Literature Summary for Master's Thesis (WIP)

Ege Yilmaz

Wednesday 9th June, 2021

Contents

1 RNNs

2 Reservoir Computing

2.1 Echo State Networks [8]

Discrete-time neural networks with K input units with activations yielding $\mathbf{u}$(n) (n:time-step), N internal recurrent units yielding $\mathbf{x}$(n) and L output units yielding $\mathbf{y}$(n).

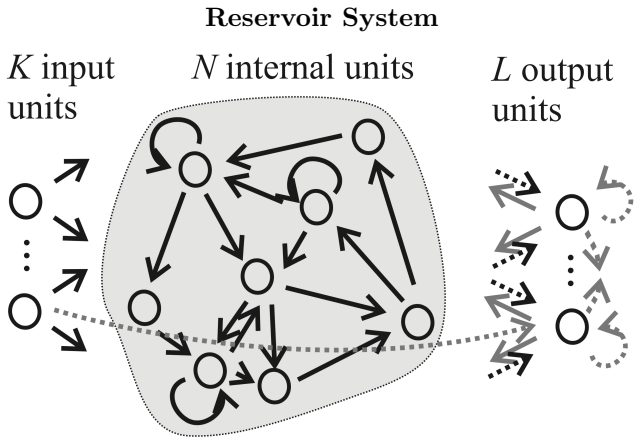


Fig. 1: Source: [8].

- Input to Reservoir with $\mathbf{W}^{\text{in}}_{N\times K}$
- Reservoir to Reservoir with $\mathbf{W}_{N\times N}$
- (Input + Reservoir + Output) to Output with $\mathbf{W}^{\text{out}}_{L\times(K+N+L)}$
- Output to Input with $\mathbf{W}^{\text{back}}_{N\times L}$

$$\mathbf{x}(n+1) = (1-\alpha)\mathbf{x}(n) + \alpha \cdot \mathbf{f}(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{back}}y(n)), \quad \mathbf{f}: \text{Reservoir activations}, \alpha: \text{Leaking rate} \tag{1}$$

$$\mathbf{y}(n+1) = \mathbf{f}^{\text{out}}(\mathbf{W}^{\text{out}} \cdot concat(\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n))), \qquad \mathbf{f}^{\text{out}}: \text{Output activations} \tag{2}$$

Definition 1. ***Standard compactness conditions:***
(i) input is drawn from a compact input space U
(ii) network states lie in a compact set A.

From now on assume standard compactness conditions and a network without output feedback.

Definition 2. *Iterator/state updater* $T(\boldsymbol{x}(n), \boldsymbol{y}(n), \bar{\boldsymbol{u}}^h) := \boldsymbol{x}(n+h)$, *where* $\bar{\boldsymbol{u}}^h$ *is the input sequence* $\boldsymbol{u}(n+1), ..., \boldsymbol{u}(n+h)$

Definition 3. *Echo State Property (ESP) or Uniqueness of Solutions Property. Following statements are equivalent:*

- Echo States $\boldsymbol{x}$ *are states uniquely determined by any* $\bar{\boldsymbol{u}}^{-\infty}$ *and* T *if network has no output feedback* $\boldsymbol{W}^{back}$.

- $\exists$ *input echo functions* $E = (e_1, ..., e_N)$, *where* $e_i : U^{-N} \to R$, *s.t.* $\forall$ *left-infinite input histories* $..., u(n-1), u(n) \in U^{-N}$ *the current network state is* $x(n) = E(..., u(n-1), u(n))$.

Definition 4. *Additional network properties*

I. Network is state contracting $\iff$ *all reservoir states are similar on right-infinite inputs extending sufficiently far into the future.*

II. Network is state forgetting $\iff$ *all reservoir states are similar on left-infinite inputs extending sufficiently far into the past.*

III. Network is state input forgetting $\iff \lim_{t\to\infty} \left\| H_U(\boldsymbol{u}\boldsymbol{z}_t^1) - H_U(\boldsymbol{v}\boldsymbol{z}_t^1) \right\| = 0$, H_U *is functional associated to reservoir filter* U, $\boldsymbol{u}$,$\boldsymbol{v}$ *are semi-infinite sequences.* $\boldsymbol{z}_t^1$ *is input at time* t. *So concatenating different inputs with the same input at infinite future yields indistinguishable output.*

