Where the hell does the loss and the overfitting come from?

KL divergence, MAP, MLE view of regression, classification and overfitting

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

Prerequisites: Bayes theorem

$$p(A, B) = p(A \mid B) \cdot p(B) = p(B \mid A) \cdot p(A)$$

$$p(A \mid B) = \frac{p(B \mid A) p(A)}{p(B)}$$

The same valid even if all probabilities conditioned by another event C

$$p(A, B \mid C) = p(A \mid B, C) \cdot p(B \mid C) = p(B \mid A, C) \cdot p(A \mid C)$$

$$p(A \mid B, C) = \frac{p(B \mid A, C) p(A \mid C)}{p(B \mid C)}$$

Prerequisites: Independence

Bayes theorem: $p(A, B) = p(A \mid B) \cdot p(B) = p(B \mid A) \cdot p(A)$

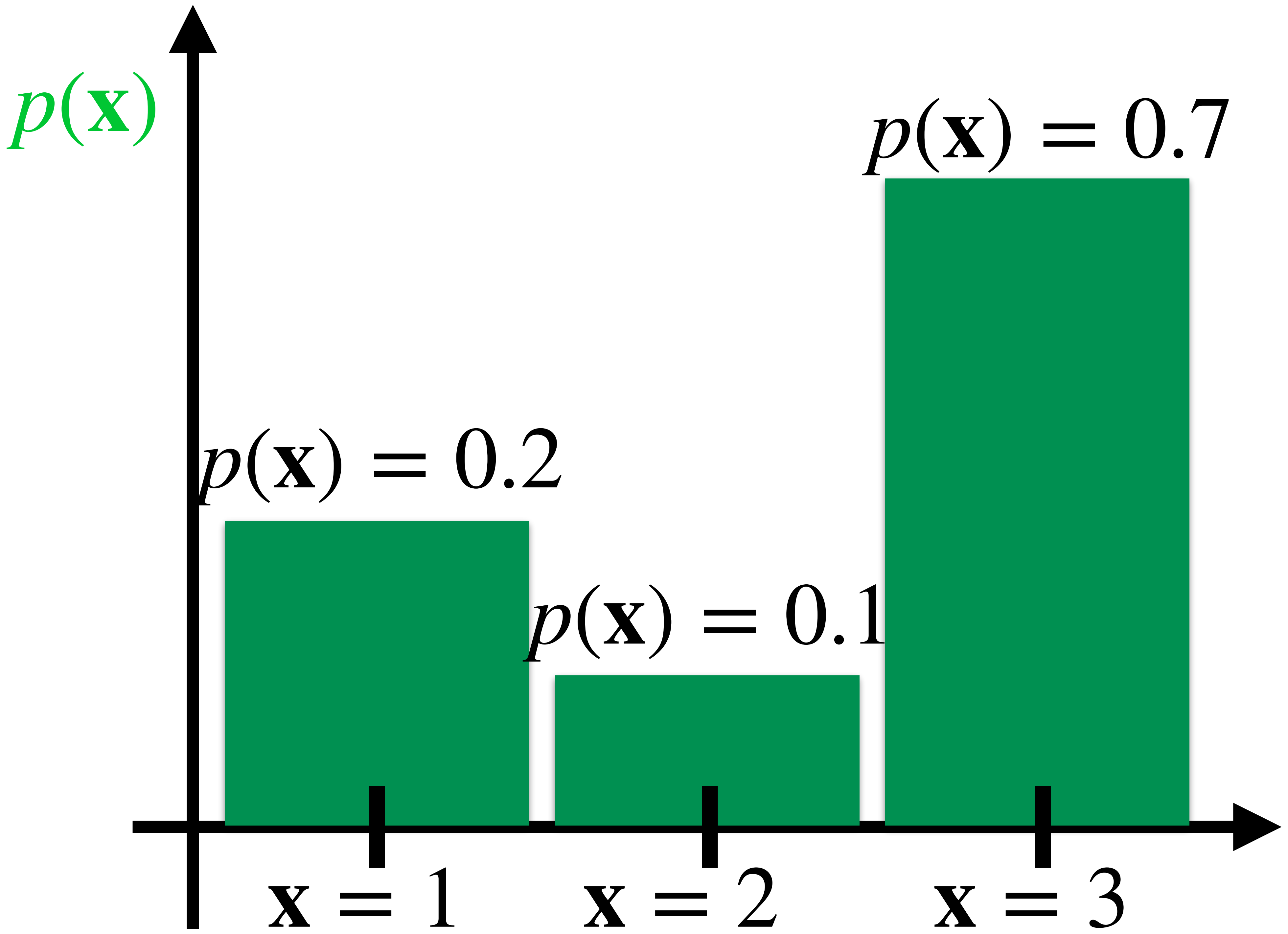
If A and B independent: $p(A, B) = p(A) \cdot p(B)$

Let’s put it together: $p(A, B) = p(A) \cdot p(B) = p(A \mid B) \cdot p(B)$

$$p(A) = p(A \mid B)$$

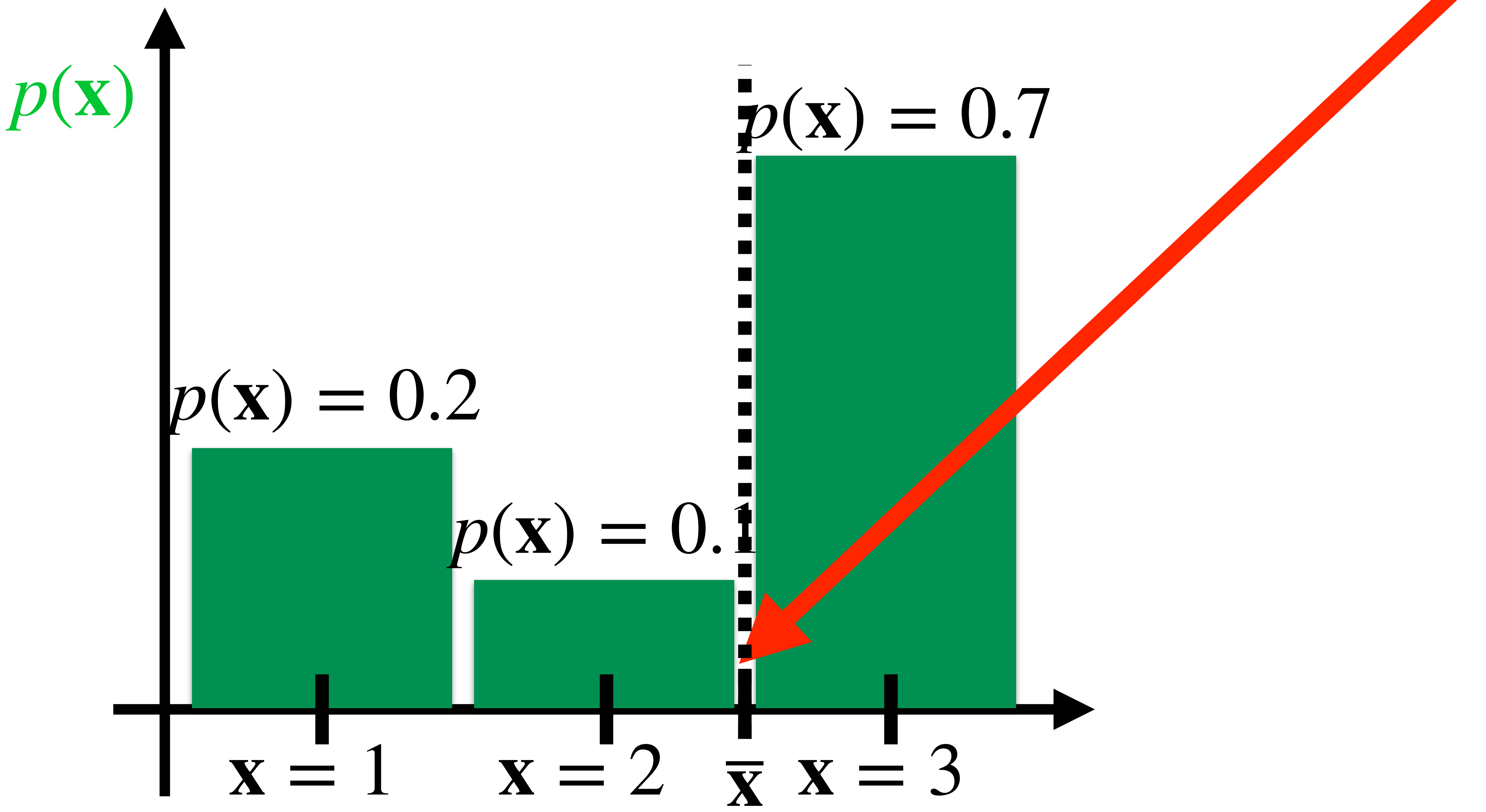
Prerequisites: Mean and average

$$\bar{\mathbf{x}} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \mathbf{x} = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[\mathbf{x}] = ??$$



Prerequisites: Mean and average

$$\bar{\mathbf{x}} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\mathbf{x}\right] = 0.2 \cdot 1 \; + \; 0.1 \cdot 2 \; + \; 0.7 \cdot 3 = 2.5$$

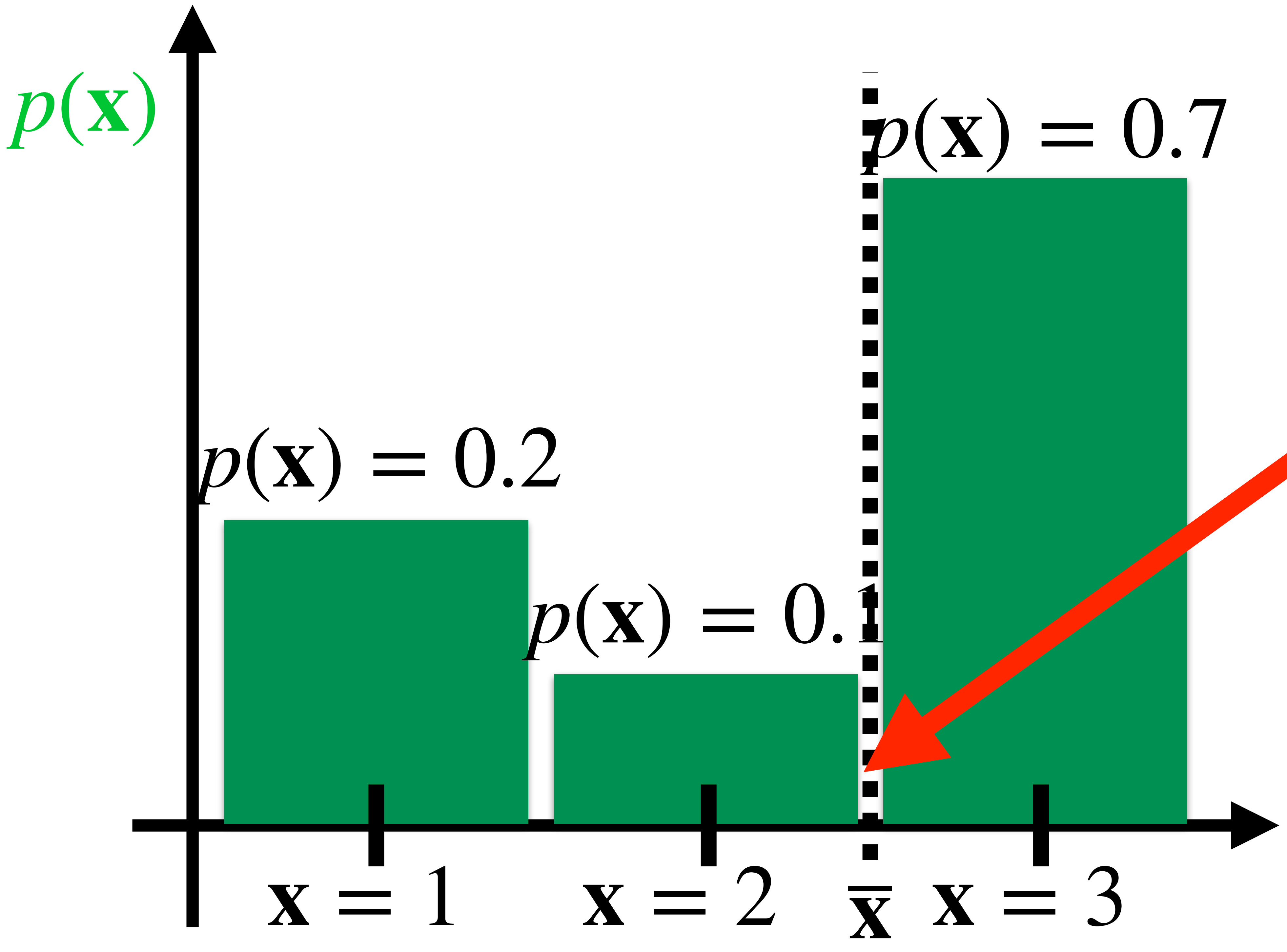


Prerequisites: Mean and average

$$\bar{\mathbf{x}} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\mathbf{x}\right] = 0.2 \cdot 1 \ + \ 0.1 \cdot 2 \ + \ 0.7 \cdot 3 = 2.5$$

$$\approx \frac{1}{N}\sum_{i} \mathbf{x}_i \ = \frac{1}{10}(1+1+2+3+3+3+3+3+3+3) = 2.5$$

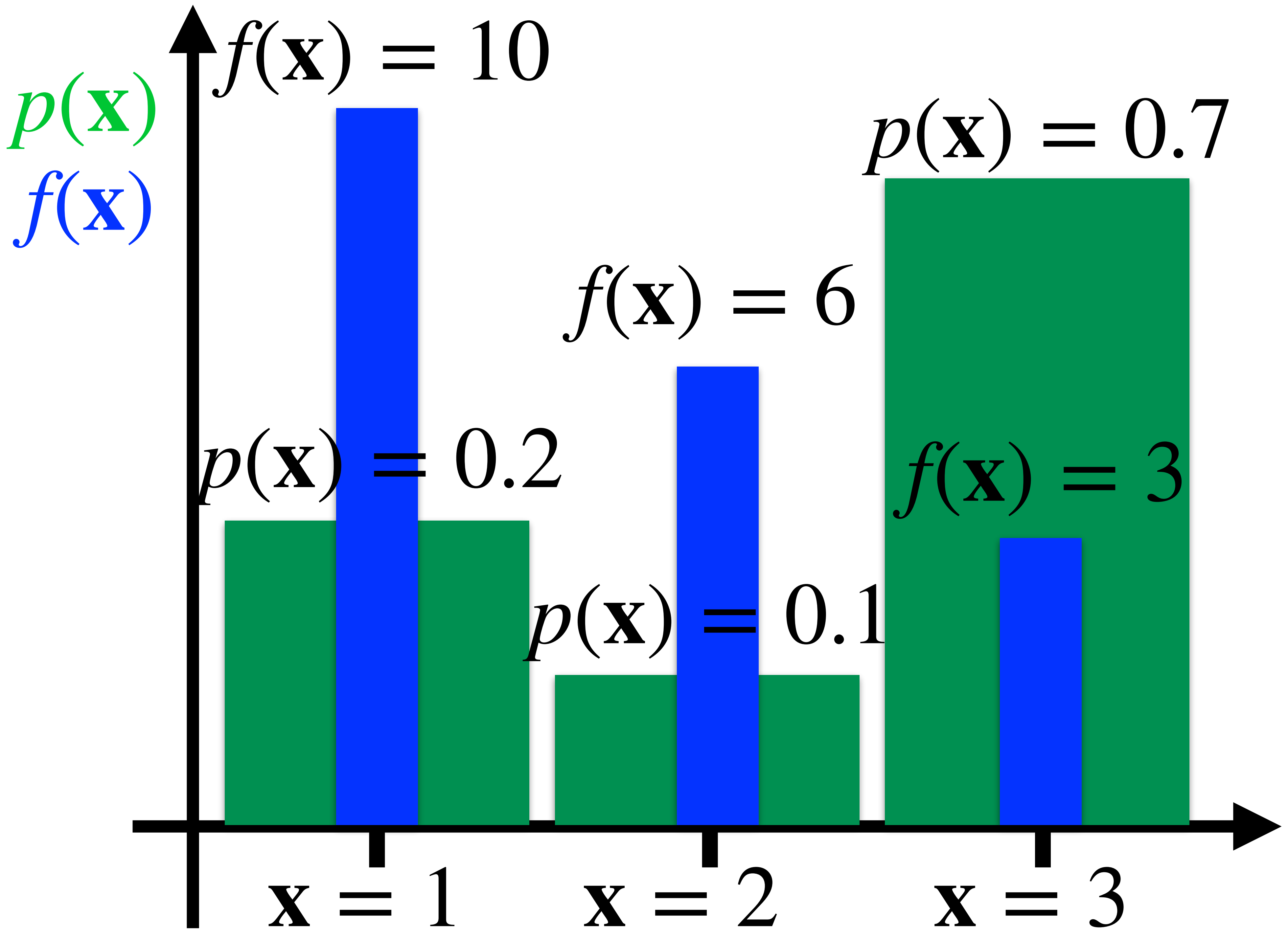
where $\mathbf{x}_i \sim p$



Prerequisites: Mean and average

$$\bar{f} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot f(\mathbf{x}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}\left[f(\mathbf{x})\right] = ??$$

where $\mathbf{x}_i \sim p$

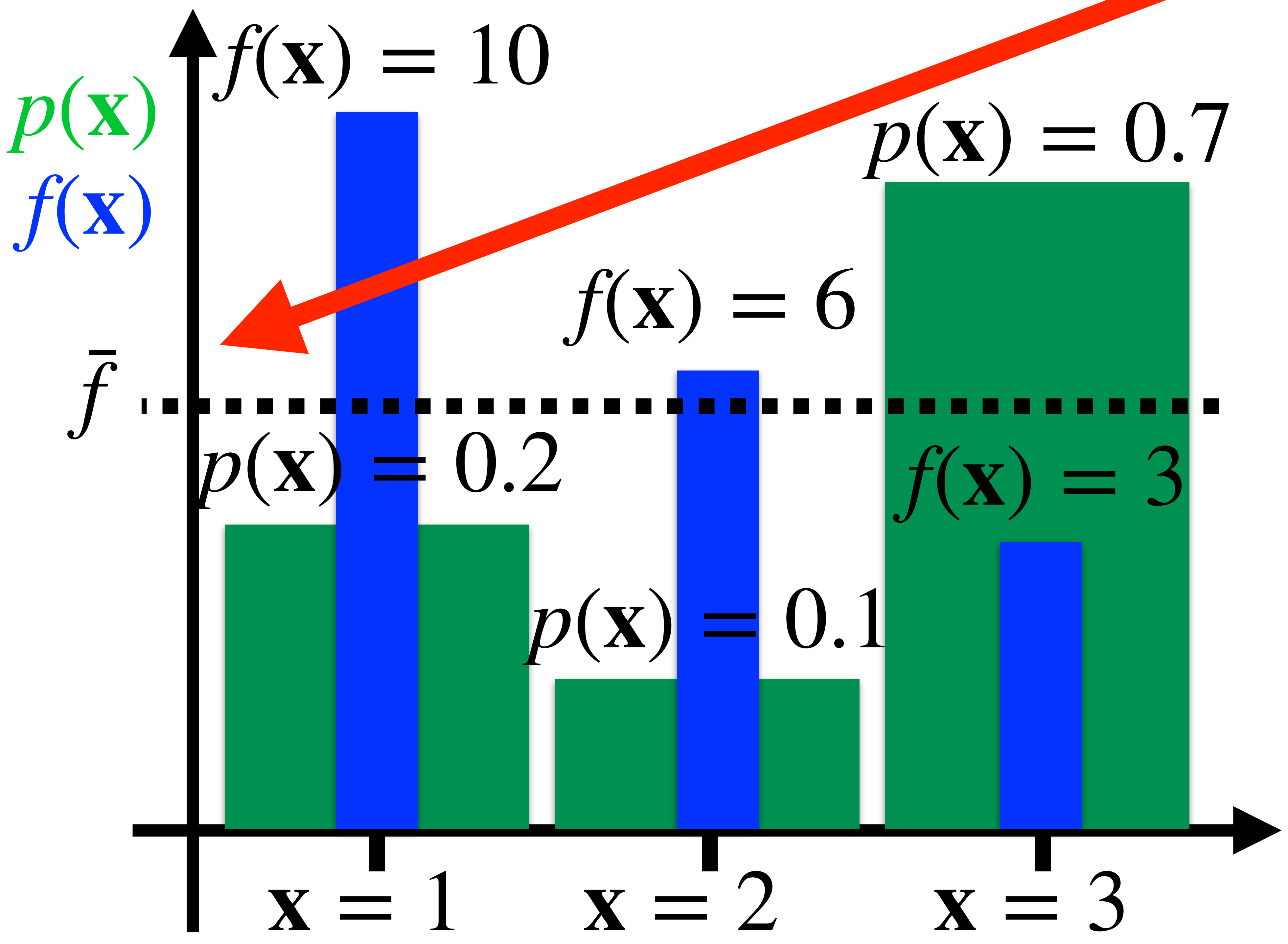


Prerequisites: Mean and average

$$\bar{f} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot f(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f(\mathbf{x})\right] = 0.2 \cdot 10 \ + \ 0.1 \cdot 6 \ + \ 0.7 \cdot 3 = 4.7$$

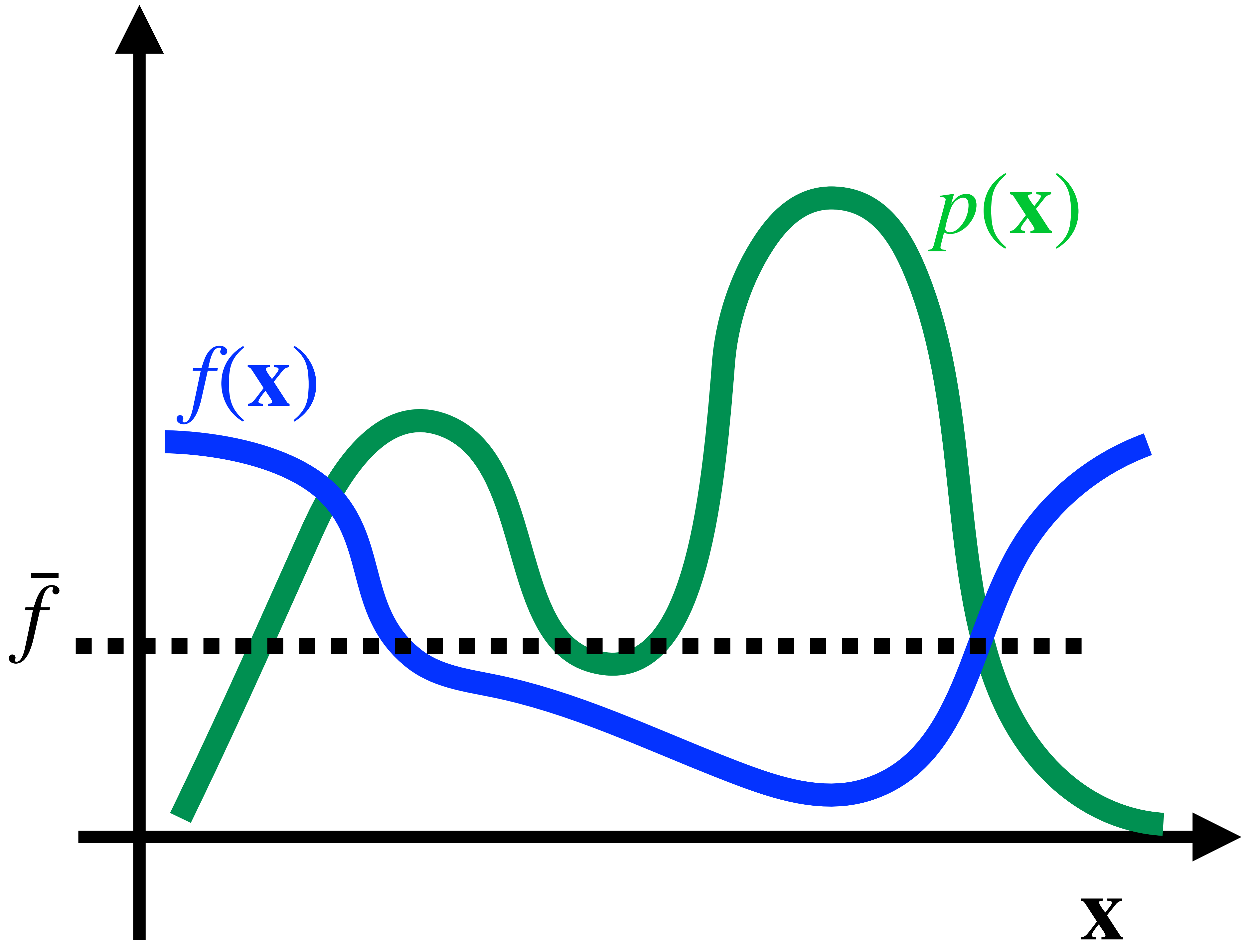
$$\approx \frac{1}{N}\sum_{i} f(\mathbf{x}_i) = \frac{1}{10}(10 + 10 + 6 + 3 + 3 + 3 + 3 + 3 + 3 + 3) = 4.7$$

where $\mathbf{x}_i \sim p$



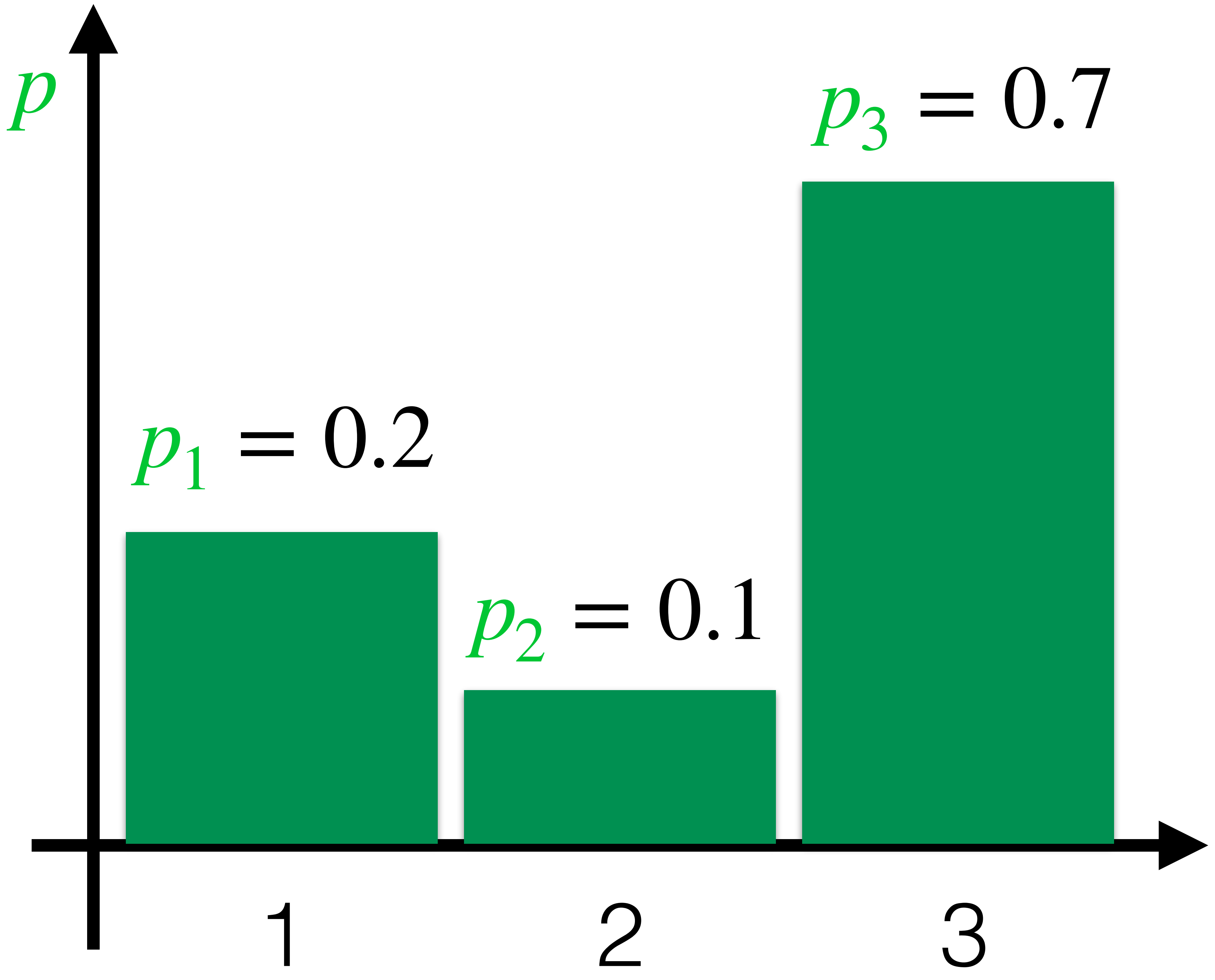
Prerequisites: Mean and average

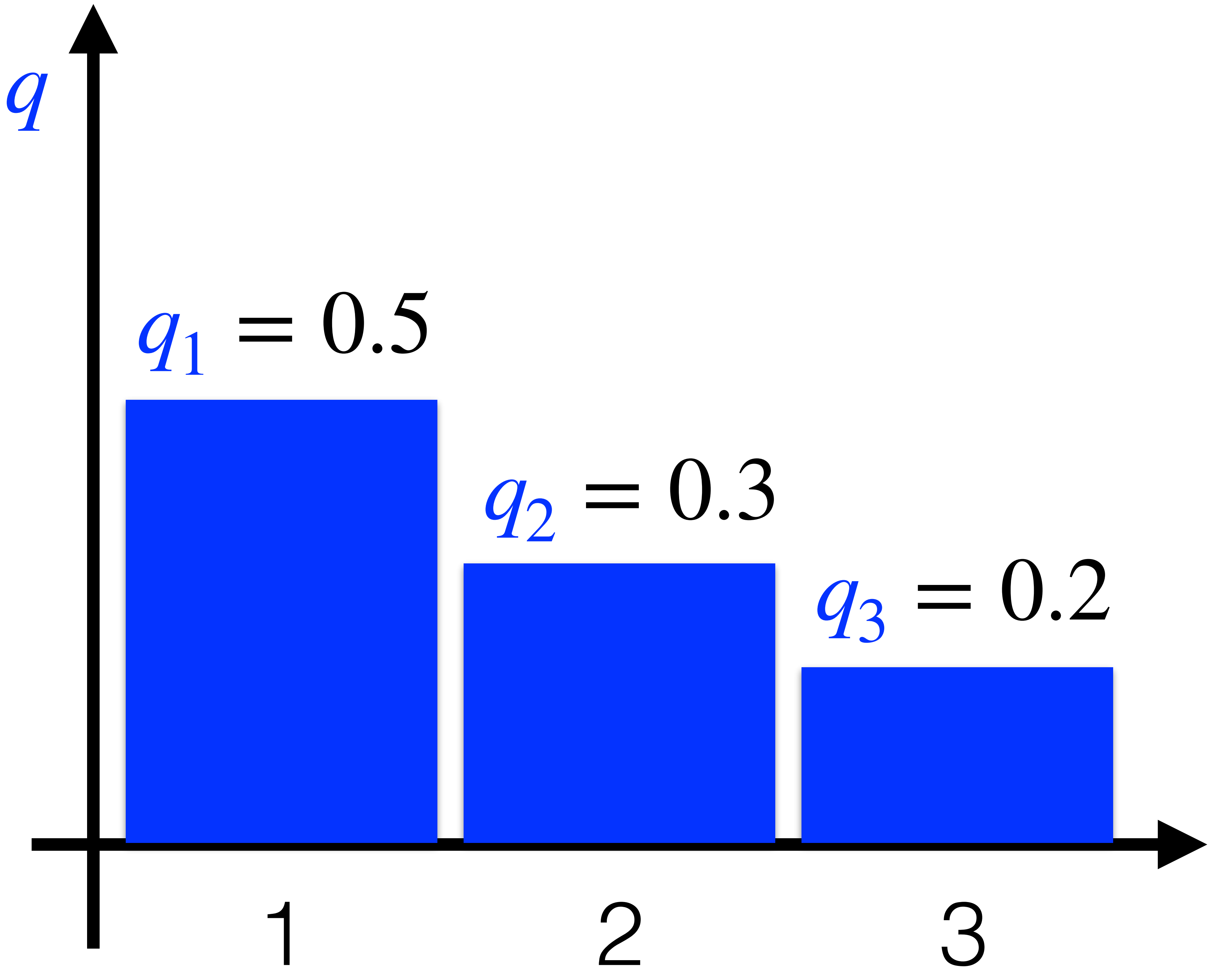
$$\bar{f} = \int_{\mathbf{x}} p(\mathbf{x}) \cdot f(\mathbf{x})\, d\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f(\mathbf{x})\right] \approx \frac{1}{N}\sum_i f(\mathbf{x}_i) \qquad \text{where } \mathbf{x}_i \sim p$$



Prerequisites: KL-divergence

$$D_{KL}(p \parallel q) = \sum_i p_i \cdot \log \frac{p_i}{q_i} = 0.2 \cdot \log \frac{0.2}{0.5} + 0.1 \cdot \log \frac{0.1}{0.3} + 0.7 \cdot \log \frac{0.7}{0.2} = 0.2535$$



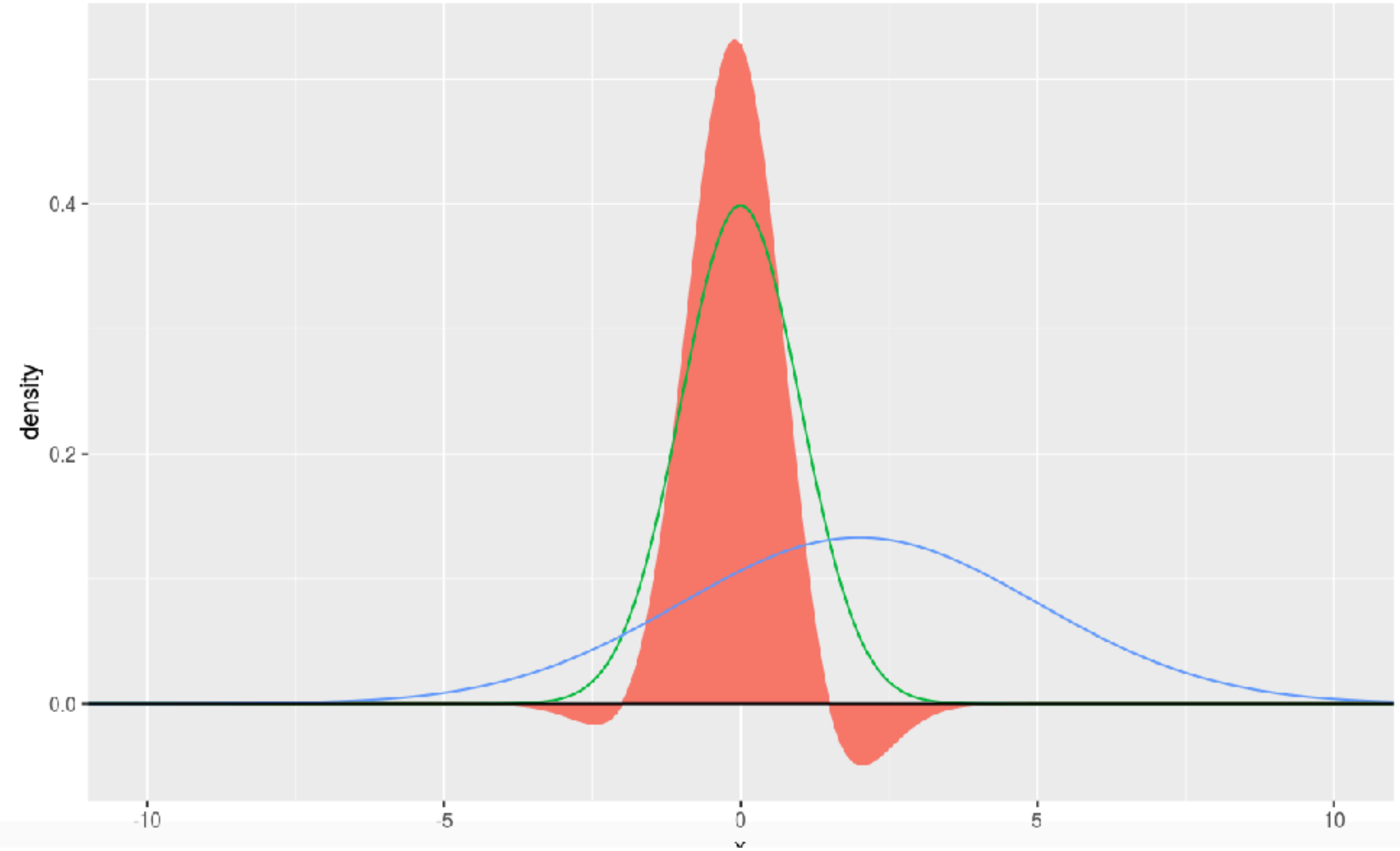


- What if $q_k = 0$ and $p_k > 0$?
- What if $q_k = p_k$ for k=1,2,3 ?
- Is it symmetrical $D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$?

