Domain-specific networks: *Sequence models*

Dr. Eran Raviv

July 2022

Agenda for today

- Recap from last time: what have we learned?
- Domain specific networks: when sequence matters.
- Embeddings

Domain-specific networks: Recurrent Neural Networks (RNN)

Why do we need RNNs?

- Because we have many situations where sequence plays a role.
- Markov models are not enough.
- To continue using "just" deep neural networks would mean we underutilize specific, relevant, domain information.

Recurrent Neural Networks (RNN)

- Domain-specific class of neural networks.
- A-priori unknown number of inputs (words in a sentence).
- When sequence matters - meaningful sequential relation.

Example: order is important for making sense in NLP related tasks

- This course is actually quite good, not as horrible as I thought before I registered. ☺

- This course is actually quite horrible, not as good as I thought before I registered. ☹

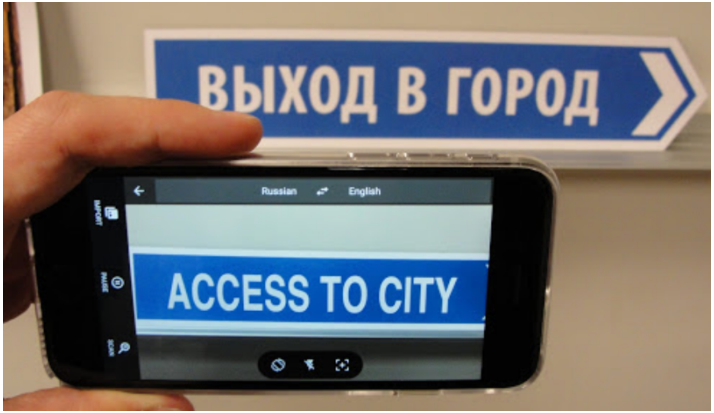
Recurrent Neural Network - variations

- One to one (standard)
- One to many
- Many to one
- Many to many

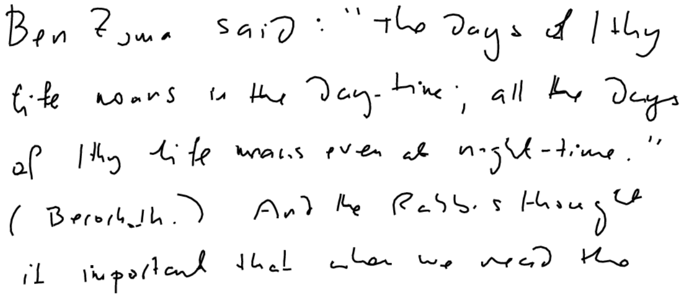
Recurrent Neural Network - examples

- Time series (many to one) - stocks, climate, demographics, more.
- Image captioning (one to many: image to sentence).
- Music generation (many to one: amplitudes to next note).
- Text - entity detection, sentiment classification (many to one).
- Speech - translation, recognition (many to many).
- Chatbots (many to many).

RNN - example application



RNN - example application



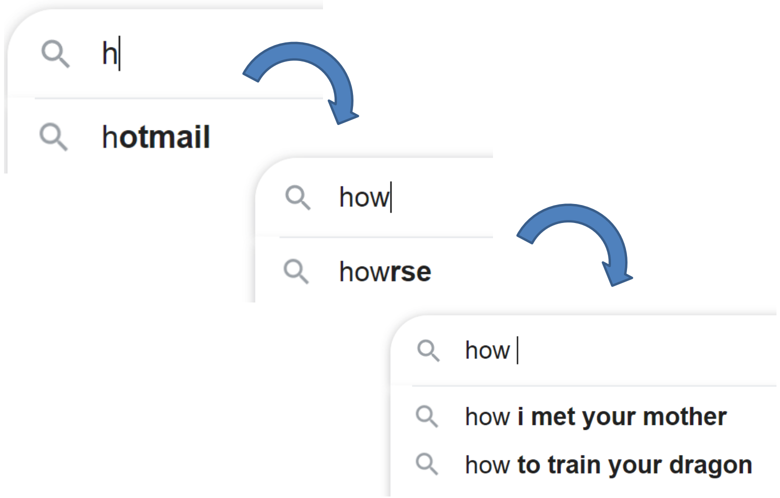
Ben Zoma said: "the days of 1thy
life means in the day-time; all the days
of 1thy life means even at night-time."
(Berochoth.) And the Rabbis thought
it important that when we read the

