Speeding up Hadron Correlator Calculations with Machine Learning

Giovanni Pederiva

4 March 2020

Michigan State University

Outline

Quark Propagators and Hadron Two-Point Correlation Functions

Hadron Correlators

An important hadronic observable is the two-point correlator. They can be interpreted via spectral decomposition [1]

$$C(t) = a^3 \sum_{\mathbf{x}} \langle O(t,\mathbf{x}) \bar{O}(0,\mathbf{0}) \rangle = \sum_k \langle 0| \hat{O} |k\rangle \langle k| \hat{\bar{O}} |0\rangle \, e^{-tE_k}$$

Where $O(t)$ is an interpolating operator of the desired hadron state. For example:

Pion: $O_{\pi^+}(x) = \bar{d}(x)_{\alpha,c} (\gamma_5)_{\alpha\beta} u(x)_{\beta,c}$

Proton: $O_P(x) = \epsilon_{abc} u(x)_{\alpha,a} \left(u(x)^T_{\beta,b} C(\gamma_5)_{\beta\gamma} d(x)_{\gamma,c} \right)$

[1] Notation taken from *C. Gattringer and C. B. Lang, "Quantum Chromodynamics on the Lattice", Springer, 2010*

The Quark Propagator

The term, for a generic quark flavor q

$$\langle q(x)_{\alpha,a}\bar{q}(y)_{\beta,b}\rangle = D^{-1}(y,x)^{\alpha\beta}_{ab}$$

is the inverse of the Dirac operator. In principle, one needs to invert the whole matrix. However, one can set:

$$\eta_{\beta,b}(0) = \delta_{b,c_1}\delta_{\beta,\alpha_1}\delta_{y,0}$$

A simple point-like source. The problem is then reformulated as:

$$D(0,x)^{\alpha\beta}_{ab}\, q(x)_{\alpha,a} = \eta_{\beta,b}(0)$$

Equivalent to computing only one column of the inverse matrix.

A Linear System

The problem is now reduced to a linear system of the very simple form $Dq = \eta$, where η are $3 \times 4 = 12$ different source vectors (Dirac and color indices).

The matrix D is the Dirac operator, a very sparse matrix (its exact form depends on the lattice action).

Note: for every quark flavor, we have an ensemble of linear systems.

Iterative Solvers

This kind of linear systems is usually solved using iterative methods. One of the simplest ones it the Conjugate Gradient, but many variations are used. For example, the BiCGStab is a common choice because it works for non-hermitian operators.

ALGORITHM 1 (BICGSTAB). *For solving $Az = b$ choose an initial approximation $z_0 \in \mathbb{C}^N$ and set $\tilde{r}_0 := \tilde{s}_0 := b - Az_0$. Choose $y_0 \in \mathbb{C}^N$ such that $\tilde{\delta}_0 := y_0^H \tilde{r}_0 \neq 0$ and $\varphi_0 := y_0^H A\tilde{s}_0/\tilde{\delta}_0 \neq 0$. Then compute for $n = 0, 1, \dots$*

$$\omega_n := 1/\varphi_n \,, \tag{25a}$$

$$\tilde{w}_{n+1} := \tilde{r}_n - A\tilde{s}_n\omega_n \,, \tag{25b}$$

$$\chi_n := (A\tilde{w}_{n+1})^H \tilde{w}_{n+1}/||A\tilde{w}_{n+1}||^2 \,, \tag{25c}$$

$$\tilde{r}_{n+1} := \tilde{w}_{n+1} - A\tilde{w}_{n+1}\chi_n \,, \tag{25d}$$

$$z_{n+1} := z_n + \tilde{s}_n\omega_n + \tilde{w}_{n+1}\chi_n \,, \tag{25e}$$

$$\tilde{\delta}_{n+1} := y_0^H \tilde{r}_{n+1} \,, \tag{25f}$$

$$\psi_{n+1} := -\omega_n\tilde{\delta}_{n+1}/(\tilde{\delta}_n\chi_n) \,, \tag{25g}$$

$$\tilde{s}_{n+1} := \tilde{r}_{n+1} - (\tilde{s}_n - A\tilde{s}_n\chi_n)\,\psi_{n+1} \,, \tag{25h}$$

$$\varphi_{n+1} := y_0^H A\tilde{s}_{n+1}/\tilde{\delta}_{n+1} \,. \tag{25i}$$

These iterative solvers are terminated at convergence, when for a small parameter ϵ:

$$||Ax_n - b|| < \epsilon$$

Some Machine Learning

Machine Learning Algorithms

Machine Learning, in particular Supervised Learning, can be used to build predictive models from data. It is mostly useful when the correlation between two sets of data is hard to define functionally.

