Learning for vision I

Karel Zimmermann
http://cmp.felk.cvut.cz/~zimmerk/

Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

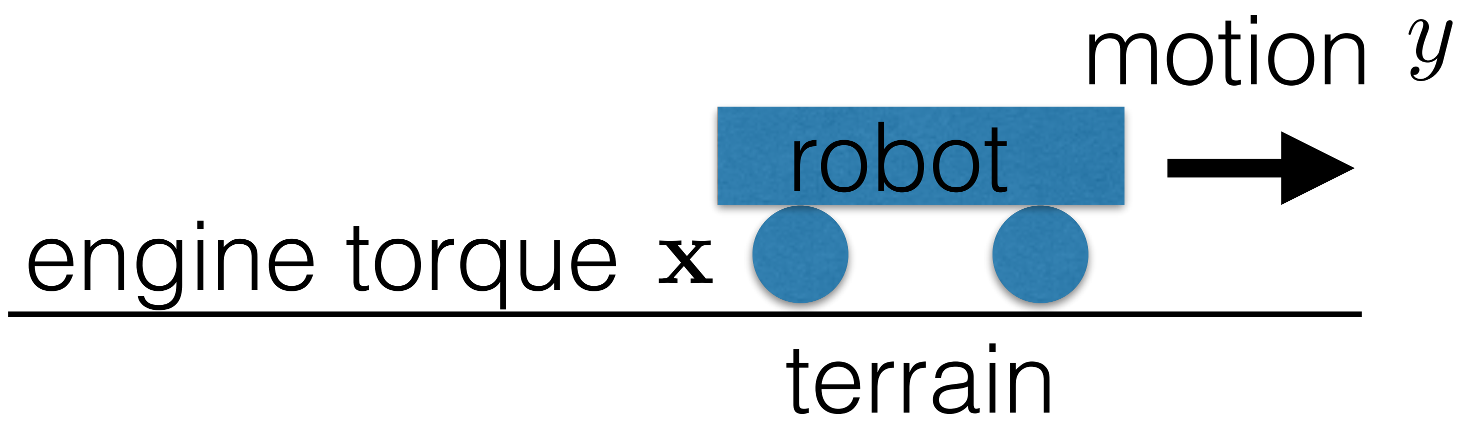
Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

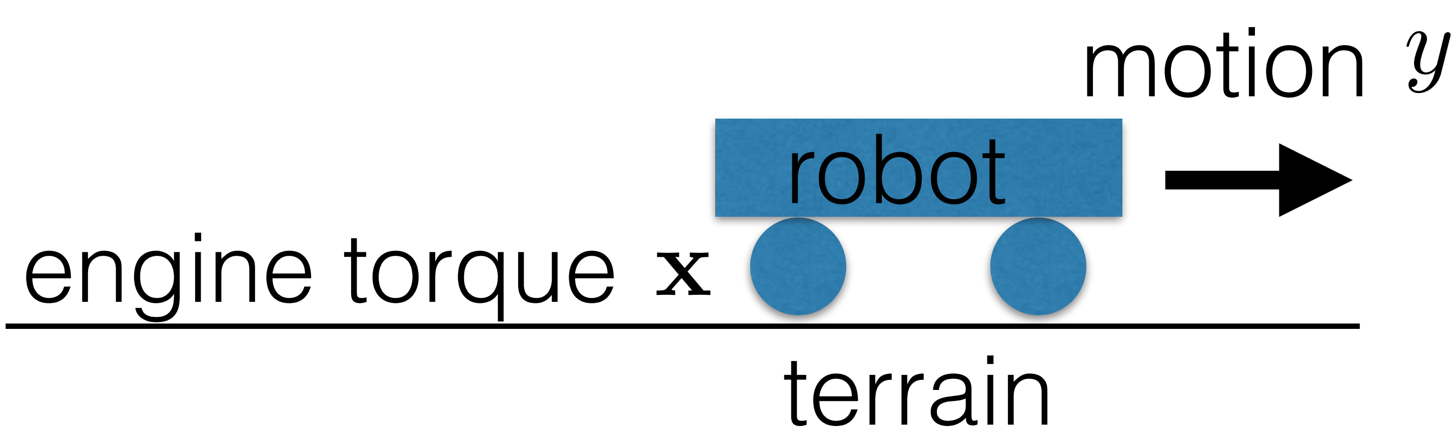
Outline

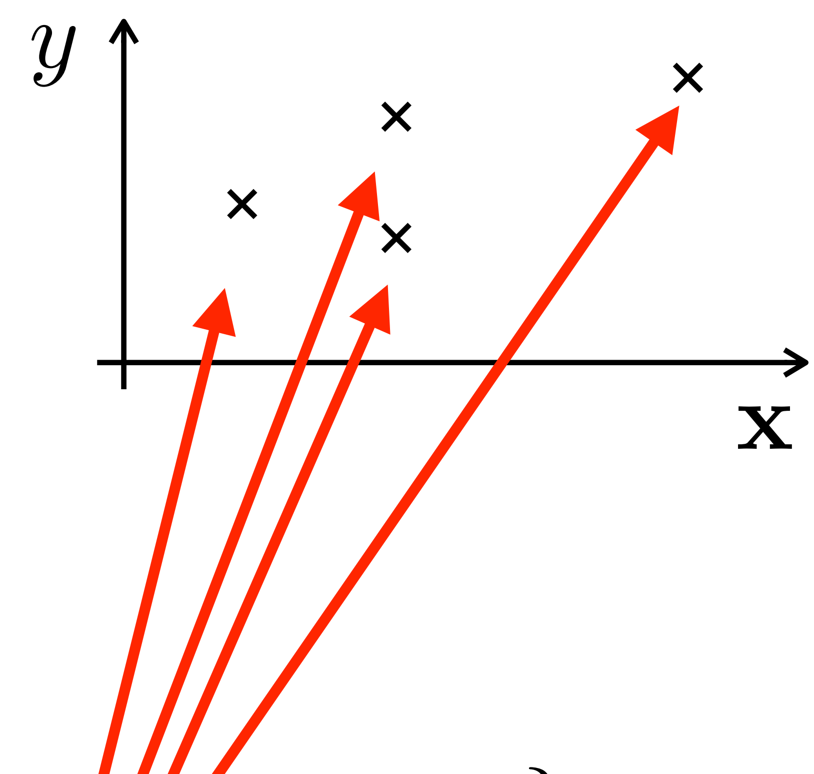
- Pre-requisites: linear algebra, Bayes rule
- MAP estimation, prior and overfitting
- Linear regression
- Linear classification

- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model



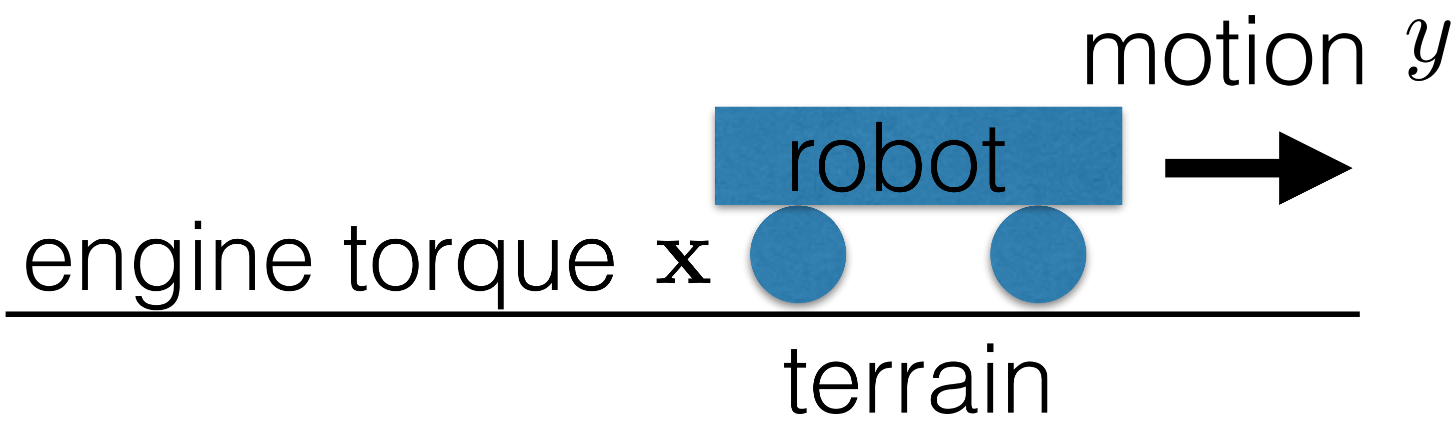
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model

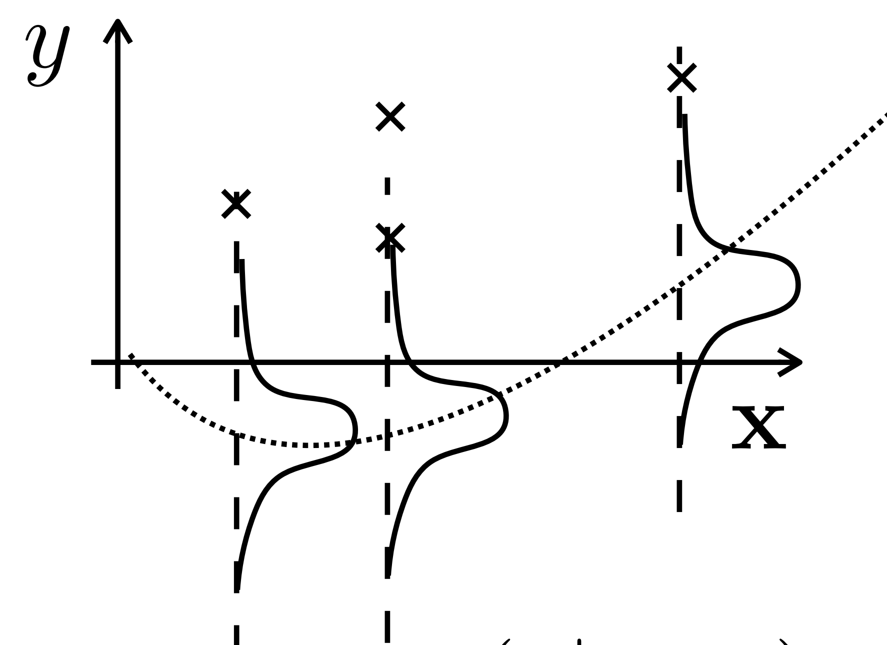




$$\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$$

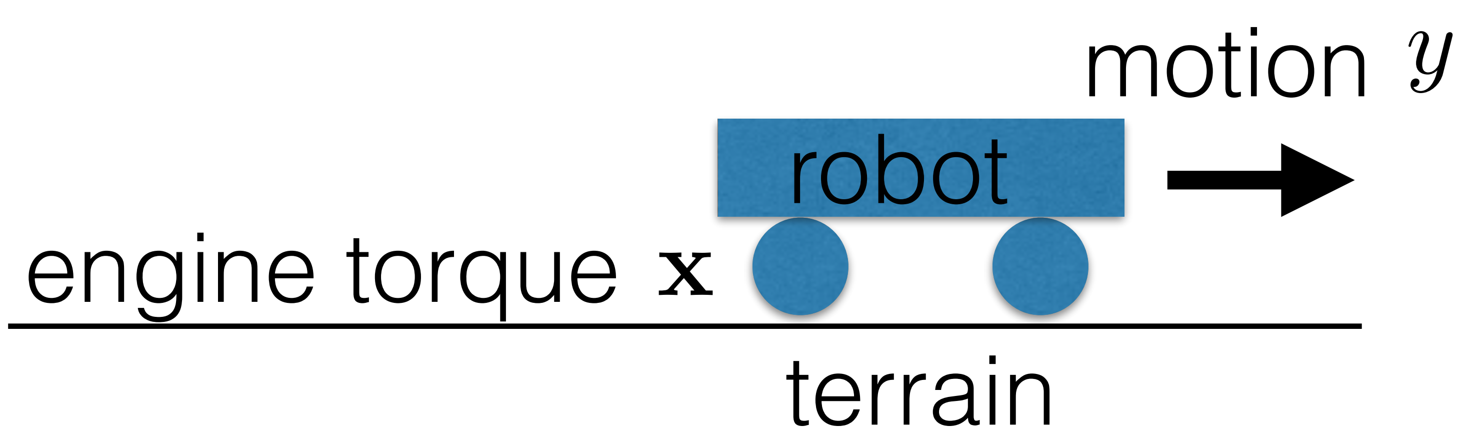
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model

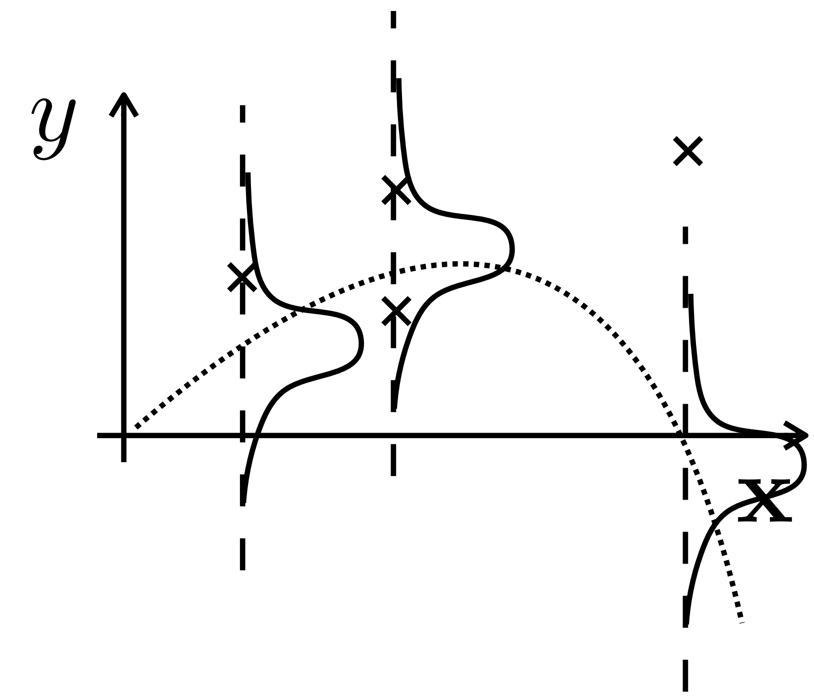




- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

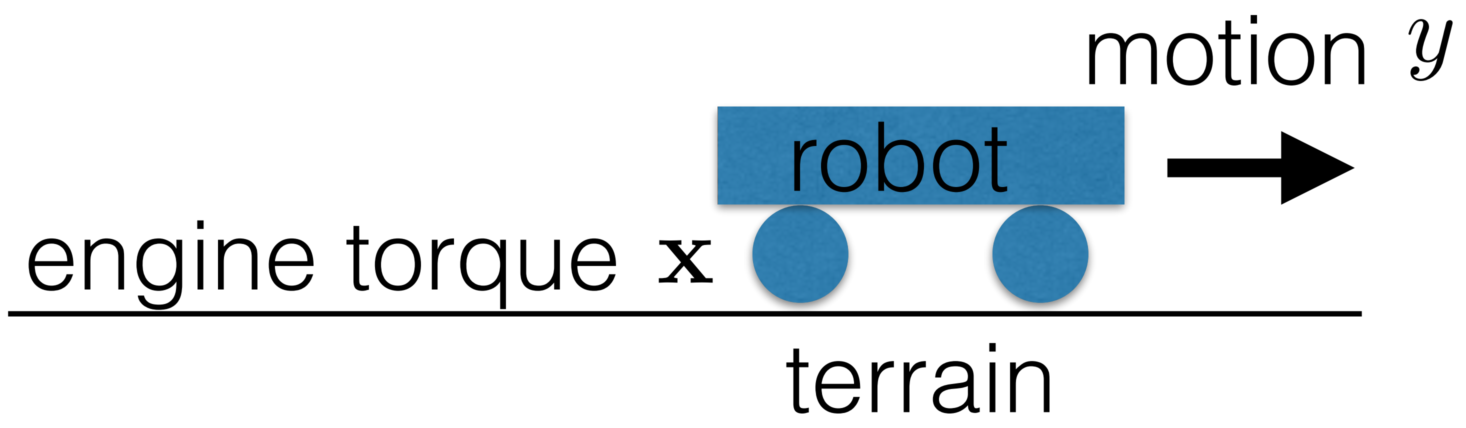
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model

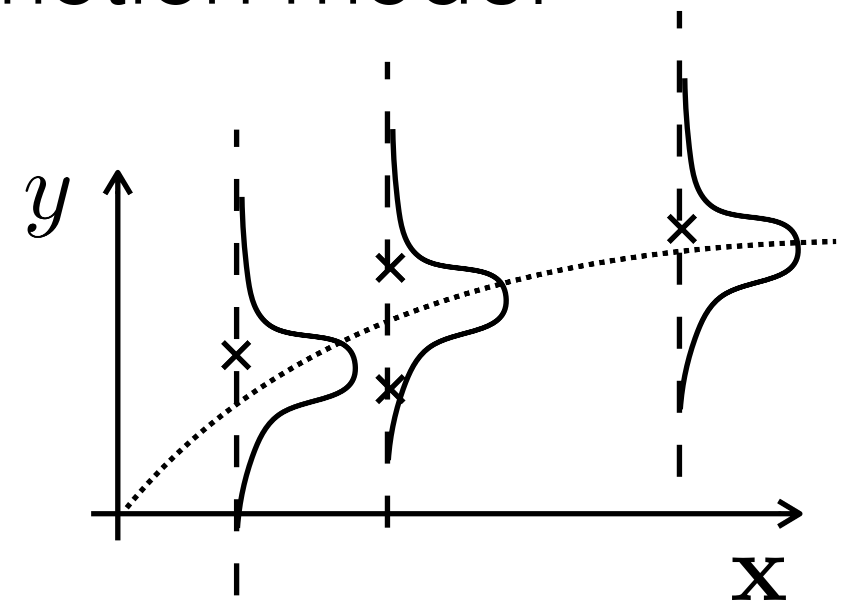




- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model





- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

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

- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x},\mathbf{w})$ given i.i.d. measurements $\mathcal{D}=\{\mathbf{x}_1,y_1\dots\mathbf{x}_N,y_N\}$

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

- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

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

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

- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

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

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

i.i.d.

