Classification

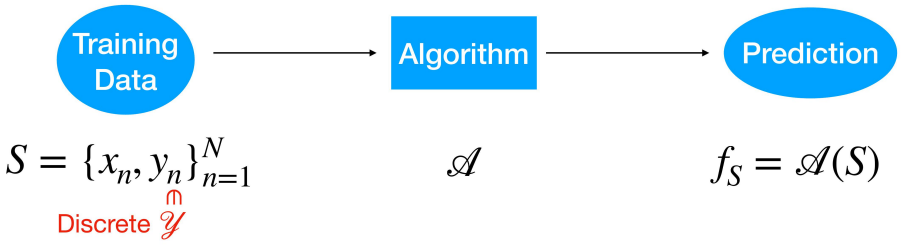
Machine Learning Course - CS-433
Oct 17, 2023
Nicolas Flammarion

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:



Classification: relates input to a categorical variable

$$(x, y) = \underbrace{0}_{\text{Discrete Set}} \times \underbrace{0}_{b}$$

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

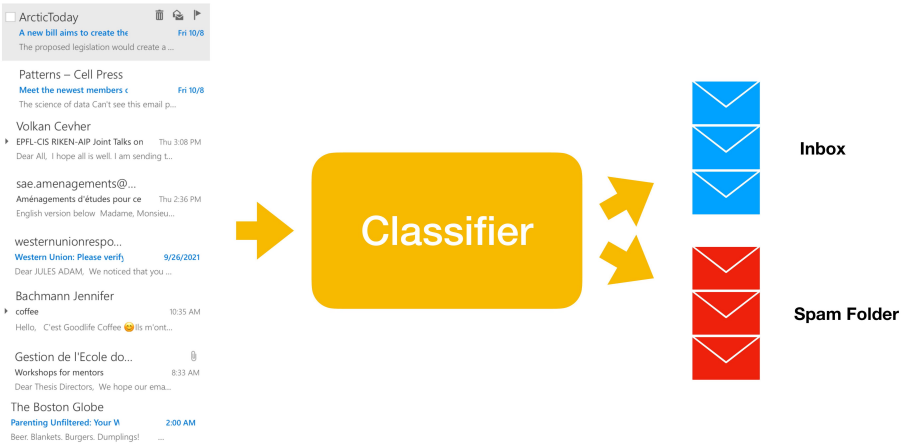
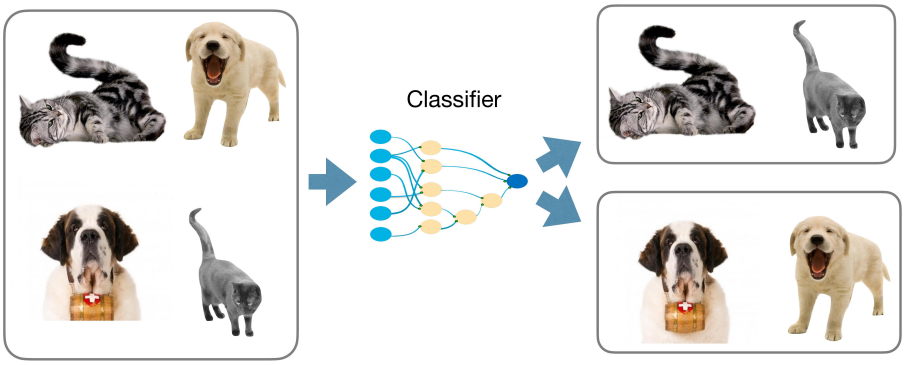
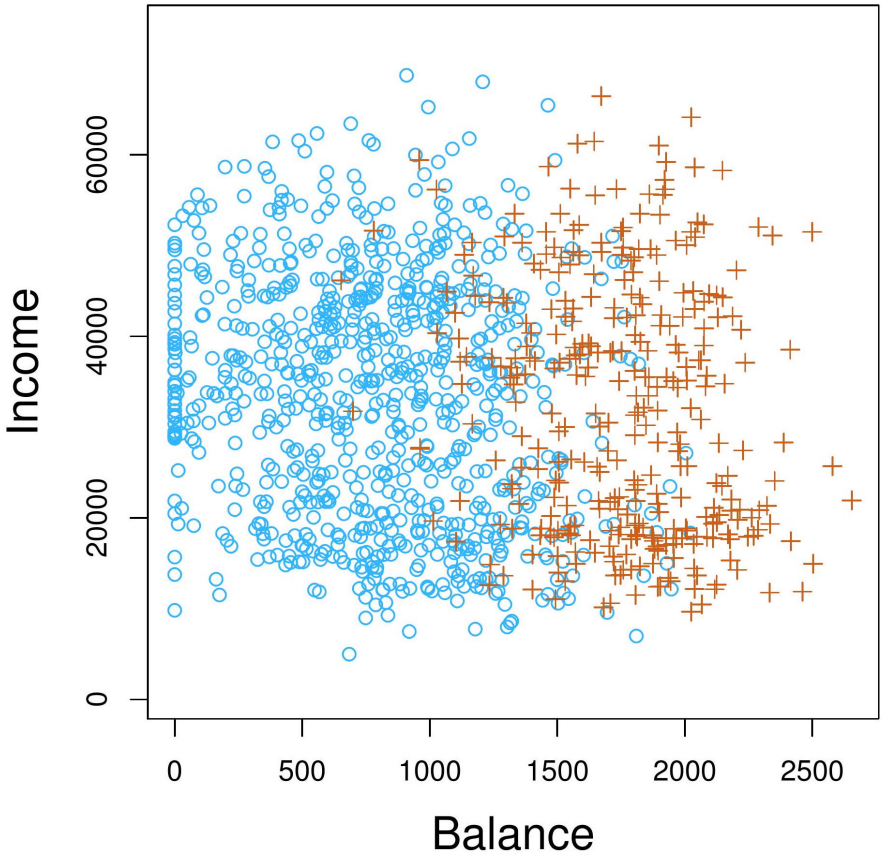


