Classification

Machine Learning Course - CS-433
Oct 17, 2023
Nicolas Flammarion

EPFL

Definition of classification

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

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

How:

$S = \{x_n, y_n\}_{n=1}^{N}$ (Discrete $y_n \in \mathcal{Y}$)

$\mathcal{A}$

$f_S = \mathcal{A}(S)$

Classification: relates input to a categorical variable

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

Binary Classification: y can take two values

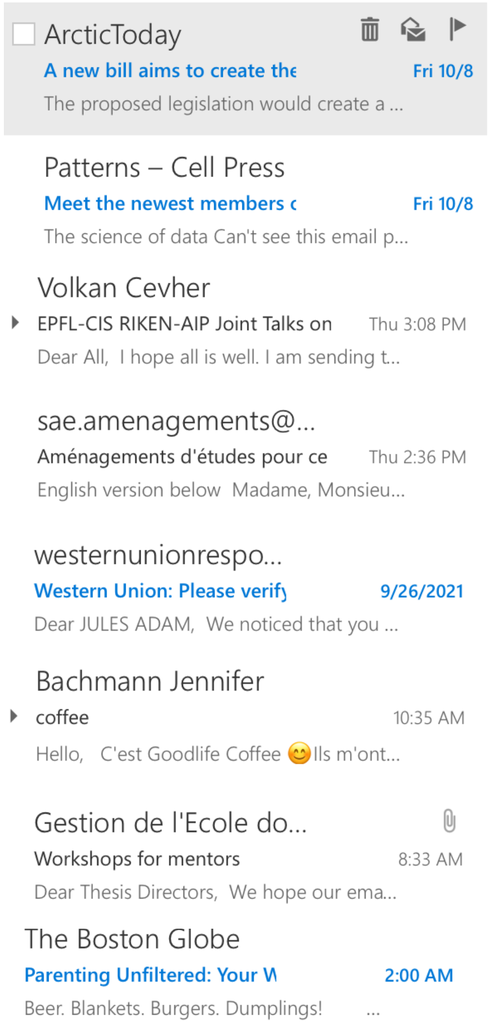
$y \in \{c_1, c_2\}$ where c_i are the class labels. We often use $\{0,1\}$ or $\{-1,1\}$

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

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

no ordering between classes

Spam Detection



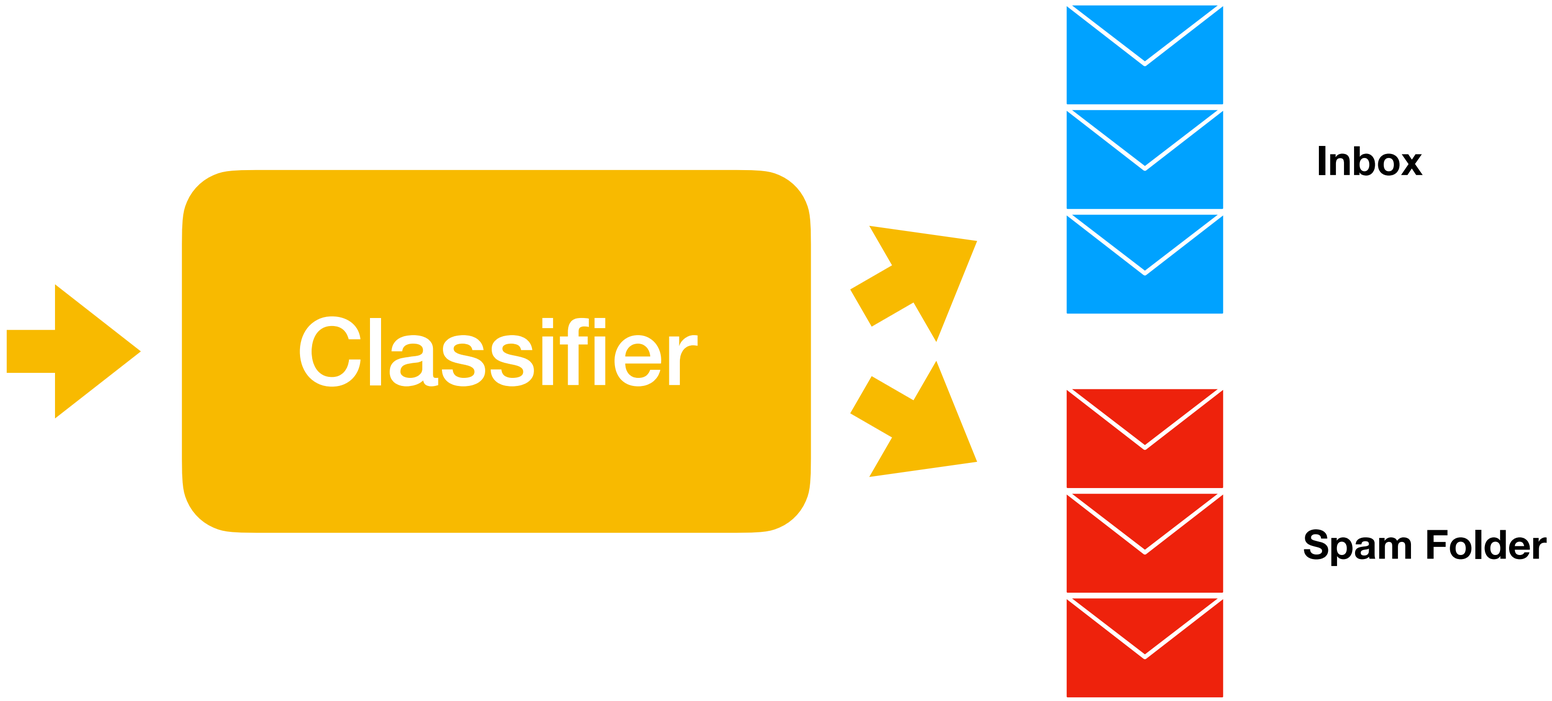
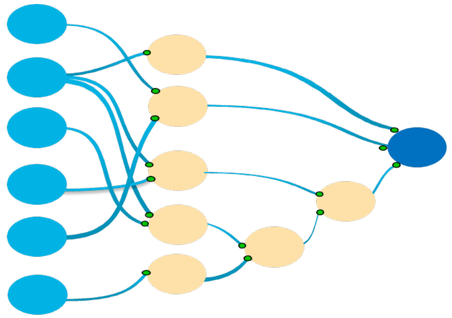
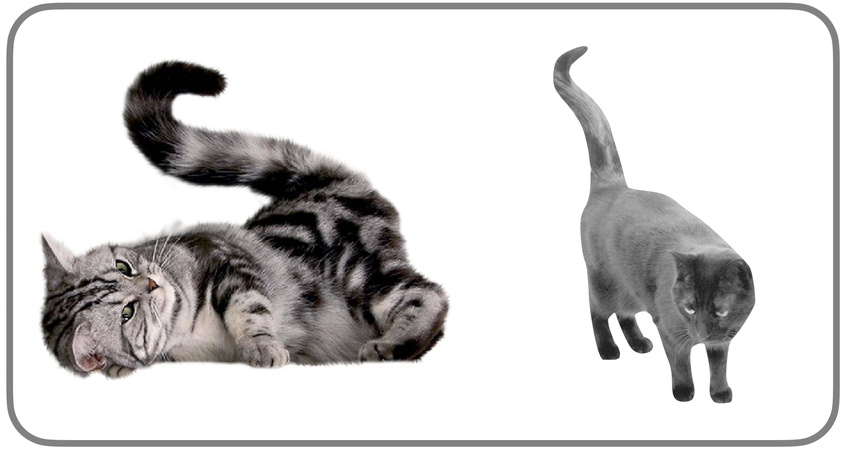