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

- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \ldots \mathbf{x}_N, y_N\}$

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

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

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

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

- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

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

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

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

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

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

- We search for parameters $\mathbf{w}$ of motion model $p(y|\mathbf{x}, \mathbf{w})$ given i.i.d. measurements $\mathcal{D} = \{\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N\}$

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

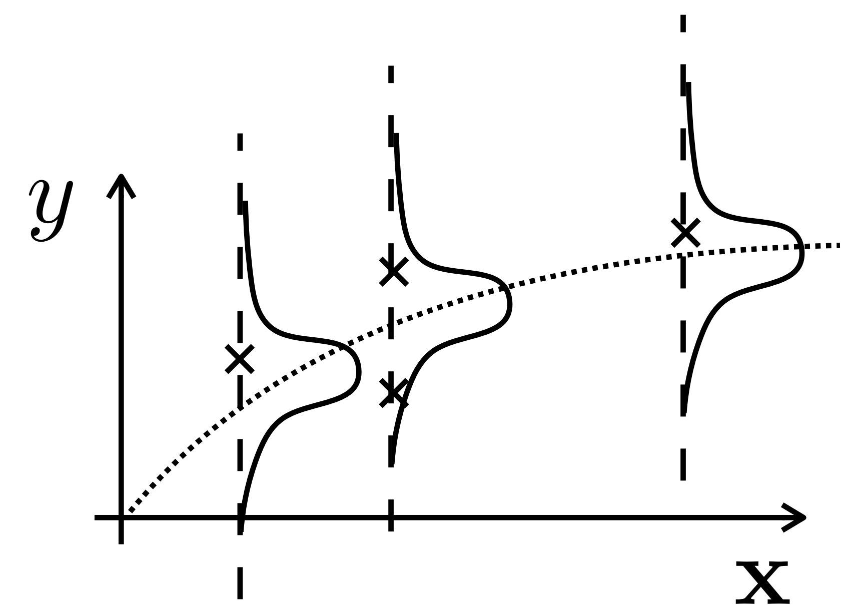
$$= \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function prior/regulariser

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

loss function prior/regulariser

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

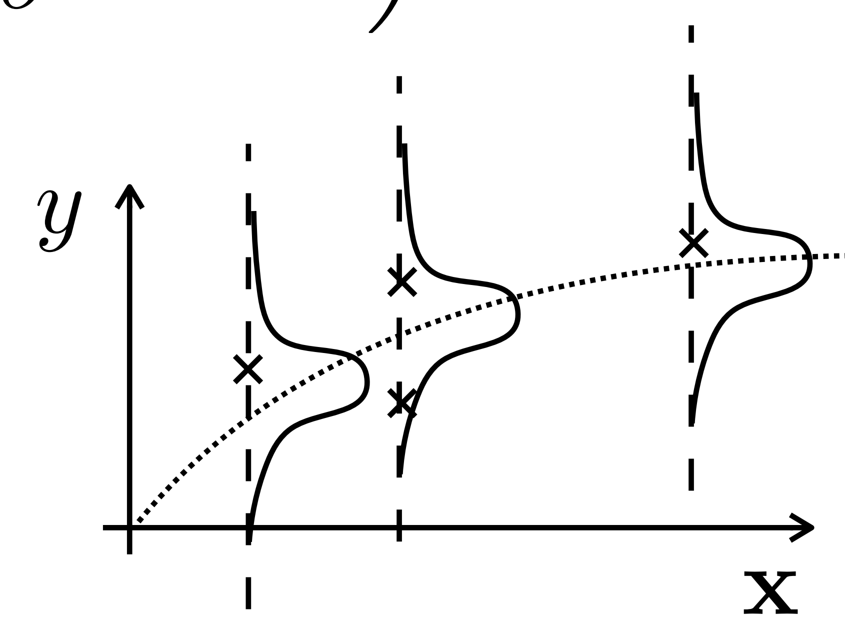


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function prior/regulariser

- **Regression:** $p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

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



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

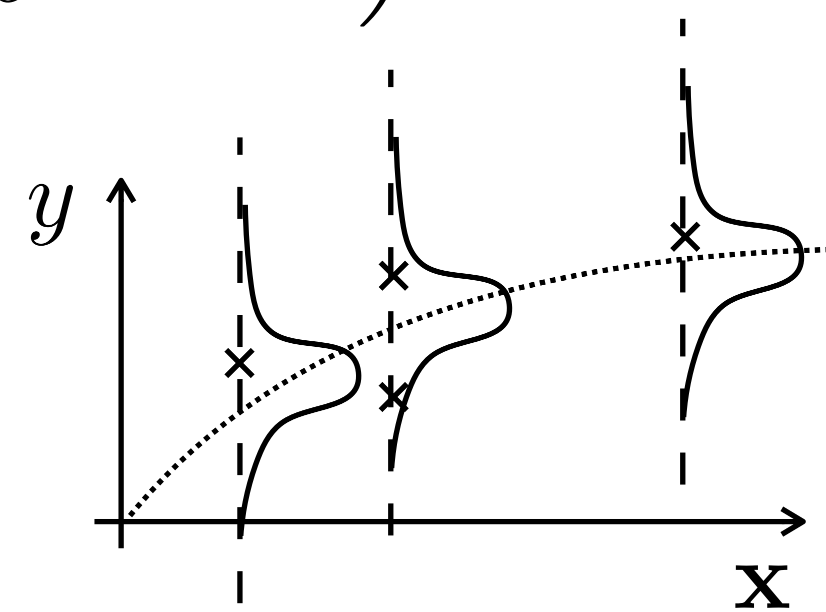
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right)$$

loss function prior/regulariser

- **Regression:** $p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

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

- Let us substitute it into the loss function (ignore prior for now)



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

loss function prior/regulariser

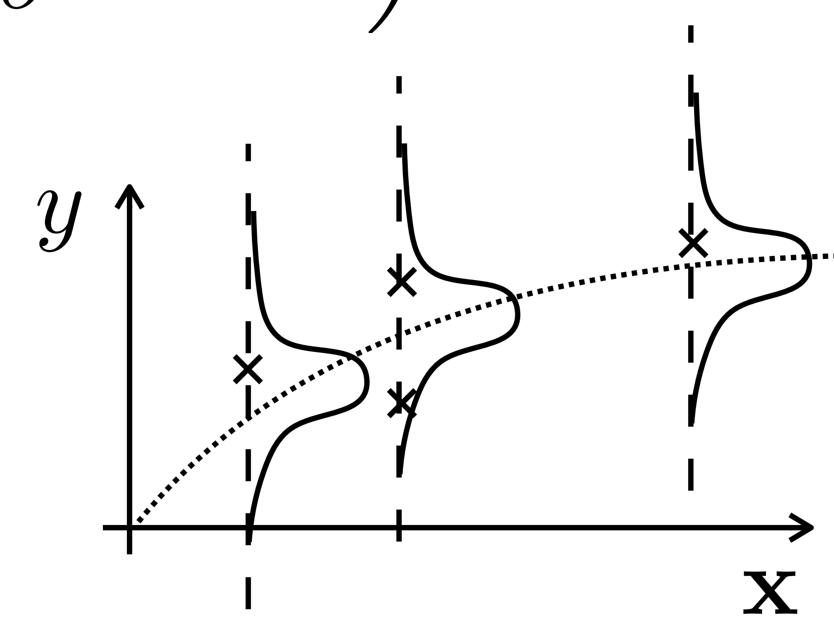
- **Regression:** $p(y|\mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

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

- which yields well known L2 loss

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

- Especially $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \overline{\mathbf{x}}$



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

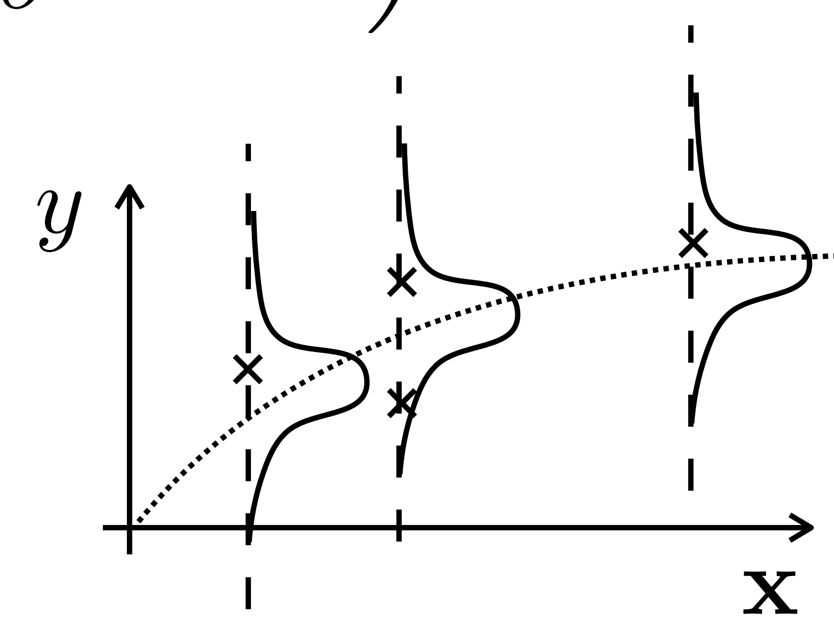
loss function prior/regulariser

- **Regression:** $p(y|\mathbf{x},\mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x},\mathbf{w}), \sigma^2)$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

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

- which yields well known L2 loss

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

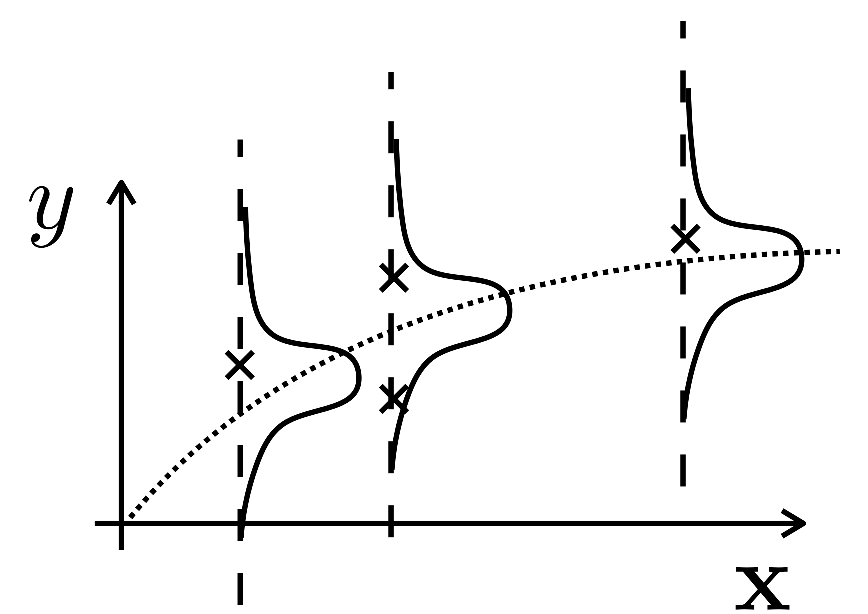


- Especially $f(\mathbf{x},\mathbf{w}) = \mathbf{w}^\top \overline{\mathbf{x}}$ yields Least squares solution

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

loss function prior/regulariser

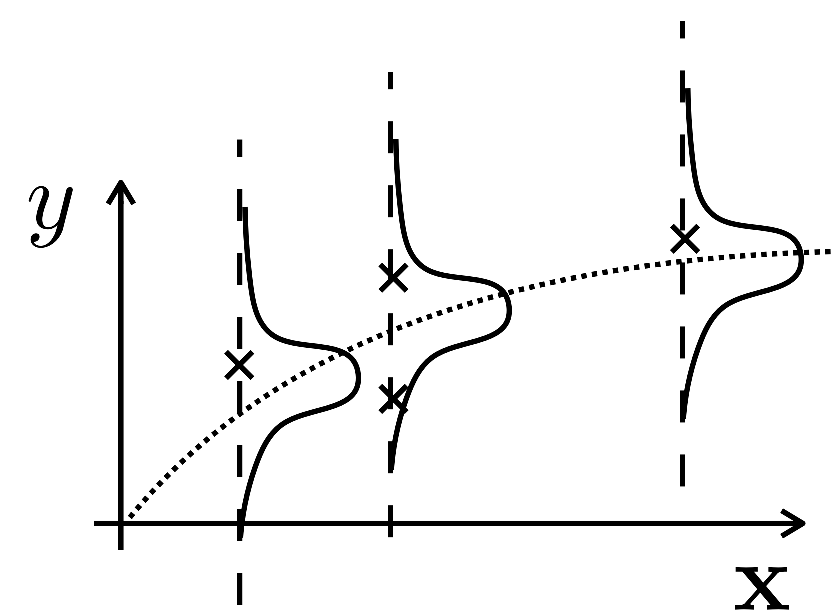
- Especially $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \overline{\mathbf{x}}$ yields Least squares solution
- What if $f(\mathbf{x}, \mathbf{w})$ is polynomial function of a certain degree?



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

loss function prior/regulariser

- Especially $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \overline{\mathbf{x}}$ yields Least squares solution
- What if $f(\mathbf{x}, \mathbf{w})$ is polynomial function of a certain degree?
- How to choose $f(\mathbf{x}, \mathbf{w})$?

