

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

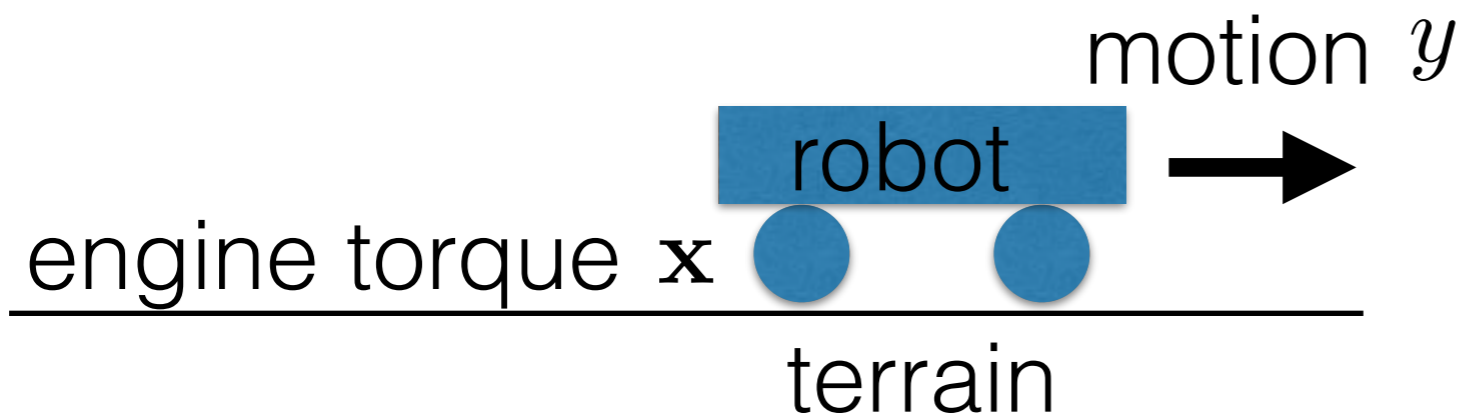


Outline

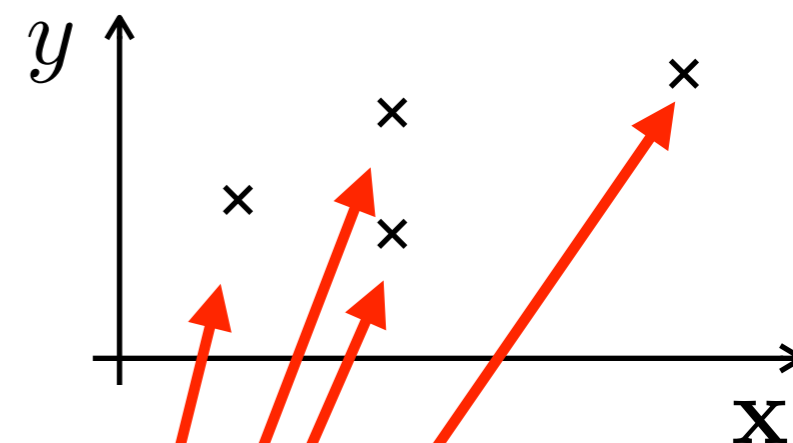
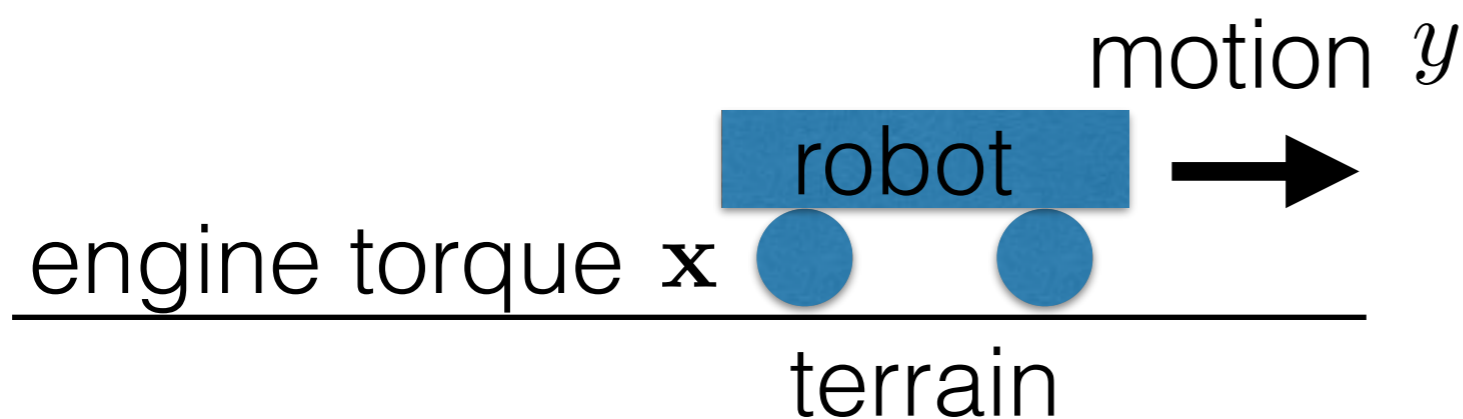
- Pre-requisites: linear algebra, Bayes rule
- MAP/ML estimation, prior and overfitting
- Linear regression
- Linear classification



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



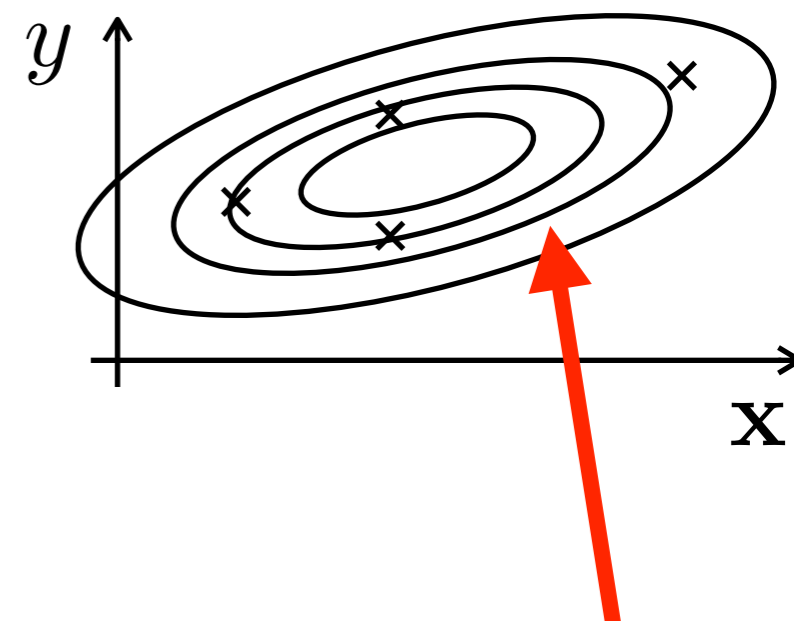
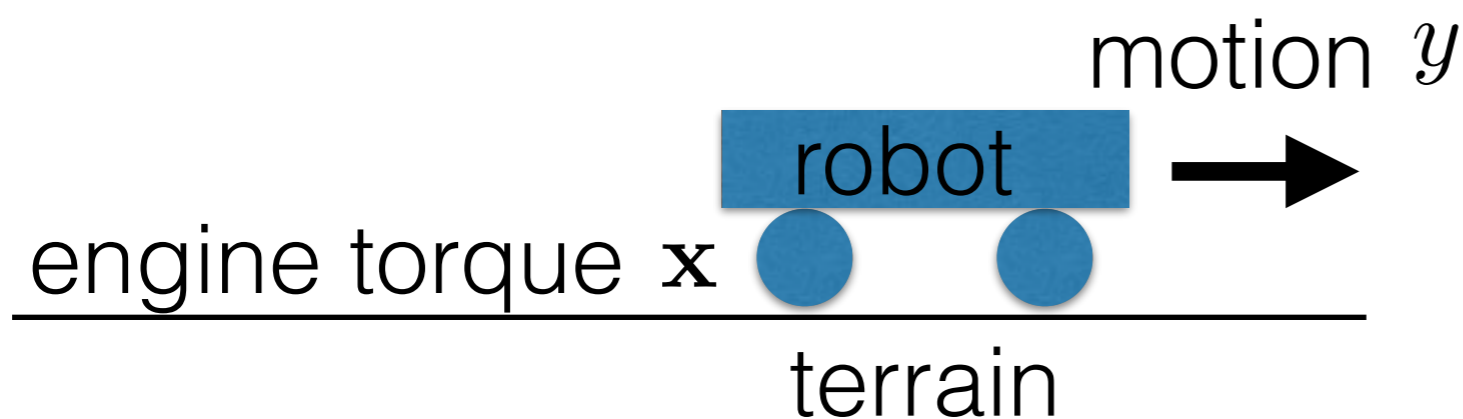
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model



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



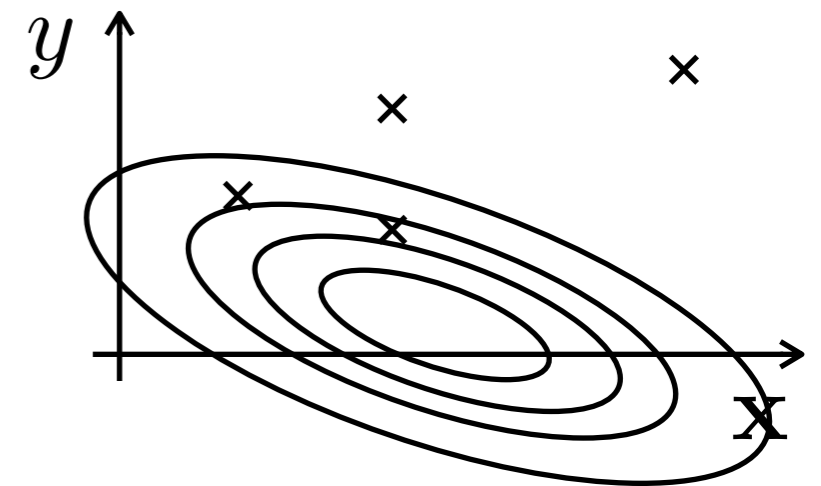
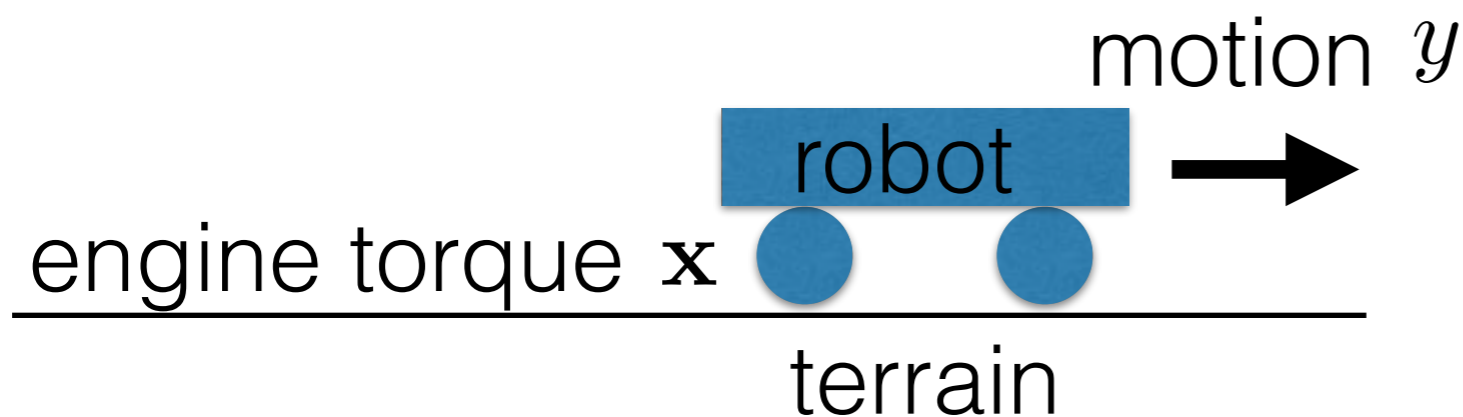
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model



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



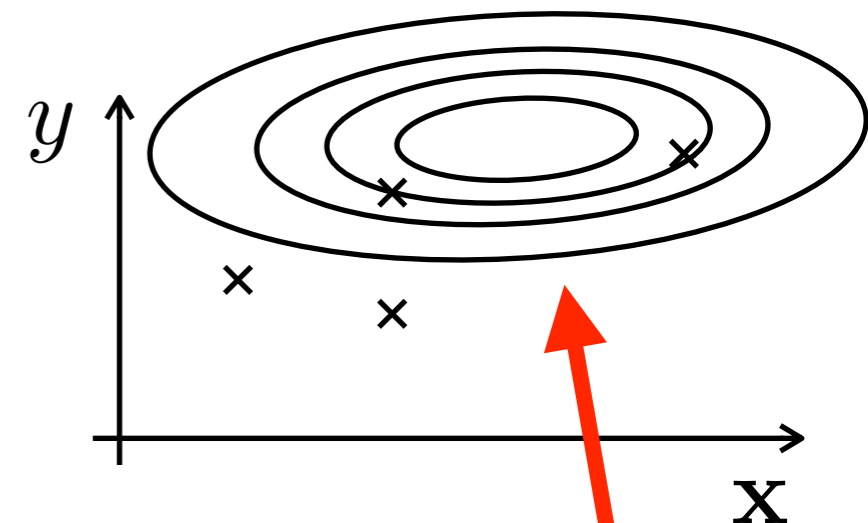
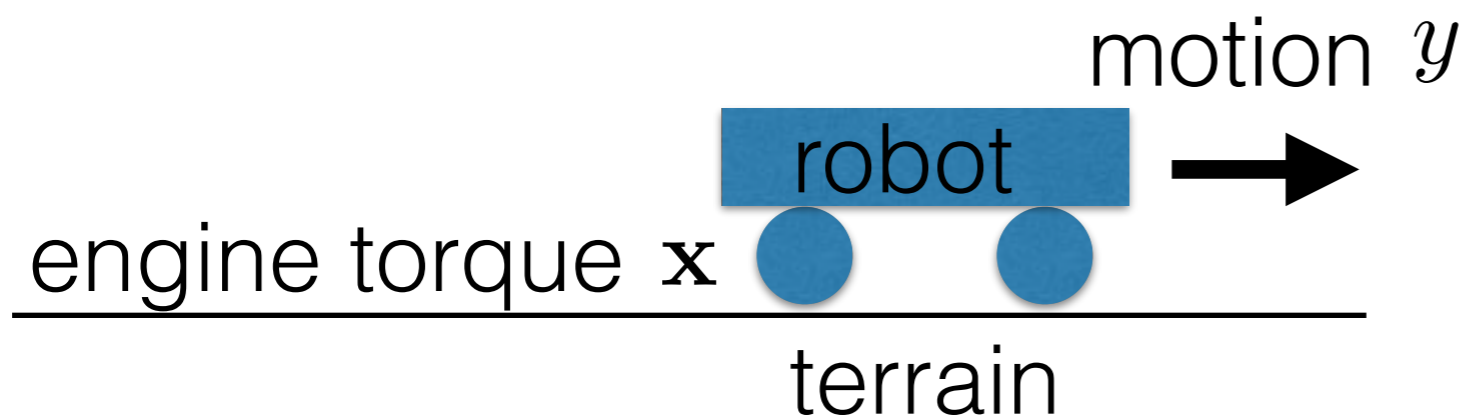
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model



