

Foundations of Supervised Machine Learning

From data to guarantees: an introduction to ensemble methods and PAC-Bayes

Guillaume Metzler

Université Lumière Lyon 2 – Laboratoire ERIC, UR 3083

guillaume.metzler@univ-lyon2.fr

Master MIAHS & MALIA – 2026/2027

Goal. Revisit the foundations of supervised machine learning *with the vocabulary and the tools* that we will use in the rest of the course:

- ▶ ensemble methods (bagging, random forests, boosting, stacking);
- ▶ PAC-Bayesian theory (generalization guarantees for randomized and majority-vote classifiers).

Philosophy. Very ML-oriented: every notion comes with a picture, an algorithm, or an experiment. Practice first, then algorithms, then ensembles, and the theory at the end, to explain and certify what we have observed.

Organisation

Lectures + exercises (pen & paper)

5 practical sessions in Python:

TP1 protocol – TP2 bagging

TP3 boosting – TP4–5 PAC-Bayes

Lecture notes: *Statistical Supervised Machine Learning*. Section numbers on the slides (§) refer to the notes.

Prerequisites: linear algebra, basic probability, a bit of Python.

Outline: four movements

I. Learning in practice

1. What is machine learning?
2. Data and representations
3. The learning framework: losses and risks
4. Underfitting, overfitting, bias and variance
5. Empirical risk minimization and regularization
6. Evaluating and selecting models

II. Algorithms

III. Combining models

IV. Guarantees

7. The main learning algorithms
8. Ensemble methods
9. A probabilistic toolbox
10. Generalization bounds
11. PAC-Bayes: a theory tailored to majority votes
12. Summary and references

I: the concepts and the methodology, valid for any model. II: the classical models seen through these concepts. III: how and why to combine them. IV: how to *prove* that a (combined) model generalizes – PAC-Bayes being the theory best suited to ensembles.

How to read these slides

Definition – Name of the notion

A **definition**: the precise meaning of a word we will use all the time.

Theorem / Proposition

A **result**: what is true, under which assumptions.

Block

Details, examples, variants.

Key message: what you should remember, or the link with the rest of the course.

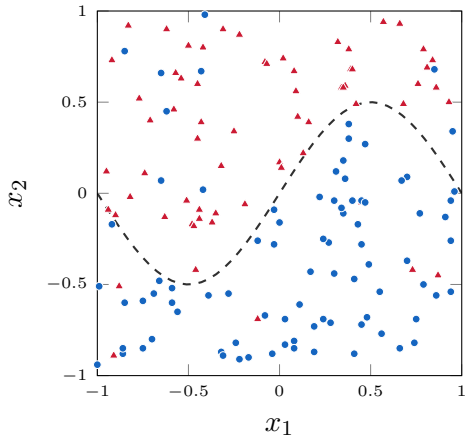
A question to think about, or a trap to avoid.

Warning

A frequent mistake.

Each part ends with a **take-home** slide. Numbers in the figures are **real experiments** (scikit-learn), most of them on the running example of the next slide.

The running example of this lecture (and of TP1)



- ▶ $\mathbf{x} = (x_1, x_2)$ uniform on $[-1, 1]^2$.
- ▶ “Clean” label: +1 (▲) above the curve $x_2 = \frac{1}{2} \sin(\pi x_1)$ (dashed), -1 (●) below.
- ▶ Each label is then **flipped with probability 10%** (noise).
- ▶ We observe $n = 150$ examples.

Why a synthetic problem? Because we **know the truth**: the best possible classifier is the dashed curve and its error is exactly 10%. We can therefore measure the *true* error of any model on a huge independent sample (20 000 points) and see what really happens.

Part 1

What is machine learning?

- ▶ Spam filtering, fraud detection, credit scoring
- ▶ Medical diagnosis from images or biological signals
- ▶ Recommendation (films, music, products)
- ▶ Speech recognition, machine translation, chatbots
- ▶ Predictive maintenance, forecasting of demand
- ▶ Search engines, ranking, advertising

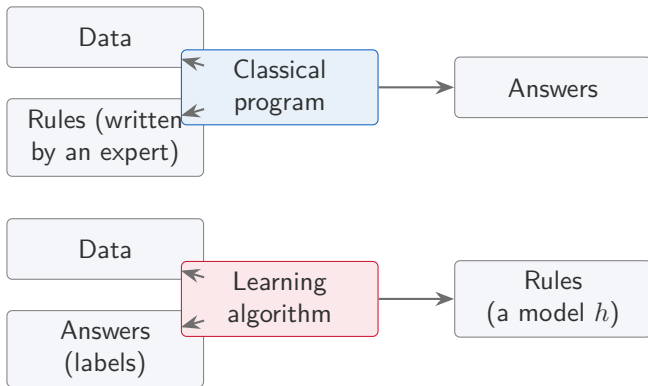
Definition – Machine learning (informal)

Design algorithms that **improve their performance on a task from experience**, i.e. from data, rather than from explicitly programmed rules.

“A computer program is said to learn from experience E with respect to some task T and performance measure P if its performance at T , as measured by P , improves with experience E .” –

T. Mitchell (1997)

Rules vs. learning



Learning is useful when the rules are **unknown**, **too complex** to be written (recognize a cat), or **change over time** (spam). The output of learning is itself a program: a *model h* .

A running example: cats vs. dogs

- ▶ Input x : a picture ($256 \times 256 \times 3$ pixels $\Rightarrow \approx 2 \cdot 10^5$ numbers).
- ▶ Output $y \in \{\text{cat}, \text{dog}\}$, coded $\{-1, +1\}$.
- ▶ We are given n labelled pictures $(x_1, y_1), \dots, (x_n, y_n)$.
- ▶ We want a function h such that $h(x) = y$ on new pictures.

Which **description** of a picture should we give to the algorithm? Raw pixels? Hand-crafted features? Learned features?



$y = -1$ (cat)



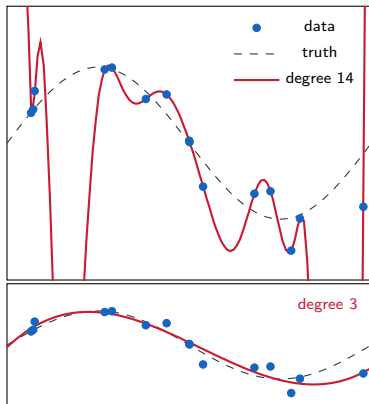
$y = +1$ (dog)

Same task, three generations of ML:
raw features $\rightarrow$ *engineered features* $\rightarrow$
learned features (deep learning).

Learning is approximating an unknown function

- ▶ There is an unknown relation between x and y (possibly noisy).
- ▶ We only see it through n **examples**.
- ▶ We choose a family of candidate functions $\mathcal{H}$ (lines, trees, ...) and pick in it the one that “best explains” the examples.
- ▶ The real test: what happens on **new** x 's?

On the right: which of the two curves would you trust to predict y at $x = 0.5$? The red one passes through every point. . .



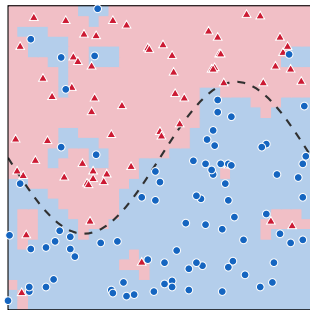
Memorizing is not learning

A student who learns by heart the solutions of last year's exam gets 20/20 on last year's exam... and fails this year.

A model that memorizes its training data (e.g. 1-nearest neighbour, a very deep tree) has **0 training error**, but its error on new data can be large.

The goal of learning is **generalization**: perform well on data that were *not* used to build the model. All the methodology of ML (test sets, cross-validation, regularization, bounds) exists to measure or to guarantee it.

1-NN on the running example



Training error **0%**, true error **20.1%** (best possible: 10%). Each noisy point creates its own small island.

Three ways of describing the data

Raw features

Pixel intensities, raw words, raw signal.

- + no expertise needed
- very high dimension, little structure (a shift of one pixel changes everything)

Engineered features

Shape of the ears, colour histograms, word counts, statistics of a signal.

- + compact, interpretable
- need domain knowledge, may miss information

Learned features

Intermediate layers of a neural network (§9), embeddings.

- + very powerful on images/text/sound
- need lots of data and compute, less interpretable

In this course, data are mostly **tabular**: each example is a vector $\mathbf{x} \in \mathbb{R}^d$ of (engineered) features. On tabular data, **tree ensembles** (random forests, gradient boosting) are still the state of the art $\Rightarrow$ one more reason to study them.

The main learning settings

Supervised

Data $(\mathbf{x}_i, y_i)$, labels known.
Learn $h : \mathcal{X} \rightarrow \mathcal{Y}$ predicting y from $\mathbf{x}$.

Classification, regression, ranking.

Unsupervised

Data $\mathbf{x}_i$ only.
Discover structure: groups, low-dimensional representation, density.
Clustering (k -means), PCA, density estimation.

Reinforcement

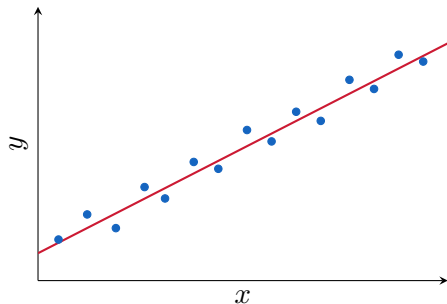
An agent acts in an environment and receives *rewards*.
Learn a policy maximizing the cumulative reward.
Games, robotics.

Between them: **semi-supervised** (few labels), **weakly supervised**, **self-supervised** (labels built from the data), **transfer learning** / **domain adaptation** (train and test distributions differ).

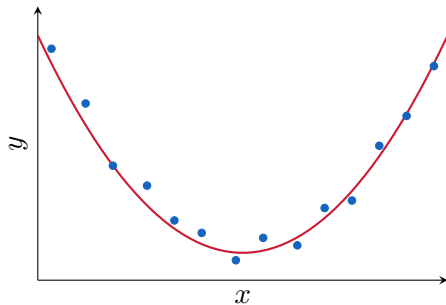
This course: **supervised learning**, mostly **binary classification** with $\mathcal{Y} = \{-1, +1\}$.

Supervised learning: regression

linear model



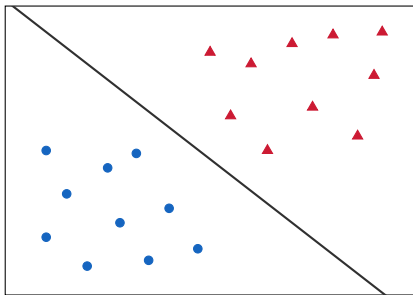
quadratic model



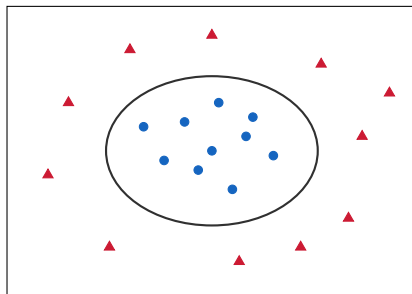
$\mathcal{Y} = \mathbb{R}$: predict a **real value** (price of a flat, temperature, consumption). The model may be linear in x or linear in *transformed features* $(x, x^2, \dots)$ – §8.4, §8.6.

Supervised learning: classification

linear separator

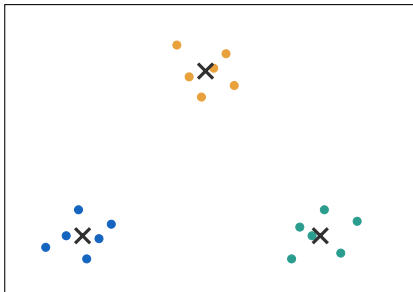


non-linear separator (kernel)



$\mathcal{Y}$ finite: predict a **class**. Binary ($\mathcal{Y} = \{-1, +1\}$, e.g. spam/not spam) or multi-class ($\mathcal{Y} = \{1, \dots, K\}$, e.g. digits). A classifier splits $\mathcal{X}$ into regions; the frontier is the *decision boundary*.

Unsupervised learning: an example with k -means



k -means (Lloyd's algorithm)

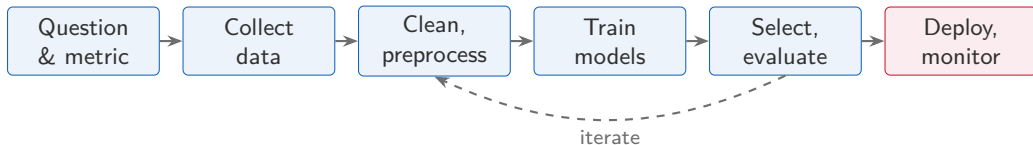
1. choose k initial centres;
2. assign each point to its nearest centre;
3. move each centre to the mean of its points;
4. repeat 2–3 until stabilization.

It minimizes the within-cluster inertia

$$\sum_j \sum_{i \in C_j} \|\mathbf{x}_i - \mu_j\|^2 \text{ (locally).}$$

No label, hence no “error”: how do we *evaluate* a clustering? This is why supervised learning is much easier to assess.

The machine learning workflow



- ▶ Most of the time is spent on **data** (collection, cleaning, features), not on the algorithm.
- ▶ **Evaluation** must mimic deployment: the model will face *new* data (§7, §11).
- ▶ The chosen **metric** must reflect the real cost of errors (a missed tumour $\neq$ a false alarm).

The core question of the course: *how well will my model perform on data it has never seen?*
⇒ empirical answers (protocol, Part 6) and theoretical answers (bounds, Parts 9–11).

- ▶ Machine learning builds a **model** (a program) **from examples** rather than from hand-written rules.
- ▶ Supervised learning: examples come with the answer y ; we want to predict y for **new** x .
- ▶ Classification (y in a finite set) vs. regression ($y \in \mathbb{R}$).
- ▶ The representation of the data (features) is often what matters most.
- ▶ The goal is **generalization**, not memorization: 0 training error means nothing by itself.

Next: data. How to turn real objects into vectors, and the pitfalls of real data.

Part 2

Data and representations

Data come in many forms

- ▶ **Tabular:** rows = individuals, columns = variables (customers, patients).
- ▶ **Images, videos:** grids of pixels.
- ▶ **Text:** sequences of words / tokens.
- ▶ **Time series, signals:** sensors, finance, EEG.
- ▶ **Graphs:** social networks, molecules.
- ▶ **Sets, trees, distributions...**

Most algorithms of this course need **vectors** $\mathbf{x} \in \mathbb{R}^d$.

Hence a first, often decisive, step: build a **representation** $\phi : \text{object} \mapsto \mathbf{x} \in \mathbb{R}^d$.

Examples of representations

Text $\rightarrow$ bag of words / TF-IDF / embeddings.

Time series $\rightarrow$ statistics (mean, variance, spectrum).

Molecule $\rightarrow$ fingerprint, graph kernel.

Tabular data: the design matrix

id	age	income	city	smoker	y
1	34	2 100	Lyon	no	+1
2	51	3 400	Paris	yes	-1
3	27	1 650	Lyon	no	+1
$\vdots$					$\vdots$
n	45	2 900	Nice	yes	-1

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n \times d}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

- ▶ n = number of **examples** (rows, observations, instances).
- ▶ d = number of **features** (columns, variables, attributes).
- ▶ x_{ij} = value of feature j for example i .
- ▶ y_i = **label** (target, response) of example i .

The sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is called the **training set** (or learning sample).

Quantitative

Continuous: income, temperature, blood pressure.

Discrete: number of children, of visits.

Qualitative (categorical)

Nominal: city, colour, blood group (no order).

Ordinal: education level, satisfaction (low < medium < high).

Binary: smoker yes/no.

Encoding categorical features

- ▶ *One-hot:* $\text{city} \in \{\text{Lyon}, \text{Paris}, \text{Nice}\} \mapsto (1, 0, 0), (0, 1, 0), (0, 0, 1)$.
- ▶ *Ordinal:* low/medium/high $\mapsto 0/1/2$ – only if the order is meaningful!
- ▶ *Target encoding:* replace a category by the mean of y in it – **risk of leakage**, fit it on the training data only.

Trees can split on categories directly; linear models, k -NN and SVMs need numbers *and* comparable scales.

Encoding a categorical feature: a worked example

id	city
1	Lyon
2	Paris
3	Lyon
4	Nice

⇒

id	city=Lyon	city=Paris	city=Nice
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1

Why not Lyon = 1, Paris = 2, Nice = 3?

It invents an order and distances: “Nice – Lyon = 2 × (Paris – Lyon)”. A linear model or a k -NN would use this meaningless geometry.

One-hot: one binary column per category, all categories at the same distance $\sqrt{2}$ from each other. With many categories (zip codes), d explodes ⇒ group rare categories, or use trees / target encoding.

- ▶ **Missing values:** drop, impute (mean, median, model), or add an indicator “was missing”.
- ▶ **Outliers:** errors or rare but real events? Robust losses and trees are less sensitive.
- ▶ **Duplicates** (dangerous if one copy is in train and the other in test).
- ▶ **Label noise:** annotators make mistakes (boosting is sensitive to it – TP3).
- ▶ **Class imbalance:** 1% fraud, 99% normal.

Imbalance: a trap

With 1% of positives, the constant classifier $h(\mathbf{x}) = -1$ has an accuracy of 99%... and is useless.

⇒ use adapted metrics (precision, recall, F1, AUC – Part 8) or re-weighting.

Every preprocessing step that uses the data (imputation by the mean, scaling, selection of features...) is **part of the model**: it must be learned on the training data only (§11.3–11.4).

Missing values: a worked example

id	age	income	y
1	34	2 100	+1
2	?	3 400	-1
3	27	?	+1
4	45	2 900	-1

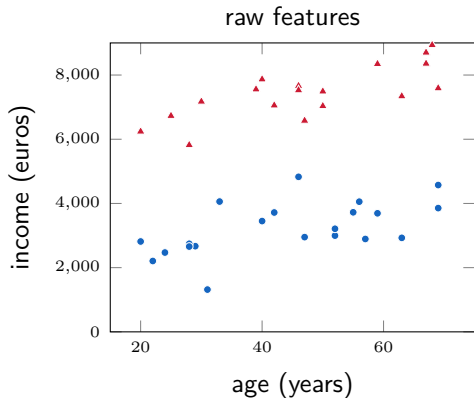
strategy	row 2 (age)	row 3 (income)
drop rows	removed	removed
mean	35.3	2 800
median	34	2 900
+ indicator	35.3, miss=1	2 800, miss=1

- ▶ Dropping rows wastes data and can **bias** the sample (who does not declare their income?).
- ▶ Mean/median imputation is simple; the **indicator** keeps the information “was missing”, which can be predictive.
- ▶ Model-based imputation (k -NN, iterative regression) is better but more costly.

Leakage

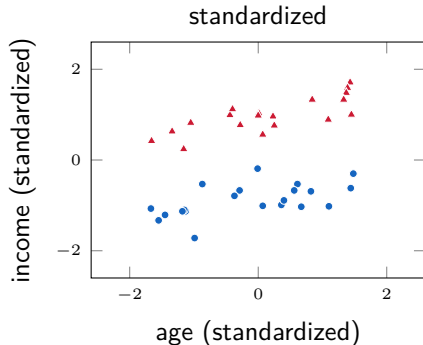
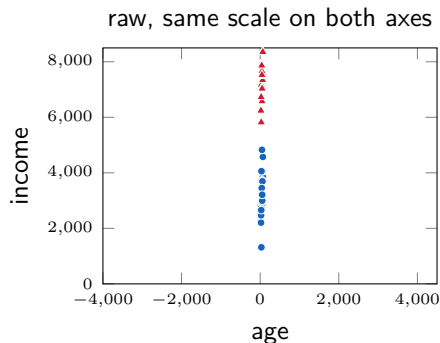
The mean must be computed on the **training** examples only.

Why should we normalize? A picture



- ▶ Euclidean distance:
 $\|\mathbf{x} - \mathbf{x}'\|^2 = (\Delta\text{age})^2 + (\Delta\text{income})^2$.
- ▶ A difference of 40 years counts as much as a difference of 40 euros!
- ▶ Distances are **entirely** driven by income: on this plot, the axes are not at the same scale, so the picture lies – for the algorithm, the cloud is a vertical line.
- ▶ Same problem for every method based on **distances, dot products or norms**: k -NN, kernels / SVM, k -means, PCA, ridge / Lasso penalties, gradient descent.

Standardization: before and after



Left: with the true geometry (same unit on both axes) the age has no influence at all. Right: after standardization both features have mean 0 and standard deviation 1 and **both** can be used. TP1: Gaussian SVM on australian: test error 0.342 $\rightarrow$ 0.147. Trees are unchanged (0.154): they only compare a feature with a threshold.

Normalization: three classical choices

For each feature j , computed **on the training set only**:

Min-max

$$x'_{ij} = \frac{x_{ij} - m_j}{M_j - m_j} \in [0, 1]$$

m_j , M_j : min and max of feature j . Sensitive to outliers.

Standardization

$$x'_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

mean 0, std 1 for each feature. The default choice.