Image classification



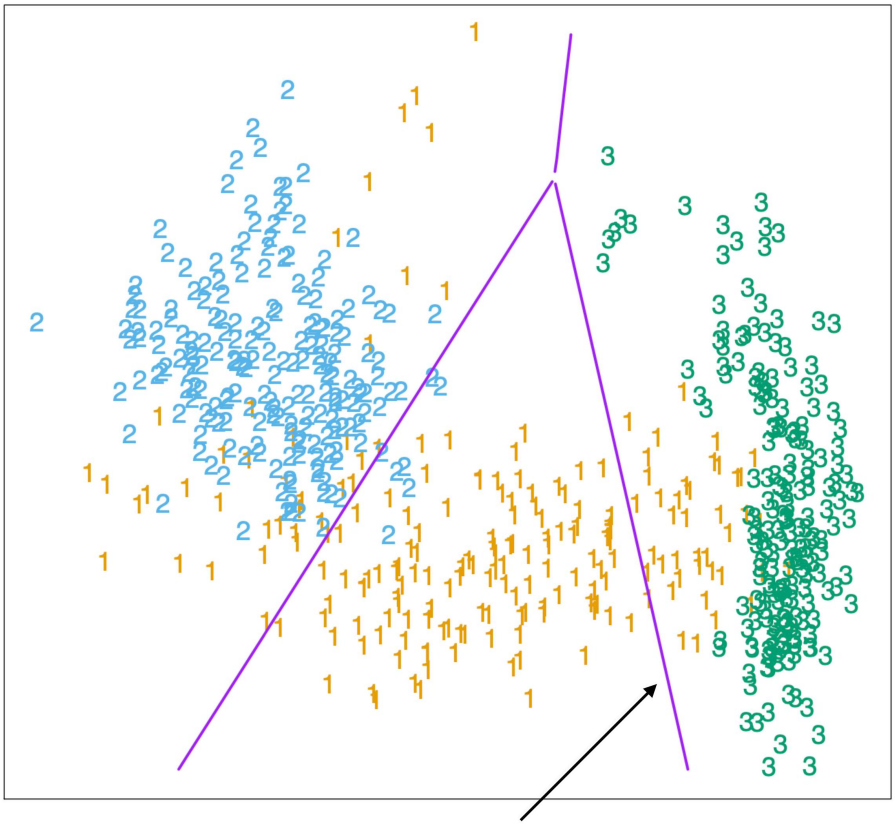
Credit Card Default



- individual who defaulted on their credit card payments individual who did not

Classifier

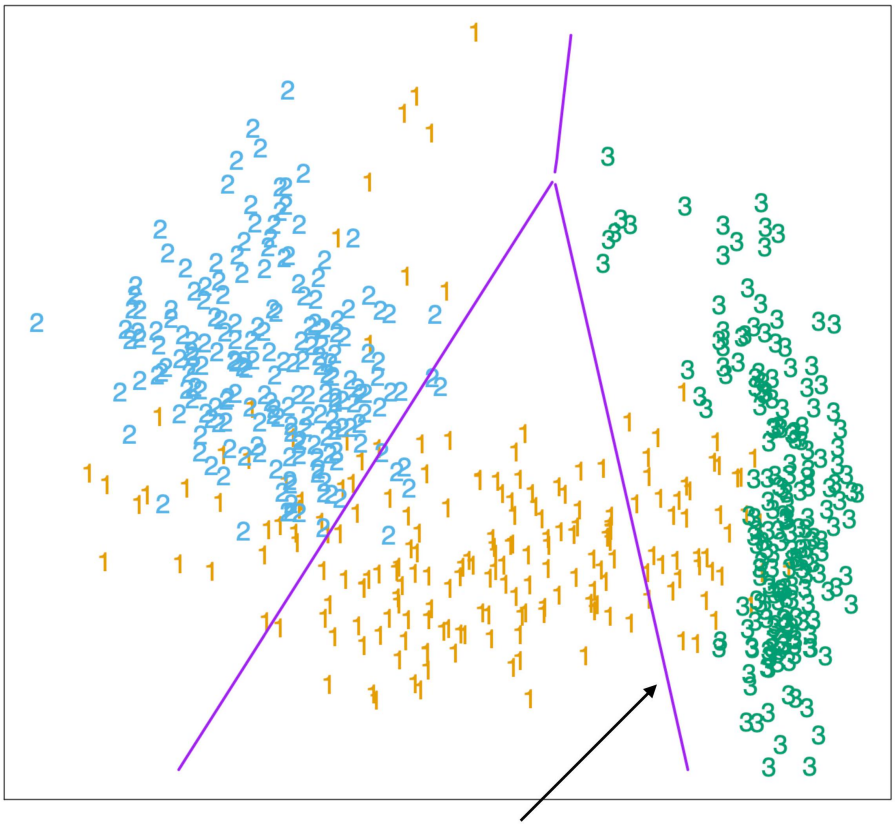
A classifier $f : \mathscr{X} \to \mathscr{Y}$ divides the input space into a collection of regions belonging to each class



Classifier

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

It can be linear

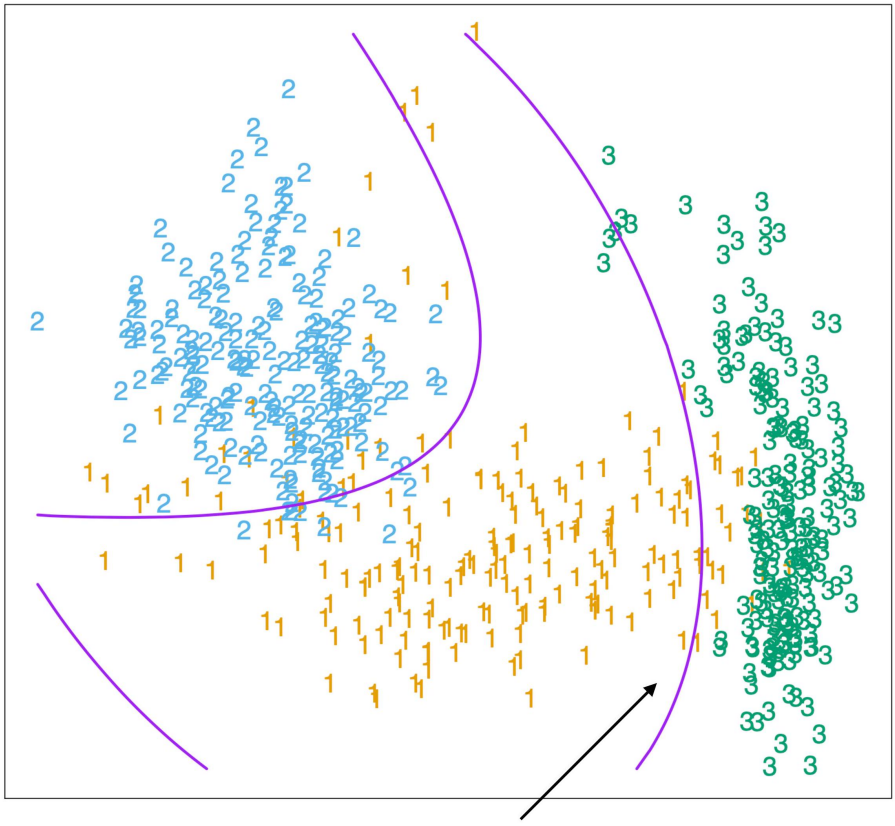


Linear Decision boundary

Classifier

A classifier $f : \mathscr{X} \to \mathcal{Y}$ 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:

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

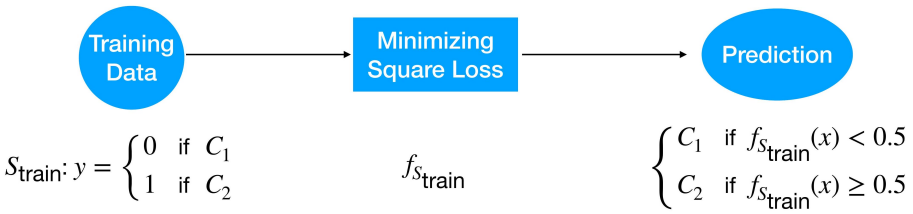
Could we use previously seen regression methods to solve it?