Multiple methods exist, for example:

- Linear regression: very simple, few parameters, can capture simple correlations.

Machine Learning Algorithms

Machine Learning, in particular Supervised Learning, can be used to build predictive models from data. It is mostly useful when the correlation between two sets of data is hard to define functionally.

Multiple methods exist, for example:

- Linear regression: very simple, few parameters, can capture simple correlations.
- Boosted Decision Trees: moderately simple piece-wise constant function, more parameters.

Machine Learning Algorithms

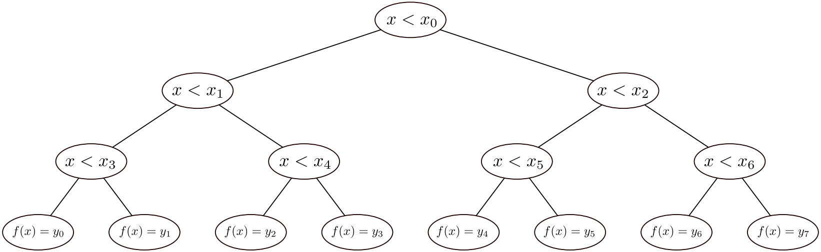
Machine Learning, in particular Supervised Learning, can be used to build predictive models from data. It is mostly useful when the correlation between two sets of data is hard to define functionally.

Multiple methods exist, for example:

- Linear regression: very simple, few parameters, can capture simple correlations.
- Boosted Decision Trees: moderately simple piece-wise constant function, more parameters.
- Neural Networks: complicated functional form, many parameters, can capture very complicated correlations.

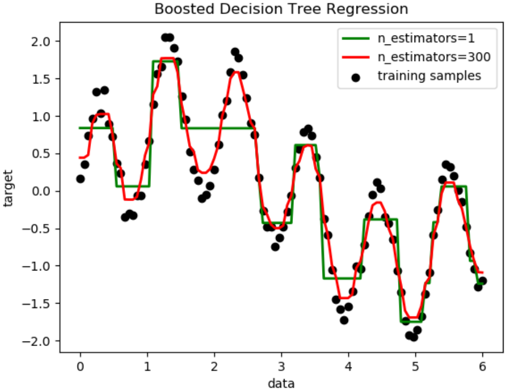
Boosted Decision Trees

Each tree is a simple binary split tree that leads to a piece-wise constant function. Parametrized by the split points x_i and the constant values y_j:



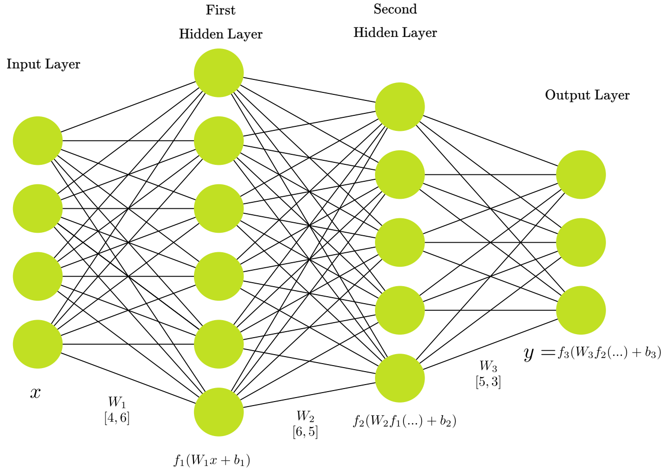
Boosted Decision Trees

A linear combination of piece-wise constant functions, where both the section bounds and constant values are parameters. [2]



[2] Image from `scikit-learn.org/`

Neural Networks



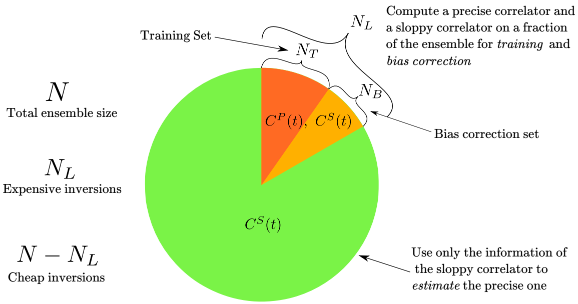
A chain of affine transformations and non-linear functions f at each layer. Universal approximant of functions $\mathbb{R}^n \to \mathbb{R}^m$.

ML for Two-Point Correlators

The Goal

The main idea of this work is to try to accelerate the computation of the linear system for the quark propagator. We use numerical data for different stopping parameters ϵ to as training and prediction data sets.

