Deep Learning Essentials

3. Maximum Likelihood Estimation (MLE), KL Divergence

Where does loss function come from, what causes overfitting?

Lukáš Neumann

Adapted from B3B33UROB slides of Karel Zimmerman

Motivating example: Fair coin

- *TASK #1: We have a coin and we want to estimate what is the probability of Heads*
  - Fair coin = probability of heads and tails is the same
  - In statistics, fair coin is term for
    a sequence of trials, each with a probability of 1/2

https://izbicki.me/blog/how-to-create-an-unfair-coin-and-prove-it-with-math.html

Maximum Likelihood Estimation

1. We assume data come from some distribution $\mathbf{p}(\mathrm{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are some parameters of the distribution
2. We observe data coming from the distribution $\mathbf{p}(\mathrm{x}; \boldsymbol{\theta})$, i.e. we have training data
3. We want to find the (unknown) value of $\boldsymbol{\theta}$

- How? We pick the $\boldsymbol{\theta}$ which maximizes the probability that we actually had observed the data we did

Maximum Likelihood Estimation

- How? We pick the $\boldsymbol{\theta}$ which maximizes the probability that we actually had observed the data we did
- We define the likelihood function $L(\boldsymbol{\theta})$ of the training data

$$L(\boldsymbol{\theta}) = \prod_{\mathrm{i}=1}^{\mathrm{N}} \mathbf{p}(\mathrm{x_i}; \boldsymbol{\theta})$$

- and find parameters $\boldsymbol{\theta}^*$ which maximize the likelihood

$$\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmax}}\, L(\boldsymbol{\theta})$$

Maximum Likelihood Estimation

- *TASK #1: We have a coin and we want to estimate what is the probability for Heads*
- Independent trials, each trial has only two outcomes → Bernoulli distribution

$$\mathbf{p}(\text{heads}) = \kappa$$

$$\mathbf{p}(\text{tails}) = 1 - \kappa$$

- In our training data, we have observed H times heads and T times tails
- The likelihood function is then given as

$$L(\kappa) = \prod_{\mathrm{i}=1}^{\mathrm{N}} \mathbf{p}(\mathrm{x_i}; \kappa) = \mathbf{p}(\text{heads})^{\mathrm{H}} \mathbf{p}(\text{tails})^{\mathrm{T}} = \kappa^{\mathrm{H}} (1 - \kappa)^{\mathrm{T}}$$

Maximum Likelihood Estimation

- We want find κ which maximizes likelihood $L(\kappa)$
- We take the derivative $\frac{\partial L}{\partial \kappa}$ and set it to 0
- In practice, is almost always easier to maximize log-likelihood rather than likelihood itself
  - logarithm is monotonically increasing function, it does not change the minimum

$$L(\kappa) = \kappa^{\mathrm{H}}(1-\kappa)^{\mathrm{T}}$$

$$\log L(\kappa) = \log \kappa^{\mathrm{H}}(1-\kappa)^{\mathrm{T}}$$

$$l(\kappa) = \mathrm{H} \cdot \log \kappa + \mathrm{T} \cdot \log (1-\kappa)$$

$$\frac{\partial l}{\partial \kappa} = \frac{\mathrm{H}}{\kappa} - \frac{\mathrm{T}}{1-\kappa} = 0$$

$$\mathrm{T} \cdot \kappa = \mathrm{H} \cdot (1-\kappa)$$

$$\kappa = \frac{\mathrm{H}}{\mathrm{H}+\mathrm{T}}$$

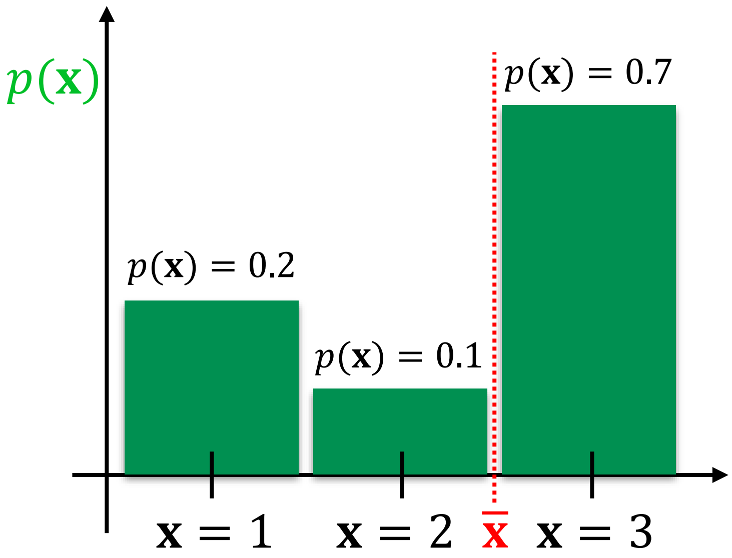
Maximum Likelihood Estimation

Coin	Heads	Tails	κ
0	53	47	0.53
1	55	45	0.55
2	49	51	0.49
3	41	59	0.41
4	39	61	0.39
5	27	73	0.27
6	0	100	0

Mean and Average

- Mean

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



Mean and Average

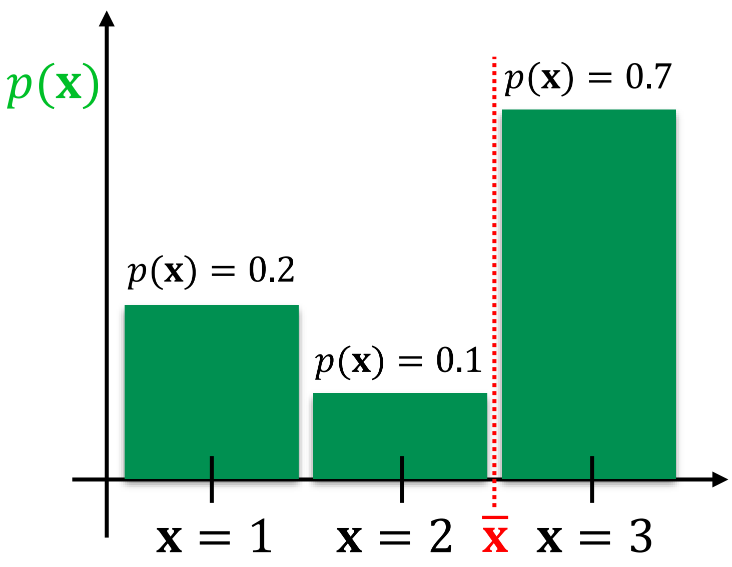
- Mean

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

- Average

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

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



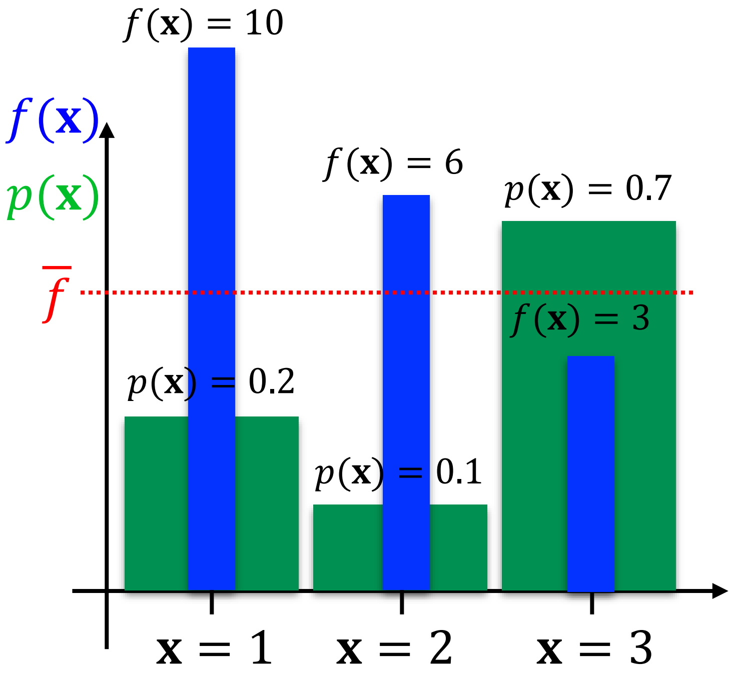
Mean and Average

- Mean

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

- Average

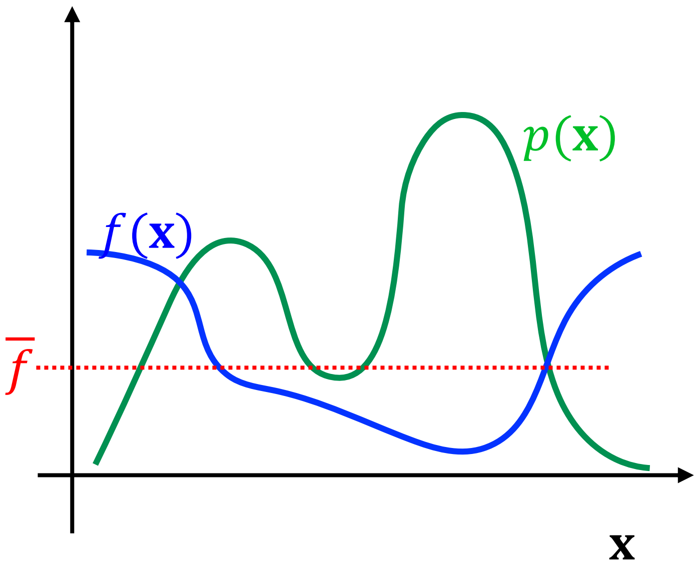
$$\approx \frac{1}{N} \sum_i f(\mathbf{x}_i) = \frac{1}{10}(10 + 10 + 6 + 3 + 3 + 3 + 3 + 3 + 3 + 3) = 4.7$$



