No subject

https://mailman.srv.cs.cmu.edu/pipermail/connectionists/1989-January/011586.html

John Kruschke kruschke at cogsci.berkeley.edu
Tue Jan 3 03:30:12 EST 1989

```
Here is the compilation of responses to my request for info on
weight decay.

I have kept editing to a minimum, so you can see exactly what the
author of the reply said. Where appropriate, I have included some
comments of my own, set off in square brackets.  The responses are
arranged into three broad topics: (1) Boltzmann-machine related;
(2) back-prop related; (3) psychology related.

Thanks to all, and happy new year!  --John



-----------------------------------------------------------------

ORIGINAL REQUEST:

I'm interested in all the information I can get regarding
WEIGHT DECAY in back-prop, or in other learning algorithms.

*In return* I'll collate all the info contributed and send the
complilation out to all contributors.

Info might include the following:

  REFERENCES:
  - Applications which used weight decay
  - Theoretical treatments
  Please be as complete as possible in your citation.

  FIRST-HAND EXPERIENCE
  - Application domain, details of I/O patterns, etc.
  - exact decay procedure used, and results

(Please send info directly to me: kruschke at cogsci.berkeley.edu
 Don't use the reply command.)

T H A N K S !  --John Kruschke.


-----------------------------------------------------------------

From:  Geoffrey Hinton <hinton at ai.toronto.edu>
Date:  Sun, 4 Dec 88 13:57:45 EST


Weight-decay is a version of what statisticians call "Ridge
Regression".
```

We used weight-decay in Boltzmann machines to keep the energy barriers small. This is described in section 6.1 of:
   Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984)
   Boltzmann Machines: Constraint satisfaction networks that learn.
   Technical Report CMU-CS-84-119, Carnegie-Mellon University.

I used weight decay in the family trees example. Weight decay was used to improve generalization and to make the weights easier to interpret (because, at equilibrium, the magnitude of a weight = its usefulness). This is in:
   Rumelhart, D.~E., Hinton, G.~E., and Williams, R.~J. (1986)
   Learning representations by back-propagating errors.
   {\it Nature}, {\bf 323}, 533--536.

I used weight decay to achieve better generalization in a hard generalization task that is reported in:
   Hinton, G.~E. (1987)
   Learning translation invariant recognition in a massively
   parallel network. In Goos, G. and Hartmanis, J., editors,
   {\it PARLE: Parallel Architectures and Languages Europe},
   pages~1--13, Lecture Notes in Computer Science,
   Springer-Verlag, Berlin.

Weight-decay can also be used to keep "fast" weights small. The fast weights act as a temporary context. One use of such a context is described in:
   Hinton, G.~E. and Plaut, D.~C. (1987)
   Using fast weights to deblur old memories.
   {\it Proceedings of the Ninth Annual Conference of the
   Cognitive Science Society}, Seattle, WA.

--Geoff

[In his lecture at the International Computer Science Institute, Berkeley CA, on 16-DEC-88, Geoff also mentioned that weight decay is good for wiping out the initial values of weights so that only the effects of learning remain.

In particular, if the change (due to learning) on two weights is the same for all updates, then the two weights converge to the same value. This is one way to generate symmetric weights from non-symmetric starting values.

--John]

From: Michael.Franzini at SPEECH2.CS.CMU.EDU
Date: Sun, 4 Dec 1988 23:24-EST

My first-hand experience confirms what I'm sure many other people have told you: that (in general) weight decay in backprop increases generalization. I've found that it's particulary important for small training sets, and its effect diminishes as the training set size increases.

Weight decay was first used by Barak Pearlmutter. The first mention of weight decay is, I believe, in an early paper of Hinton's (possibly the Plaut, Nowlan, and Hinton CMU CS tech report), and it is attributed to "Barak Pearlmutter, Personal Communication" there.

The version of weight decay that (i'm fairly sure) all of us at CMU use is one in which each weight is multiplied by 0.999 every epoch. Scott Fahlman has a more complicated version, which is described in his QUICKPROP tech report. [QuickProp is also described in his paper in the Proceedings of the 1988 Connectionist Models Summer School, published by Morgan Kaufmann. --John]

The main motivation for using it is to eliminate spurious large weights which happen not to interfere with recognition of training data but would interfere with recognizing testing data. (This was Barak's motivation for trying it in the first place.) However, I have heard more theoretical justifications (which, unfortunately, I can't reproduce.)

In case Barak didn't reply to your message, you might want to contact him directly at bap at cs.cmu.edu.

