

**Homework 2**

**Due: Thursday, October 4**

1. Prove the assertion given by equation (12.42) on page 388 of the text.

Hint: Proceed by contraction: if the result is not true,  $\hat{\beta}_m$  must be outside the span of  $\mathbf{H}$ ; show that you can reduce the objective function by projecting into that span.

2. Text, Ex 4.2.

3. Neural Networks and Back Propagation

- Text Ex 11.2; demonstrate why weights cannot be initialized at exactly zero.
- Text Ex 11.3. Show that the back-propagation formulation can be made to work for multi-hidden layer networks.

4. Early Stopping and Weight Decay

- (a) Consider a quadratic error criterion function of the form

$$Q(\mathbf{w}) = Q_0 + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^t \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where  $\mathbf{w}^*$  represents its minimum, and the (constant) Hessian is positive definite. Suppose that instead of minimizing this error function we apply “weight decay”

$$\tilde{Q}(\mathbf{w}) = Q(\mathbf{w}) + \frac{\nu}{2} \mathbf{w}^t \mathbf{w}$$

to obtain the “regularized” solution

$$\tilde{\mathbf{w}} = \arg \min_{\mathbf{w}} \tilde{Q}(\mathbf{w}).$$

Let  $\{\lambda_i, \mathbf{u}_i\}$  be the eigenvalues and corresponding eigenvectors of  $\mathbf{H}$ . Show that

$$\tilde{\mathbf{w}}^t \mathbf{u}_j = \frac{\lambda_j}{\lambda_j + \nu} \mathbf{w}^{*t} \mathbf{u}_j.$$

Show further that this result implies that weight decay tends to preferentially shrink the components of the weight vector along directions in the weight space in which the gradient of  $Q(\mathbf{w})$  is least rapidly varying.

- (b) Now we consider early stopping under the same assumptions. Suppose the initial weight vector  $\mathbf{w}_0$  is chosen at the origin in weight space, and it is updated using simple gradient descent

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \mathbf{G}_{t-1}$$

where  $t$  denotes the step number, and  $\eta$  is a (small positive) “learning rate”.

- i. Show that after  $t$  steps the components of the weight vector parallel to the eigenvectors of  $\mathbf{H}$  can be written as

$$\mathbf{w}_t^t \mathbf{u}_j = [1 - (1 - \eta \lambda_j)^t] \mathbf{w}^{*t} \mathbf{u}_j$$

where  $\{\lambda_i, \mathbf{u}_i\}$  are the eigenvalues and corresponding eigenvectors of  $\mathbf{H}$ .

- ii. Show that as  $t \rightarrow \infty$  one obtains  $\mathbf{w}_t \rightarrow \mathbf{w}^*$  as expected, provided  $|1 - \eta \lambda_j| < 1$ . Now suppose the training is stopped after a finite number  $t$  of steps. Show that the components of the weight vector parallel to the eigenvectors of the Hessian satisfy

$$\mathbf{w}_t^t \mathbf{u}_j \simeq \mathbf{w}^{*t} \mathbf{u}_j, \text{ when } \lambda_j \gg 1/\eta t$$

and

$$|\mathbf{w}_t^t \mathbf{u}_j| \ll |\mathbf{w}^{*t} \mathbf{u}_j| \text{ when } \lambda_j \ll 1/\eta t.$$

- (c) Compare this result with that of Problem 4a using regularization based on simple weight decay. Hence, show that  $1/\eta t$  is analogous to the regularization parameter  $\nu$ . These results show that early stopping of steepest-descent training is a regularization procedure with the degree of regularization controlled by training time  $t$ . Less time gives rise to increased regularization.

5. **Spam Email.** For this problem we will consider the HP spam data base. Documentation for the data is available from the class website. It has already been split into test and training data.

- (a) We first examine the stability of `nnet`. Fit three models to the training data using a 10 hidden units and the program defaults (these are not very sophisticated). Plot the fitted values *for the training data* for these models against each other. Does there appear to be satisfactory agreement between them? Compare the result to using a weight decay parameter of 0.1 and 1000 iterations.

You can obtain fitted values from the output of `nnet`. If these are placed in a data frame labeled `preds`, then `plot(preds)` provides a trellis plot.

- (b) Fit on the *training set* one hidden layer neural networks with 1, 2, ..., 10 hidden units and different sets of starting values for the weights (obtain in this way one model for each number of units). Which structural model performs best at classifying on the test set?

- (c) Choose the optimal regularization (weight decay for parameters  $0, 0.1, \dots, 1$ ) for the structural model found above by averaging your estimators of the misclassification error on the *test.set* (`spam.test`). The average should be over 10 runs with different starting values. Describe your final best model obtained from the tuning process: number of hidden units and the corresponding value of the regularization parameter. What is an estimation of the misclassification error of your model?
- (d) The goal now is to obtain a spam filter. Repeat the previous point requiring this time the proportion of misclassified good emails to be less than 1%.
- (e) Compare the test set performance of your neural network spam filter to that of linear discriminant analysis and logistic regression. The function `glm` will perform logistic regression.

Bonus: Experiment with defining new features for logistic regression and linear discriminant analysis. Can you make these competitive with the neural network?

HINTS:

- (a) For the data problems you need to first attach the neural networks library. This can be done by executing the command `library(nnet)`. Examples using the SPlus functions `nnet` and `predict.nnet` are in the book *“Modern Applied Statistics with SPlus* by Venables and Ripley and the online R help system. Chapter 11 of the text may be also useful for a quick overview of the methodology.
- (b) For classification use the default logistic output units. A way to choose the percentage of misclassified emails to be less than 1% is to threshold the probability of being a good email at a appropriate value. Using the option `type='raw'` in `predict.nnet`, the output consists of vectors of probabilities.
- (c) Make sure that you start *early*. The data problem (tuning neural networks), although not difficult from the conceptual point of view will be time consuming.