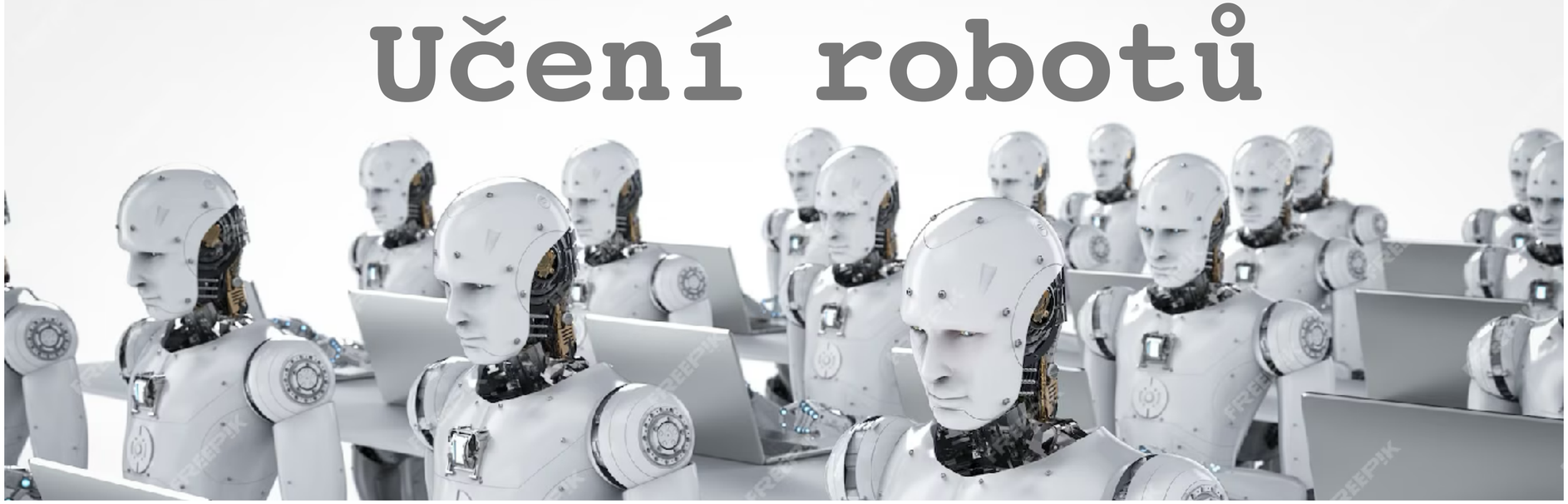


Where the hell does the loss and the overfitting come from?

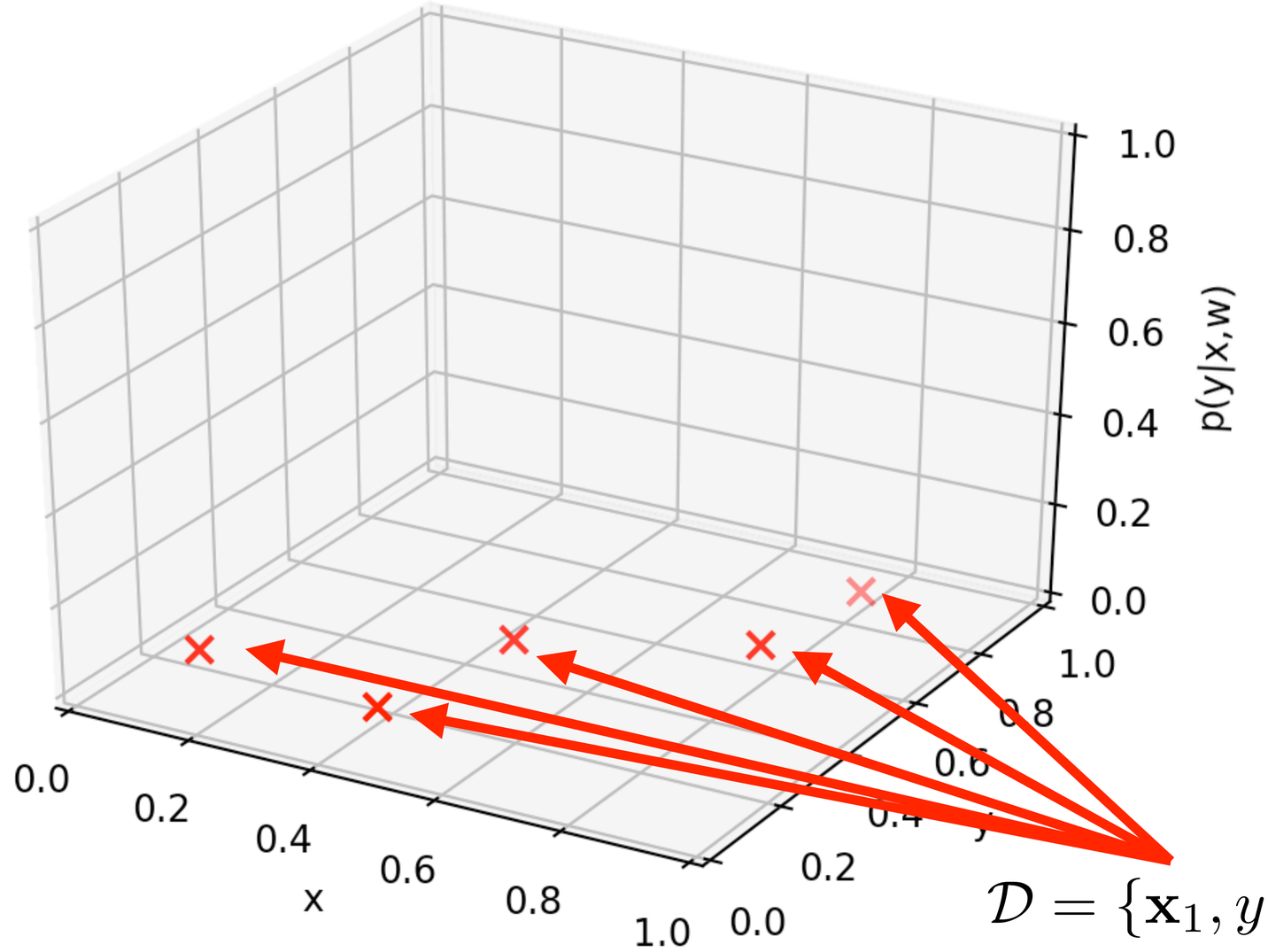
KL divergence, MAP, MLE view of regression, classification and overfitting source and countermeasures

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

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

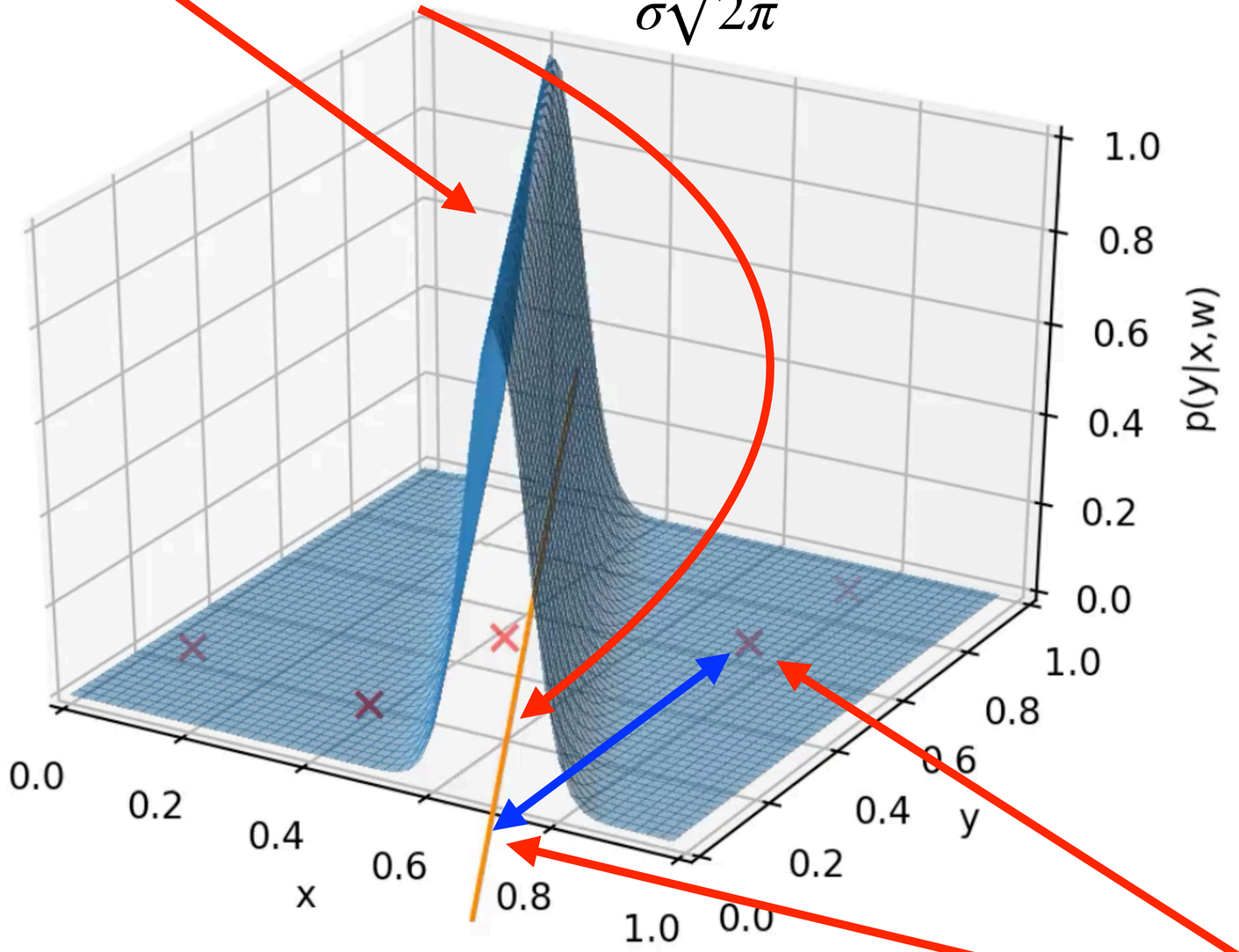
Prerequisites: MLE

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



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

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



Prove it !!!

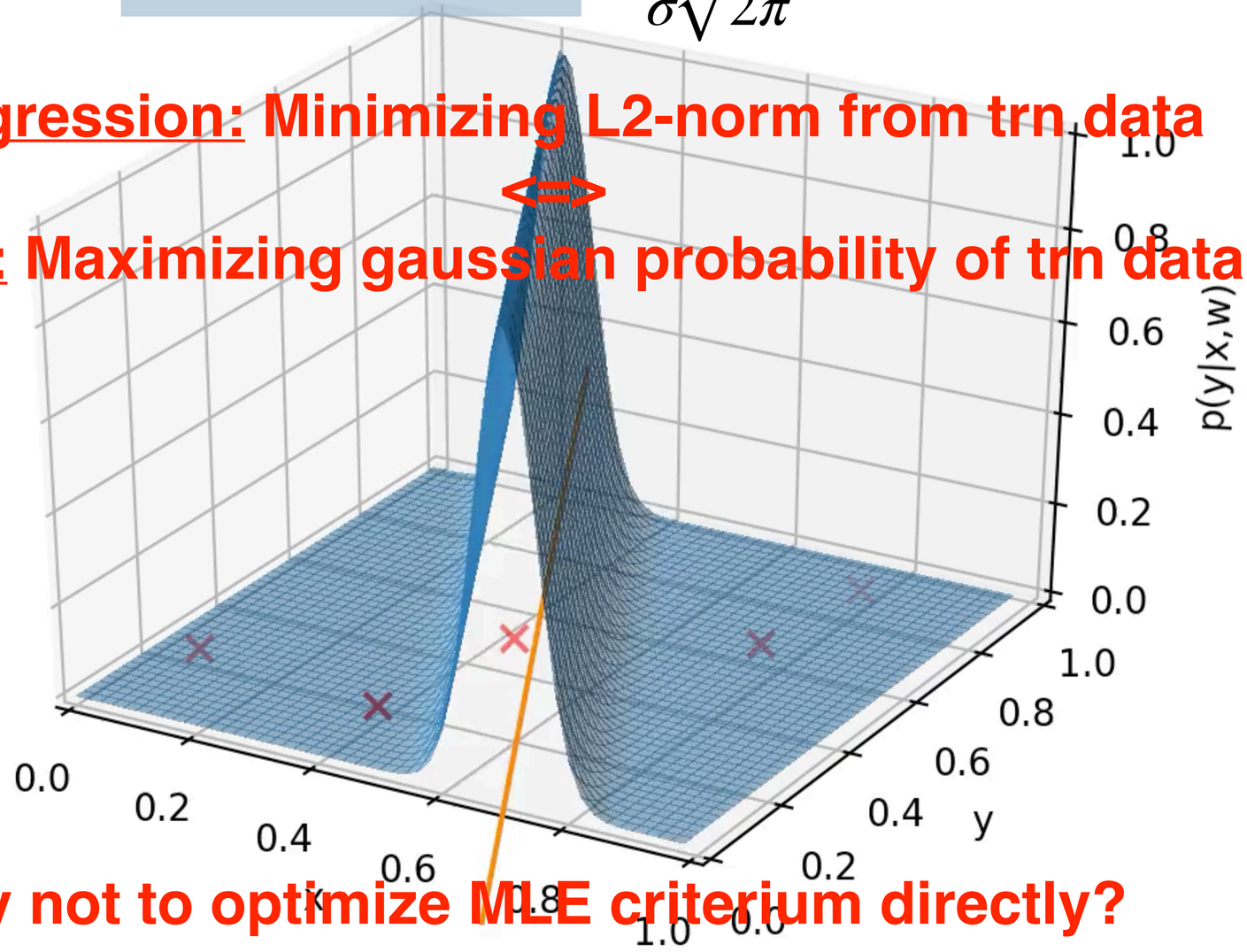
$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_i \mathcal{N}(y_i;\ w_1 x_i + w_0, \sigma) = \arg\min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$$

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

Regression: Minimizing L2-norm from trn data

<=>

MLE: Maximizing gaussian probability of trn data

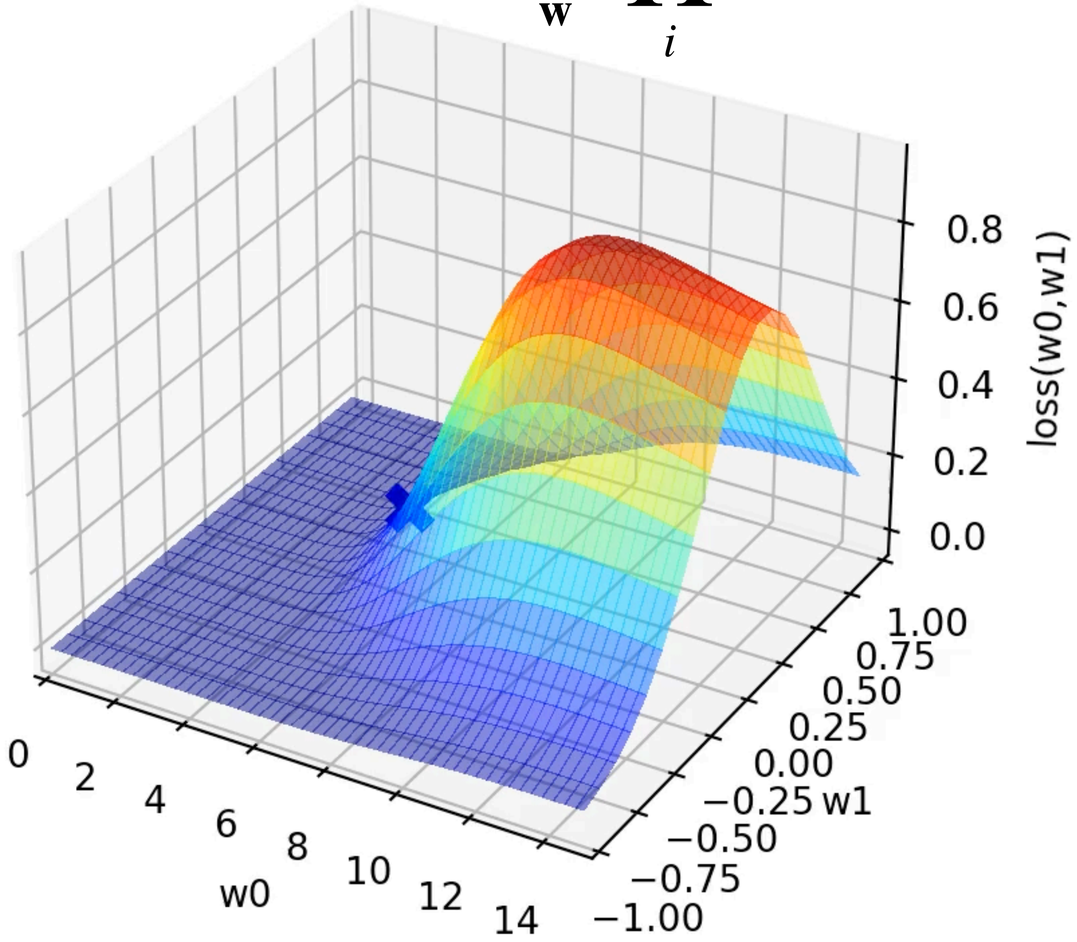


Why not to optimize MLE criterium directly?

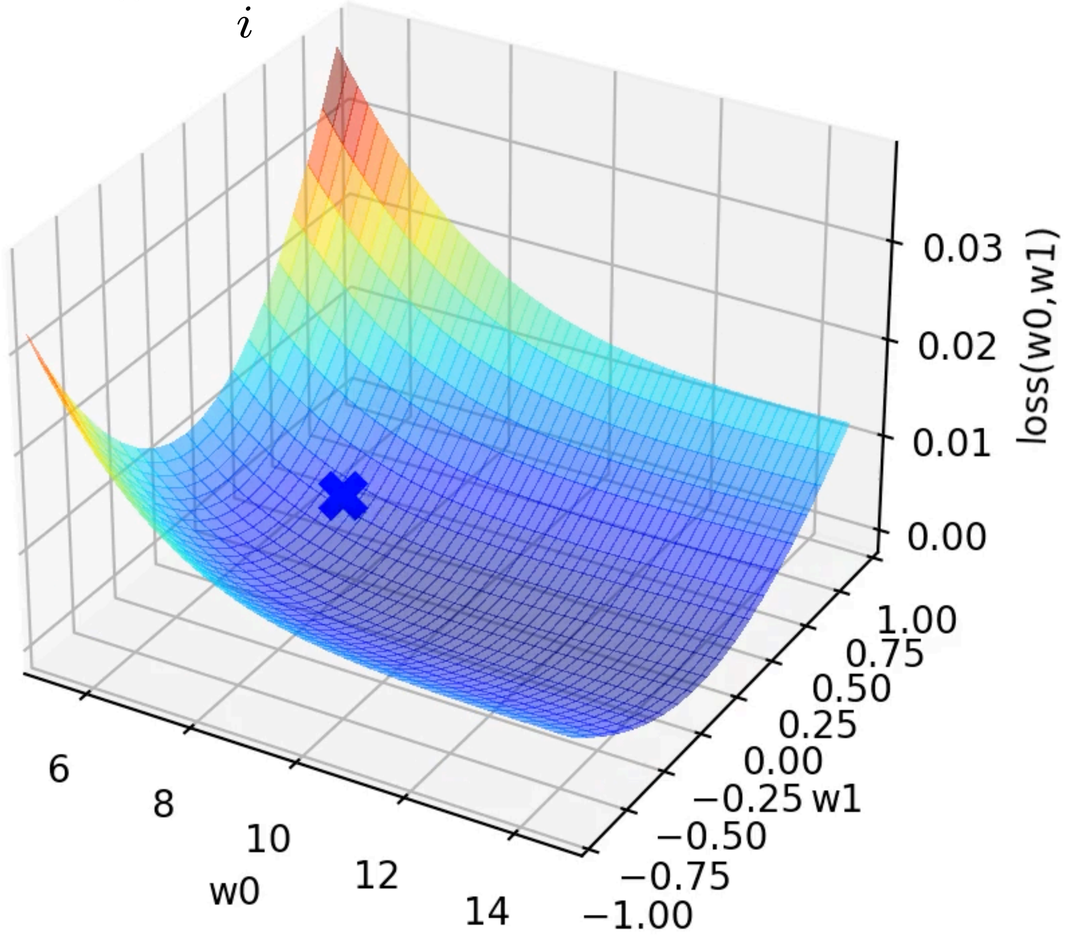
$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_i \mathcal{N}(y_i;\ f(\mathbf{x}_i, \mathbf{w}), \sigma) = \arg\min_{\mathbf{w}} \sum_i (w_1 x_i + w_0 - y_i)^2$$

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

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



MLE



LSQ

Prerequisites: Bayes theorem

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

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

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

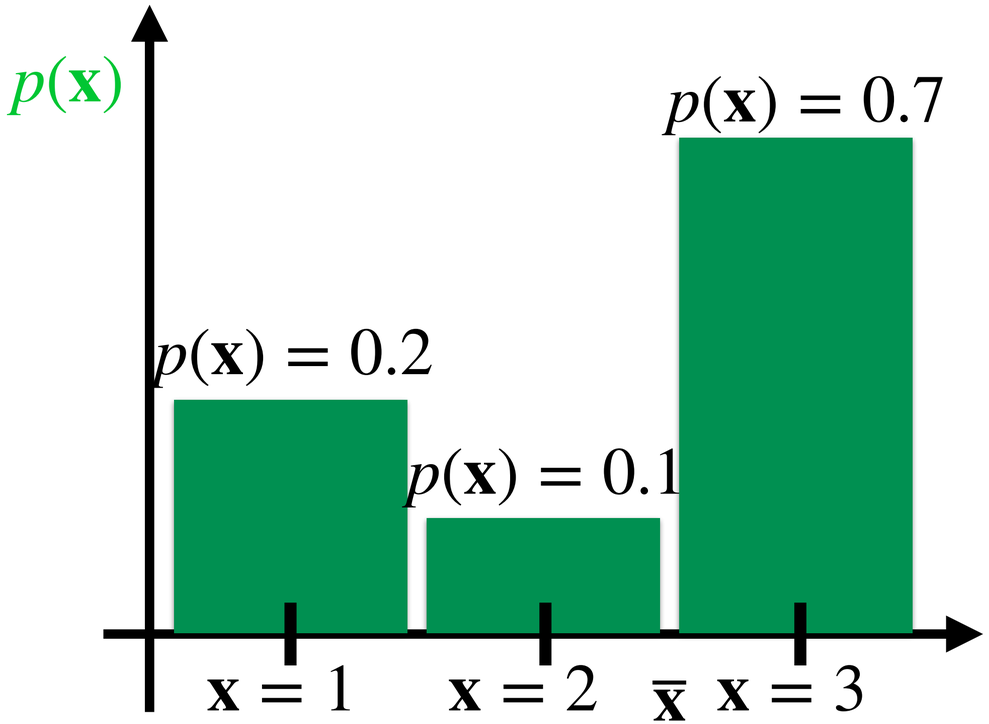
Let's put it together: $p(A \mid B) \cdot p(B) = p(A, B) = p(A) \cdot p(B)$

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

Prerequisites: Mean and average

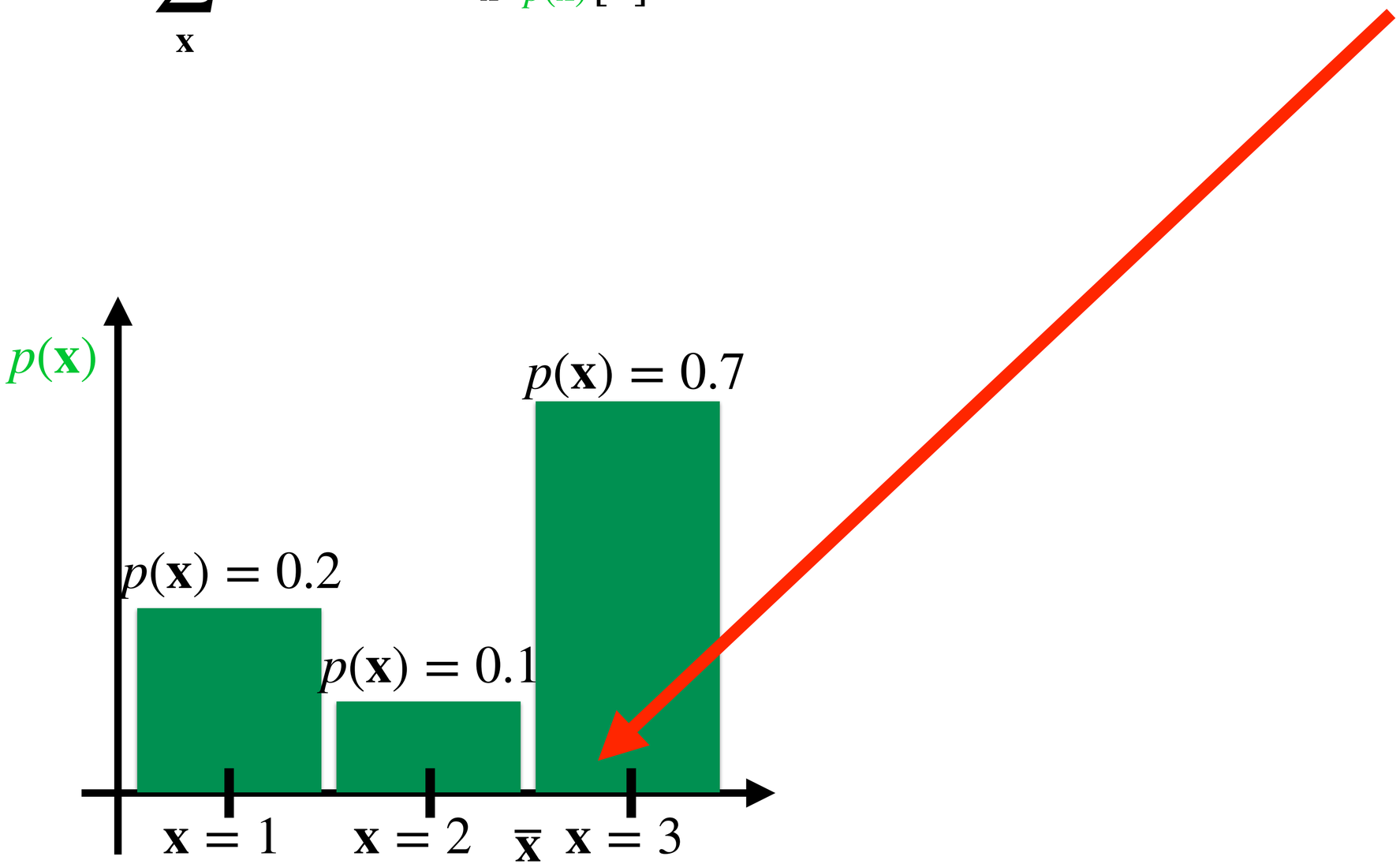
$$\bar{\mathbf{x}} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \mathbf{x} = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[\mathbf{x}] = \text{??}$$

