Linear Regression
Linear Regression with Shrinkage

Introduction

- Regression means predicting a continuous (usually scalar) output *y* from a vector of continuous inputs (features) *x*.
- Example: Predicting vehicle fuel efficiency (mpg) from 8 attributes:

y	x				
	cyls	disp	hp	weight	. . .
18.0	8	307.0	130.00	3504	. . .
26.0	4	97.00	46.00	1835	. . .
33.5	4	98.00	83.00	2075	. . .
. . .					

Linear Regression

The linear regression model: $f(X) = w_0 + \sum_{j=1}^{M} w_j X_j$

The w are unknown parameters and X can come from different sources such as:

- Inputs
- Transformation of inputs (Log, Square root, etc…)
- Basis expansions

Linear Regression

Typically we have a set of training data $(x_1, y_1)...(x_n, y_n)$ from which to estimate the parameters w.

Each $x_i = (x_{i1},.., x_{iM})$ is a vector of measurements for ith case.

or

$h(x_i) = \{h_0(x_i),..., h_M(x_i)\}$ a basis expansion of x_i

$$y(x) \approx f(x; w) = w_0 + \sum_{j=1}^{M} w_j h_j(x) = w^t h(x)$$

(define $h_0(x) = 1$)

Basis Functions

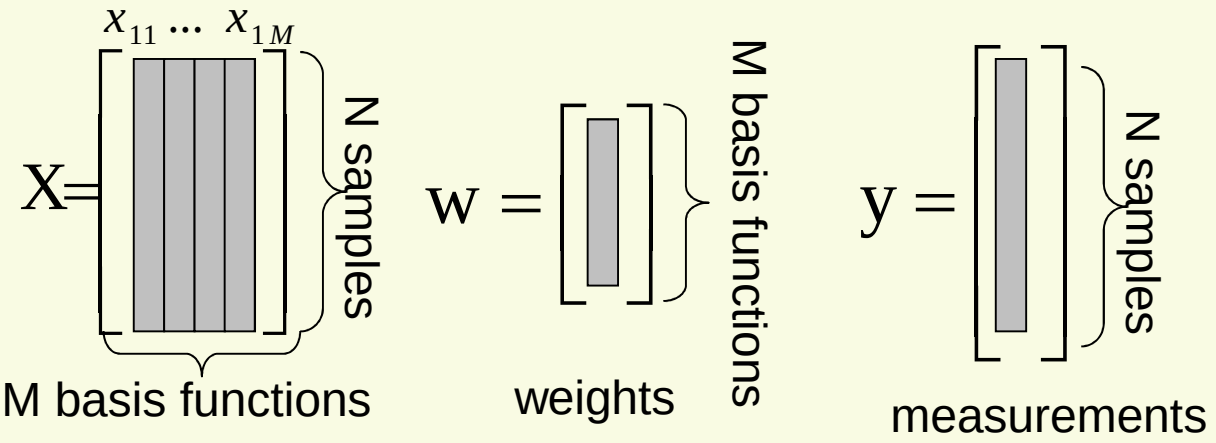
- There are many basis functions we can use e.g.
  - Polynomial $h_j(x) = x^{j-1}$
  - Radial basis functions $h_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$
  - Sigmoidal $h_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$
  - Splines, Fourier, Wavelets, etc

Linear Regression Estimation

- Minimize the residual error – prediction loss in terms of mean squared error on n training samples.

$$J_n(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(x_i; w)\right)^2$$ empirical squared loss

$$J_n(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{M} w_j x_j\right)^2 = \left(\mathrm{Xw} - \mathrm{y}\right)^t\left(\mathrm{Xw} - \mathrm{y}\right)$$



Linear Regression Solution

- By setting the derivatives of $(\mathrm{Xw}-\mathrm{y})^t(\mathrm{Xw}-\mathrm{y})$ to zero, we get the solution (as we did for MSE):

$$\hat{w}=\left(\mathrm{X}^{\mathrm{t}}\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{t}}\mathrm{y}$$

The solution is a linear function of the outputs y.

