Introduction to machine learning and neural networks

Toby Dylan Hocking
toby.hocking@nau.edu
toby.hocking@r-project.org

July 7, 2020

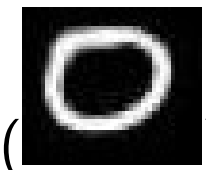
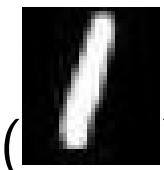
Introduction and applications

Demonstration of overfitting in regression / first steps with R

Classifying images of digits

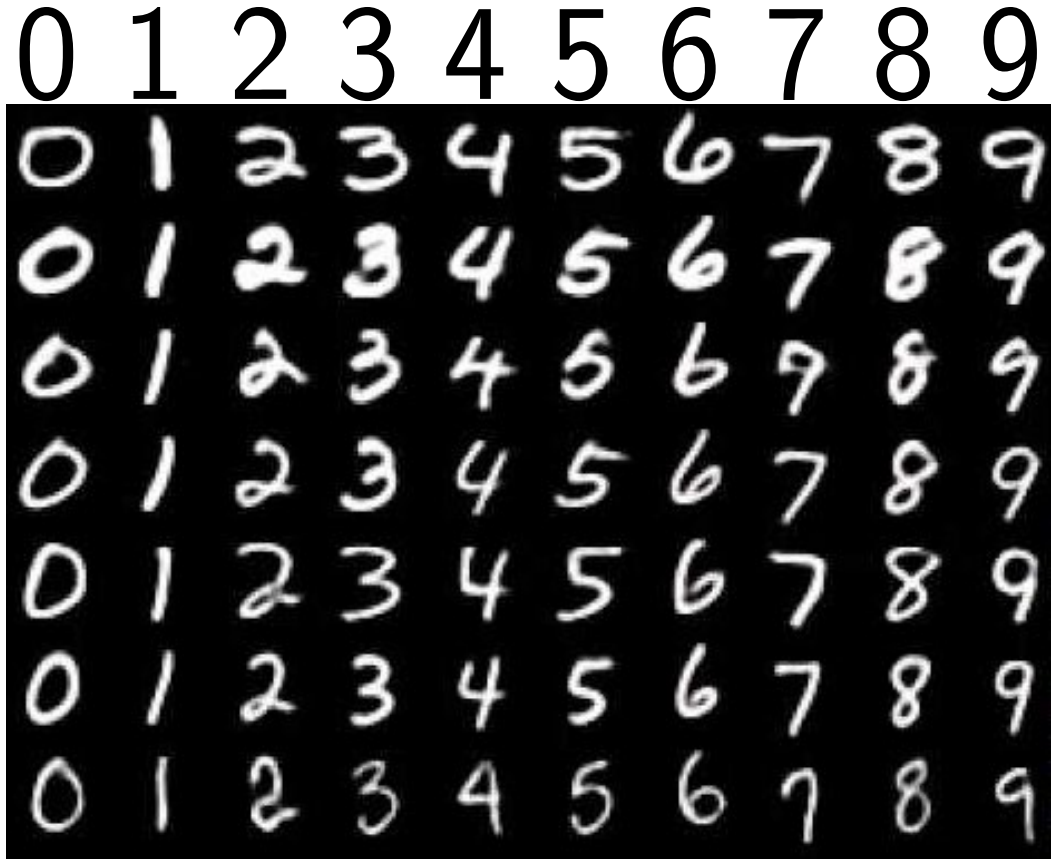
Machine learning intro: image classification example

ML is all about learning predictive functions $f(x) \approx y$, where

- Inputs/features x can be easily computed using traditional algorithms, e.g. matrix of pixel intensities in an image.
- Outputs/labels y are what we want to predict, easy to get by asking a human, but hard to compute using traditional algorithms, e.g. image class.
- Input x = image of digit, output $y \in \{0, 1, \ldots, 9\}$,
  – this is a classification problem with 10 classes.
  $f($  $) = 0$, $f($  $) = 1$
- Traditional/unsupervised algorithm: I give you a pixel intensity matrix $x \in \mathbb{R}^{16 \times 16}$, you code a function f that returns one of the 10 possible digits. Q: how to do that?

Supervised machine learning algorithms

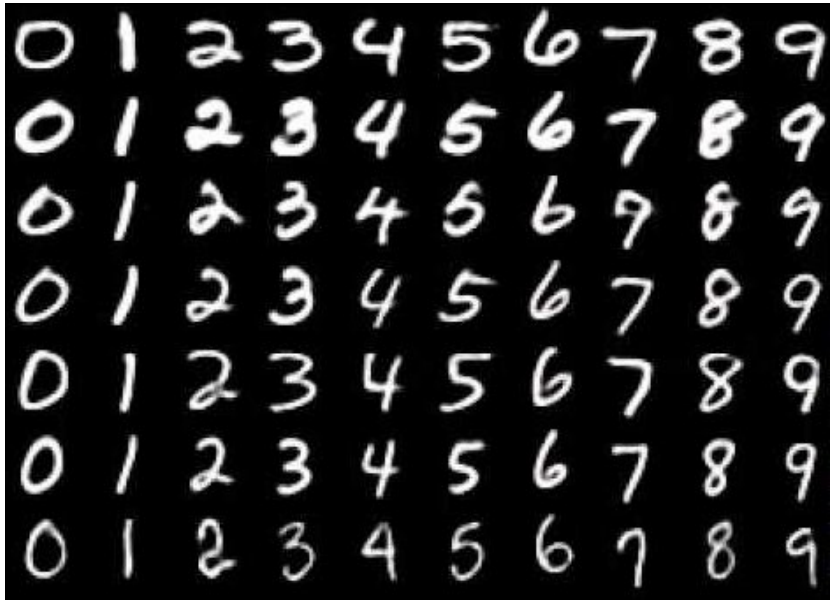
I give you a training data set with paired inputs/outputs, e.g.

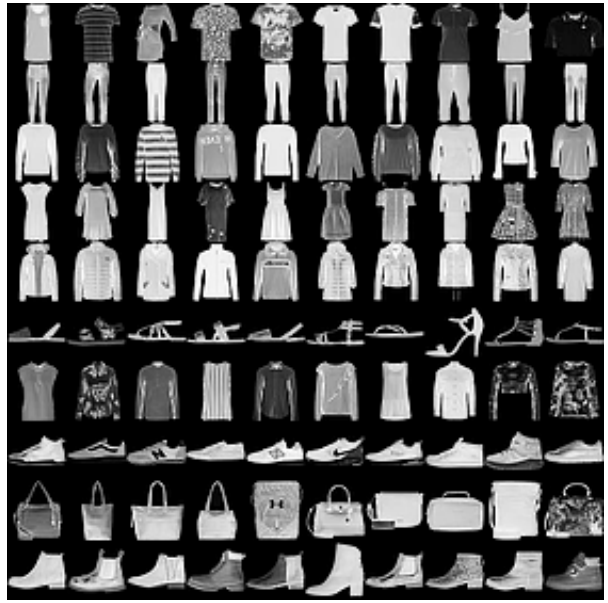


Your job is to code an algorithm that learns the function f from the training data. (you don't code f)

Source: github.com/cazala/mnist

Advantages of supervised machine learning



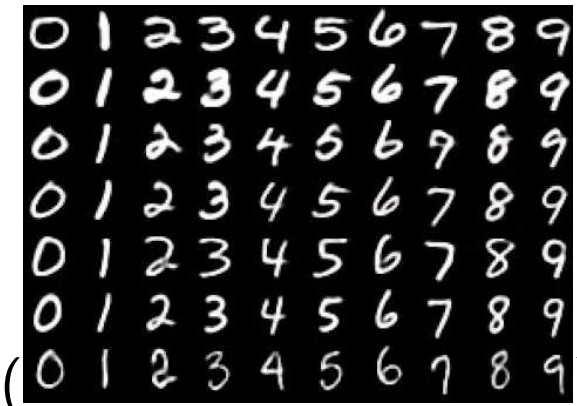
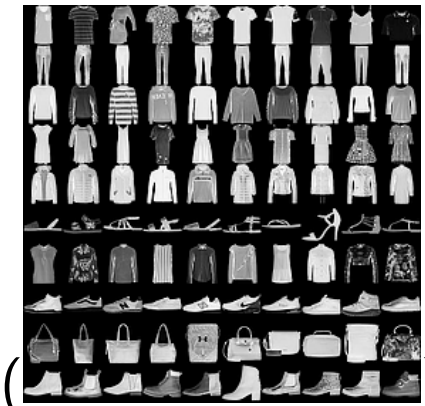


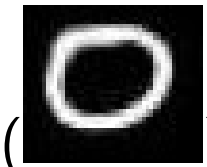
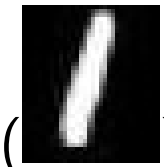
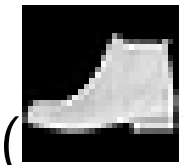
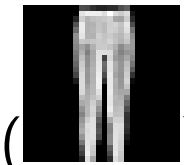
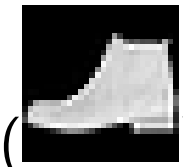
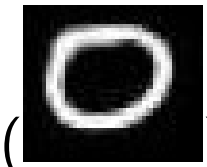
- Input $x \in \mathbb{R}^{16\times16}$, output $y \in \{0, 1, \ldots, 9\}$ types the same!
- Can use same learning algorithm regardless of pattern.
- Pattern encoded in the labels (not the algorithm).
- Useful if there are many un-labeled data, but few labeled data (or getting labels is long/costly).
- State-of-the-art accuracy (if there is enough training data).

Sources: github.com/cazala/mnist, github.com/zalandoresearch/fashion-mnist

Learning two different functions

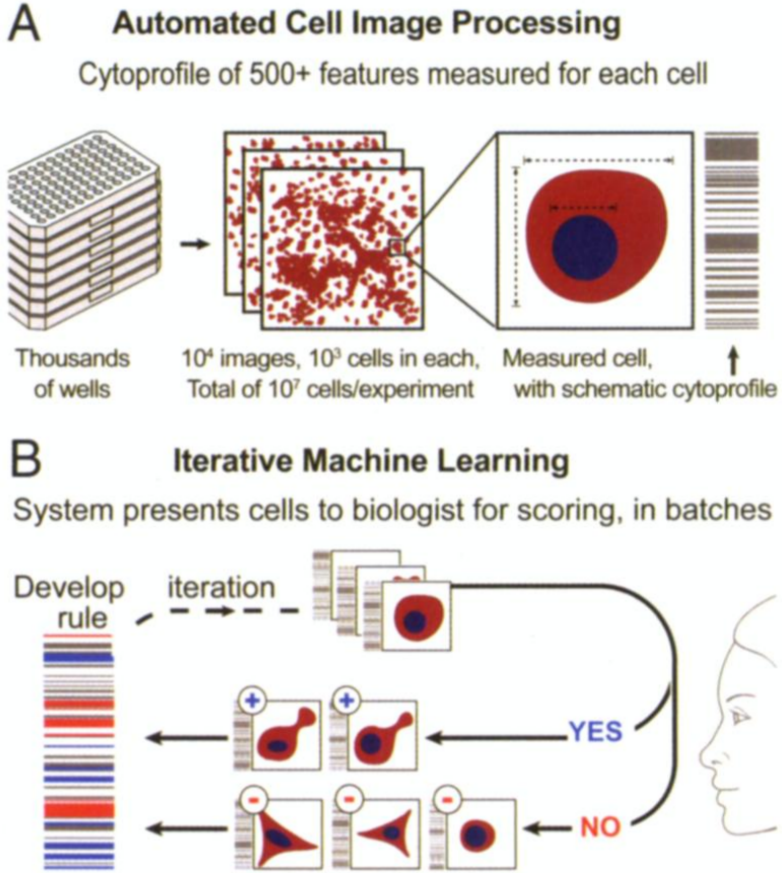
Say LEARN is a learning algorithm you have coded.

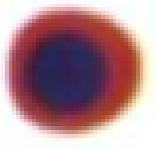
LEARN() $\rightarrow f$, LEARN() $\rightarrow g$

- Then we would expect $f($  $) = 0$, $f($  $) = 1$
- $g($  $) =$ shoe$/0$, $g($  $) =$ pants$/1$
- Q: what happens if you do $f($  $)$, or $g($  $)$?

Machine learning for cell image classification (CellProfiler)

Jones *et al.* PNAS 2009. Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning.



- Input $x =$ image of cell,
- Output $y \in \{\text{yes}, \text{no}\}$ (binary classification),
- $f($  $) =$ yes,
- $f($  $) =$ no.

Machine learning for image segmentation (LabelMe)

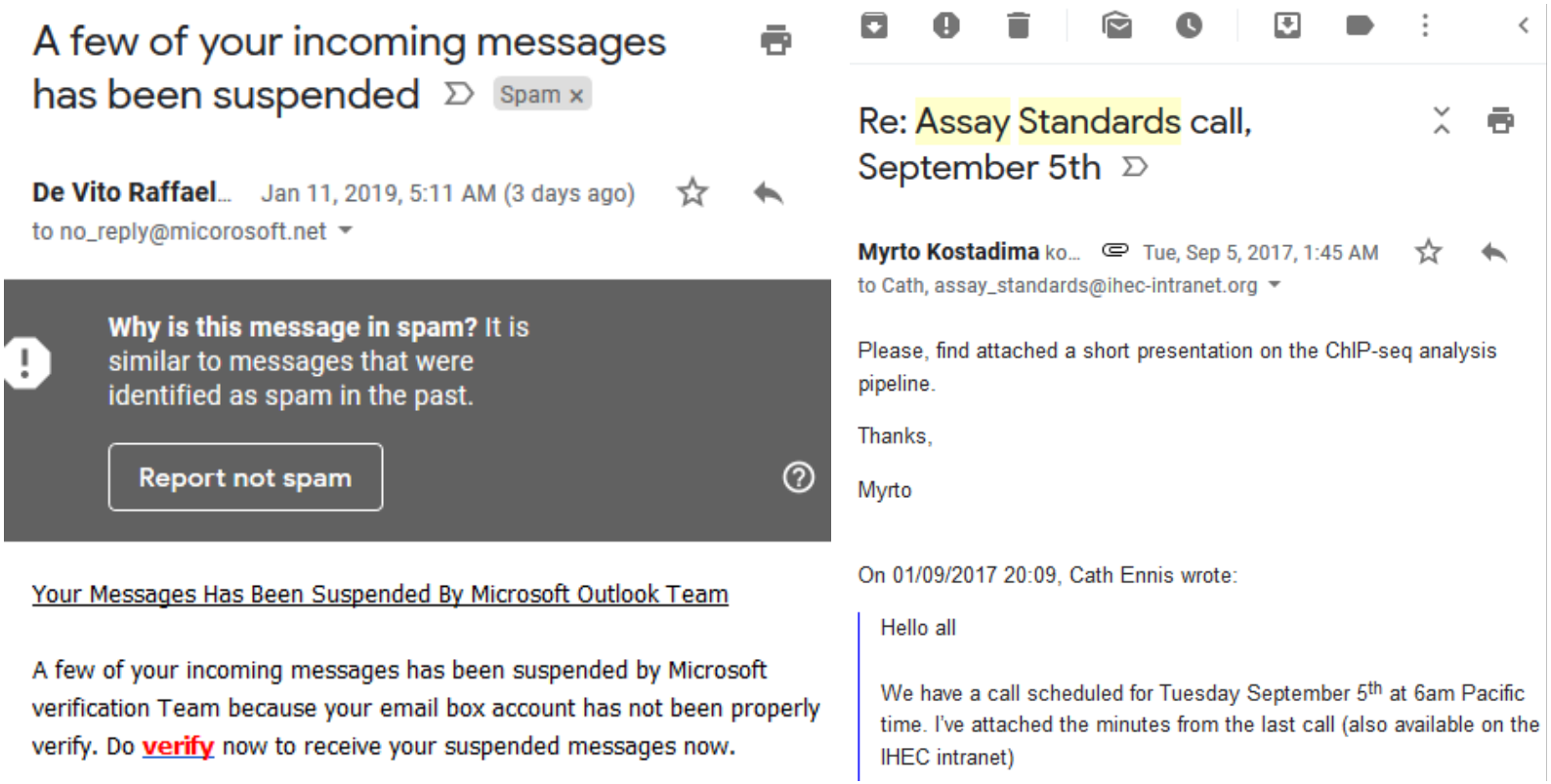
Figure 1. A screenshot of the labeling tool in use. The user is shown an image along with possibly one or more existing annotations, which are drawn on the image. The user has the option of annotating a new object by clicking along the boundary of the desired object and

Machine learning for image segmentation

Russell *et al.* 2007.

