Logistic regression

Patrick Breheny

April 17, 2025

Introduction

- The general paradigm for penalized regression is

$$Q(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}) = L(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}) + P_\lambda(\boldsymbol{\beta})$$

- So far, we have discussed a variety of choices for $P_\lambda(\boldsymbol{\beta})$, but $L(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y})$ has always been the least squares loss function (i.e., linear regression)
- In our next two lectures, we'll explore some different types of loss functions in order to see how penalized regression methods extend to other types of data

Logistic regression

- One of the most common, perhaps even more common than linear regression itself, are studies in which the outcome is binary
- Such studies are particularly common in medical research, where it is common to consider presence/absence of a disease as the outcome
- In this setting, it is natural to model the outcome using a binomial distribution, allowing $\pi_i \equiv \mathbb{P}(Y_i)$ to depend on the features according to

$$\log \frac{\pi_i}{1-\pi_i} = \mathbf{x}_i^\top \boldsymbol{\beta};$$

this is known as the *logistic regression* model

Logistic regression objective function

- As with linear regression, maximum likelihood estimation of $\boldsymbol{\beta}$ will be problematic if the number of features is large, making penalized likelihood estimation desirable
- Thus, we will estimate $\boldsymbol{\beta}$ by minimizing the objective function

$$-\frac{1}{n}\sum_{i=1}^{n}\{y_i \log \pi_i + (1-y_i)\log(1-\pi_i)\} + P_\lambda(\boldsymbol{\beta}),$$

where $\mathbf{X}$ and $\boldsymbol{\beta}$ are included in the likelihood implicitly, as $\boldsymbol{\pi}$ is a function of $\mathbf{X}\boldsymbol{\beta}$

Similarities and differences

- For the most part, everything we have talked about so far this semester with respect to linear regression carries over to logistic regression: ridge, lasso, MCP, elastic net, etc., penalties have similar effects on regression coefficients as we have seen
- However, two differences are worth discussing:
  - We need new algorithms for model fitting
  - We need new measures of predictive accuracy
- Inference concerning $\boldsymbol{\beta}$, of course, would also be different, although this is beyond the scope of a single lecture; some approaches (sample splitting, knockoffs) are straightforward to extend while others (selective inference) are not

Logistic regression: Notation, intercept

- First, let's discuss algorithms for minimizing $Q(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y})$ in the logistic regression case
- We begin by noting that for logistic regression, it is not possible to eliminate the need for an intercept by centering the response variable
- Thus, in the derivations that follow, $\mathbf{y}$ will denote the original vector of 0-1 responses
- The design matrix $\mathbf{X}$ is standardized as before, but now contains an unpenalized column of 1's for the intercept, with corresponding coefficient β_0

Iteratively reweighted least squares algorithm

- Like unpenalized logistic regression itself, algorithms for penalized logistic regression employ Taylor series expansions to produce quadratic approximations to the loss function, thereby allowing us to use our previously derived solutions for linear regression
- This two-step approach is known, generally speaking, as the iteratively reweighted least squares (IRLS) algorithm:
  (1) Approximate the loss function based on $\boldsymbol{\beta}^{(m)}$
  (2) Solve for $\boldsymbol{\beta}^{(m+1)}$, the value that minimizes the approximated loss function
- These two steps are alternated until convergence

Taylor series expansion

Approximating the loss function via Taylor series expansion, we have

$$L(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}) \approx \frac{1}{2n}(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{W}(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta}),$$

where

- $\tilde{\mathbf{y}}$, the working response, is defined by

$$\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}^{(m)} + \mathbf{W}^{-1}(\mathbf{y} - \boldsymbol{\pi})$$

- $\mathbf{W}$ is a diagonal matrix of weights, with elements $w_i = \pi_i(1 - \pi_i)$
- $\boldsymbol{\pi}$ is evaluated at $\boldsymbol{\beta}^{(m)}$

Generalized linear models

- We are focusing today on logistic regression, but the same approach can be applied to fit penalized versions of any generalized linear model (GLM)
- To do so, one simply has to replace $\tilde{\mathbf{y}}$ and $\mathbf{W}$ with the appropriate expressions for the corresponding response distribution and link function used in the model
- For example, both `glmnet` and `ncvreg` have options for fitting Poisson regression models (`family="poisson"`) using the same technique we describe here for logistic regression

Coordinate descent

- Following the quadratic approximation, the objective function closely resembles the objective we're used to from linear regression, but now with observation weights $\{w_i\}$
- We can still employ coordinate descent, but the presence of these weights changes the form of the updates
- Let

$$\begin{aligned} v_j &= n^{-1}\mathbf{x}_j^\top \mathbf{W}\mathbf{x}_j \\ \mathbf{r} &= \mathbf{W}^{-1}(\mathbf{y} - \boldsymbol{\pi}) \\ z_j &= \frac{1}{n}\mathbf{x}_j^\top \mathbf{W}(\tilde{\mathbf{y}} - \mathbf{X}_{-j}\boldsymbol{\beta}_{-j}^{(m)}) \\ &= \frac{1}{n}\mathbf{x}_j^\top \mathbf{W}\mathbf{r} + v_j\beta_j^{(m)} \end{aligned}$$