Proposition 1. *Assume that* T *is continuous in state and input. State contracting + state forgetting + input forgetting* $\iff$ *ESP.*

Example 1. $u(n) = sin(2\pi n/P)$, *P: periodicity. Because of ESP the activations* $x_i(n)$ *are also periodic signals with the same period length P; but the network's inhomogeneity induces conspicuous deviations from the input sinusoidal form. See paper for plots.*

The 100-unit network used in the example was randomly connected; weights were set to values of 0, +0.4 and -0.4 with probabilities 0.95, 0.025, 0.025 respectively. This means a sparse connectivity of 5%. *The value of 0.4 for non-null weights resulted from a global scaling such that* $|\lambda_{max}| \approx 0.88 < 1$ *(spectral radius) was obtained. How to scale is explained in the paper.*

Example 2. *(House of the Rising Sun) With output feedback with uniform random weigths.* $f^{out} = tanh$. *A 400 unit sigmoid network was used. Internal connections were randomly as- signed values of 0, 0.4, -0.4 with probabilities 0.9875, 0.00625, 0.00625. This resulted in a weight matrix W with a sparse connectivity of 1.25%. The maximal eigenvalue of W was* $|\lambda_{max}| \approx 0.908$. *The fact that spectral radius is close to 1 means that the network exhibits a long-lasting response to a unit impulse input. Generally, the closer spectral radius is to unity, the slower is the decay of the network's response to an impulse input. A relatively long-lasting "echoing" of inputs in the internal network dynamics is a requisite for a sizable short-term memory performance of the network.*

Problem: Reservoir states become periodic. Thus, minimization problem yields less equations in effect (linear dependence). Less than the dimension of W^{out} *making the system of equations underdetermined. This results in many possible perfect solutions. The 'naive' solution is unstable. Answer is to add uniform noise to* $\boldsymbol{y}(n)$ *which results in 'wobbling' states* $\boldsymbol{x}(n)$.

2.1.1 Multiple attractor learning

TBP

2.1.2 ESNs with leaky integrator neurons

TBP. See for Mackey-Glass example.

2.1.3 Key Takeaways

- Only the weights of connections leading to the output units are trained; all other connections remain un- changed. This makes it possible to employ any of the many available fast, constructive linear regression algorithms for the training. No special, iterative gradient-descent procedure is needed.
- We want that the eigenvalue of W with the largest absolute value (spectral radius) is smaller than 1. Otherwise network has an asymptotically unstable null state. This means no echo states for any input set U containing **0**. There should be no problem if the set does not contain **0**.
- We want sparse and random connections.
- On one hand waste of units (400 reservoir units in ESN can be done by say 20 LSTMs with gradient descent), on the other hand multi-tasking possibilities.

2.2 ESNs are universal [6, 7]

$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{z}_t) \qquad F\text{: reservoir}, \mathbf{x}_t\text{: reservoir state}, \mathbf{z}_t\text{: input signal}, \tag{3}$$

$$\mathbf{y}_t = h(\mathbf{x}_t) \qquad h\text{: readout}, \mathbf{y}_t\text{: output signal} \tag{4}$$

ESNs can be used as universal approximants in the context of discrete-time fading memory filters with uniformly bounded inputs defined on negative infinite times.

A major breakthrough was the generalization to infinite time intervals carried out by Boyd and Chua in [1], who formulated a uniform approximation theorem using Volterra series for operators endowed with the so called fading memory property on continuous time inputs.

Fading memory property says that inputs which are close in the recent past but not close in the remote past still yield close outputs in the present. Close here is in terms of peak deviation. From causal and TI filters one can go to the associated functional via the bijection map [1]. Causal means inputs which were the same until now, yield same filter outputs now and their future values do not play a role. Then the condition to achieve FMP is a strengthened continuity of the functional (using weighted norm with a decaying weight sequence).

A discrete filter (not necessarily RC) with FMP can be approximated by SAS family (external approx.) and SAS can be approximated by ESNs (internal approx.).

ESNs are RCs with ESP. ESP holds when there is a state sequence $\mathbf{x}$ which solve the reservoir equation (3) for a given input sequence $\mathbf{z}$ and this state sequence is unique.

