# Where the hell does the loss come from?

## MAP, MLE view of regression, classification and other problems

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

Procrastination idea: come up with original logo
https://beta.dreamstudio.ai/dream

Pre-requisites:

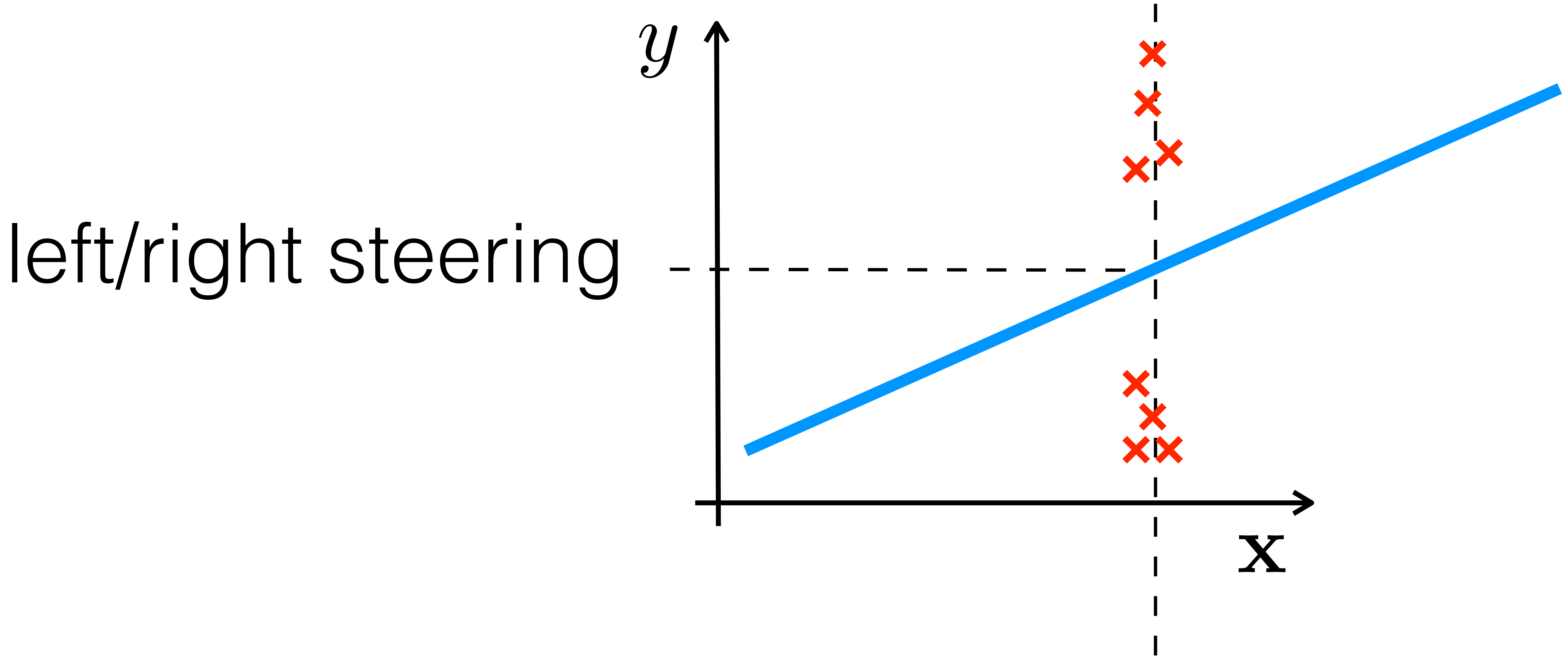
- linear algebra,
- probability laws:
  - Bayes rule: $p(A,B) = p(B,A) = p(A|B)*p(B) = p(B|A)*p(A)$
  - Independence of A and B: $p(A,B) = p(A)p(B)$

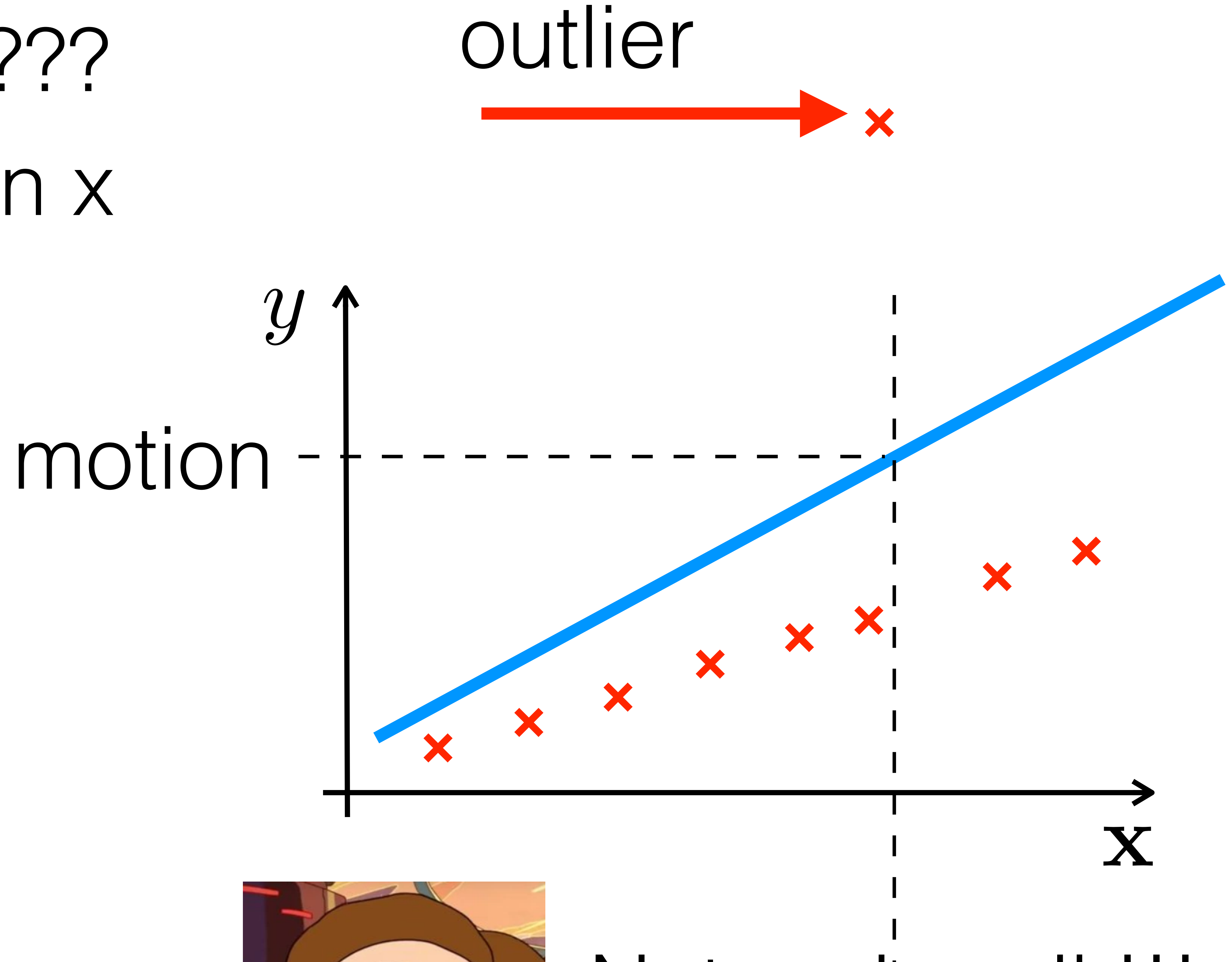
This lecture is (at least for me) take-home message of this course.

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

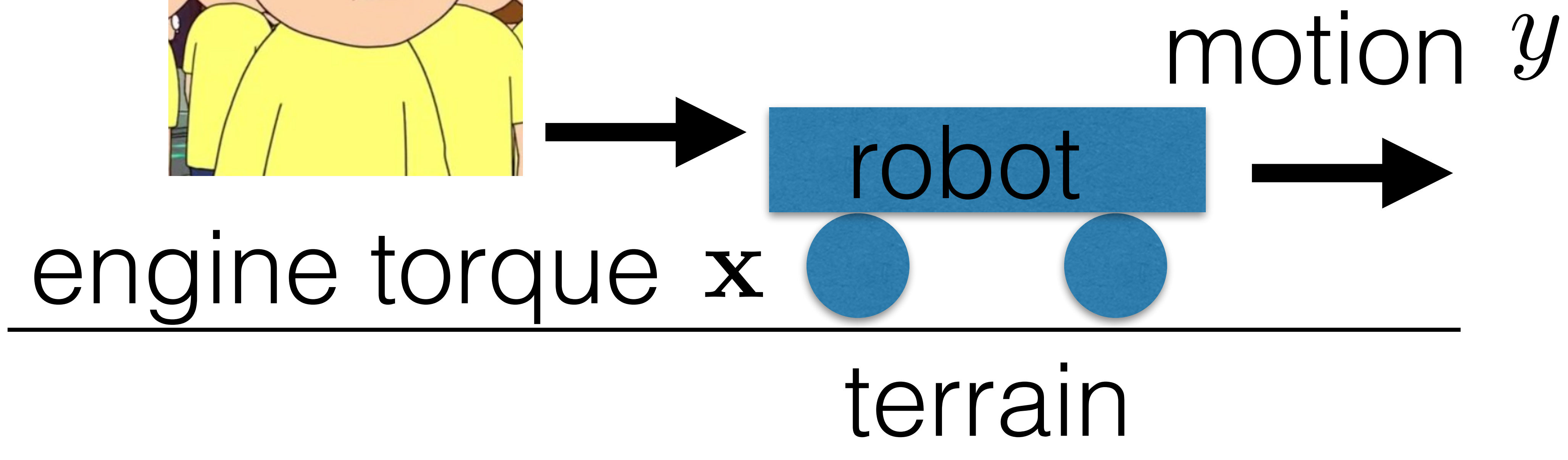
What should I do instead of fitting a curve???

Search for probability distribution of y given x



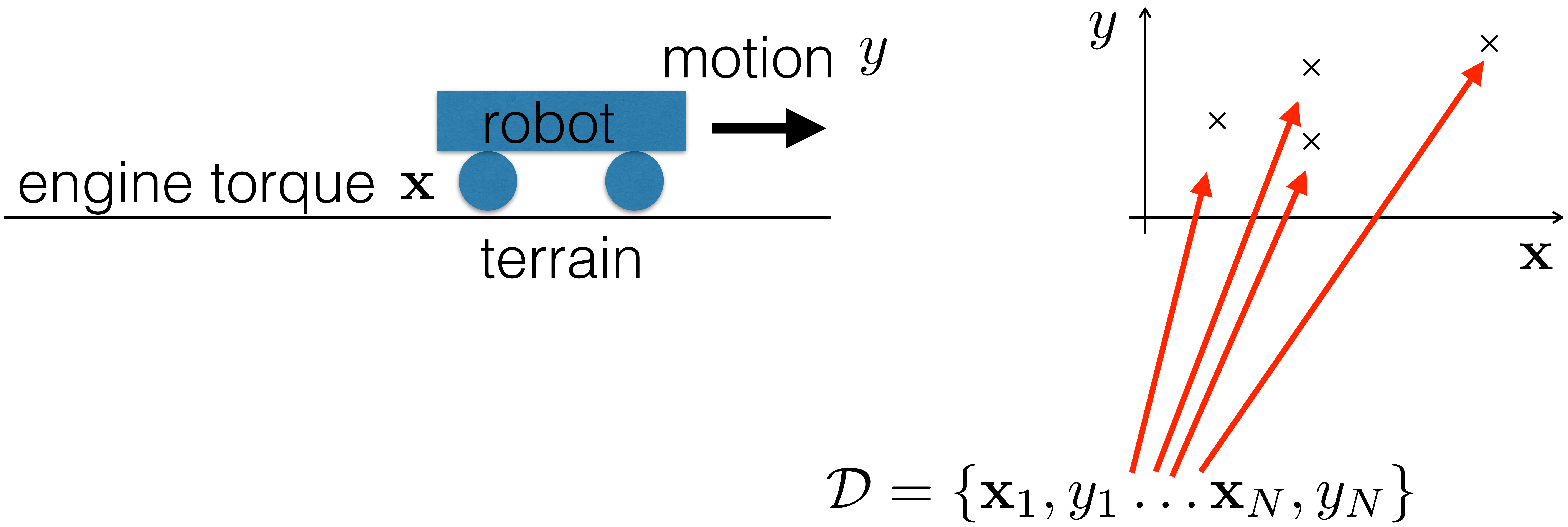




Nature is evil !!!

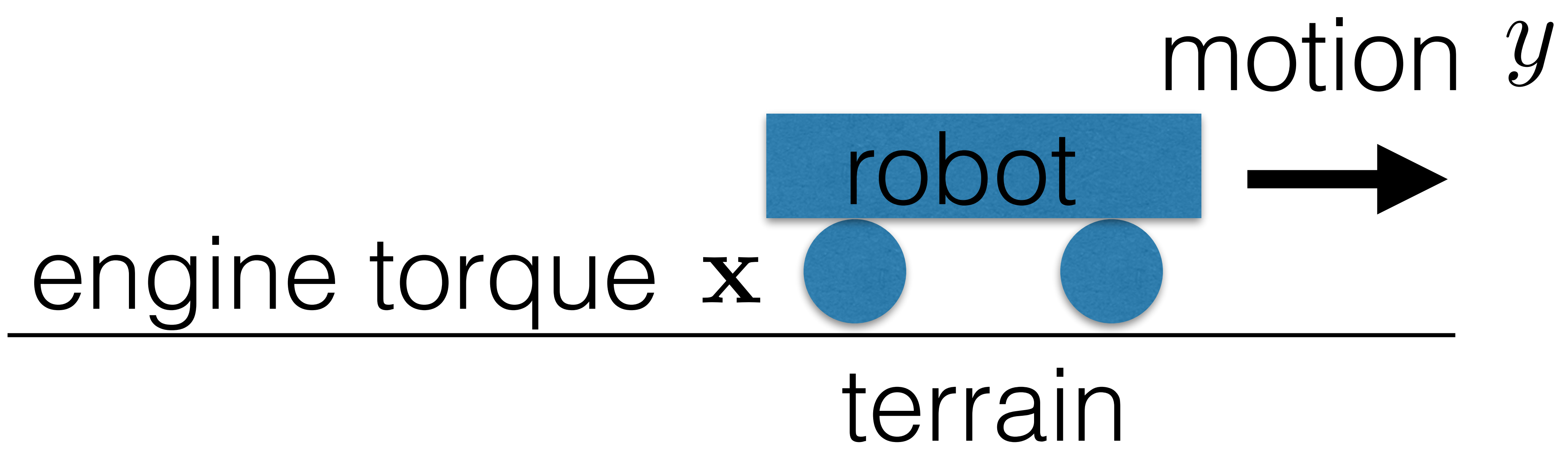


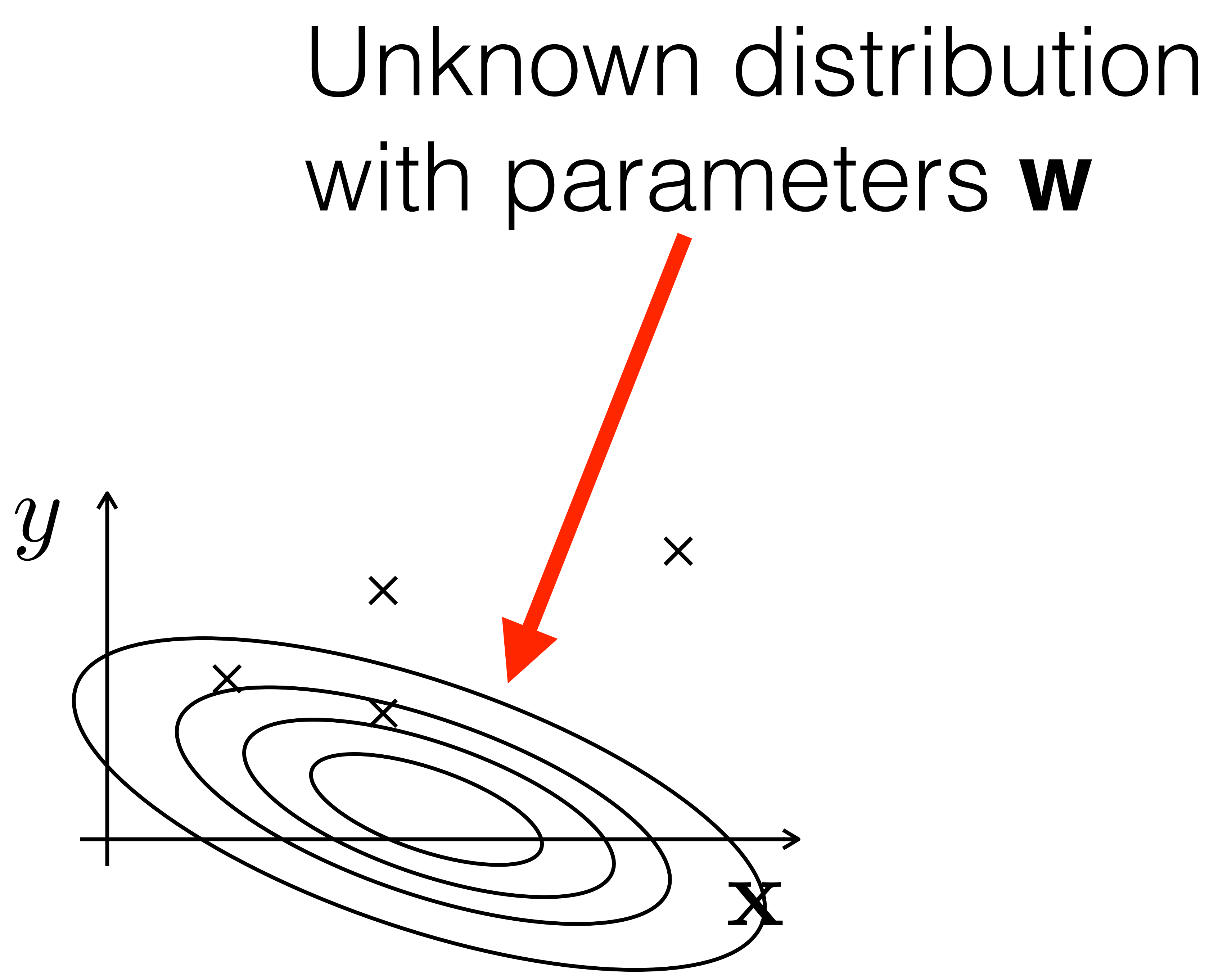

https://rickandmorty.fandom.com/wiki/Evil_Morty

# Motivation example: estimation of a motion model



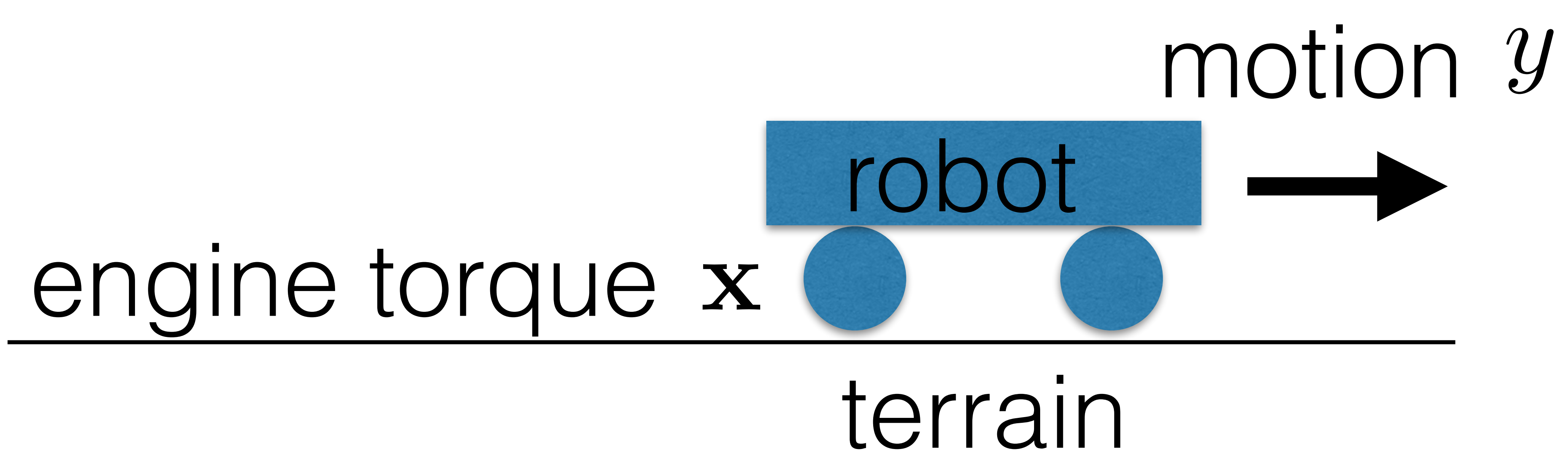
# Motivation example: estimation of a motion model

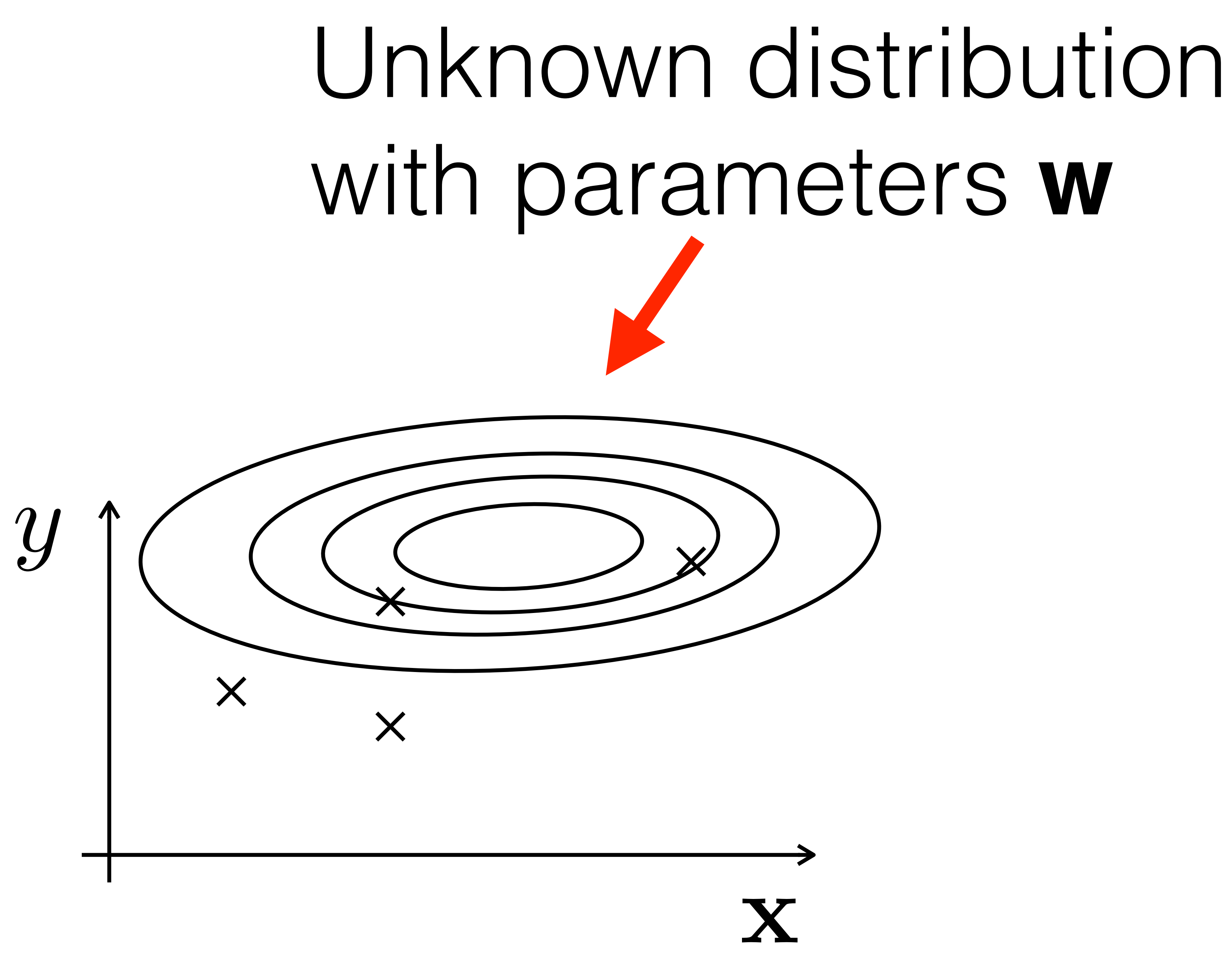



- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

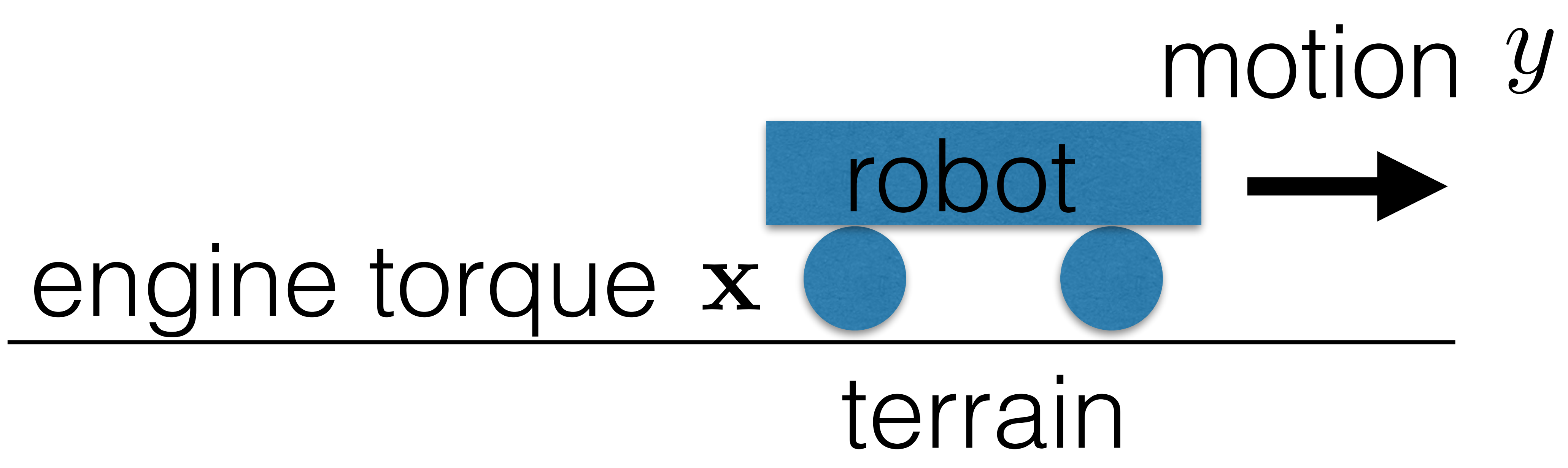
# Motivation example: estimation of a motion model