For example, using a precise measurement of the propagator ($\epsilon = 10^{-8}$) on a subset of the ensemble and a less precise (sloppy) one ($\epsilon = 10^{-1}, 10^{-2}, 10^{-3}$) on the whole ensemble.



To properly estimate the uncertainty bias-correction and boostrap are used.

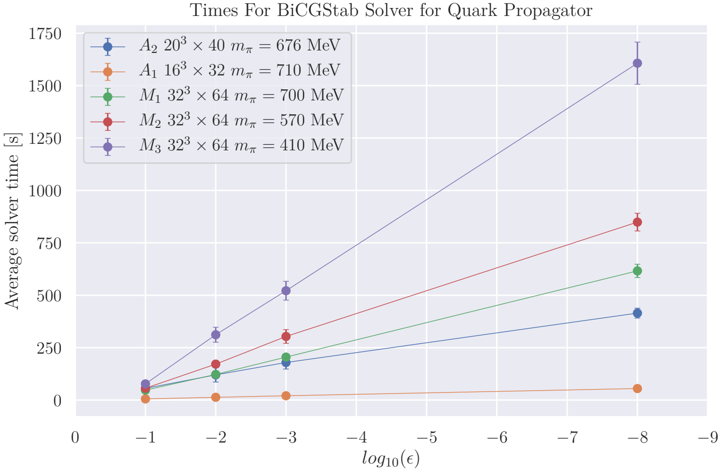
Gauge Field Ensembles Used

	β	κ_l	κ_s	L/a	T/a	$a\ [fm]$	$m_\pi\ [MeV]$	N
M_1	1.90	0.13700	0.1364	32	64	0.0907(13)	699.0(3)	399
M_2	1.90	0.13727	0.1364	32	64	0.0907(13)	567.6(3)	400
M_3	1.90	0.13754	0.1364	32	64	0.0907(13)	409.7(7)	450
A_1	1.83	0.13825	0.1371	16	32	0.1095(25)	710(1)	800
A_2	1.90	0.13700	0.1364	20	40	0.0936(33)	676.3(7)	790

Ensembles from the PACS-CS collaboration[3], with clover fermions.
Physical quantites calculated for another work[4]

[3] PACS-CS, S. Aoki et al., Phys. Rev.D79, 034503 (2009), 0807.1661
[4] J. Dragos, A. Shindler et al., (2019), arXiv:1902.03254v2

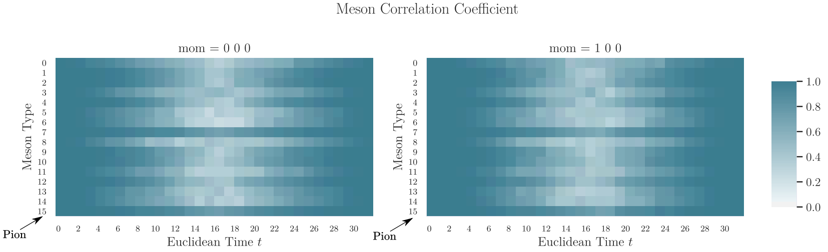
Times



Times For BiCGStab Solver for Quark Propagator
A_2 $20^3 \times 40$ $m_\pi = 676$ MeV
A_1 $16^3 \times 32$ $m_\pi = 710$ MeV
M_1 $32^3 \times 64$ $m_\pi = 700$ MeV
M_2 $32^3 \times 64$ $m_\pi = 570$ MeV
M_3 $32^3 \times 64$ $m_\pi = 410$ MeV
Average solver time [s]
1750
1500
1250
1000
750
500
250
0
0
−1
−2
−3
−4
−5
−6
−7
−8
−9
$log_{10}(\epsilon)$

Correlations Maps

Meson Correlation Coefficient:

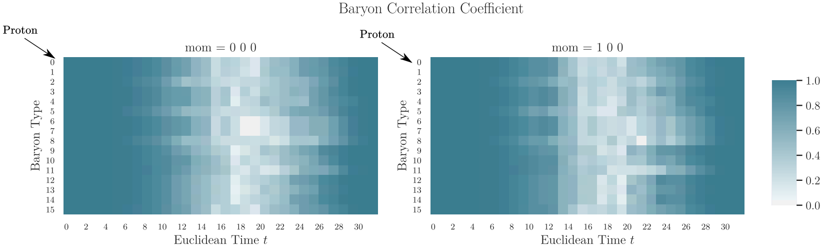


$$\Gamma(P,S) = \frac{1}{N\sigma_P\sigma_S}\sum_i^N (C_i^P - \bar{C}^P)(C_i^S - \bar{C}^S)$$

Correlation between sloppy meson correlator data and precice meson correlators. Calculated on ensemble A_1, between $\epsilon = 10^{-2}$ and $\epsilon = 10^{-8}$.

