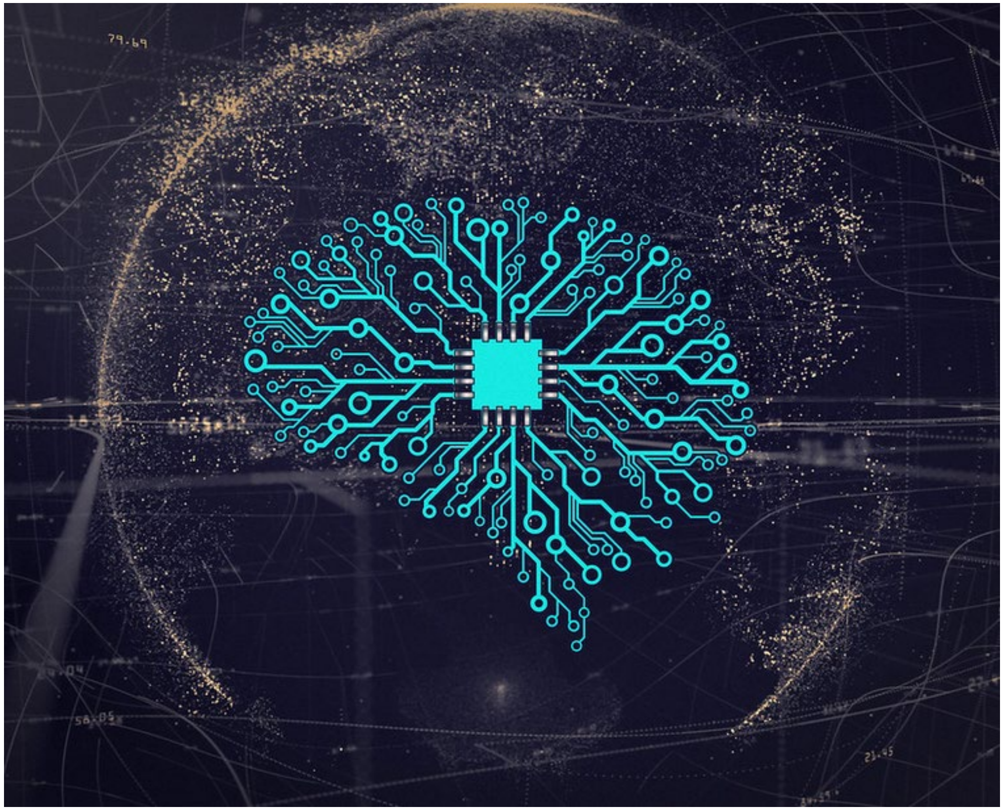
# Part I: Intro to Learning Basics



Intelligent Visual Computing

Evangelos Kalogerakis

# Learning basics: Classification example

Suppose you want to predict **spam** or **no spam** for an email

**Output:** $y = 1 \ [spam], \quad y = 0 \ [no\, spam]$

**Input:** $\mathbf{x} = [x_1, x_2, \ldots] \ where \ x_1 = \text{freq. of word "account" in email}$

$x_2 = \text{freq. of word "bank" in email}$

$x_3 = \text{freq. of word "credit" in email etc}$

Dear Beloved Friend,
I know this message will come to you as surprised but permit me of my desire to go into business relationship with you.
I am Miss Naomi Surugaba a daughter to late Al-badari Surugaba of Libya whom was murdered during the recent civil war in Libya in March 2011, before his death my late father was a strong supporter and a member of late Moammar Gadhafi Government in Tripoli.
Meanwhile before the incident, my late Father came to Cotonou Benin republic with the sum of USD4, 200,000.00 (US$4.2M) which he deposited in a Bank here in Cotonou Benin Republic West Africa for safe keeping.
I am here seeking for an avenue to transfer the fund to you in only you`re reliable and trustworthy person to Investment the fund. I am here in Benin Republic because of the death of my parent`s and I want you to help me transfer the fund into your bank account for investment purpose.
Please I will offer you 20% of the total sum of USD4.2M for your assistance. Please I wish to transfer the fund urgently without delay into your account and also wish to relocate to your country due to the poor condition in Benin, as to enable me continue my education as I was a medical student before the sudden death of my parent`s. Reply to my alternative email:missnaomisurugaba2@hotmail.com, Your immediate response would be appreciated.
Remain blessed,

**Spam?**

# Learning basics: Classification example

Suppose you want to predict **mug** or **no mug** for a shape.

**Output:** $y = 1\ [\mathit{coffee\,mug}], \quad y = 0\ \ [\mathit{no\,coffee\,mug}]$

**Input:** $\mathbf{x} = [x_1, x_2, \ldots]\ \mathit{where}\ x_1 = \text{pixel } (0,0),\ \text{x}_2 = \text{pixel } (0,1),\ \mathit{etc}$

**Mug?**

**Mug?**

**Mug?**

# Learning basics: Classification

Suppose you want to predict **mug** or **no mug** for a shape.

**Output:** $y = 1 \;\; [coffee\, mug], \quad y = 0 \;\; [no\, coffee\, mug]$

**Input:** $\mathbf{x} = [x_1, x_2, \ldots]$ *where* $x_1$=pixel (0,0), $\mathrm{x}_2$ = pixel (0,1), *etc*

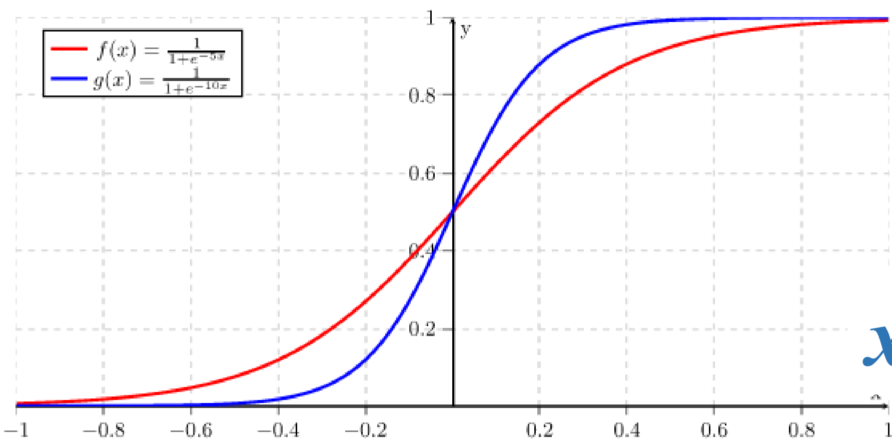
Classification function:

$$P(\mathbf{y} = \mathbf{1} \mid \mathbf{x}) = \mathbf{f}(\mathbf{x}) = \sigma(\mathbf{x}^T \cdot \mathbf{w})$$

where $\mathbf{w}$ is a **weight vector**

$$\sigma(\mathbf{x}^T \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{x}^T \cdot \mathbf{w})}$$

$P(\mathbf{y} = \mathbf{1} \mid \mathbf{x})$




$\boldsymbol{x}$

# Learning basics: Classification

Suppose you want to predict **mug** or **no mug** for a shape.