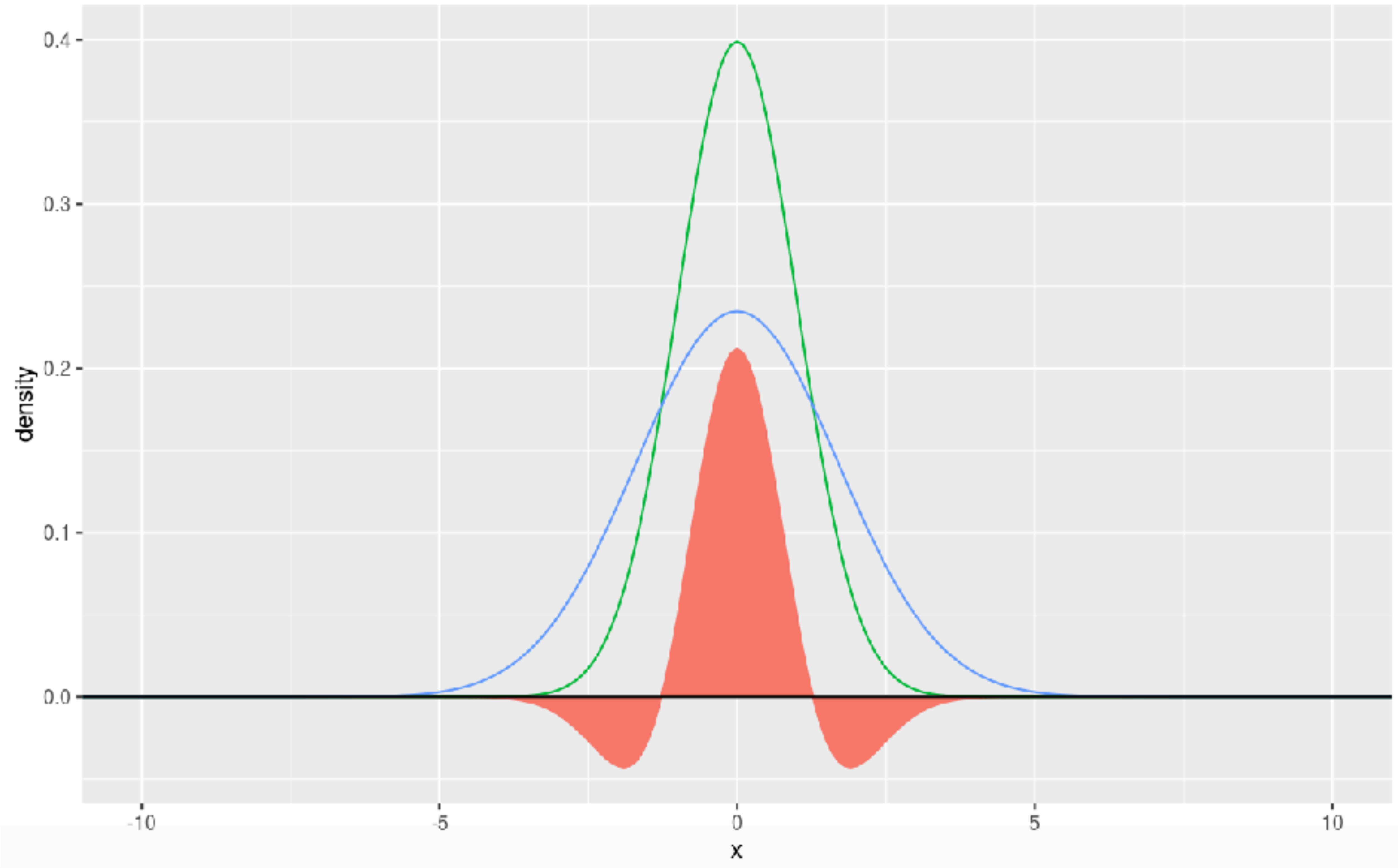
Prerequisites: KL-divergence

$$D_{KL}\left(p(\mathbf{x}) \parallel q(\mathbf{x})\right) = \int_{\mathbf{x}} p(\mathbf{x}) \cdot \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \, d\mathbf{x}$$

$D_{KL}\left(p(\mathbf{x}) \parallel q(\mathbf{x})\right)$ =0.8764



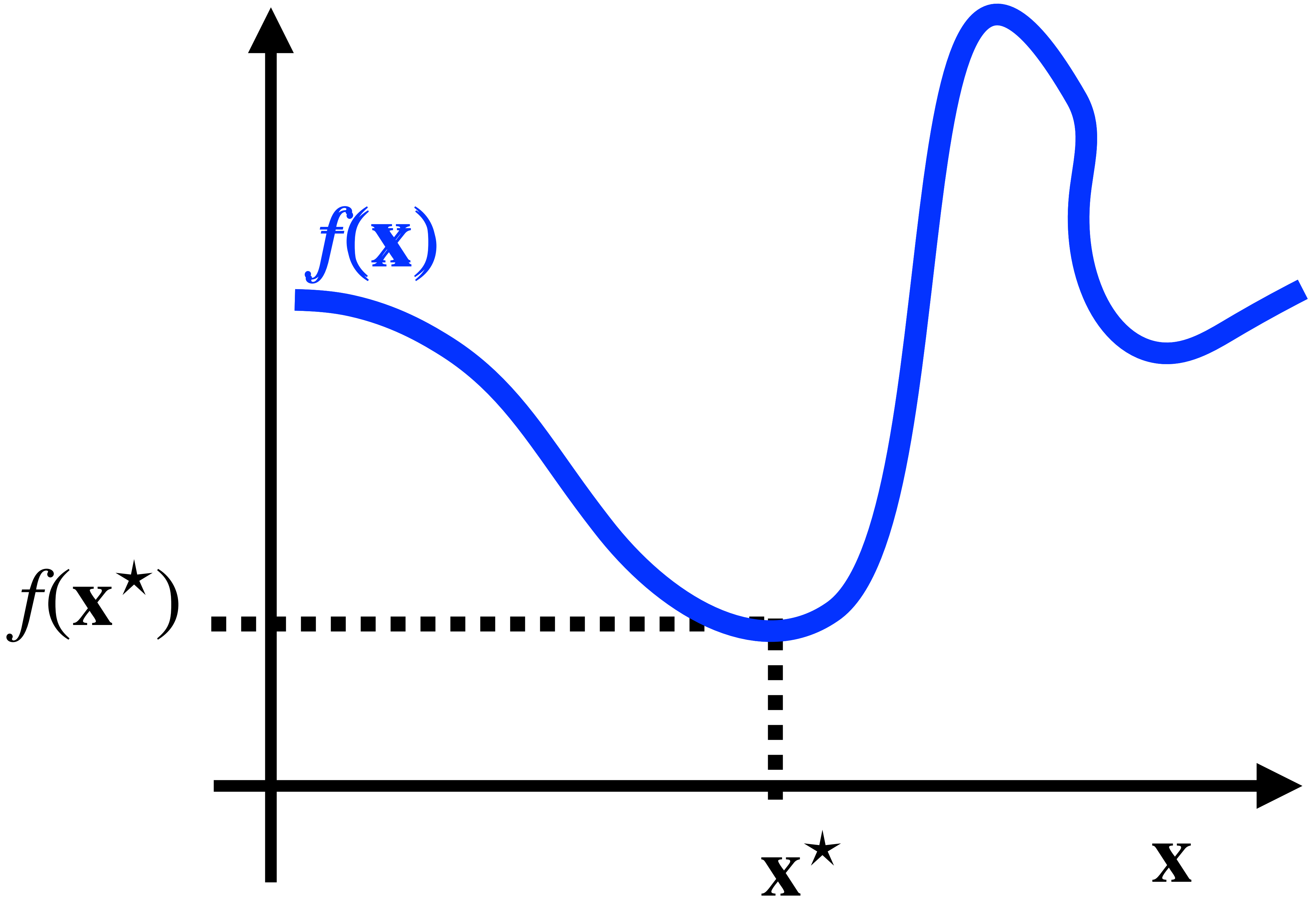
$D_{KL}\left(p(\mathbf{x}) \parallel q(\mathbf{x})\right)$ =0.2036



https://gnarlyware.com/blog/kl-divergence-online-demo/

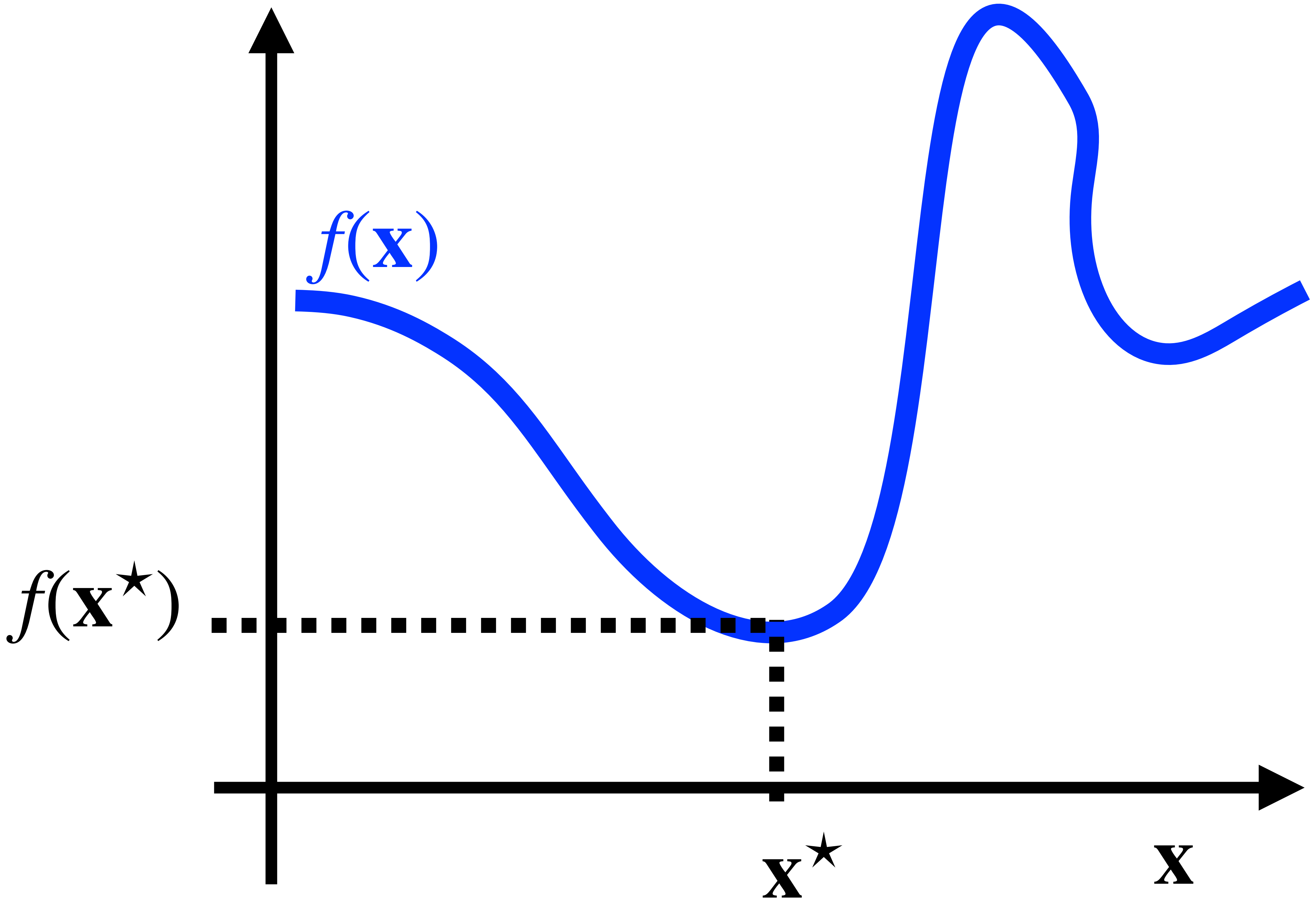
Prerequisites: argmin

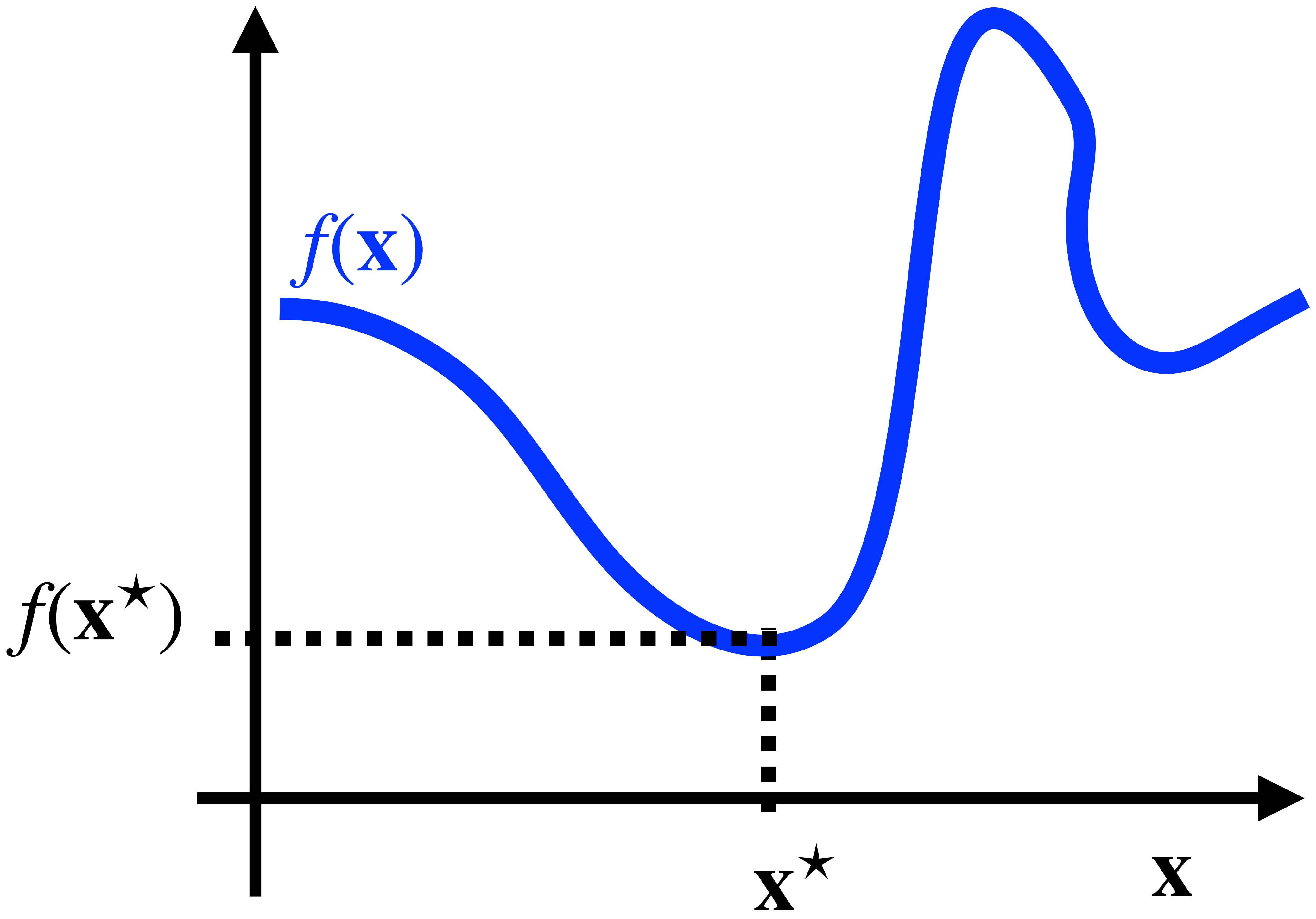
$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \log\big(f(\mathbf{x})\big)$$



Prerequisites: argmin

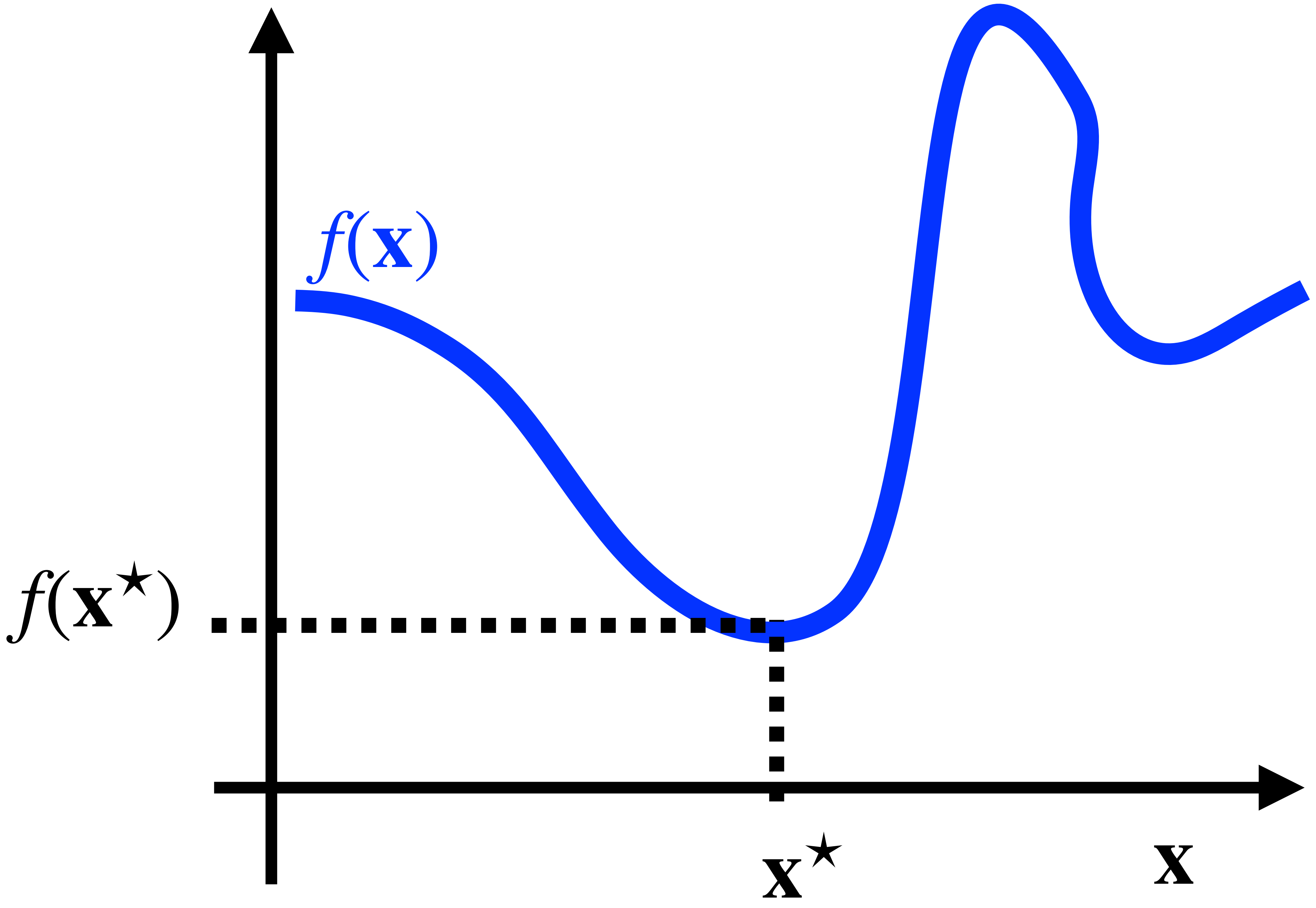
$$\mathbf{x}^{\star} = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \log\big(f(\mathbf{x})\big)$$

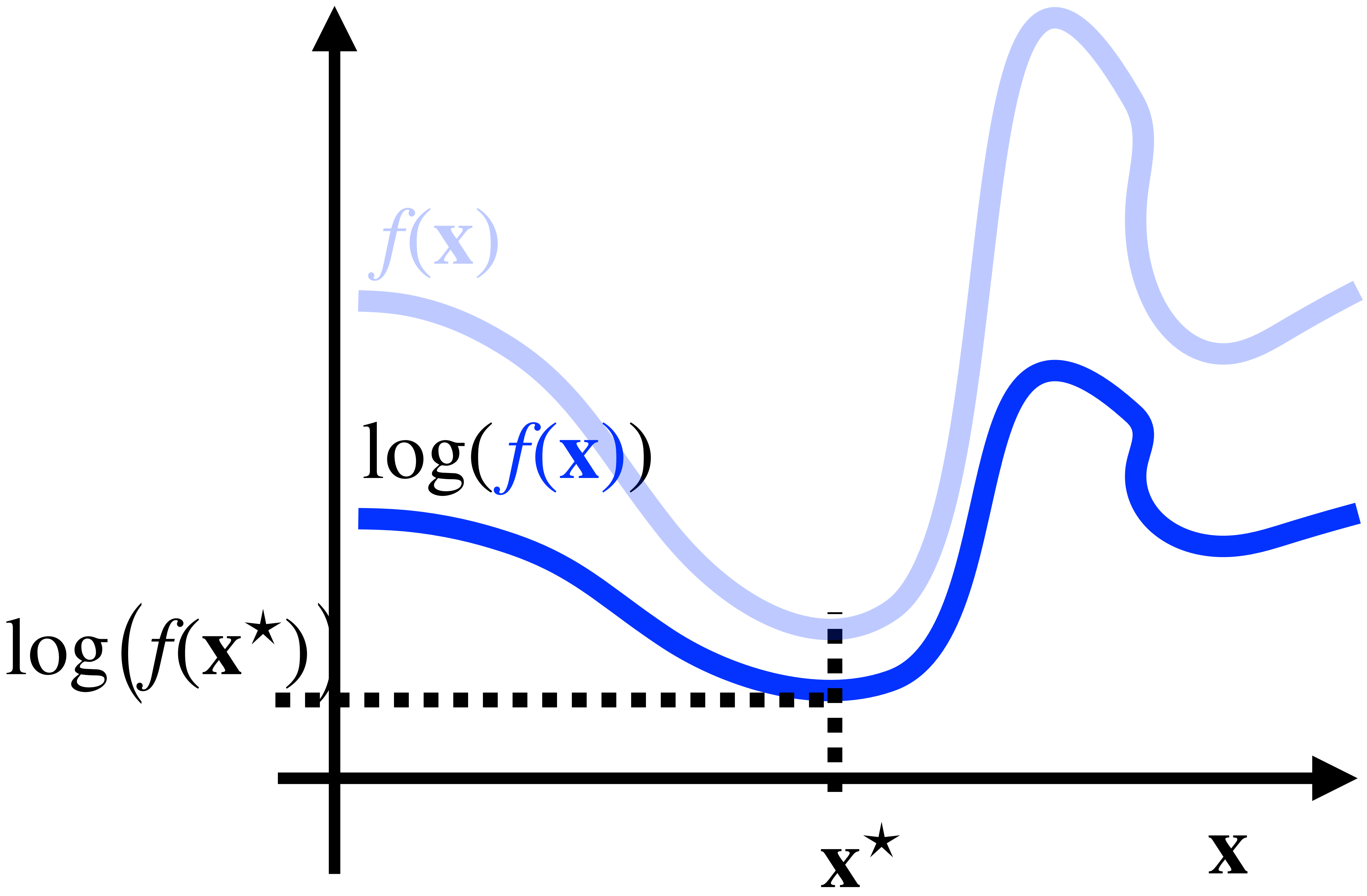




Prerequisites: argmin

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \log\big(f(\mathbf{x})\big)$$



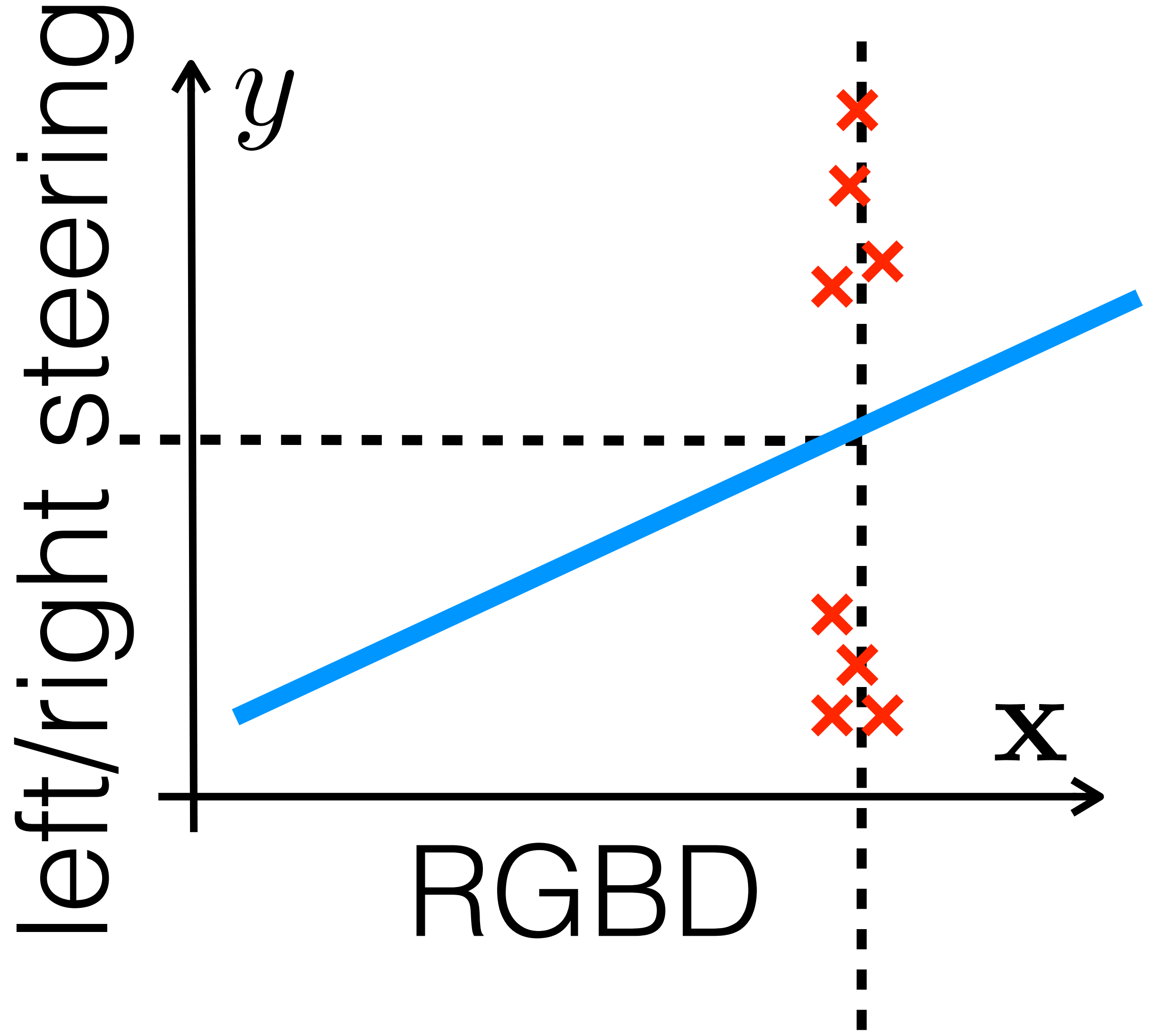


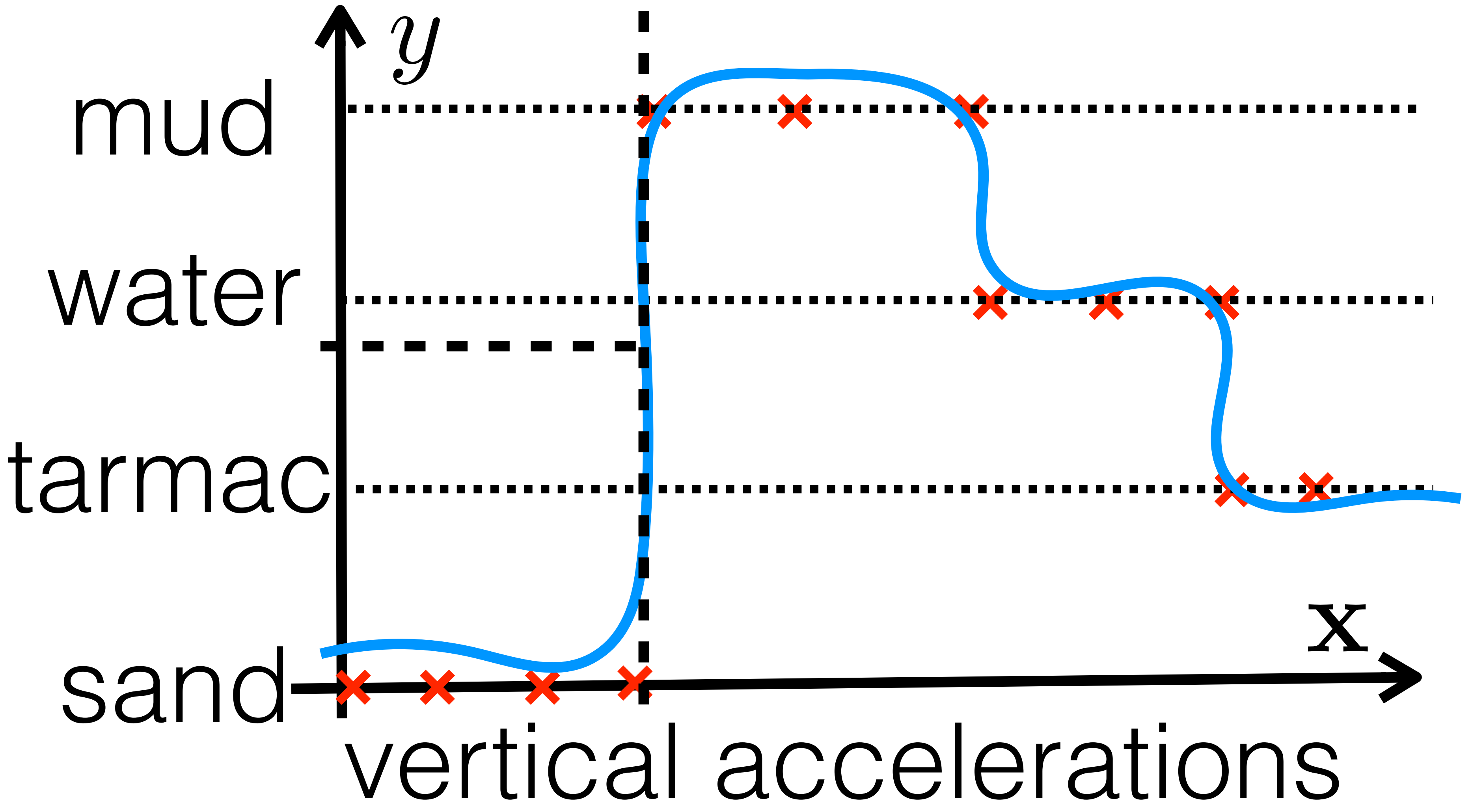
Where does the **loss function** come from?

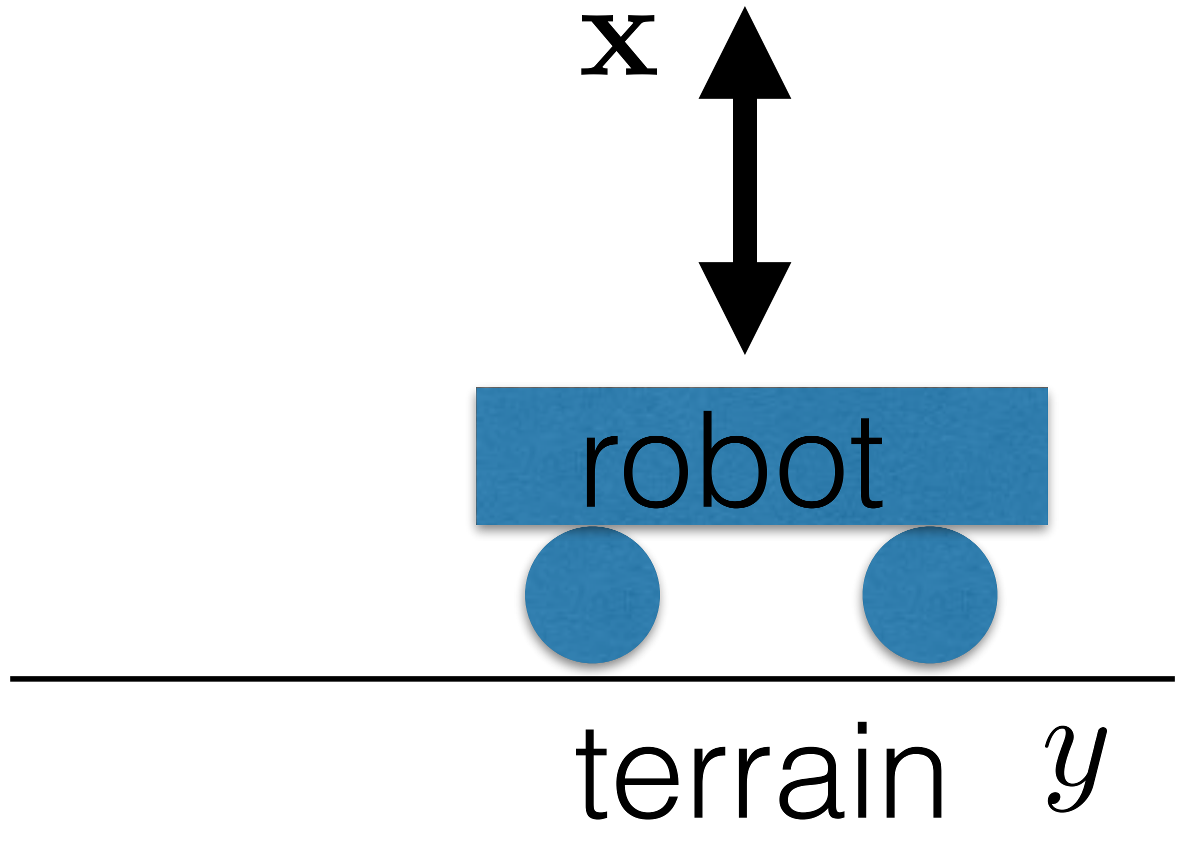
Lec 01: what can go wrong: **inappropriate choice of loss function**

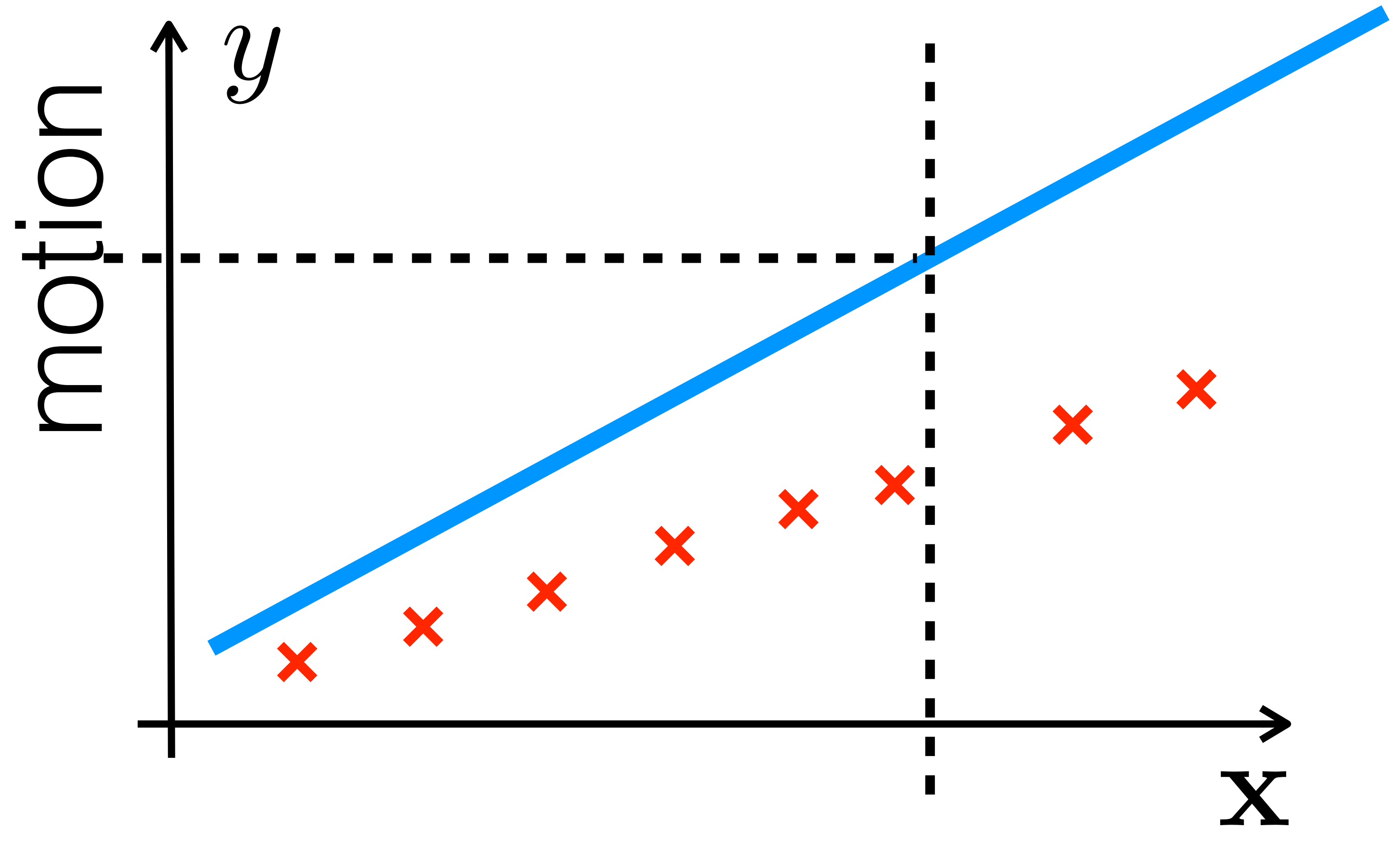
What should I do instead of fitting a curve???

Search for probability distribution of y given x

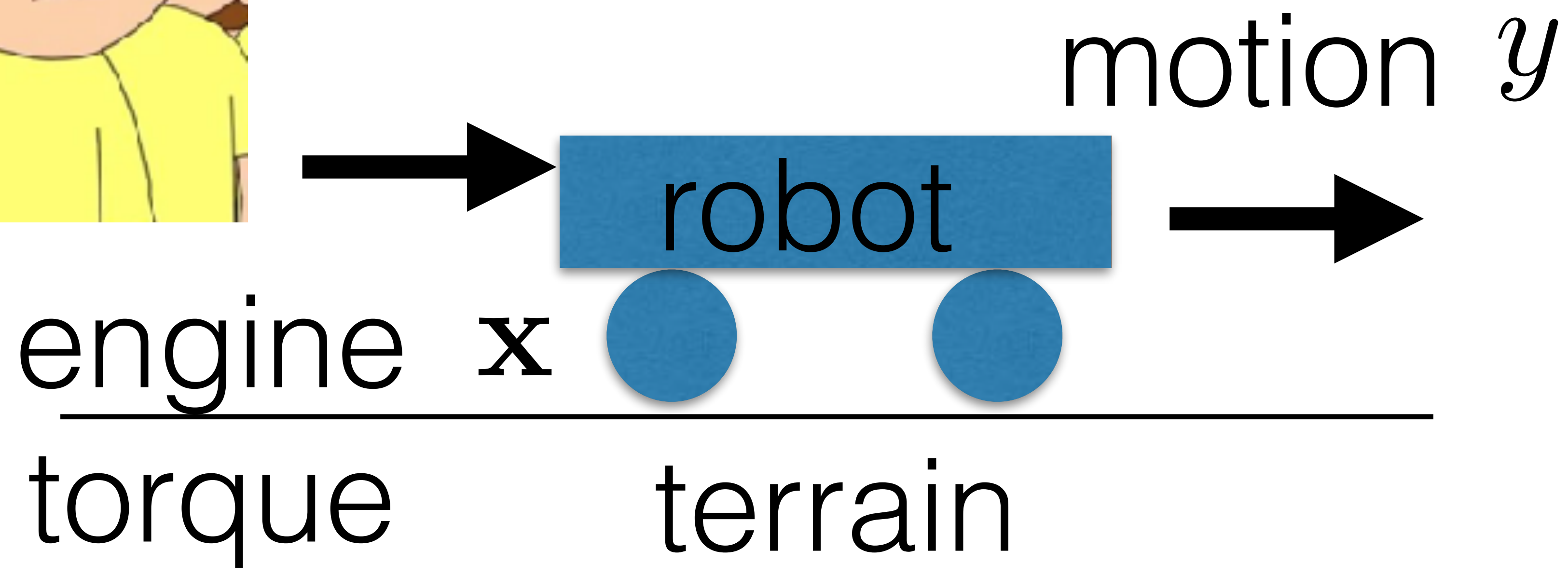






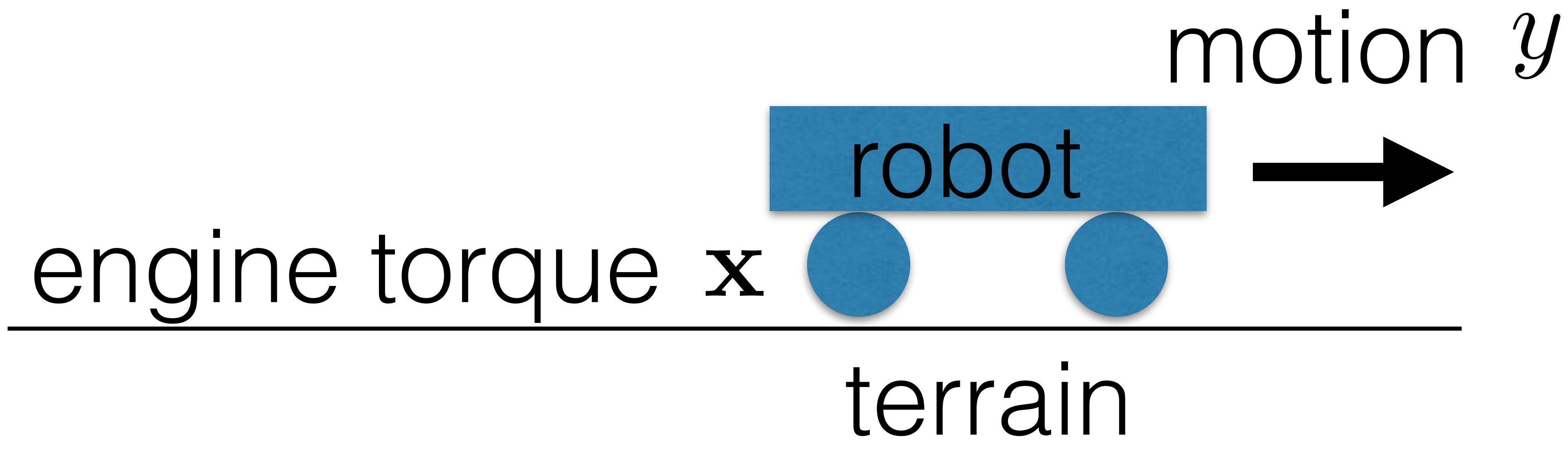


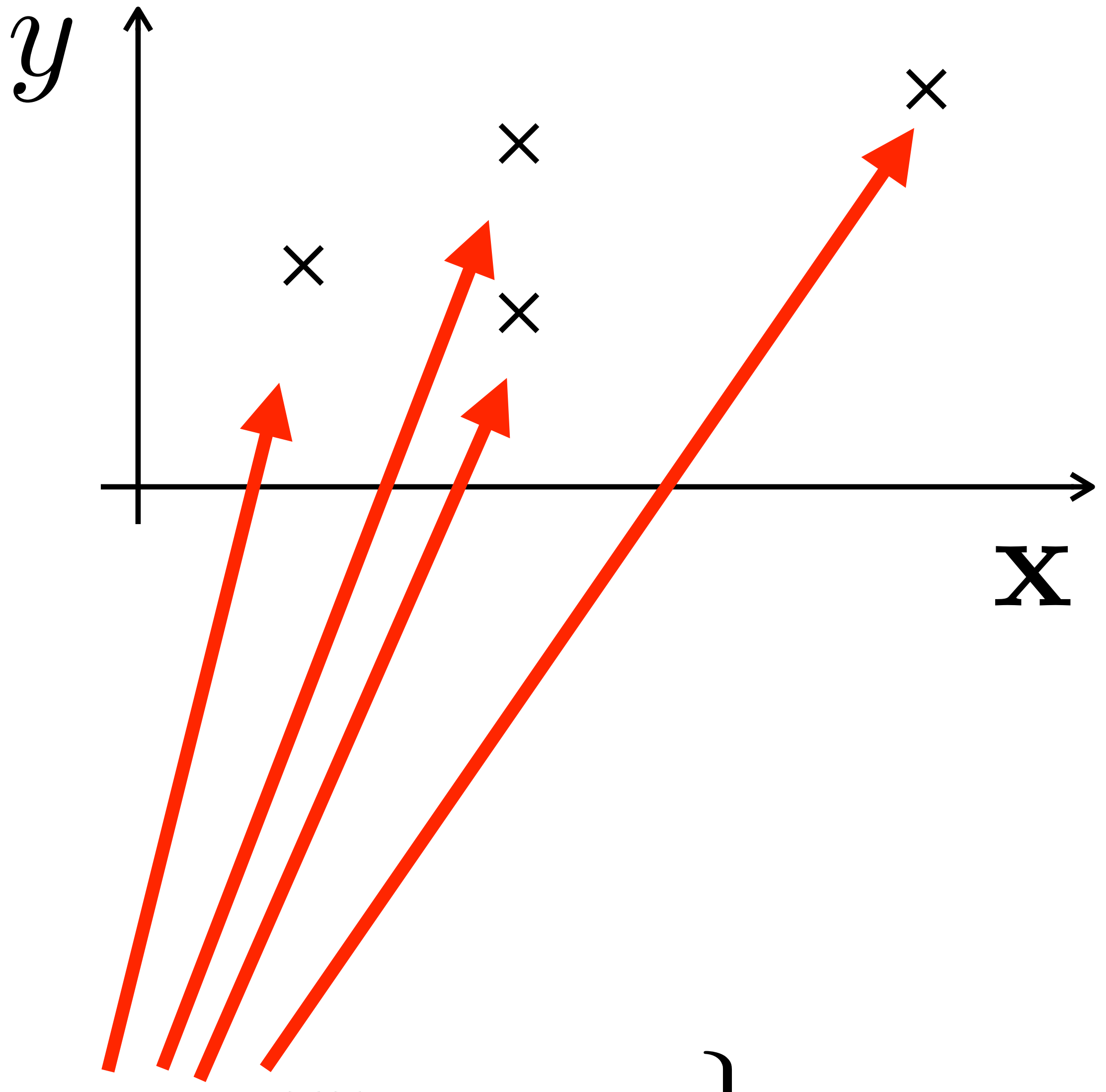
Nature is evil !!!