Coordinate-wise updating

- Now, the coordinate-descent update for lasso is

$$\beta_j \leftarrow \frac{S(z_j|\lambda)}{v_j}$$

- For the elastic net,

$$\beta_j \leftarrow \frac{S(z_j|\lambda_1)}{v_j + \lambda_2}$$

- For MCP,

$$\beta_j \leftarrow \begin{cases} \frac{S(z_j|\lambda)}{v_j - 1/\gamma} & \text{if } |z_j| \leq v_j\gamma\lambda \\ \frac{z_j}{v_j} & \text{if } |z_j| > v_j\gamma\lambda \end{cases}$$

Remarks

- Note that the coordinate descent portion (2) of the algorithm must also involve the updating of the intercept term:

$$\beta_0 \leftarrow \frac{z_j}{v_j}$$

- Speaking of step (2), we typically do not iterate until convergence in order to obtain $\widehat{\boldsymbol{\beta}}^{(m+1)}$, but simply make one pass of coordinate updates, then re-approximate
- As is the case for the traditional IRLS algorithm for GLMs, this algorithm is not guaranteed to converge
- Typically, however, this is not a problem in practice, at least not for pathwise approaches, as the "warm starts" phenomenon provides protection against bad initial values

Model saturation

- One exception to this remark is that convergence does tend to be a problem with saturated models
- To protect against this, the `glmnet` and `ncvreg` packages will terminate the pathwise algorithm early if saturation ($R^2 > 0.99$) is detected
- Another numerical issue worth mentioning is that weights are typically capped at a minimum value ϵ to prevent fitted probabilities of 0 or 1, which also tends to happen as models become saturated

Reweighting and γ

- Before moving on, it is worth considering how the reweighting affects nonconvex penalties such as MCP and SCAD with respect to convexity and the choice of γ
- In linear regression, the scaling factor by which solutions are adjusted toward their unpenalized solution is a constant ($1 - 1/\gamma$ for MCP) for all values of λ and for each covariate
- Furthermore, for standardized covariates, this constant has a universal interpretation for all linear regression problems, meaning that defaults such as $\gamma = 3$ can be used and will be universally reasonable (though not necessarily optimal, of course)

Reweighting and γ (cont'd)

- In logistic regression, however, this scaling factor ($v_j - 1/\gamma$ for MCP) is different for each data set and for each feature
- This makes choosing an appropriate value for γ considerably more difficult and robs the parameter of a consistent interpretation
- For example, suppose we attempt to use $\gamma = 3$; for logistic regression, w_i cannot exceed 0.25, γ cannot exceed $1/v_j$, and $Q(\beta_j|\boldsymbol{\beta}_{-j})$ is no longer convex and does not have a unique minimum

Adaptive rescaling

- To resolve these difficulties, `ncvreg` takes an approach known as *adaptive rescaling*, which replaces $p_{\lambda,\gamma}(|\beta_j|)$ by $p_{\lambda,\gamma}(|v_j\beta_j|)$
- The consequence is that the updating steps become simple extensions of the linear regression updating steps:

$$\beta_j \leftarrow \frac{F(z_j|\lambda,\gamma)}{v_j} \qquad \text{(MCP)}$$

$$\beta_j \leftarrow \frac{T_S(z_j|\lambda,\gamma)}{v_j} \qquad \text{(SCAD)},$$

  where F and T_s are the firm and SCAD thresholding operators
- The purpose of this is to give γ a consistent meaning again and allow simple reasonable defaults such as $\gamma = 3$ to have the same meaning in other GLMs as they do in linear regression

Leukemia data (refresher)

- For the second half of this lecture, I'd like to (a) discuss various measures of predictive accuracy (b) demonstrate how to fit penalized logistic regression models with `glmnet` and `ncvreg` and (c) illustrate using a real data set
- For our case study data set, we will use the Leukemia data that we previously analyzed back at the beginning of the course using a multiple testing approach
- To refresh your memory, the data involved expression levels for 7,129 genes and 72 patients, of whom 47 patients had acute lymphoblastic leukemia (ALL) and the other 25 patients had acute myeloid leukemia (AML)

Deviance

- The most natural measure of predictive accuracy is to simply use the loss function (i.e., the log-likelihood); this is equivalent to the idea of deviance in GLM theory
- Specifically, let

