SMLDM HT 2014 - MSc Week 7
Assessed Group Practical

In this practical you will investigate the use of cross-validation, logistic regression, multinomial logistic regression, neural networks and kNN for handwriting digit recognition.

We will use the USPS dataset of handwritten digits, which you can obtain from the course webpage. Each handwritten digit is 16×16 in size, so that data vectors are $p = 256$ dimensional. There are 2000 digits (200 digits of each class) in each of the training set and test set. The R script `loadusps.R` will read in the dataset into memory.

The first part will not be assessed and you may obtain help from the TAs in answering the questions. Hand in the second part for assessment.

Include R code as part of your report, incorporating relevant snippets as part of the main text, rather than in the appendix. Report results in plots and tables as appropriate. Discuss your findings clearly, *but succinctly*.

Non-assessed Component

0. We will start with logistic regression. Use the `glmnet` library as it can incorporate both L_1 and L_2 regularization. You may want to read the manual `cran.r-project.org/web/packages/glmnet/glmnet.pdf`.

   Logistic regression is a binary classification model, while our problem has $K = 10$ classes. In this non-assessed component we will only look at discriminating between class 9 versus the other classes. The following command will train the logistic regression model:

   ```
   m <- glmnet(as.matrix(trainx),trainy==9,family="binomial")
   ```

   The command will in fact fit a logistic regression model for a whole sequence of regularization parameters λ, with $\lambda = 0$ being no regularization.

   (a) Report the error rates on the training and test set for each model, at $\lambda = 0$ (i.e. no regularization). For a fitted model `m`, you can use the command:

   ```
   testp <- predict(m,newx=as.matrix(testx),type="class",s=c(0))
   sum((testy==9)!=testp)/2000
   ```

   to compute the predictions on a test set. The `s=c(0)` option sets the tuning parameter λ to 0 (no regularization).

   Is the model overfitting?

   (b) `glmnet` has built-in cross-validation support, via the command `cv.glmnet`. Use it to tune the hyperparameter λ governing the amount of regularization, and train the logistic regression model. Use the option `type.measure="class"` to determine the appropriate hyperparameter using the 0-1 misclassification loss. You can use the following command for training a logistic regression model discriminating between class 9 and the other classes with cross-validated `glmnet`:

   ```
   m <- cv.glmnet(trainx,trainy==9,family="binomial",
                                   type.measure="class")
   ```

The procedure is quite computationally intensive and can take a few minutes.

Plot the training, validation, and test set learning curves as functions of $\log(\lambda)$.

Comment on what aspects of these curves correspond or do not correspond to your expectations.

(c) You will now consider using neural networks for classifying the handwritten digits, using the `nnet` library. You will find the documentation for the `nnet` library useful: `http://cran.r-project.org/web/packages/nnet/nnet.pdf`.

Train a 1 hidden layer neural network with $S = 10$ hidden units for $T = 200$ iterations.

You can use the command:

```
net <- nnet(trainx,trainy==9,size=10,
            MaxNWts=100000,entropy=TRUE,maxit=200)
```

Report the training and test errors. Find out what each option does.

(d) Plot the learning curves of the training and test errors, as a function of the number of iterations. Hint: You can use the `Wts` option of `nnet` to pass it an initial set of weights.

Assessed Component

1. We will start with log-linear models: logistic regression and multinomial logistic regression. As noted previously, logistic regression is a binary classification model, while our problem has $K = 10$ classes. We can convert our problem to 10 binary classification problems using a 1-vs-rest encoding. For each class k, train a logistic regression model to predict images of class k versus images of other classes.

   (a) Report the error rates on the training and test set for each model, at $\lambda = 0$ (i.e. no regularization).

   Are the models overfitting?

   (b) Use cross validation to determine the appropriate hyperparameter using the 0-1 misclassification loss.

   Plot the training, validation, and test set learning curves as functions of $\log(\lambda)$. For each k overlay all three curves on the same plot.

   Comment on what aspects of these curves correspond or do not correspond to your expectations.

   (c) Each logistic regression model only discriminates between a class and the other classes. Given a test image, we can heuristically predict its class by reporting the class k, with highest conditional probability of class k, under the model which discriminates class k against others.

   How does the test error of this procedure compare with the test errors obtained by each individual logistic regression model?

   (d) You can also use `glmnet` for multinomial logistic regression, by using the option `type="multinomial"`. Train such a model, again using cross-validation to tune the hyperparameter λ.

Report on the validation set error rate.

How does this compare to doing K 1-vs-rest logistic regression analyses?

2. (a) Using `nnet`, train a neural network with softmax nonlinearity to discriminate among all 10 classes.

   Report the training and test errors after every 10 iterations of training the network. Use networks of size $S = 10$ and up to $T = 200$ total iterations, and do not use weight decay.

   Plot these as a function of the number of iterations. Comment on any salient features of the learning curve, and elaborate on whether the learning curve agrees with what you learnt in class.

   (b) Repeat the above 10 times. Do the learning curves differ across different runs?

   (c) Using a validation set, or cross validation (which you either have to programme yourselves, or use `cvTools` library), determine the best weight decay parameter to use. Use `exp(-5:3)` as decay parameters to try.

   Plot the training and validation errors. What value would you pick for the decay parameter?

   (d) Fold in the validation set, and train the network on the full training set.

   Report the test error achieved.

   (e) With the chosen decay parameter, use a validation set to determine the best network size, trying $S = 10, 20, 30, 50, 70, 100$. Use $T = 200$. The larger networks may take a while to learn.

   Report the training and validation errors achieved and the chosen network size. Comment on any salient features of the learning curves, and elaborate on whether the learning curve agrees with what you learnt in class.

   (f) Is the above procedure the best way to tune the two hyperparameters? Describe an alternative procedure which may produce better results, and comment on the pros and cons.