Classification

Machine Learning Course - CS-433
Oct 19, 2021
Nicolas Flammarion

EPFL

Definition of classification

We observe some data $S = \{X_i, Y_i\}_{i=1}^n \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete Set}}$

Goal: given a new X, we want to predict its label Y

How:

Training Data $\longrightarrow$ Algorithm $\longrightarrow$ Prediction

$S = \{X_i, \underset{\substack{\cap \\ \text{Discrete } \mathcal{Y}}}{Y_i}\}_{i=1}^n$ $\qquad$ $\mathcal{A}$ $\qquad$ $f_S = \mathcal{A}(S)$

Classification: relates input to a categorical variable

$$(X, Y) \in \mathscr{X} \times \underbrace{\mathscr{Y}}_{\text{Discrete Set}}$$

Binary Classification: Y can take two values

$Y \in \{c_1, c_2\}$ where c_i are the class labels . We often use $\{0,1\}$ or $\{-1,1\}$

Multi-class classification: Y can take more than two values

$Y \in \{c_1, \cdots, c_{K-1}\}$ for a K class problem. We often use $\{0, \cdots, K-1\}$

no ordering between classes

Spam Detection

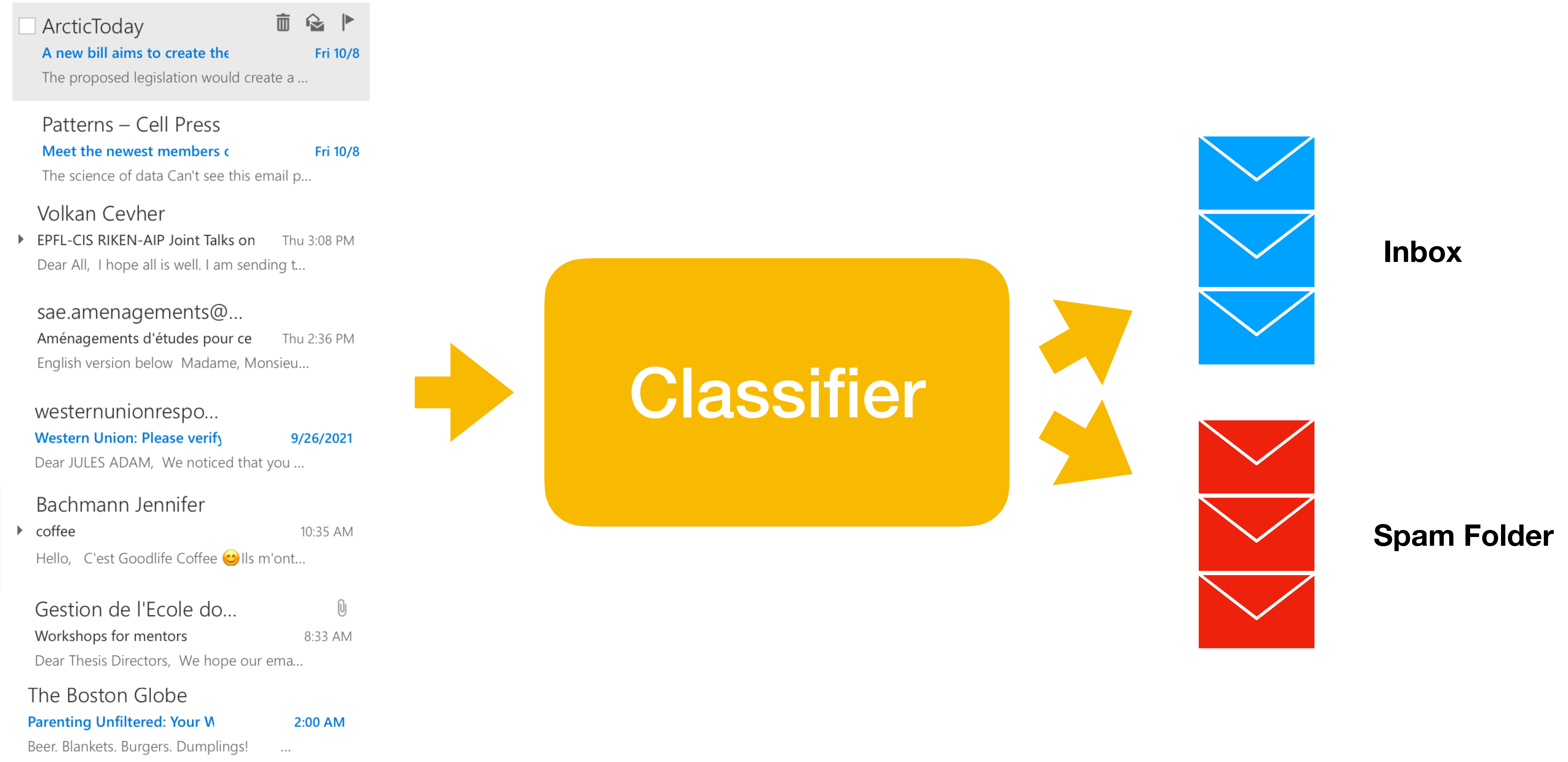
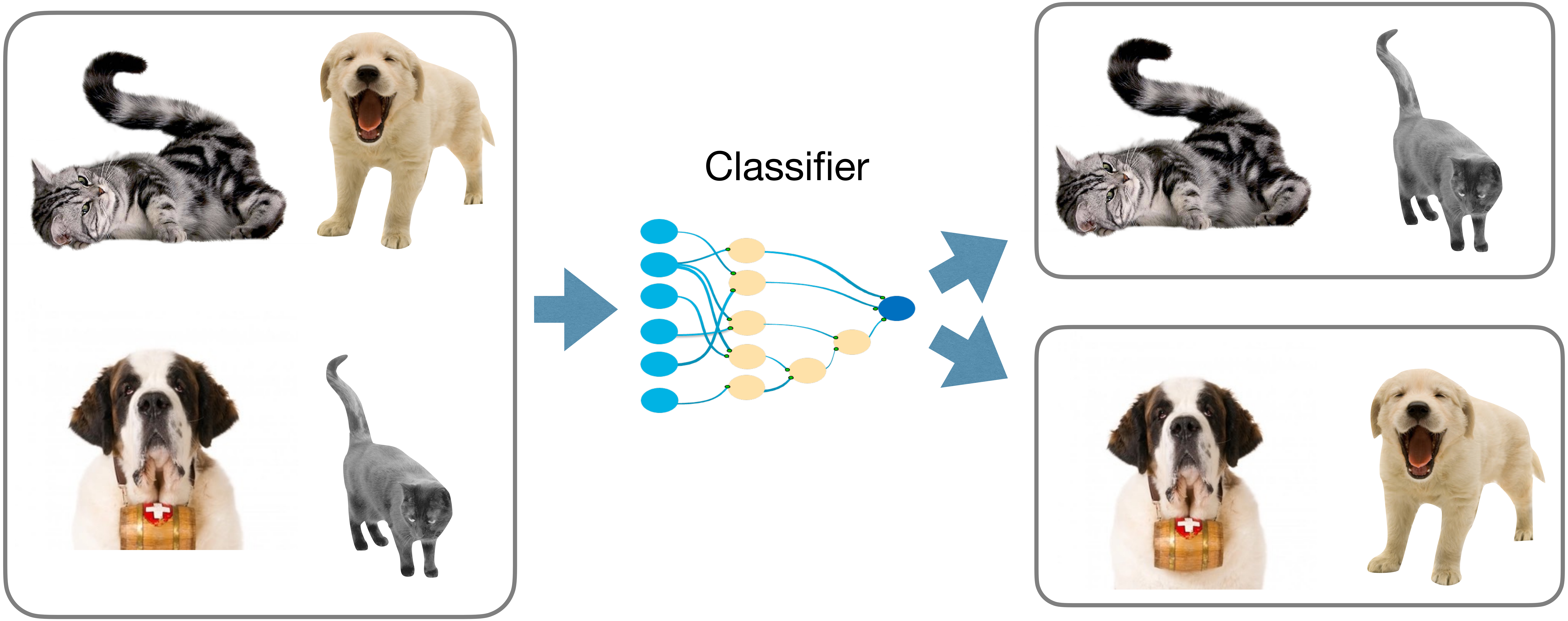
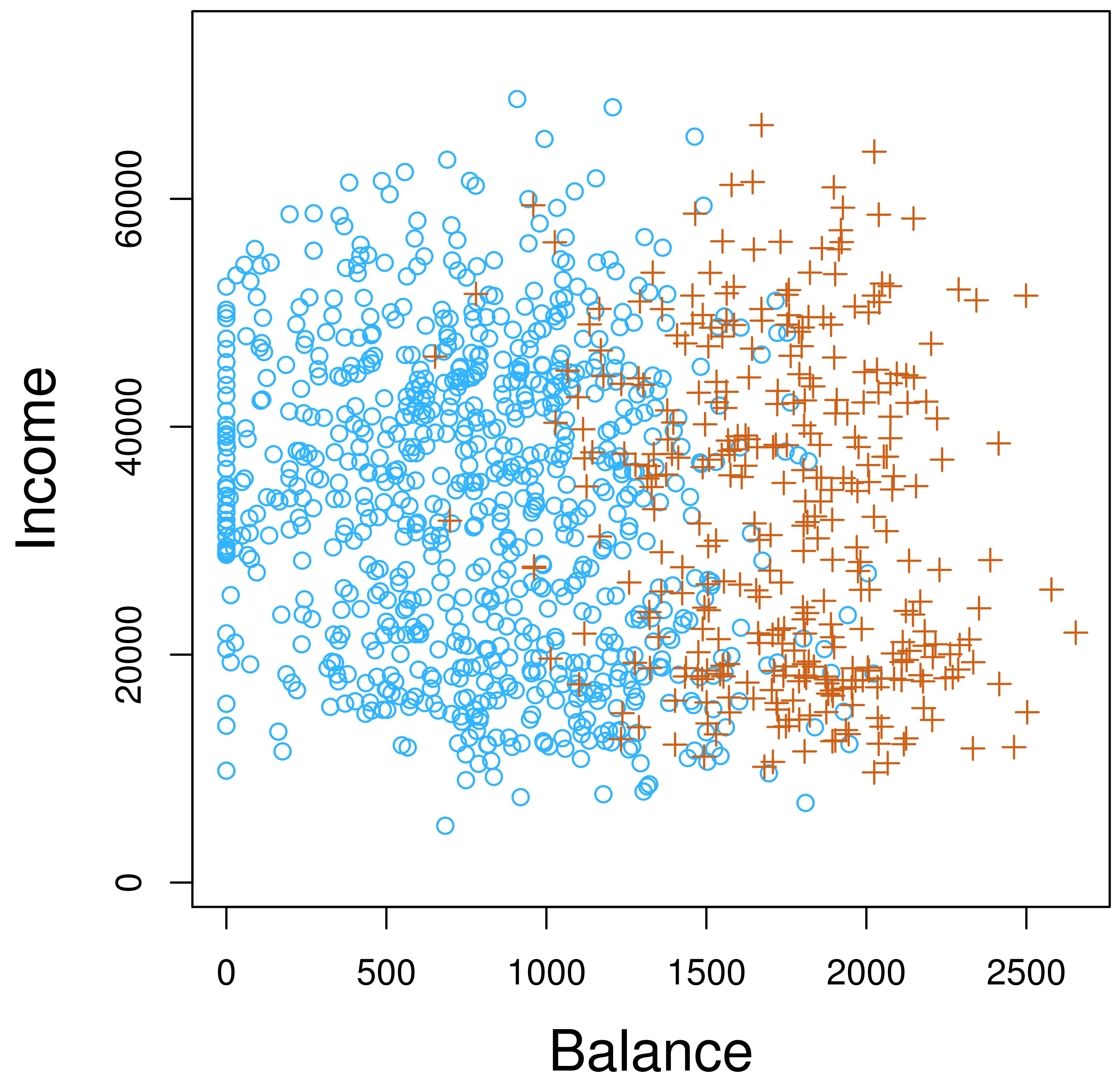


