

# Logistic Regression

Machine Learning Course - CS-433

9 Oct 2025

Robert West

(Slide credits: Nicolas Flammarion)

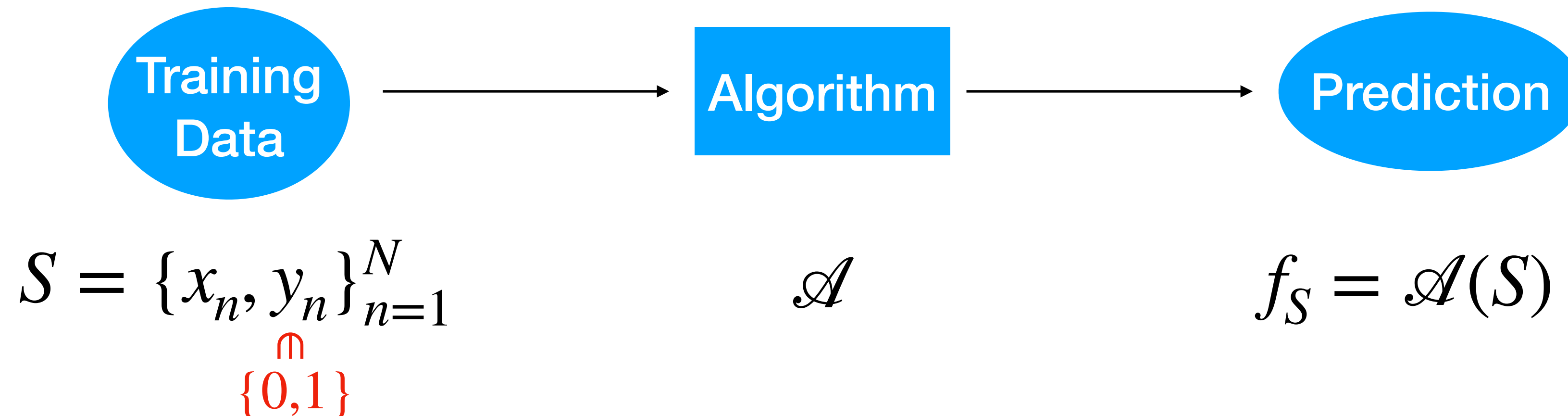


# Binary classification

We observe some data  $S = \{x_n, y_n\}_{n=1}^N \in \mathcal{X} \times \{0,1\}$

Goal: given a new observation  $x$ , we want to predict its label  $y$

How:

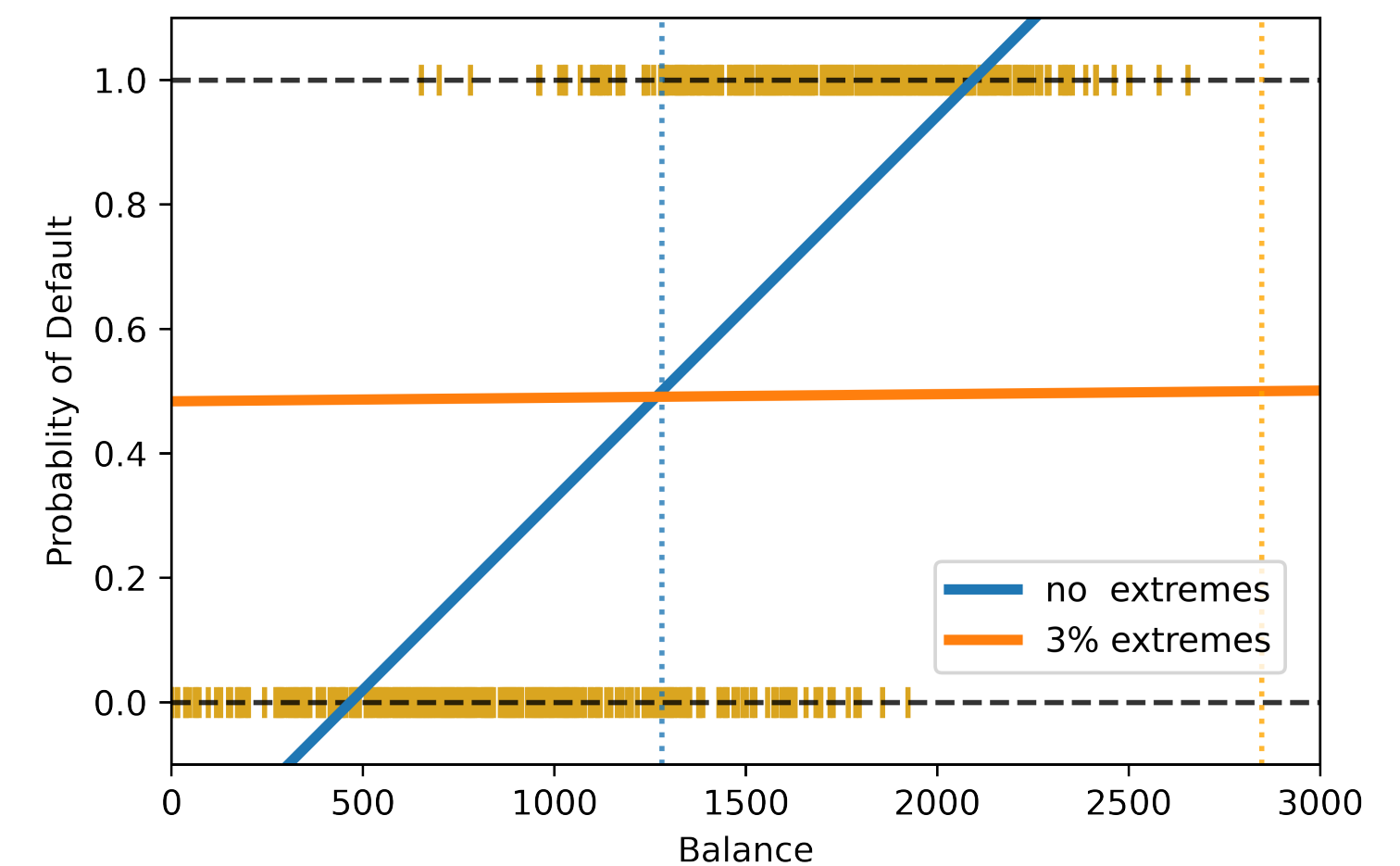


# Motivation for logistic regression

Instead of directly modeling the output  $Y$ , we can **model the probability** that  $Y$  belongs to a specific class. How?

In the previous lecture, we used a linear regression model  $\mathbb{P}(Y = 1 | X = x) = x^\top w + w_0$  but

- The predicted value is not in  $[0,1]$
- Very large or small prediction values contribute to the error even when they suggest high confidence in the resulting classification



**Solution:** map the prediction from  $(-\infty, +\infty)$  to  $[0,1]$

# From probs to log odds

Trick: don't deal with probabilities  $\in [0,1]$ , but with logs odds  $\in ]-\infty, \infty[$

Probability:  $y \in [0,1]$

$\Leftrightarrow$  odds:  $\frac{y}{1-y} \in [0, \infty[$

$\Leftrightarrow$  log odds:  $\log \frac{y}{1-y} \in ]-\infty, \infty[$

Model log odds as linear function of weights:  $\log \frac{y}{1-y} = x^T w + w_0$

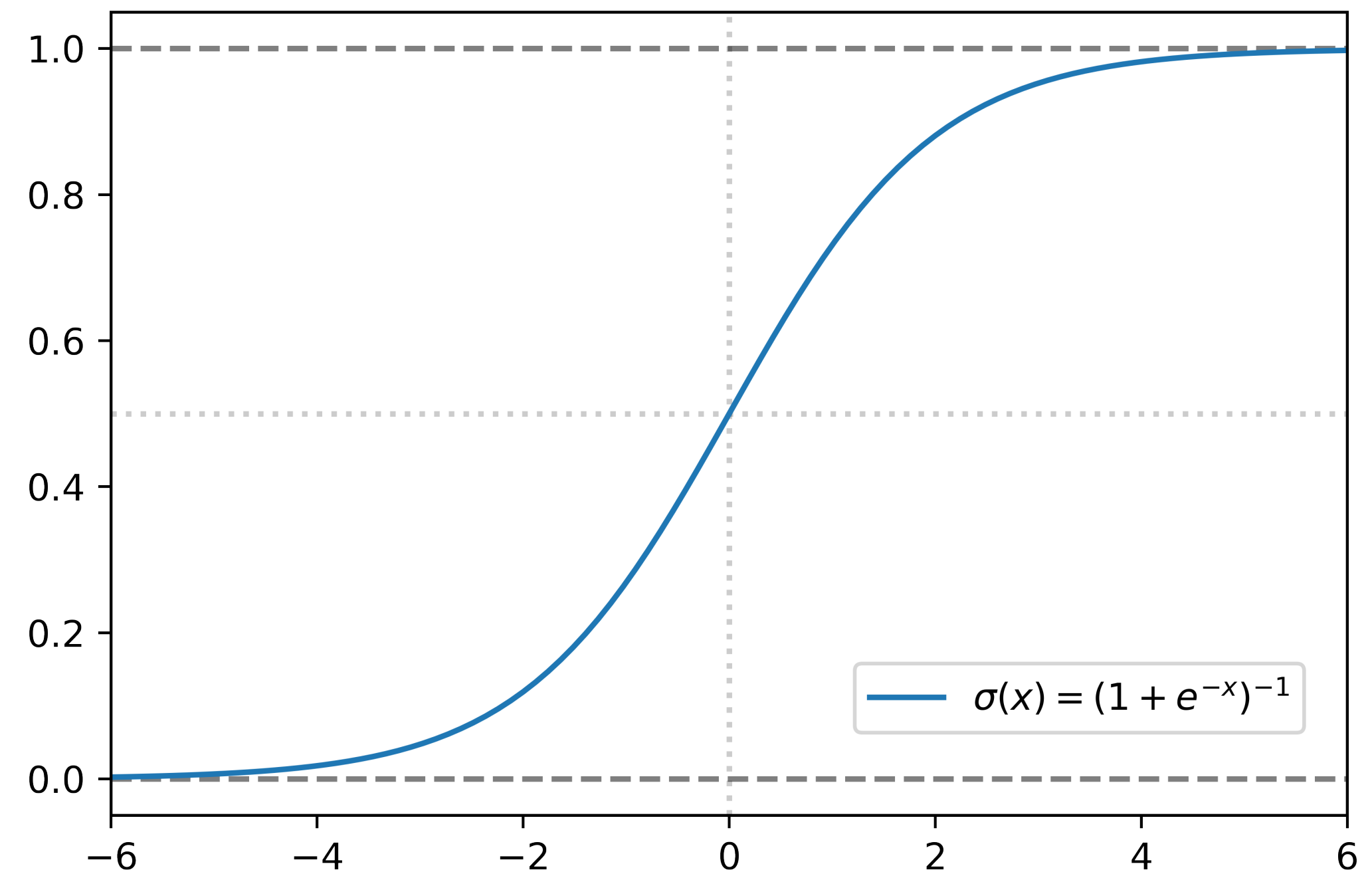
$$y = \frac{e^{x^T w + w_0}}{1 + e^{x^T w + w_0}} = \frac{1}{1 + e^{-(x^T w + w_0)}}$$

# The logistic function

$$\sigma(\eta) := \frac{e^\eta}{1 + e^\eta} = \frac{1}{1 + e^{-\eta}}$$

Properties of the logistic function:

- $1 - \sigma(\eta) = \frac{1 + e^\eta - e^\eta}{1 + e^\eta} = \frac{1}{1 + e^\eta}$
- $\sigma'(\eta) = \frac{e^\eta(1 + e^\eta) - e^\eta e^\eta}{(1 + e^\eta)^2} = \frac{e^\eta}{(1 + e^\eta)^2} = \sigma(\eta)(1 - \sigma(\eta))$



# Logistic Regression

$$p(1 | x) := \mathbb{P}(Y = 1 | X = x) = \sigma(x^\top w + w_0)$$

$$p(0 | x) := \mathbb{P}(Y = 0 | X = x) = 1 - \sigma(x^\top w + w_0)$$

**Logistic regression** models the **probability** that  $Y$  belongs to a particular class using the **logistic function**  $\sigma$

Label prediction: **quantize** the probability:

If  $p(1 | x) \geq 1/2$ , you predict the class 1

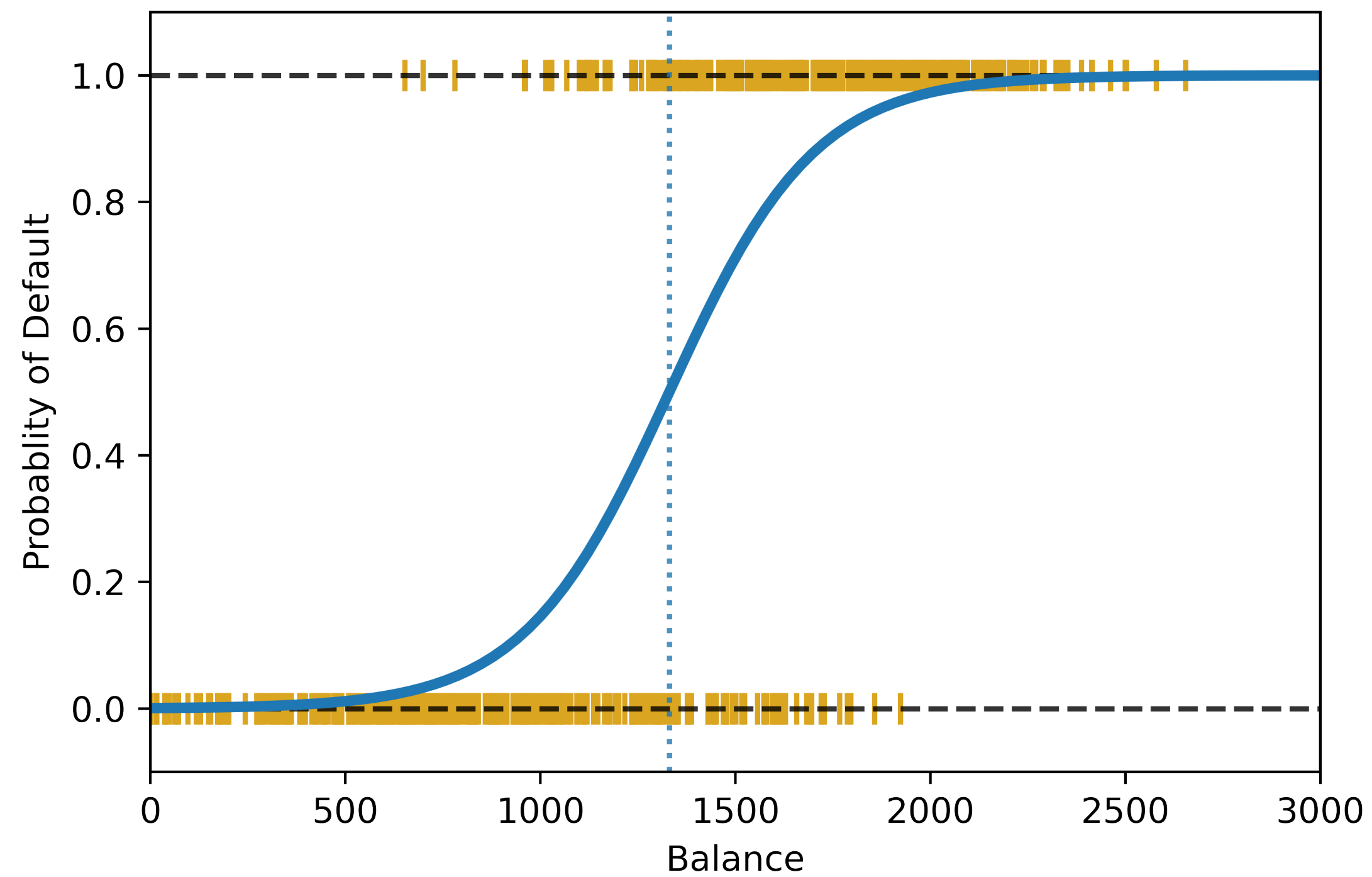
If  $p(1 | x) < 1/2$ , you predict the class 0

Note: Only sign of  $x^\top w + w_0$  matters for class prediction!

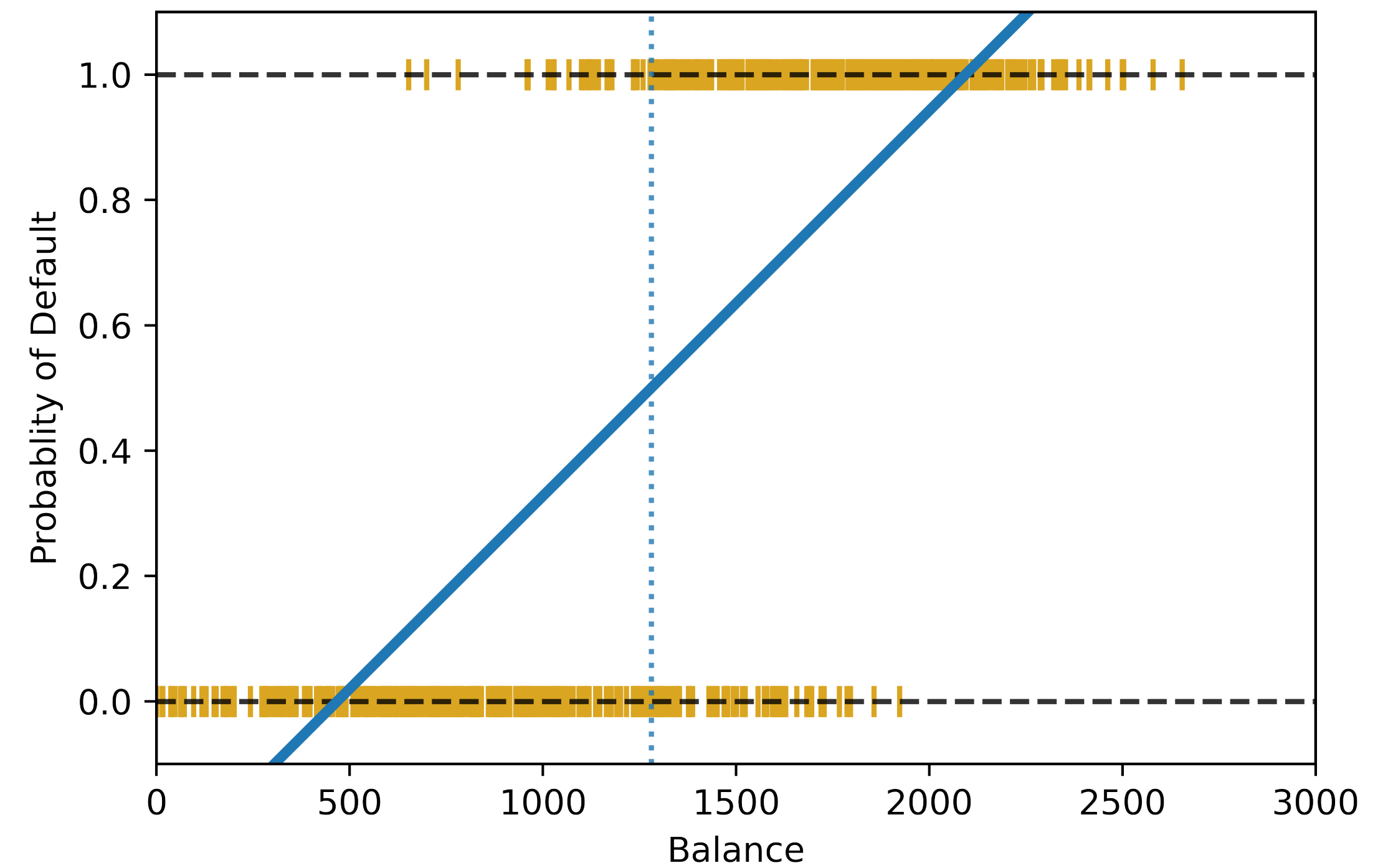
Interpretation:

- Very large  $|x^\top w + w_0|$  corresponds to  $p(1 | x)$  very close to 0 or 1 (high confidence)
- Small  $|x^\top w + w_0|$  corresponds to  $p(1 | x)$  very close to .5 (low confidence)

