Learning from Data, Tutorial Sheet for Week 9: Answers

School of Informatics, University of Edinburgh

Instructor: Amos Storkey

- **Question 1: Derivatives**
  We have a layered neural network model here, with one hidden layer. The model is defined as

$$p(c=1|\mathbf{x}) = \sigma(b_0 + v_1 g(\mathbf{w}_1^T\mathbf{x} + b_1) + v_2 g(\mathbf{w}_2^T\mathbf{x} + b_2))$$

  To help visualise what this model is actually doing, draw yourself a diagram of the neural network.
  If we let $\sigma^\mu = \sigma(b_0 + v_1 g(\mathbf{w}_1^T\mathbf{x}^\mu + b_1) + v_2 g(\mathbf{w}_2^T\mathbf{x}^\mu + b_2))$. then using the i.i.d assumption we write the log likelihood as

$$L(\mathbf{x};\boldsymbol{\theta}) = \sum_{\mu=1}^{P} c^\mu \log(\sigma^\mu) + (1-\sigma^\mu)\log(1-\sigma^\mu)$$

  for parameters $\boldsymbol{\theta}$.
  Note that $\sigma(b_0 + v_1 g(\mathbf{w}_1^T\mathbf{x} + b_1) + v_2 g(\mathbf{w}_2^T\mathbf{x} + b_2))$ can be written as

$$\sigma(\mathbf{v}^T\boldsymbol{\phi}(\mathbf{x}))$$

  where $\mathbf{v}^T = (b_0, v_1, v_2)$ and

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} 1 \\ g(\mathbf{w}_1^T\mathbf{x} + b_1) \\ g(\mathbf{w}_2^T\mathbf{x} + b_2) \end{pmatrix}$$

  In other words the mapping from the hidden units to the output is a linear parameter model, and we have seen from an earlier tutorial that for such a model

$$\nabla_{\mathbf{v}} L = \sum_{\mu=1}^{P} [c^\mu - \sigma(\mathbf{v}^T\boldsymbol{\phi}(\mathbf{x}^\mu))]\boldsymbol{\phi}(\mathbf{x}^\mu)$$

  Hence we can write out the derivatives using this directly

$$\frac{\partial L}{\partial b_0} = \sum_{\mu=1}^{P} [c^\mu - \sigma(\mathbf{v}^T\boldsymbol{\phi}(\mathbf{x}^\mu))]1$$

$$\frac{\partial L}{\partial v_1} = \sum_{\mu=1}^{P} [c^\mu - \sigma(\mathbf{v}^T\boldsymbol{\phi}(\mathbf{x}^\mu))]g(\mathbf{w}_1^T\mathbf{x}^\mu + b_1)$$

$$\frac{\partial L}{\partial v_2} = \sum_{\mu=1}^{P} [c^\mu - \sigma(\mathbf{v}^T\boldsymbol{\phi}(\mathbf{x}^\mu))]g(\mathbf{w}_2^T\mathbf{x}^\mu + b_2)$$

  For the parameters of the input to hidden mapping we need to use the chain rule. If we have a model of the form $p(c=1|xB;\boldsymbol{\theta}) = \phi(h(\mathbf{x};\boldsymbol{\theta}))$ the derivatives are of the form

$$\frac{\partial L}{\partial \theta_i} = \sum_{\mu=1}^{P} [c^\mu - \sigma(h(\mathbf{x};\boldsymbol{\theta}))]\frac{\partial h}{\partial \theta_i}$$

Hence

$$\nabla_{\mathbf{w}_1} L = -\sum_{\mu=1}^{P} (c^\mu - \sigma^\mu) v_1 g'(w_1^T x^\mu + b_1) x^\mu$$

$$\nabla_{\mathbf{w}_2} L = -\sum_{\mu=1}^{P} (c^\mu - \sigma^\mu) v_2 g'(w_2^T x^\mu + b_2) x^\mu$$

$$\frac{\partial L}{\partial b_1} = -\sum_{\mu=1}^{P} (c^\mu - \sigma^\mu) v_1 g'(w_1^T x^\mu + b_1)$$

$$\frac{\partial L}{\partial b_2} = -\sum_{\mu=1}^{P} (c^\mu - \sigma^\mu) v_2 g'(w_2^T x^\mu + b_2)$$

- **Question 1 Part 2: Relation with Logistic Regression**
  Logistic Regression could be thought of as a special case of the model we have here, where the function g is linear, in which case the input and hidden layers collapse to one (that is, no hidden layer), because a linear function of a linear function is just another linear function, and we therefore have exactly logistic regression. What the more general model is doing is taking a linear combination of *nonlinear* functions of the input: see Equation 1.

- **Question 1 Part 3: Decision Boundary**
  The decision boundary for this model is given by the equation

  $$b_0 + v_1 g(\mathbf{w}_1^T \mathbf{x} + b_1) + v_2 g(\mathbf{w}_2^T \mathbf{x} + b_2) = 0$$

  and therefore the decision boundary depends on the choice of g. In the case here g is a nonlinear function of the input, and therefore the decision boundary is nonlinear (compare again with logistic regression, where the decision boundary is linear).

- **Question 2: Softmax/Logit model**
  The softmax model is

  $$p(c = i|x) = \frac{\exp(f(x, i))}{\sum_{j=1}^{C} \exp(f(x, j))}$$

  where $f(x, i)$ is some function of the input, whose parameters depend on the class i. If we similarly write the logistic regression model (with 2 classes) as

  $$p(c = 1|x) = \sigma(g(x)) = \frac{1}{1 + \exp(-g(x))}$$

  where $g(x)$ is some other function of the input, then we can equate the two models, and get an expression for $g(x)$ in terms of $f(x, 1)$ and $f(x, 2)$. Doing this we find that $g(x) = f(x, 1) - f(x, 2)$.
  So if we had a softmax model with 2 components $f(x, i) = \mathbf{w}_i^T \mathbf{x} + b_i,\ i = 1, 2$, then we could rewrite it as a logistic regression model with $g(x) = (\mathbf{w}_1 - \mathbf{w}_2)^T \mathbf{x} + (b_1 - b_2)$, and then only have to train one set of parameters, instead of two (because we know that we can get $p(c = 2|\mathbf{x})$ from $p(c = 1|\mathbf{x})$).

- **Question 2 Part 2: Bag of Words**
  The vector representation needs to have a component for each word that one wishes to capture, so setting up this vector is critical for getting any meaningful results. If one knows about words that will not help in distinguishing between the classes, such as "stop words", it makes sense to leave them out of the model. The problem with the "bag of words" technique is that it only takes into account frequencies of words, and ignores any other linguistic information that might be useful in

these documents (like grammar). Also, word dependencies (such as ordering) are ignored. Despite these problems this approach works reasonably well in general. Training this model would be similar to training a logistic regression model, where one would need to train a set of parameters for each class.