Regression Models

Linear models

- Linear models are an extremely important basic tool in mathematical finance, and in statistical machine learning.
- Most statistical machine learning methods (such as neural networks, regression trees, lasso and elastic-net, logistic classifiers, etc.) are direct generalizations of linear regression models (ie. they reduce to regression as a special case), or they use linear regression as an intermediate step.
- Time-series analysis often reduces to some sort of regression of values on lagged values from the same time series.
- It is impossible to overstate the importance of these techniques; every good statistical arbitrage researcher knows them.

- A large number of different kinds of regression models exist, both of a linear and a non-linear nature.
- They are defined by the task they share in common: to attempt to learn the relationship between a real-valued response

$$y \in \mathbb{R}$$

as a function of a p-dimensional input vector

$$\boldsymbol{x} \in \mathbb{R}^{p},$$

as if there were a (possibly non-linear function f), controlled by internal parameters θ, such that

$$\mathbb{E}[y \mid \boldsymbol{x}, \theta] = f(\boldsymbol{x}; \theta).$$

- Any model having broadly this structure can be called a *regression* model.

- In the linear case, the model has parameters $\sigma^2 > 0$ and $\beta \in \mathbb{R}^p$, collected into the full parameter vector

$$\theta = (\sigma^2, \beta_1, \ldots, \beta_p) \in \mathbb{R}^{p+1}$$

  and then, Conditional on $\boldsymbol{x}$ and on θ, the variable y is normal with mean $\beta \cdot \boldsymbol{x}$ and variance σ^2.
- In other words, the distribution

$$p(y \mid \boldsymbol{x}, \beta, \sigma)$$

  is Normal.
- However, before proceeding with a detailed analysis of the normal/linear case, we make some statements concerning the random process model for the predictors $\boldsymbol{x}$ which apply more generally.

- The data space in regression is, in principle, the space of all pairs

$$(y, \boldsymbol{x}) \in \mathbb{R} \times \mathbb{R}^p$$

and a Bayesian statistical model would then specify the full likelihood

$$p(y, \boldsymbol{x} \mid \theta)$$

and a prior $p_0(\theta)$.

- Once a data set has been specified, in the form

$$D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$$

  we can collect the values $\{y_i : i = 1, \ldots, n\}$ into a column vector $\boldsymbol{y}$ and similarly collect the vectors $\boldsymbol{x}_i$ into a matrix $\boldsymbol{X}$ of dimension $n \times p$.
- Hence, equivalently, $D = (\boldsymbol{y}, \boldsymbol{X})$.
- The assumption

$$p(\boldsymbol{X} \mid \theta) = p(\boldsymbol{X})$$

  is a modeling choice, but a reasonable one, since the role of θ is to model the dependence of y, the response, on attribute vectors $\boldsymbol{x}$, as predictors.

- The complete data likelihood is then

$$\begin{aligned} p(D \mid \theta) &= p(\boldsymbol{y}, \boldsymbol{X} \mid \theta) \\ &= p(\boldsymbol{y} \mid \boldsymbol{X}, \theta) \underbrace{p(\boldsymbol{X} \mid \theta)}_{=p(\boldsymbol{X})} \\ &= p(\boldsymbol{y} \mid \boldsymbol{X}, \theta) p(\boldsymbol{X}) \end{aligned}$$

- This tells us that if the goal is to perform inference concerning the parameters θ, it suffices to focus on the part of the likelihood that is conditional on $\boldsymbol{X}$:

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \theta)$$

- Intuitively, the assumption $p(\boldsymbol{X} \mid \theta) = p(\boldsymbol{X})$ means that θ was not involved in the random process that generated $\boldsymbol{X}$; consequently, realized samples $\boldsymbol{X}$ cannot help us to narrow down the range of parameter space where θ lives.

- More precisely, we can see that if we were performing maximum-likelihood estimation, we'd be maximizing

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}, \theta) + \log p(\boldsymbol{X})$$

  over θ, and the second term $\log p(\boldsymbol{X})$ would be a constant term with respect to that optimization.
- The same applies to maximum-a-posteriori optimization.

- More generally, any inference we can do on θ using the posterior will not depend on the details of the random process model for $\boldsymbol{X}$, because, again with $D = (\boldsymbol{y}, \boldsymbol{X})$,

$$\begin{aligned} p(\theta \mid D) &= \frac{p(D \mid \theta) p(\theta)}{p(D)} \\ &= \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \theta) p(\theta)}{\text{const}} \end{aligned}$$

  where the normalization constant is determined by setting the integral over θ to equal one.

- Thus, as long as our goal is inference about the parameters θ, and not about any assumed parameters hidden in the data-generating process for $\boldsymbol{X}$, then we are justified in treating the matrix $\boldsymbol{X}$ as known constants in all of the regression problems we will deal with in this course.

- Let us return to the Linear Gaussian case:

$$\boldsymbol{y} \mid \beta, \sigma^2, X \sim N(X\beta, \sigma^2 I). \tag{2.1}$$

- The model (2.1) is sometimes referred to as an *ordinary least squares (OLS)* model, although least squares actually refers to an expression appearing in the maximum likelihood estimator.
- For simplicity we will stop explicitly denoting every distribution as conditional on X, although that is implicitly what is going on.

- The model (2.1) was studied extensively in classical statistics.
- The full set of $p+1$ parameters is $\theta = (\beta, \sigma^2)$ and the likelihood function is

$$p(y \mid \theta) = (2\pi\sigma^2)^{-n/2} \exp\Big[-\frac{1}{2\sigma^2} \|y - X\beta\|^2 \Big] \qquad (2.2)$$

- Recall one can equivalently maximize the log-likelihood.

$$\log p(y \mid \theta) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \ell(\beta)$$

where

$$\ell(\beta) := \|y - X\beta\|^2$$

- Maximizing the log-likelihood is equivalent to minimizing the norm appearing in the second term,

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\, \ell(\beta) = \underset{\beta}{\operatorname{argmin}}\, \|y - X\beta\|^2. \tag{2.3}$$

- Note that $\operatorname{rank}(X) = \operatorname{rank}(X'X)$ (exercise).
- If X is full rank, then $X'X$ is full rank *and also square*, hence invertible.
- When $X'X$ is invertible, the minimization problem (2.3) has a unique solution which can be written down in closed form as

$$\hat{\beta} = (X'X)^{-1}X'y.$$

- If X has less than full rank, it is said to be *rank-deficient*.
- In the rank-deficient case, $X'X$ has no inverse, but we can still study the minimization problem (2.3).

- For any X, the objective function $\ell(\beta)$ is bounded below (by zero), and one may show that minimizers always exist, but the minimizer need not be unique in the rank-deficient case.
- One way to render the problem well-defined in the rank-deficient case is to introduce an L^2 penalty term:

$$\hat{\beta}_\lambda^{\text{ridge}} = \underset{\beta}{\text{argmin}}\{\|y - X\beta\|^2 + \lambda\|\beta\|^2\} \tag{2.4}$$

$$= (X'X + \lambda I)^{-1}X'y \quad \text{if} \quad \lambda > 0 \tag{2.5}$$

- This is called *ridge regression*.

- The estimator $\hat{\beta}_\lambda^{\text{ridge}}$ exists independent of any conditions on X.
- To see this, note that

$$\langle v, (X'X + \lambda I)v \rangle \geq \lambda \|v\|^2.$$

- It follows that $X'X + \lambda I$ is invertible for any $\lambda > 0$.
- Statistically, ridge regression is equivalent to imposing a prior on β, of the form

$$-\log \pi(\beta) = \text{const} + \lambda \|\beta\|^2.$$

- Interestingly, the limit as $\lambda \to 0$, from the right, of ridge regression also exists.
- Indeed we define the notation

$$X^+ = \lim_{\lambda \to 0+} (X'X + \lambda I)^{-1} X' \tag{2.6}$$

- From which we can easily deduce

$$\lim_{\lambda \to 0+} \hat{\beta}_\lambda^{\text{ridge}} = X^+ y$$

An ingenious idea due to Moore (1920) and Penrose (1955) is that whenever there exist multiple minimizers of (2.3), we can pick one out as having the smallest norm.

Definition 2.1

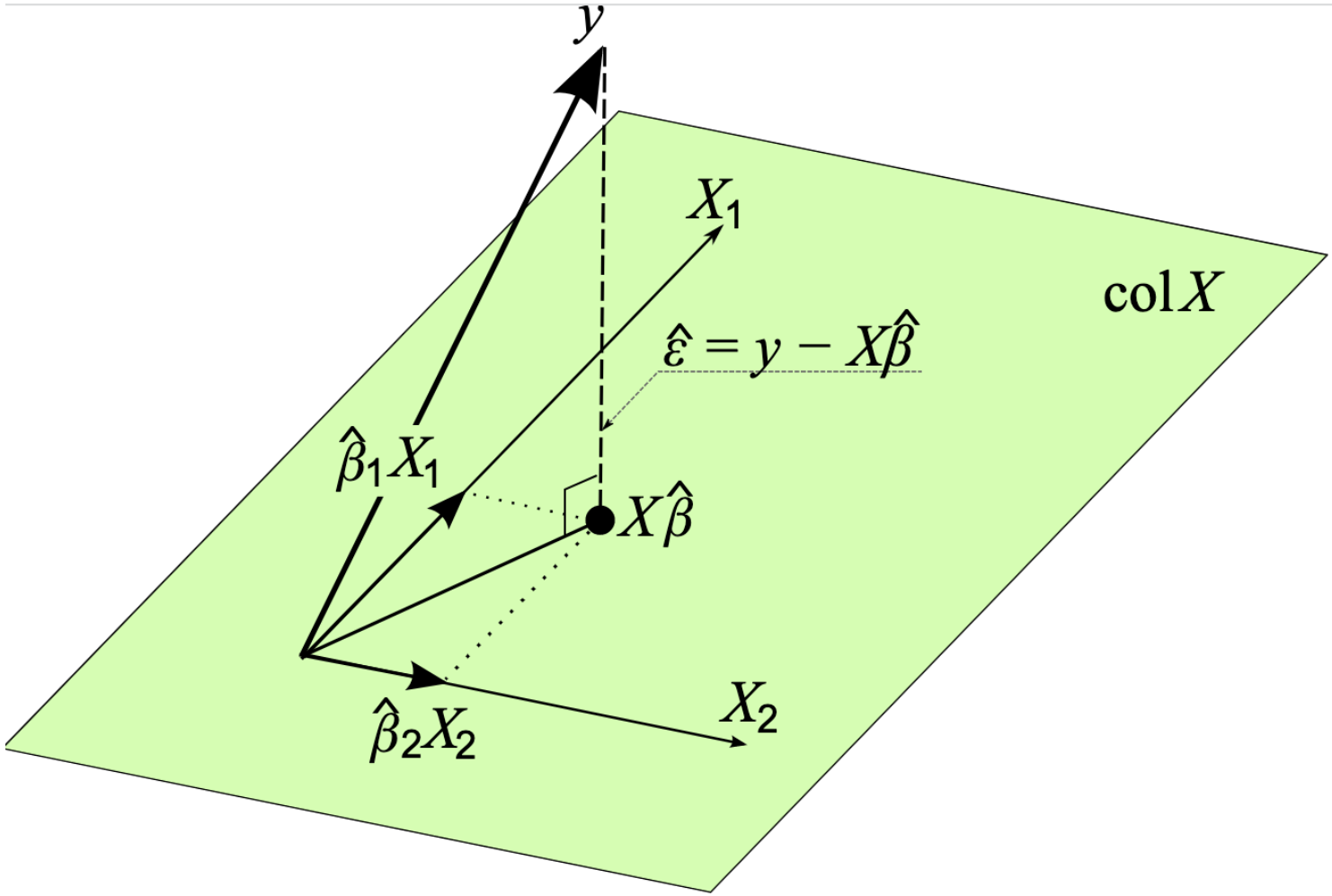
The Moore-Penrose pseudoinverse X^+ is defined so that X^+y is the minimum-norm vector among all minimizers of $\|y - X\beta\|^2$.

