Linear image classification

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

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

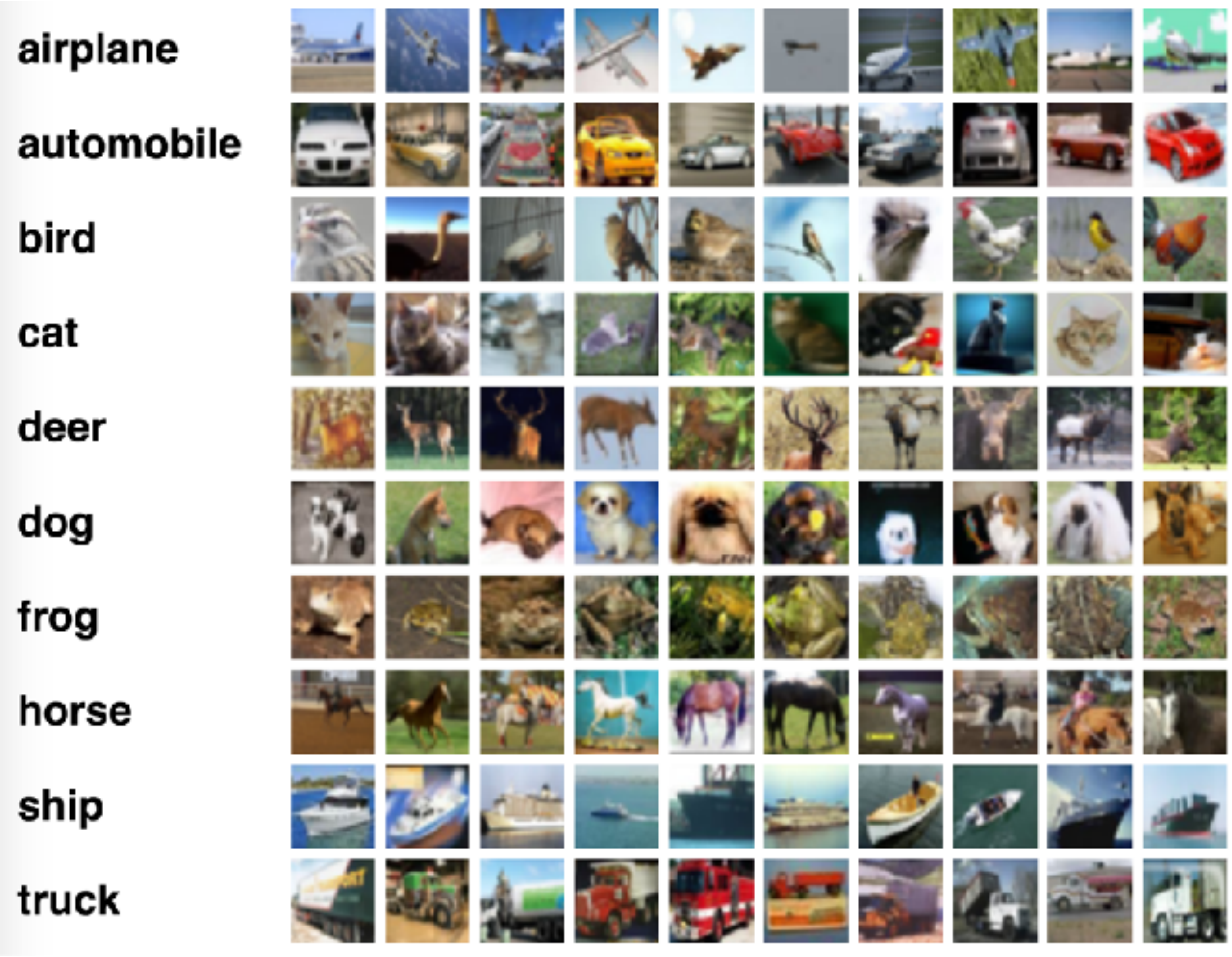
Center for Machine Perception
https://cmp.felk.cvut.cz

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

Recognition problem



Why is it hard?

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

Recognition problem

Huge within-class variability & among-class similarity !

Why it is hard?

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

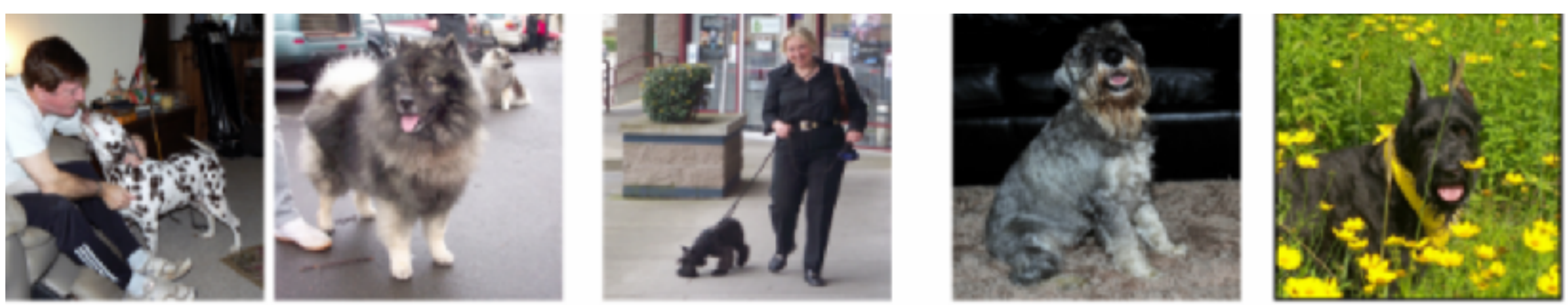
Recognition problem

Huge within-class variability & among-class similarity !

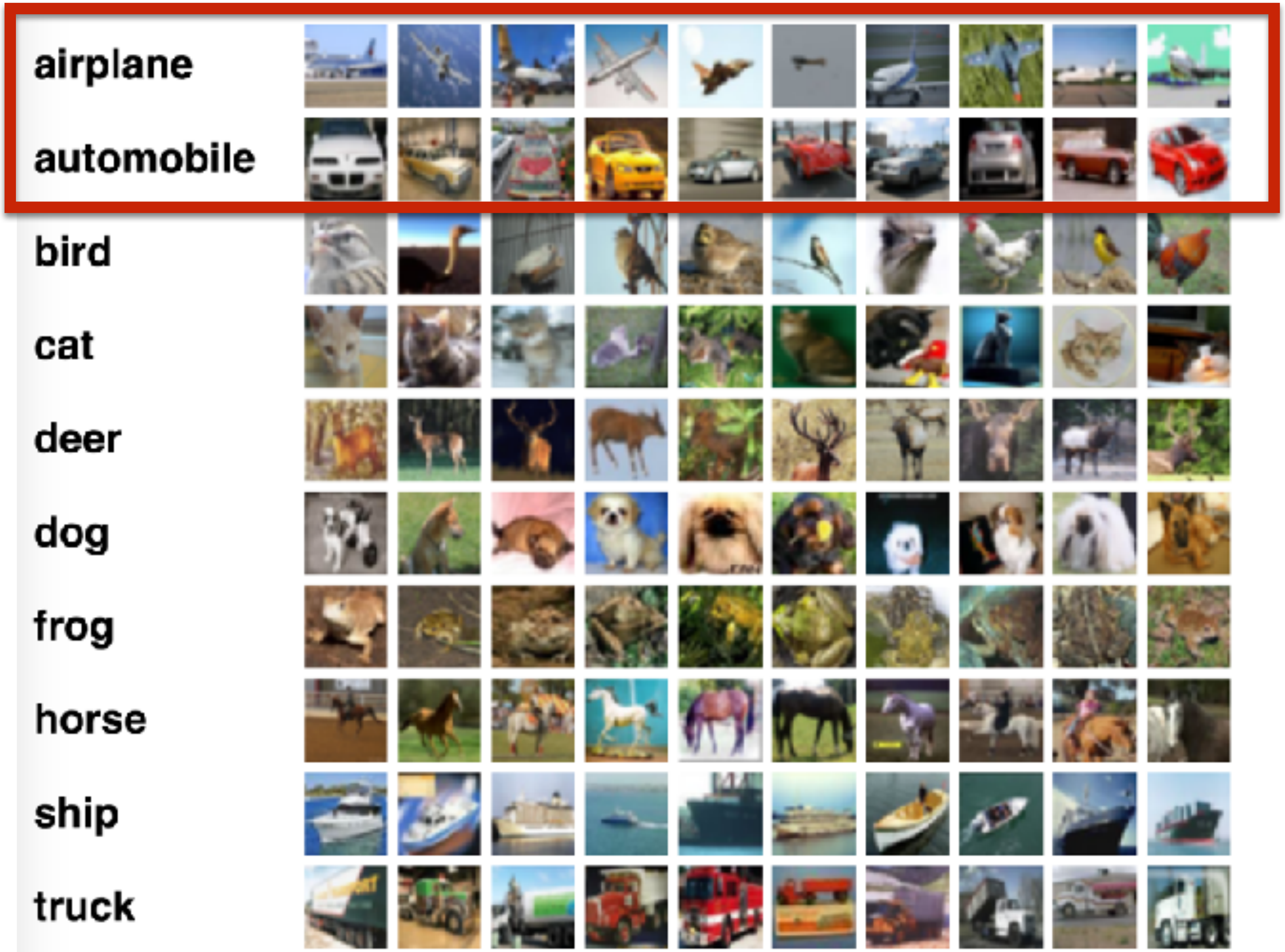
bird

cat

dog



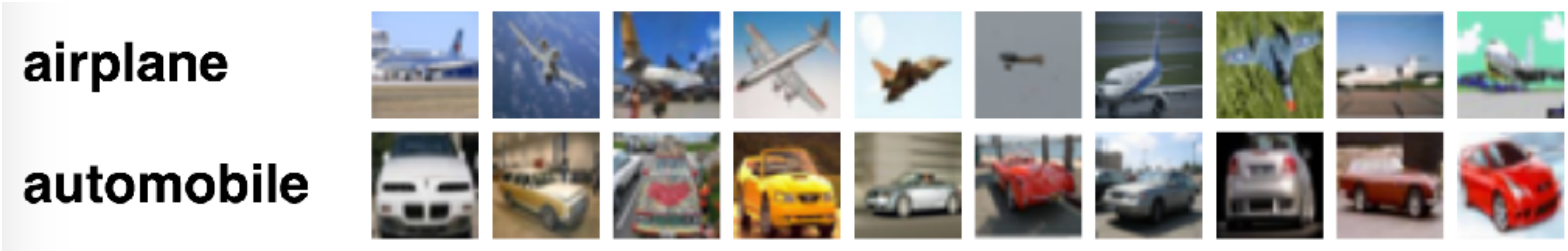
Recognition problem



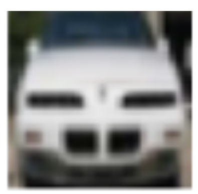
CIFAR-10: classify 32x32 RGB images into 10 categories

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

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



Two-class recognition problem: classify airplane/automobile

def classify():

    ???

    return p

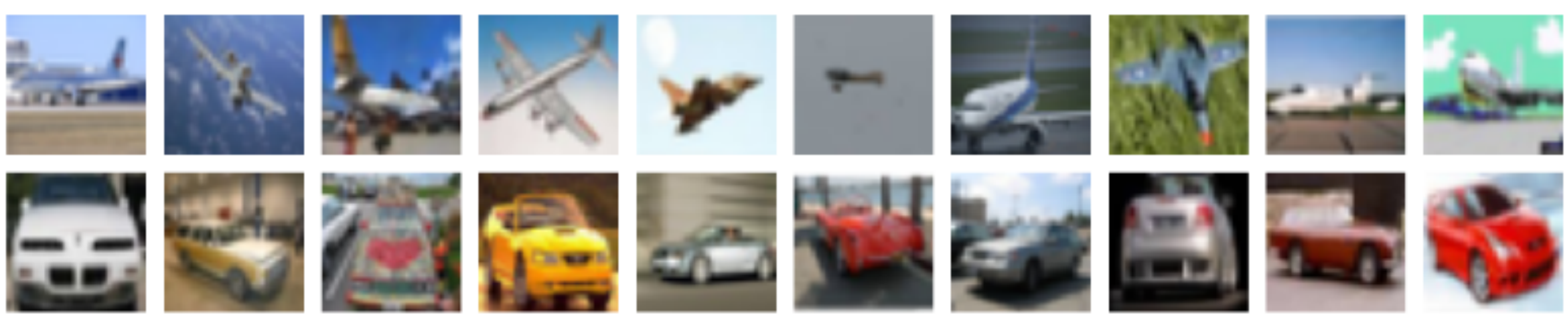
Probability of image being from the class airplane

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

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

Classification

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

$+1$

-1

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

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

where

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

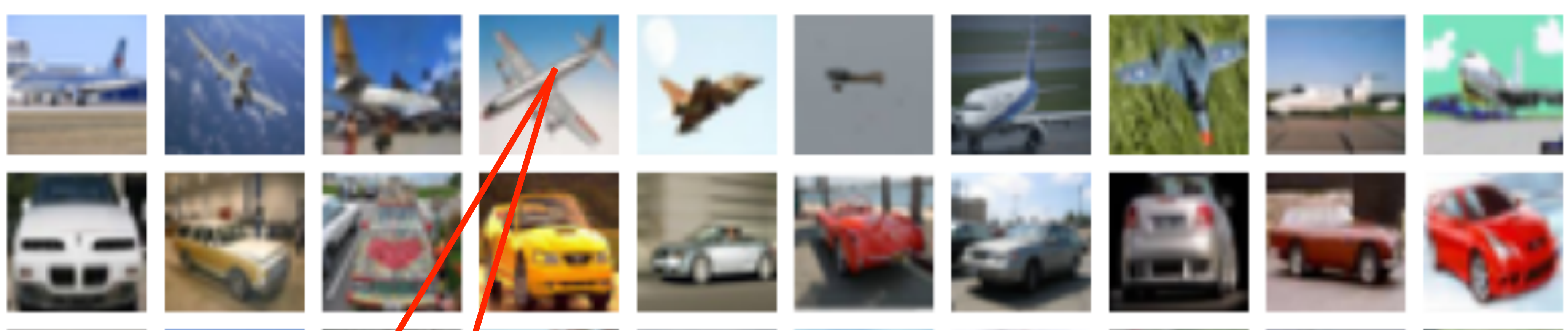
is sigmoid function.