Image classification



Credit Card Default



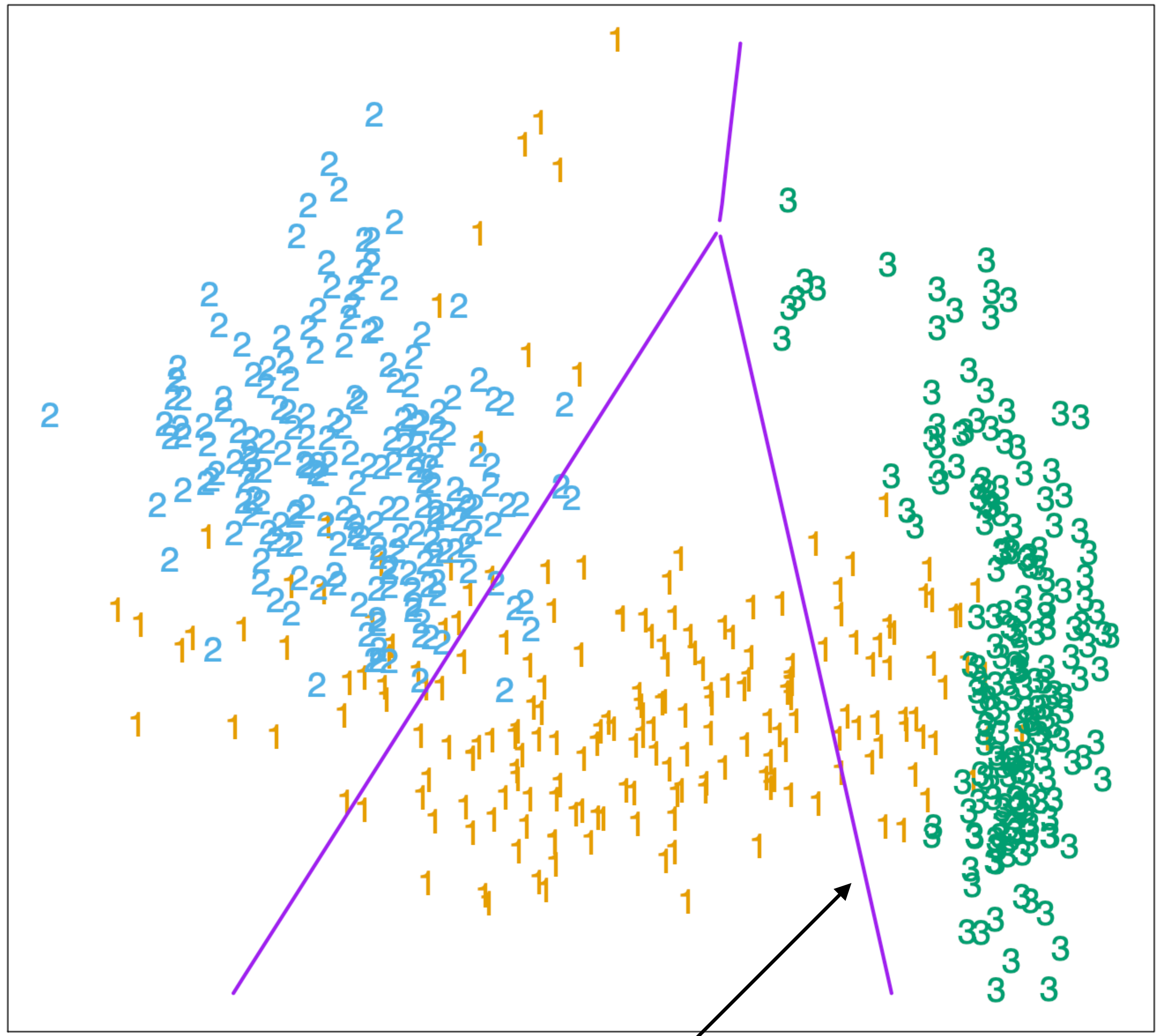
+ individual who defaulted on their credit card payments

○ individual who did not

Classifier

A classifier $f : \mathscr{X} \to \mathscr{Y}$ divides the input space into a collection of region belonging to each class

It can be linear

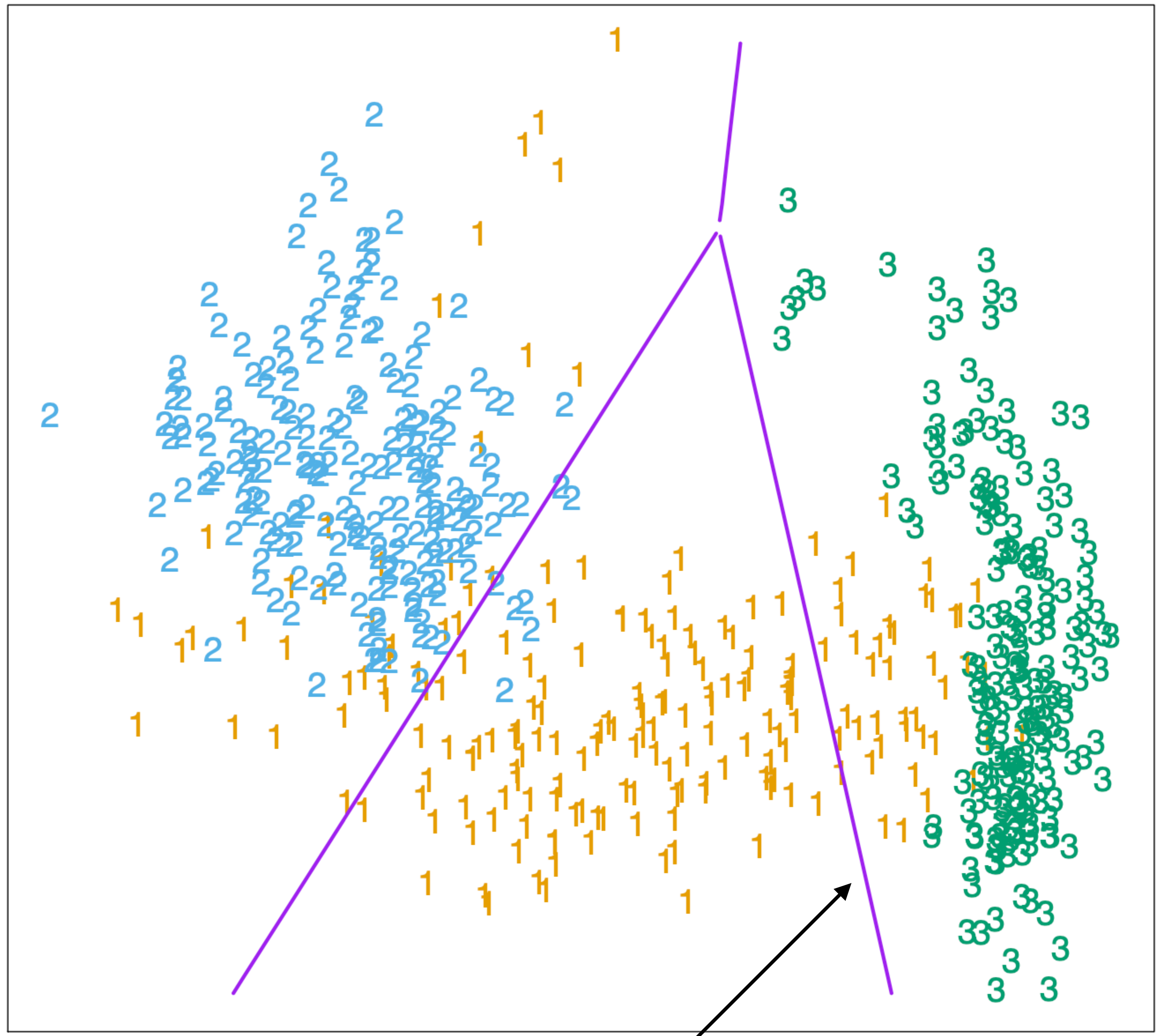


Decision boundary

Classifier

A classifier $f : \mathscr{X} \rightarrow \mathscr{Y}$ divides the input space into a collection of region belonging to each class

