TGDR: An Introduction

Julian Wolfson

Vaccine Efficacy

March 7, 2006

A motivating example

- In class, we discussed and analysed the VaxGen data
- Data not provided includes sequences of gp120 envelope protein of infecting virus for each infected subject
- Would like to link these sequences (mutations, insertions, deletions) with outcomes (eg. survival, viral load, etc.)
- Could also imagine having a panel of immunological assay outcomes (or some other high-dimensional covariate) for each subject

High dimensionality

gp120 protein sequence has 581 sites, 21 possible AAs per site = 12,201 covariates under typical coding:

Sequence ADF...

```
A C D E F G H I K L M N P Q R S T V W Y _
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
...
...
```

So what's the problem?

- Most regression approaches break down with this many covariates, particularly Cox regression, which typically fails with even modestly large numbers of covariates
- This is bad, since we have time-to-event data available and would like to use it

The Challenge

How do we find the small number of relevant needles in the covariate haystack?

So what's the problem?

- Most regression approaches break down with this many covariates, particularly Cox regression, which typically fails with even modestly large numbers of covariates
- This is bad, since we have time-to-event data available and would like to use it

The Challenge

How do we find the small number of relevant needles in the covariate haystack?

So what's the problem?

- Most regression approaches break down with this many covariates, particularly Cox regression, which typically fails with even modestly large numbers of covariates
- This is bad, since we have time-to-event data available and would like to use it

The Challenge

How do we find the small number of relevant needles in the covariate haystack?

The Idea

- When we have a number of covariates much larger than the sample size, we need to **regularize** (i.e. put restrictions on) our coefficient estimates
- One way to do this is to introduce a **penalty function** $P(\vec{\beta})$ and new parameter λ to the expression we minimise to obtain our coefficient estimates
- For linear regression, we have

$$\hat{\beta}(\lambda) = \min_{\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2 + \lambda P(\vec{\beta})$$

Two common penalties are $\sum \beta_i^2$ (ridge regression) and $\sum |\beta_i|$ (LASSO). We usually exclude the intercept parameter β_0.

The Idea

- When we have a number of covariates much larger than the sample size, we need to **regularize** (i.e. put restrictions on) our coefficient estimates
- One way to do this is to introduce a **penalty function** $P(\vec{\beta})$ and new parameter λ to the expression we minimise to obtain our coefficient estimates
- For linear regression, we have

$$\hat{\beta}(\lambda) = \min_{\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2 + \lambda P(\vec{\beta})$$

Two common penalties are $\sum \beta_i^2$ (ridge regression) and $\sum |\beta_i|$ (LASSO). We usually exclude the intercept parameter β_0.

The Idea

- When we have a number of covariates much larger than the sample size, we need to **regularize** (i.e. put restrictions on) our coefficient estimates
- One way to do this is to introduce a **penalty function** $P(\vec{\beta})$ and new parameter λ to the expression we minimise to obtain our coefficient estimates
- For linear regression, we have

$$\hat{\beta}(\lambda) = \min_{\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2 + \lambda P(\vec{\beta})$$

Two common penalties are $\sum \beta_i^2$ (ridge regression) and $\sum |\beta_i|$ (LASSO). We usually exclude the intercept parameter β_0.

The Idea

- When we have a number of covariates much larger than the sample size, we need to **regularize** (i.e. put restrictions on) our coefficient estimates
- One way to do this is to introduce a **penalty function** $P(\vec{\beta})$ and new parameter λ to the expression we minimise to obtain our coefficient estimates
- For linear regression, we have

$$\hat{\beta}(\lambda) = \min_{\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2 + \lambda P(\vec{\beta})$$

Two common penalties are $\sum \beta_i^2$ (ridge regression) and $\sum |\beta_i|$ (LASSO). We usually exclude the intercept parameter β_0.