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

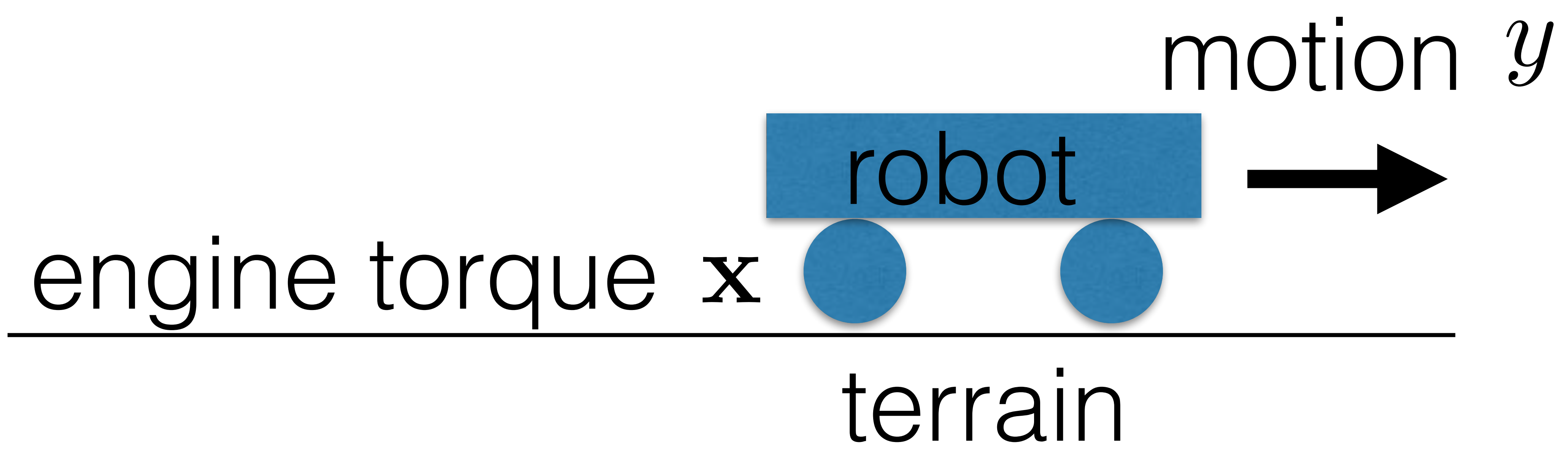
Motivation example: estimation of a motion model

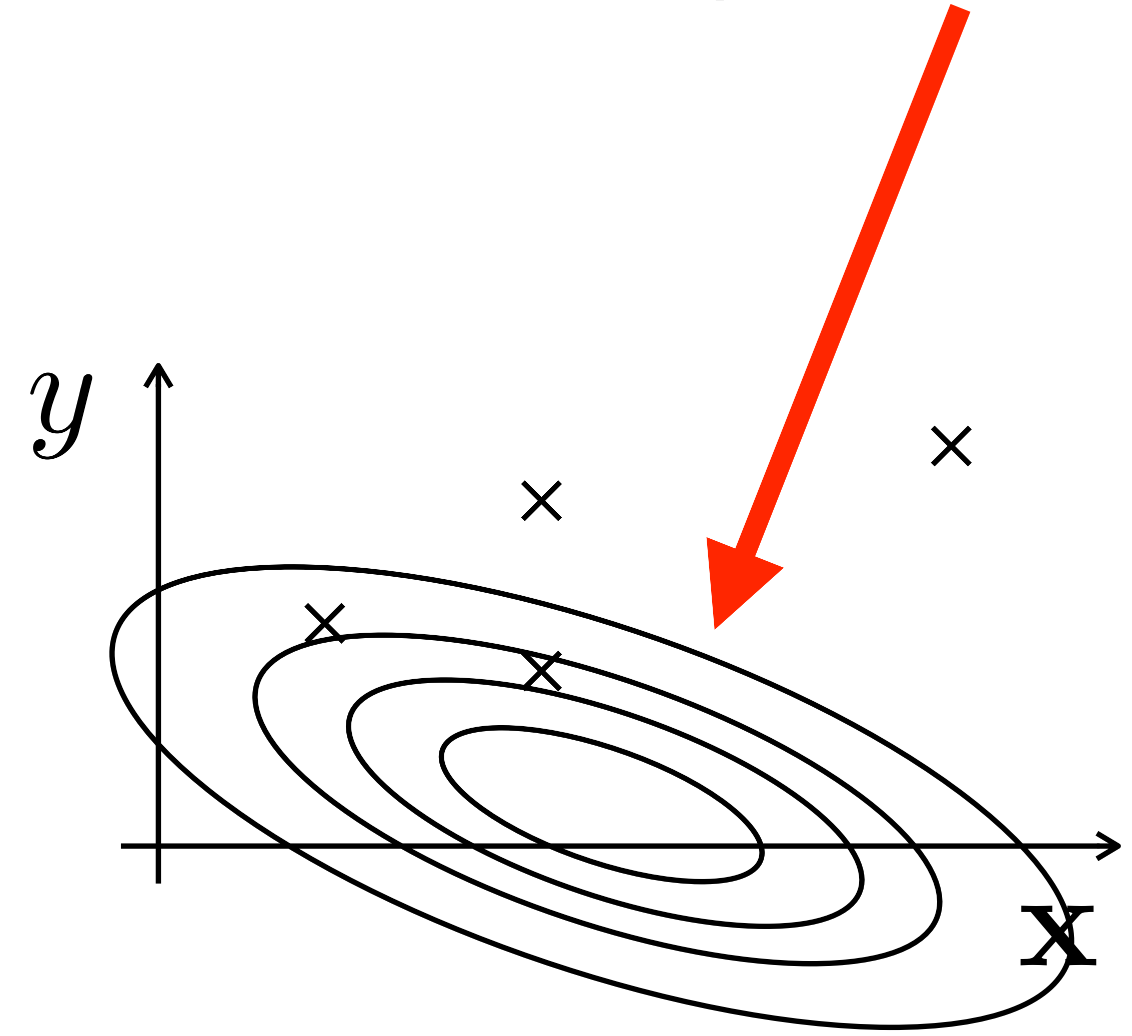




$$\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$$

Motivation example: estimation of a motion model

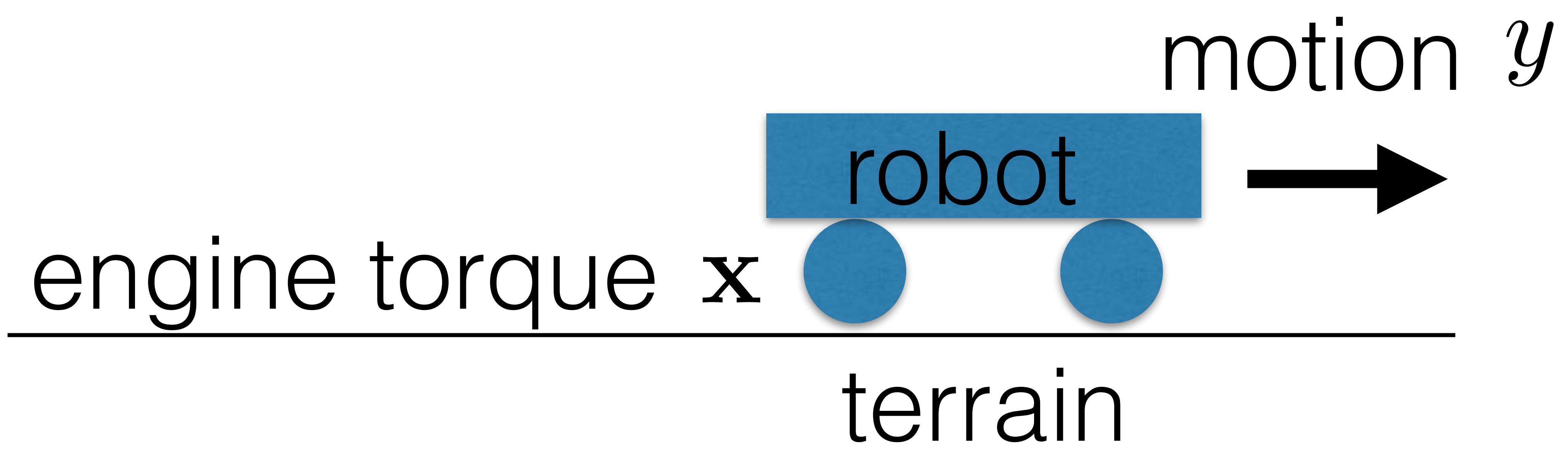




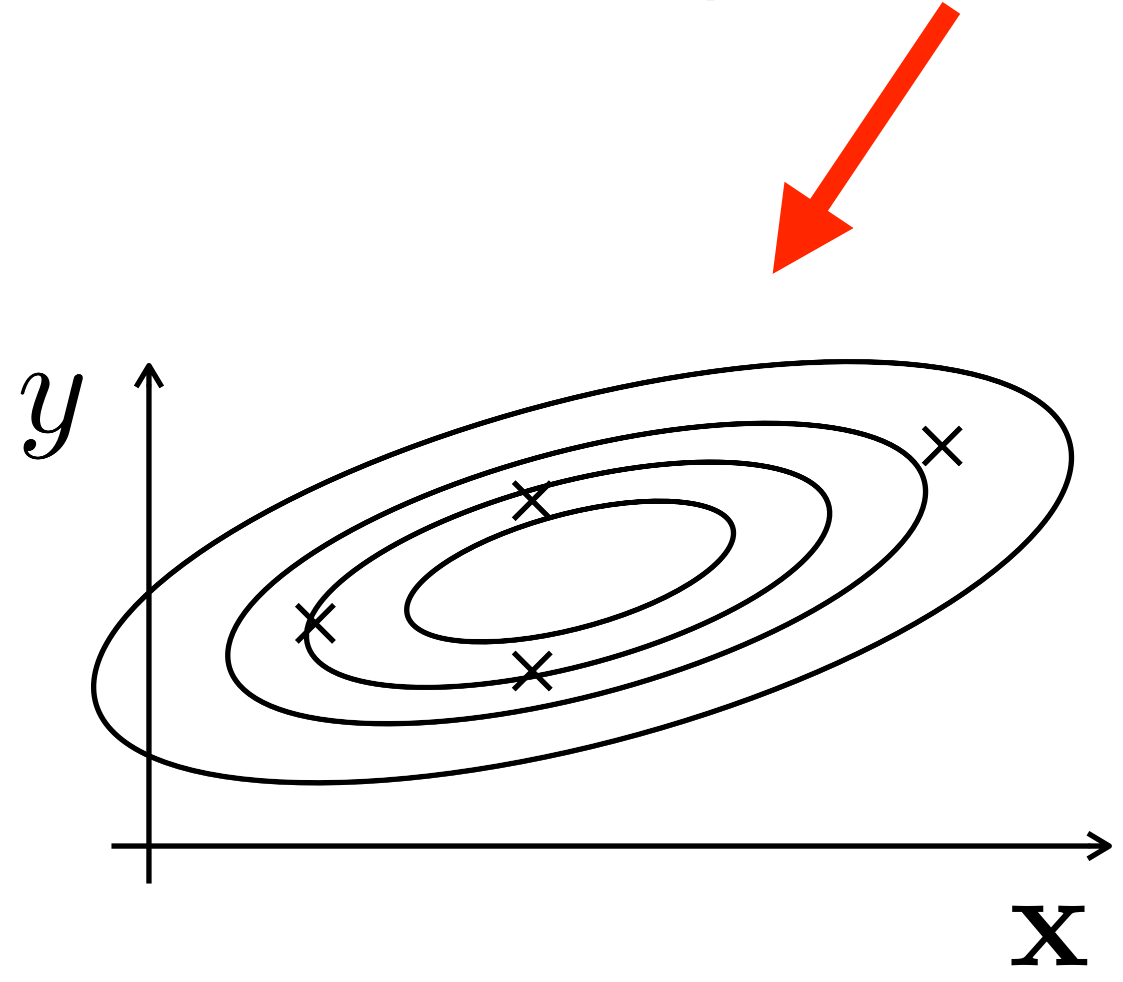
Unknown distribution with parameters **w**

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

Motivation example: estimation of a motion model



Unknown distribution with parameters **w**



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

How should we fit distribution into data?

How should we fit distribution into data?

- Many robotics tasks contain perception problems: given $\mathbf{x}$ estimate y

$\mathbf{x}$

y

🟪 road

🟪 sideway

🟥 pedestrian

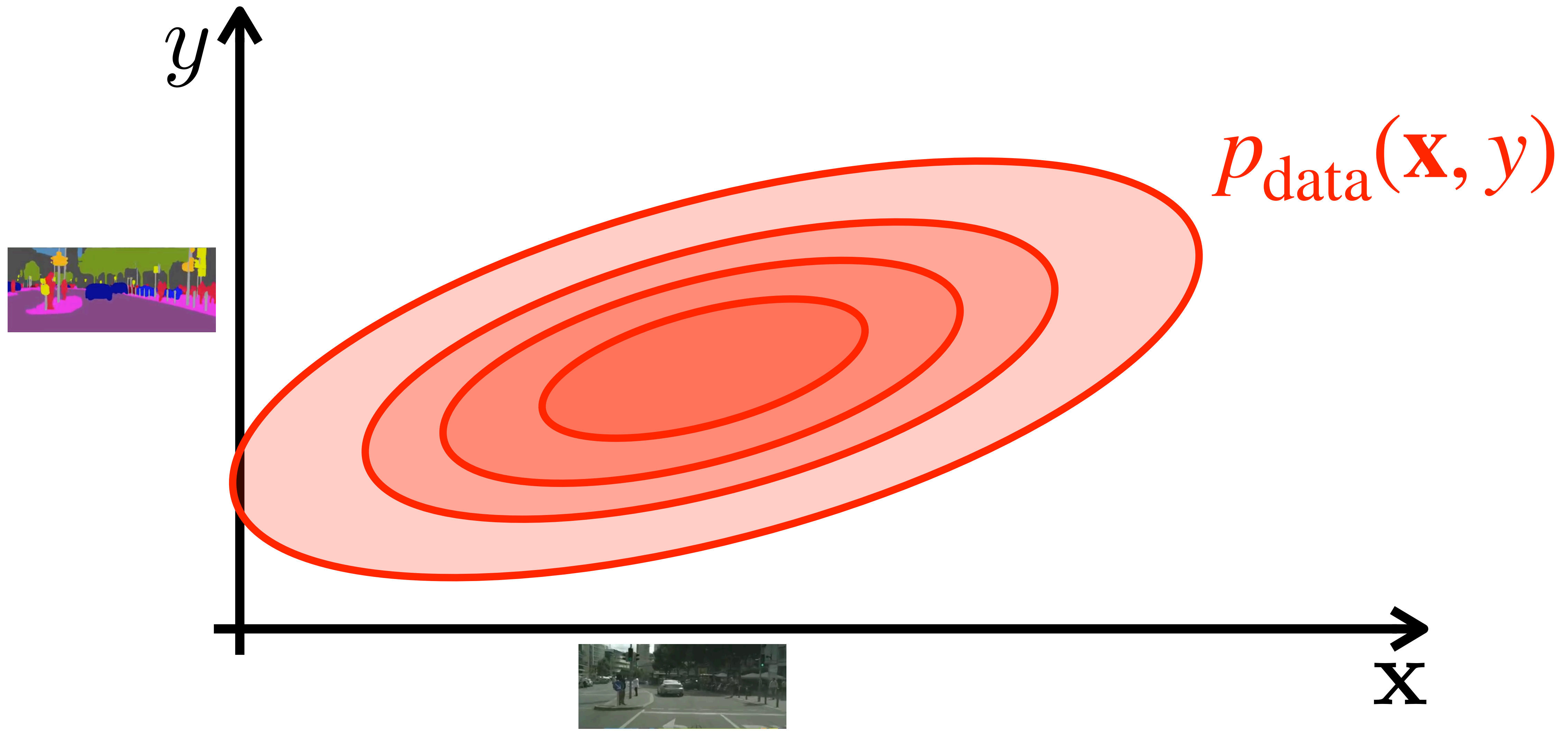
🟨 traffic sign

🟩 trees

🟦 sky

Why should I minimize -log(p)?

- $(\mathbf{x}, y)$-tuples live on unknown distribution $p_{\text{data}}(\mathbf{x}, y)$



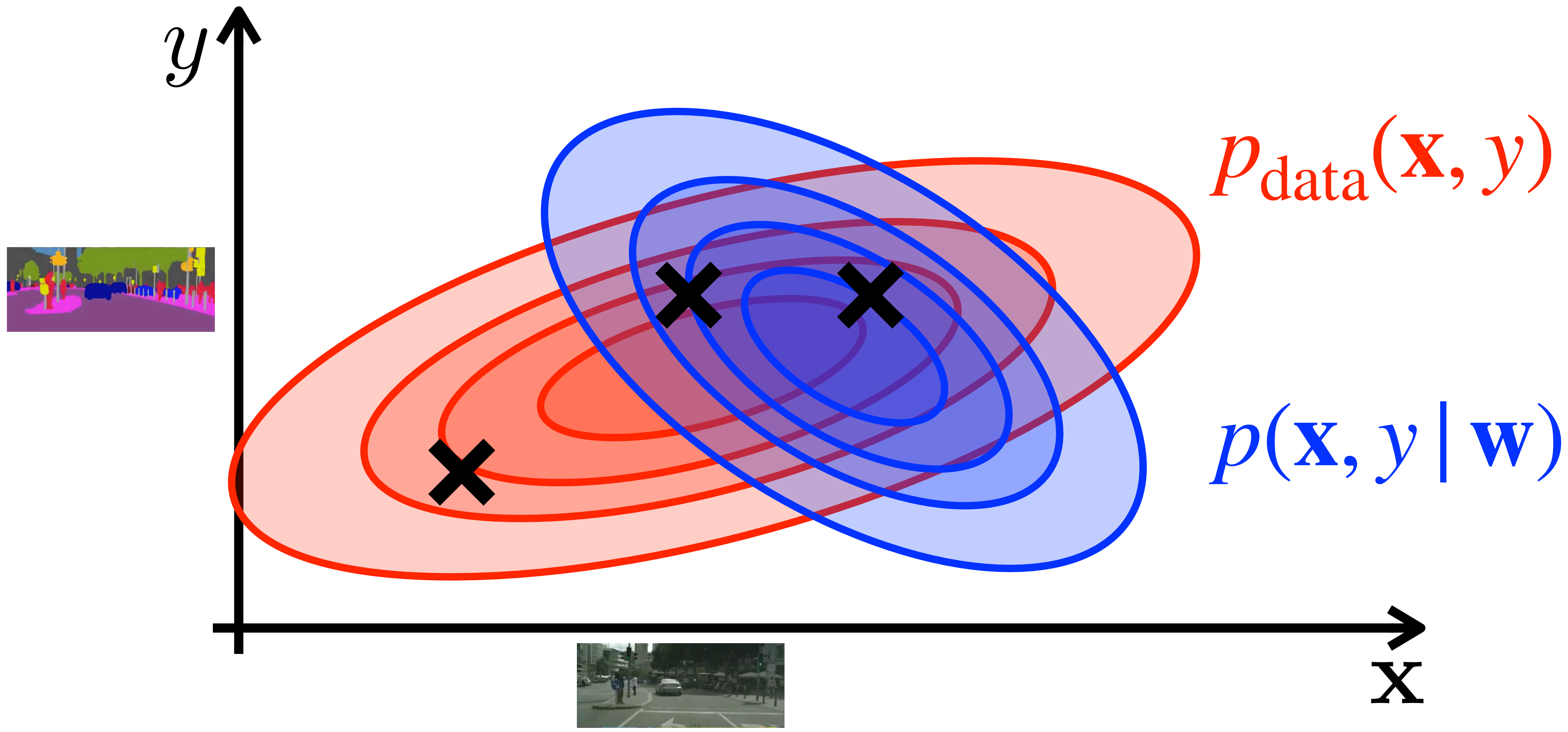
Why should I minimize -log(p)?

- $(\mathbf{x}, y)$-tuples live on unknown distribution $p_{\text{data}}(\mathbf{x}, y)$
- We approximate it by $p(\mathbf{x}, y \,|\, \mathbf{w})$
- We search for weigths $\mathbf{w}$ that makes $p(\mathbf{x}, y \,|\, \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^{\star} = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y \,|\, \mathbf{w})\big) = \arg\min_{\mathbf{w}} \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}(\mathbf{x},y)}\left[\log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}\right] = \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)}\big[\cancel{\log p_{\text{data}}(\mathbf{x}, y)} - \log p(y \,|\, \mathbf{x}, \mathbf{w})\cancel{p(\mathbf{x})}\big]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)}\big[-\log p(y \,|\, \mathbf{w}, \mathbf{x})\big] \approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i,y_i)\,\sim\, p_{\text{data}}(\mathbf{x},y)} \big[-\log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})\big]$$ …KL justification



MLE:

$$= \arg\min_{\mathbf{w}} -\log \prod_{(\mathbf{x}_i,y_i)} p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$$

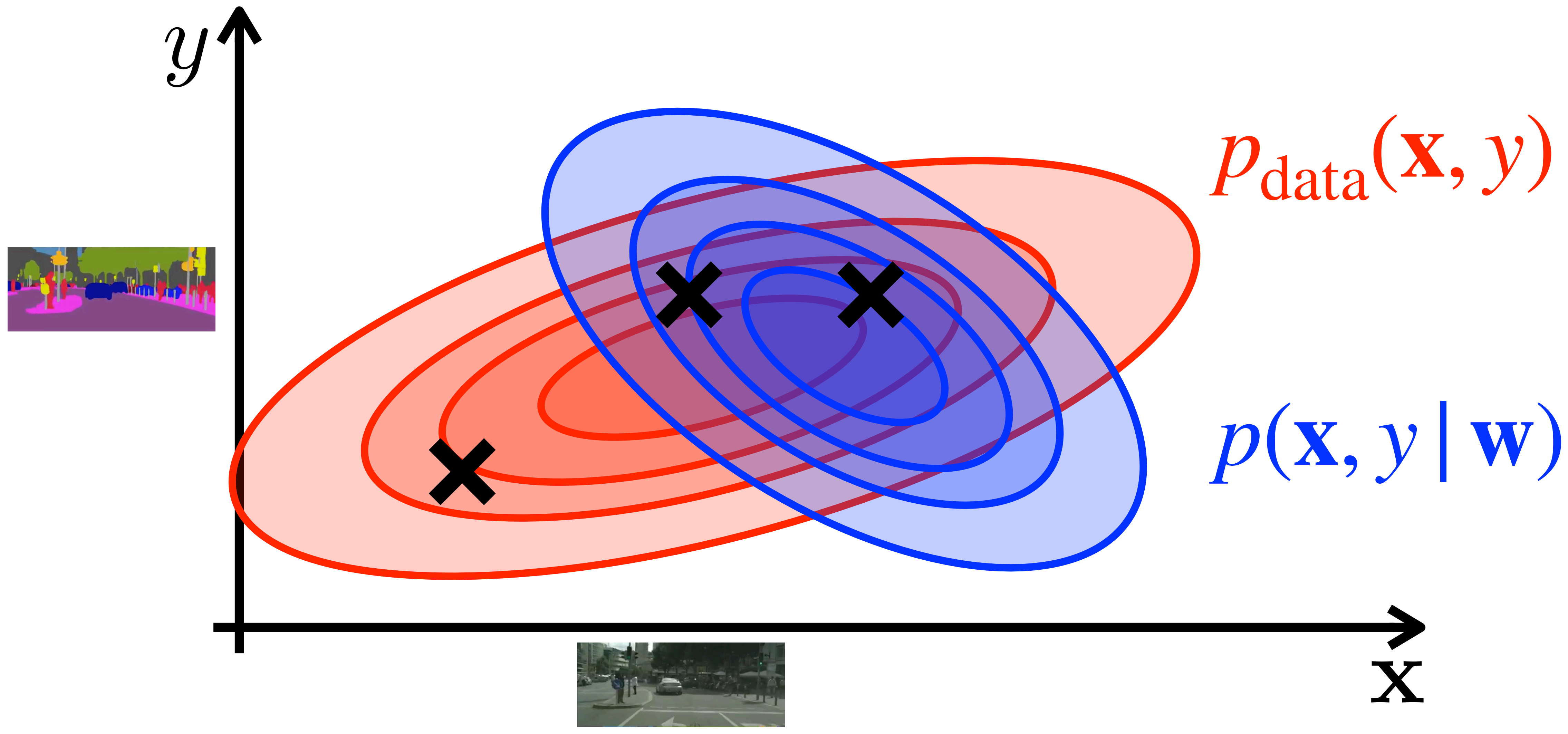
Why should I minimize -log(p)?

- $(\mathbf{x}, y)$-tuples live on unknown distribution $p_{\text{data}}(\mathbf{x}, y)$
- We approximate it by $p(\mathbf{x}, y \,|\, \mathbf{w})$
- We search for weigths $\mathbf{w}$ that makes $p(\mathbf{x}, y \,|\, \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^{\star} = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y \,|\, \mathbf{w})\big) = \arg\min_{\mathbf{w}} \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}(\mathbf{x},y)}\left[\log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}\right] = \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)}\left[\cancel{\log p_{\text{data}}(\mathbf{x}, y)} - \log p(y \,|\, \mathbf{x}, \mathbf{w}) \cancel{p(\mathbf{x})}\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)}\left[-\log p(y \,|\, \mathbf{w}, \mathbf{x})\right] \approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i,y_i) \sim p_{\text{data}}(\mathbf{x}, y)} \left[-\log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})\right] \text{…KL justification}$$



MLE:

$$= \arg\min_{\mathbf{w}} -\log \prod_{(\mathbf{x}_i,y_i)} p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$$

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

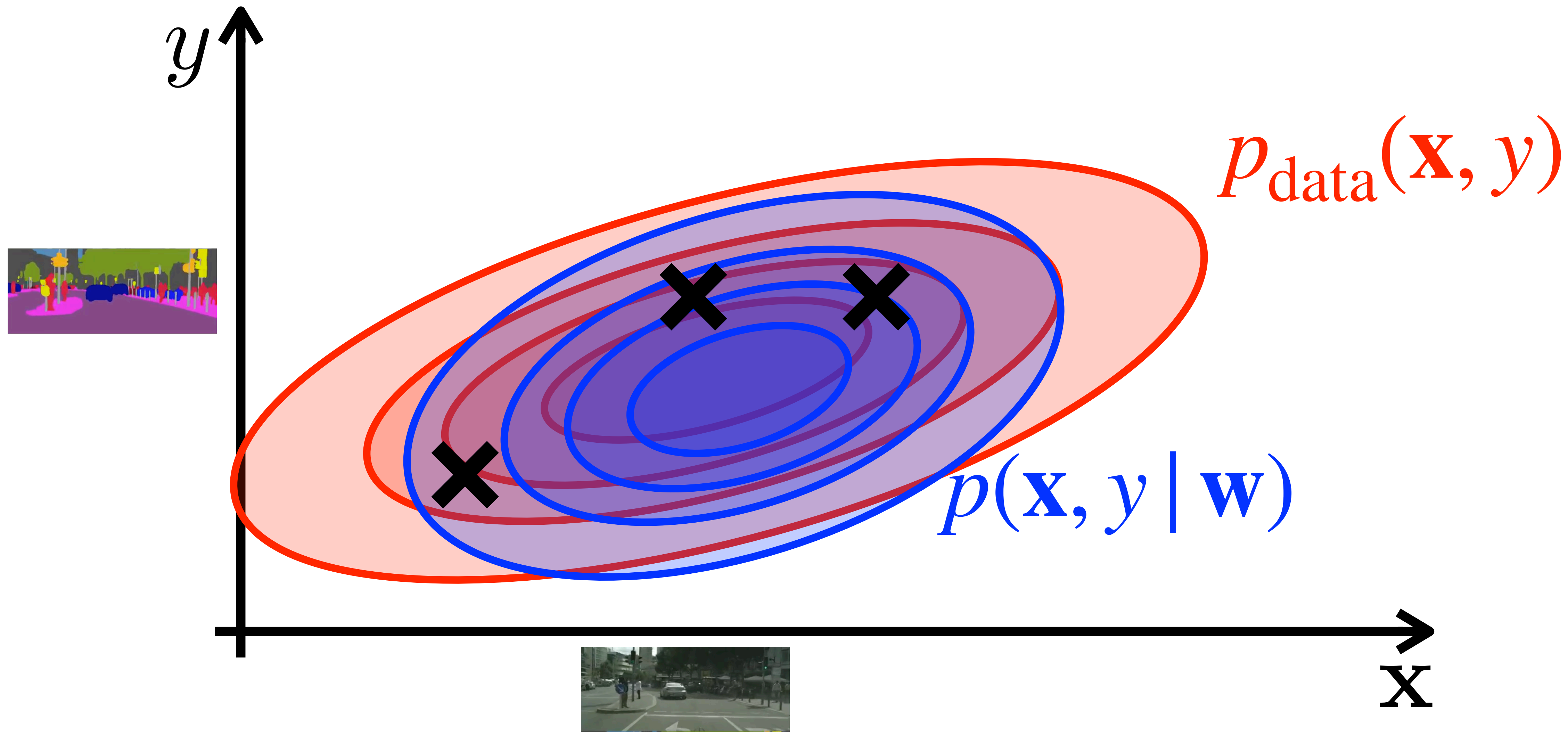
Why should I minimize -log(p)?

- $(\mathbf{x}, y)$-tuples live on unknown distribution $p_{\text{data}}(\mathbf{x}, y)$
- We approximate it by $p(\mathbf{x}, y \,|\, \mathbf{w})$
- We search for weigths $\mathbf{w}$ that makes $p(\mathbf{x}, y \,|\, \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) = \arg\min_{\mathbf{w}} \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}(\mathbf{x},y)}\left[\log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}\right] = \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)}[\cancel{\log p_{\text{data}}(\mathbf{x}, y)} - \log p(y \,|\, \mathbf{x}, \mathbf{w})\cancel{p(\mathbf{x})}]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)}\big[-\log p(y \,|\, \mathbf{w}, \mathbf{x})\big] \approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i,y_i)\,\sim\, p_{\text{data}}(\mathbf{x}, y)} \big[-\log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})\big] \ldots \text{KL justification}$$

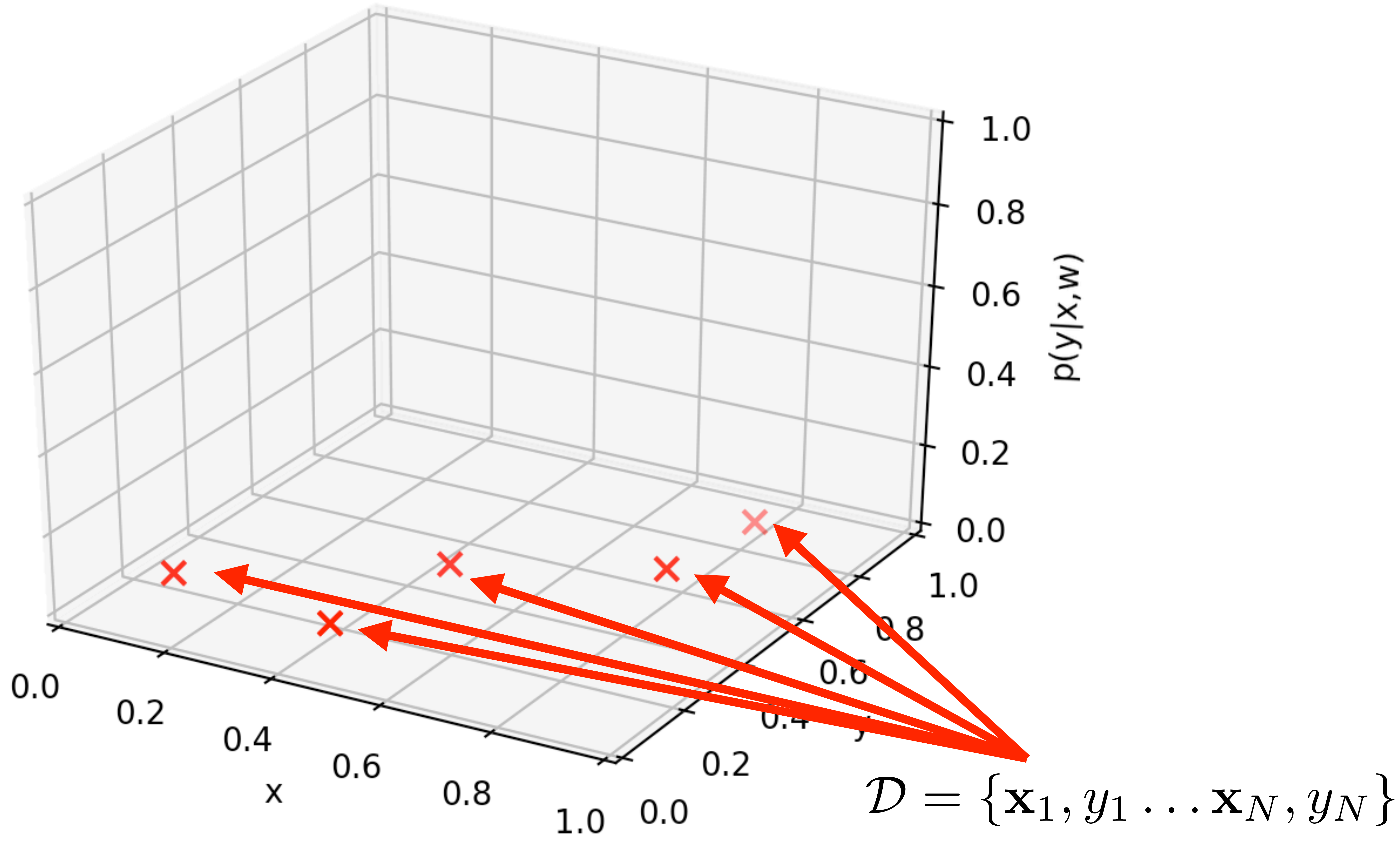


MLE:

$$= \arg\min_{\mathbf{w}} -\log \prod_{(\mathbf{x}_i,y_i)} p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$$

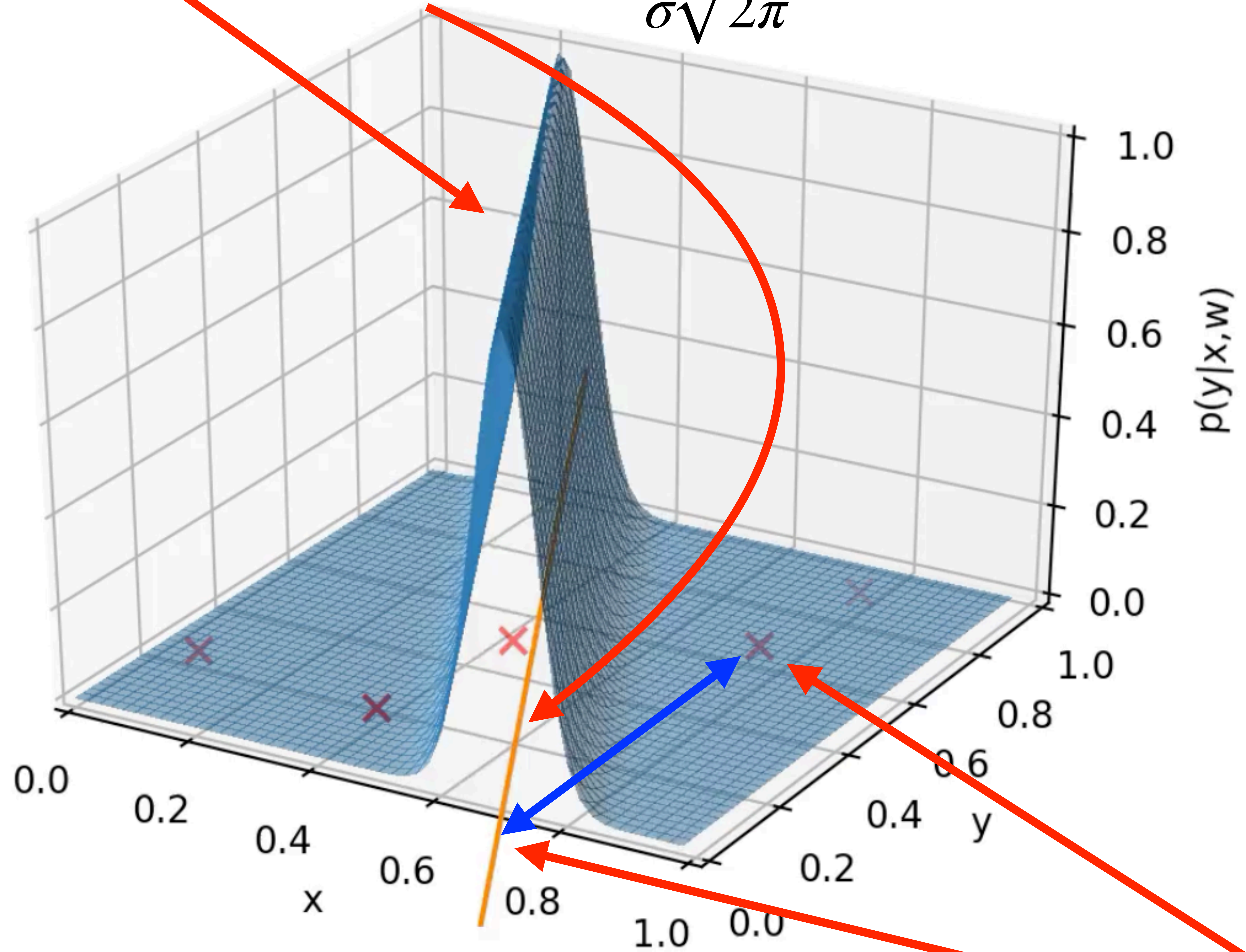
$$= \arg\max_{\mathbf{w}} \prod_{(\mathbf{x}_i,y_i)} p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$$

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



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

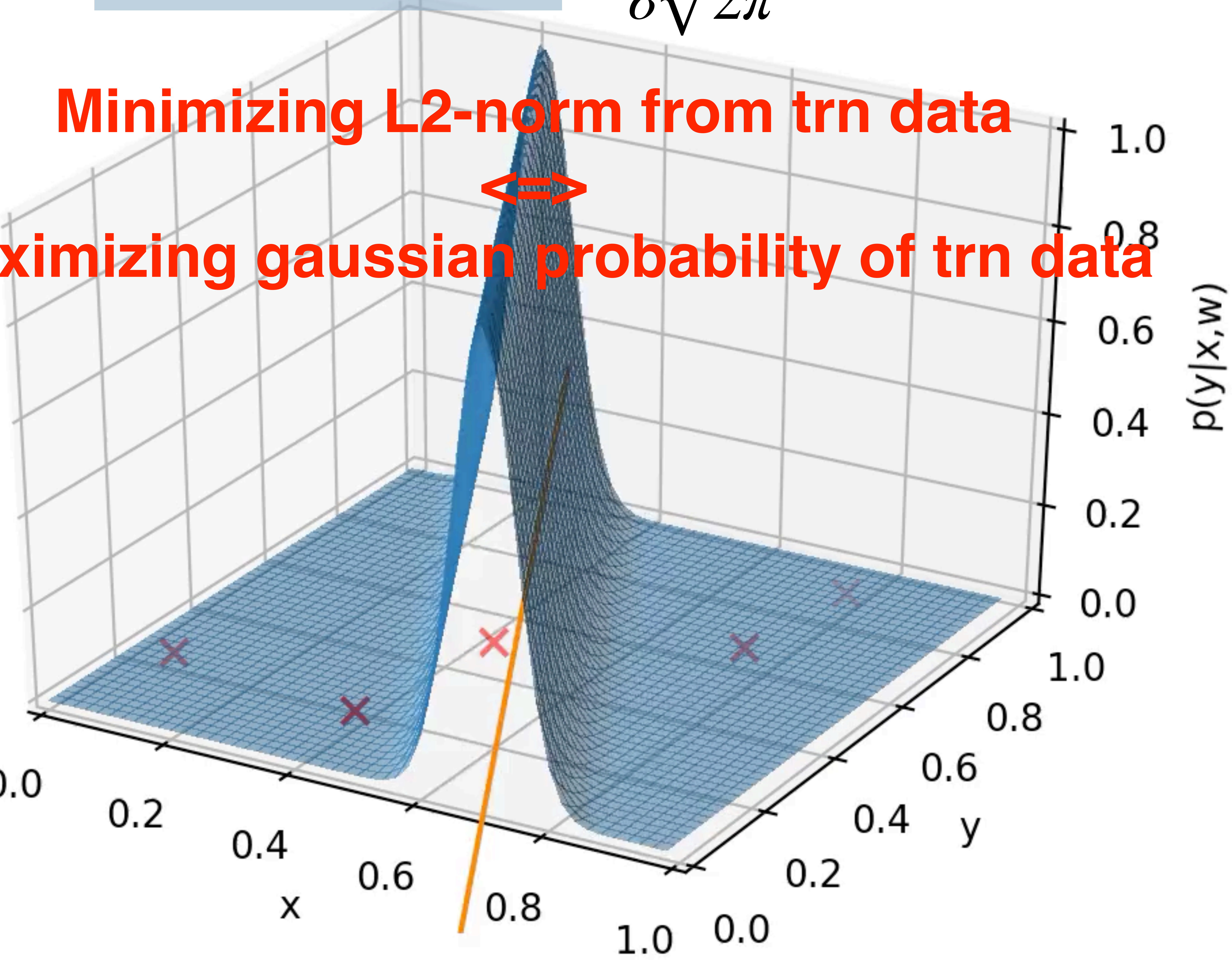
$$p(y\,|\,\mathbf{x}, \mathbf{w}) = \mathcal{N}(y;\ w_1x + w_0, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{w_1x+w_0-y}{\sigma}\right)^2}$$



Prove it !!!

