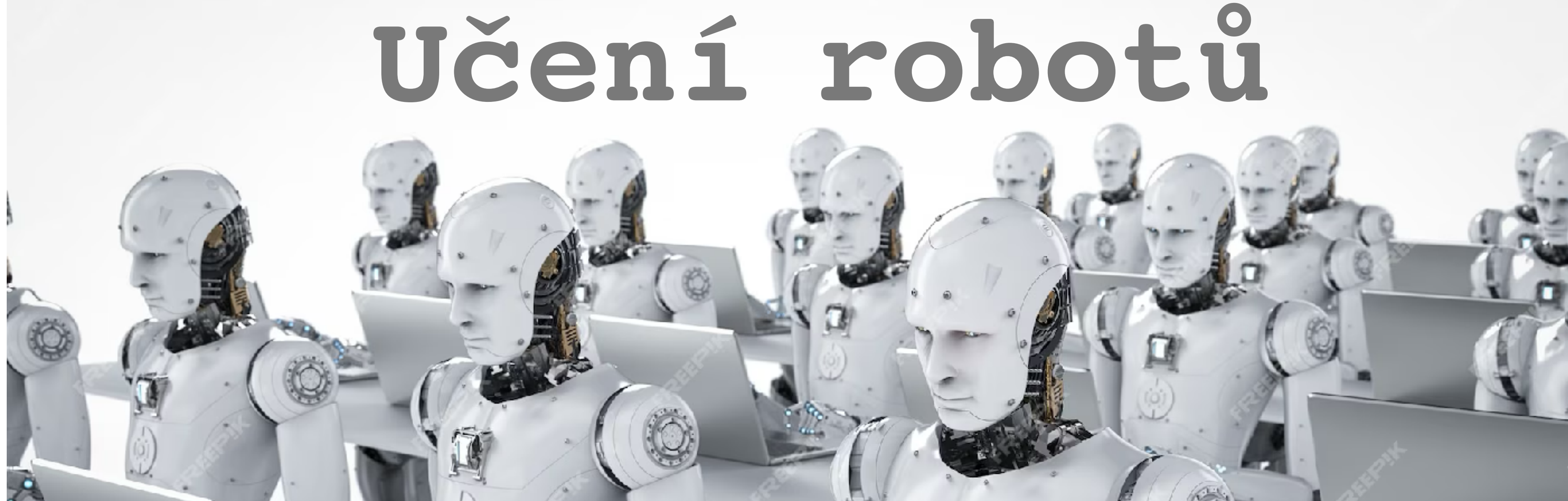


Učení robotů



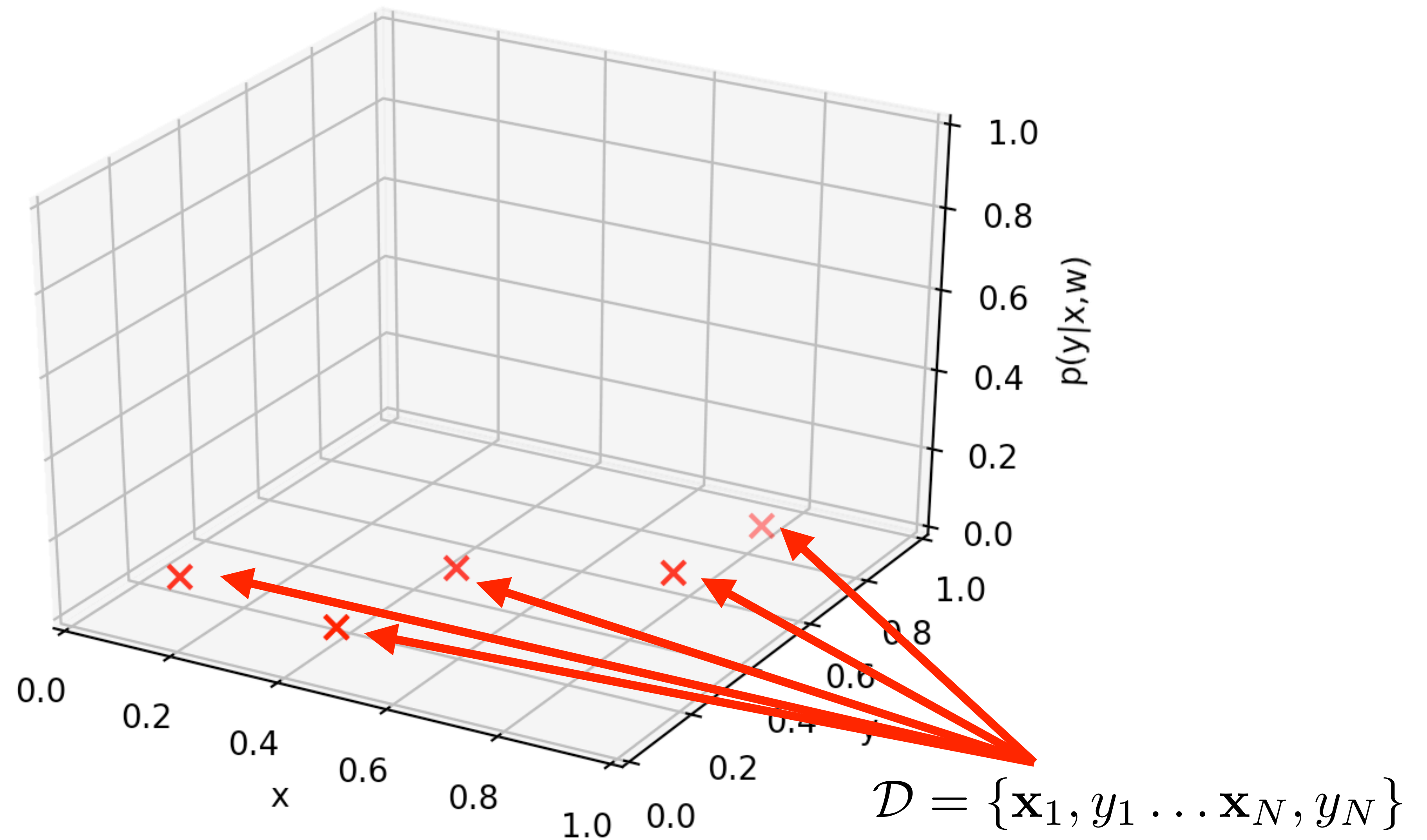
Where the hell does the loss and the overfitting come from?

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

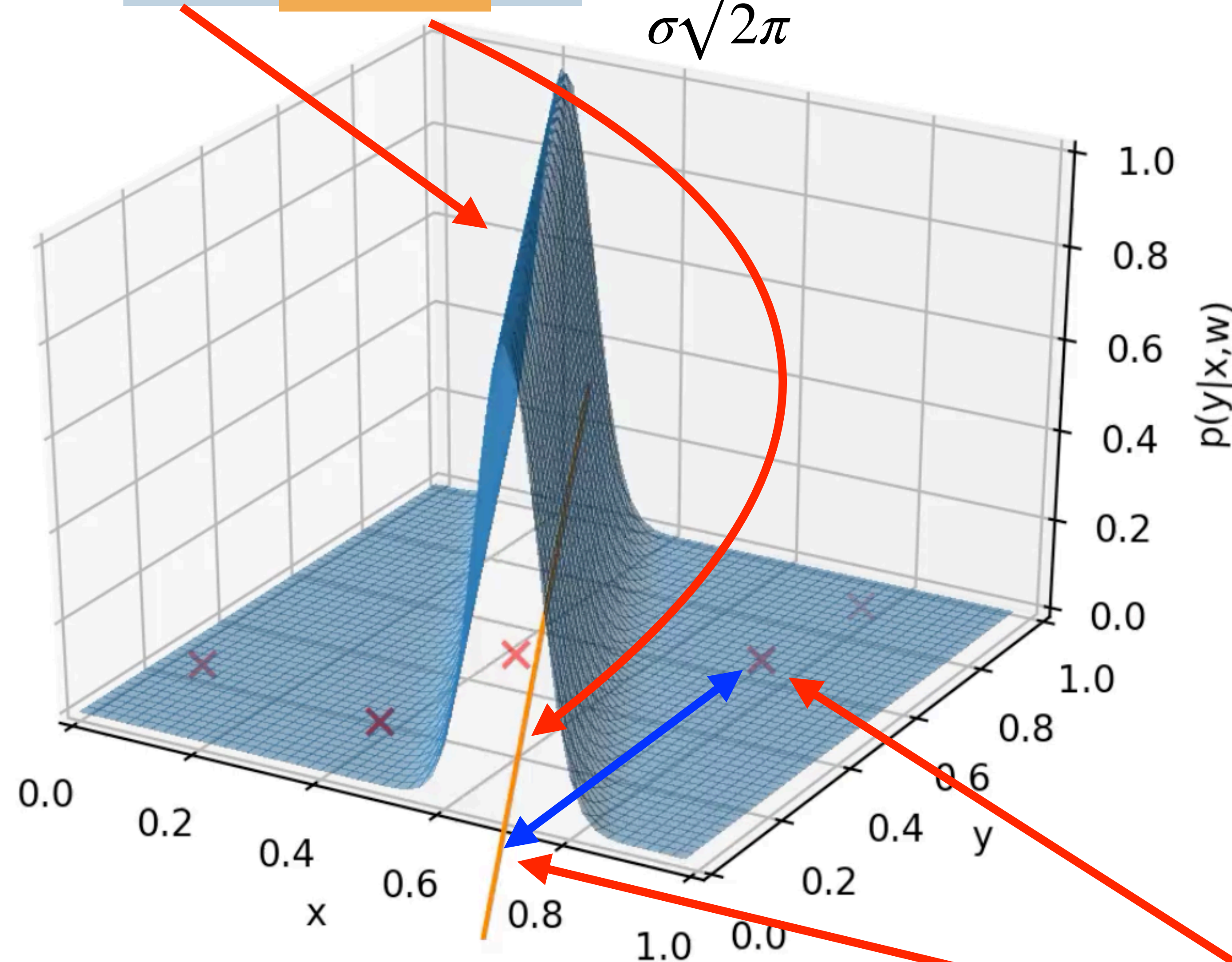
Karel Zimmermann
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Prerequisites: MLE

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



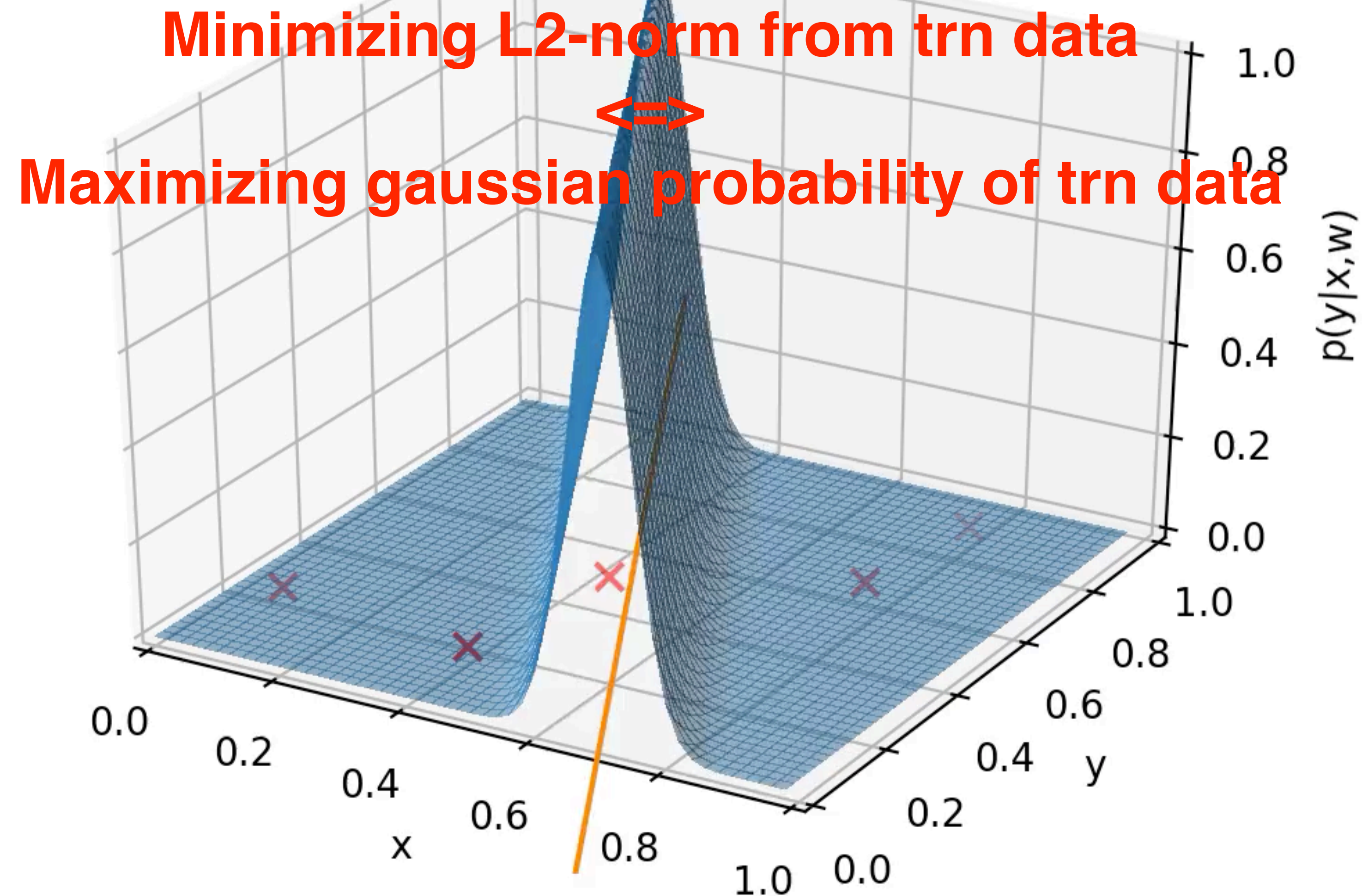
$$p(y | \mathbf{x}, \mathbf{w}) = \mathcal{N}(y; w_1 x + w_0, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp^{-\frac{1}{2} \left(\frac{w_1 x + w_0 - y}{\sigma} \right)^2}$$



Prove it !!!

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \prod_i \mathcal{N}(y_i; w_1 x_i + w_0, \sigma) = \arg \min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$$

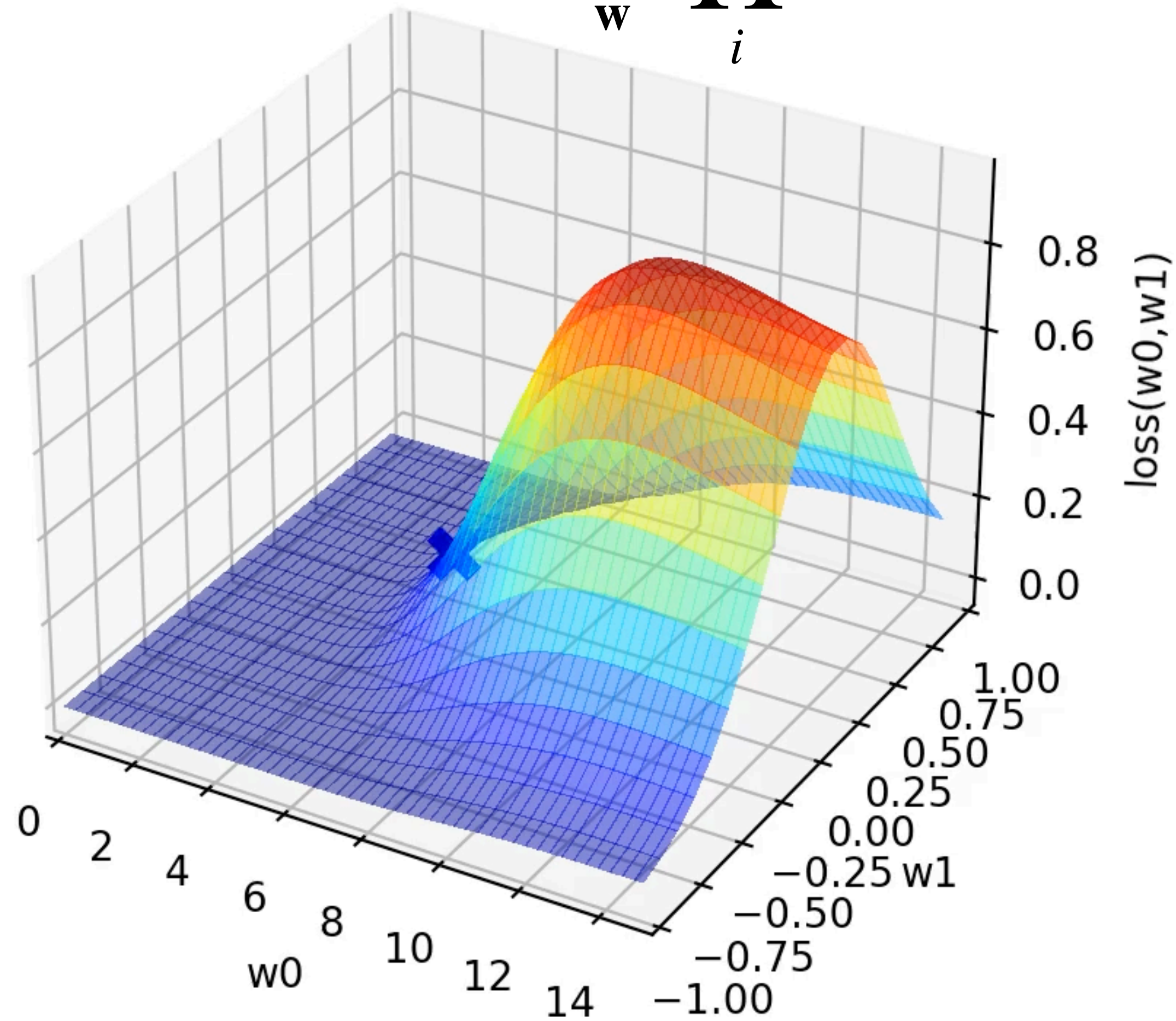
$$p(y | \mathbf{x}, \mathbf{w}) = \mathcal{N}(y; w_1 x + w_0, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp^{-\frac{1}{2} \left(\frac{w_1 x + w_0 - y}{\sigma} \right)^2}$$



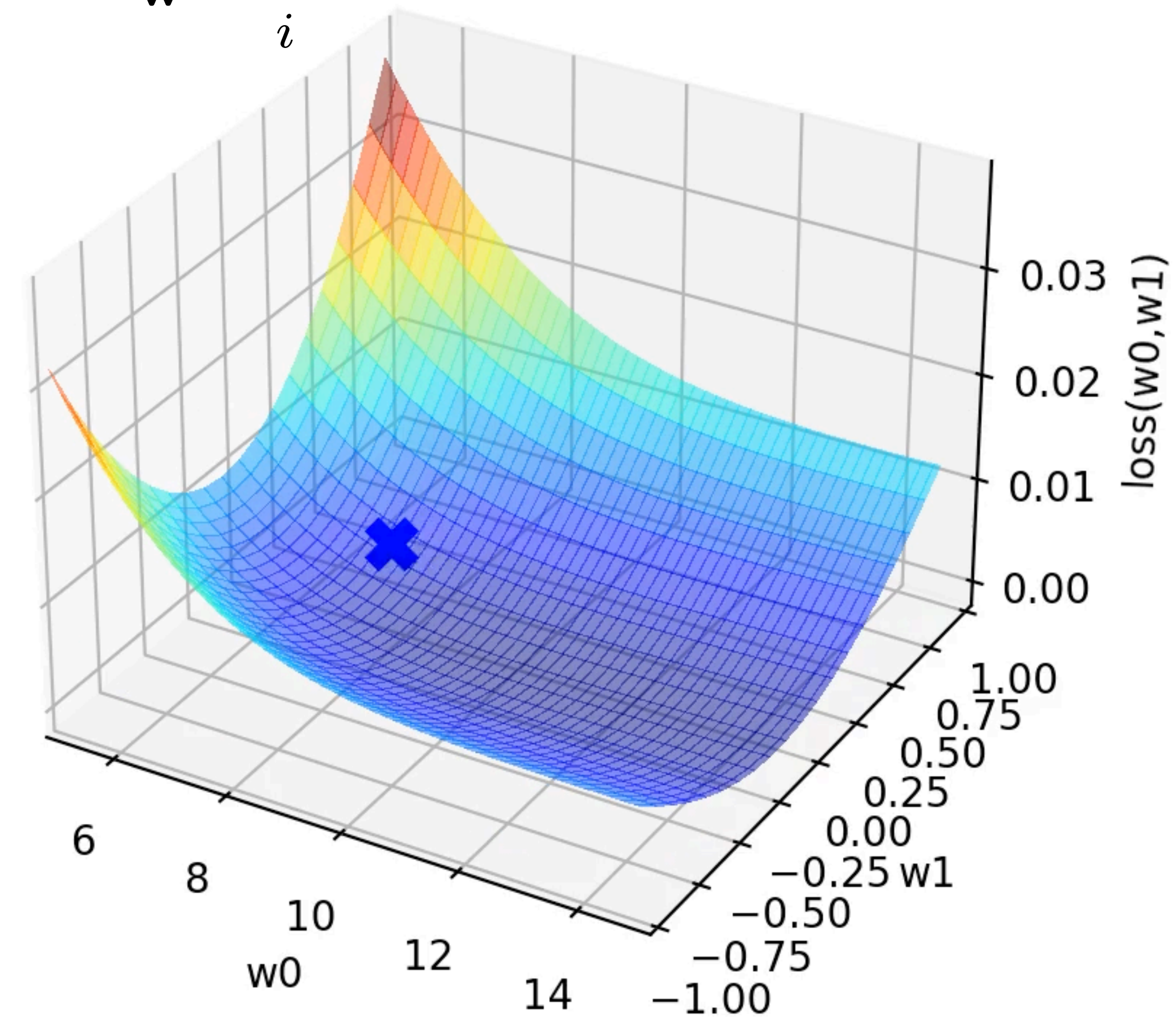
$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \prod_i \mathcal{N}(y_i; f(\mathbf{x}_i, \mathbf{w}), \sigma) = \arg \min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$$

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

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



MLE



LSQ

Prerequisites: Bayes theorem

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

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

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

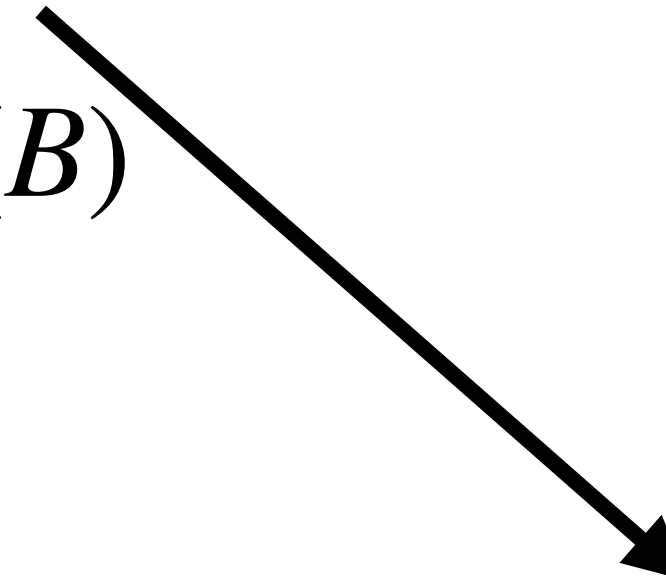
$$p(A, B | C) = p(A | B, C) \cdot p(B | C) = p(B | A, C) \cdot p(A | C)$$

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

Prerequisites: Independence

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

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

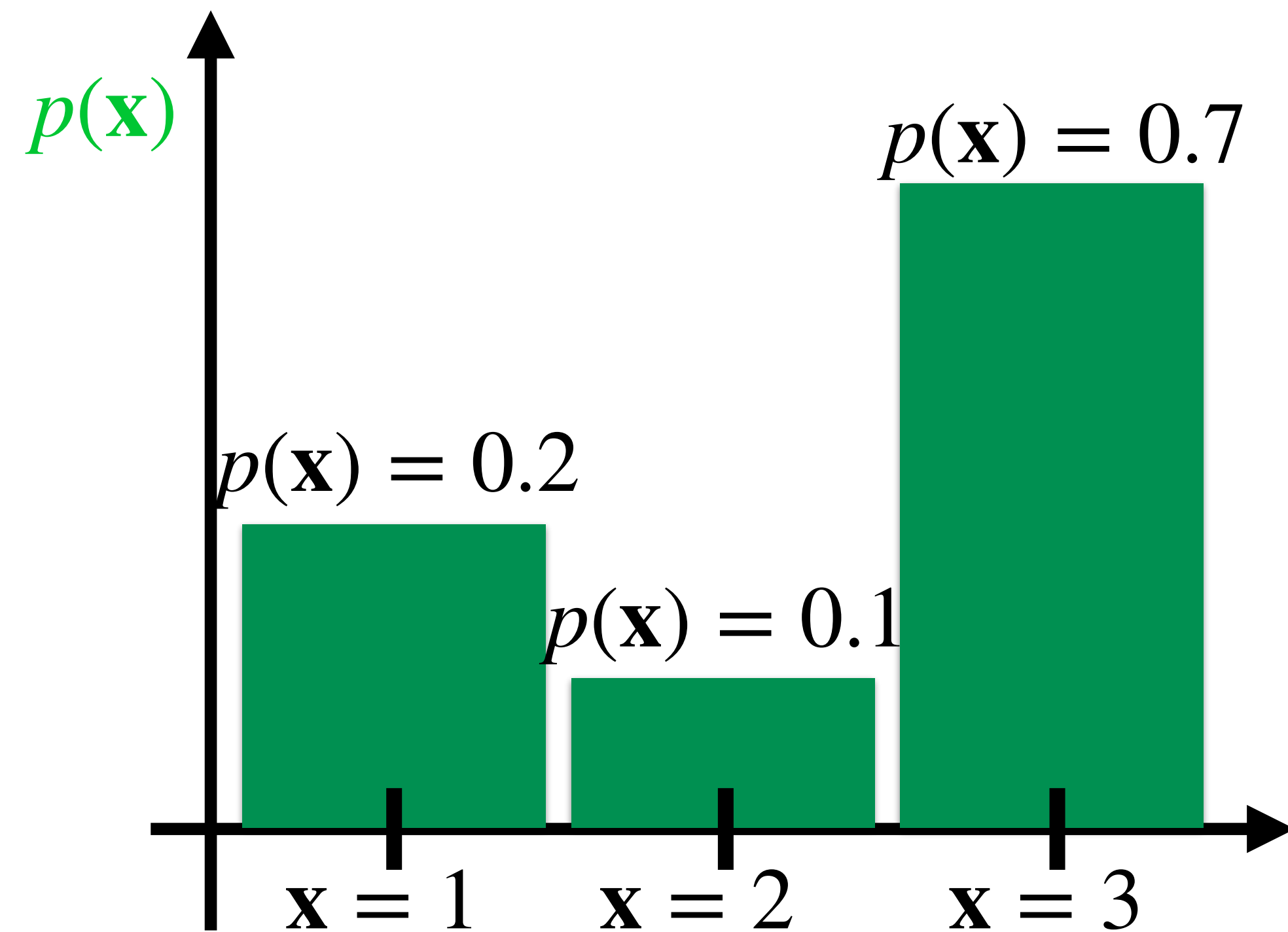


Let's put it together: $p(A, B) = p(A) \cdot p(B) = p(A | B) \cdot p(B)$

$$p(A) = p(A | B)$$

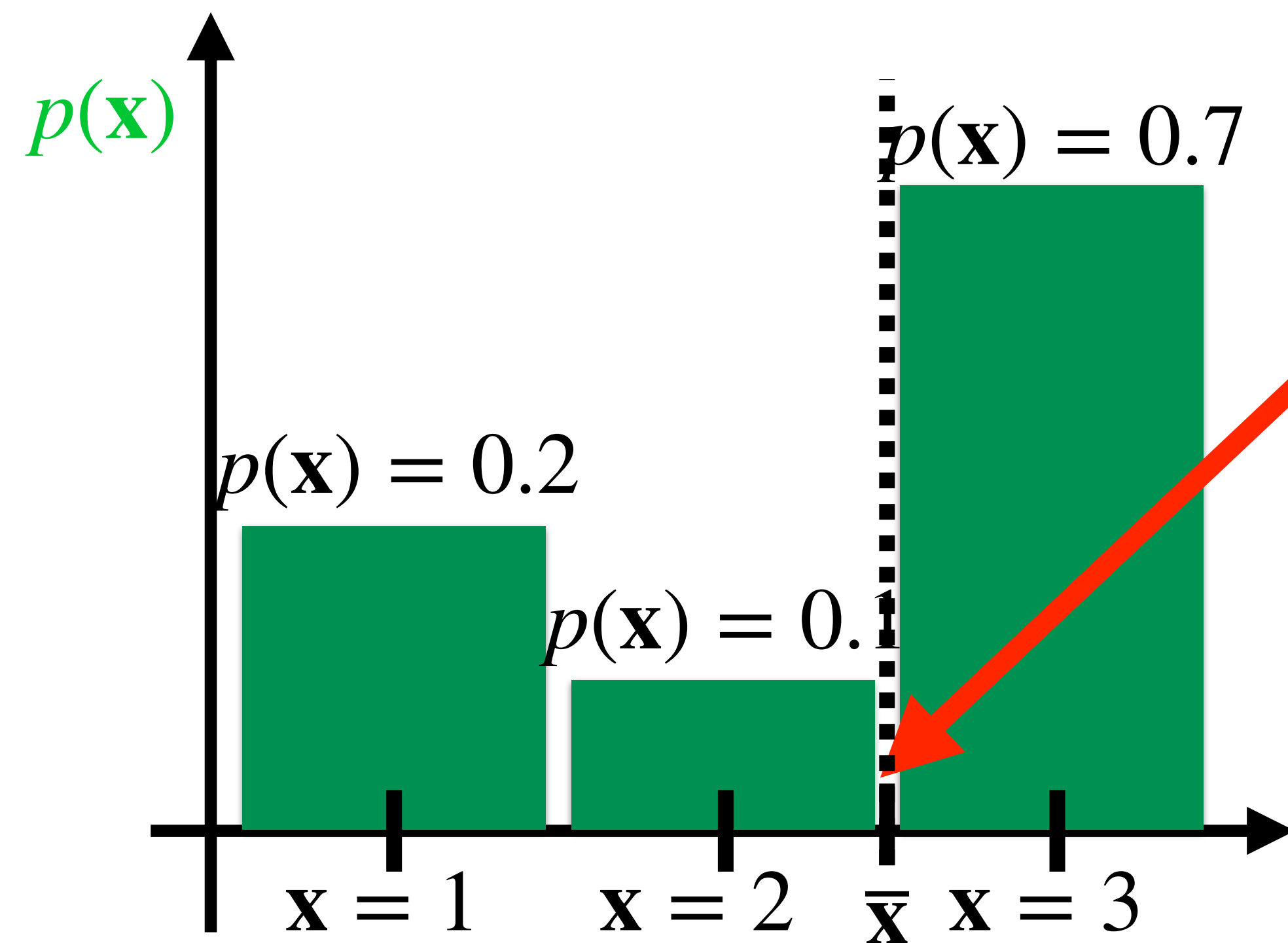
Prerequisites: Mean and average

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



Prerequisites: Mean and average

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

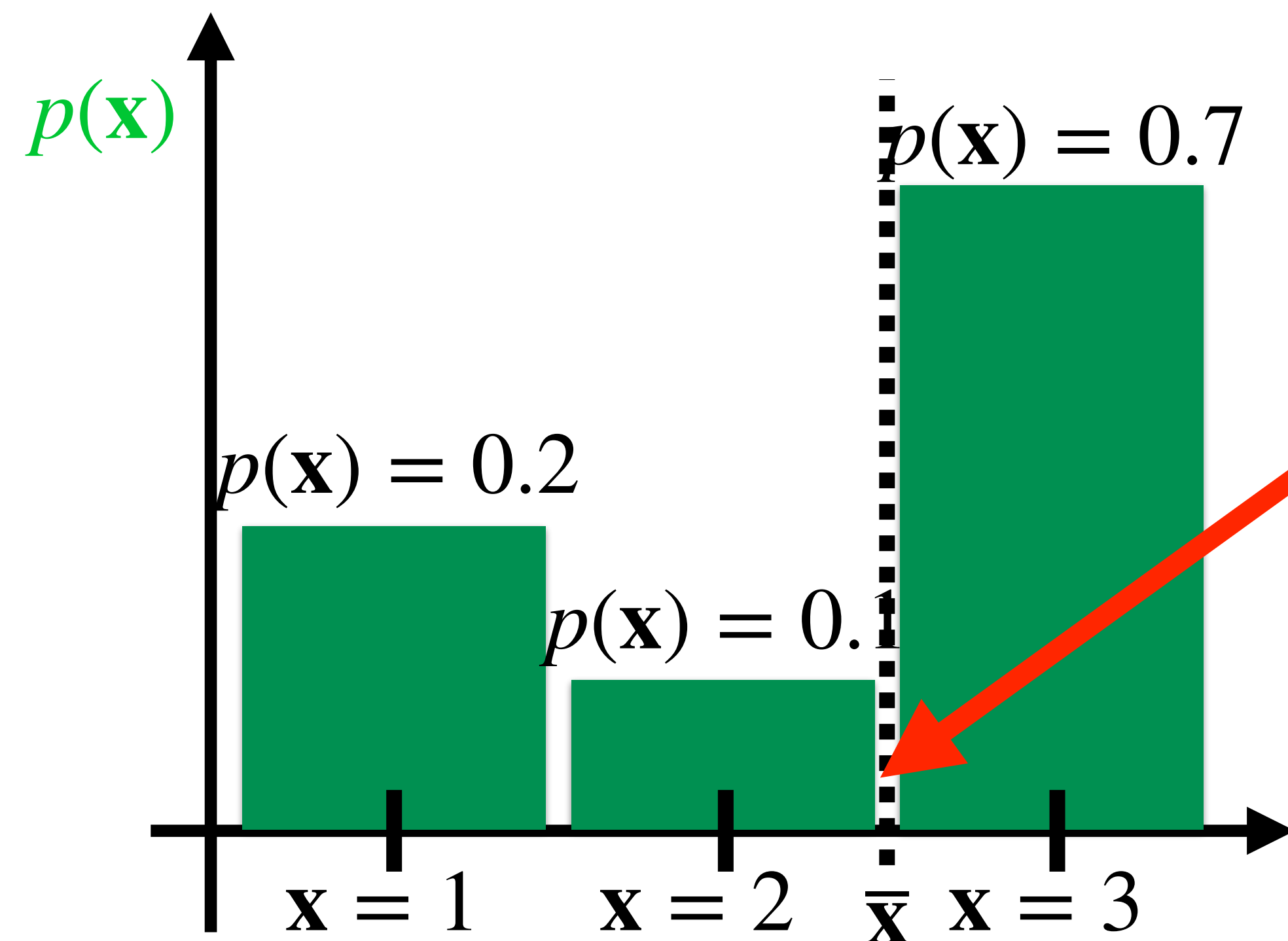


Prerequisites: Mean and average

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

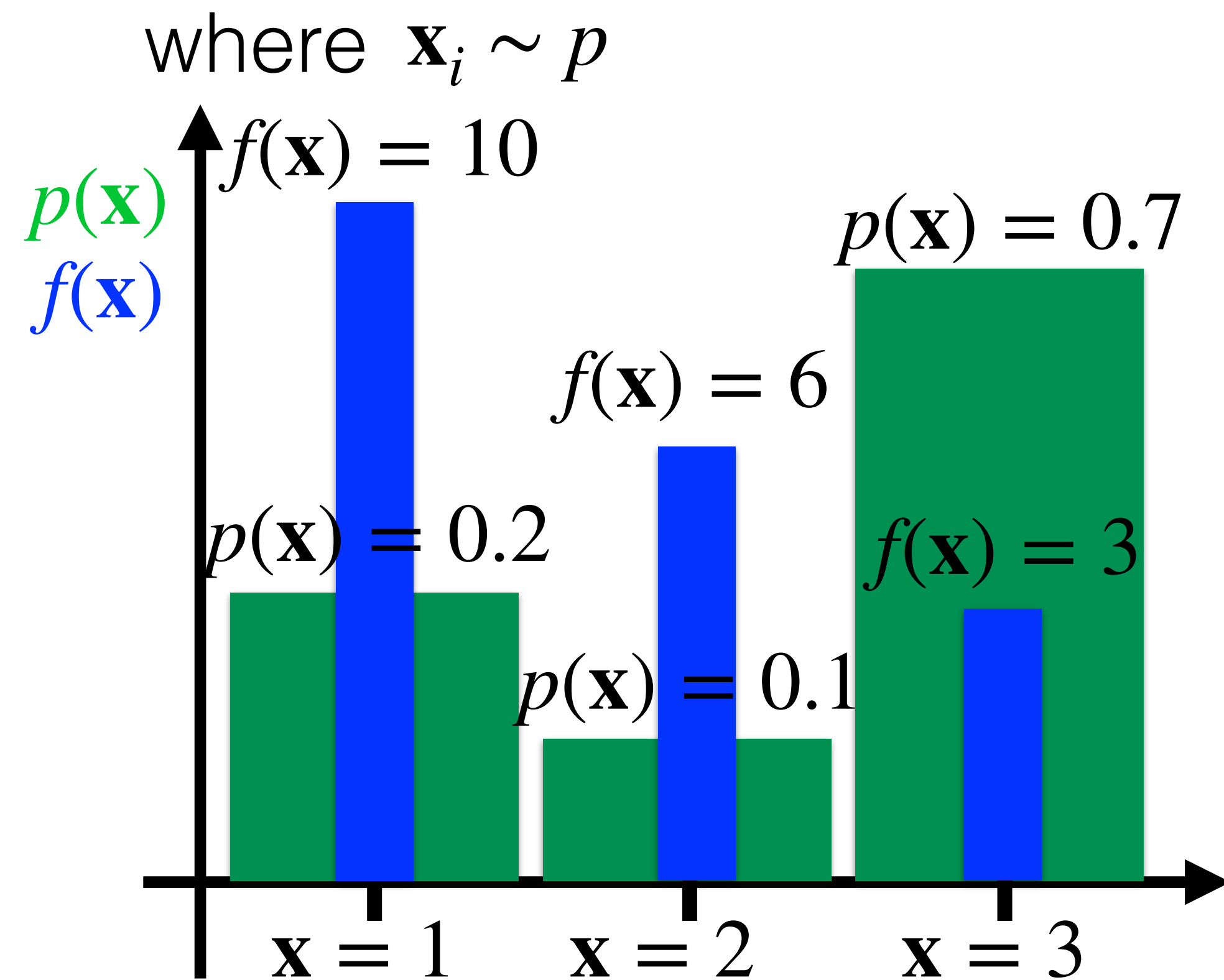
$$\approx \frac{1}{N} \sum_i \mathbf{x}_i = \frac{1}{10}(1 + 1 + 2 + 3 + 3 + 3 + 3 + 3 + 3 + 3) = 2.5$$

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



Prerequisites: Mean and average

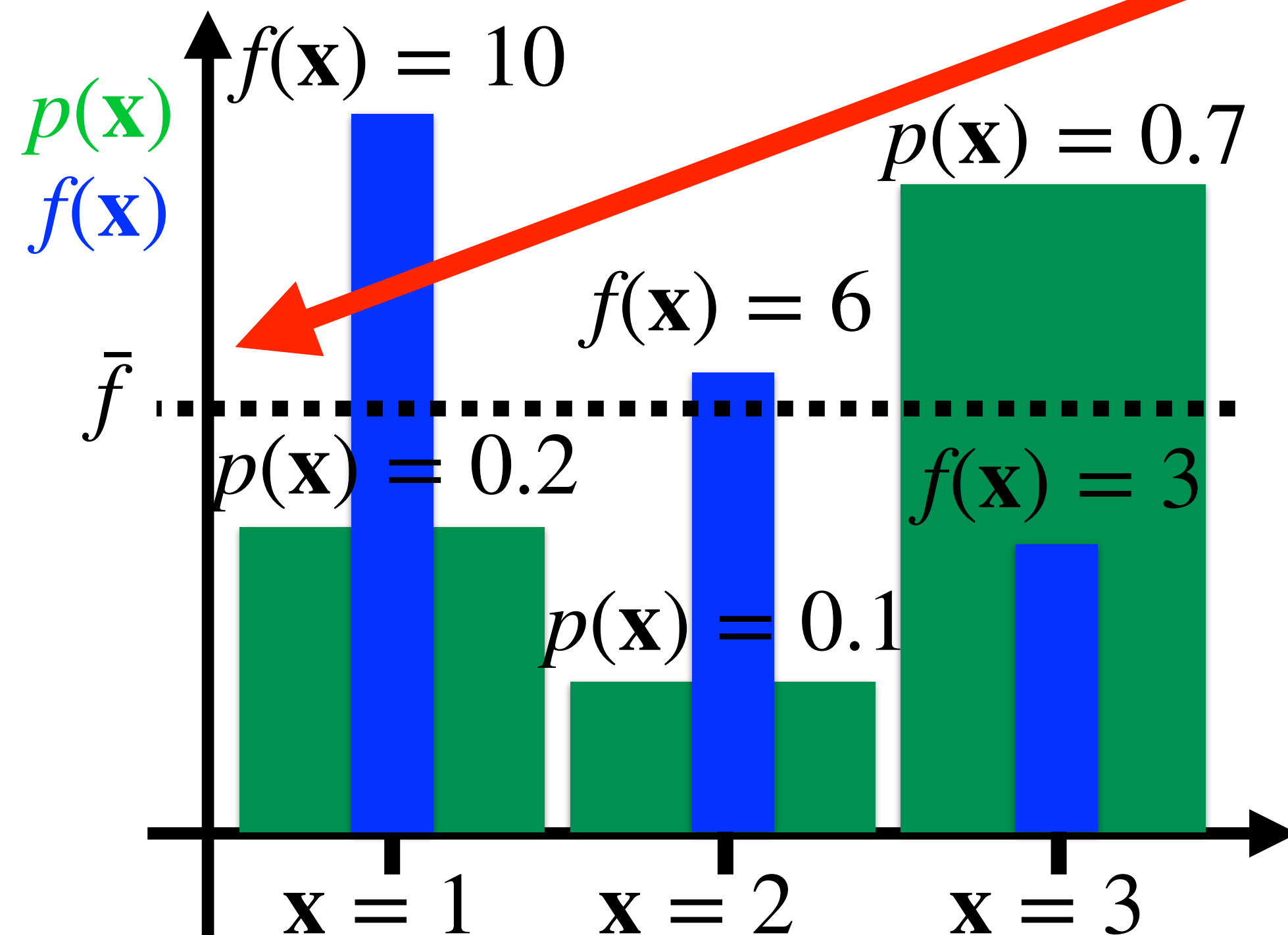
$$\bar{f} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot f(\mathbf{x}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [f(\mathbf{x})] = ??$$



