PHYSICAL REVIEW D **88,** 075016 (2013)

Determination of the CMSSM parameters using neural networks

Nicki Bornhauser and Manuel Drees

Physikalisches Institut and Bethe Center for Theoretical Physics, Universität Bonn, Nußallee 12, D–53115 Bonn, Germany

(Received 22 July 2013; published 24 October 2013)

In most (weakly interacting) extensions of the Standard Model the relation mapping the parameter values onto experimentally measurable quantities can be computed (with some uncertainties), but the inverse relation is usually not known. In this paper we demonstrate the ability of artificial neural networks to find this unknown relation, by determining the unknown parameters of the constrained minimal supersymmetric extension of the Standard Model from quantities that can be measured at the LHC. We expect that the method works also for many other new physics models. We compare its performance with the results of a straightforward χ^2 minimization. We simulate LHC signals at a center of mass energy of 14 TeV at the hadron level. In this proof-of-concept study we do not explicitly simulate Standard Model backgrounds, but apply cuts that have been shown to enhance the signal-to-background ratio. We analyze four different benchmark points that lie just beyond current lower limits on superparticle masses, each of which leads to around 1000 events after cuts for an integrated luminosity of 10 fb^{-1}. We use up to 84 observables, most of which are counting observables; we do not attempt to directly reconstruct (differences of) masses from kinematic edges or kinks of distributions. We nevertheless find that m_0 and $m_{1/2}$ can be determined reliably, with errors as small as 1% in some cases. With 500 fb^{-1} of data $\tan\beta$ as well as A_0 can also be determined quite accurately. For comparable computational effort the χ^2 minimization yielded much worse results.

DOI: 10.1103/PhysRevD.88.075016 PACS numbers: 11.30.Pb

I. INTRODUCTION

The Large Hadron Collider (LHC) is running successfully. After the next long shutdown the center of mass energy will be raised from 8 to 13 or 14 TeV. This higher center of mass energy will increase the reach for finding new physics. Here we are concerned with supersymmetric extensions of the Standard Model (SM) of particle physics. Within the simplest potentially realistic supersymmetric model, the minimal supersymmetric extension of the SM (MSSM) [1], the 14 TeV LHC will increase the mass reach for first generation squarks and gluinos from the current lower bounds, which reach nearly 1.5 TeV for equal squark and gluino masses[1] [4], to values between 2 and 3 TeV [5]. This leaves plenty of room for new discoveries. In particular, the "natural" range of parameters of supersymmetric theories will then be probed decisively.

Discovering a signal for physics beyond the SM, important as it would be, would certainly not be the end of the LHC physics program. One would then not only have to ascertain what kind of new physics has been discovered, but also determine the values of the free parameters as accurately as possible. In the context of the MSSM, this should help to unravel the mechanism responsible for the breaking of supersymmetry.

There is a great deal of literature on ways to determine the parameters of supersymmetric theories. Most methods start from kinematic features, in particular end points or "edges" of invariant mass distributions [6] or kinks in slightly more complicated kinematic distributions [7]. These kinematic features directly allow us to determine (differences of) superparticle masses; at least at the tree level, in most cases there is a direct relation between the mass of a superpartner and a weak-scale parameter of the underlying theory. In many cases the experimental resolution that can be achieved is such that at least one-loop corrections should be included; e.g. the difference between pole (on-shell) masses, which determine the kinematics, and $\overline{\text{DR}}$ masses, which are "fundamental" free parameters in the supersymmetric Les Houches accord [8], can easily reach several percent for strongly interacting superparticles [9]. The derived $\overline{\text{DR}}$ masses will then depend on many (pole) masses. Moreover, in the chargino and neutralino sector, as well as for third generation sfermions, the relation between pole masses and fundamental parameters is complicated by mixing [1]. Nevertheless the basic kinematic quantities that are used for parameter determination can be determined from a single (simulated) experiment. While the step from there to the determination of the basic parameters and their errors may entail many calls of spectrum calculators, there is no need to simulate event

[1]Mass limits are usually cited for the constrained MSSM (CMSSM) or for certain simplified scenarios; even in these scenarios the gluino mass bound reduces to about 900 GeV if squarks are much heavier than gluinos. In the more general MSSM the bounds may be somewhat weaker [2]; however, they can be reduced significantly only if strongly interacting sparticles are quite close in mass to the lightest superparticle [3], which is not particularly natural from the model building point of view.

generation for different sets of parameters, which is usually far more time consuming than the calculation of the spectrum of superparticles.[2]

On the other hand, even in constrained scenarios purely kinematical determinations of the underlying parameters work well only if sufficiently many events contain two (or more) charged leptons (meaning electrons or muons). Kinematic reconstructions based on jets suffer not only from the much poorer energy resolution of jets, but also from larger combinatorial backgrounds (since the production and decay of strongly interacting superparticles typically leads to events with many jets).

In any case, it is clear that the number of signal events in certain categories contains a lot of information about the underlying physics. Even if kinematic reconstruction works well, it would be wasteful to ignore this information. To mention a well-known example, the cross section for the pair production of a new color triplet complex scalar boson (like the top squark) is much smaller than that for spin-$1/2$ quarks of the same mass [1]. Moreover, in constrained supersymmetric scenarios strongly interacting superparticles tend to be heavier than those without strong interactions. The production of strongly interacting superparticles therefore frequently leads to long ''cascade'' decays [11], which can populate many ''topologically different'' final states, i.e. final states characterized by different numbers (and charges) of leptons as well as different numbers (and flavors) of jets. It has been recognized quite early that the relative abundance of these final states contains a great deal of information about the sparticle spectrum [12].

However, these early studies mostly focused on distinguishing qualitatively different spectra of superparticles. Information on the total signal rate has only quite recently been included in fits attempting to determine the underlying parameters from (simulated) events [13]. We are not aware of any study that attempts to determine the values of the underlying parameters using (mostly) counting observables, although a recent analysis showed that these observables can be very useful for discriminating between discrete sets of model parameters [14]. One major difficulty with this approach is that it requires us to generate event samples for many different assumed sets of input parameters. For example, even in the CMSSM, which has only four free parameters, a simple grid scan over all parameters with a step size comparable to or smaller than the anticipated statistical accuracy of the method is prohibitively CPU expensive in most circumstances.

The purpose of this paper is to demonstrate the usefulness of artificial neural networks for the parameter determination of such a new physics theory. Largely due to the limitation of our computational resources, we do this in the framework of the CMSSM; the method should also be useful for other theories, supersymmetric or otherwise. Again for computational simplicity we ignore detector effects, but we work at full hadron level, including initial and final state radiation, hadronization, and the underlying event. Similarly, we ignore Standard Model backgrounds, but we include cuts that should keep them at a manageable level. We consider four benchmark scenarios, all of which lie (slightly) beyond current lower bounds on superparticle masses, but have qualitatively different spectra. We find that 10 fb^{-1} of data at $\sqrt{s} = 14$ TeV are sufficient to determine the common scalar mass parameter m_0 and the common gaugino mass parameter $m_{1/2}$ to few percent accuracy *without* any direct kinematic mass reconstruction. With 500 fb^{-1} of data the neural networks can also determine the trilinear soft breaking parameter A_0 and the ratio of vacuum expectation values $\tan\beta$ quite accurately for these benchmark scenarios. In contrast, in many cases a simple χ^2 minimization failed to converge; i.e. it could not reliably determine the parameters and their errors. The likely reason is that the minimization of χ^2 is very sensitive to fluctuations in the predictions due to finite Monte Carlo statistics.

In this paper we are only interested in the production and decay of superparticles at the LHC, as an example for an extension of the SM containing many new parameters that can hopefully be determined from future LHC data. In our numerical analysis we will therefore respect the experimental bounds on the masses of superparticles, but we will not try to reproduce the recently discovered (increasingly) Higgs-like boson [15] in our CMSSM spectra, nor will we try to describe dark matter through thermally produced superparticles. Instead we are using the CMSSM as a toy model whose parameter space is manageable even without requiring the correct Higgs mass and dark matter relic density. Obviously imposing these constraints, or other constraints not directly related to LHC data, would simplify the task of fixing the free parameters. Here we wish to show that data on the production and decay of superparticles at the LHC by themselves can be used for this task, even if mostly counting observables are used.

The remainder of this article is organized as follows. In Sec. II we first introduce the general setting of the simulation. One important issue is the choice of observables. An automated reconstruction of the underlying parameters can only succeed if one has sufficiently many

[2]This statement may no longer be strictly true in the presence of additional hard radiation, which changes the shapes of kinematic distributions near the features (edges or kinks) used to determine masses [10]. Beyond the collinear approximation, the amount of radiation emitted may depend on (combinations of) superparticle masses in a complicated manner. However, in practice the shapes of the relevant kinematic distributions are fitted to (simulated) data rather than directly taken from theory. While we are not aware of an analysis of kinematical fitting based on a fully NLO corrected event simulation, we expect that by fitting the shapes of the relevant distributions one can still determine the locations of the relevant kinematic features without prior knowledge of superparticle masses.

observables to be sensitive to all parameters everywhere in parameter space. On the other hand, including too many observables can dilute the statistical power. We present a set of observables which we showed to be useful for discriminating between different parameter sets for a more general supersymmetric model with 15 parameters [14]. In the second part of Sec. II we introduce four different benchmark points in the CMSSM framework. In Sec. III we discuss both our attempts at parameter reconstruction, first using artificial neural networks and second a χ^2 minimization. We explain the general setup as well as each step of the creation of the neural networks for this specific application. We also estimate the errors on the CMSSM parameters, including their correlations, using different methods that yield consistent results. In the second part of Sec. III the attempted χ^2 minimization is discussed; as already mentioned, it does not perform very well. The results obtained by the artificial neural networks for all four benchmark points are discussed in Sec. IV. We also compare them to the results from the χ^2 minimization. Finally, the last section contains a summary and some conclusions.

II. SIMULATION

We simulate future LHC data at a center of mass energy of 14 TeV. As mentioned in the Introduction, we work in the framework of the CMSSM, where the entire spectrum of superparticles and Higgs bosons is defined by four continuous parameters and a sign. The continuous parameters are the common scalar mass parameter m_0, the common gaugino mass $m_{1/2}$, the common trilinear soft breaking parameter A_0 and the ratio $\tan\beta$ of vacuum expectation values of the two Higgs doublets. As usual, m_0, $m_{1/2}$ and A_0 are specified at the scale of Grand Unification, $M_X \simeq 2 \times 10^{16}$ GeV, whereas $\tan\beta$ is given at the electroweak scale. We fix the sign of the supersymmetric Higgsino mass parameter μ to be positive.

We use SOFTSUSY [16] to compute the CMSSM superparticle and Higgs boson spectra from the values of the four input parameters. The weak-scale spectrum is then passed on to SUSY-HIT [17], which calculates the branching ratios of all kinematically allowed decays. Knowledge of the superparticle masses and branching ratios is needed for the simulation of the production and decay of pairs of superparticles at the LHC, which is handled by the event generator Herwig ++ [18]. In a first step 10,000 events are simulated in order to determine the total cross section for the production of all superparticles for the given set of input parameters. Next, the appropriate number of events is simulated which corresponds to the assumed integrated luminosity; we will show results for 10 and 500 fb^{-1}.

Each simulated event is assigned to one of twelve mutually exclusive event classes, based on the number, charges and flavors of charged leptons. In addition, for each event a small number of mostly counting observables is kept, from which we construct our 84 observables. This is described in more detail in the following subsection. We do this, first of all, for four benchmark scenarios, which lie in qualitatively different regions of CMSSM parameter space, as described in the second subsection. Of course, in the attempt to determine the values of the CMSSM parameters from the four simulated measurements, the procedure from spectrum calculation to event generation has to be performed for many additional parameter sets, as described in Sec. III.

A. Observables and their covariances

In this subsection we summarize our observables, which we introduced in detail in Sec. 3 of [14]. In particular, the precise definitions of the objects (isolated charged leptons, hadronically decaying τ leptons, hadronic jets with or without b tag) we use to characterize the events, and the applied cuts, can be found in the appendixes of [14].

As already noted, we group all accepted events into twelve mutually exclusive classes, which differ by the number, charges and flavors of charged leptons. Here only isolated electrons or muons with transverse momentum $p_T > 10$ GeV and pseudorapidity $|\eta| < 2.5$ are counted. These classes are

(1) $0l$: Events with no charged leptons
(2) $1l^-$: Events with exactly one charged lepton, with negative charge (in units of the proton charge)
(3) $1l^+$: Events with exactly one charged lepton, with positive charge
(4) $2l^-$: Events with exactly two charged leptons, with total charge -2
(5) $2l^+$: Events with exactly two charged leptons, with total charge $+2$
(6) $l_i^+ l_i^-$: Events with exactly two charged leptons, with opposite charge but the same flavor; i.e. $e^- e^+$ or $\mu^+ \mu^-$
(7) $l_i^+ l_{j;j\neq i}^-$: Events with exactly two charged leptons, with opposite charge and different flavor; i.e. $e^- \mu^+$ or $e^+ \mu^-$
(8) $l_i^- l_j^- l_j^+$: Events with exactly three charged leptons with total charge -1. There is an opposite-charged lepton pair with same flavor. For example $e^- \mu^- \mu^+$ or $e^- e^- e^+$
(9) $l_i^+ l_j^+ l_j^-$: Events with exactly three charged leptons with total charge $+1$. There is an opposite-charged lepton pair with same flavor. For example $e^+ \mu^- \mu^+$ or $e^+ e^- e^+$
(10) $l_i^- l_j^- l_{k;k\neq j,i \text{ for } +}^{\pm}$: Events with exactly three charged leptons with total negative charge, i.e. there are at least two negatively charged leptons. There is *no* opposite-charged lepton pair with same flavor. For example $e^- e^- \mu^+$ or $e^- e^- e^-$
(11) $l_i^+ l_j^+ l_{k;k\neq j,i \text{ for } -}^{\pm}$: Events with exactly three charged leptons with total positive charge,

i.e. there are at least two positively charged leptons. There is *no* opposite-charged lepton pair with same flavor. For example $e^+e^+\mu^-$ or $e^+e^+e^+$

(12) $4l$: Events with four or more charged leptons

We distinguish between different charges of leptons since the initial state at the LHC carries charge +2. In general the number of events with positively charged leptons can therefore differ from those with negatively charged leptons. Moreover, we distinguish between lepton pairs with opposite charge but the same flavor, which can originate from leptonic neutralino decays, $\tilde{\chi}_a^0 \to l_i^+ l_i^- \tilde{\chi}_b^0$, and all other lepton pairs, which have to come from the decays of two different particles. This explains why we have two different classes of events with exactly one charged lepton, and four different classes each for events with exactly two and exactly three charged leptons, respectively. In principle we could also define several different classes of four lepton events. However, the number of such events is in any case rather small; further separating these few events into several classes is therefore not very useful.[3]

Our first observable is the total number of events after cuts, N. Note that the cuts differ for the different event classes, as described in Ref. [14]. In addition, for each of these twelve classes $c \in \{1, 2, \ldots, 12\}$, the values of seven observables $O_{i,c}$, $i \in \{1, 2, \ldots, 7\}$ are computed:

(i) $O_{1,c} = n_c/N$: The number of events n_c contained in the given class c divided by the total number of events N, i.e. the fraction of all events contained in a given class

(ii) $O_{2,c} = \langle \tau^- \rangle_c$: Average number of tagged hadronically decaying τ^- of all events within a given class c

(iii) $O_{3,c} = \langle \tau^+ \rangle_c$: Average number of tagged hadronically decaying τ^+ of all events within a given class c

(iv) $O_{4,c} = \langle b \rangle_c$: Average number of tagged b jets of all events within a given class c

(v) $O_{5,c} = \langle j \rangle_c$: Average number of non-$b$ jets of all events within a given class c

(vi) $O_{6,c} = \langle j^2 \rangle_c$: Average of the square of the number of non-b jets[4] of all events within a given class c

(vii) $O_{7,c} = \langle H_T \rangle_c$: Average value of H_T of all events within a given class c, where H_T is the scalar sum of the transverse momenta of all hard objects, including the missing p_T

Both τ and b jets have to have transverse momentum $p_T > 20$ GeV and pseudorapidity $|\eta| < 2.5$. In addition, a τ jet needs to be isolated, and a b jet has to contain a b-flavored hadron. Jets satisfying these criteria are tagged with an assumed tagging efficiency of 50%. Finally, H_T is the scalar sum of the transverse momenta of all hard objects (jets and charged leptons) and the absolute value of the missing p_T. We again refer to Ref. [14] for further details.

Three of those observables are different than the ones used in Ref. [14]. The number of events in a given class c that contain at least one tagged hadronically decaying τ^- divided by the total number of events in this class, $n_{c,\tau^-}/n_c$, has been replaced by the average number of tagged hadronically decaying τ^- of all events within a given class c, $\langle \tau^- \rangle_c$, and similarly for positively charged τ jets. In the parameter sets considered in [14] the number of events containing a tagged τ was rather small and the number of events containing two of those even smaller. Therefore it was sufficient to just count the number of events containing at least one tagged τ. Now in the case of the CMSSM there can be more events with a higher number of τ leptons. Therefore here we switched the observable to preserve more information about the measurement. The same applies to the observable $\langle b \rangle_c$, which is used instead of $n_{c,b}/n_c$.

Out of the 85 observables listed above, one should be discarded. Obviously the fractions of events n_c/N which belong to a certain class c add up to 1, because $\sum_{c=1}^{12} n_c = N$. We therefore do not include the fraction of events without charged leptons, n_{0l}/N, among our observables; note that this does not lead to any loss of information. We thus end up with 84 observables.

For the calculation of χ^2, and also in order to improve the performance of our artificial neural networks, we need the covariance matrix of all 84 observables. The variance of the total number of events after cuts, N, is

$$\sigma^2(N) = N. \tag{2.1}$$

The next twelve observables are the fractions of events n_c/N that belong to each class c. As mentioned before they are not independent. The covariance between the fraction of events in two different classes c and c' is then

$$\operatorname{cov}\left(\frac{n_c}{N}, \frac{n_{c'}}{N}\right) = \delta_{cc'} \frac{n_c}{N^2} - \frac{n_c n_{c'}}{N^3} \qquad (c, c' \in \{1, 2, \ldots, 12\}). \tag{2.2}$$

The covariance for identical classes ($c = c'$) equals the variance. Note that this matrix would be singular if we included all twelve $O_{1,c}$ among our observables. In contrast, n_c/N and the total number of events N are not correlated, i.e.

$$\operatorname{cov}\left(\frac{n_c}{N}, N\right) = 0 \qquad (c \in \{1, 2, \ldots, 12\}). \tag{2.3}$$

The remaining observables can be written as averages over all events in a given class, $O_{i,c} = \langle o_i \rangle_c$ with $o_2 = \tau^-$, $o_3 = \tau^+$, $o_4 = b$, $o_5 = j$, $o_6 = j^2$ or $o_7 = H_T$. Their variances can be calculated directly from the simulated data using the formula

[3] In fact, separating the events into too many distinct classes is harmful. The reason is that we wish to use Gaussian statistics; our observables become approximately Gaussian only in the limit of large event numbers. We therefore only use event classes containing some minimal number of events, as described in Sec. III.