Correlations Maps

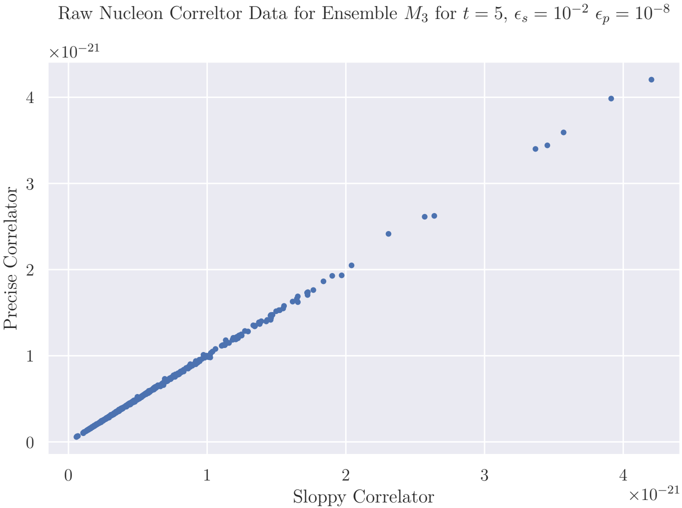
Baryon Correlation Coefficient:



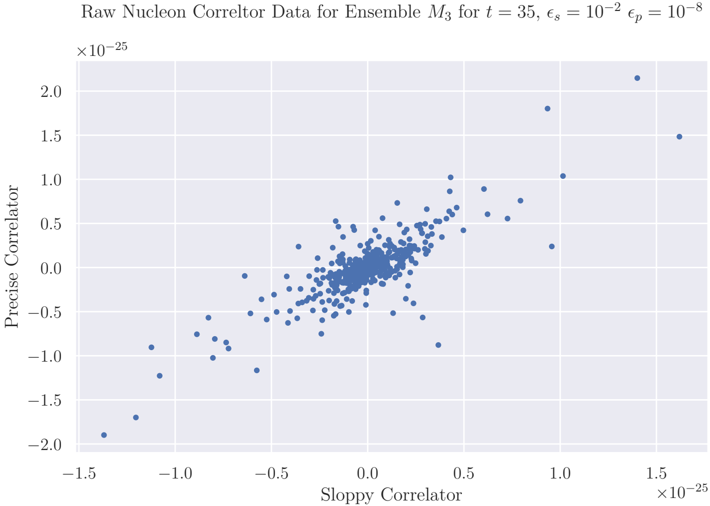
$$\Gamma(P,S) = \frac{1}{N\sigma_P\sigma_S}\sum_i^N (C_i^P - \bar{C}^P)(C_i^S - \bar{C}^S)$$

Correlation between sloppy baryon correlator data and precice baryon correlators. Calculated on ensemble A_1, between $\epsilon = 10^{-2}$ and $\epsilon = 10^{-8}$.

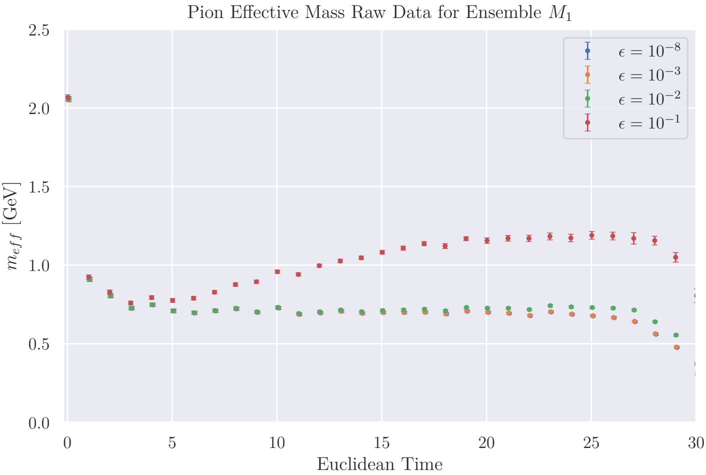
Correlation Between Different Precisions



