Linear models

Federica Eduati

Eindhoven University of Technology
Department of Biomedical Engineering

2022

Learning goals

During this lecture we will:

- Define regression and classification problems using linear models.
- Describe the concept of regularization applied to linear models and understand the idea behind different regularisation approaches.
- Illustrate application of different regularization approaches to a specific case study and interpret the results.
- Compare the different regularisation methods from a theoretical prospective and using the case study

Materials:

- Chapters 3 and 4 from Friedman et al., *The Elements of Statistical Learning*

Overview

Topics covered in this lecture:

Linear models for regression

Subset selection

Shrinkage methods

Classification

Logistic regression

Introduction to linear models

Linear models: a linear combination of the *inputs* (also known as *predictors*, *features* or *independent variables*) is used to predict one or more *outputs* (also known as *responses* or *dependent variables*).

The prediction task is defined as:

- *Regression*: when we predict *quantitative outputs*.
- *Classification*: when we predict *qualitative outputs* (also referred to as *categorical or discrete variables*).

Linear regression model

Given a vector of inputs $X^T = (X_1, X_2, \ldots, X_p)$, where p is the number of features, a linear regression model has the form:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

where β_j are the unknown parameters or coefficients.

Note that the model remains linear in the parameters even if the the variables X_j are polynomial (e.g. $X_2 = X_1^2$) or derive from interactions between variables (e.g. $X_3 = X_1 \cdot X_2$).

Estimation via least squares

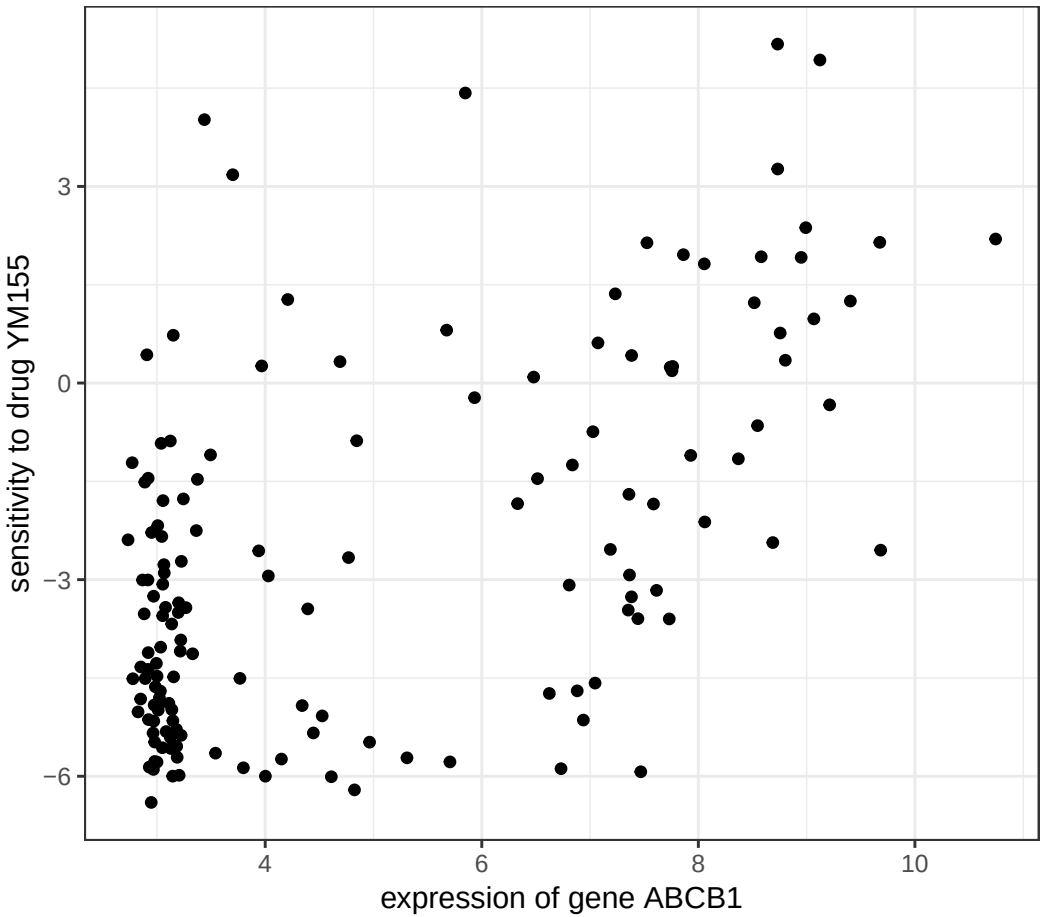
Parameters β_j can be estimated from a set of training data $(x_1, y_1) \ldots (x_N, y_N)$, where each $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ is a vector of feature measurements for the ith case.

With the *least squares* estimation methods, the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ are selected by minimizing the residual sum of squares:

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

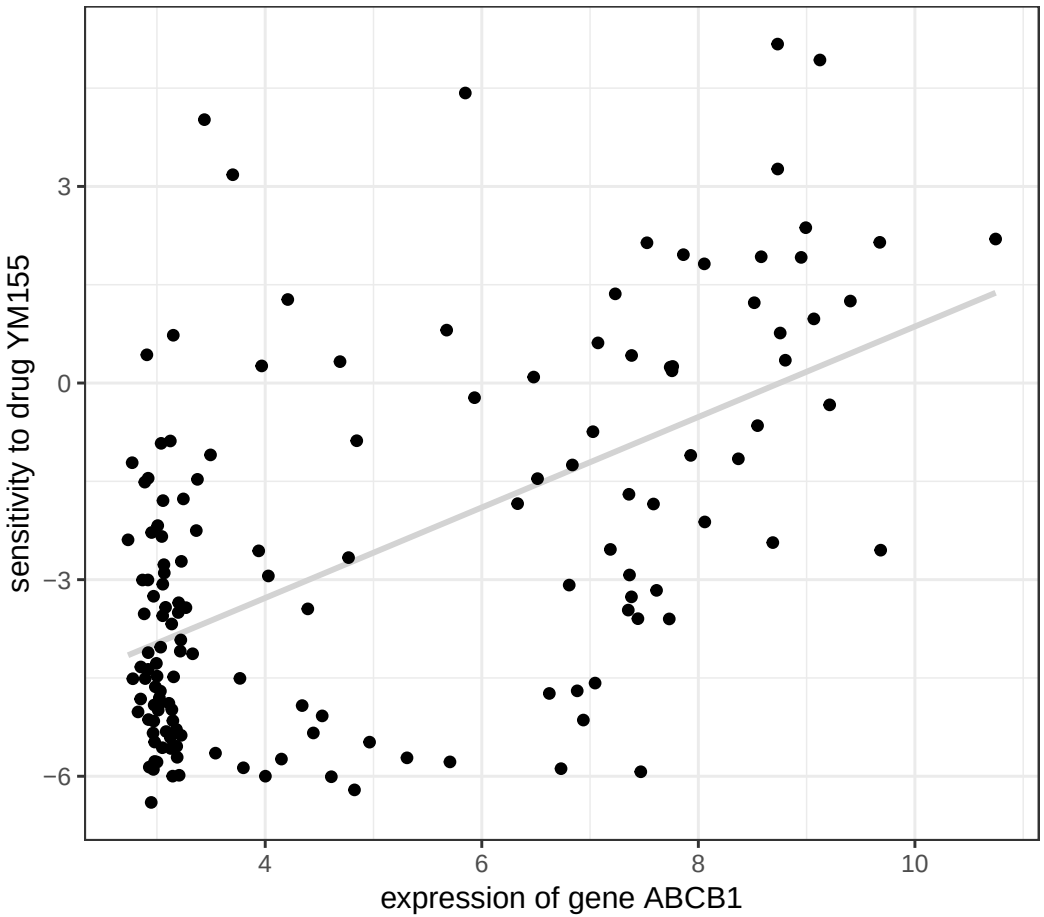
Example of univariate linear regression

$\hat{y} = \beta_0 + \beta_1 x$, where output variable Y represents the sensitivity to a certain drug and input variable X the expression of a gene.



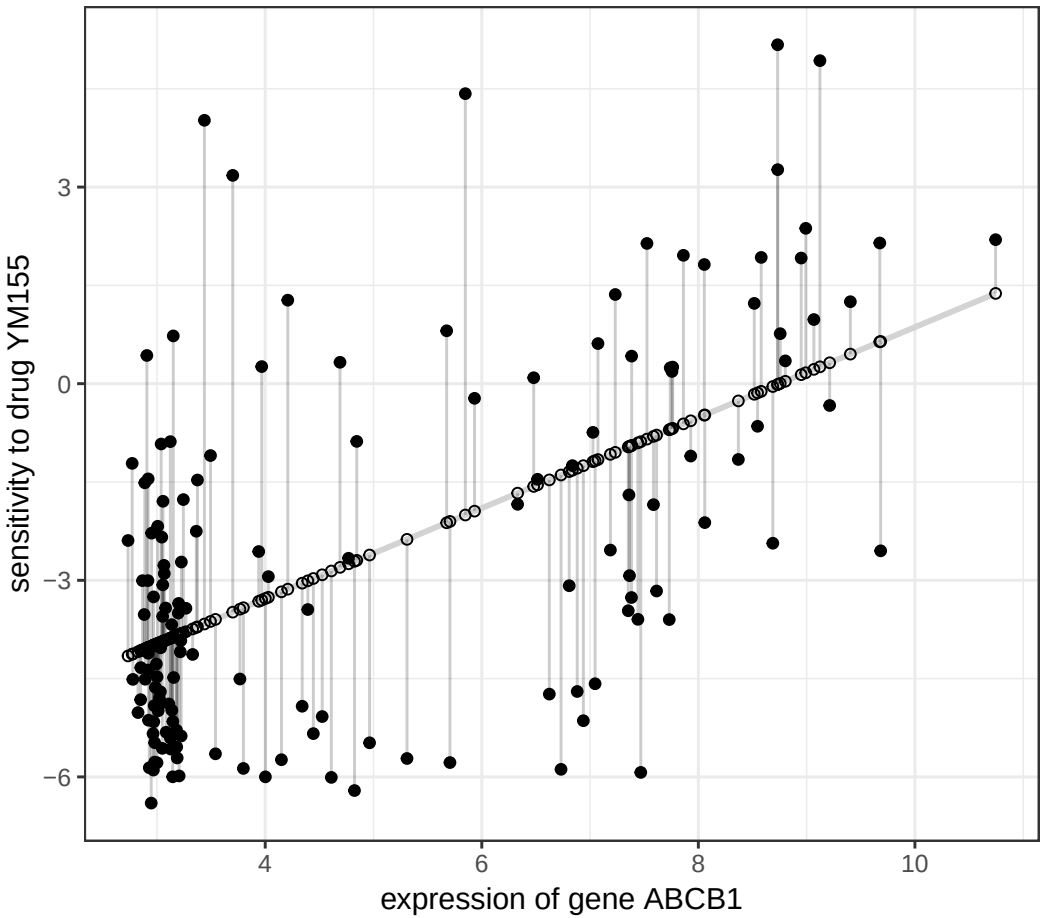
Example of univariate linear regression

Least square fitting: linear function of X that minimises the sum of squared residuals.