The Idea

- When we have a number of covariates much larger than the sample size, we need to **regularize** (i.e. put restrictions on) our coefficient estimates
- One way to do this is to introduce a **penalty function** $P(\vec{\beta})$ and new parameter λ to the expression we minimise to obtain our coefficient estimates
- For linear regression, we have

$$\hat{\beta}(\lambda) = \min_{\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2 + \lambda P(\vec{\beta})$$

Two common penalties are $\sum \beta_i^2$ (ridge regression) and $\sum |\beta_i|$ (LASSO). We usually exclude the intercept parameter β_0.

$$\hat{\beta}(\lambda) = \min_{\vec{\beta}} \frac{1}{N} \sum (y_i - X_i \beta)^2 + \lambda P(\vec{\beta})$$

- The parameter λ controls how much the estimates are penalized
- It also indexes a one-dimensional path through the parameter space, and our goal is to find λ^* such that $\hat{\beta}(\lambda^*)$ is "closest" (often in terms of expected loss) to the true parameter vector $\vec{\beta}$.

So, we want to **R**egularize our coefficient estimates using the **T**hreshold λ... two letters down, two to go.

The GD of TGDR

In a 2004 paper, Friedman and Popescu propose a **G**radient **D**escent method for defining a parameter path:

1. Set $\nu = 0$
2. Start at a point in the parameter space $\hat{\beta}(\nu)$
3. "Descend" to the next point on the path via the update rule

$$\hat{\beta}(\nu + \Delta\nu) = \hat{\beta}(\nu) + \Delta\nu g(\nu)$$

   where $\Delta\nu$ is an increment and $g(\nu)$ is the gradient of the empirical risk (i.e. average loss). In the case of linear regression, we have

$$g(\nu) = -\frac{d}{d\vec{\beta}}\frac{1}{N}\sum (y_i - X_i\beta)^2$$

   evaluated at $\vec{\beta} = \hat{\beta}(\nu)$.

In other words, we update by stepping some small amount in the direction of greatest decrease in the loss function.

The GD of TGDR

In a 2004 paper, Friedman and Popescu propose a **G**radient **D**escent method for defining a parameter path:

1. Set $\nu = 0$
2. Start at a point in the parameter space $\hat{\beta}(\nu)$
3. "Descend" to the next point on the path via the update rule

$$\hat{\beta}(\nu + \Delta\nu) = \hat{\beta}(\nu) + \Delta\nu g(\nu)$$

where $\Delta\nu$ is an increment and $g(\nu)$ is the gradient of the empirical risk (i.e. average loss). In the case of linear regression, we have

$$g(\nu) = -\frac{d}{d\vec{\beta}}\frac{1}{N}\sum(y_i - X_i\beta)^2$$

evaluated at $\vec{\beta} = \hat{\beta}(\nu)$.

In other words, we update by stepping some small amount in the direction of greatest decrease in the loss function.

The GD of TGDR

In a 2004 paper, Friedman and Popescu propose a **G**radient **D**escent method for defining a parameter path:

1. Set $\nu = 0$
2. Start at a point in the parameter space $\hat{\beta}(\nu)$
3. "Descend" to the next point on the path via the update rule

$$\hat{\beta}(\nu + \Delta\nu) = \hat{\beta}(\nu) + \Delta\nu g(\nu)$$

   where $\Delta\nu$ is an increment and $g(\nu)$ is the gradient of the empirical risk (i.e. average loss). In the case of linear regression, we have

$$g(\nu) = -\frac{d}{d\vec{\beta}}\frac{1}{N}\sum(y_i - X_i\beta)^2$$

   evaluated at $\vec{\beta} = \hat{\beta}(\nu)$.

In other words, we update by stepping some small amount in the direction of greatest decrease in the loss function.

The GD of TGDR

In a 2004 paper, Friedman and Popescu propose a **G**radient **D**escent method for defining a parameter path:

1. Set $\nu = 0$
2. Start at a point in the parameter space $\hat{\beta}(\nu)$
3. "Descend" to the next point on the path via the update rule

$$\hat{\beta}(\nu + \Delta\nu) = \hat{\beta}(\nu) + \Delta\nu g(\nu)$$

where $\Delta\nu$ is an increment and $g(\nu)$ is the gradient of the empirical risk (i.e. average loss). In the case of linear regression, we have

$$g(\nu) = -\frac{d}{d\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2$$

evaluated at $\vec{\beta} = \hat{\beta}(\nu)$.

In other words, we update by stepping some small amount in the direction of greatest decrease in the loss function.