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



Prerequisites: Mean and average

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

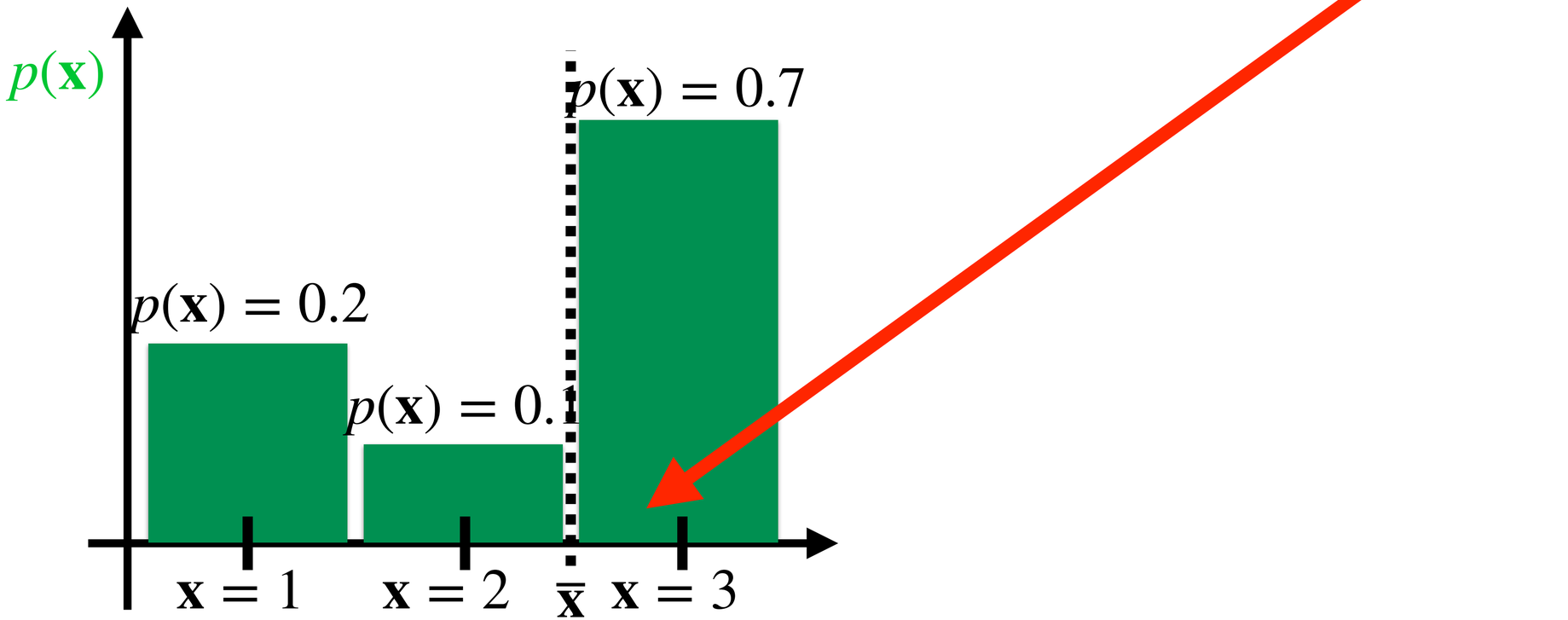


Prerequisites: Mean and average

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

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

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



Prerequisites: Mean and average

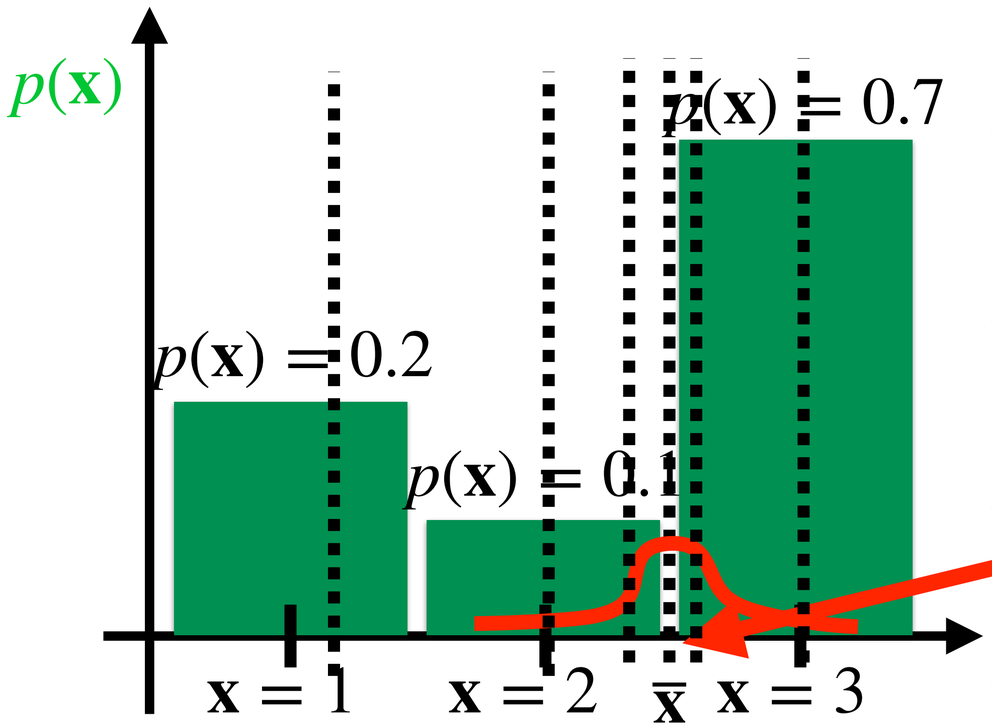
$$\bar{\mathbf{x}} = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\mathbf{x}] = 0.2 \cdot 1 \; + \; 0.1 \cdot 2 \; + \; 0.7 \cdot 3 = 2.5$$

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

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

For $N \to \infty$

$$\mathcal{N}(\bar{\mathbf{x}}_i; \; \bar{\mathbf{x}}, \frac{\sigma_{\mathbf{x}}^2}{\sqrt{N}})$$



$$\bar{\mathbf{x}}_1 = \frac{1}{10}(1+1+1+1+3+3+3+3+3+3) = 2.2$$

$$\bar{\mathbf{x}}_2 = \frac{1}{10}(3+3+3+3+3+3+3+3+3+3) = 3.0$$

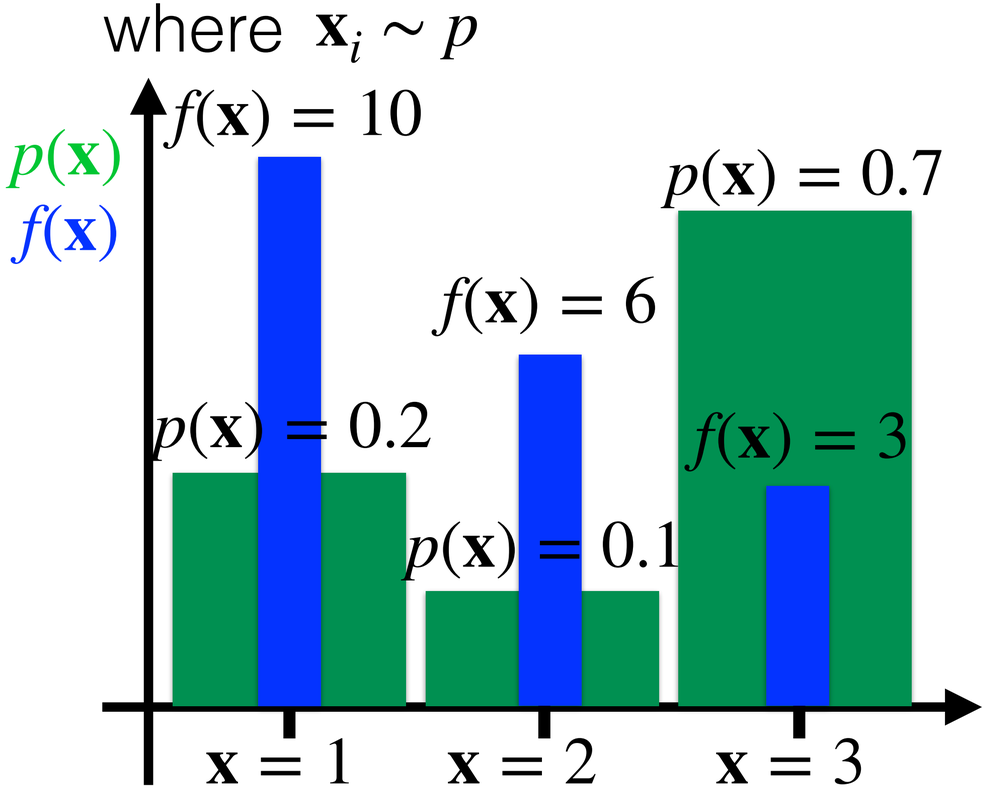
$$\bar{\mathbf{x}}_3 = \frac{1}{10}(2+2+2+2+3+3+3+3+3+3) = 2.6$$

$$\bar{\mathbf{x}}_4 = \frac{1}{10}(1+1+1+1+1+1+1+1+2+2) = 1.2$$

$$\bar{\mathbf{x}}_5 = \frac{1}{10}(1+1+1+1+1+3+3+3+3+3) = 2.0$$

Prerequisites: Mean and average

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

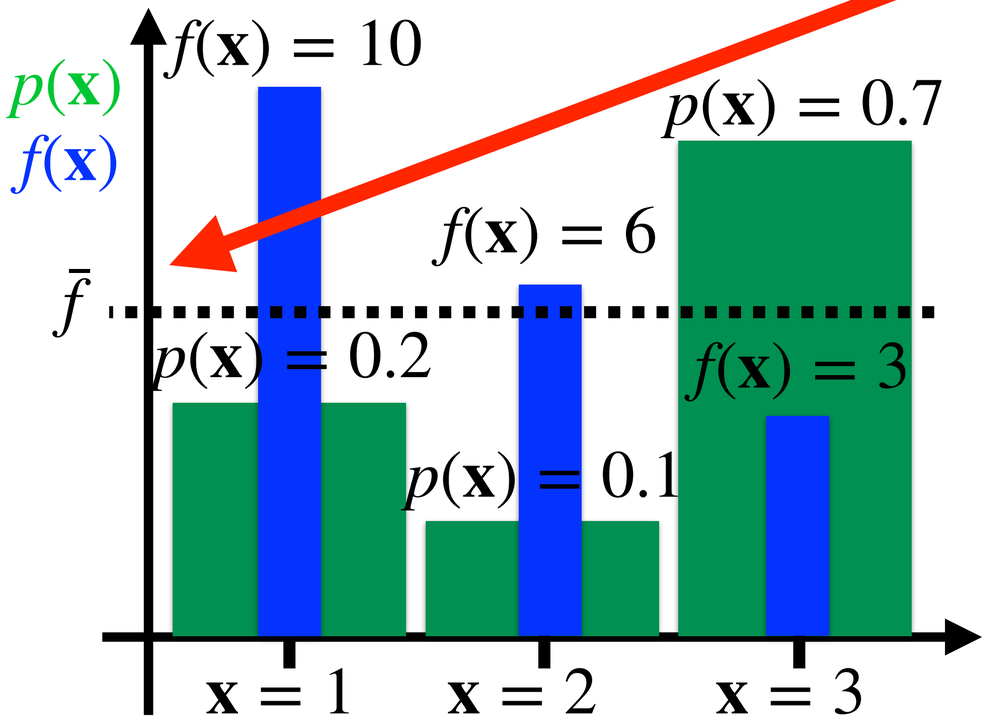


Prerequisites: Mean and average

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

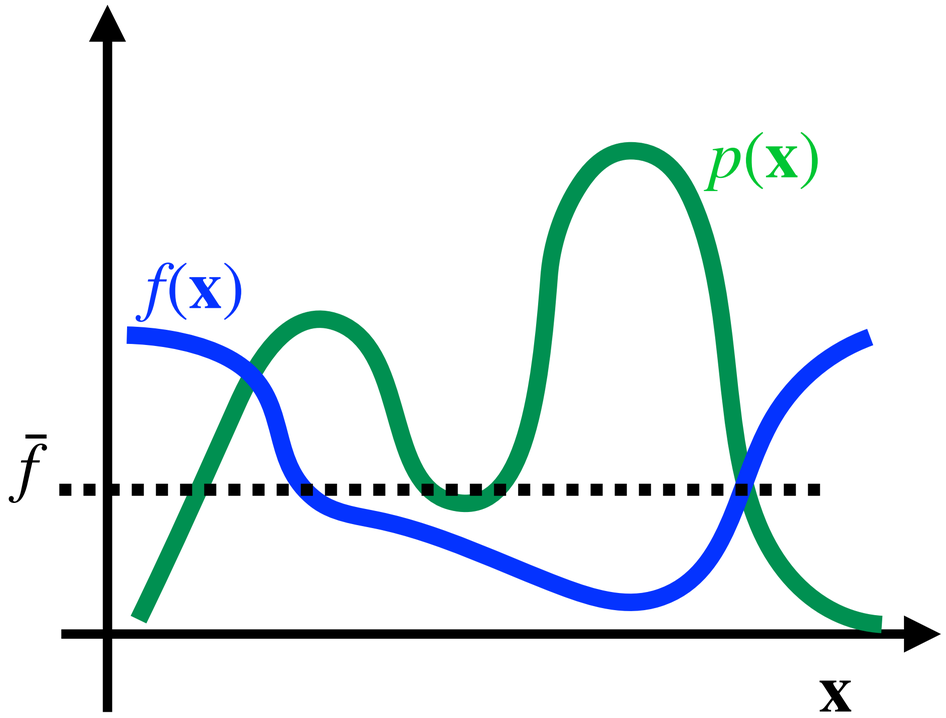
$$\approx \frac{1}{N} \sum_{i} f(\mathbf{x}_i) = \frac{1}{10}(10 + 10 + 6 + 3 + 3 + 3 + 3 + 3 + 3 + 3) = 4.7$$

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



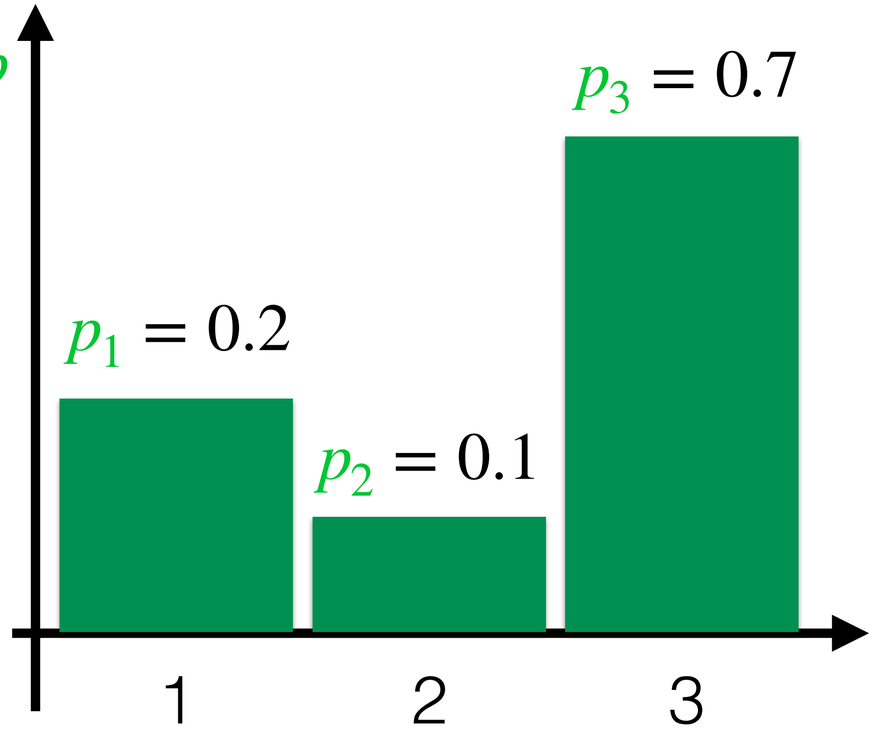
Prerequisites: Mean and average

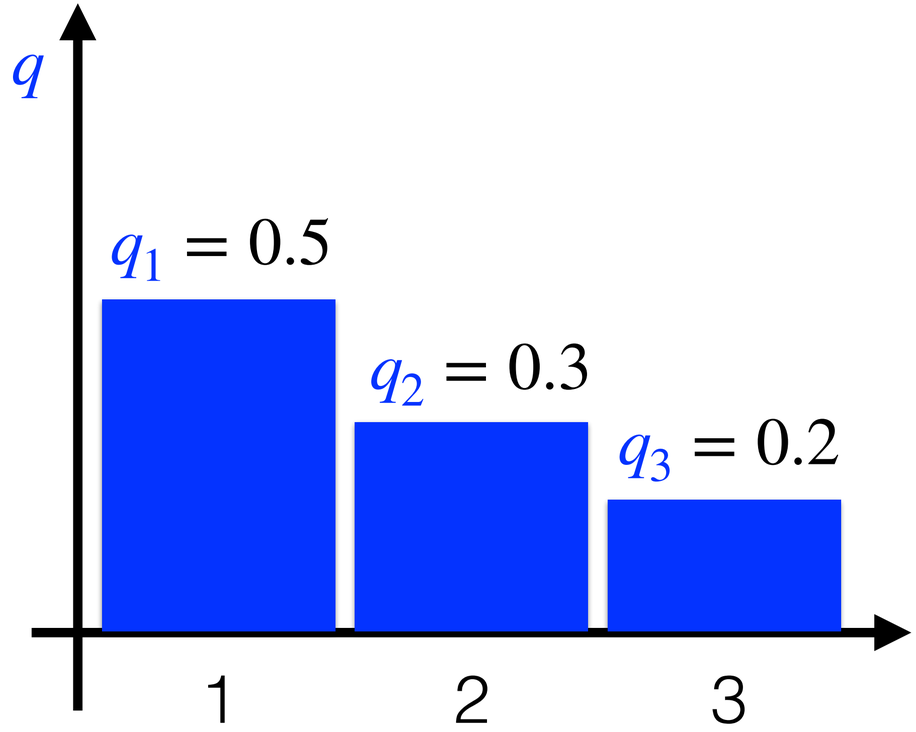
$$\bar{f} = \int_{\mathbf{x}} p(\mathbf{x}) \cdot f(\mathbf{x})\, d\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f(\mathbf{x})\right] \approx \frac{1}{N}\sum_{i} f(\mathbf{x}_i) \qquad \text{where } \mathbf{x}_i \sim p$$



Prerequisites: KL-divergence

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



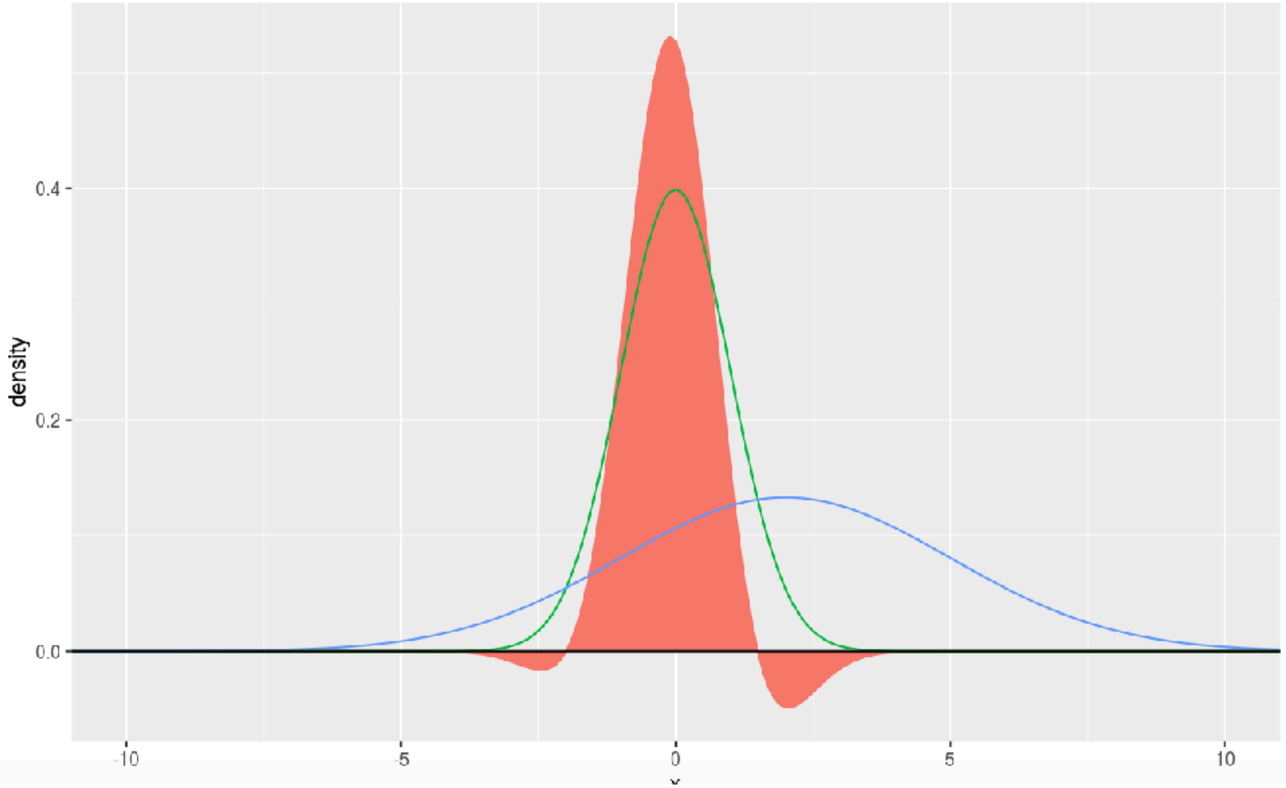


- What if $q_k = p_k$ for k=1,2,3 ?
- What if $q_k = 0$ and $p_k > 0$?
- Is it symmetrical $D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$?

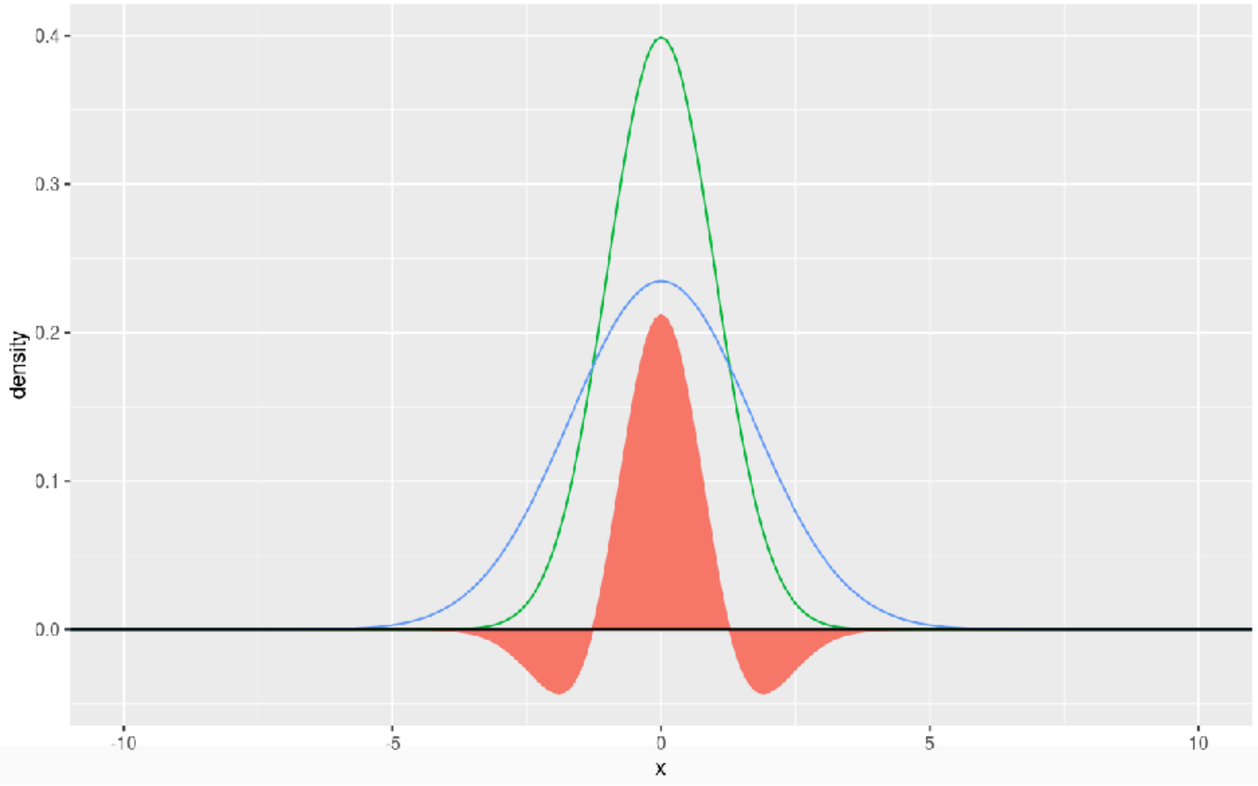
Prerequisites: KL-divergence

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

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



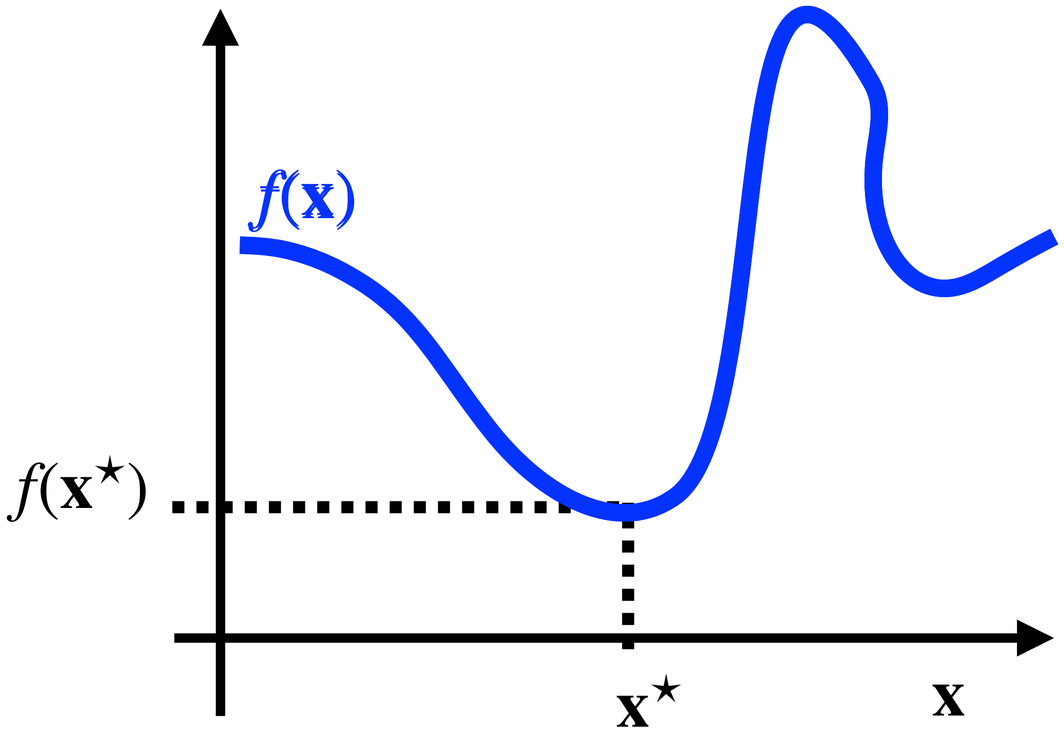
$D_{KL}\big(p(\mathbf{x}) \parallel q(\mathbf{x})\big)$ =0.2036