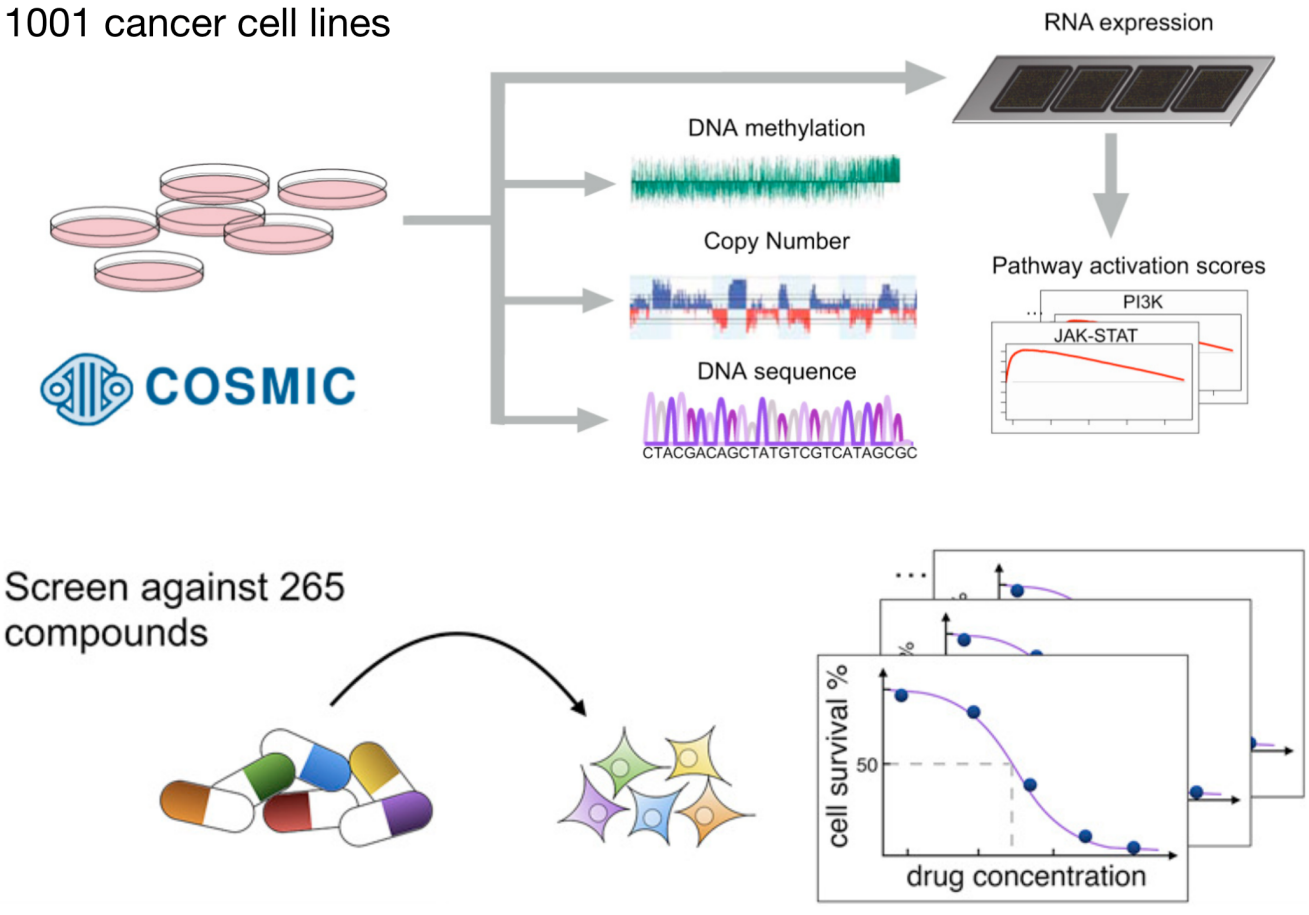
Example of univariate linear regression

Least square fitting: linear function of X that minimises the sum of squared residuals. $RSS = \sum_{i=1}^{N}(y_i - \beta_0 - \beta_1 x_i)^2$



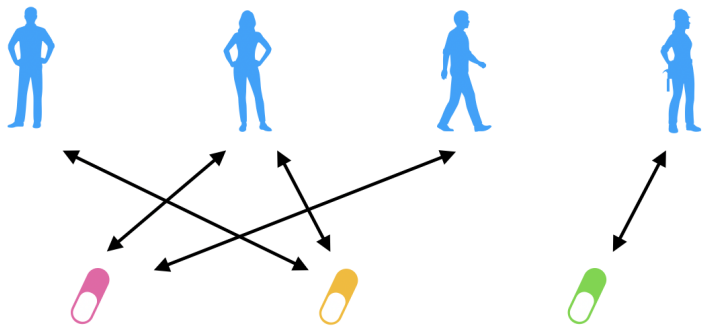
Genomics of Drug Sensitivity in Cancer (GDSC) project

We will use as example data from the GDSC project (https://www.cancerrxgene.org/) Iorio et al., "A landscape of pharmacogenomic interactions in cancer".



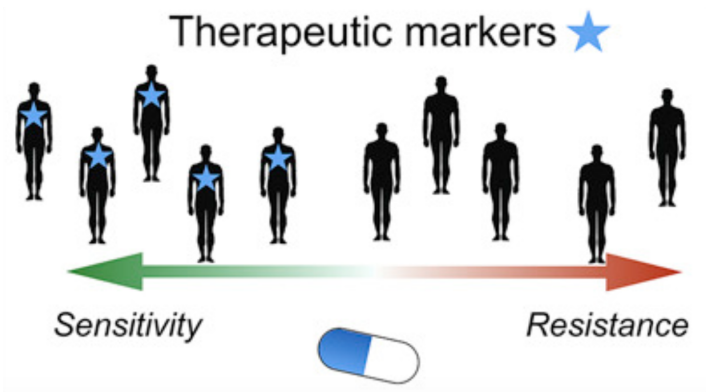
Genomics of Drug Sensitivity in Cancer (GDSC) project

Cancer patients tends to respond very differently to drugs. A big challenge in cancer research is to find ways to assign the optimal treatment to each patient.



Precision oncology:
match the right drug
to the right patient

The aim of the study is to find biomarkers of drug sensitivity.



Our dataset

We will consider data for 148 cell lines from four cancer types.

ID	Cancer type
COAD/READ	Colorectal adenocarcinoma
NB	Neuroblastoma
KIRC	Kidney renal clear cell carcinoma
BRCA	Breast carcinoma

We can interpret the prediction of drug sensitivity as a linear regression problem using:

- Output variable: Sensitivity to YM155 (Sepantronium bromide), as natural log of the fitted IC50.
- Predictors: expression of 244 genes (the ones with higher variance), as RMA normalised expression.

Feature scaling

Idea: make sure features are on a similar scale.

$$x_i = \frac{x_i - \mu_i}{\sigma_i}$$

where μ_i and σ_i are respectively the average and the standard deviation of all the values of feature i.

This makes the estimated coefficients comparable and speeds up the optimization algorithm.

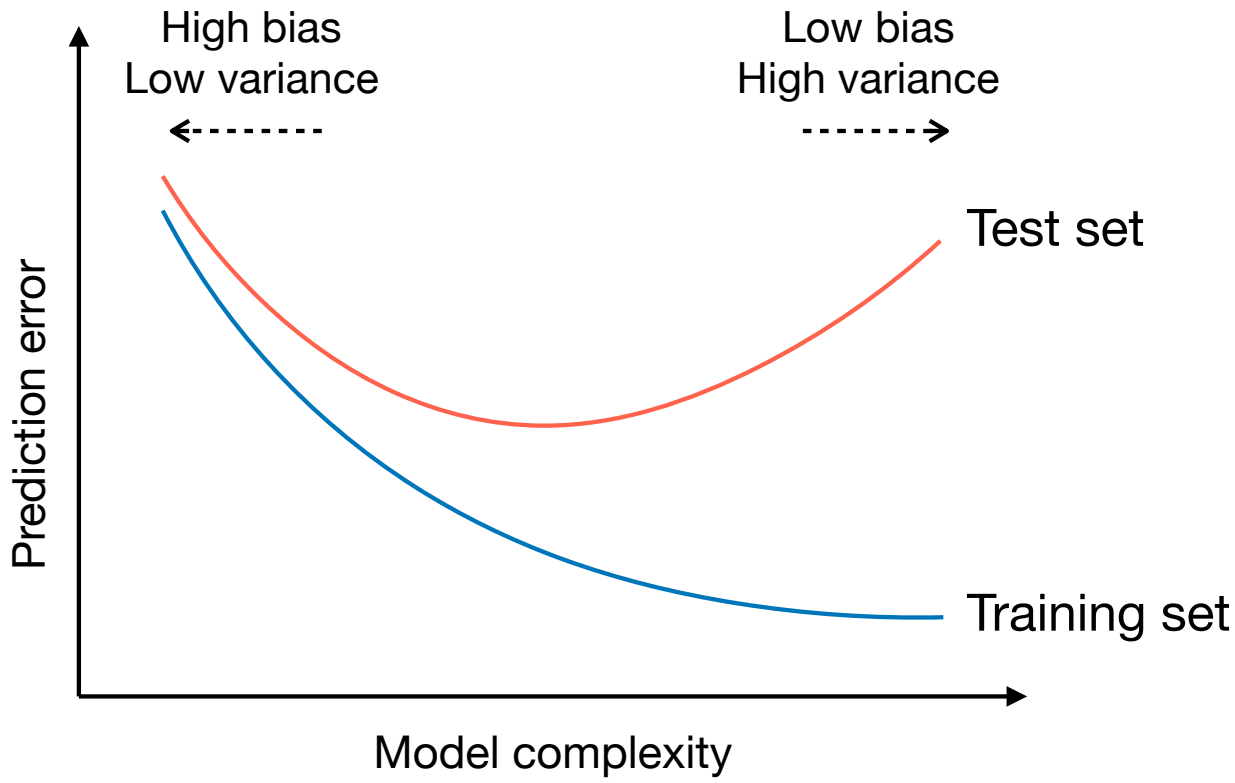
If we standardize the predictors, the solution for $\hat{\beta}_0$ is $\overline{y}$, therefore we fit a model without the intercept.

