Report

A study of machine learning algorithms for reconstruction of missing energy in high energy physics experiments.

Max Isacson, max.isacson@physics.uu.se
Mikael Mårtensson, mikael.martensson@physics.uu.se
Camila Rangel Smith, camila.rangel@physics.uu.se
Henrik Öhman, ohman@cern.ch

April 21, 2016

1 Introduction and Objective

High energy physics experiments have to deal with the so-called ”missing energy”, which refers to energy coming from particles that do not interact with the particle detector. This particles are usually weakly interacting neutrinos, but it could also come from physics beyond the Standard Model.

This missing energy is estimated by requiring conservation of energy an momentum in the system, and is reconstructed in the transverse plane. Although this quantity is generally attributed to undetected particles, apparent missing energy may also be caused by mis-measurement on the energy/momentum of the detected particles, this affects the precision on the measurement of the truth missing energy coming from undetected particles.

In this project we plan to improve the reconstruction of the missing energy by predicting the real contribution from undetected particles. Having a more precise measurement of these contributions could improve the sensitivity of physics to new physics that could be observed in the Large Hadron Collider.

2 The Dataset

2.1 Structure

The dataset consists of simulated pp collision events, in which a charged Higgs is produced and decays as $H^+ \to \tau\nu$. Each event is described by one set of observable variables and one set of unobservable variables.

Observable variables:

- E_x^{miss}, E_y^{miss}— The x- and y-components of the missing energy.

- $P_{\tau_{\text{vis.}}}$ — The 4-momentum of the visible (hadronic) part of the τ decay.
- P_{b_0}, P_{b_1}, P_{q_0}, P_{q_1} — The 4-momenta of the two b-jets and two light jets.

Unobservable variables:

- P_{ν_τ} — The 4-momentum of the neutrino from the charged Higgs decay.
- $P_{\bar{\nu}_\tau}$ — The 4-momentum of the neutrino from the τ decay.
- For simplicity, in this particular study it is assumed that the contribution from the neutrino from the τ decay to the total missing energy is negligible, and only focus on the neutrino from the charged Higgs decay as the unobserved variable.

2.2 Production

`MG5_aMC@NLO` [1] is used for the matrix element computation of $gg/q\bar{q} \to H^+$ and the event simulation. The events are then passed to `PYTHIA8` [2] for the showering and hadronization, and for the $H^+ \to \tau\nu$ decay. Finally, the detector response is simulated using `DELPHES` [3] with an ATLAS-like geometry.

2.3 Estimated and true quantities

The true quantities for both the observable and the unobservable variables are available as output from the event simulation. The estimated values for the observable variables are reconstructed from the output from the detector response simulation.

3 Methods and Implementation

3.1 Strategy

The neutrino originating from the decay of the Higgs boson will carry the major part of the missing energy in the events. We can approximate the missing transverse energy ($E_{\mathrm{T}}^{\mathrm{miss}}$) by the true transverse momentum (p_{T}) of the neutrino from the Higgs boson decay. By using this quantity as the target parameter for the training, we aim to find a better estimate of the missing transverse energy. This will result in increased discriminating power of signal from background, since the variables used for discrimination are strongly dependent on this quantity.

3.2 Feature selection

In order to select the variables that bring most information into the training, a feature selection was performed based on the p-value of each feature with respect to the target. The best variables in this case are the following:

- E_x^{miss}, E_y^{miss}.

- $P_{\tau_{\text{vis.}}}$
- P_{b_0}, P_{q_0}, P_{q_1} .
- Number of reconstructed b-jets in the event.

All algorithms presented in this section use these features as inputs in the prediction of the transverse momentum from the neutrino.

3.3 Regression optimization

For all methods in the following a simple scheme is used to optimize their hyperparameters. We want to predict the p_{T} of the τ neutrino coming from the Higgs boson decay, and the corresponding reconstructed quantity that we want to compare with is the missing transverse energy $E_{\text{T}}^{\text{miss}}$. In order to judge the quality of the prediction we use two figures of merit:

- the resolution of the predicted neutrino p_{T} compared to the resolution of the $E_{\text{T}}^{\text{miss}}$, where the resolution is defined as $\frac{x - p_{\text{T}}{}_{\nu}^{\text{truth}}}{p_{\text{T}}{}_{\nu}^{\text{truth}}}$, and x is $p_{\text{T}}{}_{\nu}^{\text{predicted}}$ or $E_{\text{T}}^{\text{miss}}$
- the shape of the predicted neutrino p_{T} compared to the true neutrino p_{T} and the $E_{\text{T}}^{\text{miss}}$; we want the shape to ”make physical sense”