Mean and Average

- For continuous case

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

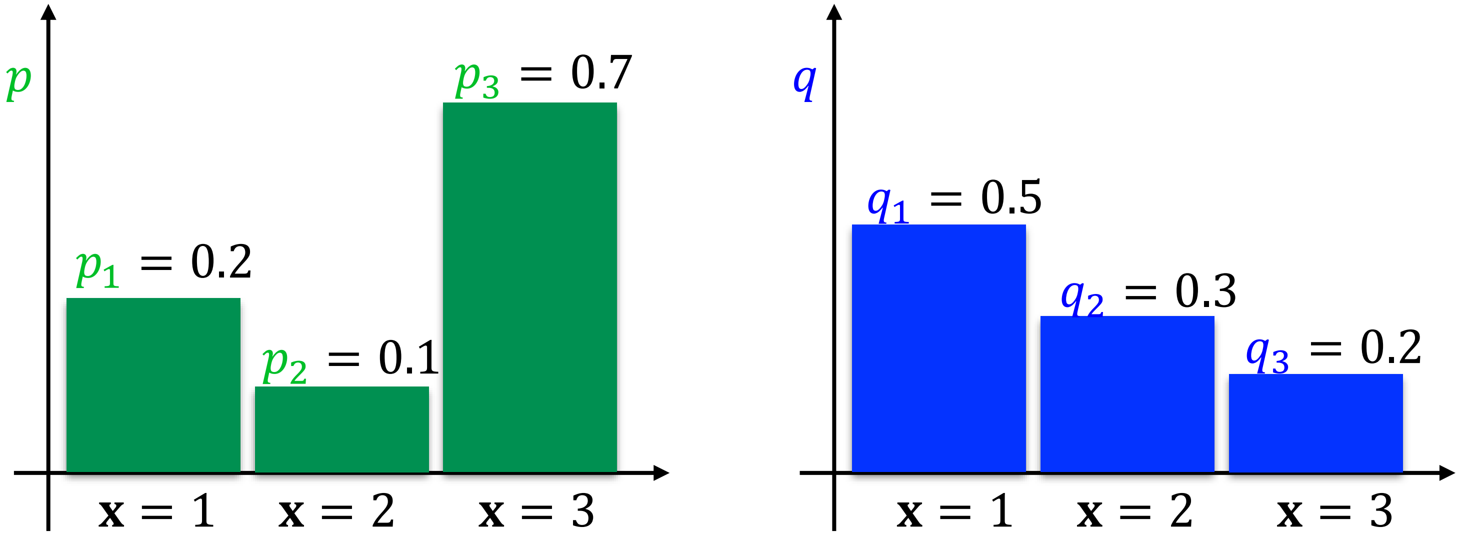


KL-divergence

- Kullback–Leibler (KL) divergence measures of how much the probability distribution p is similar to q

$$D_{KL}(p \parallel q) = \sum_i p_i \cdot \log \frac{p_i}{q_i}$$

- Informally, it measures the "expected surprise" when using q as approximation of p



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

KL-divergence

- Kullback–Leibler (KL) divergence measures of how much the probability distribution p is similar to q

$$D_{KL}(p \parallel q) = \sum_i p_i \cdot \log \frac{p_i}{q_i}$$

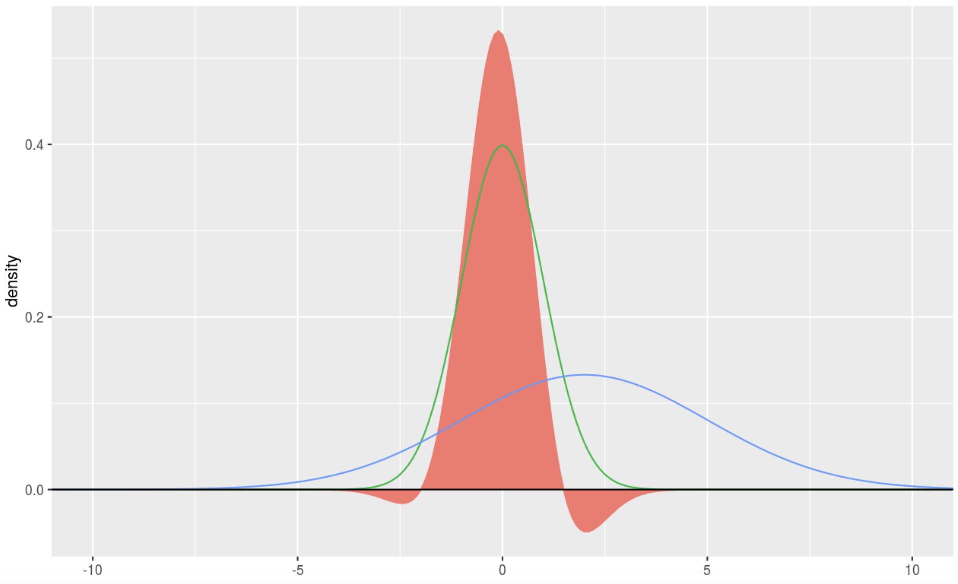
- Informally, it measures the “expected surprise” when using q as approximation of p

  - What if $q_k = p_k$? $D_{KL}(p \parallel q) = 0$
  - What if $q_k \to 0$ and $p_k > 0$? $D_{KL}(p \parallel q) \to \infty$
  - Is it symmetrical? $D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$