It can be linear

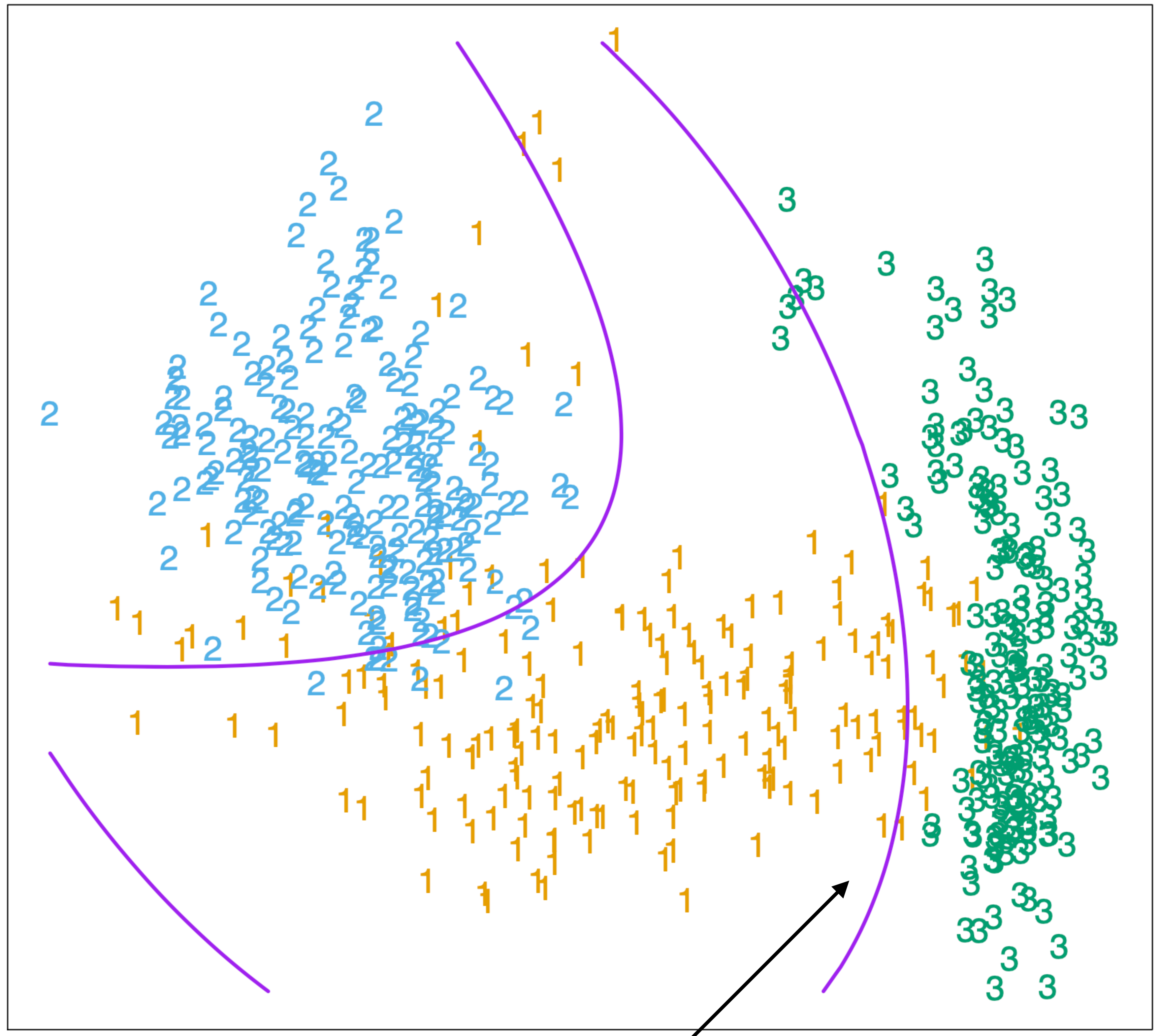


Linear Decision boundary

Classifier

A classifier $f : \mathscr{X} \rightarrow \mathscr{Y}$ divides the input space into a collection of region belonging to each class

It can also be nonlinear



Nonlinear Decision boundary

Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(x, y) \in \mathcal{X} \times \{0,1\} \subset \mathcal{X} \times \mathbb{R}$$

Could we use previously seen regression methods to solve it?

Training Data $\longrightarrow$ Least-Squares $\longrightarrow$ Prediction

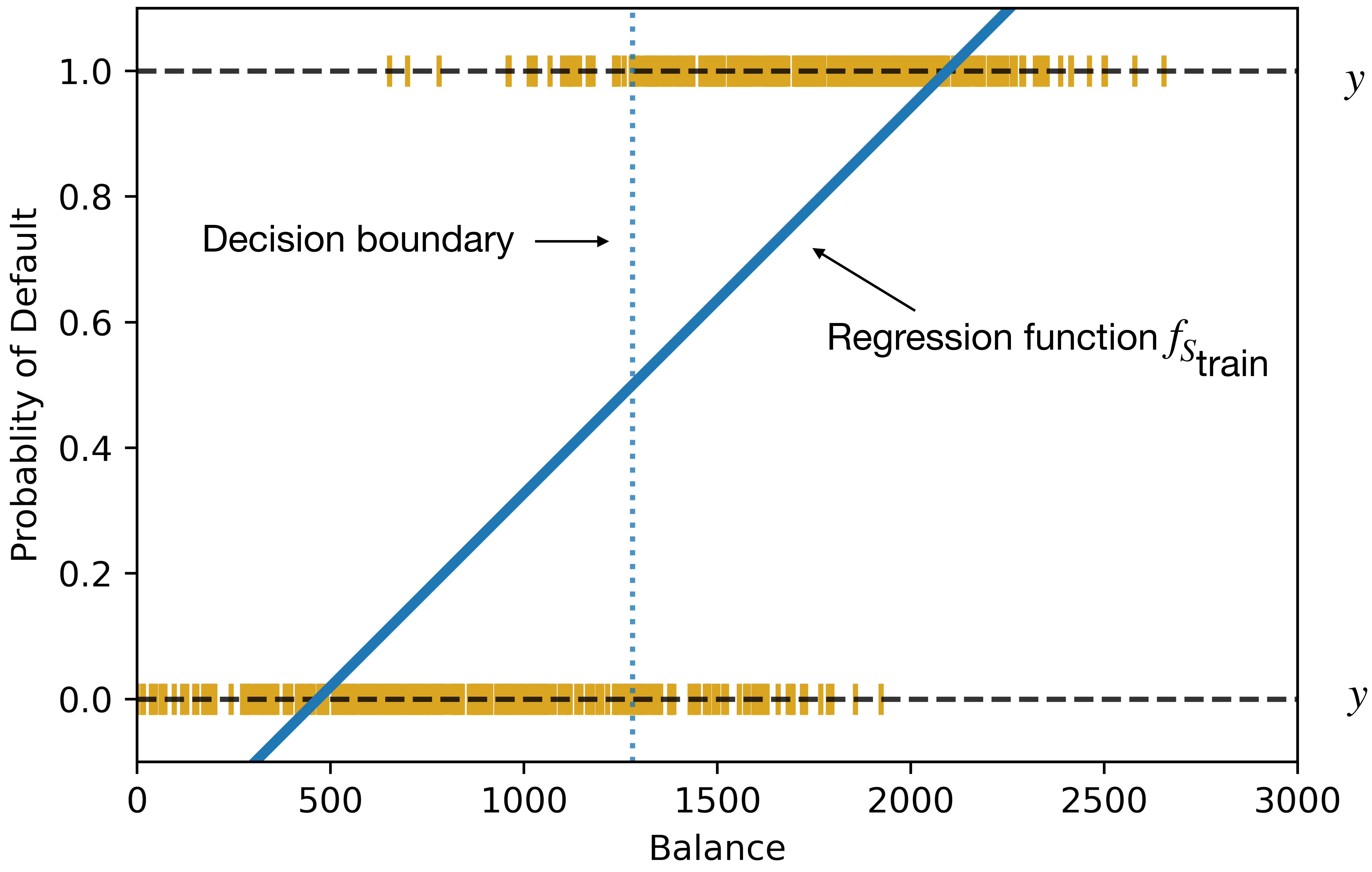
$$S_{\text{train}}: y = \begin{cases} 0 & \text{if } C_1 \\ 1 & \text{if } C_2 \end{cases}$$

$$f_{S_{\text{train}}}$$

$$\begin{cases} C_1 & \text{if } f_{S_{\text{train}}}(x) < 0.5 \\ C_2 & \text{if } f_{S_{\text{train}}}(x) \geq 0.5 \end{cases}$$

Is it a good idea?

Credit-card default problem:



