Further experiments with Machine Learning tools at BESSY II

Luis Vera Ramírez

Helmholtz-Zentrum Berlin (HZB), Germany

AMALEA Meeting, 09/09/2019

Table of Contents

Improved results on beam lifetime prediction

Self-optimization (RLControl)
- Optimization of booster current
- Optimization of injection efficiency

Further development

References

Backup

Table of Contents

Improved results on beam lifetime prediction

- New prediction tests for ICALEPCS. Focus on **time-series-based beam lifetime prediction** restricted to a ***blind* scenario** - the input given to the prediction model consists only of **context variable readbacks**, i.e. **omitting previous beam lifetime measurements.**
- New variables, models (RFF) and preprocessing techniques have led to an improvement of the results in comparison to our previous report at the AMALEA meeting in May.

Improved results on beam lifetime prediction: Measurement

- Beam lifetime: defined via the **current decay rate**

$$\frac{1}{\tau} = -\frac{\mathring{I}}{I}$$

- EPICS Variable can't be used - very delayed
- First approach: exact calculation from measurements - unstable due to measurement errors

$$\frac{1}{\tau} = -\frac{\ln(I_t) - \ln(I_{t_0})}{t - t_0}$$

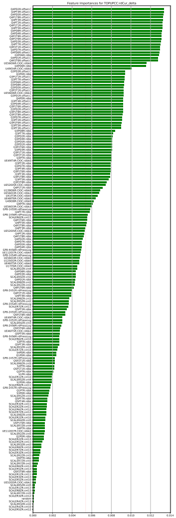
- Final approach: **piecewise linear regression** with k previous measurements (experiments with $k = 20$)

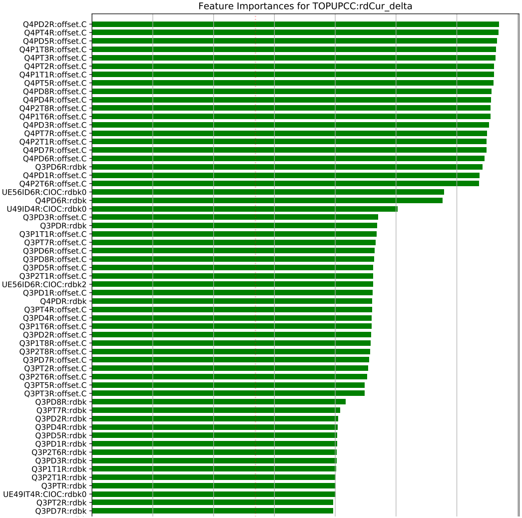
$$\frac{1}{\tau} \approx -\frac{1}{I_t} \frac{\sum_{i=0}^{k} (I_{t-i} - I_{t_0})(t - i - t_0)}{\sum_{i=0}^{k} (t - i - t_0)^2}$$

Improved results on beam lifetime prediction: Input variables

- Gap and shift of **insertion devices** (elliptical) undulators affecting the dynamic aperture (21 readback variables).
- **Power supply currents** into **quadrupoles** define the linear optics (58 readback variables), into **sextupoles** define non linear behavior (7 variables).
- **Offsets** to power supplies for quadrupoles define the feed forward compensations (38 variables).
- **Collisions** with rest gas particles, **vacuums pressure** measured by getter pump current (12 variables).
- **Local beam loss fractions**, monitored by counters close by (49 variables).

Improved results on beam lifetime prediction: Feature Importances





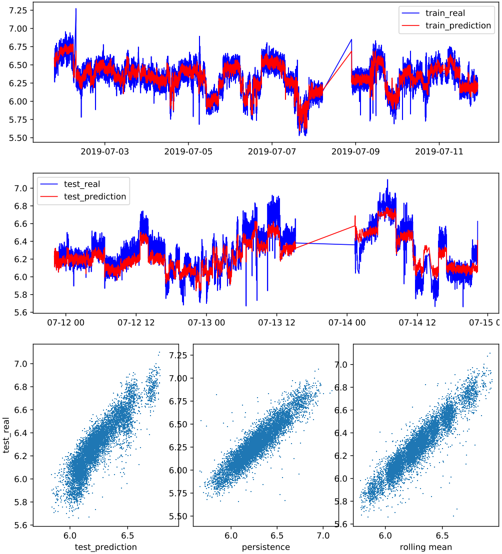
- Analysis with `RandomForest`.
- Evenly distributed feature importances - **quadrupoles** (offsets) and **insertion devices** stand out.