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

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

$+1$

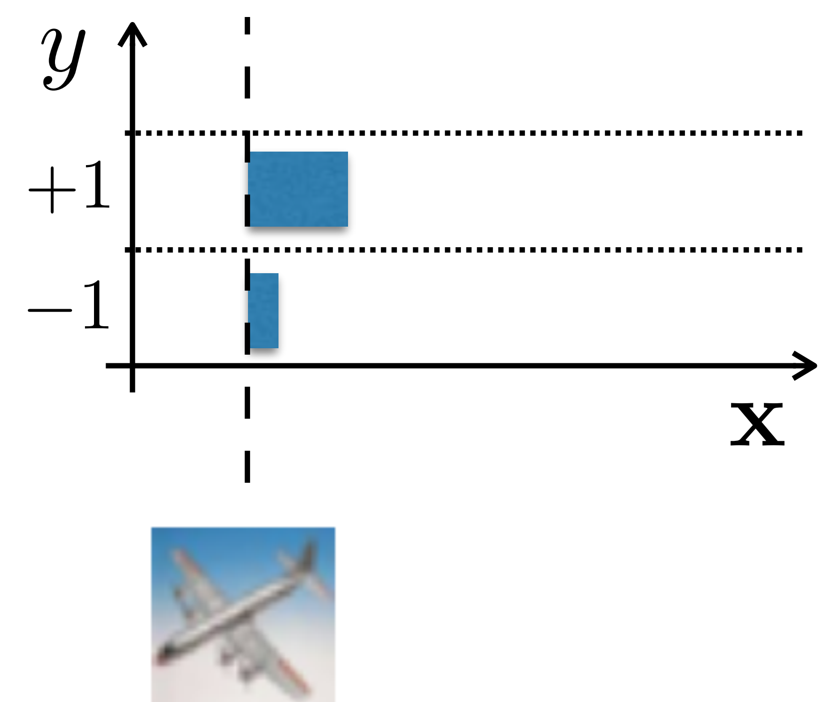
-1



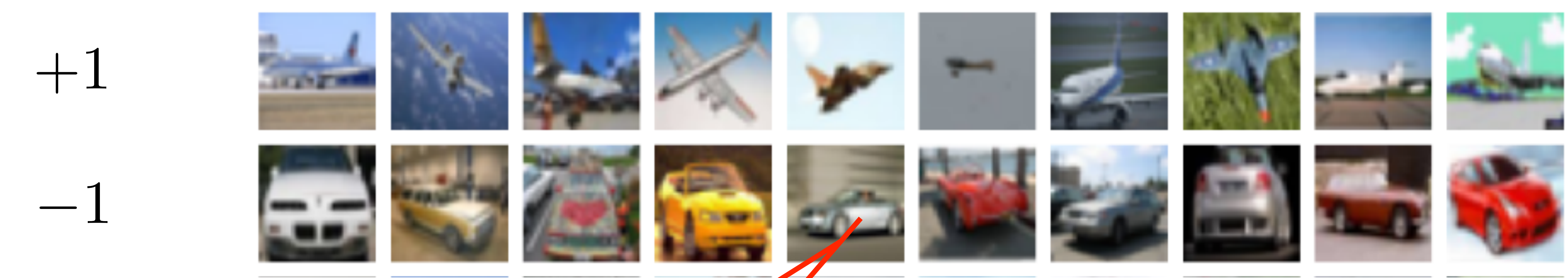
Classification

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

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



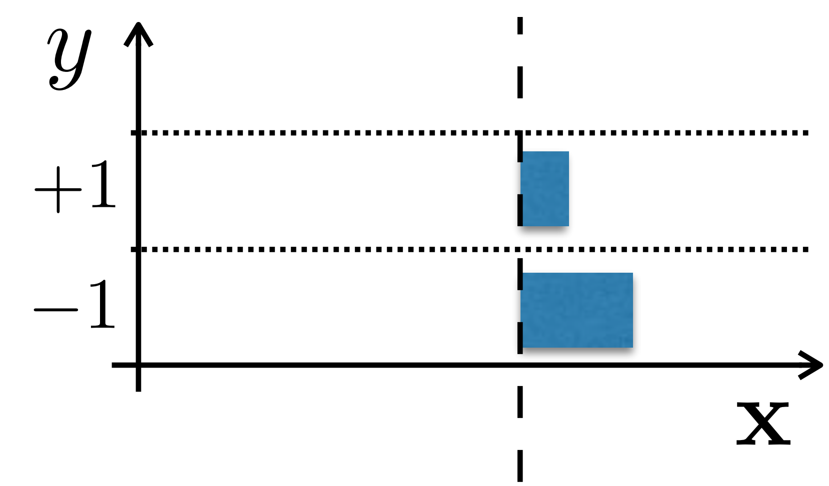
Labels (y_i) RGB images ($\mathbf{x}_i$)



Classification

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

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

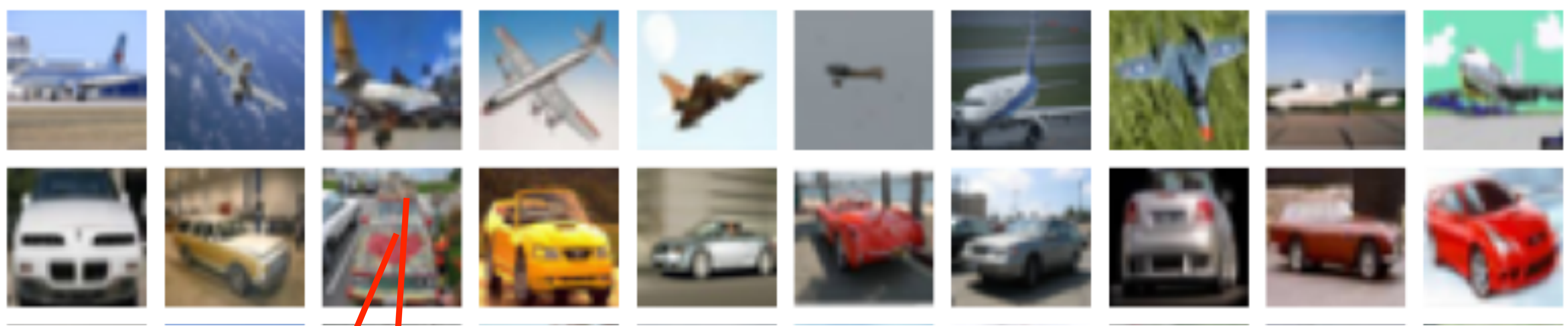


Classification

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

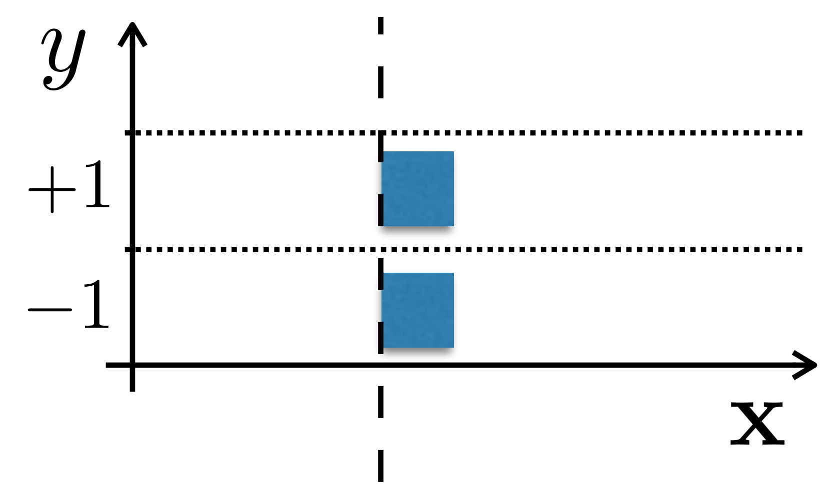
$+1$

-1



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

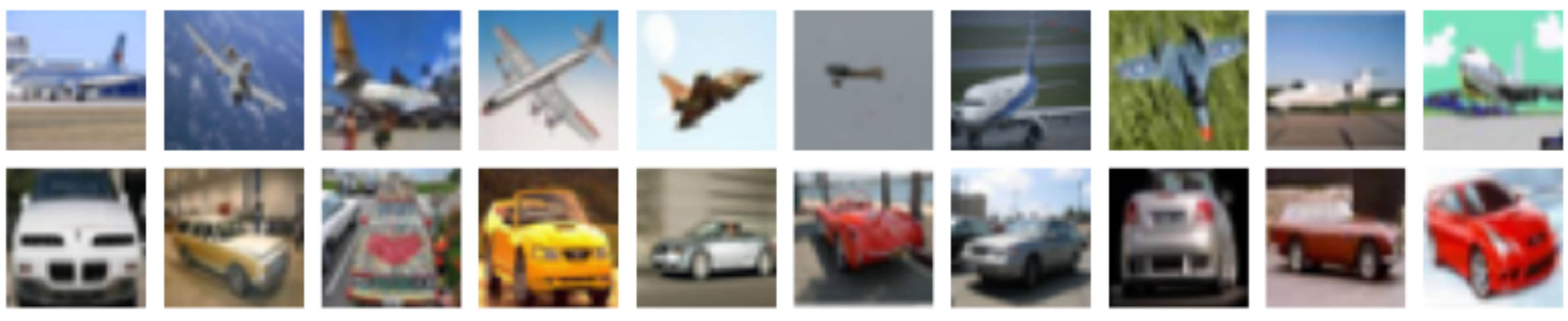
$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \end{cases}$$



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

$+1$

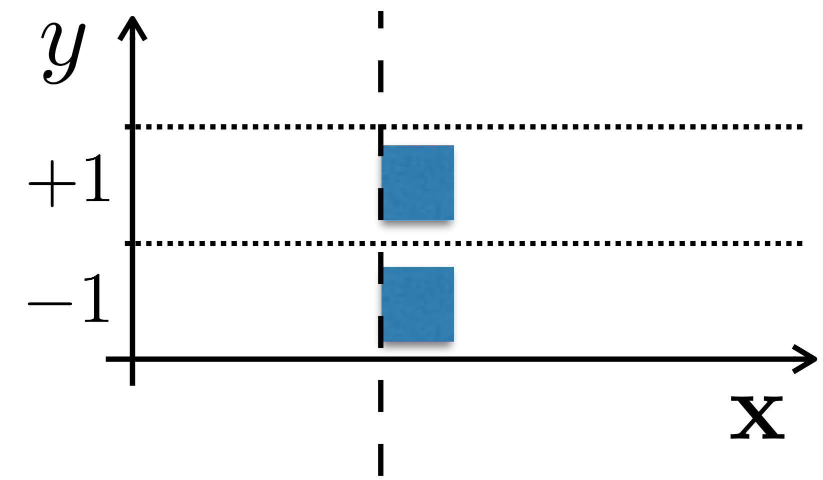
-1



Classification

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

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



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 $\mathbf{w}$?

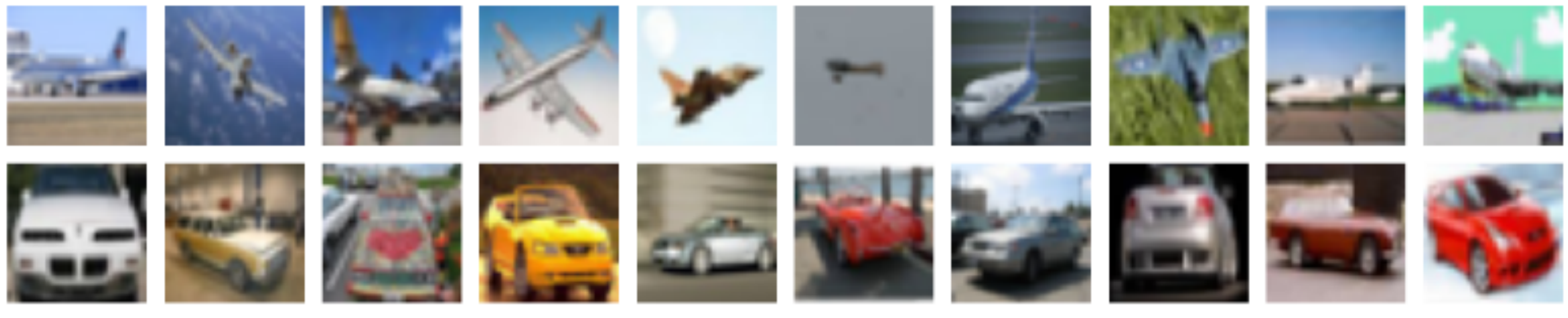
Classification

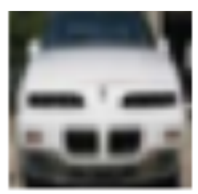
Example: Linear classifier

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

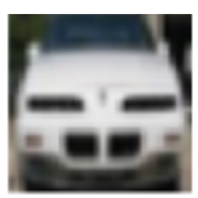
$+1$

-1



def classify():

 # Linear classifier

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

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

return p

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

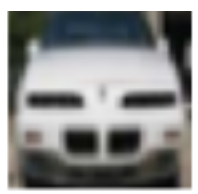
Classification

Example: Linear classifier

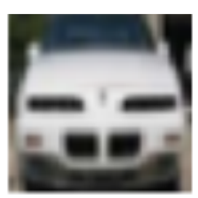
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

def classify():

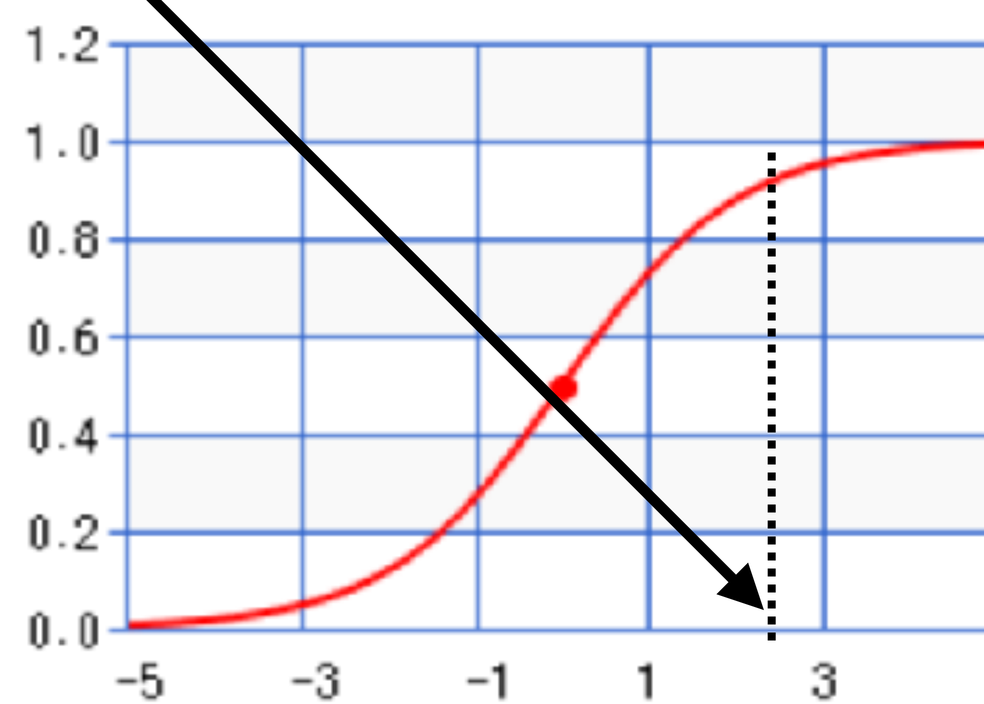
 # Linear classifier

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

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

return p

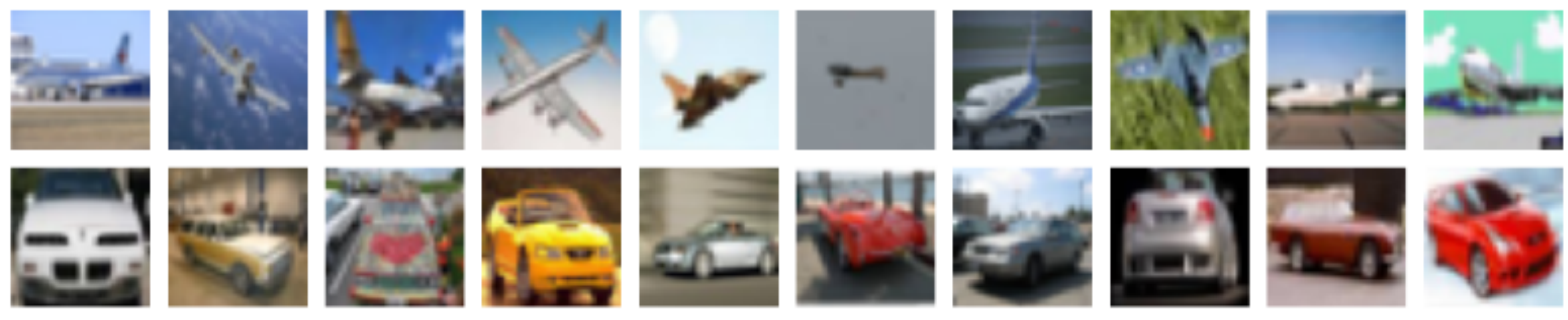
$\mathbf{w}^\top \overline{\mathbf{x}} = 2.5$ score



