Lecture 18: Constrained & Penalized Optimization

36-350
27 October 2014

Agenda

- Optimization under constraints
- Lagrange multipliers
- Penalized optimization
- Statistical uses of penalized optimization

Maximizing a multinomial likelihood

I roll dice \(n\) times; \(n_1, \ldots n_6\) count the outcomes

Likelihood and log-likelihood:

\[
L(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \frac{n!}{n_1!n_2!n_3!n_4!n_5!n_6!} \prod_{i=1}^{6} \theta_i^{n_i}
\]

\[
\ell(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \log \frac{n!}{n_1!n_2!n_3!n_4!n_5!n_6!} + \sum_{i=1}^{6} n_i \log \theta_i
\]

Optimize by taking the derivative and setting to zero:

\[
\frac{\partial \ell}{\partial \theta_1} = \frac{n_1}{\theta_1} = 0
\]
\[
\therefore \theta_1 = \infty
\]

Maximizing a multinomial likelihood

We forgot that \(\sum_{i=1}^{6} \theta_i = 1\)

We could use the constraint to eliminate one of the variables

\[
\theta_6 = 1 - \sum_{i=1}^{5} \theta_i
\]

Then solve the equations

\[
\frac{\partial \ell}{\partial \theta_i} = \frac{n_1}{\theta_i} - \frac{n_6}{1 - \sum_{j=1}^{5} \theta_j} = 0
\]

BUT eliminating a variable with the constraint is usually messy
Lagrange Multipliers

\[ g(\theta) = c \iff g(\theta) - c = 0 \]

Lagrangian:

\[ \mathcal{L}(\theta, \lambda) = f(\theta) - \lambda(g(\theta) - c) \]

= \( f \) when the constraint is satisfied

Now do \textit{unconstrained} minimization over \( \theta \) and \( \lambda \):

\[ \nabla_\theta \mathcal{L}|_{\theta^*, \lambda^*} = \nabla f(\theta^*) - \lambda^* \nabla g(\theta^*) = 0 \]

\[ \frac{\partial \mathcal{L}}{\partial \lambda}|_{\theta^*, \lambda^*} = g(\theta^*) - c = 0 \]

optimizing \textbf{Lagrange multiplier} \( \lambda \) enforces constraint

More constraints, more multipliers

\begin{align*}
\mathcal{L} &= \log \prod_i n_i! + \sum_{i=1}^{6} n_i \log (\theta_i) - \lambda \left( \sum_{i=1}^{6} \theta_i - 1 \right) \\
\frac{\partial \mathcal{L}}{\partial \theta_i}|_{\theta_i = \theta_i^*} &= \frac{n_i}{\theta_i^*} - \lambda^* = 0 \\
\frac{n_i}{\lambda^*} &= \theta_i^* \\
\sum_{i=1}^{6} \frac{n_i}{\lambda^*} &= \sum_{i=1}^{6} \theta_i^* = 1 \\
\lambda^* &= \sum_{i=1}^{6} n_i \Rightarrow \theta_i^* = \frac{n_i}{\sum_{i=1}^{6} n_i}
\end{align*}

\textbf{Thinking About the Lagrange Multipliers}

Constrained minimum value is generally higher than the unconstrained

Changing the constraint level \( c \) changes \( \theta^* \), \( f(\theta^*) \)

\[ \frac{\partial f(\theta^*)}{\partial c} = \frac{\partial \mathcal{L}(\theta^*, \lambda^*)}{\partial c} \]

\[ = |\nabla f(\theta^*) - \lambda^* \nabla g(\theta^*)| \frac{\partial g(\theta^*)}{\partial c} - [g(\theta^*) - c] \frac{\partial \lambda^*}{\partial c} + \lambda^* = \lambda^* \]

\( \lambda^* \) = Rate of change in optimal value as the constraint is relaxed

\( \lambda^* \) = “Shadow price”: How much would you pay for minute change in the level of the constraint

\[ \begin{align*}
\lambda^* &= \frac{\partial f(\theta^*)}{\partial c} \\
\n\end{align*} \]
Inequality Constraints

What about an inequality constraint?

\[ h(\theta) \leq d \iff h(\theta) - d \leq 0 \]

The region where the constraint is satisfied is the feasible set

Roughly two cases:

1. Unconstrained optimum is inside the feasible set \( \Rightarrow \) constraint is inactive
2. Optimum is outside feasible set; constraint is active, binds or bites; constrained optimum is usually on the boundary

Add a Lagrange multiplier; \( \lambda \neq 0 \iff \) constraint binds

Mathematical Programming

Older than computer programming...