**Output:** $y = 1 \ \ [\mathit{coffee\,mug}], \quad y = 0 \ \ [\mathit{no\,coffee\,mug}]$

**Input:** $\mathbf{x} = [x_1, x_2, \ldots]$ *where* $x_1$=pixel (0,0), $\mathrm{x}_2$ = pixel (0,1), *etc*

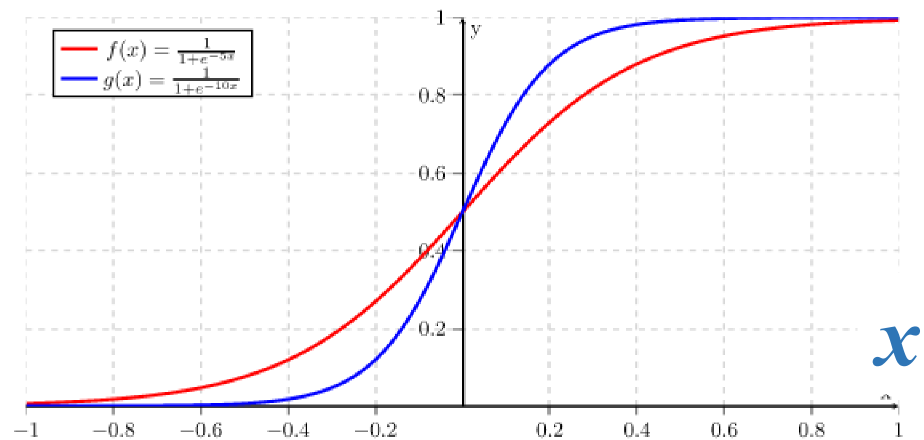
Classification function:

$$P(\mathbf{y} = \mathbf{1} \mid \mathbf{x}) = \mathbf{f}(\mathbf{x}) = \sigma(\mathbf{x}^T \cdot \mathbf{w})$$

where $\mathbf{w}$ is a **weight vector**

$$\sigma(\mathbf{x}^T \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{x}^T \cdot \mathbf{w})}$$

$P(\mathbf{y} = \mathbf{1} \mid \mathbf{x})$



# Learning basics: Classification

Suppose you want to predict **mug** or **no mug** for a shape.

**Output:** $y = 1 \ [coffee\, mug], \quad y = 0 \ \ [no\, coffee\, mug]$

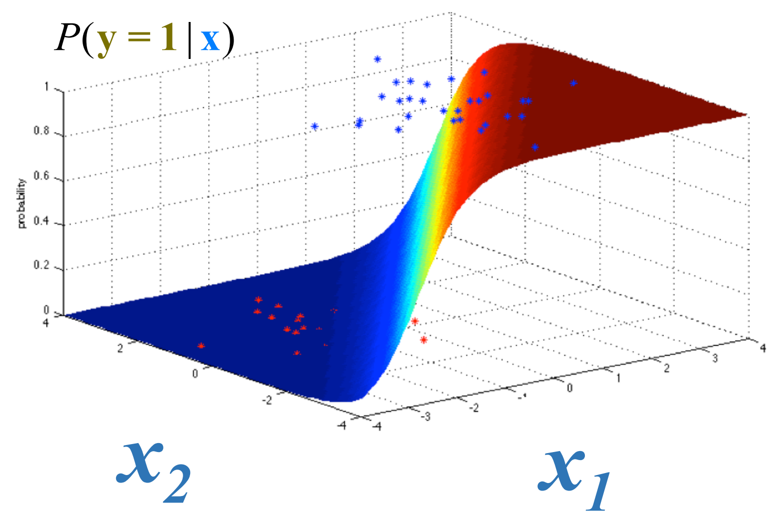
**Input:** $\mathbf{x} = [x_1, x_2, \ldots]$ *where* $x_1$=pixel (0,0), $x_2$ = pixel (0,1), *etc*

Classification function:

$$P(\mathbf{y} = \mathbf{1} \,|\, \mathbf{x}) = \mathbf{f}(\mathbf{x}) = \sigma(\mathbf{x}^T \cdot \mathbf{w})$$

where $\mathbf{w}$ is a **weight vector**

$$\sigma(\mathbf{x}^T \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{x}^T \cdot \mathbf{w})}$$



# Training

Need to estimate parameters $w$ from training data e.g., shapes of objects $\mathbf{x}_i$ and given labels $\mathbf{y}_i$ (mugs/no mugs) *(i=1…K* training shapes*)*

Find parameters that **maximize probability of training data**

$$\max_{\mathbf{w}} \prod_{i=1}^{K} P(\mathbf{y}=1 \mid \mathbf{x}_i)^{[\mathrm{y}_i^{(gt)}==1]} [1-P(\mathbf{y}=1 \mid \mathbf{x}_i)]^{[\mathrm{y}_i^{(gt)}==0]}$$

# Training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) *(i=1…K* training shapes*)*

Find parameters that **maximize probability of training data**

$$\max_{\mathbf{w}} \prod_{i=1}^{K} \sigma(\mathbf{x}_i{}^T \cdot \mathbf{w})^{[\mathrm{y}_i^{(gt)}==1]} [1-\sigma(\mathbf{x}_i{}^T \cdot \mathbf{w})]^{[\mathrm{y}_i^{(gt)}==0]}$$

# Training

Need to estimate parameters $\mathbf{w}$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) *(i=1…K* training shapes*)*