Q: What are the types/dimensions of x, y, f in this example?

Machine learning for spam filtering (Gmail)



Want: $f(\text{email message}) \in \{0, 1\}$ – binary classification, spam=1 or not=0.

Machine learning for translation (google books)

LE COMTE

DE

MONTE-CRISTO

PAR

ALEXANDRE DUMAS

ABRIDGED AND ANNOTATED

BY

EDGAR EWING BRANDON, A.M.

Professor of French in Miami University

NEW YORK

HENRY HOLT AND COMPANY

1900

THE

COUNT OF MONTE-CRISTO.

BY

ALEXANDRE DUMAS.

WITH

Twenty Illustrations,

DRAWN ON WOOD BY M. VALENTIN,

AND EXECUTED BY THE MOST EMINENT ENGLISH ENGRAVERS,
UNDER THE SUPERINTENDENCE OF MR. CHARLES HEATH.

IN TWO VOLUMES.

VOL. II.

LONDON:

CHAPMAN AND HALL, 186 STRAND.

1846.

Machine learning for translation (google translate)

— Fernand! lui cria-t-il, de mes noms, je n'aurais besoin de t'en dire qu'un seul pour te foudroyer; mais ce nom, tu le devines, n'est-ce pas ? ou plutôt tu te le rappelles ? car, malgré tous mes chagrins, toutes mes tortures, je te montre aujourd'hui un visage que le bonheur de la vengeance rajeunit, un visage que tu dois avoir vu bien souvent dans tes rêves depuis ton mariage . . . avec Mercédès, ma fiancée!

Le général, la tête renversée en arrière, les mains étendues, le regard fixe, dévora en silence ce terrible spectacle; puis, allant chercher la muraille comme point d'appui, il s'y glissa lentement jusqu'à la porte par laquelle il sortit à reculons, en laissant échapper ce seul cri lugubre, lamentable, déchirant:

— Edmond Dantès!

Want: $f($ $)=$

" Fernand," cried he, " of my hundred names I need only tell you one to overwhelm you ! But you guess it now; do you not ? — or, rather, you remember it ? For, notwithstanding all my sorrows and my tortures, I shew you to-day a face which the happiness of revenge makes young again — a face you must often have seen in your dreams since your marriage with Mercédès, my betrothed ! "

The general, with his head thrown back, hands extended, gaze fixed, looked silently at this dreadful apparition ; then seeking the wall to support him, he glided alon close to it until he reached the door, through which he went out backwards, uttering this single mournful, lamentable, distressing cry,—

" Edmond Dantès ! "

Machine learning for medical diagnosis

Original

f( ) =

Thanks to artificial intelligence and machine learning, diagnostic tools can be trained to read tissue samples and radiologic scans. Google researchers fed more than a quarter-million patients' retinal scans into algorithms that recognize patterns—and the technology "learned" to spot which patterns predict a patient has high blood pressure or is at increased risk for heart attack or stroke. In some comparisons, digital tools produced more accurate analyses than did human pathologists, dermatolo-

Introduction and applications

Demonstration of overfitting in regression / first steps with R

Classifying images of digits

Machine learning algorithms in practice

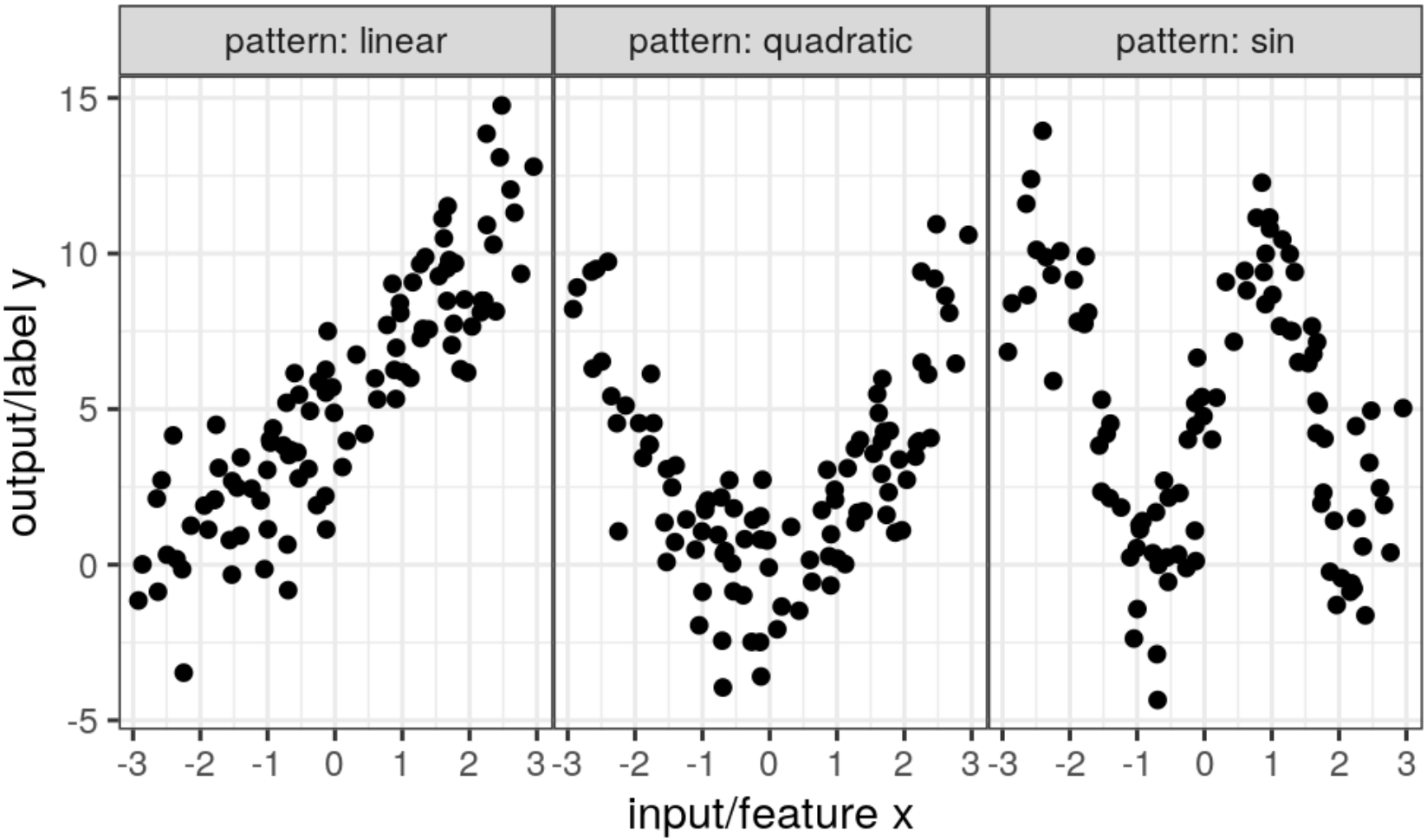
To learn a good prediction function for any of these problems (and any other problems you encounter in your work) you should

- First gather a data set that you will use to train/test the machine learning algorithms, typically a CSV file with rows for observations and columns for features.
- Use K-fold Cross-Validation to repeatedly divide the observations into train/test sets. Use the train set to learn machine learning model parameters, and use the test set to estimate prediction error.
- Try a variety of different learning algorithms (e.g. linear models, random forests, boosting, neural networks, support vector machines, ...), because you don't know which one will result in most accurate predictions for any given data set.
- Each learning algorithm has different hyper-parameters which control complexity/regularization/overfitting, and typically must be chosen by the user (you) by dividing the train data into subtrain/validation sets. Use the hyper-parameter which is best with respect to the held-out validation set.

Goal of this section: demonstrate overfitting

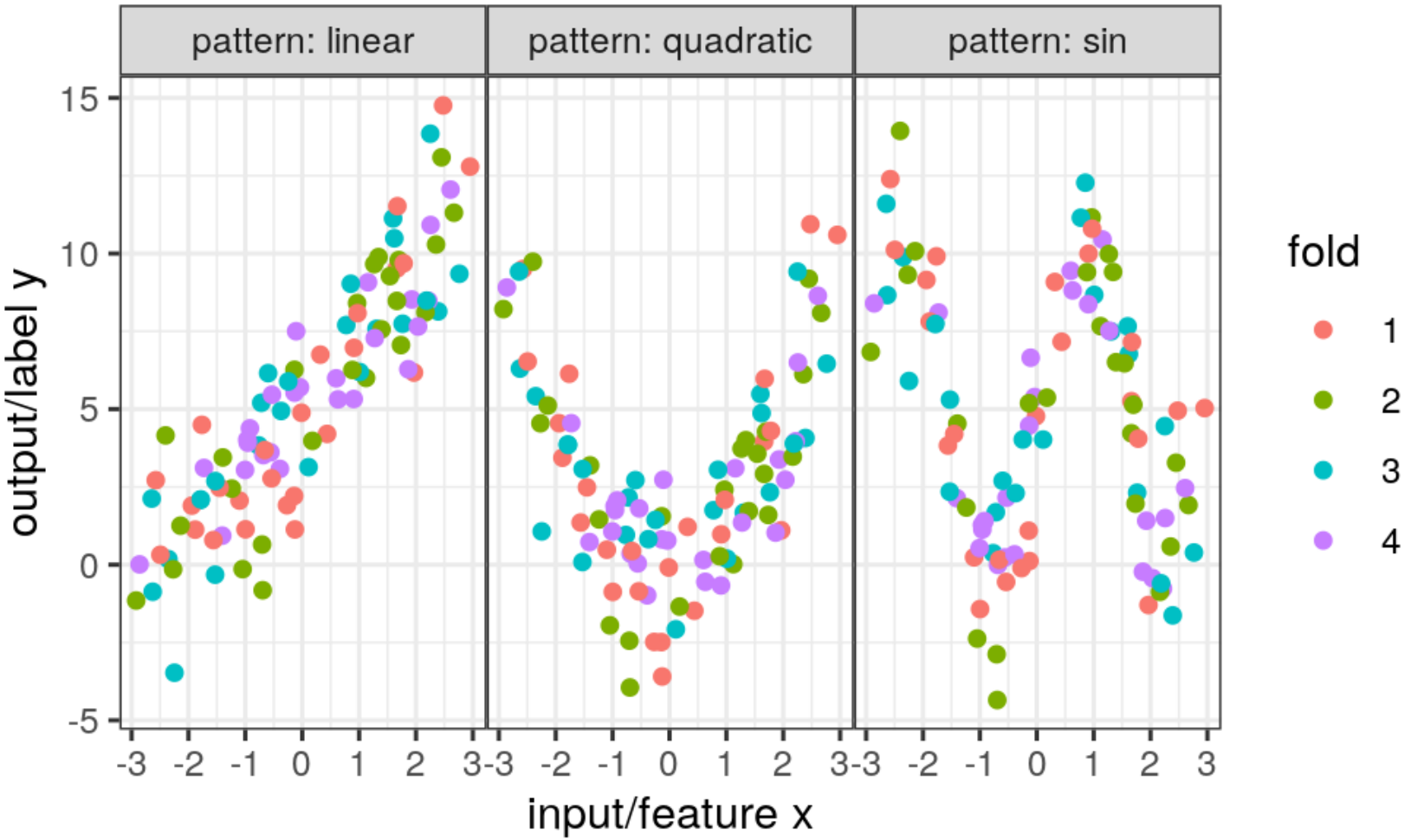
- The goal of supervised machine learning is to get accurate predictions on new/unseen/held-out test data.
- Any machine learning algorithm is prone to overfit, which means providing better predictions on the train/subtrain set than on a held-out validation/test set. (BAD)
- To learn a model which does NOT overfit (GOOD), you need to first divide your train set into subtrain/validation sets.
- Code for figures in this section: `https://github.com/tdhock/2020-yiqi-summer-school/blob/master/figure-overfitting.R`

Three different data sets/patterns



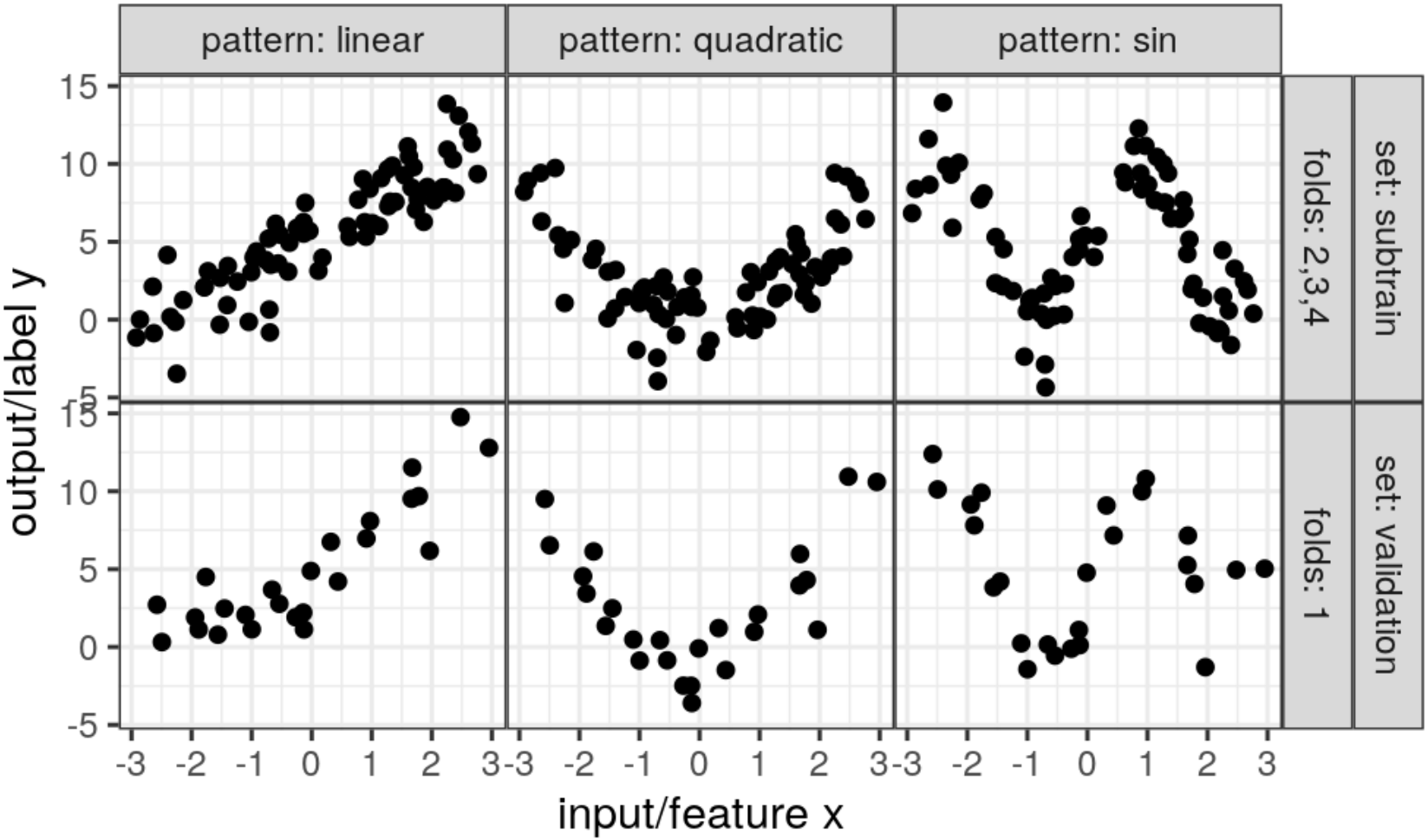
- We illustrate this using a single input/feature $x \in \mathbb{R}$.
- We use a regression problem with outputs $y \in \mathbb{R}$.
- Goal is to learn a function $f(x) \in \mathbb{R}$.

