Deep Learning Essentials

2. Linear Classifier

Under the hood of a linear classifier of RGB images

Lukáš Neumann

Adapted from B3B33UROB slides of Karel Zimmerman

Image Classification

- CIFAR-10: Classify 32x32 pixel RGB images into 10 classes

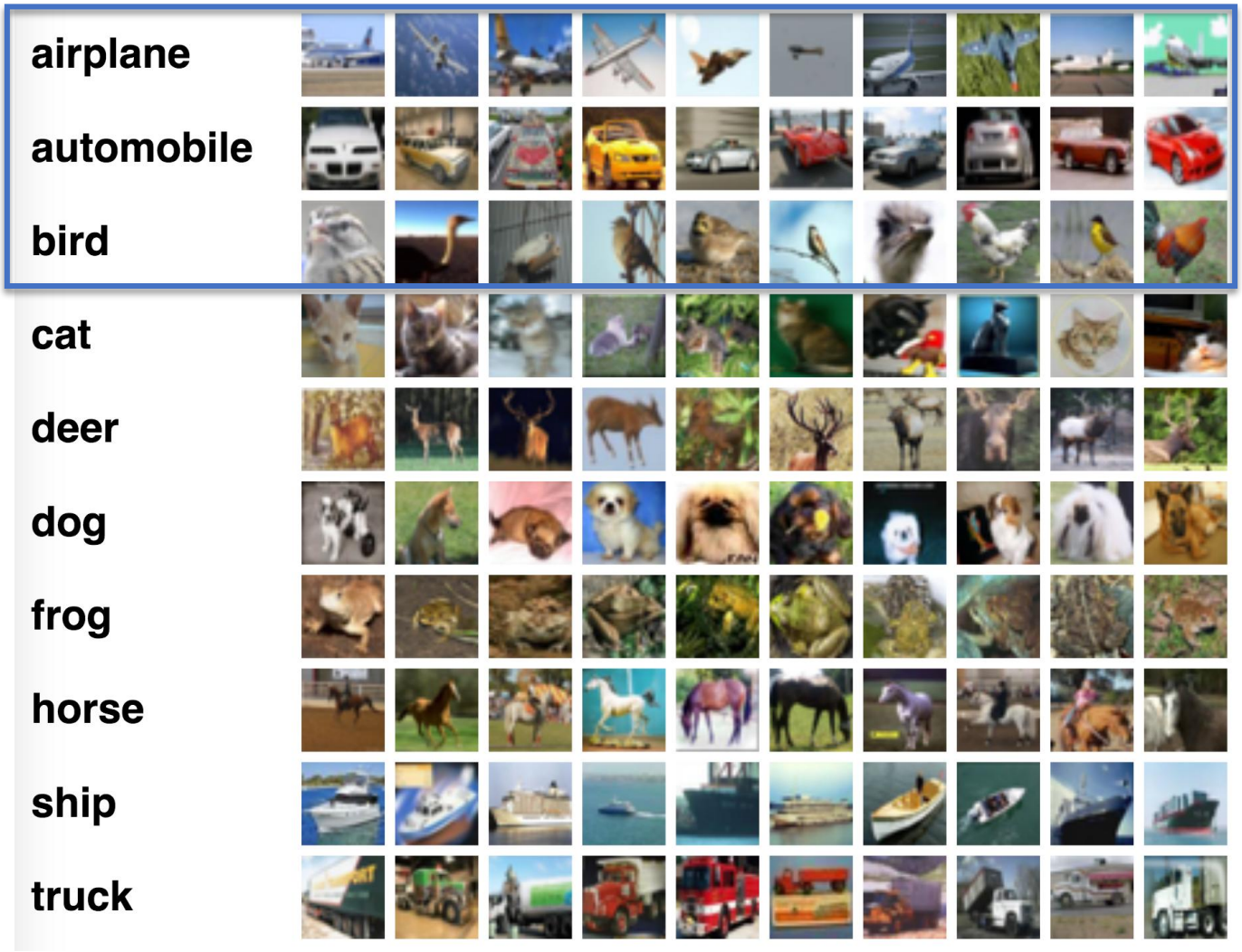
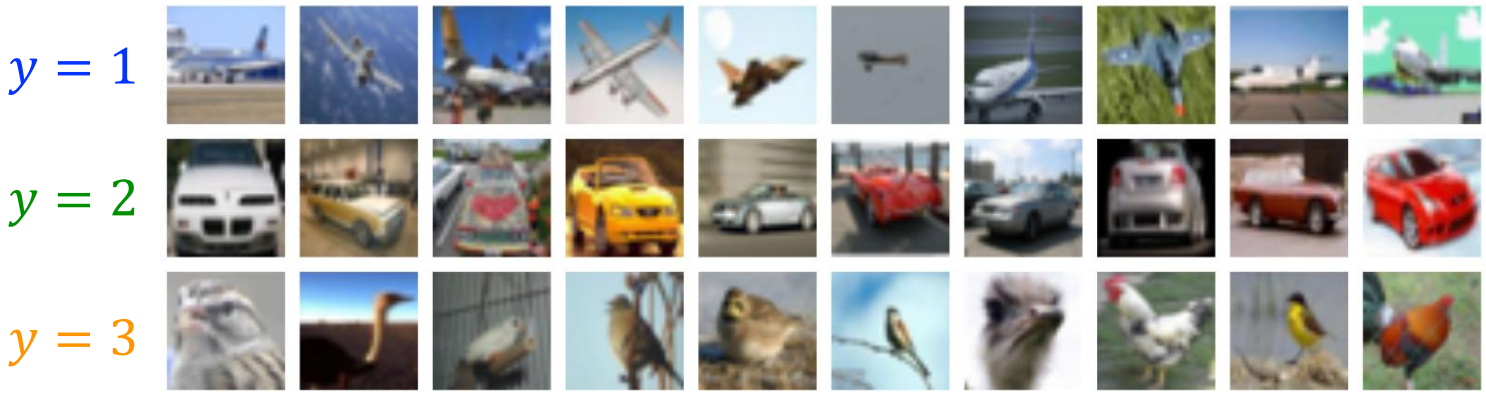
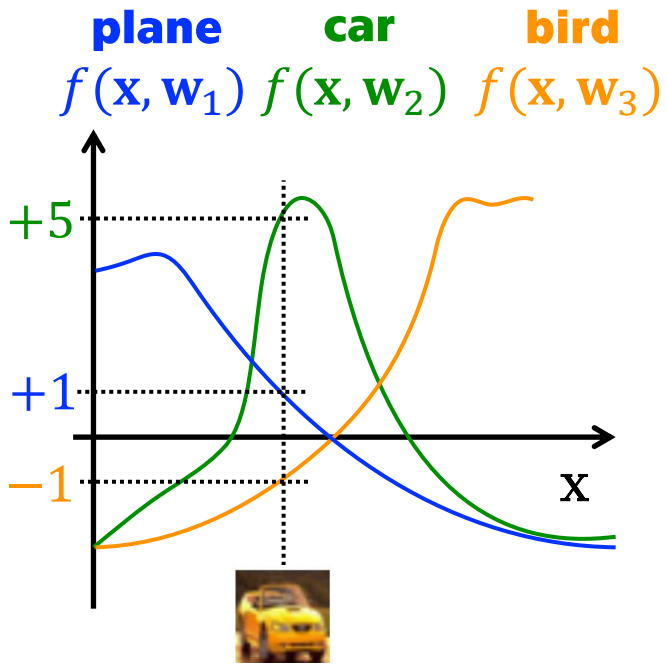


Image Classification



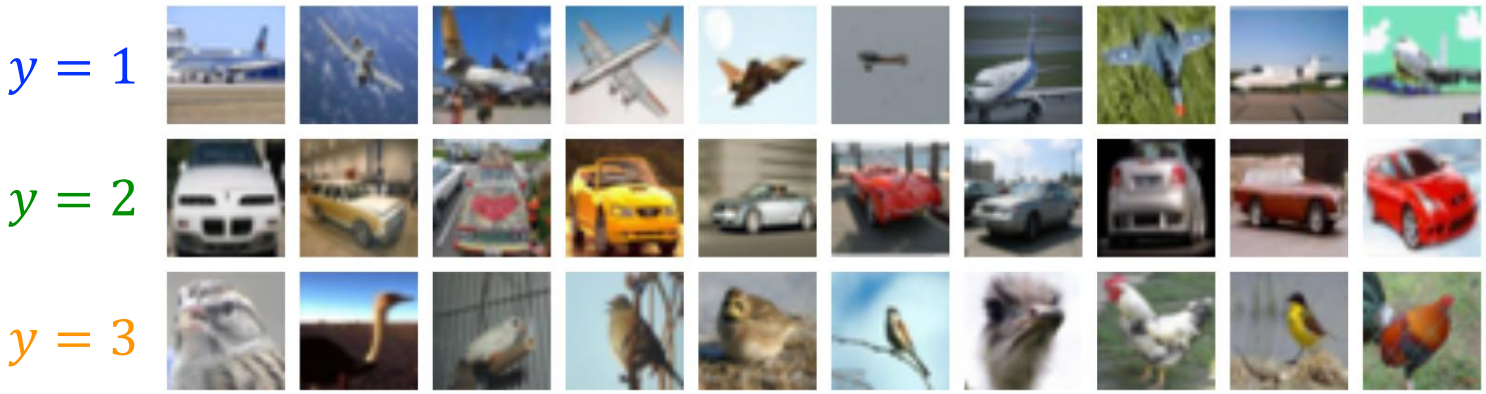


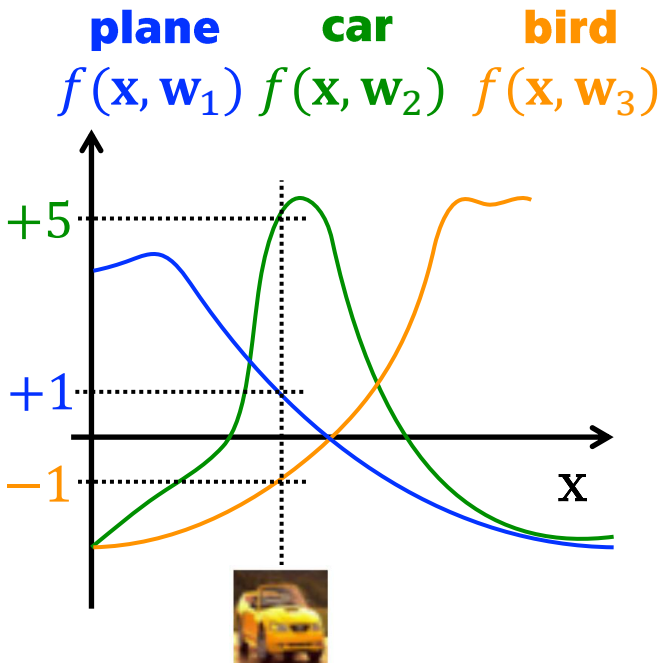
logits

$$f(\mathbf{x}, \mathbf{W}) = \boldsymbol{z} = \begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix}$$

Logits are the raw (unnormalized) outputs of the classifier. The higher the score, the more the classifier „thinks“ the observation belongs to given class.

