Data science and analysis in Neuroscience

Kevin Allen

December 12, 2019

A brief introduction to machine learning

1. Definition
2. Prediction versus inference
3. Supervised versus unsupervised
4. Regression versus classification
5. Instance-based versus model-based learning
6. Trainind and testing set
7. Quizz!
8. Linear regression
9. Classification
10. Challenges

Objective

The aim is to understand what machine learning is and experiment with a few examples.

Definition of machine learning

Machine learning is the field of study that gives computer the ability to learn without being explicitely programmed.

– Arthur Samuel, 1959

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

– Tom Mitchell, 1997

Examples : A program learns to decide whether an email is spam or not based on training set.

Definition of machine learning

- p different inputs (predictors): $X_1, X_2, X_3, \ldots, X_p$
- Response: Y
- Unknown function: $f()$
- Random error: ϵ

$$Y = f(X) + \epsilon$$

Machine learning refers to a set of approaches for estimating f.

Prediction versus inference

Why do we want to estimate f?

$$\hat{Y} = \hat{f}(X)$$

Prediction

- We focus on predicting Y ($\hat{Y}$).
- $\hat{f}$ is treated as a black box.

Inference

- Understand how Y is affected as $X_1, \ldots, X_p$ changes.
- Which predictors are associated with the response?
- Is the relation between Y and each predictor adequately summarized using a linear equation?

Supervised versus unsupervised versus reinforcement learning

Supervised

- The training set contains labelled data.
- For each observation of the predictors $X_i, i = 1, \ldots, n$ there is a known response measurement y_i.
- Example: linear regression

Unsupervised

- Uncovering hidden patterns from unlabelled data.
- For each observation $i = 1, \ldots, n$, we observed a vector of measurments X_i, but no response y_i.
- Example: cluster analysis

Regression versus classification

- If Y is a continuous variable, then it is a regression task.
- If Y is a categorical variable, then it is a classification task.

Training and test sets

A **training set** is our observed data points that is used to estimate f. Our training set has n observations.

A **test set** is used to test how accurate our model is. Not used for training!

Time for a quizz!

Link

or

https://tinyurl.com/s6gxeuo

Our task

The mice performing rewarded alternation on the t-maze appeared to have improved their performance across the training blocks. Can we estimate how much they improved between each block?

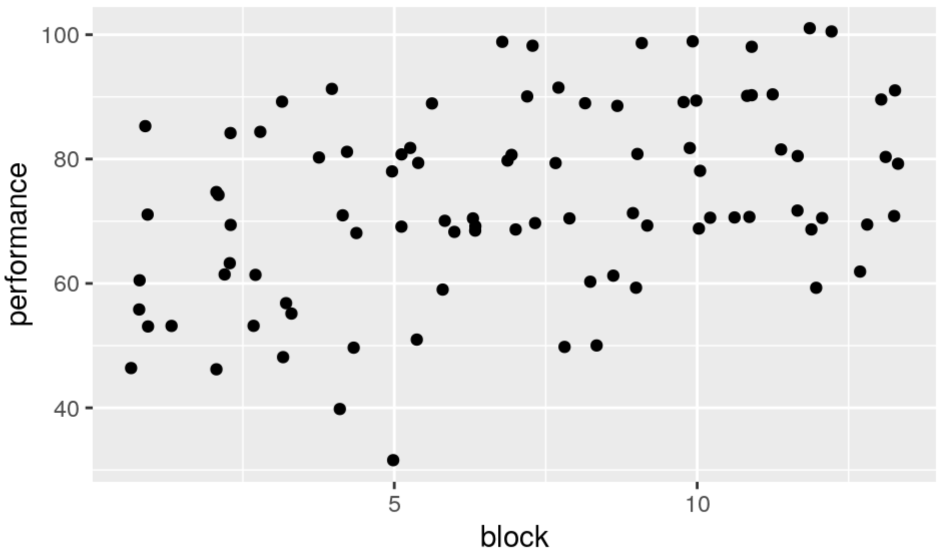
```
{
myFile="~/repo/dataNeuroCourse/dataSets/tmaze.csv"
df<-read_csv(myFile)
df<-mutate(df, correct = sample != choice)

df1 <- df %>%
  group_by(mouse,block) %>%
  summarise(performance = 100 * mean(correct))
}
```

```
## Parsed with column specification:
## cols(
##   mouse = col_character(),
##   date = col_date(format = ""),
##   injection = col_character(),
```

11/27

Our data

```
df1 %>%
  ggplot(mapping = aes(x=block,y=performance)) +
  geom_point(position="jitter")
```



What analysis could we do to estimate the rate of improvment in performance?