Illustration of 4-fold cross-validation



Randomly assign each observation a fold ID from 1 to 4.

Illustration of subtrain/validation split



- For validation fold 1, all observations with that fold ID are considered the validation set.
- All other observations are considered the subtrain set.

How to do this in practice? Use R with an IDE

- Machine learning model training/evaluation is easy to code in programming languages which support vectors/matrices.
- I recommend using R because it has so many useful packages (each one for a different machine learning model/algorithm).
- Download R at `https://cloud.r-project.org/`
- If you already use Emacs (classic text editor), I recommend using R inside Emacs Speaks Statistics, which is a great Interactive Development Environment (IDE) for R `http://ess.r-project.org/` Also see my screencasts about ESS, `https://www.youtube.com/playlist?list=PLwc48KSH3D1Onsed66FPLywMSIQmAhUYJ`
- Otherwise, I recommend using RStudio, which is another popular IDE for R `https://rstudio.com/products/rstudio/download`

Creating/reading CSV data tables

- One row for each observation.
- One column for the output/label (in regression the label is a real number, in classification the label is a class/category).
- The other columns should be inputs/features that will be used to predict the corresponding output/label.

Examples:

- TODO

K-fold cross-validation for evaluating prediction accuracy

Randomly create a vector of fold ID numbers from 1 to K, one fold ID for each observation:

```
set.seed(1) #for reproducibility.
n.folds <- 4
unique.folds <- 1:n.folds
fold.vec <- sample(rep(unique.folds, l=N.observations))
```

The results:

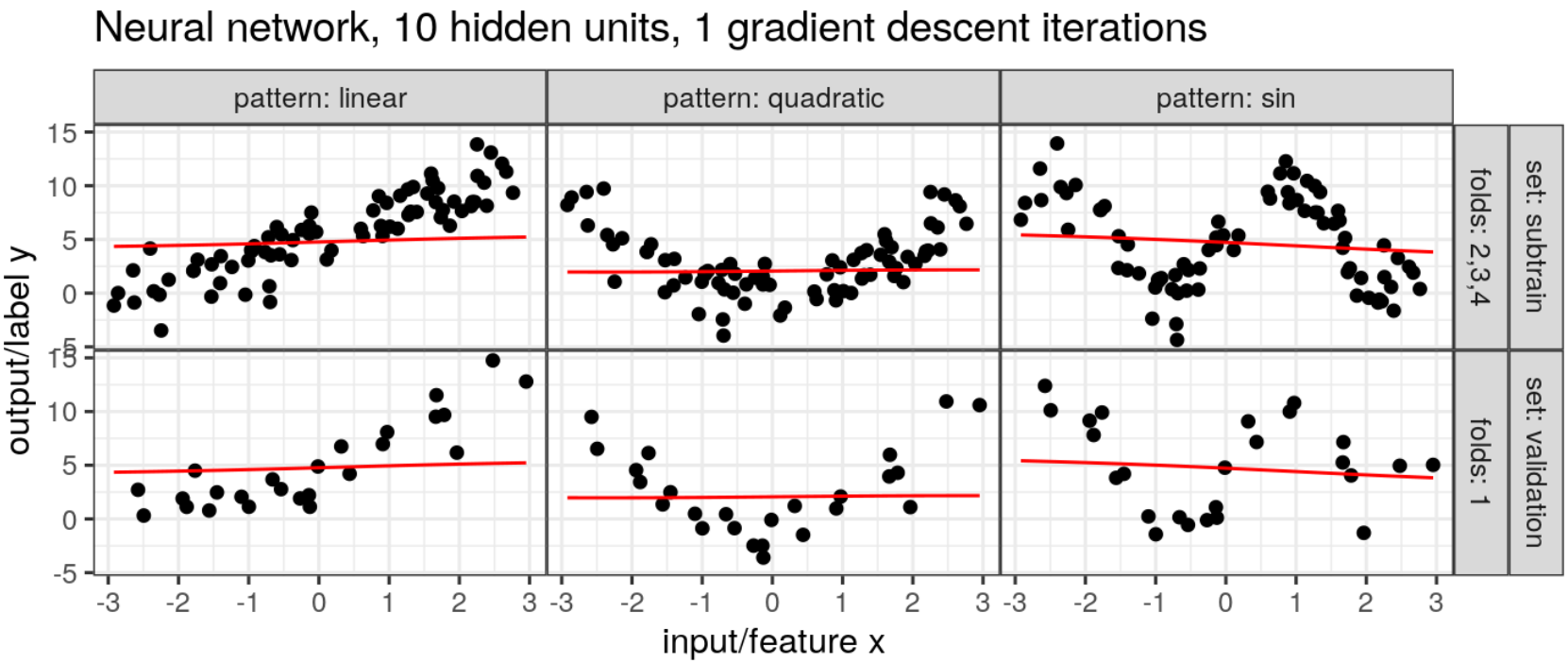
```
> unique.folds
[1] 1 2 3 4
> str(fold.vec)
 int [1:100] 4 3 1 2 3 3 2 2 3 3 ...
```

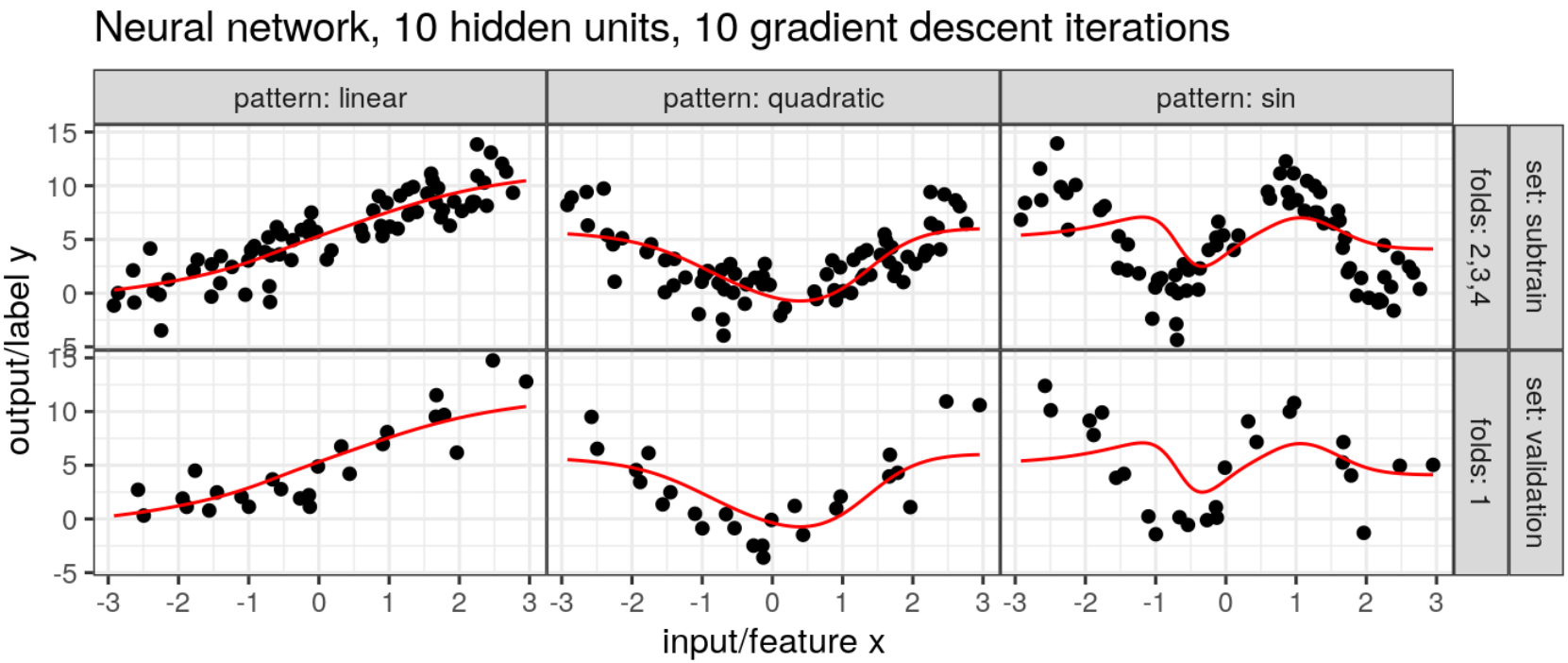
Hyper-parameter tuning

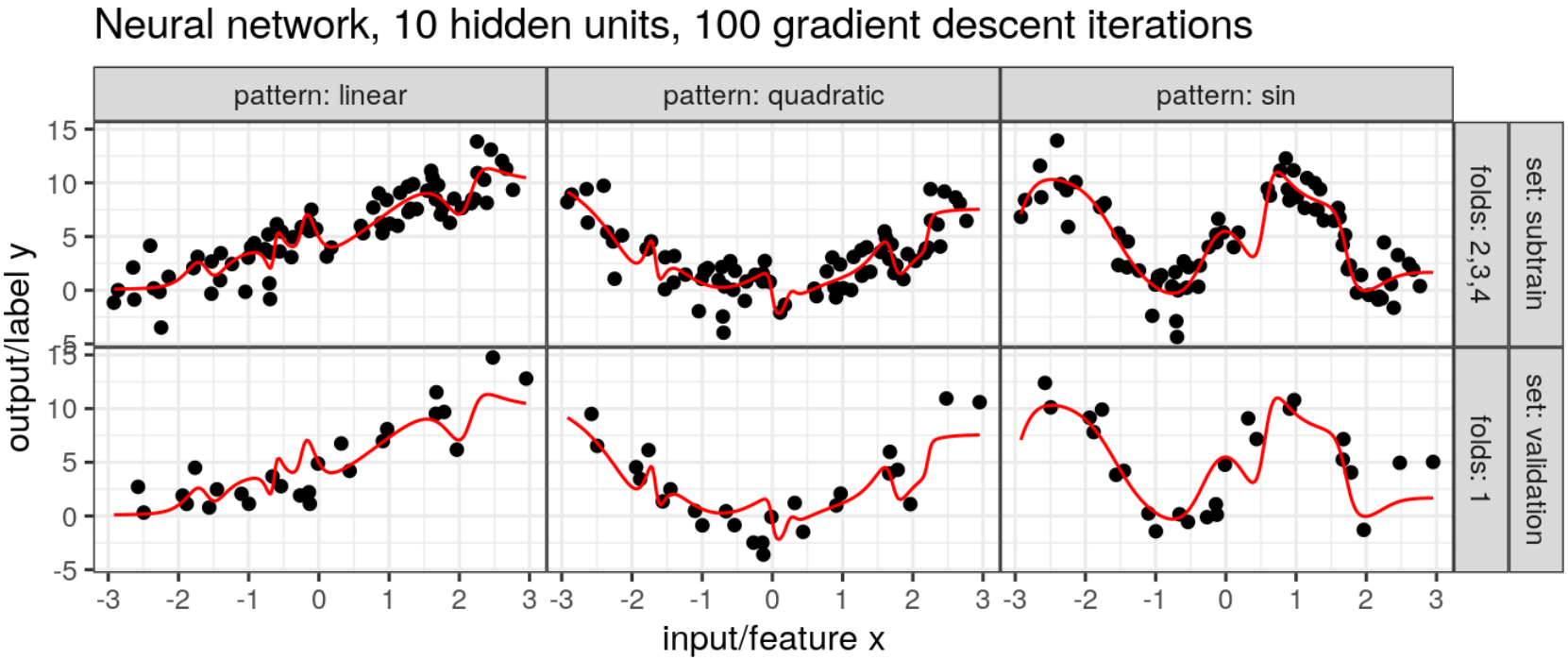
Fix a hyper-parameter value and use the subtrain data to fit use the subtrain data to learn a model for each of hyper-parameters, then use the hyper-parameters that result in maximal prediction accuracy with respect to the .