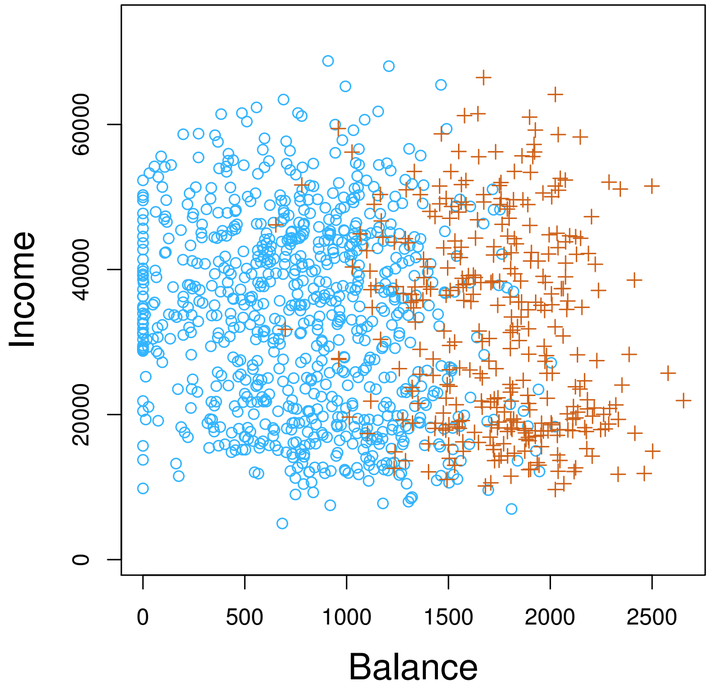
Image classification

Classifier





Credit Card Default

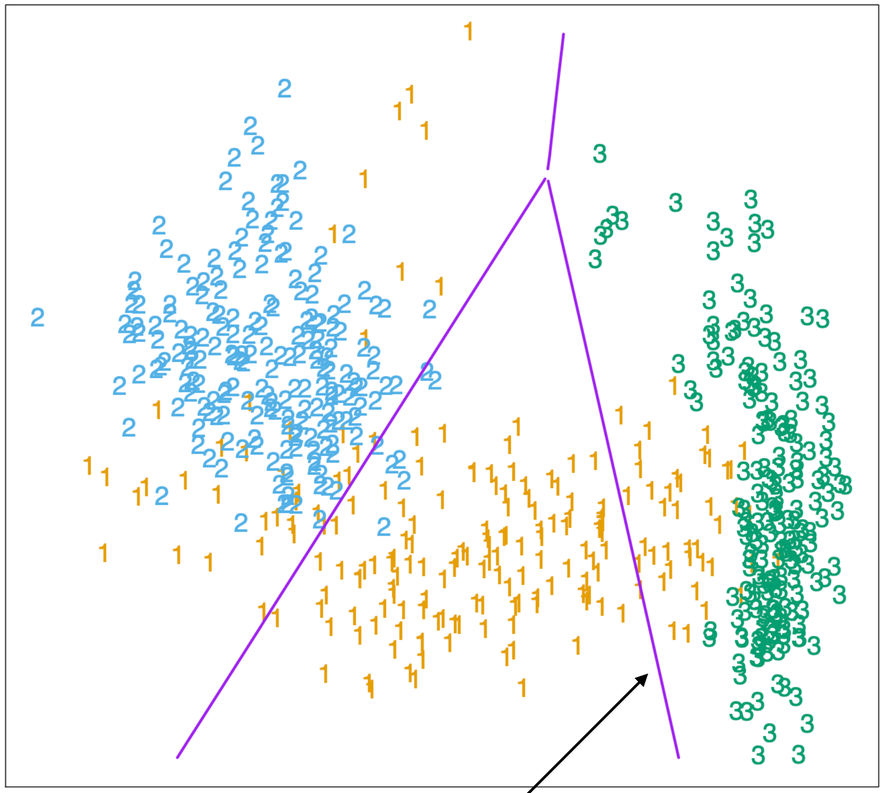


+ individual who defaulted on their credit card payments
○ individual who did not

Classifier

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

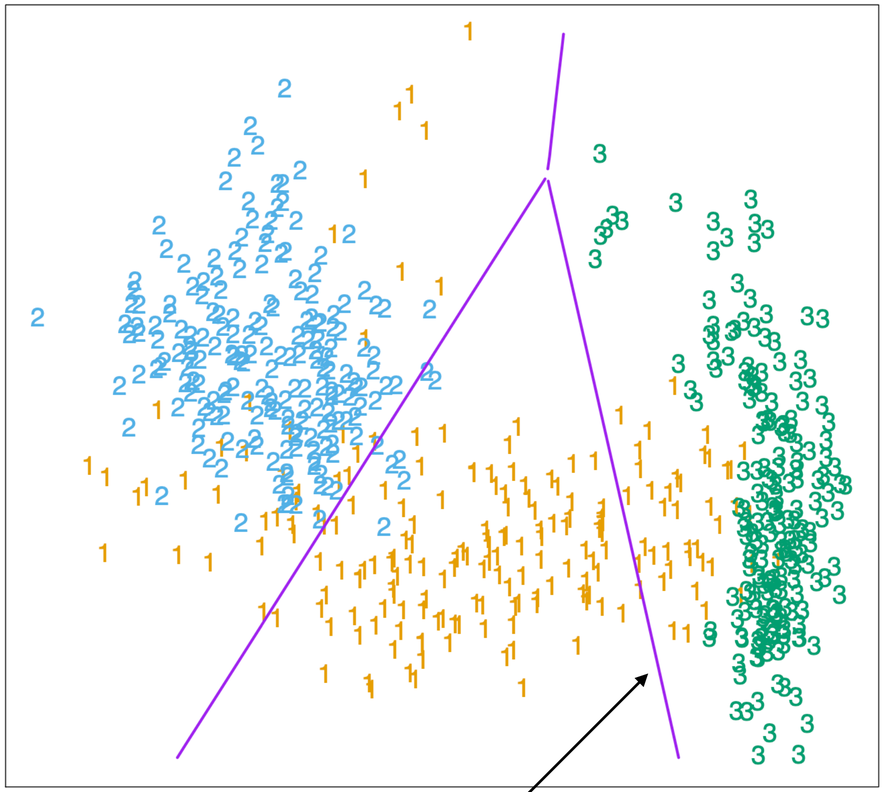
It can be linear



Classifier

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

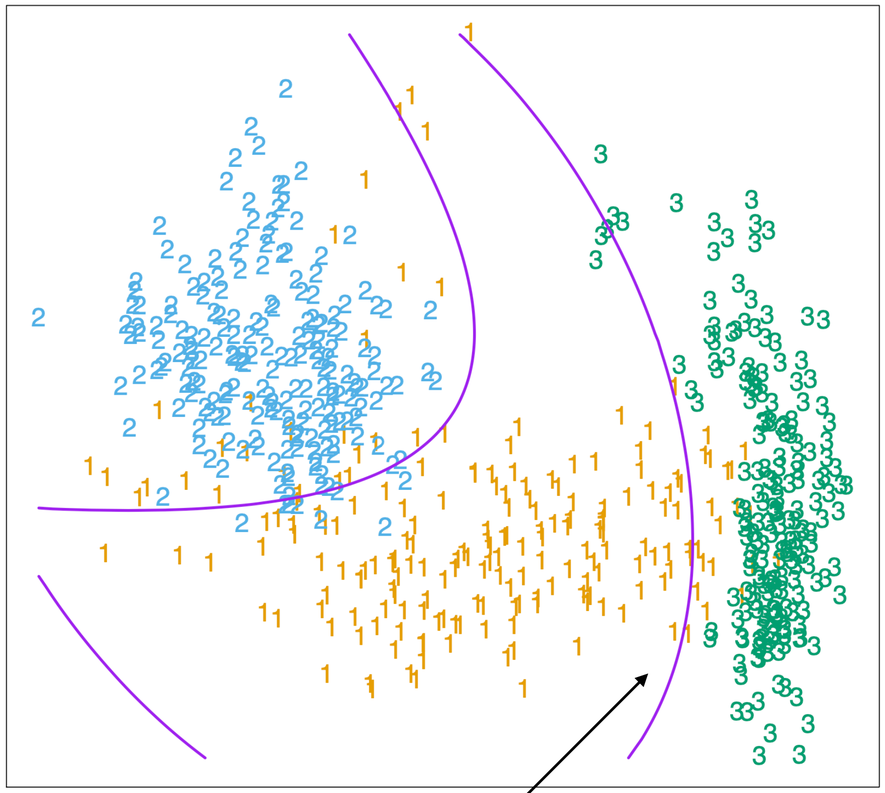
It can be linear



Classifier

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

It can also be nonlinear



Classification: a special case of regression?

