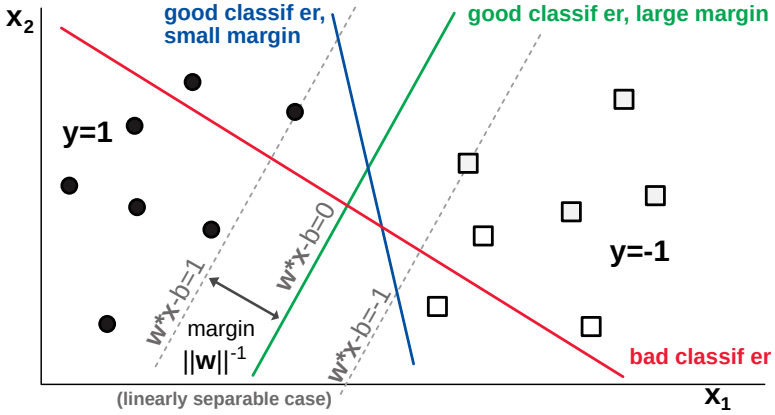
Extensions

Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

Introduction to the simple case

A popular tool from mid-1990s to late 2000s. The idea: **separate** the space via **hyperplanes** ($\neq$ trees). Ex: a linear function $h(x_1, x_2) = w_1 x_1 + w_2 x_2 - b$. If $h(x) > 0$, predict 1, if not, predict -1 (classification task).



In optimisation terms

In the **linearly separable case**, the goal is to maximise the margin under the constraint of correct classification, i.e., to find

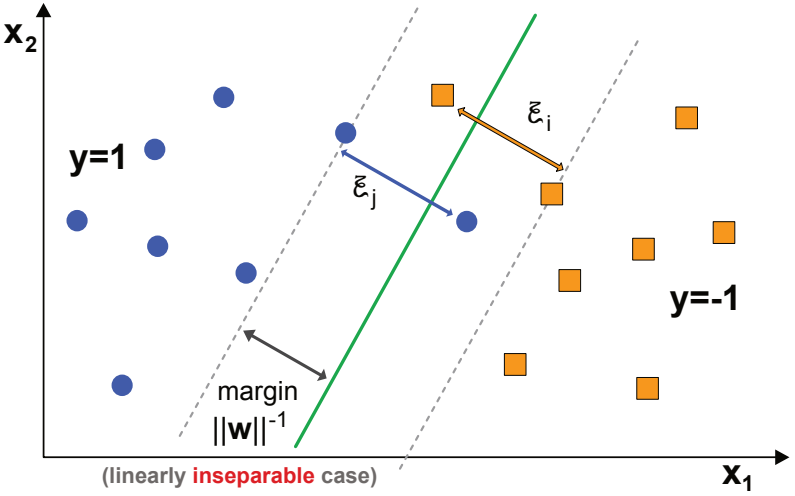
$$\min_{\boldsymbol{w},b} ||\boldsymbol{w}||^2, \quad \text{s.t.} \;\; y_i(\boldsymbol{x}_i'\boldsymbol{w} - b) \geq 1 \quad \forall i = 1, \ldots, I.$$

In most (all?) real life applications, the data is not linearly separable. Hence, the constraint must be relaxed: we introduce some margin of error through so-called slack variables ξ_i:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} ||\boldsymbol{w}||^2 + C\sum_{i=1}^{I} \xi_i, \quad \text{s.t.} \; \begin{cases} y_i(\boldsymbol{x}_i'\boldsymbol{w} - b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad \forall i = 1, \ldots, I.$$

The C constant penalises the use of the ξ_i: it serves as factor for misclassification tolerance.

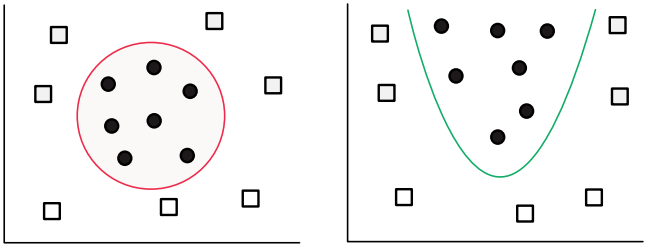
Slack errors: illustration



Extensions

Some additional features include:

- a kernel transform for the features: $\phi(\boldsymbol{x}_i)$ with ϕ polynomial, radial or sigmoid
- other objective functions for (possibly multi-class) classification AND **regression**!