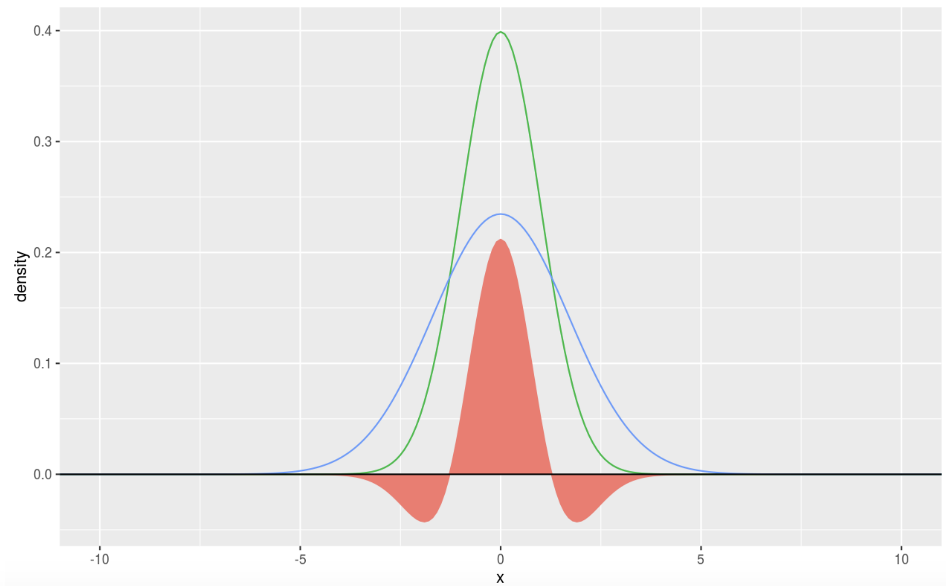
KL-Divergence

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

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



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



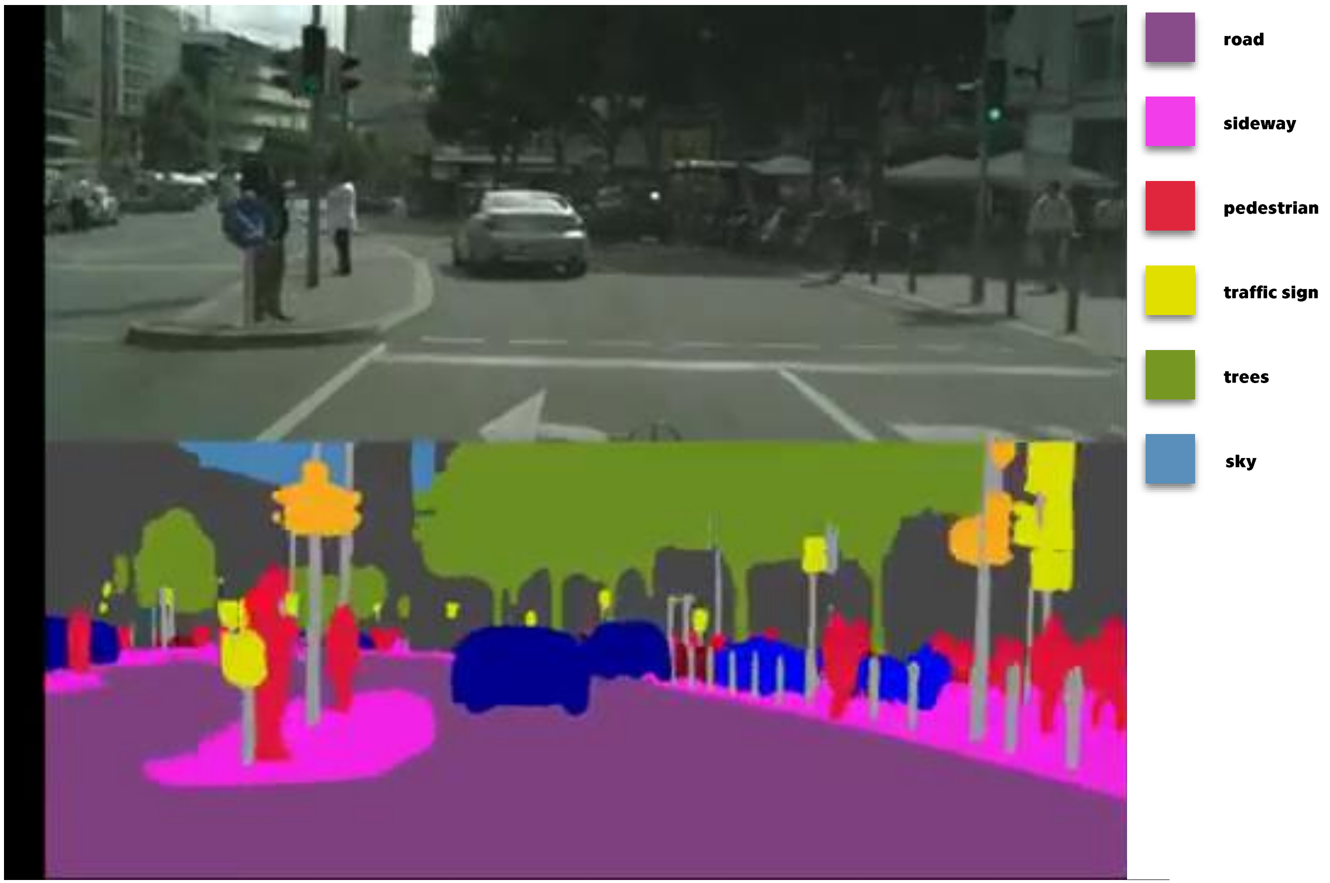
https://gnarlyware.com/blog/kl-divergence-online-demo/

Where does loss function come from?

- Most tasks come in this form: *given* $\mathbf{x}$, *estimate* y

$\mathbf{x}$

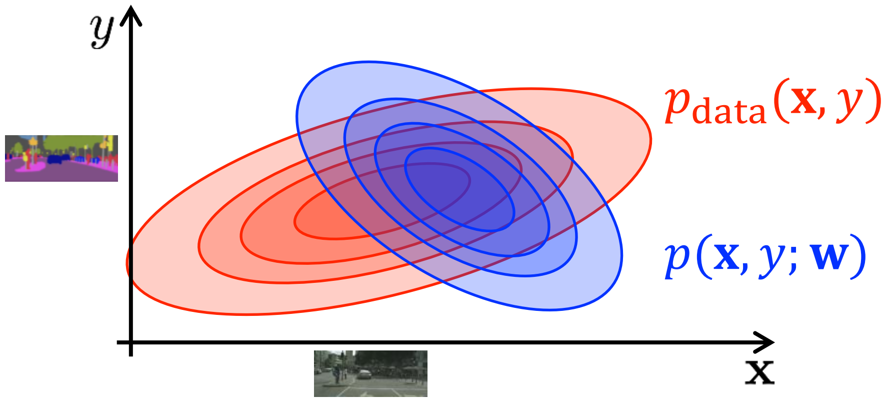
y



Where does loss function come from?

- The $(\mathbf{x}, y)$ tuples are from an unknown distribution $p_{\text{data}}(\mathbf{x}, y)$
- We try to approximate it by $p(\mathbf{x}, y; \mathbf{w})$
- We search for parameters (weights) $\mathbf{w}$ that makes $p(\mathbf{x}, y; \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \operatorname*{argmin}_{\mathbf{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w}))$$



Where does loss function come from?

- We search for parameters (weights) $\mathbf{w}$ that makes $p(\mathbf{x}, y; \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}}\, D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w})) = \underset{\mathbf{w}}{\operatorname{argmin}} \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y; \mathbf{w})}$$

$$= \underset{\mathbf{w}}{\operatorname{arg}min}\, \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}(\mathbf{x},y)}[\log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y; \mathbf{w})}]$$

$$= \underset{\mathbf{w}}{\operatorname{arg}min}\, \mathbb{E}_{(\mathbf{x},y)}[\log \cancel{p_{\text{data}}(\mathbf{x}, y)} - \log p(y|\mathbf{x}; \mathbf{w})\cancel{p(\mathbf{x})}]$$

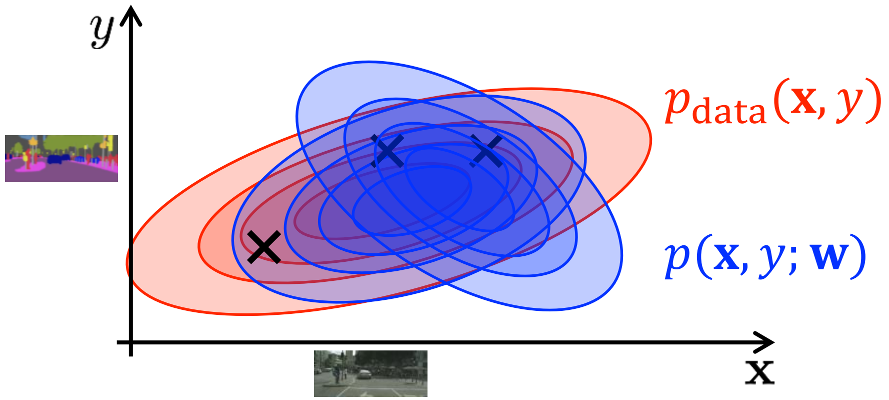
$$= \underset{\mathbf{w}}{\operatorname{arg}min}\, \mathbb{E}_{(\mathbf{x},y)}[-\log p(y|\mathrm{x}; \mathbf{w})]$$

$$\approx \underset{\mathbf{w}}{\operatorname{arg}min} \frac{1}{N} \sum_{(\mathbf{x}_i,y_i)\sim\, p_{\text{data}}(\mathbf{x}, y)} [-\log p(y_i|\mathbf{x}_i; \mathbf{w})] \qquad \text{MLE: } \Big(\underset{\mathbf{w}}{\operatorname{arg}max} \prod_{(\mathbf{x}_i,y_i)} p(y_i|\mathbf{x}_i; \mathbf{w})\Big)$$

Where does loss function come from?

- We search for parameters (weights) $\mathbf{w}$ that makes $p(\mathbf{x}, y; \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} [-\log p(y_i | \mathbf{x}_i; \mathbf{w})]$$



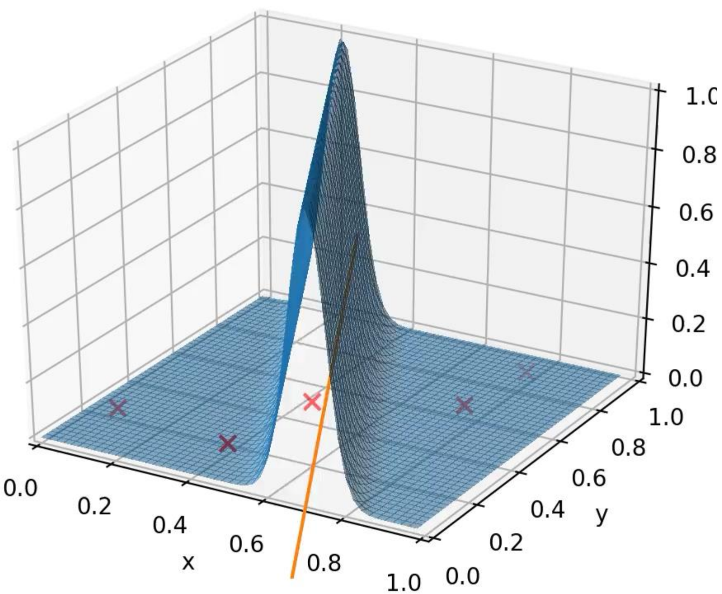
Where does loss function come from?