Classification: a special case of regression?

Classification is a regression problem with discrete labels:

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

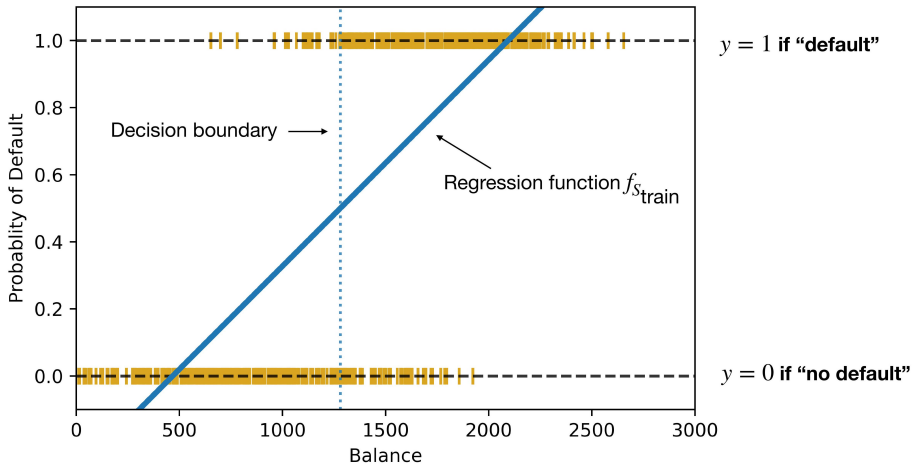
Could we use previously seen regression methods to solve it?



Is it a good idea?

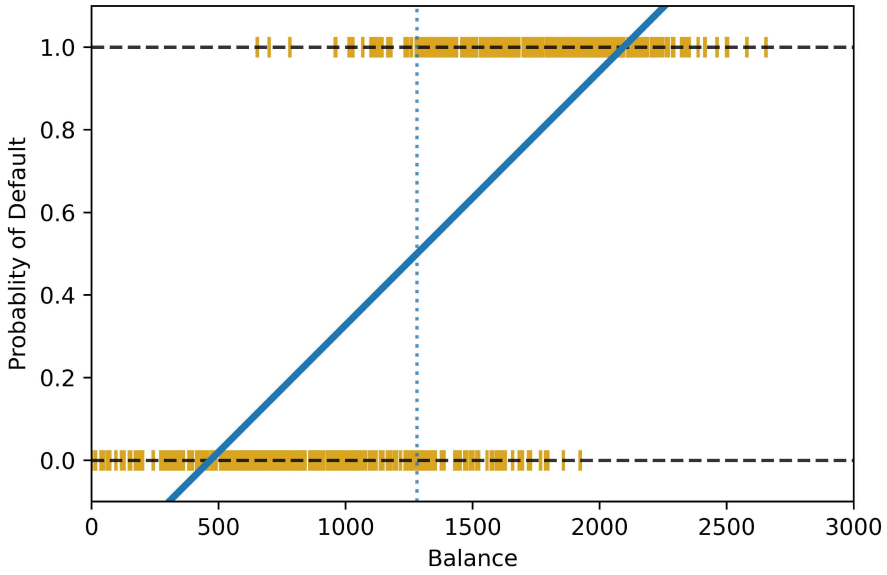
Credit-card default problem:

We label the output as probability for sake of interpretation



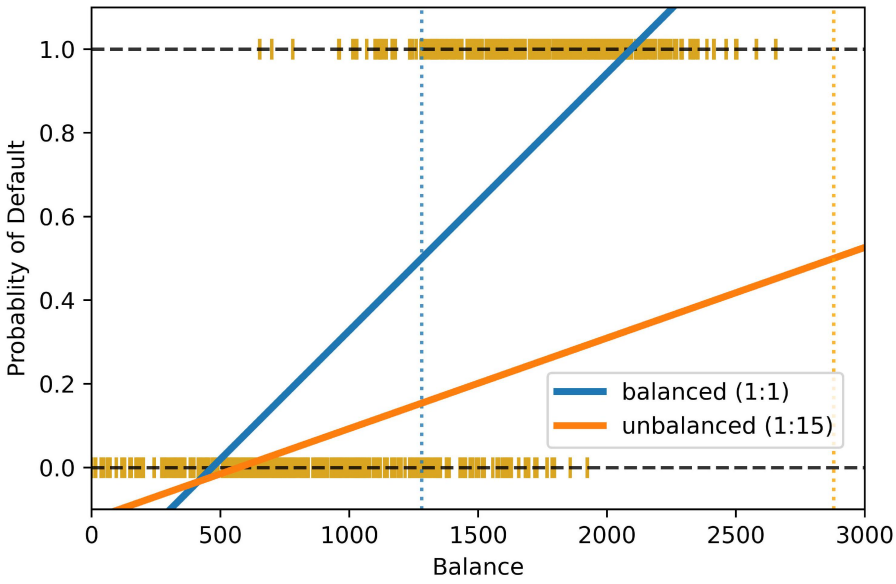
Classification is not just a special form of regression

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



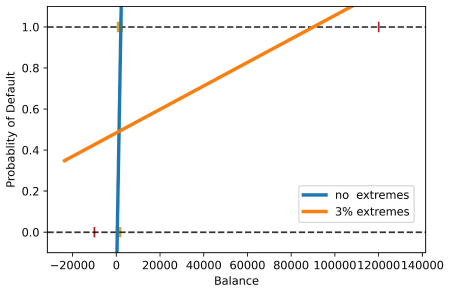
Classification is not just a special form of regression

