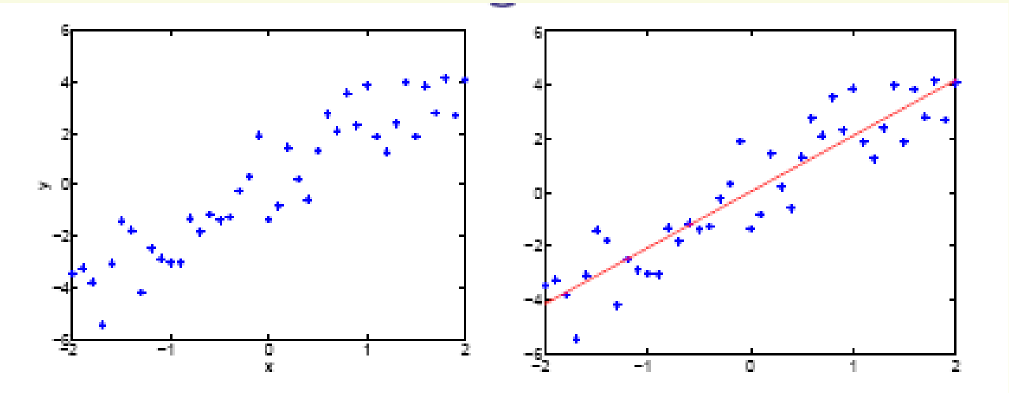
Linear Regression
Linear Regression with Shrinkage

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



- **Instances:** <$\mathbf{x}_j$, y_j>
- **Learn:** Mapping from x to y(**x**)
- Given, basis functions, $h(x) = \{h_0(x), \ldots, h_M(x)\}$,

  $(\text{define}\, h_0(x) = 1)$
- Find coefficients $w = \{w_0, \ldots, w_M\}$

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

data

assumes the functional mapping is linear in its M parameters *w*

Basis Functions

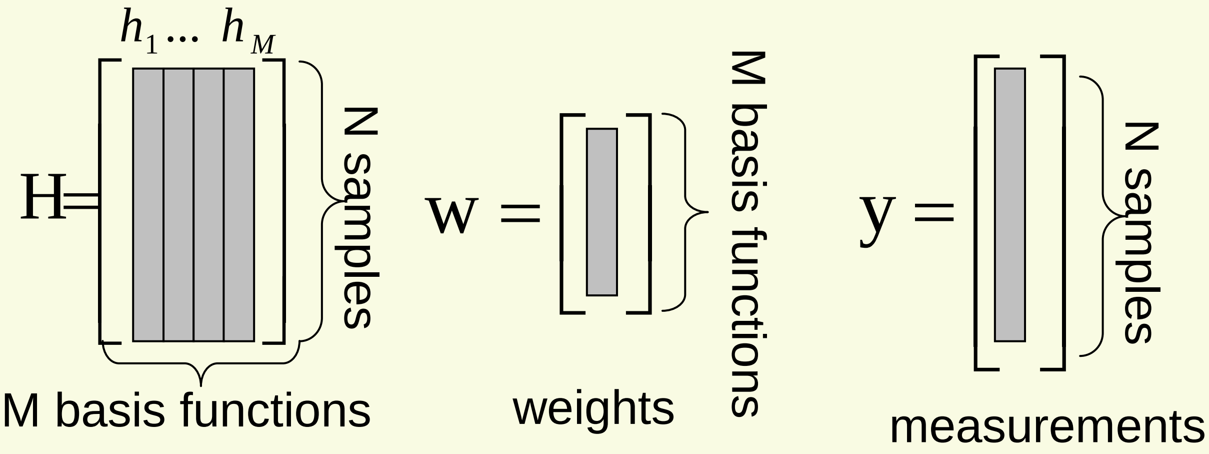
- There are many basis functions we can use eg
  - Polynomial $h_j(x) = x^{j-1}$
  - Radial basis functions $h_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2} \right)$
  - Sigmoidal $h_j(x) = \sigma\left(\frac{x-\mu_j}{s} \right)$
  - Splines, Fourier, Wavelets, etc

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

$$J_n(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_j w_j h_j(x_i)\right)^2 = \left(\mathrm{Hw}-\mathrm{y}\right)^t\left(\mathrm{Hw}-\mathrm{y}\right)$$

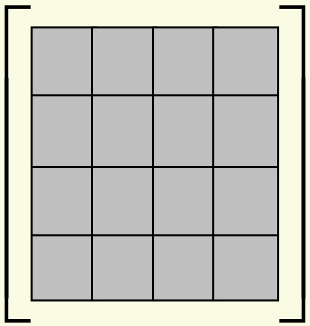
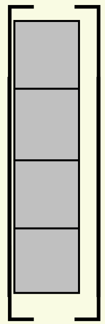


Linear Regression Solution

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

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

The solution is a linear function of the outputs y.