[4] If event i in the given class contains $N_j^{(i)}$ non-b jets, then $\langle j^2 \rangle_c = 1/n_c \sum_{i=1}^{n_c} (N_j^{(i)})^2$.

$$\sigma^2(O_{i,c}) = \frac{1}{n_c - 1} \cdot (\langle o_i^2 \rangle_c - \langle o_i \rangle_c^2) \quad (i \in \{2, 3, \ldots, 7\}). \tag{2.4}$$

Of these observables, only $\langle j \rangle_c$ and $\langle j^2 \rangle_c$ are correlated within a given class:

$$\mathrm{cov}(\langle j \rangle_c, \langle j^2 \rangle_c) = \frac{1}{n_c - 1} \cdot (\langle j^3 \rangle_c - \langle j \rangle_c \langle j^2 \rangle_c). \tag{2.5}$$

Here $\langle j^3 \rangle_c$ is also determined directly from the (simulated) events. Observables from different classes are not statistically correlated. We also ignore the possible correlation between $\langle \tau^- \rangle_c$ and $\langle \tau^+ \rangle_c$. The validity of this approximation was checked for the closely related observables $n_{c,\tau^-}/n_c$ and $n_{c,\tau^+}/n_c$ in [14] and should also be fine here.

B. Benchmark points

We look at four different reference points in the CMSSM parameter space each yielding $O(1000)$ events after cuts for 10 fb^{-1} of data:

(1) $m_0 = 150$ GeV, $m_{1/2} = 700$ GeV, $\tan\beta = 10$ and $A_0 = 0$ GeV
(2) $m_0 = 2000$ GeV, $m_{1/2} = 450$ GeV, $\tan\beta = 10$ and $A_0 = 0$ GeV
(3) $m_0 = 1000$ GeV, $m_{1/2} = 600$ GeV, $\tan\beta = 10$ and $A_0 = 1500$ GeV
(4) $m_0 = 400$ GeV, $m_{1/2} = 700$ GeV, $\tan\beta = 30$ and $A_0 = 0$ GeV.

All points have a positive Higgs(ino) mass parameter μ. The resulting spectra of superparticles and Higgs bosons are given in Table I.[5]

The first parameter set has a low m_0, leading to small slepton masses. The sleptons can therefore be produced on shell in decays of the winolike states $\tilde{\chi}_2^0$ and $\tilde{\chi}_1^\pm$, which in turn are produced frequently in decays of $SU(2)$ doublet squarks $\tilde{u}_L$, $\tilde{d}_L$. Since these squarks can be produced either directly or in decays of gluinos, we expect this benchmark point to lead to relatively strong leptonic signatures.

In contrast, the second point has $m_0^2 \gg m_{1/2}^2$, so that squarks and sleptons have similar masses. The winolike $\tilde{\chi}_2^0$ and $\tilde{\chi}_1^\pm$ are still produced rather copiously in $\tilde{g}$ three-body decays via virtual $SU(2)$ doublet squark exchange; note that $\tilde{g}$ does not have any tree-level two-body decay in this scenario.[6] However, $\tilde{\chi}_2^0$ mostly decays into the light CP-even scalar Higgs boson h plus the lightest superparticle (LSP) $\tilde{\chi}_1^0$ here, while $\tilde{\chi}_1^\pm \to \tilde{\chi}_1^0 W^\pm$ decays have a branching ratio of 100%. A gluino decay on average produces 0.99 top (anti)quarks in this scenario, whose semileptonic decays can yield additional hard leptons; these final states are favored since renormalization group (RG) running and $L - R$ mixing reduce the masses of $\tilde{b}_1$ and $\tilde{t}_1$ relative to those of the other squarks [1]. Nevertheless we expect the strengths of the signals with three or more leptons to be much weaker in this scenario. Moreover, the signal in class 6 (two leptons with opposite charge and equal flavor) will not be enhanced relative to that in class 7 (two leptons with opposite charge and different flavor).

The third benchmark point also has a relatively high m_0, so that sleptons are again not produced in the decays of strongly interacting superparticles. However, it also has a rather high value of $|A_0|$, which enhances both the RG running and the $\tilde{t}_L - \tilde{t}_R$ mixing. This leads to a relatively large mass splitting between $\tilde{t}_1$ and the remaining squarks. The gluino mass is chosen such that all gluinos decay into a top quark and an anti–top squark, or vice versa. Since 60% of all $\tilde{t}_1$ decay into a top quark plus one of the neutralinos, a gluino decay therefore produces on average 1.6 top (anti) quarks. This yields high-multiplicity final states, including several b jets and frequently also hard leptons. However, the $\tilde{t}_1$ mass is still so high that direct $\tilde{t}_1 \tilde{t}_1^*$ pair production does not contribute appreciably to the overall signal for supersymmetry in this scenario.

TABLE I. CMSSM input parameters and selected superparticle and Higgs boson masses for our four benchmark points. All mass parameters are in GeV. Note that first and second generation sfermions with the same gauge quantum numbers have identical masses. Moreover, $m_{\tilde{d}_L} \simeq m_{\tilde{u}_L}$ in these scenarios, while $m_{\tilde{d}_R} \simeq m_{\tilde{u}_R}$, and $m_{\tilde{\nu}} \simeq m_{\tilde{e}_L}$. Similarly, $m_{\tilde{\chi}_1^\pm} \simeq m_{\tilde{\chi}_2^0}$ in all cases, while $m_{\tilde{\chi}_4^0} \simeq m_{\tilde{\chi}_2^\pm}$ are about 10 to 30 GeV above $m_{\tilde{\chi}_3^0}$. Finally, in all cases $m_H \simeq m_A \simeq m_{H^\pm}$.

	Point 1	Point 2	Point 3	Point 4
m_0	150	2000	1000	400
$m_{1/2}$	700	450	600	700
A_0	0	0	1500	0
$\tan\beta$	10	10	10	30
$m_{\tilde{g}}$	1570	1133	1407	1578
$m_{\tilde{u}_L}$	1437	2171	1575	1483
$m_{\tilde{u}_R}$	1382	2161	1541	1430
$m_{\tilde{b}_1}$	1322	1816	1402	1323
$m_{\tilde{b}_2}$	1371	2145	1529	1377
$m_{\tilde{t}_1}$	1118	1384	1168	1145
$m_{\tilde{t}_2}$	1357	1824	1414	1366
$m_{\tilde{e}_R}$	306	2005	1024	480
$m_{\tilde{e}_L}$	494	2015	1074	616
$m_{\tilde{\tau}_1}$	298	1988	1010	414
$m_{\tilde{\tau}_2}$	494	2007	1068	606
$m_{\tilde{\chi}_1^0}$	291	187	249	293
$m_{\tilde{\chi}_2^0}$	551	344	468	555
$m_{\tilde{\chi}_3^0}$	848	463	668	830
m_h	116	116	114	117
m_A	969	2040	1244	889

[5]Recall that we are not requiring the CMSSM to contain a Higgs boson with mass near 125 GeV in our analysis.

[6]Decays of the kind $\tilde{g} \to \tilde{\chi}_i^0 g$, which proceed via one-loop diagrams, are also included in SUSY-HIT [17], but their branching ratios are small.

TABLE II. Distribution of class events in percent into the twelve mutually exclusive lepton classes (determined from 500 fb^{-1} of simulated data), and their statistical errors for an integrated luminosity of 10 fb^{-1}, for the four benchmark points.

Class	Point 1	Point 2	Point 3	Point 4
1. $0l$	46.6 ± 2.1	47.1 ± 2.1	54.3 ± 2.2	61.2 ± 2.5
2. $1l^-$	12.2 ± 1.1	16.2 ± 1.2	15.0 ± 1.1	12.4 ± 1.1
3. $1l^+$	18.0 ± 1.3	16.9 ± 1.3	16.8 ± 1.2	15.5 ± 1.3
4. $2l^-$	1.6 ± 0.4	2.8 ± 0.5	1.7 ± 0.4	1.3 ± 0.4
5. $2l^+$	3.9 ± 0.6	3.3 ± 0.6	2.2 ± 0.4	1.9 ± 0.4
6. $l_i^+ l_i^-$	7.7 ± 0.8	4.3 ± 0.6	3.2 ± 0.5	2.8 ± 0.5
7. $l_i^+ l_{j;j\neq i}^-$	3.5 ± 0.6	5.0 ± 0.7	3.8 ± 0.6	2.9 ± 0.5
8. $l_i^- l_j^- l_j^+$	1.9 ± 0.4	1.4 ± 0.4	0.9 ± 0.28	0.6 ± 0.25
9. $l_i^+ l_j^+ l_j^-$	3.4 ± 0.6	1.4 ± 0.4	1.0 ± 0.3	0.7 ± 0.27
10. $l_i^- l_j^- l_{k;k\neq j,i \text{ for } +}^+$	0.1 ± 0.10	0.5 ± 0.22	0.3 ± 0.16	0.2 ± 0.14
11. $l_i^+ l_j^+ l_{k;k\neq j,i \text{ for } -}^-$	0.2 ± 0.14	0.5 ± 0.22	0.3 ± 0.16	0.3 ± 0.17
12. $4l$	0.8 ± 0.27	0.6 ± 0.24	0.5 ± 0.21	0.2 ± 0.14

Finally, the fourth benchmark point again has $m_0 < m_{1/2}$, but rather large $\tan\beta$. This increases the value of the τ Yukawa coupling, which in turn reduces the mass of the lighter $\tilde{\tau}$ eigenstate through RG effects as well as $\tilde{\tau}_L - \tilde{\tau}_R$ mixing. As a result, $\tilde{\tau}_1$ is significantly lighter than the other sleptons in this scenario. Although $m_{\tilde{e}_R} < m_{\tilde{\chi}_2^0} \simeq m_{\tilde{\chi}_1^\pm}$, essentially no $\tilde{\chi}_1^\pm$ and just 0.2% of all $\tilde{\chi}_2^0$ decay into a first or second generation charged slepton. The reason is that $\tilde{\chi}_1^\pm$ and $\tilde{\chi}_2^0$ are both winolike, whereas $\tilde{e}_R$ and $\tilde{\mu}_R$ are $SU(2)$ singlets. On the other hand, $\tilde{\tau}_1$ has a significant $SU(2)$ doublet component. As a result, about 77% each of all $\tilde{\chi}_1^\pm$ and $\tilde{\chi}_2^0$ decay into a $\tilde{\tau}_1$, which in turn always decays into a τ lepton and a LSP $\tilde{\chi}_1^0$; the remaining $\tilde{\chi}_2^0$ decay mostly into $h + \tilde{\chi}_1^0$, whereas the remaining $\tilde{\chi}_1^\pm$ decay into $W^\pm + \tilde{\chi}_1^0$. The decay products of strongly interacting superparticles therefore frequently contain one or more τ lepton(s). On the other hand, the branching ratios for gluino decays into third generation squarks are again enhanced, so that each gluino decay produces on average 0.99 top (anti)quarks. In this scenario supersymmetric events therefore can contain relatively soft leptons from leptonic τ decays and/or hard leptons from semileptonic top decays, in addition to τ and/or b jets.

Our qualitative expectations are confirmed by Table II, which lists the fractions of events (in percent) that are assigned to the twelve event classes. We also note that all benchmark points predict that more positively than negatively charged leptons are produced. This results because the proton contains more u quarks than d quarks, so that the production of $\tilde{u}$ squarks is enhanced relative to that of $\tilde{d}$ squarks. The asymmetry between positively and negatively charged leptons becomes large when squark production contributes prominently to the total event sample (after cuts) and (some) first generation squarks have sizable semileptonic decay branching ratios. Both conditions are satisfied for point 1 and, to a lesser extent, in point 4, whereas point 2 has a very small asymmetry owing to the large squark masses and small leptonic branching ratios of $\tilde{\chi}_1^\pm$.

Since we assume exact conservation of R-parity, the LSP is stable. In the four benchmark scenarios, it is the lightest neutralino $\tilde{\chi}_1^0$. Each supersymmetric event will therefore contain at least two $\tilde{\chi}_1^0$'s, which are not detected, leading to the celebrated missing transverse momentum signature. Indeed, our cuts always include the requirement of a sizable missing p_T, the exact value of the cut depending on the number of charged leptons in the event; we again refer to the appendix of Ref. [14] for further details.[7]

We find the following numbers of events for an integrated luminosity of 10 fb^{-1}:

(1) 1940 events before and 1080 after cuts
(2) 4080 events before and 1047 after cuts
(3) 1970 events before and 1135 after cuts
(4) 1618 events before and 991 after cuts

These numbers have been obtained from a simulation corresponding to an integrated luminosity of 500 fb^{-1}.

Evidently the number of events after cuts differs much less among the four benchmark points than the raw event number does. In particular, for the second benchmark point the large number of events before cuts is due to the small masses of the electroweak charginos and neutralinos. In principle these final states—in particular, $\tilde{\chi}_2^0 \tilde{\chi}_1^\pm$ production—can lead to final states with three charged leptons and little hadronic activity [19], which could contribute to our event classes 8 or 9. Recall, however, that $\tilde{\chi}_2^0$ has a very small leptonic branching ratio in this scenario. The $\tilde{\chi}$ pair

[7]Note that we apply a "Z veto" cut on events of class 6, which is not applied on events in class 7. Without this cut, in a theory respecting $e - \mu$ universality, as the CMSSM does, class 6 would have to contain at least as many events as class 7, since uncorrelated lepton pairs would contribute equally to both classes, while correlated same flavor lepton pairs only contribute to class 6.

events therefore have a very low cut efficiency; i.e. most of these events do not pass our cuts.

It is amusing to note that the total number of events after cuts alone would evidently not be sufficient to distinguish between benchmark points 1 and 3, nor between points 2 and 4. This illustrates the need to include (many) more observables when trying to discriminate scenarios, let alone for the quantitative determination of the values of the free parameters. Indeed, Table II shows that the event fractions in the twelve classes are quite sufficient to distinguish among the four benchmark scenarios. However, we do not merely wish to distinguish among benchmark points that are well separated in parameter space; we wish to quantitatively determine the values of the underlying parameters. This is discussed in the subsequent section.

III. STRATEGIES FOR DETERMINING THE PARAMETERS

In this section we discuss the strategies we employed to extract the values of the four CMSSM parameters from our 84 observables for the four benchmark points described in the previous subsection. We first describe the construction of an artificial neural network (ANN), and then an attempt based on the minimization of an overall χ^2 variable.

A. Neural network

Artificial neural networks have been used in high energy physics for more than 25 years [20]. Originally they were designed for comparatively simple tasks, like reconstructing single tracks from hit patterns in wire chambers. By now ANNs are used for a wide variety of purposes, from optimizing experimental searches for superparticles [21] to parametrizing parton distribution functions [22]. However, we are not aware of a previous application of ANNs to the LHC inverse problem.

Here we describe how to construct artificial neural networks that can find relations mapping the measured observables onto the CMSSM parameters whose values we wish to determine. Mathematically speaking, an ANN is a function mapping input parameters, to be provided by the user, onto output parameters, whose values the user wishes to determine. It can learn this function from training sets via a well-defined algorithm. A training set consists of input values (the observables of simulated experiments) and the corresponding output values (the CMSSM parameters). Note that the parameters that are the input into an event generation program (in our example, the CMSSM parameters) are the output of the neural network, whereas the output values of the event simulation (our 84 observables) are input of the neural network. It is hoped that by training the neural network on sufficiently many sets of input and output parameters, it will learn to derive the correct values of the output parameters also for sets of input parameters that are not contained in the training set. We will show that, at least for our four benchmark scenarios, this indeed works quite well.

As noted above, a neural network is a mathematical function; "learning" means that the numerical coefficients defining this function are adjusted. By choosing different values for these coefficients, in principle nearly every possible function can be reproduced with some accuracy. During the learning process these coefficients are set such that the deviations between the calculated network outputs for the specific training inputs and the known desired (correct) outputs are reduced as far as possible. If the training set is a fair representation of the investigated CMSSM parameter space, at the end the resulting neural network function should be able to compute the correct CMSSM parameters for an arbitrary set of input parameters, assuming that the latter indeed can be reproduced in the CMSSM.

1. Setup

General information about neural networks can e.g. be found in [23]. Generally speaking, an ANN consists of a set of "neurons," collected in "layers," and "weights" assigned to the connections among these neurons. A neuron can be described by a simple function computing one output value from one input value. At least two layers of neurons are needed: one for the input, and one for the output. In the input layer, there is one neuron for each observable, plus a "constant" neuron which always outputs 1; this is needed in order to be able to describe a constant function, i.e. a component of the network's output that is independent of the input. Similarly, there is one output neuron for each quantity that we wish to determine. The inputs of the neurons in the input layer are normalized versions of our observables, and the output of the neurons in the output layer gives the network's prediction for a normalized version of the quantity whose value we wish to determine; this normalization is described below. In between the input and output layers, there can be an arbitrary number of layers of hidden neurons; in physics applications this number is usually 1 or 2. All neurons in adjacent layers are connected to one another, and a numerical weight is assigned to each of these connections. These weights are the coefficients mentioned above, whose values are to be determined during the training of the network.[8]

The ANN we construct here contains a single layer of hidden neurons, as shown in Fig. 1. For all neurons in the input and output layers, the output is simply equal to the input. In our case the input layer consists of 85 neurons, one for each (normalized) observable with input values x_i, $i = 1, \ldots, 84$; the output of the ith neuron in this layer is therefore also given by x_i. As noted above, there is an additional input neuron with a fixed output, $x_0 = 1$.

[8]The term "artificial neural network" reflects the fact that this construction yields a very simple model of actual biological neural networks; however, the (dis)similarity between ANNs and actual brains is immaterial for our discussion.

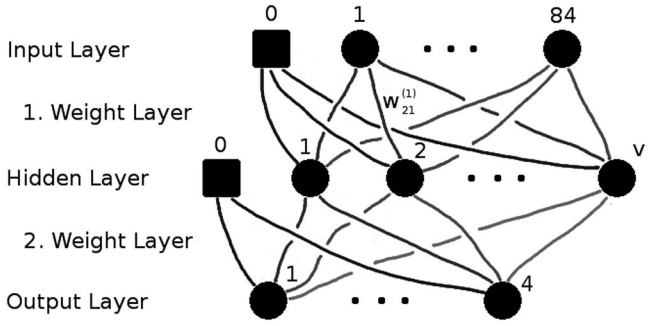

FIG. 1. Neural network with one layer of hidden neurons, i.e. two weight layers. The squares represent the neurons with constant output and the circles the neurons with variable output, which depends on the network input. All neurons of adjacent layers are connected to one another. Each connection can have a different weight; e.g. the connection between the first input neuron and the second hidden neuron has the weight $w^{(1)}_{21}$.

The second layer consists of a yet to be determined number v of "hidden" neurons with input values $z_1, \ldots, z_v$. These input values are weighted sums of the output of all neurons in the input layer:

$$z_a = \sum_{i=1}^{84} w^{(1)}_{ai} x_i + w^{(1)}_{a0} = \sum_{i=0}^{84} w^{(1)}_{ai} x_i. \tag{3.1}$$

Here and in the following we will use $i, j, \ldots$ to label the input neurons, $a, b, \ldots$ to label the hidden neurons, and $r, s, \ldots$ for the output neurons. Each hidden neuron processes its input by applying the function $\tanh(z_a)$. This function is commonly chosen for the hidden neurons, because it exhibits approximately linear behavior for small $|z_a|$ but saturates at ± 1 for large values of $|z_a|$. This ensures that the ANN can reproduce a wide variety of functions.[9]

The last layer of the ANN consists of the output neurons, four in number for CMSSM with values $y_1, \ldots, y_4$.[10] Their input values are calculated by weighted sums over the output of the hidden neurons:

$$y_r = \sum_{a=1}^{v} w^{(2)}_{ra} \tanh(z_a) + w^{(2)}_{r0} = \sum_{a=0}^{v} w^{(2)}_{ra} \tanh(z_a); \tag{3.2}$$

the output of the zeroth hidden neuron has been fixed to $\tanh(z_0) = 1$. These four output values should reproduce the (properly normalized) values of the CMSSM parameters; the normalization is explained below.