Optimize \( f(\theta) \) subject to \( g(\theta) = c \) and \( h(\theta) \leq d \)

“Give us the best deal on \( f \), keeping in mind that we’ve only got \( d \) to spend, and the books have to balance”

Linear programming (Kantorovich, 1938)

1. \( f, h \) both linear in \( \theta \)
2. \( \theta^* \) always at a corner of the feasible set

Back to the Factory

Revenue: 13k per car, 27k per truck

Constraints:

\[
\begin{align*}
40 \times \text{cars} + 60 \times \text{trucks} &< 1600 \text{hours} \\
1 \times \text{cars} + 3 \times \text{trucks} &< 70 \text{tons}
\end{align*}
\]

Find the revenue-maximizing number of cars and trucks to produce

Back to the Factory

The feasible region:
Back to the Factory

The feasible region, plus lines of equal profit

The Equivalent Capitalist Problem

… is that problem
the Walrasian model [of economics] is essentially about allocations and only tangentially about markets — as one of us (Bowles) learned when he noticed that the graduate microeconomics course that he taught at Harvard was easily repackaged as ‘The Theory of Economic Planning’ at the University of Havana in 1969. (S. Bowles and H. Gintis, “Walrasian Economics in Retrospect”, Quarterly Journal of Economics, 2000)

The Slightly More Complex Financial Problem

*Given:* expected returns $r_1, \ldots, r_p$ among $p$ financial assets, their $p \times p$ matrix of variances and covariances $\Sigma$

*Find:* the portfolio shares $\theta_1, \ldots, \theta_n$ which maximizes expected returns

*Such that:* total variance is below some limit, covariances with specific other stocks or portfolios are below some limit

e.g., pension fund should not be too correlated with parent company

Expected returns $f(\theta) = r \cdot \theta$

Constraints: $\sum_{i=1}^{p} \theta_i = 1$, $\theta_i \geq 0$ (unless you can short)

Covariance constraints are linear in $\theta$

Variance constraint is quadratic, over-all variance is $\theta^T \Sigma \theta$

Barrier Methods

(a.k.a. “interior point”, “central path”, etc.)

Having constraints switch on and off abruptly is annoying especially with gradient methods

Fix $\mu > 0$ and try minimizing $f(\theta) - \mu \log (d - h(\theta))$

“pushes away” from the barrier — more and more weakly as $\mu \to 0$

Barrier Methods

1. Initial $\theta$ in feasible set, initial $\mu$
2. While ((not too tired) and (making adequate progress))
   a. Minimize $f(\theta) - \mu \log (d - h(\theta))$
   b. Reduce $\mu$
3. Return final $\theta$

R implementation

`constrOptim` implements the barrier method

Try this:
factory <- matrix(c(40,1,60,3),nrow=2,
        dimnames=list(c("labor","steel"),c("car","truck")))
available <- c(1600,70); names(available) <- rownames(factory)
prices <- c(car=13, truck=27)
revenue <- function(output) { return(-output * prices) }
plan <- constrOptim(theta=c(5,5), f=revenue, grad=NULL,
               ui=-factory, ci=-available, method="Nelder-Mead")
plan$par

# [1] 10 20

constrOptim only works with constraints like $u \theta \geq c$, so minus signs

Constraints vs. Penalties

$$
\arg\min_{\theta : h(\theta) \leq d} f(\theta) \Leftrightarrow \arg\min_{\theta, \lambda} f(\theta) - \lambda(h(\theta) - d)
$$

$d$ doesn’t matter for doing the second minimization over $\theta$

We could just as well minimize

$$
f(\theta) - \lambda h(\theta)
$$

<table>
<thead>
<tr>
<th>Constrained optimization</th>
<th>Penalized optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint level $d$</td>
<td>Penalty factor $\lambda$</td>
</tr>
</tbody>
</table>

“A fine is a price”

Statistical Applications of Penalization

Mostly you’ve seen unpenalized estimates (least squares, maximum likelihood)

Lots of modern advanced methods rely on penalties

- For when the direct estimate is too unstable
- For handling high-dimensional cases
- For handling non-parametrics

Ordinary Least Squares

No penalization; minimize MSE of linear function $\beta \cdot x$:

$$
\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta \cdot x_i)^2 = \arg\min_{\beta} MSE(\beta)
$$
Closed-form solution if we can invert matrices:

\[
\hat{\beta} = (x^T x)^{-1} x^T y
\]

where \(x\) is the \(n \times p\) matrix of \(x\) vectors, and \(y\) is the \(n \times 1\) matrix of \(y\) values.

