Advanced Event Reweighting using multivariate analysis (MVA)

D Martschei, M Feindt, S Honc, J Wagner-Kuhr

Institut für Experimentelle Kernphysik - Karlsruhe Institute of Technology KIT, DE

E-mail: martschei@ekp.uni-karlsruhe.de

Abstract. Multivariate discrimination techniques, such as neural networks, are key ingredients in modern data analysis and play an important role in high energy physics. They are usually trained on simulated Monte Carlo (MC) samples to discriminate so called "signal" from "background" events and are then applied to data to select real events of signal type. We here address procedures that improve this workflow.

(i) Enhance data / MC agreement by reweighting MC samples on a per event basis.
(ii) Training MVAs on real data using the ${}_s$Plot technique.
(iii) Constructing MVAs whose discriminator is independent of a certain control variable, i.e. cuts in this variable will not change the discriminator shape.

1. Introduction

In order to have a well trained MVA which gives a calibrated discriminator, i.e. a discriminator which is linearly dependent on the signal probability of the given event, it is necessary to have a training sample which is as similar to data as possible. This requirement can not fully be met by generated events in most cases. Some of the possible reasons are:

- Cross sections and branching ratios of different processes have yet to be measured
- Hard process simulation is not good enough (e.g. it is only calculated to some finite precision by theorists)
- Detector simulation is not yet optimal (material budget, dead channels, ...)
- Physics "behaves" differently than initially expected in the new energy range.

Usually some of these effects, which appear as differences between data and MC, are tried to be eliminated by introducing scale factors. These scale factors are usually just constants or functions of some measured variable like p_T or η of the leading jet. This can already reduce differences between data and MC a lot, but the agreement is usually not yet perfect in all variable distributions and can even make certain distributions worse than they were without weights. As will be shown in section 2, it is possible to use MVA methods to derive individual per event weights, which will outweight differences in all input variables simultaneously.

In section 3 the ${}_s$Plot method will be introduced, which can be used to train on measured data events only, and not depend on any generated MC samples. The third section 4 will focus on the issue of biasing the distribution of a control variable through discriminator cuts. This happens if the discriminator is correlated to the variable in question, which might be the case if

the control variable is able to discriminate signal and background itself. Also in this case using appropriate weights can solve the issue by outweighting the correlation but still keeping all the discrimination power which is orthogonal to the control variable.

2. Multivariate reweighting to correct for differences between generated Monte Carlo and real data events

In physics experiments there are usually differences between the simulated MC events and measured data. These arise from limited knowledge of the underlying physics processes, detector mismodeling and other issues.

In order to still be able to use the MC ensemble as a whole qualitatively, weights may be applied to make certain groups of events more pronounced and make observable distributions agree better between MC and data.

Instead of using scale factors in bins of certain measured values, we propose to use MVA methods to derive individual weights per event.

2.1. Task description

The reweighting procedure has to emphasize those events in the MC sample which have a high probability to be found in data, based on a certain phase space under inspection. This can be achieved by performing a training in which real data events are used as target/signal events and MC is fed to the MVA as background events. The MVA will then pick up which MC events can be separated from data. Optimally, if data/MC agreement was perfect, the MVA would not be able to learn anything in this training. Everything it learns is due to mismodeling in MC.

In the next step the MVA is applied to MC events and the discriminator output is transformed into an event weight.

2.2. Derivation of transformation for MVA output to event-weight

If the MVA was trained to discriminate data from MC, the MVA will yield the probability p for an event to be data when selecting from the whole sample: $p = \frac{data}{data+MC}$. What we need is the probability w for the MC event to be data when selecting from the MC sample only $w = \frac{data}{MC}$. In the following the derivation for NeuroBayes output NB_{out} is given, which is linearly dependent on p in the range [-1,1].

$$p = \frac{NB_{out}+1}{2} = \frac{data}{data+MC} \quad \Rightarrow \quad w = \frac{data}{MC} = \frac{1+NB_{out}}{1-NB_{out}} \tag{1}$$

2.3. Illustrative example

For illustration, the above steps have been applied on simulated toy samples. Two MC-samples with 9 observables were generated from arbitrary analytical distributions. Some correlations among the observables were introduced in order to have a more realistic example, e.g. var3 and var6 have a correlation coefficient of 0.2. Two of these observables have different distributions in the two samples, while the other 7 observables are distributed equally. These distributions are the black and blue ones in figure 1.

In this example the differences were exaggerated to be visible with the naked eye and to show the power of this method. As a first step towards individual event weights, a NeuroBayes training to discriminate between pseudo data and MC has been performed. The discriminator distribution can be seen in figure 2(a).