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

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

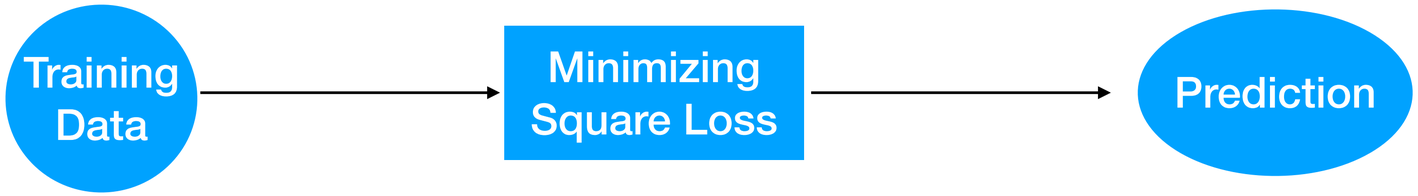
Could we use previously seen regression methods to solve it?

Classification: a special case of regression?

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

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

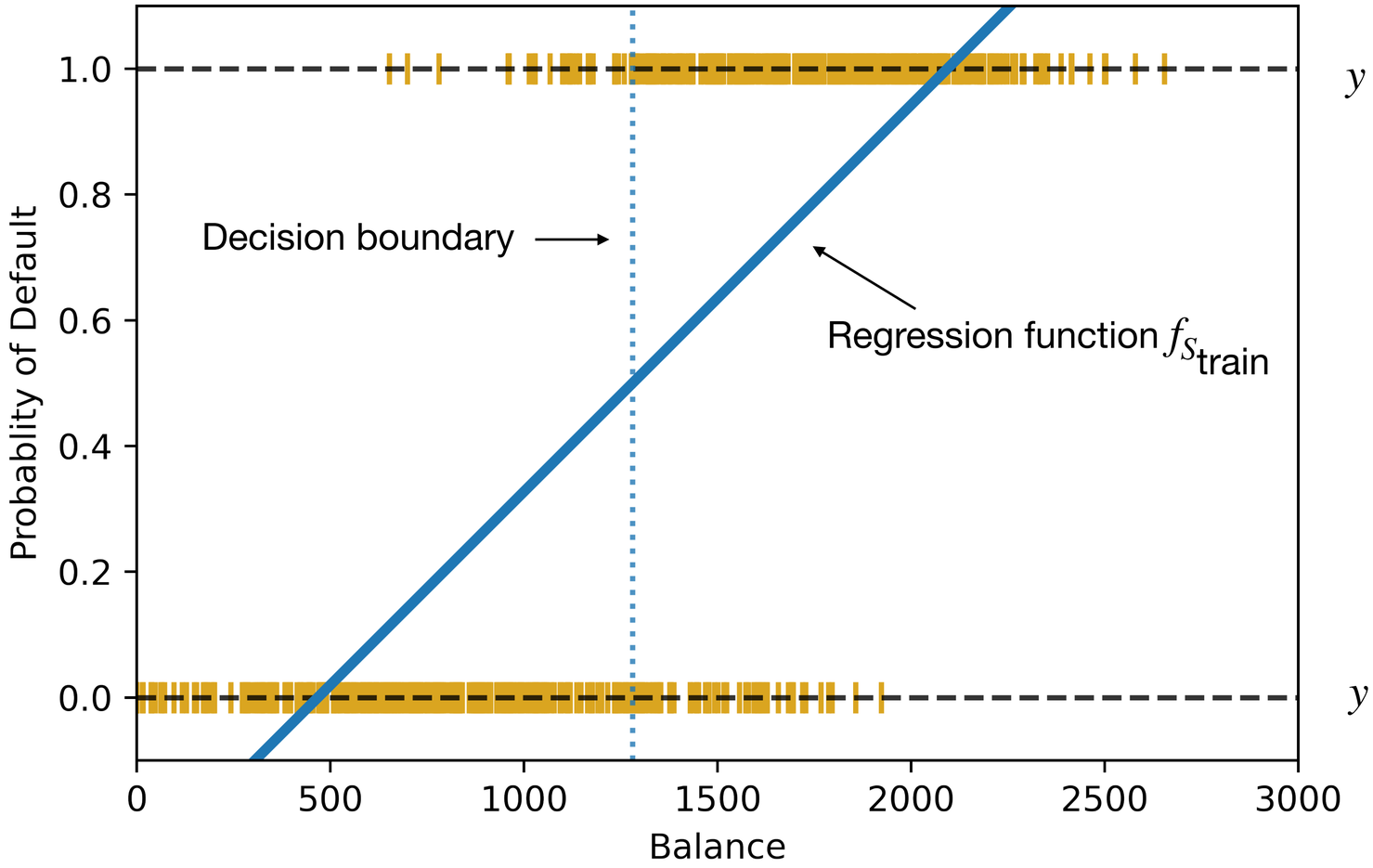
Could we use previously seen regression methods to solve it?



$$S_{\text{train}}: y = \begin{cases} 0 & \text{if } C_1 \\ 1 & \text{if } C_2 \end{cases} \qquad f_{S_{\text{train}}} \qquad \begin{cases} C_1 & \text{if } f_{S_{\text{train}}}(x) < 0.5 \\ C_2 & \text{if } f_{S_{\text{train}}}(x) \geq 0.5 \end{cases}$$

Is it a good idea?

Credit-card default problem:



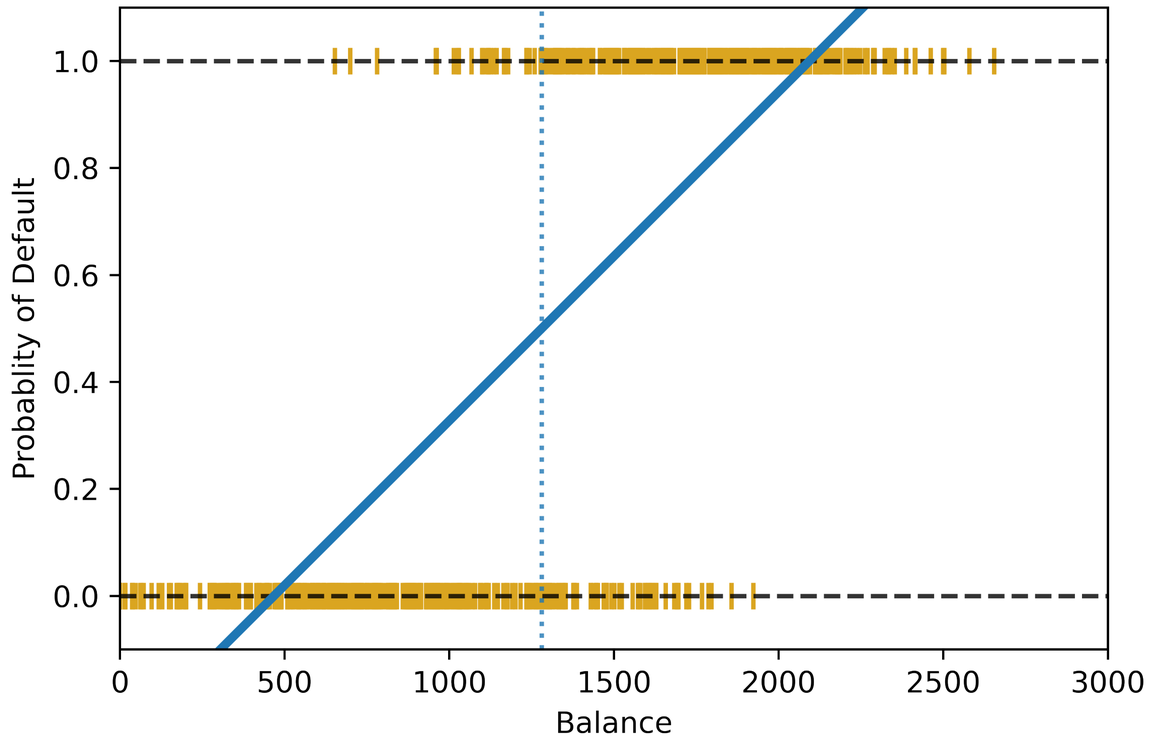
$y = 1$ **if "default"**

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

We label the output as probability for sake of interpretation

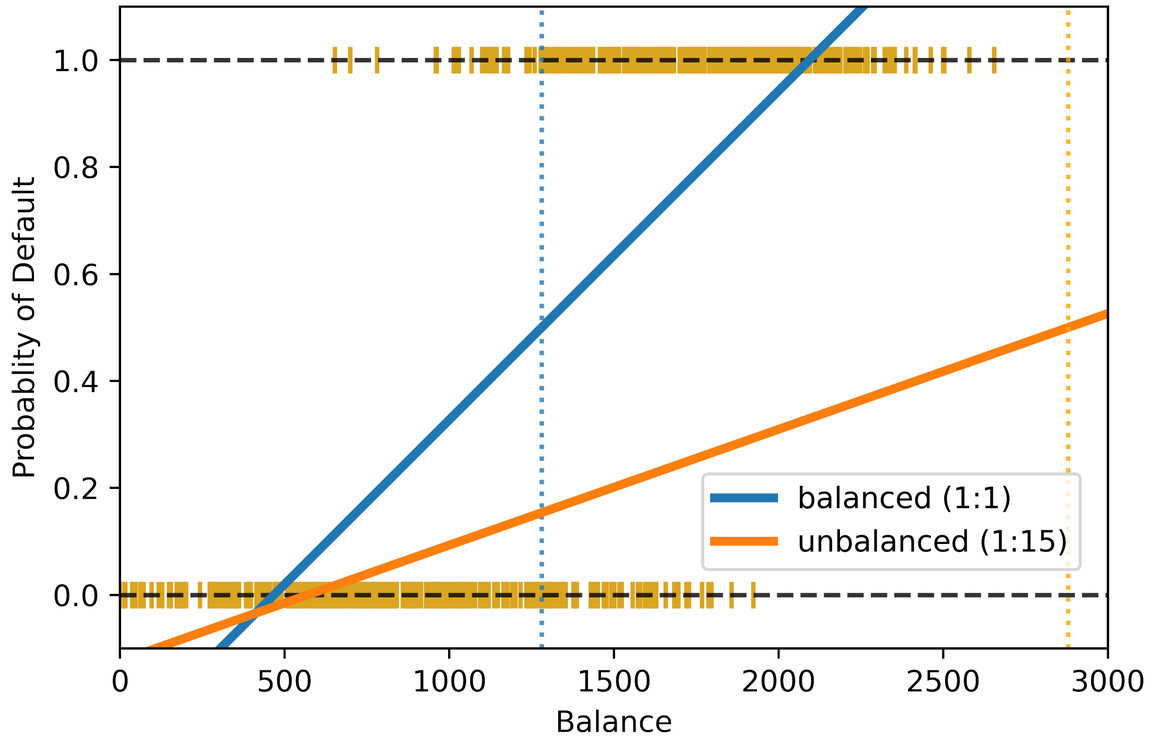
Classification is not just a special form of regression