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

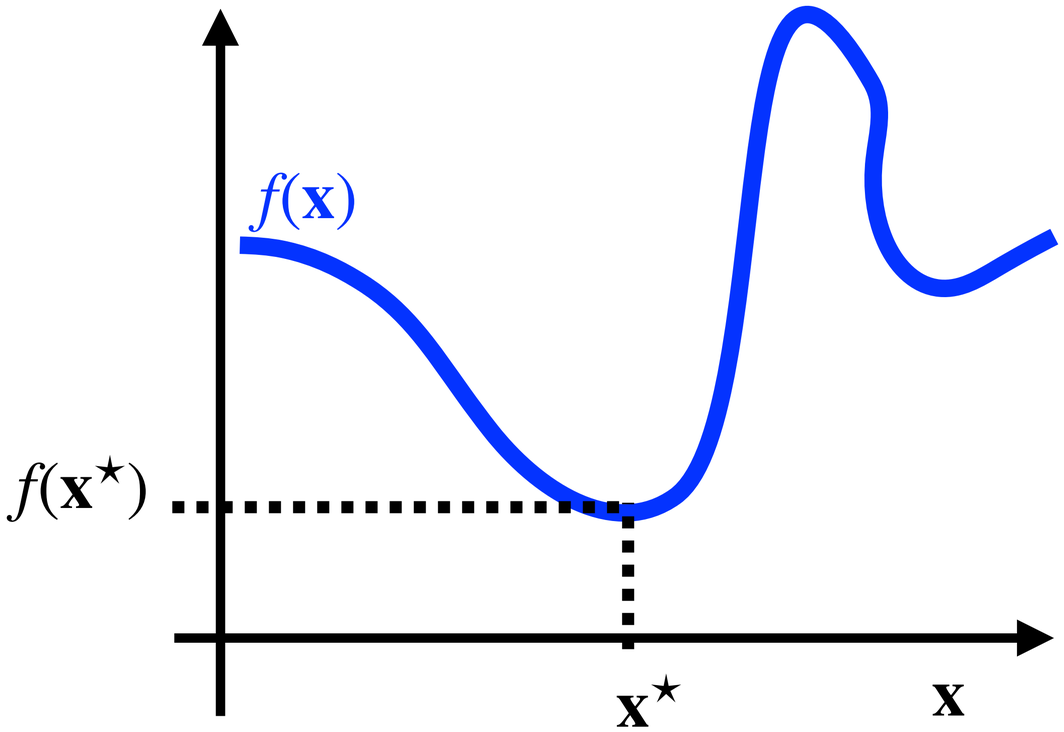
Prerequisites: argmin

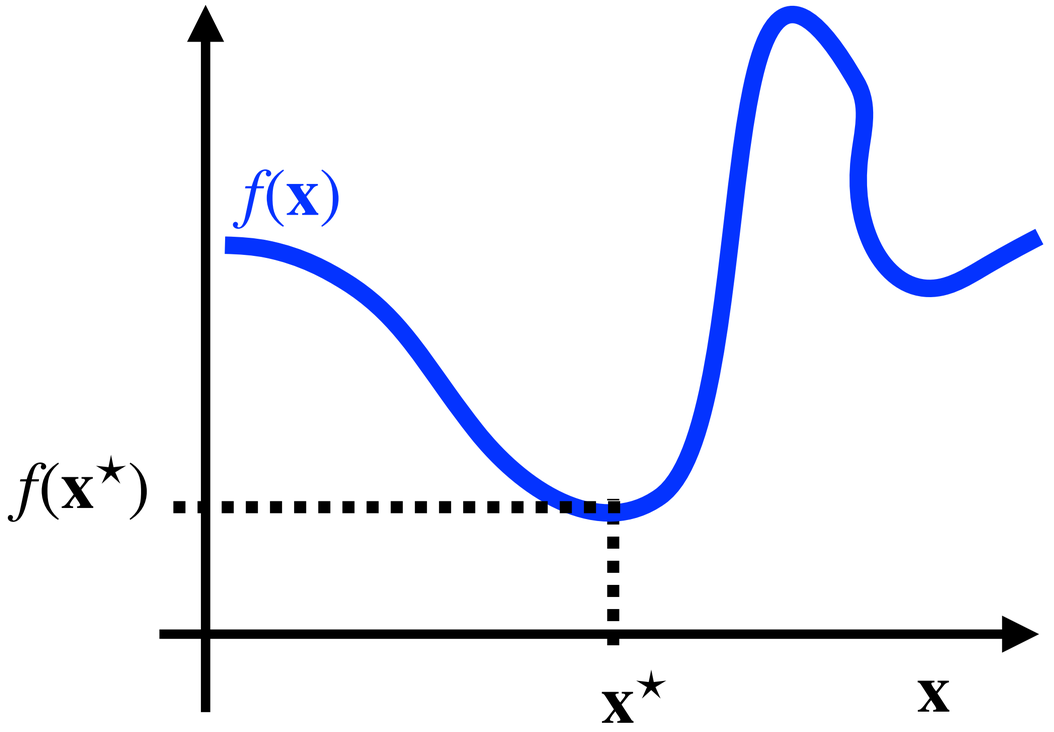
$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \log\big(f(\mathbf{x})\big)$$



Prerequisites: argmin

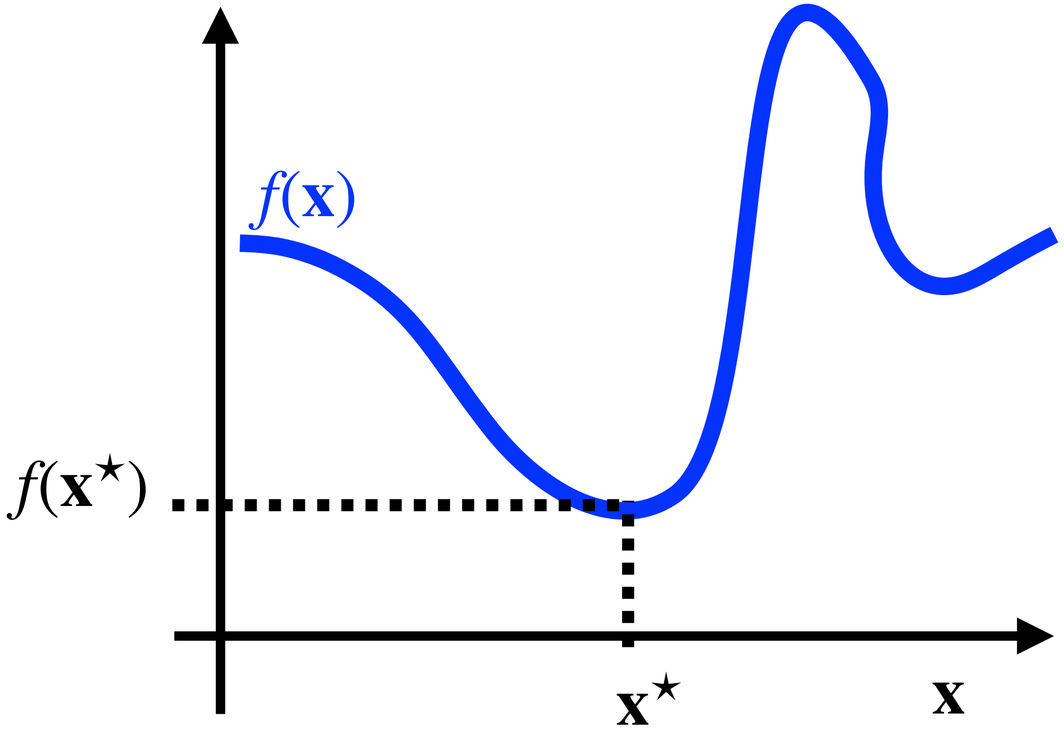
$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \log\big(f(\mathbf{x})\big)$$

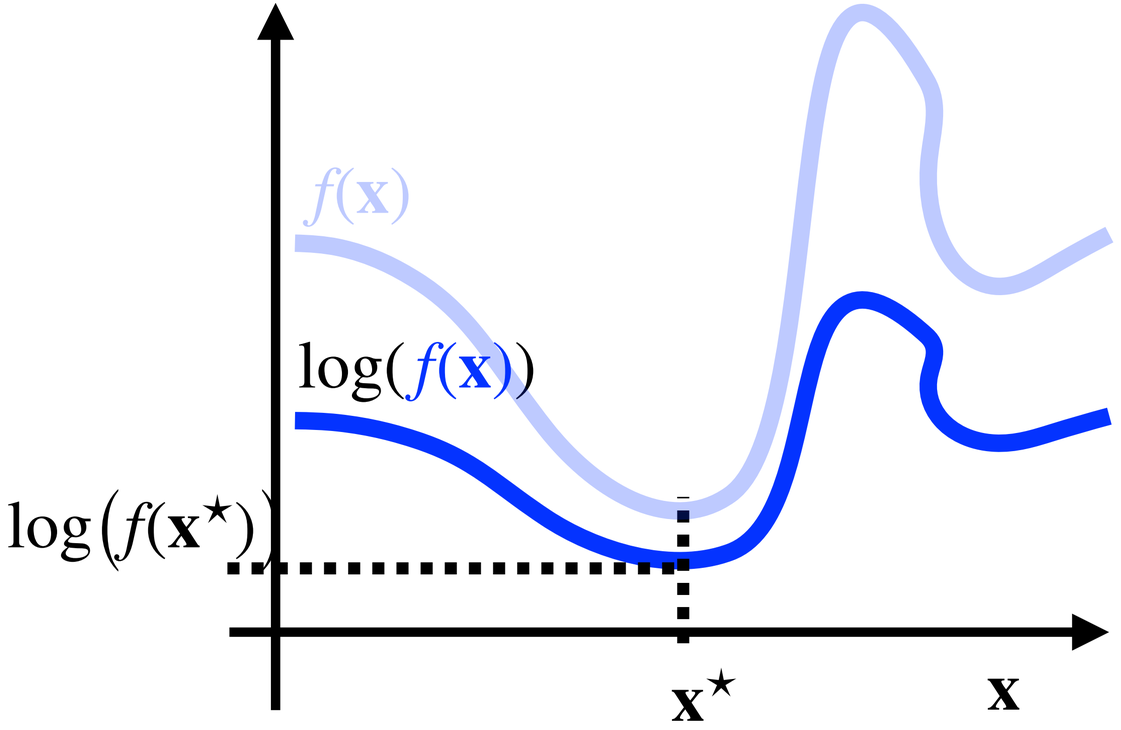




Prerequisites: argmin

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





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

How should we fit distribution into data?

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

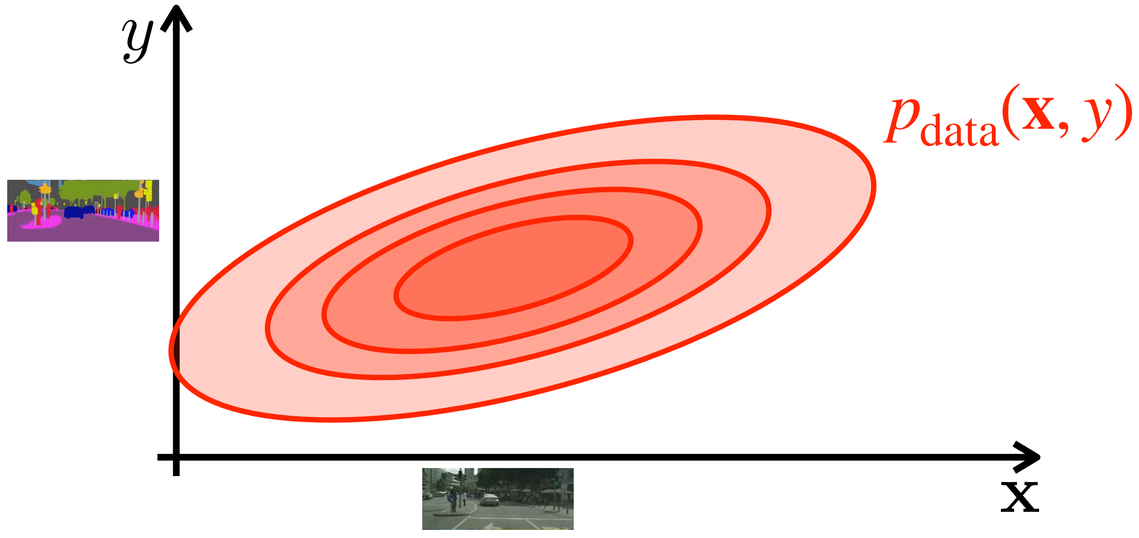
$\mathbf{x}$

y

road
sideway
pedestrian
traffic sign
trees
sky

Why should I minimize -log(p)?

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



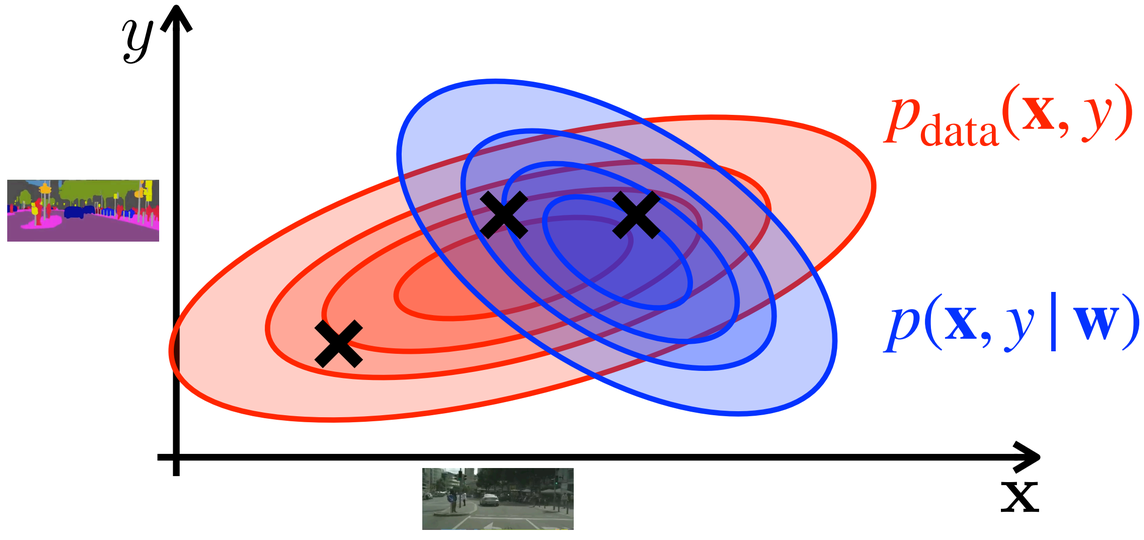
Why should I minimize -log(p)?

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

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

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

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



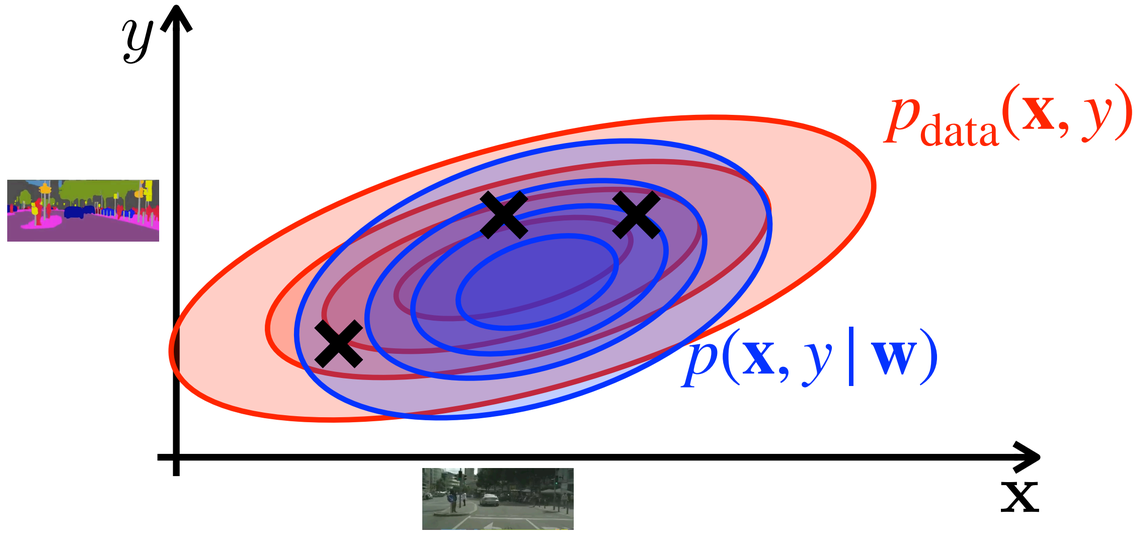
Why should I minimize -log(p)?

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

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

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

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



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

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

Regression

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

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

Classification

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

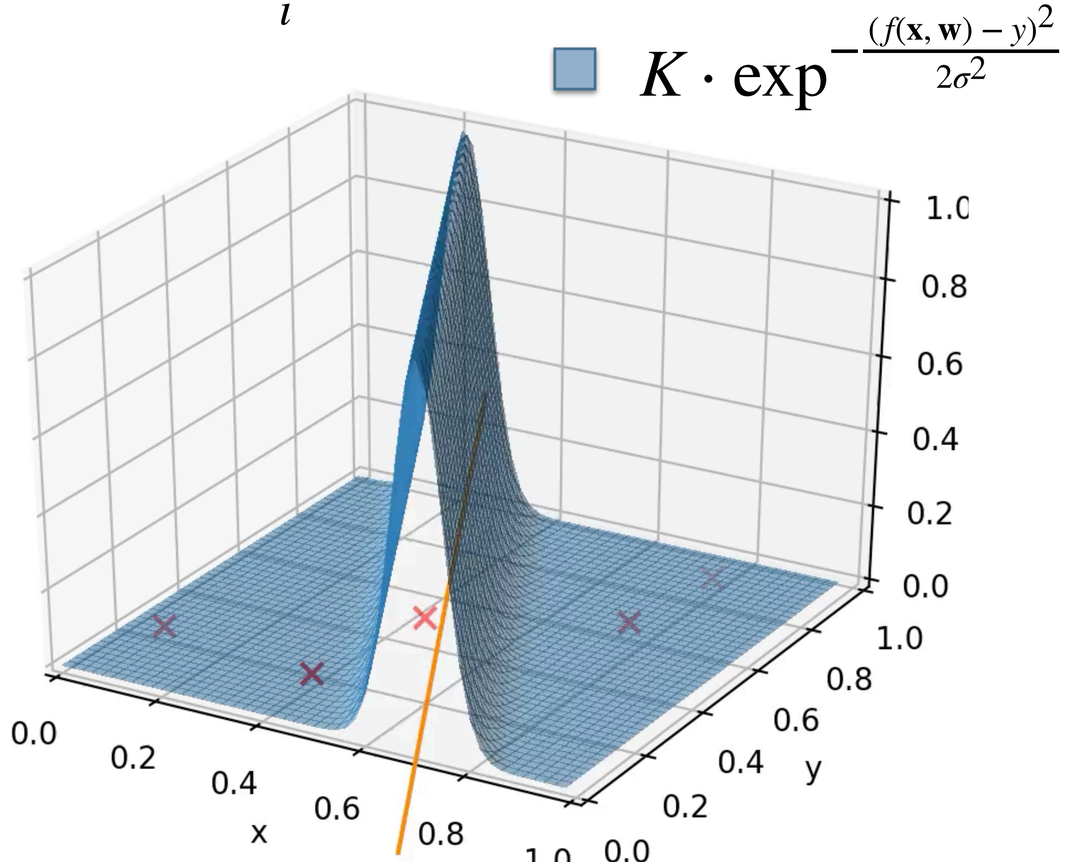
$$\mathscr{L}(\mathbf{w}) = \frac{1}{N} \sum_i \log\Big(1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w}))\Big)$$

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

Regression

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

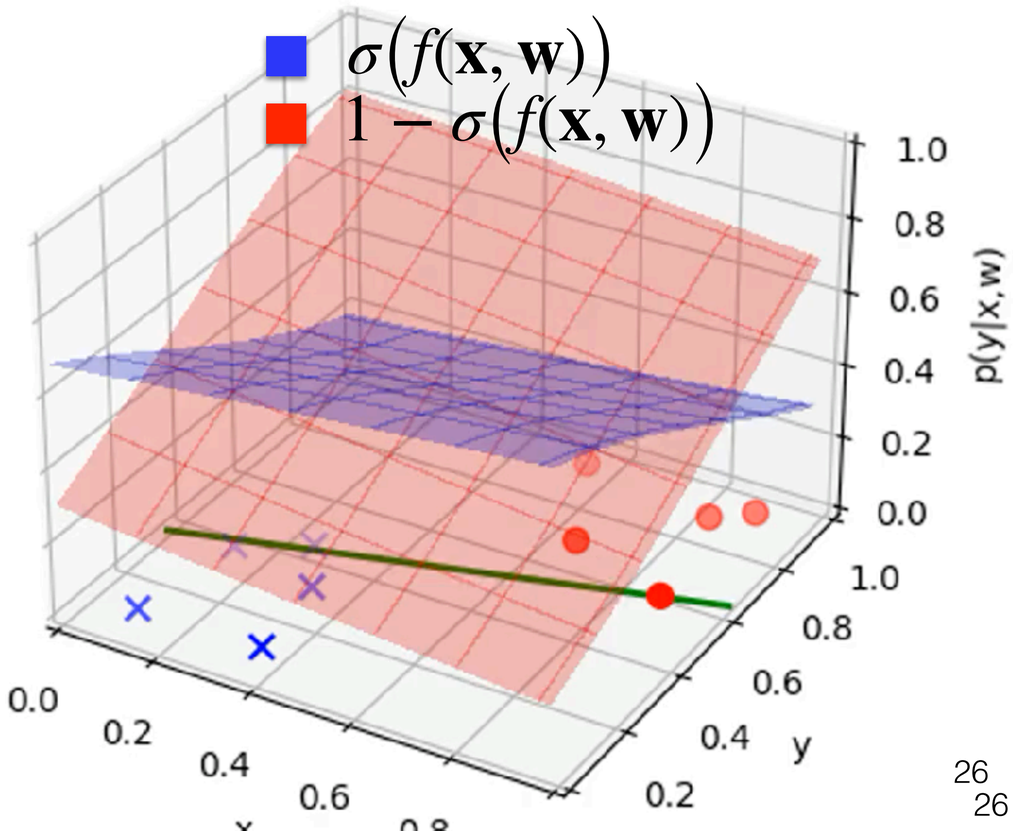
$$\mathscr{L}(\mathbf{w}) = \frac{1}{N} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$



Classification

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

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



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

$$\approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} I\Big(p(y_i \mid \mathbf{x}_i, \mathbf{w})\Big)$$

Information/suprisal minimization

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

$$\approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} I\Big(p(y_i \mid \mathbf{x}_i, \mathbf{w})\Big)$$

Information/suprisal minimization

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

$$\approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} I\big(p(y_i \mid \mathbf{x}_i, \mathbf{w})\big)$$

Information/surprisal minimization

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

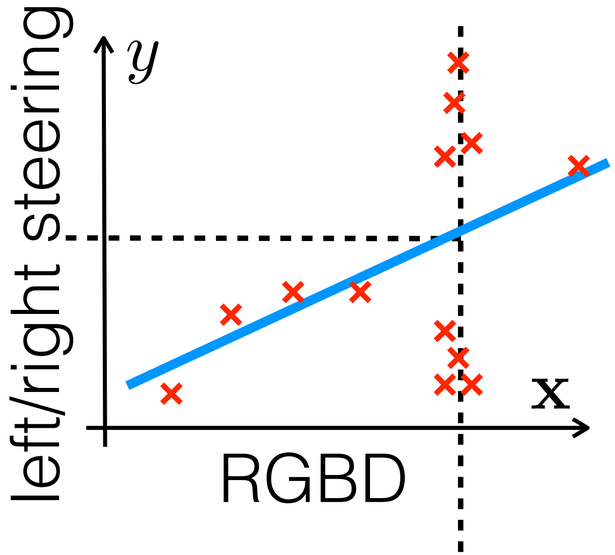
$$\approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} I\big(p(y_i \mid \mathbf{x}_i, \mathbf{w})\big)$$

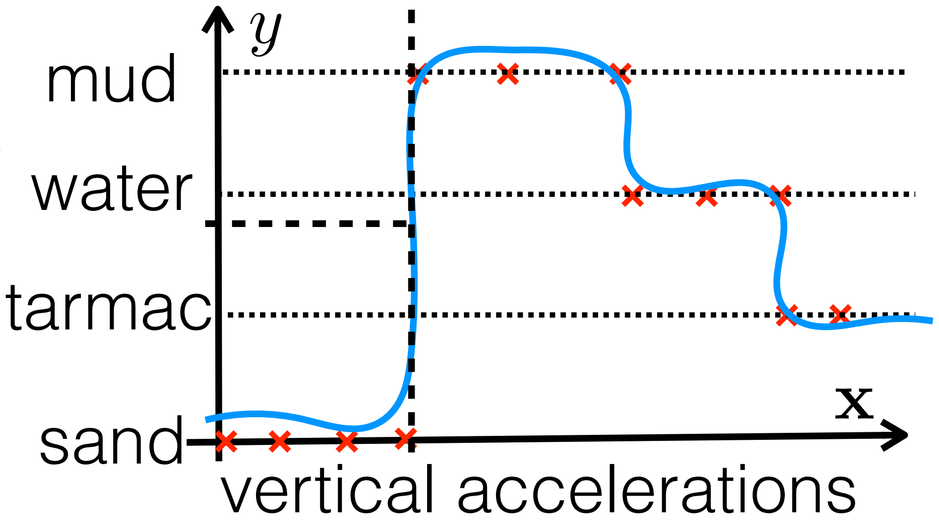
Information/suprisal minimization

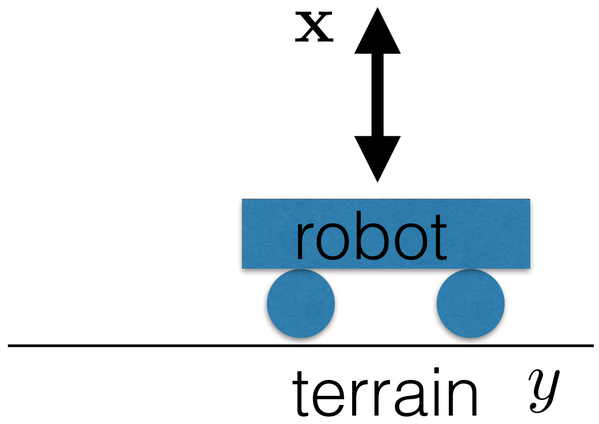
Pathak et al. Curiosity-Driven Learning: AI agents exploring without looking at any scores

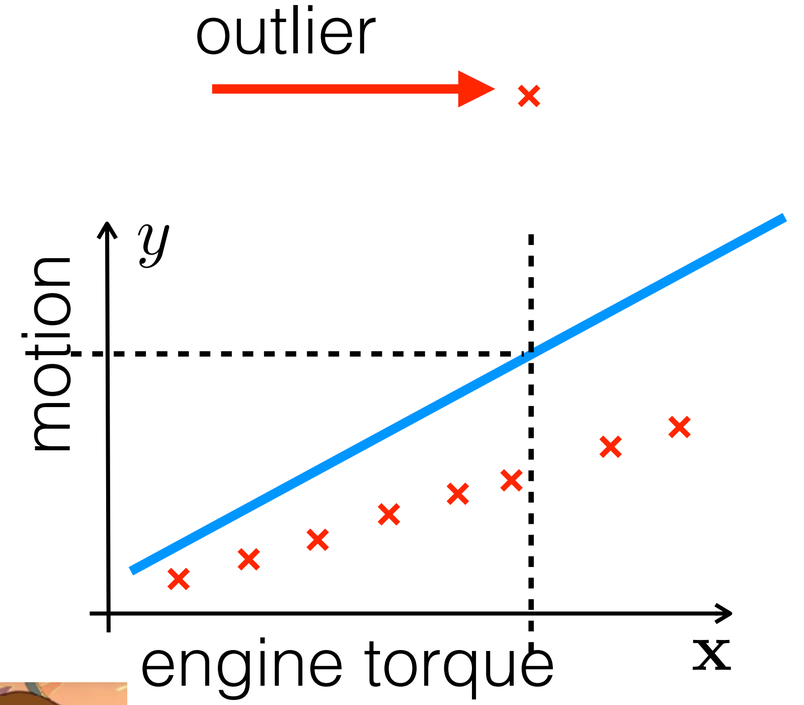
Lec 01: what can go wrong?

What is the joint source of these problems???

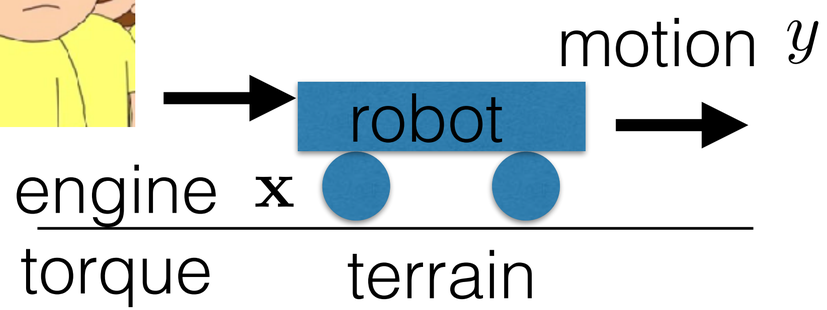








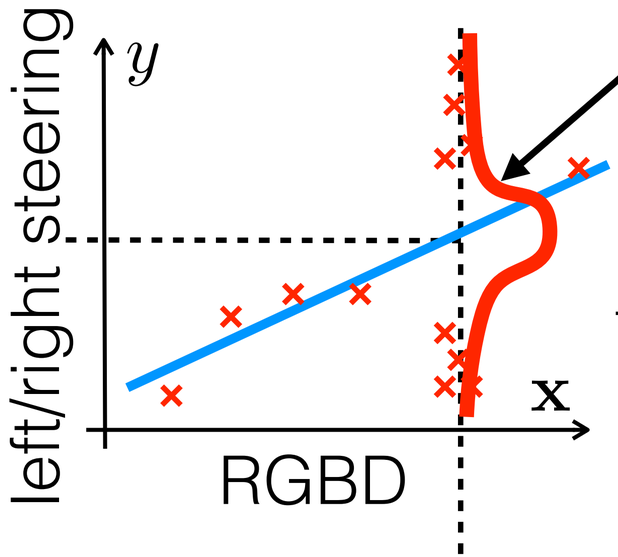
Nature is evil !!!