# Comparison of logistic and linear regression for balanced data

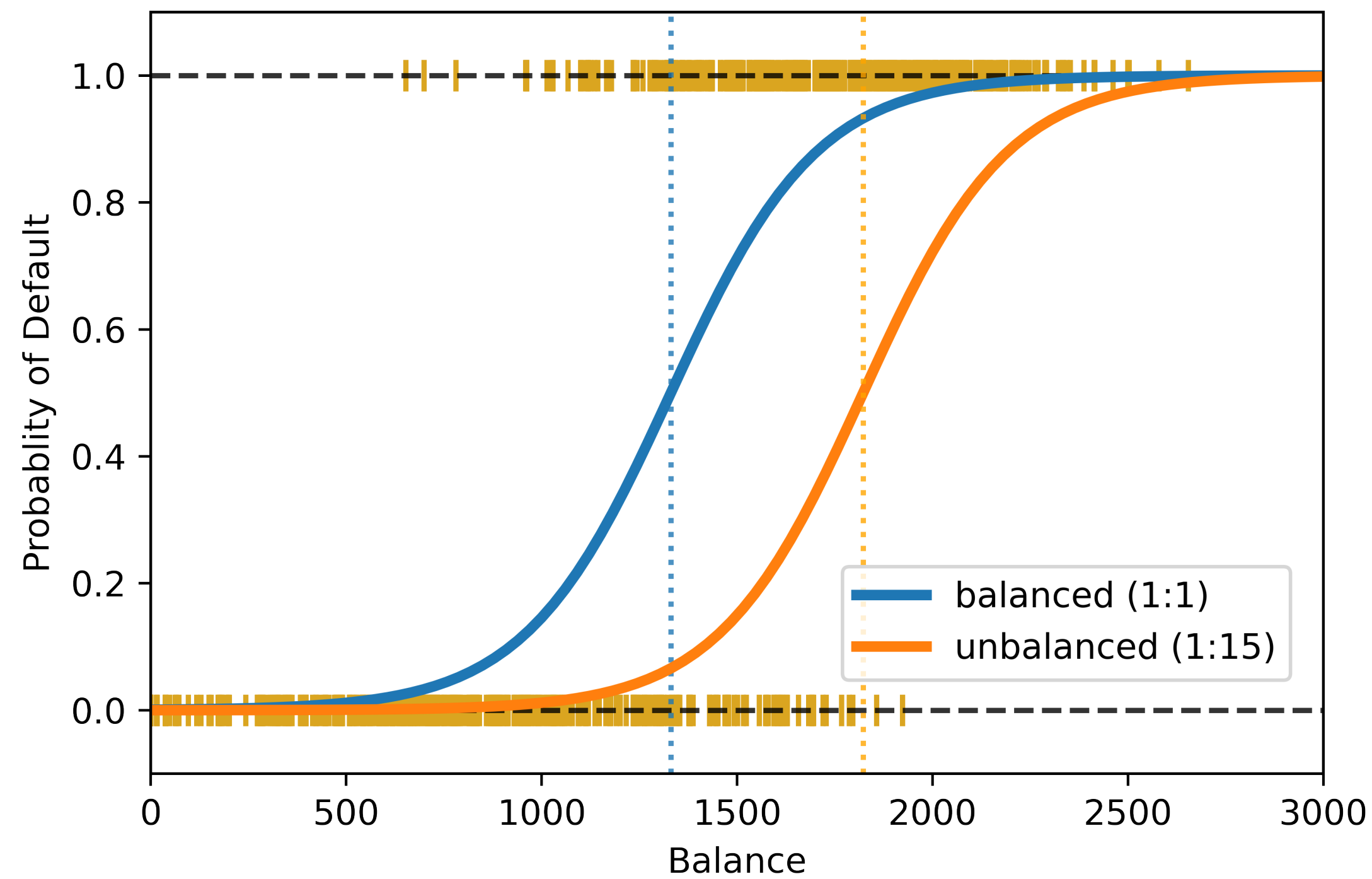


**Logistic regression**

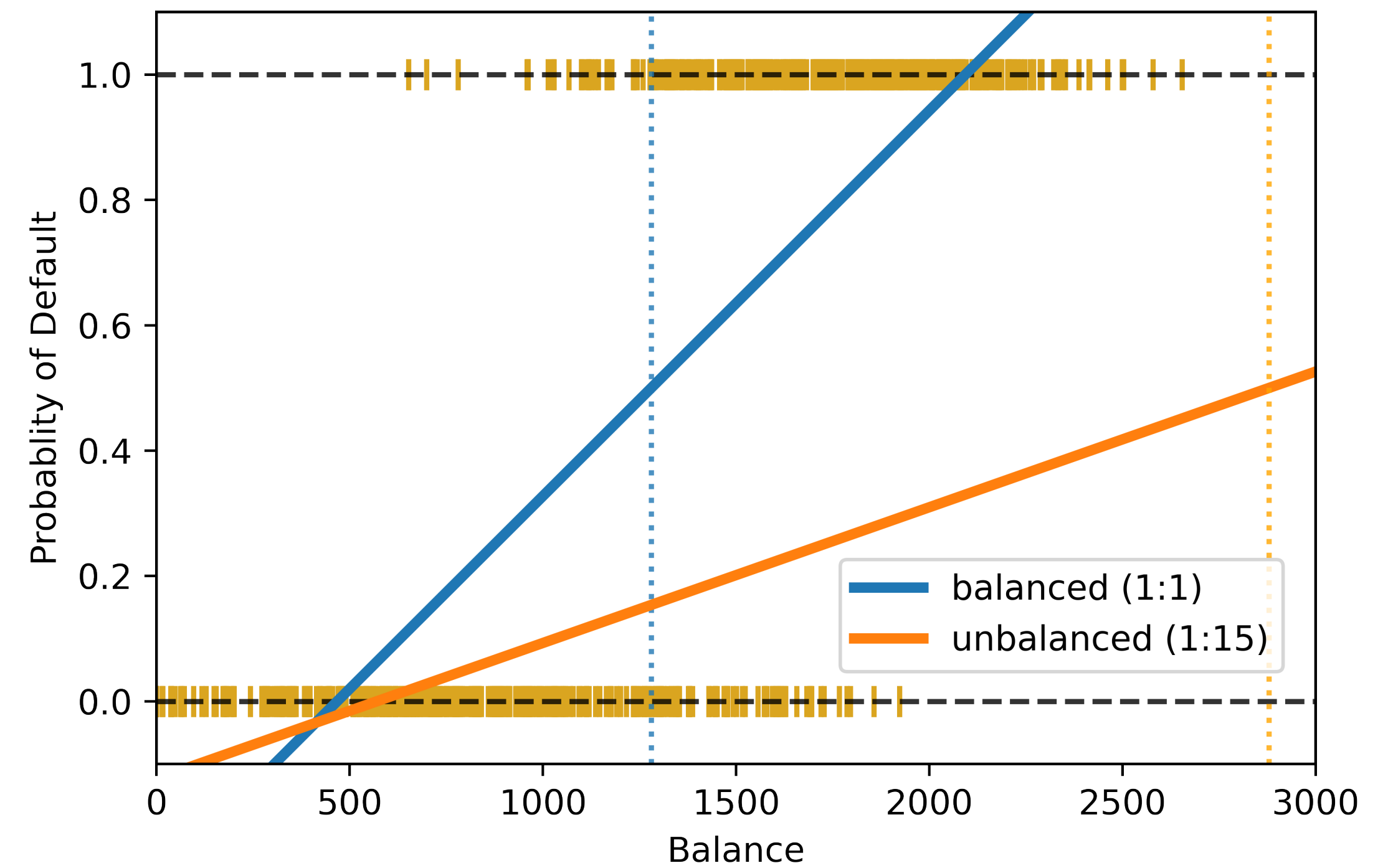


**Linear regression**

# Comparison of logistic and linear regression for unbalanced data



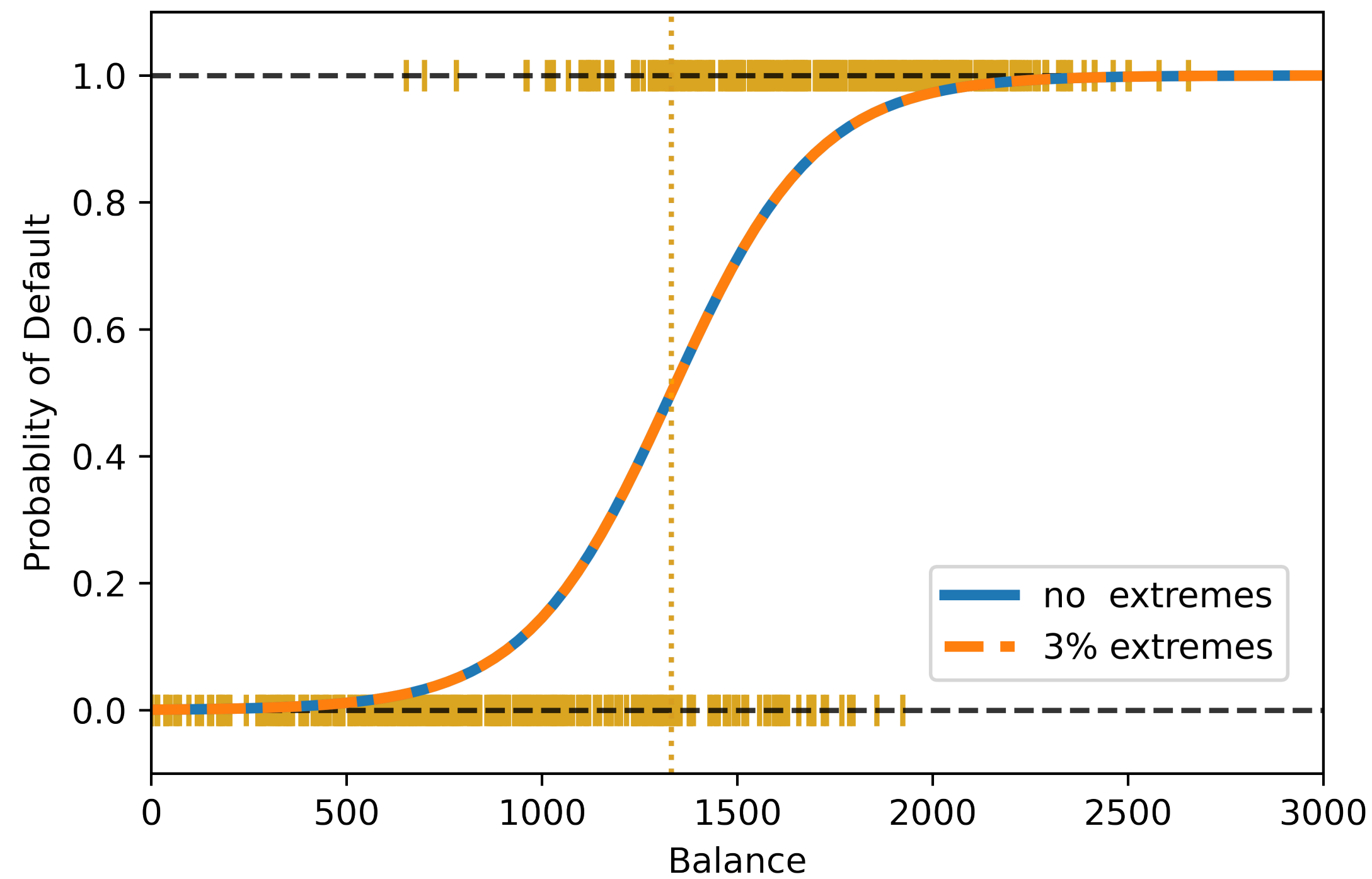
**Logistic regression**



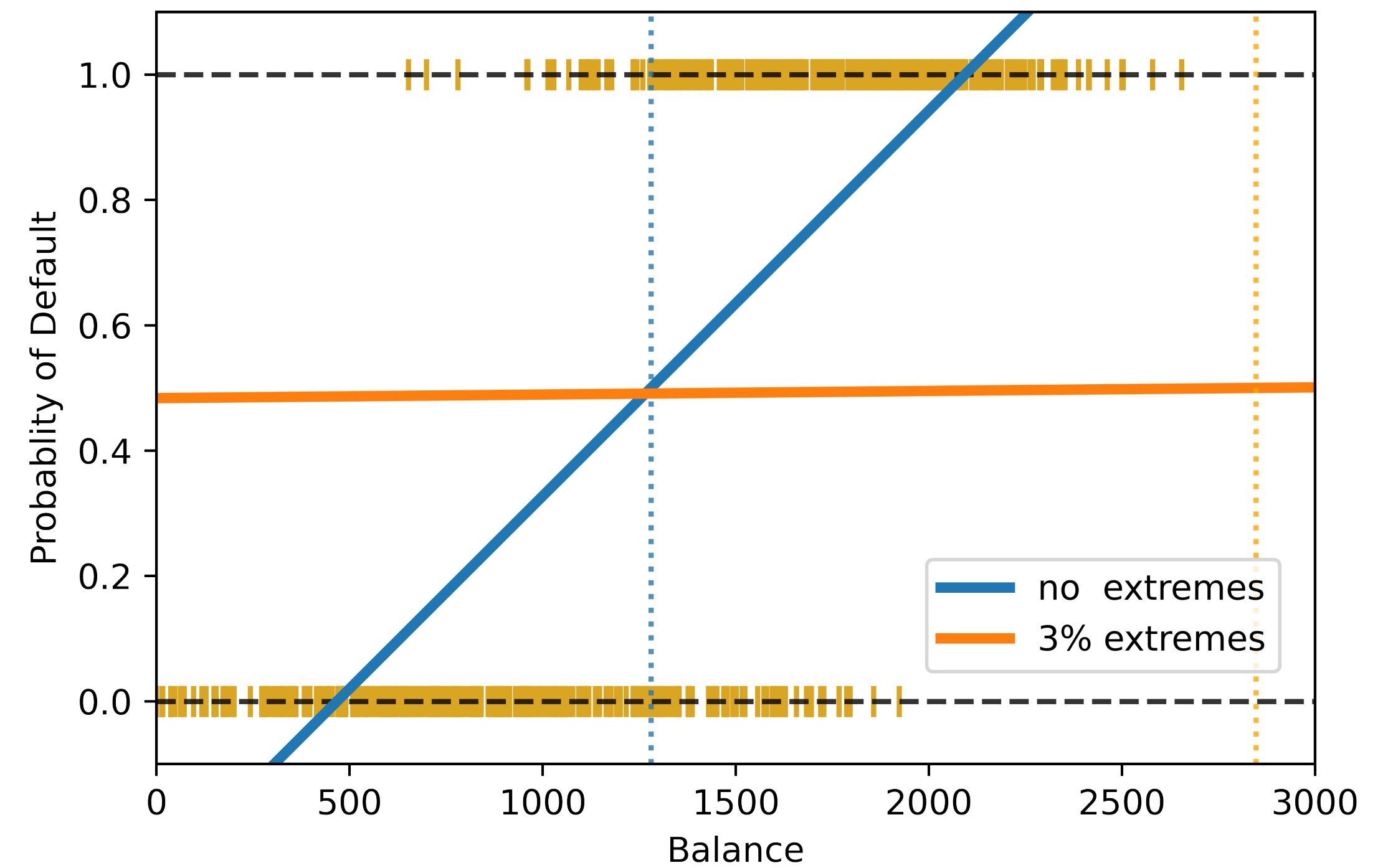
**Linear regression**



# Comparison of logistic and linear regression for data with extreme values

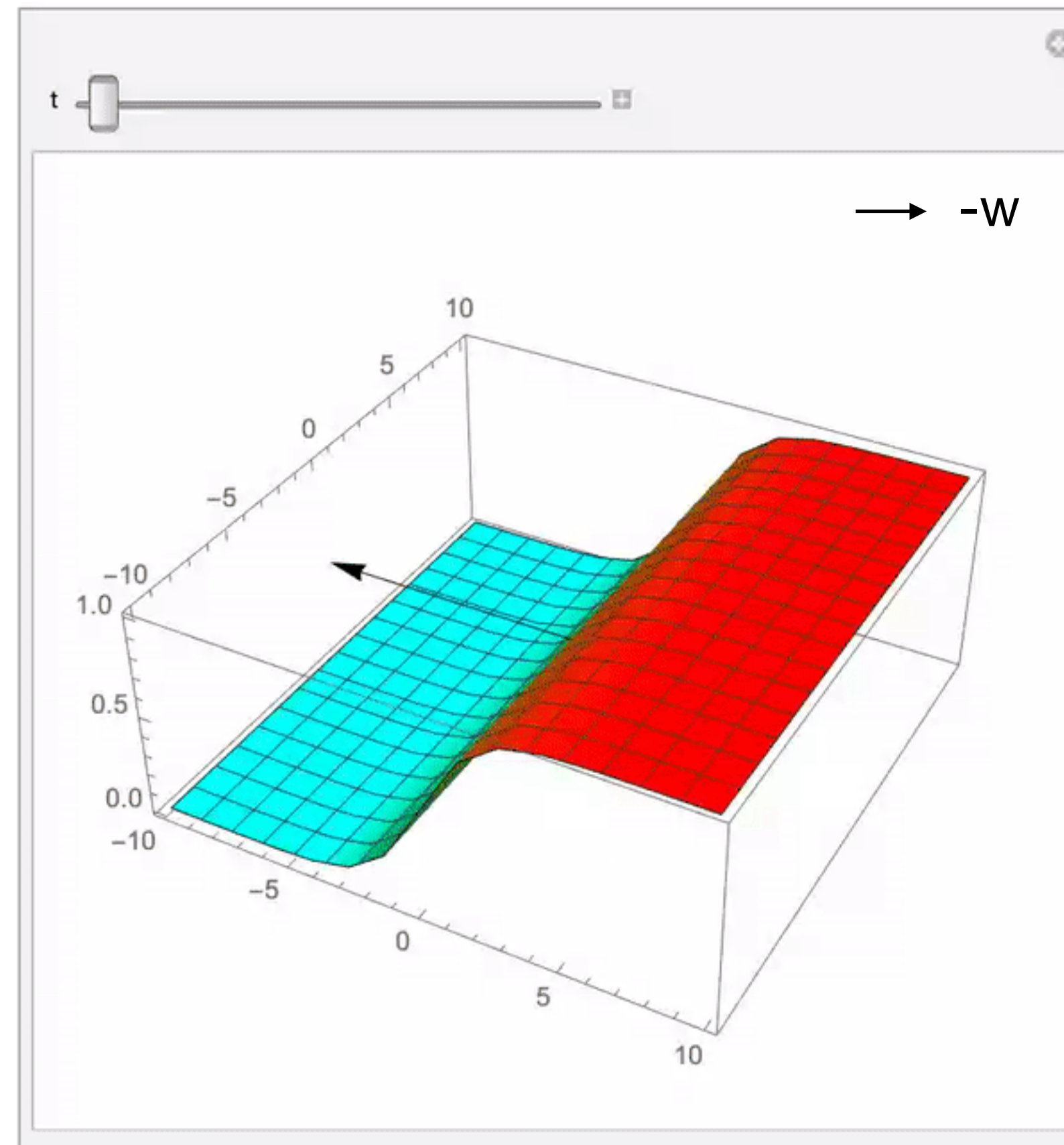


