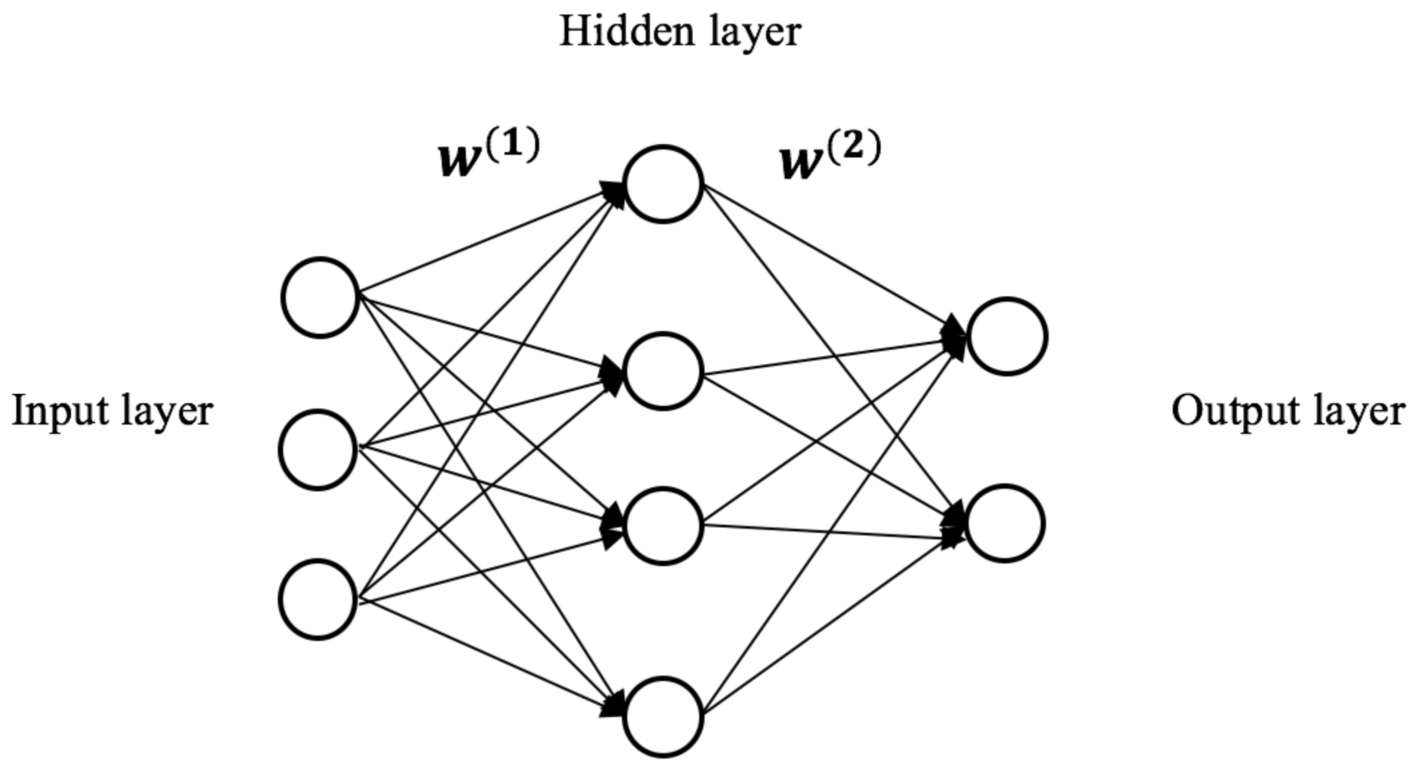
Fitting Binarized Neural Networks

Virginio Giacomo

Knowledge Representation and Learning
Data Science Master Degree
Università degli Studi di Padova

Binarized Neural Networks



Neural networks where inputs, outputs and weights belong to {-1,1}.

The objective is to find the set of wights that maximizes the classification accuracy.

Objectives

Fit a binarized neural network using SAT following these steps:

- Write a program that implements a fully connected layer of a binarized neural network with *n* elements in input and *m* in output;
- Propose a binary function $f : \{-1, 1\}^n \rightarrow \{-1, 1\}^m$ and use it to generate a set of training data;
- Train a binarized neural network composed of two fully connected layers *n* nodes, *h* intermediate and *m* output nodes;
- Generate test data and evaluate the network on different *n*’s.

Hard Costraint of the MaxSAT

- The hard costraint of the MaxSAT are based on the training dataset, assigning to each $x_i^{(n)}$ an index in SAT, then assigning the hard costraint of the index to be TRUE if $x_i^{(j)} = 1$, and FALSE if $x_i^{(j)} = -1$.