- This agrees with (2.6) and serves as an alternate definition of X^+; see Albert (1972).
- If there is only one minimizer for (2.3), then we say the regression is *identifiable*, and in the identifiable case, one can show that

$$X^+ = (X'X)^{-1}X',$$

but Definition 2.1 and (2.6) are well-defined for any real matrix X.

- In regression mathematics, it's quite useful to keep the geometric interpretations of the various quantities relatively near the front of one's mind.
- Here is an artist's interpretation of OLS as a projection onto the column span of X.


y
X_1
$\mathrm{col}X$
$\hat{\varepsilon} = y - X\hat{\beta}$
$\hat{\beta}_1 X_1$
$X\hat{\beta}$
$\hat{\beta}_2 X_2$
X_2

- In (2.3), note that

$$\hat{y} = X\hat{\beta} = XX^{+}y$$

  is the orthogonal projection of y on the linear subspace spanned by the columns of X.
- It follows that the vector of *fitted residuals*,

$$y - XX^{+}y,$$

  is the projection of y onto the so called *perpendicular space* $X^{\perp}$, the space orthogonal to the columns of X.
- These relations remain true whether or not $X'X$ is invertible, because the pseudo-inverse does not require invertibility.

Multiple Regression from Single Regression

- Consider the simplest possible regression: a univariate model with no intercept:

$$Y = X\beta + \epsilon$$

where X and Y are scalar-valued random variables.

- Given a list of observations

$$\{(x_i, y_i) : i = 1, \ldots, n\},$$

the least-squares estimates and the fitted residuals are, respectively,

$$\hat{\beta} = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}, \qquad \hat{\boldsymbol{\epsilon}} = \boldsymbol{y} - \boldsymbol{x}\hat{\beta}$$

- We can essentially build up multiple regression from repeated applications of these simple transformations.

- Suppose next that we have a multivariate regression, but the inputs

$$\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_p$$

- (the columns of the data matrix X) are pairwise orthogonal.
- Then it is easy to check that the multiple least squares estimates $\hat{\beta}_j$ are equal to

$$\frac{\langle \boldsymbol{x}_j, \boldsymbol{y} \rangle}{\langle \boldsymbol{x}_j, \boldsymbol{x}_j \rangle},$$

  which are the univariate estimates.
- In other words, when the inputs are orthogonal, they have no effect on each other's parameter estimates in the model.

- Orthogonal inputs almost never occur with observational data, but perhaps if they were orthogonalized somehow, we would once again obtain simple expressions such as the ratio of inner products given above.

- In a model with an intercept and a single input $\boldsymbol{x}$, the least squares coefficient of $\boldsymbol{x}$ has the form

$$\hat{\beta}_1 = \frac{\langle \boldsymbol{x} - \bar{x}1, \boldsymbol{y} \rangle}{\|\boldsymbol{x} - \bar{x}1\|^2}, \tag{2.7}$$

  where

$$\bar{x} = n^{-1} \sum_i x_i$$

  denotes the sample mean.

Note that (2.7) can be viewed as the application of two steps:

1. regress $\boldsymbol{x}$ on 1 to produce the residual $\boldsymbol{z} = \boldsymbol{x} - \bar{x}1$;
2. regress $\boldsymbol{y}$ on the residual $\boldsymbol{z}$ to give the coefficient $\hat{\beta}_1$.

- This recipe generalizes to the case of p inputs, as shown below.

Note that hence the simple regression coefficients computed there are in fact also the multiple regression coefficients.

1. Initialize $\boldsymbol{z}_0 = \boldsymbol{x}_0 = 1$.
2. For $j = 1, 2, \ldots, p$. Regress $\boldsymbol{x}_j$ on $\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{j-1}$ to produce coefficients

$$\hat{\gamma}_{\ell j} = \langle \boldsymbol{z}_\ell, \boldsymbol{x}_j \rangle / \langle \boldsymbol{z}_\ell, \boldsymbol{z}_\ell \rangle \tag{2.8}$$

   and residual vector

$$\boldsymbol{z}_j = \boldsymbol{x}_j - \sum_{k=0}^{j-1} \hat{\gamma}_{kj} \boldsymbol{z}_k$$

3. Run a univariate regression of y on z_p plus intercept; the coefficient of z_p in this regression is the multivariate coefficient $\hat{\beta}_p$.

- The inputs

$$z_0, \ldots, z_{i-1}$$

  in step 2 are orthogonal.
- Hence the multiple regression in step 2 is on uncorrelated inputs, and the coefficients can therefore be computed using the formula for univariate regression.
- This justifies our assertion that the multivariate coefficients $\gamma_{\ell j}$ are actually given by (2.8).

- Since the $\boldsymbol{z}_i$ are all orthogonal, they form a basis for the column space of X.
- Indeed, if we want the coefficients of $\boldsymbol{x}_j$ in this basis, they are given by the equation in step 2:

$$\boldsymbol{z}_j + \sum_{k=0}^{j-1} \hat{\gamma}_{kj} \boldsymbol{z}_k = \boldsymbol{x}_j$$

- Therefore the projection $\hat{y}$ of y onto the column space of X is the same as the projection of y onto the space spanned by the $\boldsymbol{z}_j$.
- Since $\boldsymbol{z}_p$ alone involves $\boldsymbol{x}_p$ (with coefficient 1), we see that $\hat{\beta}_p$ as defined in step 3 above is indeed the multiple regression coefficient of y on $\boldsymbol{x}_p$.

- The above argument only gave the last component $\hat{\beta}_p$.
- However, by rearranging the $\boldsymbol{x}_j$, any one of them could be in the last position.
- Hence stated more generally, we have shown that the j-th multiple regression coefficient is the univariate regression coefficient of y on the residual after regressing $\boldsymbol{x}_j$ on

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1}, \boldsymbol{x}_{j+1}, \ldots, \boldsymbol{x}_p.$$

- Here is a very simple R script so you can experiment with the above procedure.

```
n <- 500; k <- 3; ## n observations, k independent variables

X <- matrix(rnorm(n*k), n, k)
X[,3] <- X[,1] + X[,2] + rnorm(n) ## introduce some correlation

truebeta <- matrix(c(1,2,3), k,1)
y <- X
ones <- matrix(rep(1,n), n, 1) ## column of 1s
z0 <- ones

fit1 <- lm(X[,1] ~ 0 + z0)
gamma1 <- coef(fit1); z1 <- residuals(fit1)

fit2 <- lm(X[,2] ~ 0 + z0 + z1)
gamma2 <- coef(fit2); z2 <- residuals(fit2)

fit3 <- lm(X[,3] ~ 0 + z0 + z1 + z2)
gamma3 <- coef(fit3); z3 <- residuals(fit3)

print(coef(lm(y ~ z3)))
print(coef(lm(y ~ X)))
```

The QR Decomposition

- We can represent step 2 in matrix form:

$$X = Z\Gamma$$

where Z has as columns the $\boldsymbol{z}_j$ (in order), and Γ is the upper triangular matrix with entries $\hat{\gamma}_{kj}$.

- Introducing the diagonal matrix $D_{jj} = \|\boldsymbol{z}_j\|$, then $X = Z\Gamma$ becomes:

$$X = QR \tag{2.9}$$

where

$$Q = ZD^{-1}$$

is an $n \times (p+1)$ matrix, and

$$R = D\Gamma$$

is a $(p+1) \times (p+1)$ upper triangular matrix.

- The matrix Q is not a true orthogonal matrix since it may not be square, but it is a "matrix of orthonormal columns" since

$$Q'Q = I.$$

- Eq. (2.9) is the QR decomposition of X, which represents a convenient orthogonal basis for the column space of X.

- Lots of quantities of interest in a regression problem can be computed once we have the QR decomposition of X.
- For example

$$X'X = (QR)'QR = R'Q'QR = R'R,$$

so as long as R is invertible one has

$$\hat{\beta} = (X'X)^{-1}X'y = (R'R)^{-1}R'Q'y = R^{-1}Q'y. \qquad (2.10)$$

and the fitted response is

$$\hat{y} = X\hat{\beta} = QRR^{-1}Q'y = QQ'y$$

- Moreover R^{-1} can be computed more efficiently than the inverse of a general matrix since R is upper triangular, so this method has many advantages.

- In practice, least-squares coefficients are usually computed using the QR decomposition of X as in (2.10), as long as the full X matrix is available.
- In problems with a very large number of observations, one may calculate an online regression by only saving the $p(p+1)/2$ independent elements of $X'X$ and the p elements of $X'y$.
- In this case, one may calculate regression coefficients using the Cholesky decomposition of $X'X$.
- With n observations and p features, the Cholesky decomposition requires $p^3 + np^2/2$ operations, while the QR decomposition requires np^2 operations.
- Depending on the relative size of n and p, the Cholesky method can sometimes be faster; on the other hand, it can be less numerically stable, so the QR method is preferred whenever X is available.