Two "weight layers" connect the three layers of neurons. The first weight layer connects each input neuron with each hidden neuron (except for the zeroth hidden neuron whose output value is fixed to 1). The weights $w^{(1)}_{ai}$ with $a = 1, \ldots, v$ and $i = 0, \ldots, 84$ are assigned to these connections. Similarly, the connections between he hidden and output neurons have weights $w^{(2)}_{ra}$ with $r = 1, \ldots, 4$ and $a = 0, \ldots, v$ and form the second weight layer. Evidently the number of hidden neurons determines the number of free parameters describing the ANN. The goal of the training process is to find appropriate (ideally, optimal) values for these weights. Before we describe the algorithm used to set the weights, we discuss the normalization of the input and output values, and the initialization of the weights.

2. Normalization

The 84 inputs into our ANN are obtained by normalizing the 84 measured observables. This simplifies the initialization of the weights (see below), which in turn results in a shorter learning process. To that end, both the input and output values of the ANN should be (dimensionless) $\mathcal{O}(1)$ quantities. Because the hyperbolic tangent function also naturally leads to hidden layer output values of $\mathcal{O}(1)$, the weights in both layers could be of a similar order. In most cases the tanh functions will then not be in the saturation regime, where it becomes almost independent of the input value. In general this should speed up the learning process.

We normalize the 84 input quantities O_i independently. Let O^n_i be the value of observable O_i for the nth training set, with $n = 1, \ldots, N$. The mean value $\bar{O}^{tg}_i$ and the variance $(\sigma^{tg}_i)^2$ over all N training sets are then

$$\bar{O}^{tg}_i = \frac{1}{N} \sum_{n=1}^{N} O^n_i; \tag{3.3}$$

$$(\sigma^{tg}_i)^2 = \frac{1}{N-1} \sum_{n=1}^{N} (O^n_i - \bar{O}^{tg}_i)^2. \tag{3.4}$$

From these, we compute normalized inputs x_i for the ANN:

$$x_i = \frac{O_i - \bar{O}^{tg}_i}{\sigma^{tg}_i}. \tag{3.5}$$

The mean value of the x_i over the training sets is thus zero, with a standard deviation equal to 1; hence the (absolute values of) the inputs should be $\mathcal{O}(1)$, as desired.[11]

[9] Due to the saturation of the tanh function at large absolute values of its arguments, an ANN constructed in this way may have difficulty reproducing a function which grows or shrinks very quickly when its input variables are varied. Recall, however, that the input of the ANN (during training) is the output of event simulation. In extensions of the Standard Model, the number of events of a given type expected at the LHC typically changes quickly when the model parameters are varied. This implies that the true values of the model parameters change relatively slowly when the observables are varied. Our ANN should be well suited to describe this kind of behavior.

[10] For reasons that will be explained below, at the end we actually do not use one common network for all four CMSSM parameters but instead four separate ANNs with one output parameter each.

[11] Equivalently, one could use the original observables O_i as input, and assign the function (3.5) to the ith input neuron instead of the identity function.

Since we want the ANN outputs y_r to also be $\mathcal{O}(1)$ in absolute value, they are related to the CMSSM parameters $r \in \{m_0, m_{1/2}, \tan\beta, A_0\}$ which we wish to determine via

$$y_r = \frac{r - \bar{r}^{tg}}{\sigma_r^{tg}}. \tag{3.6}$$

The averages $\bar{r}^{tg}$ and the variances $(\sigma_r^{tg})^2$ are calculated analogously to Eqs. (3.3) and (3.4), respectively.

The normalization of both the input and the output is thus calculated from the training sets. The same normalization is then used for the control sets, and for every other ANN input including the (actual or simulated) measurement. The implicit assumption is that the training sets are a good representation of the investigated CMSSM parameter space. If this condition is not satisfied, the ANN results for the CMSSM parameters are in any case likely to be quite inaccurate.

3. Initialization

The normalization of the overall input and output of the ANN ensures that these quantities are $\mathcal{O}(1)$ in absolute size, without preference for either positive or negative values. The weights in the two weight layers should be initialized such that these conditions are also satisfied for the inputs of the neurons in the hidden and output layers. One simple, and by experience quite effective, way to ensure this is to initialize all weights with Gaussian random numbers with mean zero and variance σ^2 equal to the number of neurons in the layer beneath the given weight layer. Hence we set the Gaussian variance for the weights in the first weight layer to $1/85$, and for the second layer to $1/(\upsilon + 1)$, where υ is as before the number of hidden neurons.

4. Learning procedure

The initial ANN will in general provide a very poor approximation of the desired function. In the case at hand, the initial ANN will most likely give values for the CMSSM parameters that are very far from the true values.

The ANN therefore needs to be "trained," so that it can "learn" to closely reproduce the desired function. To that end it has to be confronted with sufficiently many "training sets," where *both* the input and the desired output are known. In the case at hand, this means that we had to simulate many sets of CMSSM parameters, and compute the corresponding values of our 84 observables, along the lines described in Sec. II. More details on the choice of the training sets are given below.

An ANN which is trained with specific training sets reproduces the desired output for these training sets more and more exactly with every learning step. If the network includes sufficiently many hidden neurons it will eventually simply "memorize" the training sets, i.e. reproduce their outputs exactly, if the training runs long enough. This may seem desirable at first sight, but actually it is not. The task of the ANN is to interpolate between the training sets; i.e. it should provide (approximately) the correct output also for inputs that are *not* part of the training sets. Experience shows that at some point further improvement in the reproduction of the training sets *degrades* the performance of the ANN when applied to different inputs.

In order to determine when the training of the ANN should be terminated one therefore also needs "control sets." These are generated exactly like the training sets, but they are *not* used in the training of the ANN. Instead, they are used to define a "control error." The training of the ANN is stopped when this control error reaches its minimum. This strategy ensures good performance of the neural network as long as both the training and the control sets are good representations of the investigated CMSSM parameter space.

We use the following normalized control error:

$$\tilde{F} = \sqrt{\frac{\sum_{n=1}^{M} (\vec{y}^n - \vec{k}^n)^2}{\sum_{n=1}^{M} (\bar{\vec{k}} - \vec{k}^n)^2}}, \tag{3.7}$$

where M is the number of control sets, $\vec{y}^n$ are the four output values combined in one vector, calculated by the ANN from the input values of the nth control set, $\vec{k}^n$ are the correct output values and $\bar{\vec{k}}$ is the mean value over the output values of all control sets:

$$\bar{\vec{k}} = \frac{1}{M} \sum_{n=1}^{M} \vec{k}^n. \tag{3.8}$$

The only quantities in Eq. (3.7) that change in the course of the training are the vectors $\vec{y}^n$, which depend on the current values of the weights defining the ANN. These weights are modified iteratively, such that an error analogous to that defined in Eq. (3.7), but computed from the training sets rather than from the control sets, is minimized. Mathematically this amounts to minimizing a (complicated) function of the (many) variables $w_{ai}^{(1)}$, $w_{ra}^{(2)}$. We do this using the "conjugate gradient" algorithm, as described in the Appendix.

Recall that $\vec{k}^n$ and $\vec{y}^n$ are normalized output values. If the ANN has several outputs, the contribution of each output variable to the total error therefore depends on the spread of this variable within the training sets. The algorithm for adjusting the weights ensures that the total training error decreases during the training; however, this does not guarantee that the performance for each individual output, i.e. for each component of $\vec{y}$, improves monotonically. We therefore found it convenient to define four separate ANNs for the four CMSSM parameters, as already noted. For each of our ANNs the vectors $\vec{y}$ and $\vec{k}$ collapse to simple real numbers. The splitting into four separate network also offers the possibility of a further specialization of each network. Furthermore we have smaller

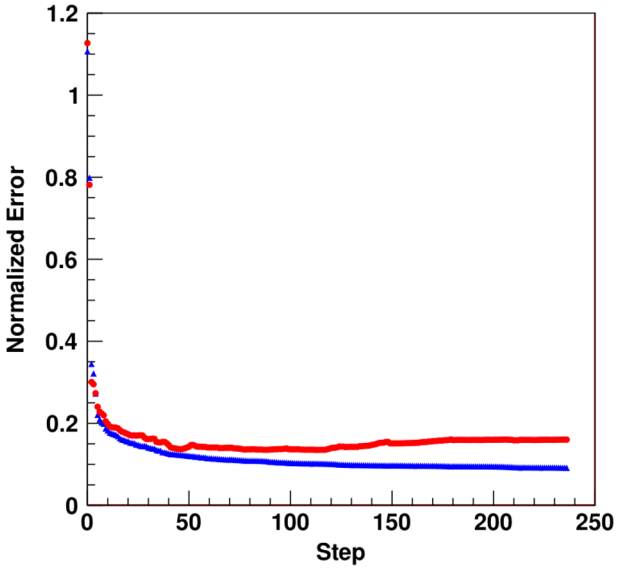

FIG. 2 (color online). Error evolution of a neural network with 20 hidden neurons for the parameter $m_{1/2}$ of benchmark point 3. The blue triangles show the normalized training error and the red dots the corresponding normalized control error. The minimum of the control error is reached with 0.136 at learning step 107.

networks, whose training takes less time than the training of one big network with a higher number of hidden neurons. Note, however, that all four ANNs have identical inputs; i.e. each of them has 84 input neurons.

The training of the ANNs thus proceeds as follows. The initial values of the weights are used to define initial errors for the training and control sets. The weights are then adjusted iteratively, such that the error computed for the training sets is minimized. After each learning step the control error is calculated. The learning process is stopped when this control error reaches its minimum.[12] Since the control error is not used for the determination of the new weights, it need not decrease monotonically during the iteration. The learning procedure should therefore not be stopped until one can be reasonably sure that the *absolute* minimum of the control error has been passed. Finally, the weights have to be reset to the values that gave this absolute minimum.

This is illustrated in Fig. 2. The triangles (blue) show the normalized training errors and the dots (red) the control errors. We see that both errors initially decrease very quickly. The control error reaches a first minimum after about 45 learning steps, increases again, and finally reaches its absolute minimum after 107 steps. In contrast, the training error keeps decreasing until the iteration is stopped after 235 steps; the control error at the end of the iteration is nearly 10% larger than its absolute minimum. At the end the ANN is therefore reset to its status after 107 learning steps.

5. Training and control sets

The performance of the trained ANNs obviously depends on the quality of the training and control sets. Note first of all that these sets are supposed to show the true relation between the 84 observables and the CMSSM parameters. The errors on the observables in the training and control sets, in particular the error due to finite Monte Carlo statistics, should therefore be (much) smaller than the statistical accuracy of the (simulated) measurement that the ANNs are finally meant to analyze. We therefore used 500 fb^{-1} of simulated data for the training and control sets; of course, the number of events is then rescaled to the assumed luminosity of the "measurement."

Moreover, we already emphasized that the training and control sets should be good representations of the investigated CMSSM parameter space. Therefore for each benchmark point we set different ranges for the input parameters of the simulation; the actual CMSSM parameters for each training and control set are chosen randomly using a flat distribution.[13]

In general it should be easier to determine the parameters m_0 and $m_{1/2}$, since they influence the superparticle spectrum more strongly than the other two continuous parameters, A_0 and $\tan\beta$, do. One combination of m_0 and $m_{1/2}$ can e.g. be determined with some accuracy simply from the total rate of signal events after cuts. Next, one could use some inclusive kinematical observable (e.g. H_T), and/or the jet multiplicity in signal events, to determine the region of interest in the $(m_0, m_{1/2})$ plane. In the following we therefore use smaller parameter ranges for these parameters than for A_0 and $\tan\beta$. We do this just to save CPU hours. This slight "cheat" allows us to generate, with our limited computer resources, sufficiently many training and control sets in the vicinity of our benchmark points to allow good interpolation. The above qualitative discussion indicates that our ANNs could most likely also be trained on the "entire" CMSSM parameter space. However, we would then need to generate (many) more training and control sets. Moreover, the ANNs would presumably need more hidden neurons, making their training even more time consuming.

We use the following parameter ranges for our four benchmark points:

(1) m_0: 100–350 GeV, $m_{1/2}$: 660–740 GeV, $\tan\beta$: 5–45 and $|A_0| \leq 2 \cdot m_0$

[12]Had we used a single ANN, with a combined control error as defined in Eq. (3.7), the training would have to be stopped when the *total* error was minimal. This does not guarantee that each of the four individual errors, on the four CMSSM parameters, is minimal. This is another argument in favor of using four separate ANNs.

[13]Furthermore each simulation in Herwig ++ is done with a different flatly, randomly chosen seed to prevent any possible correlations.

(2) m_0: 1850–2200 GeV, $m_{1/2}$: 410–490 GeV, $\tan\beta$: 5–45 and $|A_0| \leq 2 \cdot m_0$
(3) m_0: 850–1200 GeV, $m_{1/2}$: 560–640 GeV, $\tan\beta$: 5–45 and $|A_0| \leq 2 \cdot m_0$
(4) m_0: 300–550 GeV, $m_{1/2}$: 660–740 GeV, $\tan\beta$: 10–50 and $|A_0| \leq 2 \cdot m_0$

The range for A_0 depends on the value chosen for m_0 in order to make sure that there is no problem with the generation of the superparticle spectrum in SOFTSUSY.[14] We simulate slightly more than 1000 training sets and around 300 control sets for each benchmark point. We checked that adding additional control sets does not improve the results; recall that these sets are only needed to determine when the training of the ANNs should be terminated. The total number of simulated parameter sets was determined by our available computing resources. Note, however, that the average distance between neighboring values of any CMSSM parameter in the training sets is much smaller than our final estimate of the error with which this parameter can be determined by our ANNs (see Sec. IV), at least for the smaller integrated luminosity of 10 fb^{-1}; i.e. the final (multidimensional) error ellipsoid should already contain many training sets. Further increasing the number of training sets is therefore not likely to significantly improve the final performance of our ANNs.

6. Improving the performance

The "measured" observables resulting from each simulated parameter set have statistical uncertainties, which are saved in the covariance matrix. Recall, however, that only the values of the observables themselves, but not the corresponding covariance matrix, are used as input into our ANNs. It would be desirable if the ANN knew with what precision an input observable is determined in order to be able to decide how important its value is for the determination of the CMSSM parameters. Recall in particular that different observables can have quite different (relative) errors; see Table II. All else being equal, observables with larger uncertainties should contribute with smaller weights. It is therefore quite evident that knowledge of the covariance matrix should improve the performance of our ANNs significantly.

Recall that we use 500 fb^{-1} of simulated data for the training and control sets. This means that the Monte Carlo (theory) error is essentially negligible when comparing with simulated measurements based on an integrated luminosity of 10 fb^{-1}. We will also show results for simulated measurements which also assume an integrated luminosity of 500 fb^{-1}. Here the Monte Carlo theory error is still significant; our computer resources do not allow us to reduce it further.

Directly feeding the (nonvanishing) entries of the covariance matrix as additional inputs into our ANNs would greatly increase their complexity. Instead, we take two measures in order to include the effect of (statistical) uncertainties.

First, we simply omit very noisy observables, i.e. quantities that have very large errors. To that end, as in the χ^2 calculation in [14], we require a minimal number of events of a given class c for taking the observables $O_{i,c}$ into account. Obviously observables computed from a small number of events will have a large error. For the 500 fb^{-1} training and control sets the observables $O_{i,c}$ are only included if the total number of events in a given class after cuts satisfies $n_c \geq 500$; the total number of events is only included if it exceeds 50 (in practice this is always the case). For the 10 fb^{-1} measurements the corresponding thresholds are changed to 10 and 1, respectively. The luminosity dependence of these thresholds means that for a specific set of CMSSM parameters the same observables are included (up to statistical fluctuations).

Recall that an ANN is defined with a fixed number of input neurons; therefore we still have to assign some value to each observable. The normalized input value of an observable which does not fulfill the minimal number of events is therefore set to zero. This means that this input neuron does not contribute to the weighted sums which are calculated within the neural network, independent of the weight of the connections exiting this neuron.

This procedure ensures that all nonzero inputs into our ANNs should have some statistical power for determining the CMSSM parameters we are after. However, this still does not tell the ANNs the relative accuracies of these observables. To that end, we use our knowledge of the covariance matrix of the observables for a particular training set to determine an 84-dimensional (correlated) Gaussian distribution. Here both the mean values and the covariance matrix are taken from the training sets, independent of the luminosity of the (simulated) measurement to which the ANN will eventually be applied. Of course, the overall width of these multidimensional Gaussian distributions (one for each original set of CMSSM parameters chosen for a given training set) should scale like the inverse square root of the integrated luminosity. However, the purpose of this trick is to teach the ANN the *relative* size of the errors on the various observables that are fed into the ANN, so that it can assign appropriate weights. This *relative* weighting should be independent of the luminosity.

For each combination of CMSSM parameters chosen for the training sets, we then randomly generate 100 sets of observables from the corresponding up-to-84-dimensional Gaussian distribution.[15] From the point of view of the ANNs, these sets have (slightly) different inputs, but exactly *the same* outputs (CMSSM parameters). Altogether

[14] $|A_0| \gg m_0$ could e.g. result in tachyonic sfermions.

[15] Note that these "satellite sets" are produced directly from the multidimensional Gaussian; no additional event generation is needed here.

each ANN is thus trained on slightly more than 100,000 sets of inputs yielding slightly more than 1000 different outputs.

It is interesting to note that this enlarged set of training sets can no longer be described as a function in the mathematical sense: due to statistical fluctuations, there might be training sets with the same (sets of) observable(s) but *different* output values.[16] In fact, for each observable we have a certain range of values, depending on the corresponding variance, that can lead to the same CMSSM parameters. The bigger this range is the less important the input value should be for the parameter determination. If the variance is large, during the learning process the ANNs are confronted with strongly varying input values for the same output value. This should allow them to recognize that this observable cannot contribute much to the determination of the CMSSM parameters, by reducing the appropriate weight. So the creation of Gaussian distributed variants of the original training set should lead to a higher weighting of input values with small errors.

7. Output error

We finally end up with four trained ANNs per benchmark point, one for each CMSSM parameter. The 84 observables obtained from any (simulated) experiment can now be used as input for the ANNs, which will then produce their estimates of the corresponding CMSSM parameters. In order to be able to judge the accuracy of these estimates, it is crucial to also obtain estimates for the uncertainties; ideally we would like to be able to quote the parameters with well-defined (Gaussian) statistical errors. Our ANNs by themselves do not provide such estimated errors.

We investigated two different ways to determine the errors on the outputs. The first possibility is to expand the set of input values of the measurement to multiple Gaussian distributed input sets. As in the construction of the "satellite sets" from the original training sets, the known covariance matrix of the measurement can be used to create a multitude of Gaussian distributed versions of the (simulated) measurement. Each version is then used as an input for the ANNs. If Gaussian statistics is applicable, the ANNs' outputs should also form Gaussian distributions. The square root of the variance of an output distribution then gives the error of the corresponding CMSSM parameter.

This is illustrated in Fig. 3, which shows the distribution of reconstructed $m_{1/2}$ values for benchmark point 3. This distribution has been obtained by feeding the appropriate ANN with 100,000 Gaussian distributed versions of the simulated 10 fb^{-1} measurement. We see that the true value $m_{1/2} = 600$ GeV is recovered without significant offset.

