Cross Validation

Overfitting

Overfitting

Recall: We are assuming that

$$(\text{output}) = f(\text{input}) + (\text{noise})$$

and we would like to estimate f.

Here's a suggestion:

$$f(\,x\,) := y$$

for every (x,y) we observe

$$\hat{y}_i = y_i$$

We win! MSE of zero!

Overfitting

Recall from Assignment 1:

```
ins <- read_csv("https://www.dropbox.com/s/bocjjyo1ehr5auz/insurance_costs_1.csv?dl=1")
head(ins)
```

```
## # A tibble: 6 x 6
##     age sex      bmi smoker region    charges
##   <dbl> <chr>  <dbl> <chr>  <chr>       <dbl>
## 1    19 female  27.9 yes    southwest  16885.
## 2    33 male    22.7 no     northwest  21984.
## 3    32 male    28.9 no     northwest   3867.
## 4    31 female  25.7 no     southeast   3757.
## 5    60 female  25.8 no     northwest  28923.
## 6    25 male    26.2 no     northeast   2721.
```

Overfitting

More flexible models fit the data better!

```
lr_mod <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

poly_mod_1 <- lr_mod %>%
  fit(charges ~ bmi, data = ins)


poly_mod_20 <- lr_mod %>%
  fit(charges ~ poly(bmi, 20), data = ins)
```

```
ins <- ins %>%
  mutate(
    preds_1 = predict(poly_mod_1,
                      new_data = ins,
                      type = "raw"),
    preds_20 = predict(poly_mod_20,
                       new_data = ins,
                       type = "raw")
  )
```

Overfitting

More flexible models fit the data better!

```
##
## Call:
## stats::lm(formula = charges ~ bmi, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15452  -8361  -2971   4605  41164
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1476.85    2894.60   0.510 0.610166
## bmi           351.66      92.28   3.811 0.000159 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11690 on 429 degrees of freedom
## Multiple R-squared:  0.03275,    Adjusted R-squared:  0.03049
## F-statistic: 14.52 on 1 and 429 DF,  p-value: 0.0001587
```

Overfitting

More flexible models fit the data better!

```
##
## Call:
## stats::lm(formula = charges ~ poly(bmi, 20), data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -16401  -8206  -2636   4114  41743
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12297.1      560.1  21.955  < 2e-16 ***
## poly(bmi, 20)1    44565.4    11628.3   3.832 0.000147 ***
## poly(bmi, 20)2    -7612.9    11628.3  -0.655 0.513035
## poly(bmi, 20)3     1448.5    11628.3   0.125 0.900930
## poly(bmi, 20)4     7823.3    11628.3   0.673 0.501464
## poly(bmi, 20)5     7346.4    11628.3   0.632 0.527892
## poly(bmi, 20)6   -18995.8    11628.3  -1.634 0.103115
## poly(bmi, 20)7   -18479.1    11628.3  -1.589 0.112798
## poly(bmi, 20)8   -15573.9    11628.3  -1.339 0.181211
## poly(bmi, 20)9    -9784.8    11628.3  -0.841 0.400579
## poly(bmi, 20)10 -13239.3    11628.3  -1.139 0.255559
## poly(bmi, 20)11  -6672.8    11628.3  -0.574 0.566392
```

Overfitting

More flexible models fit the data better!

```
ins %>%
  rmse(truth = charges,
          estimate = preds_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      11667.
```

```
ins %>%
  rmse(truth = charges,
          estimate = preds_20)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      11341.
```

Overfitting

More flexible models fit the data better!

Overfitting = "unnecessarily wiggly"

Bias and variance

In this context:

bias = how much the model is fit to the data it is trained on, instead of being generalizeable to *any* data

variance = how much *prediction error* there is on the training data

Bias and variance

Solutions to overfitting:

Theoretical approaches

Theoretical solutions to overfitting

One idea is to come up with a *metric* that penalizes complexity/flexibility in the model.

