Factor models via Autoencoders

A clever way of using Neural Networks to solve a familiar but important problem in Finance was proposed by [Gu, Kelly, and Xiu, 2019 (https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3335536)](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3335536).

It is an extension of the Factor Model framework of Finance, combined with the tools of dimensionality reduction (to find the factors) of Deep Learning: the Autoencoder.

You can find code (https://github.com/stefan-jansen/machine-learning-for-trading/blob/main/20_autoencoders_for_conditional_risk_factors/06_conditional_autoenc for this model as part of the excellent book by Stefan Jansen (https://github.com/stefan-jansen/machine-learning-for-trading/blob/main/20_autoencoders_for_conditional_risk_factors/06_conditional_autoenc

- Github (https://github.com/stefan-jansen/machine-learning-for-trading)
- In order to run the code notebook, you first need to run a notebook for data prepara jansen/machine-learning-for-trading/blob/main/20_autoencoders_for_conditional_risk_factors/05_conditional_a
    - This notebook relies on files created by notebooks from earlier chapters of
    - So, if you want to run the code, you have a lot of preparatory work ahead of
    - Try to take away the ideas and the coding

Factor Model review

We will begin with a quick review/introduction to Factor Models in Finance.

First, some necessary notation:

- $\mathbf{r}_s^{(d)}$: Return of ticker s on day d.
- $\hat{\mathbf{r}}_s^{(d)}$: approximation of $\mathbf{r}_s^{(d)}$
- n_{tickers}: **large** number of tickers
- n_{dates} number of dates
- n_{factors}: **small** number of factors: independent variables (features) in our approximation
- Returns matrix $\mathbf{R}$ indexed by *date*
    - $\mathbf{R} : (n_{\text{dates}} \times n_{\text{tickers}})$
    - $||\mathbf{R}^{(d)}|| = n_{\text{tickers}}$
        - $\mathbf{R}^{(d)}$ is vector of returns for each of the n_{tickers} on date d
- $\mathbf{r}$ will denote a vector of single day returns: $\mathbf{R}^{(d)}$ for some date d

Notation summary

term	meaning		
s	ticker		
n_{tickers}	number of tickers		
d	date		
n_{dates}	number of dates		
n_{chars}	number of characteristics per ticker		
m	number of examples		
	$m = n_{\text{dates}}$		
i	index of example		
	There will be one example per date, so we use i and d interchangeably.		
$[\mathbf{X^{(i)}}, \mathbf{R^{(i)}}]$	example i		
	$	\X^\ip	= (\ntickers \times \nchars)$
	$	\R^\ip	= \ntickers$
$\mathbf{X}_s^{(d)}$	vector of ticker s's characteristics on day d		
	$	\X^\dp_s	= \nchars$

Note

The paper actually seeks to predict $\hat{\mathbf{r}}_s^{(d+1)}$ (forward return) rather than approximate the current return $\hat{\mathbf{r}}_s^{(d)}$.

We will present this as an approximation problem as opposed to a prediction problem for simplicity of presentation (i.e., to include PCA as a model).

A **factor model** seeks to approximate/explain the return of a *number* of tickers in terms of common "factors" $\mathbf{F}$

- $\mathbf{F} : (n_{\text{dates}} \times n_{\text{factors}})$

$$
\begin{aligned}
\mathbf{R}_1^{(d)} &= \beta_1^{(d)} \cdot \mathbf{F}^{(d)} + \epsilon_1 \\
\vdots \\
\mathbf{R}_{n_{\text{tickers}}}^{(d)} &= \beta_{n_{\text{tickers}}}^{(d)} \cdot \mathbf{F}^{(d)} + \epsilon_{n_{\text{tickers}}}
\end{aligned}
$$

There are several ways to create a factor model.

Pre-defined factors, solve for sensitivities

First: supposed $\mathbf{F}$ is given

- For each date d, returns for: market, several industries, large/small cap
- Solve for β_s, for each s
    - n_{tickers} separate Linear Regression models: $\langle \mathbf{X}^{(d)}, \mathbf{y}^{(d)} \rangle = \langle \mathbf{r}_s^{(d)}, \mathbf{F}^{(d)} \rangle$
    - Regression of time-series of a ticker's return agains a time-series of Factor returns
    - Solve for β_s

Pre-defined sensitivities, solve for factors

Alternately: suppose β is given

- For each ticker s, sensitivity of s to β_j
- Solve for $\mathbf{F}^{(d)}$, for each d
    - n_{dates} separate Linear Regression models $\langle \mathbf{X}^{(s)}, \mathbf{y}^{(s)} \rangle = \langle \beta_s, \mathbf{r}_s^{(d)} \rangle$
    - Regression of *cross-section* of tickers returns against a cross-section of ticker sensitivities
    - Solve for $\mathbf{F}^{(d)}$

Solve for sensitivities and factors: PCA

Yet another possibility: solve for β and $\mathbf{F}$ *simulataneoulsy*.

