A crash course in machine learning

Florent Krzakala & Antoine Baker

https://sphinxteam.github.io/mlcourse_2019/

florent.krzakala@gmail.com **antoine.baker@gmail.com**

Artificial Intelligence?

I used to think we shouldn't call the field artificial intelligence but artificial stupidity. Really, our machines are dumb, and we're just trying to make them less dumb.

Y. Bengio

machine learning

Le Monde.fr

Comment le « deep learning » révolutionne l'intelligence artificielle

Cette technologie d'apprentissage, basée sur des réseaux de neurones artificiels, a complètement bouleversé le domaine de l'intelligence artificielle en moins de cinq ans.

Le Monde.fr | 24.07.2015 à 13h59 • Mis à jour le 28.07.2015 à 10h40 | Par **Morgane Tual**

« Je n'ai jamais vu une révolution aussi rapide. On est passé d'un système un peu obscur à un système utilisé par des millions de personnes en seulement deux ans. » Yann LeCun, un des pionniers du « deep learning », n'en revient toujours pas. Après une longue traversée du désert, « l'apprentissage profond », qu'il a contribué à inventer, est désormais la méthode phare de

'AI IS THE NEW ELECTRICITY'

"Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don't think AI will transform in the next several years."

Andrew Ng

Former chief scientist at Baidu, Co-founder at Coursera

Artificial Intelligence: The fourth industrial revolution

International edition

The Guardian

Weaponised AI is coming. Are algorithmic forever wars our future?

Ben Tarnoff

The US military is creating a more automated form of warfare – one that will greatly increase its capacity to wage war everywhere forever.


ECOLE NORMALE SUPERIEURE
LABORATOIRES


ECOLE NORMALE SUPERIEURE
LABORATOIRES

This person does not exist

Ian Goodfellow
@goodfellow_ian

Follow

4.5 years of GAN progress on face generation. arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434 arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196 arxiv.org/abs/1812.04948

A new tool in the box!

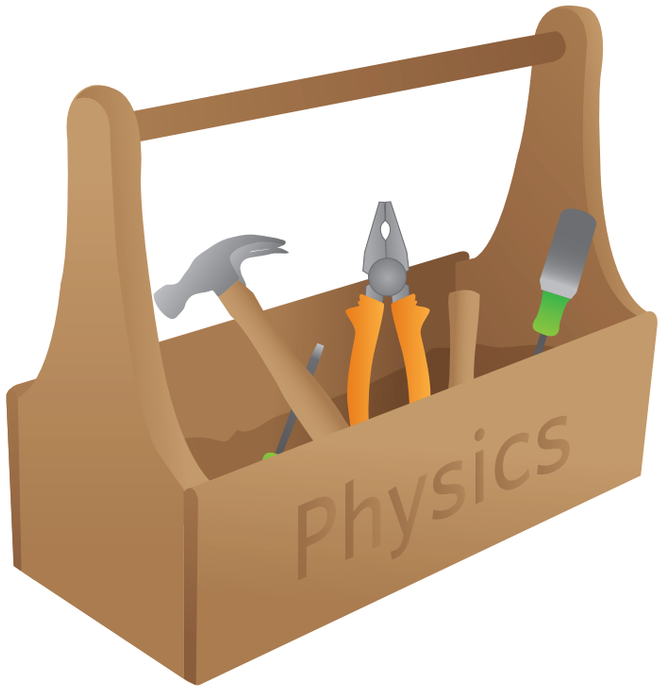
MACHINE LEARNING

New tool in the box

A recent burst of activity in applying machine learning to tackle fundamental questions in physics suggests that associated techniques may soon become as common in physics as numerical simulations or calculus.

Lenka Zdeborová

The goal of machine learning, broadly speaking, is to design a computer code — the eponymous machine — capable of discovering meaningful structure in data. The last decade saw a game-changing revolution unfold in this field: with the development of deep neural networks[1], tasks that were considered inaccessible to automated learning became possible. This prompted fierce competition in the artificial intelligence market, but it also brought promise to many areas of data-intensive fundamental science — with physics being no exception. Current machine-learning systems are not yet able to divine the laws of general relativity from planetary data, but they are able to reliably recognize human faces, detect objects in photographs and even beat world champions of Go[2]. And now, writing in *Nature Physics*, two groups have used artificial neural networks to recognize different phases of matter and localize associated phase transitions[3,4].



It should be stressed that saying one applied machine learning to a given problem is about as generic as saying that one used numerical simulations. It is clear to every researcher in physics that there are many kinds of numerical

learning[5]. Another group recently showed that a support vector machine can be used to classify which particles in a glassy system are susceptible to rearrangement[6]. Decision forests have been used to classify metals from insulators based on the hybridization function, combined with kernel ridge regression to predict correlation functions in many-

https://arxiv.org/pdf/1903.10563.pdf

Statistical Physisc

Particle Physics and Cosmology

Many-Body Quantum Matter

Quantum computing

Chemistry and Materials

Machine learning and the physical sciences

Giuseppe Carleo

Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA

Ignacio Cirac

Max-Planck-Institut fur Quantenoptik, Hans-Kopfermann-Straße 1, D-85748 Garching, Germany

Kyle Cranmer

Center for Cosmology and Particle Physics, Center of Data Science, New York University, 726 Broadway, New York, NY 10003, USA

Laurent Daudet

LightOn, 2 rue de la Bourse, F-75002 Paris, France

Maria Schuld

University of KwaZulu-Natal, Durban 4000, South Africa National Institute for Theoretical Physics, KwaZulu-Natal, Durban 4000, South Africa, and Xanadu Quantum Computing, 777 Bay Street, M5B 2H7 Toronto, Canada

Naftali Tishby

The Hebrew University of Jerusalem, Edmond Safra Campus, Jerusalem 91904, Israel

Leslie Vogt-Maranto

Department of Chemistry, New York University, New York, NY 10003, USA

Lenka Zdeborová

Institut de physique théorique, Université Paris Saclay, CNRS, CEA, F-91191 Gif-sur-Yvette, France

Machine learning encompasses a broad range of algorithms and modeling tools used for a vast array of data processing tasks, which has entered most scientific disciplines in recent years. We review in a selective way the recent research on the interface between machine learning and physical sciences. This includes conceptual developments in machine learning (ML) motivated by physical insights, applications of machine learning techniques to several domains in physics, and cross-fertilization between the two fields. After giving basic notion of machine learning methods and principles, we describe examples of how statistical physics is used to understand methods in ML. We then move to describe applications of ML methods in particle physics and cosmology, quantum many body physics, quantum computing, and chemical and material physics. We also highlight research and development into novel computing architectures aimed at accelerating ML. In each of the sections we describe recent successes as well as domain-specific methodology and challenges.

Contents

Understanding vs Solving

The « science » way of solving electromagnetism

James Clerk Maxwell (1831–1879)

$$1.\quad \nabla \cdot \mathbf{D} = \rho_v$$

$$2.\quad \nabla \cdot \mathbf{B} = 0$$

$$3.\quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$4.\quad \nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J}$$

The «electric fish» way of solving electromagnetism

Electric fish

An electric fish is any fish that can generate electric fields. A fish that can generate electric fields is said to be electrogenic while a fish that has the ability to detect electric fields is said to be electroreceptive. Wikipedia

If we know the equations, we can compute everything

Electric fishes can invert electromagnetic fields much better than the state of the art in inverse problem of electric tomography in signal processing

Machine learning is like building an artificial fish to solve a problem

THINK LIKE A FISH

What will be discussed in these lectures?

Goals

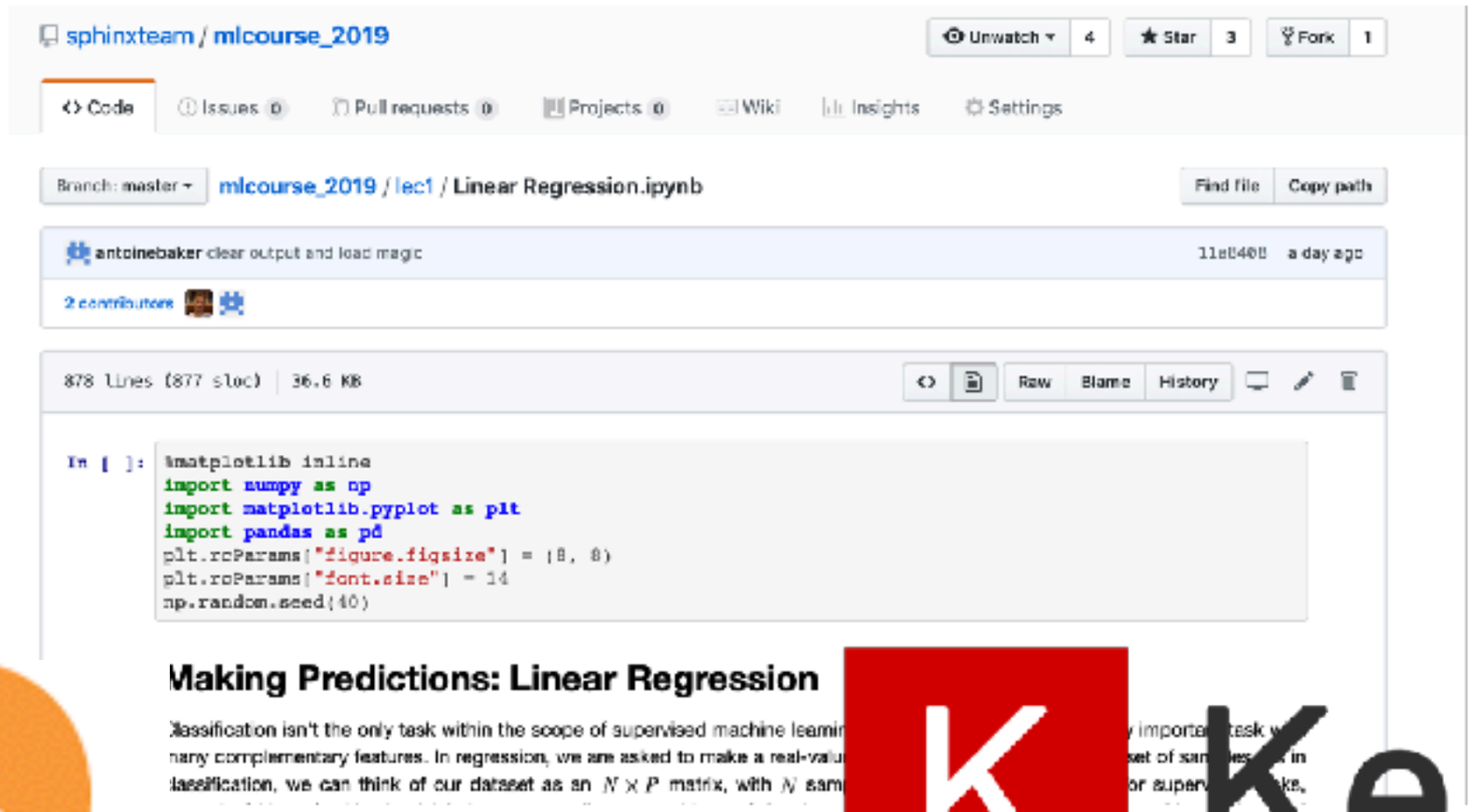
➡️ Expose the Principles behind some of the classical technics in machine learning

➡️ Hand-on introduction to solving simple problems through the use of publicly available softwares

Jupyter notebooks

Download the set of notebooks at:

https://sphinxteam.github.io/mlcourse_2019/



scikit learn

Keras

Inria
INVENTORS FOR THE DIGITAL WORLD

TensorFlow

Google

A selection of topics

- Statistical theory : Maximum likelihood, Bayes, VC Bounds and Uniform convergence
- Supervised learning : Linear Regression, Ridge, Lasso, high Dimensional Data, Kernel methods, Boosting
- Deep learning: multi-layer net, conv-net, auto-encoder
- Unsupervised learning : Mixture Models, PCA & Kernel PCA
- Basics of Generative models & Reinforcement learning

A selection of topics

- **Statistical theory** : Maximum likelihood, Bayes, **VC Bounds and Uniform convergence**
- **Supervised learning : Linear Regression, Ridge, Lasso, high Dimensional Data**, Kernel methods, Boosting
- Deep learning: multi-layer net, conv-net, auto-encoder
- Unsupervised learning : Mixture Models, PCA & Kernel PCA
- Basics of Generative models & Reinforcement learning

Supervised problems

Classification & Regression

Given a dataset with label, find a function that can assign labels to new unlabelled data

Regression

Classification

Labeled data: $\begin{cases} \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_n\} & \vec{x} \in \mathbb{R}^d \\ \{y_1, y_2, y_3, \ldots, y_n\} & y \in \mathbb{R} \quad \text{or} \quad \mathrm{y} \in \mathbb{N} \end{cases}$

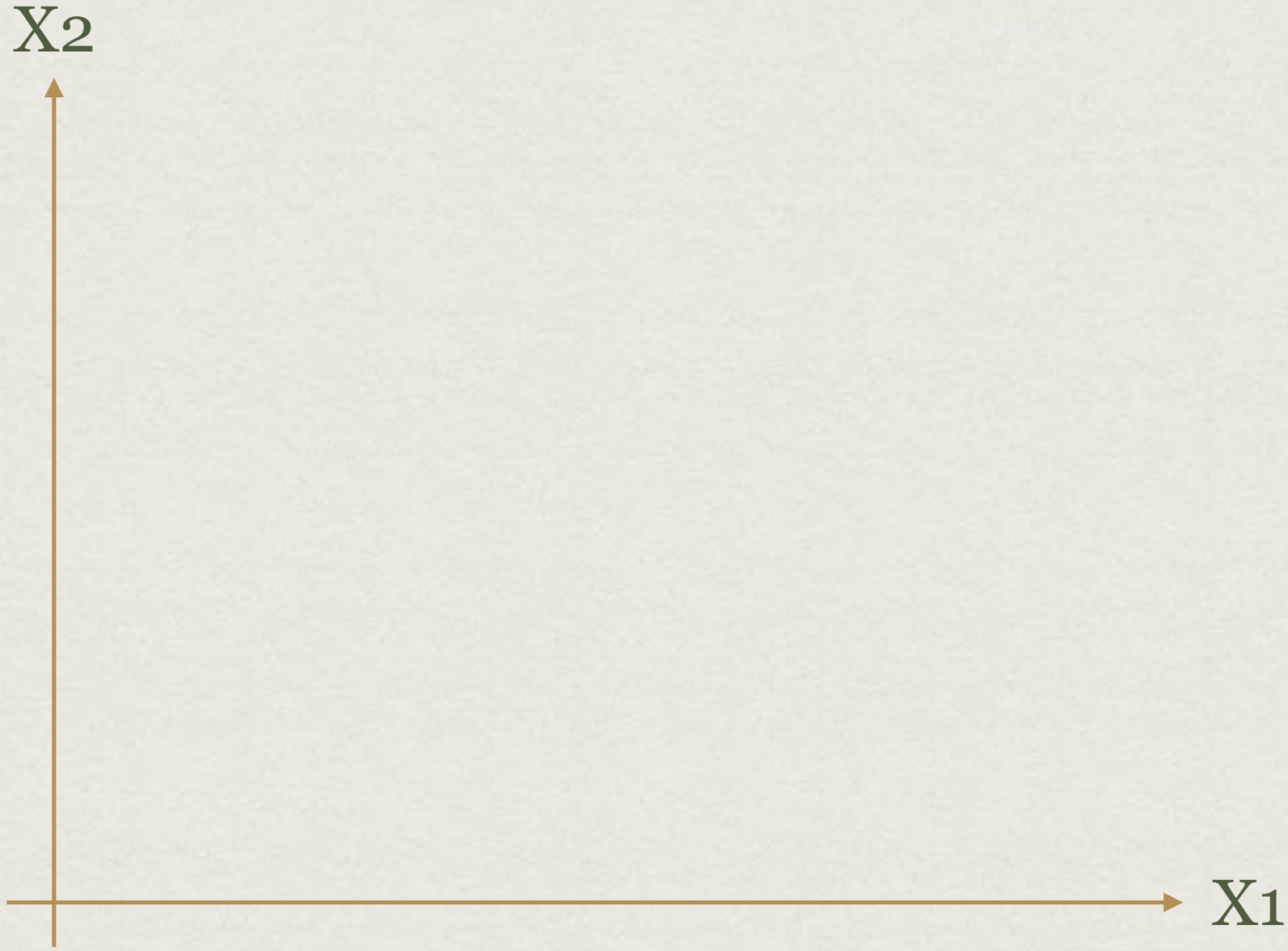
Goal: Find a function $f_W(\vec{x})$ that outputs the right class/value for an object $\vec{x}$

Cats vs dogs classification

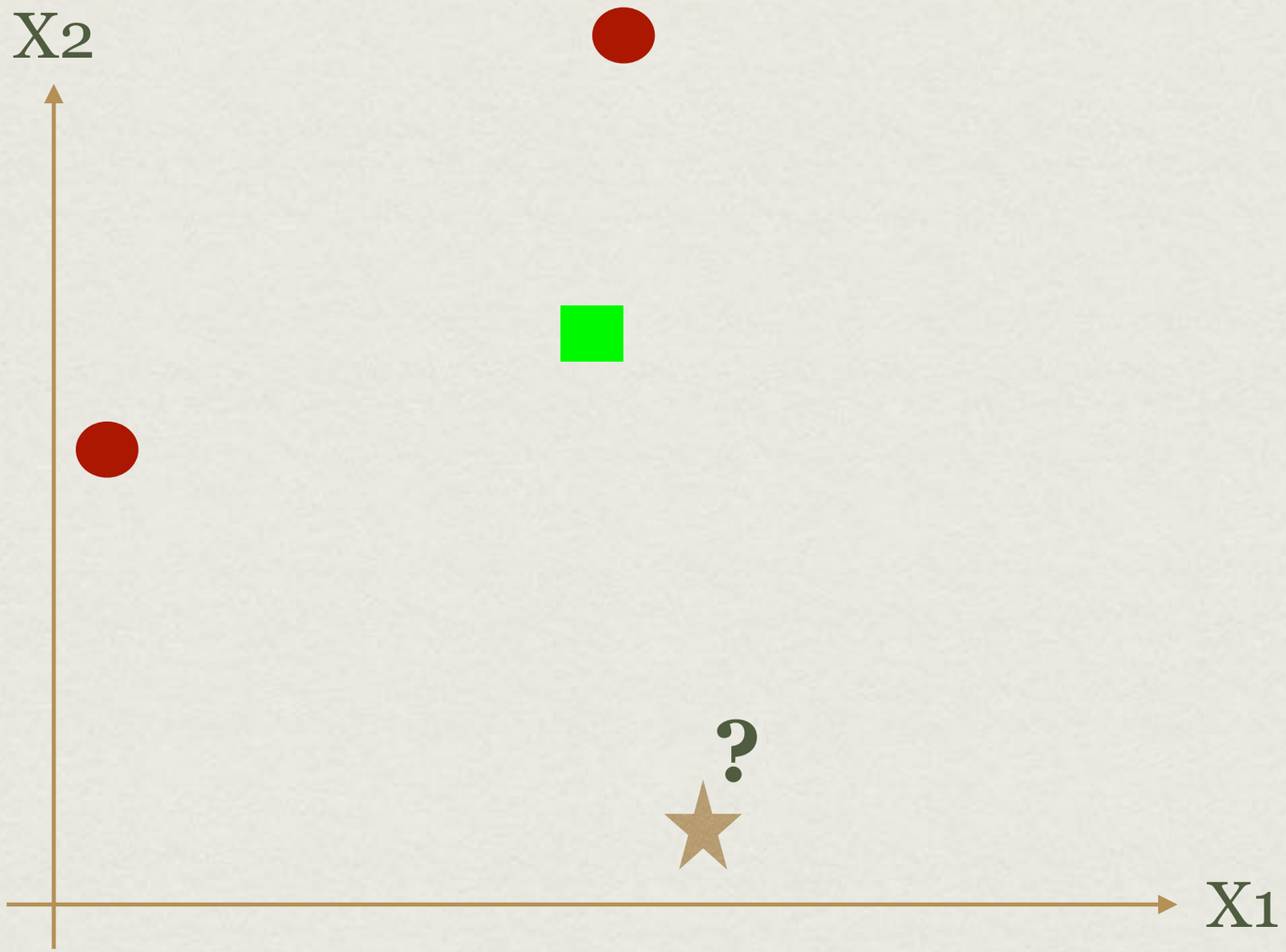
Sample of cats & dogs images from Kaggle Dataset

? 0 = dog
1 = cat

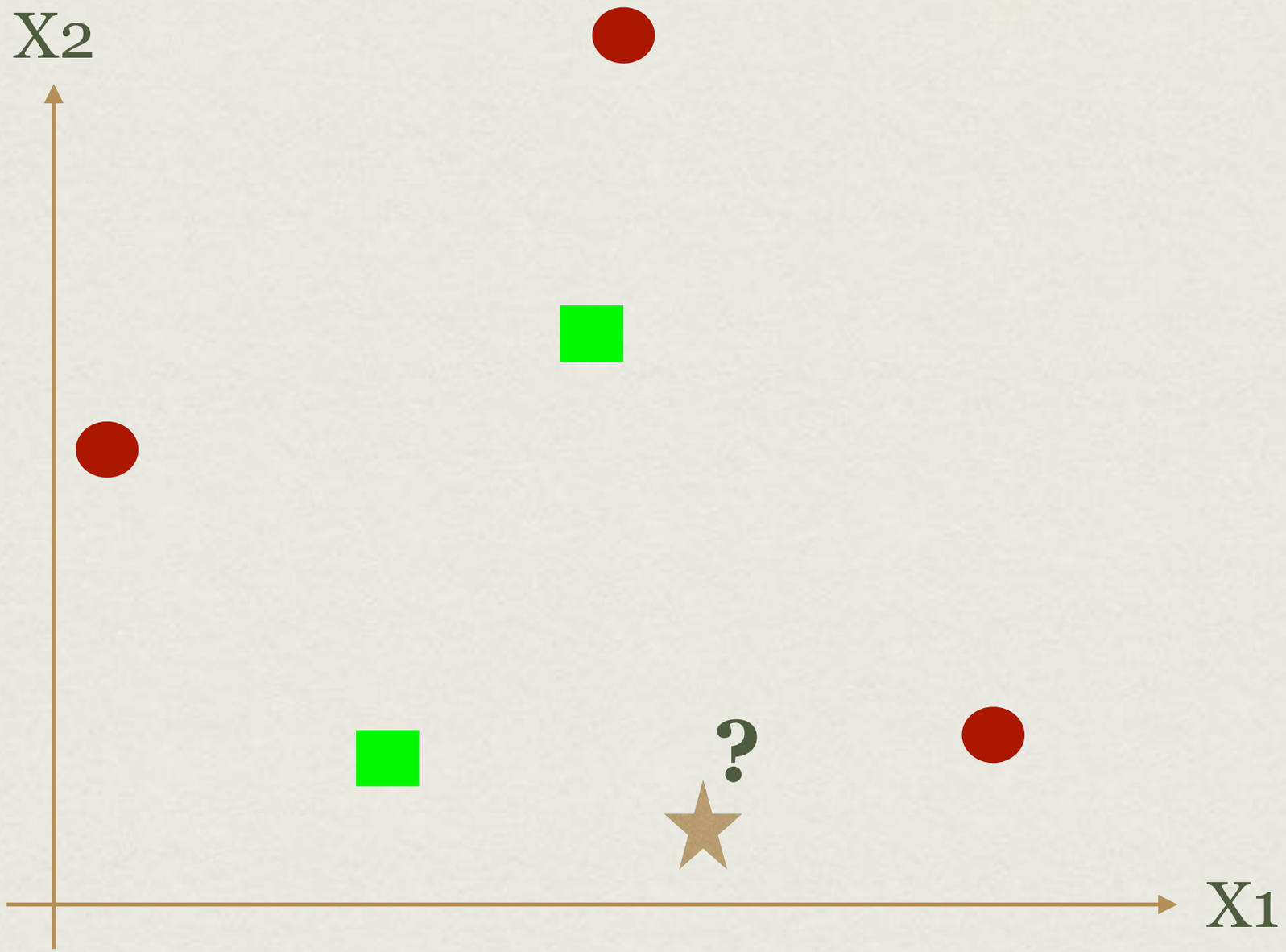
WHAT IS THE RULE?



