Stock Embeddings

Learning Distributed Representations for Financial Assets

(Dolphin et al. (2022))

http://lellep.xyz/blog/reading-group-materials.html

Paper presentation for

Statistical Physics Reading Group

2023-05-03

The University of Edinburgh

By:

Martin Lellep

(http://lellep.xyz)

Overview

- ML primer
- Introduction
- Maths
- Implementation
- Use cases

ML primer

(ML := machine learning)

This is a ML project!

What you need for ML project:

- Problem to solve
- Data
- Algorithm:
  - Prediction algorithm
  - Loss fct
  - Optimiser

Introduction

Problem to solve:

Turn an asset a_i like "Apple Inc" into a vector $\underline{e}_{Apple} \in \mathbb{R}^N$. Generally, $\underline{e}_{a_i} \in \mathbb{R}^N$ should be dense, i.e. all its entries should be non-zero & thereby be used.
E.g. $\underline{e}_{a_i}$ should not be a unit vector (which are also called one-hot encodings).

Data:

Prices of 500 largest US companies against time, i.e. $P_t^{a_i}$.

Specifically, this set of data is called S&P 500.

Inspiration

Comes from "Natural language processing" (NLP), e.g. "continuous bag of words" approach.

There, words are turned into vectors ("word2vec" by [Mikolov 2013], [Pennington 2014]),

e.g. $\underline{e}_{hi} = (0.7, 0.3)^T$ &

$\underline{e}_{hello} = (0.8, 0.35)^T$ &

$\underline{e}_{bye} = (-0.1, 0.5)^T$.

Useful for similarity,

$$sim(\underline{e}_i, \underline{e}_j) = \frac{\underline{e}_i \cdot \underline{e}_j}{\|\underline{e}_i\| \cdot \|\underline{e}_j\|}$$ ~ cosine of angle between $\underline{e}_i$ & $\underline{e}_j$

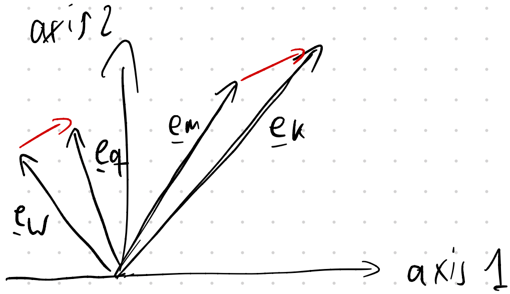
so that

$sim(\underline{e}_{hi}, \underline{e}_{hello}) = 0.992$ ← similar

$sim(\underline{e}_{hi}, \underline{e}_{bye}) = 0.206$ ← not so similar

& arithmetics [Mikolov 2013]:

$$\underline{e}_{King} - \underline{e}_{man} + \underline{e}_{woman} \approx \underline{e}_{queen}$$



Also useful in many downstream ML tasks that operate on language.

E.g. Transformer models that power large language models like GPT models, & hence also ChatGPT, use word (or to be precise: token) embeddings [Vaswani (2017) NIPS].

Maths

2023-04-28

Framework & prediction algo

- $\mathcal{U} = \{ a_1, \dots, a_{|\mathcal{U}|} \}$ ~ asset universe

- $\underline{p}_{a_i} = \{ p_0^{a_i}, \dots, p_T^{a_i} \}$ ~ prices

  $\rightarrow \underline{r}_{a_i} = \{ r_1^{a_i}, \dots, r_T^{a_i} \}$ w/

  $$r_t^{a_i} = \frac{p_t^{a_i} - p_{t-1}^{a_i}}{p_{t-1}^{a_i}} \quad \sim \text{returns}$$

- Construct context dataset by selecting $\forall a_i$ & t $S(a_i, t)$ the C assets a_j that minimise $| r_t^{a_i} - r_t^{a_j} |$.

