Ch 5: Neural Networks

ifding

Abstract

Feed-forward Network, Network Training, Mixture Density Networks, Bayesian Neural Networks

1. Feed-forward Network

First we construct M linear combinations of the input variables $x_1, \dots, x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{1}$$

where $j = 1, \dots, M$, and the superscript (1) indicates that the corresponding parameters are in the first 'layer' of the network. Each of them is transformed using a differentiable, nonlinear *activation function* $h(\cdot)$ to give

$$z_j = h(a_j) \tag{2}$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{3}$$

where $k = 1, \dots, K$, and K is the total number of outputs.

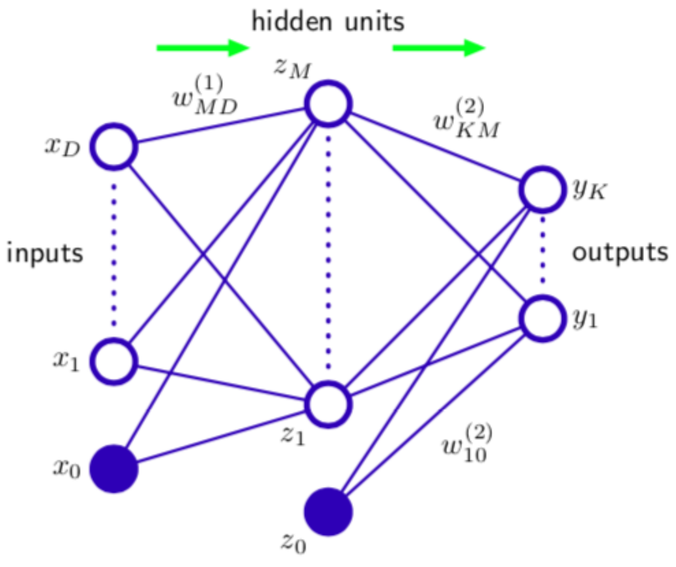

Figure 1: Two-layer neural network.

As shown in Figure 1, for binary classification problems,

$$y_k = \sigma(a_k) \tag{4}$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{5}$$

We can absorb the biases into the layers' weights,so

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right) \tag{6}$$

2. Network Training

A simple approach to the problem of determining the network parameters is to minimize a sum-of-squares error function. Given a data set of N independent, identically distributed observations $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ and target vectors $\mathbf{t} = \{\mathbf{t}_n\}$, where $n = 1, \dots, N$, we can construct the corresponding likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} p\left(t_n | \mathbf{x}_n, \mathbf{w}, \beta\right) \tag{7}$$

Taking the negative logarithm, we obtain the error function

$$\frac{\beta}{2} \sum_{n=1}^{N} \left\{y\left(\mathbf{x}_n, \mathbf{w}\right) - t_n\right\}^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln(2\pi) \tag{8}$$

Maximizing the likelihood function is equivalent to minimizing the sum-of-squares error function given by

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{y\left(\mathbf{x}_n, \mathbf{w}\right) - t_n\right\}^2 \tag{9}$$

where we have discarded additive and multiplicative constants. In practice, the nonlinearity of the network function $y(\mathbf{x}_n, \mathbf{w})$ causes the error $E(\mathbf{w})$ to be nonconvex.

The conditional distribution of targets given inputs is then a Bernoulli distribution of the form

$$p(t|\mathbf{x}, \mathbf{w}) = y(\mathbf{x}, \mathbf{w})^t \{1 - y(\mathbf{x}, \mathbf{w})\}^{1-t} \tag{10}$$

The error function, which is given by the negative log likelihood, is then a *cross-entropy* error function of the form

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\} \tag{11}$$

where y_n denotes $y(\mathbf{x}_n, \mathbf{w})$.

Because the error $E(\mathbf{w})$ is a smooth continuous function of w, its smallest value will occur at a point in weight space such that the gradient of the error function vanishes, so that

$$\nabla E(\mathbf{w}) = 0 \tag{12}$$

as otherwise we could make a small step in the direction of $-\nabla E(\mathbf{w})$ and thereby further reduce the error.

The simplest approach to using gradient information is to choose the weight update to comprise a small step in the direction of the negative gradient, so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E\left(\mathbf{w}^{(\tau)}\right) \tag{13}$$

where the parameter $\tau > 0$ is known as the *learning rate*.

3. Mixture Density Networks

The goal of supervised learning is to model a conditional distribution $p(\mathbf{t}|x)$, which for many simple regression problems is chosen to be Gaussisan, However, practical machine learning problems can often have significantly non-Gaussian distributions.

mixture density network, for any given value of $\mathbf{x}$, the mixture model provides a general formalism for modelling an arbitrary conditional density function $p(\mathbf{t}|\mathbf{x})$.