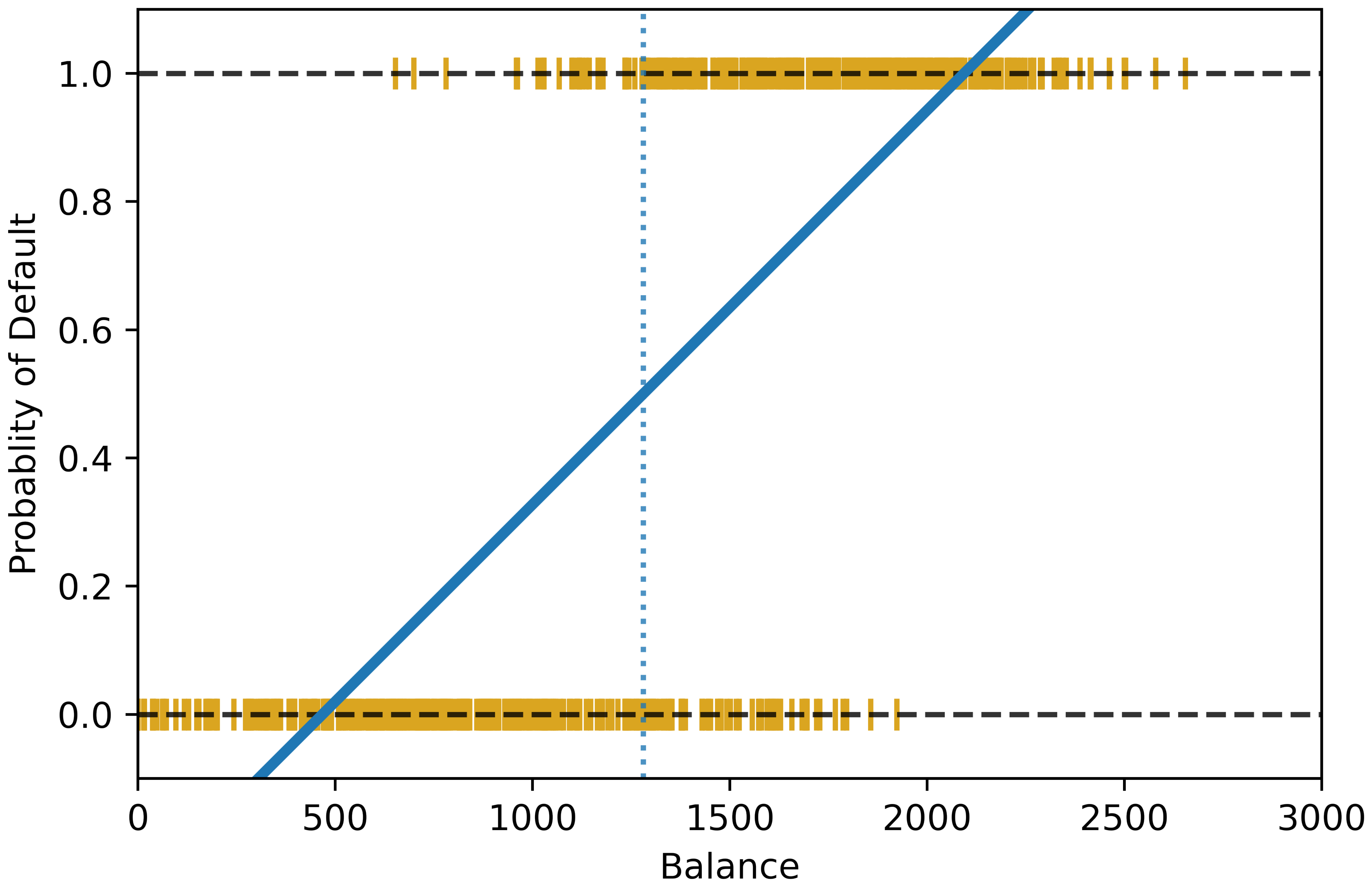
$y = 1$ **if "default"**

$y = 0$ **if "no default"**

We label the output as probability for sake of interpretation

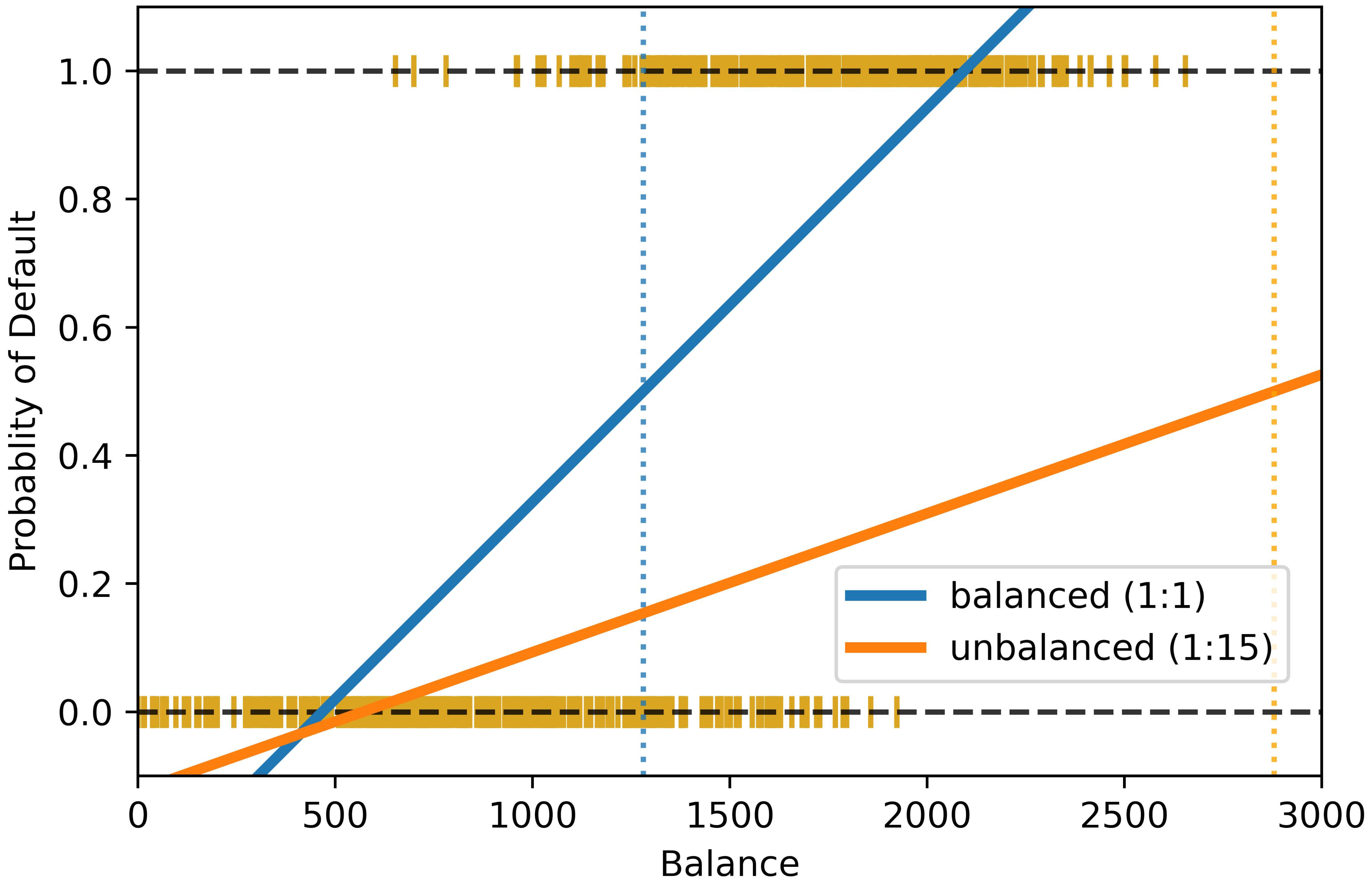
Classification is not just a special form of regression

