# Softmax regression



## Classification by minimising cross-entropy loss

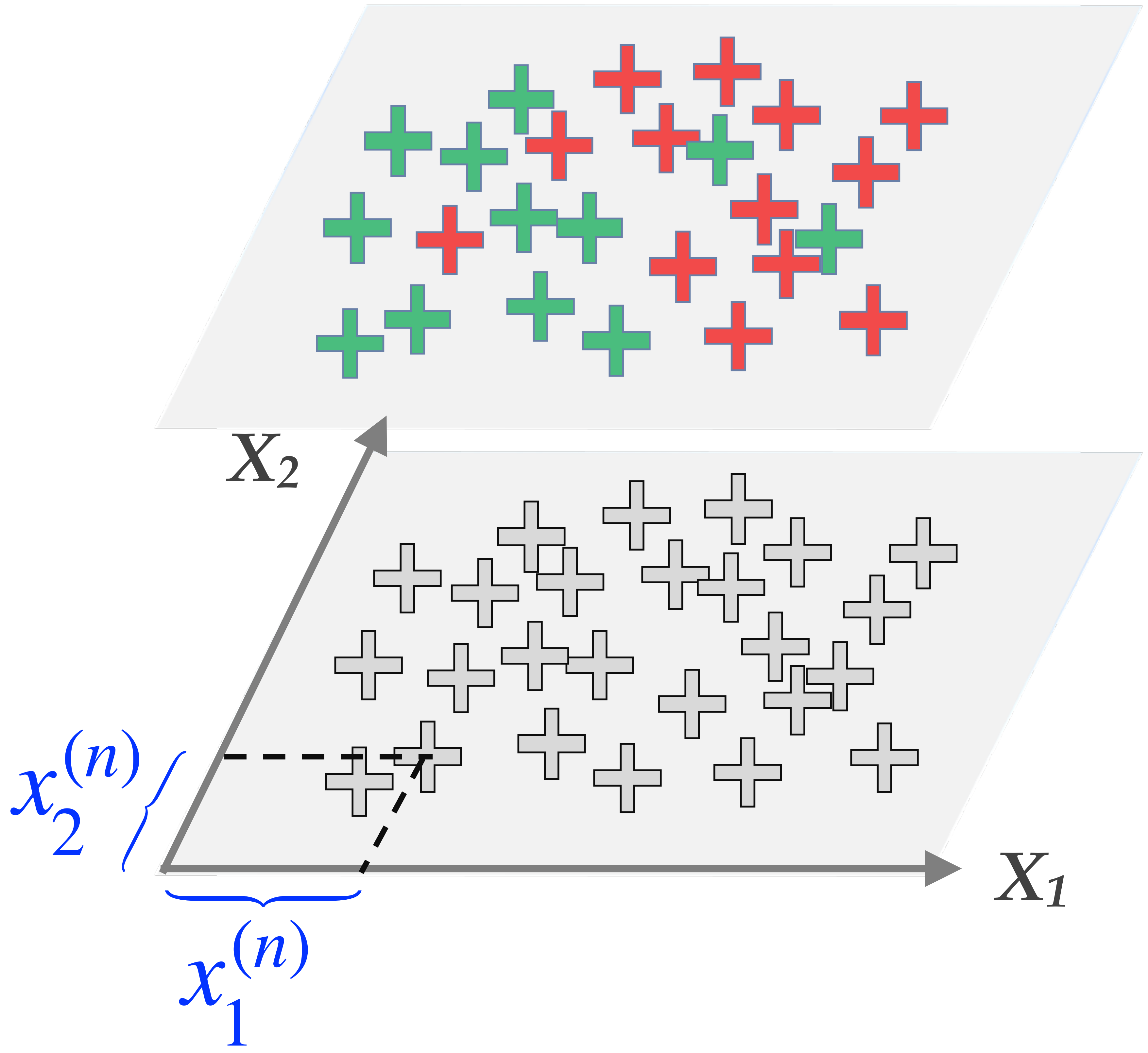
Srinandan Dasmahapatra

# Classification: discrete output

## Minimise deviation of prediction from annotation

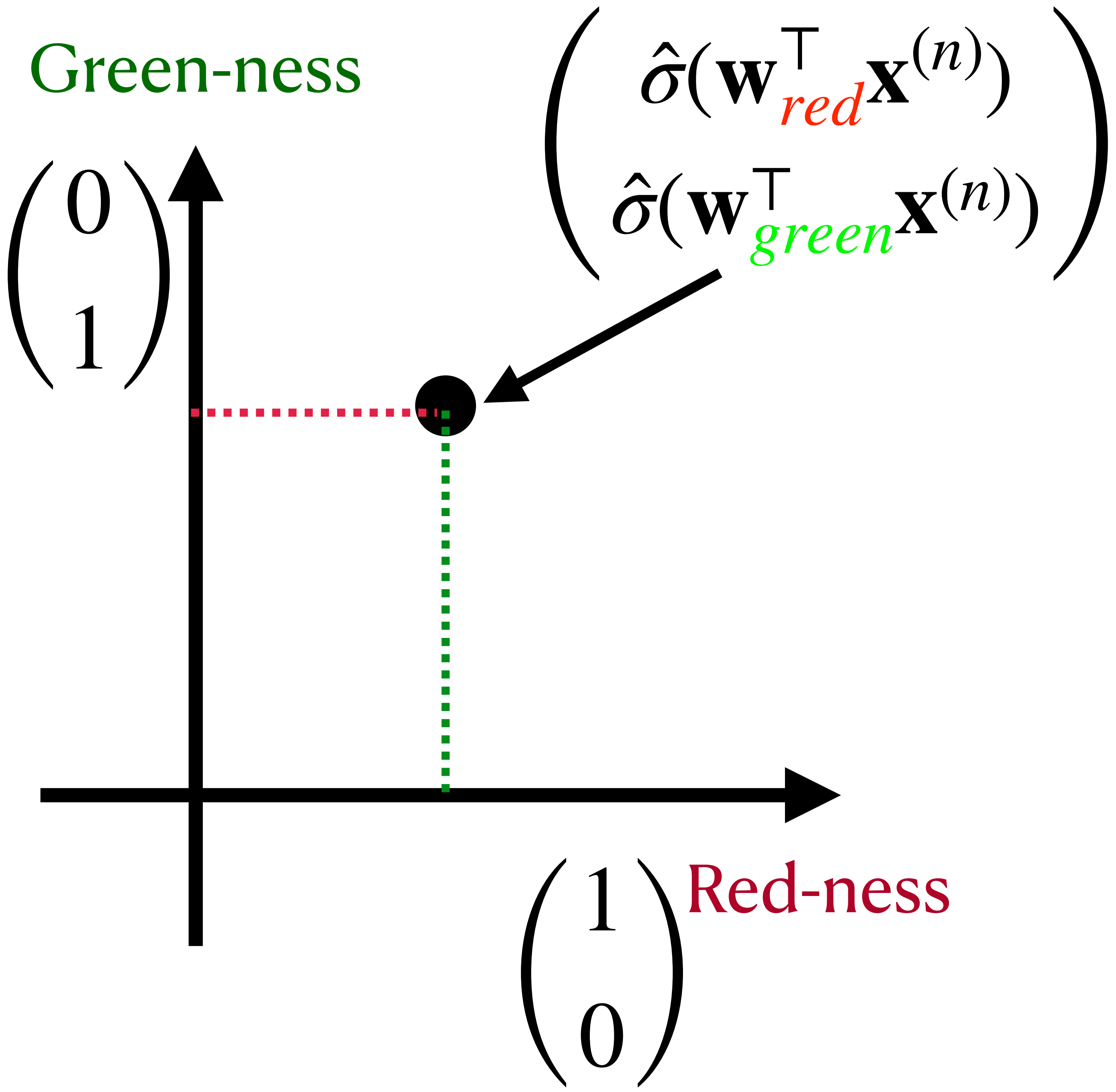
- Given training set represented by points labelled green and red, variable $Y$ ...
- ... where each point has two features
  $\mathscr{D} := \{((x_1{}^{(n)}, x_2{}^{(n)}), y^{(n)}) \,|\, n = 1,\ldots, N\}$
- Task: find function $f(x_1{}^{(n)}, x_2{}^{(n)}) = \hat{y}^{(n)}$ that reproduces given labels