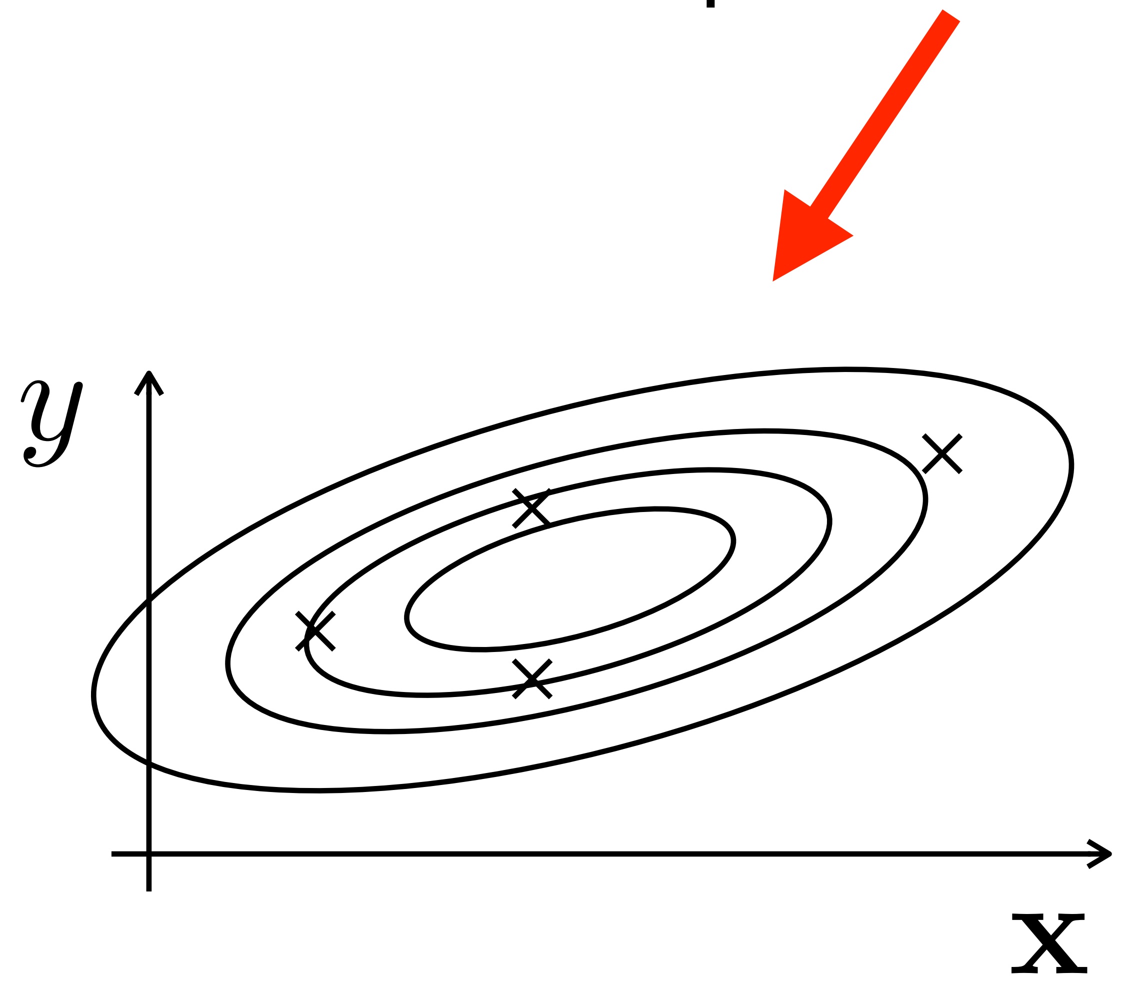



- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

# Motivation example: estimation of a motion model




- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\cancel{p(\mathcal{D})}}$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\cancel{p(\mathcal{D})}}$$

$$= \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

$$= \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg\max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\cancel{p(\mathcal{D})}}$$

$$= \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg\max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w})$$

i.i.d.

$$= \arg\max_{\mathbf{w}} \left( \prod_i p(\mathbf{x}_i, y_i|\mathbf{w}) \right) p(\mathbf{w})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\cancel{p(\mathcal{D})}}$$

$$= \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg\max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w})$$

i.i.d.

$$= \arg\max_{\mathbf{w}} \left(\prod_i p(\mathbf{x}_i, y_i|\mathbf{w})\right) p(\mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w})p(\mathbf{x}_i)\right) p(\mathbf{w})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\cancel{p(\mathcal{D})}}$$

$$= \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg\max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w})$$

i.i.d.

$$= \arg\max_{\mathbf{w}} \left(\prod_i p(\mathbf{x}_i, y_i|\mathbf{w})\right) p(\mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w})p(\mathbf{x}_i)\right) p(\mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \left(\sum_i \log(p(y_i|\mathbf{x}_i, \mathbf{w})) \cancel{+ \log p(\mathbf{x}_i)}\right) + \log p(\mathbf{w})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

$$= \arg\max_{\mathbf{w}} \left( \sum_i \log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + \log p(\mathbf{w})$$

- We search for parameters $\mathbf{w}$ of unknown distribution given measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

$$= \arg\max_{\mathbf{w}} \left( \sum_i \log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + \log p(\mathbf{w})$$

$$= \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

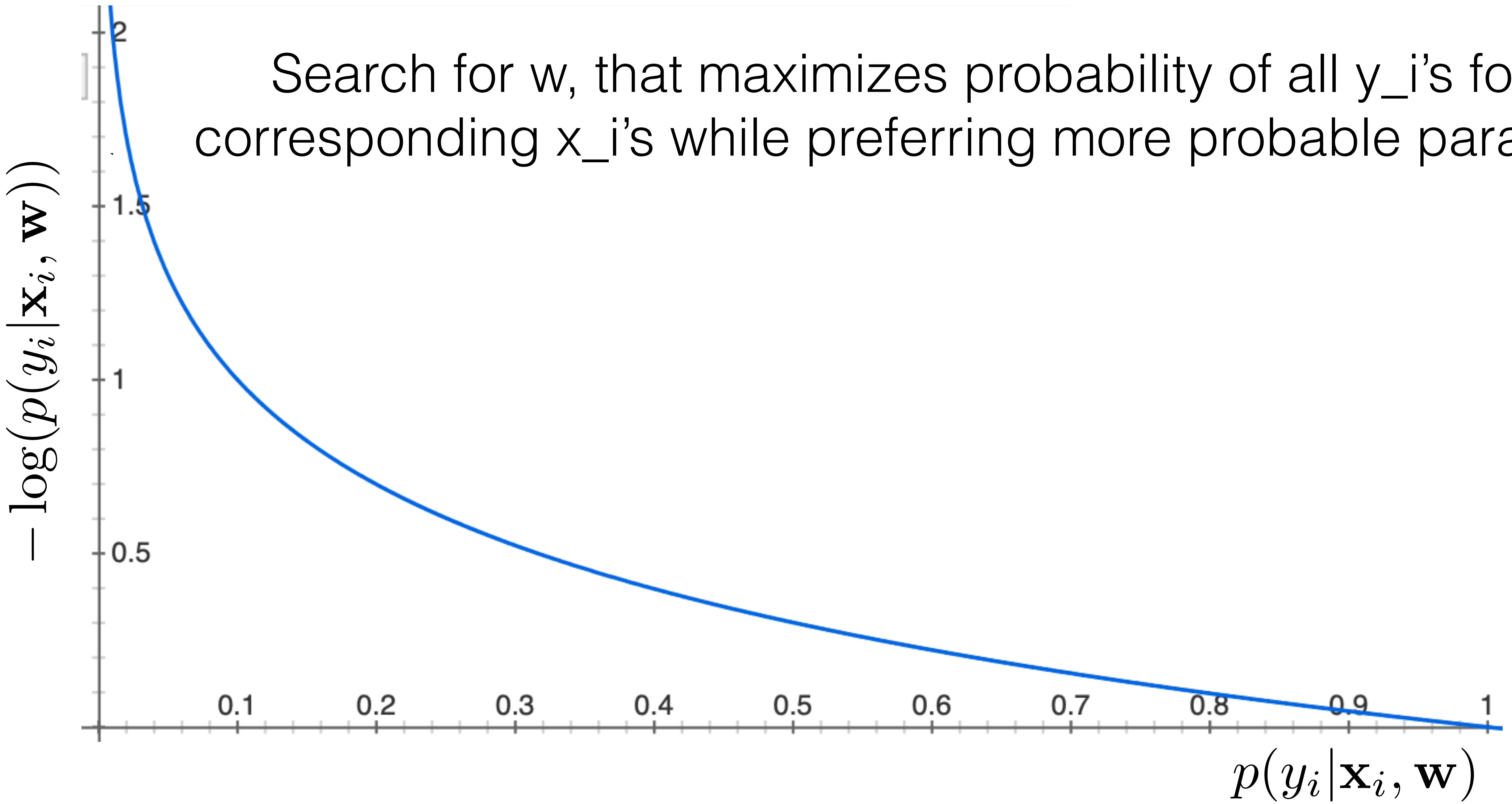
log likelihood    prior/regulariser

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) \;\; = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

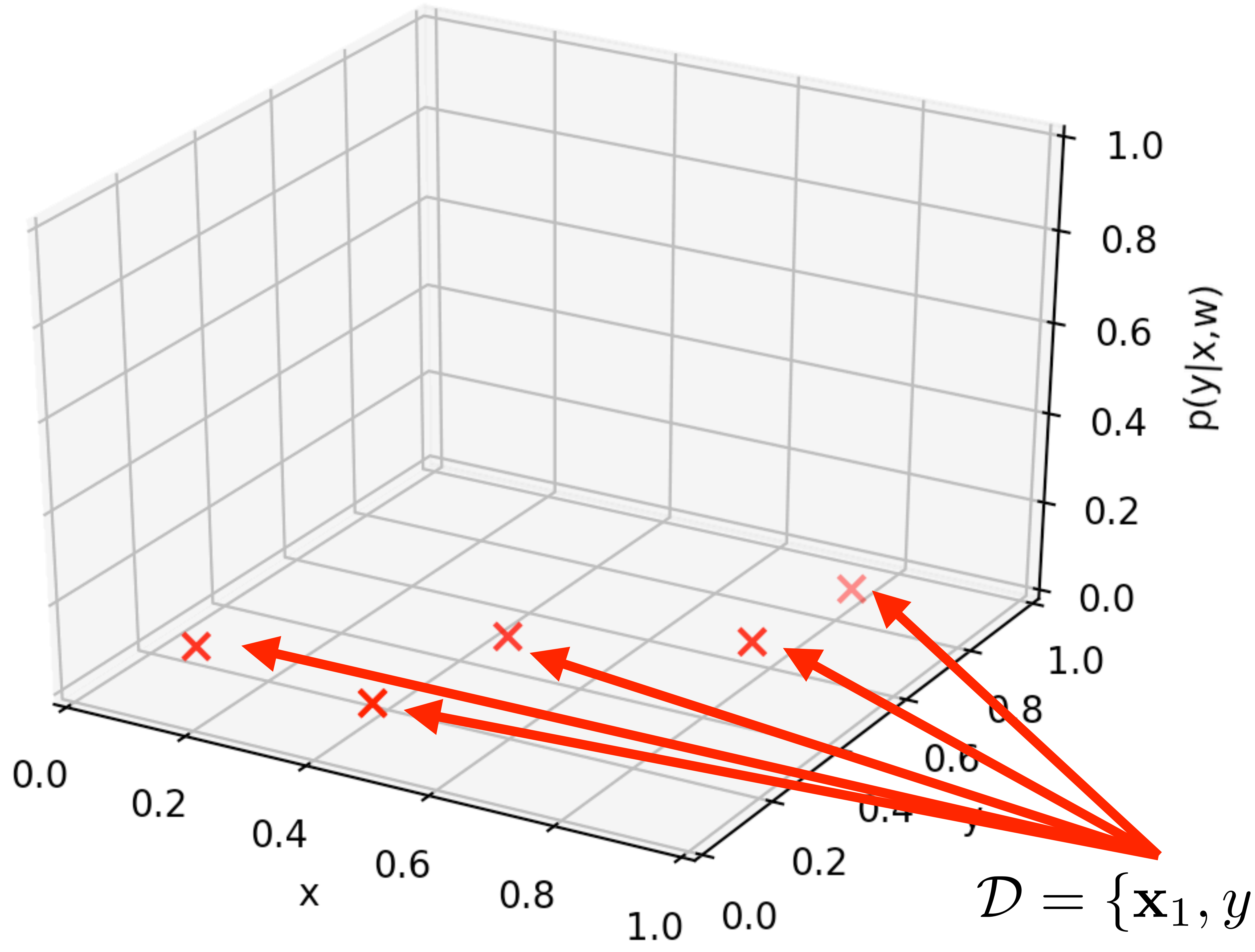
log likelihood

prior/regulariser

Search for w, that maximizes probability of all y_i's for given corresponding x_i's while preferring more probable parameters w.

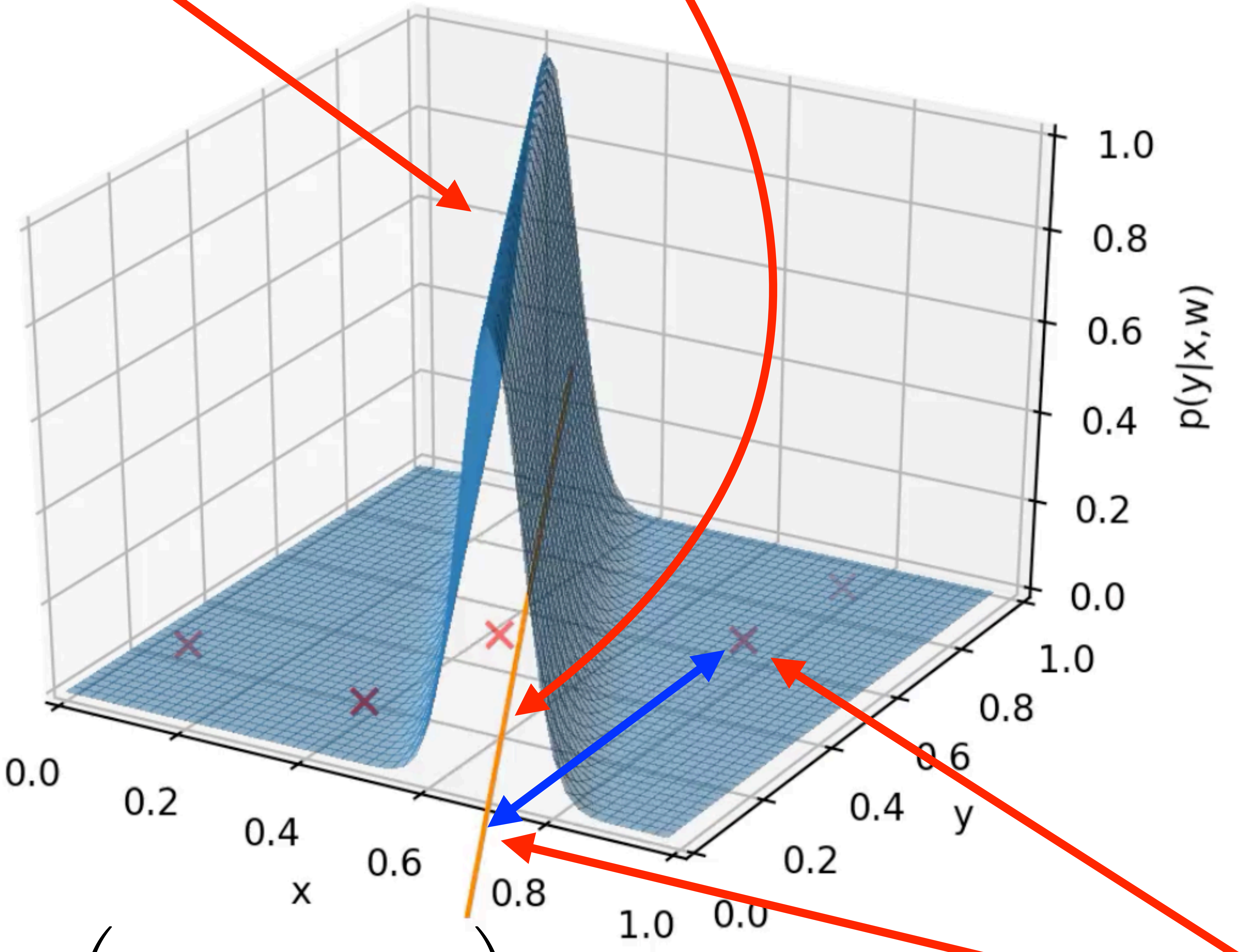


$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(w_1 x + w_0,\ \sigma^2)$$



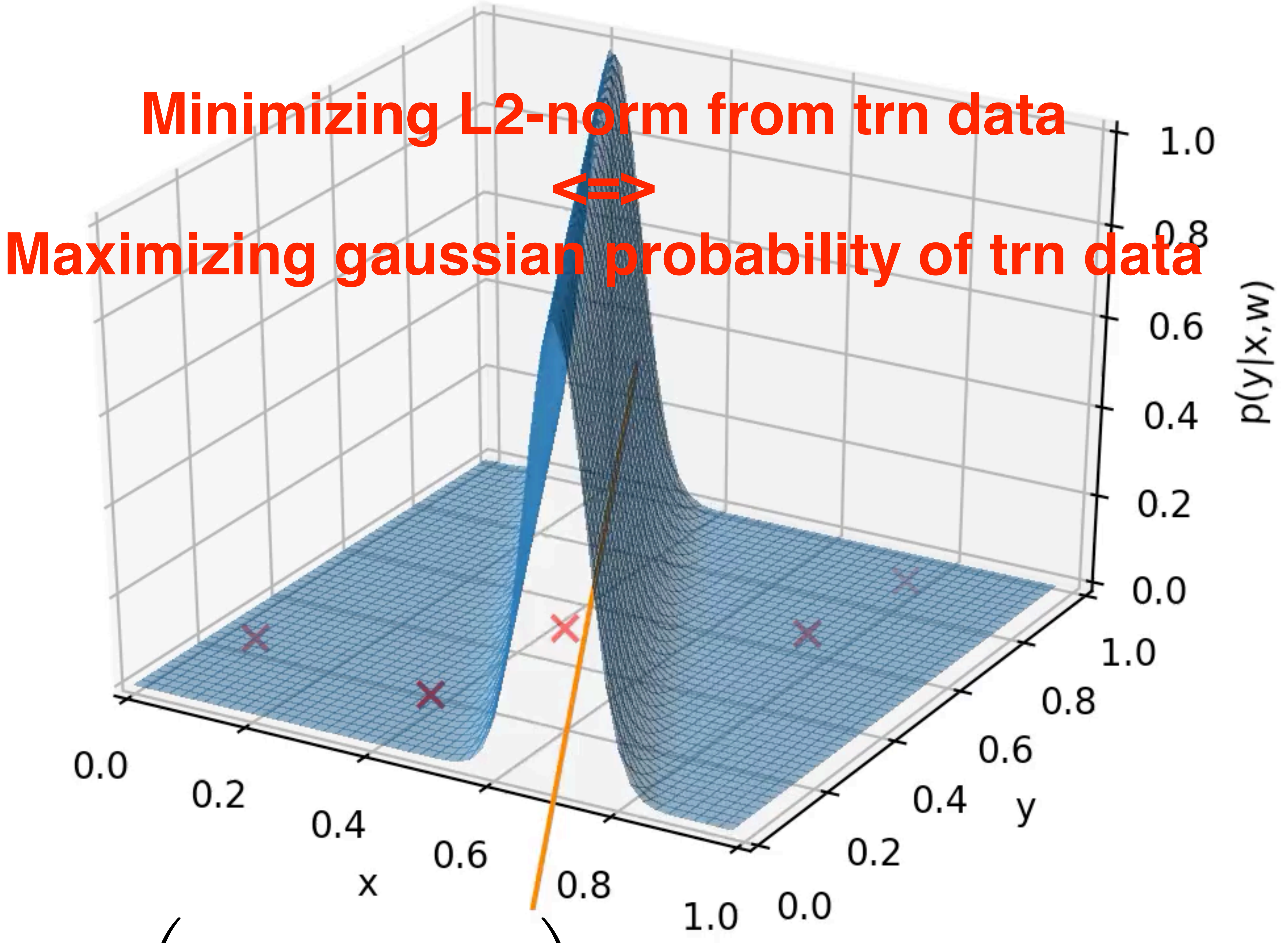

$$\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$$

$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(w_1 x + w_0, \ \sigma^2)$$




$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$$

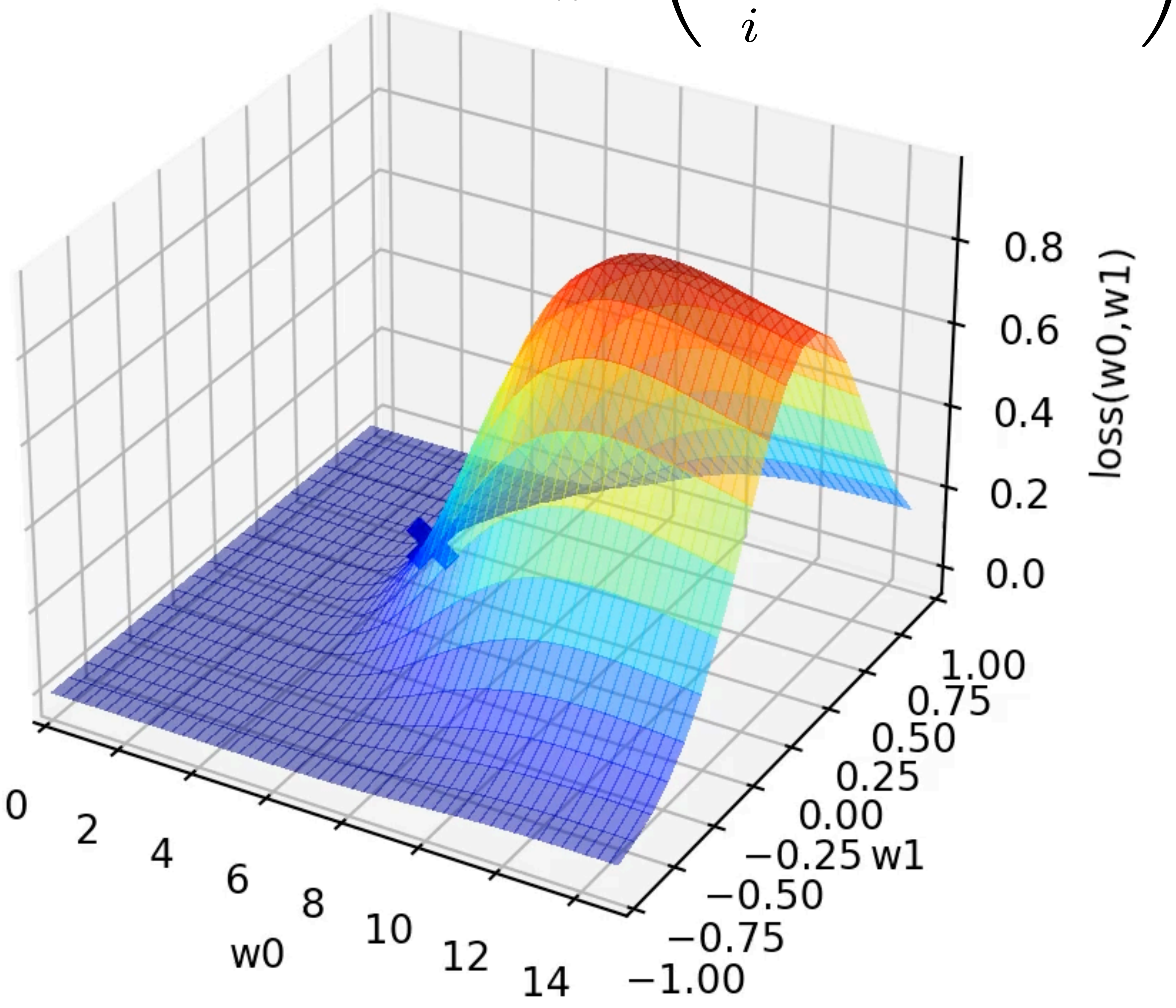
$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(w_1 x + w_0,\ \sigma^2)$$



