Question 1: Bayesian Estimation

(a) An exponential distribution with parameter λ follows a distribution $p(x) = \lambda e^{-\lambda x}$. Given some i.i.d. data $\mathcal{X} = \{x_1, \ldots, x_N\}$ assumed to follow the exponential distribution, derive the maximum likelihood estimate (MLE) λ^{MLE}.

(b) The probabilistic density function (pdf) of a gamma distribution with parameters α, β is $p(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}$ where $\Gamma(t)$ is the gamma function.
If the posterior distribution is in the same family as the prior distribution, then we say that the prior distribution is the conjugate prior for the likelihood function. Show that the Gamma distribution (that is $\lambda \sim \text{Gamma}(\alpha, \beta)$) is a conjugate prior of the exponential distribution. In other words, show that if the i.i.d. data samples $\mathcal{X} = \{x_1, \ldots, x_N\}$ follow an exponential distribution, i.e. $x_n \sim \text{Exp}(\lambda)$, and $\lambda \sim \text{Gamma}(\alpha, \beta)$, then the posterior $p(\lambda|\mathcal{X}) \sim \text{Gamma}(\alpha^*, \beta^*)$ for some values of α^* and β^*.
Tip: Use the Bayes theorem. You will also need to calculate the data likelihood.

(c) Derive the maximum a posteriori estimator (MAP) λ^{MAP} as a function of α and β. What happens as n gets large?

Question 2: Principal Component Analysis

Consider 3 data points in the 2-d space: $(-1, -1)$, $(0, 0)$, $(1, 1)$.

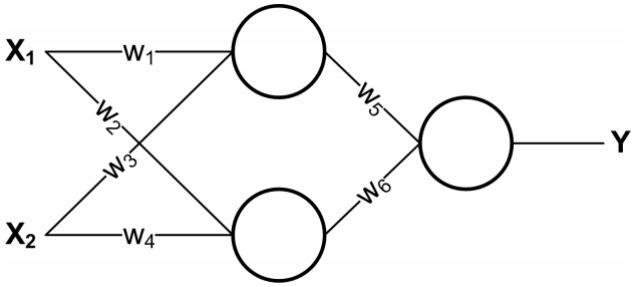
(a) What is the first principal component (write down the actual vector)?

(b) If we project the original data points into the 1-d subspace by the principal component you choose, what are their coordinates in the 1-d subspace? And what is the variance of the projected data?

(c) For the projected data you just obtained above, now if we represent them in the original 2-d space and consider them as the reconstruction of the original data points, what is the reconstruction error?

Question 3: Neural networks

Consider the following neural network, consisting of two input units, a single hidden layer containing two units, and one output unit. For a given unit U, **A** is the vector of activations of units that send their output to U, and **W** is the weight vector corresponding to these outputs



(a) Assuming that the network has linear units, i.e. defining **W** the weights and **A** as above, the output of a unit is $c\mathbf{W}^T\mathbf{A}$ for some constant c. Re-design the neural network to compute the same function without using any hidden units. Express the new weights in terms of the old weights and the constant c.

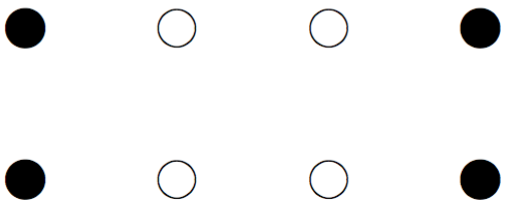
(b) Is it always possible to express a neural network made up of only linear units without a hidden layer?

(c) Using the same architecture as in the original figure, choose an activation function for each unit that will cause the network to learn a similar expression between the input and the output as the logistic regression would learn.

Question 4: Boosting

(a) Consider the following 2-class binary problem. Using the error and weight update rule of AdaBoost, What are the first 2 decision stumps (i.e. vertical or horizontal lines), and what are their corresponding thresholds?

$$\begin{aligned} C_{y=-1} &= \{(-3,-1),(-3,1),(3,-1),(3,1)\} \\ C_{y=1} &= \{(-1,-1),(-1,1),(1,-1),(1,1)\} \end{aligned}$$



(b) Suppose we have built a classifier $f_{t-1}(\mathbf{x})$, and we want to improve it by adding a new classifier $h_t(\mathbf{x})$ to construct a new classifier:

$$f_t(\mathbf{x}) = f_{t-1} + \beta_t h_t(\mathbf{x}) \tag{1}$$

We will examine how we can optimally choose the weight parameter β_t.

(b.i) The strategy we will use is to greedily minimize the exponential loss function.

$$E(\beta_t) = \sum_n e^{-y_n f(\mathbf{x_n})} \tag{2}$$

where $(\mathbf{x_n}, y_n)$ are the training data. Show that this expression is equivalent to

$$E(\beta_t) = (e^{\beta_t} - e^{-\beta_t}) \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\mathbf{x_n})] + e^{-\beta_t} \sum_n w_t(n) \tag{3}$$

where $w_t(n) = e^{-y_n f_{t-1}(\mathbf{x_n})}$
Hint: Take into account equation (1). Take into account that for a binary classification problem with $y_n \in \{-1, 1\}$, this relation holds $y_n h_t(\mathbf{x_n}) = 1$, if $\mathbf{x_n}$ is correctly classified, -1 otherwise

(b.ii) Show that minimizing (3) results in $\beta = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$, where ϵ_t is the misclassification error.