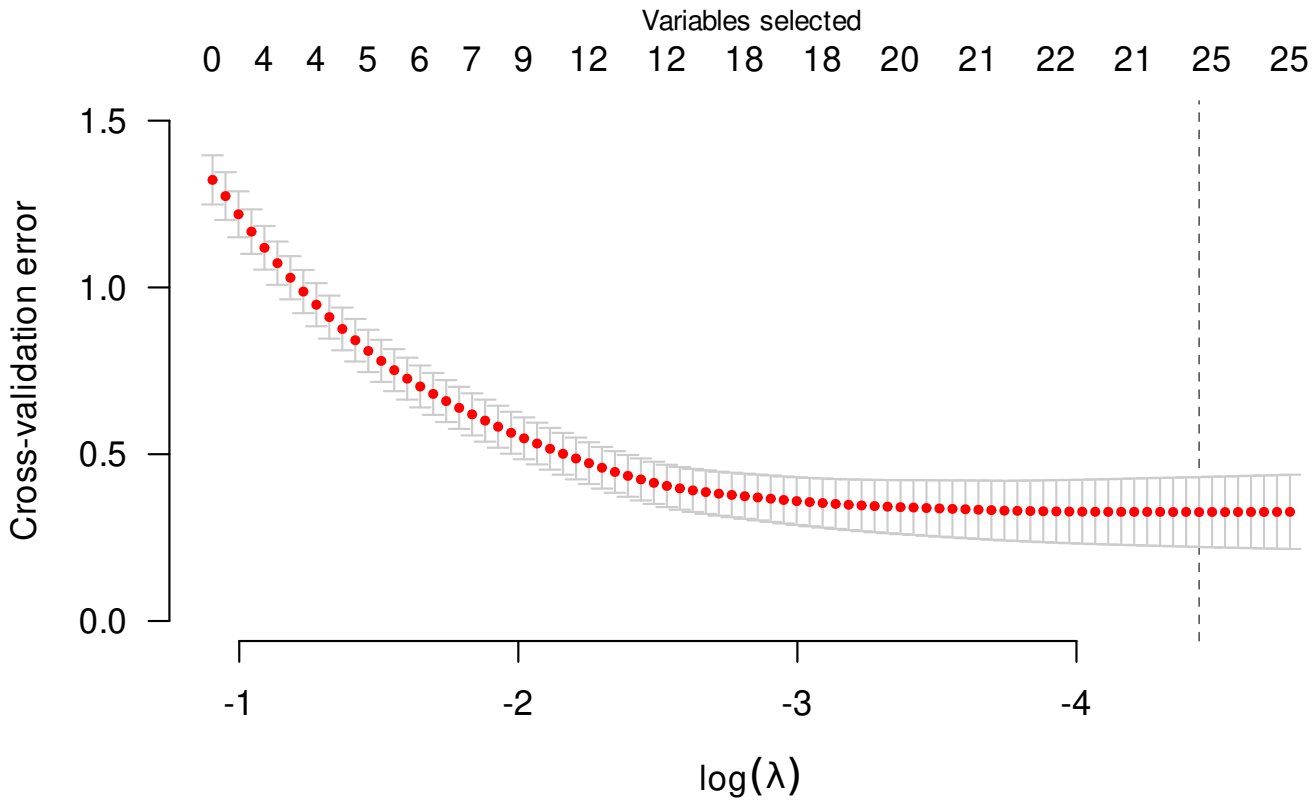
$$d_i^2 = -2\{y_i \log \hat{\pi}_{(i)} + (1 - y_i) \log (1 - \hat{\pi}_{(i)})\},$$

  where $\hat{\pi}_{(i)}$ denotes the predicted value of π_i based on the cross-validation fold from which observation i was excluded
- Then $D = \sum_i d_i^2$ is known as the *deviance*; the factor of 2 is so that the difference in deviance between two nested models will follow a χ^2 distribution in classical likelihood theory, but is essentially arbitrary for our purposes

```
# [1] 0.002001001
# [1] 1.386294
# [1] 13.81551
```

