Statistical Machine Learning 2018

Exercises, week 10

23 November 2018

TUTORIAL

Exercise 1 – Classification with Labeling Errors

(See Exercise 5.4 in Bishop) Consider a binary classification problem in which the target values are $t \in \{0, 1\}$, with a network output $y(\mathbf{x}, \mathbf{w})$ that represents $p(t = 1|\mathbf{x}, \mathbf{w})$.

1. Let us consider a training set of independent and identically distributed observations $\mathcal{D} = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), ..., (\mathbf{x}_N, t_N)\}$. Derive the expression for the negative log-likelihood of this data set and show that it is equivalent to the *cross-entropy* error function defined in Bishop, Equation (5.21).

2. In the expression of the *cross-entropy* error function there is no analogue for the target noise precision β, which we have previously encountered in Bayesian linear regression (Chapter 3). This is because the target values $\mathbf{t}$ are assumed to be correctly labeled. Suppose now that there is a probability ϵ that the class label on a training data point has been incorrectly set. If we denote the true label of the data point corresponding to input $\mathbf{x}_n$ by t_n and the given label by $\tilde{t}_n$, then show that the probability of a single 'noisy' observation can be written as

$$p(\tilde{t}_n|\mathbf{x}_n, \mathbf{w}) = y_n^{\tilde{t}_n}(1 - y_n)^{1-\tilde{t}_n}(1 - \epsilon) + y_n^{1-\tilde{t}_n}(1 - y_n)^{\tilde{t}_n}\epsilon.$$

   Hint: Use the same expression for the probability of a correctly labeled data point and then relate the true label to the noisy (given) label.

3. Show that the previous probability can be written equivalently as

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

   where $\tilde{y}_n = \tilde{y}(\mathbf{x}_n, \mathbf{w}, \epsilon) = y_n + \epsilon(1 - 2y_n)$. Then, assuming again independent and identically distributed data, write down the error function corresponding to the negative log-likelihood. Verify that the obtained error function is just the *cross-entropy* function when $\epsilon = 0$.

4. The error function obtained for noisy observations makes the model robust to incorrectly labeled data, in contrast to the usual *cross-entropy* error function. To see this, consider the case when our network function $y(\mathbf{x}, \mathbf{w}) = p(t|\mathbf{x}, \mathbf{w})$ (not $\tilde{y}$) takes the values one and zero, i.e., when we are certain that the data point belongs to one class or the other. What is the corresponding value for $\tilde{y}(\mathbf{x}, \mathbf{w})$ and how does it account for the error in labeling? What happens to $\tilde{y}$ when $y(\mathbf{x}, \mathbf{w}) = 0.5$, which indicates maximum uncertainty in the classification?

Exercise 2

We look at backpropagation in a neural network. (see §5.3) In figure 1 a two-layer neural network is depicted with one input node x, one output node y and three hidden nodes z_1, z_2 and z_3. In addition two fixed value bias nodes have been included, $x_0 = 1$ and $z_0 = 1$. The links between them represent weight parameters, with $w_3^{(1)}$ indicating a weight between x and node z_3, and with $b_1^{(2)}$ representing a weight between bias node z_0 and node y. (*Note:* the distinction between 'normal' and bias nodes/weights is purely for notational purposes.) The network is used in a regression problem to learn the function $f(x) = x^2$. The output node has a linear activation function ($y = \sum_j w_j^{(2)} z_j + b_1^{(2)}$), and the z_j have $\tanh(\cdot)$ activation functions.

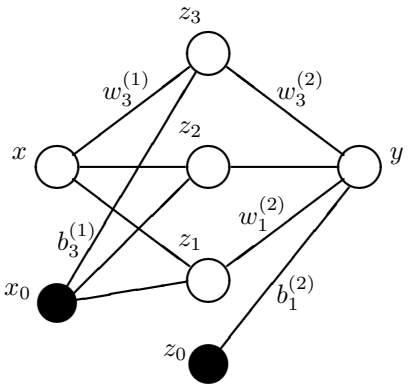

Figure 1: Neural network, 3 hidden nodes

After initialization of the weights, training of the network essentially involves three steps

- Propagate new input value x through network to calculate output y (eq. 5.48/49)
- Backpropagate error $\delta = (y - t)$ to establish the derivatives $\frac{\partial E}{\partial w_{ji}}$ (eq. 5.53/56)
- Use the estimated derivatives in gradient descent (or similar) to update weights (eq. 5.43)

and repeat until convergence.

We want to study how the network adapts to new input/output data through backpropagation.