Improved results on beam lifetime prediction: Results

- Data from 2019-07-01 19:00:00 until 2019-07-16 19:00:00, restricted to **top-up** and **multibunch**.
- 80% (31631 samples) is used for training and 20% for test (7908 samples).
- Tests both with **random** and **chronological** split.
- Baselines:
  - Test set average.
  - Persistence: previous target measurement.
  - *Moving persistence*: moving average of the last 5 target measurements.

Test set	Algorithm	RMSE				R^2		
		Avg.	Pers.	Mov. pers.	Model	Pers.	Mov. pers.	Model
Random 20%	ExtraTrees	0.201319	0.099248	0.091464	0.068175 ± 0.000038	0.756961	0.79359	0.885322 ± 0.000128
	SVR-RFF				0.077432 ± 0.000216			0.852064 ± 0.000825
	DNN				0.069457 ± 0.000342			0.880964 ± 0.001177
Last 20%	ExtraTrees	0.231393	0.095732	0.078776	0.194755 ± 0.000952	0.828836	0.884099	0.291586 ± 0.006932
	SVR-RFF				0.121407 ± 0.003349			0.724506 ± 0.015291
	DNN				0.125046 ± 0.005757			0.707345 ± 0.027032

Improved results on beam lifetime prediction: SVR-RFF and DNN with chronological split



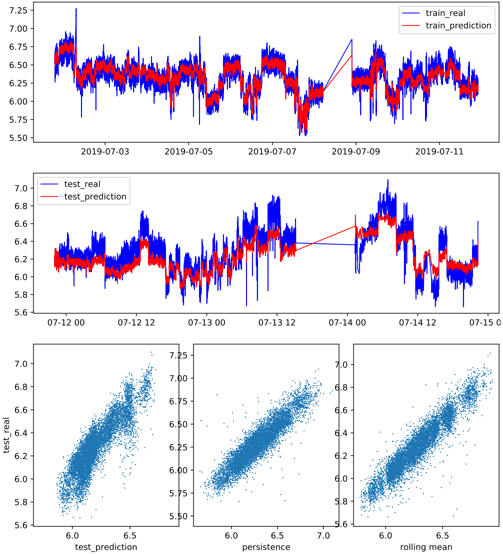


Table of Contents

- **Deep Deterministic Policy Gradient** (Lillicrap et al. (2016)): Actor-critic Reinforcement Learning algorithm for continuous environments.
- Off-policy data and the Bellman equation used to learn the Q-function.
- Q-function used to learn the policy.
- *Approximated* with NNs.

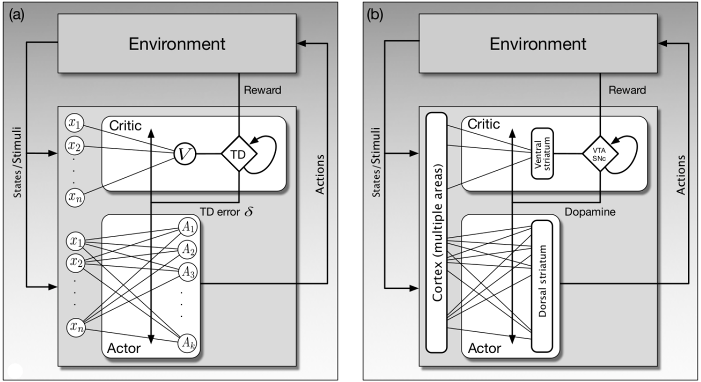
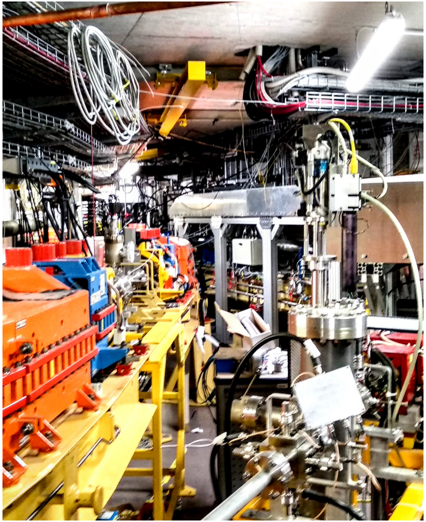