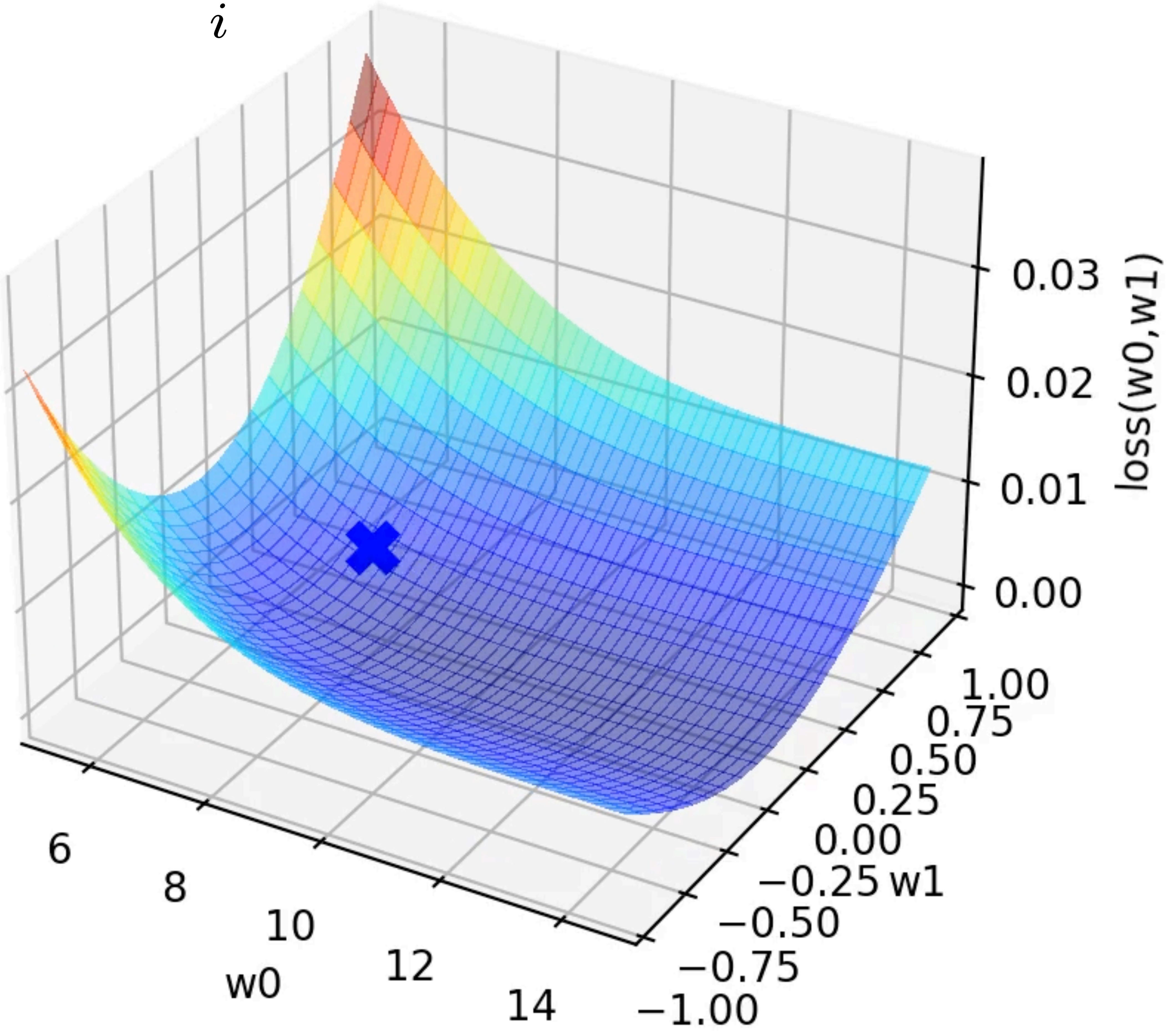

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$$

- In what sense is the MLE and the LSQ formulations equivalent?

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) \quad = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

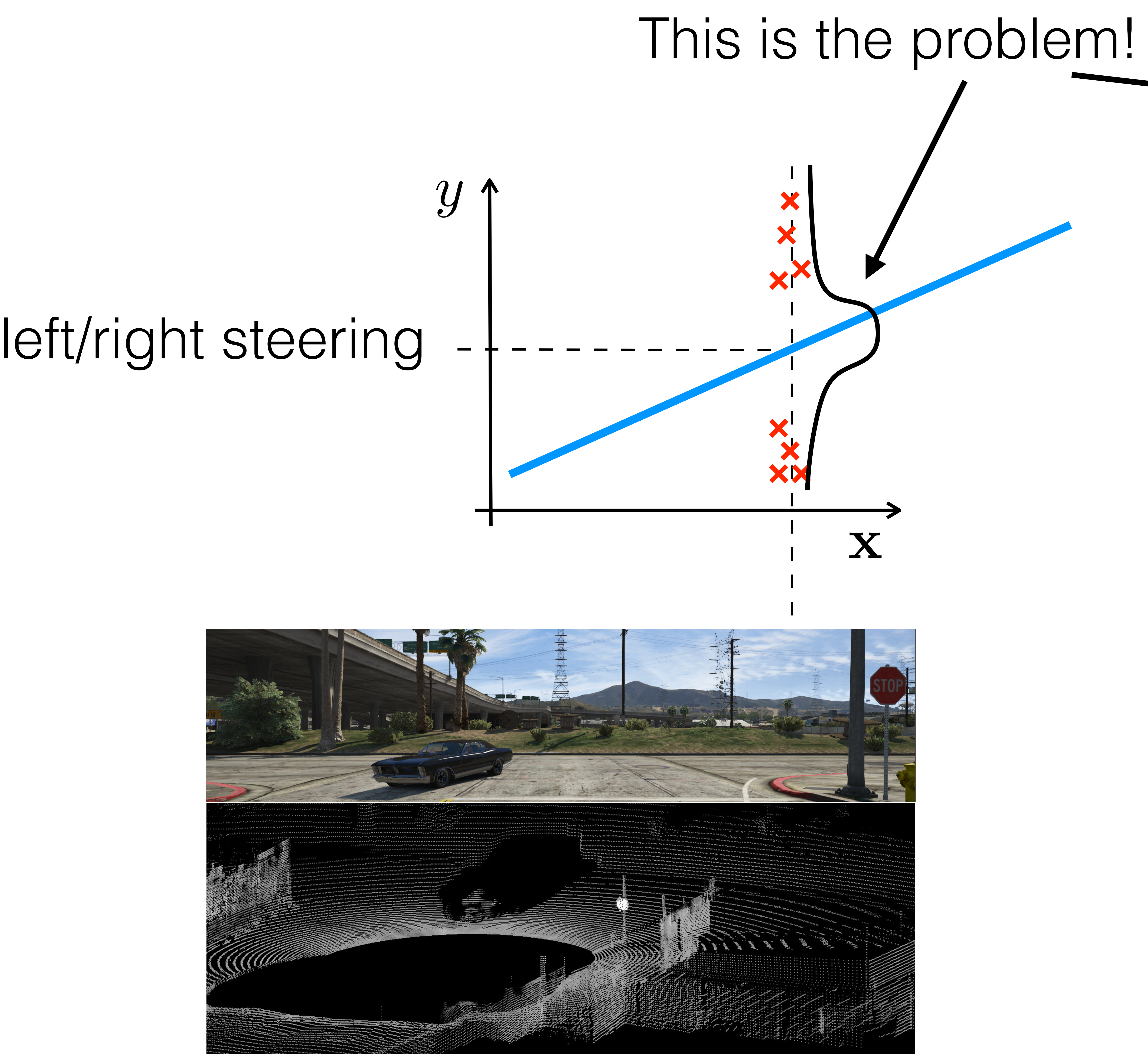


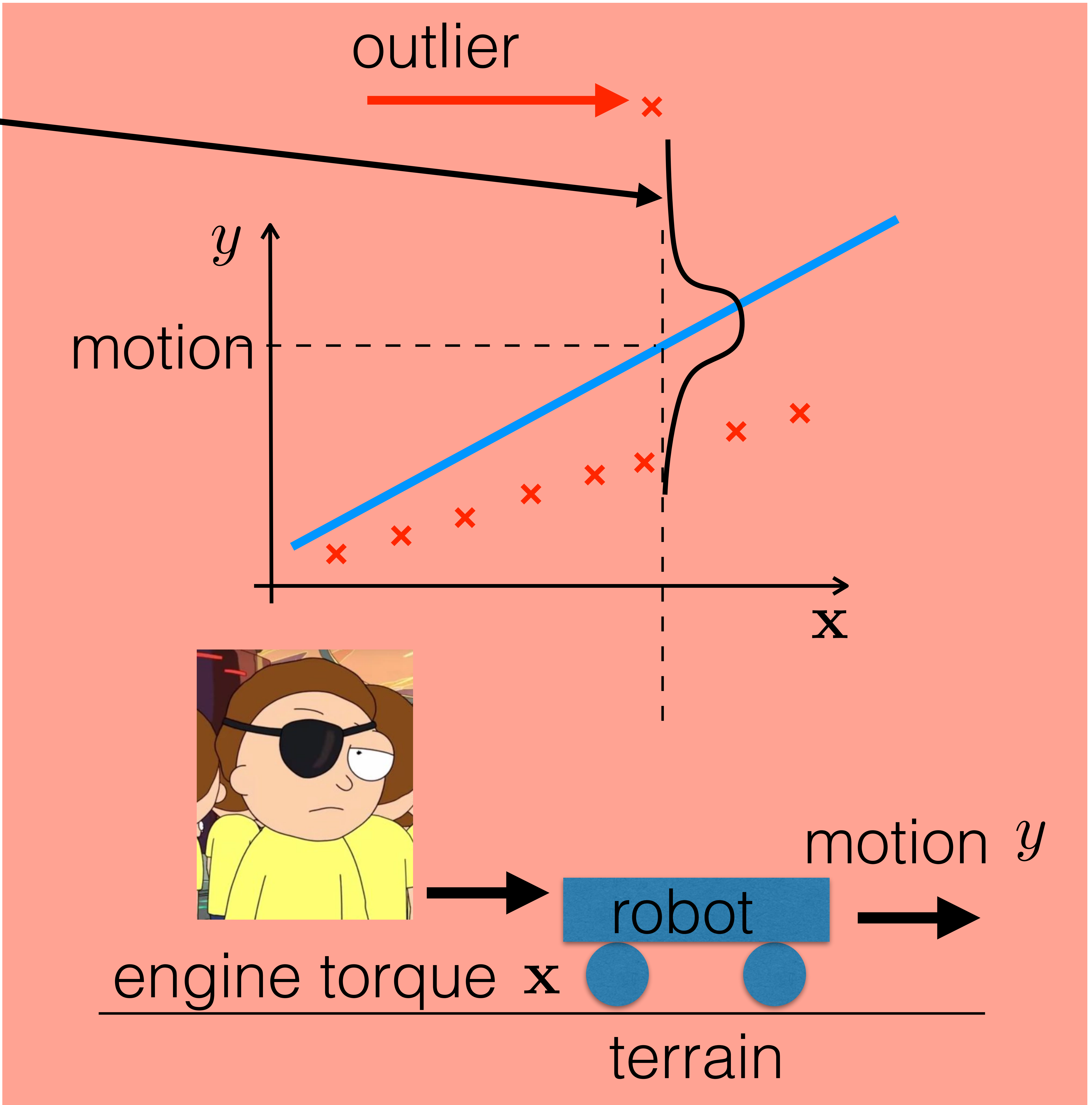

MLE




LSQ

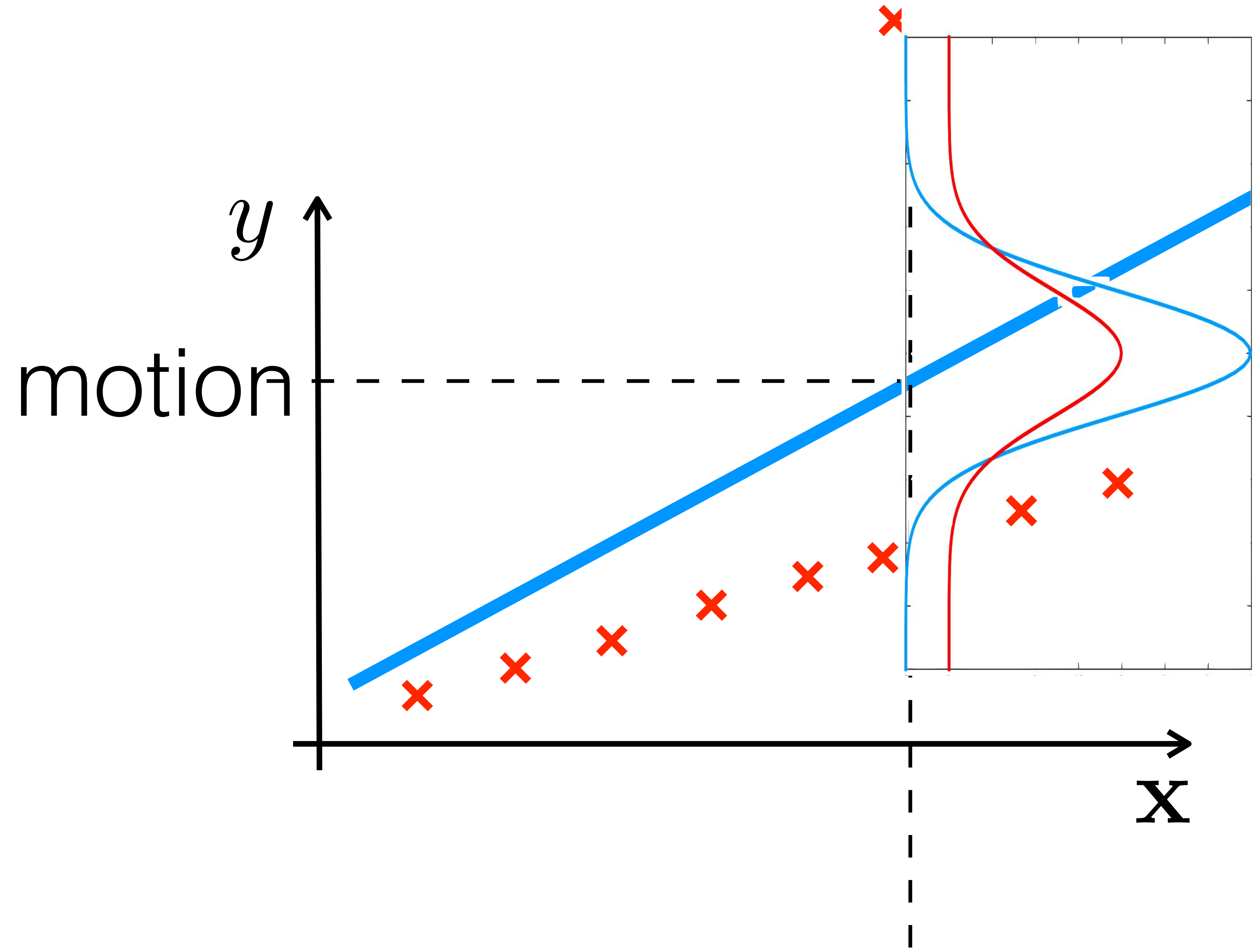
# What can go wrong: **inappropriate choice of loss function**