Find parameters that **maximize the log prob. of training data**

$$\max_{\mathbf{w}} \log\{\prod_{i=1}^{K}\sigma(\mathbf{x}_i^{T}\cdot\mathbf{w})^{[\mathrm{y}_i^{(gt)}==1]}[1-\sigma(\mathbf{x}_i^{T}\cdot\mathbf{w})]^{[\mathrm{y}_i^{(gt)}==0]}\}$$

# Training

Need to estimate parameters $w$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) $(i=1 \dots K$ training shapes$)$

Find parameters that **maximize the log prob. of training data**

$$\max_{\mathbf{w}} \sum_{i=1}^{K} [\mathrm{y}_i^{(gt)} == 1] \log \sigma(\mathbf{x}_i^T \cdot \mathbf{w}) + [\mathrm{y}_i^{(gt)} == 0] \log(1 - \sigma(\mathbf{x}_i^T \cdot \mathbf{w}))$$

# Training

Need to estimate parameters $w$ from training data e.g., shapes of objects $\mathbf{x}_i$ and given labels $\mathbf{y}_i$ (mugs/no mugs) $(i=1 \ldots K$ training shapes$)$

This is called **log-likelihood**

$$\max_{\mathbf{w}} \boxed{\sum_{i=1}^{K} [\mathrm{y}_i^{(gt)} == 1] \log \sigma(\mathbf{x}_i^T \cdot \mathbf{w}) + [\mathrm{y}_i^{(gt)} == 0] \log(1 - \sigma(\mathbf{x}_i^T \cdot \mathbf{w}))}$$

$$\boldsymbol{L(w)}$$

# Training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) *(i=1…K* training shapes*)*

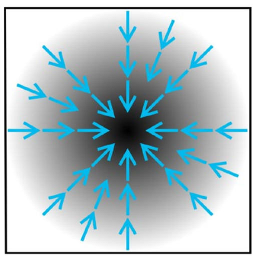
We have an **optimization problem.**

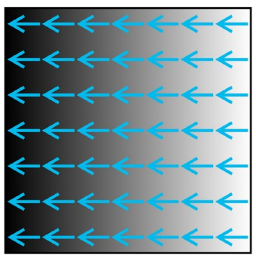
$$\max_{\mathbf{w}} \underbrace{\sum_{i=1}^{K} [\mathrm{y}_i^{(gt)} == 1] \log \sigma(\mathbf{x}_i{}^T \cdot \mathbf{w}) + [\mathrm{y}_i^{(gt)} == 0] \log(1-\sigma(\mathbf{x}_i{}^T \cdot \mathbf{w}))}_{L(w)}$$

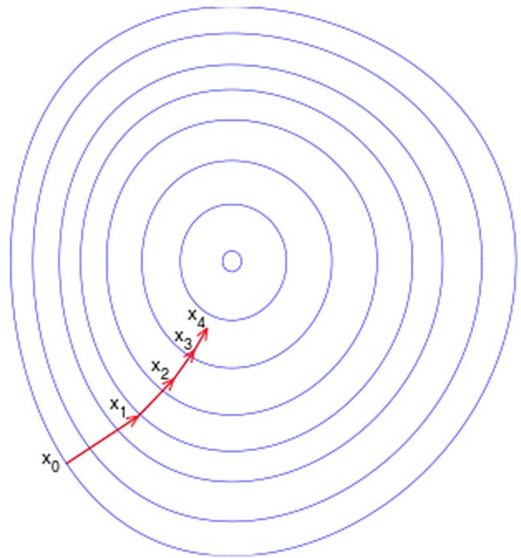
$$\frac{\partial L(\mathbf{w})}{\partial w_d} = \sum_i x_{i,d} [\, \mathrm{y}_i^{(gt)} - \sigma(\mathbf{x}_i{}^T \cdot \mathbf{w}) \,]$$

(partial derivative for $d^{th}$ parameter)