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



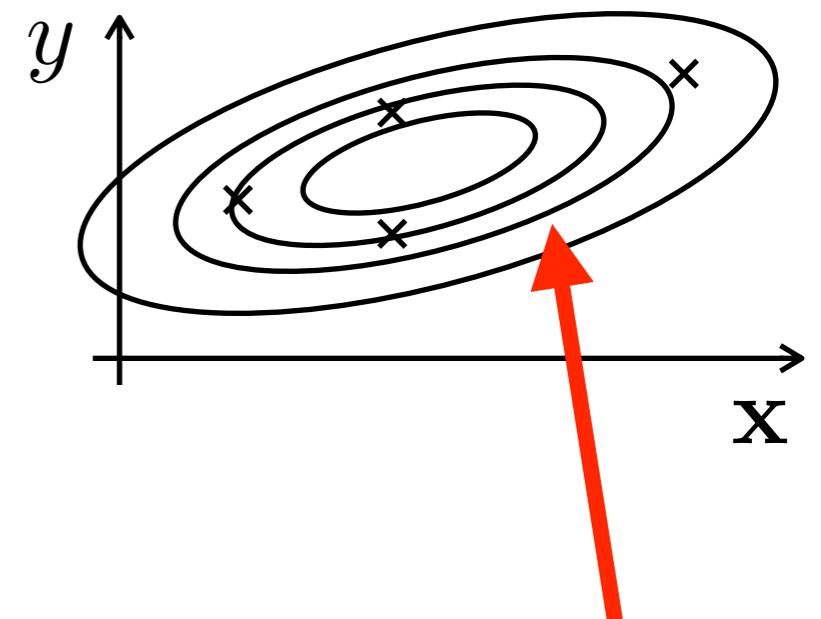
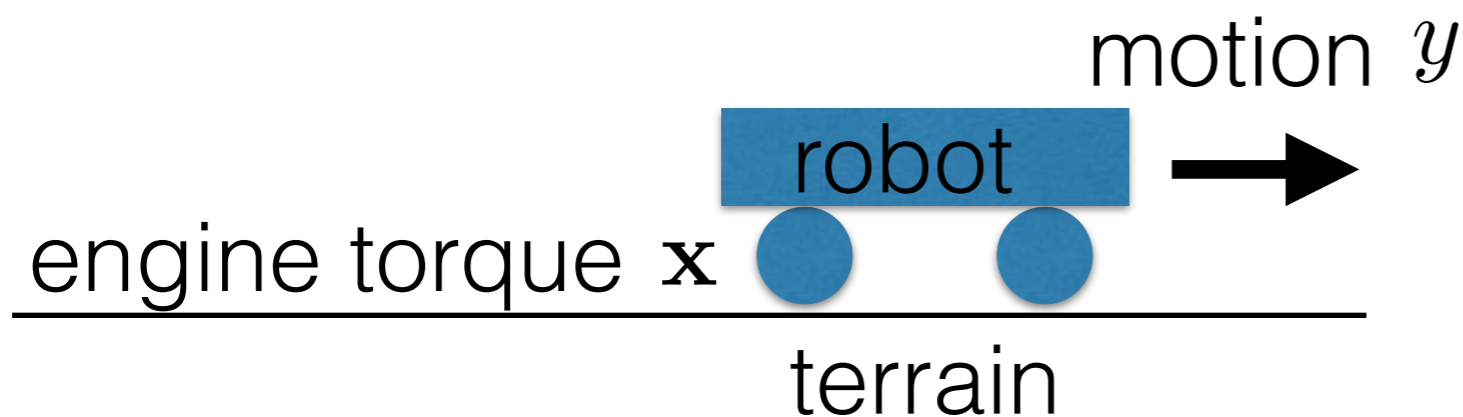
- Fast summary of Maximum A-Posteriori estimation of parameters of a probability distribution
- Motivation example: estimation of a motion model



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



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



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

$$\begin{aligned}\mathbf{w}^* &= \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg \max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \\ &= \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg \max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w})\end{aligned}$$



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

$$\begin{aligned}
 \mathbf{w}^* &= \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg \max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \\
 &= \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg \max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w}) \\
 &= \arg \max_{\mathbf{w}} \left(\prod_i p(\mathbf{x}_i, y_i|\mathbf{w}) \right) p(\mathbf{w}) \\
 &= \arg \max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w})p(\mathbf{x}_i) \right) p(\mathbf{w})
 \end{aligned}$$



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

$$\begin{aligned}
\mathbf{w}^* &= \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg \max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \\
&= \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) = \arg \max_{\mathbf{w}} p(\mathbf{x}_1, y_1 \dots \mathbf{x}_N, y_N|\mathbf{w})p(\mathbf{w}) \\
&= \arg \max_{\mathbf{w}} \left(\prod_i p(\mathbf{x}_i, y_i|\mathbf{w}) \right) p(\mathbf{w}) \\
&= \arg \max_{\mathbf{w}} \left(\prod_i p(y_i|\mathbf{x}_i, \mathbf{w})p(\mathbf{x}_i) \right) p(\mathbf{w}) \\
&= \arg \max_{\mathbf{w}} \left(\sum_i \log(p(y_i|\mathbf{x}_i, \mathbf{w})) + \log p(\mathbf{x}_i) \right) + \log p(\mathbf{w})
\end{aligned}$$



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

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

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

log likelihood

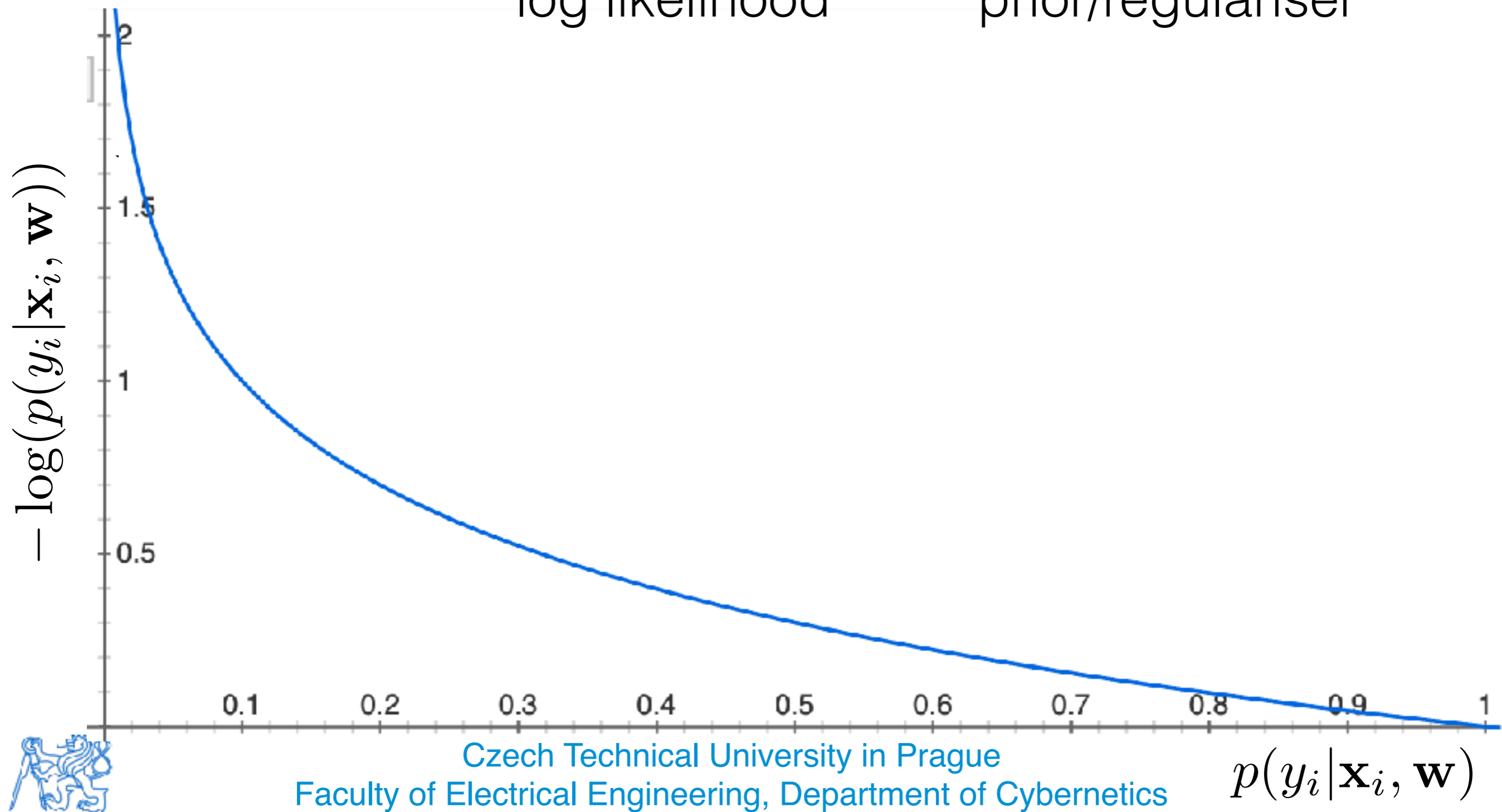
prior/regulariser



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

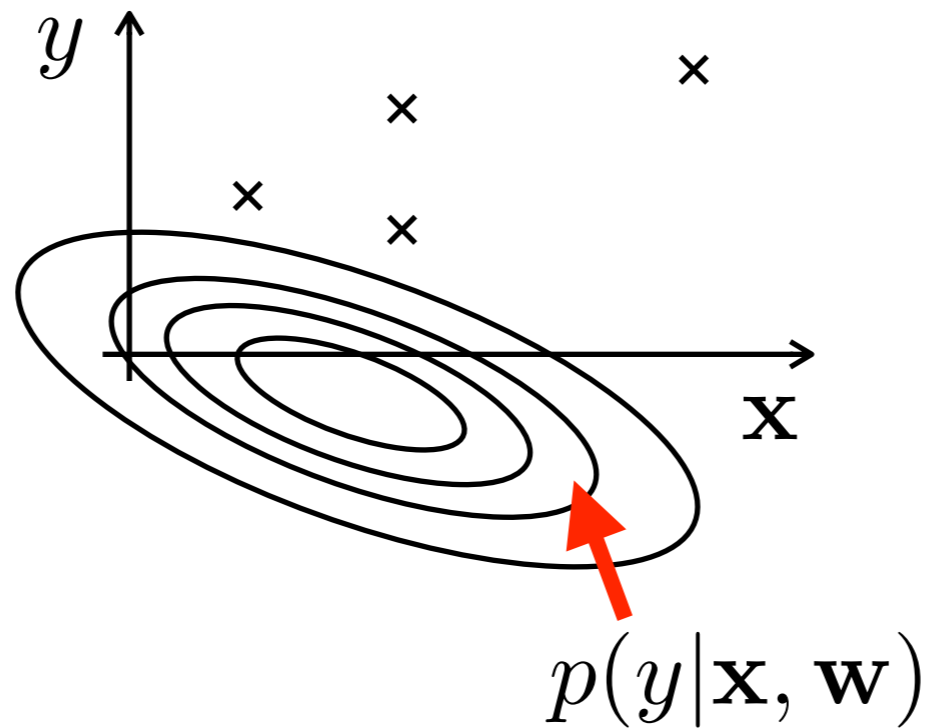
log likelihood

prior/regulariser



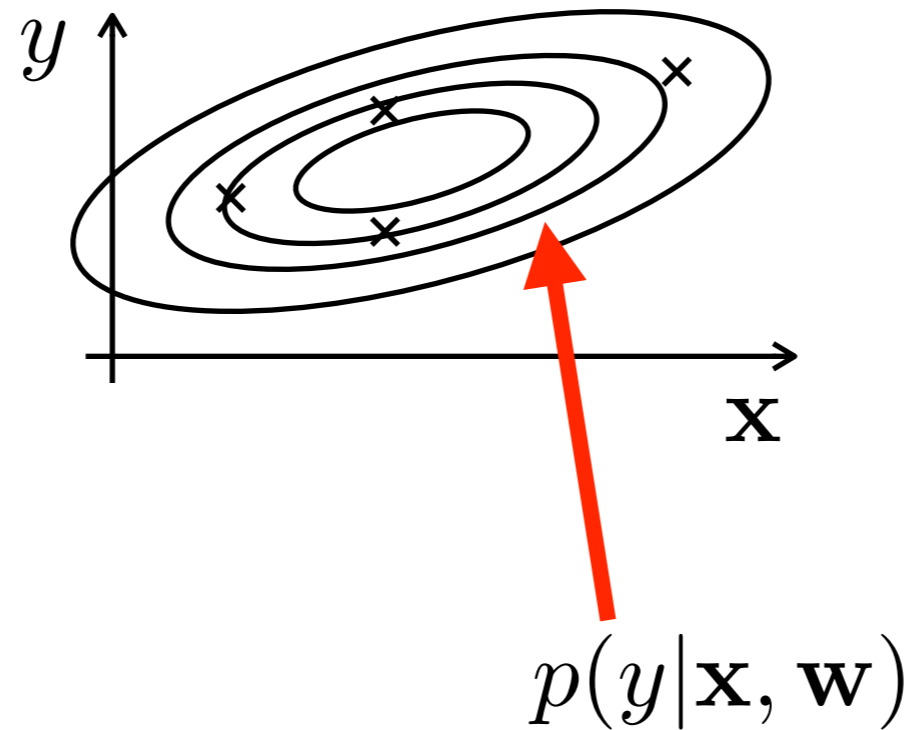
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood
prior/regulariser



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

log likelihood
prior/regulariser

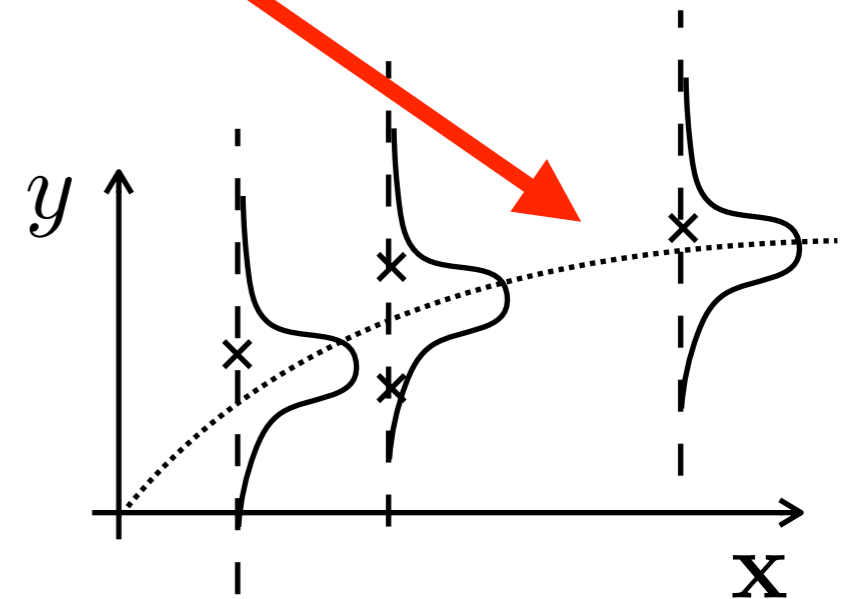


$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood

prior/regulariser

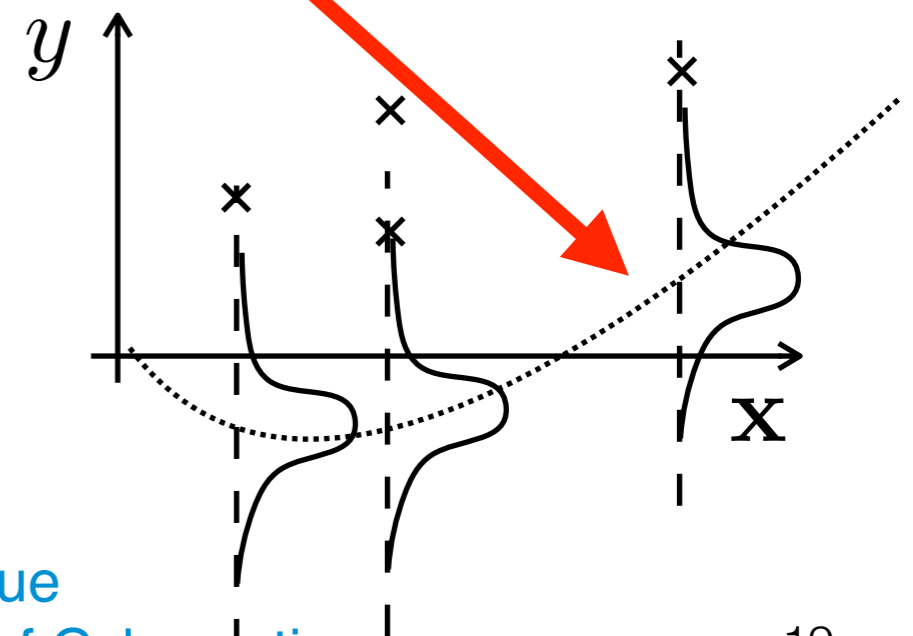
- **Regression:** $p(y | \mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$



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

log likelihood prior/regulariser

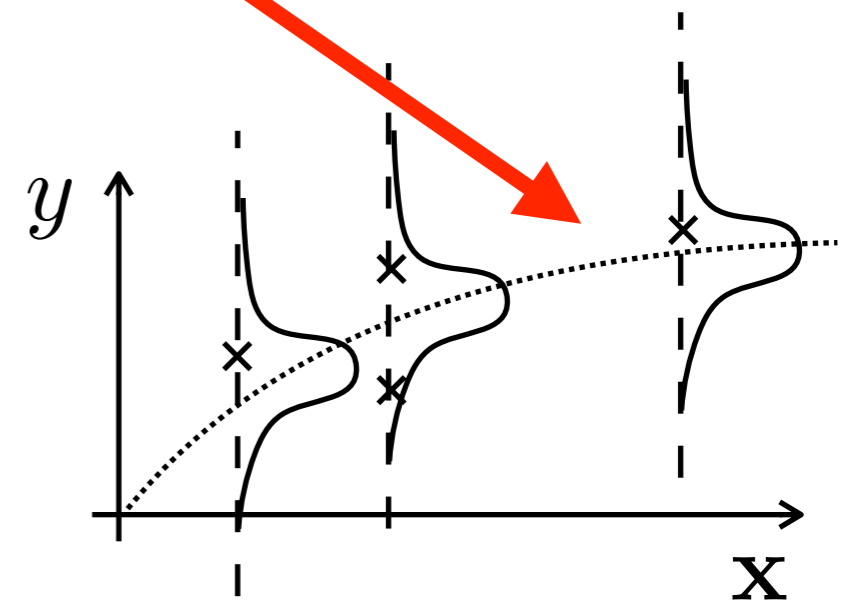
- **Regression:** $p(y | \mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$