ESP and FMP are difficult to check directly but there are other conditions that guarantee them. For example, contracting continuous reservoir maps with contraction constant less than 1 induce reservoir filters that automatically have the echo state and the fading memory properties. Similar but more restrictive yet easy to apply sufficient condition is to have spectral radius less than 1 and sigmoid activation function and states in $[-1, 1]^N$.

2.2.1 Notation and definitions

- **Filters:** $U : (D_n)^{\mathbb{Z}} \to \mathbb{R}^{\mathbb{Z}}$
- **Functionals:** $H : (D_n)^{\mathbb{Z}} \to \mathbb{R}$
- **Causal:** $z, w \in (D_n)^{\mathbb{Z}}$ with $z_\tau = w_\tau, \forall \tau \leq t$ and $U(z)_t = U(w)_t \implies$ Filter is causal
- **Time Delay Operator:** $U_\tau(z)_t = z_{t-\tau}$
- **Time Invariant:** $[U_\tau, U] = 0$
- **Filter determined by reservoir map:** $U^F : (D_n)^{\mathbb{Z}} \to (D_N)^{\mathbb{Z}}$
- **Reservoir Filter:** $U_h^F(\mathbf{z})_t := h(U^F(\mathbf{z})_t)$, h:readout, F: Reservoir map
- **Filter - Functional bijection:** Given a time-invariant filter U, we can associate to it a functional $H_U(\mathbf{z}) := U(\mathbf{z}^e)_0$, where $\mathbf{z}^e$ is an arbitrary extension of left-semi-infinite $\mathbf{z}$ to infinity. Conversely, for any functional H we can define a time-invariant causal filter $U_H(\mathbf{z})_t := H(\mathbb{P}_{\mathbb{Z}_-} \circ U_{-t})(\mathbf{z}))$, where U_{-t} is the $(-t)$-time delay operator and $\mathbb{P}_{\mathbb{Z}_-} : (D_n)^{\mathbb{Z}} \to (D_n)^{\mathbb{Z}_-}$ is the natural projection.
- **Weighted Norm (of left semi-infinite sequences):** $\|\mathbf{z}\|_w := \sup_{t \in \mathbb{Z}_-} \|\mathbf{z}_t w_{-t}\|$, where $w_{n \in \mathbb{N}} \in (0, 1]$ a monotonous zero sequence.
- l_w^∞ is the bounded sequence space with the weighted norm and is a Banach space.
- **(Exponential) Fading Memory Property:** $(w_n = \lambda^n, \lambda \in (0, 1)) H_U : ((D_n)^{\mathbb{Z}_-}, \|\cdot\|_w) \to \mathbb{R}$. If $\exists w$ s.t. $|H_U(\mathbf{z}) - H_U(\mathbf{s})| < \epsilon$ with $\|\mathbf{z} - \mathbf{s}\|_w = \sup_{t \in \mathbb{Z}} \{\|(\mathbf{z}_t - \mathbf{s}_t) w_{-t}\|\} < \delta(\epsilon)$. That means whenever there is a w s.t. H_U is continuous.

2.2.2 Key Takeaways

- Fading Memory Property implies input forgetting property.
- ESP and FMP imply uniqueness and continuity of solutions, respectively.
- ESP and FMP are difficult to check directly but there are other conditions that guarantee them such as local contractivity: If the reservoir map is a contraction and its contraction constant is less than 1. See also [3, 12].
- Echo state property of the reservoir map F implies causality and invariance of the filter U^F.
- When a filter is causal and time-invariant it suffices to work with the restriction $U : (D_n)^{\mathbb{Z}_-} \to (D_n)^{\mathbb{Z}_-}$ instead of the original $U : (D_n)^{\mathbb{Z}} \to (D_n)^{\mathbb{Z}}$ since the former uniquely determines the latter.
- ESNs can be used as universal approximants in the context of discrete-time fading memory filters with uniformly bounded inputs defined on negative infinite times. Proven with internal (approximating unique RC filters) and external approximation (approximating some filter in general) properties.