Normalization (rows)

$$\mathbf{x}'_i = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$$

Each *example* on the unit sphere (text, histograms).

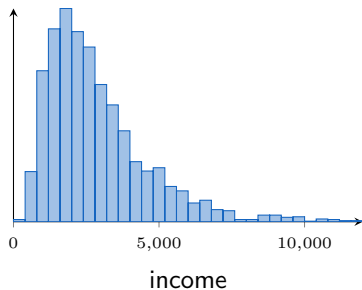
Also: robust scaling (median / interquartile range), log transform of skewed variables (incomes, counts).

Golden rule

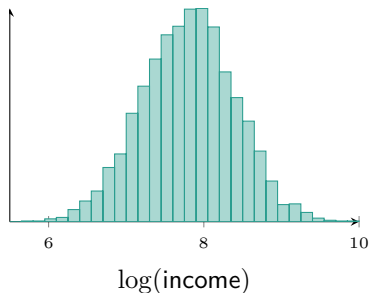
Fit the scaler on the training set; *apply* the same transformation to validation/test data. Never compute $\bar{x}_j$, s_j on the whole data set before splitting.

Skewed features: the log transform

raw: long right tail



after log: symmetric



- ▶ Incomes, prices, counts, durations are often **log-normal**: a few huge values dominate means, variances and squared losses.
- ▶ $x \mapsto \log(1 + x)$ compresses the tail; then standardize. A ratio becomes a difference: “twice richer” is the same step everywhere.
- ▶ Again, trees do not care (monotone transformation); linear models, k -NN and SVMs do.

