Understanding Dropout

Authored by:

Pierre Baldi
Peter J. Sadowski

Abstract

Dropout is a relatively new algorithm for training neural networks which relies on stochastically dropping out" neurons during training in order to avoid the co-adaptation of feature detectors. We introduce a general formalism for studying dropout on either units or connections, with arbitrary probability values, and use it to analyze the averaging and regularizing properties of dropout in both linear and non-linear networks. For deep neural networks, the averaging properties of dropout are characterized by three recursive equations, including the approximation of expectations by normalized weighted geometric means. We provide estimates and bounds for these approximations and corroborate the results with simulations. We also show in simple cases how dropout performs stochastic gradient descent on a regularized error function."

1 Paper Body

Dropout is an algorithm for training neural networks that was described at NIPS 2012 [7]. In its most simple form, during training, at each example presentation, feature detectors are deleted with probability q = 1 ? p = 0.5 and the remaining weights are trained by backpropagation. All weights are shared across all example presentations. During prediction, the weights are divided by two. The main motivation behind the algorithm is to prevent the co-adaptation of feature detectors, or overfitting, by forcing neurons to be robust and rely on population behavior, rather than on the activity of other specific units. In [7], dropout is reported to achieve state-of-the-art performance on several benchmark datasets. It is also noted that for a single logistic unit dropout performs a kind of ?geometric averaging? over the ensemble of possible subnetworks, and conjectured that something similar may occur also in multilayer networks leading to the view that dropout may be an economical approximation to training and using a very large ensemble of networks. In spite of the impressive results that have been reported, little is known about dropout from a theoretical standpoint, in particular about its averaging, regularization, and convergence properties. Likewise little is known about the importance of using q = 0.5, whether different values of q can be used including different values for different layers or different units, and whether dropout can be applied to the connections rather than the units. Here we address these questions.

2

Dropout in Linear Networks

It is instructive to first look at some of the properties of dropout in linear networks, since these can be studied exactly in the most general setting of a multilayer feedforward network described by an underlying acyclic graph. The activity in unit i of layer h can be expressed as: Sih (I) =

XX l¡h

hl l wij Sj

j

1

with Sj0 = Ij

(1)

where the variables w denote the weights and I the input vector. Dropout applied to the units can be expressed in the form Sih =

XX l¡h

hl l l wij ?j Sj

with

Sj0 = Ij

(2)

j

where ?jl is a gating 0-1 Bernoulli variable, with P (?jl = 1) = plj . Throughout this paper we assume that the variables ?jl are independent of each other, independent of the weights, and independent of the activity of the units. Similarly, dropout applied to the connections leads to the random variables Sih =

XX l¡h

hl hl l ?ij wij Sj

with

Sj0 = Ij

(3)

j

For brevity in the rest of this paper, we focus exclusively on dropout applied to the units, but all the results remain true for the case of dropout applied to the connections with minor adjustments. For a fixed input vector, the expectation of the activity of all the units, taken over all possible realizations of the gating variables hence all possible subnetworks, is given by: E(Sih) =

XX l¡h

hl l wij pj E(Sjl)

for h ¿ 0

(4)

j

with E(Sj0) = Ij in the input layer. In short, the ensemble average can easily be computed by hl hl l feedforward propagation in the original network, simply replacing the weights wij by wij pj .

3 3.1

Dropout in Neural Networks Dropout in Shallow Neural Networks

Pn Consider first a single logistic unit with n inputs O = ?(S) = 1/(1 + ce??S) and S = 1 wj Ij . To achieve the greatest level of generality, we assume that the unit produces different outputs OP 1 , . . . , Om , corresponding to different sums S1 . . . , Sm with different probabilities P1 , . . . , Pm (Pm = 1). In the most relevant case, these outputs and these sums are associated with the m = 2n possible subnetworks of the unit. The probabilities P1 , . . . , Pm could be generated, for instance, by using Bernoulli gating variables, although this isP not necessary for this derivation. It is useful to define the following four quantities: the mean E = Pi Oi ; the mean of the complements P Q E0 = Pi (1 ? Oi) = 1 ? E; the weighted geometric mean (W GM) G = i OiPi ; and the Q weighted geometric mean of the complements G0 = i (1 ? Oi)Pi . We also define the normalized weighted geometric mean N W GM = G/(G + G0). We can now prove the key averaging theorem for logistic functions: 1 = ?(E(S)) 1 + ce??E(S)

