ELL409: Assignment 2

Maximum points: 6

Demo Schedule: 10th April 2016 (tentative)

April 1, 2016

Handwritten Digit Recognition

Automated handwritten digit recognition is widely used today, for e.g., recognising postal codes on envelopes for sorting the mail. You will use logistic regression and neural networks to recognise handwritten digits (0 to 9).

A data set containing 5000 training examples of handwritten digits can be downloaded from here: `http://privateweb.iitd.ac.in/~seshan/a2/handwritten_image_set.rar`. The data has been saved in a tab-delimited text (ASCII) format. Each training example is a 28 pixel by 28 pixel grayscale image of the digit. Each pixel is represented by a floating point number indicating the grayscale intensity at that location. The 28 by 28 grid of pixels is reshaped into a 784-dimensional vector. Each of these training examples becomes a single row in our data file. Reading the data from this text file into a matrix gives a 5000 by 784 matrix where every row is a training example for a handwritten digit image.

The second part of the training set is a 5000-dimensional vector that contains labels for the training set. The digits “0” to “9” are labelled as “0” to “9” in their natural order.

1. **Logistic Regression** (2 points)

   Using multiple one-vs-all logistic regression models build a multi-class classifier. Since there are 10 classes, you will need to train 10 separate logistic regression classifiers. Recall that in (unregularised) logistic regression, the cost function is given by:

   $$E(w) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_w(x^{(i)})) - (1-y^{(i)})\log(1-h_w(x^{(i)}))\right]$$

   where $(x^{(i)}, y^{(i)})$ is the i^{th} training example, m is the total number of training examples, and $h_w(x^{(i)}) = g(w^T x^{(i)}) = \frac{1}{1+e^{-w^T x^{(i)}}}$. The j^{th} element of the gradient of the above cost function can be shown to be:

   $$\frac{\partial E(w)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left((h_w(x^{(i)}) - y^{(i)})x_j^{(i)}\right).$$

   - Using the above expressions you can compute the cost function and its gradient which can be used by an unconstrained optimisation routine to obtain the optimal classifier parameter $w \in \mathbb{R}^{(N+1)}$ where $(N+1)$ is the dimensionality of the inputs $x^{(i)}$ (including the bias term). After training a separate classifier for each of the K classes, your code should return all the classifier parameters in a matrix $W \in \mathbb{R}^{K\times(N+1)}$, where each row of W corresponds to the learned logistic regression parameters for one class.
   - The multi-class classifier trained using the steps above can now be used to predict the digit contained in a given image. Using the learned parameter matrix W, make predictions on the training set and determine the training set accuracy.
   - Add quadratic regularisation to the cost function so that:

   $$E(w) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_w(x^{(i)})) - (1-y^{(i)})\log(1-h_w(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{N} w_j^2.$$

Note that the bias term w_0 is not considered for regularisation. Obtain the corresponding expression for the gradient of the cost function and retrain your classifier with this regularised cost function. Comment on the differences.

- During the demonstration you will asked to show your classifier output and performance on a separate set of test images.

2. **Neural Networks** (4 points)

In this part, you will implement a neural network to recognize handwritten digits using the same training set as before. The neural network will be able to represent more complex models that form non-linear hypotheses. You are expected to implement the forward propagation algorithm to determine the activations of the nodes in the neural network and the backpropagation algorithm for learning the neural network parameters. Do not use any of the built-in neural network model libraries.

- You can assume a neural network with only 3 layers.
- Implement the feedforward computation that computes $h_w(x^{(i)})$ for every training example i and sum the cost over all examples.
- When training neural networks, the parameters are randomly initialised. One strategy is to randomly select values for $W^{(l)}$ uniformly in the range $[-\epsilon, \epsilon]$ where $\epsilon = \frac{\sqrt{6}}{\sqrt{S_l+S_{l+1}}}$ and S_l, S_{l+1} are the number of neurons in the layers adjacent to $W^{(l)}$.
- Implement the backpropagation algorithm by computing the error terms associated with the output layer and propagating these errors back towards the input layer. Use the following expression for the hidden layers:
$$\delta^{(l)} = (W^{(l)})^T \delta^{(l+1)} .* g\prime(z^{(l)})$$
where $.*$ represents elementwise multiplication and $g\prime(z) = g(z)(1-g(z))$ is the derivative of the sigmoid function.
- The gradient of the neural network cost function is given by:
$$\frac{\partial E(W)}{\partial W_{ij}^{(l)}} = \frac{1}{m}\Delta_{ij}^{(l)}$$
where $\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)}(a^l)^T$.
- Learn a good set of parameters for the neural network by providing the neural network cost function and gradient to an unconstrained optimisation routine. Apply the learned model to determine the training set accuracy.
- Now introduce regularisation and comment on the differences. Note that with regularisation an additional term is introduced in the gradient computation:
$$\frac{\partial E(W)}{\partial W_{ij}^{(l)}} = \frac{1}{m}\Delta_{ij}^{(l)} + \frac{\lambda}{m}W_{ij}^{(l)} \quad \text{for } j \geq 1.$$

You should prepare a report, compiling all your results and your interpretation of them, along with your overall conclusions. In particular, you should attempt to answer all of the questions posed above. Any graphs or other visualisations should also be included in the report. A soft copy of the report should be submitted along with your code to the TAs at the time of the demonstration.