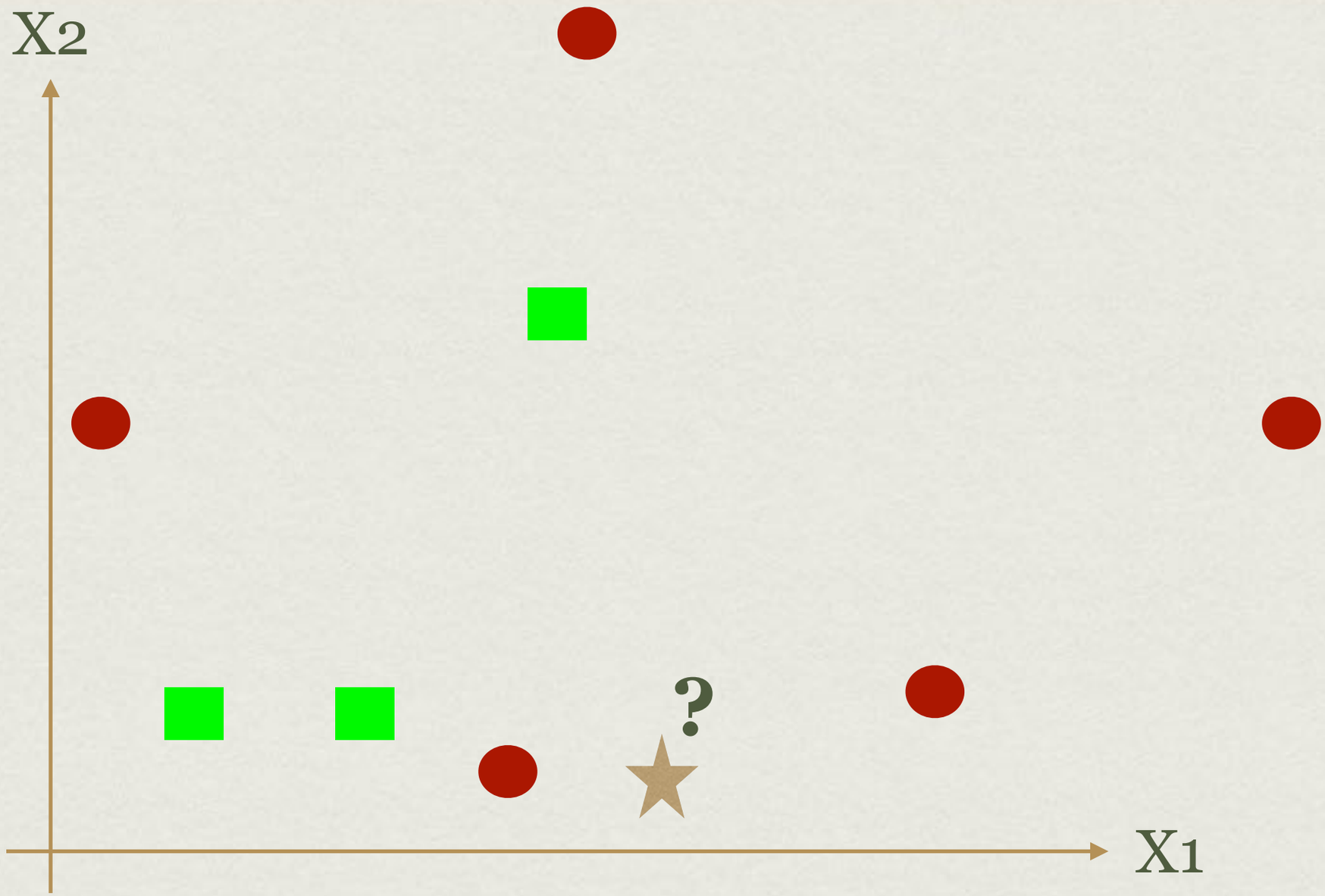
WHAT IS THE RULE?



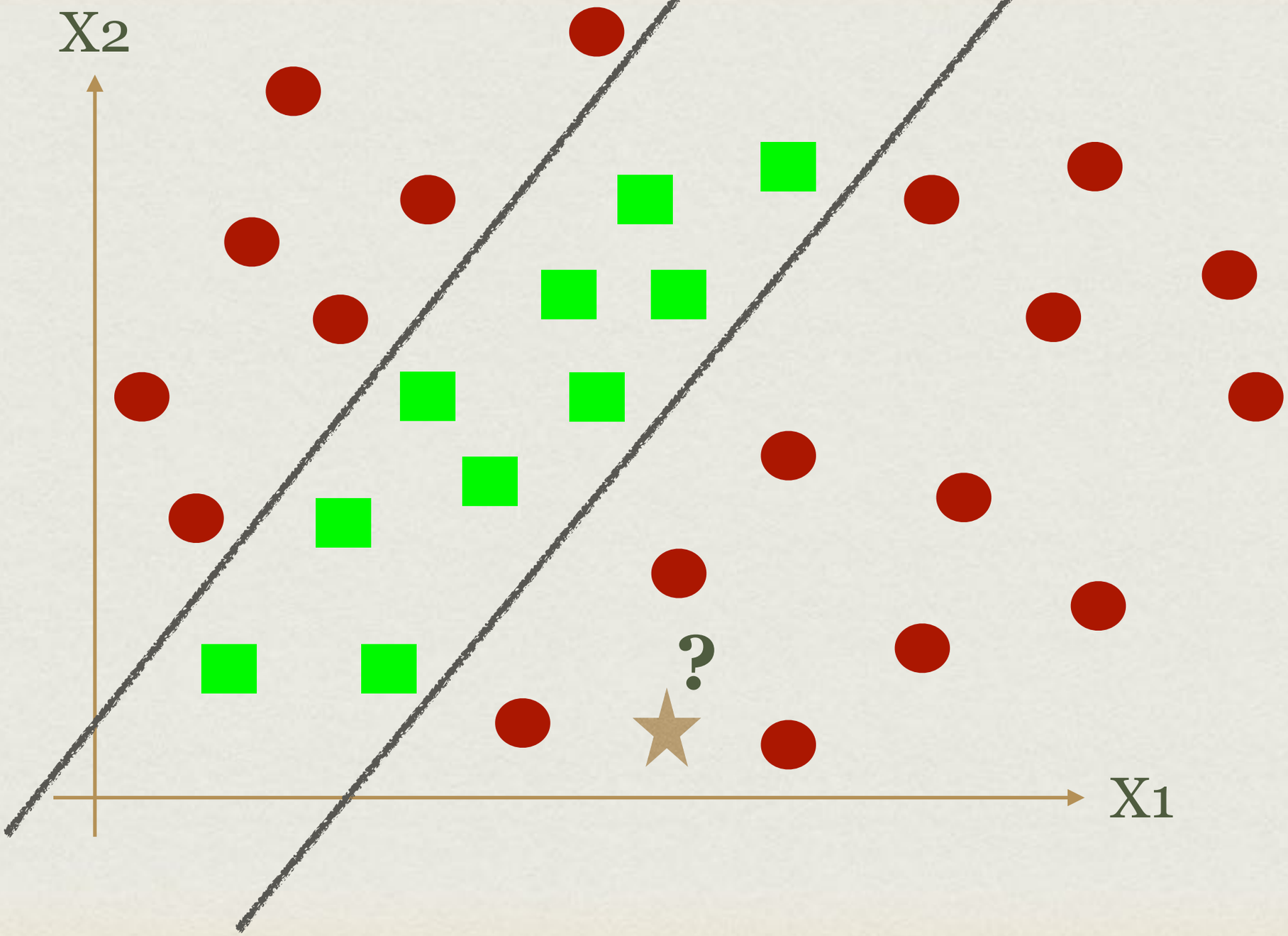
WHAT IS THE RULE?



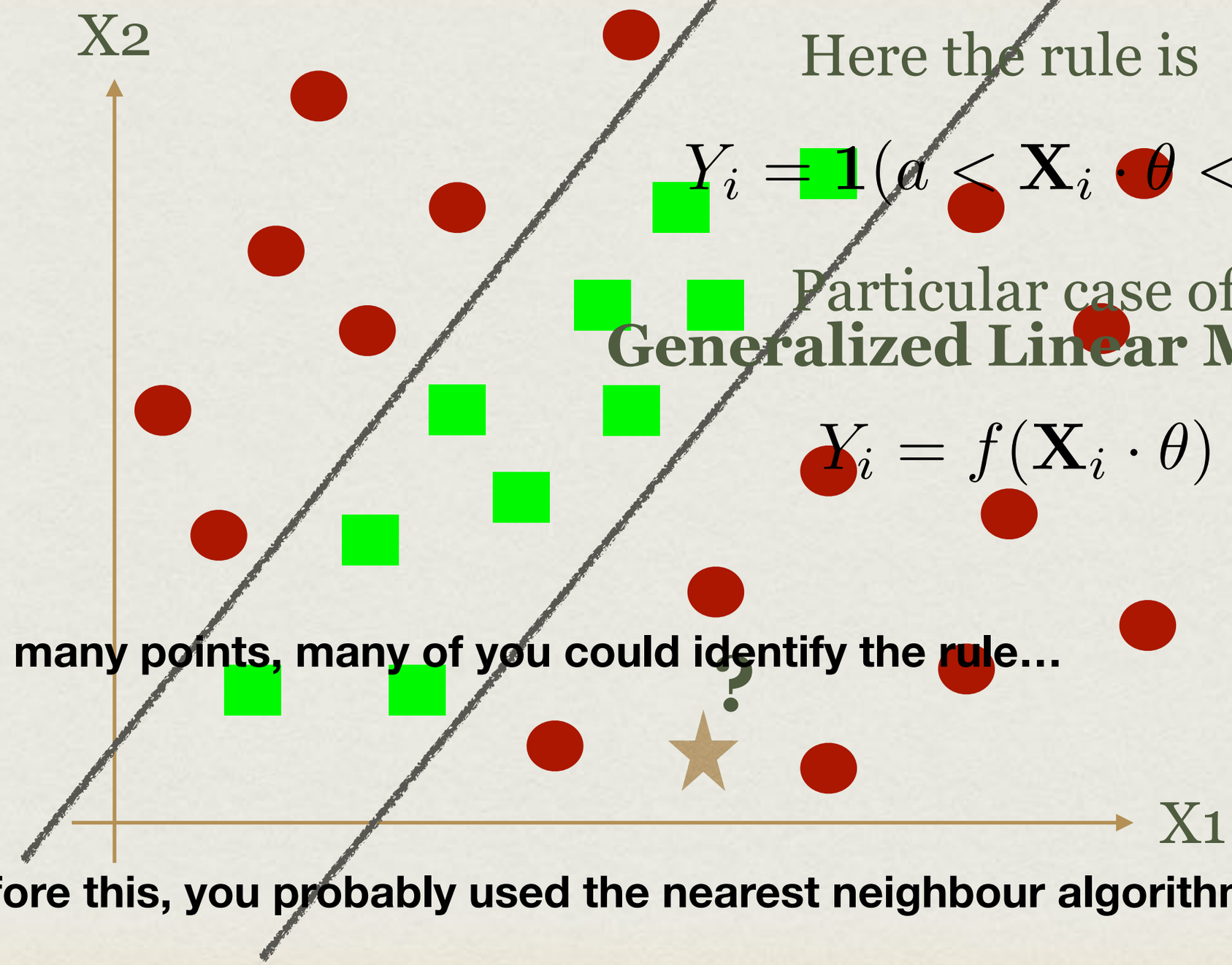
WHAT IS THE RULE?



WHAT IS THE RULE?



WHAT IS THE RULE?



Here the rule is

$$Y_i = \mathbf{1}(a < \mathbf{X}_i \cdot \theta < b)$$

Particular case of

Generalized Linear Models

$$Y_i = f(\mathbf{X}_i \cdot \theta)$$

With many points, many of you could identify the rule…

… but before this, you probably used the nearest neighbour algorithm

Today

i) Knn

ii) a bit of theory

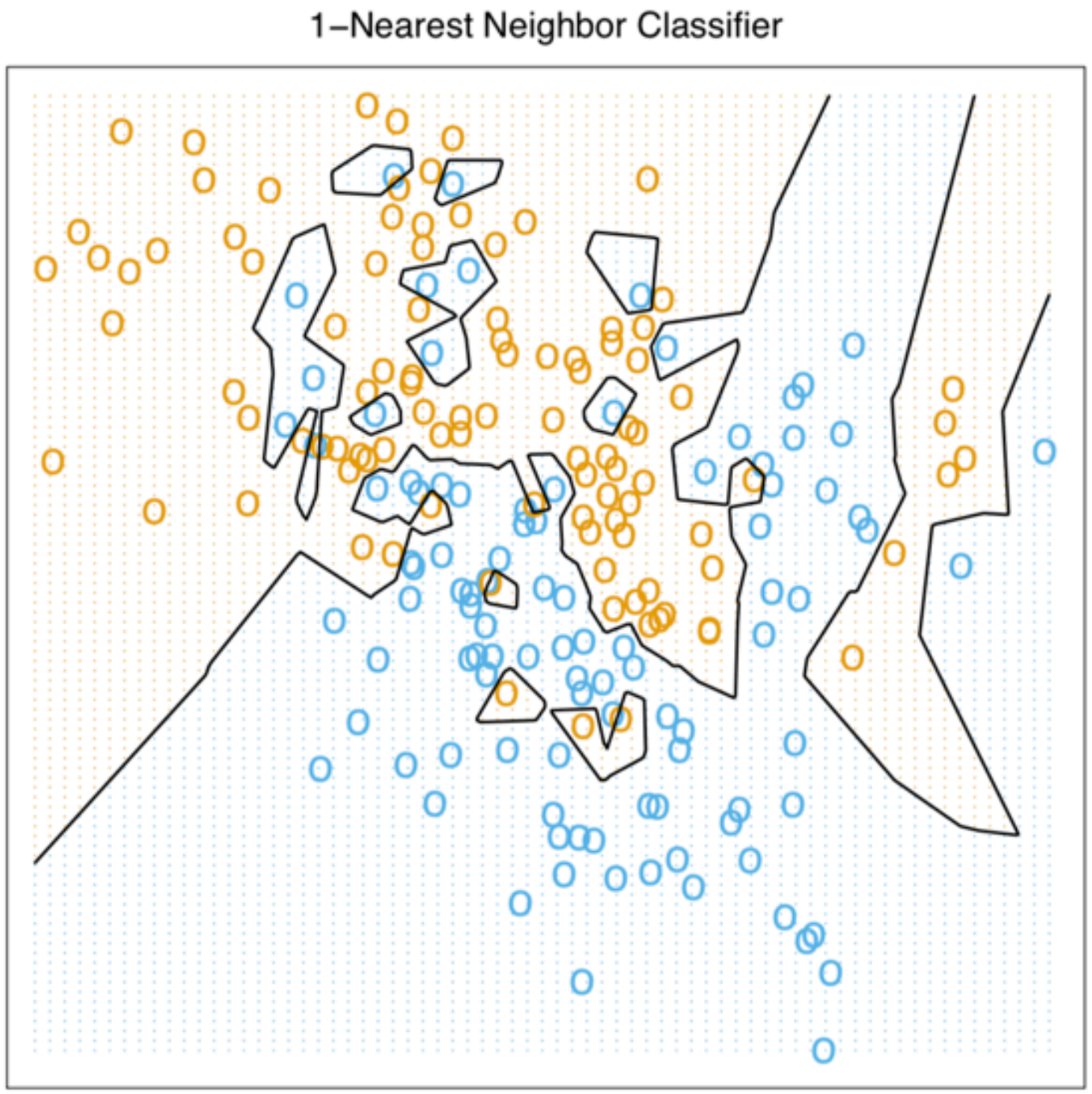
iii) Linear models

iv) Regularization

v) Loss functions

vi) Final remarks

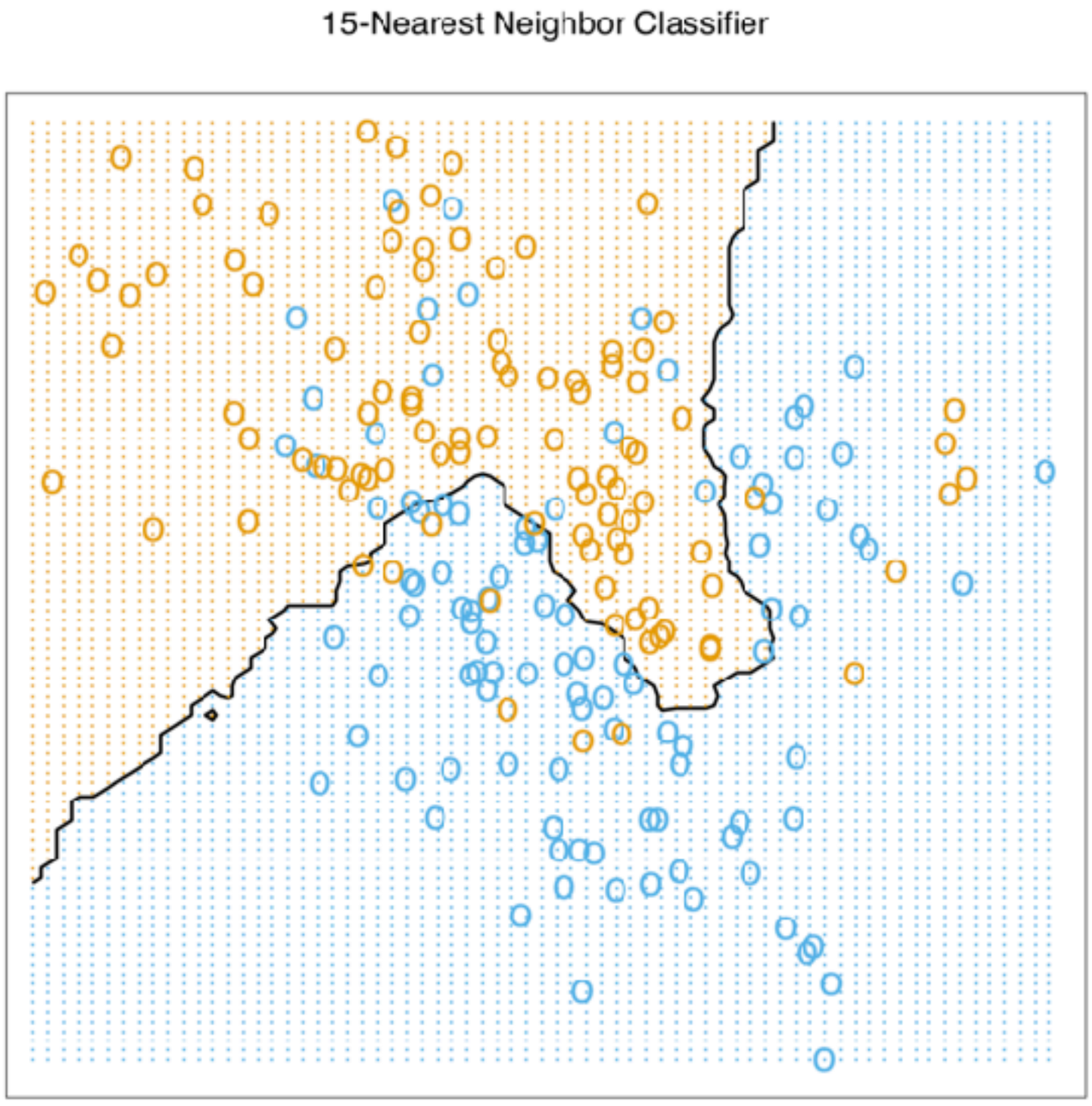
Nearest-Neighbour classifier: Example



- Memorize all labels
- Predict the label of the most similar training point

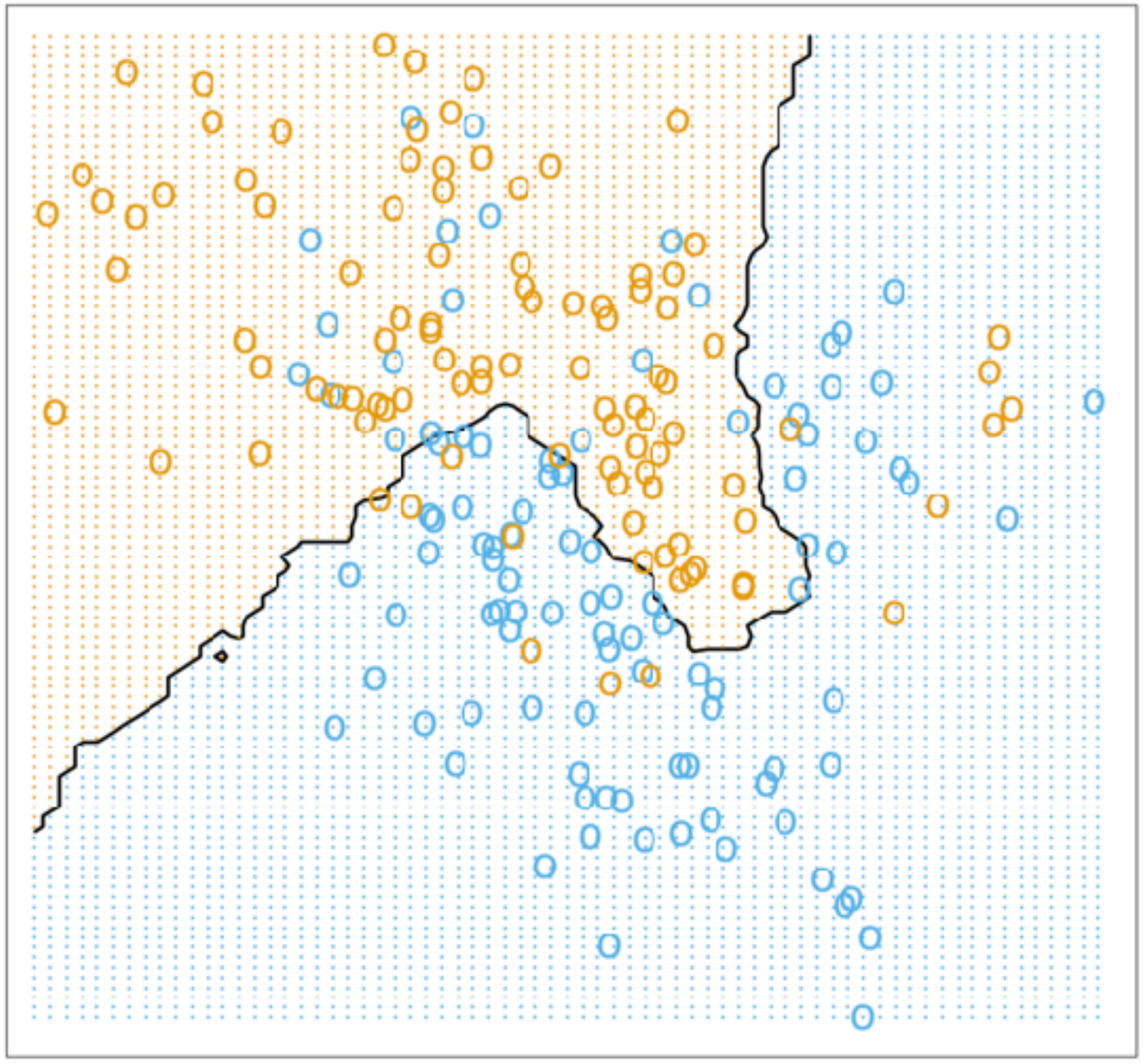
(from Hastie, Tibshirani, Friedman)

Nearest-Neighbour classifier: Example

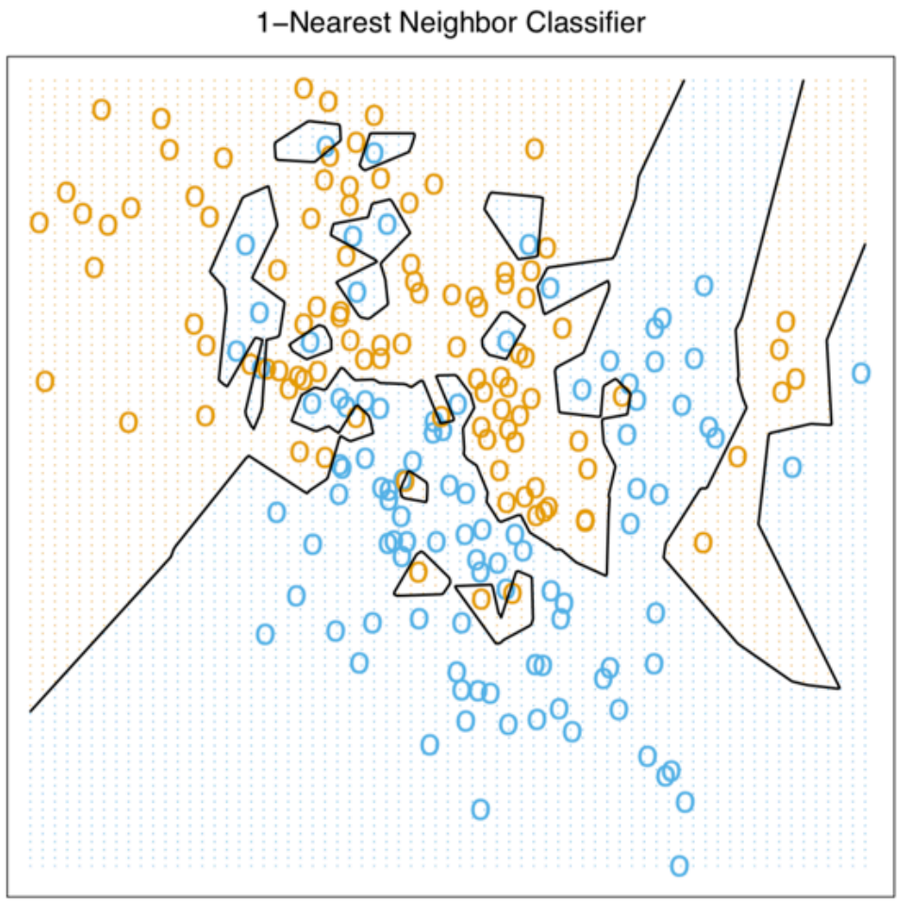


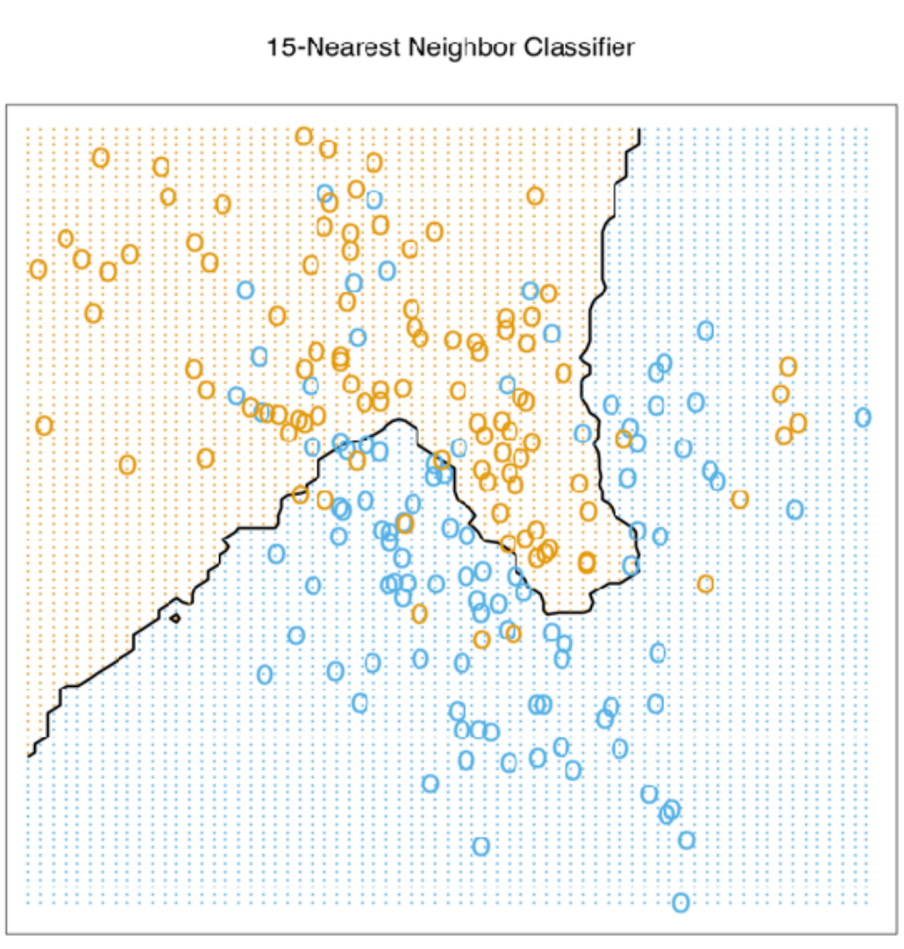
- Memorize all labels
- Predict the label averaging over many similar training points

How to choose k ?