(a)

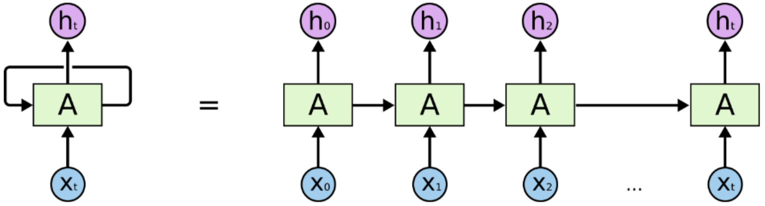
```
Ben Zoma said: "The days of 1thy
life means in the day-time; all the days
of 1thy life means even at night-time ."
(Berochoth .) And the Rabbis thought
it important that when we read the
```

(b)

RNN - example application


h
hotmail
how
howrse
how
how i met your mother
how to train your dragon

First look



- We don't need to define the length of the sequence beforehand.
- Theoretically at least, $\mathbf{h}_t$ can provide output based on all inputs up to time t.

The order in which we will explain it:

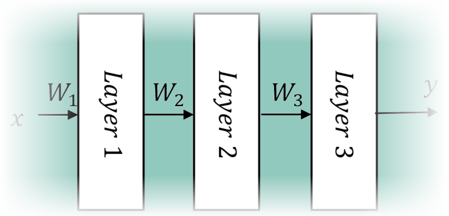
- Moving average
- Autoregression
- Nonlinear autoregression
- Vanilla neural network
- RNN

More formally

- $\bar{h}_t = \tanh\left(W_{xh}\bar{x}_t + W_{hh}\bar{h}_{t-1}\right)$
- $\bar{y}_t = W_{hy}\bar{h}_t$
- As a side note: we can also concatenate W_{xh} and W_{hh}, but it is useful to have this separation.

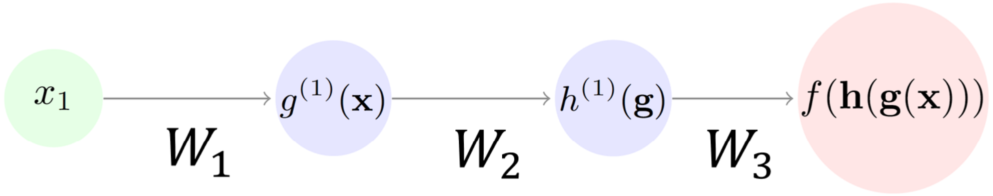
Observe that

- RNNs are not very different from architectures you have seen already:



- However, weights are **shared**. Not each prediction (or lag) is being sent forward with a different weight, as would otherwise be.

Compare with the "classic" neural net


$$\begin{tikzcd} x_1 \arrow[r, "W_1"'] & g^{(1)}(\mathbf{x}) \arrow[r, "W_2"'] & h^{(1)}(\mathbf{g}) \arrow[r, "W_3"'] & f(\mathbf{h}(\mathbf{g}(\mathbf{x}))) \end{tikzcd}$$