Linear regression

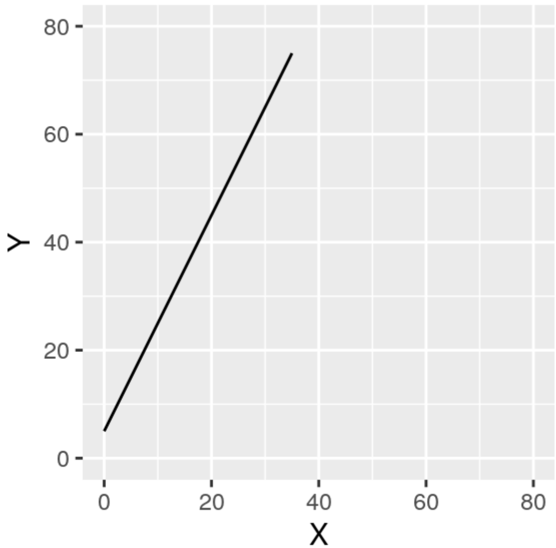
- One of the simplest model to explain your data.
- $Y = aX + b$
- Y: target
- X: features (inputs)
- a and b are parameters of the model.
- a is the slope and b is the intercept.
- The task is to find the best a and b.
- Define an error or loss function to assess any possible line (a and b).
- Find the line that minimize the error function.

Linear regression

- Normally, you use the function `lm()` to find the best fitting (regression) line.
- But to understand how machine learning works, we will do it step-by-step.

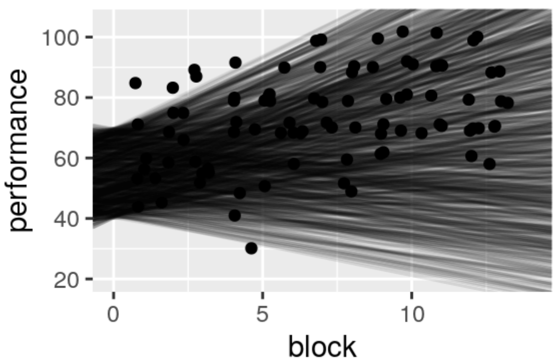
Example of a line

```
a=2 # slope
b=5 # intercept
X=seq(from = 0, to = 35, by = 1) # some input values in X
df<-data.frame(X = X, Y = X * a + b) # our line formula
df %>% ggplot(mapping=aes(x=X,y=Y)) +
  geom_line() +
  xlim(0,80) +
  ylim(0,80)
```



Which line is the best fit for our data?