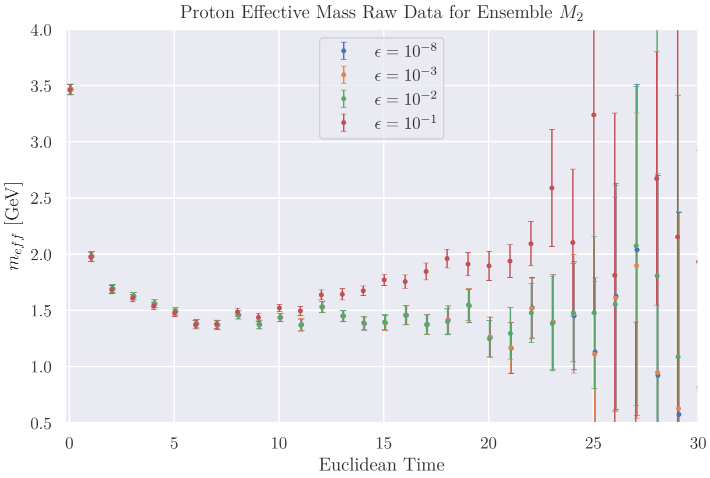
Correlation Between Different Precisions



Example Effective Mass

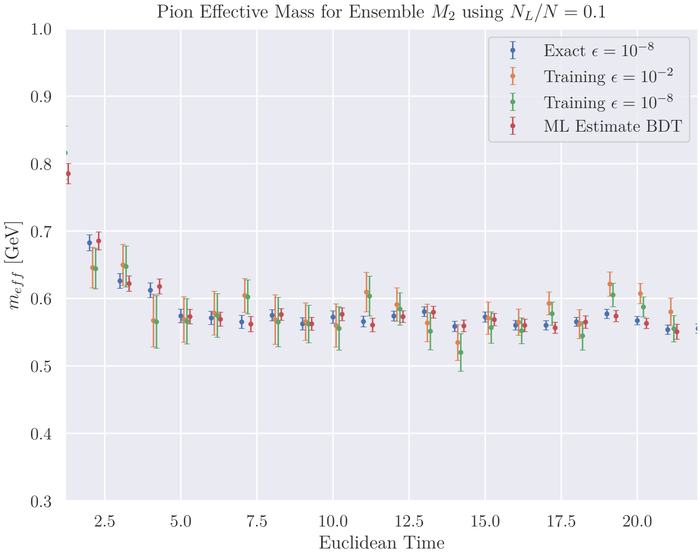


Example Effective Mass

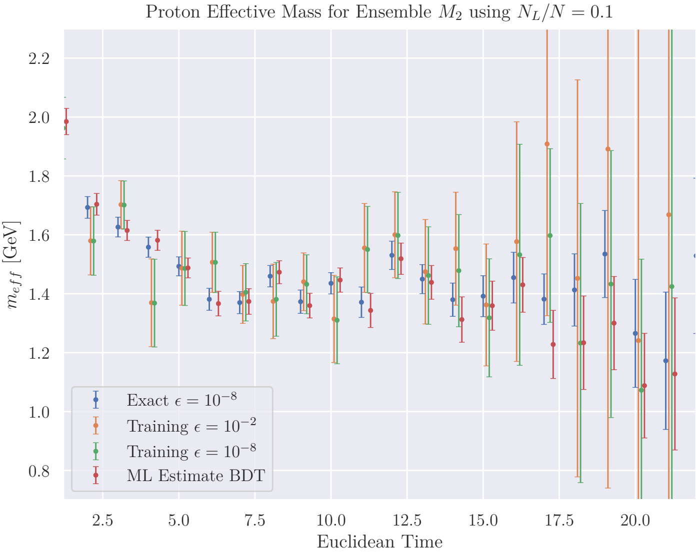


Results

Example Effective Mass from ML



Example Effective Mass from ML



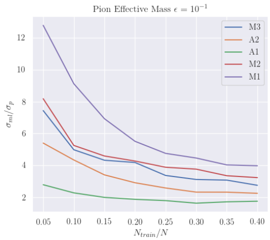
Performance Analysis

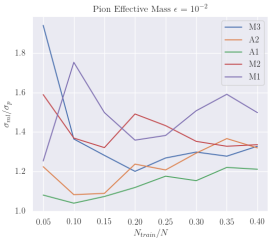
Ens.	ϵ	N	N_L	t_s	t_p	τ_{ML}	Pion $\sigma(m_{eff}^{ML})/\sigma(m_{eff})$	Nucl. $\sigma(m_{eff}^{ML})/\sigma(m_{eff})$	Pion h	Nucl. h
A_1	10^{-1}	800	240	5.497	54.958	0.370	1.628	2.185	0.981	1.767
A_1	10^{-2}	800	240	13.238	54.958	0.469	1.154	1.073	0.624	0.539
A_1	10^{-3}	800	240	20.542	54.958	0.562	1.162	1.088	0.759	0.665
A_2	10^{-1}	790	237	55.855	414.755	0.394	2.322	2.263	2.127	2.018
A_2	10^{-2}	790	237	120.163	414.755	0.503	1.293	1.320	0.841	0.876
A_2	10^{-3}	790	237	179.169	414.755	0.602	1.252	1.191	0.944	0.855
M_1	10^{-1}	399	119	47.269	616.181	0.354	4.462	2.210	7.041	1.728
M_1	10^{-2}	399	119	122.318	616.181	0.439	1.508	1.409	0.998	0.872
M_1	10^{-3}	399	119	204.950	616.181	0.533	1.355	1.302	0.978	0.904
M_2	10^{-1}	400	120	56.410	848.848	0.347	3.765	2.590	4.911	2.325
M_2	10^{-2}	400	120	171.468	848.848	0.441	1.353	1.142	0.808	0.576
M_2	10^{-3}	400	120	303.351	848.848	0.550	1.181	1.083	0.767	0.646
M_3	10^{-1}	450	135	77.236	1606.996	0.334	3.116	2.536	3.239	2.145
M_3	10^{-2}	450	135	311.664	1606.996	0.436	1.299	1.349	0.735	0.793
M_3	10^{-3}	450	135	521.557	1606.996	0.527	1.222	1.231	0.787	0.799