(5)

1 1 Q Q = (1?Oi)Pi (1??(Si))Pi 1 + Q Pi 1 + Q ?(S)Pi

(6)

N W GM (O1 , . . . , Om) = To prove this result, we write N W GM (O1 , . . . , Om) =

Oi

i

??x

The logistic function satisfies the identity [1 ? ?(x)]/?(x) = ce

and thus

1 1 P Q ??S P = N W GM (O1 , . . . , Om) = = ?(E(S)) (7) i i ?? Pi Si 1 + [ce] 1 + ce Thus in the case of Bernoulli gating variables, we can compute the N WP GM over all possible n dropout configurations by simple forward propagation by: N W GM = ?(1 wj pj Ij). A similar result is true also for normalized exponential transfer functions. Finally, one can also show that the only class of functions f that satisfy N W GM (f) = f (E) are the constant functions and the logistic functions [1]. 2

3.2

Dropout in Deep Neural Networks

We can now deal with the most interesting case of deep feedforward networks of sigmoidal units 1 , described by a set of equations of the form Oih = ?(Sih) = ?(

XX

hl l wij Oj)

with

Oj0 = Ij

(8)

j
l¡h
where Oih is the output of unit i in layer h. Dropout on the units can be described by Oih = ?(Sih) = ?(
XX l¡h
hl l l wij ?j Oj)
with Oj0 = Ij
(9)
j
using the Bernoulli selector variables ?jl . For each sigmoidal unit N W GM (Oih) = Q
h P (N) N (Oi) Q h P (N) + N (1 ? N (Oi)
Q
Oih)P (N)
(10)
where N ranges over all possible subnetworks. Assume for now that the N W GM provides a good approximation to the expectation (this point will be analyzed in the next section). Then the averaging properties of dropout are described by the following three recursive equations. First the approximation of means by NWGMs: E(Oih) ? N W GM (Oih)
(11)
Second, using the result of the previous section, the propagation of expectation symbols:
N W GM (Oih) = ?ih E(Sih)
(12)
And third, using the linearity of the expectation with respect to sums, and to products of independent random variables: E(Sih) =
XX l¡h
hl l wij pj E(Ojl)
(13)
j
Equations 11, 12, and 13 are the fundamental equations explaining the averaging properties of the dropout procedure. The only approximation is of course Equation 11 which is analyzed in the next section. If the network contains linear units, then Equation 11 is not necessary for those units and their average can be computed exactly. In the case of regression with linear units in the top layers, this allows one to shave off one layer of approximations. The same is true in binary classification by requiring the output layer to compute directly the N W GM of the ensemble rather than the expectation. It can be shown that for any error function that is convex up (?), the error of the mean, weighted geometric mean, and normalized weighted geometric mean of an ensemble is always less than the expected error of the models [1]. Equation 11 is exact if and only if the numbers Oih are identical over all possible subnetworks N . h Thus it is useful to measure the consistency C(Oi , I) of neuron i in layer h for input I by using the variance V ar Oih (I) taken over all subnetworks N and their distribution when the input I is fixed. The larger the variance is, the less consistent the

neuron is, and the worse we can expect the approximation in Equation 11 to be. Note that for a random variable O in [0,1] the variance cannot exceed 1/4 anyway. This is because V ar(O) = E(O2) ? (E(O))2 ? E(O) ? (E(O))2 = E(O)(1 ? E(O)) ? 1/4. This measure can also be averaged over a training set or a test set. 1 Given the results of the previous sections, the network can also include linear units or normalized exponential units.