Prerequisites: Mean and average

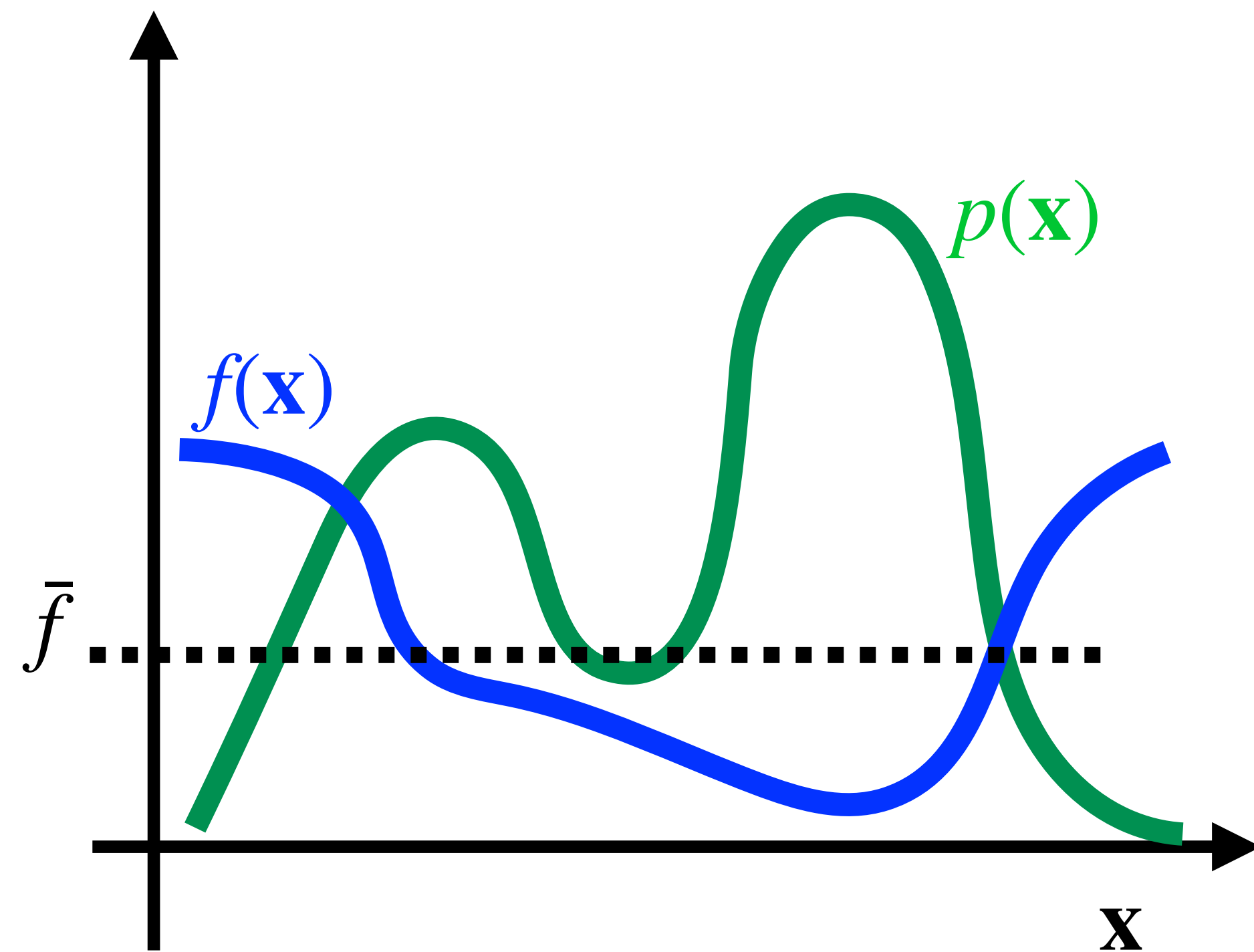
$$\bar{f} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot f(\mathbf{x}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [f(\mathbf{x})] = 0.2 \cdot 10 + 0.1 \cdot 6 + 0.7 \cdot 3 = 4.7$$
$$\approx \frac{1}{N} \sum_i f(\mathbf{x}_i) = \frac{1}{10} (10 + 10 + 6 + 3 + 3 + 3 + 3 + 3 + 3 + 3) = 4.7$$

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



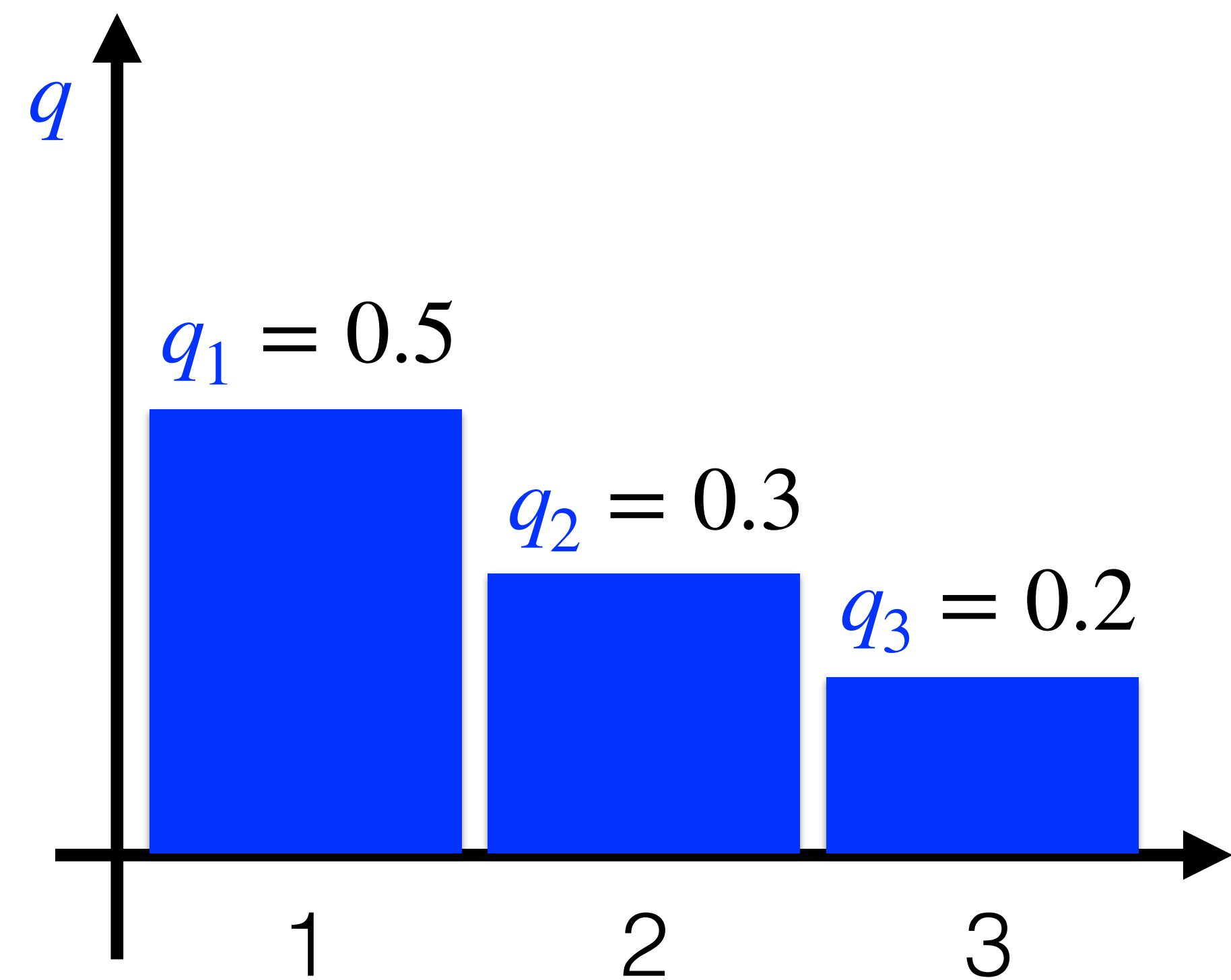
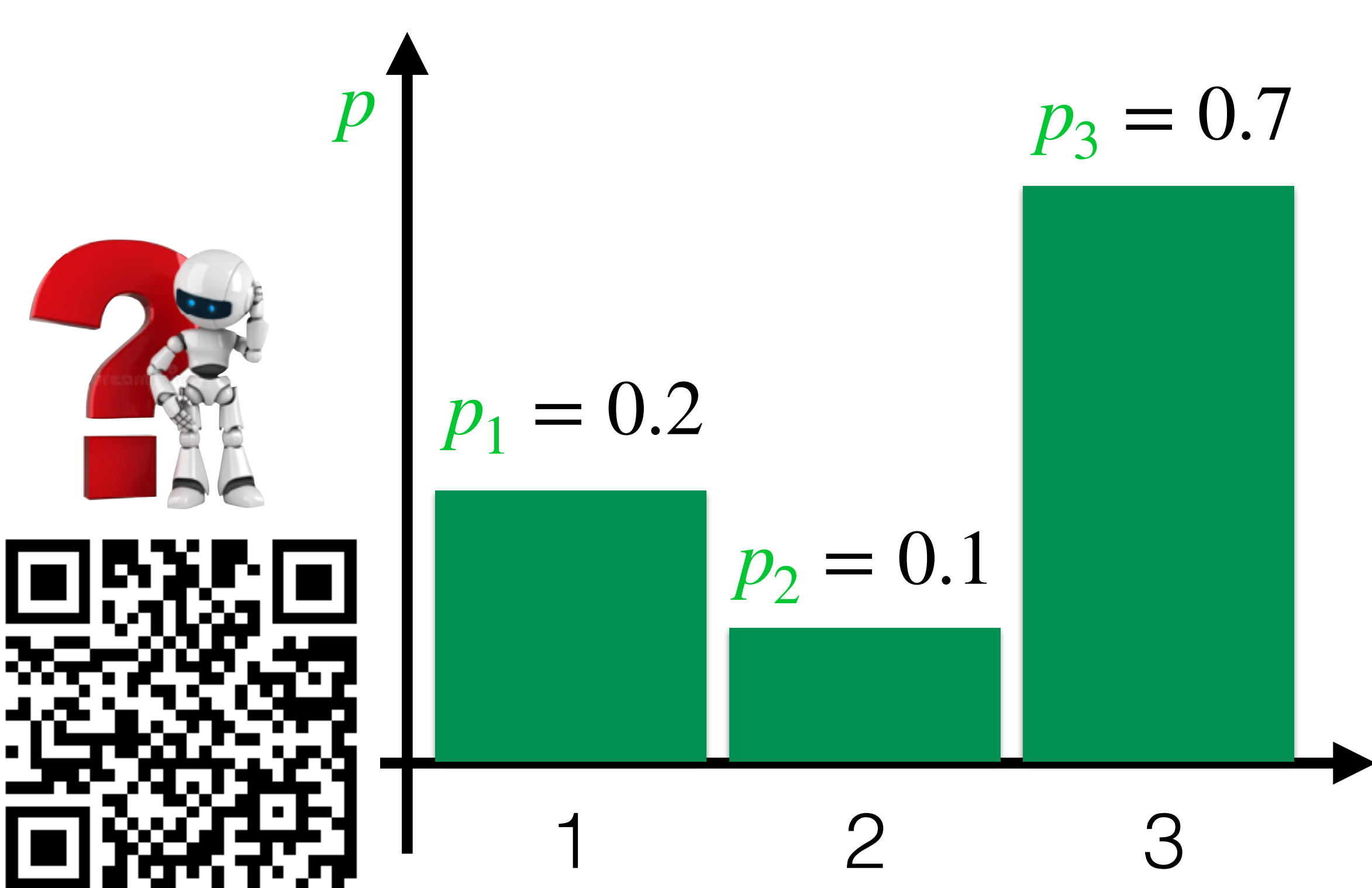
Prerequisites: Mean and average

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



Prerequisites: KL-divergence

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

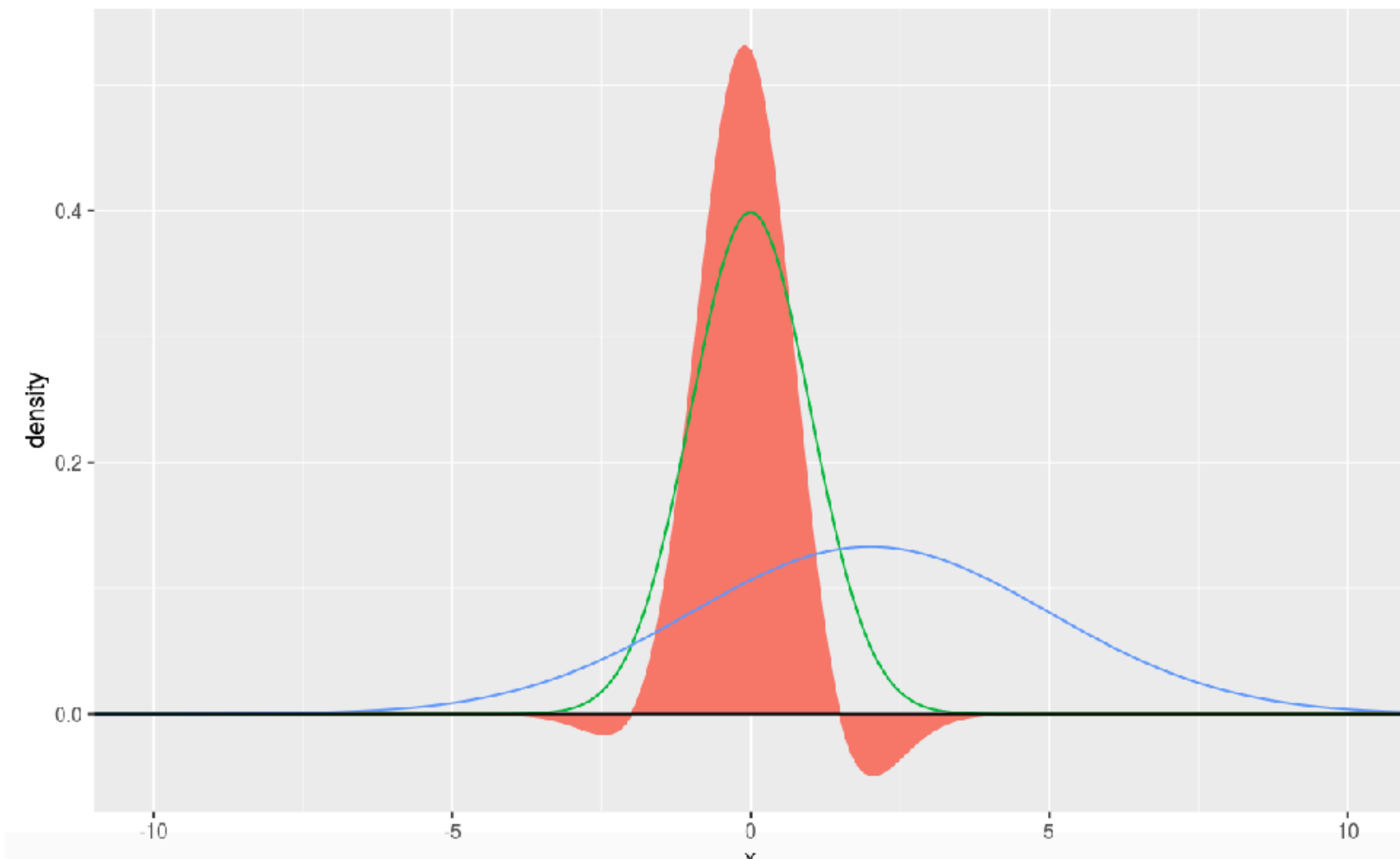


- What if $q_k = p_k$ for $k=1,2,3$?
- What if $q_k = 0$ and $p_k > 0$?
- Is it symmetrical $D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$?

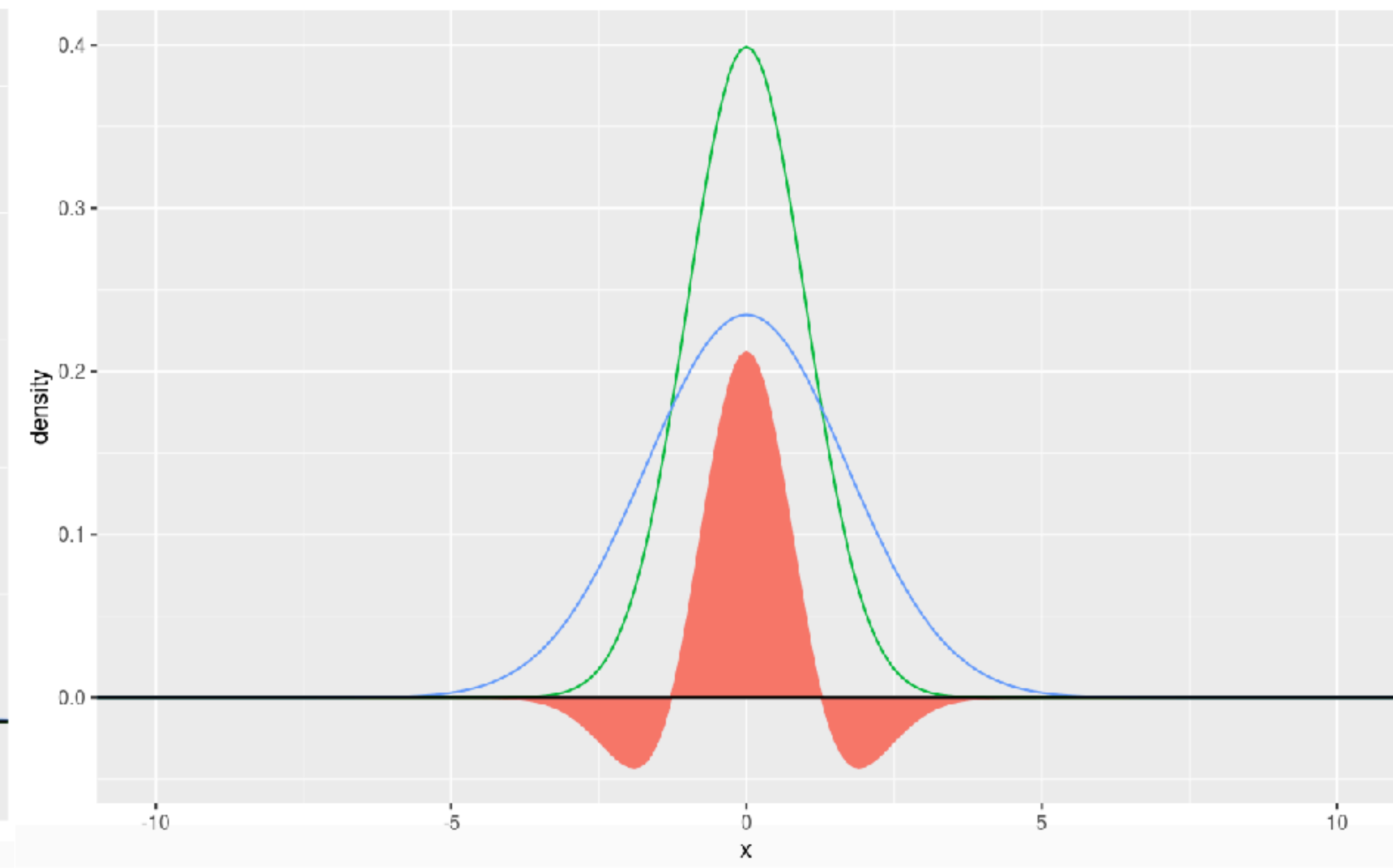
Prerequisites: KL-divergence

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

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

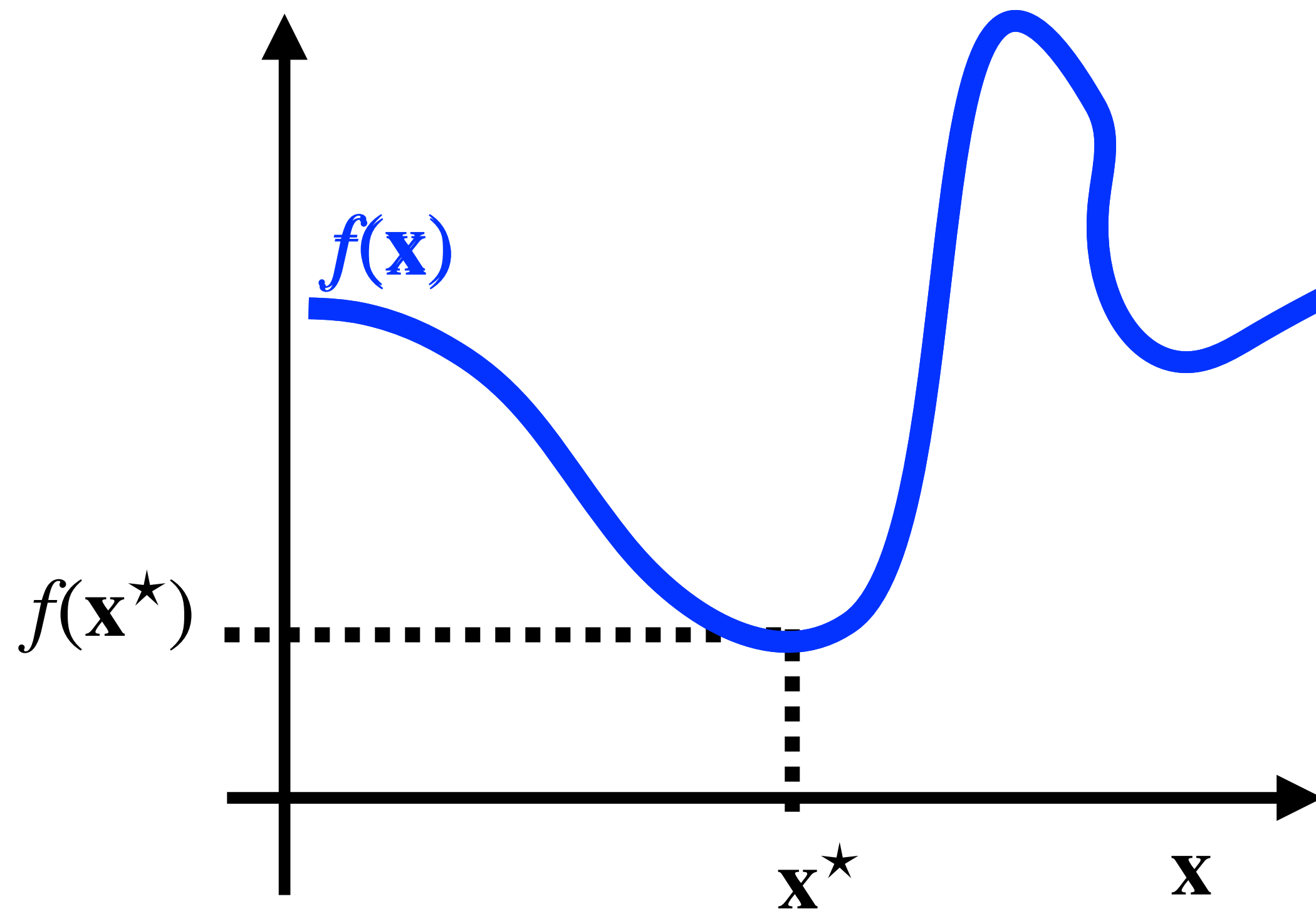


$$D_{KL}(p(\mathbf{x}) \parallel q(\mathbf{x})) = 0.2036$$



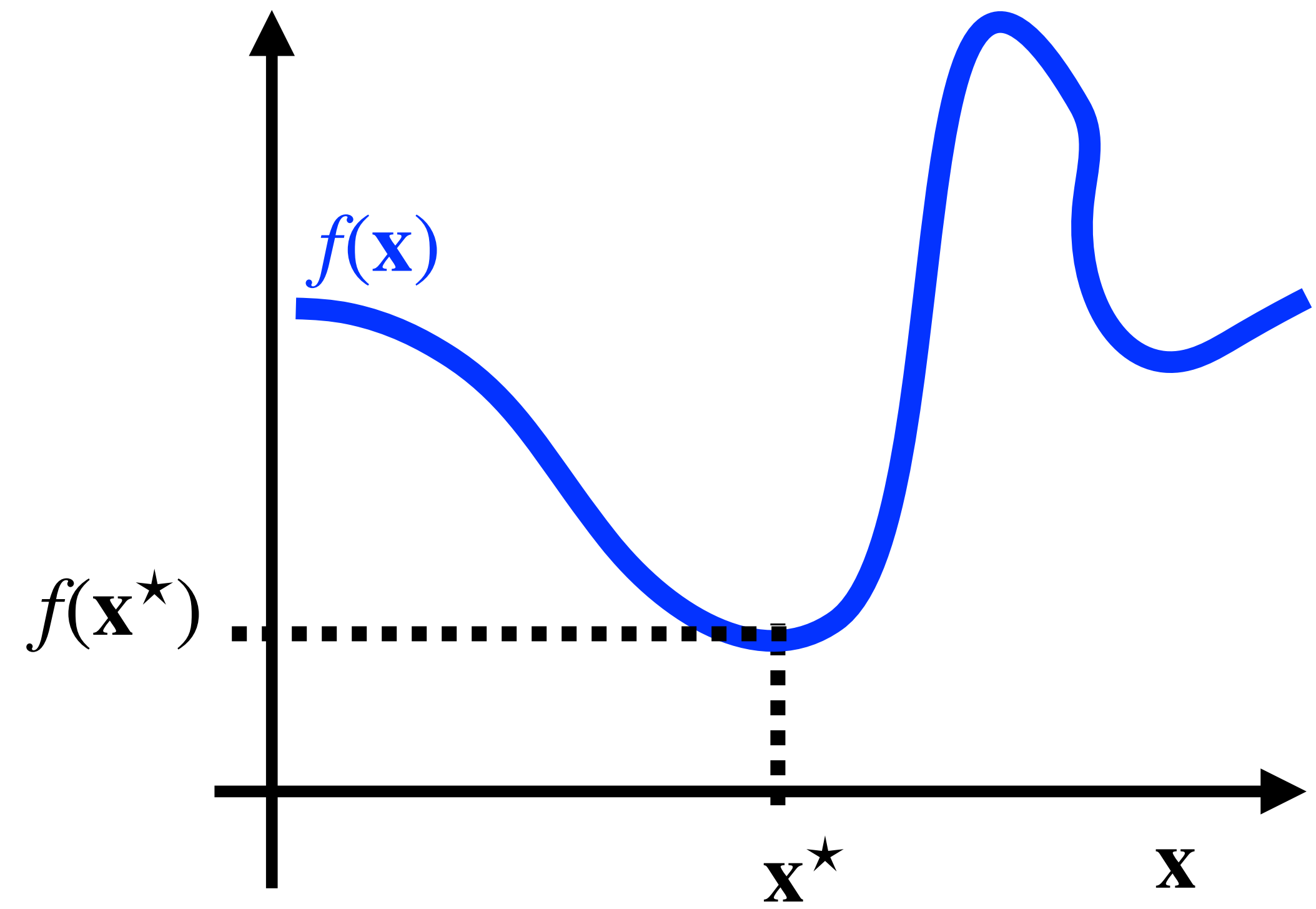
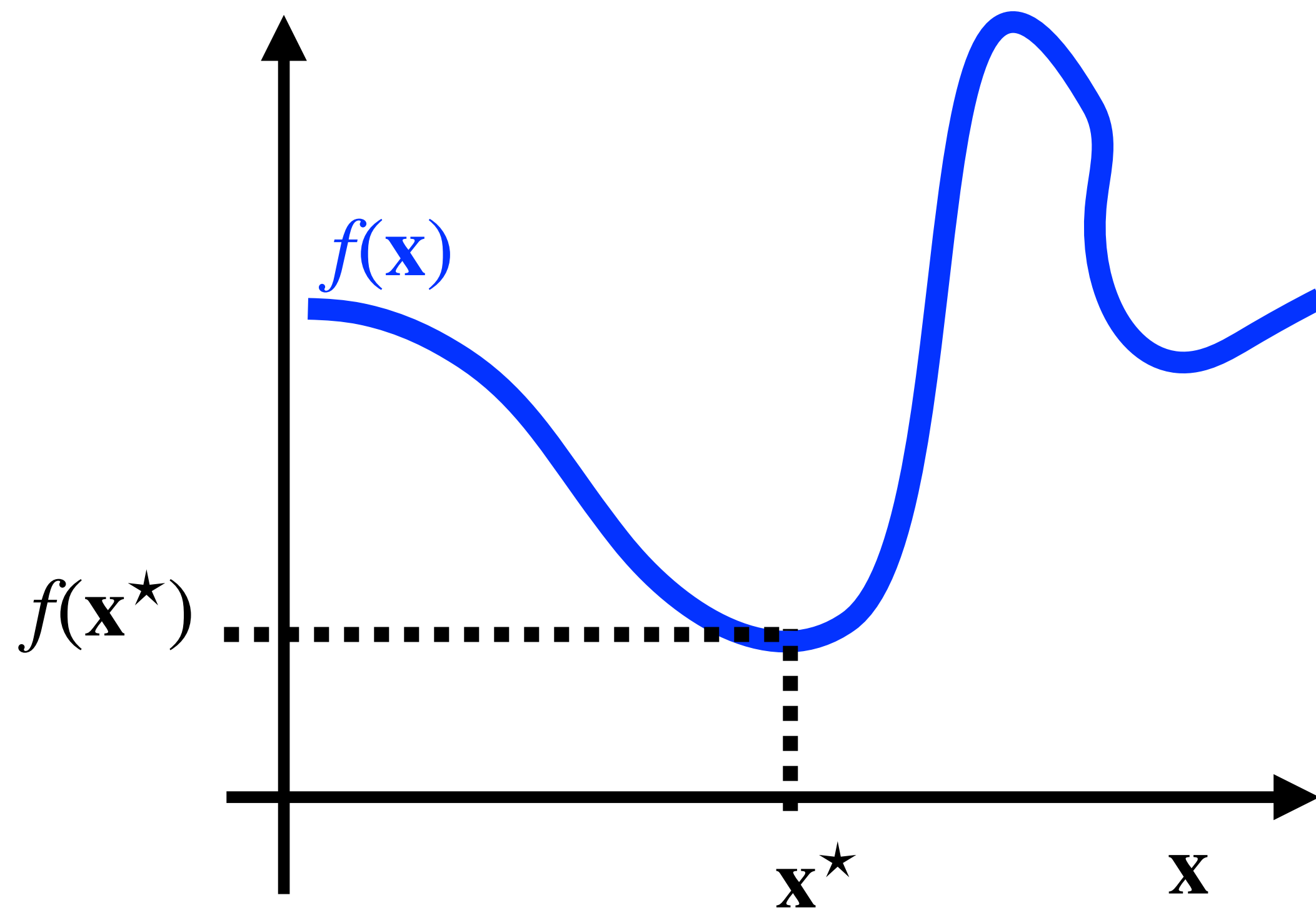
Prerequisites: argmin

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



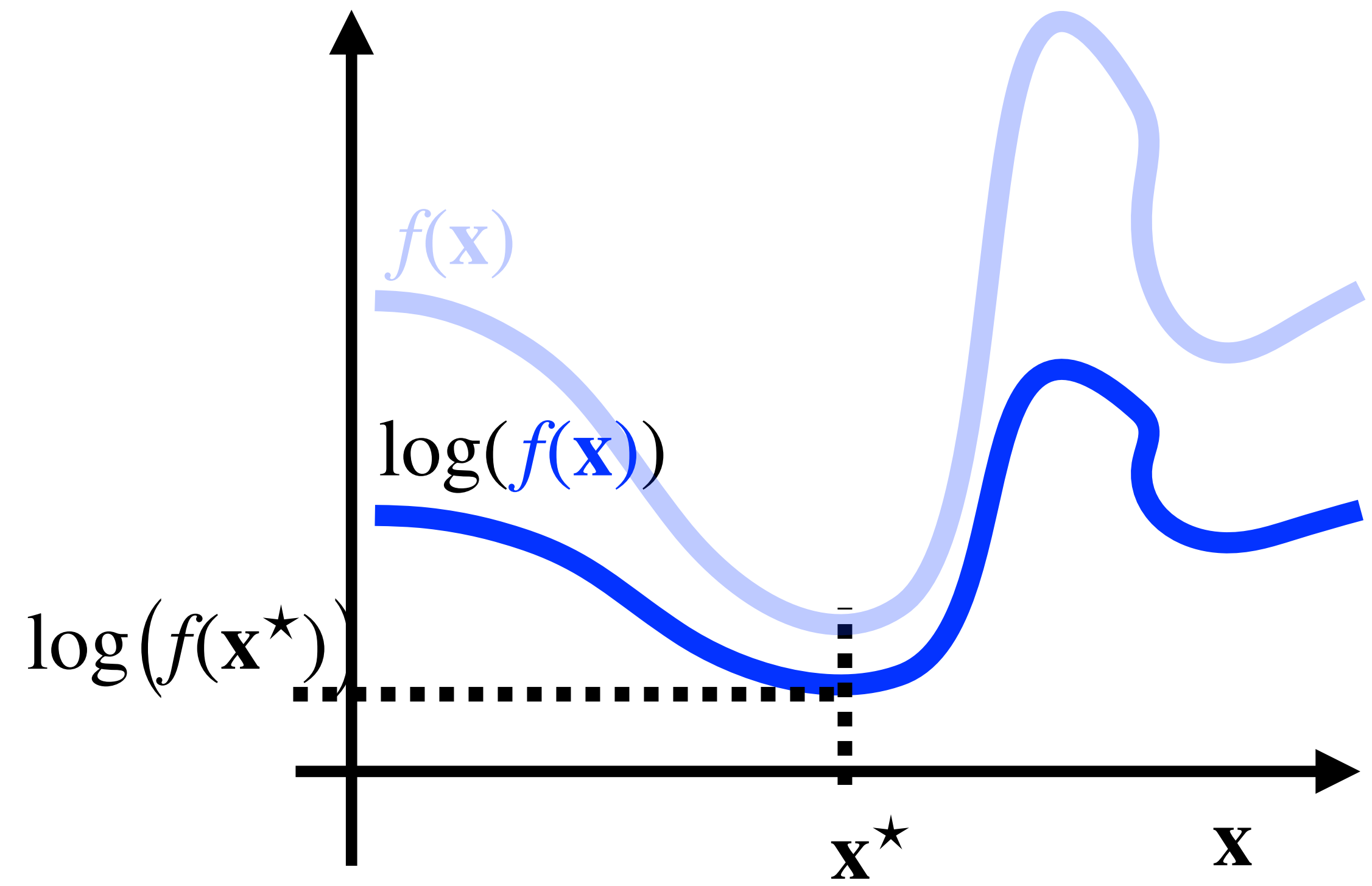
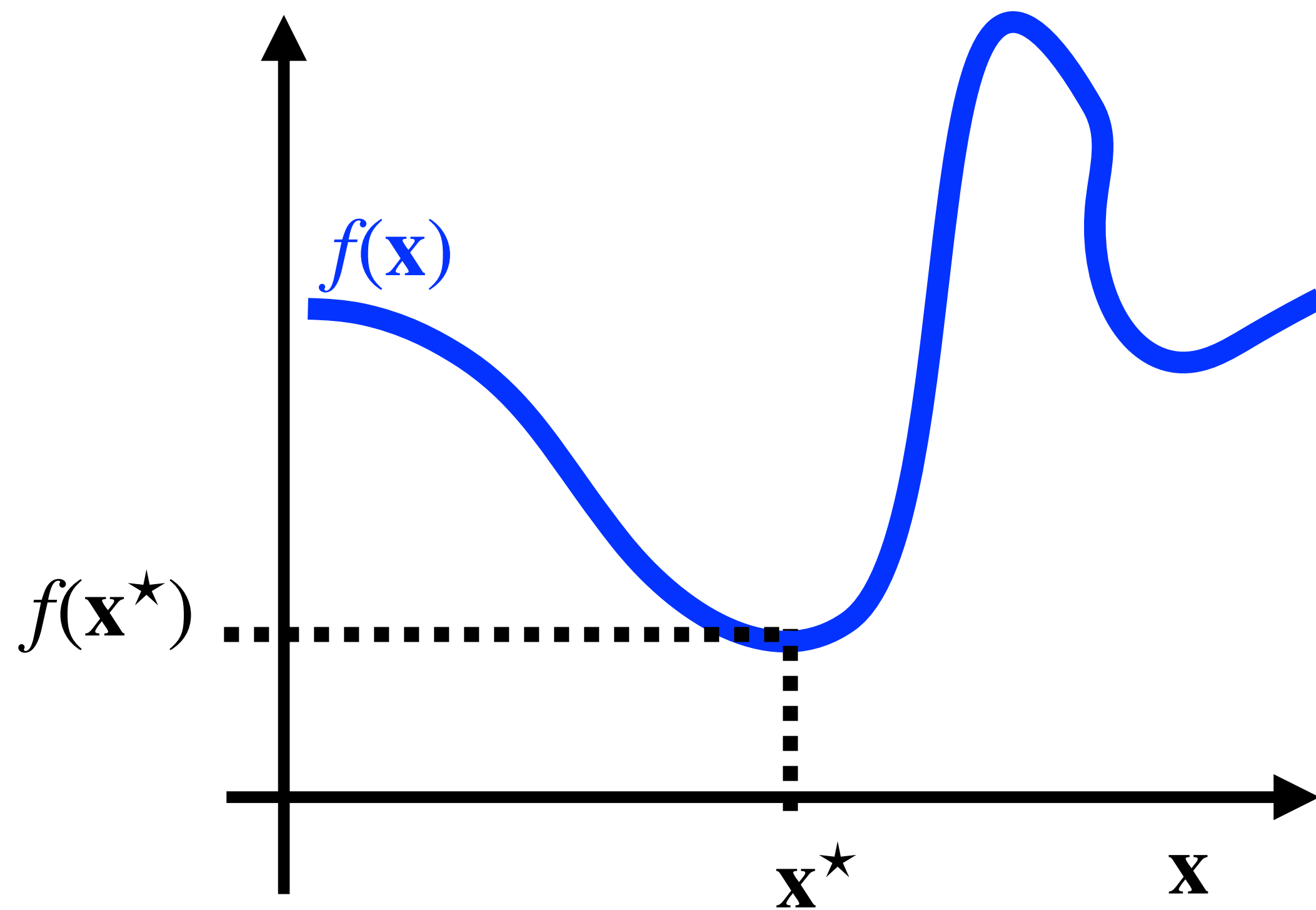
Prerequisites: argmin

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



Prerequisites: argmin

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



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

How should we fit distribution into data?

- Many robotics tasks contain perception problems: given x estimate y

x

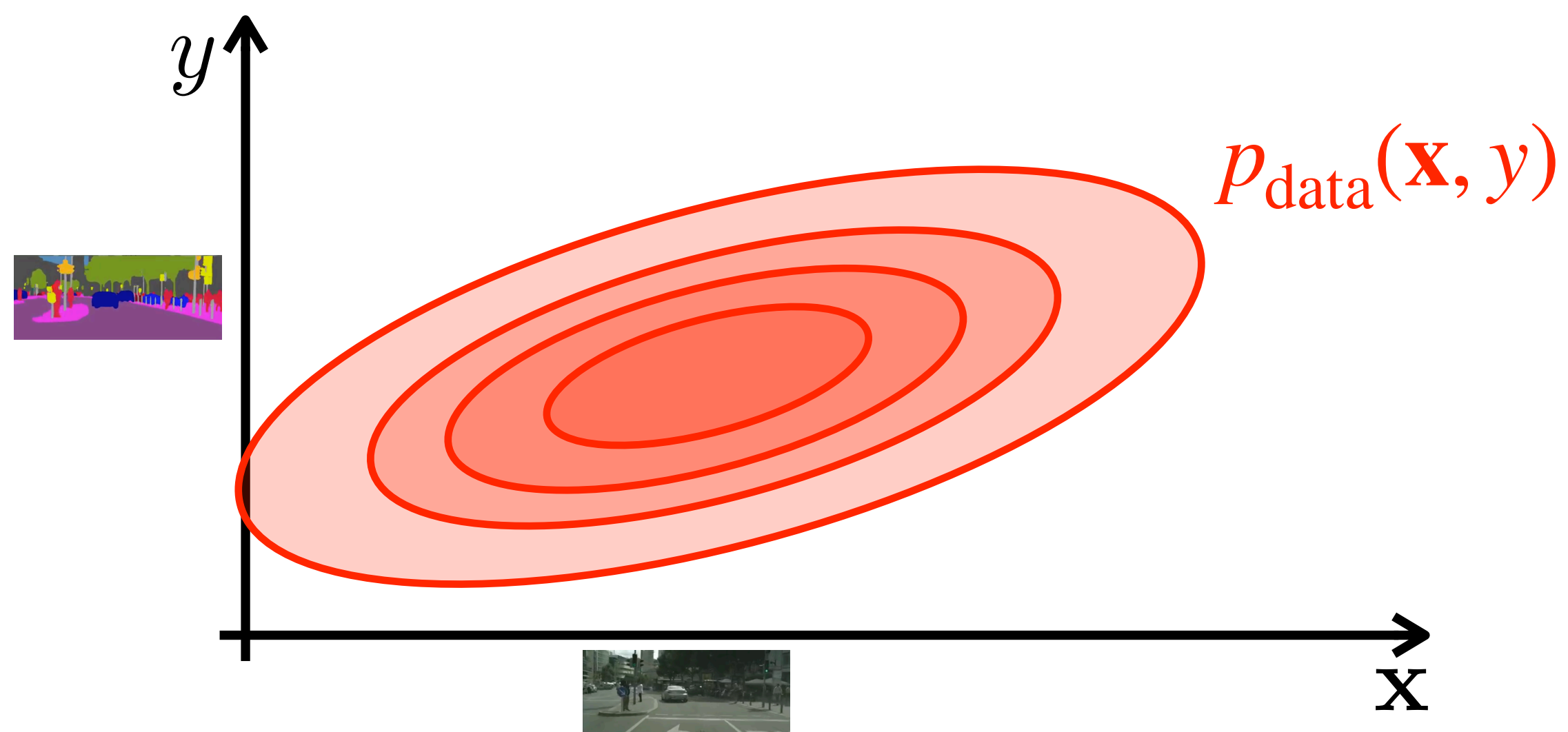


y

- road
- sideway
- pedestrian
- traffic sign
- trees
- sky

Why should I minimize $-\log(p)$?