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

log likelihood prior/regulariser

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



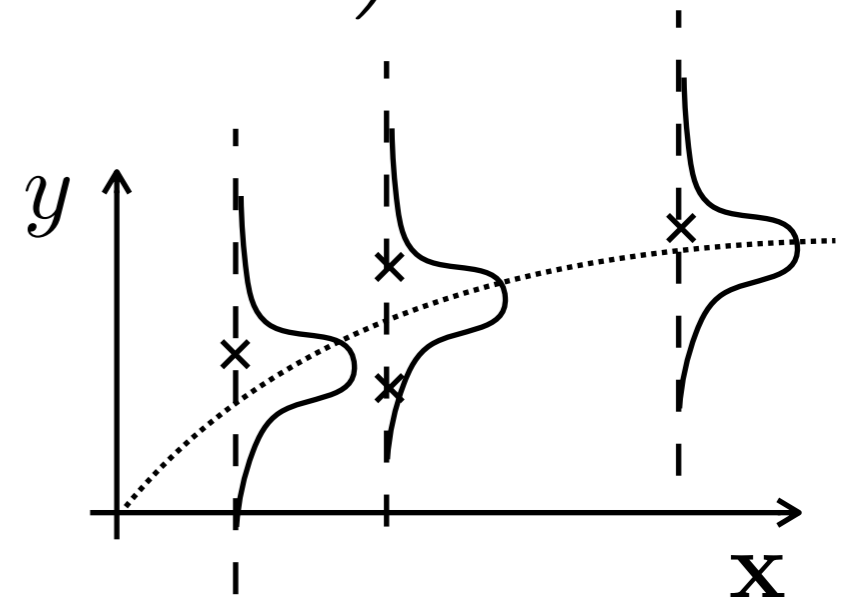
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood

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

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



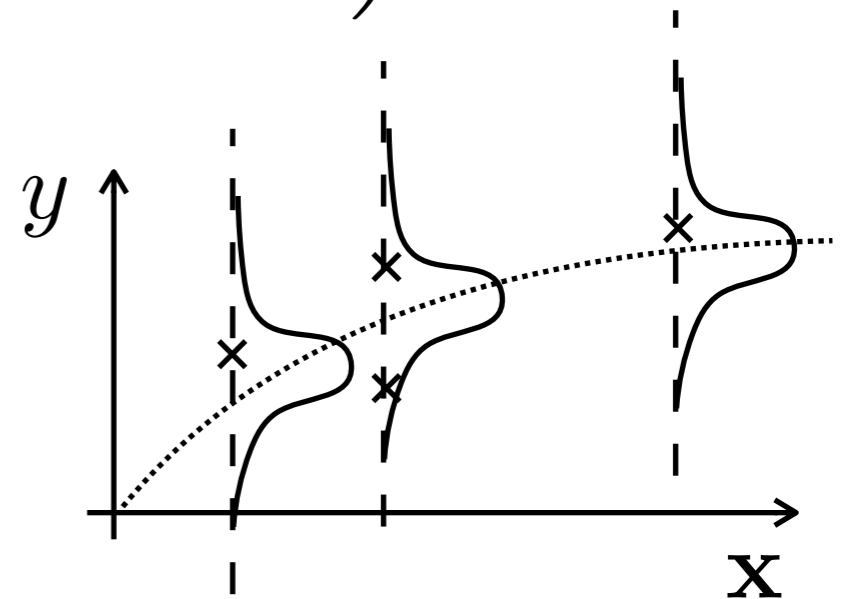
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) \cdot \square$$

log likelihood 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)



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



log likelihood

prior/regulariser

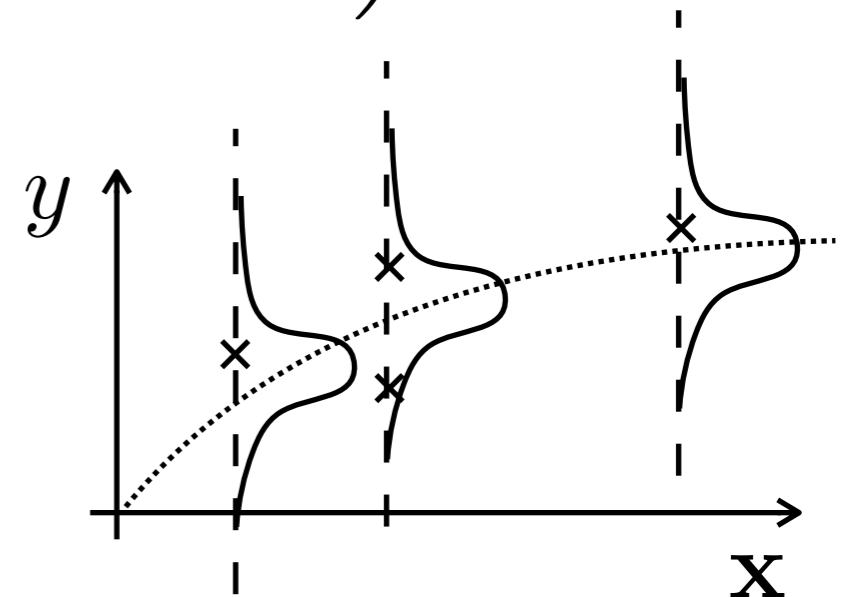
- **Regression:** $p(y | \mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$
- Probability of observing y_i when measuring \mathbf{x}_i is

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

- which yields well known L2 loss

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

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



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



log likelihood

prior/regulariser

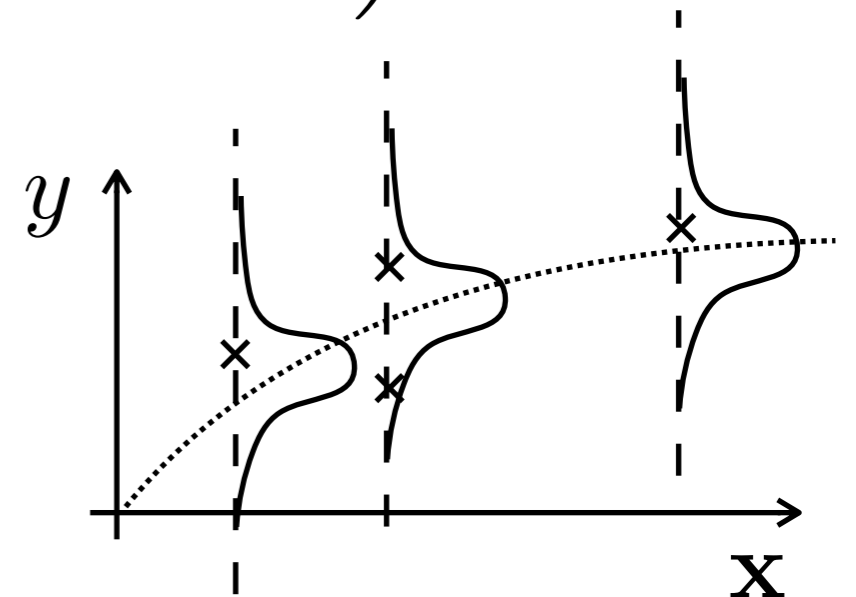
- **Regression:** $p(y | \mathbf{x}, \mathbf{w}) \sim \mathcal{N}_y(f(\mathbf{x}, \mathbf{w}), \sigma^2)$
- Probability of observing y_i when measuring \mathbf{x}_i is

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

- which yields well known L2 loss

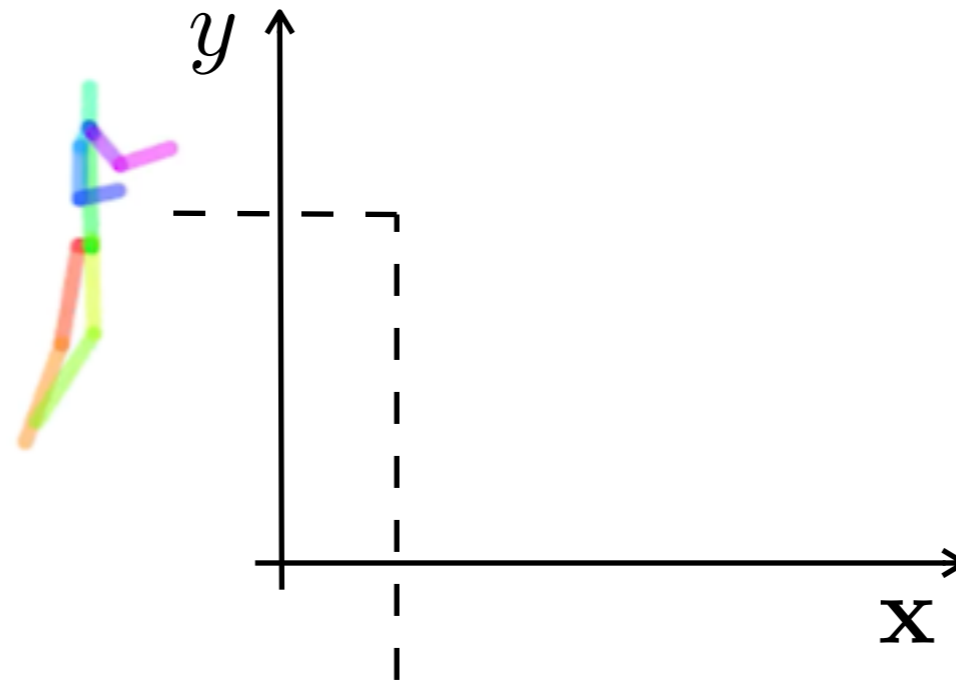
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

- Especially $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \bar{\mathbf{x}}$ yields closed-form solution



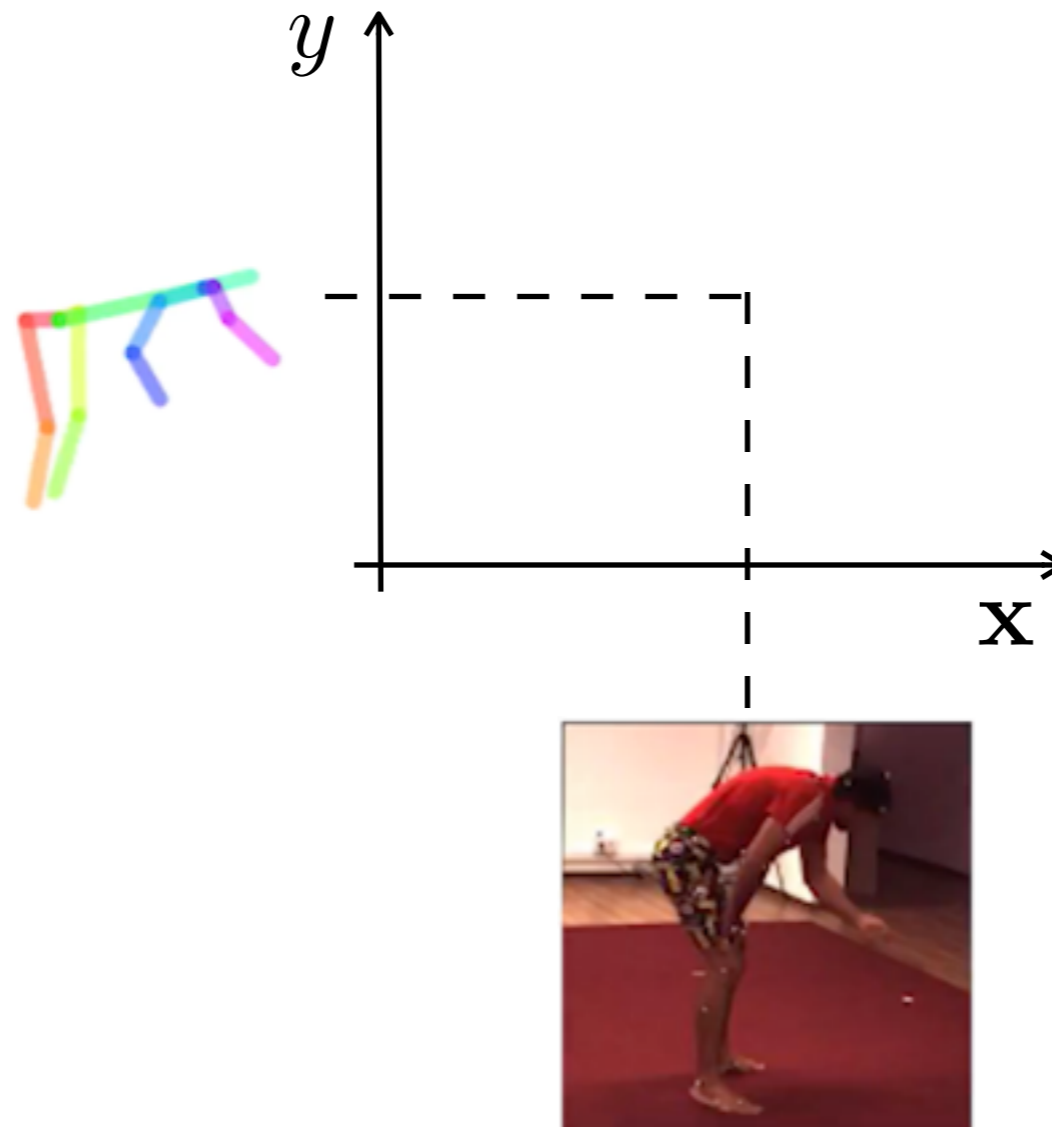
Other examples discussed during the course

3D pose regression



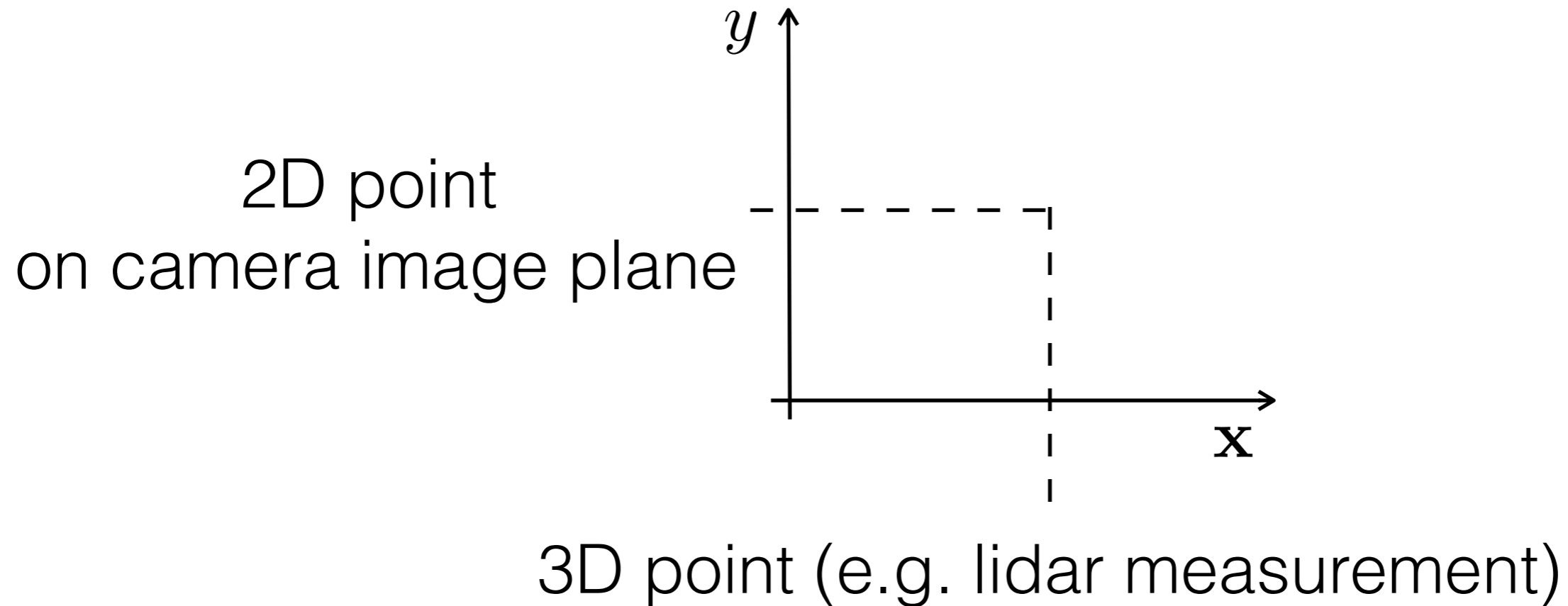
Other examples discussed during the course