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



Classification is not just a special form of regression

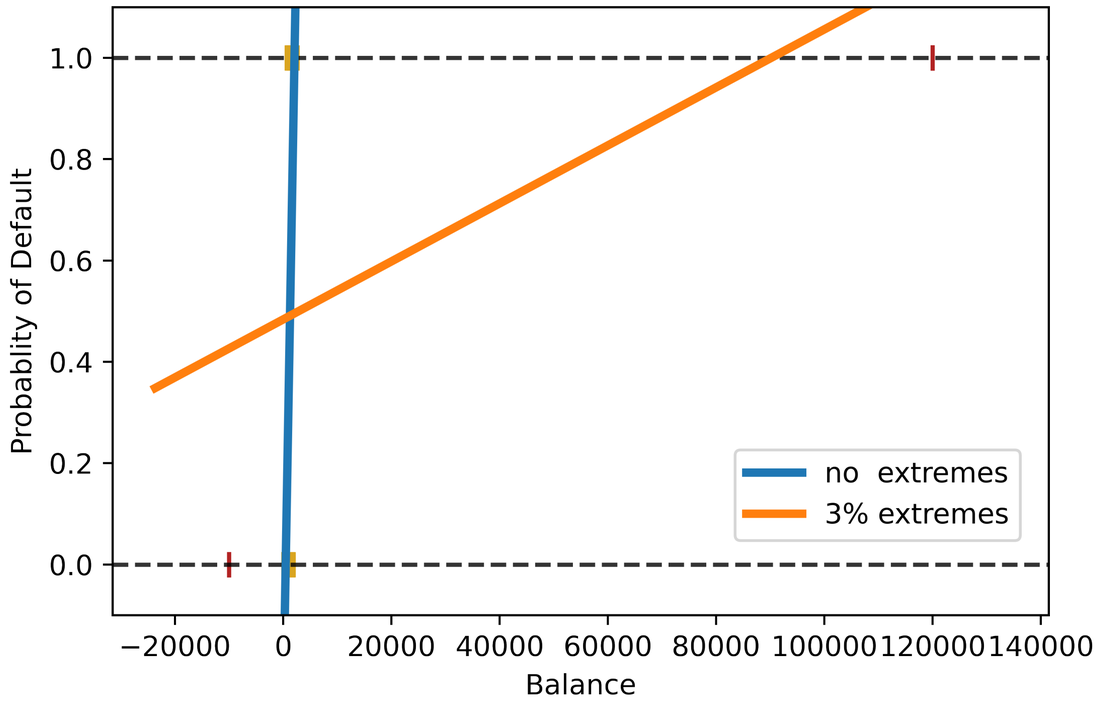
B. Sensitivity to unbalanced data

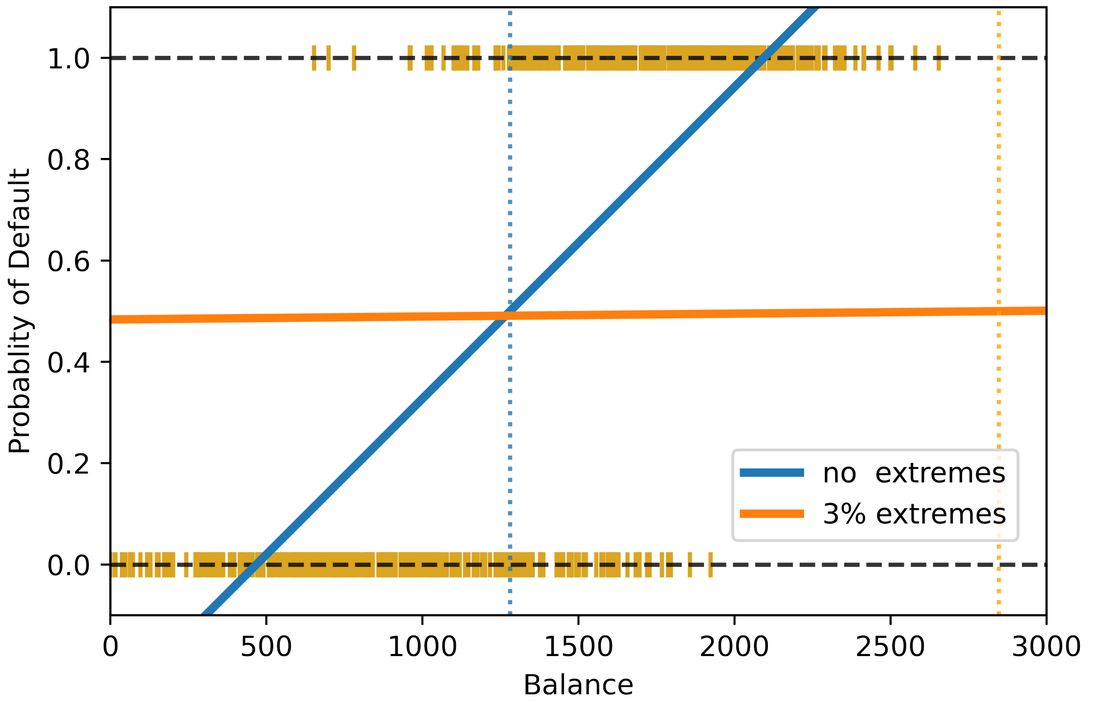


The position of the line depends crucially on how many points are in each class

Classification is not just a special form of regression

C. Sensitivity to extreme values:



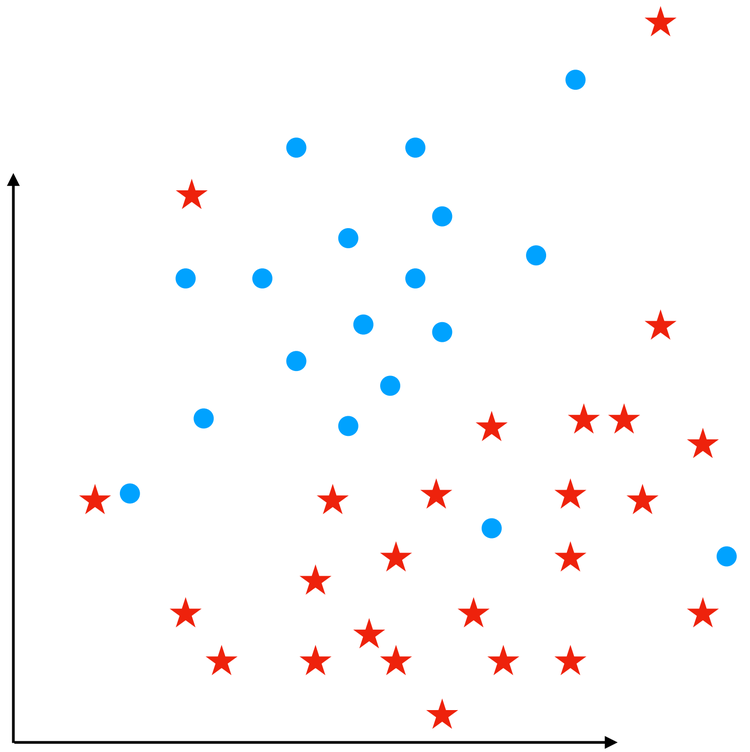


The position of the line depends crucially on where the points lie

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

How to perform classification?

- Many approaches have been developed
- We won't cover them in detail today
- Instead, we will provide quick introductions



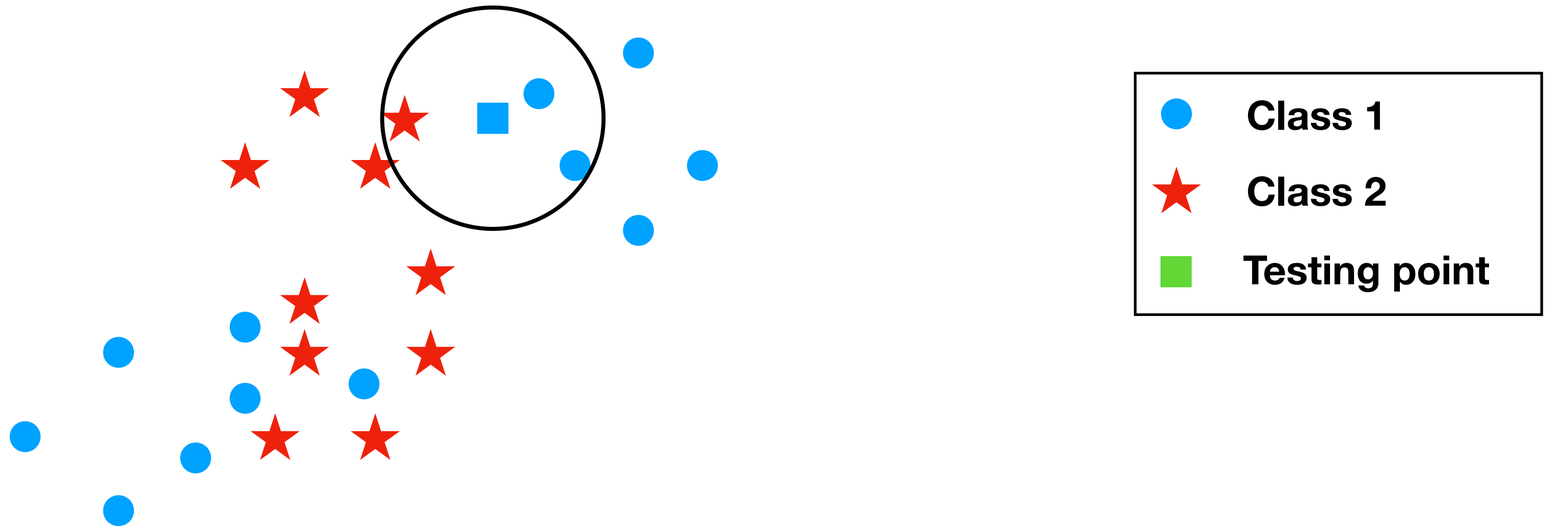
- Fundamental task of classification:

Divide the space into distinct decision regions

k-Nearest Neighbor

k-Nearest Neighbor

Assume that nearby points are likely to have similar labels



A new point x is classified based on the **majority vote of its k-nearest neighbors**

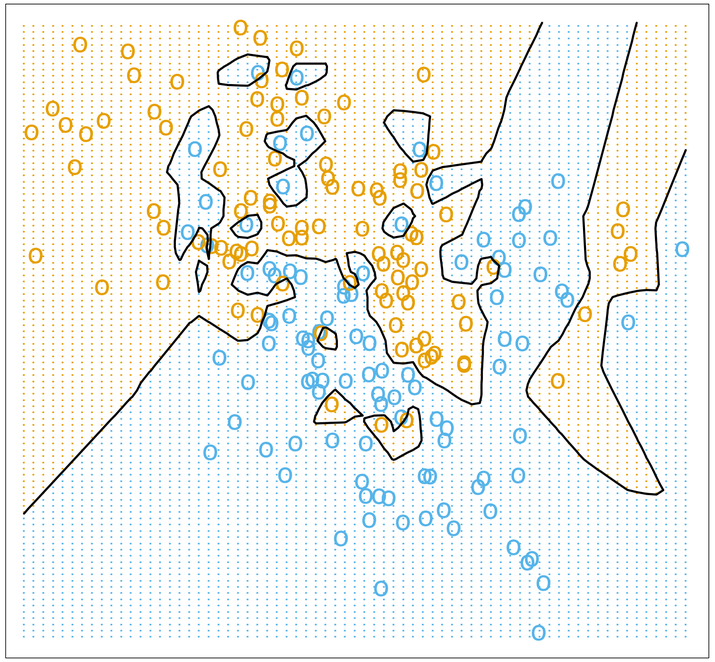
k-Nearest Neighbor

Pros:

- **No optimization** or training
- **Easy** to implement
- Works well in **low dimensions,** allowing for very complex decision boundaries

Cons:

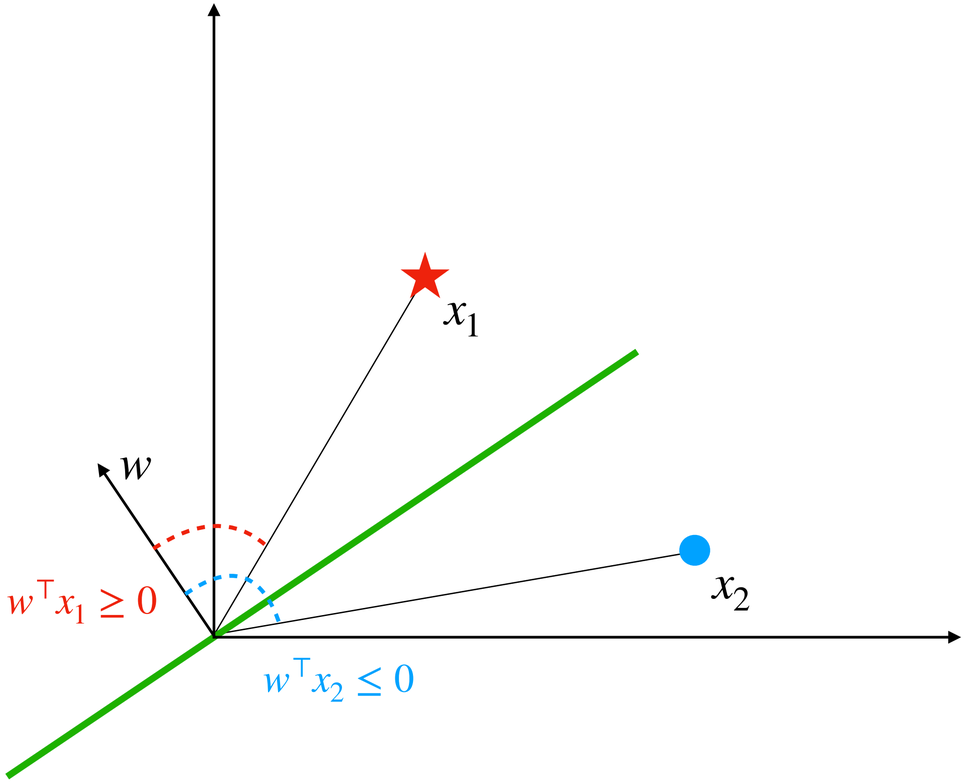
- **Slow** at query time
- Not suitable for high-dimensional data
- Choosing the right local distance is crucial



Linear Decision boundaries

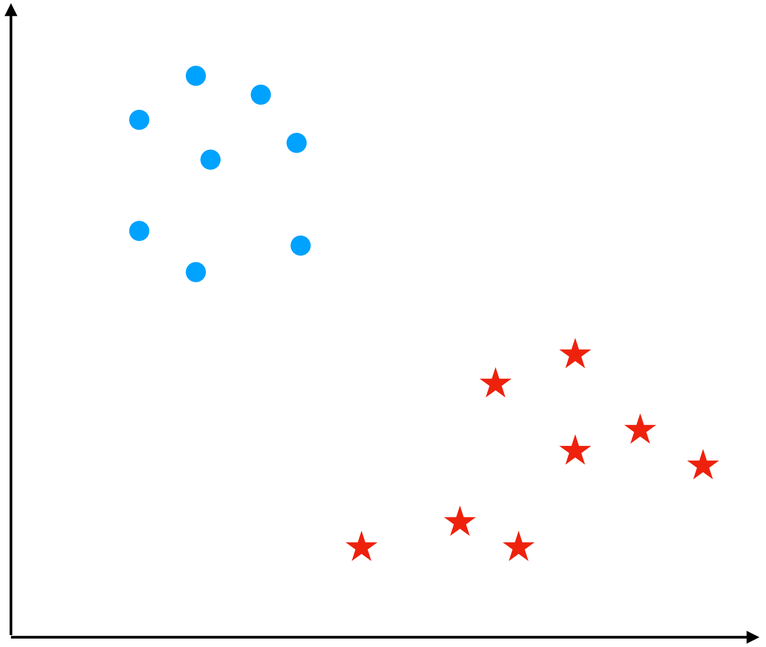
Assume we restrict ourselves to linear decision boundaries (hyperplane):

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



Separating hyperplane

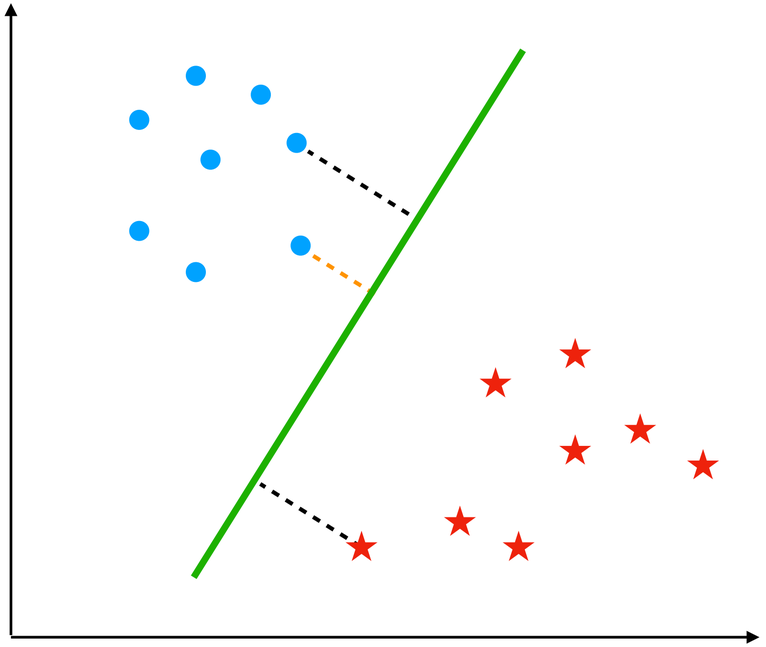
Assume we restrict ourselves to linear decision boundaries (hyperplane):
Assume the data are linearly separable, i.e., a separating hyperplane exists



Which separating hyperplane would you pick?

Margin

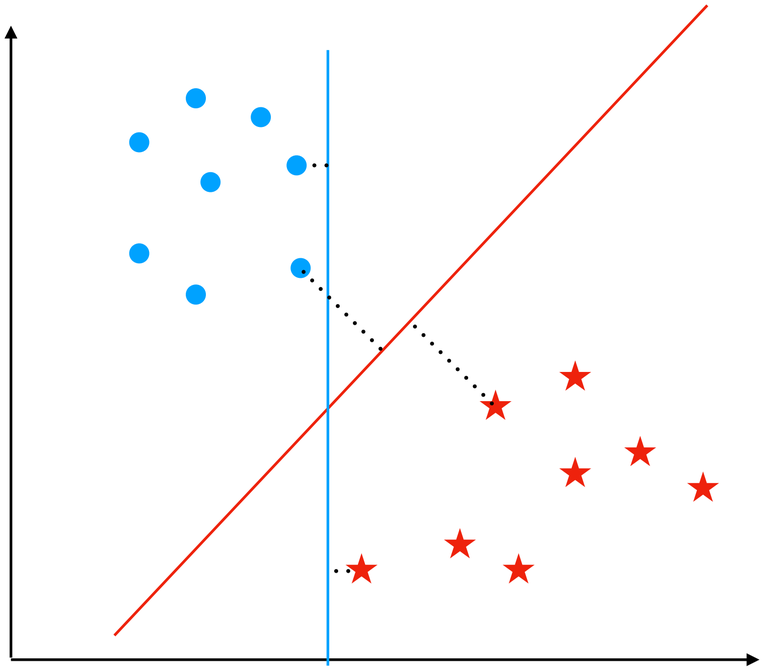
Key concept: The margin is the distance from the hyperplane to the closest point



➡ Take the one with the largest margin!