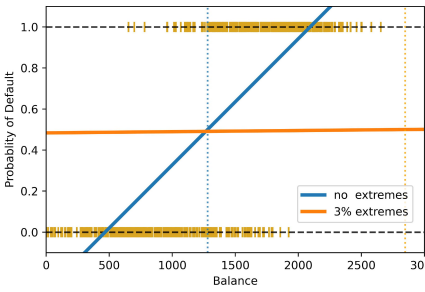
B. Sensitivity to unbalanced data



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

Classification is not just a special form of regression





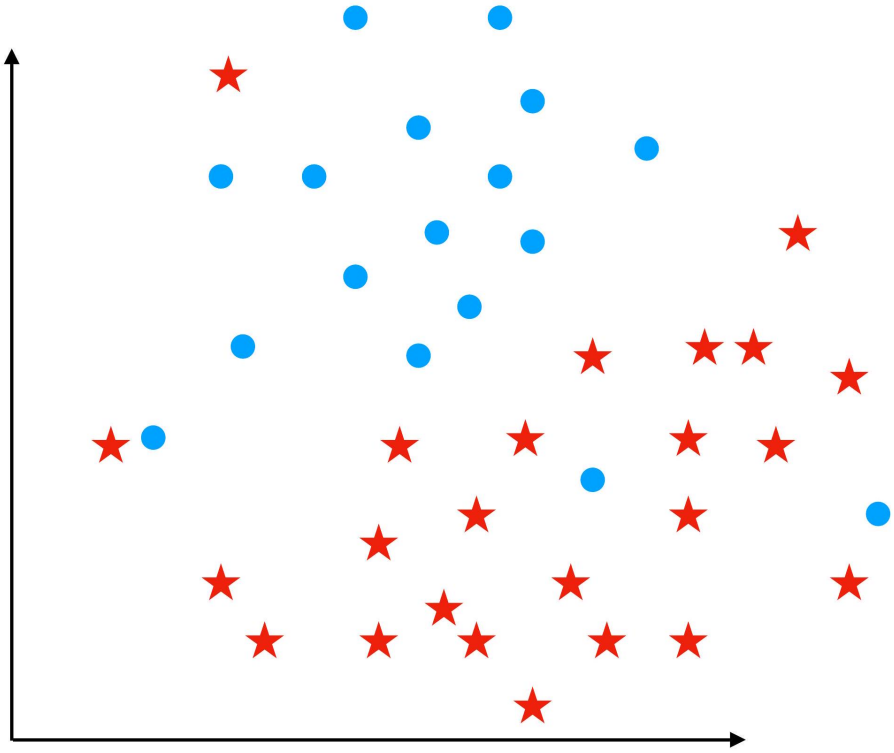
C. Sensitivity to extreme values:

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

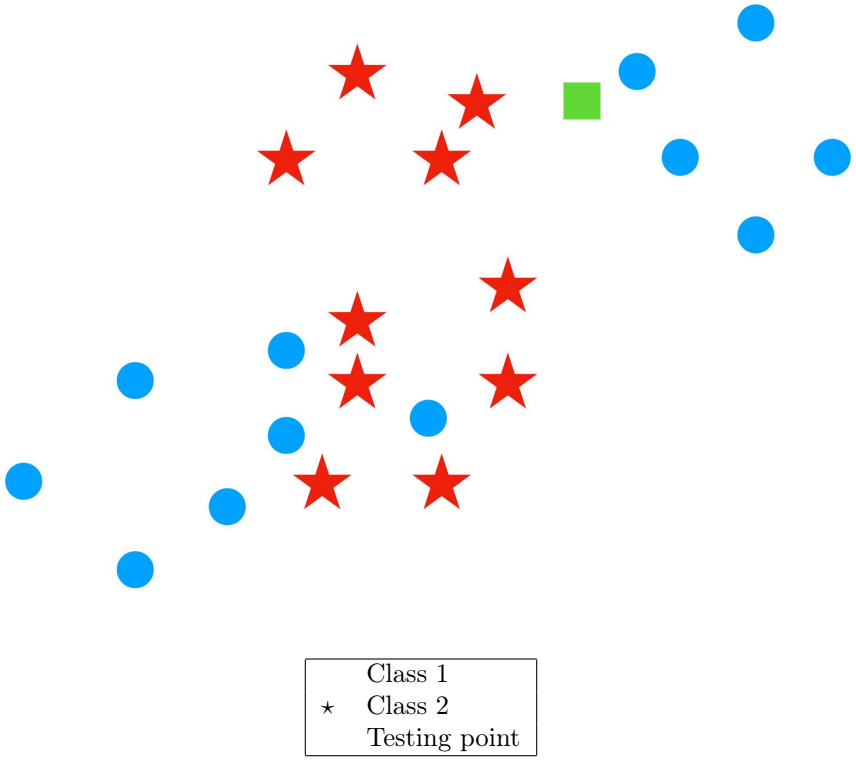
How to perform classification?

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



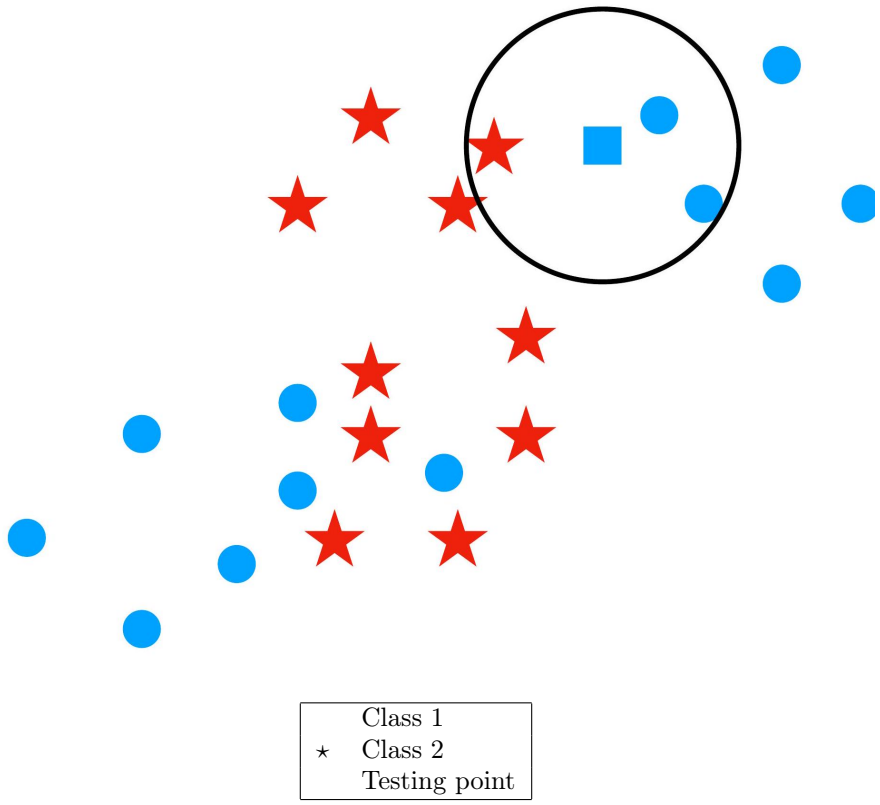
Divide the space into distinct decision regions