3D pose regression



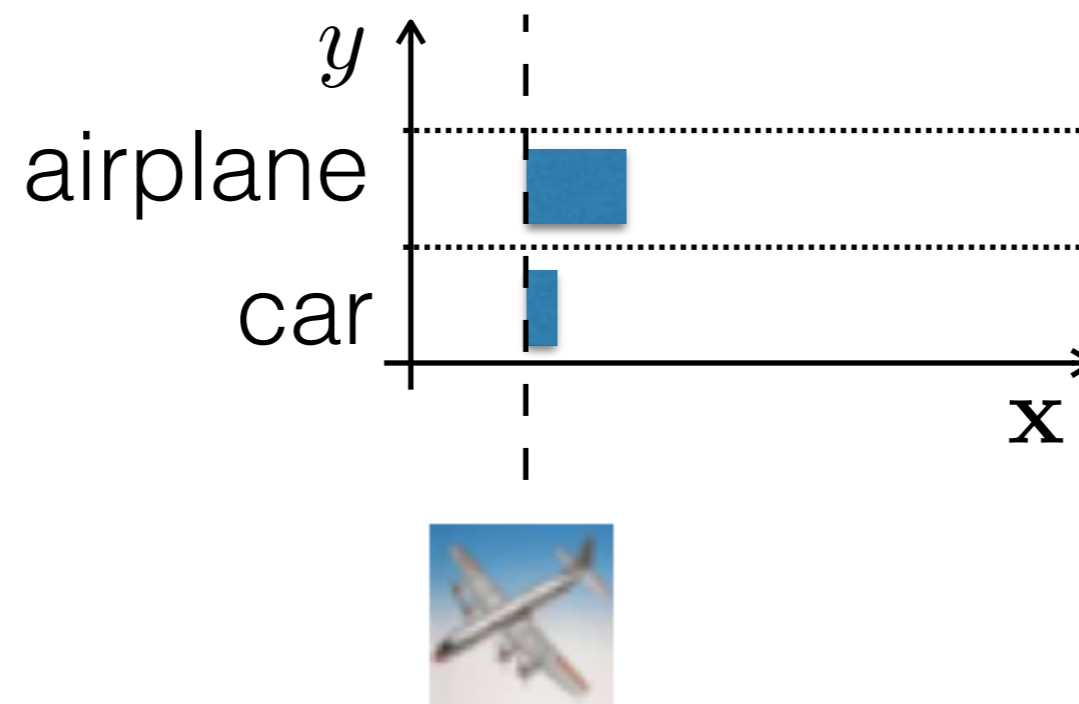
Other examples discussed during the course

Camera calibration



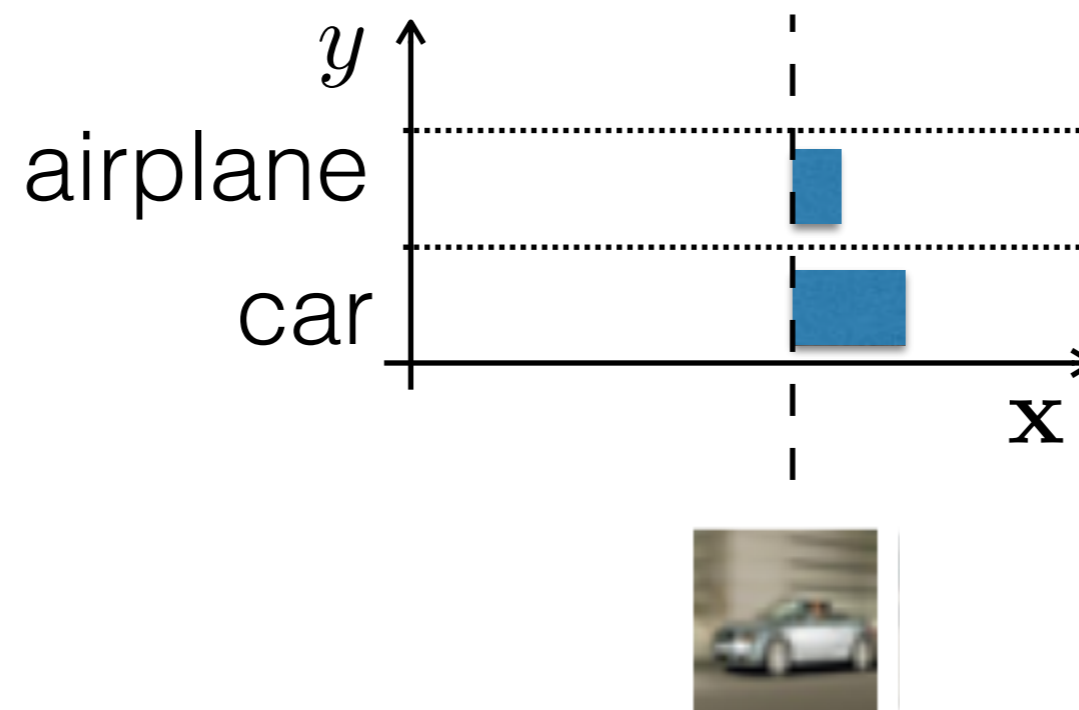
Other examples discussed during the course

Two-class object classification from RGB images



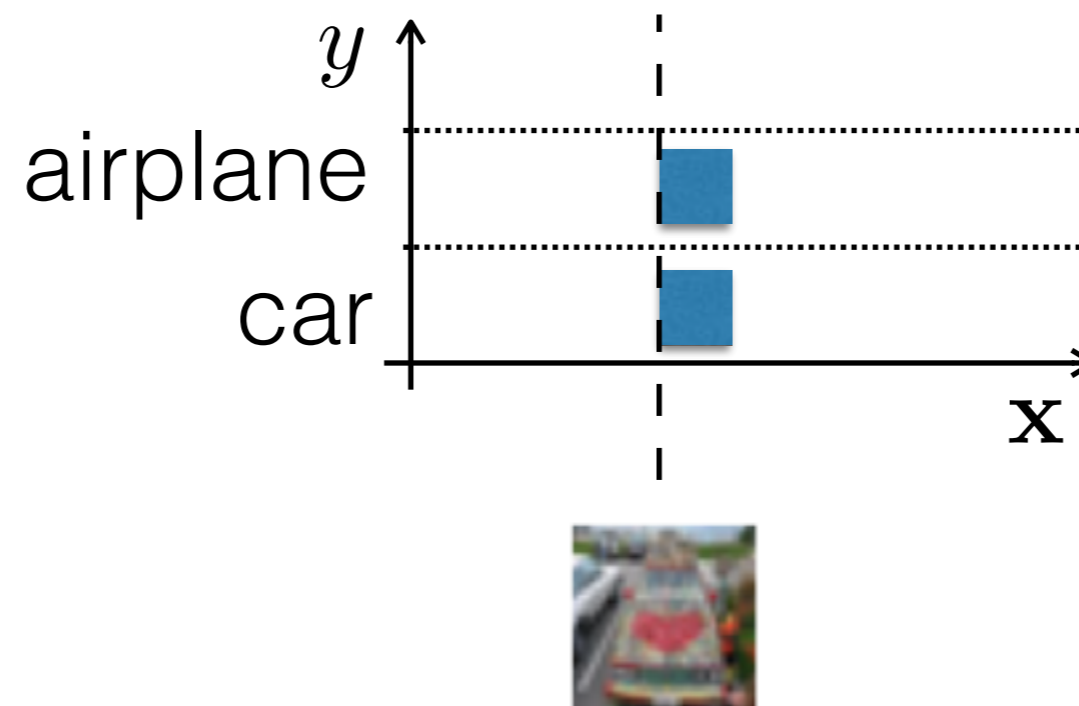
Other examples discussed during the course

Two-class object classification from RGB images



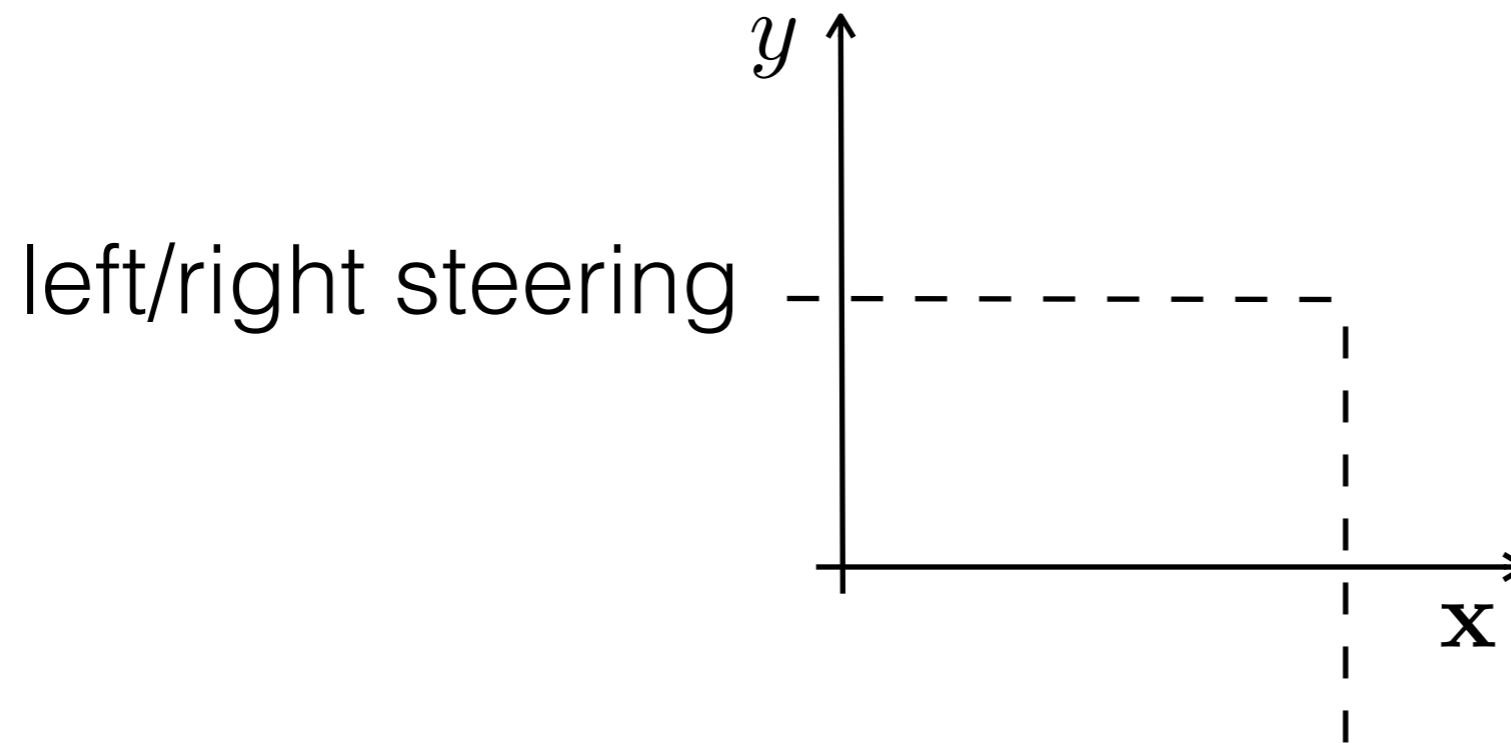
Other examples discussed during the course

Two-class object classification from RGB images



Other examples discussed during the course

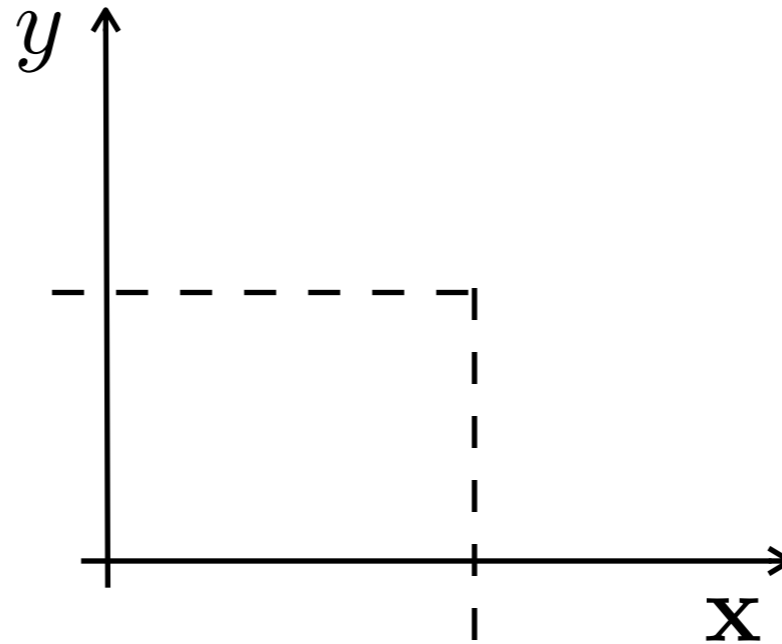
Reactive control



Other examples discussed during the course

Generative networks

winter image



summer image



Other examples discussed during the course

- “x” and/or “y” could be high-dimensional
- Assuming Gaussian noise is in many cases myopic
 - Pose regression left/right hand is often indistinguishable
 - Right/left avoiding of an obstacle should be replaced by a mean (center).
 - Coloring of grayscale images is also obviously not gaussian
- Linear function is obviously insufficient in many cases => more complex models needed.



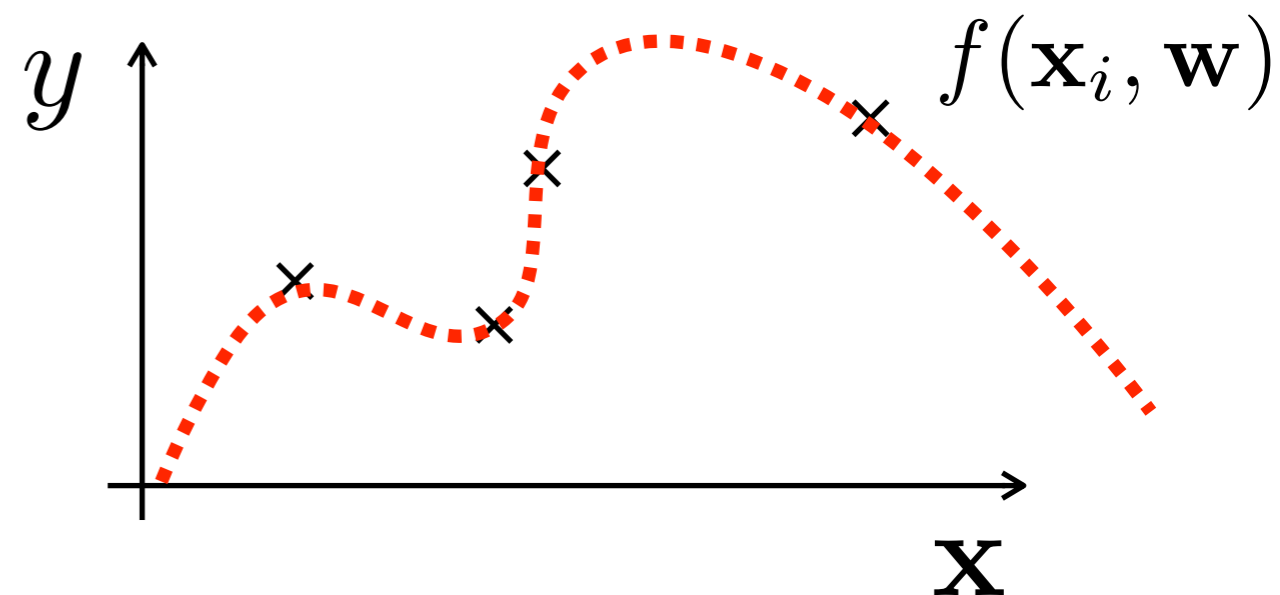
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) \cdot \square$$

log likelihood

prior/regulariser

- **Prior** is important:

no prior, powerful f => overfitting



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

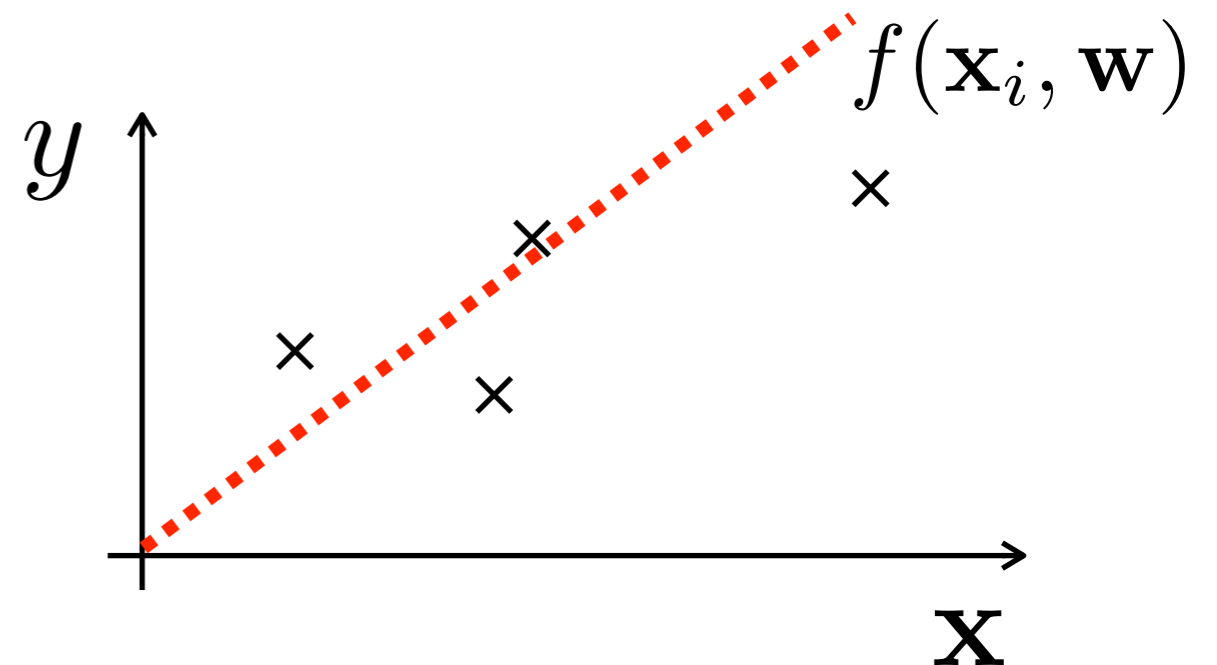


log likelihood

prior/regulariser

- **Prior** is important:

no prior, simple $f \Rightarrow$ underfitting



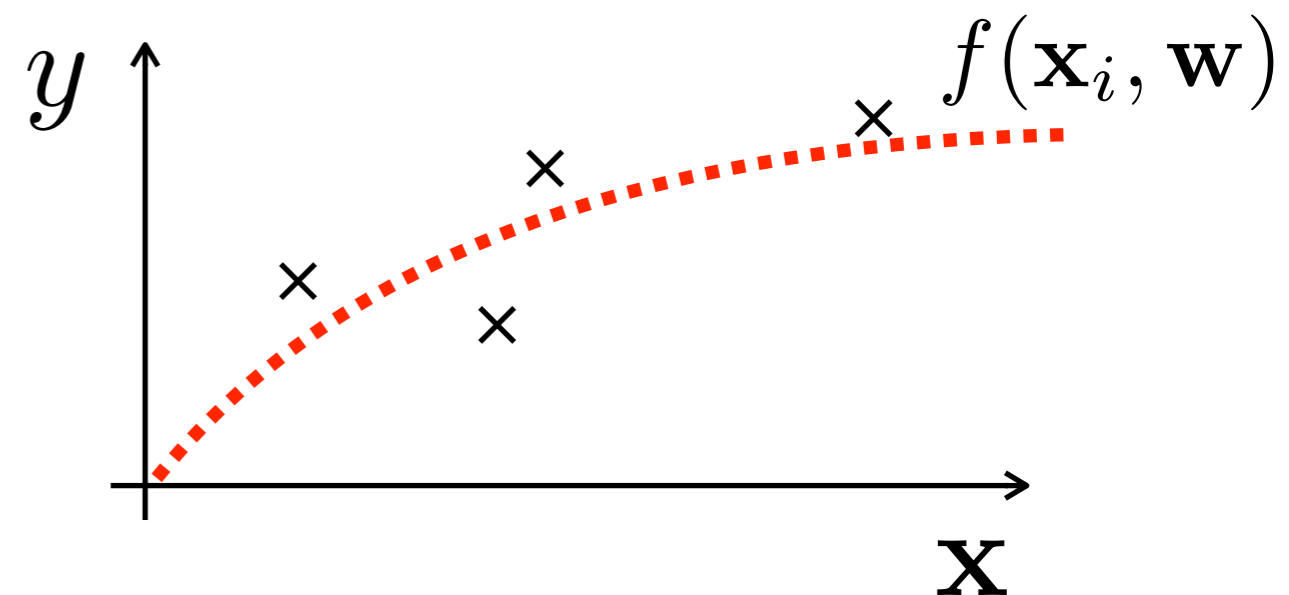
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

log likelihood

prior/regulariser

- **Prior** is important:

good prior



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

log likelihood

prior/regulariser

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

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



leprechauns can be
involved in any explanation



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

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



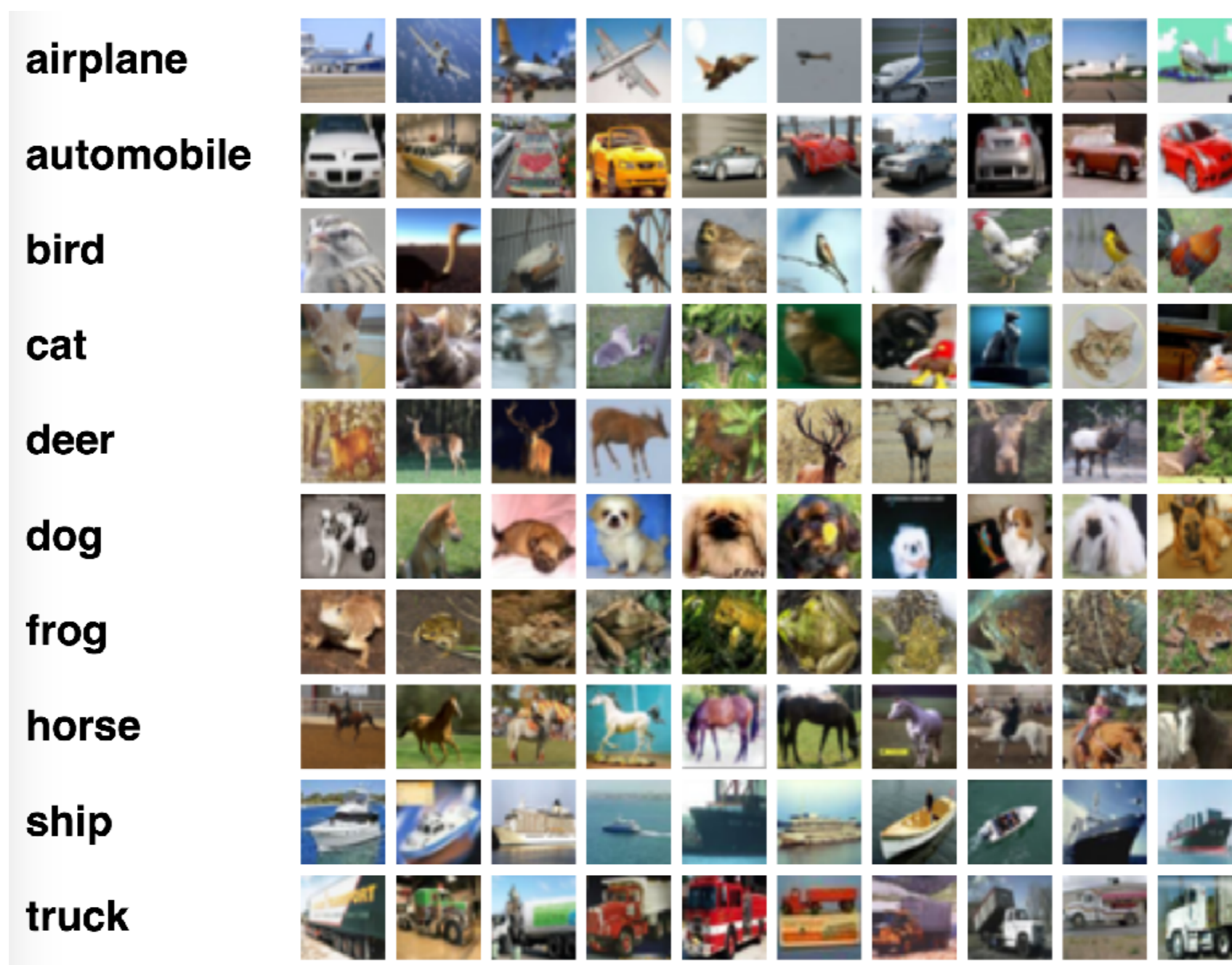
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

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, Multiview 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 !**

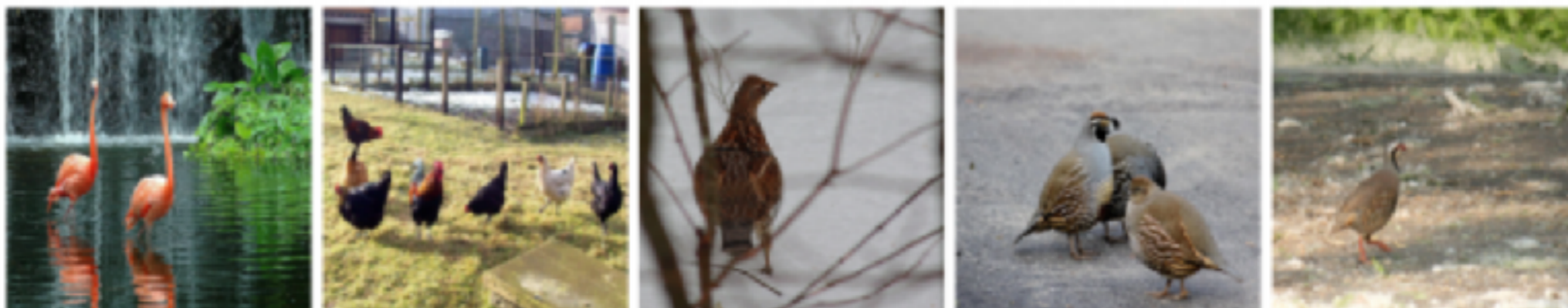
- Viewpoint
- Occlusion
- Illumination
- Pose
- Type
- Context



Recognition problem

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

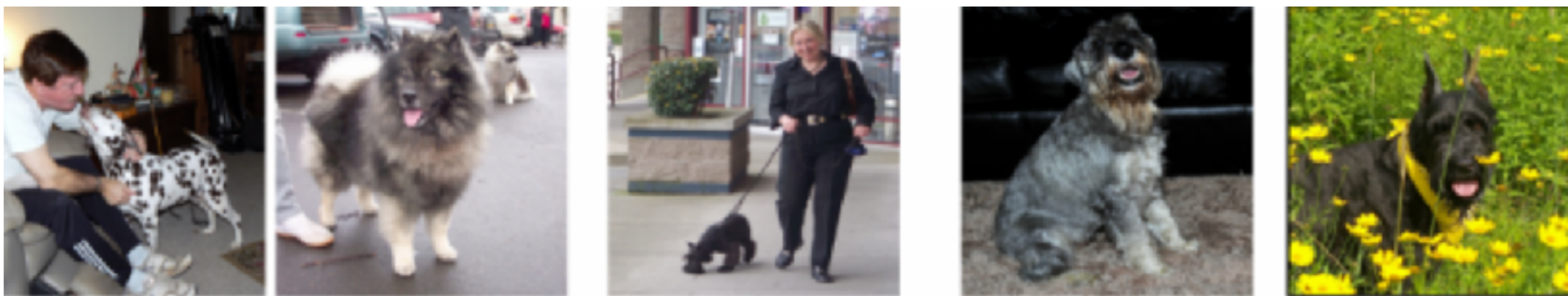
bird



cat



dog



Recognition problem

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

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

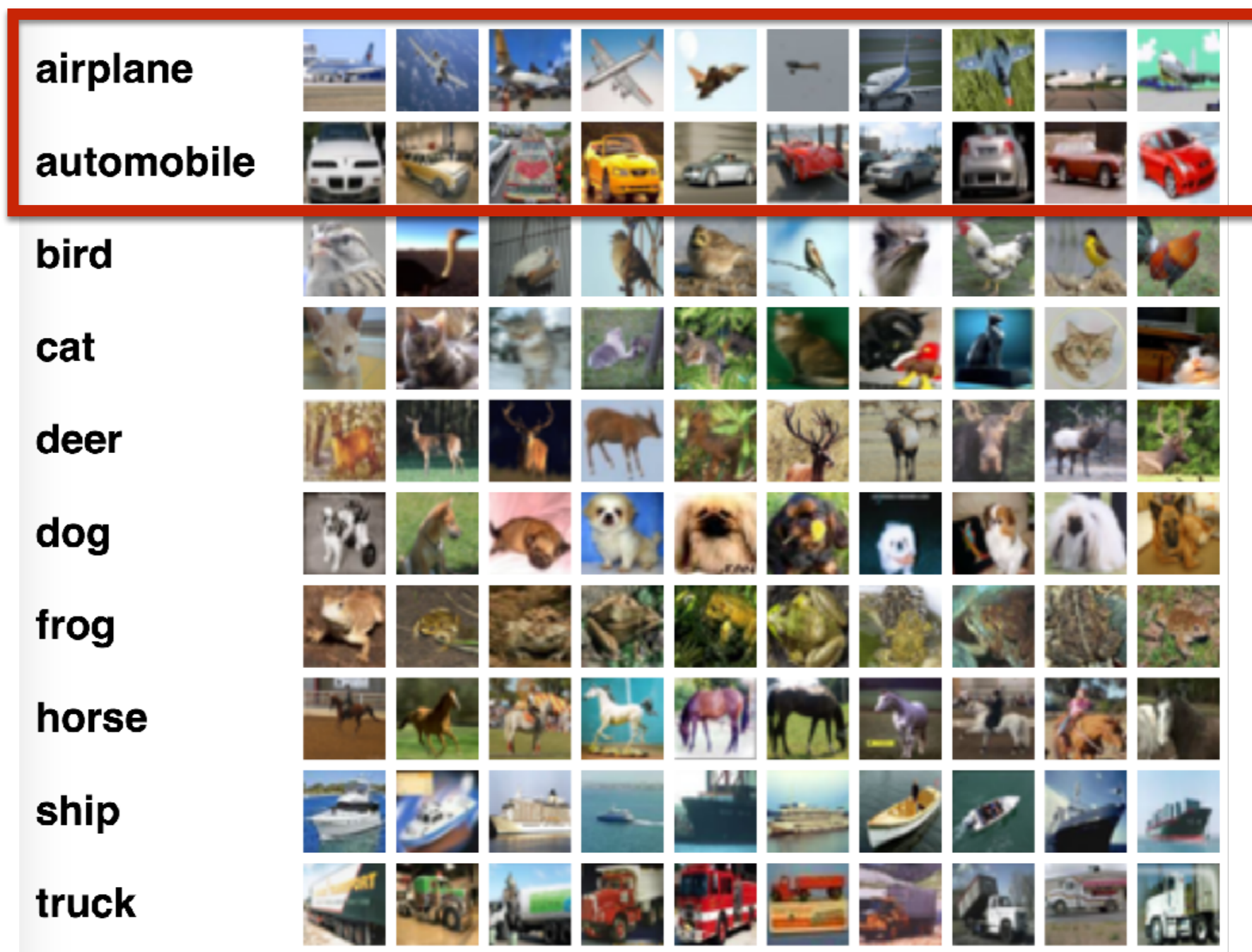


Timofte, Zimmermann, van Gool, Multiview traffic-sign detection, recognition and 3D localisation, MVA, 2014

<https://link.springer.com/content/pdf/10.1007/s00138-011-0391-3.pdf>



Recognition problem



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

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



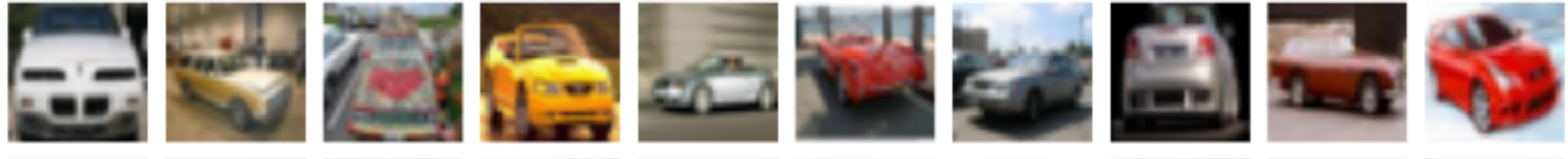
Labels (y_i)

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.