**Logistic regression**



**Linear regression**

# The vector $w$ is orthogonal to the “surface of transition”

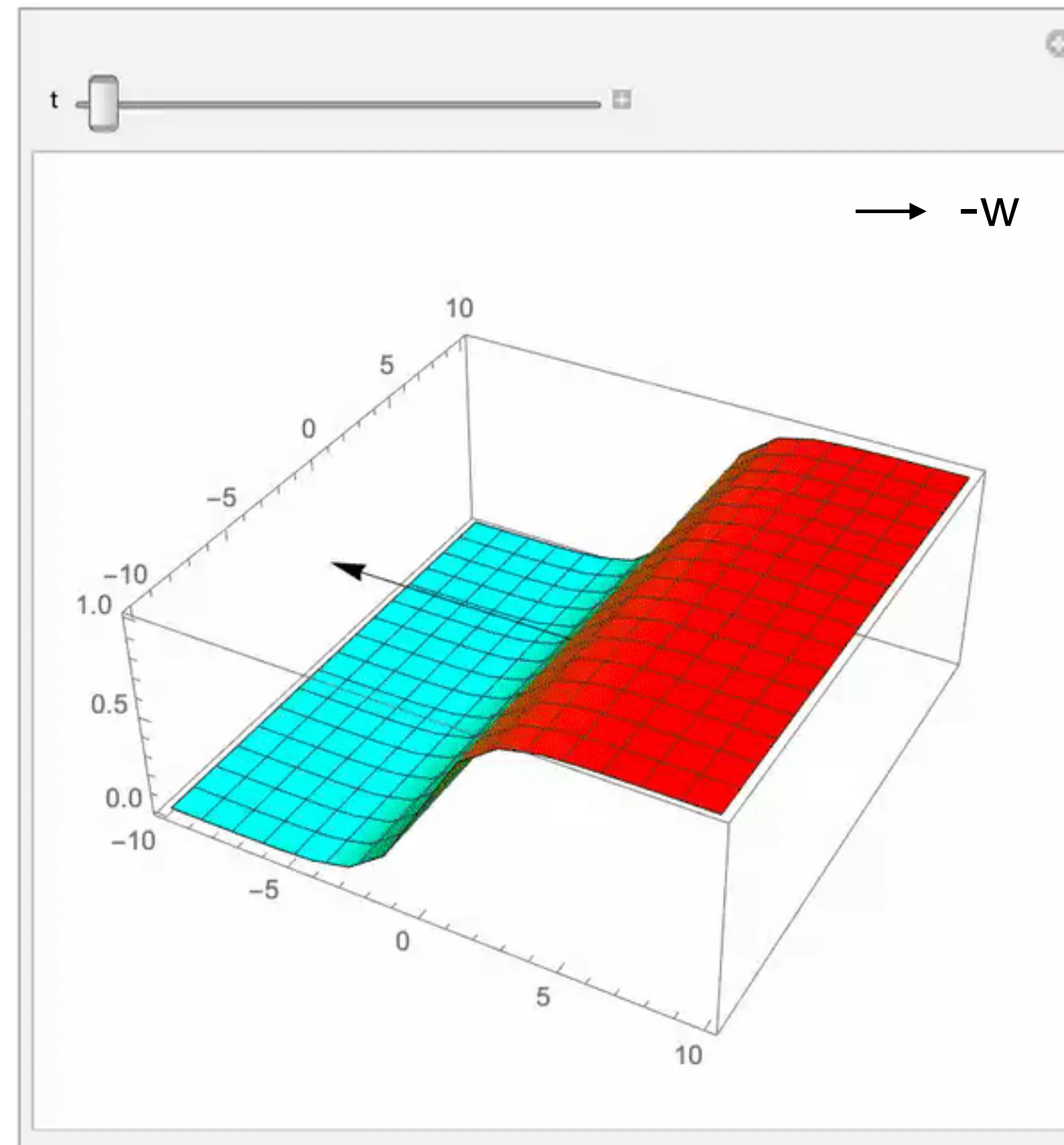


(See video)

$$\sigma(w^T x) \text{ for } \|w\| = 1$$

The transition between the two levels happens at the hyperplane  $w^\perp = \{v : v^T w = 0\}$

# The vector $w$ is orthogonal to the “surface of transition”



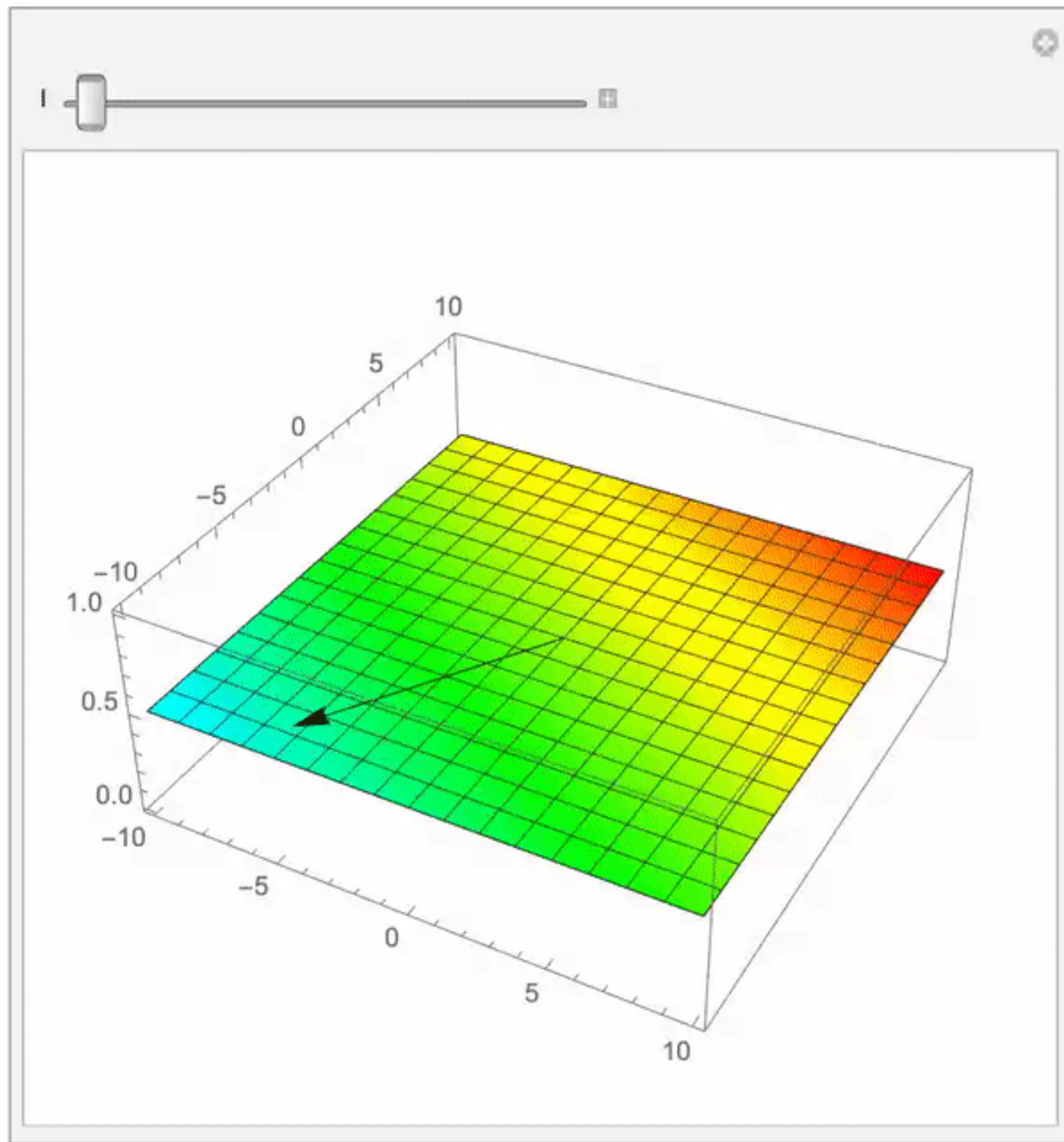
(See video)

$$\sigma(w^T x) \text{ for } \|w\| = 1$$

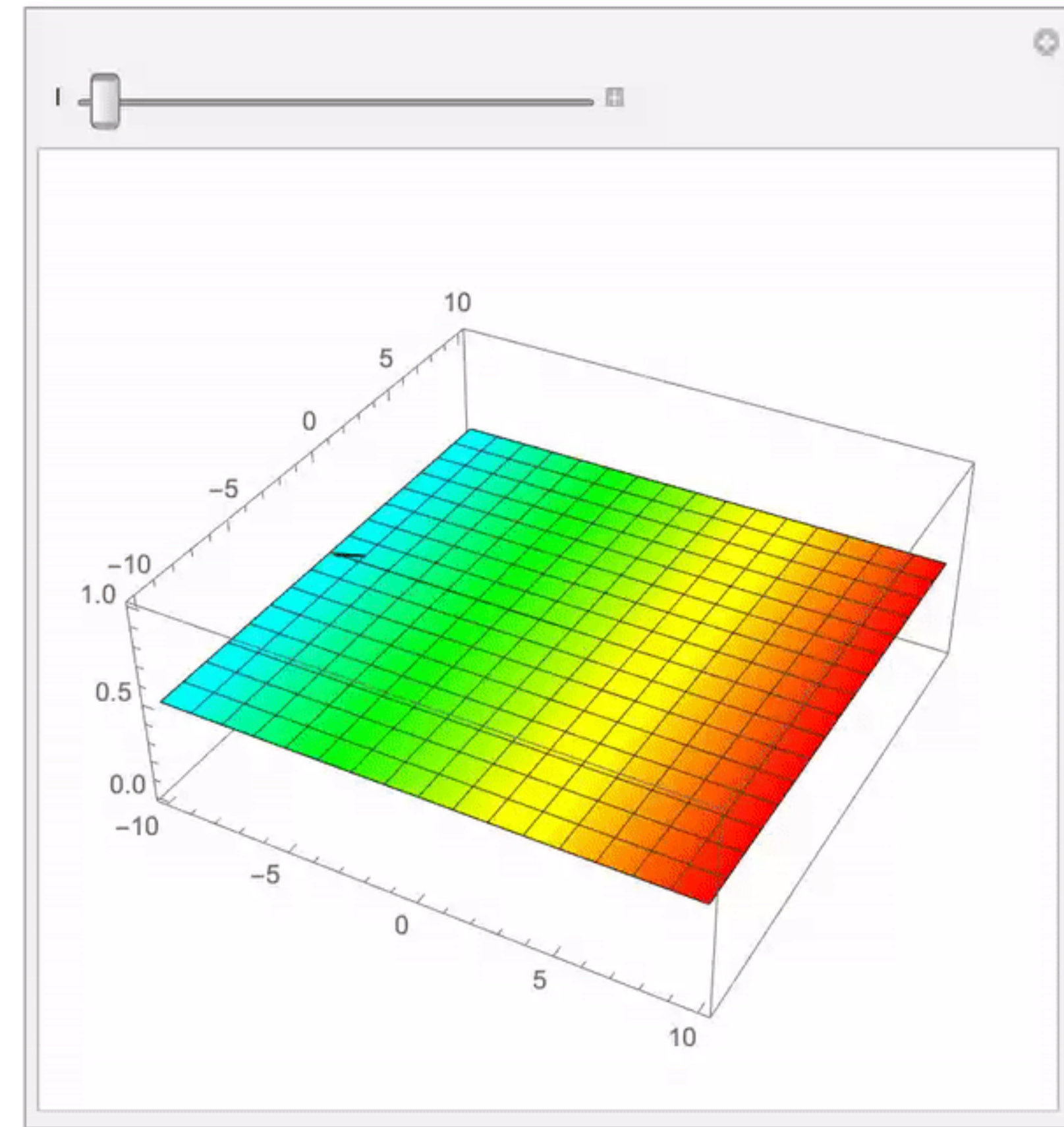
The transition between the two levels happens at the hyperplane  $w^\perp = \{v : v^T w = 0\}$