Instability

- If $\boldsymbol{x}_p$ is highly correlated with some of the other $\boldsymbol{x}_k$s, the residual vector $\boldsymbol{z}_p$ will be close to zero, so its norm $\|\boldsymbol{z}_p\|$ will be small, and hence the coefficient estimate

$$\hat{\beta}_p = \frac{\langle \boldsymbol{z}_p, \boldsymbol{y} \rangle}{\langle \boldsymbol{z}_p, \boldsymbol{z}_p \rangle} = \frac{\langle \boldsymbol{z}_p, \boldsymbol{y} \rangle}{\|\boldsymbol{z}_p\|^2}$$

  will be very sensitive to $\|\boldsymbol{z}_p\|^2$.
- The presence of instability can of course be seen directly from the multivariate formulation.
- Note that (2.3) suffers from instability when $X'X$ has a small eigenvalue.
- We illustrate this instability in the following numerical example.

Example 2.2

Consider estimating the coefficients from data which was actually generated from the model $y = 3 + x_1 + x_2$, and observe that the coefficient estimates for x_1 and x_2 are huge in magnitude relative to their true values, and of opposite sign to each other.

```
N <- 20;
x1 <- rnorm(N);
x2 <- rnorm(N, mean = x1, sd = .01);
y <- rnorm(N, mean = 3 + x1 + x2)
lm(y ~ x1 + x2)$coef
(Intercept)          x1          x2
2.994764  -35.833870   37.642712
```

- One might be tempted to think of *ad hoc* procedures by means of which the matrix X could be transformed to make $X'X$ stably invertible, by eg. removing some of the columns, or collapsing several into one.
- Any such method amounts to variable selection, and better methods for variable selection will be discussed later on.

Bayesian Regression

- Continue to consider the linear model above.
- To avoid certain kinds of confusion later, in this section we will refer to the coefficients as "weights" and denote them $\boldsymbol{w}$ instead of $\boldsymbol{\beta}$.
- We will assume that the residuals ϵ_i are i.i.d. with known variance σ_n^2.

- We put a zero mean Gaussian prior with covariance Σ_p on the weights:

$$\boldsymbol{w} \sim N(0, \Sigma_p)$$

- Inference is based on the posterior distribution over the weights, which is given by

$$\log p(\boldsymbol{w} \mid \boldsymbol{y}, X) = \log p(\boldsymbol{y} \mid X, \boldsymbol{w}) + \log p(\boldsymbol{w}) - \log p(\boldsymbol{y} \mid X)$$

- Writing only the terms from the likelihood and prior which depend on the weights, and completing the square we obtain

$$-\log p(\boldsymbol{w} \mid \boldsymbol{y}, X) = \frac{1}{2\sigma_n^2}(\boldsymbol{y} - X\boldsymbol{w})^\top(\boldsymbol{y} - X\boldsymbol{w}) + \frac{1}{2}\boldsymbol{w}^\top\Sigma_p^{-1}\boldsymbol{w} \tag{2.11}$$

$$= \frac{1}{2}(\boldsymbol{w} - \overline{\boldsymbol{w}})^\top A(\boldsymbol{w} - \overline{\boldsymbol{w}}) \tag{2.12}$$

$$\text{where } A := \frac{1}{\sigma_n^2}X^\top X + \Sigma_p^{-1} \tag{2.13}$$

and where

$$\overline{\boldsymbol{w}} = \sigma_n^{-2}A^{-1}X\boldsymbol{y}$$

- We recognize the form of the posterior distribution as Gaussian with mean $\overline{\boldsymbol{w}}$ and covariance matrix A^{-1} To make predictions for a test case we average over all possible parameter values, weighted by their posterior probability.
- Suppose we observe a new sample with predictors $\boldsymbol{x}_*$ and we wish to determine the posterior predictive density over responses f_*, it is

$$p(f_* \mid \boldsymbol{x}_*, X, \boldsymbol{y}) = \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w} \mid X, \boldsymbol{y}) d\boldsymbol{w} \qquad (2.14)$$

- Some algebra shows that the latter distribution is normal, as follows

$$f_* \sim N(\sigma_n^{-2} \boldsymbol{x}_*^\top A^{-1} X \boldsymbol{y}, \sigma_n^{-2} \boldsymbol{x}_*^\top A^{-1} \boldsymbol{x}_*)$$

Variable selection

- The *variable selection problem* arises when the design matrix X contains more variables (i.e. columns) than we necessarily want to use in a more refined version of the model.
- This is distinct from the problem of inference conditional on X, but intertwined because we would often like to construct the proper X to use in whatever inference is to follow.
- In empirical work in econometrics, we almost never know, at the outset, which variables are truly important enough to include in the model.

- Variable selection is typically approached by constructing an alternative estimator of β having the property that some coefficient estimates (presumably those corresponding to removal candidates) are set to precisely zero.
- Those variables for which

$$\hat{\beta}_j \neq 0$$

are said to be "selected."

Penalized Regression

- A wide class of statistical estimation problems can be written in the form

$$\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p} f(\beta_0, \beta) \tag{2.15}$$

where:

$$f(\beta_0, \beta) \quad := \quad \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - g(x_i; \beta))^2 + \lambda P_\alpha(\beta)$$

or, in no-intercept models, a similar form which constrains $\beta_0 = 0$.

- Many of the alternative estimators used in variable selection take the form (2.15), so we will study this problem in some detail.
- In linear models,

$$g(x_i; \beta) = x_i^T \beta.$$

- The term proportional to $P_\alpha(\beta)$ is called a *penalty term*; it penalizes vectors $\beta \in \mathbb{R}^p$ with large values of the penalty function $P_\alpha(\beta)$.
- The penalty function is allowed to have its own internal parameters which we denote by α.
- The parameter(s) α are hyper-parameters which are usually viewed as exogenous, while the "complexity parameter" λ is usually chosen via cross-validation.
- The parameters α need not be present at all; for example, in LASSO one has

$$P_\alpha(\beta) = \sum_j |\beta_j|$$

which does not depend on any hyperparameters.

- As long as the problem is convex, and the non-differentiable part of $P_\alpha(\beta)$ is separable, then coordinate descent may be used.
- One iterates over

$$\beta_0, \beta_1, \ldots, \beta_p,$$

  optimizing over each coordinate in turn.
- In many models of interest, the single-coordinate optimizations can be done very quickly by exploiting special structure; we describe below, in complete detail, how to compute these updates.

- Another interesting speedup is as follows.
- First note that, if one is using such a model in the first place, one presumably needs to choose λ by cross-validation.
- Hence one is always solving the optimization problem for a sequence of values of λ, also called a "lambda path."
- For nearby values of λ, the solutions are very close, so there is a great computational advantage to using solutions for nearby values of λ as a "warm start."

- We will treat the problem with intercept, and without separately.
- In ordinary least squares, without loss of generality one can assume there is no intercept term, but note that the design matrix X may contain a column of 1s which is mathematically the same as including an intercept term in the model's equation.
- In penalized regression, one must consider intercept terms with some care, noting that β_0 is included in the least-squares term, but NOT included in the penalty term, ie. $P_\alpha(\beta)$ does not depend on β_0.

Lemma 2.3

Soft-thresholding lemma. *Let $\gamma > 0$ and $b \in \mathbb{R}$ be arbitrary, fixed constants. Let $x^* \in \mathbb{R}$ be the solution to the convex optimization problem*

$$\min_x \left[\frac{1}{2}(x-b)^2 + \gamma |x| \right]$$

Then

$$x^* = S(b,\gamma) \equiv sign(b) \max(0, |b| - \gamma)$$

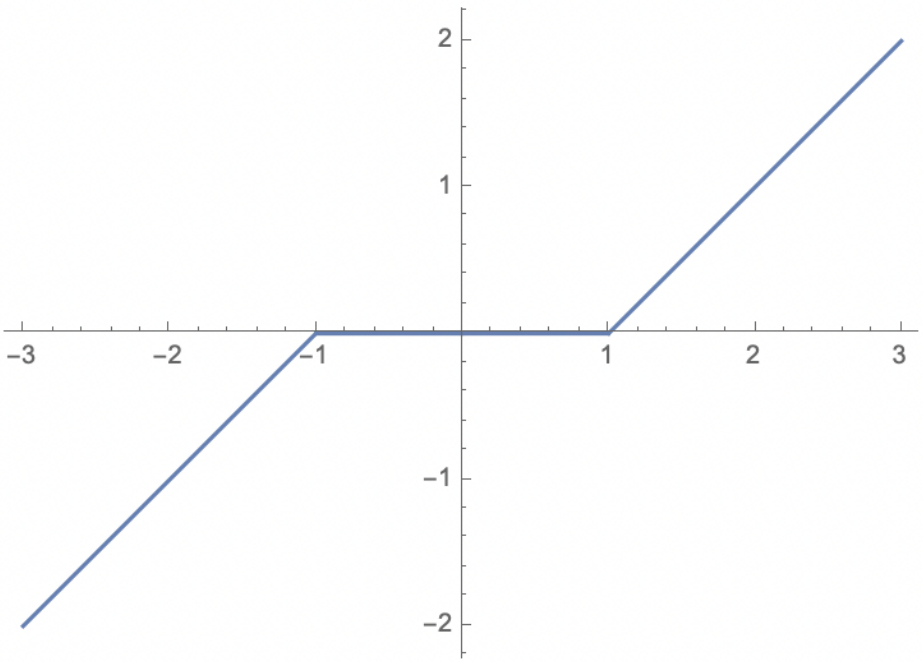
where the expression on the right serves to define the function $S(b,\gamma)$, called the soft-threshold function.

Proof. exercise for the reader.

```
In[1]:= s[b_, g_] := (b / Abs[b]) Max[0, Abs[b] - g];

In[4]:= Plot[s[b, 1], {b, -3, 3}, AspectRatio → Automatic]
```

Out[4]=



- We first treat penalized regression with an intercept, i.e. model equations of the form

$$y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \epsilon$$

- It is conventional to define the predictor variables x_j so that they are assumed to come from mean-zero distributions.
- This is necessary in order that β_0 can be interpreted as the whole intercept; otherwise β_0 would have the more-complicated and less-appealing interpretation as the part of the intercept not already explained by the intercepts in the various x_j's.

- We seek

$$\min_{\beta_0,\beta} f(\beta_0, \beta)$$

where:

$$f(\beta_0, \beta) := \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Note that, in the second term, the sum starts from $j = 1$, meaning the intercept term β_0 is not penalized.
- The motivation for the $1/(2n)$ coefficient in the first term is as follows.
- The $1/2$ is merely for convenience when calculating gradients, and matches the $1/2$ in the soft-thresholding problem.
- The $1/n$ is presumably so that similar numerical ranges for λ make sense across vastly different sample sizes.