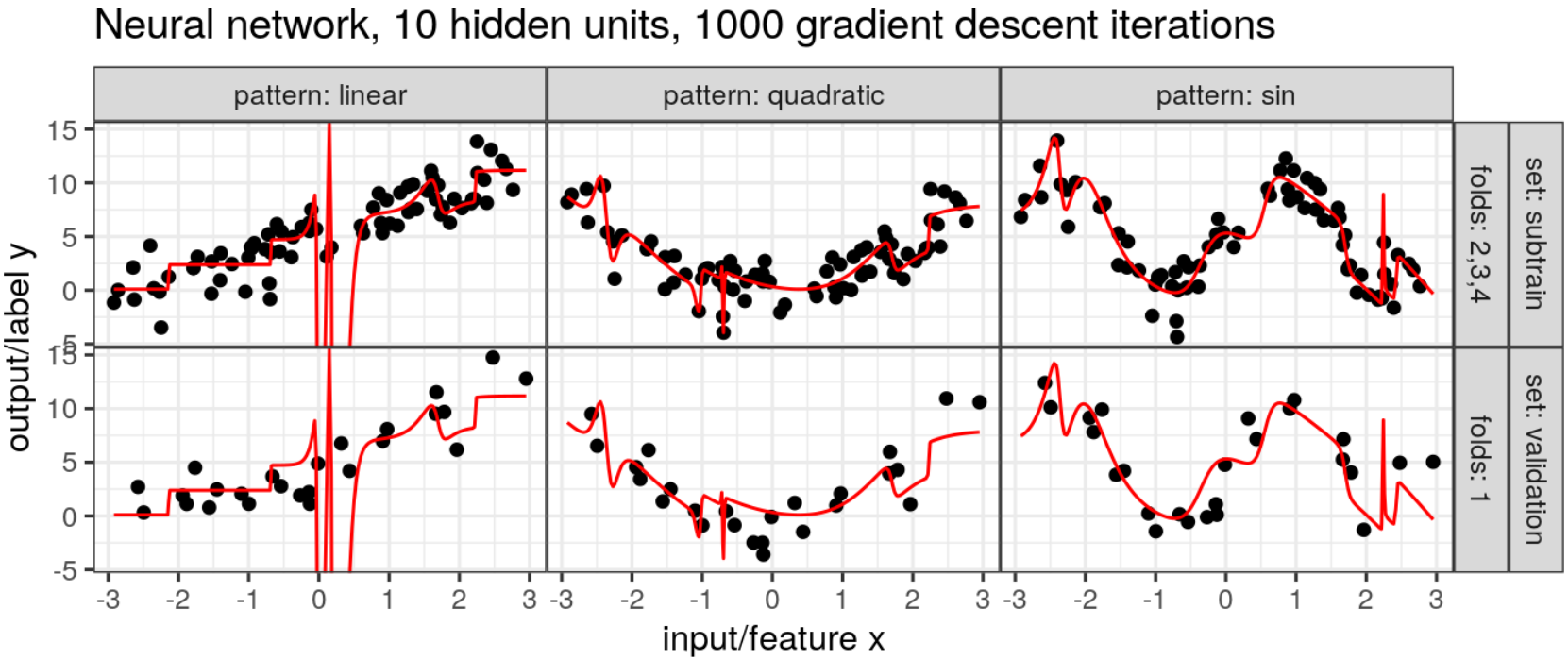
Neural network with one hidden layer, 10 hidden units

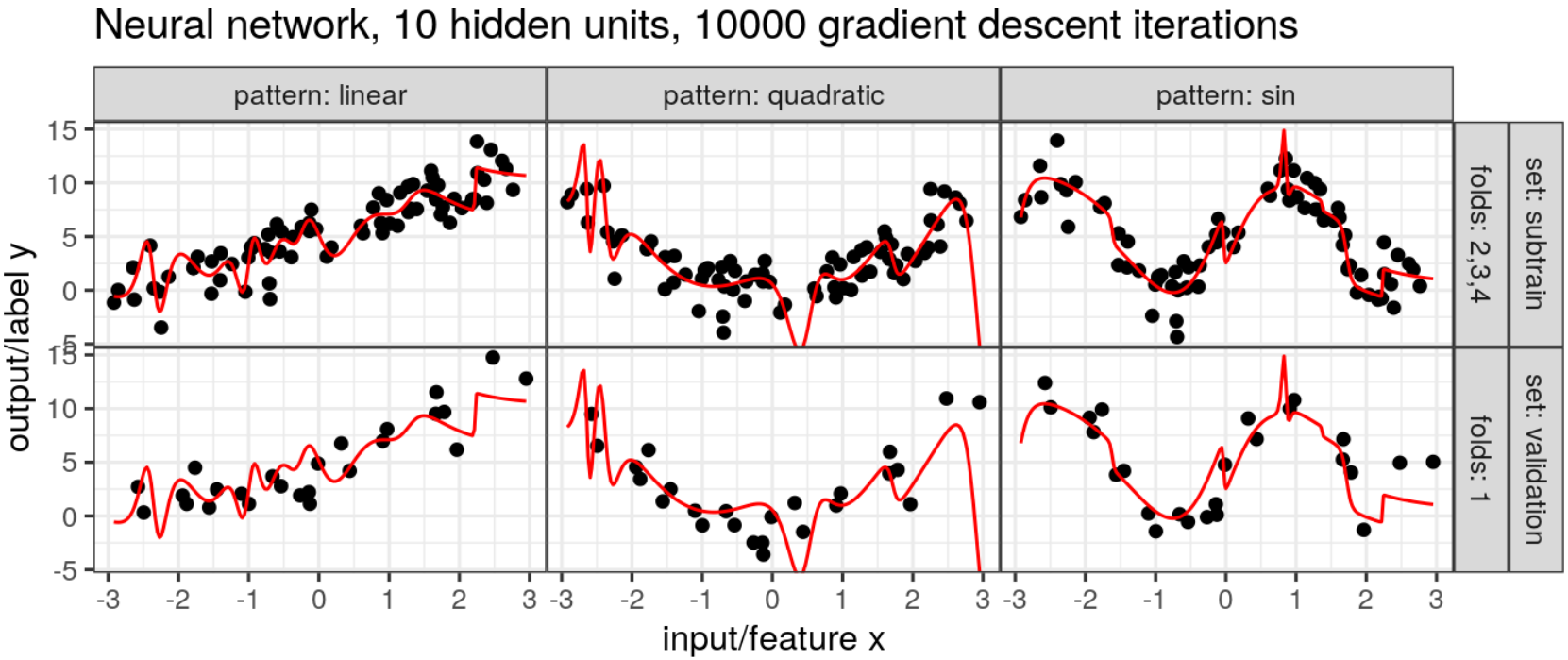
```
for(maxit in c(1, 10, 100, 1000, 10000)){
  fit <- nnet::nnet(
    y ~ x,
    set.data$subtrain,
    size=10,      #hidden units
    skip=FALSE,   #no units which skip the hidden layer
    linout=TRUE, #for regression
    maxit=maxit) #max number of iterations
```

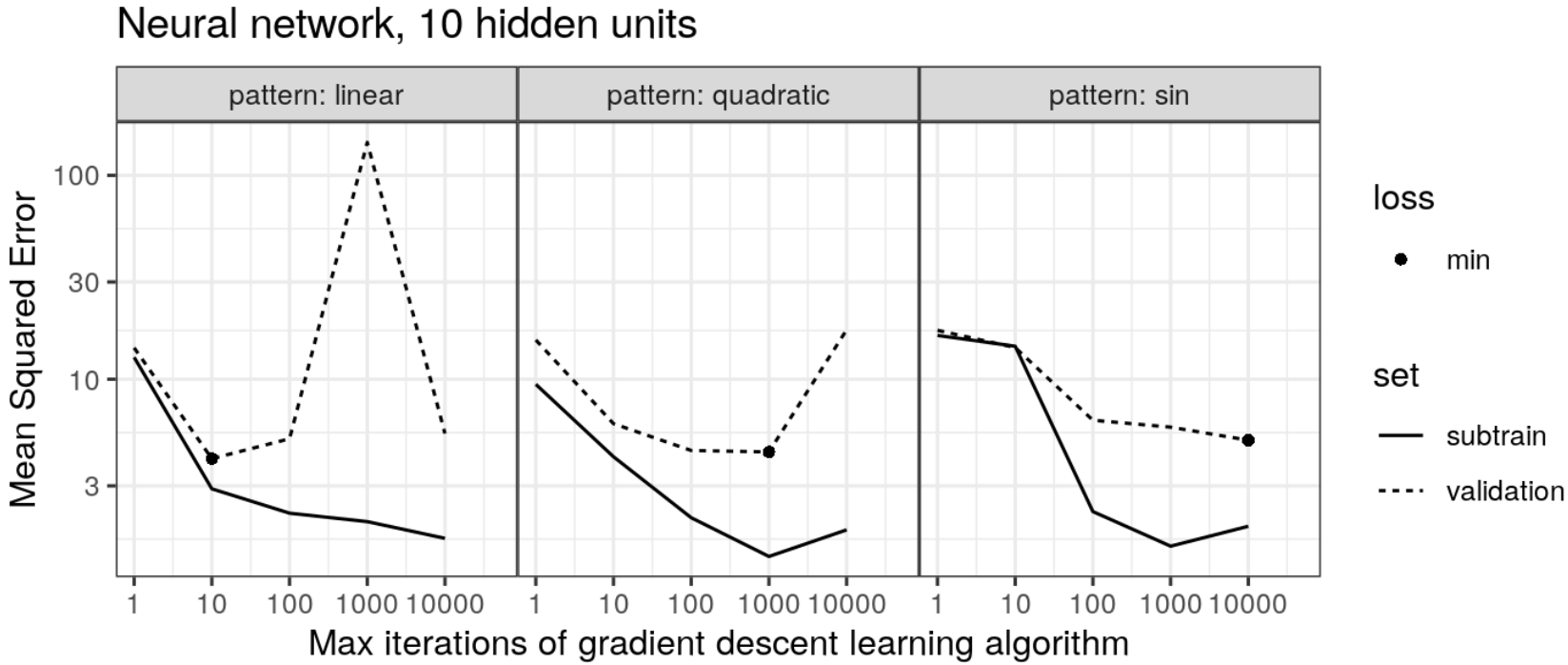

Neural network, 10 hidden units, 1 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

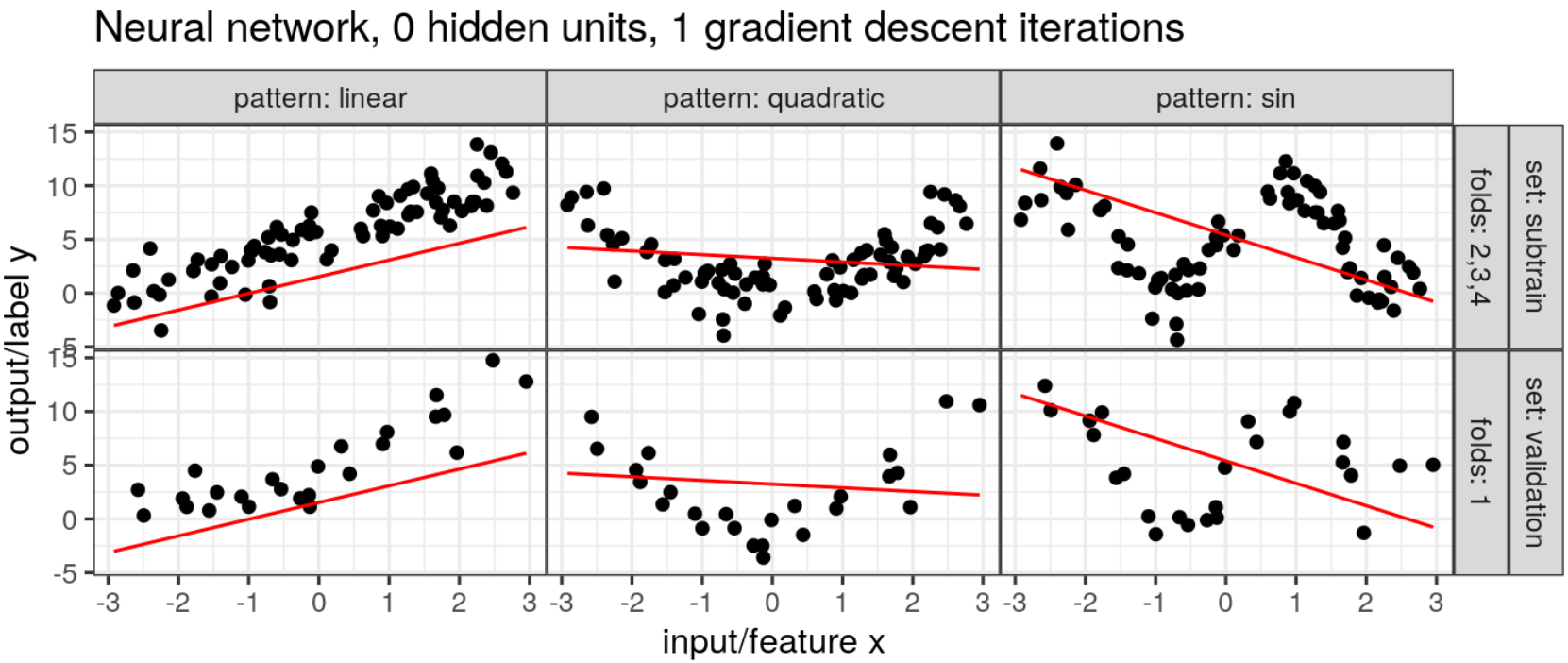

Neural network, 10 hidden units, 10 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

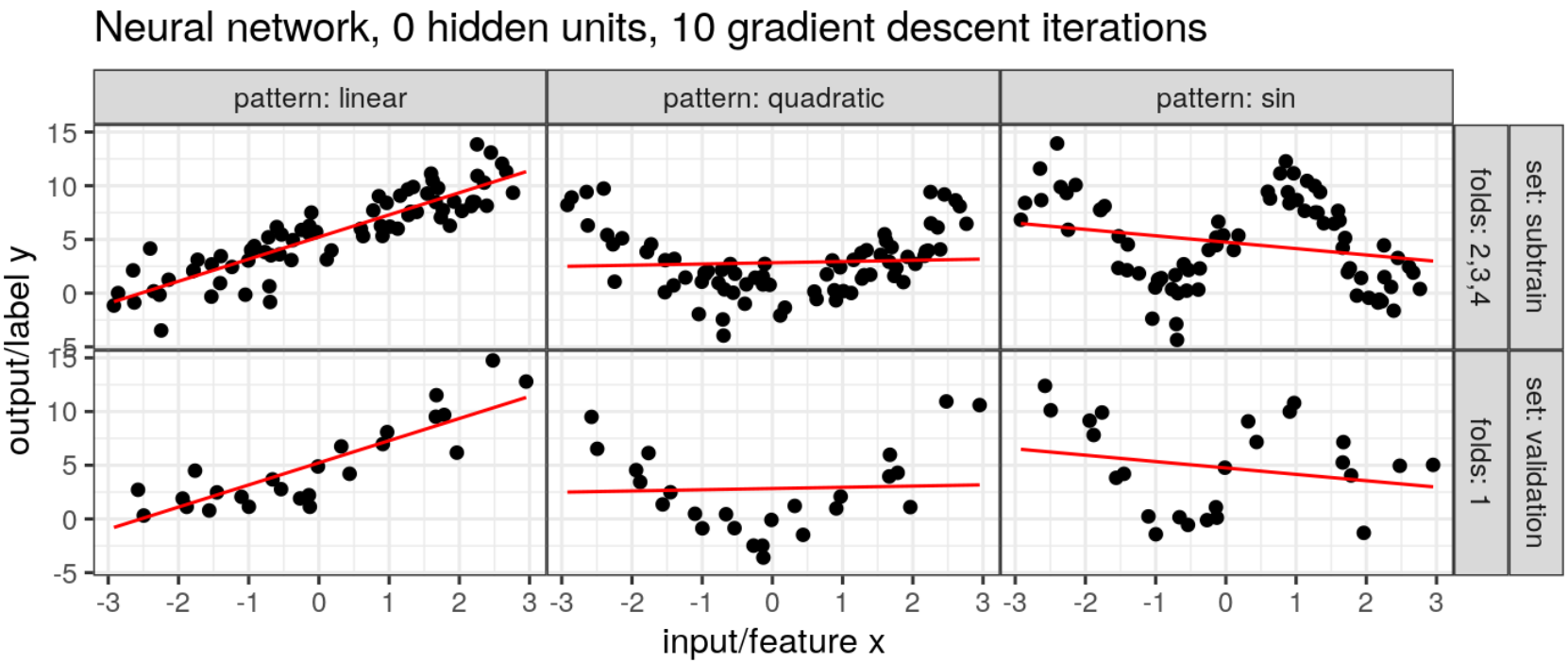

Neural network, 10 hidden units, 100 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