# Scaling $w$ makes the transition faster or slower



$$\sigma(t \cdot w_1^T x) \text{ for } t \in [e^{-10}, e^{10}]$$

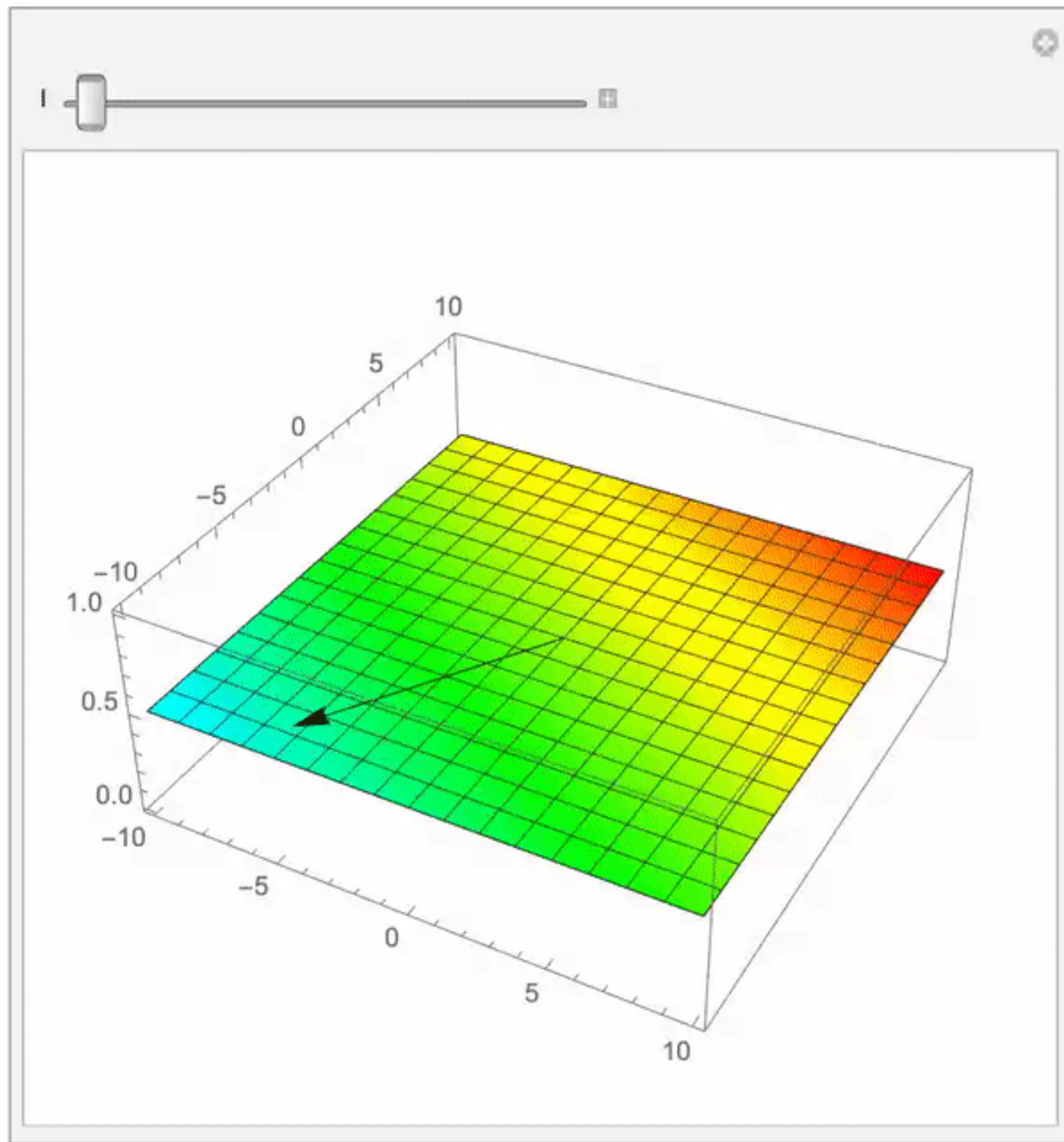


$$\sigma(t \cdot w_2^T x) \text{ for } t \in [e^{-10}, e^{10}]$$

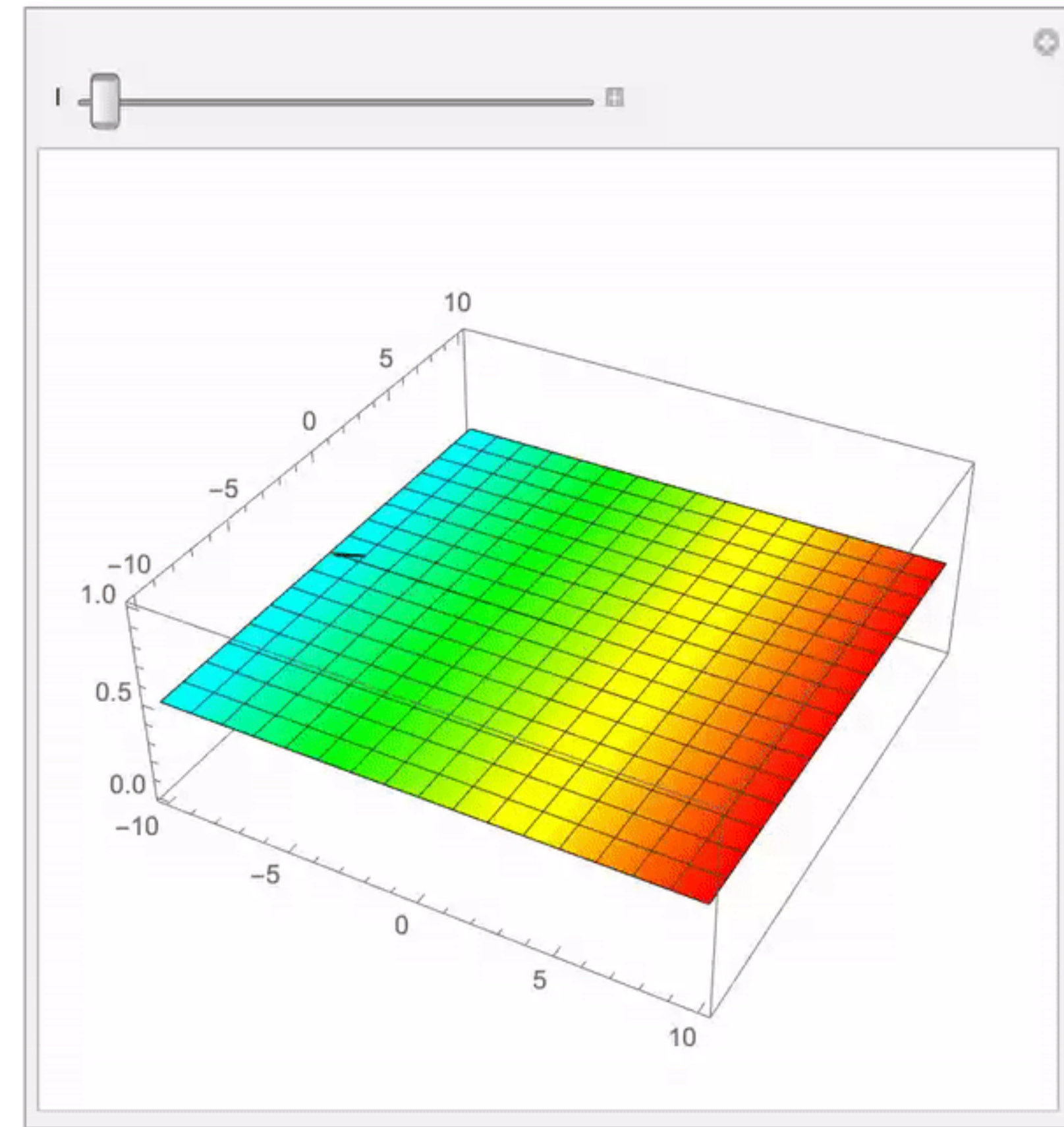
(See video)



# Scaling $w$ makes the transition faster or slower



$$\sigma(t \cdot w_1^T x) \text{ for } t \in [e^{-10}, e^{10}]$$

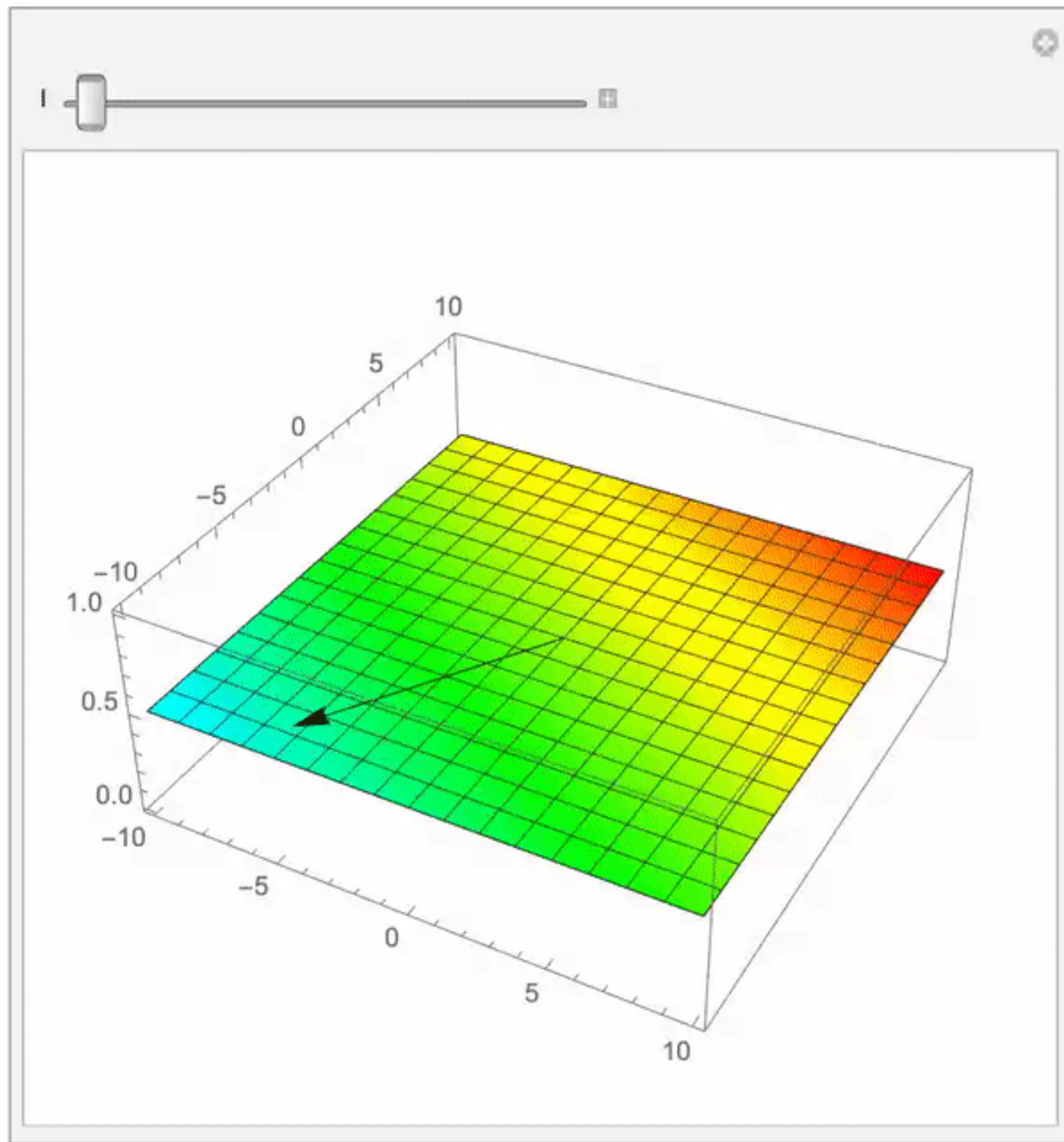


$$\sigma(t \cdot w_2^T x) \text{ for } t \in [e^{-10}, e^{10}]$$

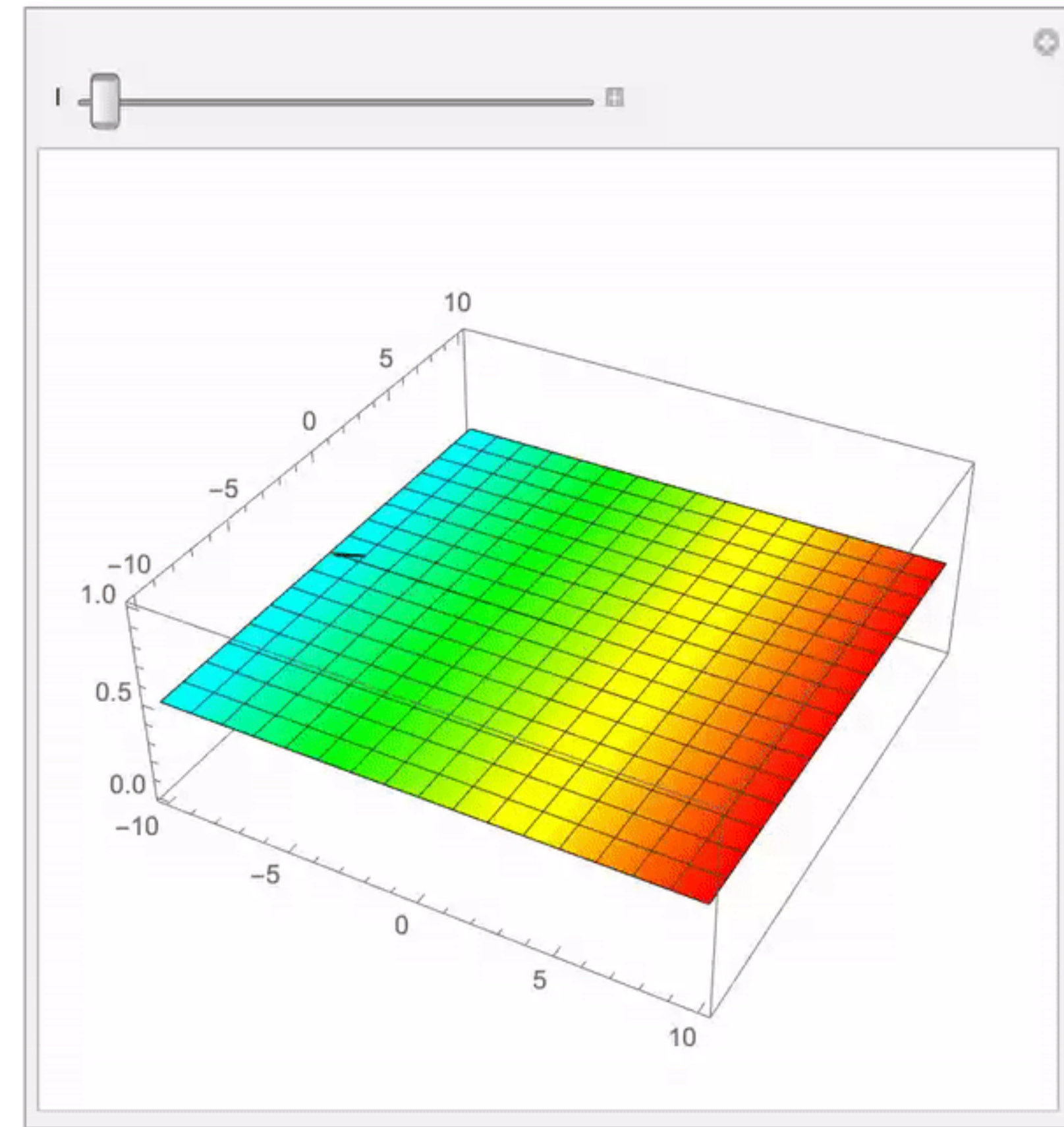
(See video)



