Module 11: Recommended Exercises

Statistical Learning V2021

alexaoh

22 mai, 2021

Problem 1

a)

Write down the equation that describes and input is related to output in this network, using general activation functions ϕ_o, ϕ_h and ϕ_{h^*} and bias nodes in all layers. What would you call such a network?

The given network is called a 4-4-4-3 feedforward network, with one bias node in each layer (except the output of course). The equation that describes the network can be written as shown in the following. The nodes in the first hidden layer are denoted by

$$z_m(x) = \phi_{h^*}(\alpha_{0m} + \sum_{j=1}^{4} \alpha_{jm} x_j), \quad m = 1, \dots, 4,$$

where α_{jm} is the weight from input j to hidden node m and α_{0m} is the bias term for the m^{th} hidden node. The nodes in the second hidden layer can be denoted in the similarly, e.g. with the letter $y_l(x), \quad l = 1, \dots 4$, with weights γ_{ij} and $l = 1, 2, 3, 4$ nodes. Finally, the output layer can be denoted by

$$y_c(\boldsymbol{x}) = \phi_o(\beta_{0c} + \sum_{l=1}^{4} \beta_{lc} y_l) = \phi_o(\beta_{0c} + \sum_{l=1}^{4} \beta_{lc} \phi_h(\gamma_{0l} + \sum_{m=1}^{4} \alpha_{ml} z_m) = \phi_o(\beta_{0c} + \sum_{l=1}^{4} \beta_{lc} \phi_h(\gamma_{0l} + \sum_{m=1}^{4} \alpha_{ml} \phi_{h^*}(\alpha_{0m} + \sum_{j=1}^{4} \alpha_{jm} x_j))),$$

for $c = 1, 2, 3$, i.e. all the output nodes.

b)

The following image is the illustration of an artificial neural network at Wikipedia.

- What can you say about this network architecture? *Answer*: It is feedforward. Moreover, the hidden layer has four nodes, but no bias node, since there are weights "coming into" each of the nodes. The input layer has either 3 input nodes or 2 input nodes and a bias node. The output layer has two nodes.
- What do you think it can be used for (regression/classification)? *Answer*: It can be used for binary or tertiary classification (depending on how the nodes are defined), since there are two nodes in the output layer. Softmax activation function can be used for two classes. However, for two classes, one output node is most commonly used. Moreover, it can be used for regression with two responses.

c)

What are the similarities and differences beween a feedforward neural network with one hidden layer with `linear` activation and `sigmoid` output (one output) and logistic regression?

- Similarities: The neural network gives approximately the same coefficients as the logistic regression, since the input is "squashed" into the interval $[0,1]$ when output from the network, because of the sigmoid activation function. They will not give the exact same coefficient estimates, since they use different algorithms to reach minimum.
- Differences: Use different algorithms. The neural network must estimate a lot more coefficients. The logistic regression is much better for interpretation compared to the neural network. However, the hidden layer may find latent structures in the data, with the help of the hidden layer, despite the activation function of the hidden layer being linear.

d)

In a feedforward neural network you may have $10'000$ weights to estimate but only 1000 observations. How is this possible?

This is possible because you may have 1000 observations in the input layer, but an arbitrarily deep neural network, with arbitrarily wide layers. This is possible to calculate because of the use of iterative methods (gradient descent and backpropagation), which means that there are no unique solutions. The network will benefit greatly by adding some sort of regularization, like weight decay and early stopping.

Problem 2

a)

Which network architecture and activation functions does this formula correspond to?

$$\hat{y}_1(\mathbf{x}) = \beta_{01} + \sum_{m=1}^{5} \beta_{m1} \cdot \max(\alpha_{0m} + \sum_{j=1}^{10} \alpha_{jm} x_j, 0)$$

The hidden layer has the ReLu activation function. Moreover, the network has 10 input nodes (in addition to a bias node) and one hidden layer, with 5 nodes (in addition to a bias node).

This formula corresponds to a feedforward network with one input layer (with 10 input nodes and one bias node), one hidden layer and one output layer with one node. The hidden layer has a ReLU activation function and 5 nodes in addition to a bias node.

How many parameters are estimated in this network?

Hence, in this network, $11 \cdot 5 + 6 = 61$ parameters are estimated.

b)

Which network architecture and activation functions does this formula give?

$$\hat{y}_1(\mathbf{x}) = (1 + \exp(-\beta_{01} - \sum_{m=1}^{5} \beta_{m1} \max(\gamma_{0m} + \sum_{l=1}^{10} \gamma_{lm} \max(\sum_{j=1}^{4} \alpha_{jl} x_j, 0), 0))^{-1}$$

This corresponds to a network with 4 input nodes, one hidden layer with 10 nodes (in addition to a bias node), another hidden layer with 5 nodes (in addition to a bias node) and one output node. The first and second hidden layers have the ReLU activation function, while the output layer has a sigmoid activation function.

How many parameters are estimated in this network?

Hence, in this network, $(4 \cdot 10) + (10 + 1) \cdot 5 + (5 + 1) = 101$ parameters are estimated.

c)

In a regression setting: Consider

- A sum of non-linear functions of each covariate in Module 7.
- A sum of many non-linear functions of sums of covariates in feedforward neural networks (one hidden layer, non-linear activation in hidden layer) in Module 11.

Explain how these two ways of thinking differ? Pros and cons?

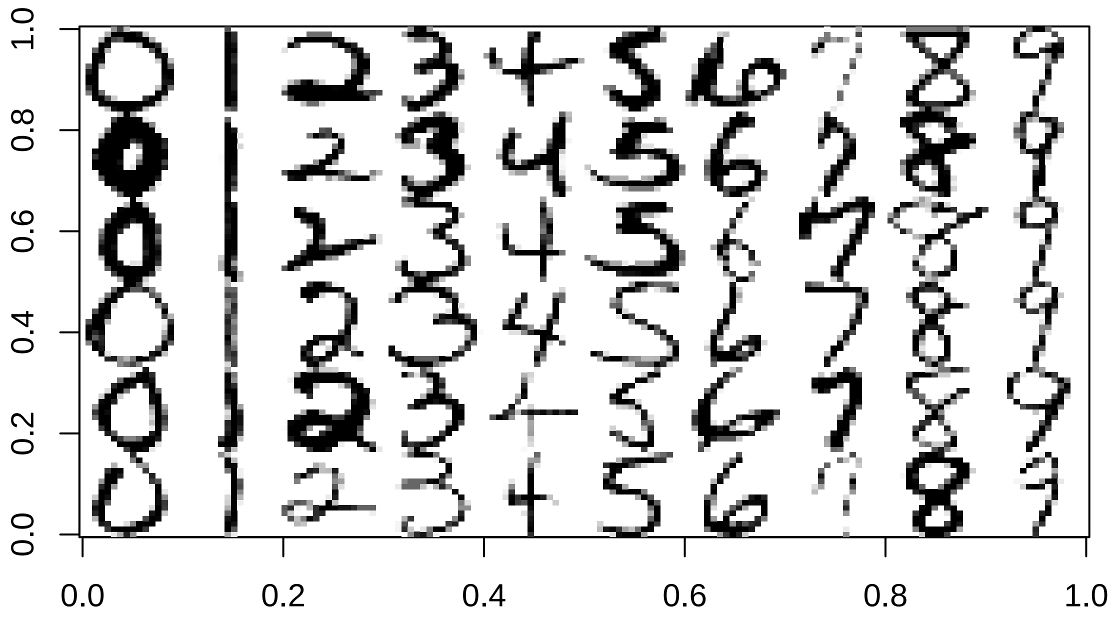
Answer: The first model is simpler, while the second is more complex. The second way gives two transformations of the data, while the first only transforms it once. Pros with the second way is that it opens to as many non-linear transformations of the data as one could desire, since one can add more hidden layers in the network. Cons with the second way is that it needs iterative solvers and it cannot be guaranteed that a global minimizer of the cost function is found. A pro with the first method is that the parameters in the model can be estimated precisely using linear algebra. Another important pro of the first method is that the estimated coefficients are easy to interpret and, hence, it can be used for inference, where as this is extremely hard (if not impossible) in the neural network setting. Interactions are automatically handled in the neural network setting, since there are non-linear functions of sums from the previous layers, while the interactions need to be added manually in the additive model in the first method. Lastly, a neural net with at least one hidden layer and a "squashing type" activation function, any function can be fit.

Problem 3: Handwritten digit recognition data

The following problem involves classification of handwritten digits from 0 to 9. For more details see `?zip.train` in `ElemStatLearn` R package or see Section 11.7 in the Book Elements of Statistical Learning.

Note: the `ElemStatLearn` package has been archived from CRAN, so to get ahold of the data, download the archived package from https://cran.r-project.org/src/contrib/Archive/ElemStatLearn/ElemStatLearn_2015.6.26.tar.gz, and then in RStudio, click the "Packages" tab, then click "Install", select "Install from: Package Archive File", and then select the `.tar.gz` file you downloaded above.

We have images of handwritten digits in grayscale of size 16×16 which are stored as vectors of size 256. There are 7291 training observations and 2007 test observations



a)

Preprocessing is important before we fit a neural network. Standardize the features and transform the digits to factors for both train and test data.