Basic idea:

(measure of fit) - (number of predictors)

- Adjusted R-squared
- AIC (Akaike Information Criterion)
- BIC (Bayesian Information Criterion)
- Mallow's Cp

Theoretical solutions to overfitting

Pros:

- Easy to compare models quickly: only one number to compute per model.
- Basis for each metric has some mathematical justification

Cons:

- Which one is *most* justified?
- What if they don't agree? (which is common!)

Solutions to overfitting:

Training and test splits

Training and test data

What if we randomly set aside 10% of our dataset to be our test data?

We train the model using only the remaining 90%.

Then we check the prediction accuracy on the test data, which the model could not possibly have *overfit*?

Training and test data

```
# Set seed, so our "randomness" is consistent
set.seed(190498)

# Establish division of data
ins_split <- ins %>% initial_split()

# Save test and training as separate datasets

ins_test <- ins_split %>% testing()
ins_train <- ins_split %>% training()

# Check what happened

dim(ins_test)
```

```
## [1] 107   8
```

```
dim(ins_train)
```

```
## [1] 324   8
```

Training and test data

Fit the models on the training data only

```
poly_mod_1 <- lr_mod %>%
  fit(charges ~ bmi, data = ins_train)


poly_mod_20 <- lr_mod %>%
  fit(charges ~ poly(bmi, 20), data = ins_train)
```

Training and test data

Find model predictions on the test data only

```
ins_test <- ins_test %>%
  mutate(
    preds_1 = predict(poly_mod_1,
                      new_data = ins_test,
                      type = "raw"),
    preds_20 = predict(poly_mod_20,
                       new_data = ins_test,
                       type = "raw")
  )
```

Training and test data

Check model metrics on the test data only

```
ins_test %>%
  rmse(truth = charges,
          estimate = preds_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      11078.
```

```
ins_test %>%
  rmse(truth = charges,
          estimate = preds_20)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard   42212317.
```

YOUR TURN

Open "Activity-Test-Training.Rmd" (or equivalent)

Cross-Validation

Cross-Validation

If the *test/training* split helps us measure model success...

... but it's random, so it's not the same every time...

... why not do it a bunch of times?

k-fold Cross-Validation

k-fold Cross-Validation

Make the folds:

```
ins_cvs <- vfold_cv(ins, v = 10)

ins_cvs
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits             id
##    <list>             <chr>
##  1 <rsplit [387/44]> Fold01
##  2 <rsplit [388/43]> Fold02
##  3 <rsplit [388/43]> Fold03
##  4 <rsplit [388/43]> Fold04
##  5 <rsplit [388/43]> Fold05
##  6 <rsplit [388/43]> Fold06
##  7 <rsplit [388/43]> Fold07
##  8 <rsplit [388/43]> Fold08
##  9 <rsplit [388/43]> Fold09
## 10 <rsplit [388/43]> Fold10
```

k-fold Cross-Validation

Fit the model on each fold:

```
poly_1_cv <- lr_mod %>%
  fit_resamples(charges ~ bmi,
                resamples = ins_cvs)

poly_2_cv <- lr_mod %>%
  fit_resamples(charges ~ poly(bmi, 2),
                resamples = ins_cvs)
```

k-fold Cross-Validation

Find the average rmse across all 10 splits:

```
poly_1_cv %>% collect_metrics()
```

```
## # A tibble: 2 x 5
##   .metric .estimator       mean     n  std_err
##   <chr>   <chr>           <dbl> <int>    <dbl>
## 1 rmse    standard   11554.        10 662.
## 2 rsq     standard       0.0447    10   0.0155
```

```
poly_2_cv %>% collect_metrics()
```

```
## # A tibble: 2 x 5
##   .metric .estimator       mean     n  std_err
##   <chr>   <chr>           <dbl> <int>    <dbl>
## 1 rmse    standard   11580.        10 647.
## 2 rsq     standard       0.0424    10   0.0145
```

YOUR TURN

Open "Activity-CV.Rmd" or equivalent