$\text{where } \mathrm{A} = \mathrm{H}^{t}\mathrm{H} =$  $\mathrm{b} = \mathrm{H}^{t}\mathrm{y} =$ 

MxM matrix of basis functions

Mx1 vector

Statistical view of linear regression

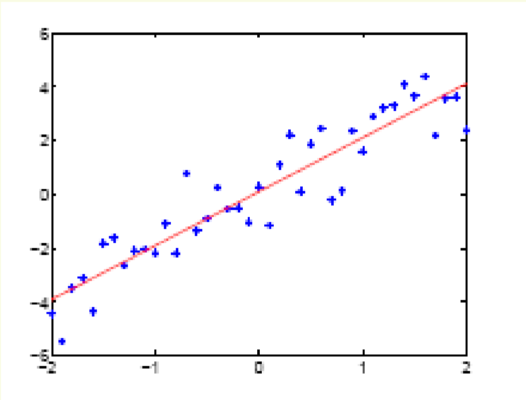
- In a statistical regression model we model both the function and noise

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 P)

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

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

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

- Maximize log-likelihood

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

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

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

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

Pseudo Inverse

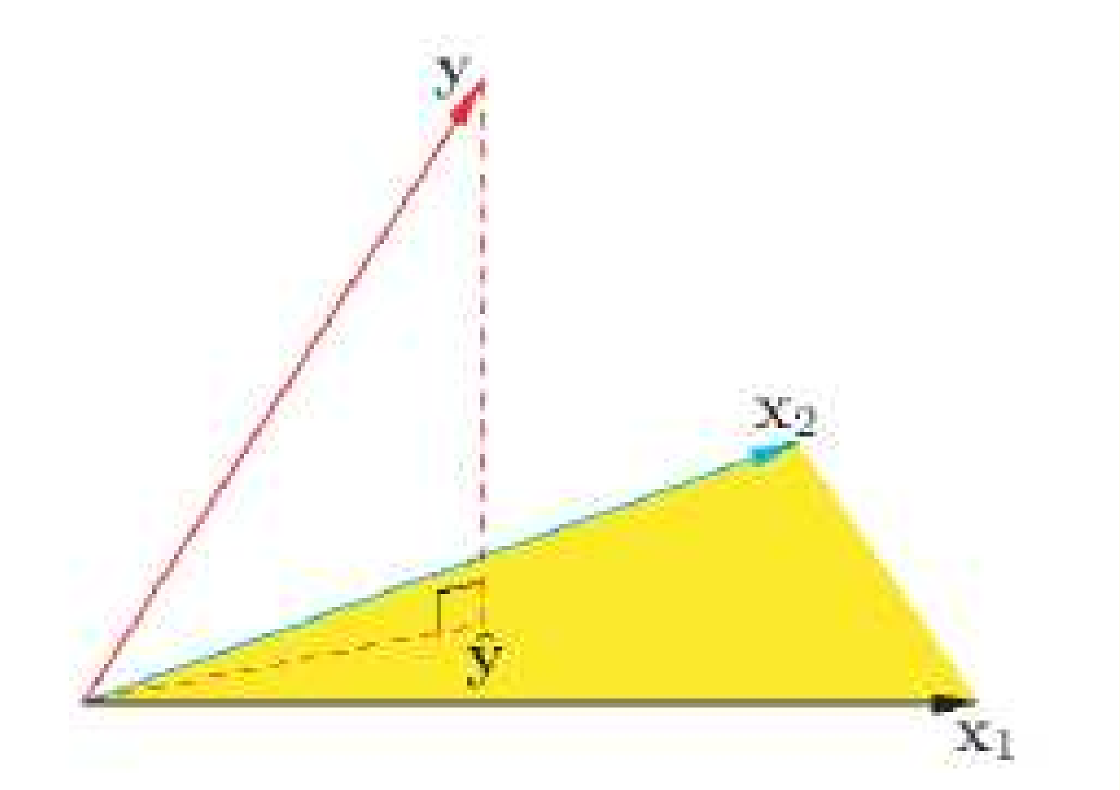
$$\hat{w} = \left(\mathrm{H}^{t}\mathrm{H}\right)^{-1}\mathrm{H}^{t}\mathrm{y} = \mathrm{A}^{-1}\mathrm{b}$$