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

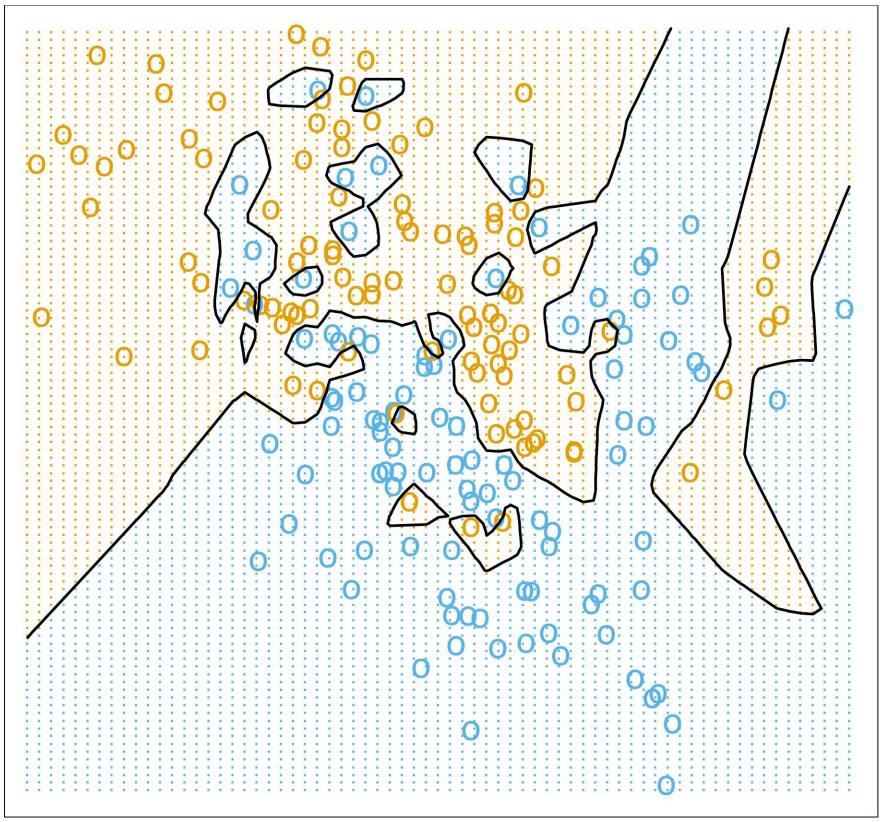
k-Nearest Neighbor

Pros:

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

Cons:

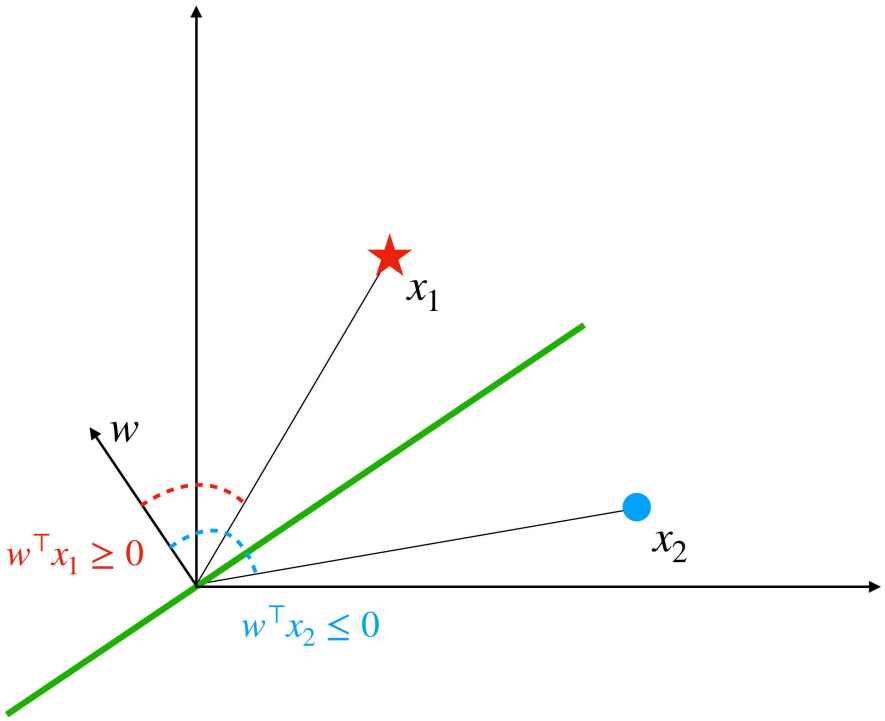
- Slow at query time
- Not suitable for high-dimensional data
- Choosing the right local distance is crucial



Linear Decision boundaries

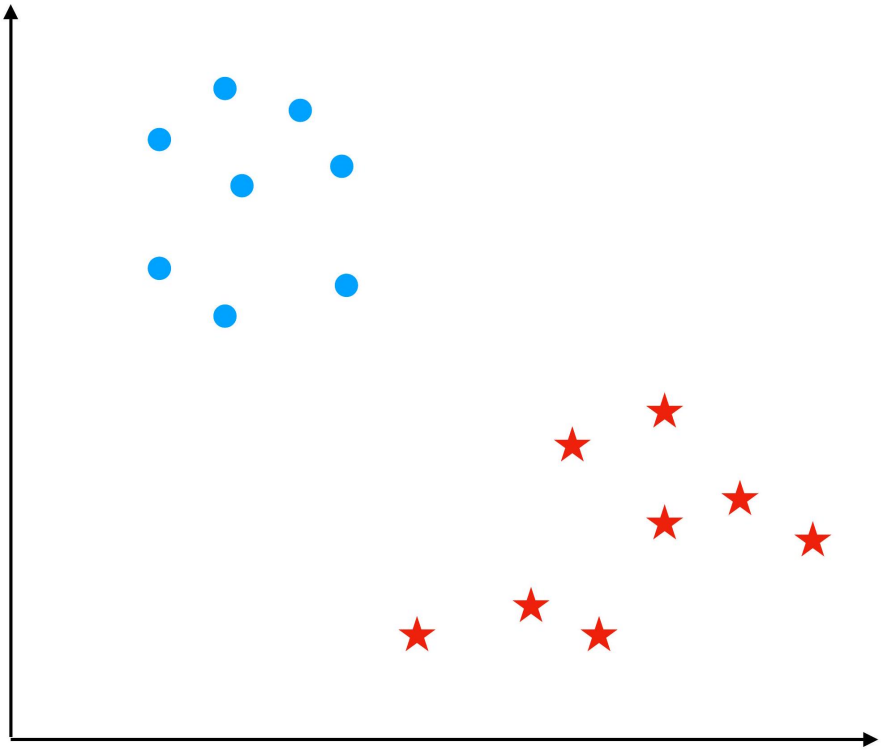
Assume we restrict ourselves to linear decision boundaries (hyperplane):