  $\rightarrow$ one obtains dataset of size $|\mathcal{U}| \times T$

- $\underline{x}_{a_i}$ ~ one-hot encoded vector, i.e. in components $(\underline{x}_{a_i})_j = \delta_{ij}$

- Define embeddings matrix

$$\underline{\underline{W}} = \begin{pmatrix} - & e_1 & - \\ & \vdots & \\ - & e_{|u|} & - \end{pmatrix} \in \mathbb{R}^{|u| \times N}$$

- Given $S(a_i, t)$, compute hidden state as mean embedding

$$\underline{h} = \underline{\underline{W}}^T \left(\frac{1}{C} \sum_{j=1}^{C} \underline{x}_{a_{j_i}} \right) \text{ with } a_{j_i} \in S(a_i, t)$$

- Lastly compute neural network (NN) prediction as

$$p(\text{target} \mid \text{context}) = \text{softmax}(\underline{\underline{W}}\, \underline{h})$$

with $(\text{softmax}(\underline{z}))_i = \dfrac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for $i = 1, \dots, K$

& $\underline{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$, i.e. softmax produces a discrete probability distribution.

Two possible noise reduction strategies:

- Noise red 1: different weighting
- Noise red. 2: Exclude $S(a_i, t)$ if $r_t^{a_i}$ statistically common

Loss fct

Loss function defines optimisation objective. Here we want to predict target asset given context to then obtain $\underline{\underline{W}}$ which stores the embeddings.
Consider a training sample $S(a_i, t)$ of target asset a_i at time t.
We now compare the predicted probability over assets from the NN,

$$\underbrace{p_p}_{\text{predicted}} \left(a_j \mid S(a_i, t), \underbrace{\theta}_{\text{NN parameters } \underline{\underline{W}} \text{ to optimise}} \right) \in \mathbb{R}^{|u|}$$

to the ground truth,

$$\underbrace{p_t}_{\text{true}} \left(a_j \mid S(a_i, t) \right) = \left(\delta_{ik} \right)_{k=1,\dots,|u|} \in \mathbb{R}^{|u|} .$$

The loss function is chosen to be the categorical crossentropy so that the loss of a single sample is computed as:

$$l(p_t, p_p) = -\sum_a p_t(a) \cdot \log p_p(a)$$

Motivation behind categorical crossentropy loss: Use Kullback-Leibler (KL) divergence

$$KL(p_t, p_p) = \sum_a p_t(a) \log\left(\frac{p_t}{p_p}\right)$$

$\log\left(\frac{M}{N}\right) = \log M - \log N$

$$= \sum_a p_t(a) \cdot [\log p_t - \log p_p]$$

$$= \underbrace{\sum_a p_t(a) \log p_t(a)}_{= \text{const} = 0} - \underbrace{\sum_a p_t(a) \log p_p(a)}_{=: H(p_t, p_p)}$$

$\Rightarrow$ omitted for optimisation $\underline{\underline{W}}$.

$- H(p_t)$

Optimisation

Optimise $\underline{\underline{W}}$ to minimise loss to approach p_t with p_e, which is actually only a proxy to obtain $\underline{e}_{a_i}$ $\forall a_i$.

Basic idea is called stochastic gradient descent (SGD). Consider loss over batch:

$$L(\theta) = \frac{1}{b} \sum_{i=1}^{b} \ell \left(p_t(a_j | D_i), p_p(a_j | D_i, \theta) \right)$$

w/ $b \hat{=}$ batch size, which is a hyper-parameter that is to be tuned manually, and D_i the i-th sample from data batch D. Side note: Joining all batches, one obtains the full training dataset $\mathcal{D}$,

$$\mathcal{D} = [\underbrace{S(a_1, t_1), S(a_1, t_2), \ldots}_{D}, \ldots, \underbrace{S(a_{|A|}, T)}_{D}].$$

Lastly, the optimisation step is performed as

$$\theta_{new} = \theta_{old} - \eta \nabla_\theta L(\theta),$$

w/ η as learning rate that is yet another hyperparameter that is to be tuned manually.

Side note 1: There are much more sophisticated optimisers than SGD. Most people use those more sophisticated ones.

Side note 2: An algorithm called "back propagation" is used to compute $\nabla_\theta L(\theta)$ for NNs of almost arbitrary topology.

Implementation

Implemented to compute one batch, that is then used for optimisation step, simultaneously.

Use cases & benchmarks

Use embeddings as measure for similarity between assets; traditionally, correlations are used for that.

Quality of embeddings:

- Neighbours: Table I
- Arithmetics: Table II
- Visualisation: Fig. 3

Potential use:

- Construct hedged portfolio: Fig. 4

(Tables & Figs refer to Dolphin et al. (2022).)