Linear Classifier





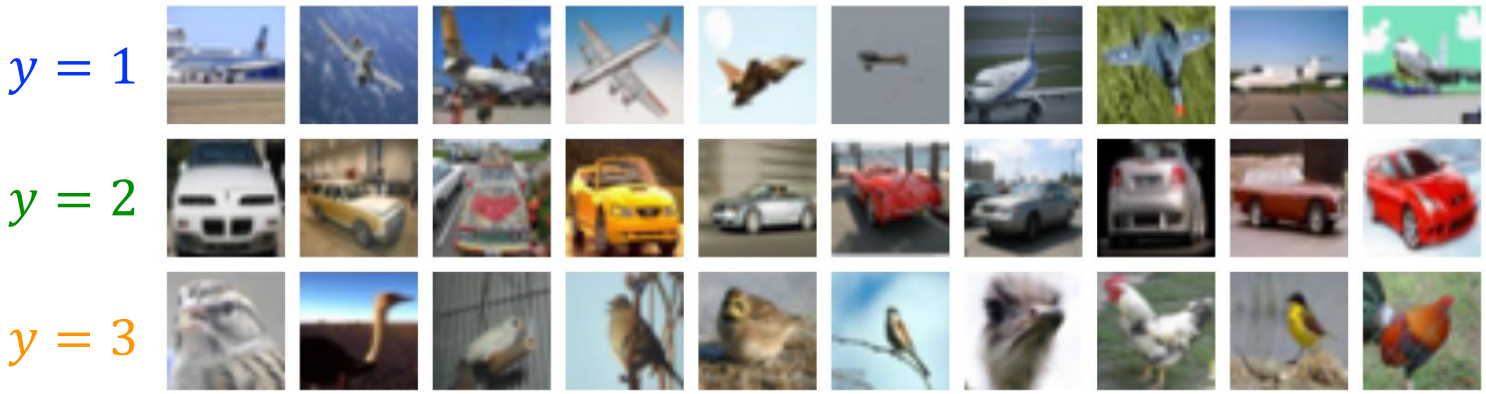
$$f(\bar{\mathbf{x}}, \mathbf{W}) = \boxed{\mathbf{W}} \boxed{\bar{\mathbf{x}}} = \begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix}$$

vectorization

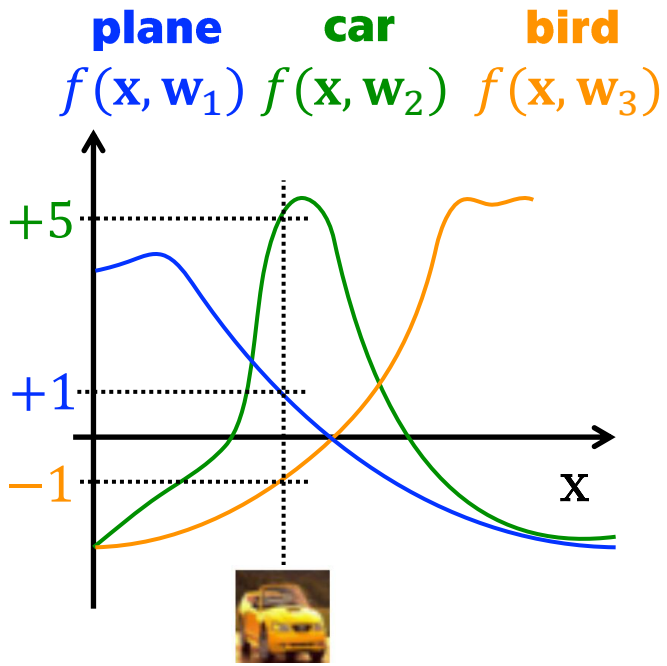
$$\bar{\mathbf{x}} = [\text{vec}(\text{  }) | 1]$$

$$\text{vec}: \mathbb{N}^{32\text{x}32\text{x}3} \rightarrow \mathbb{N}^{3072}$$

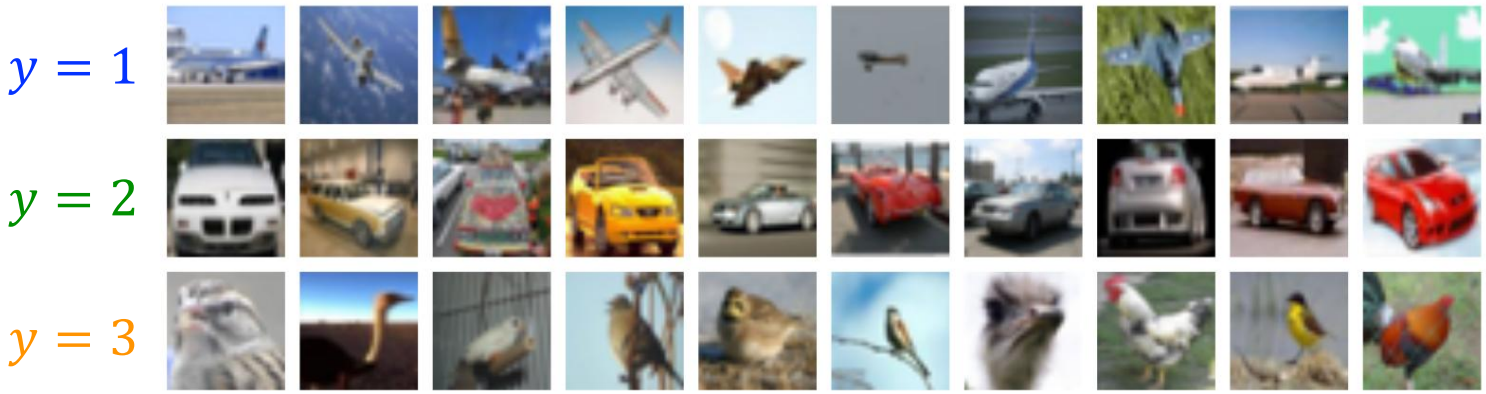
Linear Classifier



$$f(\bar{\mathbf{x}}, \mathbf{W}) = \boldsymbol{z} = \begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix} \quad \mathrm{p}(\mathrm{y} \mid \bar{\mathbf{x}}, \mathbf{W}) = ?$$

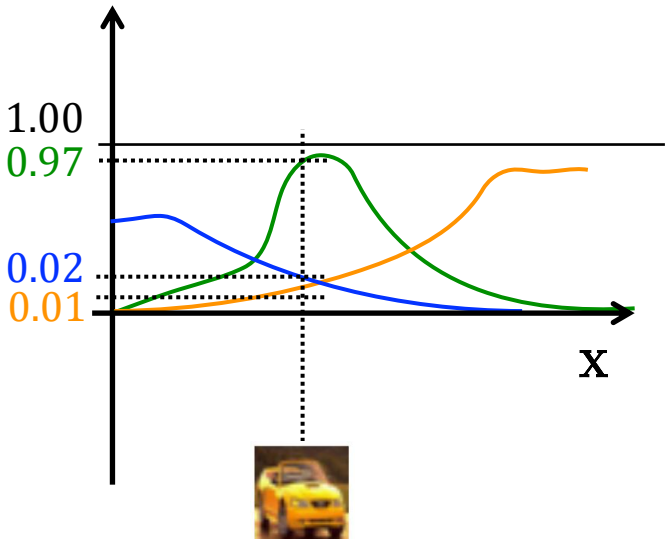


Linear Classifier



$$f(\bar{\mathbf{x}}, \mathbf{W}) = \boldsymbol{z} = \begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix} \quad \mathrm{p}(\mathrm{y} \mid \bar{\mathbf{x}}, \mathbf{W}) = ?$$

plane $\mathrm{p}(\mathrm{y} = 1 | \mathbf{x}, \mathbf{w}_1)$ **car** $\mathrm{p}(\mathrm{y} = 2 | \mathbf{x}, \mathbf{w}_2)$ **bird** $\mathrm{p}(\mathrm{y} = 3 | \mathbf{x}, \mathbf{w}_3)$

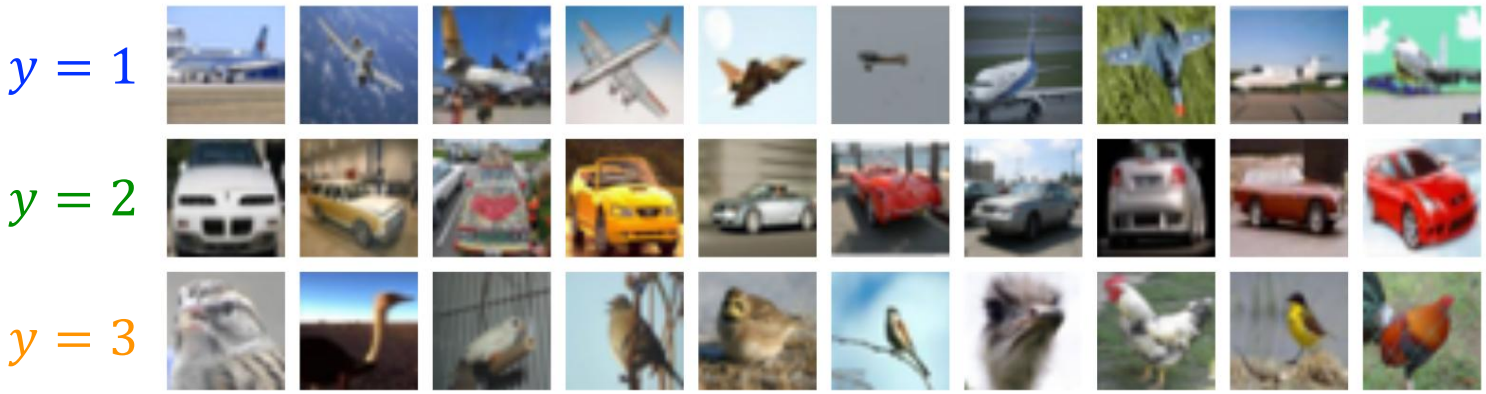


$$\mathrm{p}(\mathrm{y} \mid \bar{\mathbf{x}}, \mathbf{W}) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \\ \exp(f(\mathbf{x}, \mathbf{w}_3)) \end{bmatrix} / \sum_{\mathrm{k}} \exp(f(\mathbf{x}, \mathbf{w}_{\mathrm{k}}))$$

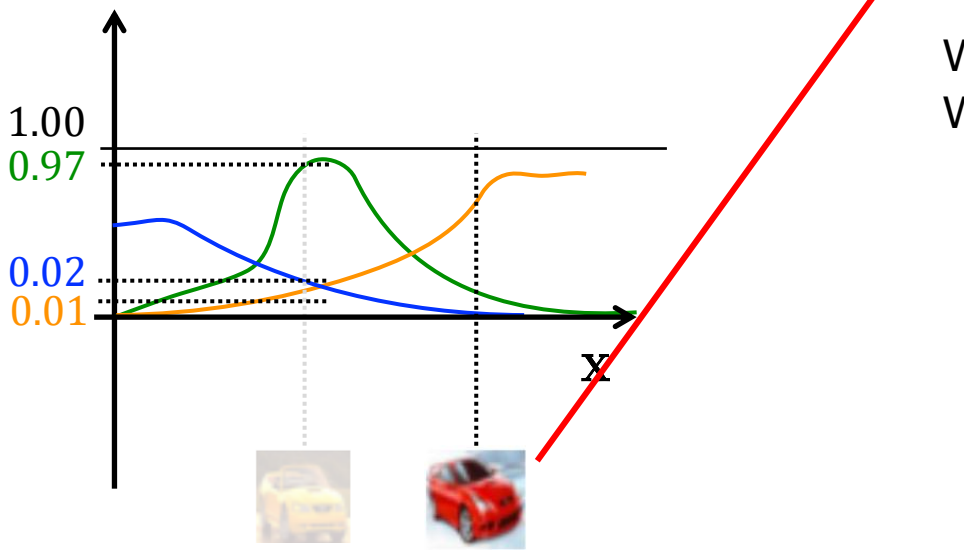
$$= \mathrm{softmax}(\boldsymbol{z}) = \frac{\mathrm{e}^{z_{\mathrm{j}}}}{\sum_{\mathrm{k}} \mathrm{e}^{z_{\mathrm{k}}}}$$

$$= \mathrm{softmax}\left(\begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.02 \\ 0.97 \\ 0.01 \end{bmatrix}$$

Linear Classifier



plane $p(y = 1|\mathbf{x}, \mathbf{w}_1)$ **car** $p(y = 2|\mathbf{x}, \mathbf{w}_2)$ **bird** $p(y = 3|\mathbf{x}, \mathbf{w}_3)$



$$\text{softmax}(f(\bar{\mathbf{x}}_i, \mathbf{W})) = \begin{bmatrix} 0.01 \\ 0.31 \\ 0.68 \end{bmatrix}$$