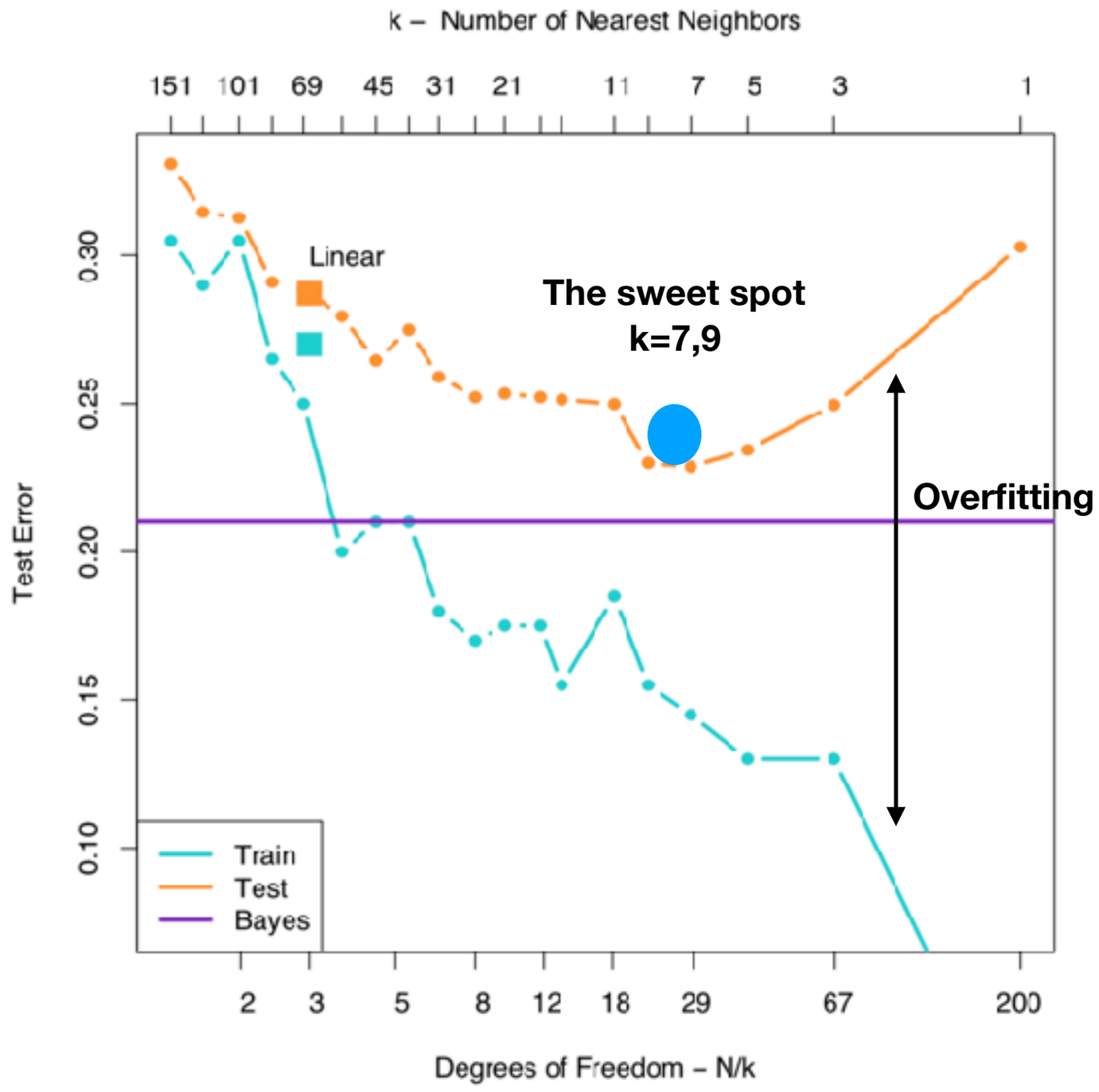


- Memorize all labels
- Predict the label averaging over many similar training points



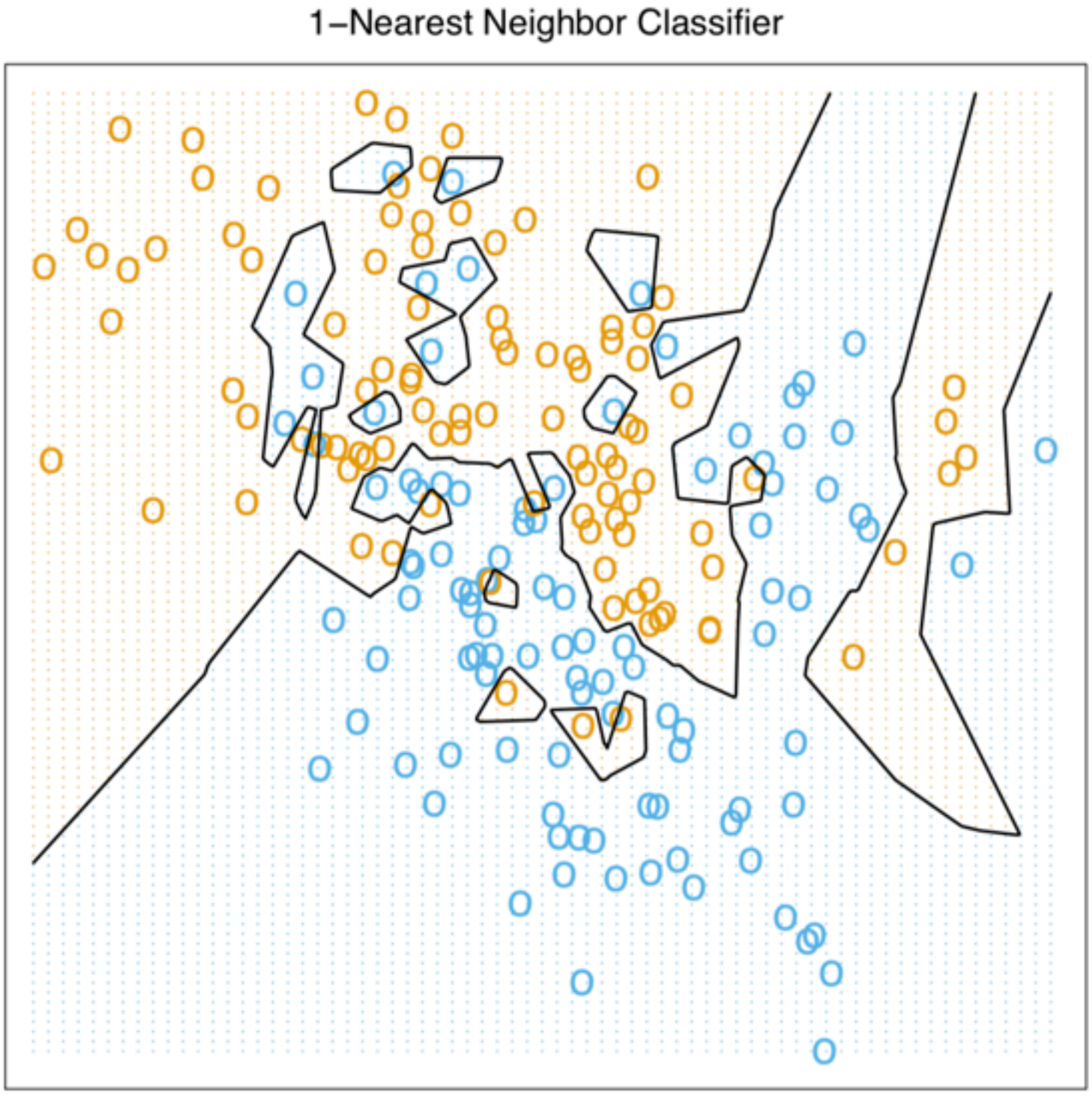


Testing on points not used in the training set « Cross-validation »



Finding « hyper-parameters » requires Cross-validation

Nearest-Neighbour classifiers



Pro:

– Simple to implement
– Flexible to feature/distance choices
– Naturally handles multi-class cases
– Can do well in practice
(with enough representative data)

Con:

– Large search problem
– Storage of data
– Must have a meaningful distance
– Curse of dimensionality

Applications?

K-Nearest Neighbor classification
Examples

- Where in the world?

im2gps [Hays 2008]

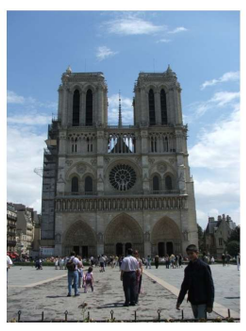
K-Nearest Neighbor classification
Examples

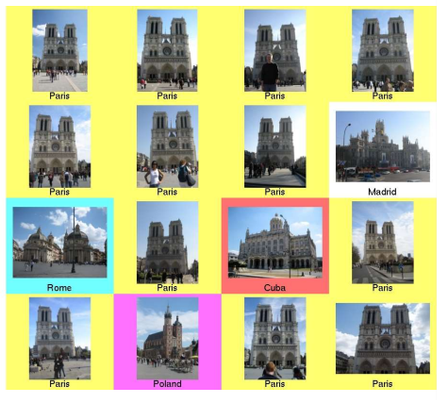
- Where in the world?

im2gps [Hays 2008]

K-Nearest Neighbor classification
Examples

- Where in the world?





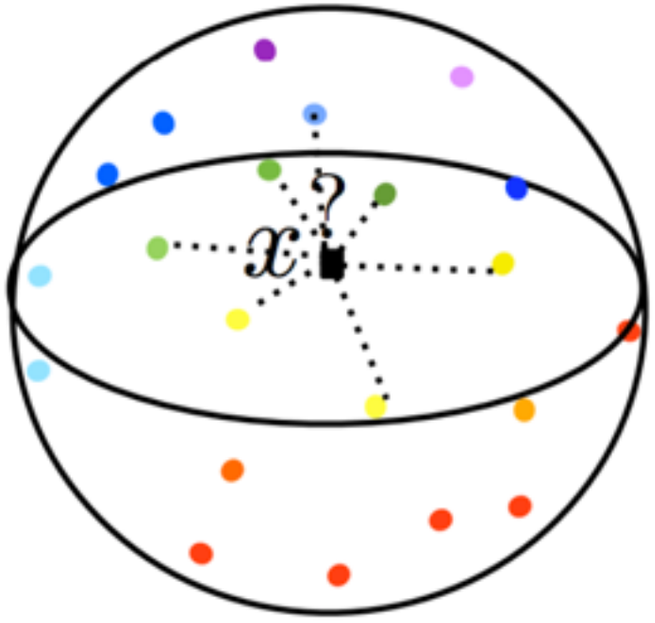
im2gps [Hays 2008]

K-Nearest Neighbor classification
Examples

- Where in the world?

im2gps [Hays 2008]

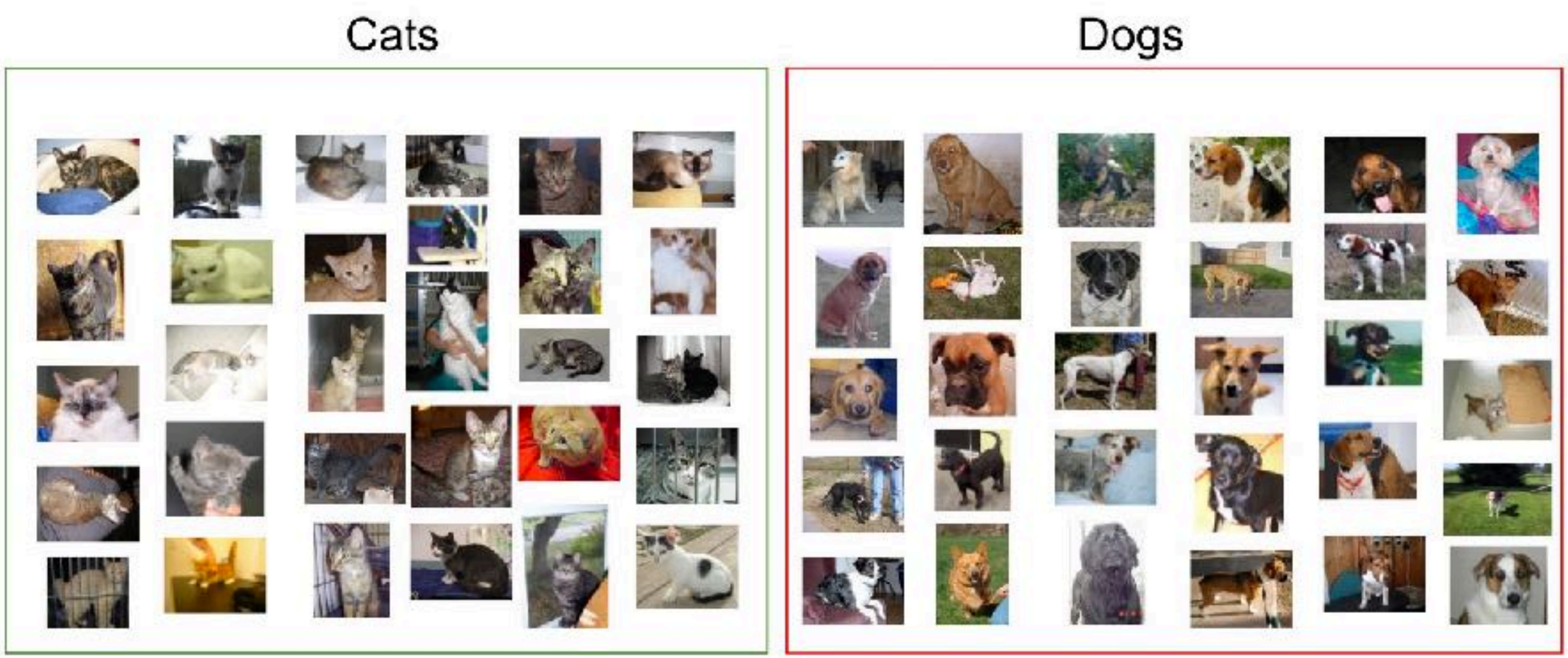
The curse of dimensionality



Need $O(\varepsilon^{-d})$ points to cover $[0, 1]^d$
at a Euclidean distance ε

Images: $d = 10^6$

Distance are always large



Funes the memorious

Knn just memorise everything…
no « understanding/learning » whatsoever!

‘Not only was it difficult for him to understand that the generic term ‘dog’ could embrace so many disparate individuals of diverse size and shapes, it bothered him that the dog seen in profile at 3:14 would be called the same dog at 3:15 seen from the front.’

Without effort, he had learned English, French, Portuguese, Latin. I suspect, nevertheless, that he was not very capable of thought. To think is to forget a difference, to generalize, to abstract. In the overly replete world of Funes there were nothing but details, almost contiguous details.

Jorge Louis Borges, « *Funes the Memorious* » *1942*

Today

i) Knn

ii) a bit of theory

iii) Linear models

iv) Regularization

v) Loss functions

vi) Final remarks

Supervised machine learning

Labeled data: $\begin{cases} \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_n\} & \vec{x} \in \mathbb{R}^d \\ \{y_1, y_2, y_3, \ldots, y_n\} & y \in \mathbb{R} \quad \text{or} \quad \mathrm{y} \in \mathbb{N} \end{cases}$

Goal: Find a function $f_W(\vec{x})$ **that outputs the right class/value for an object** $\vec{x}$

$f_W(\vec{x})$ **has many parameters, denoted** $W \in \mathbb{R}^p$

Ideal : Find $f_W(\vec{x})$ **such that the prediction error is minimal among all unseen vectors**

Supervised machine learning

Labeled data: $\begin{cases} \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_n\} & \vec{x} \in \mathbb{R}^d \\ \{y_1, y_2, y_3, \ldots, y_n\} & y \in \mathbb{R} \quad \text{or} \quad \mathrm{y} \in \mathbb{N} \end{cases}$

Goal: Find a function $f_W(\vec{x})$ **that outputs the right class/value for an object** $\vec{x}$

$f_W(\vec{x})$ **has many parameters, denoted** $W \in \mathbb{R}^p$

Ideal : Find $f_W(\vec{x})$ **such that the prediction error is minimal among all unseen vectors**

Instead : Find $f_W(\vec{x})$ **such that the prediction error is minimal among all vectors in the dataset**

Example for cats and dogs classification

Ideally: find a function

$$f_W(\vec{x})$$

that minimises the error on <u>all</u> possible images of cats and dogs!

Define the <u>loss</u> as

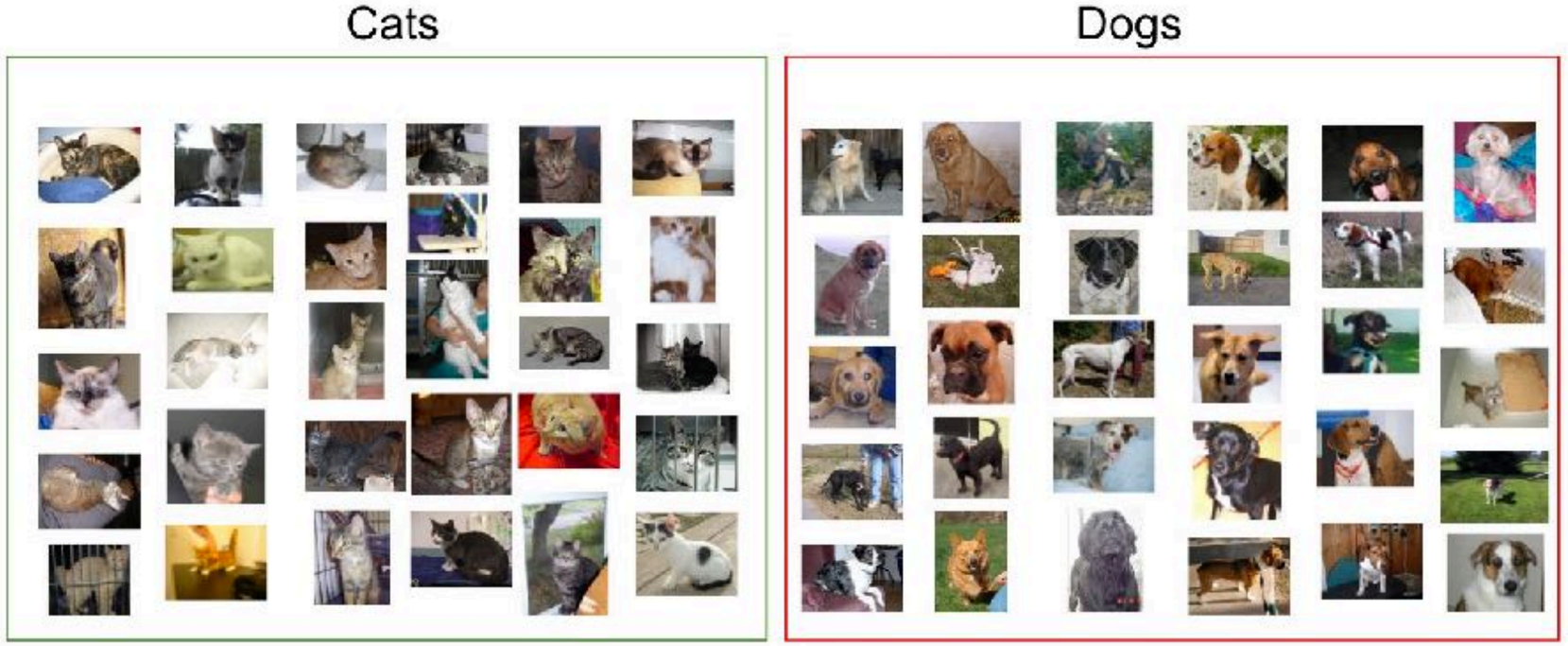
$$\text{loss}(F_W(.), \vec{x}_i, y_i) = (F_W(\vec{x}_i) - y_i)^2$$

We want to minimise the <u>population risk</u> defined as

$$\mathcal{R}_{\text{pop}}(F_W(.)) = \mathbb{E}_{\text{population}}(F_W(\vec{x}) - y)^2$$

Cannot do this: I do not have access to ALL images in the universe, and most of them are not labeled anyway

Example for cats and dogs classification



Sample of cats & dogs images from Kaggle Dataset

Instead: find a function

$$f_W(\vec{x})$$

That minimise the error on <u>all images in the dataset</u>

Define the <u>loss as</u>

$$\text{loss}(F_W(.), \vec{x}_i, y_i) = (F_W(\vec{x}_i) - y_i)^2$$

We want to minimise the empirical risk defined as

$$\mathcal{R}_{\text{empirical}}(F_W(.)) = \frac{1}{N} \sum_{i}^{\text{dataset}} (F_W(\vec{x}_i) - y_i)^2$$

We are actually minimizing the wrong function (but we have no choice)

The workhorse: Empirical Risk Minimisation

Minimize

$$\mathcal{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_{i}^{\text{dataset}} \ell(W, (\vec{x}_i), y_i)$$

Rationale: it should be close to

$$\mathcal{R}_{\text{population}}(W) = \mathbb{E}\ell(W, (\vec{x}), y)$$

Indeed, the law of large number implies that $\lim_{n\to\infty} \mathcal{R}_{\text{empirical}} \to \mathcal{R}_{\text{population}}$

QUESTIONS

- **Q1**: How close it the empirical risk to the population risk?
- **Q2:** How do we minimise the empirical risk?

A back-of-the-envelope bound

Call ε the fraction of error

$$\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$$
$$f(\,.\,) \in \mathscr{F}$$

$$\epsilon_{\text{pop}} = \epsilon_{\text{empirical}} + \Delta$$

What is the worst that can happen ?

We are looking for a rule while there *is no rule* and the labels are actually random!

$\epsilon^{\text{random}}_{\text{empirical}}$ **can be optimised…** **… but** $\epsilon^{\text{random}}_{\text{pop}} = \frac{1}{2}$

So, in reality, we expect:

$$\epsilon_{\text{pop}} - \epsilon_{\text{empirical}} \leq \epsilon^{\text{random}}_{\text{pop}} - \epsilon^{\text{random}}_{\text{empirical}}$$

A back-of-the-envelope bound

Call ε the fraction of error

$$\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$$
$$f(\,.\,) \in \mathscr{F}$$

$$\epsilon_{\text{pop}} = \epsilon_{\text{empirical}} + \Delta$$

What is the worst that can happen ?

We are looking for a rule while there *is no rule* and the labels are actually random!

