

Algorithms in Nature - Problem Set 1

Due: Wednesday, 2/10, at the end of class

1. Regression and MLE [20 points]

Assume we have a large dataset of k tuples, in which the i^{th} tuple has one real-valued input attribute x_i and one real-valued output attribute y_i . To fit the data, we assume the following model with unknown parameter w that needs to be learned from the data.

$$y_i = \log(w^2 x_i) \tag{1}$$

Q1.1 Derive the gradient descent update for w .

Suppose we want to learn the following quadratic regression model with k input attributes:

$$\begin{aligned} y = w_0 + & \quad w_1 x_1 + & \quad w_2 x_2 + & \quad w_3 x_3 + \dots & \quad + w_k x_k \\ & w_{11} x_1^2 + & w_{12} x_1 x_2 + & w_{13} x_1 x_3 + \dots & + w_{1k} x_1 x_k \\ & + & w_{22} x_2^2 + & w_{23} x_2 x_3 + \dots & + w_{2k} x_2 x_k \\ & \vdots & \vdots & \vdots & \vdots \\ & \dots & \dots & \dots & w_{kk} x_k^2 \end{aligned}$$

Q1.2 What is the computational complexity in terms of k of learning the MLE weights using matrix inversion? Use big-O notation.

Q1.3 Same question as above, but assume a cubic regression model (this will include terms such as $w_{111}x_1^3$, $w_{112}x_1^2x_2$, etc).

Q1.4 What is the computational complexity of one iteration of batch gradient descent of the quadratic model?

Q1.5 Same question as above, but using the cubic model.

2. Neural networks short answer [30 points]

Consider neural networks with a single hidden layer. Each unit has a weight \mathbf{w} and uses the following activation functions:

- Hard-threshold function: for a constant a ,

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} \geq a \\ 0 & \text{otherwise} \end{cases}$$

- Linear:

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Q2.1 How many boolean functions $f(x_1, x_2)$, where $x_{1,2} \in \{0, 1\}$, are there?

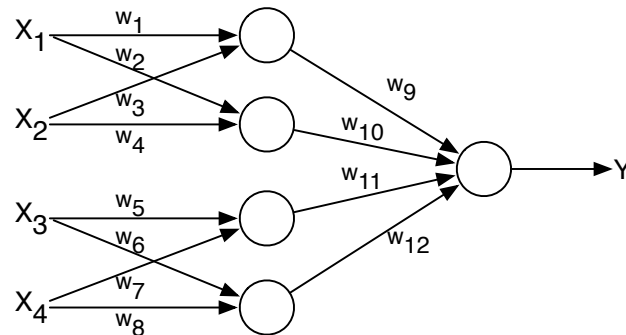
Q2.2 For any boolean function $f(x_1, x_2)$, can we build a neural network to compute f ? If so, prove by *construction*. If not, explain why not.

Consider neural networks with a hidden layer with only one input signal x and one output signal y . Using hard threshold and/or linear activation functions, which of the following cases can be *exactly* represented by a neural network? For each case, if representable, draw the network with choice of activations for both levels and briefly explain how the function is represented. If not representable, explain why not.

Q2.3 Polynomials of degree one.

Q2.4 Polynomials of degree two.

Assume we have the following neural network with linear activation units. The output of each unit is a constant c multiplied by the weighted sum of inputs.



Q2.5 Is it possible for any function that can be represented by the above network to also be represented using a single unit network? If so, draw such a network, including the weights and activation function. If not, explain why not.

Q2.6 Is it possible for any function that can be represented by the above network to also be represented by linear regression? If so, provide an equation for \mathbf{Y} . If not, explain why not.

Q2.7 Draw a fully connected three unit neural network that has binary inputs X_1, X_2 and output Y , where Y implements the XOR function: $Y = (X_1 \text{ XOR } X_2)$. Assume that each unit uses a *hard threshold* activation function.

