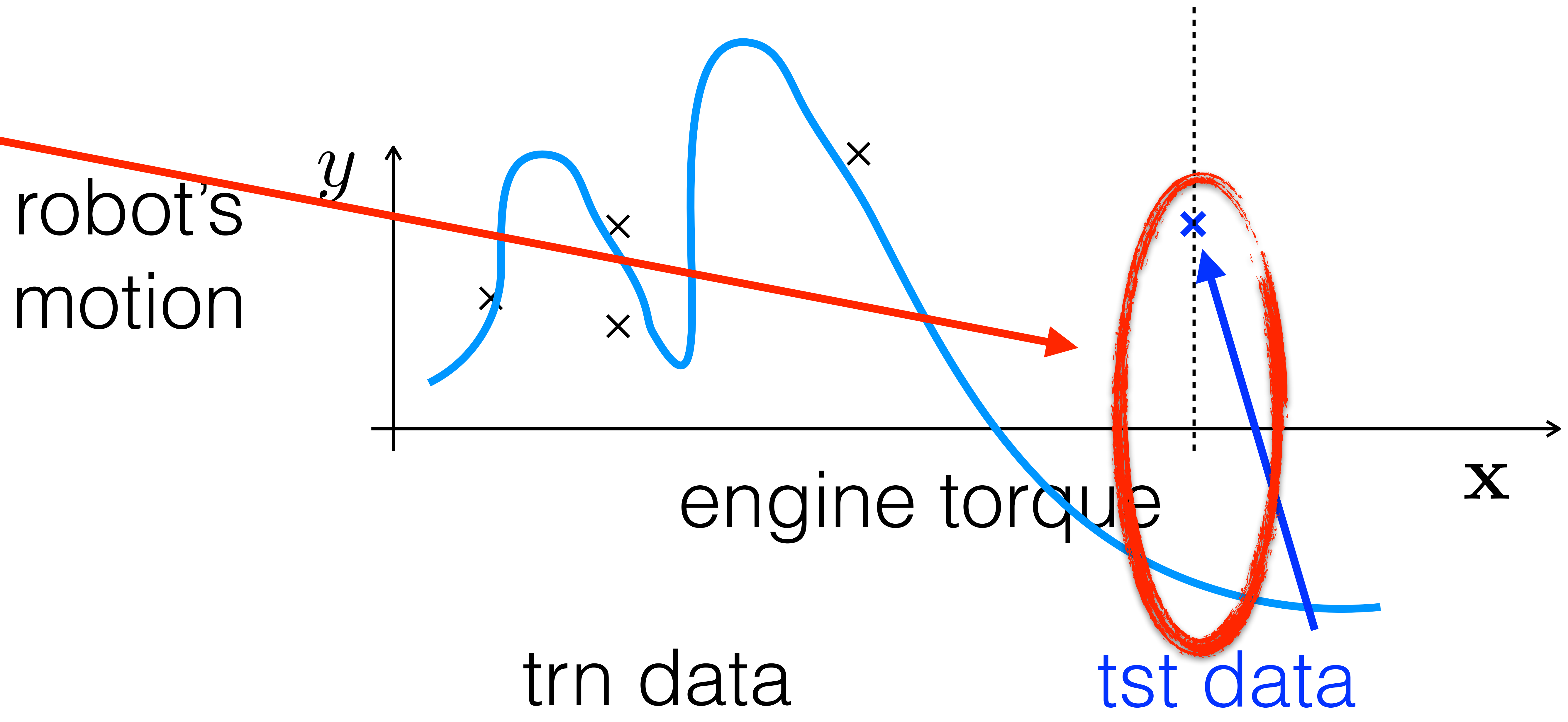
What can go wrong: **inappropriate model**



complicated function=>overfitting

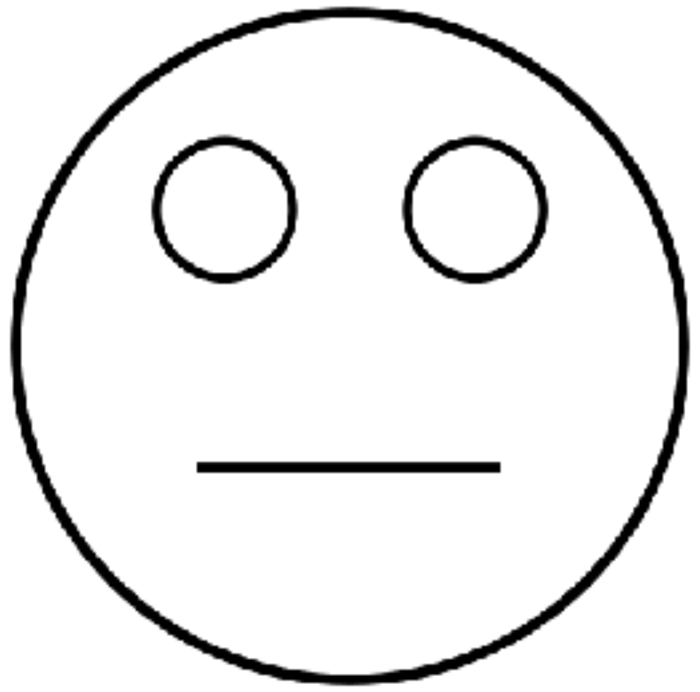
Overfitting:

- bad generalization due to overcomplex model



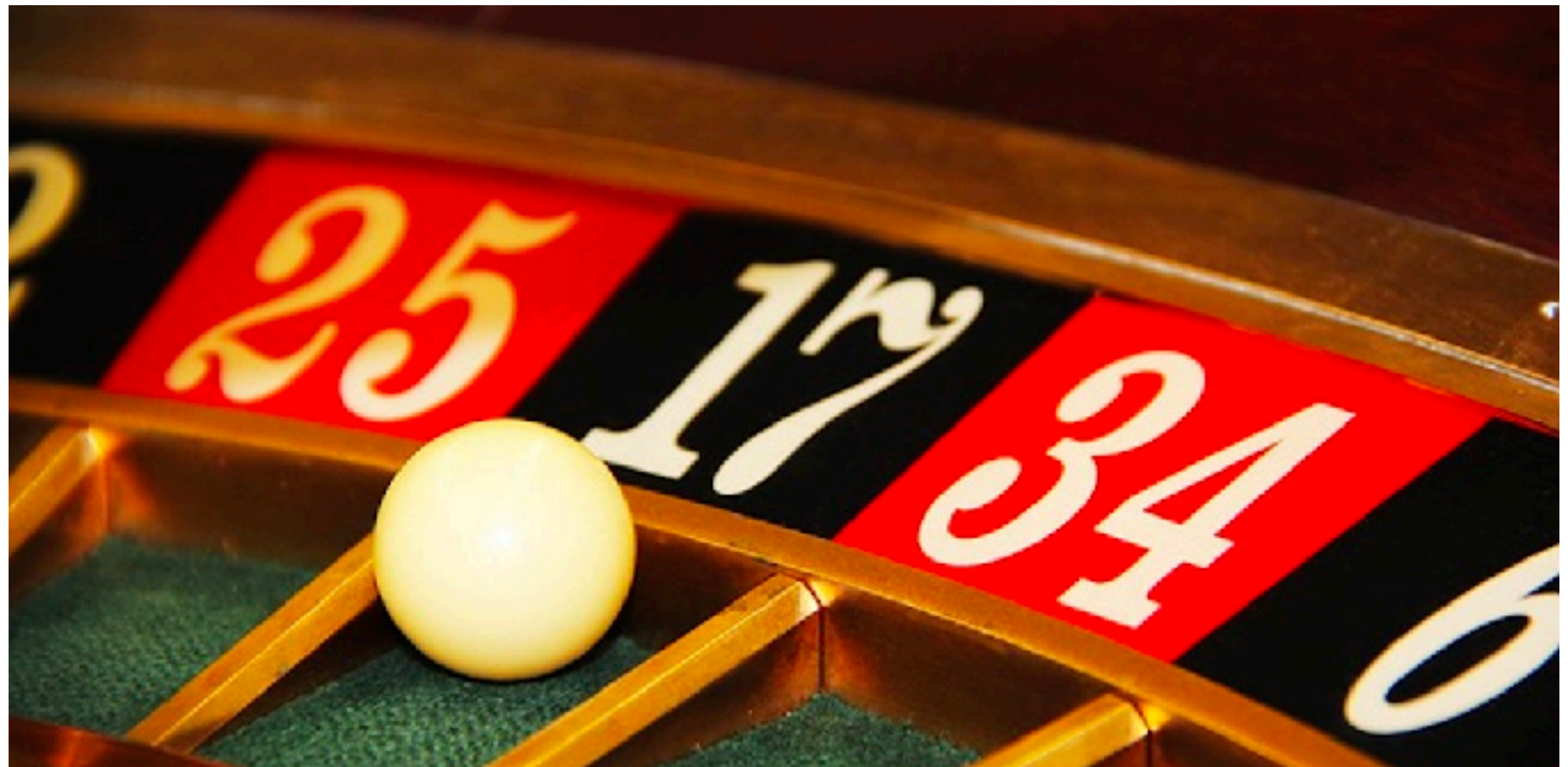
Do humans overfit?