Remember that it is important to decide on the scale based on the training data, and then apply the same standardization for both training and test data (i.e., centering around the same mean and transforming to the same variance).

```r
library(ElemStatLearn)
train.data <- zip.train[, -1]
train.labels <- factor(zip.train[, 1])
test.data <- zip.test[, -1]
test.labels <- factor(zip.test[, 1])

# Standardize the data.
mean <- apply(train.data, 2, mean)
std <- apply(train.data, 2, sd)
train.data <- scale(train.data, center = mean, scale = std)
test.data <- scale(test.data, center = mean, scale = std)
```

b)

Use a feedforward-network with one hidden layer to train and predict using the `nnet()` function from the `nnet` R package. Include 5 hidden nodes plus a bias node in the input and hidden layers. What is the number of parameters? Look at the confusion table for the test data. How good does the network perfom?

```r
library(nnet)
fit.nnet <- nnet(train.labels ~ ., data = train.data, size = 5, MaxNWts = 3000,
    maxit = 5000)
```

```
#> # weights:  1345
#> initial  value 19347.364680
#> iter  10 value 5533.822230
#> iter  20 value 3291.297740
#> iter  30 value 2625.749957
#> iter  40 value 2109.777587
#> iter  50 value 1845.579112
#> iter  60 value 1655.091546
#> iter  70 value 1512.842843
#> iter  80 value 1436.527071
#> iter  90 value 1383.142541
#> iter 100 value 1332.222746
#> iter 110 value 1287.044270
#> iter 120 value 1250.262741
#> iter 130 value 1224.496257
#> iter 140 value 1206.172335
#> iter 150 value 1191.908854
#> iter 160 value 1178.074790
#> iter 170 value 1160.268426
#> iter 180 value 1145.890436
#> iter 190 value 1132.753611
#> iter 200 value 1119.320270
#> iter 210 value 1112.530692
#> iter 220 value 1096.483968
#> iter 230 value 1082.231422
#> iter 240 value 1072.790098
#> iter 250 value 1055.936926
#> iter 260 value 1046.252561
#> iter 270 value 1033.956200
#> iter 280 value 1024.669721
#> iter 290 value 1022.458399
#> iter 300 value 1019.243717
#> iter 310 value 1016.201499
#> iter 320 value 1013.971446
#> iter 330 value 1011.863315
#> iter 340 value 1009.692683
#> iter 350 value 1005.156060
#> iter 360 value 1001.529539
#> iter 370 value 998.778417
#> iter 380 value 996.365982
#> iter 390 value 992.813698
#> iter 400 value 991.415243
#> iter 410 value 990.367827
#> iter 420 value 989.023348
#> iter 430 value 987.337111
#> iter 440 value 984.401971
#> iter 450 value 978.956486
#> iter 460 value 973.937997
#> iter 470 value 971.398013
#> iter 480 value 968.565181
#> iter 490 value 966.334059
#> iter 500 value 965.297248
#> iter 510 value 963.842126
#> iter 520 value 960.288720
#> iter 530 value 959.035036
#> iter 540 value 953.662800
#> iter 550 value 950.103261
#> iter 560 value 948.079195
#> iter 570 value 946.318681
#> iter 580 value 945.146358
#> iter 590 value 943.555044
#> iter 600 value 941.906797
#> iter 610 value 937.743156
#> iter 620 value 936.416034
#> iter 630 value 935.465135
#> iter 640 value 934.826319
#> iter 650 value 934.426341
#> iter 660 value 934.111097
#> iter 670 value 933.861786
#> iter 680 value 933.341802
#> iter 690 value 932.725839
#> iter 700 value 931.715426
#> iter 710 value 930.755125
#> iter 720 value 928.507847
#> iter 730 value 927.178478
#> iter 740 value 925.796219
#> iter 750 value 924.346152
#> iter 760 value 922.809247
#> iter 770 value 919.019487
#> iter 780 value 918.368700
#> iter 790 value 917.918763
#> iter 800 value 917.729765
#> iter 810 value 917.565902
#> iter 820 value 917.384796
#> iter 830 value 917.249227
#> iter 840 value 917.118204
#> iter 850 value 916.964782
#> iter 860 value 916.711150
#> iter 870 value 915.441782
#> iter 880 value 915.098295
#> iter 890 value 914.790760
#> iter 900 value 914.679798
#> iter 910 value 914.597819
#> iter 920 value 914.526138
#> iter 930 value 914.418375
#> iter 940 value 914.330488
#> iter 950 value 914.233396
#> iter 960 value 914.140758
#> iter 970 value 914.032684
#> iter 980 value 913.663023
#> iter 990 value 913.479786
#> iter1000 value 913.314387
#> iter1010 value 913.230944
#> iter1020 value 913.172241
#> iter1030 value 913.071006
#> iter1040 value 912.963881
#> iter1050 value 912.768879
#> iter1060 value 912.585571
#> iter1070 value 912.399799
#> iter1080 value 912.318092
#> iter1090 value 912.094222
#> iter1100 value 911.990732
#> iter1110 value 911.880931
#> iter1120 value 911.610833
#> iter1130 value 911.138789
#> iter1140 value 910.161975
#> iter1150 value 909.872452
#> iter1160 value 909.742609
#> iter1170 value 909.649197
#> iter1180 value 909.528398
#> iter1190 value 909.407604
#> iter1200 value 909.275914
#> iter1210 value 909.186826
#> iter1220 value 909.103558
#> iter1230 value 909.027622
#> iter1240 value 905.571741
#> iter1250 value 905.050676
#> iter1260 value 904.701220
#> iter1270 value 904.580208
#> iter1280 value 904.493026
#> iter1290 value 904.391476
#> iter1300 value 904.214193
#> iter1310 value 904.112134
#> iter1320 value 904.029873
#> iter1330 value 903.956634
#> iter1340 value 903.892380
#> iter1350 value 903.845812
#> iter1360 value 903.783988
#> iter1370 value 903.703802
#> iter1380 value 903.578979
#> iter1390 value 902.600459
#> iter1400 value 902.011924
#> iter1410 value 901.689256
#> iter1420 value 901.350282
#> iter1430 value 901.192793
#> iter1440 value 900.929182
#> iter1450 value 900.727907
#> iter1460 value 900.600443
#> iter1470 value 900.547657
#> iter1480 value 900.489291
#> iter1490 value 900.288974
#> iter1500 value 899.928200
#> iter1510 value 899.757537
#> iter1520 value 899.625442
#> iter1530 value 899.574753
#> iter1540 value 899.531420
#> iter1550 value 899.490866
#> iter1560 value 899.434892
#> iter1570 value 898.748445
#> iter1580 value 897.013674
#> iter1590 value 896.769884
#> iter1600 value 896.683227
#> iter1610 value 896.626250
#> iter1620 value 896.575030
#> iter1630 value 896.498763
#> iter1640 value 896.401974
#> iter1650 value 880.253071
#> iter1660 value 878.637526
#> iter1670 value 878.444982
#> iter1680 value 878.255879
#> iter1690 value 878.046918
#> iter1700 value 877.783017
#> iter1710 value 877.490771
#> iter1720 value 876.166085
#> iter1730 value 875.883764
#> iter1740 value 875.709416
#> iter1750 value 875.620446
#> iter1760 value 875.565892
#> iter1770 value 875.524748
#> iter1780 value 875.487460
#> iter1790 value 875.456868
#> iter1800 value 875.417763
#> iter1810 value 875.371407
#> iter1820 value 875.307684
#> iter1830 value 875.260882
#> iter1840 value 875.230252
#> iter1850 value 875.183055
#> iter1860 value 875.136586
#> iter1870 value 875.094568
#> iter1880 value 875.054310
#> iter1890 value 875.010454
#> iter1900 value 874.962320
#> iter1910 value 874.910892
#> iter1920 value 874.852209
#> iter1930 value 874.787211
#> iter1940 value 874.637590
#> iter1950 value 874.504315
#> iter1960 value 874.254467
#> iter1970 value 874.133040
#> iter1980 value 873.376660
#> iter1990 value 873.320664
#> iter2000 value 873.287489
#> iter2010 value 873.271303
#> iter2020 value 873.257027
#> iter2030 value 873.245621
#> iter2040 value 873.231959
#> iter2050 value 873.222207
#> iter2060 value 873.209013
#> iter2070 value 873.195485
#> iter2080 value 873.185321
#> iter2090 value 873.174211
#> iter2100 value 873.143201
#> iter2110 value 873.124956
#> iter2120 value 873.109214
#> iter2130 value 873.090264
#> iter2140 value 873.069225
#> iter2150 value 873.046943
#> iter2160 value 873.023745
#> iter2170 value 873.002683
#> iter2180 value 872.974097
#> iter2190 value 872.934055
#> iter2200 value 872.883084
#> iter2210 value 872.781740
#> iter2220 value 872.752262
#> iter2230 value 872.732750
#> iter2240 value 872.722185
#> iter2250 value 872.710830
#> iter2260 value 872.697307
#> iter2270 value 872.685590
#> iter2280 value 872.675299
#> iter2290 value 872.666363
#> iter2300 value 872.657007
#> iter2310 value 872.646466
#> iter2320 value 872.637599
#> iter2330 value 872.628336
#> iter2340 value 872.616862
#> iter2350 value 872.603163
#> iter2360 value 872.590149
#> iter2370 value 872.578020
#> iter2380 value 872.563494
#> iter2390 value 872.544802
#> iter2400 value 872.527520
#> iter2410 value 872.513574
#> iter2420 value 872.495091
#> iter2430 value 872.466353
#> iter2440 value 872.431241
#> iter2450 value 872.381076
#> iter2460 value 872.312130
#> iter2470 value 872.241460
#> iter2480 value 872.154448
#> iter2490 value 872.015277
#> iter2500 value 871.850307
#> iter2510 value 871.732909
#> iter2520 value 871.590474
#> iter2530 value 871.417605
#> iter2540 value 871.038474
#> iter2550 value 870.899510
#> iter2560 value 870.804707
#> iter2570 value 870.769611
#> iter2580 value 870.753659
#> iter2590 value 870.739965
#> iter2600 value 870.723682
#> iter2610 value 870.701529
#> iter2620 value 870.681043
#> iter2630 value 870.666747
#> iter2640 value 870.656232
#> iter2650 value 870.646138
#> iter2660 value 870.636920
#> iter2670 value 870.621166
#> iter2680 value 870.612107
#> iter2690 value 870.604496
#> iter2700 value 870.596101
#> iter2710 value 870.588922
#> iter2720 value 870.581894
#> iter2730 value 870.576618
#> iter2740 value 870.558648
#> iter2750 value 870.553578
#> iter2760 value 870.549332
#> iter2770 value 870.546318
#> iter2780 value 870.544050
#> iter2790 value 870.542188
#> iter2800 value 870.540136