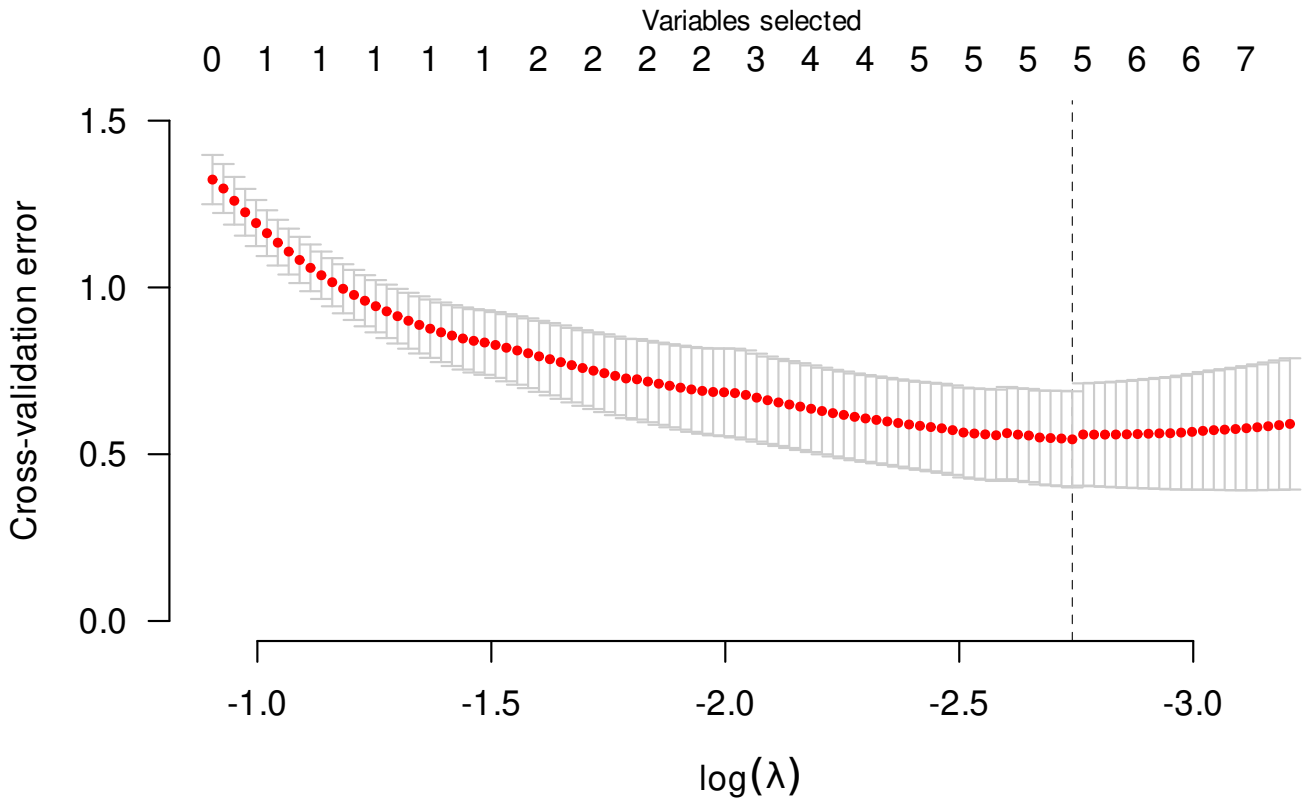
Deviance for the Leukemia data (lasso)

```
cv.glmnet(x, y, family = 'binomial')
```



Deviance for the Leukemia data (MCP)

```
cv.ncvreg(x, y, family = 'binomial', gamma = 4)
```



R^2

- As with linear regression, it is often more interpretable to look at the fraction of RSS/deviance explained, rather than the total (although of course the two are equivalent in terms of choosing an optimal value of λ)
- There are multiple ways of doing this; all revolve around the difference between $\Delta D(\lambda)$, the difference between D_0, the deviance of the null model, and $D(\lambda)$, the (cross-validated) deviance at a given value of λ, and

