Devon Zuegel // 29 Sept 2015 // STATS 202

Problem Set 1

Problem 1

Thoughts
A flexible model:

- allows you to take full advantage of a large sample size (large n).
- will be necessary to find the nonlinear effect.
- fits too much of the noise in the problem (when variance of the error terms is high).
- when you have a lot of predictors, you need to:
  - pay heed to the the bias variance tradeoff
  - be careful to guard against spurious signal

(a) **large sample size `n`, small number of predictors:** When you have few predictors, your model will generally tend to have low variance but high bias; meanwhile, inflexible models tend to encourage even lower variance but higher bias, while flexible models tend to encourage the opposite. Thus, to strike a healhty balance with the variance-bias tradeoff, the *flexible model* will likely perform better.

(b) **small sample size `n`, large number of predictors:** *inflexible* will likely perform better, because with such a small sample size the flexible model would have a greater tendency to overfit the data. Also, the more predictors you have, the more carefully you need to constrain your model space to prevent overfitting. That said, with a small sample size both models will likely be quite inaccurate.

(c) **relationship between the predictors and response is highly non-linear:** assuming our non-flexible model is linear (or something that doesn't fit the true relationship), a *flexible model* will yield better results because then it can react to relationships in the data rather than trying to force the data into a strict mold like non-flexible would.

(d) **extremely high variance of the error terms:** *non-flexible* is better, because it will smooth out the noise from the high variance error terms.

Problem 2

Thoughts

- **prediction:** sing data to predict an event that has yet to occur
- **inference:** inferring the value of a population quantity such as the average income of a country or the proportion of eligible voters who say they will vote "yes"

PART A

regression, inference, `n = 500, p = 4`

PART B

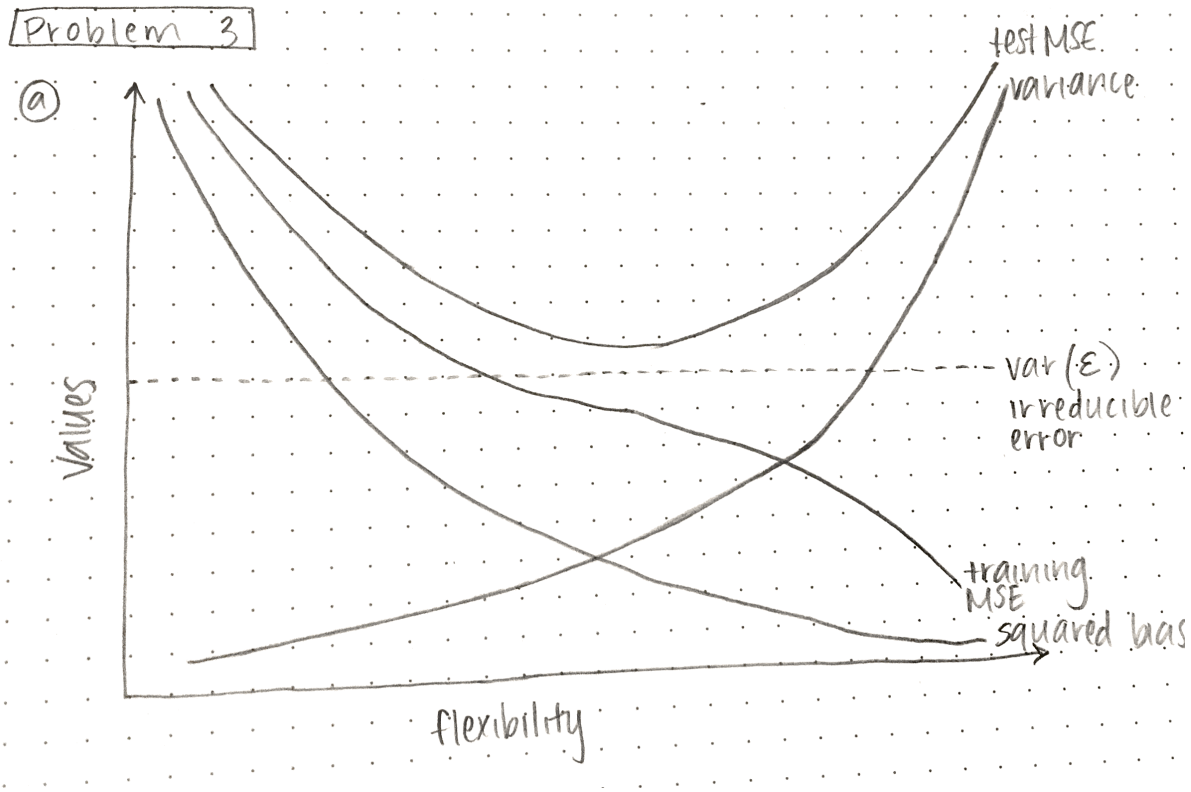
classification, prediction, `n = 20, p = 13`