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

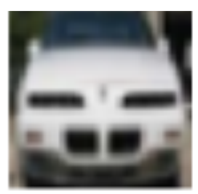
$+1$

-1

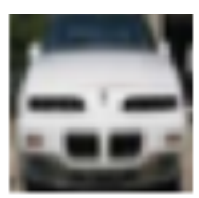


Classification

Example: Linear classifier

def classify():

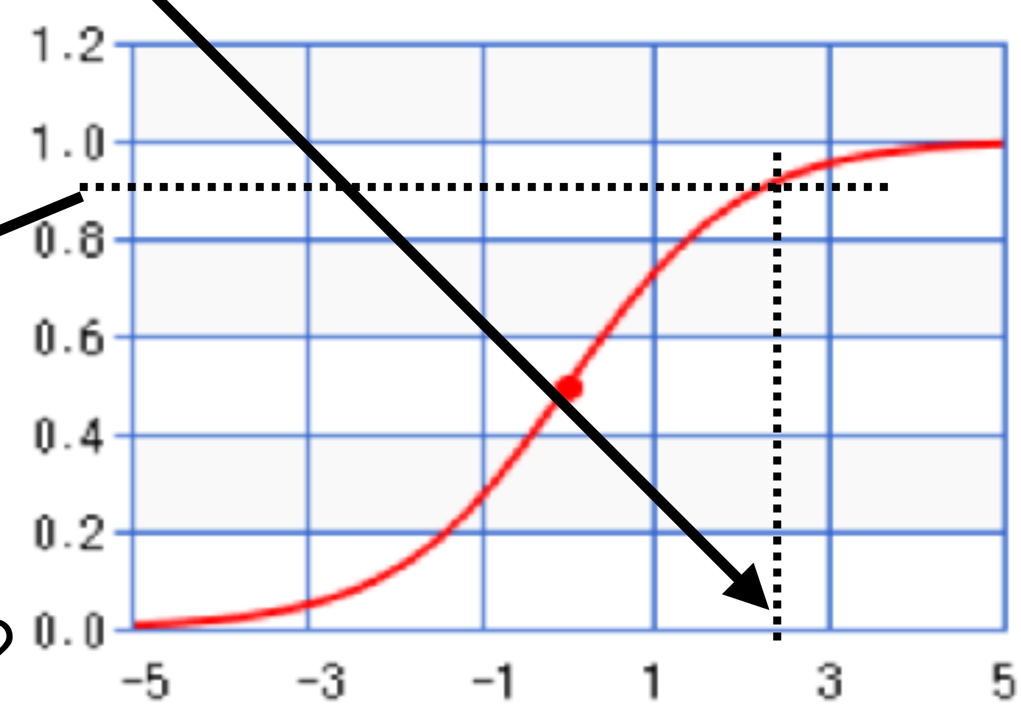
 # Linear classifier

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

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

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

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



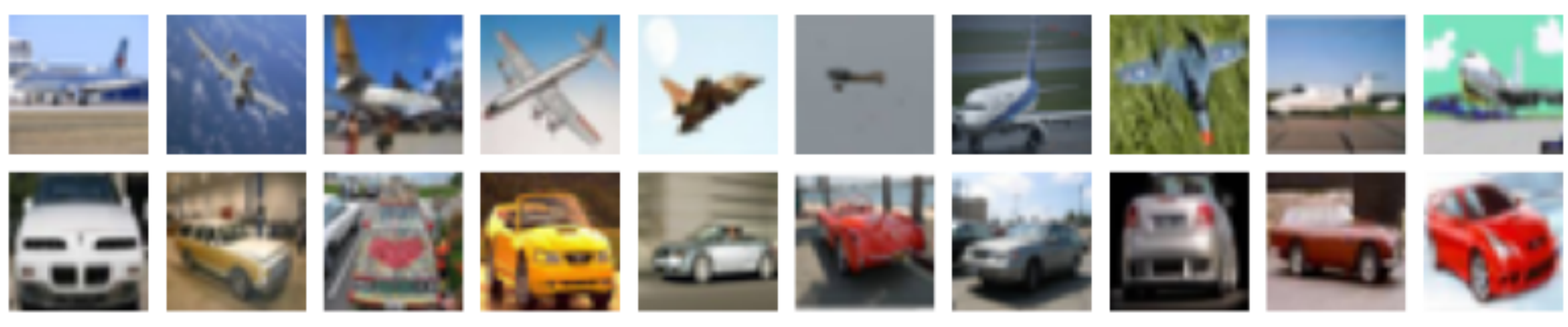
is it a good classifier?

Show python code

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

$+1$

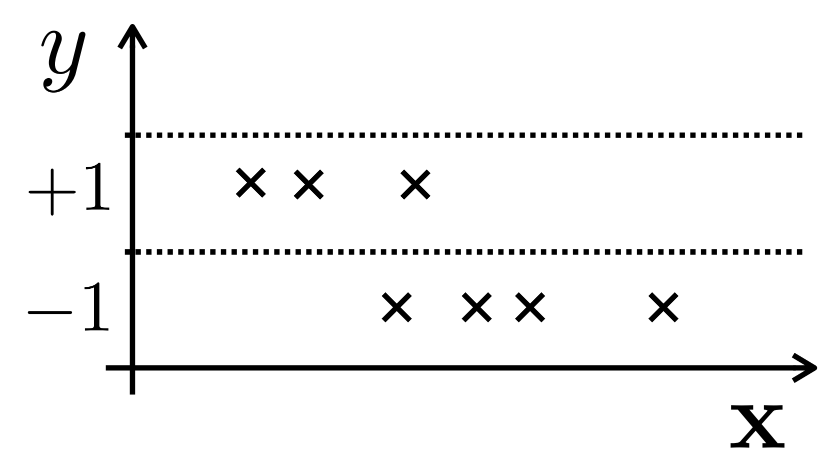
-1



Training

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

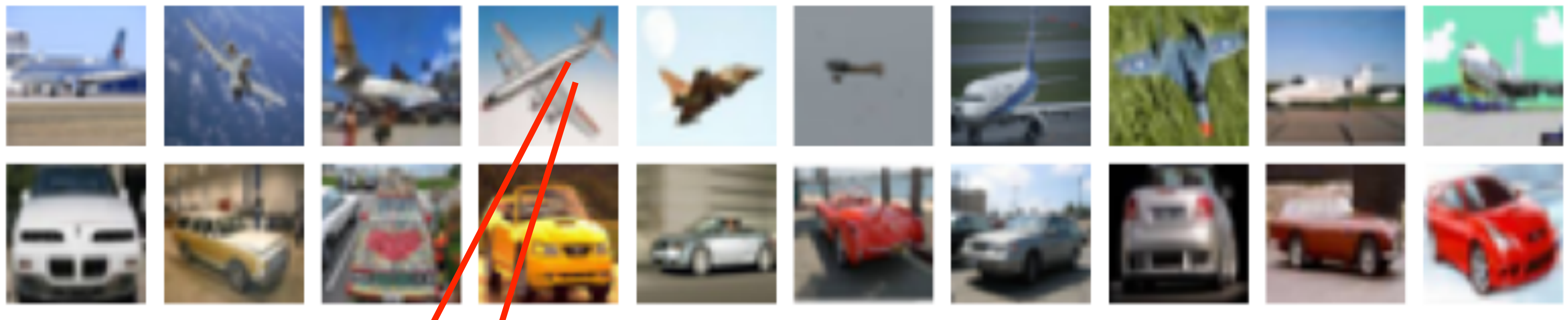
Training data



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

$+1$

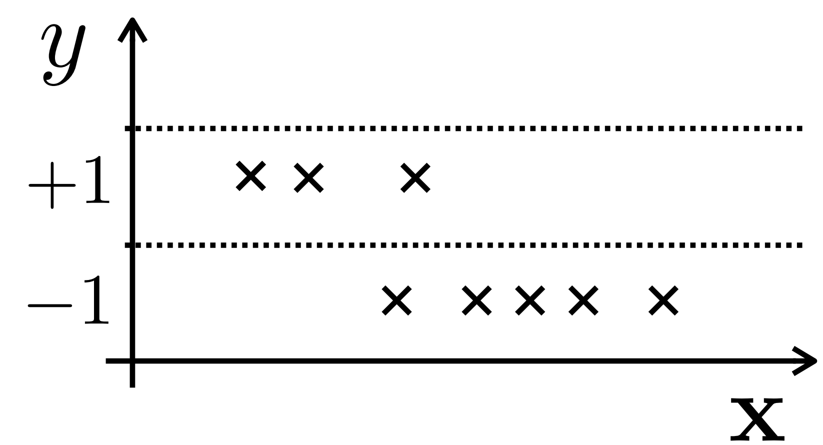
-1



Training

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

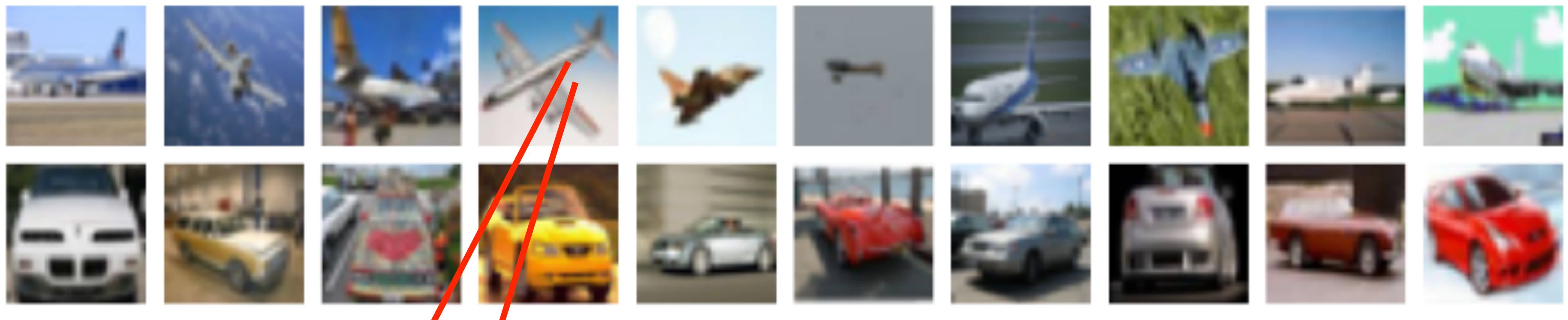
$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x},\mathbf{w})) & y = -1 \end{cases}$$



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

$+1$

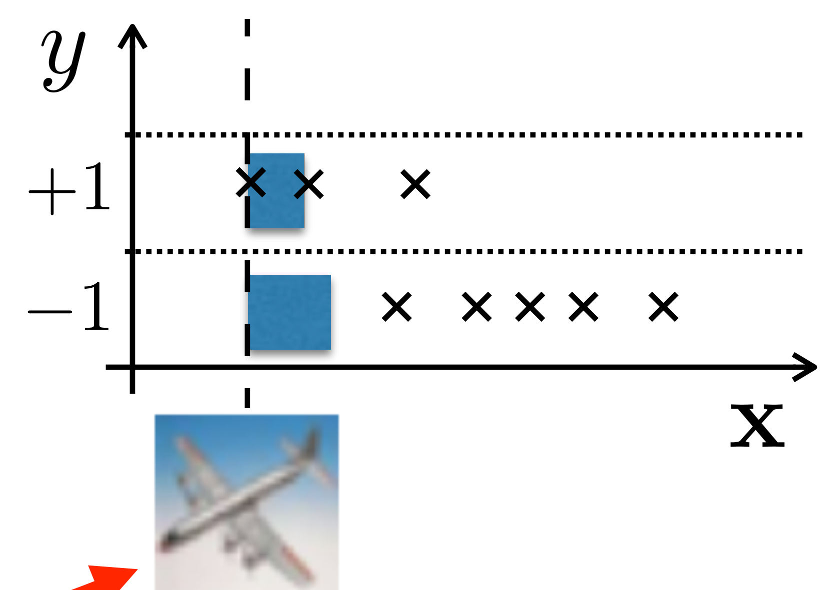
-1



Training

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

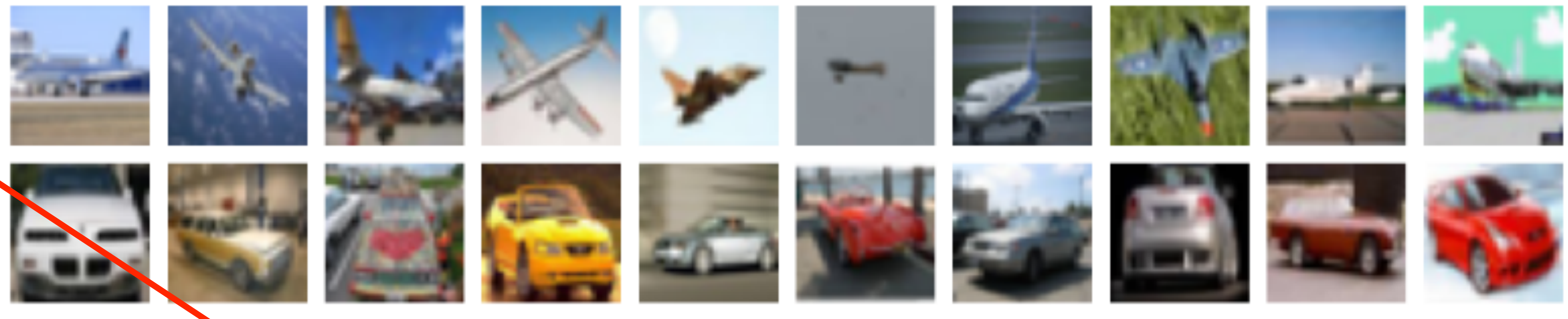
$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x},\mathbf{w})) & y = -1 \end{cases}$$



