# Application of Neural Networks Optimized by Genetic Algorithms to Higgs Boson Search

František Hakl*, Marek Hlaváček**, and Roman Kalous**

*Institute of Computer Science AS CR, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic

**Mathematics department, Faculty of Nuclear Science and Physical Engineering, Czech Technical University, Prague, Czech Republic

**Abstract.** This paper describe an application of a neural network approach to SM (standard model) and MSSM (minimal supersymetry standard model) Higgs search in the associated production $t\bar{t}H$ with $H \to b\bar{b}$. This decay channel is considered as a discovery channel for Higgs scenarios for Higgs boson masses in the range 80 - 130 GeV. Neural network model with a special type of data flow is used to separate $t\bar{t}jj$ background from $H \to b\bar{b}$ events. Used neural network combine together a classical neural network approach and linear decision tree separation process. Parameters of these neural networks are randomly generated and population of predefined size of those networks is learned to get initial generation for the following genetic algorithm optimization process. A genetic algorithm principles are used to tune parameters of further neural network individuals derived from previous neural networks by GA operations of crossover and mutation. The goal of this GA process is optimization of the final neural network performance.
Our results show that NN approach is applicable to the problem of Higgs boson detection. Neural network filters can be used to emphasize difference of $M_{bb}$ distribution for events accepted by filter (with better $\frac{signal}{background}$ rate) and $M_{bb}$ distribution for original events (with original $\frac{signal}{background}$ rate) under condition that there is no loss of significance. This improvement of the shape of $M_{bb}$ distribution can be used as a criterion of existence of Higgs boson decay in considered discovery channel.

## 1 Introduction

This work is devoted to application of neural network to high energy physic. There is a broad consensus in physics community that newly building Large Hadron Collider (LHC) detector at CERN, Geneve, should be able to produce showers of particles, whose will have capability to confirm presence of Higgs boson. Using theoretical background of high energy physic there is possible to postulate theoretical properties of some distributions of selected values whose

* This work is supported by grant of Ministry of Trade and Industry of the Czech Republic, Project No. RP-4210/69/97 and by grant of Ministry of Education of the Czech Republic, Project No. LN00B096.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 554–563, 2002.

describe properties of Higgs boson decay in the LHC. Using simulated output from LHC based on simulation package PYTHIA, we have available two sets of shower parameters, one set corresponding to case with Higgs boson decay (we denote this as signal) and second one corresponding to case without Higgs boson decay (we denote this as background). So we can see the problem of Higgs boson search as a classical problem of pattern recognition with this exception that the quality of separation is measured as difference between two distribution curves corresponding to separated sets.

Neural nets are broadly used in pattern recognition problems and functions approximation tools. There are many types of artificial neural networks whose differ in architecture, in the type of implemented transfer functions and strategy of learning. Regard to universal approximation property we declined to use a special kind of neural nets, namely neural network with switching units [1], [2], [3] to solve pattern recognition problem postulated above. To reach better performance of these neural networks we tune topology and parameters of such networks via genetic algorithm optimization.

## 2 Description of neural network with switching units

Neural network with switching is a combination of classical neural network architecture and decision tree. This network is actually an oriented acyclic graph (see Fig. 1) which node are structures called as building blocks. This acyclic graph will be referenced as outer graph.