Labels (y_i)

RGB images (\mathbf{x}_i)

+1



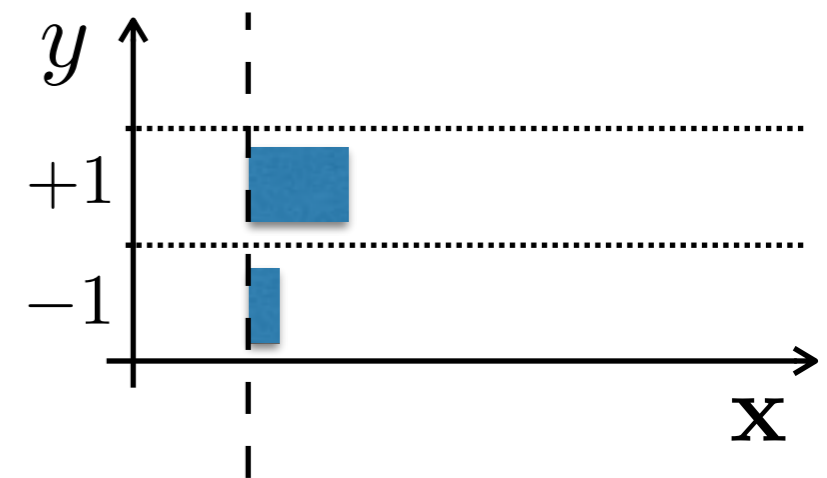
-1



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



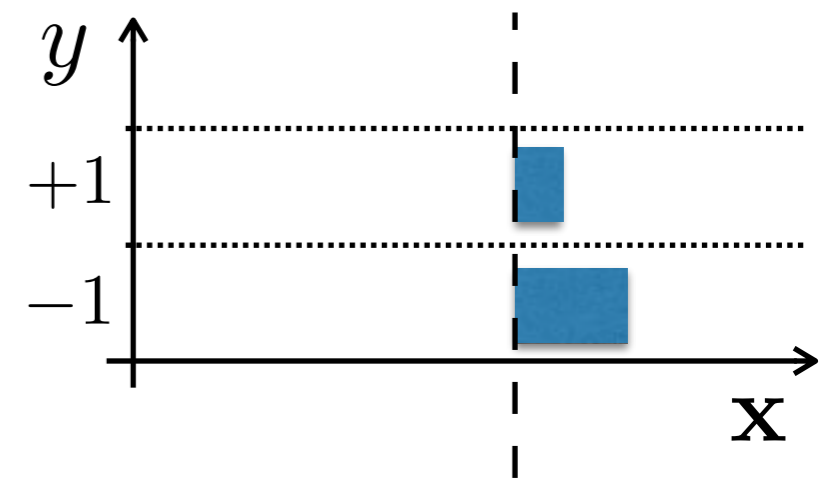
-1



Classification

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

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



Labels (y_i)

RGB images (\mathbf{x}_i)

+1



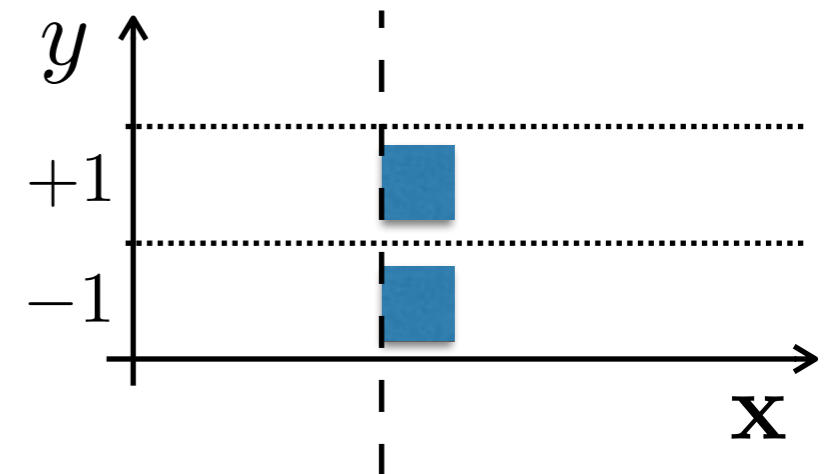
-1



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



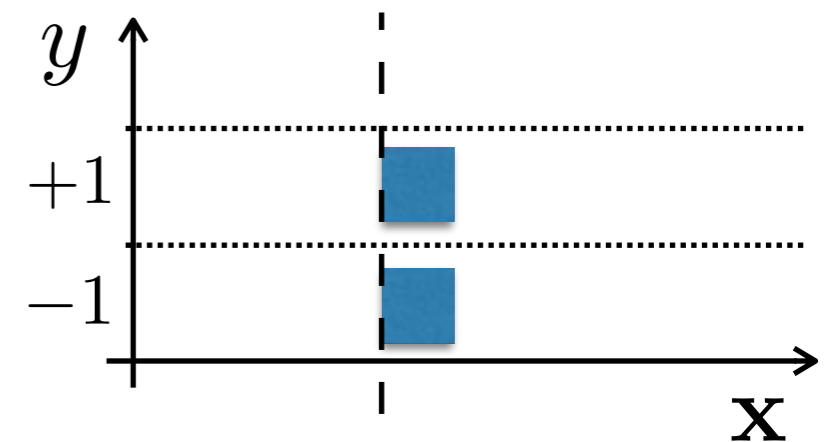
-1



Classification

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

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



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



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(  
     $p = \sigma(\mathbf{w}^\top \bar{\mathbf{x}})$   
    return  $p$ 
```

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

Labels (y_i)

RGB images (\mathbf{x}_i)

+1




-1



Classification

Example: Linear classifier

```
def classify(
$$\mathbf{w}^\top \bar{\mathbf{x}} = 2.5$$

```



Labels (y_i)

RGB images (\mathbf{x}_i)

+1




-1

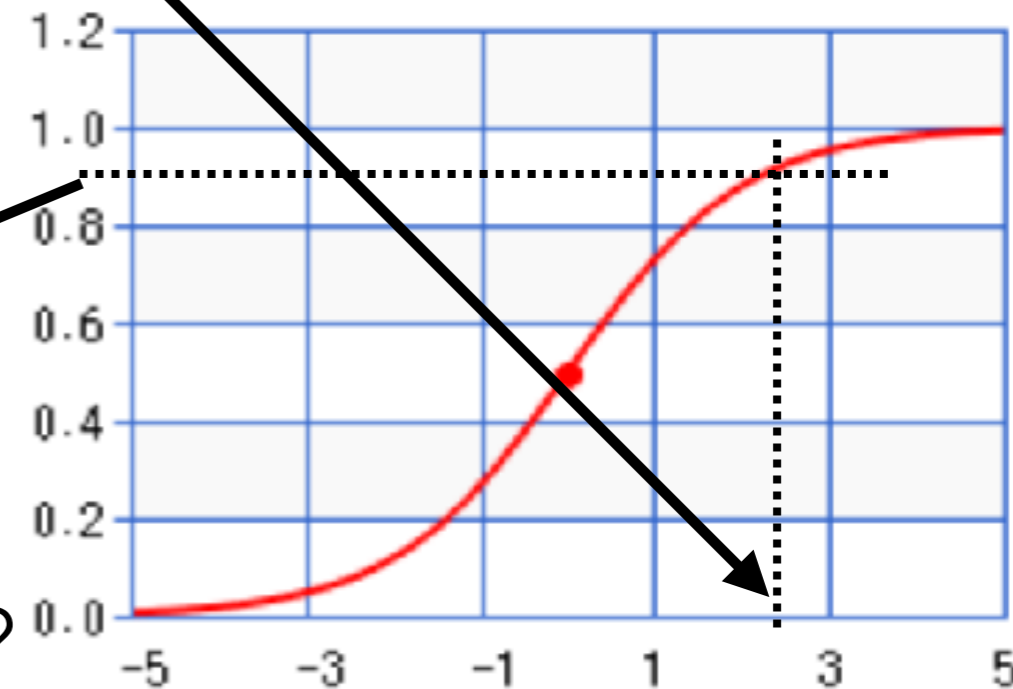


Classification

Example: Linear classifier

```
def classify(
$$\mathbf{w}^\top \bar{\mathbf{x}} = 2.5$$

```



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 +1 +1 -1 -1 -1 ):  
  
    ???  
  
    return  $\mathbf{w}^*$ 
```

???

return \mathbf{w}^*



Labels (y_i)

RGB images (\mathbf{x}_i)

+1



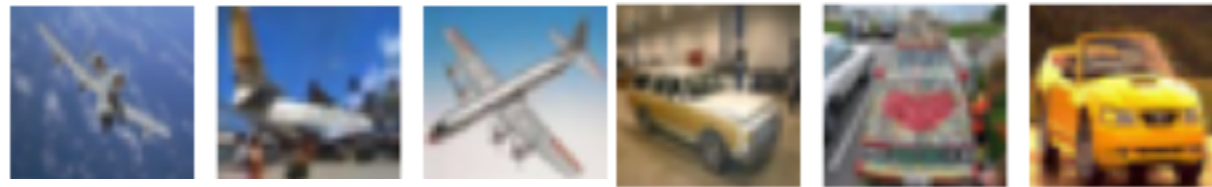
-1



Training

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

```
def train(
```

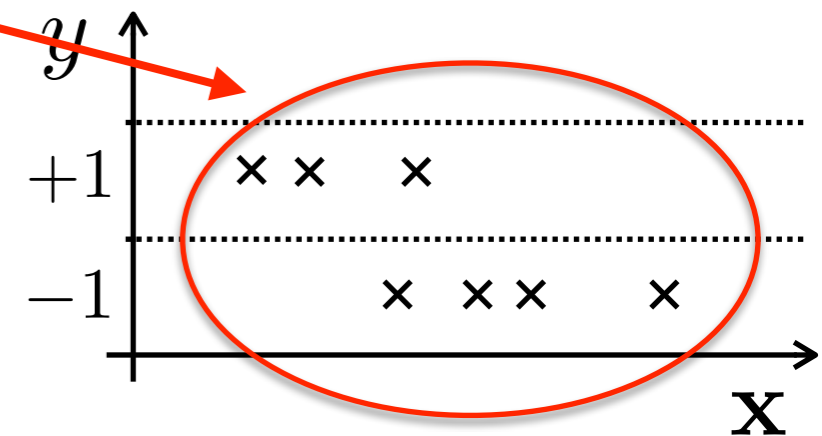


```
  +1  +1  +1  -1  -1  -1 ):
```

```
  ???
```

```
  return  $\mathbf{w}^*$ 
```

Training data



Labels (y_i)

RGB images (\mathbf{x}_i)

+1



-1

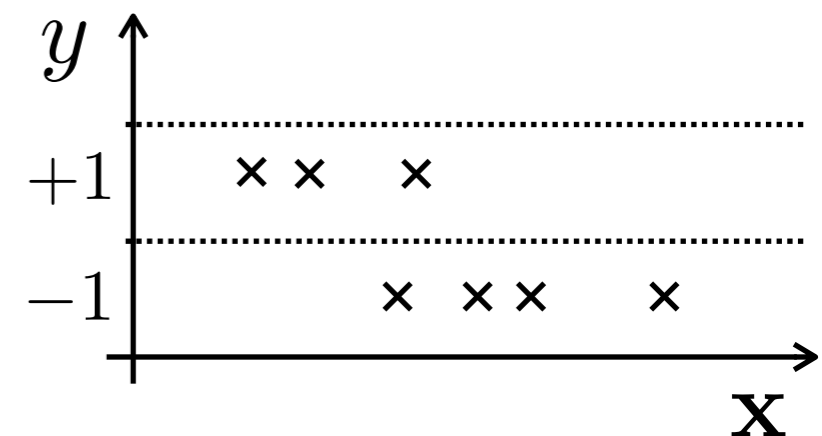


Training

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



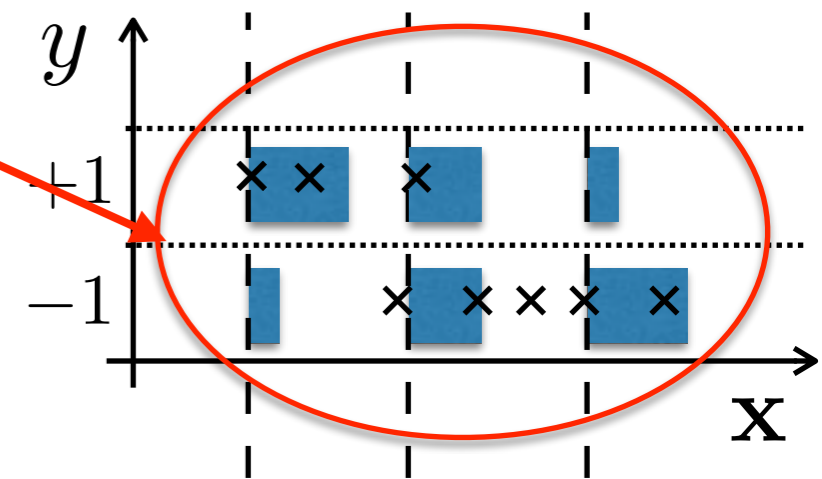
-1



Training

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

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

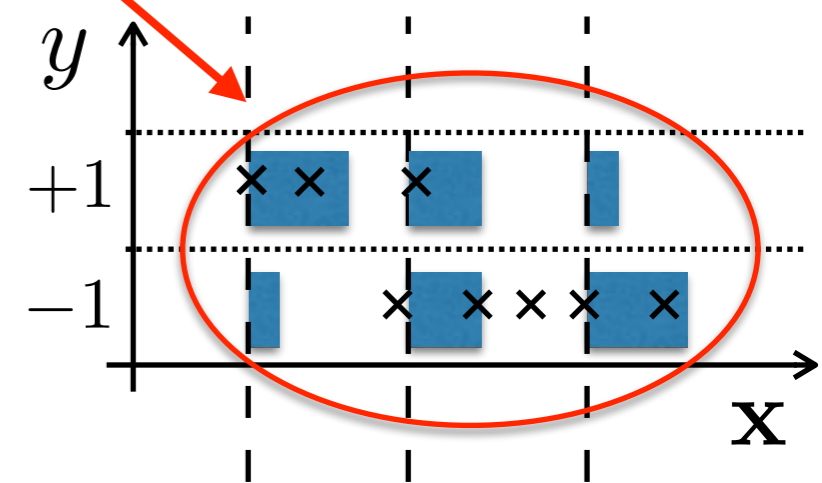


$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) \cdot \square$$

loss function

prior/regulariser

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



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

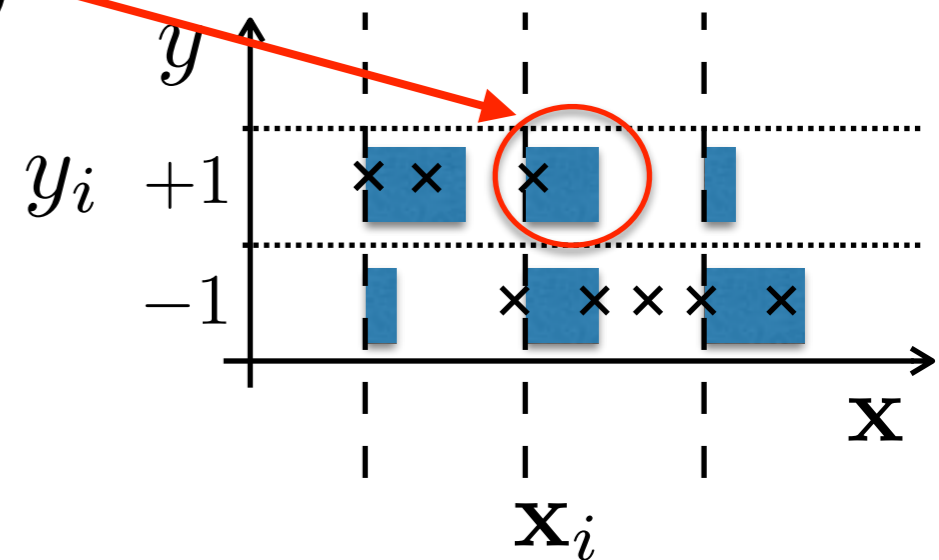
loss function

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

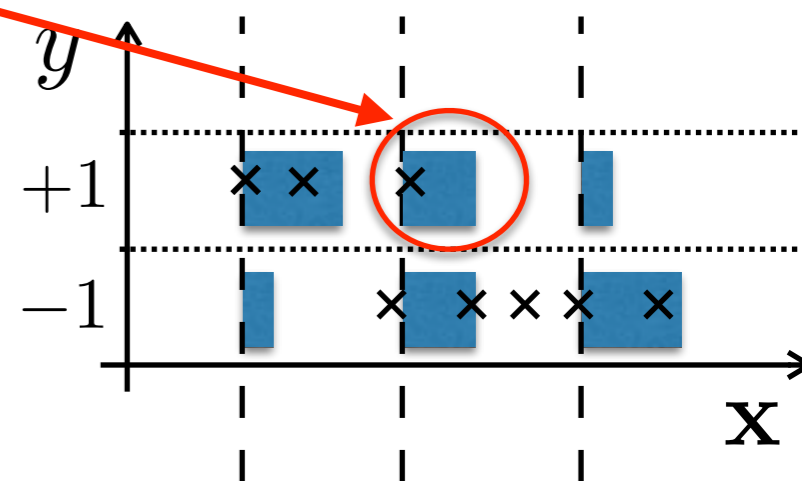
prior/regulariser

- **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?