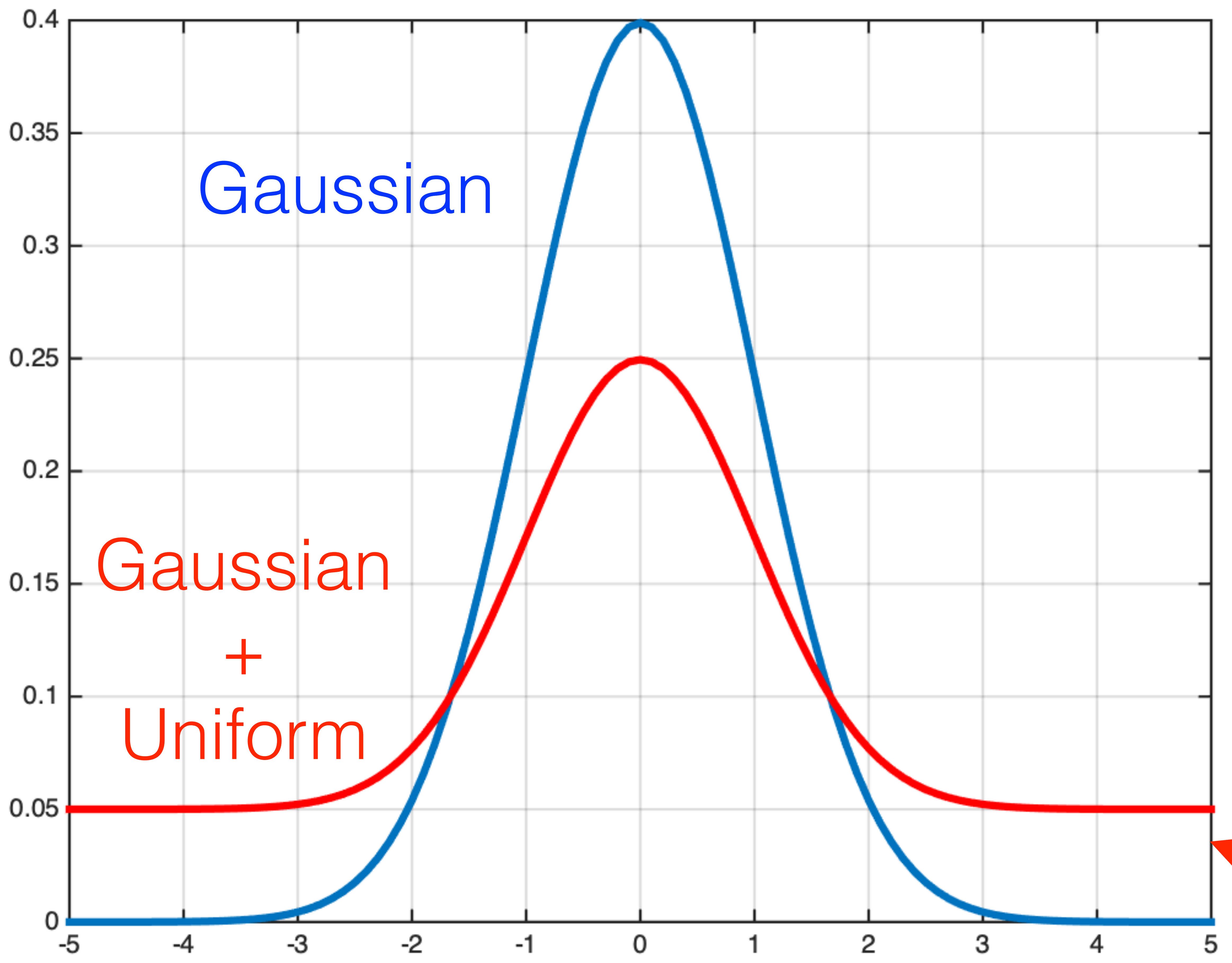



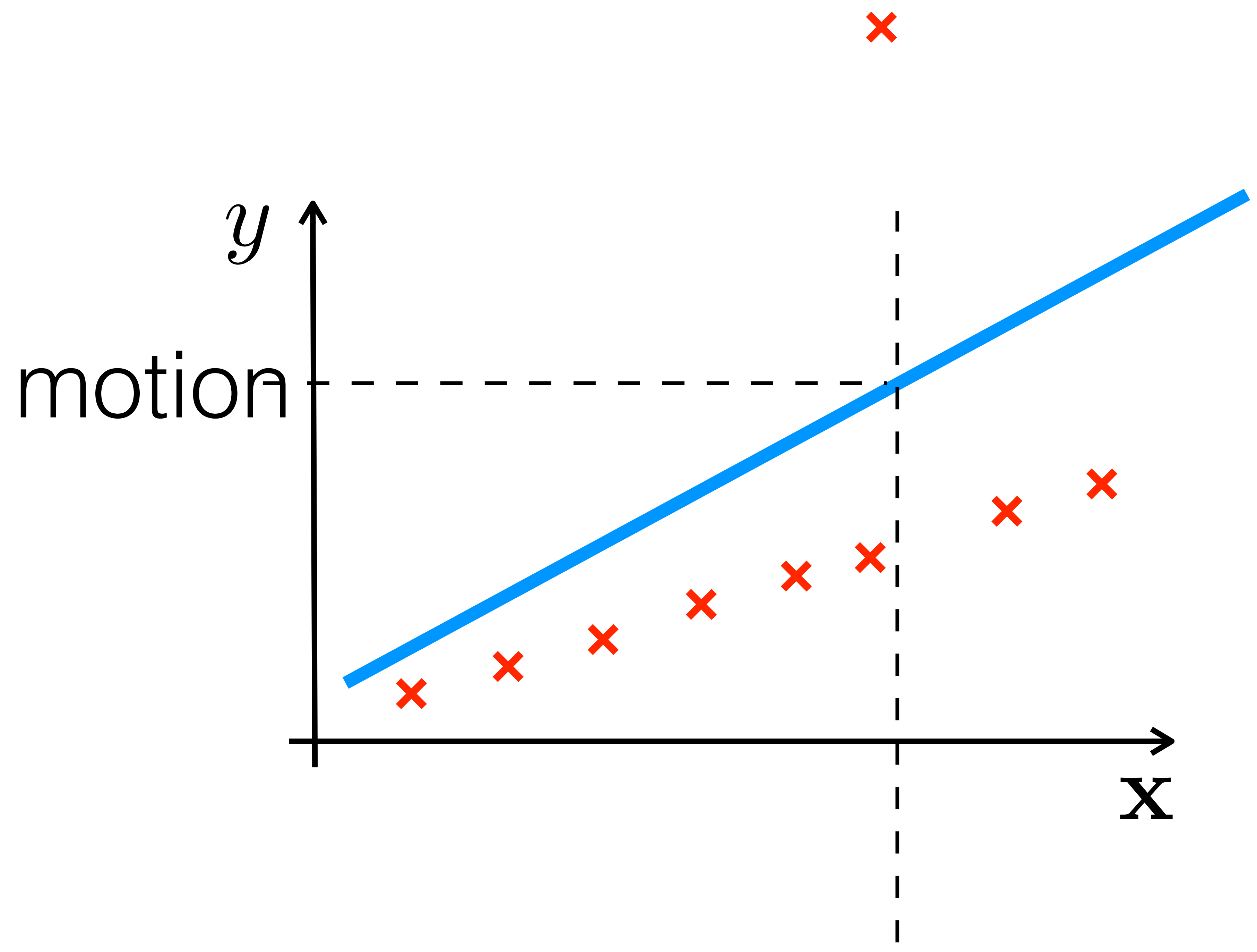

https://rickandmorty.fandom.com/wiki/Evil_Morty

# Robust regression



# Robust regression



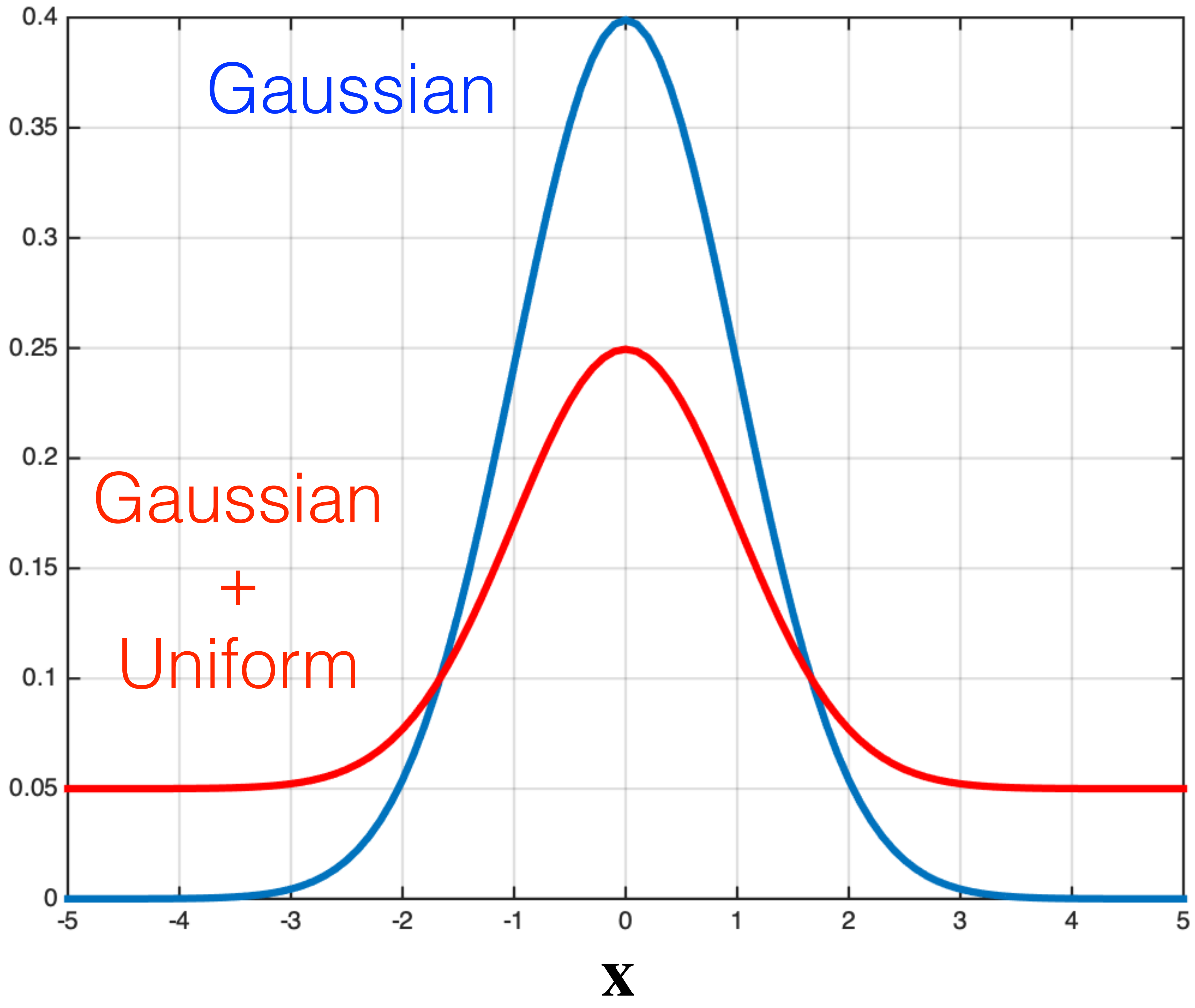




Evil source of uniform noise

# Robust regression

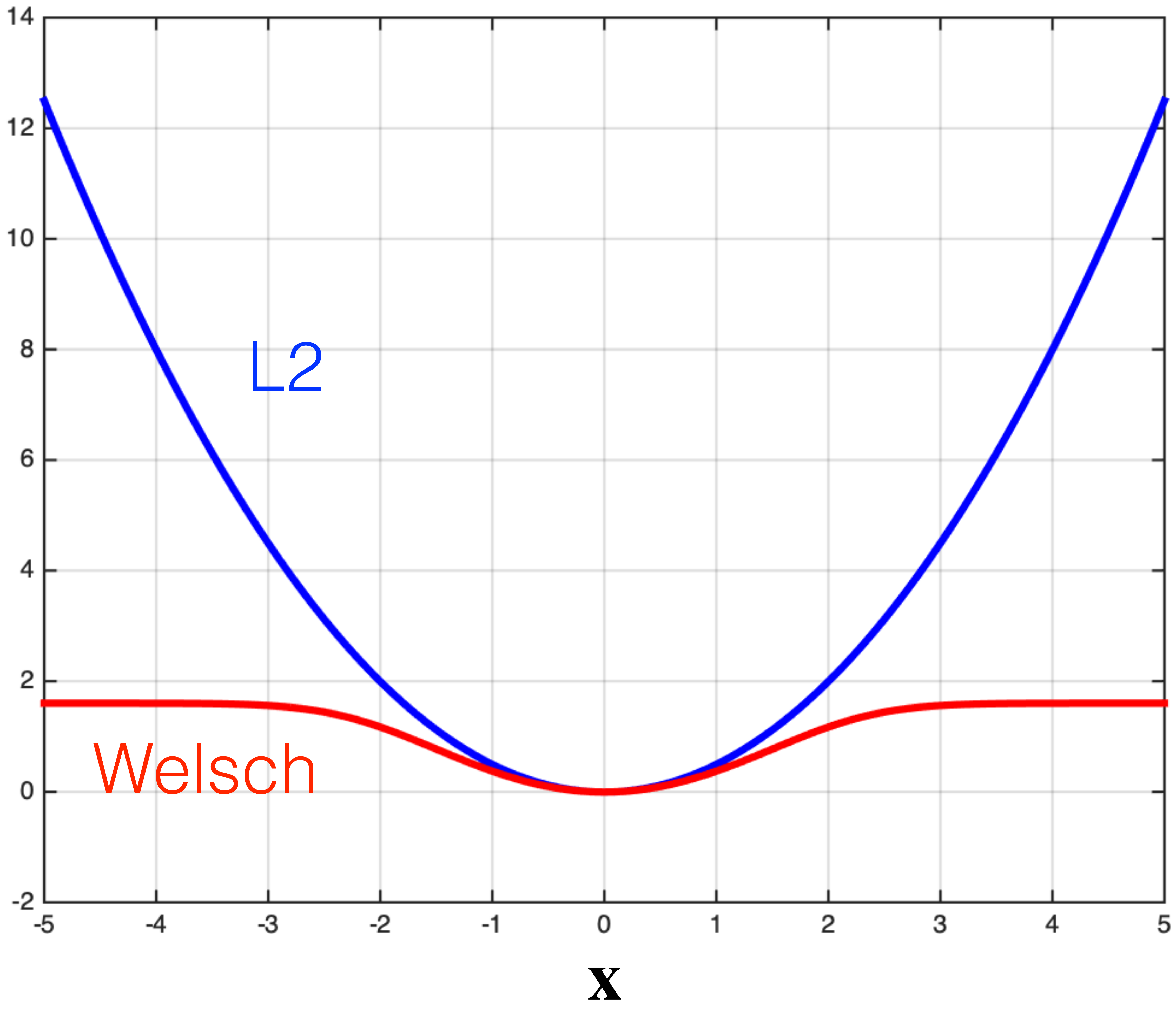
$p(y \mid \mathbf{x}, \mathbf{w})$

Probability distributions



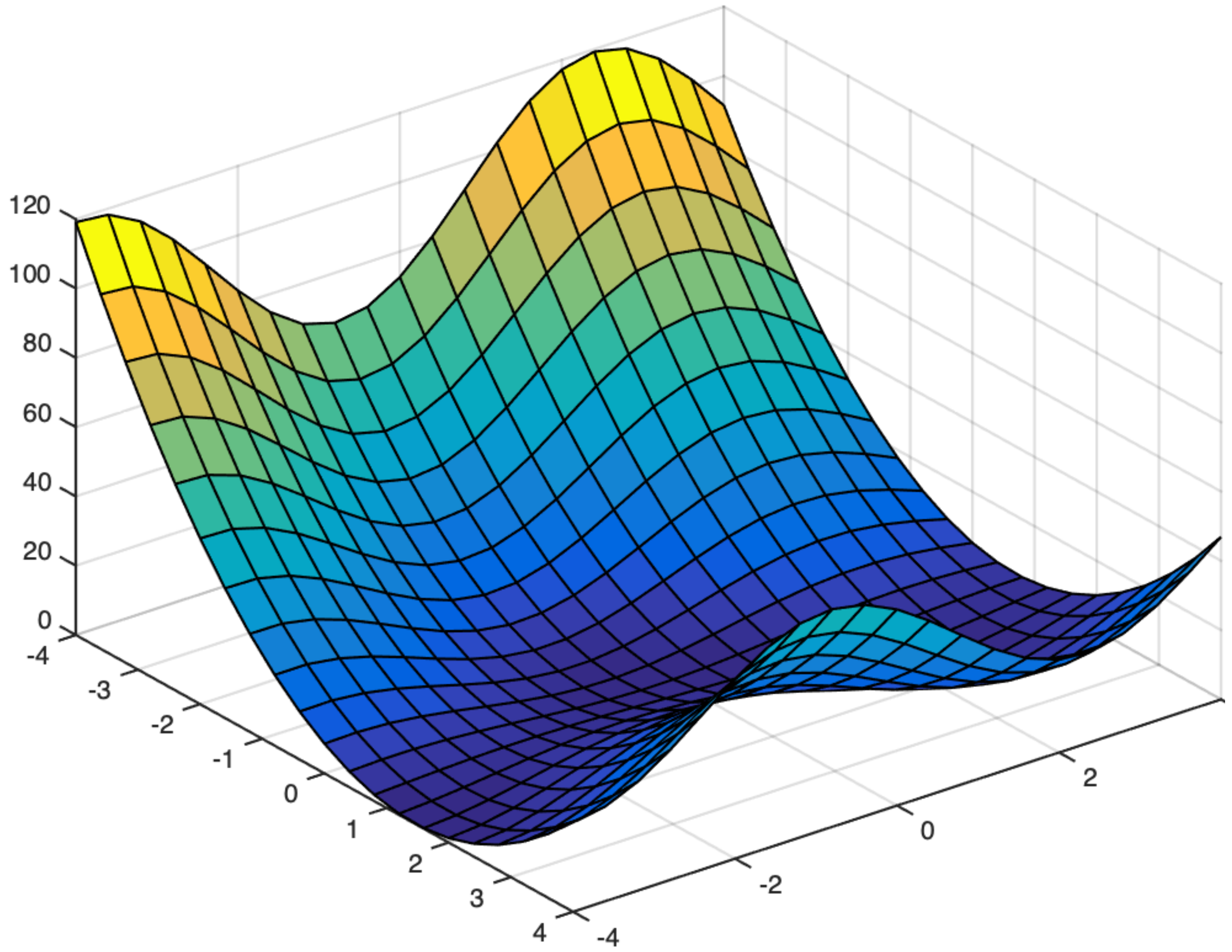

$\mathscr{L}(\mathbf{w}) = -\log(p(y \mid \mathbf{x}, \mathbf{w}))$

Corresponding losses



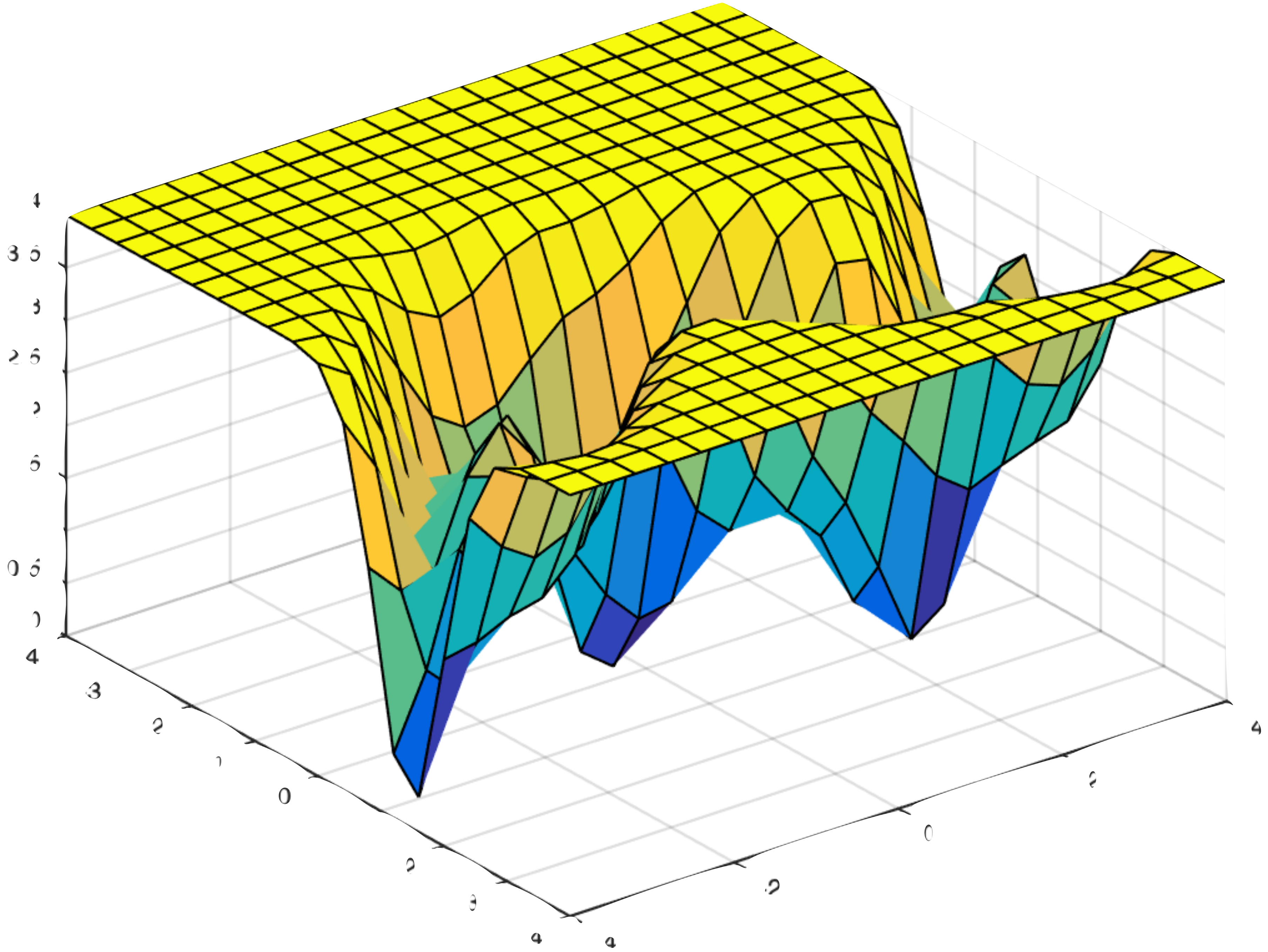
# Robust regression

## L2 landscape




- **Uniform noise omitted**
  => GD-friendly landscape
- Gradient size encodes distance
- Easy to optimize

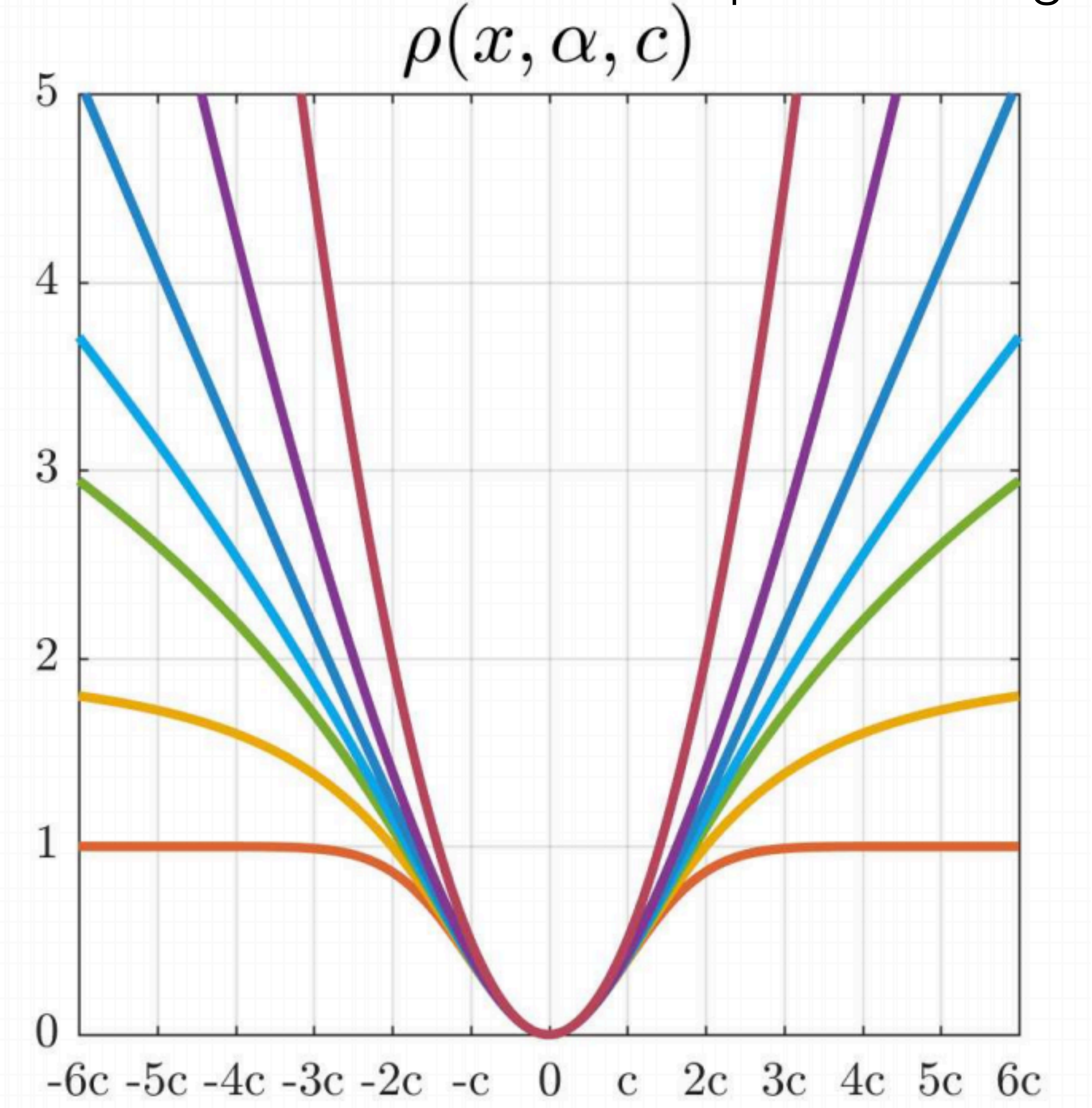
## Welsch landscape



- **Uniform noise modelled**
  => unfriendly landscape
- Non-convex: Large narrow plateaus with zero gradient
- Good initialization required

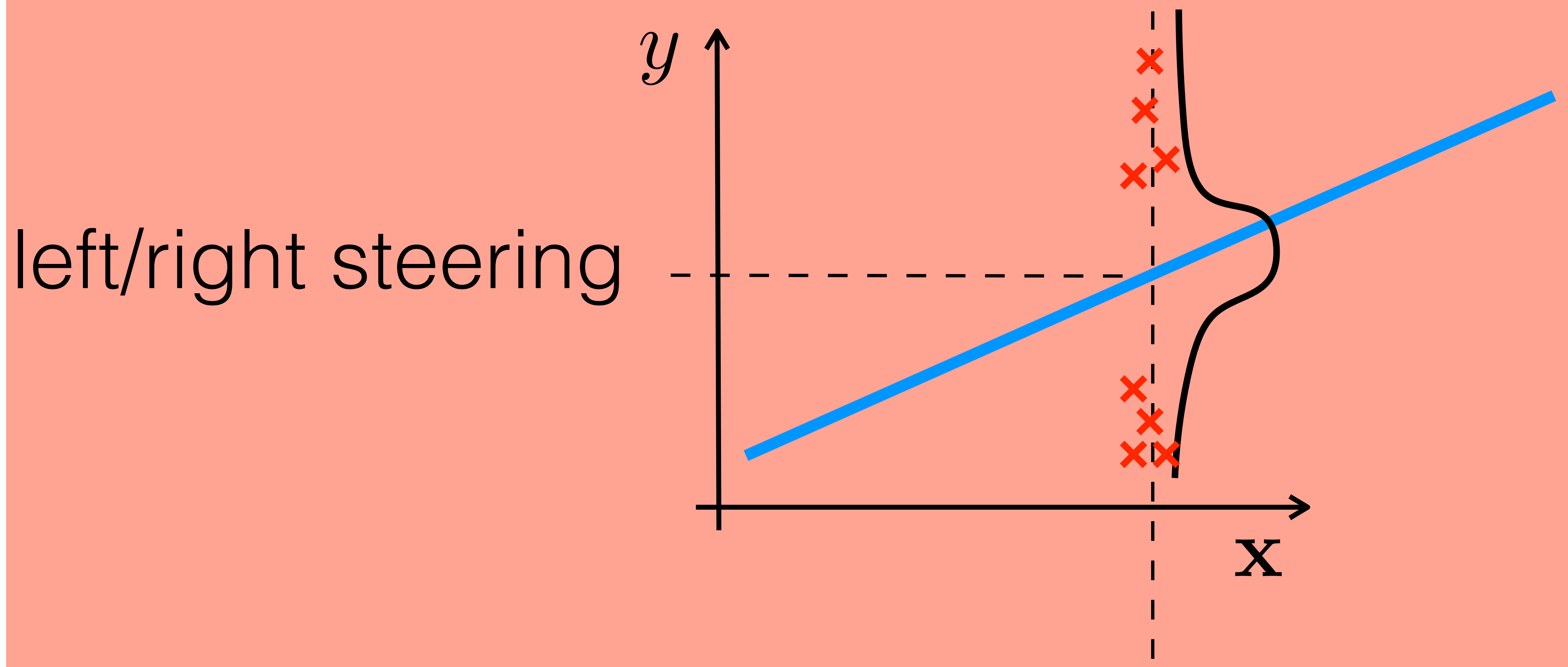
# Shape of robust regression functions [Barron CVPR 2019]

https://arxiv.org/abs/1701.03077

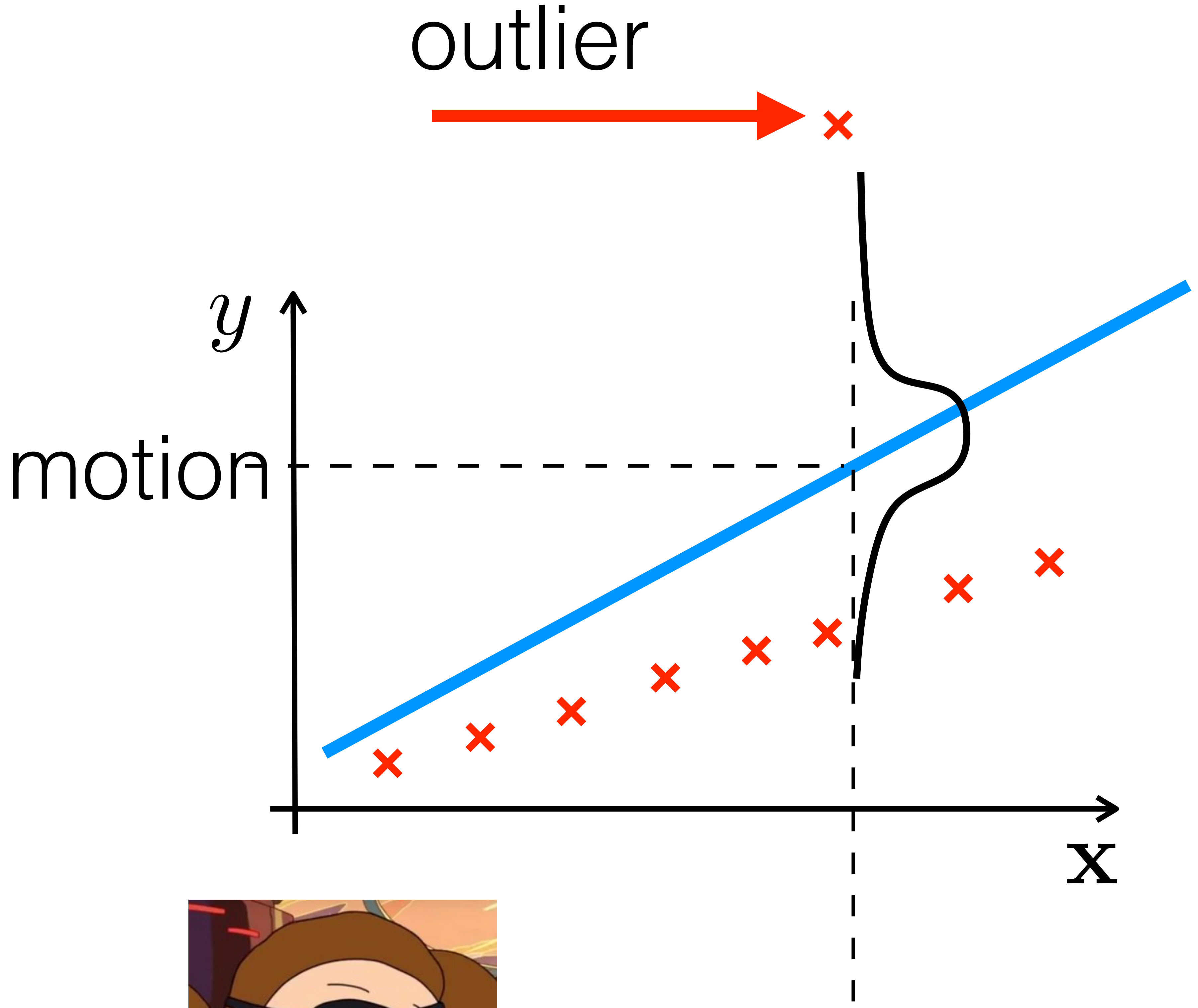


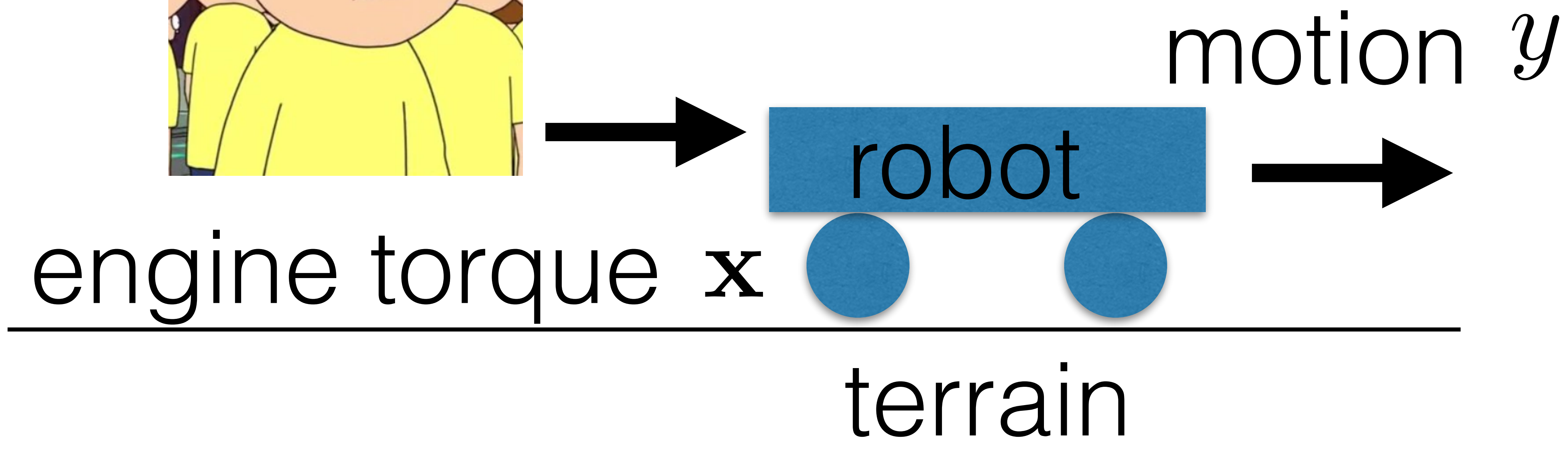

$$\rho(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left( \left( \frac{\left(x/c\right)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right)$$