Since the two input variables shown in figure 1 have quite different shapes for the two samples, the NeuroBayes output is able to discriminate the two sample types quite well. The weights w for the MC events are derived using formula (1). The weight distribution is shown in figure 2(b)

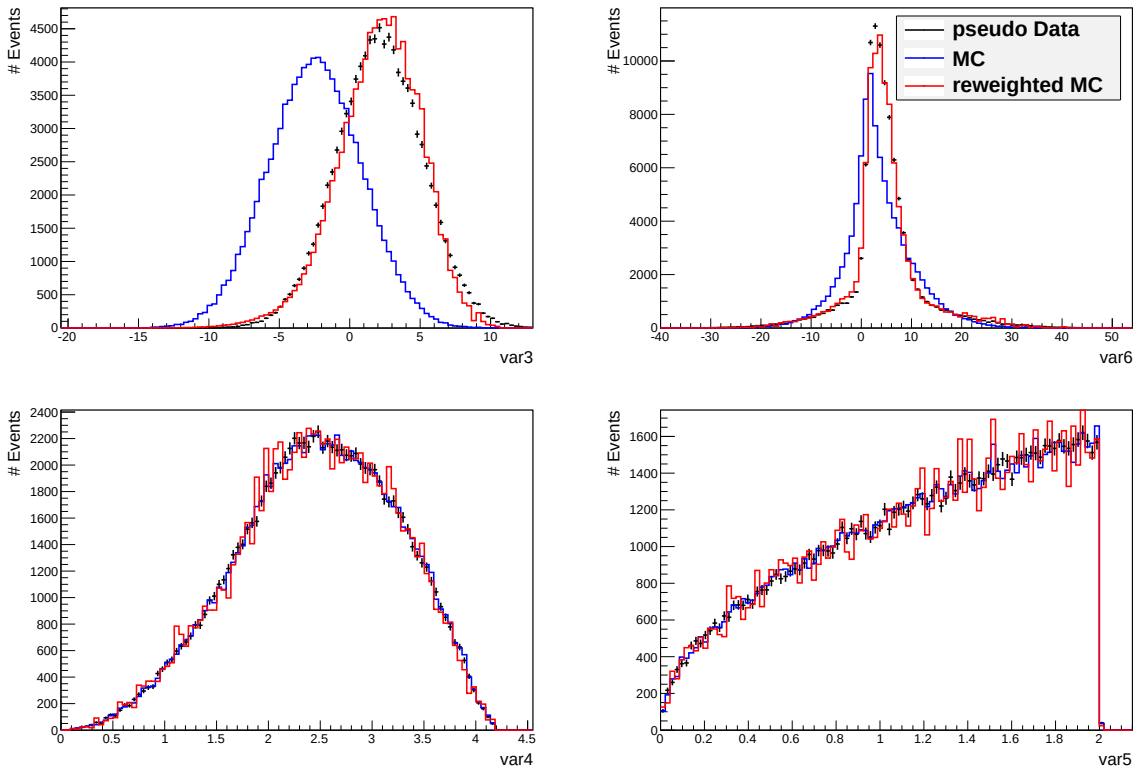
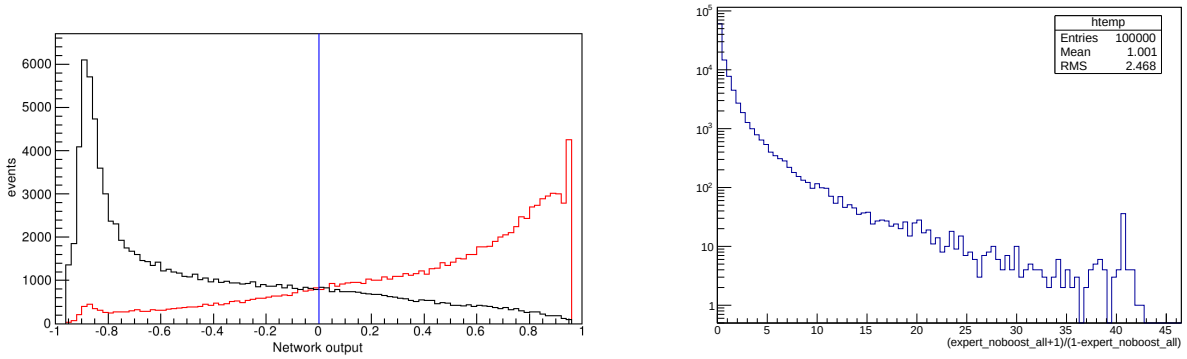


Figure 1. Distributions for 4 input variables. The MC (blue) was reweighted (red) with the weights from NeuroBayes to become comparable to the pseudo data.



(a) Distribution of NeuroBayes discriminator for pseudo data (red) and MC events (black)

(b) Histogram of event weights for MC events derived from NeuroBayesExpert output.

Figure 2. Histograms for the MC to pseudo data reweighting.

As can be seen, most events will get weights smaller than 1, while some will get huge weights of up to 45. This is mainly due to the small overlap for the distributions of var3 and the big height difference in some regions. Applying the weights to the MC events leads to the distributions shown in figure 1. The method does not work well if there are no MC events at all in regions where real data events exist since then the weights diverge. Obviously the reweighted distribution agrees quite well in those variables which were differing without weights, while those

variables which already matched are still agreeing. This would not necessarily be the case if simple histogram based weights were applied, since they don't take correlations into account.