[16]Of course, the ANNs still *are* functions, assigning a unique output value to each set of input values.

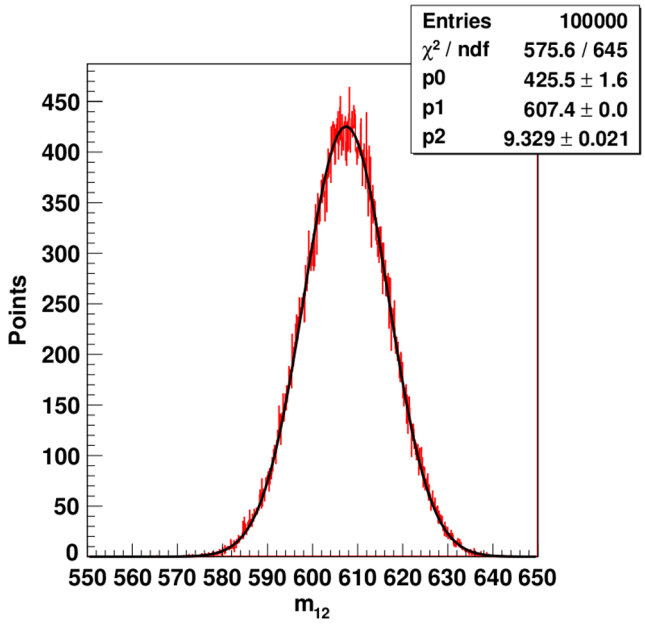

FIG. 3 (color online). One-dimensional distribution of reconstructed $m_{1/2}$ values formed by feeding an ANN with 100,000 Gaussian distributed versions of 10 fb^{-1} measurements for benchmark point 3. The fitted Gaussian distribution has the form $g(m_{1/2}) = p_0 \cdot \exp(-1/2[(m_{1/2} - p_1)/p_2]^2)$; i.e. p_1 is the mean value and p_2 is the standard deviation. The neural network contains 20 hidden neurons and underwent 107 learning steps with a final control error of 0.136. The true value is $m_{1/2} = 600$ GeV.

Moreover, the distribution can be described very well through a Gaussian fit. The result of this ANN for benchmark point 3 is thus $m_{1/2} = (607.4 \pm 9.3)$ GeV.

In order to determine the covariance matrix among the four CMSSM parameters we fit two-dimensional Gaussians to the distribution of each pair of outputs. To that end we feed the same sets of input values into both relevant ANNs; each set of input values then gives one pair of output values. We repeat that for all Gaussian distributed input sets. The set of output pairs then forms a two-dimensional (correlated) distribution and the corresponding variances and the covariance can be determined through a fit. Within rounding errors, e.g. due to different binning, the variances should be exactly the same as the ones determined from one-dimensional Gaussians.

This is illustrated in Fig. 4, which shows the two-dimensional distribution of reconstructed $(m_0, m_{1/2})$ values for benchmark point 3. The left frame shows a three-dimensional view and the right frame a two-dimensional contour plot. The fit yields $m_0 = (1034 \pm 42.3)$ GeV and $m_{1/2} = (607.5 \pm 9.3)$ GeV; the result for $m_{1/2}$ agrees with that of Fig. 3, as expected. There is a very mild negative correlation between these two parameters, with correlation coefficient $\rho_{m_0 m_{1/2}} = -0.291$. This coefficient is related to the entries of the covariance matrix through the formula

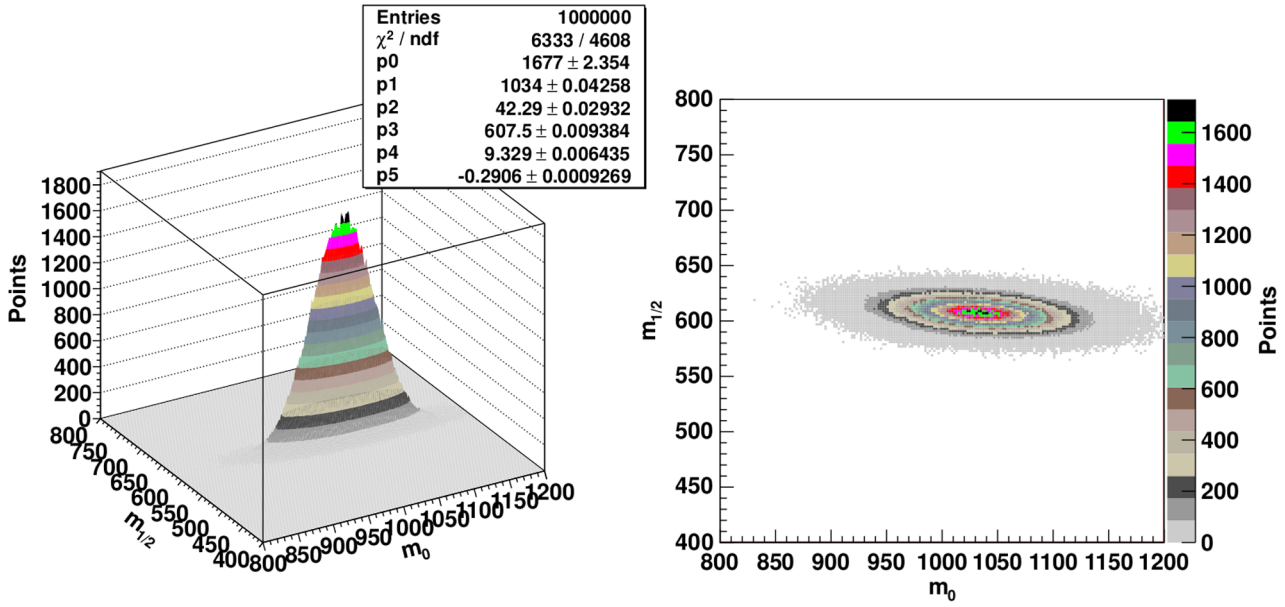

FIG. 4 (color online). Two-dimensional distribution of reconstructed $(m_0, m_{1/2})$ pairs (in GeV) formed by feeding both relevant ANNs with 1,000,000 Gaussian distributed versions of 10 fb^{-1} measurements for benchmark point 3. The fitted Gaussian distribution has the form $g(m_0, m_{1/2}) = p_0 \cdot \exp[-0.5/(1-p_5^2) \cdot ([(m_0 - p_1)/p_2]^2 + [(m_{1/2} - p_3)/p_4]^2 - 2p_5/(p_2 p_4) \cdot (m_0 - p_1) \cdot (m_{1/2} - p_3))]$; i.e. p_1 and p_2 are the mean value and standard deviation of m_0, and p_3 and p_4 are the mean value and standard deviation of $m_{1/2}$. The correlation factor $p_5 \equiv \rho$ is given by Eq. (3.9). The ANN for m_0 contains 15 hidden neurons and underwent 244 learning steps with a final control error of 0.106, and the one for $m_{1/2}$ contains 20 hidden neurons and underwent 107 learning steps with a final control error of 0.136. The true values are $m_0 = 1000$ GeV and $m_{1/2} = 600$ GeV.

$$\rho_{m_0 m_{1/2}} = \frac{\mathrm{cov}(m_0, m_{1/2})}{\sigma_{m_0}\sigma_{m_{1/2}}}. \tag{3.9}$$

Equivalently, the angle ϕ between the major axis of the error ellipse and the x-axis in the right frame of Fig. 4 is

$$\tan(2\phi) = \frac{2\rho_{m_0 m_{1/2}}\sigma_{m_0}\sigma_{m_{1/2}}}{|\sigma_{m_0}^2 - \sigma_{m_{1/2}}^2|}. \tag{3.10}$$

The ellipse is tilted with $\phi = -3.84°$ to the right bottom.

The second method of determining errors on the parameter estimates of the ANNs uses Gaussian error propagation. Since we know the covariance matrix of the input values and the neural network functions we can directly calculate the variances and covariances of the output values. The variance of the output value $r \in \{m_0, m_{1/2}, \tan\beta, A_0\}$ is then

$$\sigma_r^2 = \sum_{i=1}^{N_O}\left(\frac{\partial f_r^*}{\partial O_i}\sigma(O_i)\right)^2 + 2\cdot\sum_{i=1}^{N_O-1}\sum_{j=i+1}^{N_O}\frac{\partial f_r^*}{\partial O_i}\frac{\partial f_r^*}{\partial O_j}\mathrm{cov}(O_i, O_j). \tag{3.11}$$

Here $\sigma(O_i)$ is the standard deviation of the input observable O_i and $\mathrm{cov}(O_i, O_j)$ is the covariance between the observables O_i and O_j; explicit expressions for these quantities are listed in Sec. II A. The sums run only over those observables that have not been set to zero when we removed observables with large statistical uncertainty as described in Sec. III A 6; i.e. N_O is in practice significantly smaller than $N_{O,\max} = 84$.

Finally, f_r^* is the function mapping the (unnormalized) observables onto the estimated CMSSM values. Its derivative has the form[17]

$$\frac{\partial f_r^*}{\partial O_i} = \frac{\partial x_i}{\partial O_i}\frac{\partial f_r^*}{\partial f_r}\frac{\partial f_r}{\partial x_i} = \frac{\sigma_r^{tg}}{\sigma_i^{tg}}\cdot\frac{\partial f_r}{\partial x_i}$$
$$= \frac{\sigma_r^{tg}}{\sigma_i^{tg}}\cdot\left[\sum_{a=1}^{\upsilon} w_a^{(2)} w_{ai}^{(1)}\left(1 - \tanh^2\left(\sum_{j=1}^{N_O} w_{aj}^{(1)} x_j + w_{a0}^{(1)}\right)\right)\right]. \tag{3.12}$$

Recall that υ is the number of hidden neurons. Moreover, as described in Sec. III A 2 the ANNs, described by the functions f_r, are trained to produce inversely normalized output values from normalized inputs; the corresponding derivatives $\partial x_i/\partial O_i$ and $\partial f_r^*/\partial f_r$ can be read off Eqs. (3.5) and (3.6), respectively. The training set standard deviation σ_i^{tg} for the ith input value is calculated using Eq. (3.4) and similarly σ_r^{tg} for the output value r. The x_i are the normalized input values of the (simulated) measurement; again only the N_O inputs that have not been set to zero need to be taken into account.

Similarly the covariance between two CMSSM values r and s is

[17]Since each ANN has only one output, we removed the first index r on the weights $w_{ra}^{(2)}$ in the second layer; cf. Eq. (3.2).

$$\mathrm{cov}(r,s) = \sum_{i=1}^{N_O} \frac{\partial f_r^*}{\partial O_i}\frac{\partial f_s^*}{\partial O_i}\sigma^2(O_i) + \sum_{i=1}^{N_O-1}\sum_{j=i+1}^{N_O} \mathrm{cov}(O_i,O_j) \cdot \left(\frac{\partial f_r^*}{\partial O_i}\frac{\partial f_s^*}{\partial O_j} + \frac{\partial f_r^*}{\partial O_j}\frac{\partial f_s^*}{\partial O_i}\right). \quad (3.13)$$

It is reassuring to note that the covariances derived in this way are close to the values obtained by fitting Gaussians to the distributions of reconstructed CMSSM parameters. For example, for benchmark point 3 we find $\sigma(m_0) = 47.3$ GeV, $\sigma(m_{1/2}) = 11.5$ GeV and $\rho_{m_0 m_{1/2}} = -0.309$; recall that the corresponding values derived from the Gaussian fits are 42.3 GeV, 9.3 GeV and -0.291, respectively. We will see later that the two methods agree even more closely if a higher integrated luminosity is used for the measurement.

8. Number of hidden neurons

In order to determine the appropriate number of hidden neurons for each ANN we start with a low number (like 10) and train the corresponding network as described in Sec. III A 4. Note that each additional hidden neuron adds $85 + 1 = 86$ free parameters to the description of ANNs with one output neuron. As discussed in the Appendix, the training of the ANN involves the repeated computation of a matrix whose dimension grows linearly with υ. For a fixed number of learning steps the time needed to train an ANN therefore scales like υ^2. Hence the number of hidden neurons should not be increased needlessly.

After the training, the distribution of estimated outputs for the given benchmark point are calculated as described in the first part of Sec. III A 7. If this distribution is (approximately) a Gaussian the number of hidden layer neurons should be sufficient. On the other hand, if the distribution does not look very Gaussian the number of hidden layer neurons is increased and the process is iterated, until a satisfactory result is achieved.[18]

This completes our discussion of the construction of the ANNs. Before reporting our results for the four benchmark points, we describe an alternative strategy to determine the CMSSM parameters, based on χ^2 minimization.

B. χ^2 minimization

The preceding discussion shows that quite a lot of (computational) effort is required to construct ANNs that are able to produce reliable estimates of CMSSM parameters from simulated measurements. Moreover, a trained ANN is a "black box"; it is very difficult to get a feeling for how the input affects the output.

We therefore also attempted to derive estimates for the CMSSM parameters by minimizing a χ^2 function. At least conceptually this is far simpler than training ANNs. However, it turns out that fluctuations due to the finite Monte Carlo statistics used in deriving the predictions of the model make it very difficult to derive estimates for the parameters with statistically meaningful errors in this way. In this subsection we describe the method we used for the χ^2 minimization and show the results for benchmark point 4.

We wish to minimize the χ^2 between the simulated measurement and predictions, which is given by

$$\chi^2_{MP} = \sum_{i,j=1}^{N'_O} (O_i^M - O_i^P) V_{ij}^{-1} (O_j^M - O_j^P). \quad (3.14)$$

O_i^M are the observables of the simulated measurement and O_i^P are the predictions for these observables derived for a particular set of CMSSM parameters. The double sum runs over all observables that have been derived from a minimum number of events (see below), and V_{ij}^{-1} is the inverse covariance matrix of the relevant observables, with entries

$$V_{ij} = \mathrm{cov}(O_i^M, O_j^M) + \mathrm{cov}(O_i^P, O_j^P). \quad (3.15)$$

The second term in Eq. (3.15) takes into account the statistical error on the prediction due to the finite size of the Monte Carlo sample that has been used for its calculation. In [14] it has been shown that this function has the statistical properties of a true χ^2 distribution, provided we only include observables that have been computed from sufficiently many events. In particular, for 10 fb^{-1} of simulated integrated luminosity the total number of events is taken into account if the measurement or the prediction has at least one event after cuts; for our benchmark points this is always the case. The event ratio $n_c/N = O_{1,c}$ for a class c is included in the definition of χ^2 if the class contains at least 10 events for the measurement *or* the prediction. All other observables $O_{i,c}, i \geq 2$ are only included if $n_c \geq 10$ for both the measurement *and* the prediction.[19] For a higher integrated luminosity the thresholds are scaled up accordingly.

Depending on the number of events that are generated, i.e. the assumed integrated luminosity, computing the prediction for a single set of CMSSM parameters can already be quite costly in terms of CPU time. We therefore perform the χ^2 minimization in two steps. The first step is supposed

[18]The ANNs were often not trained until the global minimum of the control error was reached before looking at the distribution of output values. Instead this distribution was checked already when the first local minimum of the control error was reached; if this distribution looked very non-Gaussian, the ANN was discarded and υ was increased.

[19]Some of these minimal event numbers differ from the ones used in [14]. The benchmark points investigated here yield on the order of $O(1000)$ events after cuts for 10 fb^{-1} of data, as shown in Sec. II B. In contrast the parameter sets analyzed in [14] yielded on average around 25,000 events after cuts. Using 10 as minimal n_c for all classes showed slightly better results than the old choice of 10 for n_c/N and $\langle H_T\rangle_c$, 50 for $n_{c,b}/n_c$ and $\langle j\rangle_c$, and 500 for $n_{c,\tau^+}/n_c$ and $n_{c,\tau^-}/n_c$.

to roughly identify the correct region of parameter space in which the minimum lies; the second step should then pin down the location of the minimum. In both steps we compute the predictions using 10 fb^{-1} of simulated data.

In the first step we use an algorithm based on simulated annealing. We begin by computing the prediction for a randomly chosen set of parameters, and compute the corresponding χ^2. We then randomly vary one of the CMSSM parameters, leaving the other three parameters unchanged. If the resulting χ^2 is smaller than the previous one, the corresponding parameter set is taken as the new best guess for the location of the minimum. If the new χ^2 value is bigger than the previous one, there is still a finite probability that the new parameters are picked as the new best guess; this probability decreases exponentially with the difference between the two χ^2 values. This approach should prevent the algorithm from getting stuck in a local minimum. The same parameter would then be changed again, if a new minimum was not picked; this loop is terminated if a given number of tries did not improve the minimization. After that the second parameter is changed and so on. When all four parameters have been scanned in this manner, the scanning starts at the first parameter again. The procedure is continued until a given number of steps (here 250) is reached.

Recall that our goal in the first step is just to get a rough estimate of the CMSSM parameters. Therefore, we did not put much effort into setting the available parameters of the simulated annealing algorithm. Nevertheless this step is useful, in particular for narrowing down the range for m_0. The size of m_0 approximately determines the maximal allowed range of A_0. This is important for the second step of the minimization, in which the parameters should be determined more exactly.

In this second step we use the χ^2 minimization algorithms "Simplex" and "Migrad" of TMinuit [24] in the program ROOT. Using this algorithm we set allowed parameter ranges in order to avoid parameter selections where simulation problems occur (e.g. very large $|A_0|$ can lead to problems with the generation of the superparticle and Higgs spectrum). These algorithms will also work better if the starting point is closer to the true minimum and its distance to the minimum is roughly known. Therefore we use the output of the simulated annealing algorithm as starting point; the p value corresponding to this starting point gives us a rough idea how close we are to the true minimum.

These algorithms eventually do converge onto a new best guess for the location of the minimum. Since Minuit has been designed specifically for χ^2 minimizations, it even gives estimates for the statistical errors of the determined parameters. However, it turns out that these error estimates are almost always much too small. This can be traced back to the statistical fluctuations of the predictions. The final minimum found is nearly always produced by a rather extreme Monte Carlo fluctuation in the prediction. Such a fluctuation is likely to occur only in a region of parameter space around the true CMSSM parameters; i.e. the location of the minimum is likely not too far from the true location of the minimum computed from predictions with negligible error. However, in our simulation the fluctuation frequently reduced χ^2 by several units, leading to a very steep, but spurious, minimum. The width of this spurious minimum, which is what Minuit attempts to estimate, is then a very poor estimate for the actual error on the CMSSM parameters.

This problem can be ameliorated by increasing the integrated luminosity used for the computation of the predictions, which reduces the fluctuations. Alas, these fluctuations only decrease inversely to the square root of the number of generated events, whereas the CPU time needed to generate them obviously scales linearly with this number. Note that Minuit needs several hundred search steps to converge on a minimum; i.e. several hundred sets of CMSSM parameters have to be simulated for each χ^2 minimization by Minuit. The available time then limits the number of events we can generate for each parameter set. We found that even using an integrated luminosity of 500 fb^{-1} instead of 10 fb^{-1} for the predictions did not solve the problem of the extreme statistical fluctuations leading to too small errors.

