Elements of Statistical Learning
Solution Sheet 4 (Rev 2, Apr 28 2021)

1. *Suppose $X$ is a centred and sphered $n \times p$ data matrix. Let $y = Xa$, where $a$ is a unit norm projection vector, where $y = (y_1, \ldots, y_n)$. Show that the mean and variance of $y$ are zero and one respectively. Suppose $b$ is another unit norm vector and $b^T a = 0$. Show that the data $z = Xb$ is uncorrelated with $y$. Why is it pointless to apply principal components analysis to $X$?*

   If $X$ is centred and sphered then $\bar{X} = n^{-1}\mathbf{1}^T X = 0$ and $S = X^T X/n = I_p$. Let $Y = Xa$. Then

$$\bar{Y} = n^{-1}\mathbf{1}^T Y = n^{-1}\mathbf{1}^T Xa = 0, \tag{1}$$

   and

$$\begin{aligned} S_y &= n^{-1}(Y - \bar{Y})^T (Y - \bar{Y}) & (2)\\ &= n^{-1} Y^T Y & (3)\\ &= n^{-1} a^T X^T X a & (4)\\ &= a^T I_p a = a^T a = 1. & (5) \end{aligned}$$

   For the second part $z = Xb$ so

$$\operatorname{cov}(Z, Y) = n^{-1} b^T X^T X a = b^T a = 0. \tag{6}$$

   There's no point apply PCA to a centred and sphered matrix, $X$, as the variance matrix of $X$ is the identity and already diagonalized and the eigenbasis is just the standard basis (i.e. the one you are in). Moreover, all eigenvalues are the same (1)

2. *Let $X$ be a $p$-dimensional random vector with mean zero and identity covariance matrix. Let $\theta$ be a unit norm vector and write the projection of $X$ onto $\theta$ as $Y_\theta = X^T\theta$. Let $f_\theta(y)$ be the density of $Y_\theta$. Then, the L2 distance between $f_\theta(y)$ and $\phi(y)$ is*

$$J(\theta) = \int_{-\infty}^{\infty} \{f_\theta(y) - \phi(y)\}^2 \, dy, \tag{7}$$

   *where $\phi(y)$ is the standard normal density. Let $H_0, H_1, \ldots$ be Hermite polynomials, orthogonal on $\mathbb{R}$ with respect to the weight function $\phi^2(x)$ and standardised by*

$$\int H_j^2(y)\phi^2(y)\, dy = j!\pi^{-1/2}2^{j-1}, \tag{8}$$

   *and that the term of the highest degree in $H_i$ has positive coefficient. Note: $H_0(x) = 1$.*

   *Show that the Hermite functions*

$$h_j(y) = (j!)^{-1/2}\pi^{1/4}2^{-(i-1)/2}H_j(y)\phi(y), \quad -\infty < y < \infty, \tag{9}$$

   *are orthonormal.*

*The Hermite function representation of $f_a(x)$ is given by*

$$f_\theta(y) = \sum_{i=0}^{\infty} a_i(\theta) h_i(y), \tag{10}$$

*where*

$$a_i(\theta) = \int f_\theta(y) h_i(y)\, dy = \mathbb{E}\{h_i(Y_\theta)\}. \tag{11}$$

*Show that*

$$J(\theta) = \sum_{i=0}^{\infty} a_i(\theta)^2 - (2^{1/2}/\pi^{1/4}) a_0(\theta) + (2\pi^{1/2})^{-1}. \tag{12}$$

To do this, see that

$$\begin{aligned}
\int h_i(x) h_j(x)\, dx &= \int (i!j!)^{-1/2} \pi^{1/2} 2^{-(i-1)/2-(j-1)/2} H_i(x) H_j(x) \phi^2(x)\, dx \\
&= (i!j!)^{-1/2} \pi^{1/2} 2^{-(i+j-2)/2} \int H_i(x) H_j(x) \phi^2(x)\, dx \quad (13) \\
&= 0 \quad (14)
\end{aligned}$$

if $i \neq j$ as the $H_i(x)$ are orthogonal polynomials with weight function $\phi^2(x)$ and

$$\int h_i^2(x)\, dx = (i!)^{-1} \pi^{1/2} 2^{-(i-1)} i! \pi^{-1/2} 2^{i-1} = 1. \tag{15}$$

So $\int h_i(x) h_j(x)\, dx = \delta_{i,j}$, the Kronecker delta.