3. Training on data without MC: side band subtractions and the $_s$Plot technique

3.1. Side band subtraction

Another way of getting around the issue with the disagreement between data and MC would be to directly train on data. However this is usually not possible because one cannot find absolutely pure samples of signal events. Still in some cases, one can at least define regions in phase space which are signal enriched and where a signal peak is already visible over some background. This peak can be used to estimate the yield of signal events and perform a side-band subtracted training. An overview of the different regions needed to perform such a side-band subtraction is shown in figure 3.

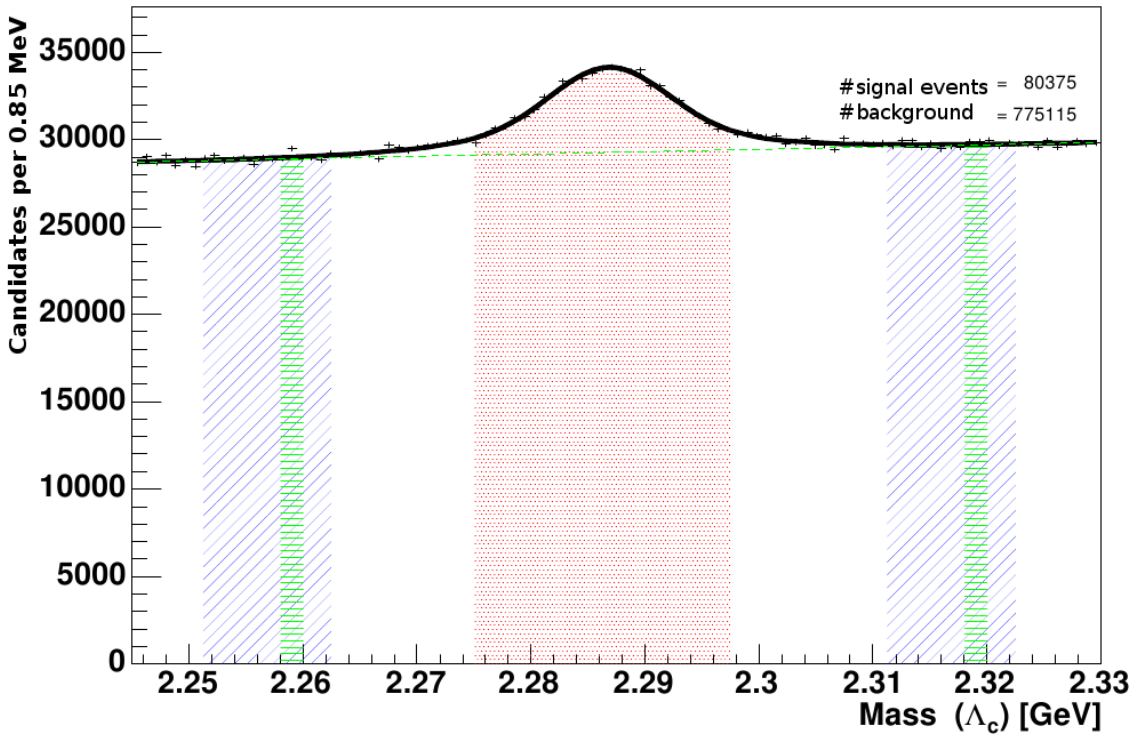


Figure 3. Mass distribution of Λ_c candidates. Red is the signal region, while blue and green are the side-bands. [3]

Three regions are defined and the events are used during the training in the following way:

region	amount of events	target type	weight
red	N_{sig} + N_{back}	signal	1
green	N_{sig}	background	1
blue	N_{back}	signal	-1

N_{sig} is the estimated yield of signal events in the red region (in the peak above the background), while N_{back} is the estimated amount of background events in this region. With the weight[1] and target definitions given above, the MVA will get a training sample which (after subtraction) consists of an equal amount of signal and background events. One has to make sure

[1] A short introduction on how weights are used during a MVA training can be found in Appendix A.

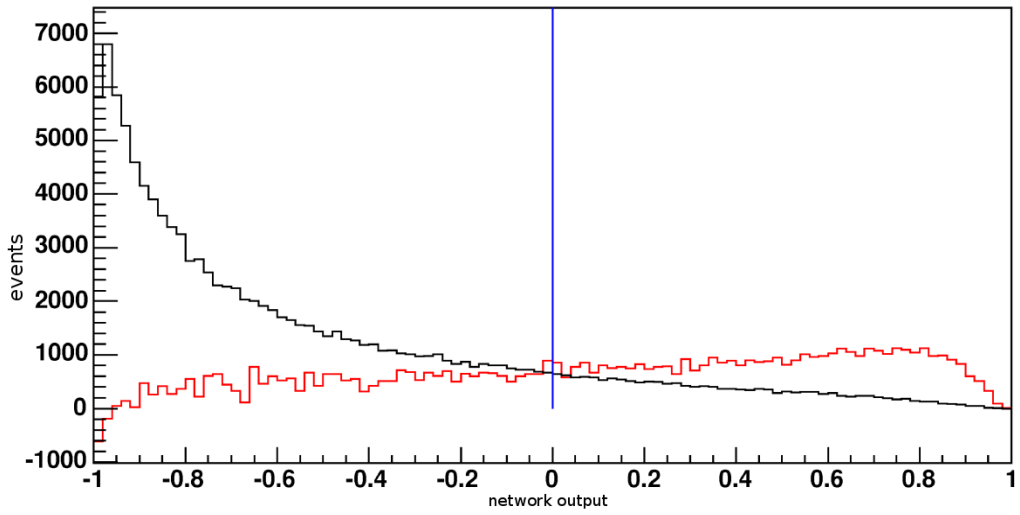


Figure 4. NeuroBayes discriminator distribution for the sideband training. [3]

