SF2935: Modern methods of statistical learning
Lecture 13
Resampling methods: Cross-Validation. Linear model selection and regularization

Tatjana Pavlenko

28 November 2016

Performance of a learning method naturally relates to its prediction accuracy on *independent* test data. We discuss *Cross-Validation*, CV, a simple and very efficient method for estimating prediction error. Initially derived by Stone (1974), but there are several versions.

- Let Y be a quantitative response variable, $\mathbf{X}$ a vector of feature variables/predictors, and a prediction model $\widehat{f}(\mathbf{X})$, which has been estimated using the training data $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- The common loss function for measuring errors (discrepancies) between Y and $\widehat{f}(\mathbf{X})$ is the squared error

$$L(Y, \widehat{f}(\mathbf{X})) = (Y - \widehat{f}(\mathbf{X}))^2.$$

- We will be considering various types of estimation of the loss.

Performance assessment: Test and training errors.

- *Test error*, also called for *generalization* error, is defined as

$$\mathrm{Err}_{\mathcal{T}} = \mathbb{E}\left(L(Y, \widehat{f}(\mathbf{X}))|\mathcal{T}\right).$$

  - $(\mathbf{X}, Y)$ is *new*, an independent (on $\mathcal{T}$) data drawn randomly from the joint population distribution of $\mathbf{X}$ and Y, assumed to be the same as for $\mathcal{T}$).
  - The training set $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is fixed.

- *Expected test error*, also known as expected prediction error, is

$$\mathrm{Err} = \mathbb{E}\left[\mathbb{E}\left(L(Y, \widehat{f}(\mathbf{X}))|\mathcal{T}\right)\right],$$

  where the expectation averages over all the randomness, including that in the training set $\mathcal{T}$ which produces the estimated model $\widehat{f}$.

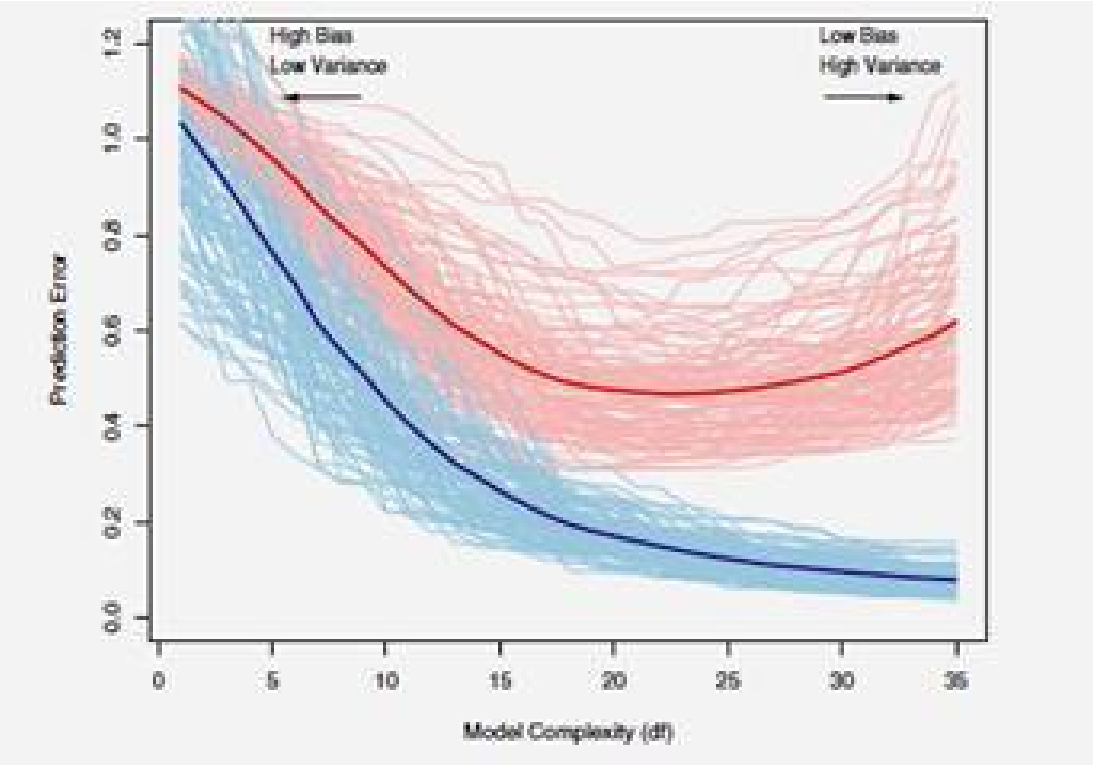
Performance assessment: Test and training errors (cont).

- *Training error* is also known as *apparent* or *re-substitution* error rate, is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \widehat{f}(\mathbf{x}_i)).$$

  - Is the most natural estimate of the error rate and shows how well we predict those *same* members of $\mathcal{T}$.
  - For almost all learning procedures, $\overline{\text{err}}$ is readily available after training.
  - Have misleadingly optimistic value since it estimates the predictive ability of the fitted model from the same $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ that was uses to fit the model! But we are interested in the accuracy of the prediction which is obtained from applying the learned/fitted model to previously unseen test data.
  - Generally, $\overline{\text{err}} \ll \text{Err}_{\mathcal{T}}$.
- This phenomena is presented on the figure below.