A. The predicted values are not probabilities (not in [0,1])



Classification is not just a special form of regression

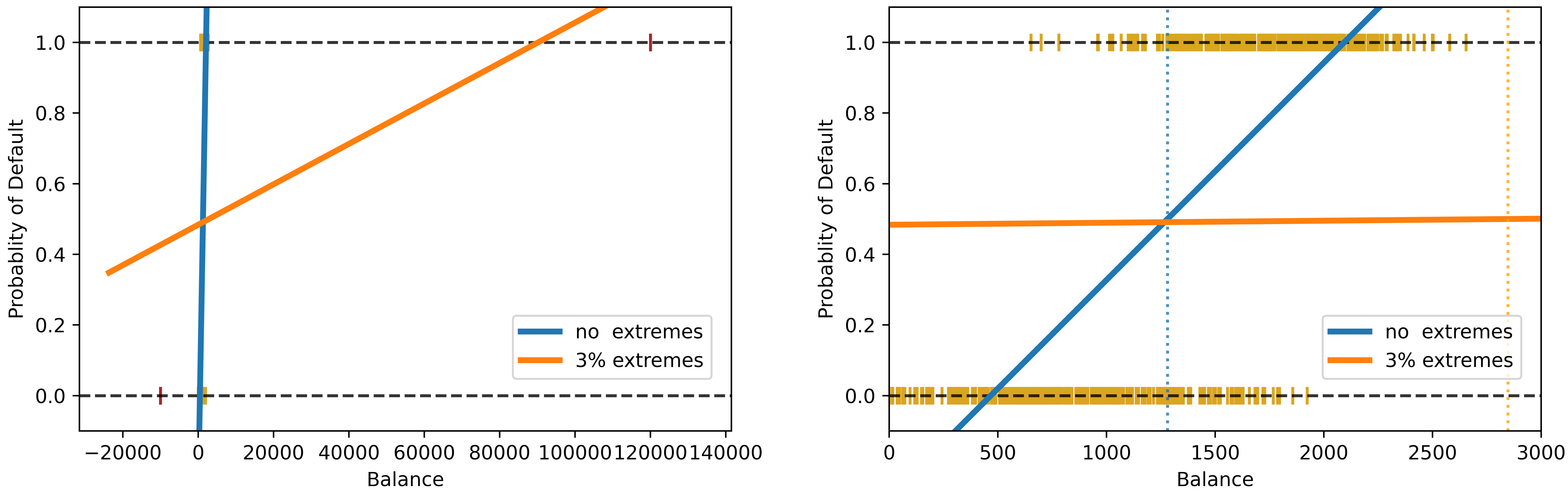
B. Sensitivity to unbalanced data



The position of the line depends crucially on how many points are in each class

Classification is not just a special form of regression

C. Sensitivity to extreme values:

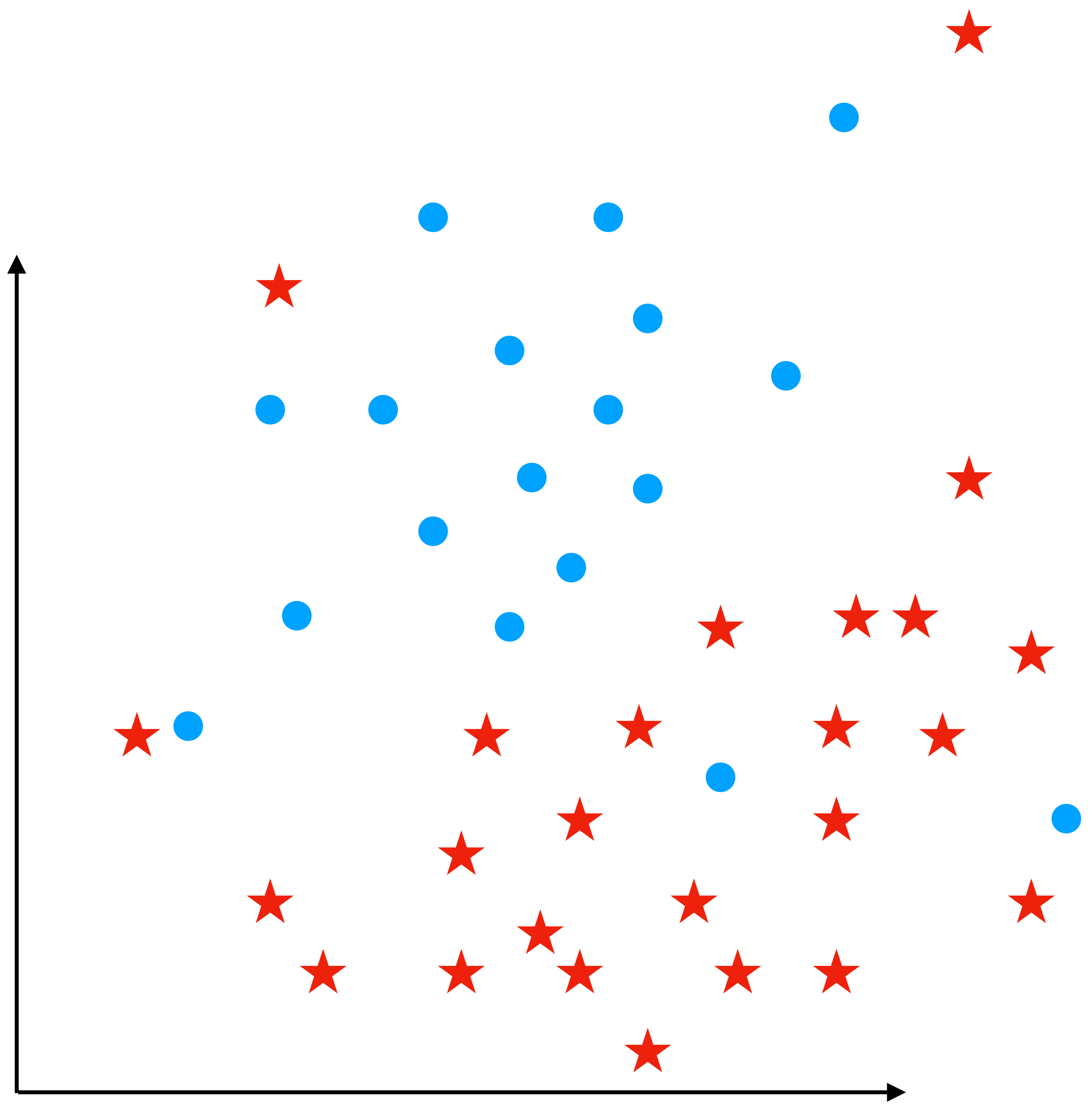


The position of the line depends crucially on where the points lie

Why: the square loss we used for regression is not suitable for classification

How to perform classification?

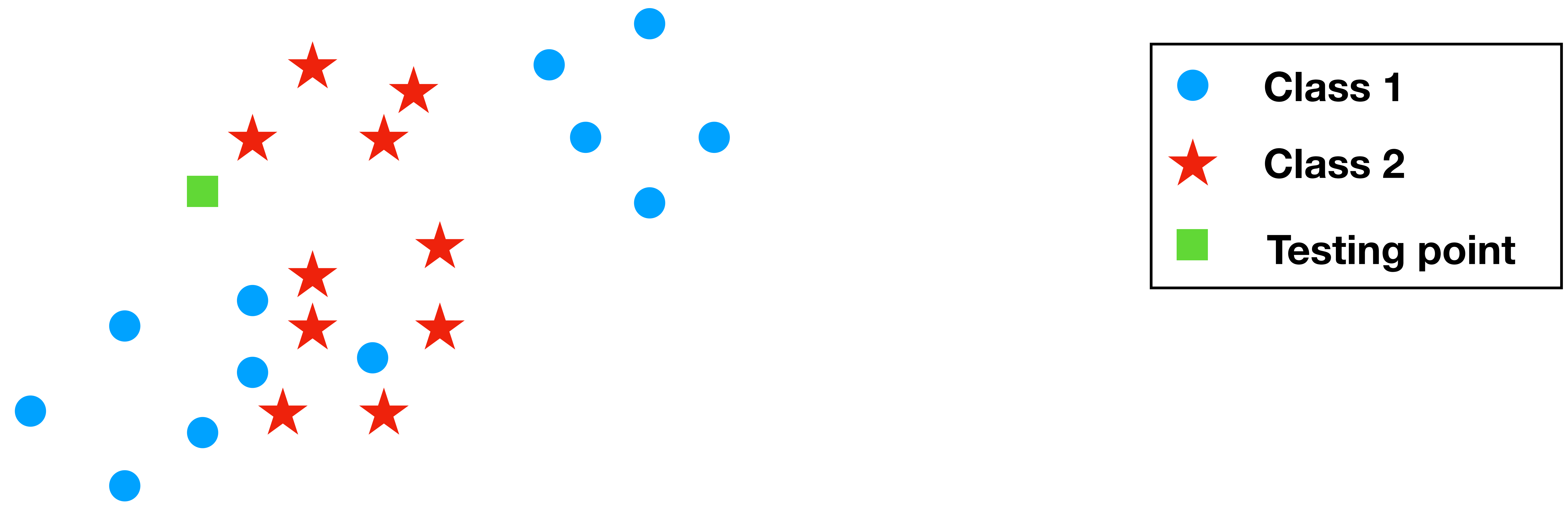
- A lot of different approaches have been developed
- We will not detail them exhaustively today
- Rather we will provide quick introductions



- Fundamental task of classification:

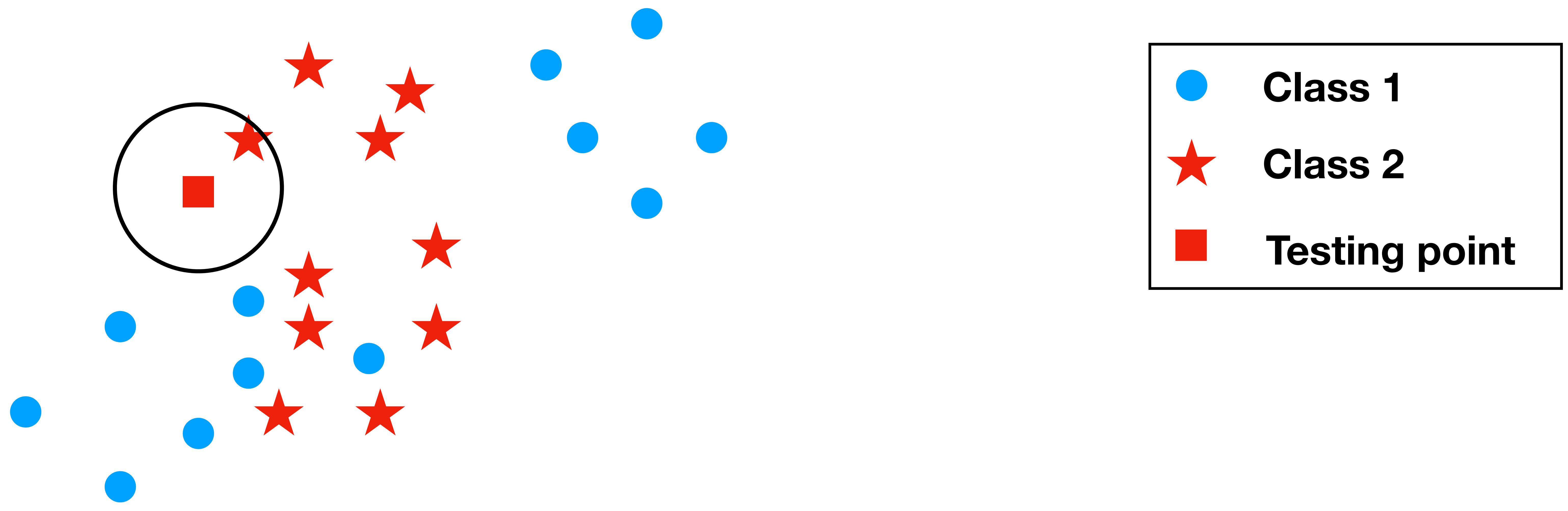
separate the space into various decision regions

Nearest Neighbor



Nearest Neighbor

Assume that nearby points are likely to have similar label

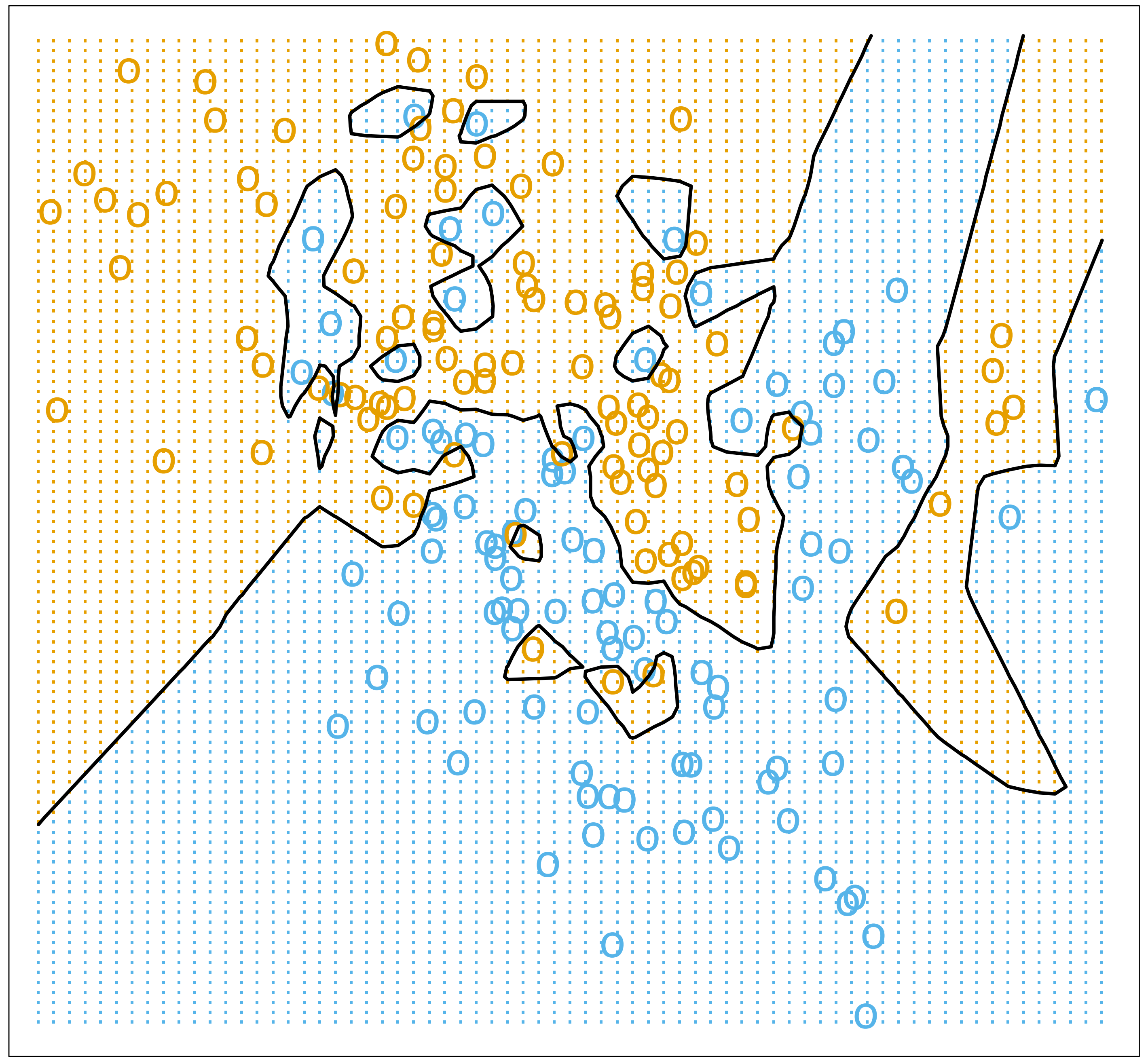


You assign the label of the closest point in your training set.