For the second part note that

$$h_0(y) = 2^{1/2} \pi^{1/4} H_0(y) \phi(y), \tag{16}$$

hence $\phi(y) = 2^{-1/2} \pi^{-1/4} h_0(y)$. So,

$$\begin{aligned}
J(\theta) &= \int \left\{ \sum_{i=0}^{\infty} a_i(\theta) h_i(y) - 2^{-1/2} \pi^{-1/4} h_0(y) \right\}^2 \quad (17) \\
&= \int \left\{ \sum_{i=0}^{\infty} b_i(\theta) h_i(y) \right\}^2 dy, \quad (18)
\end{aligned}$$

where $b_0(\theta) = a_0(\theta) - 2^{-1/2}\pi^{-1/4}$ and $b_i(\theta) = a_i(\theta)$ for $i = 1, 2, \ldots$. Since the Hermite functions are an orthogonal series expansion, we can use Plancheral's theorem from Homework Sheet 3, Question 8 to show that

$$\begin{aligned}
J(\theta) &= \sum_{i=0}^{\infty} b_i^2(\theta) \quad (19) \\
&= b_0^2(\theta) + \sum_{i=0}^{\infty} b_i(\theta)^2 \quad (20) \\
&= a_0^2(\theta) - 2^{1/2}\pi^{-1/4} a_0(\theta) + 2^{-1}\pi^{-1/2} + \sum_{i=1}^{\infty} a_i(\theta)^2 \quad (21) \\
&= \sum_{i=0}^{\infty} a_i(\theta)^2 - 2^{1/2}\pi^{-1/4} a_0(\theta) + 2^{-1}\pi^{-1/2}, \quad (22)
\end{aligned}$$

as required.

3. *Let $Y_j$ be the $j$th component of the random $p$-vector $Y$. Show that $\sum_{j=1}^{p} H(Y_j)$ is the entropy of the 'independence version' of $Y$, i.e. $\prod_{j=1}^{p} g_j(y_j)$, where $g_j$ is the marginal density of $Y_j$.*

   We have

$$H\left(\prod_{j=1}^{p} g_j(y_j)\right) = \int_{\mathbb{R}^p} \prod_{j=1}^{p} g_j(y_j) \log\left\{\prod_{i=1}^{p} g_i(y_i)\right\} d^p\mathbf{y} \tag{23}$$

$$= \int_{\mathbb{R}^p} \prod_{j=1}^{p} g_j(y_j) \sum_{i=1}^{p} \log\{g_i(y_i)\} d^p\mathbf{y} \tag{24}$$

$$= \sum_{i=1}^{p} \int_{\mathbb{R}^p} \left\{\prod_{j=1}^{p} g_j(y_j)\right\} \log\{g_i(y_i)\} d^p\mathbf{y} \tag{25}$$

$$= \sum_{i=1}^{p} R_i = \sum_{i=1}^{p} H(Y_i), \tag{26}$$

   since

$$R_i = \int_{\mathbb{R}^p} \left\{ g_i(y_i) \times \prod_{j=1, j\neq i}^{p} g_j(y_j) \right\} \log\{g_i(y_i)\} d^p\mathbf{y}, \tag{27}$$

   for $i = 1, \ldots, p$. Now

$$R_i = \int_{\mathbb{R}} g_i(y_i) \log\{g_i(y_i)\} dy_i \prod_{j=1, j\neq i}^{p} \cancelto{1}{\int_{\mathbb{R}} g_j(y_j) dy_j} = H(Y_i), \tag{28}$$

   as $g_j$ is a probability density.