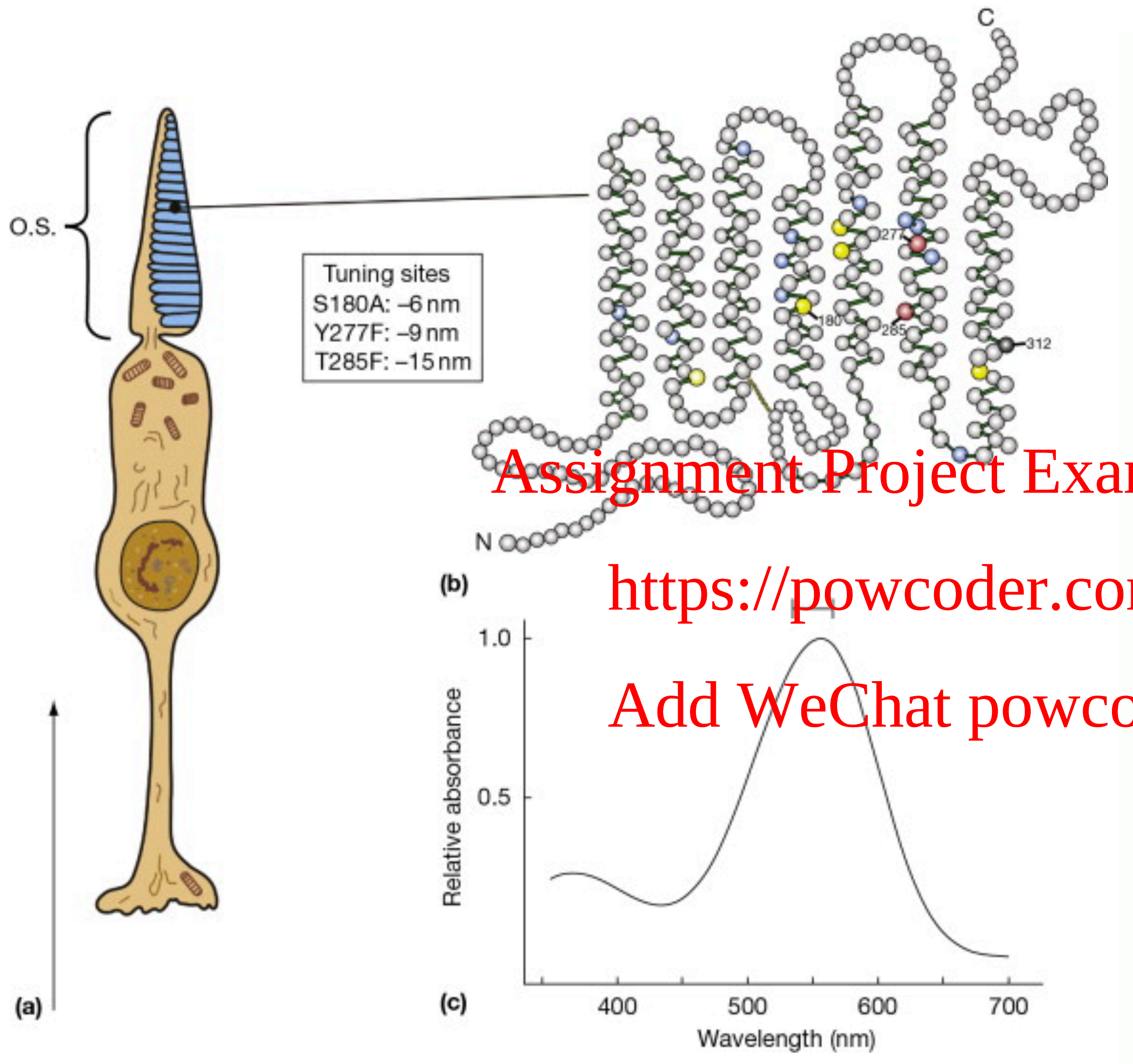


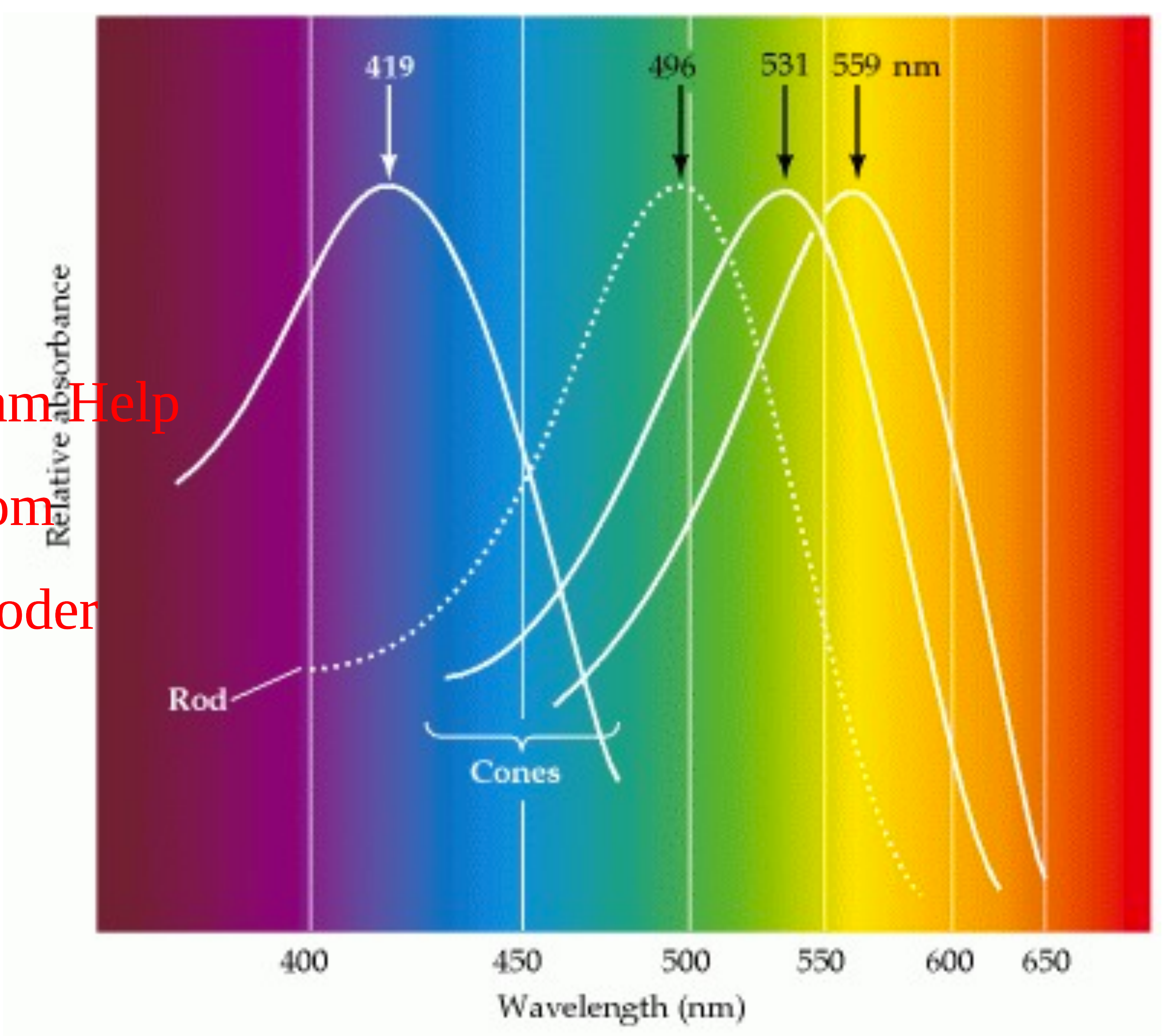


# Analogy with seeing in colour

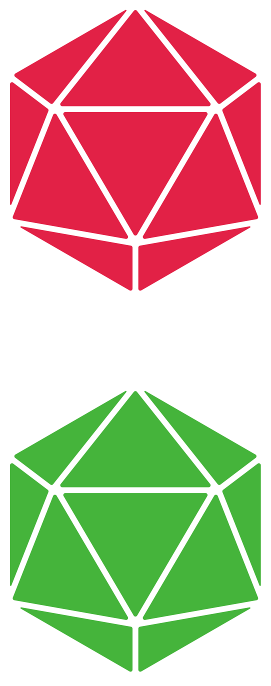
## Opsins (photopigments) in cones respond to colour preferentially