# Scaling $w$ makes the transition faster or slower



$$\sigma(t \cdot w_1^T x) \text{ for } t \in [e^{-10}, e^{10}]$$

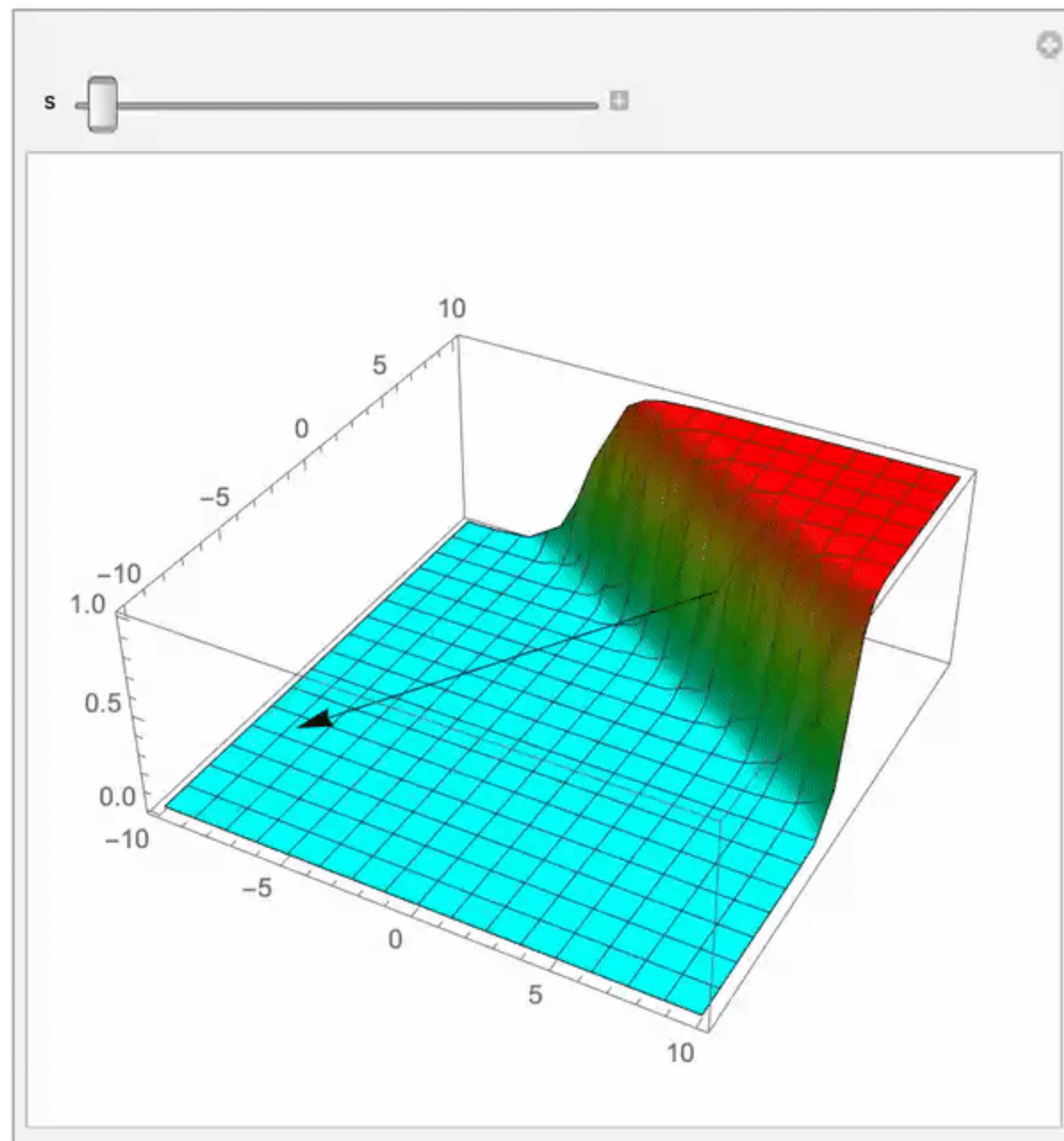


$$\sigma(t \cdot w_2^T x) \text{ for } t \in [e^{-10}, e^{10}]$$

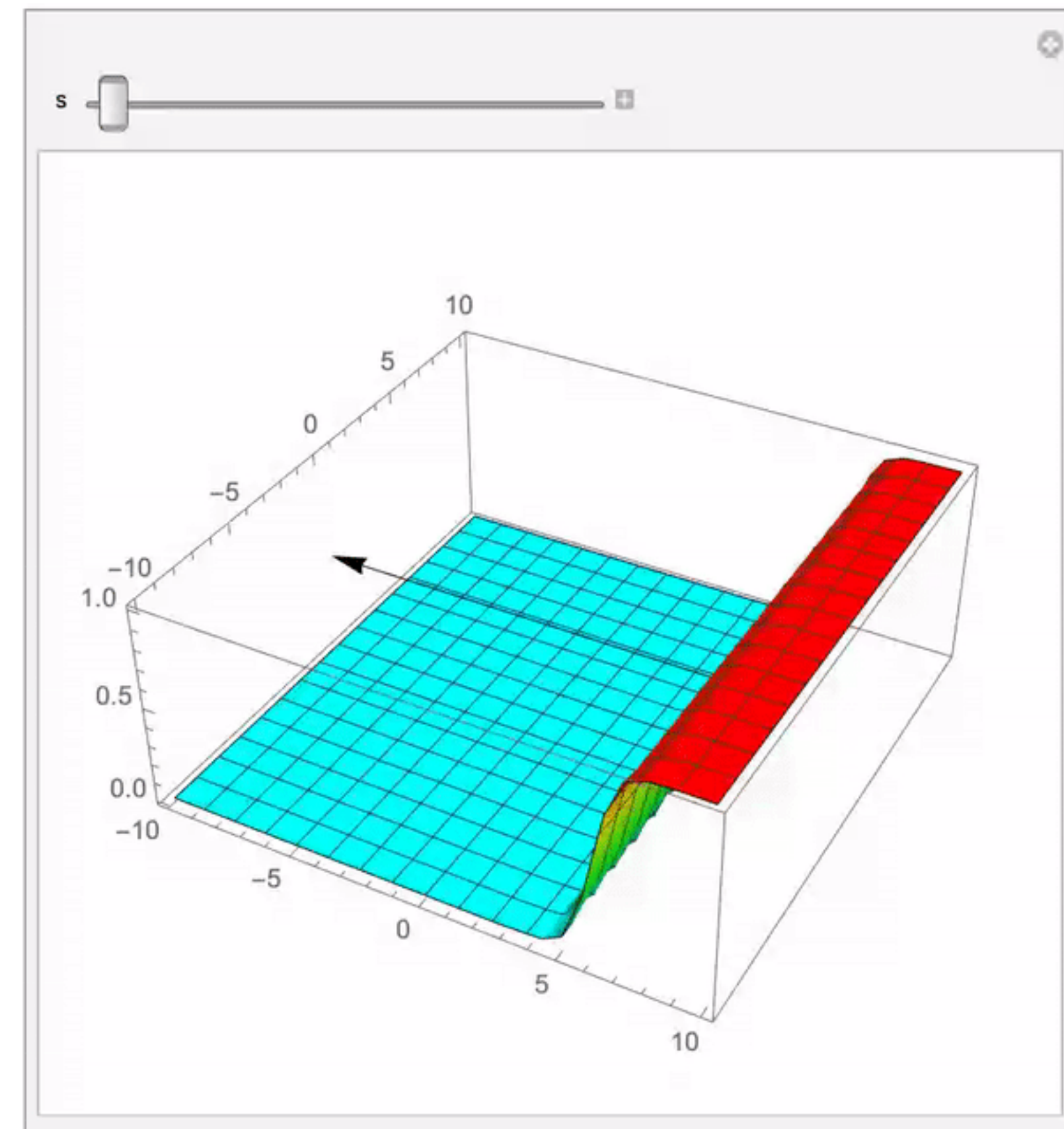
(See video)



# Changing $w_0$ shifts the decision region along the $w$ vector



$$\sigma(w_1^T x + w_0) \text{ for } w_0 \in [-6, 6]$$



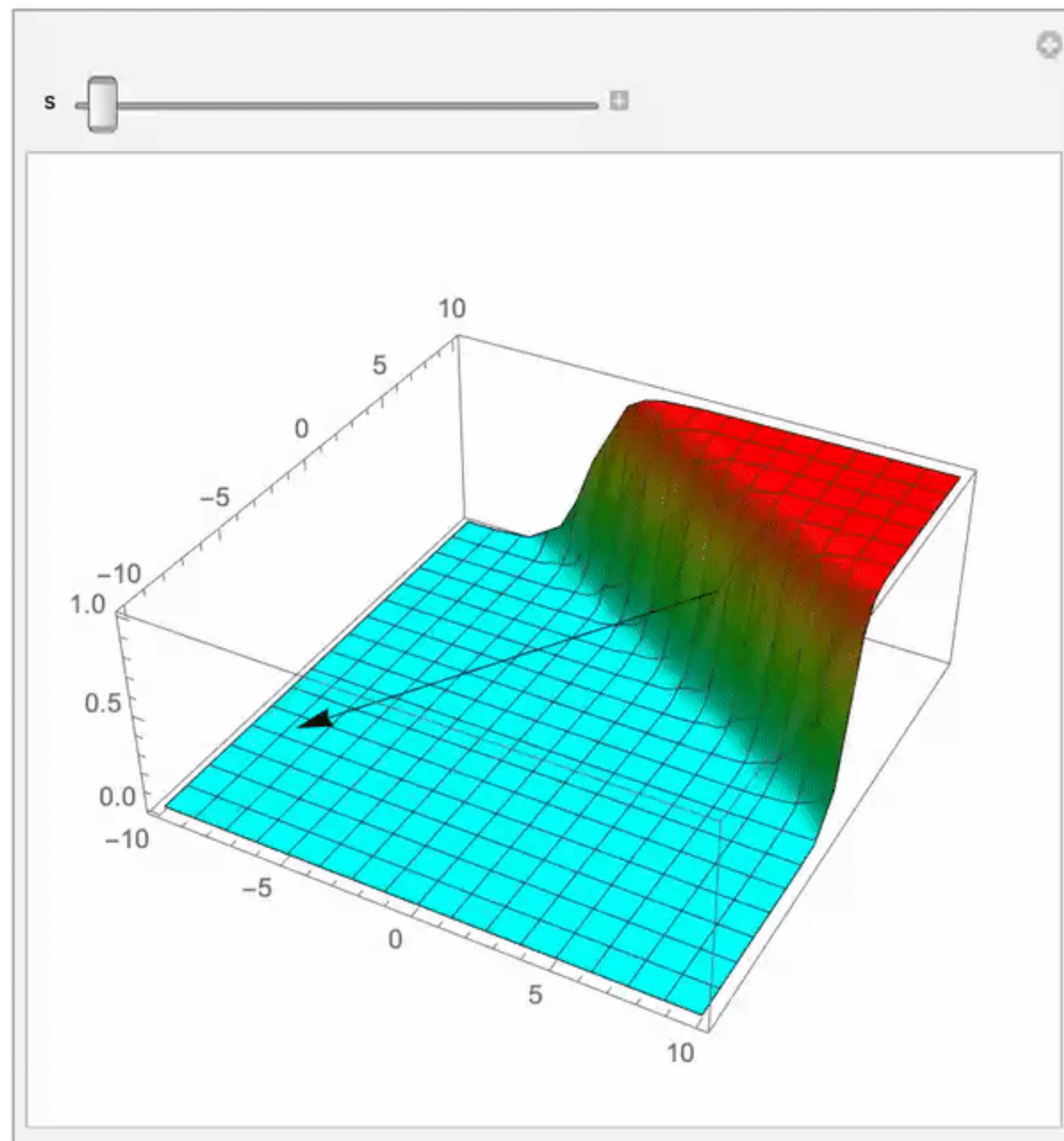
$$\sigma(w_2^T x + w_0) \text{ for } w_0 \in [-6, 6]$$

(See video)