2.3 SigSAS [5]

2.3.1 Key Takeaways

- Network (aka filter) can be approximated by volterra series on uniformly bounded left infinite inputs. When this filter has additionally FMP the truncation error can be quantified.
- There is a state system SigSAS with ESP and FMP on uniformly bounded inputs. The state map inducing SigSAS and the continuous, time-invariant and causal associated filter have explicit forms. Any fading memory filter can be approximated up to monotonically decreasing rest term by this filter together with a trained linear readout. The quality of the approximation is not filter independent, as the decreasing sequence in the rest term depends on how fast the filter U "forgets" past inputs.
- The tensor space on which the SigSAS state filter is defined is high dimensional. This can be remedied by random projections in Johnson-Lindenstrauss Lemma. The random projections of the SigSAS system yield SAS systems with randomly generated coefficients in a poten- tially much smaller dimension which approximately pre- serve the good properties of the original SigSAS system. The loss in performance that one incurs because of the projection mechanism can be quantified using the Johnson-Lindenstrauss Lemma.

2.4 Liquid State Machines (Maass) [10]

Similar to ESNs but more biologically inspired. Uses spiking (integrate & fire) neurons in the reservoir and readout and the connections from reservoir and the readout are trained via perceptron. The neurons fire (create a exponentially decaying spike train) when the successive sum of inputs pass some threshold value.

3 HFT

3.1 Path Dependence [4]

We said we do not want to concentrate on it for now.

3.2 Hawkes Processes [9]

3.3 Reinforcement Learning [2]

3.4 Irregular Time Intervals [11]

3.5 Mid-Price Strategies [13]

3.6 Bars [14]

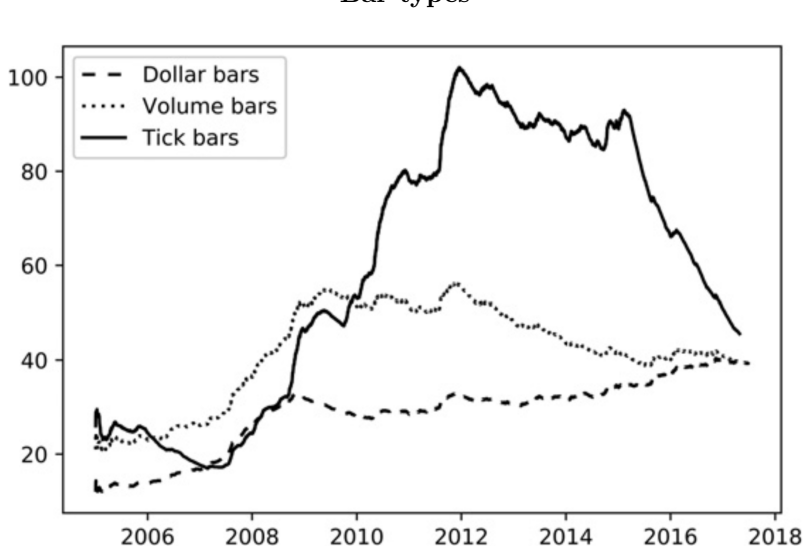


Fig. 2: Source: [14]

3.6.1 Standard Bars

3.6.1.1 Time Bars

Bars are obtained over fixed time intervals.

3.6.1.2 Tick Bars

Bars are extracted each time a predetermined number of transactions takes place. Tick bars allow for better inference than time bars.

3.6.1.3 Volume Bars

Suppose that there is one order sitting on the offer, for a size of 10. If we buy 10 lots, our one order will be recorded as one tick. Volume bars circumvent that problem by sampling every time a predefined amount of the security's units have been exchanged.

3.6.1.4 Dollar Bars

The number of shares traded is a function of the actual value exchanged. Therefore, it makes sense sampling bars in terms of (dollar) value exchanged, rather than ticks or volume, especially in case of significant price fluctuations.

3.6.2 Sampling

3.6.2.1 CUSUM Filter

3.7 Triple Barrier Method [14]

Fig. 3: Symmetrical barriers used for labelling the side of the trade. Source: [14]

Triple Barrier method is used to learn the side of each trade. It is a path-dependent labelling strategy, which takes the problems of fixed-time horizon methods into account, such as autocorrelation, heteroscedasticity and volatility-independent thresholds, by imposing the following dynamic barrier rules:

1. The upper barrier (profit-take) is hit first. Label = "1 (buy opportunity)".
2. The lower barrier (stop-loss) is hit first. Label = "-1 (sell opportunity)".
3. The vertical barrier (expiration) is hit first. Label = sign of the return or a 0.