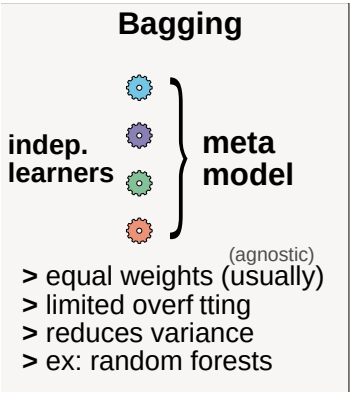
See the LIBSVM documentation (Chang & Lin (2001-2022)) for a detailed account!
In practice, SVMs have been overtaken by tree-methods & neural networks.

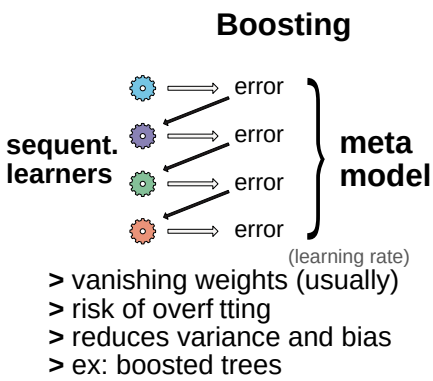
Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

Principle

See in synonyms: **model aggregation**, **forecast combination**. Main reference: **Ensemble Methods** (Zhou, 2012).





Optimal combinations (1/3)

$\rightarrow$ Minimum variance applied to models!
Suppose you have N learners (models, algos) M_n. Individually, these models yield (training) errors $\boldsymbol{e}_n$. We write $\boldsymbol{E} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N]$. We are interested in a meta-model $M = \sum_{n=1}^N w_n M_n$, with $\sum_{n=1}^N w_n = 1$.

The errors of the meta-model are $\boldsymbol{e} = \sum_{n=1}^N w_n \boldsymbol{e}_n = \boldsymbol{E}\boldsymbol{w}$. The quadratic error is thus $\boldsymbol{e}'\boldsymbol{e}$ and we seek to minimise it under the weight constraint:

$$\min_{\boldsymbol{w}} \boldsymbol{w}'\boldsymbol{E}'\boldsymbol{E}\boldsymbol{w}, \quad \text{s.t.} \quad \boldsymbol{w}'\mathbf{1} = 1,$$

with solution (the derivation is the same as for minimum variance portfolios with $\mathbf{E}'\mathbf{E} = \boldsymbol{\Sigma}$)

$$\boldsymbol{w}^* = \frac{(\boldsymbol{E}'\boldsymbol{E})^{-1}\mathbf{1}}{\mathbf{1}'(\boldsymbol{E}'\boldsymbol{E})^{-1}\mathbf{1}}.$$

Usually, the errors have zero mean so $\boldsymbol{E}'\boldsymbol{E}$ is their covariance matrix.

Optimal combinations (2/3)

Recall:

$$\boldsymbol{w}^* = \frac{(\boldsymbol{E}'\boldsymbol{E})^{-1}\mathbf{1}}{\mathbf{1}'(\boldsymbol{E}'\boldsymbol{E})^{-1}\mathbf{1}}.$$

This imposes (because of matrix inversion):

- that the number of observations be large enough (much larger than the number of models - usually easy)
- that the correlation between models be not too close to 1 in absolute value: that's the challenging part in fact! If not: huge **leverage** effects occur: betting on some algo *vs* the others.

Moreover, if you expect it to work well out-of-sample, the correlation between errors has to display some form of **stationarity**: not obvious in practice!

Optimal combinations (3/3)

Stacked ensembles

In his seminal 1996 paper, Leo Breiman argues that enforcing a **positivity constraint** is a good idea and that making bets or arbitraging models is a bad one. Hence, a much better program is:

$$\min_{\boldsymbol{w}} \boldsymbol{w}' \boldsymbol{E}' \boldsymbol{E} \boldsymbol{w}, \quad \text{s.t.} \quad \boldsymbol{w}' \mathbf{1} = 1, \quad w_i \geq 0.$$

The only drawback is that there is no closed-form solution and the optimal weights must be approximated. In the case of highly correlated models, corner solutions may appear and it may be the case that one learner gets a 100% weight.
$\rightarrow$ this will be the learner with the smallest training or validation error.