1. Why is it not a good idea to start with a network in which all the weights have been initialized to zero?

The weights are set to $\mathbf{w}^{(1)} = [1, 0.1, -1]$, $\mathbf{w}^{(2)} = [-1, 0.1, -1]$ and for the bias $\mathbf{b}^{(1)} = [1, 0, 1]$ and $\mathbf{b}^{(2)} = [2]$. After this initialization, the first point offered to the network is $\{x_1, t_1\} = \{0.5, 0.25\}$.

2. Calculate one full training cycle of the network for this datapoint. Assume a sum-of-squares error function $E(\mathbf{w})$, and use a learning rate parameter of $\eta = 0.5$ in updating the weights. Check whether or not the updated network has improved the output.

In this regression network the output node y has a linear activation function. If we wanted to do classification we would choose a sigmoidal function. However, this freedom in the choice of activation function does not apply to the hidden nodes z_i.

3. Describe why the nonlinearity of the activation functions for the hidden nodes is crucial for a network to learn more than mainly trivial regression or classification problems.
   Hint: Consider what regression/classification such a network would be able to perform.

In a sequential learning procedure, each time one of the points is selected at random and processed through one full training cycle of the network. Batch mode takes all available data points at once before making an update and repeating the process. Suppose the entire data set consists of 50 data points $\{x_i, t_i\}$ sampled over the interval $[-1, 1]$ from a noisy quadratic function: $t = x^2 + \epsilon$, with $\epsilon \sim \mathcal{N}(\epsilon|0, 0.2)$.

4. Will the sequential procedure converge for this data set? If so, is the final weight distribution the same as when a batch procedure would have been used? If not, does batch mode updating suffer from the same drawback? Explain your answer.

BONUS PRACTICE

Exercise 3

In neural networks, the nonlinear activation functions $h(\cdot)$ for the hidden units are generally chosen to be 'S-shaped' functions, such as the logistic sigmoid $\sigma(a)$ and the $\tanh(a)$ function, where

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

$$\tanh(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)} \tag{2}$$

1. Show that $\tanh(a)$ is just a stretched, squashed and shifted sigmoid:

$$\tanh(a) = 2\sigma(2a) - 1$$

2. Show that the derivative $h'(a)$ of the $\tanh(\cdot)$ activation function can be expressed in terms of the function itself:

$$\frac{d\tanh(a)}{da} = 1 - \tanh^2(a) \tag{3}$$

3. Create a Taylor expansion of (2) to argue that for small a, i.e. $a \approx 0$, the tanh activation function is essentially linear, and well approximated by

$$\tanh(a) \approx a$$

Exercise 4

(Exercise 5.25 in Bishop)

Consider a quadratic error function of the form:

$$E(\mathbf{w}) = E_0 + \frac{1}{2}(\mathbf{w}-\mathbf{w}^*)^{\mathrm{T}}\mathbf{H}(\mathbf{w}-\mathbf{w}^*), \tag{4}$$

where $\mathbf{w}^*$ minimizes the error function, and the Hessian matrix $\mathbf{H}$ is positive definite and constant. Suppose the initial weight vector $\mathbf{w}^{(0)}$ is chosen to be at the origin and is updated using simple gradient descent:

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \rho\nabla E \tag{5}$$

where τ denotes the step number, and ρ is the learning rate, which is assumed to be small. Show that, after τ steps, the components of the weight vector parallel to the eigenvectors of $\mathbf{H}$ can be written as:

$$w_j^{(\tau)} = \left[1 - (1-\rho\eta_j)^{\tau}\right] w_j^*, \tag{6}$$

where $w_j = \mathbf{w}^{\mathrm{T}}\mathbf{u}_j$, and $\mathbf{u}_j$ and η_j are the eigenvectors and eigenvalues, respectively, of $\mathbf{H}$, so that

$$\mathbf{H}\mathbf{u}_j = \eta_j\mathbf{u}_j. \tag{7}$$

Show that as $\tau \to \infty$, this gives $\mathbf{w}^{(\tau)}$ as expected, provided $|1-\rho\eta_j| < 1$. Now suppose that training is halted after a finite number τ of steps. Show that the components of the weight vector parallel to the eigenvectors of the Hessian satisfy

$$w_j^{(\tau)} \simeq w_j^* \quad \text{when} \quad \eta_j \gg (\rho\tau)^{-1},$$

$$|w_j^{(\tau)}| \ll |w_j^*| \quad \text{when} \quad \eta_j \ll (\rho\tau)^{-1}.$$

Show that $(\rho\tau)^{-1}$ is analogous to the simple weight decay regularization parameter λ (see Equation (5.112) in Bishop).