Cox-Snell R^2

- For linear regression,

$$R^2 = 1 - \frac{\text{RSS}_1}{\text{RSS}_0}$$

or in terms of deviance,

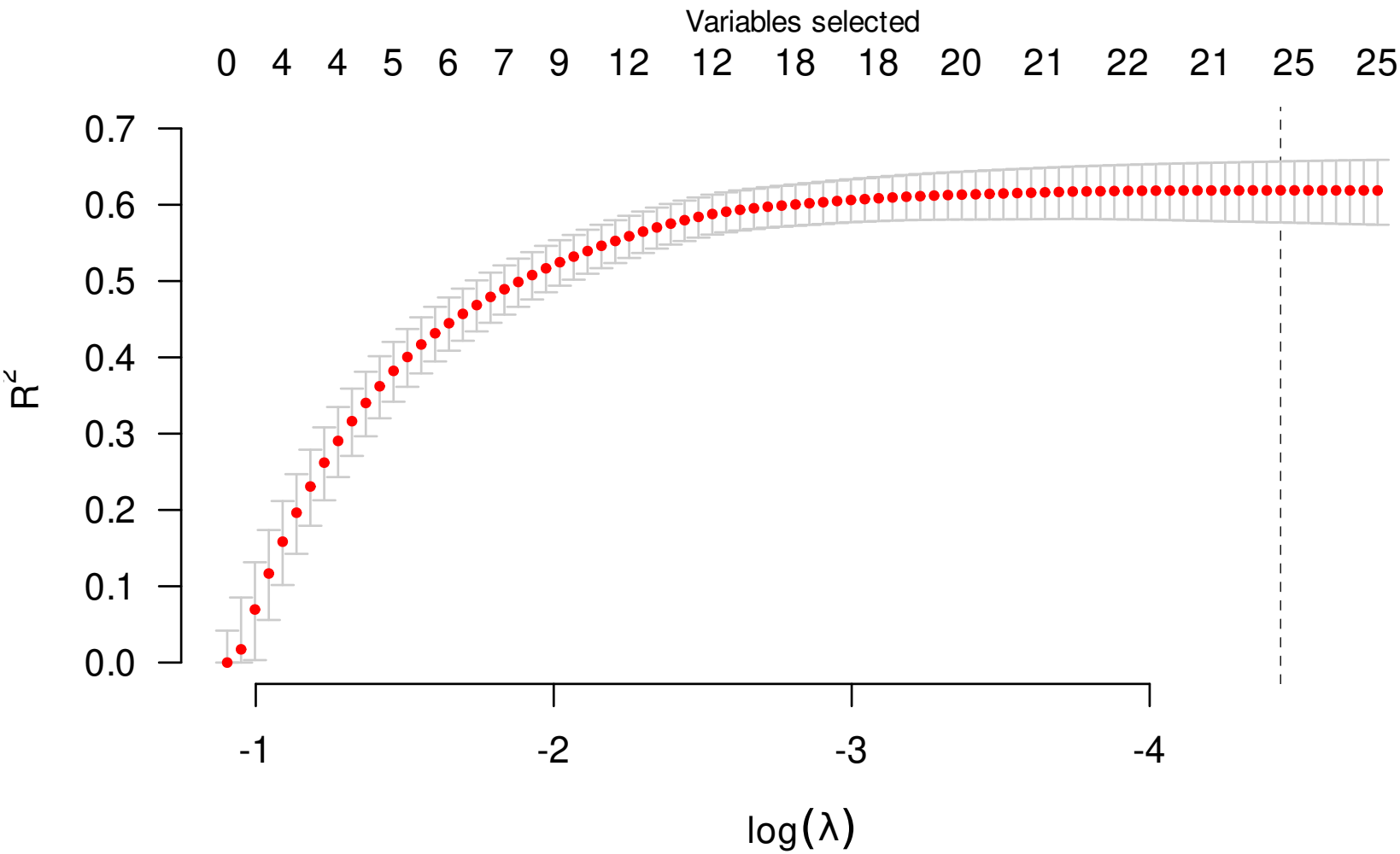
$$\Delta D = n \log \frac{\text{RSS}_1}{\text{RSS}_0}$$

- This suggests

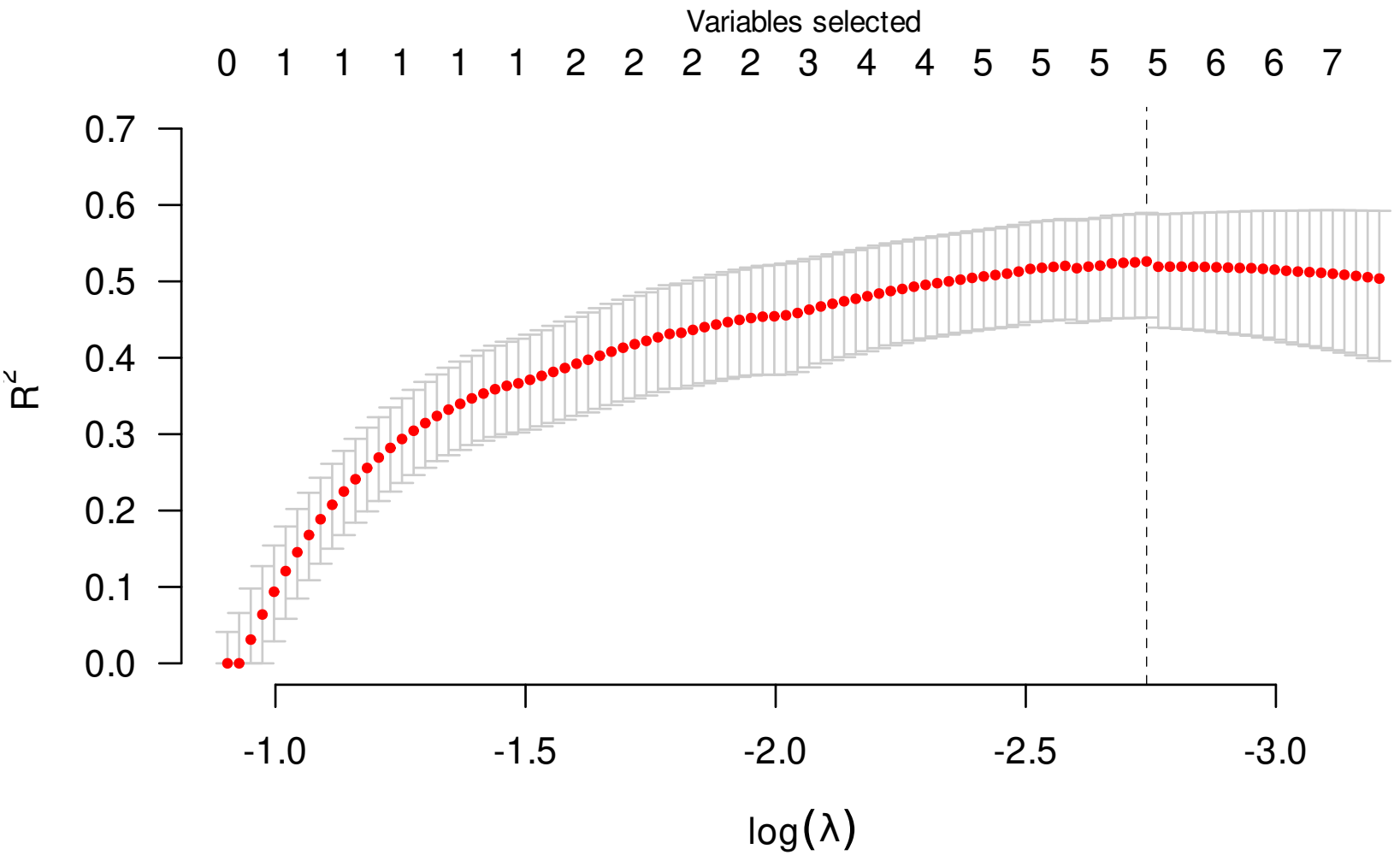
$$R^2 = 1 - \exp\{\Delta D(\lambda)/n\};$$