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} [-\log p(y_i|\mathbf{x}_i; \mathbf{w})]$$

Regression

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

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

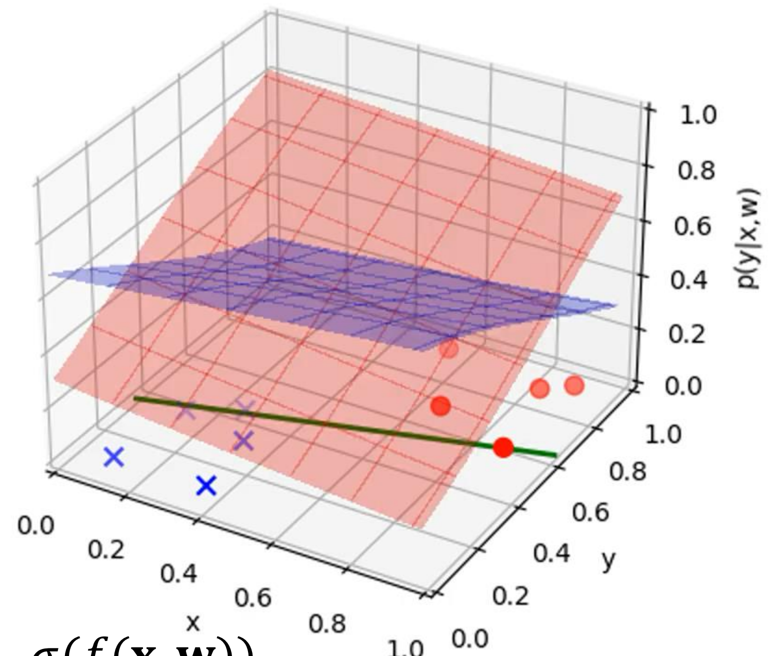


$K \cdot \exp^{-\frac{(f(\mathbf{x},\mathbf{w})-y)^2}{2\sigma^2}}$

Classification

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

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

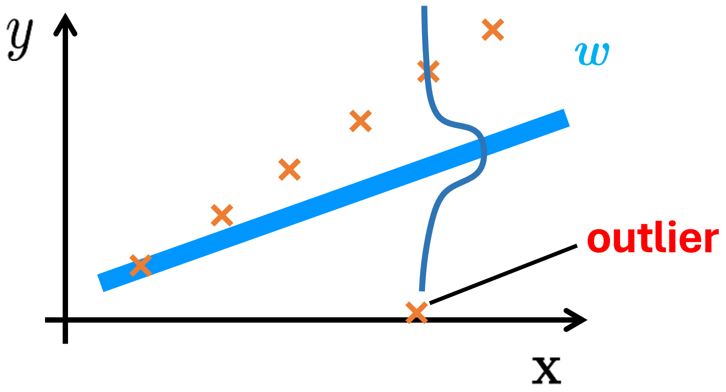


$\sigma(f(\mathbf{x},\mathbf{w}))$

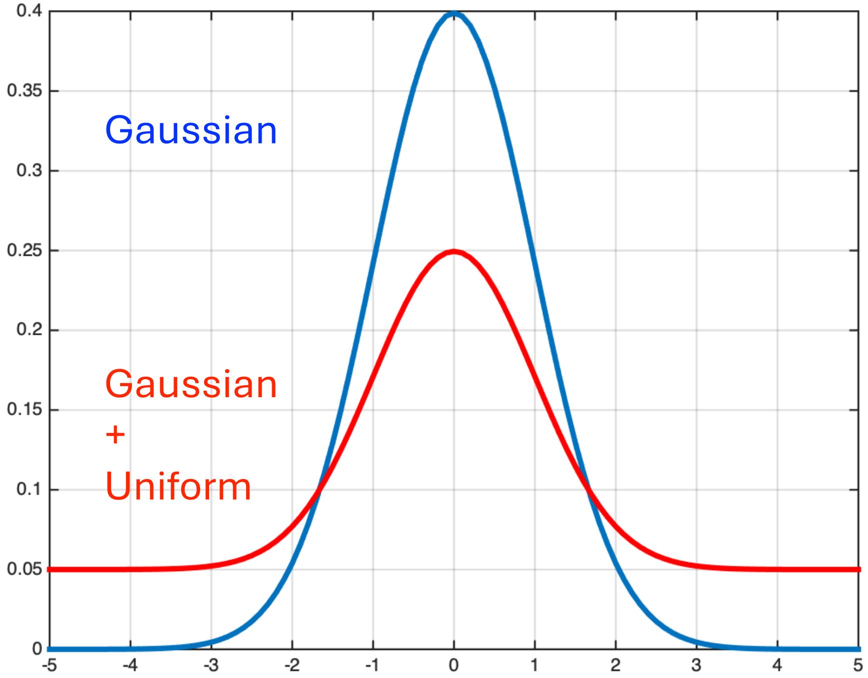
$1 - \sigma(f(\mathbf{x},\mathbf{w}))$

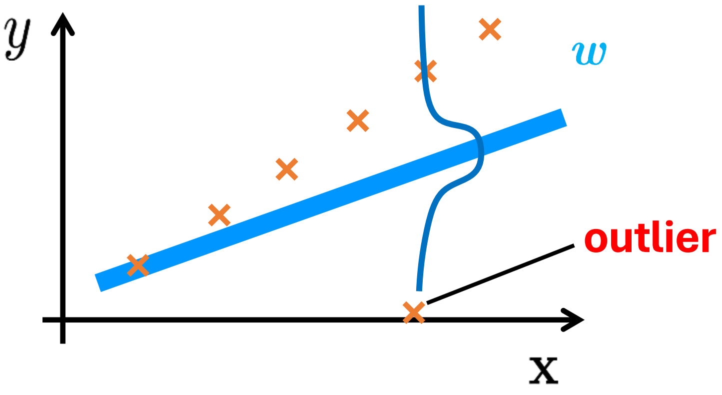
Where does loss function come from?

- **Lecture 01: What can go wrong**
- Why does the outlier skew our model?