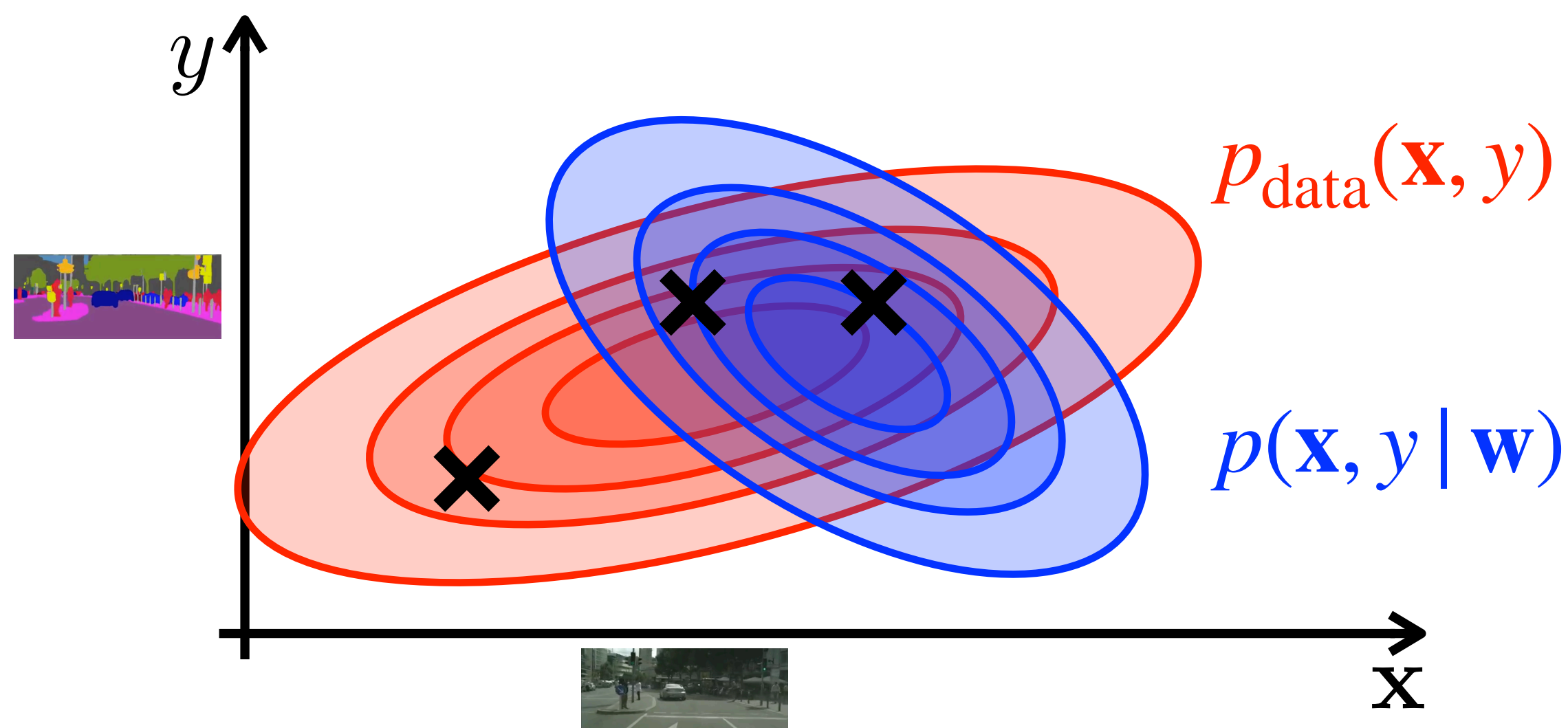
- $(\mathbf{x}, y)$ -tuples live on unknown distribution $p_{\text{data}}(\mathbf{x}, y)$



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 weights $\mathbf{w}$ that makes $p(\mathbf{x}, y | \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$:

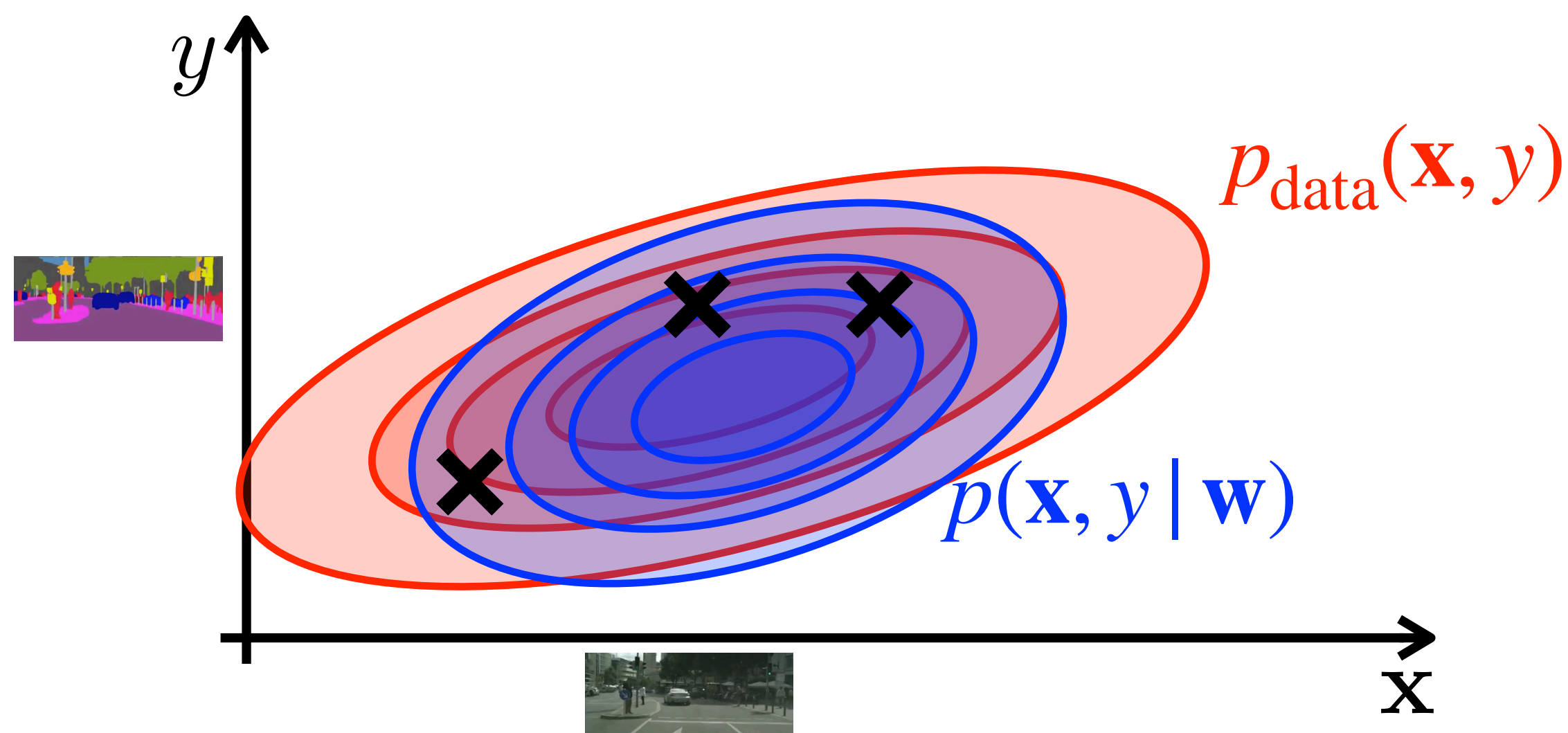
$$\begin{aligned}
 \mathbf{w}^* &= \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w})) = \arg \min_{\mathbf{w}} \int p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y | \mathbf{w})} d(\mathbf{x}, y) \\
 &= \arg \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}, y) \sim p_{\text{data}}(\mathbf{x}, y)} \left[\log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y | \mathbf{w})} \right] = \arg \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}, y)} \left[\log p_{\text{data}}(\mathbf{x}, y) - \log p(y | \mathbf{x}, \mathbf{w}) \right] \\
 &= \arg \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}, y)} \left[-\log p(y | \mathbf{w}, \mathbf{x}) \right] \approx \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} \left[-\log p(y_i | \mathbf{x}_i, \mathbf{w}) \right]
 \end{aligned}$$



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 weights $\mathbf{w}$ that makes $p(\mathbf{x}, y | \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$:

$$\begin{aligned} \mathbf{w}^* &= \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w})) = \arg \min_{\mathbf{w}} \int p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y | \mathbf{w})} d(\mathbf{x}, y) \\ &= \arg \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}, y) \sim p_{\text{data}}(\mathbf{x}, y)} \left[\log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y | \mathbf{w})} \right] = \arg \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}, y)} \left[\log p_{\text{data}}(\mathbf{x}, y) - \log p(y | \mathbf{x}, \mathbf{w}) \right] \\ &= \arg \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}, y)} \left[-\log p(y | \mathbf{w}, \mathbf{x}) \right] \approx \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} \left[-\log p(y_i | \mathbf{x}_i, \mathbf{w}) \right] \dots \text{KL justification} \end{aligned}$$



$$\left(\text{MLE: } = \arg \max_{\mathbf{w}} \prod_{(\mathbf{x}_i, y_i)} p(y_i | \mathbf{x}_i, \mathbf{w}) \right)$$

$$\mathbf{w}^\star = \arg \min_{\mathbf{w}} D_{KL}(\textcolor{red}{p}_{\text{data}}(\mathbf{x}, y) \parallel \textcolor{blue}{p}(\mathbf{x}, y \mid \mathbf{w})) \approx \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim \textcolor{red}{p}_{\text{data}}(\mathbf{x}, y)} [-\log \textcolor{blue}{p}(y_i \mid \mathbf{x}_i, \mathbf{w})]$$

Regression

$$\textcolor{blue}{p}(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(y; f(\mathbf{x}, \mathbf{w}), \sigma^2) = K \exp \frac{(f(\mathbf{x}, \mathbf{w}) - y)^2}{2\sigma^2}$$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

Classification

$$\textcolor{blue}{p}(y \mid \mathbf{x}, \mathbf{w}) = \sigma(f(\mathbf{x}, \mathbf{w}))$$

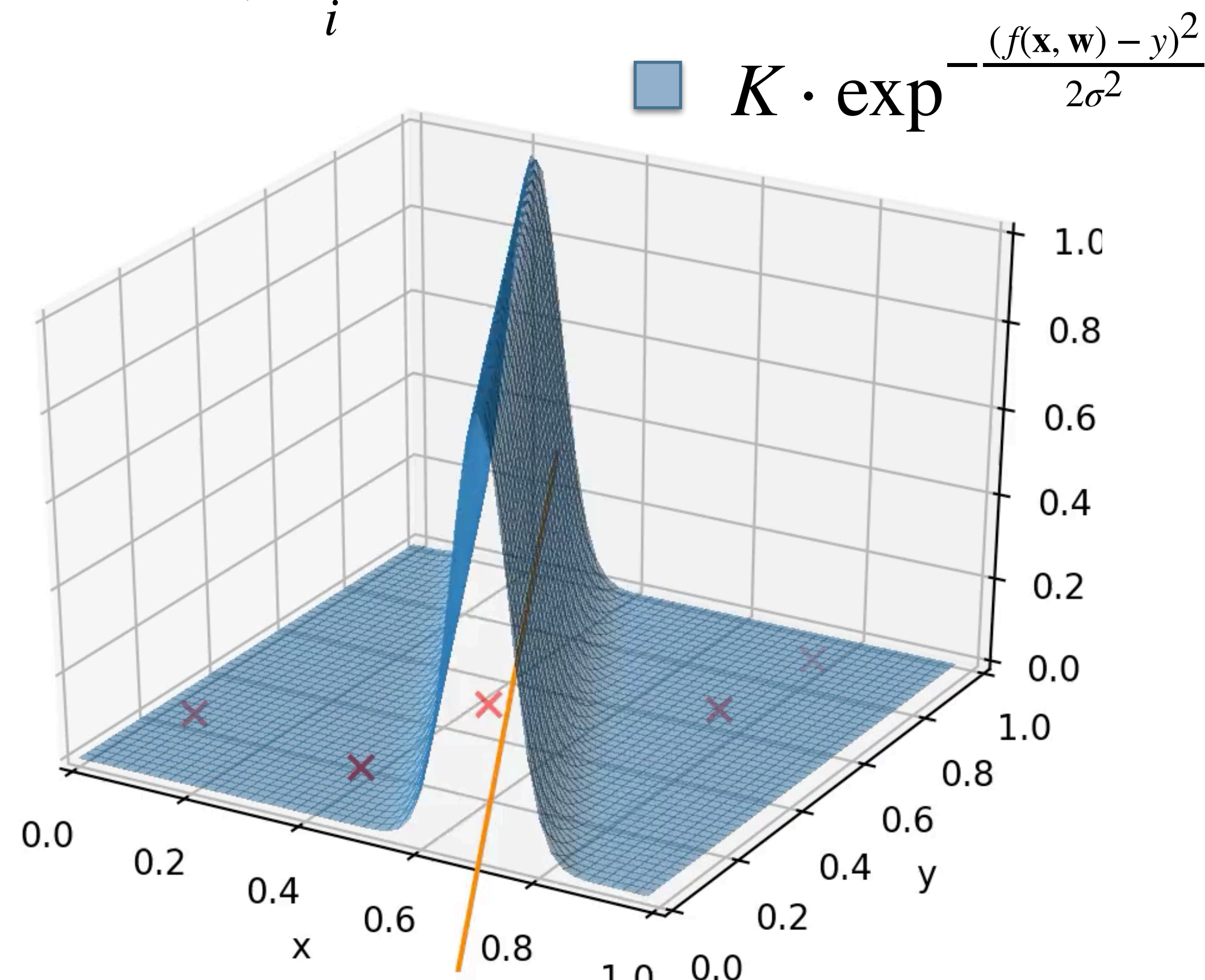
$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_i \log \left(1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w})) \right)$$

$$\mathbf{w}^{\star} = \arg \min_{\mathbf{w}} D_{KL}(\textcolor{red}{p}_{\text{data}}(\mathbf{x}, y) \parallel \textcolor{blue}{p}(\mathbf{x}, y | \mathbf{w})) \approx \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim \textcolor{red}{p}_{\text{data}}(\mathbf{x}, y)} [-\log \textcolor{blue}{p}(y_i | \mathbf{x}_i, \mathbf{w})]$$

Regression

$$\textcolor{blue}{p}(y | \mathbf{x}, \mathbf{w}) = \mathcal{N}(y; f(\mathbf{x}, \mathbf{w}), \sigma^2) = K \exp^{-\frac{(f(\mathbf{x}, \mathbf{w}) - y)^2}{2\sigma^2}}$$

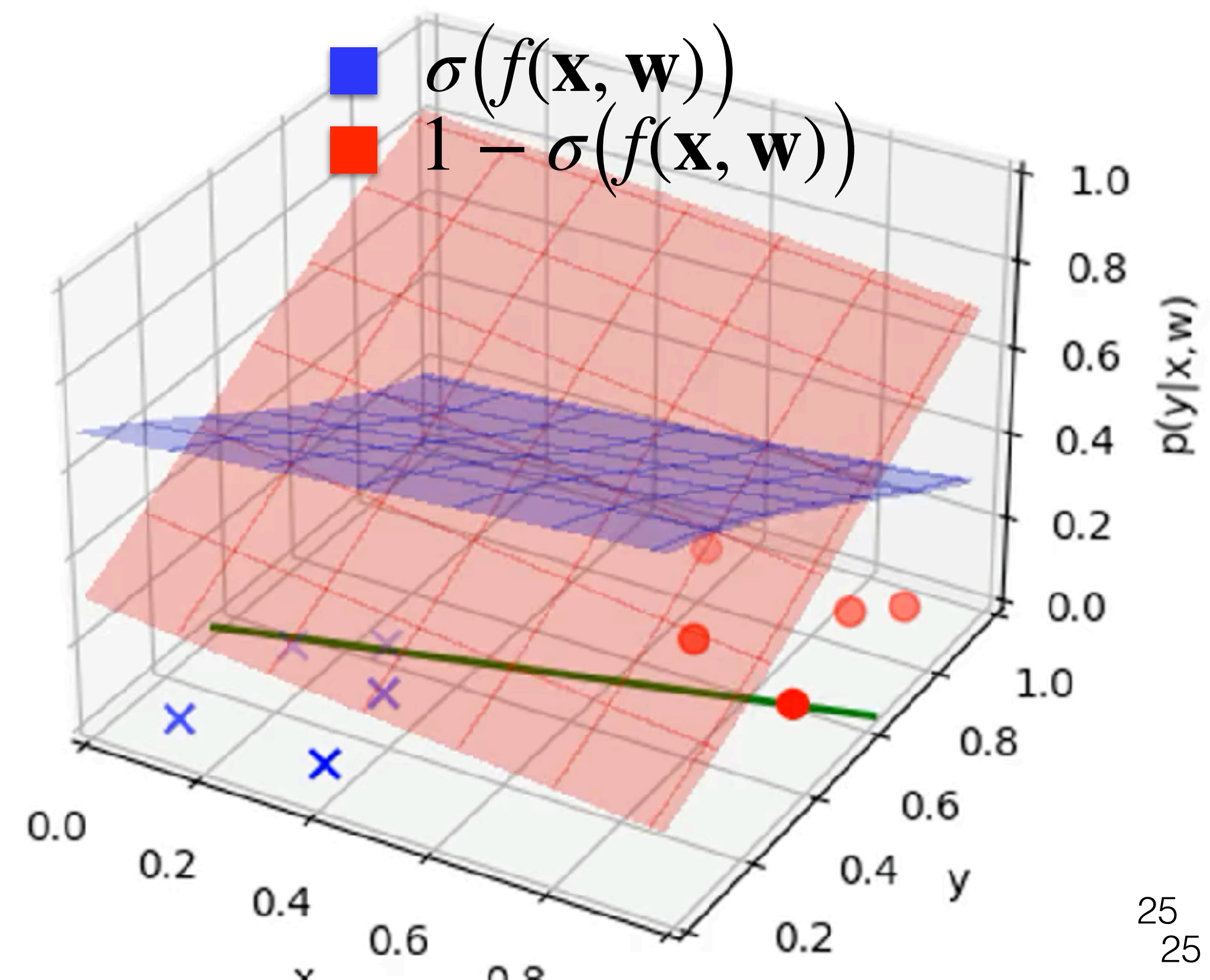
$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$



Classification

$$\textcolor{blue}{p}(y | \mathbf{x}, \mathbf{w}) = \sigma(f(\mathbf{x}, \mathbf{w}))$$

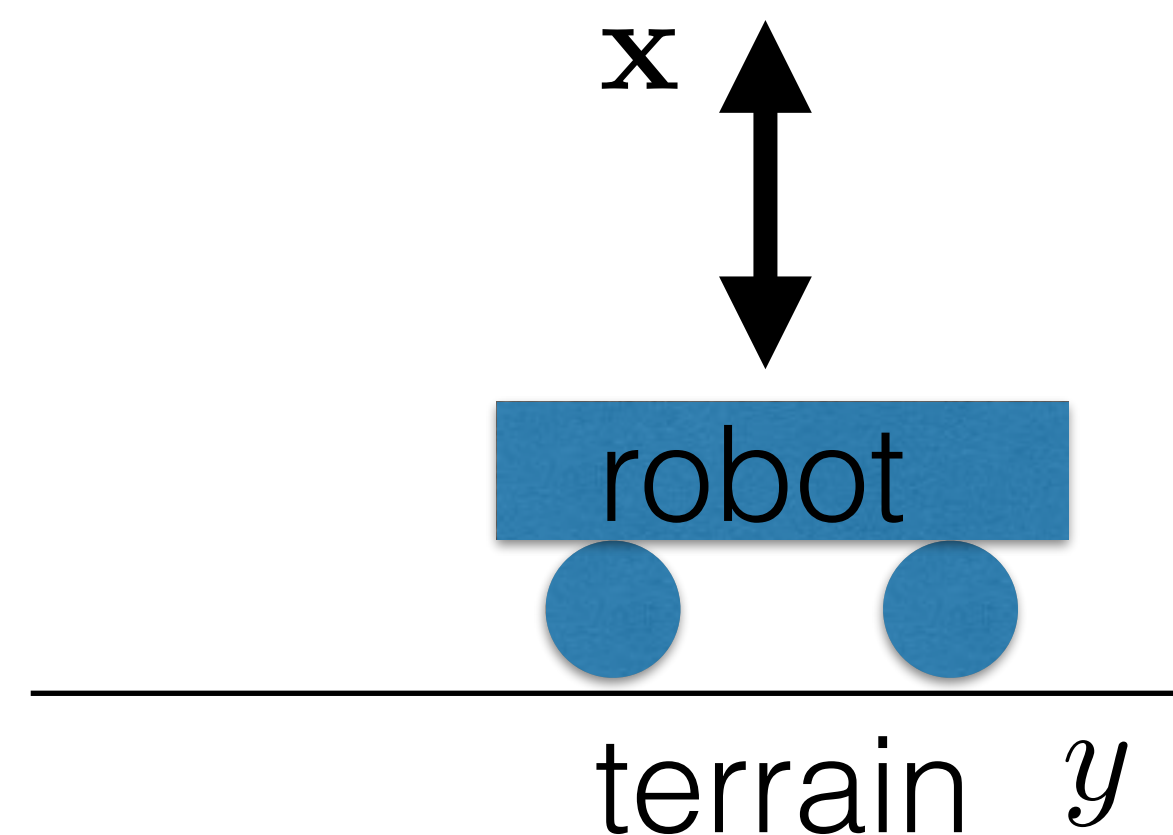
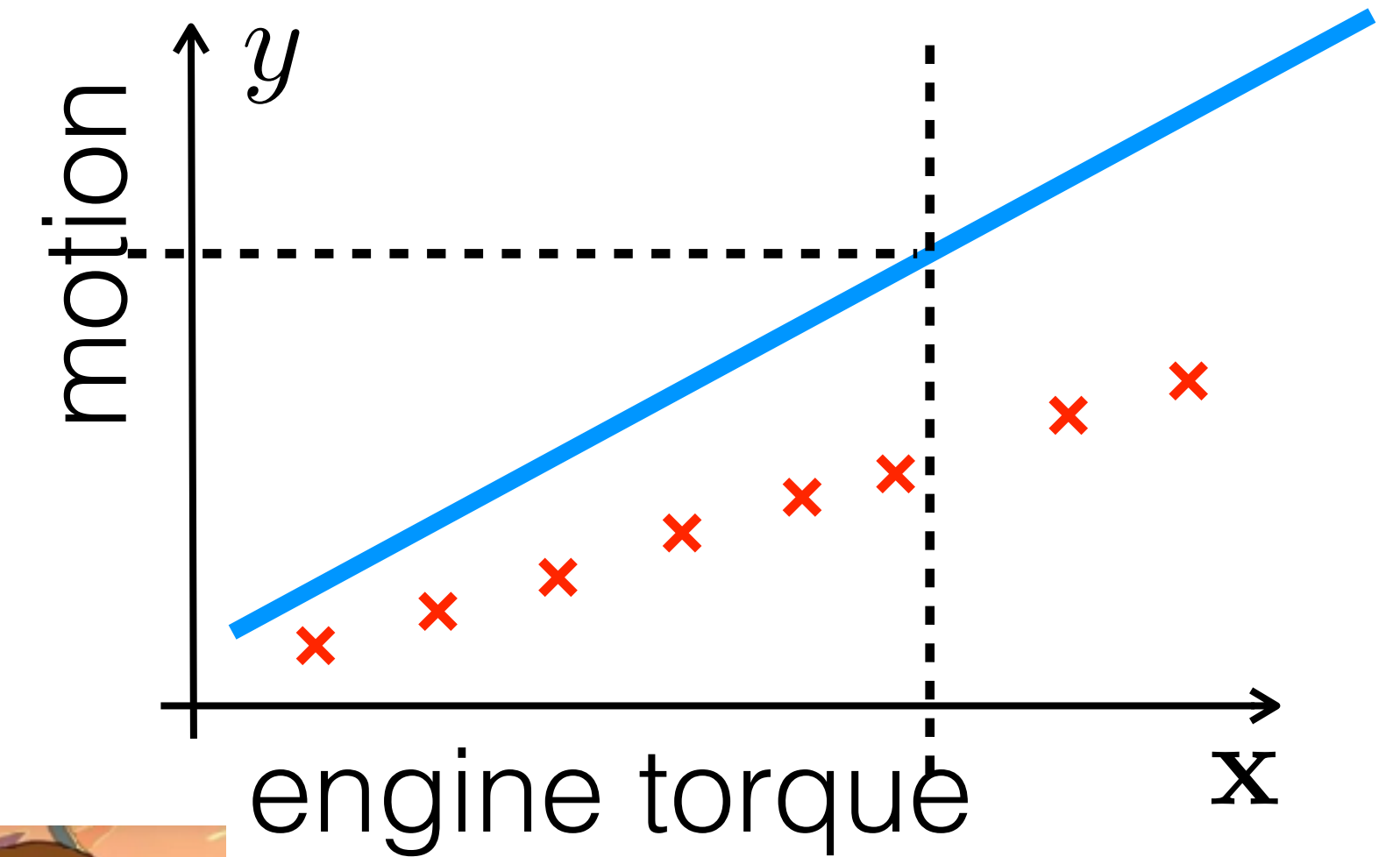
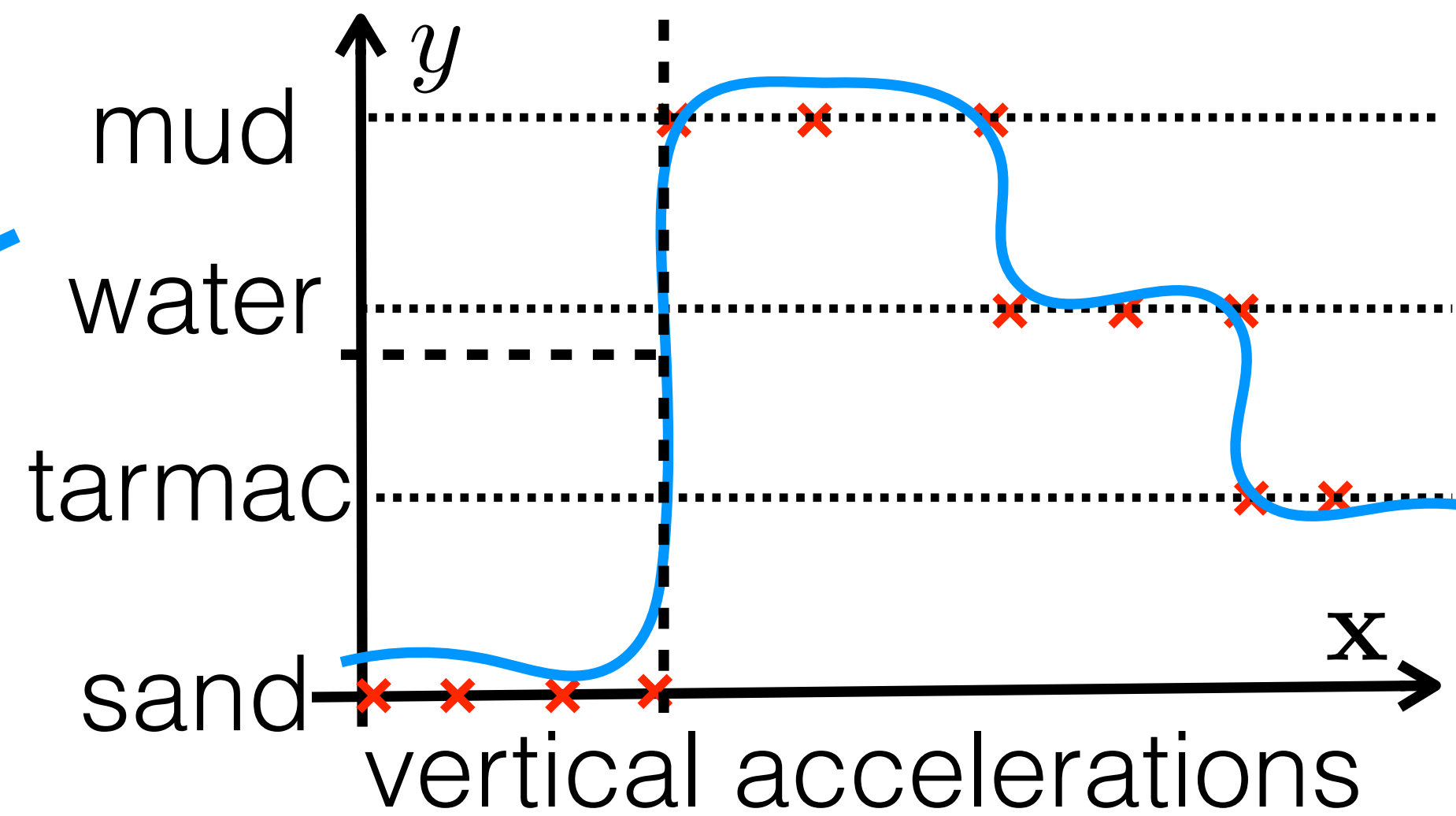
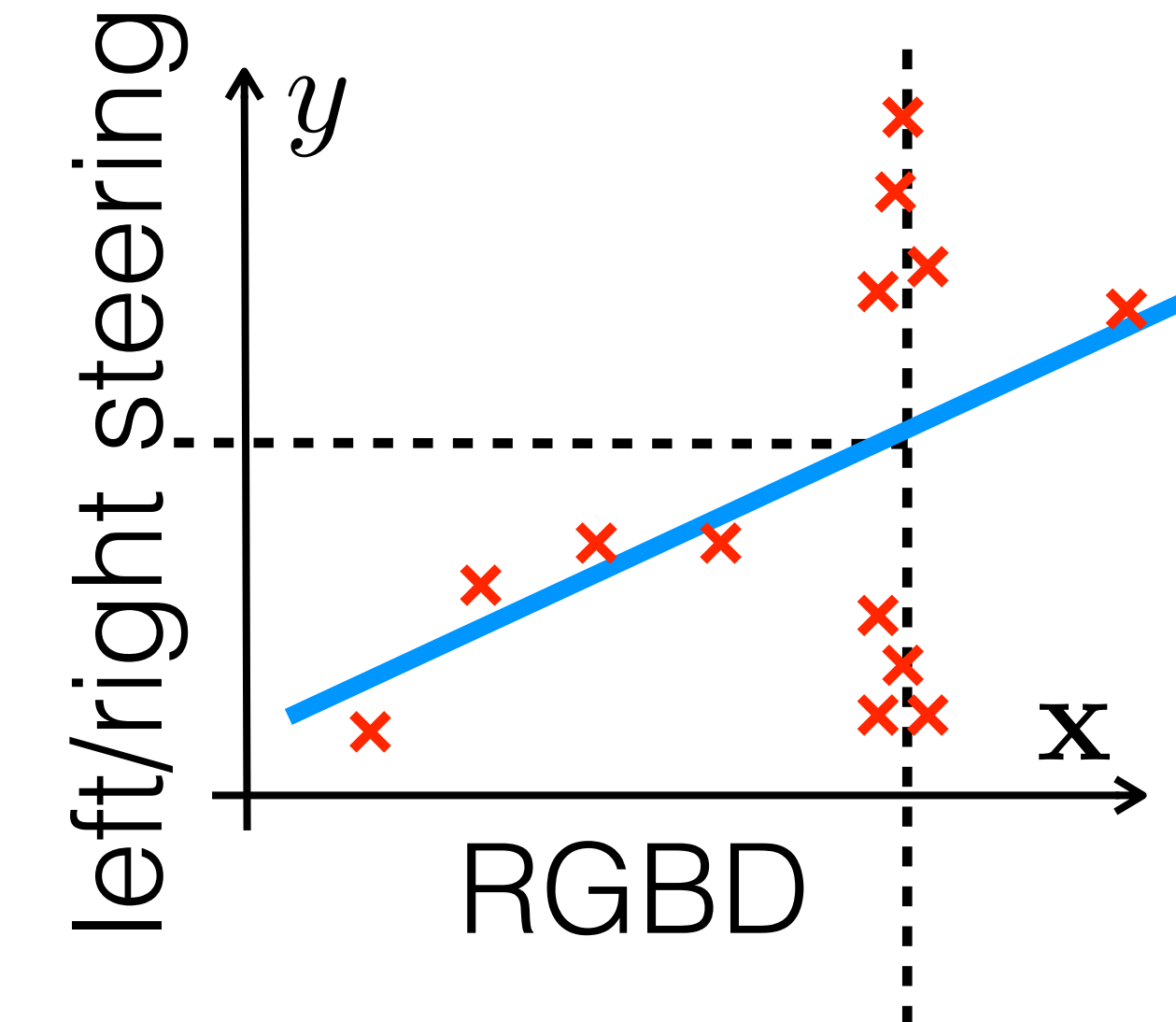
$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_i \log \left(1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w})) \right)$$



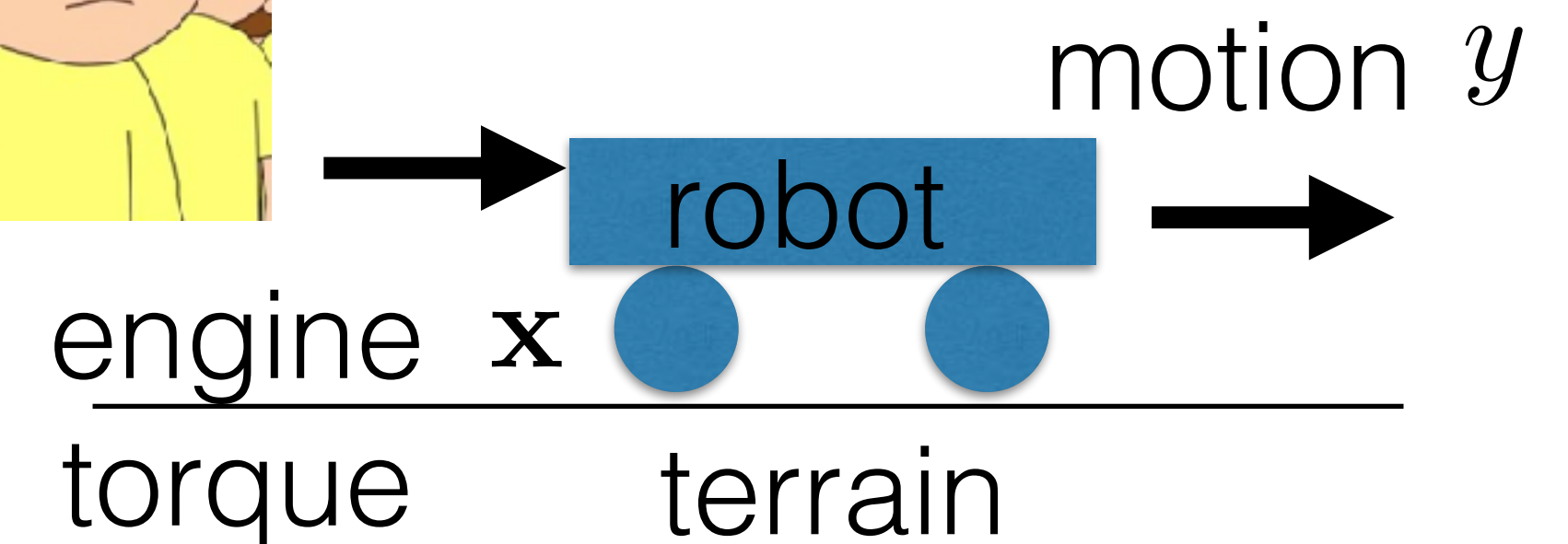
Lec 01: what can go wrong?

What is the joint source of these problems???

outlier

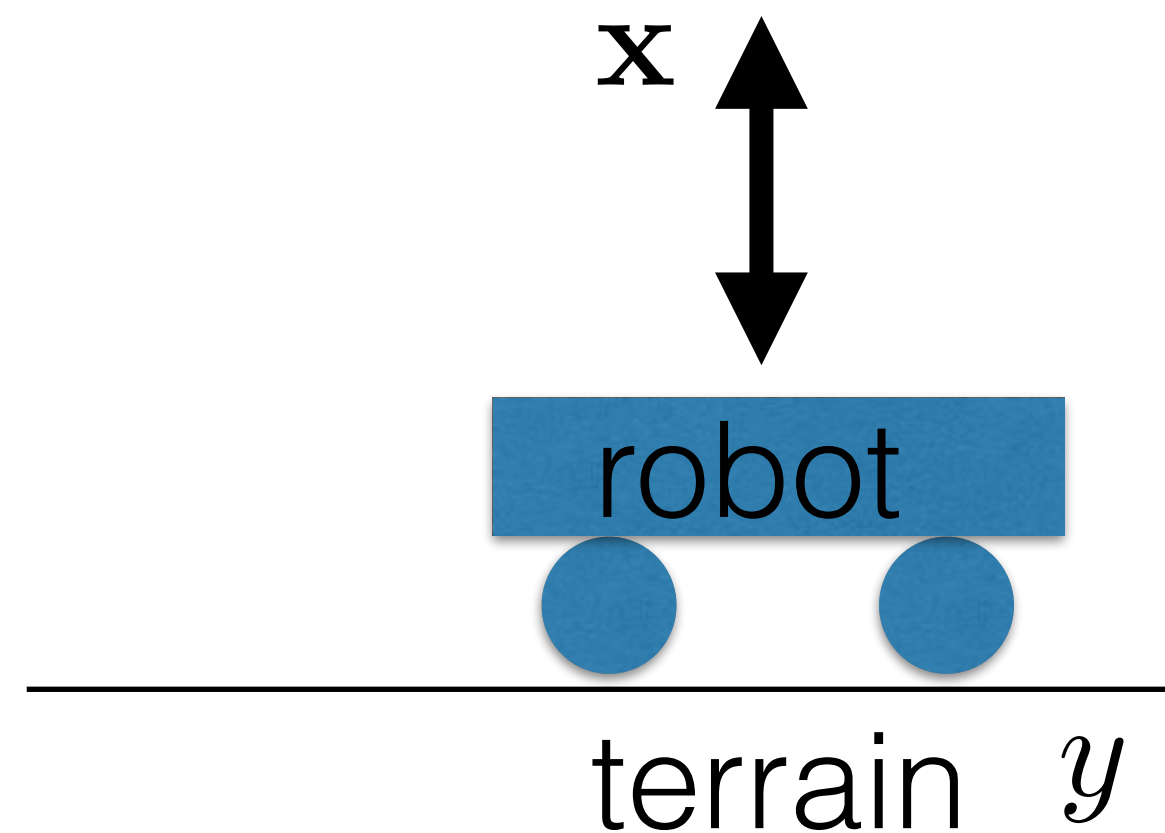
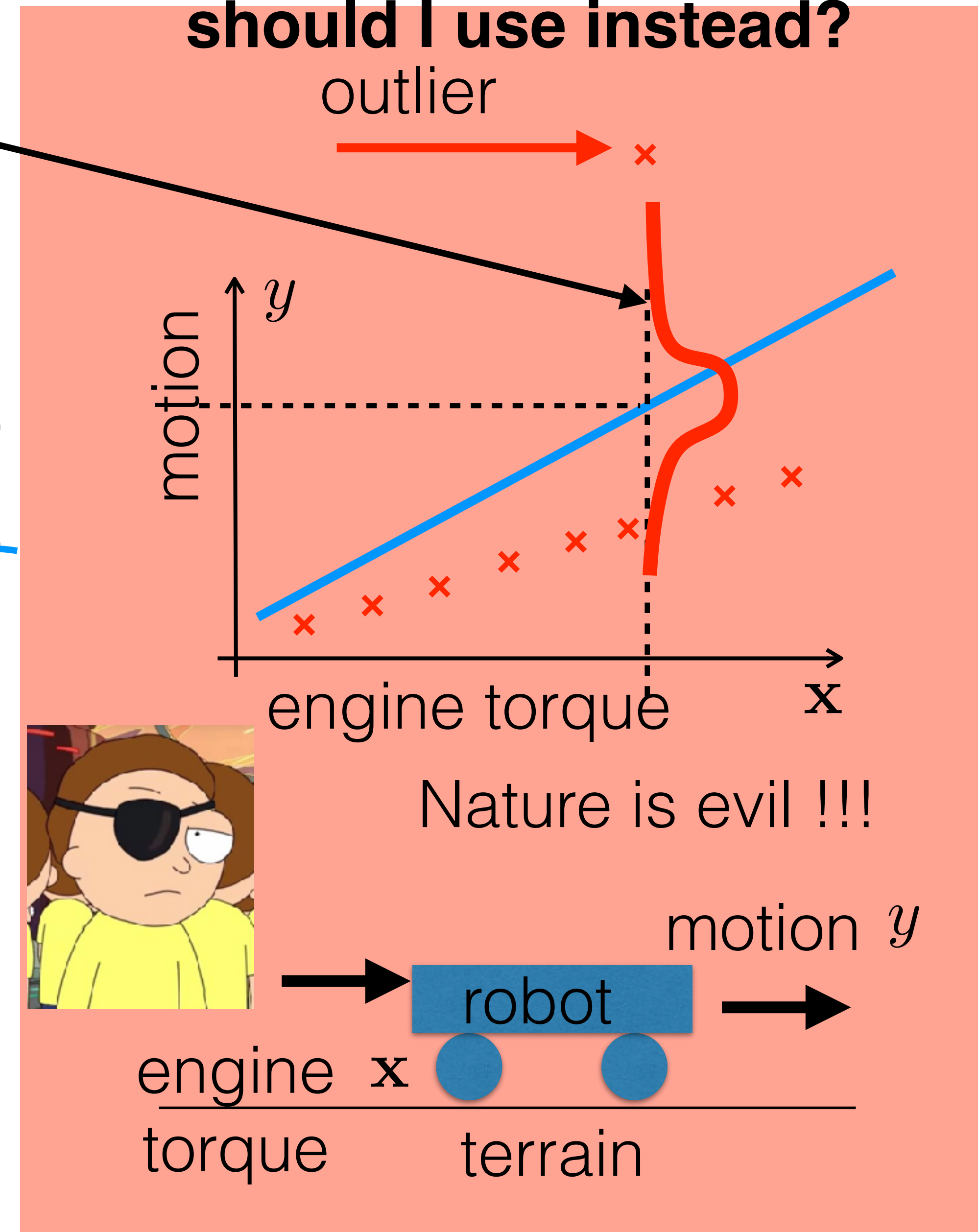
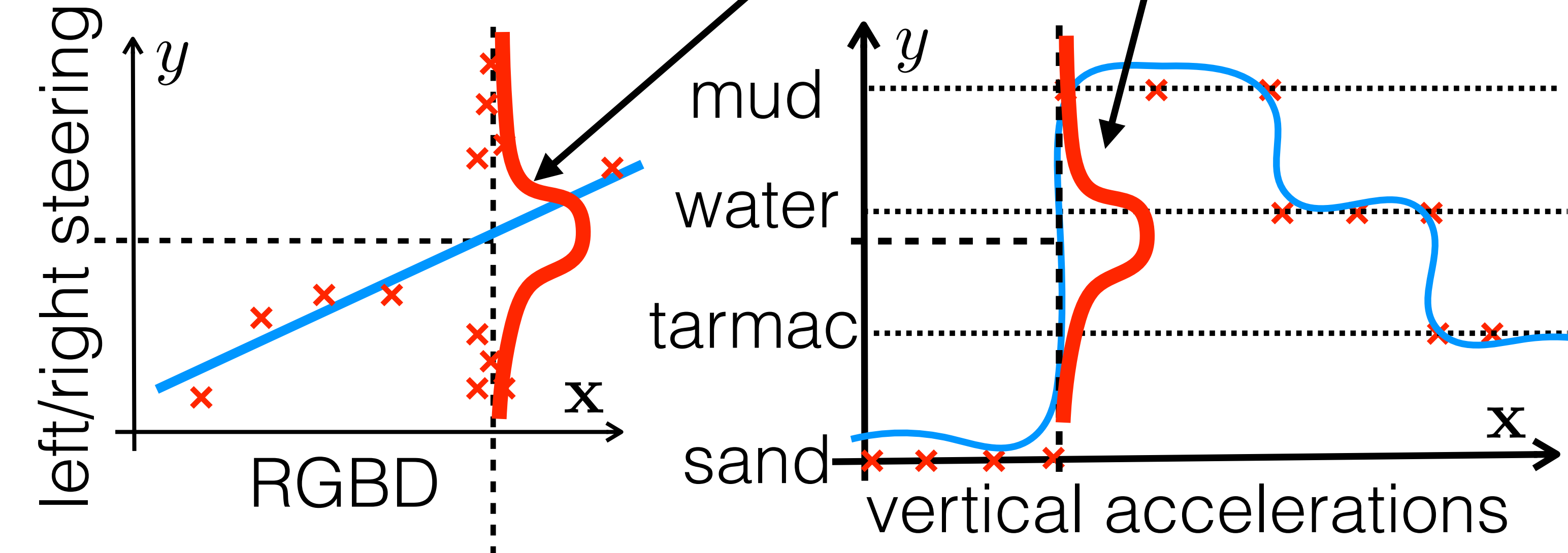


Nature is evil !!!