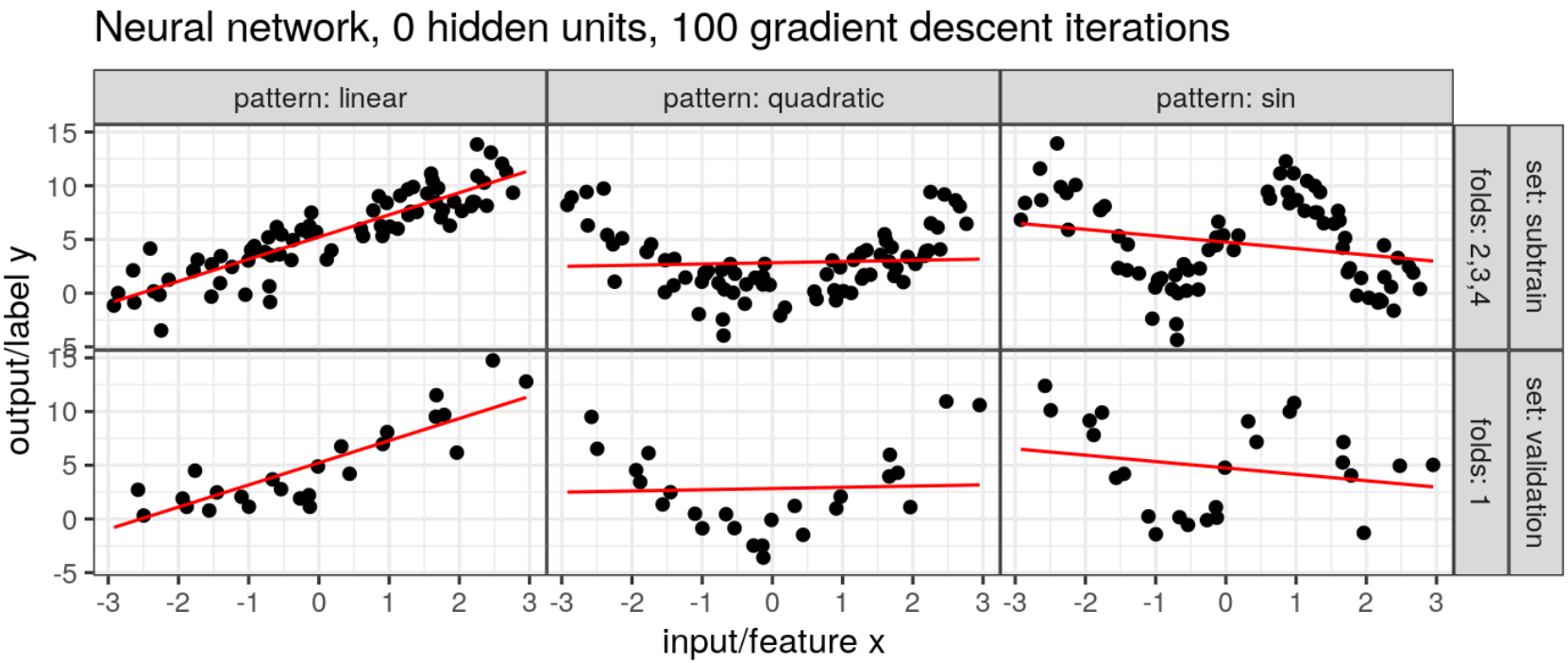

Neural network, 10 hidden units, 1000 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
folds: 2,3,4
set: validation
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

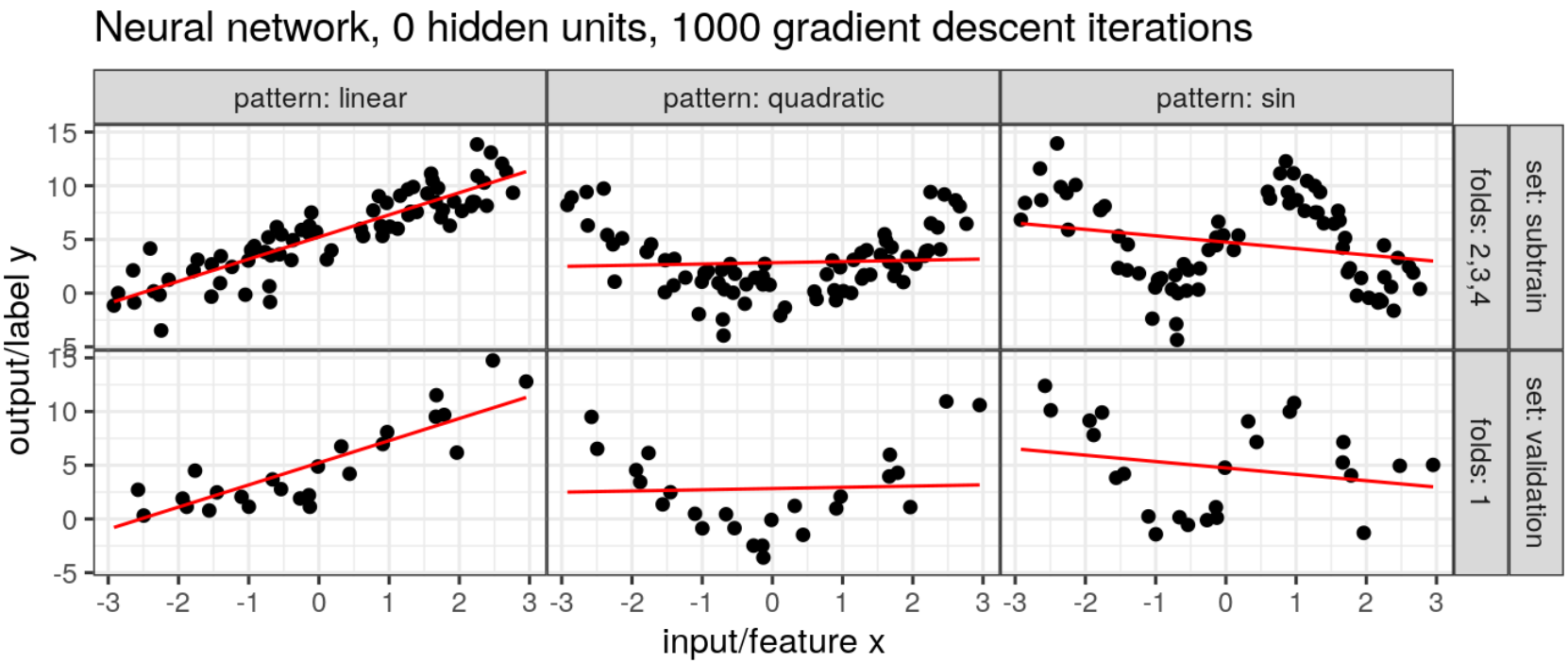

Neural network, 10 hidden units, 10000 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
folds: 2,3,4
set: subtrain
folds: 1
set: validation
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

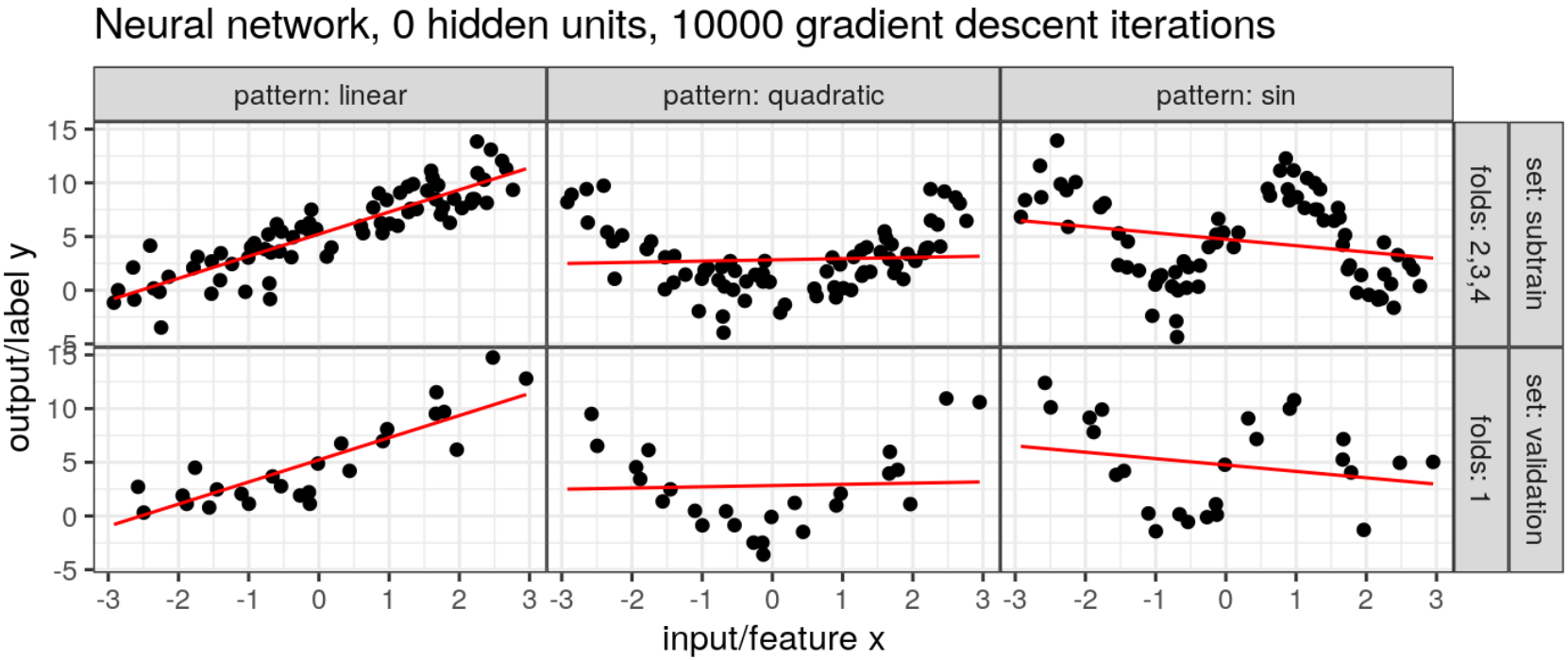

Neural network, 10 hidden units
pattern: linear
pattern: quadratic
pattern: sin
Mean Squared Error
100
30
10
3
1
10
100
1000
10000
Max iterations of gradient descent learning algorithm
loss
min
set
subtrain
validation

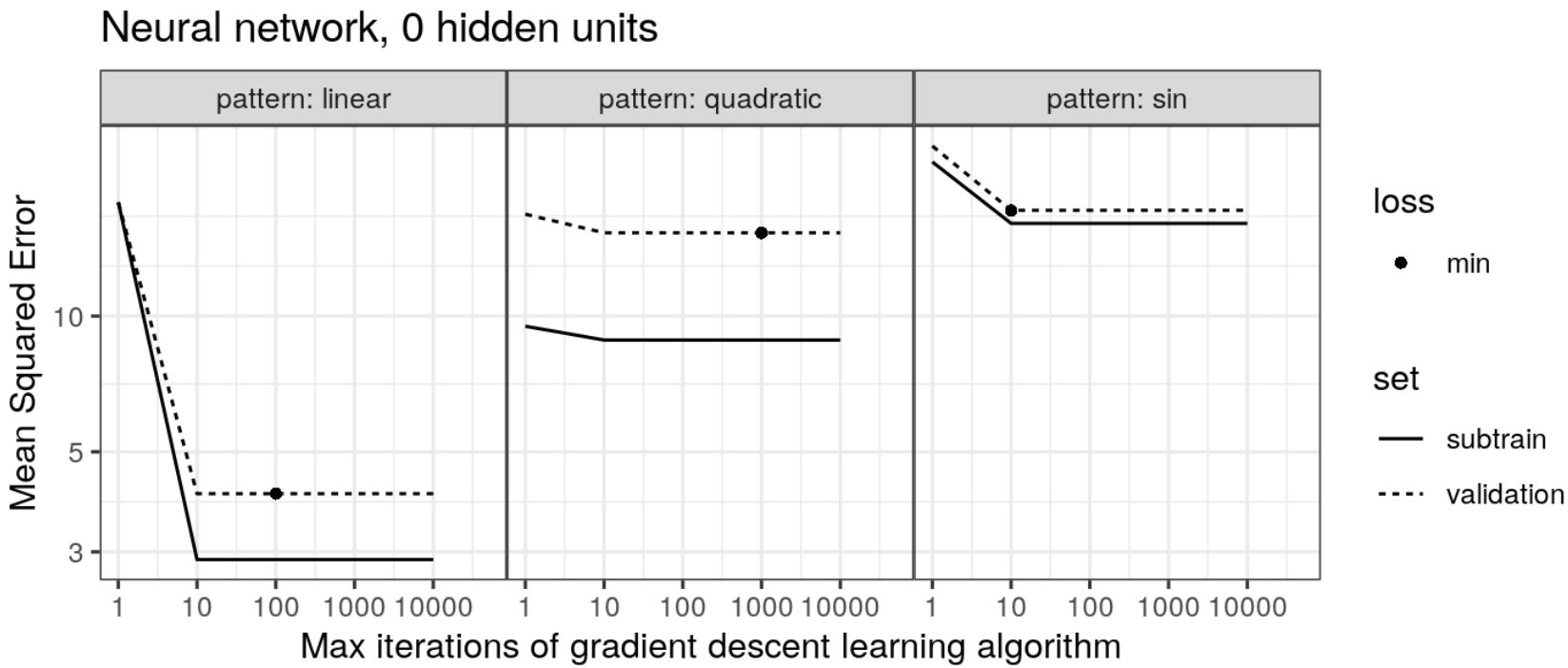

Neural network, 0 hidden units, 1 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

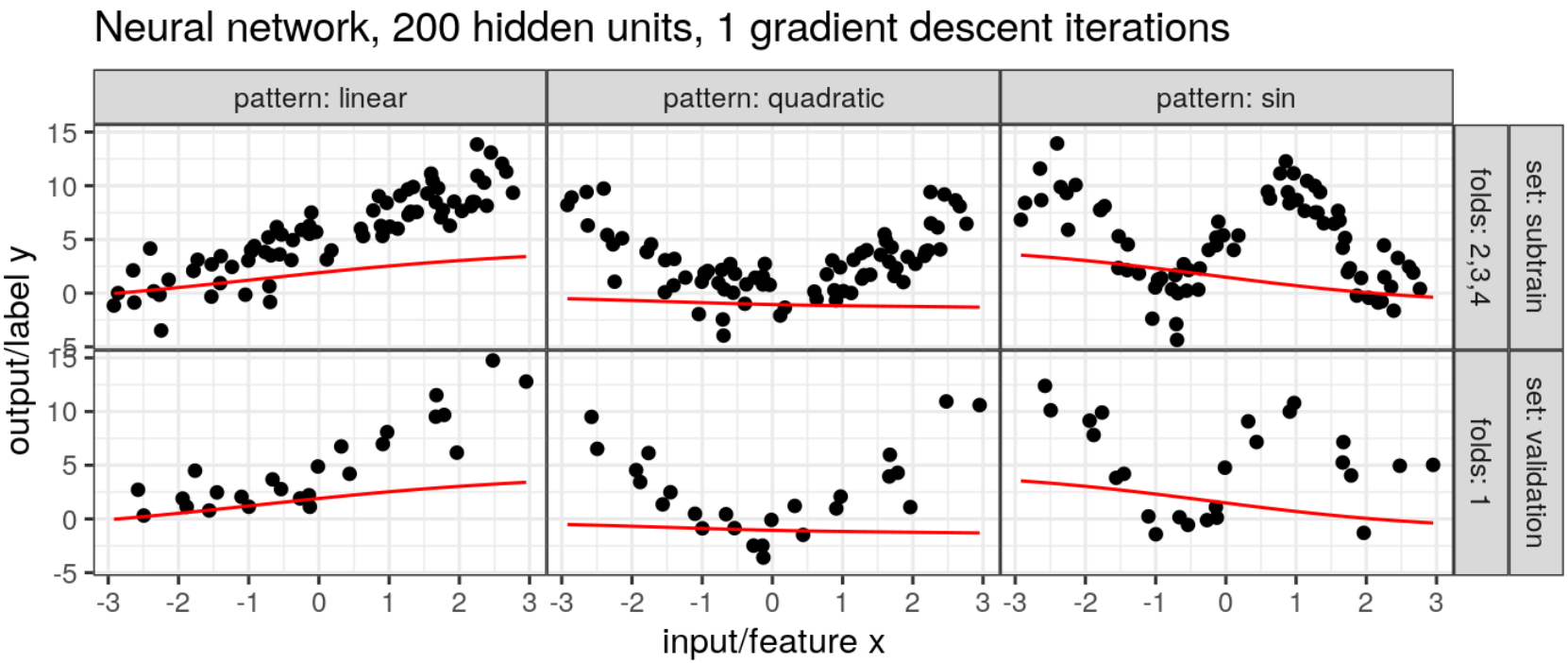

Neural network, 0 hidden units, 10 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