$\epsilon^{\text{random}}_{\text{empirical}}$ **can be optimised…** **… but** $\epsilon^{\text{random}}_{\text{pop}} = \frac{1}{2}$

So, in reality, we expect:

$$\epsilon_{\text{pop}} - \epsilon_{\text{empirical}} \leq \epsilon^{\text{random}}_{\text{pop}} - \epsilon^{\text{random}}_{\text{empirical}} = \frac{1}{2}(1 - 2\epsilon^{\text{random}}_{\text{empirical}})$$

$\mathscr{R}_n(\{\mathbf{x}\})$**is the empirical Rademacher complexity: it tells how well your method can fit random labels**

A back-of-the-envelope bound

Call ε the fraction of error

$$\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$$
$$f(\,.\,) \in \mathscr{F}$$

$$\epsilon_{\text{pop}} = \epsilon_{\text{empirical}} + \Delta$$

What is the worst that can happen ?

We are looking for a rule while there *is no rule* and the labels are actually random!

$\epsilon^{\text{random}}_{\text{empirical}}$ **can be optimised…** **… but** $\epsilon^{\text{random}}_{\text{pop}} = \frac{1}{2}$

So, in reality, we expect:

$$\epsilon_{\text{pop}} - \epsilon_{\text{empirical}} \leq \epsilon^{\text{random}}_{\text{pop}} - \epsilon^{\text{random}}_{\text{empirical}} = \frac{1}{2}(1 - 2\epsilon^{\text{random}}_{\text{empirical}}) = \frac{1}{2}\hat{\mathscr{R}}_n\{(\mathbf{x})\}$$

$\mathscr{R}_n(\{\mathbf{x}\})$**is the empirical Rademacher complexity: it tells how well your method can fit random labels**

Rademacher complexity

Consider :

1) A function in a function class $f \in \mathscr{F}$
2) N Datapoints Xi, i=1…..N
3) ε taken from +1 -1 uniformly at random

$$\mathcal{R}_n(\mathscr{F}) := \mathbb{E}_X\left[\mathcal{R}(\mathscr{F}(X_1^n)/n)\right] \;=\; \mathbb{E}_{X,\varepsilon}\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i f(X_i)\right|\right]$$

Uniform convergence

$$\mathcal{R}_n(\mathscr{F}) := \mathbb{E}_X\left[\mathcal{R}(\mathscr{F}(X_1^n)/n)\right] \;=\; \mathbb{E}_{X,\varepsilon}\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i f(X_i)\right|\right]$$

Choose loss = #of misclassified

$$\sup_f \left|\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)\right] \leq \mathscr{R}_n(\mathscr{F})$$

Uniform convergence

$$\mathcal{R}_n(\mathscr{F}) := \mathbb{E}_X\left[\mathcal{R}(\mathscr{F}(X_1^n)/n)\right] \;=\; \mathbb{E}_{X,\varepsilon}\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i f(X_i)\right|\right]$$

Choose loss = #of misclassified

$$\sup_f \left|\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)\right] \leq \mathscr{R}_n(\mathscr{F})$$

VC dimension
Can bound Rademacher

Uniform convergence

$$\mathcal{R}_n(\mathscr{F}) := \mathbb{E}_X\left[\mathcal{R}(\mathscr{F}(X_1^n)/n)\right] \;=\; \mathbb{E}_{X,\varepsilon}\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i f(X_i)\right|\right]$$

Choose loss = #of misclassified

$$\sup_f \left|\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)\right] \leq \mathscr{R}_n(\mathscr{F})$$

VC dimension
Can bound Rademacher

$$\mathscr{R}_n(\mathscr{F}) \leq C\sqrt{\frac{d_{\mathrm{VC}}}{N}}$$

Uniform convergence

$$\mathcal{R}_n(\mathscr{F}) := \mathbb{E}_X\left[\mathcal{R}(\mathscr{F}(X_1^n)/n)\right] \;=\; \mathbb{E}_{X,\varepsilon}\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i f(X_i)\right|\right]$$

Choose loss = #of misclassified

$$\sup_f \left|\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)\right] \leq \mathscr{R}_n(\mathscr{F})$$

VC dimension
Can bound Rademacher

(Vapnik–Chervonenkis)

$$\mathscr{R}_n(\mathscr{F}) \leq C\sqrt{\frac{d_{\mathrm{VC}}}{N}}$$

$$\sup_f \left|\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)\right] \leq C'\sqrt{\frac{d_{\mathrm{VC}}}{n}}$$

Statistics: How close is the empirical risk to the population risk ?

Generalization bound

Theorem (Vapnik, Chervonenkis, 1968; . . .)

Under conditions [omitted], with high probability

$$\sup_f |\,\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)] \leq \mathrm{Cst}\sqrt{\frac{d_{\mathrm{vc}}}{n}}$$

VC dimension = capacity of your classifier

(number of points it can still classify perfectly in the worst case)

Error on unseen data = **Error on training data** + **Additional term**

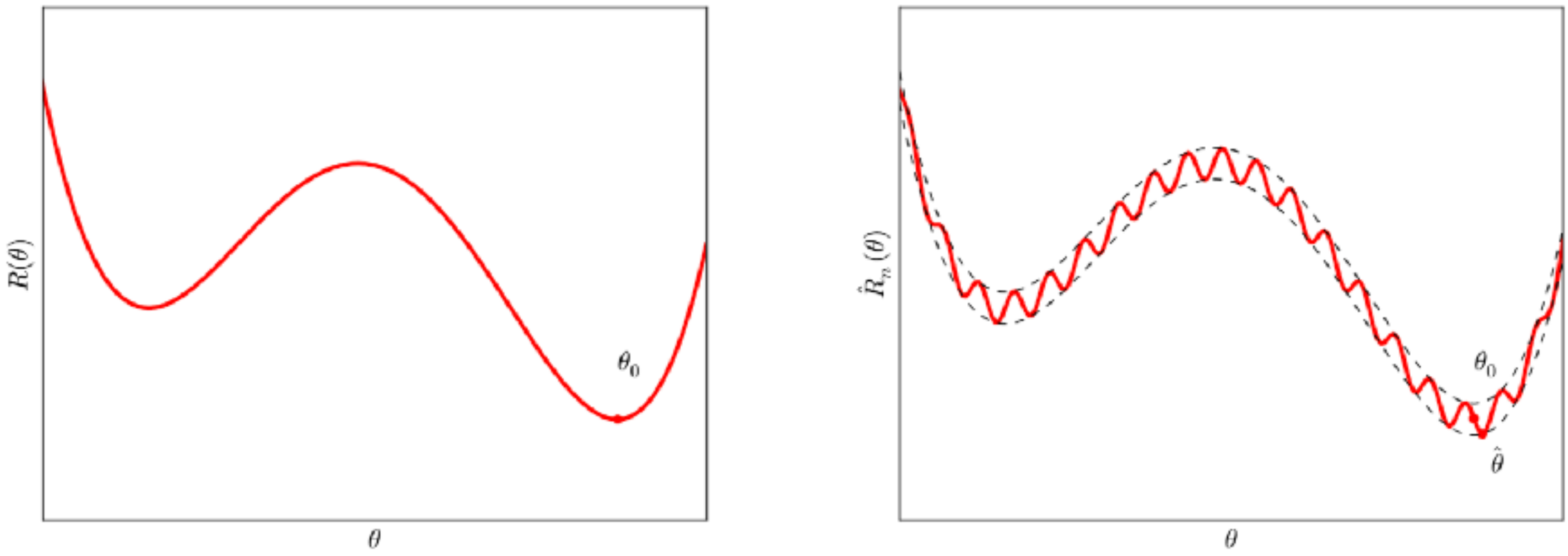
Decay with number of training sample

Increase with complexity of function f

Generalisation error — **Training error**

Statistics: How close is the empirical risk to the population risk ?

Uniform convergence



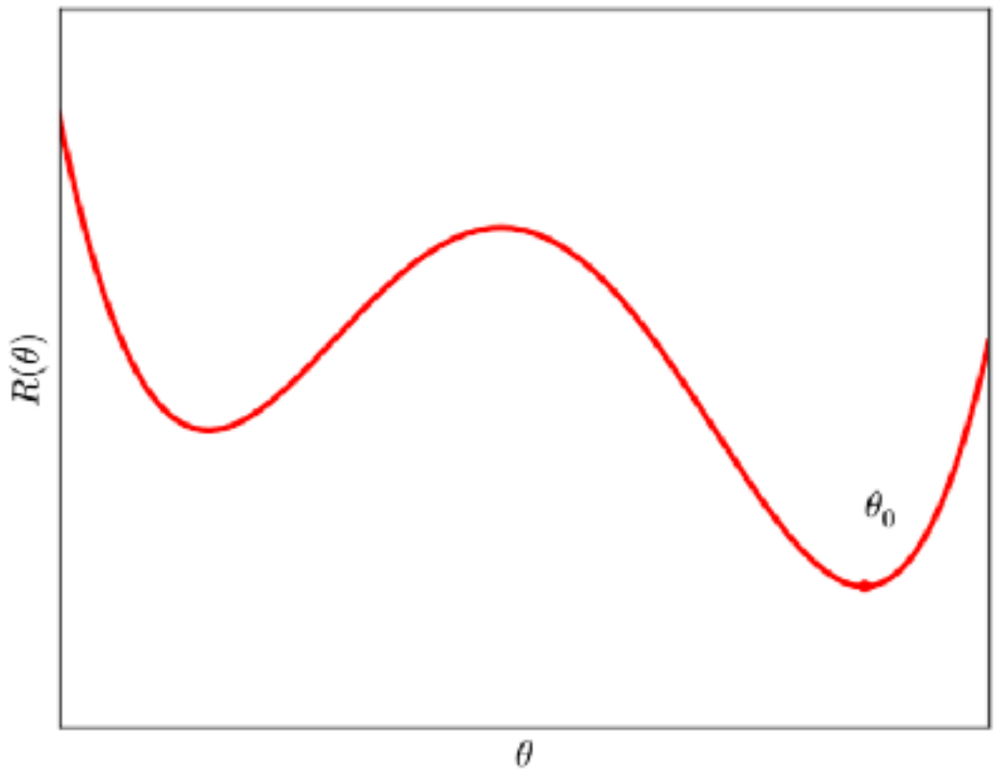
Population risk

Empirical risk

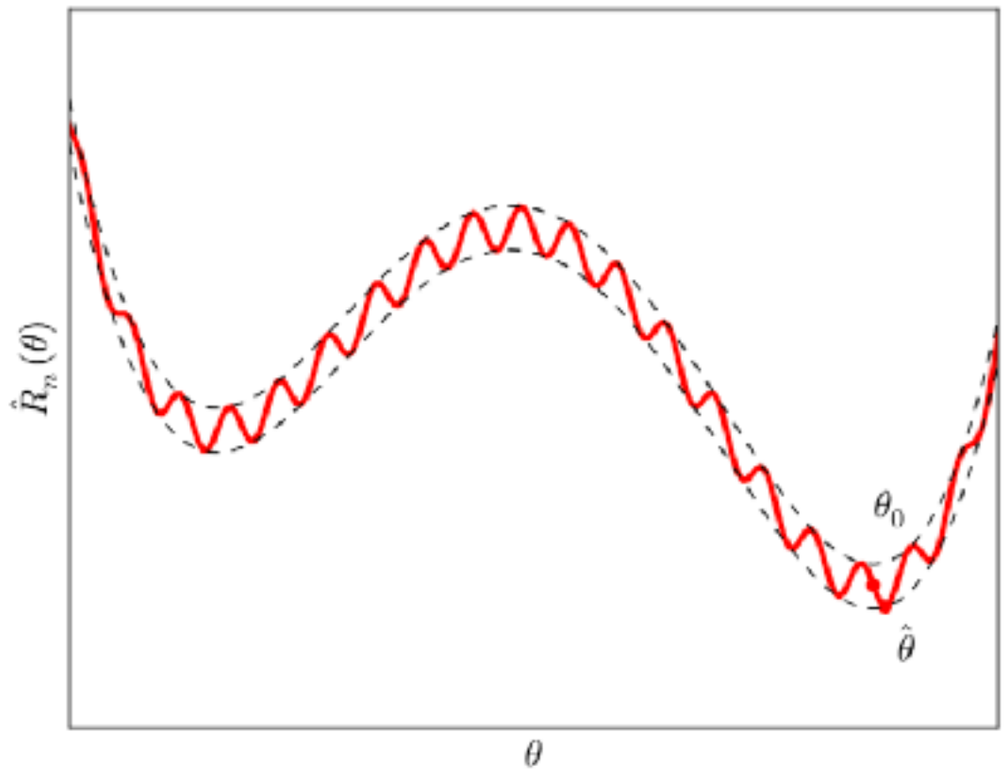
Statistics: How close is the empirical risk to the population risk ?

Uniform convergence

If there is enough data, one can also show that the derivatives get close!



Population risk

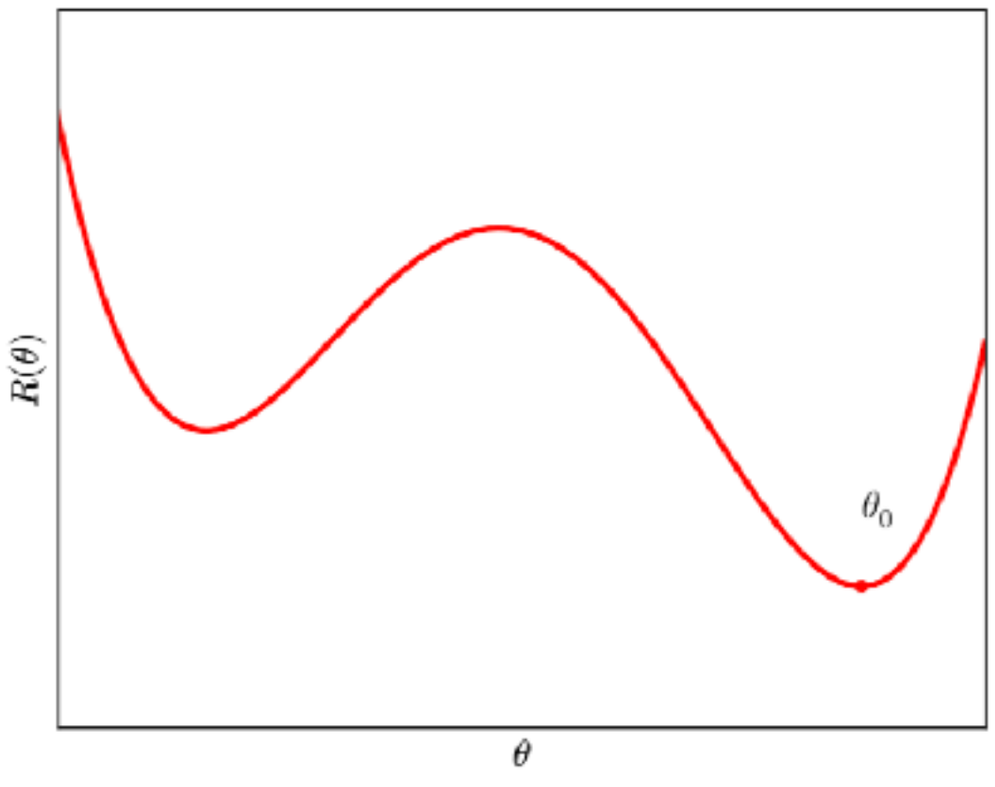


Empirical risk

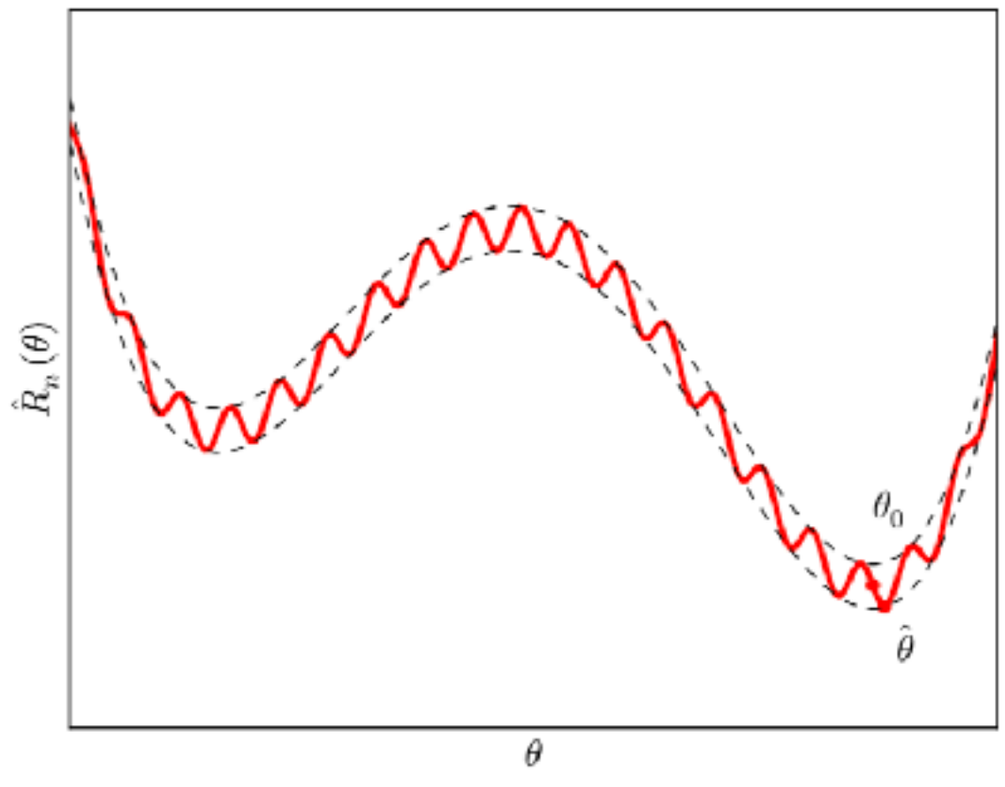
Statistics: How close is the empirical risk to the population risk ?

This cannot happen!

If there is enough data, one can also show that the derivatives get close!



Population risk

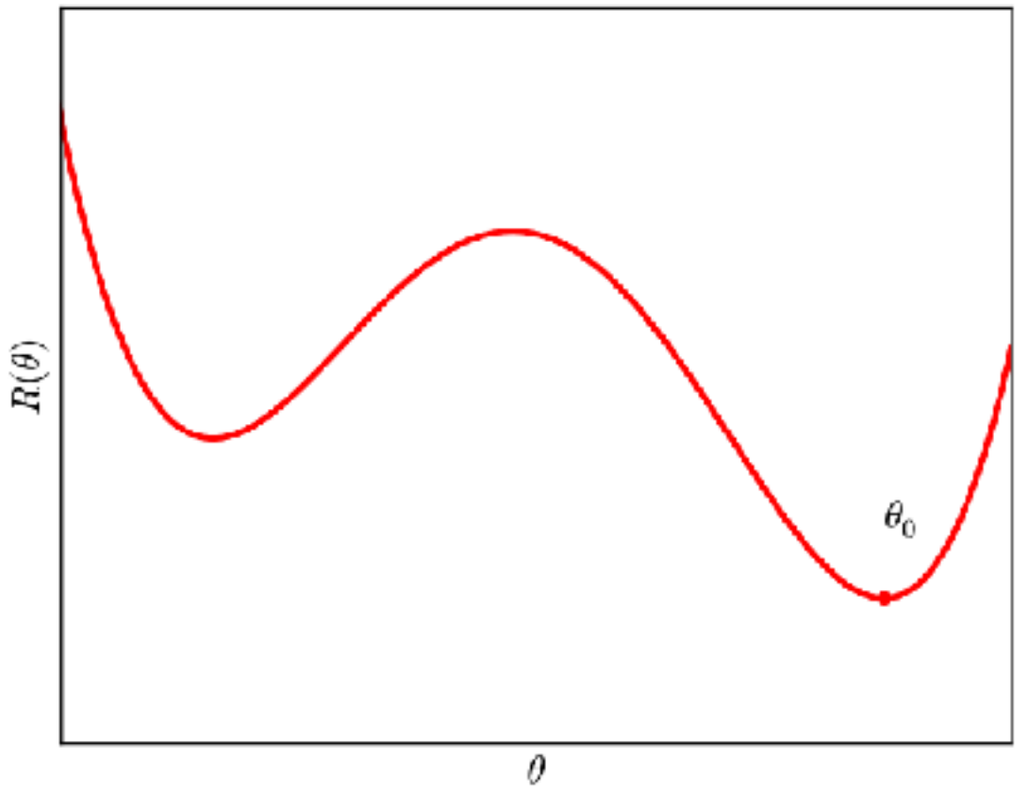


Empirical risk

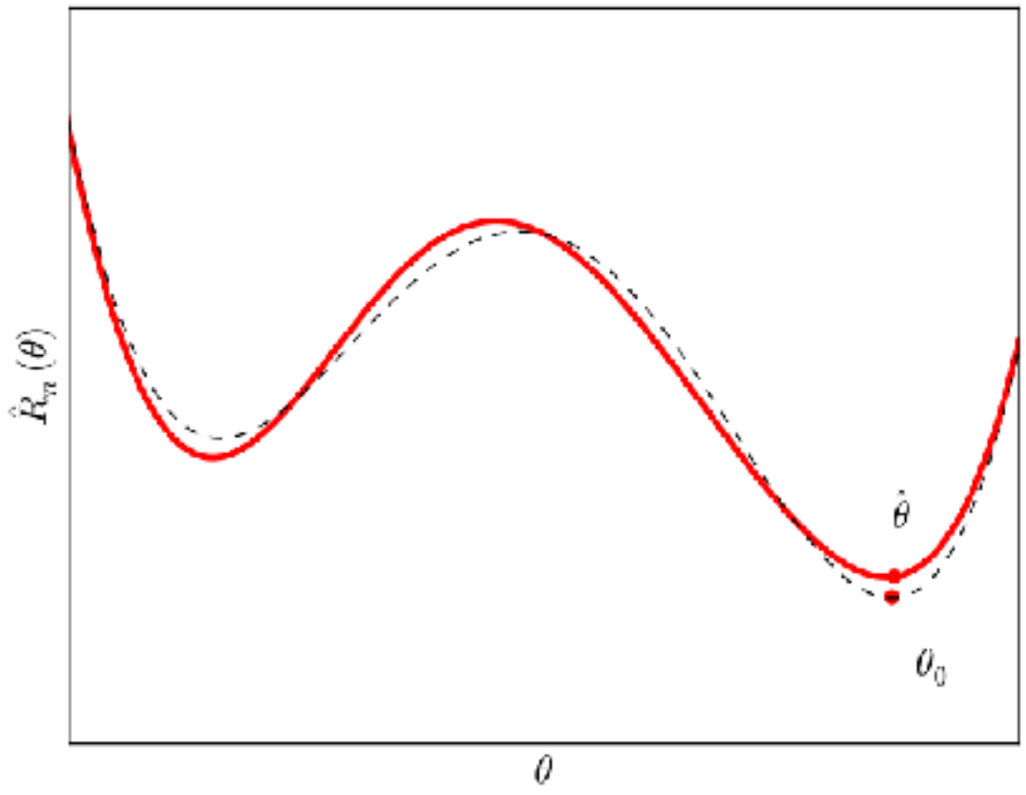
Statistics: How close is the empirical risk to the population risk ?

This can happen!

If there is enough data, one can also show that the derivatives get close!



Population risk



Empirical risk

Nice population risk ➡️ Nice empirical risk

Statistics: How close is the empirical risk to the population risk ?

Generalization bound

Theorem (Vapnik, Chervonenkis, 1968; . . .)

Under conditions [omitted], with high probability

$$\sup_f \left| \mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f) \right] \leq \mathrm{Cst} \sqrt{\frac{d_{\mathrm{vc}}}{n}}$$

VCd dimension = capacity of your classifier

(number of points it can still classify perfectly in the worst case)

Error on unseen data = **Error on training data** + **Additional term**

Decay with number of training sample

Increase with complexity of function f

Generalisation error — **Training error**

Statistics: How close is the empirical risk to the population risk ?

Generalization bound

Theorem (Vapnik, Chervonenkis, 1968; . . .)

Under conditions [omitted], with high probability

$$\sup_f |\,\mathrm{Risk}_{\mathrm{empirical}}(f) - \mathrm{Risk}_{\mathrm{population}}(f)] \leq \mathrm{Cst}\sqrt{\frac{d_{\mathrm{vc}}}{n}}$$

VCd dimension = capacity of your classifier

(number of points it can still classify perfectly in the worst case)

In practice, we are working often with n much lower than VC,
in which case these results are useless (*this is the case for deep learning*)
No good theory for this problem (*yet*)

Zhang *et al*. UNDERSTANDING DEEP LEARNING REQUIRES RE- THINKING GENERALIZATION , arxiv:1611.03530

<u>Statistics</u>: How close is the empirical risk to the population risk ?

> Those results are cool and beautiful, but they have no practical consequence. No one uses generalization bounds.