Recall Principal Components

- Representing $\mathbf{X}$ (with "standard" basis vectors) via an *alternate basis* $\mathbf{V}$
$$\mathbf{X} = \tilde{\mathbf{X}}\mathbf{V}^T$$

In this case without dimensionality reduction:
$$\mathbf{R} = \tilde{\mathbf{R}}V^T$$

where

$$\begin{aligned}
\mathbf{R}, \tilde{\mathbf{R}} &: (n_{\text{dates}} \times n_{\text{tickers}}) \\
\mathbf{V}^T &: (n_{\text{tickers}} \times n_{\text{tickers}})
\end{aligned}$$

With dimensionality reduced from n_{tickers} to n_{factors}

$$\mathbf{R} = \mathbf{F}\,\beta^T$$

- $\mathbf{F}^T$: $(n_{\text{dates}} \times n_{\text{factors}})$
    - is $\tilde{\mathbf{R}}$ with columns eliminated b/c of dimensionality reduction
- β^T : $(n_{\text{factors}} \times n_{\text{tickers}})$
    - so $\beta^{(s)}$ are sensitivities of s to factors
- Solve for $\mathbf{F}, \beta$ simultaneously

The daily observation of n_{tickers} returns $\mathbf{R}^{(d)}$ is replaced by n_{factors} returns $\mathbf{F}^{(d)}$

This paper

This paper will create a factor model that

- Solve for $\mathbf{F}, \beta$ simultaneously
    - like PCA
- **But** where $\mathbf{F}$ and β are defined by Neural Networks

Autoencoder

The paper refers to the model as a kind of Autoencoder.

Let's review the topic.

Training examples $\langle \mathbf{X}^{(d)}, \mathbf{y}^{(d)} \rangle = \langle \mathbf{R}^{(d)}, \mathbf{R}^{(d)} \rangle$

No obvious form as factor model

- $\mathbf{R}^{(d)} = \mathbf{r}$
    - mapped by Encoder to latent $\mathbf{z}$ (of length n_{factors})
    - latent $\mathbf{z}$ mapped to $\mathbf{r}$ by Decoder

Imagine instead creating an "Autoencoder" that worked as follows

- Maps $\mathbf{R}^{(d)}$ to $\beta^{(d)}$
    - sensitivity of each of the n_{tickers} on day d to day d returns of n_{factors} $\mathbf{F}^{(d)}$
- Maps $\mathbf{R}^{(d)}$ to the day d returns of n_{factors} $\mathbf{F}^{(d)}$
- Outputting $\mathbf{y}^{(d)} = \beta^{(d)} \, \mathbf{F}^{(d)}$

It acts as an Autoencoder in the senses that the Training examples
$\langle \mathbf{X}^{(d)}, \mathbf{y}^{(d)} \rangle = \langle \mathbf{R}^{(d)}, \mathbf{R}^{(d)} \rangle$

- But constrains $\hat{\mathbf{y}}^{(d)} = \hat{\mathbf{R}}^{(d)}$ to the form $\hat{\mathbf{R}}^{(d)} = \beta^{(d)} \, \mathbf{F}^{(d)}$

This model solves for $\beta^{(d)}, \mathbf{F}^{(d)}$ simultaneously

- almost what PCA does **but**, in PCA, β does not vary by day
- this model: the beta of a ticker s to a factor j changes by day d !

This paper goes one step further than the standard Autoencoder

- Standard Autoencoder maps $\mathbf{R}^{(d)}$ to $\beta^{(d)}$
- This paper allows $n_{\text{chars}} \geq 1$ daily *characteristics* $\mathbf{X}^{(d)}$ to map to $\beta^{(d)}$
    - one characteristic may be $\mathbf{R}^{(d)}$

$$\beta_s^{(d)} = \mathrm{NN}(\mathbf{X}_s^{(d)}; \mathbf{W}_\beta)$$

- $\beta_s^{(d)}$
  - parameterized by weights $\mathbf{W}_\beta$
  - is only a function of the characteristics of s
  - **not** a function of *other* ticker s' characteristics $\mathbf{X}_{s'}$, as in PCA
- $\beta_s^{(d)}$ share the same weights $\mathbf{W}_\beta$ for all s, d
  - unlike fixed factor, solve for β_s
    - different for each s
    - same for each day d

This model: nothing pre-defined, solve for sensitivities and factors

- Simultaneously solve for $\beta_s^{(d)}$ and $\mathbf{F}^{(d)}$
  - $\beta_s^{(d)}$ constrained:
$$\beta_s^{(d)} = \text{Dense}\,(n_{\text{factors}})(\mathbf{X}_s^{(d)})$$
    - combination of ticker-specific, time-varying characteristics $\mathbf{X}_s^{(d)}$
    - we solve for the *combining weights*
      - shared by all tickers and dates
  - $\mathbf{F}^{(d)}$ constrained
$$\mathbf{F}^{(d)} = \text{Dense}\,(n_{\text{factors}})(\mathbf{R}^{(d)})$$
    - combination of time-varying *raw returns* $\mathbf{R}^{(d)}$
    - we solve for *combining weights*
      - shared by all dates