Model complexity

Especially when we have a large number of features (p large compared to N), least squares estimates can suffer from:

- Low *prediction accuracy*: high model complexity gives low bias but high variance, setting some coefficients to zero can reduce the variance of predictions (at the price of increasing the bias).
- Poor *interpretability*: we might want to identify which predictors are really useful to explain the output.

Model complexity and overfitting



Complex models tend to overfit the training data and perform poorly on the test data.

Subset selection

With *subset selection* we want to retain only a subset of the predictors and eliminate the rest from the model.

We can try a lot of different models, each containing a different subset of the predictors, and check which model is the best, e.g. based on *Akaike information criterion (AIC)* or *Bayesian information criterion (BIC)*.

Problem: there are a total of 2^p models that contain subsets of p predictors!!

Subset selection

Three classical approaches to step-wise *subset selection*:

- *Forward selection*. Start from the null model (i.e. only intercept) and sequentially add to the model the variable that gives lowest RSS (i.e. higest improvement of the fit).
- *Backward selection*. Start with the full model (i.e. all variables) and sequentially remove the predictor with the largest p-value (i.e. lower impact on the fit).
- *Mixed selection*. Alternation of forward and backward steps to add variables that improves RSS while maintaining the p-value below a certain threshold.

Subset selection

Subset selection tends to:

- improve *interpretability*: only the most relevant predictors useful to explain the output are selected.
- still suffer from low *prediction accuracy*: high variance (discrete process in retaining and discarding predictors) often does not reduce prediction error.

Improving least square estimates with regularization

- least square estimates generally provide all non-zero coefficient.
- if $p > N$, solutions are not unique (i.e. multiple solutions with same minimum, often overfitting the data).

Need to constrain or regularize the estimation process.

Idea: shrink regression coefficients by imposing a penalty on their size.

Ridge regression

Ridge regression penalizes the sum of squares of the coefficients (L2 regularization).

$$\hat{\beta}^{ridge} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

$$= \underset{\beta}{\text{argmin}} \left\{ RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

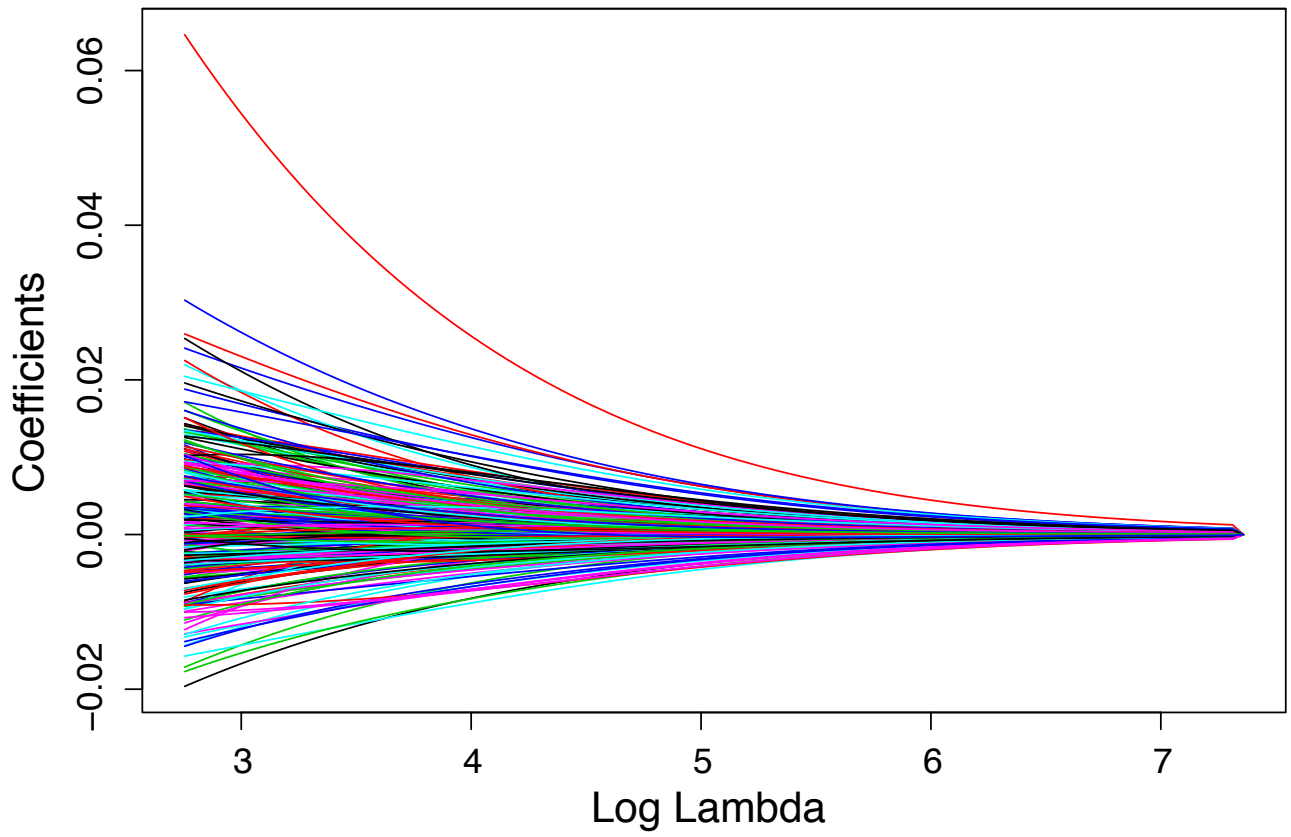
Ridge regression: the meaning of λ

$$\hat{\beta}^{ridge} = \underset{\beta}{\text{argmin}} \left\{ RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

$\lambda \geq 0$ is a tuning parameter.

- $\lambda = 0$ corresponds to the least square estimates.
- for larger values of λ, coefficients $\beta_1, \ldots, \beta_p$ will be shrinked towards zero

Ridge regression: the meaning of λ



Profile of the Ridge regression coefficients for the GDSC example.

The effect of regularization on bias and variance

In matrix form we can rewrite the Ridge regression as:

$$(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta$$

And the Ridge regression solution becomes:

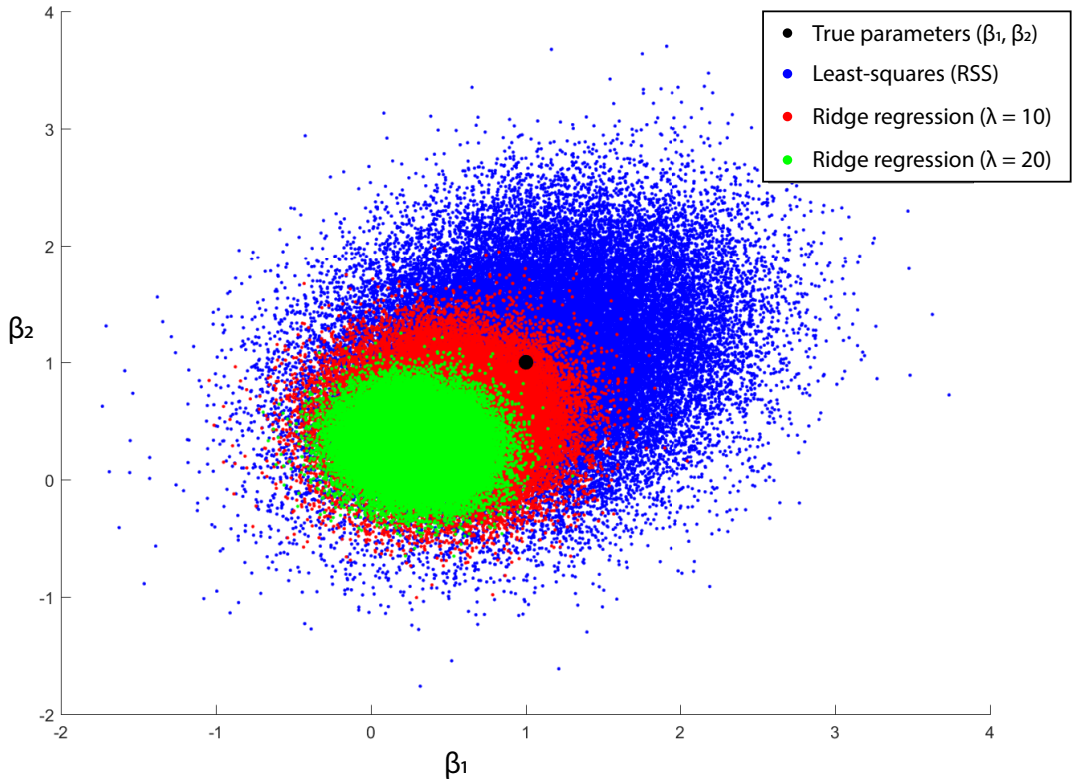
$$\hat{\beta}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix. This is linear in $\mathbf{y}$ and in closed form.
Assuming that $\mathbf{y} = \mathbf{X}\beta_{\text{true}} + \mathbf{e}$, where $\mathbf{e}$ is a random error on the output variable, the Ridge regression solution can be written as:

$$\begin{aligned}\hat{\beta}^{ridge} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T(\mathbf{X}\beta_{\text{true}} + \mathbf{e}) \\ &= \underbrace{(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}\beta_{\text{true}}}_{\text{bias}} + \underbrace{(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{e}}_{\text{variance}}\end{aligned}$$

The effect of regularization on bias and variance

Effect of λ on the bias-variance trade off in a simulated example ($y_i = x_{i1}\beta_1 + x_{i2}\beta_2 + e_i$, $i = 1, \ldots, 20$). The model is simulated 50k times, each time the noise e_i is randomly generated.