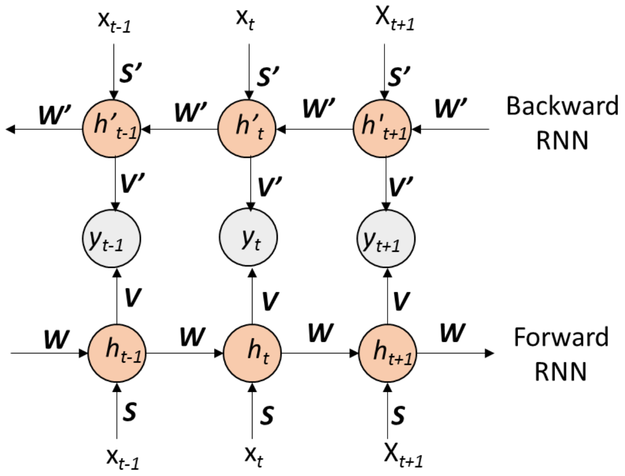
Why would we want to share weights?

- Large changes in prices of financial assets tend to cluster together. Whether it is 1929, or 2008.
- They word *am* will never appear directly after the word *they* (so there is no point in teaching the machine about this grammer rule each time anew).
- Sharing weights economizes our training $\Rightarrow$ increased efficacy. This is basically a strong form of regularization.

Bidriectional RNN

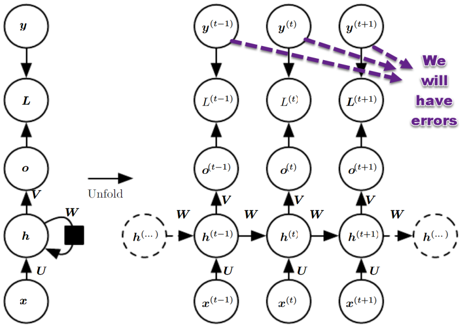
- Proposed by Schuster and Paliwal (1997)
- Pulling data from "the future" is not always a big *no-no*. It can be very useful, especially for language modeling.
- Putting two independent RNNs together.
- The output is concatenated at each time step.
- Providing context on both sides.

Bidriectional RNN - schematic visualization



Training RNNs

What does backpropogation through time (BPTT) means?

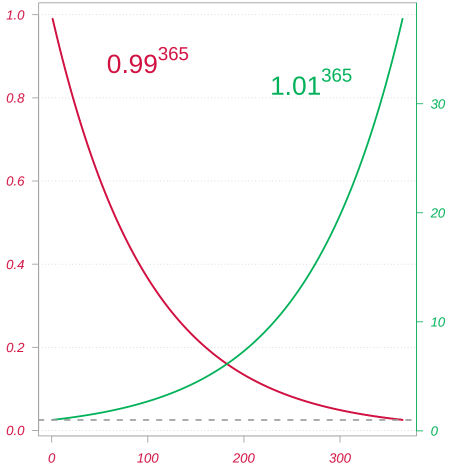


- Assume, for now, that $h^{(\cdot)}$ and $o^{(\cdot)}$ are the same.
- L is for the loss function, and (unusually I admit) $y^{(\cdot)}$ is for the error.
- What is $x^{(\cdot)}$ in this image?
- What is $h^{(\cdot)}$ in this image?

Training RNNs

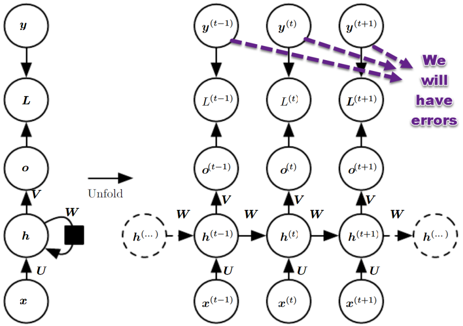
- *Almost* as usual.
- But because weights are shared, the eventual update is simply the sum of the gradients of each of the "copies".
- We have to distinguish this kind of optimization from the "usual" way. So we typically describe this numerical procedure as (BPTT).

Inside of every opportunity lies a problem..



This is also true in matrix multiplication.

