



HYPERPARAMETER TUNING IN R

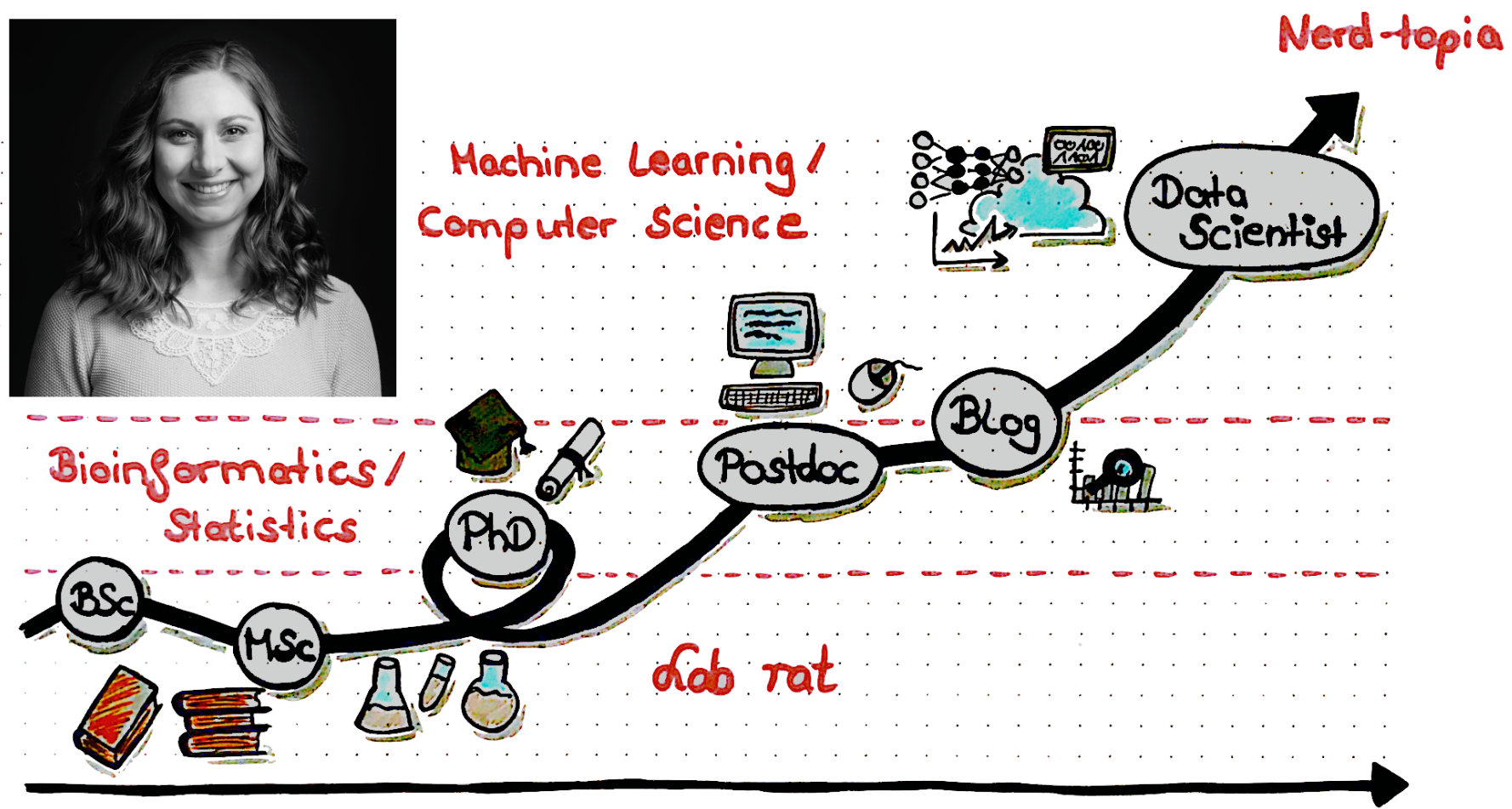
Parameters vs hyperparameters

Dr. Shirin Glander
Data Scientist



About me

www.shirin-glander.de



"Hyper"parameters vs model parameters

- Let's look at an example dataset:

```
head(breast_cancer_data)

# A tibble: 6 x 11
  diagnosis concavity_mean symmetry_mean fractal_dimension_... perimeter_se smooth
  <chr>          <dbl>          <dbl>          <dbl>          <dbl>
1 M          0.300          0.242          0.0787          8.59
2 M          0.0869         0.181          0.0567          3.40
3 M          0.197          0.207          0.0600          4.58
4 M          0.241          0.260          0.0974          3.44
5 M          0.198          0.181          0.0588          5.44
6 M          0.158          0.209          0.0761          2.22
```