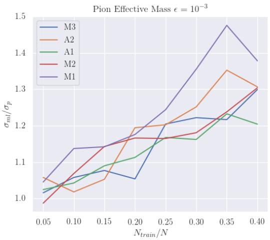
Where we have defined the scaled time τ^{ML} and the overall gain h as:

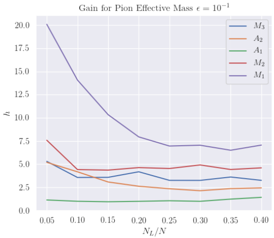
$$\tau^{ML} = \frac{t_s \cdot N_L + t_p \cdot N_P}{t_p \cdot N} \leq 1 \qquad h = \left(\frac{\sigma^{ML}_{m_{eff}}}{\sigma_{m_{eff}}}\right)^2 \cdot \tau^{ML}$$

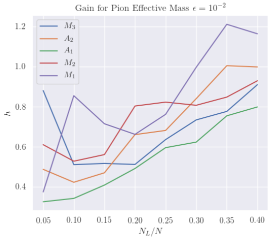
Performance Analysis

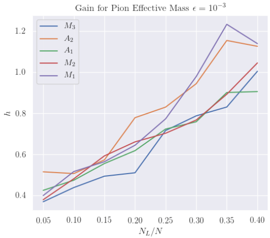










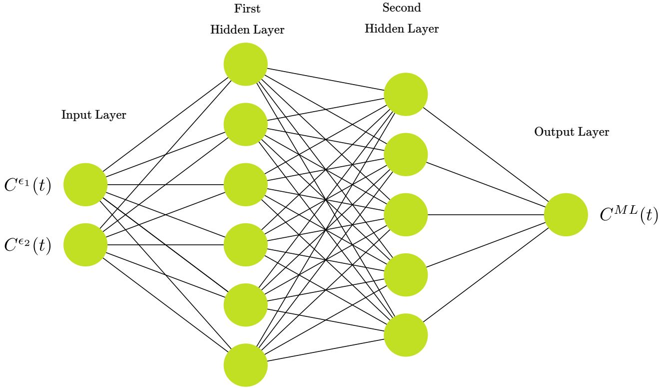


Using more information at once

As a second step, one could try to use more information at the same time. In particular we construct a function to approximate the precise data:

$$C^P(t) \approx \Gamma^{ML}\left(C^{\epsilon_1}(t), C^{\epsilon_2}(t), ..., C^{\epsilon_n}(t)\right)$$

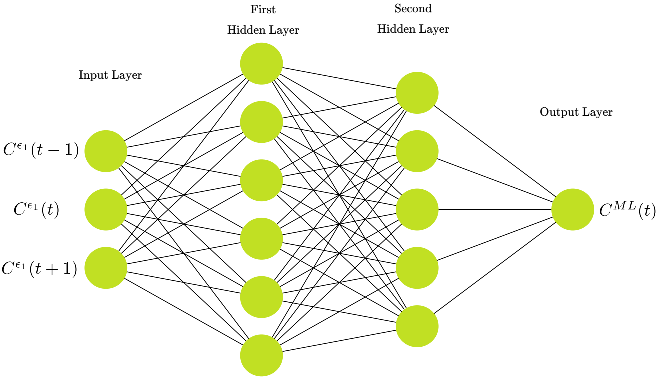
where $C^{\epsilon_i}(t)$ is the correlator at precision $\epsilon = 10^{-i}$.