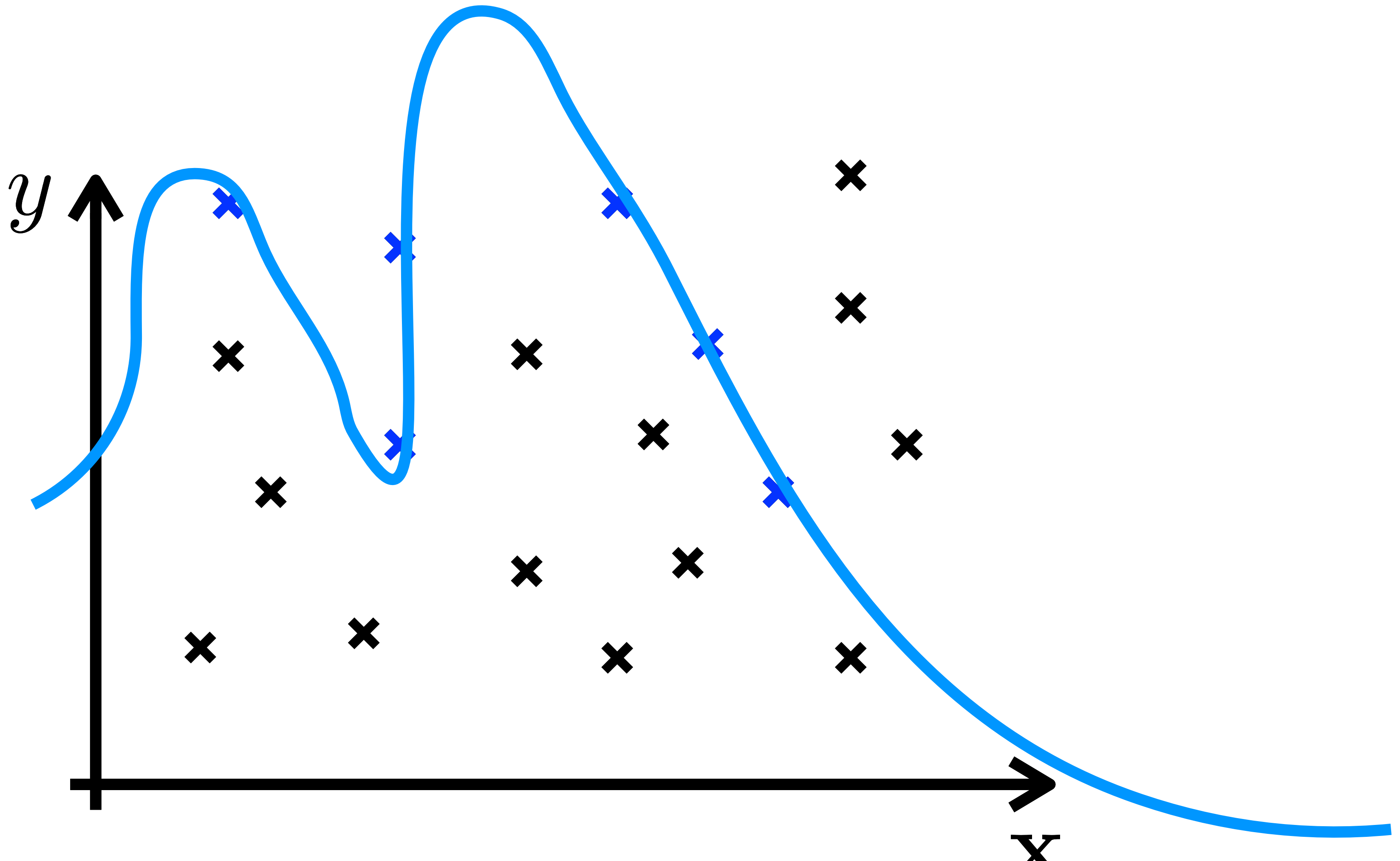
What can go wrong: **inappropriate model**



Do humans overfit?