- And build a simple **linear model**.

Let's start simple: Model parameters in a linear model

```
# Create linear model
linear_model <- lm(perimeter_worst ~ fractal_dimension_mean,
                  data = breast_cancer_data)

# Get fitted model parameters
summary(linear_model)

# Get residuals
resid(linear_model)

      Min       1Q   Median       3Q      Max
-50.094 -24.859  -7.705   22.209   89.919

# Get coefficients
linear_model$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	167.60	25.91	6.469	3.9e-09	***
fractal_dimension_mean	-926.39	392.86	-2.358	0.0204	*

Let's start simple: Model parameters in a linear model

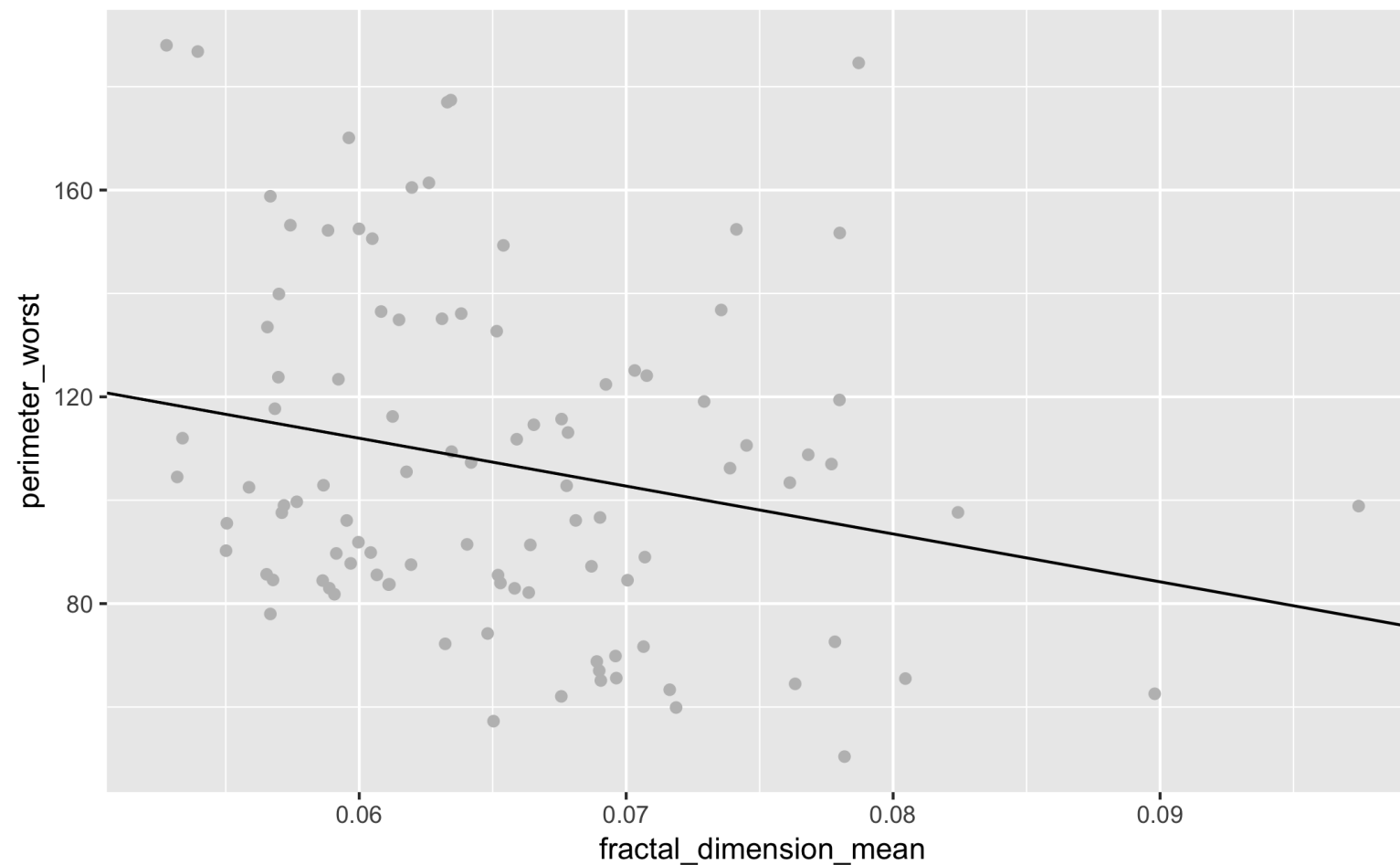
- Model **parameters** are being fit (i.e. found) during training.
- They are the **result** of model fitting or training.
- In a linear model, we want to find the **coefficients**.