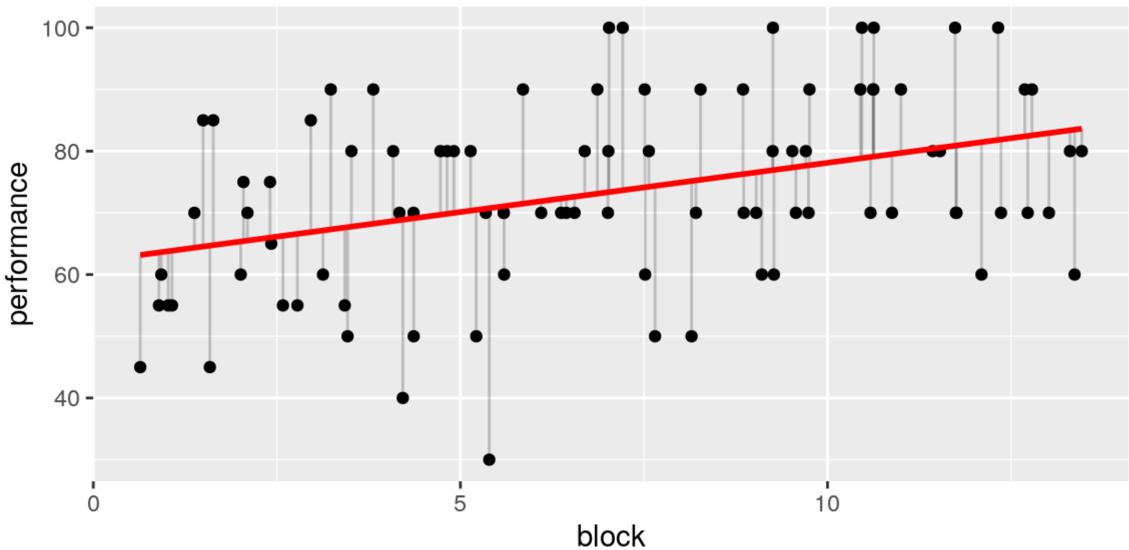
```
set.seed(30)
models<-tibble( # create a data frame with 1000 random lines
  a = runif(n = 1000, min = -2, max = 5), # slope
  b = runif(n = 1000, min = 40, max = 70) # intercept
)
ggplot()+
  geom_point(mapping=aes(x=block,y=performance),position="jitter", data=df1)+
  geom_abline(mapping=aes(intercept=b, slope=a), alpha=0.1, data=models)+
  xlim(0,14)+
  ylim(20,105)
```



16/27

Cost (loss) function

- Based on residuals.
- Residuals: difference between the observed value and the predicted value (line).
- Often used: Sum of the squares of residuals
- Find the line which minimise the loss function



17/27

Cost function

Let's write a function that will give us the predictions of a model.

```
# define a function
one_model_predictions<-function(a, b, data){
  # parameters are a and b
  # data should be our input X
  a * data$block  + b # y = a * x + b
}

# prediction of the model y = 5 * x + 50
one_model_predictions(a=5, b=50, data=df1)
```

```
##  [1]  55  60  65  70  75  80  85  90  95 100 105 110 115  55  60  65  70
## [18]  75  80  85  90  95 100 105 110 115  55  60  65  70  75  80  85  90
## [35]  95 100 105 110 115  55  60  65  70  75  80  85  90  95 100 105 110
## [52] 115  55  60  65  70  75  80  85  90  95 100 105 110 115  55  60  65
## [69]  70  75  80  85  90  95 100 105 110 115  55  60  65  70  75  80  85
## [86]  90  95 100 105 110 115
```

18/27

Cost function

Calculate the difference between the actual and predicted y values. Return root mean square of the difference.

```
measure_distance <- function(a, b, data){
  diff <- data$performance - one_model_predictions(a, b,data)
  sqrt(mean(diff^2))
}

dist <- measure_distance(a=5, b=50, data=df1)
print(paste("Root mean square of residuals: ",dist))
```

```
## [1] "Root mean square of residuals:  22.2930038796485"
```

Best fitting lines

Measure the distance between the prediction of all models and the observed performance

```
models <- models %>%
  mutate(dist = purrr::map2_dbl(a,b,measure_distance,df1))
# purrr::map2_dbl() calls measure_distance for each row of models
head(models,n=5)
```

```
## # A tibble: 5 x 3
##         a     b  dist
##     <dbl> <dbl> <dbl>
## 1 -1.31   64.4  25.3
## 2  1.42   57.2  15.4
## 3  0.548  65.7  15.1
## 4  0.944  51.4  20.9
## 5  0.107  44.0  32.3
```