The idea is to teach these labels to a machine learning algorithm in a supervised setting, where **Exponentially Weighted Standard Deviations (EWSD)** (5) computed from returns inside a predetermined time frame prior to each barrier are used to determine the horizontal barrier widths. A front (vertical) barrier is also placed acting as an expiration limit. If it is hit first and the corresponding label is chosen as 0, then it indicates that there is no trading opportunity.

$$\mathrm{EWSD}_{t_k} = \sqrt{b_k \cdot \frac{\sum_{i=0}^{k} w_i (x_{t_{k-i}} - \mathrm{EWMA}_{t_k})^2}{\sum_{i=0}^{k} w_i}} \tag{5}$$

- α : the smoothing factor
- $w_i = (1-\alpha)^i$
- $t_k \in \{t_n | t_n - t_{n-1} = \text{predetermined time length } \forall n \in \mathbb{N}^+, t_0 : \text{first registered time in dataset}\}$
- x_{t_k} : return with $x_{t_k} = price_{t_k}/price_{t_{k-1}} - 1$
- b : bias with $b_k = \frac{(\sum_{i=0}^{k} w_i)^2}{(\sum_{i=0}^{k} w_i)^2 - \sum_{i=0}^{k} w_i^2}$

- Exponentially Weighted Moving Average $\text{EWMA}_{t_k} = \frac{\sum_{i=0}^{k} w_i x_{t_{k-i}}}{\sum_{i=0}^{k} w_i}$

After the data is divided into windows with predetermined time frames and triple barrier labels are attached to these, a model is given previous prices as input and trained to learn the labels.

In our case, the data is divided into 30 minute windows. Looking at previous 30 minutes, a standard deviation (5) is calculated. Using this, the horizontal barriers are placed at $\text{price}_{t_k} \cdot (1 \pm \text{EWSD}_{t_k})$, where t_k is the time of the barrier start/position entry. A vertical barrier at 30 minutes ahead is placed. This is done for each bar in the Limit Order Book (LOB) starting from 30 minutes after the first bar in LOB. The model is given the previous 30 minutes' prices as input, including the price at the minute, where it is expected to yield a label, i.e. forecast the correct position: long, short or do nothing. So each input consists of 31 prices. The windows for which exactly 31 bars are not available in the LOB are removed during data preparation as well as time bars before 10:00 and after 17:00, belonging to single price auctions in Borsa Istanbul.

3.7.1 Meta-Labelling

A secondary model could be used to assess the confidence in the labels from the primary model described previously. In our case, binary confidences are calculated, i.e. for each prediction from the primary model a label $\in \{0, 1\}$ is given by the secondary model. The continuous case $[0, 1]$ will be explored soon.

The goal here is to reduce false positives which could be costly in some trades, at the expense of true positives, where true positive rate (TPR) will be reduced and the precision will be increased. It takes the predictions of the primary model as input and the correct labels are given as output. In our case this is done by training reservoirs. The results are shown in Figure 4.

3.7.2 Barrier Widths

Learning the side of the trade, we needed to be fair to both possible positions (long and short) and for that we have utilized symmetrical horizontal barriers. Once a model knows the correct side of a trade, putting actual profit taking and stop-loss barriers is the next step, which do not need to be symmetrical. For this purpose a model is trained to forecast the maximum available return inside a given window as well as the minimum return observed before hitting this maximum return.

In our case the model takes previous 31 prices including the price at the minute of position taking, the return of the previous 30 minute interval (see x_{t_k} in (5)), EWSD and EWMA as input with a length of 34.

3.7.3 Application

Once (meta-labelled) side predictions, profit taking targets and stop-loss thresholds are available, a trading experiment can be performed.

The data is divided into three subsets: 1. Training 2. Validation 3. Meta. A primary model is trained on the blue part in Figure 5 to learn the labels of sides. These labels are filtered through a meta-labelling model described in 3.7.1. This meta-label model is trained on the orange part in Figure 5, i.e. on the first half of the validation of the primary model. For meta-/labelling "meta_f=sigmoid/f_out=tanh" hyperparameters are chosen. Lastly another model is taught the barrier widths described in 3.7.2 with a training domain of blue+orange in Figure 5. Its validation domain is the same as meta-labelling model's validation domain, the green part. This is where the experiment is run.