3

4

The Dropout Approximation

Given a set of numbers O1 , . . . , Om between 0 and 1, with probabilities P1 , . . . , PM (corresponding to the outputs of a sigmoidal neuron for a fixed input and different subnetworks), we are primarily interested in the approximation of E by N W GM . The N W GM provides a good approximation because we show below that to a first order of approximation: E ? N W GM and E ? G. Furthermore, there are formulae in the literature for bounding the error E ? G in terms of the consistency (e.g. the Cartwright and Field inequality [6]). However, one can suspect that the N W GM provides even a better approximation to E than the geometric mean. For instance, if the numbers Oi satisfy 0 ¡ Oi ? 0.5 (consistently low), then G E G ? 0 and therefore G ? ?E (14) G0 E G + G0 This is proven by applying Jensen?s inequality to the function ln x ? ln(1 ? x) for x ? (0, 0.5]. It is also known as the Ky Fan inequality [2, 8, 9]. To get even better results, one must consider a second order approximation. For this, we write Oi = 0.5 + i with 0 ? —i — ? 0.5. Thus we have E(O) = 0.5 + E() and V ar(O) = V ar(). Using a Taylor expansion: ? ? ? X X pi (pi ? 1) X 1 Y X pi 1 G= (2i)n = ?1 + pi 2i + (2i)2 + 4pi pj i j + R3 (i)? 2 i n=0 n 2 2 i i i¡j (15) where R3 (i) is the remainder and R3 (i) =

pi (2i)3 3 (1 + ui)3?pi

(16)

where —ui — ? 2—i —. Expanding the product gives X X 1 X 1 G= + pi i +(i)2 ? pi 2i +R3 () = +E()?V ar()+R3 () = E(O)?V ar(O)+R3 () 2 i 2 i (17) By symmetry, we have G0 =

Y (1 ? Oi)pi = 1 ? E(O) ? V ar(O) + R3 ()

(18)

i

where R3 () is the higher order remainder. Neglecting the remainder and writing E = E(O) and V = V ar(O) we have G E?V G0 1?E?V ? and ? (19) 0 0 G+G 1 ? 2V G+G 1 ? 2V Thus, to a second order, the differences between the mean and the geometric mean and the normalized geometric means satisfy E?G?V

and E ?

G V (1 ? 2E) ? 0 G+G 1 ? 2V

(20)

and G0 V (1 ? 2E) ? (21) 0 G+G 1 ? 2V Finally it is easy to check that the factor (1 ? 2E)/(1 ? 2V) is always less or equal to 1. In addition we always have V ? E(1 ? E), with equality achieved only for 0-1 Bernoulli variables. Thus 1 ? E ? G0 ? V

