Invariant mass evaluation for e^+e^- events

Optimization of a Neural Network regression model

Durando Matteo

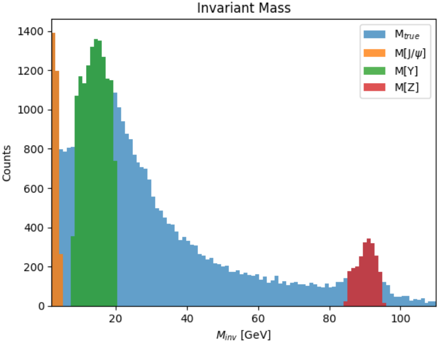
Machine Learning for Applied Physics and High Energy Physics

UniTO - A.A. 2023/2024

Index

Data and goal

Data used in this work are taken from a selection of *pp* collisions simulated for the LHC experiments resulting in e^+e^- measured in the detector.
The goal of the project is to build a model able to make a prediction of the invariant mass of the couple e^+e^- generated in the range $2 - 110$GeV. This is particularly important to identify hadronic resonances produced in the collisions. Since the final state has no electrical charge and produces 2 leptons ($+$ neutrinos eventually since it is not back-to-back emission) the resonance produced must be a neutral meson or boson.

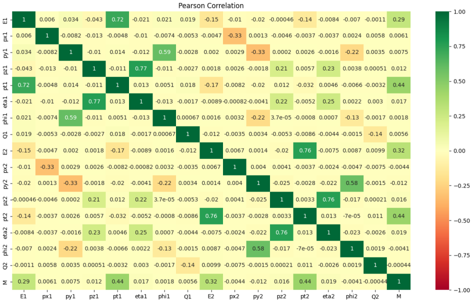


Meson	Theoretical Mass
J/ψ	~ 3 GeV
$\Upsilon(1,2,3S)$	~ 10 GeV
Z	~ 91 GeV

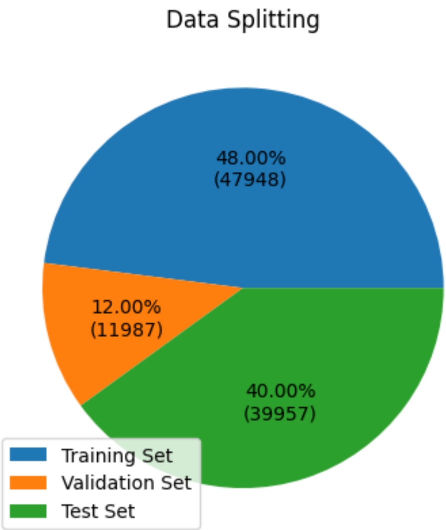
Name	Physical quantity
Run, Event	Event identifiers (not used for model evaluation)
E1, E2	Total energies [GeV]
px1, py1, pz1, px2, py2, pz2	Components of the 4-momenta [GeV]
pt1, pt2	Transverse momenta [GeV]
eta1, eta2	Pseudorapidities
phi1, phi2	Azimuthal angles
Q1, Q2	Charge of the particles (e^+/e^-)
M	Invariant mass [GeV]

No linear correlation between features and target

- $\Rightarrow$ Difficult to solve with analytical techniques
- $\Rightarrow$ Need to develop non-linear models



Dataset - PreProcessing



- *Data cleaning*: delete rows with NaN mass and duplicated events
- *Splitting data*: divide training, validation and test sets
- *Features normalization*: apply MinMaxScaler (range 0-100)

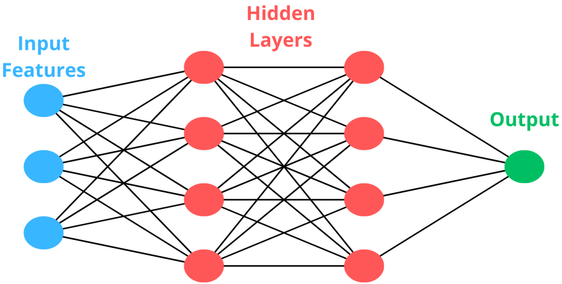
Model Architecture - Shallow Neural Network

Advantages

- General purpose model
- Faster training compared to deep learning models
- Reduce overfitting for simple taget functions