Lasso regression

Lasso regression penalizes the sum of the absolute value of the coefficients (L1 regularization).

$$\hat{\beta}^{lasso} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

$$= \underset{\beta}{\text{argmin}} \left\{ RSS + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

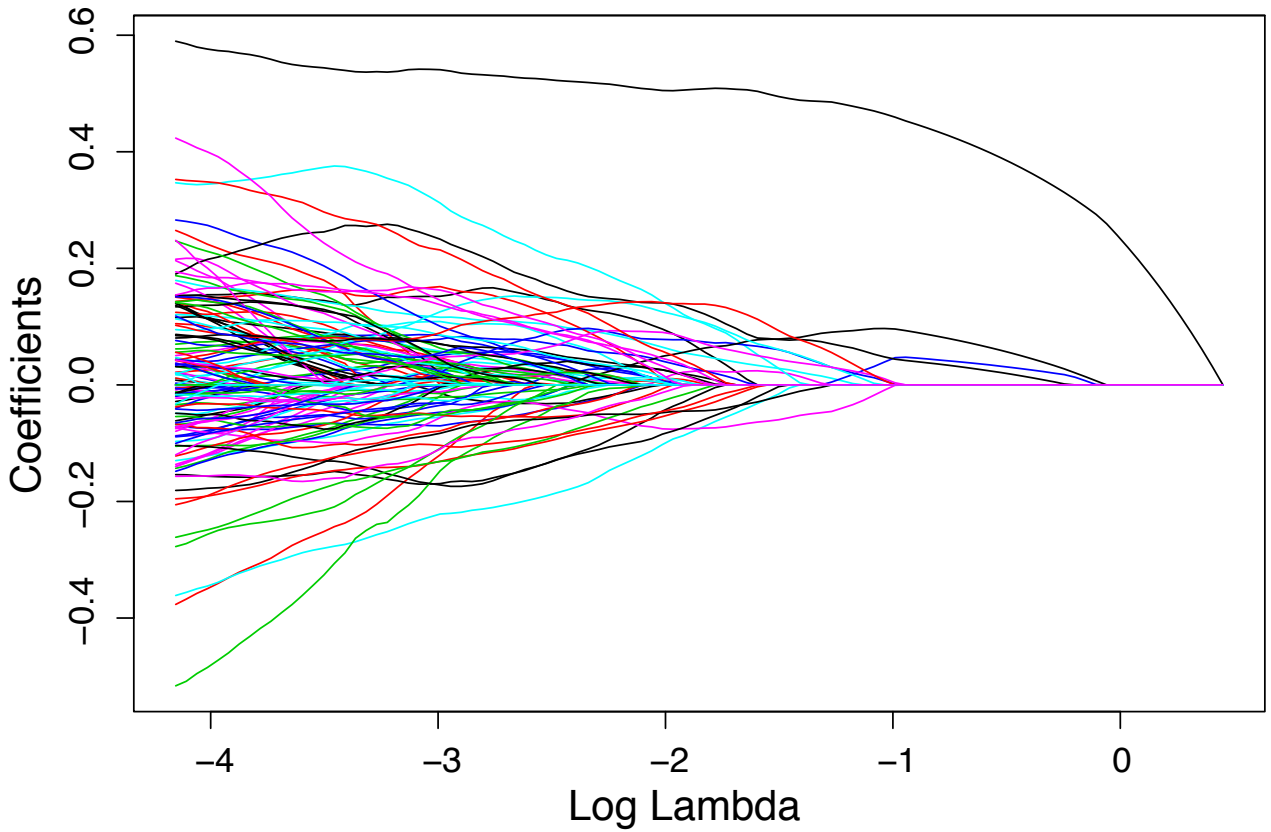
Lasso regression: the meaning of λ

$$\hat{\beta}^{lasso} = \underset{\beta}{\text{argmin}} \left\{ RSS + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

$\lambda \geq 0$ is a tuning parameter.

- $\lambda = 0$ corresponds to the least square estimates
- larger values of λ, will make some of the coefficients $\beta_1, \ldots, \beta_p$ to be exactly zero, performing a continuous subset selection.

Lasso regression: the meaning of λ



Profile of the Lasso regression coefficients for the GDSC example.

Comparing Lasso and Ridge regression

An equivalent way or writing the Ridge and Lasso problems is:

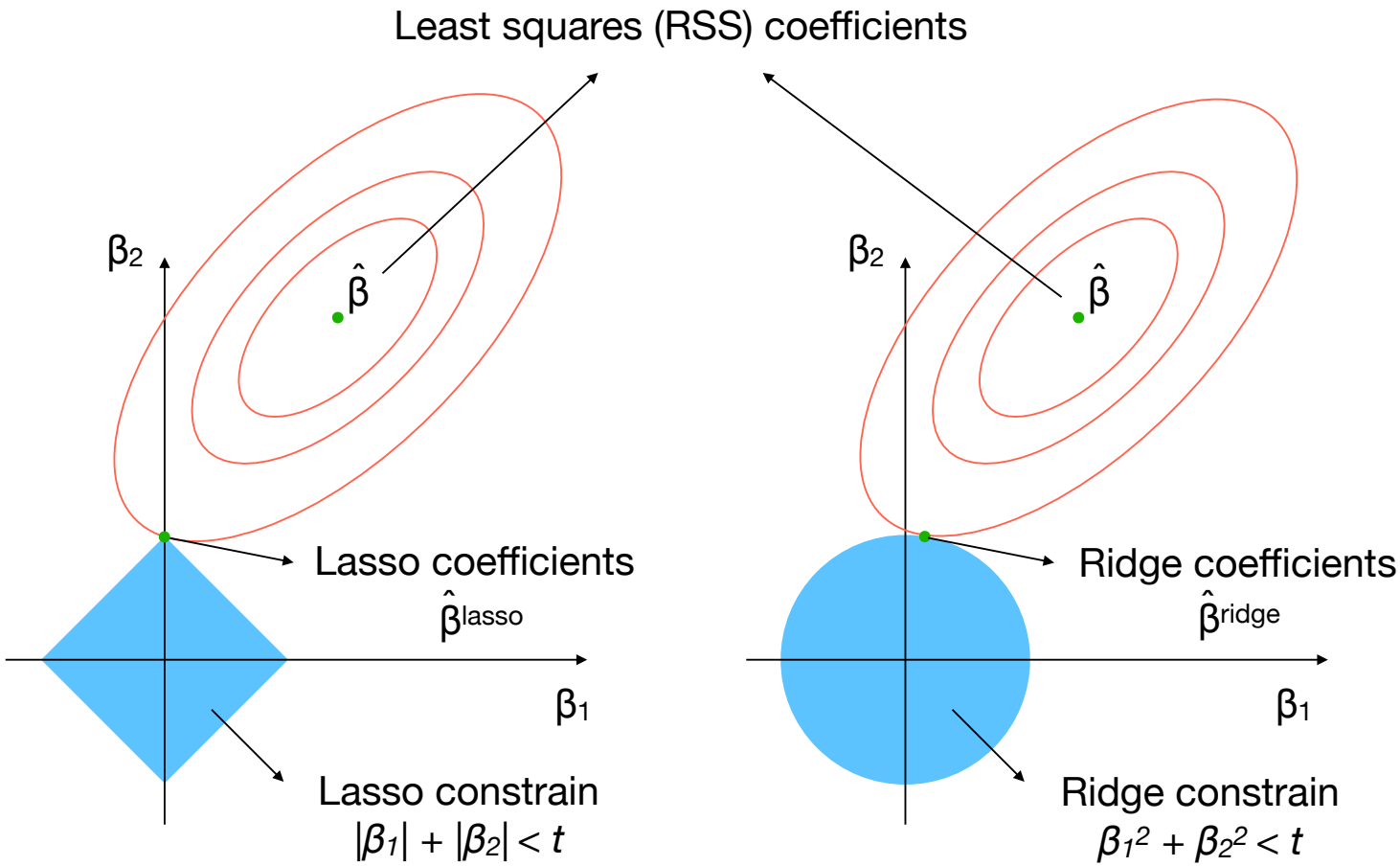
$$\underset{\beta}{\text{argmin}} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

subject to:

- $\sum_{j=1}^{p} \beta_j^2 < t$ for Ridge regression
- $\sum_{j=1}^{p} |\beta_j| < t$ for Lasso regression

It is possible to show that there is a one-to-one correspondence between the parameter λ and t.