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_i \mathcal{N}(y_i;\ w_1x_i + w_0, \sigma) = \arg\min_{\mathbf{w}} \sum_i (w_1x_i + w_0 - y_i)^2$$

$$p(y\,|\,\mathbf{x},\mathbf{w}) = \mathcal{N}(y;\ w_1x + w_0,\ \sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp^{-\frac{1}{2}\left(\frac{w_1x+w_0-y}{\sigma}\right)^2}$$



Minimizing L2-norm from trn data

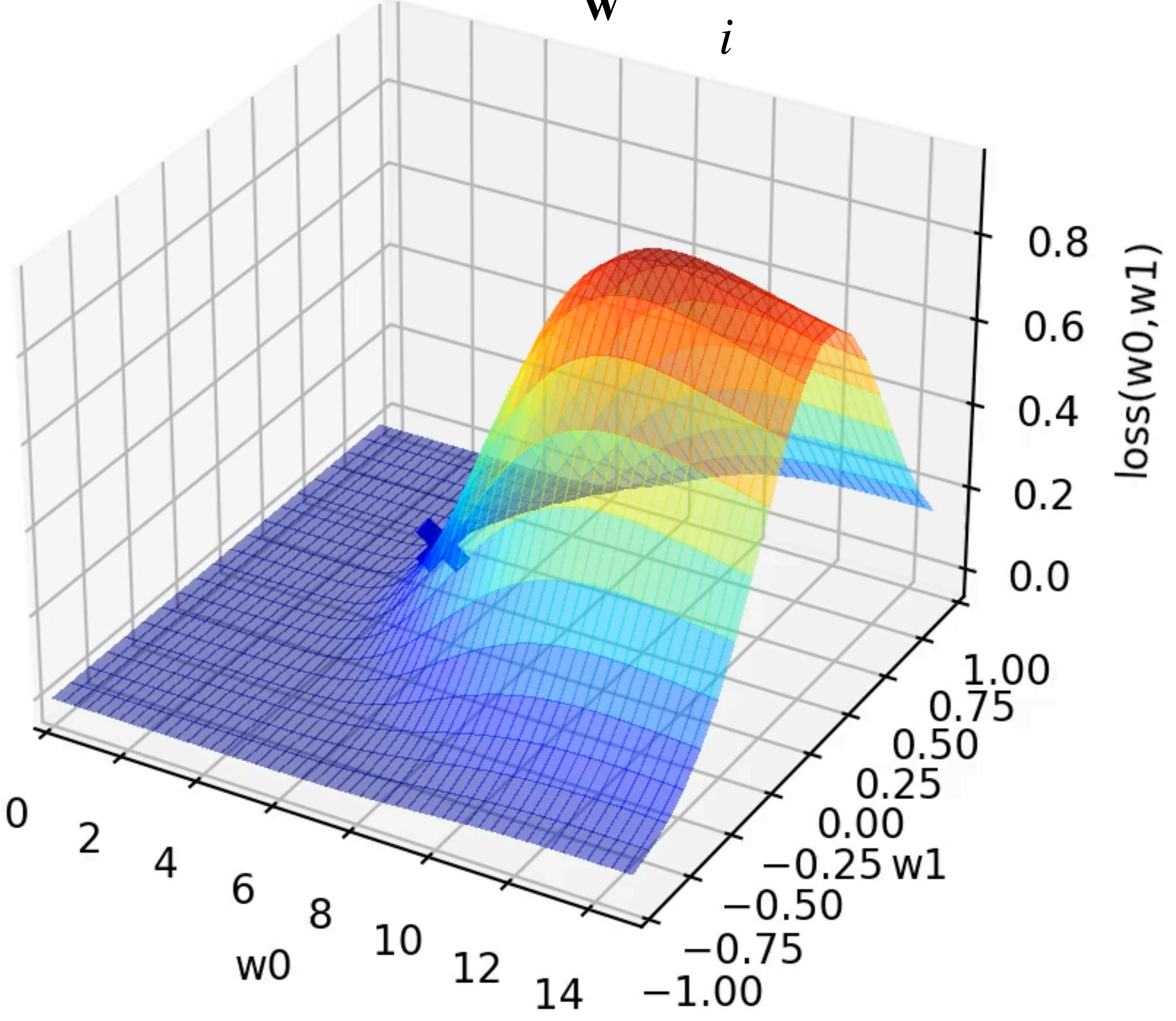
<=>

Maximizing gaussian probability of trn data

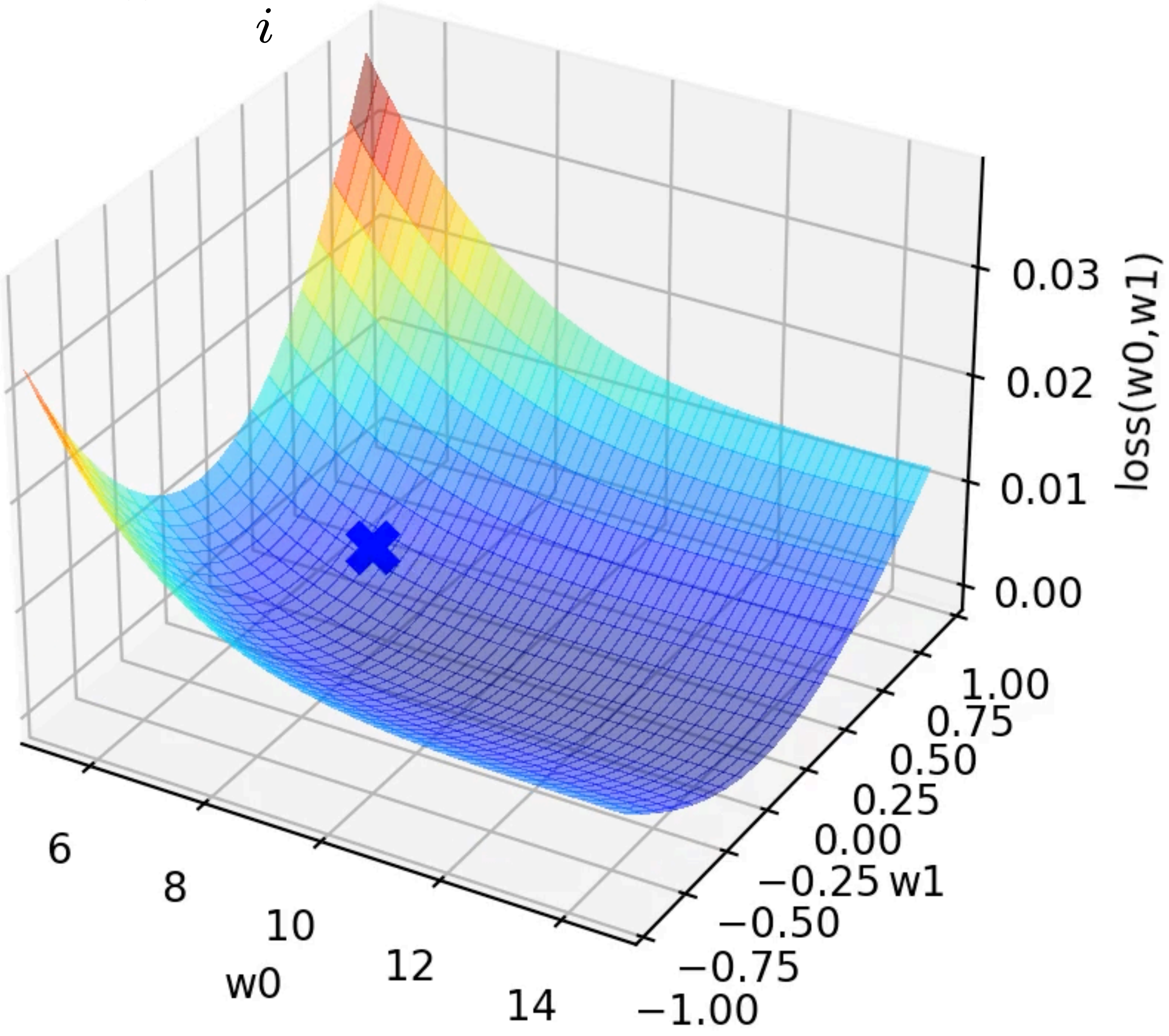
$$\mathbf{w}^{\star} = \arg\max_{\mathbf{w}} \prod_i \mathcal{N}(y_i;\ f(\mathbf{x}_i,\mathbf{w}),\sigma) = \arg\min_{\mathbf{w}} \sum_i (w_1x_i + w_0 - y_i)^2$$

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

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_i \mathcal{N}(y_i;\, f(\mathbf{x}_i, \mathbf{w}), \sigma) \quad = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$



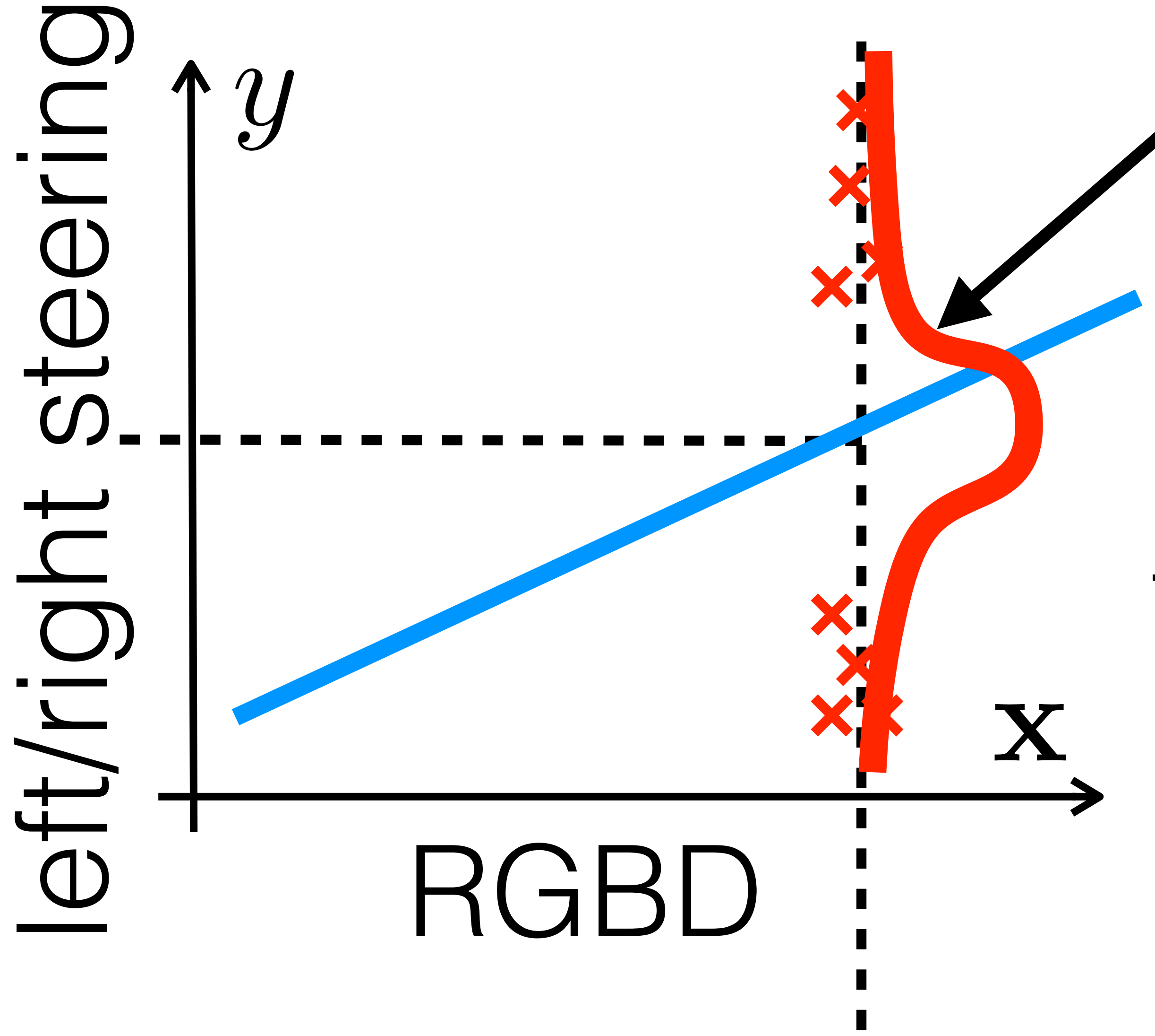
MLE

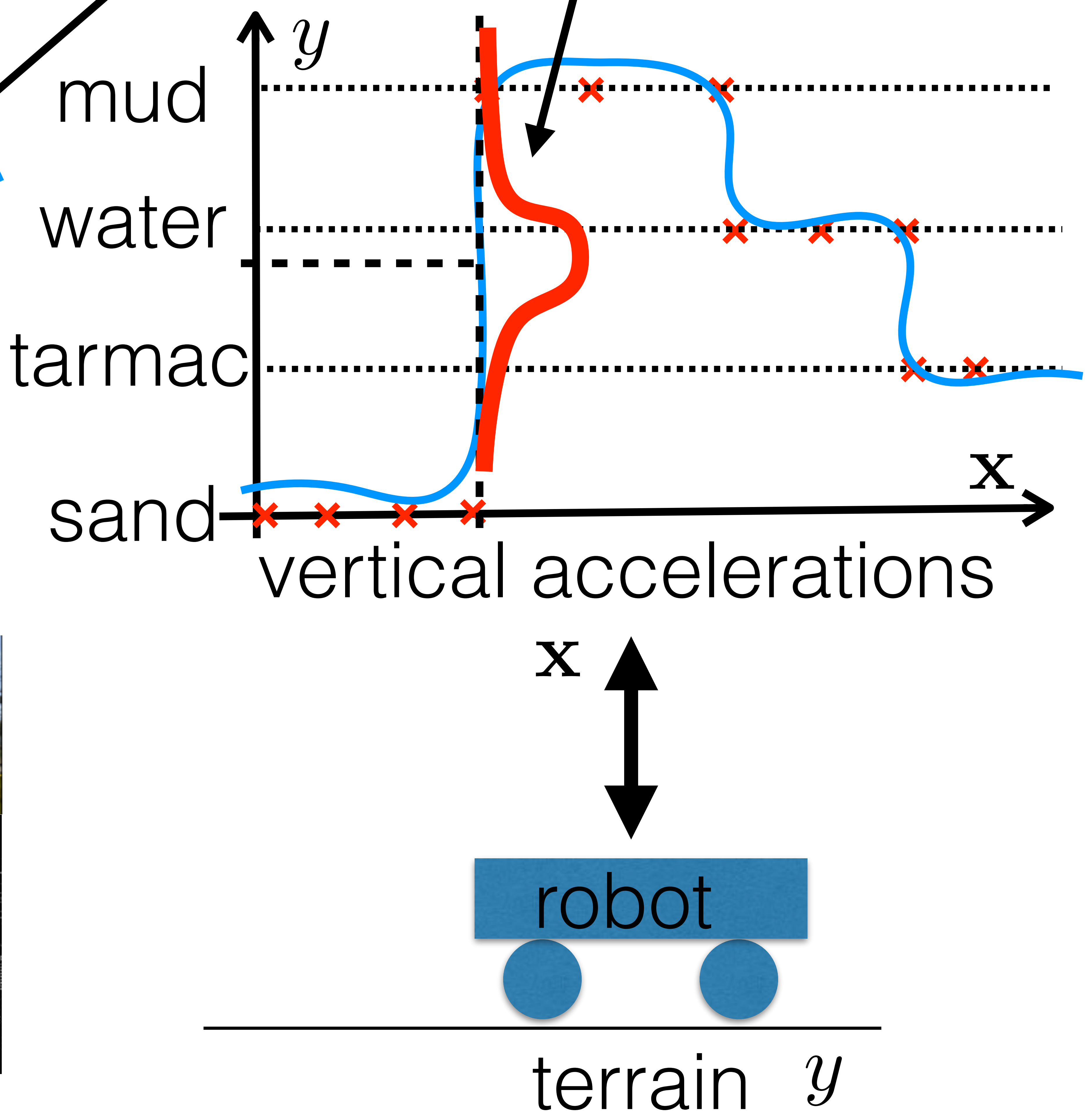


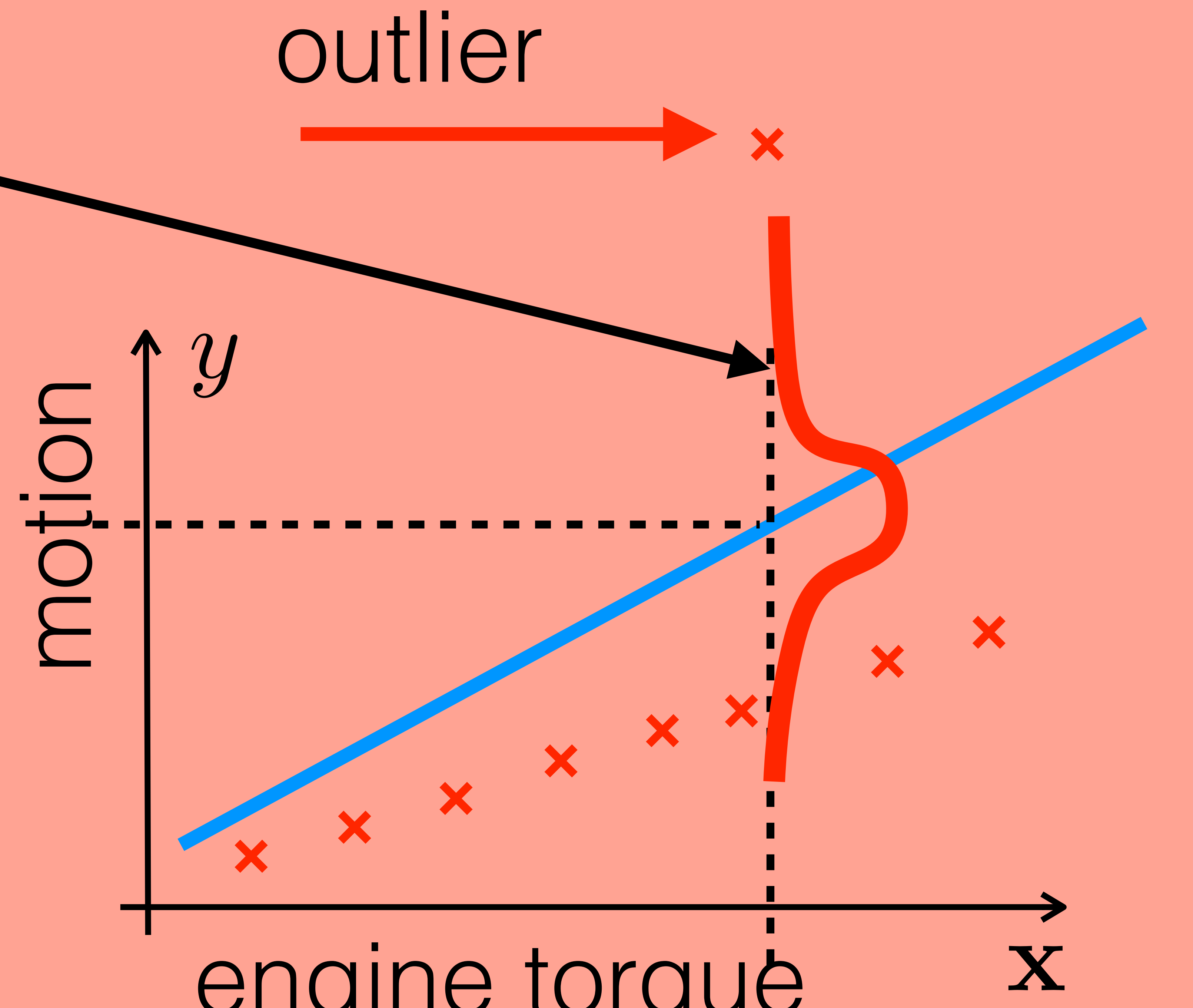
LSQ

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

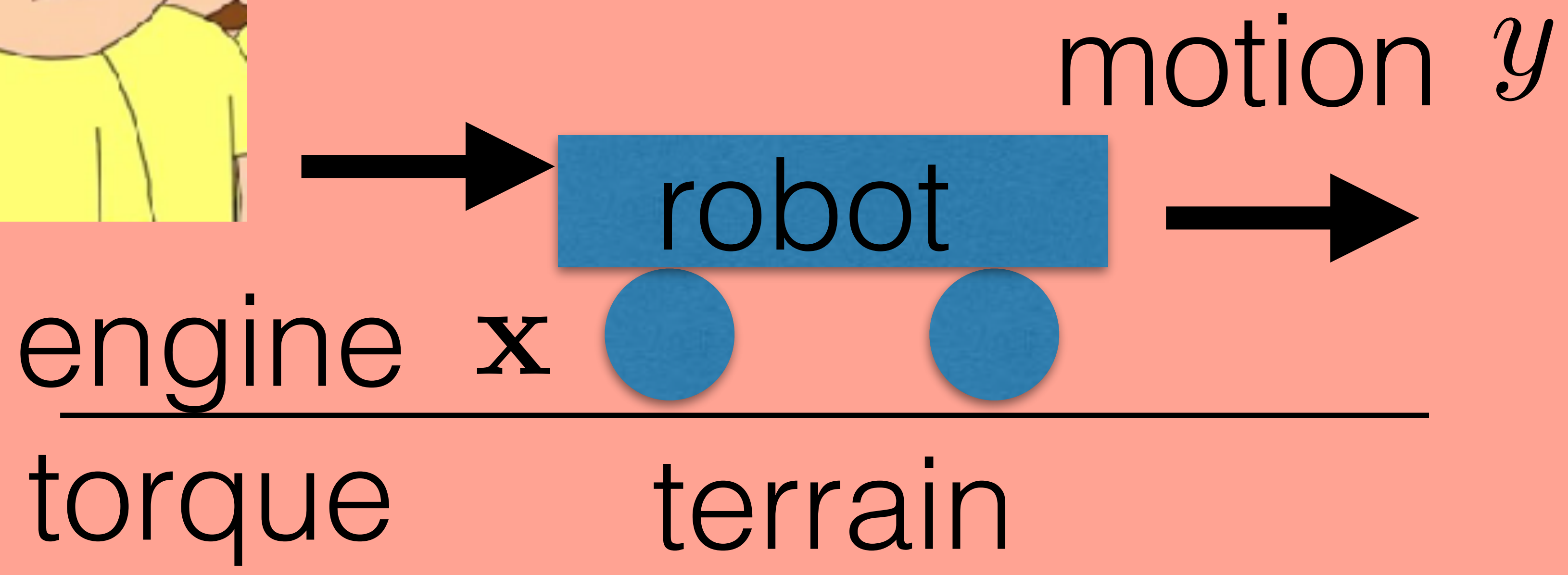
This is the problem!





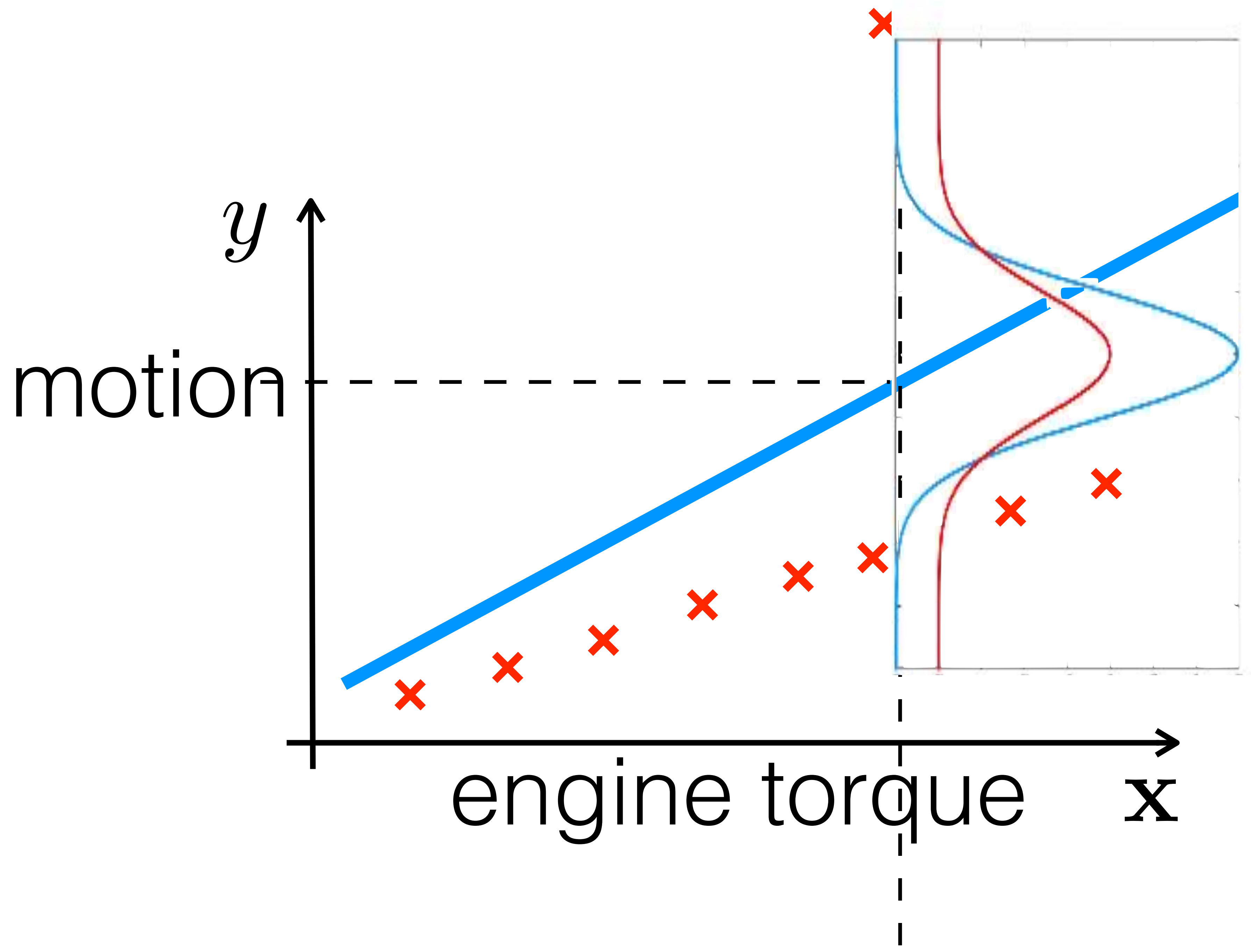


Nature is evil !!!

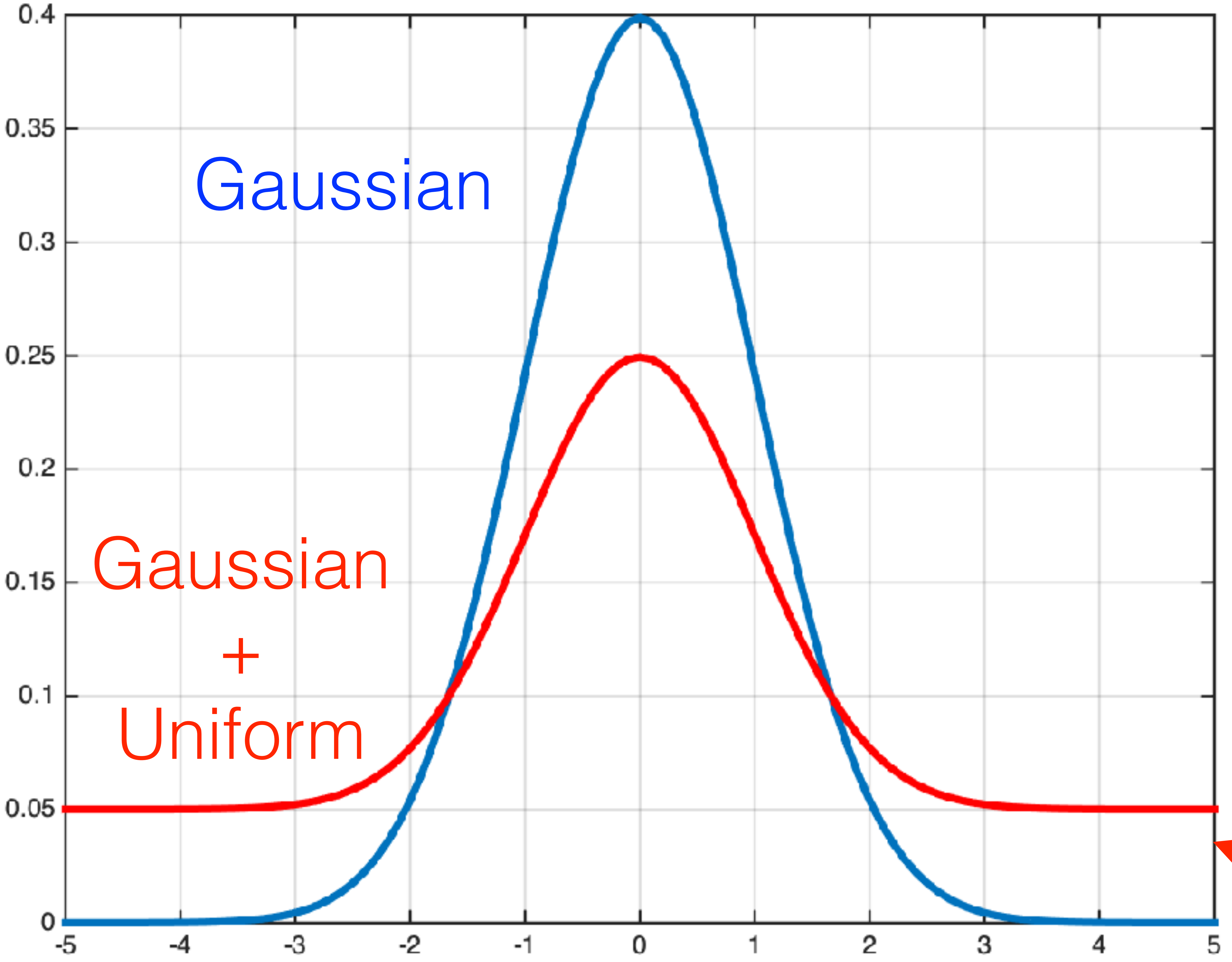


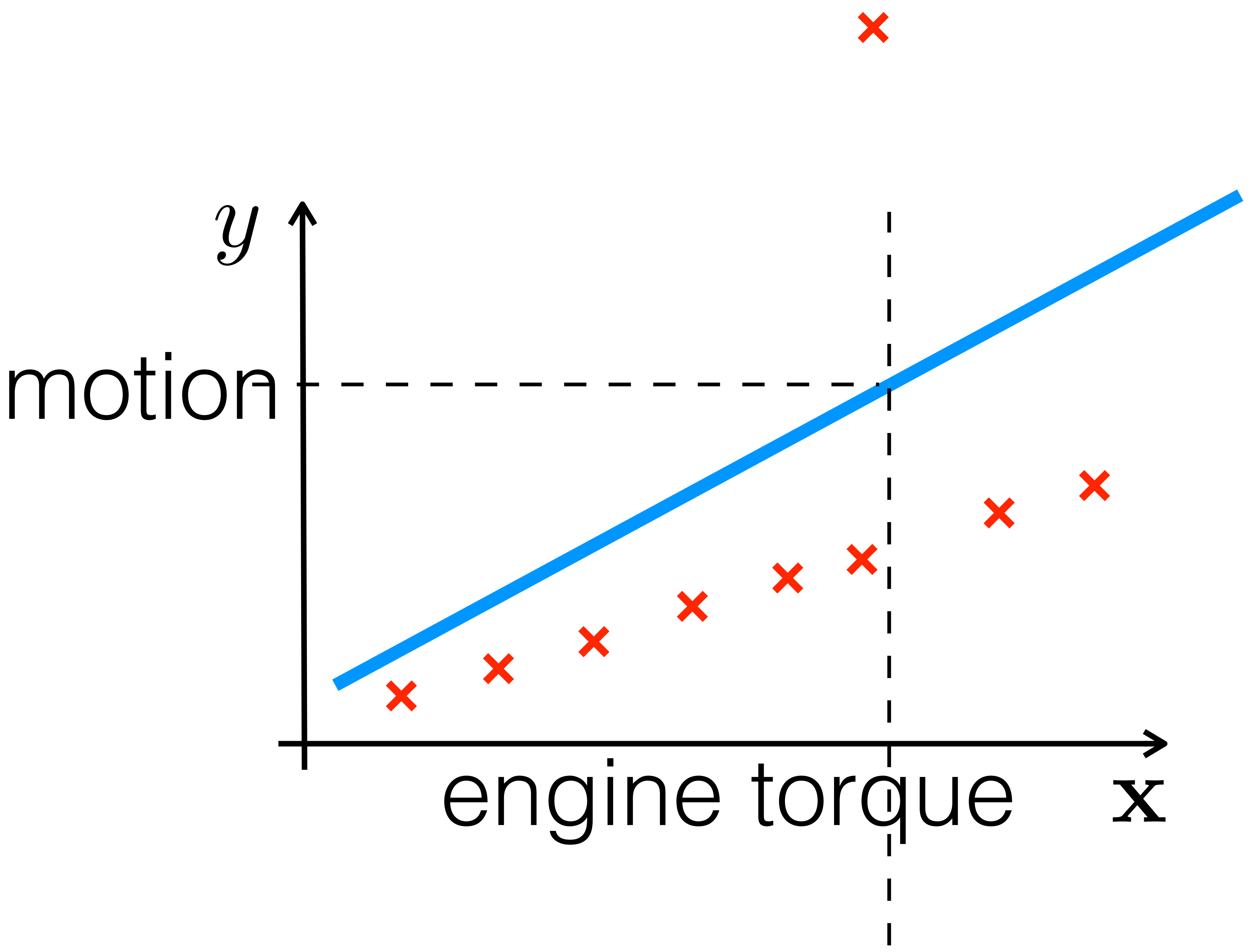
https://rickandmorty.fandom.com/wiki/Evil_Morty

Robust regression



Robust regression



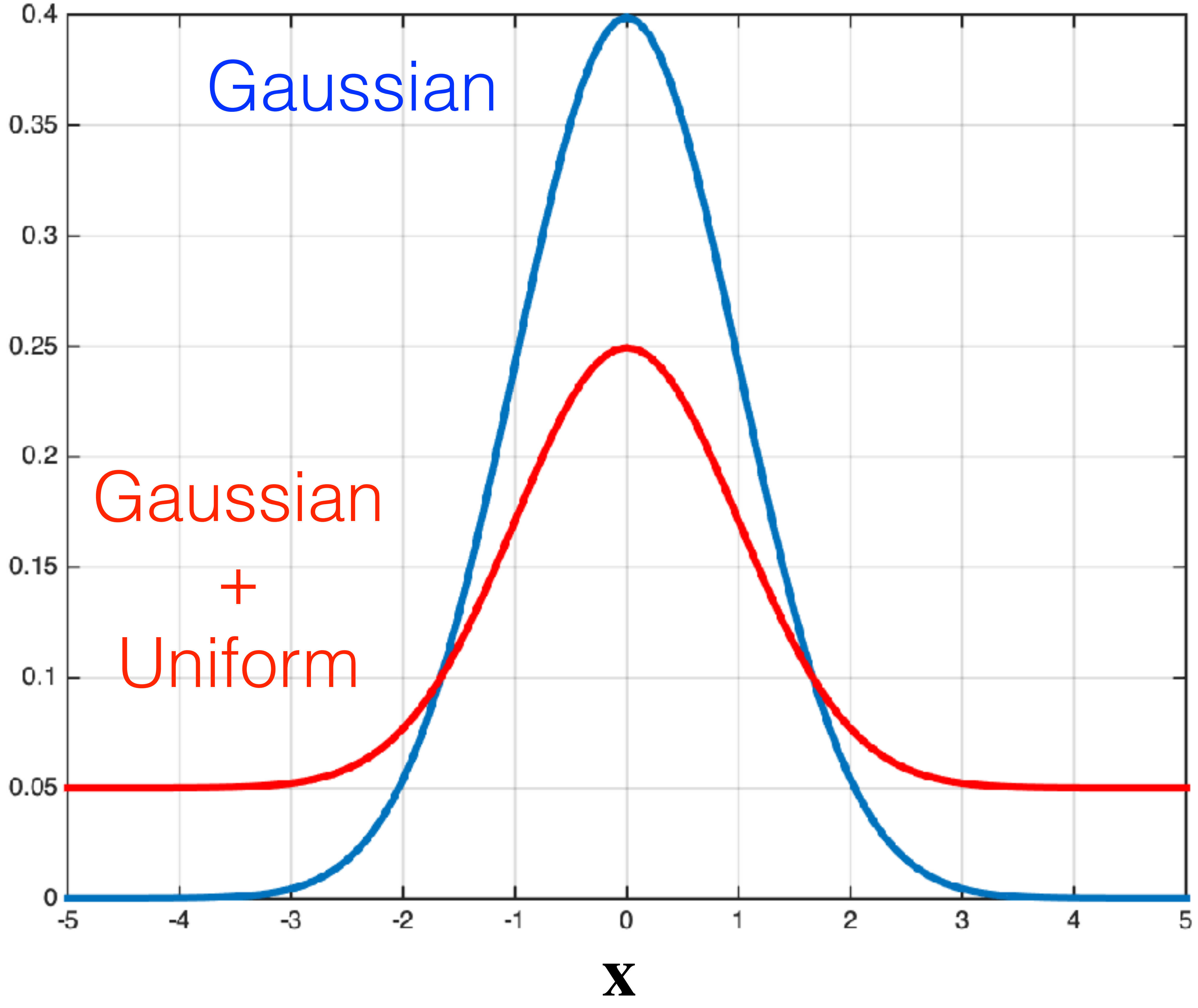


Evil source of uniform noise

Robust regression

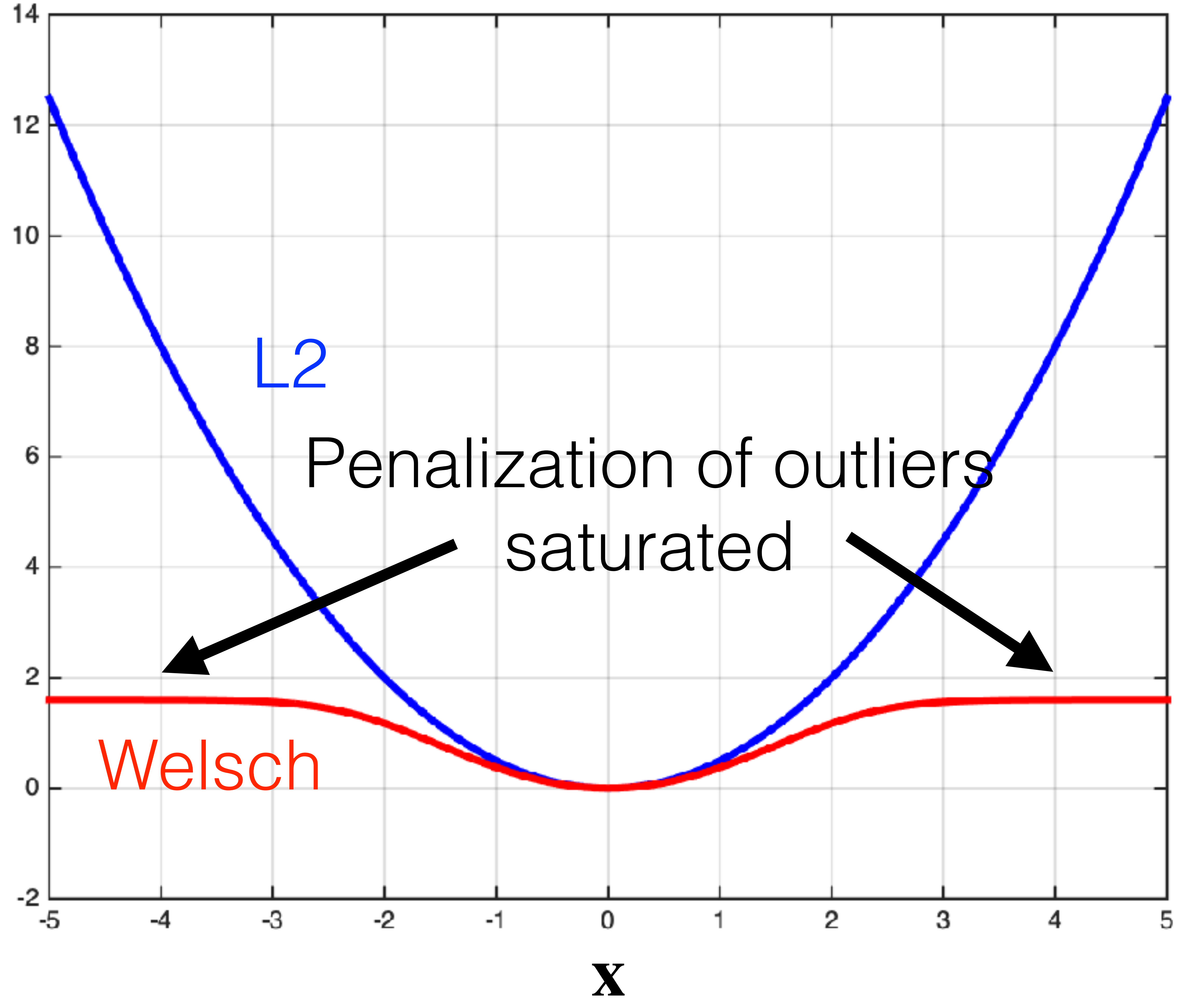
$$p(y \mid \mathbf{x}, \mathbf{w})$$

Probability distributions



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

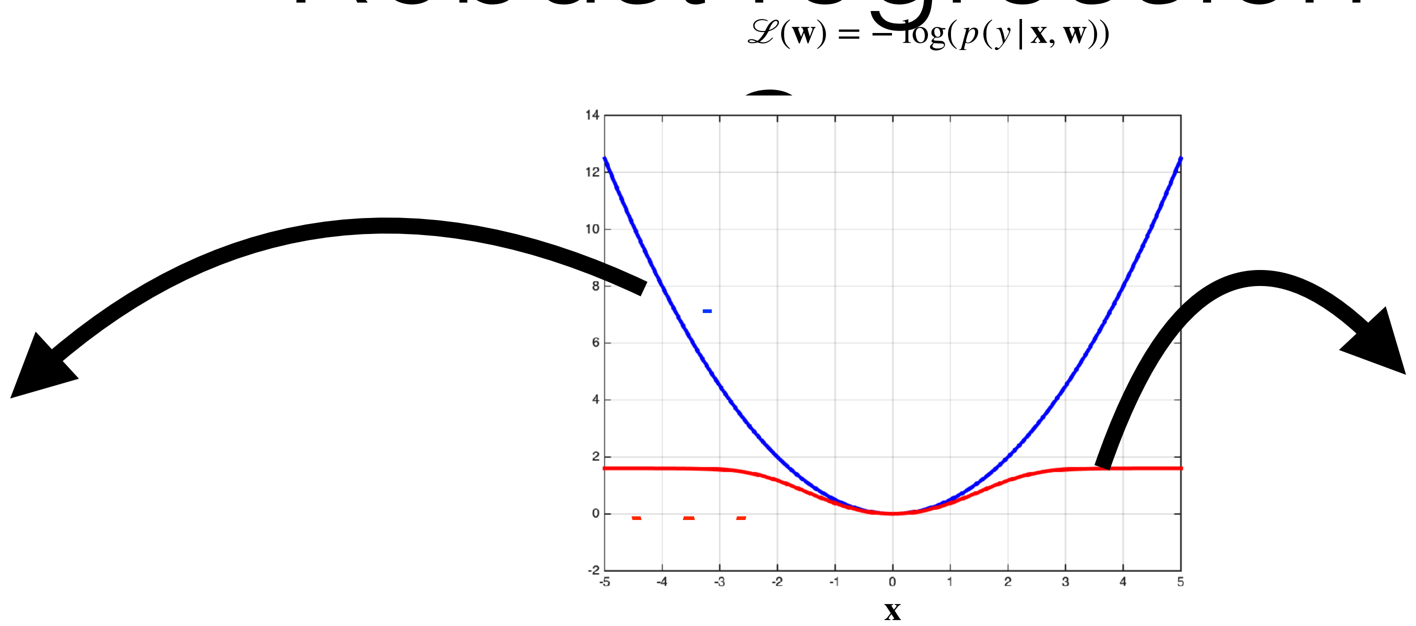
Corresponding losses



Penalization of outliers saturated

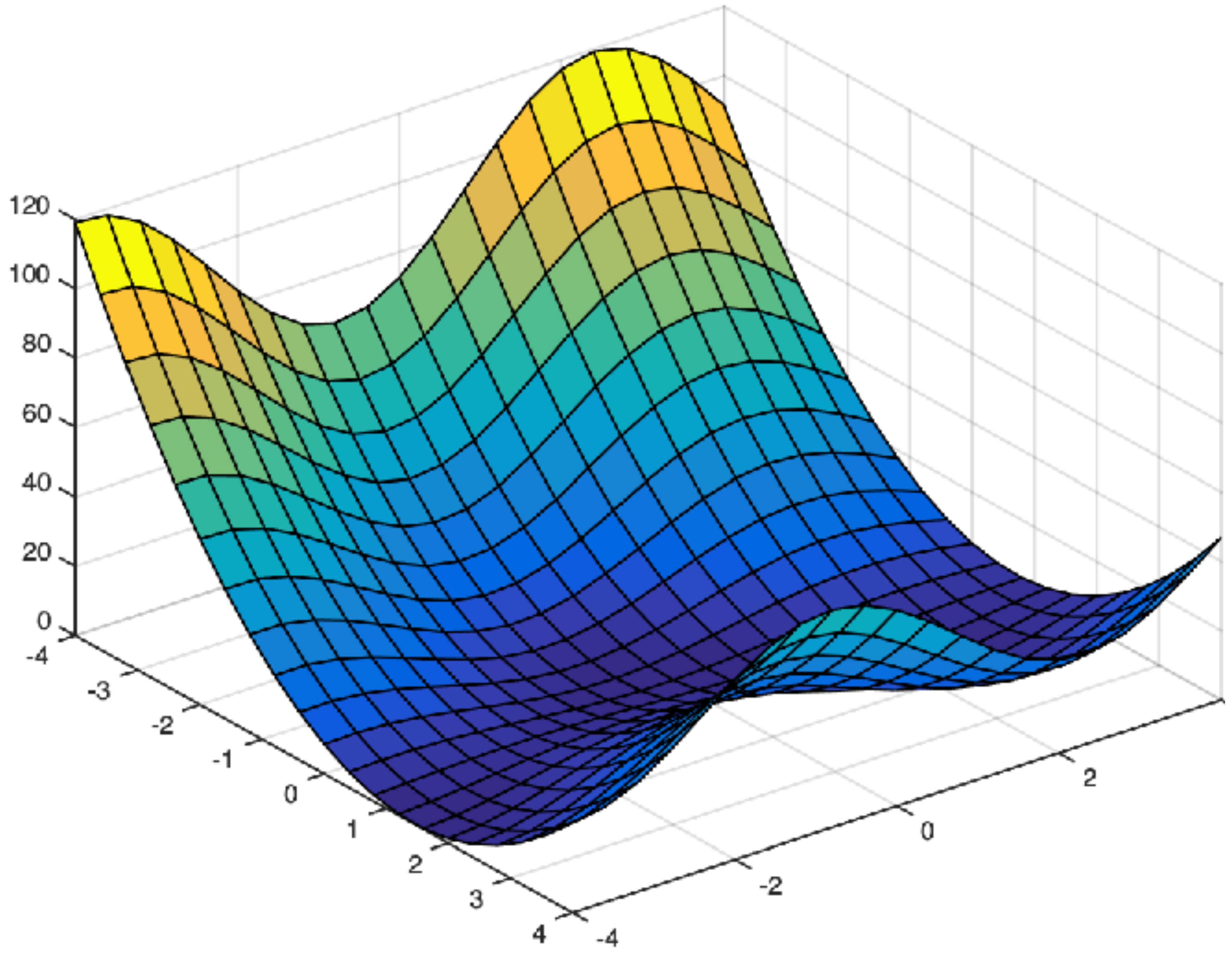
Can you guess where another problem appears?