Nearest Neighbor

Pros:

- **No optimization** or training
- **Easy** to implement
- Works well in **low dimension** where you can get some very complex decision boundaries

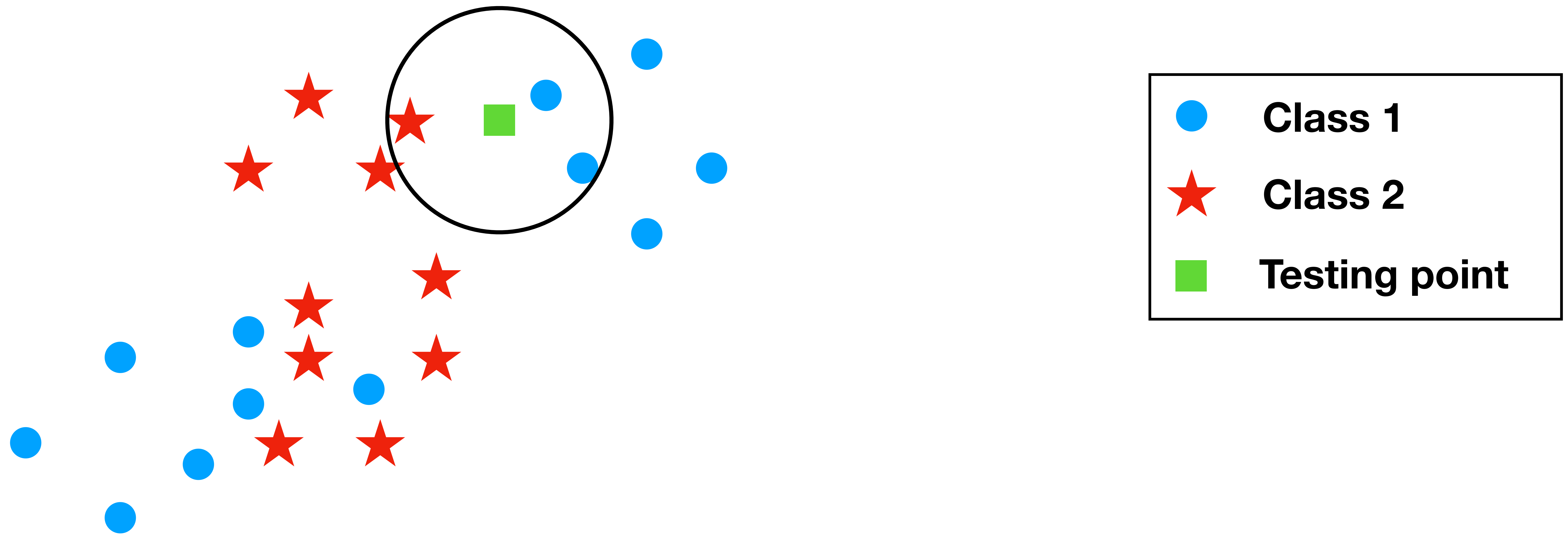


Cons:

- **Slow** at query time
- Bad for high dimensional data
- Choice of **local distance** is crucial

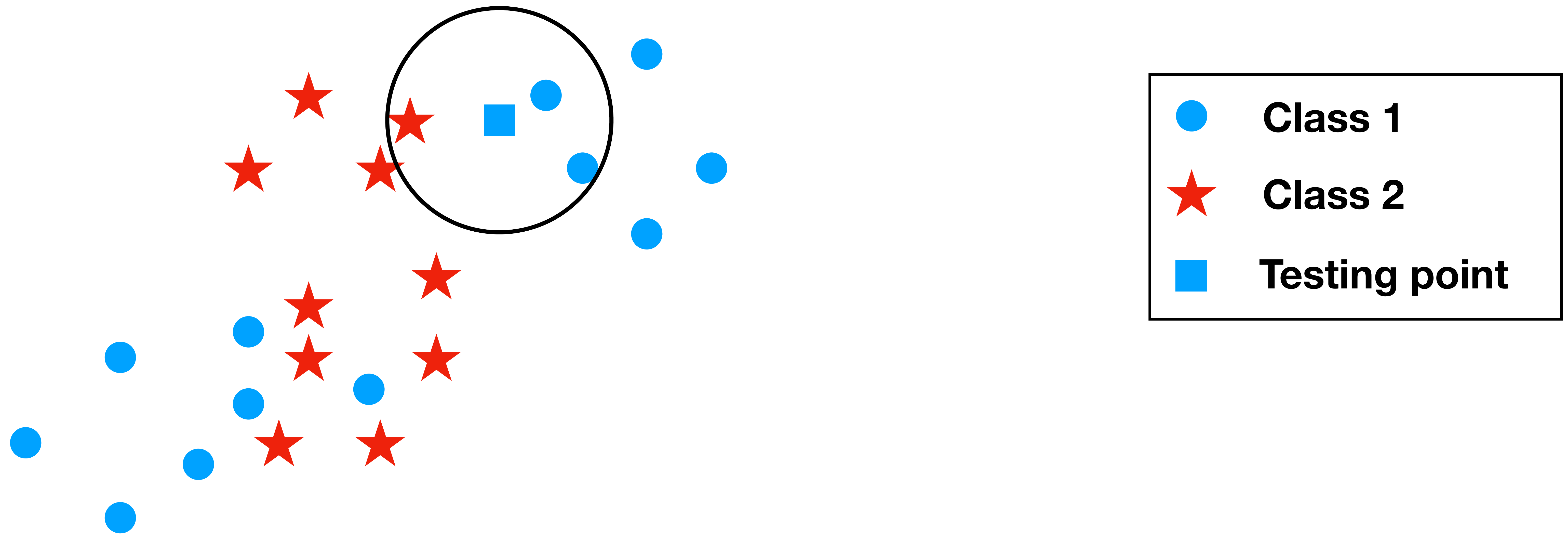
k-Nearest Neighbor

A new point X is classified by a **majority vote among the k-nearest neighbor of** X



k-Nearest Neighbor

A new point X is classified by a **majority vote among the k-nearest neighbor of** X

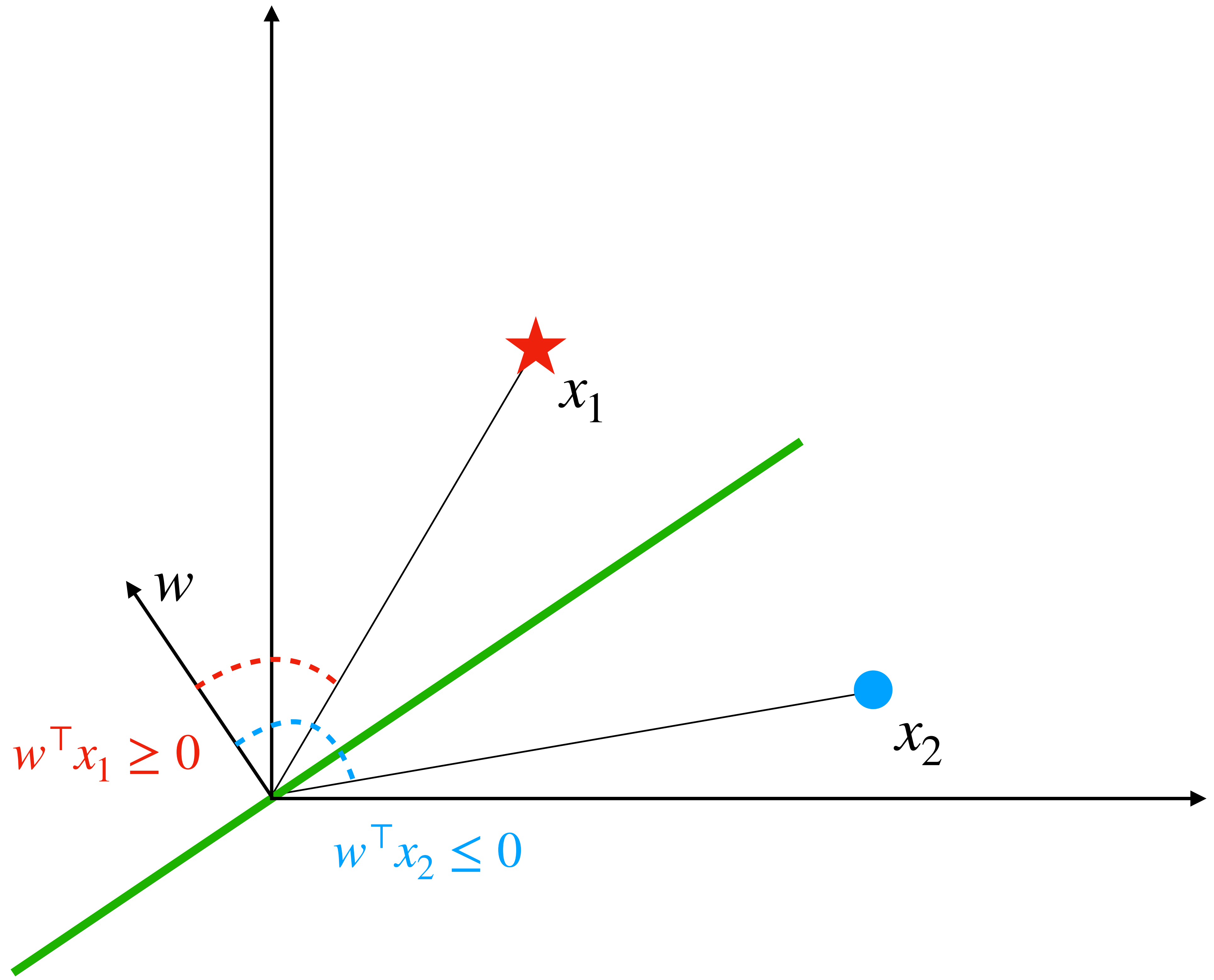


Generalization: smoothing kernels; weighted linear combination of elements

Linear Decision boundaries

Assume we restrict ourself to linear decision boundaries (hyperplane):