- Suppose we have some current guesses $(\tilde{\beta}_0, \tilde{\beta})$ and we consider minimizing over β_j while keeping the other $\tilde{\beta}_k$ fixed for $k \neq j$.
- We shall discard terms which are constants with respect to β_j; then we seek to minimize

$$\phi(\beta_j) = \frac{1}{2n} \sum_{i=1}^{n} (z_{ij} - x_{ij}\beta_j)^2 + \lambda |\beta_j|$$

where we define

$$z_{ij} := y_i - \tilde{\beta}_0 - \sum_{k \neq j} x_{ik} \tilde{\beta}_k \,,$$

- We then multiply out and perform the sum:

$$\sum_{i=1}^{n}(z_{ij} - x_{ij}\beta_j)^2 = \sum_{i=1}^{n}(x_{ij}^2\beta_j^2 - 2z_{ij}x_{ij}\beta_j + z_{ij}^2)$$
$$= \|\boldsymbol{x}^j\|^2\beta_j^2 - 2\langle \boldsymbol{z}^j, \boldsymbol{x}^j\rangle\beta_j$$

where in the right-most expression, we drop the constant term, and introduce the notation $\boldsymbol{x}^j$ for the j-th column of the X matrix, and $\boldsymbol{z}^j$ for the j-th column of the matrix with entries z_{ij}.

- Now, after some algebra involving completing the squares and dividing by a positive constant, one can show that $\phi(\beta_j)$ has the same minimum as

$$\frac{1}{2}\left(\beta_j - \hat{\beta}_j\right)^2 + \gamma\,|\beta_j| \tag{2.16}$$

$$\gamma := \frac{n\lambda}{\|\boldsymbol{x}^j\|^2} \quad \text{and} \quad \hat{\beta}_j = \frac{\langle \boldsymbol{z}^j, \boldsymbol{x}^j \rangle}{\|\boldsymbol{x}^j\|^2} \tag{2.17}$$

- By the soft-thresholding lemma, the minimum of $\phi(\beta_j)$ and also of (2.16) is given by

$$\beta_j^* = S(\hat{\beta}_j, \gamma). \tag{2.18}$$

- We also need to minimize with respect to β_0, keeping $\tilde{\beta}$ fixed.
- This is easy, since the non-differentiable penalty terms don't involve β_0.
- Taking the derivative and setting equal to zero gives

$$0 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - x_i^T\tilde{\beta})$$

- The solution is:

$$\beta_0^* = \overline{y} - \sum_{j=1}^{p}\tilde{\beta}_j\overline{\boldsymbol{x}}^j \tag{2.19}$$

where an over-line denotes the sample mean.

- We now know, for all

$$j = 0, 1, \ldots, p,$$

how to optimize over β_j keeping β_k fixed for $k \neq j$.

- The answers to those univariate optimizations are given by (2.19) for $j = 0$ and (2.18) for $j > 0$.
- This also suggests an algorithm for finding the optimal vector $\hat{\beta}$.

Definition 2.4

Coordinate descent algorithm. *Initialize* $\tilde{\beta}_j$ *for* $j = 0, \ldots, p$ *arbitrarily, possibly as a warm start from a previous problem. Repeat the following steps until, in one entire sweep over* $j = 0, \ldots, p$*, there was no meaningful change in any component of* $\tilde{\beta}$*.*

1. *Compute* β_0^* *using* (2.19) *and replace* $\tilde{\beta}_0$ *with* β_0^**.*
2. *For* $j = 1, 2, \ldots, p$ *do the following:*
    1. *Compute* β_j^* *using* (2.18)*.*
    2. *Replace* $\tilde{\beta}_j$ *with* β_j^* *from (a).*

- Many elements can be precomputed and stored, outside of the coordinate-descent loop.
- In particular,

$$\langle \boldsymbol{z}^j, \boldsymbol{x}^j \rangle = \langle \boldsymbol{y}, \boldsymbol{x}^j \rangle - \tilde{\beta}_0 \langle 1, \boldsymbol{x}^j \rangle - \sum_{k \neq j} \tilde{\beta}_k \langle \boldsymbol{x}^k, \boldsymbol{x}^j \rangle \qquad (2.20)$$

- In an implementation in computer code, the various inner products in this formula, as well as the square-norms $\|\boldsymbol{x}^j\|^2$, should all be pre-computed before the coordinate descent begins.

Forcing zero intercepts

- We now treat the case where we force the intercept to be zero, i.e. we constrain $\beta_0 = 0$.
- This changes the problem to the following:

$$\min_{\beta} f(\beta) \qquad \text{where:}$$

$$f(\beta) \quad := \quad \frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

- The derivation above for the intercept case goes through almost unchanged; we need to redefine

$$z_{ij} := y_i - \sum_{k\neq j} x_{ik}\tilde{\beta}_k, \qquad \text{(no-intercept formulation)}$$

  and skip the updating of β_0.
- Otherwise, it all works just as above.
- Computationally, we can adapt the same code from the intercept case; just initialize $\beta_0 = 0$ and never update it.

Lightning review of convex optimization

- The *standard form optimization problem* is defined to be:

$$\begin{aligned} &\text{minimize } f_0(x) \\ &\text{subject to } f_i(x) \leq 0, \;\; i = 1, \ldots, m \\ &\qquad\qquad h_i(x) = 0, \;\; i = 1, \ldots, p \end{aligned}$$

- The optimal value will be denoted

$$p^* = \inf \{ f_0(x) \mid f_i(x) \leq 0, \;\; h_j(x) = 0 \;\; (\forall i, j) \}$$

- By convention,

$$p^* = \infty$$

if the problem is infeasible (no x satisfies the constraints), and

$$p^* = -\infty$$

if the problem is unbounded below.

- A constraint $f_i, i > 0$ is said to be *active at* x_0 if

$$f_i(x_0) = 0.$$

- The standard form optimization problem has an implicit constraint

$$x \in \mathcal{D} = \bigcap_{i=0}^{m} \mathrm{dom}(f_i) \cap \bigcap_{j=1}^{p} \mathrm{dom}(h_j)$$

- This $\mathcal{D}$ is called the domain of the problem.

- For a general optimization problem in standard form, define the *Lagrangian*

$$L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R},$$

with

$$\mathrm{dom} L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$$

as follows

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x).$$

- The elements of λ, ν are called *Lagrange multipliers*.

Definition 2.5

The standard form convex optimization problem is

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{aligned}$$

where all f_i are convex.

Definition 2.1

A *primal-dual pair* is defined as

$$\zeta = \{x^*, (\lambda^*, \nu^*)\}, \quad x^* \in \mathcal{D}.$$

The primal-dual pair ζ is said to satisfy the *KKT conditions* if the following four properties hold:

1. primal constraints: $f_i(x^*) \leq 0,\ i = 1, \ldots, m,\ h_i(x^*) = 0,\ i = 1, \ldots, p$
2. dual constraints: $\lambda^* \succeq 0$
3. complementary slackness: $\lambda_i^* f_i(x^*) = 0,\ i = 1, \ldots, m$
4. first-order condition:

$$x^* \quad \text{minimizes} \quad L(x, \lambda, \nu) \quad \text{over} \quad x \in \mathbb{R}^n$$

- In a convex optimization problem, if a primal-dual pair satisfies the KKT conditions, then the "x" variable of the pair is optimal (and the λ, ν are optimal for the dual problem, but we haven't discussed that).
- Much more can be said about these problems; please read all of Boyd and Vandenberghe (2009) at some point in your life.

Lasso as a constrained problem

- It turns out that Lasso can be equivalently described as a constrained problem.
- We consider the no-intercept case for simplicity, but a similar story holds for the case with an intercept.

- Consider the following constrained optimization problem:

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \right\} \quad (2.21)$$

$$\text{subject to: } \sum_{j=1}^{p} |\beta_j| \leq t \quad (2.22)$$

- Let $\hat{\beta}$ denote the least squares estimates, i.e. the solution to (2.21) without the constraint (2.22).

- If t is chosen larger than

$$t_0 := \sum_1^p |\hat{\beta}_j|,$$

where $\hat{\beta}_j$ denote the least squares estimates, then the lasso estimates are simply the least-squares $\hat{\beta}_j$.

- Otherwise, if

$$0 < t < t_0$$

then the constraint is active.

- The Lagrangian is

$$L(\beta, \lambda) = f_0(\beta) + \lambda f_1(\beta) \tag{2.23}$$

$$f_0(\beta) = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \tag{2.24}$$

$$f_1(\beta) = \sum_{j=1}^{p} |\beta_j| - t \tag{2.25}$$

- The fourth KKT condition states that β minimizes $L(\beta, \lambda)$ which is the usual form of the Lasso problem.

- A natural standardized parameter that is often used as the x-axis for plots is:

$$s = \frac{t}{\sum_{j=1}^{p} \left| \hat{\beta}_j \right|}$$

  where again the denominator refers to least squares estimates.
- This standardized parameter then runs from 0 to 1.