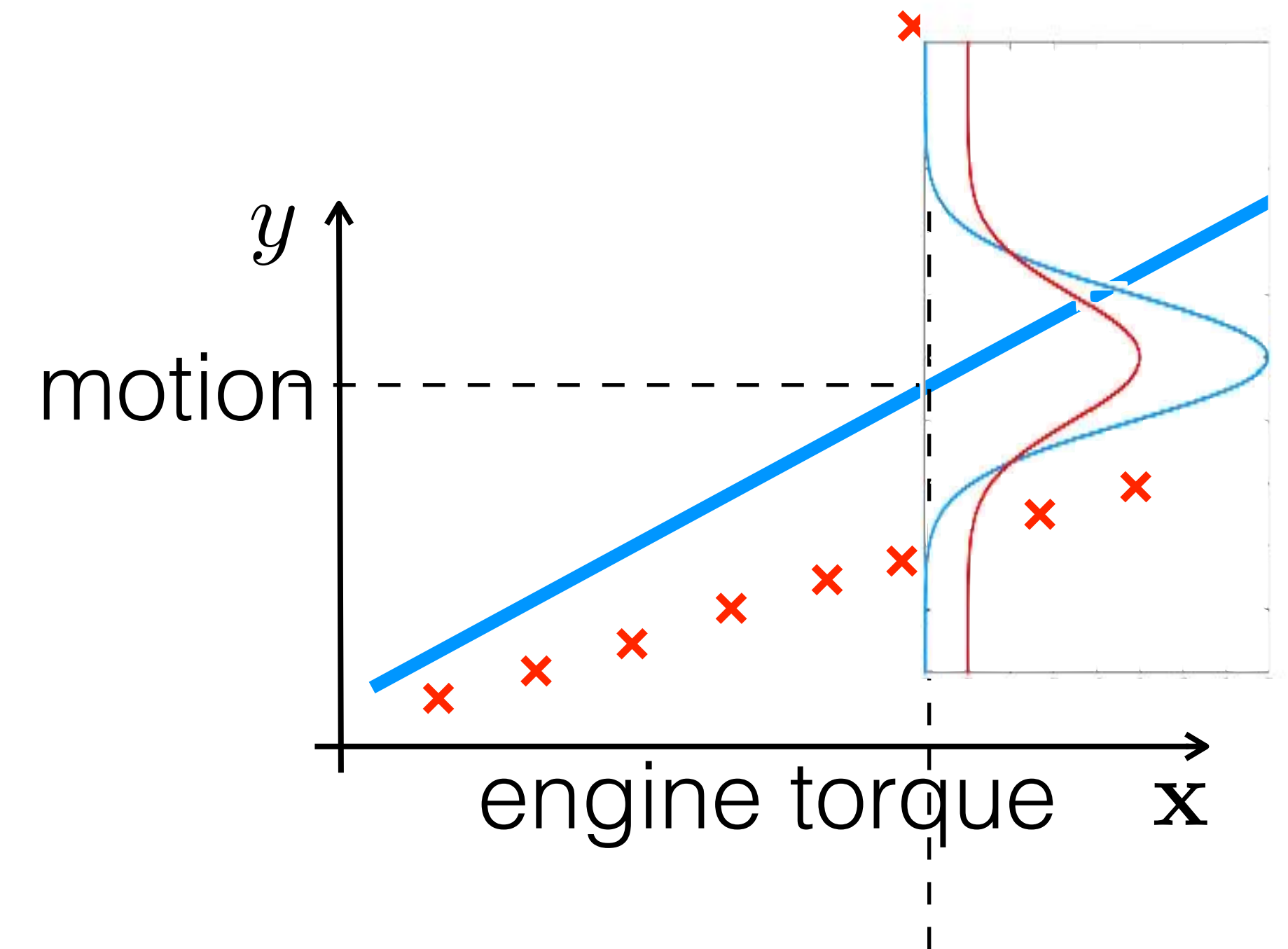


**Which distribution
should I use instead?**

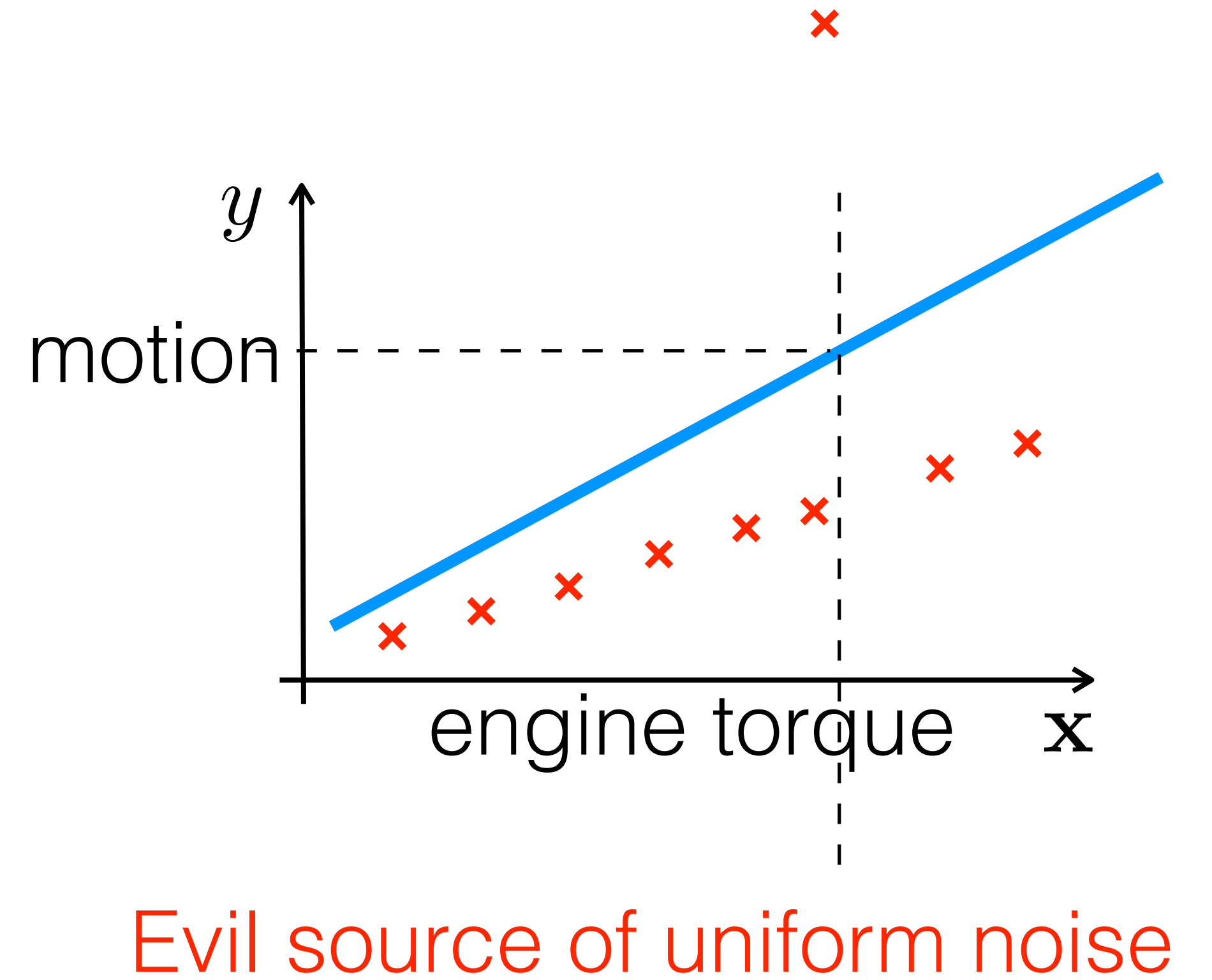
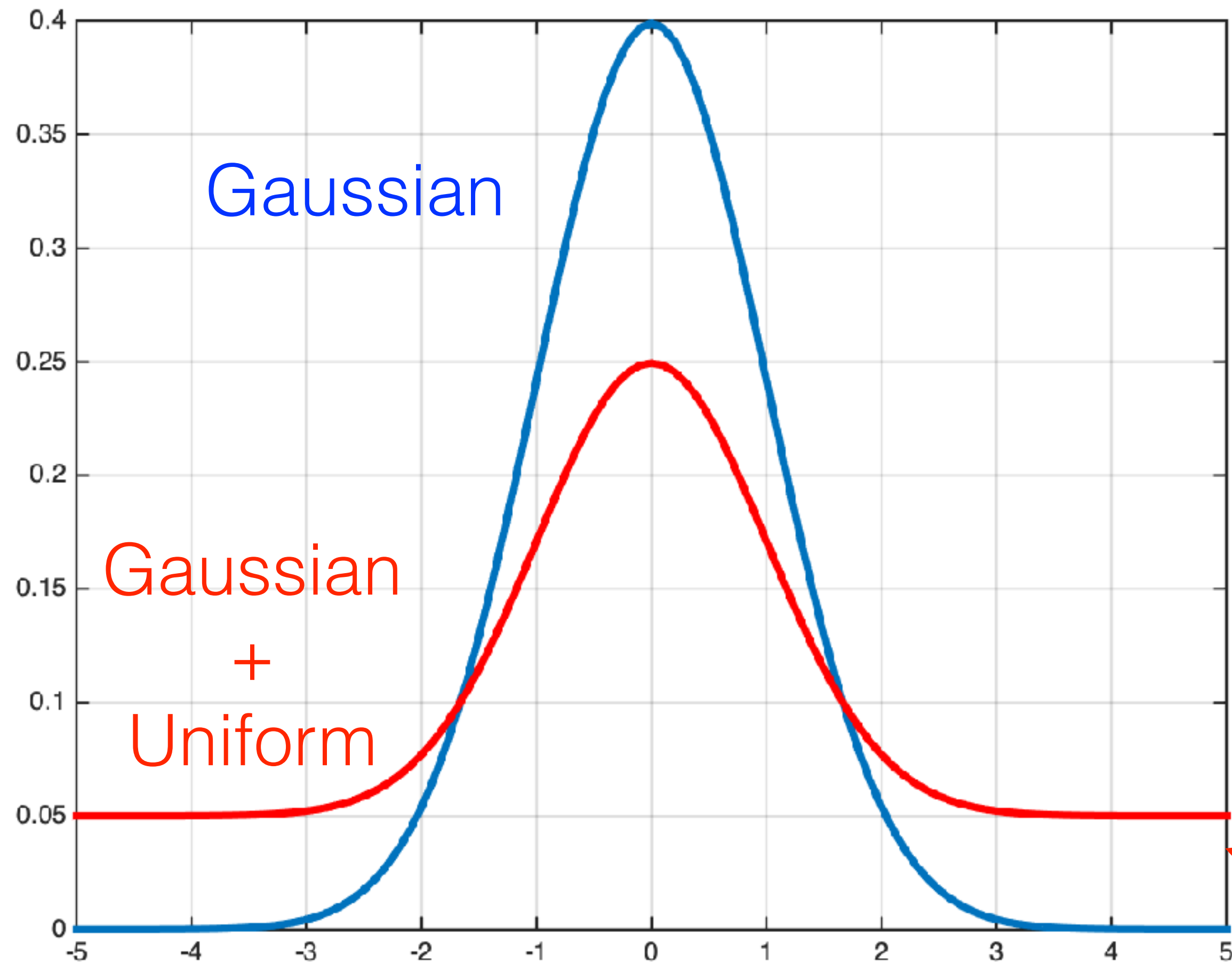
This is the problem!



Robust regression



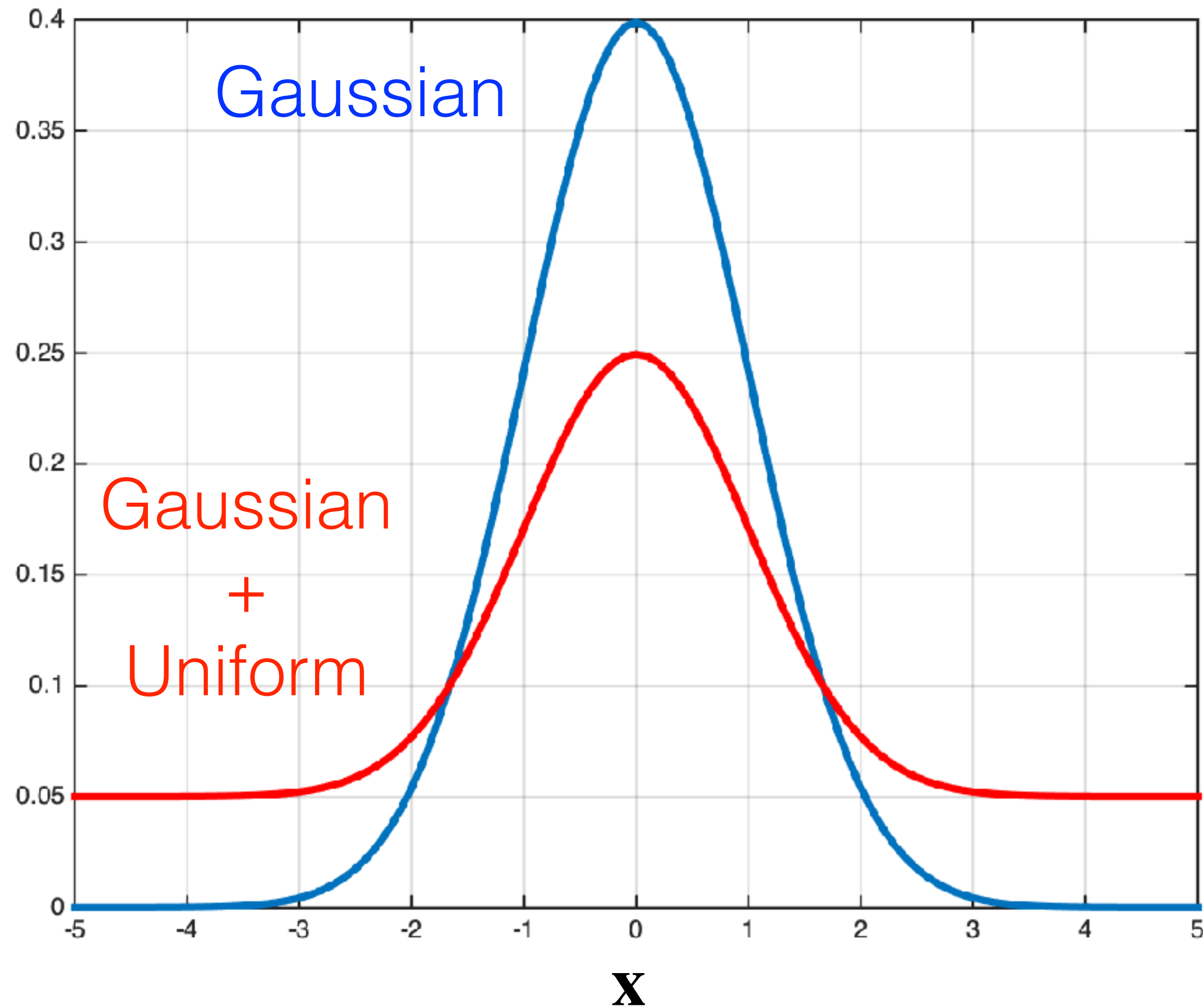
Robust regression



Robust regression

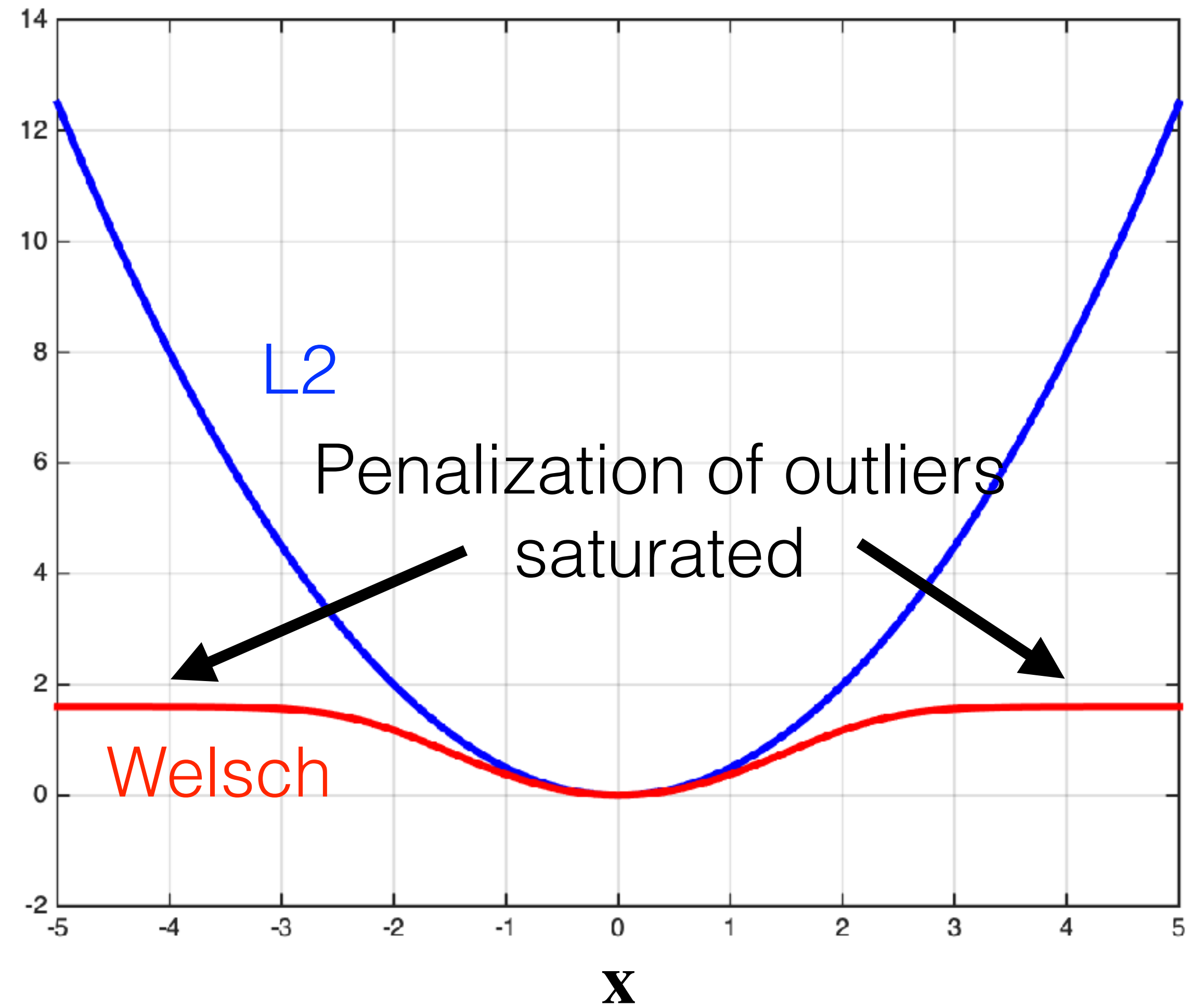
$$p(y | \mathbf{x}, \mathbf{w})$$

Probability distributions



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

Corresponding losses



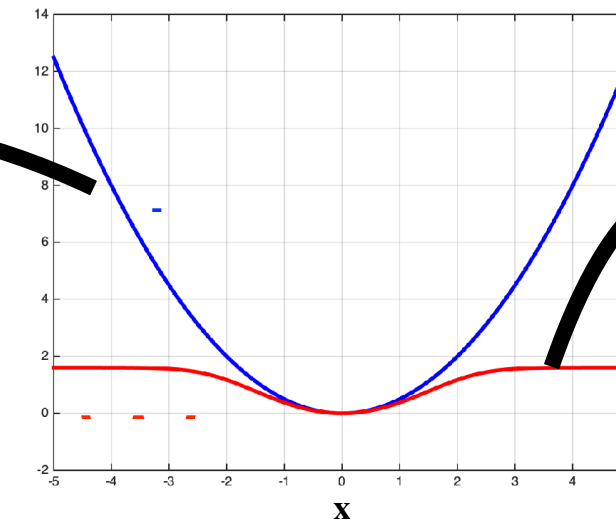
Can you guess where another problem appears?



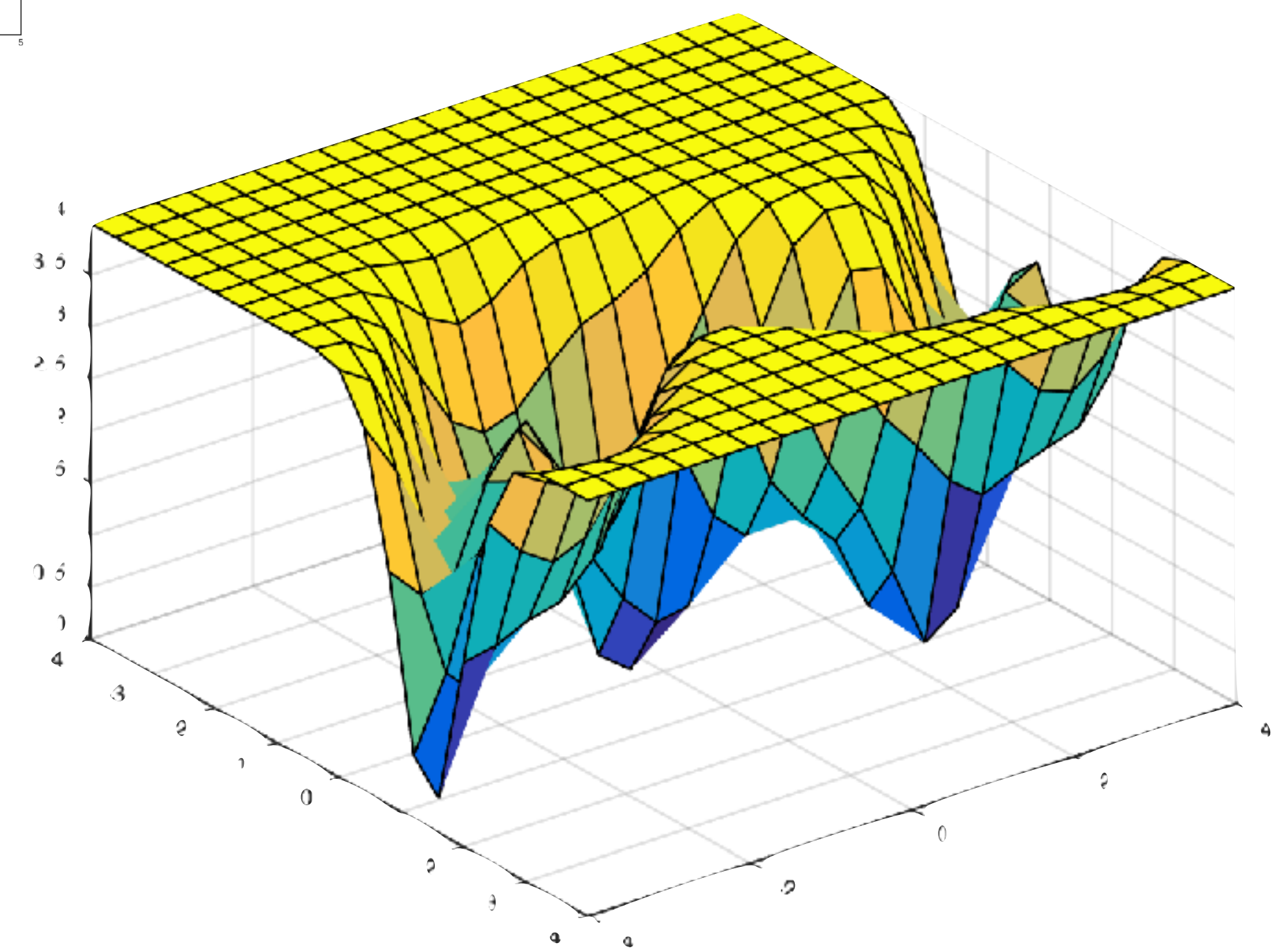
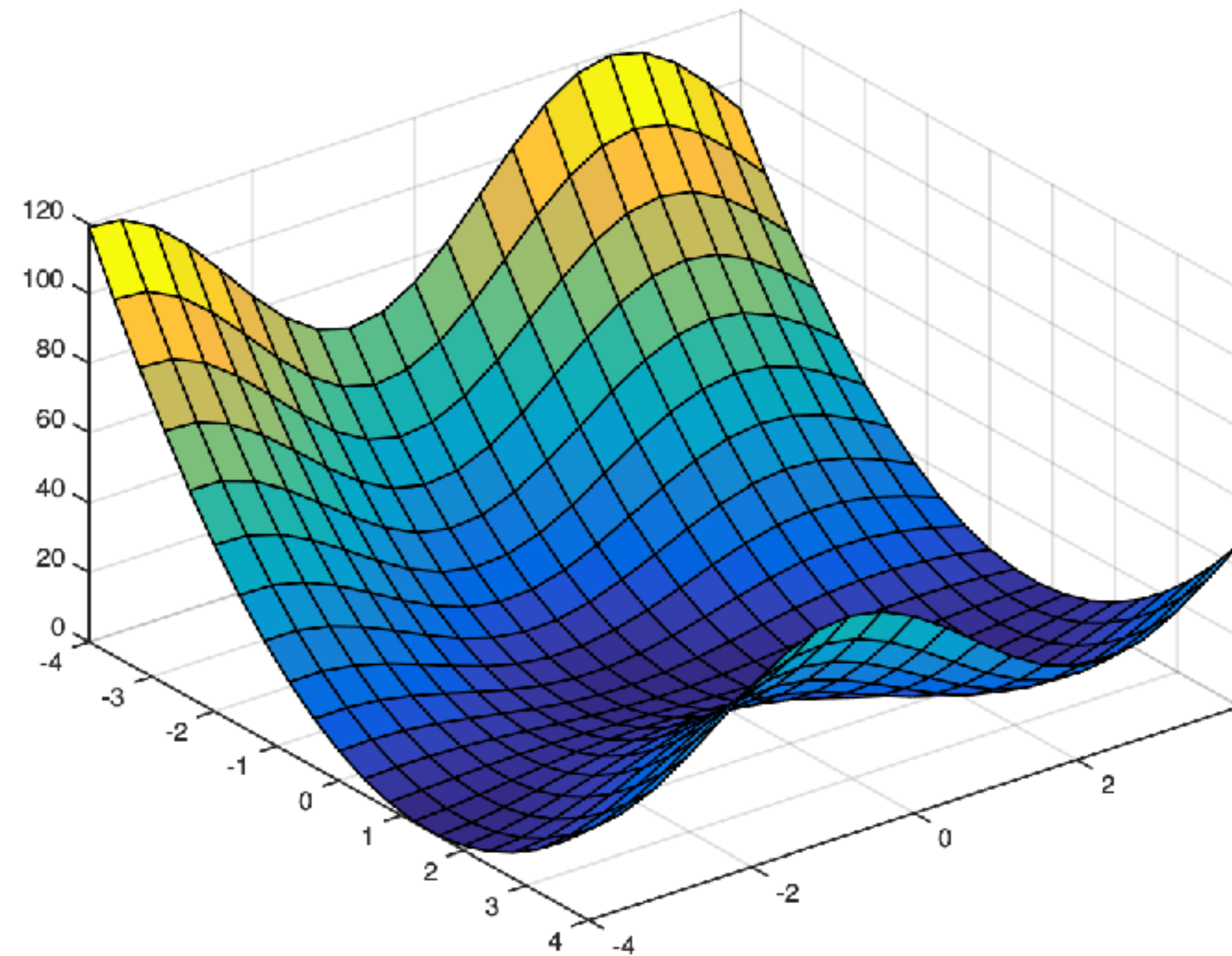
L2 landscape

Robust regression

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



Welsch landscape



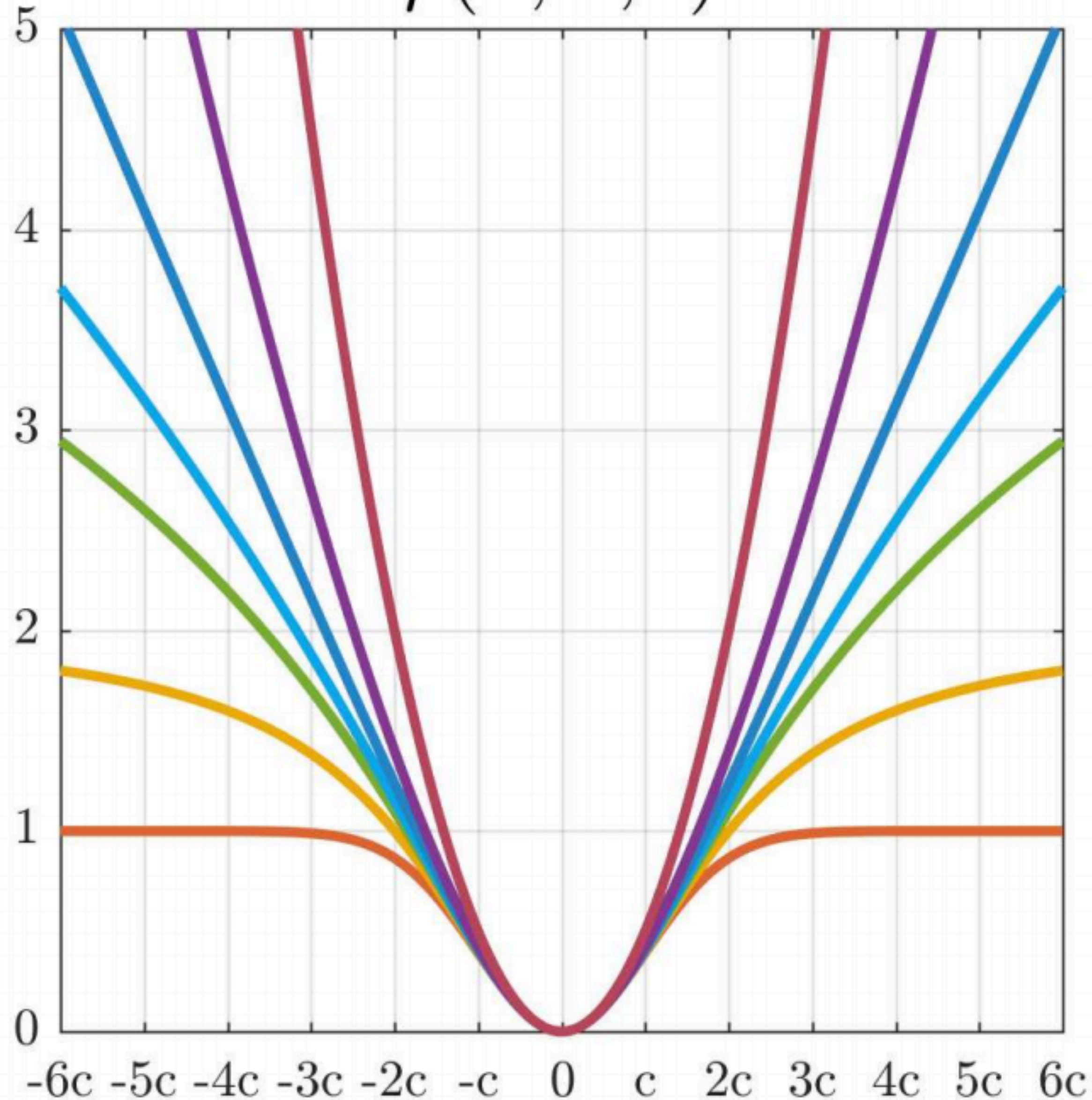
- **Uniform noise not modeled**
- GD-friendly landscape
- Gradient length encodes distance
- Easy to optimize

- **Uniform noise modeled**
- 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>

$\rho(x, \alpha, c)$



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

Trade-off:

Robustness to uniform noise (outliers)
VS

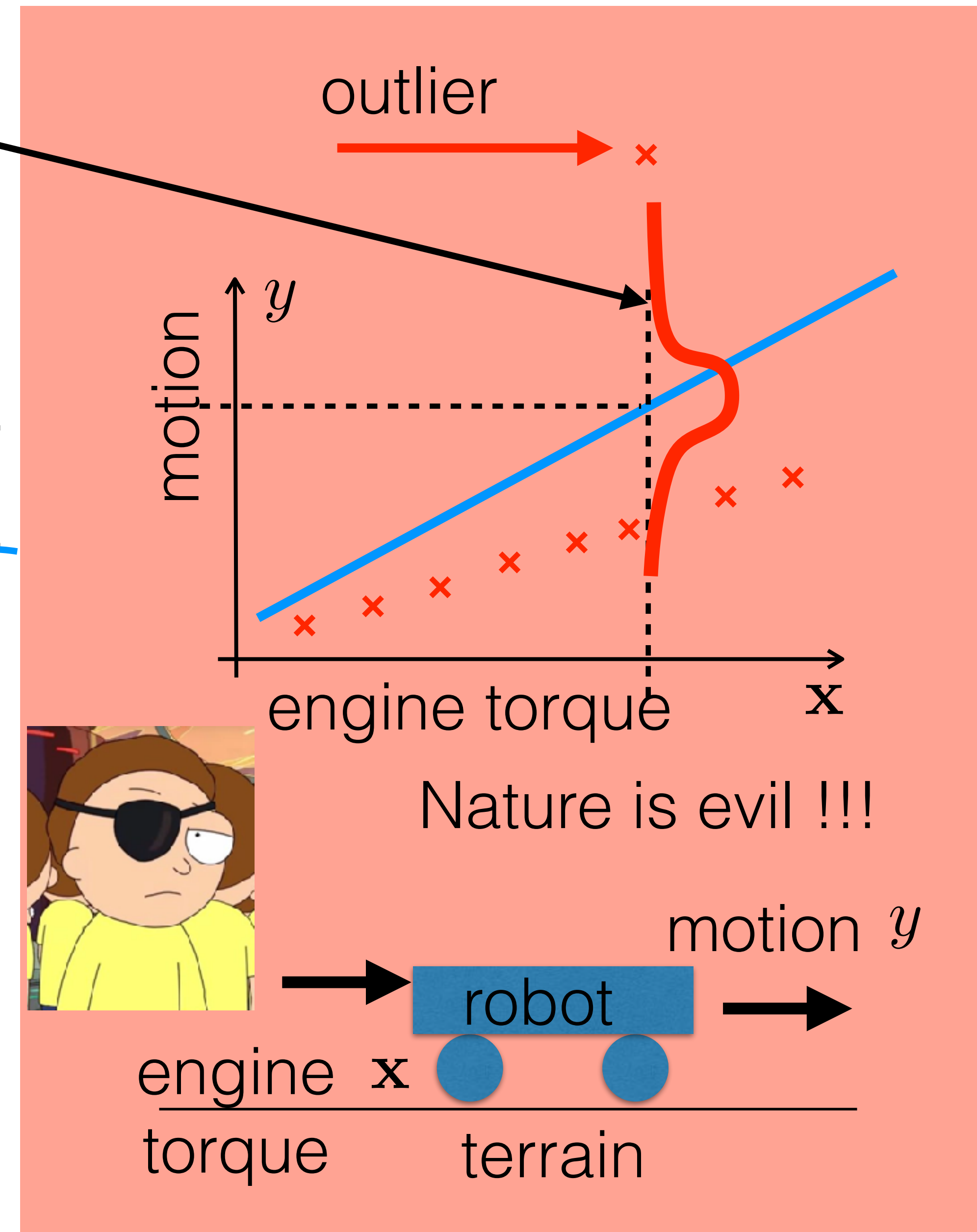
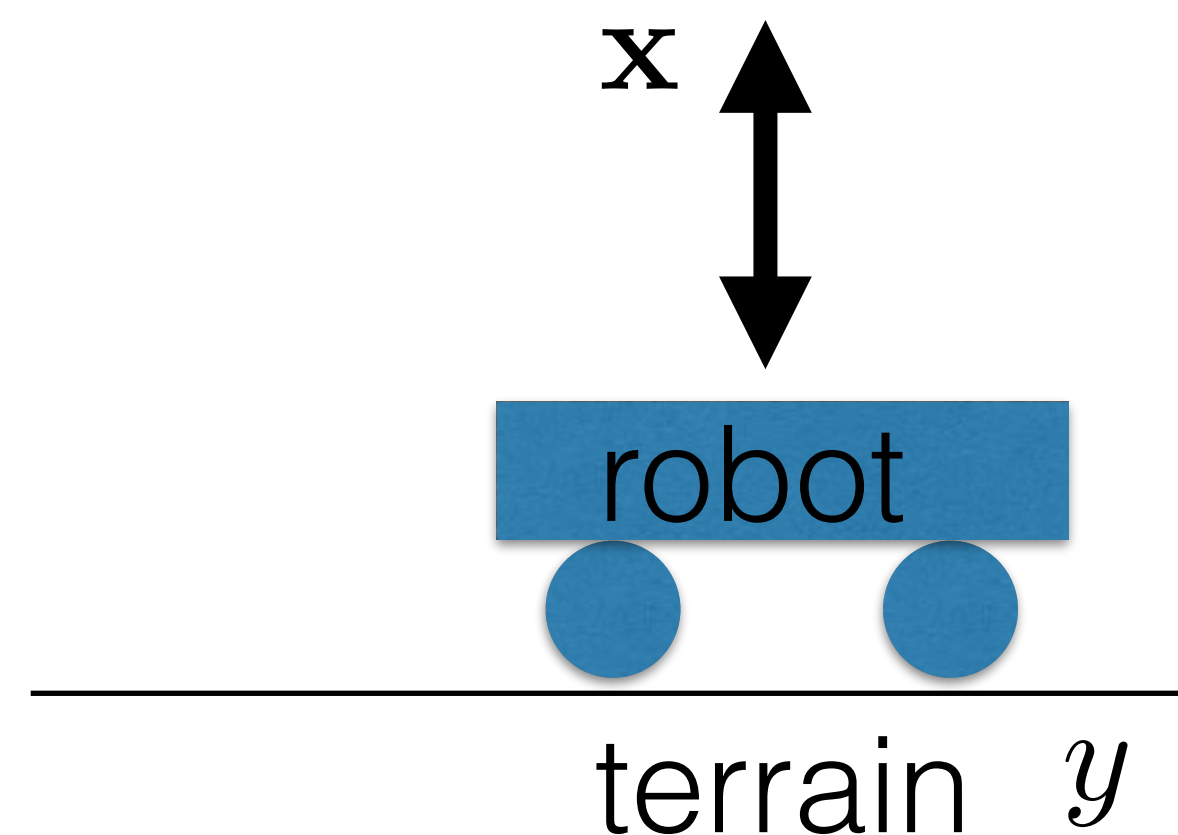
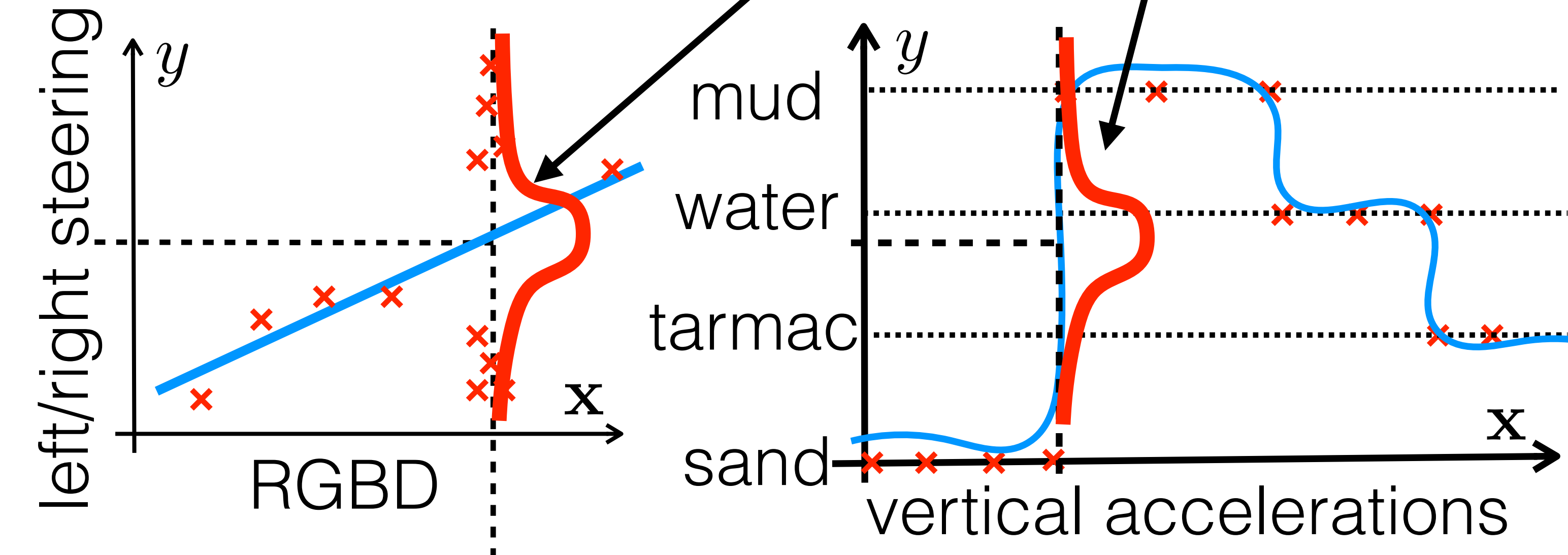
Optimization-friendly landscape

The best what you can do:

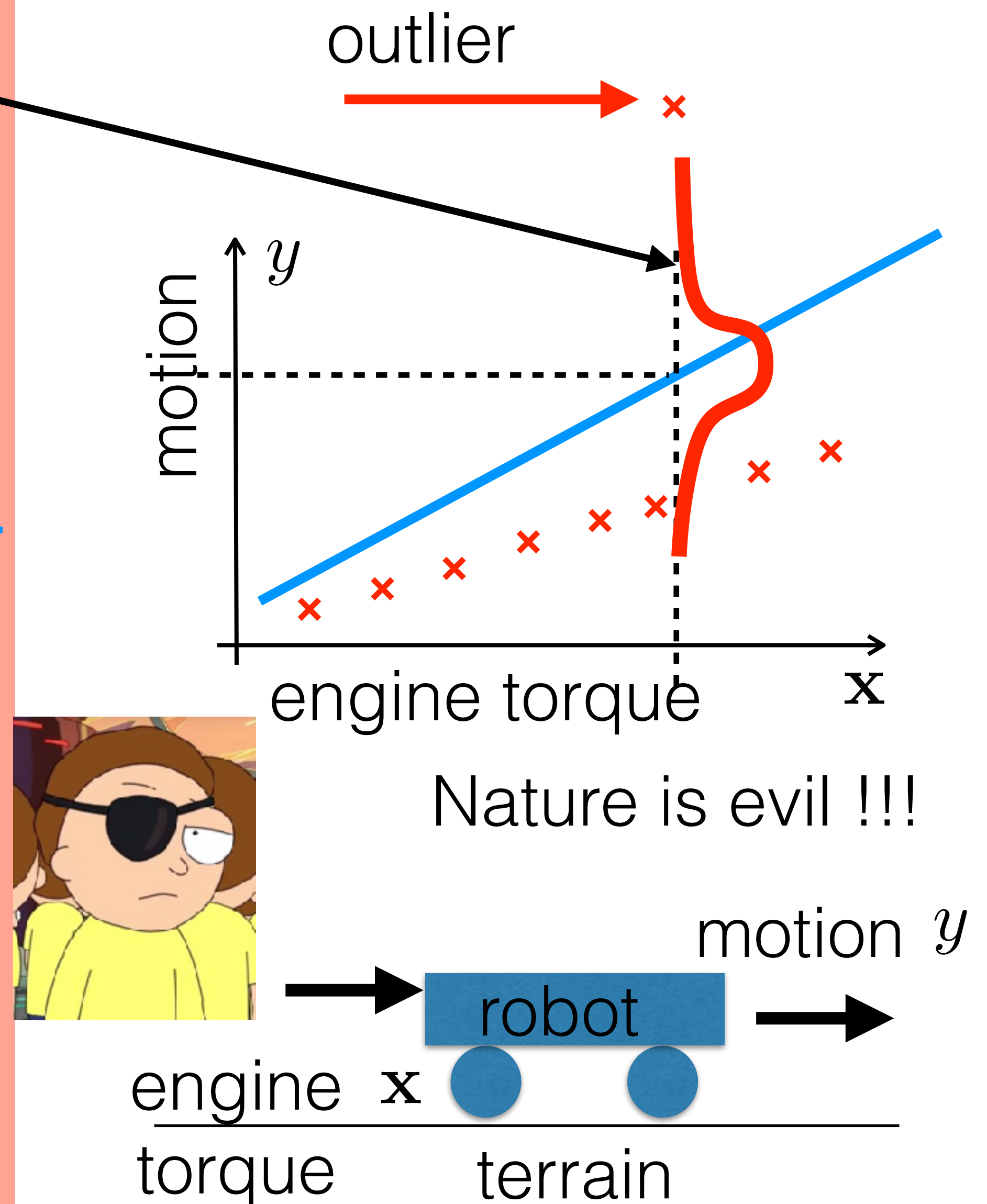
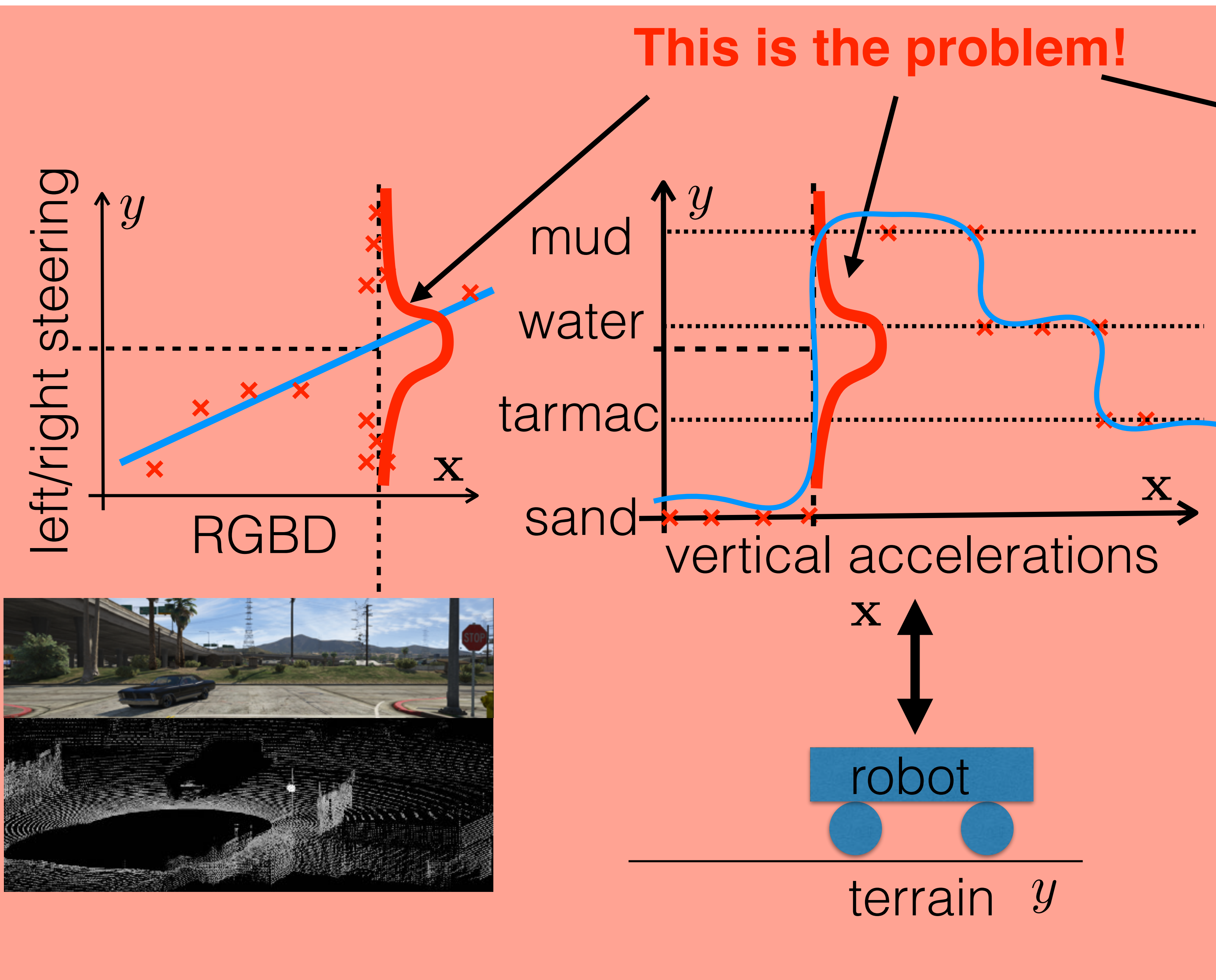


What can go wrong: inappropriate choice of loss function

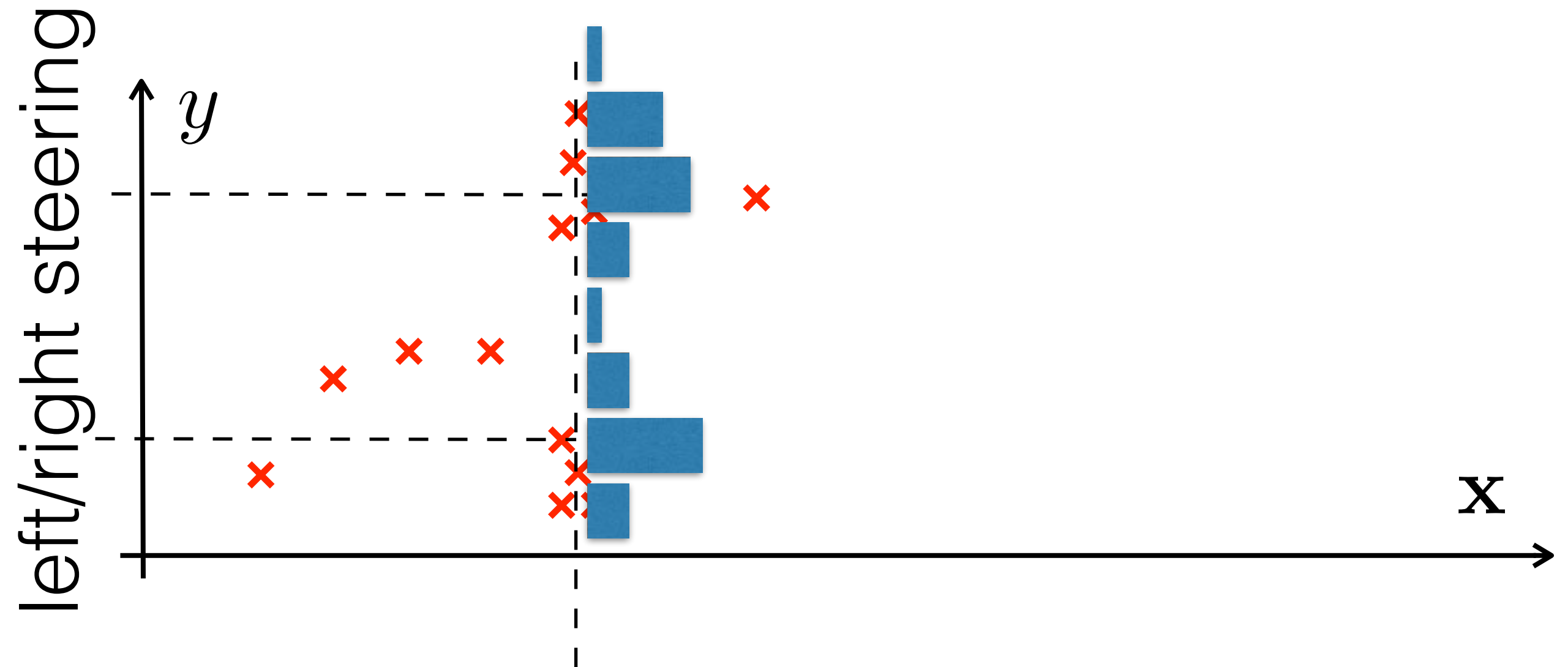
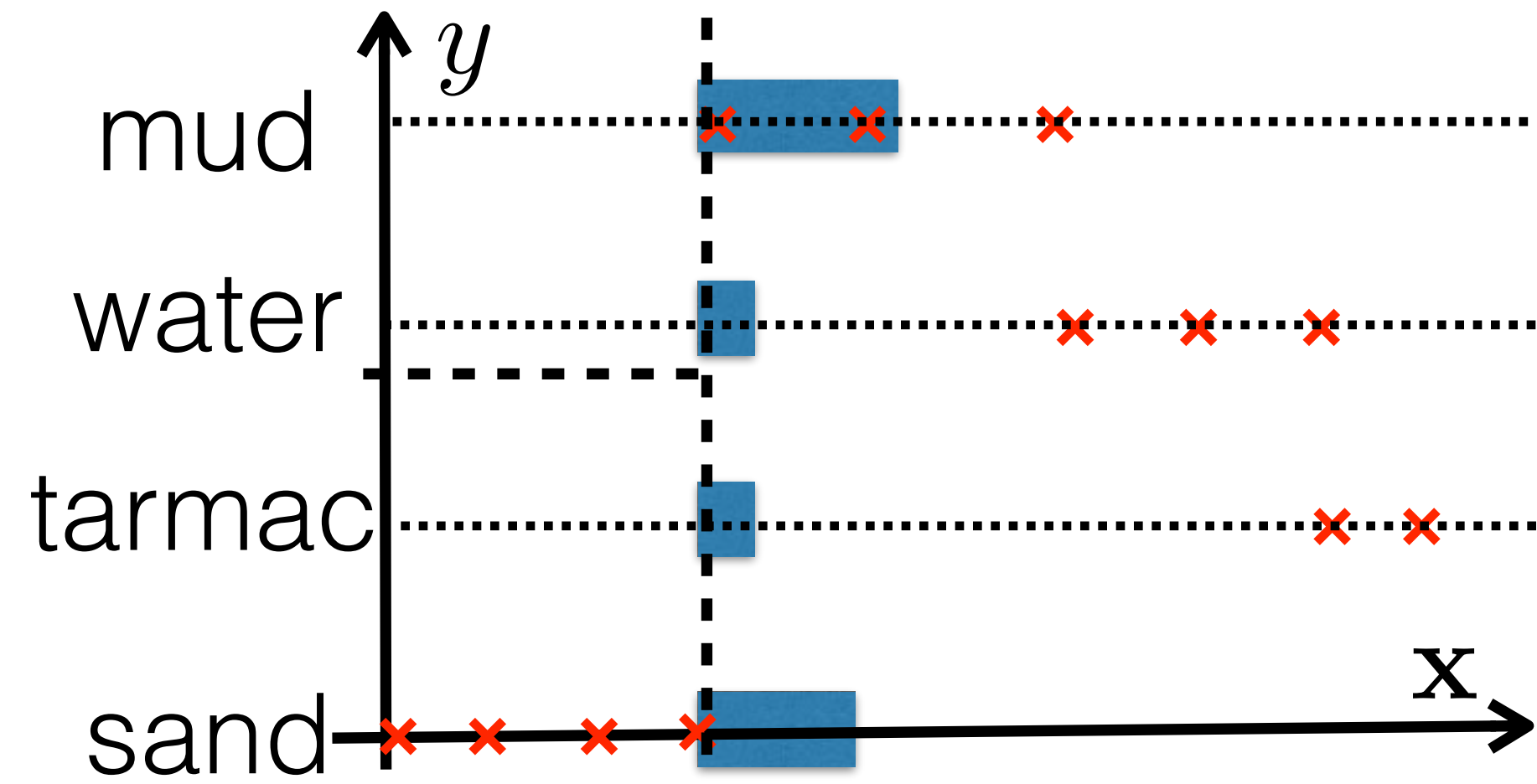
This is the problem!



What can go wrong: inappropriate choice of loss function



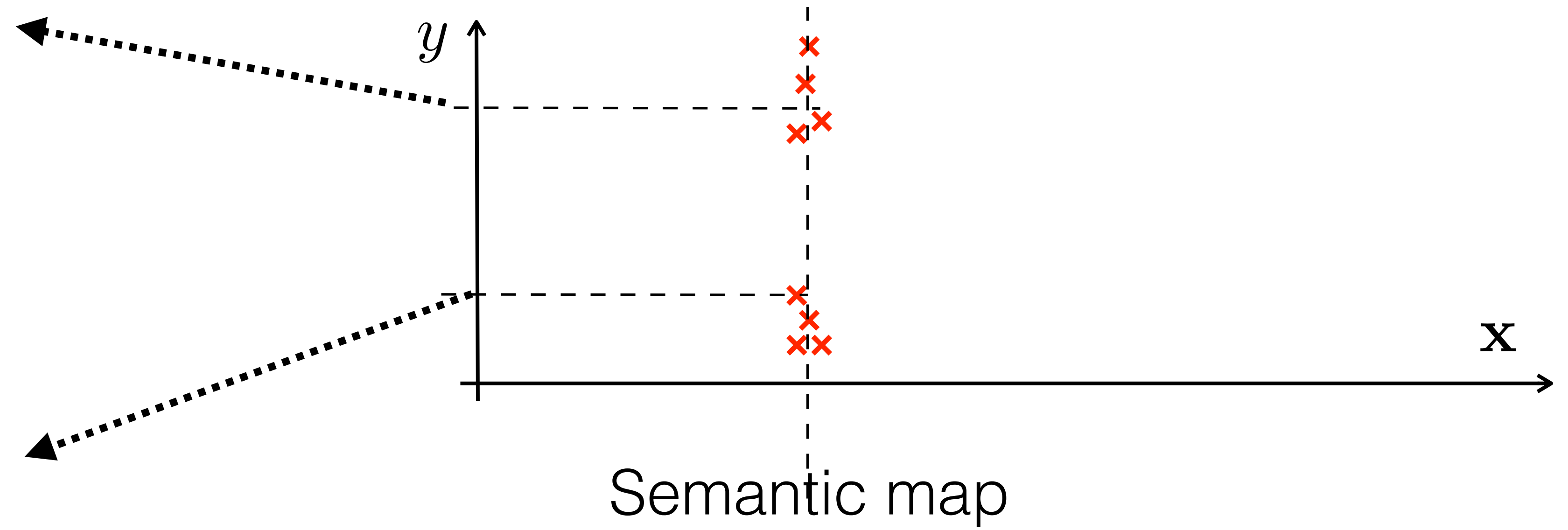
Work-around 1: discretize y-domain and treat the problem as classification



Can I use it always?

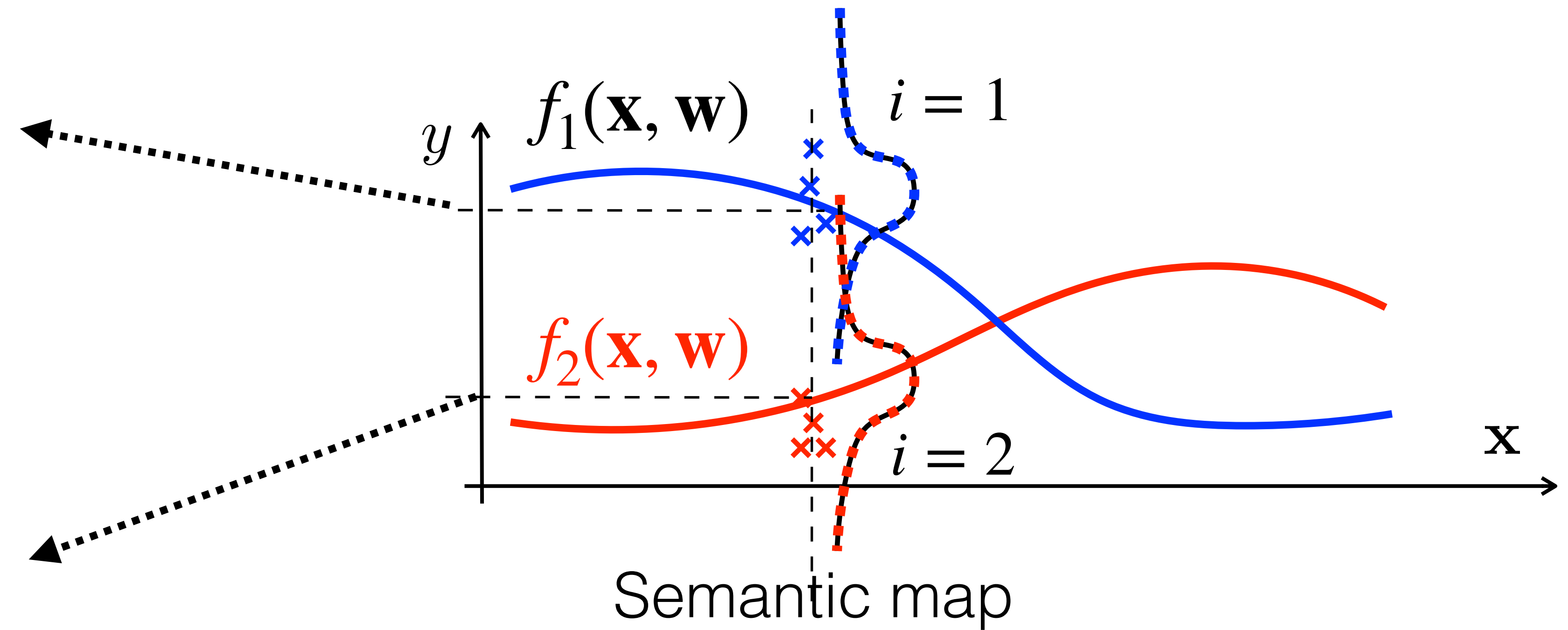
Tractable only for low-dim y





What if y are images?

Work-around 2: allow multiple hypothesis

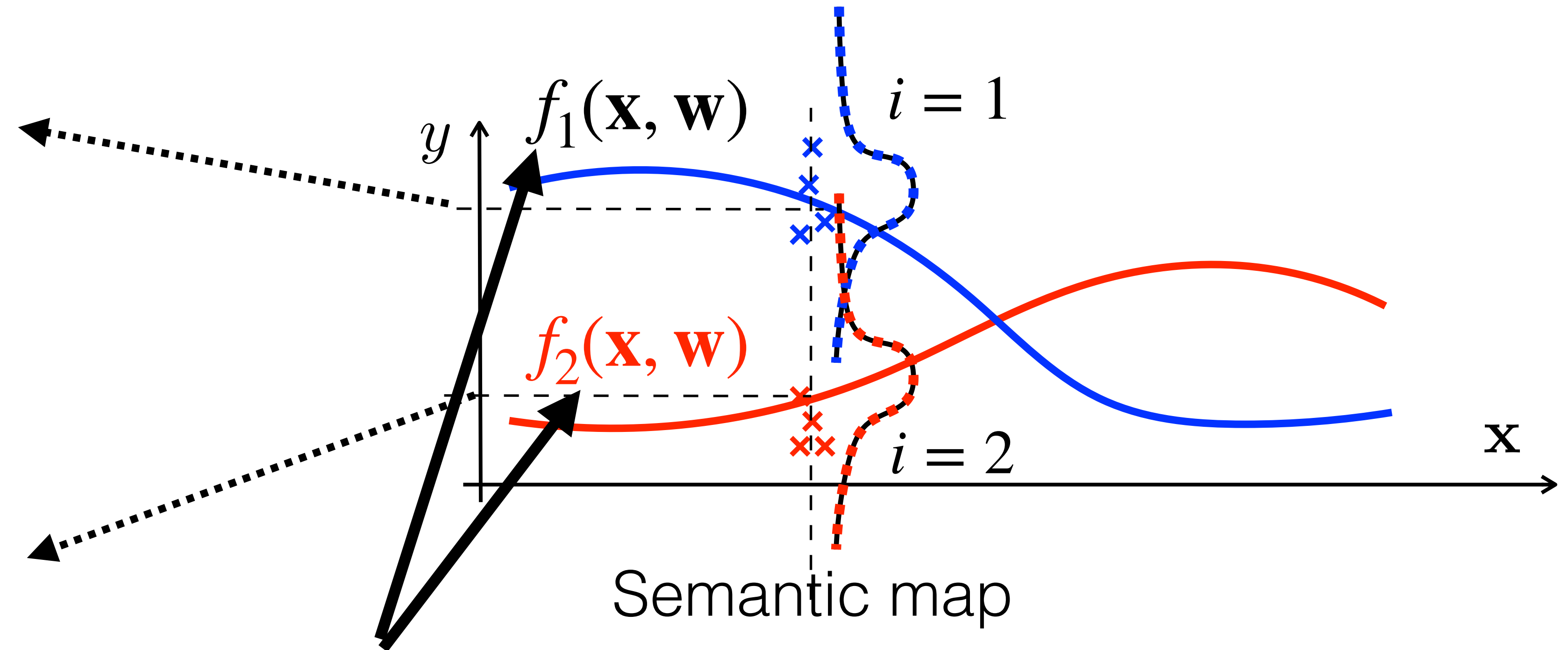


Multiple choice loss

[Microsoft, NIPS, 2012], [Koltun, ICCV, 2017]

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

Work-around 2: allow multiple hypothesis



Problem 1: number of hypothesis may grow exponentially

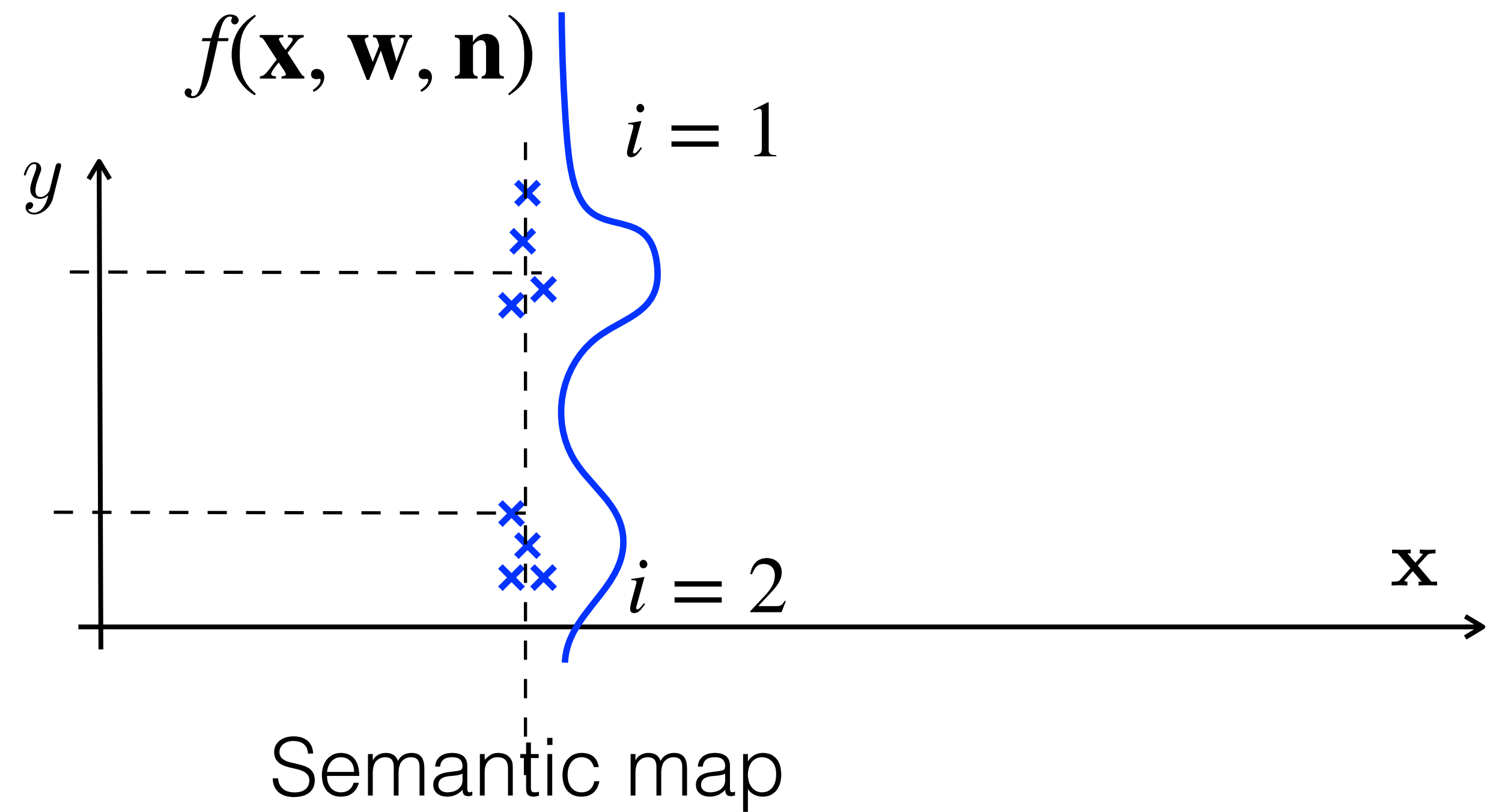
Multiple choice loss

[Microsoft, NIPS, 2012], [Koltun, ICCV, 2017]

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

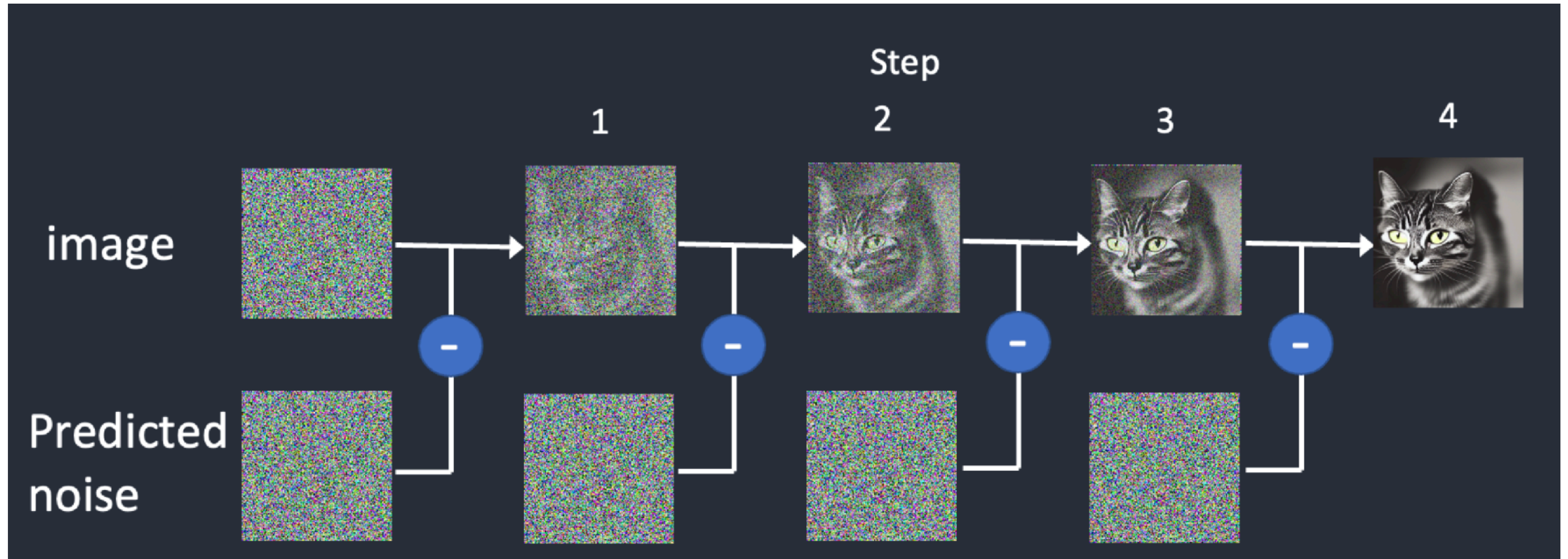
Problem 2: Measuring similarity of images

Work-around 3: use generative model



Generative models: GANs [Goodfellow 2016], VAE,
Diffusion models [Ho, NIPS 2020]

Work-around 3: use generative model

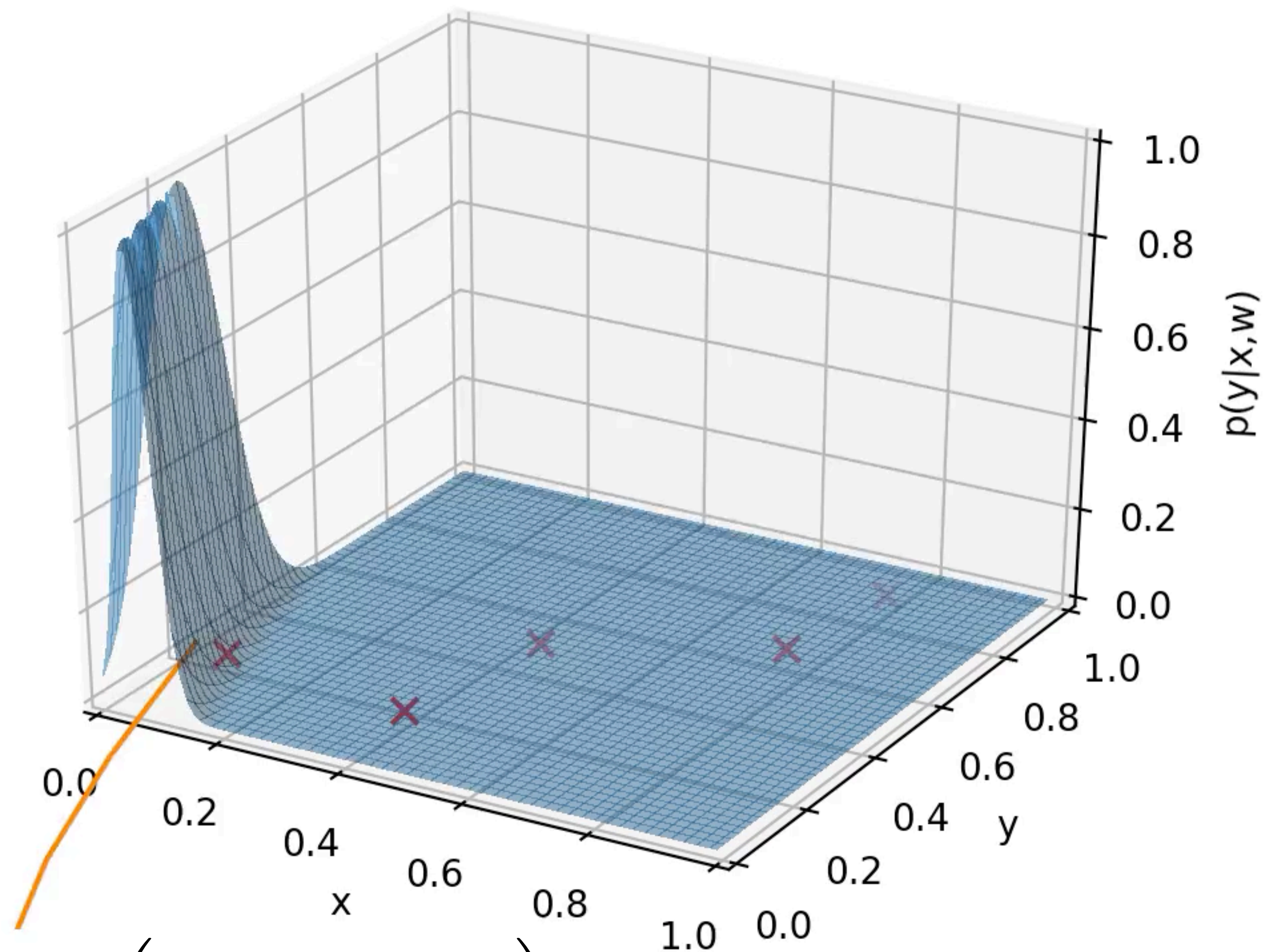


Summary loss

- Maximum Likelihood = Minimum KL-divergence = Minimum “ $-\log(p)$ ”-loss
- Different distributions suffer from optimization issues (zero gradients, sensitivity to good initialization, local optima, curse-of-dimensionality...)

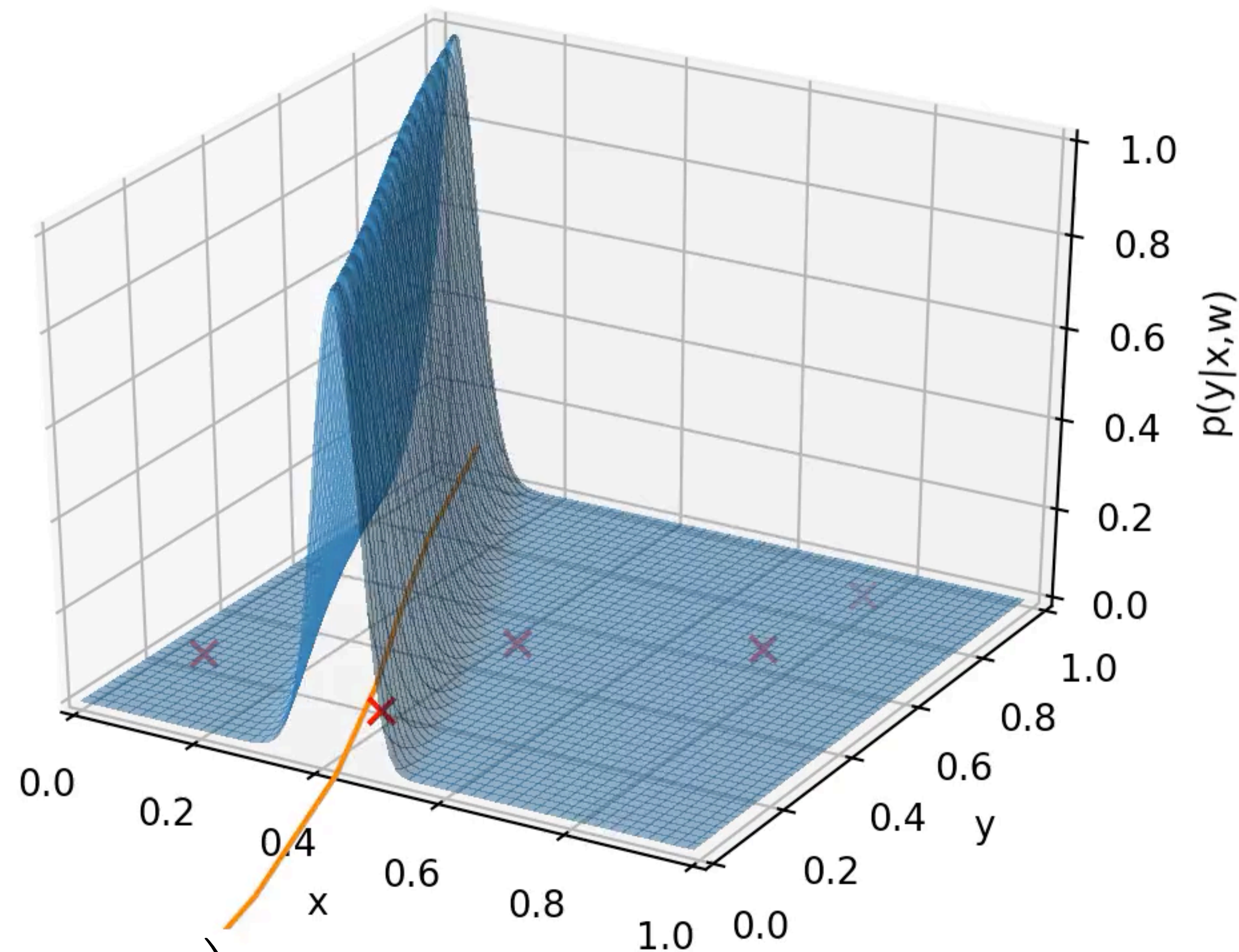
Where does the **overfitting** come from?

Where does the **overfitting** come from?



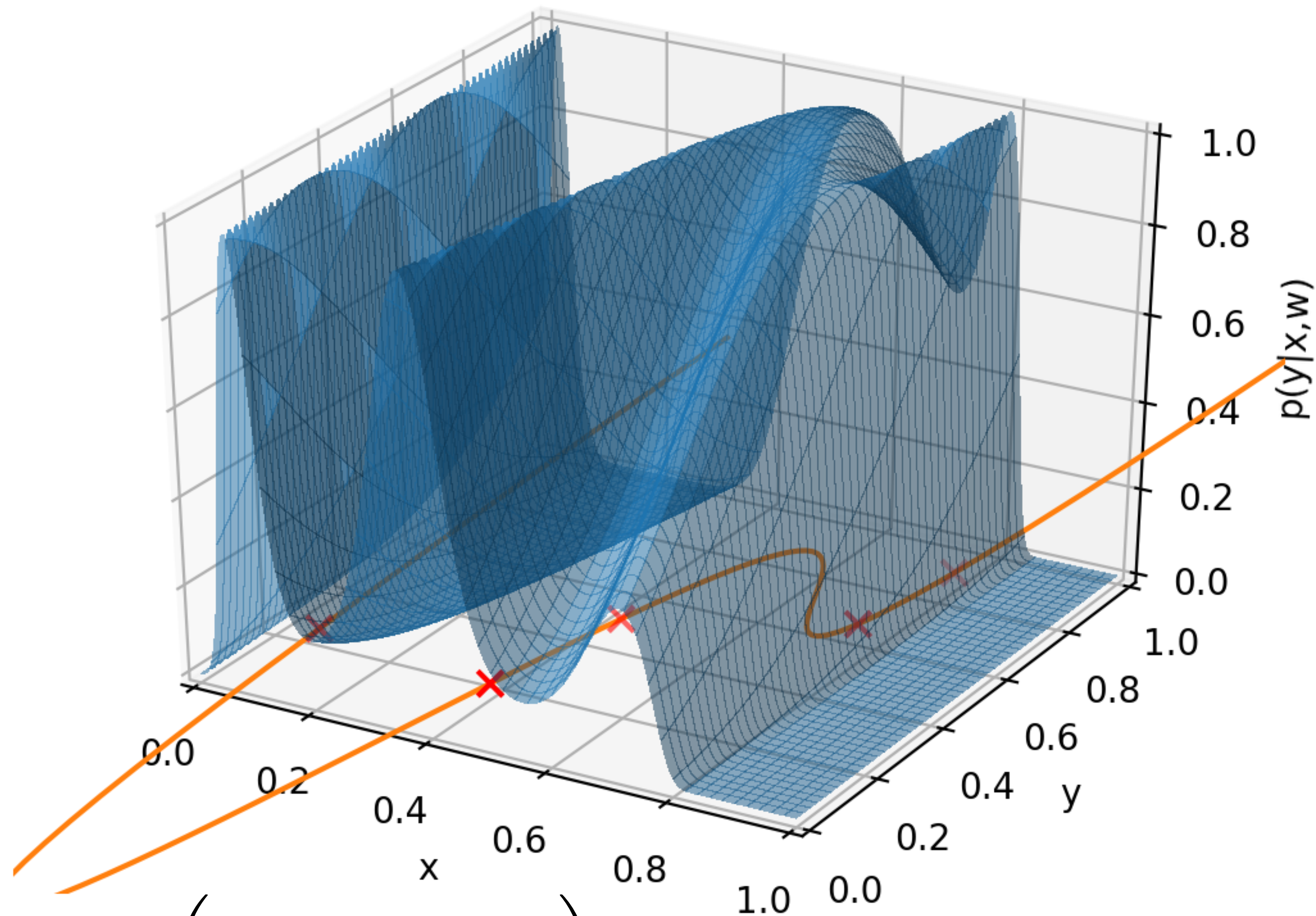
$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \left(\prod_i p(y_i | \mathbf{x}_i, \mathbf{w}) \right) = \arg \min_{\mathbf{w}} \sum_i (w_2 x_i^2 + w_1 x_i + w_0 - y_i)^2$$

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?