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



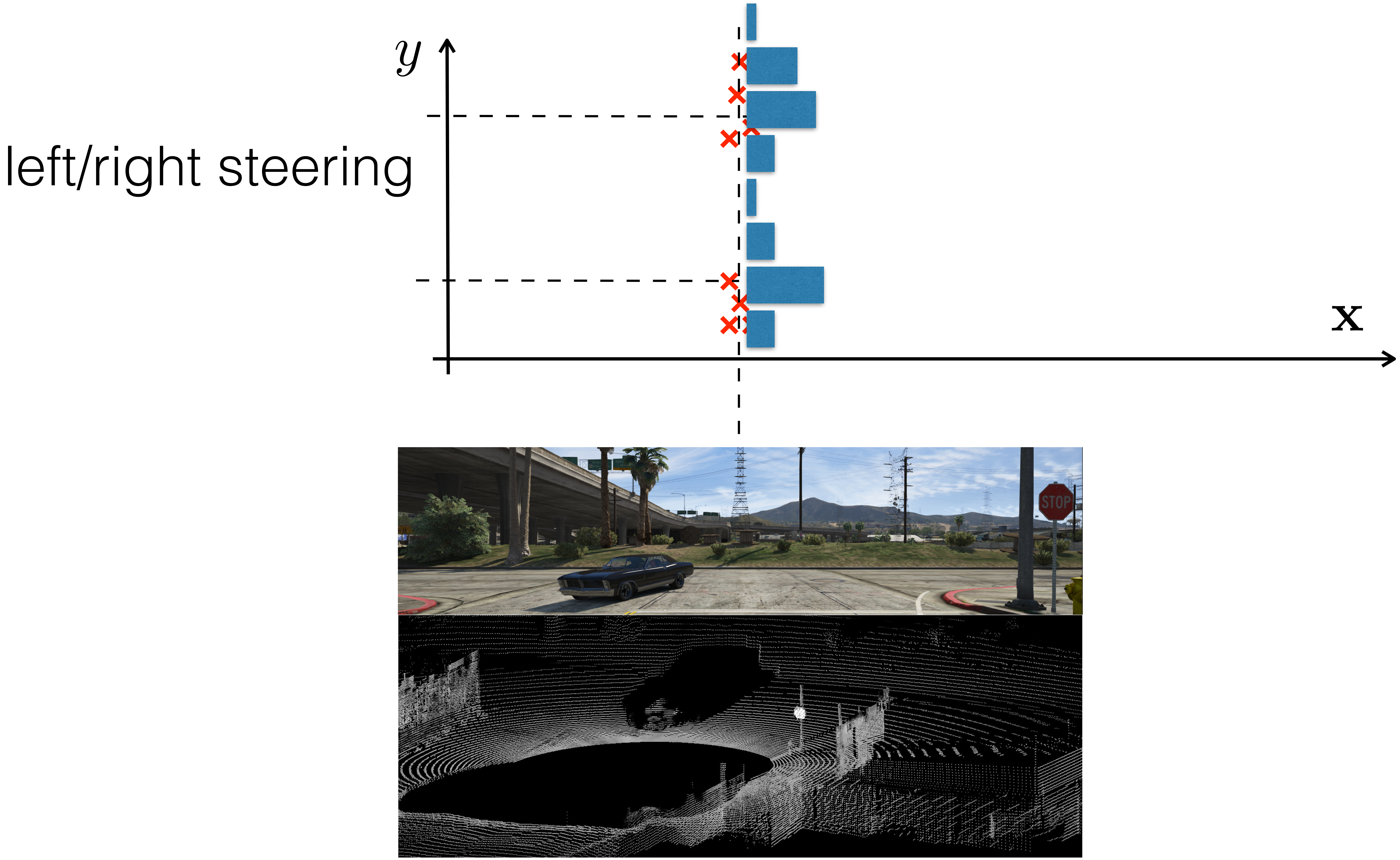




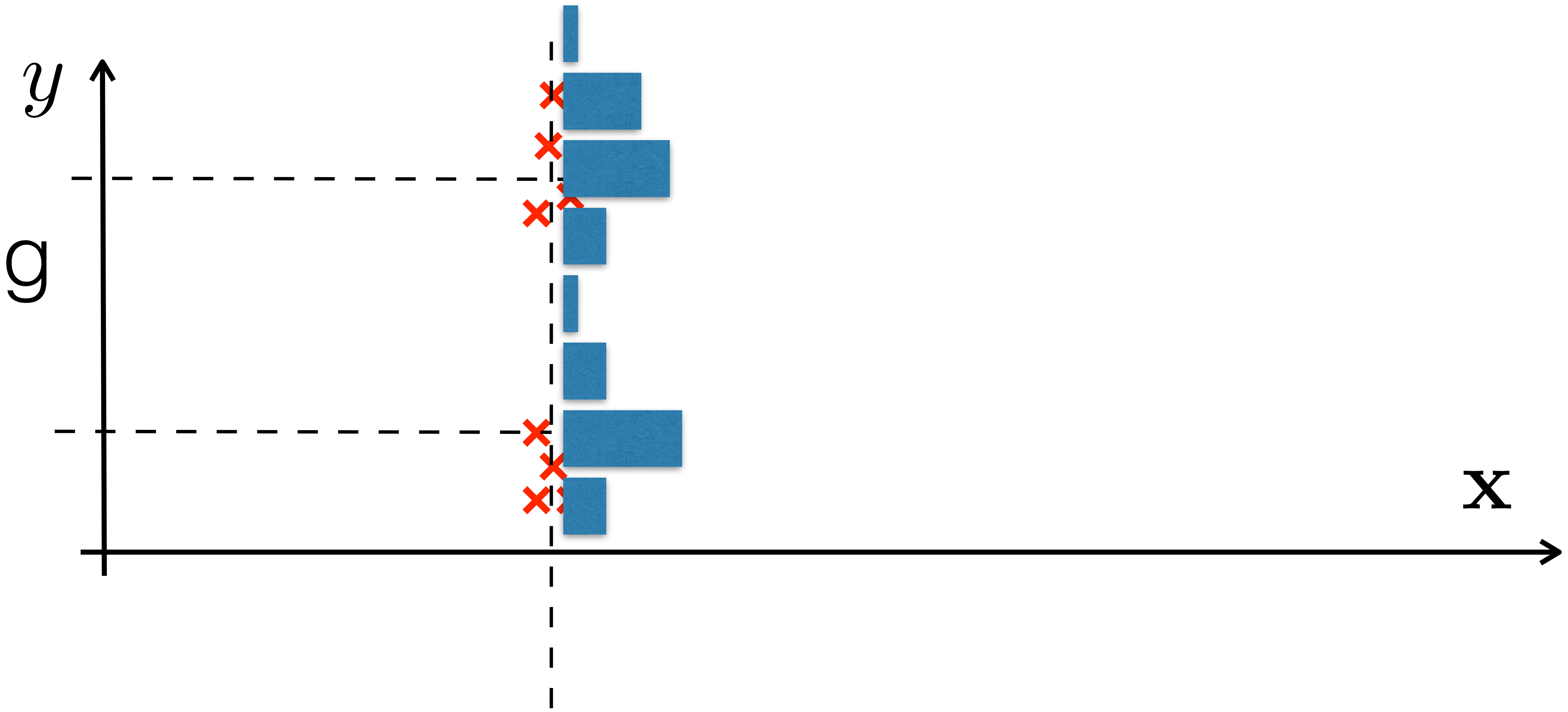

https://rickandmorty.fandom.com/wiki/Evil_Morty

# Work-around 1: discretize y-domain and treat the problem as classifcation

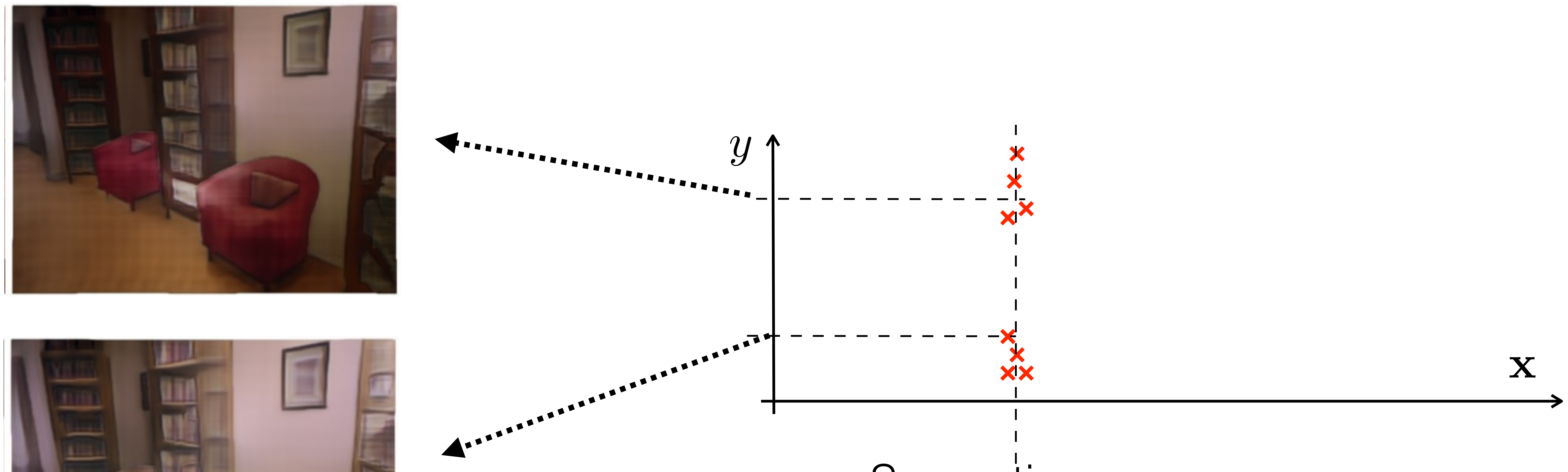


# Work-around 1: discretize y-domain and treat the problem as classifcation

left/right steering



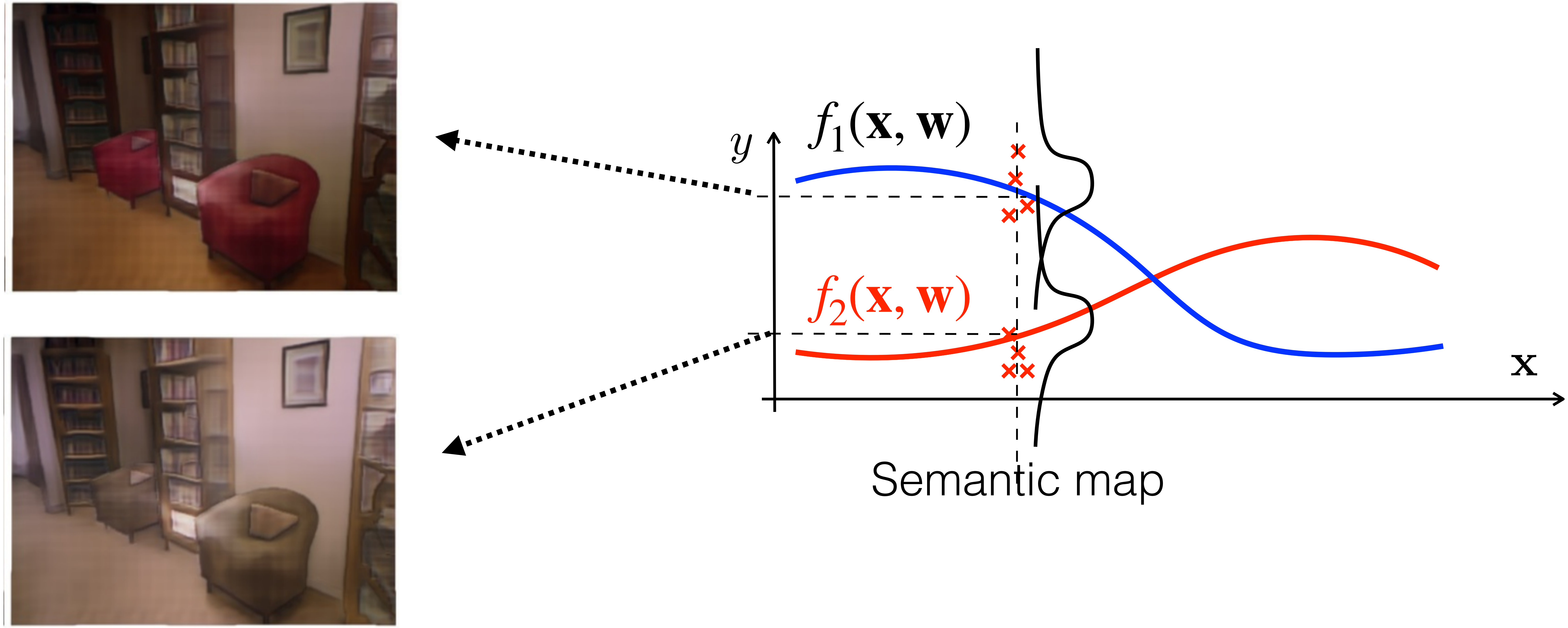

Works reasonably for low-dim y




Semantic map

What if y are images?

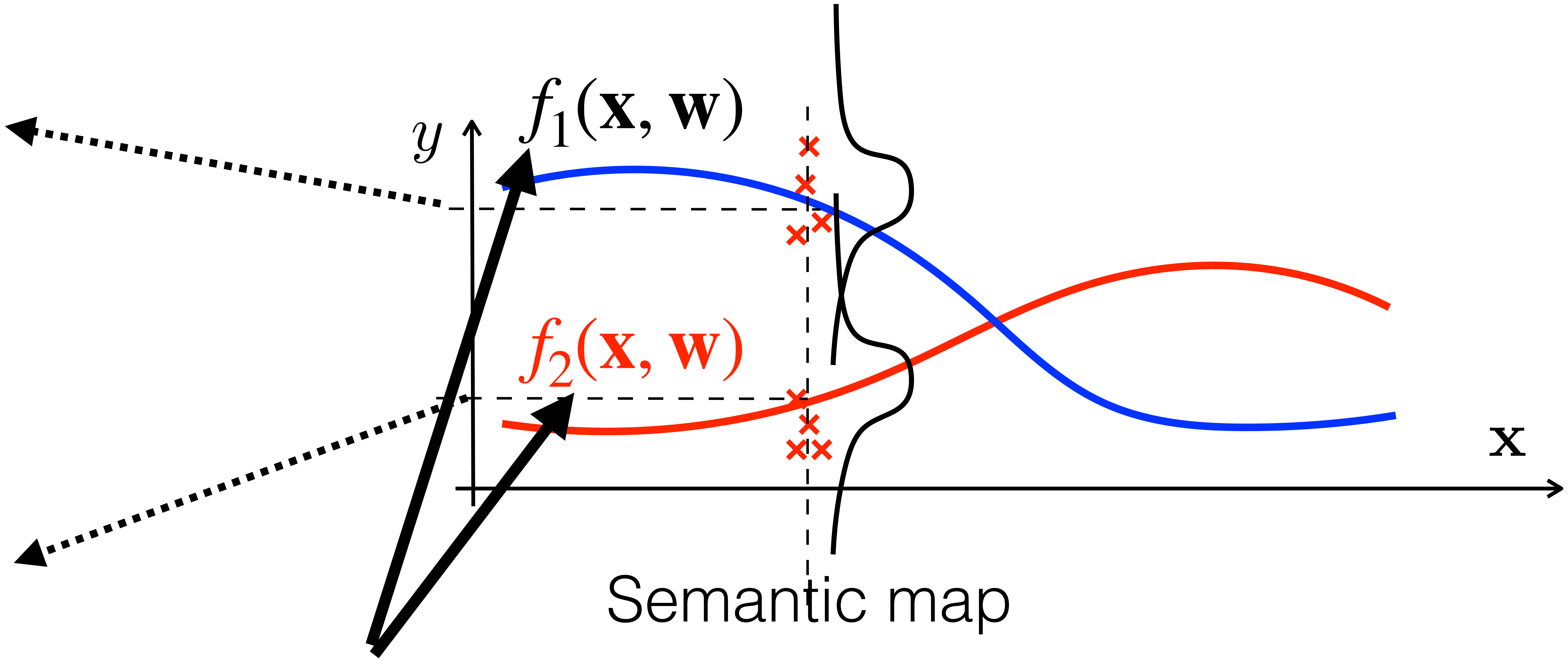
# Work-around 2: allow multiple hypothesis




Multiple choice loss
[Microsoft, NIPS, 2012], [Koltun, ICCV, 2017]

$$\mathscr{L}(\mathbf{w}) = \min_i \|f_i(\mathbf{x}, \mathbf{w}) - y\|$$

# Work-around 2: allow multiple hypothesis




Problem 1: number of hypothesis may grow exponentially

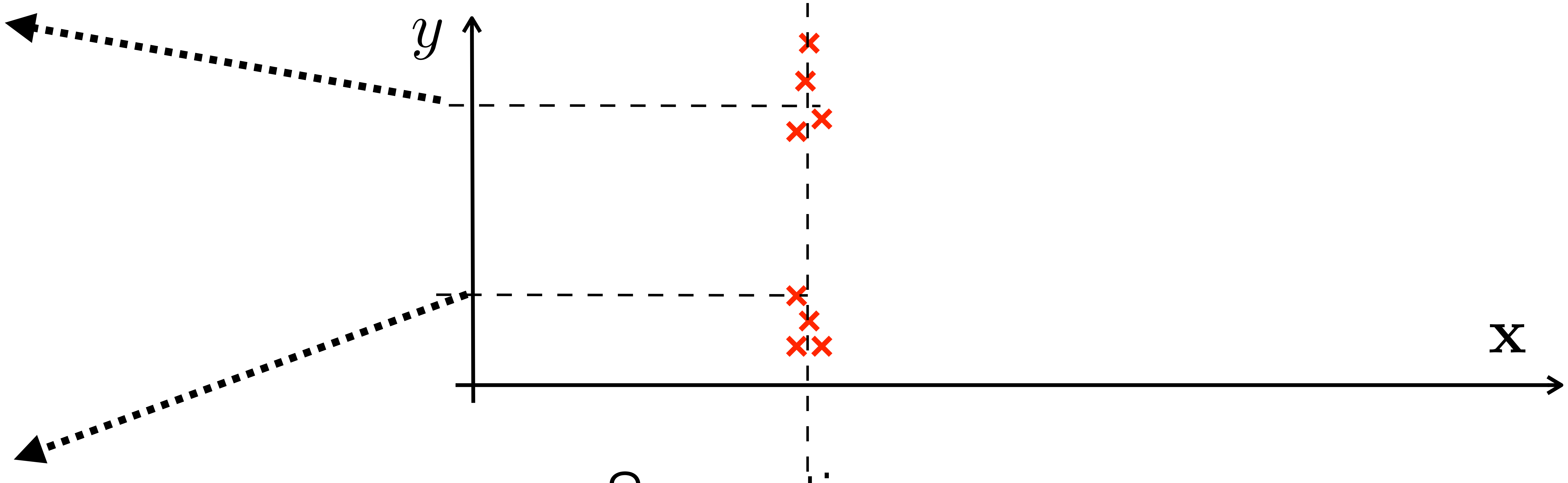
Multiple choice loss
[Microsoft, NIPS, 2012], [Koltun, ICCV, 2017]

$$\mathscr{L}(\mathbf{w}) = \min_i \|f_i(\mathbf{x}, \mathbf{w}) - y\|$$

Problem 2: Measuring similarity of images

# Work-around 3: Conditional Generative Adversarial Networks

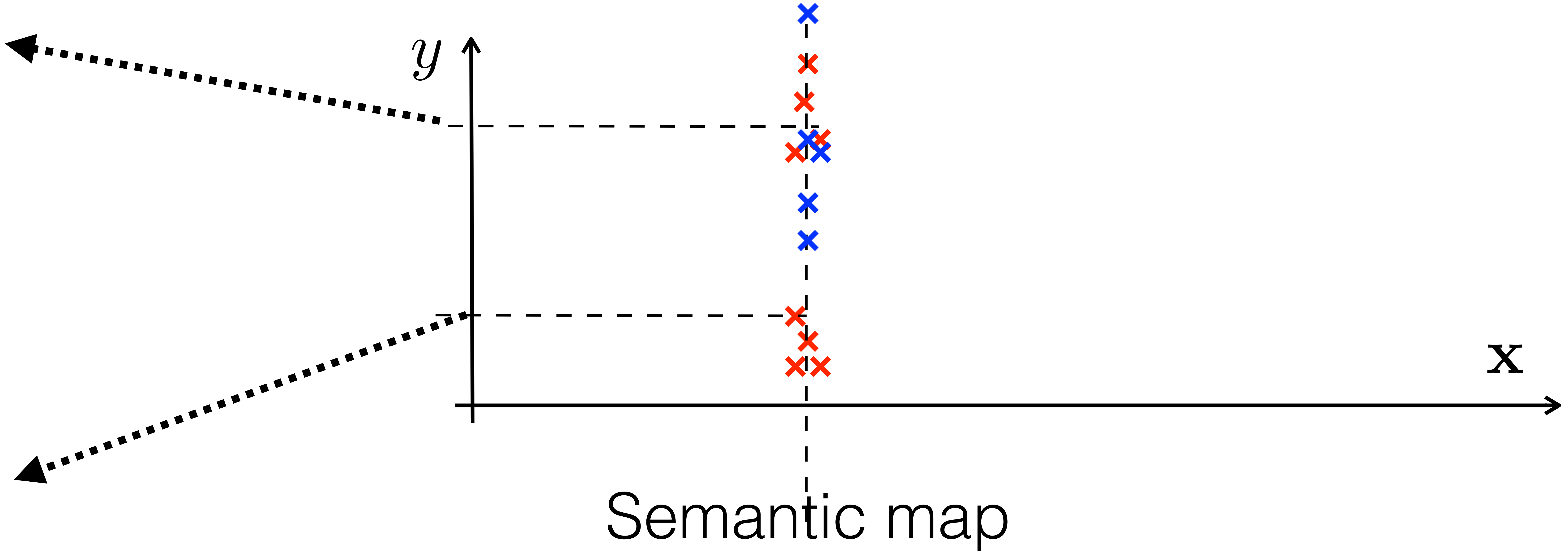
Problem 1: number of hypothesis may grow exponentially




Semantic map

# Work-around 3: Conditional Generative Adversarial Networks

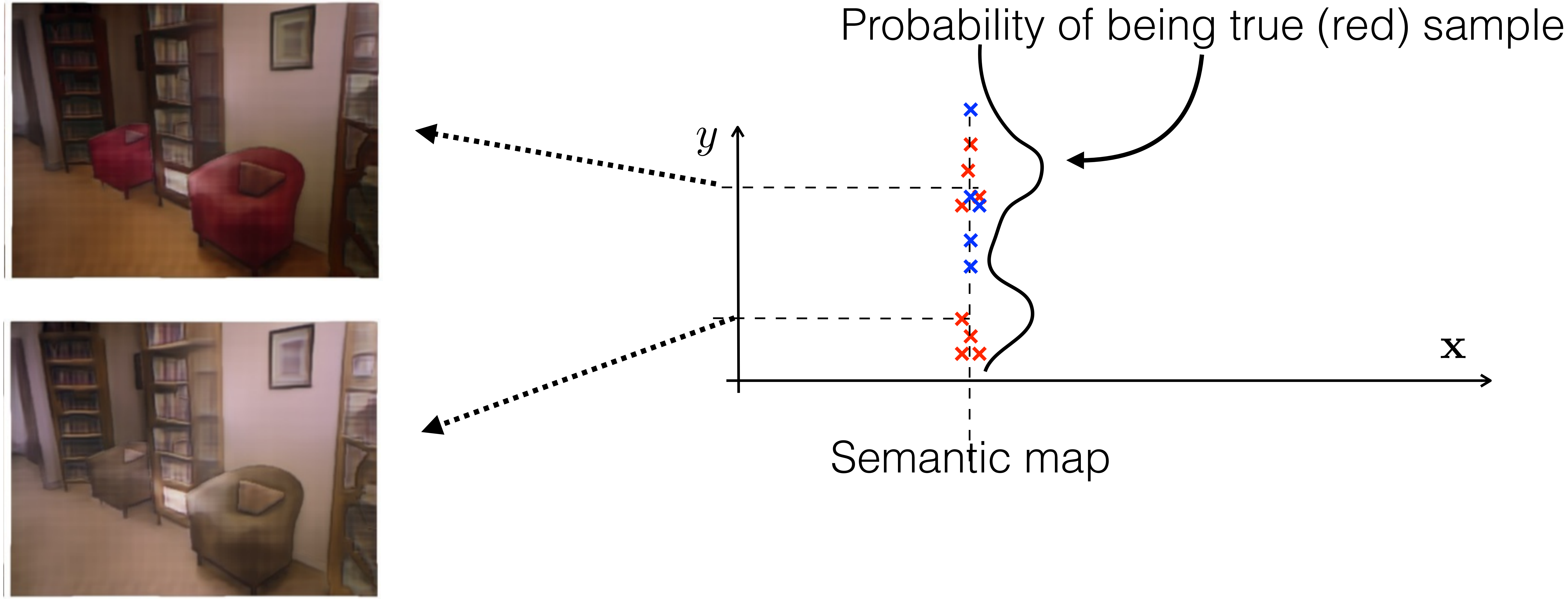
Problem 1: number of hypothesis may grow exponentially