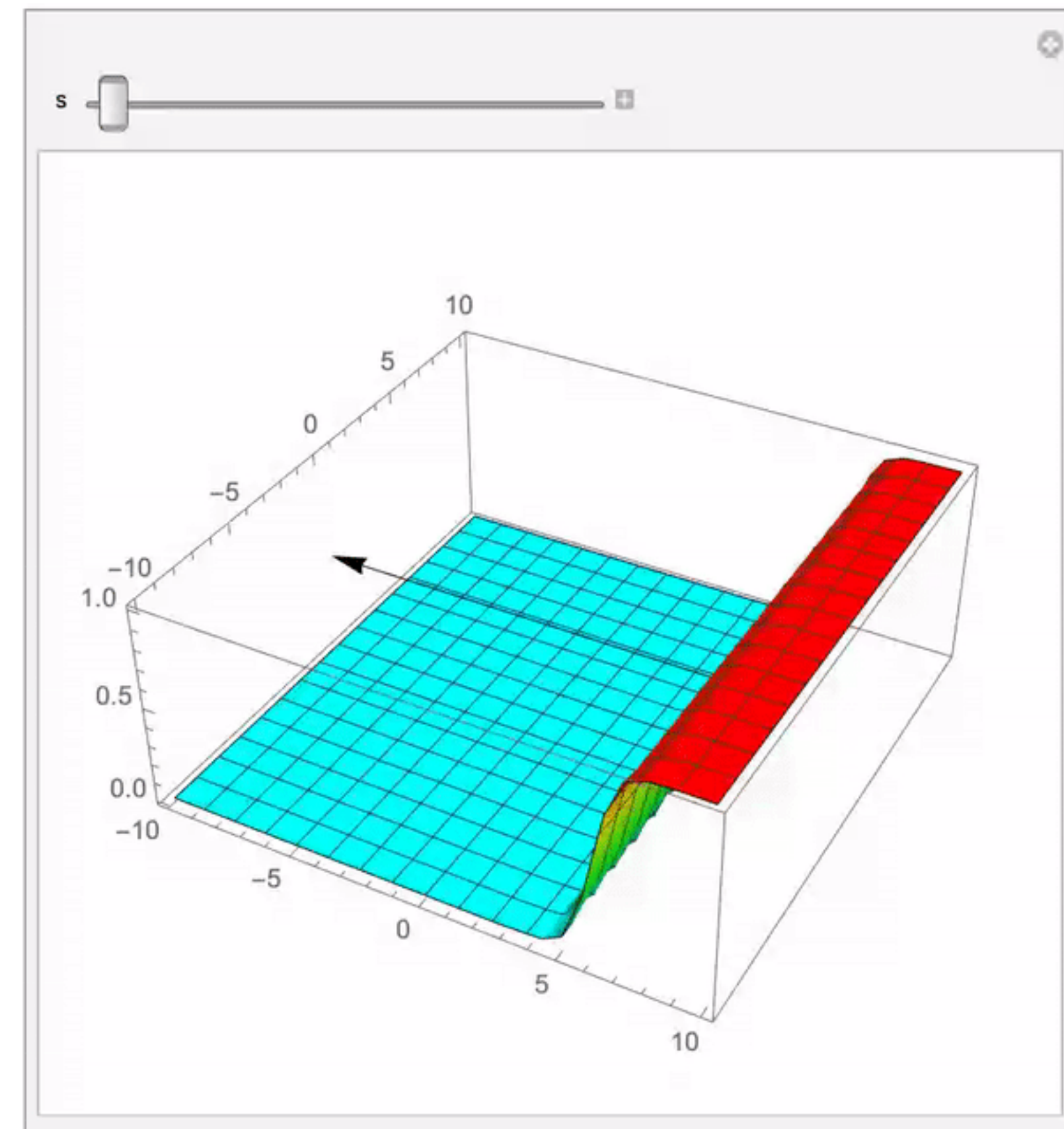
The transition happens at the hyperplane  $\{v : v^T w + w_0 = 0\}$



# Changing $w_0$ shifts the decision region along the $w$ vector



$$\sigma(w_1^T x + w_0) \text{ for } w_0 \in [-6, 6]$$



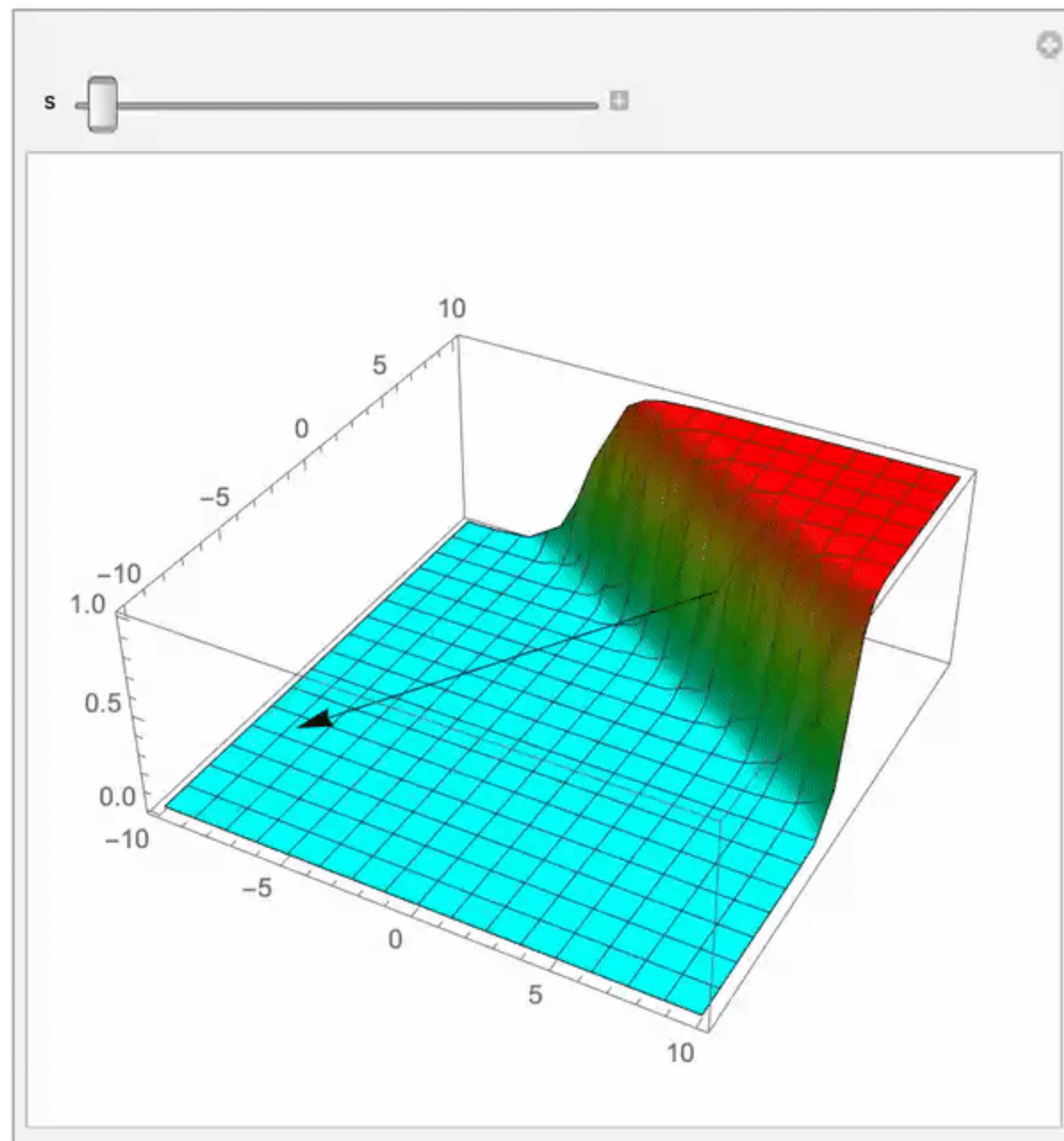
$$\sigma(w_2^T x + w_0) \text{ for } w_0 \in [-6, 6]$$

(See video)

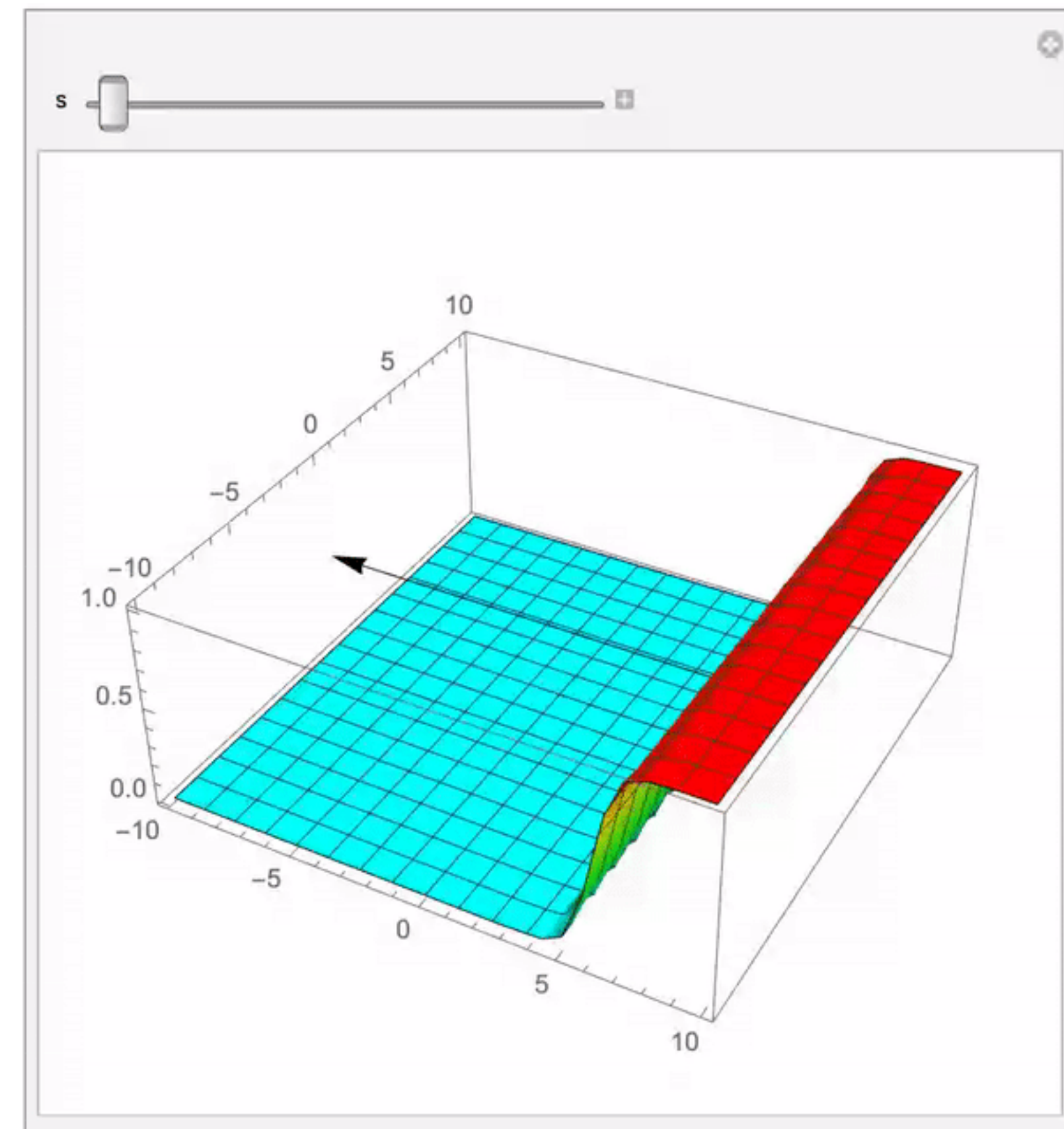
The transition happens at the hyperplane  $\{v : v^T w + w_0 = 0\}$



# Changing $w_0$ shifts the decision region along the $w$ vector



$$\sigma(w_1^T x + w_0) \text{ for } w_0 \in [-6, 6]$$



$$\sigma(w_2^T x + w_0) \text{ for } w_0 \in [-6, 6]$$

(See video)

The transition happens at the hyperplane  $\{v : v^T w + w_0 = 0\}$

# What about the bias term?

We should consider a **shift**  $w_0$  as there is no reason for the transition hyperplane to pass through the origin:

$$p(1 | x) = \sigma(w^\top x + w_0)$$

However, for simplicity, we will prefer to **add the constant 1** to the feature vector

$$x = \begin{pmatrix} x \\ 1 \end{pmatrix}$$

It is crucial for allowing to shift the decision region

Note that **both options are equivalent**

# Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model

Given i.i.d. samples  $(z_1, \dots, z_N) \sim p(z_1, \dots, z_N | w)$ , the MLE finds the **parameters**  $w_*$  **under which the observations**  $z_1, \dots, z_N$  **are the most likely**:

$$w_* = \arg \max_{\substack{\uparrow \\ \text{Likelihood function}}} \mathcal{L}(w) := p(z_1, \dots, z_N | w) \underset{\substack{\uparrow \\ \text{i.i.d. obs}}}{=} \prod_{n=1}^N p(z_n | w)$$

# Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model

Given i.i.d. samples  $(z_1, \dots, z_N) \sim p(z_1, \dots, z_N | w)$ , the MLE finds the **parameters**  $w_*$  **under which the observations**  $z_1, \dots, z_N$  **are the most likely**:

$$w_* = \arg \max_{\substack{\uparrow \\ \text{Likelihood function}}} \mathcal{L}(w) := p(z_1, \dots, z_N | w) \underset{\substack{\uparrow \\ \text{i.i.d. obs}}}{=} \prod_{n=1}^N p(z_n | w)$$

Often more convenient to work with the **negative log-likelihood**:

$$w_* = \arg \min [-\log(\mathcal{L}(w))] = \arg \min \sum_{n=1}^N -\log(p(z_n | w))$$

This estimator is **consistent\***: if the data are generated according to the model, the MLE converges to the true parameter when  $n \rightarrow \infty$

In practice, data are not generated according to it, but it still provides a theoretical justification