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

$+1$

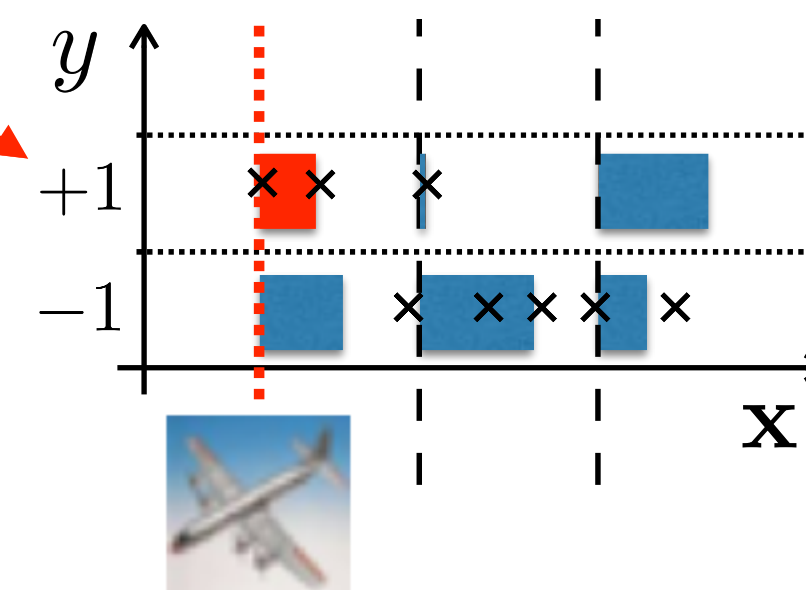
-1



Training

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

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

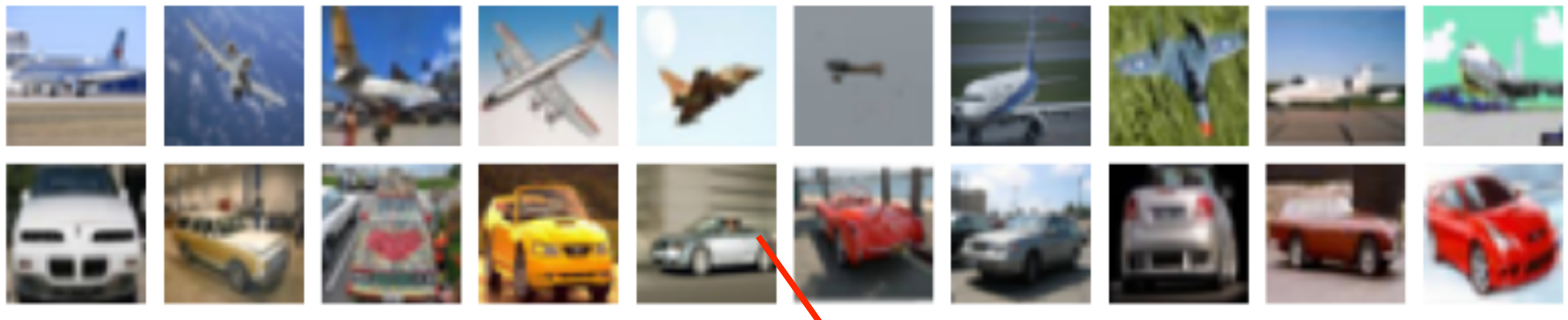


Probability of observing y_i when measuring $\mathbf{x}_i$ is

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

$+1$

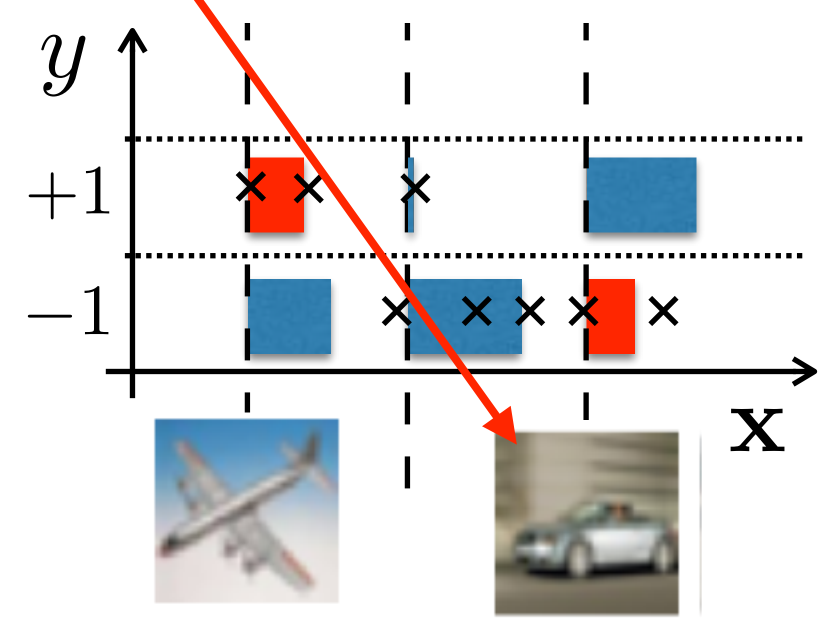
-1



Training

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

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

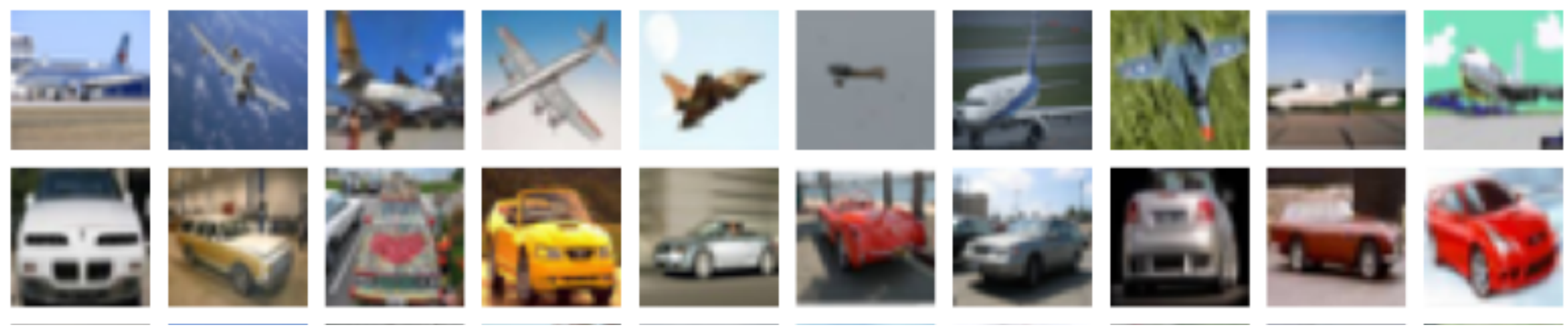


Probability of observing y_i when measuring $\mathbf{x}_i$ is

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

$+1$

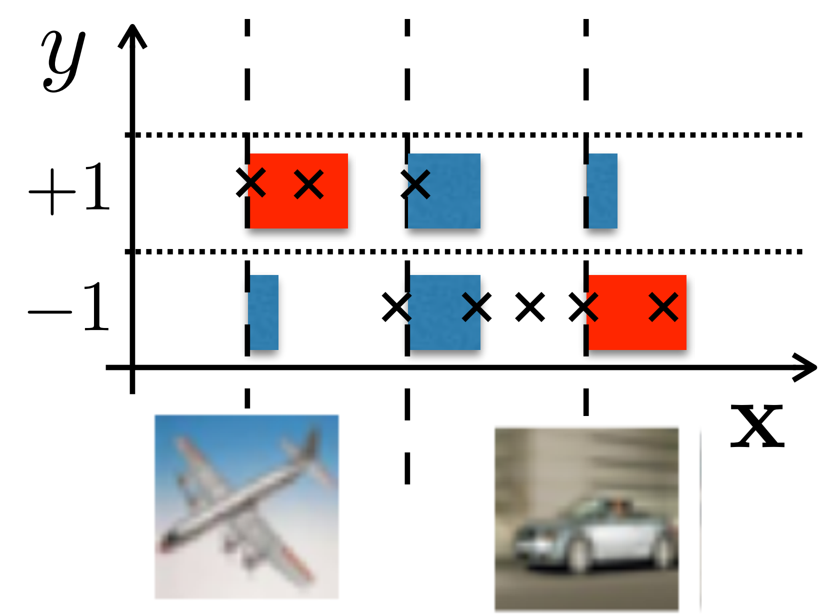
-1



Training

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

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

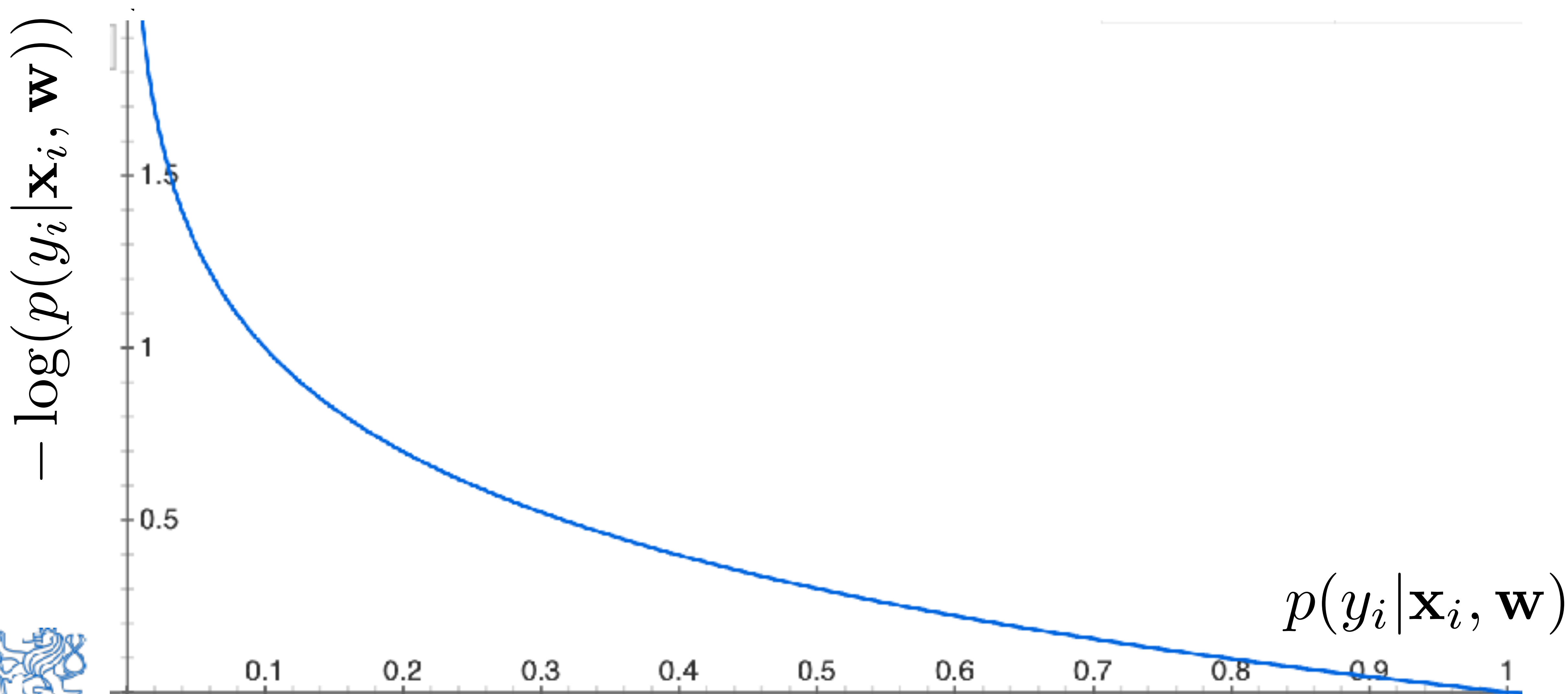


Probability of observing y_i when measuring $\mathbf{x}_i$ is

Two-class classification problem

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

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



Two-class classification problem

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

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

```
loss = 0
for each (x, y) from training set:
    p = sigmoid( f(x,w) )
    if y==1:
        loss = loss + -log(p)
    else:
        loss = loss + -log(1-p)
```

Equivalence of common binary classification losses

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

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

$$= -\log\left[\sigma\left(y_i\, f(\mathbf{x}_i, \mathbf{w})\right)\right]$$

$$\Leftrightarrow$$

Logistic loss

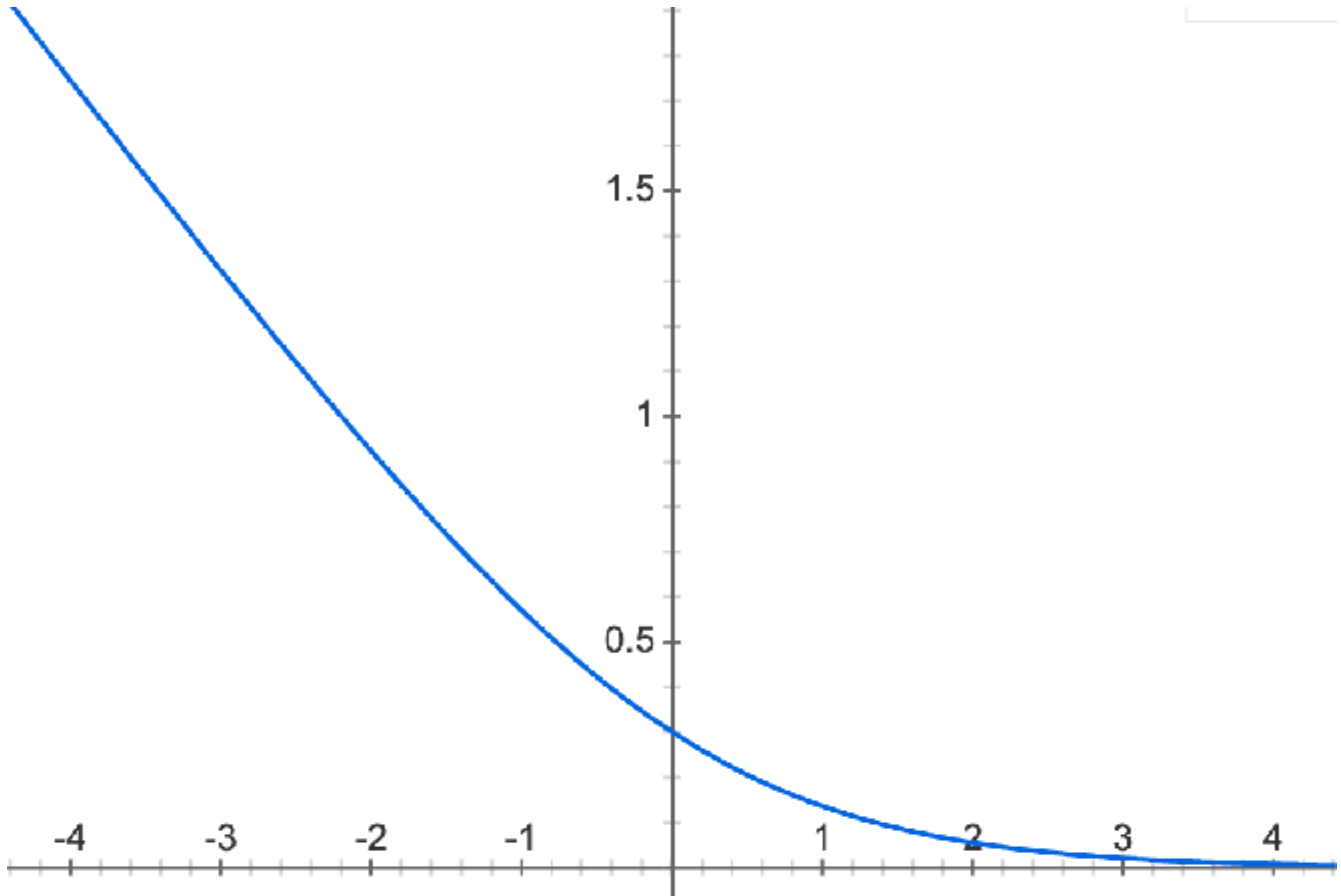
$$\log\left[1 + \exp(-y_i\, f(\mathbf{x}_i, \mathbf{w}))\right]$$

Equivalence of common binary classification losses

$\Leftrightarrow$

Logistic loss

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



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

Equivalence of common binary classification losses

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \quad \bar{y}=1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \quad \bar{y}=0 \end{cases}$$

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

```
loss = 0
for each (x, y) from training set:
    P = [1-sigmoid( f(x,w) );
         sigmoid( f(x,w) )]
    loss = loss + -log(P[y])
```

Equivalence of common binary classification losses

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \quad \bar{y}=1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \quad \bar{y}=0 \end{cases}$$

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

```
loss = 0
for each (x, y) from training set:
    P = [1-sigmoid( f(x,w) );
            sigmoid( f(x,w) )]
    loss = loss + -log(P[y])
```

$$\Leftrightarrow$$

$$-\log \begin{bmatrix} 1-\sigma(f(\mathbf{x}_i,\mathbf{w})) \\ \sigma(f(\mathbf{x}_i,\mathbf{w})) \end{bmatrix}_{\bar{y}_i}$$

Equivalence of common binary classification losses

$$p(y|\mathbf{x},\mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x},\mathbf{w})) & y=+1 \quad \bar{y}=1 \\ 1-\sigma(f(\mathbf{x},\mathbf{w})) & y=-1 \quad \bar{y}=0 \end{cases}$$