Apofenia=human overfitting

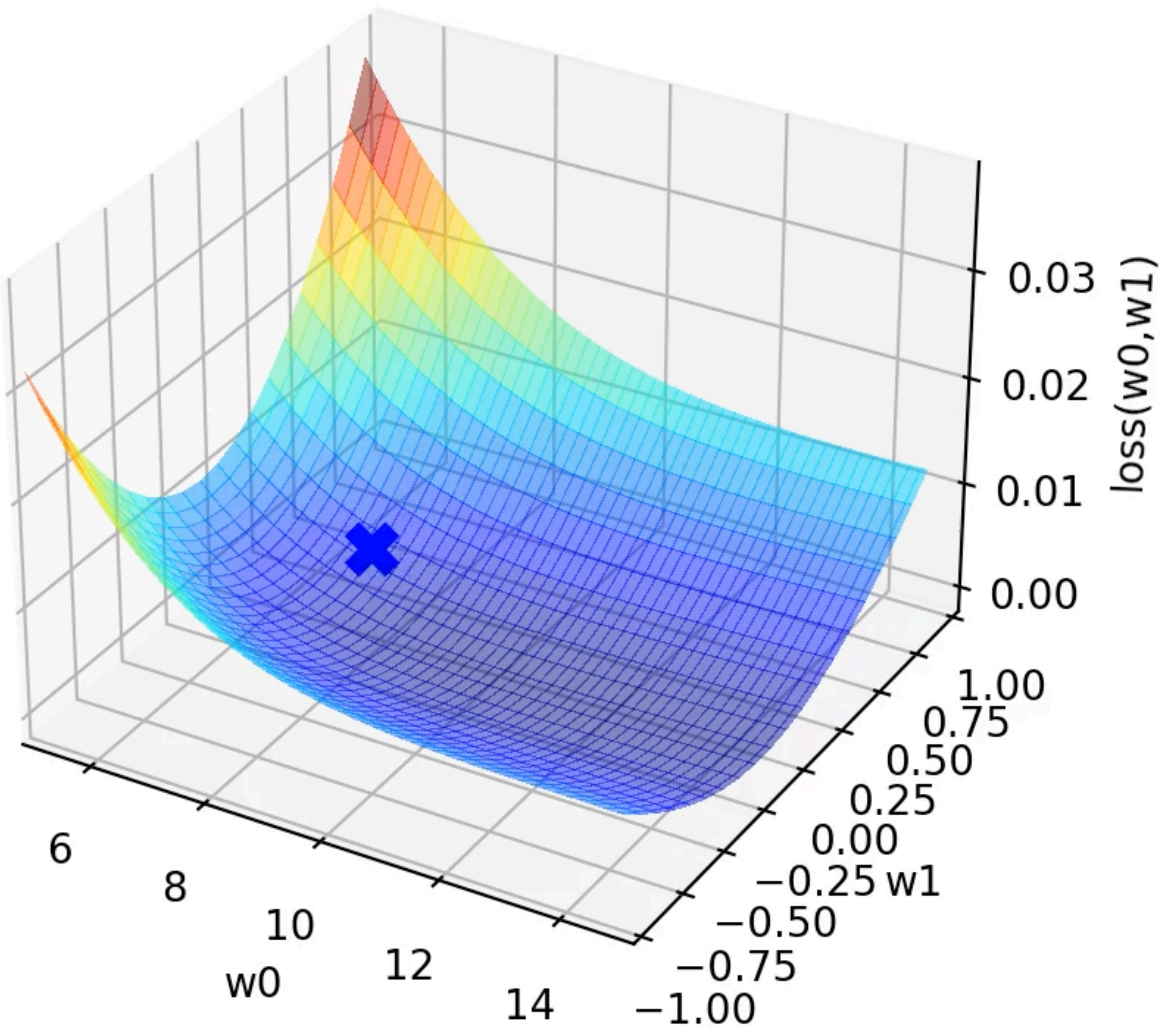




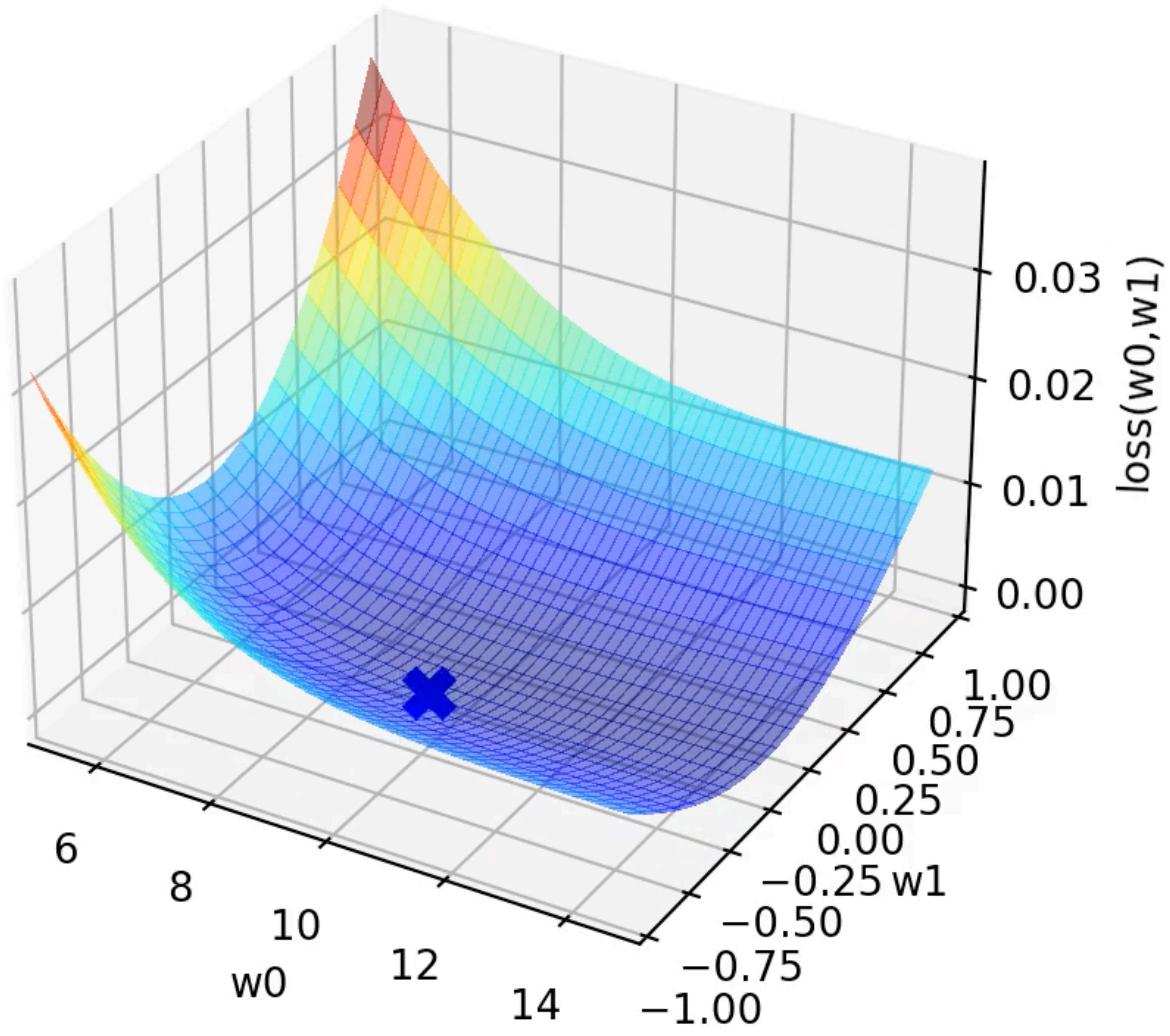
What can go wrong: **learning fails to find good model parameters**