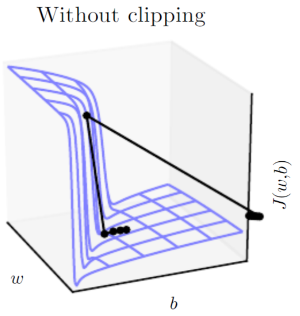
Vanishing gradient problem rears its ugly head

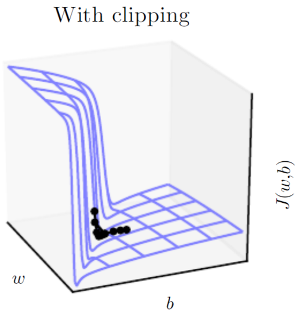


- Where, exactly, is the problem?
- We have parameters "buried" inside $h^{(\cdot)}$!

What about exploding gradients?

- The problem is that some parameters' update can severely "overshoot"; destabilizing the optimization.
- An intuitive solution: gradient clipping.





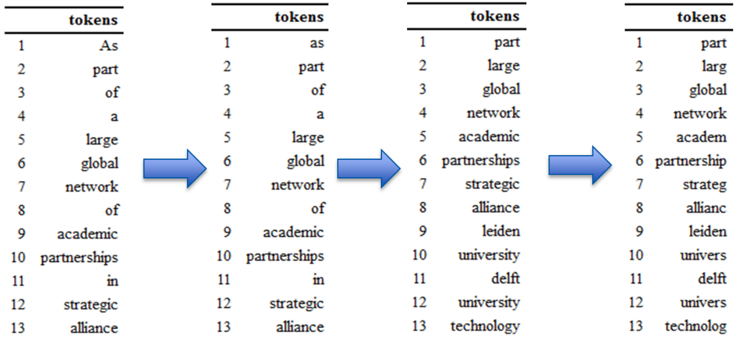
How to clip?

- Using a threshold (think winsorization, element-wise).
- Using rescaling (think rescaling $\ddot\smile$)
- E.g. using the cross sectional L_2 norm of all gradients.
- No strong dominance of neither option in experimentation.

Encoding

A bit about word representation

As part of a large global network of academic partnerships, in strategic alliance with Leiden University and Delft University of Technology and in a unique collaboration with city and port, the dynamic city of Rotterdam serves as our laboratory. The quality of research at Erasmus University, named after Rotterdam-born humanist and theologian Erasmus, is reflected in its consistent top-100 position in most major universities rankings. On the lively, modern campus, students and scholars of more than 100 nationalities are constantly encouraged to develop their talents and meet their ambition.



	tokens
1	As
2	part
3	of
4	a
5	large
6	global
7	network
8	of
9	academic
10	partnerships
11	in
12	strategic
13	alliance

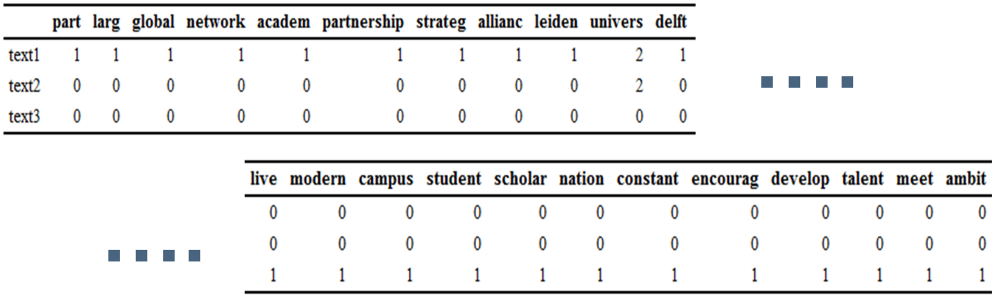
	tokens
1	as
2	part
3	of
4	a
5	large
6	global
7	network
8	of
9	academic
10	partnerships
11	in
12	strategic
13	alliance

	tokens
1	part
2	large
3	global
4	network
5	academic
6	partnerships
7	strategic
8	alliance
9	leiden
10	university
11	delft
12	university
13	technology

	tokens
1	part
2	larg
3	global
4	network
5	academ
6	partnership
7	strateg
8	allianc
9	leiden
10	univers
11	delft
12	univers
13	technolog