```
> linear_model$coefficients  
  
      (Intercept) fractal_dimension_mean  
          167.5972           -926.3866
```

- We can think of them as the **slope** and the **y-intercept** of our model.

Coefficients in a linear model

```
ggp <- ggplot(data = breast_cancer_data,  
              aes(x = fractal_dimension_mean, y = perimeter_worst)) +  
  geom_point(color = "grey")  
  
ggp + geom_abline(slope = linear_model$coefficients[2],  
                 intercept = linear_model$coefficients[1])
```



Model parameters vs hyperparameters in a linear model

- *Remember:* model **parameters** are being fit (i.e. found) during training; they are the **result** of model fitting or training.
- **Hyperparameters** are being set before training.
- They specify **HOW** the training is supposed to happen.

```
args(lm)
formals(lm)
help(lm)
?lm

linear_model <- lm(perimeter_worst ~ fractal_dimension_mean,
                  data = breast_cancer_data,
                  method = "qr")
```

Parameters vs hyperparameters in machine learning

In our **linear model**:

- Coefficients were **found** during fitting.
- `method` was an option to set **before** fitting.

In **machine learning** we might have:

- Weights and biases of neural nets that are optimized during training => model **parameters**.
- Options like learning rate, weight decay and number of trees in a Random Forest model that can be tweaked => **hyperparameters**.

Why tune hyperparameters?

- Fantasy football players ~
Hyperparameters
 - Football players' positions ~
Hyperparameter values
-
- Finding the best combination of
players and positions ~ Finding the
best **combination** of
hyperparameters





HYPERPARAMETER TUNING IN R

Let's practice!



HYPERPARAMETER TUNING IN R

Machine Learning with caret - the Basics

Dr. Shirin Glander
Data Scientist

Machine Learning with caret - splitting data

- **Splitting** into training and test data:

```
# Load caret and set seed
library(caret)
set.seed(42)

# Create partition index
index <- createDataPartition(breast_cancer_data$diagnosis, p = .70,
                             list = FALSE)

# Subset `breast_cancer_data` with index
bc_train_data <- breast_cancer_data[index, ]
bc_test_data  <- breast_cancer_data[-index, ]
```

- Training set with enough **power**.
- **Representative** test set.

Train a machine learning model with caret

- Set up **cross-validation**:

```
library(caret)
library(tictoc)

# Repeated CV.
fitControl <- trainControl(method = "repeatedcv",
                           number = 3,
                           repeats = 5)
```

- **Train a Random Forest model:**