# How can we maximize a function?



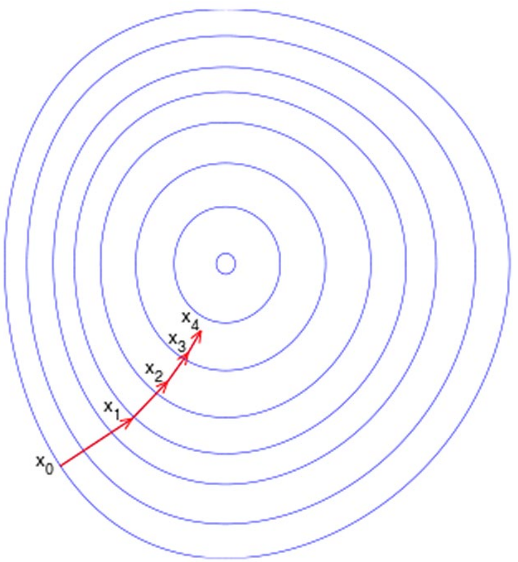





**Follow the gradient!** Given a random initialization of parameters and a step rate $\eta$, update them according to:

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \eta \nabla L(\mathbf{w})$$

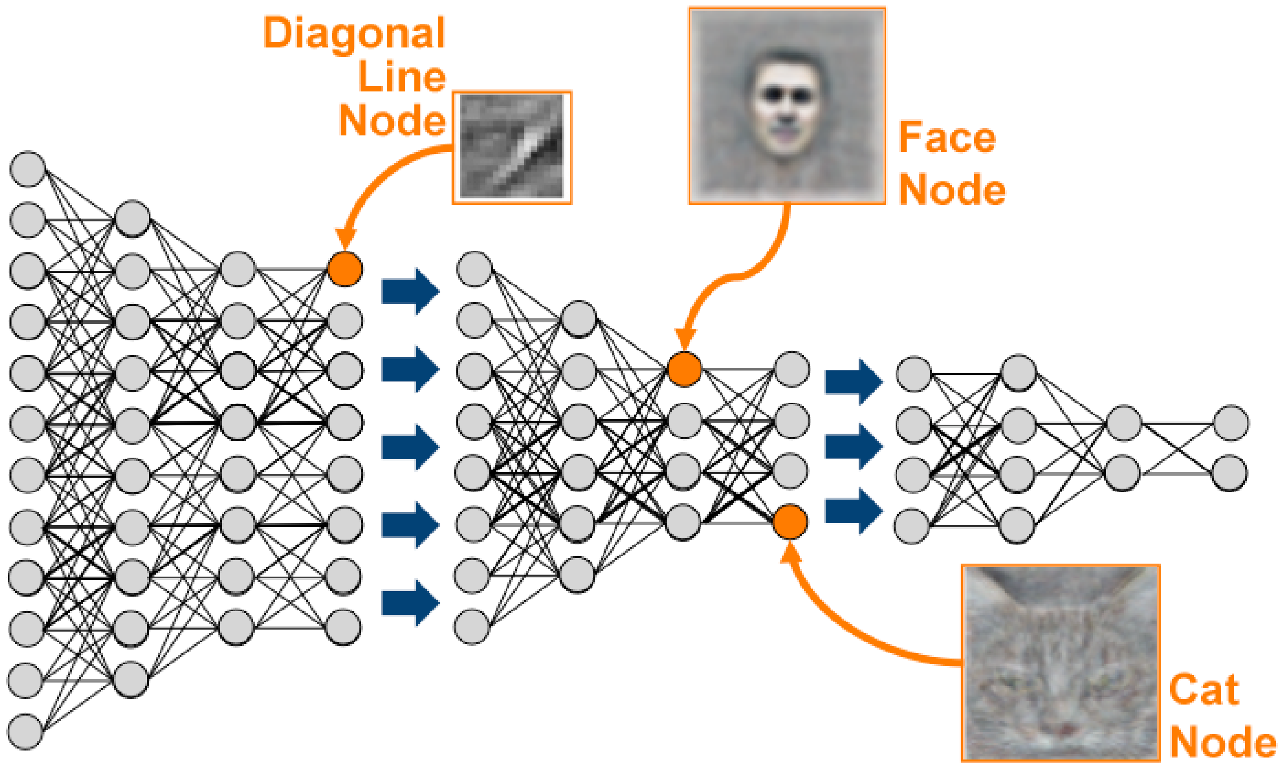
# How can we minimize a function?



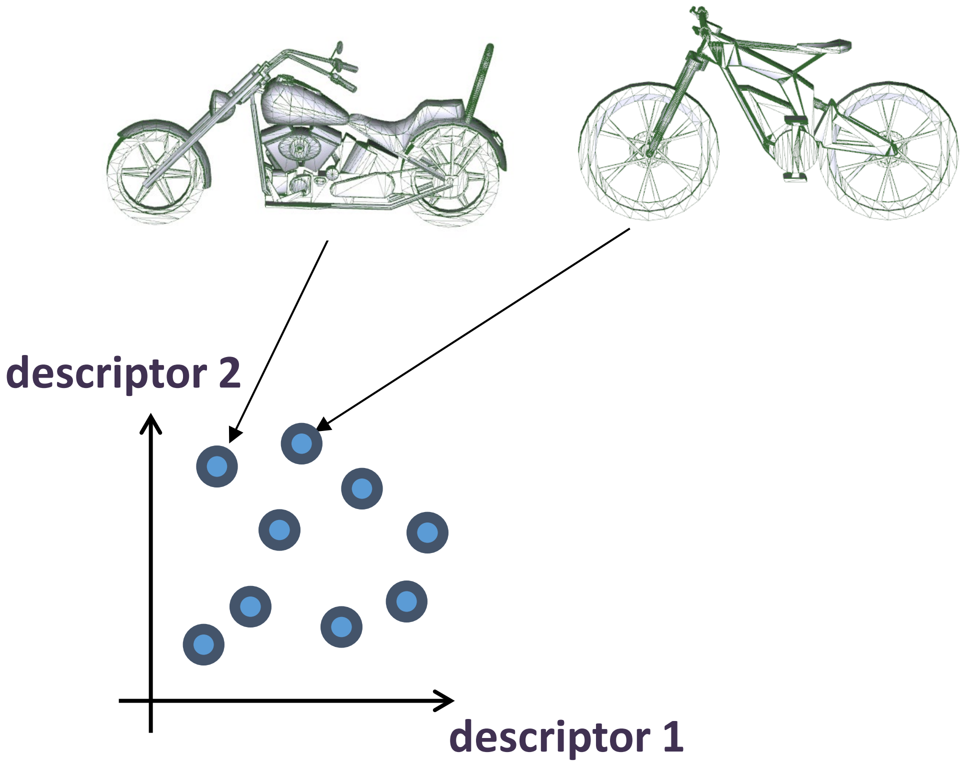

**Gradient descent:** Given a random initialization of parameters and a step rate $\eta$, update them according to:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \eta \nabla L(\mathbf{w})$$

