Elastic Net Regularization for GLMs and Extensions

Balasubramanian Narasimhan

Stanford University

Jan 24, 2024

Outline

- 1. Elastic-Net Models
  - Introduction
  - `glmnet` package
  - Some implementation details
- Glmnet version 4.0
  - Kenneth Tay's work on GLM and Survival Models
  - Compatibility and Examples
- Applications
  - SNPnet
  - Cooperative Learning
- Summary and Future Work

The Elastic Net

- Supervised learning: given features $\mathbf{X} \in \mathbb{R}^{n \times p}$, response $y \in \mathbb{R}^n$
- Elastic net (Zou & Hastie 2005): $\hat{y} = \hat{\beta}_0 + \mathbf{X}\hat{\beta}$, where

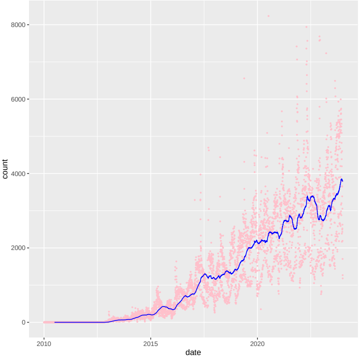
$$(\hat{\beta}_0, \hat{\beta}) = \underset{\beta_0, \beta}{\text{argmin}} \quad \frac{1}{2n} \|y - \beta_0 - \mathbf{X}\beta\|_2^2 + \lambda \left[\alpha \|\beta\|_1 + \frac{1-\alpha}{2} \|\beta\|_2^2 \right].$$

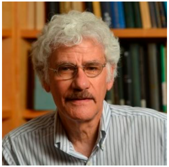
- $\lambda \geq 0$, $\alpha \in [0, 1]$ are hyperparameters
  - $\alpha = 0$: Ridge regression
  - $\alpha = 1$: Lasso
- Generalization for observation weights and relative penalty factors: Minimize

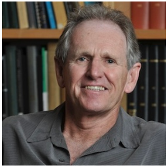
$$\frac{1}{2} \sum_{i=1}^{n} w_i \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \gamma_j \left(\alpha |\beta_j| + \frac{1-\alpha}{2} \beta_j^2 \right). \quad (*)$$

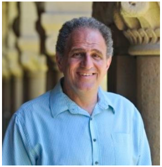
glmnet: An efficient elastic net solver

- Friedman et al. (2010): Can be solved quickly via *coordinate descent* for an entire path of λ values
- Intial package release in 2008
- Most of the computation done in FORTRAN
- glmnet package downloaded almost 6.5 million times in the last decade
- Website: https://glmnet.stanford.edu









glmnet: An efficient elastic net solver

$$J(\beta_0, \beta) = \frac{1}{2} \sum_{i=1}^{n} w_i \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \gamma_j \left(\alpha |\beta_j| + \frac{1-\alpha}{2} \beta_j^2 \right).$$

Coordinate descent to minimize J

Loop until convergence:

- For $j = 1, \ldots, p$:
  - For $i = 1, \ldots, n$, compute *partial residuals* $r_{ij} = y_i - \beta_0 - \sum_{k \neq j} \beta_k X_{ik}$.
  - Update $\beta_j \leftarrow \dfrac{S_{\lambda \gamma_j \alpha} \left(\sum_{i=1}^{n} w_i r_{ij} x_{ij} \right)}{\left(\sum_{i=1}^{n} w_i x_{ij}^2 \right) + \lambda \gamma_j (1 - \alpha)}$, where S is the soft-thresholding operator.
- For $i = 1, \ldots, n$, compute *partial residuals* $r_i = y_i - \sum_{j=1}^{p} \beta_j X_{ij}$.
- Update $\beta_0 \leftarrow \dfrac{\sum_{i=1}^{n} w_i r_i}{\sum_{i=1}^{n} w_i}$.

glmnet function

```
glmnet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "multinomial",
    "cox", "mgaussian"),
  weights = NULL,
  offset = NULL,
  alpha = 1,
  lambda = NULL,
  dfmax = nvars + 1,
  pmax = min(dfmax * 2 + 20, nvars),
  standardize = TRUE,
  intercept = TRUE,
  exclude = NULL,
  penalty.factor = rep(1, nvars),
  lower.limits = -Inf,
  upper.limits = Inf,
  ...
)
```

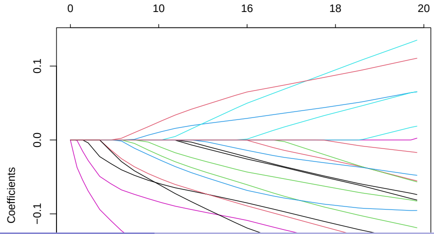
Commonly used with `cv.glmnet` and `predict` method.

cv.glmnet function

```r
cv.glmnet(
  x,
  y,
  weights = NULL,
  offset = NULL,
  lambda = NULL,
  type.measure = c("default", "mse", "deviance",
                   "class", "auc", "mae", "C"),
  nfolds = 10,
  foldid = NULL,
  alignment = c("lambda", "fraction"),
  grouped = TRUE,
  keep = FALSE,
  parallel = FALSE,
  gamma = c(0, 0.25, 0.5, 0.75, 1),
  relax = FALSE,
  trace.it = 0,
  ...
)
```