PART C

regression, prediction, `n = 53 (# of weeks in a year), p = 4`

Problem 3

PART A



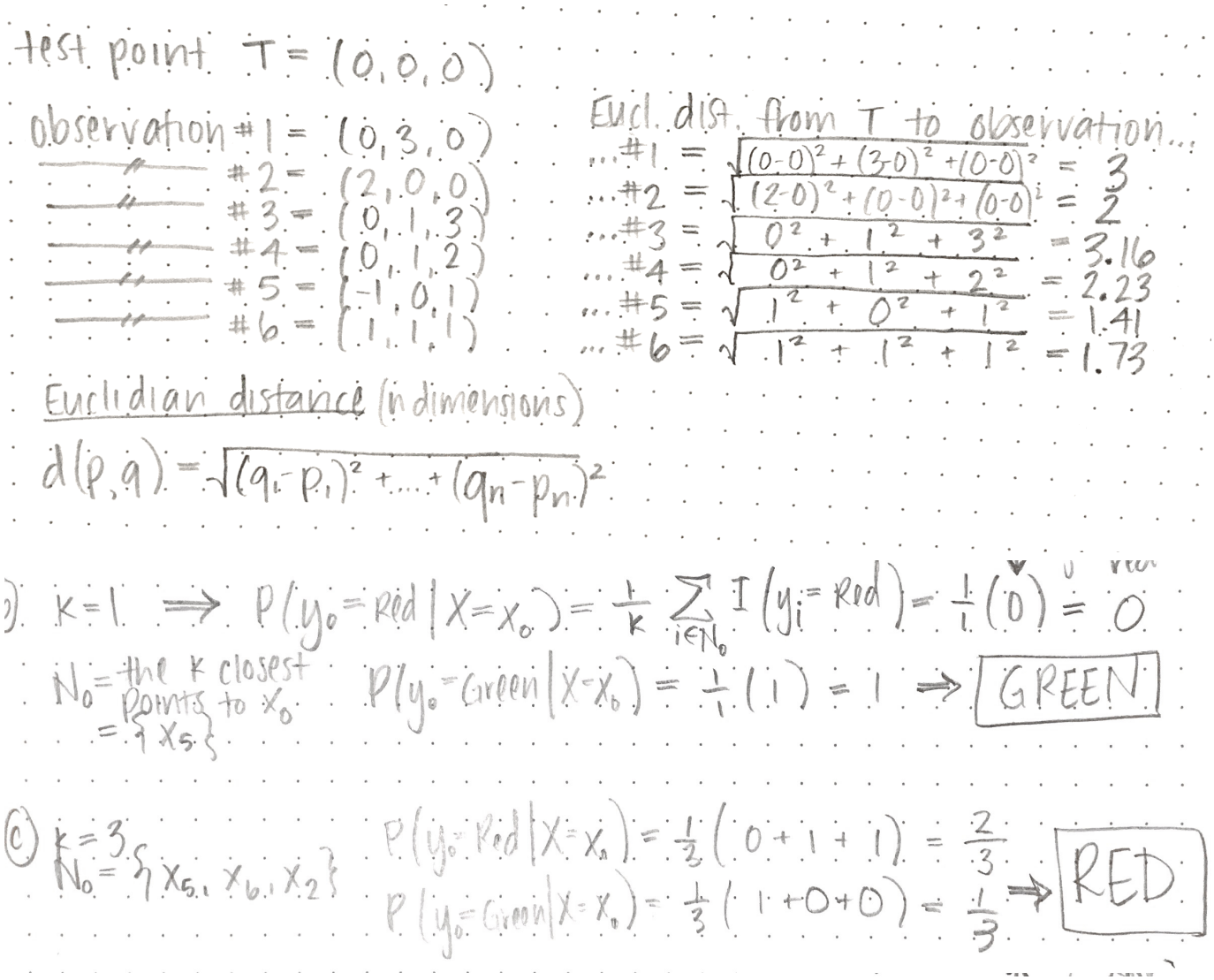
PART B

- **Variance** increases monotonically as flexibility increases, because when our computed `f'` fits the data more closely, we increase the amount by which `f'` would change if we estimated it using a different training data set.
- **Squared bias** declines monotonically as flexibility increases. With inflexible models (aka approximating data with a simple, reductionist model), we run the risk of oversimplifying the relationships within our data. If we do make this mistake, then we end up introducing systematic bias into our approximation of the true function `f`.
- **Var(ϵ), the irreducible error** is constant (though unknown to us unless we generated the data).
- **Test MSE** declines at first, because as flexibility increases the bias decreases. However, increased flexibility leads to increased variance, so at some point the benefits of decreasing bias are outweighed by the variance, which

comes from the fact that we are overfitting our model to the test data. *The test MSE never drops below the irreducible error.*

