Analele Universităţii de Vest din Timişoara
Vol. L, 2007
Seria Fizică

B-TAGGING ALGHORITMS FOR THE ILC

Madalina Stanescu-Bellu, Aura Rosca

West University of Timisoara, Faculty of Physics, Bd. V. Parvan 4, 300223 Timisoara

Abstract
In this paper we describe several algorithms to tag the bottom quarks which are used at the future International Linear Collider (ILC). Identification of the b-quarks is an essential task since many important channels in collider experiments at LHC or ILC energies contain bottom quarks. We also combine different algorithms using the Neural Network of the MATLAB tool in order to improve the efficiency of the individual algorithms.

1. Introduction

In reference [1] we explained that heavy-flavour tagging means the identification of jets originating from quarks with high mass such as bottom quarks (b) or charm quarks (c). In that paper we studied particularly the b-tagging because, among others, b-jets are signatures for important channels in HEP experiments like $H \to b\bar{b}, t \to bW$, or for SUSY particles.

Several algorithms can be used to tag the b-jets, based on their properties [1,2], such as the Impact Parameter Joint Probability Tag, the Tear Down Secondary Vertex Finder and the topological vertex finder called ZVTOP. In this paper we will continue our work from [1] and discuss the third algorithm and how to improve it using the MATLAB Neural Nets package.

2. Method and samples

2.1. The topological vertex finder ZVTOP from SIMDET as b-tagging algorithm

SIMDET [3] is a fast simulator of the International Linear Collider detector response for an e^+e^- process. It takes signal hits from generated events (by default Pythia events), free format read cards (FFRC) are used for steering the program and it reconstructs the tracks, providing, among others, arrays with vertex positions. SIMDET provides two types of topological vertex reconstruction algorithms: for the CCD (Charge Couple Device) vertex detector, respectively for the APS (Active Pixel Sensor) vertex detector. The package ZVTOP can be used if we include the external package SINT [4]. ZVTOP provides vertex finding, vertex resolving and vertex fitting, the output representing one or more reconstructed vertices. The scope of the ZVTOP algorithm is to separate b from c jets by cutting on the corrected

secondary vertex mass to be higher than a certain value. ZVTOP includes also a Neural Network approach which we currently try to enhance with Neural Nets from MATLAB.

2.2. Using MATLAB Neural Nets to improve SINT ZVTOP b-tagging performance

2.2.1. Introduction to MATLAB Neural Nets

A Neural Net is composed of one or more interconnected layers of Neurons. It accepts an arbitrary number of inputs and provides one single output by taking into account all cross effects between the inputs. For each Neuron with N inputs we have the output being a function of the so called activation value and the activation value is given

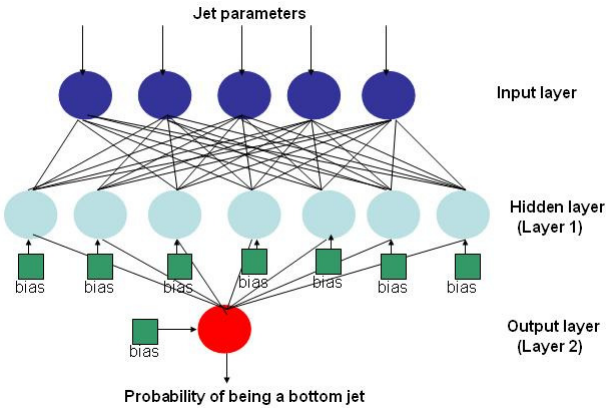
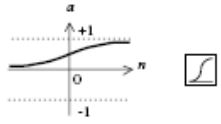


Figure 1 General architecture of a Neural Net

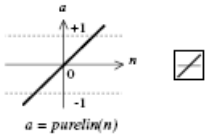
by the formula $\sum_{n=1}^{N} i_n \times w_n + b \times w_b$, where i_n are the inputs, w_n is the weight for each input, b is the associated bias to each input (-1 by default) and f is a threshold function. Let us consider a to be the activation value**.** We can use the following transfer functions [5]:



Log-Sigmoid Transfer Function

$$f(a) = \frac{1}{1+e^{-a}}$$

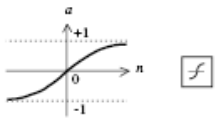
Figure 2. Log-Sigmoid Function



Linear Transfer Function

$$f(a) = \begin{cases} +1, a / givenSlope \geq 1 \\ -1, a / givenSlope \leq -1 \\ a / givenSlope, otherwise \end{cases}$$

Figure 3. Linear Function



Tan-Sigmoid Transfer Function

$$f(a) = tanh(s \times a),\ s = scale$$

Figure 4. Tan-Sigmoid Function

Training the Neural Nets

Calculating the weights is the important part, known as training the neural net. It differentiates a well performing network from a bad one. The principle is to provide inputs of which we know the answer to and the weights are being changed until the output is the desired one, such that the network is formed. Then we simulate the network response with the

current data. For good training we need a large amount of data for a 1 loop algorithm or many epochs (number of iterations) for an algorithm that loops over itself.

Batch Gradient Descent: TRAIN and TRAINGD command

We add several inputs one at a time and determine the gradient of the error between the generated output and the desired output. The weights and biases are updated after all the data was provided, until it reaches the desired goal, with the relation $w_{k+1} = w_k - \alpha_k g_k$, where α_k is the learning rate and g_k is the gradient. The training is stopped when the number of epochs, or the goal (minimum error), the minimum gradient or the maximum time allowed is reached. Extra attention is needed when choosing the learning rate, because if it is too large, the algorithm becomes unstable, and if it is too small, algorithm takes a longer time.

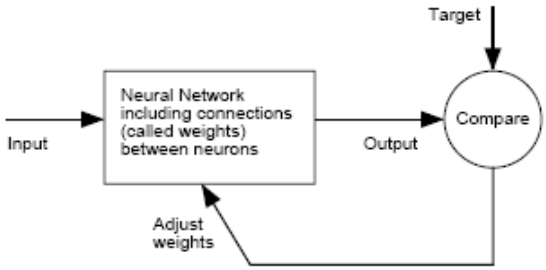


Figure 5. Back Propagation Algorithm [4]

2.2.2. Applying MATLAB Neural Nets to SIMDET

We add in SIMDET the extra package SINT in order to use the ZVTOP algorithm. We switch off the already installed Neural Nets (NN) from ZVTOP, with the FF Read Cards, keeping the results generated by ZVTOP that are not from the NN. We change the output of ZVTOP from a histogram text one, calculate the flavor tag variables as indicated below, and input them to the Neural Nets from MATLAB.

Tabel 1. Flavor-tag variables transformed to be in the range [-1,1], from reference [2].

Flavor tag variable	***Expression***	***Description***
Impact parameter joint probabilities	$P_J(r\phi)$ and $P_J(rz)$	*Always used*
Track impact parameter significances	$\tanh(d_0/\sigma_{d_0}/100)$	*Used when there are no secondary vertices*
Vertex decay length significance	$\tanh((L/\sigma_L)/300)$	*L divided by it's measurement error*
Vertex decay length	$\tanh(L)$	*Distance from primary vertex to 2nd or 3rd vertex*
Vertex mass	$\tanh(m_v/5\,\mathrm{GeV})$	*P_t corrected vertex invariant mass; MOST POWERFUL*
Vertex momentum	$\tanh(p_v/45.6\,\mathrm{GeV})$	*Always used*
Secondary vertex multiplicity	$\tanh(n_s/10)$	*Total tracks in 2nd and 3rd vertices*
Secondary vertex probability	P_s	*Probability that all tracks assigned to secondary vertices come from one common vertex*

Regarding the current Matlab Neural Network architecture implemented, we use three Neural Nets for b-tagging: one for 1-vertex sample (B-Hadron at the primary vertex), one for two-vertices sample (B-Hadron at the secondary vertex) and one for the three or more vertices sample (B-Hadron at the 3rd or further vertex).

Each of the nets has 8 inputs (8 chosen flavor tag variables from table 1), an intermediate Layer1 with 14 neurons and the output Layer2 with 1 neuron, meaning how much probable the jet is to be generated by a bottom quark.