Each minute in the green domain, using the resulting predictions from previously described stages, we look at 30 minute windows. If the profit taking barrier is exceeded during this period given that the stop-loss barrier is not hit prior to that, the model executes the trade at that minute. If the stop-loss barrier is hit first, then the position is ended there. If none of the non-symmetrical barriers are hit, the position is ended at the vertical barrier, i.e. after 30 minutes. In each case the positive (negative) return at that point is added (subtracted) from the wallet. The bet size is given by the total capital divided by the price of the asset at the time of the position.

Currently, the experiment is performed on bid prices. A more realistic case where both bid and ask prices are used for buying and selling the asset will be investigated. Furthermore, a scenario where the algorithm is only allowed to invest $\frac{\text{available volumes} \cdot \text{bid/ask price}}{100}$ TL will also be investigated. The current setting yields 223.5 B with an initial capital of 100 within a trading period of approx. 51 days.

Side Learning Scores. Total no of training/validation points: 44430, 29620

	Accuracy (%)	Precision (%)	F1 (%)	TPR (%)	TTPR (%)	TP	TN	FP	FN	FTP
f = id	90.16	95.25	92.85	90.56	97.41	1.861•E4	7650	928	1939	495
f = sigmoid	90.1	95.39	92.8	90.34	97.55	1.859•E4	7680	898	1988	466
f = tanh	90.09	95.63	92.77	90.07	97.55	1.853•E4	7732	846	2044	466
f = LeakyReLU (a=0.5)	89.95	95.55	92.67	89.95	97.56	1.851•E4	7716	862	2068	463
f_out = tanh	87.91	86.7	91.92	97.8	96.29	1.983•E4	5537	3041	447	765

(a) Scores of reservoirs trained with different activation functions. "id" stands for identity map. "f_out=tanh" means that a sigmoid activation as reservoir activation and tanh as output activation is chosen.

Confidence Learning Scores (Relative). Total no of training/validation/meta data points: 44430, 14810, 14810

		Accuracy (%)	Precision (%)	F1 (%)	TPR (%)	TTPR (%)	TP	TN	FP	FN	FTP
	f = id	-0.26•E-1	0.83•E-2	-2.04•E-2	-0.46•E-1	1.97•E-2	-3	1	-1	5	-2
	f = sigmoid	-0.2•E-1	-0.91•E-3	-0.15•E-1	-2.75•E-2	0.98•E-2	-2	0.0	0.0	3	-1
meta_f = id	**f = tanh**	-1.37•E-2	0.85•E-2	-1.14•E-2	-2.85•E-2	-0.68•E-3	-3	1	-1	3	0.0
	f = LeakyReLU (a=0.5)	-0.61•E-2	0.94•E-2	-0.53•E-2	-0.18•E-1	0.1•E-1	-1	1	-1	2	-1
	f_out = tanh	0.13	2.54•E-1	0.64•E-1	-1.82•E-1	0.41•E-1	-14	37	-37	19	-5
	f = id	-0.44•E-1	1.47•E-1	-0.46•E-1	-2.14•E-1	0.47•E-1	-18	16	-16	23	-5
	f = sigmoid	-0.8•E-1	1.08•E-1	-0.71•E-1	-2.25•E-1	0.26•E-1	-21	12	-12	24	-3
meta_f = sigmoid	**f = tanh**	-0.25•E-1	0.15	-3.14•E-2	-1.87•E-1	0.27•E-1	-17	16	-16	20	-3
	f = LeakyReLU (a=0.5)	-0.32•E-1	0.14	-0.36•E-1	-1.87•E-1	0.27•E-1	-17	15	-15	20	-3
	f_out = tanh	2.3	5.53	1.12	-4.03	0.28	-377	748	-748	420	-43
	f = id	-3.04•E-1	0.38	-0.27	-0.83	0.42•E-1	-82	43	-43	88	-6
	f = sigmoid	-4.34•E-1	4.16•E-1	-3.74•E-1	-1.05	2.75•E-2	-106	47	-47	111	-5
meta_f = tanh	**f = tanh**	-2.34•E-1	3.94•E-1	-2.15•E-1	-0.73	0.56•E-1	-71	43	-43	78	-7
	f = LeakyReLU (a=0.5)	-0.37	0.31	-0.32	-0.85	0.32•E-1	-85	35	-35	90	-5
	f_out = tanh	2.34	5.8	1.12	-4.32	0.31	-403	782	-782	450	-47
	f = id	-1.68•E-1	1.58•E-1	-1.43•E-1	-4.04•E-1	0.42•E-1	-38	18	-18	43	-5
	f = sigmoid	-0.34	0.11	-0.27	-0.6	0.38•E-1	-59	14	-14	64	-5
meta_f = LeakyReLU (a=0.5)	**f = tanh**	-2.23•E-1	1.38•E-1	-1.85•E-1	-0.46	0.31•E-1	-45	16	-16	49	-4
	f = LeakyReLU (a=0.5)	-0.24	0.9•E-1	-1.95•E-1	-0.44	0.63•E-1	-40	11	-11	47	-7
	f_out = tanh	1.65	4.69	0.71	-3.98	0.3	-370	648	-648	415	-45
	f = id	-0.45•E-1	0.17•E-1	-3.56•E-2	-0.82•E-1	3.96•E-2	-5	2	-2	9	-4
	f = sigmoid	-0.4•E-1	1.67•E-2	-0.32•E-1	-0.74•E-1	0.19•E-1	-6	2	-2	8	-2
meta_f_out = tanh	**f = tanh**	-0.46•E-1	0.76•E-2	-0.36•E-1	-0.73•E-1	2.96•E-2	-5	1	-1	8	-3
	f = LeakyReLU (a=0.5)	-0.33•E-1	1.74•E-2	-2.66•E-2	-0.64•E-1	1.94•E-2	-5	2	-2	7	-2
	f_out = tanh	0.52•E-1	0.37	-0.41•E-2	-0.49	1.64•E-1	-32	56	-56	51	-19