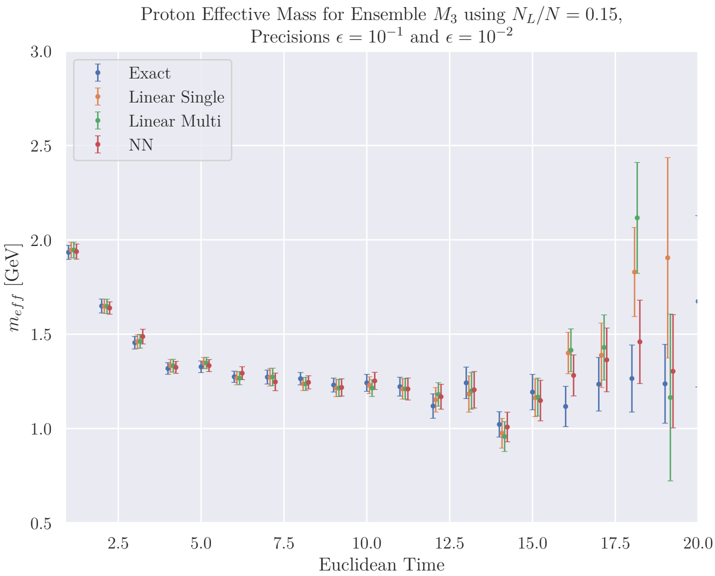
Using more information at once

Furthermore we can define

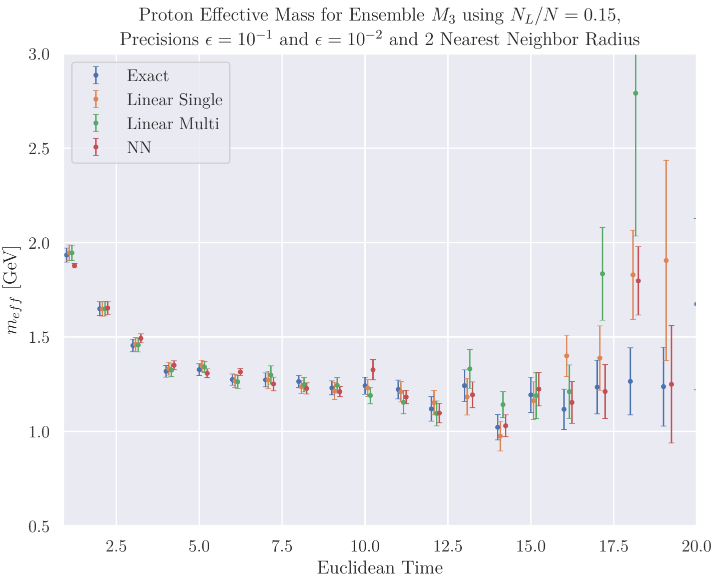
$$C^P(t) \approx \Gamma^{ML}\left(C^{\epsilon_1}(t), C^{\epsilon_2}(t), ..., C^{\epsilon_n}(t), C^{\epsilon_1}(t \pm 1), C^{\epsilon_2}(t \pm 1), ..., C^{\epsilon_n}(t \pm 1), ...\right)$$



Using more information at once

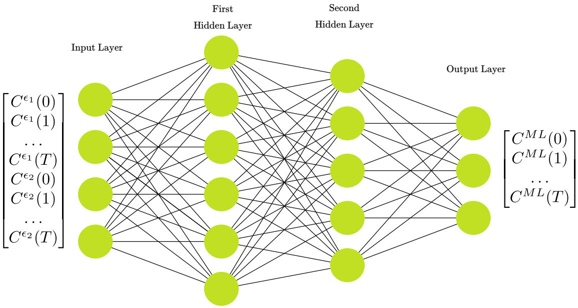


Using more information at once, nearest neighbors



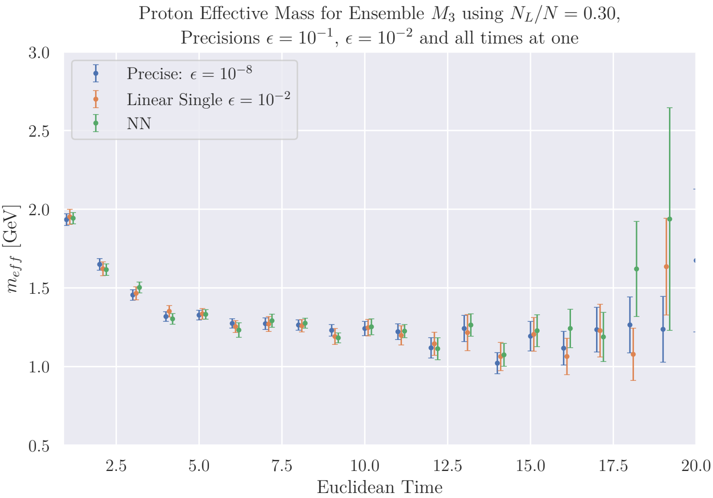
NN for Full Correlator

One further try could be to construct a Neural Network that can directly compute the correlator at all euclidean time at the same time.