**Ridge Regression**

Now put a penalty on the magnitude of the coefficient vector:

\[
\hat{\beta} = \arg\min_\beta \text{MSE}(\beta) + \mu \sum_{j=1}^{p} \beta_j^2 = \arg\min_\beta \text{MSE}(\beta) + \mu \|\beta\|_2^2
\]

Penalizing \(\beta\) this way makes the estimate more stable; especially useful for:
- Lots of noise
- Collinear data (\(x\) not of “full rank”)
- High-dimensional, \(p > n\) data (which implies collinearity)

This is called **ridge regression**, or **Tikhonov regularization**

Closed form:

\[
\hat{\beta} = (x^T x + \mu I)^{-1} x^T y
\]

**The Lasso**

Put a penalty on the sum of coefficient’s absolute values:

\[
\hat{\beta}^\dagger = \arg\min_\beta \text{MSE}(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| = \arg\min_\beta \text{MSE}(\beta) + \lambda \|\beta\|_1
\]

This is called **the lasso**

- Also stabilizes (like ridge)
- Also handles high-dimensional data (like ridge)
- Enforces sparsity: it likes to drive small coefficients exactly to 0

No closed form, but very efficient interior-point algorithms (e.g., **lars** package)

**Spline Smoothing**

“Spline smoothing”: minimize MSE of a smooth, nonlinear function, plus a penalty on curvature:

\[
\hat{f} = \arg\min_f \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \int (f''(x))^2 dx
\]

This fits smooth regressions without assuming any specific functional form

- Lets you check linear models
- Makes you wonder why you bother with linear models

Many different R implementations, starting with **smooth.spline**
How Big a Penalty?

Rarely know the constraint level or the penalty factor $\lambda$ from on high
Lots of ways of picking, but often cross-validation works well:

- Divide the data into parts
- For each value of $\lambda$, estimate the model on one part of the data
- See how well the models fit the other part of the data
- Use the $\lambda$ which extrapolates best on average

Summary

- We use Lagrange multipliers to turn constrained optimization problems into unconstrained but penalized ones
  - Optimal multiplier values are the prices we’d pay to weaken the constraints
- The nature of the penalty term reflects the sort of constraint we put on the problem
  - Shrinkage
  - Sparsity
  - Smoothness

Hypothesis Testing

Test the hypothesis that the data are distributed $\sim P$ against the hypothesis that they are distributed $\sim Q$
$P = \text{noise}, Q = \text{signal}$
Want to maximize power, probability the test picks up the signal when it’s present
Need to limit false alarm rate, probability the test claims “signal” in noise

Hypothesis Testing

Say “signal” whenever the data falls into some set $S$
Power $= Q(S)$
False alarm rate $= P(S) \leq \alpha$

$$\max_{S : P(S) \leq \alpha} Q(S)$$

With Lagrange multiplier,

$$\max_{S, \lambda} Q(S) - \lambda(P(S) - \alpha)$$

Looks like we have to do ugly calculus over set functions...
Hypothesis Testing

Pick any point $x$: should we add it to $S$?

Marginal benefit $= \frac{dQ}{dx} = q(x)$

Marginal cost $= \lambda \frac{dP}{dx} = \lambda p(x)$

Keep expanding $S$ until marginal benefit = marginal cost so $q(x)/p(x) = \lambda$

$q(x)/p(x) =$ likelihood ratio; optimal test is Neyman-Pearson test

$\lambda =$ critical likelihood ratio = shadow price of power

Lasso Example

```r
x <- matrix(rnorm(200),nrow=100)
y <- (x %*% c(2,1))+ rnorm(100,sd=0.05)

mse <- function(b1,b2) {mean((y- x %*% c(b1,b2))^2)}
coef.seq <- seq(from=-1,to=5,length.out=200)
m <- outer(coef.seq,coef.seq,Vectorize(mse))
l1 <- function(b1,b2) {abs(b1)+abs(b2)}
l1.levels <- outer(coef.seq,coef.seq,l1)
ols.coefs <- coefficients(lm(y~0+x))

contour(x=coef.seq,y=coef.seq,z=m,drawlabels=FALSE,nlevels=30,col="grey",
         main="Contours of MSE vs. Contours of L1")
contour(x=coef.seq,y=coef.seq,z=l1.levels,nlevels=20,add=TRUE)
points(x=ols.coefs[1],y=ols.coefs[2],pch="+")
points(0,0)
```

Contours of MSE vs. Contours of L1
Lasso Example

```
contour(x=coef.seq,y=coef.seq,z=m+l1.levels,drawlabels=FALSE,nlevels=30,
main="Contours of MSE+L1")
```

Contours of MSE+L1

Augmented Lagrangian Methods

A simple trick for constrained optimization: minimize

\[ f(\theta) + r(g(\theta) - c)^2 \]

over and over, letting \( r \to \infty \)

Drawback: really unstable when \( r \) is huge

Augmented Lagrangian trick: fix \( r \) and a guess at \( \lambda \), then minimize

\[ f(\theta) + \lambda(g(\theta) - c) + r(g(\theta) - c)^2 \]

Now crank up \( r \) and update \( \lambda \) by an amount that reflects how badly the constraint was violated

Often converges at finite \( r \)

Augmented Lagrangian Methods

Same ideas work for inequality constraints

Unlike interior-point methods, initial guess needn’t be in feasible set

R implementation: alabama package