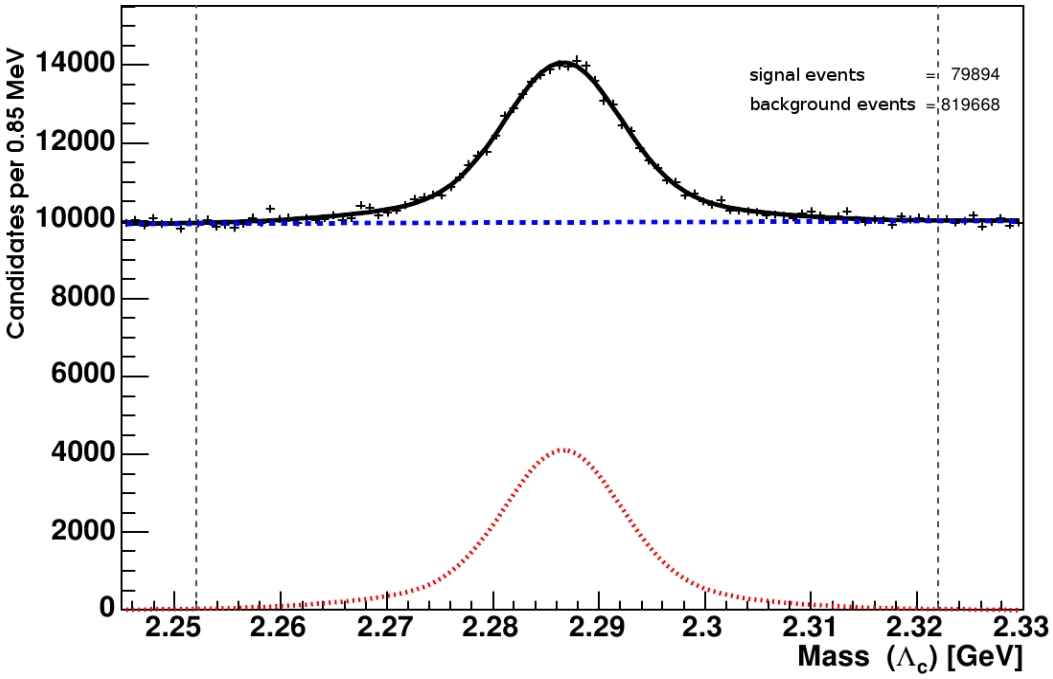


Figure 5. Mass distribution for Λ_c candidates after a discriminator cut of 0.5. [3]

that the MVA is not able to learn the dependency of the variable (mass in this case) one uses to estimate the signal and background yields, e.g. by excluding input variables that can distinguish between left and right side-band or the method in section 4. In this example mainly variables which parametrize the identification quality of the Λ_c decay particles kaons and protons and their kinematics are used, but not their full momentum or energy since these are too closely related to the Λ_c mass.

3.2. The $_s$Plot formalism: advanced side-band subtraction

An advanced version of the side-band subtraction is the $_s$Plot formalism. It exploits the complete probability densities of signal and background. Also a pure background region is not necessary, it is enough that the PDFs differ measurably.

A full derivation of the $_s$Plot weights can be found at [4]. They result is:

$$\begin{pmatrix} w_s(x) \\ w_{bg}(x) \end{pmatrix} = \frac{1}{f(x)} \cdot \mathbf{V} \cdot \begin{pmatrix} f_s(x) \\ f_{bg}(x) \end{pmatrix}$$

with:

$$f(x) = N_s \cdot f_s(x) + N_{bg} \cdot f_{bg}(x)$$

Here the following variable definitions apply:

- w_s the $_s$Plot signal weight, w_{bg} the background weight, with $w_s + w_{bg} = 1$. w_s and w_{bg} may be negative or larger than 1.
- $f_{s,bg}$ are the probability density functions of signal and background.
- $\mathbf{V}$ covariance matrix of f_s and f_{bg}.

What the $_s$Plot formalism will do, is the following. One has to deduce the probability density function for the two event types signal and background, e.g. using function fits to a mass peak. Using the given formula above, these pdfs can be transformed into event weights. By applying the signal/background $_s$Plot weights to data, we will get the pdf of any variable independent of the one (mass) we used to deduce the pdfs in the first place. In order to actually use these weights for a MVA training one has to use each event twice[2], once as signal with weight w_s and once as background with weight w_{bg}. During the training the MVA will then learn that there are certain events in the trainings sample which have a high signal weight and a lower background weight, and reside in a different phase space than the rest of the sample and will learn their properties to distinguish them.