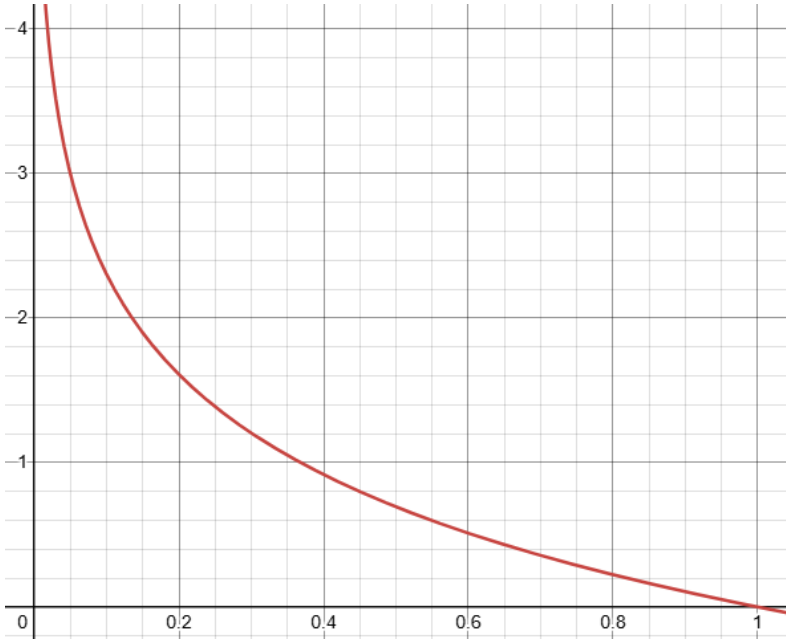
We want to train the classifier →
We need to optimize some loss

$$\mathcal{L}_i = ?$$

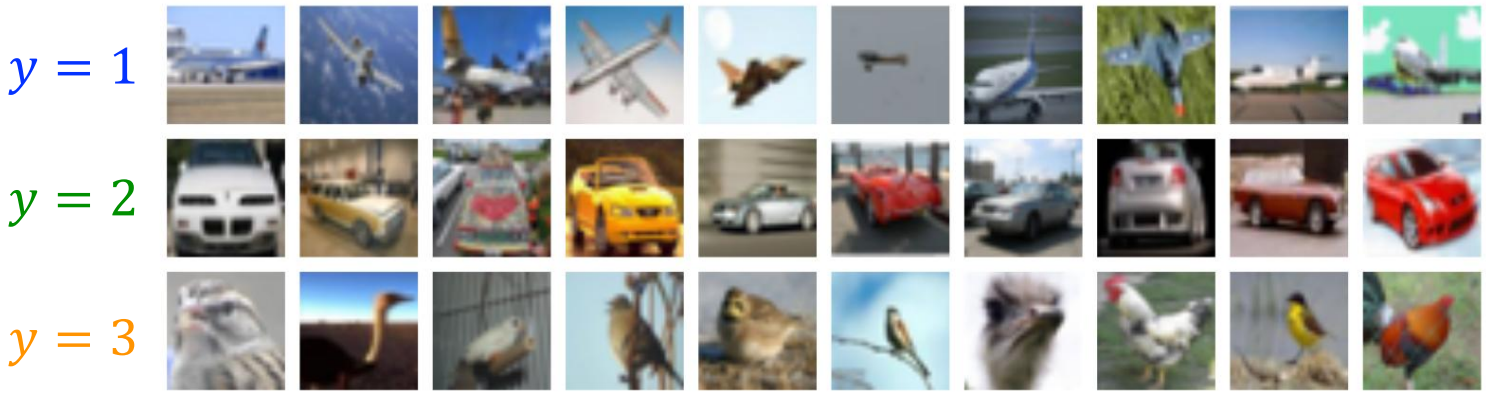
Loss Function

- What are the desired properties of good loss function for classification?
- It should push probability of the “true” class up, while push probabilities of incorrect classes down
  - **Equal formulation:** Penalty for the “true” class should be low, while penalty for incorrect classes should be high

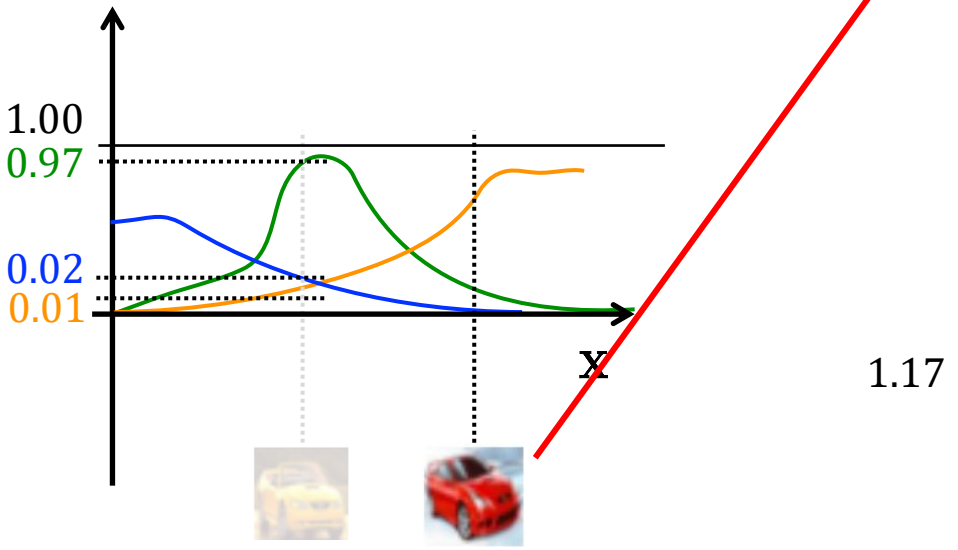
$$\mathcal{L}_i = -\log\left(p(y_i \mid x_i, \mathbf{W})\right)$$



Loss Function

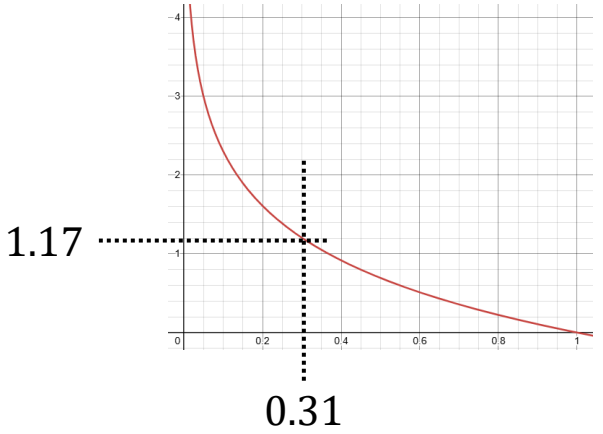


plane $p(y = 1|\mathbf{x}, \mathbf{w}_1)$ **car** $p(y = 2|\mathbf{x}, \mathbf{w}_2)$ **bird** $p(y = 3|\mathbf{x}, \mathbf{w}_3)$



$$\text{softmax}(f(\bar{\mathbf{x}}_i, \mathbf{W})) = \begin{bmatrix} 0.01 \\ 0.31 \\ 0.68 \end{bmatrix}$$

$$\mathcal{L}_i = -\log(0.31) = 1.17$$



Loss Function

- We have loss function value for a single data point. Now let's put it together!
- Assuming we have N training samples, the loss function is the

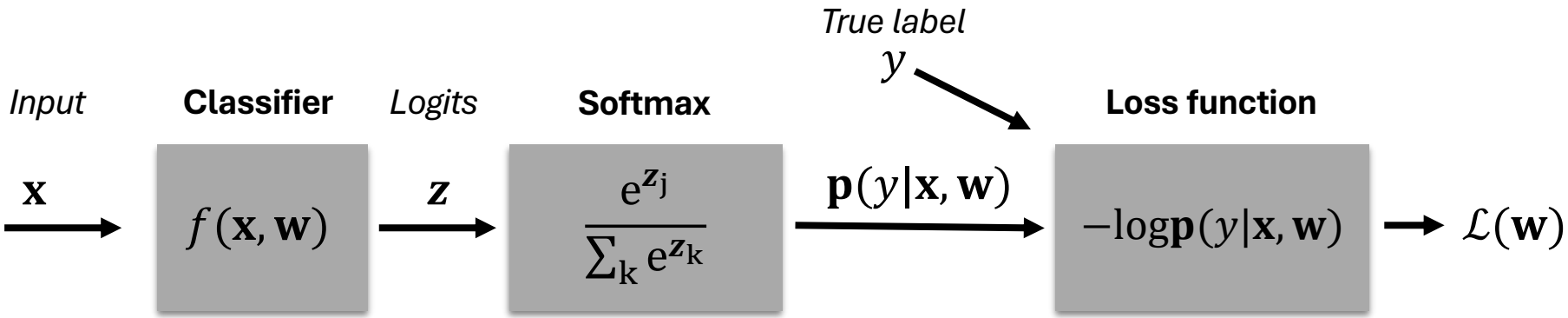
Cross-Entropy (CE)

$$\mathcal{L}(\mathbf{W}) = \mathrm{CE}(\mathbf{W}) = -\sum_{i=1}^{N} \log\left(p(y_i \mid x_i, \mathbf{W})\right)$$

- Learning = loss minimization = finding optimal weights $\mathbf{W}^*$

$$\mathbf{W}^* = \operatorname{argmin} \mathcal{L}(\mathbf{W})$$

Training pipeline

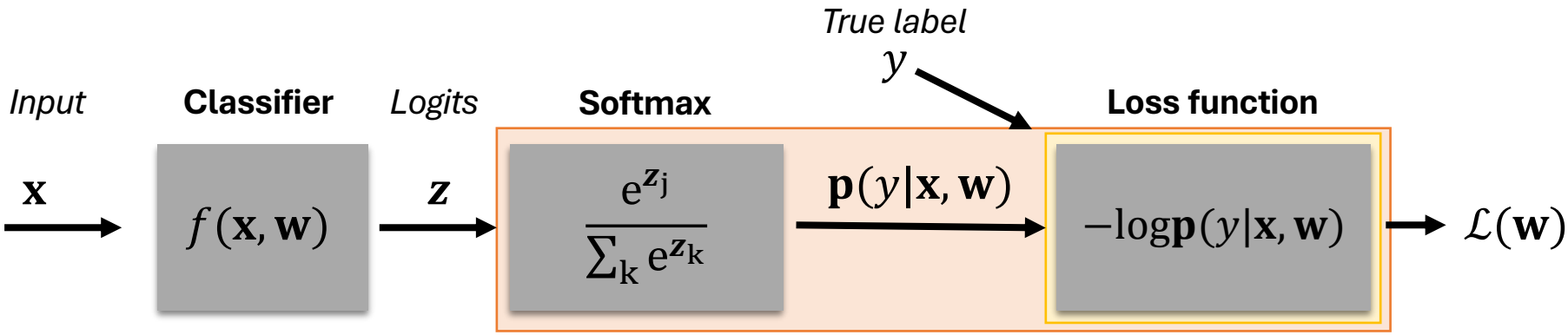


1. Calculate loss for each sample $\mathbf{x}$ and then sum (average) over a training set batch (**Loss function**)
2. Calculate weight gradient with respect to the loss $\frac{\partial \mathcal{L}}{\partial \mathrm{w}}$ (**Autograd** → Lecture 4)
3. Update model weights $\mathbf{w}^{t+1} := \mathbf{w}^{t} - \lambda \frac{\partial \mathcal{L}}{\partial \mathrm{w}}$ (**Optimizer**)

```
for _, (inputs, labels) in enumerate(training_data):
      optimizer.zero_grad()                  # Zero gradients
      logits = model(inputs)                 # Make predictions

      loss = loss_fn(logits, labels)         # 1. Compute the loss
      loss.backward()                        # 2. Calculate gradients
      optimizer.step()                       # 3. Update weights
```

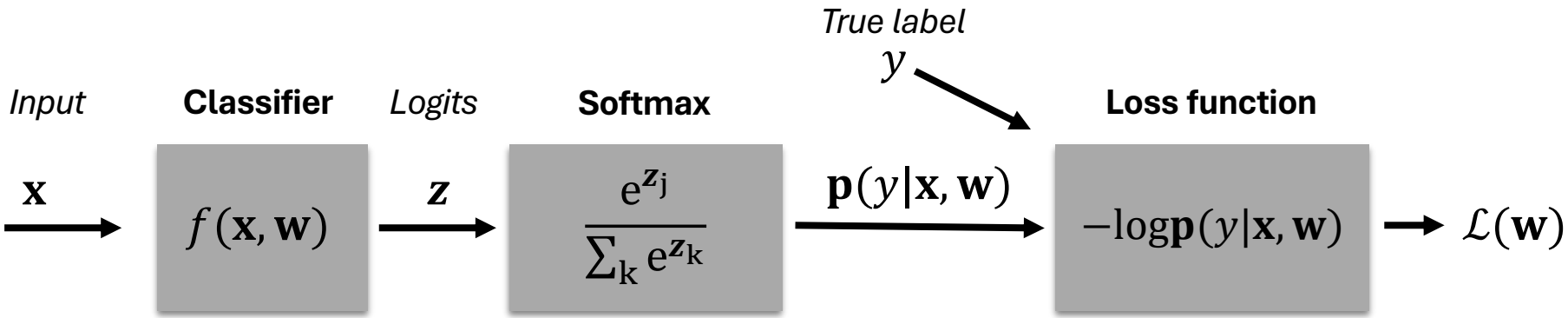
PyTorch implementation

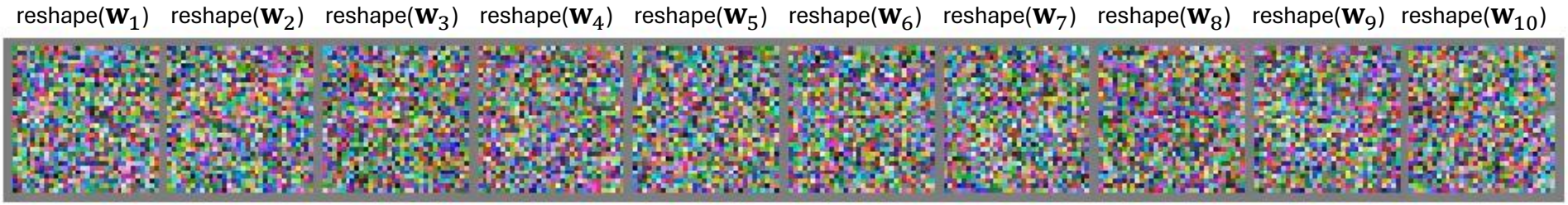


`torch.nn.NLLLoss`
Negative log-likelihood loss, takes (log-)**probabilities** as input.