More generally

Meta / super learners push the concept further to **nonlinear** models:

Stage 1:
first learning level
simple training
and prediction

Model 1
Model 2
...
Model M

⇩

I*M = nb predictions
(I = nb instances)

Stage 2:
2nd learning level
optimise combination
or feed new learner

estimate this model:

$y = f(p_1, p_2, \ldots, p_M)$

$\hat{f}$ is the aggregate meta model

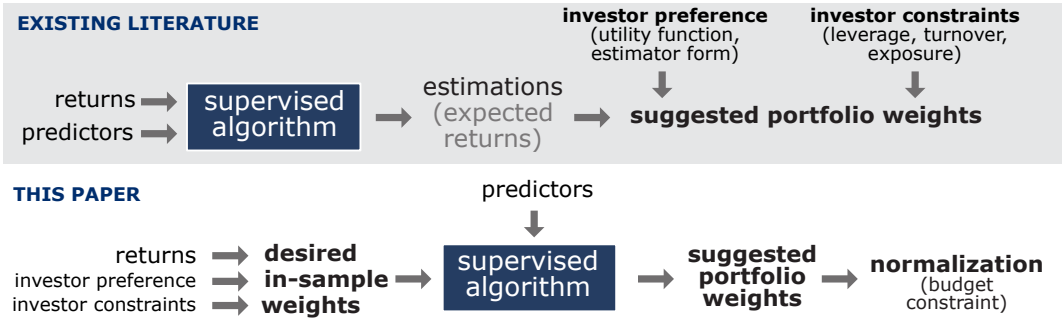
Stage 3:
Forecast!
reverse operation:
two step prediction

1. Make the forecasts at indiv. learner level
2. Feed the forecasts to the second model $\hat{f}$

→ useful only if raw predictions are **not too correlated**! (if they tell different stories and reveal complementary facets)

Supervised portfolios

Shift the learning phase at the very end of the process!



Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

Two scales

- **Global**: the characteristics of the model over all features, once the model has been trained
- **Local**: how the model behaves for one instance in particular

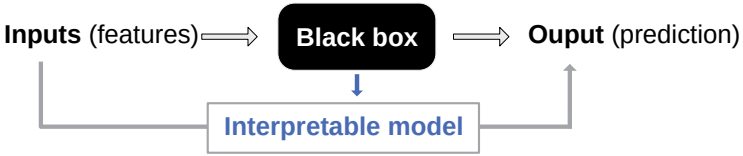
The pros/cons of each approach are straightforward: the macro view will overlook localised idiosyncrasies while the micro view cannot pretend to represent '*the big picture*' (+ there are so many instances! ok for doctor/patient, not asset predictions).

Two examples of the first category are variable importance in tree methods and *t*-stats in linear models. Variable importance can be assessed for any models by considering particular permutations of features (omitting one feature each time).

On the topic of interpretability, see Interpretable Machine Learning.
Below, we detail two techniques of local interpretability.

The idea in simple terms

Inspiration: **LIME** from Ribeiro et al. (2016):[1] **instance-based**!



Requirements:

- **simple interpretability**: e.g., not thousands of variables AND with visual or textual representation
- **local faithfulness**: the explanation holds for the vicinity of the instance

Basic blocks are regression or simple decision trees.

[1] We skip the notion of interpretable representation

The idea in math terms

By the way, LIME = **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations.

The original (complex) model is f and it is approximated at instance x by the interpretable model g that belongs to a large class G. The vicinity of x is denoted π_x and the complexity of g is written $\Omega(g)$. LIME seeks an interpretation of the form

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \, \mathcal{L}(f, g, \pi_x) + \Omega(g),$$

where $\mathcal{L}(f, g, \pi_x)$ is the loss function (error/imprecision) induced by g in the vicinity π_x of x.

The penalisation $\Omega(g)$ is for instance the number of leaves or depth of a tree or the number of predictors in a linear regression.

More precisely

The vicinity of x is defined by $\pi_x(z) = e^{-D(x,z)^2/\sigma^2}$, where D is some distance measure.
Note: this function decreases when z shifts away from x.