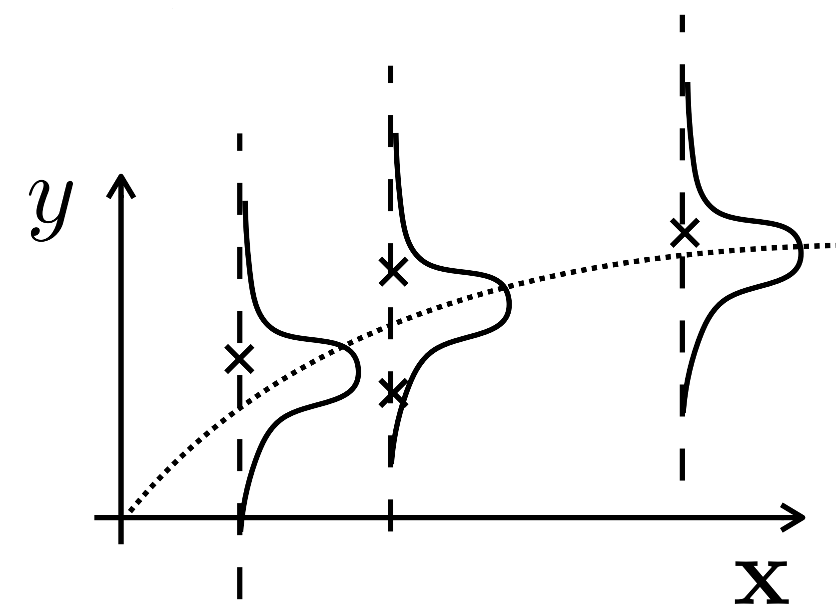


Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

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

loss function prior/regulariser

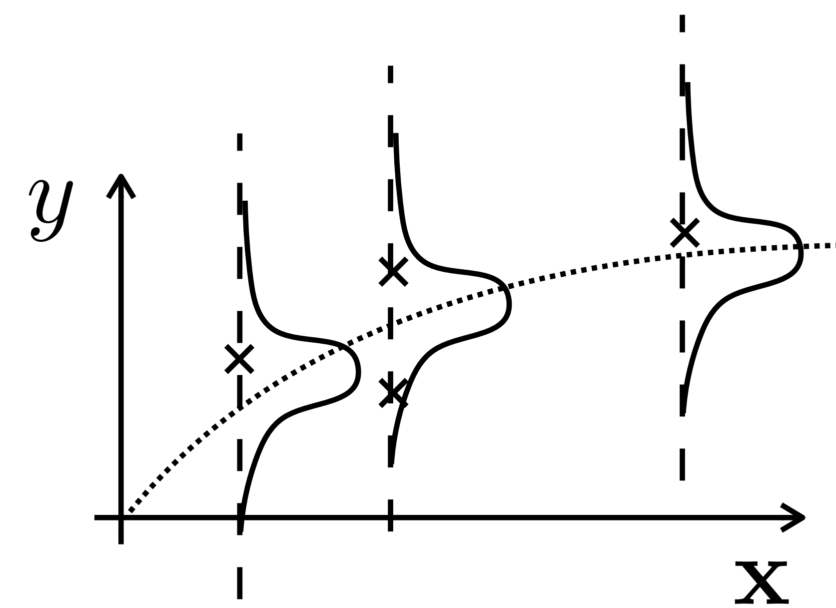
- Especially $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \overline{\mathbf{x}}$ yields Least squares solution
- What if $f(\mathbf{x}, \mathbf{w})$ is polynomial function of a certain degree?
- How to choose $f(\mathbf{x}, \mathbf{w})$?
- DKT is mapping from joint coordinates $\mathbf{x}$ to end-effector position y.



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

loss function prior/regulariser

- Especially $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \overline{\mathbf{x}}$ yields Least squares solution
- What if $f(\mathbf{x}, \mathbf{w})$ is polynomial function of a certain degree?
- How to choose $f(\mathbf{x}, \mathbf{w})$?
- DKT is mapping from joint coordinates $\mathbf{x}$ to end-effector position y.
- Why not to model it as 64-degree polynomial?

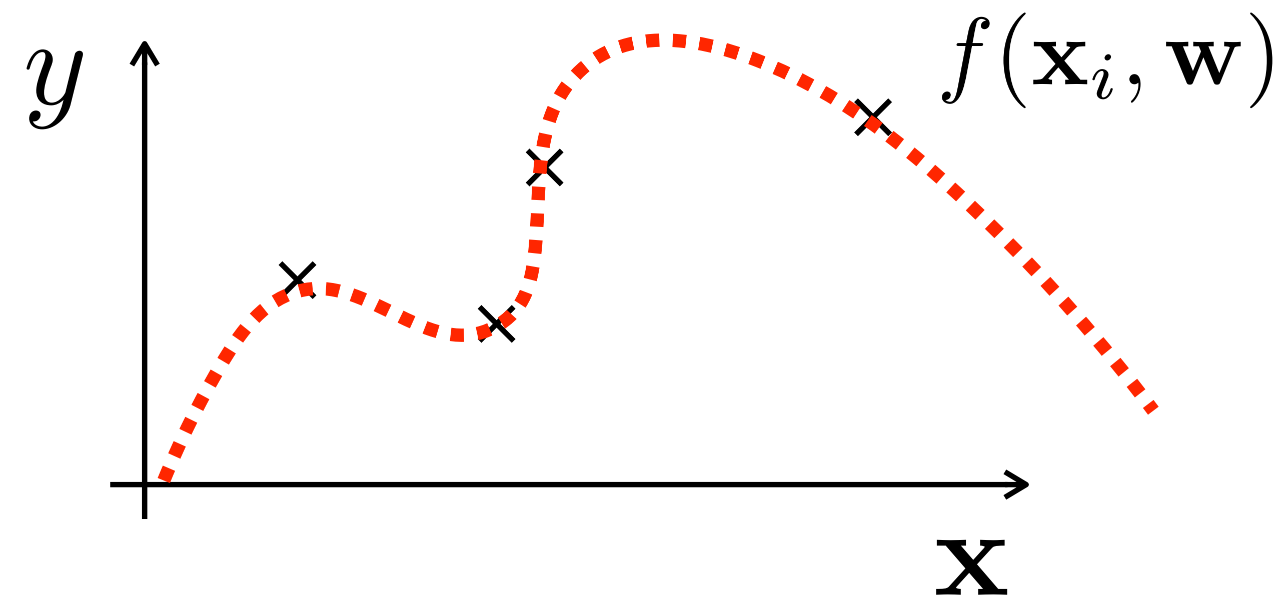


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) \cdot$$

loss function prior/regulariser

- **Prior** is important:

no prior, powerful f => overfitting

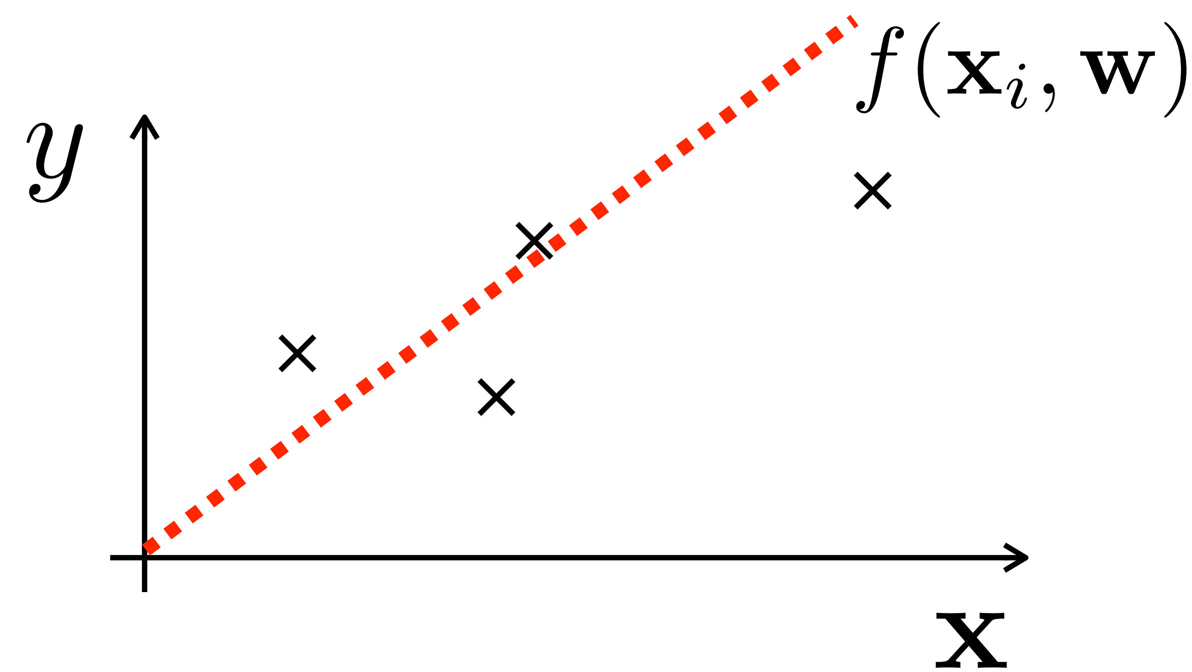


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right)$$

loss function prior/regulariser

- **Prior** is important:

no prior, simple f => underfitting

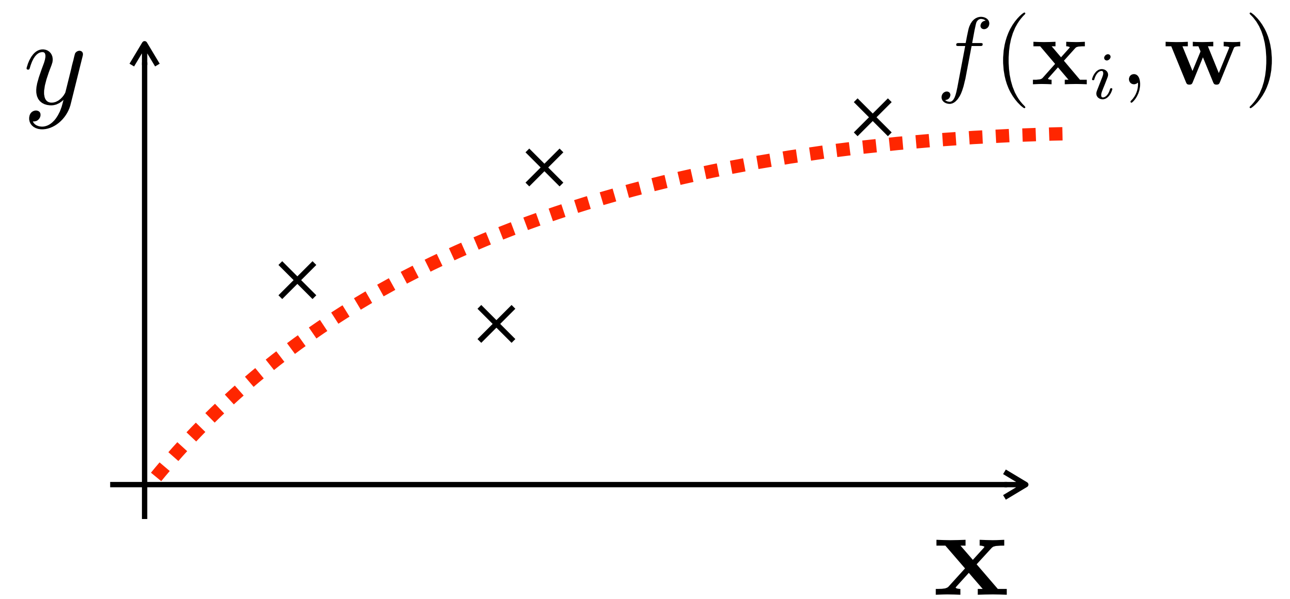


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function prior/regulariser

- **Prior** is important:

good prior



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

loss function prior/regulariser

- **Prior** is important:
    - Any prior knowledge restricts class of functions $f(\mathbf{x}_i, \mathbf{w})$ (e.g. probability of non-zero weight for higher degrees monomials is zero)

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

loss function prior/regulariser

- **Prior** is important:
  - Any prior knowledge restricts class of functions $f(\mathbf{x}_i, \mathbf{w})$ (e.g. probability of non-zero weight for higher degrees monomials is zero)
  - Gaussian prior $p(\mathbf{w}) \sim \mathcal{N}_{\mathbf{w}}(\mathbf{0}, \lambda\mathbb{I})$ yields L2 regularization (it adds eye matrix to least squares)

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

loss function prior/regulariser

- **Prior** is important:
  - Any prior knowledge restricts class of functions $f(\mathbf{x}_i, \mathbf{w})$ (e.g. probability of non-zero weight for higher degrees monomials is zero)
  - Gaussian prior $p(\mathbf{w}) \sim \mathcal{N}_{\mathbf{w}}(\mathbf{0}, \lambda\mathbb{I})$ yields L2 regularization (it adds eye matrix to least squares)
  - tuning hyperparameter λ
  - Regression with L1 regularization is known as Lasso

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

loss function prior/regulariser

- **Prior** is important:
  - Any prior knowledge restricts class of functions $f(\mathbf{x}_i, \mathbf{w})$ (e.g. probability of non-zero weight for higher degrees monomials is zero)
  - Gaussian prior $p(\mathbf{w}) \sim \mathcal{N}_{\mathbf{w}}(\mathbf{0}, \lambda\mathbb{I})$ yields L2 regularization (it adds eye matrix to least squares)
  - tuning hyperparameter λ
  - Regression with L1 regularization is known as Lasso
  - Well chosen prior partially reduces overfitting
  - Occam's Razor

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

loss function

prior/regulariser

William of Ockham (1287-1347)

leprechauns can be involved in any explanation

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

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

loss function prior/regulariser

- It is very important to avoid any "*not-well justified leprechauns*" in the model, otherwise any learning (parameter estimations) may suffer from too complex explanations => overfitting

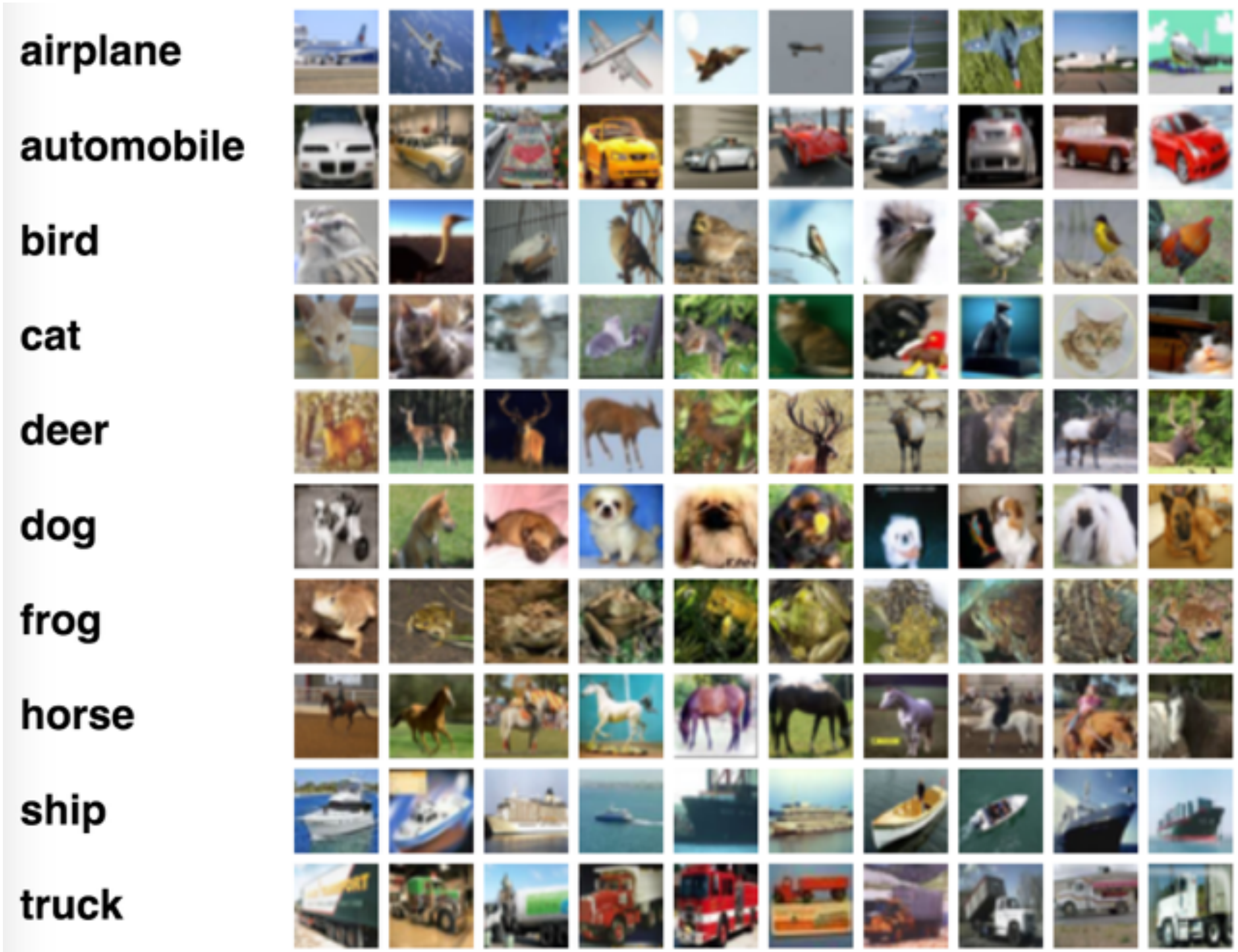
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i - \log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function prior/regulariser

- It is very important to avoid any "*not-well justified leprechauns*" in the model, otherwise any learning (parameter estimations) may suffer from too complex explanations => overfitting
- Consequently we study different phenomenas
  - animal cortex structure (for ConvNets)
  - geometry of rigid motion (for robot/scene motion or DKT)
  - projective transformation of pinhole cameras

  to create as simple (i.e.leprechauns-free) model as possible

Recognition problem



Why is it hard?