Example

```
> x <- matrix(rnorm(100 * 20),
                  100, 20)
> y <- rnorm(100)
> fit1 <- glmnet(x, y)
> print(fit1)
   Df  %Dev   Lambda
1   0  0.00 0.202800
2   1  0.59 0.184800
3   2  1.11 0.168400
4   2  1.96 0.153400
...
61 20 17.37 0.000764
62 20 17.37 0.000696
63 20 17.37 0.000634
64 20 17.37 0.000578
# Coeffs for one lambda
> coef(fit1, s = 0.01)
```



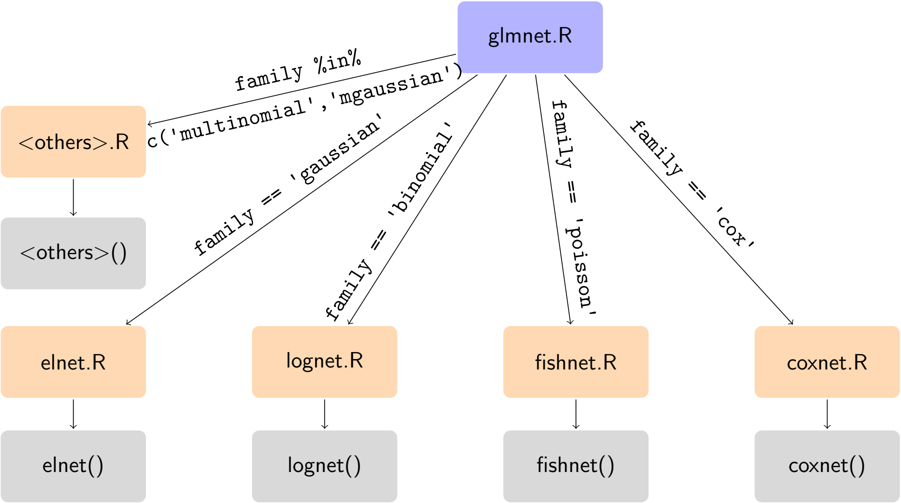
glmnet's pre-v4.0 family option

family option for original glmnet():

- "gaussian"
- "binomial"
- "poisson"
- "multinomial"
- "mgaussian"
- "cox"

First 3 are specific instances of generalized linear models (GLM).

glmnet's family parameter pre-v4.0



Every box here fits the model for a whole path of λ values.

** Grey boxes: FORTRAN subroutines, MORTRAN source.*

Extending elastic net to all GLMs!

Instead of minimizing

$$J(\beta_0, \beta) = \frac{1}{2} \sum_{i=1}^{n} w_i \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \gamma_j \left(\alpha |\beta_j| + \frac{1-\alpha}{2} \beta_j^2 \right),$$

minimize

$$J_{GLM}(\beta_0, \beta) = \sum_{i=1}^{n} w_i NLL_i + \lambda \sum_{j=1}^{p} \gamma_j \left(\alpha |\beta_j| + \frac{1-\alpha}{2} \beta_j^2 \right),$$

where NLL_i is the negative log-likelihood associated with observation i.

Kenneth Tay implemented this for *all* GLM families.

Easy Part

Same algorithm for all GLMs! Iteratively reweighted least squares (IRLS):

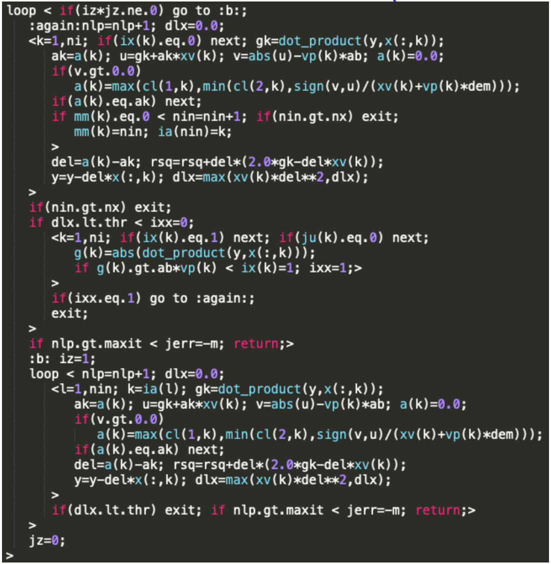
- Outer loop: quadratic approximation of the *NLL* terms
- Inner loop: coordinate descent as before!

Outer loop: compute working response $z_1, \ldots, z_n$ and working weights $v_1, \ldots, v_n$.

Inner loop: minimize

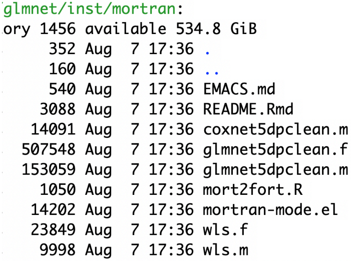
$$\frac{1}{2}\sum_{i=1}^{n} v_i \left(z_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \gamma_j \left(\alpha|\beta_j| + \frac{1-\alpha}{2}\beta_j^2 \right) \quad (*)$$