```
tic()
set.seed(42)
rf_model <- train(diagnosis ~ .,
                  data = bc_train_data,
                  method = "rf",
                  trControl = fitControl,
                  verbose = FALSE)

toc()

1.431 sec elapsed
```

Automatic hyperparameter tuning in caret

```
rf_model  
  
Random Forest  
  
80 samples  
10 predictors  
2 classes: 'B', 'M'  
  
No pre-processing  
Resampling: Cross-Validated (3 fold, repeated 5 times)  
Summary of sample sizes: 54, 54, 52, 54, 53, 53, ...  
Resampling results across tuning parameters:  
  
  mtry  Accuracy  Kappa  
    2    0.9006783 0.8015924  
    6    0.9126645 0.8253289  
   10    0.8999389 0.7999386  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was mtry = 6.
```



HYPERPARAMETER TUNING IN R

Let's start modeling!



HYPERPARAMETER TUNING IN R

Hyperparameter tuning with caret

Dr. Shirin Glander
Data Scientist

Automatic hyperparameter tuning in caret

```
rf_model  
  
Random Forest  
  
80 samples  
10 predictors  
  2 classes: 'B', 'M'  
  
No pre-processing  
Resampling: Cross-Validated (3 fold, repeated 5 times)  
Summary of sample sizes: 54, 54, 52, 54, 53, 53, ...  
Resampling results across tuning parameters:  
  
  mtry  Accuracy  Kappa  
    2    0.9006783 0.8015924  
    6    0.9126645 0.8253289  
   10    0.8999389 0.7999386  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was mtry = 6.
```



Hyperparameters are specific to model algorithms

- `modelLookup(model)`
- <https://topepo.github.io/caret/available-models.html>

Hyperparameters in Support Vector Machines (SVM)

```
> library(caret)
> library(tictoc)

> fitControl <- trainControl(method = "repeatedcv",
                             number = 3,
                             repeats = 5)

tic()
set.seed(42)
svm_model <- train(diagnosis ~ .,
                  data = bc_train_data,
                  method = "svmPoly",
                  trControl = fitControl,
                  verbose= FALSE)

toc()

3.836 sec elapsed
```

Hyperparameters in Support Vector Machines (SVM)

```
svm_model
```

```
Support Vector Machines with Polynomial Kernel
```

```
...
```

```
Resampling results across tuning parameters:
```

degree	scale	C	Accuracy	Kappa
...				
1	0.100	1.00	0.9104803	0.8211459
...				

```
Accuracy was used to select the optimal model using the largest value.
```

```
The final values used for the model were degree = 1, scale = 0.1 and C = 1.
```

Defining hyperparameters for automatic tuning

- **tuneLength**

```
tic()
set.seed(42)
svm_model_2 <- train(diagnosis ~ .,
                     data = bc_train_data,
                     method = "svmPoly",
                     trControl = fitControl,
                     verbose = FALSE,
                     tuneLength = 5)
```

```
toc()
```

```
7.458 sec elapsed
```

```
svm_model_2
```

```
...
```

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 1, scale = 1 and C = 1.

Manual hyperparameter tuning in caret

- **tuneGrid + expand.grid**

```
library(caret)
library(tictoc)

hyperparams <- expand.grid(degree = 4,
                           scale = 1,
                           C = 1)

tic()
set.seed(42)
svm_model_3 <- train(diagnosis ~ .,
                     data = bc_train_data,
                     method = "svmPoly",
                     trControl = fitControl,
                     tuneGrid = hyperparams,
                     verbose = FALSE)

toc()

0.691 sec elapsed
```

Manual hyperparameter tuning in caret

```
svm_model_3  
  
Support Vector Machines with Polynomial Kernel  
  
...  
  
  Accuracy      Kappa  
0.7772947 0.554812  
  
Tuning parameter 'degree' was held constant at a value of 4  
Tuning parameter 'scale' was held constant at a value of 1  
Tuning parameter 'C' was  
  held constant at a value of 1
```



HYPERPARAMETER TUNING IN R

It's your turn!