CIFAR-10: classify 32x32 RGB images into 10 categories
https://www.cs.toronto.edu/~kriz/cifar.html

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

Recognition problem

Why it is hard? **Huge within-class variability !**

- Viewpoint
- Occlusion
- Illumination
- Pose
- Type
- Context

Timofte, Zimmermann, van Gool, Multivew traffic-sign detection, recognition and 3D localisation, MVA, 2014
https://link.springer.com/content/pdf/10.1007/s00138-011-0391-3.pdf

Recognition problem

Why it is hard? **Huge within-class variability !**

- Viewpoint
- Occlusion
- Illumination
- Pose
- Type
- Context

Recognition problem

Why it is hard? **Huge among-class similarity!**

- Viewpoint
- Occlusion
- Illumination
- Pose
- Type
- Context

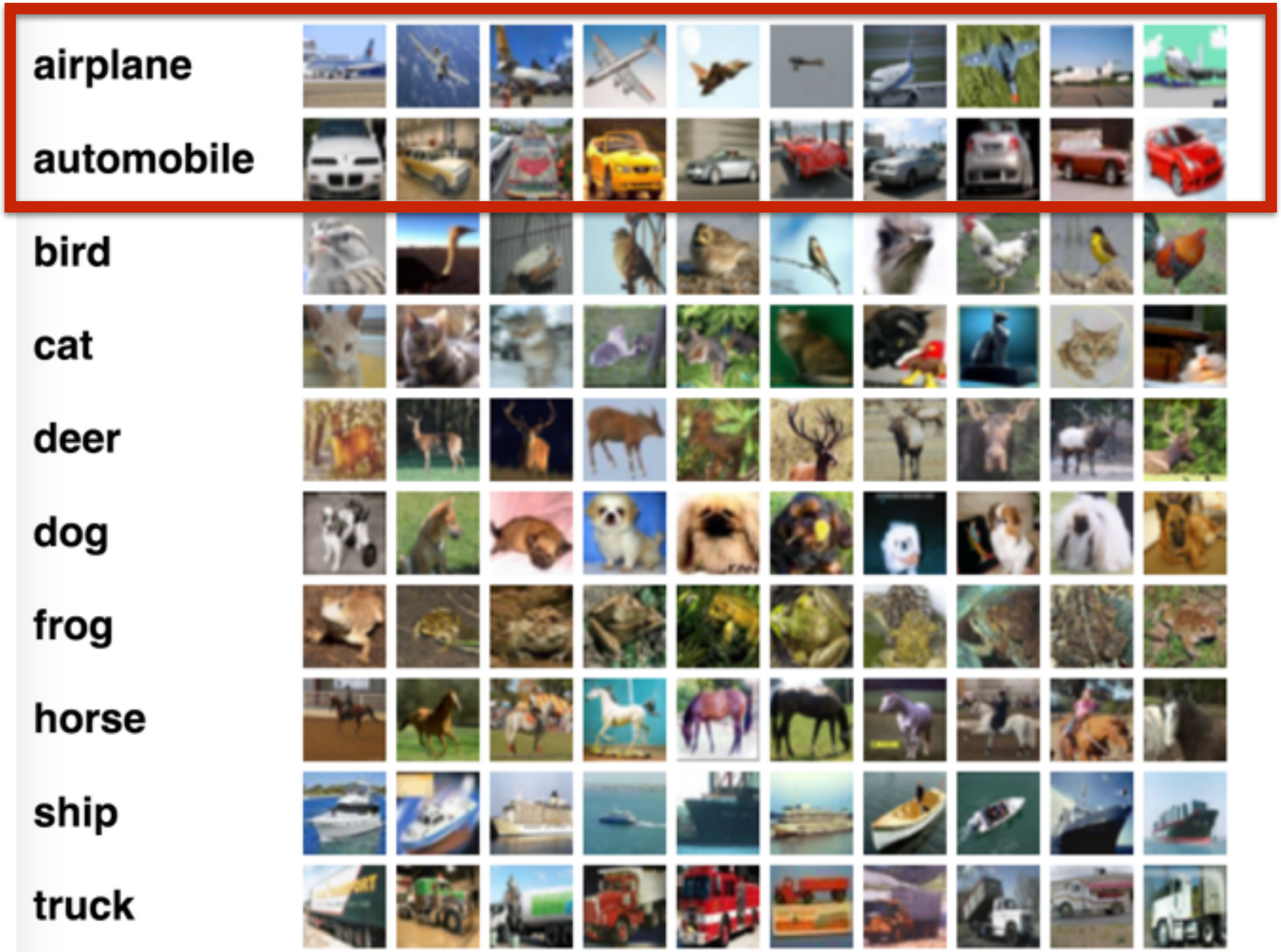
Timofte, Zimmermann, van Gool, Multivew traffic-sign detection, recognition and 3D localisation, MVA, 2014 https://link.springer.com/content/pdf/10.1007/s00138-011-0391-3.pdf

Recognition problem

Why it is hard? **Huge among-class similarity!**

- Viewpoint
- Occlusion
- Illumination
- Pose
- Type
- Context

Recognition problem



CIFAR-10: classify 32x32 RGB images into 10 categories

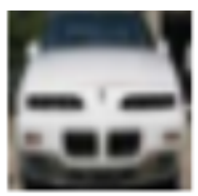
https://www.cs.toronto.edu/~kriz/cifar.html

Labels (y_i) RGB images ($\mathbf{x}_i$)

airplane

automobile

Two-class recognition problem: classify airplane/automobile

def classify():

  ???

  return p

Probability of image being from the class airplane

How to model it?

Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Classification

We model probability of image $\mathbf{x}$ being label +1 or -1 as

$$p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$$

Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Classification

We model probability of image $\mathbf{x}$ being label +1 or -1 as

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y = +1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y = -1 \end{cases}$$

where

$$\sigma(f(\mathbf{x},\mathbf{w})) = \frac{1}{1+\exp(-f(\mathbf{x},\mathbf{w}))}$$

is sigmoid function.

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

Classification

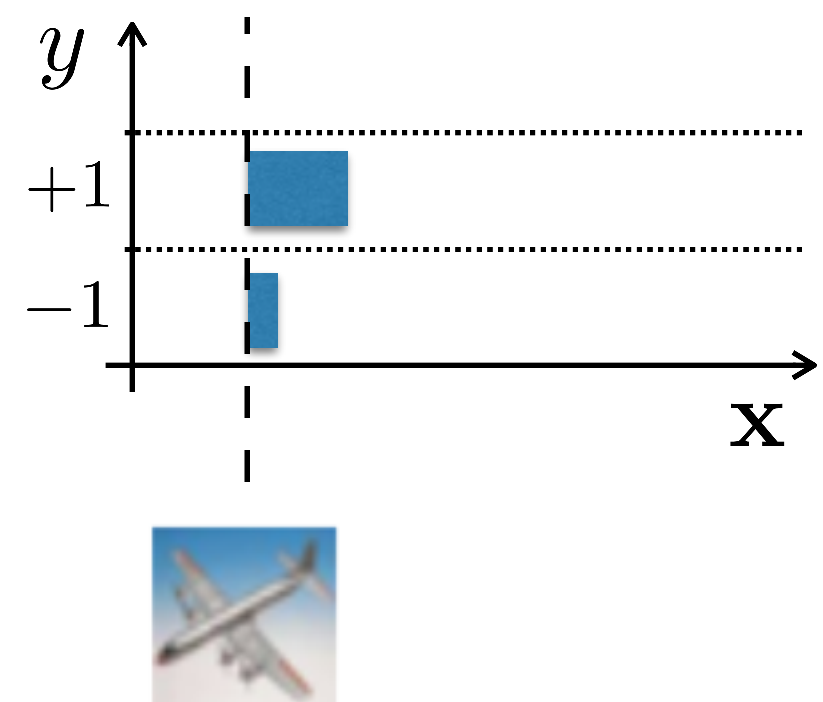
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

We model probability of image $\mathbf{x}$ being label +1 or -1 as

$$p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$$



Labels (y_i) RGB images ($\mathbf{x}_i$)

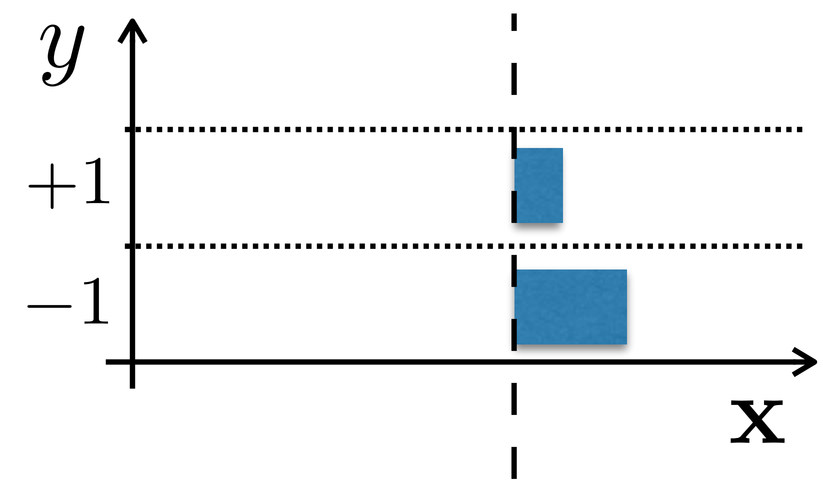
$+1$

-1

Classification

We model probability of image $\mathbf{x}$ being label +1 or -1 as

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x},\mathbf{w})) & y = -1 \end{cases}$$



Labels (y_i) RGB images ($\mathbf{x}_i$)

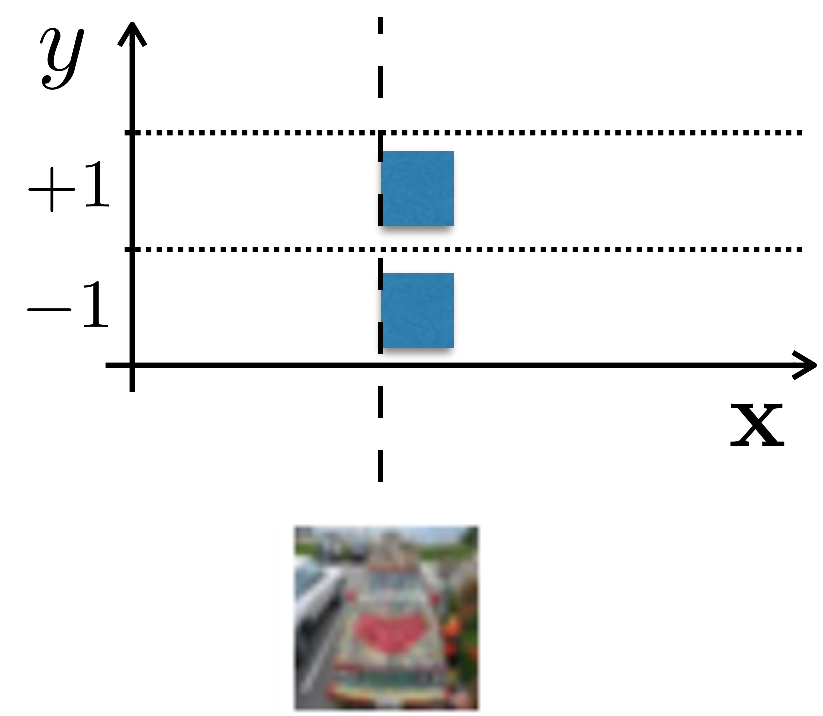
$+1$

-1

Classification

We model probability of image $\mathbf{x}$ being label +1 or -1 as

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \end{cases}$$



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

Labels (y_i) RGB images ($\mathbf{x}_i$)

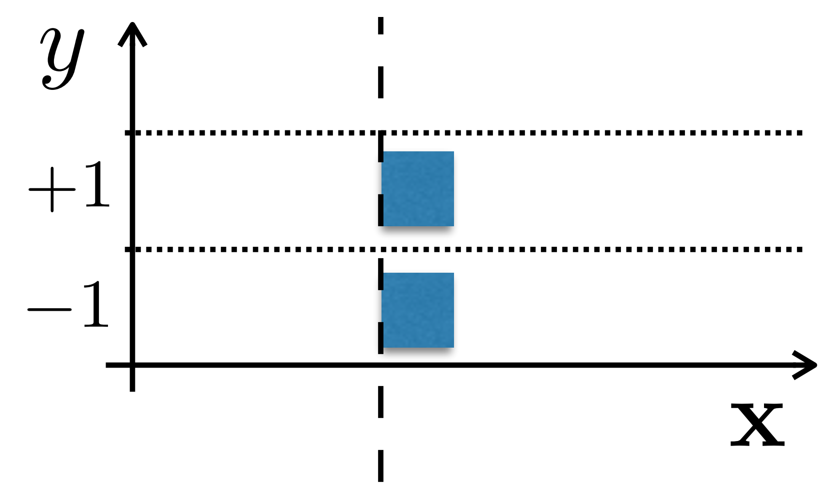
$+1$

-1

Classification

We model probability of image $\mathbf{x}$ being label +1 or -1 as

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y = +1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y = -1 \end{cases}$$



Linear classifier model probability of being from class +1 as $p = \sigma\left(\mathbf{w}^\top \overline{\mathbf{x}}\right)$

What is dimensionality of $\mathbf{x}$ and $\mathbf{w}$?