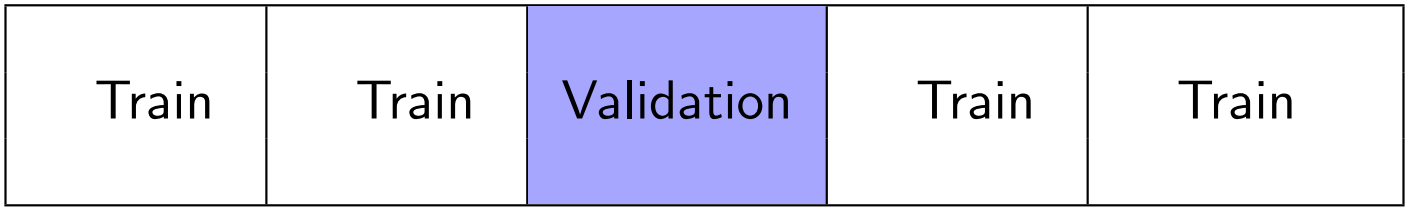
Training- vs test-performance with varying model complexity



Figur: Behavior of the test and training error as a function of model complexity. Light blue curves represent a set of training errors. Solid blue curve represents the expected training error. Light red curves represent conditional test errors. Solid red curve represents the expected test error.

K-Fold Cross-Validation: Generic idea

- Split randomly the set of available data into two parts: a *training set* and a *validation* or *hold-out set*
- For example, split into $K = 5$ roughly equal-size parts:



- Leave out part k (third above), fit model to the other $K - 1$ parts (combined together), specify the prediction error of the model fit when predicting the left-out kth part of the data.
- Perform this for $k = 1, 2, \ldots, K$ and combine the K estimates (usually by averaging over splits) into validation-set error which provides an estimate of the *test error*.
- The beauty of CV is its simplicity and generality:

  it allows for estimation of the test error rate by using (re-using) the training data!

K-Fold Cross-Validation: More precisely

- Let $\kappa : \{1, \ldots, n\} \to \{1, \ldots, K\}$ denote the indexing function that indicates the partition to which observation i is allocated by the randomization.
- Let $\widehat{f}^{-k}(\mathbf{x})$ denote the function fitted with the kth part of the data removed.
- Then the *cross-validation* estimate of the prediction error is given by

$$\mathrm{CV}_{(K)} = \frac{1}{K} \sum_{k=1}^{K} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_{\kappa(i)}} L(y_i, \widehat{f}^{-\kappa(i)}(\mathbf{x}_i)).$$

- Usual choices for K are 5 or 10.
- The special case of $K = n$ is known as *leave-one-out* cross-validation, LOOCV. In this case, $\kappa(i) = i$, and for the ith observation the fit is computed using all the data exept the ith.

K-Fold Cross-Validation: More precisely

- Given a set of models $f(x, \alpha)$ indexed by a tuning parameter α, denote by $\widehat{f}^{-k}(\mathbf{x}, \alpha)$ the αth model fit with the kth part of the data removed.
- For this set of the models define

$$\mathrm{CV}_{(K)}(\alpha) = \frac{1}{K} \sum_{k=1}^{K} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_{\kappa(i)}} L(y_i, \widehat{f}^{-\kappa(i)}(\mathbf{x}_i, \alpha)).$$

- The function $\mathrm{CV}(\alpha)$ provides an estimate of the test error curve; $\hat{\alpha}$, a minimizer of it, is the the (CV) optimal value of the tuning parameter.
- The final chosen model is $f(x, \hat{\alpha})$ which we then fit to *all the data*.

K-Fold Cross-Validation: MSE loss

- Let the K parts be $C_1, \ldots, C_k$ where C_k is the set of indices of the observations in part k, there are n_k obs. in part k.
- Assume that n is a multiple of K, so that $n_k = n/K$ and consider linear fitting under squared-error loss

$$L(\mathbf{y}, \widehat{f}^{-\kappa(i)}(\mathbf{x})) = \text{MSE}^{-k} = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i)^2,$$

where $\hat{y}_i$ is the fit for observation i obtained from the data with part k removed. Then

$$\text{CV}(\widehat{f}) = \sum_{k=1}^{K} \frac{n_k}{n} \text{MSE}^{-k}$$

- Setting $K = n$ yields n-fold of leave-one out, LOOCV.

CV ON CLASSIFICATION PROBLEM

- Let Y be a qualitative response variable and $\mathbf{X}$ a vector of feature variables/predictors. This corresponds to the classification model with $Y \in \mathcal{C} = \{1, \ldots, c\}$ given classes. The training data set $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. In the classification setting we then assign an observation $\mathbf{x}$ to the class c for which

$$\hat{y}(\mathbf{x}) = \hat{y}_c = \arg\max_{c \in \mathcal{C}} \mathbb{P}(Y = c | \mathbf{X} = \mathbf{x}).$$

- 0-1 loss function, $L(Y, \widehat{Y}(\mathbf{X})) = I(Y \neq \widehat{Y}(\mathbf{X}))$, can be used to quantifying the accuracy of the classifier.
- Then the *training* error is $\overline{\text{err}} = \frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$ with averaging over $\mathcal{T}$,
- and the *test* error is $\text{Err}_{\mathcal{T}} = E(L(Y, \widehat{Y}(\mathbf{X})|\mathcal{T})$.
- The test error associated with a *new* set of observations (independent on $\mathcal{T}$!) $\mathcal{T}_{new} = \{(\mathbf{x}_{new}, y_{new})\}$ is estimated by

$$\text{Err} = \text{Ave}(\text{Err}_i) \quad \text{where} \quad \text{Err}_i = I(y_i \neq \hat{y}_i) \text{ for } \mathcal{T}_{new}.$$

CV ON CLASSIFICATION PROBLEM (CONT.)