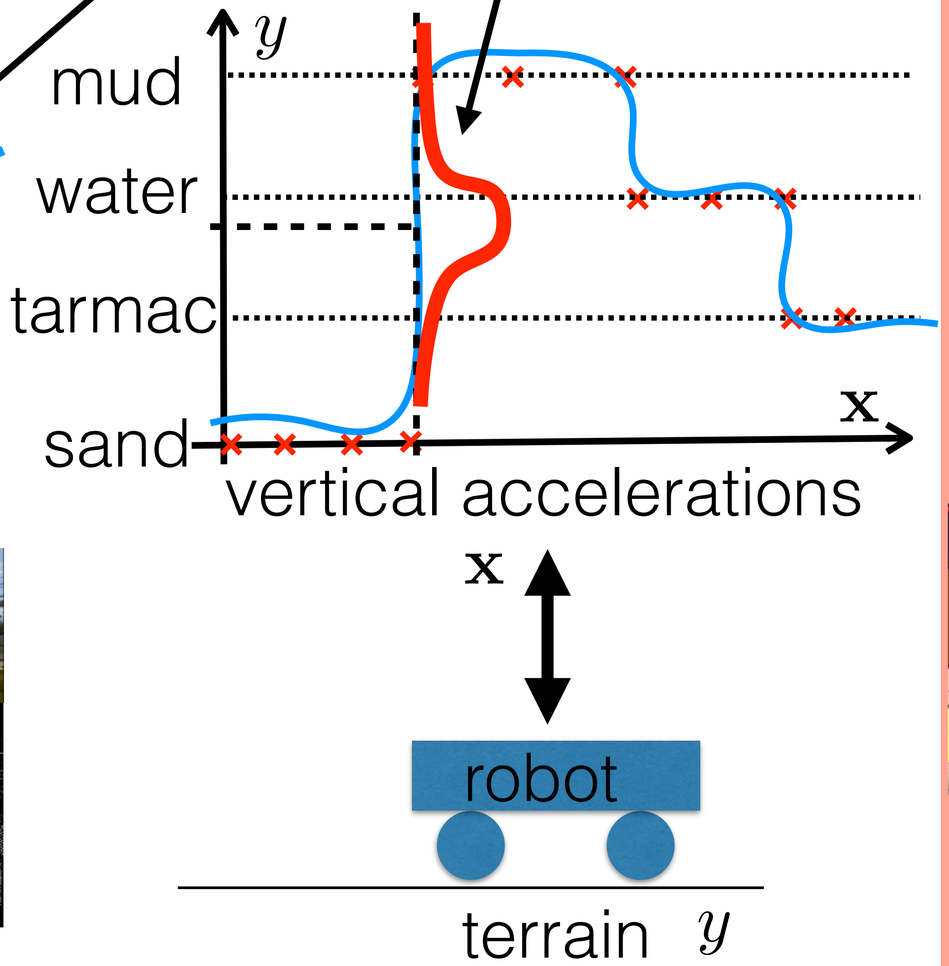


https://rickandmorty.fandom.com/wiki/Evil_Morty

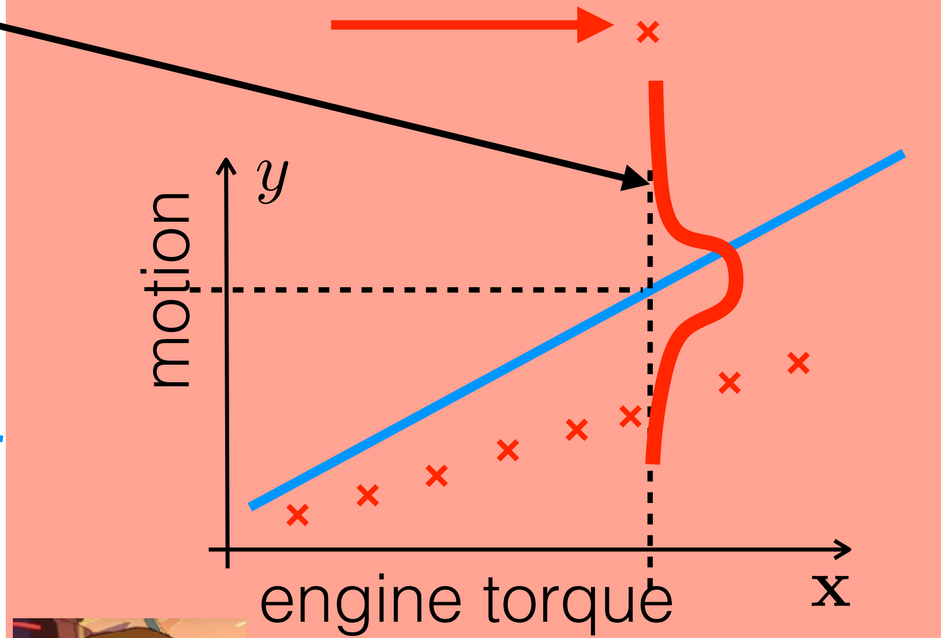
Lec 01: what can go wrong?

This is the problem!

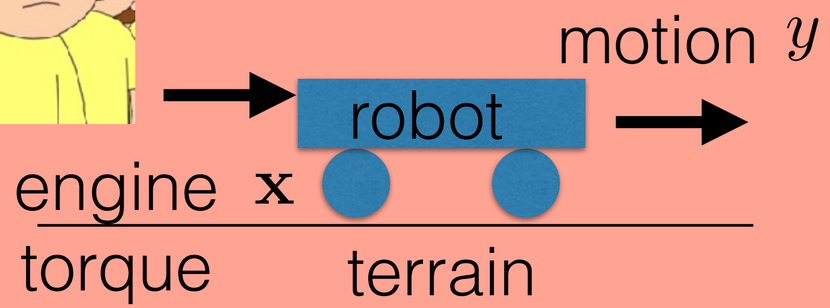




Which distribution should I use instead?

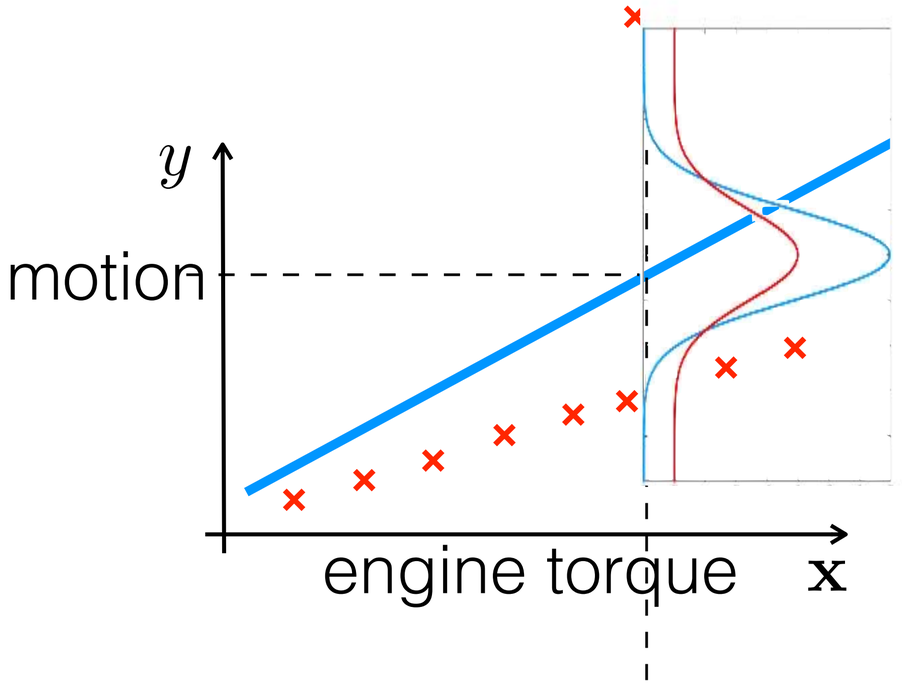


Nature is evil !!!

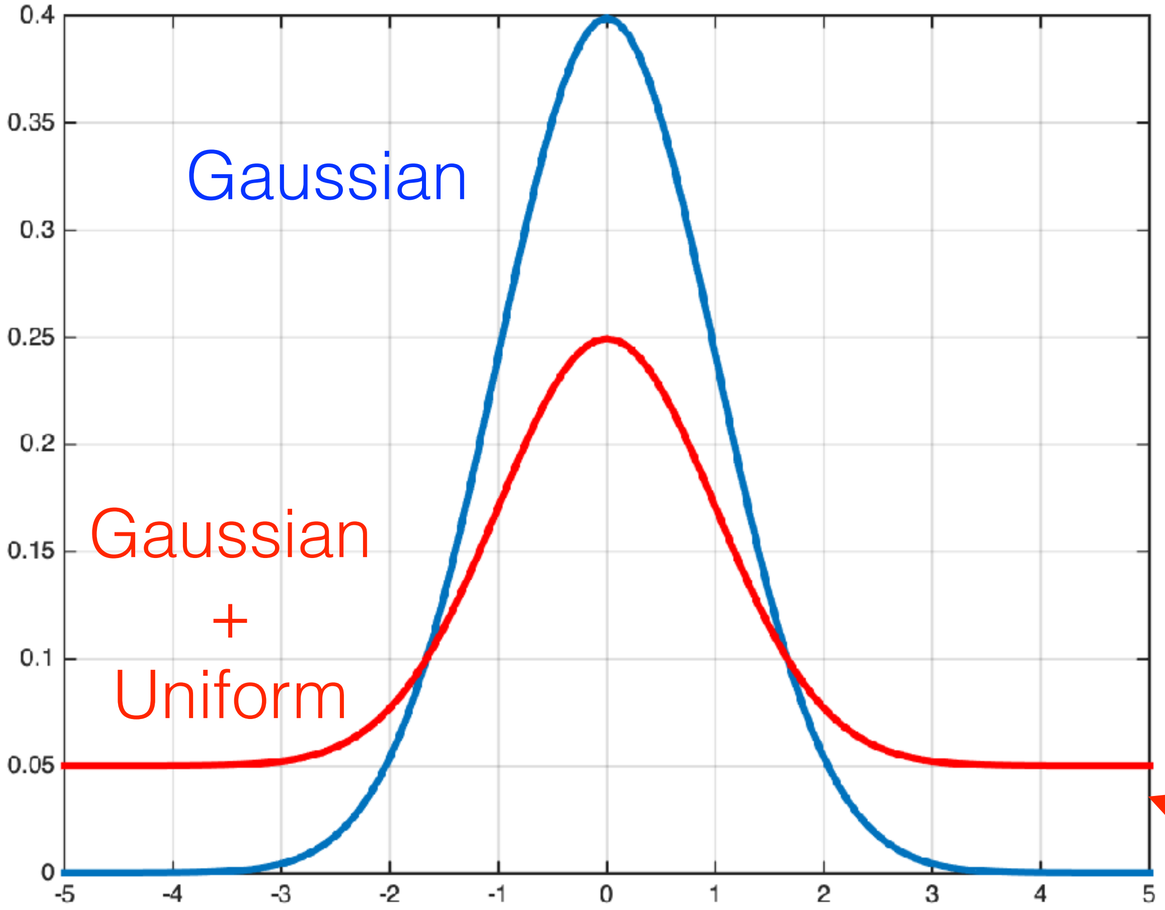


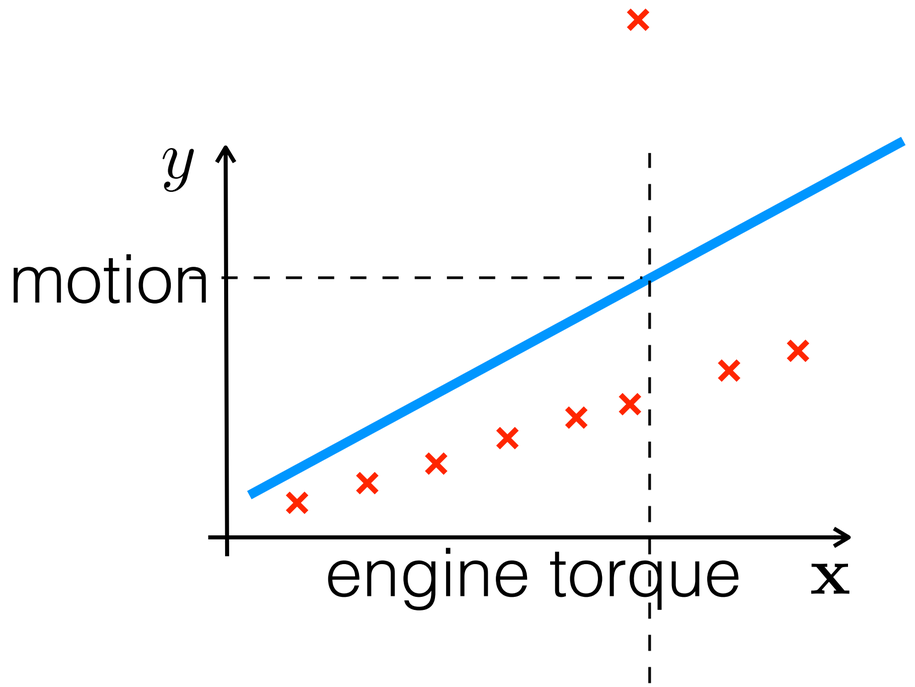
https://rickandmorty.fandom.com/wiki/Evil_Morty

Robust regression



Robust regression



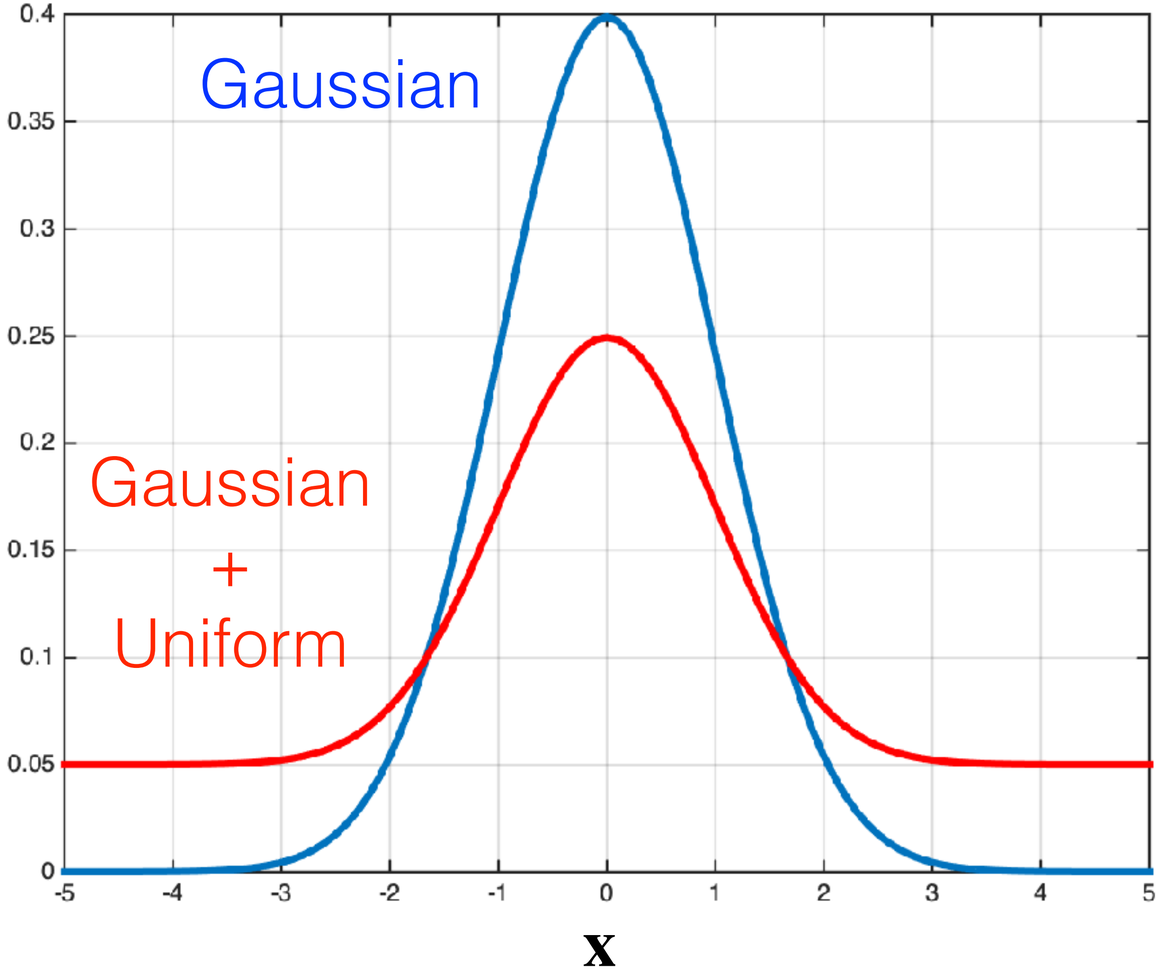


Evil source of uniform noise

Robust regression

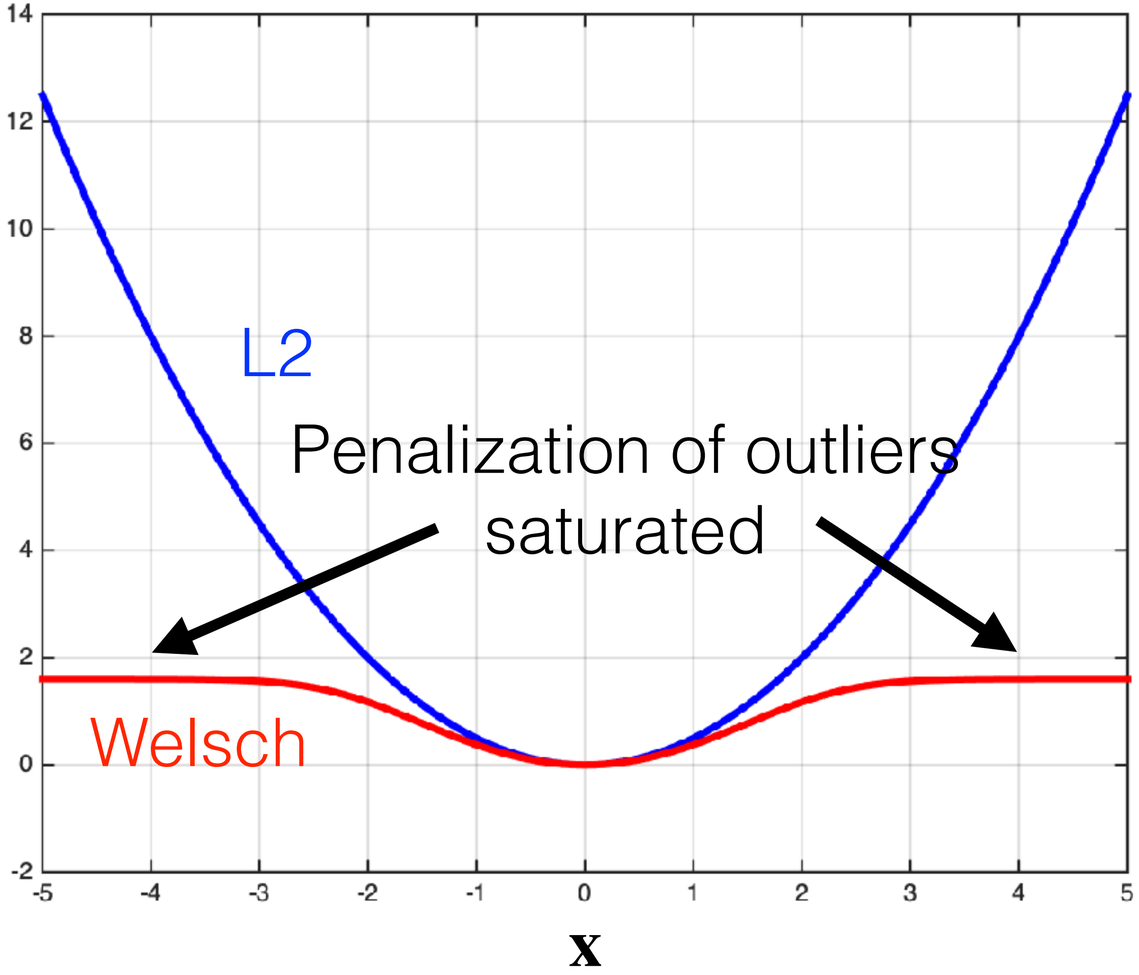
$p(y \mid \mathbf{x}, \mathbf{w})$

Probability distributions



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

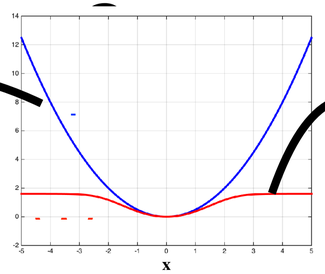
Corresponding losses



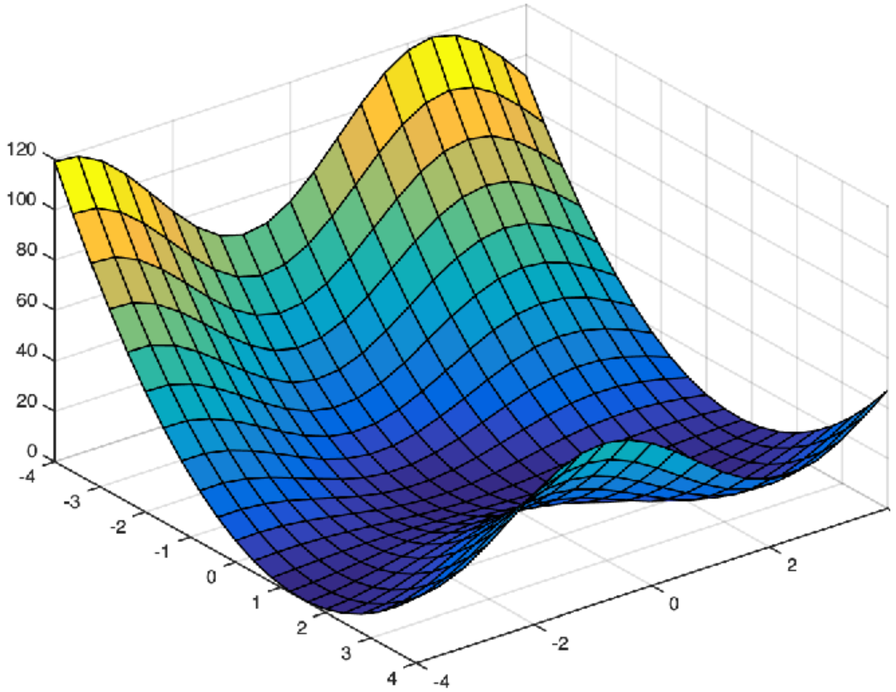
Can you guess where another problem appears?

Robust regression

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

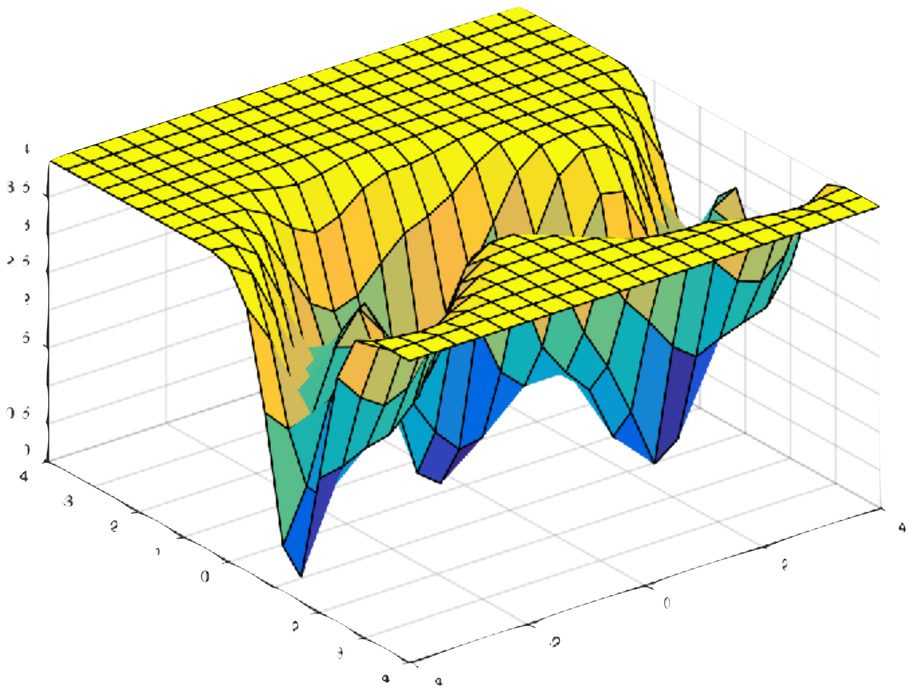


L2 landscape



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

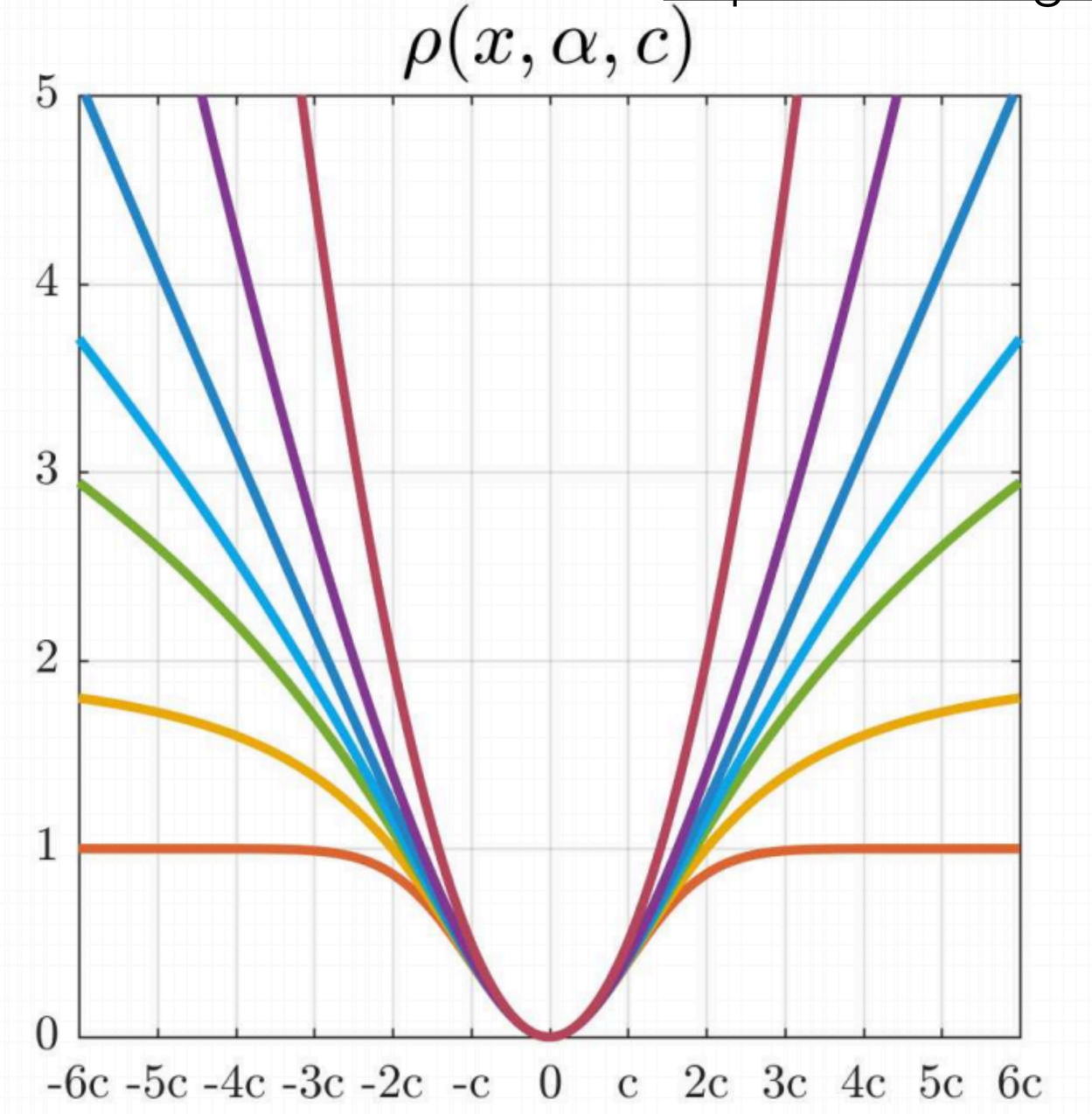
Welsch landscape



- **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



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

Trade-off:

Robustness to uniform noise (outliers)

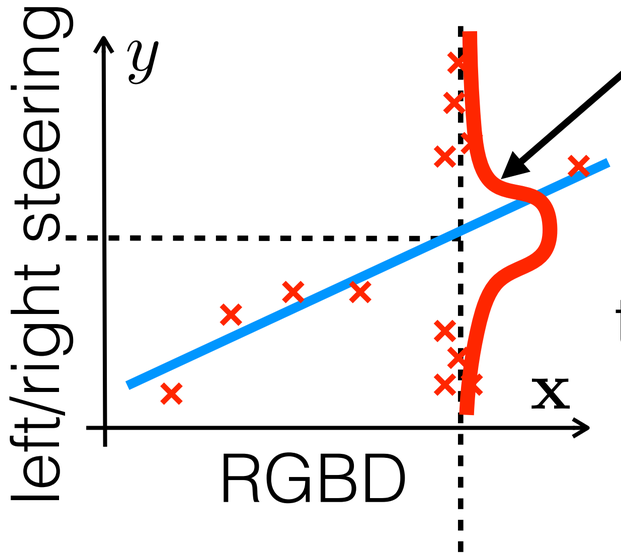
vs

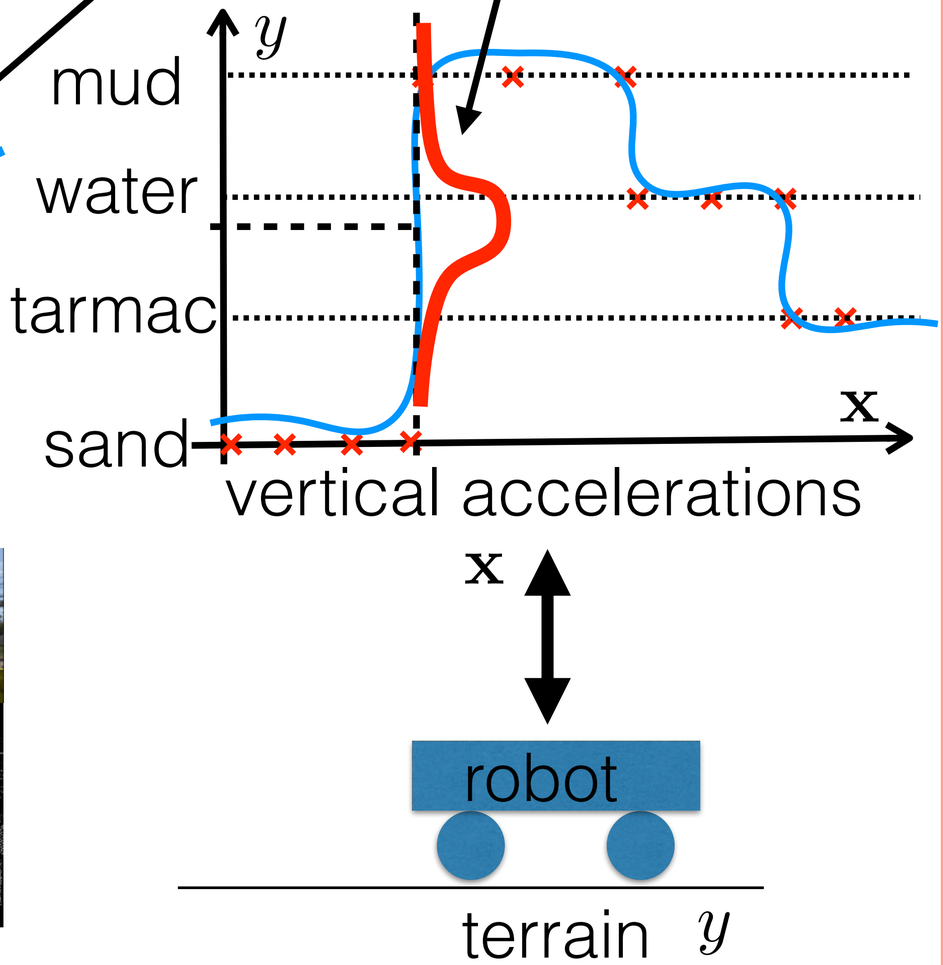
Optimization-friendly landscape

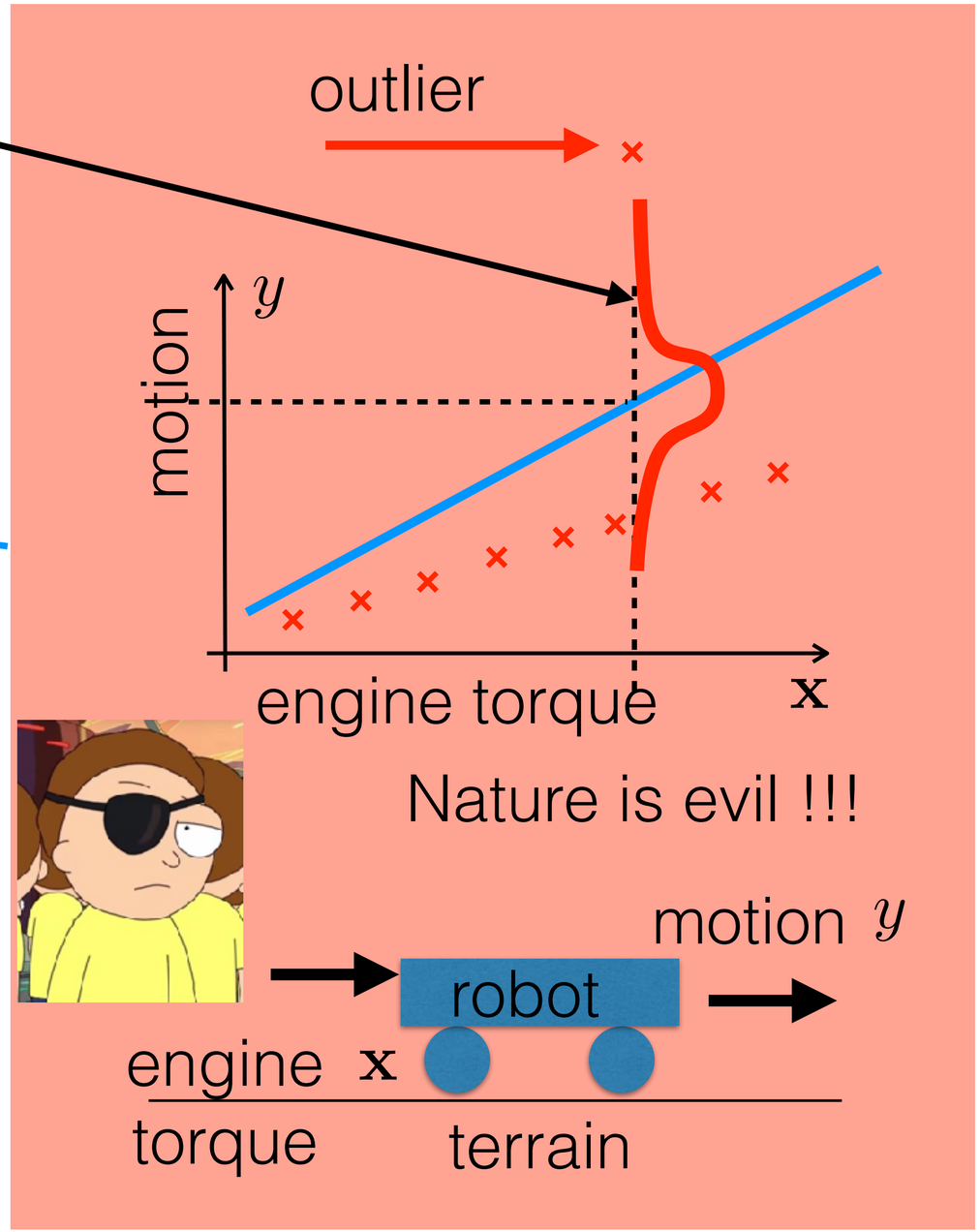
The best what you can do:

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

This is the problem!



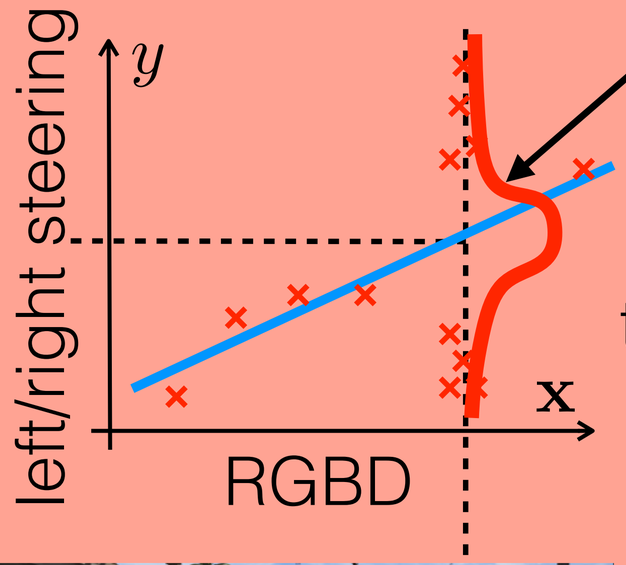


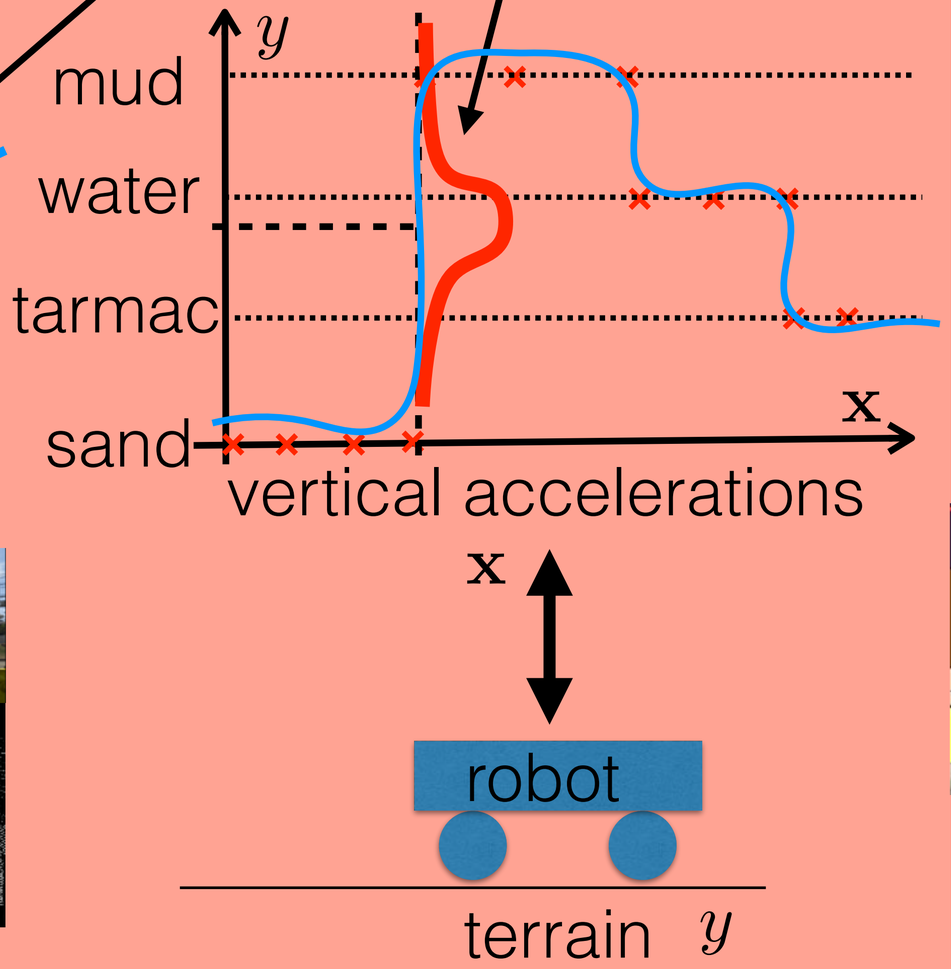


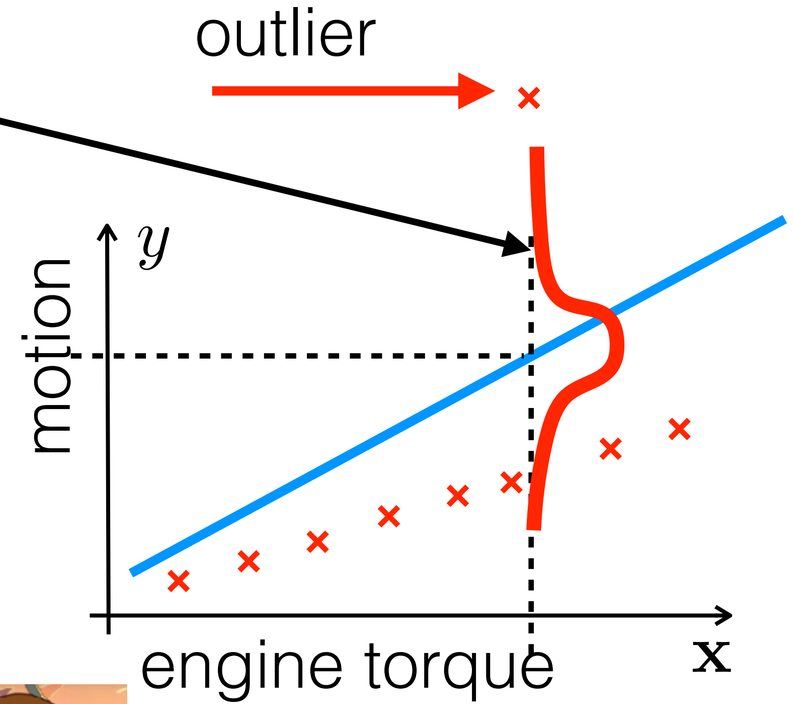
https://rickandmorty.fandom.com/wiki/Evil_Morty

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

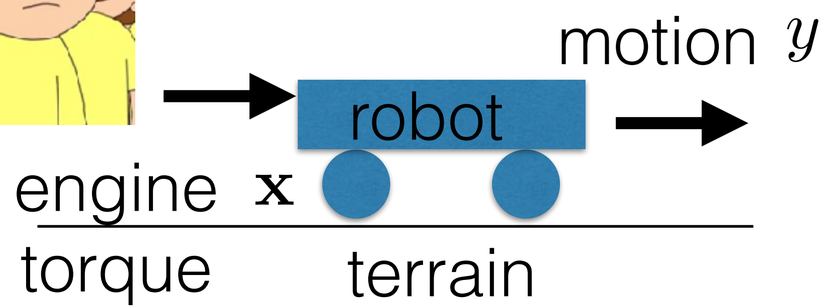
This is the problem!





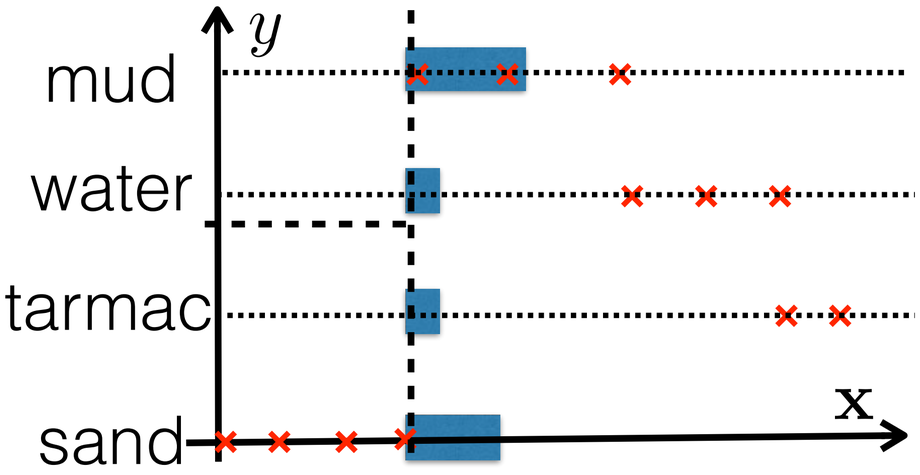


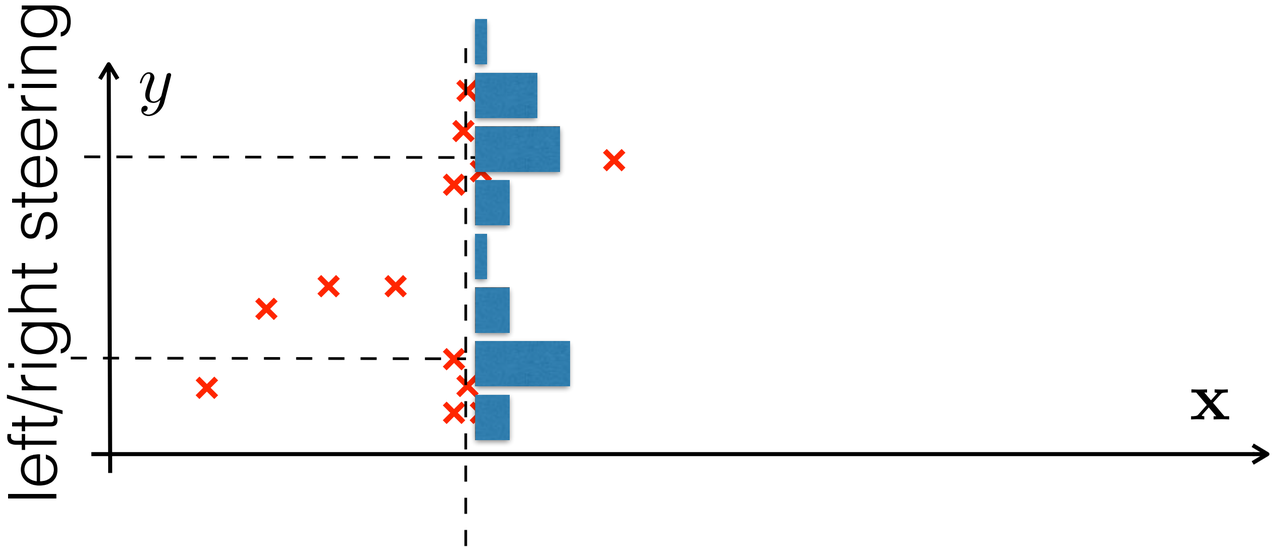
Nature is evil !!!



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

Work-around 1: discretize y-domain and treat the problem as classifcation (for **low**-dimensional outputs)



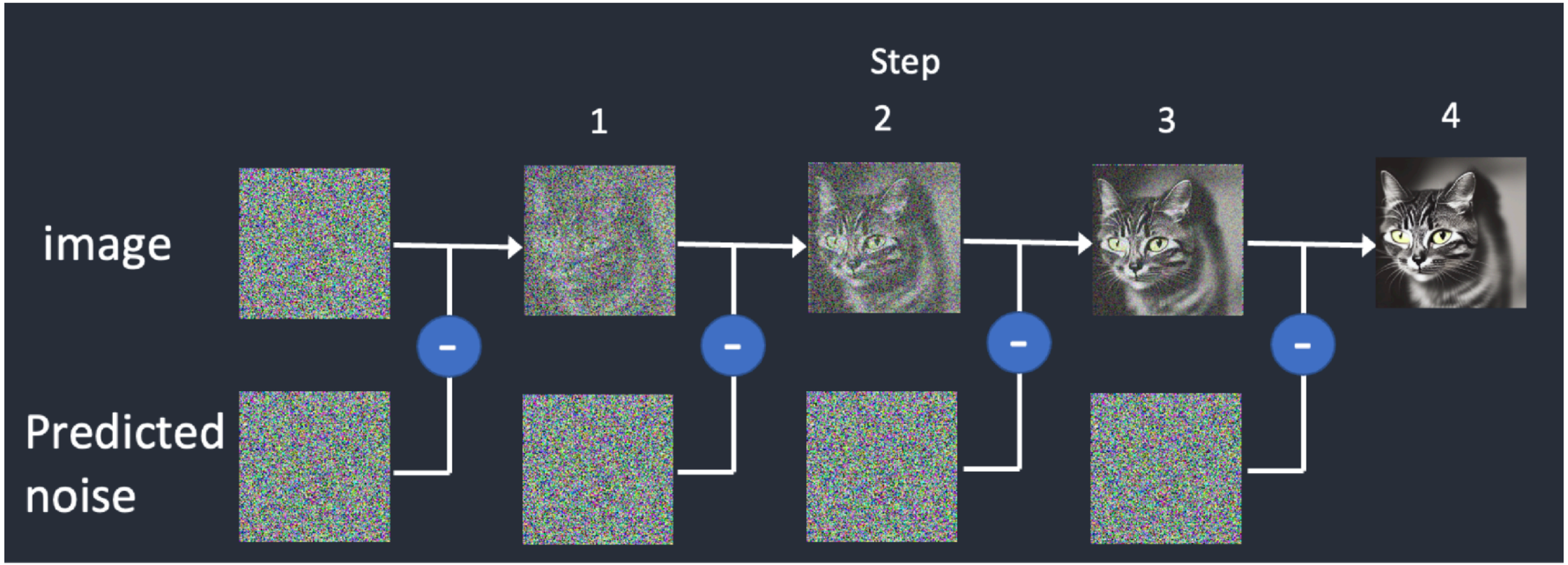


Can I use it always?

Tractable only for low-dim y

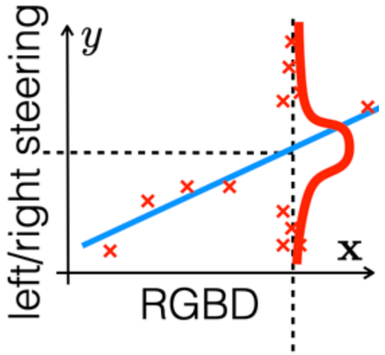
Work-around 2: use generative model

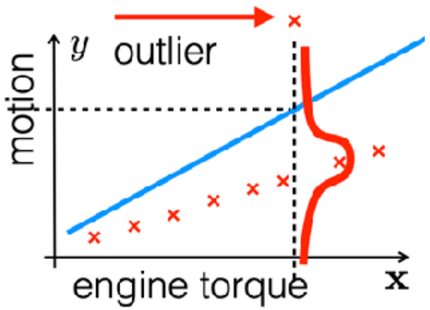
(for **high**-dimensional outputs)

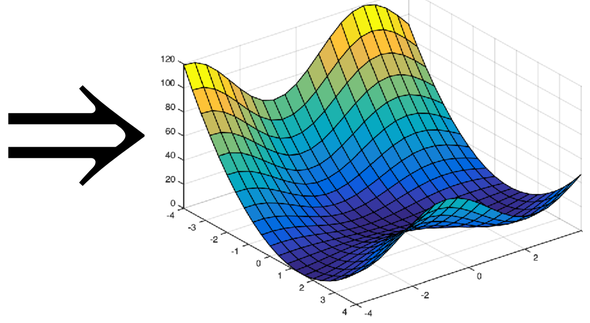


Summary loss

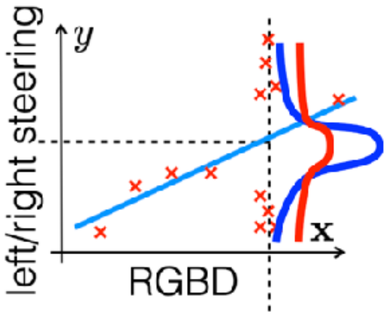
- Maximum **Likelihood** = Minimum **KL-divergence** = Minimum **“-log(p)”-loss**

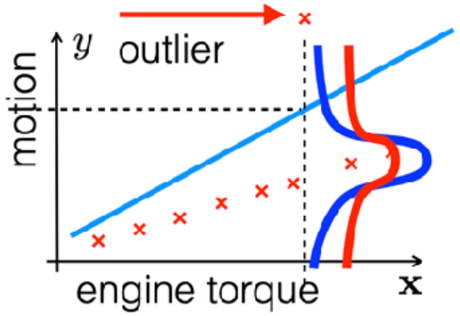


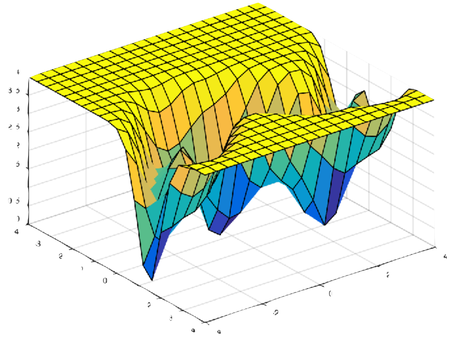




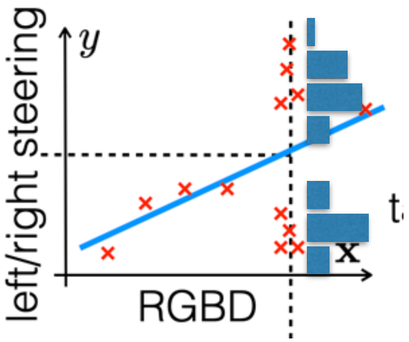
- Assuming **Gaussian** distribution
  - yields **L2 loss**
  - **easier optimization** problem
  - **suffer** from **non-gaussian data**

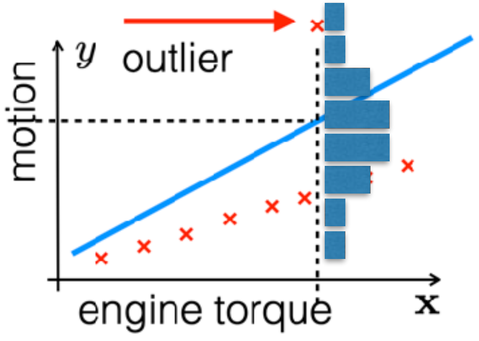






- Assuming **Welsch** distribution
  - yields **robust loss**
  - **hard optimization** problem
  - **handles outliers** better

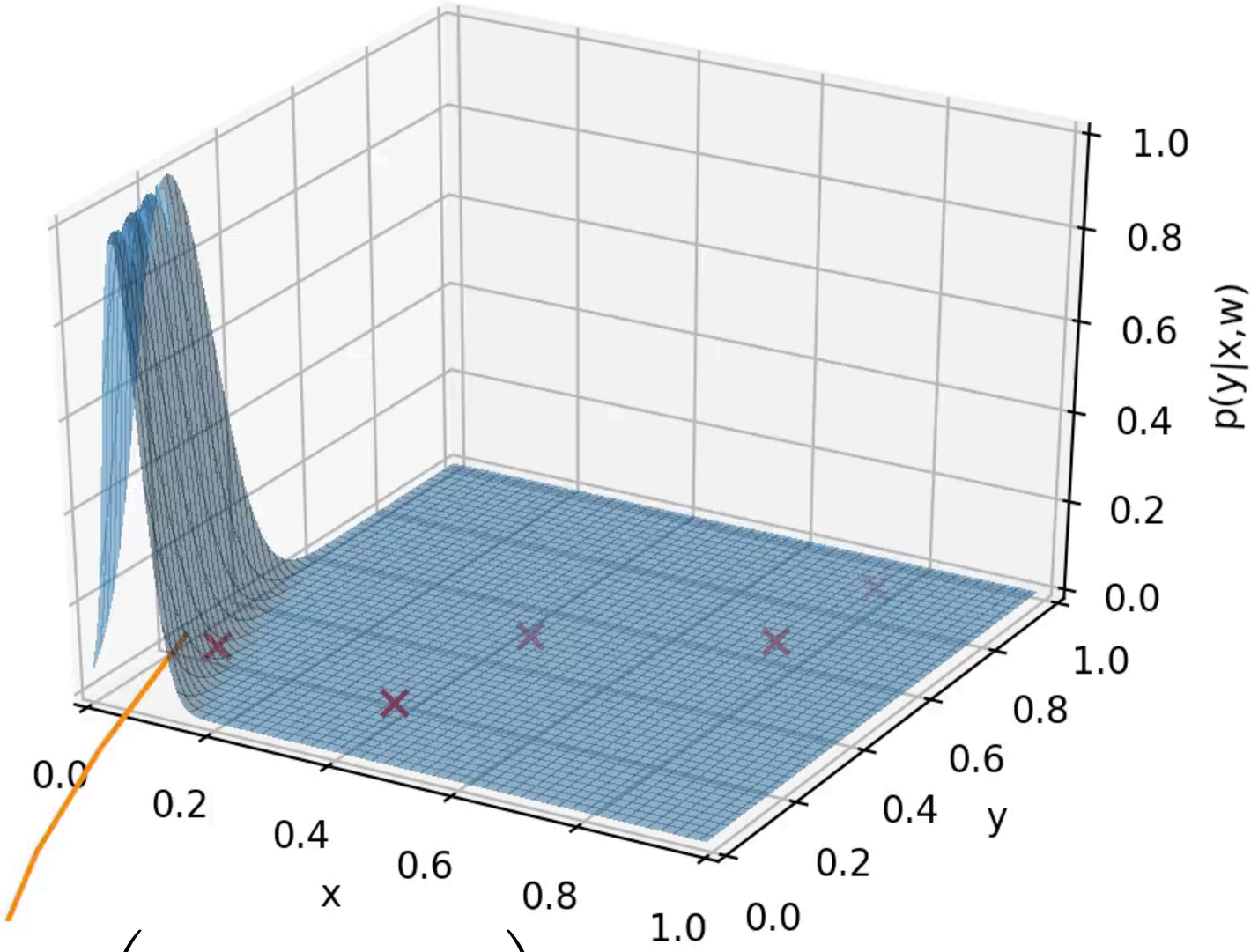




- Assuming **discrete** distribution
  - yields **cross-entropy** loss
  - suffers from **curse-of-dimensionality / overfits**
  - models **non-parametric** distribution