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) \mathcal{N}\left(\mathbf{t}|\boldsymbol{\mu}_k(\mathbf{x}), \sigma_k^2(\mathbf{x})\right) \tag{14}$$

We now take the various parameters of the mixture model, namely the mixing coefficients $\pi(\mathbf{x})$, the means $\boldsymbol{\mu}_k(\mathbf{x})$, and the variances $\sigma_k^2(\mathbf{x})$, to be governed by the outputs of a conventional neural network that takes $\mathbf{x}$ as its input.

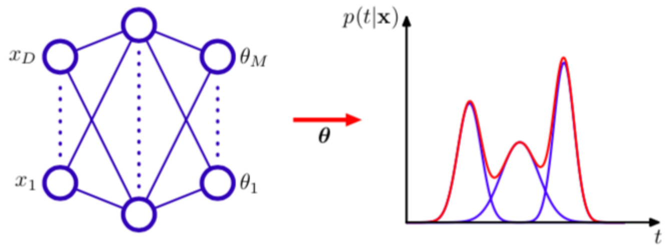

Figure 2: The mixture density network

The neural network in Figure 2 can, for example, be a two-layer network having sigmoidal ('tanh') hidden units. If there are K components in the mixture model, and if $\mathbf{t}$ has L components, then the network will have K output unit activations denoted by a_k^π that determine the mixing coefficients $\pi_k(\mathbf{x})$, K outputs denoted by a_k^σ that determine the kernel widths $\sigma_k(\mathbf{x})$, ($\sigma_k(\mathbf{x})$ is scalar ?), $L \times K$ outputs denoted by a_{kj}^μ that determine the components $\mu_{kj}(\mathbf{x})$ of the kernel centres $\boldsymbol{\mu}_k(\mathbf{x})$. The total number of network outputs is given by $(L + 2)K$.

The mixing coefficients must satisfy the constraints

$$\sum_{k=1}^{K} \pi_k(\mathbf{x}) = 1, \quad 0 \leqslant \pi_k(\mathbf{x}) \leqslant 1 \tag{15}$$

which can be achieved using a set of softmax outputs

$$\pi_k(\mathbf{x}) = \frac{\exp\left(a_k^\pi\right)}{\sum_{l=1}^{K} \exp\left(a_l^\pi\right)} \tag{16}$$

The variances must satisfy $\sigma_k^2(\mathbf{x}) \geq 0$ and so can be represented in terms of the exponentials

$$\sigma_k(\mathbf{x}) = \exp\left(a_k^\sigma\right) \tag{17}$$

Finally, the means $\boldsymbol{\mu}_k(\mathbf{x})$ can be represented directly by the network output activations

$$\mu_{kj}(\mathbf{x}) = a_{kj}^\mu \tag{18}$$

For independent data, the error function takes the form

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k\left(\mathbf{x}_n, \mathbf{w}\right) \mathcal{N}\left(\mathbf{t}_n | \boldsymbol{\mu}_k\left(\mathbf{x}_n, \mathbf{w}\right), \sigma_k^2\left(\mathbf{x}_n, \mathbf{w}\right) \mathbf{I}\right) \right\} \tag{19}$$

where we have made the dependencies on $\mathbf{w}$ explicit. The derivatives of the error $E(\mathbf{w})$ with respect to the components of $\mathbf{w}$ can be evaluated by using the standard back-propagation procedure.

4. Bayesian Neural Networks

Consider the problem of predicting a single continuous target variable t from a vector $\mathbf{t}$ of inputs. We shall suppose that the conditional distribution $p(t|\mathbf{x})$ is Gaussian, with an $\mathbf{x}-$dependent mean given by the output of a neural network model $y(\mathbf{x}, \mathbf{w})$, and with precision (inverse variance) β.

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}\right) \tag{20}$$

Choose a prior distribution over the weights $\mathbf{w}$ that is Gaussian of the form

$$p(\mathbf{w}|\alpha) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}\right) \tag{21}$$

For an i.i.d. data set of N observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$, with a corresponding set of target values $\mathcal{D} = \{t_1, \ldots, t_N\}$, the likelihood function is given by

$$p(\mathcal{D}|\mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|y\left(\mathbf{x}_n, \mathbf{w}\right), \beta^{-1}\right) \tag{22}$$

and so the resulting posterior distribution is then

$$p(\mathbf{w}|\mathcal{D}, \alpha, \beta) \propto p(\mathbf{w}|\alpha)p(\mathcal{D}|\mathbf{w}, \beta) \tag{23}$$

which, as a consequence of the nonlinear dependence of $y(\mathbf{x}, \mathbf{w})$ on $\mathbf{w}$, will be non-Gaussian.

The marginal likelihood, or evidence, for the hyperparameters α and β are obtained by integrating over the network weights

$$p(\mathcal{D}|\alpha, \beta) = \int p(\mathcal{D}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)\mathrm{d}\mathbf{w} \tag{24}$$

References

[1] Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.