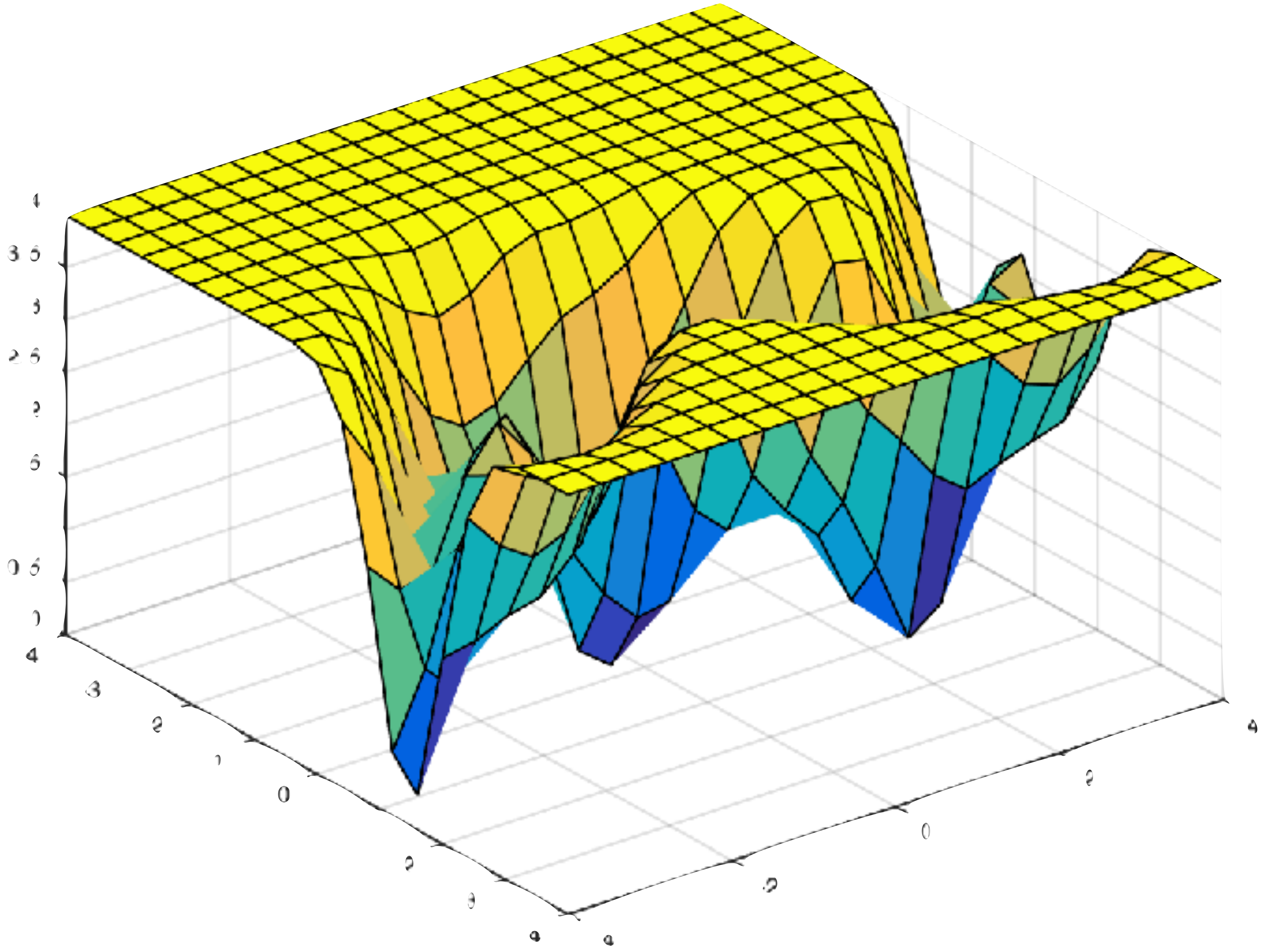
Robust regression



L2 landscape

Welsch landscape



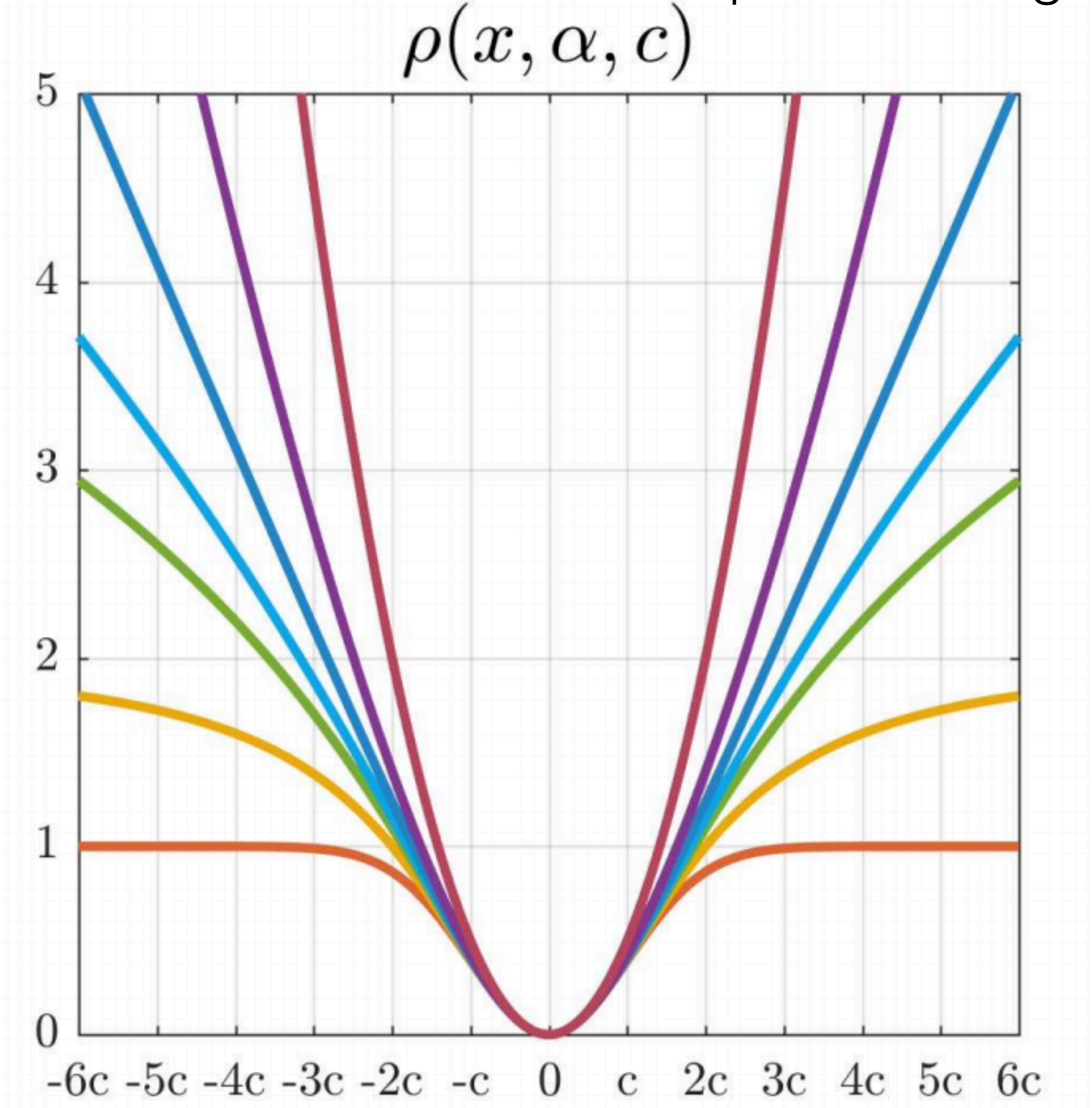


- **Uniform noise omitted**
  => GD-friendly landscape
- Gradient length encodes distance
- Easy to optimize

- **Uniform noise modelled**
  => GD-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



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

Trade-off:

Robustness to uniform noise (outliers)
vs
Optimization-friendly landscape

The best what you can do:

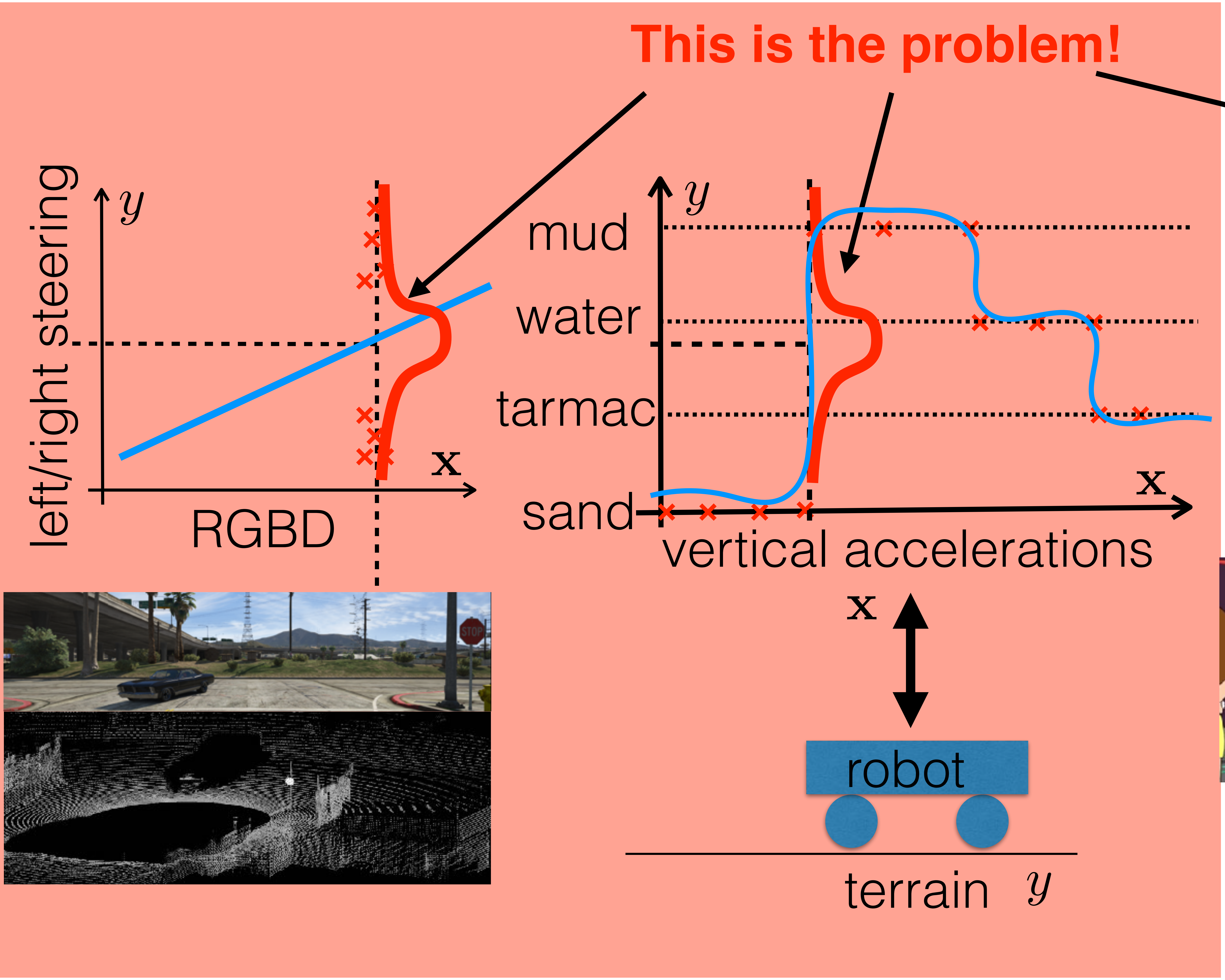
What can go wrong: **inappropriate choice of loss function**

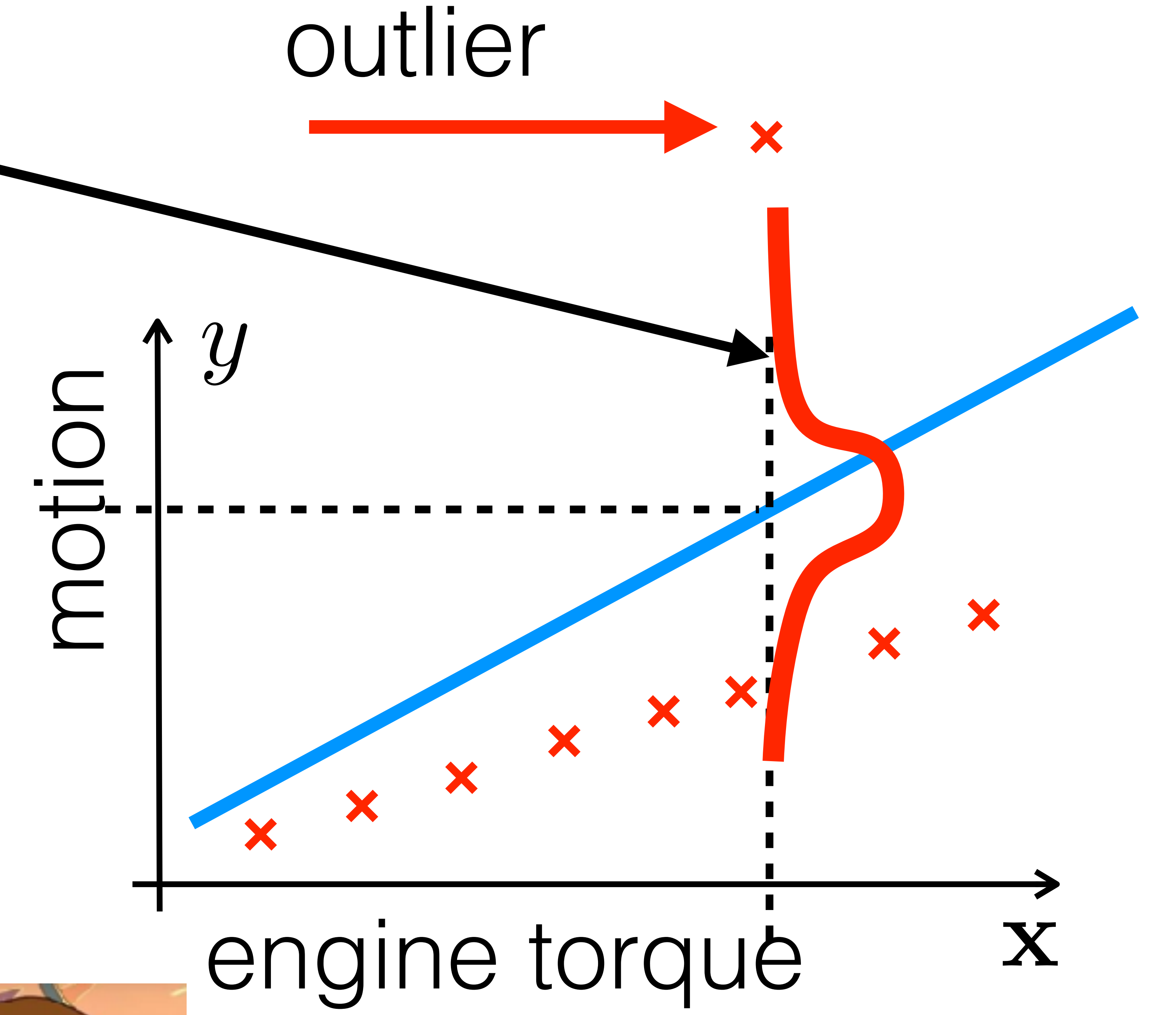
This is the problem!

Nature is evil !!!

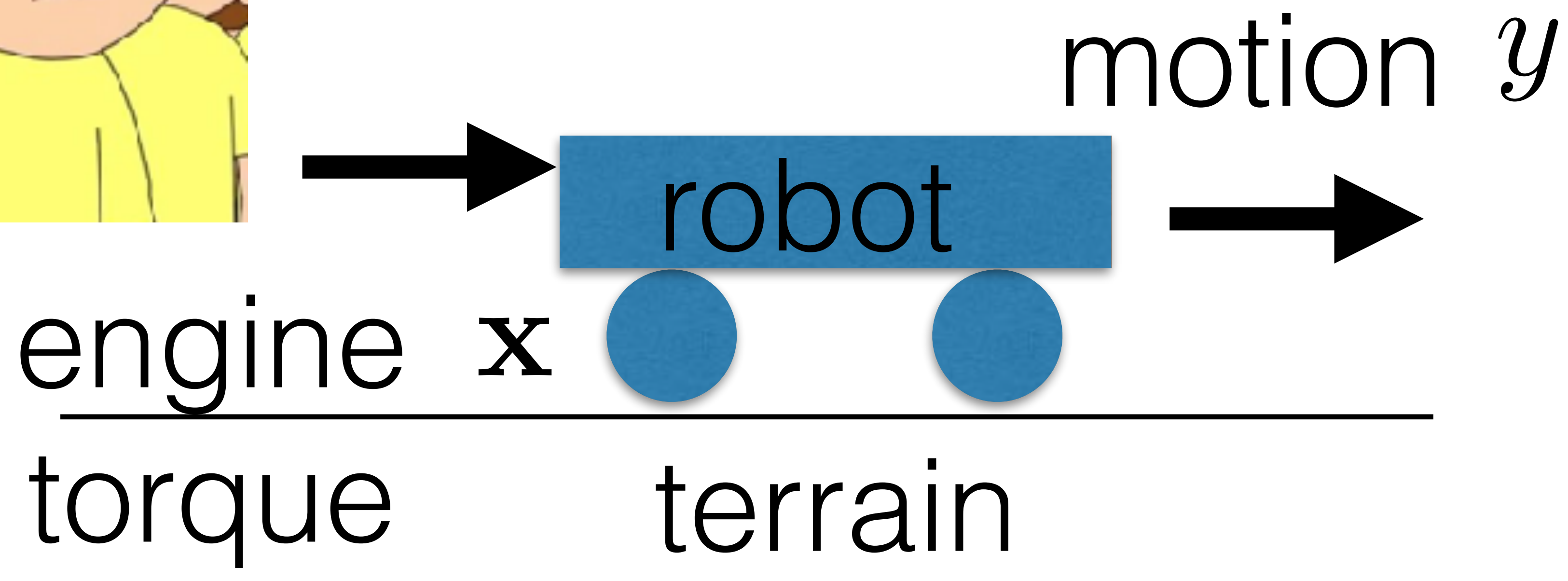
https://rickandmorty.fandom.com/wiki/Evil_Morty

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



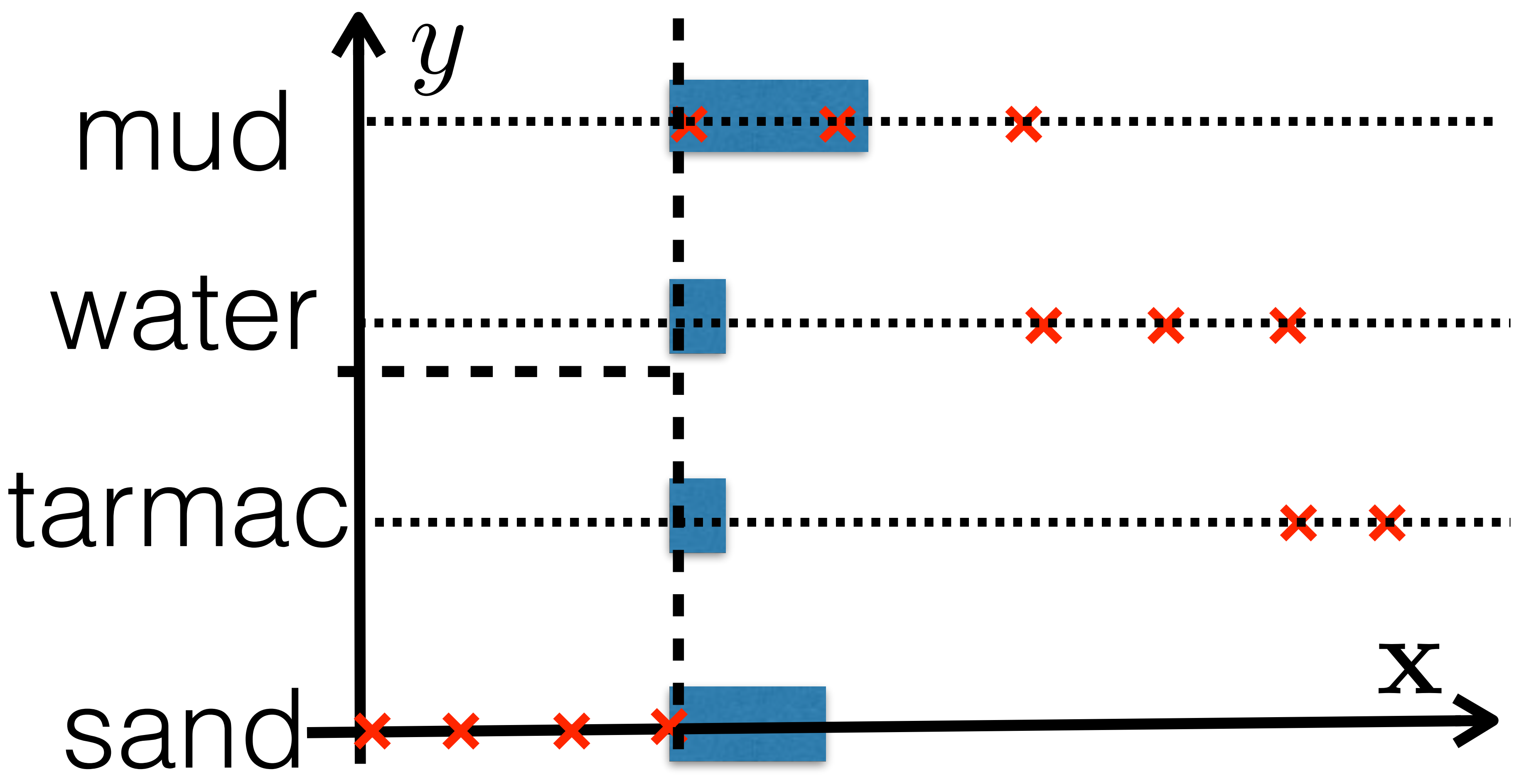


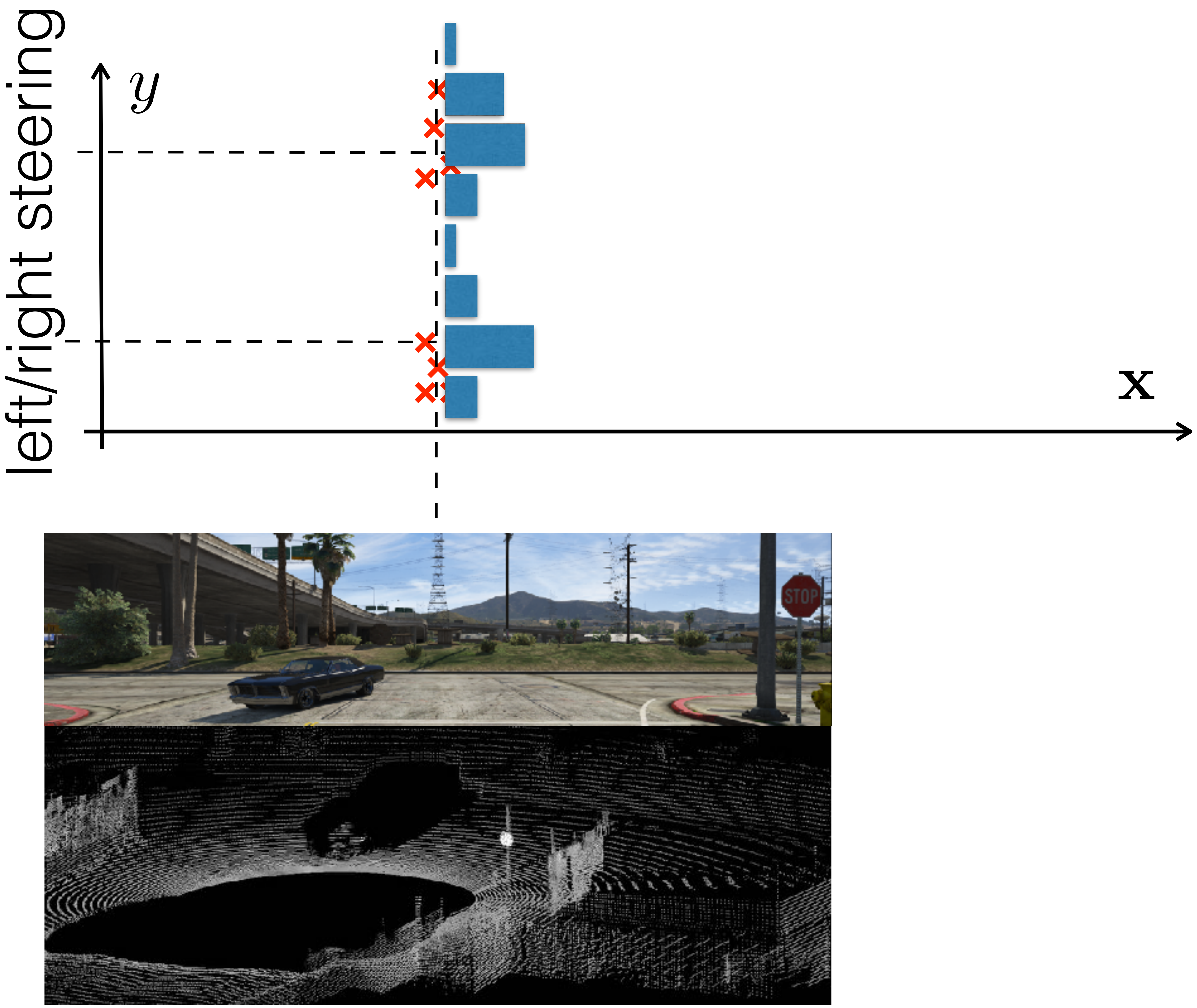
Nature is evil !!!



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

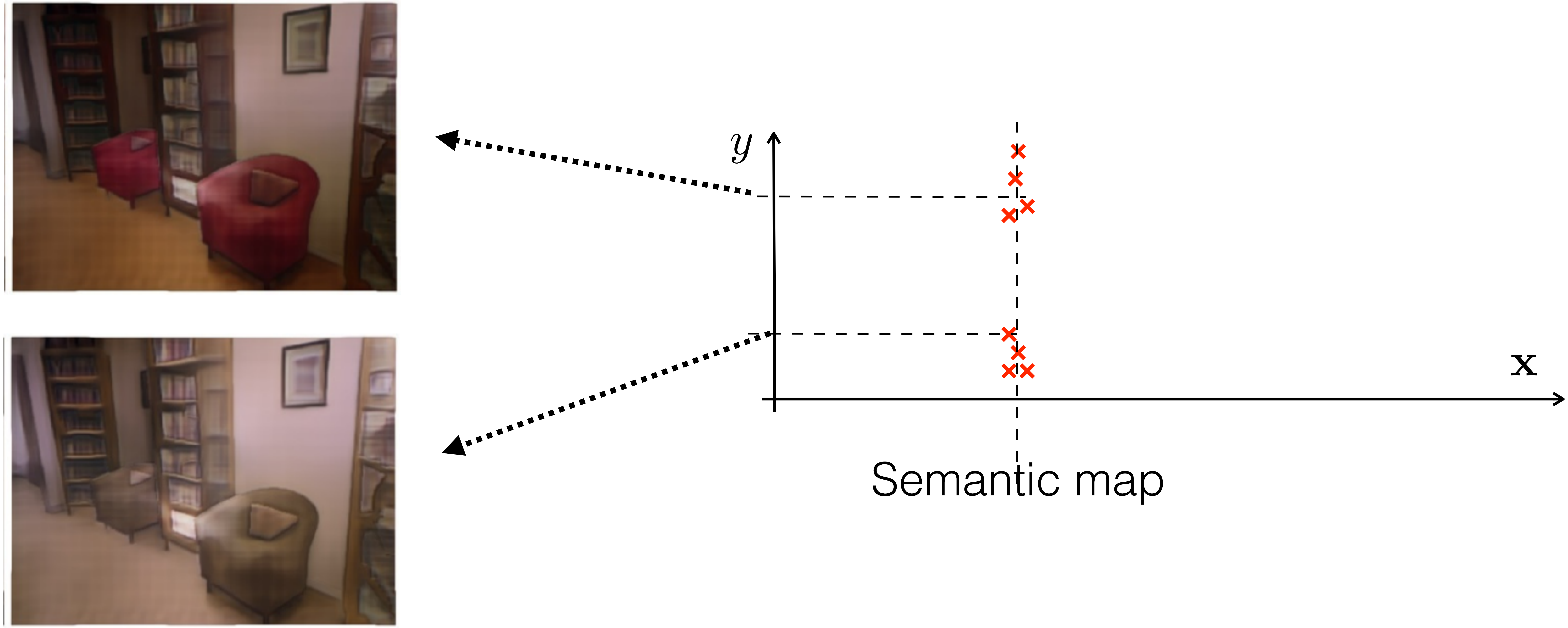
Work-around 1: discretize y-domain and treat the problem as classifcation





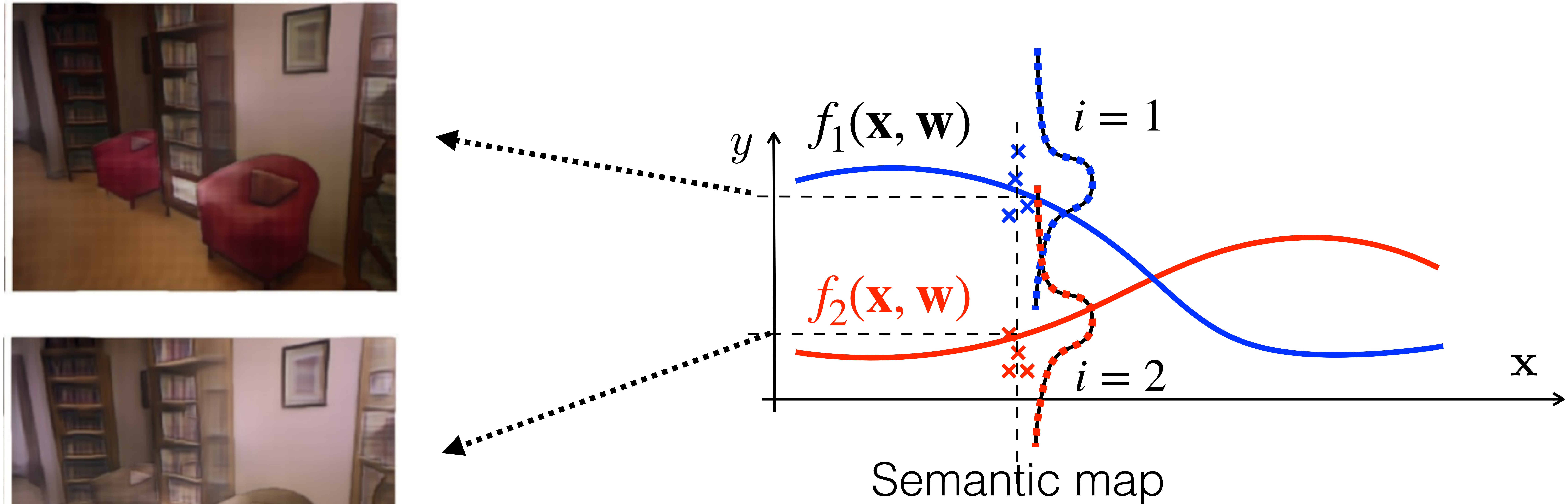
Can I use it always?

Tractable only for low-dim y



What if y are images?

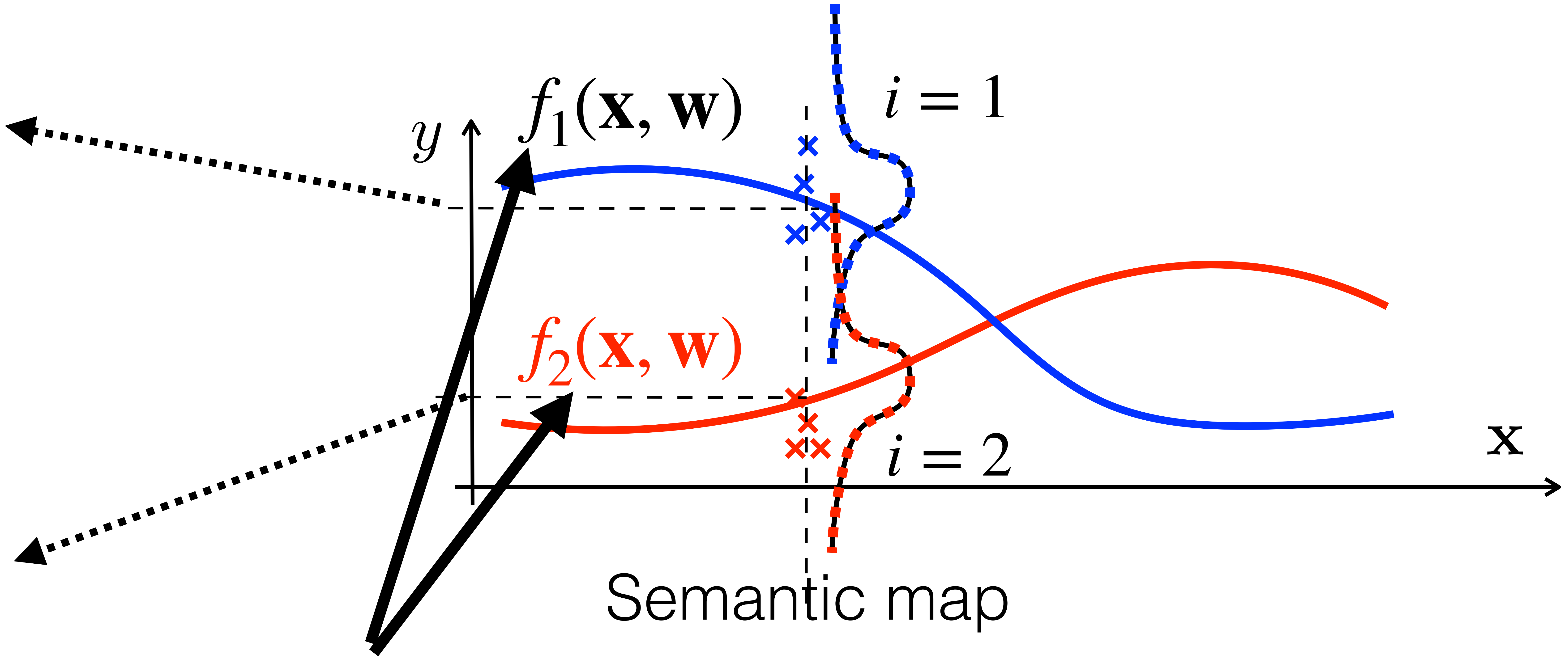
Work-around 2: allow multiple hypothesis



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



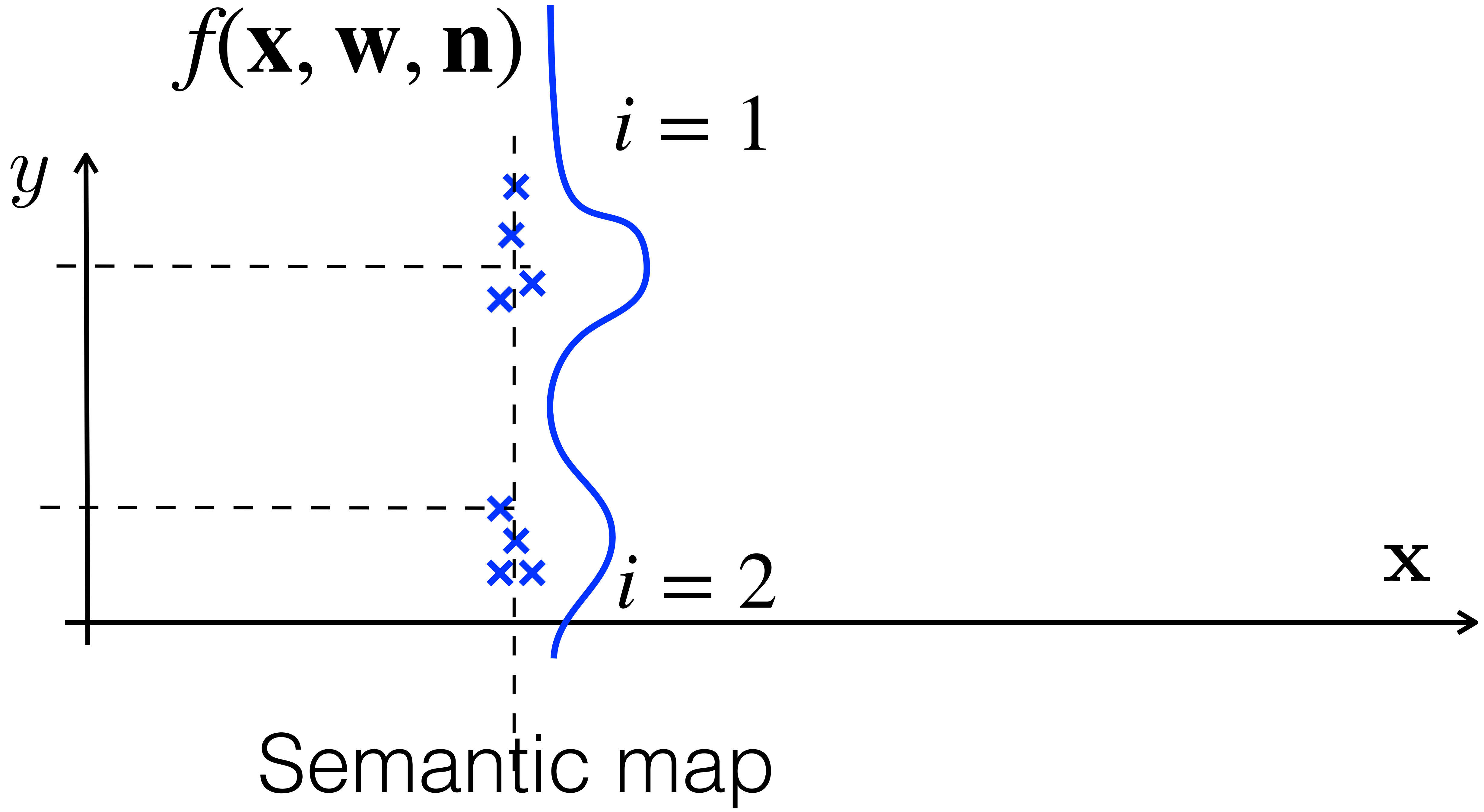
Problem 1: number of hypothesis may grow exponentially