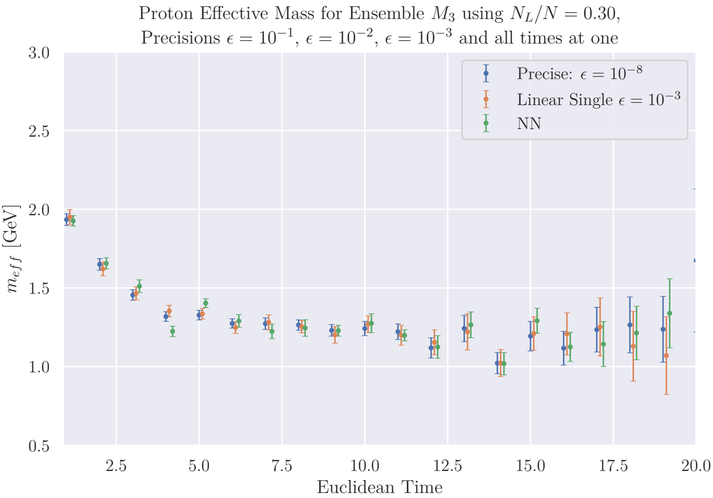


Some issues here are the increasing number of parameters to handle, the normalization of the data (correlators are exponentially decreasing with euclidean time) and the possible correlation of the data at different euclidean times in the output.

NN for Full Correlator



NN for Full Correlator



Further Work

- Define a reliable set of hyper-parameters that is usable independently of the ensemble properties

Further Work

- Define a reliable set of hyper-parameters that is usable independently of the ensemble properties
- Better test the stability as a function of the lattice spacing and quark masses

Further Work

- Define a reliable set of hyper-parameters that is usable independently of the ensemble properties
- Better test the stability as a function of the lattice spacing and quark masses
- Extend to study large volumes

Further Work

- Define a reliable set of hyper-parameters that is usable independently of the ensemble properties
- Better test the stability as a function of the lattice spacing and quark masses
- Extend to study large volumes
- Possibly define cuts for different algorithms in the euclidean time domain (linear for small t, NN for $t \approx T/2$)

Potential Impact

- Significant speedup of the calculations for quark propagators (we have time gain of ≈ 2 for $\epsilon = 10^{-3}$ and $N_L/N = 30\%$)

Potential Impact

- Significant speedup of the calculations for quark propagators (we have time gain of ≈ 2 for $\epsilon = 10^{-3}$ and $N_L/N = 30\%$)
- Possible adaptation to improve the speed of the calculation for pseudo-fermions in the HMC algorithm.

Potential Impact

- Significant speedup of the calculations for quark propagators (we have time gain of ≈ 2 for $\epsilon = 10^{-3}$ and $N_L/N = 30\%$)
- Possible adaptation to improve the speed of the calculation for pseudo-fermions in the HMC algorithm.
- Could enable the calculations on large lattices, where the Dirac operator is prohibitevely large.

Acknowledgments

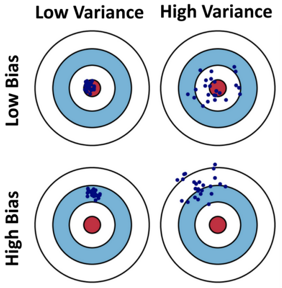
- This work is done in collaboration with A. Shindler
- We thank A. Bazavov, D. Lee, M. Rizik and J. Weber for the discussions
- The computational resources were provided by ICER at MSU

Thank You

Backup Material

Bias correction

When fitting, there could be some bias on the sample average depending on the subset used for training:



So we further split our training data set and compute the expectation value as:

$$\bar{C} = \frac{1}{N - N_L} \sum_{i \in prediction} C_i^P + \frac{1}{N_B} \sum_{i \in bias_c orr} (C_i - C_i^P)$$

Bootstrapping

To estimate the error on the expectation value of the observable, multiple bootstrap samples are used.

Boostrapping is a common resampling method used in LQCD analysis. It consists of taking a random sample of a quantity O from a given set of N data with repetitions. This is performed K times:

$$C_k = \frac{1}{N} \sum_{i}^{N} C_i^*$$

One then sets the estimator of O as:

$$\bar{C} = \frac{1}{K} \sum_{i}^{K} C_k, \qquad \sigma_C^2 = \bar{C} = \frac{1}{K} \sum_{i}^{K} (\bar{C} - C_k)^2$$

The training and prediction set are bootstrapped independently.