The Population Method for Training Neural Network Ensembles

Henrik Haraldsson and Mattias Ohlsson

Complex Systems Division, Department of Theoretical Physics, Lund University, Sweden

Introduction

A randomly initialized ensemble of artificial neural networks is trained by letting each network take small random steps in weight space. Each time all the networks have taken one random step, several identical replicas are created of well-adapted networks and poorly adapted networks vanish. The replication follows a Boltzmann distribution, with a fictitious temperature controlling the increasing level of competition between the networks. The result is a diverse ensemble. Numerical explorations show that our method yields comparable results to Bayesian learning for ANNs, and outperforms a simple ANN ensemble.

The Individual Networks

- MLPs with one hidden layer
- Energy E given by mean-square or entropy error, plus regularisation term
- The Population Method is in principle applicable to any type of model with tunable parameters and well-defined energy!

The Population Method

Iteratively repeat items 1-3:

0) Start with N networks initialized by random weight vectors $\vec{\omega}_i$ ($i = 1, \ldots, N$).

1) Random move: Update each $\vec{\omega}_i$ by making a random move: $\vec{\omega}_i \to \vec{\omega}_i + \epsilon\vec{\alpha}_i$, where $\vec{\alpha}_i$ is a vector of length 1 with a random direction and ϵ is a small number.

2) Calculate population factors: Introduce a fictitious temperature T which determines the amount of competition between different networks. Compute the energies E_i, Boltzmann factors B_i and population (fitness) factors r_i:

$$\begin{aligned} B_i &= e^{-E_i/T} \\ r_i &= \frac{B_i}{\langle B \rangle} = N \frac{e^{-E_i/T}}{\sum_j e^{-E_j/T}} \end{aligned}$$

3) Create a new population: For each network i, write r_i as

$$r_i = L_i + \delta_i$$

where $L_i \geq 0$ is the integer part of r_i and δ_i is the remainder, $0 \leq \delta_i < 1$. Place L_i replicas of network i into the new population. Add one more replica with probability δ_i; this stochastic element among other things gives poorly fitted networks (with $r_i < 1$) a chance of surviving. The population size is kept approximately constant throughout.

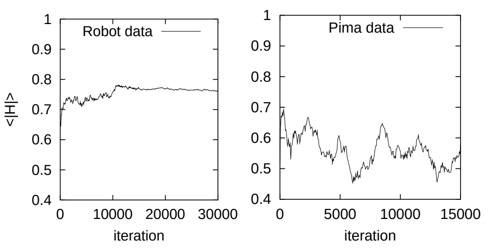
Temperature T and stepsize ϵ are annealed (lowered gradually). Large steps and low competition in beginning of training allows for long-range exploration of state space. Small steps and high competition in the end implies a detailed exploration of interesting regions. After the annealing, the population continues to evolve before the actual network ensemble is sampled.

Properties of the method

Correlation measures

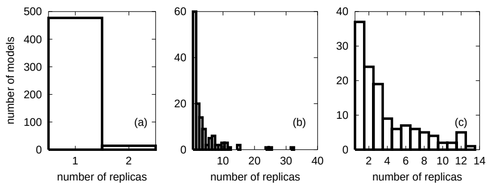
A stopping criterion, expressed in terms of the correlation length of some network quantity, would be desireable.

- We studied average absolute value of hidden unit activation, $\langle|H|\rangle$, averaged over all of population.



- Integrated correlation length of $\langle|H|\rangle$ is of same order of magnitude as the number of iterations; no such stopping criterion is available.
- We compare our method with the Bayesian software of Neal,[1] and also for that algorithm the correlation length was found unreliable as a stopping criterion.

Replication histograms



Histograms showing the number of replicas produced by each network. (a) Start of training, and the competition is low. (b) End of annealing; the energy is still decreasing, and some networks are very much better than the others. (c) End of training; all networks are spread out over some of the best regions, and no network is greatly favoured.

Experiments

The Population method is compared against the Bayesian software of Neal, and against a simple ANN ensemble created by training 20 networks with randomly initialized weights.

Data set	**Robot Arm**	**Pima**	**Scintigram**
Problem type	Regression	Class.	Class.
N inputs	2	7	30
N targets	2	1	1
Train. set size	200	200	153
Test set size	200	332	76
	Mean Sq. Error	ROC area (%)	ROC area (%)
Population	0.00309± 0.00007	86.51± 0.04	80.8± 1.0
Bayesian	0.00279± 0.00002	86.30± 0.13	80.6± 0.7
Simple ensemble	0.00362± 0.00008	86.10± 0.06	75.5± 0.1
	CPU minutes	CPU minutes	CPU minutes
Population	435	19	93
Bayesian	58	79	386
Simple ensemble	0.8	0.6	0.6

The Population method is faster and has the same predictive performance as the Bayesian method on the classification tasks, but performs slightly worse on the Robot Arm task. Both methods outperform the simple ensemble, which is however the fastest.

Reference

1. Neal, R. M. (1999). Software for Flexible Bayesian Modeling. http://www.cs.toronto.edu/~radford/fbm.1999-03-13.doc/index.html