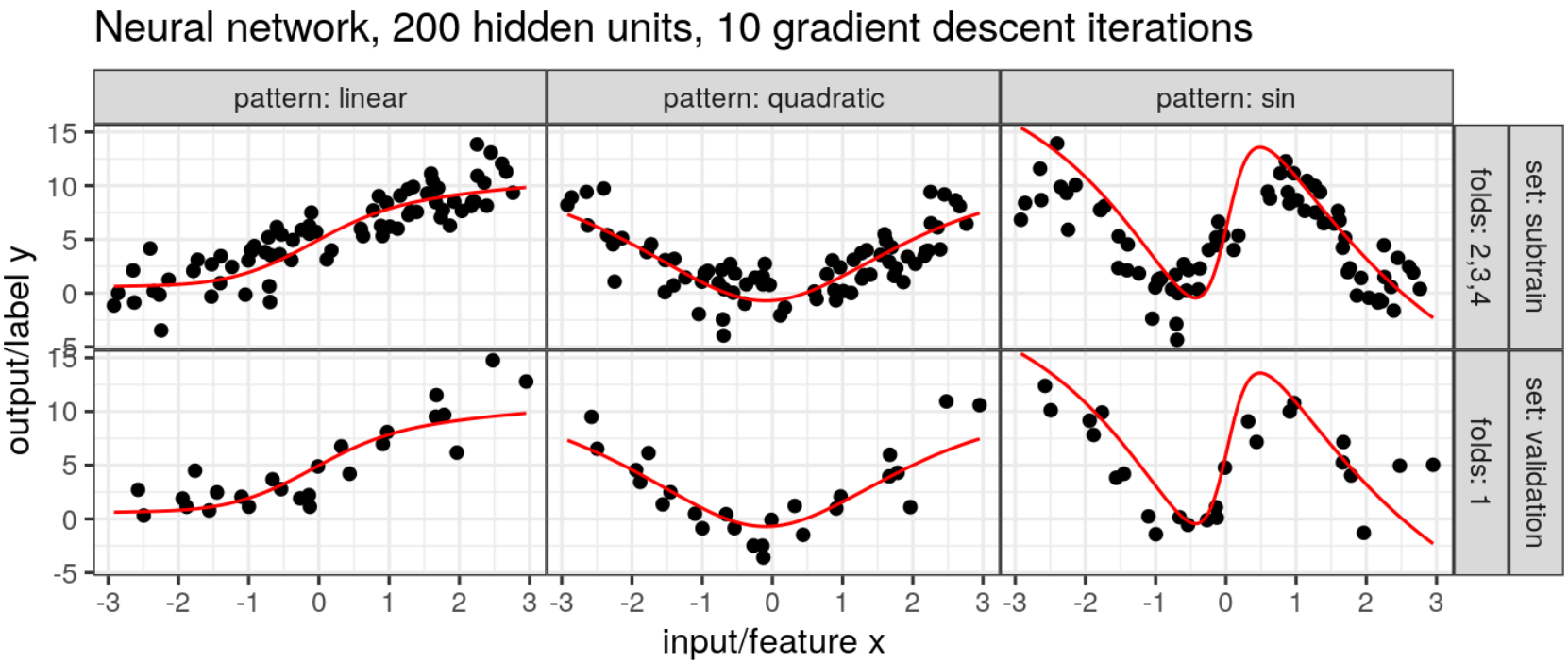

Neural network, 0 hidden units, 100 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

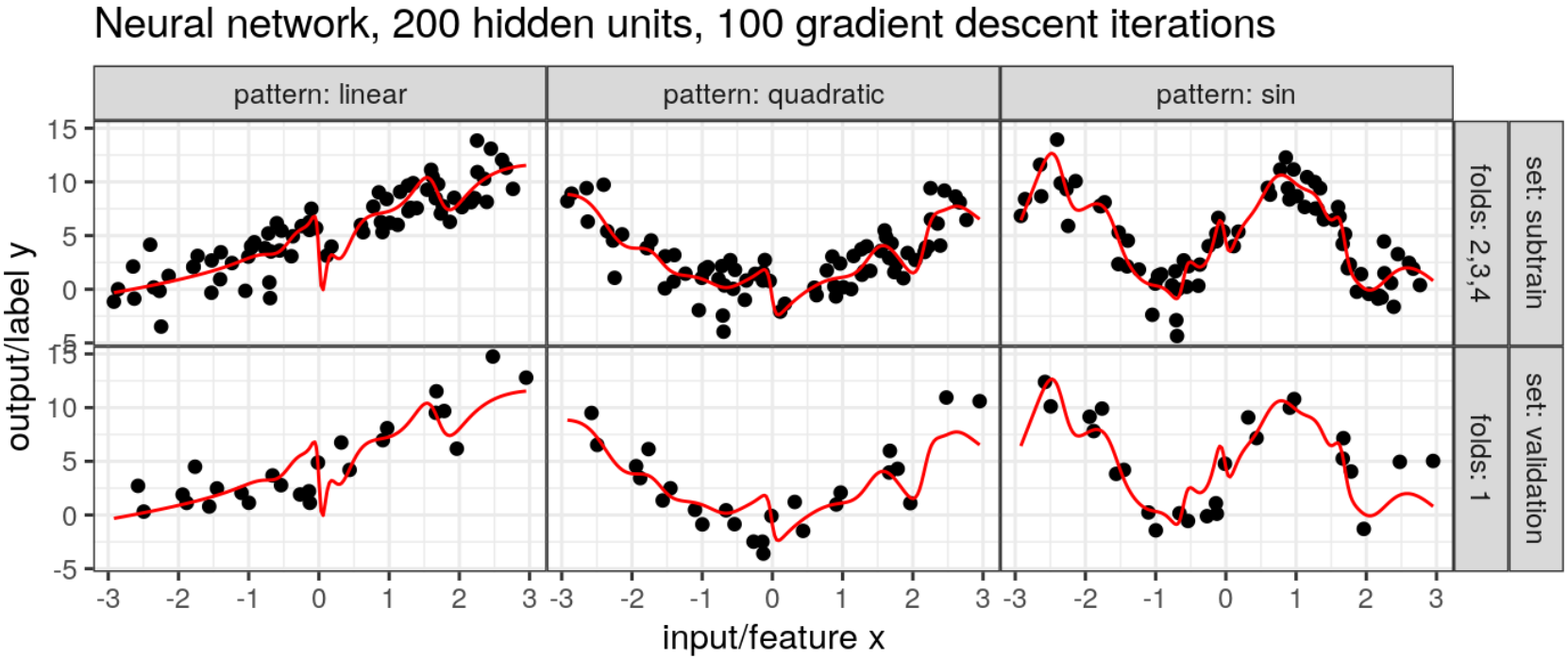

Neural network, 0 hidden units, 1000 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

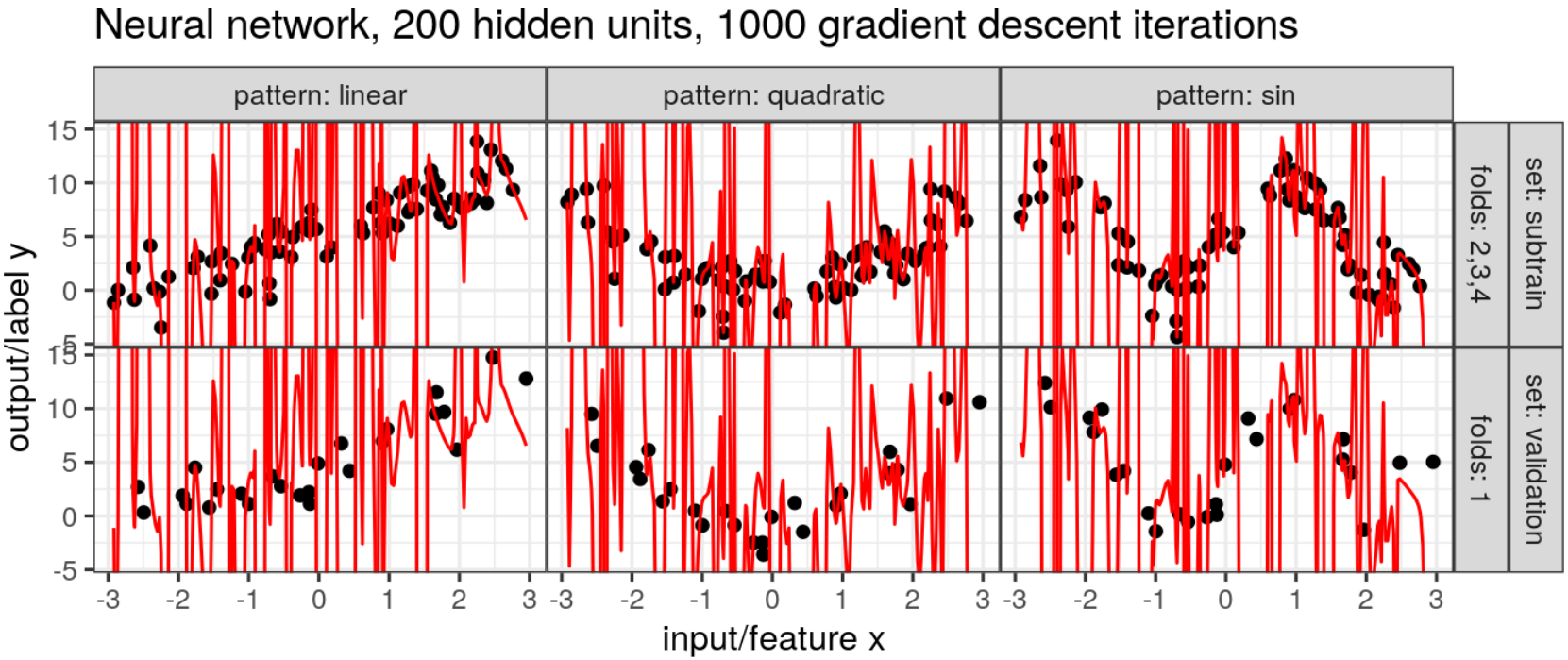

Neural network, 0 hidden units, 10000 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
folds: 2,3,4
set: validation
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

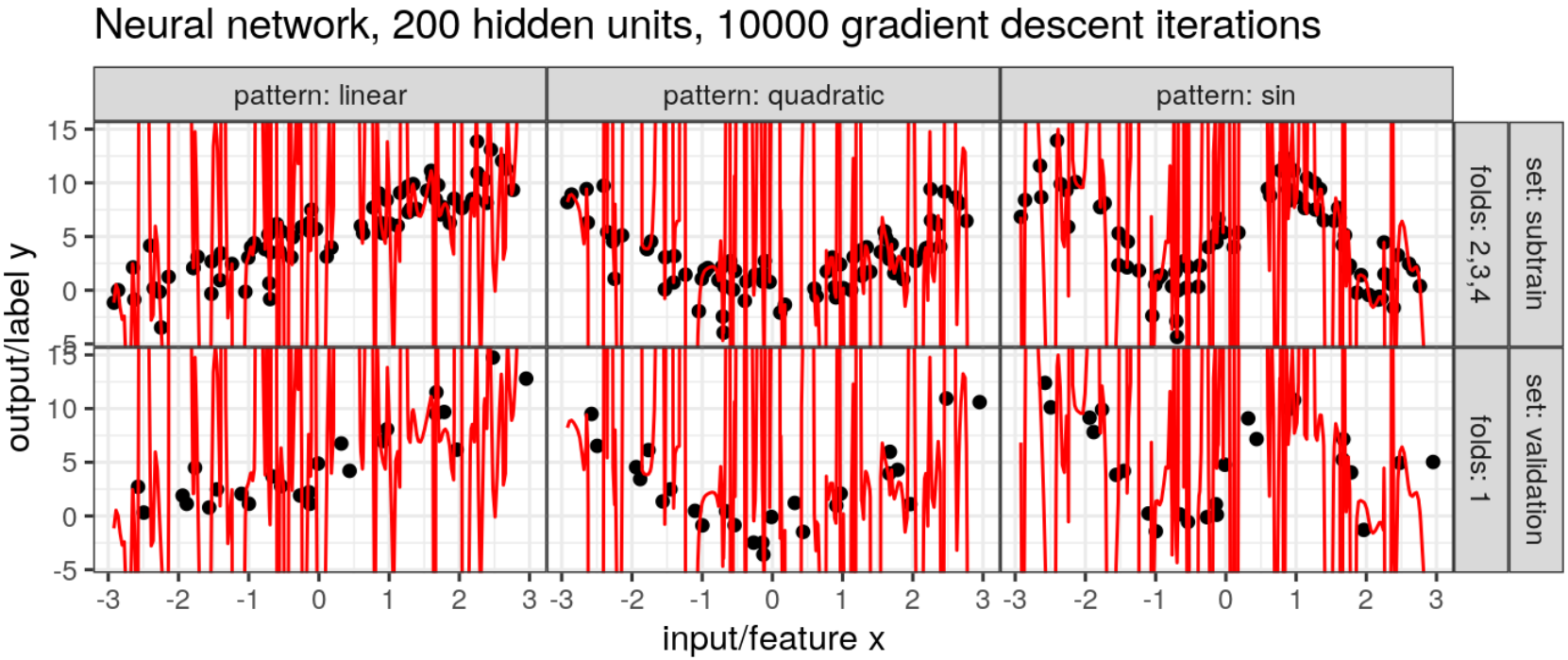

Neural network, 0 hidden units
pattern: linear
pattern: quadratic
pattern: sin
Mean Squared Error
10
5
3
1
10
100
1000
10000
Max iterations of gradient descent learning algorithm
loss
min
set
subtrain
validation