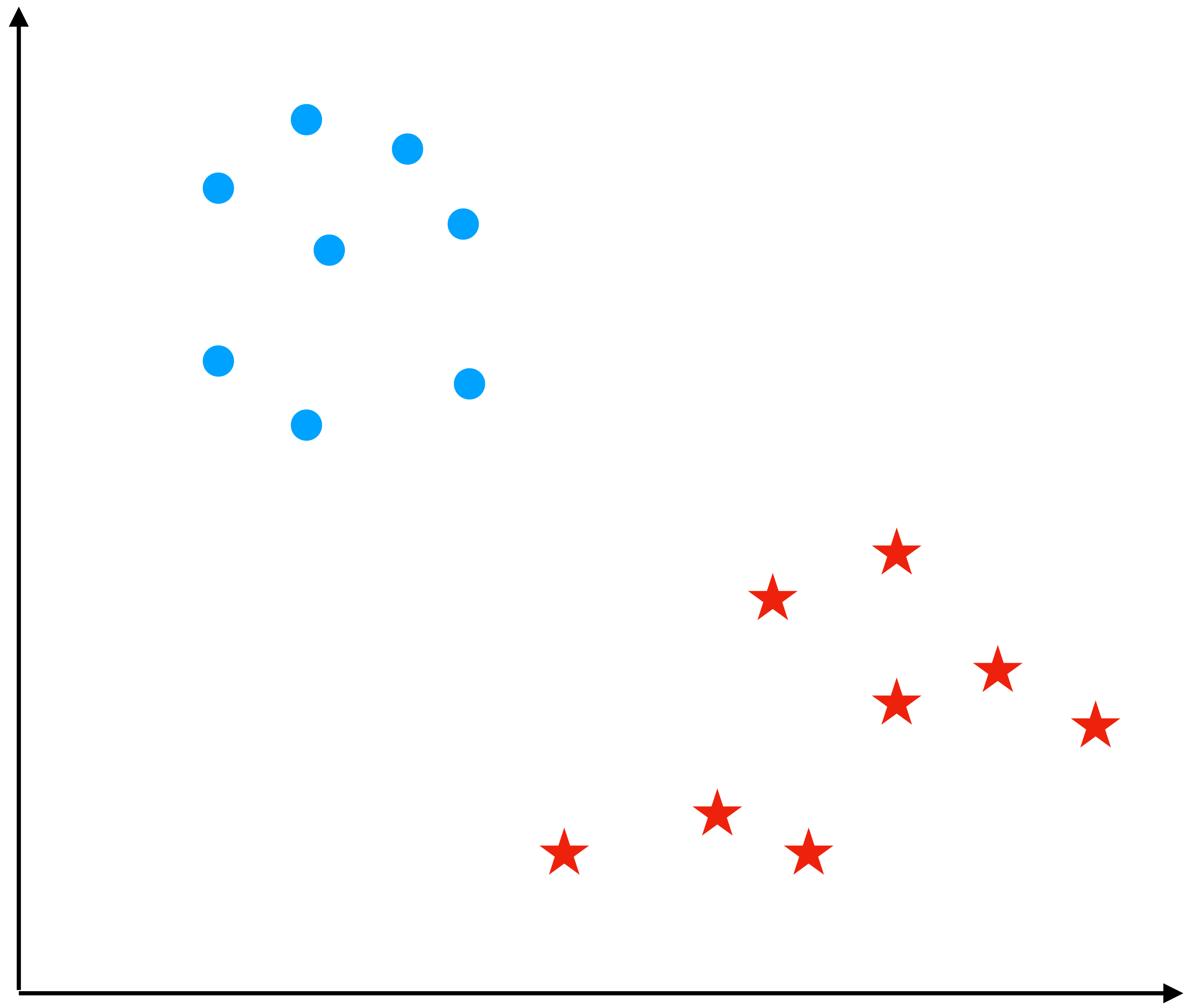
➡ Prediction: $g(x) = \mathrm{sign}(x^\top w)$



Separating hyperplane

Assume we restrict ourself to linear decision boundaries (hyperplane):

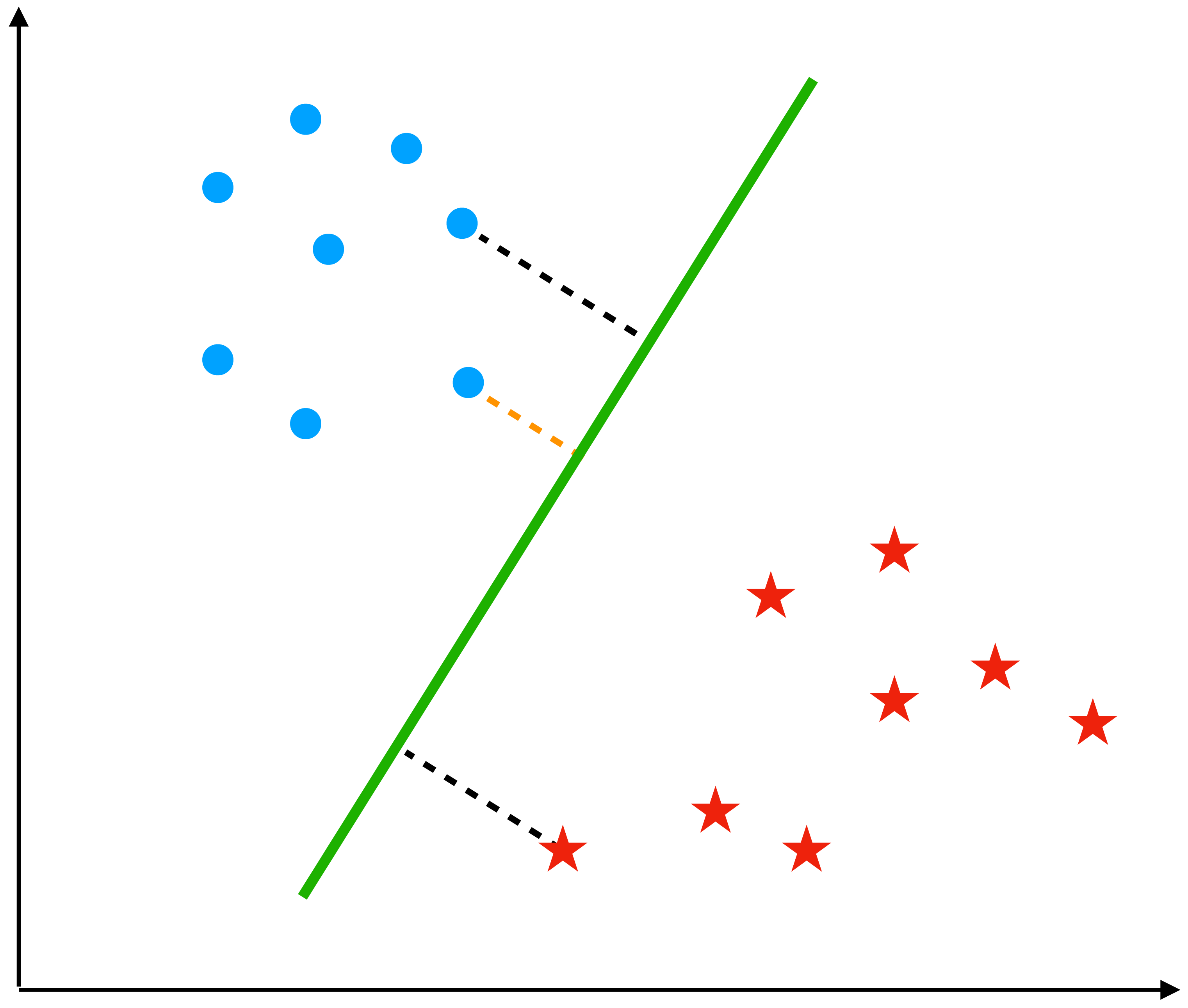
Assume the data are linearly separable, i.e, it exists a separating hyperplane



Which separating hyperplane would you pick?