--Mike

From: Barak.Pearlmutter at F.GP.CS.CMU.EDU
Date: 8 Dec 1988 16:36-EST

We first used weight decay as a way to keep weights in a boltzmann machine from growing too large. We added a term to the thing being minimized, G, so that

$$G' = G + 1/2 h \sum_{i<j} w_{ij}^2$$

where G' is our new thing to minimize. This gives

$$\partial G'/\partial w_{ij} = \partial G/\partial w_{ij} + h w_{ij}$$

which is just weight decay with some mathematical motivation. As Mike mentioned, I was the person who thought of weight decay in this context (in the shower no less), but parameter decay has been used forever, in adaptive control for example.

It sort of worked okay for Boltzmann machines, but works much better in backpropagation. As a historic note I should mention that there were some competing techniques for keeping weights small in Boltzmann machines, such as Mark Derthick's "differential glommetry" in which the effective target termperature of the wake phase is higher than that of the sleep phase. I don't know if there is an analogue for this in backpropagation, but there certainly is for mean field theory networks.

Getting back weight decay, it was noted immediately that G has the unit "bits" while w_{ij}^2 has the unit "weight^2", sort of a problem from a dimensional analysis point of view. Solving this conundrum, Rick Szeliski pointed out that if we're going to transmit our weights by telephone and know a-priori that weights have gaussian distributions, so

$$P(w_{ij}=x) \propto e^{-1/2 h x^2}$$

where h is set to get the correct variance, then transmitting a weight w will take $-1/2 h w^2$ bits, which we can add to G with dimensional confidence.

Of course, this argument extends to fast/slow split weights nicely; the other guy already knows the slow weights, so we need transmit only the fast weights.

By "ridge regression" I guess Geoff means that valleys in weight space that cause the weights to grow asymptotically are made to tilt up after a while, so that the asymptotic tailing off is eliminated. It's like adding a bowl to weight space, so minima have to be within the bowl.

An interesting side effect of weight decay is that, once we get to a minimum, so \$\partial G'/\partial w = 0\$, then

    w_{ij} \propto - \partial G/\partial w_{ij}

so we can do a sort of eyeball significance analysis, since a weight's magnitiude is proportaional to how sensitive the error is to changing it.

From: russ%yummy at gateway.mitre.org (Russell Leighton)
Date: Mon, 5 Dec 88 09:17:56 EST

We always use weight decay in backprop. It is partiuclarly important in escaping local minima. Decay moves the transfer function from all of the semi-linear (sigmoidal) nodes toward the linear region. The important point is that all nodes move proportionally so no information in the weights is "erased" but only scaled. When the nodes that have trapped the system in the local minima are scaled enough, the system moves onto a different trajectory through weight space. Oscilations are still possible, but are less likely.

We use decay with a process we call "shaping" (see Wieland and Leighton, "Shaping Schedules as a Method for Accelerating Leanring", Abstracts of the First Annual INNS Meeting, Boston, 1988) that we use to speed learning of some difficult problems.

ARPA: russ%yummy at gateway.mitre.org

Russell Leighton
MITRE Signal Processing Lab
7525 Colshire Dr.
McLean, Va. 22102
USA

From: James Arthur Pittman <hi.pittman at MCC.COM>
Date: Tue, 6 Dec 88 09:34 CST

Probably he will respond to you himself, but Alex Weiland of MITRE presented a paper at INNS in Boston on shaping, in which the order of presentation of examples in training a back-prop net was altered to reflect a simpler rule at first. Over a number of epochs he gradually changed the examples to slowly change the rule to the one desired. The nets learned much faster than if he just tossed the examples at the net in random order. He told me that it would not work without weight decay. He said their rule-of-thumb was the decay should give the weights a half-life of 2 to 3 dozen epochs (usually a value such as 0.9998). But I neglected to ask him if he felt that the number of epochs or the number of presentations was important. Perhaps if one had a significantly different training set size, that rule-of-thumb would be different?

I have started some experiments simular to his shaping, using some random variation of the training data (where the random variation grows over time). Weiland also discussed this in his talk. I haven't yet compared decay with no-decay. I did try (as a lark) using decay with a regular (non-shaping) training, and it did worse than we usually get (on same data and same network type/size/shape). Perhaps I was using a stupid decay value (0.9998 I think) for that situation.

I hope to get back to this, but at the moment we are preparing for a software release to our shareholders (MCC is owned by 20 or so computer industry corporations). In the next several weeks a lot of people will go on Christmas vacation, so I will be able to run a bunch of nets all at once. They call me the machine vulture.