known as the *Cox-Snell* R^2, this is what ncvreg returns

R^2 for the Leukemia data (lasso)



R^2 for the Leukemia data (MCP)



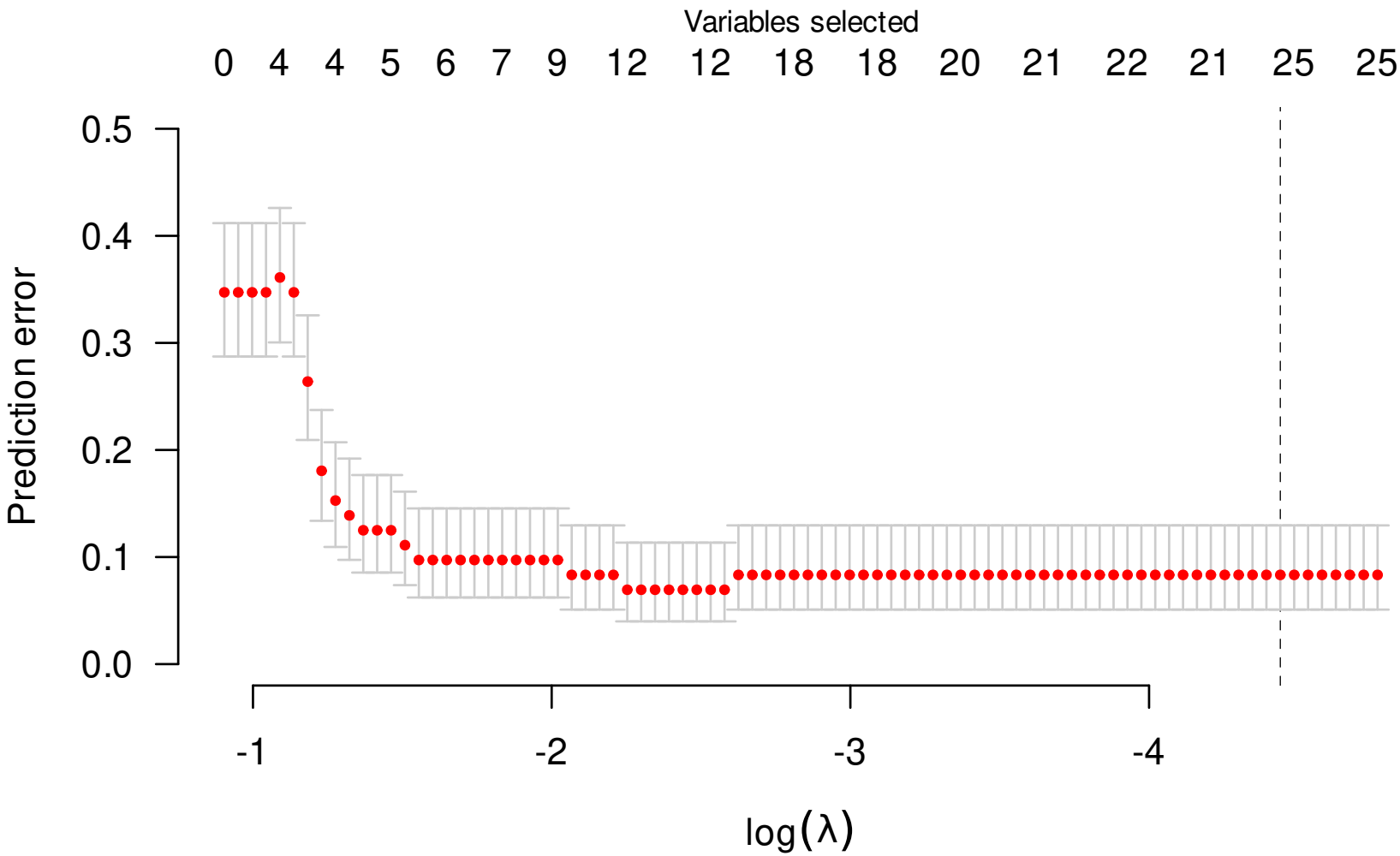
Misclassification error

- A more black-and-white measure of predictive accuracy is the ability of the model to predict the outcome (here, leukemia type) correctly, in the sense that $\hat{\pi}_{(i)} > 0.5$ implies a prediction that $Y_i = 1$
- This is generally referred to as the *misclassification error* of the model
- The advantage of misclassification error is ease of interpretation: everyone know what it means to say that the model got 78% of its predictions correct
