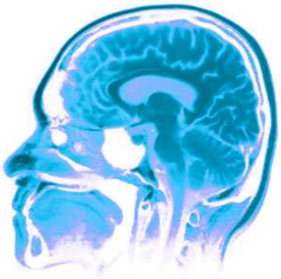
CPSC540



Linear Predictive Models

Nando de Freitas
September, 2011
University of British Columbia

Outline

Linear regression and prediction are simple supervised learning tasks. But,

- Many real processes can be approximated with linear models.
- Linear regression often appears as a module of larger systems.
- Linear problems can be solved analytically.
- Linear prediction provides an introduction to many of the core concepts of machine learning, while still allowing for analytical tractability.

We are given a training dataset of n instances of input-ouput pairs $\{\mathbf{x}_{1:n}, \mathbf{y}_{1:n}\}$. Each input $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ is a vector with d attributes. The inputs are also known as predictors or covariates. The output, often referred to as the target, will be assumed to be univariate, $\mathbf{y}_i \in \mathbb{R}$, for now.

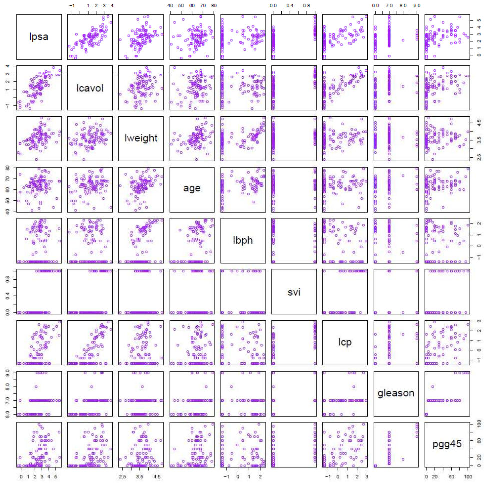
A typical dataset with $n = 4$ instances and 2 attributes would look like the following table:

Wind speed	People inside building	Energy requirement
100	2	5
50	42	25
45	31	22
60	35	18

Given the training set $\{\mathbf{x}_{1:n}, \mathbf{y}_{1:n}\}$, we would like to learn a model of how the inputs affect the outputs. Given this model and a new value of the input $\mathbf{x}_{n+1}$, we can use the model to make a prediction $\widehat{y}(\mathbf{x}_{n+1})$.

Prostate cancer example

❑ Goal: Predict a prostate-specific antigen (log of lpsa) from a number of clinical measures in men who are about to receive a radical prostatectomy.



[Hastie, Tibshirani & Friedman book]

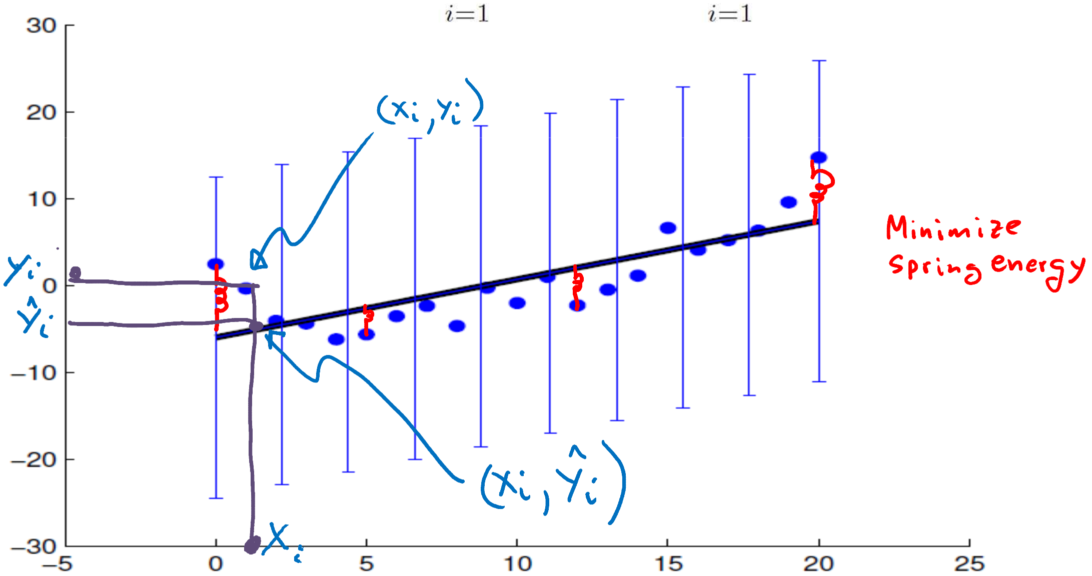
❑The inputs are:

- Log cancer volume (lcavol)
- Log prostate weight (lweight)
- Age
- Log of the amount of benign prostatic hyperplasia (lbph)
- Seminal vesicle invasion (svi) - *binary*
- Log of capsular penetration (lcp)
- Gleason score (gleason) – *ordered categorical*
- Percent of Gleason scores 4 or 5 (pgg45)

Linear regression in 1D

$$\hat{y}(\mathbf{x}_i) = \theta_1 - x_i\theta_2$$

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \theta_1 - x_i\theta_2)^2$$



Linear prediction

In general, the linear model is expressed as follows:

$$\widehat{y}_i = \sum_{j=1}^{d} x_{ij}\theta_j,$$

where we have assumed that $x_{i1} = 1$ so that θ_1 corresponds to the intercept of the line with the vertical axis. θ_1 is known as the bias or offset.