Figure: Sutton and Barto (2018)

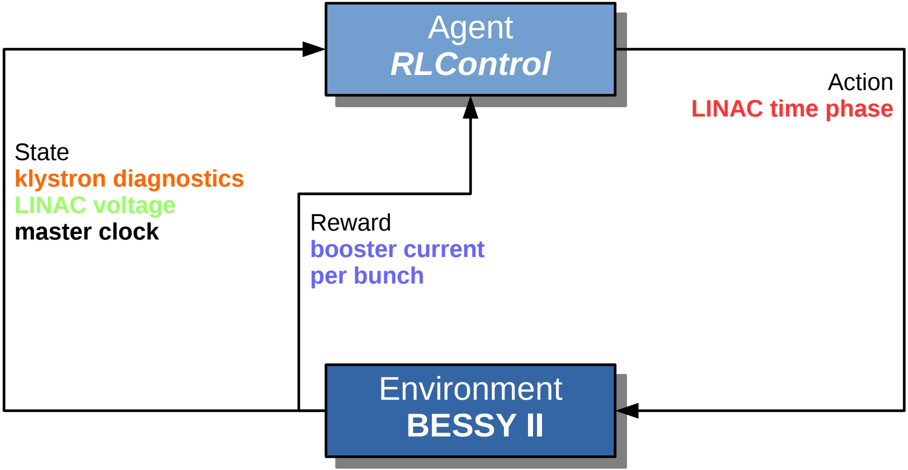
Table of Contents

Optimization of booster current: Case description

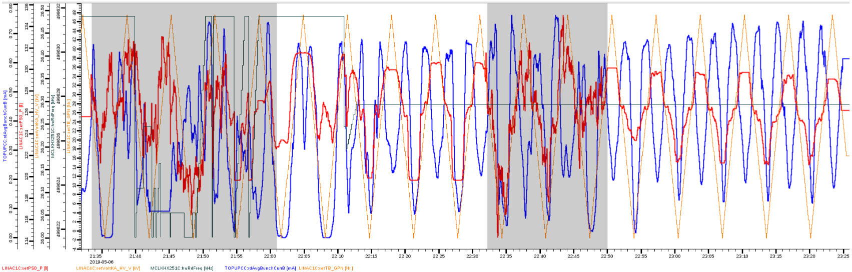
- Observation: after long interruptions of the machine operation, the booster current tends to be low. As for today, **manual parameter tuning** is required.
- State variables:
  - High (radio) frequency - master clock.
  - Voltage in LINAC.
  - Klystron current diagnostic measurements - only in last tests.
- Action variable: **time phase in LINAC.** Observations show that this parameter does not affect the injection efficiency.
- Reward: (normalized) **booster current per bunch.**

