Admistrivia

- Posted last homework (on Linguistics)

What we did last time

Last time we wrote code to read in the data and compute a variety of useful features. To see the shorten version of this, look at the .R. Look at the .Rnw file from last time to see the code interspersed with the comments.

Here is a traditional statistical regression:

|                          | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------------------|----------|------------|---------|---------|
| (Intercept)              | 14.4531  | 3.1809     | 4.54    | 0.0000  |
| avg.word.length          | -2.4233  | 0.5634     | -4.30   | 0.0001  |

The simple regression on just the average length of words generated a R-squared of 0.22.

And here are results for a small multiple regression:
|                          | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------------------|----------|------------|---------|----------|
| (Intercept)              | 0.5835   | 2.7640     | 0.21    | 0.8335   |
| avg.word.length          | -0.3539  | 0.4740     | -0.75   | 0.4582   |
| freq.the                 | -12.8236 | 3.1428     | -4.08   | 0.0001   |
| freq.of                  | 30.2978  | 5.3431     | 5.67    | 0.0000   |
| freq.to                  | 39.1855  | 5.8546     | 6.69    | 0.0000   |
| freq.and                 | -12.1863 | 8.2133     | -1.48   | 0.1432   |
| freq.in                  | 20.3164  | 8.0588     | 2.52    | 0.0144   |

The R-squared has improved to 0.69.
Here are the predictions.
Why multiple regression doesn’t work

It is nice to use the command `lm(y .)` for running a big regression. But this requires making up a new data frame that is differently organized. The following commands do this:

```r
> tiny <- cbind(federal$Y,federal[,start.freq:(start.freq+10)])
> names(tiny)[1] = "Y"
> small <- cbind(federal$Y,federal[,start.freq:(start.freq+80)])
> names(small)[1] = "Y"
> huge <- cbind(federal$Y,federal[,start.freq:end.freq])
> names(huge)[1] = "Y"
>
> Redoing our plot to ensure we haven’t messed anything up:
```

```r
> tiny.regr <- lm(Y ~ ., tiny)
> plot(predict(tiny.regr,newdata=tiny) ~ federal$number , pch=shapes
> abline(.5,0)
> ```
Now doing it with 80 variables

> small.regr <- lm(Y ~ ., small)
> plot(predict(small.regr,newdata=small) ~ federal$number , pch=shapes , col=colors)
> abline(.5,0)
Naive Bayes to the rescue

```r
> total.prediction <- rep(0,length(rownames(small)))
> for(i in 2:(length(names(small))未来的))
> + {
> +   form <- paste("Y ~ ",names(small)[i])
> +   simple.regression <- lm(formula=form,data=small)
> +   simple.predictions <- predict(simple.regression,newdata=small)
```
```
+ total.prediction = total.prediction + simple.predictions
+ }
> naive.bayes <- lm(small$Y ~ total.prediction)
> intercept = naive.bayes$coefficients[1]
> slope = naive.bayes$coefficients[2]
> plot(intercept + slope * total.prediction ~ federal$number, pch=sizes, col=colors)
> abline(.5,0)
> 
```
This isn’t amazingly better (R-squared = 0.82) but it seems to vote with Madison on all of them or close to it. We can even run it on 100’s of variables:

With 100’s of words the R-squared = 0.94).

> # This takes several hours to run...
> leave.one.out.prediction <- rep(NA,n)
> for(observation in 1:n)
+   {
total.prediction <- rep(0,n)
for(i in 2:d)
{
  form <- paste("Y ~ ", names(huge)[i])
  simple.regression <- lm(formula=form, data=huge[-observation,])
  rsq <- summary(simple.regression)$r.squared
  simple.predictions <- predict(simple.regression, newdata=huge)
  total.prediction = total.prediction + sqrt(rsq) * simple.predictions
}
naive.bayes <- lm(huge$Y ~ total.prediction)
intercept = naive.bayes$coefficients[1]
slope = naive.bayes$coefficients[2]
leave.one.out.prediction[observation] = intercept + slope * total.prediction[observation]
>
plot(leave.one.out.prediction ~ federal$number, pch=shapes, col=colors)
> abline(.5,0)
>