due to hyper-parameters, local optima, bad initialization …

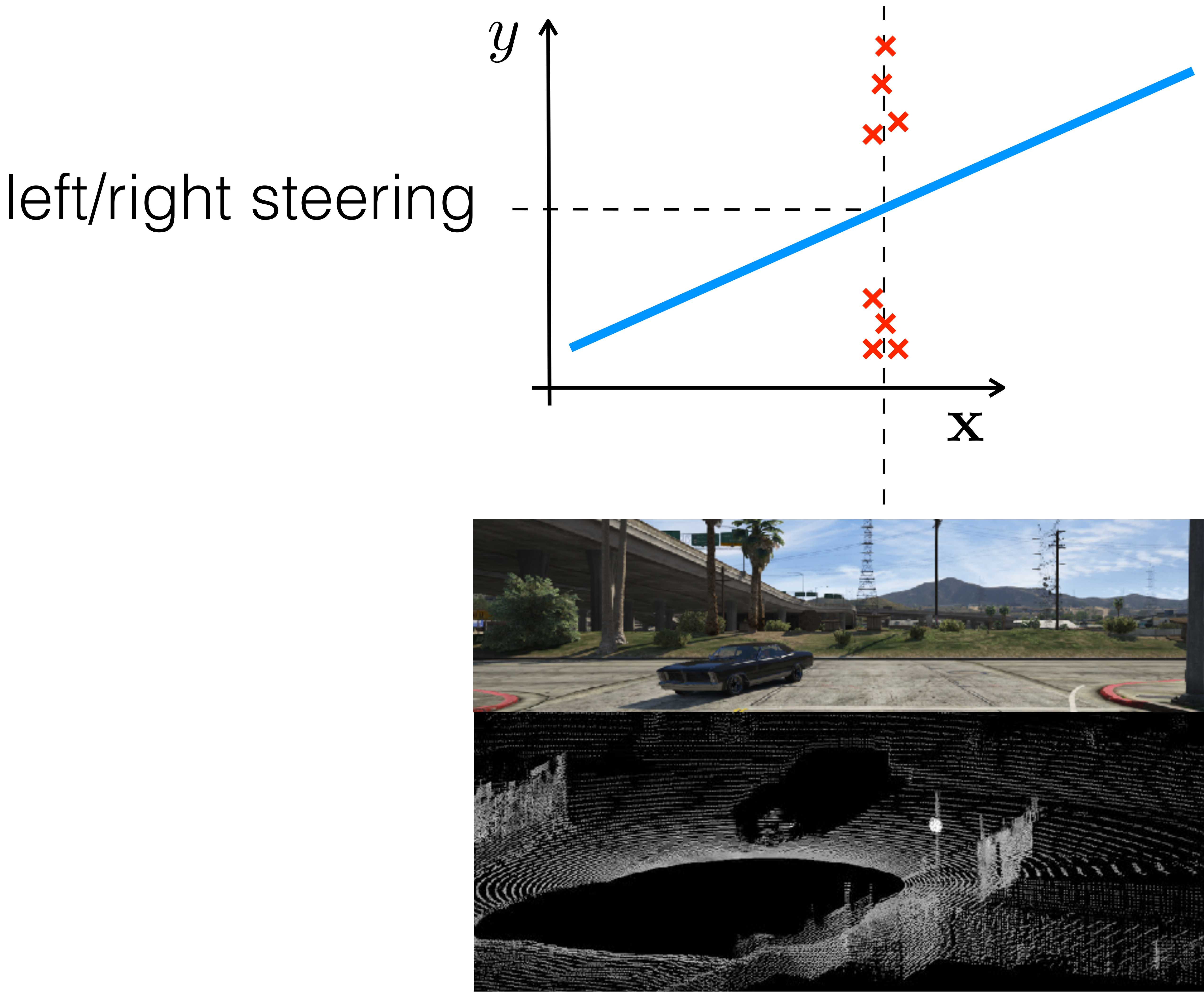
reasonable learning rate

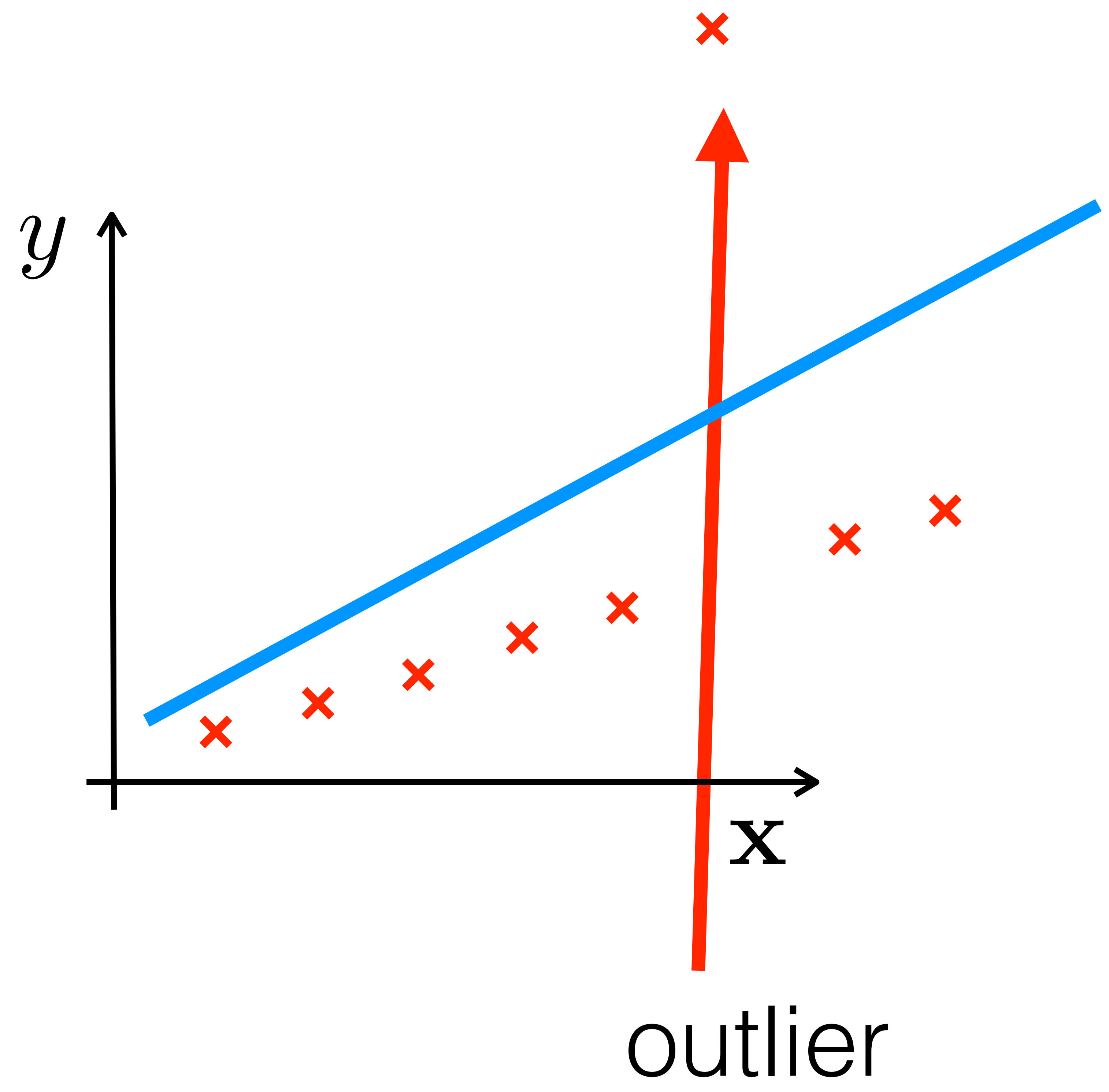


too big learning rate

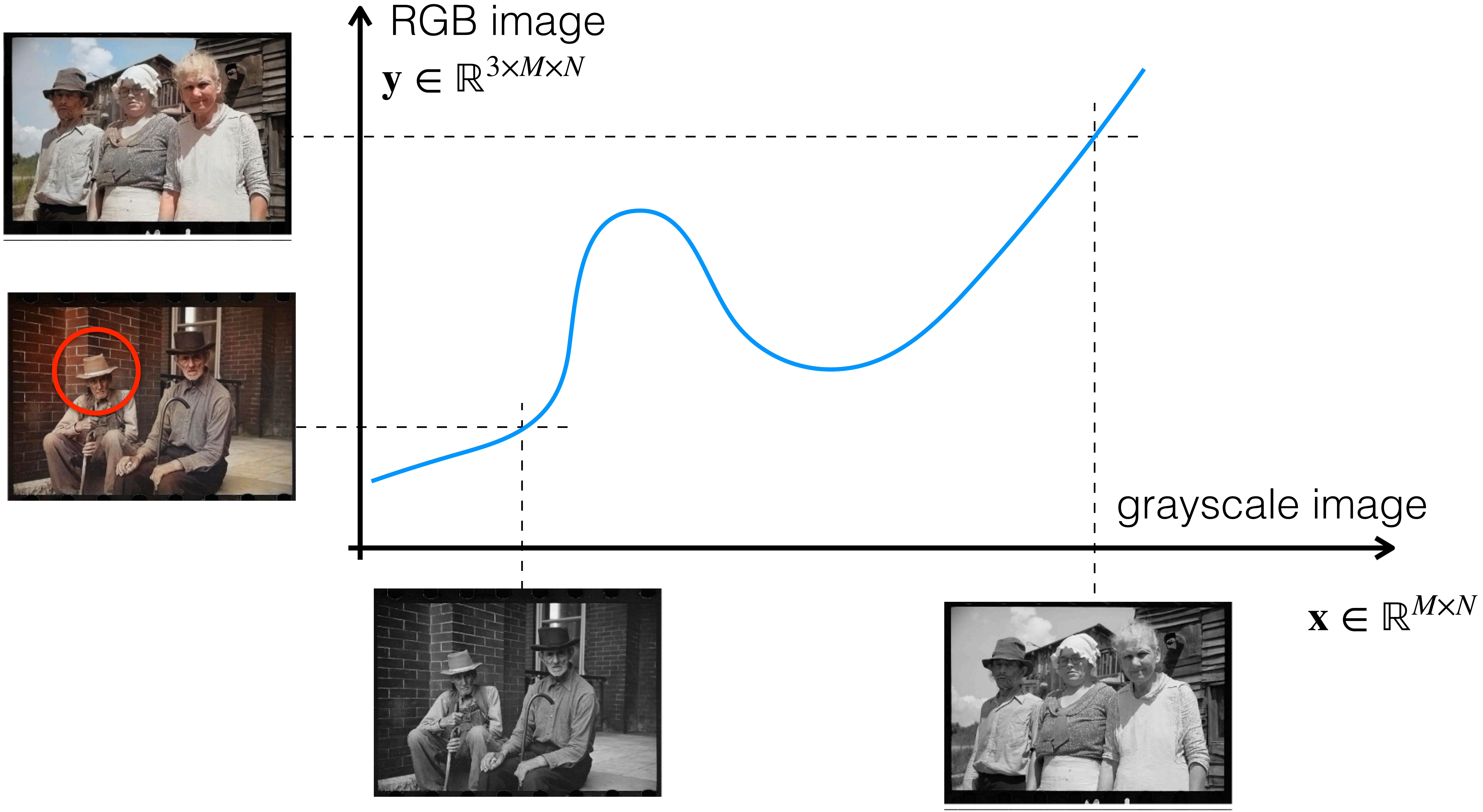


What can go wrong: **inappropriate choice of loss function**



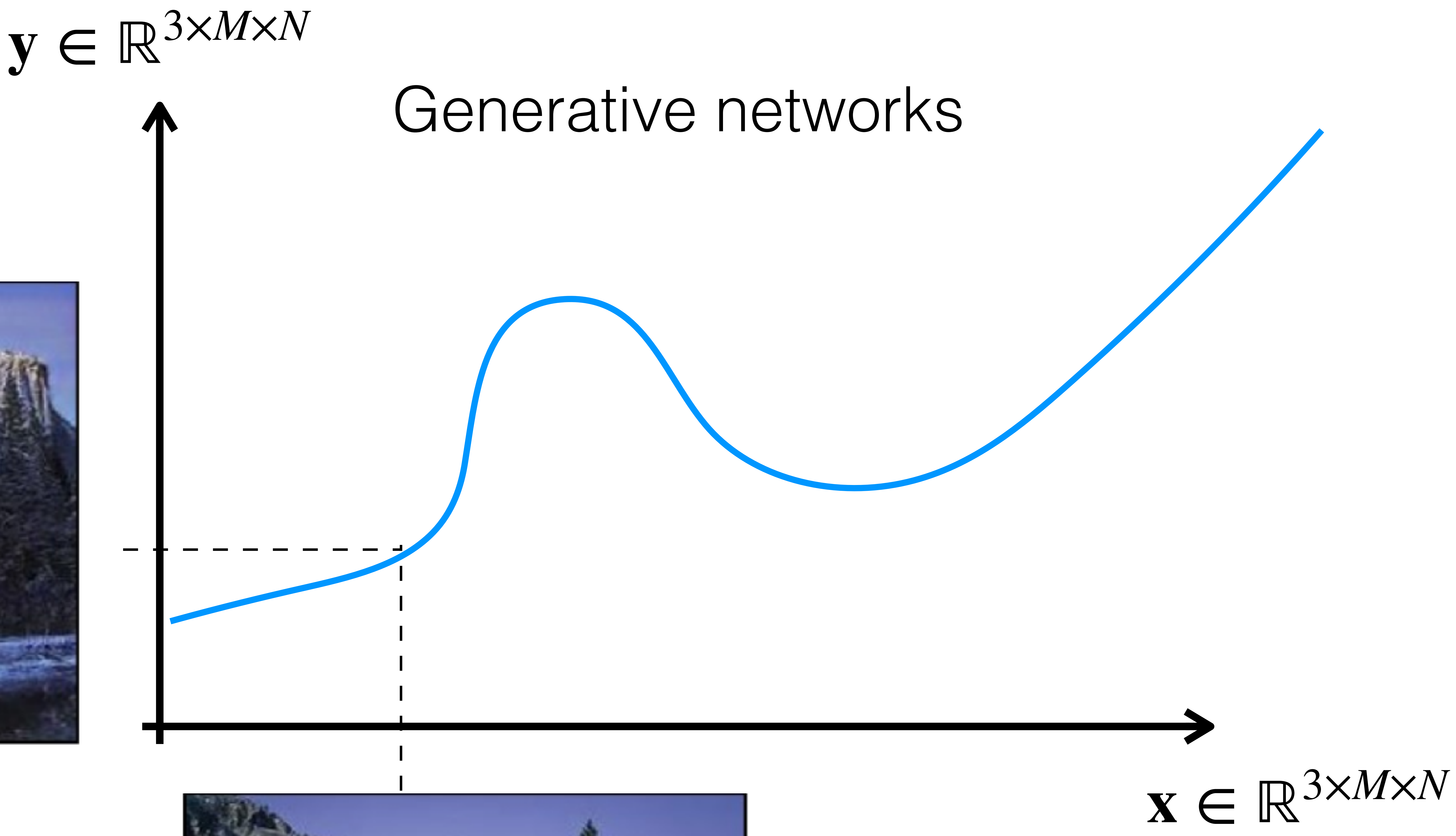


What can go wrong: **inappropriate choice of loss function**



What can go wrong: **inappropriate choice of loss function**