The GD of TGDR

In a 2004 paper, Friedman and Popescu propose a **G**radient **D**escent method for defining a parameter path:

1. Set $\nu = 0$
2. Start at a point in the parameter space $\hat{\beta}(\nu)$
3. "Descend" to the next point on the path via the update rule

$$\hat{\beta}(\nu + \Delta\nu) = \hat{\beta}(\nu) + \Delta\nu g(\nu)$$

where $\Delta\nu$ is an increment and $g(\nu)$ is the gradient of the empirical risk (i.e. average loss). In the case of linear regression, we have

$$g(\nu) = -\frac{d}{d\vec{\beta}} \frac{1}{N} \sum (y_i - X_i\beta)^2$$

evaluated at $\vec{\beta} = \hat{\beta}(\nu)$.

In other words, we update by stepping some small amount in the direction of greatest decrease in the loss function.

- Ma & Huang (2005) and Gui & Li (2005) extended this technique to Accelerated Failure Times and Proportional Hazards (Cox regression) models
- Algorithm sketch (for a given step length $\Delta\nu$ and threshold parameter $0 < \tau < 1$):
  1. Start with an estimate $\hat{\beta}_k$
  2. Compute the gradient g_k of the likelihood (or partial likelihood) w.r.t. $\vec{\beta}$ evaluated at $\hat{\beta}_k$
  3. Let $\hat{\beta}_{k+1} = \hat{\beta}_k + \Delta\nu f_k g_k$, where $f_k = 1[abs(g_k) >= \tau max(abs(g_k))]$
  4. Repeat

- This algorithm creates a parameter path $\hat{\beta}_0, \hat{\beta}_1, \ldots$.
- Perform **cross-validation** to choose our "best guess" at $\hat{\beta}$ on the path (details omitted due to time constraints)
- Look at individual coefficients with largest values to get an idea of where the "needles" are

Application: VaxGen Data

Relevant Data

- Complete gp120 sequences of the infecting virus for each infected subject (we consider infected vaccinees only)
- Viral load at follow-up visits up to two years post-infection

Endpoint of Interest

(T, C), where

- T is the time until viral load surpasses 10,000 copies
- C is the censoring indicator

Question

Which positions/AAs (mutations, insertions, deletions) are associated with time until loss of immune control of viral replication (i.e. $>$ 10,000 copies)?

Application: VaxGen Data

Relevant Data

- Complete gp120 sequences of the infecting virus for each infected subject (we consider infected vaccinees only)
- Viral load at follow-up visits up to two years post-infection

Endpoint of Interest

(T, C), where

- T is the time until viral load surpasses 10,000 copies
- C is the censoring indicator

Question

Which positions/AAs (mutations, insertions, deletions) are associated with time until loss of immune control of viral replication (i.e. $>$ 10,000 copies)?

Application: VaxGen Data

Relevant Data

- Complete gp120 sequences of the infecting virus for each infected subject (we consider infected vaccinees only)
- Viral load at follow-up visits up to two years post-infection

Endpoint of Interest

(T, C), where

- T is the time until viral load surpasses 10,000 copies
- C is the censoring indicator

Question

Which positions/AAs (mutations, insertions, deletions) are associated with time until loss of immune control of viral replication (i.e. $>$ 10,000 copies)?

The strategy

1. Define (T, C) and format the sequences Go
2. Pre-process the sequences (in a somewhat ad-hoc way):
   - Eliminate all positions (covariates) which do not vary across individuals
3. Run TGDR to obtain a parameter path $\vec{\hat{\beta}}$
4. Perform cross-validation to choose our optimal $\hat{\beta}$
5. Look for "needles", i.e. potentially interesting patterns in $\hat{\beta}$

The strategy

1. Define (T, C) and format the sequences Go
2. Pre-process the sequences (in a somewhat ad-hoc way):
   - Eliminate all positions (covariates) which do not vary across individuals
3. Run TGDR to obtain a parameter path $\vec{\hat{\beta}}$
4. Perform cross-validation to choose our optimal $\hat{\beta}$
5. Look for “needles”, i.e. potentially interesting patterns in $\hat{\beta}$

The strategy

1. Define (T, C) and format the sequences ▸ Go
2. Pre-process the sequences (in a somewhat ad-hoc way):
   - Eliminate all positions (covariates) which do not vary across individuals
3. Run TGDR to obtain a parameter path $\vec{\hat{\beta}}$
4. Perform cross-validation to choose our optimal $\hat{\beta}$
5. Look for “needles”, i.e. potentially interesting patterns in $\hat{\beta}$