#> iter2810 value 870.538275
#> iter2820 value 870.536029
#> iter2830 value 870.534323
#> iter2840 value 870.532392
#> iter2850 value 870.529904
#> final  value 870.528351
#> converged
```

```r
summary(fit.nnet)
```

```
#> a 256-5-10 network with 1345 weights
#> options were - softmax modelling
#>    b->h1   i1->h1   i2->h1   i3->h1   i4->h1   i5->h1   i6->h1   i7->h1
#>   -26.66    -1.63     3.99   -15.06     7.09    -9.25    -8.47    -7.49
#>   i8->h1   i9->h1  i10->h1  i11->h1  i12->h1  i13->h1  i14->h1  i15->h1
#>    17.11     2.00   -17.02    10.22    -8.44     4.48    13.54    -6.95
#>  i16->h1  i17->h1  i18->h1  i19->h1  i20->h1  i21->h1  i22->h1  i23->h1
#>     0.19    -5.42     2.52    16.63   -32.74    -2.71   -16.08    -1.56
#>  i24->h1  i25->h1  i26->h1  i27->h1  i28->h1  i29->h1  i30->h1  i31->h1
#>     2.99    -3.63     4.56     3.69    13.77    16.45    -0.87    -6.32
#>  i32->h1  i33->h1  i34->h1  i35->h1  i36->h1  i37->h1  i38->h1  i39->h1
#>    -1.64    -6.15     0.02   -13.44    11.12    15.33   -12.78     4.11
#>  i40->h1  i41->h1  i42->h1  i43->h1  i44->h1  i45->h1  i46->h1  i47->h1
#>    13.28    -8.33    15.46    -4.01    20.50     9.87     5.69     2.21
#>  i48->h1  i49->h1  i50->h1  i51->h1  i52->h1  i53->h1  i54->h1  i55->h1
#>     3.66   -10.20    10.83    -9.91    -5.69   -12.36     3.27    -3.85
#>  i56->h1  i57->h1  i58->h1  i59->h1  i60->h1  i61->h1  i62->h1  i63->h1
#>     2.33     0.63    18.82   -12.84     0.65     4.50    -0.12    14.66
#>  i64->h1  i65->h1  i66->h1  i67->h1  i68->h1  i69->h1  i70->h1  i71->h1
#>    -2.75   -10.19     0.97     9.64    -9.40    -9.98    -6.68    -2.70
#>  i72->h1  i73->h1  i74->h1  i75->h1  i76->h1  i77->h1  i78->h1  i79->h1
#>    -4.77    21.22     0.75    17.77     6.74     0.72     0.74   -14.06
#>  i80->h1  i81->h1  i82->h1  i83->h1  i84->h1  i85->h1  i86->h1  i87->h1
#>    -1.27   -15.79    -6.23    -9.06   -14.03    -2.93    -8.81   -10.83
#>  i88->h1  i89->h1  i90->h1  i91->h1  i92->h1  i93->h1  i94->h1  i95->h1
#>    -6.38     6.56    -4.45    -1.74    10.66    24.77     3.39     2.89
#>  i96->h1  i97->h1  i98->h1  i99->h1 i100->h1 i101->h1 i102->h1 i103->h1
#>    -1.76     9.33    22.87     3.24     9.11   -10.19    -2.30    -5.49
#> i104->h1 i105->h1 i106->h1 i107->h1 i108->h1 i109->h1 i110->h1 i111->h1
#>    -6.43     6.21     1.91    14.73    16.24    -1.98    11.73    10.11
#> i112->h1 i113->h1 i114->h1 i115->h1 i116->h1 i117->h1 i118->h1 i119->h1
#>   -10.05   -13.73   -10.18     4.68   -19.27    -1.51   -16.21    -3.17
#> i120->h1 i121->h1 i122->h1 i123->h1 i124->h1 i125->h1 i126->h1 i127->h1
#>    -1.99   -10.35    -4.57    -3.09    -8.11    28.58   -16.61   -26.48
#> i128->h1 i129->h1 i130->h1 i131->h1 i132->h1 i133->h1 i134->h1 i135->h1
#>    11.38    -6.78     0.72     5.57    -1.86    -5.47    14.81    -9.41
#> i136->h1 i137->h1 i138->h1 i139->h1 i140->h1 i141->h1 i142->h1 i143->h1
#>     1.29     9.10    -8.75     8.10    14.37    -7.16    -8.52    11.87
#> i144->h1 i145->h1 i146->h1 i147->h1 i148->h1 i149->h1 i150->h1 i151->h1
#>    -4.66    -7.30     2.64     1.43     4.38    -2.68    15.65     3.95
#> i152->h1 i153->h1 i154->h1 i155->h1 i156->h1 i157->h1 i158->h1 i159->h1
#>     3.74    -4.67    19.63    -6.64     2.21    -0.78   -18.56   -21.68
#> i160->h1 i161->h1 i162->h1 i163->h1 i164->h1 i165->h1 i166->h1 i167->h1
#>    -4.71    10.81    -4.40    15.61     0.55    -3.17    12.77    18.54
#> i168->h1 i169->h1 i170->h1 i171->h1 i172->h1 i173->h1 i174->h1 i175->h1
#>     0.52    10.23     8.45    -0.87    -3.46   -10.43     0.64   -17.76
#> i176->h1 i177->h1 i178->h1 i179->h1 i180->h1 i181->h1 i182->h1 i183->h1
#>    -0.74    -6.49     7.19     5.21    11.51     5.16   -23.77    22.35
#> i184->h1 i185->h1 i186->h1 i187->h1 i188->h1 i189->h1 i190->h1 i191->h1
#>     0.80     0.91    -7.16     5.72   -24.51   -12.71   -20.31    -6.88
#> i192->h1 i193->h1 i194->h1 i195->h1 i196->h1 i197->h1 i198->h1 i199->h1
#>    -3.07    13.30    14.33     7.86    -8.05     2.50    12.67   -14.87
#> i200->h1 i201->h1 i202->h1 i203->h1 i204->h1 i205->h1 i206->h1 i207->h1
#>    32.32    -1.94   -11.23   -19.67    -3.92    12.09   -20.91     1.07
#> i208->h1 i209->h1 i210->h1 i211->h1 i212->h1 i213->h1 i214->h1 i215->h1
#>   -21.43    -8.08     7.95    27.11    -4.17    15.45   -20.70   -26.10
#> i216->h1 i217->h1 i218->h1 i219->h1 i220->h1 i221->h1 i222->h1 i223->h1
#>     6.12    14.93   -14.80    -0.75   -11.21   -26.11    21.31    -6.44
#> i224->h1 i225->h1 i226->h1 i227->h1 i228->h1 i229->h1 i230->h1 i231->h1
#>    -4.86    14.27    12.05     8.15    -0.98     7.81     3.66     0.15
#> i232->h1 i233->h1 i234->h1 i235->h1 i236->h1 i237->h1 i238->h1 i239->h1
#>   -27.39    -9.90     7.71   -30.20    27.54    -5.92     7.01   -16.34
#> i240->h1 i241->h1 i242->h1 i243->h1 i244->h1 i245->h1 i246->h1 i247->h1
#>    12.27     2.48    10.65    -3.06    -3.41    28.96   -33.75    -7.00
#> i248->h1 i249->h1 i250->h1 i251->h1 i252->h1 i253->h1 i254->h1 i255->h1
#>   -19.19   -12.75   -11.62   -11.47   -20.13     2.20    -3.27    13.99
#> i256->h1
#>     5.68
#>    b->h2   i1->h2   i2->h2   i3->h2   i4->h2   i5->h2   i6->h2   i7->h2
#>    52.88     7.17   -25.70     8.49    18.26   -13.03    -5.76    16.10
#>   i8->h2   i9->h2  i10->h2  i11->h2  i12->h2  i13->h2  i14->h2  i15->h2
#>    14.45    37.05    -0.60    16.63     6.60    -6.38     5.92    15.64
#>  i16->h2  i17->h2  i18->h2  i19->h2  i20->h2  i21->h2  i22->h2  i23->h2
#>    -9.09    16.96     5.98   -35.98    13.42   -14.97    13.14    -7.82
#>  i24->h2  i25->h2  i26->h2  i27->h2  i28->h2  i29->h2  i30->h2  i31->h2
#>     7.71    22.33    20.06    15.52     0.23     4.11    -8.85    -3.49
#>  i32->h2  i33->h2  i34->h2  i35->h2  i36->h2  i37->h2  i38->h2  i39->h2
#>    13.45   -18.05    -5.86    -0.41   -14.66    -3.90    -8.56    -2.56
#>  i40->h2  i41->h2  i42->h2  i43->h2  i44->h2  i45->h2  i46->h2  i47->h2
#>    20.53    -3.63     7.50    17.52     5.91    29.91    -7.19     0.35
#>  i48->h2  i49->h2  i50->h2  i51->h2  i52->h2  i53->h2  i54->h2  i55->h2
#>    15.24    -5.66   -16.82     5.10    -5.57    10.28     7.85    -4.91
#>  i56->h2  i57->h2  i58->h2  i59->h2  i60->h2  i61->h2  i62->h2  i63->h2
#>    -9.99    48.05    28.85     6.75    -9.65   -14.50    12.53     8.73
#>  i64->h2  i65->h2  i66->h2  i67->h2  i68->h2  i69->h2  i70->h2  i71->h2
#>    -9.73     9.16    -9.45   -18.21   -18.24     1.75   -26.05     4.00
#>  i72->h2  i73->h2  i74->h2  i75->h2  i76->h2  i77->h2  i78->h2  i79->h2
#>   -15.42    -5.09    39.10    36.90    56.80    43.54    16.21   -10.82
#>  i80->h2  i81->h2  i82->h2  i83->h2  i84->h2  i85->h2  i86->h2  i87->h2
#>    17.51    -2.64    -2.73    19.23   -25.22    -5.14   -18.83   -11.69
#>  i88->h2  i89->h2  i90->h2  i91->h2  i92->h2  i93->h2  i94->h2  i95->h2
#>   -11.93    -2.18     0.41    13.86    20.33    17.83    39.04    -1.16
#>  i96->h2  i97->h2  i98->h2  i99->h2 i100->h2 i101->h2 i102->h2 i103->h2
#>    36.97   -13.02    17.01   -11.84     1.22   -13.90    -0.21    -0.85
#> i104->h2 i105->h2 i106->h2 i107->h2 i108->h2 i109->h2 i110->h2 i111->h2
#>   -27.93     8.25    -2.64    -1.92    -4.33     6.74    -4.73    33.73
#> i112->h2 i113->h2 i114->h2 i115->h2 i116->h2 i117->h2 i118->h2 i119->h2
#>   -26.41   -28.92   -29.05   -32.57    -9.37    21.32    -4.32     8.00
#> i120->h2 i121->h2 i122->h2 i123->h2 i124->h2 i125->h2 i126->h2 i127->h2
#>    10.03   -12.62     8.58    28.07    -6.93     1.81    -2.97    -4.77
#> i128->h2 i129->h2 i130->h2 i131->h2 i132->h2 i133->h2 i134->h2 i135->h2
#>    27.60    26.85    10.60    26.83    19.24    -8.93   -18.29    14.89
#> i136->h2 i137->h2 i138->h2 i139->h2 i140->h2 i141->h2 i142->h2 i143->h2
#>   -20.27   -24.94   -19.41   -14.34    11.64   -10.35    28.24    17.43
#> i144->h2 i145->h2 i146->h2 i147->h2 i148->h2 i149->h2 i150->h2 i151->h2
#>     4.53    19.96     0.74    -8.78   -22.97     3.55    13.32     8.07
#> i152->h2 i153->h2 i154->h2 i155->h2 i156->h2 i157->h2 i158->h2 i159->h2
#>   -17.59   -23.70    17.37    -1.09    -0.55    -7.81    16.97   -48.09
#> i160->h2 i161->h2 i162->h2 i163->h2 i164->h2 i165->h2 i166->h2 i167->h2
#>    10.13    -5.64    -6.71     6.25    30.41     9.16    -2.73   -11.52
#> i168->h2 i169->h2 i170->h2 i171->h2 i172->h2 i173->h2 i174->h2 i175->h2
#>   -22.15     9.75   -33.98    11.06   -18.23    10.68   -22.08    27.92
#> i176->h2 i177->h2 i178->h2 i179->h2 i180->h2 i181->h2 i182->h2 i183->h2
#>   -17.64     8.98    32.76   -15.82     3.84    -8.68    25.60    10.47
#> i184->h2 i185->h2 i186->h2 i187->h2 i188->h2 i189->h2 i190->h2 i191->h2
#>   -31.47    -6.58   -17.67    -4.32   -20.26     4.60    -6.48    -7.43
#> i192->h2 i193->h2 i194->h2 i195->h2 i196->h2 i197->h2 i198->h2 i199->h2
#>     3.55    15.21   -15.71     7.90    12.00     7.93    -9.22    -0.63
#> i200->h2 i201->h2 i202->h2 i203->h2 i204->h2 i205->h2 i206->h2 i207->h2
#>    16.86   -15.30     0.61   -23.31     1.19   -26.35   -12.38     1.78
#> i208->h2 i209->h2 i210->h2 i211->h2 i212->h2 i213->h2 i214->h2 i215->h2
#>     4.90     6.48   -15.70     7.00    -7.39    14.55    -3.94    14.26
#> i216->h2 i217->h2 i218->h2 i219->h2 i220->h2 i221->h2 i222->h2 i223->h2