Yann LeCun, Facebook AI

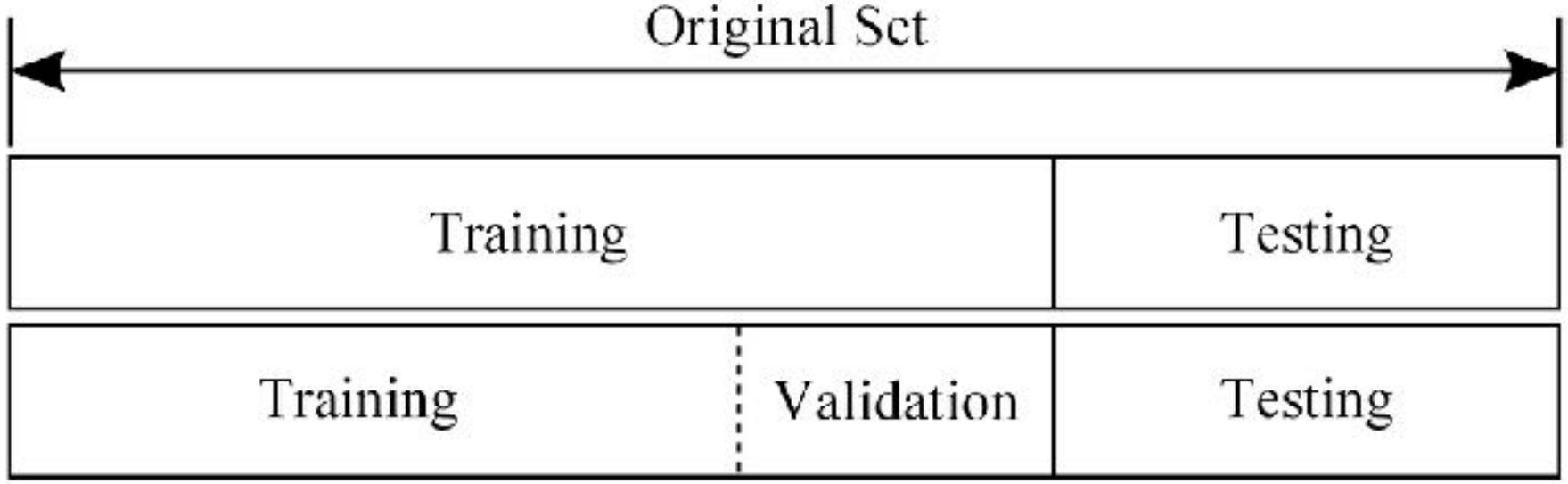
$$\underbrace{\text{Error on unseen data}}_{\text{Generalisation error}} = \underbrace{\text{Error on training data}}_{\text{Training error}} + \text{Additional term}$$

Decay with number of training sample

Increase with complexity of function f

In practice

Divide the labelled set into training, validation and testing sets



* **Training set: used to train the classifier**
* **Validation set (optional): choose between different methods, finite-tune parameters,**
* **Testing set: predict the generalization error**

No cheat: do not use the test set to train your algorithm!

No cheat: do not use the test set to train your algorithm!

MIT Technology Review

Log in

Topics+ The Download Magazine

A View from **Tom Simonite**

Why and How Baidu Cheated an Artificial Intelligence Test

Machine learning gets its first cheating scandal.

Quora

Search for questions, people, and topics

Quora uses cookies to improve your experience. Read mo

Rewrite Regression (statistics) Statistics (academic discipline)

How much reliable is the paper "Stacked Approximated Regression Machine"? Are we going to rewrite DL frameworks?

1 Answer

Zhaojun Zhang, Trained as a Bayesian for two years
Answered Sep 10, 2016

From Arxiv, the paper has been withdrawn: A Simple Deep Learning Approach.

So, it is unlikely that we are going to rewrite DL frameworks based on this paper.

326 Views

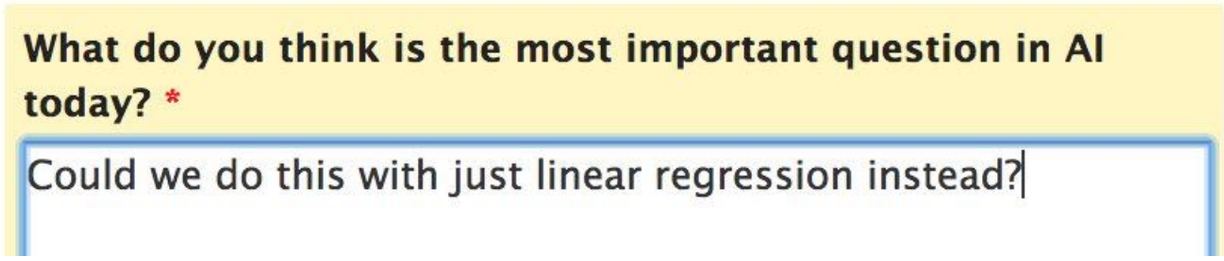
Today

i) Knn
ii) a bit of theory
iii) Linear models
iv) Regularization
v) Loss functions
vi) Final remarks

What do you think is the most important question in AI today? *

Could we do this with just linear regression instead?

The linear model



$$f(\mathbf{X}_i) = \theta \cdot \mathbf{X}_i + \alpha \qquad \theta \in \mathbb{R}^d \quad \alpha \in \mathbb{R}$$

Or equivalently

$$f(\mathbf{X}_i) = \theta' \cdot \mathbf{X}'_i \qquad \theta' = \begin{bmatrix} \theta \\ \alpha \end{bmatrix} \quad \mathbf{X}' = \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Quadratic loss

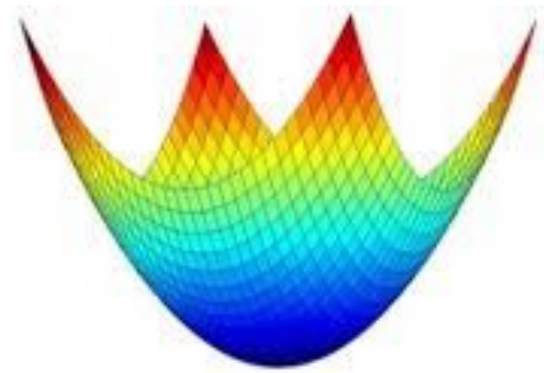
$$f(\mathbf{X}_i) = \theta \cdot \mathbf{X}_i \qquad \theta \in \mathbb{R}^d$$

Loss : $$\ell(y_i, f(\mathbf{X}_i)) = (y_i - \mathbf{X}_i \cdot \theta)^2$$

Risk : $$\mathcal{R} = \frac{1}{n}\sum_i (y_i - \mathbf{X}_i \cdot \theta)^2$$

Mean square Error (MSE)

The risk is convex in θ :
only one minima, easy minimisation



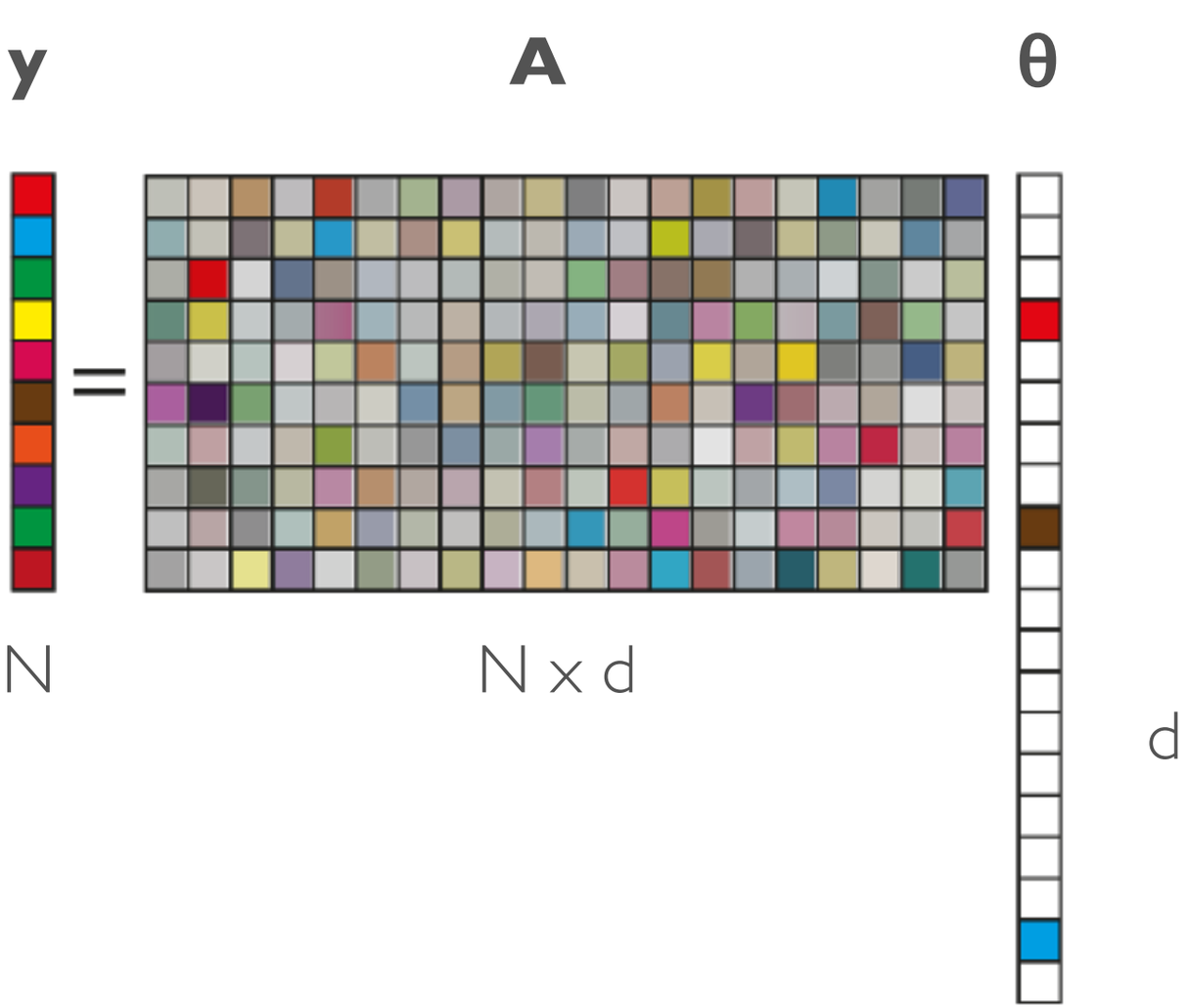
Linear algebra

Risk : $$\mathcal{R} = \frac{1}{n}\sum_i (y_i - \mathbf{X}_i \cdot \theta)^2$$

Mean square Error (MSE)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

$$A = \begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \dots \\ \mathbf{X}_n^T \end{bmatrix}$$



Linear algebra

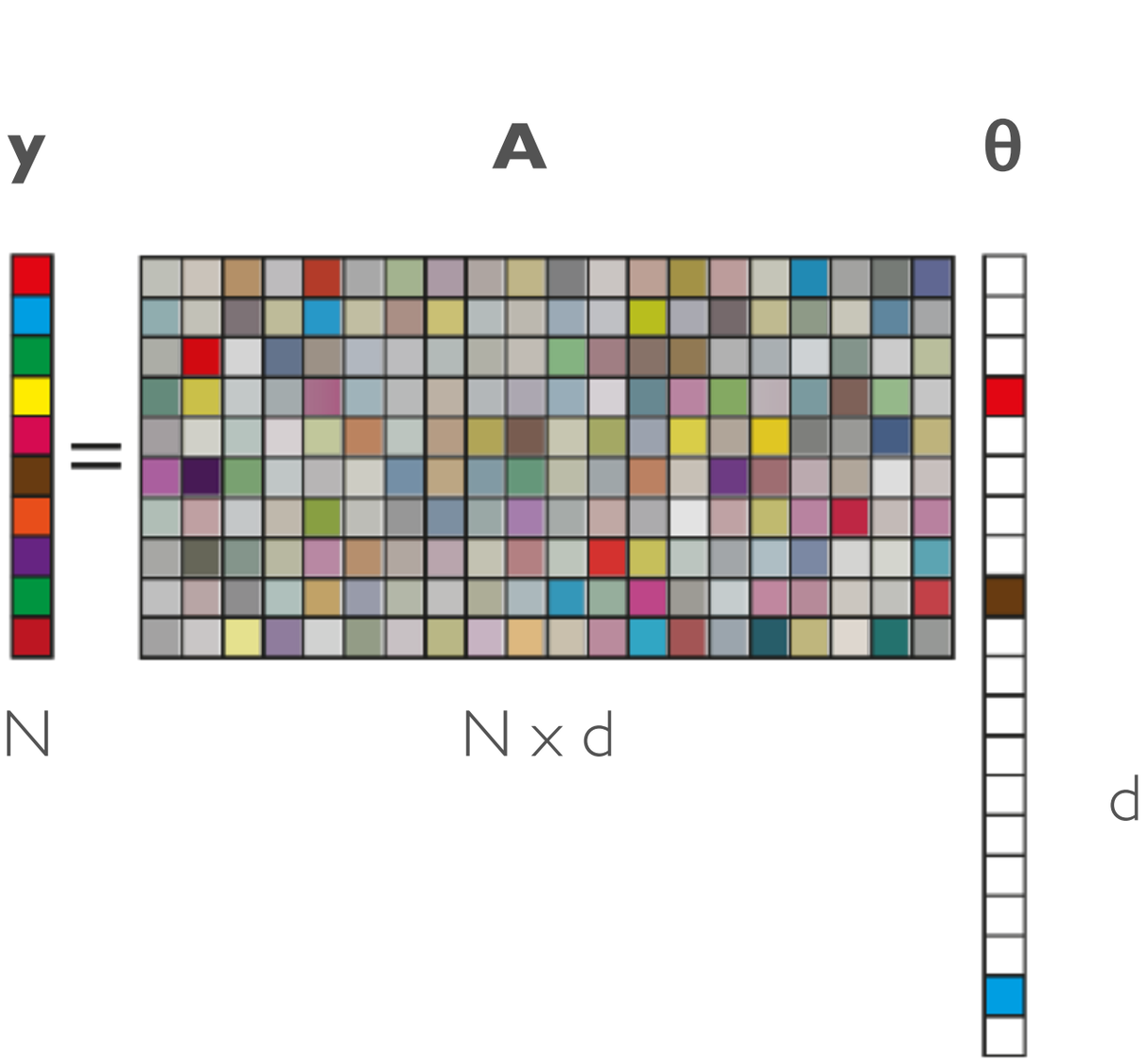
Risk :

$$\mathcal{R} = \frac{1}{n}\sum_i (y_i - \mathbf{X}_i \cdot \theta)^2 = \frac{1}{n}||\mathbf{Y} - A\theta||_2^2$$

Mean square Error (MSE)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

$$A = \begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \dots \\ \mathbf{X}_n^T \end{bmatrix}$$



Empirical Risk minimisation leads to:

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \mathrm{A}\theta||_2^2)$$

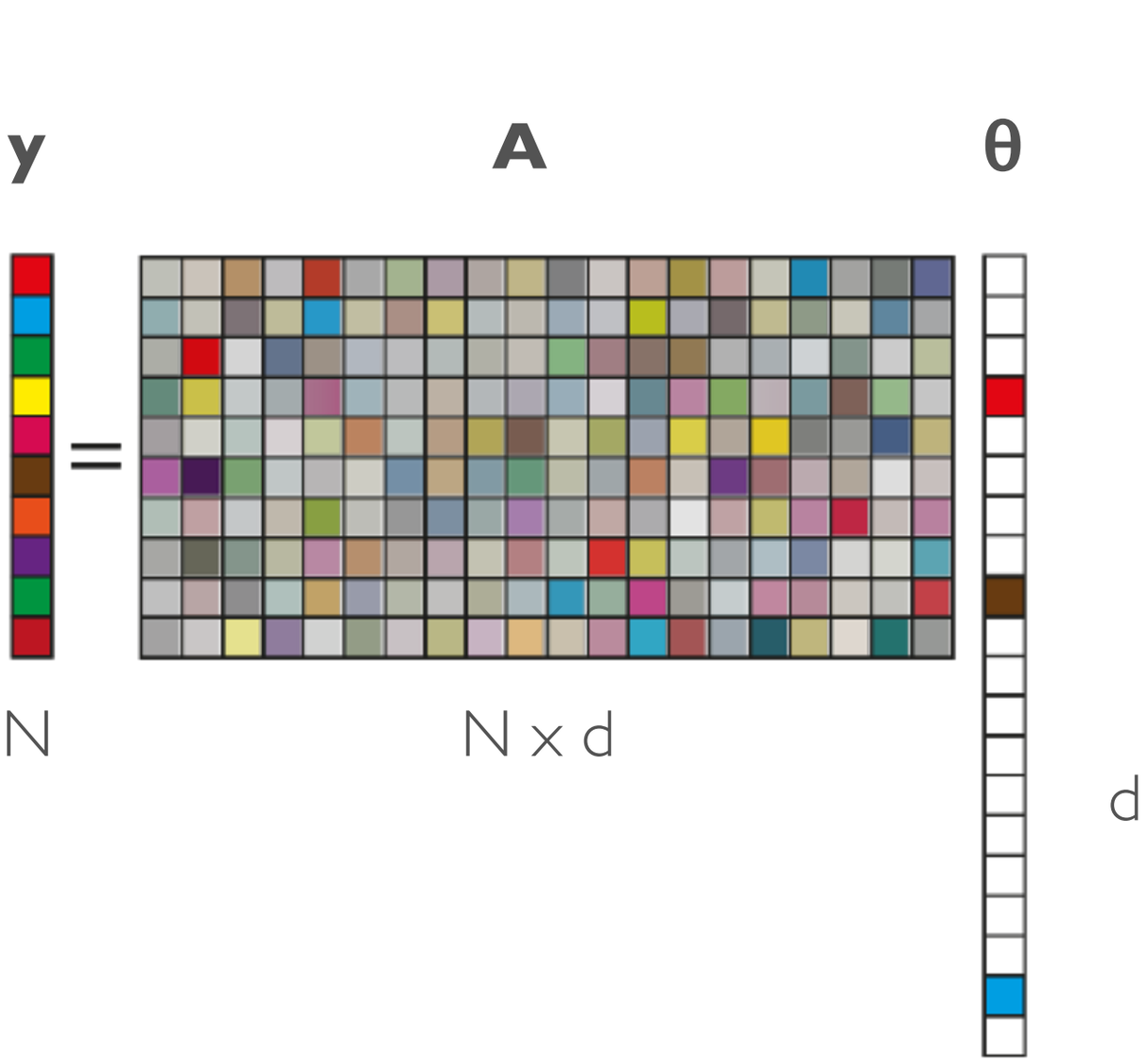
Risk :

$$\mathcal{R} = \frac{1}{n}\sum_i (y_i - \mathbf{X}_i \cdot \theta)^2 = \frac{1}{n}||\mathbf{Y} - A\theta||_2^2$$

Mean square Error (MSE)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

$$A = \begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \dots \\ \mathbf{X}_n^T \end{bmatrix}$$



Ordinary Least Square

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \text{A}\theta||_2^2)$$

$$||\mathbf{Y} - A\theta||_2^2 = (\mathbf{Y} - A\theta)^T(\mathbf{Y} - A\theta) = \mathbf{Y}^T\mathbf{Y} + \theta^T A^T A\theta - 2\mathbf{Y}^T A\theta$$

Taking the extremum yields the normal equations:

d x d d x n

$$A^T A\theta = A^T\mathbf{Y}$$

d x 1 n x 1

Unique solution if A^TA is full rank
This requires (*at least*) n>p
(no more unknown than datapoints)

$$\hat{\theta} = (A^T A)^{-1} A^T\mathbf{Y}$$

Otherwise, many solution may exists.
A popular choice leads
to the least-norm (*in l2 norm*) solution:

$$\hat{\theta}_{\text{ln}} = A^T(AA^T)^{-1}\mathbf{Y}$$

Note that (A^TA) is dxd while (AA^T) is nxn
In both cases we are required to inverse the « smaller » matrix

The obvious problems with linear models

The problems with linear models

(i) We find θ such that all label y^i=+/-1 are close enough to $X^i \cdot \theta$

(ii) If we apply our predictions to new points, we find $X^{new} \cdot \theta$: This is a <u>real</u> variable

(iii) So we use instead y^{new}=sign($X^{new} \cdot \theta$)

In the d-dimensional space, the frontier between the two classes is defined by

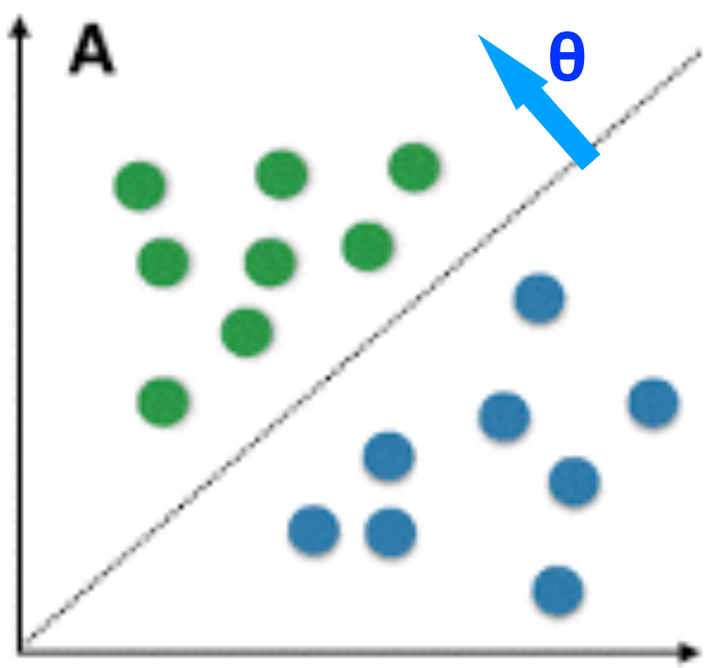
$$X \cdot \theta = 0$$

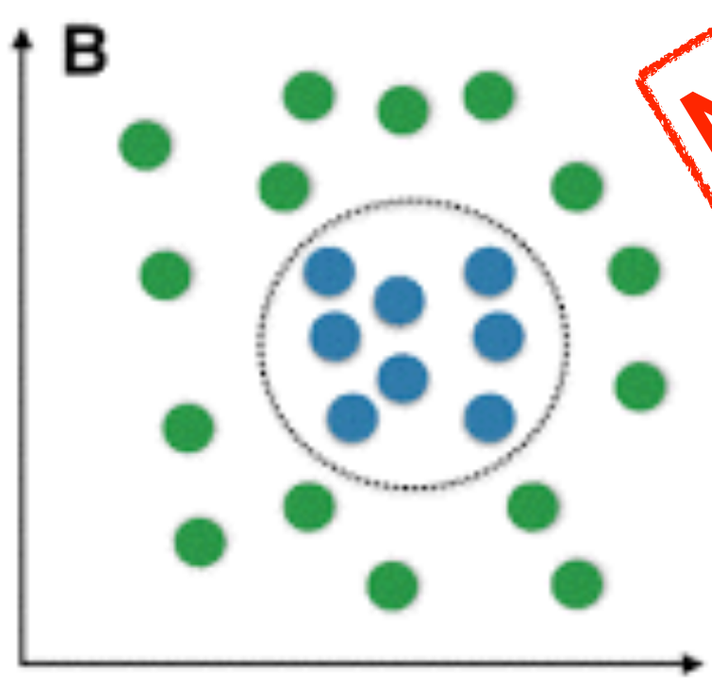
This described an hyperplane in d-dimension

(with the vector θ perpendicular to the hyperplane)