Robust regression

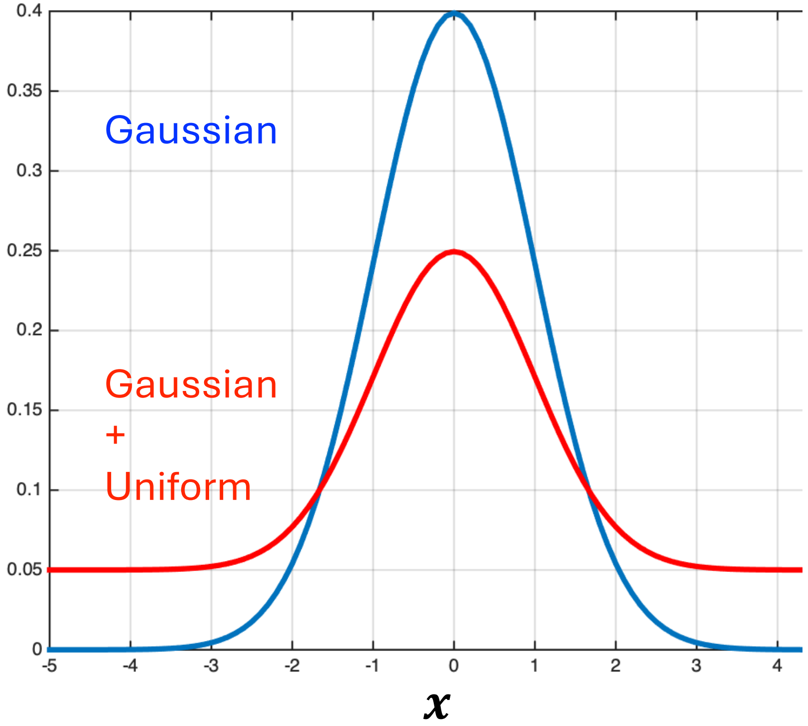




Robust regression

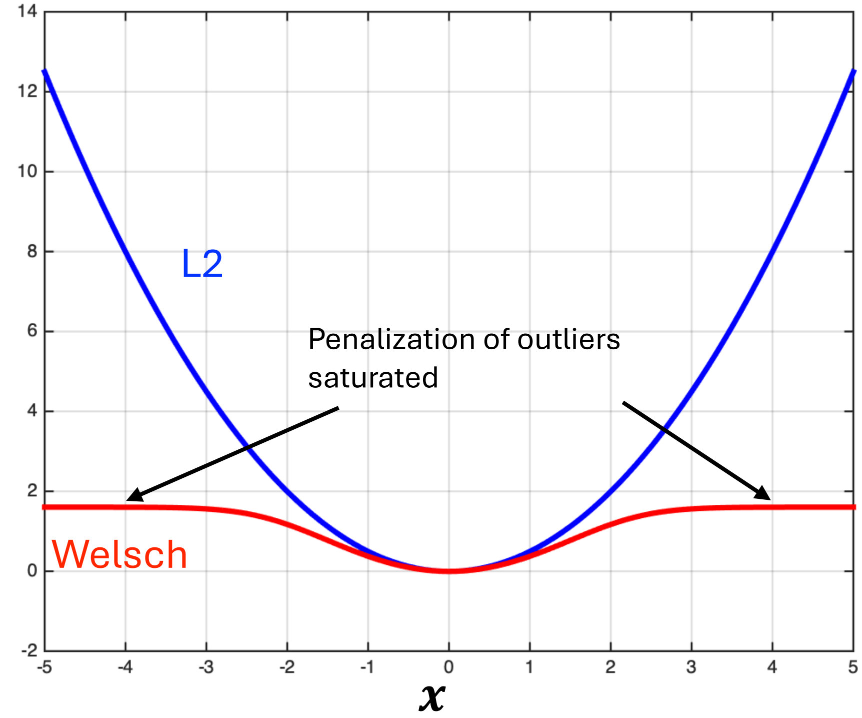
Probability distributions

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



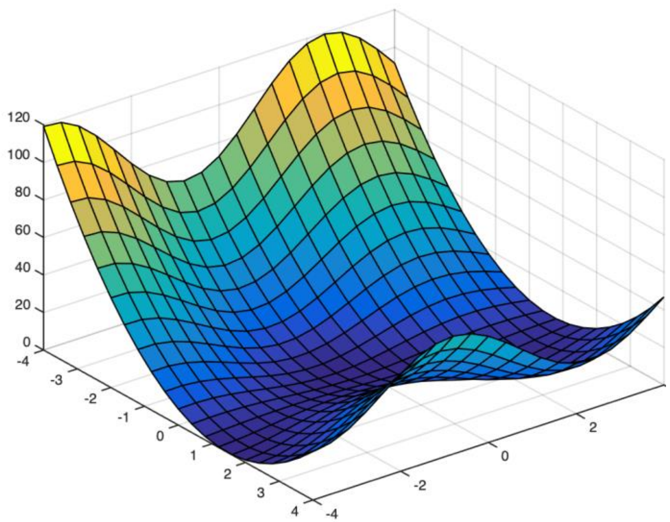
Corresponding losses

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



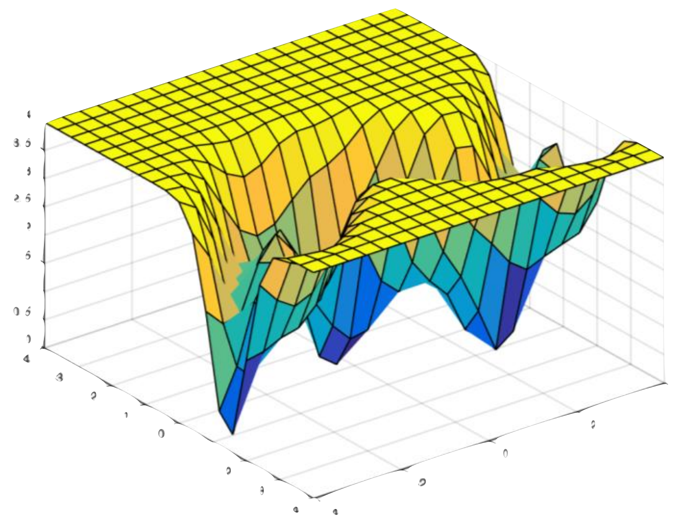
Robust regression

L2 landscape



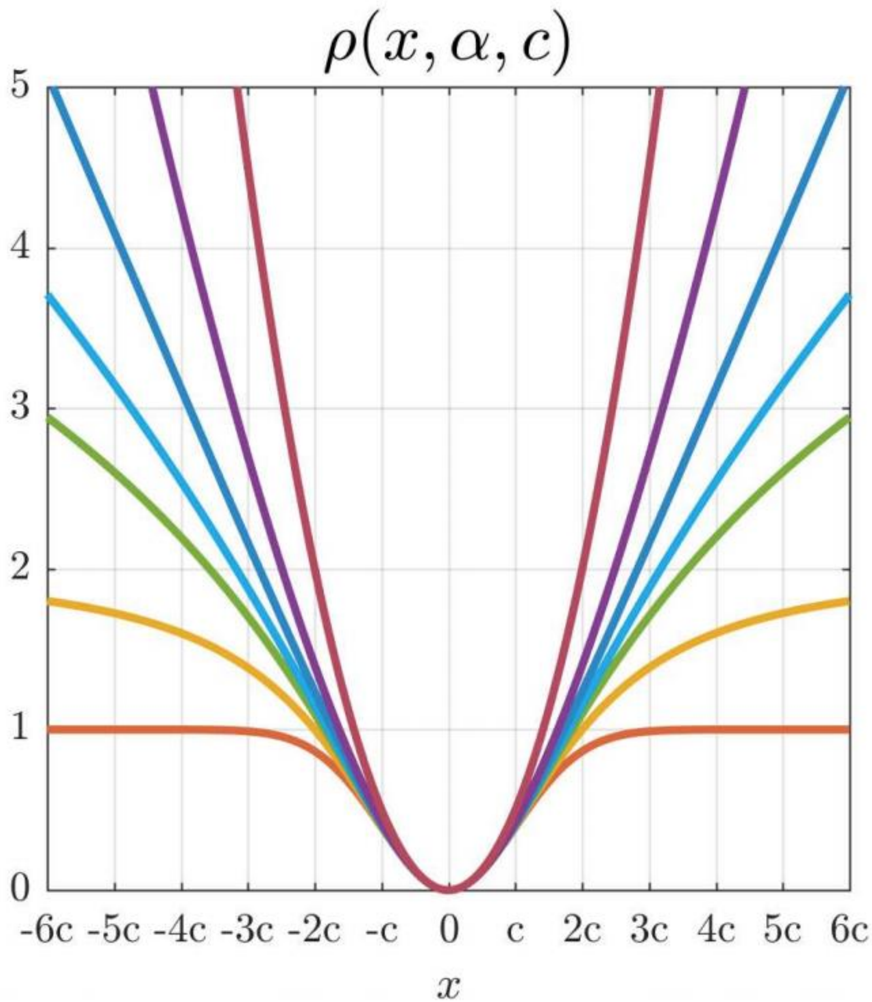
- **Uniform noise not modeled**
- GD-friendly landscape
- Gradient length encodes distance
- Easy to optimize

Welsch landscape



- **Uniform noise modeled**
- GD-unfriendly landscape
- Non-convex: Large narrow plateaus with zero gradient
- Good initialization required

Robust regression



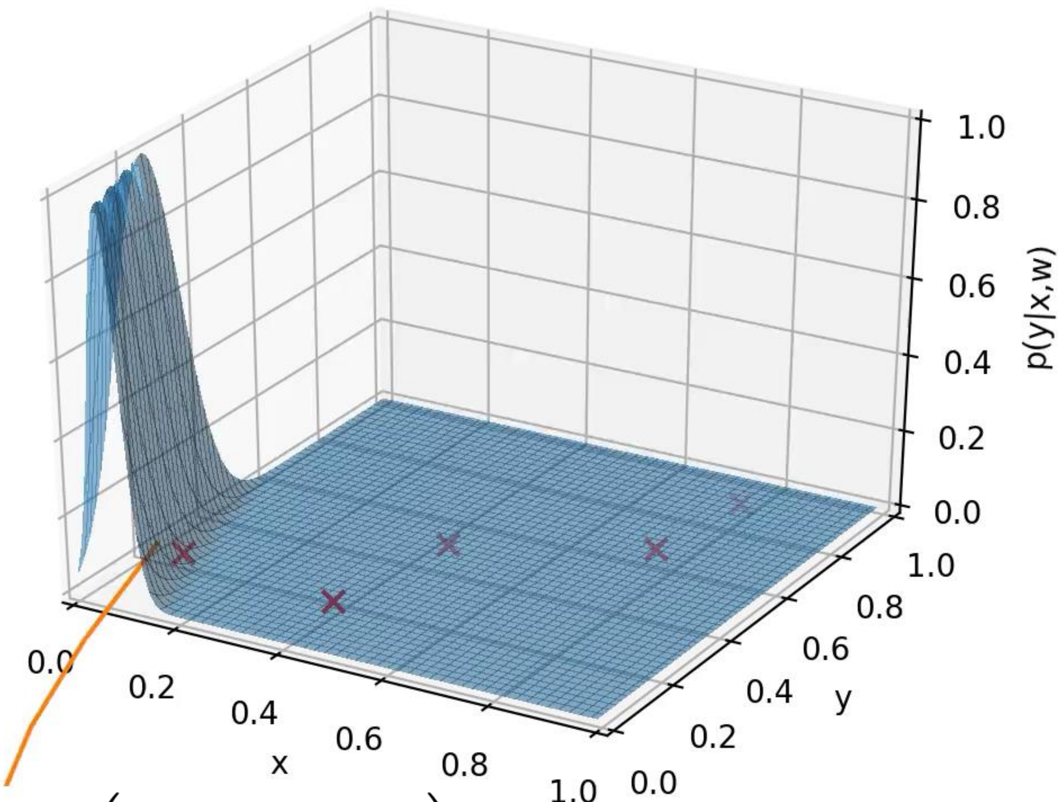
$$\rho(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right)$$

Barron, Jonathan T., "A general and adaptive robust loss function.", CVPR 2019.

Where does loss function come from?