We therefore derive final estimates for the CMSSM parameters, and their errors, by simulating up to 500 parameter sets (on several CPUs) with an integrated luminosity of 500 fb^{-1} around the minimum found by TMinuit, and calculate the corresponding χ^2 values.[20] These points are then used to fit χ^2 as a quadratic function of m_0, $m_{1/2}$, $\tan\beta$ and A_0:

$$\begin{aligned}\chi^2 &= \chi^2_{\text{min}} + \Delta\chi^2 \\ &= \chi^2_{\text{min}} + \begin{pmatrix} m_0 - m_0^{\text{min}} \\ m_{1/2} - m_{1/2}^{\text{min}} \\ \tan\beta - \tan\beta^{\text{min}} \\ A_0 - A_0^{\text{min}} \end{pmatrix}^T \\ &\quad \cdot V^{-1} \cdot \begin{pmatrix} m_0 - m_0^{\text{min}} \\ m_{1/2} - m_{1/2}^{\text{min}} \\ \tan\beta - \tan\beta^{\text{min}} \\ A_0 - A_0^{\text{min}} \end{pmatrix}. \end{aligned} \tag{3.16}$$

Here m_0^{min}, $m_{1/2}^{\text{min}}$, $\tan\beta^{\text{min}}$, A_0^{min} are our final estimates of the CMSSM parameters from the χ^2 minimization, χ^2_{min} is the corresponding minimal χ^2 value and V^{-1} is the final estimate for the inverse covariance matrix of the extracted parameters, from which we compute the errors on the estimated parameters including their correlations.

[20]Of course, the predicted number of events is divided by 50 when comparing a 500 fb^{-1} prediction to a 10 fb^{-1} measurement.

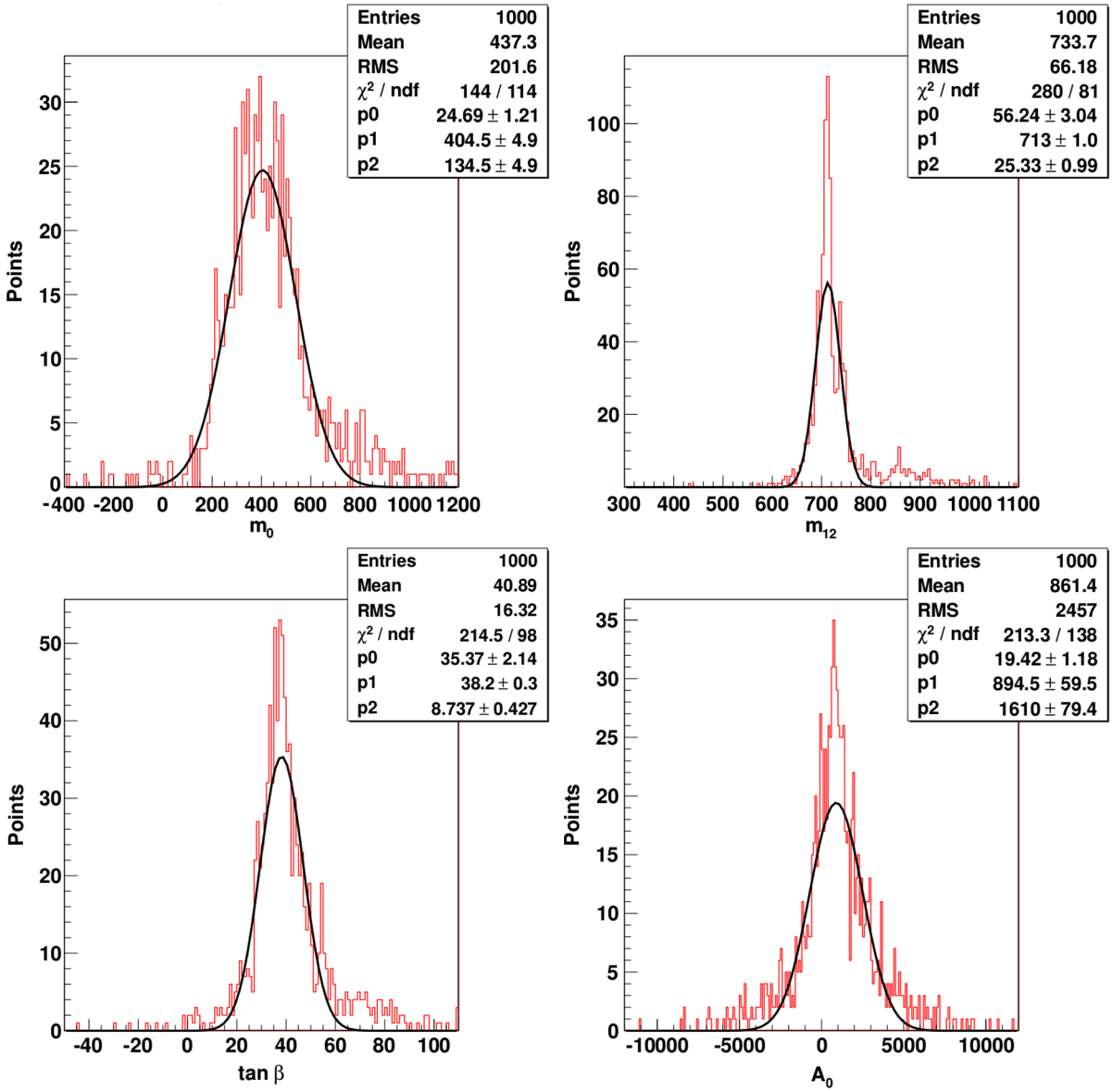

FIG. 5 (color online). Distributions of output values of the χ^2 minimization for benchmark point 4, for a simulated measurement with integrated luminosity of 10 fb^{-1}. m_0, $m_{1/2}$ and A_0 are given in GeV. 1000 Gaussian distributed versions of the original simulated measurement are fed into (3.17) with $d = 3$ to determine 1000 sets of CMSSM parameters. The distributions of these parameters (histograms in the four frames) are fitted to Gaussians (curves). "Mean" and "RMS" are the usual mean value and standard deviation of the entries in the histograms. The other parameters are derived from the Gaussian fits, $g(x) = p_0 \cdot \exp(-1/2[(x - p_1)/p_2]^2)$ with x being the parameter in question. The x axes are centered on the input values of these parameters, $m_0 = 400$ GeV, $m_{1/2} = 700$ GeV, $\tan\beta = 30$ and $A_0 = 0$.

Note that Eq. (3.16) contains 15 free parameters (χ^2_{min}, the values of the CMSSM parameters, and the entries of the covariance matrix, which is a symmetric 4×4 matrix); we again use "Simplex" and "Migrad" in TMinuit to determine them. The large number of parameter sets used in this fit, as well as the large integrated luminosity used for each parameter set, should reduce the effect of statistical fluctuations in the prediction. Specifically, we determine the free parameters of Eq. (3.16) by minimizing the summed differences

$$\sum_{n=1}^{500} \frac{[\chi^2_n - \chi^2(m_0^{(n)}, m_{1/2}^{(n)}, \tan\beta^{(n)}, A_0^{(n)})]^2}{(\chi^2_n)^d}. \tag{3.17}$$

Here χ^2_n is the χ^2 value of parameter set n and $\chi^2(m_0^{(n)}, m_{1/2}^{(n)}, \tan\beta^{(n)}, A_0^{(n)})$ is the calculated χ^2 which has been computed as in Eq. (3.16) from the current values of the fit parameters. Finally, the parameter d determines the weight of parameter sets that are far away from the χ^2 minimum; $d = 0$ means that all 500 parameter sets have

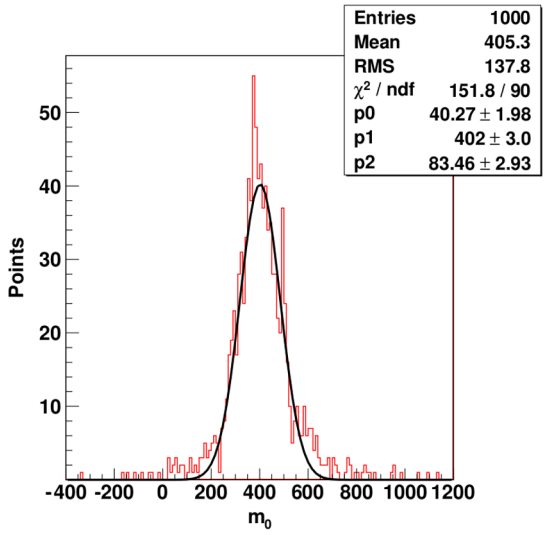

FIG. 6 (color online). The same distribution as in Fig. 5, but based on a simulated measurement with 500 fb^{-1} of data.

equal weight, whereas a positive d suppresses the weight of parameter sets with high χ^2 values which should be farther away from the minimum. Note that χ^2 can be expected to be a quadratic function of the CMSSM parameters only in the vicinity of its minimum; further away χ^2 may change quite quickly, e.g. if some new decay modes open up. Using the functional form (3.16) also for parameter sets which are far away from the minimum may therefore distort the fit. This argues in favor of using a positive d.

Unfortunately we find that the choice of d has a relatively big influence on the outcome of the fit. We therefore need a criterion to determine the optimal choice of d. Recall that in the ANN analysis we determined the number of hidden neurons by requiring that a Gaussian distribution of input sets also produces a Gaussian distribution of output values. We want to apply the same criterion here. Note, however, that for each set of measurements we need to redo the fit of Eq. (3.16); since 15 parameters need to be determined in each fit, it takes some amount of CPU time.[21] We therefore "only" create 1000 Gaussian distributed variants of the measurement (rather than 100,000 in the ANN analysis) and compare each of these "data sets" with all predictions. This leads to 1000 sets of fitted CMSSM parameters $m_0^{\min}$, $m_{1/2}^{\min}$, $\tan\beta^{\min}$ and $A_0^{\min}$. The distributions of those CMSSM parameters should then also be Gaussians. We therefore use the form of these distributions to check the reliability of our χ^2 minimization method.

An example of this check is shown in Fig. 5, which depicts the situation for benchmark point 4 and a simulated measurement with integrated luminosity of 10 fb^{-1}. Here $d = 3$ has been used in Eq. (3.17). We see that the distributions of all four CMSSM parameters have significant non-Gaussian tails. These become even more prominent if we only use 300, rather than 500, predictions for the final fit of the CMSSM parameters in Eq. (3.17); this indicates that the distribution of reconstructed CMSSM parameters might become somewhat more Gaussian if more than 500 predictions are used. However, the change between the results based on 300 and 500 predictions is not dramatic, so it seems unlikely that properly Gaussian distributions of the extracted CMSSM parameters can be obtained from a number of predictions that can be generated (by us) within a reasonable amount of time. We conclude that, while the central values of the derived CMSSM parameters are within one estimated standard deviation from their true values, the error estimates are not reliable.

This is also indicated by Fig. 6, which shows the distribution of reconstructed m_0 values for a simulated measurement with fifty times higher integrated luminosity. Under the usual scaling law, the statistical error should then decrease by a factor $\sqrt{50} \simeq 7$. Instead our procedure gives an error which is less than a factor of 2 smaller than that for 10 fb^{-1} of data. Note that we have increased the minimal number of events in a given class we require if the corresponding observables are to be included into the overall χ^2 also by a factor of fifty; i.e. observable $O_{i,c}$ is included only if $n_c \geq 500$. This should ensure that Gaussian statistics should be applicable everywhere. In Fig. 6 we nevertheless still see substantial non-Gaussian tails.

Another problem is that the results of this method depend sensitively on the power d in Eq. (3.17). This can be seen from Fig. 7, which shows the distribution of reconstructed m_0 values for the same simulated measurements based on 500 fb^{-1} used in Fig. 6, but using $d = 2$ and $d = 4$ instead of $d = 3$. Evidently these distributions no longer have much resemblance to Gaussians. Moreover, the estimated error on m_0 differs by a factor of 2 between these two choices.

As noted above, the fit based on Eqs. (3.16) and (3.17) which yields the estimates of the CMSSM parameters also provides estimates of the statistical errors on these extracted parameters. While the errors estimated in this way generally are of the same order of magnitude as the errors obtained in the Gaussian fits shown in Figs. 5 to 7, the numerical values of the errors fluctuate quite widely within the sample of 1000 versions of the measurement shown in these figures. In fact, the estimates of the errors fluctuate much more than the central values do.

[21]Strictly speaking we would have to redo the entire χ^2 minimization for each version of the simulated measurement, including the first two steps. However, the region of parameter space populated by the 500 sets of predictions used in Eq. (3.16) is sufficiently large that it should include the results of the first two steps as applied to the variants of the original measurement. Since steps 1 and 2 require additional event generation, and are thus very time consuming, redoing the entire χ^2 minimization for these sets was in any case not a realistic option for us.

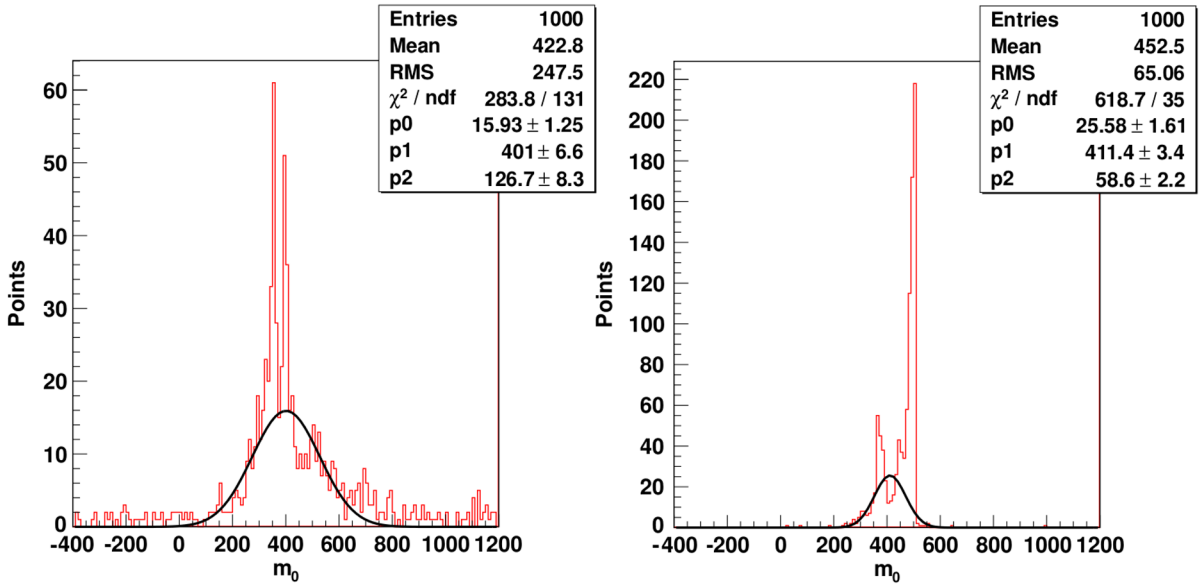

FIG. 7 (color online). The same distribution as in Fig. 6, but using $d = 2$ (left) and $d = 4$ (right) in Eq. (3.17), instead of $d = 3$.

We finally note that the error estimates derived from the χ^2 minimization, while quite uncertain, are generally significantly larger than those found by the ANNs, which *are* reliable as illustrated above. The χ^2 fit typically gives two to three times bigger errors when using simulated measurements based on 10 fb^{-1} of data; for simulated measurements assuming an integrated luminosity of 500 fb^{-1} the error estimates from the χ^2 fits are as much as a full order of magnitude larger than those of the ANNs.

We therefore conclude that a χ^2 fit yielding reliable results for both the central values and the statistical errors of the CMSSM parameters would need significantly *more* computational effort than the ANNs, in order to reduce the Monte Carlo errors on the predictions to a level that can be tolerated by this method. Moreover, while the error estimates we presented above are not yet reliable, they do suggest that the precision of the parameter estimates from χ^2 fitting will be significantly worse than that of the ANNs. We did therefore not attempt to further optimize the χ^2 fits, and instead present the results for the ANNs for all four benchmark points.

IV. RESULTS

Recall that we constructed a separate ANN, with a single output neuron, for each CMSSM parameter. Moreover, as described in Sec. III A 5, we generated different training and control sets for each benchmark point. Altogether we thus ended up constructing $4 \times 4 = 16$ different ANNs.

Some details of these 16 ANNs are collected in Table III. We need between 15 and 25 hidden neurons to construct ANNs that produce an approximately Gaussian distribution of output values when fed with a Gaussian distribution of inputs. Here we already note a first difference among the ANNs designed for different CMSSM parameters: the ones for m_0 and $m_{1/2}$ usually need slightly fewer hidden neurons than the ones for A_0 and $\tan\beta$.

Table III also shows that the control error reaches its minimum after typically hundreds of learning steps. We continued the training usually for about 100 learning steps after each minimum, in order to check that the absolute

TABLE III. Overview of the 16 ANNs we constructed: number of hidden neurons, number of learning steps and final control errors. These results are based on simulated measurements for an integrated luminosity of 10 fb^{-1}. The control error is normalized as in Eq. (3.7).

		Number of hidden neurons	Number of learning steps	Control error
Reference point 1	m_0	20	1009	0.101
	$m_{1/2}$	20	407	0.081
	$\tan\beta$	20	653	0.135
	A_0	20	561	0.532
Reference point 2	m_0	20	929	0.123
	$m_{1/2}$	15	452	0.116
	$\tan\beta$	20	499	0.387
	A_0	25	155	0.711
Reference point 3	m_0	15	244	0.106
	$m_{1/2}$	20	107	0.136
	$\tan\beta$	20	240	0.681
	A_0	15	763	0.405
Reference point 4	m_0	22	488	0.156
	$m_{1/2}$	22	637	0.067
	$\tan\beta$	25	511	0.178
	A_0	25	391	0.310

minimum has indeed been found. Moreover, as shown in the Appendix, each learning step of an ANN with υ hidden neurons entails the calculation of a matrix with $86 \cdot \upsilon + 1$ rows and columns; the computation of each element of this matrix requires a sum over all $\sim 100,000$ members of the training set for this ANN. As a result, the training of a single ANN can take up to 2000 CPU hours.[22]

The last column of Table III gives the normalized control error, as defined in Eq. (3.7). The size of this error should determine the accuracy with which a given CMSSM parameter can be determined by its ANN. We see that this error is significantly smaller for m_0 and $m_{1/2}$ than for A_0 and $\tan\beta$. This is not surprising, since the former parameters affect the spectrum of superparticles, in particular the spectrum of strongly interacting superparticles, more than the latter do. We also see that the normalized control error is usually somewhat smaller for the gaugino mass $m_{1/2}$ than for scalar mass m_0; the one exception is benchmark point 3, where m_0 is large enough to significantly affect the squark masses (in contrast to benchmark point 1, where m_0 essentially only affects the slepton masses), but not so large that the cross section for the production of final states involving one or two squarks is much smaller than that for gluino pair production (in contrast to benchmark point 2). Note also that the normalized control error for $m_{1/2}$ is smaller for points 1 and 4, where charginos and neutralinos have competing two-body decays into sleptons and into gauge or Higgs bosons, than for points 2 and 3, where charginos and neutralinos cannot decay into sleptons. Sizable branching ratios into sleptons increase the fraction of events containing several charged electrons or muons, and/or increase the average number of reconstructed τ's in the final state. These branching ratios therefore affect several of our observables.

The normalized control error for $\tan\beta$ differs quite significantly among the four benchmark points. It is smallest for point 1, where the kinematics and branching fractions for two-body decays of gluinos into $\tilde{b}$ squarks, and of neutralinos and charginos into $\tilde{\tau}$ sleptons, depend on $\tan\beta$. In contrast, in point 2 gluinos only have three-body decays, and the neutralinos and charginos cannot decay into sleptons, leading to a greatly reduced sensitivity to $\tan\beta$. In point 3, all gluinos decay into $\tilde{t}_1$, whose mass and branching ratios depend only weakly on $\tan\beta$ unless the latter is very large. Also, neutralinos and charginos again cannot decay into sleptons here. As a result, this point has the largest normalized control error on $\tan\beta$. Finally, point 4 has a rather large input value of $\tan\beta = 30$. As a result, terms proportional to $m_b \tan\beta$ and $m_\tau \tan\beta$ in the RG equations as well as sfermion mass matrices are quite sizable, yielding significant effects in some of our observables. The normalized control error on $\tan\beta$ is therefore again relatively small for this point.