Statistical view of linear regression

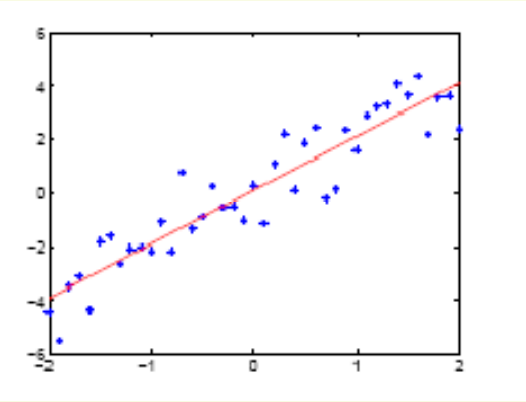
- In a statistical regression model we model both the function and noise

Observed output = function + noise

$$y(x) = f(x; w) + \varepsilon$$

where, e.g., $\varepsilon \sim N(0, \sigma^2)$

- Whatever we cannot capture with our chosen family of functions will be interpreted as noise

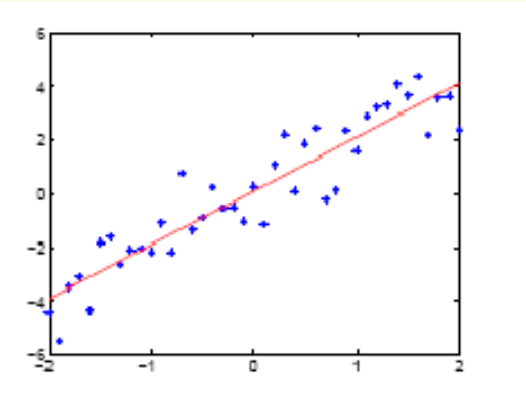


Statistical view of linear regression

- $f(x;w)$ is trying to capture the mean of the observations y given the input x:

$$E[y \mid x] = E[f(x;w) + \varepsilon \mid x] = f(x;w)$$

- where $E[y \mid x]$ is the conditional expectation of y given x, evaluated according to the model (not according to the underlying distribution of X)



Statistical view of linear regression

- According to our statistical model

$$y(x) = f(x;w) + \varepsilon, \;\; \varepsilon \sim N(0, \sigma^2)$$

the outputs y given x are normally distributed with mean $f(x;w)$ and variance σ^2:

$$p(y \mid x, w, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2} (y - f(x;w))^2 \right]$$

(we model the uncertainty in the predictions, not just the mean)

Maximum likelihood estimation

- Given observations $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ we find the parameters w that maximize the likelihood of the outputs:

$$L(w, \sigma^2) = \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma^2)$$

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_k - f\left(x_k ; w \right) \right)^2 \right\}$$

- Maximize log-likelihood

$$\log L(w, \sigma^2) = \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n - \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_k - f\left(x_k ; w \right) \right)^2 \right)$$

minimize

Maximum likelihood estimation

- Thus

$$w_{MLE} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - f\left(x_i; w\right)\right)^2$$

- But the empirical squared loss is

$$J_n(w) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f\left(x_i; w\right)\right)^2$$

Least-squares Linear Regression is MLE for Gaussian noise !!!

Linear Regression (is it "good"?)

- Simple model
- Straightforward solution

BUT

- **MLS is not a good estimator for prediction error**
- **The matrix X^TX could be ill conditioned**
  - **Inputs are correlated**
  - **Input dimension is large**
  - **Training set is small**

Linear Regression - Example

In this example the output is generated by the model (where ε is a small white Gaussian noise):

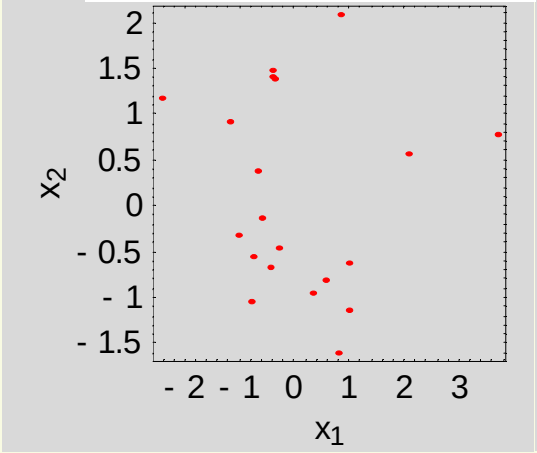
$$y = 0.8x_1 + 0.2x_2 + \varepsilon$$

Three training sets with different correlations between the two inputs were randomly chosen, and the linear regression solution was applied.