Normalization in practice: pipelines

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

model = make_pipeline(StandardScaler(),
                      SVC(kernel="rbf", C=10, gamma=0.1))
model.fit(X_train, y_train) # scaler fitted on train
err = 1 - model.score(X_test, y_test)
```

A Pipeline treats “preprocessing + model” as *one* estimator: during cross-validation, the scaler is re-fitted on each training fold $\Rightarrow$ no leakage.

Question: why is `StandardScaler().fit(X)` followed by a split a (small) mistake? Why does it become a big mistake with feature selection? (TP1: 0.85 vs 0.53.)

Definition – Data-generating distribution

There is an **unknown** probability distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. The training examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ are drawn **independently** from $\mathcal{D}$: $S \sim \mathcal{D}^n$. Future (test) examples are drawn from the **same** $\mathcal{D}$.

Why we need it

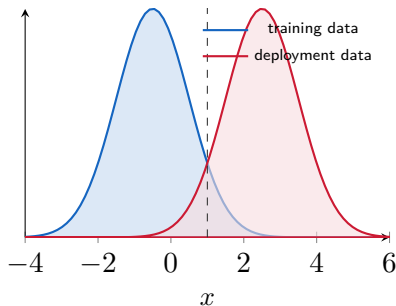
- ▶ “Same distribution”: what we learn on S is relevant for the future.
- ▶ “Independent”: each example brings new information $\Rightarrow$ laws of large numbers, concentration (Part 9) and all generalization bounds.

When it fails

- ▶ Time series (today depends on yesterday);
- ▶ several rows per patient / per customer (groups);
- ▶ distribution shift (new sensor, new country, Covid...).

$\Rightarrow$ adapt the split: temporal split, group k -fold.

When the i.i.d. hypothesis fails: distribution shift

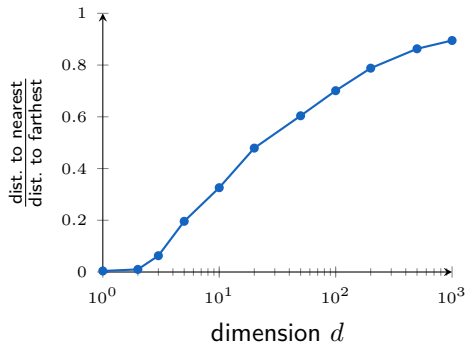


- ▶ The model was trained where the blue density lives; it is used where the red one lives.
- ▶ In the region $x > 3$ it has seen almost no example: its predictions there are **extrapolations**.
- ▶ Examples: a new hospital, a new camera, a new year of data, a new population of customers.

All the guarantees of this course assume the **same** $\mathcal{D}$ for training and test. When it is not the case: monitoring, re-training, domain adaptation.

- ▶ $n \gg d$: classical statistics, many models work.
- ▶ $d \gg n$ (genomics: $d = 20\,000$ genes, $n = 100$ patients): a linear model can fit *any* labels $\Rightarrow$ regularization is mandatory.
- ▶ Images, text: d huge but data live near a low-dimensional structure.

Rule of thumb: the more *flexible* the model and the larger d , the more data we need. The theory makes this precise through a **complexity** term (Parts 5, 10, 11).



500 uniform points in $[0, 1]^d$ and a random query point (real simulation): in high dimension the nearest and the farthest points are almost at the same distance – “nearest” loses its meaning.

Volume grows exponentially.

- ▶ To cover $[0, 1]^d$ with cubes of side 0.1, we need 10^d cubes.
- ▶ To have one example per cube when $d = 10$: 10^{10} examples.
- ▶ A neighbourhood containing 10% of the data has side $0.1^{1/d}$: 0.79 for $d = 10$, i.e. it is *not local* any more.

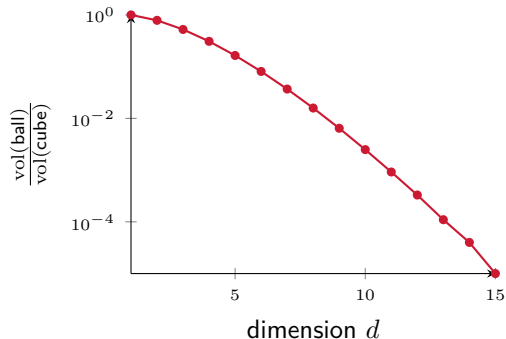
Consequence

Purely local methods (k -NN, histograms) degrade quickly with d .

What saves us

- ▶ Data are not uniform: they concentrate near low-dimensional structures.
- ▶ Feature selection / extraction (PCA, domain knowledge).
- ▶ Models with *structural* assumptions: linear, additive, trees choosing relevant variables.
- ▶ Regularization: prefer simple functions.
- ▶ **Averaging many models** (ensembles) to reduce variance (Part 8).

Where is the volume of a high-dimensional cube?



- ▶ Ball of radius $\frac{1}{2}$ inscribed in the cube $[0, 1]^d$.
- ▶ $d = 2$: the disc fills 79% of the square.
 $d = 10$: the ball fills 0.25% of the cube. $d = 20$: $2.5 \cdot 10^{-8}$.
- ▶ Almost all the volume is in the **corners**: uniform data are far from the centre and far from each other.

Our 2D intuition is misleading in high dimension. Local methods need exponentially many points; models with structure (linear, additive, trees) and **averaging** are the way out.

- ▶ Most algorithms need a **vector representation** $\mathbf{x} \in \mathbb{R}^d$: encode categories (one-hot), handle missing values, transform skewed variables.
- ▶ **Scale matters** for every method based on distances, dot products or norms; standardize. Trees do not care.
- ▶ Every data-dependent preprocessing is **part of the model**: fit it on the training data only (pipelines).
- ▶ The **i.i.d. hypothesis** links the past (training data) to the future (test data); check that it is reasonable.
- ▶ In **high dimension**, distances concentrate and volume escapes to the corners: flexible local methods need huge samples.

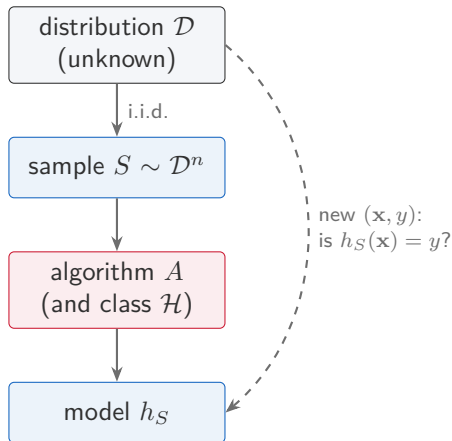
Part 3

The learning framework: losses and risks

The setting (§5)

- ▶ Input space $\mathcal{X}$ (often $\mathbb{R}^d$), output space $\mathcal{Y}$.
- ▶ Unknown distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$.
- ▶ Training sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim \mathcal{D}^n$.
- ▶ A **hypothesis class** $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ (linear functions, trees of depth $\leq 5 \dots$).
- ▶ A **learning algorithm** A :
 $S \mapsto h_S = A(S) \in \mathcal{H}$.

Goal: h_S should make **few errors on new examples** drawn from $\mathcal{D}$.



The nature of $\mathcal{Y}$ defines the task

Task	$\mathcal{Y}$	Example	Typical loss
Binary classification	$\{-1, +1\}$	spam / not spam	0-1, hinge, logistic
Multi-class classification	$\{1, \dots, K\}$	digit recognition	0-1, cross-entropy
Multi-label	$\{0, 1\}^K$	tags of a picture	Hamming
Regression	$\mathbb{R}$	price of a flat	squared, absolute
Multi-output regression	$\mathbb{R}^K$	position of a robot	$\ \cdot\ _2^2$
Ordinal regression	$\{1 < \dots < K\}$	ratings	absolute on ranks
Ranking	orders	search engine	pairwise

Throughout: binary classification with $\mathcal{Y} = \{-1, +1\}$ (convenient: $y h(\mathbf{x}) > 0 \Leftrightarrow$ correct prediction) and regression with $\mathcal{Y} = \mathbb{R}$. Many models output a **score** $f(\mathbf{x}) \in \mathbb{R}$ and predict $h(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$.

Definition – Loss

A loss is a function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$; $\ell(h(\mathbf{x}), y)$ measures the cost of predicting $h(\mathbf{x})$ when the truth is y .

Classification

- ▶ 0–1 loss: $\ell(h(\mathbf{x}), y) = \mathbb{1}[h(\mathbf{x}) \neq y]$.
- ▶ Asymmetric costs:
 $c_{FN} \mathbb{1}[h = -1, y = +1] +$
 $c_{FP} \mathbb{1}[h = +1, y = -1]$.

Regression

- ▶ squared loss $(h(\mathbf{x}) - y)^2$;
- ▶ absolute loss $|h(\mathbf{x}) - y|$ (robust);
- ▶ Huber, ε -insensitive (SVR)...

The loss encodes **what we care about**. Choosing it is a modelling decision, not a technical detail.

Definition – True risk (generalization error)

$$R_{\mathcal{D}}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\ell(h(\mathbf{x}), y)].$$

For the 0–1 loss: $R_{\mathcal{D}}(h) = \mathbb{P}_{(\mathbf{x}, y) \sim \mathcal{D}}(h(\mathbf{x}) \neq y)$, the probability of error on a new example.

- ▶ The quantity we **want** to minimize.
- ▶ But $\mathcal{D}$ is unknown $\Rightarrow R_{\mathcal{D}}(h)$ **cannot be computed**.
- ▶ Everything in this course revolves around *estimating* or *bounding* $R_{\mathcal{D}}(h)$:
 - on a test set (Part 6): unbiased estimate, confidence interval;
 - on the training set + a complexity term (Parts 9–11): generalization bounds.

Definition – Empirical risk (training error)

$$\hat{R}_S(h) = \frac{1}{n} \sum_{i=1}^n \ell(h(\mathbf{x}_i), y_i).$$

- ▶ Computable!
- ▶ For a **fixed** h (chosen *before* seeing S):
 $\mathbb{E}_{S \sim \mathcal{D}^n} [\hat{R}_S(h)] = R_{\mathcal{D}}(h)$, unbiased; and
 $\hat{R}_S(h) \rightarrow R_{\mathcal{D}}(h)$ (law of large numbers).
- ▶ For h_S **learned on** S : $\hat{R}_S(h_S)$ is **optimistic** – the model has seen these examples.

The central issue

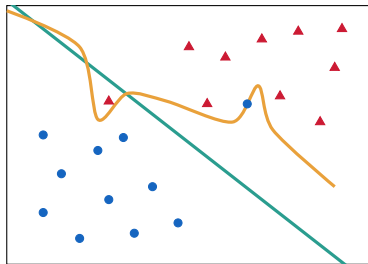
A deep tree has $\hat{R}_S(h_S) = 0$ but $R_{\mathcal{D}}(h_S) \approx 0.24$ (TP1, Bayes risk 0.10).

The gap $R_{\mathcal{D}}(h_S) - \hat{R}_S(h_S)$ is the **generalization gap**. Controlling it is the object of learning theory.

What is a good model?

A good model...

- ▶ has a **low true risk** $R_{\mathcal{D}}(h)$ – not only a low training error;
- ▶ is **stable**: small changes in S do not change it much;
- ▶ is as **simple** as possible (Occam's razor);
- ▶ ideally comes with a **guarantee**: “with probability $\geq 95\%$, $R_{\mathcal{D}}(h) \leq 0.12$ ”;
- ▶ is computationally tractable, interpretable when needed, fair...



green: simple, 2 training errors. orange: no training error, but...

The Bayes classifier: the best possible model

Let $\eta(\mathbf{x}) = \mathbb{P}(y = +1 \mid \mathbf{x})$ (the regression function).

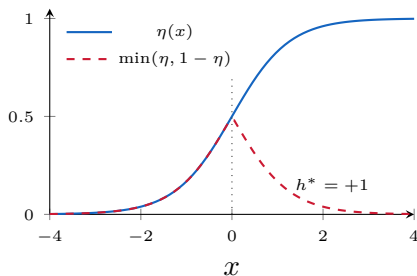
Proposition (Bayes classifier)

For the 0–1 loss, the classifier minimizing $R_{\mathcal{D}}$ over *all* functions is

$$h^*(\mathbf{x}) = \begin{cases} +1 & \text{if } \eta(\mathbf{x}) \geq 1/2, \\ -1 & \text{otherwise,} \end{cases}$$

and its risk is the **Bayes risk** $R^* = \mathbb{E}_{\mathbf{x}}[\min(\eta(\mathbf{x}), 1 - \eta(\mathbf{x}))]$.

$R^* > 0$ as soon as labels are noisy: *no algorithm* can do better. Regression: $h^*(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}]$.



The Bayes risk is the area under the dashed curve (weighted by the density of x).

Proof of the Bayes classifier

For any h and any $\mathbf{x}$, conditionally on $\mathbf{x}$:

$$\mathbb{P}(h(\mathbf{x}) \neq y \mid \mathbf{x}) = \mathbb{1}[h(\mathbf{x}) = +1] (1 - \eta(\mathbf{x})) + \mathbb{1}[h(\mathbf{x}) = -1] \eta(\mathbf{x}).$$

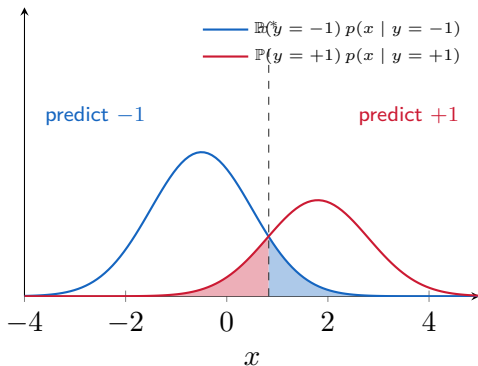
- ▶ This is minimized pointwise by choosing $h(\mathbf{x}) = +1$ iff $1 - \eta(\mathbf{x}) \leq \eta(\mathbf{x})$, i.e. $\eta(\mathbf{x}) \geq 1/2$.
- ▶ The minimum value is $\min(\eta(\mathbf{x}), 1 - \eta(\mathbf{x}))$.
- ▶ Taking the expectation over $\mathbf{x}$ gives $R_{\mathcal{D}}(h) \geq R_{\mathcal{D}}(h^*) = R^*$. □

Excess risk

$$R_{\mathcal{D}}(h) - R^* = \mathbb{E}_{\mathbf{x}} [|2\eta(\mathbf{x}) - 1| \mathbb{1}[h(\mathbf{x}) \neq h^*(\mathbf{x})]]: \text{disagreeing with } h^* \text{ costs little where } \eta \approx \frac{1}{2}.$$

Many methods are “plug-in”: estimate η (logistic regression, k -NN, a forest) and threshold at $1/2$.

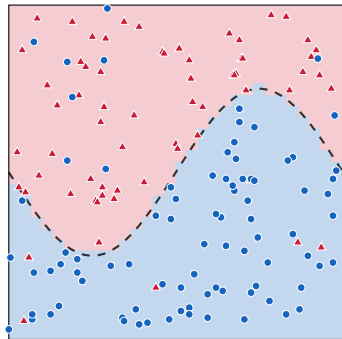
The Bayes classifier in one dimension



- ▶ Two classes with overlapping distributions.
- ▶ At each x , predict the class with the highest curve: this is $\eta(x) \geq \frac{1}{2}$.
- ▶ The frontier x^* is where the curves cross.
- ▶ The **shaded area** is the Bayes risk: the examples of the minority class on each side, which *no* classifier can recover.

Move the threshold away from x^* : which shaded area grows, which one shrinks? Why is the total always larger?

The Bayes classifier of the running example



- ▶ Above the curve:
 $\eta(\mathbf{x}) = \mathbb{P}(y = +1|\mathbf{x}) = 0.9$; below:
 $\eta(\mathbf{x}) = 0.1$.
- ▶ Bayes classifier: $h^*(\mathbf{x}) = +1$ above the curve, -1 below (background colour).
- ▶ Bayes risk: $R^* = \mathbb{E}[\min(\eta, 1 - \eta)] = 0.1$.
The 10% of flipped points (the “wrong colour” points on the plot) are errors for every classifier.

A good learning algorithm should approach 10% of true error. Getting *less* than 10% on the training set is a sign that the model has learned the noise.

The 0–1 loss is hard to optimize

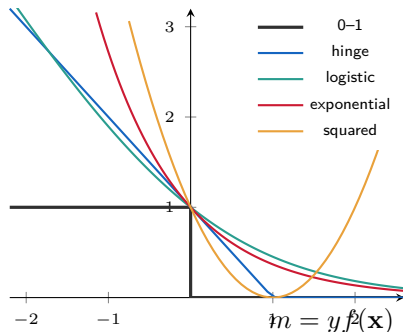
With a score f and $h = \text{sign } f$: the 0–1 loss depends only on the **margin**

$m = yf(\mathbf{x})$:

$$\ell_{01}(m) = \mathbb{1}[m \leq 0].$$

- ▶ not continuous, gradient zero almost everywhere;
- ▶ minimizing the empirical 0–1 risk of linear classifiers is NP-hard.

Idea: replace it by a **convex surrogate**
 $\phi(m) \geq \mathbb{1}[m \leq 0]$ (§8.1).



Convex surrogate losses (§8.1)

Loss	$\phi(m)$	Used by	Comment
hinge	$\max(0, 1 - m)$	SVM	sparse solutions (support vectors)
logistic	$\log_2(1 + e^{-m})$	logistic regression	gives probabilities
exponential	e^{-m}	AdaBoost	very sensitive to outliers
squared	$(1 - m)^2$	least squares, kernel ridge	penalizes $m > 1$ too

All are **upper bounds** of the 0–1 loss: minimizing them pushes the training error down, and they are *convex* $\Rightarrow$ efficient optimization.

Calibration (informal)

For these losses, minimizing the *true* ϕ -risk over all functions yields $\text{sign } f^* = h^*$ (Bayes). Bartlett, Jordan, McAuliffe (2006).

The exponential loss will reappear with **boosting** (TP3), the hinge loss and margins with SVMs, and the 0–1 loss with **PAC-Bayes** bounds (which deal with it directly!).

Losses on a worked example

A linear classifier outputs a score $f(\mathbf{x})$; we predict $\text{sign } f(\mathbf{x})$. Five test examples:

example	y	$f(\mathbf{x})$	margin $m = yf(\mathbf{x})$	0-1	hinge	logistic	exponential
A	+1	2.5	2.5	0	0	0.11	0.08
B	-1	-0.8	0.8	0	0.2	0.53	0.45
C	+1	0.1	0.1	0	0.9	0.93	0.90
D	-1	0.3	-0.3	1	1.3	1.23	1.35
E	+1	-3	-3	1	4	4.40	20.1
average (empirical risk)				0.40	1.28	1.44	4.57

- ▶ C is correct but **not confident**: surrogate losses still penalize it.
- ▶ E is a confident mistake: the exponential loss explodes ($e^3 \approx 20$).

The 0-1 loss only counts mistakes; surrogates also look at **how wrong / how sure**. One outlier like E can dominate the exponential risk: this is why AdaBoost is sensitive to label noise.

Redefining the risk with a surrogate

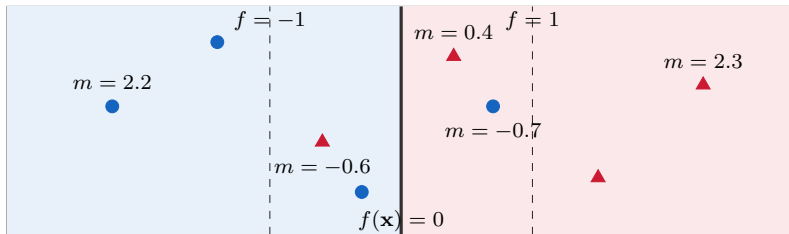
For a score $f : \mathcal{X} \rightarrow \mathbb{R}$ and a surrogate ϕ :

$$R_{\mathcal{D}}^{\phi}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\phi(yf(\mathbf{x}))], \quad \hat{R}_S^{\phi}(f) = \frac{1}{n} \sum_{i=1}^n \phi(y_i f(\mathbf{x}_i)).$$

- ▶ Since $\phi \geq \mathbb{1}[\cdot \leq 0]$: $R_{\mathcal{D}}(\text{sign } f) \leq R_{\mathcal{D}}^{\phi}(f)$.
- ▶ We *optimize* $\hat{R}_S^{\phi}$ but we *evaluate* with the 0–1 loss (or any metric of interest).
- ▶ The margin $yf(\mathbf{x})$ measures **confidence**: large positive margin = correct and confident.

Margins are a recurring theme: SVM (maximize the margin), boosting (increases margins even after the training error reaches 0), majority votes (the margin of a vote is the difference between the weights of the correct and wrong voters, Part 11).

Margins: a picture



- ▶ Horizontal axis = score $f(\mathbf{x})$. $\blacktriangle$: $y = +1$, $\bullet$: $y = -1$.
- ▶ $m = yf(\mathbf{x}) > 0$: correct; $m < 0$: error; $0 < m < 1$: correct but inside the “margin band” – the hinge loss still pays $1 - m$.
- ▶ Hinge-loss methods (SVM) push all examples beyond $m = 1$; boosting keeps increasing the smallest margins.

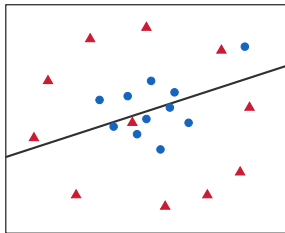
- ▶ Setting: unknown $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, i.i.d. sample S , a class $\mathcal{H}$, an algorithm $S \mapsto h_S$.
- ▶ The **loss** says what an error costs; the **true risk** $R_{\mathcal{D}}(h)$ is the expected loss on new data – what we care about, and what we cannot compute.
- ▶ The **empirical risk** $\hat{R}_S(h)$ is computable, unbiased for a fixed h , but **optimistic** for a learned h_S .
- ▶ The **Bayes classifier** $h^* = \text{sign}(2\eta - 1)$ is the best possible; its risk R^* is the irreducible error.
- ▶ We optimize **convex surrogates** of the margin $yf(\mathbf{x})$ (hinge, logistic, exponential) and evaluate with the metric of interest.

Part 4

Underfitting, overfitting, bias and variance

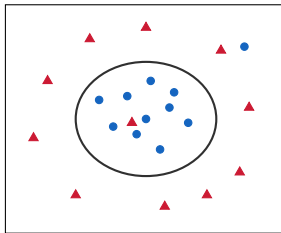
Underfitting and overfitting

underfitting



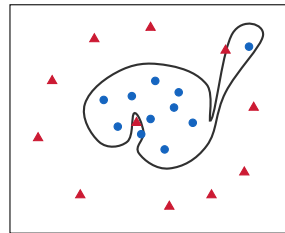
Model too simple: high training *and* test errors (**bias**).

good fit



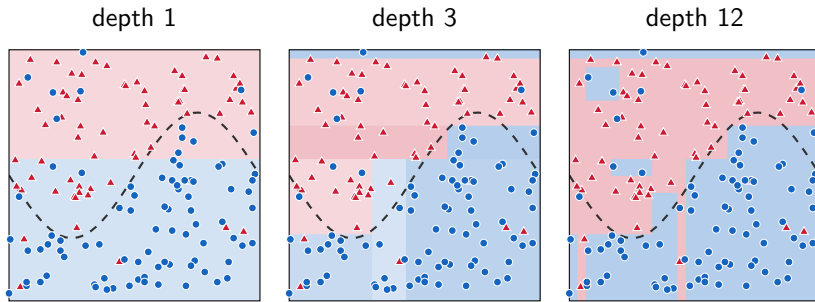
Captures the structure, ignores the noise.