Disadvantages

- Possibly not the most CPU efficient model (e.g. Decision Trees)
- Need to find the correct architecture (layers/nodes)



Model Architecture - Basic Design

Parameter	Choice
Activation	ReLU
Initializer	Glorot-Bengio
Optimizer	AdamW
Loss/Metric	MeanSquaredError
Batch size	32

Glorot-Bengio Initializer

Initial weights are drawn from a uniform normalized distribution to avoid extreme results during parameters training

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

AdamW implements weight decay for Adam optimizer

$\Rightarrow$ Avoid large weights

$\Rightarrow$ Significant improvements in generalization performance

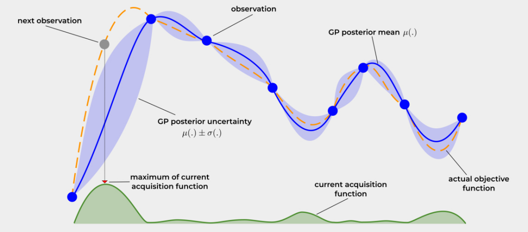
Algorithm 2 Adam with L_2 regularization and Adam with weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, w \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{x}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{x}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{x}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{x}_{t-1})$ $+w\boldsymbol{x}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1-\beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) +w\boldsymbol{x}_{t-1}\right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{x}_t$

Model Architecture - Hyperparameters Tuning

Bayesian Optimization

Sequential design strategy for global optimization of black-box functions that does not assume any functional forms



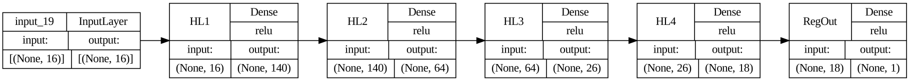
Research algorithm:

1. Define range of parameters $\Rightarrow$ Parameters space is limited
2. Define a surrogate gaussian model for the objective function
3. Pick some random sets of hyperparameters to evaluate the model
4. Update hyperparameters by evaluating the covariance matrix of the model $\Rightarrow$ Balance exploitation and exploration

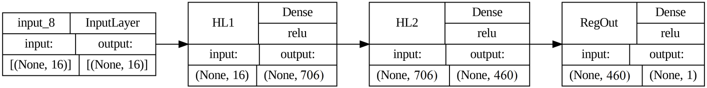
Model Architecture - Hyperparameters Results

HyperPar	Unconstrained Model		Constrained Model	
	Range	Best values	Range	Best values
N Layer	1 - 4	2	2 - 4	4
Node 0	1 - 1000	706	90 - 140	140
Node 1	1 - 1000	460	50 - 80	64
Node 2	1 - 1000	/	20 - 40	26
Node 3	1 - 1000	/	10 - 20	18
LearningRate	10^{-4} - 10^{-3}	$5 \cdot 10^{-4}$	10^{-4} - 10^{-3}	$5 \cdot 10^{-4}$
RMSE [GeV]	0.74 ± 0.09		0.80 ± 0.08	
Exec Time [s]	2700 ± 400		980 ± 300	