A bit about word representation

A term document matrix $\Rightarrow$ bag-of-words representation.
Why do they call it bag of words?



	part	larg	global	network	academ	partnership	strateg	allianc	leiden	univers	delft
text1	1	1	1	1	1	1	1	1	1	2	1
text2	0	0	0	0	0	0	0	0	0	2	0
text3	0	0	0	0	0	0	0	0	0	0	0

■ ■ ■ ■

■ ■ ■ ■

live	modern	campus	student	scholar	nation	constant	encourag	develop	talent	meet	ambit
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1

Does not preserve order.

Hard to capture meaning

- Hard to figure out relation between words.
- Dimension could be very large (think n-grams).
- Re-representation is a good idea $\Rightarrow$ what comes next?

You studied this in the past under different name

PCA as a linear encoder:

$$\sum_{i=1}^{P} \|Y_i - X\beta_i\|^2 = \sum_{i=1}^{P} \|Y_i - \widehat{Y}_i\|^2$$

- You probably applied this in the past.
- When you did, you created a particular, new, representation of your original data.
- Put differently, your original data was mapped to a lower dimension.
- Motivation for this should be clear by now.

It's all about matrix factorization.

Learning different (underlying) representation of the data

- Autoencoders falls under unsupervised learning. We try to find underlying structure in the data (density estimation, clustering)
- Autoencoders are could be viewed as special case of feedforward networks.
- You are already familiar with a private case of AE.

  ` *In fact, this simple autoencoder often ends up learning a low-dimensional representation very similar to PCAs.* '

  Andrew Ng

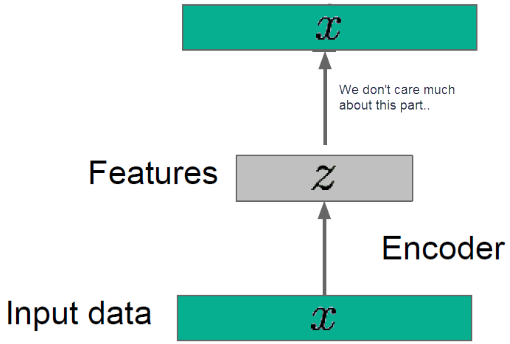
Encoding using PCA

- **PCA as a linear encoder:** We are searching for $\mathbf{W}$ which, under some conditions (orthogonality), minimizes

$$\sum_{i=1}^{P} \|\mathbf{x}_i - \mathbf{z} w_i\|^2 = \sum_{i=1}^{P} \|\mathbf{x}_i - \widehat{\mathbf{x}}_i\|^2$$

- We want the best approximation/representation possible. The above loss is called reconstruction error.
- Where does the error come from?
- Remember we are only using P, rather than N.

Autoencoder



- The Greek prefix auto- means "self." So aptly named.
- Now we are also searching for parameters, but of a *different non-linear function*.

So more generally

Instead of

$$z^* = f(x) = \text{argmin}\, \|x - Wh\|_2^2,$$

we change two things

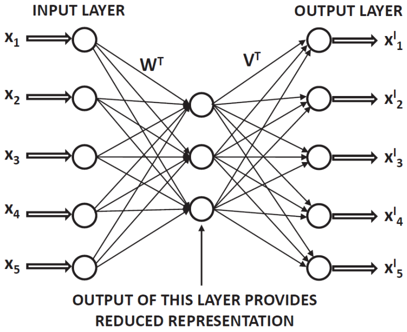
1. Different functional form. Non-linear, but still pre-specified.
2. Possible different loss.

