Predicting Happiness!

The city of Somerville, Massachusetts (a suburb of Boston) regularly asks the residents to complete a survey regarding the quality of life as can be influenced by city policy and infrastructure. The survey also asks the residents how happy they are living in Somerville. Recently, the survey has expanded to include more questions and more demographic data. We will consider a subset of the data to see if survey question responses can be used to predict the happiness score. The full data set can be obtained from the city website[1].

The data subset we will consider is provided on the course canvas page as file happiness.csv. This data set contains the 3669 survey responses from the years 2019 through 2023. The data file has 10 columns (labeled A through J) with the following descriptions:

A : Ward/Neighborhood
B : Happiness
C : Beauty of Neighborhood
D : Convenience of Getting Around
E : Housing Condition
F : Street and Sidewalk Maintenance
G : Public Schools
H : Police Department
I : Community Events
J : City Services Information

The data in each column is a positive integer with the following descriptions.

column	description
A	categorical ward designation with values 1–7
B	happiness index: 1=very unhappy, 2=unhappy, 3 = neutral, 4=happy, 5=very happy
C–J	satisfaction index: 1=very unsatisfied, 2=unsatisfied, 3=neutral, 4=satisfied, 5=very satisfied

We consider a classification problem in which happiness (value in column B) can be predicted from the satisfaction (values in columns C through J). The goal of this project is to construct and test one or more "feed-forward neural networks" (FFNN) that can be used to address the classification problem. A FFNN is an example of a classifier function $y_k = f(x_k; w)$ where $x_k \in \mathbb{R}^p$ is a feature vector of satisfaction values associated with person k, $y_k \in \mathbb{R}$ is the happiness value for person k, and $w \in \mathbb{R}^m$ is a parameter vector. That is, we assume that a good classifier is among the family of functions specified by a particular parameter choice. We wish to solve

$$\min_w \quad F(w) = \frac{1}{2} \sum_{k=1}^{n} \left(y_k - f(x_k; w) \right)^2 .$$

A FFNN can be represented as

$$f(x) = \sigma \left(W_d \sigma \left(\cdots \sigma \left(W_2 \sigma \left(W_1 x \right) \right) \cdots \right) \right),$$

[1] https://www.somervillema.gov/HappinessSurvey

where W_j is a $r_j \times c_j$ matrix (with $r_j = p$, $c_j = r_{j+1}$, and $c_d = 1$) and σ is the (elementwise) sigmoid function

$$\sigma(u) = \frac{1}{1 + \exp(-u)}.$$

The parameters of the function are contained in the weight matrices $W_1, \ldots, W_d$. That is, $m = \sum_{j=1}^{d}(r_j c_j)$. The gradient of f can be efficiently computed using the method of backpropagation. Let G_j be the matrix of gradient vector values associated elementwise with the weights in W_j. Then, we have the forward computation:

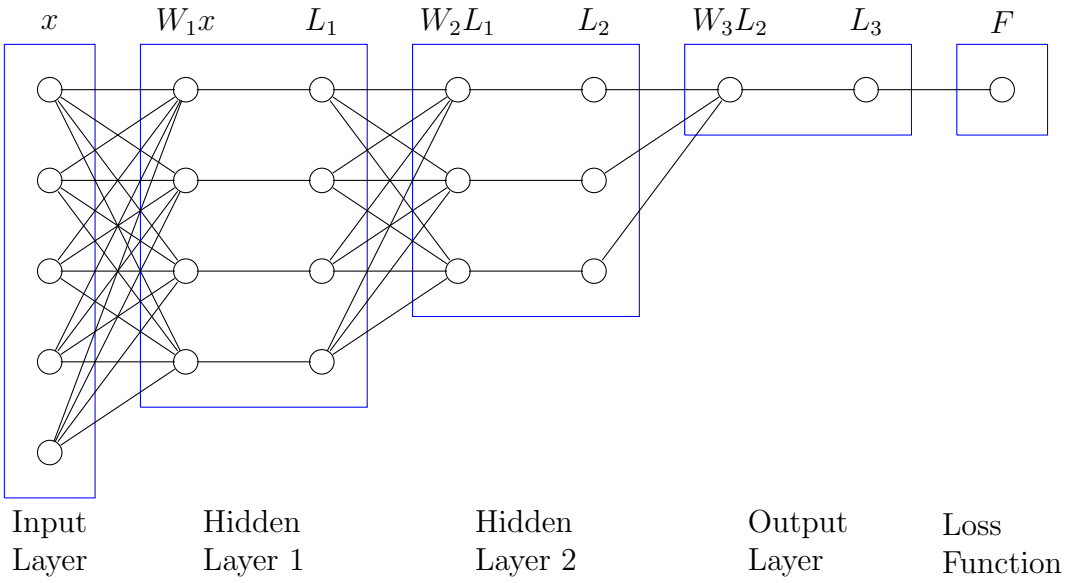
$$\begin{aligned}
L_1 &= \sigma(W_1 x) \\
L_2 &= \sigma(W_2 L_1) \\
&\vdots \\
L_d &= \sigma(W_d L_{d-1}) \\
F &= \tfrac{1}{2} \left\| L_d - y \right\|^2
\end{aligned}$$