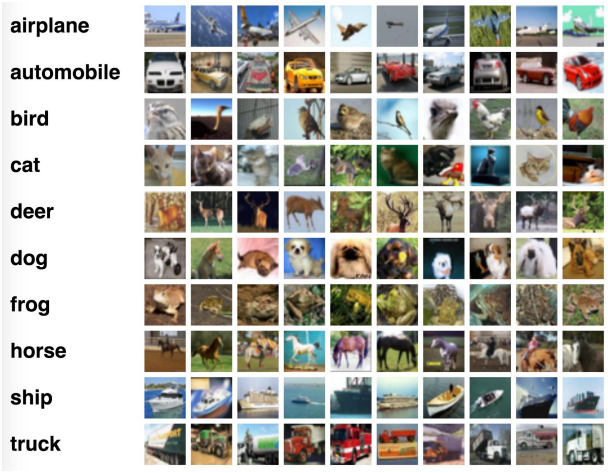
`torch.nn.CrossEntropyLoss`
Combines softmax + negative log-likelihood loss into a single block, takes unnormalized logits as inputs.

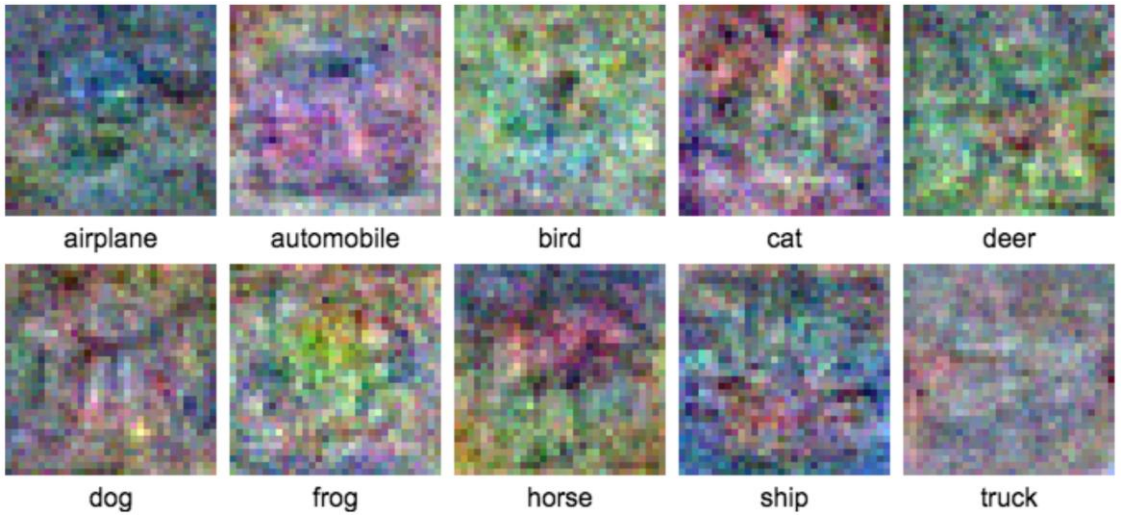
Training pipeline

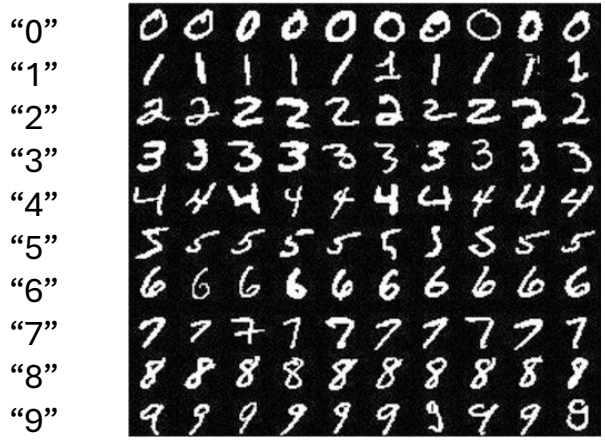


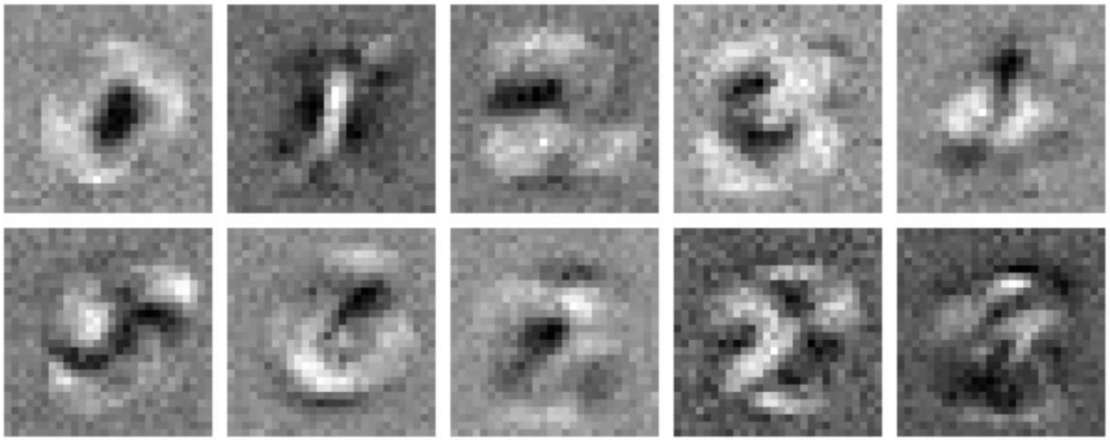


Linear Classifier

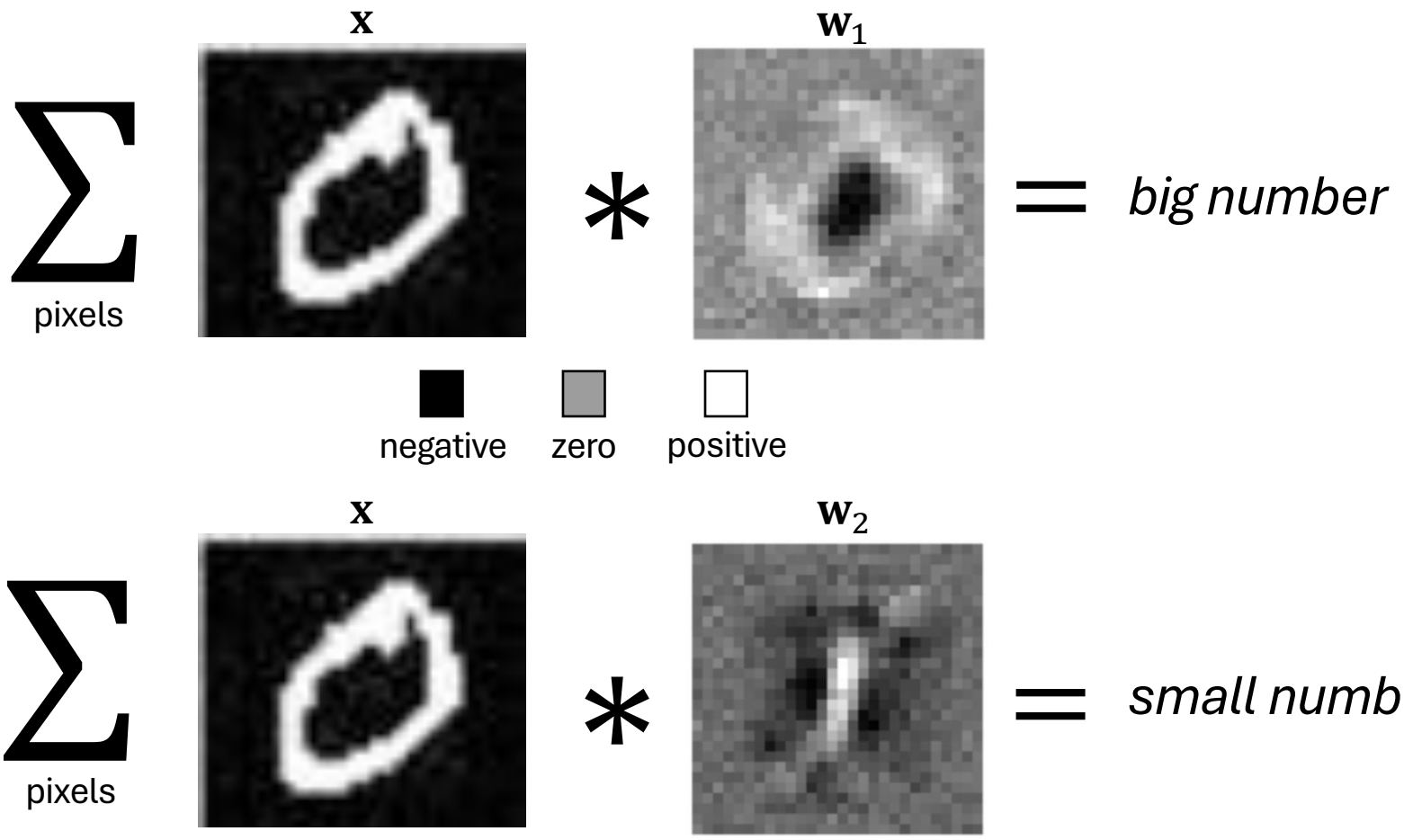




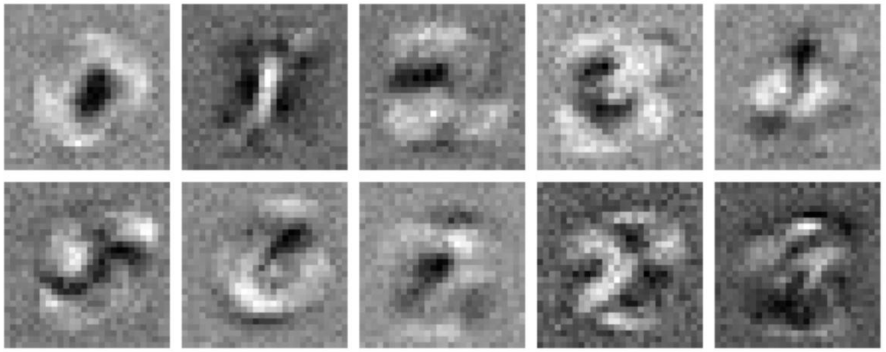
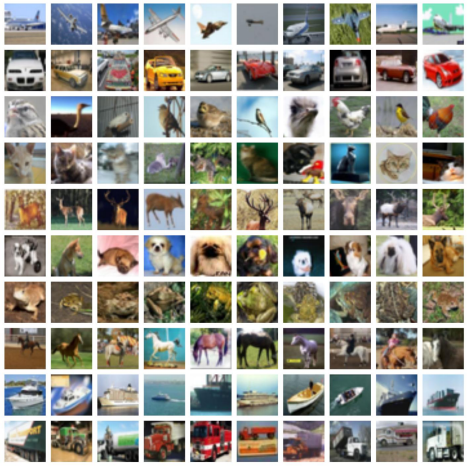
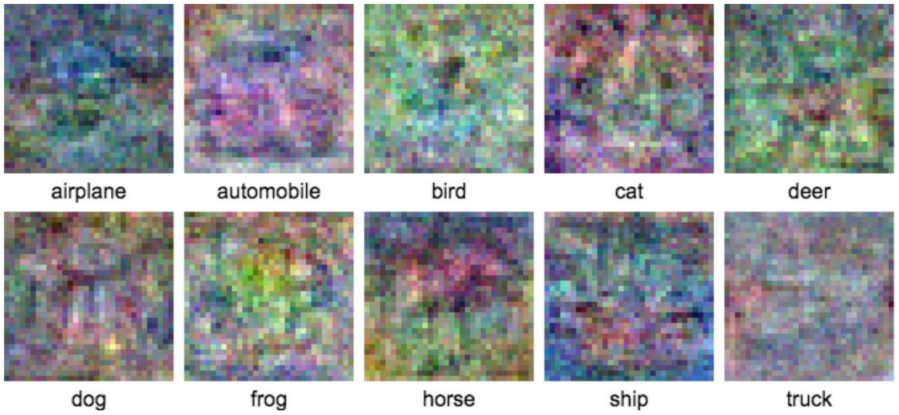