#>   -29.57     4.76   -12.05    -6.73    -5.07   -15.31    -2.44    -7.05
#> i224->h2 i225->h2 i226->h2 i227->h2 i228->h2 i229->h2 i230->h2 i231->h2
#>    -7.45    11.84     6.33    21.18     7.60    25.66    25.57    -7.38
#> i232->h2 i233->h2 i234->h2 i235->h2 i236->h2 i237->h2 i238->h2 i239->h2
#>    13.30   -21.51   -14.27   -15.53   -30.04     4.94   -14.62   -11.70
#> i240->h2 i241->h2 i242->h2 i243->h2 i244->h2 i245->h2 i246->h2 i247->h2
#>   -17.72    20.65   -16.80     5.94     6.66     4.40   -23.49    24.27
#> i248->h2 i249->h2 i250->h2 i251->h2 i252->h2 i253->h2 i254->h2 i255->h2
#>    -1.28     2.46    -7.86   -17.06   -10.70     0.68     6.60    21.34
#> i256->h2
#>    -6.77
#>    b->h3   i1->h3   i2->h3   i3->h3   i4->h3   i5->h3   i6->h3   i7->h3
#>    23.57     8.09     0.35    -2.15   -14.72    -0.72   -11.43    -7.88
#>   i8->h3   i9->h3  i10->h3  i11->h3  i12->h3  i13->h3  i14->h3  i15->h3
#>   -27.22    -4.30   -11.92    -9.45    -1.46   -21.77    -8.26    -1.86
#>  i16->h3  i17->h3  i18->h3  i19->h3  i20->h3  i21->h3  i22->h3  i23->h3
#>    11.83     9.89   -28.10     2.00    -4.50   -10.07    -7.44     6.26
#>  i24->h3  i25->h3  i26->h3  i27->h3  i28->h3  i29->h3  i30->h3  i31->h3
#>   -37.39    -7.11     7.80   -16.80    -5.35     7.58     2.64    -5.86
#>  i32->h3  i33->h3  i34->h3  i35->h3  i36->h3  i37->h3  i38->h3  i39->h3
#>     5.05    -9.02    -5.75    -0.42     4.28     8.15    -1.36    -3.06
#>  i40->h3  i41->h3  i42->h3  i43->h3  i44->h3  i45->h3  i46->h3  i47->h3
#>     0.09   -21.38    -8.02    -5.97     6.03    -8.81     7.61     3.36
#>  i48->h3  i49->h3  i50->h3  i51->h3  i52->h3  i53->h3  i54->h3  i55->h3
#>    -4.87    -4.52    16.72   -14.83     5.99    -8.15   -16.29    -2.94
#>  i56->h3  i57->h3  i58->h3  i59->h3  i60->h3  i61->h3  i62->h3  i63->h3
#>   -44.14     7.57     3.36    -2.26     3.54     3.57   -11.74    -0.95
#>  i64->h3  i65->h3  i66->h3  i67->h3  i68->h3  i69->h3  i70->h3  i71->h3
#>   -17.62   -15.04    -9.65    -6.88    -6.85    19.88    17.76    21.41
#>  i72->h3  i73->h3  i74->h3  i75->h3  i76->h3  i77->h3  i78->h3  i79->h3
#>   -14.34   -11.51     3.59     8.91    20.67    -4.15    16.91    -6.16
#>  i80->h3  i81->h3  i82->h3  i83->h3  i84->h3  i85->h3  i86->h3  i87->h3
#>   -22.66   -14.90     3.29    30.31    -3.46   -10.48    -2.90     8.23
#>  i88->h3  i89->h3  i90->h3  i91->h3  i92->h3  i93->h3  i94->h3  i95->h3
#>     9.32    -6.95     3.04     2.43   -11.16     8.92    20.44    12.25
#>  i96->h3  i97->h3  i98->h3  i99->h3 i100->h3 i101->h3 i102->h3 i103->h3
#>    16.46     6.87     2.96   -34.58    12.03     3.02    13.37    -2.36
#> i104->h3 i105->h3 i106->h3 i107->h3 i108->h3 i109->h3 i110->h3 i111->h3
#>    -2.63    -2.11     5.69    -0.02    -2.72     2.24    -2.52     2.05
#> i112->h3 i113->h3 i114->h3 i115->h3 i116->h3 i117->h3 i118->h3 i119->h3
#>    -0.31    -7.95    15.31    17.03    16.13    13.83   -12.12    10.91
#> i120->h3 i121->h3 i122->h3 i123->h3 i124->h3 i125->h3 i126->h3 i127->h3
#>   -10.40     5.63   -18.68    -2.67     2.46    -2.63     4.72    17.59
#> i128->h3 i129->h3 i130->h3 i131->h3 i132->h3 i133->h3 i134->h3 i135->h3
#>    21.93    -3.45    -5.02   -13.04    -1.24   -13.88     0.58     9.71
#> i136->h3 i137->h3 i138->h3 i139->h3 i140->h3 i141->h3 i142->h3 i143->h3
#>    -2.34     7.58    -6.94    -4.68    -1.21     6.21    19.56   -10.51
#> i144->h3 i145->h3 i146->h3 i147->h3 i148->h3 i149->h3 i150->h3 i151->h3
#>    16.17    19.03    16.94    12.39    19.93    14.62     4.45    -6.56
#> i152->h3 i153->h3 i154->h3 i155->h3 i156->h3 i157->h3 i158->h3 i159->h3
#>    16.20    -1.30    -3.43    16.08   -12.96    -9.94     2.51   -19.76
#> i160->h3 i161->h3 i162->h3 i163->h3 i164->h3 i165->h3 i166->h3 i167->h3
#>    13.70   -15.84     6.44    25.85     6.04    40.15   -21.92    14.81
#> i168->h3 i169->h3 i170->h3 i171->h3 i172->h3 i173->h3 i174->h3 i175->h3
#>    -2.64     2.65    -0.63    -4.57     5.97     7.57   -20.98    -2.52
#> i176->h3 i177->h3 i178->h3 i179->h3 i180->h3 i181->h3 i182->h3 i183->h3
#>   -20.42     8.73    15.34    16.87    23.83    20.39    60.99    21.76
#> i184->h3 i185->h3 i186->h3 i187->h3 i188->h3 i189->h3 i190->h3 i191->h3
#>     6.96    10.91    22.85     3.98    -2.88   -12.21   -10.42   -11.08
#> i192->h3 i193->h3 i194->h3 i195->h3 i196->h3 i197->h3 i198->h3 i199->h3
#>    -1.76    26.99    22.61    -1.88     6.53    -7.90     8.81    -6.77
#> i200->h3 i201->h3 i202->h3 i203->h3 i204->h3 i205->h3 i206->h3 i207->h3
#>   -21.41     0.23   -17.55    -9.43   -15.86   -12.95    -3.23     6.15
#> i208->h3 i209->h3 i210->h3 i211->h3 i212->h3 i213->h3 i214->h3 i215->h3
#>   -17.08   -34.88   -10.00    16.86    30.64    31.13   -14.79    -6.96
#> i216->h3 i217->h3 i218->h3 i219->h3 i220->h3 i221->h3 i222->h3 i223->h3
#>    -9.26   -12.92    -2.70    -0.25    -2.04   -19.34   -10.66   -16.09
#> i224->h3 i225->h3 i226->h3 i227->h3 i228->h3 i229->h3 i230->h3 i231->h3
#>     2.30     9.40    14.40     7.21     9.26   -29.35     7.91     7.81
#> i232->h3 i233->h3 i234->h3 i235->h3 i236->h3 i237->h3 i238->h3 i239->h3
#>    18.00   -15.61     7.70    -4.45    -2.24    13.41    -1.36   -17.69
#> i240->h3 i241->h3 i242->h3 i243->h3 i244->h3 i245->h3 i246->h3 i247->h3
#>     6.65     3.62    12.69     0.25    15.79    -8.01    19.98    -8.44
#> i248->h3 i249->h3 i250->h3 i251->h3 i252->h3 i253->h3 i254->h3 i255->h3
#>    -2.99    20.03   -14.84   -13.39    -0.94   -10.75     1.31     5.51
#> i256->h3
#>     7.69
#>    b->h4   i1->h4   i2->h4   i3->h4   i4->h4   i5->h4   i6->h4   i7->h4
#>    36.36     3.91    -2.74     8.71   -17.76     4.26    -8.19     4.18
#>   i8->h4   i9->h4  i10->h4  i11->h4  i12->h4  i13->h4  i14->h4  i15->h4
#>    -1.01   -11.40    23.62    -8.58    23.55    -4.47    -2.77    20.28
#>  i16->h4  i17->h4  i18->h4  i19->h4  i20->h4  i21->h4  i22->h4  i23->h4
#>    -6.23    -1.07    21.90    -2.62    -7.97    -7.12   -11.36    -9.54
#>  i24->h4  i25->h4  i26->h4  i27->h4  i28->h4  i29->h4  i30->h4  i31->h4
#>    -8.06    11.27     1.05    19.60    21.08    24.32    19.16     5.64
#>  i32->h4  i33->h4  i34->h4  i35->h4  i36->h4  i37->h4  i38->h4  i39->h4
#>   -20.88    15.82    -8.04    -4.39   -30.34    -7.28   -11.38     5.24
#>  i40->h4  i41->h4  i42->h4  i43->h4  i44->h4  i45->h4  i46->h4  i47->h4
#>    -3.57     4.59     5.54     0.48    14.53     6.34    13.48   -15.85
#>  i48->h4  i49->h4  i50->h4  i51->h4  i52->h4  i53->h4  i54->h4  i55->h4
#>    23.23     2.40   -25.62     7.21   -21.06   -15.13   -19.61    10.16
#>  i56->h4  i57->h4  i58->h4  i59->h4  i60->h4  i61->h4  i62->h4  i63->h4
#>     1.33    -5.50     3.27    11.02    18.25    29.25    18.56    24.33
#>  i64->h4  i65->h4  i66->h4  i67->h4  i68->h4  i69->h4  i70->h4  i71->h4
#>     3.90     8.68   -17.96   -19.54     2.15    -2.71   -12.33   -33.84
#>  i72->h4  i73->h4  i74->h4  i75->h4  i76->h4  i77->h4  i78->h4  i79->h4
#>    -1.20   -15.34   -13.73    13.46     0.99   -18.11    10.21    -3.65
#>  i80->h4  i81->h4  i82->h4  i83->h4  i84->h4  i85->h4  i86->h4  i87->h4
#>    16.81    -9.67    17.24   -26.13   -24.71   -27.51   -10.34    -6.96
#>  i88->h4  i89->h4  i90->h4  i91->h4  i92->h4  i93->h4  i94->h4  i95->h4
#>    -9.95     2.44    -8.49    -5.29     3.57    27.14     6.23     6.03
#>  i96->h4  i97->h4  i98->h4  i99->h4 i100->h4 i101->h4 i102->h4 i103->h4
#>     0.15    13.11   -31.54    13.30     5.29   -18.47   -17.55    -7.15
#> i104->h4 i105->h4 i106->h4 i107->h4 i108->h4 i109->h4 i110->h4 i111->h4
#>   -11.74     3.73     9.51    -5.97     0.32    -7.51     3.48   -15.88
#> i112->h4 i113->h4 i114->h4 i115->h4 i116->h4 i117->h4 i118->h4 i119->h4
#>   -51.66   -16.73   -38.68   -48.80   -39.73   -12.83   -14.19    -0.23
#> i120->h4 i121->h4 i122->h4 i123->h4 i124->h4 i125->h4 i126->h4 i127->h4
#>    13.97    -3.63    -8.67     4.28   -12.27   -24.06   -19.36    11.82
#> i128->h4 i129->h4 i130->h4 i131->h4 i132->h4 i133->h4 i134->h4 i135->h4
#>    -8.28   -12.59     5.09   -22.30     5.12     1.74    -1.35    12.68
#> i136->h4 i137->h4 i138->h4 i139->h4 i140->h4 i141->h4 i142->h4 i143->h4
#>    -1.31    12.23    -3.68    -0.69     0.24    19.69    -7.69   -17.03
#> i144->h4 i145->h4 i146->h4 i147->h4 i148->h4 i149->h4 i150->h4 i151->h4
#>    16.44   -19.53   -15.83    19.97     0.28     4.61   -12.74     3.15
#> i152->h4 i153->h4 i154->h4 i155->h4 i156->h4 i157->h4 i158->h4 i159->h4
#>    36.34   -13.87     2.63   -27.18    -2.75   -20.63   -16.38    14.44
#> i160->h4 i161->h4 i162->h4 i163->h4 i164->h4 i165->h4 i166->h4 i167->h4
#>     0.60    25.64   -24.31    25.41    -5.66    20.53     1.51    10.96
#> i168->h4 i169->h4 i170->h4 i171->h4 i172->h4 i173->h4 i174->h4 i175->h4
#>    -1.87    25.72     9.22    16.86    -4.75     7.69     4.54    32.30
#> i176->h4 i177->h4 i178->h4 i179->h4 i180->h4 i181->h4 i182->h4 i183->h4
#>    -1.58    45.69    43.06    10.30     2.79     7.83     7.33    12.48
#> i184->h4 i185->h4 i186->h4 i187->h4 i188->h4 i189->h4 i190->h4 i191->h4
#>    10.23     0.65    -7.38   -11.15    -3.47   -23.22   -11.80   -35.35
#> i192->h4 i193->h4 i194->h4 i195->h4 i196->h4 i197->h4 i198->h4 i199->h4
#>    -9.64     6.20    -6.87    -4.94    44.62   -12.05     9.81     3.65