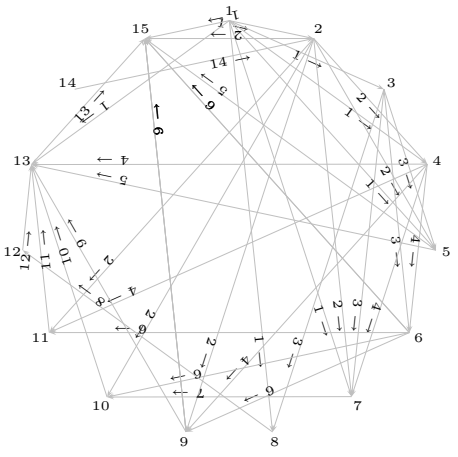
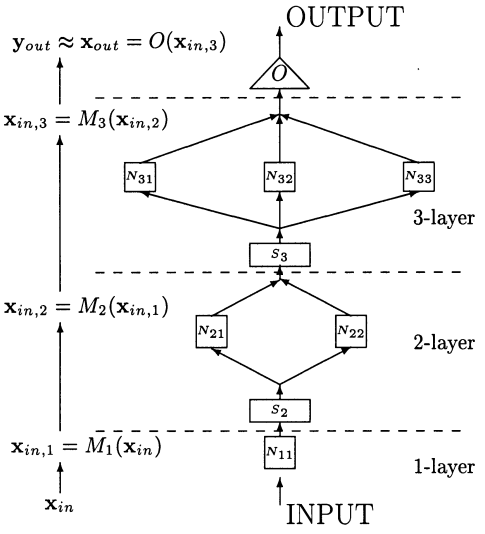


**Fig. 1.** Schema of connection between building blocks.

Each building block is a neural network consisting from two types of nodes. These nodes are connected together in such a way that they formed an acyclic graph again but with the restriction that outputs dimension of building block is the same for all building blocks in outer graph. First type of node, we refer this node as functional units, makes predefined mapping from input space corresponding to this node to output space of this node. Hence such node can be described by tuple of integers, input vector dimension and output vector dimension, and by transfer function. Definition of this transfer function include parameters of this functional unit (weight vectors, threshold etc. in current neural networks terminology).

Functional units map corresponding inputs to outputs by internal transfer function. This transfer function can differ for each functional unit. For example,




Switching net maps input $\mathbf{x}_{in} \in R^d$ into output

$$\mathbf{x}_{out} = M(\mathbf{x}_{in}) \in R^b,$$

where the mapping $M$ is a composition of mappings performed by each layer

$$M = S \odot M_{nl} \odot M_{nl-1} \odot \cdots \odot M_2 \odot M_1,$$

$n_l$ is number of layers, $M_1 = N_{11}$, and

$$\mathbf{x}_{in,k} = M_k \odot \cdots \odot M_1(\mathbf{x}_{in})$$

$$M_k = N_{kS_k(\mathbf{x}_{in,k-1})}, \quad \text{for} \quad k > 1.$$

**Fig. 2.** A topology of simplified building block.

let now we describe transfer function currently implemented. Let $\boldsymbol{x}_i \in \Re^n$, $i \in \{1, \ldots, p\}$ are input patterns into functional unit, corresponding desired outputs let be denoted as $\boldsymbol{y}_i \in \Re^m$, $i \in \{1, \ldots, p\}$ . Desired output of functional unit can be generally different from desired output of the whole network, but for the sake of explanation clarity we assume that desired output of each functional unit is the same as desired output of the whole network. A simple case of transfer function is linear mapping which minimize norm of the vector $\boldsymbol{AW} - \boldsymbol{Y}$, where $\boldsymbol{W} \in \Re^{m \times n}$ is matrix of weight parameters, $\boldsymbol{A} \in \Re^{n \times p}$ is a matrix which rows are vectors $\boldsymbol{x}_i$ and $\boldsymbol{Y} \in \Re^{m \times p}$ is a matrix which rows are formed by vectors $\boldsymbol{y}_i$.

The second type of nodes, switching units, collect all outputs from parents functional units, concatenate them together to form one vector, and search predefined number of clusters in the set of such input vectors. We use the Jancey cluster algorithm which is non-deterministic procedure described in the following schema:

(let $d$ be the number of desired clusters (which is equal to the number of switching unit children), $\boldsymbol{z}$ is concatenation of output vectors of switching unit parents.