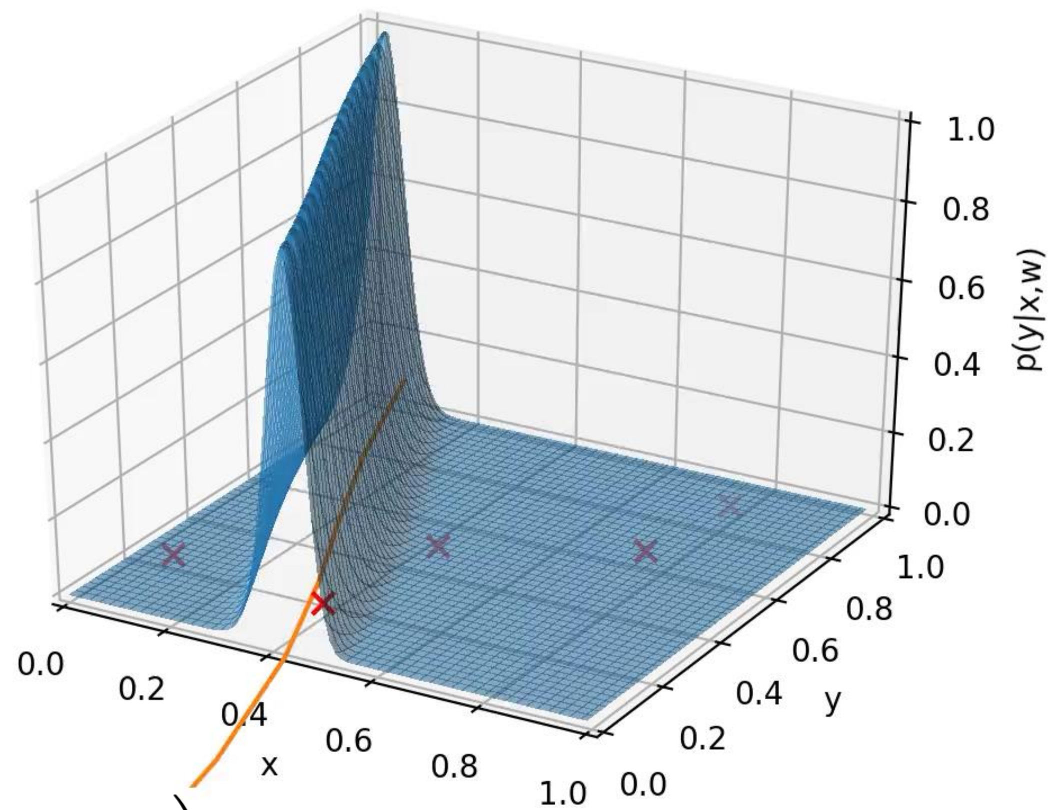
- **Summary**
  - Maximum Likelihood = Minimum KL-divergence = Minimum “-log(p)”-loss
  - Different losses suffer from different issues
  - There are trade-offs between loss function expressivity and our ability to optimize them

What causes overfitting?



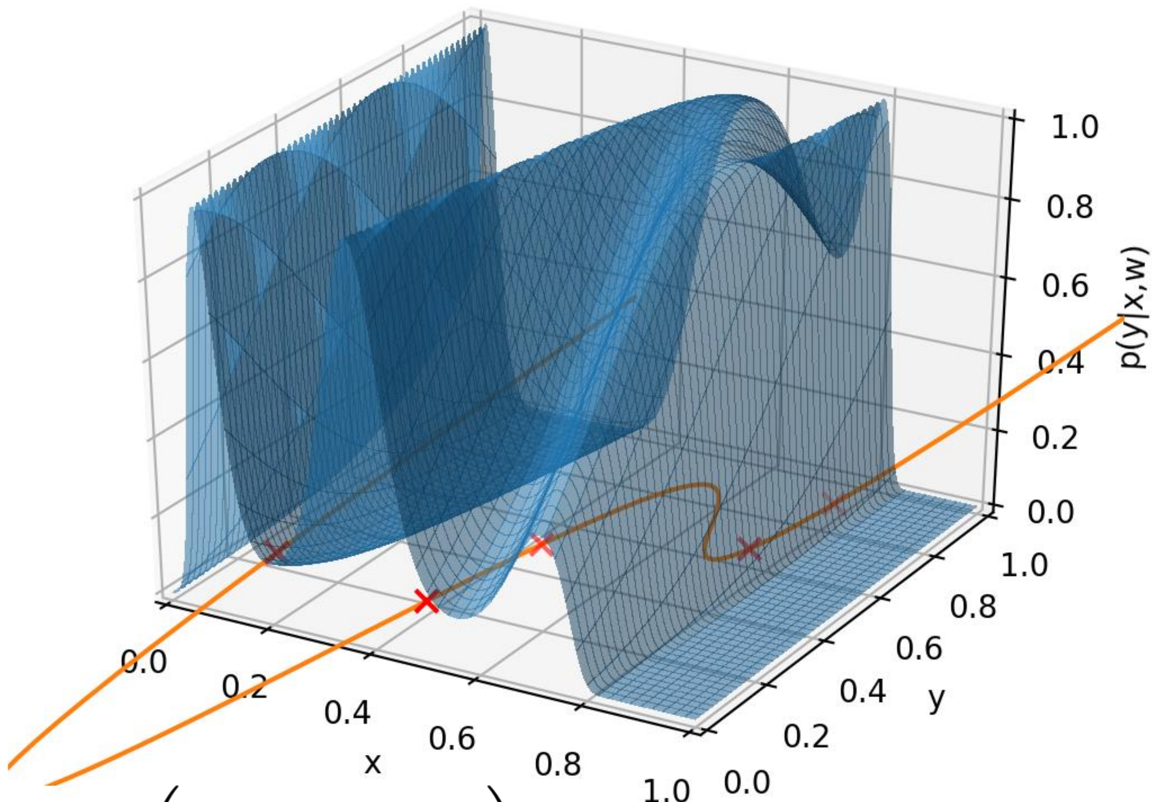
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (w_2 x_i^2 + w_1 x_i + w_0 - y_i)^2$$

What causes overfitting?



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

What causes overfitting?

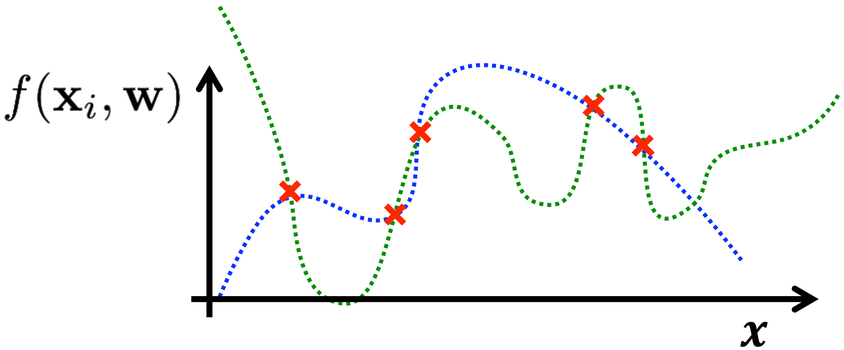


$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) \right) = \arg\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

What causes overfitting?

- **Ockham's razor**
  - “Many stories consistent with a broken vase. Lepricons can be involved in any explanation”
  - Use the simplest (the most apriori probable) explanation

William of Ockham (1287-1347)



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

What causes overfitting?

- **Phaistos Disc**
  - 16cm disc
  - 45 distinct symbols (Unicode ☺)
  - 242 characters in total
- People “deciphered” the disc as:
  - Religious text
  - Text commemorating military victory
  - Teaching tool
  - Board game
  - Calendar
  - Modern creation to attract archeology funding
- Many stories consistent with a sequence of visual symbols

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

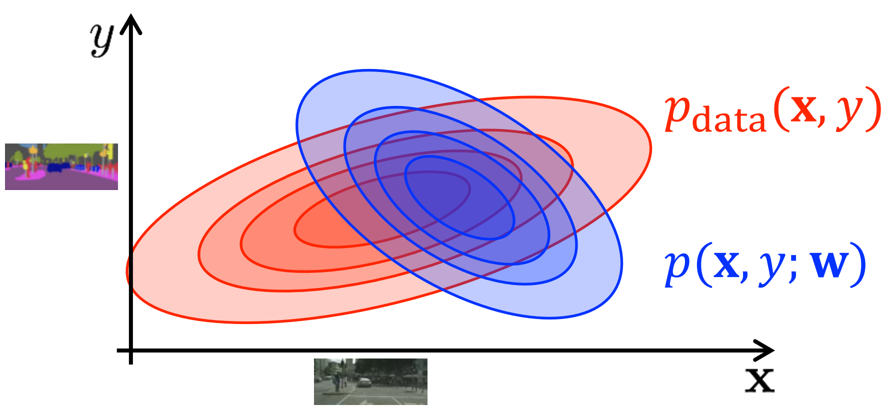
What causes overfitting?