#> i200->h4 i201->h4 i202->h4 i203->h4 i204->h4 i205->h4 i206->h4 i207->h4
#>   -14.06    10.08    13.69     1.05   -17.76   -20.63    -4.23   -24.06
#> i208->h4 i209->h4 i210->h4 i211->h4 i212->h4 i213->h4 i214->h4 i215->h4
#>    -5.50    23.14    22.50    10.90    17.27    35.88    38.19     9.54
#> i216->h4 i217->h4 i218->h4 i219->h4 i220->h4 i221->h4 i222->h4 i223->h4
#>    12.94   -28.89   -13.50    10.91     1.94     1.65   -34.98     3.43
#> i224->h4 i225->h4 i226->h4 i227->h4 i228->h4 i229->h4 i230->h4 i231->h4
#>    -4.60    10.31    18.67    -0.11    47.85    19.06     6.75    21.98
#> i232->h4 i233->h4 i234->h4 i235->h4 i236->h4 i237->h4 i238->h4 i239->h4
#>     8.94    30.00   -15.03   -32.57   -16.68     2.18    -0.27    13.02
#> i240->h4 i241->h4 i242->h4 i243->h4 i244->h4 i245->h4 i246->h4 i247->h4
#>    -9.76    12.60   -19.39    33.07    27.73    12.54    18.84    11.75
#> i248->h4 i249->h4 i250->h4 i251->h4 i252->h4 i253->h4 i254->h4 i255->h4
#>    -1.56     0.33    15.57     7.82   -16.39    -7.30    -1.82    -8.94
#> i256->h4
#>    -0.34
#>    b->h5   i1->h5   i2->h5   i3->h5   i4->h5   i5->h5   i6->h5   i7->h5
#>    45.63    -5.14    -0.01     7.15     2.11    -8.88     3.80    -2.75
#>   i8->h5   i9->h5  i10->h5  i11->h5  i12->h5  i13->h5  i14->h5  i15->h5
#>    -8.11    -3.84     2.76    -9.62   -11.88    -3.00     0.15    -6.59
#>  i16->h5  i17->h5  i18->h5  i19->h5  i20->h5  i21->h5  i22->h5  i23->h5
#>    -6.63     4.60    -5.81    -1.01     6.97    28.45    -1.78     8.82
#>  i24->h5  i25->h5  i26->h5  i27->h5  i28->h5  i29->h5  i30->h5  i31->h5
#>     6.84    -2.95    -4.44     2.30     6.05     0.83    -1.73     9.89
#>  i32->h5  i33->h5  i34->h5  i35->h5  i36->h5  i37->h5  i38->h5  i39->h5
#>    -2.96     7.85    -5.20     7.97   -10.08     6.30     3.30     4.26
#>  i40->h5  i41->h5  i42->h5  i43->h5  i44->h5  i45->h5  i46->h5  i47->h5
#>    -7.74     2.53     8.35    -6.92   -11.89    -9.97   -12.17    11.44
#>  i48->h5  i49->h5  i50->h5  i51->h5  i52->h5  i53->h5  i54->h5  i55->h5
#>   -12.74     7.68     8.06    -9.03     4.19     3.73     1.79     1.90
#>  i56->h5  i57->h5  i58->h5  i59->h5  i60->h5  i61->h5  i62->h5  i63->h5
#>    19.82     1.84    -2.67   -14.38    -0.15    -7.09    12.06   -17.80
#>  i64->h5  i65->h5  i66->h5  i67->h5  i68->h5  i69->h5  i70->h5  i71->h5
#>    -4.99    -3.63     9.16   -13.02    -5.54    -5.26     9.04    -3.37
#>  i72->h5  i73->h5  i74->h5  i75->h5  i76->h5  i77->h5  i78->h5  i79->h5
#>     4.23     7.67   -12.10     4.64   -13.50    -5.01   -32.72   -11.40
#>  i80->h5  i81->h5  i82->h5  i83->h5  i84->h5  i85->h5  i86->h5  i87->h5
#>     6.33     3.44     4.46    -4.32     6.11    -1.56    -4.82    -7.78
#>  i88->h5  i89->h5  i90->h5  i91->h5  i92->h5  i93->h5  i94->h5  i95->h5
#>   -17.35     4.50     5.39    -9.72     1.36   -21.98    15.91     2.49
#>  i96->h5  i97->h5  i98->h5  i99->h5 i100->h5 i101->h5 i102->h5 i103->h5
#>    -8.74    -1.07   -15.47    13.83   -13.18     8.88   -14.69     1.53
#> i104->h5 i105->h5 i106->h5 i107->h5 i108->h5 i109->h5 i110->h5 i111->h5
#>    -4.02   -15.35   -16.20     3.14   -12.65   -12.02   -19.24    -8.33
#> i112->h5 i113->h5 i114->h5 i115->h5 i116->h5 i117->h5 i118->h5 i119->h5
#>     8.81     6.96     8.43     5.92    -2.70     1.61     7.17    -5.15
#> i120->h5 i121->h5 i122->h5 i123->h5 i124->h5 i125->h5 i126->h5 i127->h5
#>    -9.59    -3.43   -10.36    -2.42     5.73    -5.02     4.00     2.83
#> i128->h5 i129->h5 i130->h5 i131->h5 i132->h5 i133->h5 i134->h5 i135->h5
#>     7.18     7.01   -18.36   -14.50     1.58     7.60    -3.29   -13.77
#> i136->h5 i137->h5 i138->h5 i139->h5 i140->h5 i141->h5 i142->h5 i143->h5
#>    -4.41   -12.88    -0.18     1.90     1.29     3.91     6.72    14.71
#> i144->h5 i145->h5 i146->h5 i147->h5 i148->h5 i149->h5 i150->h5 i151->h5
#>    -9.75    15.03    -7.40   -11.82    -0.89   -15.55     2.52     2.77
#> i152->h5 i153->h5 i154->h5 i155->h5 i156->h5 i157->h5 i158->h5 i159->h5
#>    -1.94    -6.11    -8.08     4.22    -8.33    14.84    11.95    -9.41
#> i160->h5 i161->h5 i162->h5 i163->h5 i164->h5 i165->h5 i166->h5 i167->h5
#>    12.30    21.55     6.76    -8.68     8.39    -2.19     6.40   -10.02
#> i168->h5 i169->h5 i170->h5 i171->h5 i172->h5 i173->h5 i174->h5 i175->h5
#>   -10.39    14.40   -11.02     5.93    16.76    -0.64     6.28    13.92
#> i176->h5 i177->h5 i178->h5 i179->h5 i180->h5 i181->h5 i182->h5 i183->h5
#>    -8.92    25.22    -5.21    19.09   -21.00    -1.03    -8.92     7.94
#> i184->h5 i185->h5 i186->h5 i187->h5 i188->h5 i189->h5 i190->h5 i191->h5
#>     0.38   -21.75     6.73    -0.40    -3.72    11.45     1.82     2.13
#> i192->h5 i193->h5 i194->h5 i195->h5 i196->h5 i197->h5 i198->h5 i199->h5
#>     0.82    -7.22     5.74    -0.22     5.48     4.96    14.51    -8.17
#> i200->h5 i201->h5 i202->h5 i203->h5 i204->h5 i205->h5 i206->h5 i207->h5
#>    -1.95     6.74     0.30    22.28     8.15    15.76     0.58    -1.34
#> i208->h5 i209->h5 i210->h5 i211->h5 i212->h5 i213->h5 i214->h5 i215->h5
#>     0.75     0.81    22.40   -13.31    15.28     0.52    -0.16    16.17
#> i216->h5 i217->h5 i218->h5 i219->h5 i220->h5 i221->h5 i222->h5 i223->h5
#>    15.68     9.25     3.51   -13.88     1.22    14.31    -0.59     1.20
#> i224->h5 i225->h5 i226->h5 i227->h5 i228->h5 i229->h5 i230->h5 i231->h5
#>    21.32     5.70     3.44     4.77    -8.36     2.23    15.46    -7.99
#> i232->h5 i233->h5 i234->h5 i235->h5 i236->h5 i237->h5 i238->h5 i239->h5
#>    -1.95    -2.96    12.02     0.28     8.89     9.43    -8.16    15.30
#> i240->h5 i241->h5 i242->h5 i243->h5 i244->h5 i245->h5 i246->h5 i247->h5
#>     2.03    -3.05    -8.70    -5.94    17.22    -3.82     2.78    -1.30
#> i248->h5 i249->h5 i250->h5 i251->h5 i252->h5 i253->h5 i254->h5 i255->h5
#>    13.79     7.19     4.58    11.28     2.57    14.05    13.00    -0.37
#> i256->h5
#>    -0.43
#>   b->o1  h1->o1  h2->o1  h3->o1  h4->o1  h5->o1
#>   16.26   -5.74   40.08   20.84   -6.57   38.52
#>   b->o2  h1->o2  h2->o2  h3->o2  h4->o2  h5->o2
#>   19.12    6.35 -221.29 -122.50    5.99   29.11
#>   b->o3  h1->o3  h2->o3  h3->o3  h4->o3  h5->o3
#>   16.22    4.59   38.92   14.98    3.40   36.69
#>   b->o4  h1->o4  h2->o4  h3->o4  h4->o4  h5->o4
#> -187.06   -2.57  252.32    8.87    1.80   32.87
#>   b->o5  h1->o5  h2->o5  h3->o5  h4->o5  h5->o5
#>   27.18   -0.10   32.27   16.29   -3.48   29.56
#>   b->o6  h1->o6  h2->o6  h3->o6  h4->o6  h5->o6
#>   29.73   -7.16   33.43   11.28   -3.63   35.24
#>   b->o7  h1->o7  h2->o7  h3->o7  h4->o7  h5->o7
#>   16.86   -5.43 -301.60   21.21   -0.04   39.92
#>   b->o8  h1->o8  h2->o8  h3->o8  h4->o8  h5->o8
#>   12.30    7.37   47.94    7.39    3.05   25.78
#>   b->o9  h1->o9  h2->o9  h3->o9  h4->o9  h5->o9
#>   24.68   -3.64   36.38   16.28    4.21   27.25
#>  b->o10 h1->o10 h2->o10 h3->o10 h4->o10 h5->o10
#>   23.82    4.37   42.20    4.23   -4.39 -292.66
```

```r
pred <- predict(fit.nnet, newdata = test.data, type = "class")
library(caret)
confusionMatrix(factor(pred), test.labels)
```

```
#> Confusion Matrix and Statistics
#> 
#>           Reference
#> Prediction   0   1   2   3   4   5   6   7   8   9
#>          0 316   0   3   3   0   3   2   0   4   0
#>          1   0 247   0   0   4   0   0   3   4   3
#>          2  21   4 160  12   7   1   8   3  15   0
#>          3   1   0   6 121   0  11   0   2  10   0
#>          4   1   1  13   1 156   3   5   8  10  13
#>          5   7   1   0  19   4 124   3   0   2   5
#>          6   8   3   5   0   4  10 150   0   7   0
#>          7   0   2   1   1   2   0   0 117   3   1
#>          8   4   3   9   5  10   4   2   0 107   0
#>          9   1   3   1   4  13   4   0  14   4 155
#> 
#> Overall Statistics
#> 
#>                Accuracy : 0.8236
#>                  95% CI : (0.8062, 0.8401)
#>     No Information Rate : 0.1789
#>     P-Value [Acc > NIR] : < 2.2e-16
#> 
#>                   Kappa : 0.8022
#> 
#>  Mcnemar's Test P-Value : NA
#> 
#> Statistics by Class:
#> 
#>                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
#> Sensitivity            0.8802   0.9356  0.80808  0.72892  0.78000  0.77500
#> Specificity            0.9909   0.9920  0.96075  0.98370  0.96956  0.97780
#> Pos Pred Value         0.9547   0.9464  0.69264  0.80132  0.73934  0.75152
#> Neg Pred Value         0.9743   0.9903  0.97860  0.97575  0.97550  0.98046
#> Prevalence             0.1789   0.1315  0.09865  0.08271  0.09965  0.07972
#> Detection Rate         0.1574   0.1231  0.07972  0.06029  0.07773  0.06178
#> Detection Prevalence   0.1649   0.1300  0.11510  0.07524  0.10513  0.08221
#> Balanced Accuracy      0.9356   0.9638  0.88442  0.85631  0.87478  0.87640
#>                      Class: 6 Class: 7 Class: 8 Class: 9
#> Sensitivity           0.88235  0.79592  0.64458  0.87571
#> Specificity           0.97986  0.99462  0.97990  0.97596
#> Pos Pred Value        0.80214  0.92126  0.74306  0.77889
#> Neg Pred Value        0.98901  0.98404  0.96833  0.98783
#> Prevalence            0.08470  0.07324  0.08271  0.08819
#> Detection Rate        0.07474  0.05830  0.05331  0.07723
#> Detection Prevalence  0.09317  0.06328  0.07175  0.09915
#> Balanced Accuracy     0.93111  0.89527  0.81224  0.92583
```