OK

Data Linearly separable

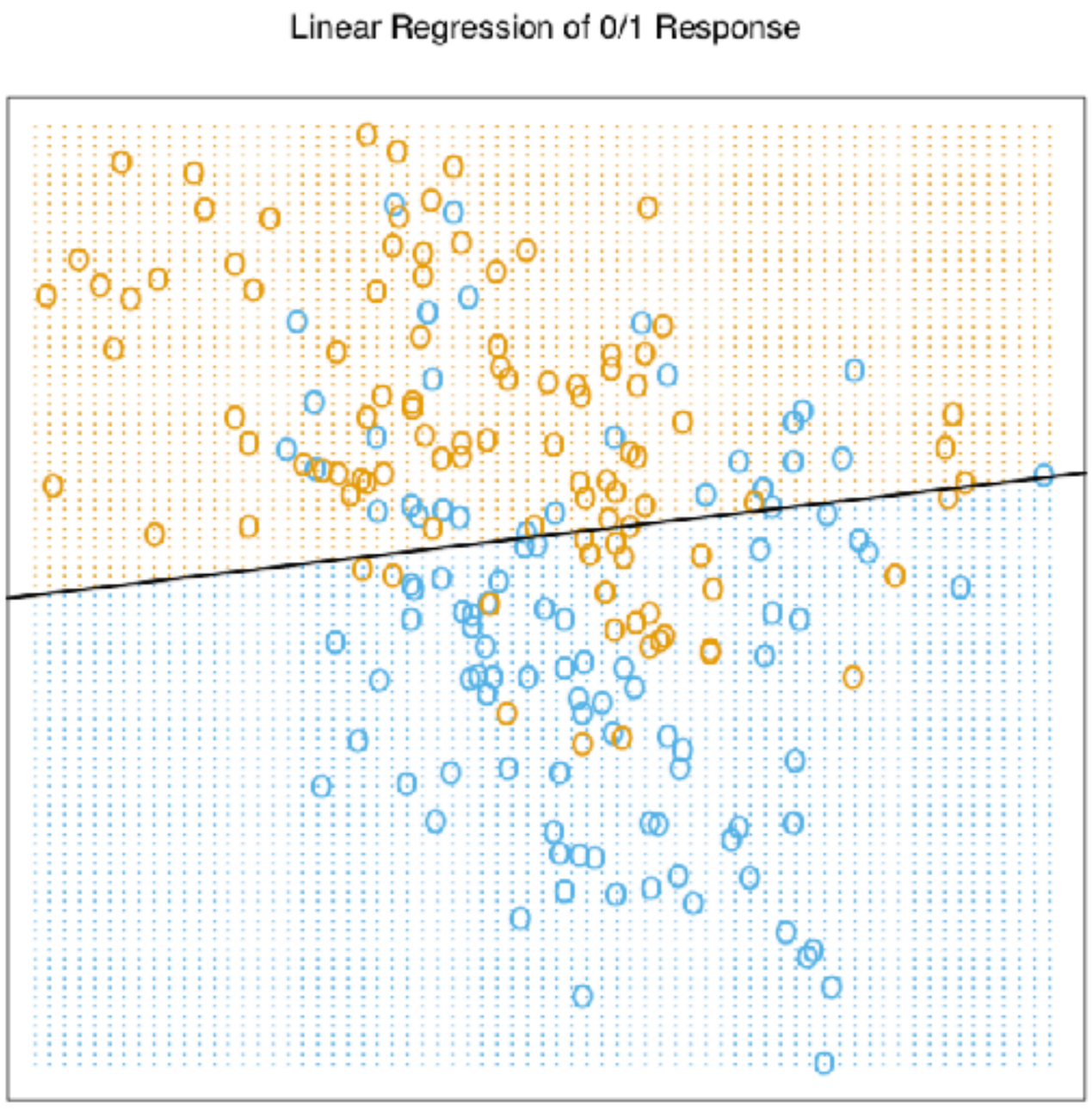




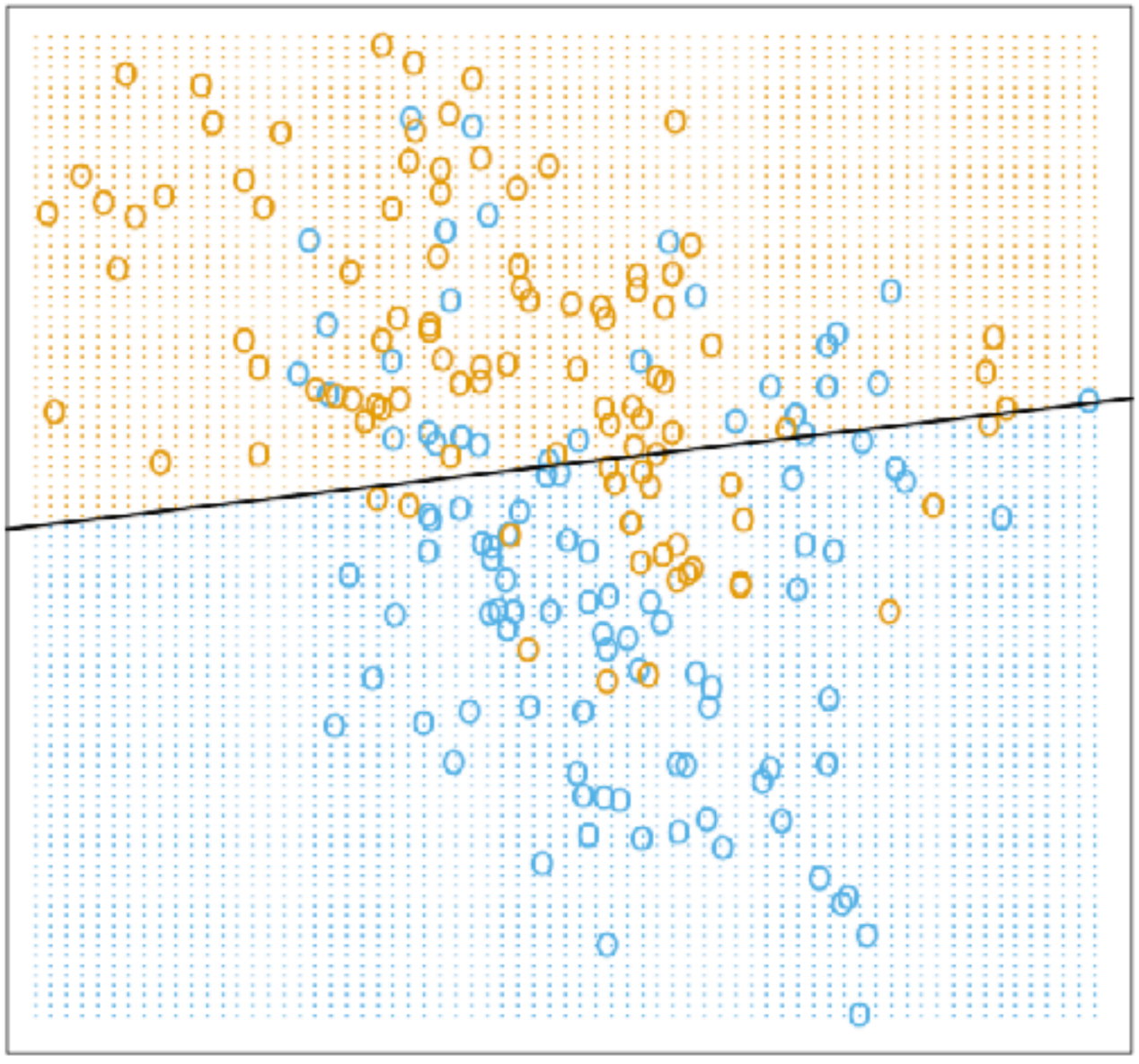
NOT OK

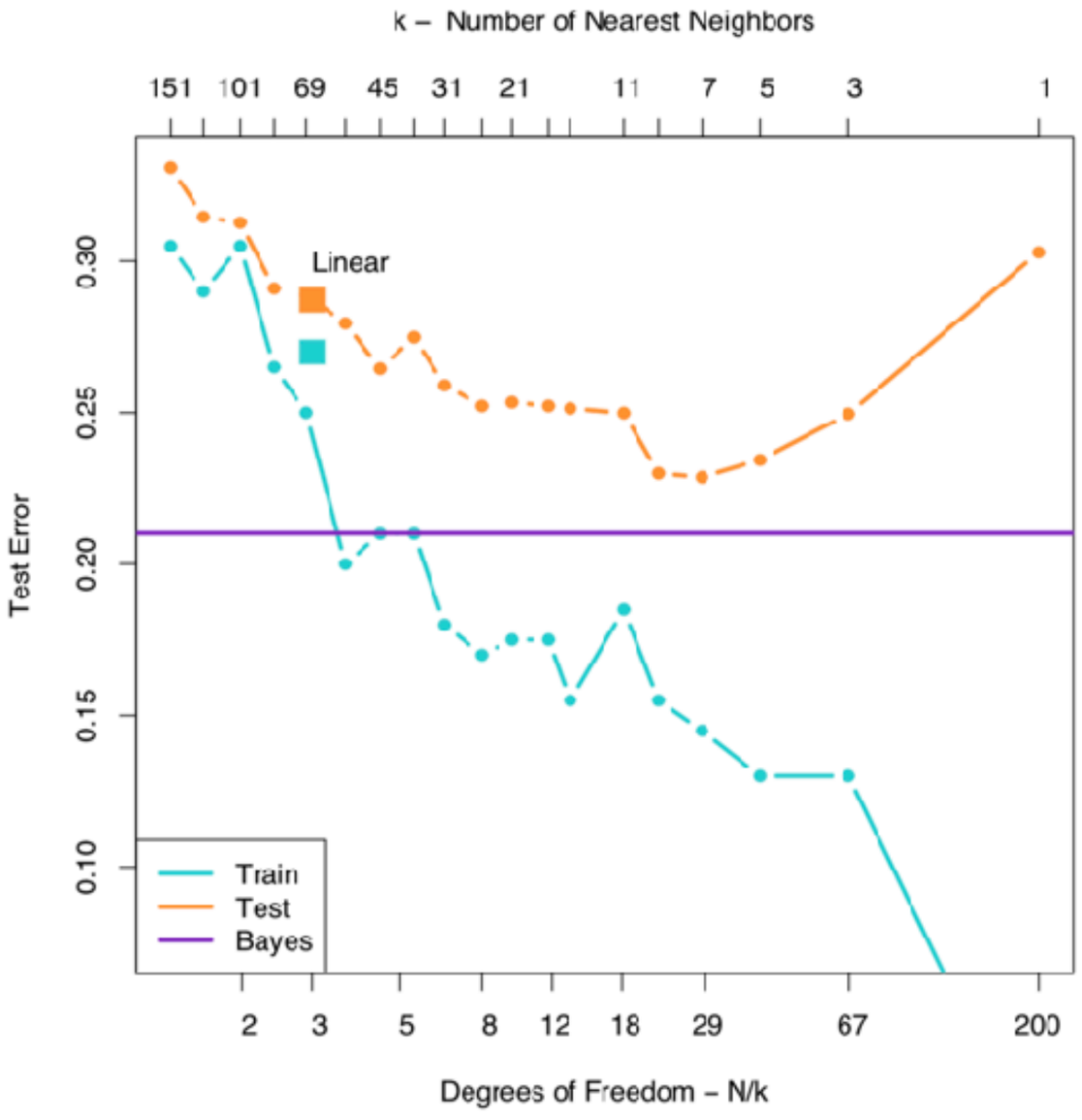
Data Not Linearly Separable

Can’t separate data very well



Can’t separate data very well





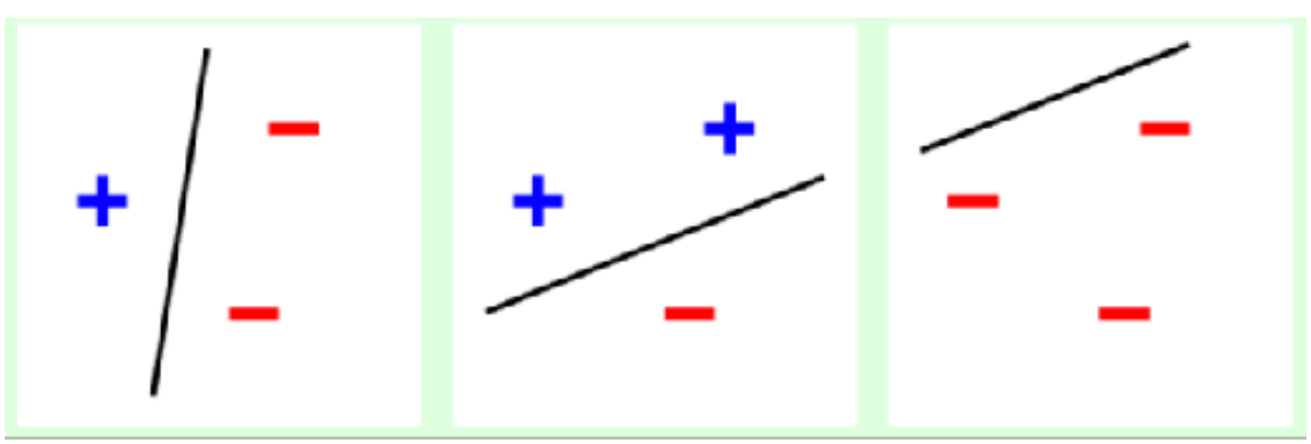
We will fix this problem next week. For now we will have to accept it

d_{VC} : VC dimension = capacity of your classifier/function class

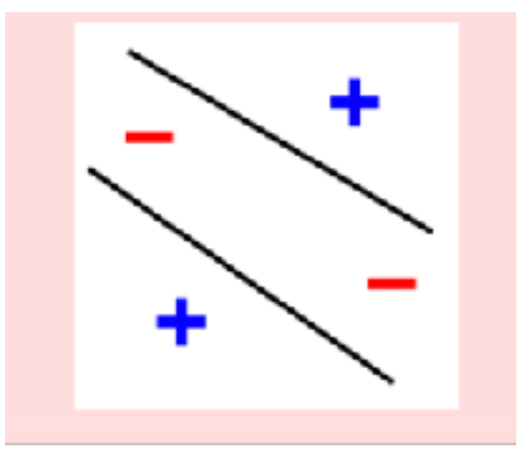
$N<d_{VC}$ means: <u>it exists a</u> set of points/patterns such that <u>all</u> assignments can be fitted

$N>d_{VC}$ means: <u>for all</u> set of points/patterns <u>some</u> assignments cannot be fitted

Ex D=2, linear fit:



N=3 -> it exists a set of points that can always be classified



N=4 -> No set of points can always be classified

VC dimension (linear classification) = D+1


Why speaking of a Linear classifier?
Shouldn’t we discuss Deep Learning?
47

With linear models,
we can builds powerful kernel models

And if we pile them up,
we can also build neural nets:

So let us spend again some time on linear models

Today

i) Knn
ii) a bit of theory
iii) Linear models
iv) Regularization
v) Loss functions
vi) Final remarks

The linear model revisited: regularisation

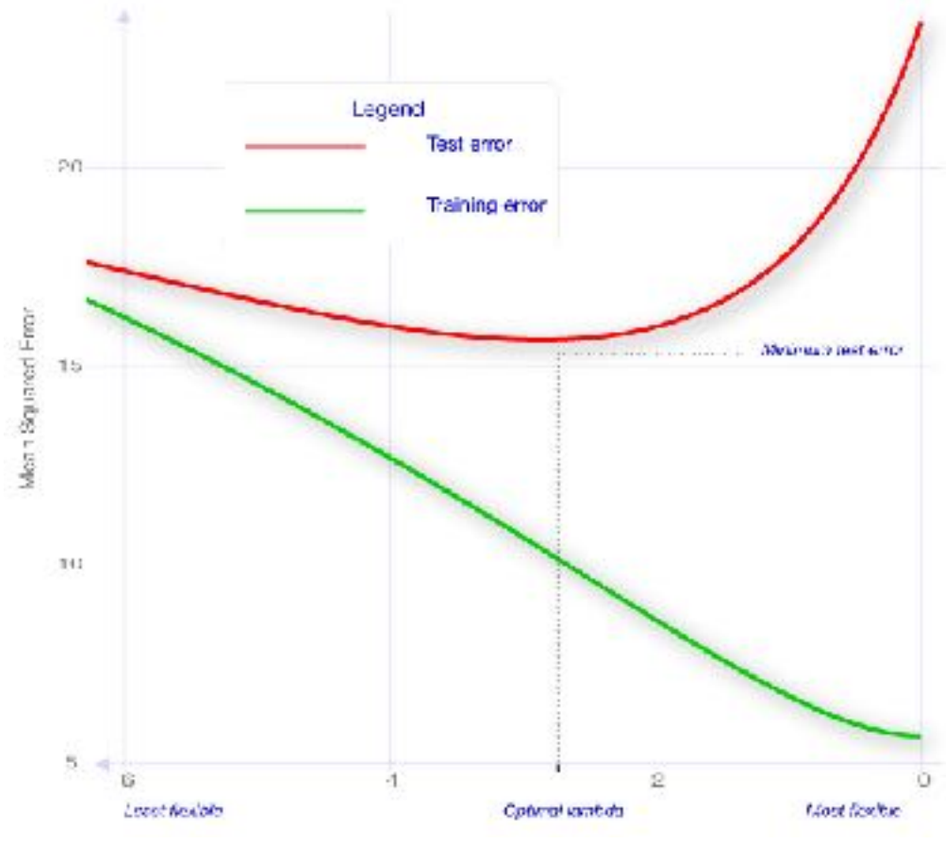
Replace $\hat{\theta} = \text{argmin}(||\mathbf{Y} - \text{A}\theta||_2^2)$

By $\hat{\theta} = \text{argmin}(||\mathbf{Y} - \text{A}\theta||_2^2) + \text{g}(\theta)$

L2-regularization* aka *Tikhonov regularization
aka *Ridge regression* aka *Weight decay*

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \text{A}\theta||_2^2) + \Gamma||\theta||_2^2$$

Find the best value for Γ using cross-validation



Analytic solution

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \text{A}\theta||_2^2)$$

$$||\mathbf{Y} - A\theta||_2^2 = (\mathbf{Y} - A\theta)^T(\mathbf{Y} - A\theta) = \mathbf{Y}^T\mathbf{Y} + \theta^T A^T A\theta - 2\mathbf{Y}^T A\theta$$

Analytic solution

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \mathrm{A}\theta||_2^2) + \Gamma||\theta||_2^2$$

$$||\mathbf{Y} - A\theta||_2^2 = (\mathbf{Y} - A\theta)^T(\mathbf{Y} - A\theta) = \mathbf{Y}^T\mathbf{Y} + \theta^T A^T A\theta - 2\mathbf{Y}^T A\theta + \Gamma\theta^T\theta$$

Taking the extremum yields the normal equations:

d x d d x l d x n n x l

$$(A^T A + \Gamma\mathbf{1})\theta = A^T\mathbf{Y}$$

All eigenvalues are positive
and the problem has been regularized

$$\hat{\theta} = (A^T A + \mathbf{1})^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{Y}$$

Analytic solution

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \mathrm{A}\theta||_2^2) + \Gamma||\theta||_2^2$$

$$||\mathbf{Y} - A\theta||_2^2 = (\mathbf{Y} - A\theta)^T(\mathbf{Y} - A\theta) = \mathbf{Y}^T\mathbf{Y} + \theta^T A^T A\theta - 2\mathbf{Y}^T A\theta + \Gamma\theta^T\theta$$

Taking the extremum yields the normal equations:

d x d d x l d x n n x l

$$(A^T A + \Gamma\mathbf{1})\theta = A^T\mathbf{Y}$$

All eigenvalues are positive: now AA^T always exists and the problem has been regularized

$$\hat{\theta} = (A^T A + \mathbf{1})^{-1} A^T\mathbf{Y}$$

$$\hat{\theta} = A^T(AA^T + \Gamma\mathbf{1})^{-1}\mathbf{Y}$$

Note that (A^TA) is dxd while (AA^T) is nxn
We should inverse the « smaller » matrix

Other regularizations

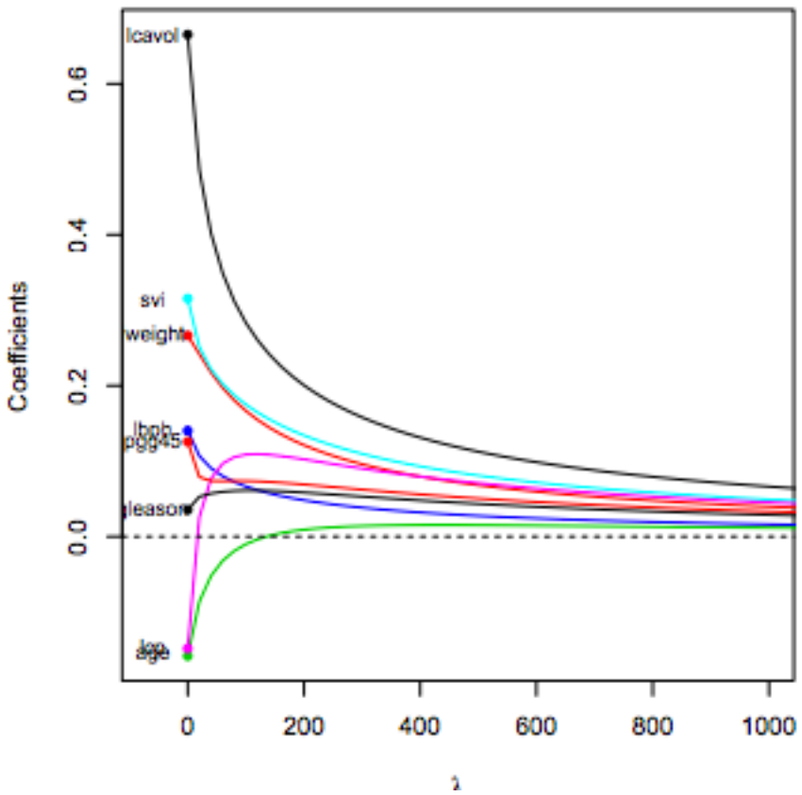
LASSO

least absolute shrinkage and selection operator

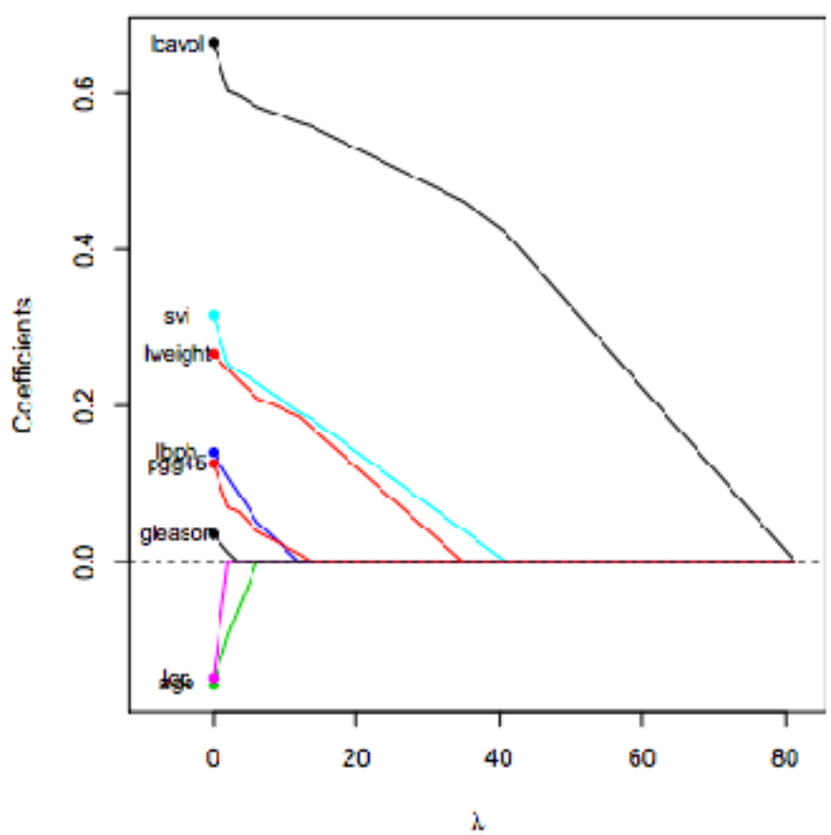
$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \mathrm{A}\theta||_2^2) + \Gamma||\theta||_1$$

$$||\theta||_1 = \sum_i |\theta_i|$$

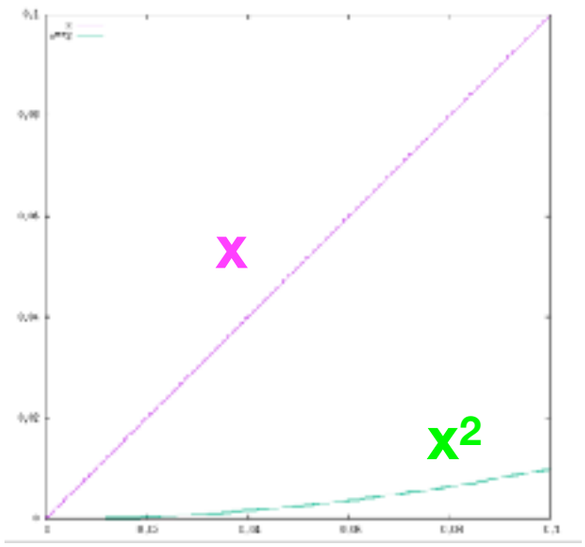
Ridge



Lasso



l2 vs l1



Due to the linear cost,
l1 norm promotes sparsity

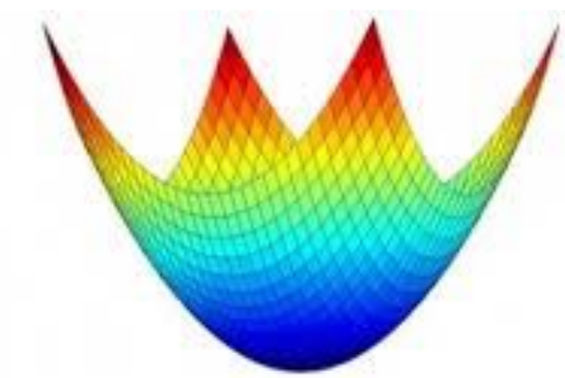
No closed-form solution, but the cost function is convex: fast gradient-descent algorithms

Coordinate gradient descents

Iterative soft-thresholding (ISTA, FISTA)

Approximate Message Passing (AMP, VAMP, EC)

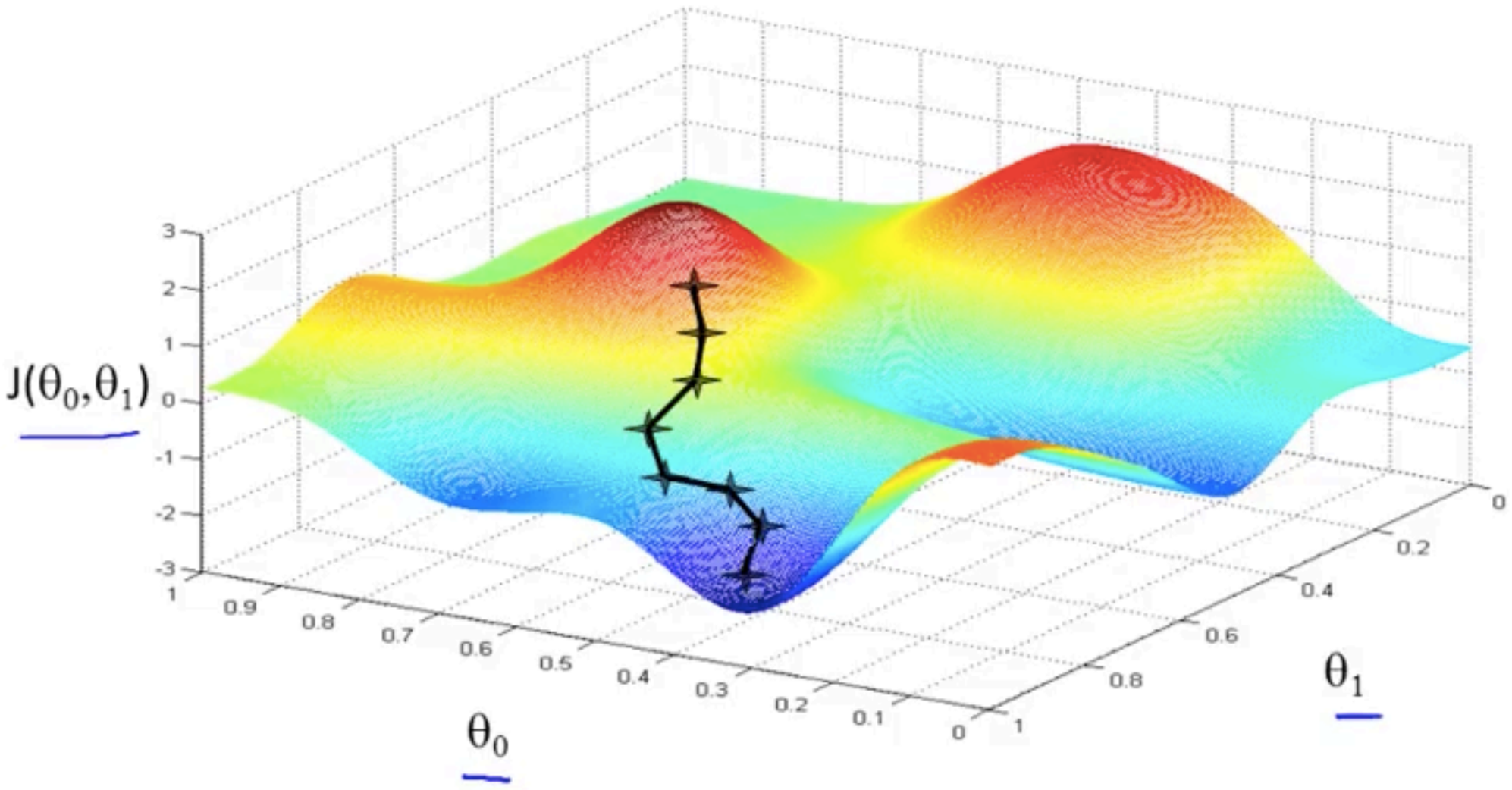
....