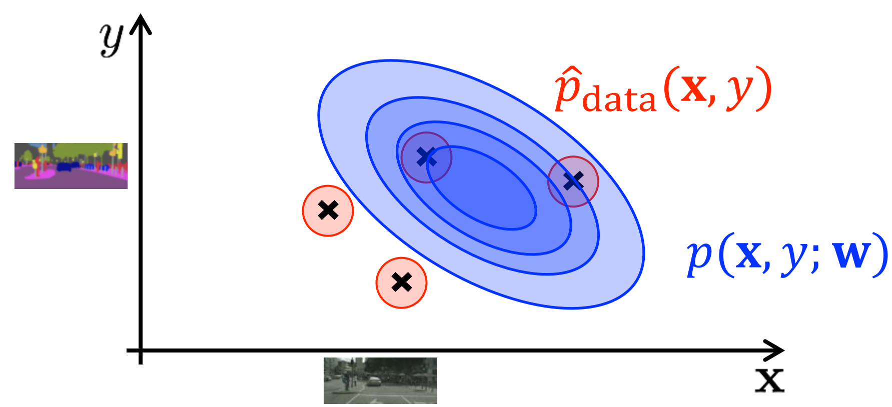
- We search for parameters (weights) $\mathbf{w}$ that makes $p(\mathbf{x}, y; \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}}\, D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w}))$$

- Because $p_{\text{data}}(\mathbf{x}, y)$ is unknown, we use a set of samples (=training set)
- But since the training set is finite, it has a different distribution $\hat{p}_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}}\, D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w})) \neq \underset{\mathbf{w}}{\operatorname{argmin}} D_{KL}(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w}))$$





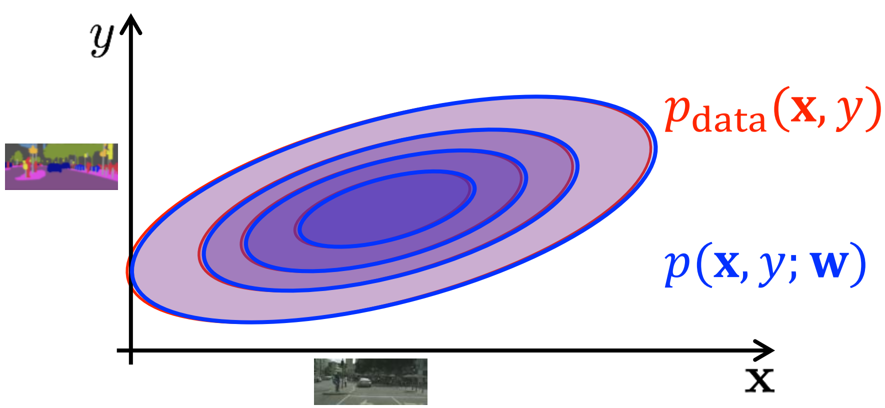
What causes overfitting?

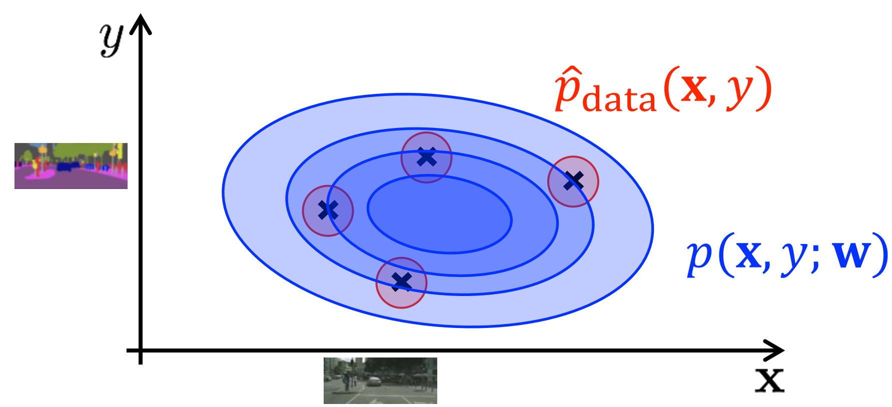
- We search for parameters (weights) $\mathbf{w}$ that makes $p(\mathbf{x}, y; \mathbf{w})$ close to $p_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \operatorname*{argmin}_{\mathrm{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w}))$$

- Because $p_{\text{data}}(\mathbf{x}, y)$ is unknown, we use a set of samples (=training set)
- But since the training set is finite, it has a different distribution $\hat{p}_{\text{data}}(\mathbf{x}, y)$

$$\mathbf{w}^\star = \operatorname*{argmin}_{\mathrm{w}} D_{KL}(p_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w})) \neq \operatorname*{arg\mathit{min}}_{\mathbf{w}} D_{KL}(\hat{p}_{\text{data}}(\mathbf{x}, y) \parallel p(\mathbf{x}, y; \mathbf{w}))$$

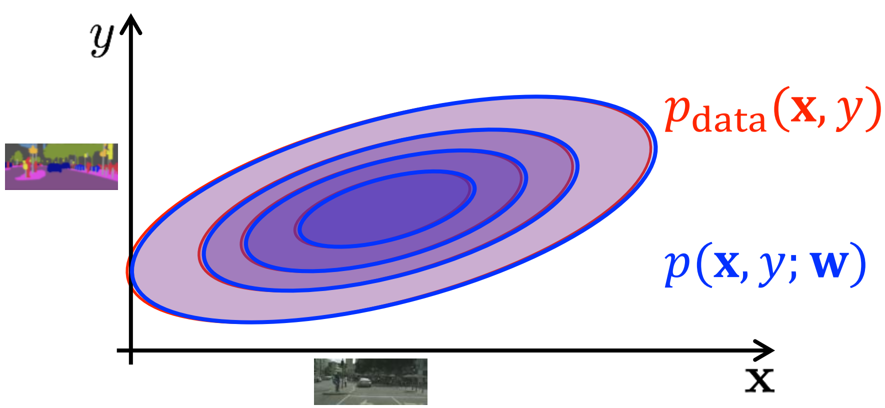


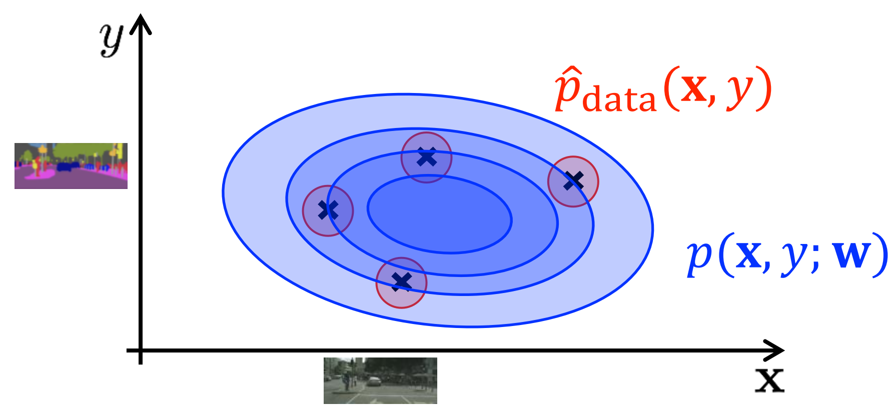


What causes overfitting?

- **In machine learning we try to optimize a criterion we don't have access to, so we only optimize its approximation**

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





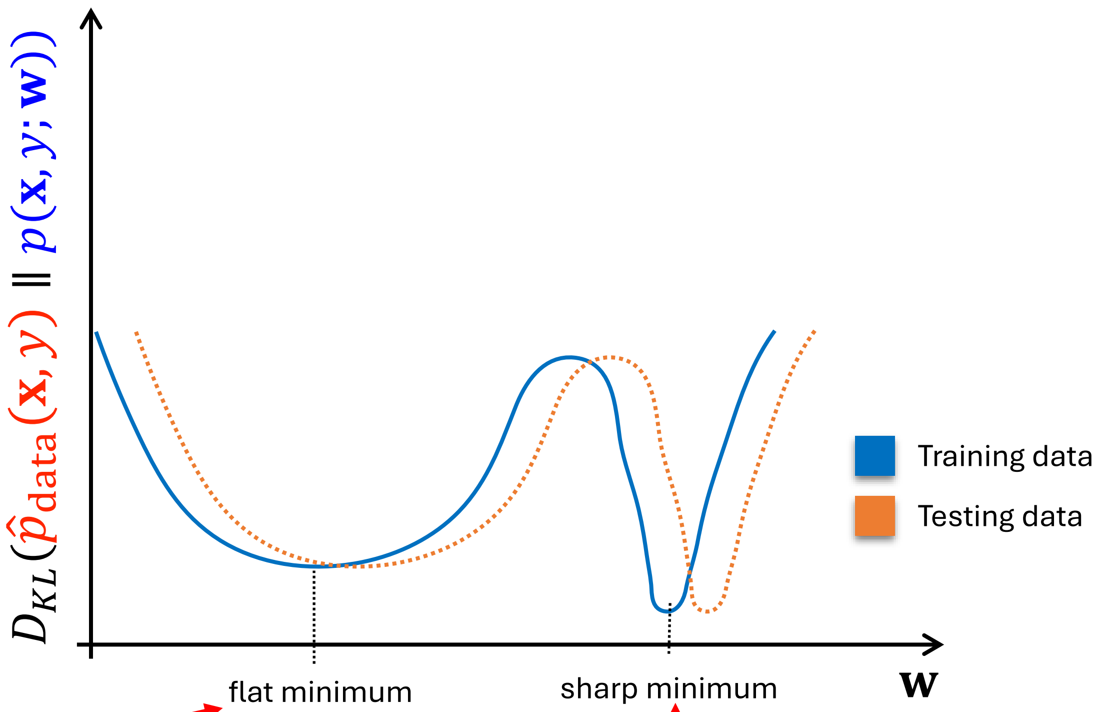
Reducing overfitting

1) Always use the right tool

- Use the right $p(\mathbf{x}, y; \mathbf{w})$ that generates only "shapes" similar to $p_{\text{data}}(\mathbf{x}, y)$
- Embed prior knowledge (physics, geometry, biology…) about the problem into 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 for robots
    - Structure of animal cortex (CNNs)

Reducing overfitting

Which minimum is better?