Logistic loss vs cross entropy loss

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

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

=

Logistic loss

$$\log[1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w}))]$$

=

Cross-entropy loss

$$-\left[\bar{y}_i \cdot \log(\sigma(y_i f(\mathbf{x}_i, \mathbf{w}))) + (1 - \bar{y}_i) \cdot \log(1 - \sigma(y_i f(\mathbf{x}_i, \mathbf{w}))) \right]$$

$$\bar{y}_i = \frac{y_i + 1}{2} \in \{0, 1\}$$



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

loss function

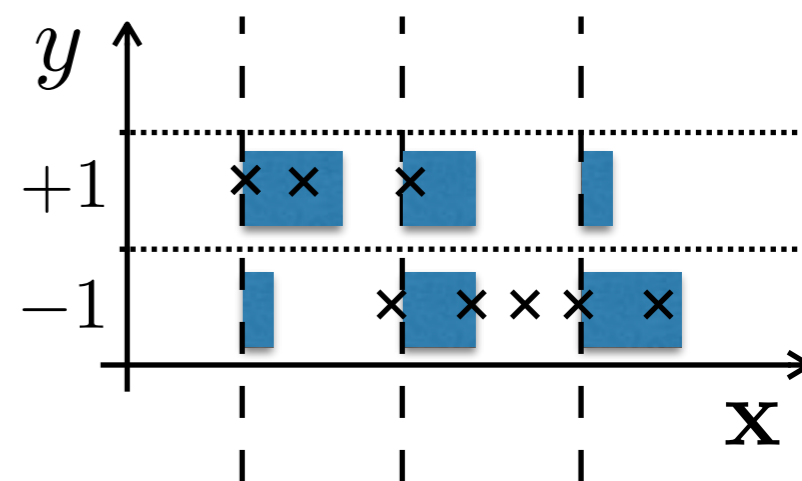
prior/regulariser

- **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

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



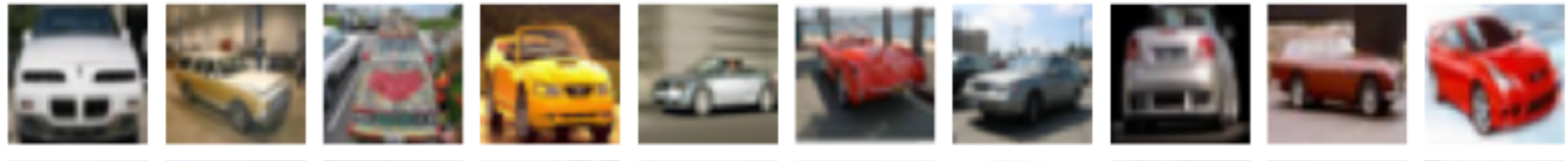
Labels (y_i)

RGB images (\mathbf{x}_i)

+1









-1



Training

Example: Training linear classifier

```
def train(       =  
          +1 +1 +1 -1 -1 -1 ):
```

$\mathbf{x}_i = \text{vec}(\img alt="car" data-bbox="282 606 352 690"/>) \quad \forall_i$

```
return  $\mathbf{w}^*$ 
```



Labels (y_i)

RGB images (\mathbf{x}_i)

+1











-1



Training

Example: Training linear classifier

```
def train(       =  ):  
     $\mathbf{x}_i = \text{vec}(\text{  }) \quad \forall_i$   
     $\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_i \log [1 + \exp(-y_i \mathbf{w}^\top \bar{\mathbf{x}}_i)]$   
    return  $\mathbf{w}^*$ 
```



Labels (y_i)

RGB images (\mathbf{x}_i)

+1









-1



Training

Example: Training linear classifier

```
def train(       =  
+1 +1 +1 -1 -1 -1 ):
```

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

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

return \mathbf{w}^*

-2.5

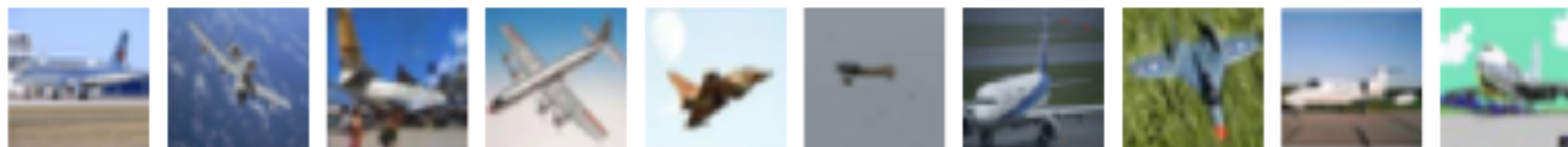
Small $\mathbf{w}^\top \bar{\mathbf{x}}_i$
while $y_i = -1$



Labels (y_i)

RGB images (\mathbf{x}_i)

+1







-1



Training

Example: Training linear classifier

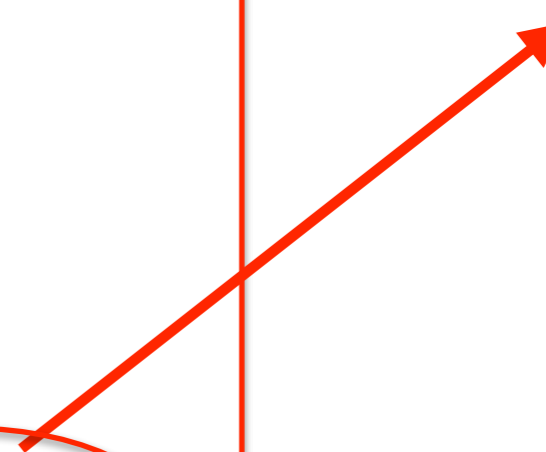
```
def train(         
          +1 +1 +1 -1 -1 -1 ):
```

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

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

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 ):
```

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

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

return \mathbf{w}^*

0.03

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



Labels (y_i)

RGB images (\mathbf{x}_i)

+1









-1



Training

Example: Training linear classifier

```
def train(       =  
+1 +1 +1 -1 -1 -1 ):
```

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

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

return \mathbf{w}^*

2.5

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



Labels (y_i)

RGB images (\mathbf{x}_i)

+1






-1



Training

Example: Training linear classifier

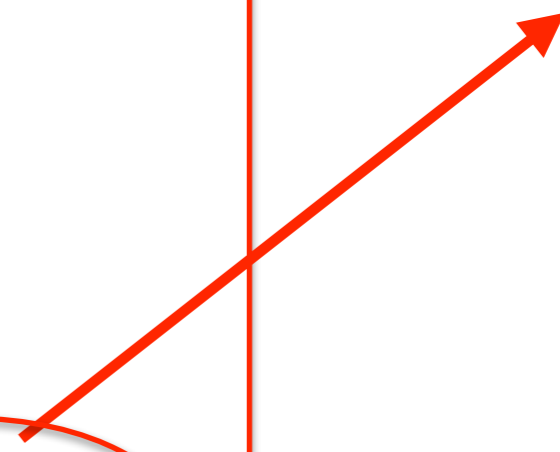
```
def train(       $\bar{\mathbf{x}} =$   $(+1 \ +1 \ +1 \ -1 \ -1 \ -1)$ ):
```

$\mathbf{x}_i = \text{vec}(\text{img alt="car" data-bbox="278 604 348 694"} \mid \text{img alt="car" data-bbox="354 604 424 694"})) \quad \forall_i$

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

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 ):
```

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

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

return \mathbf{w}^*

1.12

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



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



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

- There is no closed-form solution
- Gradient optimization

$$\mathbf{w} = \mathbf{w} - \alpha \left[\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \right]^\top \text{ where } \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = ?$$



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

$$\mathcal{L}(\mathbf{w})$$

- There is no closed-form solution
- Gradient optimization

$$\mathbf{w} = \mathbf{w} - \alpha \left[\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \right]^\top \text{ where } \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \frac{-y_i \bar{\mathbf{x}}_i^\top}{1 + \exp(y_i \mathbf{w}^\top \bar{\mathbf{x}}_i)}$$



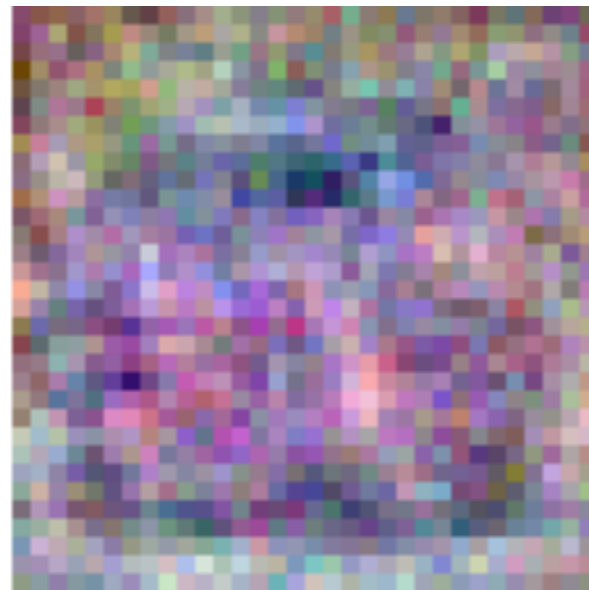
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_i \log [1 + \exp(-y_i \mathbf{w}^\top \bar{\mathbf{x}}_i)]$$

$$\mathcal{L}(\mathbf{w})$$

- There is no closed-form solution
- Gradient optimization

$$\mathbf{w} = \mathbf{w} - \alpha \left[\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \right]^\top \text{ where } \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \frac{-y_i \bar{\mathbf{x}}_i^\top}{1 + \exp(y_i \mathbf{w}^\top \bar{\mathbf{x}}_i)}$$

Learned weights
as a template:



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



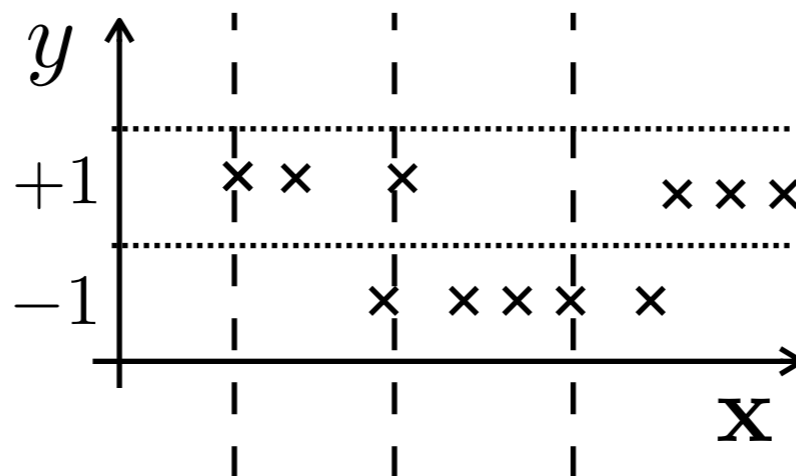
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

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

1D example:



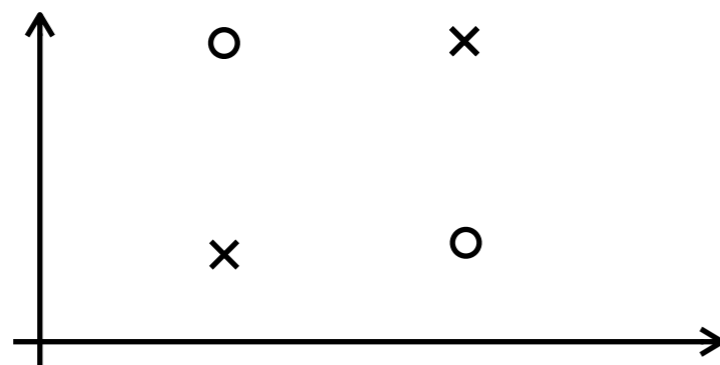
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function

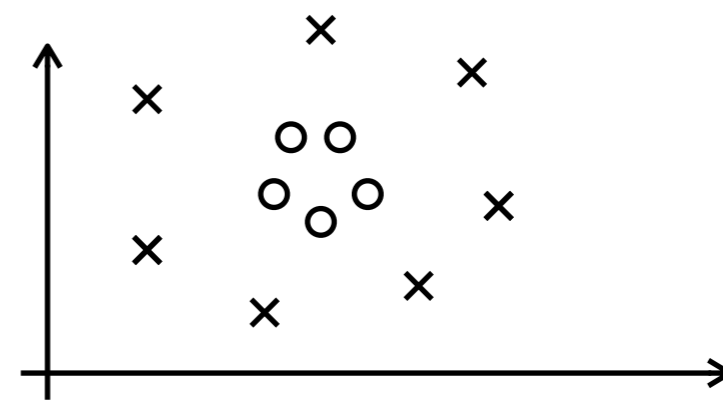
prior/regulariser

- **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

2D example:



XOR



circle

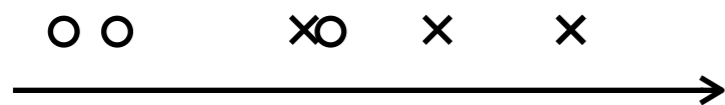


$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

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

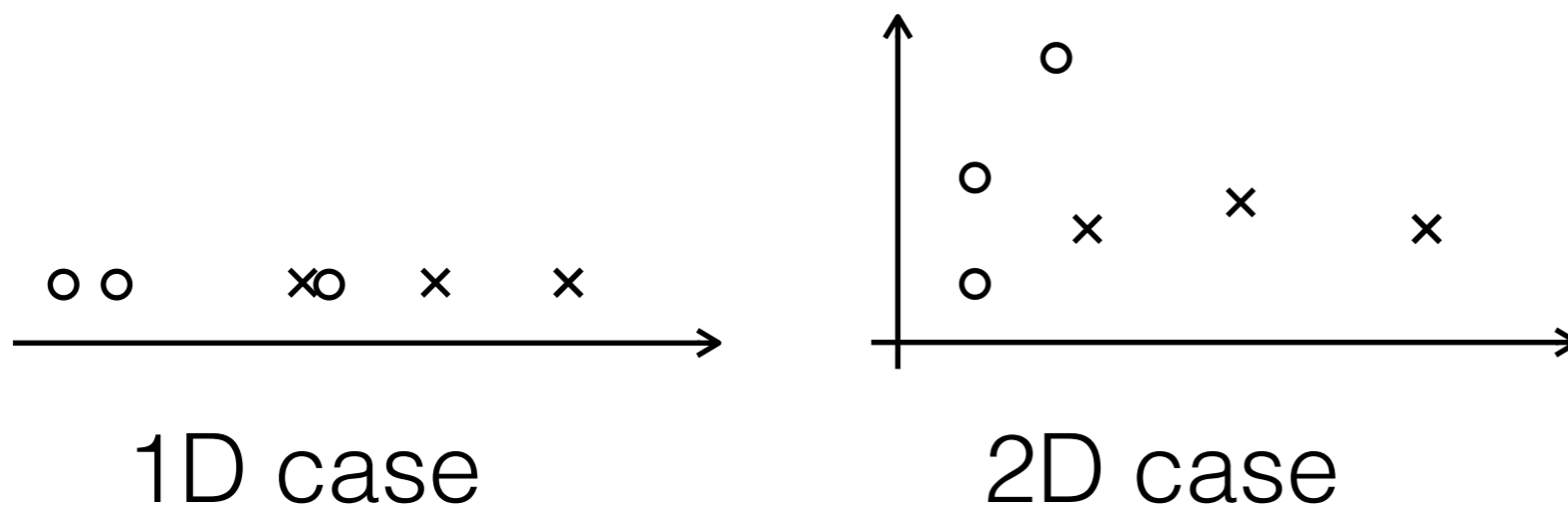


$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

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



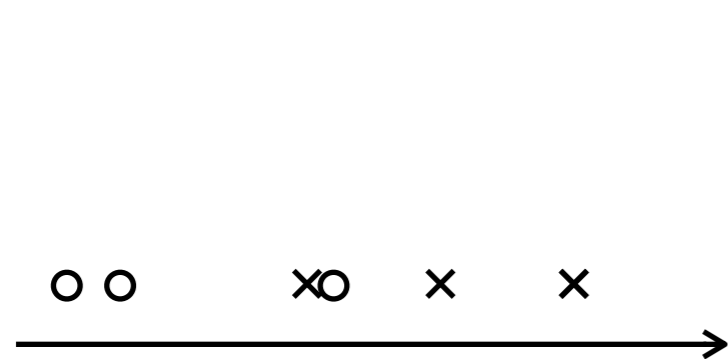
$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left(\sum_i -\log(p(y_i | \mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function