Linear Regression – Example

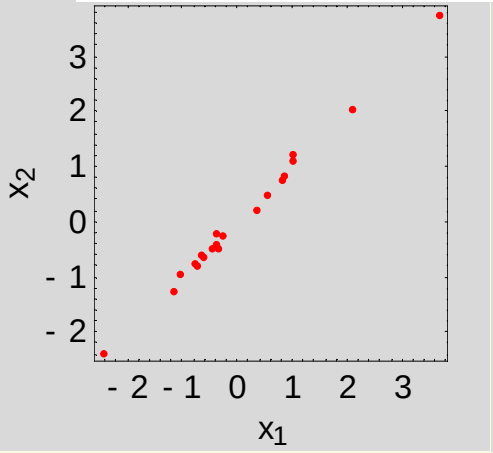
And the results are…

RSS=0.29
w=0.82, 0.209



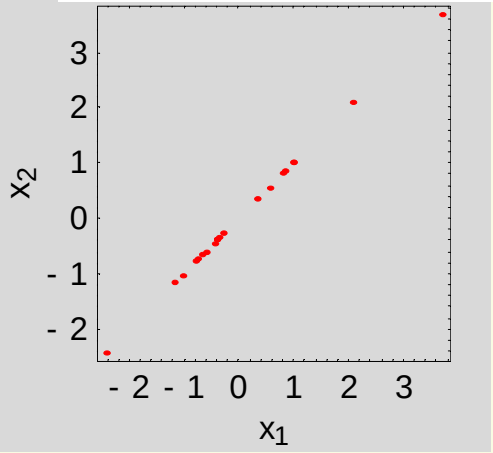
x_1,x_2 uncorrelated

RSS=0.204
w=0.42, 0.56



x_1,x_2 correlated

RSS=0.47
w=30.36, -29.21



x_1,x_2 strongly correlated

Strong correlation can cause the coefficients to be very large, and they will cancel each other to achieve a good *RSS*.

Linear Regression

What can be done?

Shrinkage methods

- Ridge Regression
- Lasso
- PCR (Principal Components Regression)
- PLS (Partial Least Squares)

Shrinkage methods

Before we proceed:

- Since the following methods are not invariant under input scale, we will assume the input is standardized (mean 0, variance 1):

$$x'_{ij} \leftarrow x_{ij} - \bar{x}_j \qquad\qquad x_{ij} \leftarrow \frac{x'_{ij}}{\sigma_j}$$

- The offset w_0 is always estimated as $w_0 = (\sum_i y_i) / N$ and we will work with centered y (meaning $y \leftarrow y\text{-}w_0$)

Ridge Regression

- Ridge regression shrinks the regression coefficients by imposing a penalty on their size (also called weight decay)
- In ridge regression, we add a quadratic penalty on the weights:

$$J(w) = \sum_{i=1}^{N} \left(y_i - w_0 - \sum_{j=1}^{M} x_{ij} w_j \right)^2 + \lambda \sum_{j=1}^{M} w_j^2$$

where $\lambda \geq 0$ is a tuning parameter that controls the amount of shrinkage.

- The size constraint prevents the phenomenon of wildly large coefficients that cancel each other from occurring.

Ridge Regression Solution

- Ridge regression in matrix form:

$$J(w) = (y - Xw)^t (y - Xw) + \lambda w^t w$$

- The solution is

$$\hat{w}^{ridge} = (X^t X + \lambda I_M)^{-1} X^t y$$

$$\hat{w}^{LS} = (\mathrm{X}^t \mathrm{X})^{-1} \mathrm{X}^t y$$

- The solution adds a positive constant to the diagonal of $X^T X$ before inversion. This makes the problem non singular even if X does not have full column rank.
- For orthogonal inputs the ridge estimates are the scaled version of least squares estimates:

$$\hat{w}^{ridge} = \gamma \, \hat{w}^{LS} \quad 0 \le \gamma \le 1$$

Ridge Regression (insights)

The matrix X can be represented by it's SVD:

$$X = UDV^T$$

- U is a $\mathrm{N}*\mathrm{M}$ matrix, it's columns span the column space of X
- D is a $\mathrm{M}*\mathrm{M}$ diagonal matrix of singular values
- V is a $\mathrm{M}*\mathrm{M}$ matrix, it's columns span the row space of X