Minimising the cost function by gradient descent

$$\vec{\theta}^{t+1} = \vec{\theta}^{t} - \gamma \nabla R(\vec{\theta}^{t})$$

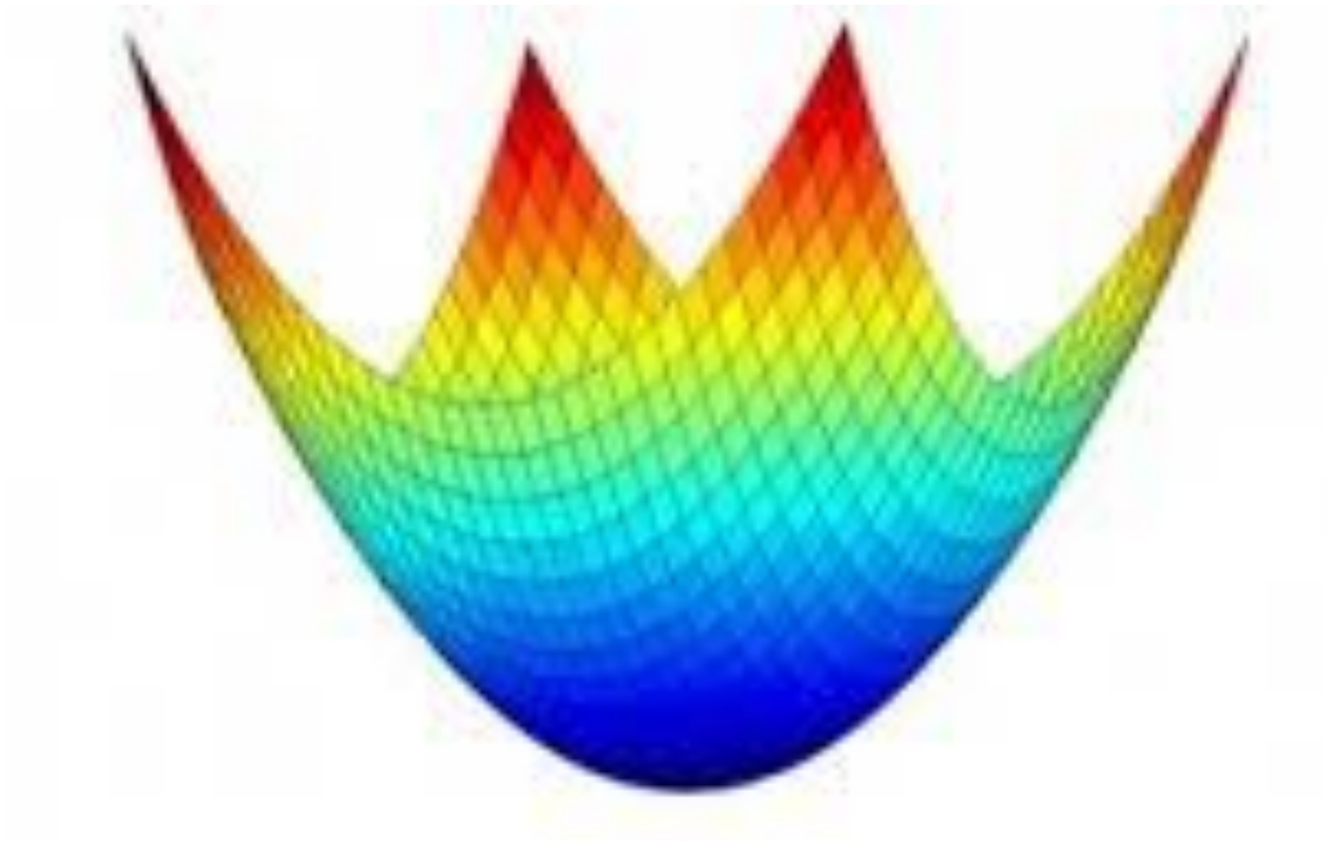
If γ small enough, converges to a (possible local) minima



Minimising the cost function by gradient descent

$$\vec{\theta}^{t+1} = \vec{\theta}^{t} - \gamma \nabla R(\vec{\theta}^{t})$$

In our case, the landscape is convex, so the problem remain easy



We shall come back on gradient descent methods later in our lectures

Elastic-net

Combine Rigde and LASSO

$$\hat{\theta} = \text{argmin}(||\mathbf{Y} - \text{A}\theta||_2^2) + \Gamma_1||\theta||_1 + \Gamma_2||\theta||_2^2$$

$$||\theta||_1 = \sum_i |\theta_i|$$

$$||\theta||_2^2 = \sum_i \theta_i^2$$

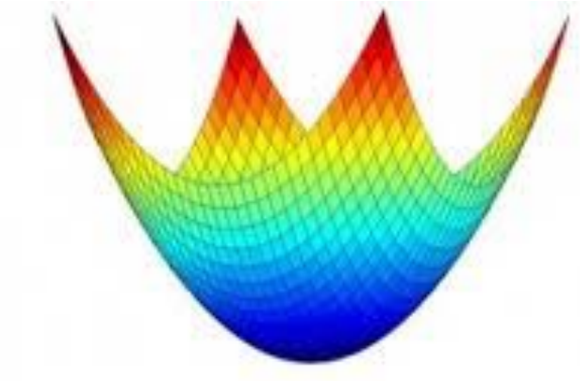
No closed-form solution, but the cost function is convex: fast gradient-descent algorithms

Coordinate gradient descents

Iterative soft-thresholding (ISTA, FISTA)

Approximate Message Passing (AMP, VAMP, EC)

….



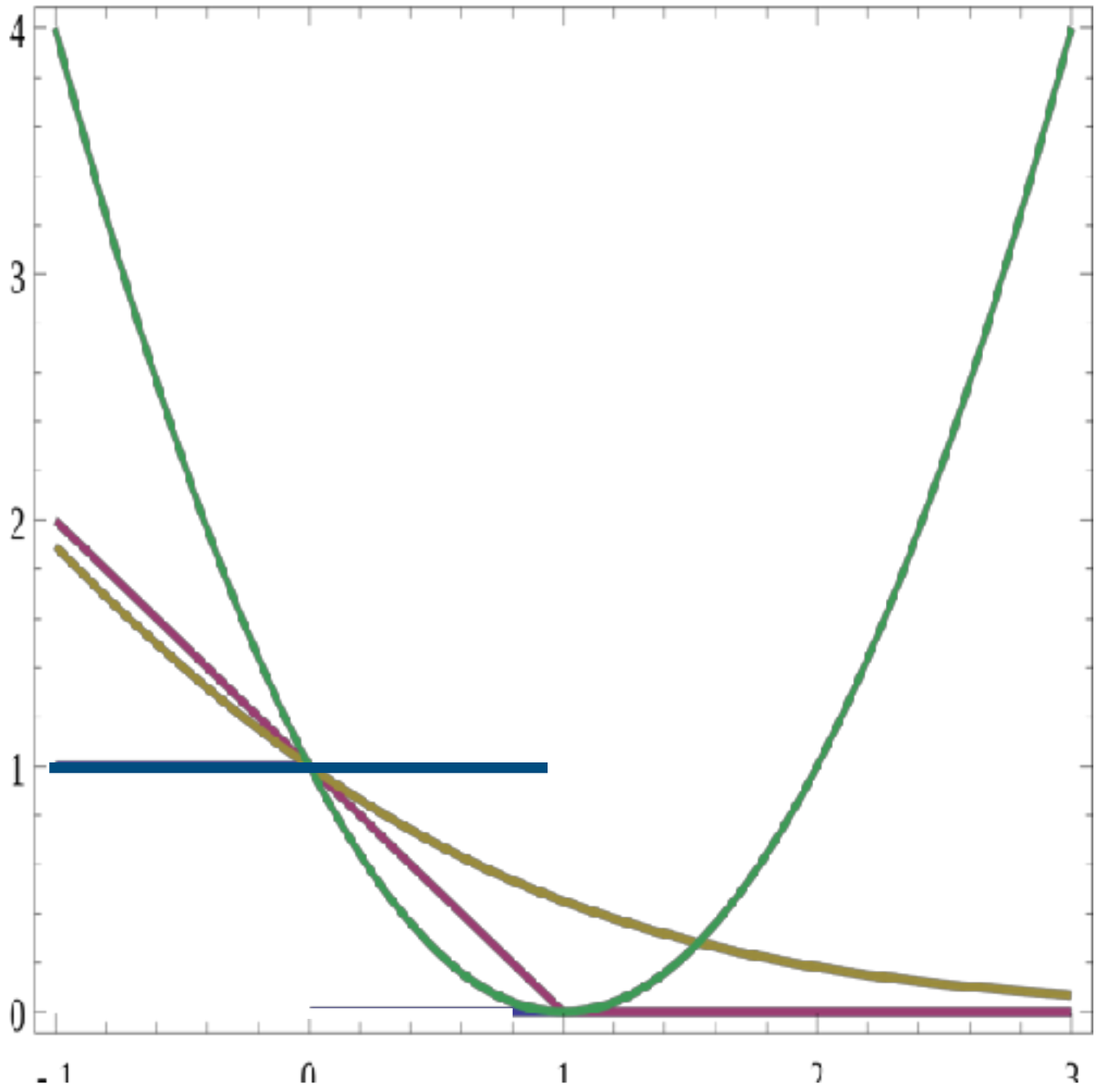
Today

i) Knn
ii) a bit of theory
iii) Linear models
iv) Regularization
v) Loss functions
vi) Final remarks

Why the focus on quadratic risk ?

Popular Loss functions

$$f(\vec{x}) = \theta \cdot \vec{x} + \alpha$$



Square Loss

$$L(f(\vec{x}), y) = (1 - yf(\vec{x}))^2$$

Hard margin

$$L(f(\vec{x}), y) = \mathbf{1}(yf(\vec{x}) > 1)$$

Hinge loss

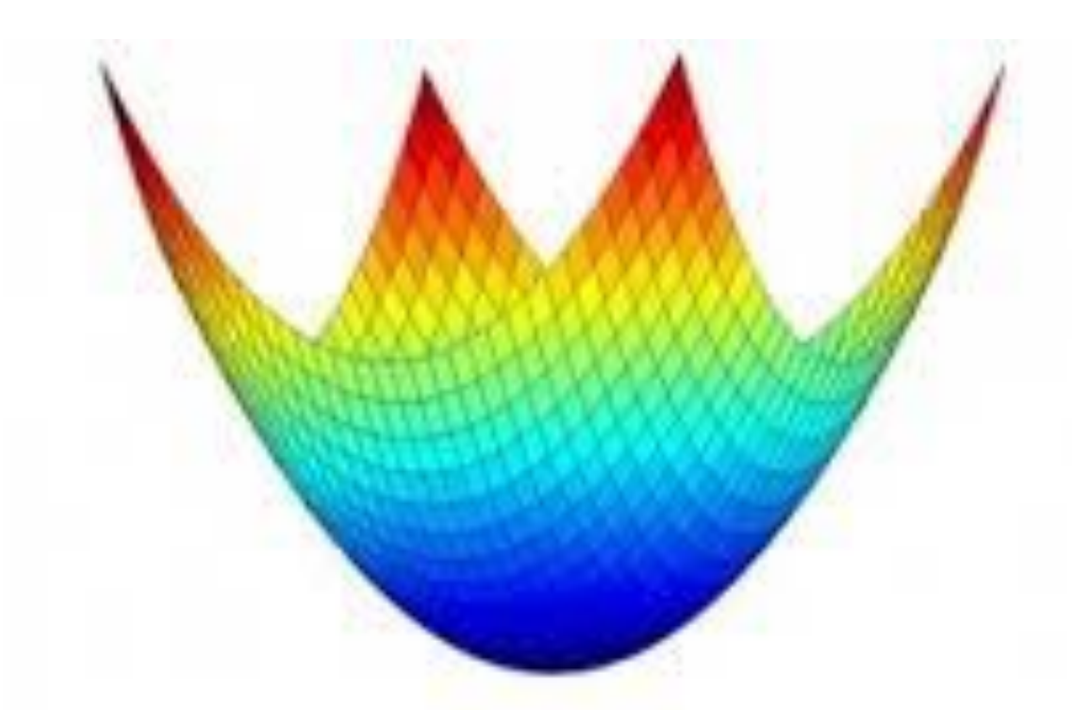
$$L(f(\vec{x}), y) = \max(0, 1 - yf(\vec{x}))$$

Logistic loss/Cross-entropy

$$L(f(\vec{x}), y) = \frac{1}{\ln 2} \ln(1 + e^{-yf(\vec{x})})$$

Optimization by gradient descent

All these function are differentiable
(or have sub-derivatives)
And leads to convex problems:
These can be optimised by Gradient descent



$$\vec{\theta}^{t+1} = \vec{\theta}^{t} - \gamma \nabla R(\vec{\theta}^{t})$$

$$\theta \in \mathbb{R}^d$$

$$f(\vec{x}) = \theta \cdot \vec{x} + \alpha$$

Square Loss

$$L(f(\vec{x}), y) = (1 - y f(\vec{x}))^2$$

~~**Hard margin**~~

~~$$L(f(\vec{x}), y) = \mathbf{1}(y f(\vec{x}) > 1)$$~~

Hinge loss

$$L(f(\vec{x}), y) = \max(0, 1 - y f(\vec{x}))$$

Logistic loss/Cross-entropy

$$L(f(\vec{x}), y) = \frac{1}{\ln 2} \ln(1 + e^{-y f(\vec{x})})$$

big n or big d ?

For linear system, we had the choice:

$$\hat{\theta} = (A^T A + \mathbf{1})^{-1} A^T \mathbf{Y}$$

$$\hat{\theta} = A^T (A A^T + \Gamma \mathbf{1})^{-1} \mathbf{Y}$$

If n is smaller than d then use the second formula

$$X_{\text{new}}^T \cdot \hat{\theta} = X_{\text{new}}^T \cdot A^T (A A^T + \Gamma \mathbf{1})^{-1} \mathbf{Y}$$

$$X_{\text{new}}^T \cdot \hat{\theta} = \sum_{i=1}^{n} (X_{\text{new}}^T X_i) \alpha_i$$

$$\hat{\theta} = \sum_{i=1}^{n} \alpha_i X_i$$

Can we have a similar formula for our generalized loss?

Representer theorem

For any loss function such that

$$\mathcal{R} = \frac{1}{n}\sum_i \mathcal{L}(y_i \theta \cdot \mathbf{X}_i) \qquad \hat{\theta} = \text{argmin}\mathcal{R}(\theta)$$

We can always write the minimiser as: $\hat{\theta} = \sum_{i=1}^{n} \alpha_i X_i$

Proof $\hat{\theta} \in \mathcal{R}^d \quad \mathbf{X}_i \in \mathcal{R}^d \; \forall i \quad \mathbb{R}^d = \text{span}(\{X\}) + \text{null}(\{X\})$

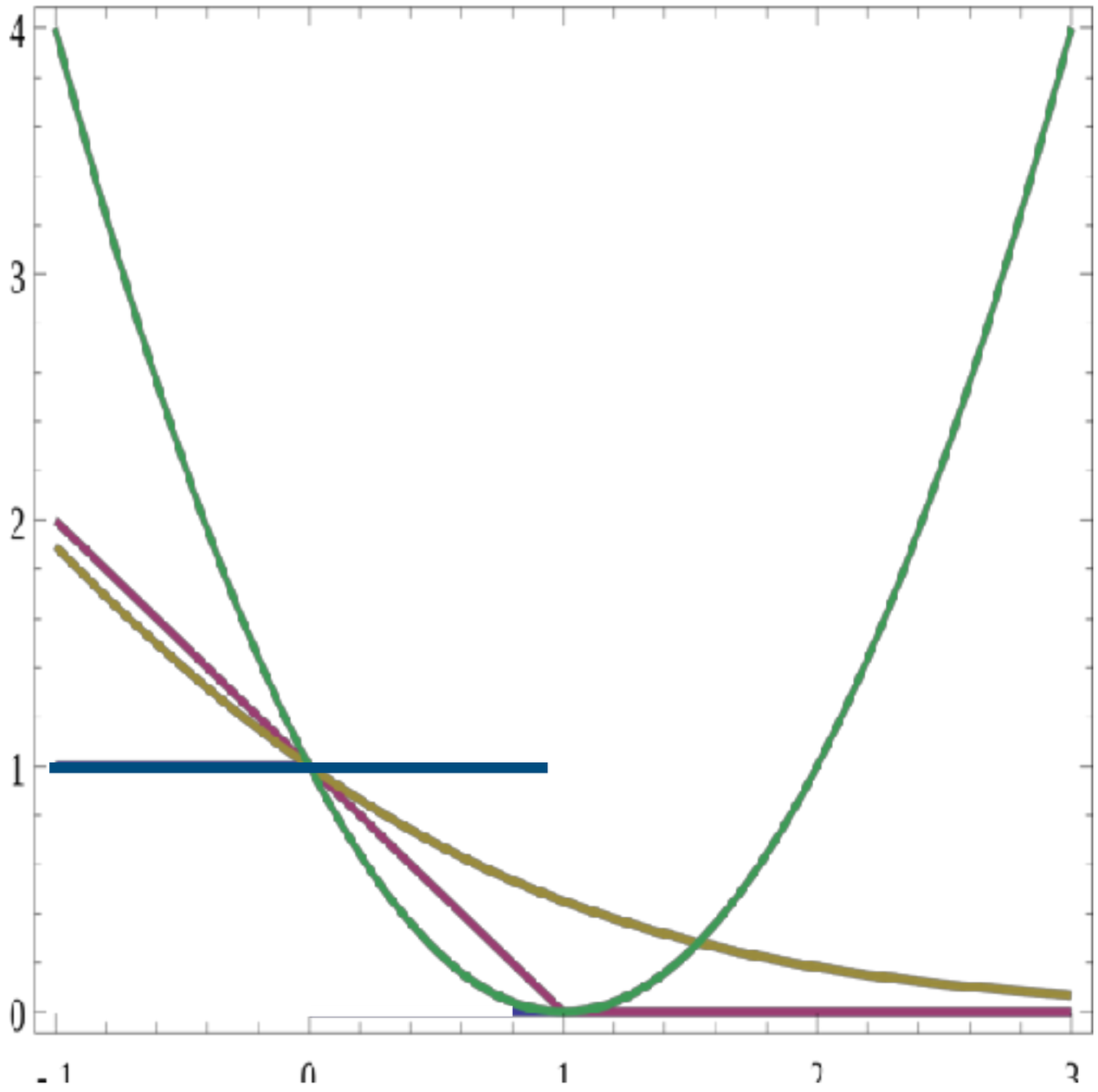
So we can write: $\hat{\theta} = \sum_{i=1}^{n} \alpha_i \mathbf{X}_i + \vec{n}$

But: $\hat{\theta} \cdot \mathbf{X_i} = \left(\sum_{i=1}^{n} \alpha_i \mathbf{X}_i\right) \cdot \mathbf{X_i} + \vec{n} \cdot \mathbf{X_i} = \left(\sum_{I=1}^{n} \alpha_i \mathbf{X}_i\right) \cdot \mathbf{X_i}$

So we might as well write: $\hat{\theta} = \sum_{i=1}^{n} \alpha_i X_i$ ■

Pushing the boundaries

$$f(\vec{x}) = \theta \cdot \vec{x} + \alpha$$



Square Loss

$$L(f(\vec{x}), y) = (1 - yf(\vec{x}))^2$$

Hard margin

$$L(f(\vec{x}), y) = \mathbf{1}(yf(\vec{x}) > 1)$$

Hinge loss

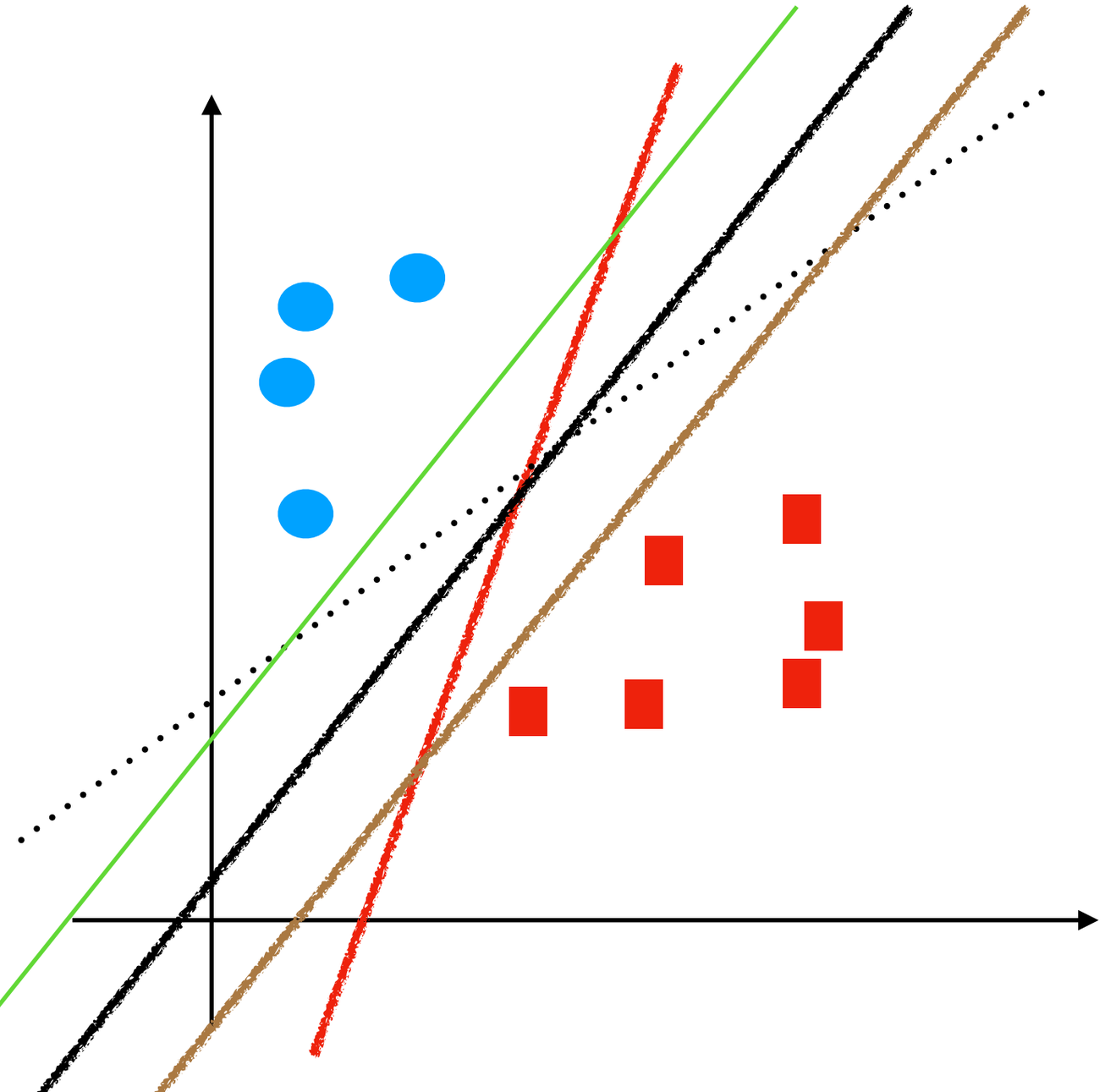
$$L(f(\vec{x}), y) = \max(0, 1 - yf(\vec{x}))$$