1. for randomly chosen sequence $1 \leq j_1 < j_2 < \cdots < j_d \leq p$ set

$$\boldsymbol{c}_q^{new} = \boldsymbol{c}_q^{old} = \boldsymbol{z}_{j_q} \quad \text{and} \quad \bar{S}_q^{new} = \bar{S}_q^{old} = \{\boldsymbol{z}_{j_q}\}, q \in \{1, \ldots, d\}$$

2. let $r_1, \cdots, r_p$ is random permutation of the $1, \cdots, p$,

3. FOR ALL $k = r_1, \cdots, r_p$
   DO
   let $q$ be such index that $\boldsymbol{z}_k \in \bar{S}_q^{old}$,
   $i = \min\left\{v \left| \left\|\boldsymbol{c}_v^{old} - \boldsymbol{z}_k\right\|_E = \min_{q\in\{1,\ldots,h\}} \left\{\left\|\boldsymbol{c}_h^{old} - \boldsymbol{z}_k\right\|_E\right\}\right.\right\}$,
   $c_q^{old} = \boldsymbol{c}_q^{old} - \frac{\boldsymbol{z}_k - \boldsymbol{c}_q^{old}}{|S_q^{old}|}$, $c_i^{old} = \boldsymbol{c}_i^{old} + \frac{\boldsymbol{z}_k - \boldsymbol{c}_i^{old}}{|S_i^{old}|}$
   $\bar{S}_q^{old} = \bar{S}_q^{old} \setminus \{\boldsymbol{z}_k\}$, $\bar{S}_i^{old} = \bar{S}_i^{old} \cup \{\boldsymbol{z}_k\}$,
   END
4. IF $(\exists q)(\bar{S}_q^{new} \neq \bar{S}_q^{old})$
   THEN for all such $q$ let $\boldsymbol{c}_q^{new} = \boldsymbol{c}_q^{old}$, $\bar{S}_q^{new} = \bar{S}_q^{old}$ and GOTO 2
5. STOP

After clustering each cluster is jointed with a corresponding child functional unit and consequently parameters of this functional unit are adjusted with regard patterns in the corresponding cluster only. In fact, division of input patterns into two or more disjoint sets, and consecutive learning over these subsets of patterns, put a separation hypersurface into the input space. The type of these hypersurfaces is defined by the type of transfer functions of switching unit parents.

So each building block is learned, output from each building block is propagated to all children and output of the top building block is considered as final output from the neural network.

## 3 Tuning of neural nets via GA procedure

Computational power of neural network depends in general on two aspects

1. structure and connection state (topology of the net)
2. learning method.

The second one is more or less question of amount of learning data and 'quality' of learning method. Main goal of each learning method is optimization of some fitness function which is defined on the set of all possible neural net parameters. We can insight this problem as global optimization of fitness function. There are many gradient algorithm based methods whose provide a possibility to find a local solution (minimum or maximum). But these methods does not allow to find a global extreme point of fitness function due to non-continuous substance of some parameters (types of transfer function, number of inputs, graph connections, etc.). We need to use some nongradient global optimization method to do this. But many deterministic techniques of global optimization, like divide and conquer, or analytic extreme, search aren't the most efficient on problems like acyclic graph construction and operations on it. It could be convenient to use some more natural ways to do so. Therefore we decided use genetic algorithms [7], [6] to set up parameters and topology of neural net. Theoretical analysis of GA follows that no best solution is reached but an average fitness increases at all in new generations. This is done due a special property of GA which is known as 'implicit parallelism' (see [5]). Shortly implicit parallelism theorem imply that

an exponentially large sets of parameter space are searched for domains with above-average fitness in polynomial time.