\*under mild technical conditions

# MLE for logistic regression

Assumption: The inputs  $\mathbf{X}$  do not depend on the parameter  $w$  we choose:

$$\mathcal{L}(w) = p(\mathbf{y}, \mathbf{X} | w) = p(\mathbf{X} | w) p(\mathbf{y} | \mathbf{X}, w) \underset{\mathbf{X} \perp\!\!\!\perp w}{=} p(\mathbf{X}) p(\mathbf{y} | \mathbf{X}, w)$$

↖ cst independant of  $w$

$$\begin{aligned} p(\mathbf{y} | \mathbf{X}, w) &= \prod_{n=1}^N p(y_n | x_n, w) \\ &= \prod_{n:y_n=1} p(y_n = 1 | x_n, w) \prod_{n:y_n=0} p(y_n = 0 | x_n, w) \\ &= \prod_{n=1}^N \sigma(x_n^\top w)^{y_n} [1 - \sigma(x_n^\top w)]^{1-y_n} \end{aligned}$$

The likelihood is proportional to:

$$\mathcal{L}(w) \propto \prod_{n=1}^N \sigma(x_n^\top w)^{y_n} [1 - \sigma(x_n^\top w)]^{1-y_n}$$

# Minimum of the negative log likelihood

It is more convenient to work with the negative log-likelihood:

$$\begin{aligned} -\log(p(\mathbf{y} | \mathbf{X}, w)) &= -\log\left(\prod_{n=1}^N \sigma(x_n^\top w)^{y_n} [1 - \sigma(x_n^\top w)]^{1-y_n}\right) \\ &= -\sum_{n=1}^N y_n \log \sigma(x_n^\top w) + (1 - y_n) \log(1 - \sigma(x_n^\top w)) \\ &= \sum_{n=1}^N y_n \log\left(\frac{1 - \sigma(x_n^\top w)}{\sigma(x_n^\top w)}\right) - \log(1 - \sigma(x_n^\top w)) \\ &= \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w}) \quad \leftarrow 1 - \sigma(\eta) = \frac{1}{1 + e^\eta} \Rightarrow \frac{1 - \sigma(\eta)}{\sigma(\eta)} = e^{-\eta} \end{aligned}$$

We obtain the following cost function we will minimize to learn the parameter  $w_*$

$$w_* = \arg \min L(w) := \frac{1}{N} \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w})$$

\*If we are considering  $y \in \{-1, 1\}$ , we will have a different function

\*\* minimizing  $L$  is exactly equivalent to maximize the likelihood  $\mathcal{L}$  since  $p(X) \perp\!\!\!\perp w$



# A side note on logistic loss

In logistic regression, the **negative log likelihood** is equivalent to ERM for the **logistic loss** (a surrogate for 0-1 loss, as discussed yesterday)

- Logistic loss for  $y \in \{0,1\}$ :

$$\ell(y, g(x)) = -yg(x) + \log(1 + \exp(g(x)))$$

- Logistic loss for  $y \in \{-1,1\}$ :

$$\ell(y, g(x)) = \log(1 + \exp(-yg(x)))$$

Note: the logistic loss can be applied in modern machine learning as well:  $g(x)$  can represent the output of a neural network

# Gradient of the negative log likelihood

To minimize  $L$ , let's first look at its stationary points by computing its gradient:

$$\nabla L(w) = \nabla \left[ \frac{1}{N} \sum_{n=1}^N \log(1 + e^{x_n^\top w}) - y_n x_n^\top w \right] = \frac{1}{N} \sum_{n=1}^N \frac{e^{x_n^\top w} x_n}{1 + e^{x_n^\top w}} - y_n x_n = \frac{1}{N} \sum_{n=1}^N (\sigma(x_n^\top w) - y_n) x_n$$

Which can be written under the matrix form  $\mathbf{X} = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}$  and  $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$

$$\nabla L(w) = \frac{1}{N} \mathbf{X}^\top (\sigma(\mathbf{X}w) - \mathbf{y})$$

- Same gradient as in LS but with  $\sigma$
- No closed form solution to  $\nabla L(w) = 0$
- Good news: the cost function  $L$  is convex



# Convexity of the loss function $L$

Claim: The function

$$L(w) = \frac{1}{N} \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w})$$

is convex in the weight vector  $w$

Proof:  $L$  is obtained through simple convexity preserving operations:

1. Positive combinations of convex functions is convex
2. Composition of a convex and a linear functions is convex
3. A linear function is both convex and concave
4.  $\eta \mapsto \log(1 + e^\eta)$  is convex

# Convexity of the loss function $L$

Claim: The function

$$L(w) = \frac{1}{N} \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w})$$

is convex in the weight vector  $w$

Proof:  $L$  is obtained through simple convexity preserving operations:

1. Positive combinations of convex f
2. Composition of a convex and a lin
3. A linear function is both convex and
4.  $\eta \mapsto \log(1 + e^\eta)$  is convex

Proof of 4:  $h(\eta) := \log(1 + e^\eta)$  is cvx

$$h'(\eta) = \frac{e^\eta}{1 + e^\eta} = \sigma(\eta)$$

$$h''(\eta) = \sigma'(\eta) = \frac{e^\eta}{(1 + e^\eta)^2} \geq 0$$

# Proof of the convexity of $L$

- 2. Composition of a convex and a linear functions is convex
- 4.  $\eta \mapsto \log(1 + e^\eta)$  is convex

$$\log(1 + e^{x_n^\top w}) \text{ is convex}$$

- 3. A linear function is both convex and concave

$$-y_n x_n^\top w \text{ is convex}$$

- 1. Positive combinations of convex functions is convex

$$L(w) = \frac{1}{N} \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w}) \text{ is convex}$$

# Second proof: Hessian of $L$ is psd

The Hessian  $\nabla^2 L$  is the **matrix** whose entries are the **second derivatives**  $\frac{\partial^2}{\partial w_i \partial w_j} L(w)$

$$\begin{aligned}\nabla^2 L(w) &= \nabla [\nabla L(w)]^\top \\ &= \nabla \left[ \frac{1}{N} \sum_{n=1}^N x_n (\sigma(x_n^\top w) - y_n) \right]^\top \\ &= \frac{1}{N} \sum_{n=1}^N \nabla \sigma(x_n^\top w) x_n^\top = \frac{1}{N} \sum_{n=1}^N \sigma(x_n^\top w) (1 - \sigma(x_n^\top w)) x_n x_n^\top\end{aligned}$$

It can be written under the matrix form:

$$\nabla^2 L(w) = \frac{1}{N} \mathbf{X}^\top S \mathbf{X}, \quad \text{where } S = \text{diag} [\sigma(x_n^\top w) (1 - \sigma(x_n^\top w))] \succeq 0$$

➡  $L$  is convex since  $\nabla^2 L(w) \succeq 0$

# How to minimize the convex function $L$ ?

Gradient descent:

$$\begin{cases} w_0 \in \mathbb{R}^d \\ w_{t+1} = w_t - \gamma_t \nabla L(w_t) \end{cases}$$

can be slow

# How to minimize the convex function $L$ ?

Gradient descent:

$$\begin{cases} w_0 \in \mathbb{R}^d \\ w_{t+1} = w_t - \frac{\gamma_t}{N} \sum_{n=1}^N (\sigma(x_n^\top w_t) - y_n) x_n \end{cases}$$

can be slow

# How to minimize the convex function $L$ ?

Gradient descent:

$$\begin{cases} w_0 \in \mathbb{R}^d \\ w_{t+1} = w_t - \frac{\gamma_t}{N} \sum_{n=1}^N (\sigma(x_n^\top w_t) - y_n) x_n \end{cases}$$

can be slow

Stochastic gradient descent

$$\begin{cases} w_0 \in \mathbb{R}^d \\ w_{t+1} = w_t - \gamma_t (\sigma(x_{n_t}^\top w_t) - y_{n_t}) x_{n_t} \end{cases} \text{ where } \mathbb{P}[n_t = n] = 1/N$$

is faster but converges slower

# Newton's method uses second order information

Newton's method **minimizes** the **quadratic approximation**:

$$L(w) \sim L(w_t) + \nabla L(w_t)^\top (w - w_t) + \frac{1}{2}(w - w_t)^\top \nabla^2 L(w_t)(w - w_t) := \phi_t(w)$$

$$\tilde{w} = \arg \min \phi_t(w) \implies \nabla L(w_t) + \nabla^2 L(w_t)(\tilde{w} - w_t) = 0$$

Newton's method:  $w_{t+1} = w_t - \gamma_t \nabla^2 L(w_t)^{-1} \nabla L(w_t)$

The step-size is needed to ensure convergence (damped Newton's method)

The convergence is typically **faster than with gradient descent** but the **computational complexity is higher** (computing Hessian and solving a linear system)



# Problem when the data are linearly separable

$$\inf_w L(w) = 0 = \lim_{\alpha \rightarrow \infty} L(\alpha \cdot \bar{w})$$

The inf value is not attained for a finite  $w$

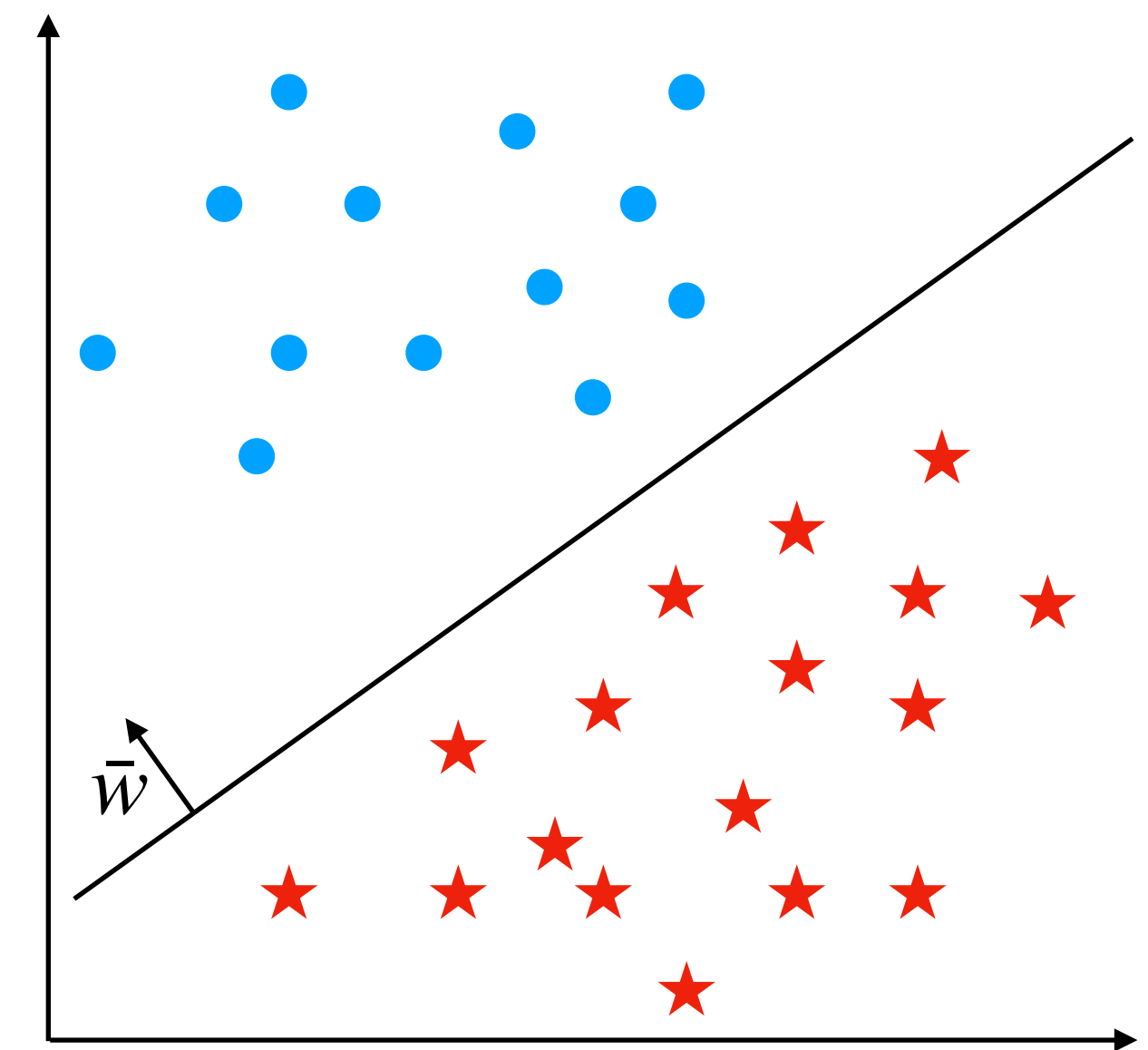
If we use an optimization algorithm, the weights will go to  $\infty$

Solution: add a  $\ell_2$ -regularization

➔ **ridge logistic regression**:

$$\frac{1}{N} \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w}) + \frac{\lambda}{2} \|w\|_2^2$$

- Optimization perspective: stabilize the optimization process
- Statistical perspective: avoid overfitting



$$L(w) = \frac{1}{N} \sum_{n=1}^N -y_n x_n^\top w + \log(1 + e^{x_n^\top w})$$

# Recap

- Logistic regression:
  - Maps inputs to output class probabilities
  - Exhibits robustness towards unbalanced data and extreme values
- How to solve logistic regression?
  - By minimizing the negative log-likelihood (a.k.a. logistic loss)
  - Using gradient methods or second-order methods
- Not ideal when data is linearly separable?
  - Weights go to infinity
  - A solution is to add a penalty term, e.g.  $\ell_2$ -regularization