Logistic loss/Cross-entropy

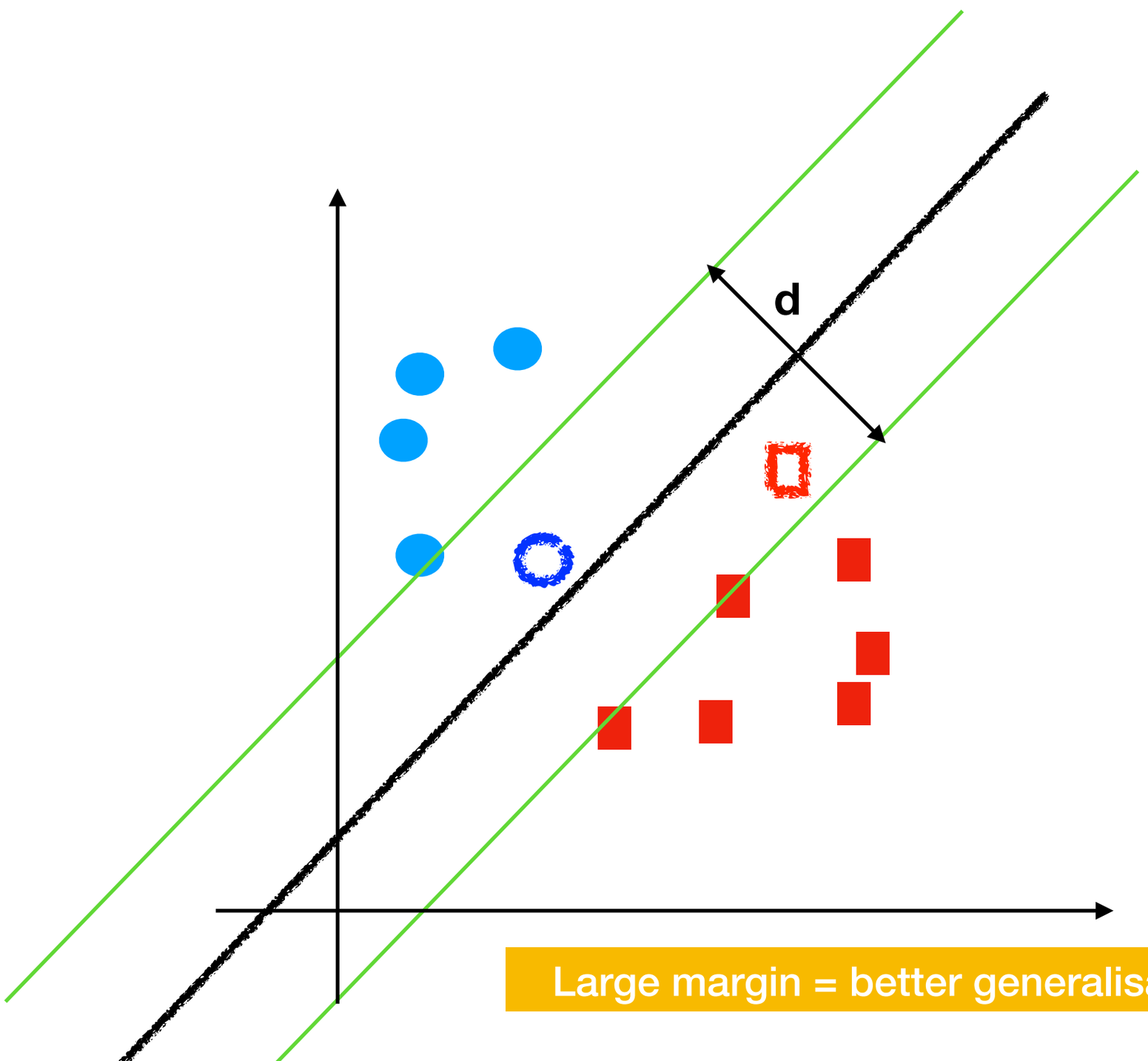
$$L(f(\vec{x}), y) = \frac{1}{\ln 2} \ln(1 + e^{-yf(\vec{x})})$$

Pushing the boundaries

Which frontier should we choose?

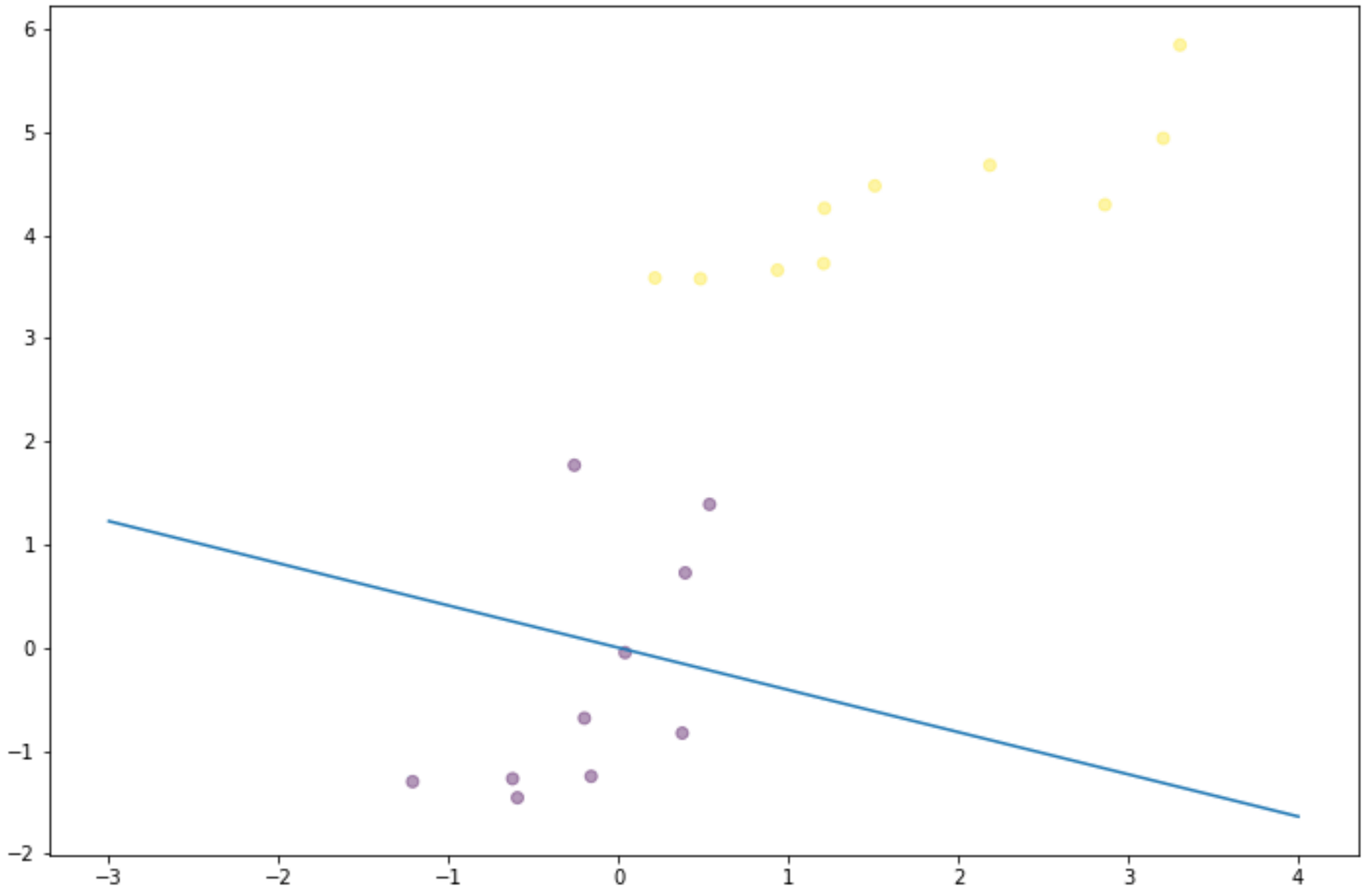


Pushing the boundaries

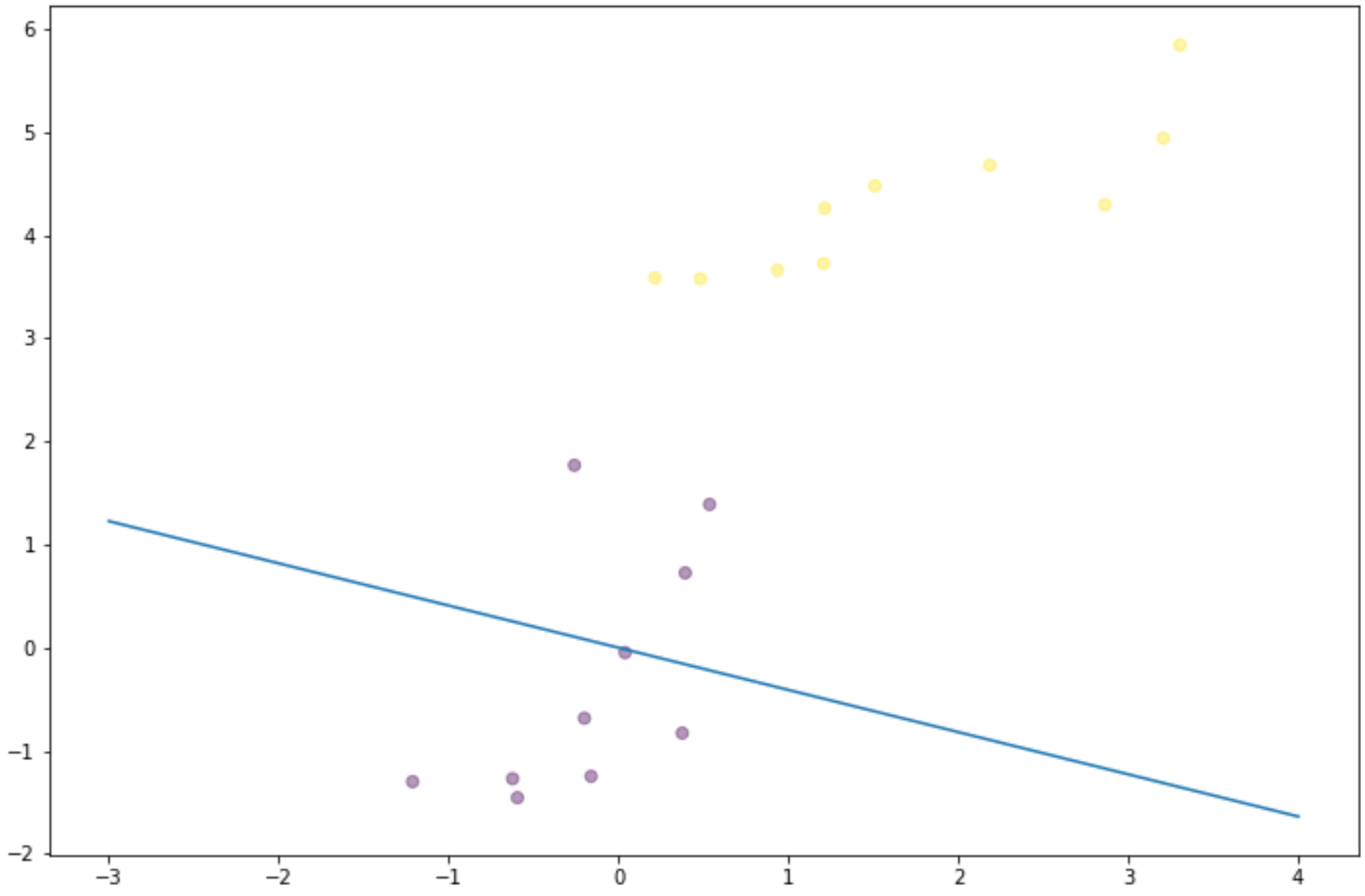


Large margin = better generalisation properties!

Ex: l2 LOSS



Ex: logistic LOSS



Support Vector Machines

L2 regularization and max-margin classification

$$f(\vec{x}) = \theta \cdot \vec{x} + \alpha$$

Hinge Loss

$$L(f(\vec{x}), y) = \max(0, 1 - yf(\vec{x}))$$

$$\theta \cdot \vec{x} + \alpha = 1$$

$$\theta \cdot \vec{x} + \alpha = 0$$

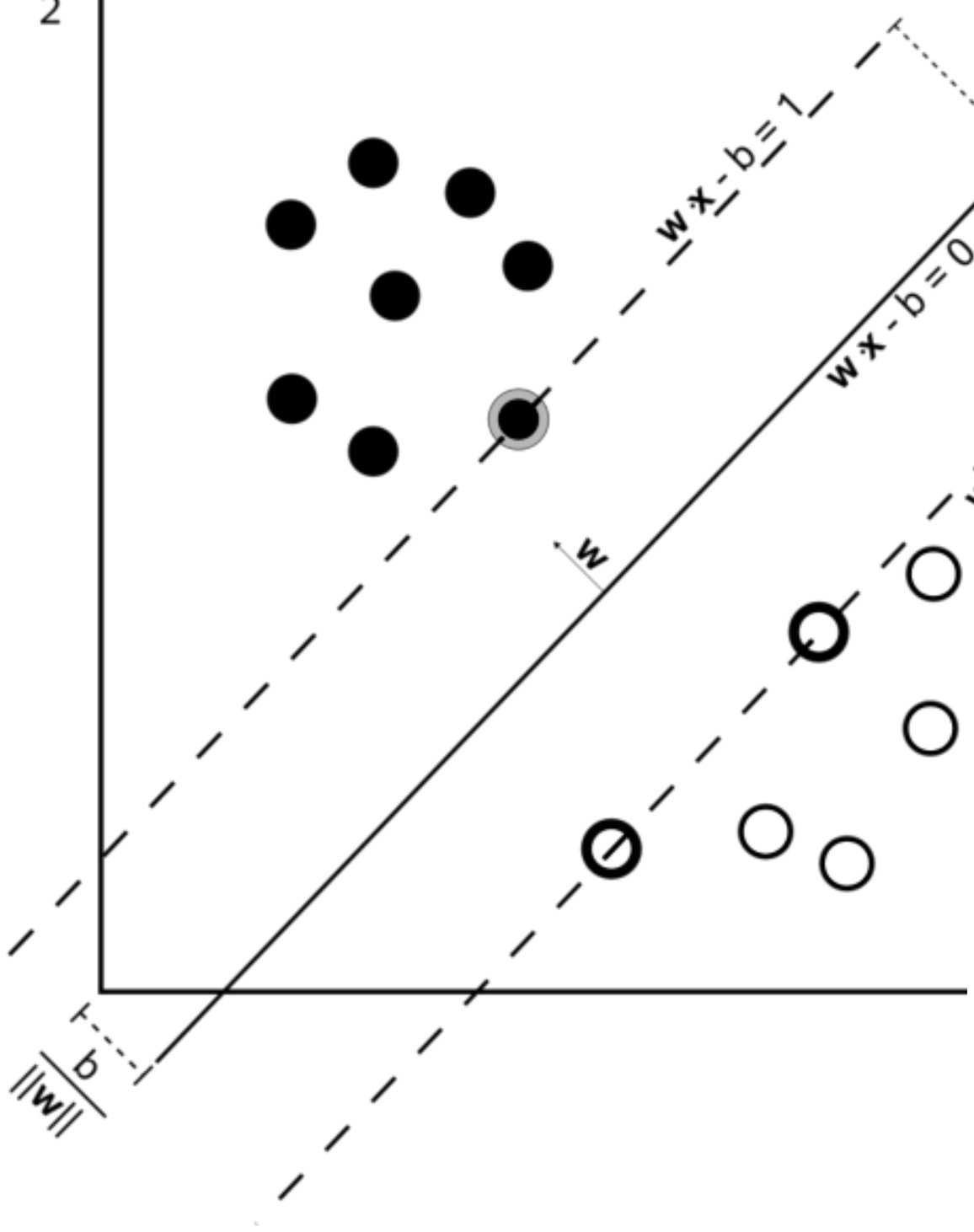
$$\theta \cdot \vec{x} + \alpha - 1$$

$$d = \frac{2}{||\theta||_2}$$

d

Support vectors

Large margin = better generalisation properties!



Let $\mathbf{x}_0$ be a point in the hyperplane $\mathbf{wx} - b = -1$, i.e., $\mathbf{wx}_0 - b = -1$. To measure the distance between hyperplanes $\mathbf{wx} - b = -1$ and $\mathbf{wx} - b = 1$, we only need to compute the perpendicular distance from $\mathbf{x}_0$ to plane $\mathbf{wx} - b = 1$, denoted as r.

Note that $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ is a unit normal vector of the hyperplane $\mathbf{wx} - b = 1$. We have

$$\mathbf{w}(\mathbf{x}_0 + r\frac{\mathbf{w}}{\|\mathbf{w}\|}) - b = 1$$

since $\mathbf{x}_0 + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$ should be a point in hyperplane $\mathbf{wx} - b = 1$ according to our definition of r.

Expanding this equation, we have

$$\begin{aligned}
&\mathbf{wx}_0 + r\frac{\mathbf{ww}}{\|\mathbf{w}\|} - b = 1 \\
\Longrightarrow\ &\mathbf{wx}_0 + r\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} - b = 1 \\
\Longrightarrow\ &\mathbf{wx}_0 + r\|\mathbf{w}\| - b = 1 \\
\Longrightarrow\ &\mathbf{wx}_0 - b = 1 - r\|\mathbf{w}\| \\
\Longrightarrow\ &-1 = 1 - r\|\mathbf{w}\| \\
\Longrightarrow\ &r = \frac{2}{\|\mathbf{w}\|}
\end{aligned}$$

Support Vector Machines

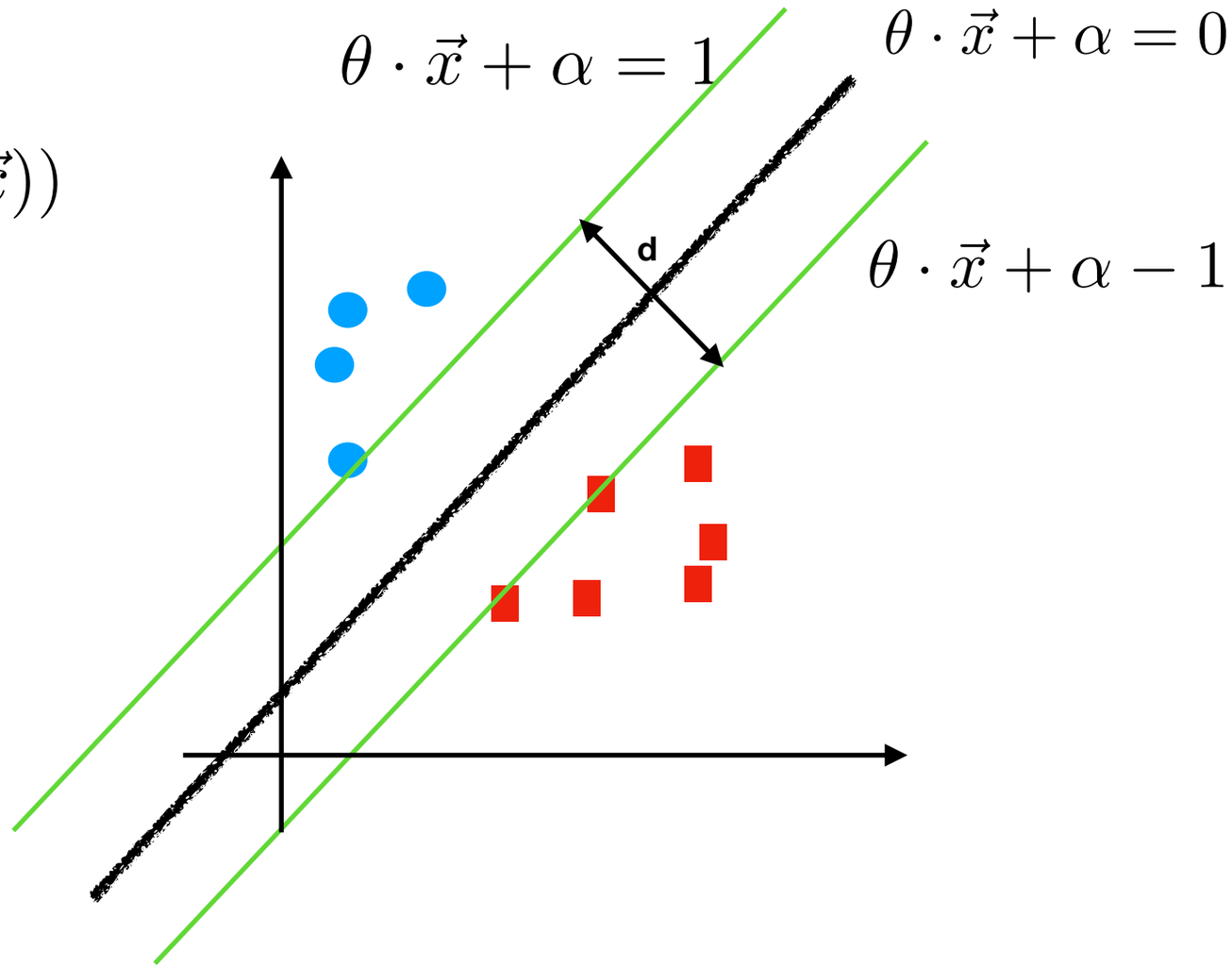
L2 regularization and max-margin classification

$$f(\vec{x}) = \theta \cdot \vec{x} + \alpha$$

Hinge Loss

$$L(f(\vec{x}), y) = \max(0, 1 - yf(\vec{x}))$$

$$d = \frac{2}{||\theta||_2}$$



Large margin = better generalisation properties!

Small l2 norm for the parameters implies large margin:
l2 regularization improve the generalisation error!

Support Vector Machines

L2 regularization and max-margin classification

Theorem (Vapnik, 1982):

- Given N data points in $\mathbb{R}^D$: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $||\mathbf{x}_n|| \leq R$
- Define $\mathcal{H}_\gamma$: set of classifiers in $\mathbb{R}^D$ having margin γ on $\mathbf{X}$

The VC dimension of $\mathcal{H}_\gamma$ is bounded by:

$$VC(\mathcal{H}_\gamma) \leq \min\left\{D, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil\right\}$$

Today

i) Knn
ii) a bit of theory
iii) Linear models
iv) Regularization
v) Loss functions
vi) Final remark

Remark 1

Are these that different from knn ?

According to the representer theorem, we are actually predicting using

$$y^{\text{new}} = \text{sign}\left(\sum_{i=1}^{n}(X_{\text{new}}^{T}X_i)\alpha_i\right)$$

Consider a knn that average over a distance d:

$$y_{\text{variant knn}}^{\text{new}} = \text{sign}\left(\sum_{i=1}^{n}\mathbf{1}(||X_{\text{new}}^{T} - X_i||_2 < d) \times y_i\right)$$

Both are not very different: they works by comparing with <u>each</u> training sample

« Funes, the memorious » all over again!

Remark 2

Are these that different from knn ?

According to the representer theorem, we are actually predicting using

$$y^{\text{new}} = \text{sign}\left(\sum_{i=1}^{n}(X_{\text{new}}^{T}X_i)\alpha_i\right)$$

Consider a knn that average over a distance d:

$$y^{\text{new}}_{\text{variant knn}} = \text{sign}\left(\sum_{i=1}^{n}\mathbf{1}(||X_{\text{new}}^{T} - X_i||_2 < d) \times y_i\right)$$

The first method can deal with liner problem, but the second confit anything!

Is there anything in between?

(Spoiler: Yes, Kernel methods)

Next time: preview
Going beyond linear models

* Random projections and kernel methods

* Unsupervised learning, PCA and Kernel PCA