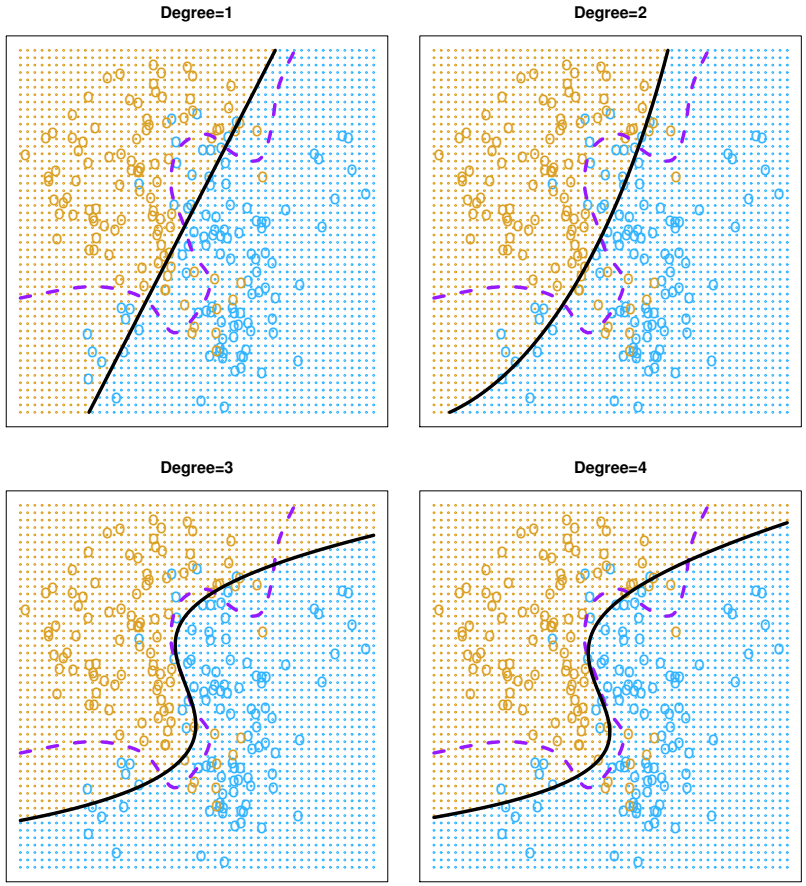
- K-**fold CV**. Split the data $\mathcal{T}$ randomly into K parts, $\mathcal{T}_{C_1}, \ldots, \mathcal{T}_{C_K}$ where C_k is the set of indices of the observations in part k and let $n_k = n/K$.
  - For each k compute $\text{Err}_k = \sum_{i \in \mathcal{T}_{C_k}} I(y_i \neq \hat{y}_i)/n_k$, n_k is the number of obs. in $\mathcal{T}_{C_k}$.
  - Compute $\text{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \text{Err}_k$.
- Example from ISL: Logistic regression with a non-linear decision boundary.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_2 X_2^2.$$

- For the simulated data, we can compute the *true* test error rate which is 0.201 and use the Bayes error rate, which is 0.133 as a benchmark. For real data, Bayes decision boundary and test error rates are unknown.
  Q: How to choose between the four logistic models?

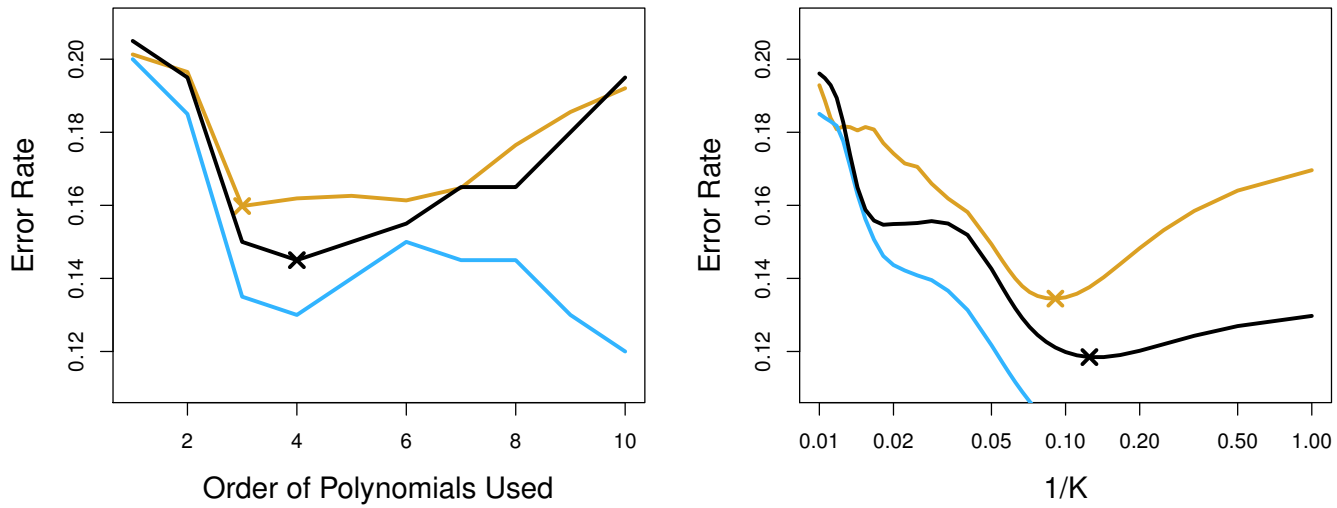
  A: use CV to make decision.