$\Rightarrow$ Prediction: $f(x) = \text{sign}\left(x^\top w\right)$



Separating hyperplane

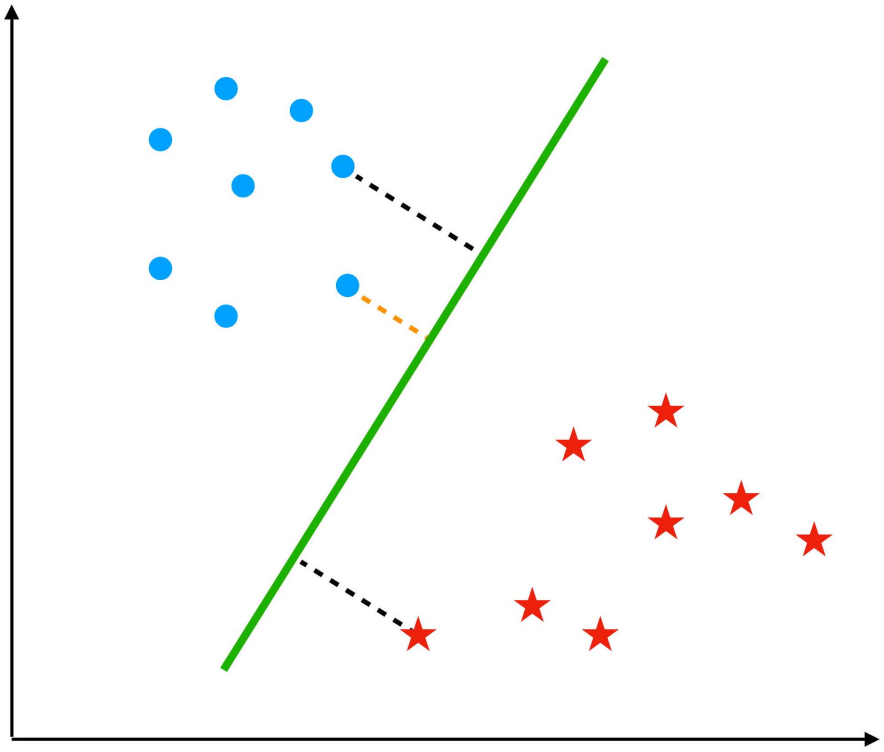
Assume we restrict ourselves to linear decision boundaries (hyperplane): Assume the data are linearly separable, i.e., a separating hyperplane exists



Which separating hyperplane would you pick?

Margin

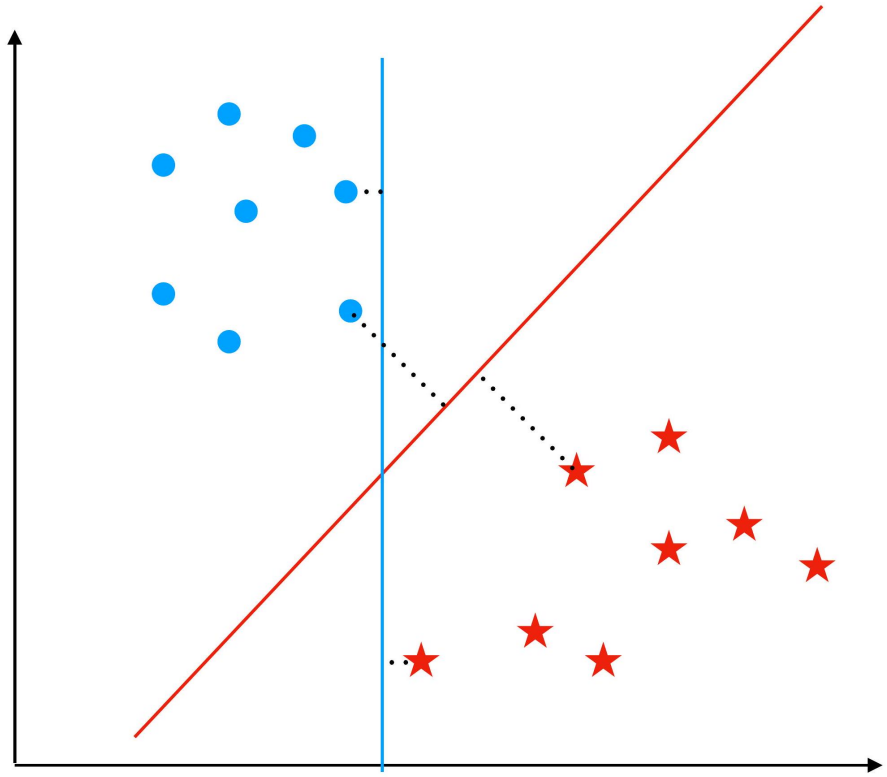
Key concept: The margin is the distance from the hyperplane to the closest point



⇒ Take the one with the largest margin!

Max-margin separating hyperplane

Choose the hyperplane which maximizes the margin

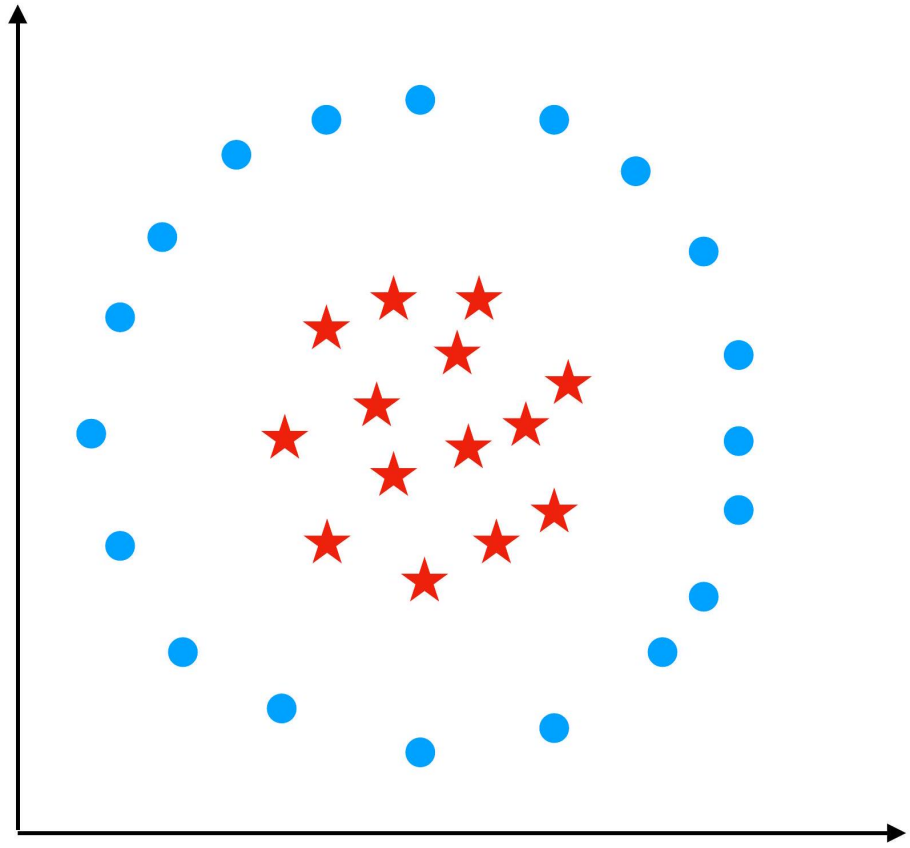


Why: If we slightly change the training set, the number of misclassifications will stay low