# Part II: Neural Network Intro



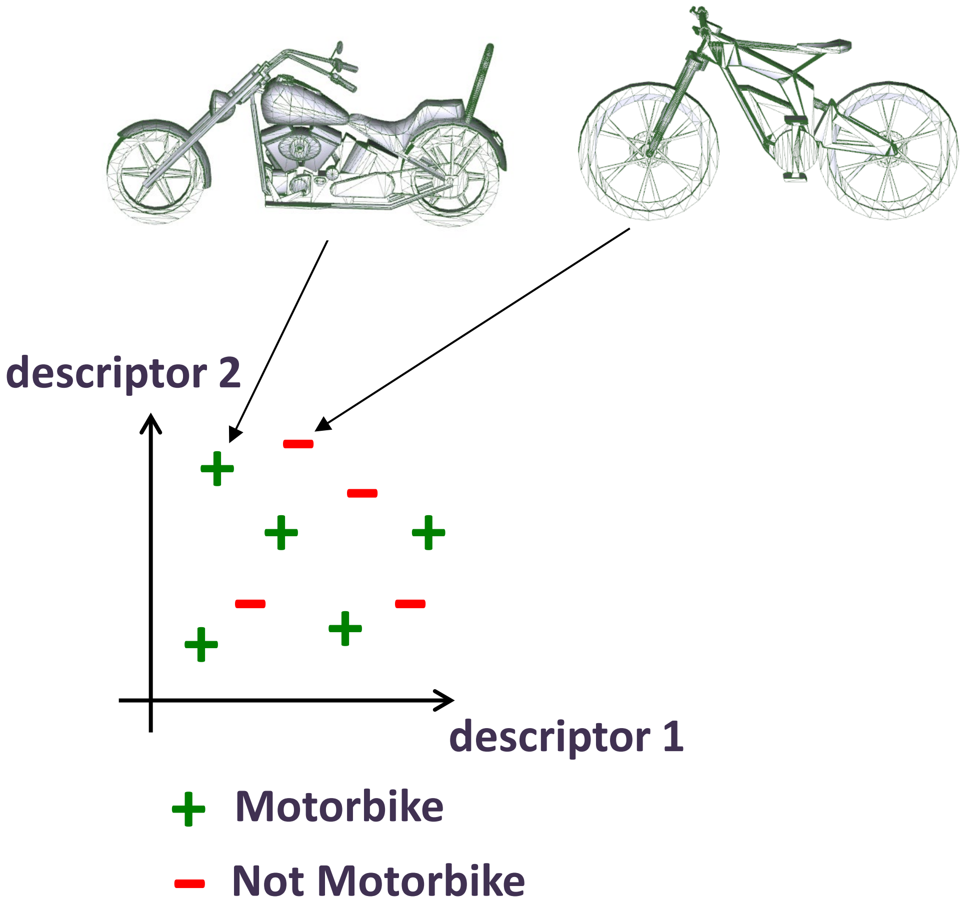
# The importance of good descriptors




**“Traditional approach”:**

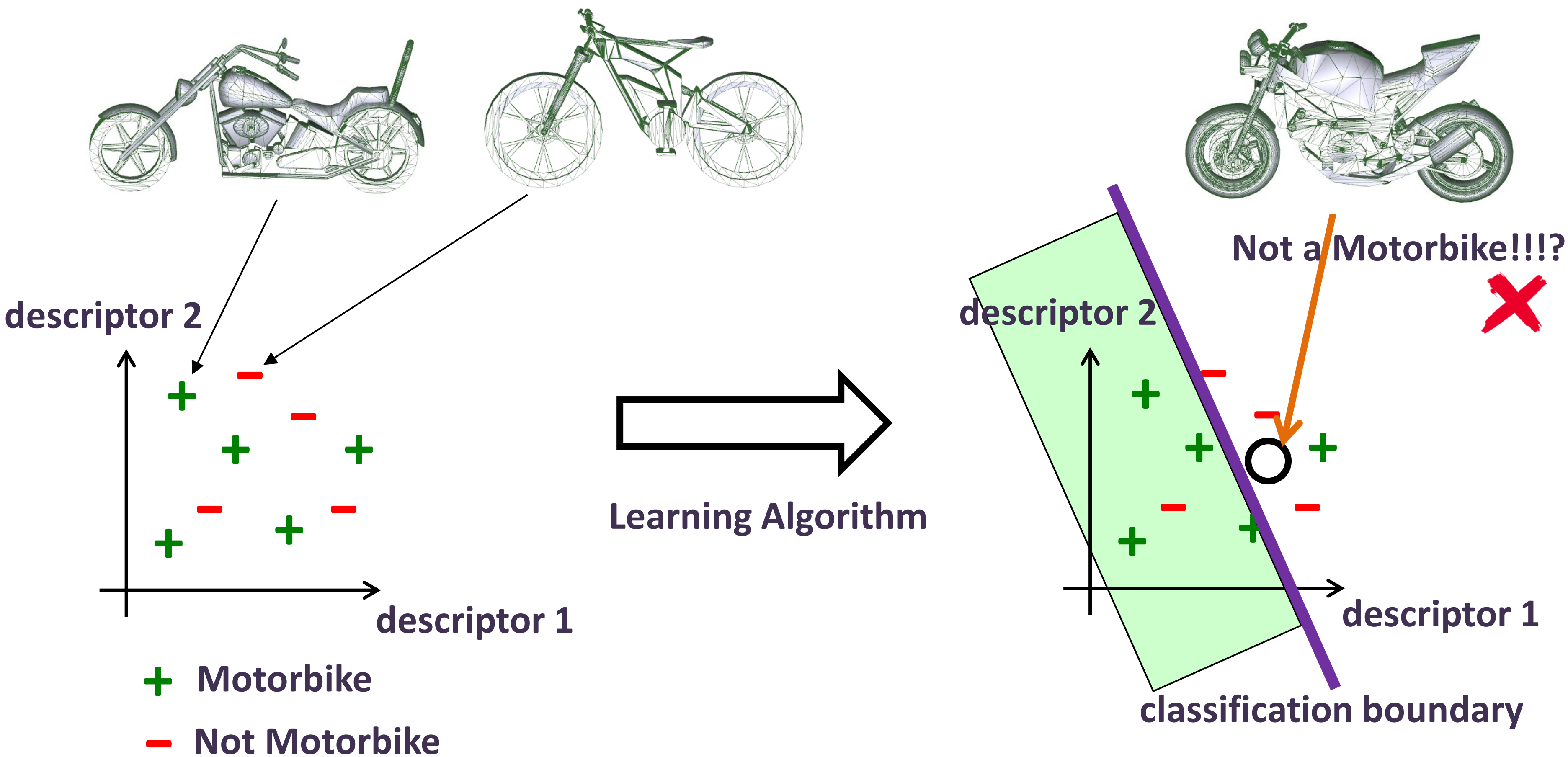
**Engineer descriptors: shape curvature, histograms of normals, pixel intensities…**