CV FOR CLASSIFICATION USING LOGISTIC REGRESSION



FIGUR: Simulated two-class bivariate data along with the logistic regression fit and Bayesian decision boundary (purple dashed line). Estimated classification boundaries from polynomial (degrees 1 to 4) logistic regression with the test errors estimated as $0.201, 0.197, 0.160$ and 0.162, respectively for each polynomial function. Bayes (irreducible) error rate is 0.133.

CV FOR CLASSIFICATION USING LOGISTIC REGRESSION (CONT.)



FIGUR: Test error (brown), training error (blue) and 10-fold CV error (black). Left: polynomial logistic regression based classifier. Right: KNN-classifier (K here is the number of neighbors, not a number of folds).

Right and wrong way to perform cross-validation

Example: Consider a two-class classification problem. Assume that the number of feature variables is $p = 5000$ and total sample size is $n = 50$, i.e. need to construct a classifier for a very high-dimensional case.

One possible strategy is to first select a subset of informative variables and then validate the performance of the resulting classifier, we call it for

Strategy I:

1. Screen the feature variables for their discriminative power. Select, say, 100 variables having largest correlation with the class label.
2. Construct a classifier based on the selected subset of highly discriminative features.
3. Perform cross-validation to estimate the unknown tuning parameters and to estimate Err of the final classifier.

Q: What is wrong with Strategy I ?

Answer: The discriminative feature variables were chosen after seeing all the data!

Correct way is **Strategy II:**

1. Split the samples into K folds randomly.
2. For each fold $k = 1, \ldots, K$
   - Find a subset of discriminative feature variables using all the samples except the kth.
   - Construct the classifier using all the samples except the kth. Apply the classifier to predict the class labels for the samples in kth fold.
   - Select the final model which minimizes CV error.

General approach to performing multi-step model fitting with CV:

- CV must be applied to the **entire sequence of model fitting steps**.
- Any supervised procedure which has seen the labels of the training data and used them, is a form of *training* and must be included in the validation procedure.

One important special case of LOOCV!

- LOOCV has a potential to be computationally expensive to implement since the model must be re-fit n times. For example, 10-fold CV requires model fitting only 10 times which might be much more feasible.
- Generalized cross-validation (GCV) provides a convenient approximation to LOOCV, **only for linear fitting under squared-error loss**.

One important special case of LOOCV (cont.)

- **Idea:** For the linear model fitting methods, we have $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$ where

$$\mathbf{H} = \mathbf{X}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}$$

is known as a *hat* matrix We then can express the LOOCV model fit error as

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}^{-i}(\mathbf{x}_i)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(\mathbf{x}_i)}{1 - H_{ii}}\right]^2,$$

where H_{ii} is the diagonal element of $\mathbf{H}$.

- Then the GCV approximation to $\mathrm{CV}_{(n)}$ is

$$\mathrm{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(\mathbf{x}_i)}{1 - \mathrm{trace}(\mathbf{H})/n}\right]^2.$$

- $\mathrm{trace}(\mathbf{H})$ is the effective number of parameters in the linear model.
- **Obs!** The GCV approximation to $\mathrm{CV}_{(n)}$ does not hold in general, i.e. for other types of models.

Linear model selection: Regularization-based approach

Recall that

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon.$$

Motivation to considering alternatives to ordinary least squares (OLS):

- *Prediction accuracy*, especially when $p > n$.
- *Model interpretability*: discarding irrelevant feature variables produces the model which is interpretable and has possibly lower prediction error than the full model. Corresponding methods are known as *Subset Selection*.

Subset selection is a discrete process: variables are either retained or discarded, which can exhibit high variance. *Shrinkage methods* provide a kind of *continuous* subset selection; the model involving all p feature variables is fitted but the estimated coefficients are *constrained* or regularized, or equivalently, shrunken towards zero relative to OLS estimators.

The shrinkage/regularization can significantly reduce the variance of the coefficient estimators and can also perform variable selection.

Ridge-regression

Ridge regression shrinks the regression coefficients by imposing a *penalty* of their size. The ridge coefficients minimize the RSS,

$$\widehat{\boldsymbol{\beta}}^{\text{Ridge}} = \arg\min_{\beta \in R^p} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where $\lambda \geq 0$ is a tuning parameter that controls the amount of shrinkage. The coefficients are shrunk towards zero and each other.

- $\lambda \sum_{j=1}^{p} \beta_j^2$ is a *shrinkage penalty* is small when $\beta_1, \ldots, \beta_p$ are close to zero, has effect of shrinking estimators of β_j to 0.
- For $\lambda = 0$ we get OLS.
- Unlike OLS the ridge-regression generates a *path of solutions*, $\widehat{\boldsymbol{\beta}}^{\text{Ridge}}(\lambda)$. Adaptive selection of the optimal λ to minimize an estimate of the expected prediction error.

An equivalent way to express the ridge regression problem is the constraint minimization of RSS:

$$\widehat{\beta}^{\text{Ridge}} = \arg\min_{\beta \in R^p} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq s,$$

which makes explicit the size constraint on the parameters, one-to-one correspondence between λ and s.

Ridge-regression (cont.)

Technical details of using ridge-regression.