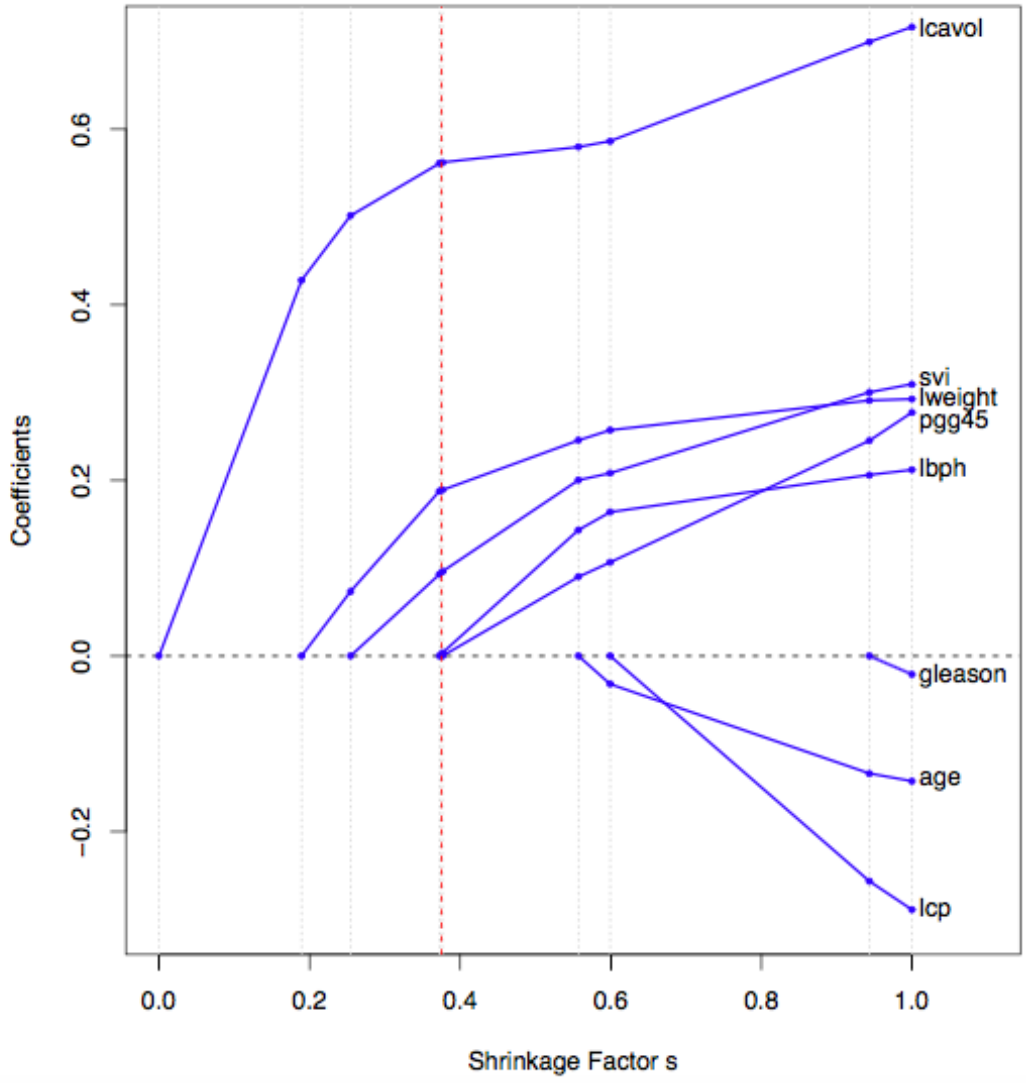
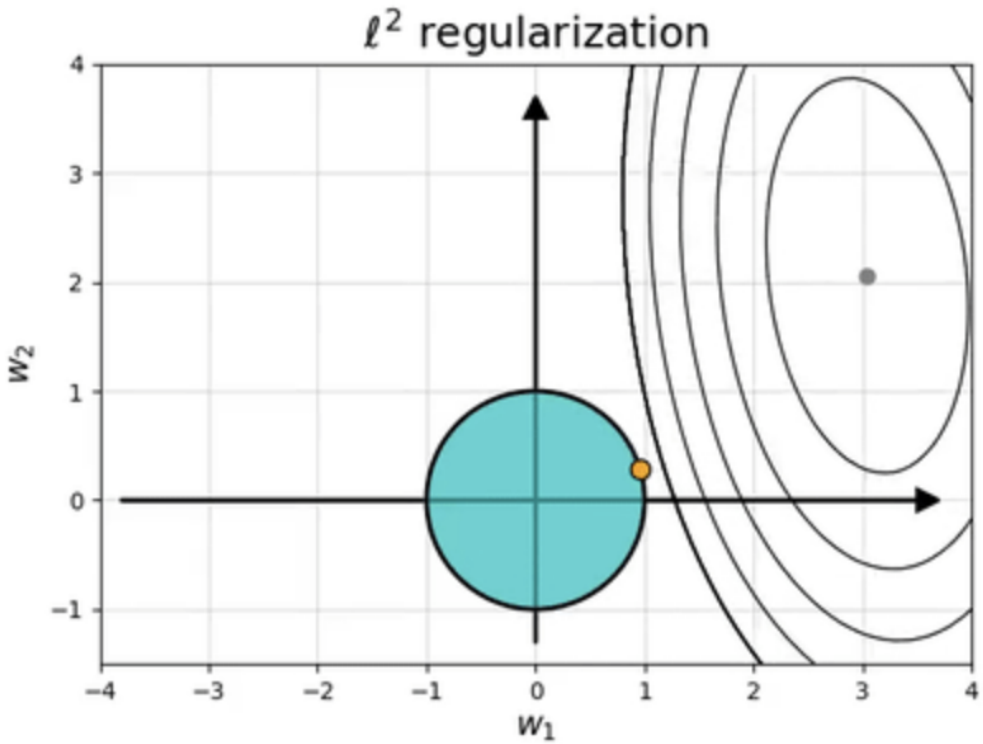


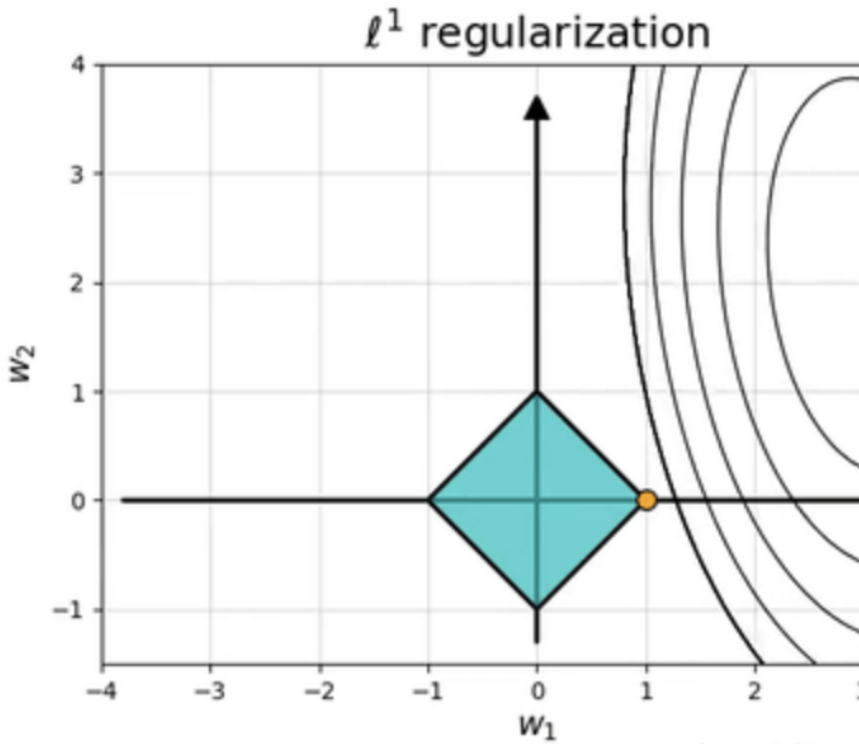
Figure: Profiles of lasso coefficients, as the tuning parameter is varied. From Friedman, Hastie, and Tibshirani (2001)

The Elastic Net

- Lasso can perform variable selection by returning sparse solution vectors.
- Here is an interesting graphic that may help us understand why this is the case.

The contours are level curves of the least squares objective.





However, it was observed by the inventors of lasso (Tibshirani, 1996) that

(a) If there is a group of variables among which the pairwise correlations are very high, then the lasso tends to select only one variable from the group and does not care which one is selected.

(b) For $n > p$, if there are high correlations between predictors, it has been empirically observed that the prediction performance of the lasso is dominated by ridge.

- Motivated by point (b) and presumably trying to get the best of both lasso and ridge, Zou and Hastie (2005) introduced the elastic-net, which is a nice compromise between the two.
- The elastic-net selects variables like the lasso, and shrinks together the coefficients of correlated predictors like ridge.
- It also has considerable computational advantages.

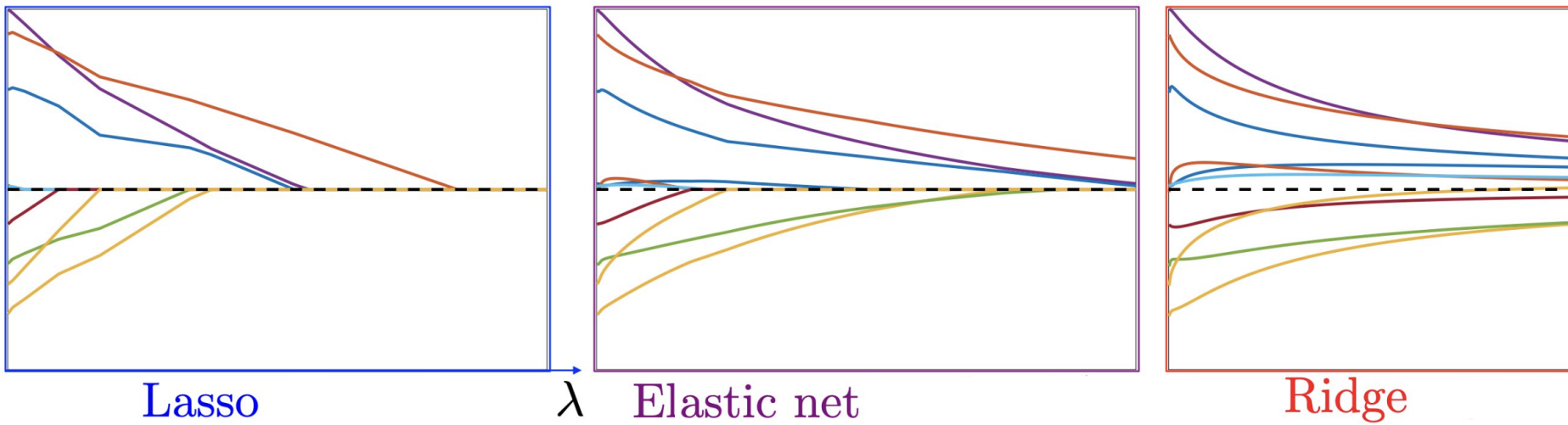
- The above coordinate descent technique can be generalized to also provide solution paths for the elastic net.
- For coefficients and factors of 1/2, we adopt the conventions of Friedman, Hastie, and Tibshirani (2010).
- They define elastic net as solving

$$\min_{\beta_0,\beta} f(\beta_0, \beta)$$

where:

$$f(\beta_0, \beta) := \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda\alpha\sum_{j=1}^{p}|\beta_j| + \frac{1}{2}\lambda(1-\alpha)\sum_{j=1}^{p}\beta_j^2$$

- We will omit the details, but the optimization procedure can be derived as before, by showing that the single-variable optimizations are equivalent to the soft-thresholding lemma.


Lasso
λ
Elastic net
Ridge

Generalization performance and Cross-validation

- We follow Friedman, Hastie, and Tibshirani (2001) here.
- The generalization performance of a learning method relates to its prediction capability on independent test data.
- Assessment of this performance is extremely important in practice, since it guides the choice of learning method or model, and gives us a measure of the quality of the ultimately chosen model.

