Neural Networks

Data Mining
Prof. Dawn Woodard
School of ORIE
Cornell University

Outline

Training a Neural Network

Basic idea of neural net training (via "Back-Propagation"):

- Start with some arbitrary values for the paramters α, β
- Calculate predictions $\hat{y}_i$ for all observations *i* in the training data ("forward sweep"). Use these to calculate the error.
- Update the parameters by changing them in a way that decreases that error ("backward sweep")
- Repeat this iterative process until:

Training a Neural Network

Continuous-outcome case:

Training error is measured using RSS = $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ where n is the # training observations

Because each node is a differentiable function of its inputs, it is possible to calculate $\frac{\partial \text{RSS}}{\partial \alpha_{mj}}$ and $\frac{\partial \text{RSS}}{\partial \beta_{km}}$, the derivative of RSS w.r.t. the parameters

Training a Neural Network

Continuous-outcome case:

1. Start with arbitrary initial values $\{\alpha_{mj}^{(0)}, \beta_{km}^{(0)} : m \in \{0, \ldots, M\}, j \in \{0, \ldots, p\}, k \in \{1, \ldots, K\}\}$ of the parameters

2. Do the following update until the validation-set RMSE starts to increase:

$$\beta_{km}^{(r+1)} =$$

$$\alpha_{mj}^{(r+1)} =$$

Training a Neural Network

The amount by which the parameters are allowed to change in each iteration is controlled by the "learning rate" $\gamma \in (0, 1)$

Why don't we just keep iterating until the parameters stop changing (i.e. find a local minimum of RSS)?

Digit Recognition

Handwritten digits from U.S. postal envelopes (Example from Hastie, Tibshirani, and Friedman 2009):

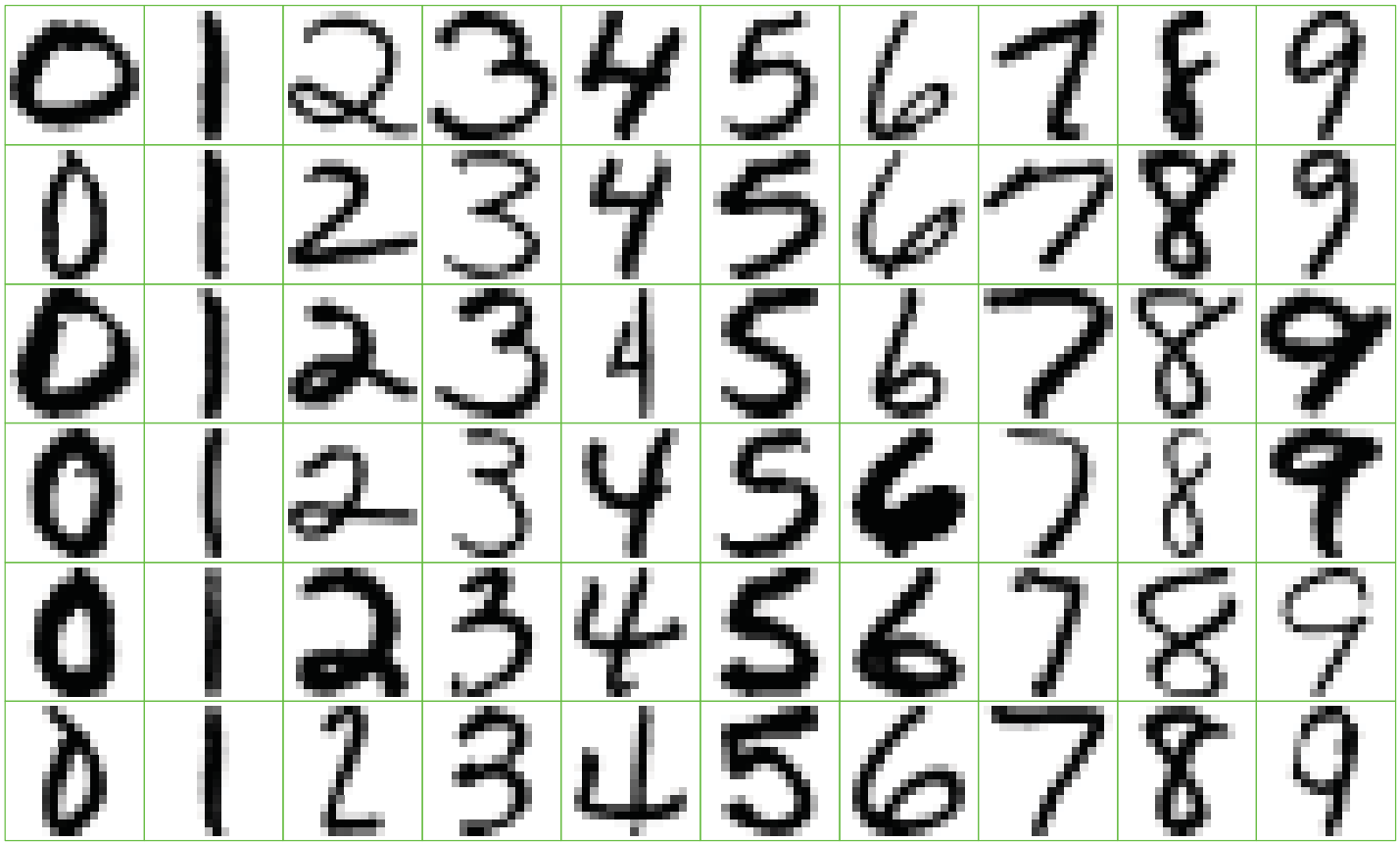

FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Digit Recognition

- Have a dataset with labeled digits. Goal is to learn a decision rule for identifying a digit from its image
- The predictors are the grayscale intensities of each of the pixels in the 16x16 image (256 continuous predictors that take values 0 to 255)
- Outcome variable is:

Digit Recognition

- This is a classification problem with $K = 10$ categories
- Such a decision rule would be used for automatic sorting of mail according to its destination
- Very high accuracy is required
- Some preprocessing has been done to align the digits spatially
- For illustration there are 320 digits in training set and 160 in test set

Digit Recognition

- To apply neural networks here we have to have 10 output nodes, one for each possible value of the outcome variable
- Various neural net structures were considered, inculding using two hidden layers that are “locally connected.” Locally connected means that not all possible edges are included in the graph; this makes the model less prone to overfitting
- Also the technique of “weight sharing” was used, meaning that some nodes in the hidden layer are forced to have the same parameters α (although they have slightly different inputs due to the local connectivity). Again this is done to avoid overfitting.

Digit Recognition

Five neural nets were trained on the training data:

1. No hidden layer, equivalent to “multinomial logistic regression” (logistic regression with >2 outcome values)
2. One hidden layer with 12 nodes
3. Two hidden layers locally connected
4. Two hidden layers, locally connected with weight sharing
5. Two hidden layers, locally connected, two levels of weight sharing

Digit Recognition

Results:

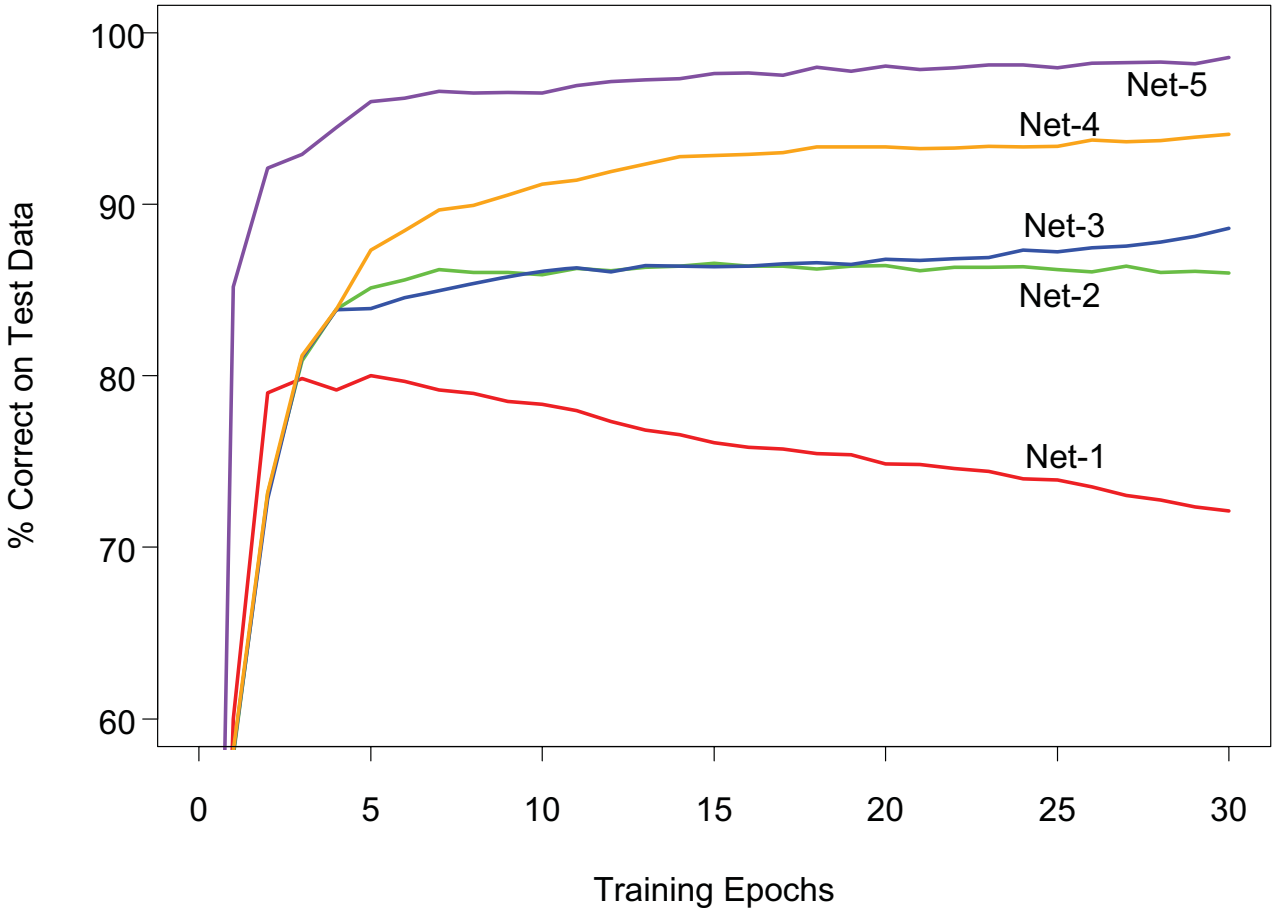

FIGURE 11.11. *Test performance curves, as a function of the number of training epochs, for the five networks of Table 11.1 applied to the ZIP code data. (Le Cun, 1989)*

Digit Recognition

Accuracy of 5 models on test data:

TABLE 11.1. *Test set performance of five different neural networks on a handwritten digit classification example (Le Cun, 1989).*

	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

Other Applications

The company “Alyuda” has created a commercial software product exclusively for fitting and applying neural networks.

Find their demo by googling “Alyuda”, and clicking on “NeuroIntelligence” and then “Take a Tour”