$$\begin{pmatrix} x_1^{(n)} \\ x_2^{(n)} \\ \vdots \\ x_d^{(n)} \end{pmatrix}$$

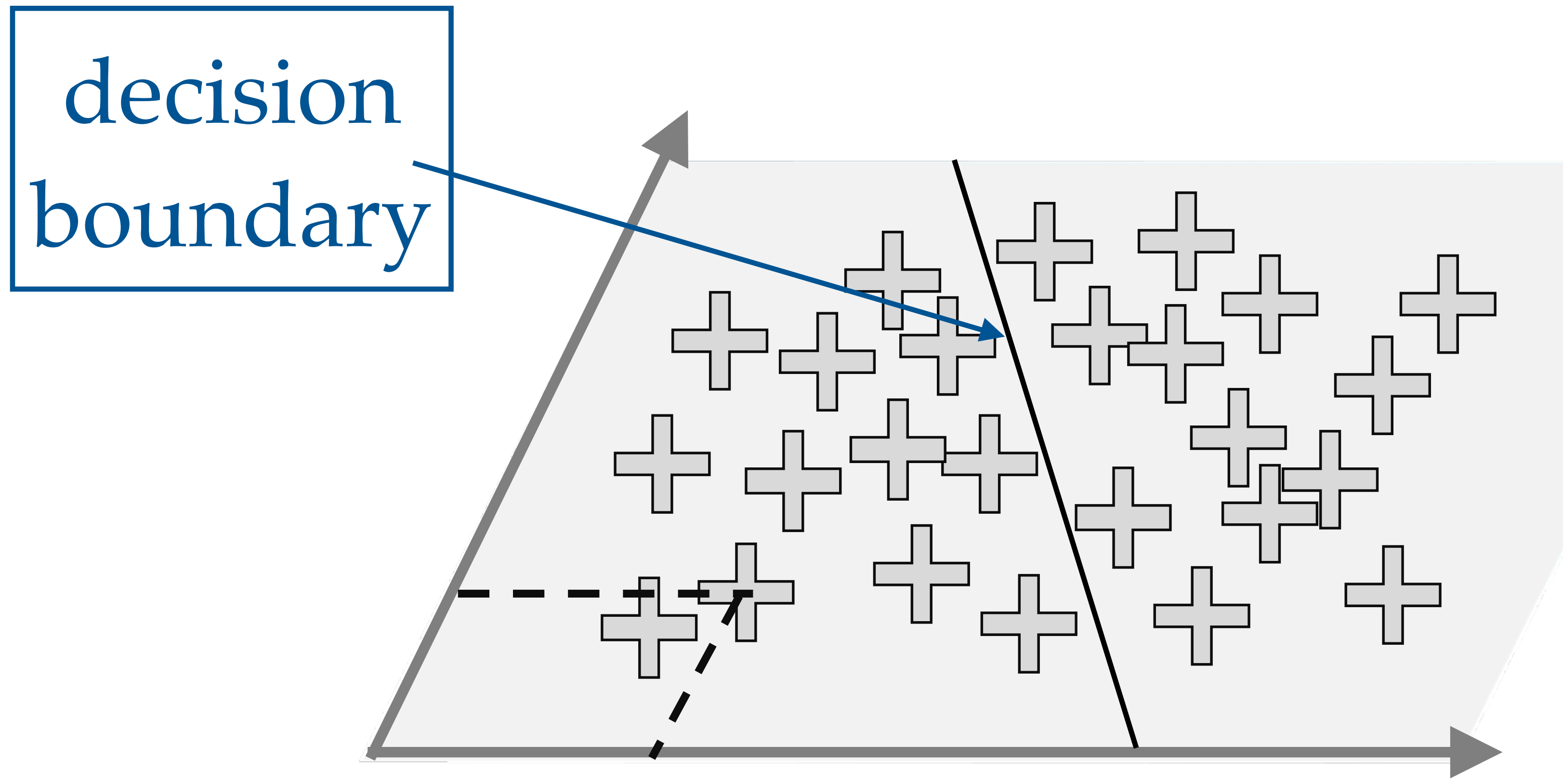


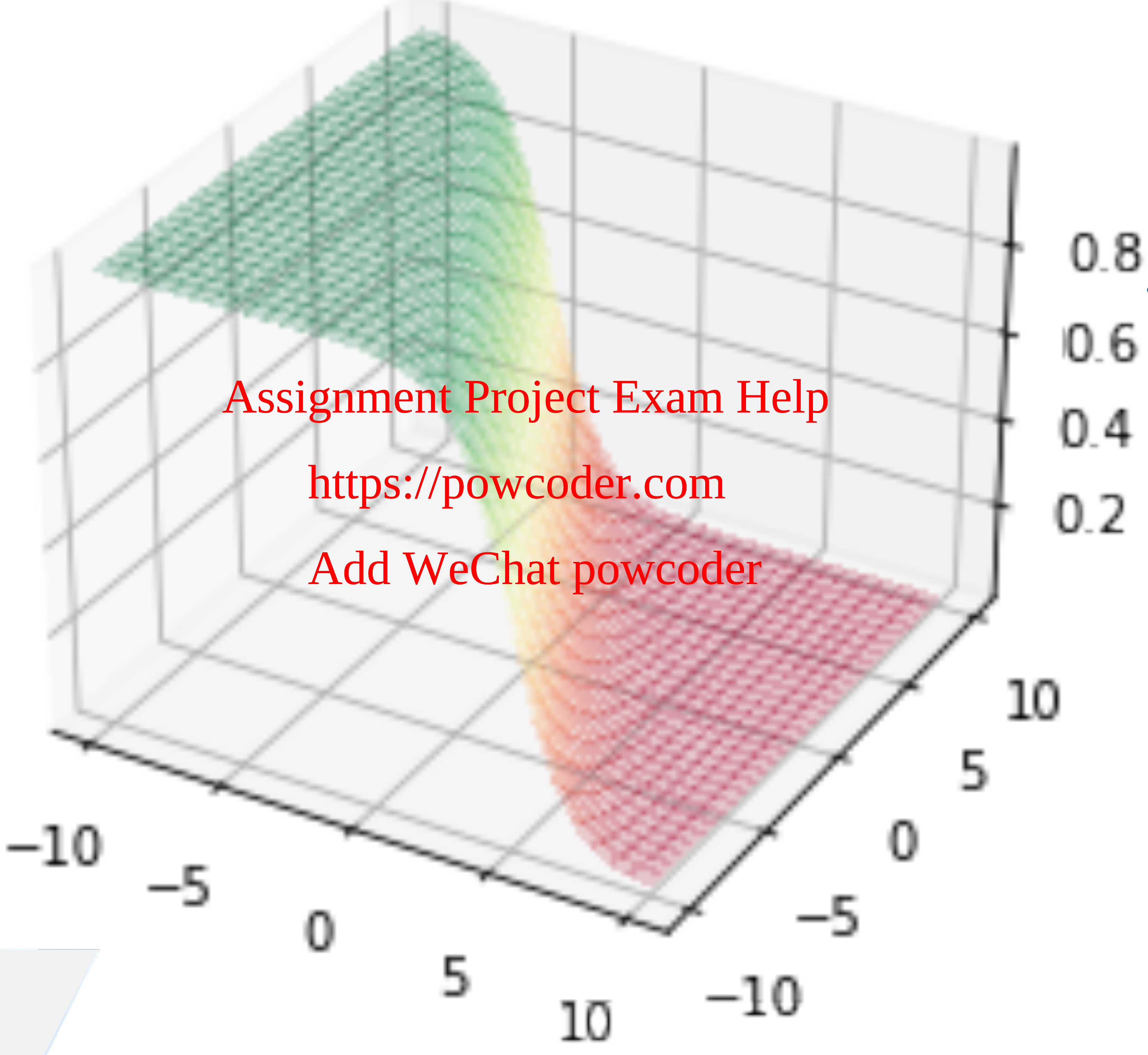


$$\mathbf{w}_{red} := (w_{red,1}, w_{red,2}, \ldots, w_{red,d})$$
$$\mathbf{w}_{green} := (w_{green,1}, w_{green,2}, \ldots, w_{green,d})$$