Max-margin separating hyperplane

Choose the hyperplane which maximizes the margin

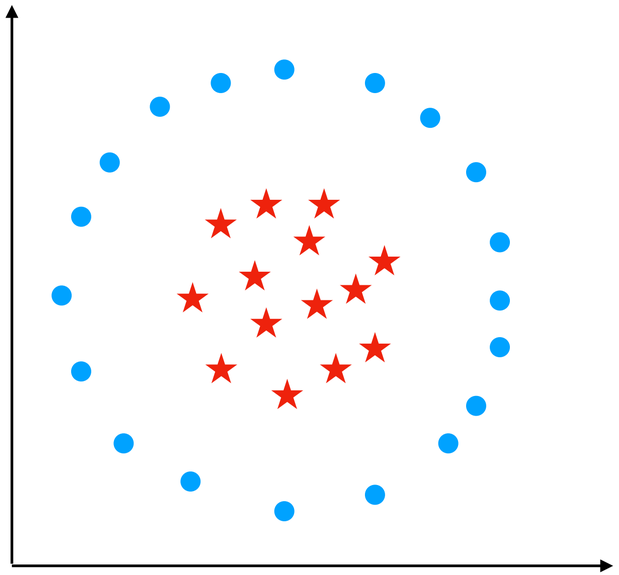


Why: If we slightly change the training set, the number of misclassifications will stay low

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

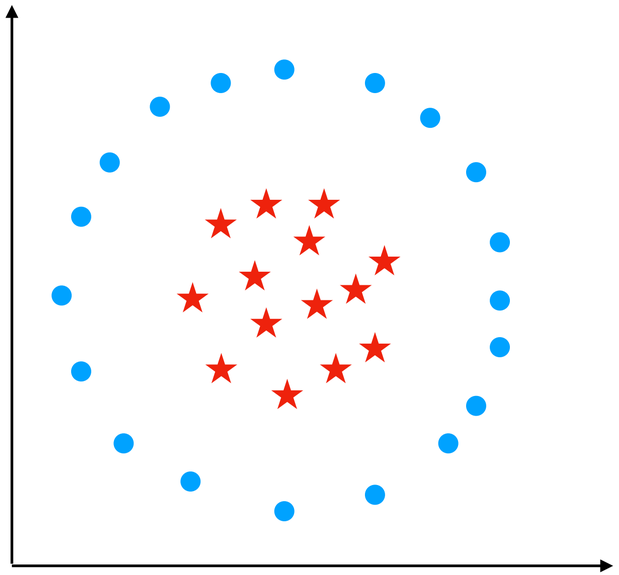
Nonlinear classifier

- Linear decision boundaries will not always work



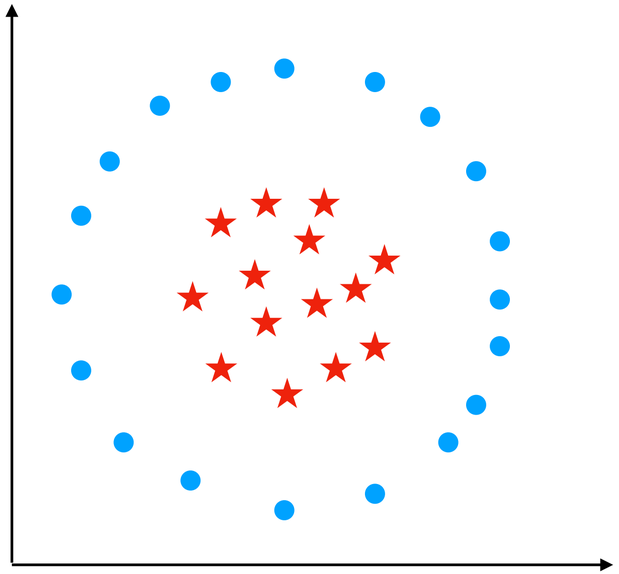
Nonlinear classifier

- Linear decision boundaries will not always work
- Feature augmentation (x, x^2, x^3, x^4)



Nonlinear classifier

- Linear decision boundaries will not always work
- Feature augmentation (x, x^2, x^3, x^4)
- Kernel Method



Formalizing Binary Classification

Setting: $(X, Y) \sim \mathscr{D}$ with ranges $\mathscr{X}, \mathscr{Y} = \{-1, 1\}$

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

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

Goal: minimize $L_{\mathscr{D}}(g)$

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:

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

Note: Bayes classifier is an *unattainable* gold standard, as we never know the underlying data distribution $\mathscr{D}$ in practice

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

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

Two classes of classification algorithms

- **Non-parametric**: approximate the conditional distribution $\mathbb{P}(Y = y \,|\, X = x)$ via local averaging

  ➡ Follow nearest neighbors' decisions (KNN)

- **Parametric**: approximate true distribution $\mathscr{D}$ via training data

  ➡ Minimize the empirical risk on training data (ERM)

Classification by empirical risk minimization

How: minimize the empirical risk instead of the true risk:

$$\min_{g:\mathcal{X}\to\mathcal{Y}} L_{\text{train}}(g) := \frac{1}{N}\sum_{n=1}^{N} 1_{g(x_n)\neq y_n} = \frac{1}{N}\sum_{n=1}^{N} 1_{y_n g(x_n)\leq 0}$$

Problem: L_{train} is not convex:

1. The set of classifiers is not convex because $\mathcal{Y}$ is discrete
2. The indicator function 1 is not convex because it is not continuous

Convex relaxation of the classification risk

1. Instead of learning $g : \mathcal{X} \to \mathcal{Y}$, learn $h : \mathcal{X} \to \mathbb{R}$ in a convex subset of continuous functions $\mathcal{H}$, and predict with $g(x) = \text{sign}(h(x))$. The problem becomes

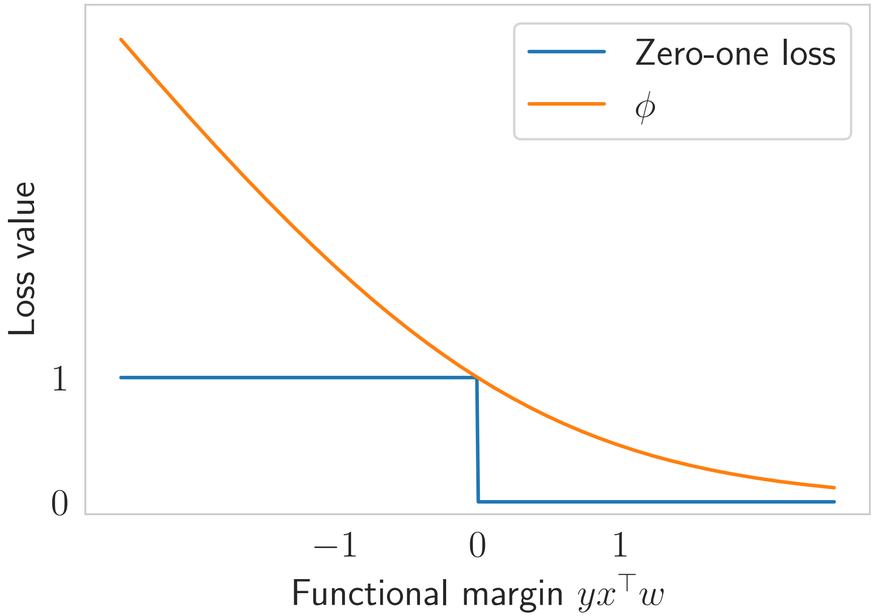
$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} 1_{y_n h(x_n) \leq 0}$$

