Artificial Neural Networks (Gerstner). Exercises for week 2

BackProp

Exercise 1. Derive BackProp

A network with 2 hidden layers followed by an output layer has the output in the final layer (=3rd layer)

$$\begin{aligned}
\hat{y}_n^\mu &= x_n^{(3)} && (1)\\
&= g^{(3)}[a_n^{(3)}] && (2)\\
&= g^{(3)}[\sum_i w_{ni} g^{(2)}[a_i^{(2)}]] && (3)\\
&= g^{(3)}[\sum_i w_{ni}[g^{(2)}[\sum_j w_{ij}^{(2)} x_j^{(1)}]] && (4)\\
&= g^{(3)}[\sum_i w_{ni}[g^{(2)}[\sum_j w_{ij}^{(2)} g^{(1)}(a_j^{(1)})]] && (5)\\
&= g^{(3)}[\sum_i w_{ni}[g^{(2)}[\sum_j w_{ij}^{(2)} g^{(1)}(\sum_k w_{jk}^{(1)} x_k^\mu)]] && (6)
\end{aligned}$$

We use an error function

$$E(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \boldsymbol{w}^{(3)}) = \frac{1}{2} \sum_\mu \sum_n [t_n^\mu - \hat{y}_n^\mu]^2 \quad (7)$$

Calculate the derivative $\partial E / \partial w_{52}^{(1)}$, hence the derivative with respect to the weight connecting the second input to the fifth hidden neuron in the first layer.

Exercise 2. Algorithmic complexity of Backprop

a. Estimate the number of steps for one loop of gradient calculations with Backprop (applying each pattern once) with $P = 2000$ patterns, N_{in} input units, N_{h} hidden units, and $N_{\text{out}} = 5$ output units.

   (To do so, imagine that you have to write a program for Backprop. Sketch the steps that must be taken. Count each multiplication and each summation as one step and each evaluation of the gain function g or its derivative g' as one step. Ignore biases for simplicity.)

   Show that for large networks, the algorithm scales linearly with the number of weights.

b. Let us assume your laptop makes 10^{10} floating point operations per second. What is the minimal time needed for one cycle if $N_{\text{in}} = N_{\text{h}} = 100$?

   Moreover, BackProp often needs more than 10 000 cycles (presentation of each pattern once) to find the minimum - do you want to wait in front of the machine or is there time to have a coffee break?

c. Finally, let us assume that the input consists of pixel patterns with 256x256 pixels. You have two hidden layers of 1000 units each. Calculate the number of time steps for one loop. Also calculate the number of weights.

   Consider now each weight as a free parameter which has to be estimated from the data. How many data samples do you need to estimate (= optimize) all weights of your network?

d. In a realistic application you would have to optimize the number of hidden neurons by testing the performance for different network sizes. For the test you apply previously unknown patterns (which were not used during training) and you take the network which finally performs best. Let us assume that in addition to the 2000 training patterns you have also 2000 test patterns (both of dimension 256x256). How many networks (of what size of hidden neurons) can you train and test in 48 hours?

Exercise 3. Error functions for Backprop

Several researchers have tried to use error functions other than the quadratic error.

a. Convince yourself that a change of the error function only affects the δs of the output layer. More precisely, the formula for the output δs in the algorithm has to be changed, but the formula for the back-propagation step remains the same.

b. To be specific, consider error functions of the type

$$E_p = \sum_\mu \sum_i |t_i^\mu - \hat{y}_i^\mu|^p$$

where the index i runs over all output neurons and $\hat{y}_i = g^{(n)}(a_i)$. What are the output δs for quadratic error ($p = 2$)? What are the δs for linear error ($p = 1$)? Note that you have to keep track of the absolute value signs.

c. Calculate the output δ for the 'cross-entropy' error function which will be introduced next week in the context of maximum likelihood of statistical learning

$$E_{log} = \sum_\mu \sum_i \left[t_i^\mu \log \frac{t_i^\mu}{\hat{y}_i^\mu} + (1 - t_i^\mu) \log \frac{1 - t_i^\mu}{1 - \hat{y}_i^\mu} \right] .$$

Here, $t_i^\mu = 0$ means 'target output feature i not present for input μ' and $t_i^\mu = 1$ means 'feature present'. The outputs $\hat{y}_i^\mu = 1$ can be interpreted as the probability of 'output feature i is present' for input μ. $\hat{y}_i^\mu = 1$ means that the hypothesis 'feature i is present' is definitely true.

d. Assume the cross-entropy on the transfer function

$$g^{(n)}(a) = \frac{1}{2}[1 + \tanh(a)] \tag{8}$$

and show that the δs of the output layer are simply $\delta_i^\mu = 2[t_i^\mu - \hat{y}_i^\mu]$.

e. Relate the gain function g in (d) to the sigmoidal function in class.

Exercise 4. BackProp with Gaussian units in the first layer (From the exam of 2018)

Our data base $(\boldsymbol{x}^\mu, \boldsymbol{t}^\mu)$ has N-dimensional input and N-dimensional target output. We have a network with two hidden layers.

You use a quadratic error function for each patter μ

$$E(\mu) = \sum_{m=1}^{N} [t_m^\mu - \hat{y}_m^\mu]^2,$$

where $\hat{y}_m^\mu = \sum_{k=1}^{K} w_{mk}^{(3)} x_k^{(2)}$.

The second hidden layer has normal sigmoidal units with gain function $g(a)$, while the first hidden layer contains Gaussian basis functions. Thresholds are implicit (by adding an extra unit) and will not be treated explicitly.

Thus

$$x_k^{(2)} = g\big(\sum_{j=1}^{J} w_{kj}^{(2)} \exp[-0.5(\boldsymbol{x}^\mu - \boldsymbol{c}_j)^2] \big),$$

where $\boldsymbol{c}_j$ is the center of the Gaussian of unit j.

Our aim is to calculate the derivative of the error function with respect to the parameters c_{ji}, that is, component i of Gaussian unit j.

A direct calculation with the chain rule yields that the derivative with respect to c_{45} is

$$\frac{dE(\mu)}{dc_{45}} = \sum_{m=1}^{N} [t_m^\mu - \hat{y}_m^\mu] \sum_{k=1}^{K} w_{mk}^{(3)} g'(a_k) w_{k4}^{(2)} (x_5^\mu - c_{45}) \exp[-0.5(\boldsymbol{x}^\mu - \boldsymbol{c}_j)^2]$$

Reorder the terms of the gradient calculation for arbitrary c_{ij} so as to arrive at an efficient backpropagation algorithm with a forward pass and a backward pass and an update step for the parameters c_{ij}. Summarize your results in pseudo-code.