Lets see how this method looks in the Principal Components coordinates of X

Ridge Regression (insights)

$$X' = XV$$

$$Xw = (XV)(V^T w) = X'w'$$

The least squares solution is given by:

$$\hat{w}'^{ls} = V^T \hat{w}^{ls} = V^T \left(VDU^T UDV^T\right)^{-1} VDU^T y = D^{-1}U^T y$$

The Ridge Regression is similarly given by:

$$\hat{w}'^{ridge} = V^T \hat{w}^{ridge} = V^T \left(VDU^T UDV^T + \lambda I\right)^{-1} VDU^T y =$$

$$= \left(D^2 + \lambda I\right)^{-1} DU^T y = \left(D^2 + \lambda I\right)^{-1} D^2 \hat{w}'^{ls}$$

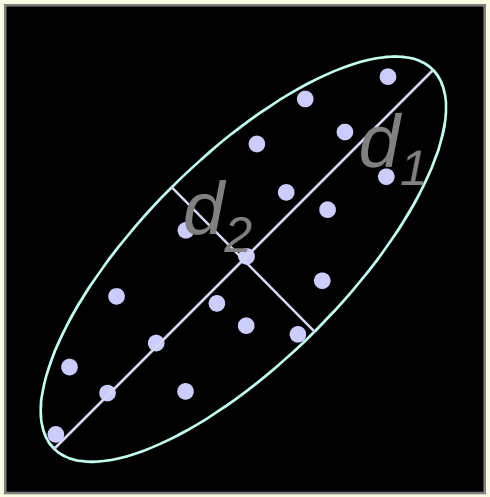
Diagonal

Ridge Regression (insights)

$$\hat{w}_j^{ridge} = \frac{d_j^2}{d_j^2 + \lambda} \hat{w}_j^{ls}$$

In the PCA axes, the ridge coefficients are just scaled LS coefficients!

The coefficients that correspond to smaller input variance directions are scaled down more.



Ridge Regression

In the following simulations, quantities are plotted versus the quantity

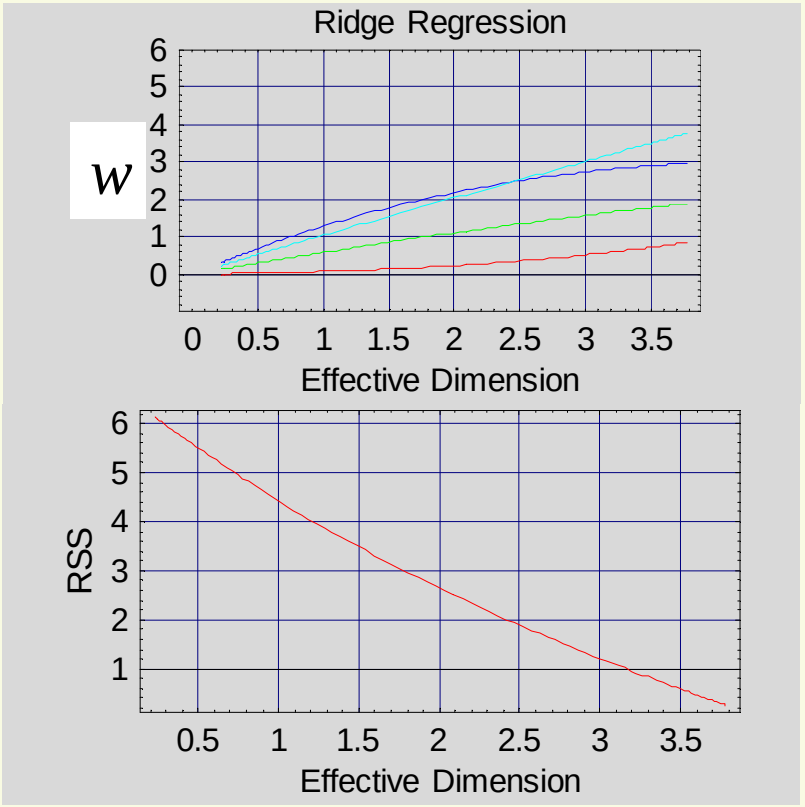
$$df(\lambda) = \sum_{j=1}^{M} \frac{d_j^2}{d_j^2 + \lambda}$$

This monotonic decreasing function is the *effective degrees of freedom* of the ridge regression fit.