(1) Define **generative model** that generates blue samples (net with injected noise)

## Work-around 3: Conditional Generative Adversarial Networks

Problem 2: Measuring similarity of images

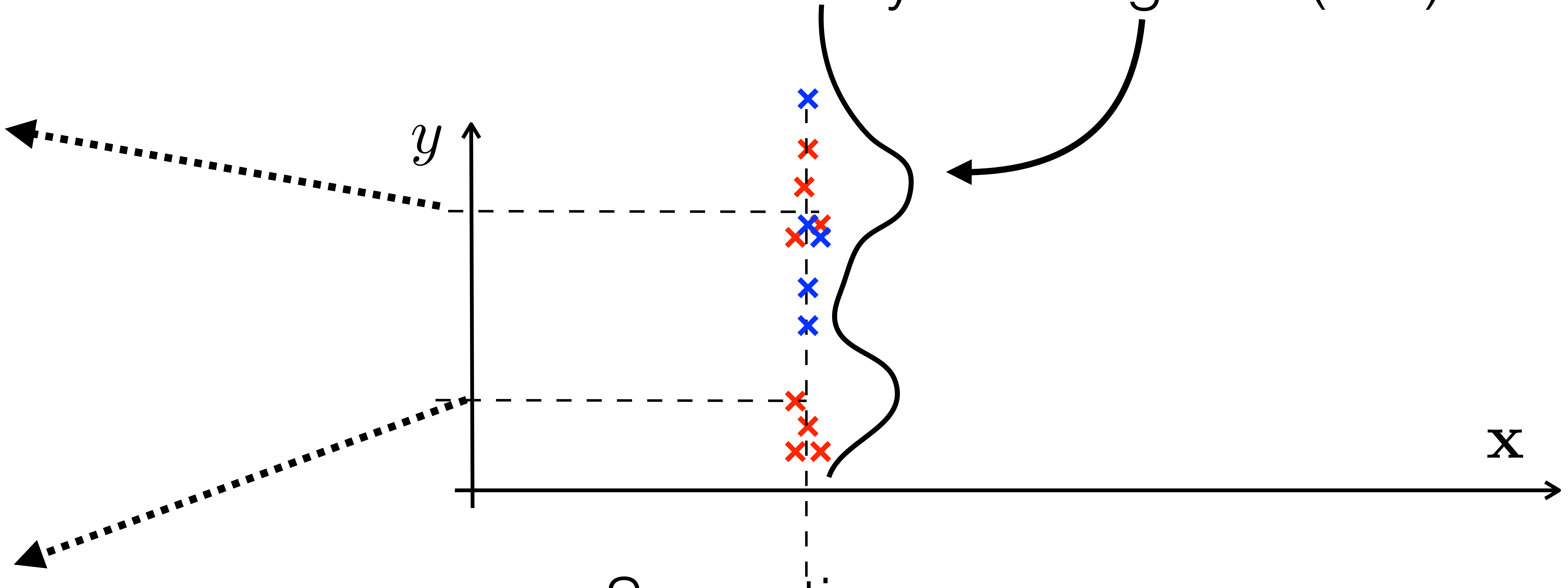



(1) Define **generative model** that generates blue samples (net with injected noise)

(2) Learn **discriminator** (i.e. 2-class classifier that discriminate blue from red)

# Work-around 3: Conditional Generative Adversarial Networks

## Problem 2: Measuring similarity of images



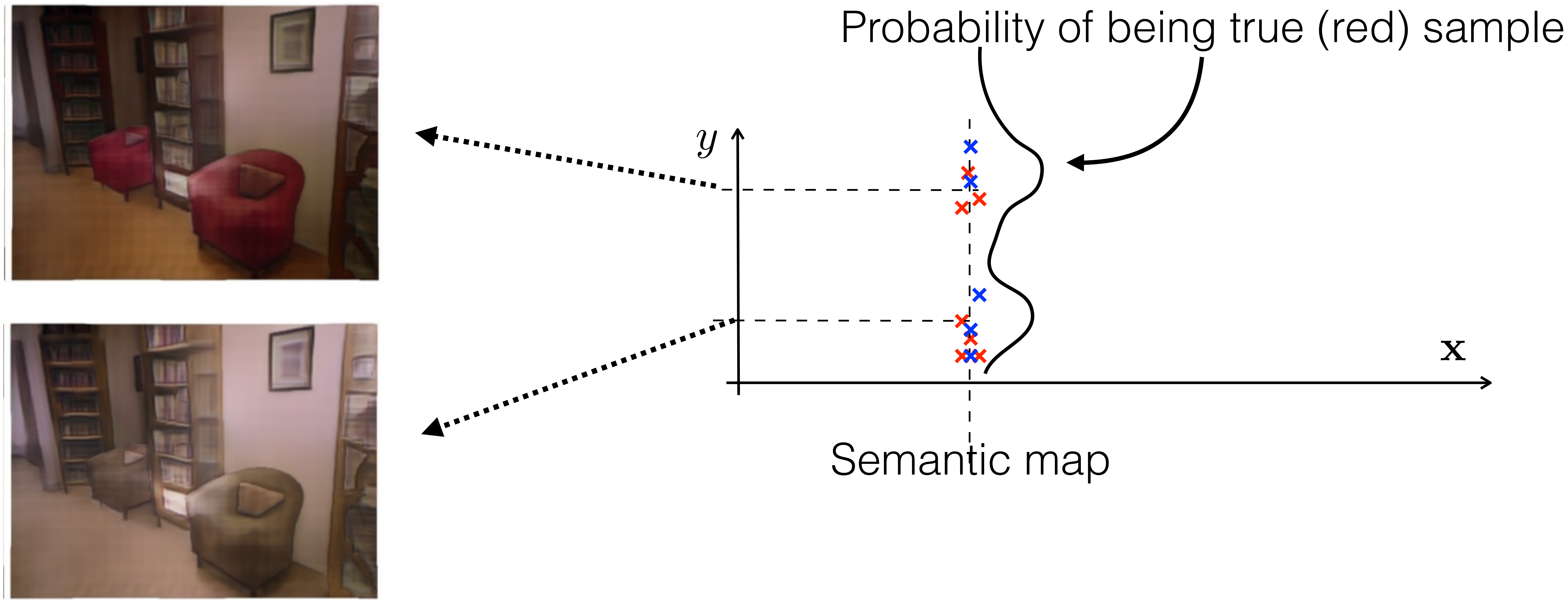

(1) Define **generative model** that generates blue samples (net with injected noise)

(2) Learn **discriminator** (i.e. 2-class classifier that discriminate blue from red)

(3) Learn the blue sample generator to maximize discriminative loss (fool discrimin.)

# Work-around 3: Conditional Generative Adversarial Networks

## Problem 2: Measuring similarity of images



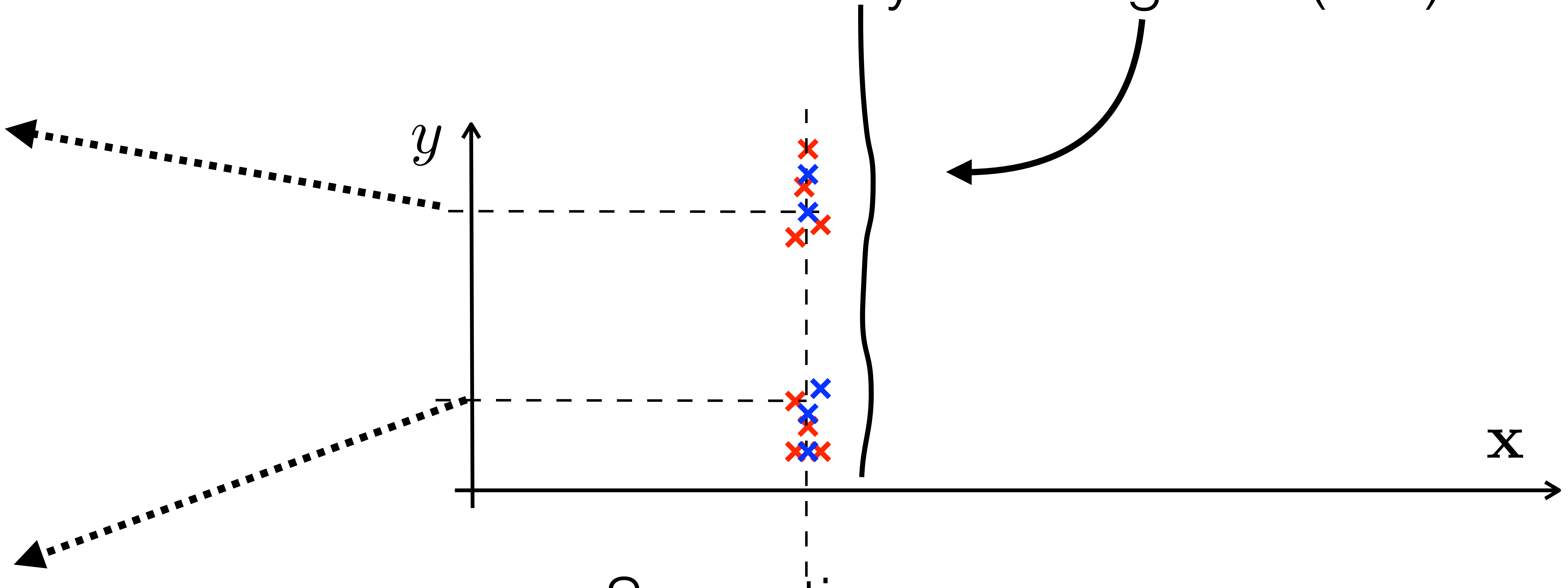

(1) Define **generative model** that generates blue samples (net with injected noise)

(2) Learn **discriminator** (i.e. 2-class classifier that discriminate blue from red)

(3) Learn the blue sample generator to maximize discriminative loss (fool discrimin.)

# Work-around 3: Conditional Generative Adversarial Networks

## Problem 2: Measuring similarity of images




(1) Define **generative model** that generates blue samples (net with injected noise)

(2) Learn **discriminator** (i.e. 2-class classifier that discriminate blue from red)

(3) Learn the blue sample generator to maximize discriminative loss (fool discrimin.)

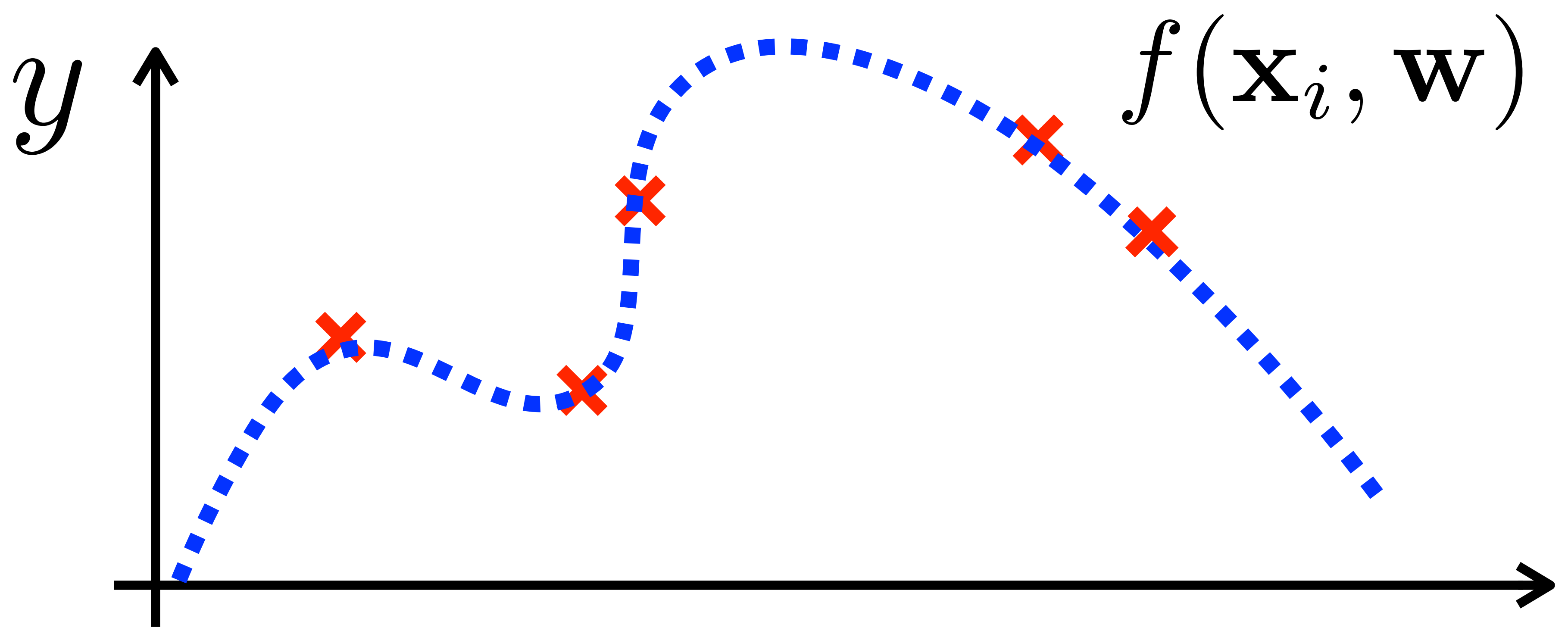
(4) Iterate

# **Prior** is important

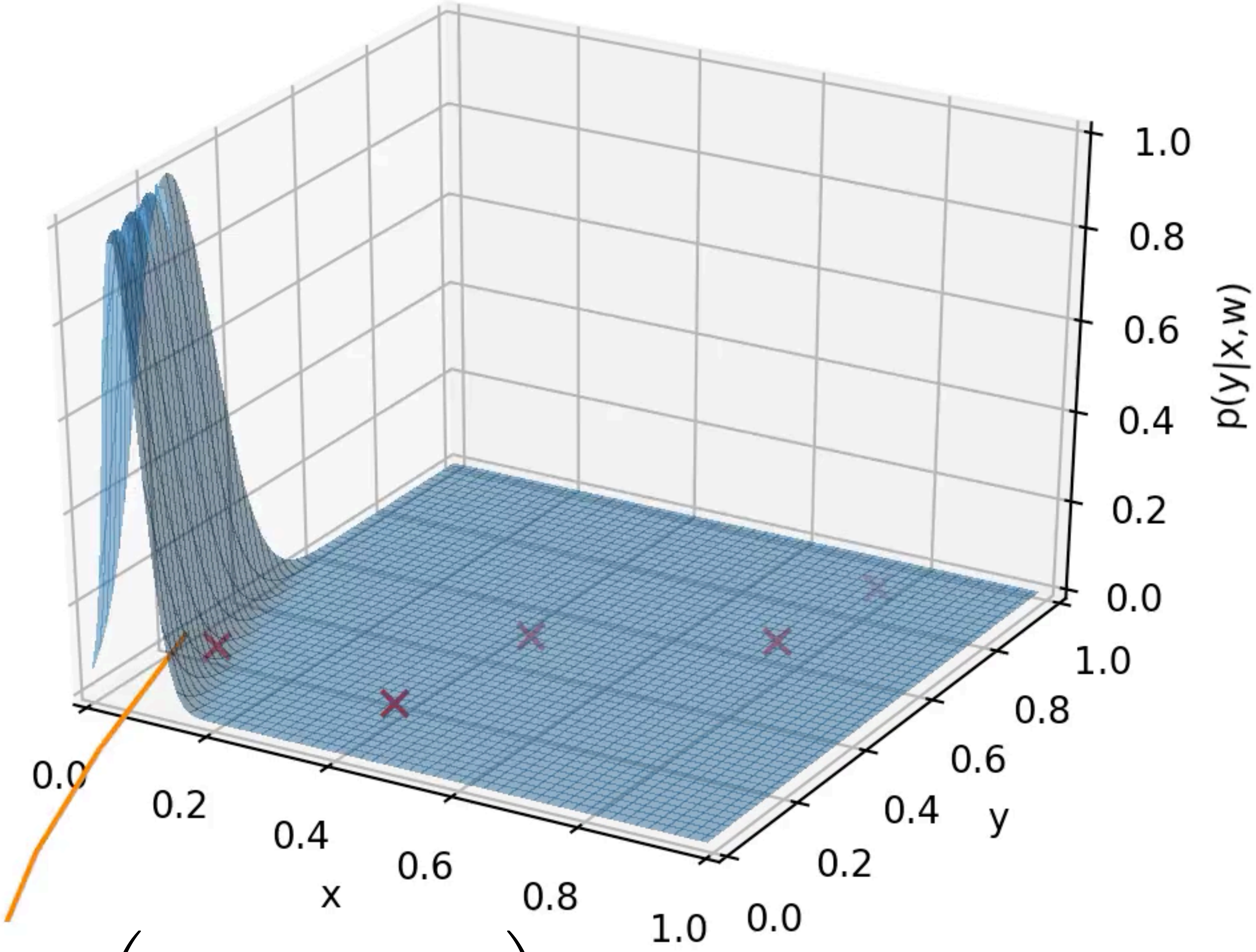
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) \cdot \quad \boxed{\phantom{xxxx}}$$

log likelihood    prior/regulariser

no prior, powerful class of f, learner is oraculum => zero trn error + **overfitting**

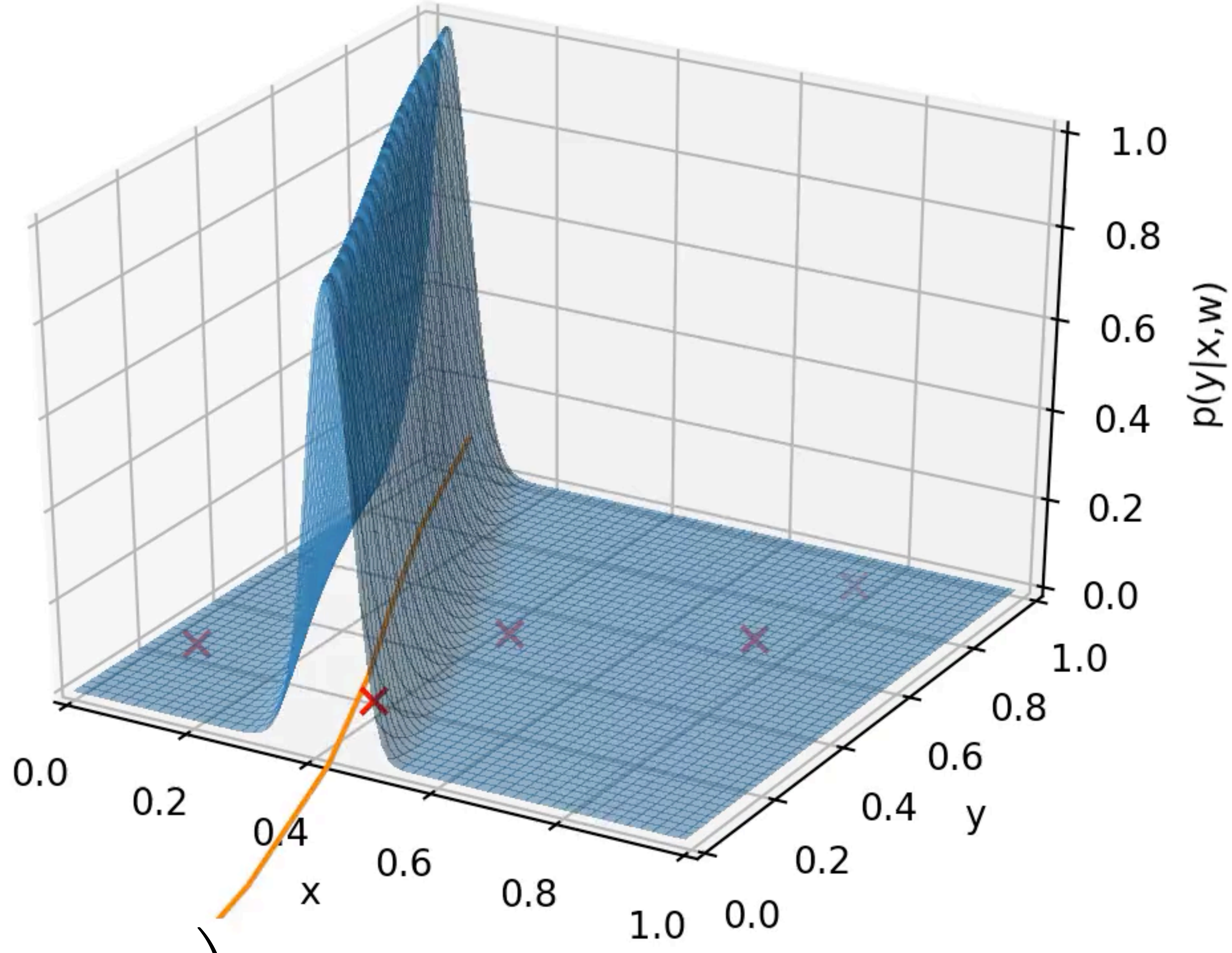


$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(w_2 x^2 + w_1 x + w_0,\ \sigma^2)$$



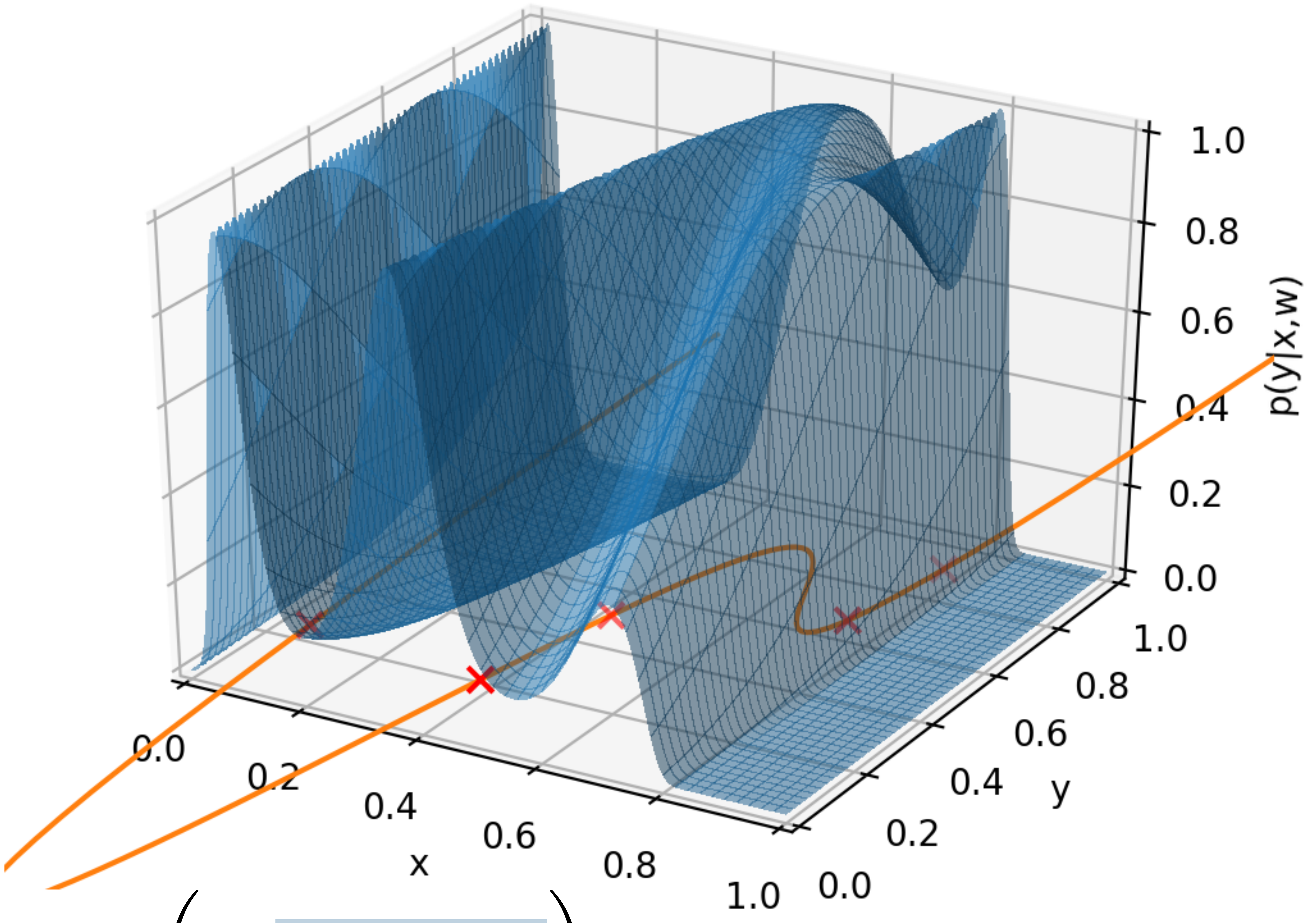

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_2 x_i^2 + w_1 x_i + w_0 - y_i)^2$$