3.3. Example for the application of the $_s$Plot formalism

An example is taken from the PhD thesis of Claudia Marino [2]. The $_s$Plot formalism was used there to train NeuroBayes on the Υ(1s) resonance in CDF data. The goal was to reduce the background under the Υ 1s,2s,3s resonances for further studies. The result can be seen in figure 6. The background could be reduced drastically while barely loosing signal events. Again this was done by training on real, measured data events and not using any kind of generated events. Another successful application of this method was the excited charm baryon analysis by the CDF collaboration [5].

4. Orthogonal discriminator: preventing distribution bias through event selection

In some analyses the final result (e.g. signal yield or certain properties) should be deduced by fitting a template to a certain control variable distribution. We will refer to this variable as varX in the following. This variable is chosen such that it will show a clear shape difference between the distribution of signal and background events. This is essential for the template fit to converge. Usually the amount of background has to be reduced drastically to have a decent signal/background ratio, which is done by cutting on several other variables and/or with a cut on a MVA discriminator. Especially the latter may introduce a bias in the control variable varX if the MVA learned the dependence of the signal probability on varX. This might even happen

[2] For NeuroBayes trainings this is not necessary any more, since it provides a special $_s$Plot training mode. It will automatically use each event as signal and background event if given the $_s$**Plot weight as target** and will account for the *real* given statistics and the correlation of the two entries of a given event.

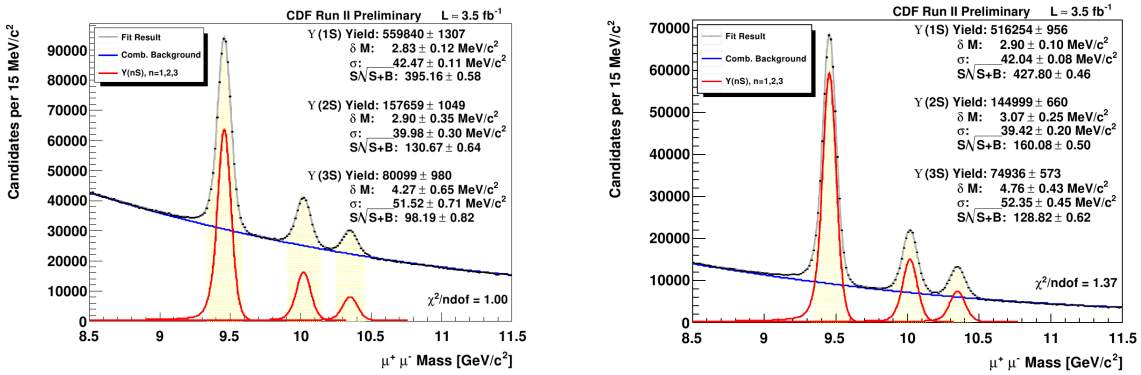


Figure 6. Υ resonances before and after cutting on a $_s$Plot trained NeuroBayes network.
Extracted from Phd thesis Claudia Marino figure 5.8

if varX is not one of the input variables because the other variables may be correlated to varX. In any case what might happen is illustrated in figure 7.

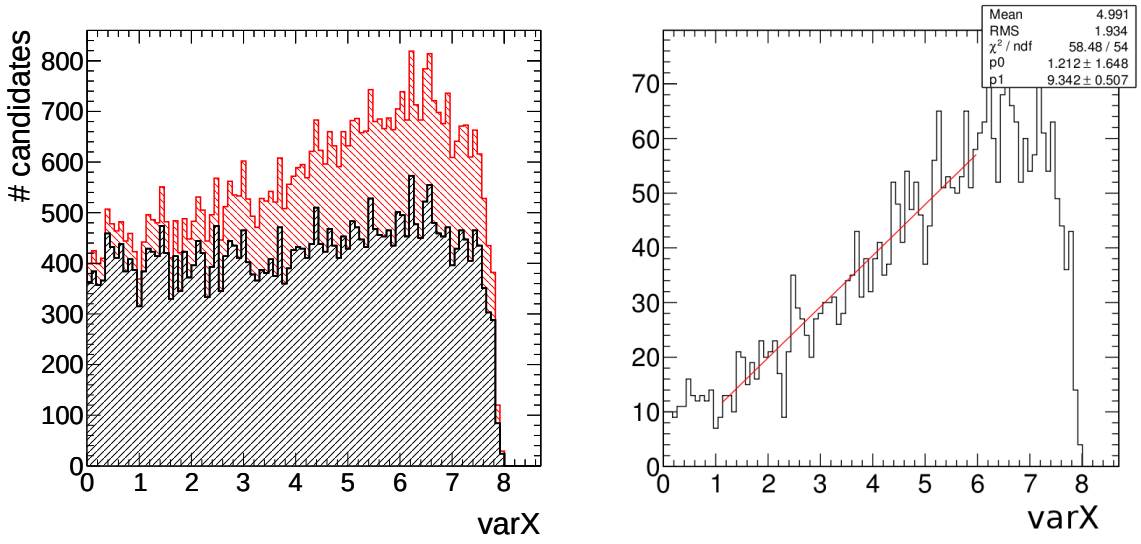


Figure 7. The left, stacked plot shows the original varX distribution for signal(red) and background(black). The background distribution is relatively flat and has only a small slope, while the signal events are more prominent at higher values of varX. Due to the rising of the signal distribution in varX, a cut on a discriminator which uses varX as one of its inputs, will result in a background distribution which is shown on the right.