3. Neural networks training [30 points]

We train a neural network on a set of input vectors $\{\mathbf{x}_n\}$ where $n = 1, \dots, N$, together with a corresponding set of target K -dimensional vectors $\{\mathbf{t}_n\}$. We assume that t has a Gaussian distribution with an \mathbf{x} -dependent mean, so that :

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

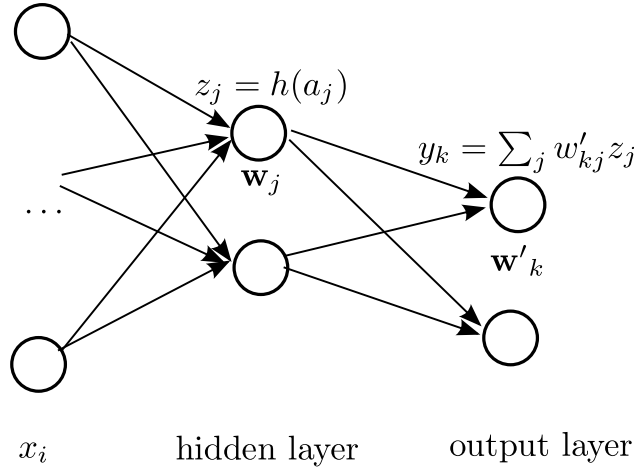
where β is the precision of the Gaussian noise. We would like to find the parameters \mathbf{w} using the MLE principle:

$$\mathbf{w}_{\text{MLE}} = \operatorname{argmax}_{\mathbf{w}} \prod_{n=1}^N p(t_n|\mathbf{x}_n, \mathbf{w}) \quad (2)$$

Q3.1 Show that solving (2) is equivalent to minimizing the sum-of-squares error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|y(\mathbf{x}_n, \mathbf{w}) - t_n\|_2^2 \quad (3)$$

Consider a network with one hidden layer as shown below. Assume that each hidden unit j computes a weighted sum of its inputs \mathbf{x} of the form: $a_j = \sum_i w_{ji}x_i$, where \mathbf{w}_j and \mathbf{z} are the weights and input of this hidden unit, respectively. The activation function h is $h(a_j) = z_j$. Each output unit k computes a weighed sum of its inputs \mathbf{z} of the form $b_k = \sum_j w'_{kj}z_j$, where \mathbf{w}'_k are the weights of this output unit. This unit outputs $y_k = b_k$. Together, the output of this network is: $y(\mathbf{x}_n, \mathbf{w}) = [y_1, y_2, \dots, y_K]^T$.



To train a neural network, we need to compute the derivatives of $E(\mathbf{w})$. We can do so by considering a simpler quantity, $E_n = \frac{1}{2} \|y(\mathbf{x}_n, \mathbf{w}) - t_n\|_2^2$.

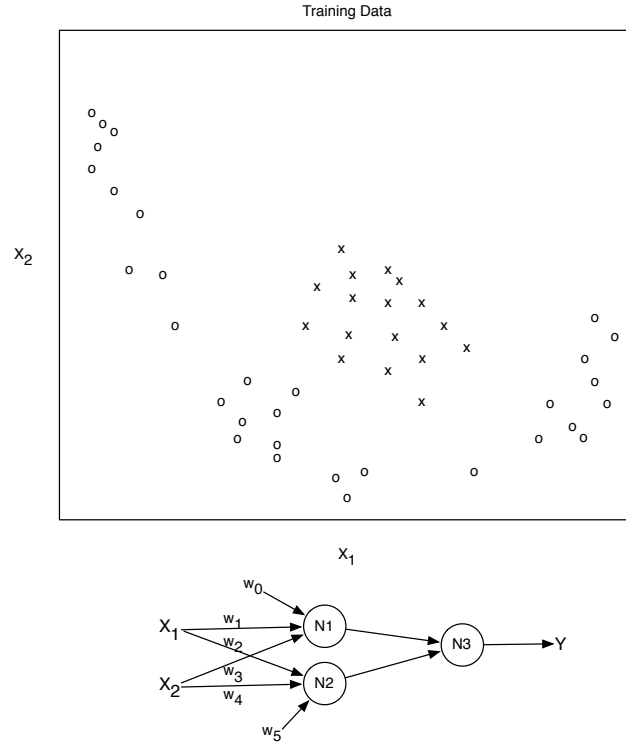
Q3.2 Show that:

$$\frac{\partial E_n}{\partial w_{ji}} = (h'(a_j) \sum_k w'_{kj} \delta_k) x_i$$

where w'_k are weights of the output units and $\delta_k = y_k - t_k$.

4. Neural networks example [20 points]

Consider the following classification training data (where 'x' means True and 'o' means False) and neural network that uses a sigmoid response function ($g(t) = \frac{1}{1+\exp^{-t}}$).



Q4.1 We want to set the weights w of the neural network so that it is capable of correctly classifying the entire dataset. Plot the decision boundaries of N_1 and N_2 (e.g. for neuron N_1 , plot the line where $w_{10} + w_{11}x_1 + w_{12}x_2 = 0$) on the following two graphs.