- $\left(\mathrm{H}^{t}\mathrm{H}\right)^{-1}$ is called the pseudo-inverse of H (since H will not usually be square).
- The predictions on the training data are

$$\hat{y} = \mathrm{H}^{t}\hat{w} = \mathrm{H}^{t}(H^{t}H)^{-1}H^{t}y \equiv Sy$$

- where *S* is called the "hat" matrix. This computes an orthogonal projection of y into the space spanned by the columns of H

Geometric interpretation of linear regression with two input points



Numerical issues in computing A^{-1}

- Recall that H is an $M \times n$ matrix.
- If $n < M$ or if some of the columns (features) are colinear, then A is not full rank (so det(A) = 0)
- Even if A is singular, $\hat{y} = \mathrm{H}^{t}\hat{w}$ is still the projection of y onto the column space of H
  - there is just more than one way to express that projection in terms of the columns of H (the model is unidentifiable).

- How do we compute A^{-1} if it is not of full rank?
  - Use SVD
  - Use regularization

SVD FOR NON SQUARE MATRICES

- A is $m \times n$, ie m equations and n unknowns.
- If $m > n$, the system is over-determined. SVD will find the least squares solution. If there are degenerate columns in A (due to colinearity), you should set small σ_j's to 0 before inverting.
- If $m < n$, then there is an $n - m$ dimensional family of solutions. SVD will set $n - m$ σ_j's to 0. (If there are degeneracies in A, you should set small σ_j's to 0.)
- In both cases, pinv will do the right thing.

Linear regression with regularization

- If there are correlated features, their coefficients might become poorly determine and exhibit high variance.
- A large positive coefficient on one variable can be canceled by a similarly large negative coefficient on its correlated cousin.
- Solutions:
  - Select a subset of strong inputs – subset selection
  - Add a regularization term to control weights.
  - Methods using Derived Input Directions

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 λ ≥ 0 is a tuning parameter that controls the amount of shrinkage.

- This is equivalent to

$$\hat{w}^{ridge} = \arg\min_{w} \sum_{i=1}^{N}\left(y_i - w_0 - \sum_{j=1}^{M} x_{ij} w_j\right)^2 \text{ subject to } \sum_{j=1}^{M} w_j^2 \le s$$

  where s is related to λ

Standardizing

- In ridge regression, we add a quadratic penalty to all the weights except the offset w_0

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

- We do not penalize the bias term w_0, since we want a shift in input to shift the output by the same amount.
  - We can estimate the offset w_0 by $\bar{y} = (\sum_i y_i) / N$
  - The remaining coefficients are estimated using ridge regression without w_0, using the centered data $x_{ij} - \bar{x}_j$
  - Now the input matrix X (centered) has M (not M+1) columns.
- Since ridge is not invariant to scaling of inputs, we usually also standardize the inputs, i.e., we use

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$$

Ridge Regression Solution

- Ridge regression in matrix form:

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

where $Z_{ij} = (X_{ij} - \overline{X}_{ij})$ is the centered matrix

- The solution is

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

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

- The problem is non singular even if $Z^t Z$ is not full rank.
- Still linear in y.
- For orthogonal inputs the ridge estimates are the scaled version of least squares estimates:

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

SVD and LS

- Assume X is centered. Let the SVD be $X = UDV^t$, where
  - U is nxM orthogonal matrix with its columns spanning the column space of X
  - V is Mxn orthogonal matrix with its columns spanning the row space of X
  - D is MxM diagonal matrix with diagonal entries $d_1 \geq d_2 \geq \ldots d_M \geq 0$ called the singular values of X.
- It is easy to show (do it!) that the predictions on the training set are

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

Ridge and SVD

- The ridge solutions are

$$X\,\hat{w}^{ridge} = X(X^tX+\lambda I)^{-1}X^ty$$

$$= UD(D+\lambda I)^{-1}DU^ty = \sum_{j=1}^{M} u_j \frac{d_j^2}{d_j^2+\lambda} u_j^t y$$

where u_j are the columns of U.