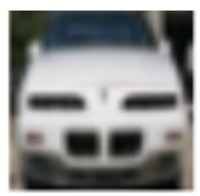
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

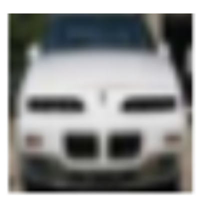
-1

Classification

Example: Linear classifier

def classify():

 # Linear classifier

$\mathbf{x} = \text{vec}($  $)$

$p = \sigma\left(\mathbf{w}^\top \overline{\mathbf{x}}\right)$

return p

$$\mathbf{w}^\top \overline{\mathbf{x}} = 2.5$$

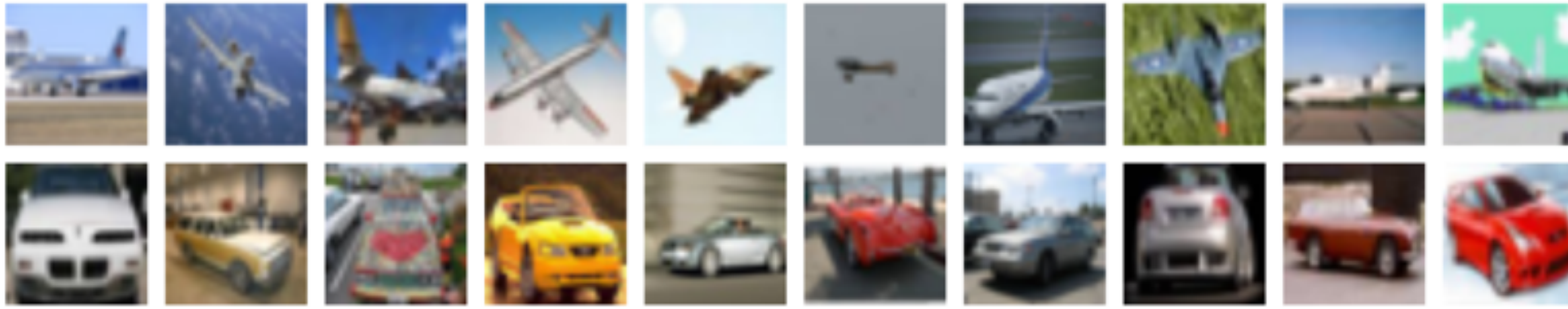
Classification

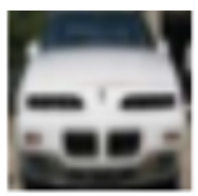
Example: Linear classifier

Labels (y_i) RGB images ($\mathbf{x}_i$)

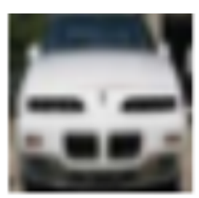
$+1$

-1



def classify():

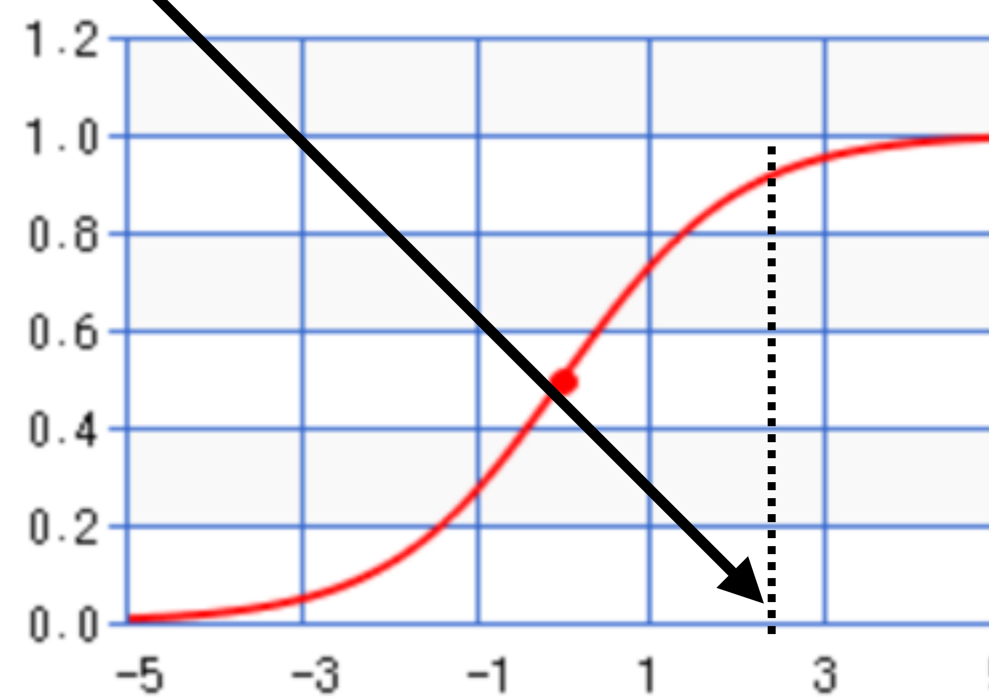
 # Linear classifier

$\mathbf{x} = \text{vec}($  $)$

$p = \sigma\left(\mathbf{w}^\top \overline{\mathbf{x}}\right)$

return p

$\mathbf{w}^\top \overline{\mathbf{x}} = 2.5$



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

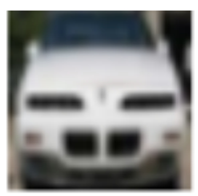
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

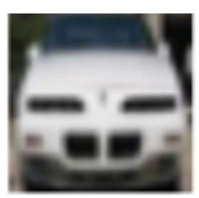
-1

Classification

Example: Linear classifier

def classify():

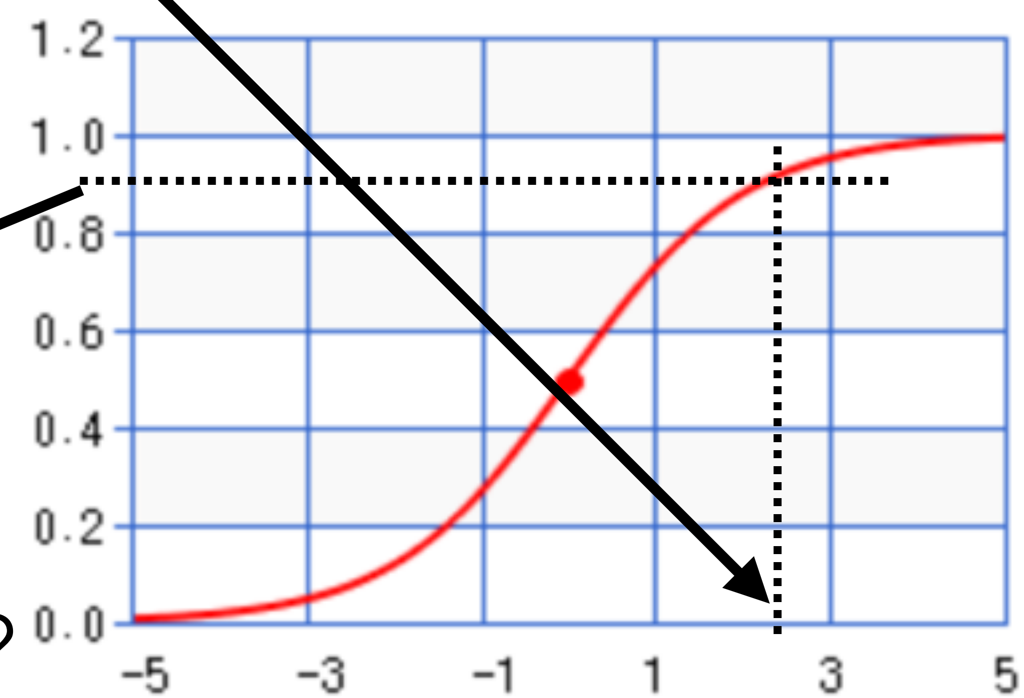
 # Linear classifier

$\mathbf{x} = \text{vec}($  $)$

$p = \sigma\left(\mathbf{w}^\top \overline{\mathbf{x}}\right)$

return $p = \sigma(2.5) = 0.92$

$\mathbf{w}^\top \overline{\mathbf{x}} = 2.5$



is it a good classifier?

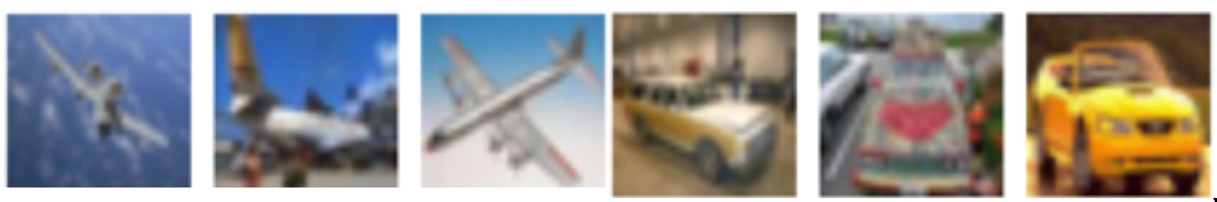
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Training = search for unknown parameters $\mathbf{w}$ which fits a given data

def train(

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

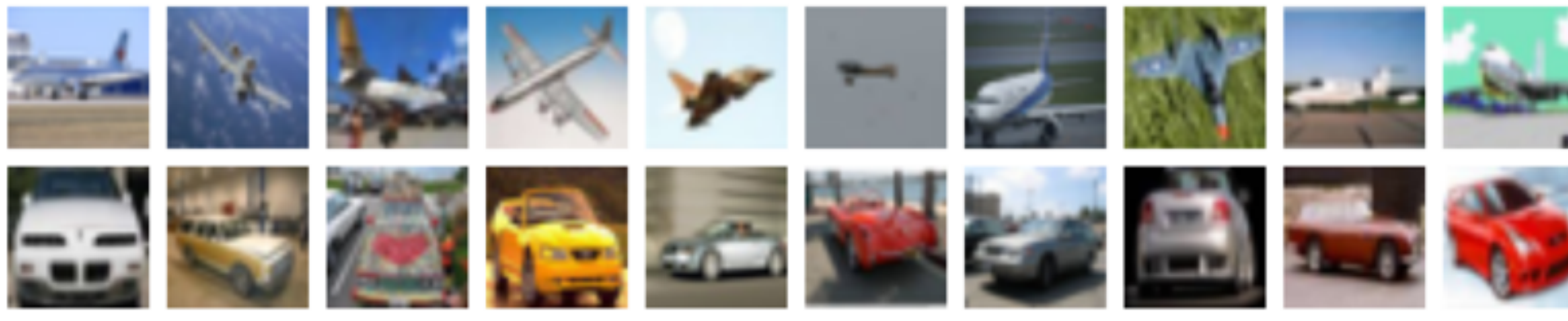
???

return $\mathbf{w}^*$

Labels (y_i)   RGB images ($\mathbf{x}_i$)

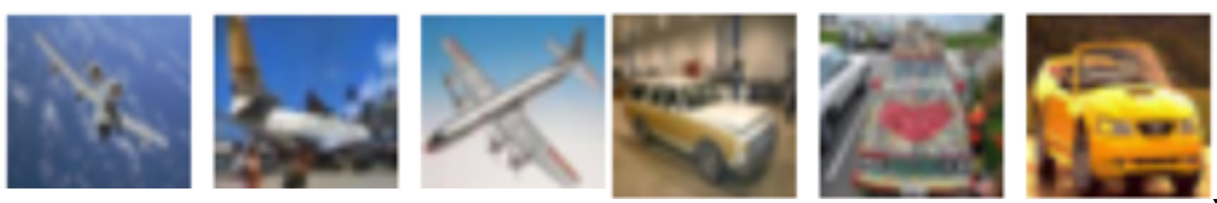
$+1$

-1



Training

Training = search for unknown parameters $\mathbf{w}$ which fits a given data

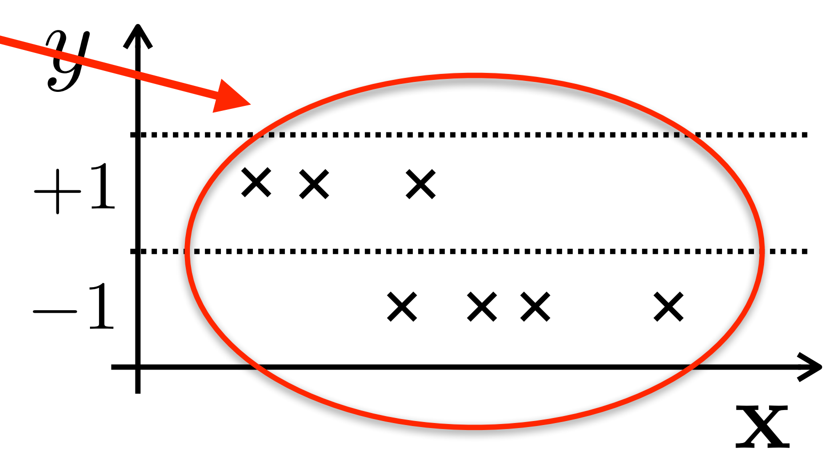
def train(

$+1$ $+1$ $+1$ -1 -1 -1):

???

return $\mathbf{w}^*$

Training data



Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

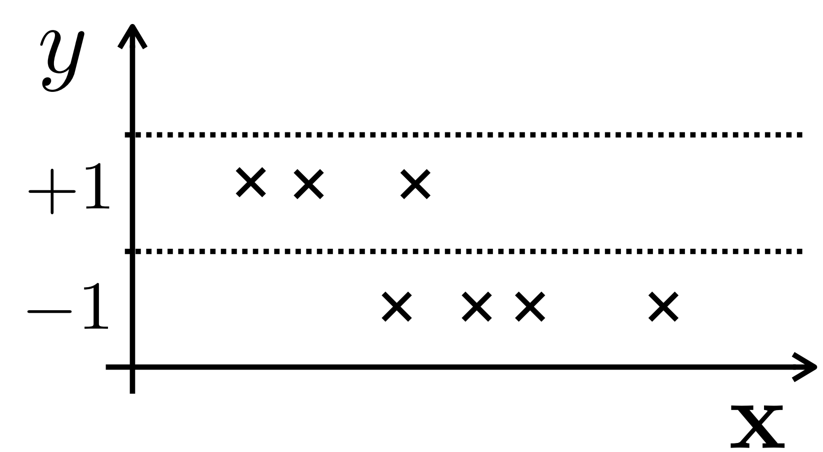
-1

Training

Training = search for unknown parameters $\mathbf{w}$ which fits a given data

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \end{cases}$$

Training data



Labels (y_i) RGB images ($\mathbf{x}_i$)

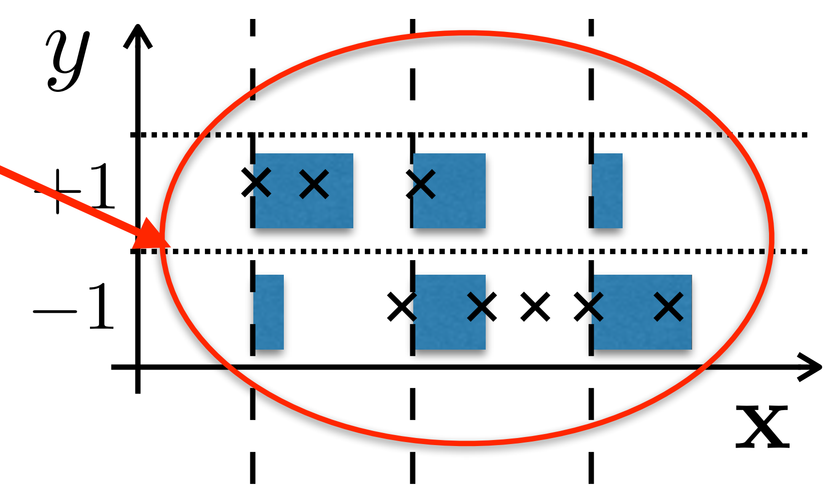
$+1$

-1

Training

Training = search for unknown parameters $\mathbf{w}$ which fits a given data

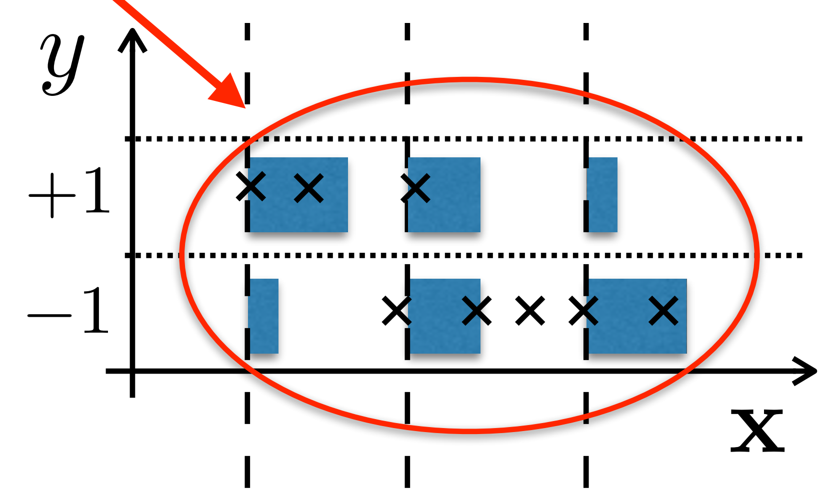
$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \end{cases}$$