Problem 4: Deep Learning with Keras

We again work on a handwritten digit recognition problem, but now we use the data from the keras library. This dataset consists of 60000 training data of 28×28 grayscale images and 10000 test data.

For more info see here

If you have already installed `keras` ignore the following chunk of code

```
# Install the keras R package install.packages('keras') Install the
# core Keras library + TensorFlow
library(keras)
install_keras()
# for machines with NVIDIA GPU install_keras(tensorflow = 'gpu') Did
# not switch to NVIDIA GPU this time.
```

Data Preprocessing

```
library(keras)
mnist = dataset_mnist()
x_train = mnist$train$x
y_train = mnist$train$y
x_test = mnist$test$x
y_test = mnist$test$y
# reshape
x_train = array_reshape(x_train, c(nrow(x_train), 28 * 28))
x_test = array_reshape(x_test, c(nrow(x_test), 28 * 28))
# rescale
x_train = x_train/255
x_test = x_test/255
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

a)

In order to fit a model in keras we can use the following steps. We fit a densely (or fully) conected neural network with 2 hidden layers. Fill in the missing inputs and run the model.

```

model <- keras_model_sequential() %>%
  layer_dense(units = 8, activation = 'relu', input_shape = c(28*28)) %>% # fill in the length of the i
  layer_dense(units = 8, activation = 'relu') %>%
  layer_dense(units = 10, activation = "softmax") # fill in the name of the activation function
summary(model)
```

1. Define the model

```
#> Model: "sequential"
#> ________________________________________________________________________________
#> Layer (type)                        Output Shape                    Param #
#> ================================================================================
#> dense_2 (Dense)                     (None, 8)                       6280
#> ________________________________________________________________________________
#> dense_1 (Dense)                     (None, 8)                       72
#> ________________________________________________________________________________
#> dense (Dense)                       (None, 10)                      90
#> ================================================================================
#> Total params: 6,442
#> Trainable params: 6,442
#> Non-trainable params: 0
#> ________________________________________________________________________________
```

```r
model %>% compile(optimizer = "rmsprop", loss = "categorical_crossentropy", metrics = c("accuracy"))
```

Possible choices of the activation function are 'sigmoid', 'softmax' or the identity function (which means that we don't have to define any activation argument in the last layer). What is the use of identity function?

```r
# Does not work to compile in a different code block than the model
# definition for some reason (?) model %>% compile(optimizer =
# 'rmsprop', loss = 'categorical_crossentropy', metrics =
# c('accuracy'))
```

2. Compile Possible choices of loss function are 'binary_crossentropy', 'categorical_crossentropy' and 'mse'. For metrics you can use 'mean_absolute_error' or 'accuracy'.