overfitting



Fits the noise: training error 0, high test error (**variance**).

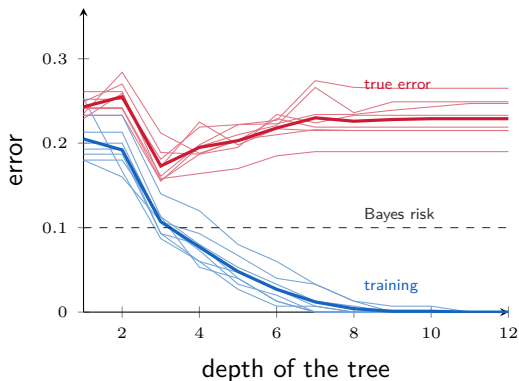
Overfitting on the running example: trees of increasing depth



	depth 1	depth 3	depth 12
training error $\hat{R}_S(h_S)$	0.207	0.100	0.000
true error $R_D(h_S)$	0.235	0.178	0.216

Decision trees (Part 7) of increasing depth: deeper trees always lower the training error. The true error (Bayes: 0.10) first decreases, then increases.

Overfitting is also instability

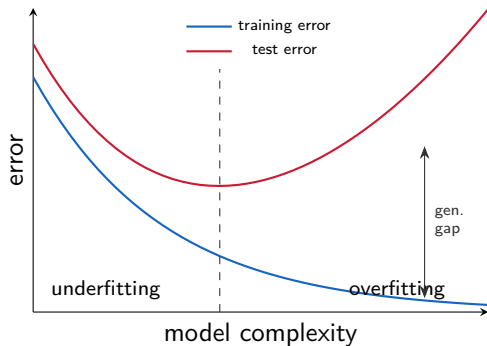


8 different training samples ($n = 150$) of the running example, one thin line per sample, thick line = average.

- ▶ Training error $\rightarrow 0$ for every sample.
- ▶ True error: minimal around depth 3–4, then increases.
- ▶ For deep trees the thin red lines **spread out**: the result depends a lot on the particular sample. This is the **variance** of the procedure.

Bagging and random forests reduce exactly this variance (Part 8, TP2).

The classical U-shaped curve

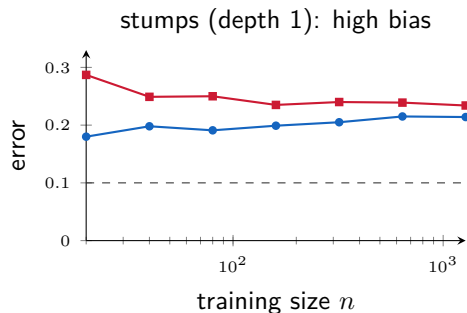


- ▶ Training error always decreases with complexity.
- ▶ Test error first decreases (bias $\downarrow$), then increases (variance $\uparrow$).
- ▶ The gap train/test grows with complexity.

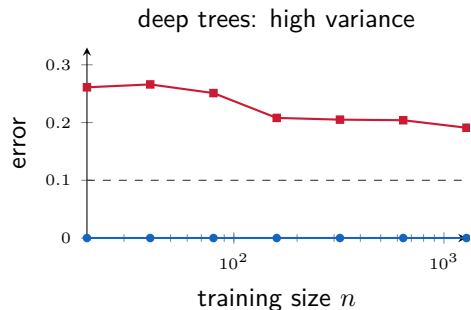
“Complexity” = depth of a tree, $1/k$ in k -NN, γ or C in SVM, $1/\lambda$, degree of a polynomial, number of boosting rounds. . .

Modern twist: very over-parameterized models (deep nets, large forests) can show a *second descent*.

Diagnosing: learning curves (running example)



Both errors meet at ≈ 0.22 , far above 0.10:
more data will not help; use a richer model.



Training error 0, large gap that closes slowly:
more data helps; or regularize, or **average many trees**.

Blue: training error. Red: true error. Dashed: Bayes risk. Averages over 10 samples for each n .

What to do?

Underfitting (high bias)

- ▶ richer class of models (deeper trees, kernels, more features);
- ▶ less regularization ($\lambda \downarrow$);
- ▶ better features / representation;
- ▶ **boosting**: combine many weak (biased) models sequentially to reduce bias.

Overfitting (high variance)

- ▶ more data, data augmentation;
- ▶ more regularization, simpler model, fewer features;
- ▶ early stopping;
- ▶ **bagging / random forests**: average many unstable models to reduce variance.

The two families of ensemble methods attack the two terms of the error: **bagging** → **variance**, **boosting** → **bias** (and also variance in practice). Let us make “bias” and “variance” precise.

Bias–variance decomposition (regression, §10.1)

Model $y = f^*(\mathbf{x}) + \varepsilon$, $\mathbb{E}[\varepsilon] = 0$, $\text{Var}(\varepsilon) = \sigma^2$. The learned h_S is random through S . Let $\bar{h}(\mathbf{x}) = \mathbb{E}_S[h_S(\mathbf{x})]$ be the *average model*. At a fixed point $\mathbf{x}$:

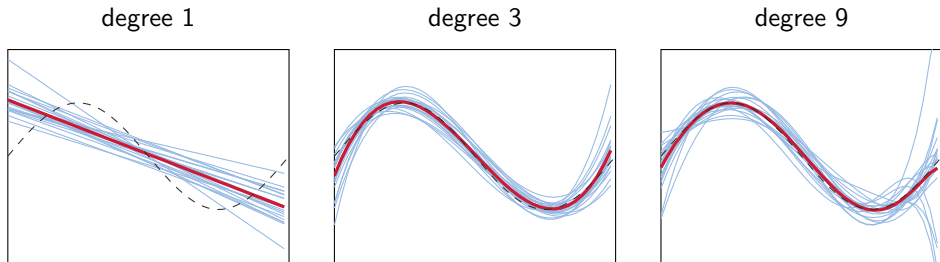
$$\mathbb{E}_{S,\varepsilon}[(y - h_S(\mathbf{x}))^2] = \underbrace{\sigma^2}_{\text{noise}} + \underbrace{(f^*(\mathbf{x}) - \bar{h}(\mathbf{x}))^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_S[(h_S(\mathbf{x}) - \bar{h}(\mathbf{x}))^2]}_{\text{variance}}.$$

- ▶ **Noise:** irreducible (Bayes risk).
- ▶ **Bias:** systematic error of the average model – the class is too poor.
- ▶ **Variance:** sensitivity of h_S to the particular sample.

Proof in three lines

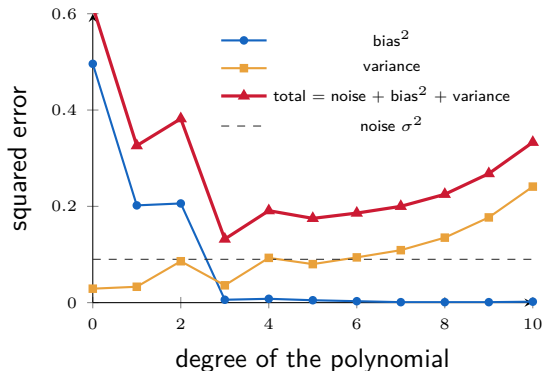
Write $y - h_S = \varepsilon + (f^* - \bar{h}) + (\bar{h} - h_S)$ and expand the square. Cross terms vanish: $\mathbb{E}[\varepsilon] = 0$ with ε independent of S , and $\mathbb{E}_S[\bar{h} - h_S] = 0$ by definition of $\bar{h}$.

Bias and variance, seen on 20 training samples



Each thin blue curve is the model learned on a different sample of 20 points; red = their average $\bar{h}$; dashed = the truth f^* . Degree 1: the curves agree with each other (low variance) but $\bar{h}$ is far from f^* (high bias). Degree 9: $\bar{h} \approx f^*$ (low bias) but each curve is wild (high variance).

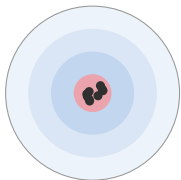
The trade-off, measured



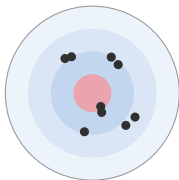
- ▶ 300 training samples of 20 points for each degree; bias² and variance averaged over x .
- ▶ Bias² collapses at degree 3 (a cubic can follow one period of a sine).
- ▶ Variance grows with the degree.
- ▶ The total error is minimal at degree 3.

The noise floor $\sigma^2 = 0.09$ is the regression analogue of the Bayes risk.

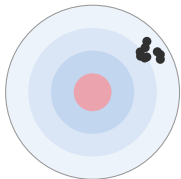
Bias and variance: the dartboard



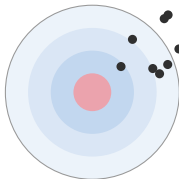
low bias, low
variance



low bias, high
variance



high bias, low
variance



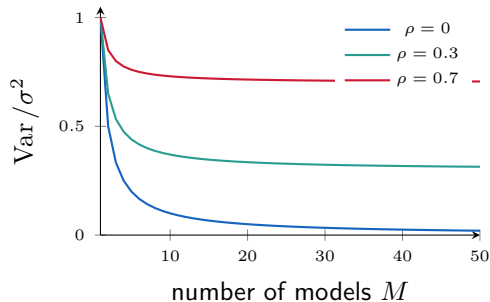
high bias, high
variance

Each dot = a model learned on a different sample S . Centre = f^* . **Averaging** the dots moves the prediction to their centre: it kills the variance but does not change the bias.

Averaging reduces variance

Let $h_1, \dots, h_M$ be M models, each with variance σ^2 at $\mathbf{x}$ and pairwise correlation ρ . Their average $\bar{h}_M = \frac{1}{M} \sum_m h_m$ has

$$\text{bias}(\bar{h}_M) = \text{bias}(h_m), \quad \text{Var}(\bar{h}_M) = \rho \sigma^2 + \frac{1 - \rho}{M} \sigma^2.$$



- ▶ Independent models: variance $\rightarrow 0$.
- ▶ Correlated models: floor at $\rho \sigma^2$.
- ▶ Models learned on the *same* data are highly correlated.

Bagging creates diversity by **resampling** the data; random forests add **random feature subsets** to lower ρ (TP2).

- ▶ For the 0–1 loss there is no exact additive decomposition, but the intuition remains (Domingos, 2000; §10.1 for the general error decomposition).
- ▶ Unstable classifiers (deep trees, 1-NN): **high variance** $\Rightarrow$ averaging helps a lot.
- ▶ Stable classifiers (linear, large- k k -NN, SVM with small γ): little to gain by averaging.
- ▶ In classification, averaging can even **reduce the bias**: a majority vote of trees can represent boundaries that no single tree of the class represents.

Majority vote of M classifiers: $B(\mathbf{x}) = \text{sign} \left(\sum_m q_m h_m(\mathbf{x}) \right)$, $q_m \geq 0$, $\sum_m q_m = 1$. This object is the main character of the rest of the course: **ensemble methods** build (h_m, q_m) ; **PAC-Bayes** bounds $R_{\mathcal{D}}(B)$.

- ▶ Underfitting = high **bias** (model too rigid); overfitting = high **variance** (model too sensitive to the sample).
- ▶ Test error vs. complexity is U-shaped; training error always decreases.
- ▶ **Learning curves** tell which side you are on: plateau $\Rightarrow$ bias, gap $\Rightarrow$ variance.
- ▶ Regression: error = noise + bias² + variance (exact decomposition).
- ▶ **Averaging** M models keeps the bias and divides the variance – down to the correlation floor $\rho\sigma^2$.
- ▶ Bagging/forests fight variance, boosting fights bias. The object of study: the **majority vote**.

Part 5

Empirical risk minimization and regularization

Definition – ERM

Given a class $\mathcal{H}$ and a loss ℓ :
$$h_S \in \arg \min_{h \in \mathcal{H}} \hat{R}_S(h) = \frac{1}{n} \sum_{i=1}^n \ell(h(\mathbf{x}_i), y_i).$$

- ▶ Natural: replace the unknown $R_{\mathcal{D}}$ by its estimate $\hat{R}_S$.
- ▶ Least squares, logistic regression, SVM, trees (greedily), neural networks are all (variants of) ERM.
- ▶ Works if $\hat{R}_S(h) \approx R_{\mathcal{D}}(h)$ **uniformly** over $\mathcal{H}$.

The danger

If $\mathcal{H}$ is too rich, $\hat{R}_S$ can be made 0 by memorizing S . Example: $h(\mathbf{x}) = y_i$ if $\mathbf{x} = \mathbf{x}_i$, -1 otherwise: $\hat{R}_S(h) = 0$ and $R_{\mathcal{D}}(h) = \text{risk of the constant classifier}$.

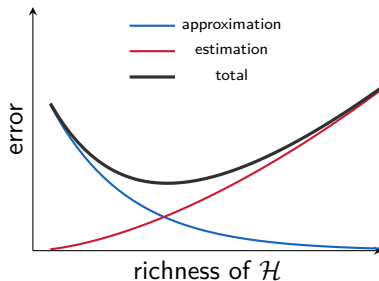
Need to control the **richness** of $\mathcal{H}$: choose it smaller, or **penalize** complex functions.

Approximation and estimation errors (§10.1)

Let $h_{\mathcal{H}}^* \in \arg \min_{h \in \mathcal{H}} R_{\mathcal{D}}(h)$ be the best model of the class and R^* the Bayes risk.

$$R_{\mathcal{D}}(h_S) - R^* = \underbrace{R_{\mathcal{D}}(h_S) - R_{\mathcal{D}}(h_{\mathcal{H}}^*)}_{\text{estimation error}} + \underbrace{R_{\mathcal{D}}(h_{\mathcal{H}}^*) - R^*}_{\text{approximation error}}.$$

- ▶ **Approximation:** is the class rich enough to contain good models? Decreases when $\mathcal{H}$ grows.
- ▶ **Estimation:** can we find the good one with n examples? Increases when $\mathcal{H}$ grows, decreases with n .
- ▶ (+ an **optimization** error if we do not find the exact minimizer.)



Structural risk minimization and penalization

Two equivalent ways to control complexity:

Constrained form

$$\min_{h \in \mathcal{H}} \hat{R}_S(h) \quad \text{s.t.} \quad \Omega(h) \leq \tau$$

(nested classes $\mathcal{H}_\tau$: structural risk minimization, Vapnik).

- ▶ $\Omega = \textbf{regularizer}$: norm of the weights, depth or number of leaves of a tree, number of features. . .
- ▶ Under convexity, for every τ there is a λ giving the same solution (Lagrangian duality).

Penalized form

$$\min_{h \in \mathcal{H}} \hat{R}_S(h) + \lambda \Omega(h)$$

$\lambda \geq 0$ trades data fit for simplicity.

Generalization bounds have exactly the form “*empirical risk + complexity*”: regularization is minimizing (a proxy of) a bound. PAC-Bayes makes this literal: minimize $\hat{R}_S(Q) + \text{KL}(Q \| P) / \lambda$ (Part 11).

Regularized empirical risk

For a linear score $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$:

$$\min_{\mathbf{w}, b} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\mathbf{w}}(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|.$$

Loss	Regularizer	Name
squared	$\ \mathbf{w}\ _2^2$	ridge regression
squared	$\ \mathbf{w}\ _1$	Lasso (sparse)
squared	$\alpha \ \mathbf{w}\ _1 + (1 - \alpha) \ \mathbf{w}\ _2^2$	elastic net
hinge	$\ \mathbf{w}\ _2^2$	SVM
logistic	$\ \mathbf{w}\ _2^2$ or $\ \mathbf{w}\ _1$	regularized logistic regression

The intercept b is usually not penalized. Features must be **standardized**: otherwise the penalty treats variables with different scales differently.

Ridge in one dimension: the shrinkage effect

No intercept, one feature: $\min_w \frac{1}{n} \sum_{i=1}^n (y_i - wx_i)^2 + \lambda w^2.$

Setting the derivative to zero:

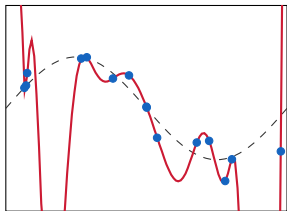
$$-\frac{2}{n} \sum_i x_i (y_i - wx_i) + 2\lambda w = 0 \implies \hat{w}_\lambda = \frac{\sum_i x_i y_i}{\underbrace{\sum_i x_i^2 + n\lambda}_{\in (0,1]}} \hat{w}_{\text{OLS}}.$$

- ▶ $\lambda = 0$: ordinary least squares.
- ▶ $\lambda > 0$: the estimate is **shrunk towards 0** by a factor that depends on how much information ($\sum x_i^2$) the data carry.
- ▶ With standardized x ($\frac{1}{n} \sum x_i^2 = 1$): factor $\frac{1}{1+\lambda}$.

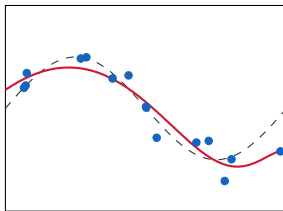
Shrinking introduces a small **bias** but reduces the **variance** of $\hat{w}$ (it is less sensitive to the noise). In d dimensions: $\hat{\mathbf{w}}_\lambda = (\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$, always invertible, even when $d > n$.

Polynomial of degree 14 fitted on 15 points, ridge penalty on the coefficients:

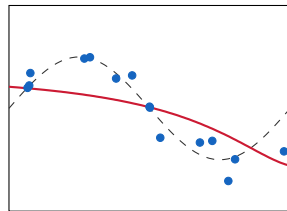
$\lambda = 0$: overfits



$\lambda = 10^{-4}$: good



$\lambda = 10^{-1}$: underfits



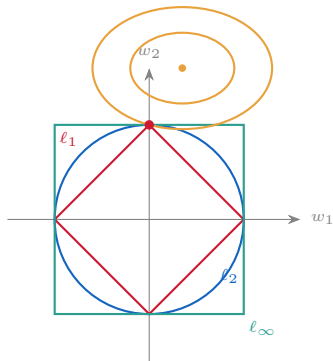
Same class of models (degree 14) in the three panels: only the penalty changes. Regularization lets us use a **rich class** while controlling its **effective complexity**. Dashed: the truth $\sin(2\pi x)$.

For $\mathbf{w} \in \mathbb{R}^d$ and $p \geq 1$:

$$\|\mathbf{w}\|_p = \left(\sum_{j=1}^d |w_j|^p \right)^{1/p}.$$

- ▶ $\|\mathbf{w}\|_2$: Euclidean norm;
- ▶ $\|\mathbf{w}\|_1 = \sum_j |w_j|$: favours sparsity;
- ▶ $\|\mathbf{w}\|_\infty = \max_j |w_j|$;
- ▶ “ $\|\mathbf{w}\|_0$ ” = $\#\{j : w_j \neq 0\}$ (not a norm, non-convex).

Why does ℓ_1 give sparse solutions? The ball has **corners** on the axes: the level sets of the loss often touch it there.

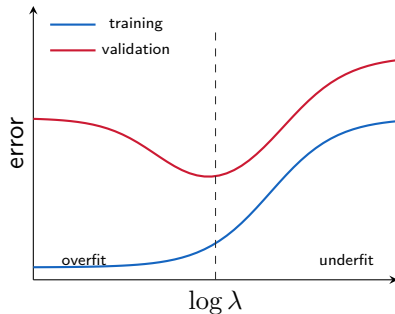


Unit balls; the level sets of the loss (orange) first touch the ℓ_1 ball at a corner: $w_1 = 0$.

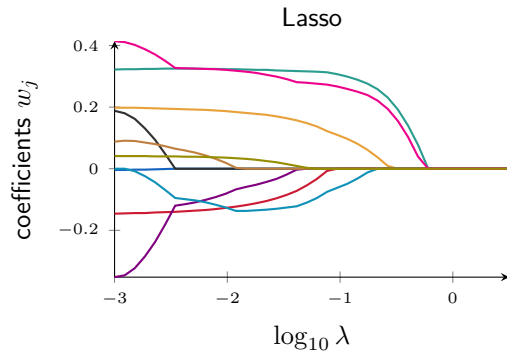
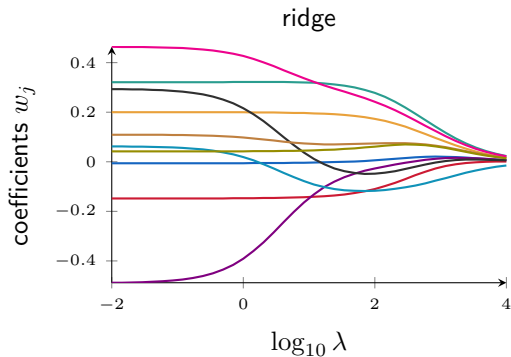
- ▶ $\lambda = 0$: pure ERM $\Rightarrow$ risk of **overfitting**.
- ▶ $\lambda \rightarrow \infty$: $\mathbf{w} \rightarrow 0$, constant model $\Rightarrow$ **underfitting**.
- ▶ In between: a **trade-off** between fit and simplicity.

λ is a hyperparameter

It is not learned by minimizing the training error (that would always give $\lambda = 0$). It is chosen on **held-out data**: validation set or cross-validation (Part 6).



Regularization paths (real data)



diabetes data ($n = 442$, $d = 10$ standardized features, one colour per feature). Ridge shrinks all coefficients smoothly; Lasso sets them **exactly to zero** one after the other $\Rightarrow$ variable selection. Read from right to left, the Lasso path adds one variable at a time – a cousin of **boosting** with tiny steps (forward stagewise regression, §10.3).

Other forms of regularization

- ▶ **Trees:** maximum depth, minimum number of examples per leaf, pruning (cost-complexity $\hat{R}_S(T) + \alpha|T|$).
- ▶ **k -NN:** larger k = smoother.
- ▶ **Kernels:** bandwidth γ of the Gaussian kernel.
- ▶ **Neural networks:** weight decay, dropout, early stopping, data augmentation.
- ▶ **Boosting:** number of iterations, learning rate (shrinkage), depth of the weak learners.
- ▶ **Bagging / random forests:** averaging is itself a regularization (it reduces variance).
- ▶ **Bayesian / PAC-Bayes:** a *prior* P on models; the penalty is $\text{KL}(Q\|P)$.

Every family of models has its “complexity knob”. Learning in practice = choosing the model **and** the right position of its knobs, with an honest protocol.

- ▶ **ERM**: minimize the empirical risk over a class $\mathcal{H}$. Works when $\hat{R}_S \approx R_D$ uniformly over $\mathcal{H}$.
- ▶ Error of $h_S =$ **approximation** (is $\mathcal{H}$ rich enough?) + **estimation** (can we find the good model with n examples?). Enriching $\mathcal{H}$ trades the first for the second.
- ▶ For a finite class: $R_D(h) \leq \hat{R}_S(h) + \sqrt{(\ln K + \ln \frac{1}{\delta})/2n}$ for all h simultaneously.
- ▶ **Regularization** = empirical risk + $\lambda \times$ complexity. λ is a hyperparameter chosen on held-out data. Ridge shrinks, Lasso selects.
- ▶ Every model has its complexity knobs; ensembles bring new ones (number of models, learning rate).

Part 6

Evaluating and selecting models

Two different questions

Model selection

Among several models / hyperparameters, **which one** should I use?

⇒ compare estimates of $R_{\mathcal{D}}$; only the *ranking* matters.

Model assessment

Once chosen, **how good** is it on new data?

⇒ an unbiased estimate (or a bound) of $R_{\mathcal{D}}(h)$ for the final model.

The fundamental rule

Data used to *train* or to *select* a model cannot be used to *assess* it: the estimate would be optimistic.

Hence the three sets: **train** (fit the parameters), **validation** (choose hyperparameters), **test** (final assessment, used *once*) – §7.1, §11.4.

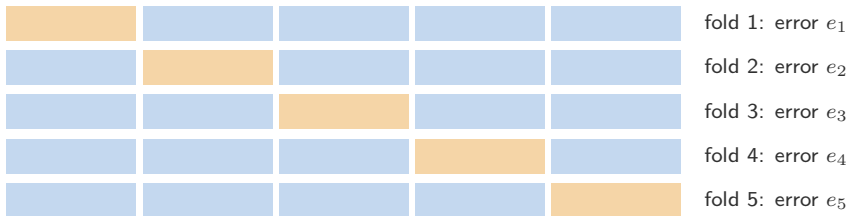


1. For each hyperparameter value θ : train h_θ on the training set, compute its validation error.
 2. Choose $\hat{\theta}$ with the smallest validation error.
 3. Re-train $h_{\hat{\theta}}$ on $\text{train} \cup \text{validation}$ (optional but usual).
 4. Report the error of the final model on the test set **once**.
- ▶ Simple, cheap.
 - ▶ Stratify (same class proportions) and shuffle (unless temporal data).

Drawbacks

Wastes data; estimates depend a lot on the split (high variance) when n is small.

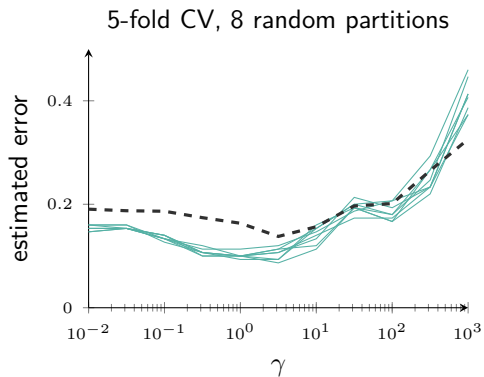
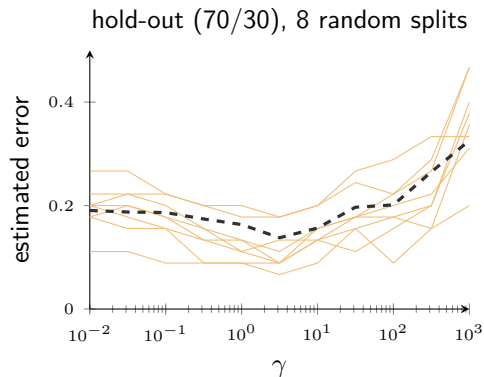
k -fold cross-validation