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

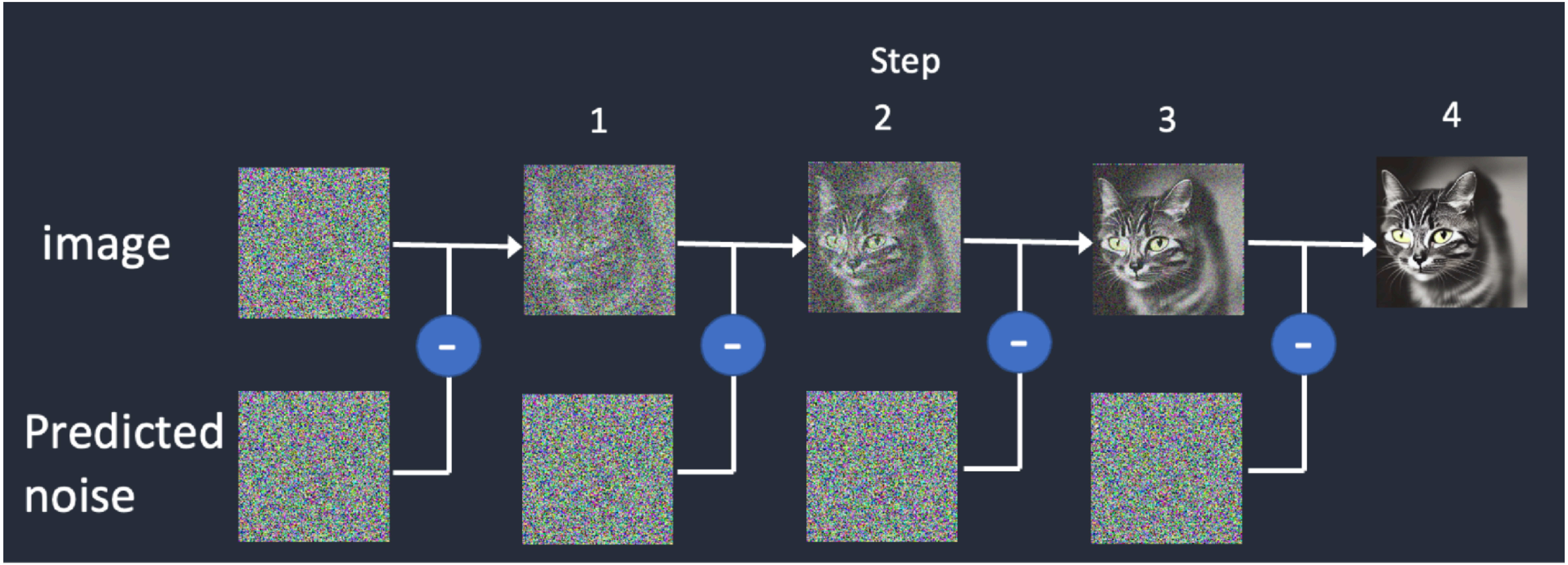
Work-around 3: use generative model



Generative models: GANs [Goodfellow 2016], VAE,

Diffusion models [Ho, NIPS 2020]

Work-around 3: use generative model

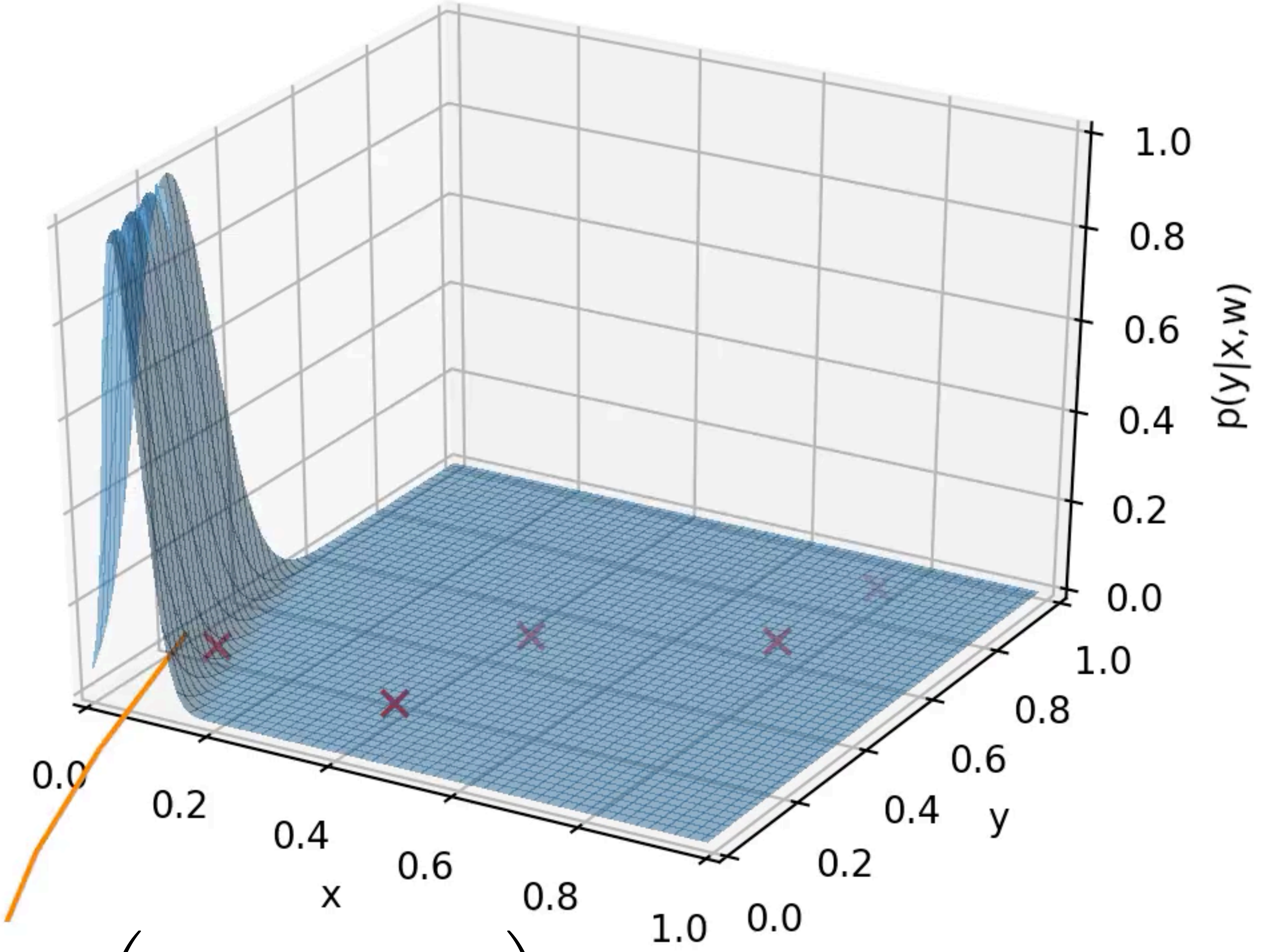


Summary loss

- -log(p)-loss stems from fitting the network parameterized p(y|x,w) distr. into data
- Maximum Likelihood = Minimum KL-divergence = Minimum -log(p)-loss
- Different distributions suffer from optimization issues (zero gradients, sensitivity to good initialization, local optima, …)

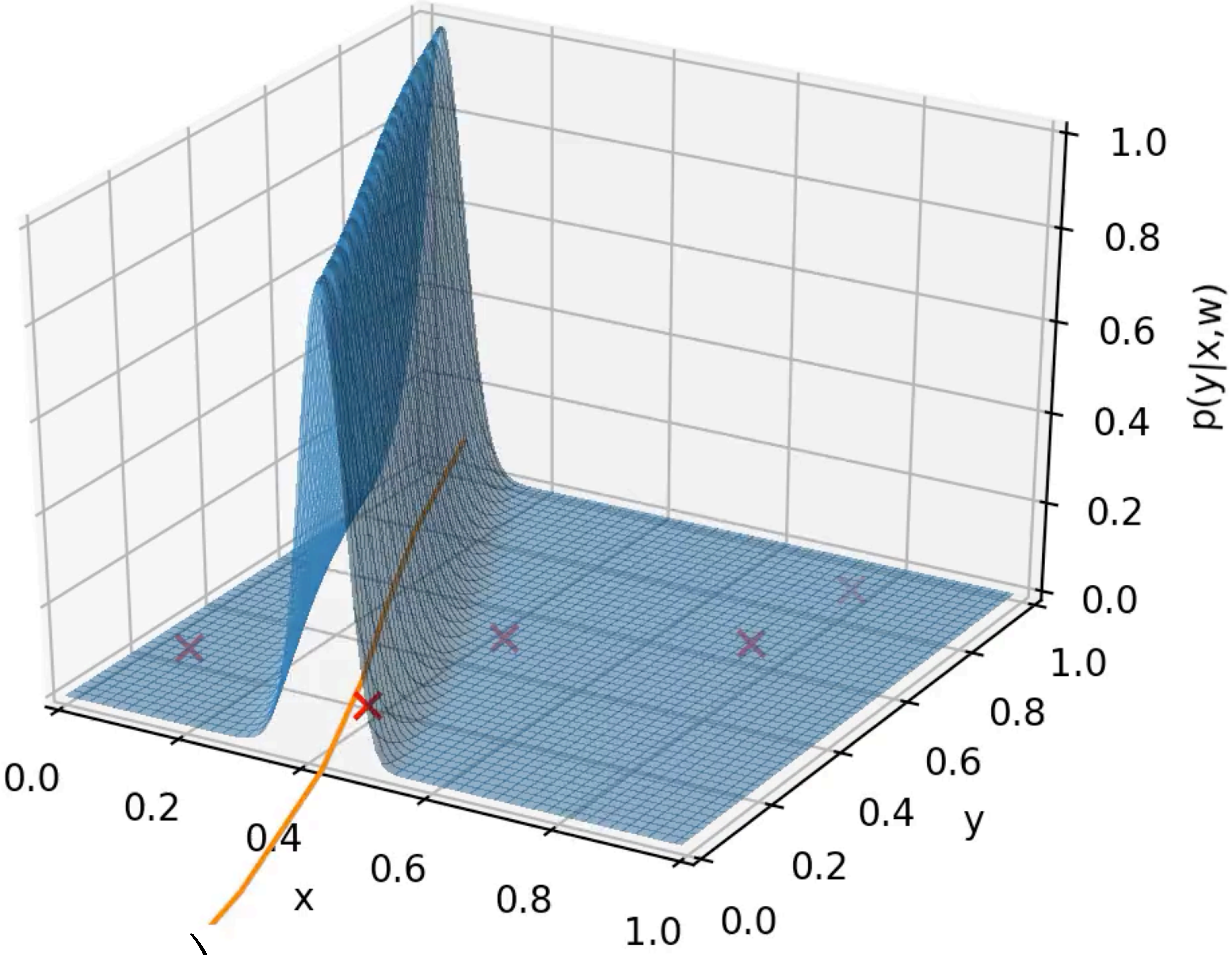
Where does the **overfitting** come from?

Where does the **overfitting** come from?



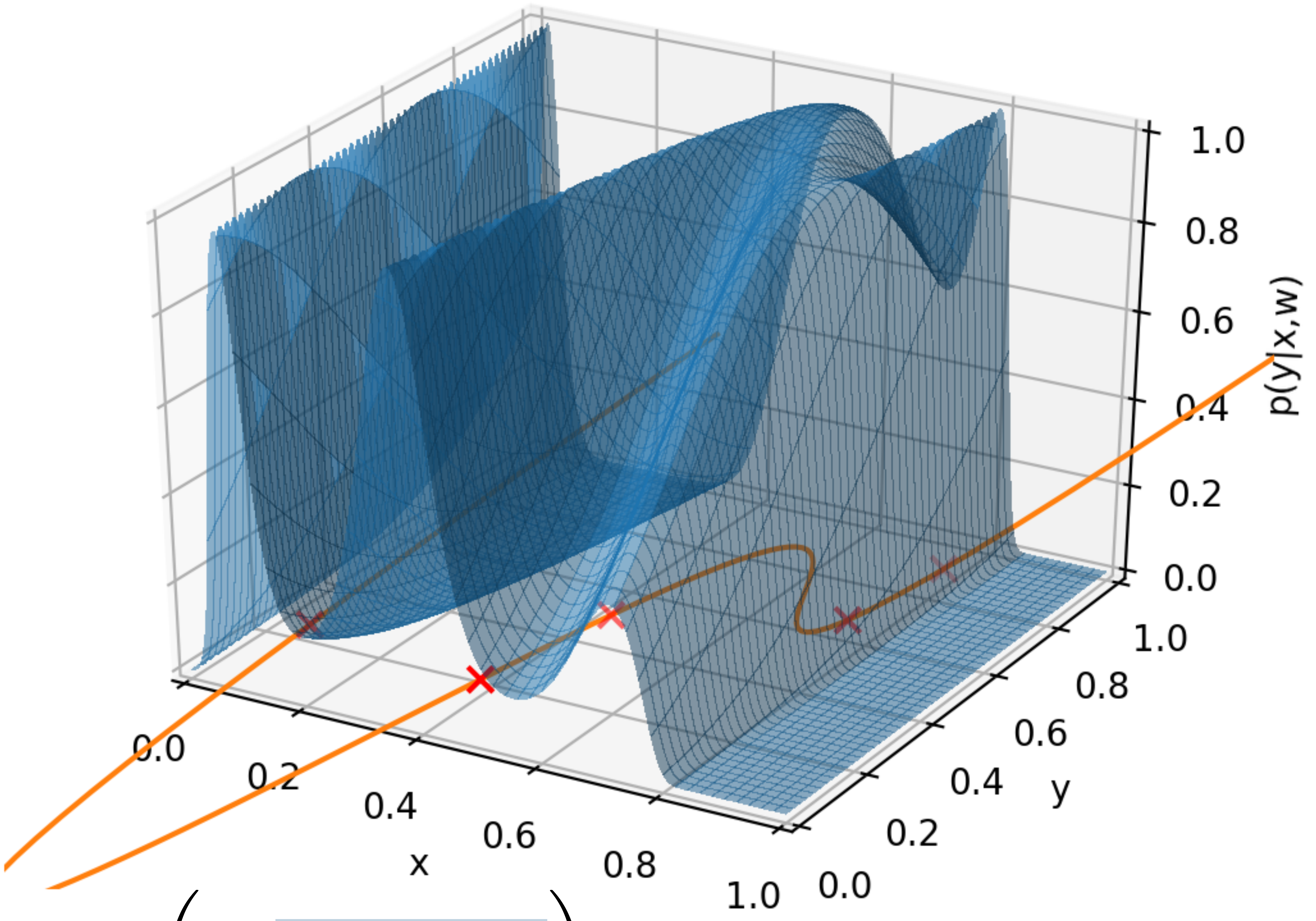
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_2 x_i^2 + w_1 x_i + w_0 - y_i)^2$$

Where does the **overfitting** come from?



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

Where does the **overfitting** come from?



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

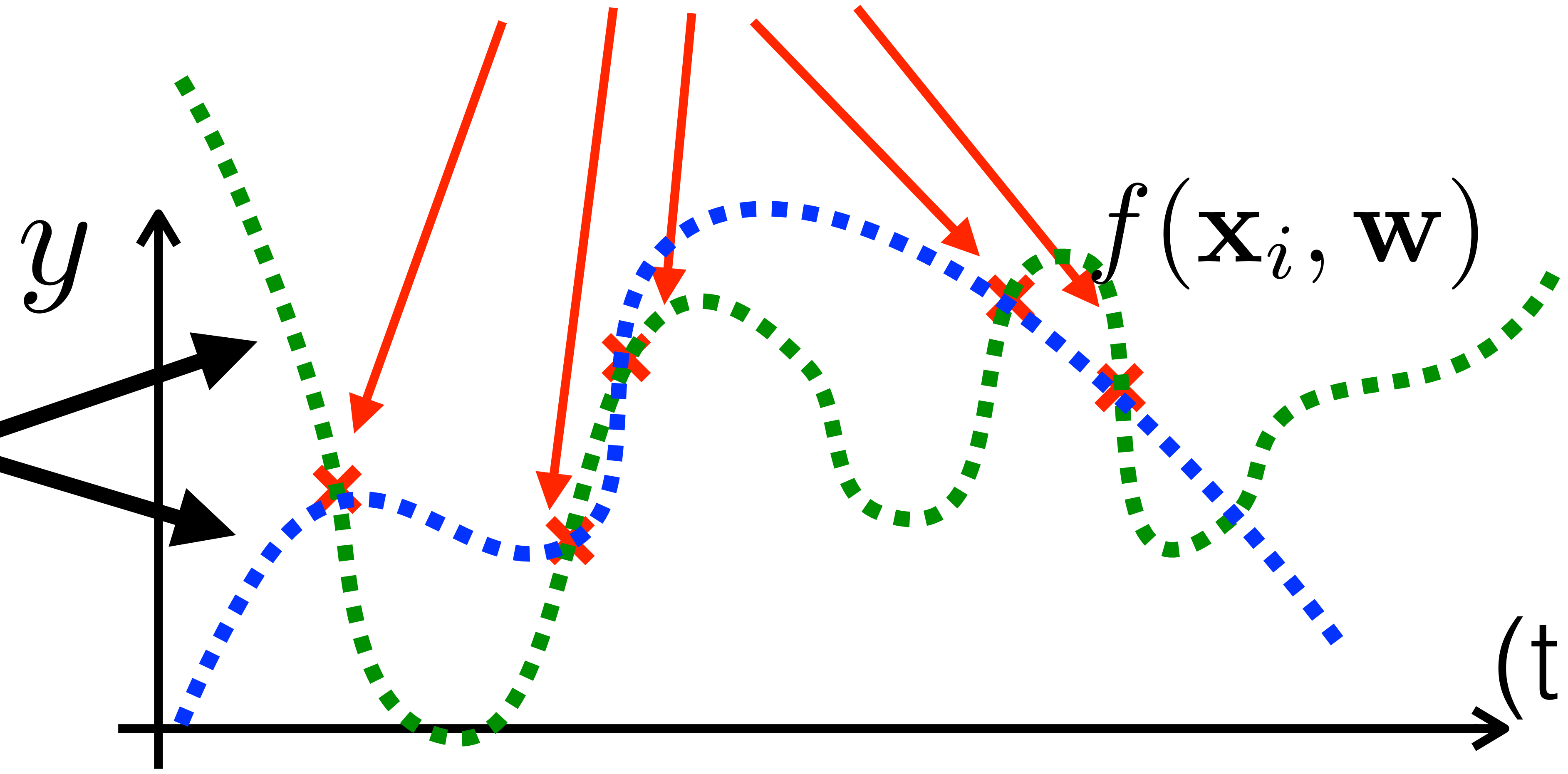
Where does the **overfitting** come from?

William of Ockham
(1287-1347)

leprechauns can be
involved in any explanation

observations

Many stories consistent with
the broken vase observation



The space of possible
stories is too wild

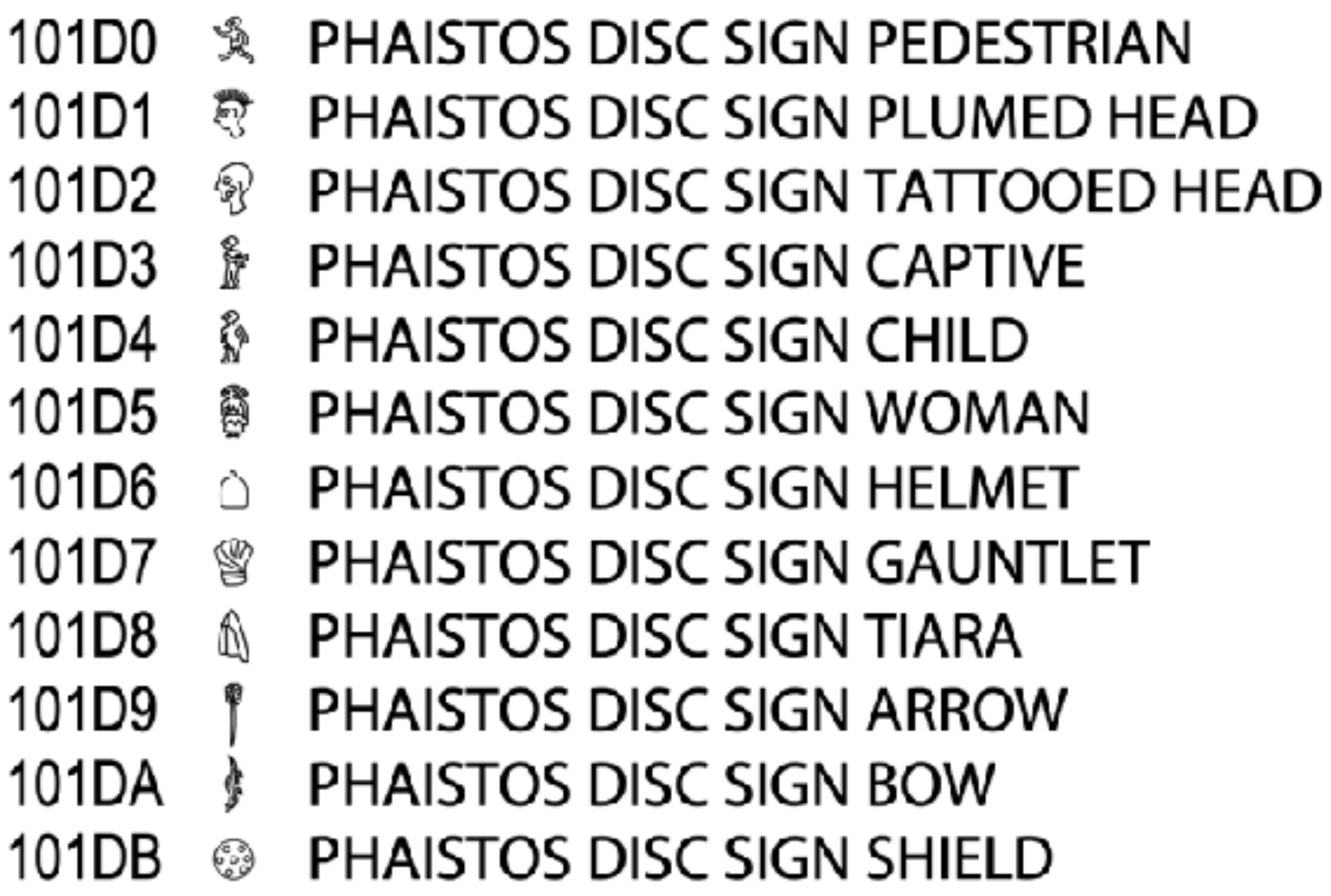
=> Use the simplest
(the most apriori probable)

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

Where does the **overfitting** come from?

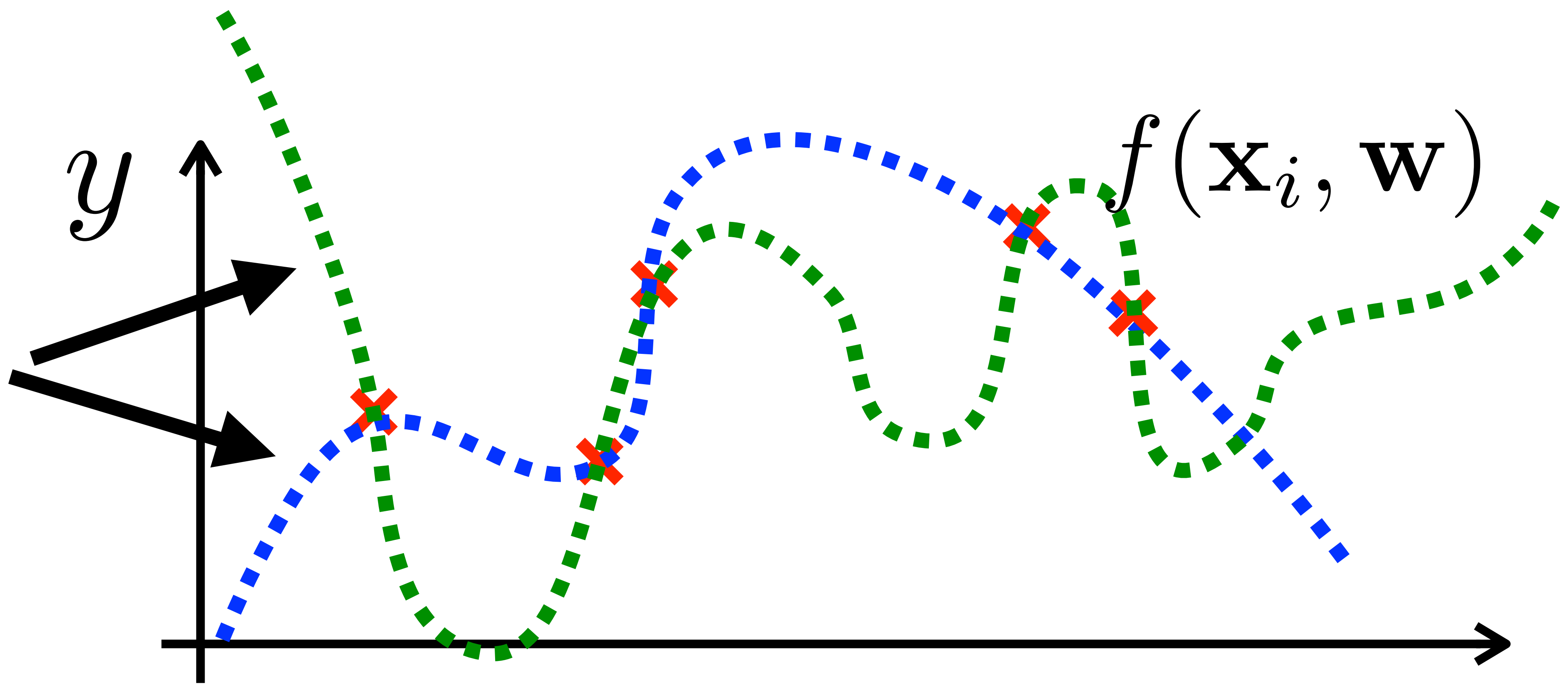
Phaistos disc

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

Many stories consistent with sequence of visual symbols



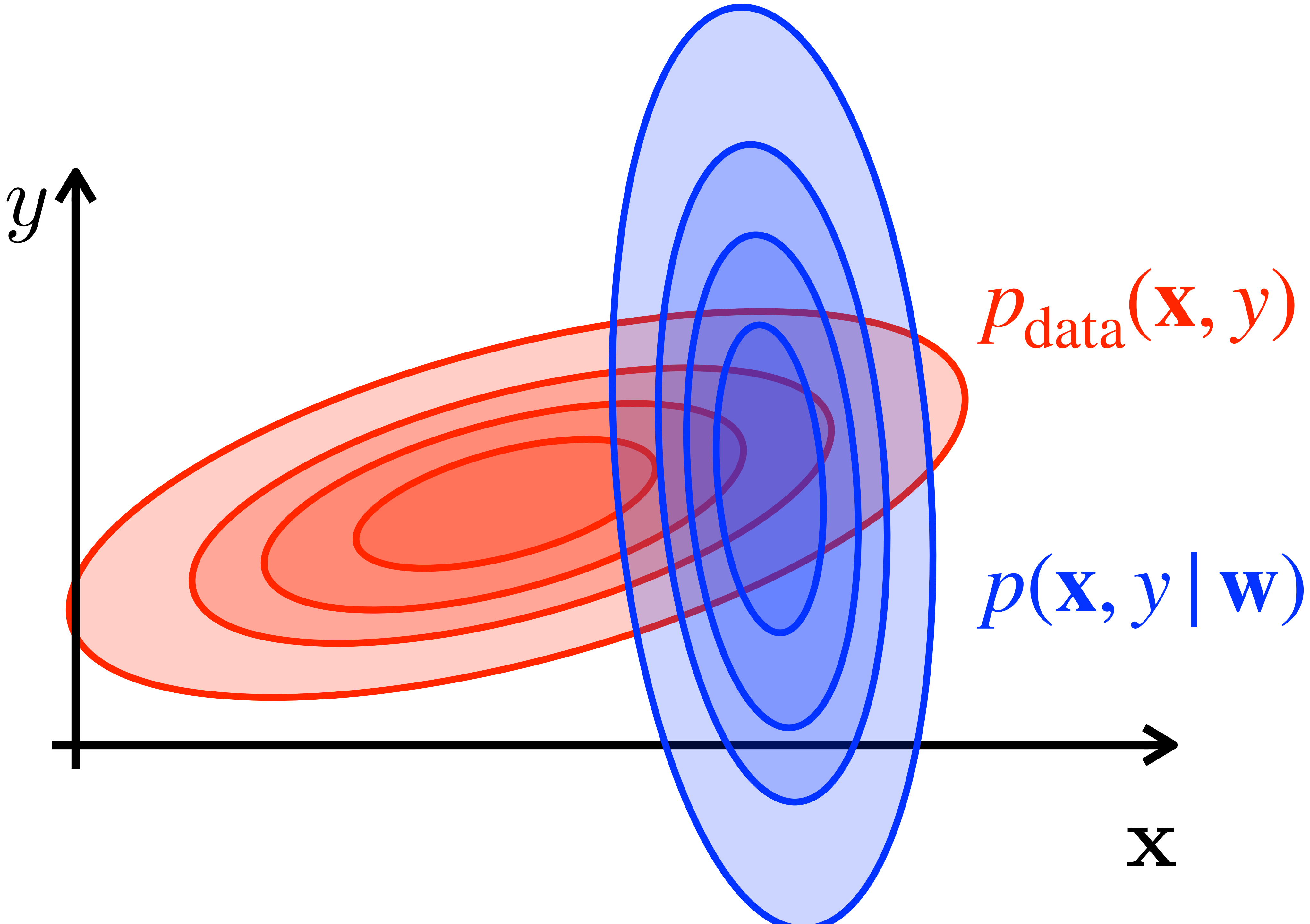
The space of possible stories is too wild

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

Where does the **overfitting** come from?

- We fit $p(\mathbf{x}, y \,|\, \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^{\star} = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \;\|\; p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$

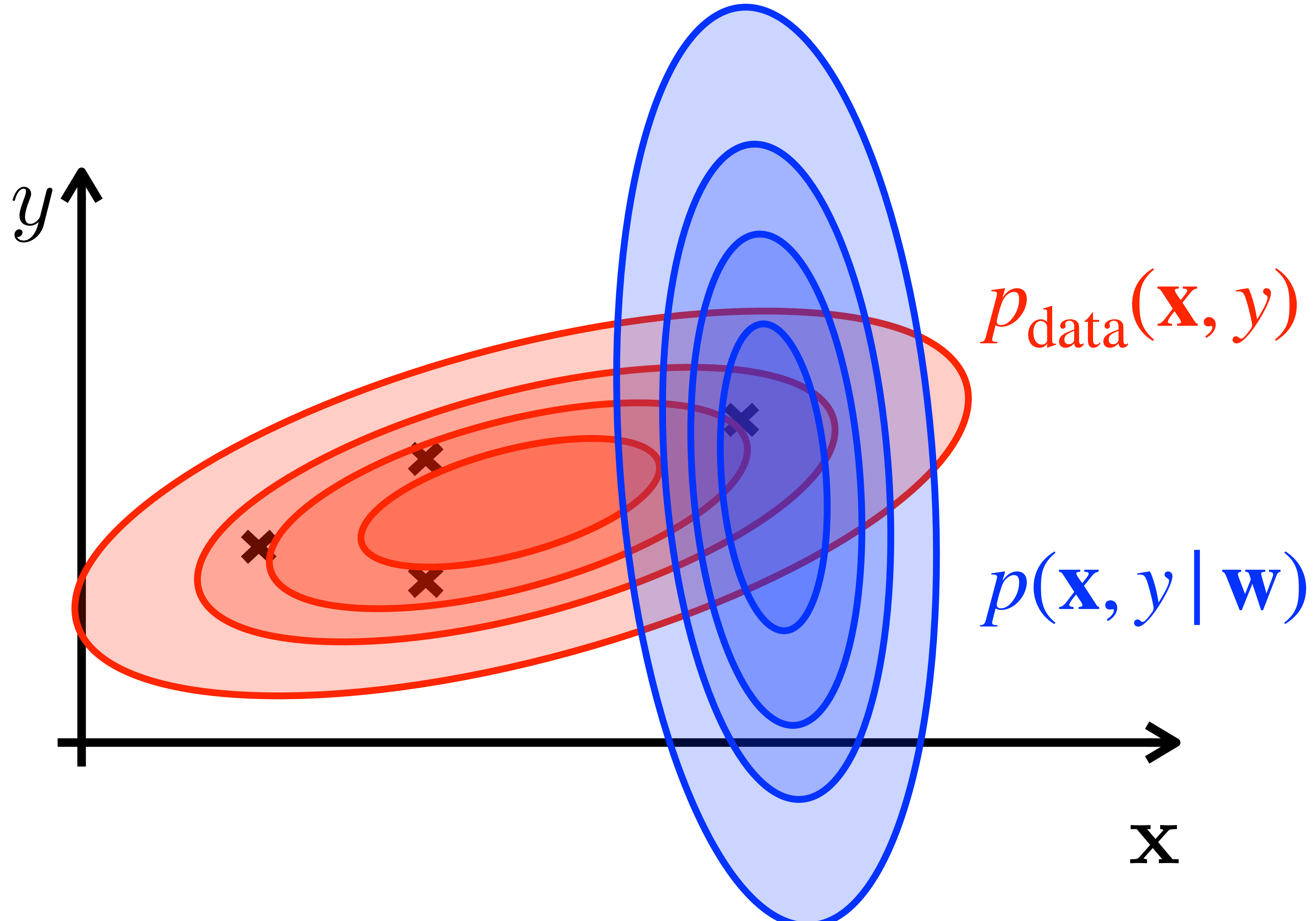


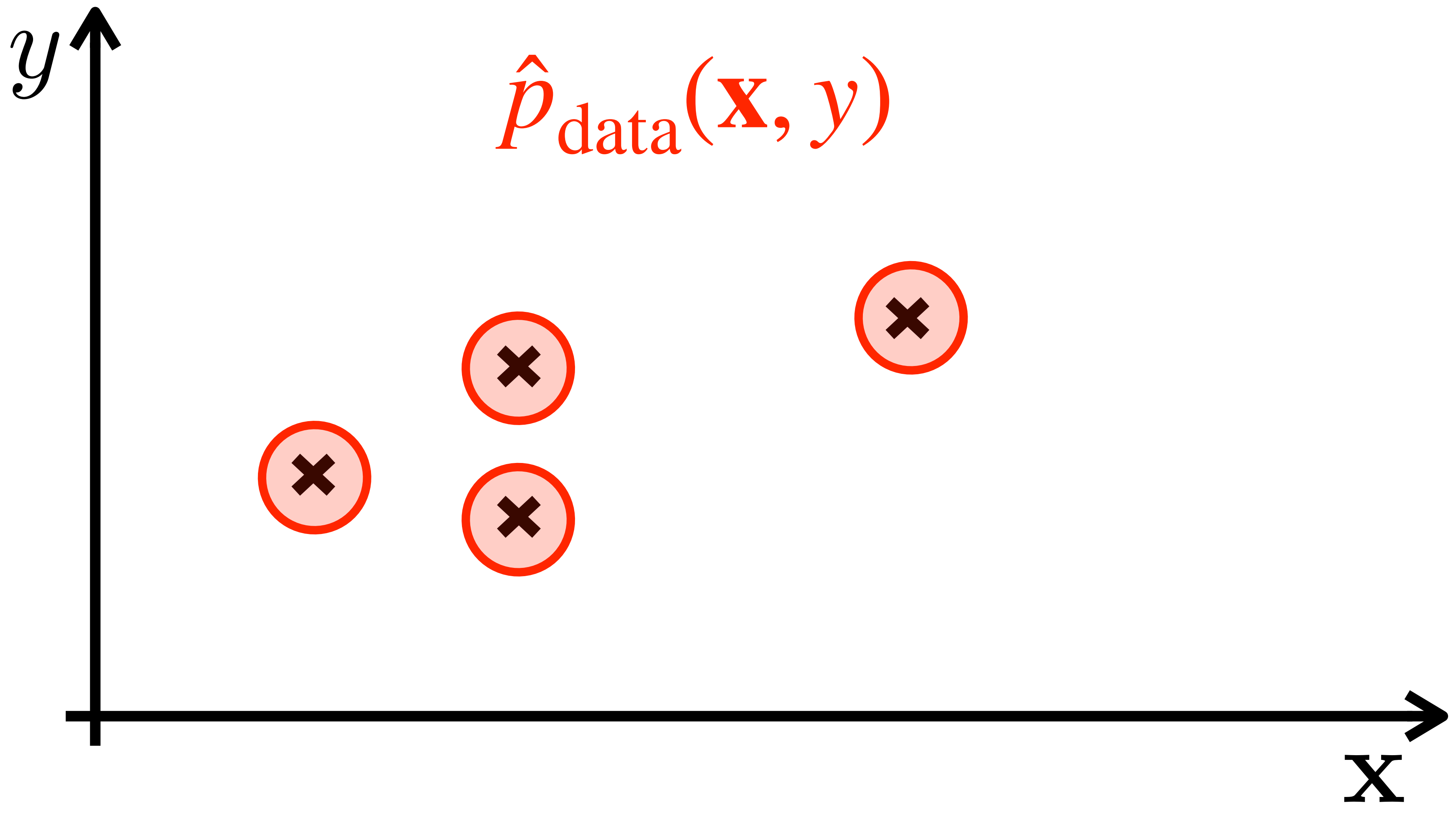
Where does the **overfitting** come from?

- We fit $p(\mathbf{x}, y \,|\, \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$

- $p_{\text{data}}(\mathbf{x}, y)$ is unknown => use samples (training set)
- Since the traning set is finite, we actually used different $\hat{p}_{\text{data}}(\mathbf{x}, y)$





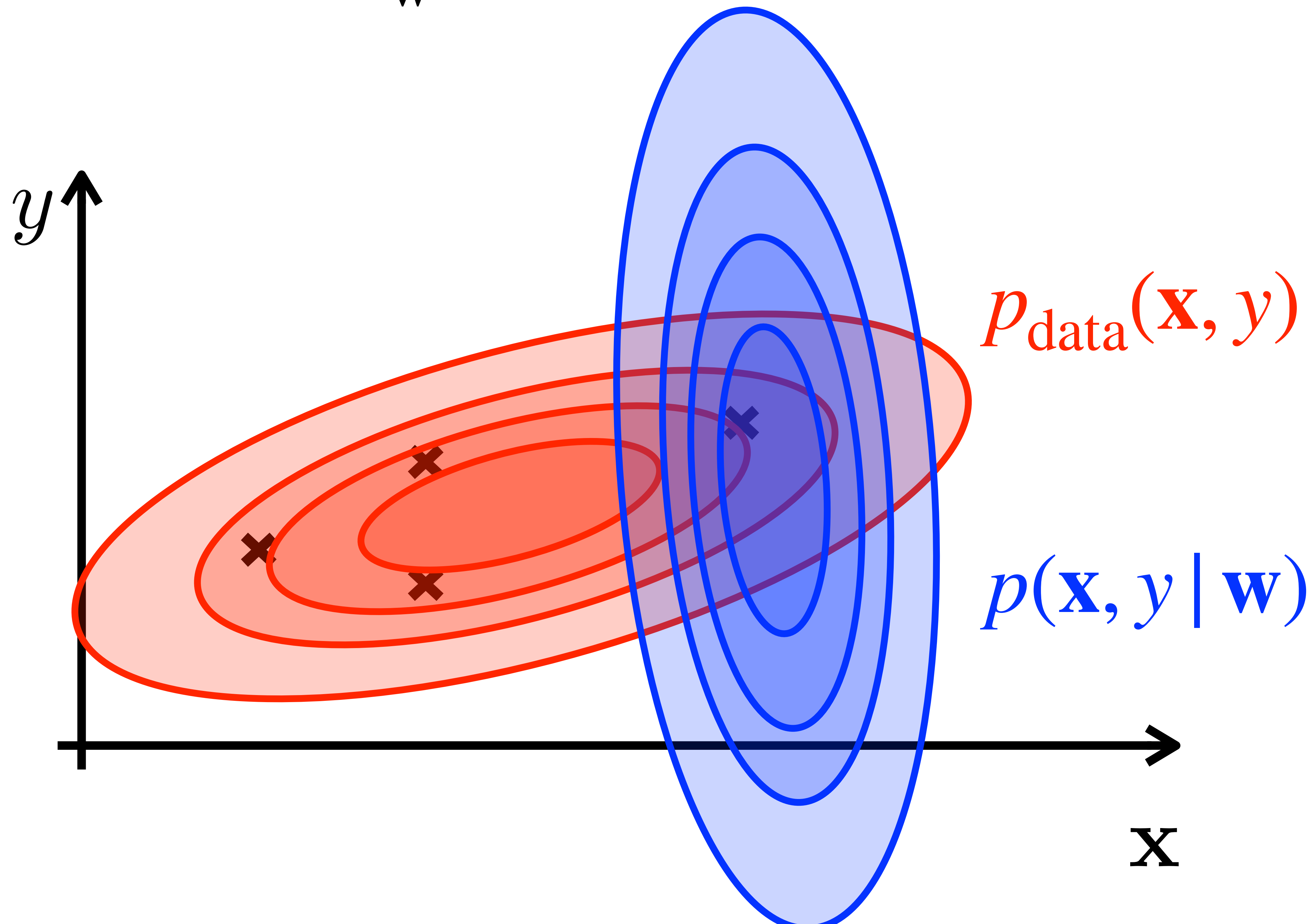
Where does the **overfitting** come from?