$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(w_4 x^4 + w_3 x^3 + w_2 x^2 + w_1 x + w_0,\ \sigma^2)$$




$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_4 x_i^4 + w_3 x_i^3 + w_2 x_i^2 + w_1 x_i + w_0 - y_i)^2$$

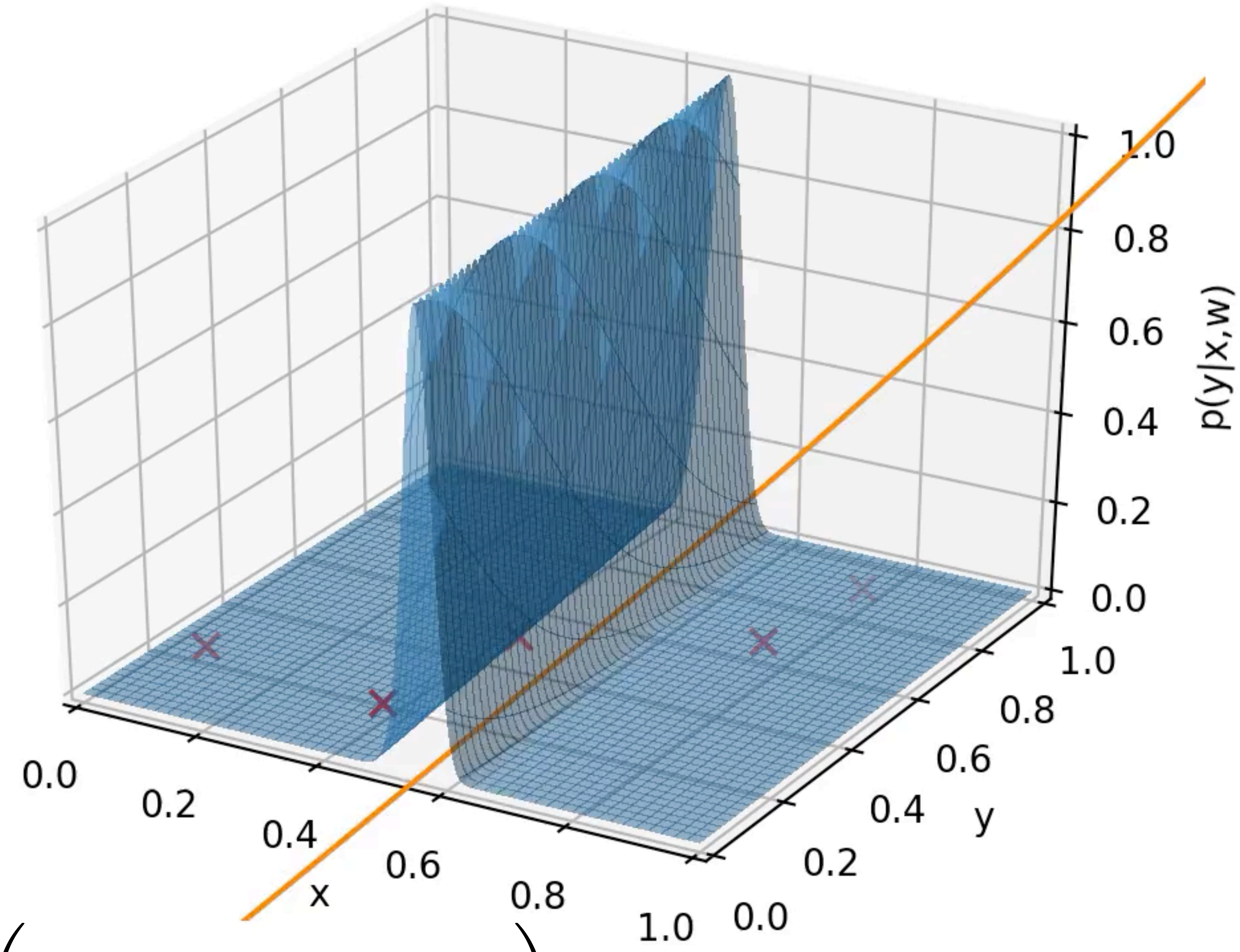
$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$$




$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$$

$$p(\mathbf{w}) \sim \mathcal{N}_w(\mathbf{0},\ \sigma^2)$$




$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left( \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) p(\mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2 + \mathbf{w}^\top \mathbf{w}$$

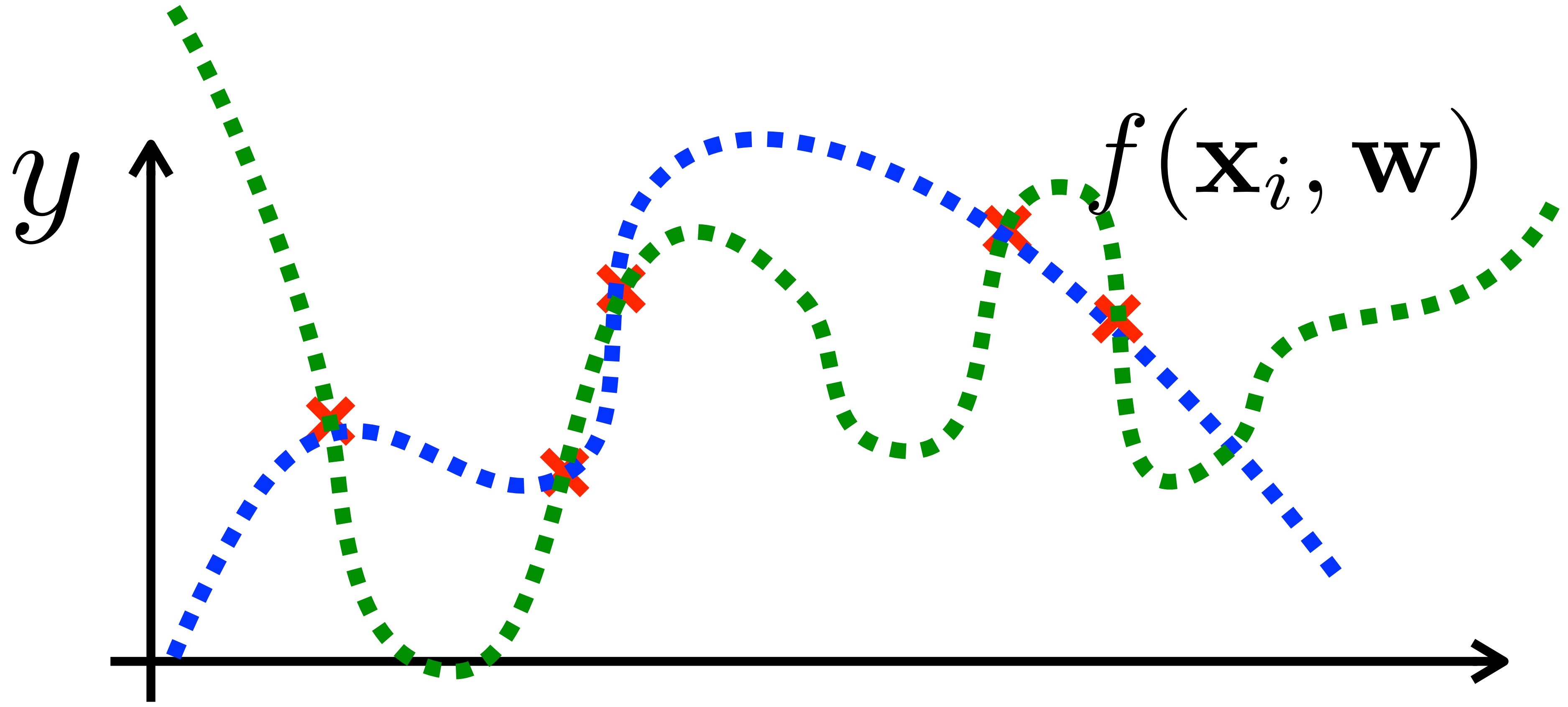
# **Prior** is important

Phaistos disc

Unicode

101D0 PHAISTOS DISC SIGN PEDESTRIAN
101D1 PHAISTOS DISC SIGN PLUMED HEAD
101D2 PHAISTOS DISC SIGN TATTOOED HEAD
101D3 PHAISTOS DISC SIGN CAPTIVE
101D4 PHAISTOS DISC SIGN CHILD
101D5 PHAISTOS DISC SIGN WOMAN
101D6 PHAISTOS DISC SIGN HELMET
101D7 PHAISTOS DISC SIGN GAUNTLET
101D8 PHAISTOS DISC SIGN TIARA
101D9 PHAISTOS DISC SIGN ARROW
101DA PHAISTOS DISC SIGN BOW
101DB PHAISTOS DISC SIGN SHIELD

no prior, powerful class of f, learner is oraculum => zero trn error + **overfitting**



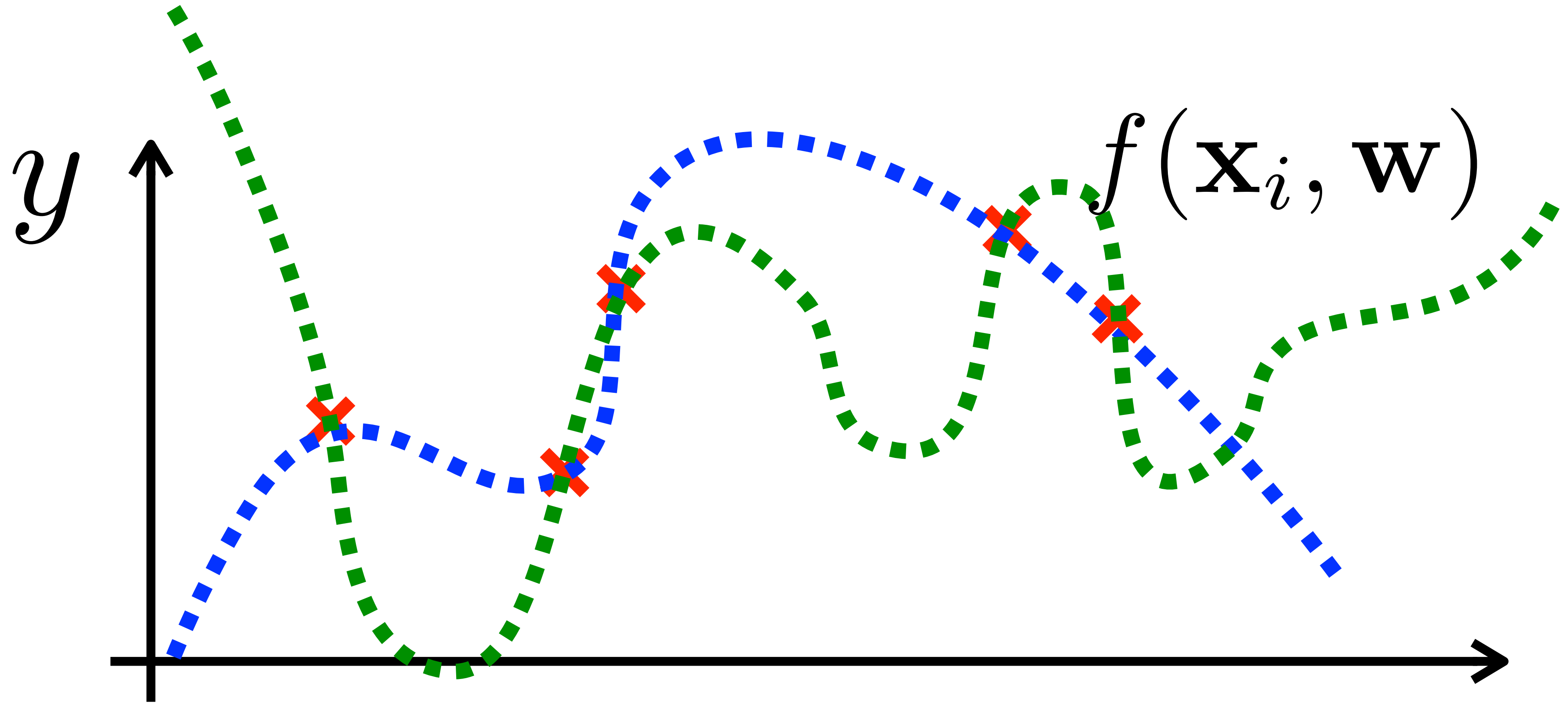

https://en.wikipedia.org/wiki/Phaistos_Disc

## **Prior** is important

William of Ockham
(1287-1347)

leprechauns can be involved in any explanation

no prior, powerful class of f, learner is oraculum => zero trn error + **overfitting**



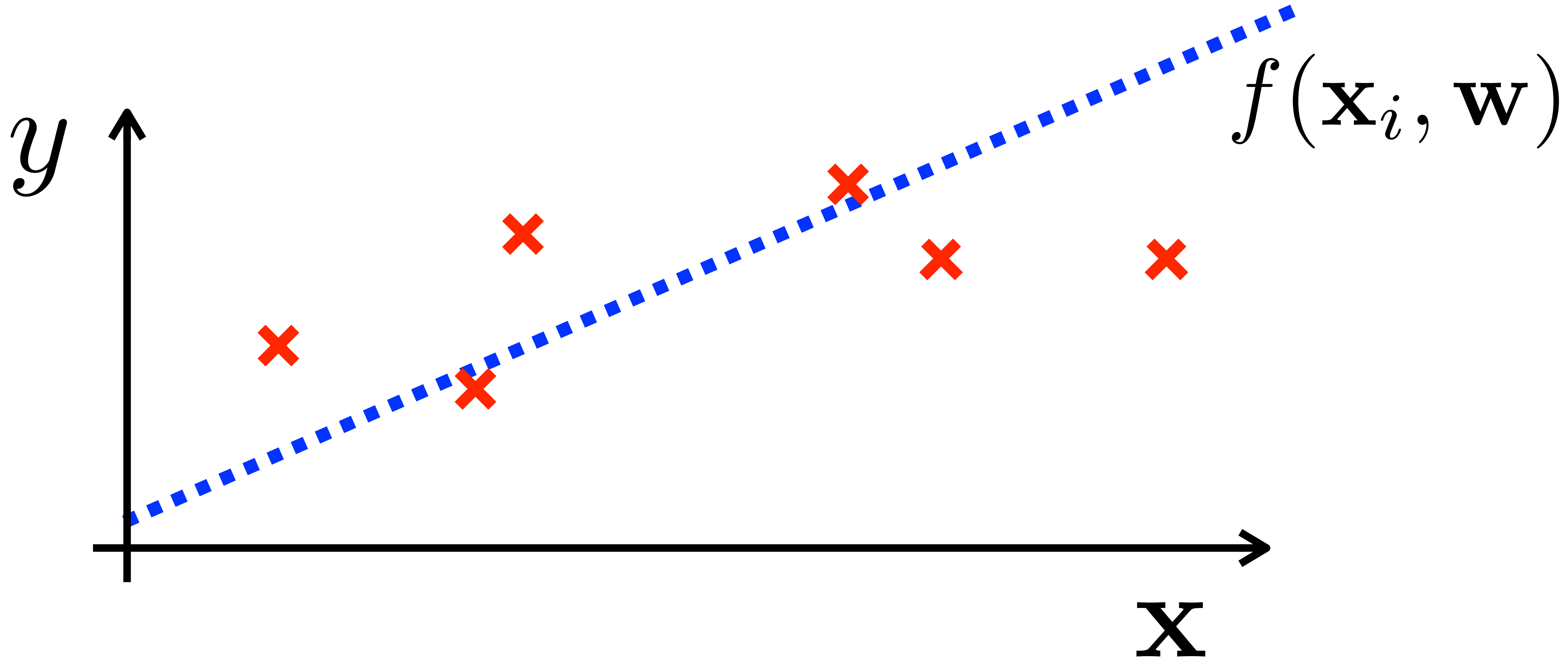

https://en.wikipedia.org/wiki/Occam%27s_razor

# **Prior** is important

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood prior/regulariser

too strong prior, simple f => **underfitting**

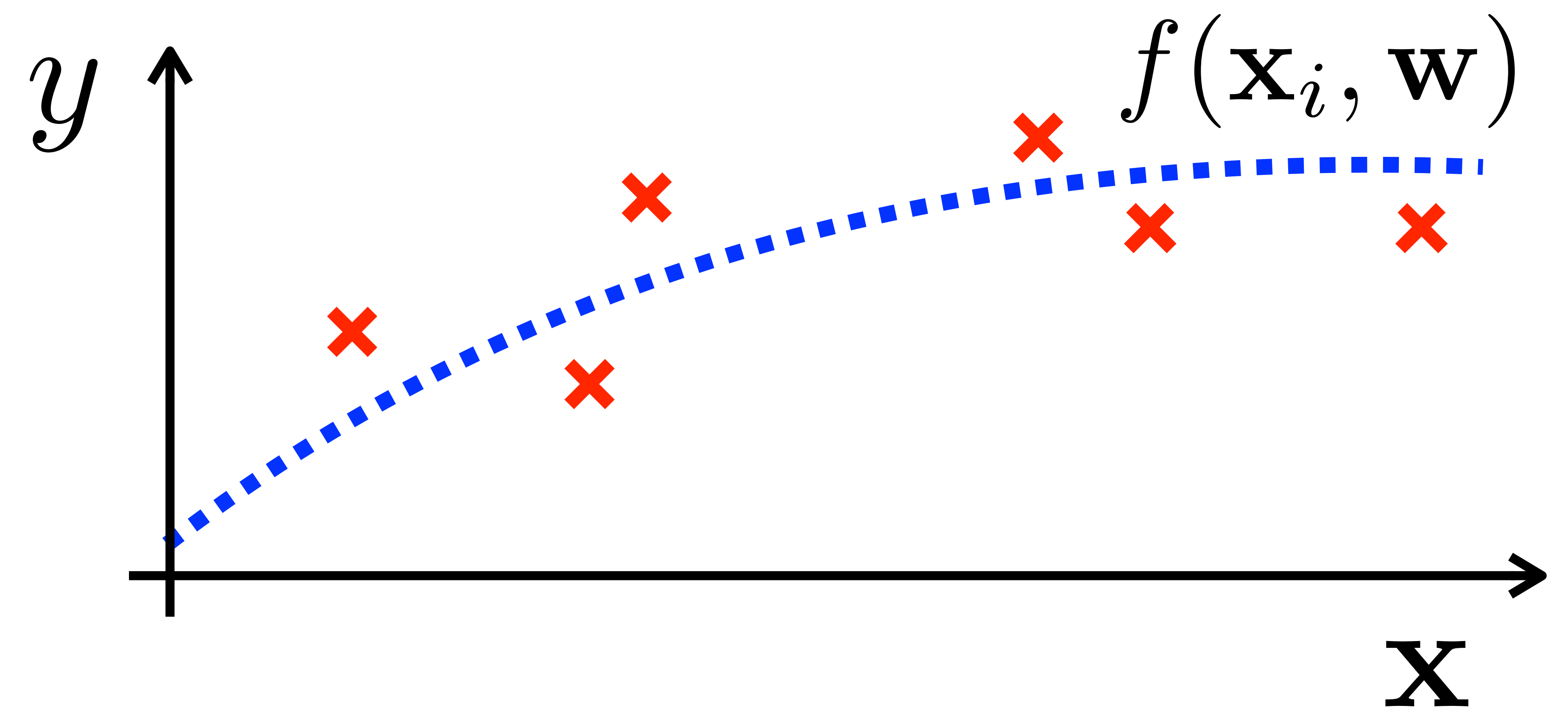


# **Prior** is important

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

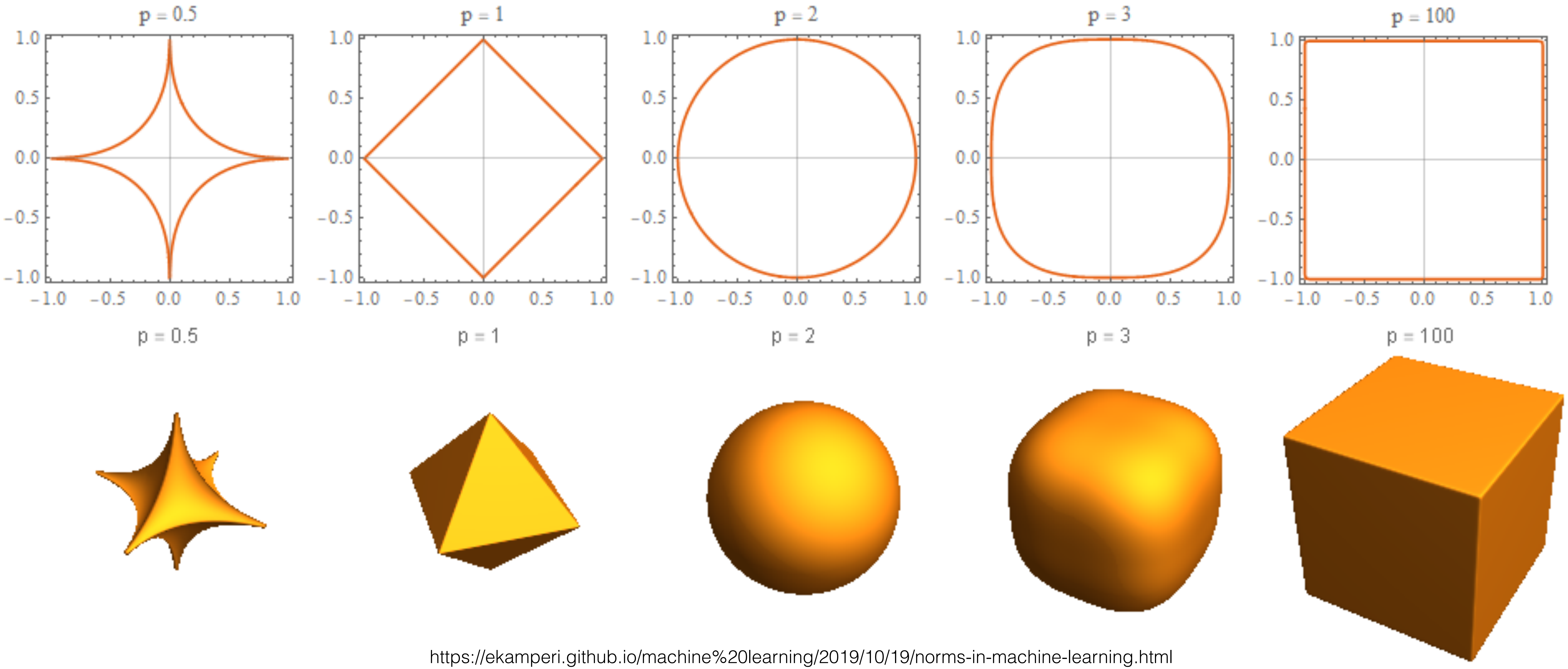
log likelihood prior/regulariser

good prior



# **Prior** is important

Lp-norm: $$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$




https://ekamperi.github.io/machine%20learning/2019/10/19/norms-in-machine-learning.html

# **Prior** is important

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood — prior/regulariser

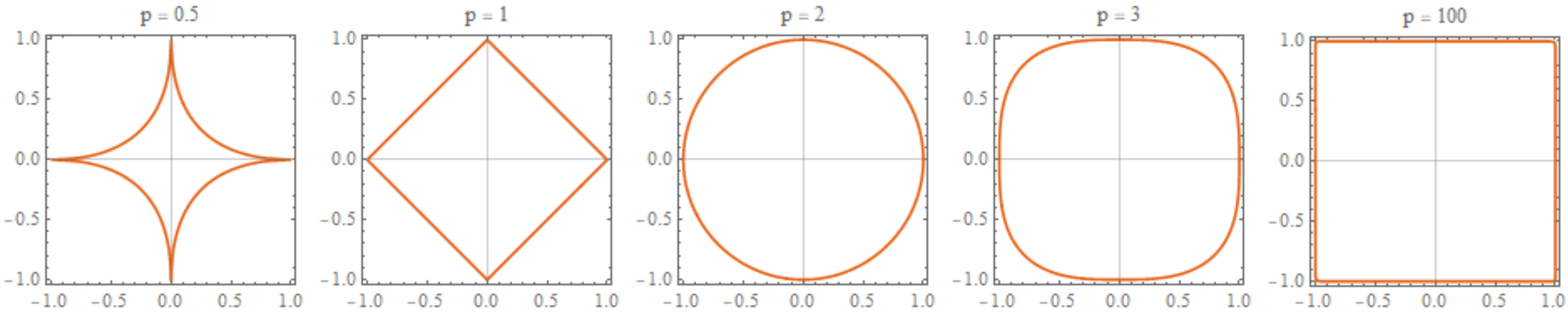
- Gaussian prior $p(\mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}}$ => L2-regularization: $\|\mathbf{w}\|_2^2$

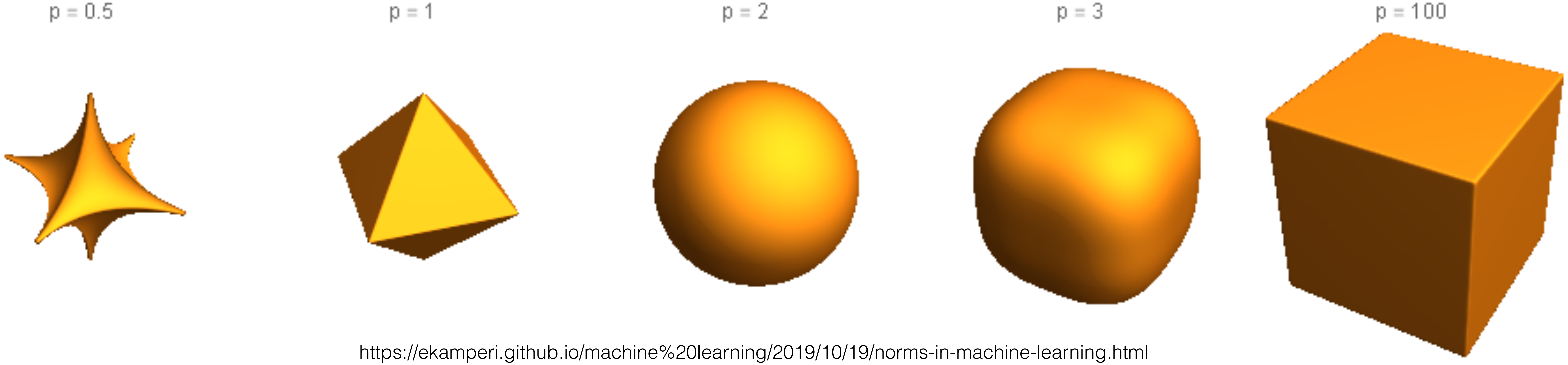
  It says: the smaller the better (does everyone agree???)