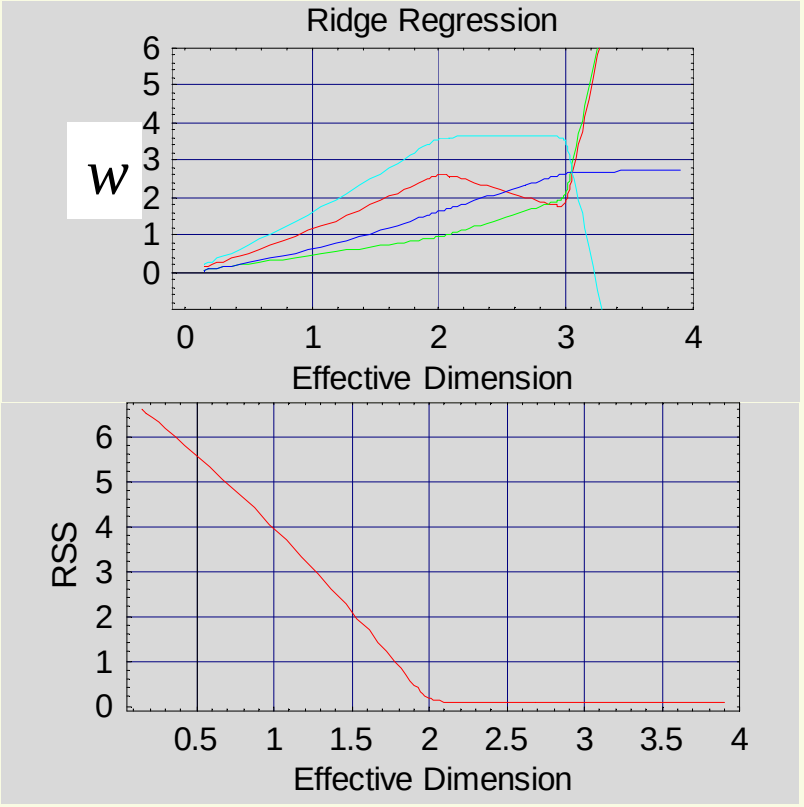
Ridge Regression

Simulation results: 4 dimensions (w=1,2,3,4)

No correlation



Strong correlation



Ridge regression is MAP with Gaussian prior

$$J(w) = -\log P(D \mid w) P(w)$$

$$= -\log\left[\prod_{i=1}^{n} N(y_i \mid w^t x_i, \sigma^2) N(w \mid 0, \tau^2)\right]$$

$$= \frac{1}{2\sigma^2}(y - Xw)^t (y - Xw) + \frac{1}{2\tau^2} w^t w + const$$

This is the same objective function that ridge solves, using $\lambda = \sigma^2 / \tau^2$

Ridge: $J(w) = (y - Xw)^t (y - Xw) + \lambda w^t w$

Lasso

Lasso is a shrinkage method like ridge, with a subtle but important difference. It is defined by

$$\hat{w}^{lasso} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{M} x_{ij} w_j \right)^2 \right\}$$

$$\text{subject to} \sum_{j=1}^{M} \left| w_j \right| \leq t$$

There is a similarity to the ridge regression problem: the L_2 ridge regression penalty is replaced by the L_1 lasso penalty.

The L_1 penalty makes the solution non-linear in y and requires a quadratic programming algorithm to compute it.

Lasso

If t is chosen to be larger then t_0: $t \geq t_0 \equiv \sum_{j=1}^{M} \left| \hat{w}^{ls} \right|$

then the lasso estimation is identical to the least squares. On the other hand, for say $t=t_0/2$, the least squares coefficients are shrunk by about 50% on average.