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

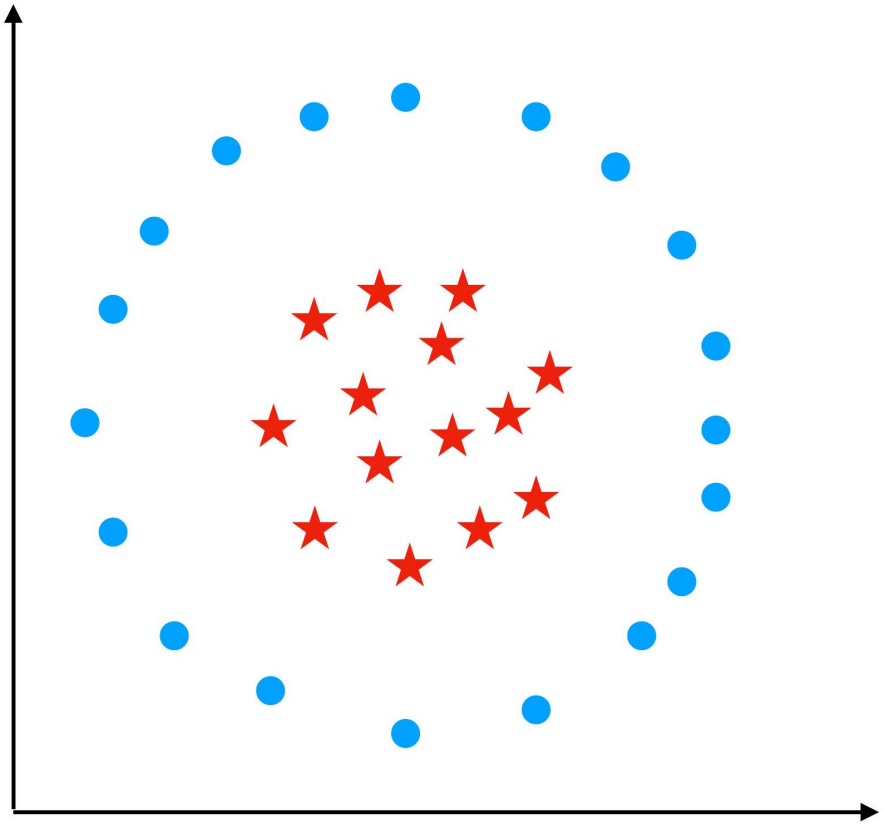
Nonlinear classifier

- Linear decision boundaries will not always work



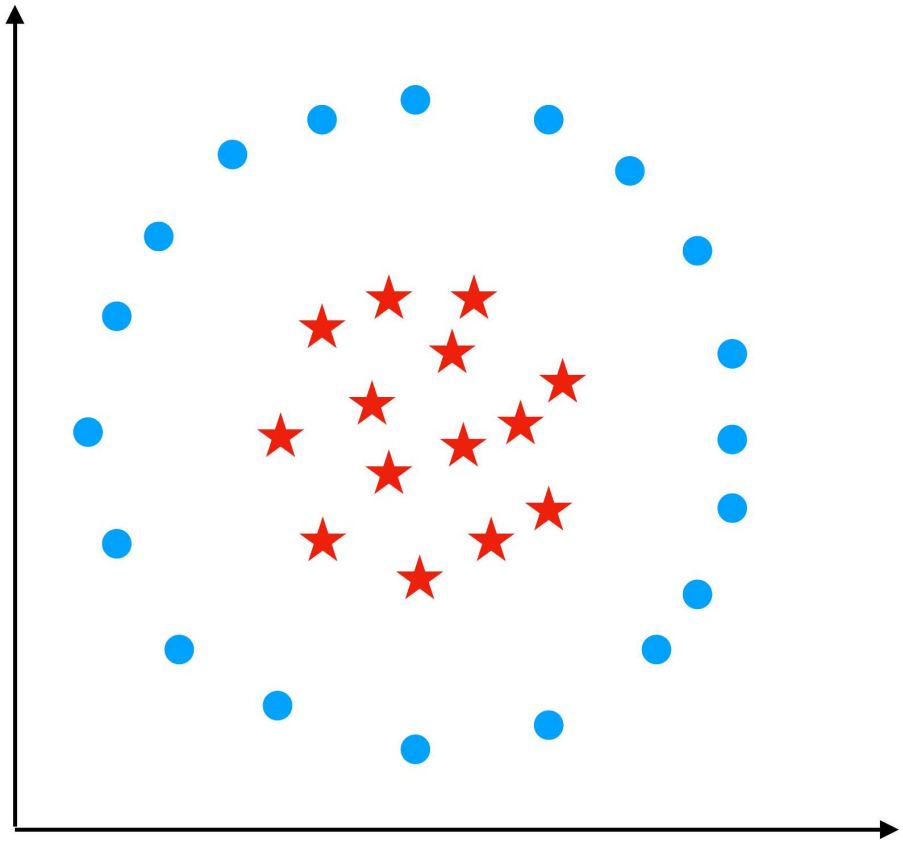
Nonlinear classifier

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



Nonlinear classifier

- Linear decision boundaries will not always work
- Feature augmentation $\left(x, x^2, x^3, x^4\right)$
- 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:

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

classification error

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

Bayes classifier

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

Def: The classifier $f_* = \arg\min L_{\mathscr{D}}(f)$ is called the Bayes classifier f
Claim:

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

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: $\forall x \in \mathscr{X}, f_*(x) \in \arg\min_{y \in \mathscr{Y}} \mathbb{P}(Y \neq y \mid X = x) \Longrightarrow f_* \in \arg\min_{f: \mathscr{X} \to \mathscr{Y}} L_{\mathscr{D}}(f)$

$$\begin{aligned} L_{\mathscr{D}}(f) = \mathbb{E}_{X,Y}\left[1_{Y \neq f(X)}\right] &= \mathbb{E}_X\left[\mathbb{E}_{Y|X}\left[1_{Y \neq f(X)} \mid X\right]\right] \\ &= \mathbb{E}_X[\mathbb{P}(Y \neq f(X) \mid X)] \\ &\geq \mathbb{E}_X\left[\min_{y \in \mathscr{Y}} \mathbb{P}(Y \neq y \mid X)\right] \\ &= \mathbb{E}_X\left[\mathbb{P}\left(Y \neq f_*(X) \mid X\right)\right] = \mathbb{E}_{X,Y}\left[1_{Y \neq f_*(X)}\right] = L_{\mathscr{D}}\left(f_*\right) \end{aligned}$$

Claim 2: $f_*(x) = \arg\min_{y \in \mathscr{Y}} \mathbb{P}(Y \neq y \mid X = x)$

$$f_*(x) = \arg\max_{y \in \mathscr{Y}} \mathbb{P}(Y = y \mid X = x) = \arg\min_{y \in \mathscr{Y}} \mathbb{P}(Y \neq y \mid X = x)$$

Two classes of classification algorithms

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

$\Rightarrow$ Follow nearest neighbors' decisions (KNN)

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

$\Rightarrow$ Minimize the empirical risk on training data (ERM)