(b) Scores of meta-reservoirs trained with different activation functions on labels of different reservoirs.

Fig. 4: Triple Barrier label scores. TTPR is given by TP/(TP+FTP) and FTP is the number of correct identification of trading opportunities with incorrect side.

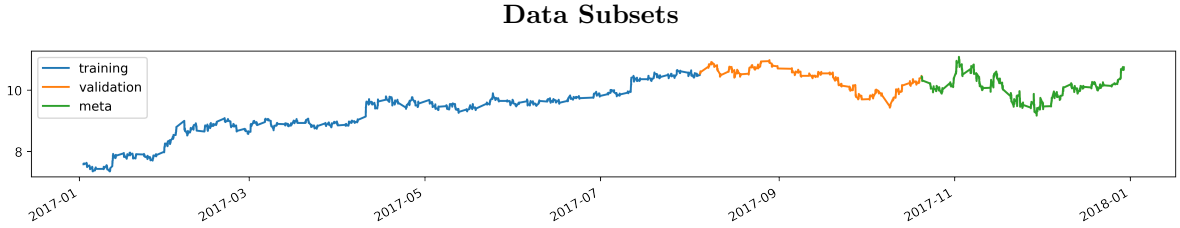
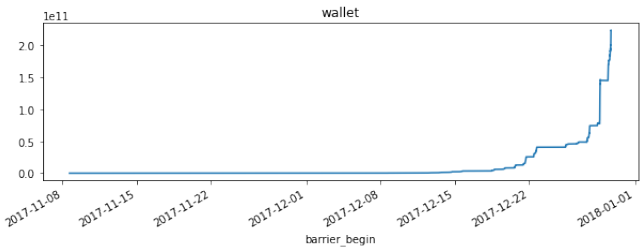


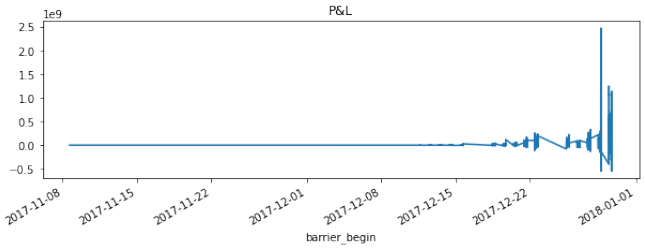
Fig. 5: Ratios of subsets (from left to right) to total length is 3:1:1