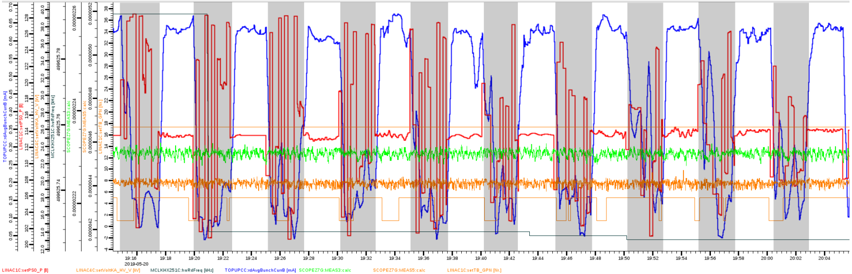


Self-optimization (RLControl): Case description

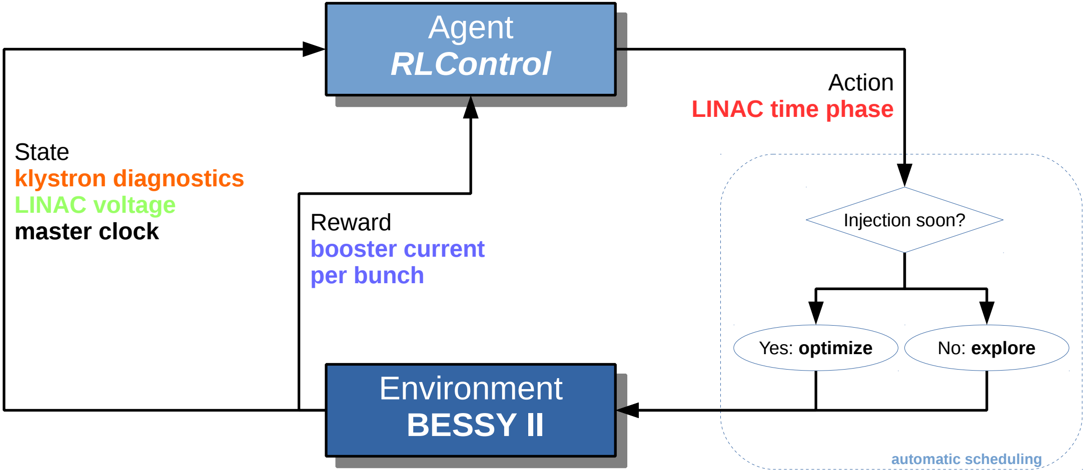


Optimization of booster current: Preliminary tests (already presented at AMALEA meeting)





Self-optimization (RLControl): Automatic scheduling



Optimization of booster current: Test during user operation

*Long test during user time with **automatic exploration schedule** (09/07/19)*.
Pre-training with 30 days of historical data. Pure **exploration** is scheduled to take place only in the **meantime between injections** during the first hour, in order to avoid disturbing user operation. **Optimization** is activated always **shortly before each injection**. The agent runs successfully during the next 8.5 hours.

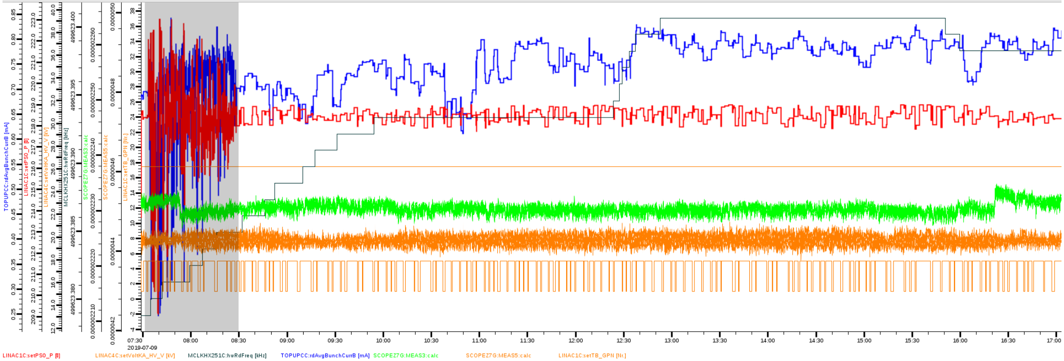
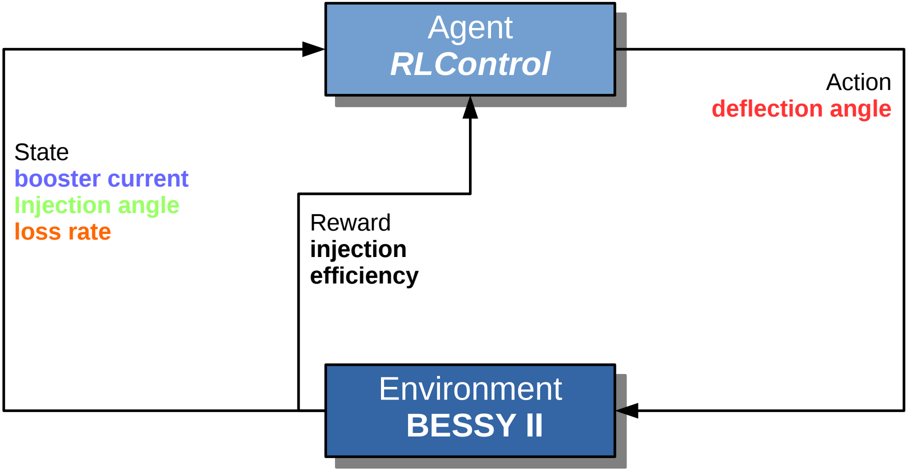