Comparing Lasso and Ridge regression



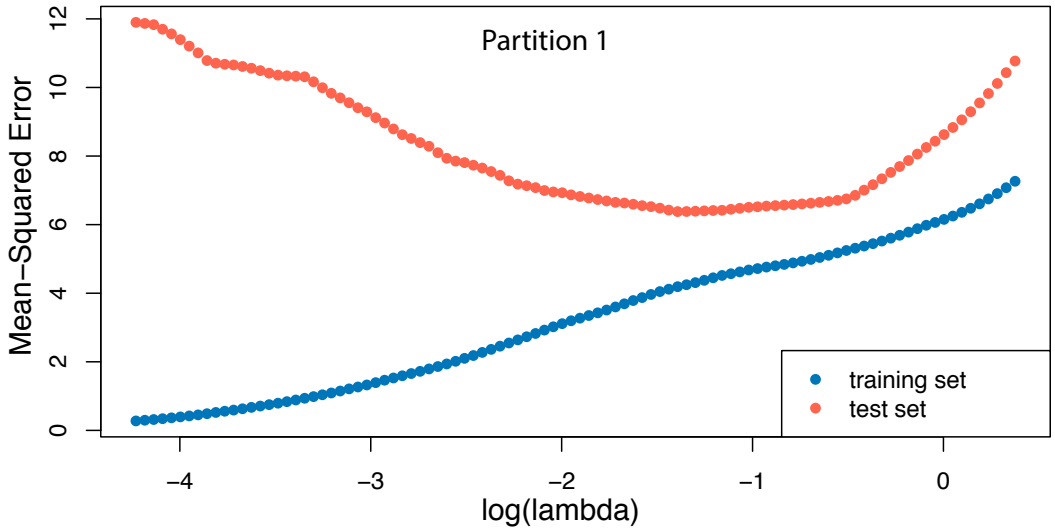
How to select optimal λ

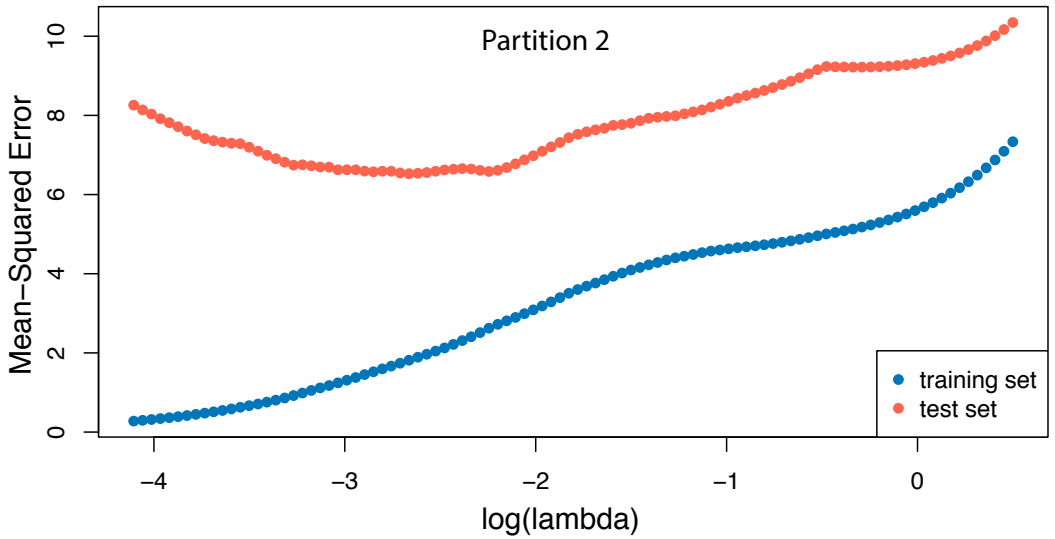
We need a method to select the tuning parameter λ for both Lasso and Ridge regression.

We want the model that provides the best predictions on the test set. In general this is very sensitive to the data used for training.

How to select optimal λ

Two data partitions - GDSC example (90% training, 10% test).



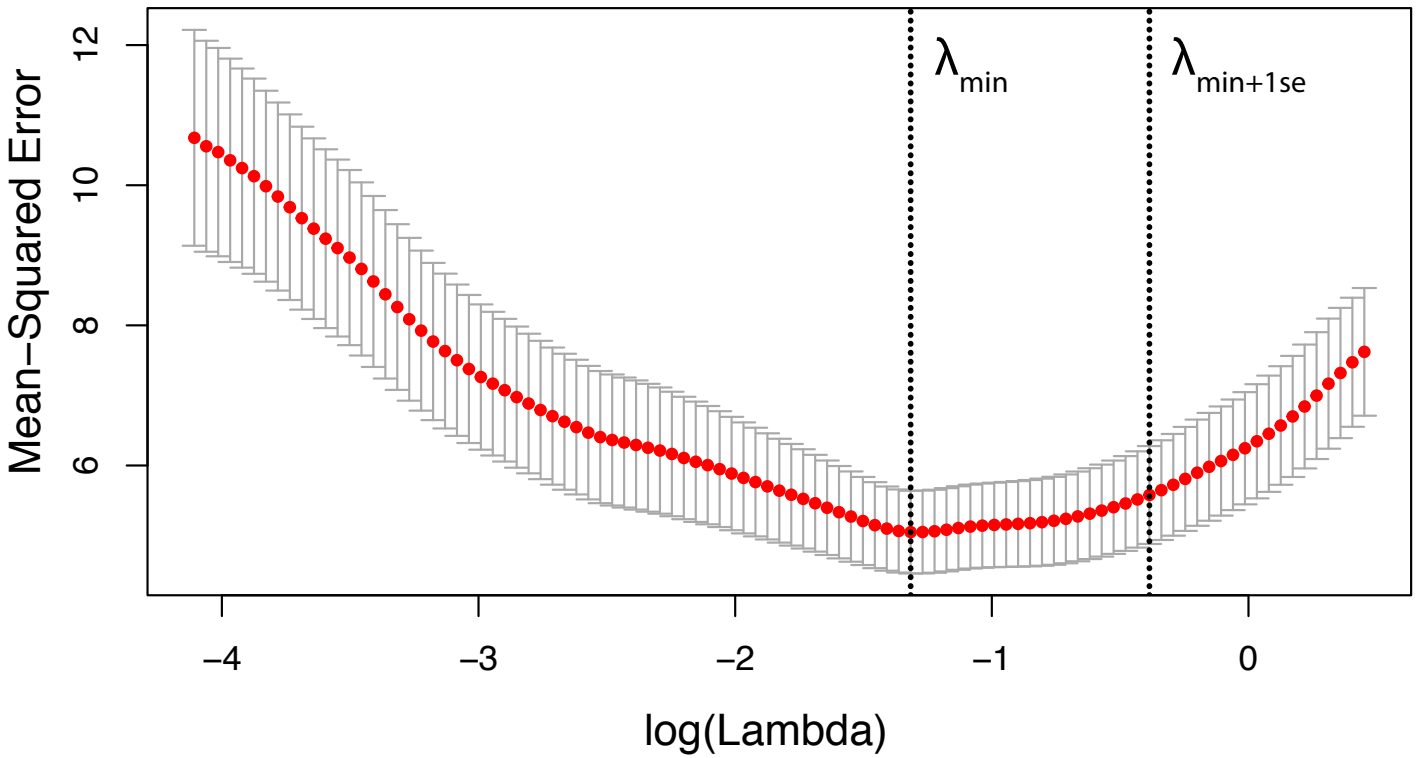


How to select optimal λ

We can use cross-validation following these steps:

1. Choose a grid of λ.
2. Optimize the model for each training set and compute Mean-Squared Error (MSE) for each validation set.
3. Choose the model with smallest cross-validation error.
4. Re-fit using all the available observations and the selected tuning parameter.

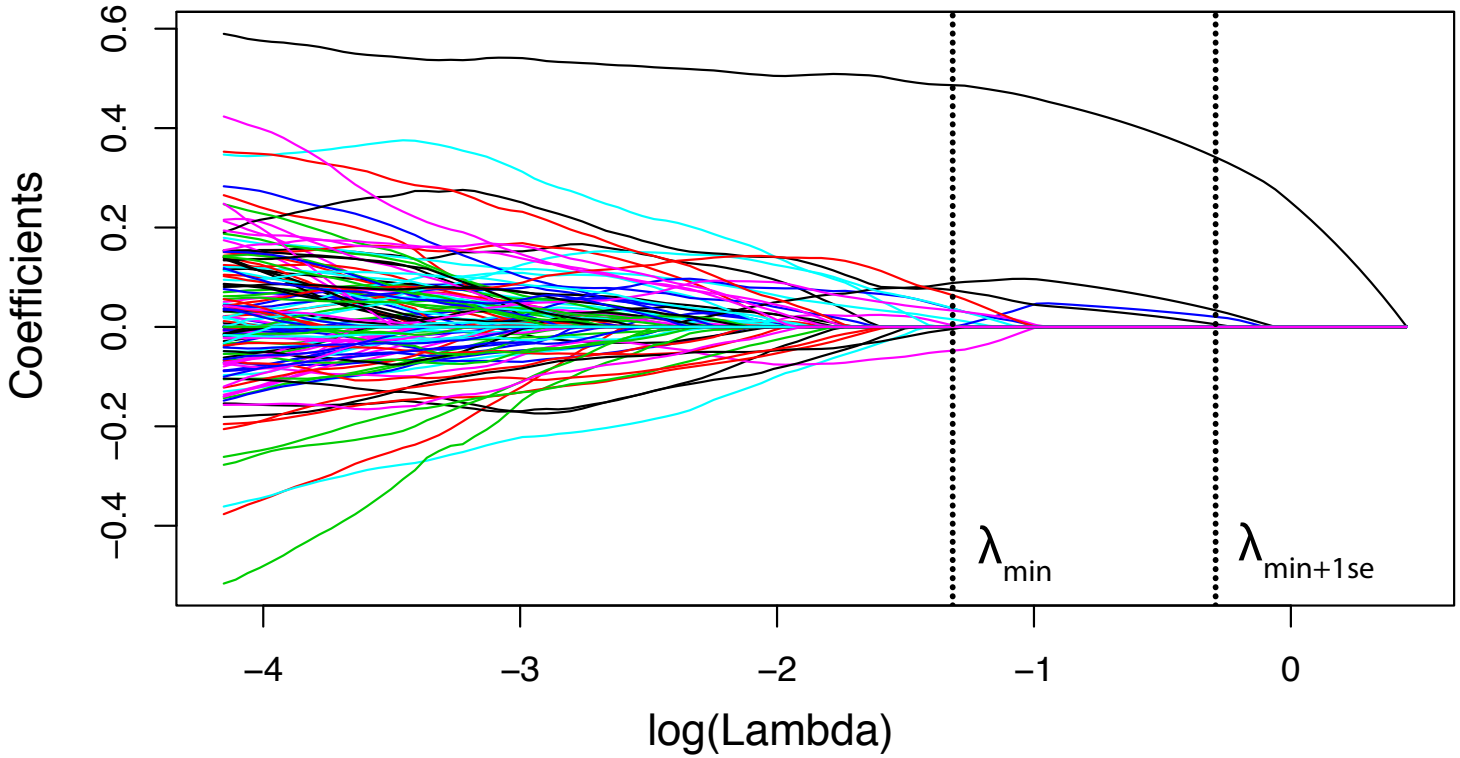
λ selection: GDSC example 10-fold cross validation



- λ_{min}: value of λ that gives the minimum MSE
- $\lambda_{min+1se}$: largest value of λ such that error is within 1 standard error of the minimum

Lasso for feature selection

GDSC example 10-fold cross validation



$\lambda_{min+1se}$ gives more sparse solutions.