# Find equation for decision boundary

## Assign probability for each point being green/red

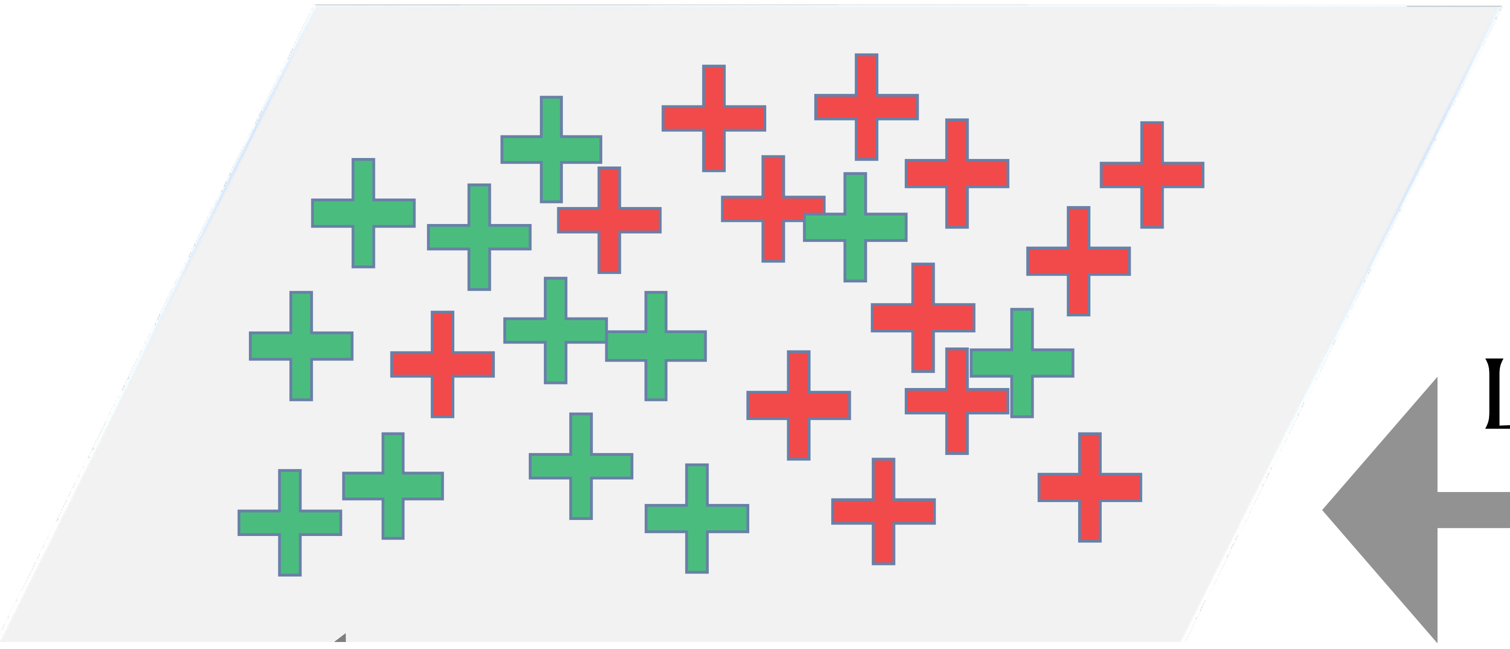





$$f(x_1, x_2; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2$$

$$\sigma(f) = \frac{1}{1 + \exp(-f)}$$



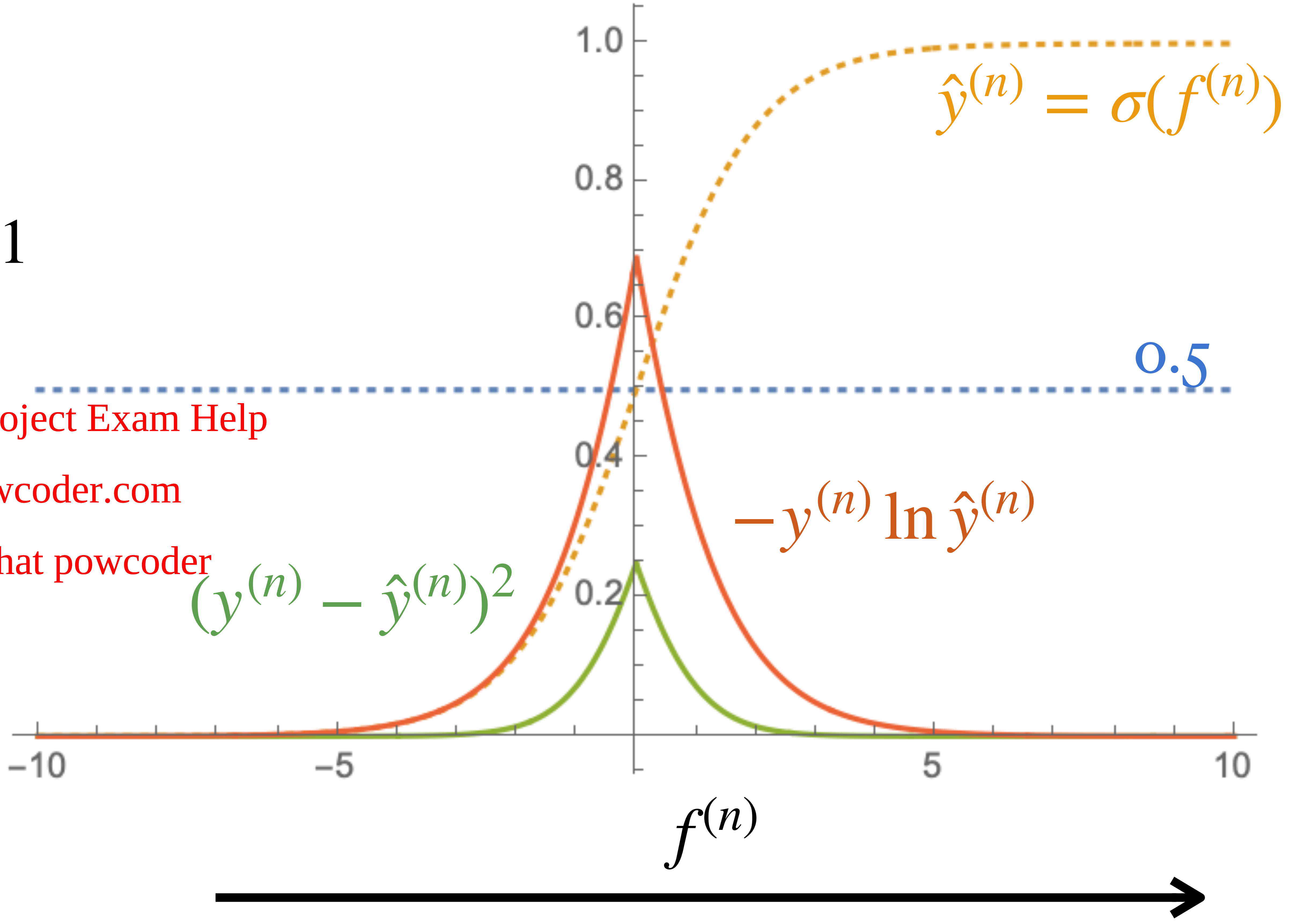
Learning = adjusting weights until agreement with data

$w_i$

# Constructing scale for comparing predictions with training labels

- Output $f(\mathbf{x}^{(n)}; \mathbf{w}^i) \triangleq f_i^{(n)}$, $i$ = red/green, $C = 2$,
- $0 \leq \sigma(f_i^{(n)}) \leq 1$ probability, with $\sum_{i=1}^{C} \sigma(f_i^{(n)}) = 1$
- Let $\hat{y}_i^{(n)} = \sigma(f_i^{(n)})$
- $y^{(n)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ or $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ for red/green
- Evaluation of classification: cost( $y^{(n)}, \hat{y}^{(n)}$)
- Compare two different costs — quadratic and logarithmic
- Logarithm penalises mistakes more, also has a sharper drop (large gradient to guide weights to lower loss)