Obviously the background distribution gets tilted by the discriminator cut, which is an issue in some analysis and we will therefore present a way to prevent this.

4.1. Task description

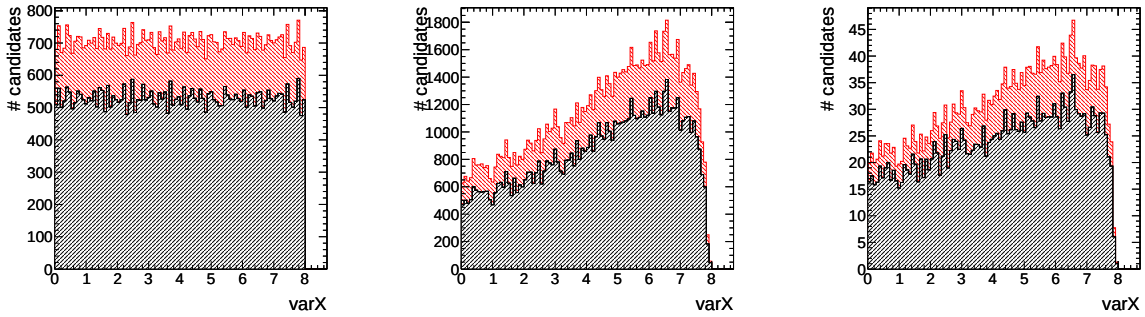
The task is to find a way to train the MVA such that its discriminator value is independent of varX and that cuts will equally remove background events from all bins in varX.

The easiest way would be to eliminate those variables which carry information about varX from the set of input variables. On the one hand this would remove the discriminator's

dependence on varX but it will probably also remove some discrimination power from the MVA which is orthogonal to varX and should be saved.

Since the problem mainly arises from the different dependence of the signal and background on varX, another possible solution is to reweight the training sample such that this dependence vanishes. For this, the ratio between the desired distribution and the current height of the bin in each histogram is used as weight for events in this bin. Any distribution would be possible of which two have been studied here.

- Signal and background are both reweighted to be flat in varX. This is shown in figure 8(a).
- Both distributions are reweighted to the shape of the inclusive distribution, as shown in figure 8(b). This has the advantage over the flat shape that especially in the low varX region for the signal distribution the weights are not as large. Large weights should be avoided during MVA trainings as they will make the internal error calculation more difficult.



(a) Signal and background reweighted to be flat in varX

(b) Signal and background reweighted such that each distribution has the same shape as the original inclusive distribution.

(c) Event reweighted with individual weight-network values

Figure 8. Reweighted shapes of varX with the three different weight types for signal(red) and background(black).

The weight calculation based on the varX histograms for signal and background has several shortcomings:

- The differences between the signal and background are only taken into account in bins and not continuously or individually per event.
- Only reweighting based on varX does not take complex correlations between varX and the other input variables into account. Thus some of the unwanted discriminator dependence might still be present.

The third approach we have studied is more elaborate. The basic idea is to train a MVA to discriminate between events which have high or low values of varX. This network will carry all information about varX which is contained in the input variables. The output of this MVA can then be translated into a weight for the actual signal/background training, which will be insensitive to varX. In the following we will give a more detailed explanation on how this MVA based weights are derived.

4.2. MVA based weight derivation

In order to deduce individual per event weights a more complex approach was developed. Here we use a chain of NeuroBayesExperts to calculate the weights for the final signal against background

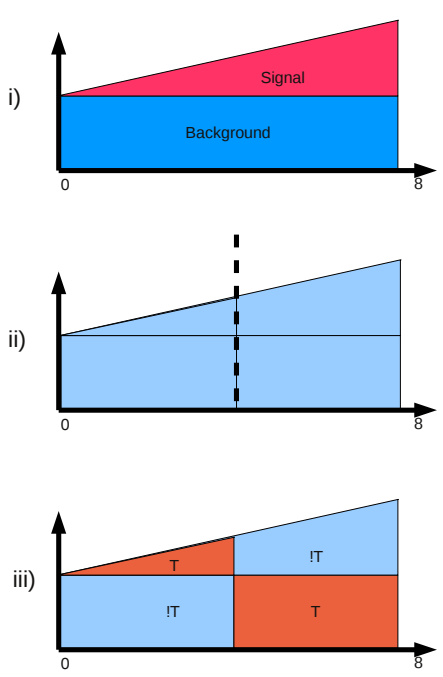
training. In the following we have to distinguish between those NeuroBayes trainings which are needed for the weight derivation (called weight-network) and the one which does the actual signal against background discrimination (called discriminator-network).