- Note $\lambda \geq 0,\ d_j^2/(d_j^2+\lambda) \leq 1.$
- Like linear regression, ridge computes the coordinates of y with respect to the orthonormal basis U. It then shrinks these coordinates by the factor of $d_j^2/(d_j^2+\lambda)$
- Thus the greater shrinkage is applied to basis vectors with smaller d_j^2. What does small d_j^2 mean?

PCA and Ridge

- If $X = UDV^t$, , then the Eigen decomposition of the sample covariance matrix is

$$X^t X = VD^2 V$$

- The eigenvectors v_j are the principle components directions of X. The first principle component

$$z_1 = Xv_1 = u_1 d_1$$

has the largest variance

$$Var(z_1) = Var(Xv_1) = d_1^2 / n$$

- Hence small singular values d_j correspond to directions in the column spaces of X having small variance, and ridge shrinks these directions the most.

PCA and Ridge

- It is easier to determine the gradient of the plane in the long direction than the short.
- Ridge protects against potentially high variance of gradient estimates in the short direction.

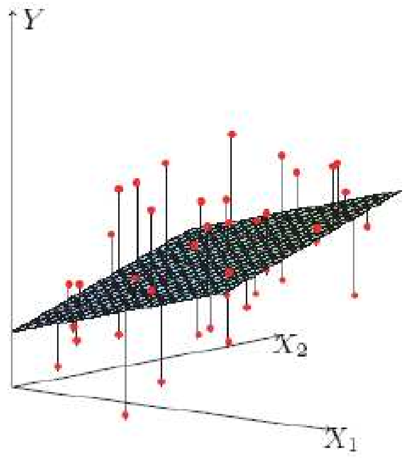
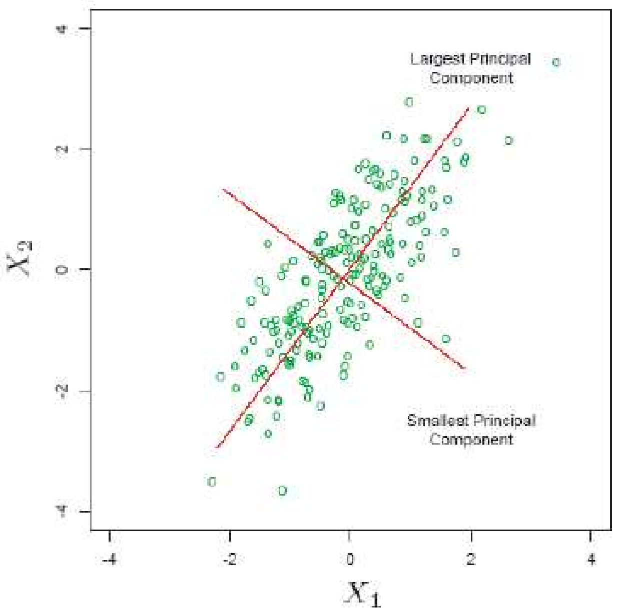


Figure 3.1: *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*



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

The Lasso (L1-Penalty)

- Lasso (least absolute shrinkage and selection operator) uses an *L*1 penalty on the weights