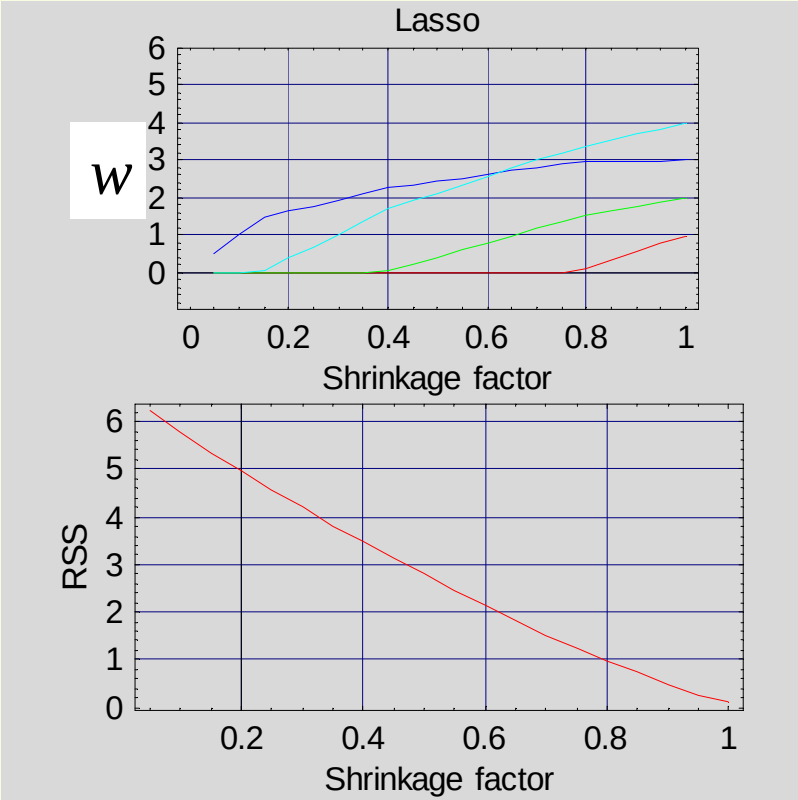
For convenience we will plot the simulation results versus shrinkage factor s:

$$s \equiv \frac{t}{t_0} = \frac{t}{\sum_{j=1}^{M} \left| \hat{w}^{ls} \right|}$$

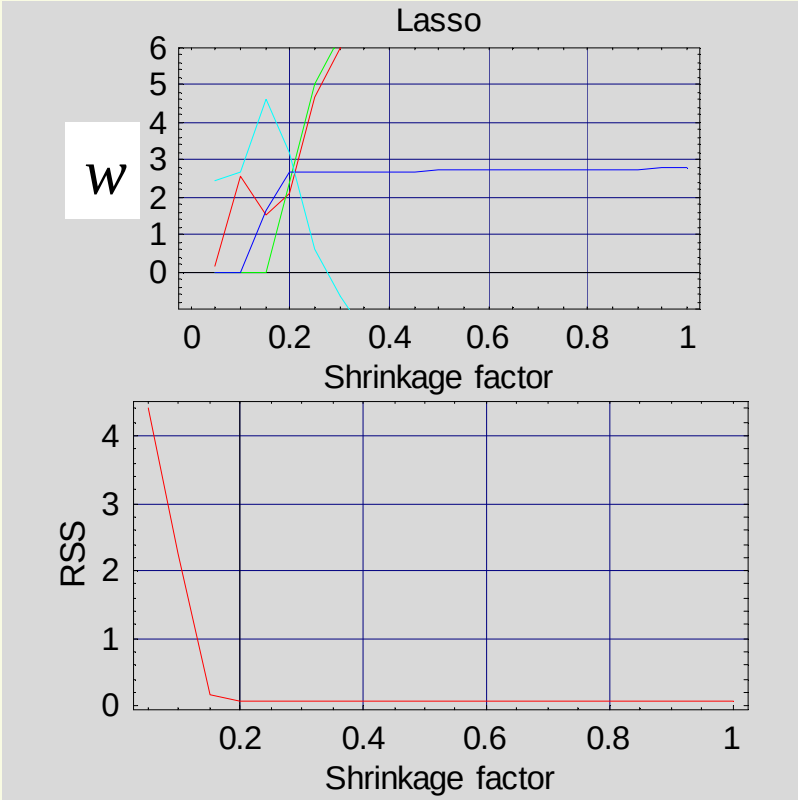
Lasso

Simulation results:

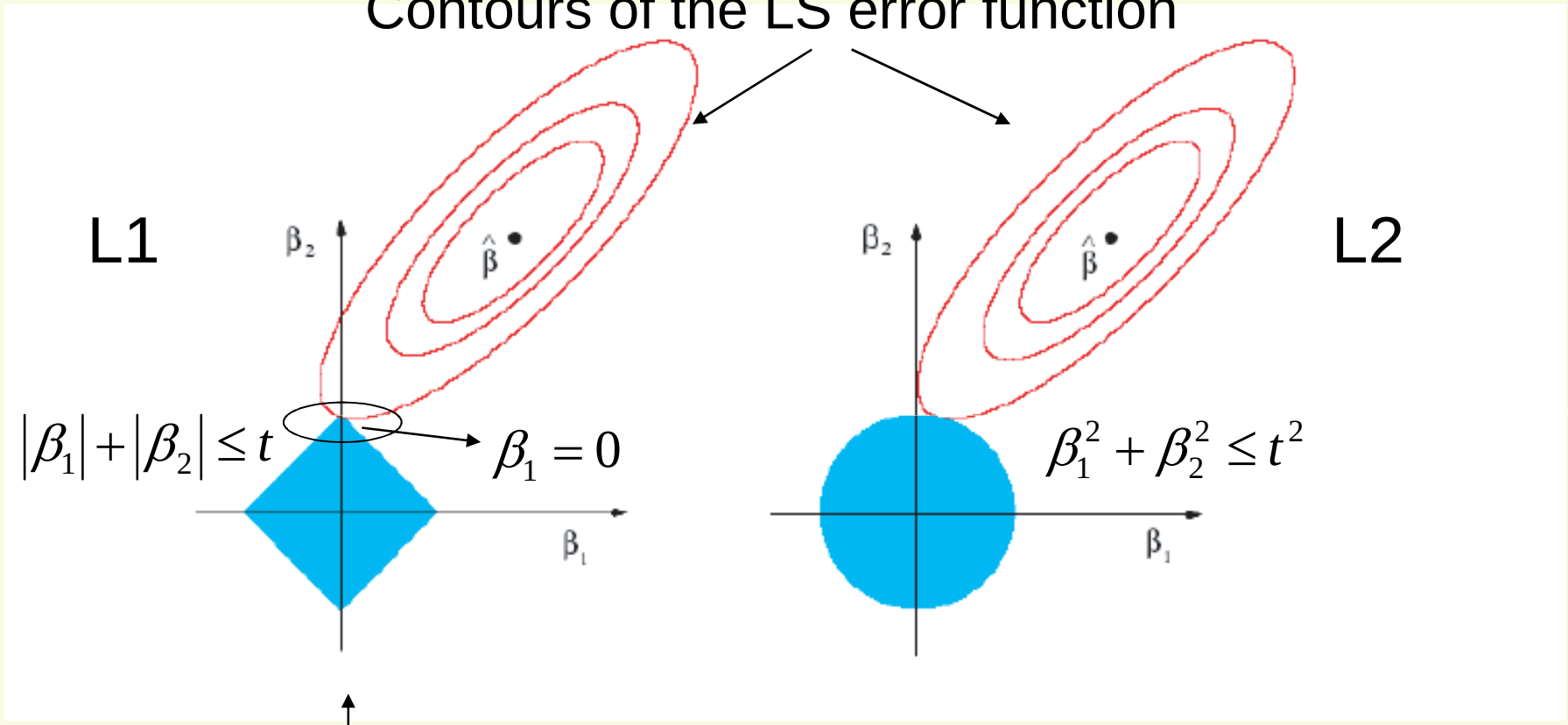
No correlation



Strong correlation



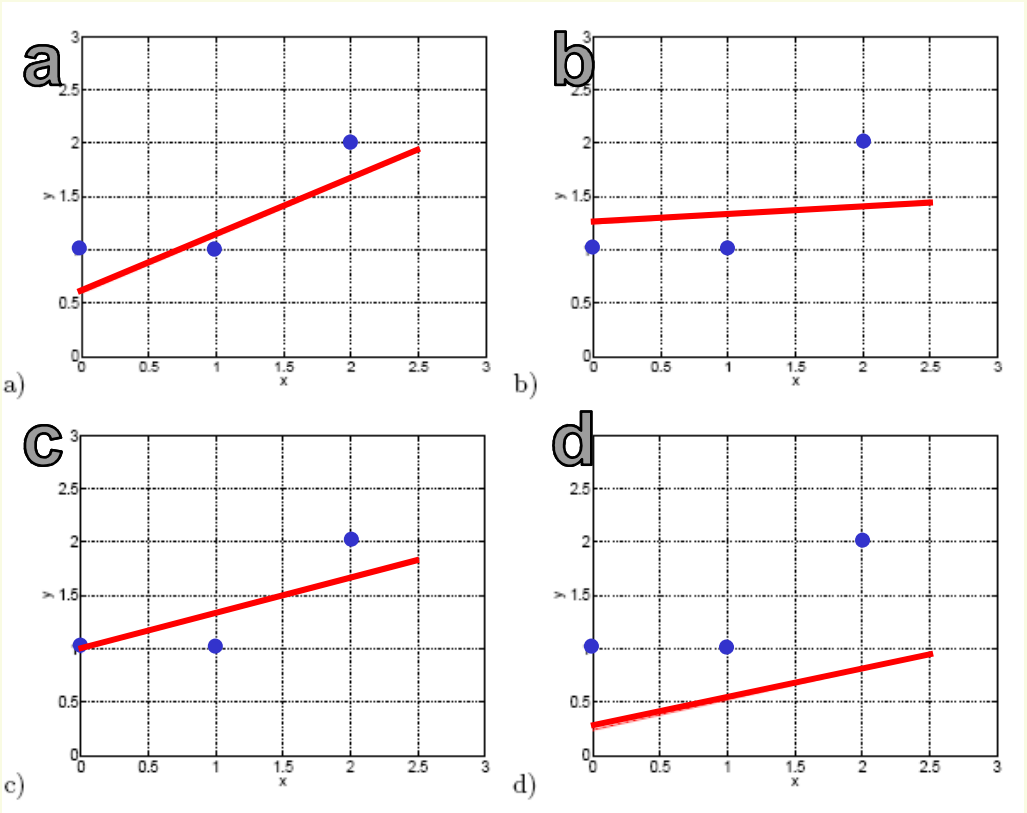
L2 vs L1 penalties



In Lasso the constraint region has corners; when the solution hits a corner the corresponding coefficients becomes 0 (when M>2 more than one).

Problem:

Figure 1 plots linear regression results on the basis of only three data points. We used various types of regularization to obtain the plots (see below) but got confused about which plot corresponds to which regularization method. Please assign each plot to one (and only one) of the following regularization method.



c

$$\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda\theta^2 \text{ where } \lambda = 1$$

b

$$\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda\theta^2 \text{ where } \lambda = 10$$

a

$$\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda(\theta^2 + \theta_0^2) \text{ where } \lambda = 1$$

d

$$\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda(\theta^2 + \theta_0^2) \text{ where } \lambda = 10$$

Consider a regression problem where the two dimensional input points $\mathbf{x} = [x_1, x_2]^T$ are constrained to lie within the unit square: $x_i \in [-1, 1]$, $i = 1, 2$. The training and test input points $\mathbf{x}$ are sampled uniformly at random within the unit square. The target outputs y are governed by the following model

$$y \sim N(x_1^3 x_2^5 - 10 x_1 x_2 + 7 x_1^2 + 5 x_2 - 3,\ 1)$$

In other words, the outputs are normally distributed with mean given by

$$x_1^3 x_2^5 - 10 x_1 x_2 + 7 x_1^2 + 5 x_2 - 3$$

