Université de Montréal

Convex Neural Networks

Yoshua Bengio
Nicolas Le Roux
Pascal Vincent
Olivier Delalleau
Patrice Marcotte

Core Ideas

1. Popularity of neural networks suffers from their lack of convexity.

Proposition 1: Let $Q(w \cdot h(x_t), y_t)$ be the cost associated with the network output $w \cdot h(x_t)$ and target y_t, and $\Omega(w)$ the regularization term. If Q is convex in its first argument and if Ω is convex, then the cost function $C = \lambda\Omega(w) + \sum_{t=1}^{n} Q(w \cdot h(x_t), y_t)$ is convex in w.

Consequence of proposition 1: With the usual cost functions, optimizing only the output weights of a neural network is a convex problem.

2. A neural network with all possible hidden units has only its output weights to optimize.

3. Such a network can thus be optimized in a convex way.

3. With only a finite number of non-zero output weights, this becomes an ordinary neural network.

4. A convex neural network is a usual neural network where the number of hidden units is not fixed.

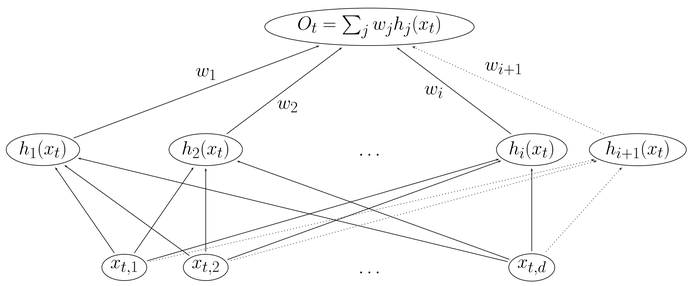
5. If the regularization term is the $\mathcal{L}^1$-norm, then at the optimum, the number of hidden units will be finite and directly depending on the value of λ.

NNets only optimize output weights ?

1. We can see a usual neural net as a network with an infinite number of hidden neurons and the output weights being almost all 0.

2. With this parametrization, optimizing an input weight (v_j becomes $v_{j'}$) is equivalent to optimizing the corresponding output weights ($w_{j'}$ takes the value of w_j and w_j becomes 0).

Incremental procedure



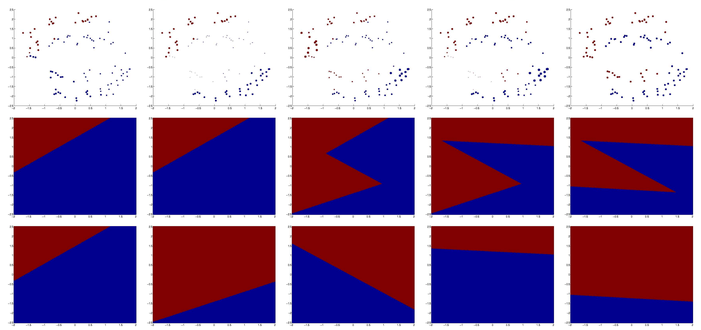
$$q_t = \frac{\partial Q\left(O_t, y_t\right)}{\partial O_t}$$
$$h_j(x_t) = g(x_t, v_j)$$
$$v_{i+1} = \operatorname{argmax}_v \sum_t g(x_t, v) q_t$$