winter image

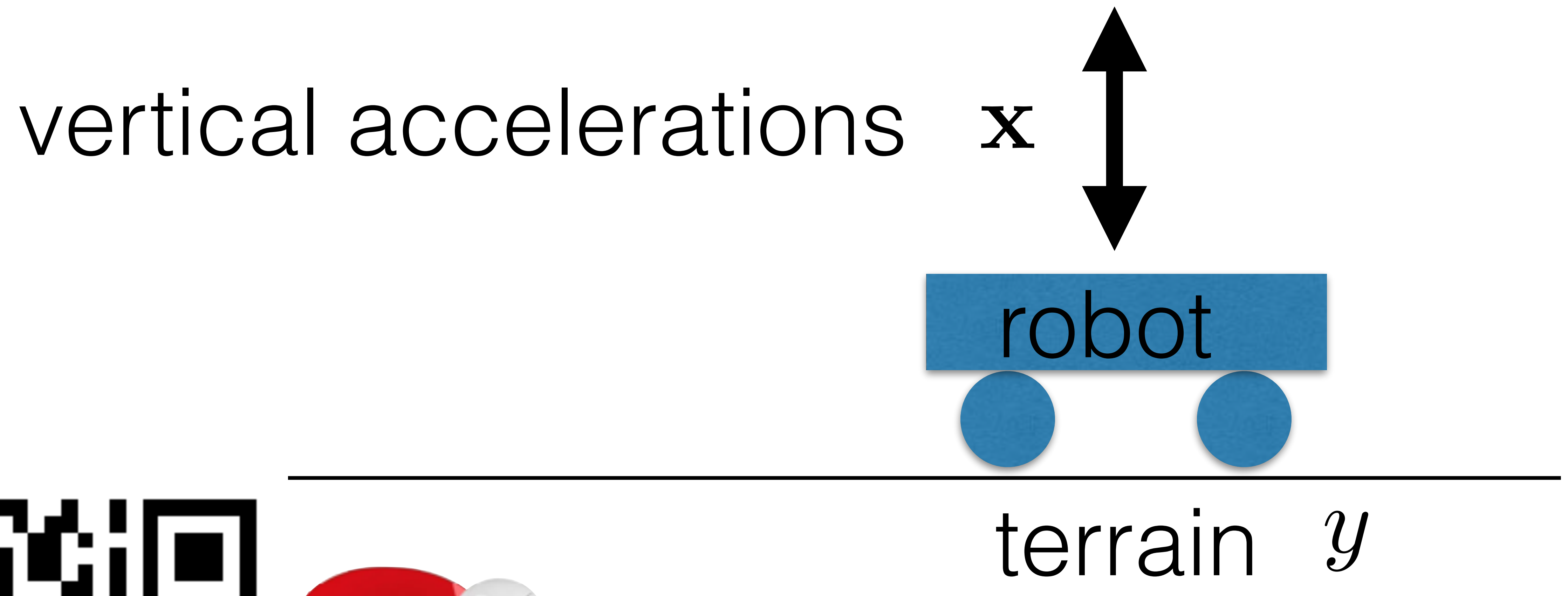


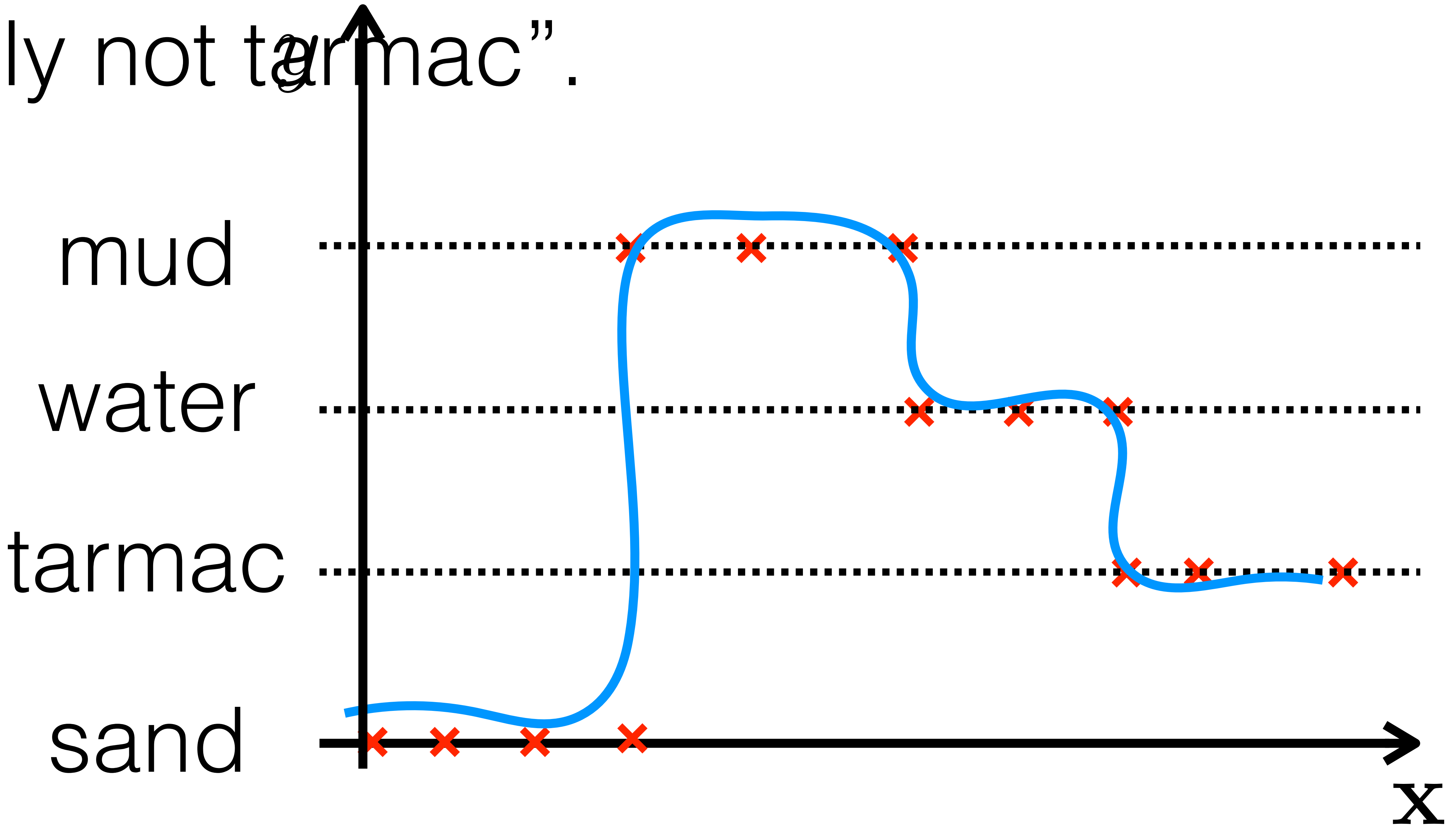
summer image

What can go wrong: **inappropriate choice of architecture**

Motivation example: classification

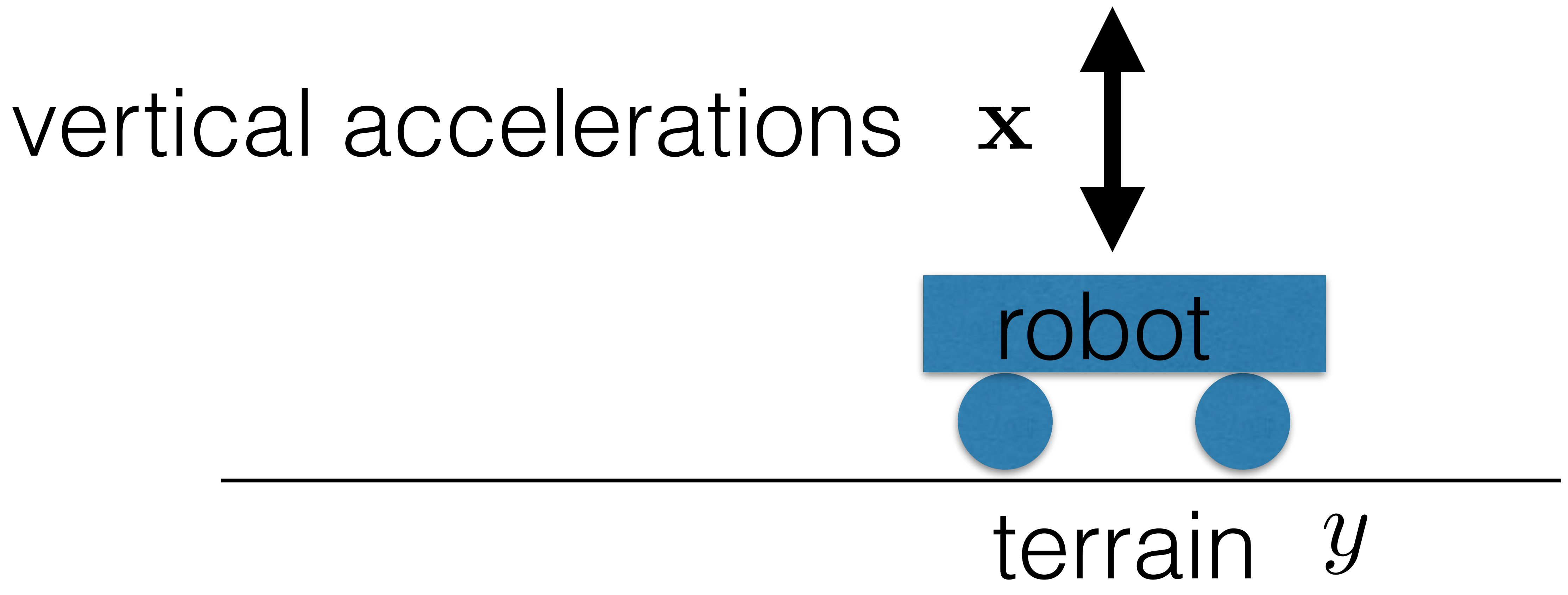
- Can I treat problem as regression?
- Suffers from:
  - complicated optimization,
  - enforced ordering
    - loss for misclasifying mud-to-water << mud-to-sand)
    - cannot model: “mud or sand but definitely not tarmac”.

