Machine Learning—Tutorial Notes

Feed Forward Neural Networks # 2

1 the vanishing gradient

Problem 1: Why does the gradient vanish?

2 Regularisation

Problem 2: How can we prevent overfitting?

Problem 3: A convolutional network is, in fact, a neural network where weights are shared. After all, mapping a filter at various "places" of the image can be seen as multiplying many neurons, each having the same input weights.

Consider a neural network in which multiple weights are constrained to have the same value. Discuss how the standard backpropagation algorithm must be modified in order to ensure that such constraints are satisfied when evaluating the derivatives of an error function with respect to the adjustable parameters in the network.

Problem 4: This exercise is taken mostly from [1], chapter 5.5.7. One way to reduce the effective complexity of a network with a large number of weights is to constrain weights within certain groups to be equal (*weight sharing*). However, this is only applicable to particular problems in which the form of the constraints can be specified in advance. Here we consider a form of *soft weight sharing* [2] in which the hard constraint of equal weights is replaced by a form of regularisation in which groups of weights are encouraged to have similar values. The division of weights into groups, the mean weight value for each group, and the spread of values within the groups are all determined as part of the learning process.

In general, we consider the distribution over the weights to be a *mixture of Gaussians*, i.e., for a given weight w_i:

$$p(w_i) = \sum_{j=1}^{N} \pi_j \mathcal{N}(w_i|\mu_j, \sigma_j^2)$$

(π_j are called the *mixing coefficients*) and weights are i.i.d:

$$p(\boldsymbol{w}) = \prod_i p(w_i)$$

Taking the negative logarithm then leads to a regularisation function of the form

$$\Omega(\boldsymbol{w}) = -\sum_i \ln \left(\sum_{j=1}^{M} \pi_j \mathcal{N}(w_i|\mu_j, \sigma_j^2) \right)$$

The total error function is thus given by

$$\widetilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w})$$

where $E(\boldsymbol{w})$ is the usual standard error function (sum-of-squares, cross-entropy, ...) and λ is the regularisation coefficient. This error is minimised both with respect to the weights w_i and with respect to the parameters $\{\pi_j, \mu_j, \sigma_j\}$ of the mixture model. In order to evaluate the derivatives of the error function with respect to these parameters, it is convenient to regard the $\{\pi_j\}_j$ as prior probabilities and to introduce the corresponding posterior probabilities which are in the form (and you should prove that)

$$\gamma_j(w) = \frac{\pi_j \mathcal{N}(w|\mu_j, \sigma_j^2)}{\sum_k \pi_k \mathcal{N}(w|\mu_k, \sigma_k^2)} \tag{1}$$

Now, prove the following:

$$\frac{\partial \widetilde{E}}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda \sum_j \gamma_j(w_i) \frac{(w_i - \mu_j)}{\sigma_j^2} \tag{2}$$

$$\frac{\partial \widetilde{E}}{\partial \mu_j} = \lambda \sum_i \gamma_j(w_i) \frac{(\mu_j - w_i)}{\sigma_j^2} \tag{3}$$

$$\frac{\partial \widetilde{E}}{\partial \sigma_j} = \lambda \sum_i \gamma_j(w_i) \left(\frac{1}{\sigma_j} - \frac{(w_i - \mu_j)^2}{\sigma_j^3} \right) \tag{4}$$

Note that in a practical implementation, new variables η_j defined by

$$\sigma_j^2 = \exp(\eta_j)$$

would be introduced, to ensure that σ_j remains positive, and thus the minimisation must be performed with respect to the η_j. Similarly, the mixing coefficients π_j are expressed in terms of a set of auxiliary variables $\{\phi\}_j$ using the softmax function given by

$$\pi_j = \frac{\exp(\phi_j)}{\sum_k \exp(\phi_k)}$$

The derivatives thus take the form (prove this too)

$$\frac{\partial \widetilde{E}}{\partial \phi_j} = \sum_i (\pi_j - \gamma_j(w_i)) \tag{5}$$

References

[1] C. M. Bishop: Pattern Recognition and Machine Learning. Springer, 2006.

[2] S. Nowlan and G Hinton: Simplifying neural networks by soft weight sharing. Neural Computation 4(4), 473-493, 1992.