We try to minimize the loss:

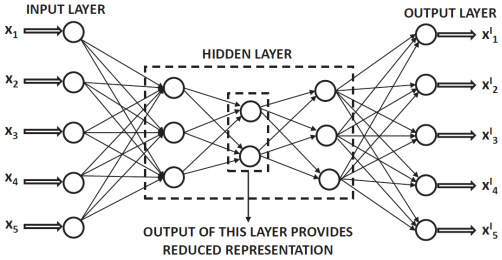
$$L_{\mathcal{AE}}(\mathbf{w}) = \sum L\left(\mathbf{x}, g_{\mathbf{w}}\left(f_{\mathbf{w}}\left(\mathbf{x}\right)\right)\right)$$

Autoencoder versus SVD

SVD style



Autoencoder



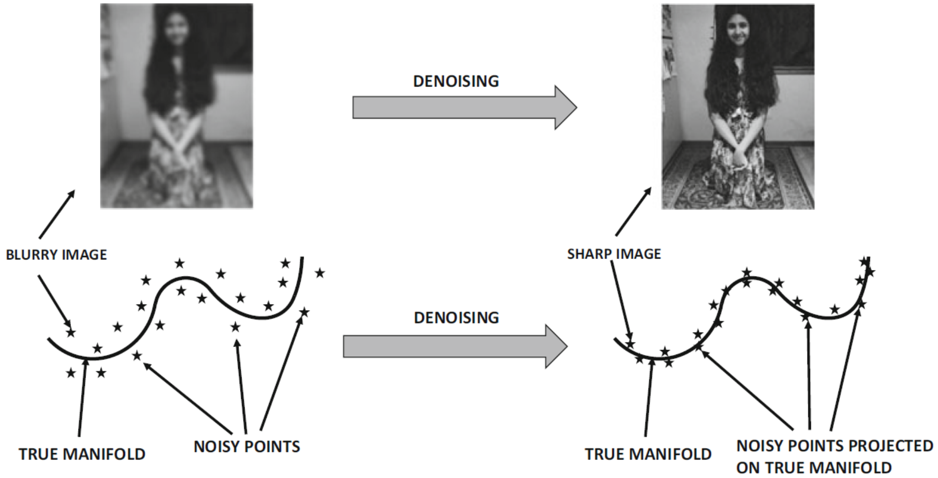
Regularized AE

- Sparse AE: use a penalized loss function á la LASSO to impose sparsity constraint on the hidden units:

$$L(\boldsymbol{x}, g(\boldsymbol{x})) + \Omega(\boldsymbol{h}; \lambda),$$

- We don't know enough about the difference in performance between undercomplete AE or overcomplete AE with sparsity constraints.
- De-noising AE

De-noising AE

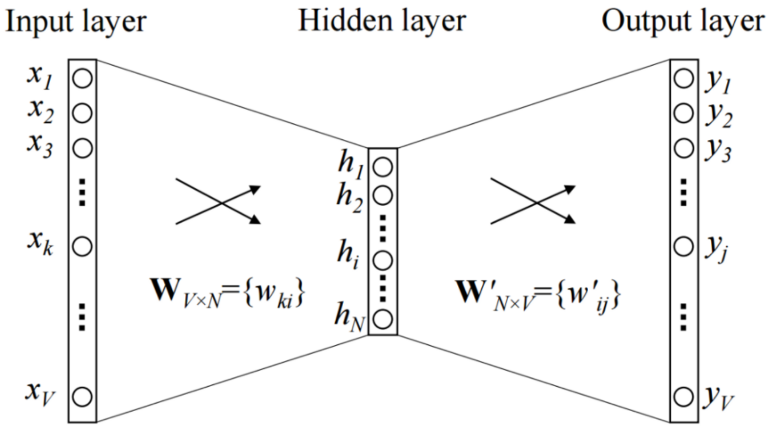


Pointers

- The reduced representation is often referred to as (1) feature-vector (2) representation, or (3) code.
- Both version of regularization echos very much the classical ridge regression from 1970 (future exercise).
- “A method is a trick you use twice”
- Now you can directly understand other-things embedding (i.e. word, graph).