and
(1 ? E) ?
4
V —1 ? 2E— E(1 ? E)—1 ? 2E— G —? ? ? 2E(1 ? E)—1 ? 2E— (22) G + G0 1 ? 2V 1 ? 2V The first inequality is optimal in the sense that it is attained in the case of a Bernoulli variable with expectation E and, intuitively, the second inequality shows that the approximation error is always small, regardless of whether E is close to 0, 0.5, or 1. In short, the NWGM provides a very good approximation to E, better than the geometric mean G. The property is always true to a second order of approximation and it is exact when the activities are consistently low, or when N W GM ? E, since the latter implies G ? N W GM ? E. Several additional properties of the dropout approximation, including the extension to rectified linear units and other transfer functions, are studied in [1]. —E ?
5
Dropout Dynamics
Dropout performs gradient descent on-line with respect to both the training examples and the ensemble of all possible subnetworks. As such, and with the appropriately decreasing learning rates, it is almost surely convergent like other forms of stochastic gradient descent [11, 4, 5]. To further understand the properties of dropout, it is again instructive to look at the properties of the gradient in the linear case. 5.1
Single Linear Unit
In the case of a single linear unit, consider the two error functions EEN S and ED associated with the ensemble of all possible subnetworks and the network with dropout. For a single input I, these are defined by:
EEN S
n X 1 1 2 = (t ? OEN S) = (t ? pi wi Ii)2 2 2 i=1
(23)
n X 1 1 (t ? OD)2 = (t ? ?i wi Ii)2 2 2 i=1
(24)
ED =
We use a single training input I for notational simplicity, otherwise the errors of each training example can be combined additively. The learning gradient is given by ?EEN S ?OEN S = ?(t ? OEN S) = ?(t ? OEN S)pi Ii ?wi ?wi X ?ED ?OD = ?(t ? OD) = ?(t ? OD)?i Ii = ?t?i Ii + wi ?i2 Ii2 + wj ?i ?j Ii Ij ?wi ?wi
(25)
(26)
j6=i
The dropout gradient is a random variable and we can take its expectation. A short calculation yields
E
?ED ?wi
=
?EEN S ?EEN S + wi pi (1 ? pi)Ii2 + wi Ii2 V ar(?i) ?wi ?wi

(27)

Thus, remarkably, in this case the expectation of the gradient with dropout is the gradient of the regularized ensemble error n

E = EEN S +

1X 2 2 w I V ar(?i) 2 i=1 i i

(28)

The regularization term is the usual weight decay or Gaussian prior term based on the square of the weights to prevent overfitting. Dropout provides immediately the magnitude of the regularization term which is adaptively scaled by the inputs and by the variance of the dropout variables. Note that pi = 0.5 is the value that provides the highest level of regularization. 5

5.2

Single Sigmoidal Unit

The previous result generalizes to a sigmoidal unit O = ?(S) = 1/(1 + ce??S) trained to minimize the relative entropy error E = ?(t log O + (1 ? t) log(1 ? O)). In this case, ?ED ?S = ??(t ? O) = ??(t ? O)?i Ii (29) ?wi ?wi The terms O and Ii are not independent but using a Taylor expansion with the N W GM approximation gives

?EEN S ?ED ? + ?? 0 (SEN S)wi Ii2 V ar(?i) (30) E ?wi ?wi P with SEN S = j wj pj Ij . Thus, as in the linear case, the expectation of the dropout gradient is approximately the gradient of the ensemble network regularized by weight decay terms with the proper adaptive coefficients. A similar analysis, can be carried also for a set of normalized exponential units and for deeper networks [1].

5.3

Learning Phases and Sparse Coding

During dropout learning, we can expect three learning phases: (1) At the beginning of learning, when the weights are typically small and random, the total input to each unit is close to 0 for all the units and the consistency is high: the output of the units remains roughly constant across subnetworks (and equal to 0.5 with c = 1). (2) As learning progresses, activities tend to move towards 0 or 1 and the consistency decreases, i.e. for a given input the variance of the units across subnetworks increases. (3) As the stochastic gradient learning procedure converges, the consistency of the units converges to a stable value. Finally, for simplicity, assume that dropout is applied only in layer h where the units have an output P hl l l of the form Oih = ?(Sih) and Sih = l¡h wij ?j Oj . For a fixed input, Ojl is a constant since dropout is not applied to layer l. Thus V ar(Sih) =

X

hl 2 (wij) (Ojl)2 plj (1 ? plj)

(31)

l¡h

under the usual assumption that the selector variables ?jl are independent of each other. Thus V ar(Sih) depends on three factors. Everything else being equal, it is reduced by: (1) Small weights which goes together with the regularizing effect of dropout; (2) Small activities, which shows that dropout is not symmetric with respect to small or large activities. Overall, dropout tends to favor small activities and thus sparse coding; and (3) Small (close to 0) or large (close to 1) values of the dropout probabilities plj . Thus values plj = 0.5 maximize the regularization effect but may also lead to slower convergence to the consistent state. Additional results and simulations are given in [1].