# The importance of good shape descriptors



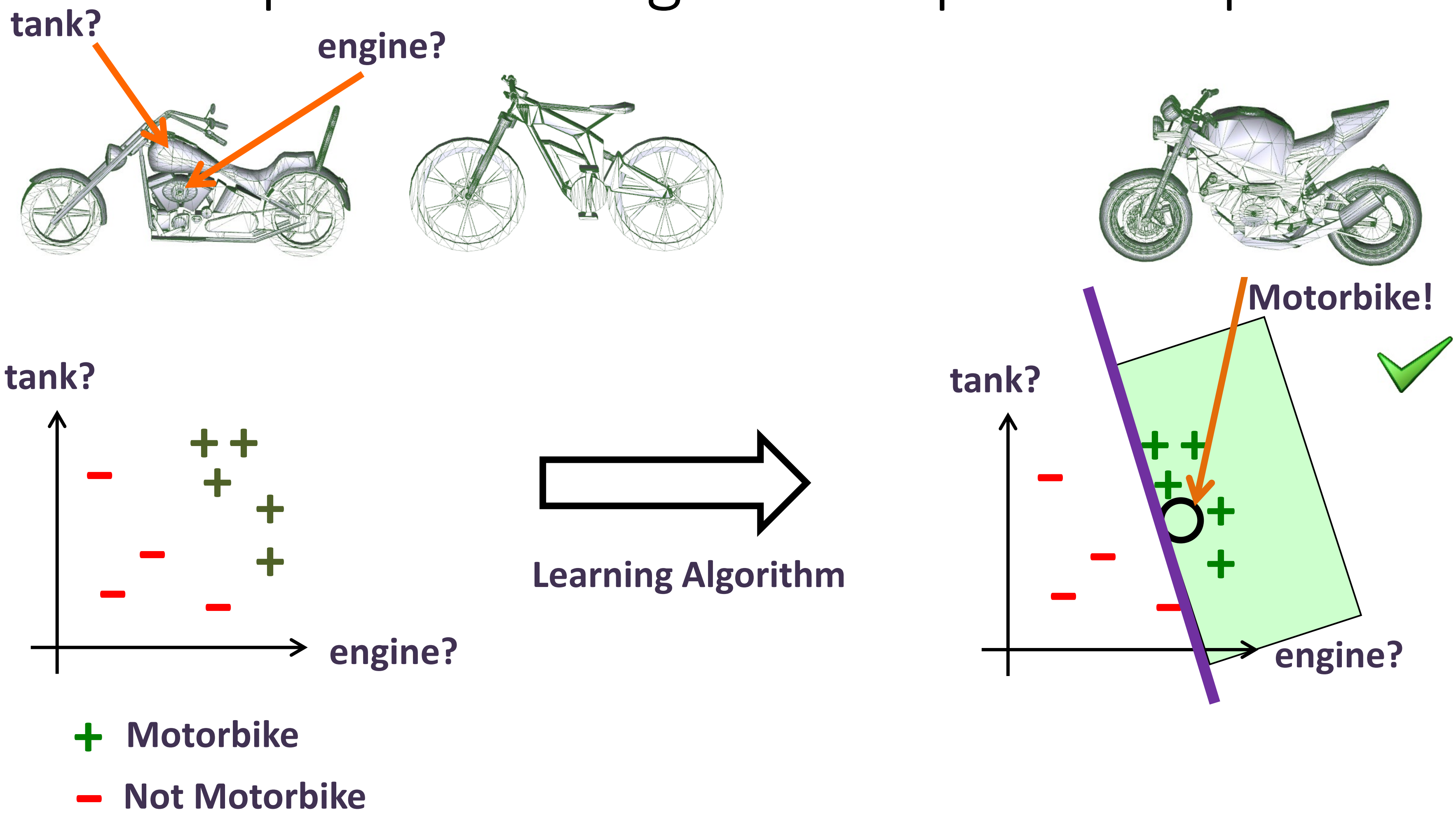

**Gather training labels**

# The importance of good shape descriptors




**Train a classifier.... easily fails to generalize**

# The importance of good shape descriptors

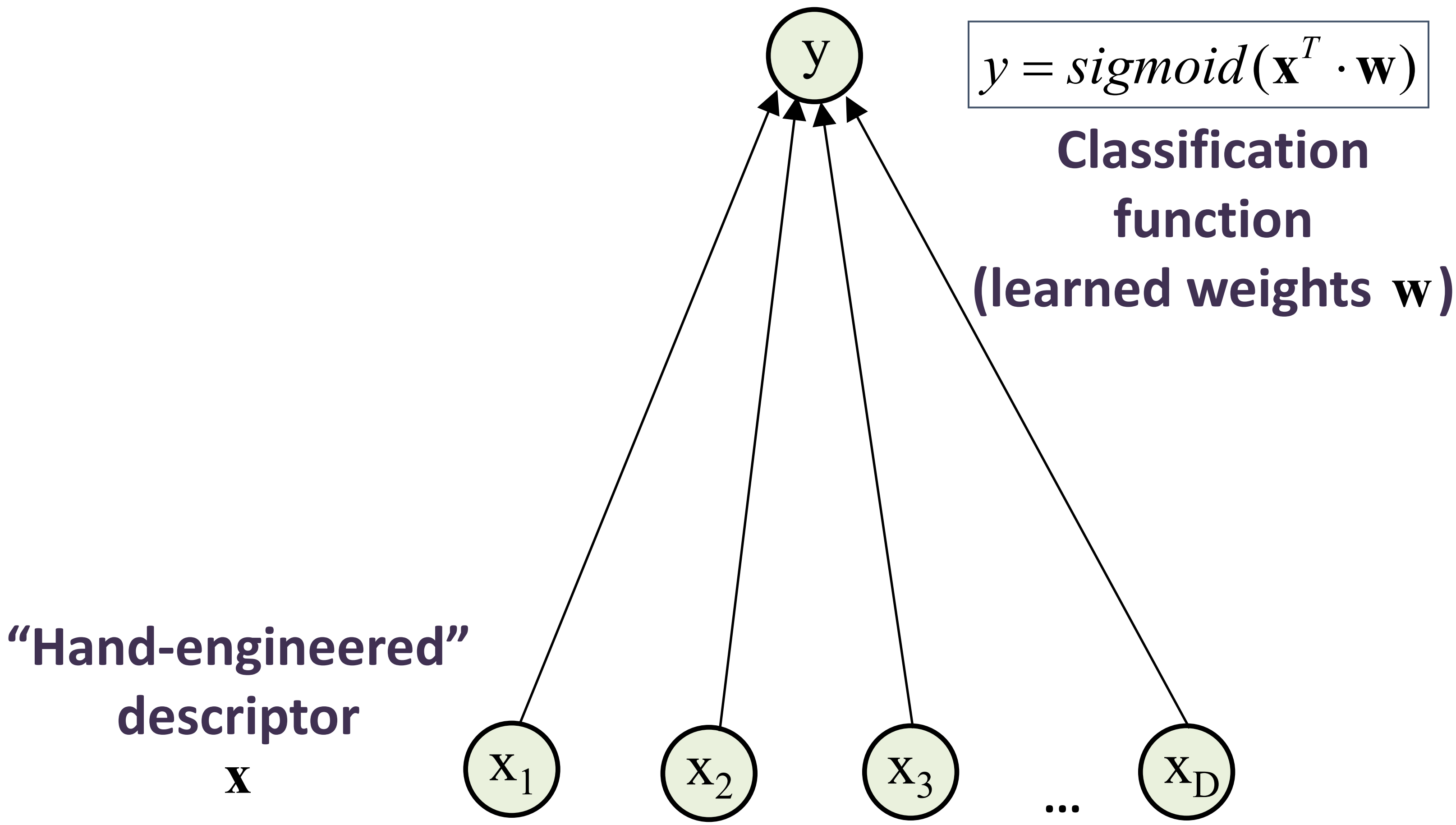