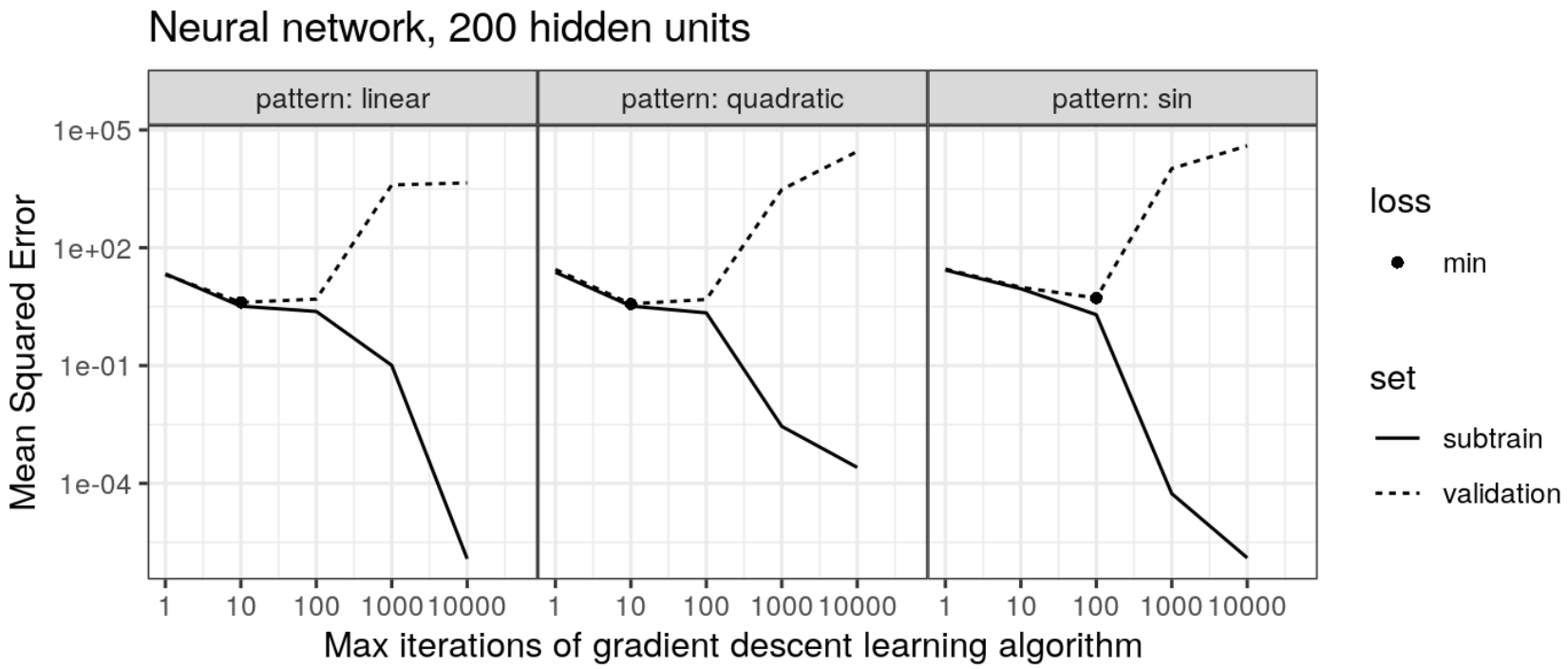

Neural network, 200 hidden units, 1 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3


Neural network, 200 hidden units, 10 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3


Neural network, 200 hidden units, 100 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3


Neural network, 200 hidden units, 1000 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3

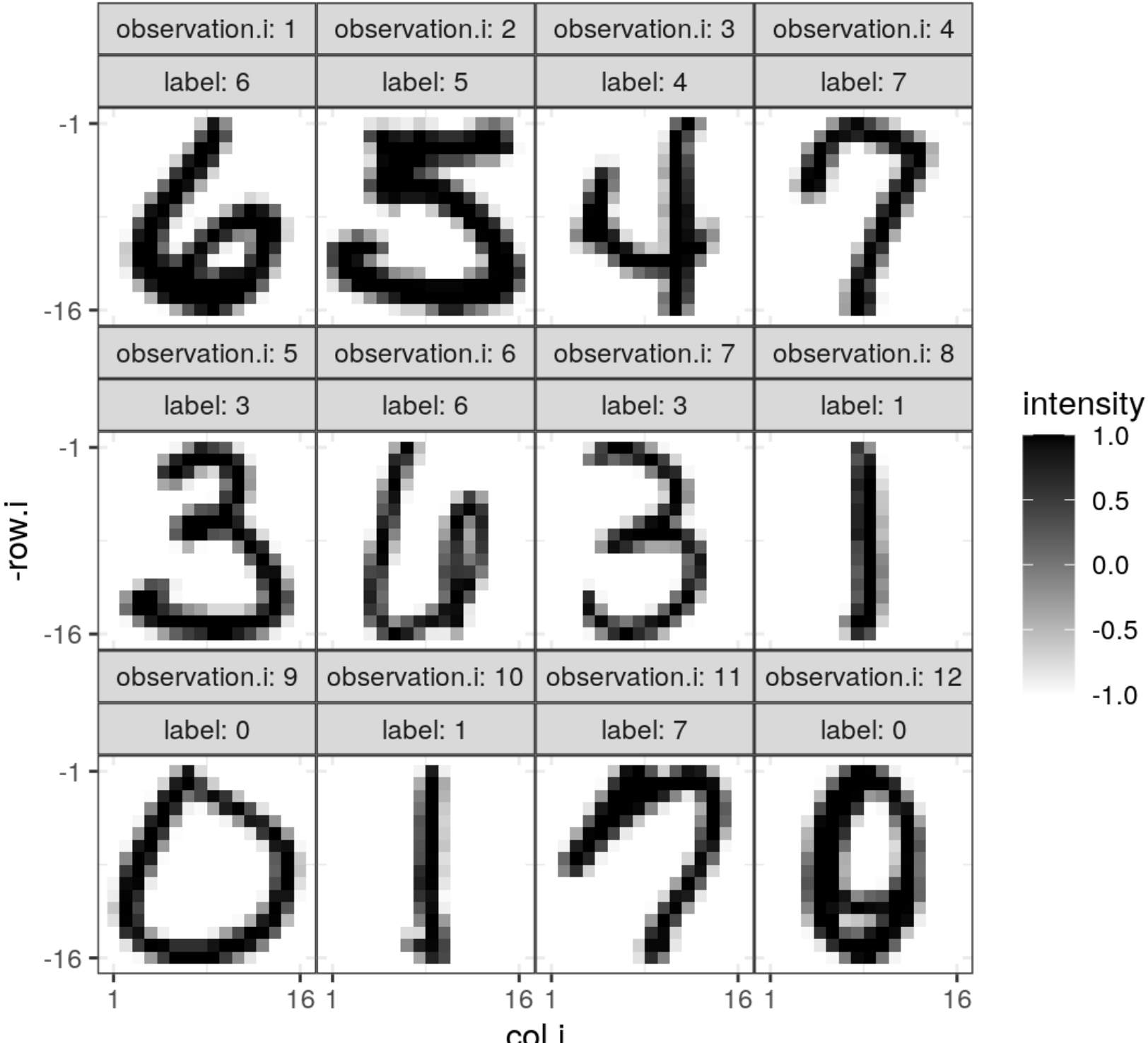

Neural network, 200 hidden units, 10000 gradient descent iterations
pattern: linear
pattern: quadratic
pattern: sin
set: subtrain
set: validation
folds: 2,3,4
folds: 1
output/label y
input/feature x
15
10
5
0
-5
-3
-2
-1
0
1
2
3


Neural network, 200 hidden units
pattern: linear
pattern: quadratic
pattern: sin
Mean Squared Error
1e+05
1e+02
1e-01
1e-04
1
10
100
1000
10000
Max iterations of gradient descent learning algorithm
loss
min
set
subtrain
validation

Introduction and applications

Demonstration of overfitting in regression / first steps with R

Classifying images of digits


observation.i: 1
label: 6
observation.i: 2
label: 5
observation.i: 3
label: 4
observation.i: 4
label: 7
observation.i: 5
label: 3
observation.i: 6
label: 6
observation.i: 7
label: 3
observation.i: 8
label: 1
observation.i: 9
label: 0
observation.i: 10
label: 1
observation.i: 11
label: 7
observation.i: 12
label: 0
-1
-16
1
16
-row.i
col.i
intensity
1.0
0.5
0.0
-0.5
-1.0

Representation of digits in CSV

- Each image is one row.
- First row is output/label/class to predict.
- Other 256 columns are pixel intensity values.

```
 1:  6 -1 -1  ... -1.000 -1.000  -1
 2:  5 -1 -1  ... -0.671 -0.828  -1
 3:  4 -1 -1  ... -1.000 -1.000  -1
 4:  7 -1 -1  ... -1.000 -1.000  -1
 5:  3 -1 -1  ... -0.883 -1.000  -1
 6:  6 -1 -1  ... -1.000 -1.000  -1
 7:  3 -1 -1  ... -1.000 -1.000  -1
 8:  1 -1 -1  ... -1.000 -1.000  -1
 9:  0 -1 -1  ... -1.000 -1.000  -1
10:  1 -1 -1  ... -1.000 -1.000  -1
11:  7 -1 -1  ... -1.000 -1.000  -1
12:  0 -1 -1  ... -1.000 -1.000  -1
```

Converting label column to matrix for neural network

This is a "one hot" encoding of the class labels.

```
zip.dt <- data.table::fread("zip.gz")
zip.y.mat <- keras::to_categorical(zip.dt$V1)
```

```
      0 1 2 3 4 5 6 7 8 9
 [1,] 0 0 0 0 0 0 1 0 0 0
 [2,] 0 0 0 0 0 1 0 0 0 0
 [3,] 0 0 0 0 1 0 0 0 0 0
 [4,] 0 0 0 0 0 0 0 1 0 0
 [5,] 0 0 0 1 0 0 0 0 0 0
 [6,] 0 0 0 0 0 0 1 0 0 0
 [7,] 0 0 0 1 0 0 0 0 0 0
 [8,] 0 1 0 0 0 0 0 0 0 0
 [9,] 1 0 0 0 0 0 0 0 0 0
[10,] 0 1 0 0 0 0 0 0 0 0
[11,] 0 0 0 0 0 0 0 1 0 0
[12,] 1 0 0 0 0 0 0 0 0 0
```

Conversion to array for input to neural network

Use array function with all columns except first as data.

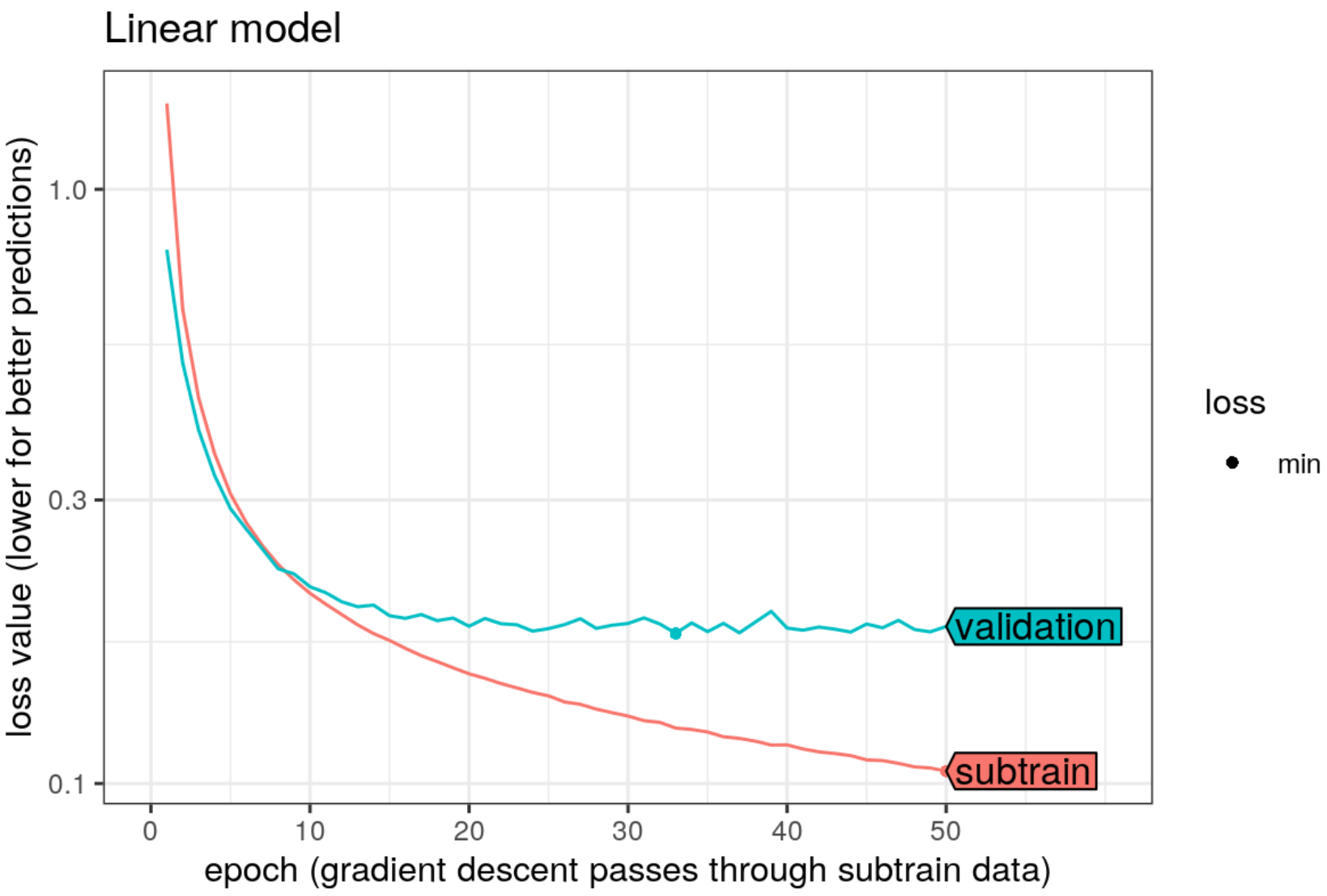
```
zip.size <- 16
zip.X.array <- array(
  data = unlist(zip.dt[1:nrow(zip.dt),-1]),
  dim = c(nrow(zip.dt), zip.size, zip.size, 1))
```

Need to specify dimensions of array:

- Observations: same as the number of rows in the CSV table.
- Pixels wide: 16.
- Pixels high: 16.
- Channels: 1 (greyscale image).