The first step in developing the weight-network is based on the idea, of letting NeuroBayes learn the information whether varX is high or low in an event based on the input variables which will be used in the discriminator-network. So the target definition for the weight-network training is "Is varX higher than a certain threshold?". The steps to get to the definition of what is target (T) and non-target (!T) during the weight-network training is illustrated in figure 9(a) and are as follows:

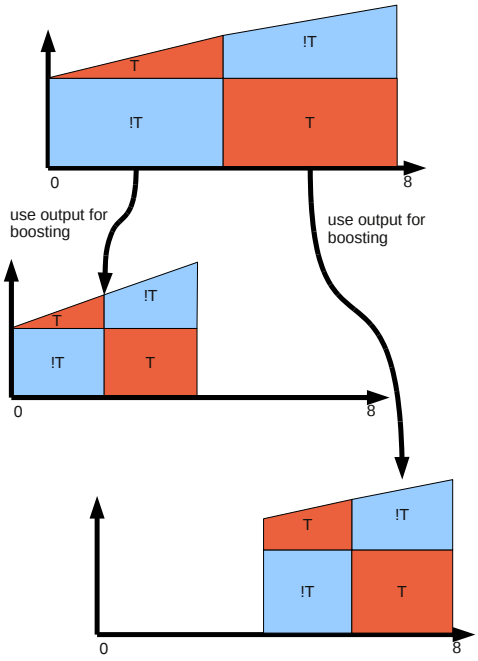
(i) Draft of signal and background in a stacked plot.

(ii) We do not really care of what is signal or background for the time being, but we divide the whole dataset in the middle of varX (4 in this case).

(iii) Only defining the high values as target and the low values as non-target is not quite sufficient, because then the weight-network will implicitly already learn what signal and background look like. Therefore the target and non-target definition is opposite between signal and background for the weight-network training in one half of the dataset.



(a) Definition of target(T) and non-target (!T) region for weight-network training



(b) Illustration of first boost step. Divide the sample into two subsamples and run a boost training on each sample separately.

Figure 9. Structure illustration of the weight-network boost chain. The output from the first network is used as boost weight in the successive layers weight-networks.

One weight-network will learn the differences between events with very high and very low values of varX but it will not be very sensitive to differences in events which are closer in varX. Thus in order to refine the weight quality the dataset is split in a Haar-wavelet expansion. In

each step, the previous training sample is split in the middle of varX into two subsamples and on both a boosted weight-network is trained with the adapted target definition. An illustration of the first step of this procedure is shown in figure 9(b). It is repeated several times (3 times in our example) until the following boost networks are not able to learn new information.

A "boosted network" in this case refers to a special training mode, in which the output from one network is used as weight for the trainings in the next layer of networks. The weight-network boost weight is derived by transforming the NeuroBayes output into the probability $p = \frac{\text{NBout}+1}{2}$. This is already the weight if the event was a non-target (!T) event in the previous training, otherwise the weight is $1 - p$. In general this choice of weights leads to a enhancement of events which have been misclassified by the previous network, since these will get higher weights.

For the example shown here, a tree with 4 layers was used, which consists of 1+2+4+8=15 NeuroBayes networks which were trained successively on different subsamples as boosted networks (except the first one of course). In order to calculate the discriminator-training weight for each event one has to multiply the returned probability of the four corresponding weight-networks depending on its value of varX.

The weighted distribution of varX is shown in figure 8(c). It looks similar to the ones reweighted to the inclusive distribution (sum of signal and background), but it yields better performance as we will see next.

4.3. Performance comparison

In summary six different NeuroBayes trainings with different weights were performed. The full list of all trainings and whether they used varX as input or not can be seen in table 1.

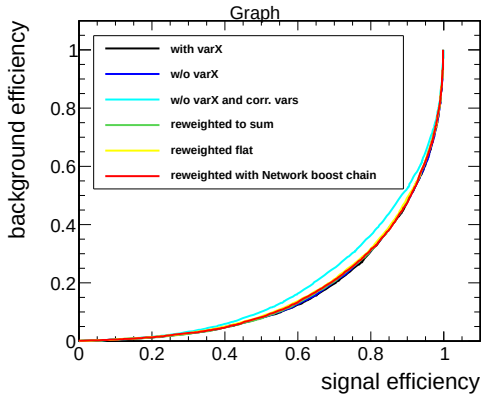
training title	variables left out of training inputs	weights
with varX	none	none
w/o varX	varX	none
w/o varX and corr. vars	varX and variables highly correlated	none
reweighted to sum	varX	histogram weights for flat shapes
reweighted flat	varX	histogram weights for inclusive shape
reweighted with NB boost chain	varX	NeuroBayes boost chain derive weights

Table 1. List of trainings, showing which variables were left out and which weights are applied

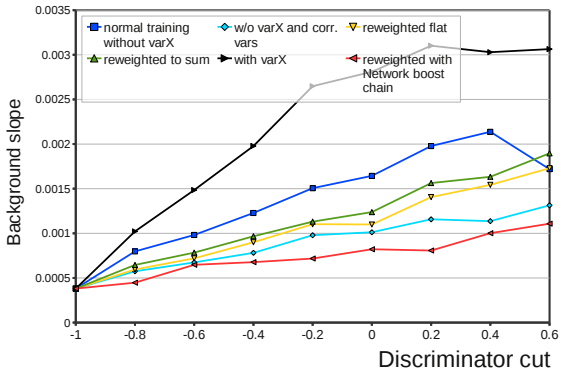
There are 31 variables available in the trainings sample, including varX. In most trainings all but varX are used. For one training also varX is included and in another varX and those variables which are highly correlated to it are excluded.