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

$$\Leftrightarrow$$

$$-\log \begin{bmatrix} 1-\sigma(f(\mathbf{x}_i,\mathbf{w})) \\ \sigma(f(\mathbf{x}_i,\mathbf{w})) \end{bmatrix}_{\bar{y}_i}$$

$$\Leftrightarrow$$

Cross-entropy loss

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

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

Equivalence of common binary classification losses

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

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

$$= -\log\left[\sigma\left(y_i\, f(\mathbf{x}_i, \mathbf{w})\right)\right]$$

$$\Leftrightarrow$$

Logistic loss

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

$$\Leftrightarrow$$

Cross-entropy loss

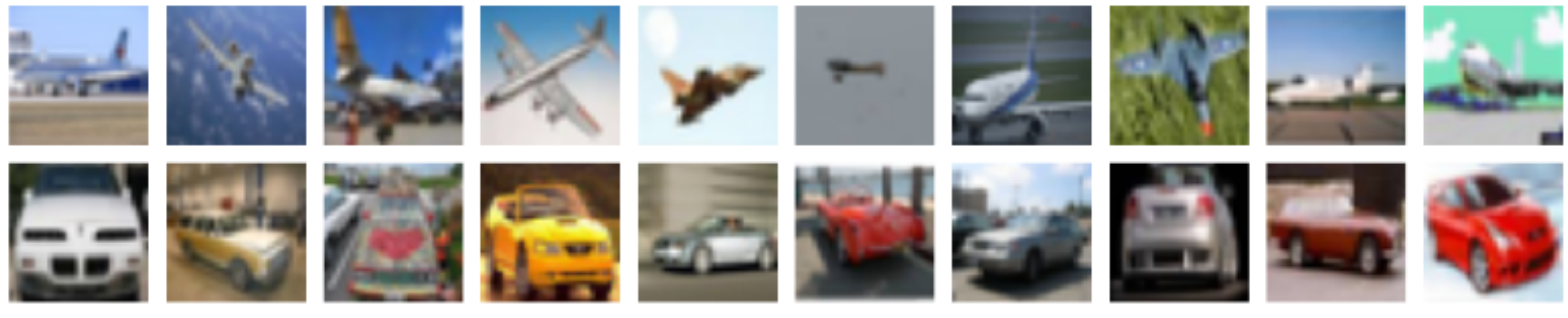
$$-\left[\ \bar{y}_i \cdot \log\left(\sigma(\ f(\mathbf{x}_i, \mathbf{w}))\right) + (1 - \bar{y}_i) \cdot \log\left(1 - \sigma(\ f(\mathbf{x}_i, \mathbf{w}))\right)\ \right]$$

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

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

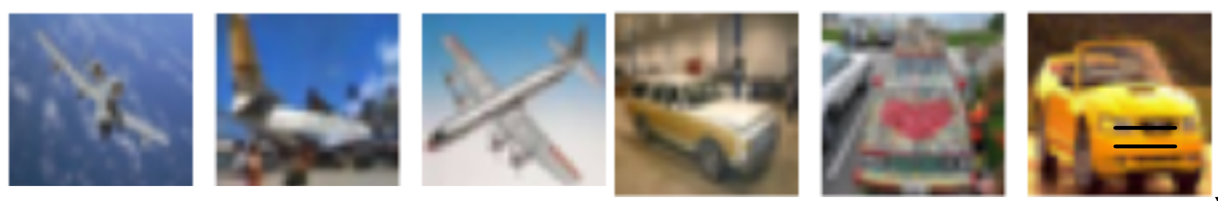
$+1$

-1



Training

Example: Training linear classifier

def train(

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

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

return $\mathbf{w}^*$

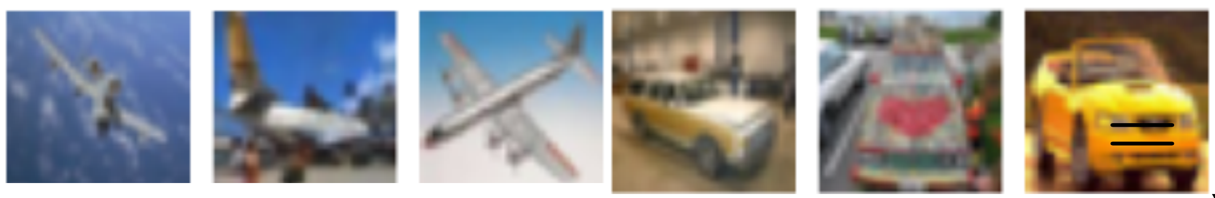
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(

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

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

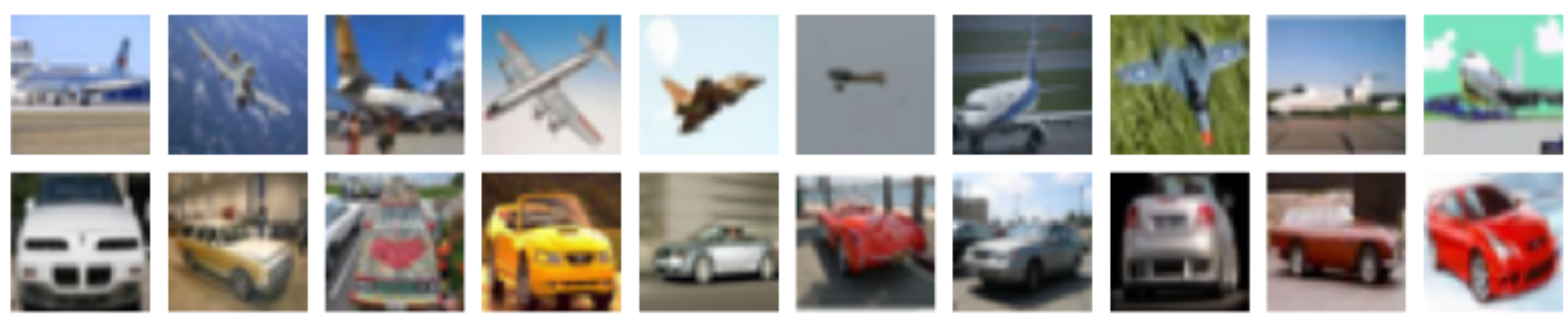
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \log\left[1 + \exp(-y_i\, \mathbf{w}^\top \overline{\mathbf{x}}_i)\right]$$

return $\mathbf{w}^*$

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

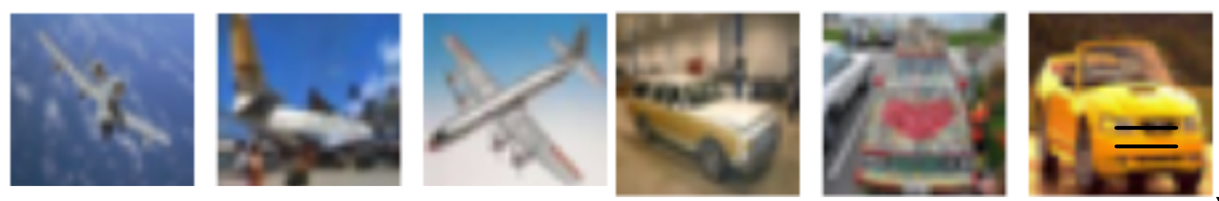
$+1$

-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 $\mathbf{w}^*$

score

-2.5

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

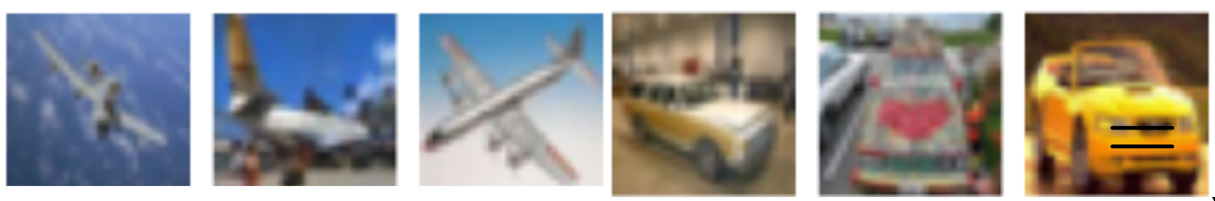
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(
$+1$ $+1$ $+1$ -1 -1 -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 $\mathbf{w}^*$

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

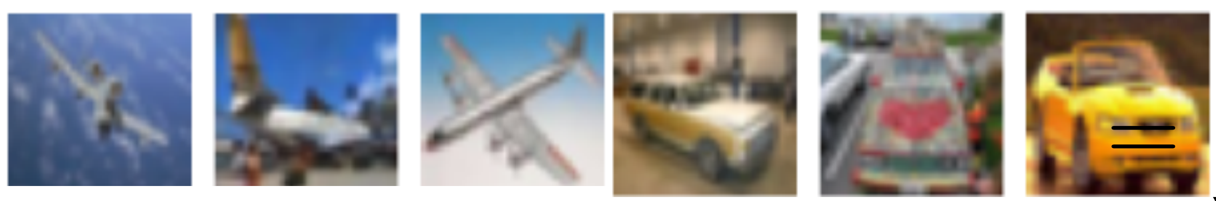
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(

$+1$ $+1$ $+1$ -1 -1 -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 $y_i = -1$

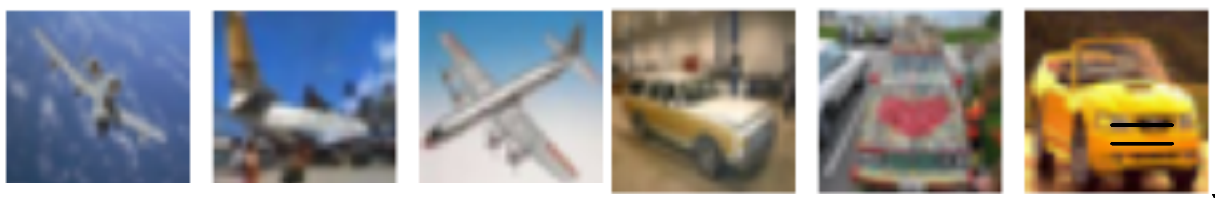
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(

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

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

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

return $\mathbf{w}^*$

2.5

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

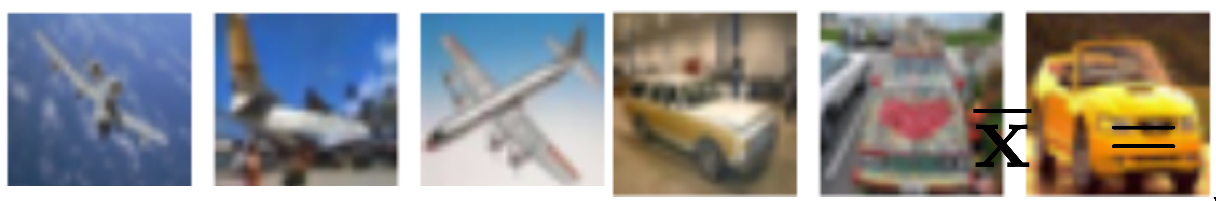
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

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

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

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

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

return $\mathbf{w}^*$

$-(-1) \times 2.5$

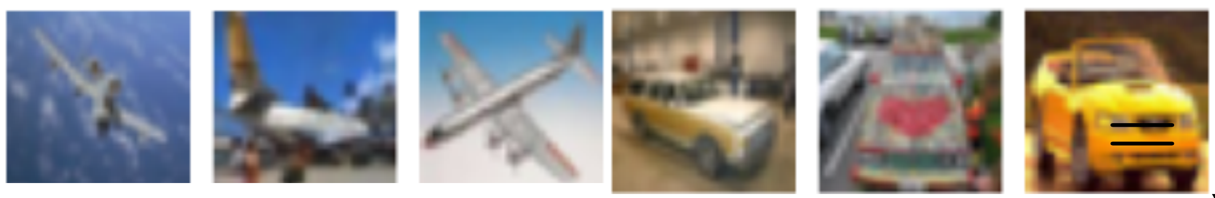
Labels (y_i) RGB images ($\mathbf{x}_i$)

$+1$

-1

Training

Example: Training linear classifier

def train(
$+1$ $+1$ $+1$ -1 -1 -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 $\mathbf{w}^*$

1.12

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

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

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

- There is no closed-form solution
- Gradient optimization

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

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

- There is no closed-form solution
- Gradient optimization

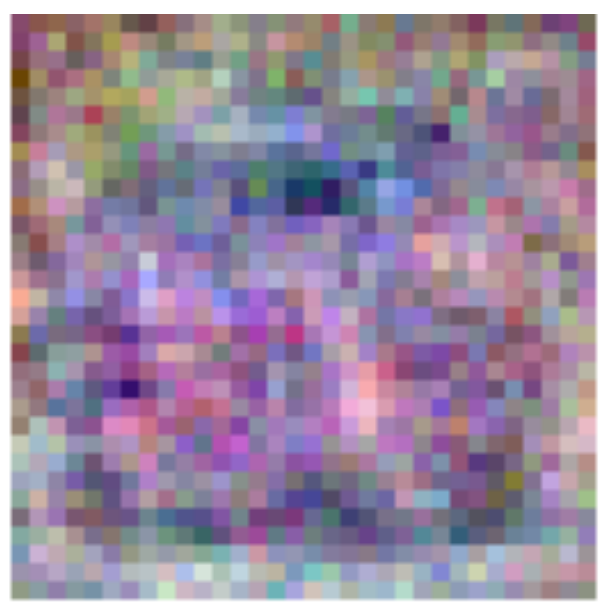
$$\mathbf{w} = \mathbf{w} - \alpha \left[\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}\right]^\top \text{where } \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \frac{-y_i \overline{\mathbf{x}}_i^\top}{1 + \exp(y_i \mathbf{w}^\top \overline{\mathbf{x}}_i)}$$

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

- There is no closed-form solution
- Gradient optimization

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

Learned weights as a template:



automobile

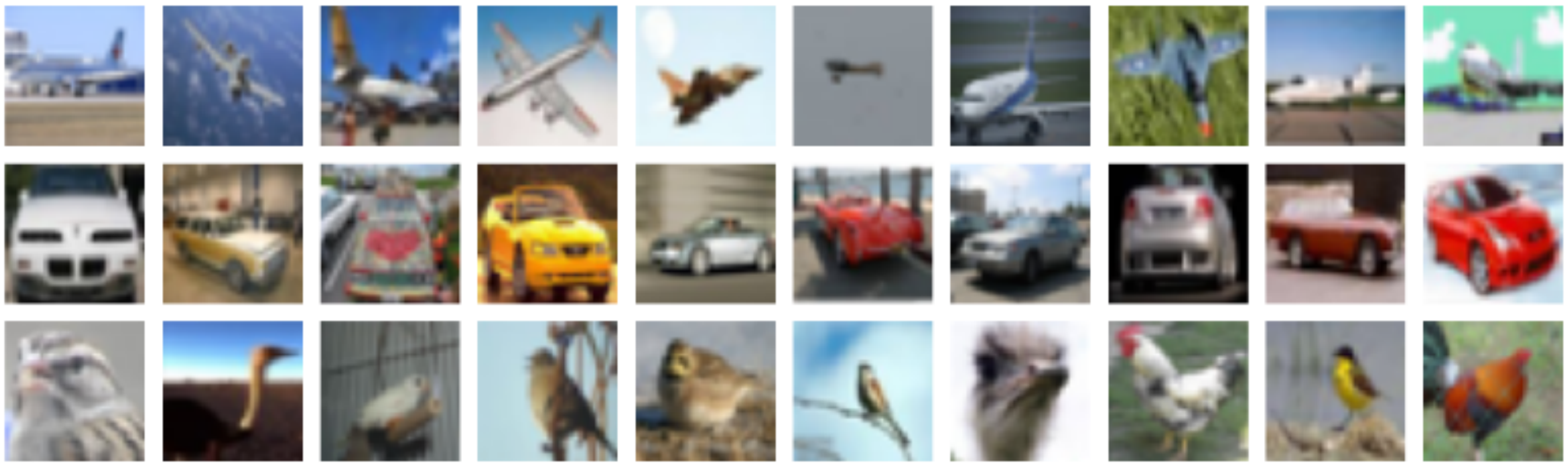
Show python code

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

1

2

3



Three-class recognition problem:

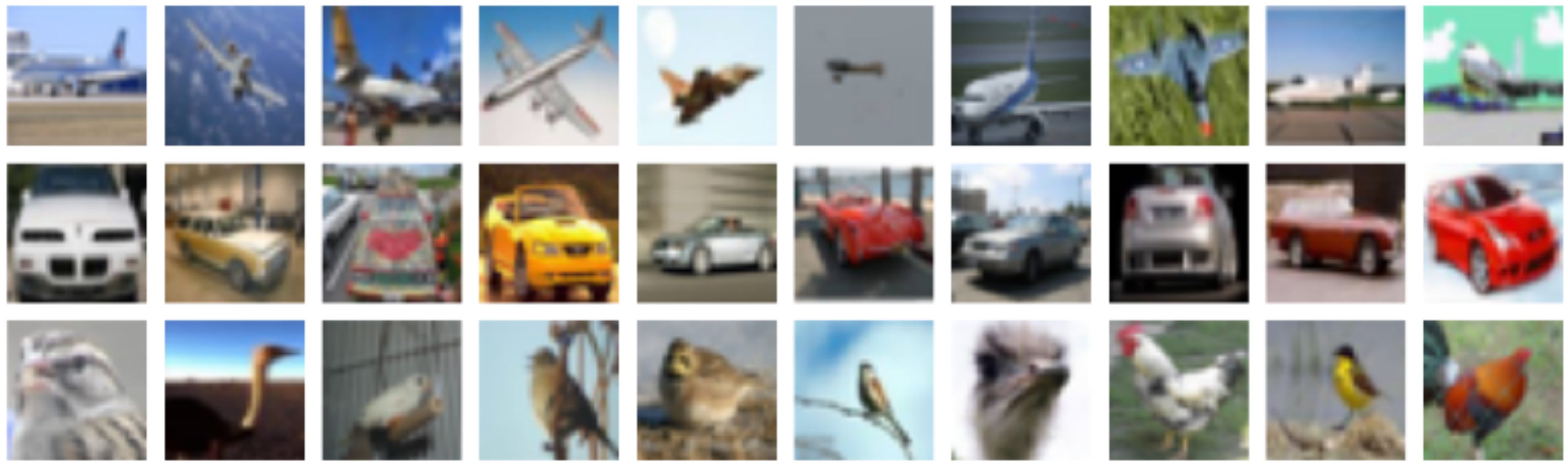
Labels (y_i)

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

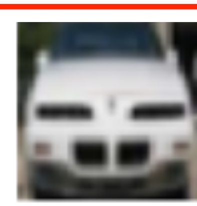
1

2

3



Three-class recognition problem:

def classify():

???

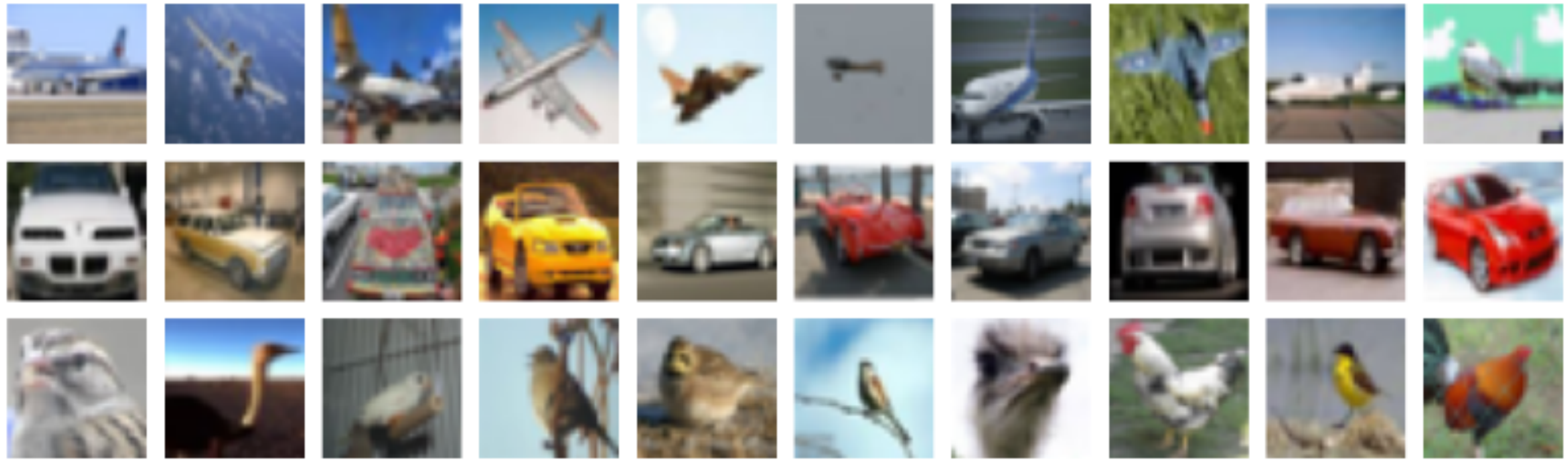
return $\mathbf{p}$

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

1

2

3

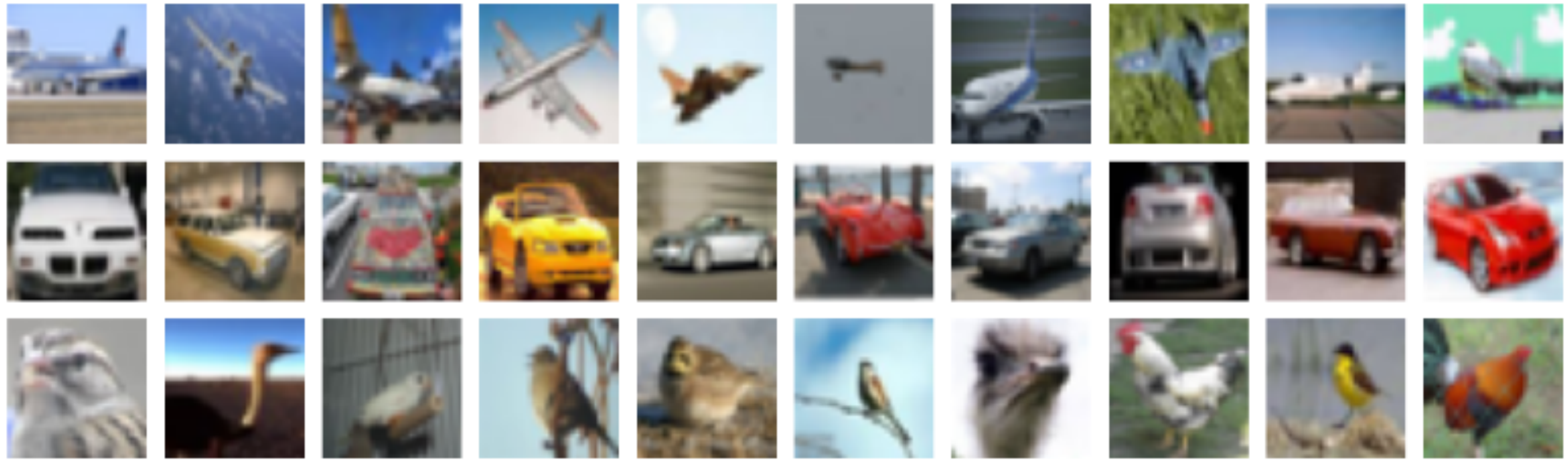


Model probability distribution over classes by softmax function

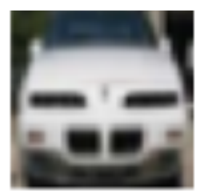
$$p(y|\mathbf{x}, \mathtt{W}) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \\ \exp(f(\mathbf{x}, \mathbf{w}_3)) \end{bmatrix} \Big/ \sum_k \exp(f(\mathbf{x}, \mathbf{w}_k) = \mathbf{s}(\mathbf{f}(\mathbf{x}, \mathtt{W}))$$

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

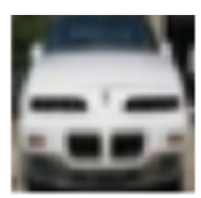
1
2
3



Three-class recognition problem:

def classify():

 # Linear classifier

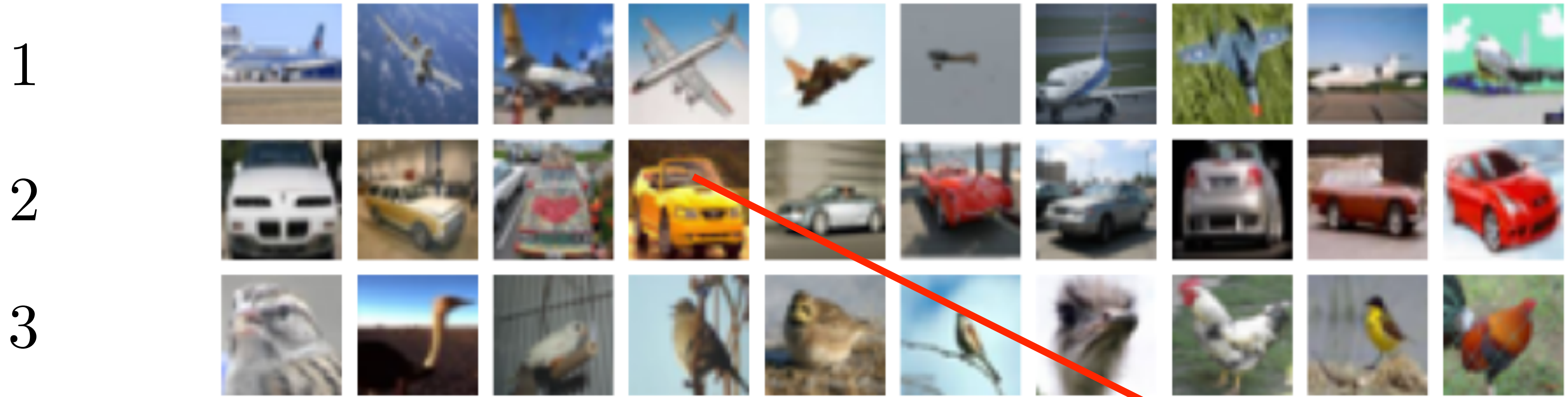
$\mathbf{x} = \text{vec}($  $)$

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

return $\mathbf{p}$

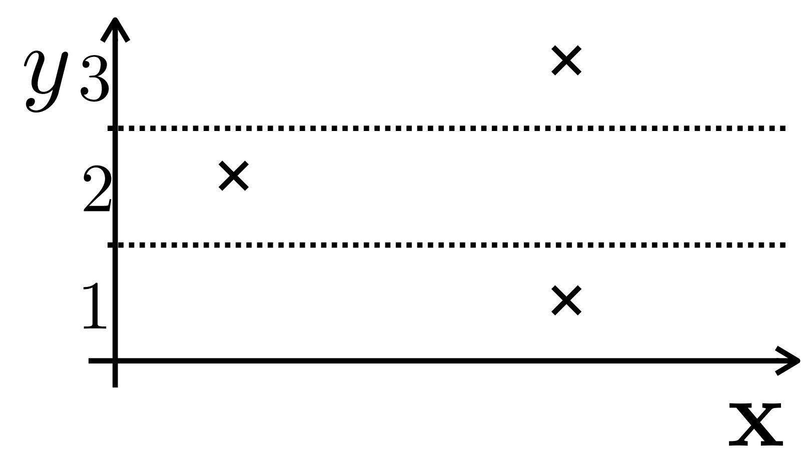
$$\mathtt{W}\;\overline{\mathbf{x}} = \begin{bmatrix} -2 \\ +1 \\ 0 \end{bmatrix}$$

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

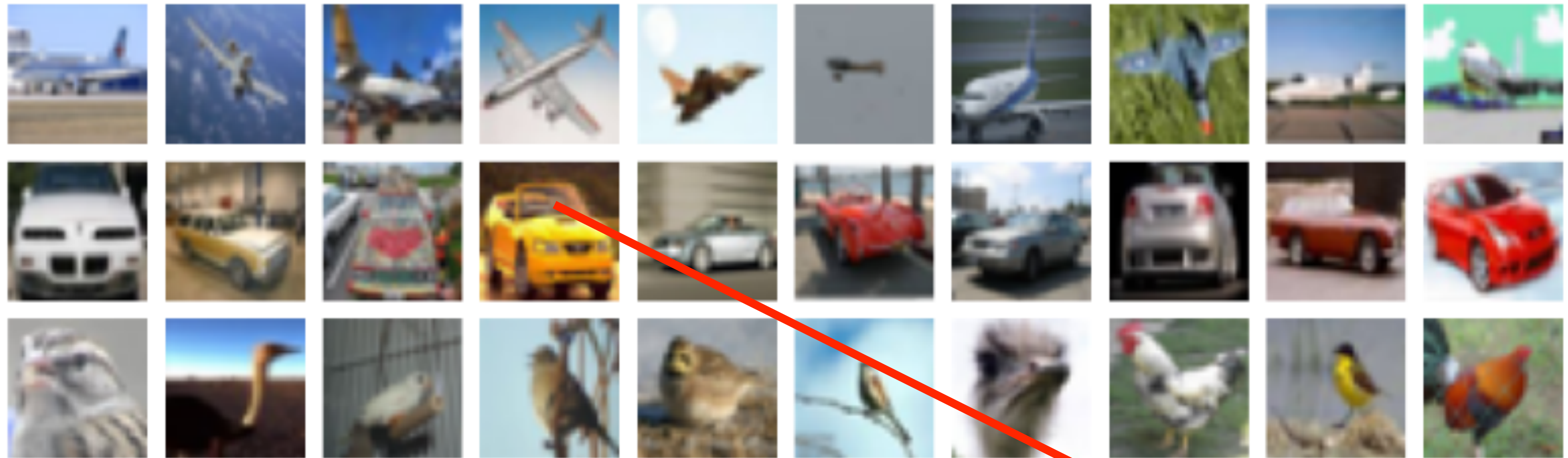


- **Classification** (probability modeled by soft-max function):