Classification by empirical risk minimization

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

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

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

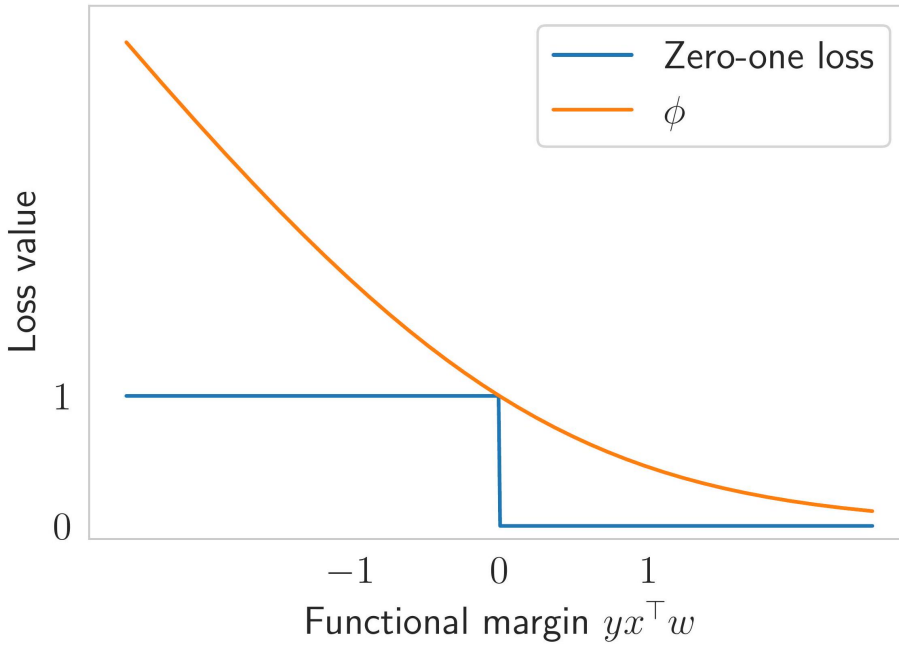
1. Instead of learning $f : \mathcal{X} \to \mathcal{Y}$, learn $g : \mathcal{X} \to \mathbb{R}$ in a convex subset of continuous functions $\mathcal{G}$, and predict with $f(x) = \text{sign}(g(x))$. The problem becomes

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

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

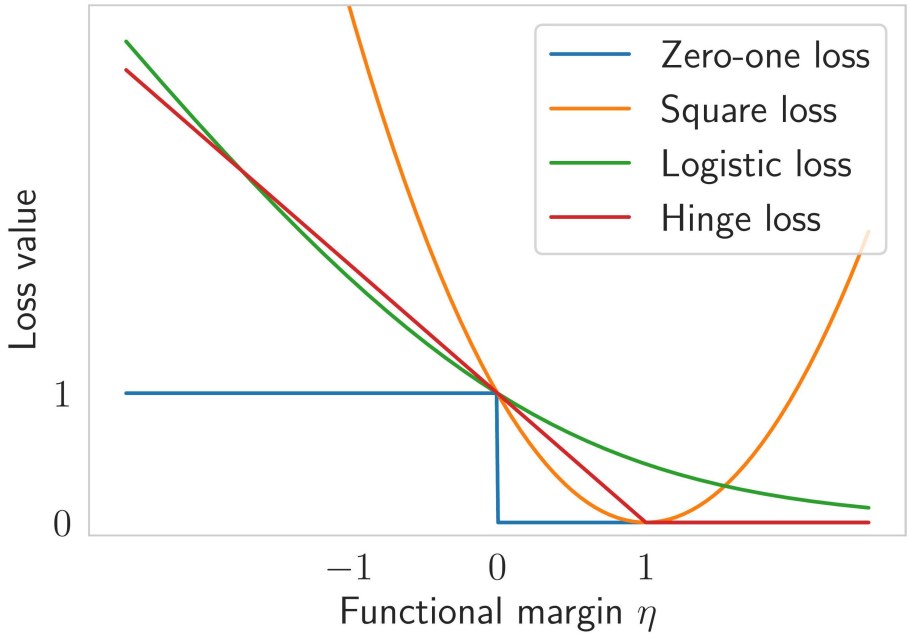
$$\min_{g \in \mathcal{G}} \frac{1}{N} \sum_{n=1}^{N} \phi\left(y_n g\left(x_n\right)\right)$$

ϕ is a function of the functional margin $y_n g\left(x_n\right)$
$\Rightarrow$ This is a convex problem!



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

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

$$\mathscr{X} \to \{0, 1\}$$
$$f_\eta : x \mapsto 1_{\eta(x) \geq 1/2}$$

Claim:

$$L_{\mathscr{D}}^{\text{classif}}(f_\eta) - L_{\mathscr{D}}^{\text{classif}}(f^*) \leq 2\sqrt{L_{\mathscr{D}}^{\ell_2}(\eta) - L_{\mathscr{D}}^{\ell_2}(\eta^*)}$$

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

$\Rightarrow$ If η is good for regression then f_η is good for classification too (converse is not true)

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

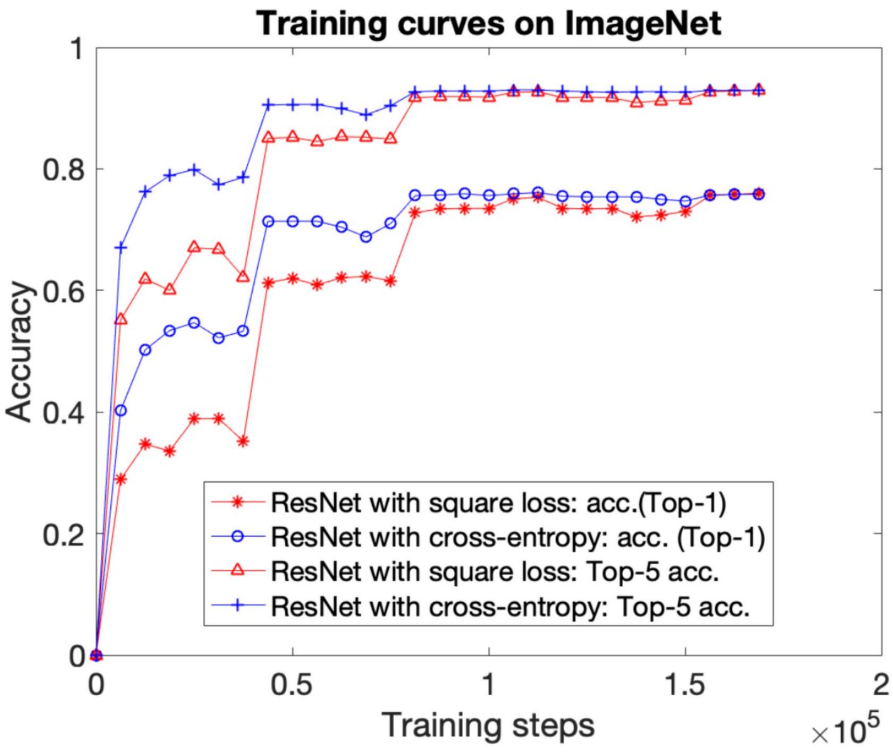
Assume sufficient over-parameterization ($n \ll d$), i.e. all training points are equally close to the separating hyperplane, with high probability

Optimization (training): the outcome of optimization with gradient descent, is the same for both logistic loss and square loss

With square loss: BERT
With cross-entropy: BERT
With square loss: LSTM+Attention
With cross-entropy: LSTM+Attention
With square loss: LSTM+CNN
With cross-entropy: LSTM+CNN

- With cross-entropy: LSTM+CNN

(a) NLP tasks
Training curves on ImageNet



(c) Vision 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