- **Training MSE** declines as flexibility increases, because the `f'` curve computed from a more flexible model will fit the training data more closely, thus decreasing the mean squared error (MSE).

Problem 4



Problem 5

(a) Yes, we can estimate the test MSE for a fixed point x_0 not included in $x_1, \ldots, x_n$

(b) No, because in order to compute the bias we must know what the true function `f` is. Without it, we have nothing to compare out test data to.

(c) Yes, by definition MSE measures variance

(d) No, for the same reasons as (b)

Problem 6

PART A

```
training_target <- read.csv('training_target.csv')
n_patients = nrow(training_target)
cat("There are", n_patients, "patients in 'training_target.csv'.")
```

```
## There are 2424 patients in 'training_target.csv'.
```

```
summary(training_target)
```

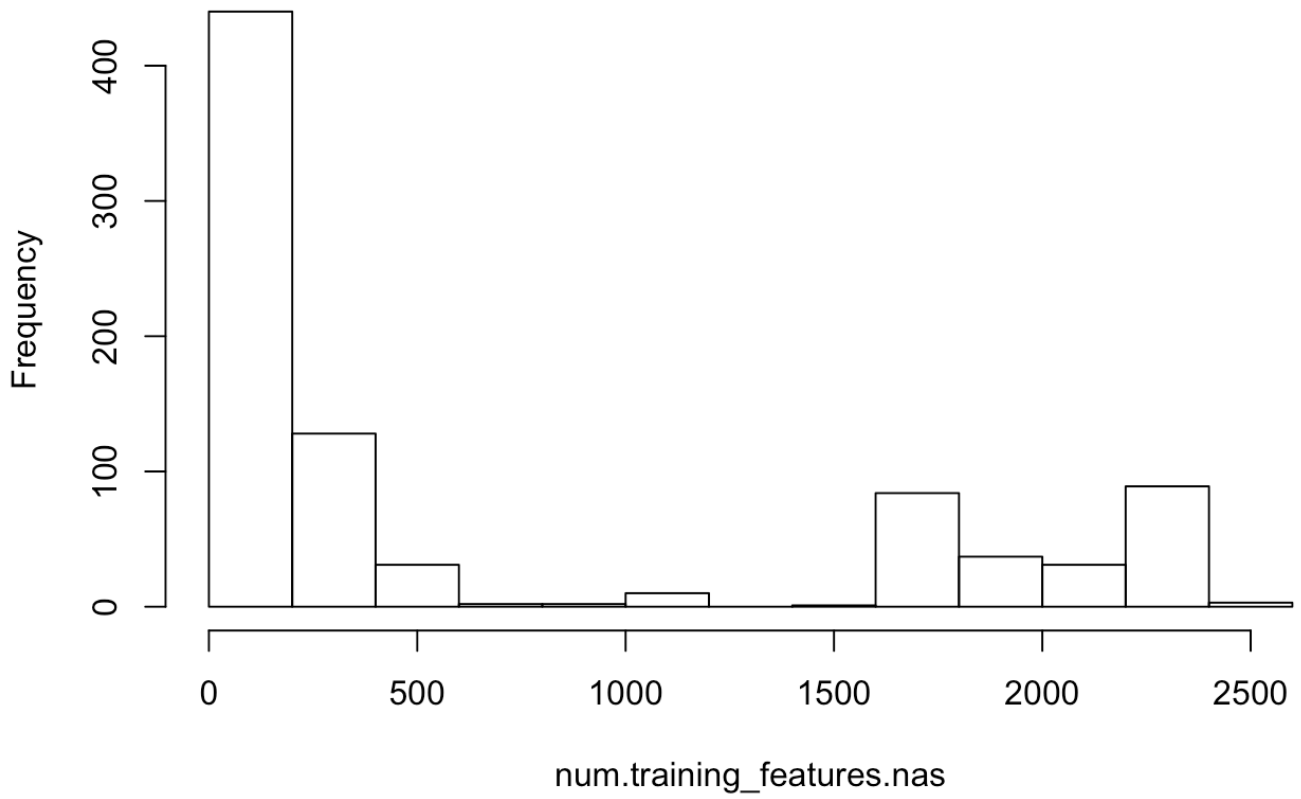
```
##    subject.id      ALSFRS_slope
##  Min.   :   533   Min.   :-4.3452
##  1st Qu.:238072   1st Qu.:-1.0863
##  Median :496688   Median :-0.6207
##  Mean   :497913   Mean   :-0.7308
##  3rd Qu.:753016   3rd Qu.:-0.2742
##  Max.   :999482   Max.   : 1.2070
```