4. *From slide 11 in Lecture 16 we assumed $X$ is a random vector with covariance $I$, $Y = A^T X$, where $A$ is orthogonal. Show that*

$$I(Y) = \sum_{j=1}^{p} H(Y_j) - H(X), \tag{29}$$

   *where $H(Y)$ is the entropy of random variable $Y$.*

   If we can show that $H(Y) = H(X) - \log|\det A|$, then, since $\det A = \pm 1$ for an orthogonal matrix we have that $H(Y) = H(X)$ since $\log|\pm 1| = 0$. So, the entropy of $Y$ is $H(Y) = H(AX)$. The probability density of $X$, $f_X$, bestows probability density status on $Y = AX$, which is $g_Y$. The formula for writing the density of $Y$ in terms of that for $X$ is given by

$$g_Y(\mathbf{y}) = f_X\{x_1(\mathbf{y}), \ldots, x_p(\mathbf{y})\} \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right|. \tag{30}$$

This is the multivariate version of the usual change of variable formula for probability densities. So,

$$
\begin{aligned}
H(Y) &= -\int g(\mathbf{y}) \log\{g(\mathbf{y})\} d^p\mathbf{y} \qquad (31)\\
&= -\int f_X\{x_1(\mathbf{y}), \ldots, x_p(\mathbf{y})\} \left|\frac{\partial \mathbf{x}}{\partial \mathbf{y}}\right| \log\left\{ f_X\{x_1(\mathbf{y}), \ldots, x_p(\mathbf{y})\} \left|\frac{\partial \mathbf{x}}{\partial \mathbf{y}}\right| \right\} d^p\mathbf{y}\\
&= -\int f_X(A^T\mathbf{y}) |\det A^T| \left[\log\left\{f_X(A^T\mathbf{y})\right\} + \log(|\det A|)\right] d^p\mathbf{y} \qquad (32)
\end{aligned}
$$

Now effect a change of variable from $\mathbf{y}$ to $\mathbf{x}$ using $X = YA^T$, which is a linear transformation with Jacobian $|\det A|$. So,

$$
\begin{aligned}
H(Y) &= -\int f_X(\mathbf{x}) \left[\log\left\{f_X(\mathbf{x})\right\} + \log(|\det A|)\right] d^p\mathbf{x} \qquad (33)\\
&= -\int f_X(\mathbf{x}) \log\left\{f_X(\mathbf{x})\right\} d^p\mathbf{x} - \log(|\det A|) \cancelto{1}{\int f_X(\mathbf{x}) d^p\mathbf{x}}\\
&= H(X) - \log(|\det A|), \qquad (34)
\end{aligned}
$$

as required.

5. *Apply independent components analysis to the* `iris` *data within* R. *E.g. use the* `fastICA` *package. ICA here should produce projections/solutions that are non-Gaussian (and assumed independent) and so should do a job similar to exploratory projection pursuit.* Perhaps start with

```
fastICA(X=iris[,1:4], n.comp=3)
```

and experiment with the arguments.

6. *Visit the site:*

```
https://developers.google.com/machine-learning/crash-course/
 introduction-to-neural-networks/playground-exercises
```

*and play with some of the nice interactive graphics to build your own neural networks and assess performance.* Just visit and play!

7. *(This is question 11.7 from ELS) Fit a neural network to the* `spam` *data of Section 9.1.2, and compare the results to those for the additive model given in that chapter. Compare both the classification performance and interpretability of the final model.* The `spambase` data that is referred to in ELS can be found in the `nutshell` package. Unfortunately, this package is no longer active, but can be found in the CRAN archive section. I downloaded the tarball `nutshell_2.0.tar.gz` and then applied the following unix commands (on a mac) by

```
gunzip nutshell_2.0.tar.gz
tar xvf nutshell_2.0.tar
```

this created a `nutshell_2.0` directory and then in the `data` subdirectory of this you can find the `spambase.rda` data file containing the data.