Hard Parts: FORTRAN (and MORTRAN)



```
loop < if(iz*jz.ne.0) go to :b:;
   :again:nlp=nlp+1; dlx=0.0;
   <k=1,ni; if(ix(k).eq.0) next; gk=dot_product(y,x(:,k));
      ak=a(k); u=gk+ak*xv(k); v=abs(u)-vp(k)*ab; a(k)=0.0;
      if(v.gt.0.0)
         a(k)=max(cl(1,k),min(cl(2,k),sign(v,u)/(xv(k)+vp(k)*dem)));
      if(a(k).eq.ak) next;
      if mm(k).eq.0 < nin=nin+1; if(nin.gt.nx) exit;
         mm(k)=nin; ia(nin)=k;
      >
      del=a(k)-ak; rsq=rsq+del*(2.0*gk-del*xv(k));
      y=y-del*x(:,k); dlx=max(xv(k)*del**2,dlx);
   >
   if(nin.gt.nx) exit;
   if dlx.lt.thr < ixx=0;
      <k=1,ni; if(ix(k).eq.1) next; if(ju(k).eq.0) next;
         g(k)=abs(dot_product(y,x(:,k)));
         if g(k).gt.ab*vp(k) < ix(k)=1; ixx=1;>
      >
      if(ixx.eq.1) go to :again:;
      exit;
   >
   if nlp.gt.maxit < jerr=-m; return;>
   :b: iz=1;
   loop < nlp=nlp+1; dlx=0.0;
      <l=1,nin; k=ia(l); gk=dot_product(y,x(:,k));
         ak=a(k); u=gk+ak*xv(k); v=abs(u)-vp(k)*ab; a(k)=0.0;
         if(v.gt.0.0)
            a(k)=max(cl(1,k),min(cl(2,k),sign(v,u)/(xv(k)+vp(k)*dem)));
         if(a(k).eq.ak) next;
         del=a(k)-ak; rsq=rsq+del*(2.0*gk-del*xv(k));
         y=y-del*x(:,k); dlx=max(xv(k)*del**2,dlx);
      >
      if(dlx.lt.thr) exit; if nlp.gt.maxit < jerr=-m; return;>
   >
   jz=0;
>
```

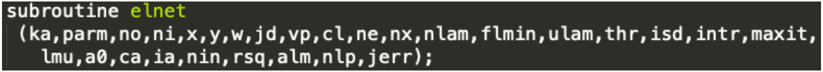
Coordinate descent loop in `elnet2` *subroutine*



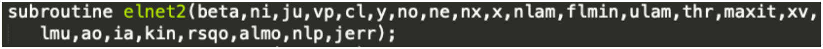
```
glmnet/inst/mortran:
ory 1456 available 534.8 GiB
     352 Aug  7 17:36 .
     160 Aug  7 17:36 ..
     540 Aug  7 17:36 EMACS.md
    3088 Aug  7 17:36 README.Rmd
   14091 Aug  7 17:36 coxnet5dpclean.m
  507548 Aug  7 17:36 glmnet5dpclean.f
  153059 Aug  7 17:36 glmnet5dpclean.m
    1050 Aug  7 17:36 mort2fort.R
   14202 Aug  7 17:36 mortran-mode.el
   23849 Aug  7 17:36 wls.f
    9998 Aug  7 17:36 wls.m
```

mortran subdirectory in `glmnet` *package*

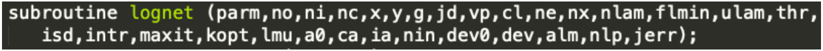
Hard Parts: Variable names



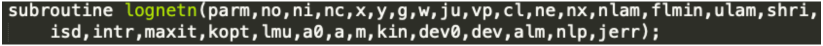
```
subroutine elnet
 (ka,parm,no,ni,x,y,w,jd,vp,cl,ne,nx,nlam,flmin,ulam,thr,isd,intr,maxit,
   lmu,a0,ca,ia,nin,rsq,alm,nlp,jerr);
```



```
subroutine elnet2(beta,ni,ju,vp,cl,y,no,ne,nx,x,nlam,flmin,ulam,thr,maxit,xv,
   lmu,ao,ia,kin,rsqo,almo,nlp,jerr);
```

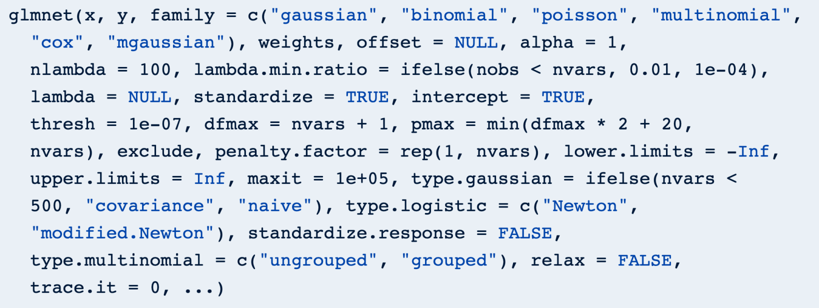


```
subroutine lognet (parm,no,ni,nc,x,y,g,jd,vp,cl,ne,nx,nlam,flmin,ulam,thr,
   isd,intr,maxit,kopt,lmu,a0,ca,ia,nin,dev0,dev,alm,nlp,jerr);
```



```
subroutine lognetn(parm,no,ni,nc,x,y,g,w,ju,vp,cl,ne,nx,nlam,flmin,ulam,shri,
    isd,intr,maxit,kopt,lmu,a0,a,m,kin,dev0,dev,alm,nlp,jerr);
```