- The disadvantage is that it is less stable and well-behaved than the deviance; small changes in λ can lead to large changes in the misclassification error as $\hat{\pi}_{(i)}$ moves from one side of 0.5 to the other

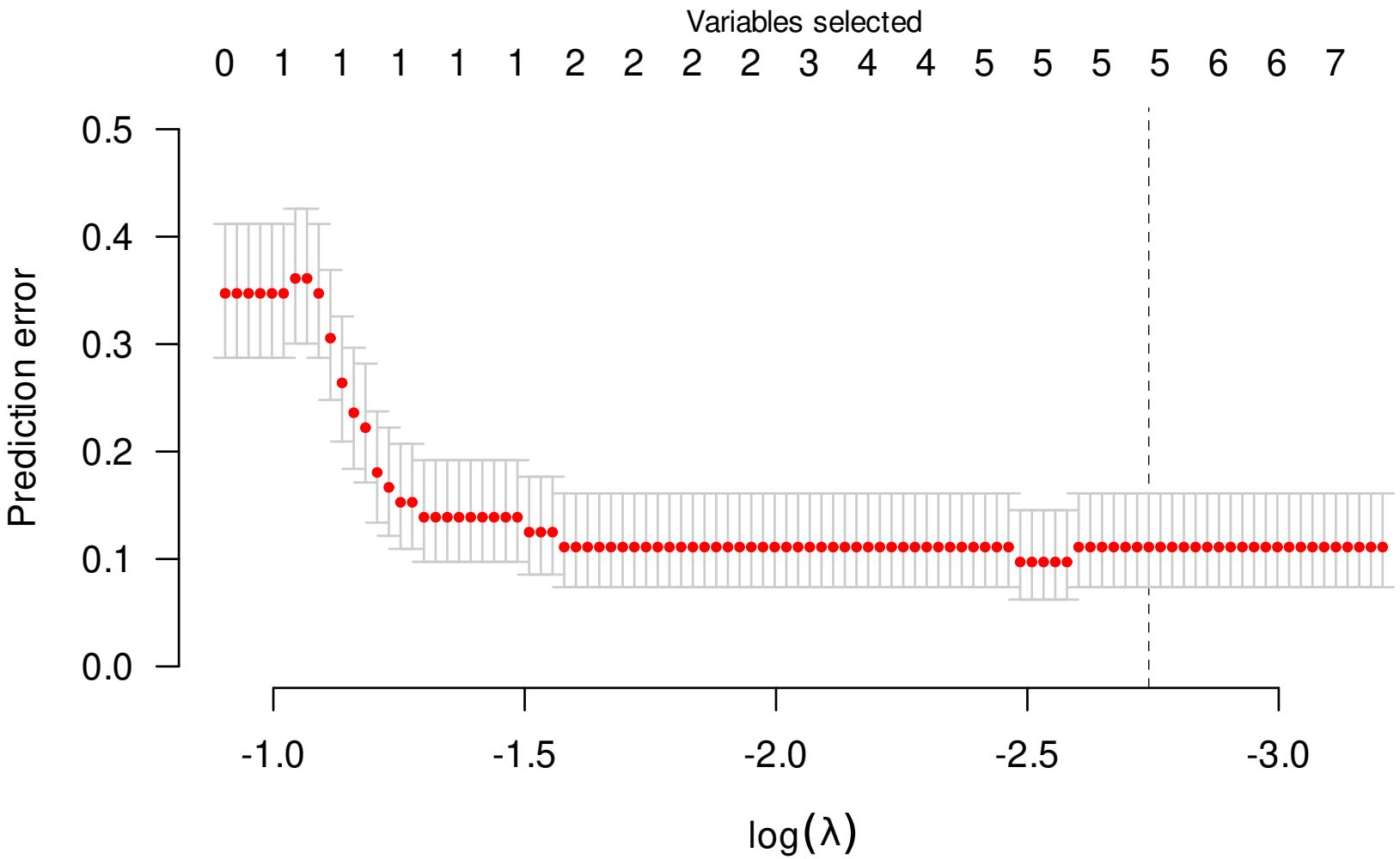
Standard errors

- We've been showing standard errors, but I have not yet discussed how they are estimated
- For deviance and R^2, `glmnet` and `ncvreg` estimate the standard error of $\bar{d}^2$ through the simple formula $\text{SE} = \text{SD}(d_i^2)/\sqrt{n}$
- `glmnet` uses the same approach for misclassification error, though `ncvreg` calculates exact (Clopper-Pearson) confidence intervals based on the observed number of misclassifications and trials