and variance 1.

We learn to predict y given $\mathbf{x}$ using linear regression models with 1st through 10th order polynomial features. The models are nested in the sense that the higher order models will include all the lower order features. The estimation criterion is the mean squared error.

We first train a 1st, 2nd, 8th, and 10th order model using $n = 20$ training points, and then test the predictions on a large number of independently sampled points.

$$y \sim N(x_1^3 x_2^5 - 10 x_1 x_2 + 7 x_1^2 + 5 x_2 - 3,\ 1)$$

Select all the appropriate models for each column.

	Lowest training error	Highest training error	Lowest test error (typically)
1st order		X	
2nd order			X
8th order	X		
10th order	X		

Briefly explain your selection in the last column, i.e., the model you would expect to have the lowest test error:

$$y \sim N(x_1^3 x_2^5 - 10 x_1 x_2 + 7 x_1^2 + 5 x_2 - 3,\ 1)$$

Select all the appropriate models for each column.

	Lowest training error	Highest training error	Lowest test error (typically)
1st order		X	
2nd order			X
8th order	X		
10th order	X		

Briefly explain your selection in the last column, i.e., the model you would expect to have the lowest test error:

The 10th order regression model would seriously overfit when presented only with $n = 20$ training points. The second order model on the other hand might find some useful structure in the data based only on 20 points. The true model is also dominated by the second order terms. Since $|x_1| \leq 1$ and $|x_2| \leq 1$ any higher order terms without large coefficients are vanishingly small.