Here is the R code I used. Note the scaling.

```
library("neuralnet") # Load in the neural networks library
load("spambase.rda") # Load in the spambase data

# Spambase is_spam is a factor, convert it to 0-1
spambase$is_spam <- as.numeric(spambase$is_spam)-1

# Generate training and test data set (using same numbers as in ELS)
train.index <- sample(1:nrow(spambase), 3065)
sp.train <- spambase[ train.index,]
sp.test <- spambase[-train.index,]

# Apply scaling, as per lectures
maxss <- as.numeric(apply(sp.train, 2, max))
minss <- as.numeric(apply(sp.test, 2,min))
sp.train.scaled <- as.data.frame(scale(sp.train, center=minss,
    scale=maxss-minss))

sp.test.scaled <- as.data.frame(scale(sp.test, center=minss,
    scale=maxss-minss))

# Get names of spambase
nm <- names(sp.train)

# Construct formula for the neural network
form <- as.formula(paste("is_spam ~",
    paste(nm[!nm %in% "is_spam"], collapse="+ ")))

# Train the neural network
sp.train.nn <- neuralnet(f=form, data=sp.train.scaled, hidden=c(5,3),
    linear.output=FALSE)

# Plot it
plot(sp.train.nn)

#
# Now compute predictions on test set

sp.test.pred <-compute(ans, sp.test.scaled[,1:57])$net.result


# Return cross-classified table
 sp.test.adj <- sp.test.pred
 # note, some predictions from compute a very small, but not zero.
 # Make them zero; and similarly for any that are close to 1,
 # but not exactly.
 sp.test.adj[sp.test.adj < 0.5] <- 0
 sp.test.adj[sp.test.adj > 0.5] <- 1
 table(sp.test.adj, sp.test[,58])
```

If you run this code, it plots the neural network and outputs the following table:

```
sp.test.adj   0   1
          0 896  32
          1  43 565
```

You can divide the table by 1536 to get the cross-classification rates:

```
 >round(100*ans/1536, 2)

sp.test.adj     0     1
          0 58.33  2.08
          1  2.80 36.78
```

The total misclassification rate is $(32 + 43)/1536 = 4.9\%$, which is a little better than the error rate of $5.3\%$ from the additive logistic regression model from ELS §9.1.2 and a bit better than the linear logistic regression mentioned there of $7.6\%$. However, this is but one neural network model, possibly a more complicated one, with more layers, might do better still.

More interesting, perhaps, is the note the simplicity of the additive model fit given in ELS Table 9.2. That model indicates only 16 significant predictors that achieve the rate of $5.3\%$ in an *additive* model - so pretty simple. Compare that to the monster that is presented by the neural network plot in Figure 1. For me, this example summarises neural networks. They *CAN* do a great job at prediction and do not need a lot of human intervention in fitting, although it can take time. However, the returned models are often extremely hard to interpret. This can make it harder to answer other forms of prediction questions — e.g. a simple 'what if' we change something.

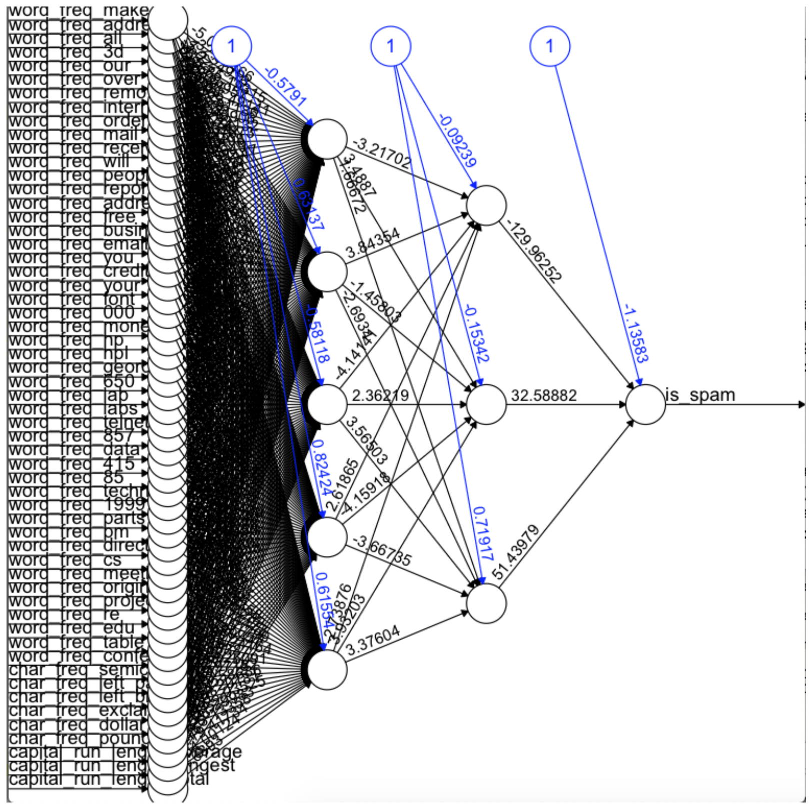

Figure 1: Neural network for the spambase data.