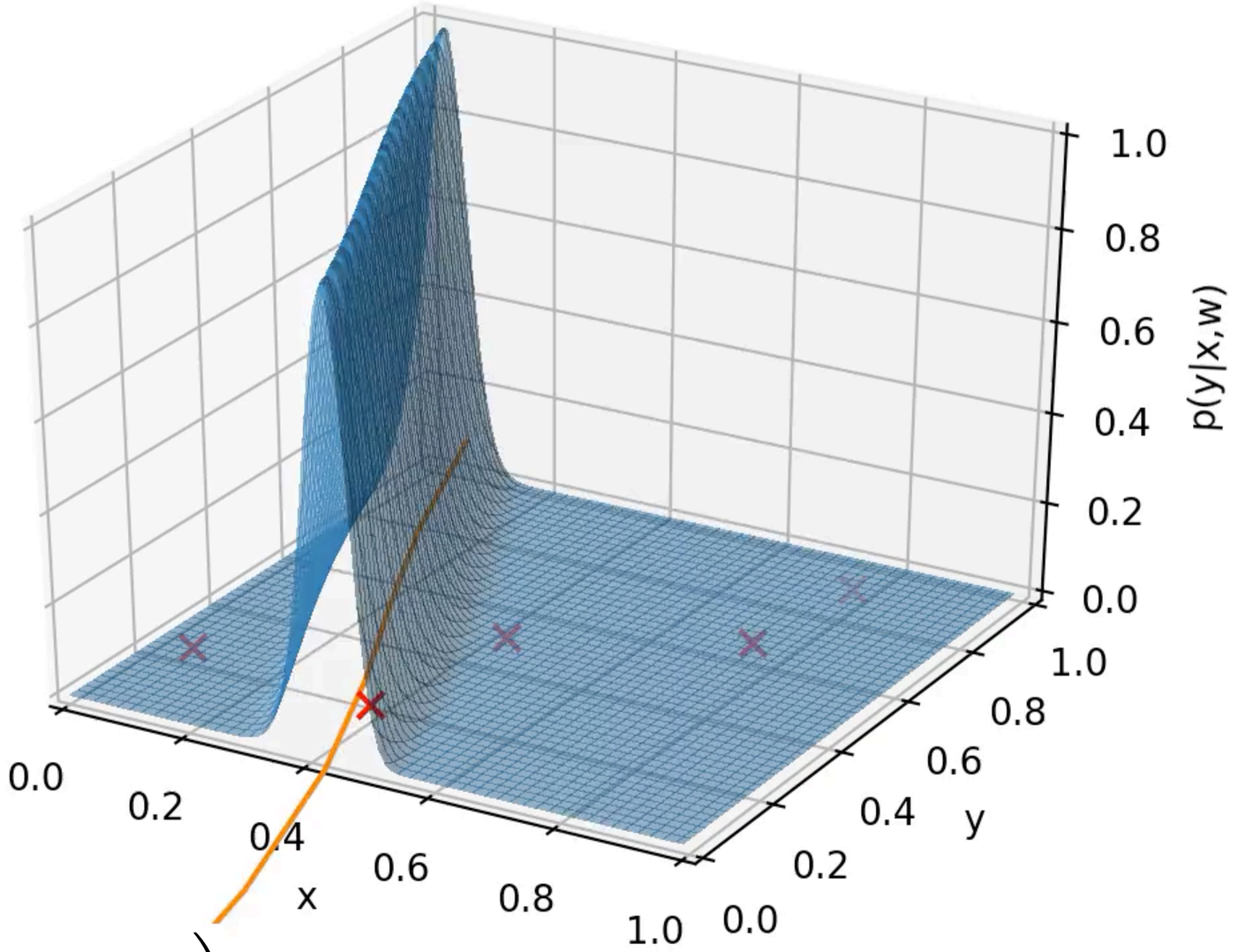
Where does the **overfitting** come from?

Where does the **overfitting** come from?



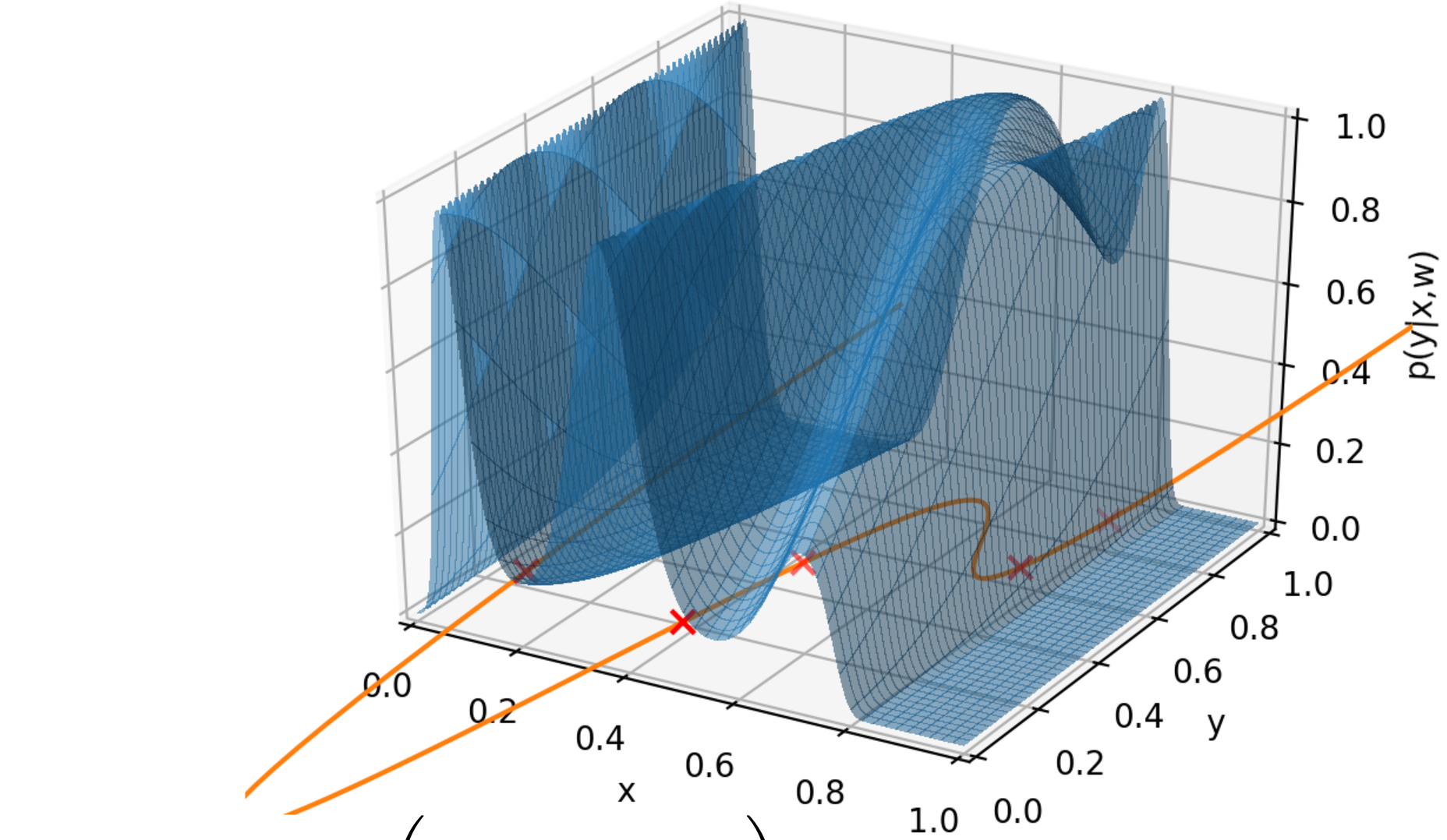
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_2 x_i^2 + w_1 x_i + w_0 - y_i)^2$$

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?



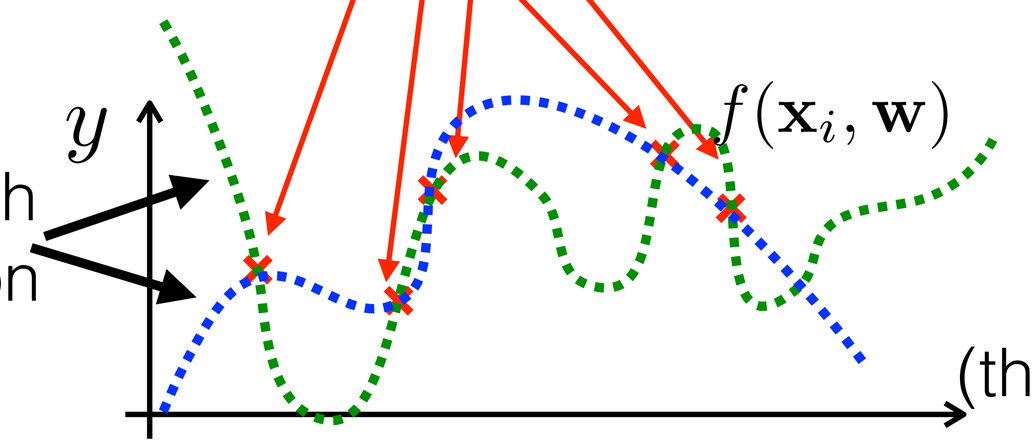
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

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

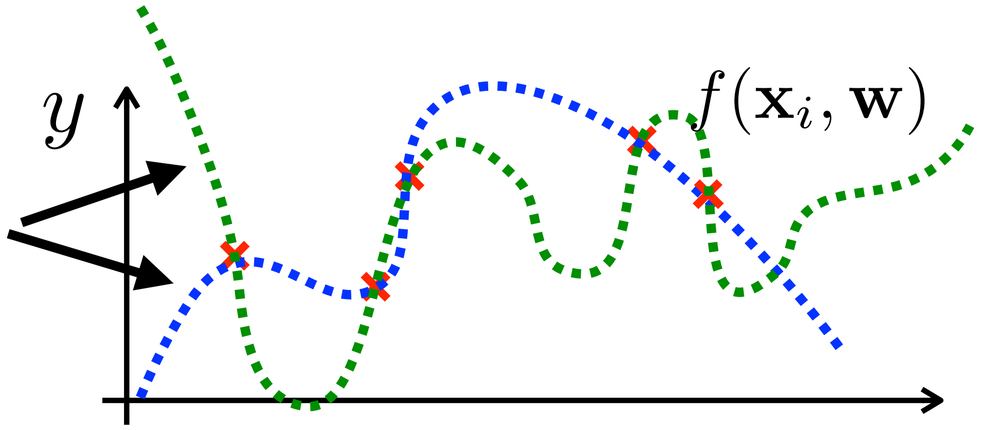
Where does the **overfitting** come from?

Phaistos disc

Unicode

101D0	𐃐	PHAISTOS DISC SIGN PEDESTRIAN
101D1	𐃑	PHAISTOS DISC SIGN PLUMED HEAD
101D2	𐃒	PHAISTOS DISC SIGN TATTOOED HEAD
101D3	𐃓	PHAISTOS DISC SIGN CAPTIVE
101D4	𐃔	PHAISTOS DISC SIGN CHILD
101D5	𐃕	PHAISTOS DISC SIGN WOMAN
101D6	𐃖	PHAISTOS DISC SIGN HELMET
101D7	𐃗	PHAISTOS DISC SIGN GAUNTLET
101D8	𐃘	PHAISTOS DISC SIGN TIARA
101D9	𐃙	PHAISTOS DISC SIGN ARROW
101DA	𐃚	PHAISTOS DISC SIGN BOW
101DB	𐃛	PHAISTOS DISC SIGN SHIELD

Many stories consistent with sequence of visual symbols



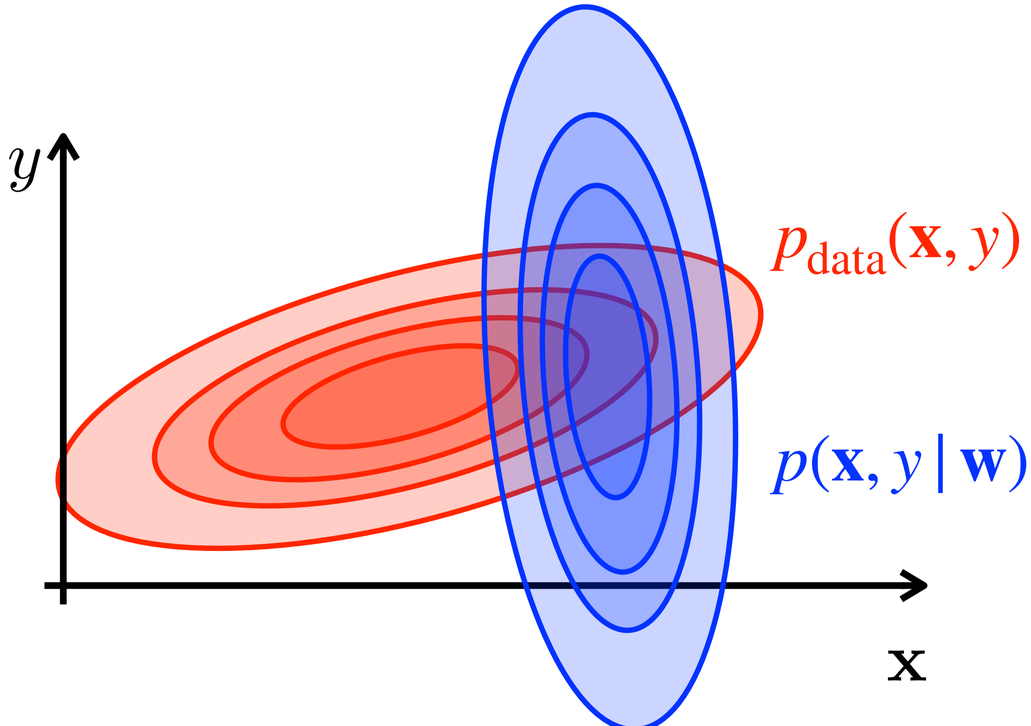
The space of possible stories is too wild

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

Where does the **overfitting** come from?

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

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

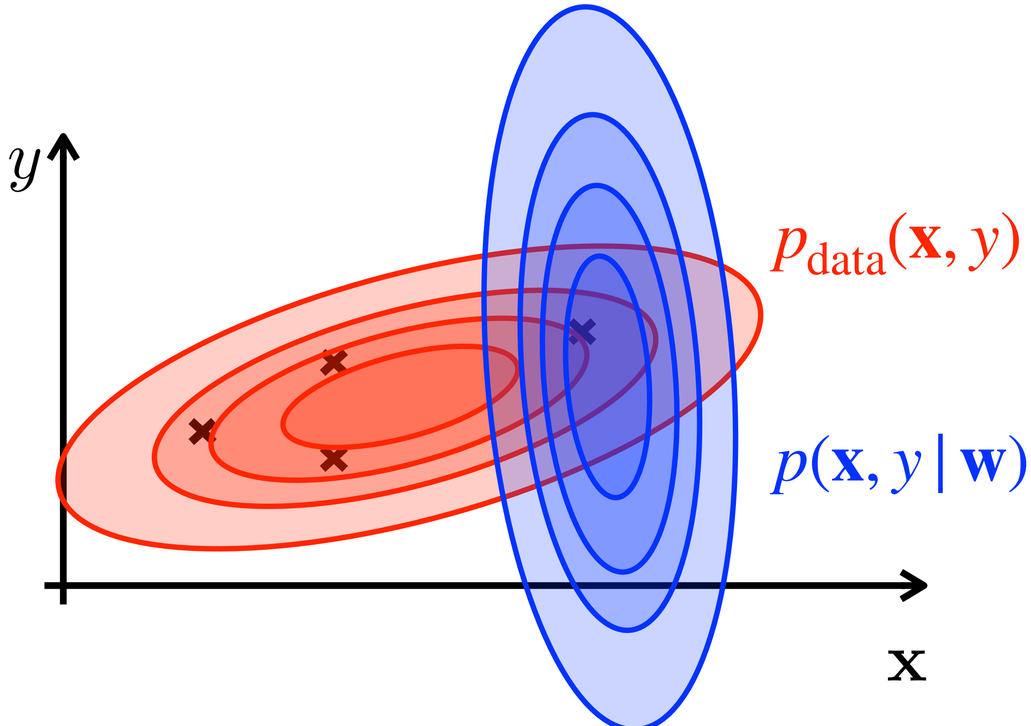


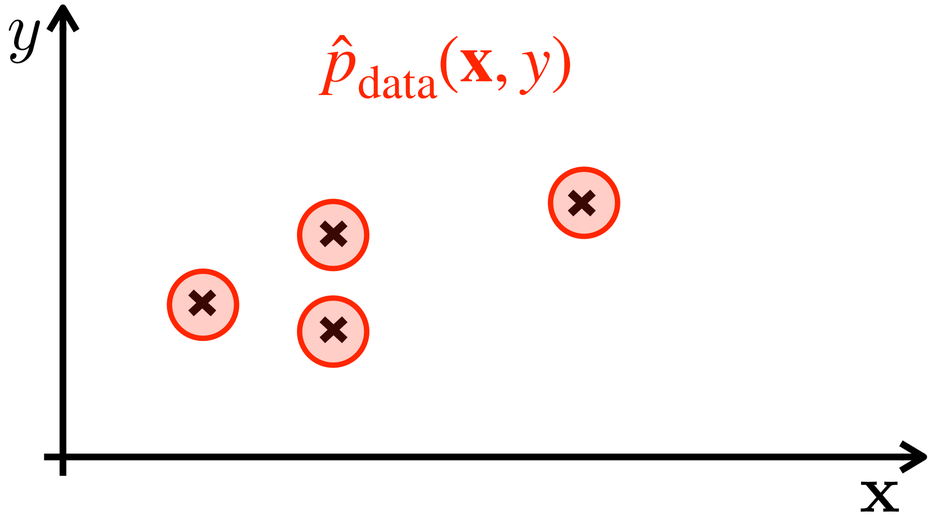
Where does the **overfitting** come from?

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

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

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





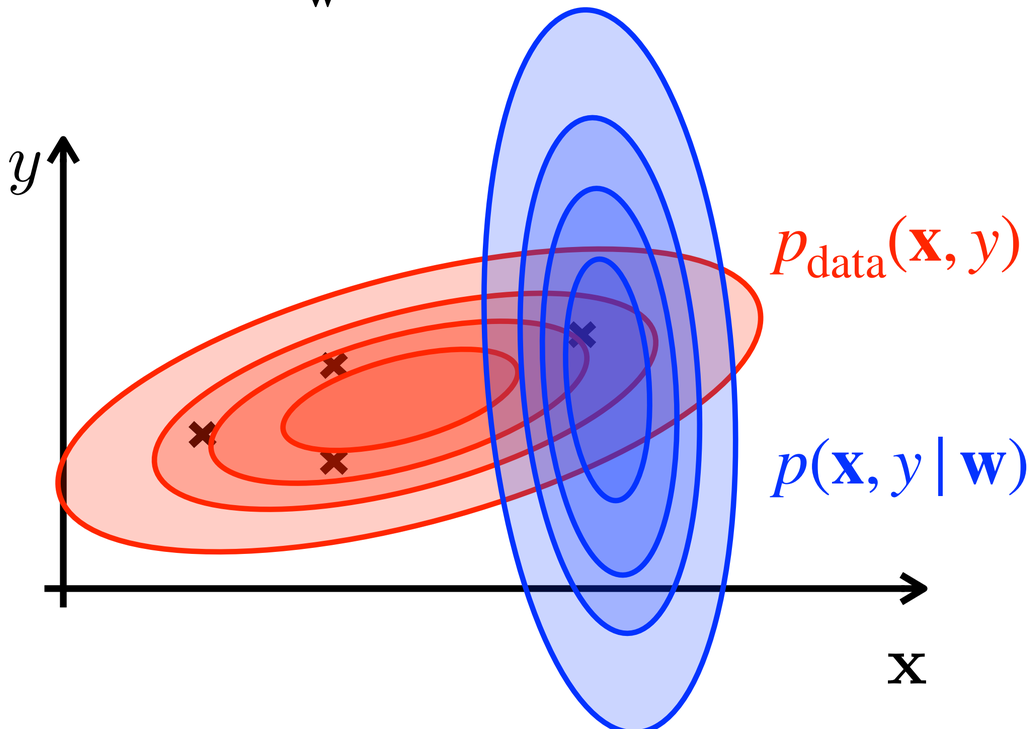
Where does the **overfitting** come from?

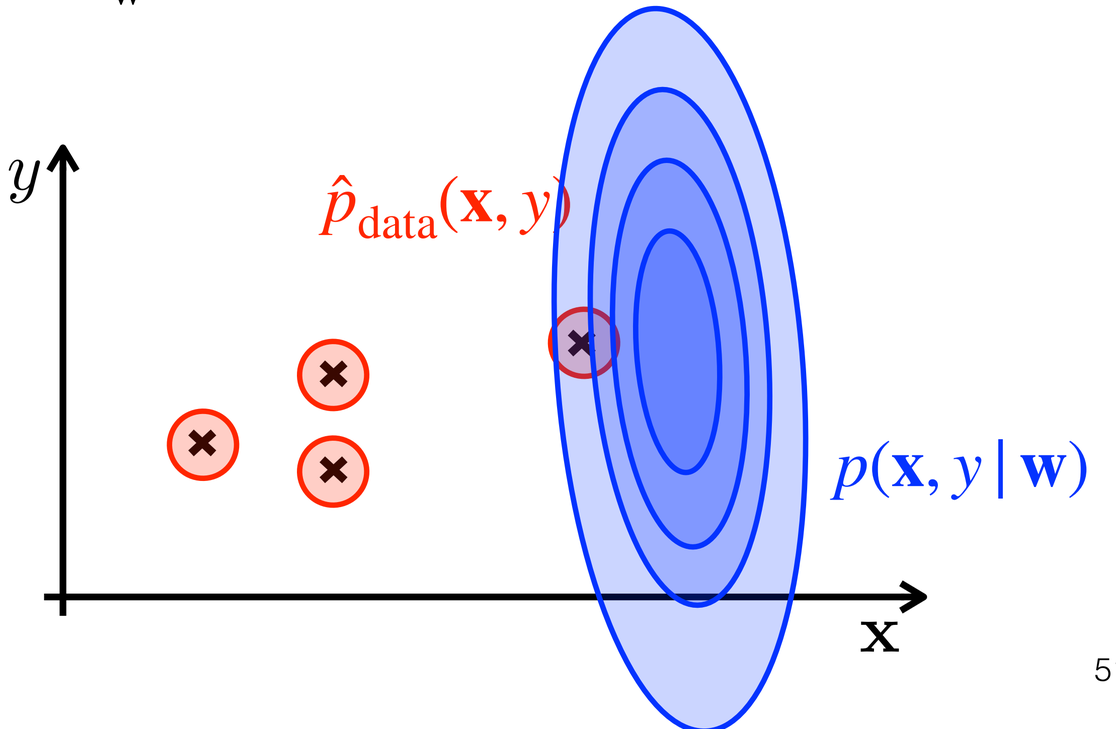
- We fit $p(\mathbf{x}, y \mid \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

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

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

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





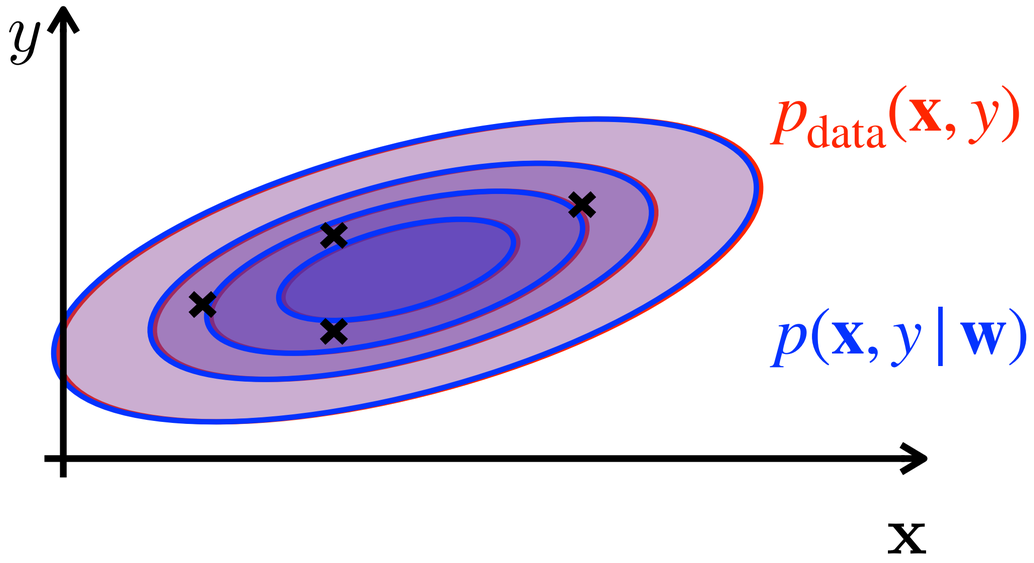
Where does the **overfitting** come from?

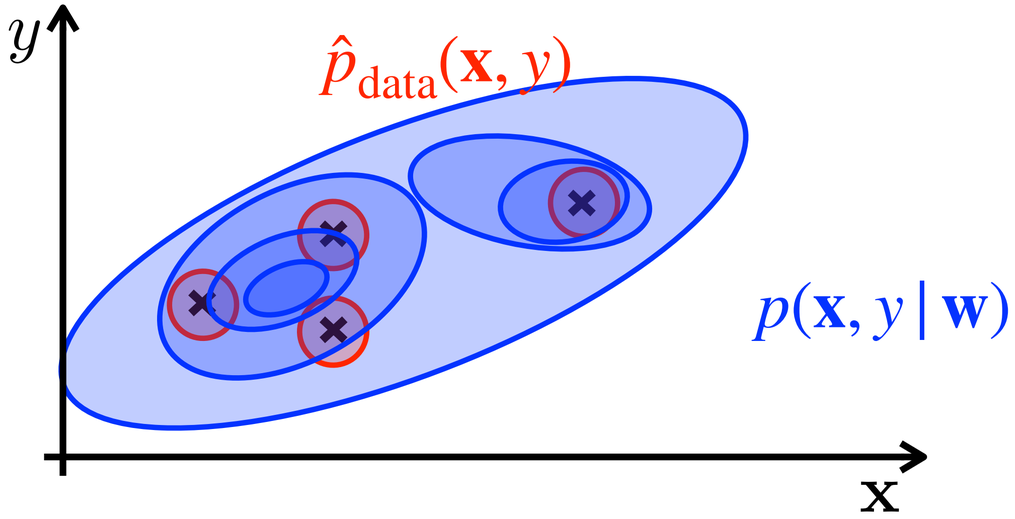
- We fit $p(\mathbf{x}, y \mid \mathbf{w})$ into unknown distribution $p_{\text{data}}(\mathbf{x}, y)$:

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

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

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





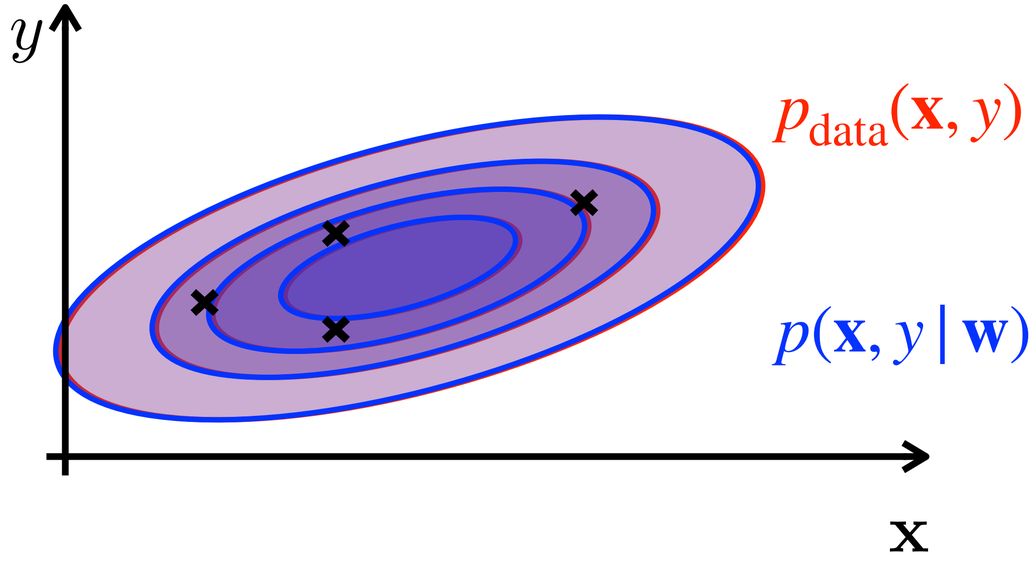
Where does the **overfitting** come from?