**Prior** is important

Lp-norm: $$\|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

Spikes are always good
to fight leprechauns




https://ekamperi.github.io/machine%20learning/2019/10/19/norms-in-machine-learning.html

# **Prior** is important

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood    prior/regulariser

- Gaussian prior $p(\mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}}$ => L2-regularization: $\|\mathbf{w}\|_2^2$

  It says: the smaller the better (does everyone agree???)
- Laplace prior $p(\mathbf{w}) = \frac{1}{2b} e^{(-\frac{|\mathbf{w}|}{b})}$ => L1-regularization: $\|\mathbf{w}\|_1$

  It says: the sparser the better
- L2-regression with L1-regularization is known as Lasso
- Interesting way to avoid overfitting is using a weak learner

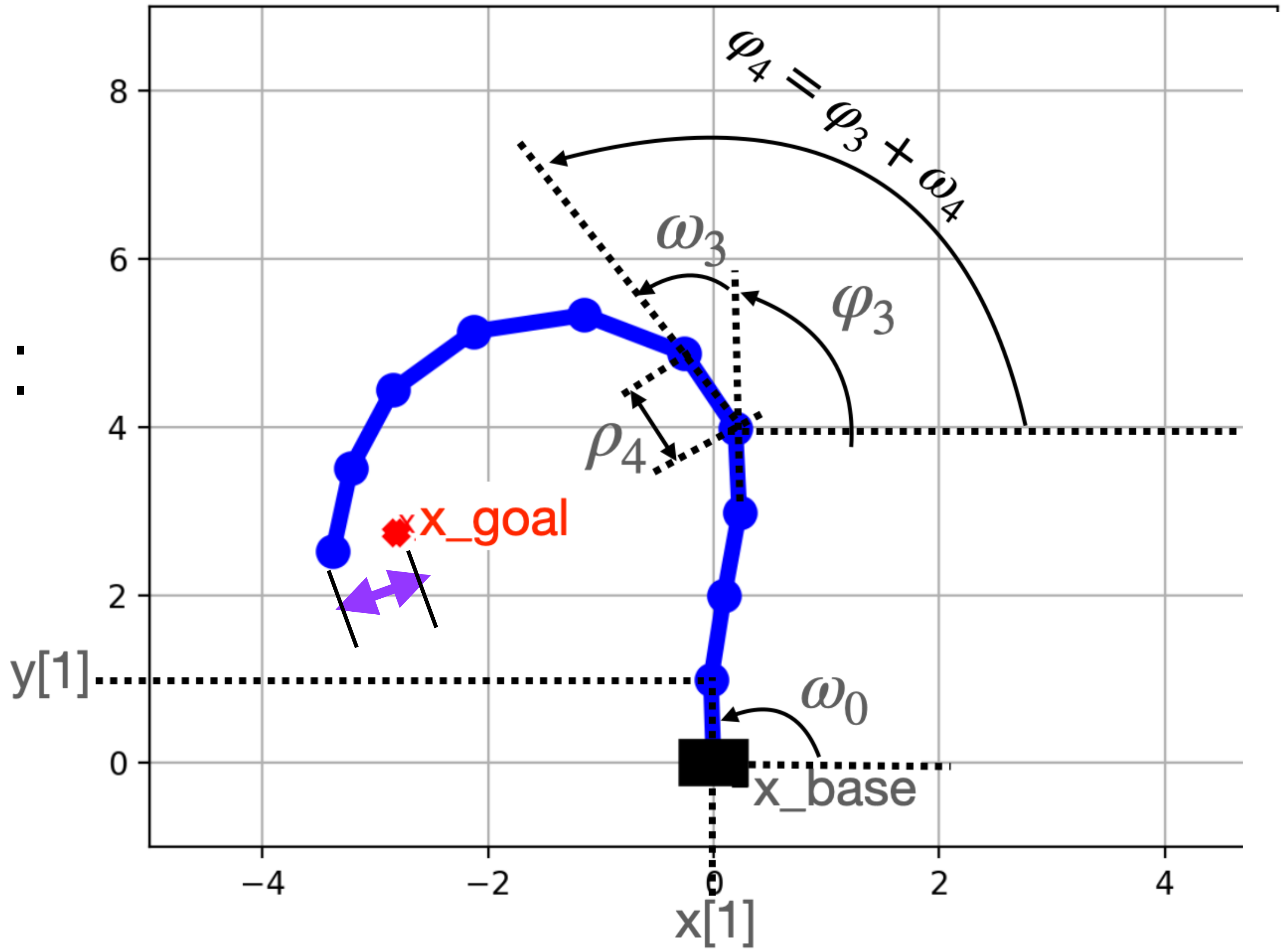
# **Prior** is important

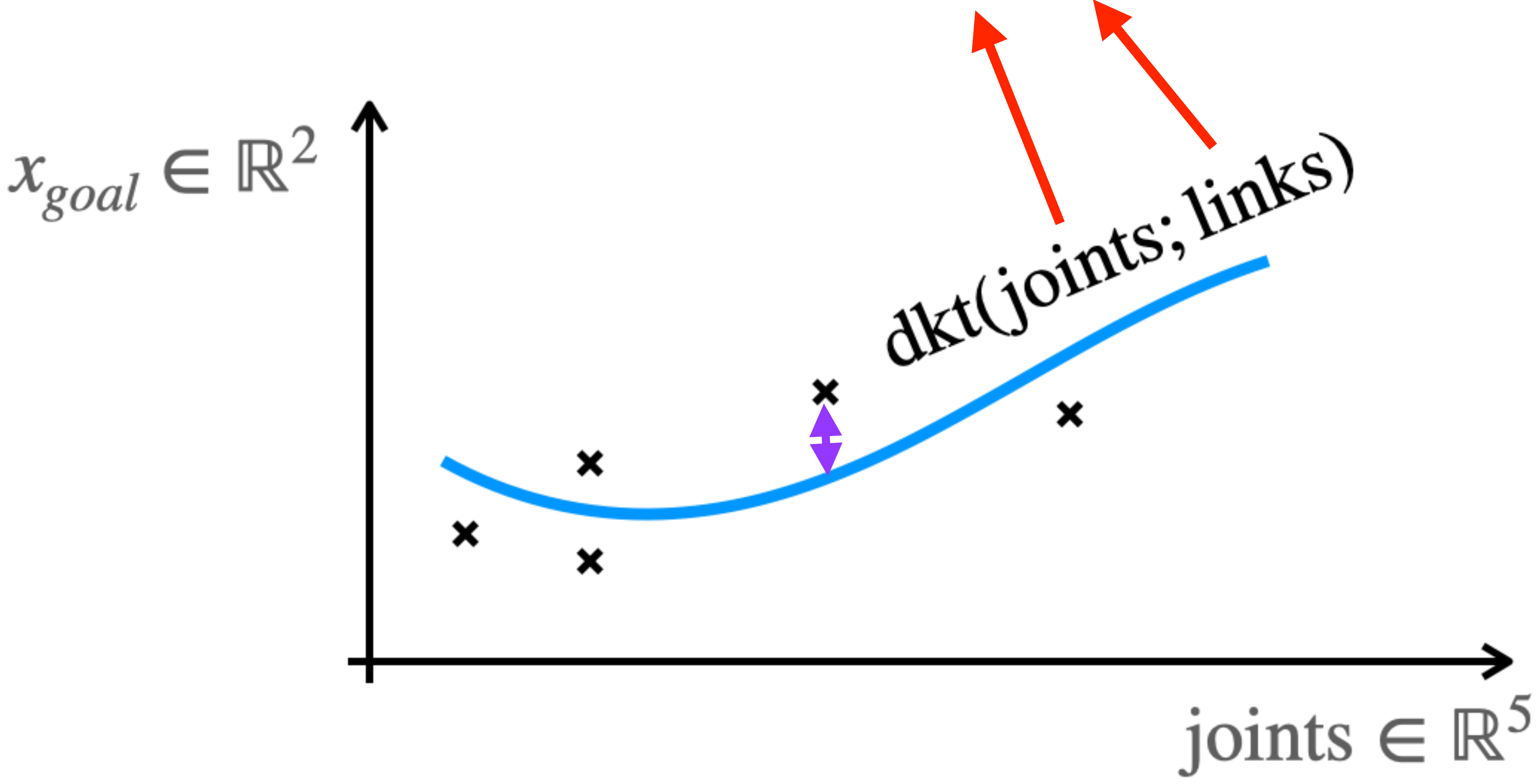
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood prior/regulariser

- Any prior knowledge that restricts the class of functions $f(\mathbf{x}_i, \mathbf{w})$ is $p(\mathbf{w})$

HW1:




Well restricted class of functions is your fortress, where you hide from leprechauns

# Summary

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right)$$

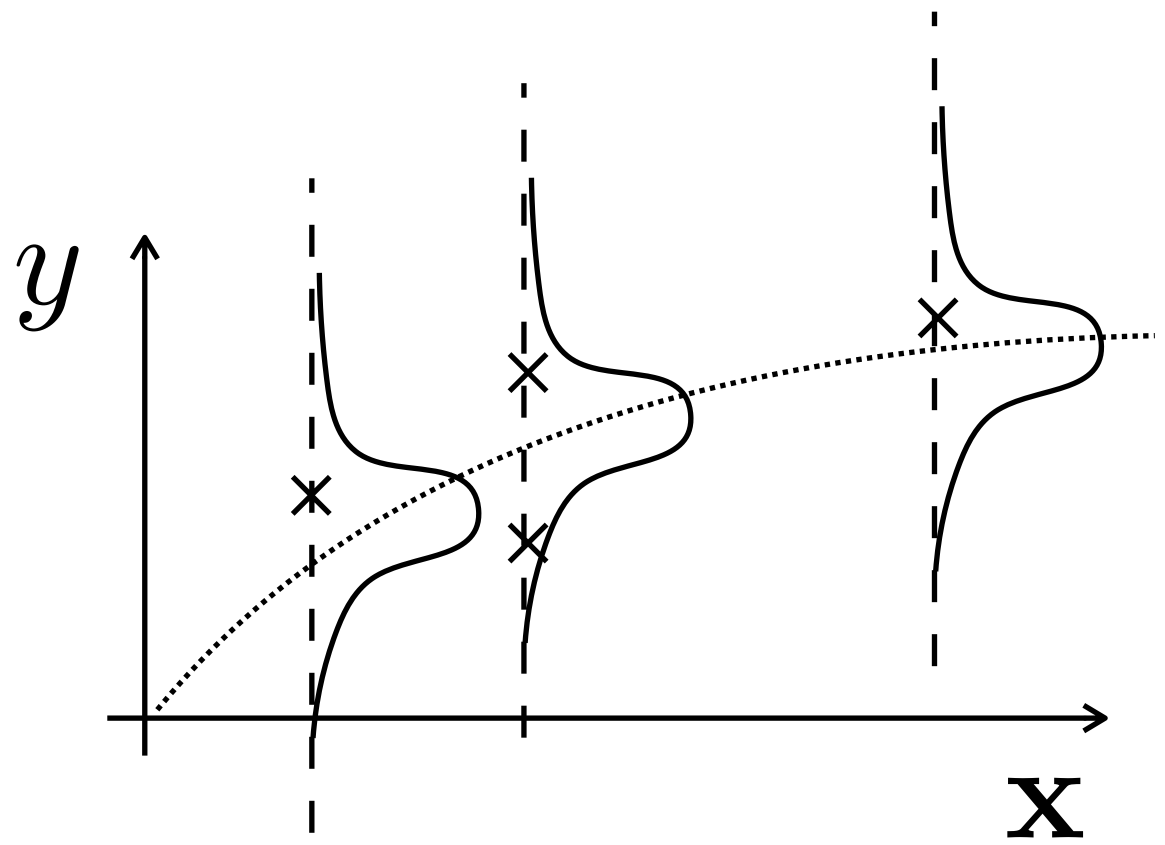
log likelihood

- **Regression:** ML estimate of cont. unimodal distribution (Gauss, Laplace) with mean in $f(\mathbf{x}, \mathbf{w})$

$$p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$$

- Probability of observing $y_i$ when measuring $\mathbf{x}_i$ is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(f(\mathbf{x}_i, \mathbf{w}) - y_i)^2}{2\sigma^2} \right)$$




- **Training:** minimize L2 loss

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

- Especially $f(\mathbf{x}, \mathbf{w})$ linear in **w** yields quadratic loss and has closed-form solution

# Summary

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left( \sum_i -\log(p(y_i|\mathbf{x}_i,\mathbf{w})) \right)$$

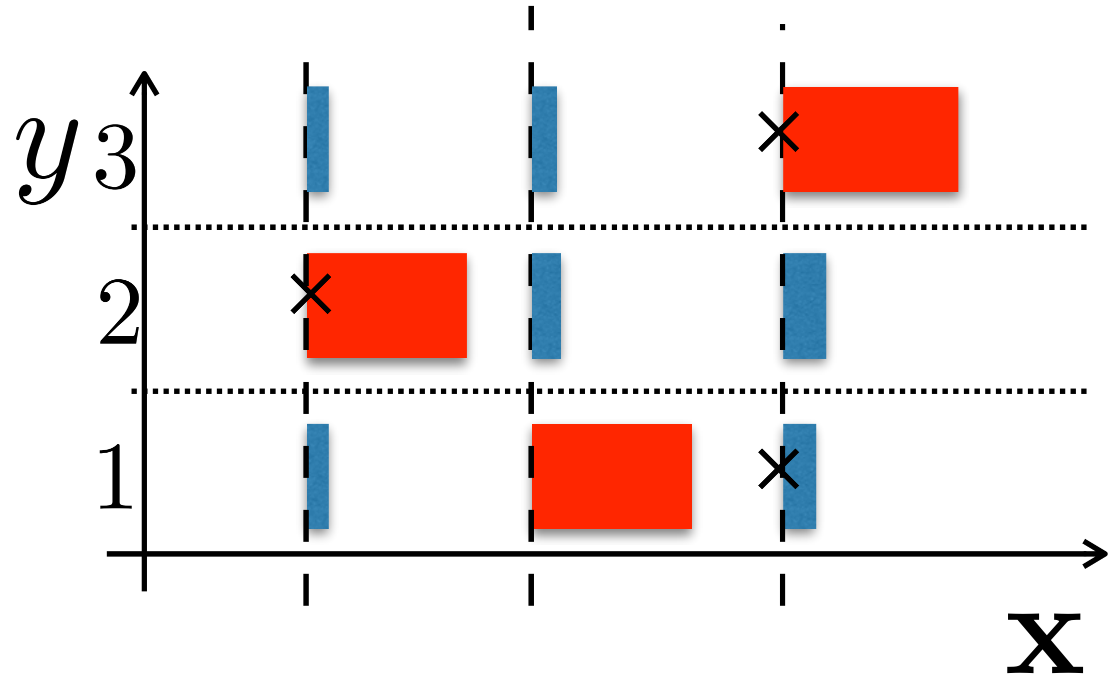
log likelihood

- **Classification:** ML estimate of discrete prob.d. modelled by soft-max function:

$$p(y|\mathbf{x},\mathtt{W}) = \begin{bmatrix} \exp(f(\mathbf{x},\mathbf{w}_1)) \\ \exp(f(\mathbf{x},\mathbf{w}_2)) \\ \exp(f(\mathbf{x},\mathbf{w}_3)) \end{bmatrix} / \sum_k \exp(f(\mathbf{x},\mathbf{w}_k) = \mathbf{s}(\mathbf{f}(\mathbf{x},\mathtt{W}))$$

- Probability of observing $y_i$ when measuring $\mathbf{x}_i$ is

$$p(y_i|\mathbf{x}_i,\mathtt{W}) = \mathbf{s}_{y_i}(\mathbf{f}(\mathbf{x}_i,\mathtt{W}))$$




- **Training:** minimization of cross-entropy/logistic loss

$$\mathtt{W}^* = \arg\min_{\mathtt{W}} \sum_i -\log \mathbf{s}_{y_i}(\mathbf{f}(\mathbf{x}_i,\mathtt{W}))$$

# Summary

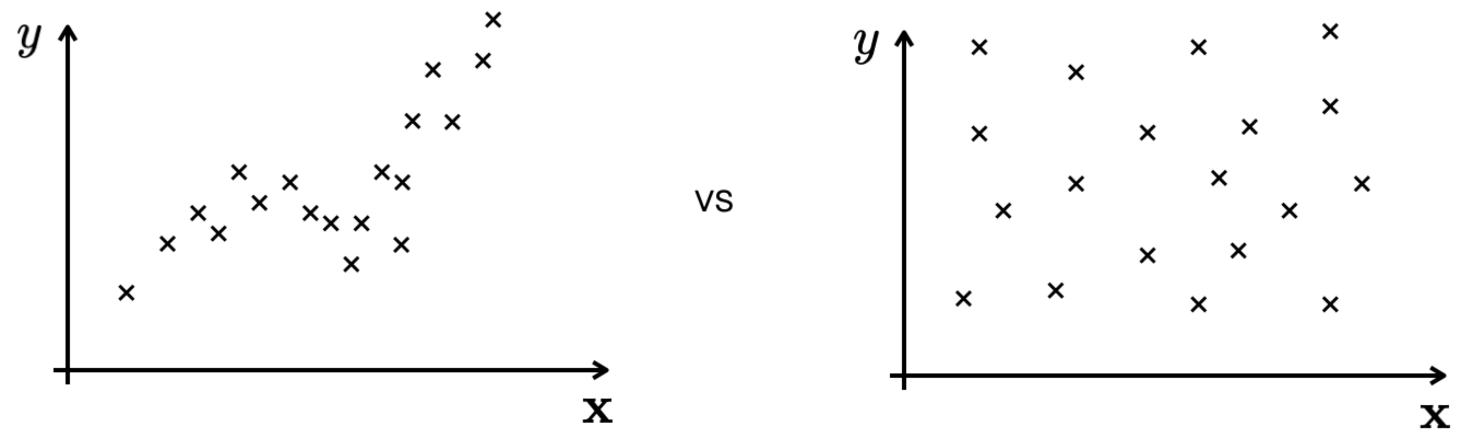
$$+ \left(-\log p(\mathbf{w})\right)$$

prior/regulariser

- Prior is important: Ideally restricts the model to the class of functions that:
  - Avoid any “*not-well justified leprechauns*” in the model, => avoid overfitting
  - Avoid oversimplifications of the model, => avoid underfitting
- Robotics study different models to solve different problems:
  - Projective transformation of pinhole cameras (for camera calibration or stereo)
  - Geometry of Euclidean motion (for point cloud alignment, direct kinematic tasks)
  - Motion model of robots such Dubins car, flight, pendulum (for planning/control)
  - Structure of animal cortex (for ConvNets)

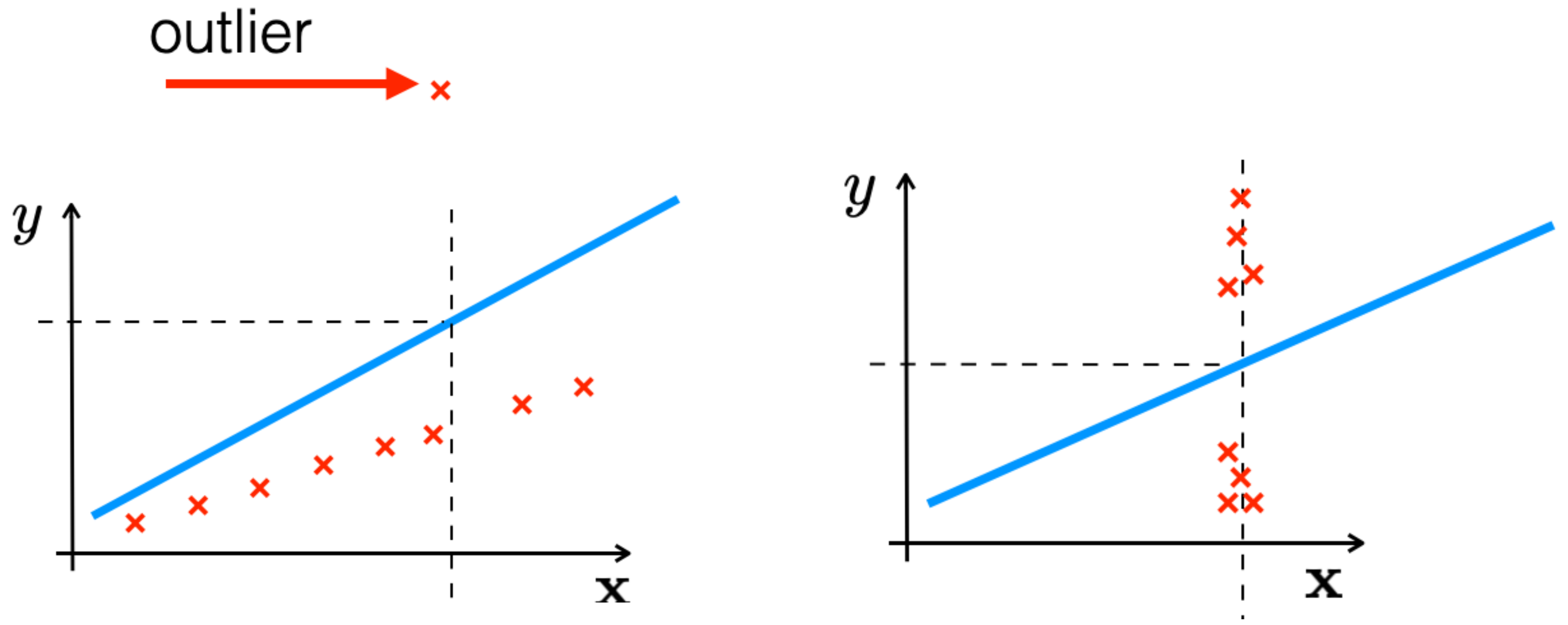
# Golden grale:

- Solve only “Pilcik-free” problems

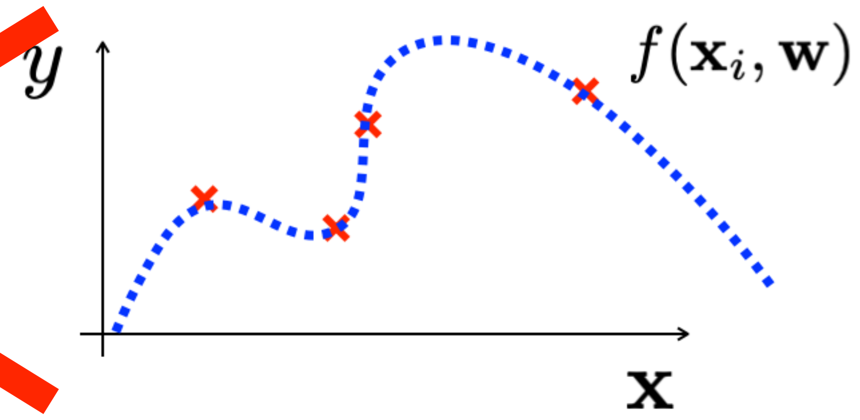

Lecture 1

- Use “Morty-free” data (or at least correct noise model)


Lecture 3

- Provide “sufficiently rich” + “lepricon-free” model.


Lecture 4
ConvNets

- Avoid traps in learning

Lecture 5
Optimizers

# Conclusions

- Explained regression and classification as MAP/ML estimator
- Discussed under/overfitting and regularisations
- Summarized

**Competencies required for the test T1**

- Derive MAP/ML estimate for regression and classification for different noise models
- Derive L2/L1/cross-entropy/logistic losses,
- Understand difference between loss, likelihood and prior
- Understand role of prior in underfitting/overfitting.