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

loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$

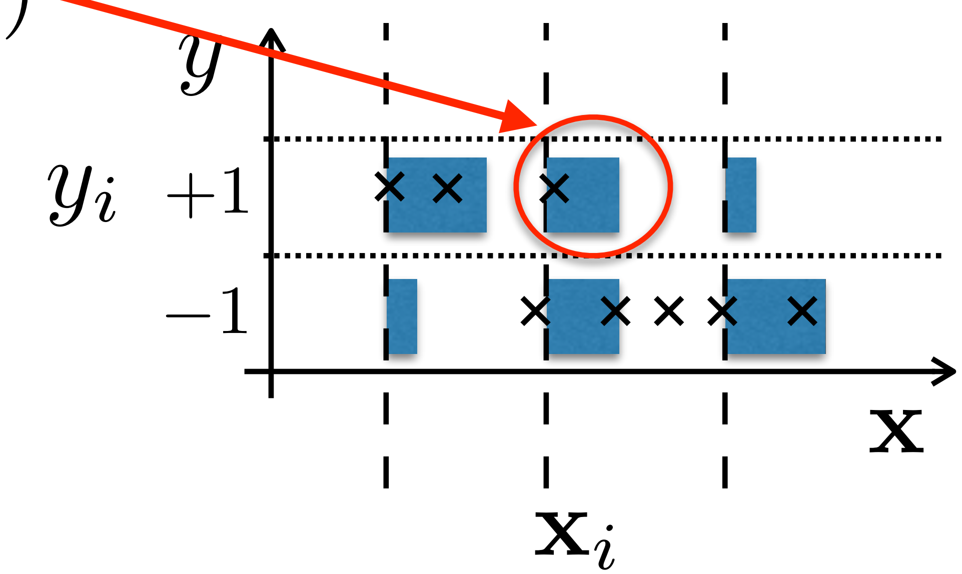


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) \cdot \boxed{}$$

loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \sigma(y_i\, f(\mathbf{x}_i, \mathbf{w}))$$



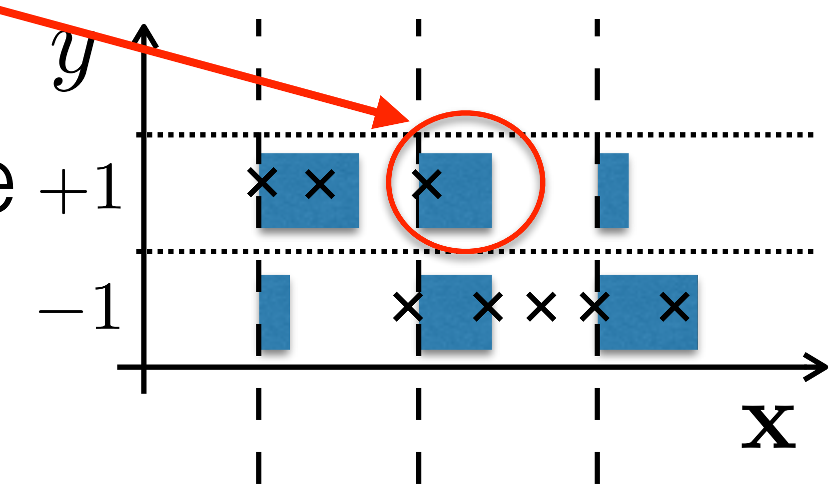
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right)$$

loss function | prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \sigma(y_i\, f(\mathbf{x}_i, \mathbf{w}))$$

- how to find distribution which maximize probability of training data?

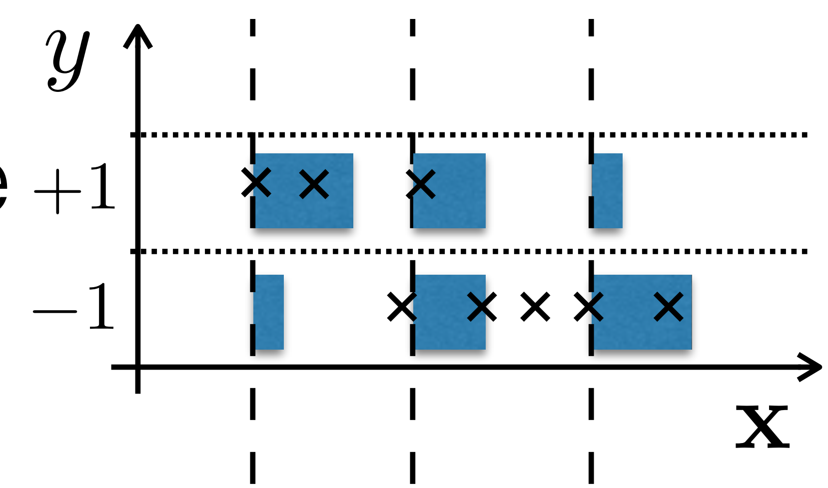


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) \cdot$$

loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Probability of observing y_i when measuring $\mathbf{x}_i$ is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \sigma(y_i\, f(\mathbf{x}_i, \mathbf{w}))$$



- how to find distribution which maximize probability of training data?
- substitution yields logistic loss

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, f(\mathbf{x}_i, \mathbf{w}))\right]$$

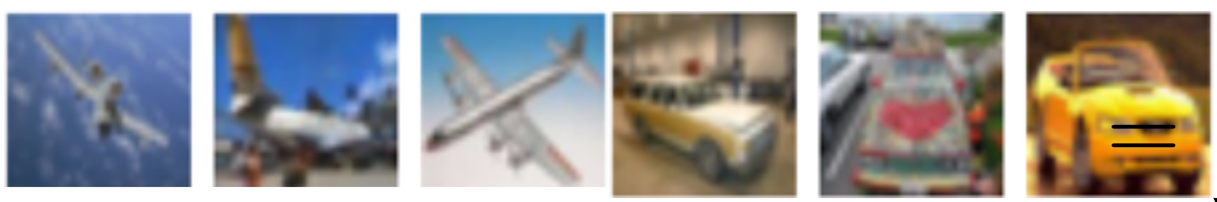
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$\mathbf{x}_i = \text{vec}($  $) \quad \forall_i$

return $\mathbf{w}^*$

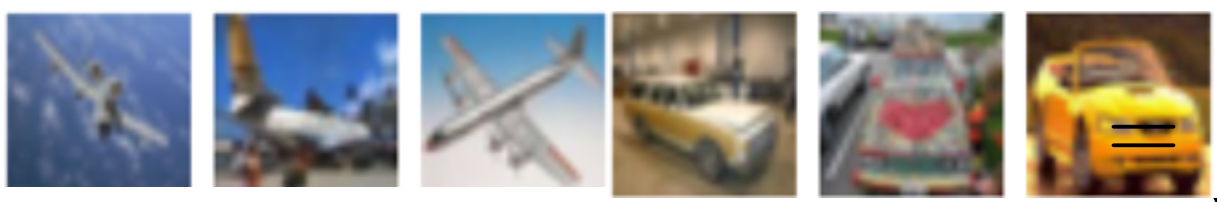
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$$\mathbf{x}_i = \text{vec}(\quad)\quad \forall i$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

Training

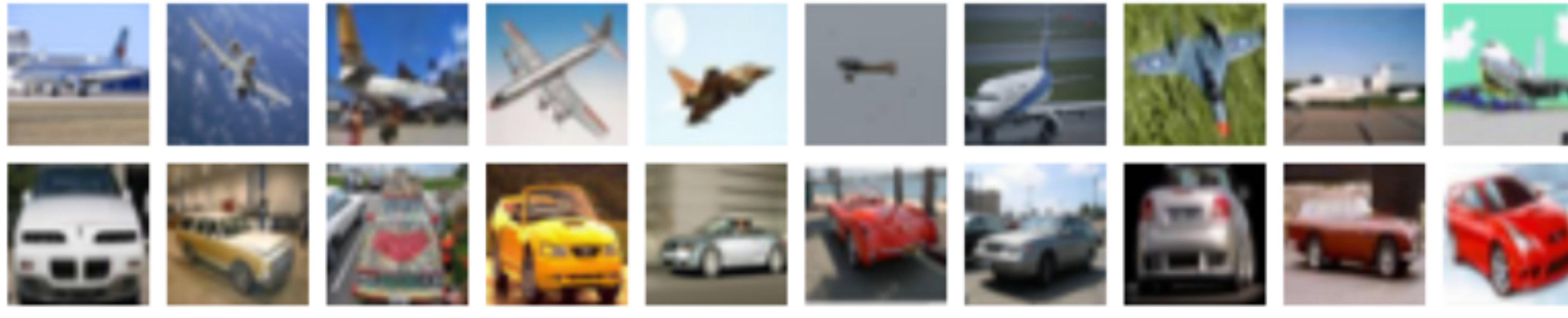
Example: Training linear classifier

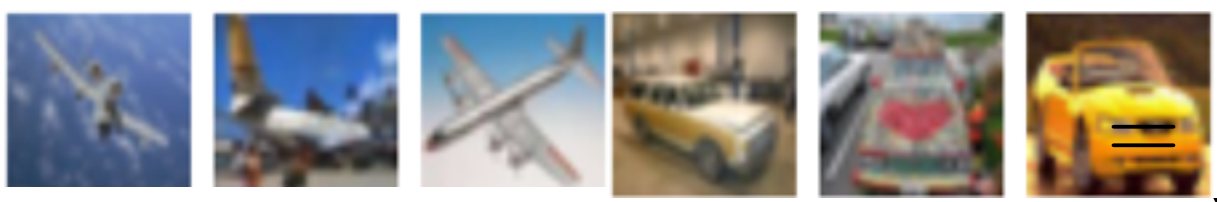
Labels (y_i)

RGB images ($\mathbf{x}_i$)

$+1$

-1



def train(

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$$\mathbf{x}_i = \text{vec}(\ \text{<image>}\ ) \quad \forall i$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

-2.5

Small $\mathbf{w}^\top \overline{\mathbf{x}}_i$

while $y_i = -1$

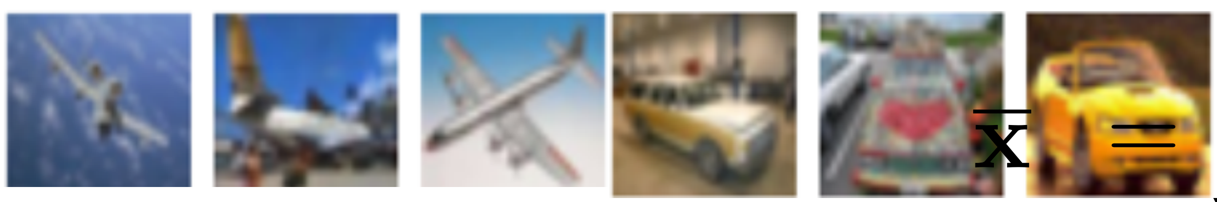
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train( $\overline{\mathbf{x}} =$

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$\mathbf{x}_i = \text{vec}($  $) \quad \forall_i$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

$-(-1) \times (-2.5)$

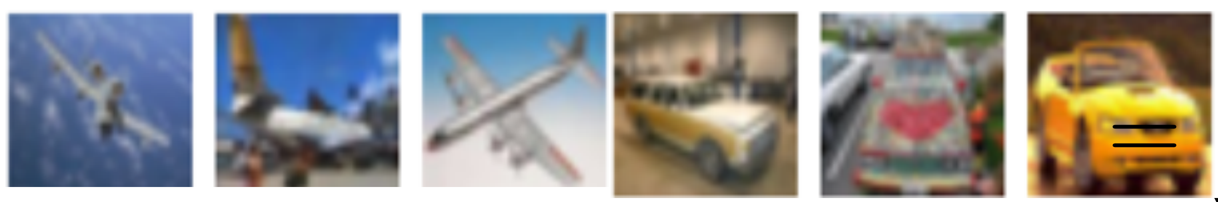
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$\mathbf{x}_i = \text{vec}($  $) \quad \forall_i$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

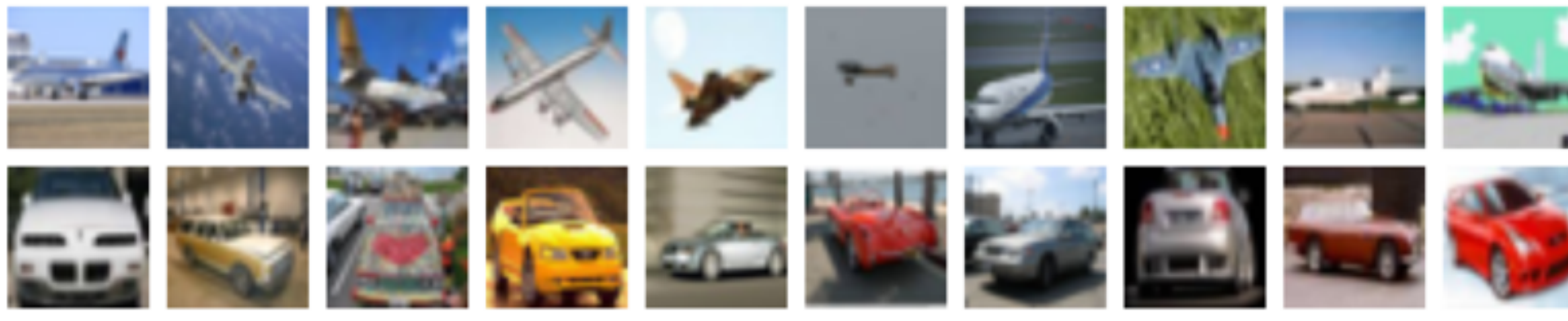
0.03

Small loss for for small $\mathbf{w}^\top \overline{\mathbf{x}}_i$ while $y_i = -1$

Labels (y_i) RGB images ($\mathbf{x}_i$)

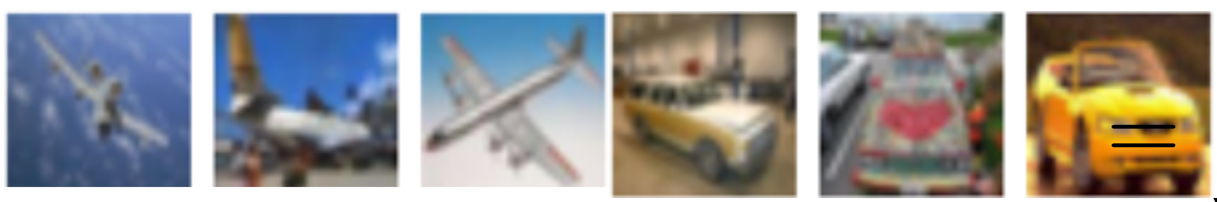
$+1$

-1



Training

Example: Training linear classifier

def train(

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$\mathbf{x}_i = \text{vec}($  $) \quad \forall_i$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

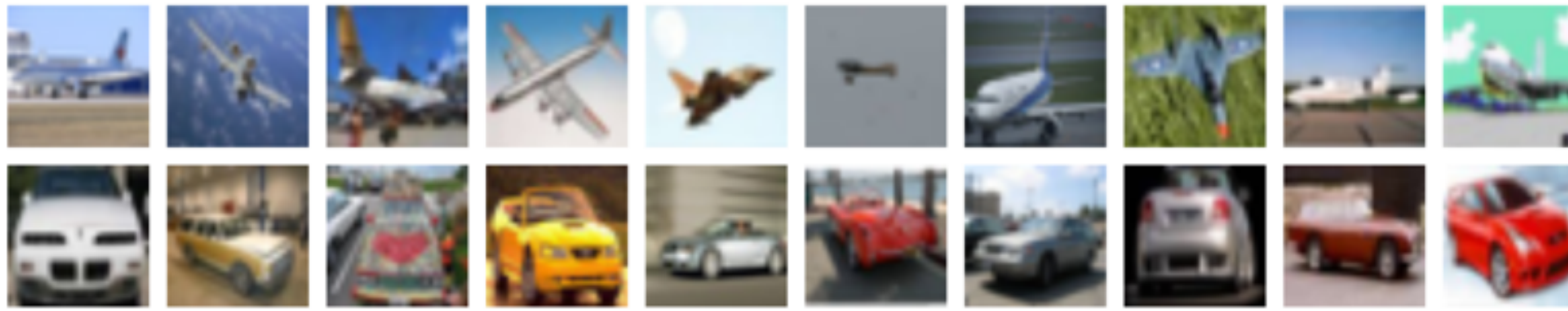
2.5

Large $\mathbf{w}^\top \overline{\mathbf{x}}_i$ while $y_i = -1$

Labels (y_i) RGB images ($\mathbf{x}_i$)