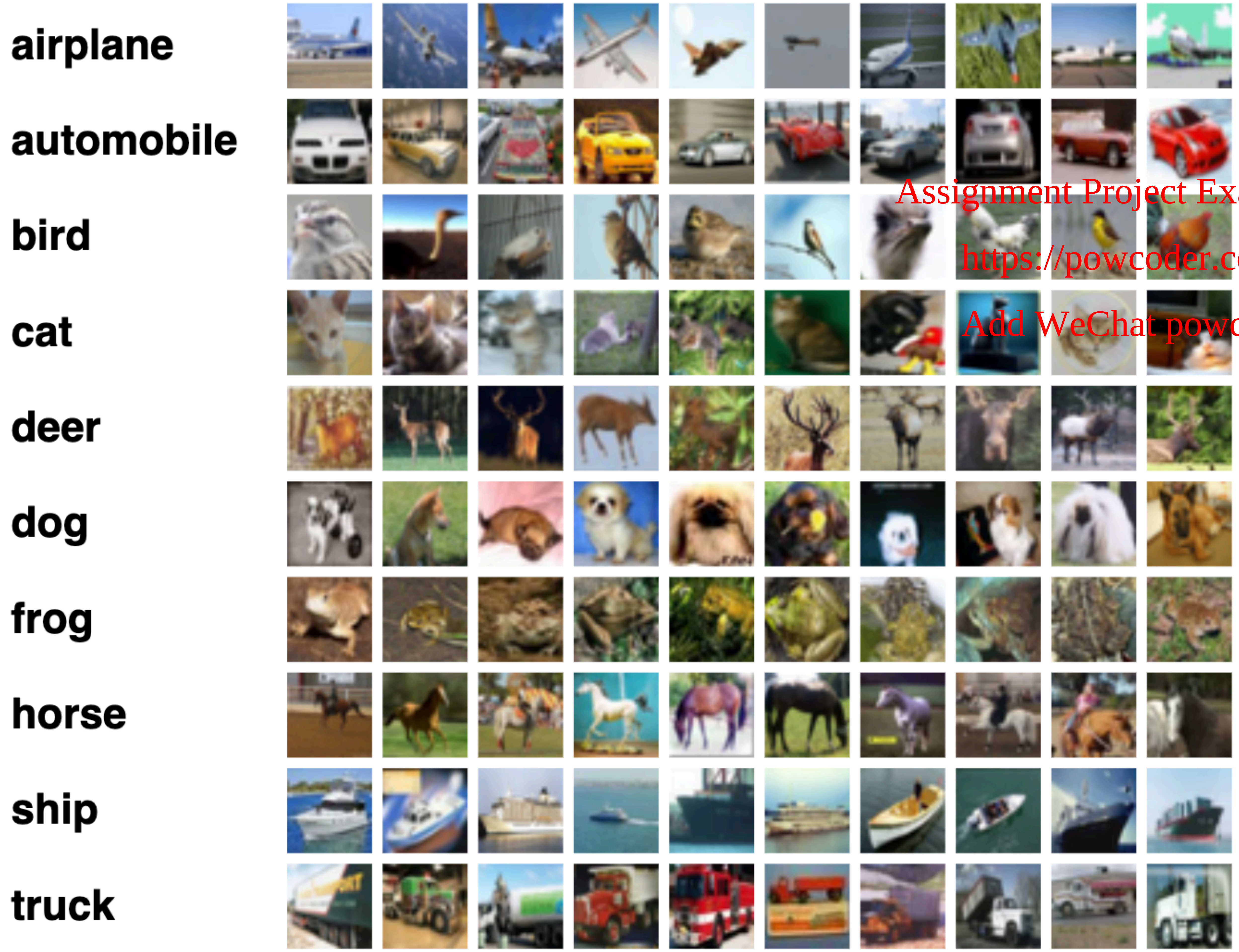



For a one component (scalar) output

# Multiclass classification

## Input: images 32 x 32 x 3 dimensions, Output: one-hot encodings: 10 dimensions

Here are the classes in the dataset, as well as 10 random images from each:

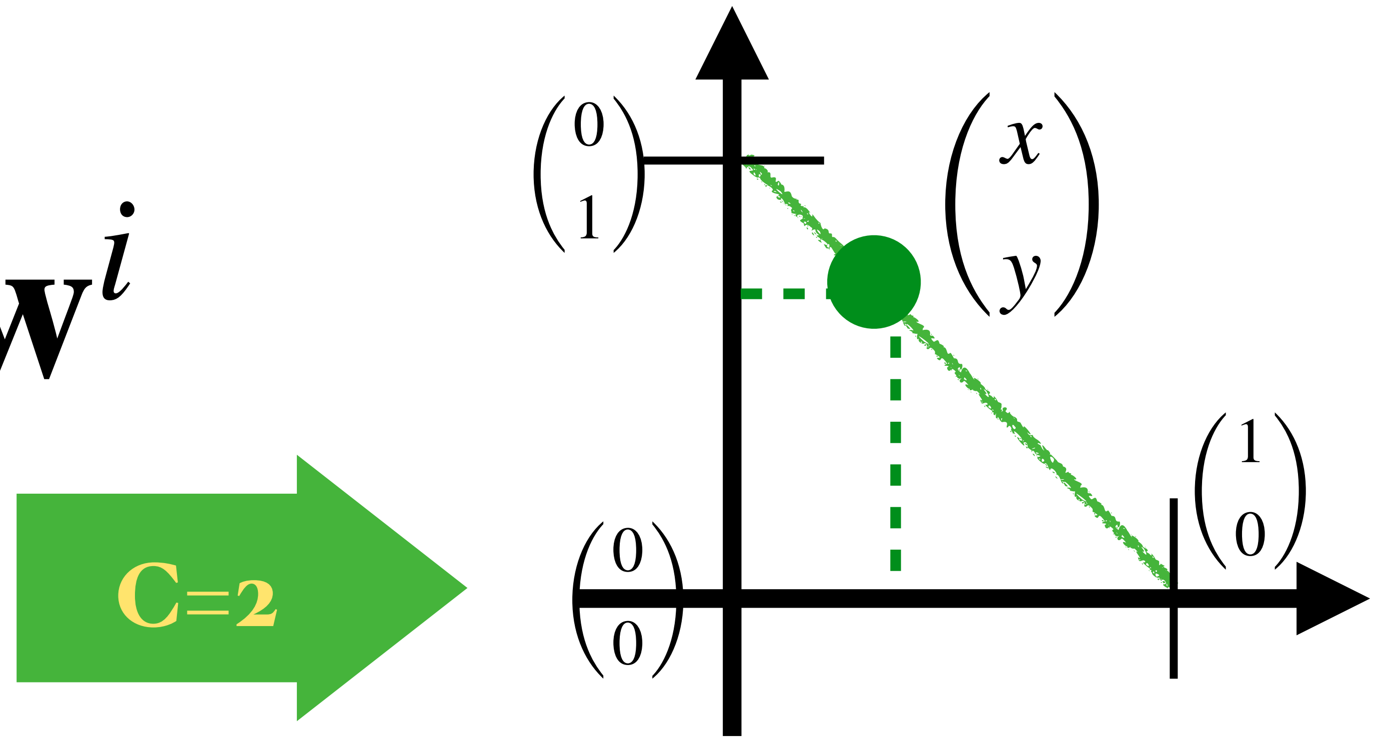



CIFAR-10: Example dataset for multi class classification



$$\text{CAT} \mapsto \mathbf{e}_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \text{SHIP} \mapsto \mathbf{e}_9 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