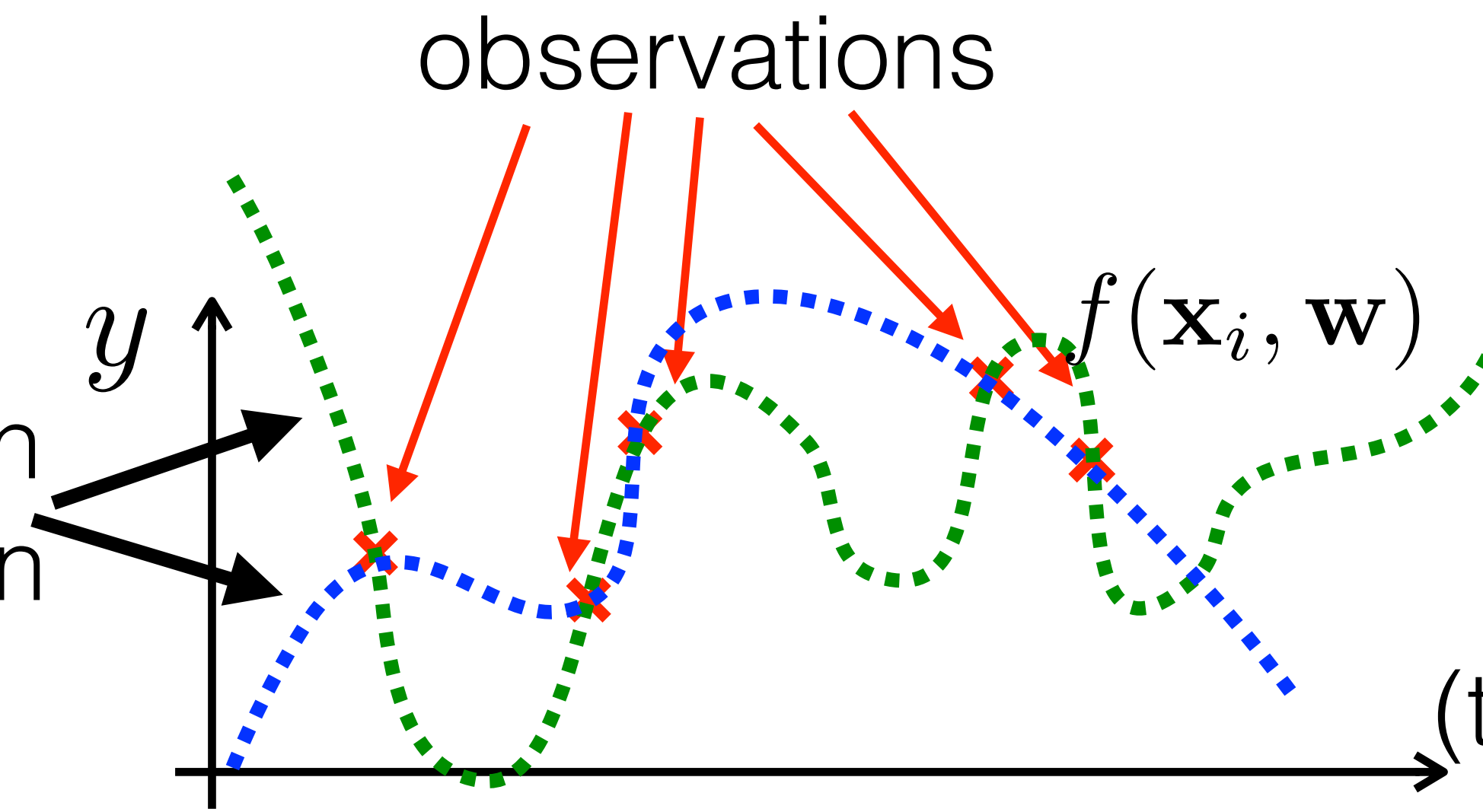
William of Ockham
(1287-1347)



leprechauns can be
involved in any explanation



Many stories consistent with
the broken vase observation



The space of possible
stories is too wild

=> Use the simplest
(the most apriori probable)

Where does the **overfitting** come from?

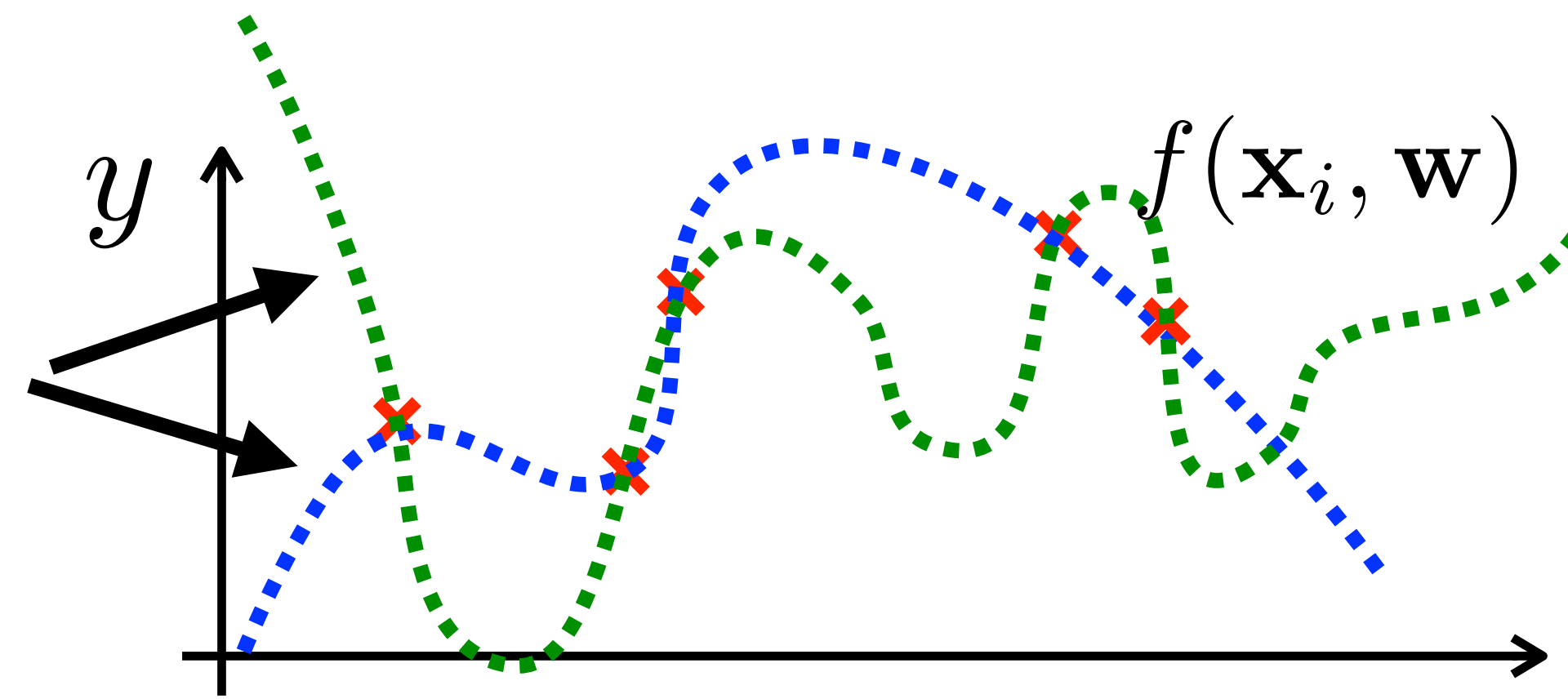
Phaistos disc



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

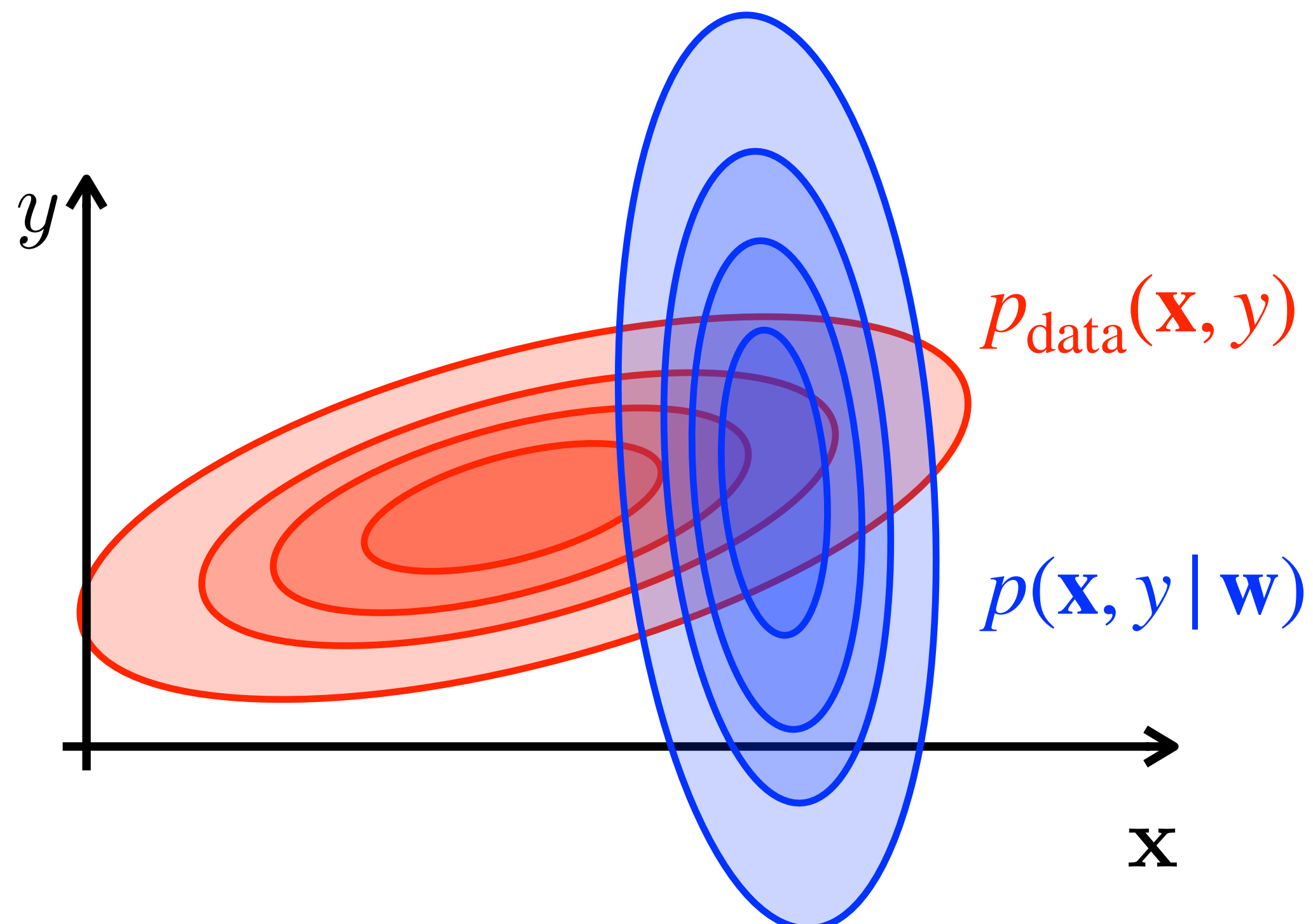


The space of possible
stories is too wild

Where does the **overfitting** come from?

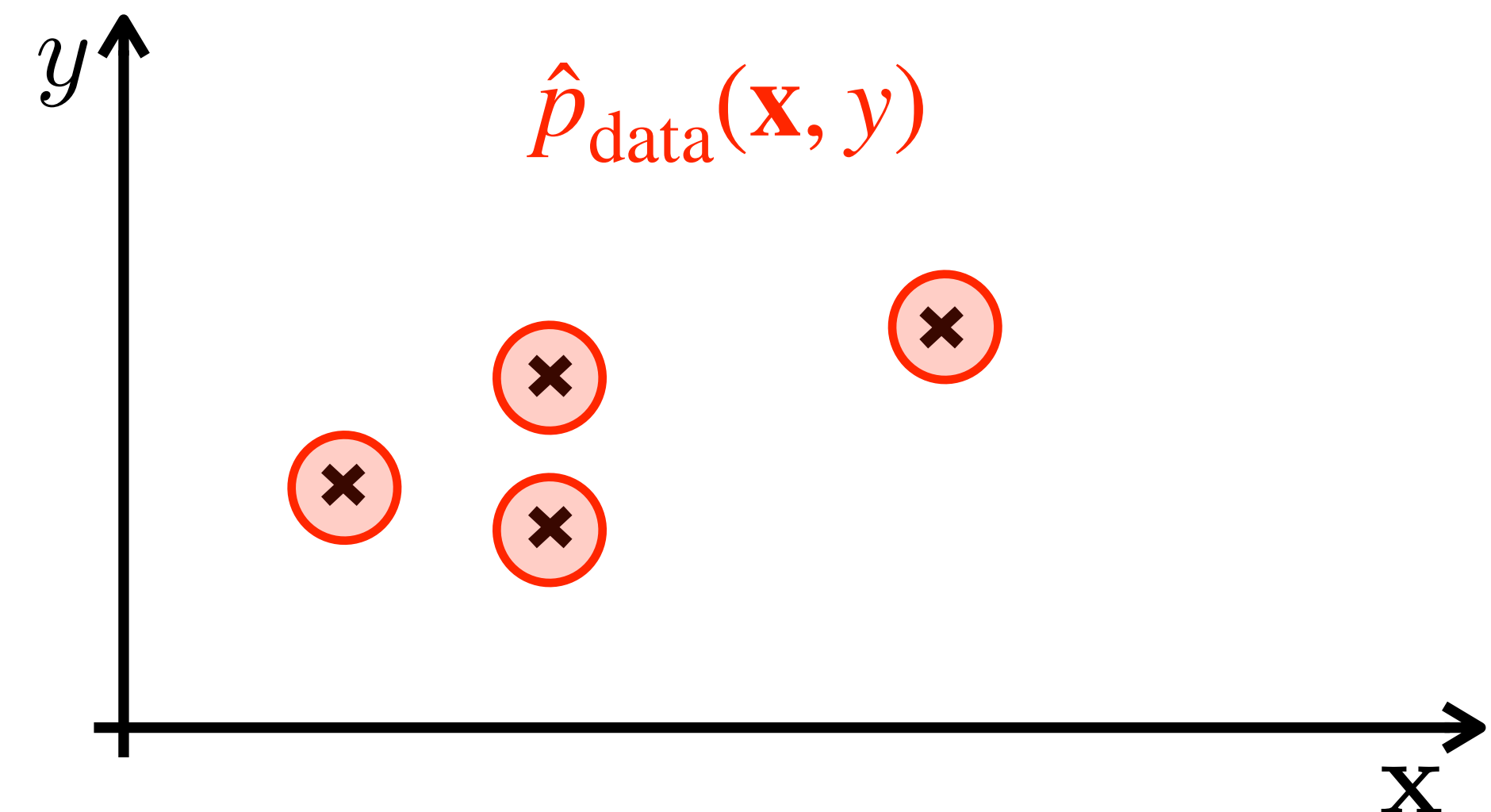
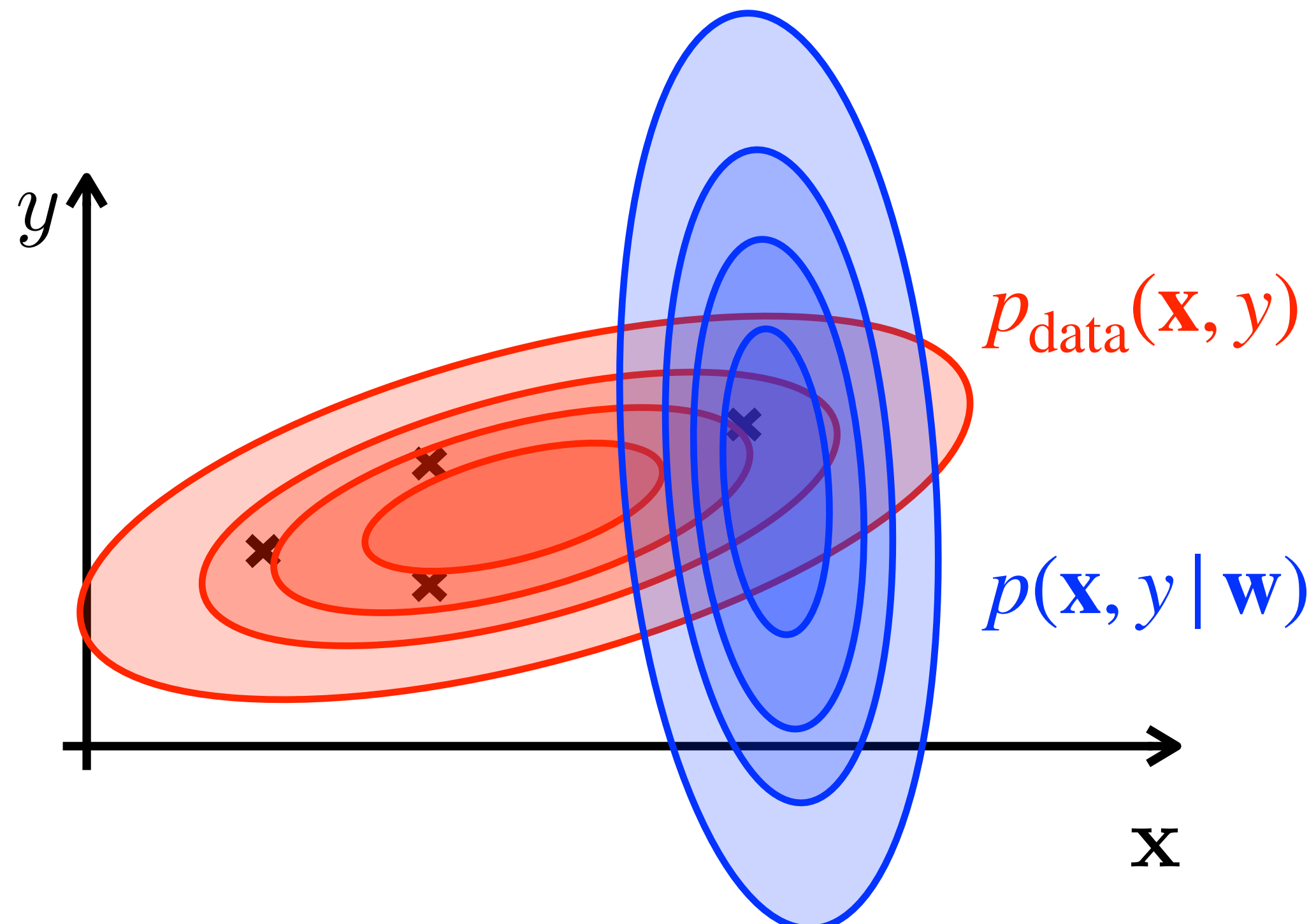
- We fit $p(\mathbf{x}, y | \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

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



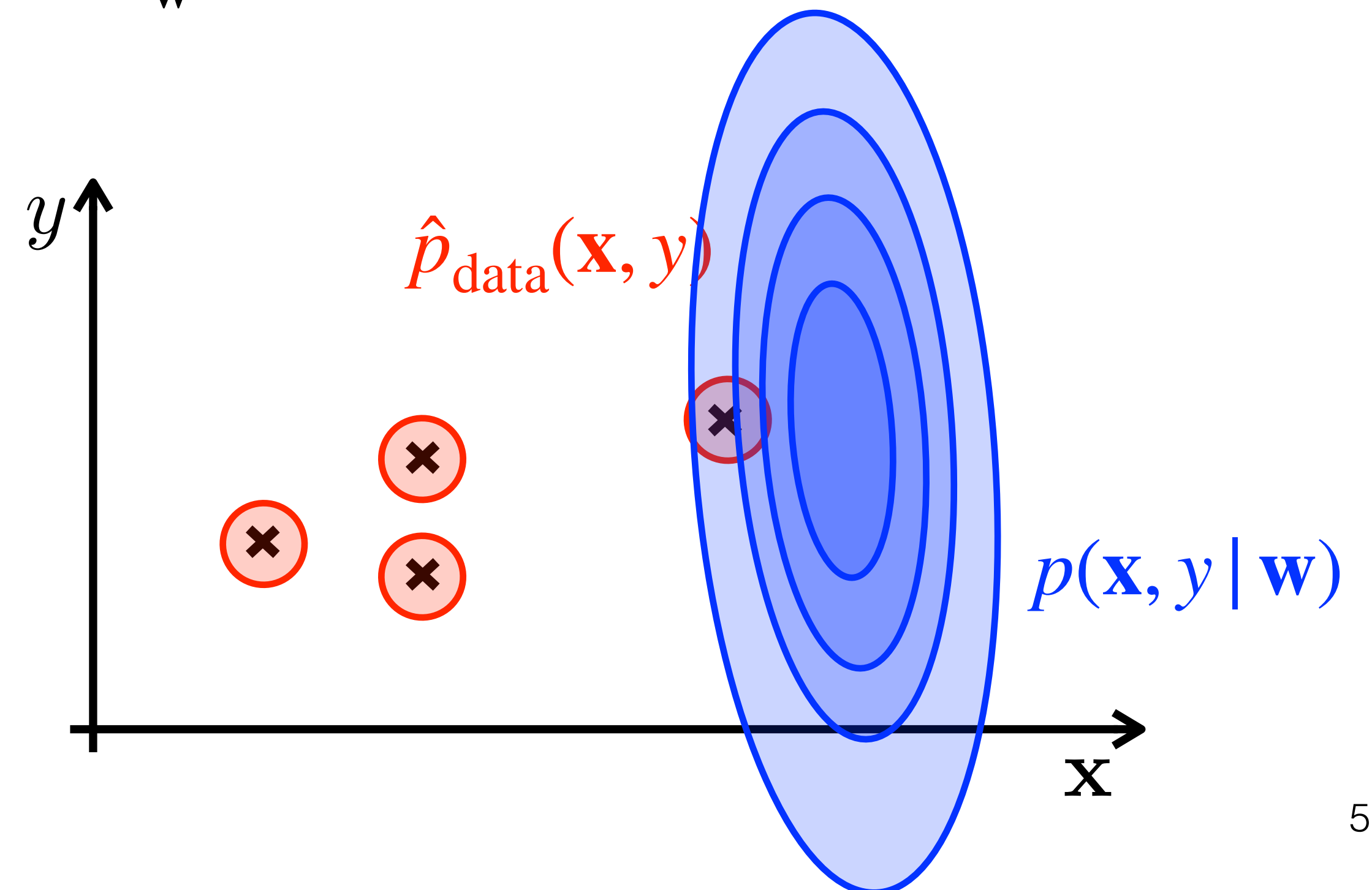
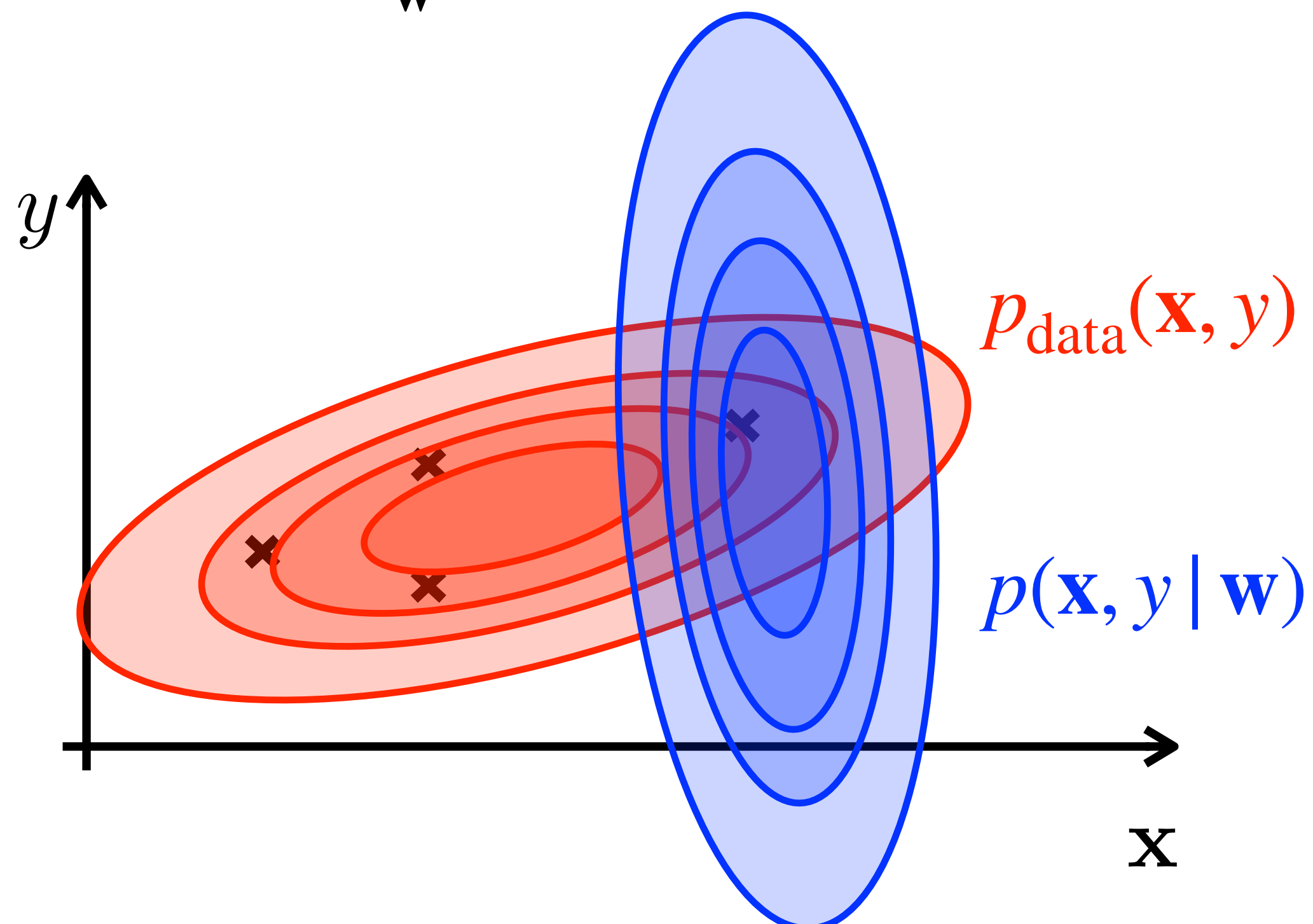
Where does the **overfitting** come from?

- We fit $p(\mathbf{x}, y | \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:
$$\mathbf{w}^{\star} = \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w}))$$
- $p_{\text{data}}(\mathbf{x}, y)$ is unknown $\Rightarrow$ use samples (training set)
- Since the training set is finite, we actually used different $\hat{p}_{\text{data}}(\mathbf{x}, y)$



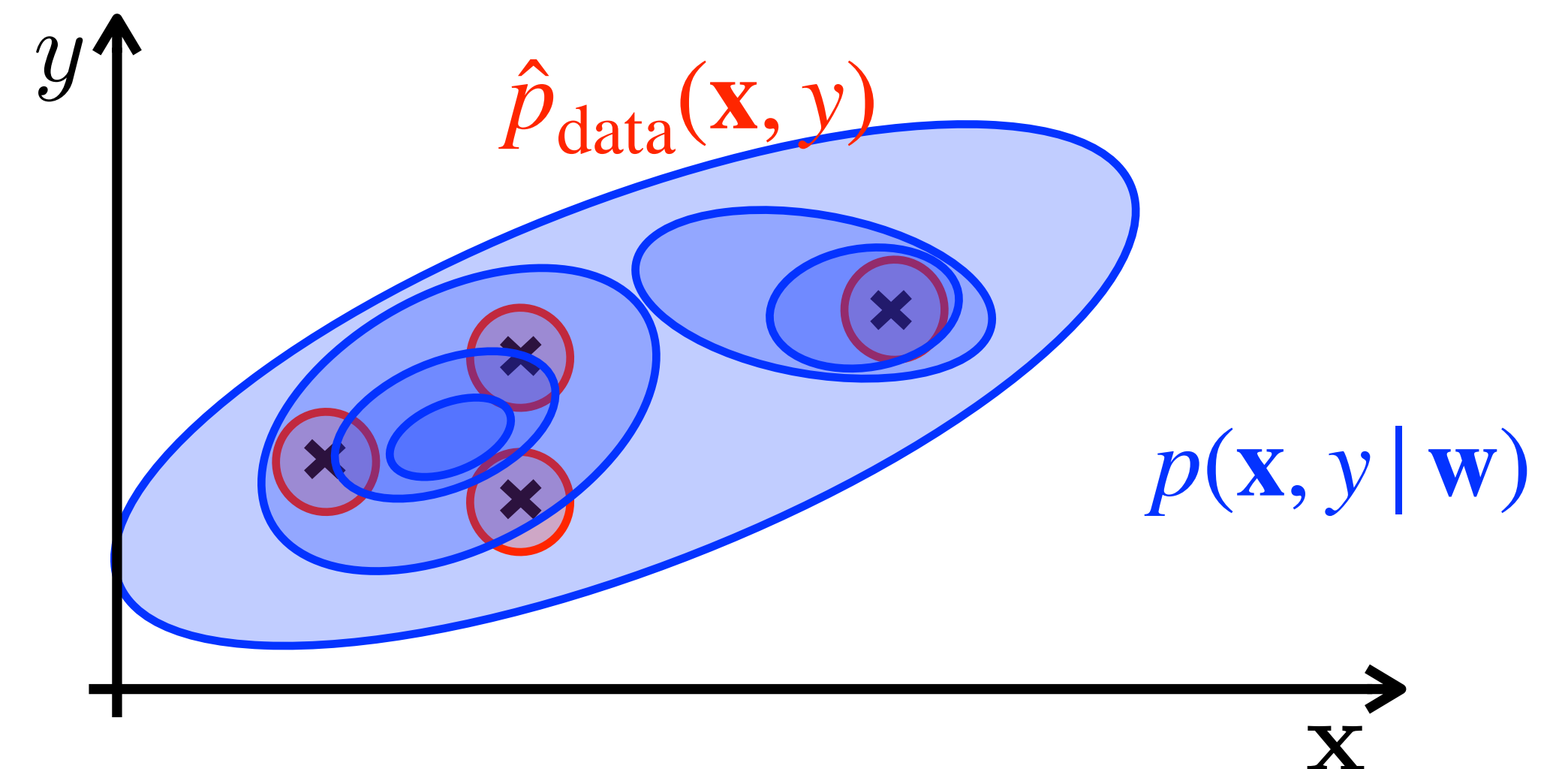
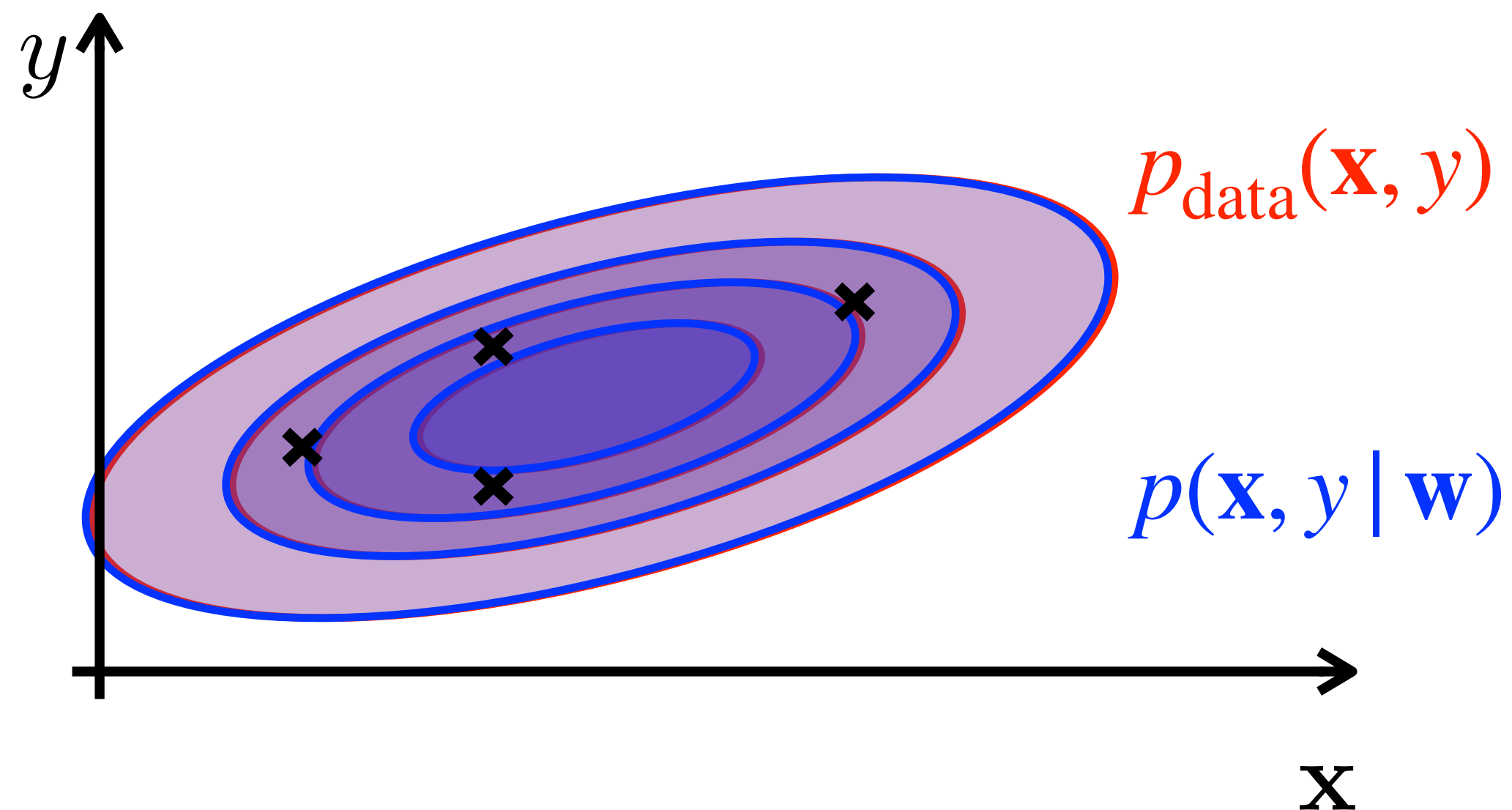
Where does the **overfitting** come from?

- We fit $p(\mathbf{x}, y | \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:
$$\mathbf{w}^{\star} = \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w}))$$
- $p_{\text{data}}(\mathbf{x}, y)$ is unknown $\Rightarrow$ use samples (training set)
- Since the training set is finite, we actually used different $\hat{p}_{\text{data}}(\mathbf{x}, y)$
$$\mathbf{w}^{\star} = \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w})) \neq \arg \min_{\mathbf{w}} D_{KL}(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w}))$$



Where does the **overfitting** come from?

- We fit $p(\mathbf{x}, y | \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:
$$\mathbf{w}^{\star} = \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w}))$$
- $p_{\text{data}}(\mathbf{x}, y)$ is unknown $\Rightarrow$ use samples (training set)
- Since the training set is finite, we actually used different $\hat{p}_{\text{data}}(\mathbf{x}, y)$
$$\mathbf{w}^{\star} = \arg \min_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w})) \neq \arg \min_{\mathbf{w}} D_{KL}(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w}))$$



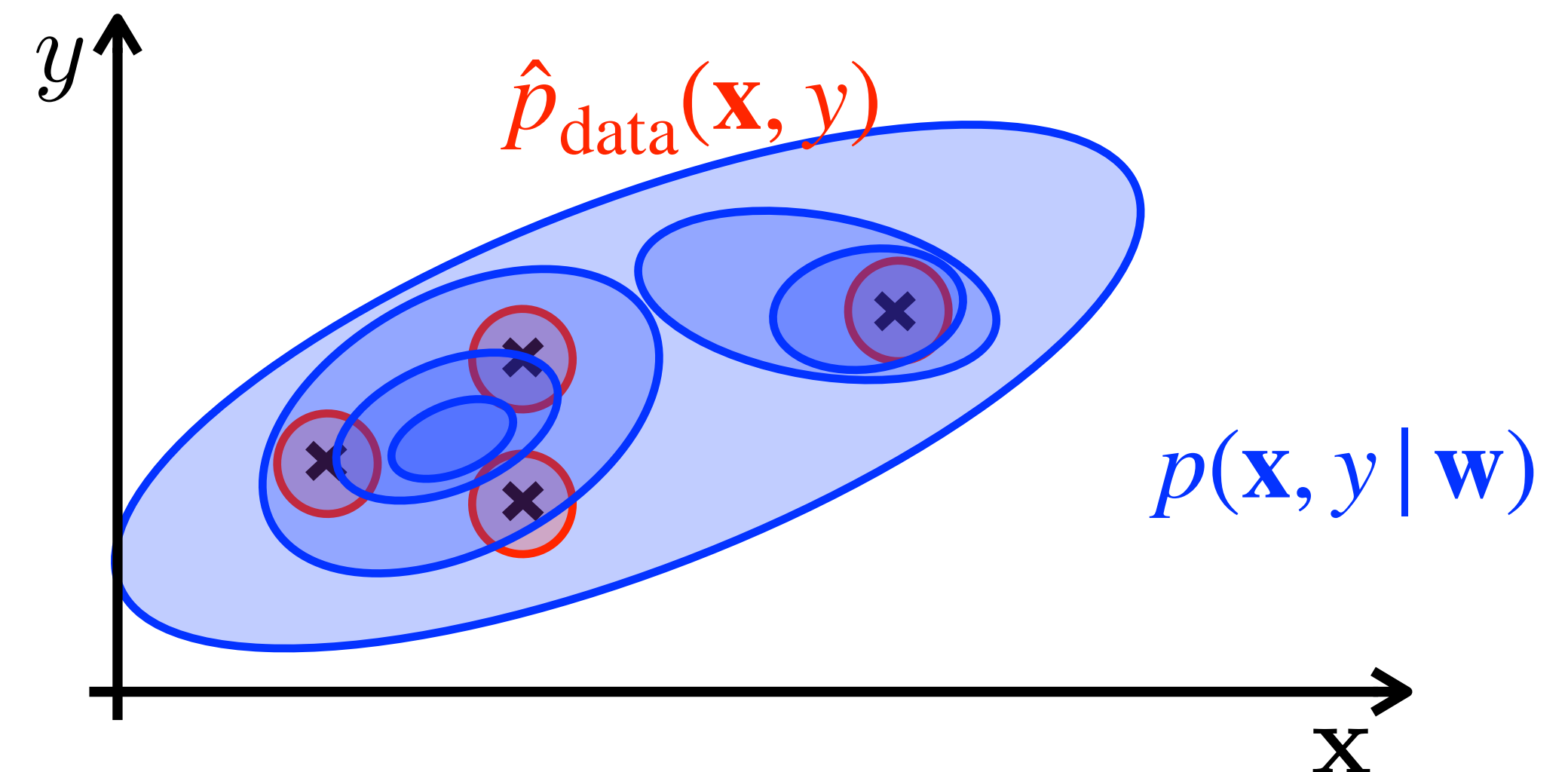
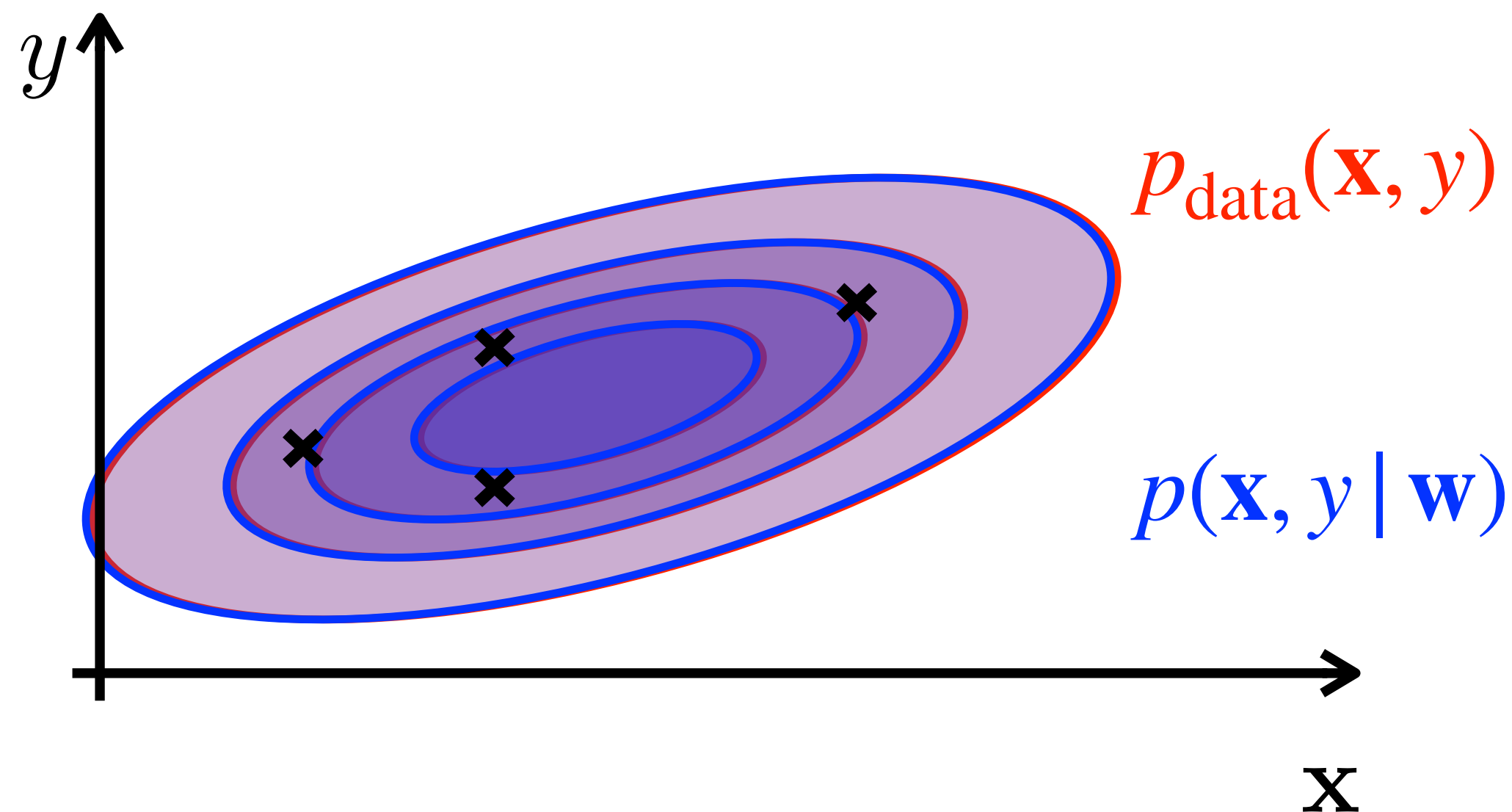
Where does the **overfitting** come from?

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

Take home message: **Optimization $\neq$ Machine learning**

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

Therefore approximated criterion is optimized instead



Where does the **overfitting** come from?

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

Take home message: **Optimization** $\neq$ **Machine learning**

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

Therefore approximated criterion is optimized instead

How to suppress overfitting?

How to suppress overfitting?