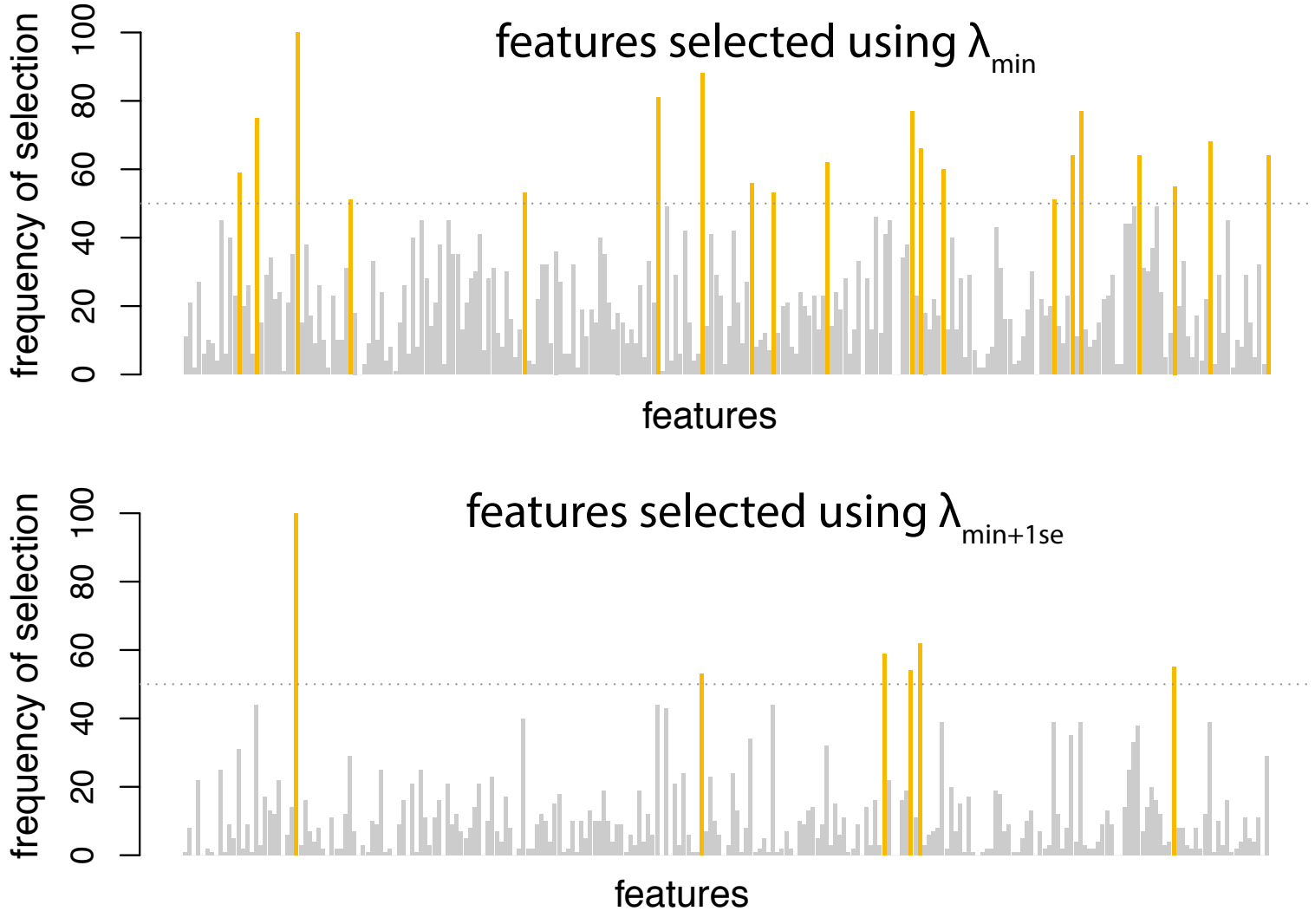
Lasso for feature selection

Selected features can be sensitive to the partition of the cross-validation.

To select only robust features we can use *bootstrap*: feature selection procedure is repeated M times, using every time a different bootstrapped dataset (i.e. sampling N samples with replacement).

Lasso for feature selection: using bootstrap

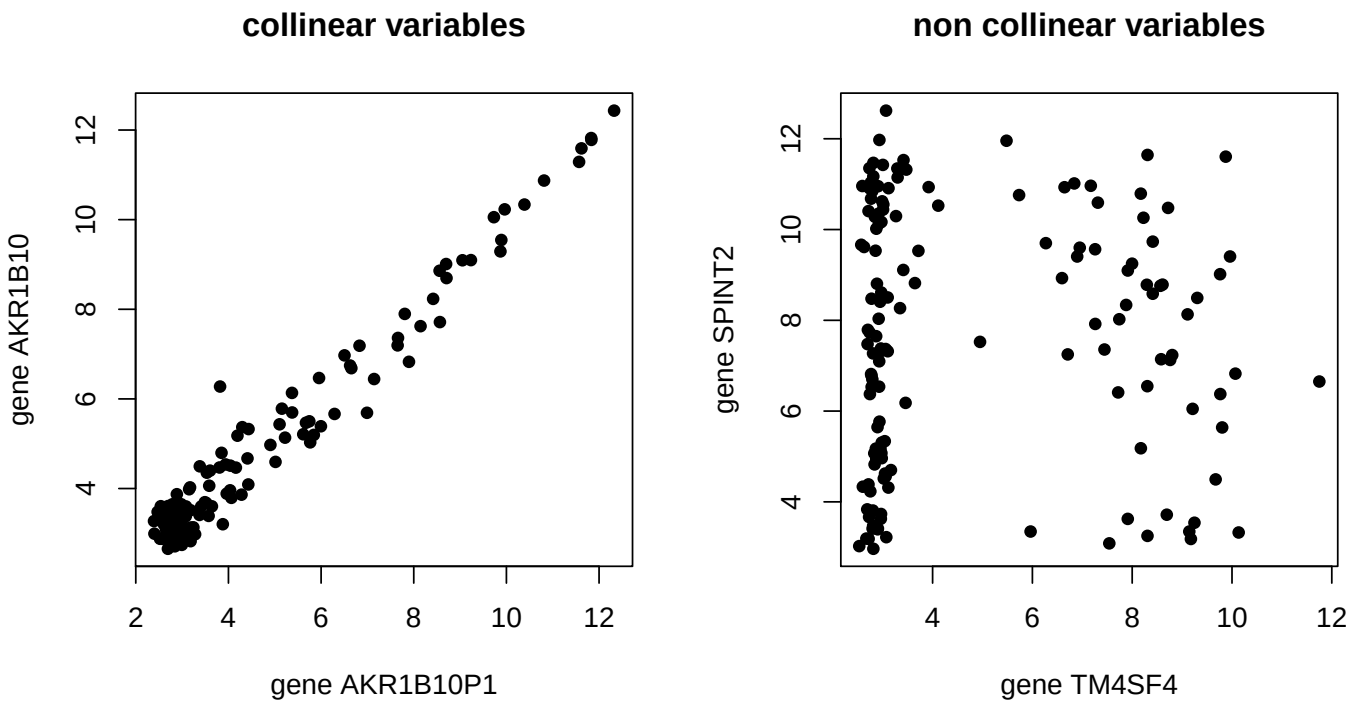
GDSC example: 100 times bootstrap (with 10-fold cross-validation)



Collinearity

Two or more predictor variables can be closely related to each other and show a high correlation.

This is common in genomic data, because genes can be redundant or have related functions.



Collinearity

The effect of *collinear variables* can be difficult to separate in the regression context.

Collinearity in general reduces the accuracy of the estimates of the coefficients of the correlated variables.

When using Lasso, *collinearity* can result in an arbitrary selection of one or the other correlated feature, depending on the data used for training.

Elastic Net regression

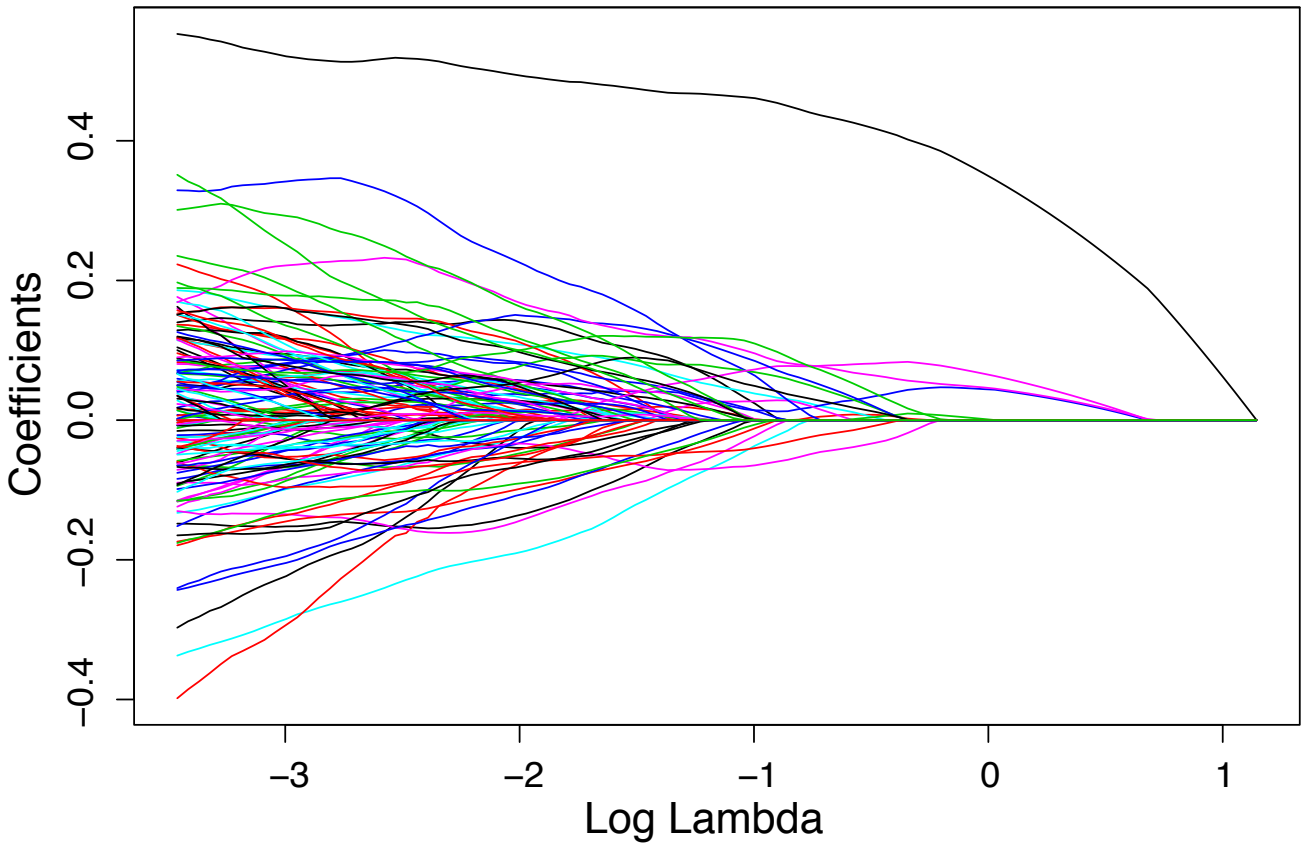
Elastic Net provides a compromise between Ridge and Lasso, by selecting variables like Lasso and shrinking together correlated variables like Elastic Net.