- Reparametrization using *centered* inputs: x_{ij} gets replaced with $x_{ij} - \bar{x}_j$.
- We estimate β_0 by $\bar{y} = \sum_{i=1}^{n} y_i / n$ i.e left it out from the penalty term.
- Scaling of predictor variables. In OLS, the coefficient estimators are scale invariant: multiplying X_j by a constant κ leads to the factor $1/\kappa$ in the estimator of β, i.e $X_j\hat{\beta}_j$ remains the same. The ridge regression coefficients can change when multiplying a predictor variable by a constant, this is due to the sum of squared coefficients term in the penalty part of the ridge objective. It is best to apply ridge-regression *after* standardizing the feature variables using

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_{ij})^2}}.$$

Ridge-regression (cont.)

- In the matrix form,

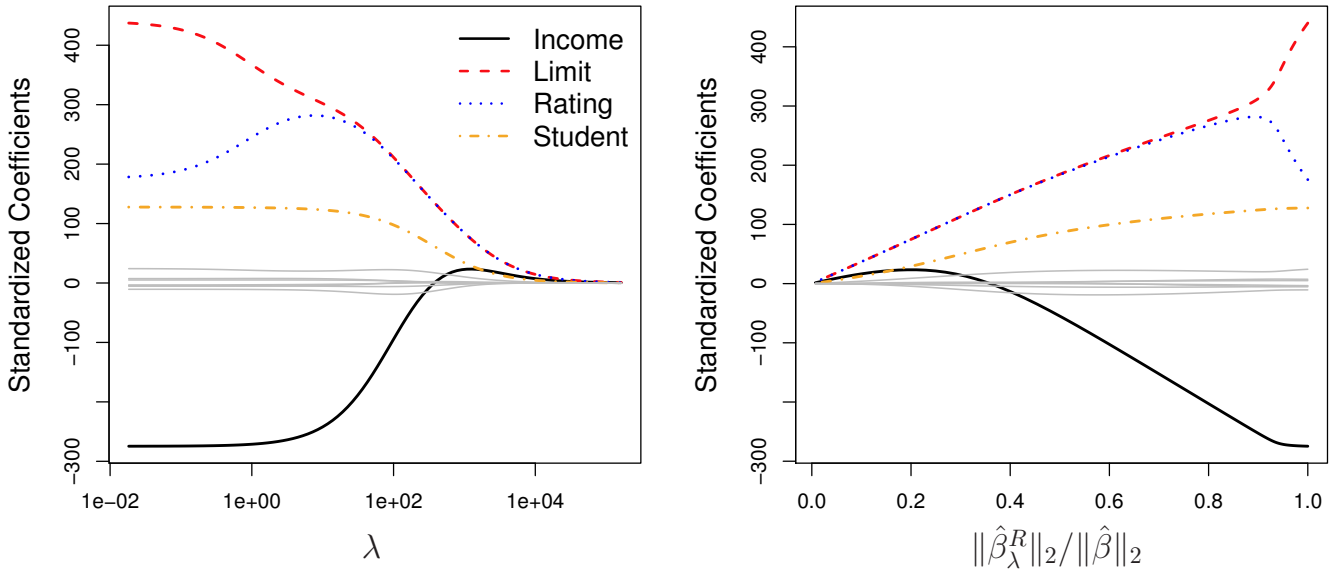
$$\mathrm{RSS}(\lambda) = (\mathbf{y} - \boldsymbol{X}\boldsymbol{\beta})' (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}' \boldsymbol{\beta},$$

  the ridge regression solution is

$$\widehat{\boldsymbol{\beta}}^{\mathrm{Ridge}} = \left(\mathbf{X}'\mathbf{X} + \lambda I\right)^{-1} \mathbf{X}'\mathbf{y}.$$

- Adding a positive constant to the diagonal of $\mathbf{X}'\mathbf{X}$ makes the problem non-singular, even if $\mathbf{X}'\mathbf{X}$ is not of full rank. This was a main motivation to ridge-regression when it was firstly introduced by Hoerl and Kennard (1970).

RIDGE REGRESSION COEFFICIENTS



FIGUR: Profiles of ridge coefficients for the `Credit+` data, as tuning parameter λ is varied.

Lasso

The ℓ_2-penalty term $\sum_{j=1}^{p} \beta_j^2 = \|\boldsymbol{\beta}\|_{\ell_2}^2$ in the ridge regression shrinks the estimated β_j's toward zero but not set any of them to *exactly zero*.

- We consider the Lasso (Least Absolute Shrinkage and Selection Operator) that involves penalizing the absolute size of the regression coefficients and results in coefficient estimates that are exactly zero.
- Replace ℓ_2- with ℓ_1-penalty on $\boldsymbol{\beta}$ in ridge regression. The lasso coefficients are

