Machine Learning in Finance

–Presession–

PD Dr. Volker Britz
Center of Economic Research at ETH Zürich

Welcome

Welcome to the ETH Risk Center!

PD Dr. Volker Britz
Course Coordination "Machine Learning in Finance"
Center of Economic Research at ETH Zürich
Zürichbergstrasse 18
vbritz@ethz.ch

Program for this Talk

1. What is (supervised) Machine Learning?
2. What are the key concepts in ML?
3. What kind of ML algorithms exist?
4. How do we evaluate the quality of ML algorithms?
5. How do you get started?
6. What are some common pitfalls?
7. What are examples in industry and academia?

What is (supervised) Machine Learning?

An algorithm learns (from training data) how a variable of interest (class, labels) is related to other observable variables (features). When confronted with new data (test data), it then predicts labels from features.

Classical example: The Titanic dataset.

- *Class / Labels:* Did the passenger survive the Titanic disaster?
- *Features:* Man or Woman? Age? 1st/2nd/3rd class? ...
- *Splitting Training / Test data:* Random choice of holdout data, or Cross-Validation

You can take a closer look here: https://www.kaggle.com/c/titanic

Supervised Machine Learning: Severity of Road Accidents

- *Labels:* Each car accident is "slight" or "serious."
- *Features:* When was the accident? Which day? What time? Which speed limit? Urban or Rural?

The Road Accidents data and many other datasets can be found on Kaggle: https://www.kaggle.com/tsiaras/uk-road-safety-accidents-and-vehicles

To improve predictions, work (at least) on the following:

1. Data collection and relevance
2. Data preparation
3. ML algorithm

Point of Caution: Better algorithms or better data?

Example: Try to predict who is most at risk from COVID.

Possible features: Number of contacts, commuting, home office, household size, mobility etc.

Missing information: What is patients' profession? How much do they use public transport? Do they have children in school? How much home office do they do?

$\rightarrow$ In this example, probably (my guess!) better data offers more opportunities than a sophisticated algorithm: Data collection, cleaning, and preparation is of the essence!

Some Examples of ML Algorithms

- *Nearest Neighbors*
  1. Use a metric for feature similarity b/w instances
  2. To make prediction:
     Look for the 5 most similar instances
- *Tree-Based Algorithms*
  - Which feature best separates classes?
  - Use that feature to create branches.
  - Repeat this process within each branch.
- *Neural Network*
  - Data sent through *layers* of neurons.
  - Iterative incremental changes to parameters to boost accuracy.
  - Sometimes: Use of different filters to detect specific features.

A Simple Notion of Accuracy

"Which share of predictions is correct?"
Basic answer: "If 830 out of 1000 cases are predicted correctly, then we have an accuracy of 0.83."

Some (potential) issues:

- Distinguish between accuracy on training and test sets.
  $\rightarrow$ Detect *overfitting*.
- Data relevance and quality affect what "good" performance is.
- Relative class size distorts accuracy measures.

Class Imbalance

Class Imbalance: In many applications, one has a small "positive" class and a (much) larger "negative" class:

- Most people screened for a disease are not sick.
- Most payments / transactions are not fraudulent.
- Most road accidents are not serious.

Predicting only negatives trivially accomplishes high accuracy.

Remedies for Class Imbalance

1. Downsampling the majority class
2. Upsampling the minority class
3. Smarter accuracy measures

In order to describe accuracy:

- Which share of positive predictions is accurate? ("Precision")
- Which share of positives is predicted accurately? ("Recall")
- Give greater weight to accurate predictions in minority class
- *Cohen's Kappa* adjusts directly for "luck" in prediction
- Visualize performance in a *confusion matrix*

Working with ML Algorithms

Good news: It's easy to get started!

All you need:

- Some basic coding skill, ideally in Python
- Some knowledge of an ML library like Keras
- And of course: Data!

(Some) challenges:

- Collect good quality data
- Define suitable questions and engineer features
- Choose suitable algorithms
- Assess performance adequately

Two Fundamental Problems

"Why can you not predict the financial crisis?"
$\rightarrow$ "Why can you not predict the global pandemic?"

Fundamental problems with "predictions":

- Predictions may fail when behavior is adapted to them.
- If a machine learns from data on the last five years, it cannot and will not predict the once-in-a-lifetime event that is around the corner.
  $\rightarrow$ Although in principle, ML techniques can be applied to time series data.

Use Cases in Political Economy

- Treating word use as features, use ML to classify news stories as CNN or Fox News.
- Using voting behavior of members of Congress, classify them as Democrat or Republican.

$\rightarrow$ Measure political polarization and media bias by the accuracy with which such predictions are possible.

See for instance work in the Law, Economics, and Data Science group at ETH: https://lawecondata.ethz.ch/research/workingpapers.html

Interpretability of ML Predictions

Compared to plain regression analysis, sophisticated ML algorithms detect more complex pattern in data.

Drawback: Results are hard to interpret.
("What drives the prediction?")

"If it works, why do we care how it works?"

- *Industry:* Interpretability may be ethically or legally required (scoring, credit decisions, discrimination)
- *Academia:* Understanding things is the ultimate goal!

Game-theoretic Approach to Interpretability

Computing *Shapley values* reveals how much each feature contributes to classification of a particular instance.

Example: Use voting behavior to classify House members as Democrats or Republicans.

$\rightarrow$ For each House member, how did each feature affect prediction?

Note: Feature importance may differ for each instance!
$\rightarrow$ Contrary to the interpretation of regression coefficients, for example.

Thank you for your attention!

Now over to Sebastian Becker for...

Coding in Python and Keras!