From: Tony Robinson <ajr at digsys.engineering.cambridge.ac.uk>
Date: Sat, 3 Dec 88 11:10:20 GMT

Just a quick note in reply to your message to `connectionists' to say that I have tried to use weight decay with back-prop on networks with order 24 i/p, 24 hidden, 11 o/p units. The problem was vowel recognition (I think), it was about 18 months ago, and the problem was of the unsolvable type (i.e. non-zero final energy).

My conclusion was that weight decay only made matters worse, and my justification (to myself) for abandoning weight decay was that you are not even pretending to do gradient descent any more, and any good solution formed quickly becomes garbaged by scaling the weights.

If you want to avoid hidden units sticking on their limiting values, why not use hidden units with no limiting values, for instance I find the activation function f(x) = x * x works better than f(x) = 1.0 / (1.0 + exp(- x)) anyway.

Sorry I havn't got anything formal to offer, but I hope these notes help.

Tony Robinson.

From: jose at tractatus.bellcore.com (Stephen J Hanson)
Date: Sat, 3 Dec 88 11:54:02 EST

Actually, "costs" or "penalty" functions are probably better terms. We had a poster last week at NIPS that discussed some of the pitfalls and advantages of two kinds of costs. I can send you the paper when we have a version available.

Stephen J. Hanson (jose at bellcore.com)

[In a conversation in his office on 06-DEC-88, Dave Rumelhart described to me several cost functions he has tried.

The motive for the functions he has tried is different from the motive for standard weight decay. Standard weight decay,

\sum_{i,j} w_{i,j}^2 ,

is used to *distribute* weights more evenly over the given connections, thereby increasing robustness (cf. earlier replies).

He has tried several other cost functions in an attempt to *localize*, or concentrate, the weights on a small subset of the given connections. The goal is to improve generalization. His favorite is

\sum_{i,j} (w_{i,j}^2 / (K + w_{i,j}^2))

where K is a constant, around 1 or 2. Note that this function is negatively accelerating, whereas standard weight decay is positively accelerating. This function penalizes small weights (proportionally) more than large weights, just the opposite of standard weight decay.

He has also tried, with less satisfying results,

\sum (1 - \exp - (\alpha w_{i,j}^2))

and

\sum \ln (K + w_{i,j}^2).

Finally, he has tried a cost function designed to make all the fan-in weights of a single unit decay, when possible. That is, the unit is effectively cut out of the network. The function is

\sum_i (\sum_j w_{i,j}^2) / (K + \sum_j w_{i,j}^2).

Each weight is thereby penalized (inversely) proportionally to the total fan-in weight of its node.

--John]

[This is also a relevant place to mention my paper in the Proceedings of the 1988 Connectionist Models Summer School, "Creating local and distributed bottlenecks in back-propagation networks". I have since developed those ideas, and have expressed the localized bottleneck method as gradient descent on an additional cost term. The cost term is quite general, and some forms of decay are simply special cases of it. --John]

From: john moody <moody-john at YALE.ARPA>
Date: Sun, 11 Dec 88 22:54:11 EST

Scalettar and Zee did some interesting work on weight decay with back prop for associative memory. They found that a Unary Representation emerged (see Baum, Moody, and Wilczek; Bio Cybernetics Aug or Sept 88 for info on Unary Reps). Contact Tony Zee at UCSB (805)961-4111 for info on weight decay paper.

--John Moody

From: gluck at psych.Stanford.EDU (Mark Gluck)
Date: Sat, 10 Dec 88 16:51:29 PST

I'd appreciate a copy of your weight decay collation. I have a paper in MS form which illustrates how adding weight decay to the linear-LMS one-layer net improves its ability to predict human generalization in classification learning.

mark gluck
dept of psych
stanford univ,
stanford, ca 94305

From: INAM000 <INAM%MCGILLB.bitnet at jade.berkeley.edu> (Tony Marley)
Date: SUN 04 DEC 1988 11:16:00 EST

I have been exploring some ideas re COMPETITIVE LEARNING with "noisy weights" in modeling simple psychophysics. The task is the classical one of identifying one of N signals by a simple (verbal) response -e.g. the stimuli might be squares of different sizes, and one has to identify the presented one by saying the appropriate integer. We know from classical experiments that people cannot perform this task perfectly once N gets larger than about 7, but performance degrades smoothly for larger N.

I have been developing simulations where the mapping is learnt by competitive learning, with the weights decaying/varying over time when they are not reset by relevant inputs. I have not got too many results to date, as I have been taking the psychological data seriously, which means worrying about reaction times, sequential effects, "end effects" (stimuli at the end of the range more accurately identified), range effects (increasing the stimulus range has little effect), etc..