$$\hat{\beta}^{elasticnet} = \underset{\beta}{\text{argmin}} \left\{ RSS + \lambda \sum_{j=1}^{p} ((\alpha\beta_j^2 + (1-\alpha)|\beta_j|) \right\}$$

Where α is a tunable parameter. This corresponds to:

- Ridge for $\alpha = 1$
- Lasso for $\alpha = 0$

Elastic Net regression

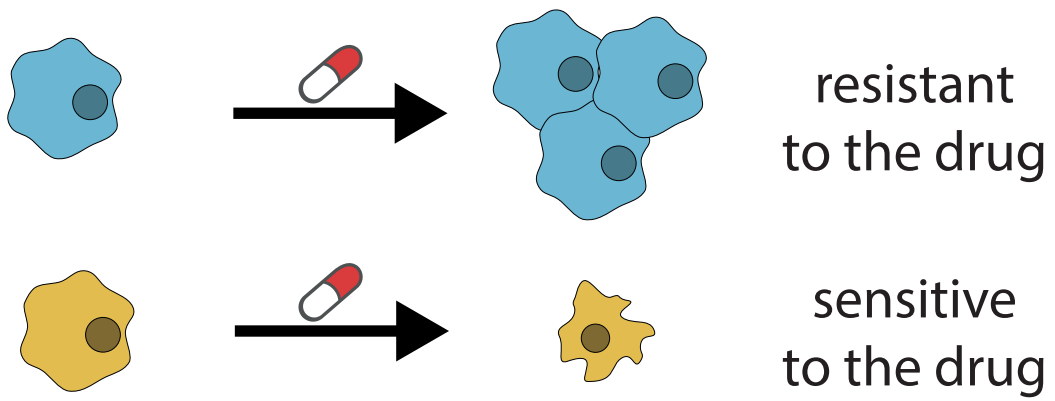


Profile of the Elastic net (with $\alpha = 0.5$) regression coefficients for the GDSC example.

Classification

In a clinical setting, instead of predicting the continuous IC50 for each patient, we might be interested in classifying patients in sensitive or resistant to a specific drug.

Using the cell lines from the GDSC study this corresponds to divide them in sensitive and resistant to the drug.

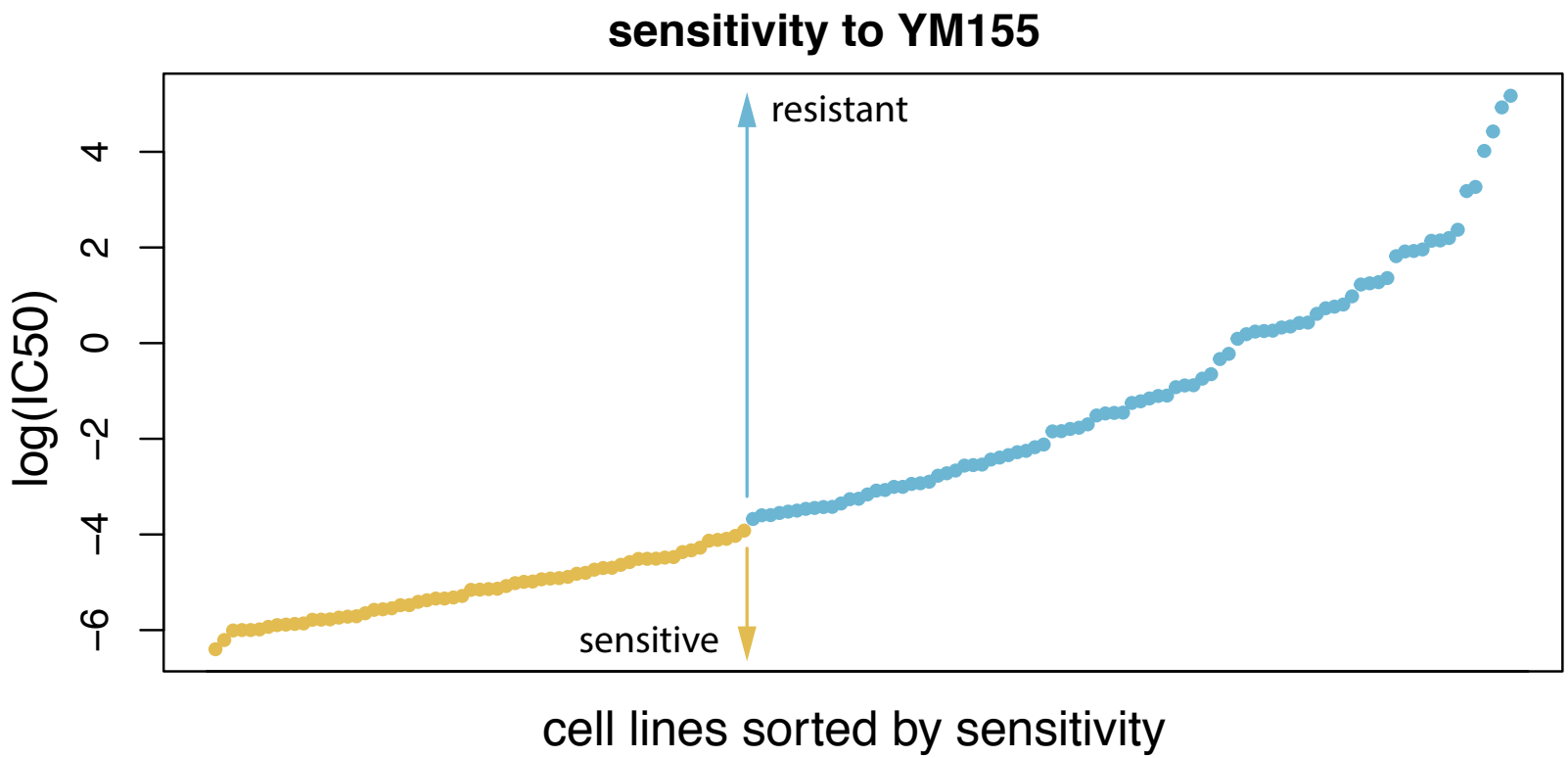


Cell lines are separated in sensitive and resistant based on the average response across all cell lines in the panel.

Classification

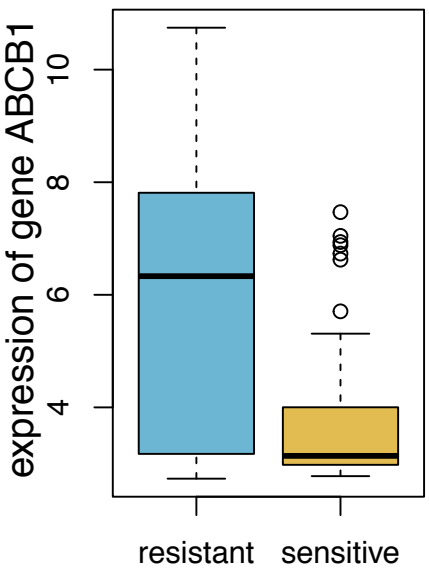
For binary classification we will consider a positive and a negative class:

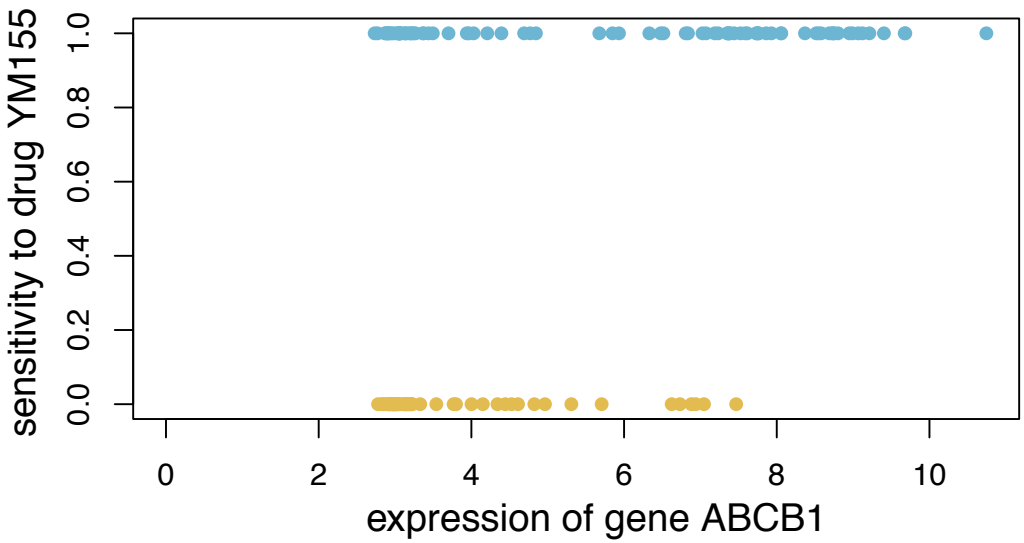
- 0 ("negative class"): sensitive
- 1 ("positive class"): resistant