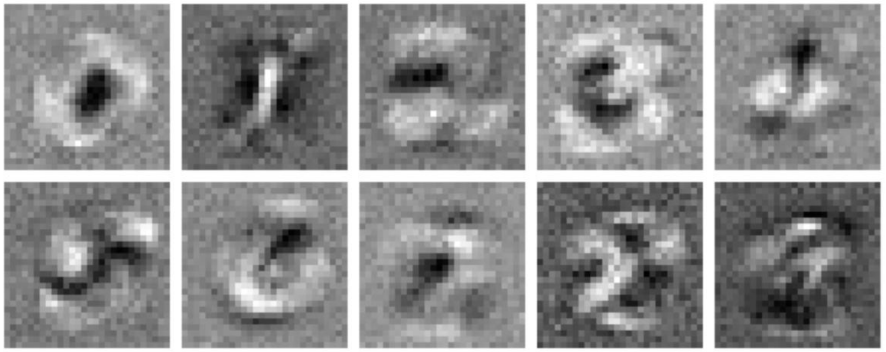
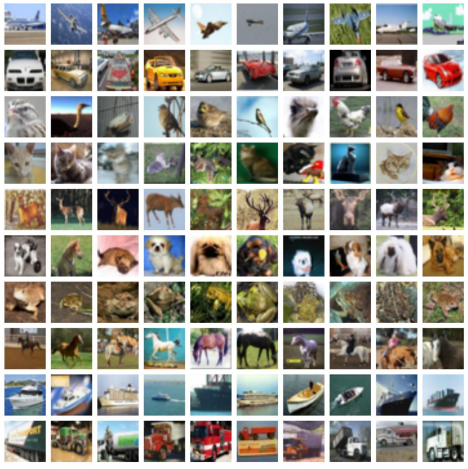
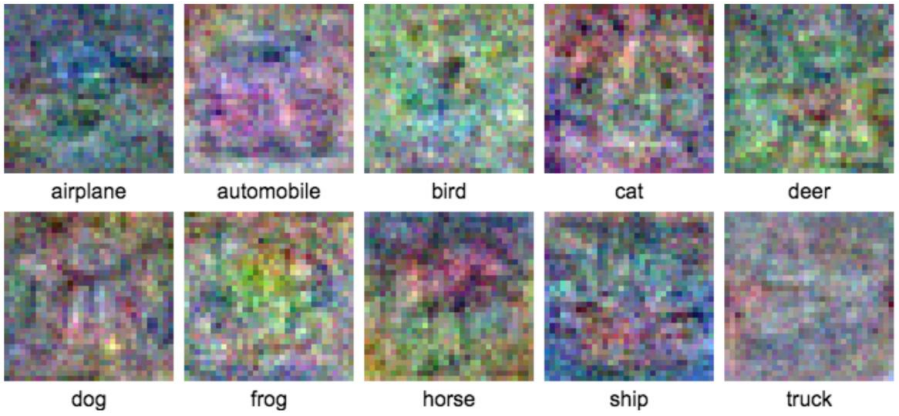
Linear Classifier



Linear Classifier

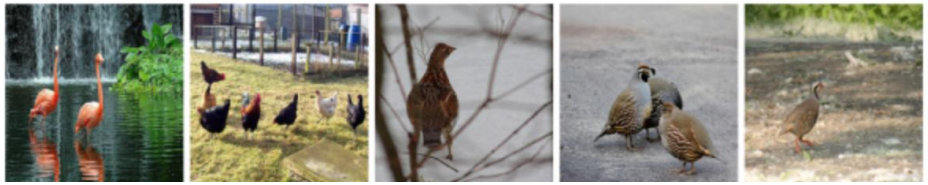
Dataset	Label	Images	Learned weights of linear classifier	Error
MNIST	“0” “1” “2” “3” “4” “5” “6” “7” “8” “9”			???
CIFAR-10	airplane automobile bird cat deer dog frog horse ship truck			???

Linear Classifier

Dataset	Label	Images	Learned weights of linear classifier	Error
MNIST	“0” “1” “2” “3” “4” “5” “6” “7” “8” “9”			8%
CIFAR-10	airplane automobile bird cat deer dog frog horse ship truck			63%

Why is it hard?

bird



cat

dog

- **Intra-class variability**
  - Viewpoint change
  - Occlusion
  - Illumination
  - Pose
  - Different species

Why is it hard?

- **Intra-class variability**
  - Viewpoint change
  - Occlusion
  - Illumination
  - Pose
  - Different species

Why is it hard?

- **Inter-class similarity**

Why is it hard?

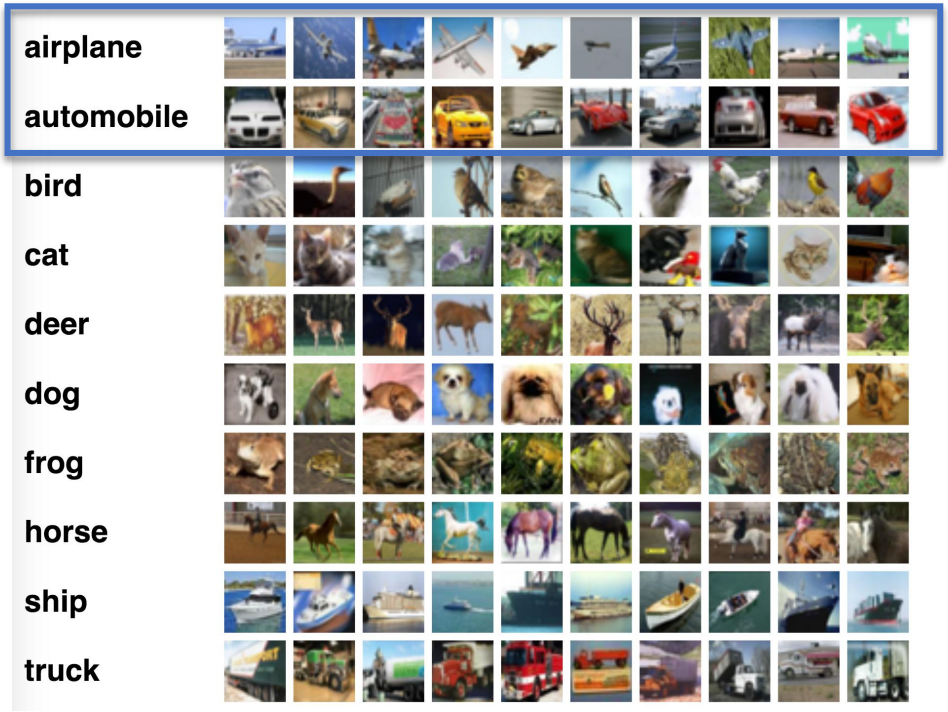
- Inter-class similarity

Why is it hard?

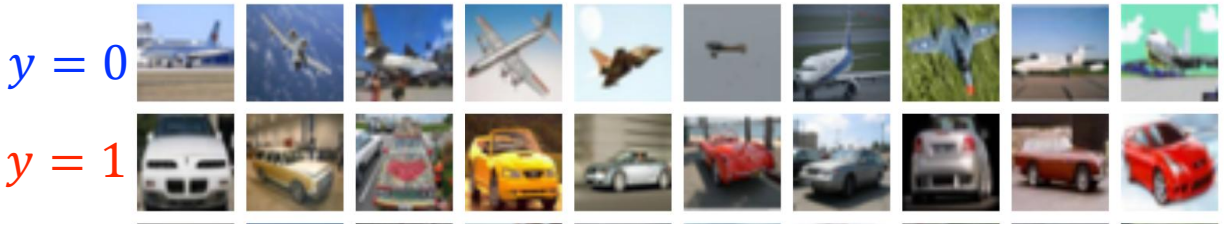
- Inter-class similarity

Binary Classification

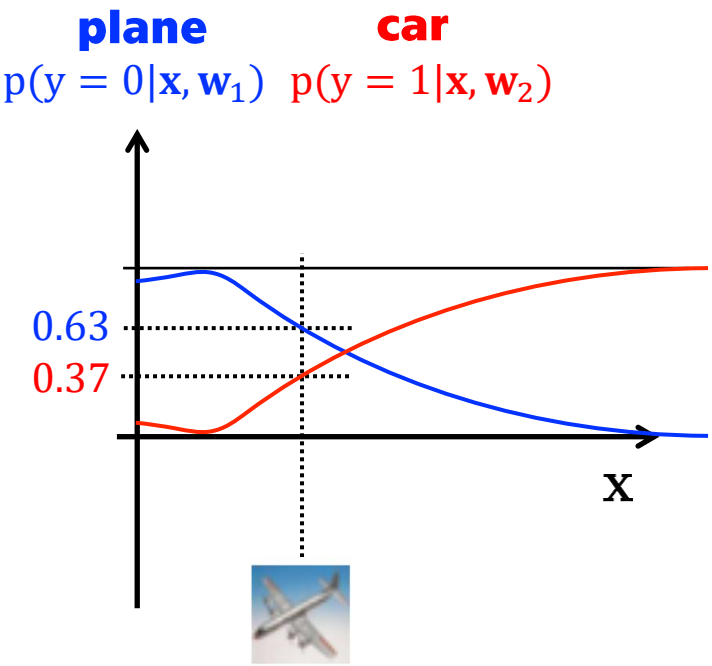
- Special but very important use case (K=2)



Binary Classification

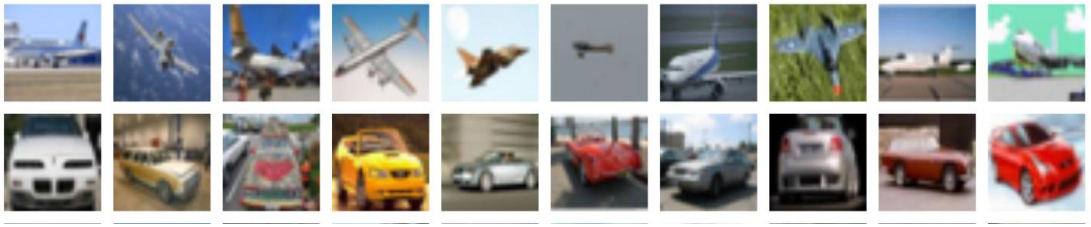


$$p(y \mid \bar{\mathbf{x}}, \mathbf{W}) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \end{bmatrix} / \sum_k \exp(f(\mathbf{x}, \mathbf{w}_k)) = \text{softmax}(\boldsymbol{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



Binary Classification

$y = 0$ $y = 1$



$$p(y \mid \bar{\mathbf{x}}, \mathbf{W}) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \end{bmatrix} / \sum_k \exp(f(\mathbf{x}, \mathbf{w}_k)) \equiv \text{softmax}(\boldsymbol{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

$$p(y \mid \bar{\mathbf{x}}, \mathbf{w}) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w})/2) \\ \exp(-f(\mathbf{x}, \mathbf{w})/2) \end{bmatrix} / (\exp(f(\mathbf{x}, \mathbf{w})/2) + \exp(-f(\mathbf{x}, \mathbf{w})/2))$$

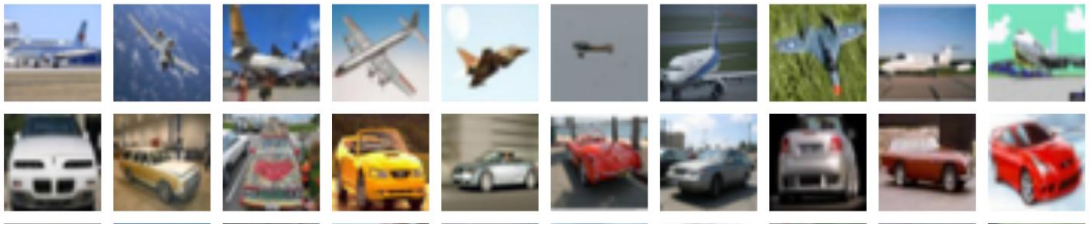
$$= \begin{bmatrix} \dfrac{1}{1 + \dfrac{\exp(-f(\mathbf{x}, \mathbf{w})/2)}{\exp(f(\mathbf{x}, \mathbf{w})/2)}} \\ 1 - \dfrac{1}{1 + \dfrac{\exp(-f(\mathbf{x}, \mathbf{w})/2)}{\exp(f(\mathbf{x}, \mathbf{w})/2)}} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1 + \exp(-f(\mathbf{x}, \mathbf{w}))} \\ 1 - \dfrac{1}{1 + \exp(-f(\mathbf{x}, \mathbf{w}))} \end{bmatrix}$$

$$= \begin{bmatrix} \sigma(f(\mathbf{x}, \mathbf{w})) \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) \end{bmatrix}$$