PART B

```
training_features <- read.csv('training_features.csv')
n_features = ncol(training_features)
cat("There are", n_features, "feature columns in 'training_features.csv'.")
```

```
## There are 858 feature columns in 'training_features.csv'.
```

```
num.nas <- function(x) sum(is.na(x))
num.training_features.nas <- apply(training_features, 2, num.nas)
hist(num.training_features.nas)
```



PART C

```
feature.name <- "weight.slope"
dummy.name <- paste0("is.na.",feature.name)
is.na.feature <- is.na(training_features[,feature.name])
training_features[,dummy.name] <- as.integer(is.na.feature)  # Convert boolean val
ues to binary

training_features[is.na.feature,feature.name] <- median(training_features[,featur
e.name], na.rm = TRUE)

feature.names <- names(training_features)
for (feature.name in feature.names[-1]) {  # The [-1] excludes the subject id
    dummy.name <- paste0("is.na.",feature.name)
    is.na.feature <- is.na(training_features[,feature.name])
    training_features[,dummy.name] <- as.integer(is.na.feature)  # Convert boolean
values to binary
}
```

PART D

```
validation_target   <- read.csv('validation_target.csv')
validation_features <- read.csv('validation_features.csv')

cat("There are", ncol(validation_target), "validation patients in 'validation_targ
et.csv' and",
    ncol(validation_features), "validation features in 'validation_features.cs
v'.")
```

```
## There are 2 validation patients in 'validation_target.csv' and 858 validation f
eatures in 'validation_features.csv'.
```

```
summary(training_target)
```

```
##    subject.id      ALSFRS_slope
##  Min.   :   533   Min.   :-4.3452
##  1st Qu.:238072   1st Qu.:-1.0863
##  Median :496688   Median :-0.6207
##  Mean   :497913   Mean   :-0.7308
##  3rd Qu.:753016   3rd Qu.:-0.2742
##  Max.   :999482   Max.   : 1.2070
```

```
summary(validation_target)
```

```
##    subject.id      ALSFRS_slope
##  Min.   : 16979   Min.   :-3.0417
##  1st Qu.:291223   1st Qu.:-1.2674
##  Median :549896   Median :-0.6565
##  Mean   :517002   Mean   :-0.7859
##  3rd Qu.:752533   3rd Qu.:-0.3259
##  Max.   :987497   Max.   : 0.3694
```

The patient in the training data with the minimum ALSFRS slope has one of `-4.3`, while the minimum in the validation data is `-3.0`, a huge difference. Also, the maximim for the training set is `1.2` while the maximum for the validation set is just `0.3`. However, the first quartiles, medians, means, and third quartiles are quite similar across the two data sets. In other words, the training data is much more spread out, while the validation data has lower variation.

```
summary(training_features[,'weight.slope'])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -25.12000  -0.39570   0.00000   0.04101   0.62340   7.60900
```