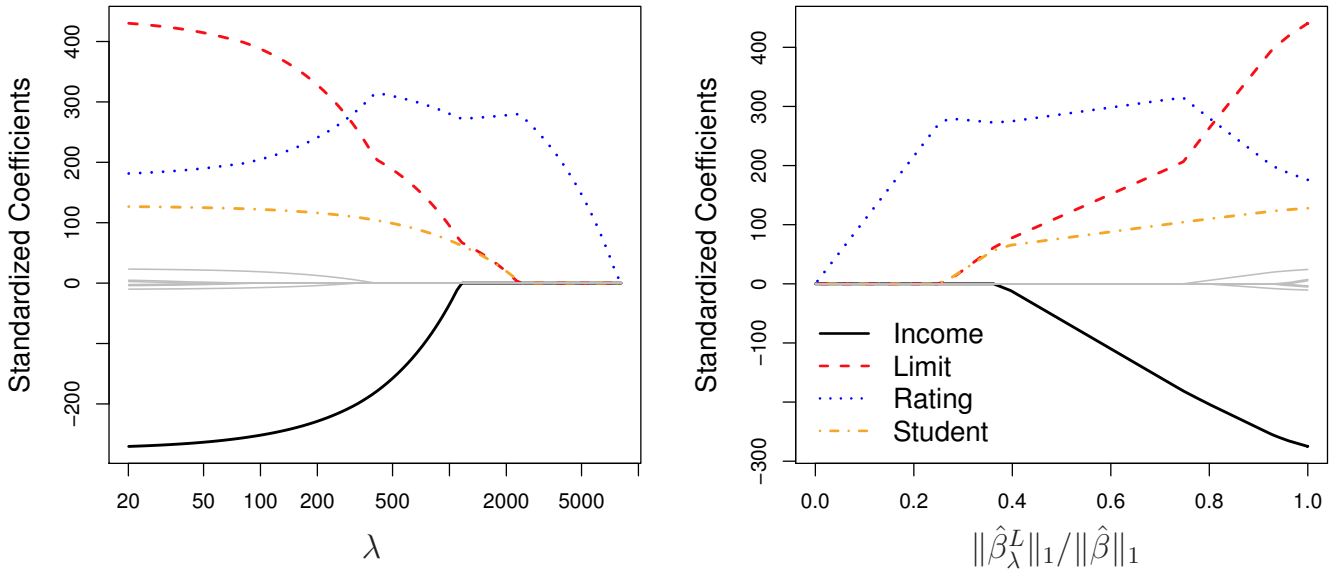
$$\widehat{\beta}^{\text{Lasso}} = \arg\min_{\beta \in R^p} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\},$$

- or in the alternative formulation

$$\widehat{\boldsymbol{\beta}}^{\text{Lasso}} = \arg\min_{\beta \in R^p} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

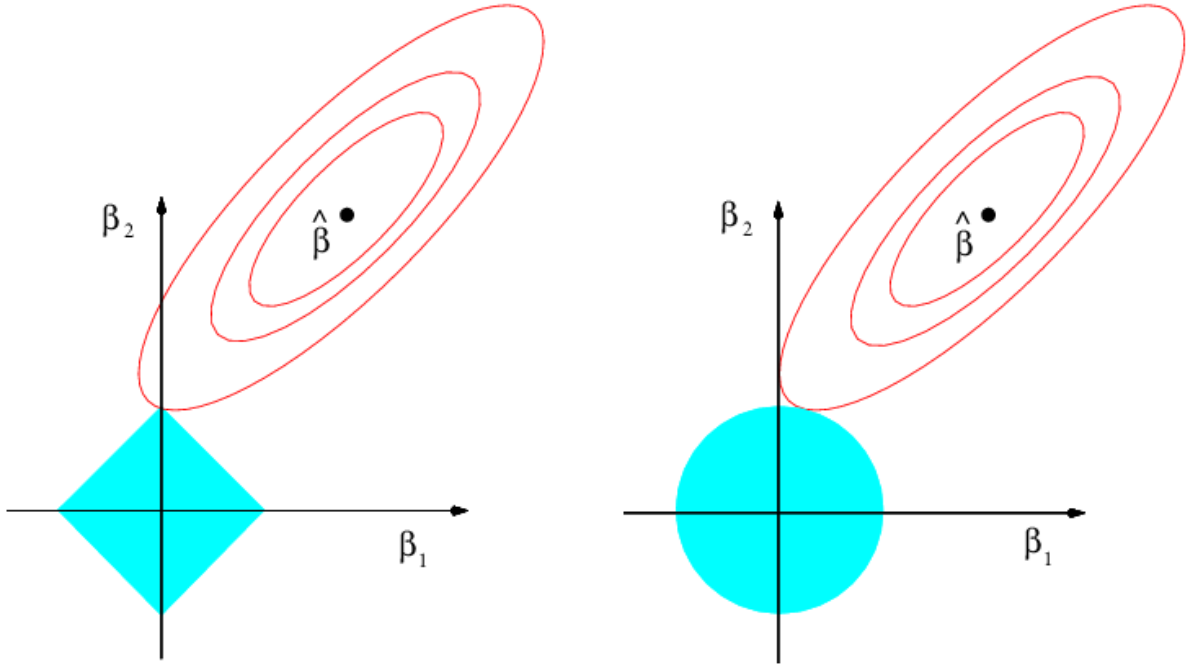
$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq s.$$

Lasso regression coefficients



Figur: Profiles of lasso coefficients for the `Credit+` data, as tuning parameter λ is varied. The lasso profiles hit zeros *exactly* (unlike the ridge coefficients) and yields a *sparse* solution - fitted models that involve only a subset of variables.