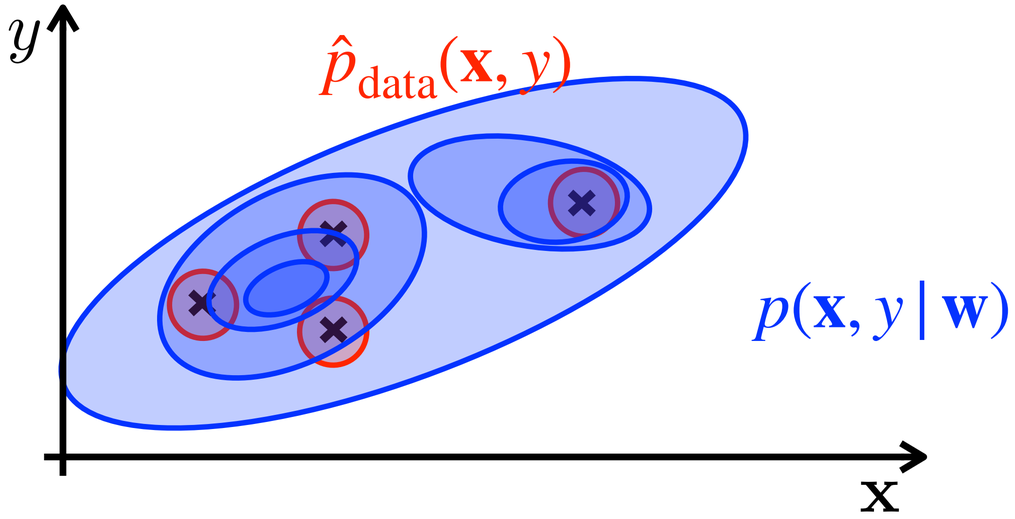
$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y \mid \mathbf{w})\big) \neq \arg\min_{\mathbf{w}} D_{KL}\big(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y \mid \mathbf{w})\big)$$

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





Where does the **overfitting** come from?

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

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

How to suppress overfitting?

How to suppress overfitting?

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

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

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

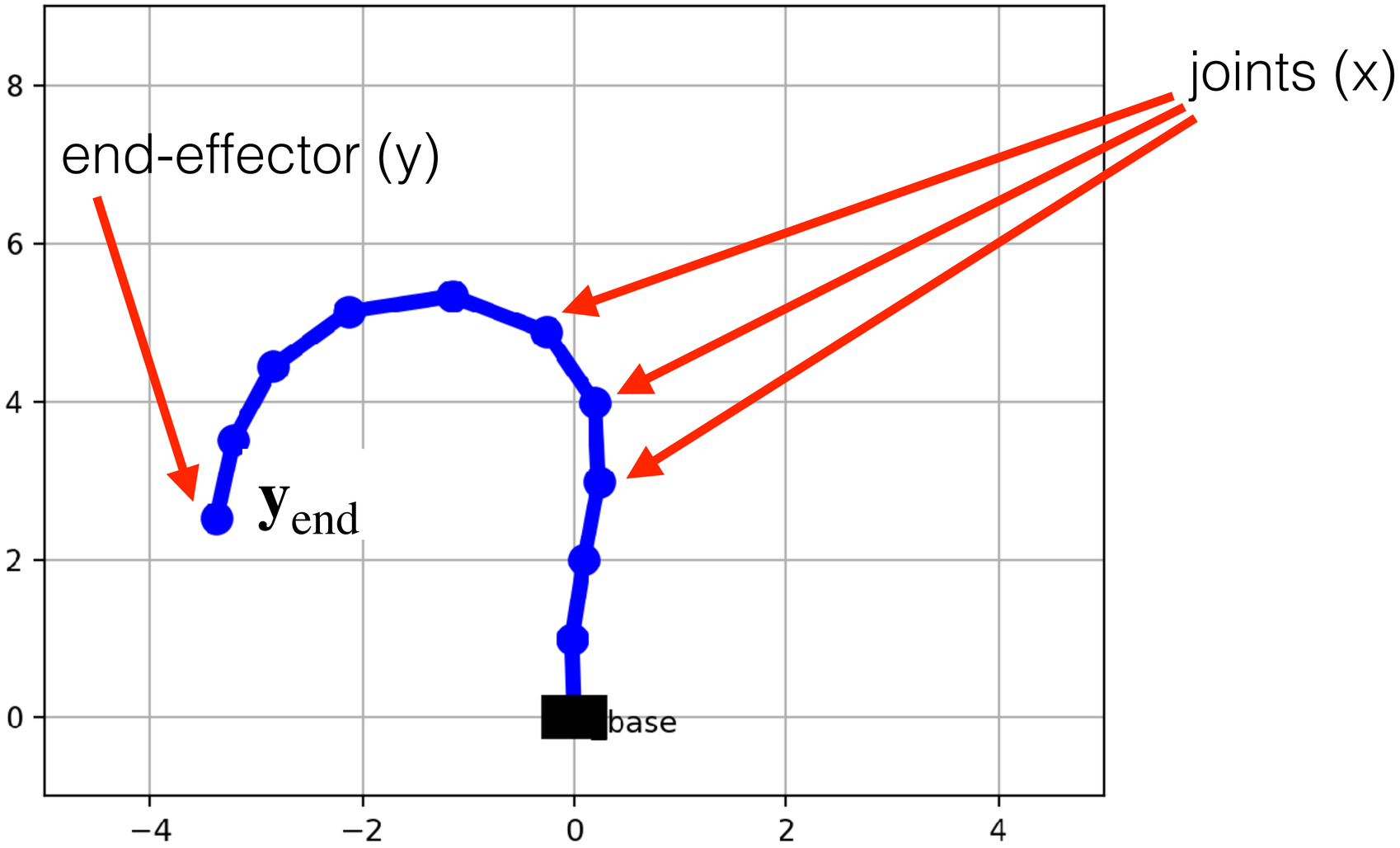
Examples:

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

If you cannot do it, at least penalize wild solutions

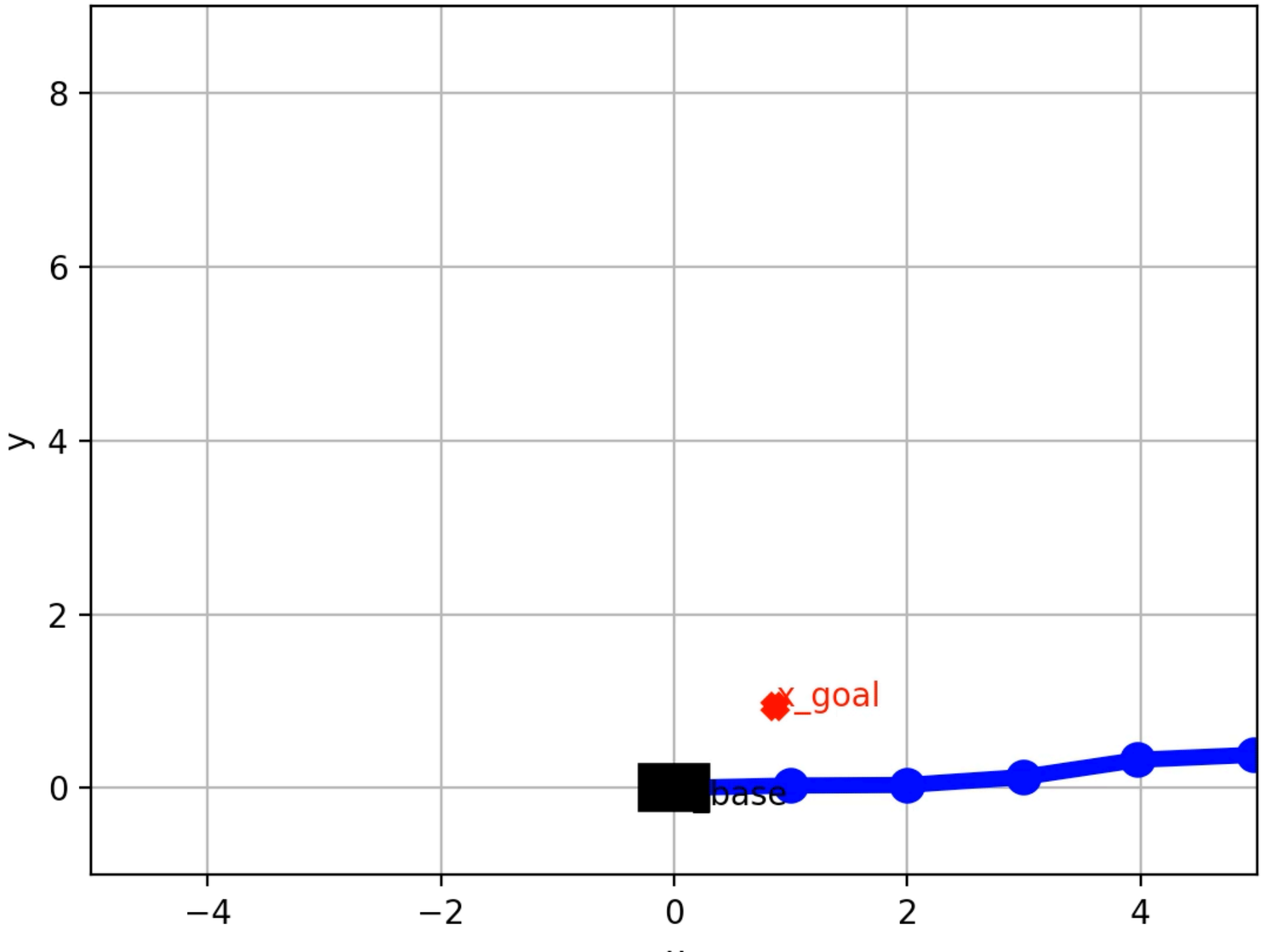
1) Use the right $p(\mathbf{x}, y \mid \mathbf{w})$ that generates only shapes similar to $p_{\text{data}}(\mathbf{x}, y)$

2D manipulator prediction: joints (x) => position of end-effector (y)



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 prediction: joints (x) => position of end-effector (y)



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

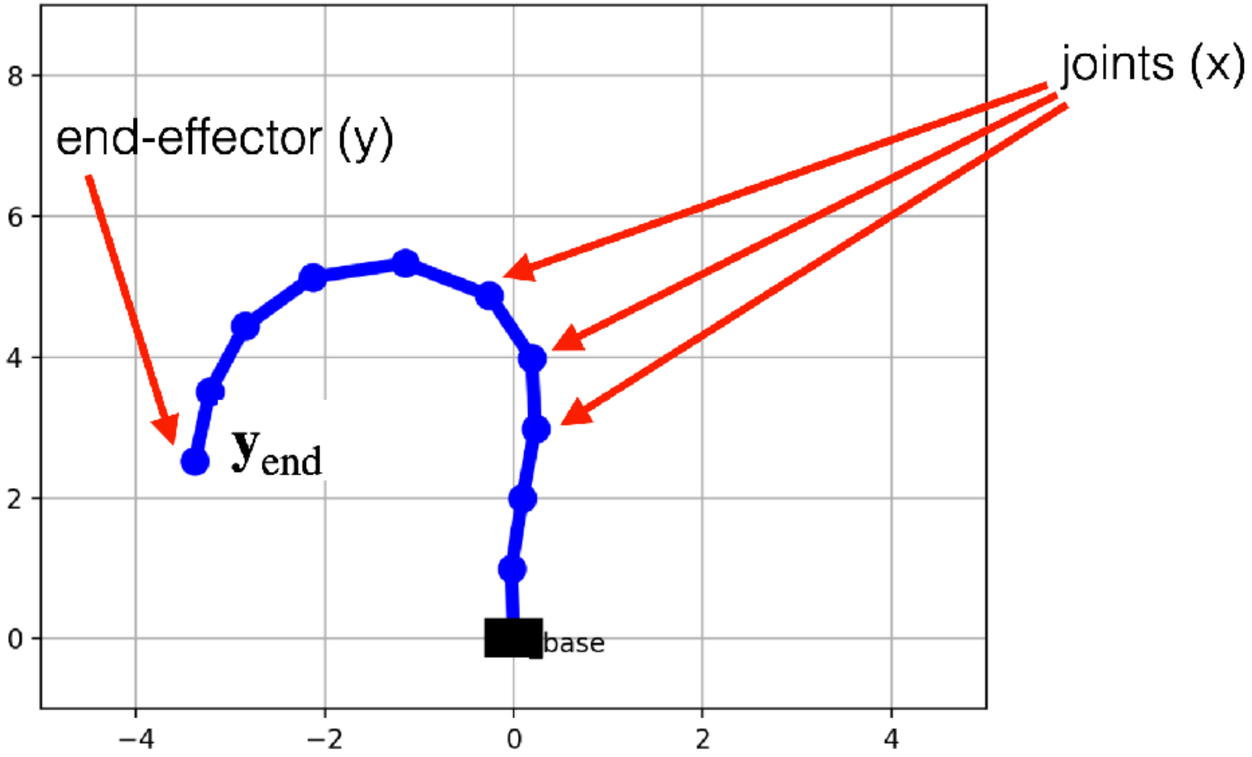
model: $p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(y_{\text{end}};\ f(\mathbf{x}, \mathbf{w}), \sigma)$

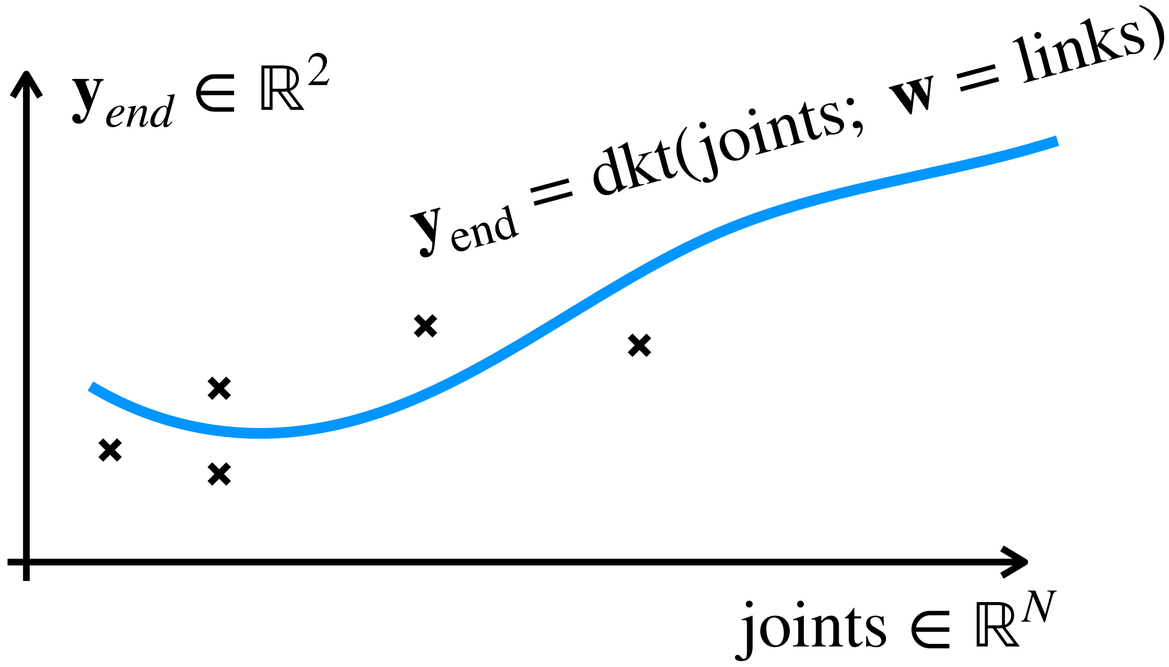
What is the best $f(\mathbf{x}, \mathbf{w})$?

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

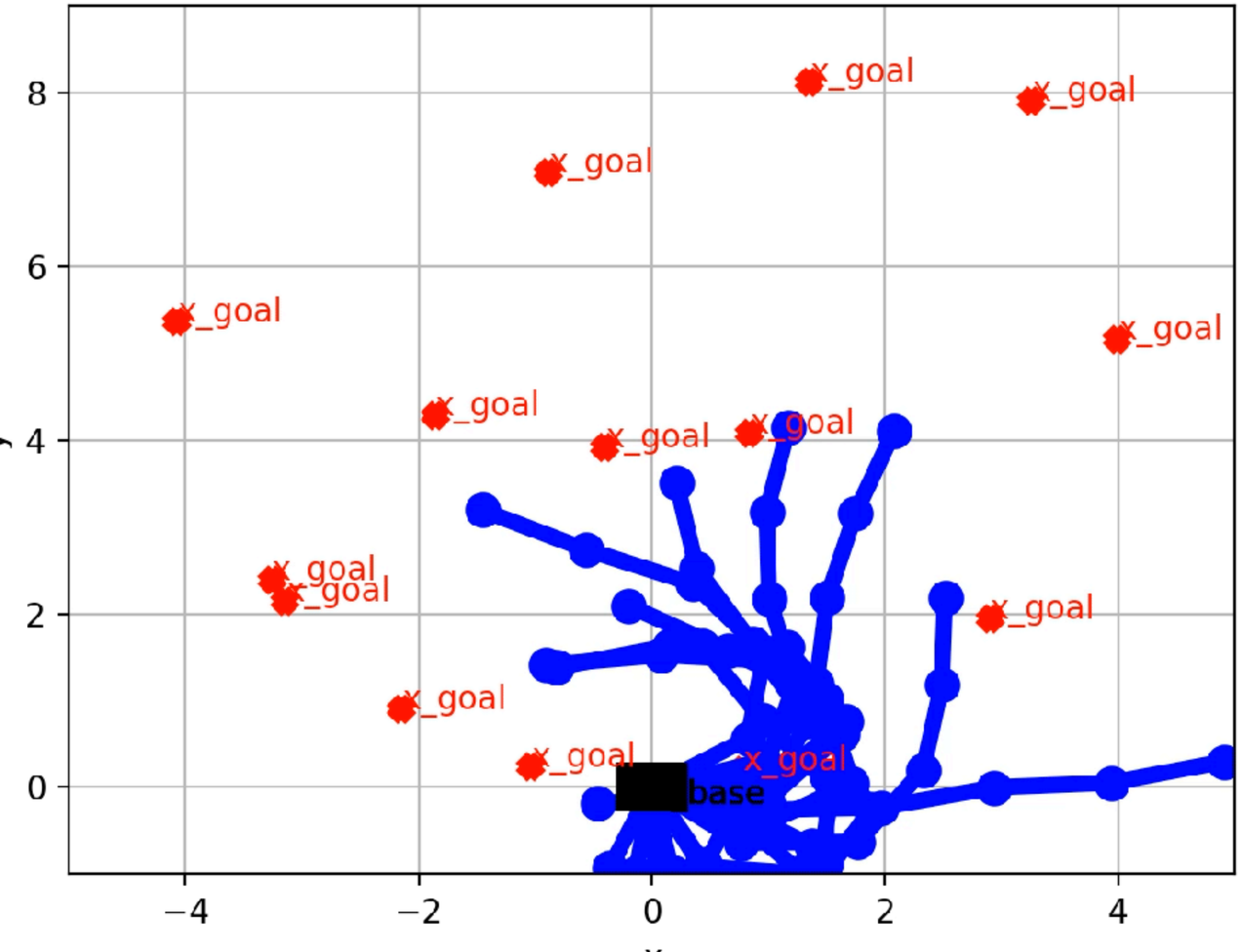
(b) deep ConvNet $\mathbf{y}_{\text{end}} = f(\text{joints}; \mathbf{w})$ = "deep conv net" (overfit)

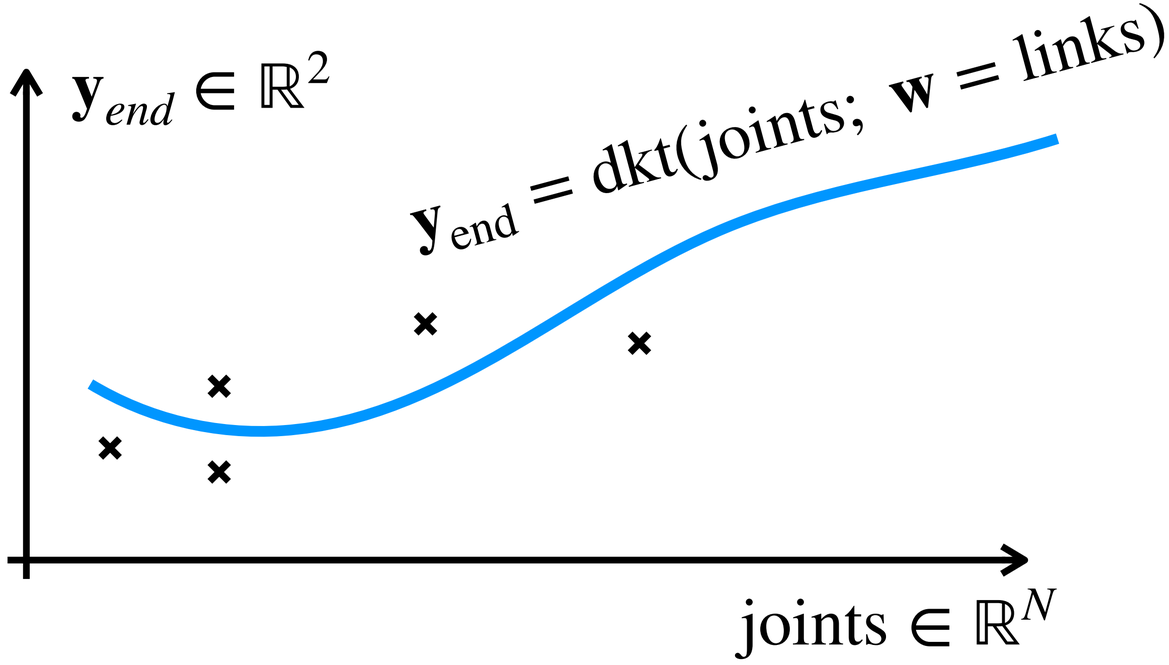
(c) DKT: $\mathbf{y}_{\text{end}} = \text{dkt}(\text{joints};\ \mathbf{w} = \text{links})$ (well-justified model)





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





How to suppress overfitting?

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

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

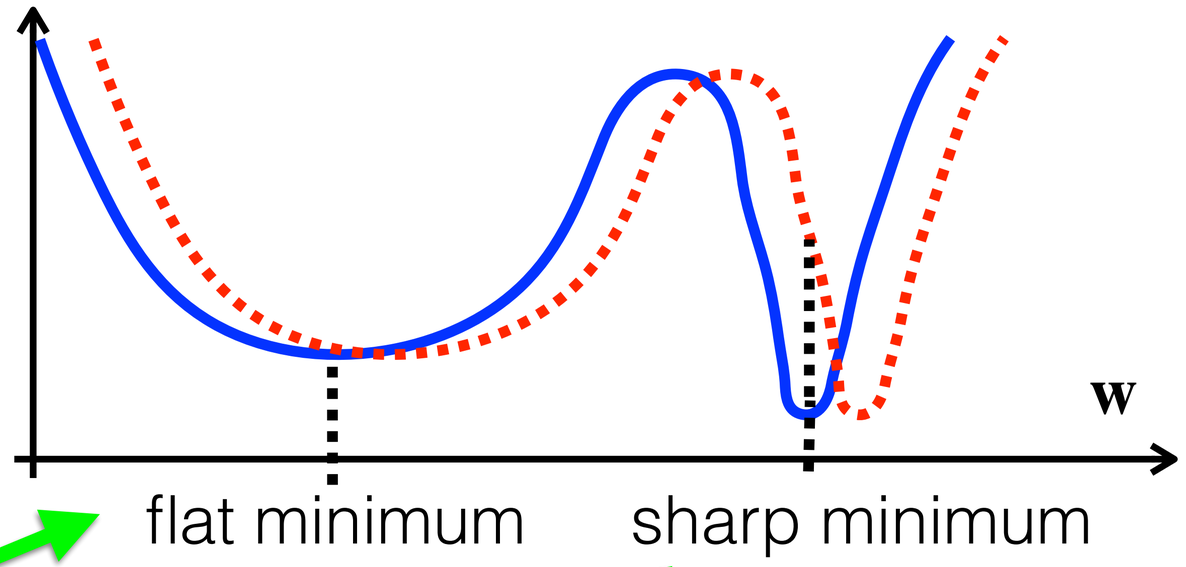
How to suppress overfitting?

Which one is better???

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

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

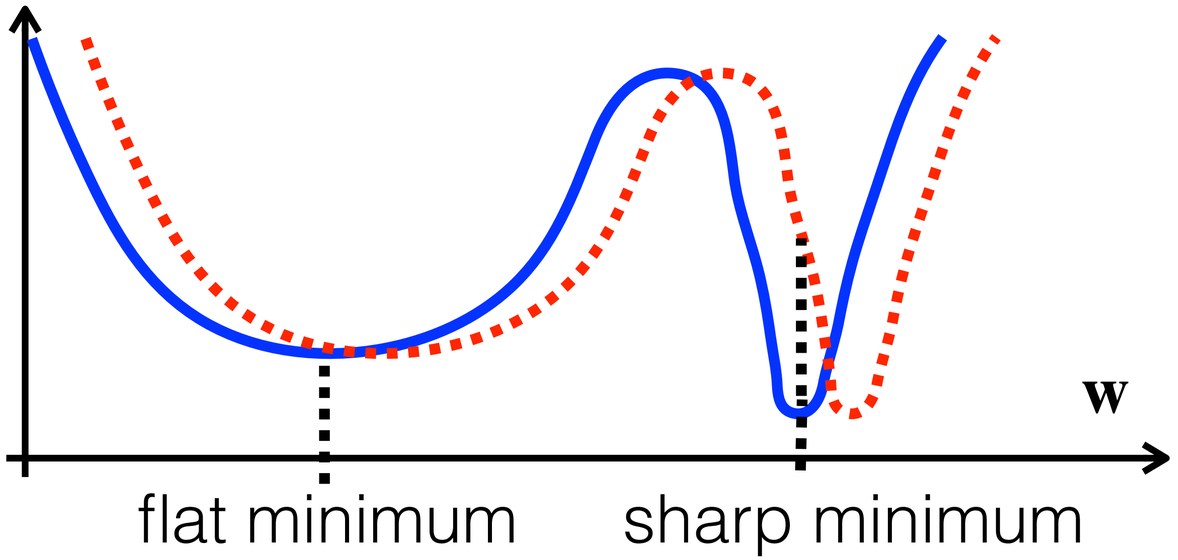
How to suppress overfitting?