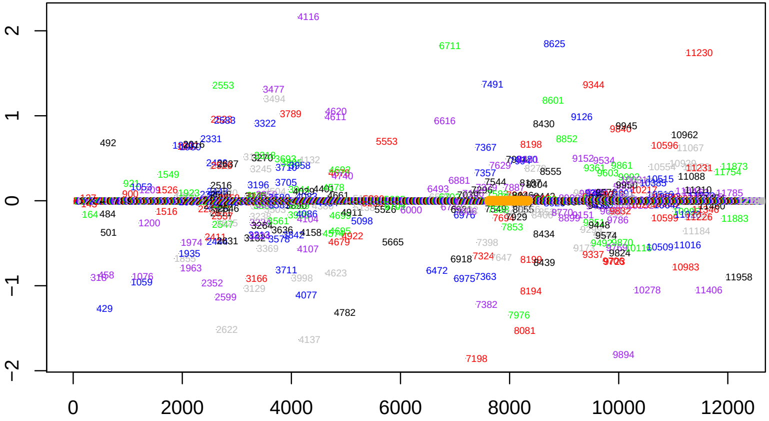
The strategy

1. Define (T, C) and format the sequences ▸ Go
2. Pre-process the sequences (in a somewhat ad-hoc way):
   - Eliminate all positions (covariates) which do not vary across individuals
3. Run TGDR to obtain a parameter path $\vec{\hat{\beta}}$
4. Perform cross-validation to choose our optimal $\hat{\beta}$
5. Look for “needles”, i.e. potentially interesting patterns in $\hat{\beta}$

The strategy

1. Define (T, C) and format the sequences ▸ Go
2. Pre-process the sequences (in a somewhat ad-hoc way):
   - Eliminate all positions (covariates) which do not vary across individuals
3. Run TGDR to obtain a parameter path $\vec{\hat{\beta}}$
4. Perform cross-validation to choose our optimal $\hat{\beta}$
5. Look for "needles", i.e. potentially interesting patterns in $\hat{\beta}$

Looking for needles



Some interesting needles...

... which may or may not be relevant:

- Position 320: Approx. half N (Asparagine), half D (Aspartic Acid). Coefficient for D = 0.05, for N = 1.83
- Position 411: Predominantly Q (Glutamine). Coefficient for Q = 0, for mutation R = 1.85
- Position 472: Predominantly N (Asparagine). Coefficient for N = 0.17, for mutation D = -1.81

Extensions

A number of extensions to this method are possible, and in various stages of development:

- Allowing for time-varying covariates
    - Code is written, but not debugged
- Incorporating missing data, interval censoring, time-varying coefficients (?)
- "Optimal" pre-processing of high-dimensional covariates
- And many others

Extensions

A number of extensions to this method are possible, and in various stages of development:

- Allowing for time-varying covariates
  - Code is written, but not debugged
- Incorporating missing data, interval censoring, time-varying coefficients (?)
- "Optimal" pre-processing of high-dimensional covariates
- And many others

Extensions

A number of extensions to this method are possible, and in various stages of development:

- Allowing for time-varying covariates
  - Code is written, but not debugged
- Incorporating missing data, interval censoring, time-varying coefficients (?)
- “Optimal” pre-processing of high-dimensional covariates
- And many others

Extensions

A number of extensions to this method are possible, and in various stages of development:

- Allowing for time-varying covariates
  - Code is written, but not debugged
- Incorporating missing data, interval censoring, time-varying coefficients (?)
- “Optimal” pre-processing of high-dimensional covariates
- And many others

Extensions

A number of extensions to this method are possible, and in various stages of development:

- Allowing for time-varying covariates
  - Code is written, but not debugged
- Incorporating missing data, interval censoring, time-varying coefficients (?)
- "Optimal" pre-processing of high-dimensional covariates
- And many others

A word about LaTeX and presentations

This presentation is a PDF file generated from a LaTeX (text) document, with the help of a package called `beamer`. More info available at

$$\texttt{http}://\texttt{latex}-\texttt{beamer.sourceforge.net}/$$

Ask me if you have any questions... but no guarantees.

Thanks!

Questions?