About the implementation, a queue (FIFO) of randomly generated networks is constructed (this queue has user predefined length). Then every network is learned, tested and evaluated with fitness, at this moment implemented in following ways (denote $S_a$ accepted signal, e.g. all signals correctly classified, $B_a$ accepted background, $S_r$ rejected signal, e.g. all misclassified signal, $B_r$ rejected background,):

1. maximal enrichment factor, e.g. $\max \frac{S_a}{S_r}$ under condition that the amount of accepted signal will be statistically significant
2. minimal loss of signal under condition on amount of rejected background, e.g. $\max S_a$ and $B_r$ =predefined value
3. maximal rejection of signal under condition on amount of accepted signal, e.g. $\max B_r$ and $S_a$ =predefined value
4. maximization of quality factor, e.g.
$$\max \frac{S_a}{\sqrt{S_a + B_a}}.$$
5. negative value of total mean square error over all events

The values of fitness of the network in queue builds actually the base for probability, according to which the parents of newborn network are chosen. After parents are given, some analysis comes to figure out, how those can be crossovered. Crossover itself is considered as an interchange of corresponding blocks (note that blocks have the same number of inputs and the same number of outputs, so they are replaceable), whose represent here the 'separated gen'. Since mutation should be of tiny use and its effect weak at all, we have some operation (edge removing, edge adding, activation function change, etc.) for disposal. The new structures is grown, then learned, tested and evaluated. And so on, until some criteria aren't reached. As those conditions may serve given generation count, acceptable level of specified fitness, etc.

### 3.1 Paralleling of neural net

The merit of our work lies in the application of GA procedures of crossover and mutation to find better topology and parameters of neural networks. A reasonable applicability of this approach assume to take a great population of individuals (learned and tested neural networks) and many generation of such populations. It is well known that learning of a neural net on large set of learning data should be very time consuming, hence GA procedures over such individuals should be time consuming naturally too. This leads us to such implementation which should be able to run effectively on more processors. Primarily we intend run our experiments on cluster of Linux PCs which is available in ISC AC CR. Todays, we have at our disposal 20-node cluster of PCs (from 600MHz one processor machines to 1GHz two processors machines) running Linux operating system. This allows us to run our experiments on parallel mode under

Portable Batch System environment, which is able to distribute separated tasks to processors with regard to their load.

To improve overall performance we implemented parallel version of neural net learning process also. First question about paralleling of our learning process solves the problem of level of paralleling. In our case, because of incomparable complicated optimization of transfer function with regard to other operations in neuron learning process, we decided to use the paralleling at this point. Program uses Parallel Virtual Machines library (PVM), so it could be executed, only if PVM installed. The count of slaves (whose are machines on PC-cluster or processors on multiprocessors system) is configured in PVM console. Significant advantage of paralleling based on PVM is that this allow us to run our networks effectively on wide range of architectures.

## 4 Application of neural network with switching units and genetic algorithms to Higgs boson search

As we already mentioned objective of our work is a search of $H \rightarrow b\bar{b}$ decay. Those data are produced during proton–proton collision with energy 14 TeV (in centroid mass system). There is a certain probability that Higgs bosons are produced in this collision (see Fig. 3, a)). In the case that mass of this Higgs boson is $m \leq 200 GeV/c^2$ a decay $H \rightarrow b\bar{b}$ is dominating. The main background of process above is process without production of Higgs boson but with the same final state (see Fig. 3, b)). Instead of Higgs boson in this case a gluon is radiated and produce a $b\bar{b}$ pair.

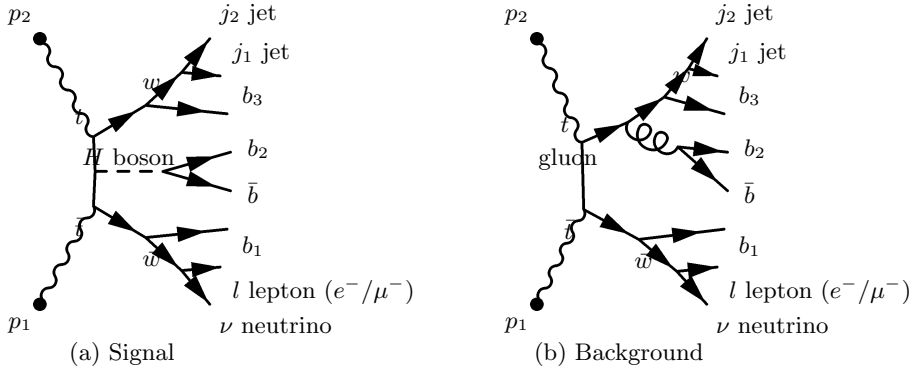



**Fig. 3.** Feynman diagram of decay trees.

Thus product of this type of collision are 2 jets from one $w$ decay, 4 b-jets and one lepton (electron or muon) plus missing energy from unobserved neutrino. Each visible particle (2 jets, 4 b-jets and lepton) is described by three values $P_T$ – transversal momentum $[GeV/c^2]$, $\eta$ – pseudorapidity and $\phi$ – direction