Table of Contents

Optimization of injection efficiency: Environment description

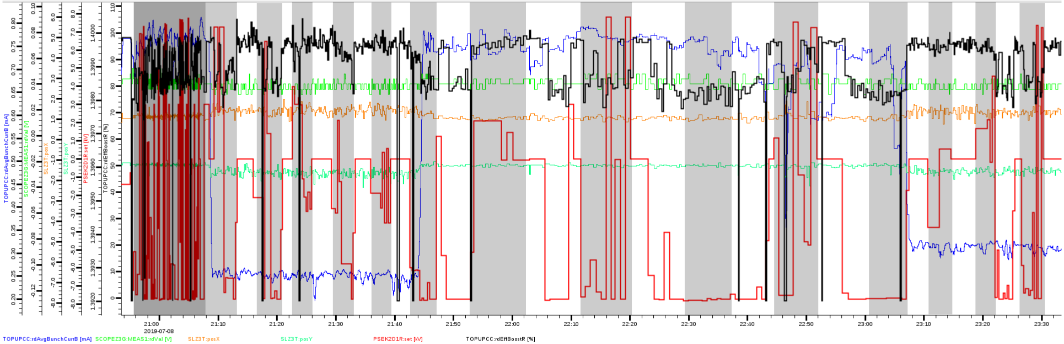
- Injection efficiency affected by temperature $\rightarrow$ needs manual tuning.
- State variables:
  - Number of bunches generated by the LINAC (1, 3 or 5)
  - Injection angle mismatch, measured by the horizontal and vertical beam position in the transfer line.
  - Current measured during the booster acceleration phase.
  - Measured loss rate after extraction from the booster.
- Action variable: **Deflection angle into the storage ring**, generated by the 2^{nd} ring septum sheet.
- Reward: **last injection efficiency**, measured as fraction of current increase generated in the storage ring by the charge accelerated in the booster.

Self-optimization (RLControl): Case description



Optimization of injection efficiency: Preliminary tests

Short test (08/07/19). Demonstration with historical data presented some problems, so the agent had to learn **from scratch**. The agent performed well during the first phase (only ca. 200 steps) and apparently found stable actions with good efficiency. After booster current was increased (ca 21:45) the optimization presented some ***pathological* behaviors**.



Optimization of injection efficiency: Issues

- Observation: the actor network gets stuck in local minima, producing constant (extreme) actions.
- Solution 1: accurate normalization - action boundaries that have been visited during the pretraining period.
- Solution 2: *irregularization* term in the actor loss to avoid *lazy* policies:

$$J^*_\beta(\mu_\theta) = \mathbb{E}_{s\sim\rho^\beta}\left[Q^{\mu_\theta}(s, \mu_\theta(s)) - \lambda \| e^{-[\nabla_\theta \mu_\theta(s)]^2} \|\right]$$

- Both solutions avoid constant actions during pretraining - it has to be tested whether it avoids the pathological behaviors observed during the experiments.
- Alternative: different approach to pretraining (e.g. Zhang and Ma (2018)).