The ANNs for A_0 typically have the largest normalized control errors. At the tree level, A_0 practically only affects the $\tilde{t}$ sector; $L-R$ mixing in the $\tilde{b}$ and $\tilde{\tau}$ sectors is dominated by a term $\propto \mu \tan\beta$, which is much larger than the relevant (weak-scale) A parameter for all our benchmark points. A_0 also affects third generation sfermion and Higgs boson masses via the RG running, and hence also the value of $|\mu|$ which is determined from the requirement that the mass of the Z boson comes out correctly. From dimensional analysis and the structure of the RG equations, any weak-scale soft breaking mass can be written as

$$m_i^2 = a_i m_0^2 + b_i m_{1/2}^2 + c_i A_0^2 + d_i m_{1/2} A_0; \quad (4.1)$$

the coefficients depend on the dimensionless couplings in the theory, and hence also on $\tan\beta$. A very similar equation therefore also holds for the derived weak-scale value of μ^2.[23] Benchmark points 1, 2 and 4 have $A_0 = 0$, so the sensitivity to A_0 via the quadratic coefficients c_i is small. Moreover, the d_i are typically significantly smaller in magnitude than $\max\{|a_i|, |b_i|\}$, further reducing the sensitivity to A_0. On the other hand, benchmark point 3 does have a large A_0; moreover, we saw that this point has an especially large normalized control error for $\tan\beta$. As a result, benchmark point 3 actually has a smaller normalized control error for A_0 than for $\tan\beta$. Altogether, the biggest error for A_0 (0.711, for benchmark point 2) is over ten times larger than the smallest error for $m_{1/2}$ (0.067, for benchmark point 4).

In fact, the entries of Table III underestimate the differences between the relative size of the errors on the CMSSM parameters. The reason is that the control error is normalized, as shown in Eq. (3.7); it thus scales inversely with the size of the region covered by the respective training set. We saw in Sec. III A 5 that these sets only cover a range of 80 GeV in $m_{1/2}$ and 250 to 350 GeV in m_0, but much of the theoretically allowed parameter space for $\tan\beta$ and A_0. We thus expect the final errors on the former two parameters to be much smaller than those on the latter two.

This is confirmed by Table IV, where we list the values of the CMSSM parameters reconstructed from our ANNs fed with the 10 fb^{-1} and 500 fb^{-1} simulated measurements at the four benchmark points. The standard deviations and correlation coefficients are calculated with the propagation of uncertainty method as described at the end of Sec. III A 7. If we divide the final estimate of the standard deviation by the product of the normalized control error of Table III and the size of the parameter region

[22] In particular for the ANNs with high numbers of completed learning steps, the last hundreds of learning steps lead only to relatively small improvements of the control errors and calculated standard deviations. Therefore, for those ANNs nearly as good results can be reached with much less computational effort.

[23] There is also a contribution $\propto M_Z^2$, which is however much smaller than the term $\propto m_{1/2}^2$ in the CMSSM.

TABLE IV. Values of the CMSSM parameters reconstructed by our ANNs for the four benchmark points, their uncertainties, and the corresponding correlation coefficients. The standard deviations and correlation coefficients are calculated via error propagation; see Eqs. (3.11) to (3.13). The dimensionful parameters m_0, $m_{1/2}$ and A_0 are given in GeV.

	Point 1	Point 2	Point 3	Point 4
			10/fb	
$\bar{m}_0 \pm \sigma_{m_0}$	167.25 ± 33.78	1998.93 ± 92.20	1055.97 ± 47.26	482.94 ± 61.23
$\bar{m}_{1/2} \pm \sigma_{m_{1/2}}$	697.51 ± 7.36	446.55 ± 11.30	607.47 ± 11.53	695.48 ± 7.87
$\tan\bar{\beta} \pm \sigma_{\tan\beta}$	21.35 ± 5.96	9.67 ± 18.81	23.41 ± 37.42	23.37 ± 11.08
$\bar{A}_0 \pm \sigma_{A_0}$	463.43 ± 326.00	1406.37 ± 2898.67	1453.49 ± 1891.58	-73.52 ± 628.16
$\rho_{m_0 m_{1/2}}$	-0.037	-0.051	-0.309	-0.407
$\rho_{m_0 \tan\beta}$	0.131	-0.068	0.044	0.057
$\rho_{m_0 A_0}$	0.021	-0.251	0.200	-0.671
$\rho_{m_{1/2}\tan\beta}$	0.201	0.174	-0.084	-0.350
$\rho_{m_{1/2}A_0}$	0.121	-0.119	0.065	0.139
$\rho_{\tan\beta A_0}$	-0.035	0.249	-0.218	0.238
			500/fb	
$\bar{m}_0 \pm \sigma_{m_0}$	156.23 ± 4.86	2004.05 ± 10.61	1015.66 ± 4.49	391.86 ± 7.70
$\bar{m}_{1/2} \pm \sigma_{m_{1/2}}$	701.05 ± 0.87	451.15 ± 1.00	598.75 ± 1.21	700.71 ± 0.88
$\tan\bar{\beta} \pm \sigma_{\tan\beta}$	9.39 ± 0.70	14.66 ± 2.17	20.79 ± 5.20	30.50 ± 1.24
$\bar{A}_0 \pm \sigma_{A_0}$	-43.61 ± 55.37	261.47 ± 474.68	774.39 ± 295.63	183.05 ± 101.48
$\rho_{m_0 m_{1/2}}$	-0.125	-0.519	-0.550	-0.695
$\rho_{m_0 \tan\beta}$	-0.233	-0.131	0.294	0.441
$\rho_{m_0 A_0}$	0.038	-0.068	-0.557	-0.232
$\rho_{m_{1/2}\tan\beta}$	0.259	0.076	-0.110	-0.416
$\rho_{m_{1/2}A_0}$	0.033	0.165	0.509	0.250
$\rho_{\tan\beta A_0}$	-0.211	-0.158	-0.055	0.464

spanned by the training sets, we obtain values that cluster around 1.2 ± 0.4 (0.15 ± 0.04) for the simulated measurements with 10 fb^{-1} (500 fb^{-1}) of data. In other words, the final estimate of the statistical uncertainty is approximately proportional to this product. The main exceptions to this rule occur for benchmark point 2, where the estimated uncertainties for m_0 (A_0) are nearly two times bigger than (less than half as big as) the value obtained from this simple scaling. A strict scaling is not expected, since the normalized control error measures the *average* performance of the ANN against 300 CMSSM scenarios in the control set, whereas we are now considering specific benchmark points.

Moreover, the control error determines the deviation from the true value, not the estimated size of the uncertainty of the extracted CMSSM parameters. However, Table IV also shows that the estimated standard deviations reflect the differences to the true values quite well. This is true in particular for the simulated measurement based on 10 fb^{-1} of data, where 12 of the 16 estimated parameter values are less than one estimated standard deviation away from the true value, and the remaining four estimates differ from the true values by less than two estimated standard deviations.

Another indication for the reliability of the estimated standard deviations is that they decrease approximately by the expected factor $\sqrt{50} \simeq 7$ when going to the simulated measurement based on 500 fb^{-1} of data. Recall that we only include statistical uncertainties, which should of course decrease proportional to the inverse square root of the accumulated luminosity.

However, upon closer examination some systematic deviation from the expected reduction of the estimated uncertainties by a factor of $\sqrt{50}$ becomes apparent. First, when averaging over all four benchmark points the estimated errors on $m_{1/2}$, m_0 and $\tan\beta$ actually decrease by factors of 9.6, 8.5 and 8.3. This indicates that the uncertainties for the 500 fb^{-1} measurements might be somewhat underestimated. In fact, Table IV shows that in this case 10 out of 16 reconstructed parameter values are more than one estimated standard deviation away from their true values; four of the reconstructed values are more than two standard deviations off, and one ANN output (m_0 for benchmark point 3) differs from its true value by more than three estimated standard deviations. The likely explanation for this is that the predictions in the training and control sets were also "only" based on 500 fb^{-1} of simulated data; i.e. they had the same uncertainty due to finite Monte Carlo statistics as our simulated measurements. This "theoretical uncertainty" is not included in our estimates of the uncertainty of the ANN outputs. Statistical fluctuations in the training and/or control sets might also lead to systematic offsets of the ANN outputs relative to the true values. This might explain the relatively poor performance of the ANNs for benchmark point 3 when fed the simulated 500 fb^{-1} measurement, where all four output values are more than

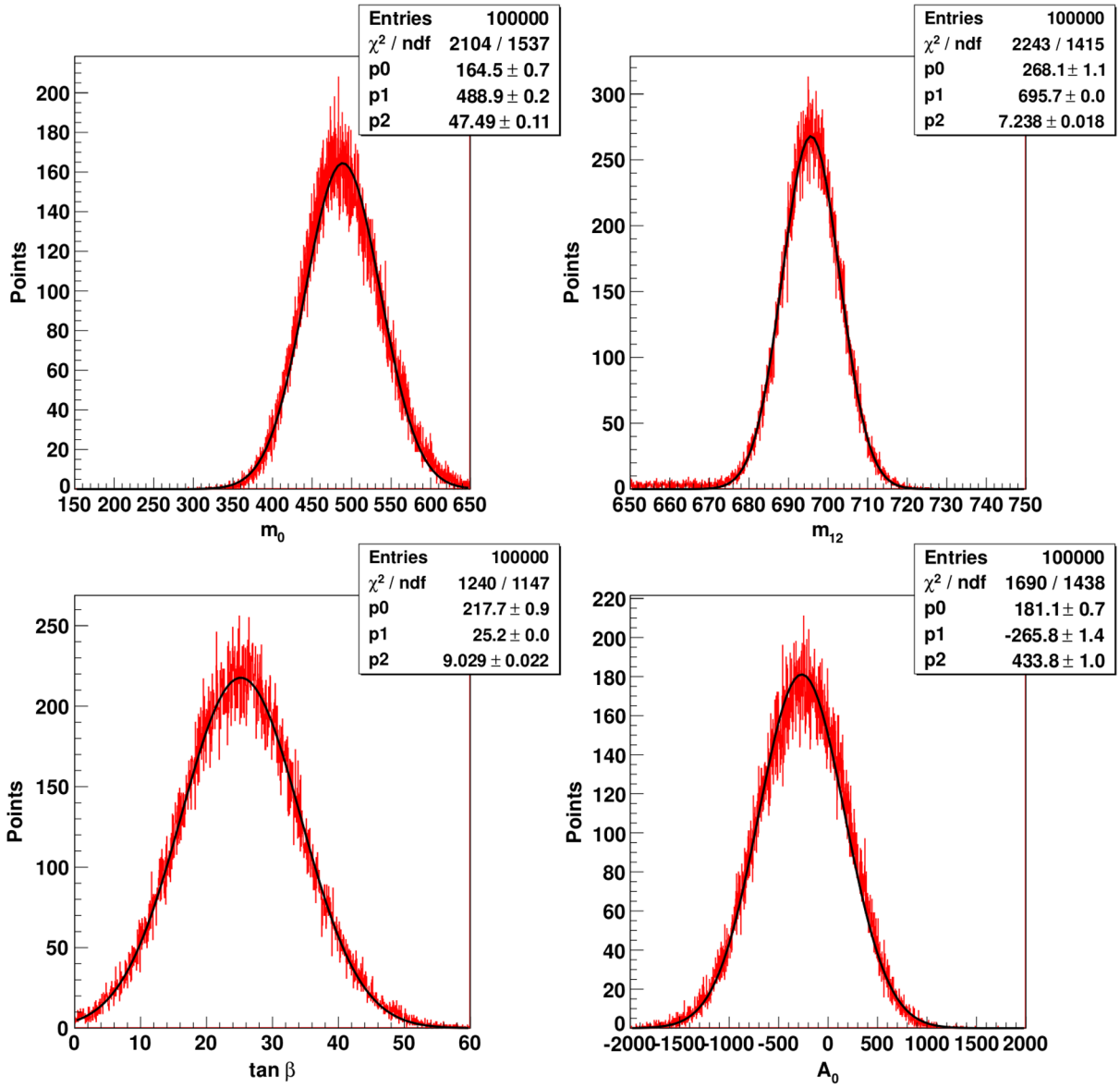

FIG. 8 (color online). The histograms show one-dimensional distributions of ANN outputs for benchmark point 4, obtained by feeding the neural networks with 100,000 Gaussian distributed versions of the 10 fb^{-1} measurement. The dimensionful parameters m_0, $m_{1/2}$ and A_0 are given in GeV. The distributions are fitted to Gaussians (solid curves) of the form $g(x) = p_0 \cdot \exp(-1/2[(x - p_1)/p_2]^2)$ with x being the appropriate CMSSM parameter, p_1 the mean value and p_2 the standard deviation. The input values are in the center of the x axis of the respective plot. The neural network settings are given in Table III.

one estimated standard deviation off, and three of the four values are more than two estimated standard deviations off. In order to check this, we simulated the 500 fb^{-1} measurement of benchmark point 3 two more times with different seeds in Herwig + + . For both additional versions the estimated value of m_0 was less than one standard deviation away from the true value. Therefore, the more than three standard deviations in Table IV seems to be a rather extreme statistical fluctuation. On the other hand, for all three versions of the measurement, A_0 differed between two and three standard deviations from the true value. The estimated value was always smaller than the true value $A_0 = 1500$ GeV. This slight tendency to lower values might originate from the fact that the true value lies near the upper bound $A_{0,\text{max}} = 2m_0$ enforced in the selection of the training sets.

In contrast, the error estimates for the four values of A_0 only decrease by a factor of ~6.2. For the reasons listed above, the errors for the 500 fb^{-1} measurements might still be slightly underestimated. However, in the case at hand the errors for the 10 fb^{-1} measurements are also somewhat suspect. The reason is that in three cases the estimated "1σ" interval for A_0 extends beyond the range covered in the training and control sets, which satisfy

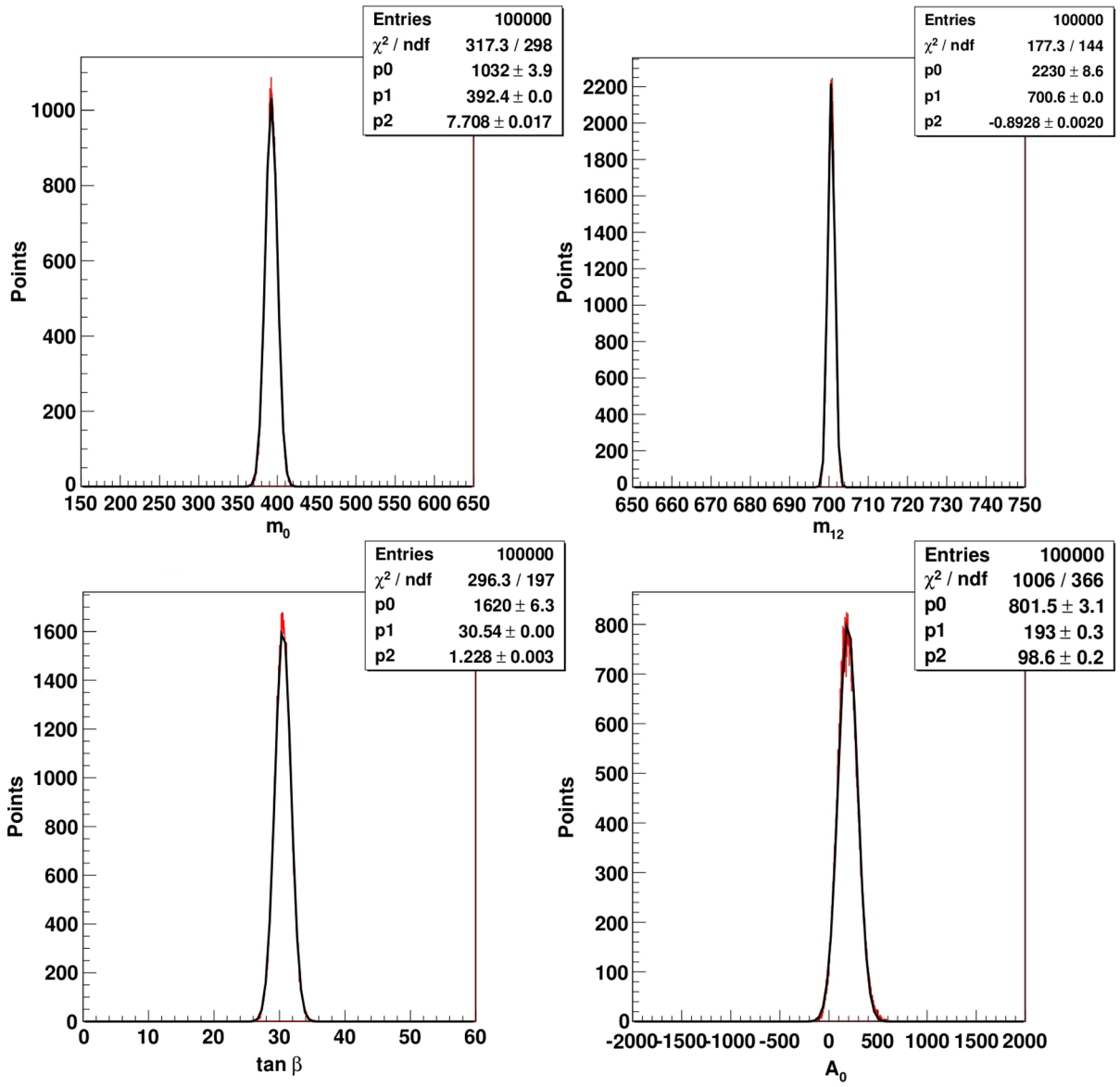

FIG. 9 (color online). As in Fig. 8, but using 500 fb^{-1} instead of a 10 fb^{-1} of simulated data.

$|A_0| \leq 2m_0$. This means that the ANNs are forced to at least partially extrapolate, rather than interpolate, when estimating these errors. Recall also that values of $|A_0|$ significantly larger than $2m_0$ often lead to problems with the calculation of the spectrum.

In spite of these caveats, we consider the overall performance of our ANNs to be quite satisfactory. Already with 10 fb^{-1} of simulated data the gaugino mass parameter $m_{1/2}$ can be determined with a relative accuracy of 1% to 2.5%. If m_0 is large enough to significantly affect the squark masses (benchmark points 2 and 3), it can be determined with a relative accuracy of about 4.5%; if m_0 is very small, as in point 1, this relative accuracy deteriorates.

Meaningful determinations of A_0 and $\tan\beta$ will need more data. For the simulated measurements with 500 fb^{-1}, the estimated uncertainty on $\tan\beta$ varies between 4% for point 4, which had a large input value of $\tan\beta$, and 25% for point 3, which, as we have seen before, has the largest normalized control error for this quantity. The estimated error on A_0 is roughly 25% to 35% of the input value of m_0.

Finally, Table IV also lists the correlation coefficients. We see that most correlations are quite weak. The true correlation coefficients should be independent of the luminosity, but our estimates of these coefficients should, and do, fluctuate when the (simulated) data set is increased.