Margin

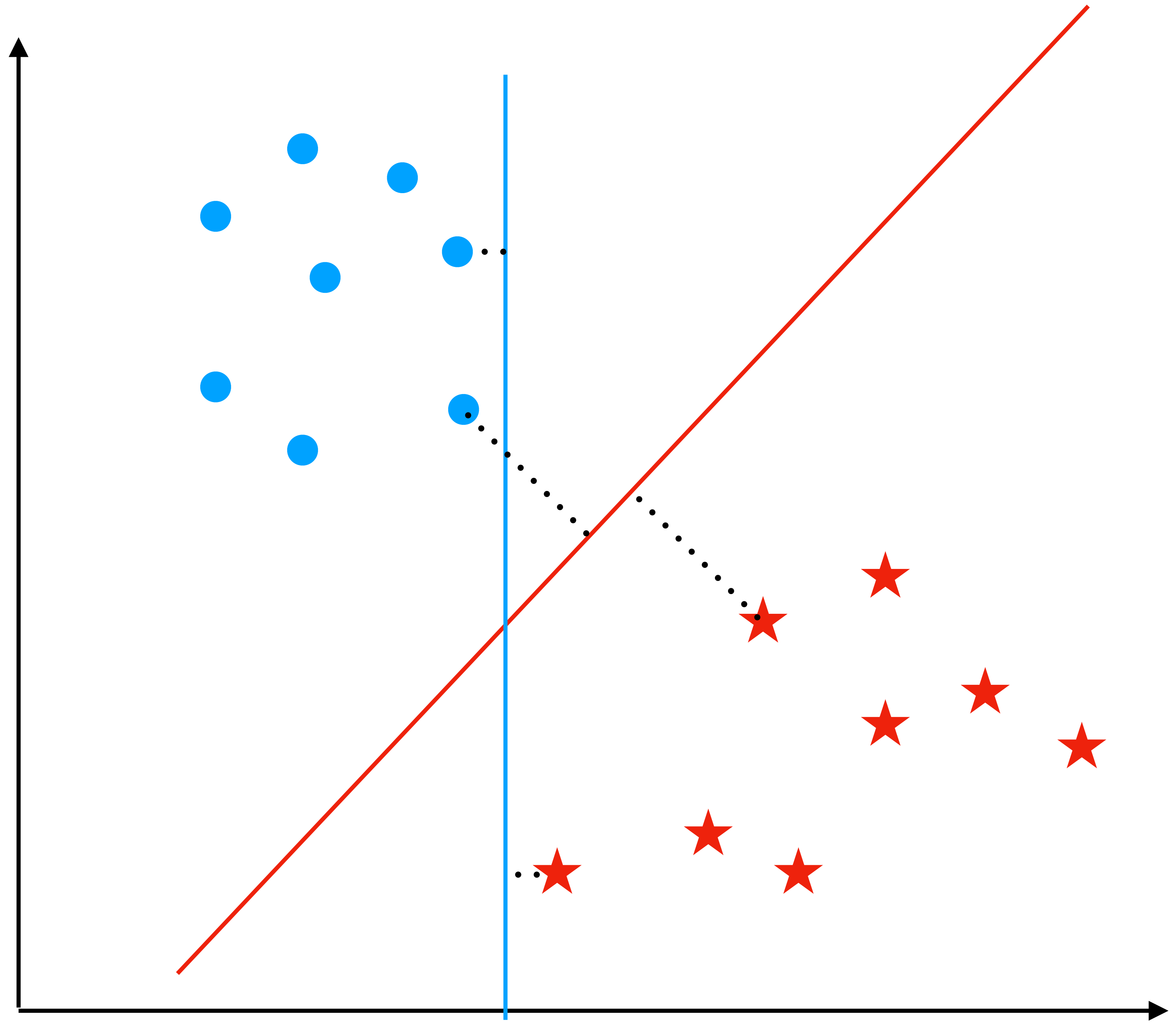
Key concept: The margin is the distance between the hyperplane and the closest point



➡ Take the one with the largest margin!

Max-margin separating hyperplane

Pick the hyperplane which maximizes the margin

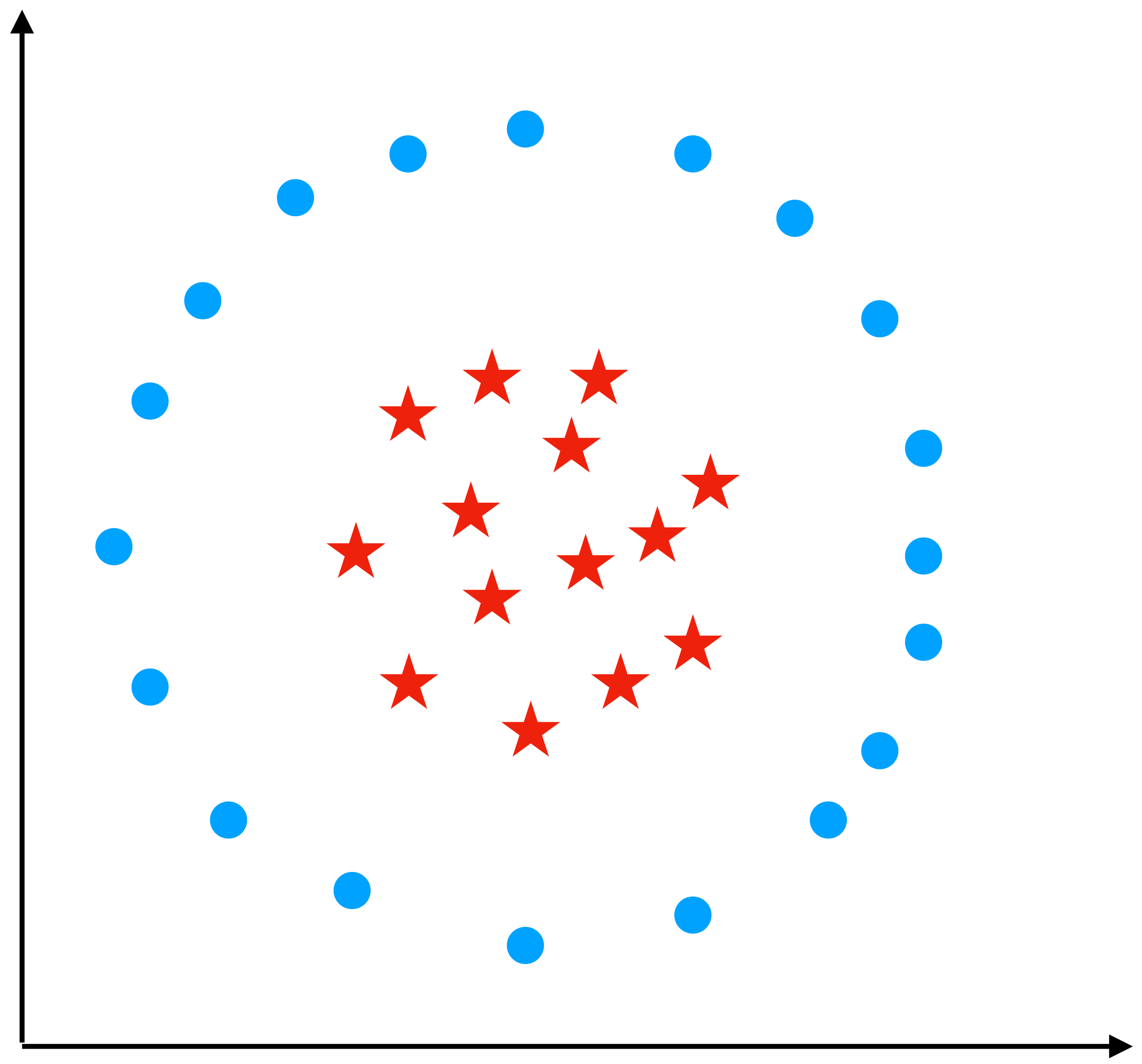


Why: If we slightly change the training set, the number of misclassification will stay low

➡ It will lead us to support vector machine (SVM) and logistic regression

Non linear classifier

- Linear decision boundaries will not always works.
- Features augmentation (X, X^2, X^3, X^4)
- Kernel Method



Do we still have time?

A little bit of theory

$$(X, Y) \sim \mathscr{D} \text{ with } X \in \mathscr{X}, Y \in \mathscr{Y} = \{0,1\}$$

Loss function:

$$\ell(y, y') = 1_{y \neq y'} = \begin{cases} 1 & \text{if } y \neq y' \\ 0 & \text{if } y = y' \end{cases}$$

True risk for the classification:

$$L_{\mathscr{D}}(g) = \mathbb{E}_{\mathscr{D}}[1_{Y \neq g(X)}] = \mathbb{P}_{\mathscr{D}}[Y \neq g(X)]$$

classification error

probability of making an error

Bayes classifier

What is the **optimal performance**, regardless of the finiteness of the training data?

Def: The classifier $g_* = \arg\min_g L_{\mathscr{D}}(g)$ is called the **Bayes classifier**

Claim:

$$g_*(x) = \arg\max_{y \in \{0,1\}} \mathbb{P}(Y = y \,|\, X = x)$$

Proof of the Bayes classifier

Claim 1: $\forall x \in \mathcal{X}, h_*(x) \in \arg\min_{y \in \mathcal{Y}} \mathbb{P}(Y \neq y \,|\, X = x) \implies h_* \in \arg\min_{h:\mathcal{X}\to\mathcal{Y}} L_{\mathcal{D}}(h)$

$$\begin{aligned}
L_{\mathcal{D}}(h) = \mathbb{E}_{X,Y}[1_{Y \neq h(X)}] &= \mathbb{E}_X[\mathbb{E}_{Y|X}[1_{Y \neq h(X)} \,|\, X]] \\
&= \mathbb{E}_X[\mathbb{P}(Y \neq h(X) \,|\, X)] \\
&\geq \mathbb{E}_X[\min_{y \in \mathcal{Y}} \mathbb{P}(Y \neq y \,|\, X)] \\
&= \mathbb{E}_X[\mathbb{P}(Y \neq h_*(X) \,|\, X)] = \mathbb{E}_{X,Y}[1_{Y \neq h_*(X)}] = L_{\mathcal{D}}(h_*)
\end{aligned}$$

Claim 2: $g_*(x) = \arg\min_{y \in \mathcal{Y}} \mathbb{P}(Y \neq y \,|\, X = x)$

$$g_*(x) = \arg\max_{y \in \mathcal{Y}} \mathbb{P}(Y = y \,|\, X = x) = \arg\min_{y \in \mathcal{Y}} \mathbb{P}(Y \neq y \,|\, X = x)$$

Bonus: a good regressor implies a good classifier

For all regression functions $\eta : \mathcal{X} \to \mathbb{R}$ we can define a classifier as

$$\begin{aligned} \mathcal{X} &\to \{0,1\} \\ g_\eta : \ x &\mapsto 1_{\eta(x) \geq 1/2} \end{aligned}$$

<u>Claim:</u>

$$L_{\mathcal{D}}^{\text{classif}}(g_\eta) - L_{\mathcal{D}}^{\text{classif}}(g^*) \leq 2\sqrt{L_{\mathcal{D}}^{\ell_2}(\eta) - L_{\mathcal{D}}^{\ell_2}(\eta^*)}$$

Where $L_{\mathcal{D}}^{\text{classif}}(g_\eta) = \mathbb{E}_{\mathcal{D}}[1_{g(X) \neq Y}]$, $L_{\mathcal{D}}^{\ell_2}(f) = \mathbb{E}_{\mathcal{D}}[(Y - f(X))^2]$ and $\eta_* = \arg\min_\eta L_{\mathcal{D}}^{\ell_2}(\eta)$

➡ If η is good for regression then g_η is good for classification too (converse is not true)