angles $[rad]$, corresponding to particle vectors with negligible mass. Neutrino is described by two missing energy values $E_x^{miss}$, $E_y^{miss}$. Pseudorapidity is an angular variable defined by

$$\eta = -\widetilde{ln}\left(\widetilde{\tan}\left(\frac{\theta}{2}\right)\right)$$

whose inverse function is

$$\theta = 2\widetilde{arctan}\left(e^{-\eta}\right).$$

Cartesian coordinates of each such particle can be expressed as $p_x = P_T \cos\phi$, $p_y = P_T \sin\phi$, $p_z = \frac{P_T}{\tan\theta}$. Using these values, we can evaluate values of energy $E_i$ for each jet

$$E_i = \sqrt{(p_x)_i^2 + (p_y)_i^2 + (p_z)_i^2}.$$

Those values allow evaluate effective masses $M_{i,j}$

$$M_{i,j} = \sqrt{\left(E_i + E_j\right)^2 - \left((p_x)_i + (p_x)_j\right)^2 - \left((p_y)_i + (p_y)_j\right)^2 - \left((p_z)_i + (p_z)_j\right)^2}$$

for all sensible tuples of particles.

Fundamental variable which can be used in Higgs boson search is a effective mass $M_{bb}$ of two $b$-'s which can arise either form Higgs boson decay or from gluon decay after $p\bar{p}$ collision. There is lot of events with gluon decay (background) and much less events with Higgs decay (signal). Each of these two classes of events have different statistics of effective mass $M_{bb}$. Statistics corresponding to Higgs boson decay is theoretically of Gaussian distribution with mean 120 $GeV/c^2$ and $\sqrt{\sigma} = 15GeV/c^2$, whereas statistics corresponding to gluon decay is much broader. Difference between those two statistics can be exploited to decide if Higgs boson decay is present in the data or not.

In addition, other physical reasons reject all events in which at least one of following conditions has been satisfied:

- at least one jets has $P_T < 15GeV$,
- at least one jet has pseudorapidity out of the range $(-2.5, 2.5)$ ,
- lepton is electron and $P_T^{lep} < 20GeV$
- lepton is muon and $P_T^{lep} < 6GeV$

All events passed those restrictions form histogram on Fig. 4.