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

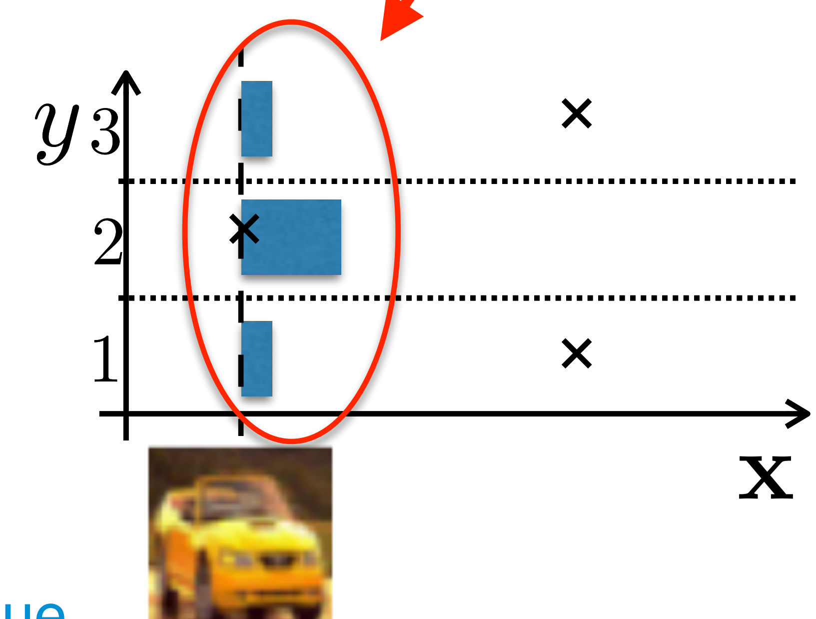


1
2
3

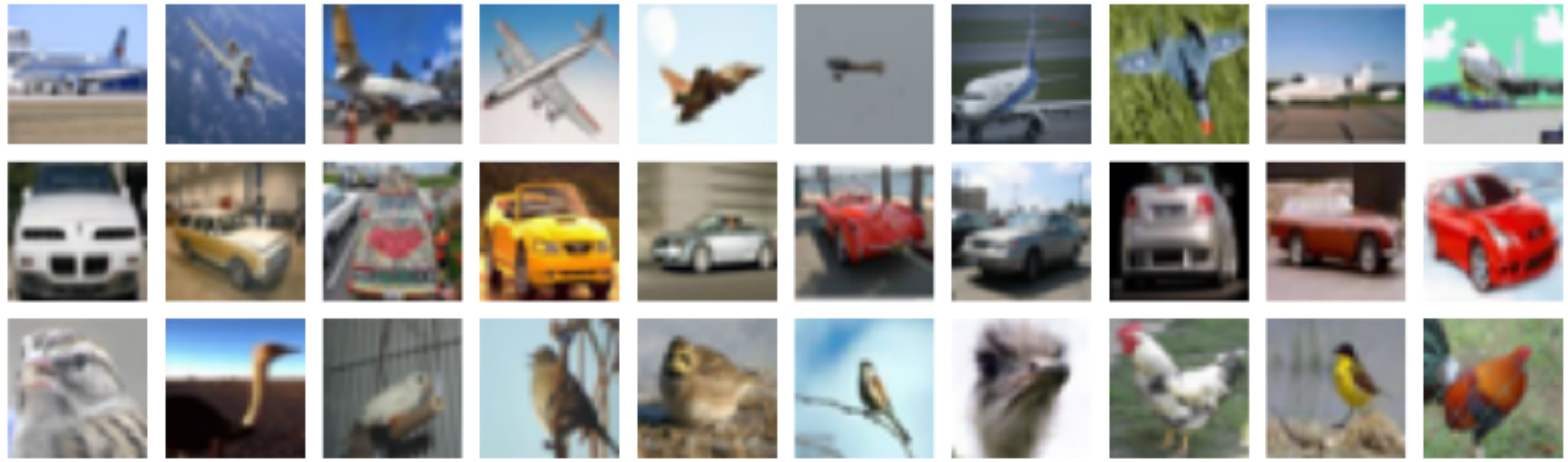


- **Classification** (probability modeled by soft-max function):

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



1
2
3

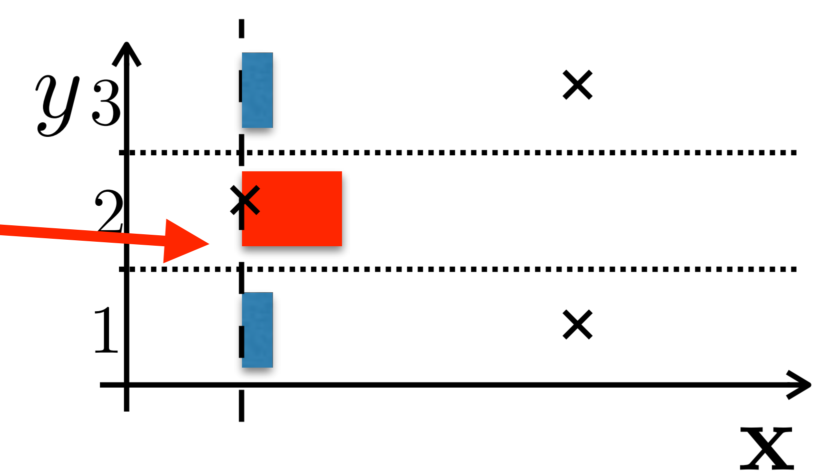


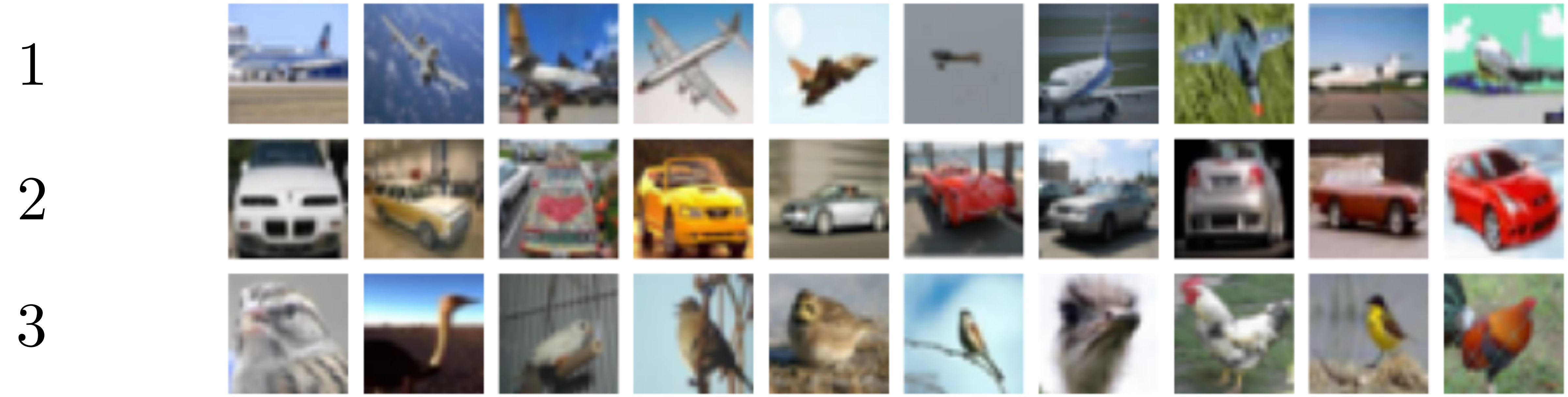
- **Classification** (probability modeled by soft-max function):

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

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

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



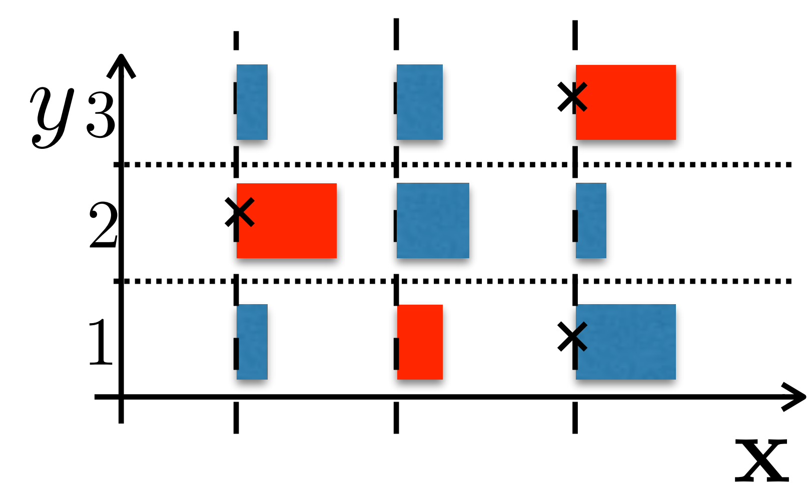


- **Classification** (probability modeled by soft-max function):

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

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

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



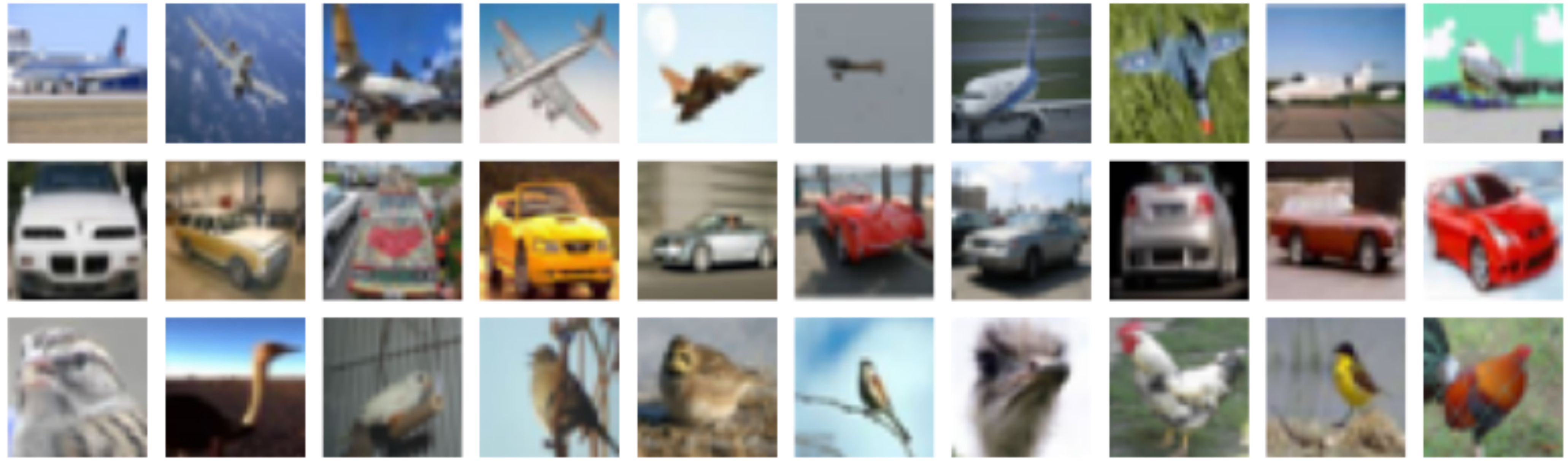
- **Training:** MLE estimate of W

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

1

2

3

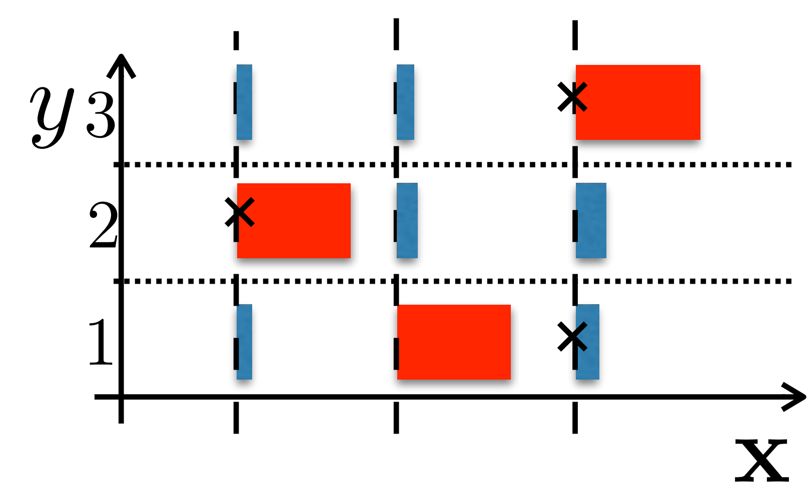


- **Classification** (probability modeled by soft-max function):

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

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

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



- **Training:** MLE estimate of W

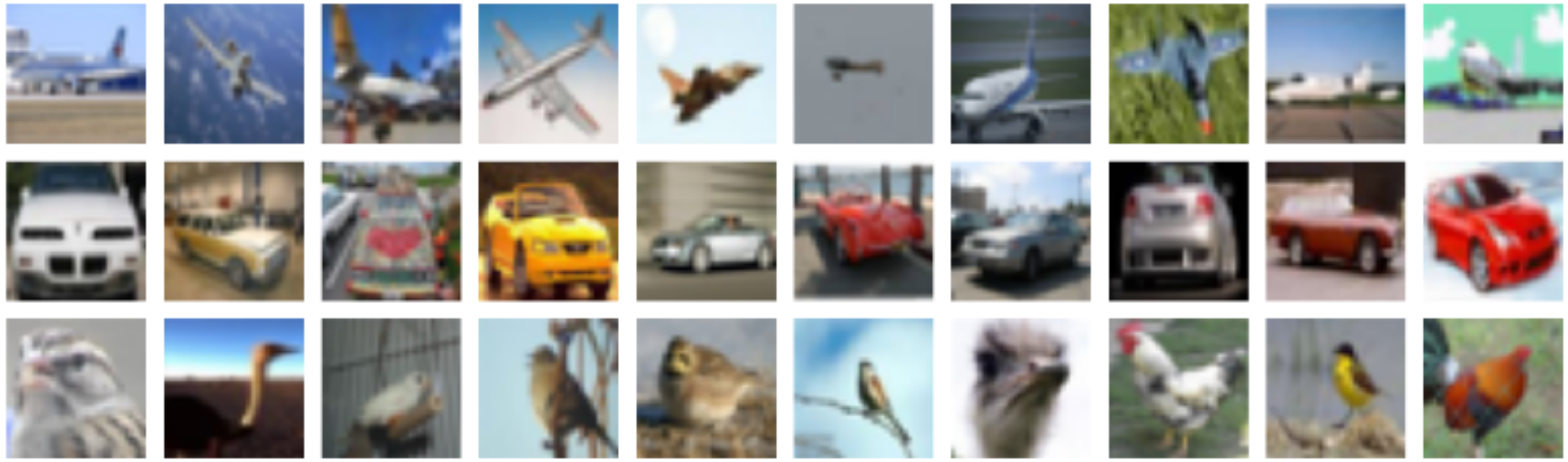
$$\mathtt{W}^* = \arg\min_{\mathtt{W}} \sum_i - \log \mathbf{s}_{y_i}(\mathbf{f}(\mathbf{x}_i, \mathtt{W}))$$

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

1

2

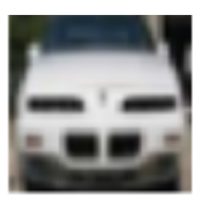
3



def train(

1 1 1 2 2 2 3 3 3):

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



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

return $\mathtt{W}^*$

$$y_i = 2$$

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

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

Car classified as car yields small loss

def train():

1 1 1 2 2 2 3 3 3

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

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

return $\mathtt{W}^*$

$y_i = 1$

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

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

Plane classified as car yields huge loss

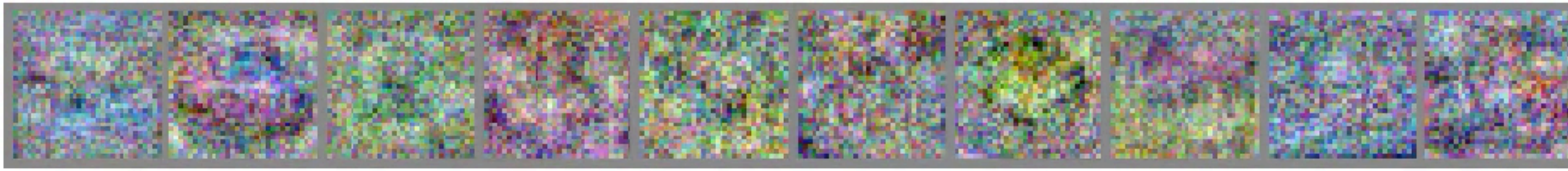
def train(

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 $\mathtt{W}^*$



def train(

1 1 1 2 2 2 3 3 3):