The tricky part is the loss function. In order to minimise it, LIME generates artificial samples close to x and averages/sums the error on the label that the simple representation makes. The formulation is the following:

$$\mathcal{L}(f, g, \pi_x) = \sum_z \pi_x(z)(f(z) - g(z))^2$$

the errors are weighted according to their distance from the initial instance x.

Visually

Source: Ribeiro et al. (2016)

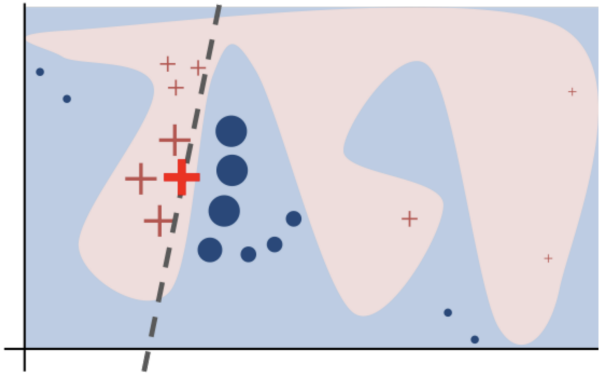


Figure 3: Toy example to present intuition for LIME.

In the current version of the **lime** R package, the approximation is indeed locally linear.

Another approach: Shapley values

From cooperative game theory. What is the value of feature x_k? Well, it depends on what happens if you remove it! The formula is the following:

$$\phi_k = \sum_{S \subseteq \{x_1, \dots, x_K\} \setminus x_k} \underbrace{\frac{|S|!(K - |S| - 1)!}{K!}}_{\text{weight of coalition}} \underbrace{\left(f_{S \cup \{x_k\}}(S \cup \{x_k\}) - f_S(S)\right)}_{\text{gain when adding } x_k}$$

S is any subset of the "**coalition**" that doesn't include feature k; its size/cardinal is $|S|$.
In the equation above, the model f must be altered because it's impossible to evaluate f when features are missing. In this case, several possible options:

- set the missing value to its average or median value (in the whole sample) so that its effect is some 'average' effect
- directly compute an average value $\int_{\mathbb{R}} f(x_1, \dots, x_k, \dots, x_K) d\mathbb{P}_{x_k}$, where $d\mathbb{P}_{x_k}$ is the empirical distribution of x_k in the sample

Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

Deflated Sharpe ratio (1/2)

source: Bailey and Lopez de Prado (2014)

The core idea:

- When backtesting strategies, the investor/fund manager will usually pick the one that performs best.
- In practice, this will probably **not be the case**.
- Hence, the performance metrics obtained from the backtest must be curtailed.
- A rule of thumb is for instance to divide the SR by two!

The idea here is to create a test comparing the best SR obtained in the backtest to a **theoretical value** for the average maximum SR (given the characteristics of all tested strategies: it requires to store all backtest results).

Deflated Sharpe ratio (2/2)

The formula is:

$$t = \phi\left(\frac{(SR - SR^*)\sqrt{T-1})}{\sqrt{1 - \gamma_3 SR + \frac{\gamma_4 - 1}{4} SR^2}}\right),$$

where SR is the Sharpe Ratio obtained by the best strategy, and

$$SR^* = \mathbb{E}[SR_m] + \sqrt{\mathbb{V}[SR_m]}\left((1-\gamma)\phi^{-1}\left(1 - \frac{1}{N}\right) + \gamma\phi^{-1}\left(1 - \frac{1}{Ne}\right)\right),$$

is the theoretical average best SR. Moreover,

- γ_3 and γ_4 are the *skewness* and *kurtosis* of the returns of the chosen strategy.
- ϕ is the cdf of the standard Gaussian law and γ is the Euler-Mascheroni constant.
- The index m refers to the number of strategy trials.

If t defined above is below a certain threshold (e.g., 0.95), then the SR **cannot be deemed significant: the best strategy is not outstanding**.

Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

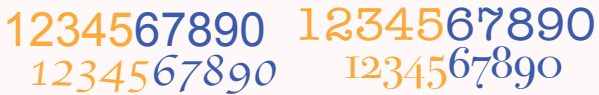
Bias

A first example from computer vision

▷ Imagine a model trained on a dataset of cows eating grass and camels in the desert.
▷ What would happen if we tried to predict the animal when providing an image of a cow in the desert? ($\rightarrow$ Camel !?)

