Machine Learning Worksheet 4

Feed Forward Neural Networks

1 Activation functions

Problem 1. Consider a two-layer network function of the form in which the hidden-unit nonlinear activation functions $g(\cdot)$ are given by logistic sigmoid functions of the form

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Show that there exists an equivalent network, which computes exactly the same function, but with hidden unit activation functions given by $\tanh(x)$.

Problem 2. Show that the derivative of the logistic sigmoid activation function can be expressed in terms of the function value itself. Also derive the corresponding result for the tanh activation function.

2 Multiple targets

Problem 3. If we have multiple target variables, and we assume that they are independent conditional on $\boldsymbol{x}$ and $\boldsymbol{w}$ with shared noise precision β then the conditional distribution of the target values is given by

$$p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}\boldsymbol{I})$$

Show that maximizing the resulting likelihood function under the above conditional distribution for a multioutput neural network is equivalent to minimizing a sum-of-squares error function.

Problem 4. Consider a regression problem involving multiple target variables in which it is assumed that the distribution of the targets, conditioned on the input vector $\boldsymbol{x}$, is a Gaussian of the form

$$p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{\Sigma})$$

where $\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w})$ is the output of a neural network with input vector $\boldsymbol{x}$ and a weight vector $\boldsymbol{w}$, and $\boldsymbol{\Sigma}$ is the covariance of the assumed Gaussian noise on the targets. Given a set of independent observations of $\boldsymbol{x}$ and $\boldsymbol{t}$, write down the error function that must be minimized in order to find the maximum likelihood solution for $\boldsymbol{w}$, if we assume that $\boldsymbol{\Sigma}$ is fixed and known. Now assume that $\boldsymbol{\Sigma}$ is also to be determined from the data and write down an expression for the maximum likelihood solution for $\boldsymbol{\Sigma}$. Note that the optimizations of $\boldsymbol{w}$ and $\boldsymbol{\Sigma}$ are now coupled, in contrast to the case of independent target variables discussed in the exercise above.

3 Error functions

Problem 5. Show that maximizing likelihood for a multiclass neural network model in which the network outputs have the interpretation $y_k(\boldsymbol{x}, \boldsymbol{w}) = p(t_k = 1|\boldsymbol{x})$ is equivalent to the minimization of the cross-entropy error function.

Problem 6. Show the derivative of the error function

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)$$

(y_n denotes $y(\boldsymbol{x}_n, \boldsymbol{w})$) with respect to the activation a_k for an output unit having a logistic sigmoid activation function satisfies

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

Problem 7. Show the derivative of the standard multiclass error function

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(\boldsymbol{x}_n, \boldsymbol{w})$$

with respect to the activation a_k for output units having a softmax activation function satisfies

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

4 Robust classification

Problem 8. Consider a binary classification problem in which the target values are $t \in \{0, 1\}$, with a network output $y(\boldsymbol{x}, \boldsymbol{w})$ that represents $p(t = 1|\boldsymbol{x})$, and suppose that there is a probability ε that the class label on a training data point has been incorrectly set. Assuming independent and identically distributed data, write down the error function corresponding to the negative log likelihood. Verify that the well known error function for binary classification is obtained when $\varepsilon = 0$. Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function.

5 Regularization

Problem 9. $\star$ This exercise is taken mostly from [**?**], 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* [**?**] in which the hard constraint of equal weights is replaced by a form of regularization 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 regularization 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 regularization coefficient. This error is minimized 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, proof 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 minimization 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.