**Need descriptors that capture semantics, function…**

# From "shallow" mappings…

**Old-style approach:** output is a **direct function** of hand-engineered shape descriptors

$$y = sigmoid(\mathbf{x}^T \cdot \mathbf{w})$$

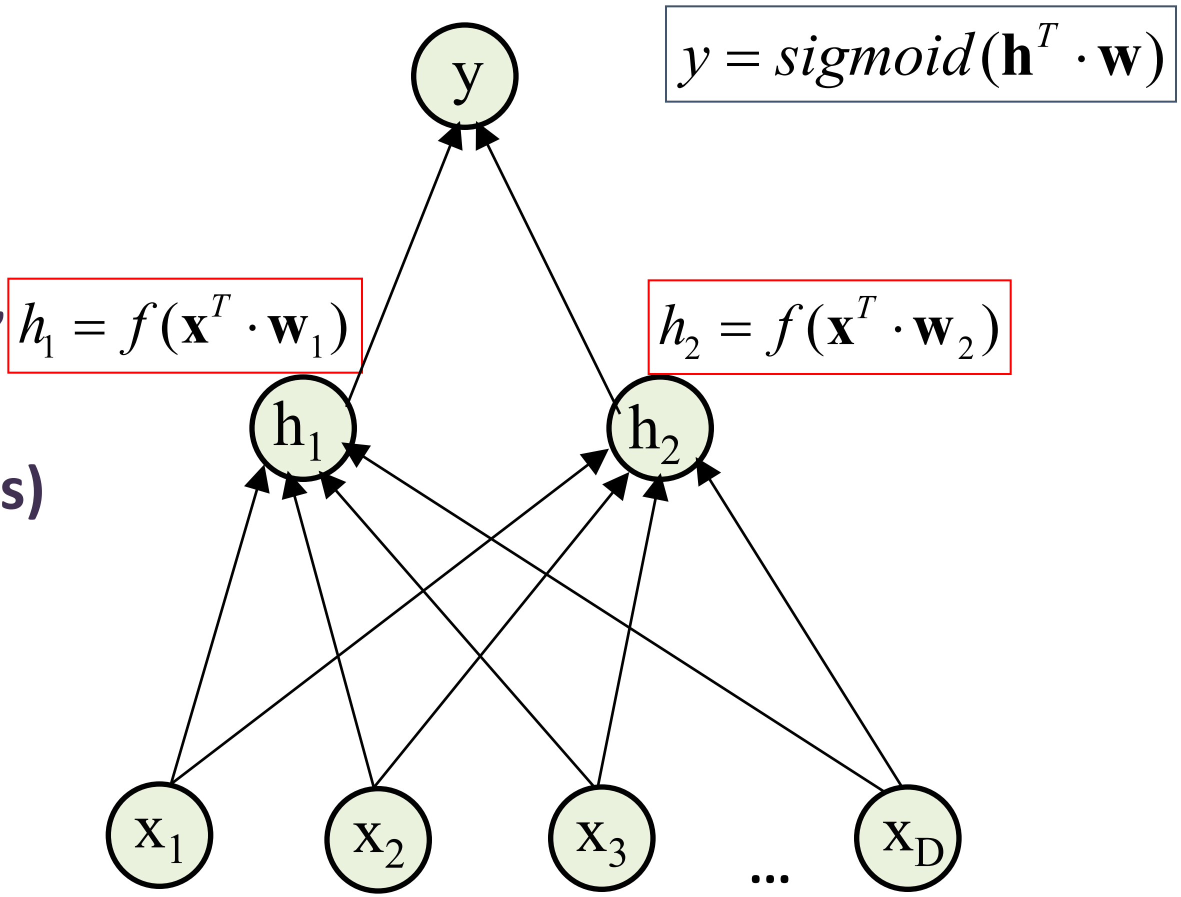


# … to neural nets

Introduce **intermediate learned functions** that yield optimized descriptors.