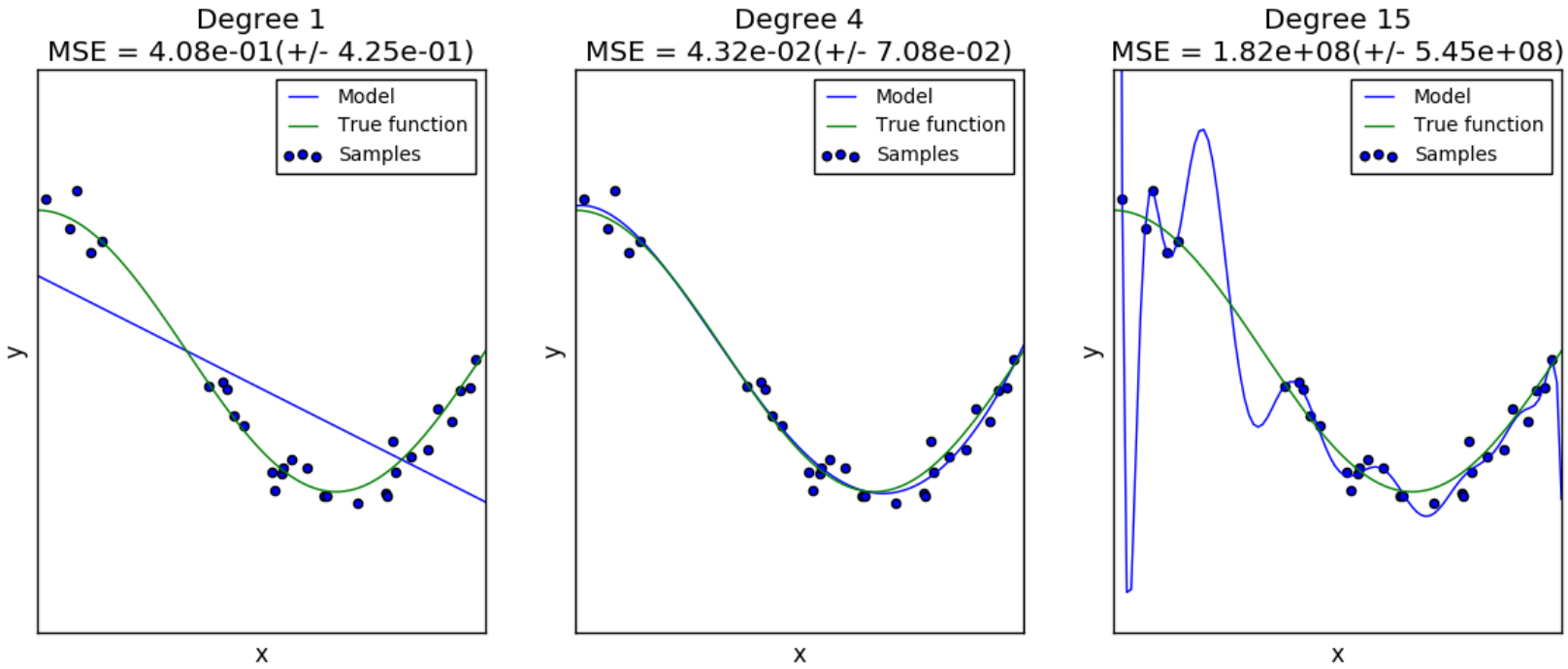


Figure: Classic overfitting of a polynomial function. Neither of the extremes possess prediction capability on independent test data.

- We have a target variable Y, a vector of inputs X, and a prediction model $\hat{f}(X)$ that has been estimated from a training set $\mathcal{T}$.
- The loss function for measuring errors between Y and $\hat{f}(X)$ is denoted by

$$L(Y, \hat{f}(X)).$$

- Typical choices for L are squared error and absolute error.

- Test error, also referred to as generalization error, is the prediction error over an independent test sample

$$\mathrm{Err}_{\mathcal{T}} = E[L(Y, \hat{f}(X)) \,|\, \mathcal{T}]$$

  where both X and Y are drawn randomly from their joint distribution (population).
- Here the training set $\mathcal{T}$ is fixed, and test error refers to the error for this specific training set.

- A related quantity is the expected prediction error (or expected test error)

$$\mathrm{Err} = E[L(Y, \hat{f}(X))] = E[\mathrm{Err}_{\mathcal{T}}].$$

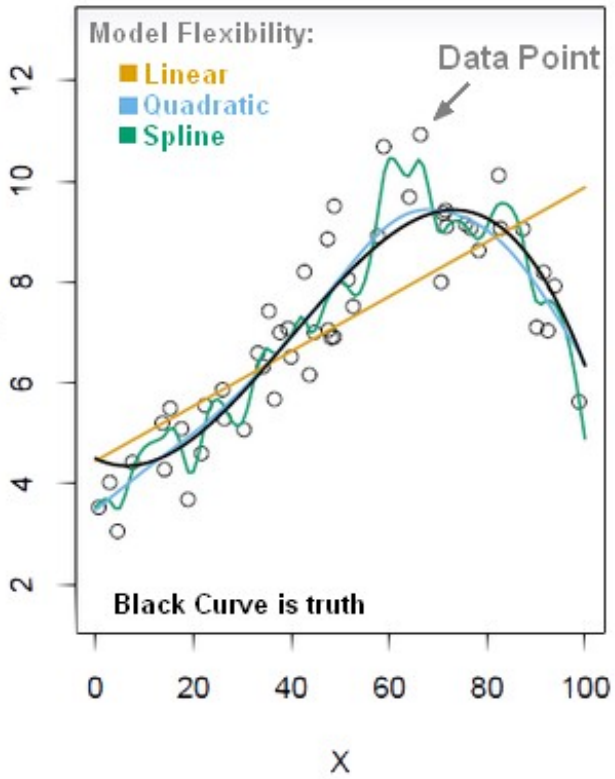
- This expectation averages over everything that is random, including the randomness in the training set that produced $\hat{f}$.

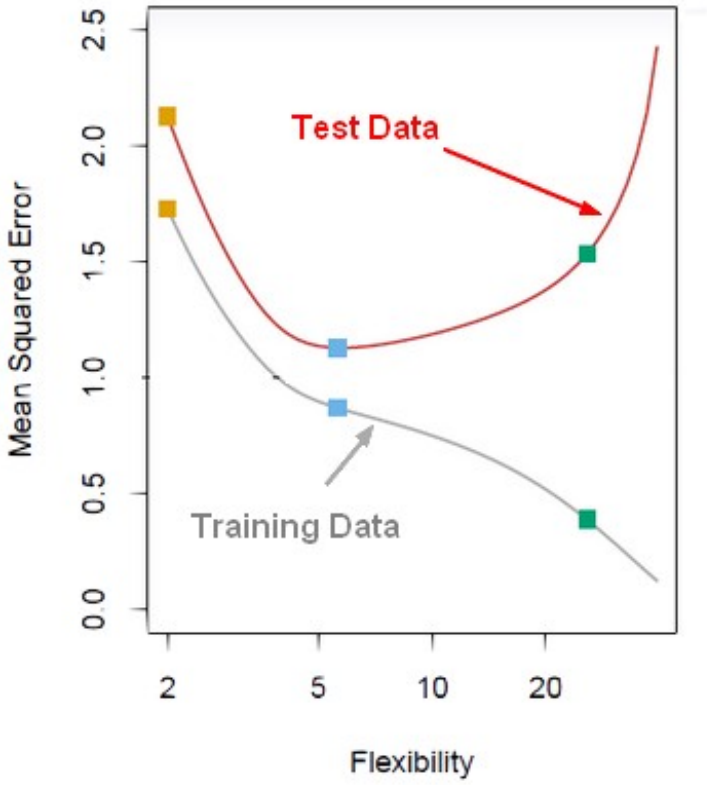
- Training error is the average loss over the training sample:

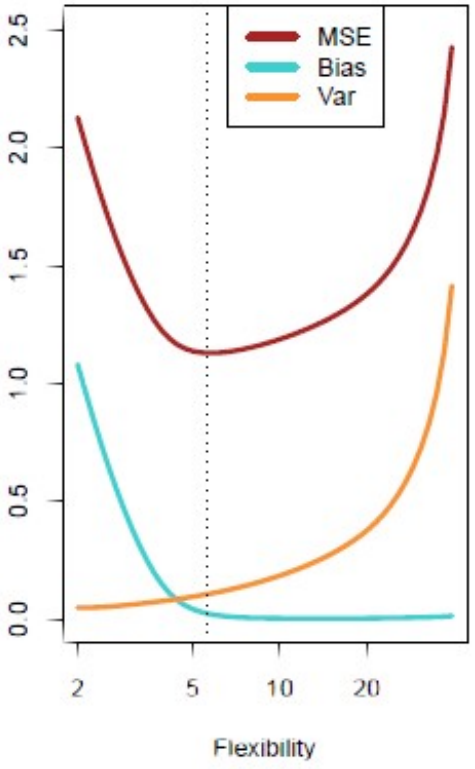
$$\overline{\mathrm{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

- We would like to know the expected test error of our estimated model $\hat{f}$.
- As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures.
- Hence there is a decrease in bias but an increase in variance.
- There is some intermediate model complexity that gives minimum expected test error.

- The bias-variance tradeoff can be visualized intuitively for a polynomial fitting problem as follows.


Model Flexibility:
Linear
Quadratic
Spline
Data Point
Black Curve is truth
X


Test Data
Training Data
Mean Squared Error
Flexibility


MSE
Bias
Var
Flexibility

- In a similar vein, Hastie, Tibshirani and Friedman presented an example in which we can see the bias-variance tradeoff emerge explicitly in the context of the lasso.