# C-classes C different weight vectors $\mathbf{w}^i$



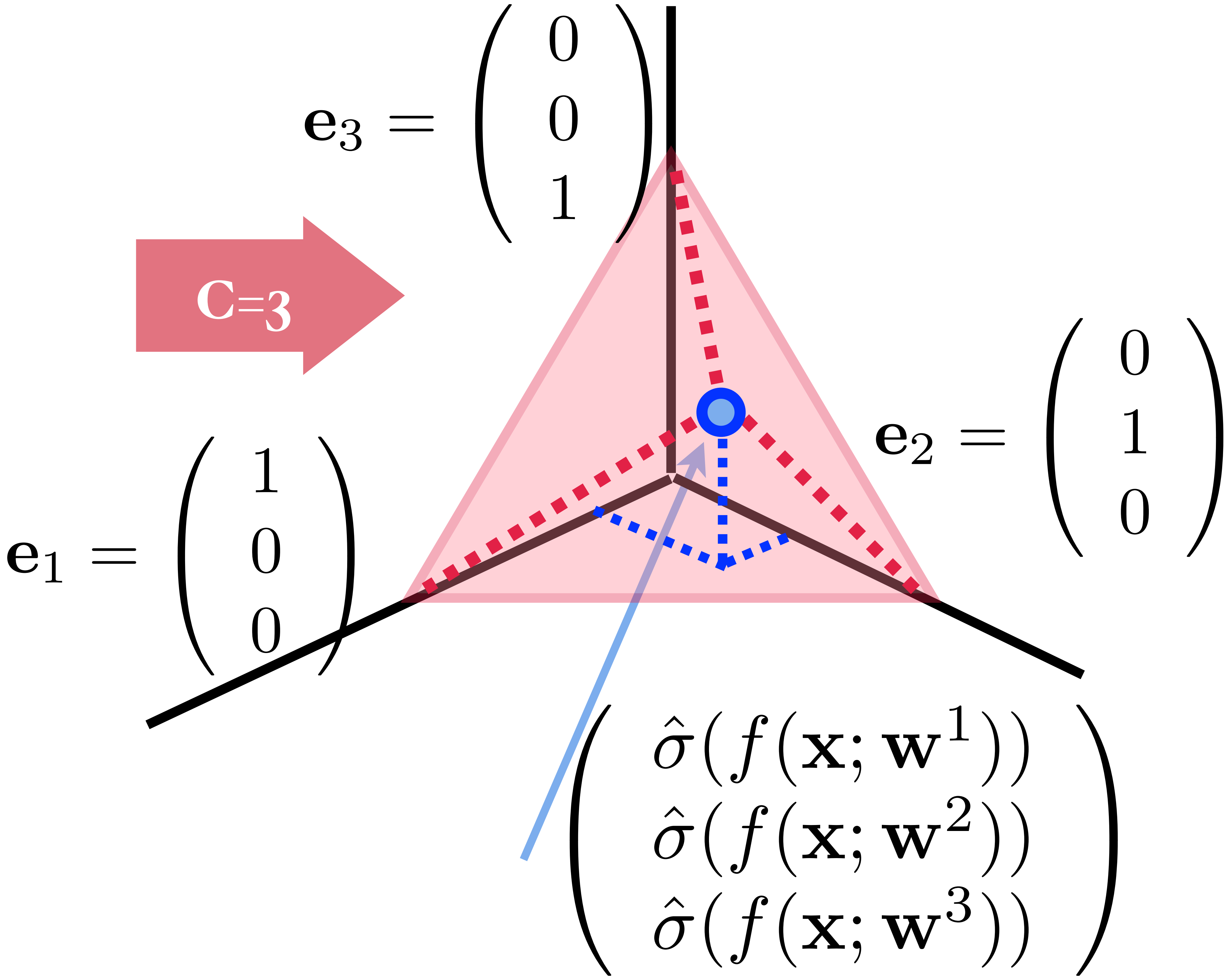

$x \geq 0, y \geq 0, x + y = 1$

- For each input vector (say a representation of an image or sound file), produce an output on the (C-1)-dimensional surface embedded in C-dimensional Euclidean space



- Cost for input $\mathbf{x}^{(n)}$ = measure of mismatch between C-dimensional prediction $\hat{\sigma}(f(\mathbf{x}^{(n)}; \mathbf{w}^i))$ and true label $\mathbf{e}_i$

- Hat $\hat{\cdot}$ on $\hat{\sigma}$ indicates normalisation: entries add up to one: $\hat{\sigma}(f(\mathbf{x}; \mathbf{w}^1)) + \hat{\sigma}(f(\mathbf{x}; \mathbf{w}^2)) + \hat{\sigma}(f(\mathbf{x}; \mathbf{w}^3)) = 1$



# Multiclass classification

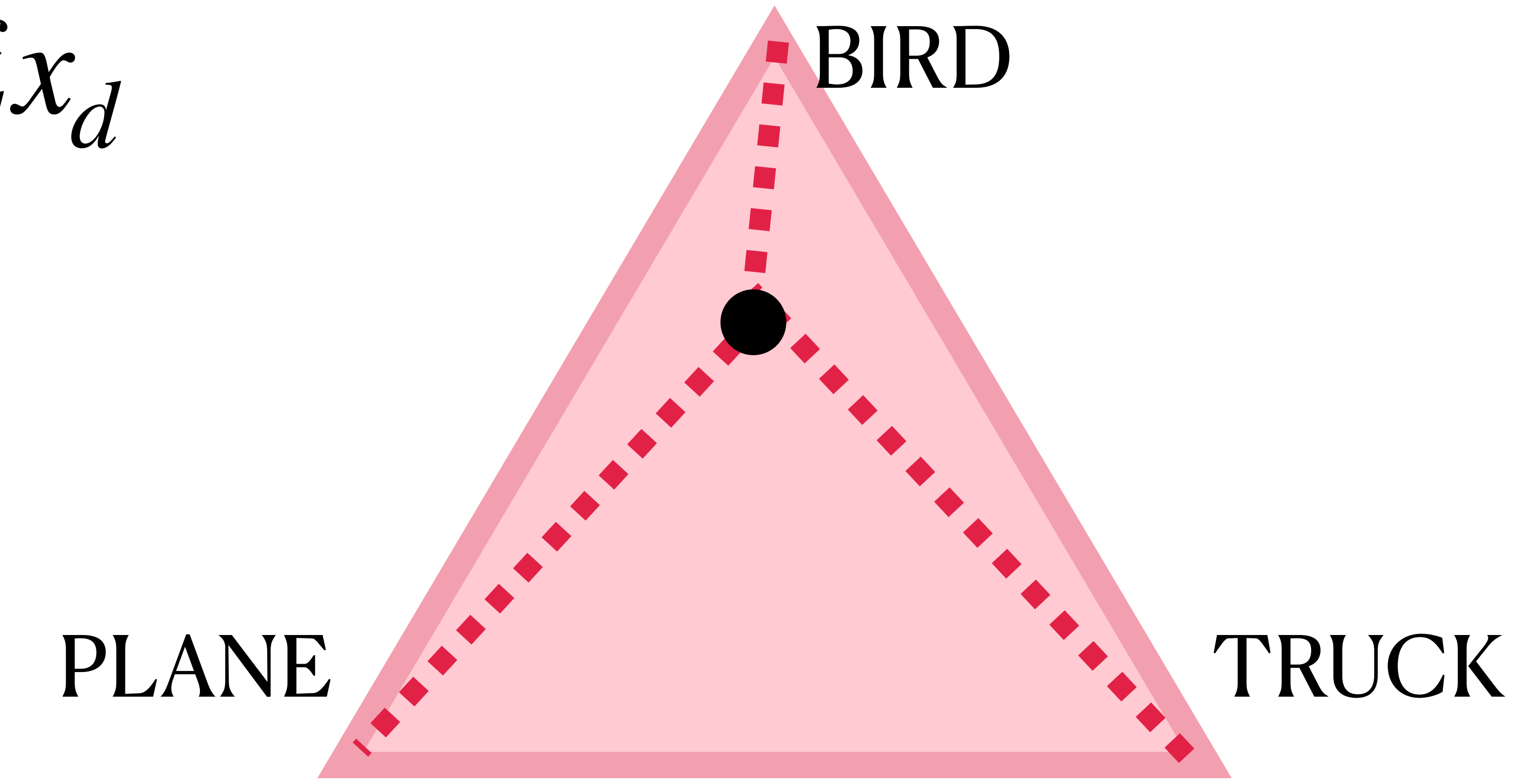
## Weight vectors for each class

- $\mathbf{w}^c = (w_0^c, w_1^c, \ldots, w_d^c), c = 1, \ldots, C$.
- $C$ - number of classes, 10 for CIFAR-10
- $d$ - dimensionality of data, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$
- $f(\mathbf{x}; \mathbf{w}^c) = w_0^c \cdot 1 + w_1^c x_1 + \cdot + w_d^c x_d$ : for each input data point, compute output for all classes