Binary Classification

$y = 0$

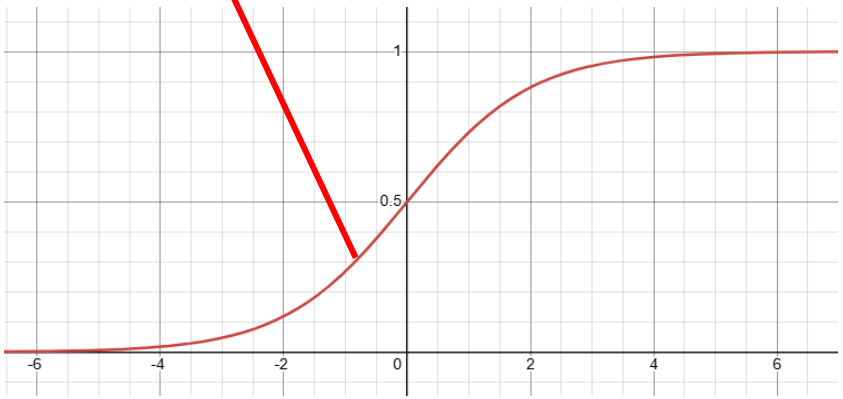
$y = 1$



$$p(y \mid \bar{\mathbf{x}}, \mathbf{W}) = \begin{bmatrix} \exp(f(\mathbf{x}, \mathbf{w}_1)) \\ \exp(f(\mathbf{x}, \mathbf{w}_2)) \end{bmatrix} / \sum_k \exp(f(\mathbf{x}, \mathbf{w}_k)) \equiv \text{softmax}(\boldsymbol{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

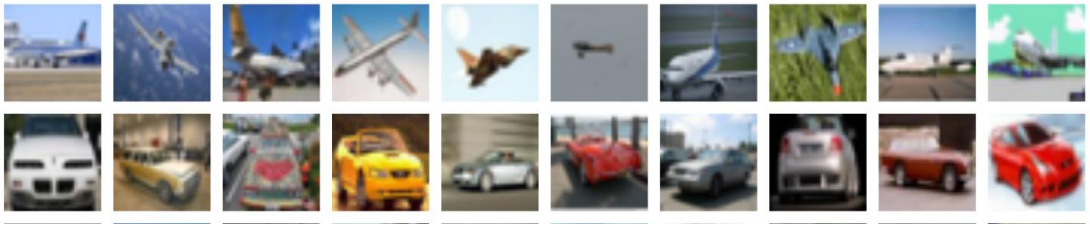
$$p(y \mid \bar{\mathbf{x}}, \mathbf{w}) = \begin{bmatrix} \frac{1}{1+\exp(-f(\mathbf{x},\mathbf{w}))} \\ 1 - \frac{1}{1+\exp(-f(\mathbf{x},\mathbf{w}))} \end{bmatrix} = \begin{bmatrix} \sigma(f(\mathbf{x}, \mathbf{w})) \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) \end{bmatrix}$$

sigmoid $\sigma(z)$



Binary Classification

$y = 0$ $y = 1$



$$\mathrm{p}(\mathrm{y} \mid \bar{\mathbf{x}}, \mathbf{w}) = \begin{bmatrix} \sigma(f(\mathbf{x}, \mathbf{w})) \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) \end{bmatrix} \in \mathbb{R}^2$$

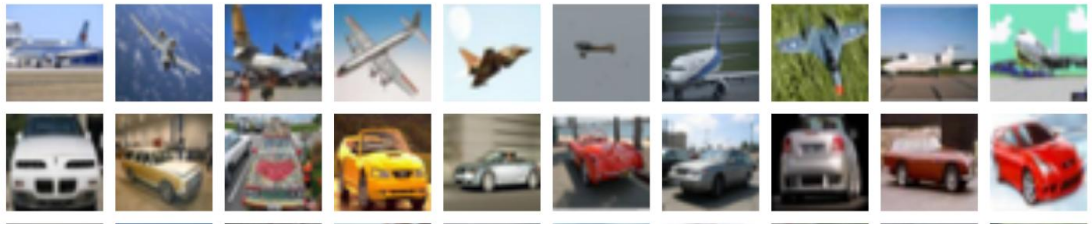
$$\mathrm{p}(\mathrm{y} \mid \bar{\mathbf{x}}, \mathbf{w}) = \begin{cases} \sigma(f(\mathbf{x}, \mathbf{w})) & \mathrm{y} = 1 \\ 1 - \sigma(f(\mathbf{x}, \mathbf{w})) & \mathrm{y} = 0 \end{cases} \in \mathbb{R}^1$$

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

Binary Classification

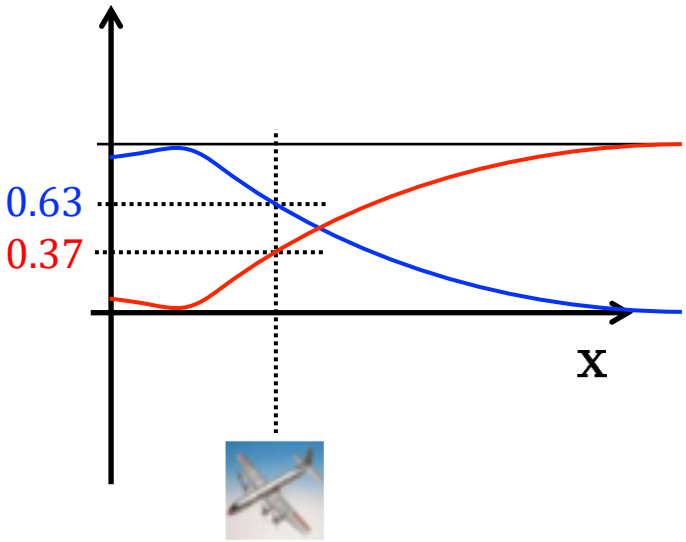
$y = 0$

$y = 1$



$$p(y \mid \bar{x}, w) = y\,\sigma(f(x, w)) + (1 - y)\,(1 - \sigma(f(x, w)))$$

plane $p(y = 0|x, w_1)$ **car** $p(y = 1|x, w_2)$



$$\begin{aligned}
\mathcal{L}_i &= -\log\left(p(y_i \mid x_i, w)\right) = \\
&= -\log\left(y_i\sigma(f(x_i, w)) + (1 - y_i)\,(1 - \sigma(f(x_i, w)))\right) \\
&= -\log\left(1 \cdot 0.63 + (1 - 1) \cdot (1 - 0.63)\right) \\
&= -\log\left(0.63\right) = 0.2
\end{aligned}$$

Binary Classification

- **Binary Cross-Entropy (BCE) Loss**

$$p(y \mid \bar{x}, w) = y \cdot \sigma(f(x, w)) + (1 - y)\,(1 - \sigma(f(x, w))) \qquad y \in \{0, 1\}$$

$$BCE(W) = -\sum_{i=1}^{N} \log\left(y_i \sigma(f(x_i, w)) + (1 - y_i)\,(1 - \sigma(f(x_i, w)))\right)$$

- **Logistic Loss**

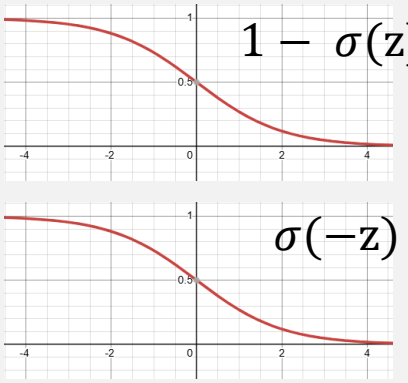
$$p(y \mid \bar{x}, w) = \sigma(y \cdot f(x, w)) \qquad y \in \{-1, +1\}$$

$$\mathcal{L}(W) = -\sum_{i=1}^{N} \log\left(y_i \sigma(f(x_i, w))\right)$$

$$= \sum_{i=1}^{N} \log\left(1 + \exp(y_i f(x_i, w))\right)$$

Identity

$$1 - \sigma(z) = \sigma(-z)$$



Binary Classification

- **Binary Cross-Entropy (BCE) Loss**

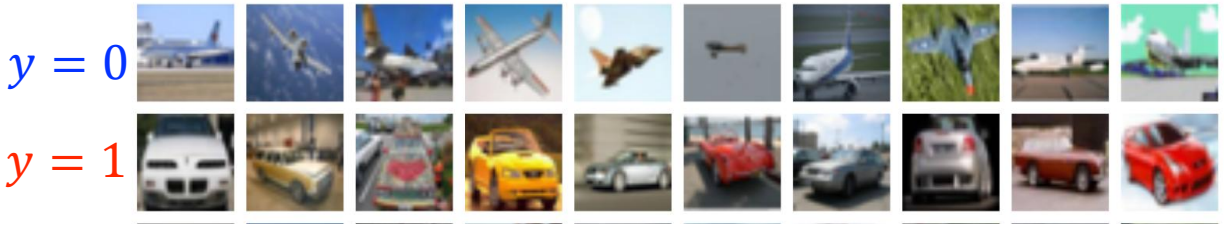
```
loss = y*(-torch.log(torch.sigmoid(w@x)))
       + (1-y)*(-torch.log(1-torch.sigmoid(w@x)))
tensor(3.9876, requires_grad=True)
```

- **Logistic Loss**

```
loss = -torch.log(torch.sigmoid(y*(w@x)))
tensor(3.9876, requires_grad=True)

loss = torch.log(1+torch.exp(-y*(w@x)))
tensor(3.9876, requires_grad=True)
```

Linear Classification

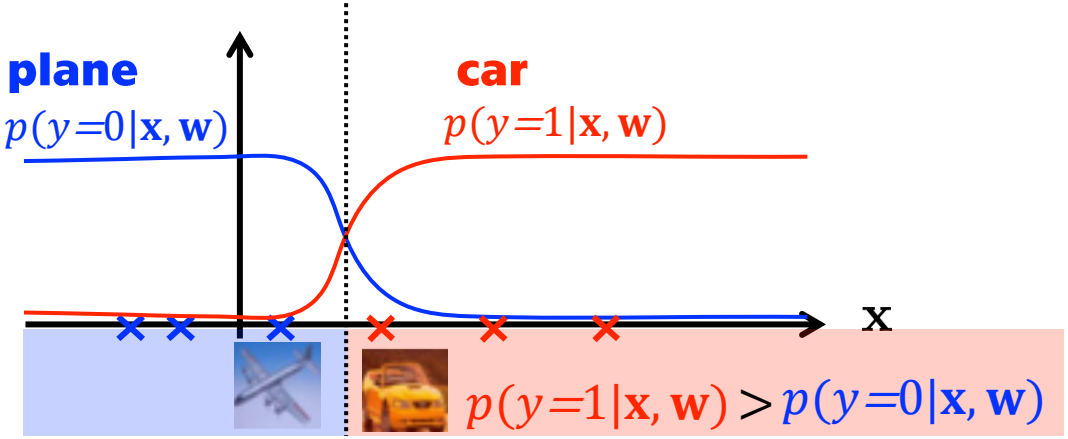


$$p(y \mid \bar{\mathbf{x}}, \mathbf{w}) = y \cdot \sigma(f(\mathbf{x}, \mathbf{w})) + (1 - y)\,(1 - \sigma(f(\mathbf{x}, \mathbf{w})))$$

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \bar{\mathbf{x}}$$

1D linear classifier

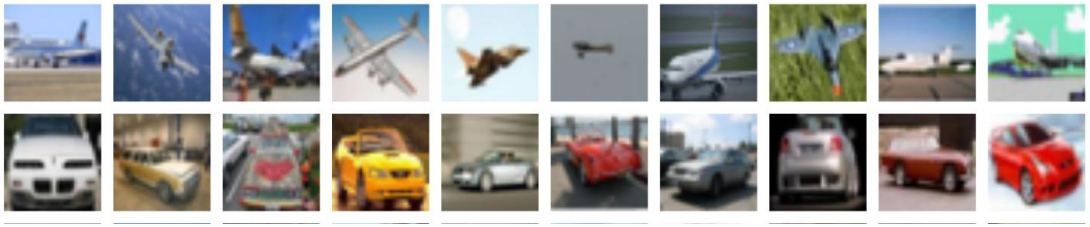
Which value of f(x) corresponds to this threshold? $f(\mathbf{x}, \mathbf{w}) = 0$