Each Neuron has the Log-Sigmoid Transfer Function shown in figure 2, and the network uses the Batch Gradient Descend training algorithm, *a* type of Back Propagation algorithm, looping over itself.

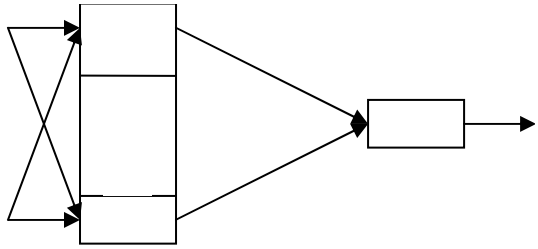


Figure 6. Structure of Layers and Neurons used in the simulation

2.2.3. Summary of the algorithm

We present in the following the complete algorithm flow:

1) Generate data with Pythia;
2) Read MonteCarlo true jet flavor and Input Data;
3) Simdet determines calorimeter and energy flow information;
4) ZVTOP finds vertices, interaction points, estimates flavours and applies kinematical cuts;
5) SINT calculates inputs flavor tag for Neural Network;
6) Train the MATLAB NNs as long as they are not sufficiently trained for b flavour tag;
7) Use pre-trained MATLAB Neural Networks to obtain the b flavour tag;
8) Calculate b-tagging purity and efficiency based on the MATLAB NN output.
   Production of summary histograms

Efficiency represents the number of detected b flavours divided by the total number of the Monte Carlo b flavours produced originally and the purity is the number of real b flavours out of the detected ones divided by the total number of detected b flavours.

3. Results and Discussion

We evaluate the performance of each of the 3 networks, comparing them with the ZVTOP algorithm based solely on the invariant mass tag. The performance should be at least as the one in reference [2]: at high purity, NN should have similar performance to ZVTOP, at high efficiency, NN should have 10-15% more performance, as in figure 7.

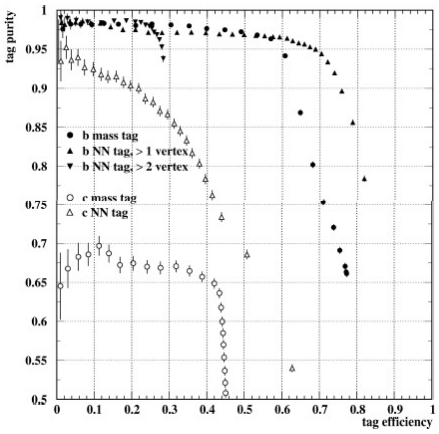


Figura 7. Efficiency vs. purity for b and c tags in $Z^0 \to q\bar{q}$ decays. Comparison between the ZVTOP vertex mass tagging and the ZVTOP NN tagging, from reference [2].

Next, we should run the MATLAB Neural Network and compare the performance with the one in figure 7. To see which architecture fits best, we should then vary the number and types of neurons. For a better training, we should add more events, and for better performance, create new kinematical cuts and eventually check to eliminate badly reconstructed tracks from ZVTOP.

4. Conclusions

The maximum purity is limited due to the relatively small size of the NN training sample (60.000 events) and badly reconstructed tracks by the ZVTOP algorithm. Running the Neural Networks may take a large amount of computing time and memory due to the high number of parallel processes.

Acknowledgements

This work was supported by the CEEX Program of the Romanian Ministry of Education, Research and Youth, under contracts D81 and 118/01.08.2006.

References

1. M. Stanescu-Bellu, A.Rosca, "Heavy Flavor Tagging for Linear Colliders", presented at Students' Conference of Science and Technology, 18 May 2007, Ann. UVT, in press
2. R. Hawkings, "Vertex detector and flavor tagging studies for the TESLA linear collider", LC-PHSM-2000-021-TESLA, February 20, 2000
3. M. Pohl, H.J. Schreiber, "SIMDET – a Parametric Monte Carlo for a TESLA Detector", LC-DET-2002-005, 2002.
4. T.Kuhl, K Harder, "ZVTOP-B Hadron Tagger for SIMDET", http://irfu.cea.fr/ecfadesy-stmalo/Higgs/session1/kuhl.ps/; /afs/desy.de/user/k/kuhl/public/sint/
5. MATLAB Neural Networks Toolbox User's Guide, http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/