# standGL: Standardized Group Lasso

Noah Simon
Rob Tibshirani

June 8, 2012

## 1 Introduction

We will give a short tutorial on using standGL. We will give the tutorial for linear regression, though it is just as straightforward to run for logistic regression.

## 2 Example

We first load our data and set up the response. In this case $X$ must be an $n$ by $p$ matrix of covariate values by observation ($n$ patients with $p$ covariates) and $y$ is an $n$ length vector of responses. `index` is a $p$ length vector of group memberships. For this example our data has 200 covariate measurements (divided into 5 groups of 40) on 50 observations

```
> set.seed(10)
> library("standGL")
> load("VignetteExample.rdata")
> X <- data$covariates
> y <- data$response
> index <- data$groupMemberships
```

Say, we would also like to fit an intercept — in that case we include a new column of 1s to our covariate matrix, and a new group to our index:

```
> X <- cbind(rep(1,length(y)), X)
> index <- c(0,index)
```

Now we must indicate that this new column is not to be penalized:

```
> is.pen <- c(0,rep(1,(length(unique(index)) - 1)))
> dim(X)

[1]  50 201

> length(y)

[1] 50

> length(is.pen)
```

```
[1] 6
```

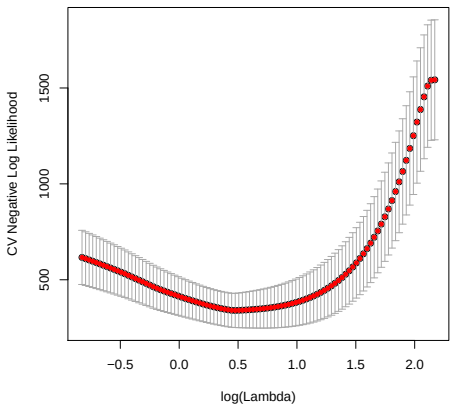
We then call our functions to fit with the standardized group lasso penalty, and cross validate.

```
> cv.fit <- cv.standGL(y,X,index, family="linear", is.pen = is.pen)

NFOLD  1
 ************
NFOLD  2
 ************
NFOLD  3
 ************
NFOLD  4
 ************
NFOLD  5
 ************
NFOLD  6
 ************
NFOLD  7
 ************
NFOLD  8
 ************
NFOLD  9
 ************
NFOLD  10
 ************
```

```
> fit <- standGL(y,X,index, family="linear", is.pen = is.pen)
```

Once fit, we can view the optimal $\lambda$ value and a cross validated error plot to help evaluate our model.

```
> plot(cv.fit)
```




```
> cv.fit$lambda.min
```

```
[1] 1.609562
```

In this case, we see that the minimum was achieved by a fairly regularized model. We can check which covariates our model chose to be active, and see the coefficients of those covariates.

```
> our.Model  <- which(cv.fit$lambdas == cv.fit$lambda.min)
> Active.Index <- which(fit$beta[,our.Model] != 0)
> Active.Coefficients <- fit$beta[Active.Index, our.Model]
> Active.Index

 [1]   1  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
[20] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
[39] 119 120 121
```

We see that there our optimal model chose coefficients 82 to 121 to be active (this corresponds to the third group) and fit an intercept term.

Now, because group sizes are large compared to the number of observations, suppose that we are worried about overfitting within each group. In this case, we can add some slight ridge penalty within group:

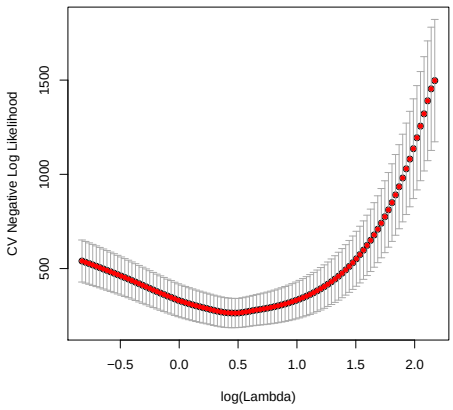
```
> cv.fit.ridge <- cv.standGL(y,X,index, family="ridge", is.pen = is.pen, alpha = 0.95)

NFOLD  1
 ************
NFOLD  2
 ************
NFOLD  3
 ************
NFOLD  4
 ************
NFOLD  5
 ************
NFOLD  6
 ************
NFOLD  7
 ************
NFOLD  8
 ************
NFOLD  9
 ************
NFOLD  10
 ************

> fit.ridge <- standGL(y,X,index, family= "ridge", is.pen = is.pen, alpha = 0.95)
```

We have no strong theoretical rationale for choosing `alpha`= 0.95. We note that `alpha`= 1 corresponds to the standardized group lasso with no ridge penalty, and `alpha`= 0 corresponds to ridge regression with no group penalty. Also, we have seen that, as in the elastic net, adding a slight amount of ridge

regression helps keep our solution well behaved.

Again, we can view the optimal $\lambda$ value and a cross validated error plot to help evaluate our model.

```
> plot(cv.fit.ridge)
```




```
> cv.fit.ridge$lambda.min
```

```
[1] 1.609562
```

Based on the cv-curve we see that the model here fits the data maybe slightly better than the vanilla standardized group lasso (though the error bars certainly overlap). If we desired we could also find the coefficients of our optimal model as before.