Algorithm ConvexNN$_{\mathcal{L}^1}(D,Q,\lambda,s)$

Input: training set $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, convex loss function Q, and scalar $\mathcal{L}^1$ regularization penalty λ. s is either the *sign* function or the *tanh* function, so that the hidden units can be written $h(x_t) = s(v \cdot \tilde{x}_t)$ with $\tilde{x}_t = (x_t^T, 1)^T$.
(1) Set $i = 1$, $v_1 = (0, 0, \dots, 1)$ and select $w_1 = \operatorname{argmin}_{w_1} \sum_t Q(w_1 s(1), y_t) + \lambda|w_1|$.
(2) Repeat
(3) Set $i = i + 1$.
(4) Let $q_t = Q'(\sum_{j=1}^{i-1} w_j h_j(x_t), y_t)$
(5) If $s = \text{sign}$
(5a) train linear classifier $h_i(x) = \text{sign}(v_i \cdot \tilde{x})$ with examples $\{(x_t, \text{sign}(q_t))\}$ and errors weighted by $|q_t|$, $t = 1 \dots n$ (i.e., *maximize* $\sum_t q_t h_i(x_t)$)
(5b) else ($s = \tanh$)
(5c) train "linear classifier" $h_i(x) = \tanh(v_i \cdot \tilde{x})$ to *maximize* $\sum_t q_t h_i(x_t)$.
(6) If $\sum_t q_t h_i(x_t) < \lambda$ then set $i = i - 1$ and **stop**.
(7) Select $w_1, \dots, w_i$ (and optionally $v_1, \dots, v_i$) minimizing (exactly or approximately) $C = \sum_t Q(\sum_{j=1}^{i} w_j h_j(x_t), y_t) + \lambda \sum_{j=1}^{i} |w_j|$, so that $\frac{\partial C}{\partial w_j} = 0$ for $j = 1 \dots i$.
(8) Return the predictor $\hat{y}(x) = \sum_{j=1}^{i} w_j h_j(x)$.

Theorem 1: If there is no h such that $\sum_t q_t h(x_t) > \lambda$, the optimum is obtained.
Theorem 2: At the optimum, there is no more than $n_{samples} + 1$ hidden units (Rätsch, Demiriz and Bennett, 2001; Hettich and Kortanek, 1993) and this value is controlled by λ.

Example: the double-moon problem



Top: Output value for each sample (color = sign and size = amplitude).
Middle: The decision surface of the network.
Bottom: The optimal linear classifier found at step i.

Exact and approximate minimization

1. The exact algorithm to find the best separating hyperplane is in $O(n^d \log(n))$.

2. Need for an approximate algorithm: linear SVM, logistic regression classifier, variants of the perceptron algorithm, ...

3. Might be stuck in local minimum and thus no guarantee of reaching the global optimum.

4. When a new hidden unit is added, we train the entire neural network for a few iterations.

5. In practice, the test error is very similar to the one obtained with the exact algorithm.

Extension to multiple output neurons

1. A neural network with p output neurons can be represented as p neural networks with 1 output neuron each (even though this is not of great interest).

2. Therefore, the stopping criterion remains (one for each output neuron).

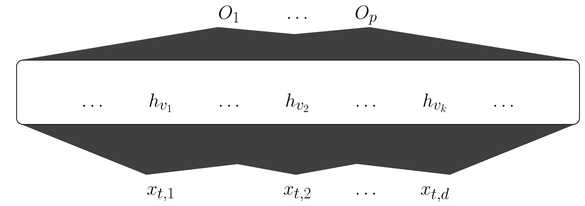
3. If we prefer to have fewer hidden units (and we do), then the function to consider to find the optimal hidden unit is a bit trickier, yet optimizable.

Continuous neural networks: $\mathcal{L}^2 \Rightarrow$ Kernel

1. As it seemed impossible to have all input weights at the same time, we performed in the former algorithm a constructive iterative procedure.

2. With an $\mathcal{L}^2$-norm regularization, it is possible to optimize over all output weights at the same time.

3. It becomes a kernel method whose kernel depends on the transfer function of the hidden units.



$$O(x) = \int_v w(v) h_v(x)\, dv$$

Conclusion

1. Training a NN can be seen as a convex optimization problem.

2. Even though we never limit the number of hidden neurons, the final number will be no more than the number of examples (and less in practice, depending on the value of λ) with $\mathcal{L}^1$ regularization.

3. We proved the existence of an exact algorithm to find the global optimum of a neural net.

4. With an approximate minimization to find the next hyperplane, **continuing to optimize the previous ones** helps getting out of local minima and speeds up convergence.

References

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *J. American Statistical Association*, 76(376):817–823.

Gallant, S. (1990). Perceptron-based learning algorithms.

Hettich, R. and Kortanek, K. (1993). Semi-infinite programming: theory, methods, and applications. *SIAM Review*, 35(3):380–429.

Marcotte, P. and Savard, G. (1992). Novel approaches to the discrimination problem. *Zeitschrift fr Operations Research (Theory)*, 36:517–545.

Muselli, M. (1997). On convergence properties of pocket algorithm. *IEEE Transactions on Neural Networks*, 8:623–629.

Rätsch, G., Demiriz, A., and Bennett, K. P. (2001). Sparse regression ensembles in infinite and finite hypothesis spaces.