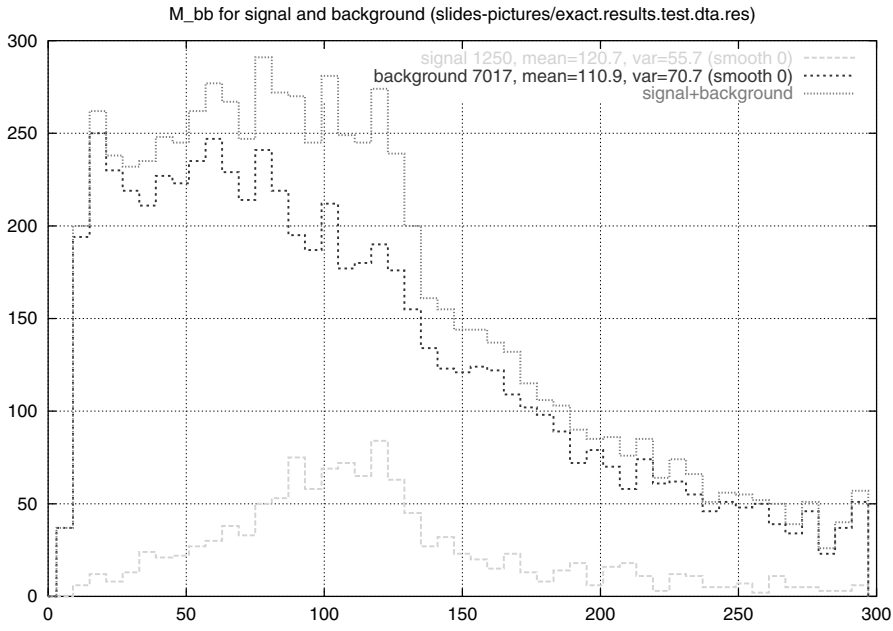



**Fig. 4.** Histogram of $M_{b\bar{b}}$ for signal and background (first number in the legend is number of events accepted by rejecting algorithm, means is average value of $M_{bb}$, var is mean square error and smooth is a smoothing factor used to plot the histogram).

Data really measured do not provide information about presence of Higgs decay in the event, hence for real data we have available total distribution of $M_{bb}$ (see Fig. 4, upper curve) only. For data simulated, we can plot two histograms of $M_{bb}$, one for background only and the second one for plain signal (see Fig. 4, two bellow curves).

Application of neural networks covers the case when we know distribution of separated signal and separated background (e.g. below curves in the Fig. 4) because neural networks should provide information if a given event is signal or background (up to some misclassification, of course). So the main idea how to exploit neural network to confirm Higgs decay presence is based on filtering of events in such a way that percentage of signal will be increased after filtering and at the same time significance $\frac{S_a}{\sqrt{S_a+B_a}}$ will stay on the same level.

## 5 Results and Conclusions

Some experiments were performed with signal and background data described in section 4. We use raw data as they were produced by package PYTHIA. Our first results are demonstrated on the plot 5.