Variable selection property of the Lasso

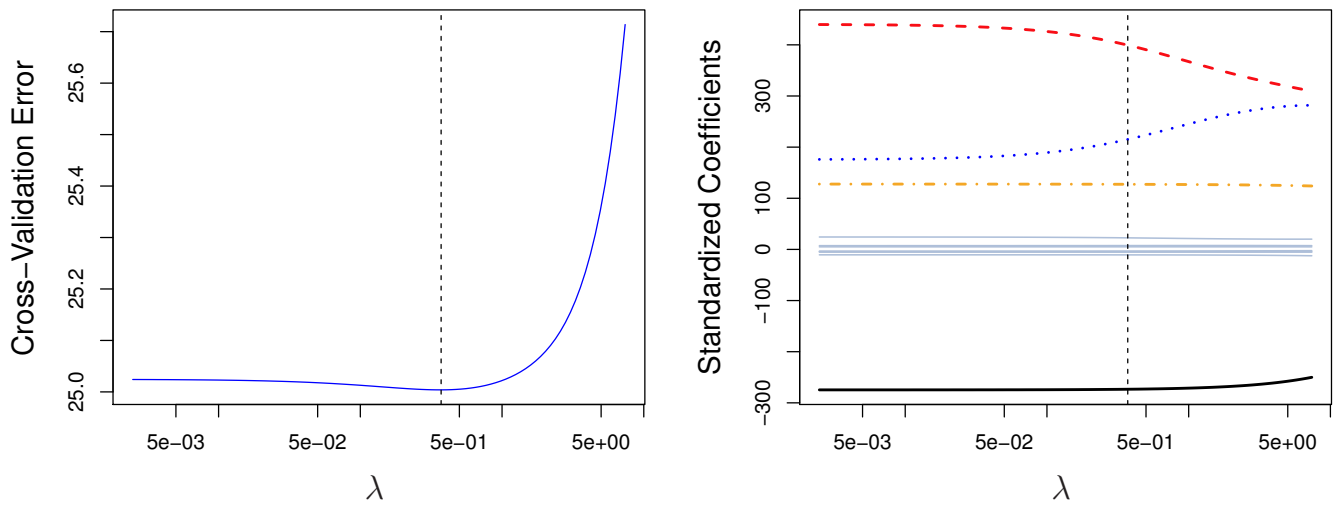


Figur: Contour plots of RSS and constraint functions for the Lasso (left, $|\beta_1| + |\beta_2| \leq s$) and ridge regression (right, $\beta_1^2 + \beta_2^2 \leq s$) for $p = 2$. Unlike the disk, the diamond has corners; if the solution occurs at a corner then it has one parameter β_j equal to zero. If $p > 2$ the diamond becomes a rhomboid, has many corners, flat edges and faces, there are many opportunities for the estimated parameters to be zero.

Cross-validation (CV) strategy:

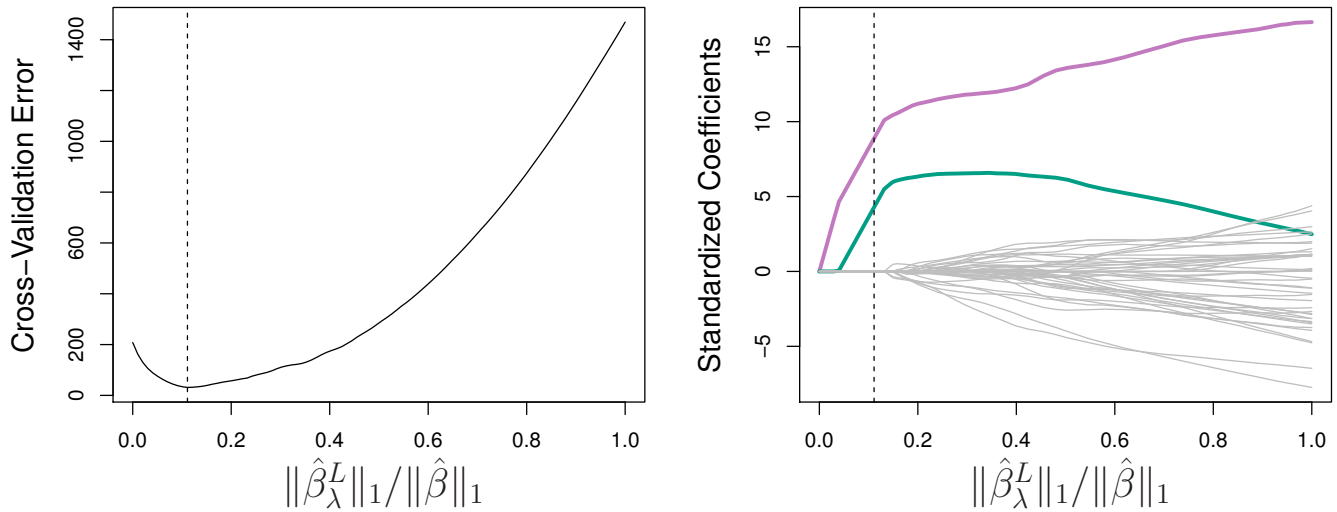
- Choose a grid of λ values and compute the CV/LOOCV error rate for each value of λ.
- Select the tuning parameter value for which the CV error is smallest. Use the *Min-Min* rule (Guo et al (2007)) if the minimum is attained at several λ's: select the minimum of these λ's. (The flat CV curve around minima, see the right panel of the figure above). The final choice of λ is $\hat{\lambda} = \min_{\lambda} CV(\lambda)$.
- Re-fit the model using all the available observations and the selected value of the tuning parameter $\hat{\lambda}$.

Selecting λ in ridge regression



Figur: Left: LOOCV for ridge regression applied to `Credit+` data. Right: The coefficient estimates as a function of λ, the value of λ (- -) selected by CV. In cases like this (very small λ) the OLS can be used.

Selecting tuning parameter in Lasso



Figur: Left: Ten-fold CV for MSE of Lasso applied to the sparse simulated data (two feature variables out of $p = 45$ are related to the response variable, $n = 45$. Right: The coefficient estimates as a function of λ, the value of λ (- -) selected by CV. Two colored lines represent the predictor variables that are related to the response, the gray lines represent the unrelated predictors; these are referred to as *signal* and *noise* variables, respectively.