Linear Classification

$y = 0$

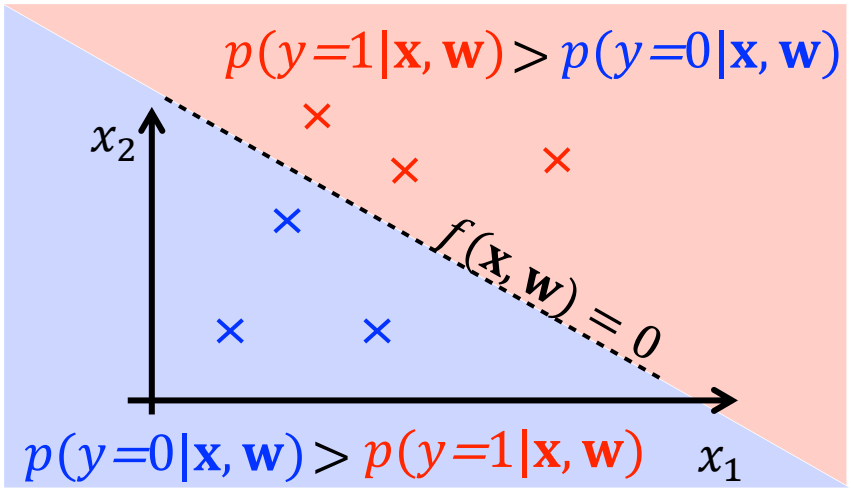
$y = 1$



$$p(y \mid \bar{\mathbf{x}}, \mathbf{w}) = y \cdot \sigma(f(\mathbf{x}, \mathbf{w})) + (1 - y)\,(1 - \sigma(f(\mathbf{x}, \mathbf{w})))$$

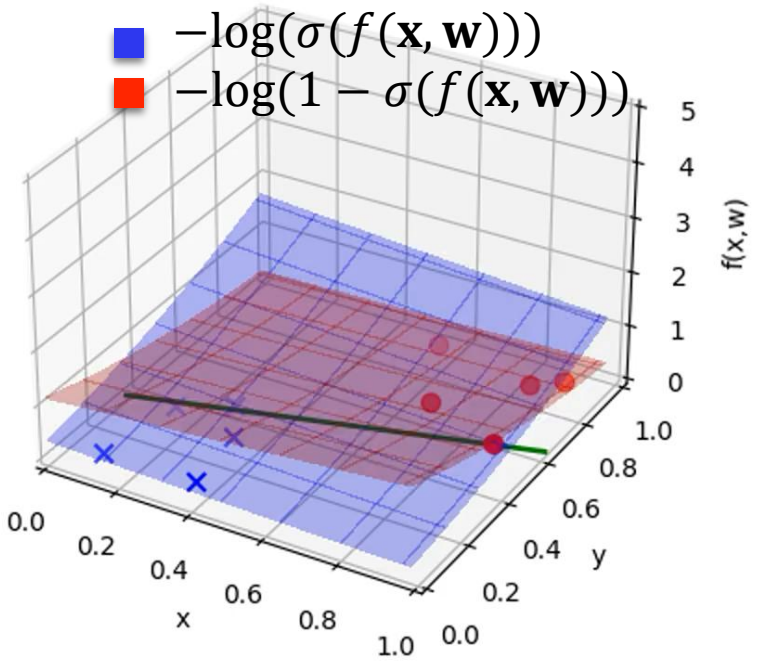
$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \bar{\mathbf{x}}$$

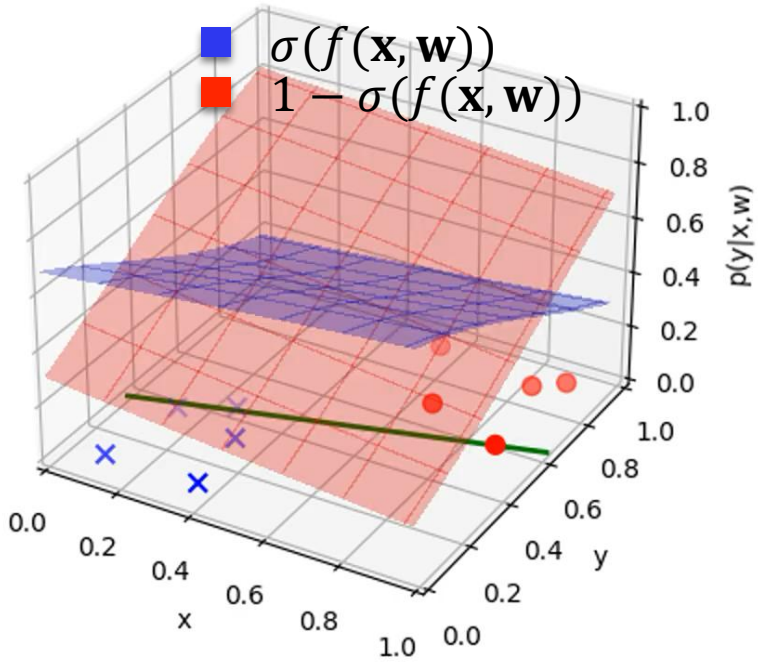
2D linear classifier



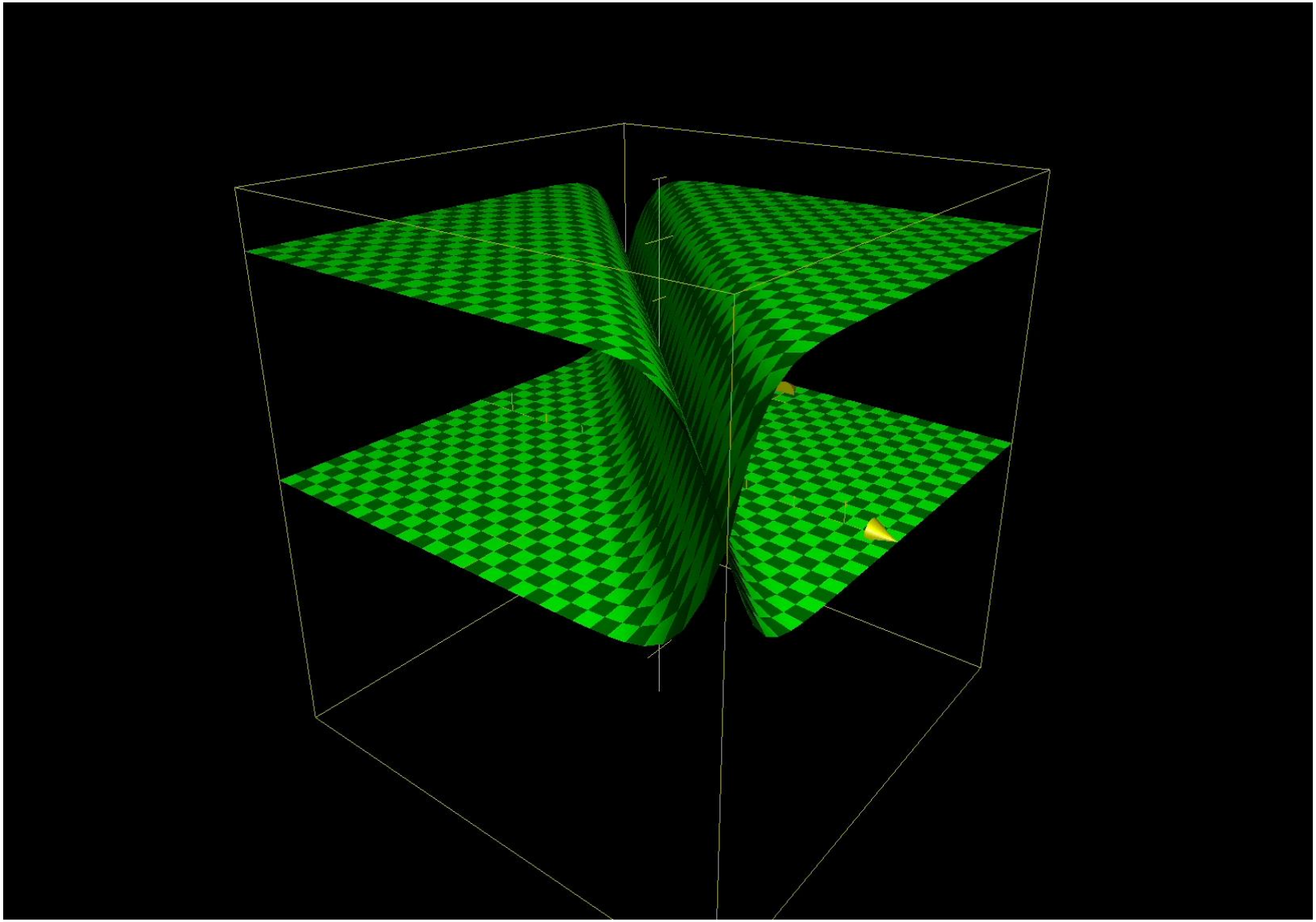
Training a Linear Classifier

```
def train_linear_classifier(x: np.ndarray, y: np.ndarray):
  w = np.array([-2.0, 2.0, 1.0]) # init
  for _ in range(30):
    f = w[0] * x[:, 0] + w[1] * x[:, 1] + w[2]
    p = sigmoid(u) # Get probability for class 0
    loss = (-np.log(p)*y + -np.log(1-p)*(1-y)).sum()
    grad = -1/p * sigmoid(f) * (1-sigmoid(f)) * … # Calculate gradient
    w = w - 0.1 * grad # Update model
```





Training a Linear Classifier



Training a Linear Classifier

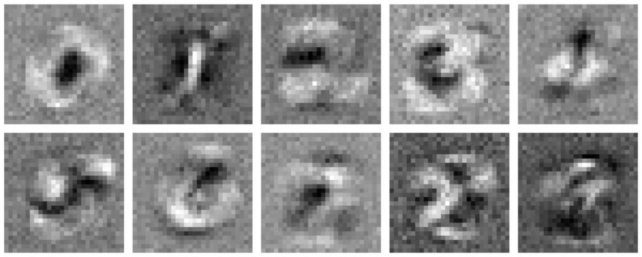
Images

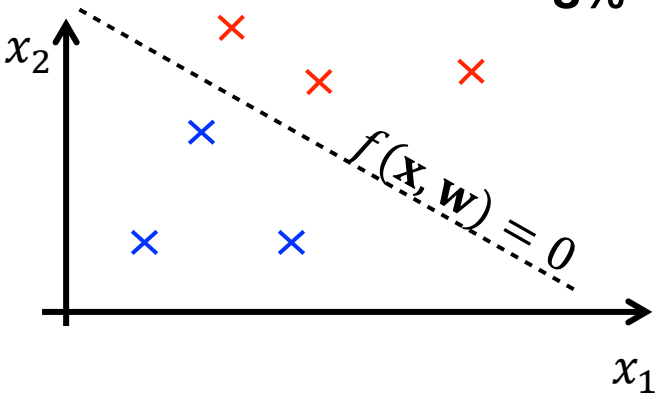
Learned weights of linear classifier

Error

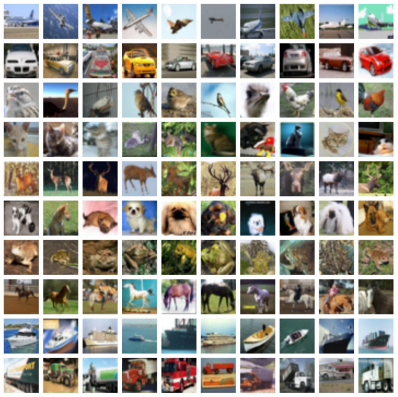
8%

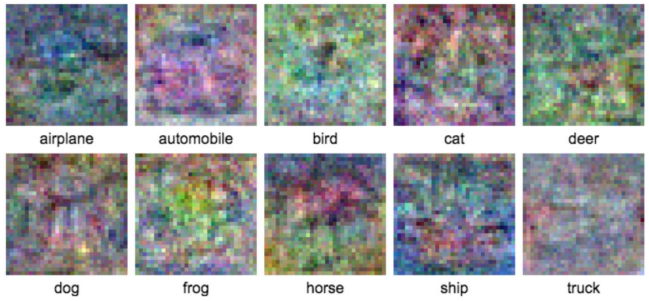
“0”
“1”
“2”
“3”
“4”
“5”
“6”
“7”
“8”
“9”