3.4 Bayesian Ridge Regression

A simple approach using linear regression techniques to predict the neutrino transverse moment was investigated. The algorithm with best performance is the Bayesian Ridge Regression, which introduces a zero mean spherical gaussian governed by a single precision parameter as prior probability distribution over the model parameters. The prior and noise parameter are estimated jointly during the fit to the data in several iterations. For an optimal result, 200 iterations are performed, it is found that a larger number of iterations does not improve the prediction.

The result presented here uses `Python` with `Numpy` [4], and `Pandas` [5] for general data processing, and `scikit-learn` [6] for the implementation of the algorithm.

3.5 Artificial Neural Network

An Artificial Neural Network (ANN) was used to predict the transverse momentum of the neutrino from the charged Higgs decay from the four-momenta of the detected decay products.

The ANN implemented here uses `Python` with `SciPy` [7], `Numpy` [4], and `Pandas` [5] for general data processing, and `Keras` [8] for the actual neural network. The number of input nodes is restricted to the 19 selected predictor variables and the output to the target dimension of 1. Since the network is used for regression, the activation functions between the last hidden layer and the output must be linear.

The data is scaled to avoid the tails of the ANN activation functions and improve the learning rate. The method used here is to scale it such that the minimum is -1 and the maximum is 1 for each training set variable. The test set data is scaled using the scaling parameters computed from the training set. Another scaling method is to scale the mean to 0 and the standard deviation to 1, but this was found to slow down the training and give worst results.

Training is performed using stochastic gradient descent with the Nesterov method [9] and a mean-squared-error loss function. The learning rate was set to 0.1, the learning rate decay to 1×10^{-6}, and the momentum to 0.9.

A large number of configurations of the hidden layers were tested by varying the number of layers, the number of perceptrons in each layer, and the activation functions (sigmoid, tanh, and softmax). Sigmoid activation functions produced the best result for a fixed training period. A network with two hidden layers (i.e. 4 layers counting the 19 node input layer and 1 node output layer) with 25 and 10 perceptrons was found to be sufficiently complex. Using deeper networks, e.g. one with 4 hidden layer with 20, 35, 25, and 15 perceptrons, did not improve the result.

3.6 Support Vector Regression

Support vector machines can be used for regression, and the method is then called support vector regression (SVR). It employs two slack variables, $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$ and the corresponding error function to minimize is

$$C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2}||w||^2$$

where C is the cost variable. The implementation used is the `SVR` class from `scikit-learn`, and it is an ε-SVR method. This implementation has four built-in kernels: RBF, sigmoid, linear, and polynomial. The selected predictors in the dataset are scaled to have a mean of 0 and a variance of 1 using the `StandardScaler` class from `scikit-learn`.

With the presented dataset, the selected SVR algorithm is not converging within a few minutes runnning time. The maximum number of iterations is therefore limited to 1000. A greater number of iterations is not found to yield significantly better results. The kernel that performs best is the RBF kernel. The value of the cost function is by default set to 1.0, but this setting yields a very poor resolution. Different values of the cost function are tested, and a value of $C = 100.0$ is found to be (near) optimal. For the soft margin the default value of $\varepsilon = 0.1$ is good.

Regardless of the choice of C and ε, the performance of the SVR is found to be no better than that of linear ridge regression. The resolution is not improved, and the shape of the true and reconstructed transverse component of the missing energy is not recovered by the prediction.

3.7 Trees and forests

Decision trees and ensemble methods, and in particular boosted decision trees (BDTs), have been a popular choice of method in particle physics. The method of decision trees are based on using multiple discriminating variables, and multiple choices in a binary tree, to find the prediction that best describes the target variable. When using ensemble methods, such as extra trees, gradient boosting, or random forests multiple such trees are created, each with its own weight describing its "importance". The tree depth plays an important role when considering the trade-off between over-training and bias, and with the ensemble methods the number of trees is also an important parameter.