SAT indexes

- The formula used to obtain the index for inputs is:
  `i * N + n + 1`
- The formula used to obtain the index for the first set of weights is:
  `len(train)* N +(n*P)+ p + 1`
- The formula used to obtain the index for the second set of weights is:
  `len(train)* N +(N*P)+(p*M)+ m + 1`

Where len(train) is the *length* of the training set, i is an **input**, N is the *number of input neurons*, n is an *input neuron*, P is the *number of hidden neurons*, p is an *hidden neuron*, M is the *number of output neurons*, m is an *output neuron.*

Activation Function

- $sign(\sum xiwi)$

- It is not doable in propositional logic, so instead we use have 1 as result of the function if $(\#xiw_i > 0) \;\; > \;\; (\#xiw_i < 0)$, and -1 otherwise.

- Since x and w ∈ {-1,1}, $x_i w_i$ in the formula is equivalent to $x_i \equiv wi$.

Activation Function in CNF

- $(\#xiw_i > 0) \;\; > \;\; (\#xiw_i < 0)$ can be interpreted in CNF as:

  "x$_i \equiv wi$ for at least more than half the i's"

- So if i is even x$_i \equiv wi$ for at least i/2+1, if i is odd x$_i \equiv wi$ for at least i/2 rounded up.

- The opposite happens instead, when i is even if x$_i \equiv \neg wi$ for at least i/2, when i is odd x$_i \equiv \neg wi$ for at least i/2 rounded up.

Activation Function in CNF

- As an example, if i=1,2,3 then the activation function will be true if any combination of x and w of two i's, where x and w with same i have opposite truth value are true in such way:

  $(x_1 \lor x_2 \lor \neg w_1 \lor \neg w_2) \land (x_1 \lor \neg x_2 \lor \neg w_1 \lor w_2) \land$
  $(\neg x_1 \lor x_2 \lor w_1 \lor \neg w_2) \land (\neg x_1 \lor \neg x_2 \lor w_1 \lor w_2) \land$

  $(x_1 \lor x_3 \lor \neg w_1 \lor \neg w_3) \land (x_1 \lor \neg x_3 \lor \neg w_1 \lor w_3) \land$
  $(\neg x_1 \lor x_3 \lor w_1 \lor \neg w_3) \land (\neg x_1 \lor \neg x_3 \lor w_1 \lor w_3) \land$

  $(x_2 \lor x_3 \lor \neg w_2 \lor \neg w_3) \land (x_2 \lor \neg x_3 \lor \neg w_2 \lor w_3) \land$
  $(\neg x_2 \lor x_3 \lor w_2 \lor \neg w_3) \land (\neg x_2 \lor \neg x_3 \lor w_2 \lor w_3)$

Activation Function in CNF with hidden layer

- Having an hidden layer adds an exponential grade of complexity, in fact, while a network with an imput layer and a single output can be modeled as in the example before, with the combination of inputs and weights, now an output is determined by the combination of hidden nodes and weights (w^2), and each hidden node is itself determined by the combinations of inputs and weights (w^1).

Activation Function in CNF with hidden layer

To bring an example we can use a similar notation to the first example, with 3 hidden layers, 3 inputs and 1 output, the activation function of the output is 1 if:

$$\begin{array}{l}
(h_1 \lor h_2 \lor \neg w^2{}_1 \lor \neg w^2{}_2) \land \\
(h_1 \lor \neg h_2 \lor \neg w^2{}_1 \lor w^2{}_2) \land \\
(\neg h_1 \lor h_2 \lor w^2{}_1 \lor \neg w^2{}_2) \land \\
(\neg h_1 \lor \neg h_2 \lor w^2{}_1 \lor w^2{}_2) \land \\
(h_1 \lor h_3 \lor \neg w^2{}_1 \lor \neg w^2{}_3) \land \\
(h_1 \lor \neg h_3 \lor \neg w^2{}_1 \lor w^2{}_3) \land \\
(\neg h_1 \lor h_3 \lor w^2{}_1 \lor \neg w^2{}_3) \land \\
(\neg h_1 \lor \neg h_3 \lor w^2{}_1 \lor w^2{}_3) \land \\
(h_2 \lor h_3 \lor \neg w^2{}_2 \lor \neg w^2{}_3) \land \\
(h_2 \lor \neg h_3 \lor \neg w^2{}_2 \lor w^2{}_3) \land \\
(\neg h_2 \lor h_3 \lor w^2{}_2 \lor \neg w^2{}_3) \land \\
(\neg h_2 \lor \neg h_3 \lor w^2{}_2 \lor w^2{}_3)
\end{array}$$

Where you have to substitute the h's with the combinations for which it is true/false: e.g. if h_1 is in the formula it will be substitued by this formula, while if $\neg h_1$ is in, by this formula.