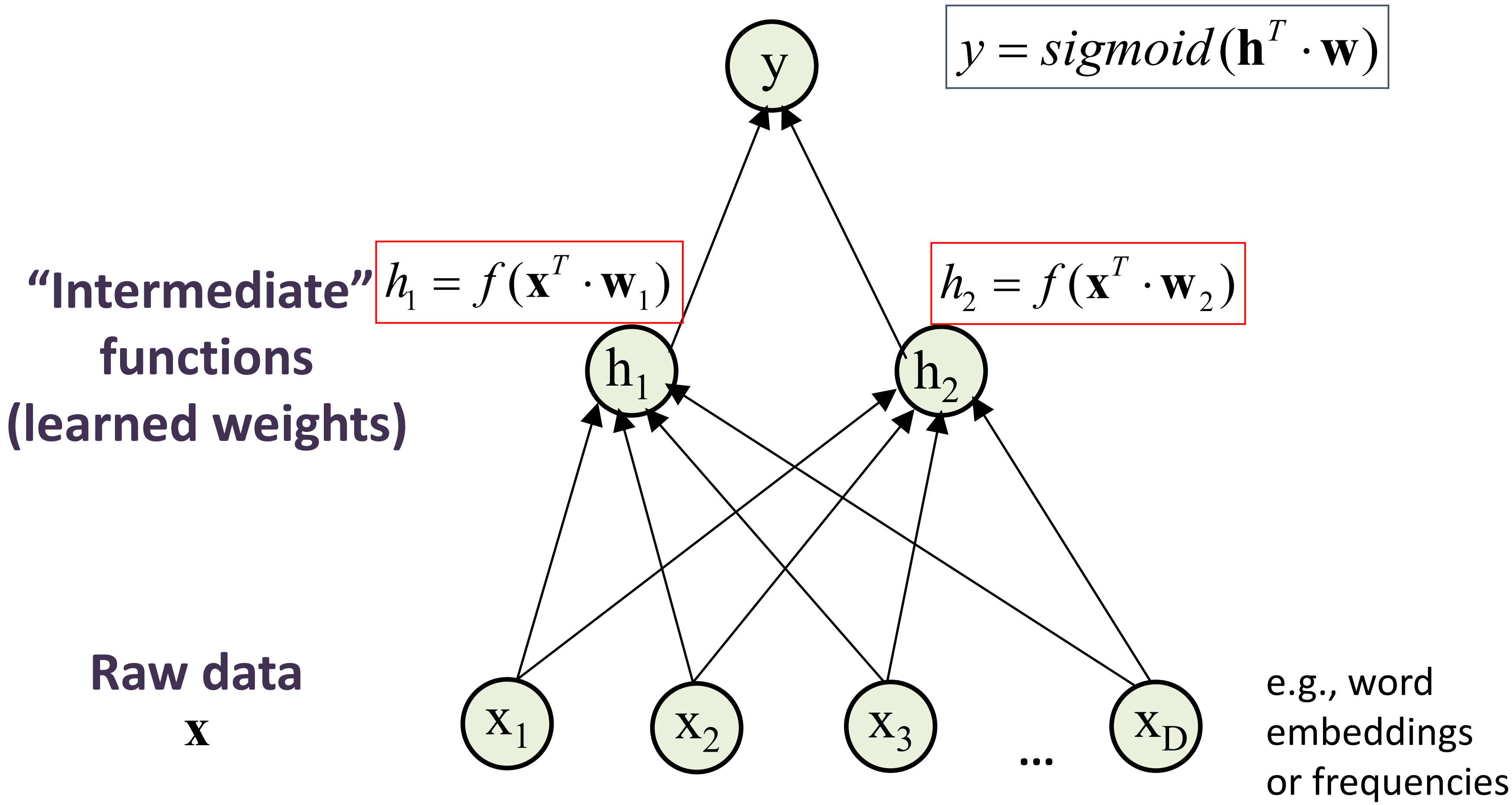
**"Intermediate" functions (learned weights)**

**Raw data**
$\mathbf{x}$



# … to neural nets

Introduce **intermediate learned functions** that yield optimized descriptors.

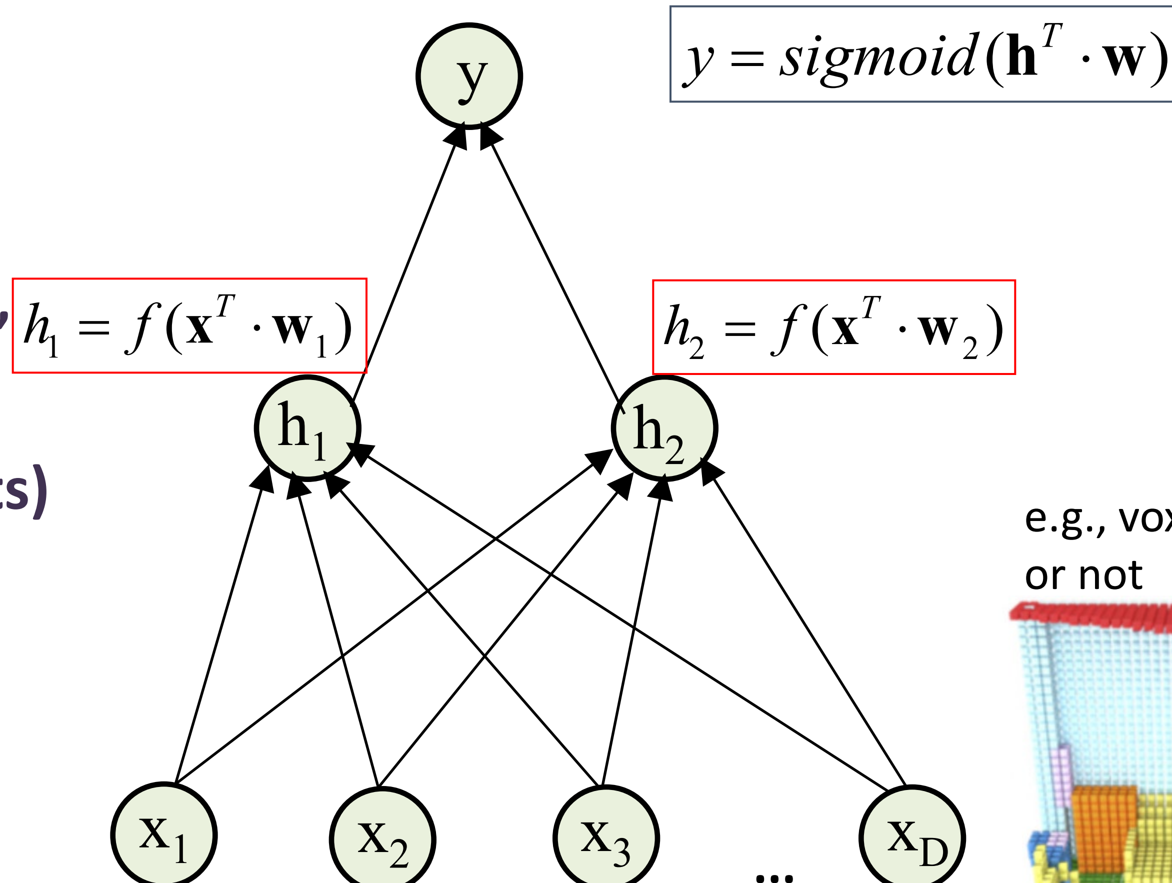


# … to neural nets

Introduce **intermediate learned functions** that yield optimized descriptors.

**"Intermediate" functions (learned weights)**

**Raw data $\mathbf{x}$**




e.g., voxels occupied or not