- 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)

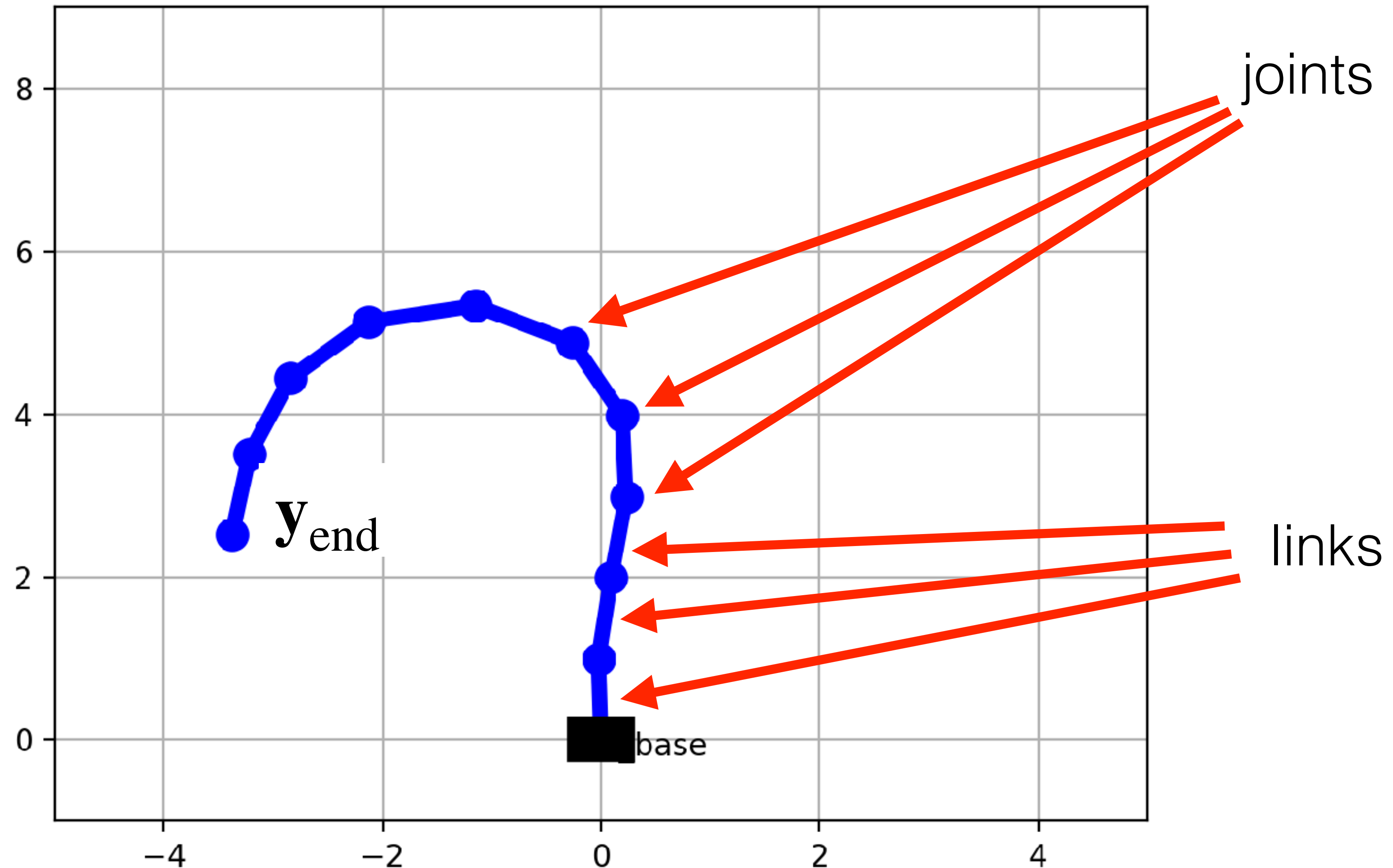
If you cannot do it, at least penalize wild solutions



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

2D manipulator

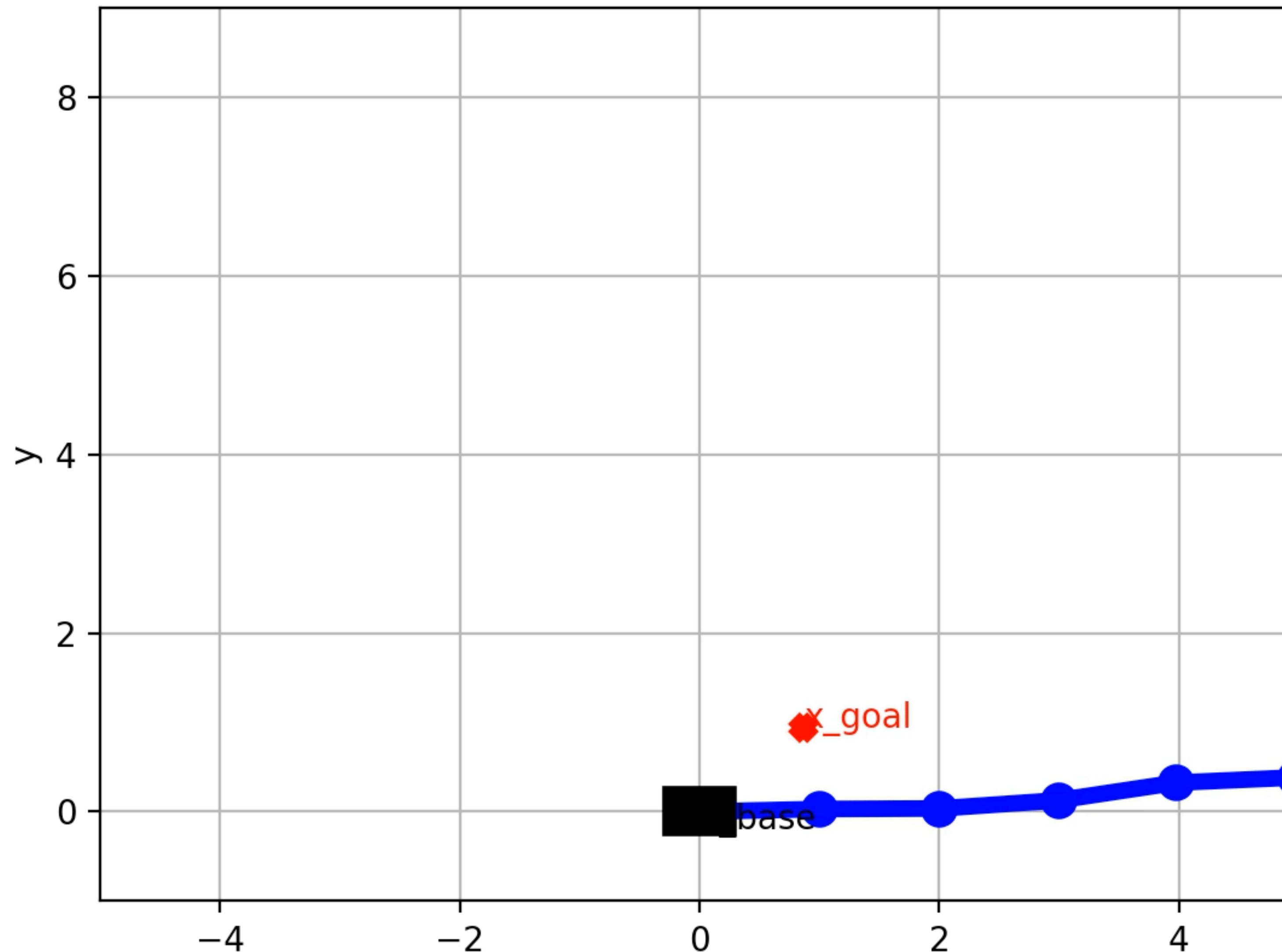
HW1 IKT: (links + end_effector) => joints



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

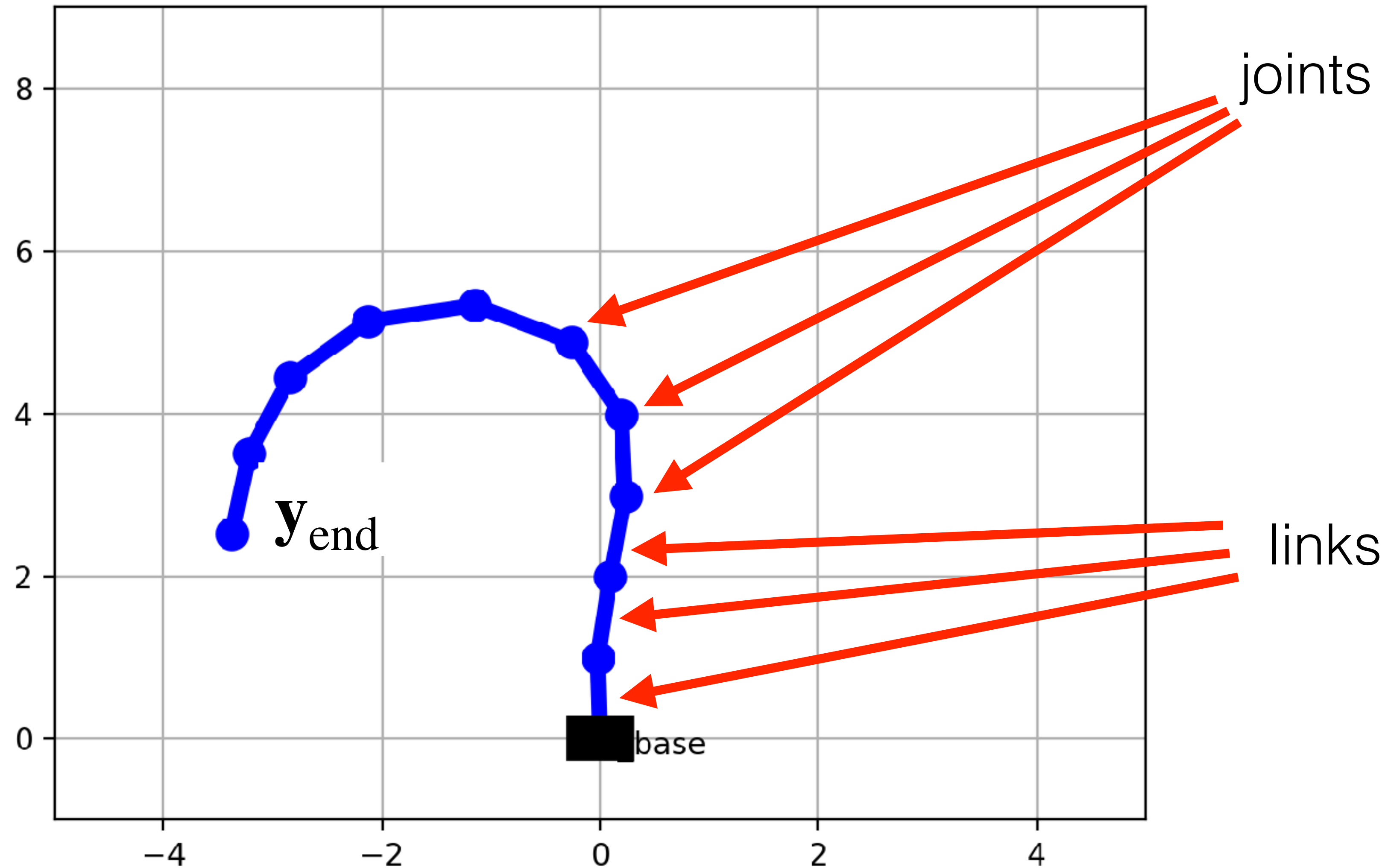
2D manipulator

HW1 IKT: (links + end_effector) => joints



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

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

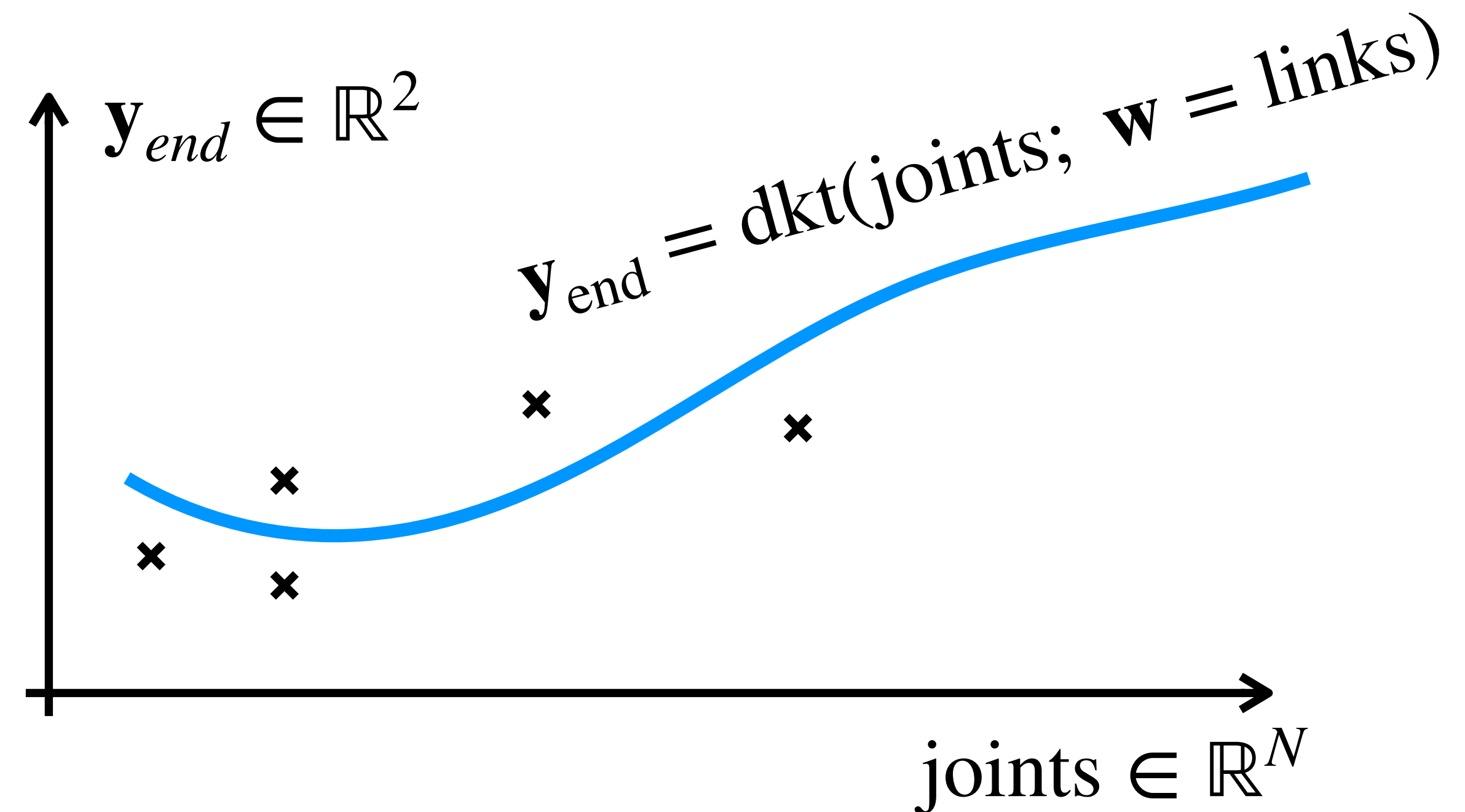
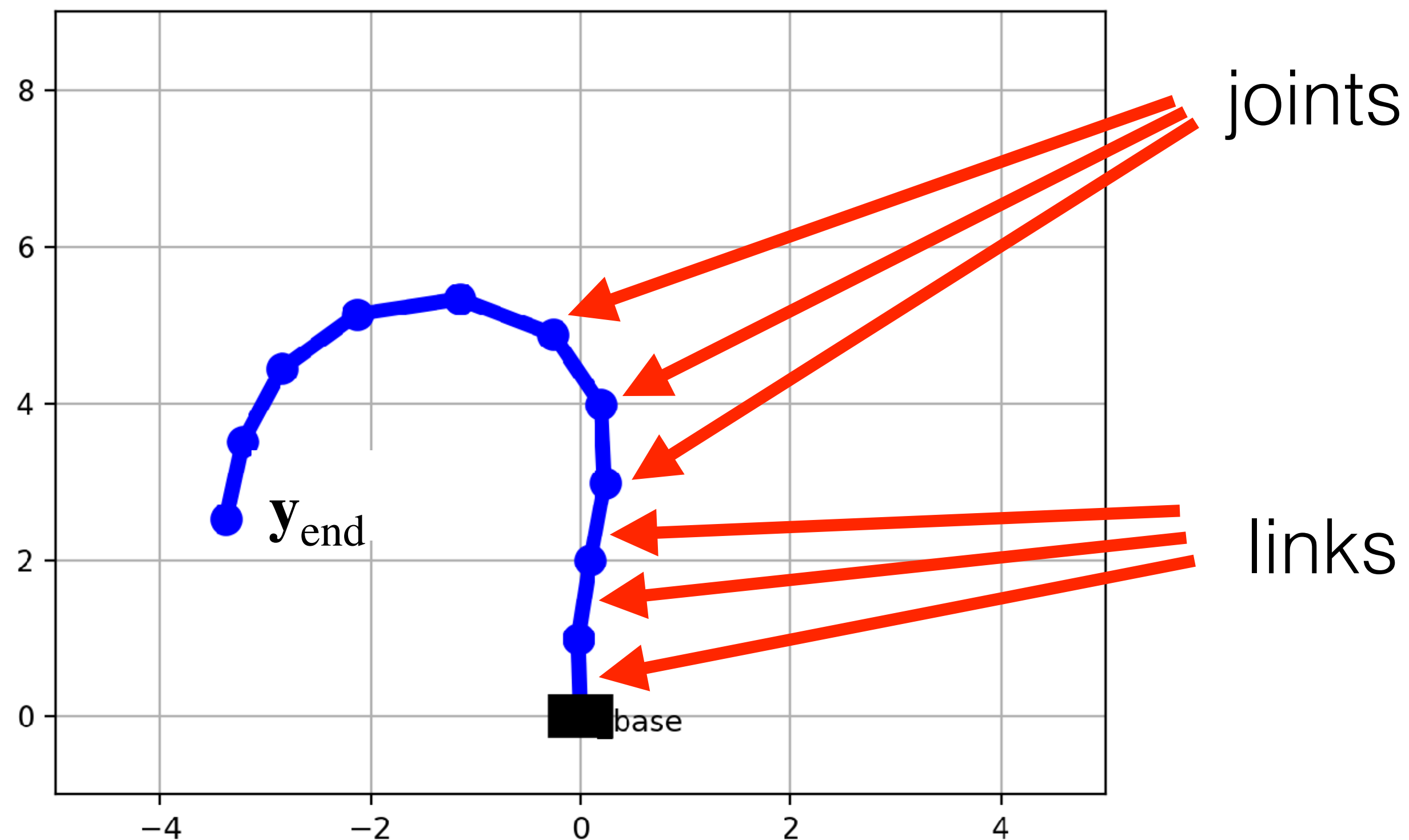


1) Use the right $p(\mathbf{x}, y | \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

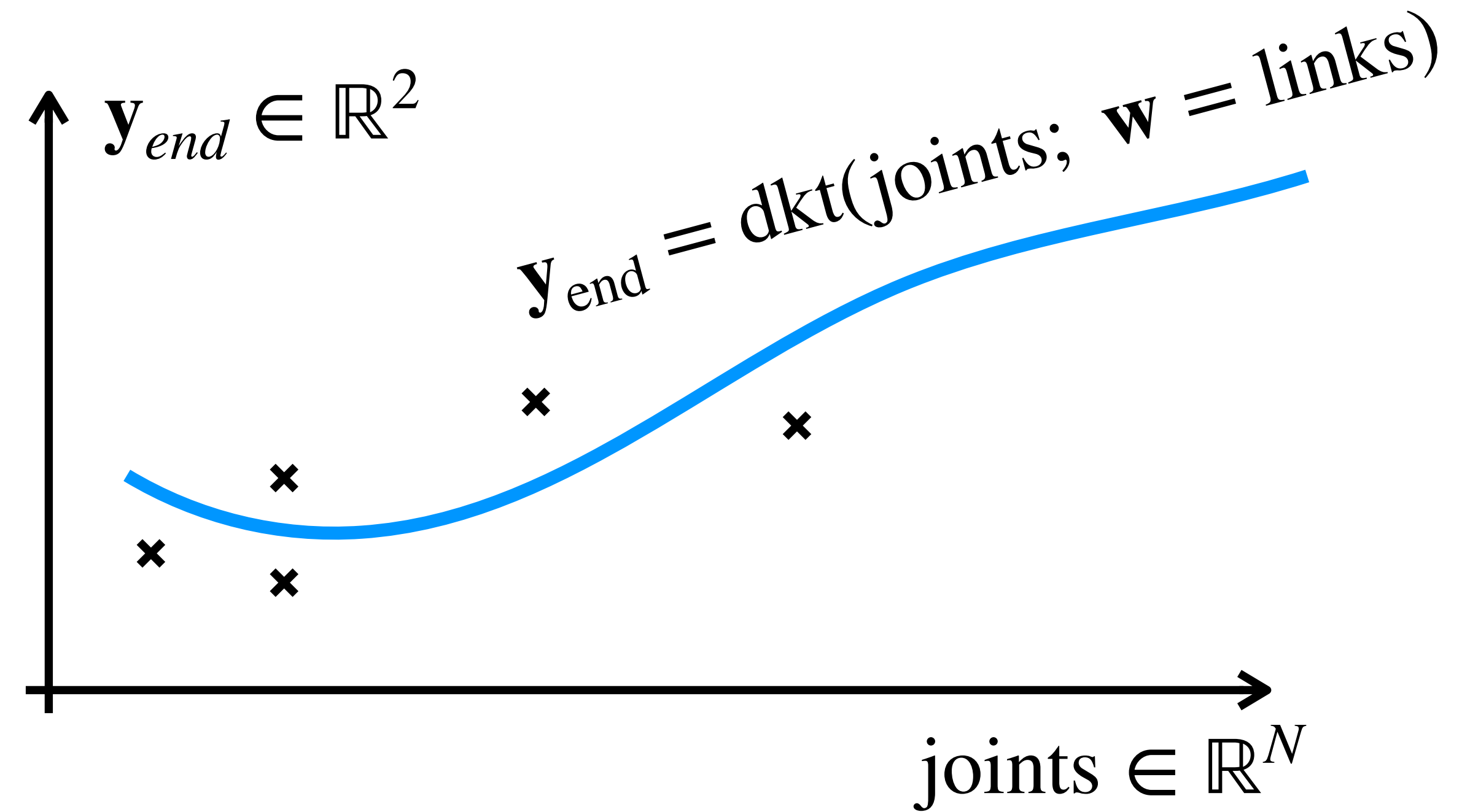
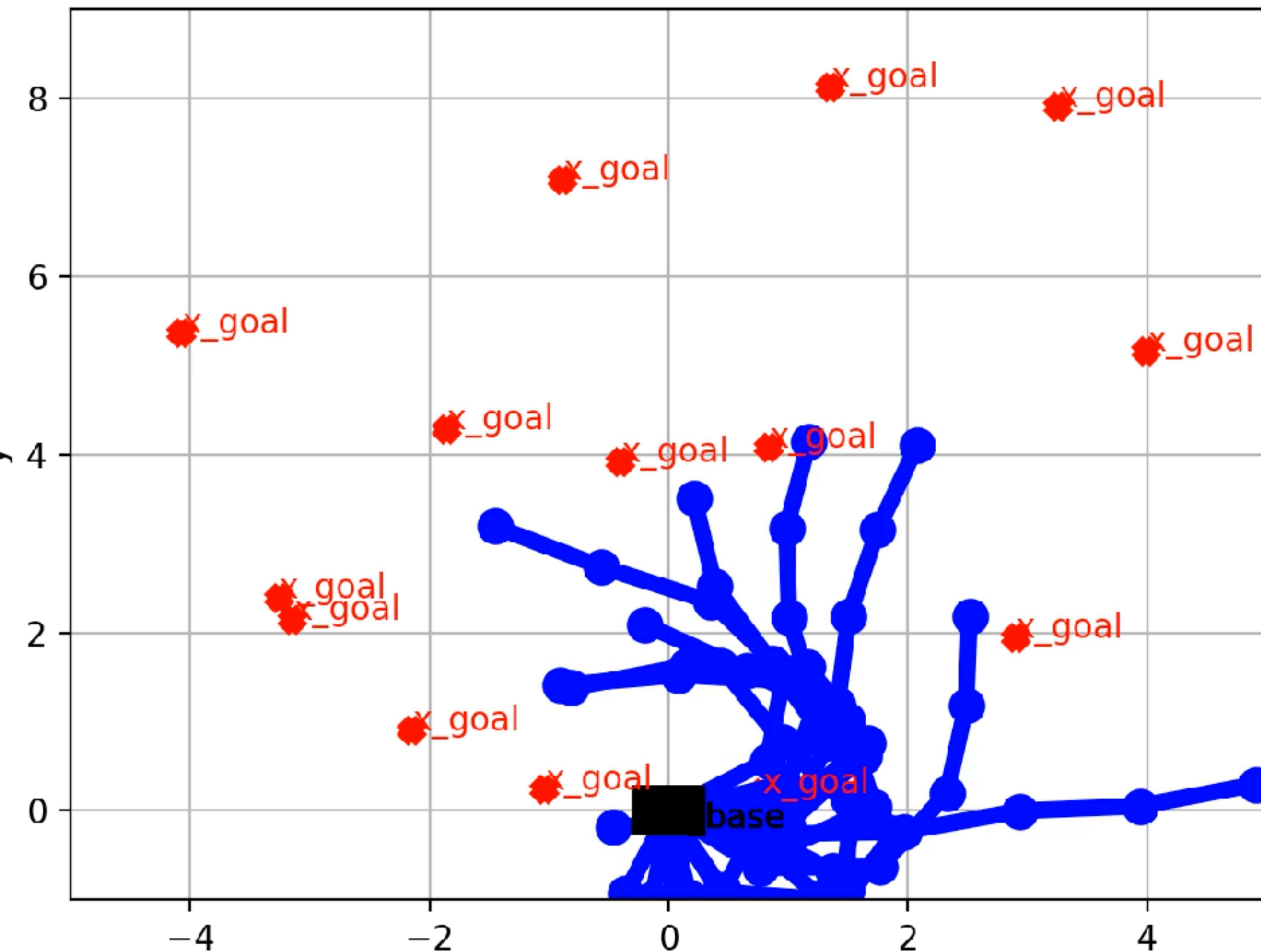
model: $p(y | \mathbf{x}, \mathbf{w}) = \mathcal{N}(y_{\text{end}}; f(\mathbf{x}, \mathbf{w}), \sigma)$

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

- (a) linear function $\mathbf{y}_{\text{end}} = f(\text{joints}; \mathbf{W}) = \mathbf{W} \cdot \text{joints}$ (underfit)
- (b) deep ConvNet $\mathbf{y}_{\text{end}} = f(\text{joints}; \mathbf{w}) = \text{"deep conv net"}$ (overfit)
- (c) DKT: $\mathbf{y}_{\text{end}} = \text{dkt}(\text{joints}; \mathbf{w} = \text{links})$ (well-justified model)



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



How to suppress overfitting?

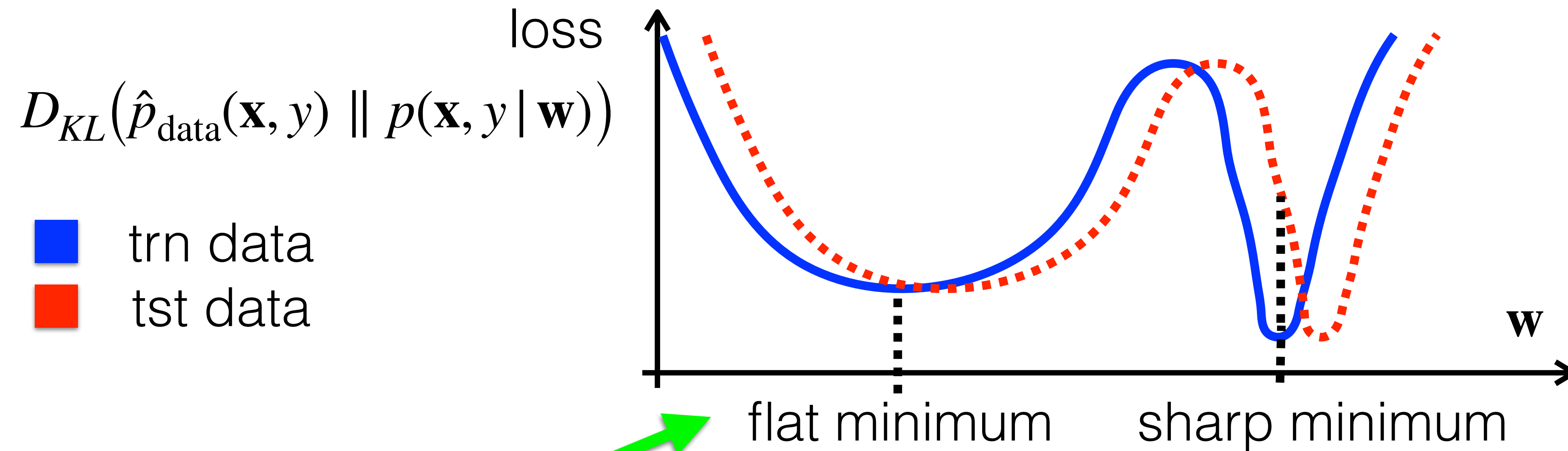
1) Use the right $p(\mathbf{x}, y | \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

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



How to suppress overfitting?

Which one is better???



Good generalization

Weak generalization => optimum prone to overfitting

Testing error remains small

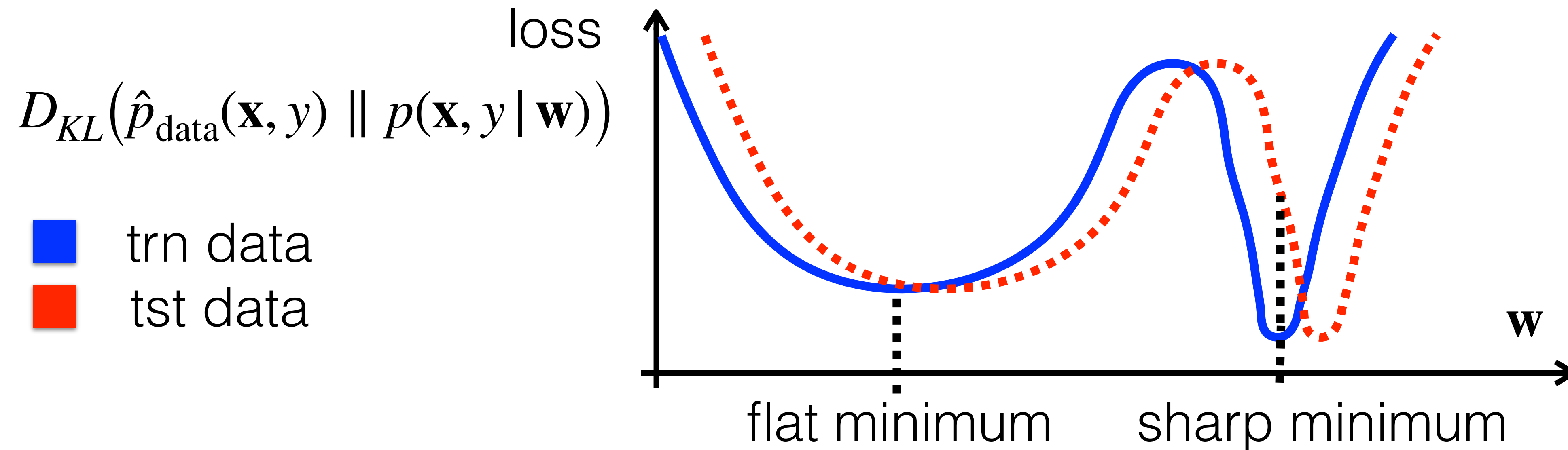
Testing error grows fast with a small trn/tst shift

Weaker learning methods are surprisingly better in generalization

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

How to suppress overfitting?

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



One can enforce flat minimum:

By optimization technique: e.g. momentum

By loss function: $\min_w \max_{\|\epsilon\|_2 \leq \rho} L_{\text{train}}(w + \epsilon)$

[Foret 2021]

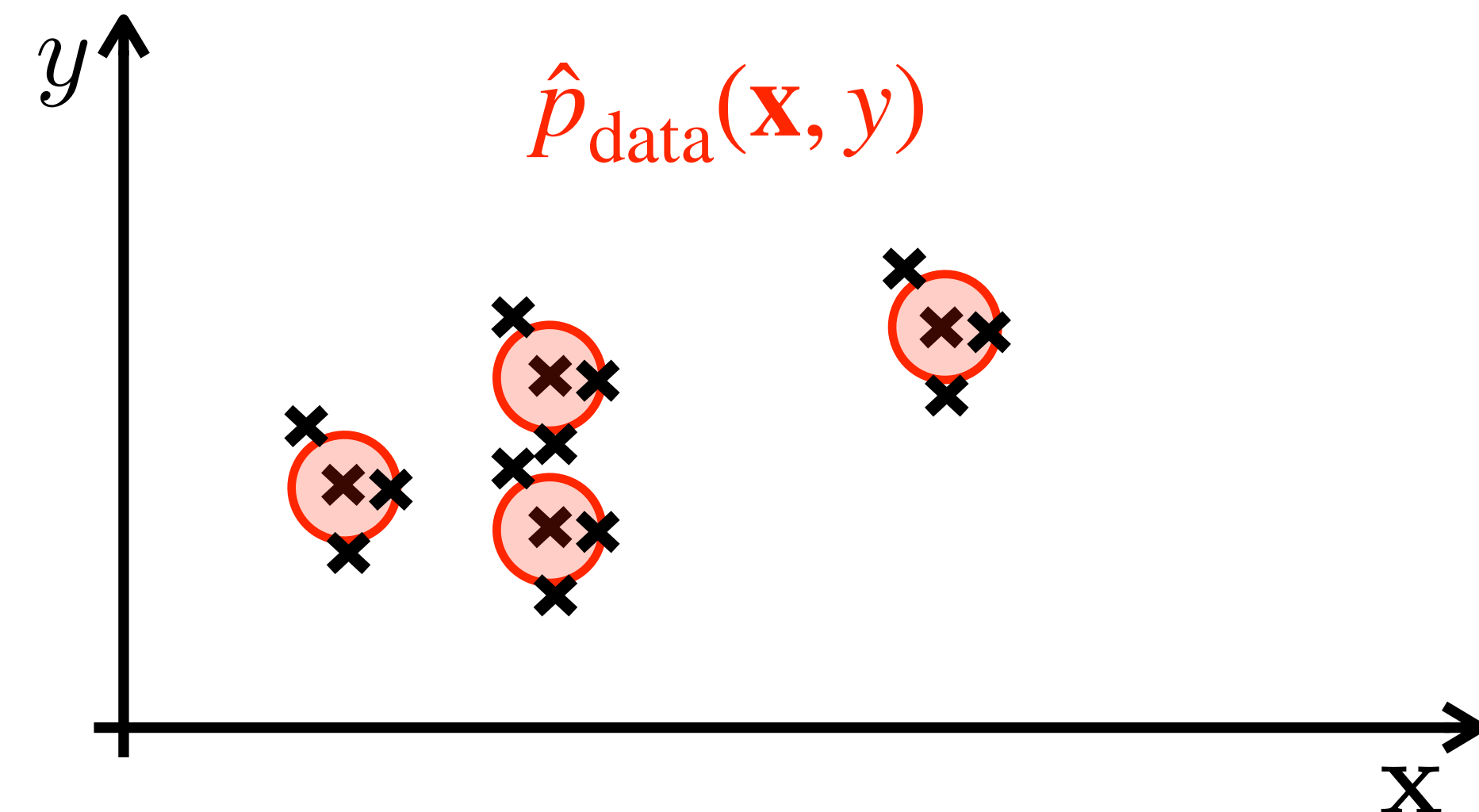
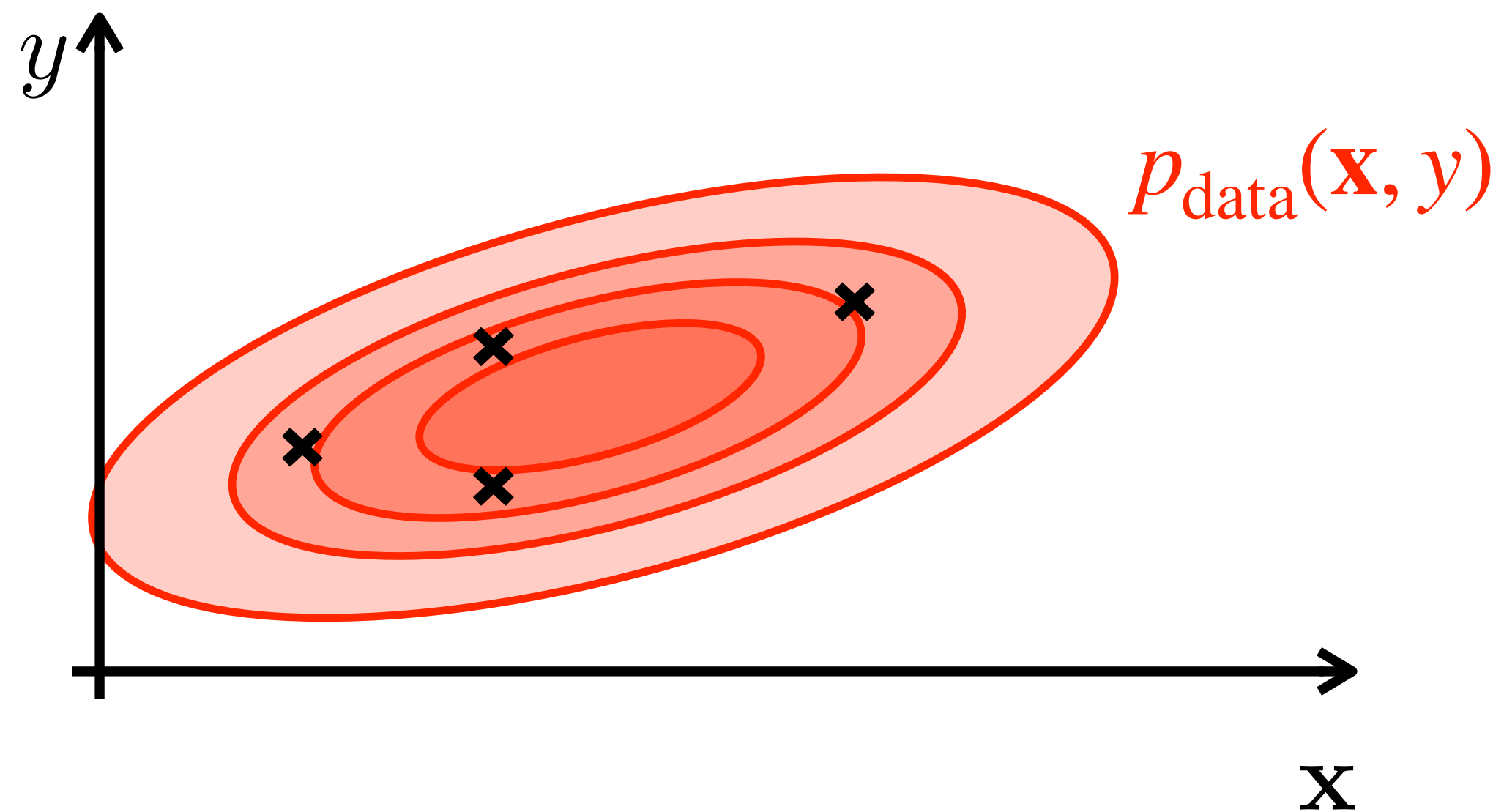
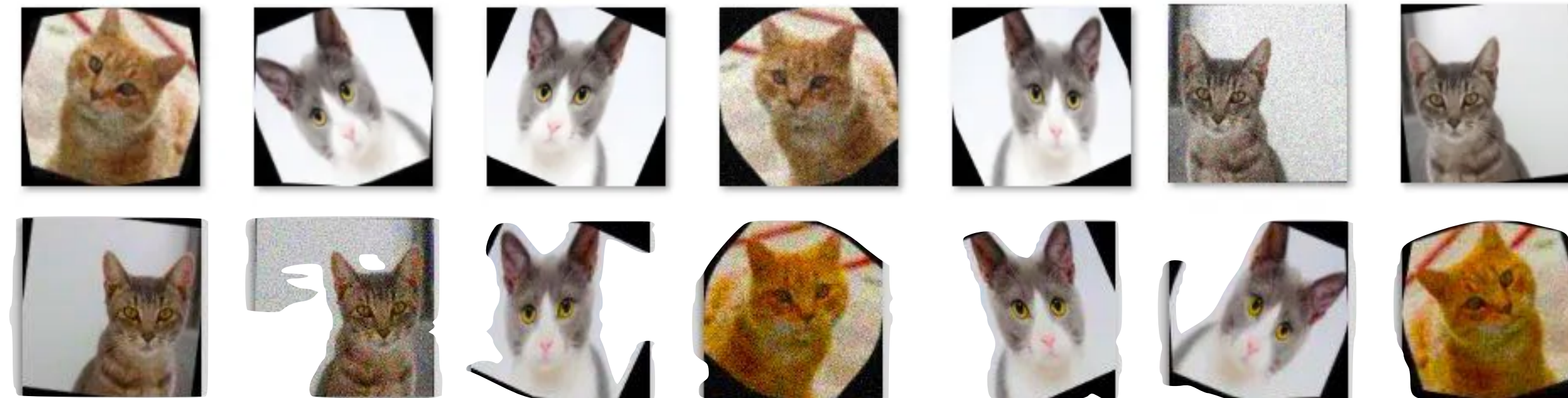
<https://arxiv.org/pdf/2106.01548.pdf>

How to suppress overfitting?