Tony Marley

From: aboulanger at bbn.com (Albert Boulanger)
Date: Fri, 2 Dec 88 19:43:14 EST

This one concerns the Hopfield model. In
    James D Keeler,
    "Basin of Attraction of Neural Network Models",
    Snowbird Conference Proceedings (1986), 259-264,
it is shown that the basins of attraction become very complicated as the number of stored patterns increase. He uses a weight modification method called "unlearning" to smooth out these basins.

Albert Boulanger
BBN Systems & Technologies Corp.
aboulanger at bbn.com

From: Joerg Kindermann <unido!gmdzi!joerg at uunet.UU.NET>
Date: Mon, 5 Dec 88 08:21:03 -0100

We used a form of weight decay not for learning but for recall in multilayer feedforward networks. See the following abstract. Input patterns are treated as ``weights'' coming from a constant valued external unit.

If you would like a copy of the technical report, please send e-mail to
joerg at gmdzi.uucp
or write to:
Dr. Joerg Kindermann
Gesellschaft fuer Mathematik und Datenverarbeitung
Schloss Birlinghoven
Postfach 1240
D-5205 St. Augustin 1
WEST GERMANY

Detection of Minimal Microfeatures by Internal Feedback
J. Kindermann & A. Linden

Abstract

We define the notion of minimal microfeatures and introduce a new method of internal feedback for multilayer networks. Error signals are used to modify the input of a net. When combined with input DECAY, internal feedback allows the detection of sets of minimal microfeatures, i.e. those subpatterns which the network actually uses for discrimination. Additional noise on the training data increases the number of minimal microfeatures for a given pattern. The detection of minimal microfeatures is a first step towards a subsymbolic system with the capability of self-explanation. The paper provides examples from the domain of letter recognition.

From: Helen M. Gigley <hgigley at note.nsf.gov>
Date: Mon, 05 Dec 88 11:03:23 -0500

I am responding to your request even though my use of decay is not with respect to learning in connectionist-like models. My focus has been on a functioning system that can be lesioned.

One question I have is what is the behavioral association to weight decay? What aspects of learning is it intended to reflect. I can understand that activity decay over time of each cell is meaningful and reflects a cellular property, but what is weight decay in comparable terms?

Now, I will send you offprints if you would like of my work and am including a list of several publications which you may be able to peruse. The model, HOPE, is a hand-tuned structural connectionist model that is designed to enable lesioning without redesign or reprogramming to study possible processing causes of aphasia. Decay factors as an integral part of dynamic time-dependent processes are one of several aspects of processing in a neural environment which potentially affect the global processing results even though they are defined only locally. If I can be of any additional help please let me know.

```
Helen Gigley


References:

Gigley, H.M. Neurolinguistically Constrained Simulation of Sentence
Comprehension:  Integrating Artificial Intelligence and Brain Theorym
Ph.D. Dissertation, UMass/Amherst, 1982.  Available University
Microfilms, Ann Arbor, MI.

Gigley, H.M.  HOPE--AI and the dynamic process of language behavior.
in Cognition and Brain Theory 6(1) :39-88, 1983.

Gigley, H.M.  Grammar viewed as a functioning part of of a cognitive
system. Proceedings of ACL 23rd Annual Meeting, Chicago, 1985 .

Gigley, H.M.  Computational Neurolinguistics -- What is it all about?
in IJCAI Proceedings, Los Angeles, 1985.

Gigley, H.M.  Studies in Artificial Aphasia--experiments in processing
change.  In Journal of Computer Methods and Programs in Biomedicine,
22 (1): 43-50, 1986.

Gigley, H.M.  Process Synchronization, Lexical Ambiguity Resolution,
and Aphasia.  In Steven L. Small, Garrison Cottrell, and Michael
Tanenhaus (eds.)  Lexical Ambiguity Resolution, Morgen Kaumann, 1988.



-------------------------------------------------------------------

From:  bharucha at eleazar.Dartmouth.EDU (Jamshed Bharucha)
Date:  Tue, 13 Dec 88 16:56:00 EST

I haven't tried weight decay but am curious about it. I am working on
back-prop learning of musical sequences using a Jordan-style net. The
network develops a musical schema after learning lots of sequences
that have culture-specific regularities. I.e., it learns to generate
expectancies for tones following a sequential context. I'm interested
in knowing how to implement forgetting, whether short term or long
term.

Jamshed.

-------------------------------------------------------------------
```

- Previous message: Benchmarks mailing list
- Next message: No subject
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the Connectionists mailing list