A second example from computer vision

▷ Imagine a model trained on a dataset of digits:



▷ What would happen if we tried to predict the number if we provided an orange 6? ($\rightarrow$ 5 !?)

Invariance

Bottomline (1/2)

What matters in ML is **invariance**, i.e. things that do not change (or change slowly).
▷ In our examples, invariance is what is common to all cows and each digit, no matter the **context** we see them in (landscape, color).
▷ This is a major in finance because invariance (and causality) is very hard to reveal. The markets move fast and absence of arbitrage ensure that past winning strategies don't work in the future.

The only constant in life is change. - Heraclitus

This is well-known in machine learning:
→ **concept drift**: change in the link between y and X.
→ **covariate shift**: change in the distribution of X

Fairness

Bottomline (2/2)

Another important takeway is that algorithms can only remember what they have been shown.

▷ In **asset management**, this is harmful because the loss in accuracy can generate false positives (and lead to negative relative performance)

▷ In **credit scoring**, this is a more important issue, because algorithms are likely to enforce status quo and discrimination: they learn and reproduce patterns. This is likely to confirm biases (gender or race-based).[2]

[2] One example: in 2019, tech entrepreneur David Heinemeier Hansson complained that the Apple Card gave him 20 times the credit limit that his wife got.

Sustainable investing

It has become mainstream - and imperative

The most obvious question for the course is: can we leverage ESG to craft performing portfolios. Imagine a panel-type model:

$$r_{t+1,n} = f(\text{esg}_{t,n}) + e_{t+1,n}$$

Does it "work"' well? Likely answer $\rightarrow$ **no**. At least: not unconditionally. There might be pockets of predictability for ESG, but not over long time scales.

Sustainable investing should not be thought of through the lens of profitability.
Good news though: if returns and ESG are not linked, it's costless to invest in green assets!

Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

Evolving context

Technological arms race

- more (**alternative**) data: credit card logs, satellite imagery, sentiment. When and how is it useful?
- more **complex models** (with the risk of overfitting): million weight NN, thousand tree aggregates, large ensembles, etc. $\rightarrow$ **double descent** results suggest the risk is not always high.
- **reinforcement learning**? (some papers exist, the economic framing is not always straightforward).
- **unsupervised learning?**: finding patterns without labels! Hard, but generalisation gains are maybe worth it.
- **other uses**: deep learning for option pricing?
- **Gen AI**: a tool (or teammate), but definitely not 100% of the solution.

Agenda

1. Support vector machines
2. Ensemble learning
3. Interpretability
4. Backtesting issues
5. Bias, invariance, fairness
6. What's next?
7. Key takeways

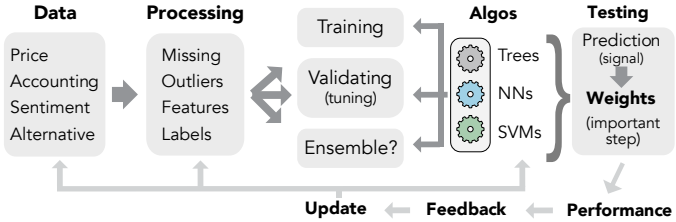
Session wrap-up

There are infinitely many possible extensions

- each step of the process is subject to many **degrees of freedom**
- the **zoo of forecasting tools** is densely populated: not all are equal; it's important to know what they do and whether or not they are relevant
- in doubt: **agnostic combinations** (EW) are good ideas; optimised ensembles are risky
- beware of interpretability: local vs global representations are **incomplete**. Also, given the number of points, local points are probably intractable (or hardly manageable to the least)
- backtesting is deceitful: even when carried out-of-sample, the best strategy will always **overestimate** its profitability.

Course wrap-up: closing the loop

- **processing** information into investment decisions requires a good **signal**
- ML tools are here to extract the signal from the data
- nonetheless, the ratio **signal**/**noise** is extremely small
- hence, extra care must be given at each step of the process:
  1. the (sound) **economic foundation** of the goal
  2. the **choice** and **engineering** of features/labels (+ consistency between the two)
  3. the coherence of **hyper-parameter** tuning
  4. the **translation** of the signal into portfolio weights



Thank you for your attention

Don’t forget to provide **feedback** on the course!

Any questions?