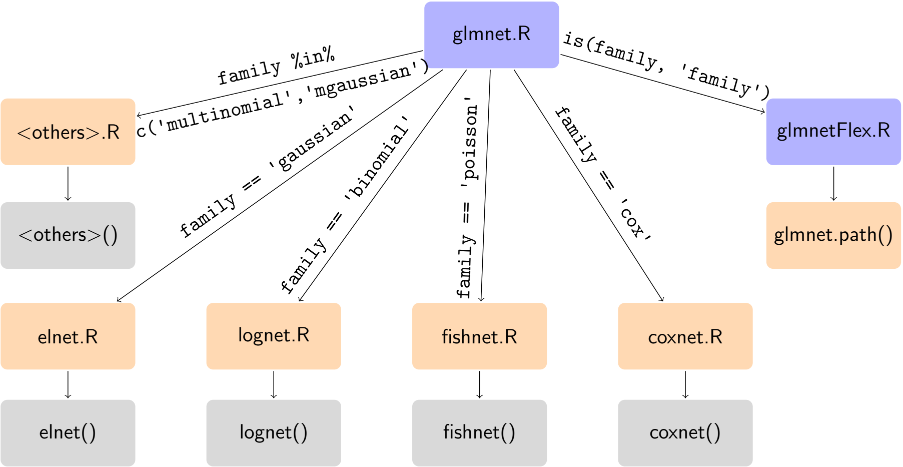
Hard Parts: Function options



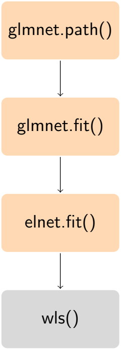
```
glmnet(x, y, family = c("gaussian", "binomial", "poisson", "multinomial",
  "cox", "mgaussian"), weights, offset = NULL, alpha = 1,
  nlambda = 100, lambda.min.ratio = ifelse(nobs < nvars, 0.01, 1e-04),
  lambda = NULL, standardize = TRUE, intercept = TRUE,
  thresh = 1e-07, dfmax = nvars + 1, pmax = min(dfmax * 2 + 20,
  nvars), exclude, penalty.factor = rep(1, nvars), lower.limits = -Inf,
  upper.limits = Inf, maxit = 1e+05, type.gaussian = ifelse(nvars <
  500, "covariance", "naive"), type.logistic = c("Newton",
  "modified.Newton"), standardize.response = FALSE,
  type.multinomial = c("ungrouped", "grouped"), relax = FALSE,
  trace.it = 0, ...)
```

glmnet()'s function signature pre-v4.0

glmnet's family parameter v4.0



glmnet's v4.0: internals



glmnet.path()

glmnet.fit()

elnet.fit()

wls()

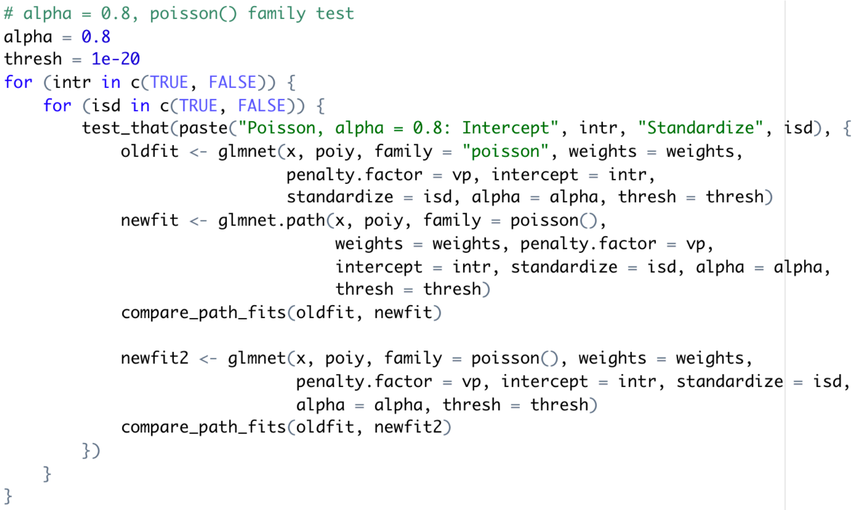
- Fits GLM elastic net model for whole path of λ values

- Fits GLM elastic net model for single λ value
- Computes working response and weights, then passes to `elnet.fit()`
- Can accept warm starts

- Solves the problem: *Minimize* $(*)$
- Essentially a wrapper for FORTRAN subroutine
- Can accept warm starts

Checks: matching pre-v4.0 results

family = gaussian() (binomial() and poisson() resp.) should match family = "gaussian" ("binomial" and "poisson" resp.).

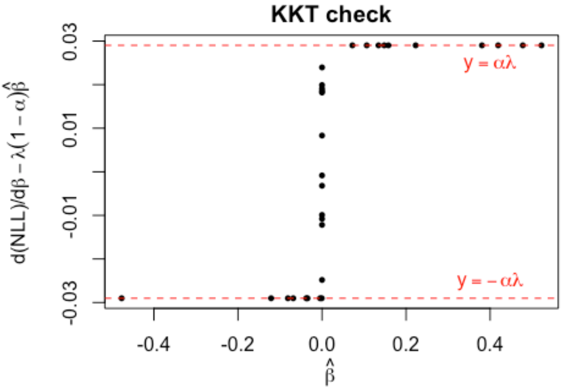


```
# alpha = 0.8, poisson() family test
alpha = 0.8
thresh = 1e-20
for (intr in c(TRUE, FALSE)) {
    for (isd in c(TRUE, FALSE)) {
        test_that(paste("Poisson, alpha = 0.8: Intercept", intr, "Standardize", isd), {
            oldfit <- glmnet(x, poiy, family = "poisson", weights = weights,
                             penalty.factor = vp, intercept = intr,
                             standardize = isd, alpha = alpha, thresh = thresh)
            newfit <- glmnet.path(x, poiy, family = poisson(),
                                  weights = weights, penalty.factor = vp,
                                  intercept = intr, standardize = isd, alpha = alpha,
                                  thresh = thresh)
            compare_path_fits(oldfit, newfit)

            newfit2 <- glmnet(x, poiy, family = poisson(), weights = weights,
                              penalty.factor = vp, intercept = intr, standardize = isd,
                              alpha = alpha, thresh = thresh)
            compare_path_fits(oldfit, newfit2)
        })
    }
}
```