Table of Contents

Further development

- **OCELOT surrogate models** (Agapov et al. (2014)) for the training of deep-RL agents, complementing or replacing the **pretraining** with historical data. Some tests with toy-examples (emittance, orbit-correction...) in small lattices have been already carried out - the major challenge is the *export* of a RL-agent trained with the **virtual BESSY-II-lattice** to the real accelerator.
- We are also investigating the possibility of using **Symplectic Networks** (Mattheakis et al., 2019) for tracking in the context of a student's thesis.
- Classification approach for prediction.
- Surrogate models.
- Bluesky integration in *RLControl*.
- User interfaces

Table of Contents

Bibliography I

Agapov, Ilya, Gianluca Geloni, Sergey Tomin, and Igor Zagorodnov (2014), "OCELOT: A software framework for synchrotron light source and FEL studies." *Nuclear instruments & methods in physics research / A*, 768, 151 – 156, URL `http://bib-pubdb1.desy.de/record/192826`. (c) Elsevier B.V.

Ba, Jimmy, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016), "Layer normalization." *ArXiv*, abs/1607.06450.

Breiman, Leo (2001), "Random forests." *Mach. Learn.*, 45, 5–32, URL `https://doi.org/10.1023/A:1010933404324`.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006), "Extremely randomized trees." *Machine Learning*, 63, 3–42, URL `https://doi.org/10.1007/s10994-006-6226-1`.

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016), "Continuous control with deep reinforcement learning." In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, URL `http://arxiv.org/abs/1509.02971`.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008), "Isolation forest." In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, 413–422, IEEE Computer Society, Washington, DC, USA, URL `http://dx.doi.org/10.1109/ICDM.2008.17`.

Bibliography II

Mattheakis, M, P Protopapas, David Sondak, M Di Giovanni, and Efthimios Kaxiras (2019), "Physical symmetries embedded in neural networks."

Rahimi, Ali and Benjamin Recht (2008), "Random features for large-scale kernel machines." In *Advances in Neural Information Processing Systems 20* (J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, eds.), 1177–1184, Curran Associates, Inc., URL `http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf`.

Rojas, Raul (1996), *Neural networks : a systematic introduction*. Springer-Verlag, Berlin ;;New York.

Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller (2014), "Deterministic policy gradient algorithms." In *Proceedings of the 31st International Conference on Machine Learning* (Eric P. Xing and Tony Jebara, eds.), volume 32 of *Proceedings of Machine Learning Research*, 387–395, PMLR, Bejing, China, URL `http://proceedings.mlr.press/v32/silver14.html`.

Smola, Alex J. and Bernhard Schölkopf (2003), "A tutorial on support vector regression." Technical report, STATISTICS AND COMPUTING.

Smola, Alexander Johannes (1998), "Learning with kernels."

Sutton, Richard S. and Andrew G. Barto (2018), *Reinforcement Learning: An Introduction*, second edition. The MIT Press, URL `http://incompleteideas.net/book/the-book-2nd.html`.

Bibliography III

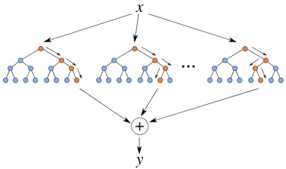
Vapnik, Vladimir N. (1995), *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg.

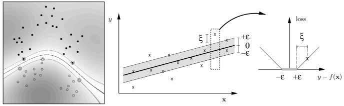
Zhang, Xiaoqin and Huimin Ma (2018), “Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations.” *CoRR*, abs/1801.10459, URL `http://arxiv.org/abs/1801.10459`.

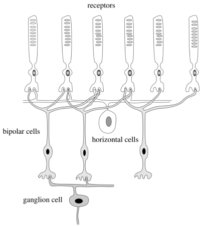
Table of Contents

Backup: Supervised learning algorithms

- **Ensemble methods**: Random Forests, Extremely Randomized Trees... (Breiman (2001), Geurts et al. (2006)). For regression, **MSE** as loss $\rightarrow$ variance as impurity measure. **Self-explaining**: allow individual analysis of each variable's behavior.
- **Support Vector Regression** Smola (1998) with **Random Fourier Features** (Rahimi and Recht (2008)). SVR extends traditional SVM (for classification) via Vapnik's ϵ-insensitive loss function (Vapnik (1995))
- **Neural Networks** (e.g. see Rojas (1996)). (Deep) Feed-forward NNs for regression (i.e. MSE as loss function).