Bayesian interpretation of the ridge regression and Lasso

General idea: Bayesian learning is based on using a *prior distribution* to express the uncertainty *before seeing the data*, and to allow the uncertainty remaining *after seeing the data* to be expressed in the form of *posterior distribution*.

Bayesian interpretation of the ridge regression and Lasso

- The linear model in vector form (X_i's are fixed and standardized)

$$\mathbf{Y} = \mu \mathbf{1}_n + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, .$$

  $\mathbf{Y}$ is $n \times 1$, $\mathbf{X}$ is $n \times p$ and $\boldsymbol{\varepsilon}$ is $n \times 1$ with mean zero and σ^2.

- Set $p(\boldsymbol{\beta}) = \prod_{j=1}^{p} g(\beta_j)$ for some density $g(\cdot)$. By Bayes' thm

$$p(\boldsymbol{\beta}|\boldsymbol{Y}, \mathbf{X}) \propto f(\boldsymbol{Y}|X, \boldsymbol{\beta}) p(\boldsymbol{\beta}|\mathbf{X}) = f(Y|\mathbf{X}, \boldsymbol{\beta}) p(\boldsymbol{\beta}).$$

- Let $g(\cdot)$ be a double-exponential (also known as Laplace) prior

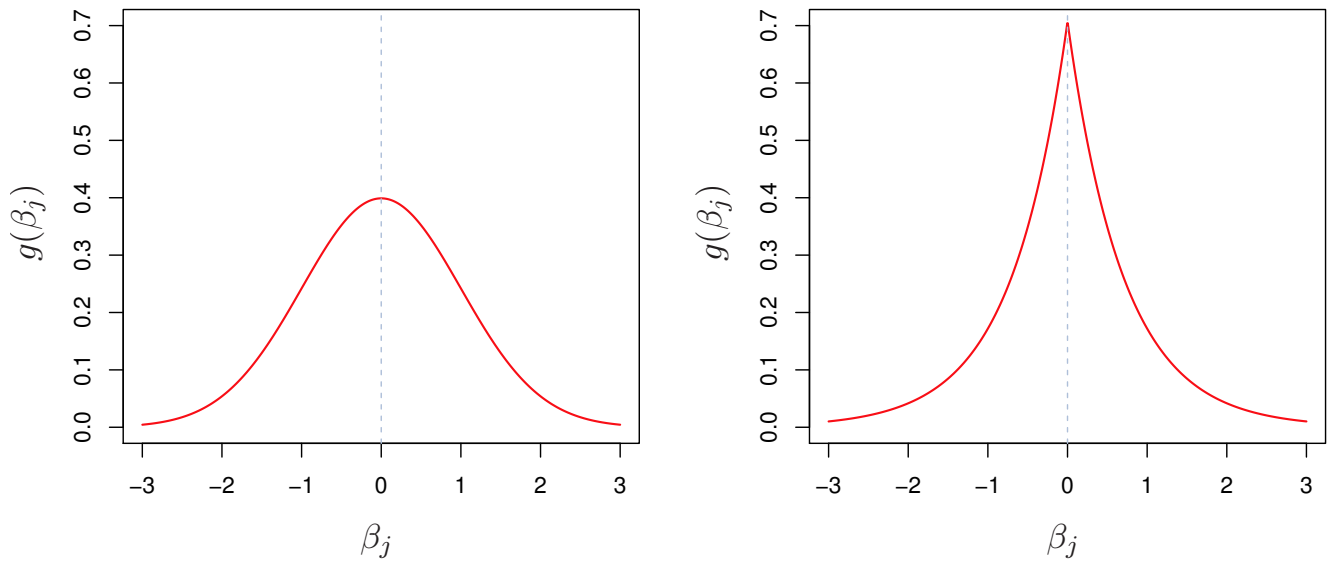
$$p(\boldsymbol{\beta}|\sigma^2) = \prod_{j=1}^{p} \frac{\lambda}{2\sqrt{\sigma^2}} e^{-\lambda|\beta_j|/\sqrt{\sigma^2}},$$

  where $\pi(\sigma^2) = 1/\sigma^2$ is non-informative prior on σ^2 imposed to ensure unimodality of the resulting posterior. See Park et al (2008).

  The posterior mode for $\boldsymbol{\beta}$ is then the Lasso solution.

Bayesian ridge and Lasso

By structure, the lasso prior is more is steeply peaked at zero, meaning that the lasso expects a priori many coefficients β_j to be zeros (sparse model). Ridge with Gaussian prior is more flat expecting that coefficients are randomly distributed around zero.



Figur: Left: Ridge regression is the posterior mode for β under Gaussian prior. Right: The Lasso is the posterior mode for β under a double-exponential prior.

Some references

1. **Sparse discriminant analysis using constrained ℓ_1-minimization in high dimensions, CV, LOOCV with applications.**
   - Cai T., Liu W. A direct estimation approach to sparse LDA. 2011 *J. of the American Statistical Association* Vol. 106, No. 496.
   - Izenman A. Modern multivariate statistical techniques. Regression, classification and manifold learning. Springer, 2008.

2. **Bayesian lasso and ridge, hierarchical prior modeling, sampling from posterior**
   - Park T., Casella G. The Bayesian Lasso. *J. of the American Statistical Association* 2008, Vol. 103, No. 482, Theory and Methods.

KTH Matematik