Checks: KKT conditions

At the solution $\hat{\beta}$, the KKT conditions should be satisfied:

$$X_j^T \frac{\partial \ell}{\partial \beta} - \lambda(1-\alpha)\hat{\beta}_j \begin{cases} = \lambda\alpha & \text{if } \hat{\beta}_j > 0, \\ = -\lambda\alpha & \text{if } \hat{\beta}_j < 0, \\ \in [-\lambda\alpha, \lambda\alpha] & \text{if } \hat{\beta}_j = 0. \end{cases}$$



glmnet v4.0 examples

Standard gaussian family:

```
glmnet(x, y, family = gaussian())
```

Binomial regression with probit/complementary log-log link:

```
glmnet(x, y, family = binomial(link = "probit"))
glmnet(x, y, family = binomial(link = "cloglog"))
```

Negative binomial regression:

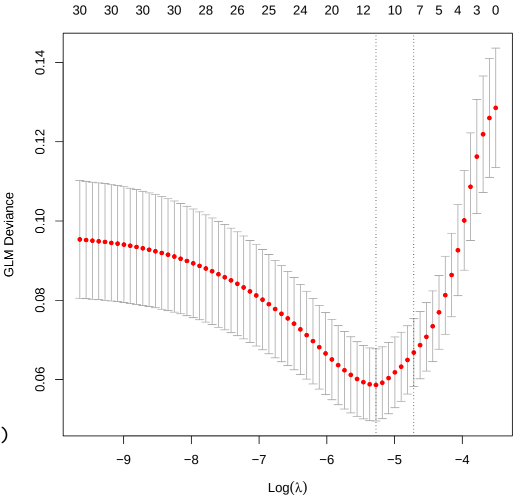
```
glmnet(x, y, family = MASS::negative.binomial(theta = 5))
```

Relaxed Fits: See vignette. New parameters:

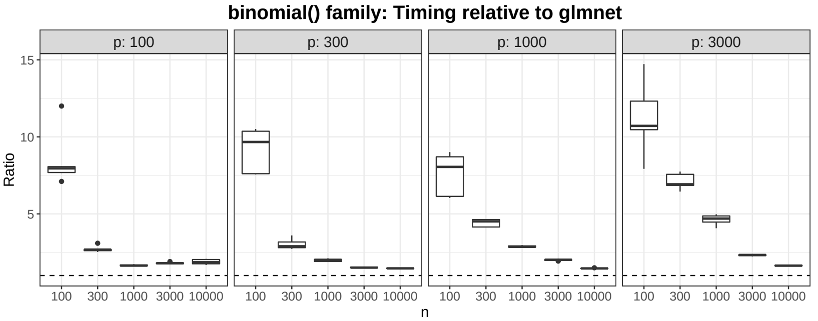
- `itrace = 0` for tracing, default none
- `epsnr = 1e-6` convergence threshold for `glmnet.fit`
- `mxitnr = 25` maximum no. of iterations for `glmnet.fit`.

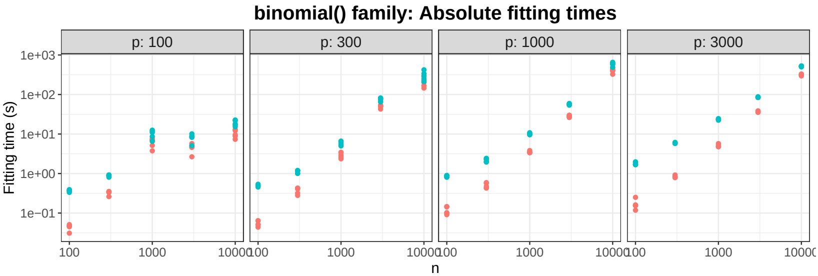
And also...

```
library(glmnet)
library(statmod)
set.seed(2212)
n <- 100; p <- 30;
x <- matrix(abs(rnorm(n * p)),
            nrow = n)
beta <- abs(rnorm(p))
# 20 betas are zero
beta[sample(p, 20)] <- 0
y <- x %*% beta + rnorm(n)
# Tweedie family w/ identity link
fam <- tweedie(var.power = 2,
               link.power = 1)
g <- glmnet(x, y, family = fam)
cg <- cv.glmnet(x, y, family = fam)
plot(cg) # ~ 10 preds at min
```



Computational comparison: `binomial()` family





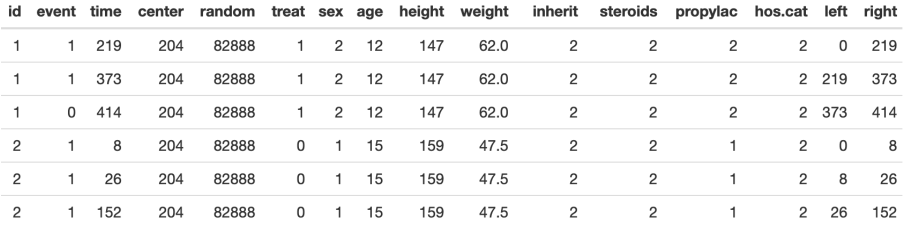
Survival Models (`family = "cox"`)