Word embedding

Common Bag Of Words

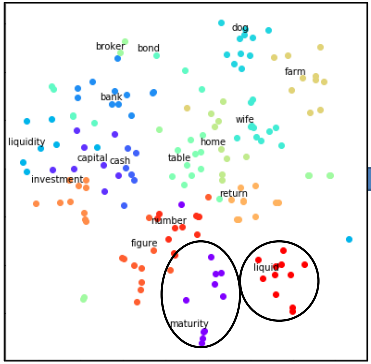


Word embedding - pointers

- Hard to train.
- Hard to evaluate. Usually application dependent.
- We can't let Jaccard sleep just yet. Limited use of off-the-shelf $\Rightarrow$ representation/vectors are not relevant outside your sphere.
- Still one of the most influential developments in NLP space.

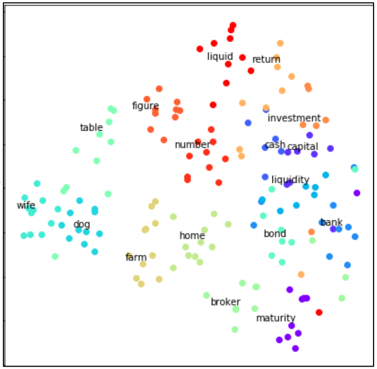
Word embedding - visualization

Google news



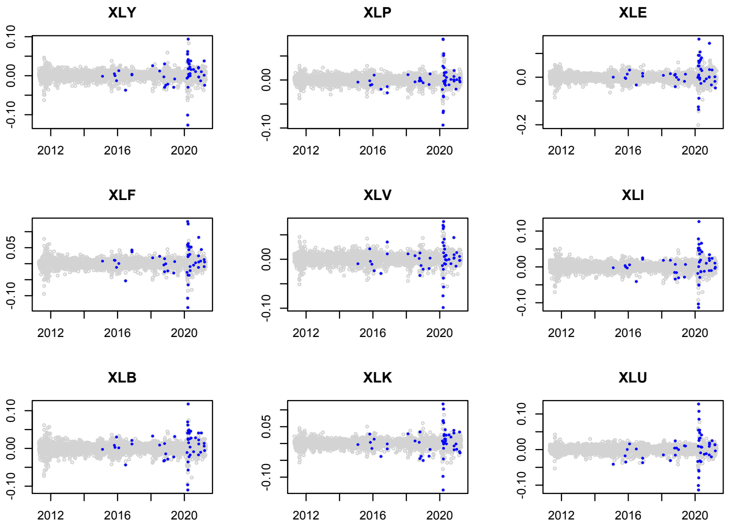
maturity		return		liquid	
maturing	0.70	returning	0.67	fluid	0.50
mature	0.64	leave	0.51	acetic	0.48
matures	0.60	depart	0.50	powder	0.46
poise	0.46	back	0.44	toxic	0.45
longevity	0.40	arrive	0.43	acid	0.44
redemption	0.38	surrender	0.42	methanol	0.43
consistency	0.38	resume	0.41	peroxide	0.42
strength	0.37	revert	0.41	mixture	0.41
age	0.37	recuperate	0.41	polymer	0.40

Financial tilt



maturity		return		liquid	
duration	0.59	returning	0.39	safer	0.44
dated	0.56	earn	0.36	risky	0.44
mature	0.53	performance	0.34	marketable	0.39
maturing	0.48	reinvested	0.33	liabilities	0.37
bond	0.46	payout	0.32	mixture	0.37
coupon	0.45	yield	0.32	quantity	0.36
tenor	0.44	redemption	0.31	safe	0.36
yield	0.44	revert	0.30	pool	0.36
term	0.43	increase	0.30	toxic	0.36

Another application: outlier detection



References

Alex Graves website

Agarwala, N., Inoue, Y., and Sly, A. (2017). Music composition using recurrent neural networks. CS 224n: Natural Language Processing with Deep Learning, Spring.

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10):2222–2232.

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In International conference on machine learning, pages 1310–1318.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11):2673–2681.

Now Let's code!