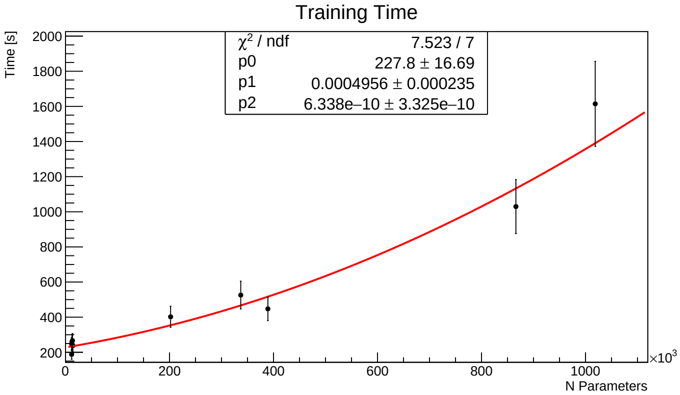
Small Layers Model



Large Layers Model

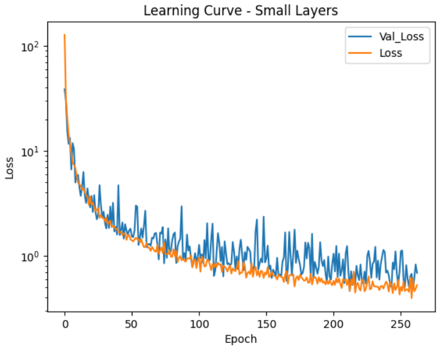


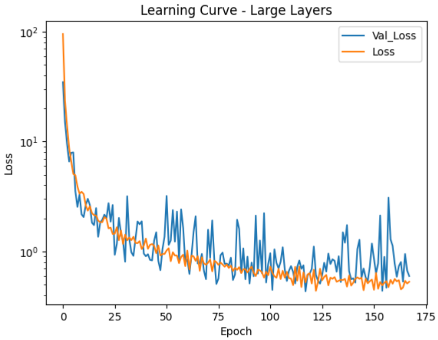
Model Architecture - Training Time scaling



Training time scales as $\mathcal{O}\left(N_{par}^2\right)$ while the performance improvement is small
$\Rightarrow$ Large models are inefficient for linear regression tasks

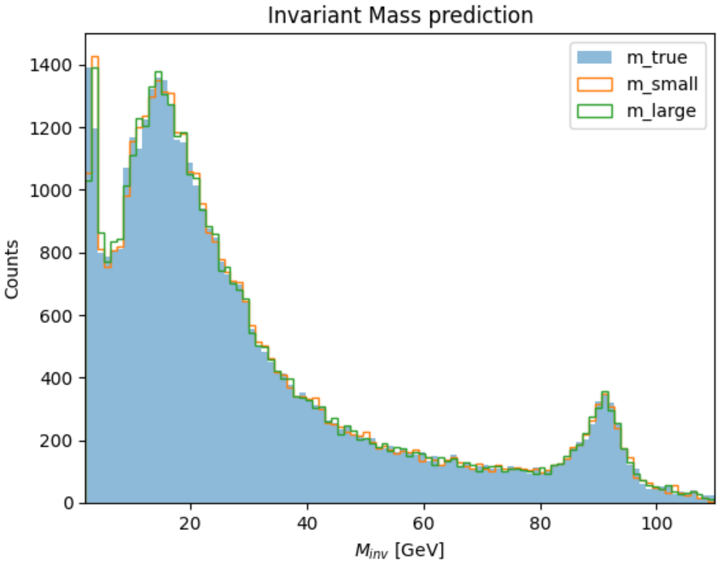
Results - Training





Oscillation in the loss function evaluation is caused by the presence of local minima during the optimisation of the parameters.
EarlyStopping criteria (patience = 50 epochs) were implemented on both `loss` and `val_loss` functions to avoid getting stuck in local minima.

Results - Invariant mass prediction



Conclusions

Achieved goals:

- Testing different models proved that shallow models are more efficient for linear regression tasks thanks to the limited number of free parameters to train
- Hyperparameters tuning resulted in significant improvements for the training of the model (especially for learning rate)
- The achieved error (RMSE $\leq 0.8 GeV$) is coherent with the experimental error usually found for $M_{inv} \gtrsim 10 GeV$ ($\Delta E/E < 10\%$)

Possible improvements:

- Compare other algorithms to improve performance and/or efficiency (e.g. DecisionTrees)
- Try different features (e.g. invariants)
- Dataset with experimental error on the features may result in better performance

References

[1] T. McCauley, “Events with two electrons from 2010,” (2014), [Online]. Available: http://doi.org/10.7483/OPENDATA.CMS.PCSW.AHVG.

[2] H. Nguyen, “Improving neural network’s performance with bayesian optimization,” (2020), [Online]. Available: https://medium.com/@hiepnguyen034/improving-neural-networks-performance-with-bayesian-optimization-efbaa801ad26.

[3] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,”, 2019. arXiv: 1711.05101 [cs.LG].

[4] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,”, Proceedings of Machine Learning Research, vol. 9, Y. W. Teh and M. Titterington, Eds., pp. 249–256, 13–15 May 2010. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html.