The performances of the classes `DecisionTreeRegressor`, `ExtraTreesRegressor`, `GradientBoostingRegressor`, and `RandomForestRegressor` in `scikit-learn` are examined. Different settings for the tree depth and (for the ensemble methods) number of trees are used. Near optimal values for the depth and the number of trees is found to be 15 and 100 for all methods and all ensemble methods respectively. The decision tree regression shows some improvement in resolution, but the shape of the transverse component of the missing energy is not recovered by the prediction. Neither in the ensemble methods is the shape of the missing energy recovered, but the resolution is improved. All three ensemble method show similar performance.

3.8 Gaussian process regression

Formally, a Gaussian process is a stochastic process defined as a probability distribution over functions $y(\mathbf{x})$, such that a set of function values $y_1, \ldots, y_N$ evaluated at inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are jointly Gaussian distributed.

Given a training data set of targets $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ corresponding to inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$, regression can be performed on the new target t_{N+1} given new input $\mathbf{x}_{N+1}$ by calculating the conditional probability $p(t_{N+1}|\mathbf{t})$. This will be a Gaussian distribution with mean and covariance given by

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \tag{1}$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k} \tag{2}$$

where the elements of $\mathbf{C}$ are $C_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}$, the elements of $\mathbf{k}$ are $k_n = k(\mathbf{x}_n, \mathbf{x}_{N+1})$, and $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$. Here $k(\mathbf{x}, \mathbf{x}')$ is the kernel function, and β is the precision of the noise in the measurements, assuming that the noise ϵ_n for each measurement is isotropically distributed as $\mathcal{N}(\mathbf{0}|\beta^{-1}\mathbb{1})$.

The squared exponential was choosen as the kernel method, defined by

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\theta||\mathbf{x} - \mathbf{x}'||^2) \tag{3}$$

where the parameter θ is estimated using maximum likelihood, and is allowed to float between 10^{-4} and 10^{-1}. The precision of the noise was taken as $\beta = 100$.

Training and testing of the Gaussian processes was done using 3-fold cross validation on 8000 measurements. Due to the non-sparseness of Gaussian processes and memory constraints, more data than this was not feasible to use in the training.

4 Results

The final metrics to measure the performance of the algorithms are the resolution (as described in Section 3.3 above), the ratio of the predicted value with respect to its true value, and the transverse mass of the τ and neutrino system. The transverse mass M_{T} is defined as:

$$M_{\mathrm{T}} = \sqrt{2 p_{\mathrm{T}}{}^{\tau} p_{\mathrm{T}}{}^{\nu} (1 - \cos \Delta\phi_{\tau,\mathrm{miss}})}, \tag{4}$$

where $\Delta\phi_{\tau,\mathrm{miss}}$ is the azimuthal angle between the τ and the direction of the missing transverse momentum.

5 Discussion

This project focused on predicting the transverse momentum p_{T} of the neutrino from the charged Higgs decay by using the reconstructed four-momenta of the detected particles and the missing energy as inputs and training on the true p_{T} given by the Monte-Carlo simulation.

Out of all the Machine Learning (ML) methods tried, ANN and the bayesian ridge regression showed the best performance. The ANN shows the smallest bias in resolution with respect to the Bayes ridge regression but observes a significantly worse variance. All methods show a narrower transverse mass distribution, which is beneficial to discriminate signal over backgrounds.

The mass of the H^+ is the ultimate discriminant variable, but is not built due to the lack of longitudinal information in the detector, one option would be is to target the H^+ mass directly and complement the missing information. This would require Monte-Carlo for a range of possible H^+ masses (3 mass-points were used in this study) and well for the background events.

As stated before, in this project it is assumed that the contribution from the neutrino from the τ decay to the total missing energy is negligible, and only focus on the neutrino from the charged Higgs decay as the unobserved variable. This approach is not correct for higher masses of the H^+. A more complete study would be to target the complete three-vector of the missing energy or the (combined or separate) four-vectors of the two neutrinos. These options were briefly looked at but were considered to be out of the scope of this project. They will be pursued in more detail in future studies.

References

[1] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014.

[2] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Pe-

ter Z. Skands. An introduction to PYTHIA 8.2. *Computer Physics Communications*, 191:159 – 177, 2015.

[3] J. Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. Delphes 3: a modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics*, 2014(2):1–26, 2014.

[4] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[5] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2016-04-19].

[8] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[9] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady*, 27(2):372–376, February 1983.