and the gradient computation (using the multivariate chain rule):

$$\begin{aligned}
h_d &= (L_d - y) L_d (1 - L_d), \qquad G_d = h_d L_{d-1}^{\mathsf{T}} \\
h_{d-1} &= W_d^{\mathsf{T}} h_d L_{d-1} (1 - L_{d-1}), \qquad G_{d-1} = h_{d-1} L_{d-2}^{\mathsf{T}} \\
&\vdots \\
h_2 &= W_3^{\mathsf{T}} h_3 L_2 (1 - L_2), \qquad G_2 = h_2 L_1^{\mathsf{T}} \\
h_1 &= W_2^{\mathsf{T}} h_2 L_1 (1 - L_1), \qquad G_1 = h_1 x^{\mathsf{T}}
\end{aligned}$$

In this derivation, we have used the fact that $\sigma'(u) = \sigma(u)(1 - \sigma(u))$. Constructing the problem in terms of matrices W_j and G_j provides convenient notation. However, it must be remembered that the parameter vector $w \in \mathbb{R}^m$ is constructed from the collective and ordered elements of $W_1, W_2, \ldots, W_d$. Similarly, the gradient vector $g \in \mathbb{R}^m$ is constructed from the identically ordered elements of $G_1, G_2, \ldots, G_d$. One interesting point is that if one employs the identity function instead of the sigmoid (or other nonlinear monotonic function) then the entire transformation is linear with both forward and backward computations collapsing into a single matrix product.

It can be helpful to visualize the FFNN as a computational directed acyclic graph. The forward computation proceeds left to right.



In this example, the network contains two hidden layers and uses three weight matrices. W_1 is a 4×5 matrix, W_2 is a 3×4 matrix, and W_3 is a 1×3 matrix. The computations within each layer are the elementwise sigmoid "activation" functions. The input x can be a single vector of (five, in this example) attributes, or it can be an array (five by m) of attributes of m individuals. Remember that the output $F \in (0, 1)$. So, if the output is to distinguish between h ordered possibilities, then exact match can be assigned values as $\left\{\frac{1}{h+1}, \frac{2}{h+1}, \ldots, \frac{h}{h+1}\right\}$. This choice leads to stability in weight determination – F is never driven to values of zero or one.

Complete the following tasks.

1. Construct a function which reads the Somerville data set, fills in missing data with justification, and normalizes the data appropriately.

2. Construct a function that computes – for a general, user-specified FFNN – the loss function for a classification problem and the (backpropagation) gradient. That is, construct a function for which the user selects the number and size of hidden layers and the number of inputs.

3. Choose a method for selecting training data and test data sets.

4. Solve the classification problem using a FFNN which includes all eight input quantities (satisfaction values), two hidden layers, of 12 and 10 nodes, and a single output layer. Test various optimization methods using the code you have developed.

5. Explore other choices of FFNN construction.