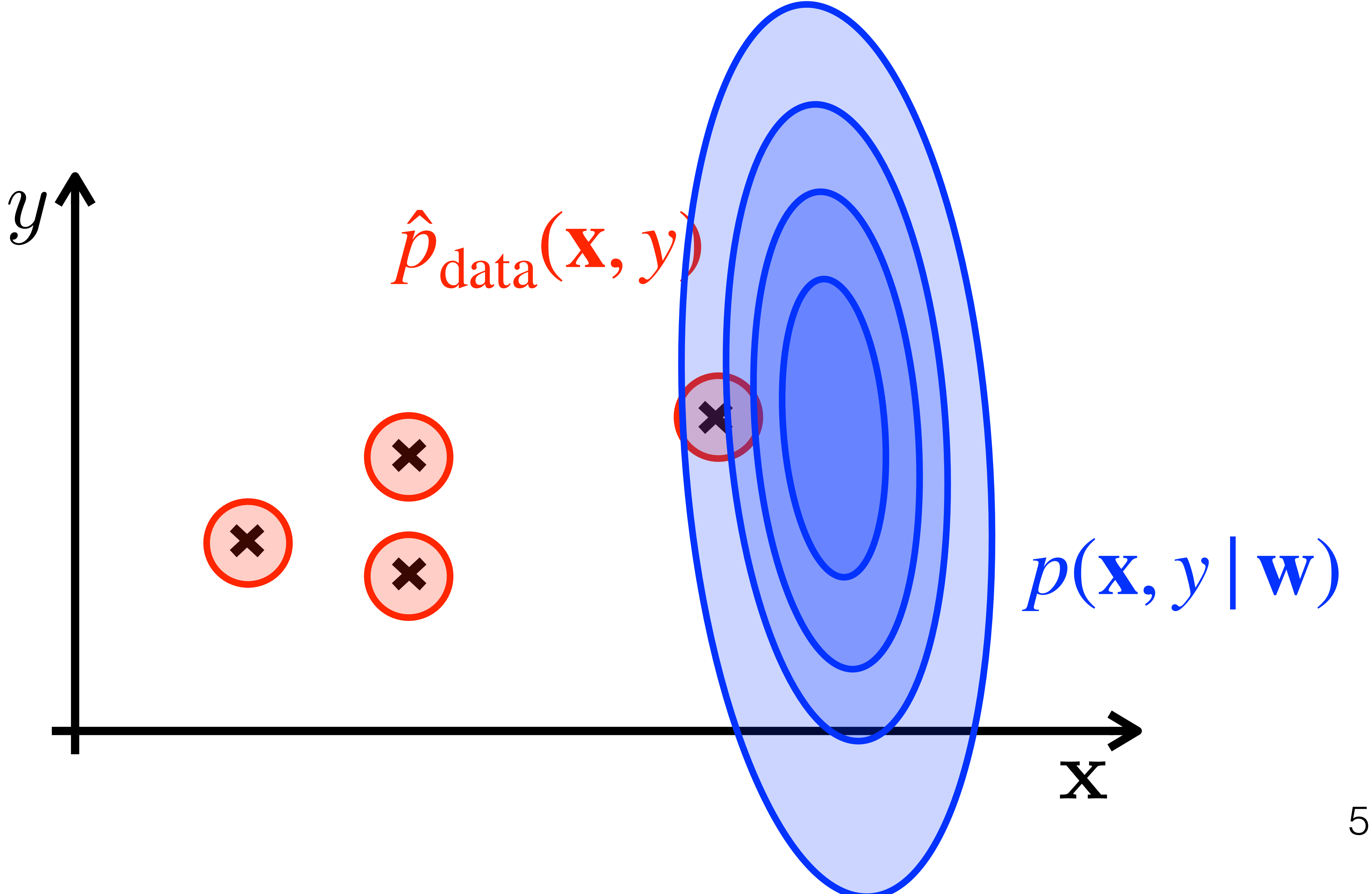
- We fit $p(\mathbf{x}, y \,|\, \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$

- $p_{\text{data}}(\mathbf{x}, y)$ is unknown => use samples (training set)
- Since the traning set is finite, we actually used different $\hat{p}_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$





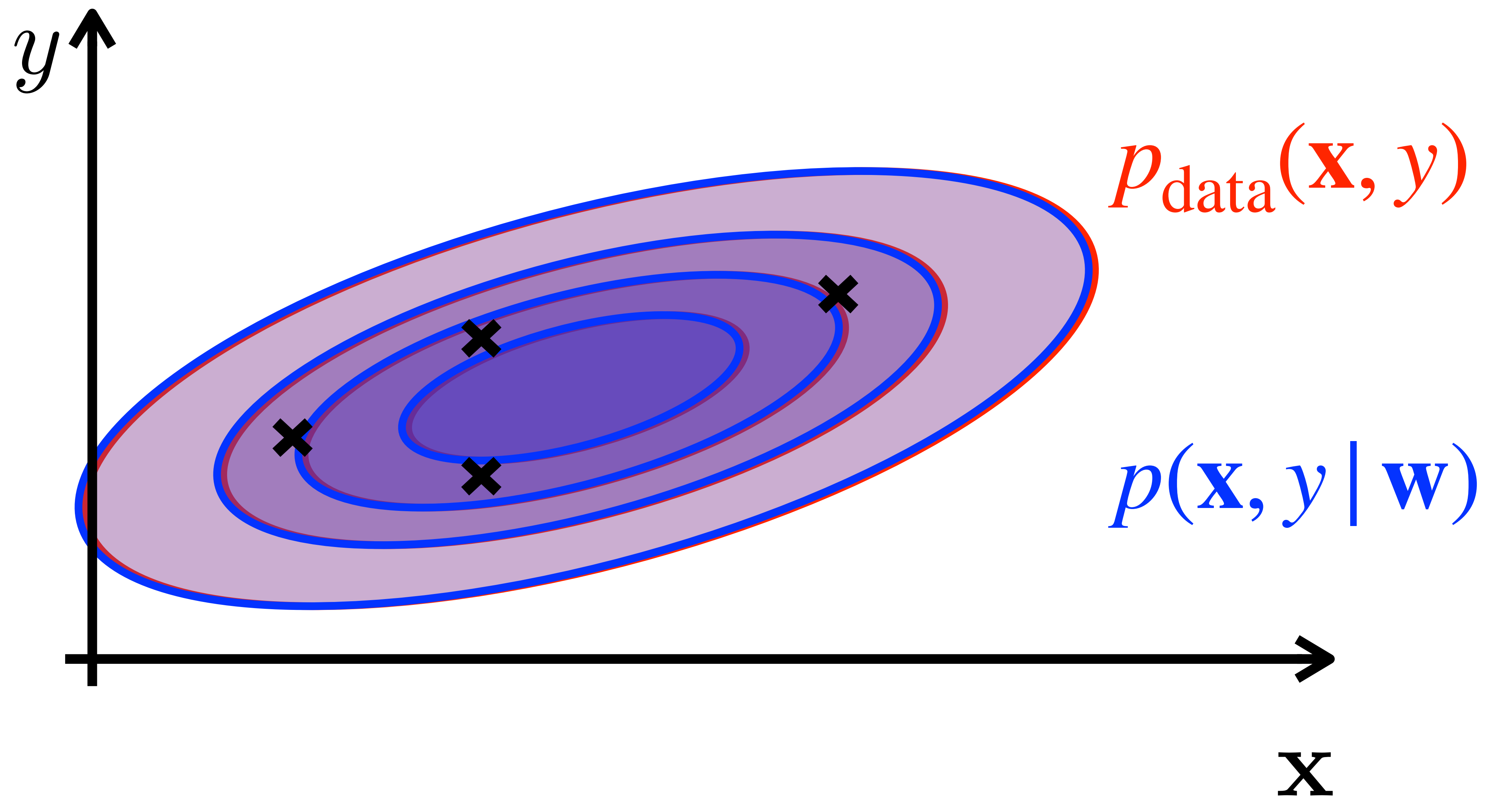
Where does the **overfitting** come from?

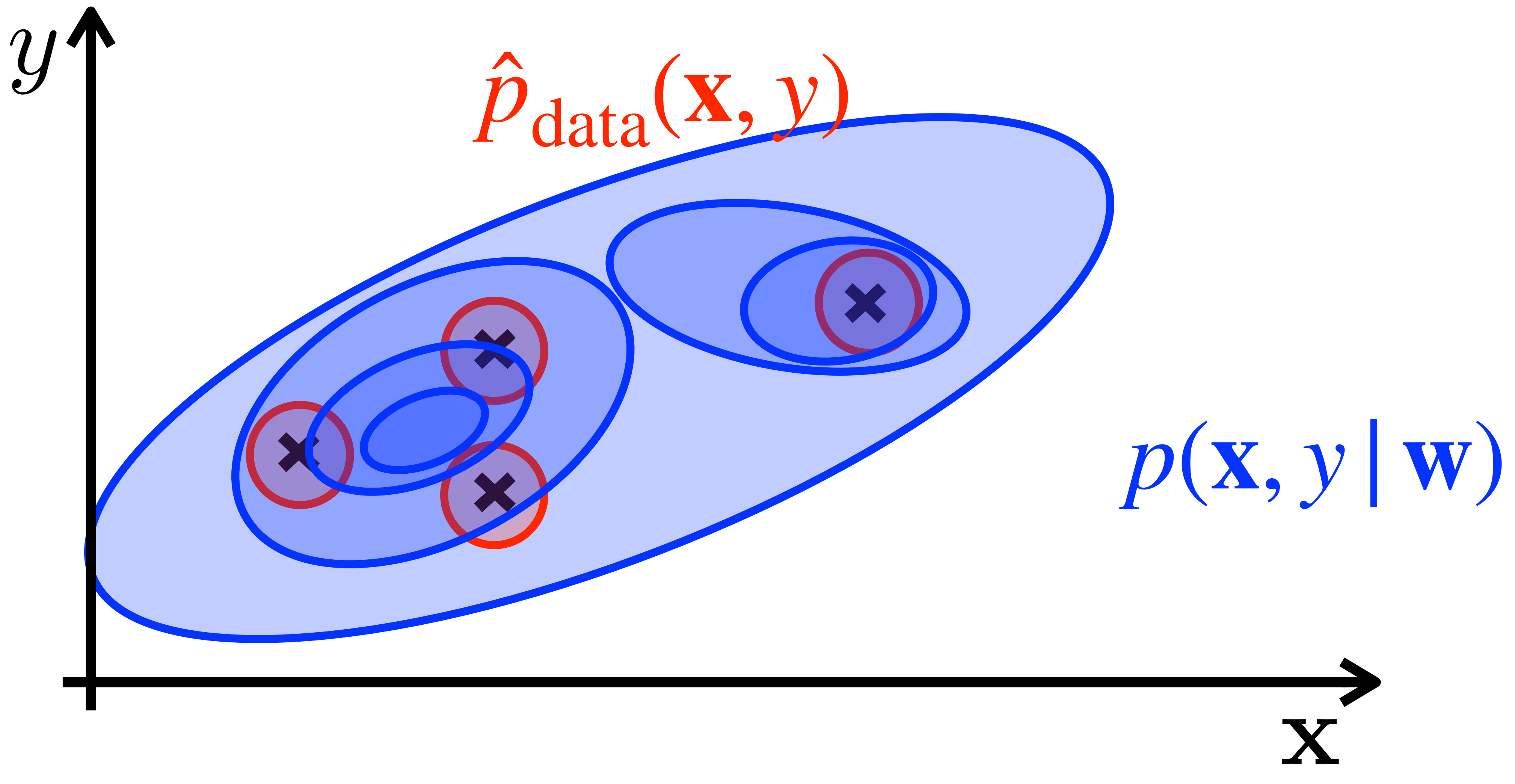
- We fit $p(\mathbf{x}, y \,|\, \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$

- $p_{\text{data}}(\mathbf{x}, y)$ is unknown => use samples (training set)
- Since the traning set is finite, we actually used different $\hat{p}_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$





Where does the **overfitting** come from?

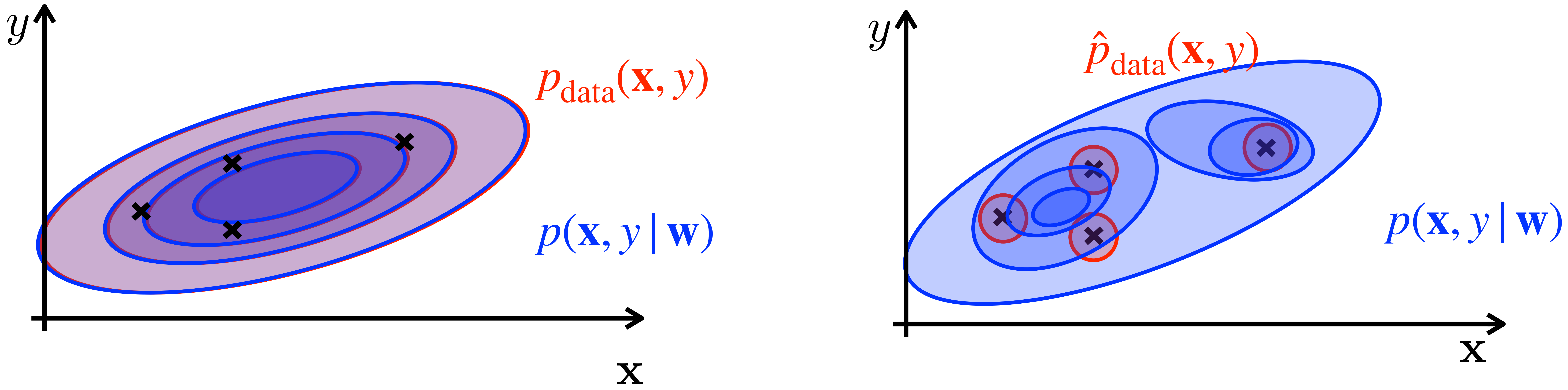
$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$

Take home message: **Optimization ≠ Machine learning**

Machine learning is optimization of the criterion, we do not have access to.

Therefore approximated criterion is optimized instead

Suppress overfitting = 1) Use the right $p(\mathbf{x}, y \,|\, \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$



Where does the **overfitting** come from?

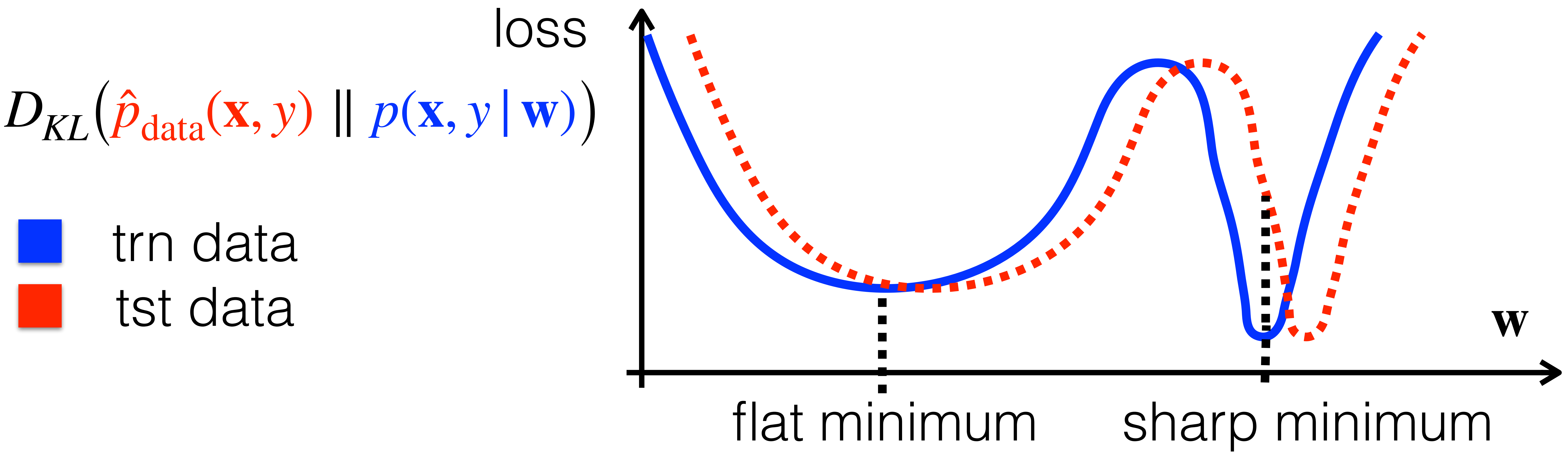
$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \mid \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \mid \mathbf{w})\big)$$

Take home message: **Optimization ≠ Machine learning**

Machine learning is optimization of the criterion, we do not have access to.

Therefore approximated criterion is optimized instead

Suppress overfitting =
1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$
2) **Which one is better???**



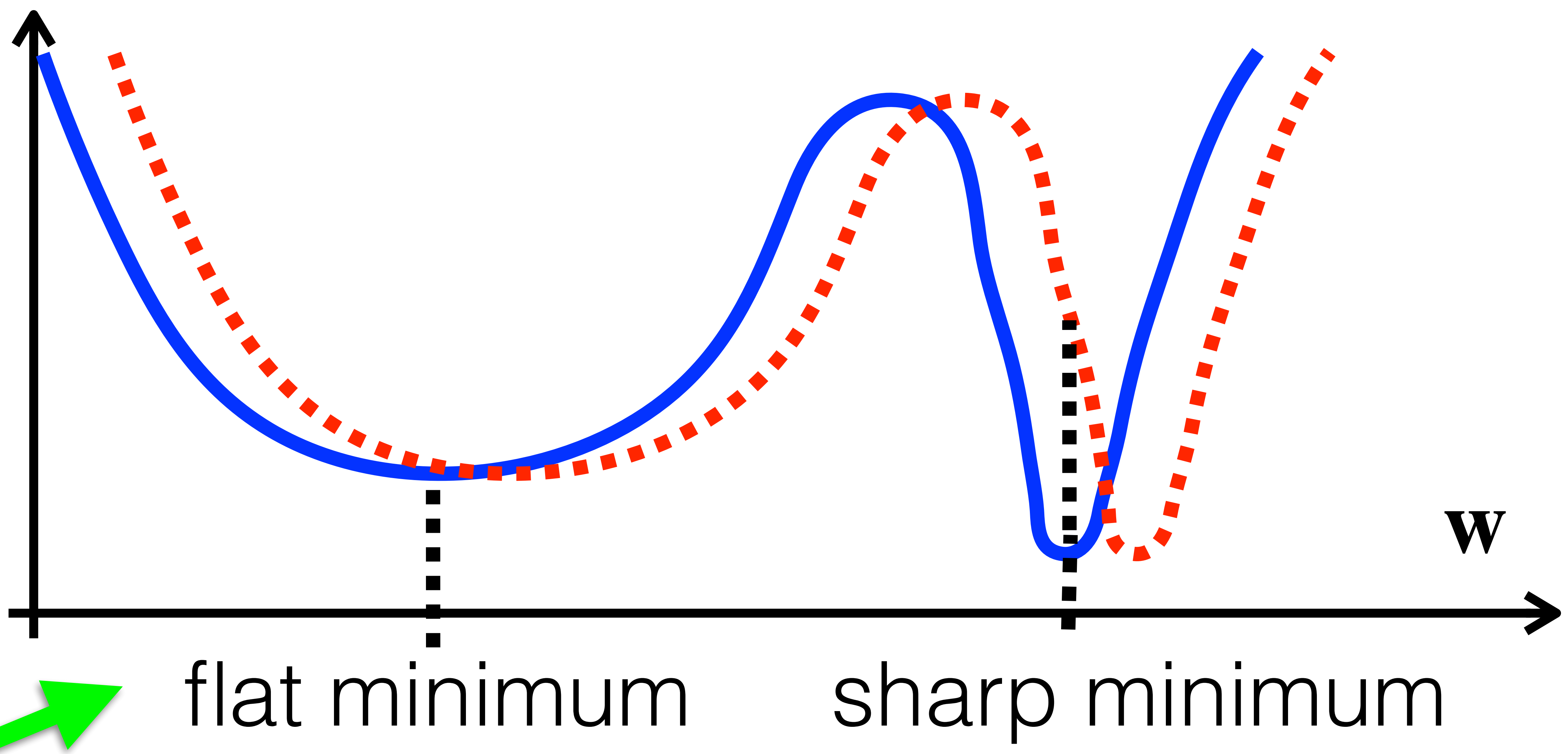
Where does the **overfitting** come from?

2) **Which one is better???**

$$D_{KL}\left(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \mid \mathbf{w})\right)$$

🟦 trn data

🟥 tst data



Good generalization

Testing error remains small

Weak generalization => optimum prone to overfitting

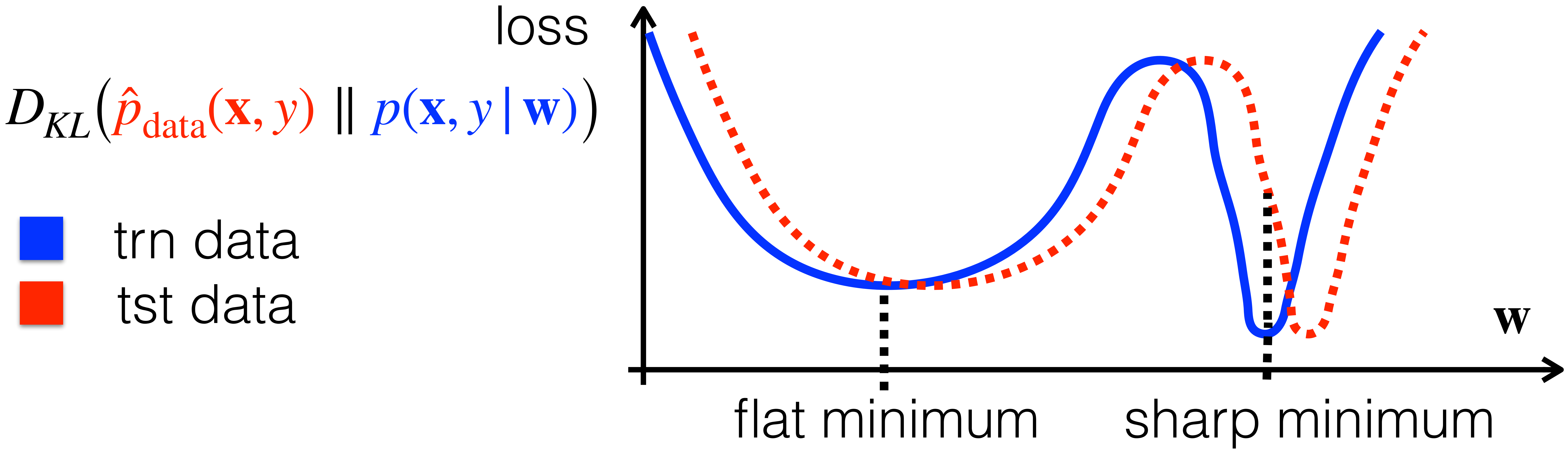
Testing error grows fast with a small trn/tst shift

Weaker learning methods are surprisingly better in generalization

[Dai, NIPS, 2018]   https://arxiv.org/pdf/1812.00542.pdf

Where does the **overfitting** come from?

2) Avoid sharp minima of $D_{KL}\left(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y \mid \mathbf{w})\right)$



Can you guess how to enforce flat minimum???

$$\min_{w} \max_{\|\epsilon\|_2 \leq \rho} L_{train}(w + \epsilon)$$

[Foret 2021]
https://arxiv.org/pdf/2106.01548.pdf

Where does the **overfitting** come from?

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$$

Take home message: **Optimization ≠ Machine learning**

Machine learning is optimization of the criterion, we do not have access to.

Therefore approximated criterion is optimized instead

Suppress overfitting =

1) Use the right $p(\mathbf{x}, y \,|\, \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

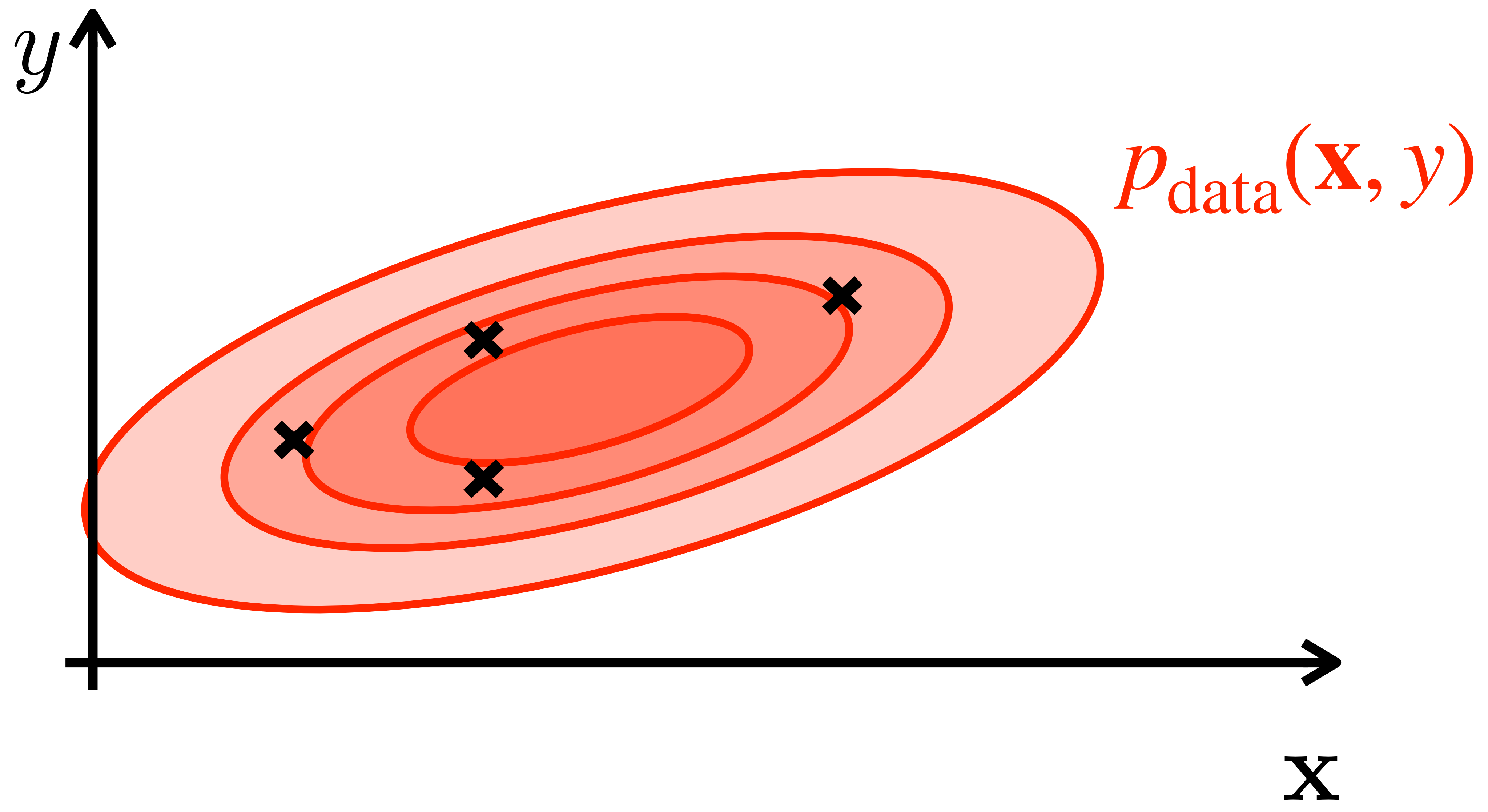
2) Avoid sharp minima of $D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big)$

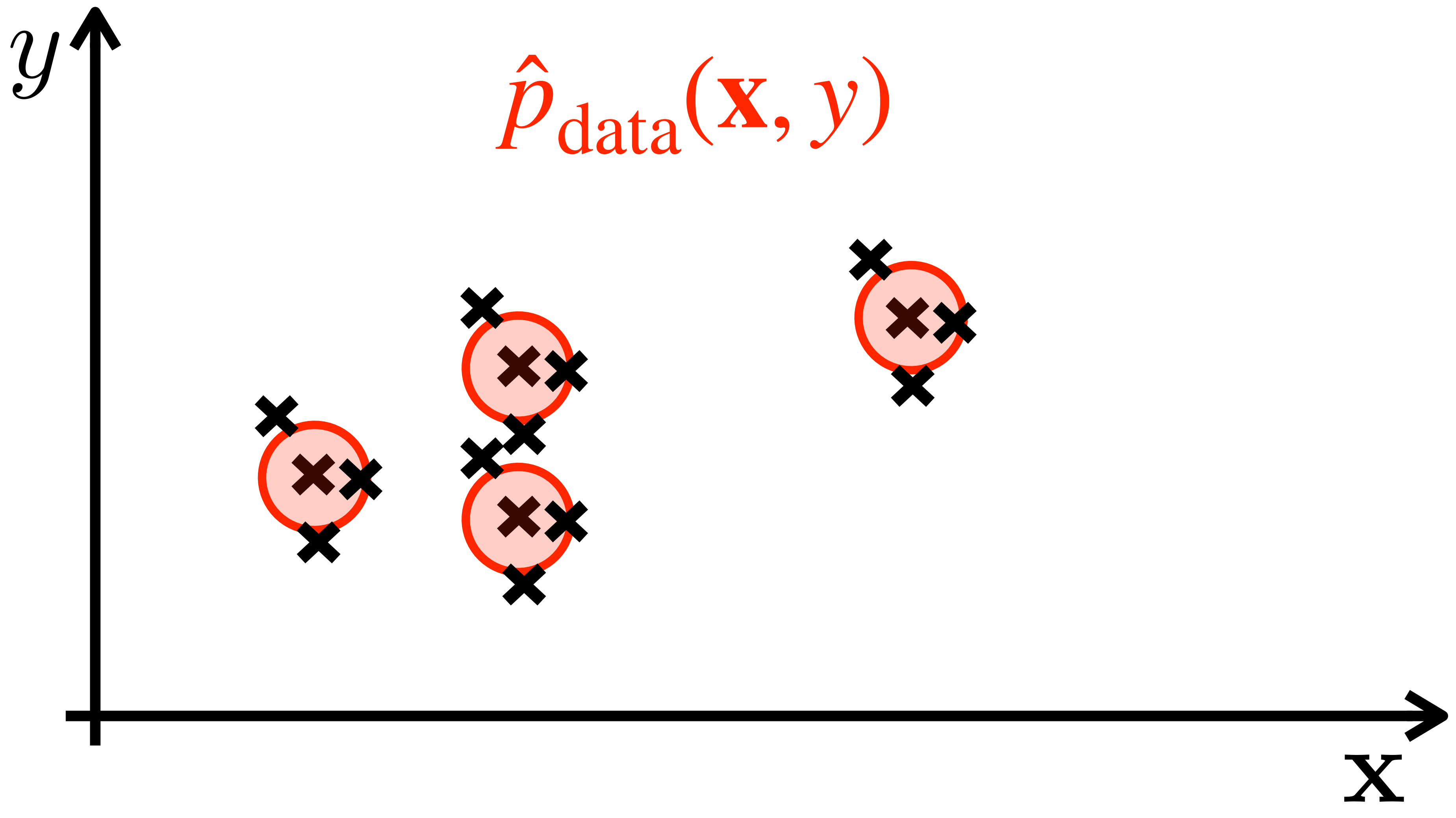
3) Use close-to-infinite dataset

Where does the **overfitting** come from?

3) Use close-to-infinite dataset

Data augmentation





Where does the **overfitting** come from?

3) Use close-to-infinite dataset

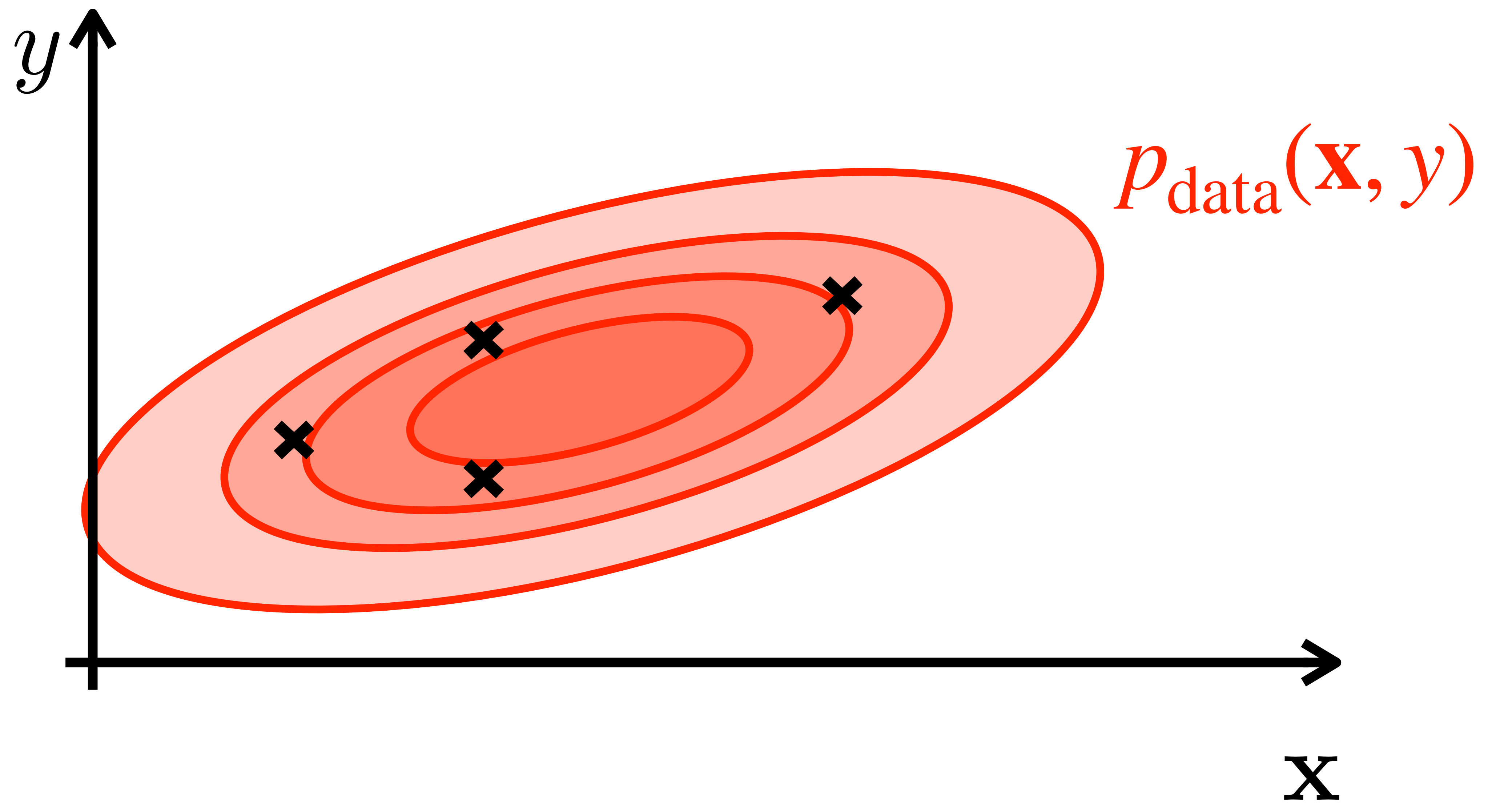
Data augmentation

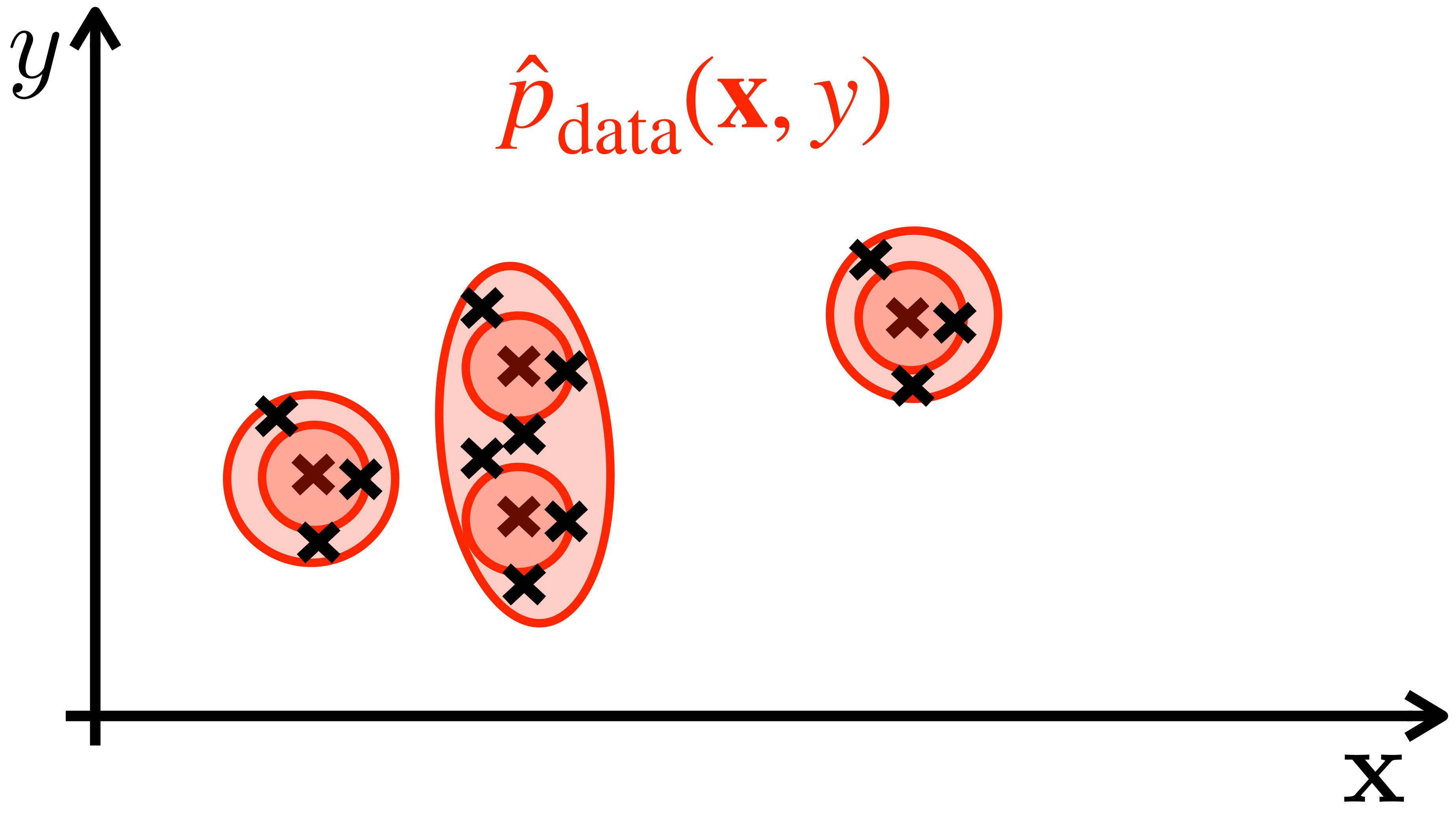
Which fake data can you generate???

Reasonable geometrical and histogram transformations:

Mirroring, scaling, rotation, squeezing, contrast, brightness, …

Or any generative model





Where does the **overfitting** come from?

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \mid \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \mid \mathbf{w})\big)$$

Take home message: **Optimization ≠ Machine learning**

Machine learning is optimization of the criterion, we do not have access to.