$$\overset{d+1}{\longleftrightarrow}$$

$$C \updownarrow \begin{pmatrix} w_0^{\text{PLANE}} & w_1^{\text{PLANE}} & \cdots & w_d^{\text{PLANE}} \\ \vdots & \vdots & \ddots & \vdots \\ w_0^{\text{BIRD}} & w_1^{\text{BIRD}} & \cdots & w_d^{\text{BIRD}} \\ \vdots & \vdots & \ddots & \vdots \\ w_0^{\text{TRUCK}} & w_1^{\text{TRUCK}} & \cdots & w_d^{\text{TRUCK}} \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \updownarrow d+1$$






$$\begin{pmatrix} \hat{\sigma}(f(\mathbf{x}; \mathbf{w}^{\text{PLANE}})) \\ \hat{\sigma}(f(\mathbf{x}; \mathbf{w}^{\text{BIRD}})) \\ \hat{\sigma}(f(\mathbf{x}; \mathbf{w}^{\text{TRUCK}})) \end{pmatrix}$$

# Set up gradient descent of loss for classification

## Re-phrasing what has been done

- For each class each data point $\mathbf{x}^{(n)}$ is assigned a score $s_c^{(n)} = f(\mathbf{x}^{(n)}; \mathbf{w}^c), c = 1,\ldots, C$
- Choose the largest of the $C$ scores as the predicted class for $\mathbf{x}^{(n)}$



  - $c^* = \arg \max_{c \in \{1,\ldots,C\}} s_c^{(n)}$

- Replace max by softmax: $\max(s_1, s_2, s_3) \longrightarrow \text{softmax}(s_1, s_2, s_3) = \ln(e^{s_1} + e^{s_2} + e^{s_3})$
- Exponential function: monotonic in argument ($x \nearrow \implies e^x \nearrow$)
- Normalise exponential scores: $s_c^{(n)} \mapsto \dfrac{e^{s_c^{(n)}}}{e^{s_1^{(n)}} + e^{s_2^{(n)}} + \cdots + e^{s_C^{(n)}}} =: \hat{\sigma}(s_c^{(n)}) = [\hat{y}^{(n)}]_c$
- Treat component $c$ of $[\hat{y}^{(n)}]_c = \hat{\sigma}(s_c^{(n)})$ as probability that $\mathbf{x}^{(n)}$ belongs to class $c : P(c \,|\, \mathbf{x}^{(n)})$

# Multi-class loss function: cross entropy

## Measures information about label distribution from input data and choice of weights

- For each data point $\mathbf{x}^{(n)}$ sum over costs $-\sum_{c=1}^{C} y_c^{(n)} \ln \hat{y}_c^{(n)}$ for all classes
- Sum costs over all data points $L(\mathbf{W}) := L(\{\mathbf{w}^1, \ldots, \mathbf{w}^C\}) = -\sum_{n=1}^{N}\sum_{c=1}^{C} y_c^{(n)} \ln \hat{y}_c^{(n)}$, called cross-entropy.



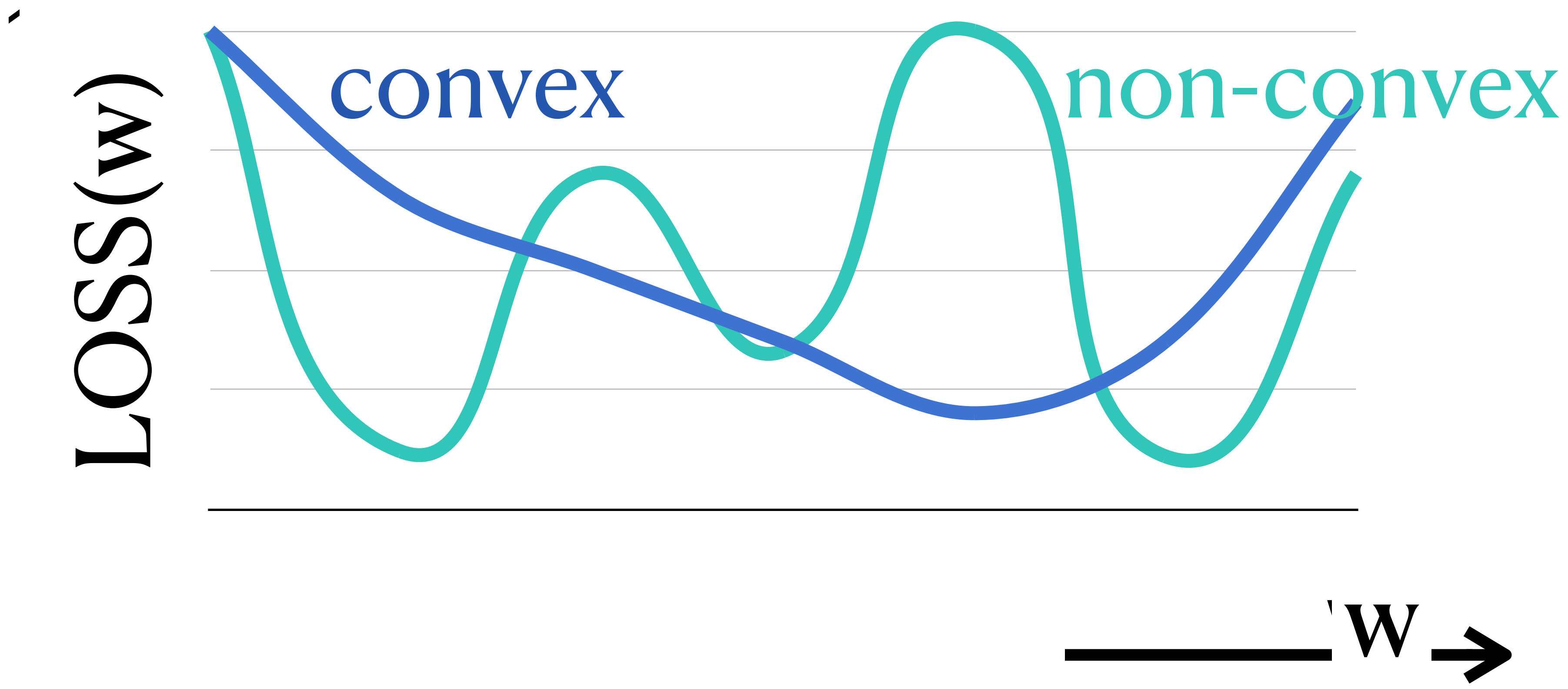
- **Eg**: target $y^{(n)} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ prediction $\hat{y}^{(n)} = \begin{pmatrix} \hat{y}_1^{(n)} \\ \hat{y}_2^{(n)} \\ \hat{y}_3^{(n)} \\ \hat{y}_4^{(n)} \end{pmatrix}$: $-(0 \cdot \ln \hat{y}_1^{(n)} + 1 \cdot \ln \hat{y}_2^{(n)} + 0 \cdot \ln \hat{y}_3^{(n)} + 0 \cdot \ln \hat{y}_4^{(n)}) = -\ln \hat{y}_2^{(n)}$
- $L(\mathbf{W}) = -\ln\left(\hat{y}_{c_1}^{(1)} \cdot \hat{y}_{c_2}^{(2)} \cdots \hat{y}_{c_N}^{(N)}\right) = -\sum_{n=1}^{N} \ln \hat{y}_{c_n}^{(n)}$ : reduce negative of log(predicted probabilities)