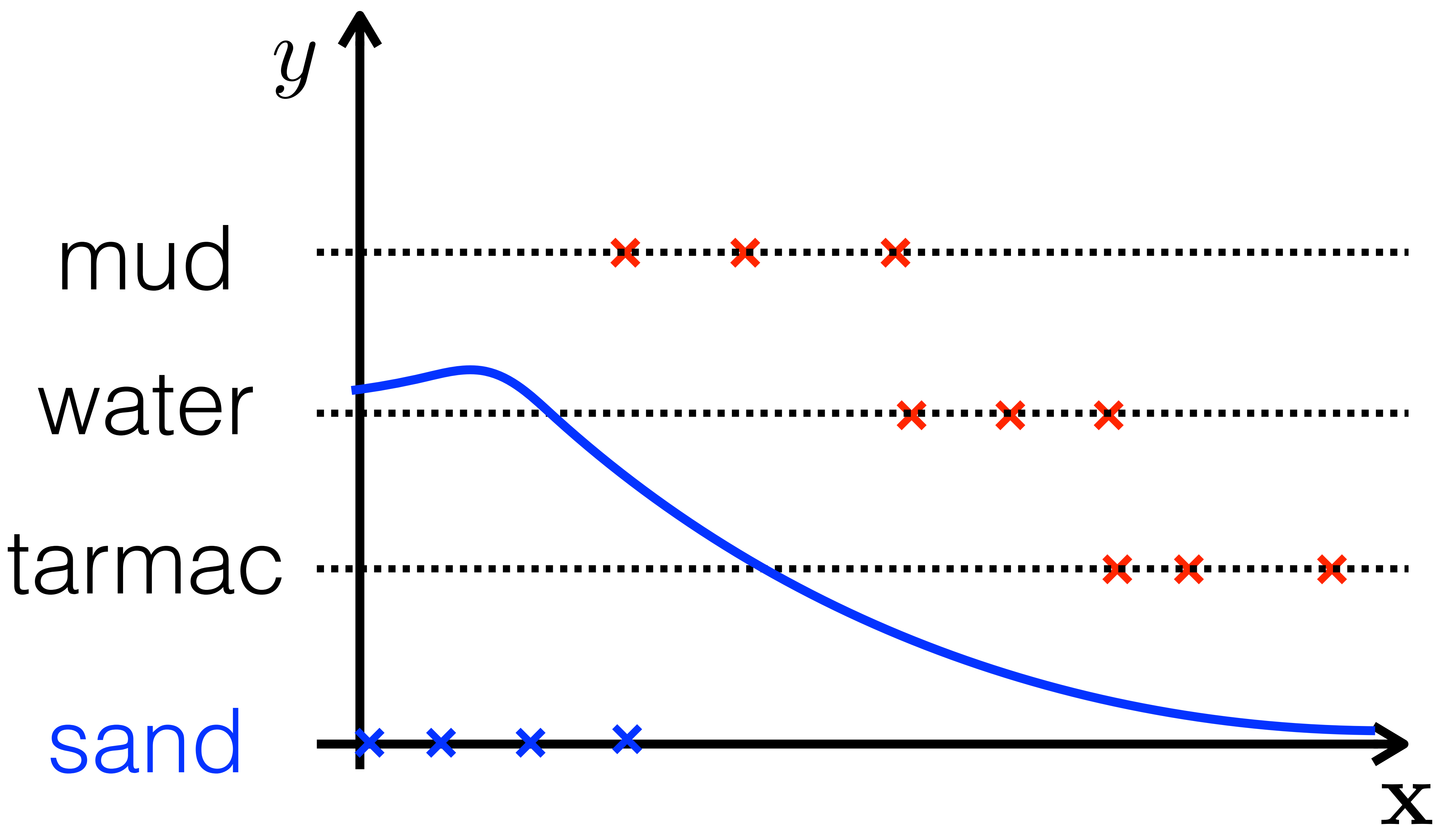




trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

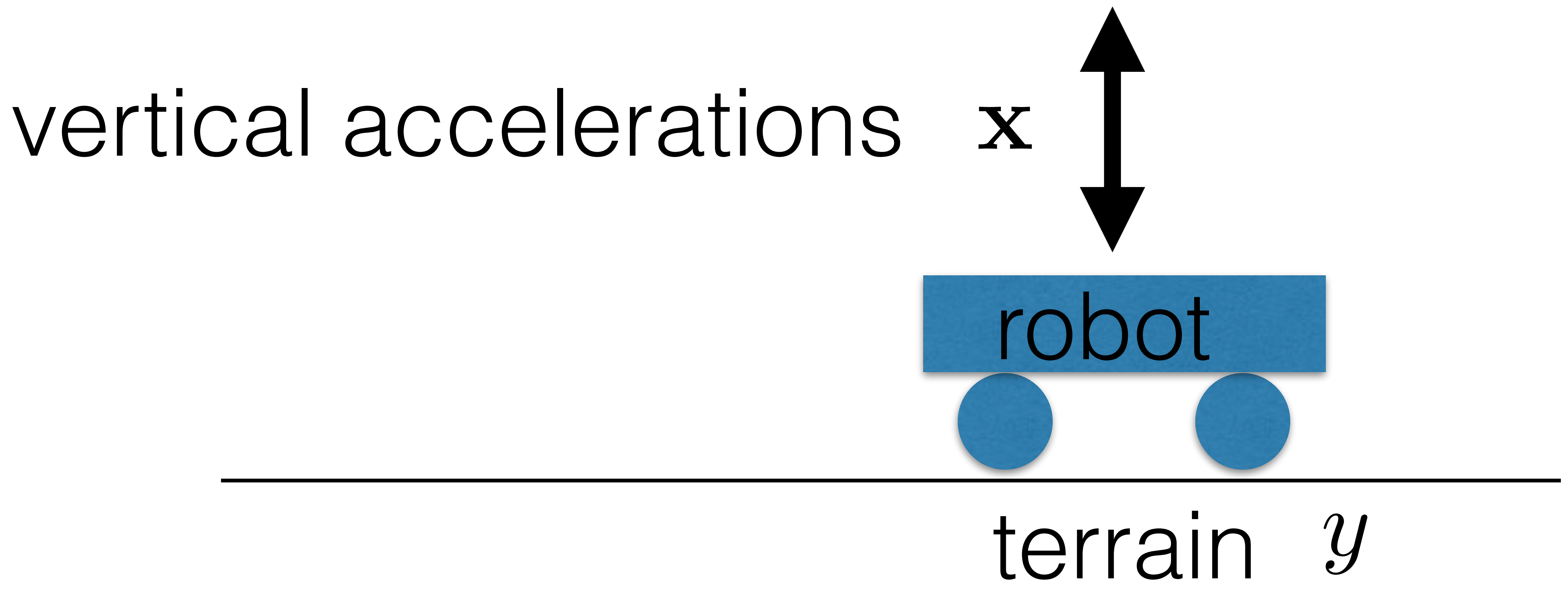
Motivation example: classification

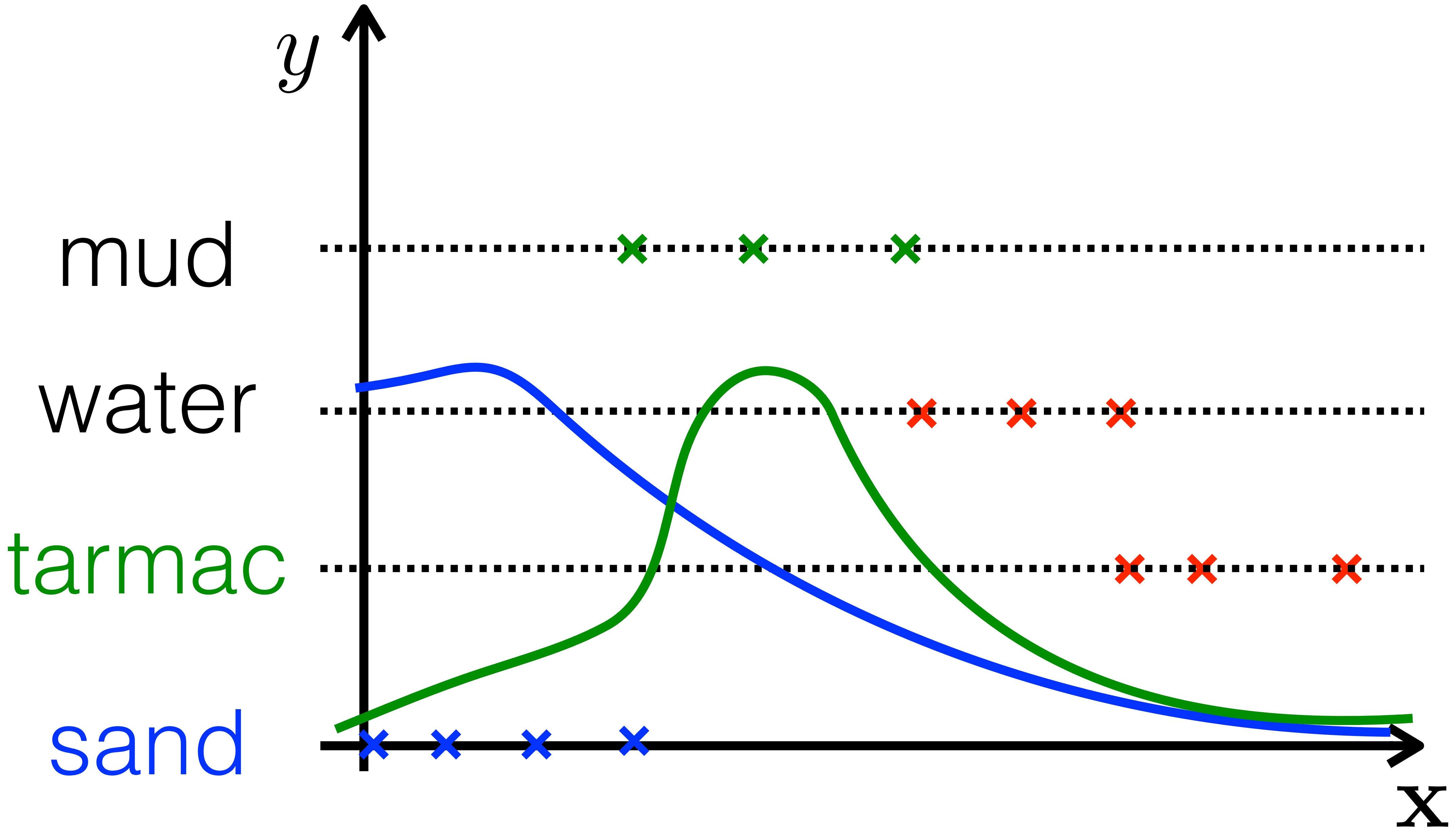




trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