```r
history <- model %>% fit(x_train, y_train, epochs = 20, batch_size = 128,
    validation_split = 0.2, verbose = 0)
```

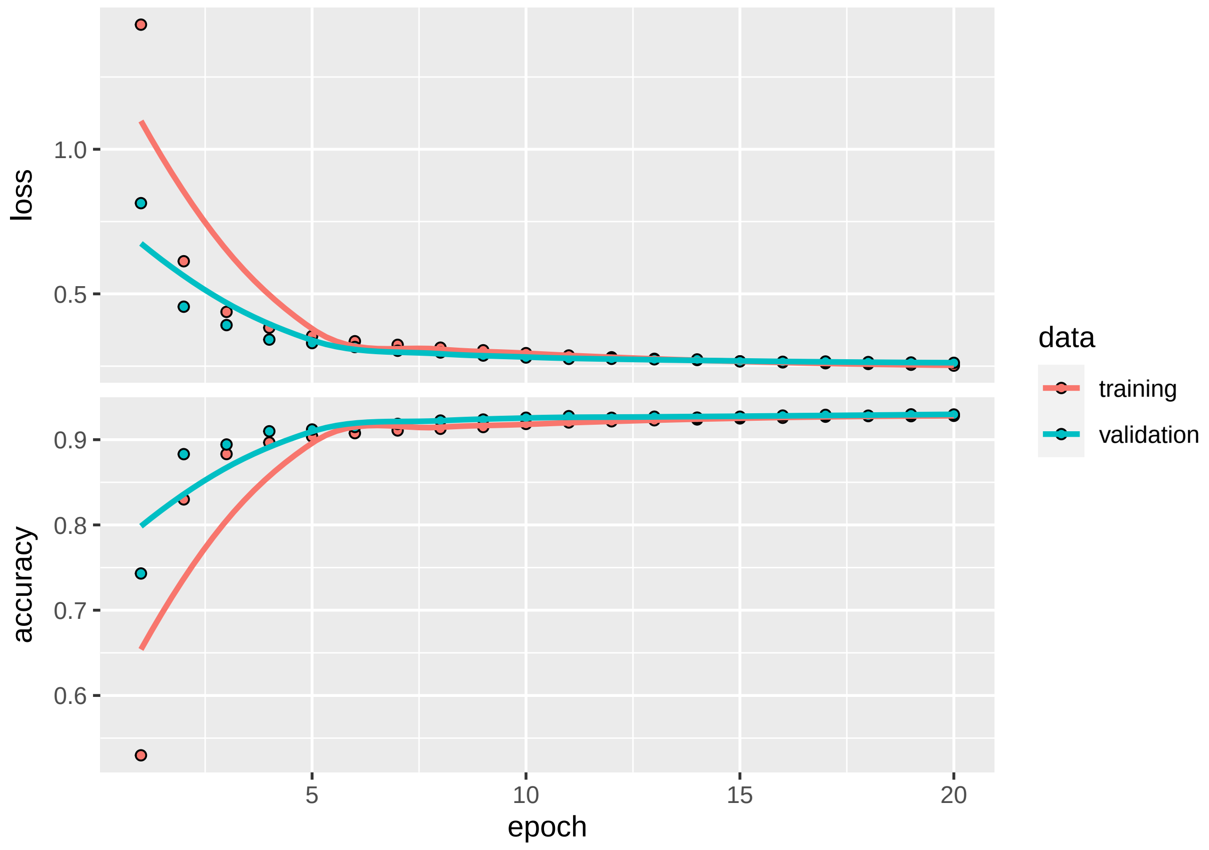
3. Train You can find more information about the different arguments in the `fit()` function here: https://keras.rstudio.com/reference/fit.html

Report the accuracy on the test data.

```r
str(history)
```

```
#> List of 2
#>  $ params :List of 3
#>   ..$ verbose: int 0
#>   ..$ epochs : int 20
#>   ..$ steps  : int 375
#>  $ metrics:List of 4
#>   ..$ loss        : num [1:20] 1.431 0.613 0.438 0.383 0.354 ...
#>   ..$ accuracy    : num [1:20] 0.53 0.83 0.883 0.897 0.904 ...
#>   ..$ val_loss    : num [1:20] 0.814 0.456 0.392 0.342 0.33 ...
#>   ..$ val_accuracy: num [1:20] 0.743 0.883 0.894 0.91 0.912 ...
#>  - attr(*, "class")= chr "keras_training_history"
```

```r
plot(history)
```



```
model %>% evaluate(x_test, y_test)
```

```
#>      loss  accuracy
#> 0.2726067 0.9254000
```

The accuracy on the test data is 92.7%, as reported above.

What is the number of parameters?

The number of parameters are 6442.

b)

Now fit a model with 2 hidden layers with 128 units each. What do you see?

```
model2 <- keras_model_sequential() %>% layer_dense(units = 128, activation = "relu",
    input_shape = c(28 * 28)) %>% layer_dense(units = 128, activation = "relu") %>%
    layer_dense(units = 10, activation = "softmax")
summary(model2)
```

```
#> Model: "sequential_1"
#> ________________________________________________________________________________
#> Layer (type)                         Output Shape                     Param #
#> ================================================================================
#> dense_5 (Dense)                      (None, 128)                      100480
#> ________________________________________________________________________________
#> dense_4 (Dense)                      (None, 128)                      16512
#> ________________________________________________________________________________
#> dense_3 (Dense)                      (None, 10)                       1290
#> ================================================================================
#> Total params: 118,282
#> Trainable params: 118,282
#> Non-trainable params: 0
#> ________________________________________________________________________________
```

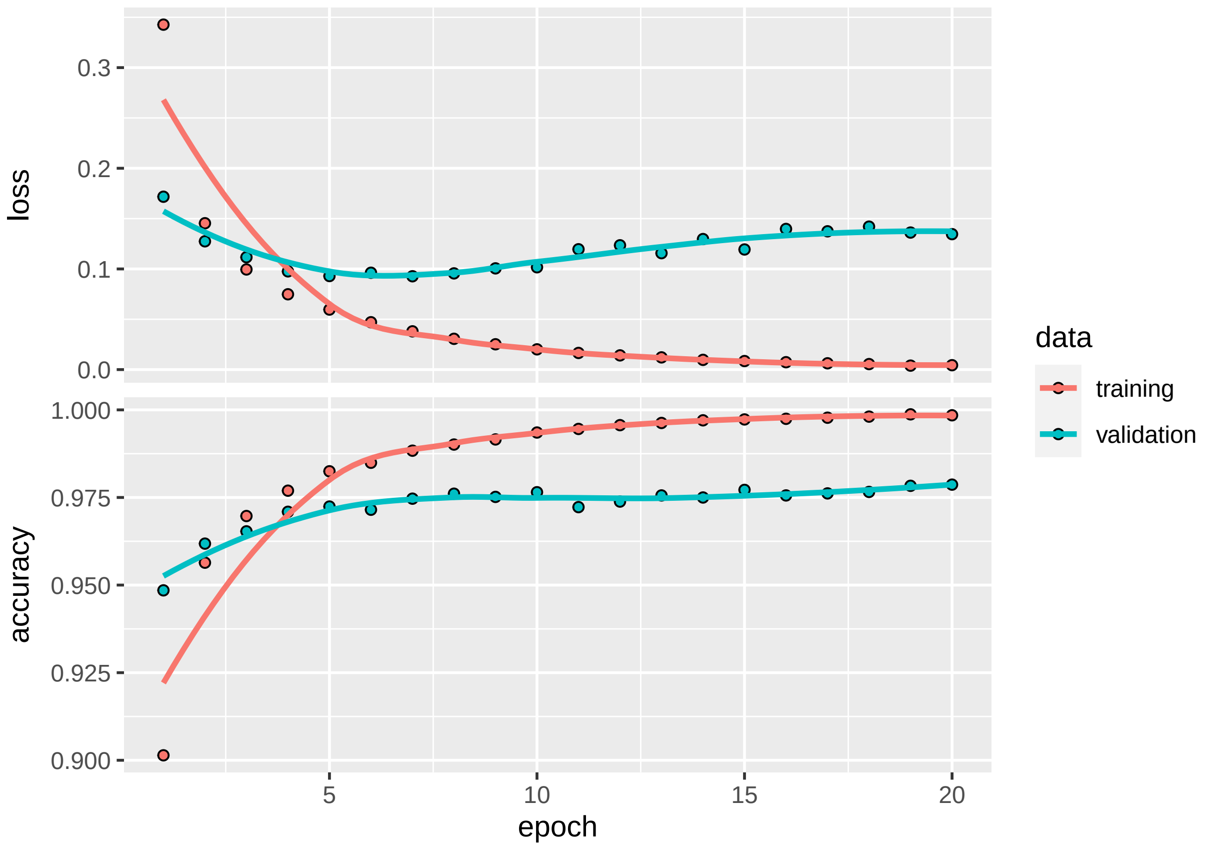
```
# Compile.
model2 %>% compile(optimizer = "rmsprop", loss = "categorical_crossentropy",
    metrics = c("accuracy"))

# Train.
history2 <- model2 %>% fit(x_train, y_train, epochs = 20, batch_size = 128,
    validation_split = 0.2, verbose = 0)

# Evaluate.
str(history2)
```

```
#> List of 2
#>  $ params :List of 3
#>   ..$ verbose: int 0
#>   ..$ epochs : int 20
#>   ..$ steps  : int 375
#>  $ metrics:List of 4
#>   ..$ loss        : num [1:20] 0.3427 0.1454 0.0995 0.0748 0.0596 ...
#>   ..$ accuracy    : num [1:20] 0.901 0.956 0.97 0.977 0.982 ...
#>   ..$ val_loss    : num [1:20] 0.1718 0.1274 0.1116 0.0976 0.0929 ...
#>   ..$ val_accuracy: num [1:20] 0.948 0.962 0.965 0.971 0.972 ...
#>  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history2)
```



```r
model2 %>% evaluate(x_test, y_test)
```

```
#>      loss  accuracy
#> 0.1203172 0.9783000
```

We see that the validation loss reaches its minimum after 5-10 epochs and increases again after this. Thus, the larger network overfits after the first 5-10 epochs, which is a problem.

c)

In order to avoid the problem we have seen in b), we can use weight regularization (L1 and L2 norms) or dropout. Apply these methods to the network from b).