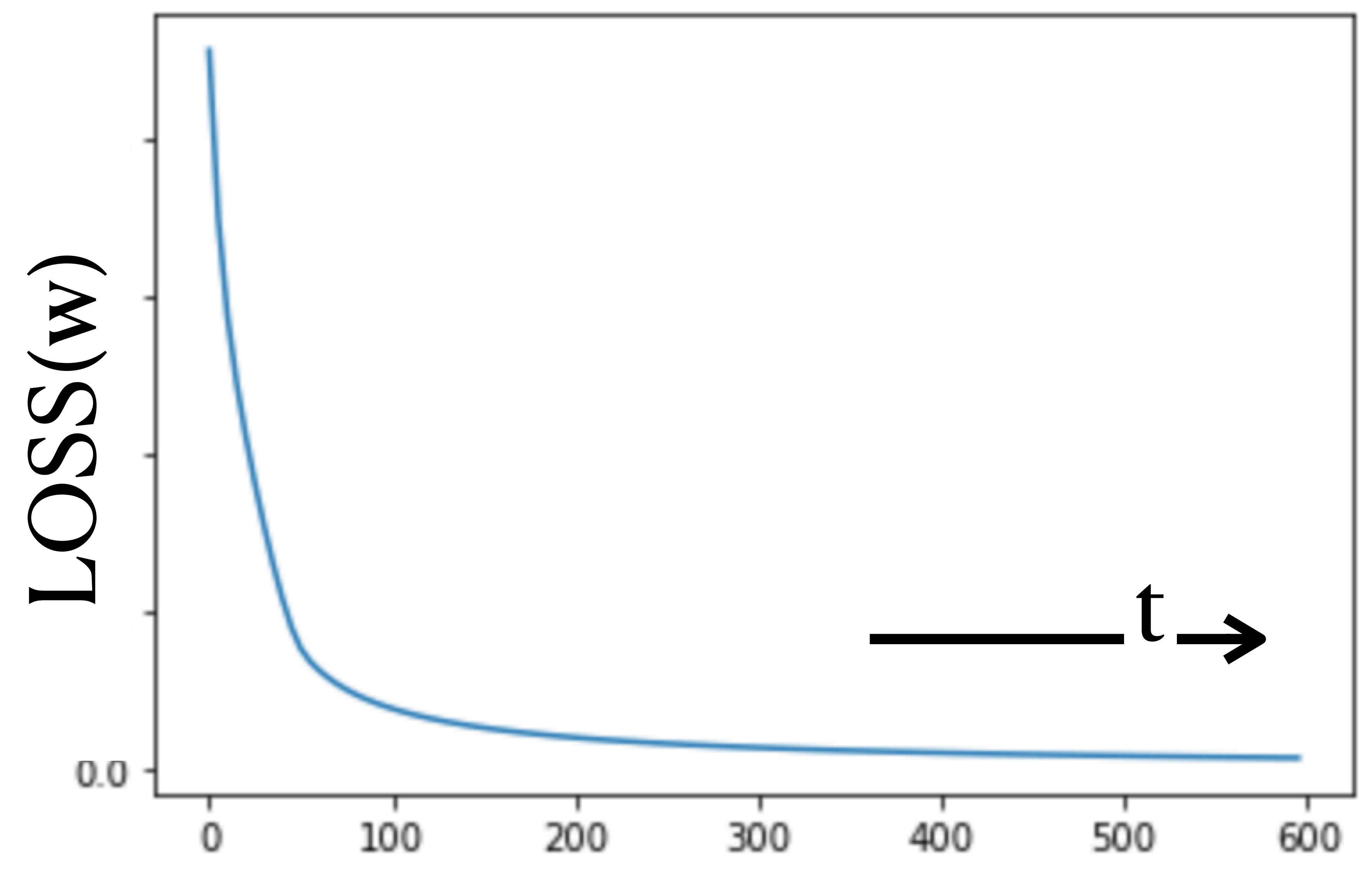
# Gradient descent on cross-entropy finds optimal weights

## For linear maps *f*, cross-entropy is convex

- Learning: Reduce $L(\mathbf{W})$ by changing weights
- $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} L(\mathbf{W})$
- **All** weights are contained in $\mathbf{w}$
- Jupyter notebook





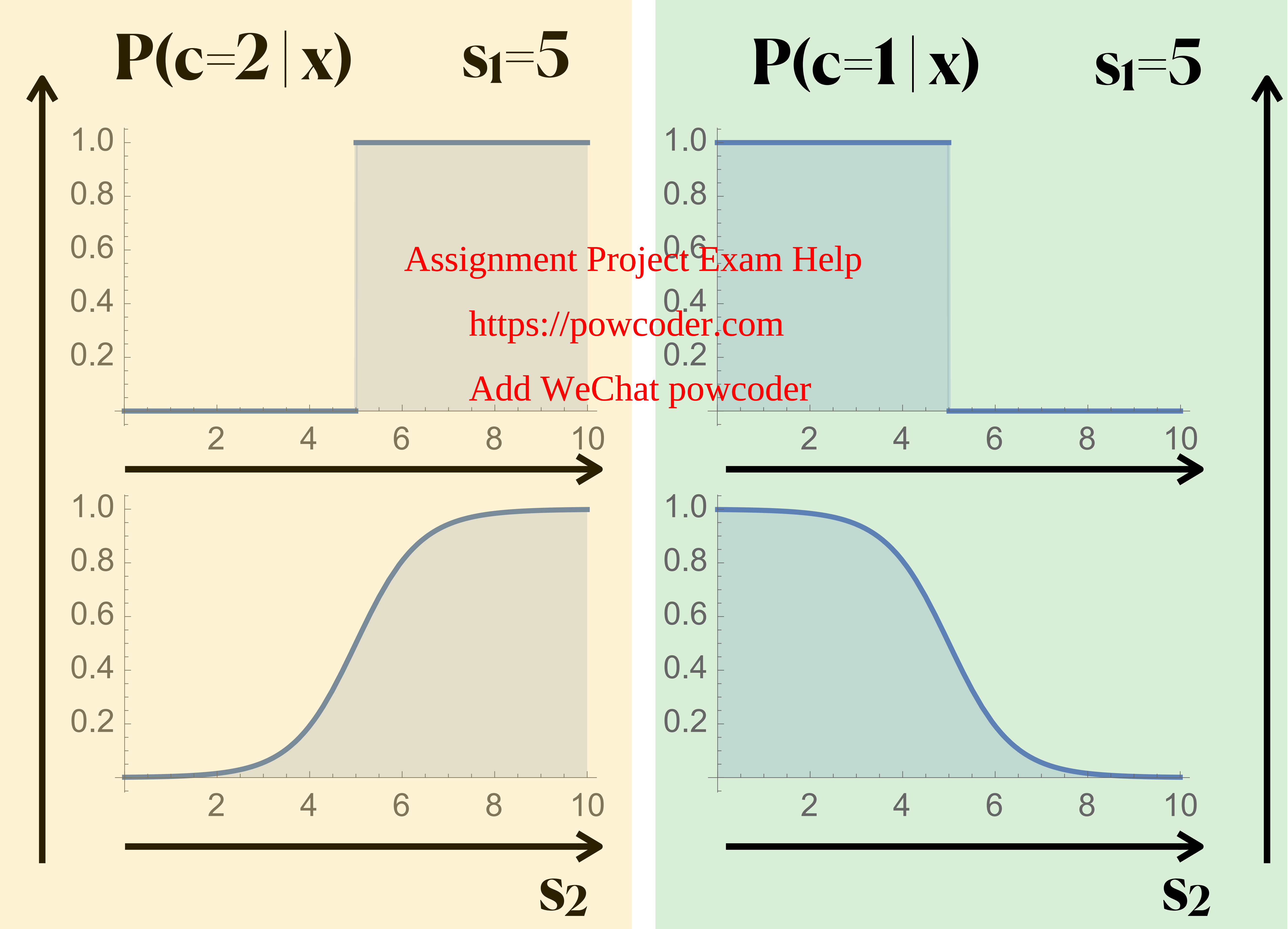
# Example: data x; 2-class problem

## Compare probability assignment for arg max with arg softmax

$$\frac{f[0,(s-5)]}{f[0,(s-5)]+f[0,-(s-5)]}$$

- f=Max

- f=Softmax



# Lab 2