Use close-to-infinite dataset



Data augmentation

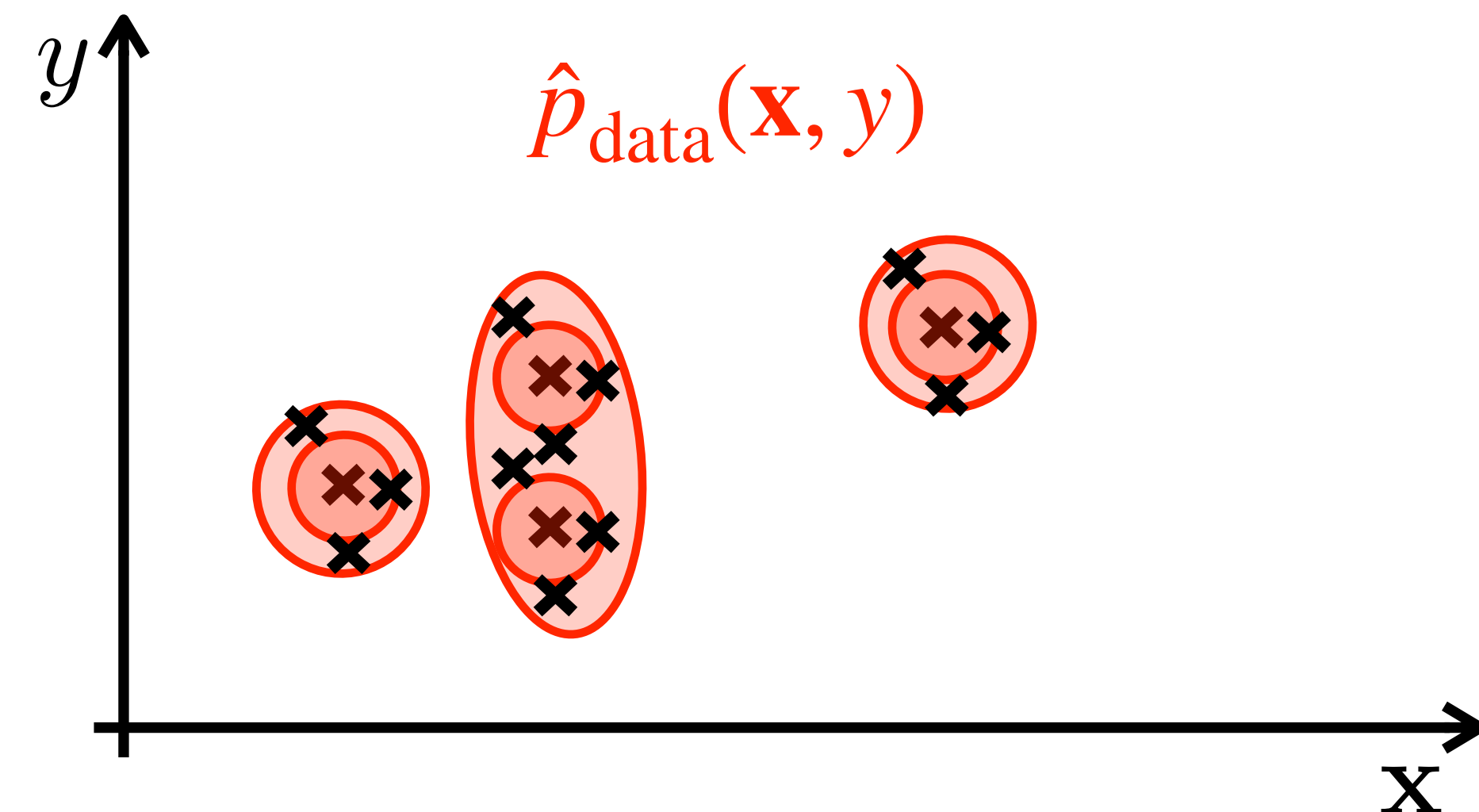
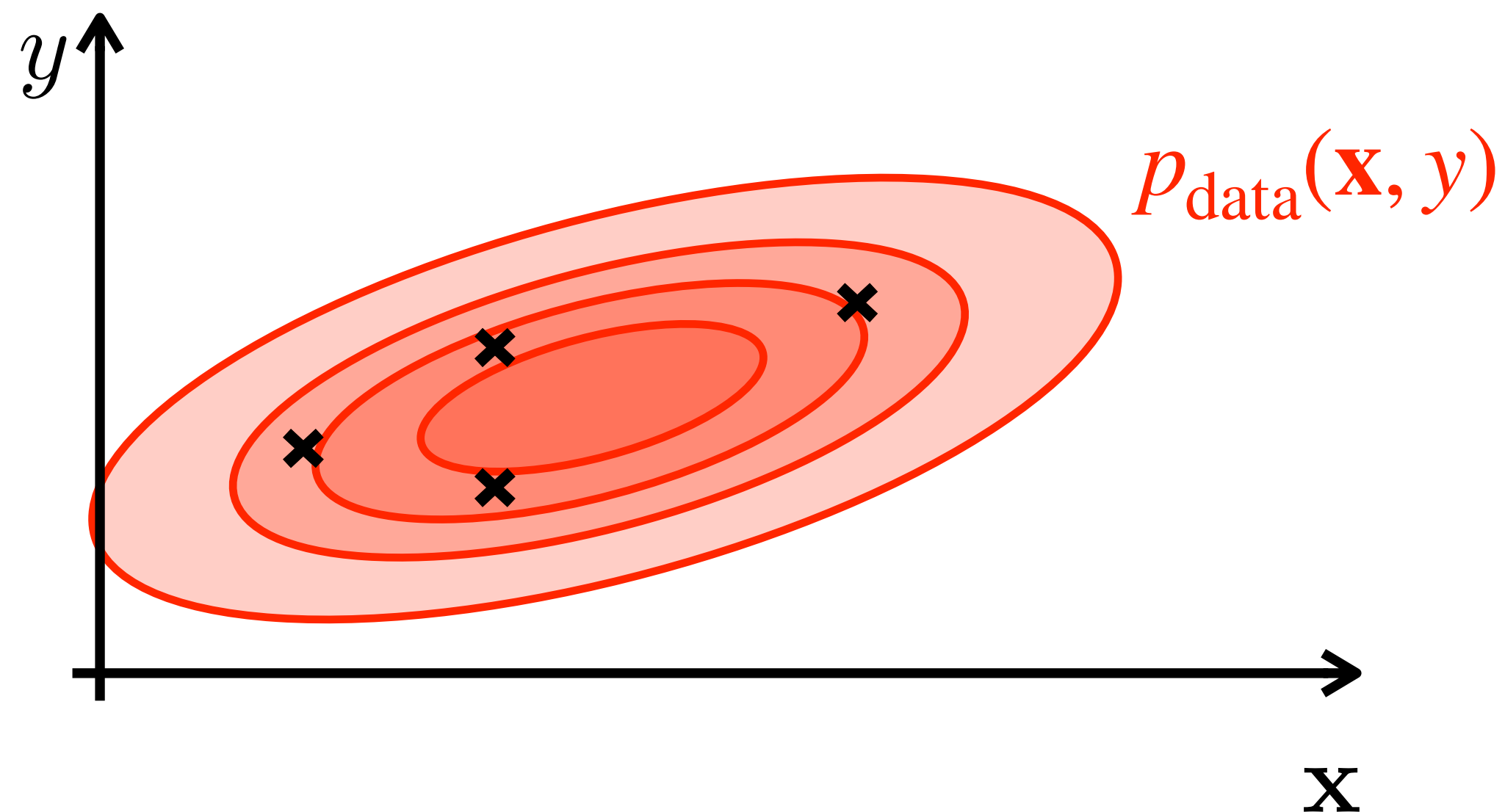
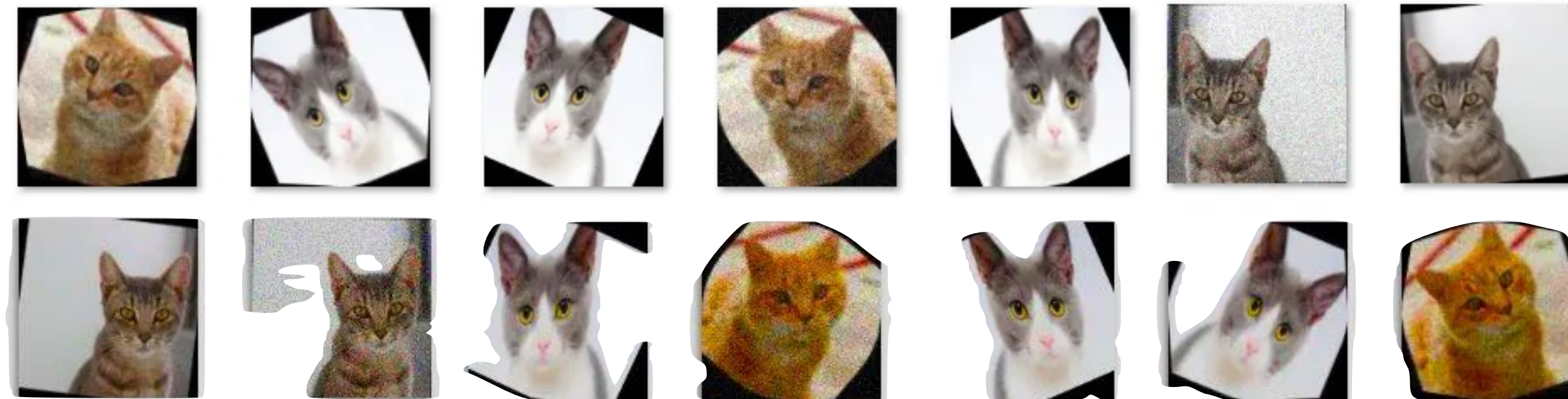


How to suppress overfitting?

Use close-to-infinite dataset

Reasonable geometrical and histogram transformations:
Mirroring, scaling, rotation, squeezing, contrast, brightness, ...

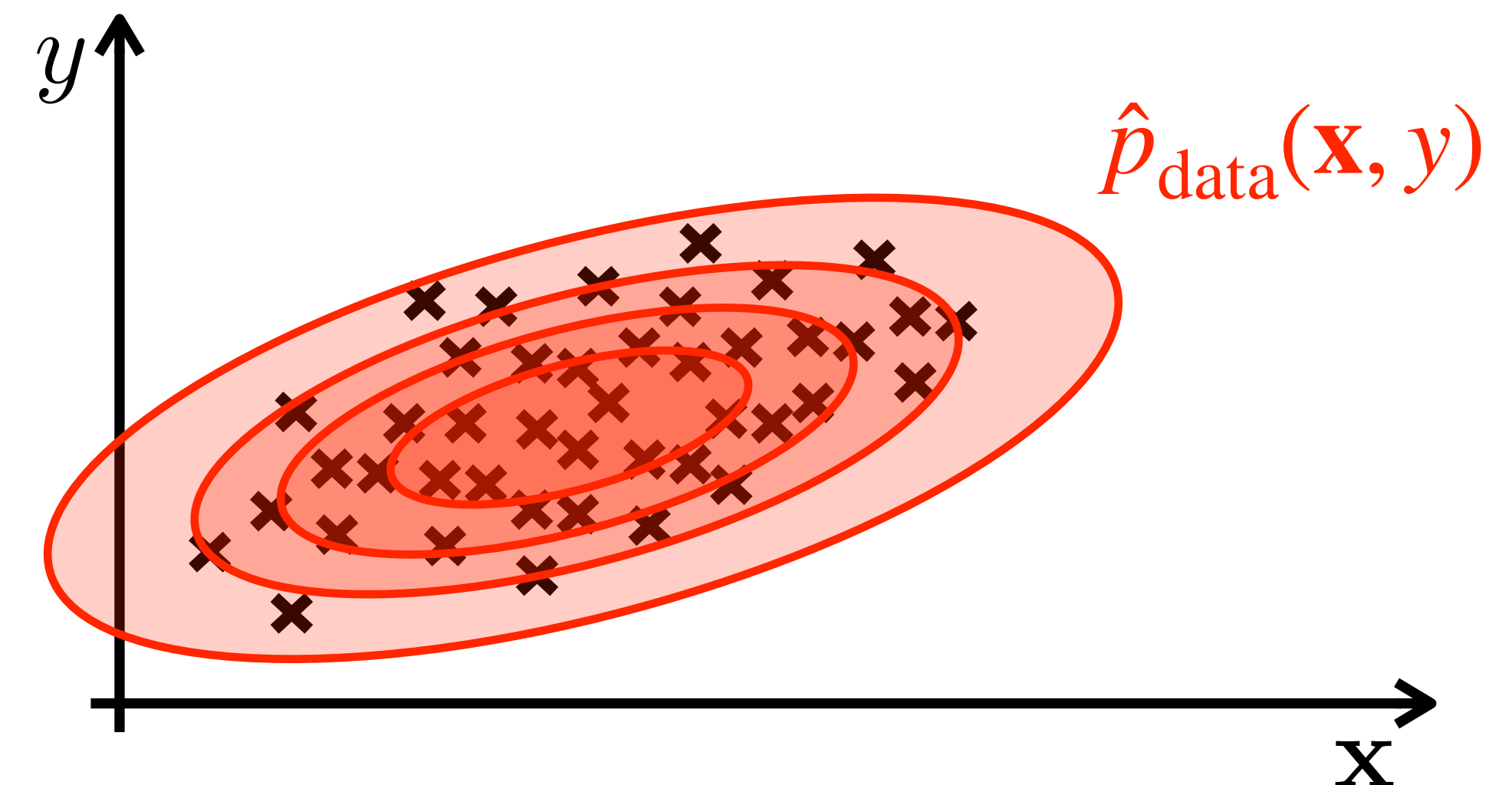
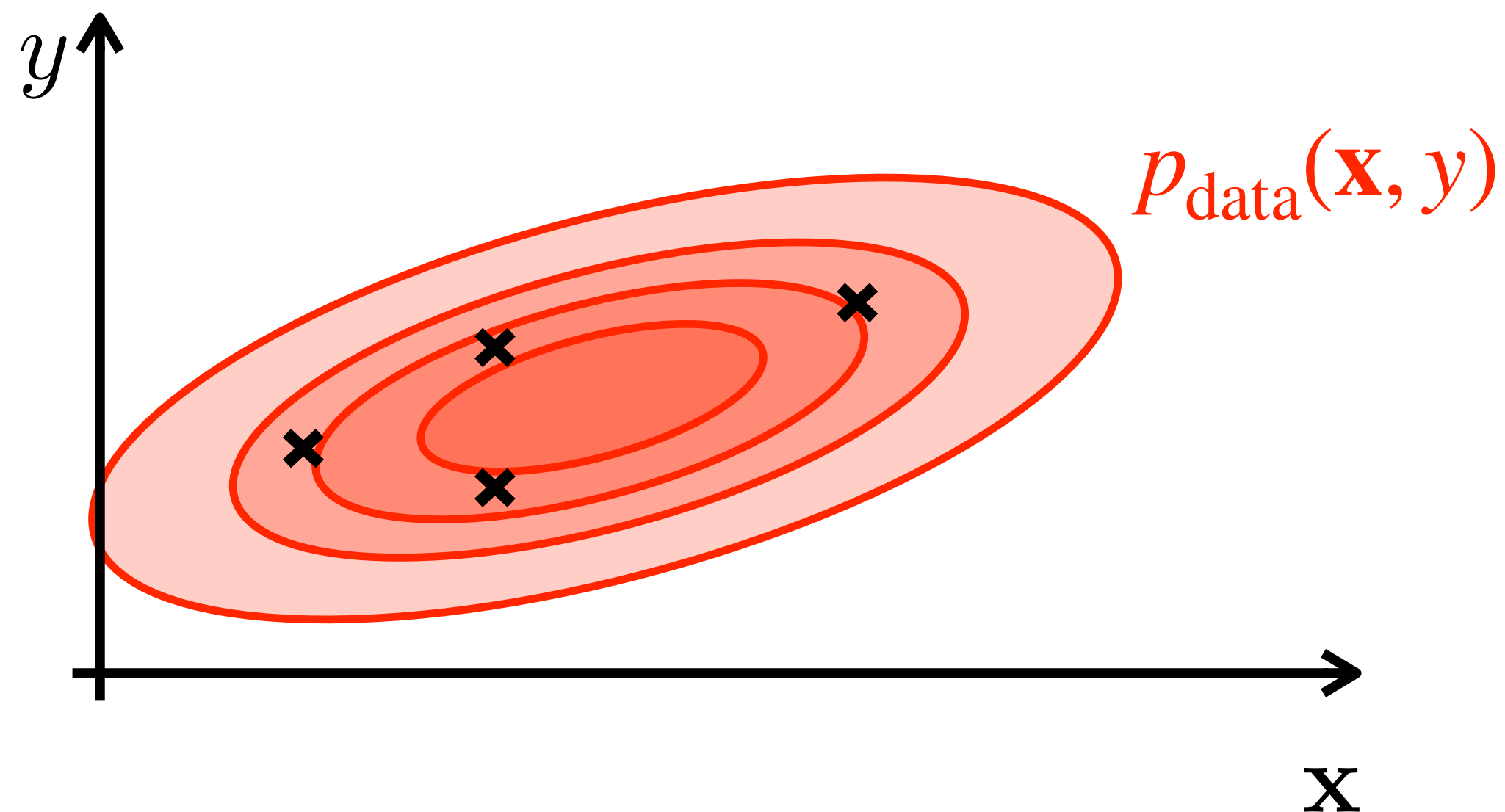
Or any generative model



How to suppress overfitting?

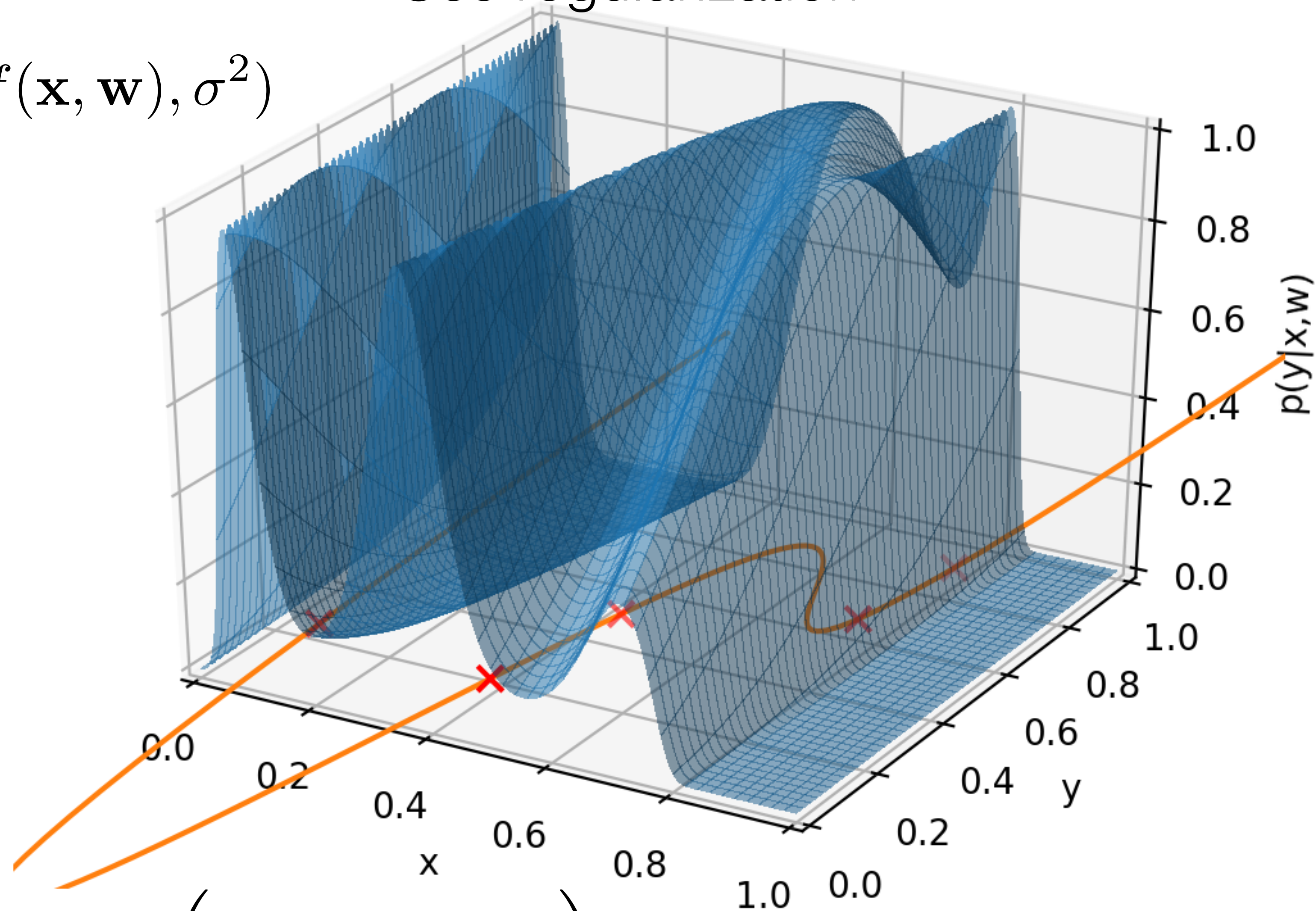
Use close-to-infinite dataset

- Some datasets grows faster than we can learn from them:
 - Colorizing images (1074 imgs/sec uploaded to Instagram)
 - Autonomous cars (predicting vertical acceleration from images)
 - Youtube videos (predicting keywords in comments from video)
- The learning bottleneck stems from computational limitations (not from trn size).



How to suppress overfitting?
Use regularization

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

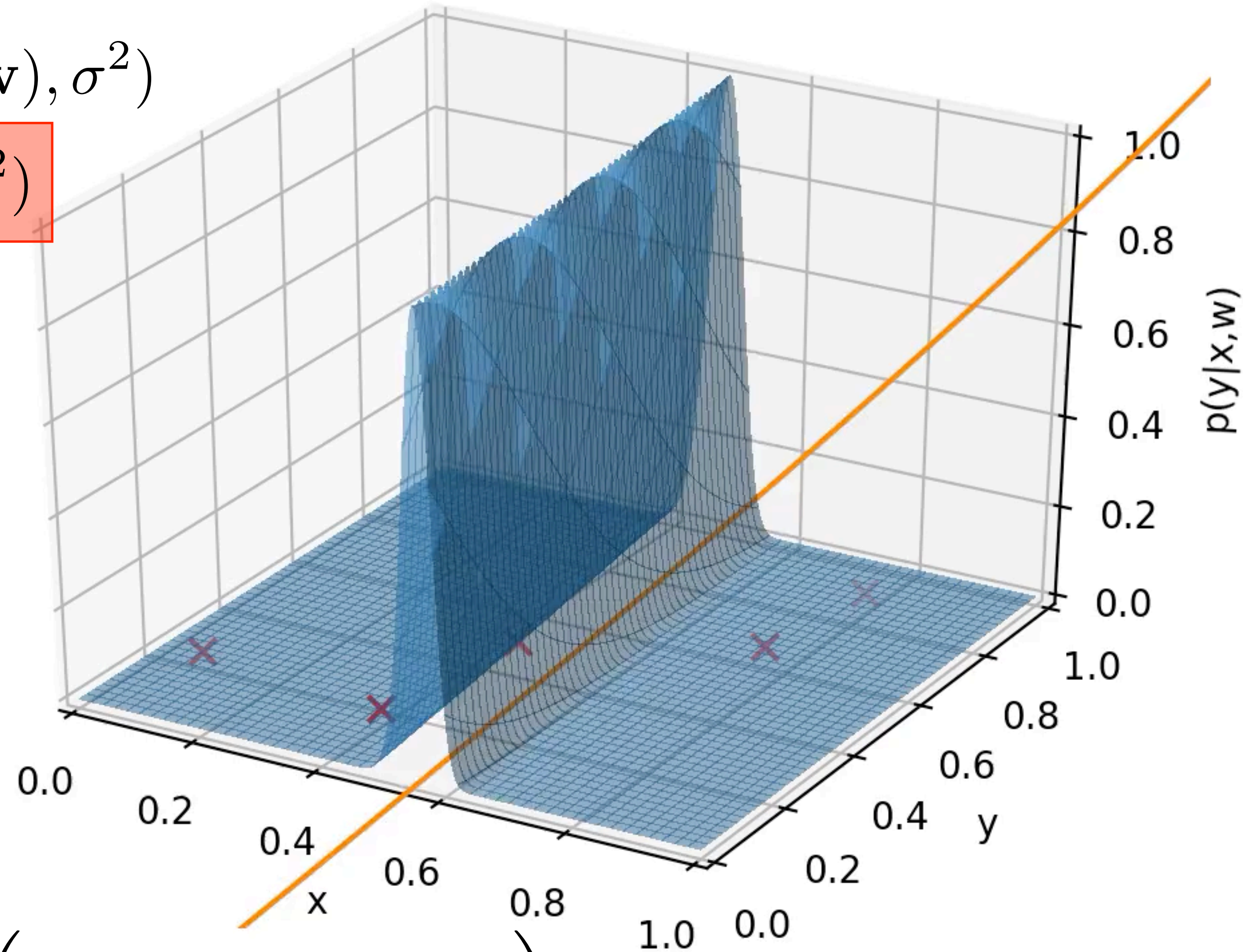


$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \left(\prod_i p(y_i | \mathbf{x}_i, \mathbf{w}) \right) = \arg \min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

How to suppress overfitting?
Use regularization

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

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



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

Where does the **overfitting** come from?

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

Take home message: **Optimization** $\neq$ **Machine learning**

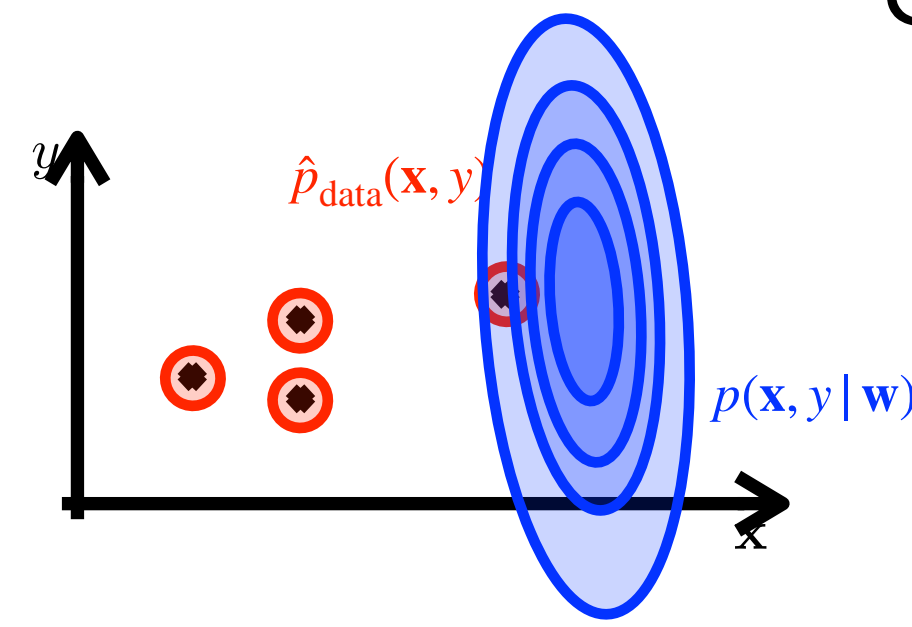
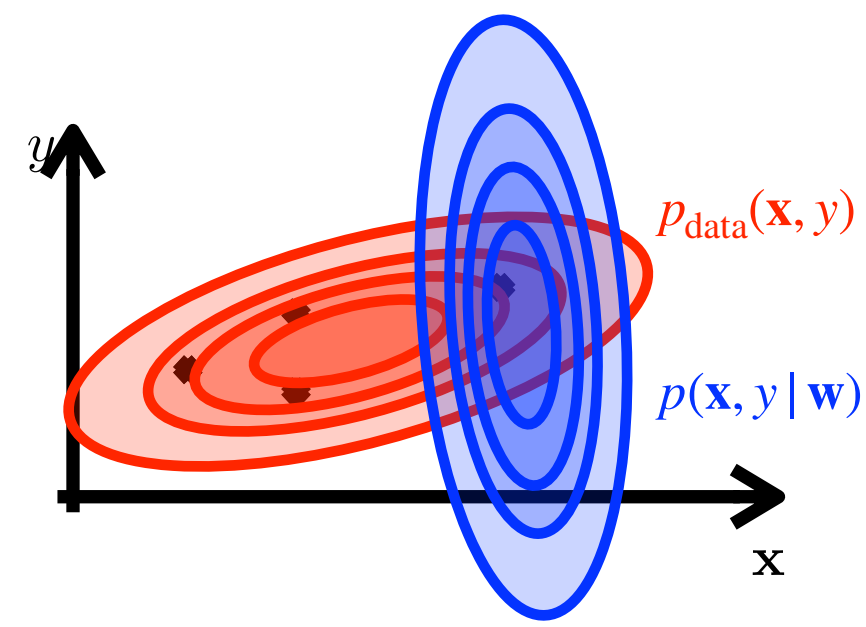
Machine learning is optimization of the criterion, we do not have access to.

Therefore approximated criterion is optimized instead

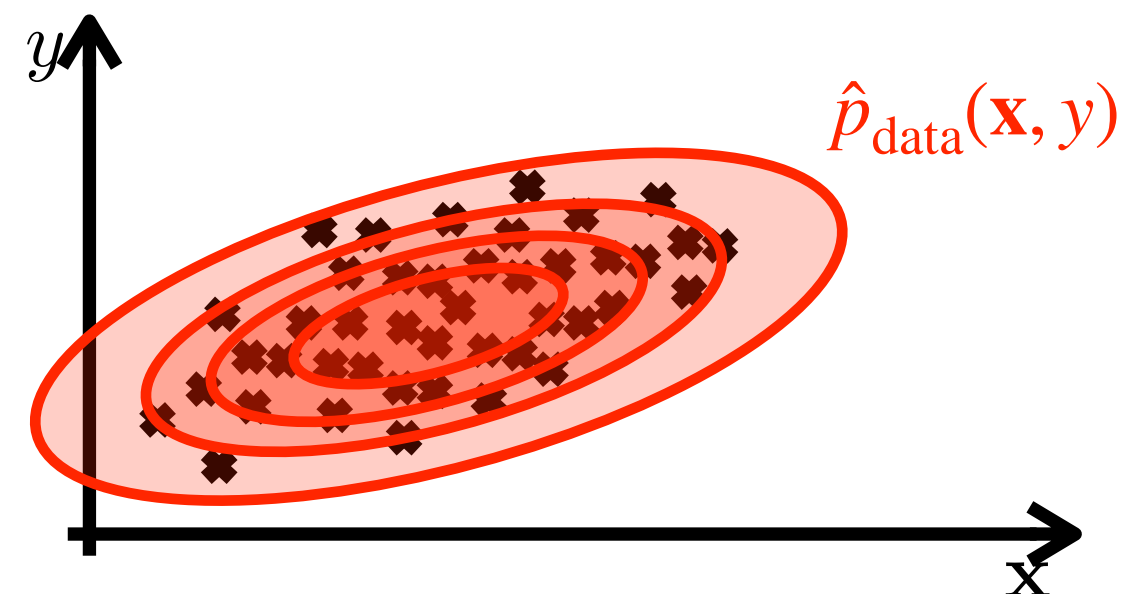
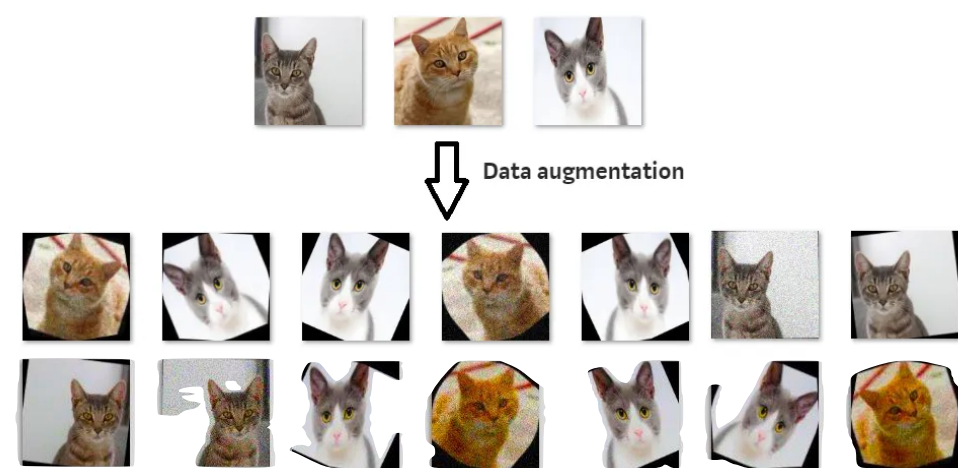
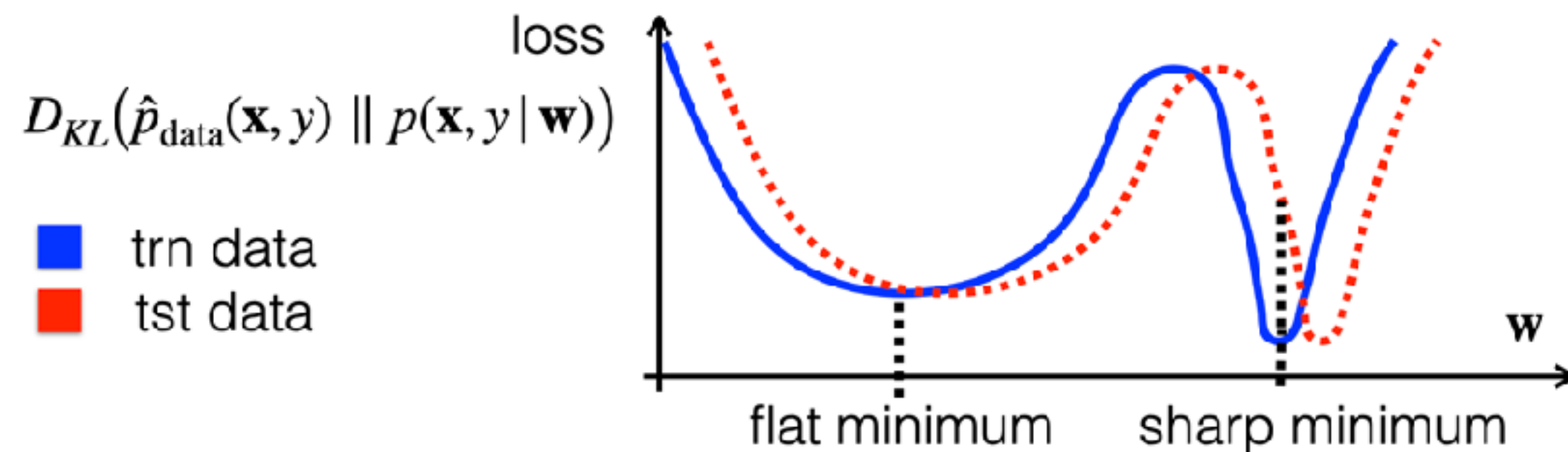
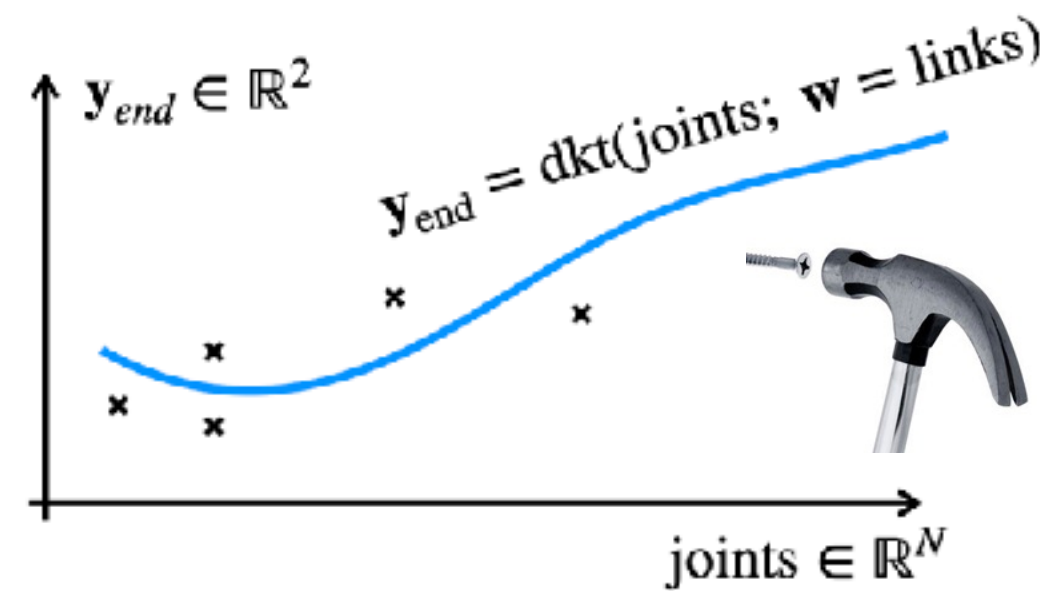
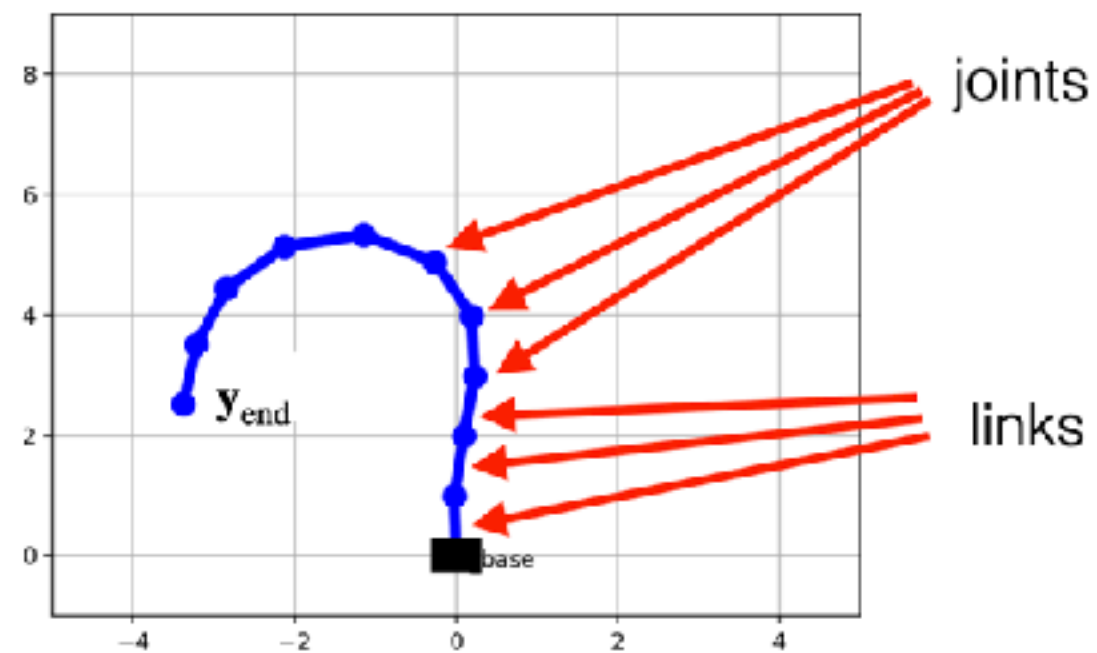
Suppress overfitting =

- 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 $D_{KL}(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y | \mathbf{w}))$
- 3) Use close-to-infinite dataset

Summary overfitting



Optimization $\neq$ Machine learning $\Rightarrow$ Overfitting



Always use the right tool/model

- Avoid any “*not-well justified leprechauns*” in the model

**Prefer simpler solutions
(flat minima, lower weights, ...)**

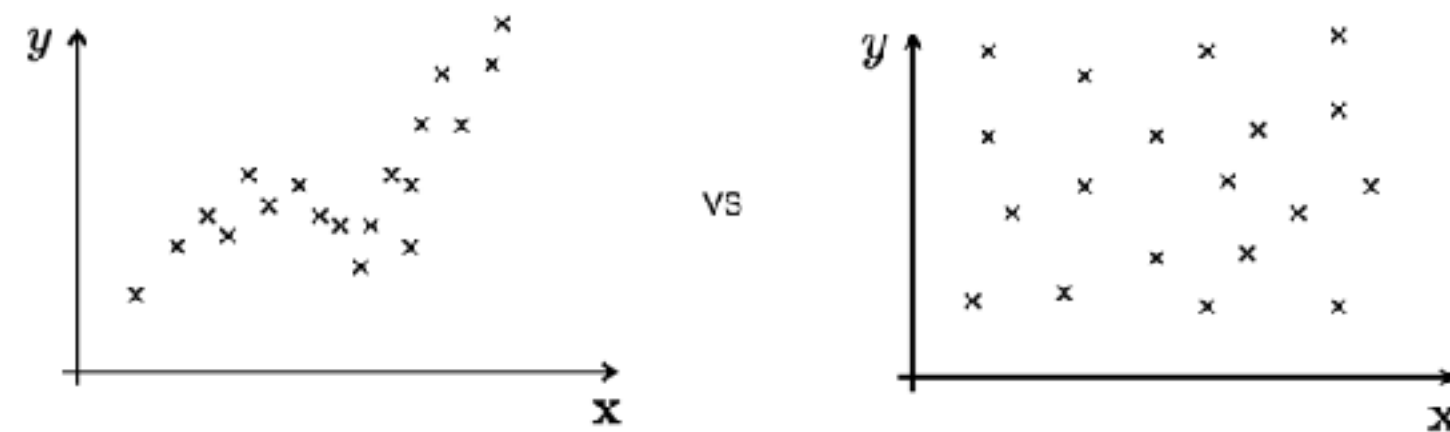
- Less is sometimes more

Use close-to-infinite training data

- More is sometimes more ;-)

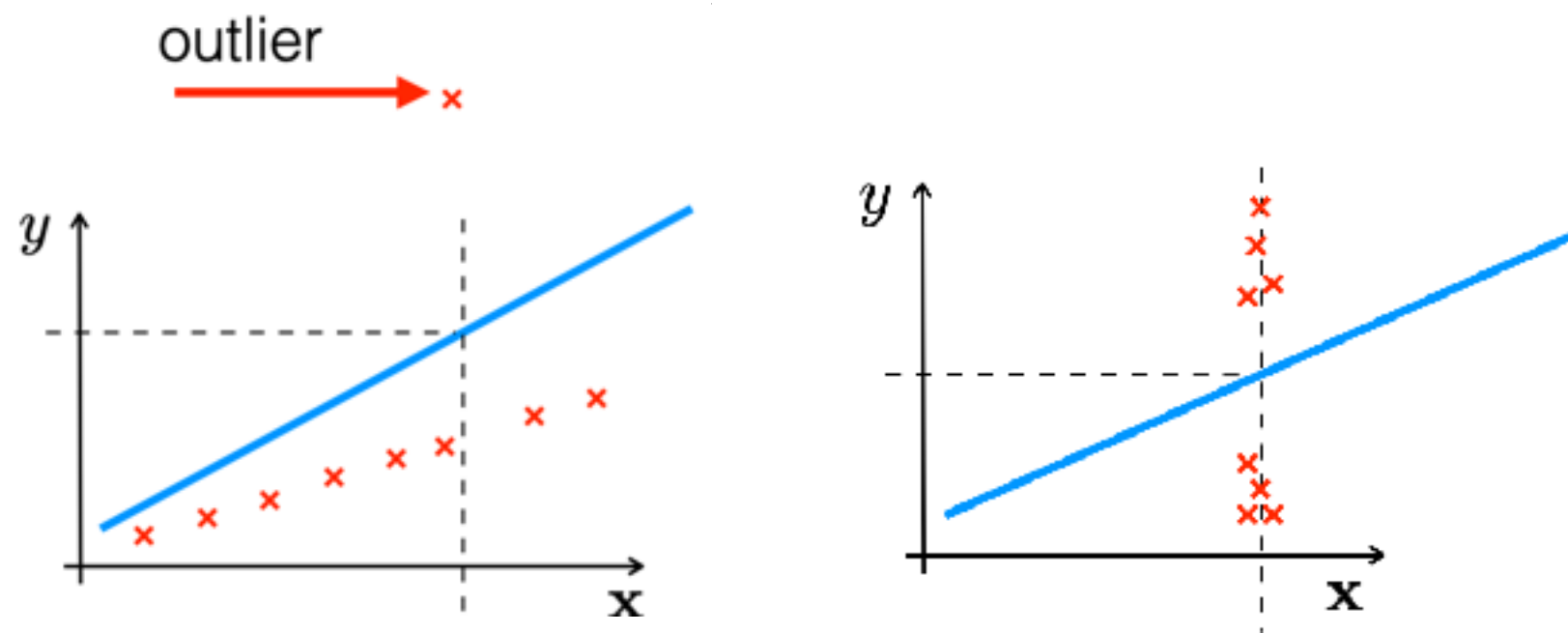
Golden grale:

- Solve only “Pilcik-free” problems



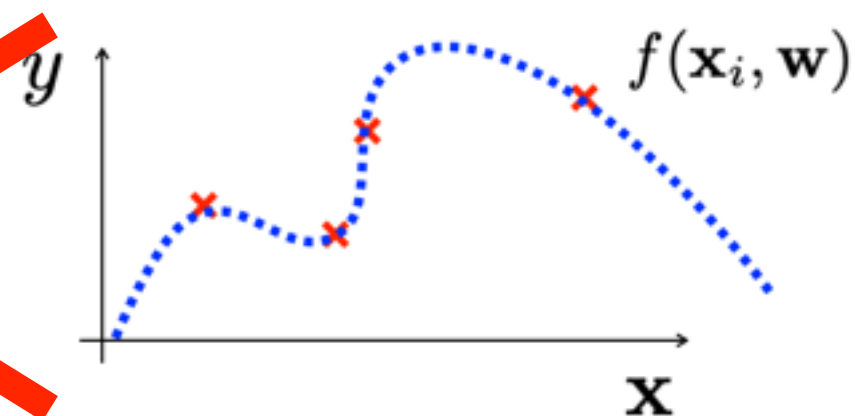
Lecture 1

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



This
Lecture

- Provide “differentiable” + “lepricon-free” + “strong prior” model.



Lecture on
ConvNets
 ∇ ODE, ∇ argmin

- Avoid traps in learning

Lecture on
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.