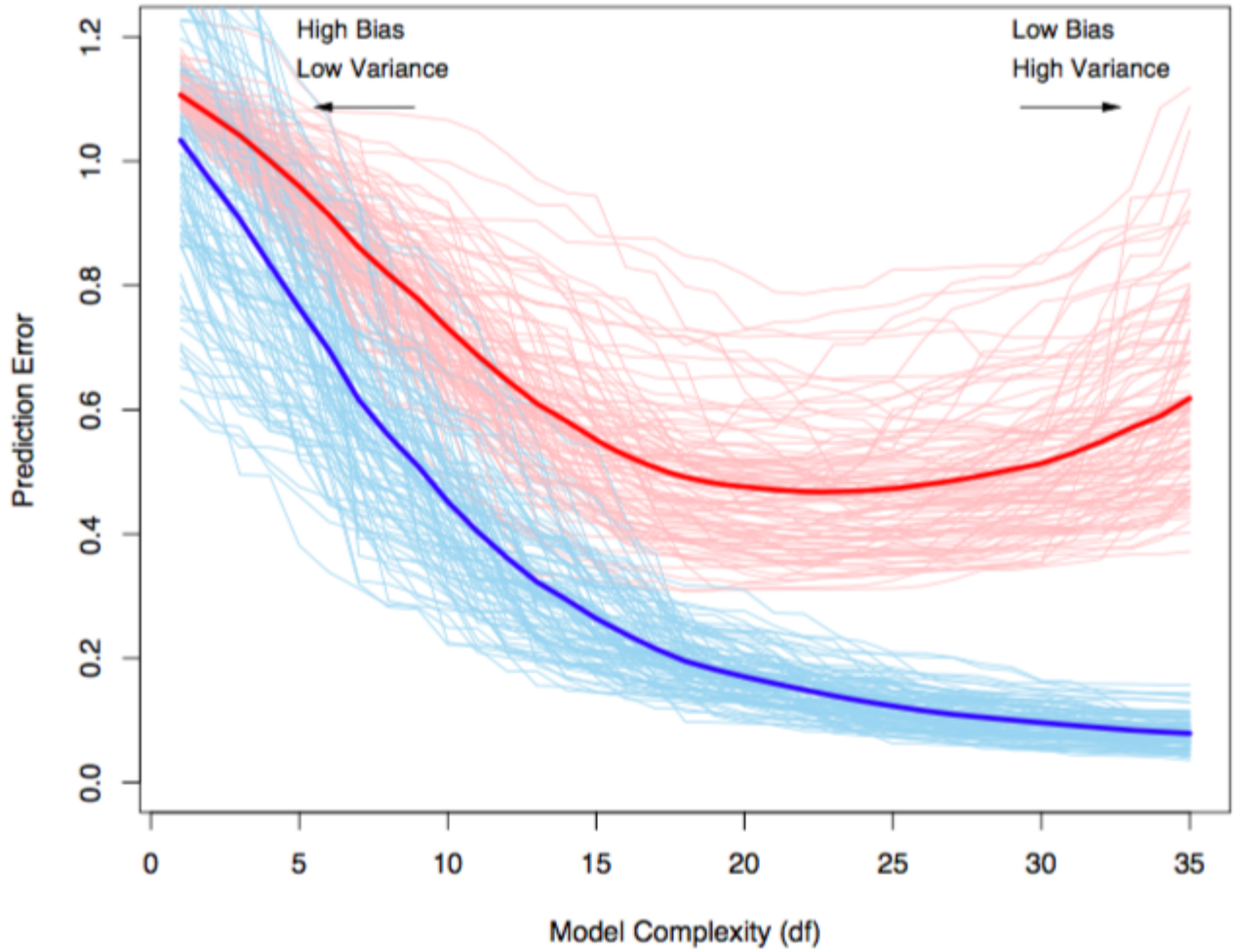


Figure: Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error $\overline{\mathrm{err}}$, while the light red curves show the conditional test error $\mathrm{Err}_{\mathcal{T}}$ for 100 training sets of size 50 each, as the model complexity is increased.

- The lasso was used to produce the sequence of fits in Figure 2.3.
- The solid red curve is the average, and hence an estimate of Err.
- The solid curves show the expected test error Err and the expected training error $E[\overline{\mathrm{err}}]$.

- Unfortunately training error is not a good estimate of the test error, as seen in Figure 2.3.
- Training error consistently decreases with model complexity, typically dropping to zero if we increase the model complexity enough.
- However, a model with zero training error is overfit to the training data and will typically generalize poorly.
- Estimation of $\mathrm{Err}_{\mathcal{T}}$ is the true goal, but Err can actually be estimated using cross-validation.

- Typically our model will have a tuning parameter or parameters α and so we can write our predictions as $\hat{f}_\alpha(x)$.
- For lasso, there was a single tuning parameter λ, so in that case, α was a one-dimensional vector containing λ.
- The tuning parameter varies the complexity of our model, and we wish to find the value of α that minimizes error, that is, produces the minimum of the average test error curve in Figure 2.3 .

It is important to note that there are in fact two separate goals that we might have in mind:

- Model selection: estimating the performance of different models in order to choose the best one.
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

- If we are in a data-rich situation, the best approach for both problems is to divide the dataset into three parts: a training set, a validation set, and a test set.
- The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model.
- Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis.

- Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error.
- Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

- The methods we describe here are designed for situations where there is insufficient data to split it into three parts.
- It is difficult to give a general rule on how much training data is enough; among other things, this depends on the signal-to-noise ratio of the underlying function, and the complexity of the models being fit to the data.

- When there is insufficient data one can approximate the validation step by efficient sample re-use (eg. cross-validation).

- Probably the simplest and most widely used method for estimating prediction error is cross-validation.
- This method directly estimates the expected extra-sample error

$$\mathrm{Err} = E[L(Y, \hat{f}(X))]$$

  the average generalization error when the method $\hat{f}(X)$ is applied to an independent test sample from the joint distribution of X and Y.
- As mentioned earlier, we might hope that cross-validation estimates the conditional error, with the training set $\mathcal{T}$ held fixed.
- But cross-validation typically only estimates the expected prediction error.

- Ideally, if we had enough data, we would set aside a validation set and use it to assess the performance of our prediction model.
- Since data are often scarce, this is usually not possible.
- To finesse the problem, K-fold cross-validation uses part of the available data to fit the model, and a different part to test it.

- For the purposes of illustration, suppose that we have 5 candidate models, ranging from more complex to less complex, that we would like to decide between.
- Each of the five models still contains some unknown parameters; i.e. each model will still need to be fit or "trained" on some training set before it can make any numerical prediction.
- The real question is which of the 5 models is expected to do the best job of making predictions, once it has been trained, in the context of whatever true underlying data-generating process is generating the dynamics in the environment we are trying to model.

- To make this decision, we would like to essentially rank the models, from 1 to 5, based on some performance metric.
- Since, presumably, the chosen model will be used to make predictions going forward (on data that was not used to train), our performance metric should emphasize generalization performance.

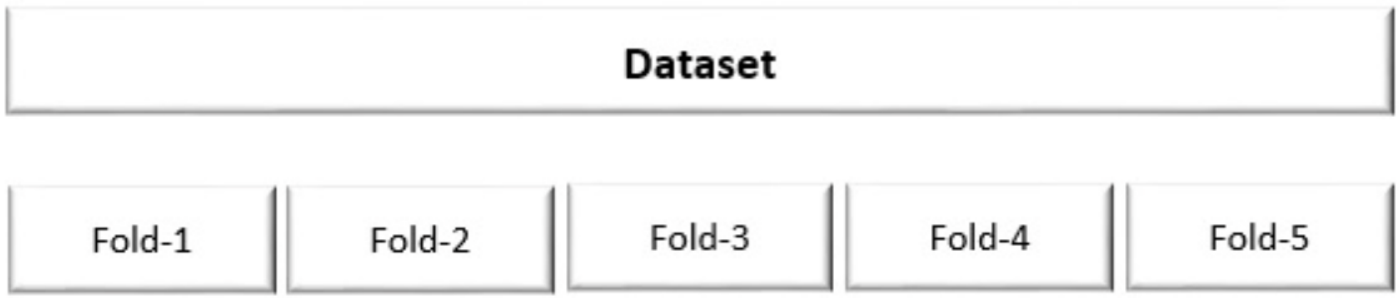
- We take all the data we have available, and split the data into:

  (a) a part that is used for training and model selection, and
  (b) an ultimate hold-out set that we put into a vault.

  The vault is made inaccessible during model selection.
- It is not advisable to select from the 5 candidate models purely based on their performance on part (a) of the data (i.e. everything outside the vault).
- That (flawed) performance metric would be analogous to finding the minimum of the blue curve and will generally favor more complex models that are better able to "memorize" the structure of the training set.

- Various defensible procedures all share in common the characteristic that they evaluate performance of a model on data that was not used in training that model.
- If one had a good model of the true data-generating process, one could run Monte Carlo simulations to generate out-of-sample data, and the performance metric could be the average performance on simulated data sets.
- However, to execute this, one still needs to select a model to use, along with a random-number generator, to execute the simulations.

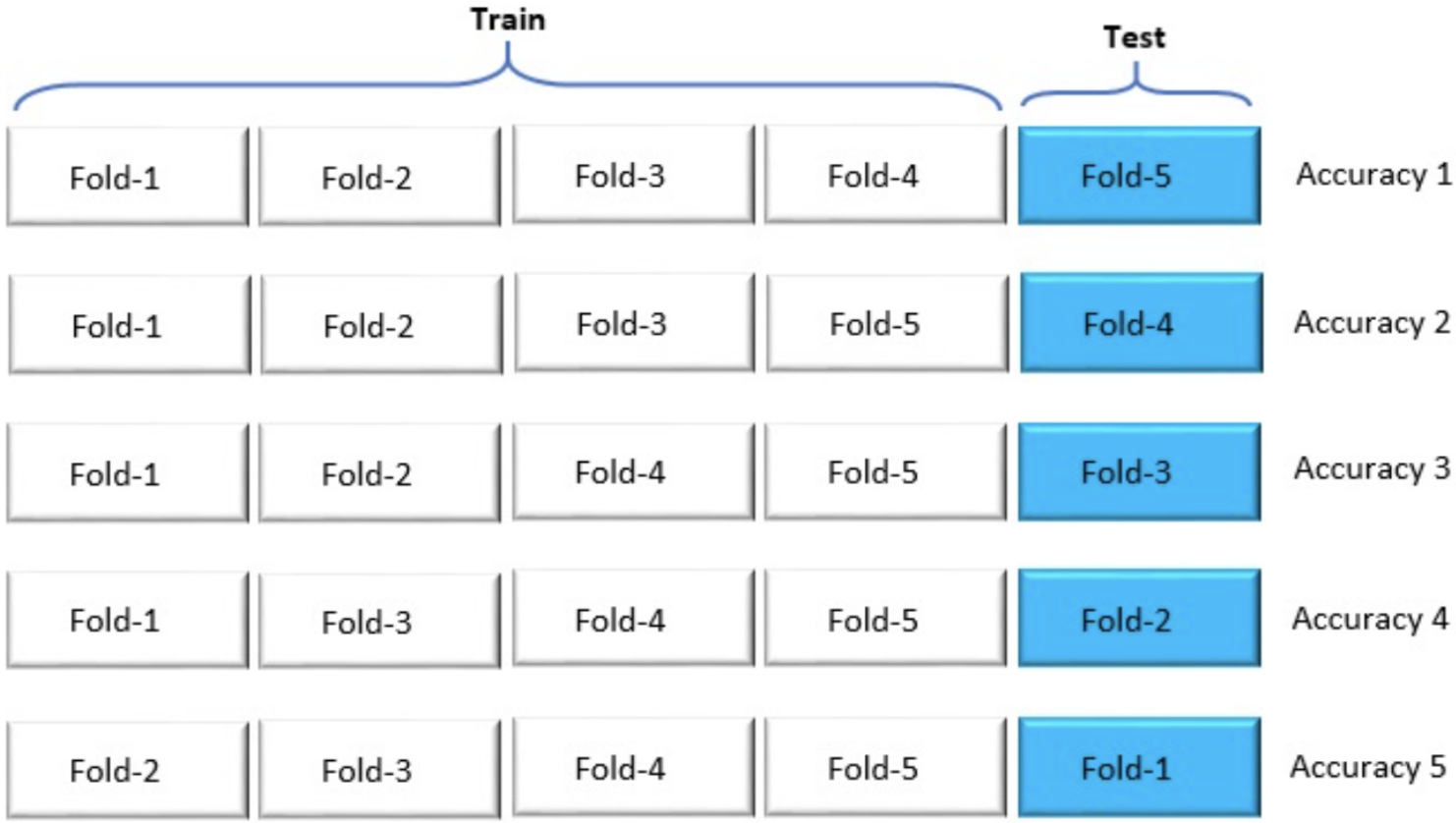
- Suppose instead that we split the training/selection data (part (a) from above) into $K = 5$ folds.