Figs. from https://dsc-spidal.github.io/harp/docs/examples/rf/, Smola and Schölkopf (2003), Rojas (1996).

- Data preprocessing:
  - Outlier detection with Isolation Forest (Liu et al. (2008)) with contamination 0.02.
  - $[-1, 1]$ linear normalization.
  - PCA of the input variables (with 185 components).
- Hyperparameter optimization: grid-search with 5-folded cross validation.

Split	Algorithm	Grid-search CV	
		Chosen hyperparameter configuration	R^2 Score
Random	ExtraTrees	(bootstrap, *True), (max_depth, None), (max_features, None), (n_estimators, 500)	0.888633 $\pm$ 0.003072
	SVR-RFF	(batch_size, *32), (epochs, 50), (gamma, *1/n_atts), (loss, mse), (mode, rff), (n_components, 5000), (optimizer, adagrad)	0.860787 $\pm$ 0.002748
	DNN	(activation, relu), (batch_size, *32), (dropout_rate, 0.1), (epochs, *20), (hidden_layers, 200+200+100+50+25+12),(intermediate_dropouts, first), (loss, mse), (optimizer, adagrad)	0.883938 $\pm$ 0.003867
Chronological		(activation, *tanh), (batch_size, *32), (dropout_rate, 0.05), (epochs, *20), (hidden_layers, *200+200), (intermediate_dropouts, all), (loss, mse), (optimizer, adagrad)	-0.459200 $\pm$ 1.653572

Backup: (Deep) Reinforcement Learning

- **Bellman equation** with a deterministic target policy μ_θ:

$$Q^\mu(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim E}\left[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_t))\right]$$

- **Critic update** (parametrized approximation Q_ϕ) through SGD with loss:

$$L(\phi) = \mathbb{E}_{s_t\sim\rho^\beta, a_t, r_t, s_{t+1}\sim E}\left[\left(Q_\phi(s_t, a_t) - (r_t + \gamma Q_{\tilde{\phi}}(s_{t+1}, \mu_{\tilde{\theta}}(s_t))\right)^2\right]$$

- **Actor update** - (off-policy) Deterministic Policy Gradient Theorem ((Silver et al., 2014)): for a performance objective $J_\beta(\mu_\theta) = \mathbb{E}_{s\sim\rho^\beta}\left[Q^{\mu_\theta}(s, \mu_\theta(s))\right]$,

$$\nabla_\theta J_\beta(\mu_\theta) \approx \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_\theta\mu_\theta(s)\nabla_a Q^{\mu_\theta}(s, a)\big|_{a=\mu_\theta(s)}\right]$$

- Implementation tricks: delayed target networks ($Q_{\tilde{\phi}}$, $\mu_{\tilde{\theta}}$), replay buffer.

Optimization of injection efficiency: Settings

- Neural networks: in both cases, `relu` used as inner activation function and `adam` as optimizer (lr = 0.001).
  - Critic network: five hidden layers (25+50+25+10+5 neurons) and concatenates actions at the first hidden layer. Linear activation at the output layer.
  - Actor network: three hidden layers (25+10+5 neurons), all of them with layer normalization (Ba et al. (2016)). `tanh` used as activation for the output layer.
  - In the injection efficiency case, the number of neurons is doubled.
- Data preprocessing: $[-1, 1]$ linear normalization, historical data downsampled to 60 seconds.
- Parameter Space Noise: $\delta = 0.01$.
- Training parameters: $\gamma = 0.2$, pretraining with 10000 steps (2000 before actor training), warm-up with 32 steps, target model update rate = 0.1.
- *Brute-force* synchronization: update every **2 seconds** through EPICS.