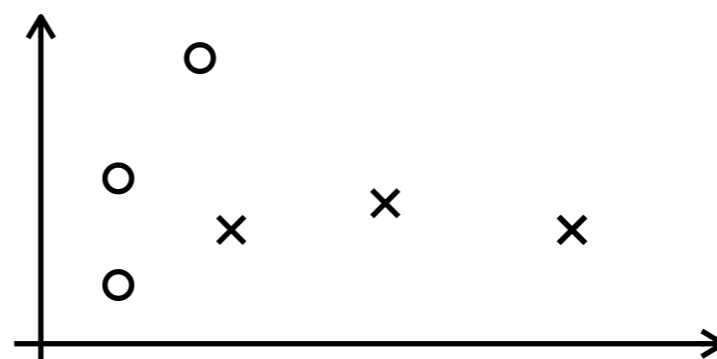
prior/regulariser

- **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



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

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



Multi-class classification problem

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



Logistic loss

$$\log [1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w}))]$$



Multi-class classification problem

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

Logistic loss

$$\log [1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w}))]$$

=

Cross-entropy loss

$$-\left[\bar{y}_i \cdot \log(\sigma(y_i f(\mathbf{x}_i, \mathbf{w}))) + (1 - \bar{y}_i) \cdot \log(1 - \sigma(y_i f(\mathbf{x}_i, \mathbf{w}))) \right]$$

$$\bar{y}_i = \frac{y_i + 1}{2} \in \{0, 1\}$$



Multi-class classification problem

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

Logistic loss

$$\log [1 + \exp(-y_i f(\mathbf{x}_i, \mathbf{w}))]$$

=

Cross-entropy loss

$$-\left[\bar{y}_i \cdot \log(\sigma(y_i f(\mathbf{x}_i, \mathbf{w}))) + (1 - \bar{y}_i) \cdot \log(1 - \sigma(y_i f(\mathbf{x}_i, \mathbf{w}))) \right]$$

$$\bar{y}_i = \frac{y_i + 1}{2} \in \{0, 1\}$$


=

$$-\log \left[\begin{array}{c} \sigma(f(\mathbf{x}_i, \mathbf{w})) \\ 1 - \sigma(f(\mathbf{x}_i, \mathbf{w})) \end{array} \right]_{\bar{y}_i}$$



Multi-class classification problem

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


$$-\log \left[\begin{array}{c} \sigma(f(\mathbf{x}_i, \mathbf{w})) \\ 1 - \sigma(f(\mathbf{x}_i, \mathbf{w})) \end{array} \right]_{\bar{y}_i}$$

Two-class
classification
problem

$$-\log \left[\begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_k \end{array} \right]_{y_i}$$

For k-class
classification
problem



Labels (y_i)

RGB images (\mathbf{x}_i)

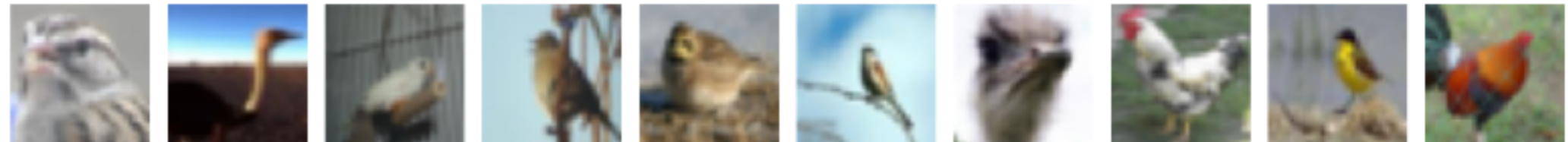
1



2



3



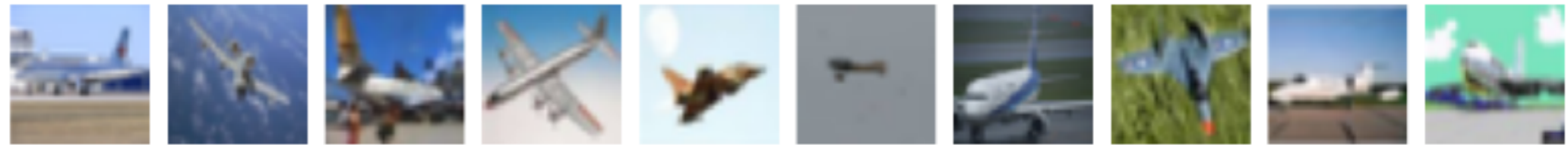
Three-class recognition problem:



Labels (y_i)

RGB images (\mathbf{x}_i)

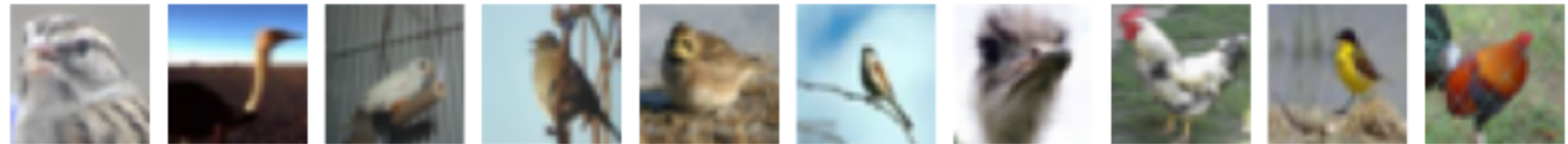
1



2



3



Three-class recognition problem:

```
def classify():
```

```
    ???
```

```
    return  $\mathbf{p}$ 
```



Labels (y_i)

RGB images (\mathbf{x}_i)

1



2



3



Model probability distribution over classes by softmax function

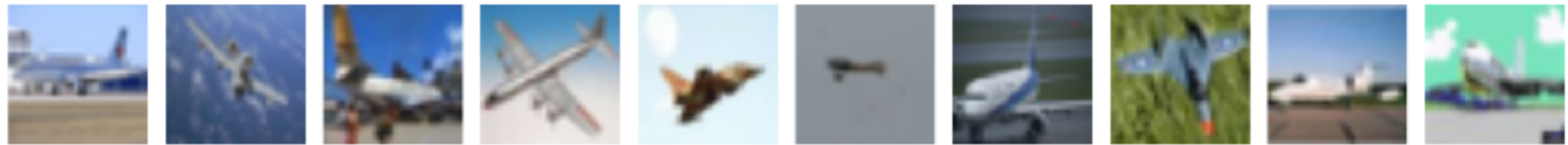
$$p(y|\mathbf{x}, W) = \frac{\begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \\ \exp(f(\mathbf{x}, \mathbf{w}_3)) \end{bmatrix}}{\sum_k \exp(f(\mathbf{x}, \mathbf{w}_k))} = \mathbf{s}(\mathbf{f}(\mathbf{x}, W))$$



Labels (y_i)

RGB images (\mathbf{x}_i)

1




2



3



Three-class recognition problem:

```
def classify()  
     $\mathbf{p} = \mathbf{s}(W \bar{\mathbf{x}})$   
    return  $\mathbf{p}$ 
```

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

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



Labels (y_i)

RGB images (\mathbf{x}_i)

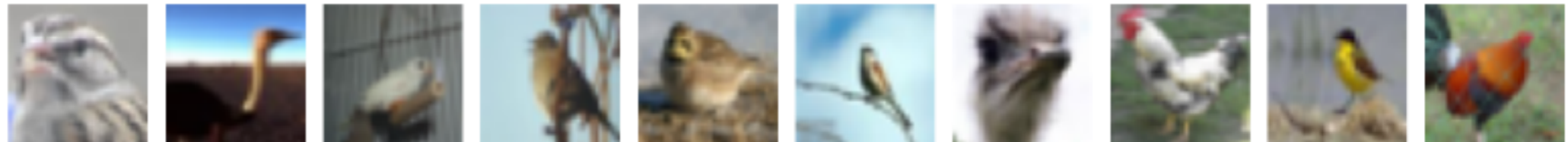
1



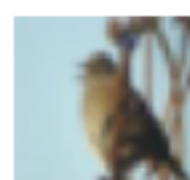
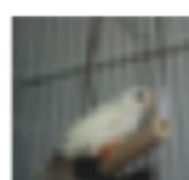
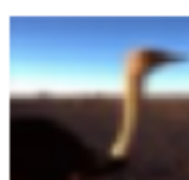
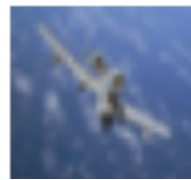
2



3



```
def train(
```



1

1

1

2

2

2

3

3

3

):

???

```
return  $W^*$ 
```



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

loss function

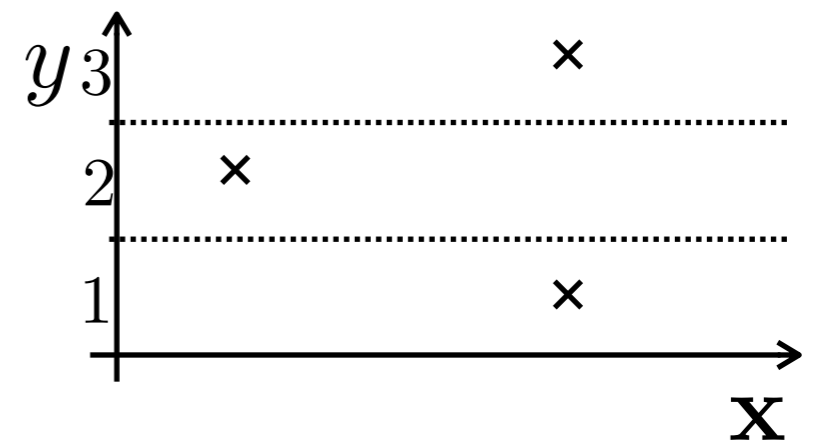
prior/regulariser

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

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

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

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



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

loss function

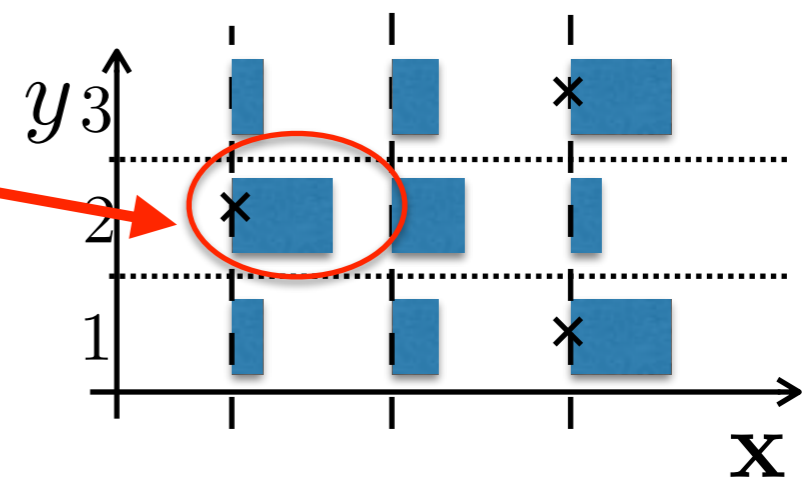
prior/regulariser

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

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

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

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



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

loss function
prior/regulariser

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

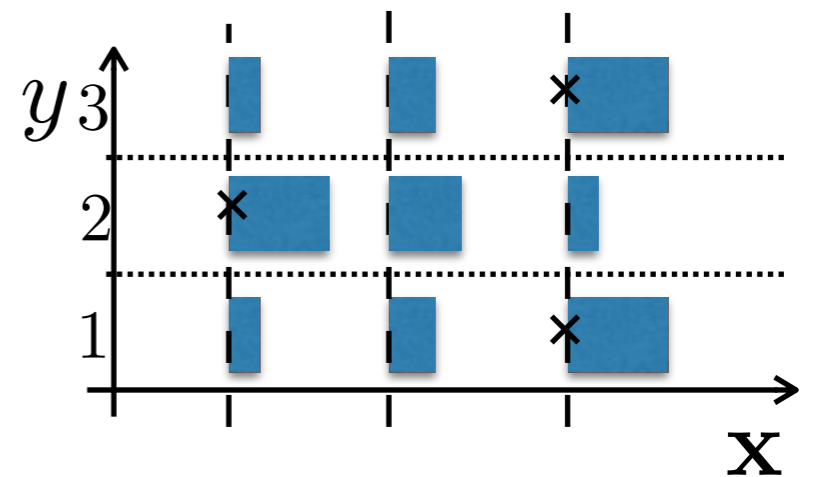
$$p(y|\mathbf{x}, W) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \\ \exp(f(\mathbf{x}, \mathbf{w}_3)) \end{bmatrix} / \sum_k \exp(f(\mathbf{x}, \mathbf{w}_k)) = \mathbf{s}(\mathbf{f}(\mathbf{x}, W))$$

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

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

- subst. yields cross-entropy loss

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



Labels (y_i)

RGB images (\mathbf{x}_i)

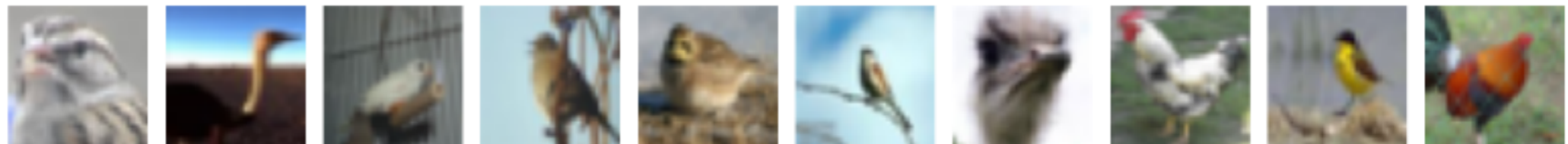
1



2



3



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

$\mathbf{x}_i = \text{vec}(\text{img alt="car" data-bbox="258 601 328 687"})$

$$W^* = \arg \min_W \sum_i -\log \mathbf{s}_{y_i}(W \bar{\mathbf{x}}_i)$$

return W^*





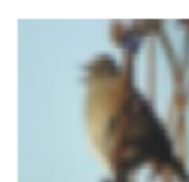
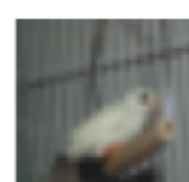
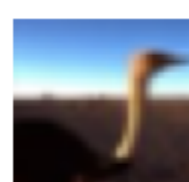
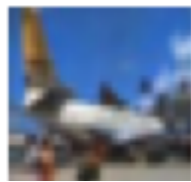
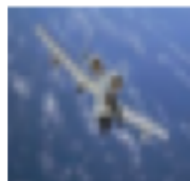
$$y_i = 2$$

$$\mathbf{s}(W \bar{\mathbf{x}}_i) = \begin{bmatrix} 0.03 \\ 0.71 \\ 0.26 \end{bmatrix}$$

$$\Rightarrow -\log \mathbf{s}_{y_i}(W \bar{\mathbf{x}}_i) = -\log(0.71) = 0.15$$

Car classified as car yields small loss

def train(



1

1

1

2

2

2

3

3

3

):

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



$$W^* = \arg \min_W \sum_i -\log \mathbf{s}_{y_i}(W \bar{\mathbf{x}}_i)$$

return W^*





$$y_i = 1$$

$$\mathbf{s}(W \bar{\mathbf{x}}_i) = \begin{bmatrix} 0.03 \\ 0.57 \\ 0.40 \end{bmatrix}$$

$$\Rightarrow -\log \mathbf{s}_{y_i}(W \bar{\mathbf{x}}_i) = -\log(0.03) = 1.52$$

Plane classified as car yields huge loss

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

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

$$W^* = \arg \min_W \sum_i -\log \mathbf{s}_{y_i}(W \bar{\mathbf{x}}_i)$$

return W^*



Conclusions

- Explained regression and linear classifier as MAP/ML estimator
- Discussed under/overfitting and regularisations
- Next lesson will go deeper

Competencies required for test

- Derive MAP/ML estimate for regression, two-class and three-class classification problem.
- Compute L2-loss, logistic-loss and entropy-loss (understand when it has high/low values).
-