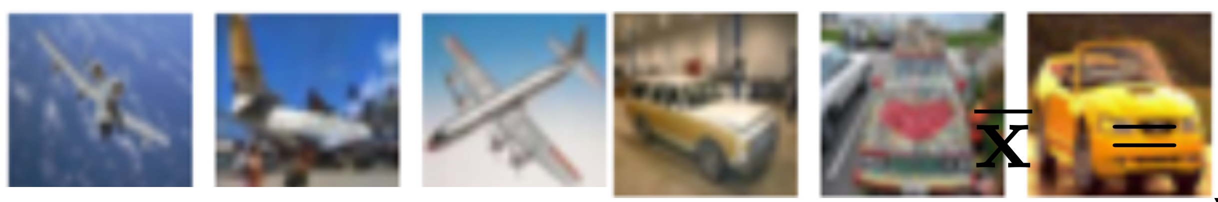
$+1$

-1



Training

Example: Training linear classifier

def train( $\overline{\mathbf{x}} =$

$+1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1$):

$\mathbf{x}_i = \mathrm{vec}($  $) \quad \forall i$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

$$-(-1) \times 2.5$$

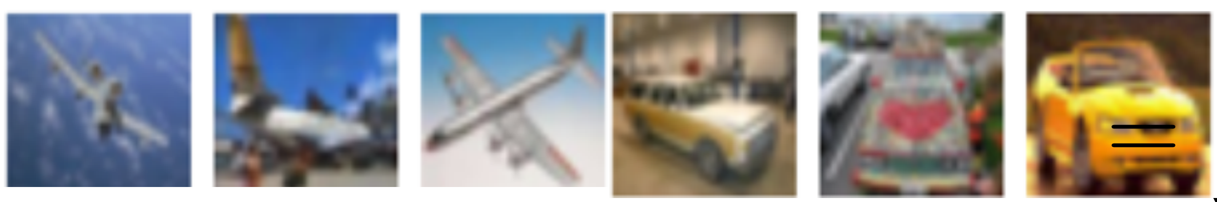
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(
$+1$ $+1$ $+1$ -1 -1 -1):

$\mathbf{x}_i = \mathrm{vec}($  $) \quad \forall_i$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

1.12

Huge loss for large $\mathbf{w}^\top \overline{\mathbf{x}}_i$ while $y_i = -1$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underbrace{\sum_i \log\left[1 + \exp(y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]}_{\mathcal{L}(\mathbf{w})}$$

- There is no closed-form solution
- Gradient optimization

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underbrace{\boxed{\sum_i \log\left[1 + \exp(y_i\,\mathbf{w}^\top \overline{\mathbf{x}}_i)\right]}}_{\mathcal{L}(\mathbf{w})}$$

- There is no closed-form solution
- Gradient optimization

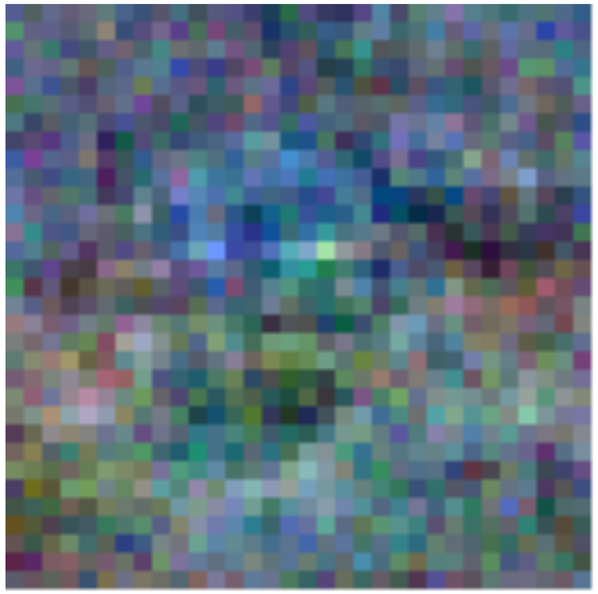
$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \quad \text{where } \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \frac{y_i \overline{\mathbf{x}}_i}{1 + \exp(-y_i \mathbf{w}^\top \overline{\mathbf{x}}_i)}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underbrace{\sum_i \log\left[1 + \exp(y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]}_{\mathcal{L}(\mathbf{w})}$$

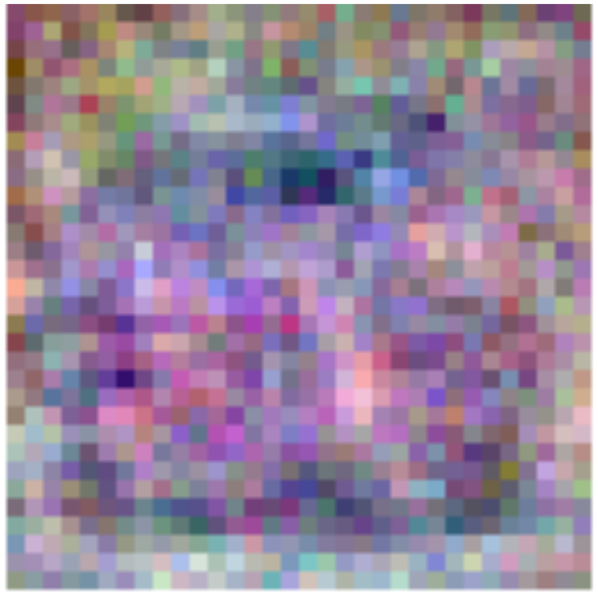
- There is no closed-form solution
- Gradient optimization

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \quad \text{where} \quad \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \frac{y_i \overline{\mathbf{x}}_i}{1 + \exp(-y_i \mathbf{w}^\top \overline{\mathbf{x}}_i)}$$

Learned weights as a template:



airplane



automobile

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

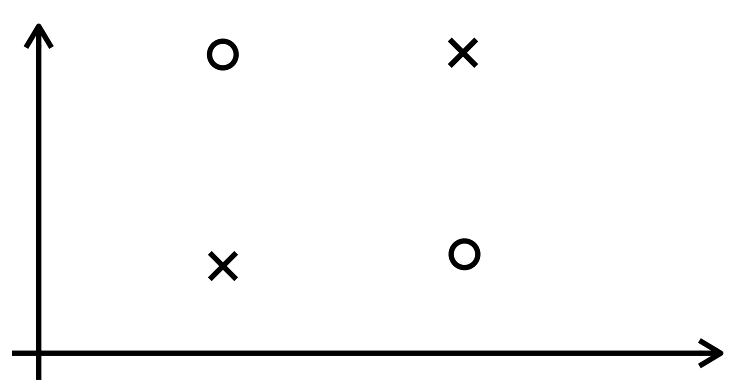
loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Choice of $f(\mathbf{x}, \mathbf{w})$ is crucial

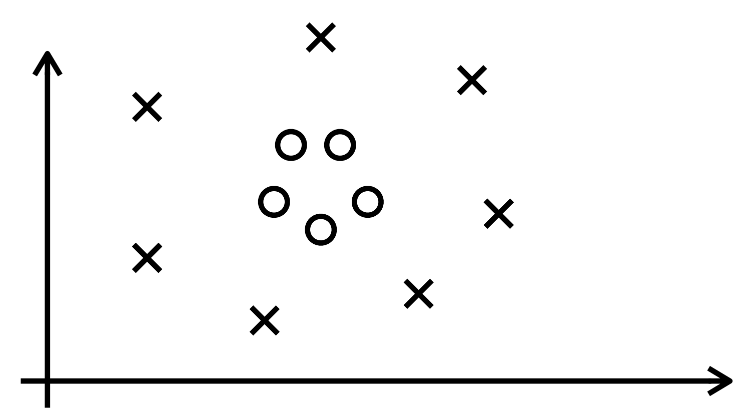
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i - \log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function   prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Linear $f(\mathbf{x}, \mathbf{w})$ cannot generate wild decision boundary



XOR



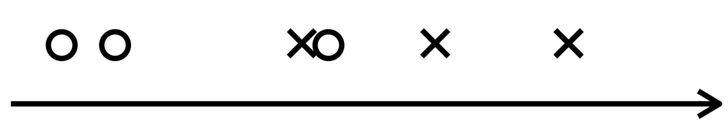
circle

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

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

loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Wild $f(\mathbf{x}, \mathbf{w})$ with high-dimensional $\mathbf{w}$ suffers from the curse of dimensionality and overfitting

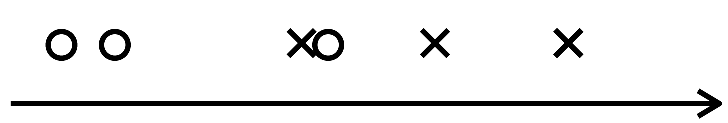


1D case

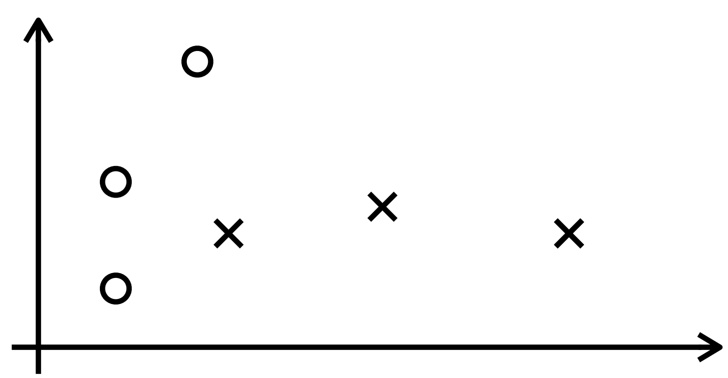
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Wild $f(\mathbf{x}, \mathbf{w})$ with high-dimensional $\mathbf{w}$ suffers from the curse of dimensionality and overfitting



1D case

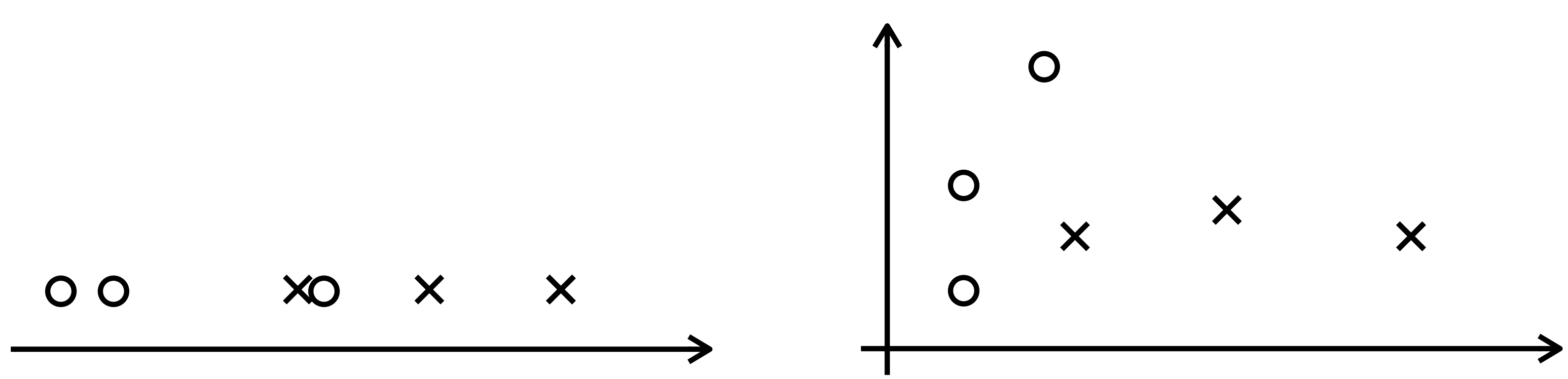


2D case

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

loss function prior/regulariser

- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Wild $f(\mathbf{x}, \mathbf{w})$ with high-dimensional $\mathbf{w}$ suffers from the curse of dimensionality and overfitting



???

1D case 2D case CIFAR case

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

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

loss function — prior/regulariser

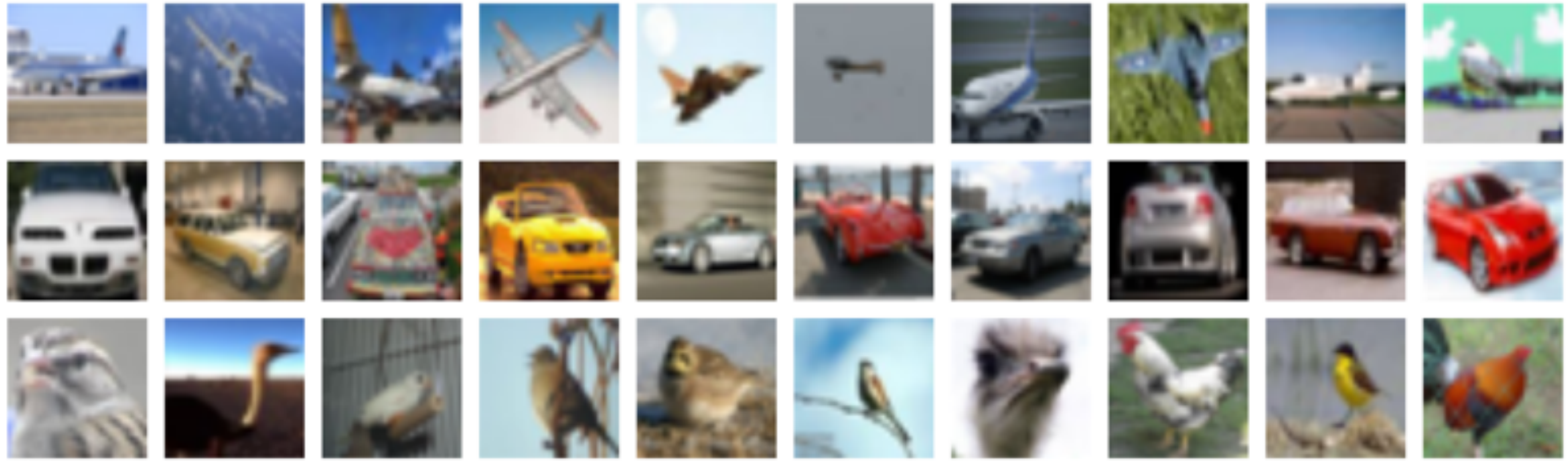
- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Wild $f(\mathbf{x}, \mathbf{w})$ with high-dimensional $\mathbf{w}$ suffers from the curse of dimensionality and overfitting
- We exploit prior $p(\mathbf{w})$ to restrict the wildness of $f(\mathbf{x}, \mathbf{w})$
  - L2 regulariser $p(\mathbf{w}) = \mathcal{N}_{\mathbf{w}}(0, \sigma^2) \Rightarrow \|\mathbf{w}\|_2^2$
  - L1 regulariser, L1+L2 regulariser (elastic net)
  - prior on $f(\mathbf{x}, \mathbf{w})$ structure (e.g. consists of convolutions)
  - batch normalization

Labels (y_i) RGB images ($\mathbf{x}_i$)

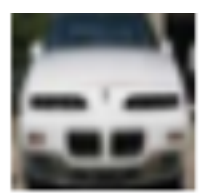
1

2

3



Three-class recognition problem:

def classify():

???

return $\mathbf{p}$

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

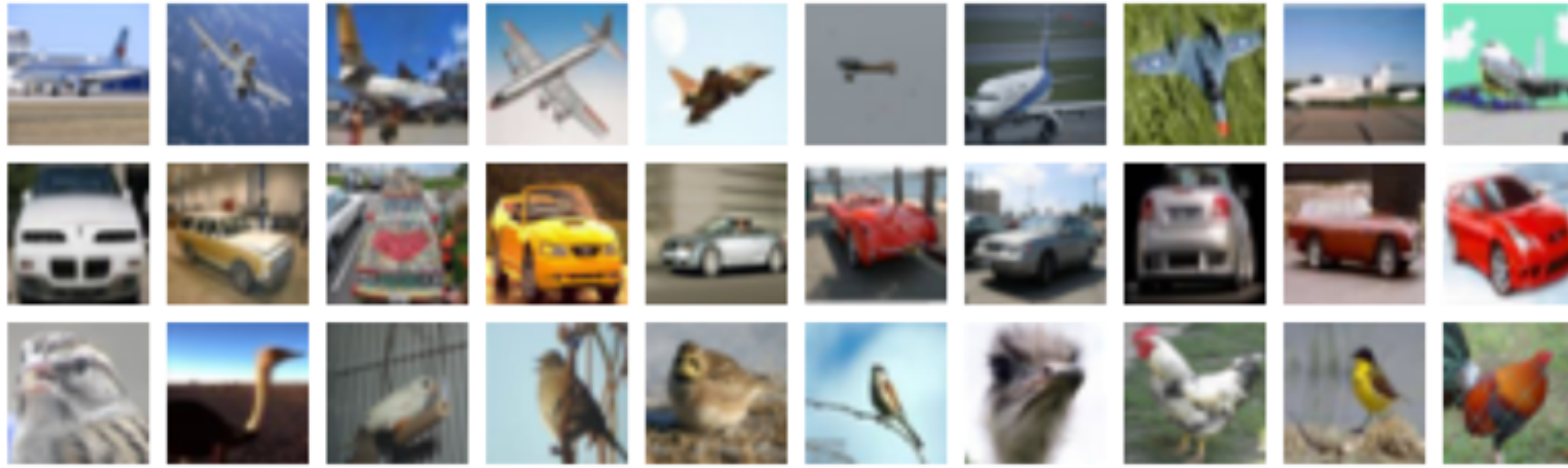
Labels (y_i)

