Learning from Data, Tutorial Sheet Week 7 - Answers

School of Informatics, University of Edinburgh

1. **Naive Bayes Classifier; Decision Boundary**

   Classification Boundary: $\log p(c=1|x^*) > \log p(c=0|x^*)$

   After the definition of the classifier and using the binary encoding $x_i \in \{0,1\}$, we get:

   $\sum_i \{x_i^* \log \theta_i^1 + (1-x_i^*)\log(1-\theta_i^1)\} + \log p_1 > \sum_i \{x_i^* \log \theta_i^0 + (1-x_i^*)\log(1-\theta_i^0)\} + \log p_0$

   $\sum_i \left\{x_i^*(\log \theta_i^1 - \log \theta_i^0) + (1-x_i^*)\left(\log(1-\theta_i^1) - \log(1-\theta_i^0)\right)\right\} + \log p_1 - \log p_0 > 0$

   $\sum_i \left\{x_i^* \log \frac{\theta_i^1}{\theta_i^0} - x_i^* \log \frac{1-\theta_i^1}{1-\theta_i^0}\right\} + \sum_i \log \frac{1-\theta_i^1}{1-\theta_i^0} + \log p_1 - \log p_0 > 0$

   $\sum_i \left\{x_i^* \log \left(\frac{\theta_i^1}{\theta_i^0}\frac{1-\theta_i^0}{1-\theta_i^1}\right)\right\} + \sum_i \log \frac{1-\theta_i^1}{1-\theta_i^0} + \log p_1 - \log p_0 > 0$

   We may define $w_i = \log \left(\frac{\theta_i^1}{\theta_i^0}\frac{1-\theta_i^0}{1-\theta_i^1}\right)$ and $b = \sum_i \log \frac{1-\theta_i^1}{1-\theta_i^0} + \log p_1 - \log p_0$ and write,

   $\sum_i x_i^* w_i + b > 0$

   $\vec{x}^T\vec{w} + b > 0$, where w defines a hyperplane in the x space.

2. **Derivative of the Log Likelihood**

   $\mathcal{L}(\vec{w}, b) = \sum_{\mu=1}^{P} c^\mu \log \sigma\left(b + \vec{w}^T\vec{x}^\mu\right) + (1-c^\mu)\log\left(1-\sigma\left(b+\vec{w}^T\vec{x}^\mu\right)\right)$

   Note that $\sigma'(y) = (1-\sigma(y))\sigma(y)y'$

   $\nabla_{\vec{w}}\mathcal{L}(\vec{w}, b) = \sum_{\mu=1}^{P} c^\mu \frac{\sigma'(\cdot)}{\sigma(\cdot)} + \frac{(1-\sigma(\cdot))'}{1-\sigma(\cdot)} - c^\mu \frac{(1-\sigma(\cdot))'}{1-\sigma(\cdot)} = \sum_{\mu=1}^{P} c^\mu(1-\sigma(\cdot))\vec{x}^\mu - \sigma(\cdot)\vec{x}^\mu + c^\mu\sigma(\cdot)\vec{x}^\mu$

   $\nabla_{\vec{w}}\mathcal{L}(\vec{w}, b) = \sum_{\mu=1}^{P} c^\mu\vec{x}^\mu - c^\mu\sigma(\cdot)\vec{x}^\mu - \sigma(\cdot)\vec{x}^\mu + c^\mu\sigma(\cdot)\vec{x}^\mu$

   $\nabla_{\vec{w}}\mathcal{L}(\vec{w}, b) = \sum_{\mu=1}^{P} \left(c^\mu - \sigma\left(\vec{w}^T\vec{x}^\mu + b\right)\right)\vec{x}^\mu$

3. **Linearly Separable**

   The hyperplane $\tilde{b} + \tilde{w}^T\vec{x} = \lambda\, b + \lambda\, \vec{w}^T\vec{x} \Rightarrow \lambda(b + \vec{w}^T\vec{x}) = 0$ is geometrically the same as $b + \vec{w}^T\vec{x} = 0$

   If the data is linearly separable, the weights will continue to increase during the maximum likelihood training, and the classifications will become extreme.

4. **Finding Hyperplanes**

   $\vec{w}^T\vec{x} + b = \epsilon^\mu$ forms a linear system with P equations and P unknowns since $\vec{w}$ and b can be expressed with P variables (in a N-dimensional space, $\vec{w}$ has $N-1$ dimensions). If all x^μ,

$\mu \in \{1, ..., P\}$, are linearly independent, then this system has an unique solution. In other words, the matrix X^T, defined as a matrix where the rows are the x^μ, is invertible if all x^μ, $\mu \in \{1, ..., P\}$, are linearly independent (solution of the equation $X^T \vec{w} = \epsilon^\mu - b$).

By considering additional ϵ terms for each data point, we are effectively considering what happens if each data point moves towards the decision boundary by some amount. The classification still must be correct in these circumstances, or it will penalise the likelihood. Hence including these terms acts as a form of regulariser.

For interest only: This approach can be used in combination with a perceptron-like classifier to obtain alternative error measures. For example in the case where we require all *epsilon* values to be greater than some value, then we obtain a support vector classifier. If we presume instead that the ϵ are sampled from a Gaussian, then we obtain a Probit model etc.