`glmnet` has always accommodated right-censored survival data (Simon et. al. JSS).
Kenneth Tay extended this to $(start, stop]$ data and strata.
Illustration using the data `survival::cgd0` (see help).

```
 1 204 082888 1 2 12 147.0  62.0 2 2 2 2 414 219 373
 2 204 082888 0 1 15 159.0  47.5 2 2 1 2 439   8  26 152 241 249 322 350
 3 204 082988 1 1 19 171.0  72.7 1 2 1 2 382
 4 204 091388 1 1 12 142.0  34.0 1 2 1 2 388
 5 238 092888 0 1 17 162.5  52.7 1 2 1 1 383 246 253
 6 245 093088 1 2 44 153.3  45.0 2 2 2 2 364
 7 245 093088 0 1 22 175.0  59.7 1 2 1 2 364 292
 8 245 093088 1 1  7 111.0  17.4 1 2 1 2 363
 9 238 100488 0 1 27 176.0  82.8 2 2 1 1 349 294
10 238 100488 1 1  5 113.0  19.5 1 2 1 1 371
```

The data has to be reformatted to a $(start, stop]$ long form for survival analysis. (A `foldid` column specifying subject fold-membership ensures subject records are together during cross-validation.)

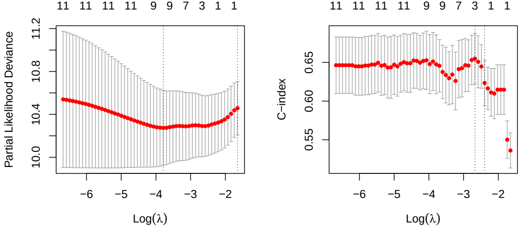
Counting Process Format



id	event	time	center	random	treat	sex	age	height	weight	inherit	steroids	propylac	hos.cat	left	right
1	1	219	204	82888	1	2	12	147	62.0	2	2	2	2	0	219
1	1	373	204	82888	1	2	12	147	62.0	2	2	2	2	219	373
1	0	414	204	82888	1	2	12	147	62.0	2	2	2	2	373	414
2	1	8	204	82888	0	1	15	159	47.5	2	2	1	2	0	8
2	1	26	204	82888	0	1	15	159	47.5	2	2	1	2	8	26
2	1	152	204	82888	0	1	15	159	47.5	2	2	1	2	26	152

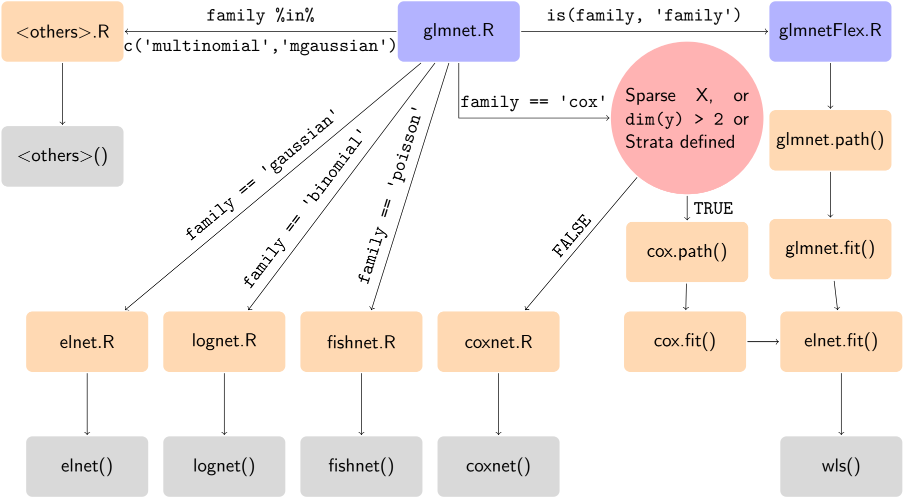
```
## Recode the binary vars coded 1 and 2
cgd$sex <- cgd$sex - 1
cgd$inherit <- cgd$inherit - 1
cgd$steroids <- cgd$steroids - 1
cgd$propylac <- cgd$propylac - 1
## hosp.cat has 4 categories
hosp.cat <- model.matrix(~factor(cgd$hos.cat))
colnames(hosp.cat) <- c("hoscat1", "hoscat2", "hoscat3", "hoscat4")

## Create x and y
x  <- as.matrix(cbind(cgd[, c("age", "height", "weight", "treat",
               "sex", "inherit", "steroids", "propylac")], hosp.cat))
y  <- Surv(cgd$tstart, cgd$tstop, cgd$infect)
```

```
## Increase iter for convergence
glmnet.control(mxitnr = 1000)
set.seed(1920)
g  <- glmnet(x = x, y = y,
family = "cox")
## Use foldid!
cg1  <- cv.glmnet(
          x = x,
          y = y,
          family = "cox",
          foldid = foldid)
cg2  <- cv.glmnet(
          x = x,
          y = y,
          family = "cox",
          type.measure = "C",
          foldid = foldid)
opar  <- par(mfrow = c(1, 2))
plot(cg1)
plot(cg2)
par(opar)
```

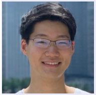


Var	cg1 λ_{1se}	cg2 λ_{1se}
age	.	-0.00211
height	.	.
weight	.	.
treat	.	-0.54316
sex	.	.
inherit	.	.
steroids	.	.
propylac	.	.
hoscat1	.	.
hoscat2	.	.
hoscat3	.	.
hoscat4	.	.