Dataset
Fold-1
Fold-2
Fold-3
Fold-4
Fold-5

- Then, for example, fold 5 is out-of-sample with respect to the union of folds 1 through 4.
- Similarly, fold 2 is out of sample with respect to the union of folds 1, 3, 4, 5.
- Each fold, j, is out-of-sample with respect to the union of folds

$$\text{Train}_j = \bigcup_{\substack{k=1\ldots K \\ k \neq j}} \text{Fold } k$$

which we suggestively name "Train$_j$" because it will be used as the training set when we evaluate out-of-sample on Fold j.



- Our performance metric will then be the average of the accuracies achieved on the different folds, labeled "Accuracy 1" through "Accuracy 5" in the diagram.

- Diagrams are good for intuition, but let's make the above more mathematically precise.
- Suppose that for all $i = 1, \ldots, N$, we know that sample i has been mapped to fold number k_i.
- For any fold number k, let

$$\hat{f}_{-k}(x)$$

denote the model, trained over all folds *except* for fold k, as in the diagram above.

- Then the cross-validation estimate of prediction error is

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}_{-k_i}(x_i)) \tag{2.26}$$

- In other words, when evaluating the i-th sample's loss metric, we are careful to use the model that was trained over the folds *not* containing the i-th sample.
- In other words, when evaluating for the i-th sample, use the model trained over all folds except for fold number k_i.
- Typical choices of K are 5 or 10 (see below).
- The case $K = N$ is known as leave-one-out cross-validation, and corresponds to $k_i = i$.

- Above we stated that estimation of $\mathrm{Err}_{\mathcal{T}}$ would be the goal in an ideal world, but Err is more amenable to estimation.
- Here we see how: $\mathrm{CV}(\hat{f})$ is an estimate of Err because it averages over many training sets, rather than working with a fixed training set.

- In most real supervised learning methods, there will typically be a tuning parameter α which roughly corresponds to model complexity.
- There can of course be multiple tuning parameters which tune different types of complexity.
- In such cases Err itself becomes a function of α; this is called the *test error curve*.
- As above we denote the model $\hat{f}_\alpha$ when it has tuning parameter α.
- In this situation, we can compute (2.26) for each model in this class of models, ie. compute $CV(\hat{f}_\alpha)$.
- This gives a function which maps tuning parameter α to a real number,

$$\alpha \mapsto CV(\hat{f}_\alpha)$$

- This function provides an estimate of the test error curve (the red curve).

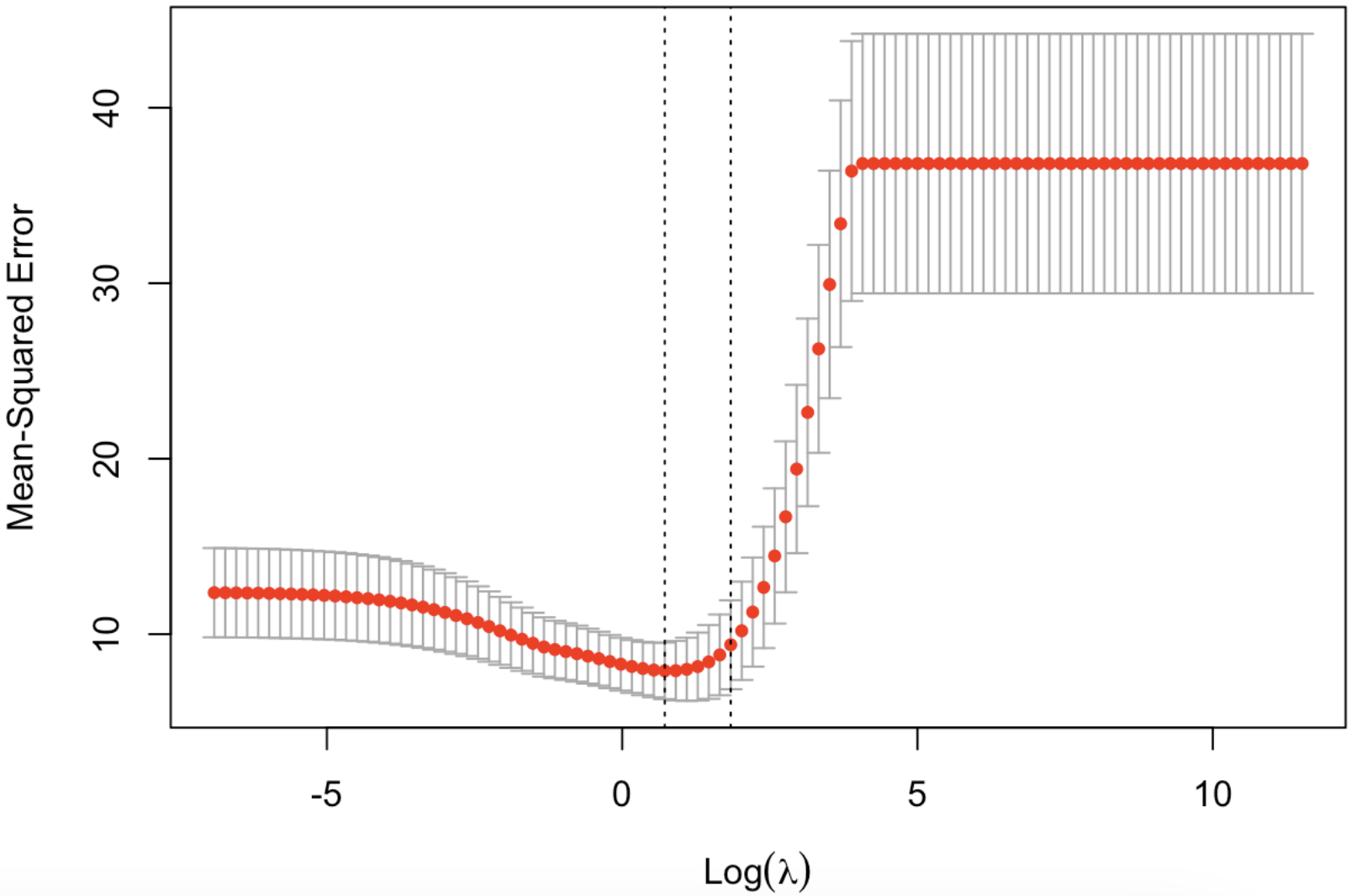
- The authors of the classic book Friedman, Hastie, and Tibshirani (2001) have provided a package in R which can reliably perform the cross-validation steps outlined above.
- Here is an example using a pre-packaged dataset for reproducibility.

```r
library(glmnet)
library(dplyr)
set.seed(42)  # folds are chosen by random sampling!

data("mtcars")
y <- mtcars %>% select(mpg) %>% scale(center = TRUE, scale = FALSE) %>% as.matrix()

X <- mtcars %>% select(-mpg) %>% as.matrix()

cv <- cv.glmnet(X, y,
                        alpha = 0.1,
                        lambda = 10^seq(-3, 5, length.out=100),
                        standardize = TRUE,
                        nfolds = 10)
plot(cv)
```


Mean-Squared Error
10
20
30
40
-5
0
5
10
Log(λ)

Problem 2.2

(a) Write a computer program which computes the LASSO solution path (for all λ in a given range) by implementing the suggested coordinate descent algorithm. Be sure to use the speedup suggested by (2.20) *and related.*
(b) Generate a simulated data set using a normal random number generator, under the following linear model with known coefficients

$$y = 3x_1 - 17x_2 + 5x_3 + \epsilon, \quad \epsilon \sim N(0, \sigma^2), \quad \sigma = 1$$

and check that on this simulated data set, you get the same solution path as the glmnet package in R.

Problem 2.3

In this problem we will reproduce the study whose results are reported in the Figure captioned "Behavior of test sample and training sample error as the model complexity is varied".
(a) Choose a known linear model of the form

$$y = \sum_{i=1}^{50} \beta_i x_i + \epsilon, \ \epsilon \sim N(0, \sigma^2)$$

Fix values of β and residual variance, which will represent the "true" values.
(b) Generate 100 training sets of size 50 each using the model from (a).
(c) For all λ in a reasonable range, fit Lasso to each of the 100 training sets, and thusly produce the blue curves.
(d) For each of the fits you did in part (c), (and hopefully you saved them!) estimate the out of sample variance by using the "true model" from part (a) to generate a large number of out-of-sample (x,y) pairs and calculate the prediction error variance of each fit.


Albert, Arthur (1972). *Regression and the Moore-Penrose pseudoinverse*. Academic Press.


Boyd, Stephen and Lieven Vandenberghe (2009). *Convex optimization*. Cambridge university press.


Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1, p. 1.


Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin.


Moore, E. H. (1920). "On the reciprocal of the general algebraic matrix". In: *Bulletin of the American Mathematical Society* 26, pp. 394–395.


Penrose, Roger (1955). "A generalized inverse for matrices". In: *Mathematical proceedings of the Cambridge philosophical society* 51.3, pp. 406–413.


Tibshirani, Robert (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.


Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.