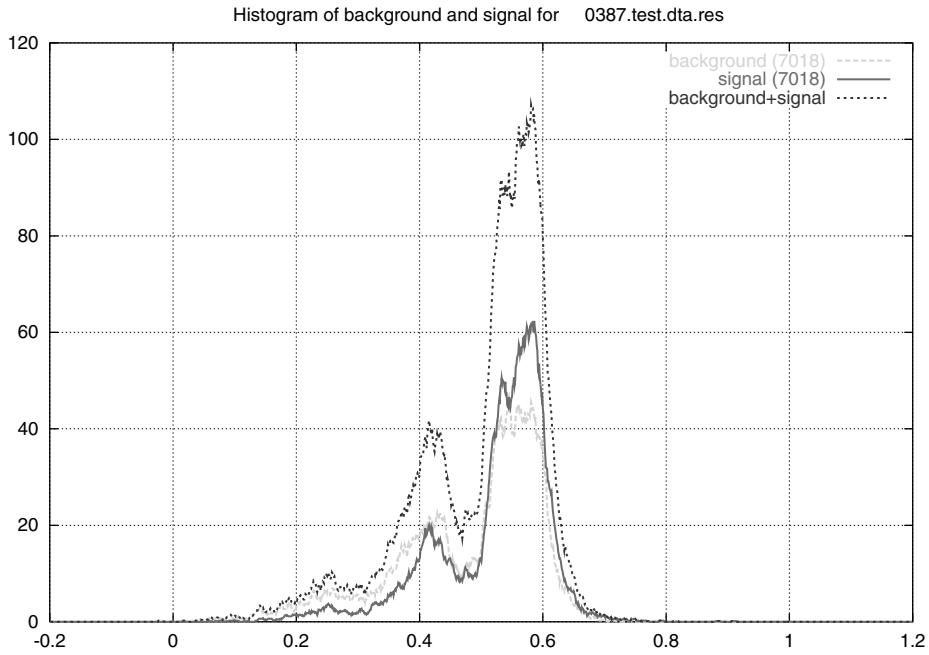


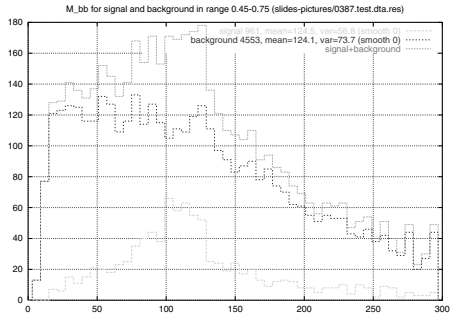

**Fig. 5.** Histogram of neural network output for signal and background.

It is evident that neural nets with switching units are able to partly separate signal with Higgs decay and background without Higgs decay. As we can seen on the plot mentioned, there is possible to choose an interval in which signal prevail over background, in case presented such interval should be $(0.45, 0.75)$ . We call such interval "best signal window". On the other hand we can take interval, in which the signal is suppressed and background will be of dominant importance. In the case discussed now as this interval should serve $(0.20, 0.45)$ . We call such interval consistently "best background window".

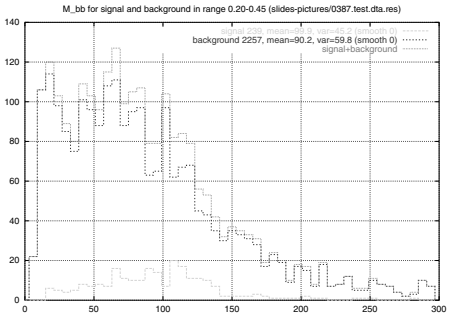
If Higgs decay is present than we can assume that plot of $M_{bb}$ over all event whose are mapped by neural network into the best signal window will differs from the next one, based on events mapped into the best background window. Really, for our simulated data these plots differ, see figures 6 a) and 6 b). We can see visual differences between these two plots. Of course these plots should be different from the resemble plot on Fig. 4.

Hence our first experiments convinced us that chosen approach to separation of Higgs decay seem to be applicable and promise useful detection methods.

Finally we point out that developed separation method based on neural networks with switching units and genetic optimization is universal separation method which can be used for various pattern recognition problem. Perhaps someone can find this method too extensive, especially GA part, but no effec-




(a) Signal window: $M_{bb}$ for events mapped into range 0.45-0.75.




(b) Background window: $M_{bb}$ for events mapped into range 0.20-0.45.

**Fig. 6.** Signal and background windows.

tive method for neural net topology and parameter tunning is known to this time. In the next, we plan to implement further transfer function for functional nodes, paralleling of transfer function optimization via taking some parallel version of optimization routine (parallel LAPACK, for example) and applicate this separation tool to another pattern recognition problems.

## References

1. F. Hakl and M. Jiřina. "Design of a neural net for level-two triggering in atlas nuclear experiments." *Neural Networks World*, vol. 6(6): pp. 951–973, 1996.
2. P. Bitzan, J. Šmejkalová, and M. Kučera, "Neural networks with switching units." *Neural Network World*, vol. 4, pp. 515–526, 1995.
3. M.Jiřina and M.Jiřina jr., "Neural network classifier based on growing hyperspheres" *Neural Network World*, vol. 3, pp. 417–428, 2000.
4. M. Sapinski, "The $t\bar{t}b\bar{b}$ background to the Higgs searches: Pythia versus CompHEP rates." Tech. rep. no. ATL-PHYS-2000-020, CERN Geneve, 2000.
5. John H. Holland. "Adaption in Natural and Artificial Systems." *A Bradford Book, The MIT Press*, 1992.
6. John R. Koza. "Genetic Programing." *A Bradford Book, The MIT Press*, 1992.
7. Lance Chambers (editors). "Practical Handbook of Genetic Algorithms." *CRC Press, Boca Racon*, 1995.
8. T. Sjöstrand. "PYTHIA 5.7 and JETSET 7.4 Physics and Manual." tech. report *CERN - TH . 7112/93*, dec 1993.