Best fitting lines

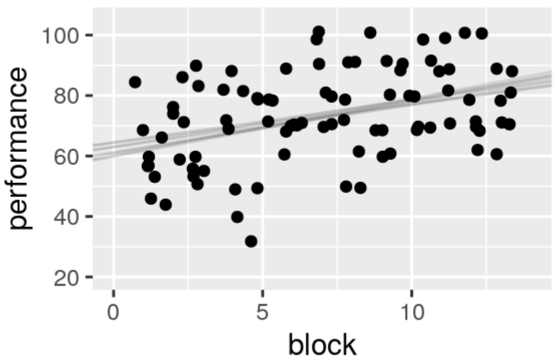
Which model is the best fit?

```
models %>%
  filter(rank(dist)<=8) %>%
  arrange(dist)
```

```
## # A tibble: 8 x 3
##       a     b  dist
##   <dbl> <dbl> <dbl>
## 1  1.71  60.9  14.1
## 2  1.60  63.0  14.1
## 3  1.59  61.2  14.1
## 4  1.39  63.0  14.1
## 5  1.84  59.9  14.1
## 6  1.89  60.1  14.1
## 7  1.29  64.4  14.1
## 8  1.40  64.5  14.1
```

Best fitting line

```
ggplot() +
  geom_point(mapping=aes(x=block,y=performance),position="jitter",
             data=df1) +
  geom_abline(mapping=aes(intercept=b, slope=a), alpha=0.1,
              data=filter(models,rank(dist)<=8)) +
  xlim(0,14) +
  ylim(20,105)
```

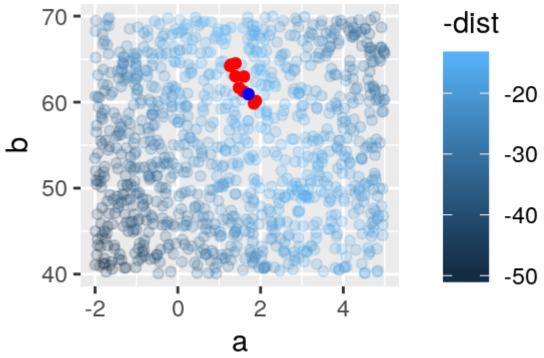


This is way better!

Our parameter search

Our machine learning algorithm found the best fits.

```
ggplot(data=models, mapping = aes(x=a,y=b)) +
  geom_point(aes(color=-dist),alpha=0.2) +
  geom_point(data=filter(models,rank(dist)<=10),color="red") +
  geom_point(data=filter(models,rank(dist)<=1),color="blue")
```



Compare our results to lm()

```
models %>%
  filter(rank(dist)==1)
```

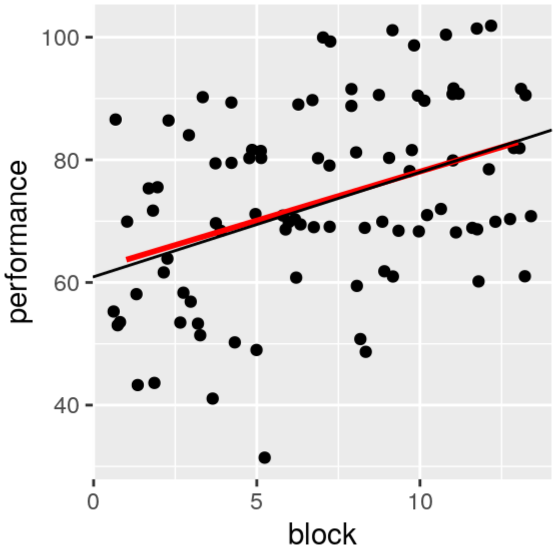
```
## # A tibble: 1 x 3
##       a     b  dist
##   <dbl> <dbl> <dbl>
## 1  1.71  60.9  14.1
```

```
lm(performance~block, data=df1)
```

```
##
## Call:
## lm(formula = performance ~ block, data = df1)
##
## Coefficients:
## (Intercept)        block
##      62.115        1.597
```

Compare our results to lm()

```
ggplot(data=df1,mapping=aes(x=block,y=performance)) +
  geom_point(position="jitter") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  geom_abline(mapping = aes(slope=a,intercept = b),data=filter(models,rank(dist)==1))
```



Impressively close!

For more on machine learning

Online courses

- Datacamp

Good books

- An Introduction to Statistical Learning: With Applications in R
- Hands-On Machine Learning with Scikit-Learn and TensorFlow

For next week

- Read a book chapter: The Machine Learning Landscape (Hands-on machine learning chapter 1)
- Read a Nature Neuroscinece paper: Deeplabcut
- Have a look at the DeepLabCut repository