$$\begin{array}{l}
(x_1 \lor x_2 \lor \neg w^1{}_{11} \lor \neg w^1{}_{21}) \land \\
(x_1 \lor \neg x_2 \lor \neg w^1{}_{11} \lor w^1{}_{21}) \land \\
(\neg x_1 \lor x_2 \lor w^1{}_{11} \lor \neg w^1{}_{21}) \land \\
(\neg x_1 \lor \neg x_2 \lor w^1{}_{11} \lor w^1{}_{21}) \land \\
(x_1 \lor x_3 \lor \neg w^1{}_{11} \lor \neg w^1{}_{31}) \land \\
(x_1 \lor \neg x_3 \lor \neg w^1{}_{11} \lor w^1{}_{31}) \land \\
(\neg x_1 \lor x_3 \lor w^1{}_{11} \lor \neg w^1{}_{31}) \land \\
(\neg x_1 \lor \neg x_3 \lor w^1{}_{11} \lor w^1{}_{31}) \land \\
(x_2 \lor x_3 \lor \neg w^1{}_{21} \lor \neg w^1{}_{31}) \land \\
(x_2 \lor \neg x_3 \lor \neg w^1{}_{21} \lor w^1{}_{31}) \land \\
(\neg x_2 \lor x_3 \lor w^1{}_{21} \lor \neg w^1{}_{31}) \land \\
(\neg x_2 \lor \neg x_3 \lor w^1{}_{21} \lor w^1{}_{31})
\end{array}$$

$$\begin{array}{l}
(x_1 \lor x_2 \lor w^1{}_{11} \lor w^1{}_{21}) \land \\
(x_1 \lor \neg x_2 \lor w^1{}_{11} \lor \neg w^1{}_{21}) \land \\
(\neg x_1 \lor x_2 \lor \neg w^1{}_{11} \lor w^1{}_{21}) \land \\
(\neg x_1 \lor \neg x_2 \lor \neg w^1{}_{11} \lor \neg w^1{}_{21}) \land \\
(x_1 \lor x_3 \lor w^1{}_{11} \lor w^1{}_{31}) \land \\
(x_1 \lor \neg x_3 \lor w^1{}_{11} \lor \neg w^1{}_{31}) \land \\
(\neg x_1 \lor x_3 \lor \neg w^1{}_{11} \lor w^1{}_{31}) \land \\
(\neg x_1 \lor \neg x_3 \lor \neg w^1{}_{11} \lor \neg w^1{}_{31}) \land \\
(x_2 \lor x_3 \lor w^1{}_{21} \lor w^1{}_{31}) \land \\
(x_2 \lor \neg x_3 \lor w^1{}_{21} \lor \neg w^1{}_{31}) \land \\
(\neg x_2 \lor x_3 \lor \neg w^1{}_{21} \lor w^1{}_{31}) \land \\
(\neg x_2 \lor \neg x_3 \lor \neg w^1{}_{21} \lor \neg w^1{}_{31})
\end{array}$$

Size of the CNF Problem

- While with no hidden layer and 3 inputs we only needed 12 clauses, now with one hidden layer of 3 nodes we need 12^3 = 1728 clauses.
- The number of clauses with n values in input and p hidden values is:

  $C(n , n/2+1) * 2^{(n/2+1)} * (C(p , p/2+1) * 2^{(p/2+1)})^2$

  Where C(a,b) is the the number of combination of b items over a, x/2 is the integer part of x/2 (hence x/2+1 is x/2 rounded up).
- With 5 inputs and 3 hidden we have 11520 clauses for each output and each batch of inputs.
- With 5 inputs and 5 hidden layers we would have 512000 clauses, which are too many for the system to handle.

MaxSAT problem

- While SAT allows to group clauses in CNF form, MaxSAT, which is used to find the best weights, doesn’t allow such thing.
- The ideal approach would be to put a weight on the whole big CNF for each output and input batch, but as we can’t do it we are forced to create many small soft clauses (of same weight).
- Having to use this approach however might skew results, as the weight of the model depends on how many small clauses are satisfied (for example with i=5, $(\#xiwi > 0) > (\#xiwi < 0)$ is not satisfied both in the cases when $(\#xiwi > 0)$ = 0 and $(\#xiwi > 0)$ = 2, however for $(\#xiwi > 0)$ = 2 some of the small clauses will still be satisfied.

Data generation

- Data is generated by all possible combinations of true/false values for n elements (so we have 2^n batches of n inputs), then it is divided in train and test dataset, with a 65/35% division.
- The following are the formulas used for three outputs for data generated with inputs of size 3, 4 and 5:

n=3
```
(x[0] or x[1]) and (x[2])
x[0] or (x[1] and x[2])
not x[1]
```

n=4
```
((x[0] or x[1]) and (x[2] or x[3])) or (not x[0] and not x[2])
(x[0] and x[1] and x[2]) or (x[3])
not x[1] and not x[3]
```

n=5
```
((x[0] or x[1]) and (x[2] or x[3])) or (x[4] and (not x[0] and not x[2]))
(x[0] and x[1] and x[2]) or (x[3] and x[4])
not x[1] and not x[3] and not x[4]
```

Accuracy and times

- `N=3, Training acc: 0.67, Test acc: 0.67, 1.26s`
- `N=4, Training acc: 0.6, Test acc: 0.39, 21.6s`
- `N=5, Training acc: 0.7, Test acc: 0.72, 309s`