Example of classification with one variable

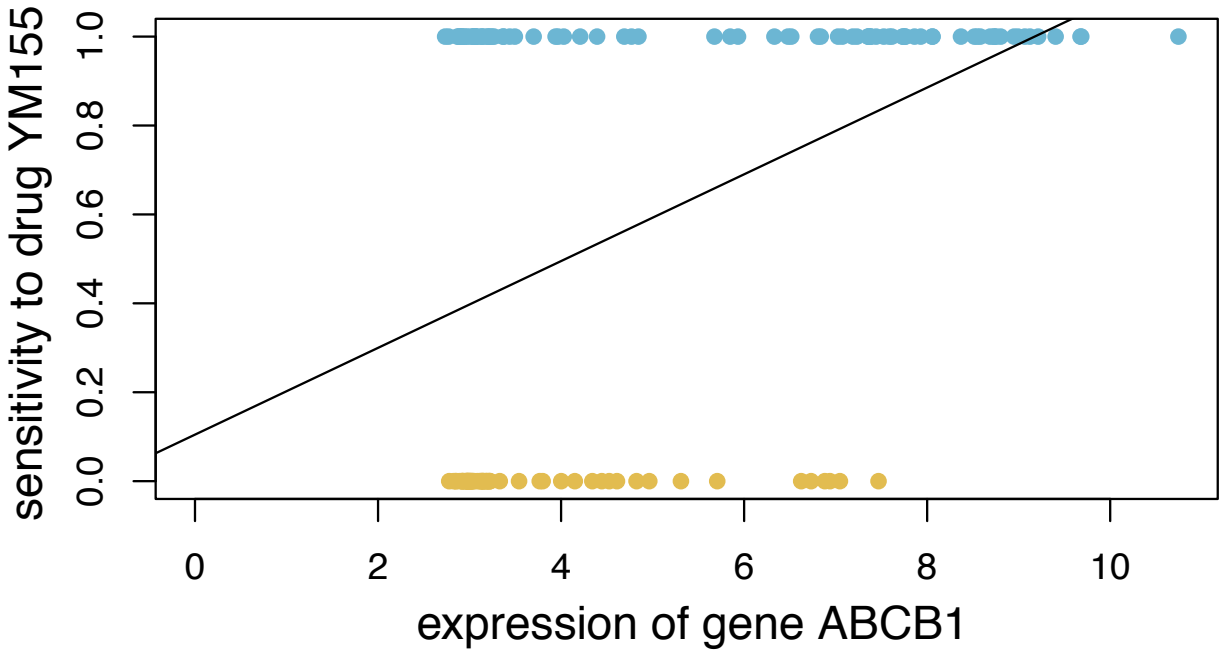
Using the same example used for univariate linear regression, we consider the ability of gene ABCB1 to classify cell lines in sensitive and resistant.





Example of classification with one variable

We can try to fit a linear regression to a binary response, e.g.using 0.5 as threshold for prediction.

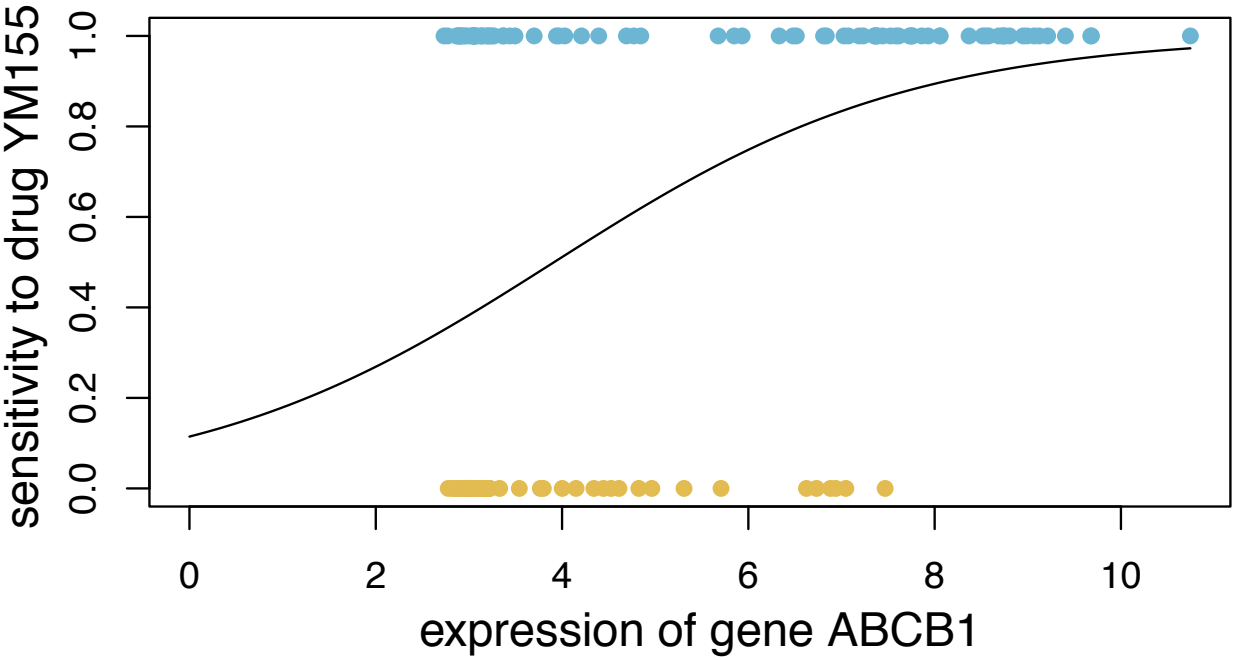


Poor interpretability, with predicted values also outside [0, 1].

Example of classification with one variable

With logistic regression we want to model the probability (between 0 and 1) that Y belongs to a particular class.

$$Pr(response = resistant | ABCB1 expression)$$



Logistic regression

$$p(X) = Pr(G = 1|X)$$

With *logistic regression* this probability is modeled as a *logistic function*:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

This corresponds to:

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

The left-hand side is called the log-odds or logit transformation and it is linear in X.

Logistic regression: multi-class classification

When having K classes, the posterior probability is modeled as:

$$\log \frac{Pr(G=1|X=x)}{Pr(G=K|X=x)} = \beta_{10} + \beta_1^T x$$
$$\log \frac{Pr(G=2|X=x)}{Pr(G=K|X=x)} = \beta_{20} + \beta_2^T x$$
$$\vdots$$
$$\log \frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

Using K-1 log-odds satisfies the constraint that probabilities sum to one. The choice of the last class to be used at denominator is arbitrary and does not affect the result.

Logistic regression: multi-class classification

It is easy to see that probabilities sum to one by rewriting the model as:

$$Pr(G = k|X = x) = \frac{e^{\beta_{k0}+\beta_k^T x}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_l^T x}}, k = 1, \ldots, K-1$$

$$Pr(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_l^T x}}$$

An alternative notation for the probabilities is $P(G = k|X = x) = p_k(x; \theta)$, to emphasize the dependence on the entire parameter set $\theta = \{\beta_{10}, \beta_1^T, \ldots, \beta_{(K-1)0}, \beta_{K-1}^T\}$.

Logistic regression: parameters estimation

The parameters for logistic regression models are generally estimated using *maximum-likelihood*.

For N observations the log-likelihood to be maximized is:

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(X_i; \theta)$$

where $p_k(x_i; \theta) = Pr(G = k | X = x_i; \theta)$.

Logistic regression: parameters estimation

For the two classes case the log-likelihood can be written

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta))\} \\ &= \sum_{i=1}^{N} \{y_i \beta^T x_i - \log(1+e^{\beta^T x_i})\}\end{aligned}$$

where $\beta = \{\beta_{10}, \beta_1\}$ and vector of inputs x_i includes the constant term 1 for the intercept.

Multiple logistic regression

When we have multiple predictors

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

where $X = (X_1, \ldots, X_p)$ are the p predictors. Or with the equivalent form

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

L1 regularized logistic regression

Similarly to what we have seen for linear regression, regularization can be used for variable selection and shrinkage also with logistic regression.

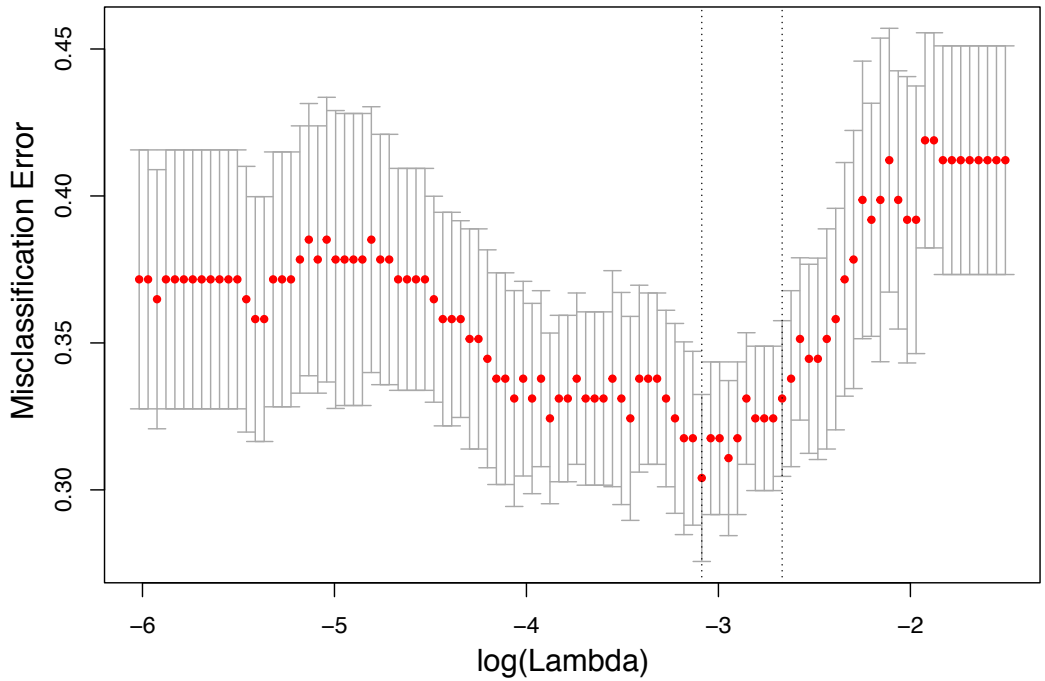
When applying L1 penalty we want to maximise a penalized version of the log-likelihood

$$\max_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} [y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{(\beta_0+\beta^T x_i)})] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Note that the intercept term is not penalized. Similarly to Lasso, the predictors should be standardized in order for the coefficients to be comparable.

Example cross-validation

Also in this case cross-validation can be used to select the tuning parameter.



References


Friedman, J., T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics New York, 2001.


Iorio, F. et al. "A landscape of pharmacogenomic interactions in cancer". In: *Cell* 166.3 (2016), pp. 740–754.