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

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

trn data

tst data



One can enforce flat minimum:

By optimization technique: e.g. momentum

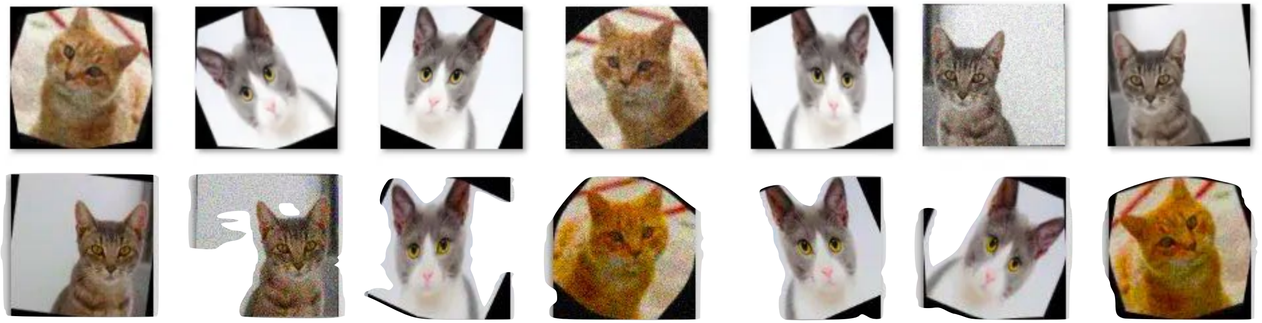
By loss function: $\min_{w} \max_{\|\epsilon\|_2 \leq \rho} L_{train}(w + \epsilon)$

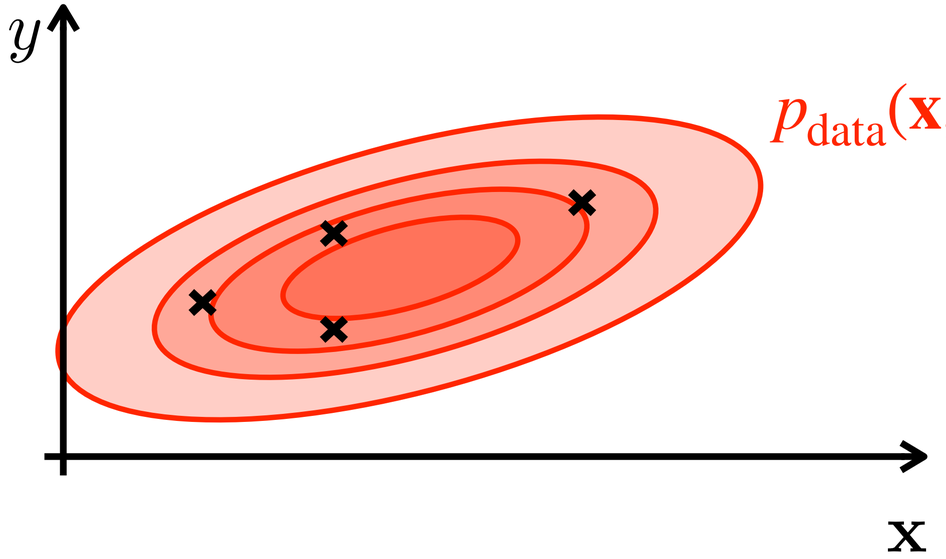
[Foret 2021]
https://arxiv.org/pdf/2106.01548.pdf

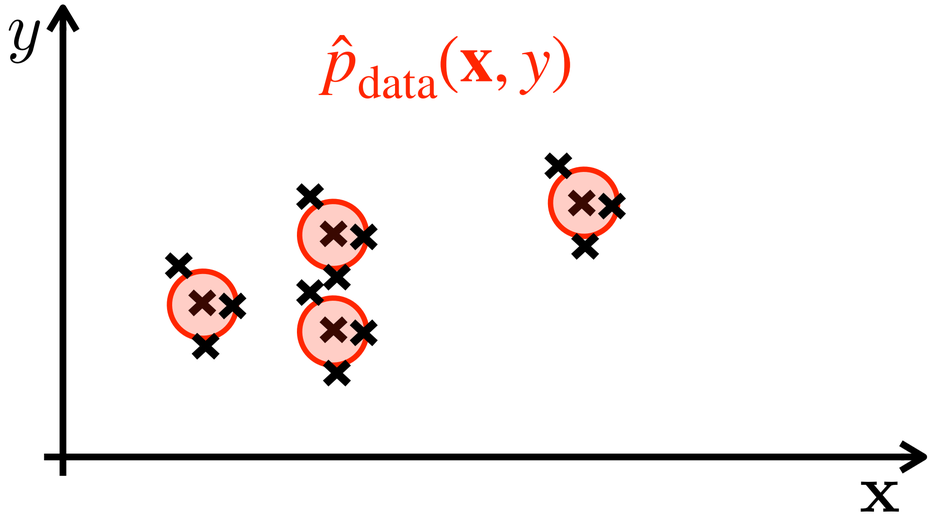
How to suppress overfitting?

Use close-to-infinite dataset

Data augmentation







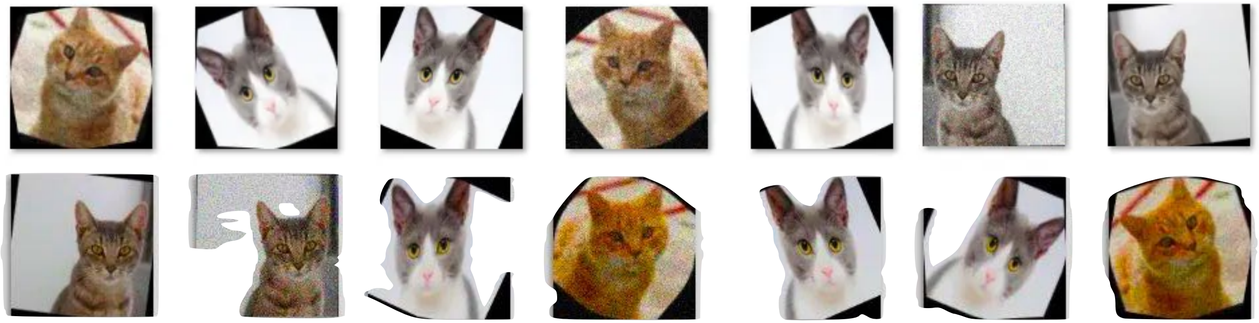
How to suppress overfitting?

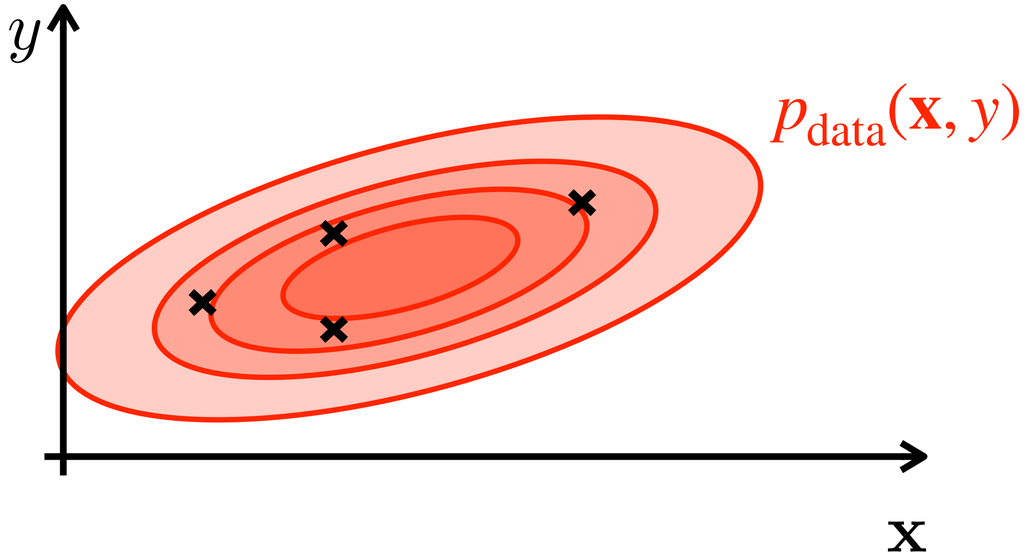
Use close-to-infinite dataset

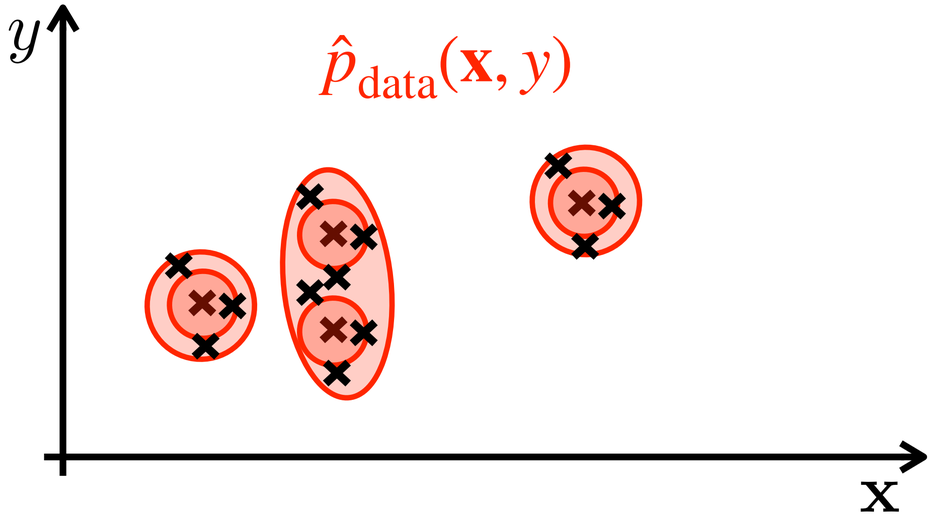
Reasonable geometrical and histogram transformations:

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

Or any generative model



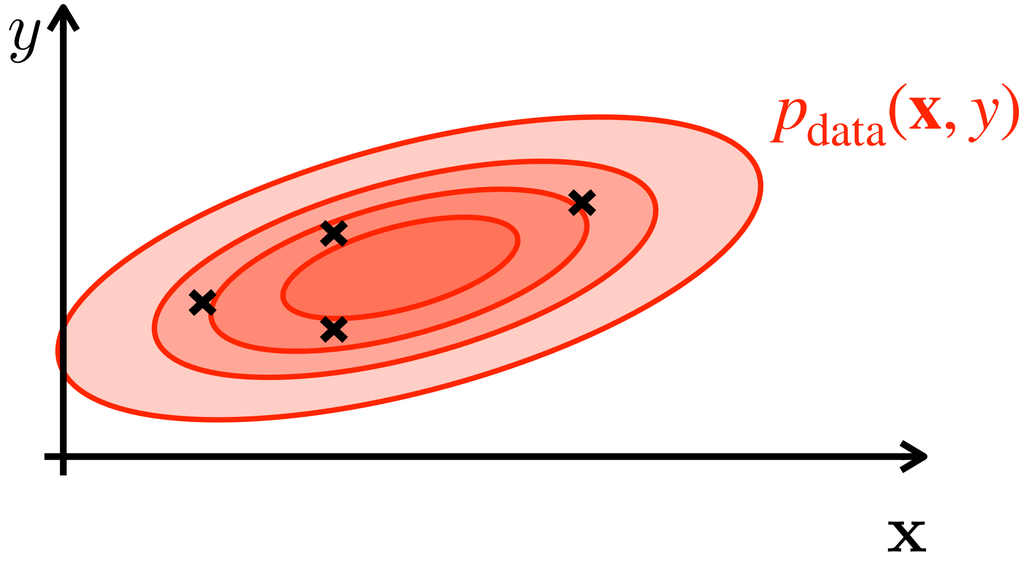


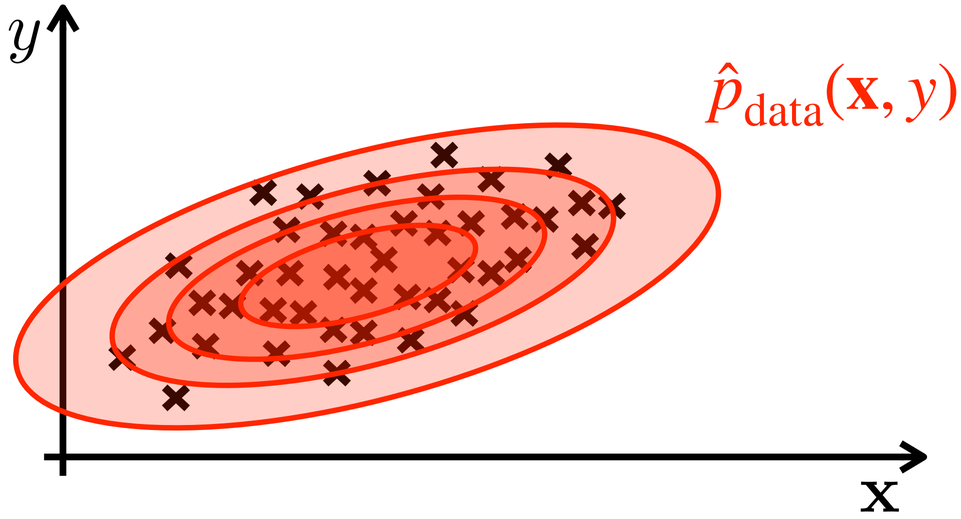


How to suppress overfitting?

Use close-to-infinite dataset

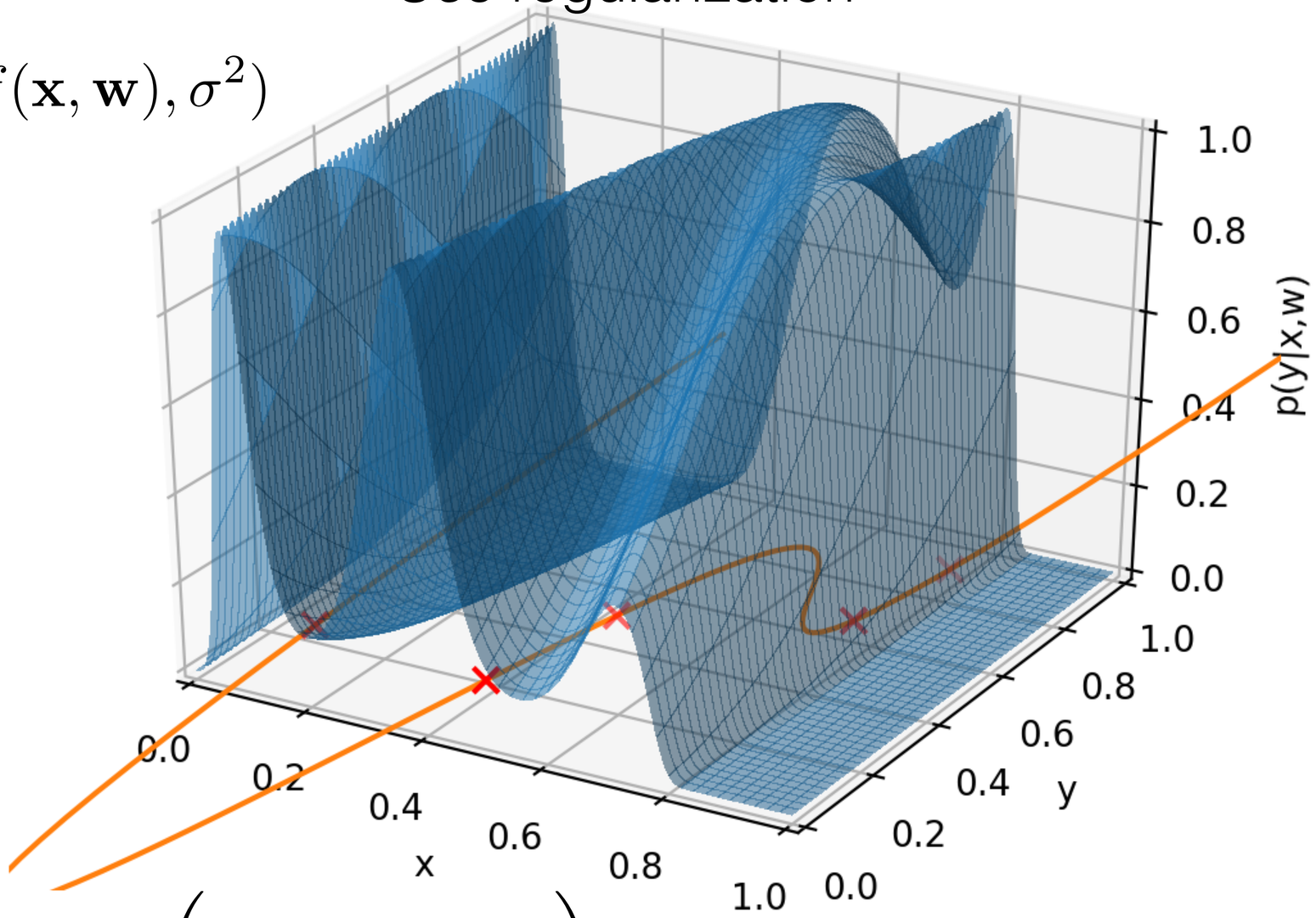
- Some datasets grows faster than we can learn from them:
  - Colorizing images (1074 imgs/sec uploaded to Instagram)
  - Autonomous cars (predicting vertical acceleration from images)
  - Youtube videos (predicting keywords in comments from video)
- The learning bottleneck stems from computational limitations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





How to suppress overfitting?
Use regularization

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



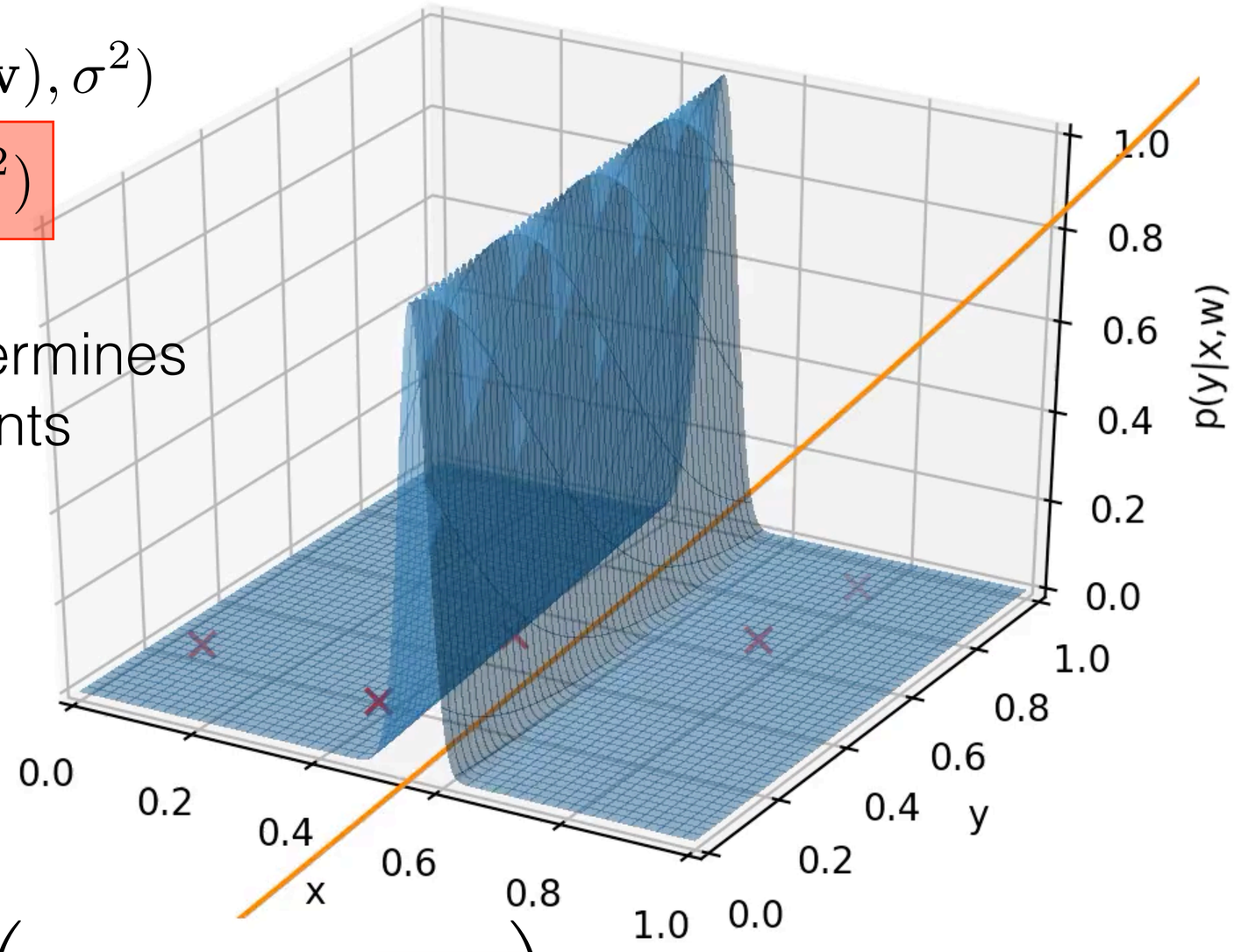
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

How to suppress overfitting?
Use regularization

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

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

Norm of weights determines Lipchitz constraints on the $\mathbf{f(x, w)}$



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

How to suppress overfitting?
Use regularization

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \mathscr{L}\big(f(\mathbf{x}, \mathbf{w}), y\big)$$

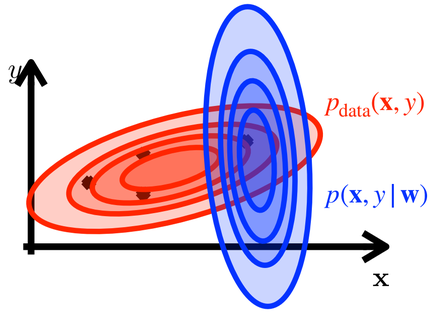
$$\mathbf{w} := \mathbf{w} - \alpha \frac{\partial \mathscr{L}\big(f(\mathbf{x}, \mathbf{w}), y\big)}{\partial \mathbf{w}}$$

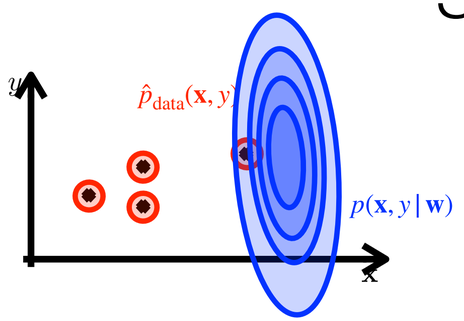
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) p(\mathbf{w}) \right) = \arg\min_{\mathbf{w}} \mathscr{L}\big(f(\mathbf{x}, \mathbf{w}), y\big) + \|\mathbf{w}\|$$

$$\mathbf{w} := \mathbf{w} - \alpha \left[\frac{\partial \mathscr{L}\big(f(\mathbf{x}, \mathbf{w}), y\big)}{\partial \mathbf{w}} + 2\mathbf{w} \right] = (1 - 2\alpha)\mathbf{w} - \alpha \frac{\partial \mathscr{L}\big(f(\mathbf{x}, \mathbf{w}), y\big)}{\partial \mathbf{w}}$$

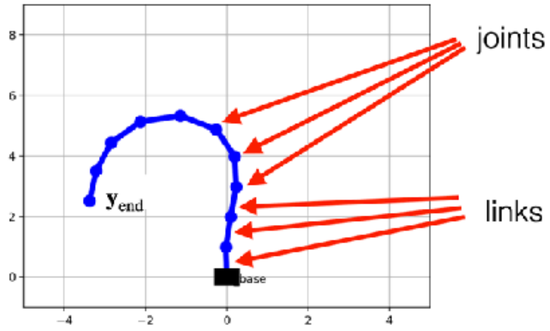
Weight decay implemented as a part of optimizer

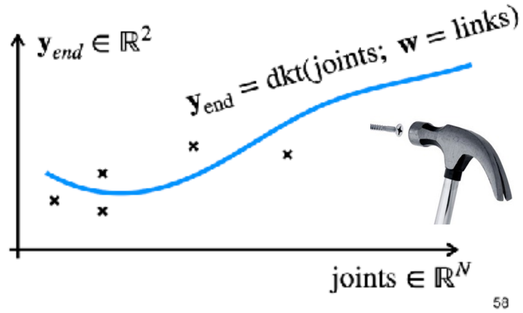
Summary overfitting

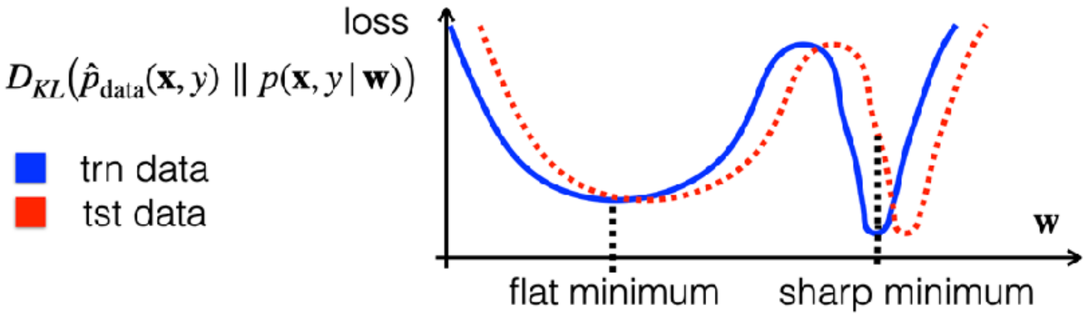




Optimization ≠ Machine learning ⇒ Overfitting





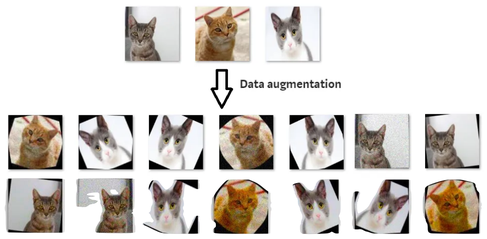


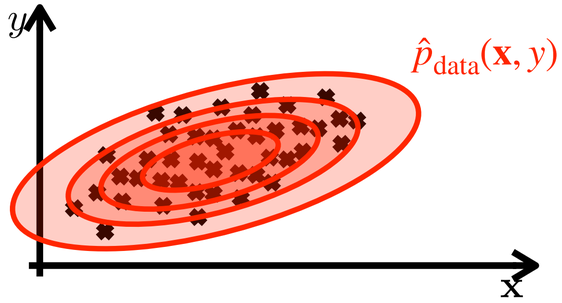
Always use the right tool/model

- Avoid any "*not-well justified leprechauns*" in the model

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

- Less is sometimes more



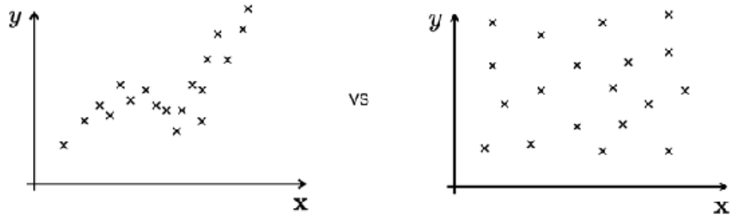


Use close-to-infinite training data

- More is sometimes more ;-)

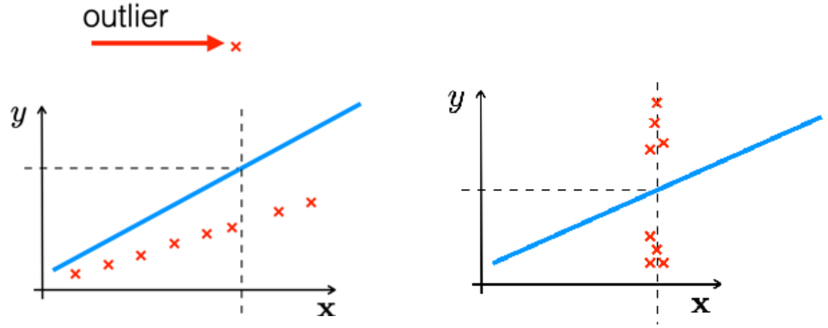
Golden grale:

- Solve only “Pilcik-free” problems



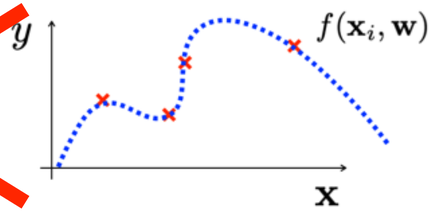
Lecture 1

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



This Lecture

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



Lecture on ConvNets
∇ODE, ∇argmin

- Improve the weight optimization algorithm (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.