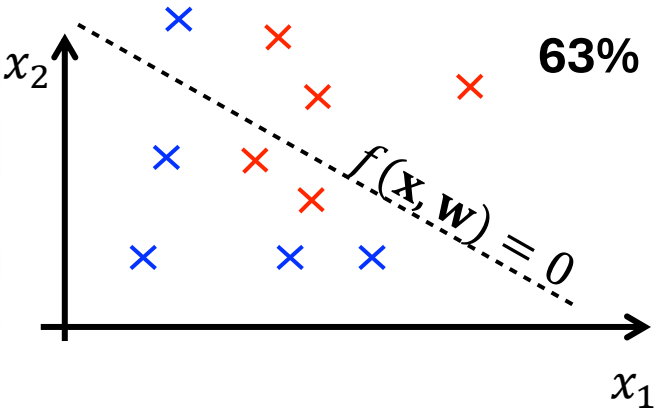


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck





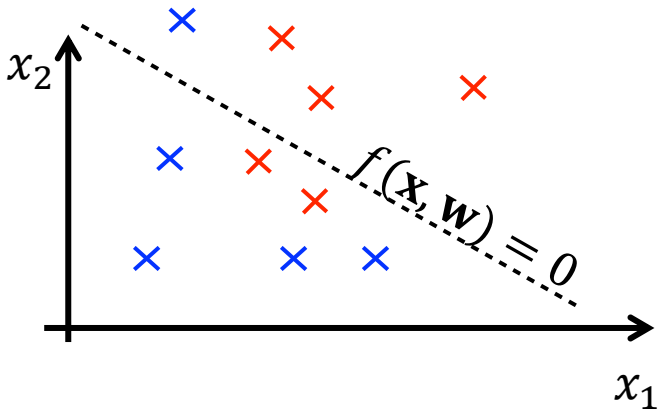
63%



Non-linear Classification

2D linear classifier

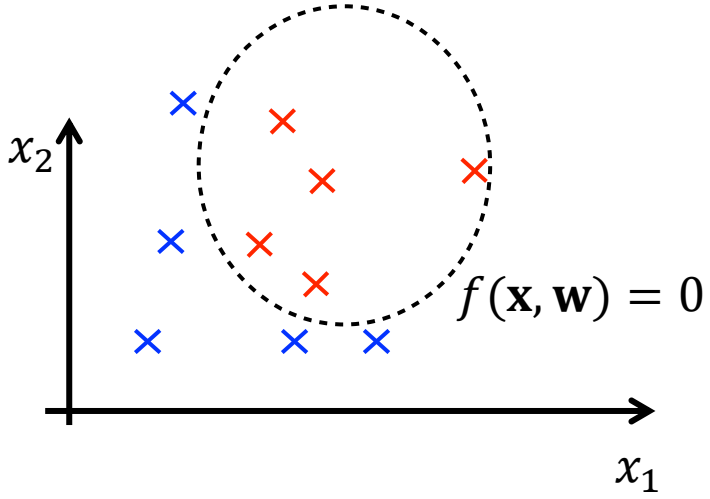
$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) = w_1 x_1 + w_2 x_2 + w_0$$



2D non-linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) =$$

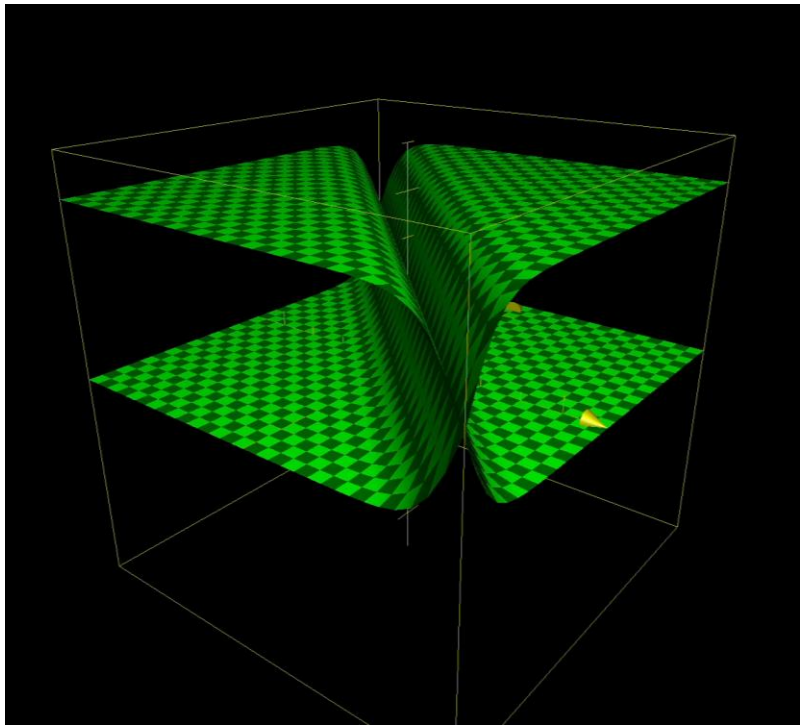
$$= w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$



Non-linear Classification

2D linear classifier

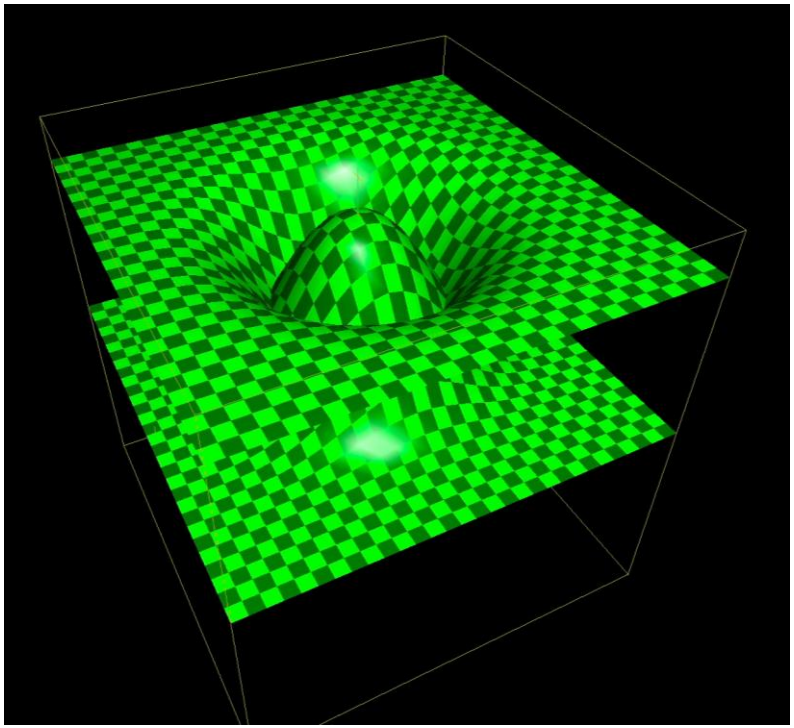
$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) = w_1 x_1 + w_2 x_2 + w_0$$



2D non-linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) =$$

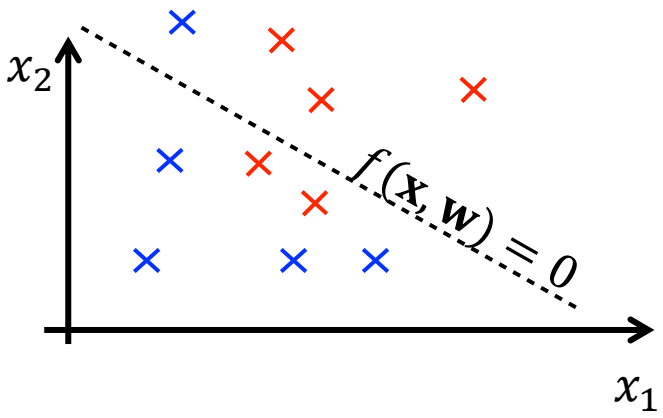
$$= w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$



Non-linear Classification

2D linear classifier

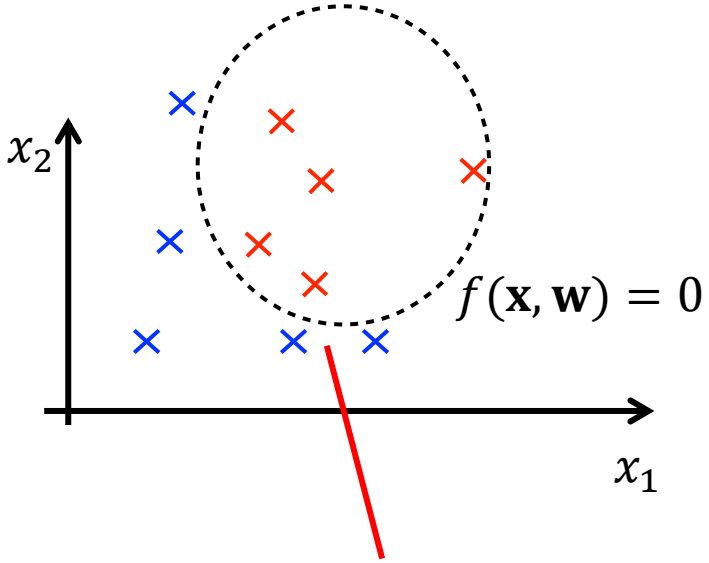
$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) = w_1 x_1 + w_2 x_2 + w_0$$



2D non-linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) =$$

$$= w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$



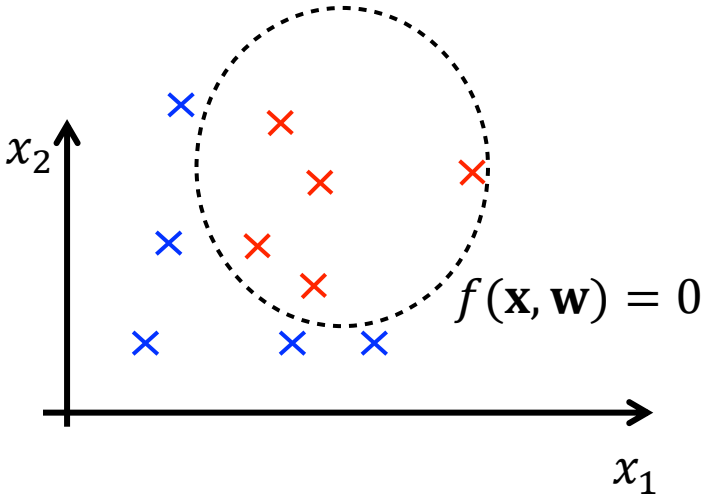
Is there a linear classifier that separates this data?

Non-linear Classification

5D linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix}, \mathbf{w}\right) =$$

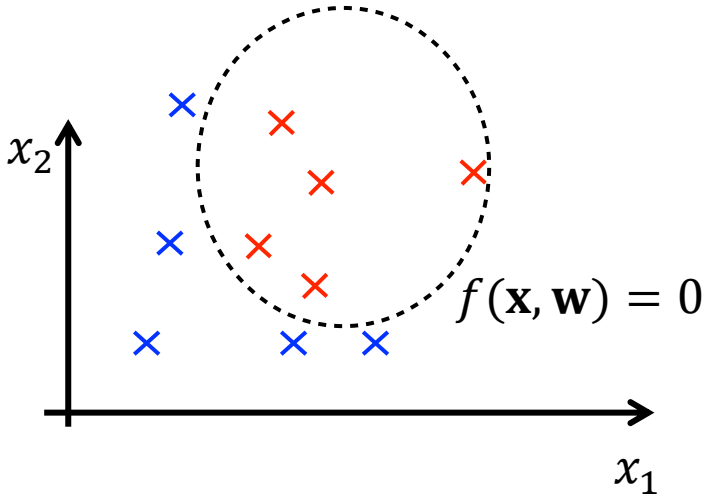
$$= w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$



2D non-linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) =$$

$$= w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$



Non-linear Classification

5D linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix}, \mathbf{w}\right) = = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$

2D non-linear classifier

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{w}\right) = = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0$$

- **Two different options how to view deep learning**
  1. Learning non-linear features so that they are linearly separable
  2. Learning non-linear classifier that separate complicated feature areas

Competencies gained for the test

- Classification loss for two-class and K-class problem
- Equivalence of common binary classification losses
- Meaning of weights in linear classifier
- 2D visualization, decision boundary, features
- Benefits and drawbacks of a linear classifier