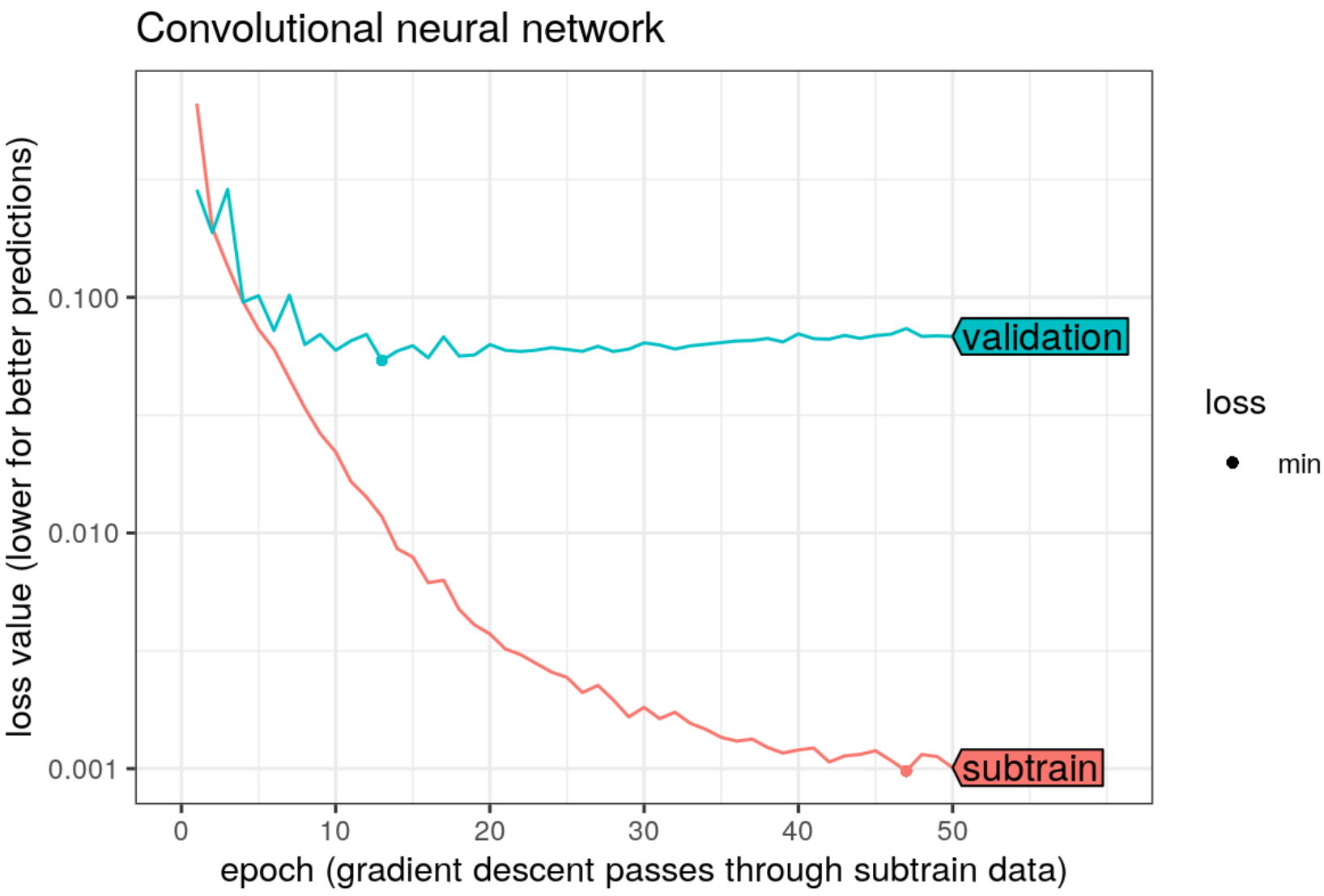
Linear model R code

```
library(keras)
linear.model <- keras_model_sequential() %>%
  layer_flatten(
    input_shape = dim(zip.X.array)[-1]) %>%
  layer_dense(
    units = ncol(zip.y.mat),
    activation = 'softmax')
```


Linear model
loss value (lower for better predictions)
1.0
0.3
0.1
0
10
20
30
40
50
epoch (gradient descent passes through subtrain data)
validation
subtrain
loss
min

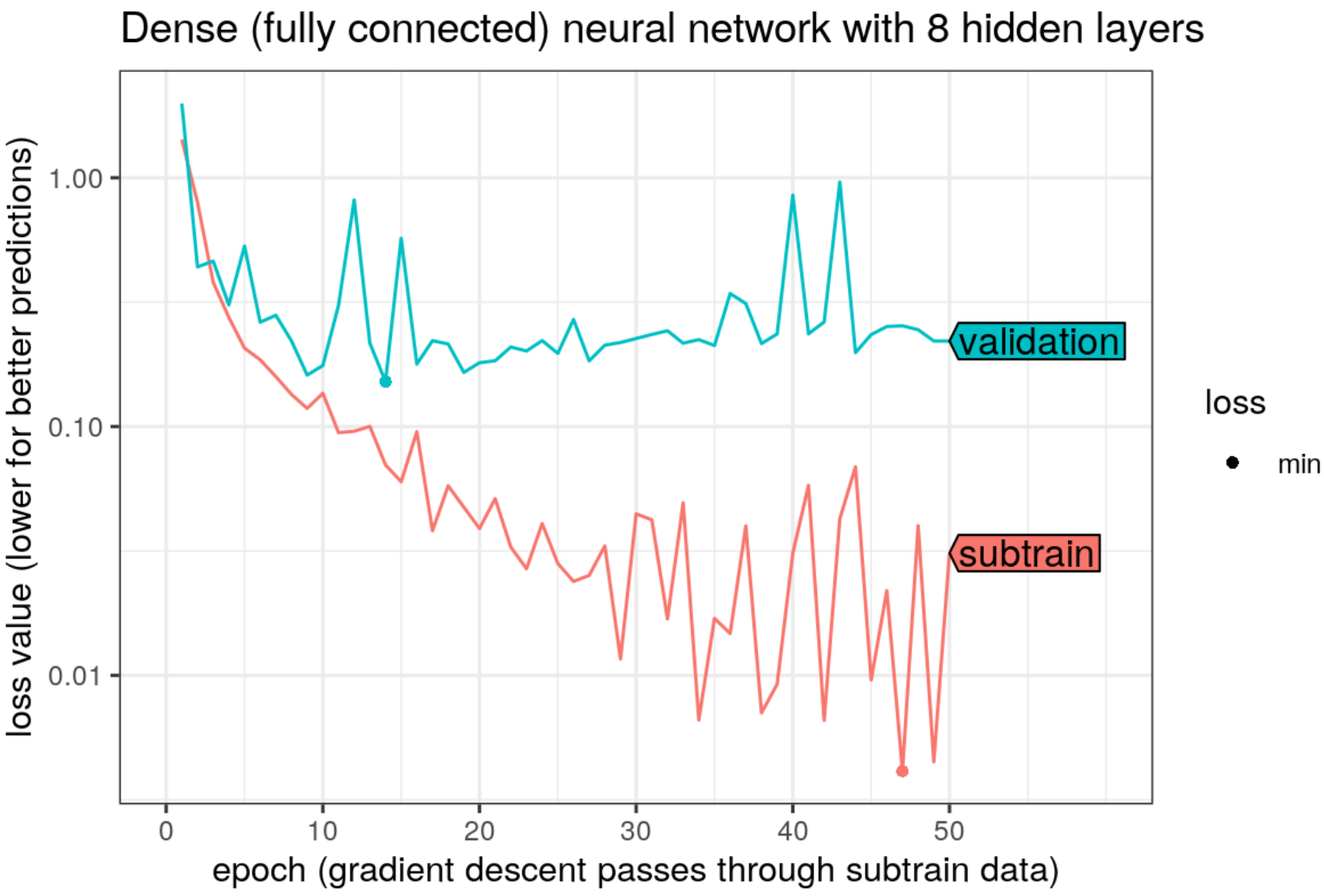
Convolutional model R code

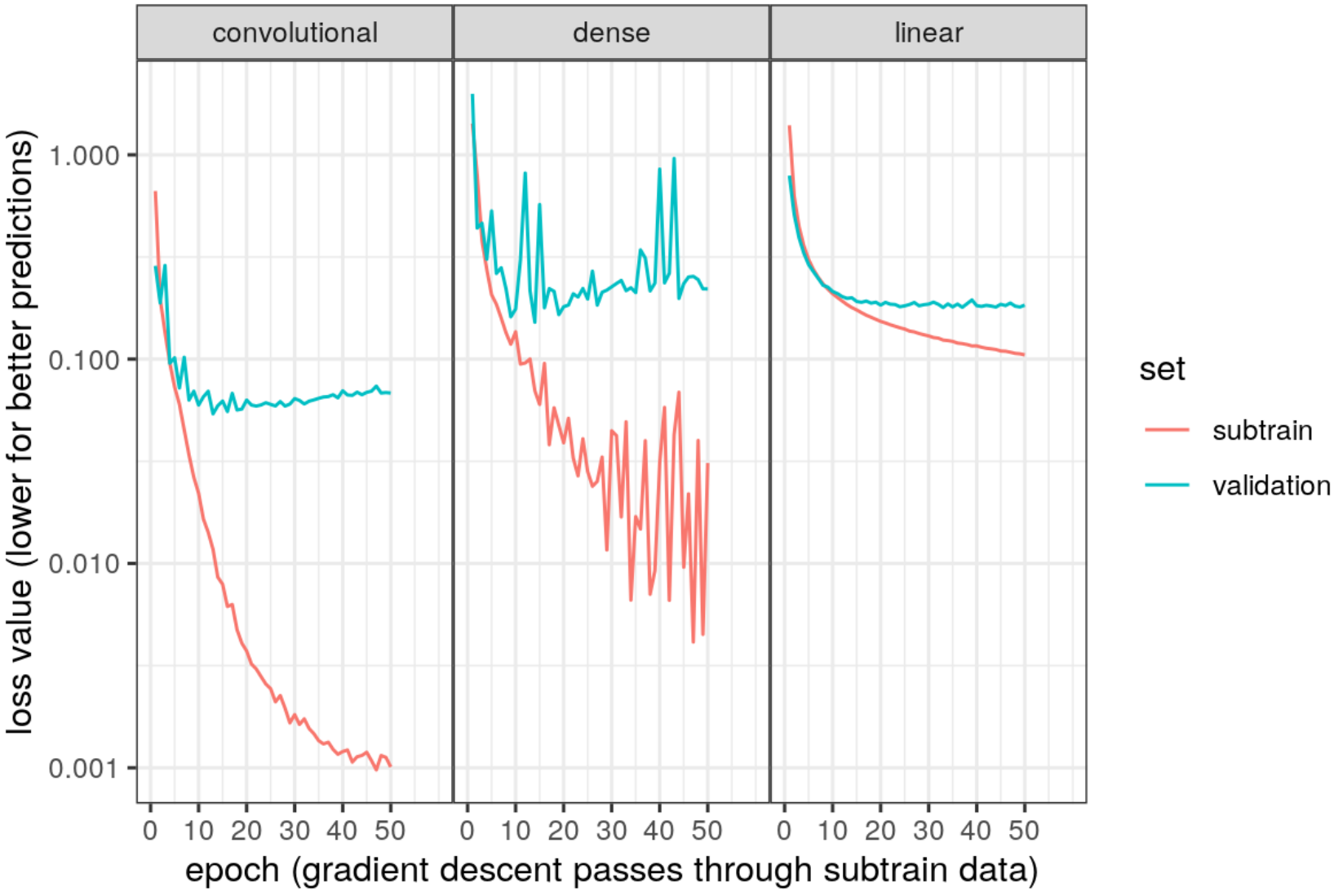
```
library(keras)
conv.model <- keras_model_sequential() %>%
  layer_conv_2d(
    input_shape = dim(zip.X.array)[-1],
    filters = 20,
    kernel_size = c(3,3),
    activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(
    units = ncol(zip.y.mat),
    activation = 'softmax')
```

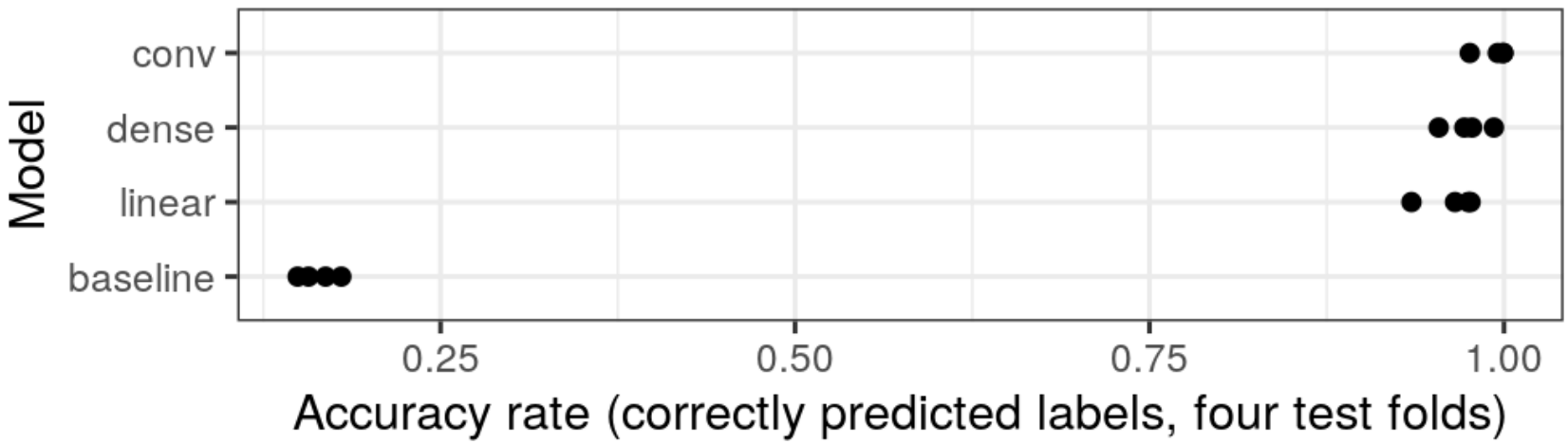

Convolutional neural network
loss value (lower for better predictions)
0.100
0.010
0.001
validation
subtrain
0
10
20
30
40
50
epoch (gradient descent passes through subtrain data)
loss
min

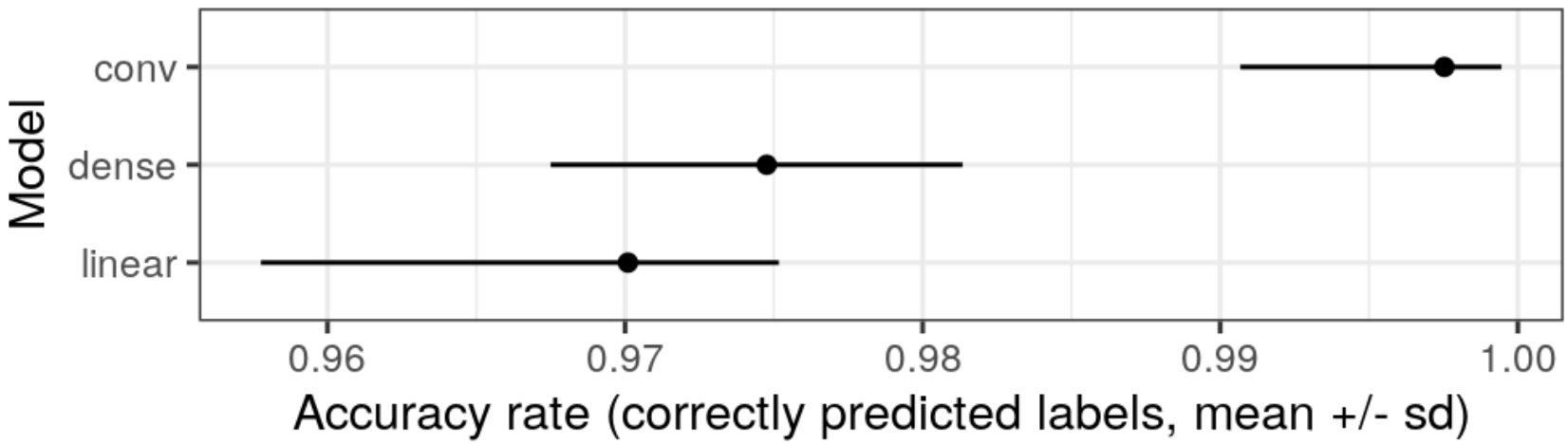
Dense (fully connected) neural network R code

```
library(keras)
dense.model <- keras_model_sequential() %>%
  layer_flatten(
    input_shape = dim(zip.X.array)[-1]) %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(
    units = ncol(zip.y.mat),
    activation = 'softmax')
```


Dense (fully connected) neural network with 8 hidden layers
loss value (lower for better predictions)
1.00
0.10
0.01
0
10
20
30
40
50
epoch (gradient descent passes through subtrain data)
validation
subtrain
loss
min


convolutional
dense
linear
loss value (lower for better predictions)
1.000
0.100
0.010
0.001
0
10
20
30
40
50
epoch (gradient descent passes through subtrain data)
set
subtrain
validation


Model
conv
dense
linear
baseline
0.25
0.50
0.75
1.00
Accuracy rate (correctly predicted labels, four test folds)


Model
conv
dense
linear
0.96
0.97
0.98
0.99
1.00
Accuracy rate (correctly predicted labels, mean +/- sd)