2. Replace the indicator function by a convex surrogate $\phi : \mathbb{R} \to \mathbb{R}$ and minimize

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} \phi(y_n h(x_n))$$

ϕ is a function of the functional margin $y_n h(x_n)$

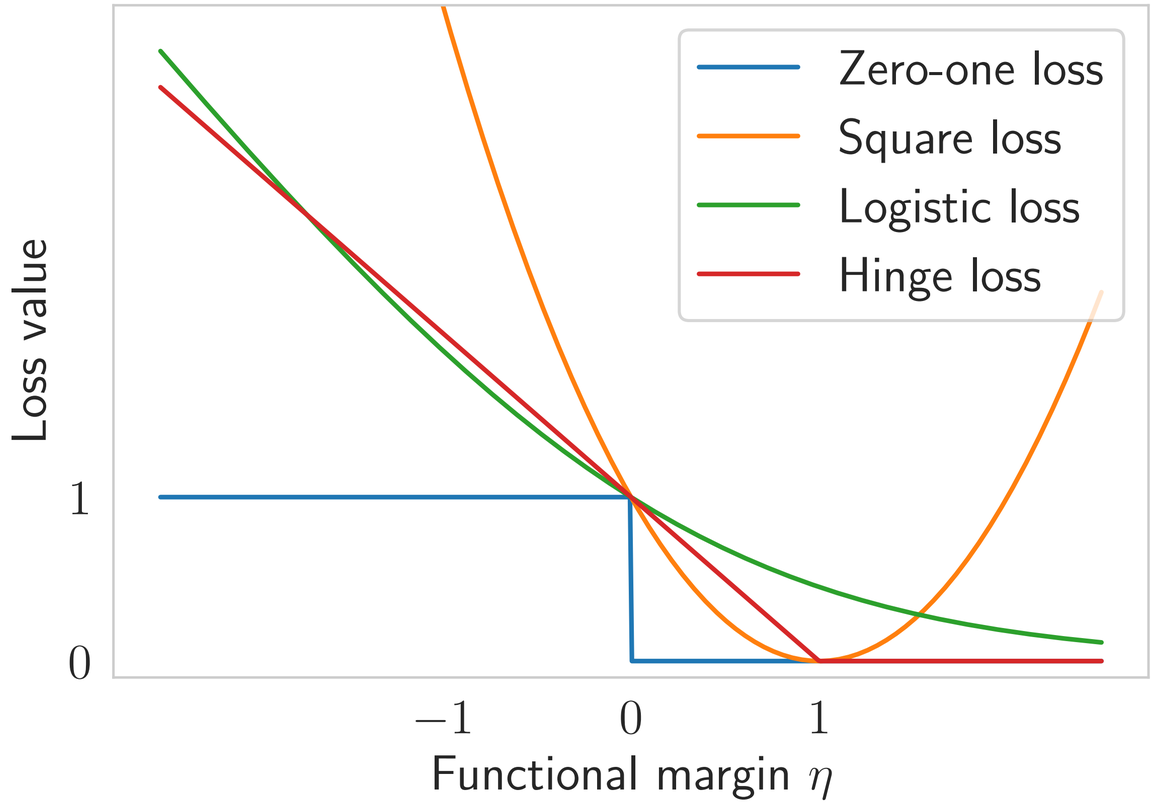
➡ This is a convex problem!



Remark: possible to bound the zero-one risk $L(g)$ by the ϕ risk *

* Under technical assumptions on the function ϕ

Losses for Classification



Logistic loss → logistic regression

Hinge loss → max margin classification

Do we still have time?

Bonus: a good regressor implies a good classifier

Consider $\mathcal{Y} = \{0,1\}$, for all regression functions $\eta : \mathcal{X} \to \mathbb{R}$ we can define a classifier as

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

Claim:

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

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 $\eta_* = \arg\min_{\eta} L_{\mathcal{D}}^{\ell_2}(\eta)$

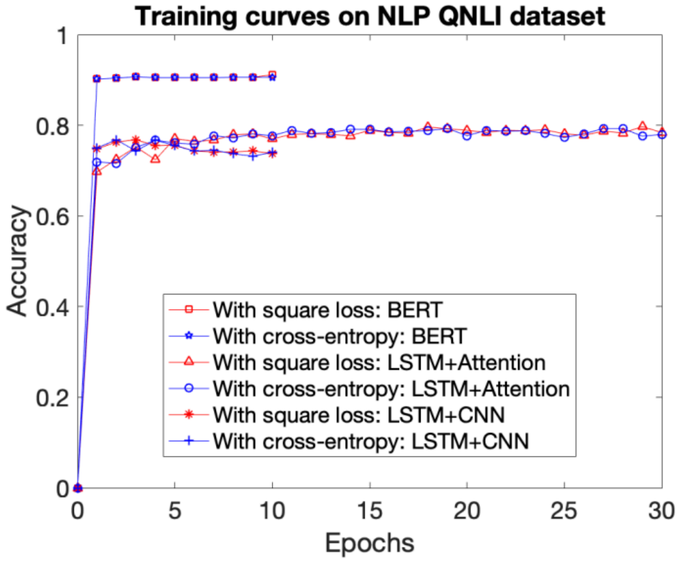
➡ If η is good for regression then g_η is good for classification too (converse is not true)

Bonus: does the loss function matter? (Over-parameterization regime)

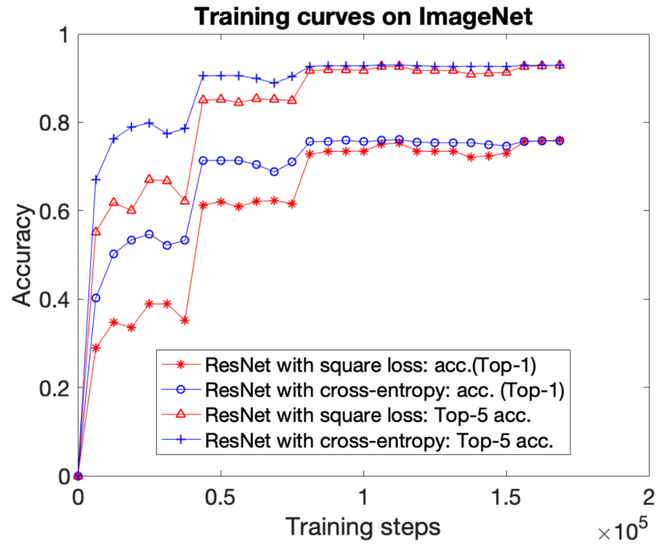
Assume sufficient over-parameterization ($n \ll d$), i.e. all training points are *equally close* to the separating hyperplane, with high probability

Optimization (training):

the outcome of optimization with *gradient descent*, is the same for both logistic loss and square loss



(a) NLP tasks



(c) Vision tasks

Reference: Classification vs regression in overparameterized regimes: Does the loss function matter?

Pics from Evaluation of Neural Architectures Trained with Square Loss vs Cross-Entropy in Classification Tasks

Recap

- Classification:
  - Mapping inputs to discrete outputs (categorical)
  - **Not** a special form of regression!
- Ways to perform classification:
  - Non-parametric: K-Nearest-Neighbors
  - Parametric: learning X-to-Y mapping via ERM
  - More complex decision boundaries? Non-linear classifiers
- Classification vs. Regression:
  - Classical regime: classification tasks cannot be well-solved using regression methods
  - Over-parameterization regime: solutions trained from both regression and classification methods are equivalent