In matrix form, the expression for the linear model is:

$$\widehat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta},$$

with $\widehat{\mathbf{y}} \in \mathbb{R}^{n\times 1}$, $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{\theta} \in \mathbb{R}^{d\times 1}$. That is,

$$\begin{bmatrix} \widehat{y}_1 \\ \vdots \\ \widehat{y}_n \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}.$$

Linear prediction

Wind speed	People inside building	Energy requirement
100	2	5
50	42	25
45	31	22
60	35	18

For our energy prediction example, we would form the following matrices with $n = 4$ and $d = 3$:

$$\mathbf{y} = \begin{bmatrix} 5 \\ 25 \\ 22 \\ 18 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & 100 & 2 \\ 1 & 50 & 42 \\ 1 & 45 & 31 \\ 1 & 60 & 35 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}.$$

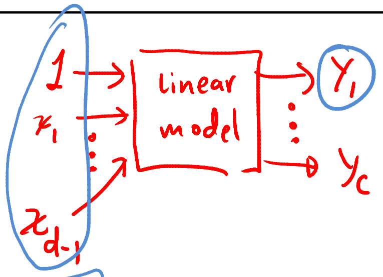
Suppose that $\boldsymbol{\theta} = [1 \quad 0 \quad 0.5]^T$. Then, by multiplying $\mathbf{X}$ times $\boldsymbol{\theta}$, we would get the following predictions on the training set:

$$\widehat{\mathbf{y}} = \begin{bmatrix} 2 \\ 22 \\ 16.5 \\ 18.5 \end{bmatrix} = \begin{bmatrix} 1 & 100 & 2 \\ 1 & 50 & 42 \\ 1 & 45 & 31 \\ 1 & 60 & 35 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0.5 \end{bmatrix}.$$

Multiple outputs

If we have several outputs $\mathbf{y}_i \in \mathbb{R}^c$, our linear regression expression becomes:

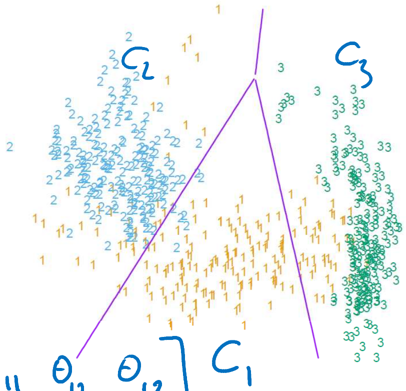
$$\underset{n\times c}{Y} = \underset{nd}{X}\ \underset{d\times c}{\Theta}$$



$$\begin{bmatrix} y_{11} & \cdots & y_{1c} \\ \vdots & \ddots & \\ y_{n1} & & y_{nc} \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ & & \\ x_{n1} & & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_{11} & & \theta_{1c} \\ & \ddots & \\ \theta_{d1} & & \theta_{dc} \end{bmatrix}$$

Linear classification with indicators

$$Y = \begin{array}{c} \begin{array}{ccc} C_1 & C_2 & C_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ & \vdots & \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$



$$\hat{Y}_c = X\Theta$$

$$\hat{Y}_c = \begin{bmatrix} 1.1 & 0.1 & 0.2 \\ & \vdots & \\ 0 & 0.1 & 0.9 \end{bmatrix} = \begin{bmatrix} \overset{1}{\cancel{x_{11}}} & \overbrace{\qquad\qquad}^{d\text{-}1 \text{ image pixels}} & \\ & \ddots & \\ x_{n1} & & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \vdots & & \\ \theta_{d1} & \theta_{d2} & \theta_{d3} \end{bmatrix}$$

$$\hat{Y}_i = \arg\max_c \hat{Y}_{ci}$$

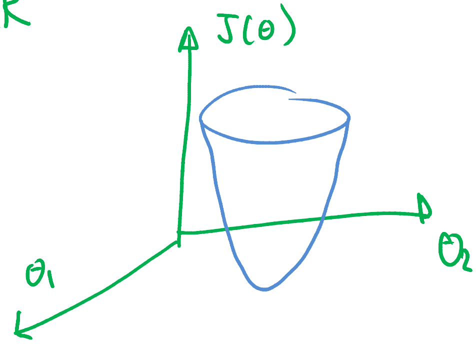
As discussed in Ch 4 of [Hastie, Tibshirani & Friedman book], this approach is easy, but not recommended. To describe better approaches, we will have to introduce notions of probability and optimization.