Motivation example: classification

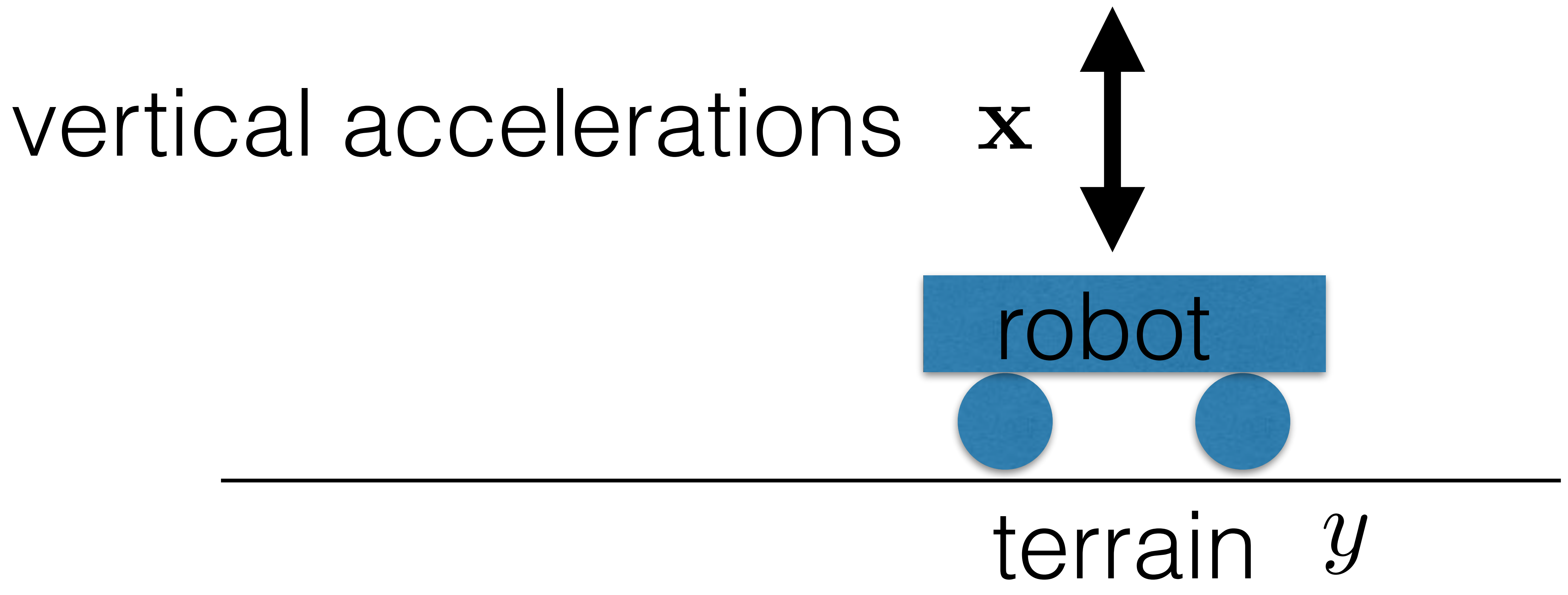


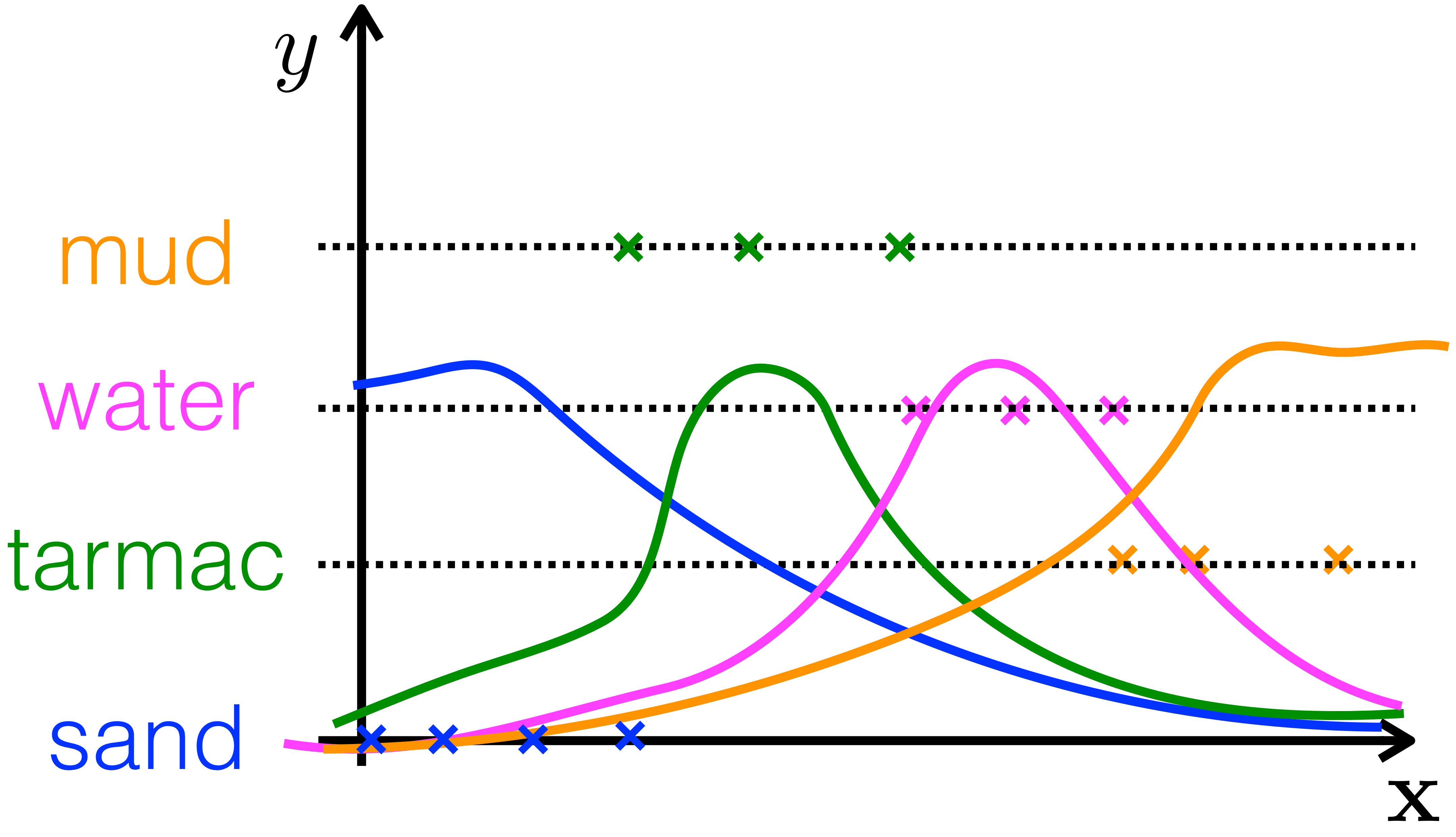


trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

Motivation example: classification

- 4 functions predicting class probabilities $p(y_1 | x, w_1),\ p(y_2 | x, w_2),\ \ldots$