The fitness of MVAs is usually compared by means of discrimination power. In order to deduce whether one MVA is better than the other, so called ROC (receiver operating characteristic) curves are compared. In figure 10(a) the background efficiency versus the signal efficiency is shown for the different NeuroBayesExperts (i.e. the part of NeuroBayes[1] which is used after the training for the actual classification).

As can be seen the NeuroBayesExpert which was trained without varX and all correlated variables yields the worst performance, as expected. The network which yields slightly better performance than the others is the one with all variables included and no weights applied but all the other four are close by.



(a) Performance curves for the different NeuroBayesExperts



(b) Background slopes after discriminator cuts for all studied NeuroBayesExperts

Figure 10. Benchmark results for different NeuroBayes trainings concerning discrimination power and background flatness after discriminator cuts.

Since a good separation power is not the only figure of merit in this task, but we also want the background distribution to remain flat after discriminator cuts, we defined a second benchmark to evaluate which training type is suited best for our needs. The goal there is to measure the insensitivity of the background distribution of varX to discriminator cuts. Therefore straight lines were fitted to the central range of the distribution after several discriminator cuts and the slopes were compared. Such an example of a fitted line is shown in figure 7 in the right plot. The slope values for all the background distributions after several cuts on each of the NeuroBayesExperts are shown in figure 10(b).

Cuts were made in steps of 0.2 in the discriminator range [-1,1]. The results of a cut at 0.8 are not shown, since the fits did not yield reasonable results due to the small statistics in the remaining distribution.

The black line illustrates the behaviour of the ordinarily trained NeuroBayes network including varX in the inputs. With harder cuts the slope of the distribution of the remaining events gets steeper, which is the unwanted feature we are trying to avoid. The second worst slope behaviour is returned by the network which was trained without varX and without any weights applied. This already gives quite an improvement, but due to the variables correlated to varX the slopes still rise. Leaving out also the correlated variables leads to the light blue points, which is the second best in this comparison, but yielded the worst discrimination performance. In this benchmark the slope remains the flattest when cutting on the network which was trained with NeuroBayes boost chain weights (since it is even better than the one without varX and correlated variables, there still seems to be some correlation left). This means the effort of using 15 NeuroBayesExperts for the weight calculation is worthwhile, since it yields a good discriminator while keeping the background distribution of the remaining events flat.

5. Summary and Conclusion

We've presented three use cases in which NeuroBayes was either used to calculate individual weights for each event and/or weights were used to train NeuroBayes in a special way. These are not just theoretical concepts, but we have shown, that they are applicable in practice.

(i) We have used NeuroBayes to calculate individual MC event weights which were able to correct significant differences in the distribution of two variables between pseudo data and MC, while keeping the already matching ones untouched. The latter one is an advantage of MVA deduced weights over most commonly calculated scale factors since they usually are not able to take all correlations fully into account.

(ii) the ${}_s$Plot method was presented which has already been successfully used in real physics analysis to train NeuroBayes on real measured events without the need for any generated MC events. The only prerequisite is that a signal peak is visible and can be modelled in order to deduce a probability density function for signal and background for calculating the ${}_s$Plot weights.

(iii) The third use case of advanced event weights we have shown will train a discriminating network in such a way, that the shape of the distribution of a certain control variable is barely influenced by making selection cuts on the discriminator. This is achieved by training a tree like chain of NeuroBayes networks in order to absorb any information about the control variable into an event weight, which will make the discriminating-network blind to any control variable information.

Appendix A. Training of MVAs and where weights are taken into account

From the mathematical point of view, a well trained discriminating MVA is a non-linear transformation

$$f(\overrightarrow{x} \in \mathbb{R}^n) \to y \in \mathbb{R}$$

such that a loss function

$$\mathrm{err}(y) = \sum_i g(y_i, t_i) \qquad (\ g(y) \text{ could be } \chi^2 \text{ for example})$$

is minimal with respect to the target value t (usually $t = 1$ for signal and $t = 0$ or -1 for background).

Introducing weights leads to

$$\mathrm{err}(y) = \sum_i w_i \cdot g(y_i, t_i)$$

So according to their weights, events are taken into account differently when minimizing the loss function. It can be understood as emphasizing certain events and suppressing others during the training procedure. This can be used to achieve several effects during a training.

References

[1] Feindt M 2004 *A Neural Bayesian Estimator for Conditional Probability Densities* arXiv:physics/0402093v1

[2] Marino C 2009, PhD thesis, IEKP-KA/2009-29

[3] Wick F 2009, Diploma thesis, IEKP-KA/2009-2

[4] Pivk M 2006, *sPlot: A Quick Introduction*, arXiv:physics/0602023v1 [physics.data-an]

[5] Wick F 2011, *Measurements of the properties of Lambda_c(2595), Lambda_c(2625), Sigma_c(2455), and Sigma_c(2520) baryons*, arXiv:1105.5995v2 [hep-ex]