Optimization approach

Our aim is to mininimise the quadratic cost between the output labels and the model predictions

$$\boxed{J(\boldsymbol{\theta})} = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\theta})^2$$

$\Theta \in \mathbb{R}^2$



Optimization approach

We will need the following results from matrix differentiation: $\frac{\partial \mathbf{A}\theta}{\partial \boldsymbol{\theta}} = \mathbf{A}^T$ and $\frac{\partial \boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta}}{\partial \boldsymbol{\theta}} = 2\mathbf{A}^T\boldsymbol{\theta}$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \theta}(Y-X\theta)^T(Y-X\theta)$$

$$= \frac{\partial}{\partial \theta}\left(Y^TY + \theta^TX^TX\theta - 2Y^TX\theta\right)$$

$$= 0 + 2X^TX\theta - 2X^TY$$

$$X^TX\theta = X^TY$$

Optimization approach

$$\underset{d\times n}{X^T}\,\underset{n\times d}{X}\,\underset{d\times 1}{\theta} = \underset{d\times n}{X^T}\,\underset{n\times 1}{y}$$

These are the **normal equations**. The solution (estimate) is:

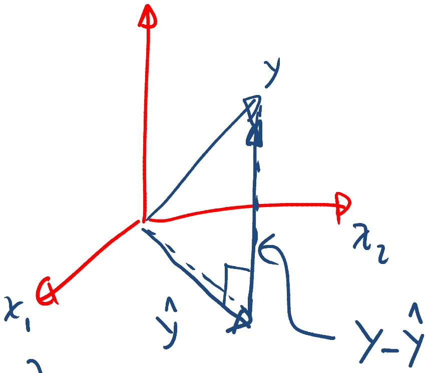
$$\widehat{\boldsymbol{\theta}} = (X^TX)^{-1}X^Ty$$

The corresponding predictions are

$$\widehat{y} = \mathbf{H}\mathbf{y} = x^*\hat{\theta} = x^*(X^TX)^{-1}X^Ty$$

where $\mathbf{H}$ is the “hat” matrix.

Geometric approach



$$X^T = \begin{bmatrix} x_1^T \\ x_2^T \end{bmatrix}$$

$$\mathbf{X}^T(\mathbf{y} - \widehat{\mathbf{y}}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \underline{0}$$

$$X^T y = X^T \hat{y}$$

$$X^T y = X^T X \theta \implies \boxed{\theta = (X^T X)^{-1} X^T y}$$

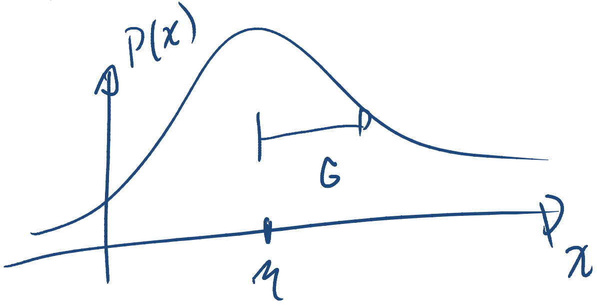
Probability approach: Univariate Gaussian distribution

The probability density function (pdf) of a Gaussian distribution is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}.$$

where μ is the mean or center of mass and σ^2 is the variance.

$x \sim \mathcal{N}(\mu, \sigma^2)$



Multivariate Gaussian distribution

Let $\mathbf{y} \in \mathbb{R}^{n \times 1}$, then pdf of an n-dimensional Gaussian is given by

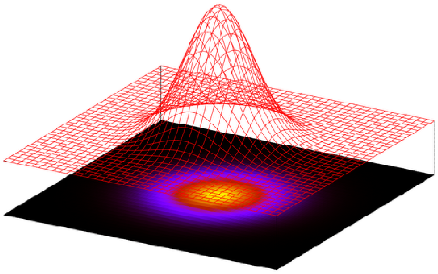
$$p(x) = |2\pi\boldsymbol{\Sigma}|^{-1/2} e^{-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})},$$

where

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} \mathbb{E}(y_1) \\ \vdots \\ \mathbb{E}(y_n) \end{pmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} \cdots \sigma_{1n} \\ \cdots \\ \sigma_{n1} \cdots \sigma_{nn} \end{pmatrix} = \mathbb{E}[(\mathbf{y}-\boldsymbol{\mu})(\mathbf{y}-\boldsymbol{\mu})^T]$$



Energy (“distance”) and probability

The exponent $\overset{d(y)}{\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})}$ is called the Mahalanobis distance. Conceptually, it measures the weighted Euclidean distance between $\mathbf{y}$ and $\boldsymbol{\mu}$.