Misclassification error (lasso)



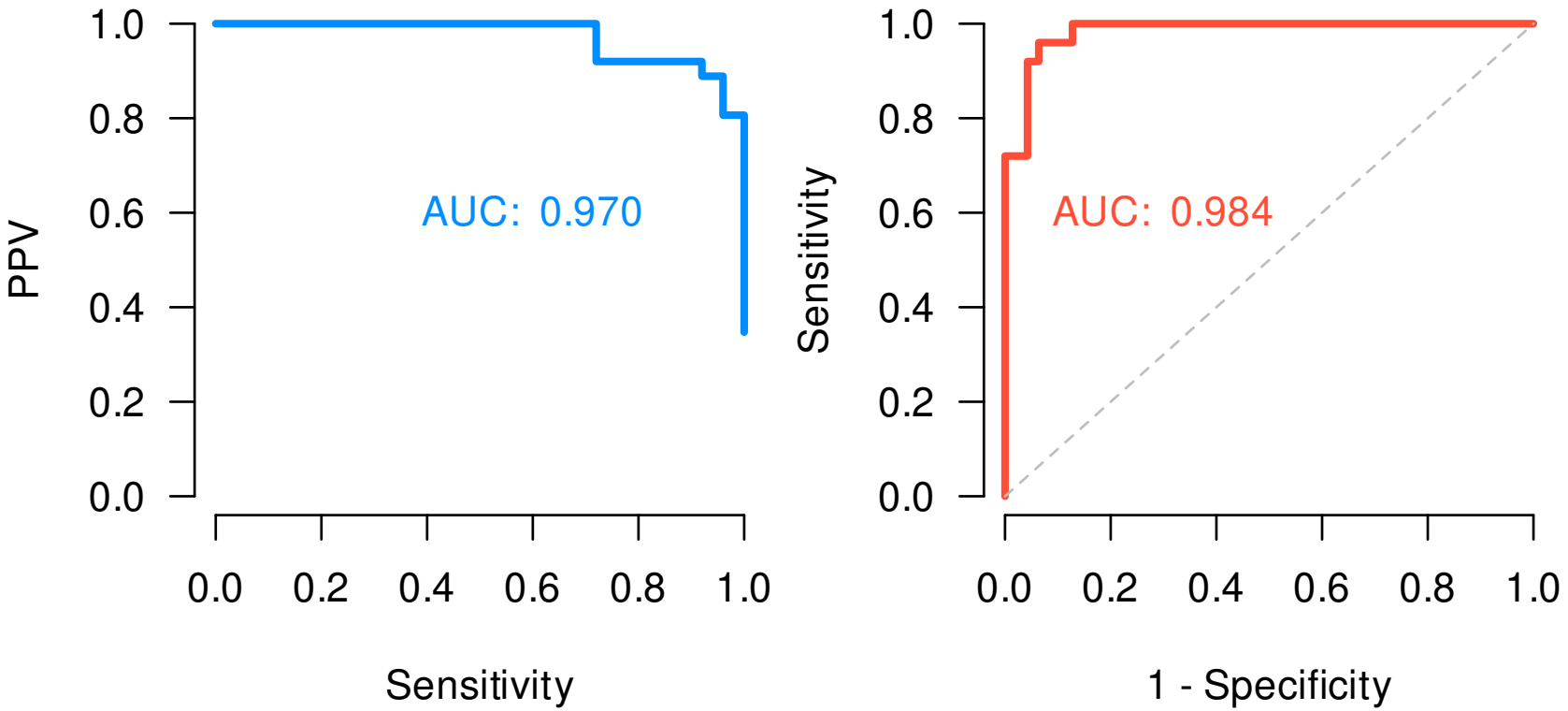
Misclassification error (MCP)



PR and ROC curves

- The preceding plots used $\hat{\pi}_{(i)} > 0.5$ for prediction, which is not always the most relevant cutoff $\hat{\pi}_{(i)} > c$
- Rather than assign a single, fixed cutoff, another approach is to consider the prediction accuracy across a range of cutoffs – this will produce a curve as the changing cutoff changes the number of false positives and false negatives
- The two most common curves are
  - ROC curves: Plotting sensitivity vs 1 - specificity
  - Precision-recall (PR) curves: Plotting precision (positive predictive value) against recall (sensitivity)

PR and ROC curves for Golub data (lasso)



Accessing cross-validated predictions

- Again, it is very important that you use the *cross-validated* linear predictors when calculating these curves
- As with all measures of predictive accuracy, AUC will be substantially inflated if you use the full-data linear predictors

```
cv.glmnet(x, y, family = 'binomial',
          type.measure = 'auc')
cv.ncvreg(x, y, family = 'binomial', returnY = TRUE)
```