- ▶ Split S into k folds of equal size. For $j = 1..k$: train on all folds but j , evaluate on fold j .
- ▶ CV estimate: $\hat{R}_{CV} = \frac{1}{k} \sum_{j=1}^k e_j$. Usual choices: $k = 5$ or 10 .
- ▶ Every example is used once for validation and $k - 1$ times for training.

TP1: over 20 random splits, the score of a configuration varies with std **0.024** with a hold-out split and only **0.005** with 5-fold CV. CV gives much more stable model selection.

Hold-out vs. cross-validation: a picture



Gaussian SVM ($C = 10$) on one sample of 150 points of the running example; dashed = true error of the model trained on the 150 points. Each coloured curve is what one experimenter would see. Average standard deviation across the 8 curves: **0.047** for hold-out, **0.012** for CV. With hold-out, the chosen γ (minimum of the curve) is largely a matter of luck.

Leave-one-out (LOO)

$k = n$: train on $n - 1$ examples, test on the remaining one, n times.

$$\hat{R}_{LOO} = \frac{1}{n} \sum_{i=1}^n \ell(h_{S \setminus \{i\}}(\mathbf{x}_i), y_i).$$

Almost unbiased (trained on $n - 1$ points), but high variance and n trainings. Closed form for least squares, k -NN.

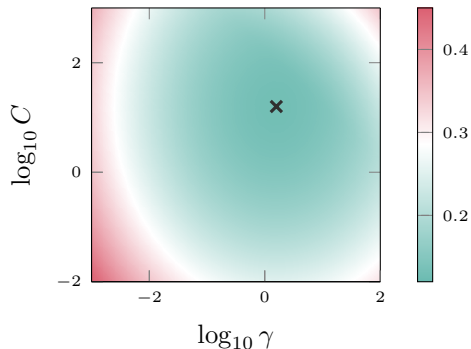
- ▶ Small k : models trained on fewer data $\Rightarrow$ pessimistic bias.
- ▶ Large k : more expensive, folds overlap a lot.
- ▶ $k = 5$ or 10 is a good compromise.
- ▶ Repeated CV: average over several random partitions.

Out-of-bag error (bagging, TP2): each model is evaluated on the examples it did not see – a free “CV” that comes with bootstrap.

Cross-validation to tune hyperparameters

1. Define a grid Θ (e.g.
 $C \in \{10^{-2}, \dots, 10^3\}$,
 $\gamma \in \{10^{-3}, \dots, 10^2\}$, log scale!).
2. For each $\theta \in \Theta$ compute $\hat{R}_{CV}(\theta)$ on the *training set*.
3. $\hat{\theta} = \arg \min_{\theta} \hat{R}_{CV}(\theta)$; re-train on the whole training set.
4. Evaluate once on the test set.

Random search is often more efficient than a full grid when only a few hyperparameters matter.



CV error of a Gaussian SVM on a (γ, C) grid
(schematic).

- ▶ $\hat{R}_{CV}(\theta)$ is a nearly unbiased estimate for a *fixed* θ .
- ▶ But $\min_{\theta} \hat{R}_{CV}(\theta)$ is **biased downwards**: we selected the configuration that was lucky on these folds.
- ▶ The more configurations, the stronger the effect (same phenomenon as the union bound, Part 9).

TP1, random labels (true error 0.5 for every model): best CV score **0.473**, test error 0.492.

Nested cross-validation

To assess a *procedure* “tune by CV + train”:

- ▶ outer loop: k folds for assessment;
- ▶ inner loop: CV on each outer training part to choose θ .

Cost: $k_{\text{out}} \times k_{\text{in}} \times |\Theta|$ trainings.

Never report the best CV score as the performance of your model.

Definition – Leakage

Information from the validation/test data (or from the future) is used, directly or not, to build the model.

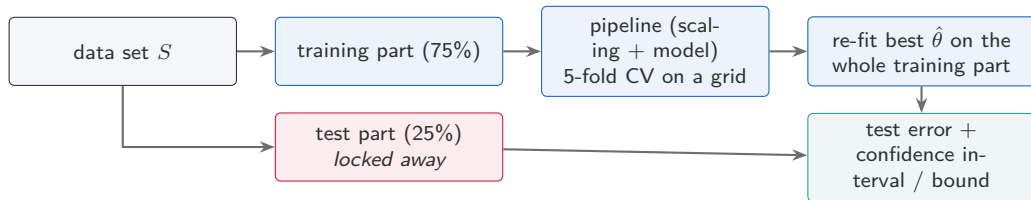
Classical leaks

- ▶ scaling/imputation fitted on all data;
- ▶ **feature selection on all data, then CV**;
- ▶ duplicates or same patient in train and test;
- ▶ random split of time series;
- ▶ a feature that is a consequence of the label (“treatment received”).

TP1: 2000 pure-noise features, 50 examples. Select the 20 features most correlated with y *on all data*, then CV: accuracy **0.852**. Selection inside the CV loop (pipeline): **0.529** – the truth (chance level).

Remedy: put every data-dependent step in a Pipeline; split **first**.

The complete protocol (TP1)



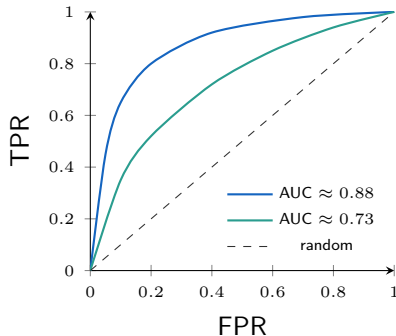
test error	wdbc	pima	australian
Gaussian SVM	0.030	0.240	0.146
decision tree	0.070	0.256	0.154

Every number of the TPs that follow is obtained with this protocol. Ensembles will improve the tree column a lot.

Performance measures beyond accuracy (§7.2)

	$y = +1$	$y = -1$
$h = +1$	TP	FP
$h = -1$	FN	TN

- ▶ accuracy = $\frac{TP+TN}{n}$, error = 1−accuracy;
- ▶ precision = $\frac{TP}{TP+FP}$;
- ▶ recall (sensitivity, TPR) = $\frac{TP}{TP+FN}$;
- ▶ specificity = $\frac{TN}{TN+FP}$;
- ▶ $F_1 = \frac{2 \cdot \text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$;
- ▶ balanced accuracy = $\frac{\text{TPR} + \text{TNR}}{2}$.



ROC curve: all thresholds of a score f . AUC = probability that a random positive gets a higher score than a random negative.

A worked example: screening a disease

1 000 patients, 50 are sick. A classifier gives:

	sick	healthy	
predicted sick	40	90	130
predicted healthy	10	860	870
	50	950	1 000

accuracy	$(40 + 860)/1000$	0.90
recall (sensitivity)	$40/50$	0.80
specificity	$860/950$	0.905
precision	$40/130$	0.31
F_1	$\frac{2 \cdot 0.31 \cdot 0.8}{0.31 + 0.8}$	0.44
balanced accuracy	$(0.80 + 0.905)/2$	0.85

“90% accuracy” sounds great, but the trivial rule “everybody is healthy” has **95%** accuracy! Here **7 out of 10** positive predictions are false alarms. Choose the metric according to the costs of each type of error, and look at the whole confusion matrix.

Regression

- ▶ $\text{MSE} = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$, $\text{RMSE} = \sqrt{\text{MSE}}$ (unit of y);
- ▶ $\text{MAE} = \frac{1}{n} \sum |y_i - \hat{y}_i|$ (robust);
- ▶ $R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$ (fraction of variance explained).

Probabilistic predictions

- ▶ log-loss $-\frac{1}{n} \sum \ln \hat{p}(y_i | \mathbf{x}_i)$;
- ▶ Brier score $\frac{1}{n} \sum (\hat{p}_i - \mathbb{1}[y_i = 1])^2$;
- ▶ calibration: among examples with $\hat{p} \approx 0.8$, are 80% positive?

Random forests tend to be *under-confident*, boosting *over-confident*: their scores often need re-calibration (Platt scaling, isotonic regression) when probabilities matter.

- ▶ Is 0.146 vs. 0.154 a real difference? On a test set of $m = 172$ examples, Hoeffding gives a width ≈ 0.10 at 95%...
- ▶ Use **paired** comparisons on the same test examples: McNemar's test on the disagreements.
- ▶ Across folds: paired t -test (with care: folds are not independent).
- ▶ Across data sets: Wilcoxon signed-rank test, critical-difference diagrams (Demšar, 2006).

McNemar

n_{01} = examples where only h_1 errs, n_{10} = only h_2 errs. Under H_0 (same error), $n_{01} \sim \mathcal{B}(n_{01} + n_{10}, 1/2)$.

What matters is where the models **disagree**. The disagreement of classifiers is also a key quantity for majority votes (Part 11).

A checklist for a sound experiment

1. Define the metric *before* looking at the results.
2. Split first (stratified / grouped / temporal).
3. Put all preprocessing in a pipeline.
4. Compare with simple baselines (majority class, logistic regression).
5. Tune with CV on the training part only.
6. Test once; report an interval or a bound, not only a point estimate.
7. Repeat over several seeds / splits; report mean $\pm$ std.
8. Account for the number of models tried.
9. Fix the random seeds; share the code.

The protocol gives an **empirical** answer to “how good is my model?”. Part 10 gives **theoretical** answers that do not need a test set. . . which is useful when data are scarce, and is the motivation for PAC-Bayes.

- ▶ **Selection** (which model?) and **assessment** (how good?) need different data: train / validation / test.
- ▶ **Cross-validation** gives much more stable estimates than a single hold-out split; LOO and OOB are variants.
- ▶ The **best CV score is optimistic**: use nested CV or an untouched test set for the final number.
- ▶ **Leakage** is the most common way to fool yourself: split first, put every preprocessing in a pipeline.
- ▶ Accuracy is not everything: confusion matrix, precision/recall, ROC/AUC, calibration.
- ▶ Report an **interval or a bound**, not only a point estimate.

Part 7

The main learning algorithms

Map of the models we need

Model	Type of h	Complexity knob	Scaling?	Role later
k -NN (§8.2)	local average	k	yes	baseline, stable
Linear / logistic (§8.4–8.5)	$\text{sign}\langle \mathbf{w}, \mathbf{x} \rangle$	λ	yes	stacking, stable
SVM / kernels (§8.3, 8.6)	$\sum_i \alpha_i y_i K(\mathbf{x}_i, \cdot)$	C, γ	yes	strong baseline
Decision trees (§8.7)	piecewise constant	depth, leaves	no	base learner
Stumps	one split	–	no	boosting
Neural nets (§9)	compositions	width, epochs	yes	–

For each model we look at: the **prediction rule**, the **learning rule** (which loss, which regularization), and its **stability** – whether small changes of S change h_S a lot. Unstable models are the ones that benefit from **bagging**; weak but cheap models are the ones **boosting** likes.

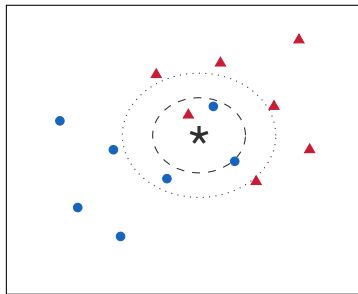
k nearest neighbours (§8.2)

Prediction. Let $N_k(\mathbf{x})$ be the k training points closest to $\mathbf{x}$.

$$h(\mathbf{x}) = \text{sign} \left(\sum_{i \in N_k(\mathbf{x})} y_i \right) \quad (\text{majority vote}),$$

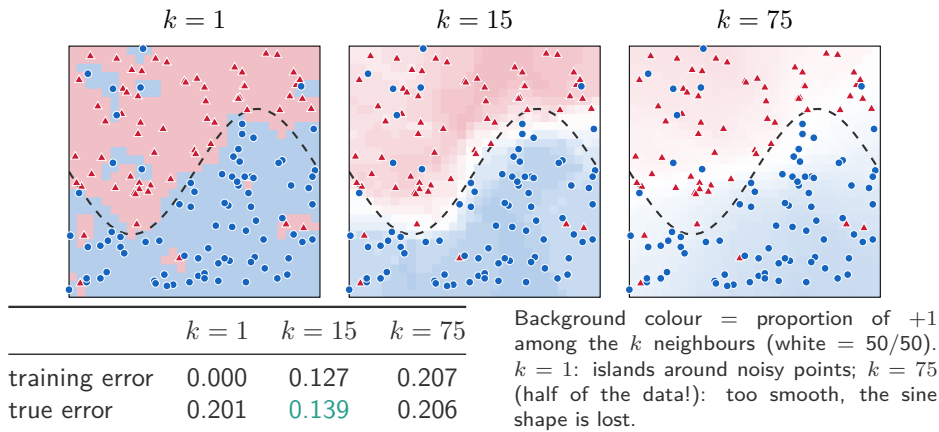
or the average $\frac{1}{k} \sum_{i \in N_k(\mathbf{x})} y_i$ in regression.

- ▶ No training (“lazy”), but prediction costs $O(nd)$.
- ▶ Needs a **meaningful distance** $\Rightarrow$ normalize!
- ▶ $k = 1$: $\hat{R}_S = 0$, high variance. $k = n$: constant, high bias.
- ▶ Consistent if $k \rightarrow \infty$ and $k/n \rightarrow 0$ (Stone, 1977).

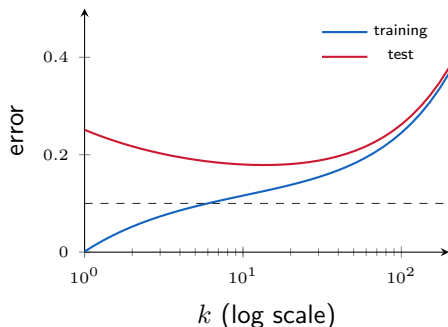


$k = 3$ (dashed): 2 triangles vs. 1 circle. $k = 7$ (dotted): 4 circles vs. 3 triangles. The prediction changes with k .

k -NN on the running example



k -NN: effect of k and a nice guarantee



Cover & Hart (1967)

As $n \rightarrow \infty$, the risk of the 1-NN rule satisfies

$$R^* \leq R_{1\text{-NN}} \leq 2R^*(1 - R^*).$$

“Half of the information is in the nearest neighbour.”

k -NN is itself a **vote**: each neighbour votes for its label. With weights decreasing with the distance, it is a kernel-weighted vote – a first “ensemble” of very simple voters.

Model: $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. Least squares:

$$\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \quad \hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

(adding a column of 1's to $\mathbf{X}$ for b).

- ▶ Geometric view: $\mathbf{X}\hat{\mathbf{w}}$ = orthogonal projection of $\mathbf{y}$ on the span of the columns.
- ▶ Statistical view: maximum likelihood if $y = \langle \mathbf{w}^*, \mathbf{x} \rangle + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.
- ▶ Ill-posed if $d > n$ or collinear features $\Rightarrow$ ridge.

Non-linear features

Linear in the parameters, not necessarily in $\mathbf{x}$:

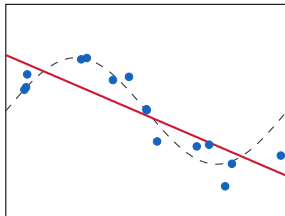
$$f(x) = w_0 + w_1x + w_2x^2 + \dots + w_px^p.$$

Polynomial degree p = complexity knob.

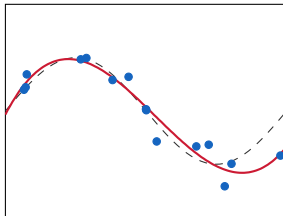
Linear models are **stable** (low variance): bagging them is useless (TP2). They are the usual *meta-learner* of **stacking**.

Polynomial regression: under- and overfitting

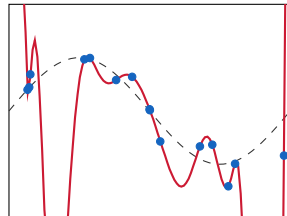
degree 1: underfit



degree 3: good



degree 14: overfit



15 noisy points around $\mathbb{E}[y|x] = \sin(2\pi x)$ (dashed), least-squares fits. Degree 14 = 15 coefficients for 15 points: the curve passes through every point (training error 0) and *interpolates the noise*, with huge oscillations between them.

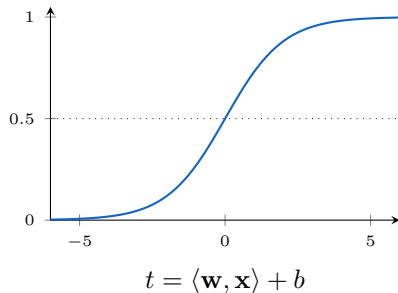
Model the probability

$$\mathbb{P}(y = +1 \mid \mathbf{x}) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b) \text{ with } \sigma(t) = \frac{1}{1+e^{-t}}.$$

Maximum likelihood $\Leftrightarrow$ minimizing the logistic loss:

$$\min_{\mathbf{w}, b} \frac{1}{n} \sum_{i=1}^n \ln(1 + e^{-y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)}) + \lambda \|\mathbf{w}\|_2^2.$$

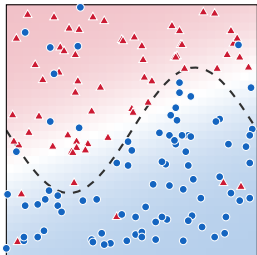
- ▶ Convex, smooth: gradient descent / Newton.
- ▶ No closed form; without λ , $\|\mathbf{w}\| \rightarrow \infty$ if the data are separable.
- ▶ Linear decision boundary $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$.



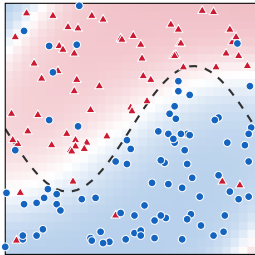
Gives **calibrated scores** (probabilities): useful to combine models, to change the threshold, and as a stacking meta-model.

Logistic regression on the running example

features (x_1, x_2)



all monomials of degree ≤ 3



- ▶ Background:
 $2\mathbb{P}(y = +1|\mathbf{x}) - 1$, the model gives **probabilities**.
- ▶ Linear features: a straight frontier, true error 0.184 (**bias**).
- ▶ Polynomial features (9 features): curved frontier, 0.182.

A linear model in well-chosen features can be very flexible: *feature engineering* moves the model along the bias–variance axis.

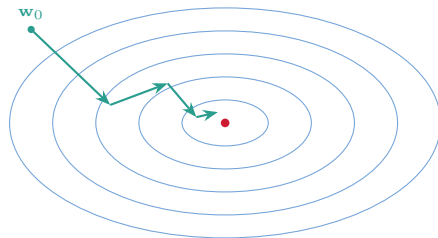
Gradient descent in one slide

To minimize a differentiable $F(\mathbf{w})$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla F(\mathbf{w}_t).$$

- ▶ η_t : *learning rate* (step size).
- ▶ Convex F with Lipschitz gradient:
converges to the minimum for η small enough.
- ▶ **Stochastic** GD: replace $\nabla \hat{R}_S$ by the gradient on one example or a mini-batch – the workhorse of deep learning.

Gradient boosting (§10.3) = gradient descent *in function space*: each step adds a tree fitted to the negative gradient.



Zig-zag of gradient descent on a badly conditioned quadratic: one more reason to standardize features.

K classes, one weight vector per class:

$$\mathbb{P}(y = k \mid \mathbf{x}) = \frac{e^{\langle \mathbf{w}_k, \mathbf{x} \rangle}}{\sum_{j=1}^K e^{\langle \mathbf{w}_j, \mathbf{x} \rangle}} \quad (\text{softmax}).$$

Loss: cross-entropy $-\ln \mathbb{P}(y_i \mid \mathbf{x}_i)$.

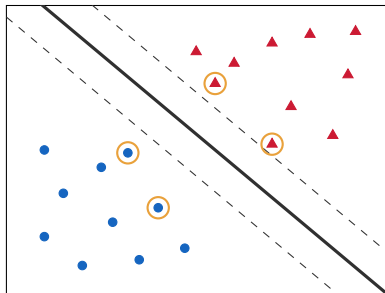
Prediction: $\arg \max_k \langle \mathbf{w}_k, \mathbf{x} \rangle$.

Reductions to binary problems

- ▶ *One-vs-rest*: K classifiers “ k vs. others”, predict the most confident.
- ▶ *One-vs-one*: $K(K-1)/2$ classifiers, **majority vote**.
- ▶ *Error-correcting output codes*.

One-vs-one is already an **ensemble**: many binary classifiers combined by a vote. Trees, k -NN and forests handle multi-class natively.

Support vector machines: the maximal margin idea (§8.3)



Separable data: among all separating hyperplanes, choose the one with the **largest margin**:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall i.$$

The margin is $1/\|\mathbf{w}\|$. Only the points on the margin (**circled**) matter: the **support vectors**.

Large margin = robustness. Margin-based bounds do not depend on d , only on $\|\mathbf{w}\|$ and the radius of the data.

Soft-margin SVM and the hinge loss

When the data are not separable, allow violations $\xi_i \geq 0$:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Equivalently (at the optimum $\xi_i = \max(0, 1 - y_i f(\mathbf{x}_i))$):

$$\min_{\mathbf{w}, b} \frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i f_{\mathbf{w}}(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|^2,$$

with $\lambda = \frac{1}{2nC}$: **hinge loss** + ℓ_2 **regularization**.

- ▶ C large: few violations, risk of overfitting.
- ▶ C small: wide margin, more violations, smoother.
- ▶ Convex quadratic program, unique solution.

The SVM is regularized ERM with a convex surrogate: exactly the template of Part 5.

The dual problem only involves **inner products**:

$$\max_{\alpha \in [0, C]^n, \sum_i \alpha_i y_i = 0} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

and $f(\mathbf{x}) = \sum_i \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b$.

Kernel trick: replace $\langle \mathbf{x}, \mathbf{x}' \rangle$ by $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ for a feature map ϕ we never compute.

Usual kernels

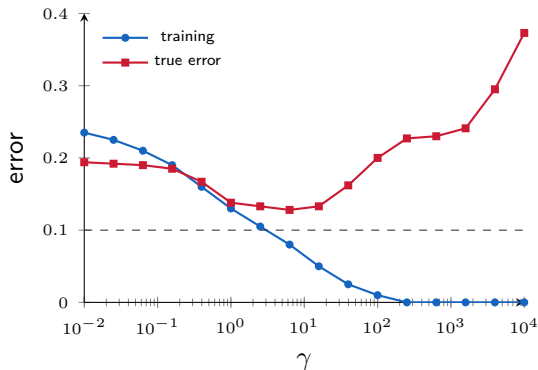
Linear: $\langle \mathbf{x}, \mathbf{x}' \rangle$

Polynomial: $(\langle \mathbf{x}, \mathbf{x}' \rangle + c)^p$

Gaussian (RBF): $e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2}$

Gaussian kernel: ϕ is infinite-dimensional. γ large = very local, wiggly frontier; γ small = almost linear.