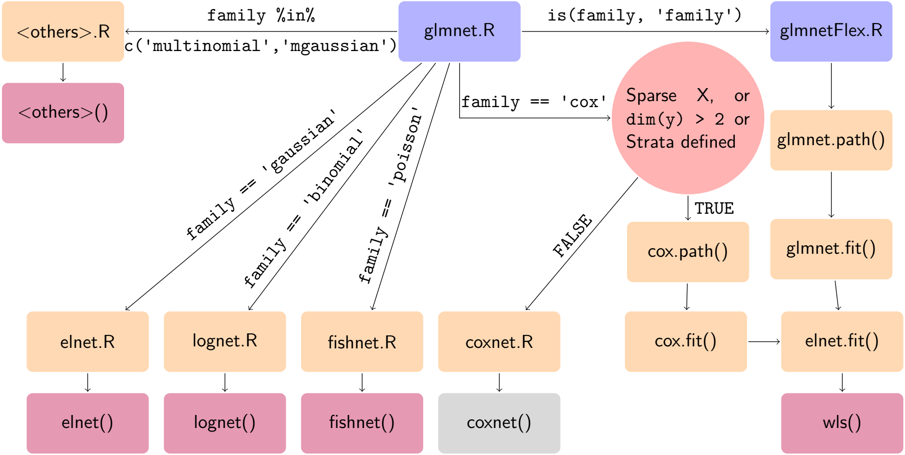
glmnet v4.0.1 (Still Fortran)



Fortran to C++



James Yang converted almost all Fortran to C++.



SNPnet: Lasso and elastic net for GWAS

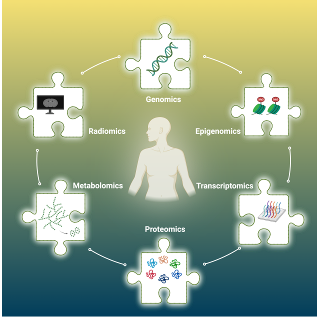
Estimation of Polygenic risk scores

- Lasso and elastic net cannot be applied to data of size say 500K (patients) by 800K (SNPs) using stock `glmnet`
- Tibshirani and colleagues developed a new approach using the idea of strong screening rules (Tibshrani et. al. JRSSB 2012) that enables such large scale computation by efficiently screening predictors that might be active.
- SNPnet is a version of `glmnet` tuned to such genomic data and successfully carries out this computation in hours.

See A Fast and Flexible Algorithm for Solving the Lasso in Large-scale and Ultrahigh-dimensional Problems by Qian et. al.

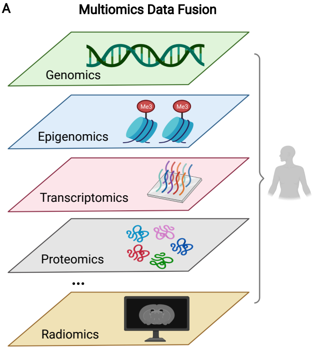
Code is on Github.

Multiomic Analysis



- Increasingly common in biology and medicine to have multi-modal data (Views) on patients.
- These are data such as genomics, proteomics, imaging features, etc. on a common set of samples
- Analysing such data represents an important challenge

Multi-view Analysis

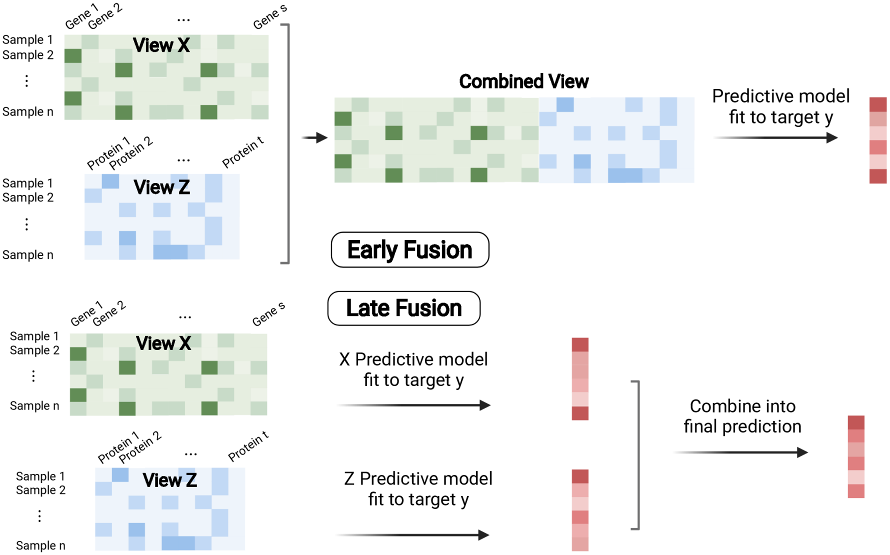


The goal is to utilize the different views on the same set of observations to model an outcome of interest.

Why?

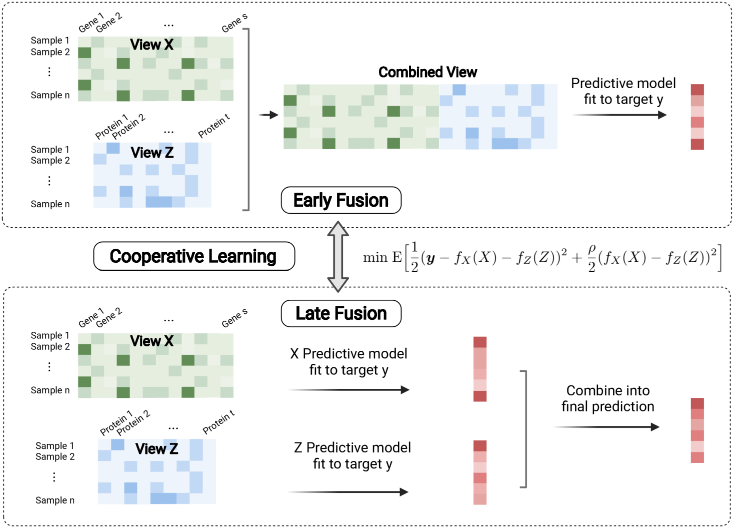
- Gain a more holistic understanding of the outcome
- Potential to make discoveries that are hidden in a single modality
- More accurate predictions