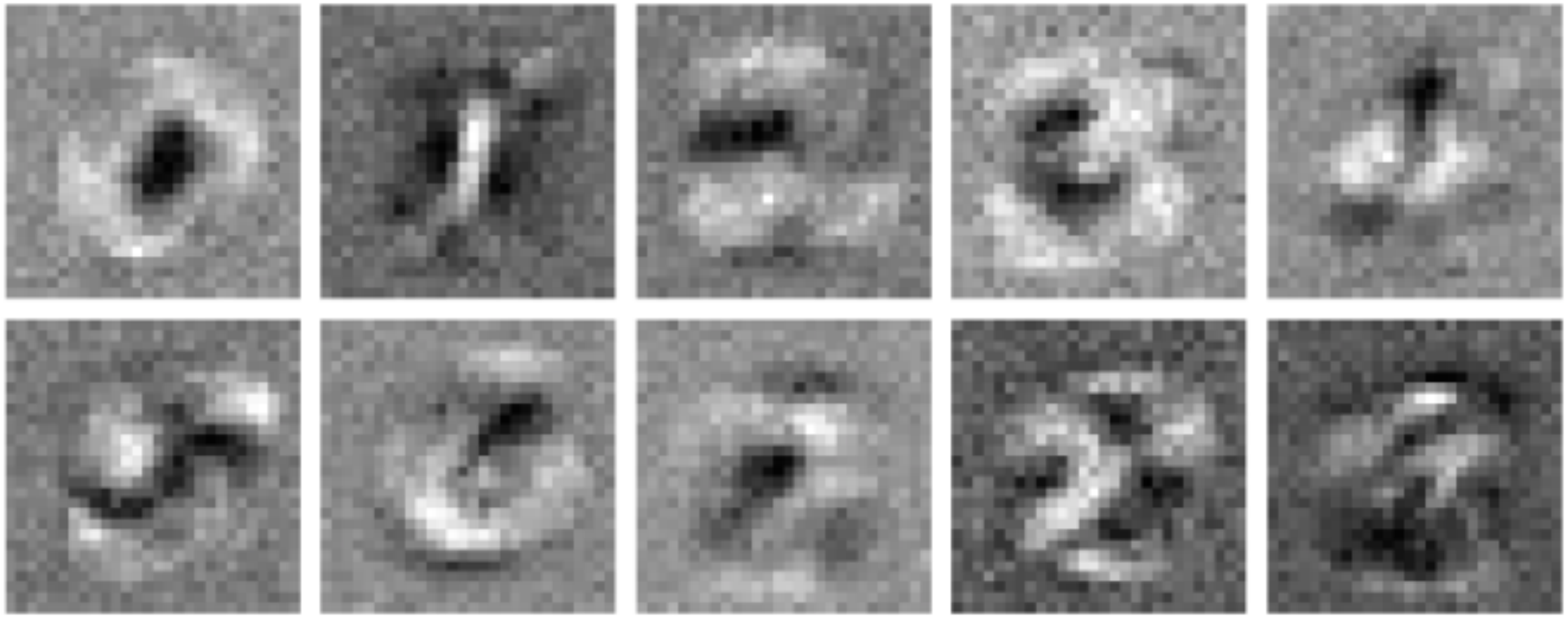
$\mathbf{x}_i = \mathrm{vec}($  $)$

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

return $\mathtt{W}^*$

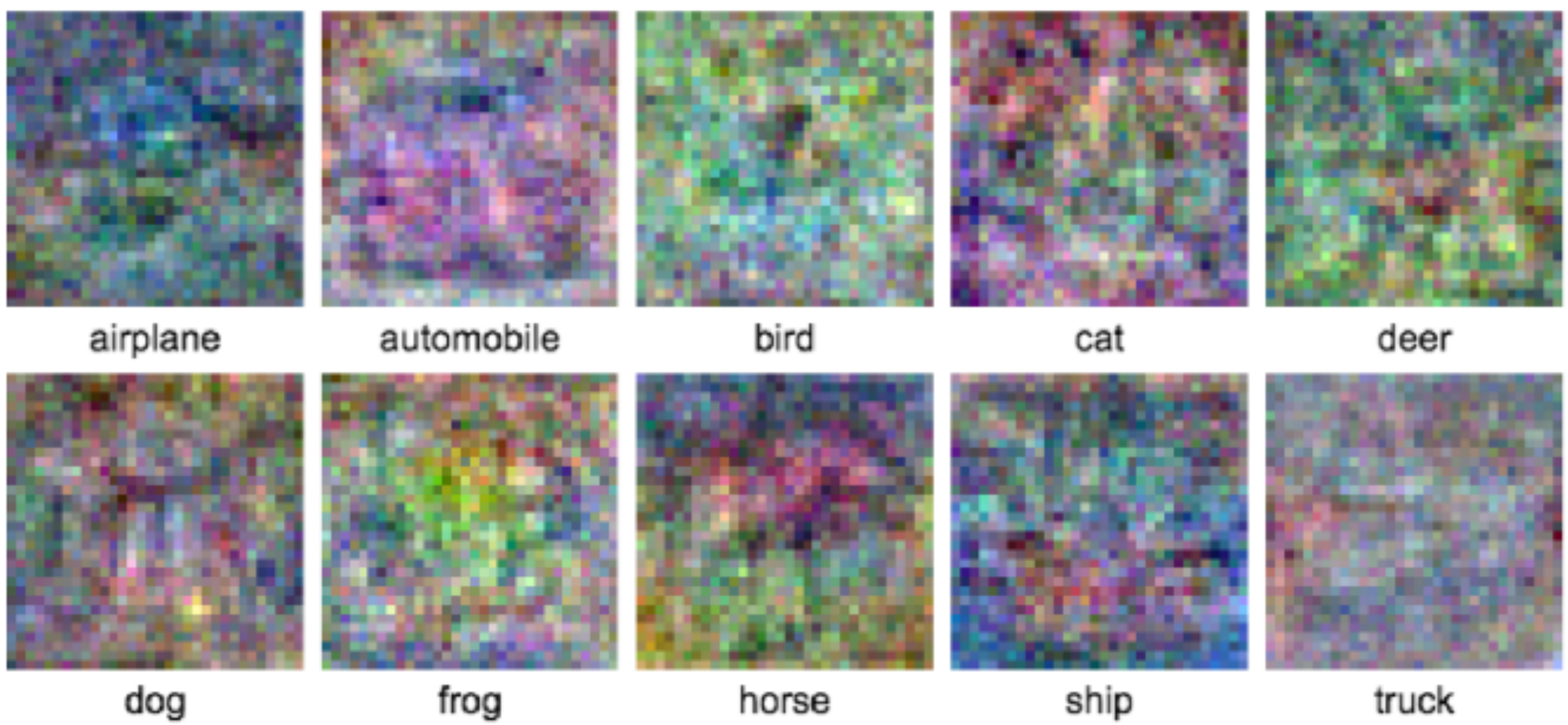
Dataset Learned weights of linear classifier Accuracy

MNIST



91%

CIFAR-10



37%

Demo: https://ml4a.github.io/ml4a/looking_inside_neural_nets/

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

loss function prior/regulariser

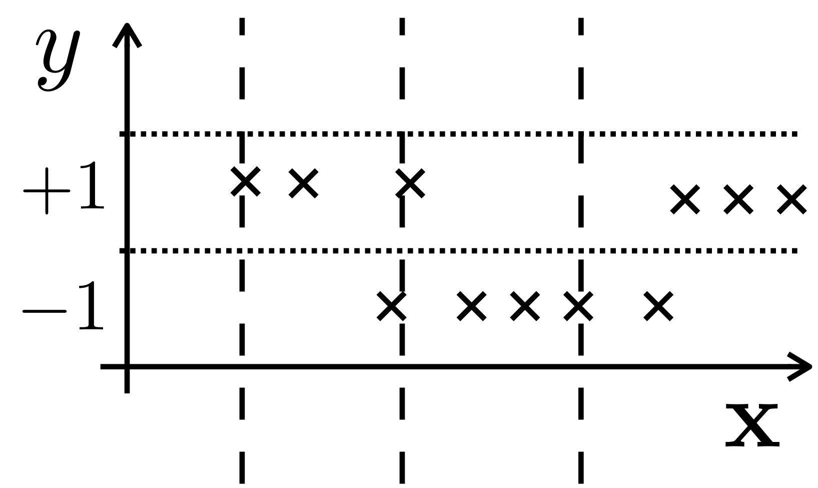
- **Classification**: $p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & y = +1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & y = -1 \end{cases}$
- Choice of $f(\mathbf{x}, \mathbf{w})$ is crucial

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

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:

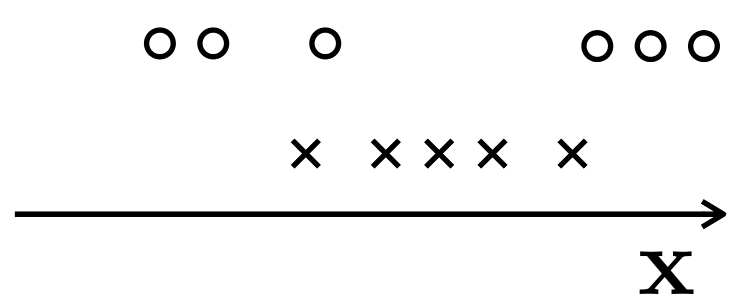


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

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

1D example:

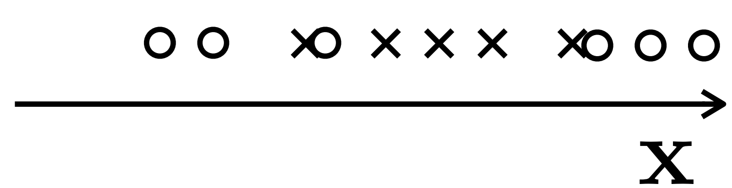


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

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

1D example:

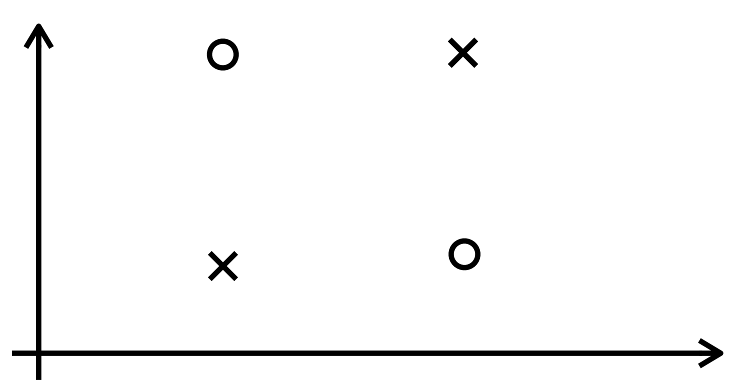


$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i - \log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

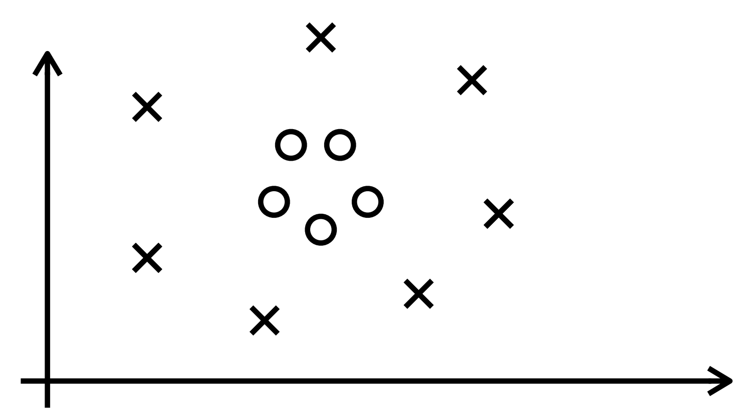
loss function prior/regulariser

- **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

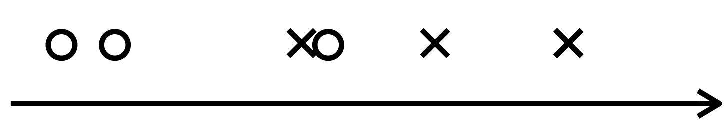


circle

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

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

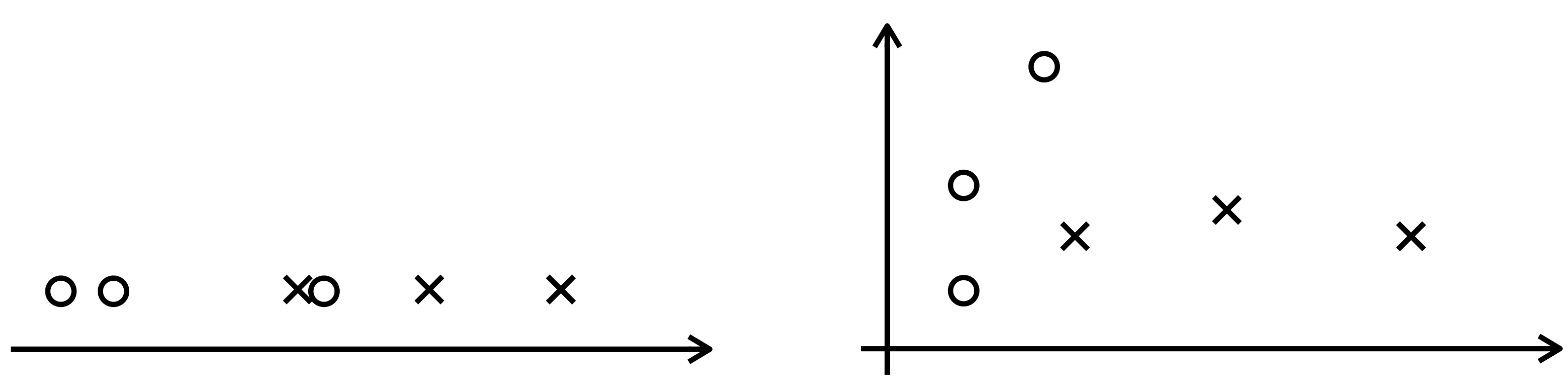


1D case

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

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

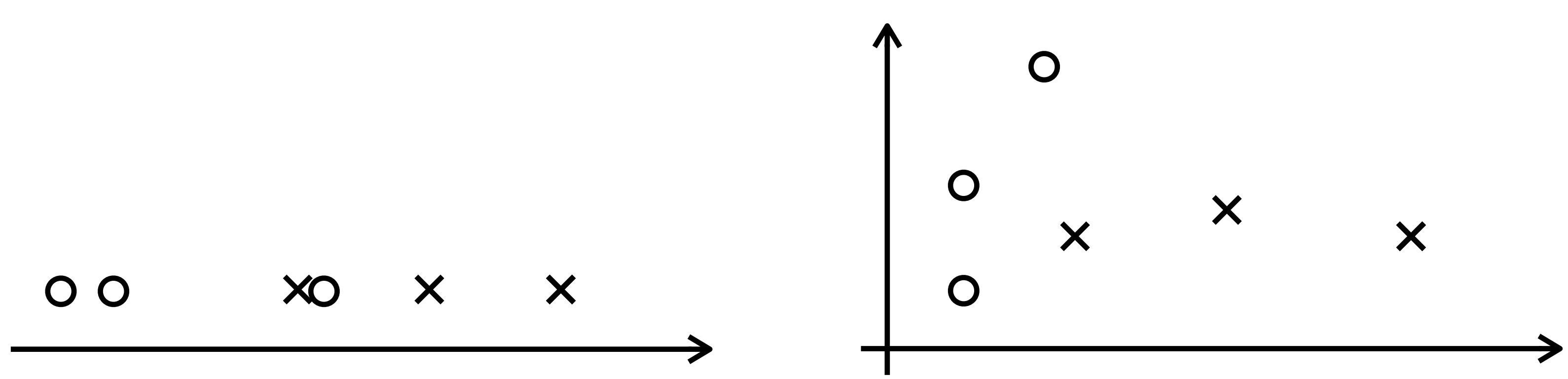


1D case 2D case

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

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

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

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

Conclusions

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

Competencies required for the test T1

- Derive MAP/ML estimate for two-class and K-class classification problem.
- Compute logistic-loss and cross-entropy-loss
- Understand when classifier has high/low values.
- Understand when loss has high/low values.