Gaussian SVM: the effect of γ (TP1)

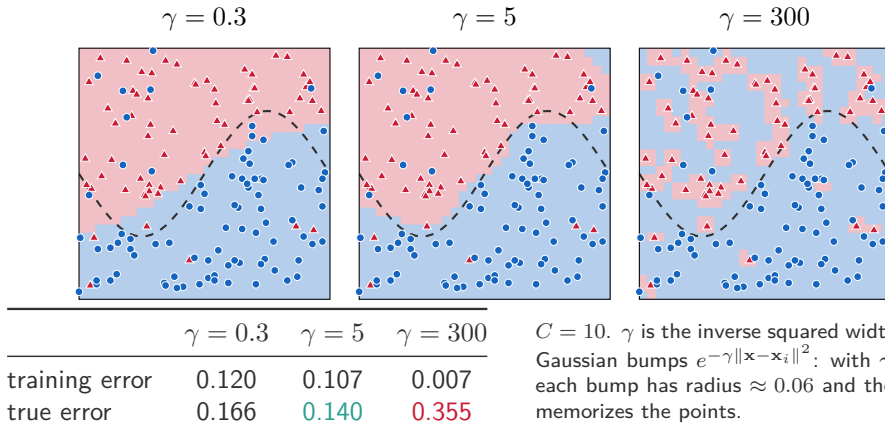


Toy problem of TP1 ($R^* = 0.1$),
 $C = 10$, γ on a log grid.

- ▶ Small γ : almost linear, underfits.
- ▶ $\gamma \approx 6$: best true error 0.128.
- ▶ Large γ : each point is its own island: training error 0, true error 0.37.

Same U-shaped curve as for the depth of trees, k in k -NN, $\lambda \dots$: the signature of the bias–variance trade-off (Part 4).

Gaussian SVM: what the frontier looks like



Nadaraya–Watson

A local weighted average:

$$f(\mathbf{x}) = \frac{\sum_i K_h(\mathbf{x}, \mathbf{x}_i) y_i}{\sum_i K_h(\mathbf{x}, \mathbf{x}_i)}.$$

Bandwidth h : small = wiggly, large = flat.

Kernel ridge regression

Ridge in feature space:

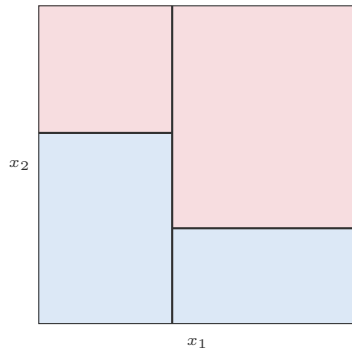
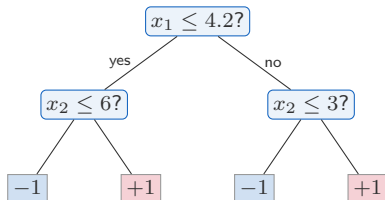
$$f(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}),$$

$$\boldsymbol{\alpha} = (\mathbf{K} + n\lambda I)^{-1} \mathbf{y}.$$

Representer theorem: the solution lies in $\text{span}\{K(\mathbf{x}_i, \cdot)\}$.

Both are **weighted combinations of n simple functions** centred on the data. Ensembles are also weighted combinations of simple functions – but the functions themselves are *learned* (trees).

Decision trees (CART, §8.7)



- ▶ A tree = recursive partition of $\mathcal{X}$ into axis-parallel boxes; one constant prediction per leaf (majority class, or mean of y).
- ▶ Interpretable, handles mixed types and missing values, invariant to monotone transformations (no scaling needed).

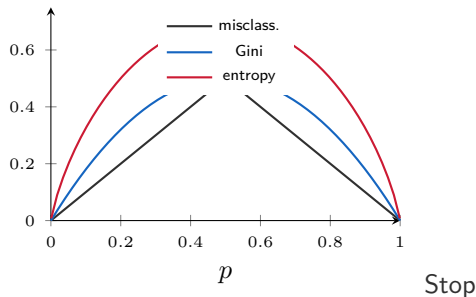
Growing a tree: greedy splits

At each node with examples S_t , try all features j and thresholds s , and choose the split maximizing the **decrease of impurity**:

$$\Delta = I(S_t) - \frac{|S_L|}{|S_t|}I(S_L) - \frac{|S_R|}{|S_t|}I(S_R).$$

Impurity measures (p = proportion of class +1):

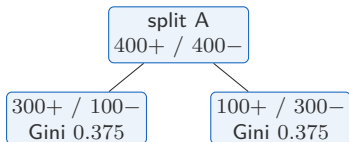
- ▶ misclassification $\min(p, 1 - p)$;
- ▶ Gini $2p(1 - p)$;
- ▶ entropy $-p \ln p - (1 - p) \ln(1 - p)$;
- ▶ regression: variance of y in the node.



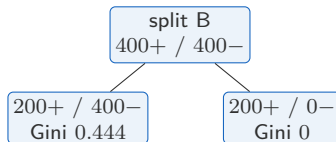
when: max depth reached, too few examples, or no impurity decrease. Then possibly **prune**.

Choosing a split: a worked example

A node contains **400 positives and 400 negatives**: $\text{Gini} = 2 \times \frac{1}{2} \times \frac{1}{2} = 0.5$. Two candidate splits:



$$\Delta_A = 0.5 - \left(\frac{1}{2} 0.375 + \frac{1}{2} 0.375 \right) = 0.125$$

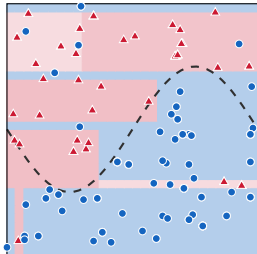


$$\Delta_B = 0.5 - \left(\frac{3}{4} 0.444 + \frac{1}{4} 0 \right) = \mathbf{0.167}$$

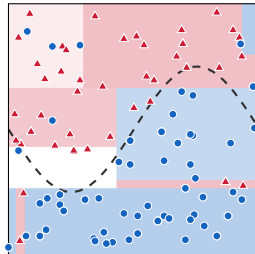
With the **misclassification rate**, both splits look equivalent ($0.5 \rightarrow 0.25$ errors). **Gini** prefers B, which creates a **pure** leaf that will not need to be split again. Gini and entropy are strictly concave: they reward purity, which makes greedy growth work better.

Trees are unstable

bootstrap sample 1



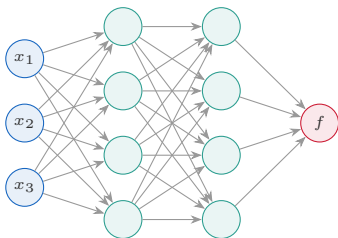
bootstrap sample 2



Same data resampled with replacement, same algorithm (depth 4): two **very different** trees.

- ▶ A small change in S changes the first splits, hence the whole tree.
- ▶ Deep trees: low bias, **high variance**.
- ▶ Stumps: high bias, low variance.

A problem for interpretation... and an **opportunity**: different trees make different errors, so their vote can be much better than each of them (TP1: one tree 0.182–0.194, vote of 50 bootstrapped trees 0.137).



$$f(\mathbf{x}) = W_L \sigma(\cdots \sigma(W_1 \mathbf{x} + b_1) \cdots) + b_L$$

- ▶ σ : non-linearity (ReLU $\max(0, t)$, sigmoid...).
- ▶ Universal approximation: one hidden layer with enough units approximates any continuous function.
- ▶ Trained by SGD on a surrogate loss; gradients by back-propagation.

Why they matter here

- ▶ The hidden layers **learn the representation** (Part 1).
- ▶ Huge models ($\gg n$ parameters) that still generalize: classical complexity bounds are vacuous for them.
- ▶ Non-vacuous bounds for deep nets were first obtained with **PAC-Bayes** (Dziugaite & Roy, 2017).

Ensembles of networks

- ▶ Deep ensembles: train the same network with M random initializations and average $\Rightarrow$ better accuracy and calibration.
- ▶ Dropout can be seen as training an implicit ensemble of sub-networks.

On **tabular** data, gradient-boosted trees remain hard to beat (Grinsztajn et al., 2022).

Which model when? A practitioner's summary

	k -NN	logistic	SVM (RBF)	tree	forest / boosting
needs scaling	yes	yes	yes	no	no
non-linear	yes	no*	yes	yes	yes
interpretable	$\sim$	yes	no	yes	$\sim$
large n	slow	fast	slow (n^2-n^3)	fast	fast
variance	medium	low	medium	high	low
bias	low-med.	high	low	low (deep)	low

* unless features are transformed.

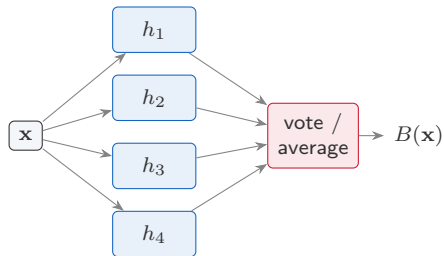
No model is best everywhere (*no free lunch*). The forests/boosting column explains why ensembles of trees are the default choice on tabular data: they keep the low bias of deep trees and remove most of their variance.

- ▶ **k -NN**: local vote; k controls smoothness; needs scaling; suffers in high dimension.
- ▶ **Linear / logistic models**: stable, interpretable, fast; flexible through feature engineering; give probabilities.
- ▶ **SVM**: hinge loss + ℓ_2 penalty, maximal margin; kernels give non-linear frontiers; C and γ to tune.
- ▶ **Trees**: greedy recursive splits, no scaling, interpretable, but **unstable** (high variance).
- ▶ Every model has the same U-shaped behaviour with respect to its complexity knob.
- ▶ Unstable, low-bias models (deep trees) are the ideal **base learners for bagging**; weak, cheap models (stumps, small trees) are the ideal **base learners for boosting**.

Part 8

Ensemble methods

- ▶ Galton (1907): 787 people guess the weight of an ox; the median guess is within 1% of the truth.
- ▶ Condorcet (1785): a jury of independent members, each right with probability $> 1/2$, is right with a probability that tends to 1 when the jury grows.



Ensemble learning: build $h_1, \dots, h_M$ and combine them. Works when the members are **accurate** (better than chance) and **diverse** (make different errors).

Condorcet's jury theorem (TP5)

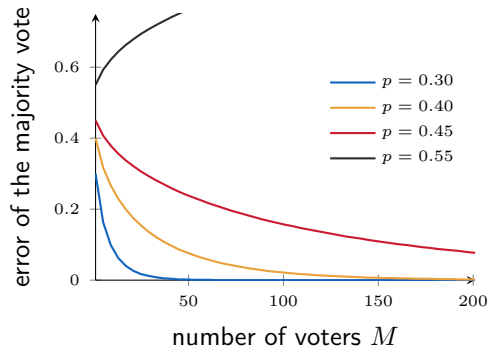
M independent voters, each wrong with probability p . The majority is wrong iff more than $M/2$ voters are wrong:

$$R_{\mathcal{D}}(B) = \mathbb{P}(\text{Bin}(M, p) > M/2).$$

- ▶ $p < \frac{1}{2}$: $R_{\mathcal{D}}(B) \rightarrow 0$ exponentially fast (Hoeffding!): $R_{\mathcal{D}}(B) \leq e^{-2M(\frac{1}{2}-p)^2}$.
- ▶ $p > \frac{1}{2}$: the vote makes things **worse**.

The catch

Independence. TP5: voters with error 0.3 and correlation 0.3: the vote error stalls at 0.169 instead of $\rightarrow 0$.



Voters $h \in \mathcal{H}$ with outputs in $\{-1, +1\}$, a distribution (weights) Q on $\mathcal{H}$.

- ▶ **Majority vote:** $B_Q(\mathbf{x}) = \text{sign}(\mathbb{E}_{h \sim Q} h(\mathbf{x}))$.
- ▶ **Gibbs classifier** G_Q : draw $h \sim Q$, predict $h(\mathbf{x})$. $R_{\mathcal{D}}(G_Q) = \mathbb{E}_{h \sim Q} R_{\mathcal{D}}(h)$.
- ▶ **Margin** of the vote:
 $M_Q(\mathbf{x}, y) = y \mathbb{E}_{h \sim Q} h(\mathbf{x}) \in [-1, 1]$. B_Q errs
iff $M_Q \leq 0$.

Bagging, random forests: Q uniform on M trees.

AdaBoost: $Q(h_t) \propto \alpha_t$.

PAC-Bayes: Q is a **posterior** distribution, compared with a **prior** P .

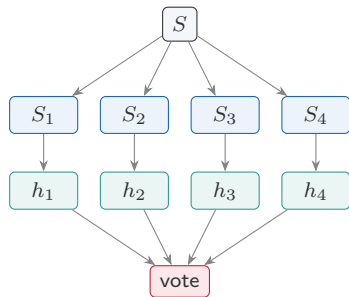
Two simple identities

$$R_{\mathcal{D}}(G_Q) = \frac{1}{2} (1 - \mathbb{E}_{\mathcal{D}}[M_Q])$$

$$R_{\mathcal{D}}(B_Q) = \mathbb{P}_{\mathcal{D}}(M_Q \leq 0)$$

Bootstrap aggregating:

1. for $m = 1..M$: draw a bootstrap sample S_m (n examples with replacement from S);
 2. learn $h_m = A(S_m)$ (typically a deep tree);
 3. predict by majority vote (classification) or average (regression).
- ▶ Each S_m contains $\approx 63\%$ of the distinct examples; the others are **out-of-bag** $\Rightarrow$ free error estimate.
 - ▶ Reduces **variance**; useful for unstable learners only.
 - ▶ Embarrassingly parallel.



TP2: bagging trees divides their variance; bagging a logistic regression changes nothing.

The bootstrap on a small example

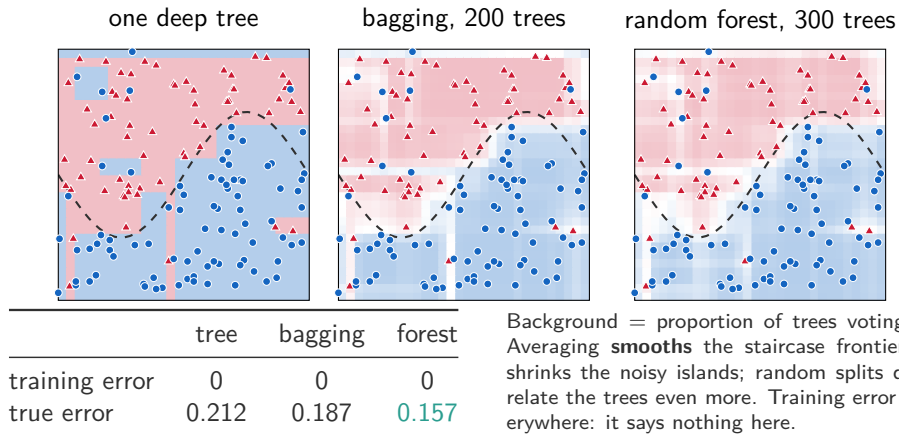
$S = \{1, 2, 3, 4, 5, 6, 7, 8\}$ (indices of the examples). Three bootstrap samples (8 draws with replacement):

	sample S_m	out-of-bag
S_1	2, 2, 3, 5, 5, 5, 7, 8	1, 4, 6
S_2	1, 1, 2, 4, 6, 6, 7, 8	3, 5
S_3	1, 3, 3, 4, 4, 5, 6, 8	2, 7

- ▶ Some examples appear several times, others not at all.
- ▶ $\mathbb{P}(\text{example } i \notin S_m) = (1 - \frac{1}{n})^n \rightarrow e^{-1} \approx 0.368$.
- ▶ Each tree sees $\approx 63\%$ of the distinct examples $\Rightarrow$ the trees differ.
- ▶ **OOB prediction** of example i : vote of the trees for which i is out-of-bag (e.g. example 3: tree 2 only).

The OOB error is an almost free estimate of the test error of the forest – and an **independent sample** for each tree, which TP5 uses to certify a forest with PAC-Bayes.

Bagging and random forests on the running example



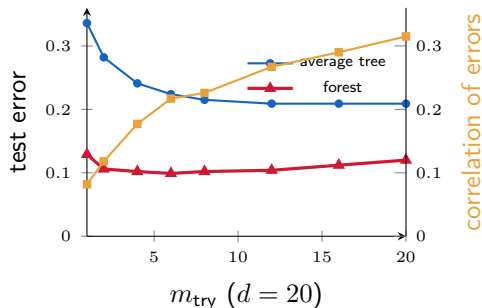
Bagging of trees + **random feature**

subsets: at each split, only m_{try} features drawn at random are candidates (default $\sqrt{d}$ in classification).

- ▶ Decorrelates the trees ($\rho \downarrow$) at the price of slightly weaker trees.
- ▶ Almost no tuning; robust; OOB error; variable importance.

Breiman's bound

$R_{\mathcal{D}}(\text{forest}) \leq \bar{\rho} \frac{1-s^2}{s^2}$, with s the expected margin (strength) and $\bar{\rho}$ the mean correlation.



Synthetic data, $d = 20$, $n = 600$, 150 trees. Trees improve with m_{try} but become correlated; the forest is best for $m_{\text{try}} \approx 4-8$.

Boosting: AdaBoost (Freund & Schapire, 1997)

Weights $w_i^{(1)} = \frac{1}{n}$. For $t = 1..T$:

1. learn a weak classifier h_t (stump) on the weighted sample; weighted error ε_t ;
2. $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$;
3. $w_i^{(t+1)} \propto w_i^{(t)} \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$:
misclassified examples get more weight.

Output: $\text{sign} \left(\sum_t \alpha_t h_t(\mathbf{x}) \right)$.

Training error (§10.2)

With $\varepsilon_t = \frac{1}{2} - \gamma_t$: $\hat{R}_S \leq \prod_t 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \leq e^{-2 \sum_t \gamma_t^2}$.

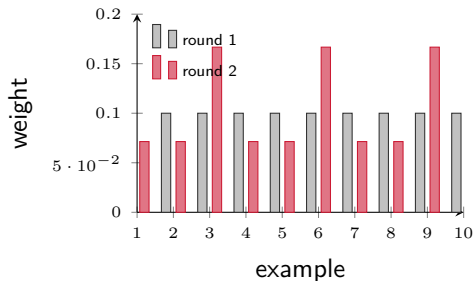
- ▶ Sequential: each model corrects its predecessors.
- ▶ Reduces **bias** (weak learners) and, in practice, variance.
- ▶ = greedy minimization of the **exponential loss** $\frac{1}{n} \sum_i e^{-y_i f(\mathbf{x}_i)}$.
- ▶ Sensitive to label noise (exponential loss!).

TP3: AdaBoost from scratch, three short proofs, margins and label noise.

One round of AdaBoost, by hand

10 examples, weights $w_i = 0.1$. The best stump misclassifies **3** of them:

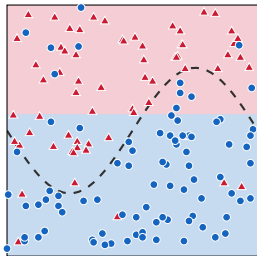
1. $\varepsilon_1 = 3 \times 0.1 = 0.3$.
2. $\alpha_1 = \frac{1}{2} \ln \frac{0.7}{0.3} = 0.424$.
3. Correct examples: $0.1 e^{-0.424} = 0.0655$;
wrong examples: $0.1 e^{0.424} = 0.153$.
4. Normalize (sum
 $= 7 \times 0.0655 + 3 \times 0.153 = 0.917$):
correct $\rightarrow 1/14 \approx 0.071$, wrong
 $\rightarrow 1/6 \approx 0.167$.



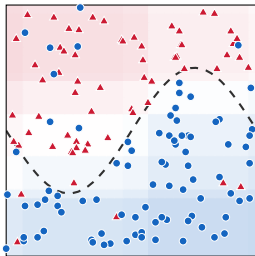
After the update, the wrong examples carry **half** of the total weight ($3/6$): h_1 has weighted error exactly $\frac{1}{2}$, so round 2 must find something *new*.

AdaBoost on the running example

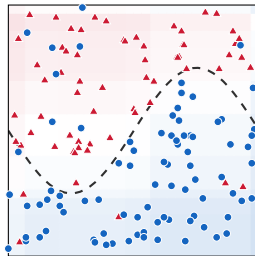
$T = 1$ stump



$T = 50$ stumps



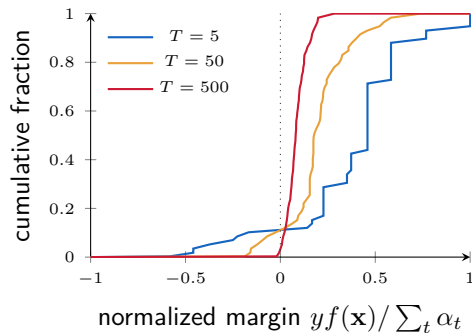
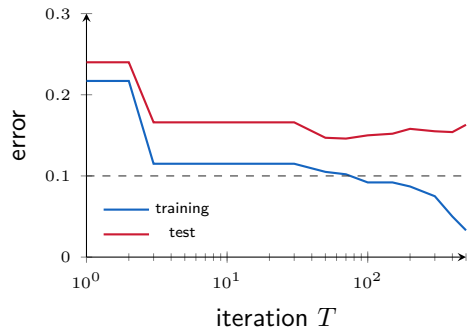
$T = 300$ stumps



	$T = 1$	$T = 50$	$T = 300$
training error	0.207	0.107	0.107
true error	0.235	0.149	0.167

Each stump cuts along one axis; their weighted sum builds a staircase that follows the sine. Learning rate 0.3. A single stump is very biased (0.235); 50 of them get close to the forest – from **weak** learners.

AdaBoost: errors and margins along the iterations



Running example, $n = 400$, trees of depth 2. Left: the training error keeps decreasing, the test error stabilizes. Right: distribution of the training margins; the value of each curve at 0 is the training error. Boosting **pushes the margins to the right** – the margin theory of Part 10.

Gradient boosting (Friedman, 2001; §10.3)

Minimize $\sum_i \ell(y_i, F(\mathbf{x}_i))$ over additive models $F = \sum_t \nu f_t$:

1. $F_0 = \text{constant}$;
2. compute pseudo-residuals
$$r_i = -\frac{\partial \ell(y_i, F)}{\partial F} \Big|_{F=F_{t-1}(\mathbf{x}_i)};$$
3. fit a small regression tree f_t to $(\mathbf{x}_i, r_i)$;
4. $F_t = F_{t-1} + \nu f_t$ (learning rate $\nu \in (0, 1]$).

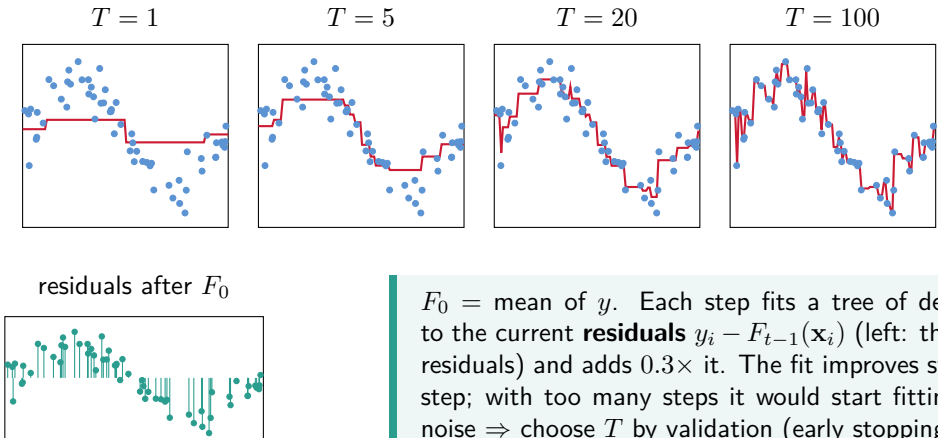
Squared loss: $r_i = y_i - F_{t-1}(\mathbf{x}_i) = \text{residuals}$.

Exponential loss: AdaBoost.

- ▶ Gradient descent *in function space*.
- ▶ Knobs: T , ν , depth of the trees, subsampling.
- ▶ XGBoost, LightGBM, CatBoost: state of the art on tabular data.

Small ν + many rounds = better generalization (shrinkage is a regularization); choose T by early stopping on validation data.

Gradient boosting step by step (regression, squared loss)



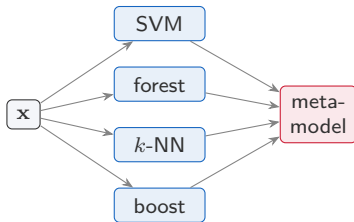
$F_0 = \text{mean of } y$. Each step fits a tree of depth 2 to the current **residuals** $y_i - F_{t-1}(\mathbf{x}_i)$ (left: the first residuals) and adds $0.3 \times$ it. The fit improves step by step; with too many steps it would start fitting the noise $\Rightarrow$ choose T by validation (early stopping).

Stacking (Wolpert, 1992)

1. Learn K different base models (SVM, forest, k -NN, boosting. . .).
2. Compute their **out-of-fold** predictions $z_{ik} = h_k^{(-j(i))}(\mathbf{x}_i)$ by CV.
3. Learn a **meta-model** (often a regularized logistic regression) on $(\mathbf{z}_i, y_i)$.

Leakage again

Training the meta-model on in-sample predictions overfits: base models are over-confident on their own training data.



A learned, data-dependent weighting of the models: a cousin of learning the posterior Q of a majority vote by minimizing a PAC-Bayes bound (TP5).

Ensembles: the big picture

Method	Base models	Diversity from	Combination	Reduces
Bagging	deep trees	bootstrap	uniform vote	variance
Random forest	deep trees	bootstrap + features	uniform vote	variance
AdaBoost	stumps	re-weighting	weighted vote	bias (+var.)
Gradient boosting	small trees	residuals	sum	bias (+var.)
Stacking	heterogeneous	different algorithms	learned	both

All of them output (essentially) a **weighted majority vote** B_Q . The theory of the course studies *when and why the vote beats its members*.

Open questions we will answer: how to bound $R_{\mathcal{D}}(B_Q)$? How to choose Q with a guarantee? How to measure diversity? $\rightarrow$ PAC-Bayes.

- ▶ A vote of **accurate** and **diverse** models beats its members (Condorcet); correlation limits the gain.
- ▶ Notation: majority vote B_Q , Gibbs classifier G_Q , margin M_Q ; often $R(B_Q) \ll R(G_Q)$ (we will prove $R(B_Q) \leq 2R(G_Q)$ in Part 11).
- ▶ **Bagging**: bootstrap + vote, reduces variance; OOB error for free. **Random forests**: + random features, less correlation.
- ▶ **AdaBoost**: re-weight the mistakes, weighted vote of weak learners; increases the margins.
- ▶ **Gradient boosting**: gradient descent in function space, fit the residuals.
- ▶ **Stacking**: learn how to combine heterogeneous models on out-of-fold predictions.

Part 9

A probabilistic toolbox

What we have observed (Parts 1–8)

- ▶ the training error is optimistic;
- ▶ complex models overfit, simple ones underfit;
- ▶ a test set or a CV gives an *estimate* of the true risk;
- ▶ a vote of many models is often much better than its members.

What we want now: statements that are **guaranteed**, such as “with probability $\geq 95\%$, $R_{\mathcal{D}}(h) \leq 0.12$ ”, and that *explain* these observations.