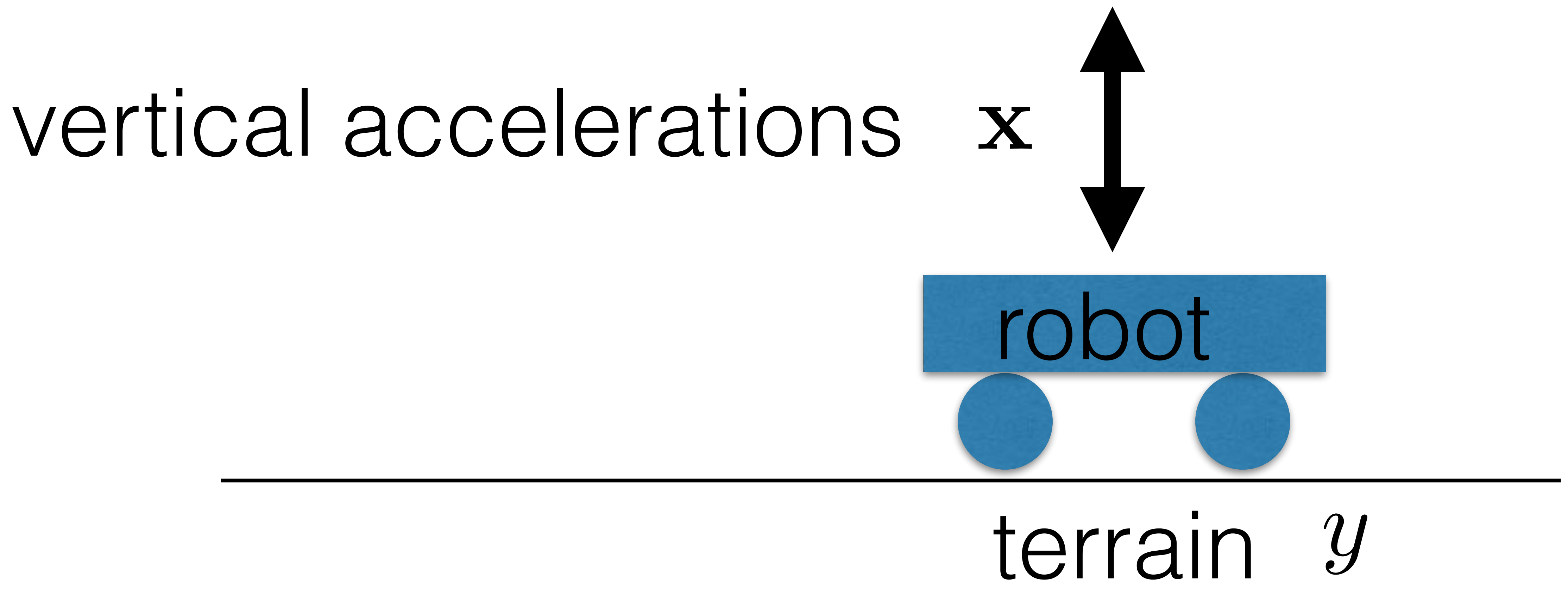


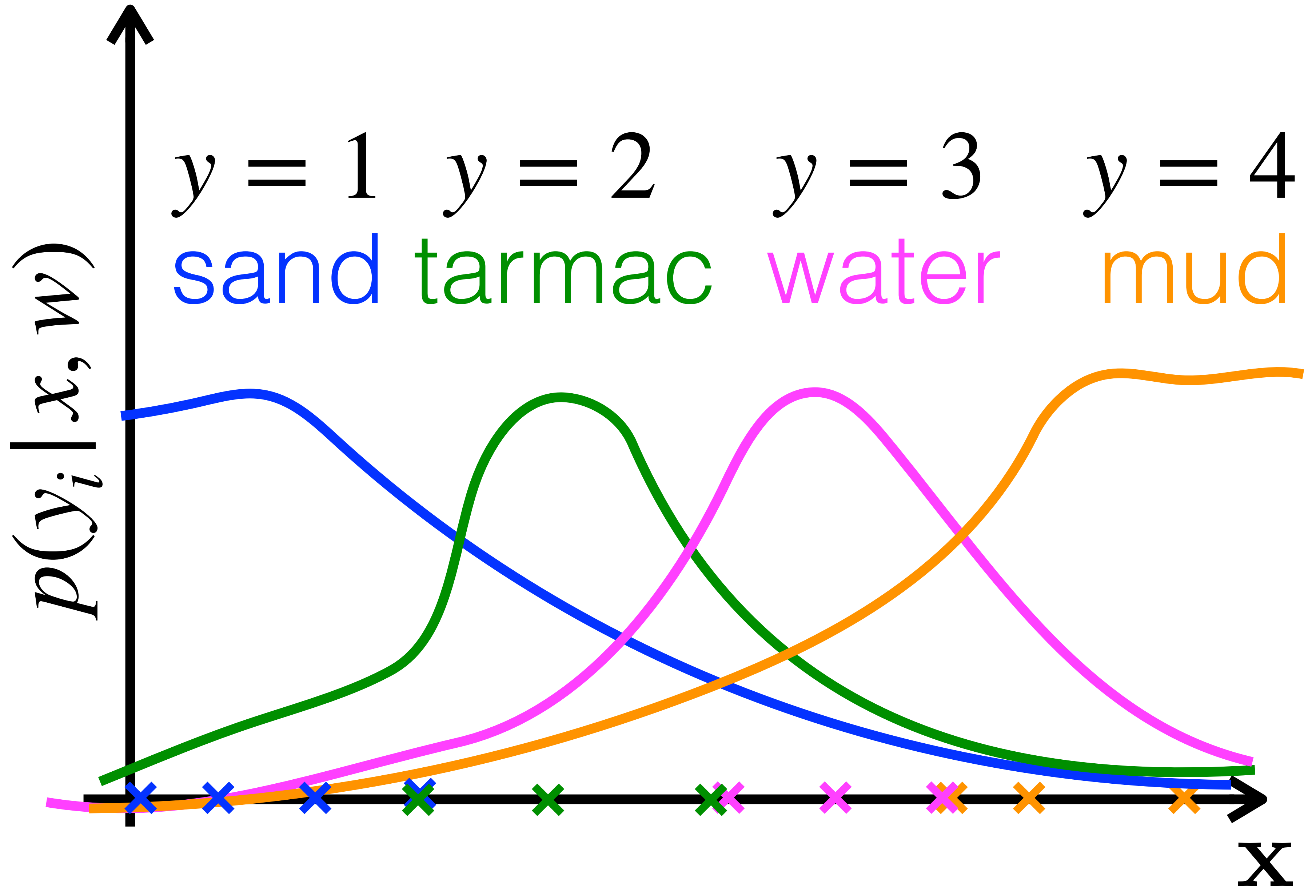


trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

Motivation example: classification

- 4 functions predicting class probabilities $p(y_1|x, w_1),\ p(y_2|x, w_2),\ \ldots$
- that sum up to one over classes for given x $\sum_i p(y_i|x, w) = 1$
- Learning:
  - Substitute blue points to blue function, etc… => "p"
  - Define loss that pushes "p"-values up
- Can you guess a suitable shape of loss function?
  - loss = -log(p)





trn data: $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

Motivation example: classification

Competencies required for the test T1

- Model (or Architecture/Program) with parameters => learning
- Learning = loss + trn data + optimization procedure
- Evaluation = measuring performance (not necessary loss) on tst data
- What could go wrong?
  - inputs x does not allow to predict y
  - trn/tst data distribution mismatch
  - model does not generalize well
  - learning fails to find good parameters
  - inappropriate choice of loss function
  - inappropriate choice of architecture (overfit, underfit, regression/classification)
- Regression vs Classification
- **Next lecture:** Linear classification of RGB images