Therefore approximated criterion is optimized instead

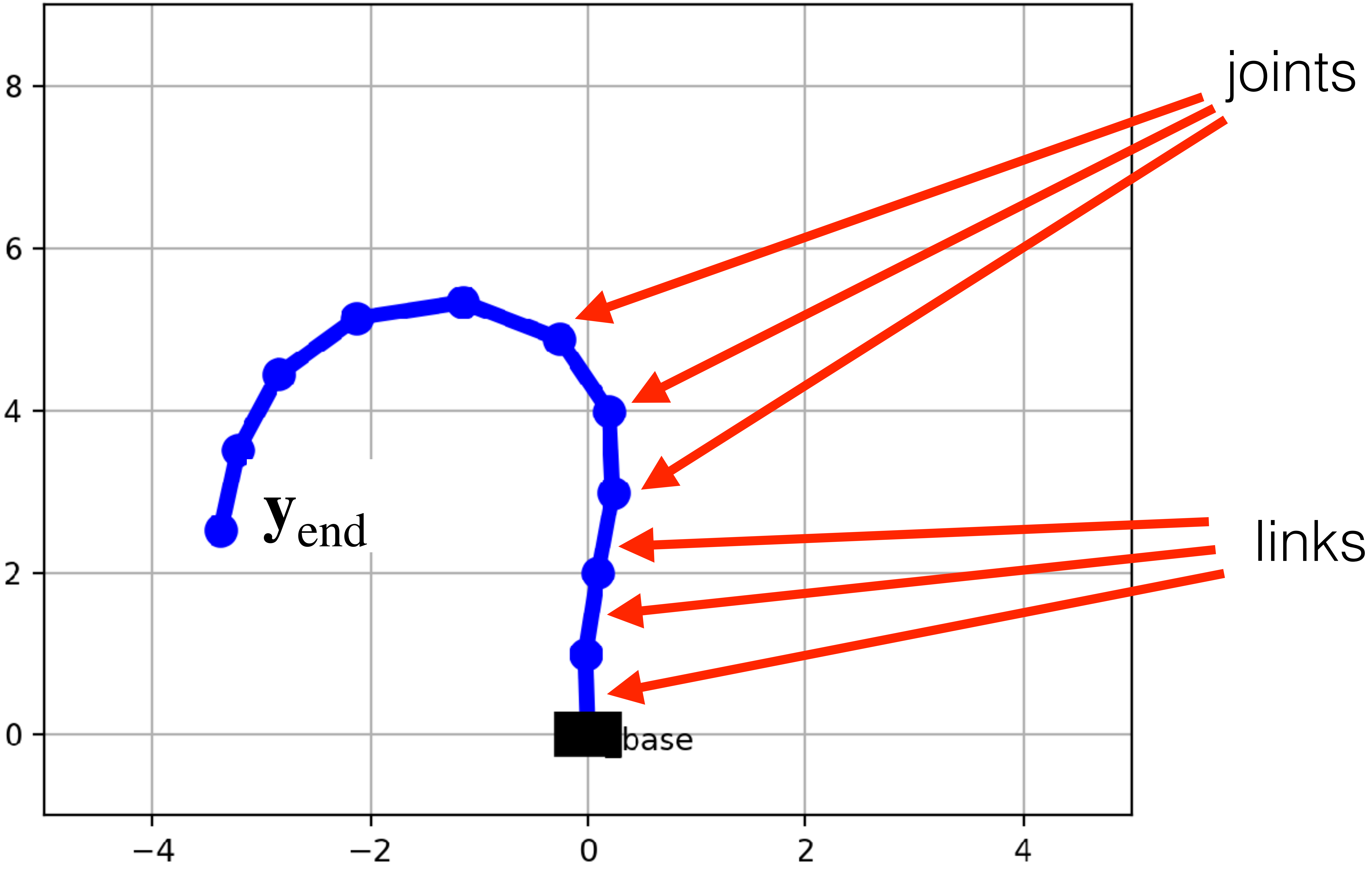
Suppress overfitting =

1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$
2) Avoid sharp minima of $D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \mid \mathbf{w})\big)$
3) Use close-to-infinite dataset

1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

2D manipulator

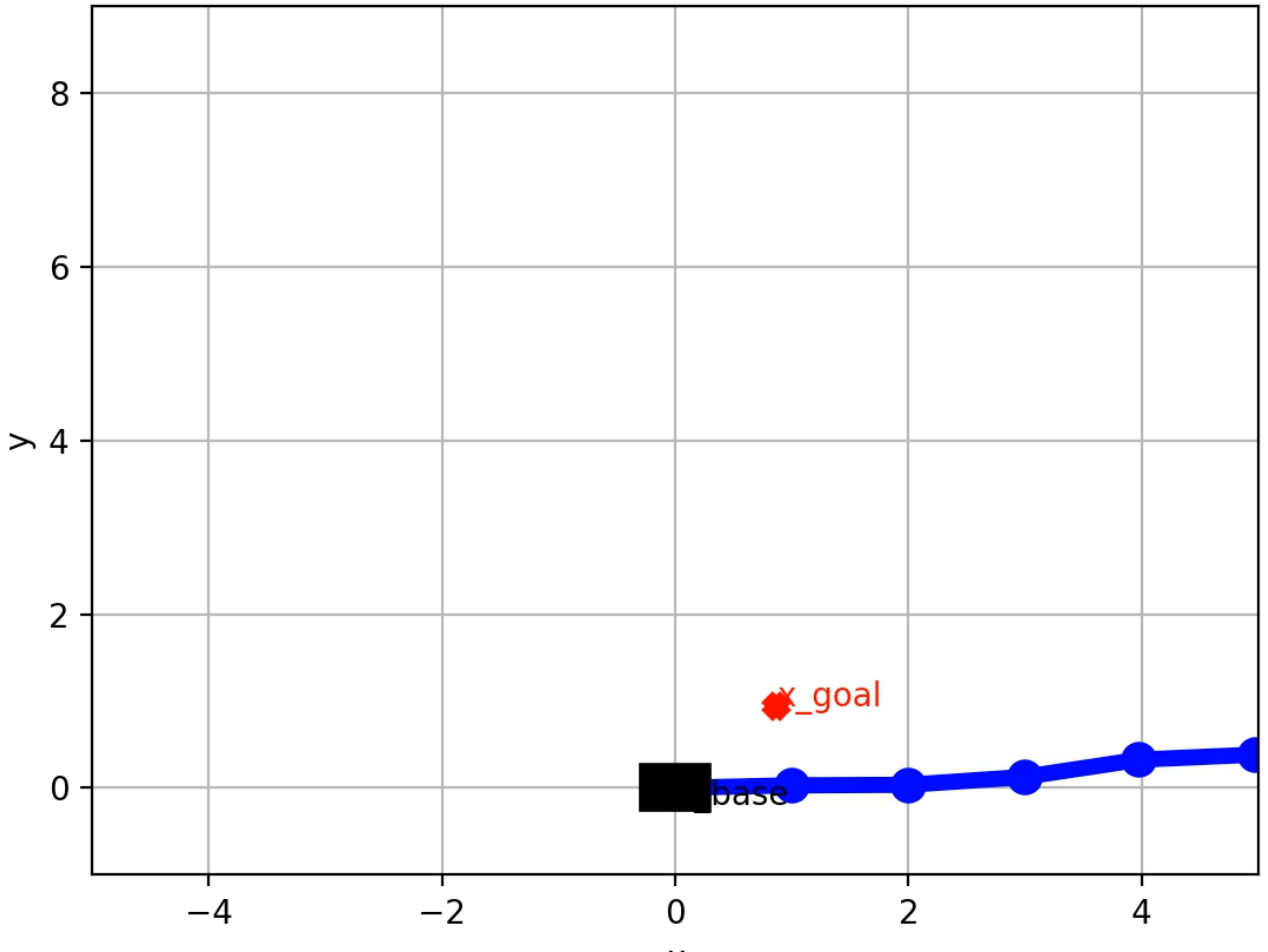
HW1 IKT: (links + end_effector) => joints



1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

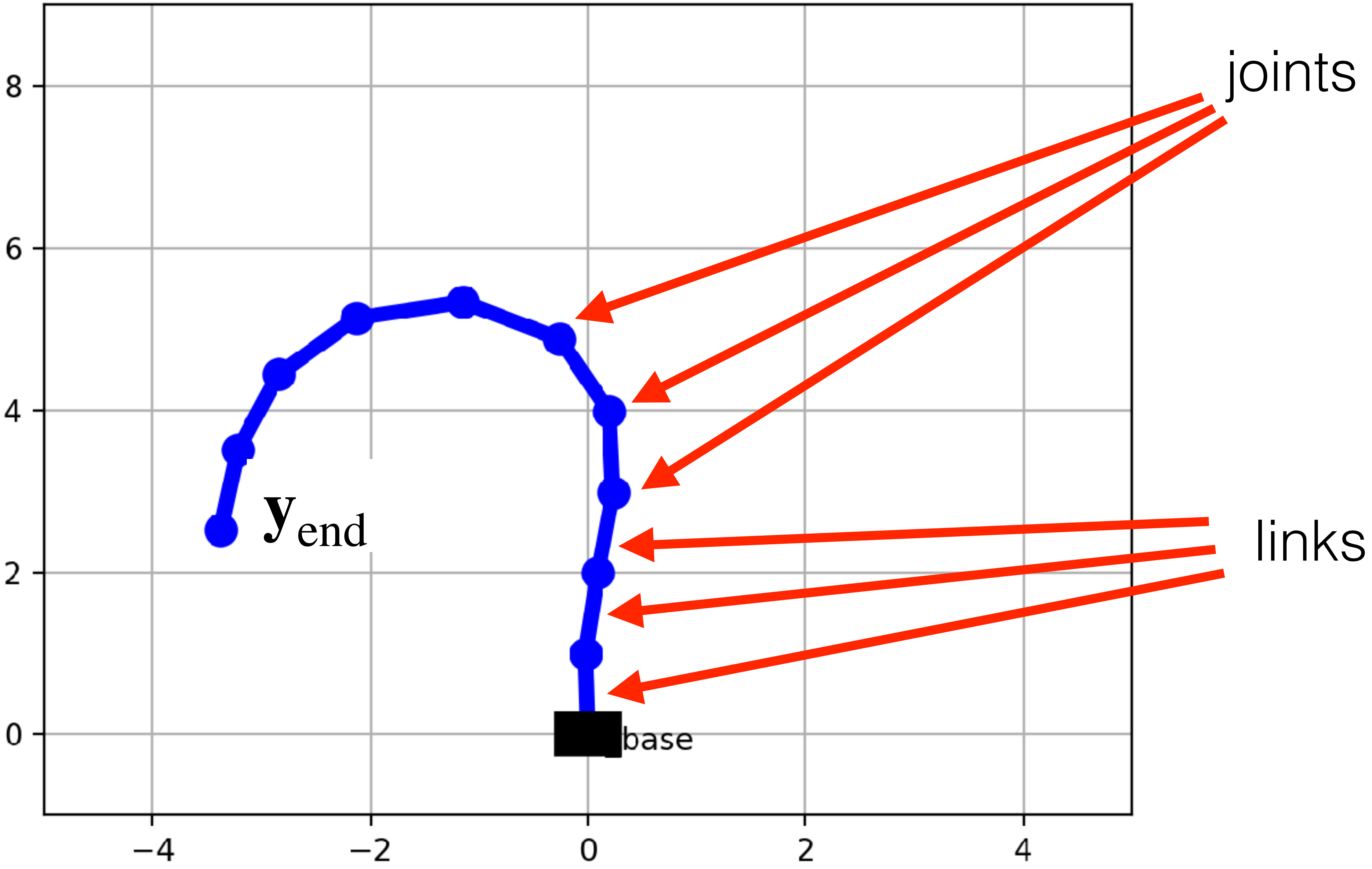
2D manipulator

HW1 IKT: (links + end_effector) => joints



1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

model: $p(y \mid \mathbf{x}, \mathbf{w})$= **???**



1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

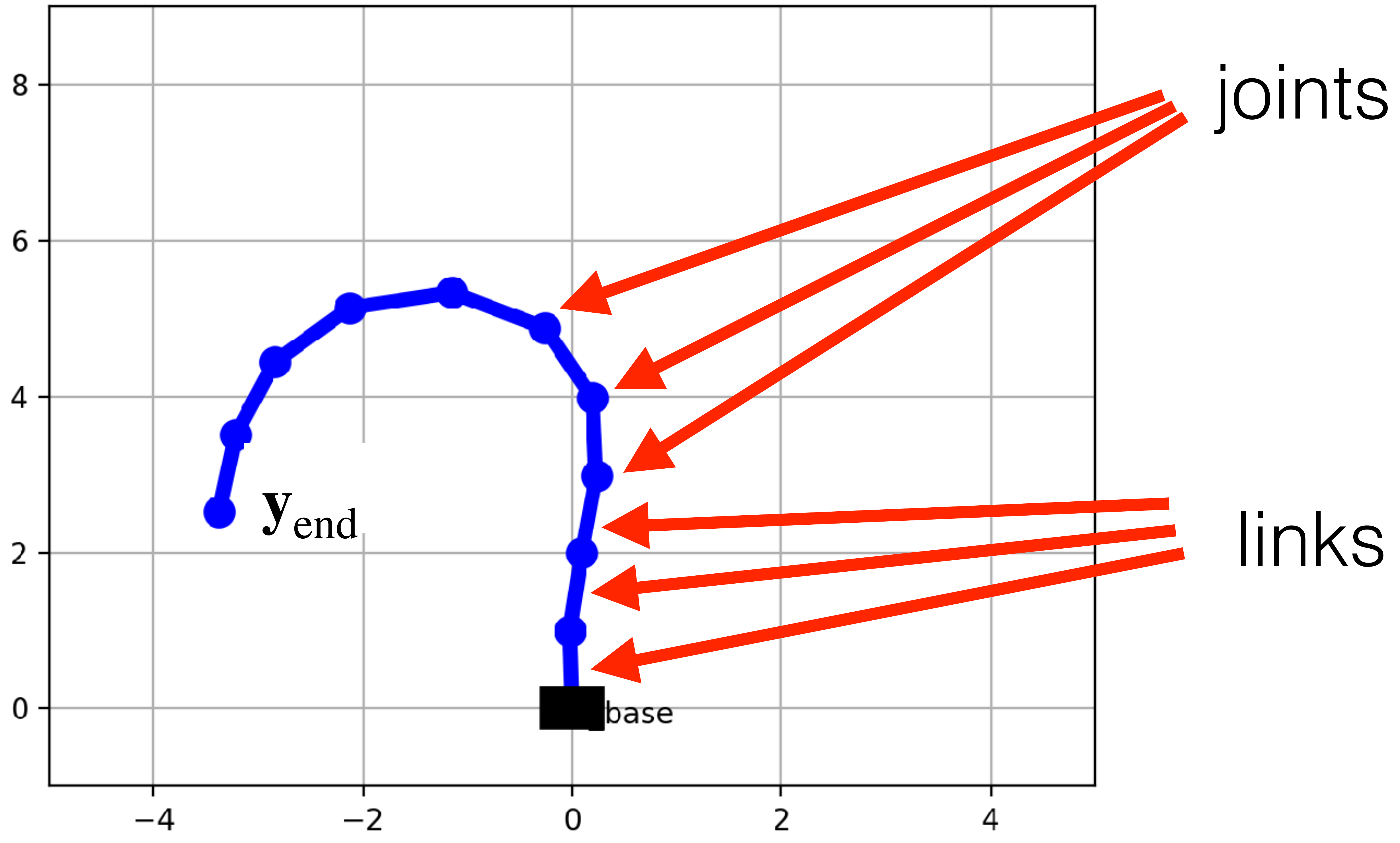
model: $p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(y_{\text{end}};\ f(\mathbf{x}, \mathbf{w}), \sigma)$

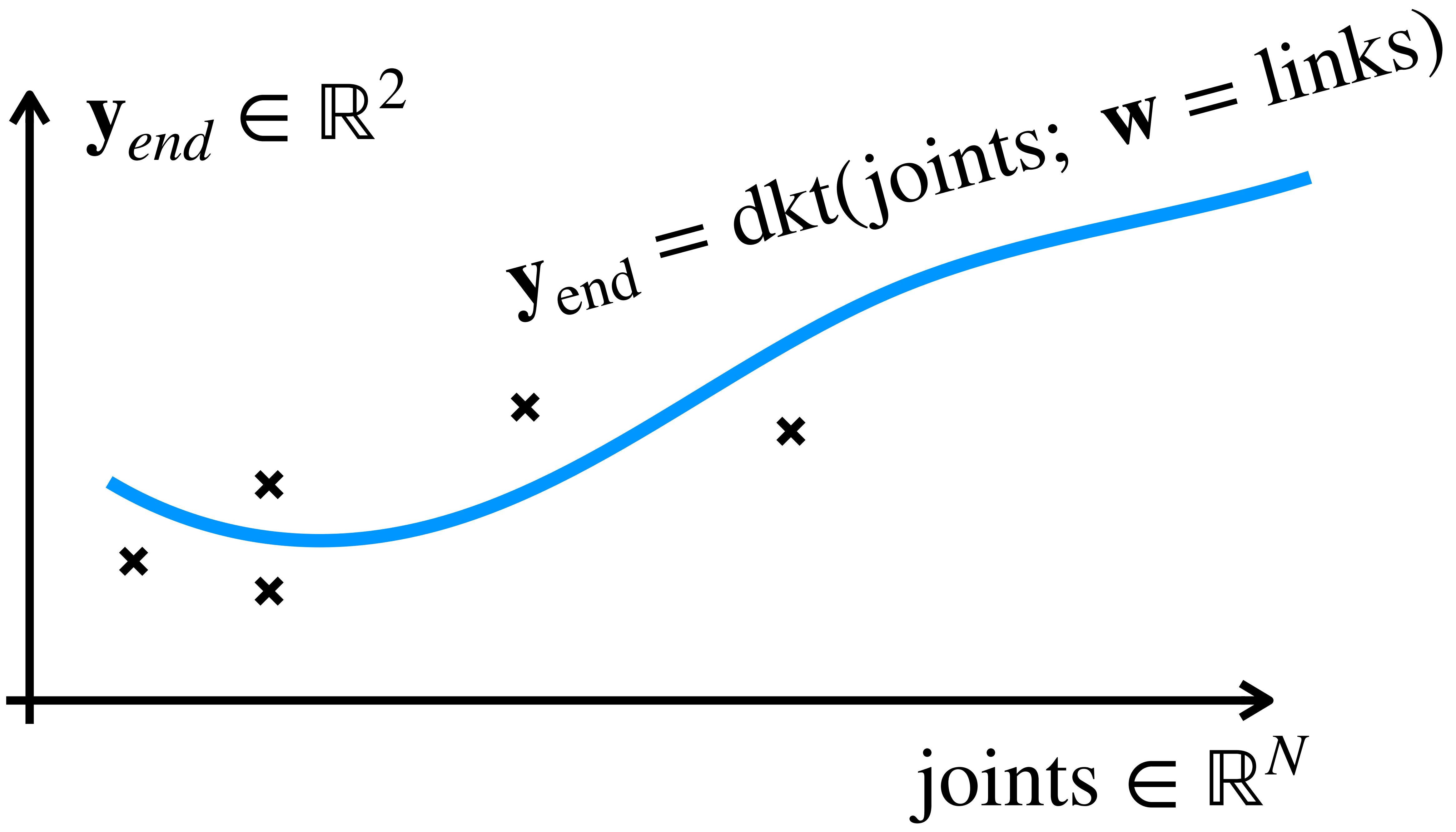
What is the right $f(\mathbf{x}, \mathbf{w})$?

(a) linear function $\mathbf{y}_{\text{end}} = f(\text{joints}; \mathbf{W}) = \mathbf{W} \cdot \text{joints}$ (underfit)

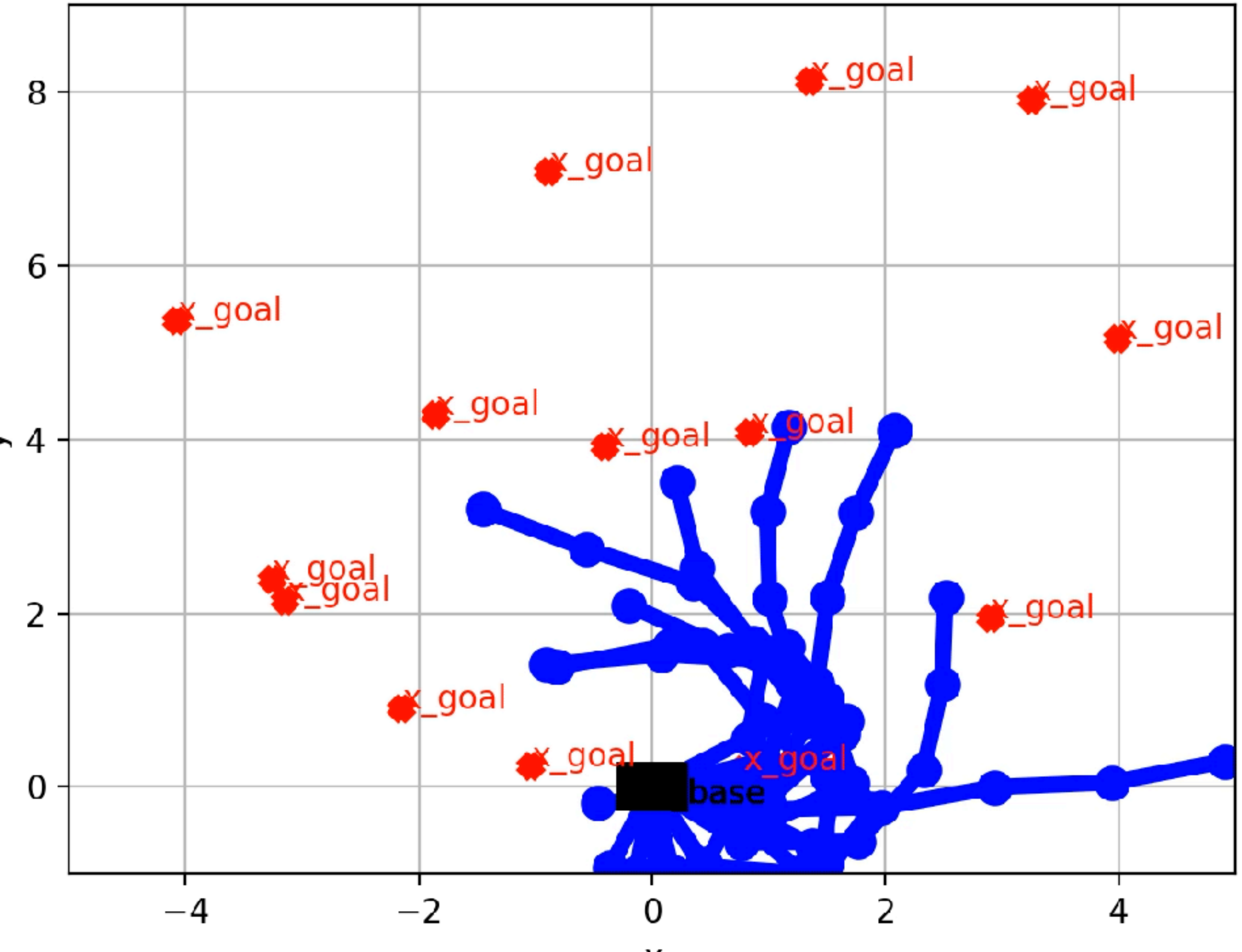
(b) deep ConvNet $\mathbf{y}_{\text{end}} = f(\text{joints}; \mathbf{w})$ (overfit)

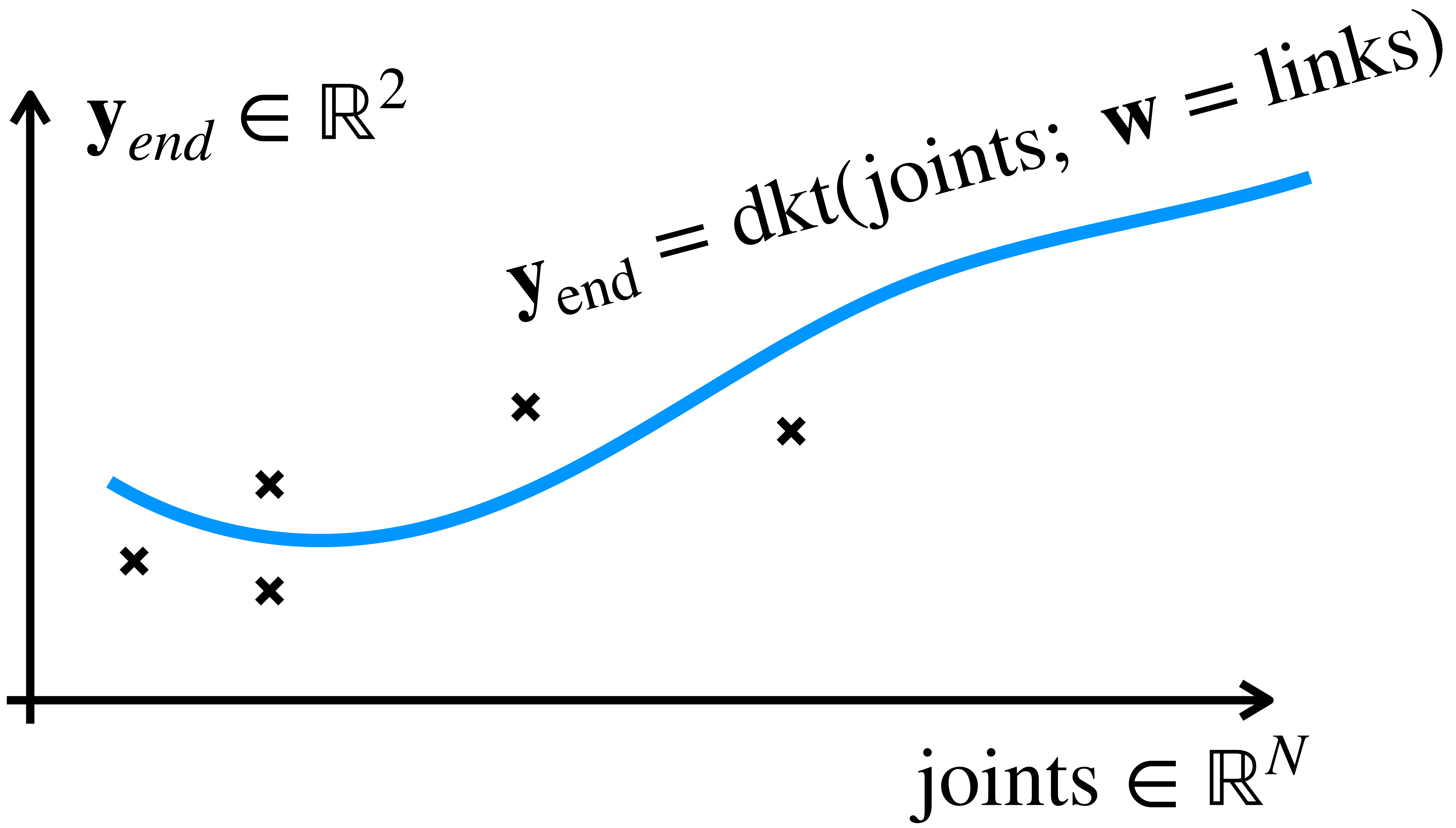
(c) DKT: $\mathbf{y}_{\text{end}} = \text{dkt}(\text{joints};\ \ \mathbf{w} = \text{links})$ (well-justified model)





1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$





1) Use the right $p(\mathbf{x}, y \,|\, \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

Take home message: **Always use the right tool/model**

Embed prior knowledge (physics, geometry, biology) about the problem into the network architecture

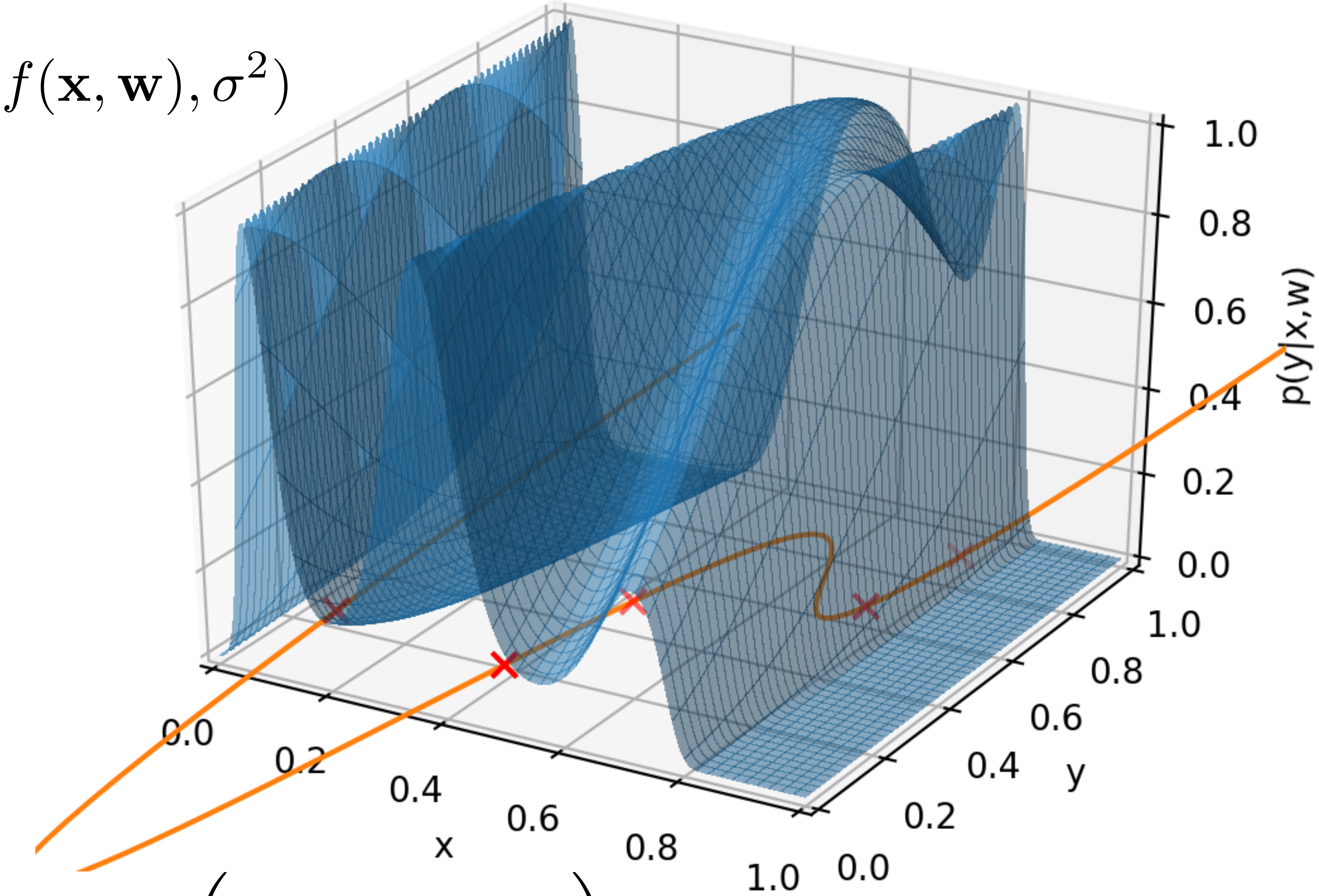
Examples:

- 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, ballistic trajectory
- Structure of animal cortex (for ConvNets)

If you cannot do it, at least penalize wild solutions

1) Use the right $p(\mathbf{x}, y \,|\, \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

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



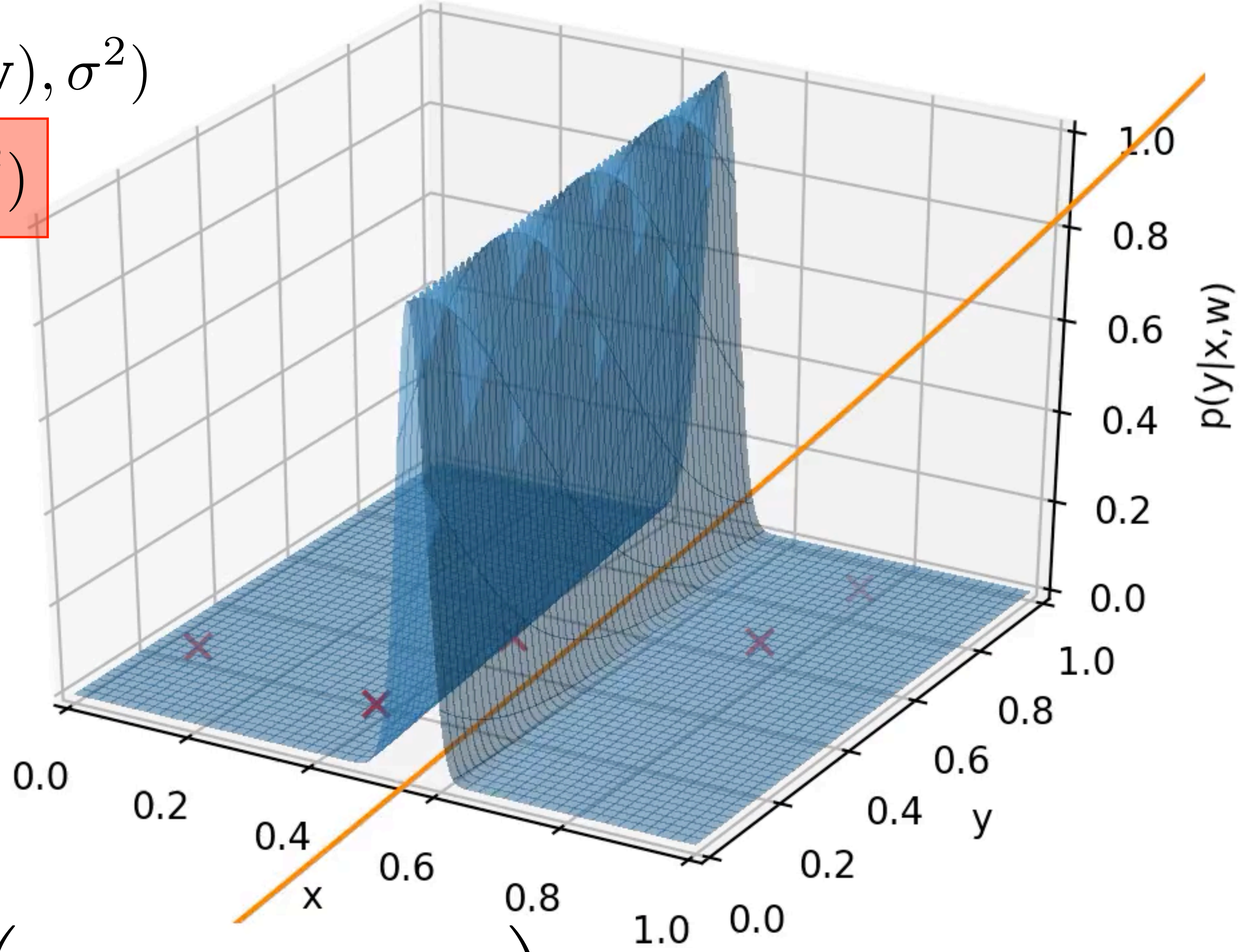
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

1) Use the right $p(\mathbf{x}, y \,|\, \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

What prior/regularizer should I use?

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

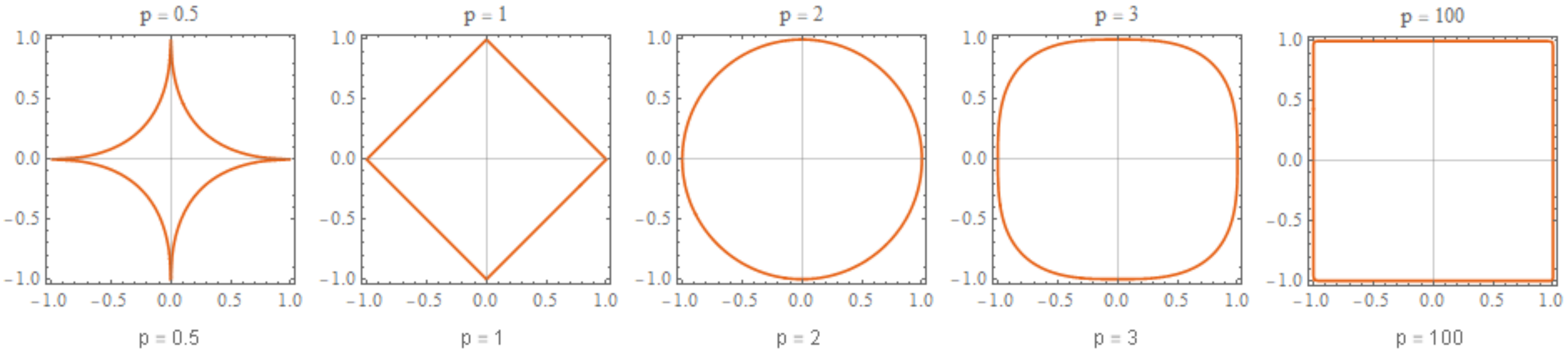
$$\boxed{p(\mathbf{w}) \sim \mathcal{N}_w(\mathbf{0},\ \sigma^2)}$$

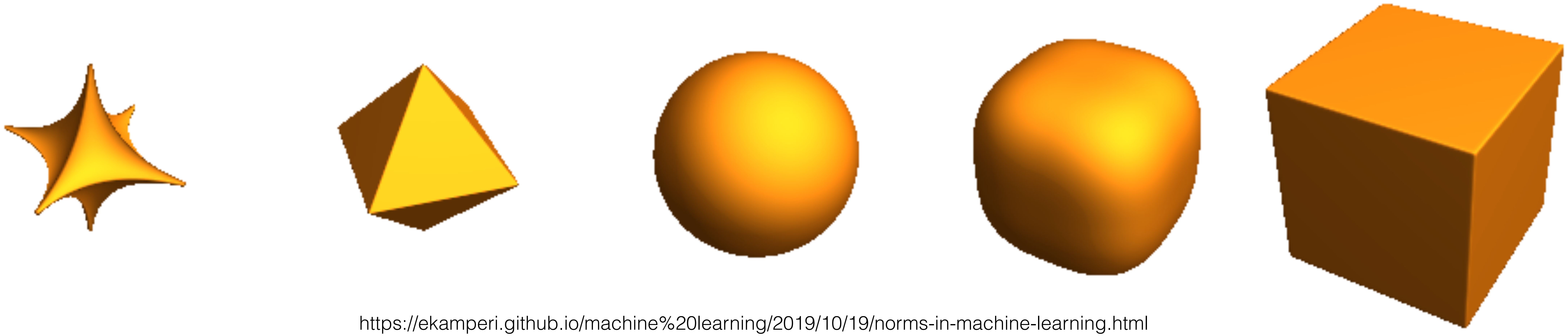


$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \boxed{p(\mathbf{w})} \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2 + \boxed{\|\mathbf{w}\|}$$

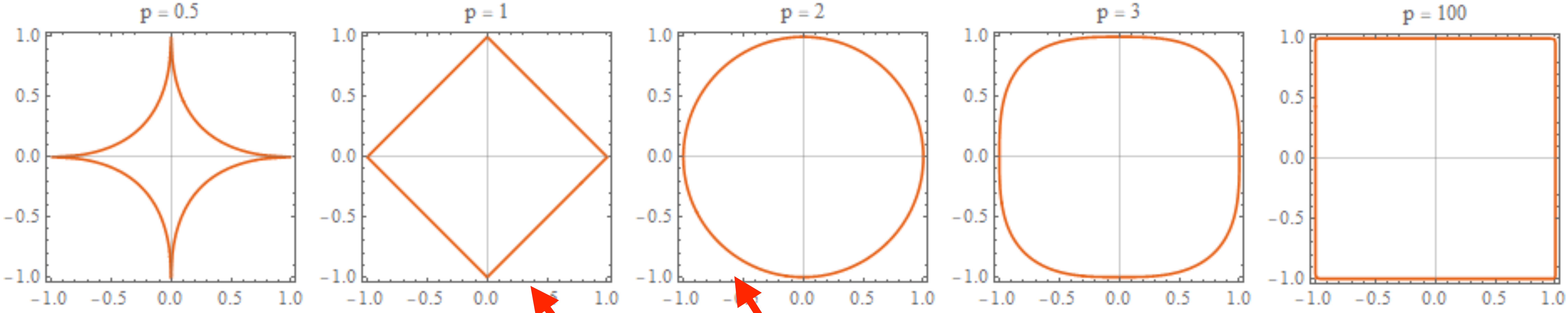
1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

Lp-norm: $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p \right)^{\frac{1}{p}}$





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



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

  It says: the smaller the better (the more probable)

- Laplace prior $p(\mathbf{w}) = \frac{1}{2b} e^{(-\frac{|\mathbf{w}|}{b})}$ => L1-regularization: $\|\mathbf{w}\|_1$

  It says: the sparser the better (the more probable)

- L2-regression with L1-regularization is known as Lasso

Summary

- Machine learning = optimization of the criterion, we do not have access to (KL divergence between true distribution and model)

Optimization $\neq$ Machine learning

- Avoid any "*not-well justified leprechauns*" in the model, => avoid overfitting

Always use the right ("leprechauns-free") architecture

  - Projective transformation of pinhole cameras (for camera calibration or stereo)
  - Geometry of Euclidean motion (point cloud alignment, direct kinematic tasks)
  - Motion model of robots such Dubins car, pendulum, ODE …
  - Structure of animal cortex (for ConvNets)

Prefer simplier solutions (flat minima, lower weights, …)

  - Less is sometimes more

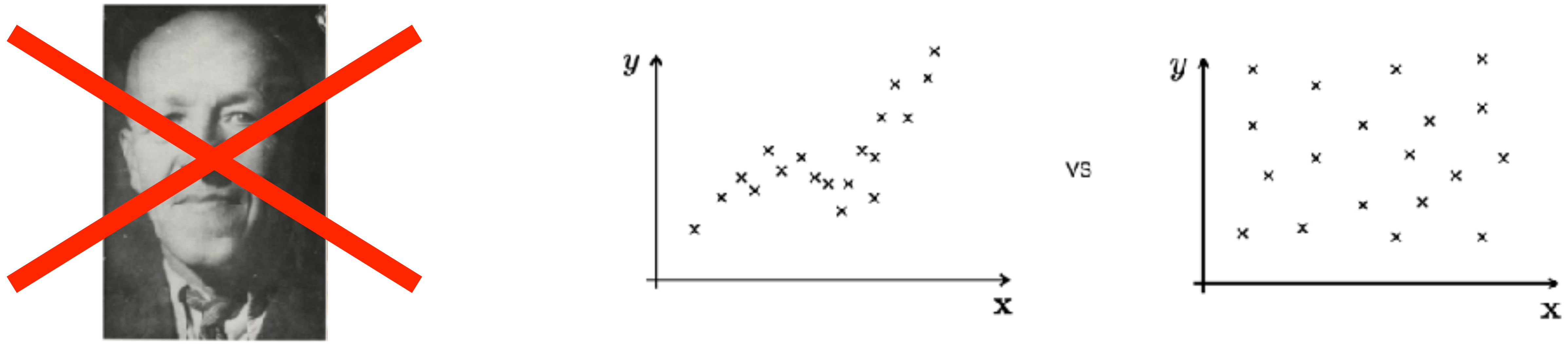
Use close-to-infinite training data

  - More is sometimes more ;-)

- Avoid oversimplifications of the model, => avoid underfitting

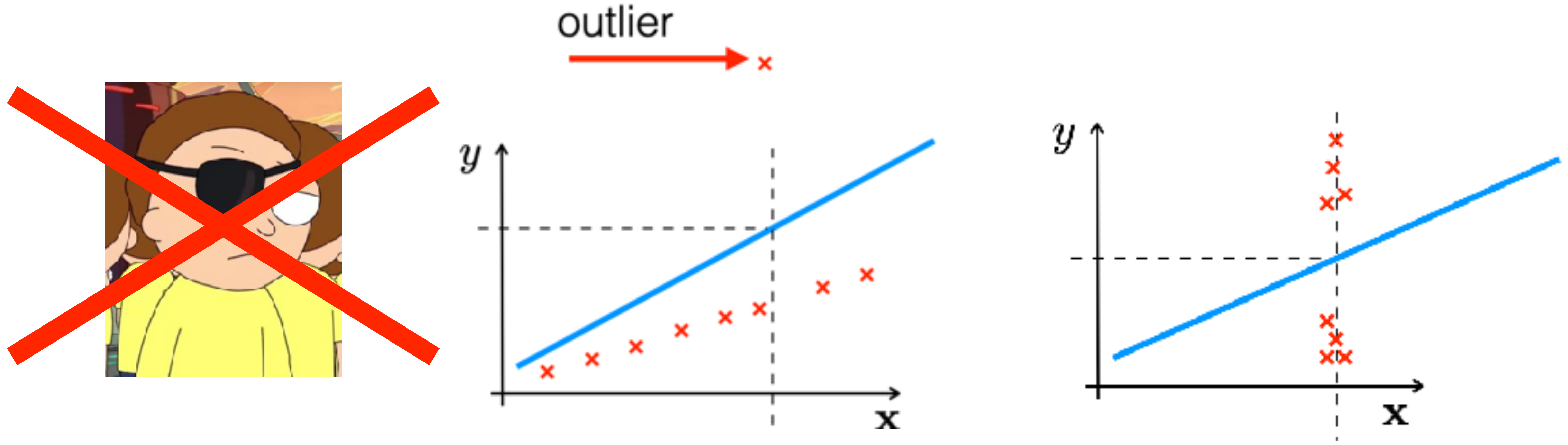
Golden grale:

- Solve only “Pilcik-free” problems



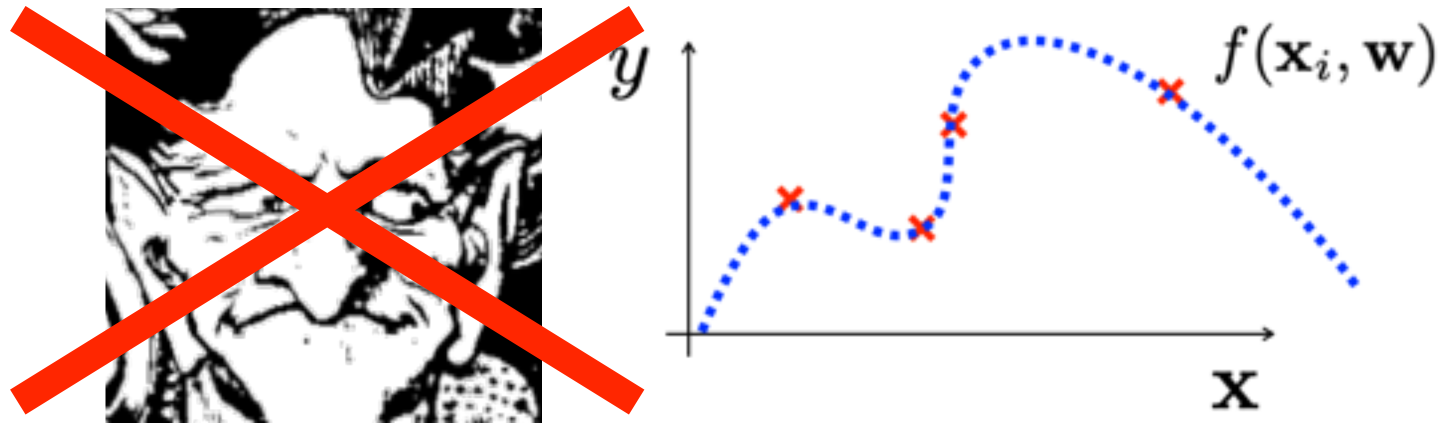
Lecture 1

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



Lecture 5

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



Lecture 4,8,10
ConvNets
∇ODE, ∇argmin

- Avoid traps in learning

Lecture 7
Optimizers

Competencies required for the test T1

- Derive MLE estimate for regression and classification for different noise models
- Derive L2/L1/cross-entropy/logistic losses,
- Understand connection between KL divergence, loss, optimization, machine learning, underfitting, overfitting and model architecture.