Part 9 – toolbox: how far is an average from its expectation? (Markov, Hoeffding, kl, union bound)



Part 10 – bounds for one model: test-set bound, finite classes and Occam, VC, Rademacher, margins



Part 11 – PAC-Bayes: bounds for a *distribution over models*, hence for **majority votes**

Why probability?

- ▶ The sample S is **random** $\Rightarrow$ the learned model h_S , its training error and its test error are random variables.
- ▶ A statement such as “ $R_{\mathcal{D}}(h_S) \leq 0.12$ ” can only hold **with high probability** over the draw of S .
- ▶ We need to quantify how far an *average* of n i.i.d. variables is from its *expectation*:

$$\underbrace{\frac{1}{n} \sum_{i=1}^n \mathbb{1}[h(\mathbf{x}_i) \neq y_i]}_{\widehat{R}_S(h)} \quad \text{vs.} \quad \underbrace{\mathbb{P}(h(\mathbf{x}) \neq y)}_{R_{\mathcal{D}}(h)}.$$

This part recalls the four tools used again and again in the lecture notes (§1.4) and in the PAC-Bayes chapters: **law of large numbers**, **Markov**, **Hoeffding**, the **kl (binomial) inequality**, plus the **union bound**.

- ▶ Linearity: $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$ (always).
- ▶ $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.
- ▶ If X, Y independent:
 $\text{Var}(X + Y) = \text{Var } X + \text{Var } Y$.
- ▶ In general: $\text{Var}(X + Y) = \text{Var } X + \text{Var } Y + 2 \text{Cov}(X, Y)$.
- ▶ Indicator: $\mathbb{E}[\mathbb{1}[A]] = \mathbb{P}(A)$.
- ▶ Tower rule: $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]]$.

Average of n variables

If $X_1, \dots, X_n$ have mean μ , variance σ^2 and pairwise correlation ρ :

$$\text{Var}\left(\frac{1}{n} \sum_i X_i\right) = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2.$$

We met this formula with bagging and random forests (Part 8): averaging helps, but only down to $\rho\sigma^2 \Rightarrow$ we need *decorrelated* models.

The law of large numbers

Law of large numbers

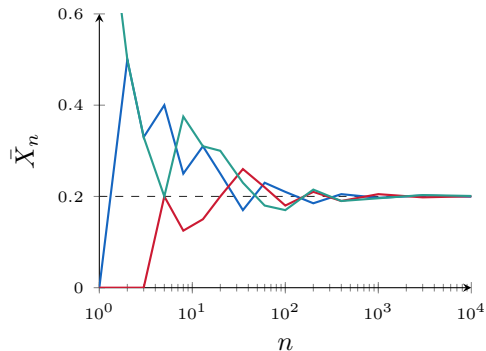
If $X_1, X_2, \dots$ are i.i.d. with $\mathbb{E}|X_1| < \infty$,

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i \xrightarrow{n \rightarrow \infty} \mathbb{E}[X_1] \quad \text{a.s.}$$

Applied to $X_i = \mathbb{1}[h(\mathbf{x}_i) \neq y_i]$ for a fixed h :
 $\hat{R}_S(h) \rightarrow R_{\mathcal{D}}(h)$.

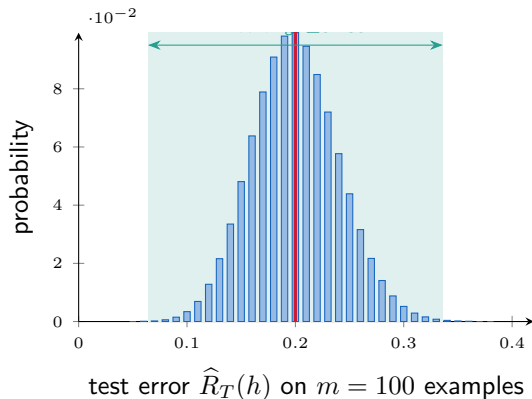
But

It says nothing about *how fast*, and nothing when h depends on S .



Three runs of the empirical error of a classifier with true risk 0.2.

The test error is a random variable



- ▶ Fix a classifier h with true risk 0.2.
- ▶ Draw a test set of $m = 100$ examples: the number of errors is $\text{Bin}(100, 0.2)$.
- ▶ The observed test error can be anywhere between ≈ 0.1 and ≈ 0.3 !
- ▶ The bars are the exact binomial probabilities.

A test error is an **estimate** with a **fluctuation**. Concentration inequalities give an interval that contains the truth with probability $\geq 1 - \delta$ (here 95%, green band).

Markov's inequality

If $X \geq 0$ and $t > 0$: $\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}[X]}{t}$.

Proof. $t \mathbb{1}[X \geq t] \leq X$, take expectations. $\square$

Chebyshev's inequality

$$\mathbb{P}(|X - \mathbb{E}X| \geq t) \leq \frac{\text{Var } X}{t^2}.$$

Proof. Markov applied to $(X - \mathbb{E}X)^2$.

Application.

$$\text{Var } \hat{R}_S(h) = R_{\mathcal{D}}(h)(1 - R_{\mathcal{D}}(h))/n \leq \frac{1}{4n}:$$

$$\mathbb{P}(|\hat{R}_S(h) - R_{\mathcal{D}}(h)| \geq t) \leq \frac{1}{4nt^2}.$$

For $\delta = 5\%$, $n = 1000$: $t \approx 0.07$.

Markov is weak but **universal**. It is the starting point of every PAC-Bayes proof: apply it to $\mathbb{E}_{h \sim P} e^{\lambda(\dots)}$ (§13–14).

Hoeffding's inequality (§1.4)

Hoeffding (1963)

Let $X_1, \dots, X_n$ be independent with $X_i \in [a, b]$ and $\bar{X}_n = \frac{1}{n} \sum X_i$. For all $t > 0$,

$$\mathbb{P}(\bar{X}_n - \mathbb{E}\bar{X}_n \geq t) \leq e^{-2nt^2/(b-a)^2}.$$

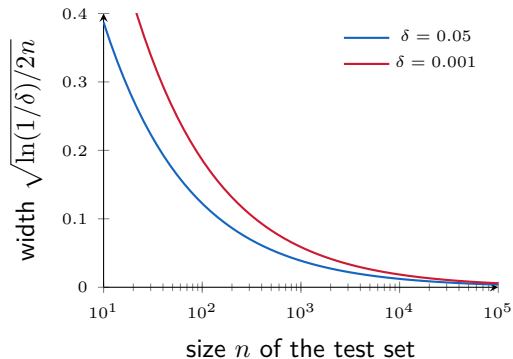
Inverted form. With probability at least $1 - \delta$ over $S \sim \mathcal{D}^n$, for a **fixed** h :

$$R_{\mathcal{D}}(h) \leq \hat{R}_S(h) + \sqrt{\frac{\ln(1/\delta)}{2n}}.$$

Exponential instead of polynomial tails:
for $n = 1000$, $\delta = 5\%$: $t \approx 0.039$ (vs. 0.07 with Chebyshev). The deviation shrinks as $1/\sqrt{n}$.

Proof idea. Chernoff: $\mathbb{P}(Z \geq t) \leq e^{-\lambda t} \mathbb{E}e^{\lambda Z}$ (Markov), Hoeffding's lemma $\mathbb{E}e^{\lambda(X-\mathbb{E}X)} \leq e^{\lambda^2(b-a)^2/8}$, independence, optimize λ .

Hoeffding: how many test examples do we need?



n	$\delta = 0.05$	$\delta = 0.001$
100	0.122	0.186
1 000	0.039	0.059
10 000	0.012	0.019

To halve the width, multiply n by 4.
The dependence in δ is only logarithmic: high confidence is cheap.

A worked example

A classifier makes **12 errors** on an independent test set of $m = 400$ examples. What can we say about its true risk, with confidence 95%?

1. Point estimate: $\hat{R}_T(h) = 12/400 = 0.03$.

2. Hoeffding (one-sided), $\delta = 0.05$:

$$R_{\mathcal{D}}(h) \leq 0.03 + \sqrt{\frac{\ln(1/0.05)}{2 \times 400}} = 0.03 + \sqrt{\frac{3.00}{800}} = 0.03 + 0.061 = \mathbf{0.091}.$$

3. kl bound (next slide): solve $\text{kl}(0.03 \parallel p) \leq \frac{\ln(2\sqrt{400/0.05})}{400} = 0.0167$ for the largest p : $R_{\mathcal{D}}(h) \leq \mathbf{0.072}$.

Reporting “3% error” hides the uncertainty: with 400 test examples, the honest statement is “**at most 7–9%** with 95% confidence”.

How many test examples do we need to guarantee a width of 0.01 with $\delta = 0.01$?
 $m \geq \ln(100)/(2 \times 0.01^2) \approx 23\,000$.

A tighter tool: the binomial kl

For $q, p \in [0, 1]$: $\text{kl}(q||p) = q \ln \frac{q}{p} + (1 - q) \ln \frac{1 - q}{1 - p}$ (KL between Bernoulli's).

Test-set bound (Langford 2005, §1.4)

With probability $\geq 1 - \delta$ over $T \sim \mathcal{D}^m$, for a fixed h ,

$$\text{kl}(\hat{R}_T(h) || R_{\mathcal{D}}(h)) \leq \frac{\ln \frac{2\sqrt{m}}{\delta}}{m},$$

$$\text{hence } R_{\mathcal{D}}(h) \leq \bar{\text{kl}}^{-1}(\hat{R}_T(h), \frac{\ln(2\sqrt{m}/\delta)}{m}).$$

$\bar{\text{kl}}^{-1}(q, \varepsilon) = \max\{p : \text{kl}(q||p) \leq \varepsilon\}$, computed by bisection.

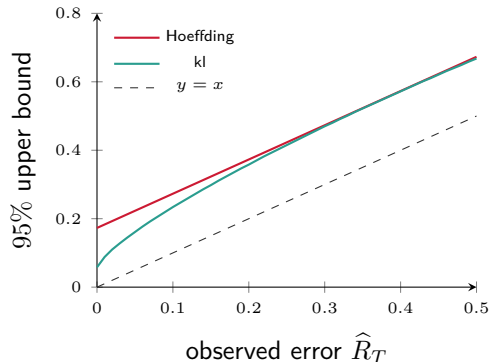
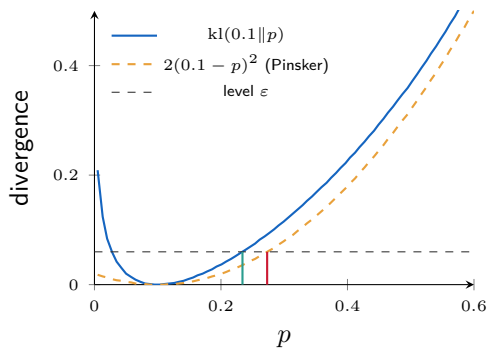
- Pinsker: $\text{kl}(q||p) \geq 2(q - p)^2 \Rightarrow$ never worse than Hoeffding (up to $\ln$ terms).
- Much tighter when the error is small.

TP1, wdbc, $m = 143$, test error 0.028:

Hoeffding bound **0.130**, kl bound **0.075**.

The kl form is the one used in Seeger's PAC-Bayes bound (Part 11).

kl vs. Hoeffding: a picture



Left: the kl bound is the largest p such that $\text{kl}(\hat{R}_T\|p)$ stays below the level ε (green); Hoeffding replaces kl by its quadratic lower bound, hence a larger p (red). Right ($m = 100$): the gain is huge for small errors. Both use the same $\varepsilon = \ln(2\sqrt{m}/\delta)/m$ here.

Union bound

$$\mathbb{P}(\cup_{k=1}^K A_k) \leq \sum_{k=1}^K \mathbb{P}(A_k).$$

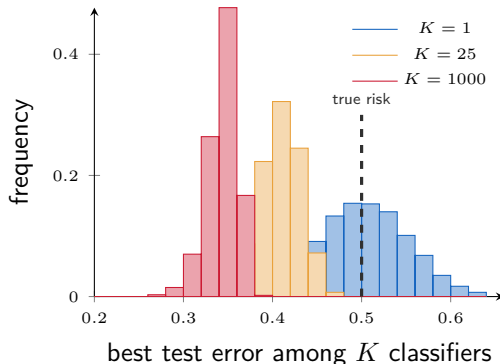
Application. If we look at K models $h_1, \dots, h_K$ on the same test set and choose the best one, each bound must hold *simultaneously*: use δ/K for each,

$$\forall k, R_{\mathcal{D}}(h_k) \leq \hat{R}_T(h_k) + \sqrt{\frac{\ln(K/\delta)}{2m}}.$$

TP1: 1000 random classifiers (true risk 0.5) evaluated on the same test set. The best one has test error 0.339 (!). Naive bound coverage: 0.4%; with the union bound: 99.7%.

Selecting = learning. The price is $\ln K$: the beginning of **Occam's razor** and of **PAC-Bayes**, where $\ln K$ becomes $\text{KL}(Q\|P)$.

The price of choosing: a simulation



- ▶ K classifiers that answer **at random** (true risk 0.5), evaluated on the same 100 test examples.
- ▶ We keep the one with the smallest test error.
- ▶ With $K = 1000$, the winner shows ≈ 0.35 test error, although it is useless.

Every time you **choose** using a sample, the sample stops being an unbiased judge of your choice. The union bound pays $\ln K$ for it.

- ▶ Errors measured on a sample are **random**: always think “estimate $\pm$ fluctuation”.
- ▶ **Markov** (universal, weak) $\rightarrow$ **Chebyshev** (variance) $\rightarrow$ **Hoeffding** (exponential tails, width $\sqrt{\ln(1/\delta)/2m}$) $\rightarrow$ **kl** (tightest for small errors).
- ▶ High confidence is cheap ($\ln \frac{1}{\delta}$), precision is expensive ($1/\sqrt{m}$).
- ▶ These bounds hold for a h **fixed before** looking at the sample. Choosing among K models costs $\ln K$ (**union bound**).
- ▶ Averages of **correlated** variables concentrate less: $\text{Var}(\bar{X}) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$.

Part 10

Generalization bounds

What is a generalization bound?

Definition – PAC guarantee (Probably Approximately Correct, Valiant 1984)

A statement of the form: for all $\delta \in (0, 1]$, with probability at least $1 - \delta$ over $S \sim \mathcal{D}^n$,

$$R_{\mathcal{D}}(h_S) \leq \hat{R}_S(h_S) + \varepsilon(n, \mathcal{H}, \delta).$$

- ▶ **Probably:** $1 - \delta$ (over the sample).
- ▶ **Approximately:** up to ε .
- ▶ Holds for *any* distribution $\mathcal{D}$ (distribution-free).
- ▶ ε decreases with n , increases with the complexity of $\mathcal{H}$ and with $\ln(1/\delta)$.

Three uses of a bound:

1. **certify** a model without a test set;
2. **understand** what drives generalization;
3. **design** algorithms: minimize the bound (SVM, boosting, PAC-Bayes).

Bound 1: the test-set bound

Learn h on S , evaluate on an independent $T \sim \mathcal{D}^m$. Since h does not depend on T :

$$\begin{aligned} R_{\mathcal{D}}(h) &\leq \overline{\text{kl}}^{-1}\left(\widehat{R}_T(h), \frac{1}{m} \ln \frac{2\sqrt{m}}{\delta}\right) \\ &\leq \widehat{R}_T(h) + \sqrt{\frac{1}{2m} \ln \frac{2\sqrt{m}}{\delta}}. \end{aligned}$$

- ▶ Tight, simple, no assumption on h .
- ▶ But the m examples of T are not used for learning.

TP1	m	$\widehat{R}_T$	kl bound
wdbc	143	0.028	0.075
pima	192	0.229	0.309

The reference: any “training-set” bound should be compared with what a test set of the same size would give.

Theorem (finite class)

Let $|\mathcal{H}| = K < \infty$, 0–1 loss. With probability $\geq 1 - \delta$ over $S \sim \mathcal{D}^n$, **for all** $h \in \mathcal{H}$,

$$R_{\mathcal{D}}(h) \leq \hat{R}_S(h) + \sqrt{\frac{\ln K + \ln(1/\delta)}{2n}}.$$

Proof. Hoeffding for each h with δ/K + union bound. □

- ▶ Holds for *all* h simultaneously $\Rightarrow$ in particular for h_S chosen by ERM.
- ▶ The complexity of the class is measured by $\ln K$ = number of bits to describe a model.

For infinite classes: VC dimension, Rademacher complexity (§6.2, §10.1). Another route, **PAC-Bayes**, replaces the union bound by a *weighted* one (Part 11).

Bound 2: finite classes and Occam's razor

Occam's razor bound

Let π be a probability distribution on a countable $\mathcal{H}$, **fixed before seeing** S . With probability $\geq 1 - \delta$, for all $h \in \mathcal{H}$:

$$R_{\mathcal{D}}(h) \leq \hat{R}_S(h) + \sqrt{\frac{\ln \frac{1}{\pi(h)} + \ln \frac{1}{\delta}}{2n}}.$$

Proof. Hoeffding with $\delta \pi(h)$ for each h ; union bound: $\sum_h \delta \pi(h) = \delta$. $\square$

- ▶ Uniform π on K models: $\ln K$ (previous theorem).
- ▶ $\pi(h) = 2^{-|h|}$ for a prefix code: $\ln \frac{1}{\pi(h)} = |h| \ln 2$ – the **description length** of h .
- ▶ Simple models (short descriptions, high prior) pay less.

The **prior** π encodes our preference *before* seeing the data. It is not a belief about the world: any π gives a valid bound, a good π gives a tight one (TP4).

Bounds on the running example: a finite class

Class: 100 classifiers

$h_{a,b}(\mathbf{x}) = \text{sign}(x_2 - a \sin(\pi x_1) - b)$,
 $a \in \{0, \frac{1}{9}, \dots, 1\}$, $b \in \{-\frac{1}{2}, \dots, \frac{1}{2}\}$ (10
values each). The Bayes classifier is
 $h_{0.5,0} \dots$ almost in the grid.

Data: $n = 200$, $\delta = 0.05$.

empirical risk of the ERM	0.120
true risk of the ERM	0.121
finite-class bound (Hoeffding)	0.258
finite-class bound (kl)	0.253
test-set bound, 200 <i>new</i> examples	0.217

- ▶ The bound is valid without any test set: it uses the **same** 200 examples for learning and certifying.
- ▶ Price of the choice among 100 models: $\ln 100 \approx 4.6$ “extra examples-worth” in the numerator.
- ▶ A test set of 200 *more* examples would give 0.217: the bound is not far from it.

We will redo this computation with PAC-Bayes (Part 11): averaging over several good classifiers instead of choosing one.

Bound 3: infinite classes – VC dimension

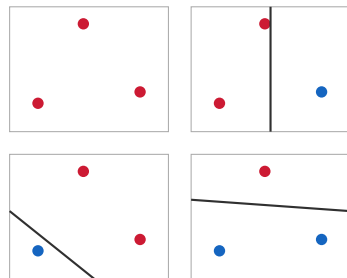
Definition – VC dimension

The largest number of points that $\mathcal{H}$ can **shatter** (label in all 2^m possible ways).

- ▶ Linear classifiers in $\mathbb{R}^d$: $d + 1$.
- ▶ Intervals on $\mathbb{R}$: 2. Axis-aligned rectangles: 4.

VC bound (informal, §6.2)

With probability $\geq 1 - \delta$, for all $h \in \mathcal{H}$,
 $R_{\mathcal{D}}(h) \leq$
 $\hat{R}_S(h) + O\left(\sqrt{\frac{d_{VC} \ln(n/d_{VC}) + \ln(1/\delta)}{n}}\right).$



Lines can shatter 3 points in the plane (4 of the 8 labellings shown), but no set of 4 points.

Worst case over $\mathcal{D}$; often **vacuous** (bound > 1) for rich models (deep trees, forests, deep nets).

Bound 4: Rademacher complexity (§10.1)

Definition – Empirical Rademacher complexity

For i.i.d. signs $\sigma_i \in \{-1, +1\}$ with probability 1/2:

$$\hat{\mathfrak{R}}_S(\mathcal{H}) = \mathbb{E}_{\sigma} \left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \sigma_i h(\mathbf{x}_i) \right].$$

How well can $\mathcal{H}$ correlate with **random noise**?

Rademacher bound

With prob. $\geq 1 - \delta$, $\forall h \in \mathcal{H}$:

$$R_{\mathcal{D}}(h) \leq \hat{R}_S(h) + \hat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\ln(2/\delta)}{2n}}.$$

(0–1 loss, $\mathcal{Y} = \{-1, +1\}$.)

Data-dependent. Key property for ensembles: the Rademacher complexity of the **convex hull** of $\mathcal{H}$ equals that of $\mathcal{H} \Rightarrow$ margin bounds for boosting (§10.2).

For a vote $f(\mathbf{x}) = \sum_m q_m h_m(\mathbf{x})$ with $q \in$ simplex, and a margin threshold $\theta > 0$:

Margin bound (Schapire et al. 1998, informal)

With prob. $\geq 1 - \delta$, for all votes f ,

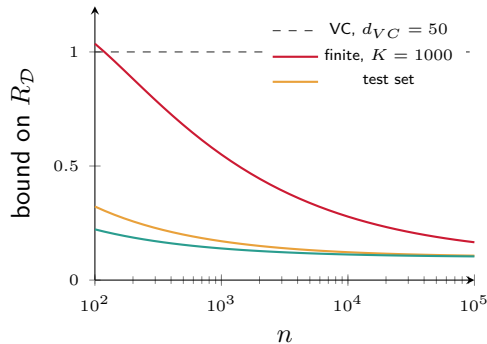
$$R_{\mathcal{D}}(\text{sign } f) \leq \hat{\mathbb{P}}_S(yf(\mathbf{x}) \leq \theta) + O\left(\frac{\mathfrak{R}_n(\mathcal{H})}{\theta} + \sqrt{\frac{\ln(1/\delta)}{n}}\right).$$

The number of voters does **not** appear!

Explains why boosting keeps improving the test error after the training error reaches 0: it keeps **increasing the margins** (TP3).

The margin of a vote $yf(\mathbf{x})$ will be the central random variable of the PAC-Bayesian analysis of majority votes (first-order, tandem, C-bound – Part 11).

Bounds in practice: are they useful?



Empirical risk 0.10, $\delta = 0.05$ (schematic constants).

- ▶ Worst-case bounds are often loose / vacuous for modern models.
- ▶ But they give the right **shape**: $\text{risk} \leq \text{empirical risk} + \sqrt{\text{complexity}/n}$.
- ▶ **PAC-Bayes** bounds are among the tightest known: non-vacuous for deep nets and forests, sometimes close to test-set bounds, *without* sacrificing data.

What makes them tight: they are data-dependent, they use the kl form, and they apply to *randomized* or *averaged* predictors – exactly ensembles.

- ▶ A **PAC bound**: with probability $\geq 1 - \delta$, true risk $\leq$ empirical risk + complexity term.
- ▶ **Test-set bound**: simplest and tight, but costs data.
- ▶ **Finite class / Occam**: complexity $\ln |\mathcal{H}|$, or $\ln \frac{1}{\pi(h)}$ with a **prior** π fixed in advance.