We nevertheless observe consistently negative correlations between the extracted values of m_0 and $m_{1/2}$ for benchmarks points 2, 3 and 4. This can be explained from the observation that increasing either $m_{1/2}$ or m_0 will increase the masses of strongly interacting superparticles, which will lead to a reduction of the total event rate, and

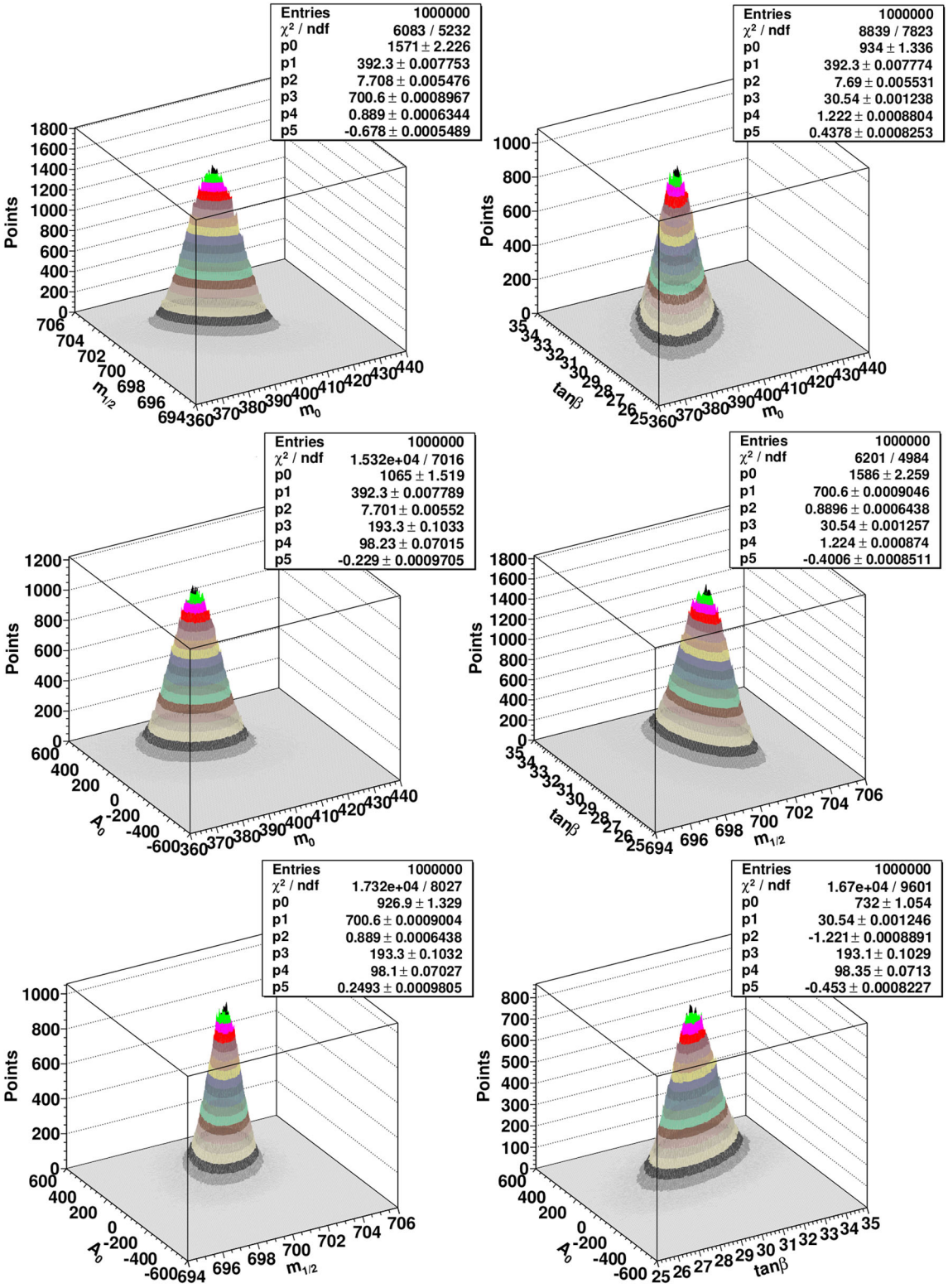

FIG. 10 (color online). Two-dimensional distributions of all pairs of CMSSM parameters as predicted from our ANNs for benchmark point 4, when fed with 1,000,000 Gaussian distributed versions of the original simulated measurement, assuming an integrated luminosity of 500 fb^{-1}. The dimensionful quantities m_0, $m_{1/2}$ and A_0 are given in GeV. These distributions are fitted to Gaussians of the form $g(x, y) = p_0 \cdot \exp[-0.5/(1 - p_5^2) \cdot ([(x - p_1)/p_2]^2 + [(y - p_3)/p_4]^2 - 2 \cdot p_5/(p_2 p_4) \cdot (x - p_1) \cdot (y - p_3))]$, with x and y being the appropriate CMSSM parameters, p_1 and p_2 the mean value and standard deviation of x, p_3 and p_4 the mean value and standard deviation of y, and p_5 the correlation coefficient of Eq. (3.9).

to an increase of the average H_T values. To some extent an increase in m_0 can therefore be compensated by a reduction of $m_{1/2}$, and vice versa. This correlation is essentially absent for benchmark point 1, which has $m_0^2 \ll m_{1/2}^2$ so that even the squark masses are essentially independent of m_0. Similarly, the mild positive correlation between A_0 and $m_{1/2}$ can be explained from the observation that the coefficients d_i in Eq. (4.1) are negative, while b_i are positive. The RG effect on the scalar masses of increasing A_0 can therefore be compensated by increasing $m_{1/2}$.

As already noted, the standard deviations and correlation coefficients listed in Table IV have been computed using error propagation. We conclude this section by comparing these with the results obtained by feeding Gaussian distributed variants of the original measurements into our ANNs, as described in the first part of Sec. III A 7. We do this for benchmark point 4; results for the other benchmark points are similar.[24]

Figure 8 shows the distribution of the output of the four ANNs for the simulated measurement with 10 fb^{-1} of data. We note first of all that the binned distributions of ANN outputs do indeed look rather Gaussian already for this smaller data sample—much more so than the corresponding distributions obtained via χ^2 minimization; see Fig. 5. Moreover, in the case of m_0, $m_{1/2}$ and $\tan\beta$ both the mean values of the Gaussians (the p_1 values given in the inserts in the figure), which are the final parameter estimates of the ANN derived in this manner, and their widths (the p_2 values), which are the final estimates for the uncertainty of these parameters, agree quite well with the results listed in Table IV. In the case of the mean values, the two methods yield estimates that agree to about 0.1 estimated standard deviations. The two estimates for the standard deviations differ slightly more. This is not unexpected, since the standard deviation estimated via error propagation itself has an error, which is not negligible for this small (simulated) data set. Recall also that we set all observables to zero that have been obtained from fewer than 10 events. Benchmark point 4 has 7 events in class 9 and 13 events in class 4. Using the method of error propagation, the observables $O_{9,i}$ are thus ignored, while $O_{4,i}$ are included. However, quite a few of the Gaussian distributed variants of this measurement will have ≥ 10 events in class 9 and/or ≤ 9 events in class 4. These variants will thus feed *qualitatively* different input in the ANNs than the original simulated measurement.

Both the statistical error of the estimated standard deviation, and the systematic difference among the ANN inputs from different variants of the same simulated measurement, are expected to decrease with increasing luminosity. For example, for an integrated luminosity of 500 fb^{-1}, benchmark point 4 has 350 events in class 9, which is about eight standard deviations away from the lower bound of 500 events now required for the inclusion of observables $O_{9,i}$; we can therefore be quite certain that none of the up to 1,000,000 variants of the simulated measurements has sufficiently many events of class 9 for the $O_{9,i}$ to be included. We therefore expect the differences between the two methods for determining the final ANN output, and its estimated uncertainty, to agree better for higher luminosity. This is confirmed by Fig. 9. In particular, the two estimates of the uncertainties now agree to better than 3% in all cases.

Recall that we used 100 Gaussian distributed variants of ANN inputs for each CMSSM training parameter set. Without this trick, the distributions of the ANN outputs for $\tan\beta$ and A_0 would look much less Gaussian, in particular for the smaller luminosity of 10 fb^{-1}. Moreover, the estimated uncertainties on the extracted CMSSM parameters would be $\sim$20% larger. The ANNs therefore clearly profit from the information on the relative errors of our observables. Recall that these additional training sets were obtained without additional event generation. On the other hand, the time needed for training the ANNs scales linearly with the size of the training sets.

Finally, Fig. 10 shows that Gaussian fits to two-dimensional distributions of ANN outputs yield almost the same estimates of the standard deviations as the one-dimensional fits or the method of error propagation. These two-dimensional fits also allow us to determine the correlation coefficients (the p_5 values given in the inserts in the figures). Again we observe quite close agreement with the results derived from Gaussian error propagation.

Above we found some evidence that the errors from the 500 fb^{-1} measurements are underestimated, since they do not include the ''theory'' error from the finite MC statistics used in the training and control sets. Since Figs. 9 and 10 have been obtained with fixed ANNs, only varying the input (i.e., the results of the measurements), they do not reflect this theory error, either. It is nevertheless reassuring that two methods which are computationally independent yield very similar results.

V. SUMMARY AND CONCLUSIONS

In this paper we investigated methods to determine the values of underlying parameters from (simulated) measurements at the LHC, with heavy emphasis on counting observables. Mostly for reasons of computational simplicity, we did this for the CMSSM where only four free parameters need to be determined; the sign of μ was fixed to be positive.

Evidently the choice of observables is crucial. Here we used the same observables as in Ref. [14], where we have shown that they perform well when trying to distinguish different (generalized) MSSM parameter sets using a χ^2 criterion. We also used the same cuts as in Ref. [14], even though the sparticle mass scale of the benchmark points we

[24]Some results for benchmark point 3 have already been shown in Sec. III A 7.

used for the present analysis, which lie just outside the currently excluded region, is significantly higher than in our previous study. Moreover, in this proof-of-concept analysis we ignored Standard Model backgrounds as well as statistical uncertainties.

Our main result is that artificial neural networks (ANNs) can be used for the determination of model parameters, including statistically reliable estimates of their errors. In contrast, an in principle straightforward χ^2 minimization did not yield reliable results, probably because we do not have the computational resources required for the calculation of theoretical predictions for sufficiently many different sets of model parameters with sufficiently small Monte Carlo uncertainty. Moreover, the estimated errors on m_0 and $m_{1/2}$ from the χ^2 minimization were about two to three times larger than those obtained from the ANNs. We thus conclude that ANNs can yield better and more reliable results with less computational effort than a χ^2 minimization.

Of course, the training of the ANNs was also affected by the finite MC statistics used for deriving our theoretical predictions. However, here the main requirement is that the error due to finite MC statistics should be (much) smaller than the statistical error of the (actual or simulated) measurement. In contrast, any "smart" algorithm attempting to minimize a χ^2 will need reliable information on systematic changes of χ^2 when the CMSSM parameters are changed by relatively small amounts. The uncertainty on χ^2 due to MC statistics therefore needs to be (much) smaller than the (typically quite small) change of χ^2 induced by this small variation of the parameters.

Mathematically, an ANN is a function mapping (typically a rather large number of) input values into one or more output value(s). In the case at hand, the inputs are the 84 observables described in Sec. II A. These observables have quite different statistical uncertainties. This should affect the weights given to these inputs. We took this into account in two ways. First, we simply set all observables to zero that have been obtained from fewer than one event per fb^{-1} of (simulated) data. Since we assume an integrated luminosity of at least 10 fb^{-1}, this effectively removes very "noisy" observables.

Second, for each training set of CMSSM parameters, we generated 100 variants where the observables were drawn from multidimensional Gaussians, whose central values and covariances were taken from the original simulation. For each set of CMSSM parameters that we simulated for the training sets, the ANNs were therefore confronted with 100 slightly different sets of inputs (observables) yielding the same outputs (CMSSM parameters). The ANNs could thus learn the relative accuracy between the various observables, which, at least for Gaussian statistics, should be independent of the integrated luminosity. The trained ANNs can thus be used on (simulated) data sets of any luminosity, as long as the statistical uncertainty of this measurement is (much) larger than the Monte Carlo uncertainty of the predictions in the training and control sets.

Since the training of the ANNs is independent of actual data, it can be done before the measurement. Once actual measurements exist, the ANN results for the CMSSM parameters can then be obtained with negligible computational effort. In contrast, a χ^2 minimization has to be (re)done for each measurement.

As expected the CMSSM parameters m_0 (scalar mass parameter) and $m_{1/2}$ (gaugino mass parameter) could be determined relatively well for all four benchmark points already with an integrated luminosity of 10 fb^{-1}. In the best cases these could be determined to 4.5% for m_0 and 1% for $m_{1/2}$. With this luminosity, leading to around 1000 events after cuts, $\tan\beta$ and A_0 could at best be determined very roughly.

On the other hand, with a luminosity of 500 fb^{-1}, $m_{1/2}$ could be determined with statistical uncertainty well below 1%. The statistical error on m_0 then amounts to 5 to 10 GeV for our four benchmark points. For three of the points, $\tan\beta$ could be determined with an error of ± 2 or better. Finally, the error on A_0 was about 25% to 35% of the input value of m_0. We also computed the full covariance matrix, and found that most correlations are quite weak.

These results were obtained with two different methods. In one method, the central values were obtained by simply feeding the simulated measurements into the trained ANNs, and the uncertainties and correlation coefficients were computed using Gaussian error propagation. Alternatively we generated numerous Gaussian distributed variants of the original simulated measurements, and fitted Gaussian distributions to the outputs of the ANNs. The results of these two methods agreed quite well.

One disadvantage of using ANNs is that they do not automatically give a measure for the goodness of the fit (at most only an indication by the shapes of the Gaussian output distributions): even if nature is described by a completely different theory, the ANNs will output some values of the free parameters of the (wrong) theory on which they have been trained when confronted with actual data. One will have to simulate the assumed theory with the values of the free parameters determined by the ANNs in order to determine the quality of the fit, e.g. by computing the χ^2. However, the numerical effort required for this is trivial compared to the effort required for the determination of the values of the free parameters.

Our results can be improved in a number of ways. First of all, we ignored all information on the Higgs sector. In the context of the MSSM, knowledge of the mass of one of the CP-even neutral Higgs bosons will greatly reduce the allowed parameter space. One can also try to tag top quarks [25] or Higgs bosons [26] in the final state using subjet techniques, or to devise dedicated sets of cuts that attempt to isolate specific decay chains. Moreover, kinematic

features (edges or kinks) could be included. All this would increase the number of inputs fed into the ANNs, and could thus increase their ability to determine the underlying parameters. Conversely it might be possible to remove some of the observables from the list of input parameters without significant loss of information. Since our algorithm should automatically assign low weights to observables with little discriminating power, this would presumably not improve the performance of the ANNs very much, but it could reduce the computational effort. Similarly, the number of Gaussian distributed variants generated for each training set of CMSSM parameters could perhaps be reduced without degrading the performance; the time needed to train the ANNs is essentially proportional to this number. Finally, we did not consider ANNs with two (or more) layers of hidden neurons; this more complicated architecture might allow us to reduce the total number of hidden neurons, and perhaps also the total number of weights that need to be determined, which would speed up the training process.

However, before trying to further optimize the performance of the ANNs one should make the setup more realistic, by including Standard Model backgrounds as well as systematic uncertainties. The cuts could then be optimized for each benchmark point separately, as e.g. done in [5,27]. This would not increase the computational effort, since we already use different ANNs for the different benchmark points (or regions). Systematic uncertainties could be introduced as in [14]. Moreover, if the method is applied to supersymmetric scenarios, one should consider benchmark points that have a Higgs boson of the correct mass and coupling, in agreement with recent data [15]. In the framework of the CMSSM this is known to push the squark mass scale to quite large values [28]; this will presumably greatly reduce the possibility to extract m_0. However, if scalar masses are not required to unify, first and second generation scalars could still have masses similar to those in our benchmark scenarios. Alternatively, one could introduce additional Higgs superfields to increase the mass of the lightest CP-even Higgs boson, as e.g. in the NMSSM [29].

The purpose of this paper was to show that artificial neural networks do have the potential to determine quantitatively the values of the parameters of the underlying theory, and the corresponding (statistical) uncertainties. Further, more realistic studies are thus well worth the effort.

ACKNOWLEDGMENTS

N. B. wishes to thank the "Bonn-Cologne Graduate School of Physics and Astronomy" for financial support. This work was partially supported by the BMBF-Theorieverbund and by the Helmholtz Alliance "Physics at the Terascale."

APPENDIX: CALCULATION OF WEIGHTS

In this appendix we discuss the calculation of the weights of a neural network with $84 + 1$ input neurons, $\upsilon + 1$ hidden neurons and one output neuron; this describes the ANNs we constructed in Sec. III A. Since there is only one output neuron, we suppress the index $r = 1$ on the corresponding weights in Eq. (3.2).

We begin by combining all weights in one vector $\vec{w}$. The first $85 \cdot \upsilon$ entries are weights in the first weight layer, and the remaining $\upsilon + 1$ entries are from the second weight layer:

$$\begin{aligned}\vec{w} &= (w_1, \ldots, w_W)^T = (w_1, \ldots, w_{\upsilon\cdot 85}, w_{\upsilon\cdot 85+1}, \ldots, w_W)^T \\ &= (w^{(1)}_{10}, w^{(1)}_{11}, \ldots, w^{(1)}_{184}, w^{(1)}_{20}, \ldots, w^{(1)}_{\upsilon 84}, w^{(2)}_0, \ldots, w^{(2)}_\upsilon)^T.\end{aligned} \tag{A1}$$

In total the weight vector thus has $W = 85 \cdot \upsilon + \upsilon + 1 = 86 \cdot \upsilon + 1$ entries. The $w_m, m = 1, \ldots, 85 \cdot \upsilon$ are related to the $w^{(1)}_{ai}, a = 1, \ldots, \upsilon, i = 0, \ldots, 84$ via $w_m = w^{(1)}_{ai}$ with $a = \lceil m/85 \rceil$[25] and $i = [(m-1) \bmod 85]$. Similarly, for $n = 85 \cdot \upsilon + 1, \ldots W$ we have $w_n = w^{(2)}_b$ with $b = (n - 1 - \upsilon \cdot 85)$.

During the training of the ANN the weight vector is adjusted iteratively. We denote the weight vector in learning step t by $\vec{w}_t$.

1. First step

As mentioned in Sec. III A 3 the first weight vector $\vec{w}_1$ is chosen randomly from a Gaussian distribution. In the first improvement, $\vec{w}$ is changed in direction $\vec{s}_1 = -\vec{g}_1$ equal to the negative gradient $\vec{g}$. The first gradient vector $\vec{g}_1 = \vec{g}(\vec{w}_1) = \vec{\nabla} F(\vec{w}_1)$ is calculated using the function F describing the error the ANN makes in reproducing the training set. We wish to minimize this function of the weights. Since the location of the minimum is independent of the normalization, we use the simple definition

$$F = \sum_{\ell=1}^{N} F^\ell = \sum_{\ell=1}^{N} \frac{1}{2} (y^\ell(\vec{x}^\ell) - k^\ell)^2. \tag{A2}$$

As in Sec. III A N is the number of training sets; in our applications, $N \simeq 100,000$. Again as in Sec. III A, $y^\ell(\vec{x}^\ell)$ is the output that the ANN computes from the normalized input $\vec{x}^\ell$ of training set ℓ, while the correct (inversely) normalized CMSSM output of the training set is labeled k^ℓ; the normalization of the input and output has been described in Sec. III A 2. The first $85 \cdot \upsilon$ entries of the gradient vector can be computed from Eqs. (A2), (3.1), and (3.2):

$$\vec{\nabla}_m F(\vec{w}) = \frac{\partial F}{\partial w_m} = \frac{\partial F}{\partial w^{(1)}_{ai}} = \sum_{\ell=1}^{N} \delta^{(1)\ell}_a x^\ell_i. \tag{A3}$$

[25] $\lceil x \rceil$ means the smallest integer that is larger than or equal to the real number x; e.g. $\lceil 1/6 \rceil = 1 = \lceil 1 \rceil$.