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

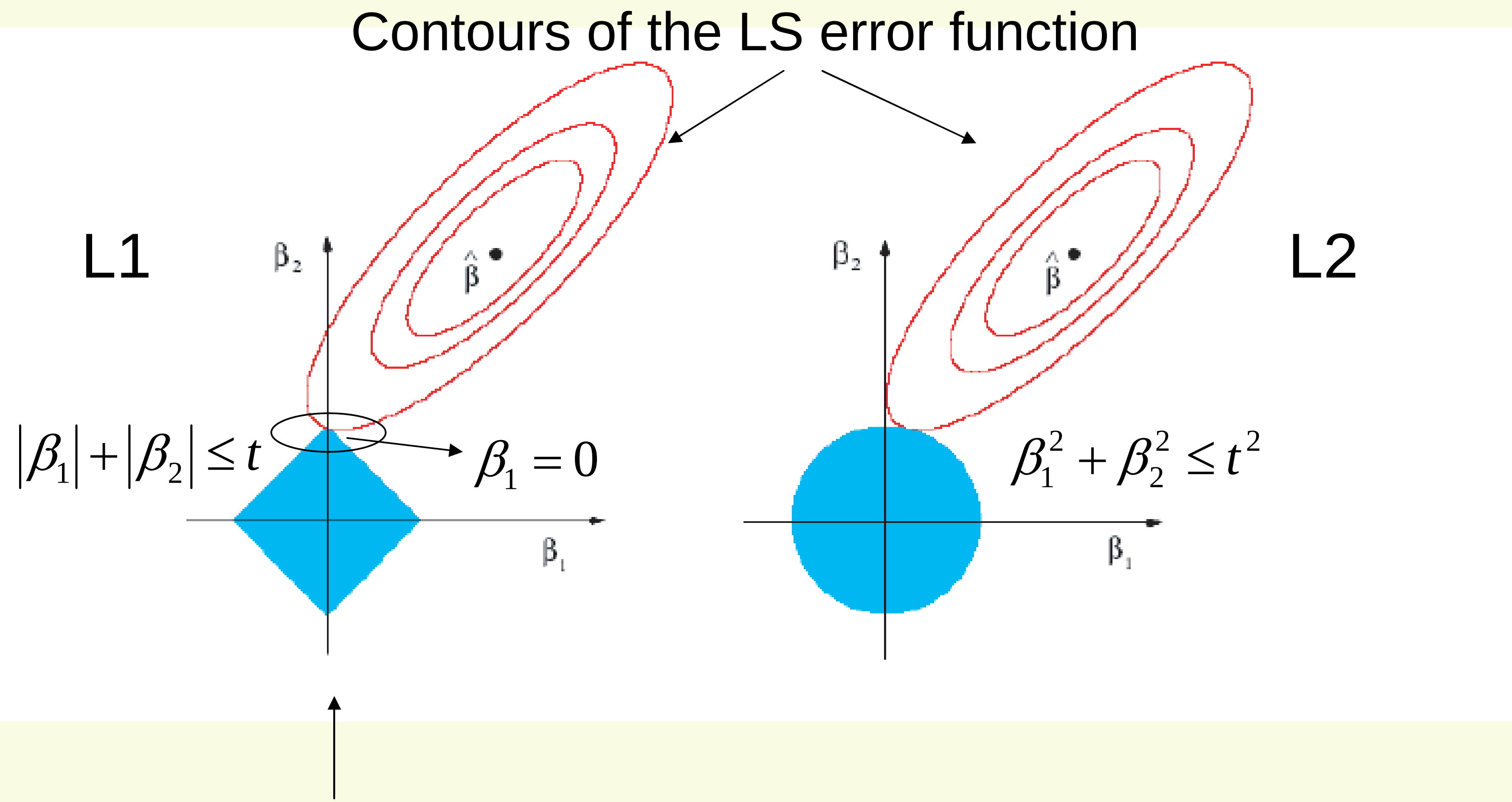
- This is equivalent to

$$\hat{w}^{ridge} = \arg\min_{w} \sum_{i=1}^{N}\left(y_i - w_0 - \sum_{j=1}^{M} x_{ij} w_j \right)^2 \text{ subject to } \sum_{j=1}^{M} \left| w_j \right| \leq t$$

where t is related to λ

- This encourages sparcity, i.e., some weights go exactly to 0.
- It is like soft feature selection.
- However, we must now use quadratic programming (or iterative methods).

L2 vs L1 penalties

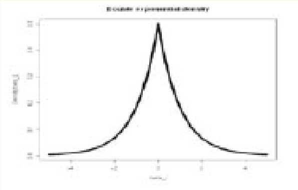


In Lasso the constraint region has corners; when the solution hits a corner the corresponding coefficients becomes 0 (when M>2 more than one).

Lasso is MAP with Laplace prior

- Consider a double-sided exponential prior

$$P(w) = \prod_{i=1}^{M} \text{Laplace}(w_i \mid \alpha) = \prod_{i=1}^{M} \frac{\alpha}{2} \exp(-\alpha |w_i|) = \left(\frac{\alpha}{2}\right)^{M} \exp(-\alpha |w|_1)$$



- Then the MAP estimate minimizes

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

$$= \frac{1}{2\sigma^2}(y - Xw)^t (y - Xw) + \alpha \sum_{i=1}^{M} |w_i| + const$$

- This is the same objective function that lasso solves, using $\lambda = 2\sigma^2 \alpha$

Examples

- See examples of regression at http://en.wikipedia.org/wiki/Linear_regression