- ▶ **VC dimension, Rademacher complexity**: infinite classes; worst-case, often loose for rich models.
- ▶ **Margin bounds**: votes with large margins generalize, whatever the number of voters.
- ▶ Toward PAC-Bayes: replace “choose one h ” by “weight all h ” and $\ln \frac{1}{\pi(h)}$ by $\text{KL}(Q\|P)$.

Part 11

PAC-Bayes: a theory tailored to majority votes

Why a theory tailored to ensembles?

Classical bounds (Part 10) certify *one* hypothesis h chosen in a class $\mathcal{H}$:

- ▶ the complexity is that of the whole class (worst case);
- ▶ a forest of 500 deep trees lives in a gigantic class $\Rightarrow$ vacuous bounds;
- ▶ they do not see *why* the vote works: averaging, diversity.

An ensemble is a distribution Q over $\mathcal{H}$ (the weights of the vote): bagging, forests: Q uniform on the trees; boosting: $Q \propto \alpha_t$.

PAC-Bayes bounds directly the risk of a **distribution** Q over models, with a complexity $\text{KL}(Q\|P)$ that decreases when Q **spreads** over many good models.

- ▶ The Gibbs risk $R(G_Q)$ = average risk of the voters: bounded by PAC-Bayes.
- ▶ The vote B_Q : related to G_Q by first-order, tandem, C-bound – which involve **diversity**.
- ▶ Bounds hold for all Q : we can **learn** the weights of a vote by minimizing them.

Occam (Part 10): prior π , *one* model h :

$$R_{\mathcal{D}}(h) \leq \hat{R}_S(h) + \sqrt{\frac{\ln \frac{1}{\pi(h)} + \ln \frac{1}{\delta}}{2n}}.$$

PAC-Bayes: prior P , *distribution* Q over models:

$$R_{\mathcal{D}}(G_Q) \leq R_S(G_Q) + \sqrt{\frac{\text{KL}(Q\|P) + \ln(2\sqrt{n}/\delta)}{2n}}.$$

- ▶ $Q = \delta_h$ (all mass on h), $P = \pi$ discrete: $\text{KL}(Q\|P) = \ln \frac{1}{\pi(h)}$: we recover Occam.
- ▶ Spreading Q over many good models **reduces** $\text{KL}(Q\|P)$: averaging is rewarded!
- ▶ Works for **infinite / continuous** classes (linear classifiers, neural nets).

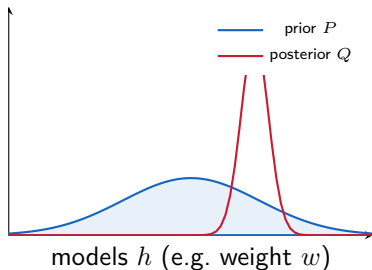
McAllester (1999), Seeger (2002), Catoni (2007). Chapters 12–19 of the notes.

Definition – Prior P

A distribution on $\mathcal{H}$ chosen **before** seeing S (or on data independent of S). Encodes which models we expect to be good.

Definition – Posterior Q

Any distribution on $\mathcal{H}$, chosen **after** seeing S . Not necessarily Bayes' posterior.



The bounds hold for **all** Q simultaneously $\Rightarrow Q$ can be optimized on S . The price is $\text{KL}(Q\|P)$: how far we moved from the prior.

Definition – KL divergence

$$\text{KL}(Q\|P) = \mathbb{E}_{h \sim Q} \left[\ln \frac{Q(h)}{P(h)} \right] \quad (\text{densities or probabilities}), \quad +\infty \text{ if } Q \not\ll P.$$

- ▶ $\text{KL}(Q\|P) \geq 0$, $= 0$ iff $Q = P$ (Gibbs' inequality, Jensen).
- ▶ Bernoulli: $\text{KL} = \text{kl}(q\|p)$ (Part 9).
- ▶ Gaussians:
 $\text{KL}(\mathcal{N}(\mu, \sigma^2 I) \| \mathcal{N}(\mu_0, \sigma^2 I)) = \frac{\|\mu - \mu_0\|^2}{2\sigma^2}.$

Gaussian case = **squared ℓ_2 penalty**:
PAC-Bayes rediscovers ridge / SVM regularization.

Example

P uniform on K models, Q uniform on k of them: $\text{KL}(Q\|P) = \sum_h \frac{1}{k} \ln \frac{1/k}{1/K} = \ln \frac{K}{k}.$
One model ($k = 1$): $\ln K$, the union bound. Spreading on k models saves $\ln k$.

Donsker–Varadhan variational formula

For any measurable $\phi : \mathcal{H} \rightarrow \mathbb{R}$ and any P :

$$\ln \mathbb{E}_{h \sim P}[e^{\phi(h)}] = \sup_Q (\mathbb{E}_{h \sim Q}[\phi(h)] - \text{KL}(Q \| P)).$$

Recipe of a PAC-Bayes proof (§13–14):

1. Take $\phi(h) = \lambda(R_{\mathcal{D}}(h) - \hat{R}_S(h))$ (or $n \text{kl}(\hat{R}_S(h) \| R_{\mathcal{D}}(h))$).
2. Change of measure: for all Q , $\mathbb{E}_Q[\phi] \leq \text{KL}(Q \| P) + \ln \mathbb{E}_P[e^{\phi}]$.
3. $\mathbb{E}_P[e^{\phi}]$ does not depend on Q and P does not depend on S : by **Markov**, w.p. $\geq 1 - \delta$,
 $\mathbb{E}_P[e^{\phi}] \leq \frac{1}{\delta} \mathbb{E}_S \mathbb{E}_P[e^{\phi}] = \frac{1}{\delta} \mathbb{E}_P \mathbb{E}_S[e^{\phi}]$.
4. Bound $\mathbb{E}_S[e^{\phi(h)}]$ for a *fixed* h : **Hoeffding's lemma** / binomial computations (Part 9).

Everything we saw in Part 9 – Markov, Hoeffding, kl – is used, plus one new ingredient: the change of measure.

With probability at least $1 - \delta$ over $S \sim \mathcal{D}^n$, **for all** posteriors Q simultaneously:

Seeger's (kl) bound

$$\text{kl}(R_S(G_Q) \parallel R_{\mathcal{D}}(G_Q)) \leq \frac{\text{KL}(Q \parallel P) + \ln \frac{2\sqrt{n}}{\delta}}{n}.$$

McAllester's bound (via Pinsker)

$$R_{\mathcal{D}}(G_Q) \leq R_S(G_Q) + \sqrt{\frac{\text{KL}(Q \parallel P) + \ln \frac{2\sqrt{n}}{\delta}}{2n}}.$$

Linear (Catoni-type) bound, for a fixed $\lambda > 0$

$$R_{\mathcal{D}}(G_Q) \leq R_S(G_Q) + \frac{\text{KL}(Q \parallel P) + \ln \frac{1}{\delta}}{\lambda} + \frac{\lambda}{8n}.$$

$$\underbrace{R_{\mathcal{D}}(G_Q)}_{\text{true risk of the randomized classifier}} \leq \underbrace{R_S(G_Q)}_{\text{fit}} + \underbrace{\sqrt{\frac{\text{KL}(Q\|P) + \ln \frac{2\sqrt{n}}{\delta}}{2n}}}_{\text{complexity = distance to the prior}}$$

- ▶ Empirical risk + complexity: **same shape** as regularized ERM (Part 5).
- ▶ Complexity is **data-dependent** (through Q), not a worst case over $\mathcal{H}$.
- ▶ Valid for any Q : we may minimize the right-hand side $\Rightarrow$ learning algorithms with guarantees.

Trade-off in Q : concentrating on the best empirical models lowers $R_S(G_Q)$ but increases $\text{KL}(Q\|P)$; a spread-out Q does the opposite. The optimum is an **average of good models**, i.e. an ensemble.

The optimal posterior: the Gibbs posterior (§15)

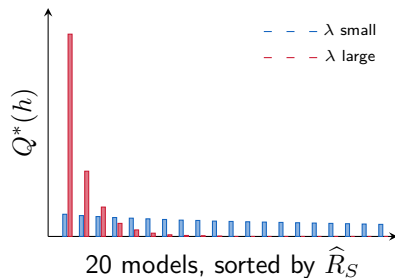
Minimizing the linear bound over Q :

$$\min_Q \mathbb{E}_{h \sim Q} \hat{R}_S(h) + \frac{\text{KL}(Q \| P)}{\lambda}$$

has the closed-form solution
(Donsker–Varadhan):

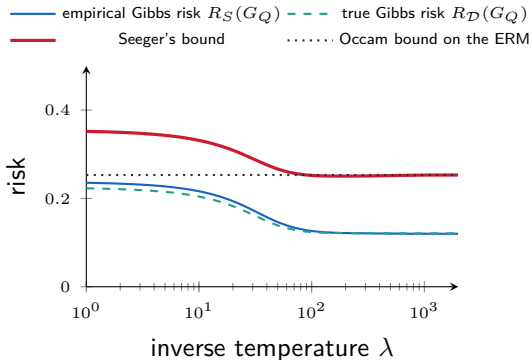
$$Q_\lambda^*(h) = \frac{P(h) e^{-\lambda \hat{R}_S(h)}}{\mathbb{E}_{h' \sim P} e^{-\lambda \hat{R}_S(h')}}.$$

- ▶ $\lambda \rightarrow 0$: $Q^* = P$ (ignore data).
- ▶ $\lambda \rightarrow \infty$: Q^* concentrates on the ERM.
- ▶ Exponential weights, simulated annealing, Bayesian posterior with “temperature” $1/\lambda$.



TP4: the Gibbs posterior on a finite set of classifiers, its bound, and coverage checked by simulation.

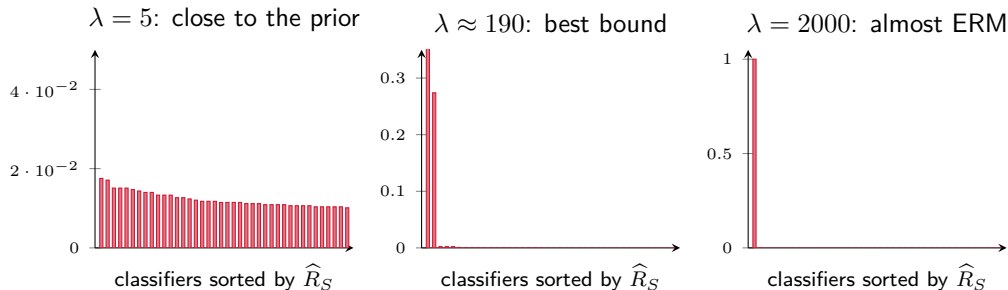
PAC-Bayes on the running example



- ▶ Same finite class of 100 classifiers and same 200 examples as in Part 10; prior P uniform.
- ▶ Posterior = Gibbs posterior
 $Q_\lambda^* \propto e^{-\lambda \widehat{R}_S(h)}$, for a range of λ .
- ▶ Small λ : $Q \approx P$. Large λ : $Q \rightarrow \text{ERM}$, $\text{KL} \rightarrow \ln 100 = 4.6$.
- ▶ Best bound **0.250** at $\lambda \approx 190$ with $\text{KL} = 3.96$; true Gibbs risk 0.121.

Averaging over the $\approx$ good classifiers pays less than selecting one ($3.96 < 4.6$). With a finite class the gain is small; with continuous classes it is what makes PAC-Bayes bounds non-vacuous.

What the posterior looks like



The 40 best classifiers (out of 100), sorted by empirical risk. The best posterior spreads its mass over the handful of classifiers whose empirical risk is close to the minimum: it does not bet everything on the one that was luckiest on S . This is the PAC-Bayesian justification of **averaging**.

The vote is at most twice worse than the Gibbs classifier

First-order bound (§16)

For all Q : $R_{\mathcal{D}}(B_Q) \leq 2 R_{\mathcal{D}}(G_Q)$.

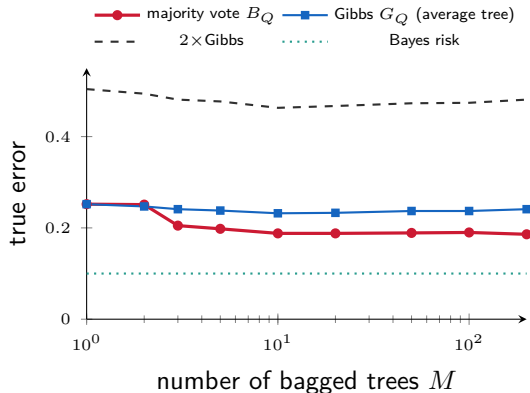
Proof. B_Q errs $\Rightarrow M_Q \leq 0 \Rightarrow 1 - M_Q \geq 1$. By Markov:

$$R_{\mathcal{D}}(B_Q) \leq \mathbb{P}(1 - M_Q \geq 1) \leq \mathbb{E}[1 - M_Q] = 2R_{\mathcal{D}}(G_Q). \quad \square$$

- ▶ Simple: bound the Gibbs risk (an *average*, easy to control) and multiply by 2.
- ▶ Loose when voters are diverse: the vote is often *better* than the average voter (TP1: vote 0.137, average tree 0.194).

To capture the benefit of the vote, we need **second-order** information: how often two voters err *together* (joint error), how often they disagree (disagreement) $\rightarrow$ tandem bound and C-bound (below, TP5).

Vote, Gibbs and first-order bound on the running example



- ▶ Deep trees on bootstrap samples of the 150 points; Q uniform on the first M trees.
- ▶ Gibbs risk = average error of a tree ≈ 0.24 , whatever M .
- ▶ The vote improves quickly and reaches 0.186.
- ▶ The first-order bound $2R(G_Q) \approx 0.48$ is true... but useless here.

The vote is better than its average member because trees **disagree**: on 40 trees, joint error 0.14, disagreement 0.19 (see Part 11).

PAC-Bayes for majority votes (§16–18)

The bounds control the **Gibbs risk**. For the **vote** we combine them with:

First-order

$$R_{\mathcal{D}}(B_Q) \leq 2R_{\mathcal{D}}(G_Q).$$

Tandem bound (Masegosa et al. 2020)

$$R_{\mathcal{D}}(B_Q) \leq 4e_{\mathcal{D}}(Q), \text{ where the } \mathbf{joint\ error} \\ e_{\mathcal{D}}(Q) = \mathbb{E}_{h,h' \sim Q^2} \mathbb{P}(h(\mathbf{x}) \neq y, h'(\mathbf{x}) \neq y).$$

C-bound (Lacasse et al. 2007)

If $R_{\mathcal{D}}(G_Q) < \frac{1}{2}$:

$$R_{\mathcal{D}}(B_Q) \leq 1 - \frac{(1 - 2R_{\mathcal{D}}(G_Q))^2}{1 - 2d_{\mathcal{D}}(Q)},$$

with the **disagreement** $d_{\mathcal{D}}(Q) = \mathbb{E}_{h,h'} \mathbb{P}(h(\mathbf{x}) \neq h'(\mathbf{x}))$.

Proofs: Markov on $1 - M_Q$ (first order), on $(1 - M_Q)^2$ (tandem), Cantelli–Chebyshev (C-bound). Each quantity $(R(G_Q), e(Q), d(Q))$ is an average over voters or pairs of voters $\Rightarrow$ bounded by PAC-Bayes. Low joint error / high disagreement = **diversity**, now quantified.

For every Q :

$$R_{\mathcal{D}}(G_Q) = e_{\mathcal{D}}(Q) + \frac{1}{2} d_{\mathcal{D}}(Q).$$

- ▶ For a fixed average accuracy $R(G_Q)$, more disagreement $\Rightarrow$ smaller joint error $\Rightarrow$ better vote.
- ▶ Bagging, random features, boosting's re-weighting: all **increase** $d(Q)$ while keeping $R(G_Q) < 1/2$.
- ▶ Disagreement does *not* need labels: it can be estimated on unlabelled data.

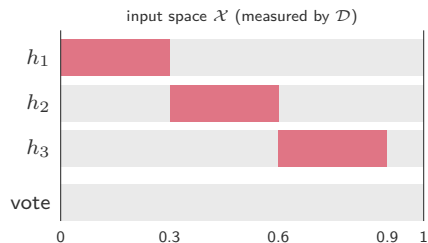
Why?

For two voters h, h' at a point: either both err (e), or exactly one errs (then they disagree), or none. Hence

$$\frac{1}{2} (\mathbb{1}[h \text{ errs}] + \mathbb{1}[h' \text{ errs}]) = \mathbb{1}[\text{both err}] + \frac{1}{2} \mathbb{1}[h \neq h']; \text{ average over } h, h' \sim Q \text{ and over } \mathcal{D}.$$

Bagged trees of the running example (40 trees): $R(G_Q) = 0.237 = 0.144 + \frac{1}{2} 0.186$.

Three voters: comparing the bounds on an example



Red = region where the voter errs (probability 0.3 each, disjoint regions).

Q uniform on $\{h_1, h_2, h_3\}$. Everywhere at most one voter errs $\Rightarrow$ the vote never errs.

quantity	value	bound on $R(B_Q)$
Gibbs risk $R(G_Q)$	0.3	first order: 0.6
joint error $e(Q)$	$\frac{3 \times 0.3}{9} = 0.1$	tandem: 0.4
disagreement $d(Q)$	$\frac{6 \times 0.6}{9} = 0.4$	C-bound: 0.2
true $R(B_Q)$		0

The more the bound uses about **pairs** of voters (joint errors, disagreements), the better it sees the benefit of diversity. In practice these quantities are estimated on data and bounded with PAC-Bayes (TP5).

TP4 – first steps

- ▶ kl, union and Occam bounds on simple classifiers;
- ▶ prior choice and its impact;
- ▶ Gibbs classifier, KL, Seeger and McAllester bounds;
- ▶ Gibbs posterior and coverage by simulation.

TP5 – majority votes

- ▶ first-order, tandem, C-bound;
- ▶ Condorcet revisited;
- ▶ certify a random forest with an independent sample (the trees' OOB data);
- ▶ learn the weights Q of a vote by minimizing a bound.

Typical outcome: a random forest certified by a **non-vacuous** PAC-Bayes bound, close to what a test set would give – a guarantee obtained *without* sacrificing a held-out test set.

Where PAC-Bayes is used today

- ▶ **Non-vacuous bounds for deep networks** (Dziugaite & Roy 2017; Pérez-Ortiz et al. 2021).
- ▶ **Certified ensembles**: random forests, boosting, weighted votes (Germain et al. 2015; Masegosa et al. 2020).
- ▶ **Self-bounding algorithms**: learn by minimizing a bound, get a certificate for free (§17–18).
- ▶ Links with **Bayesian learning** (marginal likelihood), **information theory** (mutual information bounds), **online learning** (exponential weights), **flat minima** and robustness of neural nets.
- ▶ Domain adaptation, meta-learning, reinforcement learning, fairness. . .

Message of the course: **averaging good and diverse models** is both a practical recipe (ensembles) and a theoretically sound one (PAC-Bayes).

- ▶ PAC-Bayes bounds the risk of a **distribution** Q over models (Gibbs classifier, and through it the majority vote).
- ▶ Complexity = $\text{KL}(Q\|P)$: how far the posterior moved from a prior fixed before seeing the data.
- ▶ Proof recipe: change of measure (Donsker–Varadhan) + Markov + Hoeffding/kl for a fixed h .
- ▶ Seeger (kl), McAllester ($\sqrt{\cdot}$), Catoni (linear) bounds; the **Gibbs posterior** $Q^* \propto P e^{-\lambda \hat{R}_S}$ minimizes the linear one.
- ▶ Majority votes: first-order $2R(G_Q)$, tandem $4e(Q)$, C-bound; $R(G_Q) = e(Q) + \frac{1}{2}d(Q)$ quantifies diversity.
- ▶ Bounds are valid for all Q : they can be **minimized** to learn, giving self-certified models.

Part 12

Summary and references

I. Learning in practice

- ▶ Learning = finding h with small **true risk** $R_{\mathcal{D}}(h)$ from an i.i.d. sample.
- ▶ Data must be **represented**, cleaned and normalized – inside the pipeline.
- ▶ The Bayes classifier $\text{sign}(2\eta - 1)$ is the target; R^* the floor.
- ▶ **Bias–variance**: the test error is U-shaped in the complexity.
- ▶ Regularized ERM with **convex surrogates**.
- ▶ **Protocol**: train/validation/test, CV, nested CV, no leakage, test once.

II. Algorithms

- ▶ k -NN, linear and logistic models, SVM and kernels, trees, neural networks.
- ▶ Each has its complexity knobs, its need (or not) of scaling, its stability.
- ▶ Unstable low-bias models (deep trees) → bagging; weak cheap models (stumps) → boosting.

III. Combining models

- ▶ Averaging reduces variance down to $\rho\sigma^2$
 $\Rightarrow$ **diversity** matters.
- ▶ Bagging/forests (variance), boosting (bias), stacking (learned weights).
- ▶ Majority vote B_Q , Gibbs classifier G_Q , margin M_Q .

IV. Guarantees

- ▶ Concentration: Markov, Hoeffding, **kl**, union bound.
- ▶ Bounds = empirical risk + $\sqrt{\text{complexity}/n}$; complexity = $\ln K$, $\ln \frac{1}{\pi(h)}$, VC, Rademacher.
- ▶ PAC-Bayes: prior P , posterior Q , complexity $\text{KL}(Q\|P)$; Gibbs posterior; votes: $R(B_Q) \leq 2R(G_Q)$, tandem, C-bound.

Next lectures: **ensemble methods** (§10) $\rightarrow$ TP2, TP3; then **PAC-Bayes** (§12–19) $\rightarrow$ TP4, TP5.

Books

- ▶ Hastie, Tibshirani, Friedman. *The Elements of Statistical Learning*, 2009.
- ▶ Shalev-Shwartz, Ben-David. *Understanding Machine Learning*, 2014.
- ▶ Mohri, Rostamizadeh, Talwalkar. *Foundations of ML*, 2018.
- ▶ Bach. *Learning Theory from First Principles*, 2024.
- ▶ Zhou. *Ensemble Methods*, 2012.
- ▶ Alquier. *User-friendly intro. to PAC-Bayes bounds*, 2024.

Articles

- ▶ Breiman. Bagging predictors (1996); Random forests (2001).
- ▶ Freund, Schapire. A decision-theoretic generalization of on-line learning (1997).
- ▶ Friedman. Greedy function approximation: a gradient boosting machine (2001).
- ▶ McAllester. PAC-Bayesian model averaging (1999).
- ▶ Seeger. PAC-Bayesian bounds for GP classification (2002).
- ▶ Langford. Practical prediction theory for classification (2005).
- ▶ Lacasse et al. PAC-Bayes bounds for the majority vote (2007).
- ▶ Dziugaite, Roy. Non-vacuous generalization bounds for deep networks (2017).
- ▶ Masegosa et al. Second order PAC-Bayesian bounds for the weighted majority vote (2020).

Questions?

Next time: ensemble methods – bagging and random forests