The remaining $\upsilon + 1$ entries are

$$\vec{\nabla}_n F(\vec{w}) = \frac{\partial F}{\partial w_n} = \frac{\partial F}{\partial w_b^{(2)}} = \sum_{\ell=1}^{N} \delta^{(2)\ell} h(z_b^\ell). \quad \text{(A4)}$$

Here, h is the hidden neuron processing function, $h(z_b^\ell) = \tanh(z_b^\ell)$, and $z_b^\ell = \sum_{i=0}^{84} w_{bi}^{(1)} x_i^\ell$ as in Eq. (3.1). Moreover, $\delta_a^{(1)\ell}$ in Eq. (A3), with $a = 1, \ldots, \upsilon$, and $\delta^{(2)\ell}$ in Eq. (A4) are abbreviations for

$$\delta_a^{(1)\ell} = \frac{\partial F^\ell}{\partial y^\ell} \frac{\partial y^\ell}{\partial z_a^\ell} = \delta^{(2)\ell} w_a^{(2)} \cdot h'(z_a^\ell), \quad \text{(A5)}$$

where h' stands for the derivative of h, i.e. $h'(z_b^\ell) = (1 - \tanh^2(z_b^\ell))$, and

$$\delta^{(2)\ell} = \frac{\partial F^\ell}{\partial y^\ell} = y^\ell - k^\ell. \quad \text{(A6)}$$

2. Repeated steps

The following steps of the calculation are repeated until one is satisfied that the (global) minimum of the normalized control error (3.7) has been reached. The new weight vector is calculated in step $t \geq 2$ from the expression

$$\vec{w}_t = \vec{w}_{t-1} + \alpha_t \vec{s}_{t-1}. \quad \text{(A7)}$$

The calculation of $\vec{s}_1 = -\vec{g}_1 = -\vec{\nabla} F(\vec{w}_1)$ has already been described in Eqs. (A3) to (A6). The coefficient α_t can be computed from the Hessian matrix H_{t-1}:

$$\alpha_t = -\frac{\vec{s}_{t-1}^T \vec{g}_{t-1}}{\vec{s}_{t-1}^T H_{t-1} \vec{s}_{t-1}}. \quad \text{(A8)}$$

The explicit calculation of the Hessian matrix is shown at the end of this appendix. With the new weights the normalized control error of Eq. (3.7) can be calculated and depending on the stopping criterion the learning process might be terminated.

If the stopping criterion is not fulfilled the new gradient vector $\vec{g}_t$ would be calculated, just as in Eqs. (A3) to (A6). Next, the new search direction is computed from

$$\vec{s}_t = -\vec{g}_t + \beta_t \vec{s}_{t-1}, \quad \text{(A9)}$$

where the coefficient β_t is also calculated from the Hessian matrix:

$$\beta_t = \frac{\vec{g}_t^T H_{t-1} \vec{s}_{t-1}}{\vec{s}_{t-1}^T H_{t-1} \vec{s}_{t-1}}. \quad \text{(A10)}$$

Now the next step $t \longrightarrow t + 1$ can be taken, starting with the calculation of the new weights from Eq. (A7).

3. Hessian matrix

The main numerical effort in the training process is the repeated calculation of the Hessian matrix. This is a symmetric $W \times W$ matrix; i.e. its dimension is determined by the number of hidden neurons. It is given by the matrix of second derivatives of the error function F with respect to the weights:

$$H_{mn} = \frac{\partial^2 F}{\partial w_m \partial w_n}. \quad \text{(A11)}$$

Its numerical value will in general be different for each step during the training process. For our architecture, with one layer of hidden neurons, we can distinguish three different cases: both weights are from the first layer; both weights are from the second layer; or one weight is from the first and the other from the second weight layer:

(1) Both weights are from the first weight layer, i.e. $m, n = 1, \ldots, 85 \cdot \upsilon$:

$$\begin{aligned} H_{mn} &= \sum_{\ell=1}^{N} \frac{\partial^2 F^\ell}{\partial w_{ai}^{(1)} \partial w_{bj}^{(1)}} = \sum_{\ell=1}^{N} \frac{\partial}{\partial w_{ai}^{(1)}} (\delta_b^{(1)\ell} x_j^\ell) \\ &= \sum_{\ell=1}^{N} \frac{\partial}{\partial w_{ai}^{(1)}} [\delta^{(2)\ell} w_b^{(2)} \cdot h'(z_b^\ell)] x_j^\ell \\ &= \sum_{\ell=1}^{N} [w_a^{(2)} w_b^{(2)} h'(z_a^\ell) h'(z_b^\ell) \\ &\quad + \delta_{ba} \delta^{(2)\ell} w_b^{(2)} h''(z_b^\ell)] x_i^\ell x_j^\ell. \end{aligned}$$

Here we have used Eqs. (A2), (A5), (A6), (3.1), and (3.2). The index pairs ai and bj are computed from the indices m and n as described following Eq. (A1) above. The second derivative of the hidden neuron processing function is

$$\begin{aligned} h''(z) &= 2 \tanh(z)(\tanh^2(z) - 1) \\ &= 2h(z)(h^2(z) - 1). \end{aligned} \quad \text{(A12)}$$

(2) Both weights are from the second weight layer, i.e. $m, n = \upsilon \cdot 85 + 1, \ldots, W$:

$$\begin{aligned} H_{mn} &= \sum_{\ell=1}^{N} \frac{\partial^2 F^\ell}{\partial w_a^{(2)} \partial w_b^{(2)}} = \sum_{\ell=1}^{N} \frac{\partial}{\partial w_a^{(2)}} (\delta^{(2)\ell} h(z_b^\ell)) \\ &= \sum_{\ell=1}^{N} h(z_a^\ell) h(z_b^\ell), \end{aligned}$$

where $a = m - 85 \cdot \upsilon - 1$, $b = n - 85 \cdot \upsilon - 1$.

(3) One weight each from the first and the second weight layers, i.e. $m = \upsilon \cdot 85 + 1, \ldots, W$ and $n = 1, \ldots, \upsilon \cdot 85$ or vice versa:

$$\begin{aligned} H_{mn} &= \sum_{\ell=1}^{N} \frac{\partial^2 F^\ell}{\partial w_a^{(2)} \partial w_{bi}^{(1)}} \\ &= \sum_{\ell=1}^{N} \frac{\partial}{\partial w_a^{(2)}} [\delta^{(2)\ell} w_b^{(2)} \cdot h'(z_b^\ell)] x_i^\ell \\ &= \sum_{\ell=1}^{N} (h(z_a^\ell) w_b^{(2)} + \delta_{ba} \delta^{(2)\ell}) h'(z_b^\ell) x_i^\ell. \end{aligned}$$

[1] For introductions to supersymmetry, see M. Drees, R. M. Godbole, and P. Roy, *Theory and Phenomenology of Sparticles* (World Scientific, Singapore, 2004); H. A. Baer and X. R. Tata, *Weak Scale Supersymmetry: From Superfields to Scattering Events* (Cambridge University Press, Cambridge, England, 2006).

[2] S. Sekmen, S. Kraml, J. Lykken, F. Moortgat, S. Padhi, L. Pape, M. Pierini, H. B. Prosper, and M. Spiropulu, J. High Energy Phys. 02 (2012) 075.

[3] H. K. Dreiner, M. Krämer, and J. Tattersall, Europhys. Lett. **99**, 61001 (2012); Phys. Rev. D **87**, 035006 (2013).

[4] G. Aad *et al.* (ATLAS Collaboration), Phys. Rev. D **85**, 112006 (2012); S. Chatrchyan *et al.* (CMS Collaboration), J. High Energy Phys. 10 (2012) 018; G. Aad *et al.* (ATLAS Collaboration), Phys. Rev. D **86**, 092002 (2012); G. Aad*et al.* (ATLAS Collaboration), Eur. Phys. J. C **72**, 2215 (2012); S. Chatrchyan *et al.* (CMS Collaboration), Phys. Rev. D **87**, 052006 (2013); S. Chatrchyan *et al.* (CMS Collaboration), Phys. Rev. Lett. **111**, 081802 (2013); S. Chatrchyan *et al.* (CMS Collaboration), Eur. Phys. J. C **73**, 2404 (2013).

[5] H. Baer, V. Barger, A. Lessa, and X. Tata, Phys. Rev. D **86**, 117701 (2012).

[6] I. Hinchliffe, F. E. Paige, M. D. Shapiro, J. Soderqvist, and W. Yao, Phys. Rev. D **55**, 5520 (1997); H. Bachacou, I. Hinchliffe, and F. E. Paige, Phys. Rev. D **62**, 015009 (2000); I. Hinchliffe and F. E. Paige, Phys. Rev. D **61**, 095011 (2000); B. C. Allanach, C. G. Lester, M. A. Parker, and B. R. Webber, J. High Energy Phys. 09 (2000) 004; M. Drees, Y. Kim, M. Nojiri, D. Toya, K. Hasuko, and T. Kobayashi, Phys. Rev. D **63**, 035008 (2001); J. Hisano, K. Kawagoe, and M. M. Nojiri, Phys. Rev. D **68**, 035007 (2003); K. Kawagoe, M. M. Nojiri, and G. Polesello, Phys. Rev. D **71**, 035008 (2005); G. G. Ross and M. Serna, Phys. Lett. B **665**, 212 (2008); M. M. Nojiri, G. Polesello, and D. Tovey, J. High Energy Phys. 05 (2008) 014; N. Kersting, Phys. Rev. D **79**, 095018 (2009); D. Costanzo and D. R. Tovey, J. High Energy Phys. 04 (2009) 084; M. Burns, K. T. Matchev, and M. Park, J. High Energy Phys. 05 (2009) 094; H.-C. Cheng, J. F. Gunion, Z. Han, and B. McElrath, Phys. Rev. D **80**, 035020 (2009); K. T. Matchev, F. Moortgat, L. Pape, and M. Park, J. High Energy Phys. 08 (2009) 104; B. Webber, J. High Energy Phys. 09 (2009) 124; K. T. Matchev, F. Moortgat, L. Pape, and M. Park, Phys. Rev. D **82**, 077701 (2010); V. Barger, Y. Gao, A. Lessa, and X. Tata, Phys. Rev. D **83**, 095013 (2011); H.-C. Cheng and J. Gu, J. High Energy Phys. 10 (2011) 094; B. Dutta, T. Kamon, A. Krislock, K. Sinha, and K. Wang, Phys. Rev. D **85**, 115007 (2012); W. S. Cho, D. Kim, K. T. Matchev, and M. Park, arXiv:1206.1546; N. Pietsch, J. Reuter, K. Sakurai, and D. Wiesler, J. High Energy Phys. 07 (2012) 148.

[7] C. G. Lester and D. J. Summers, Phys. Lett. B **463**, 99 (1999); C. Lester and A. Barr, J. High Energy Phys. 12 (2007) 102; B. Gripaios, J. High Energy Phys. 02 (2008) 053; A. J. Barr, B. Gripaios, and C. G. Lester, J. High Energy Phys. 02 (2008) 014; W. S. Cho, K. Choi, Y. G. Kim, and C. B. Park, Phys. Rev. Lett. **100**, 171801 (2008); J. High Energy Phys. 02 (2008) 035; M. M. Nojiri, Y. Shimizu, S. Okada, and K. Kawagoe, J. High Energy Phys. 06 (2008) 035; D. R. Tovey, J. High Energy Phys. 04 (**2008**) 034; A. J. Barr, G. G. Ross, and M. Serna, Phys. Rev. D **78**, 056006 (2008); M. M. Nojiri, K. Sakurai, Y. Shimizu, and M. Takeuchi, J. High Energy Phys. 10 (2008) 100; H.-C. Cheng and Z. Han, J. High Energy Phys. 12 (2008) 063; M. Burns, K. Kong, K. T. Matchev, and M. Park, J. High Energy Phys. 10 (2008) 081; A. J. Barr, A. Pinder, and M. Serna, Phys. Rev. D **79**, 074005 (2009); T. Han, I.-W. Kim, and J. Song, Phys. Lett. B **693**, 575 (2010); A. J. Barr, B. Gripaios, and C. G. Lester, J. High Energy Phys. 11 (2009) 096; G. Polesello and D. R. Tovey, J. High Energy Phys. 03 (2010) 030; I.-W. Kim, Phys. Rev. Lett. **104**, 081601 (2010); K. T. Matchev and M. Park, Phys. Rev. Lett. **107**, 061801 (2011); P. Konar, K. Kong, K. T. Matchev, and M. Park, J. High Energy Phys. 04 (2010) 086; T. Cohen, E. Kuflik, and K. M. Zurek, J. High Energy Phys. 11 (2010) 008; C. G. Lester, J. High Energy Phys. 05 (2011) 076; A. J. Barr, T. J. Khoo, P. Konar, K. Kong, C. G. Lester, K. T. Matchev, and M. Park, Phys. Rev. D **84**, 095031 (2011); K. Choi, D. Guadagnoli, and C. B. Park, J. High Energy Phys. 11 (2011) 117; P. Baringer, K. Kong, M. McCaskey, and D. Noonan, J. High Energy Phys. 10 (2011) 101; D. Curtin, Phys. Rev. D **85**, 075004 (2012); T. Han, I.-W. Kim, and J. Song, Phys. Rev. D **87**, 035003 (2013); **87**, 035004 (2013); M. E. Cabrera and J. A. Casas, arXiv:1207.0435.

[8] P. Z. Skands *et al.*, J. High Energy Phys. 07 (2004) 036.

[9] S. P. Martin and M. T. Vaughn, Phys. Lett. B **318**, 331 (1993); D. M. Pierce, J. A. Bagger, K. T. Matchev, and R.-j. Zhang, Nucl. Phys. **B491**, 3 (1997); S. P. Martin, Phys. Rev. D **74**, 075009 (2006).

[10] M. Drees, W. Hollik, and Q. Xu, J. High Energy Phys. 02 (2007) 032; R. Horsky, M. Krämer, A. Mück, and P. M. Zerwas, Phys. Rev. D **78**, 035004 (2008); W. Hollik, J. M. Lindert, and D. Pagani, J. High Energy Phys. 03 (2013) 139; Eur. Phys. J. C **73**, 2410 (2013).

[11] H. Baer, V. D. Barger, D. Karatas, and X. Tata, Phys. Rev. D **36**, 96 (1987).

[12] H. Baer, X. Tata, and J. Woodside, Phys. Rev. D **45**, 142 (1992); H. Baer, M. Drees, C. Kao, M. M. Nojiri, and X. Tata, Phys. Rev. D **50**, 2148 (1994); D. Feldman, Z. Liu, and P. Nath, Phys. Rev. Lett. **99**, 251802 (2007); **100**, 069902 (2008); J. High Energy Phys. 04 (2008) 054; M. Drees, J. M. Kim, and E.-K. Park, Phys. Rev. D **82**, 095005 (2010).

[13] C. G. Lester, M. A. Parker, and M. J. White, J. High Energy Phys. 01 (2006) 080; H. K. Dreiner, M. Krämer, J. M. Lindert, and B. O'Leary, J. High Energy Phys. 04 (2010) 109; J. A. Conley, H. K. Dreiner, L. Glaser, M. Krämer, and J. Tattersall, J. High Energy Phys. 03 (2012) 042.

[14] N. Bornhauser and M. Drees, Phys. Rev. D **86**, 015025 (2012).

[15] G. Aad *et al.* (ATLAS Collaboration), Phys. Lett. B **716**, 1 (2012); S. Chatrchyan *et al.* (CMS Collaboration), Phys. Lett. B **716**, 30 (2012).

[16] B. C. Allanach, Comput. Phys. Commun. **143**, 305 (2002).

[17] A. Djouadi, M. Mühlleitner, and M. Spira, Acta Phys. Pol. B **38**, 635 (2007).

[18] M. Bähr *et al.*, Eur. Phys. J. C **58**, 639 (2008).

[19] H. Baer, C.-h. Chen, F. Paige, and X. Tata, Phys. Rev. D **50**, 4508 (1994).

[20] B. H. Denby, Comput. Phys. Commun. **49**, 429 (1988).

[21] S. Chatrchyan *et al.* (CMS Collabartion), Phys. Rev. D **87**, 072001 (2013).
[22] C. S. Deans, arXiv:1304.2781, and references therein.
[23] C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, Great Britain, 2008).
[24] See http://seal.cern.ch/documents/minuit/mnusersguide.pdf.
[25] D. E. Kaplan, K. Rehermann, M. D. Schwartz, and B. Tweedie, Phys. Rev. Lett. **101**, 142001 (2008); L. G. Almeida, S. J. Lee, G. Perez, I. Sung, and J. Virzi, Phys. Rev. D **79**, 074012 (2009); T. Plehn, M. Spannowsky, M. Takeuchi, and D. Zerwas, J. High Energy Phys. 10 (2010) 078; P. Bandyopadhyay and B. Bhattacherjee, Phys. Rev. D **84**, 035020 (2011); A. Hook, M. Jankowiak, and J. G. Wacker, J. High Energy Phys. 04 (2012) 007; V. Barger and P. Huang, Phys. Lett. B **708**, 296 (2012); J. Thaler and K. Van Tilburg, J. High Energy Phys. 02 (2012) 093; T. Plehn, M. Spannowsky, and M. Takeuchi, Phys. Rev. D **85**, 034029 (2012); B. Bhattacherjee, S. K. Mandal, and M. M. Nojiri, J. High Energy Phys. 03 (2013) 105; M. R. Buckley, T. Plehn, and M. Takeuchi, J. High Energy Phys. 08 (**2013**) 086; C. Chen, Phys. Rev. D **87**, 074007 (2013).
[26] J. M. Butterworth, A. R. Davison, M. Rubin, and G. P. Salam, Phys. Rev. Lett. **100**, 242001 (2008); T. Plehn, G. P. Salam, and M. Spannowsky, Phys. Rev. Lett. **104**, 111801 (2010); D. E. Soper and M. Spannowsky, J. High Energy Phys. 08 (2010) 029; G. D. Kribs, A. Martin, T. S. Roy, and M. Spannowsky, Phys. Rev. D **81**, 111501 (2010); **82**, 095012 (2010).
[27] H. Baer, V. Barger, A. Lessa, and X. Tata, J. High Energy Phys. 09 (2009) 063.
[28] A. Arbey, M. Battaglia, A. Djouadi, F. Mahmoudi, and J. Quevillon, Phys. Lett. B **708**, 162 (2012); A. Arbey, M. Battaglia, A. Djouadi, and F. Mahmoudi, J. High Energy Phys. 09 (2012) 107; O. Buchmüller *et al.*, Eur. Phys. J. C **72**, 2020 (2012); J. L. Feng, P. Kant, S. Profumo, and D. Sanford, Phys. Rev. Lett. **111**, 131802 (2013).
[29] G. G. Ross, K. Schmidt-Hoberg, and F. Staub, J. High Energy Phys. 08 (2012) 074.