Good generalization

Testing error remains small

Weak generalization → optimum prone to overfitting

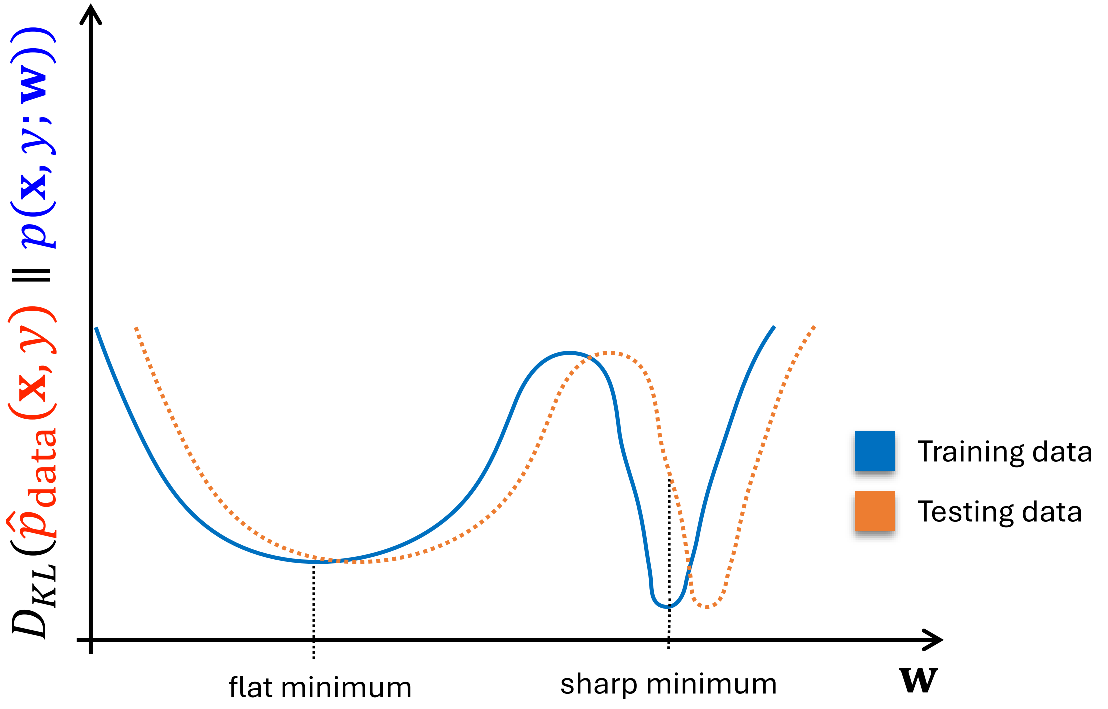
Error grows fast with a small training/testing data shift

Reducing overfitting

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

- Optimization techniques (e.g. SGD with momentum)
- Loss function (e.g. $\min_{w} \max_{\|\epsilon\|_2 \leq \rho} L_{train}(w + \epsilon)$)

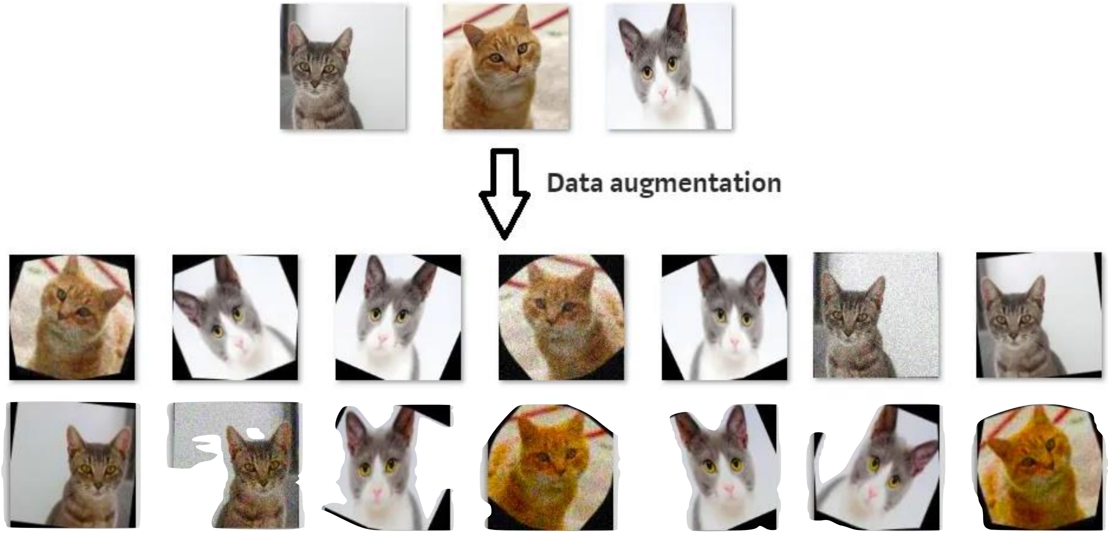
Foret, Pierre, et al. "Sharpness-aware Minimization for Efficiently Improving Generalization." ICLR 2021



Reducing overfitting

3) Get more training data

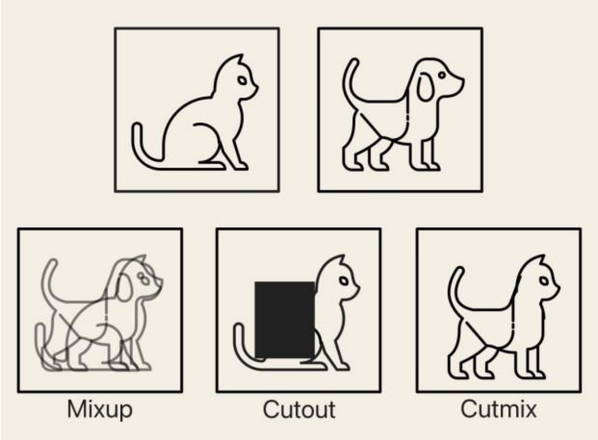
- Generate (almost infinite) dataset through data augmentation
- In every training pass, the networks actually “sees” different image

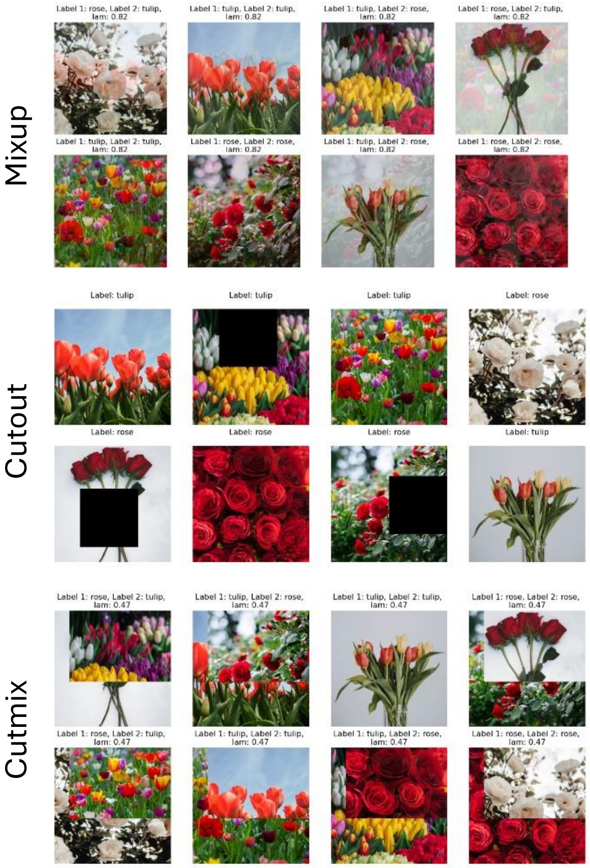


Reducing overfitting

3) Get more training data

- Generate (almost infinite) dataset through data augmentation
- In every training pass, the networks actually “sees” different inputs
- Commonly used augmentations:
  - Horizontal (and vertical) flipping
  - Rotation
  - Random crop and resize
  - Color jittering (brightness, contrast, hue)
  - Gaussian noise / blur
  - Mixup, Cutout, Cutmix (for image classification)





Reducing overfitting

- **Summary**
  - **Optimization ≠ Machine learning**, optimization can lead to overfitting
  - **Always use the right tool**, incorporate prior knowledge
  - **Prefer simpler solutions**, less is sometimes more
  - **Use as much training data as possible**, more is sometimes more ☺

Competencies gained for the test

- Derive MLE estimate of a given distribution
- Understand connection between KL divergence, loss, optimization and machine
- Understand underfitting, overfitting and model architectures
- How to reduce overfitting