RGB images ($\mathbf{x}_i$)

1

2

3



Model probability distribution over classes by softmax function

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

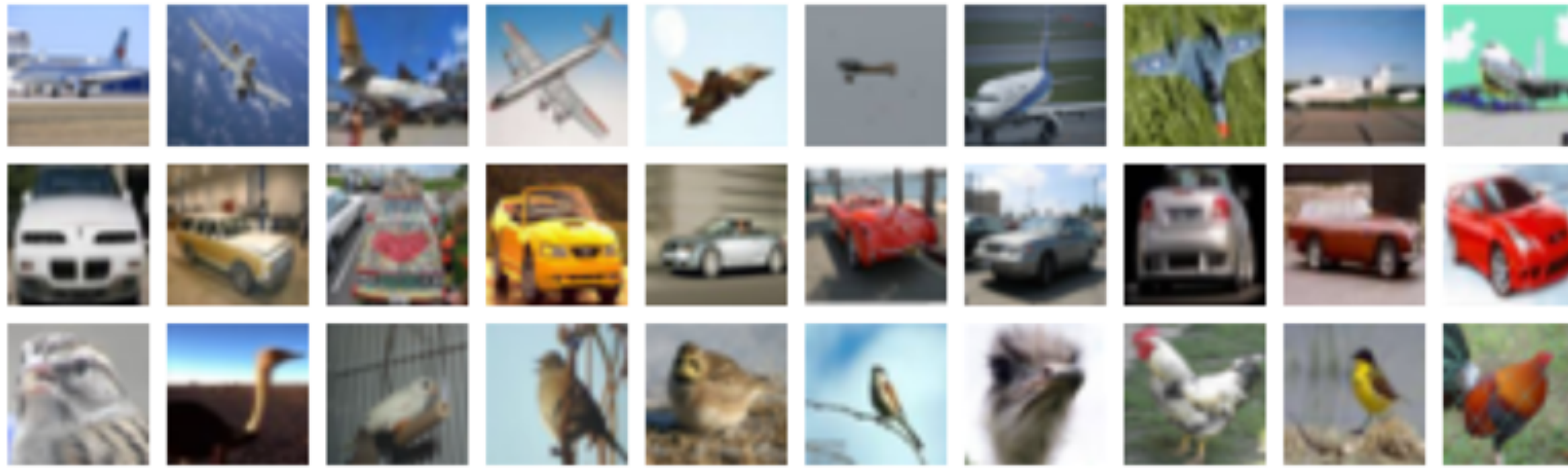
Labels (y_i)

RGB images ($\mathbf{x}_i$)

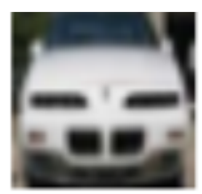
1

2

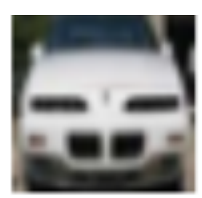
3



Three-class recognition problem:

def classify():

 # Linear classifier

$\mathbf{x} = \text{vec}($  $)$

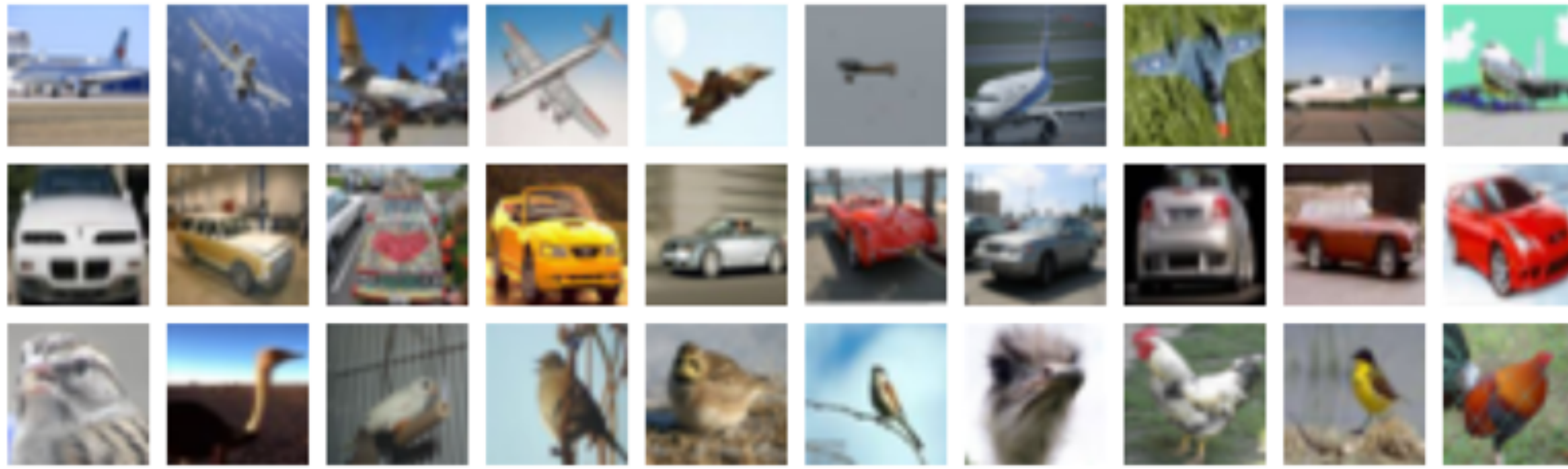
$\mathbf{p} = \mathbf{s}(\mathtt{W}\,\overline{\mathbf{x}})$

return $\mathbf{p}$

$$\mathtt{W}\;\overline{\mathbf{x}} = \begin{bmatrix} -2 \\ +1 \\ 0 \end{bmatrix}$$

$$\mathbf{s}\left(\begin{bmatrix} -2 \\ +1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.03 \\ 0.71 \\ 0.26 \end{bmatrix}$$

Labels (y_i) RGB images ($\mathbf{x}_i$)



def train(

):

???

return $\mathtt{W}^*$

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

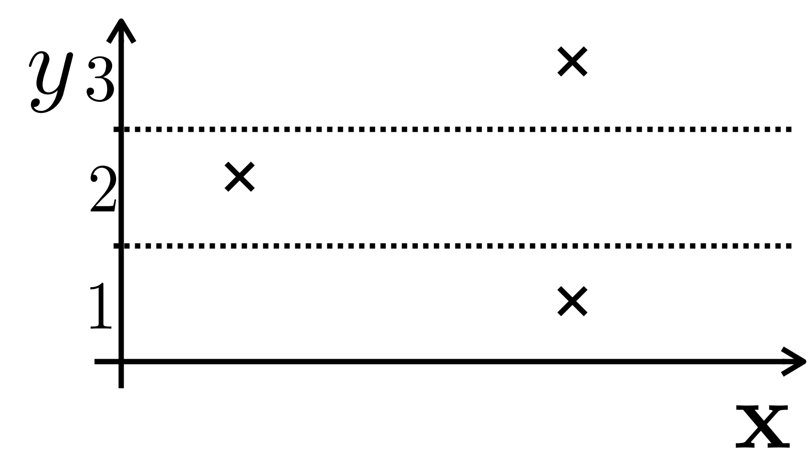
loss function prior/regulariser

- **Classification** (probability modeled by soft-max function):

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

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

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



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

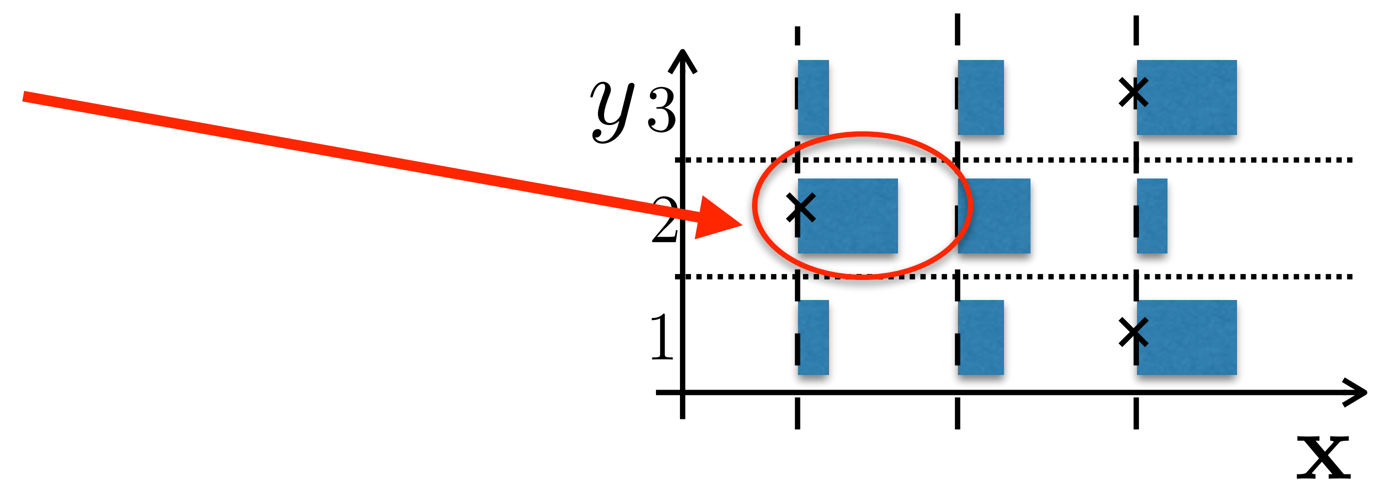
loss function prior/regulariser

- **Classification** (probability modeled by soft-max function):

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

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

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



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

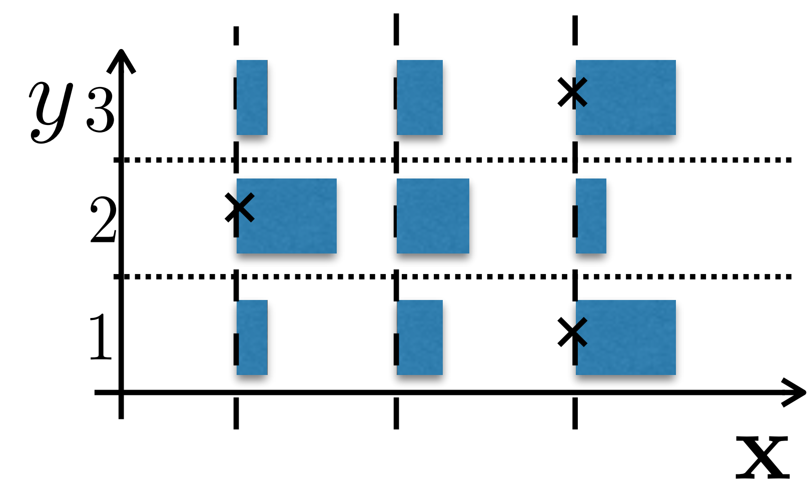
loss function — prior/regulariser

- **Classification** (probability modeled by soft-max function):

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

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

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



- subst. yields cross-entropy loss

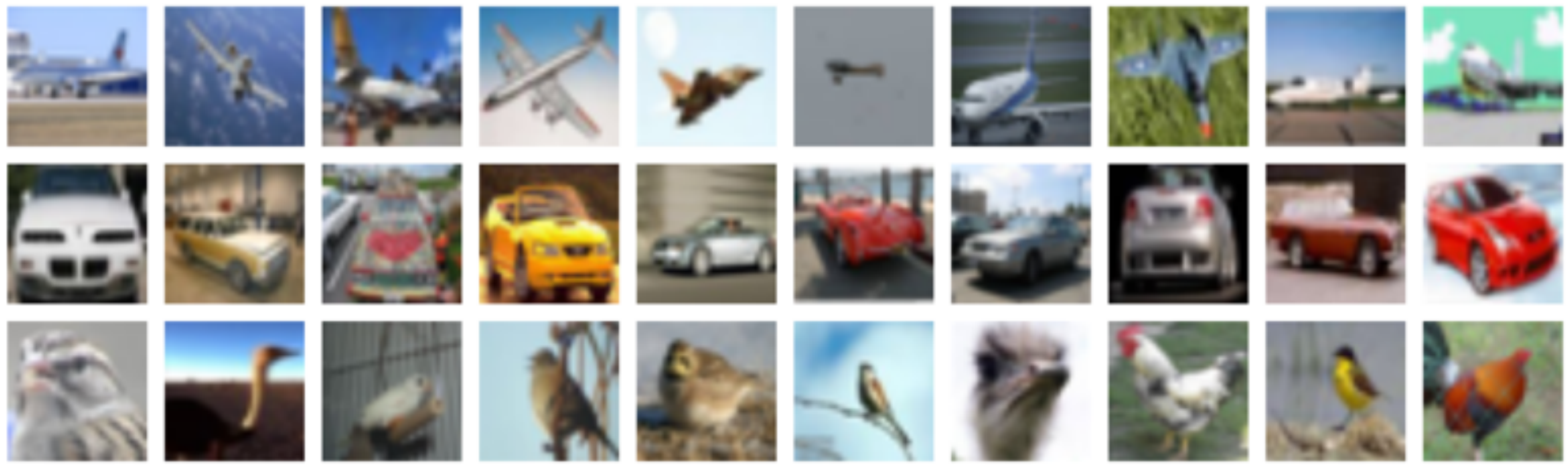
$$\mathtt{W}^* = \arg\min_{\mathtt{W}} \sum_i -\log \mathbf{s}_{y_i}(\mathbf{f}(\mathbf{x}_i, \mathtt{W}))$$

Labels (y_i) RGB images ($\mathbf{x}_i$)

1

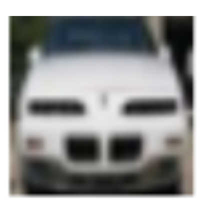
2

3



def train(

1 1 1 2 2 2 3 3 3):

$\mathbf{x}_i = \text{vec}($  $)$

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

return $\mathtt{W}^*$

$y_i = 2$

$$\mathbf{s}(\mathtt{W}\,\overline{\mathbf{x}}_i) = \begin{bmatrix} 0.03 \\ 0.71 \\ 0.26 \end{bmatrix} \Rightarrow \quad -\log \mathbf{s}_{y_i}(\mathtt{W}\,\overline{\mathbf{x}}_i) = -\log(0.71) = 0.15$$

def train():

1 1 1 2 2 2 3 3 3

$\mathbf{x}_i = \mathrm{vec}($  $)$

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

return $\mathtt{W}^*$

$y_i = 1$

$$\mathbf{s}(\mathtt{W}\,\overline{\mathbf{x}}_i) = \begin{bmatrix} 0.03 \\ 0.57 \\ 0.40 \end{bmatrix} \Rightarrow \quad -\log \mathbf{s}_{y_i}(\mathtt{W}\,\overline{\mathbf{x}}_i) = -\log(0.03) = 1.52$$

def train(
1 1 1 2 2 2 3 3 3):

$\mathbf{x}_i = \mathrm{vec}($  $)$

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

return $\mathtt{W}^*$

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

Conclusions

- Explained regression and linear classier as MAP estimator
- Discussed limitations, curse of dimensionality, overfitting and regularisations
- Next lesson will go deeper