```r
# Regularization with weight decay with L2 norm.  Build the model
model3 <- keras_model_sequential() %>% layer_dense(units = 128, activation = "relu",
    input_shape = c(28 * 28), kernel_regularizer = regularizer_l2(l = 0.001)) %>%
    layer_dense(units = 128, activation = "relu", kernel_regularizer = regularizer_l2(l = 0.001)) %>%
    layer_dense(units = 10, activation = "softmax")
summary(model3)
```

```
#> Model: "sequential_2"
#> ________________________________________________________________________________
#> Layer (type)                        Output Shape                    Param #
#> ================================================================================
#> dense_8 (Dense)                     (None, 128)                     100480
#> ________________________________________________________________________________
#> dense_7 (Dense)                     (None, 128)                     16512
#> ________________________________________________________________________________
#> dense_6 (Dense)                     (None, 10)                      1290
#> ================================================================================
#> Total params: 118,282
#> Trainable params: 118,282
#> Non-trainable params: 0
#> ________________________________________________________________________________
```

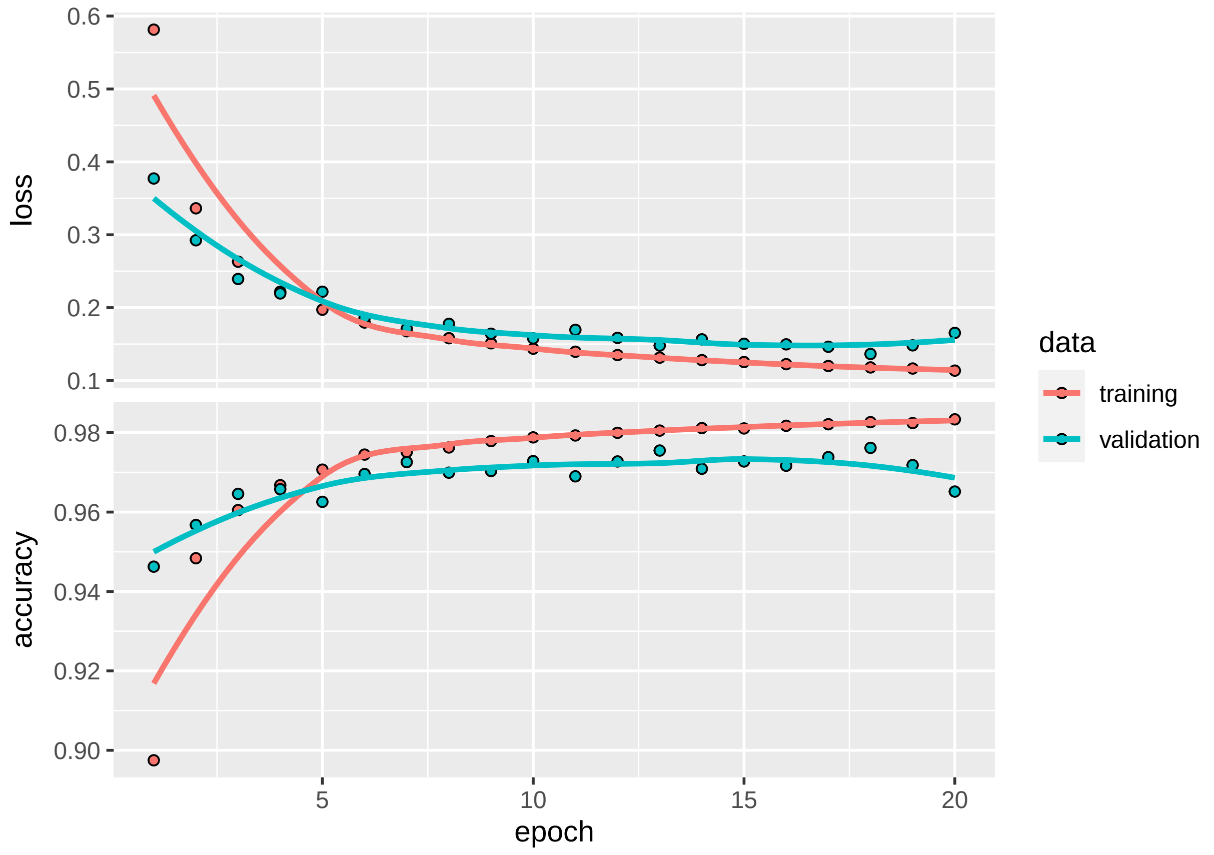
```
# Compile.
model3 %>% compile(optimizer = "rmsprop", loss = "categorical_crossentropy",
    metrics = c("accuracy"))

# Train.
history3 <- model3 %>% fit(x_train, y_train, epochs = 20, batch_size = 128,
    validation_split = 0.2, verbose = 0)

# Evaluate.
str(history3)
```

```
#> List of 2
#>  $ params :List of 3
#>   ..$ verbose: int 0
#>   ..$ epochs : int 20
#>   ..$ steps  : int 375
#>  $ metrics:List of 4
#>   ..$ loss        : num [1:20] 0.581 0.336 0.263 0.222 0.197 ...
#>   ..$ accuracy    : num [1:20] 0.897 0.948 0.961 0.967 0.971 ...
#>   ..$ val_loss    : num [1:20] 0.377 0.292 0.239 0.219 0.222 ...
#>   ..$ val_accuracy: num [1:20] 0.946 0.957 0.965 0.966 0.963 ...
#>  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history3)
```



```r
model3 %>% evaluate(x_test, y_test)
```

```
#>      loss  accuracy
#> 0.1569326 0.9690000
```

```r
# Regularization with dropout.  Build the model.
model4 <- keras_model_sequential() %>% layer_dense(units = 128, activation = "relu",
    input_shape = c(28 * 28)) %>% layer_dropout(rate = 0.6) %>% layer_dense(units = 128,
    activation = "relu") %>% layer_dropout(rate = 0.6) %>% layer_dense(units = 10,
    activation = "softmax")
summary(model4)
```

```
#> Model: "sequential_3"
#> ________________________________________________________________________________
#> Layer (type)                        Output Shape                    Param #
#> ================================================================================
#> dense_11 (Dense)                    (None, 128)                     100480
#> ________________________________________________________________________________
#> dropout_1 (Dropout)                 (None, 128)                     0
#> ________________________________________________________________________________
#> dense_10 (Dense)                    (None, 128)                     16512
#> ________________________________________________________________________________
#> dropout (Dropout)                   (None, 128)                     0
#> ________________________________________________________________________________
#> dense_9 (Dense)                     (None, 10)                      1290
#> ================================================================================
#> Total params: 118,282
#> Trainable params: 118,282
#> Non-trainable params: 0
#> ________________________________________________________________________________
```

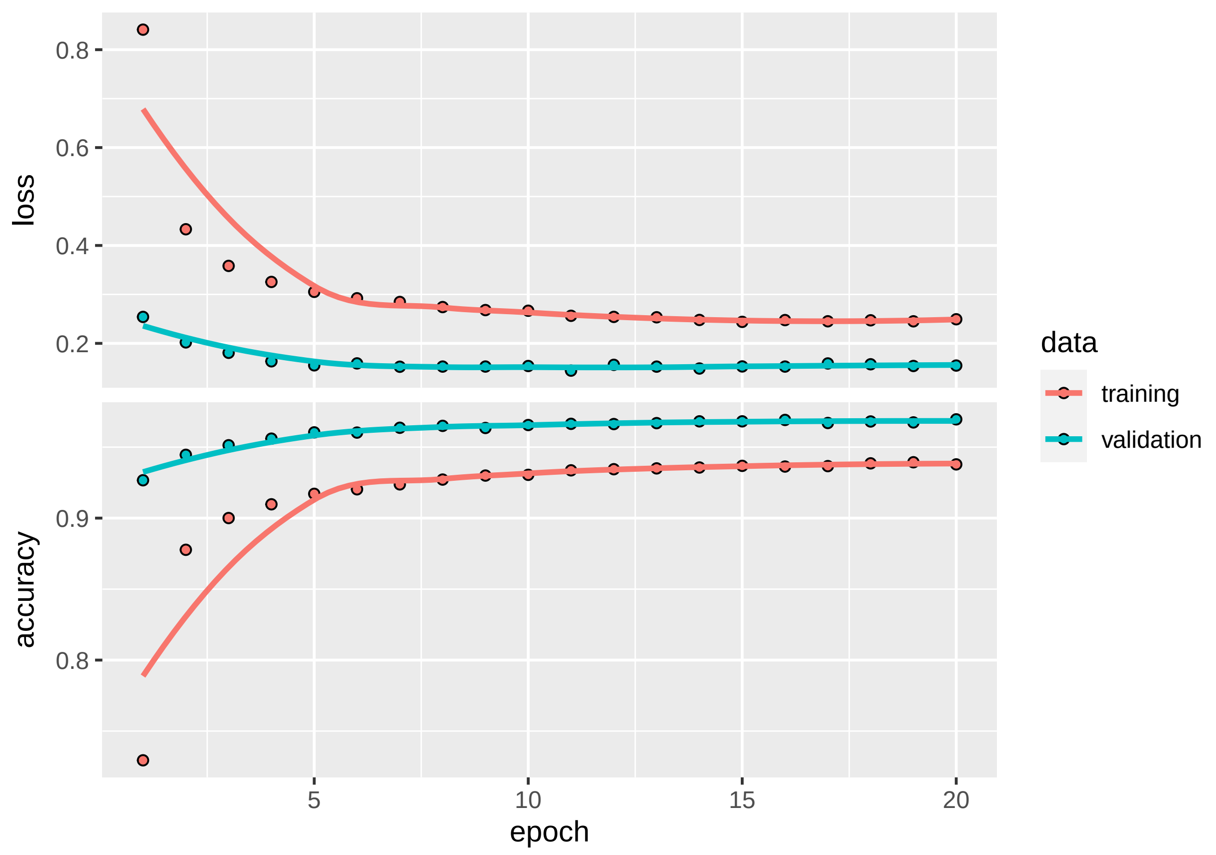
```r
# Compile.
model4 %>% compile(optimizer = "rmsprop", loss = "categorical_crossentropy",
    metrics = c("accuracy"))

# Train.
history4 <- model4 %>% fit(x_train, y_train, epochs = 20, batch_size = 128,
    validation_split = 0.2, verbose = 0)

# Evaluate.
str(history4)
```

```
#> List of 2
#>  $ params :List of 3
#>   ..$ verbose: int 0
#>   ..$ epochs : int 20
#>   ..$ steps  : int 375
#>  $ metrics:List of 4
#>   ..$ loss        : num [1:20] 0.841 0.433 0.358 0.326 0.305 ...
#>   ..$ accuracy    : num [1:20] 0.729 0.878 0.9 0.91 0.917 ...
#>   ..$ val_loss    : num [1:20] 0.254 0.202 0.181 0.163 0.155 ...
#>   ..$ val_accuracy: num [1:20] 0.927 0.945 0.951 0.956 0.96 ...
#>  - attr(*, "class")= chr "keras_training_history"
```

```r
plot(history4)
```



```
model4 %>% evaluate(x_test, y_test)
```

```
#>      loss  accuracy
#> 0.1505406 0.9661000
```

Now the validation curves look better than before.

For more details on regularization, see here.