This paper

- $\mathbf{r}^{(d)} = \beta^{(d)} * \mathbf{r}^{(d)}$
  - $\mathbf{r}^{(d)}$ shape is $(n_{\text{tickers}} \times 1)$
  - β shape is $(n_{\text{tickers}} \times n_{\text{factors}})$
  - $\mathbf{r}^{(d)}$ shape is $(n_{\text{factors}} \times 1)$

- Solve simultaneously for $\beta^{(d)}, \mathbf{r}^{(d)}$
  where $\beta_s^{(d)} = f(\mathbf{X}_s^{(d)})$

  - $\beta_s^{(d)}$ is only a function of the characteristics of s
  - **not** $f(\mathbf{r}^{(d)})$: the simultaneous returns of *other* s' as in PCA

  - $\beta_s^{(d)}$ share the same $\mathbf{W}_\beta$ for all s, d

    - unlike fixed factor, solve for β_s
      - different for each s
      - same for each day d

  and where

  $\mathbf{r}^{(d)} = f(\mathbf{r}^{(d)})$ for f fixed over all d

    - like PCA

Input side of network

Input $\mathbf{X}$

$\mathbf{X} : (n_{\text{dates}} \times n_{\text{tickers}} \times n_{\text{chars}})$

$||\mathbf{X}^{|}| = (n_{\text{tickers}} \times n_{\text{chars}})$

- one example per date
- example shape is $n_{\text{tickers}} \times n_{\text{chars}}$

Dense β

- `Dense` (n_{factors})
  - `Dense(`n_{factors}`)` : $(n_{\text{tickers}} \times n_{\text{chars}}) \mapsto (n_{\text{tickers}} \times n_{\text{factors}})$
  - threads over ticker dimension ([see (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense)](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense))
    - tickers share same weights
    - single `Dense(`n_{factors}`)` **not** n_{tickers} copies of `Dense`(n_{factors})

- $\mathbf{W}_\beta : (n_{\text{factors}} \times n_{\text{chars}})$
  - same across all d, s
  - $W_\beta^{(d)} = W_\beta^{(d')}$ like any other training (same weight for every example)
  - $W_{\beta,(s)}^{(d)} = W_{\beta,(s')}^{(d)}$: transformation of characteristics to beta *independent* of ticker
  - hence, size of $\mathbf{W}_\beta$ is $(n_{\text{factors}} \times n_{\text{chars}})$

$$\beta^{(d)} = \mathrm{Dense}\,(n_{\text{factors}})(\mathbf{X}^{(d)})$$
$$||\beta^{(d)}|| = (n_{\text{tickers}} \times n_{\text{factors}})$$

Factor side of network

Input $\mathbf{R}$

$\mathbf{R} : (n_{\text{dates}} \times n_{\text{tickers}})$

$||\mathbf{R}^{(d)}|| = (n_{\text{tickers}})$

- one set of returns per date

Dense δ (factor)

- `Dense` (n_{factors})
  - `Dense(` n_{factors} $)$: $n_{\text{tickers}} \mapsto n_{\text{factors}}$
- $\mathbf{W}_f : (n_{\text{factors}} \times n_{\text{tickers}})$
  - same across all d, s
  - $W_f^{(d)} = W_f^{(d')}$ like any other training (same weight for every example)
  - $W^d pf$: transformation of ticker returns to factor returns

$$
\begin{gathered}
\mathbf{F}^{(d)} = \mathrm{Dense}\,(n_{\text{factors}})(\mathbf{R}^{(d)}) \\
||\mathbf{F}^{(d)}|| = n_{\text{factors}}
\end{gathered}
$$

Dot

$$\hat{\mathbf{r}}^{(d)} = \beta^{(d)} \cdot \mathbf{F}^{(d)}$$

- Dot product threads over factor dimension

$$||\hat{\mathbf{r}}^{(d)}|| = n_{\text{tickers}}$$

Loss

Let $\mathcal{L}_{(s)}^{(d)}$ denote error of ticker s on day d.

$$\mathcal{L}_{(s)}^{(d)} = \mathbf{r}_s^{(d)} - \hat{\mathbf{r}}_s^{(d)}$$

or perhaps

$$\mathcal{L}_{(s)}^{(d)} = \mathbf{r}_s^{(d+1)} - \hat{\mathbf{r}}_s^{(d)}$$

- $\mathcal{L}^{(d)}$ is the loss of example d
    - this loss has n_{tickers} sub-components
    - This appears in example $i = d : \mathbf{X}^{(d)}$
    - $\mathcal{L}^{(\mathbf{i})} = \mathcal{L}^{(d)} = \sum_s \mathcal{L}_{(s)}^{(d)}$
- This is different than the loss $\mathcal{L}'$ for the case where an example is a single ticker on a single day
    - $m' = n_{\text{dates}} * n_{\text{tickers}}$ examples in this case
    - $\mathcal{L}'^{(\mathbf{i})} = \mathcal{L}_{(s)}^{(d)}$