6

Simulation Results

We use Monte Carlo simulation to partially investigate the approximation framework embodied by the three fundamental dropout equations 11, 12, and 13, the accuracy of the second-order approximation and bounds in Equations 20 and 22, and the dynamics of dropout learning. We experiment with an MNIST classifier of four hidden layers (784-1200-1200-1200-1200-10) that replicates the results in [7] using the Pylearn2 and Theano software libraries[12, 3]. The network is trained with a dropout probability of 0.8 in the input, and 0.5 in the four hidden layers. For fixed weights and a fixed input, 10,000 Monte Carlo simulations are used to estimate the distribution of activity O in each neuron. Let O? be the activation under the deterministic setting with the weights scaled appropriately. The left column of Figure 1 confirms empirically that the second-order approximation in Equation 20 and the bound in Equation 22 are accurate. The right column of Figure 1 shows the difference between the true ensemble average E(O) and the prediction-time neuron activity O? . This difference grows very slowly in the higher layers, and only for active neurons. 6

Figure 1: Left: The difference E(O) ? N W GM (O), it?s second-order approximation in Equation 20, and the bound from Equation 22, plotted for four hidden layers and a typical fixed input. Right: The difference between the true ensemble average E(O) and the final neuron prediction O? . Next, we examine the neuron consistency during dropout training. Figure 2a shows the three phases of learning for a typical neuron. In Figure 2b, we observe that the consistency does not decline in higher layers of the network. One clue into how this happens is the distribution of neuron activity. As noted in [10] and section 5 above, dropout training results in sparse activity in the hidden layers (Figure 3). This increases the consistency of neurons in the next layer.

7

(a) The three phases of learning. For a particular input, a typical active neuron (red) starts out with low variance, experiences a large increase in variance during learning, and eventually settles to some steady constant value. In contrast, a typical inactive neuron (blue) quickly learns to stay silent. Shown are the mean with 5% and 95% percentiles.

(b) Consistency does not noticeably decline in the upper layers. Shown here are the mean Std(O) for active neurons (0.1 ¡ O after training) in each layer, along with the 5% and 95% percentiles.

Figure 2

Figure 3: In every hidden layer of a dropout trained network, the distribution of neuron activations O? is sparse and not symmetric. These histograms were totalled over a set of 100 random inputs.

8

2 References

[1] P. Baldi and P. Sadowski. The Dropout Learning Algorithm. Artificial Intelligence, 2014. In press. [2] E. F. Beckenbach and R. Bellman. Inequalities. Springer-Verlag Berlin, 1965. [3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), Austin, TX, June 2010. Oral Presentation. [4] L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK, 1998. [5] L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, Advanced Lectures on Machine Learning, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146?168. Springer Verlag, Berlin, 2004. [6] D. Cartwright and M. Field. A refinement of the arithmetic mean-geometric mean inequality. Proceedings of the American Mathematical Society, pages 36?38, 1978. [7] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. http://arxiv.org/abs/1207.0580, 2012. [8] E. Neuman and J. S?andor. On the Ky Fan inequality and related inequalities i. MATHEMATICAL INEQUALITIES AND APPLICATIONS, 5:49?56, 2002. [9] E. Neuman and J. Sandor. On the Ky Fan inequality and related inequalities ii. Bulletin of the Australian Mathematical Society, 72(1):87?108, 2005. [10] S. Nitish. Improving Neural Networks with Dropout. PhD thesis, University of Toronto, Toronto, Canada, 2013. [11] H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. Optimizing methods in statistics, pages 233?257, 1971. [12] D. Warde-Farley, I. Goodfellow, P. Lamblin, G. Desjardins, F. Bastien, and Y. Bengio. pylearn2. 2011. http://deeplearning.net/software/pylearn2.

9