Existing Approaches: Early and Late Fusion



Cooperative Learning

Encompasses early and late fusion.



$$\min \mathrm{E}\Big[\frac{1}{2}(\boldsymbol{y} - f_X(X) - f_Z(Z))^2 + \frac{\rho}{2}(f_X(X) - f_Z(Z))^2\Big]$$

Cooperative Regularized Linear Regression (CRLR)

Daisy Ding et. al. PNAS 2022

Consider two views X and Z. CRLR combines the lasso penalty λ with the agreement penalty ρ, minimizing

$$\min \frac{1}{2} \|\mathbf{y} - X\theta_x - Z\theta_z\|^2 + \frac{\rho}{2} \|(X\theta_x - Z\theta_z)\|^2 + \lambda \left(\|\theta_x\|_1 + \|\theta_z\|_1\right)$$

Powerful when

- Different data views share some underlying relationship that can be leveraged to strengthen signal
- Each data view also has its idiosyncratic noise that needs to be reduced.

Lasso Formulation of CRLR

The solution to the convex problem can be computed as follows. Construct

$$\tilde{X} = \begin{pmatrix} X & Z \\ -\sqrt{\rho} X & \sqrt{\rho} Z \end{pmatrix}, \tilde{\boldsymbol{y}} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix}, \tilde{\boldsymbol{\beta}} = \begin{pmatrix} \boldsymbol{\theta}_x \\ \boldsymbol{\theta}_z \end{pmatrix}.$$

The equivalent objective now is

$$\frac{1}{2} || \tilde{\boldsymbol{y}} - \tilde{X} \tilde{\boldsymbol{\beta}} ||^2 + \lambda (|| \boldsymbol{\theta}_x ||_1 + || \boldsymbol{\theta}_z ||_1),$$

a lasso problem with $2n$ observations and $p_x + p_z$ features.

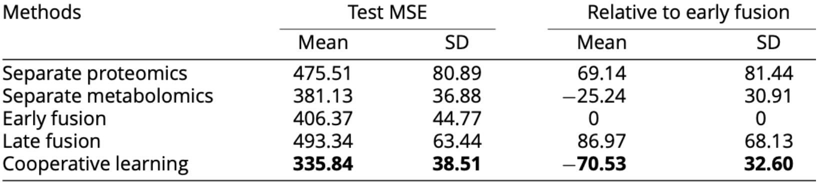
Let $\mathrm{Lasso}(X, \boldsymbol{y}, \lambda)$ denote the generic problem:

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \| \boldsymbol{y} - X \boldsymbol{\beta} \|^2 + \lambda \| \boldsymbol{\beta} \|_1 .$$

Direct Algorithm for CRLR

```
Input: $X \in \mathcal{R}^{n\times p_x}$ and $Z \in \mathcal{R}^{n\times p_z}$, the response $\vec{y} \in \mathcal{R}^n$, and a grid of
        hyperparameter values $(\rho_{\mathtt{min}}, \ldots, \rho_{\mathtt{max}})$.
for $\rho \leftarrow \rho_{\mathtt{min}}, \ldots, \rho_{\mathtt{max}}$ do
    Set
        $$\tilde{X} = \begin{pmatrix} X & Z \\ -\sqrt{\rho}X & \sqrt{\rho}Z \end{pmatrix}, \tilde{\boldsymbol{y}} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix}.$$
    Solve $\mathrm{Lasso}(\tilde{X}, \tilde{\boldsymbol{y}}, \lambda)$ over a decreasing grid of $\lambda$ values using glmnet.
end
Select the optimal value of $\rho^*$ based on the CV error and get the final fit.
```

Results: Multiomics study on labor onset prediction



Methods	Test MSE		Relative to early fusion	
	Mean	SD	Mean	SD
Separate proteomics	475.51	80.89	69.14	81.44
Separate metabolomics	381.13	36.88	−25.24	30.91
Early fusion	406.37	44.77	0	0
Late fusion	493.34	63.44	86.97	68.13
Cooperative learning	**335.84**	**38.51**	**−70.53**	**32.60**

For more details and software (package `multiview`), see the Cooperative Learning page

Underlying computational engine is `glmnet`!

Summary

- `glmnet` can perform penalized regression for any GLM family through the `family` parameter
- Cox survival models can be fit to $(start, stop]$ data, allowing for left-truncated data, time-varying covariates and strata
- Specialized version for handling large GWAS data (`SNPnet`)
- Applications of `glmnet`: Data Shared Lasso (Gross & Tibshirani), Cooperative Learning (Ding et. al., CRAN package `multiview`)

Coming Soon

Jonathan Taylor comes on board

- Python version closely matching R version
- Replace last bit of Fortran with `C++`
- Same `C++` code base for both Python and R
- Possibly compile `glmnet` for Web assembly.

glmnet.stanford.edu

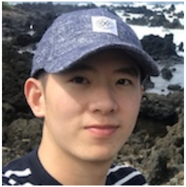
Thank You

Jerome Friedman

Trevor Hastie

Robert Tibshirani

Noah Simon



Junyang Qian

Kenneth Tay