Summary of Experiment

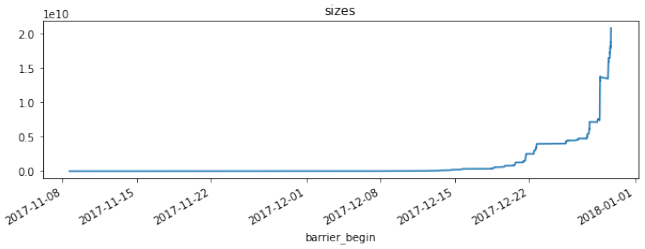
- Initial capital: 100
- Tot. Returns: $2.235 \cdot 10^{11}$
- No of loss-making trades: 3126
- No of profitable trades: 4984
- Gains: $2.588 \cdot 10^{11}$
- Losses: $3.533 \cdot 10^{10}$
- Time frame: 51 days 2 hours 3 minutes



(a) Total capital throughout the trading session



(b) Profits and Losses throughout the trading session



(c) Bet sizes throughout the trading session

Fig. 6: Plots from the experiment described in 3.7.3

References

[1] S. Boyd and L. Chua. "Fading memory and the problem of approximating nonlinear operators with Volterra series". In: *IEEE Transactions on Circuits and Systems* 32.11 (1985), pp. 1150–1161. DOI: 10.1109/TCS.1985.1085649.

[2] Antonio Briola et al. *Deep Reinforcement Learning for Active High Frequency Trading*. 2021. arXiv: 2101.07107 [cs.LG].

[3] Andrea Ceni et al. "The echo index and multistability in input-driven recurrent neural networks". In: *Physica D: Nonlinear Phenomena* 412 (Nov. 2020), p. 132609. ISSN: 0167-2789. DOI: 10.1016/j.physd.2020.132609. URL: http://dx.doi.org/10.1016/j.physd.2020.132609.

[4] Rama Cont and David-Antoine Fournié. "Functional Itô calculus and stochastic integral representation of martingales". In: *The Annals of Probability* 41.1 (Jan. 2013). ISSN: 0091-1798. DOI: 10.1214/11-aop721. URL: http://dx.doi.org/10.1214/11-AOP721.

[5] Christa Cuchiero et al. *Discrete-time signatures and randomness in reservoir computing*. 2020. arXiv: 2010.14615 [cs.NE].

[6] Lyudmila Grigoryeva and Juan-Pablo Ortega. *Echo state networks are universal*. 2018. arXiv: 1806.00797 [cs.NE].

[7] Lyudmila Grigoryeva and Juan-Pablo Ortega. *Universal discrete-time reservoir computers with stochastic inputs and linear readouts using non-homogeneous state-affine systems*. 2018. arXiv: 1712.00754 [cs.NE].

[8] Herbert Jaeger. "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note"'. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (Jan. 2001).

[9] Xiaofei Lu and Frédéric Abergel. "High-dimensional Hawkes processes for limit order books: modelling, empirical analysis and numerical calibration". In: *Quantitative Finance* 18.2 (2018), pp. 249–264. DOI: 10.1080/14697688.2017.1403142. eprint: https://doi.org/10.1080/14697688.2017.1403142. URL: https://doi.org/10.1080/14697688.2017.1403142.

[10] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations". In: *Neural computation* 14 (Dec. 2002), pp. 2531–60. DOI: 10.1162/089976602760407955.

[11] Ross A. Maller, Gernot Müller, and Alex Szimayer. "GARCH modelling in continuous time for irregularly spaced time series data". In: *Bernoulli* 14.2 (May 2008). ISSN: 1350-7265. DOI: 10.3150/07-bej6189. URL: http://dx.doi.org/10.3150/07-BEJ6189.

[12] G. Manjunath. "Stability and memory-loss go hand-in-hand: three results in dynamics and computation". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2242 (Oct. 2020), p. 20200563. ISSN: 1471-2946. DOI: 10.1098/rspa.2020.0563. URL: http://dx.doi.org/10.1098/rspa.2020.0563.

[13] Paraskevi Nousi et al. *Machine Learning for Forecasting Mid Price Movement using Limit Order Book Data*. Sept. 2018.

[14] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. 1st. Wiley Publishing, 2018. ISBN: 1119482089.