$$P(\mathbf{y}) = |2\pi\Sigma|^{-1/2} e^{-\frac{1}{2}d(y)}$$

$$\frac{d}{2} = E$$

$$\boxed{P(\mathbf{y}) = Z e^{-E(\mathbf{y})}}$$

$$-\log p(y) = E(y) - \log Z$$

Maximum likelihood

We will assume that our errors in the target predictions are Gaussian distributed as follows:

$$\underset{n\times 1}{\mathbf{y}} = \underset{n\times d}{\mathbf{X}}\,\underset{d\times 1}{\boldsymbol{\theta}} + \mathcal{N}(0, \sigma^2 I_n) \qquad I_n = \begin{bmatrix} 1 & & & \\ & 1 & & 0 \\ 0 & & \ddots & \\ & & & 1 \end{bmatrix}$$

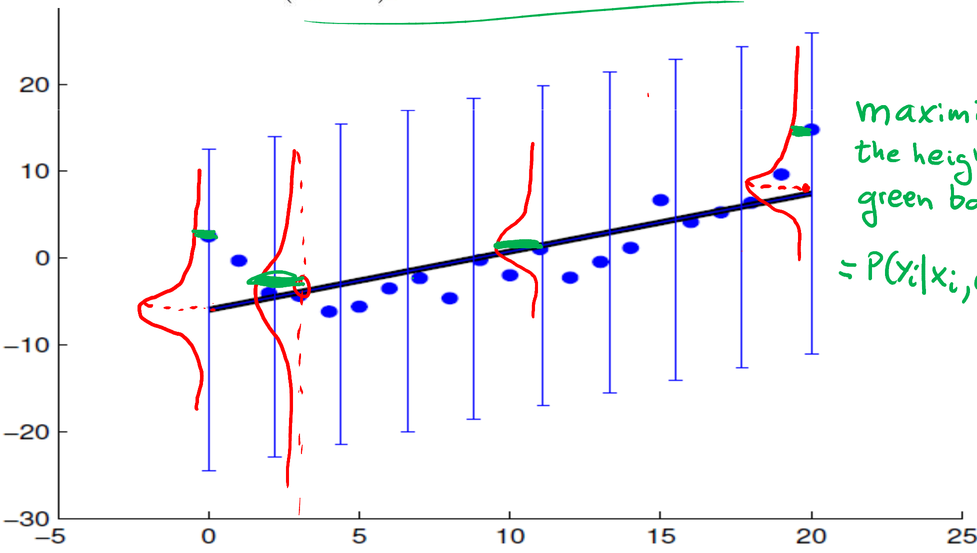
$$P(y_1, y_2) \overset{y_1 \text{ ind } y_2}{=} P(y_1)P(y_2)$$

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \sigma) &= \left(2\pi\sigma^2\right)^{-n/2} e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})} \\ &= \left(2\pi\sigma^2\right)^{-n/2} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - \mathbf{x}_i^T\boldsymbol{\theta})^2} \\ &= \prod_{i=1}^n \left(2\pi\sigma^2\right)^{-1/2} e^{-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T\boldsymbol{\theta})^2} \\ &= \prod_{i=1}^n p(y_i|\mathbf{x}_i, \boldsymbol{\theta}, \sigma). \end{aligned}$$

Maximum likelihood

$$J(\boldsymbol{\theta}) = \sum_{i=1}^n (y_i - \widehat{y}_i)^2 = \sum_{i=1}^n (y_i - \theta_1 - x_i\theta_2)^2$$

$$\begin{aligned} \underline{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \sigma)} &= \left(2\pi\sigma^2\right)^{-n/2} e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})} \\ &= \underline{\left(2\pi\sigma^2\right)^{-n/2} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - \mathbf{x}_i^T\boldsymbol{\theta})^2}} \end{aligned}$$



maximize the height of green bars

$= P(y_i|x_i, \theta, \sigma)$

Maximum likelihood

The maximum likelihood estimate (MLE) of $\boldsymbol{\theta}$ is obtained by taking the derivative of the log-likelihood, $\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\sigma)$. The goal is to maximize the likelihood of seeing the training data $\mathbf{y}$ by modifying the parameters $(\boldsymbol{\theta},\sigma)$.

$$P(y|X,\theta,\sigma^2) = (2\pi\sigma^2)^{-n/2} e^{-\frac{1}{2\sigma^2}(y-X\theta)^T(y-X\theta)}$$

$$+\log P(y|x,\theta,\sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y-x\theta)^T(y-x\theta)$$

Maximum likelihood

The ML estimate of $\boldsymbol{\theta}$ is:

$$\frac{\partial \log P(y|x,\theta,\sigma^2)}{\partial\theta} = 0 - \frac{1}{2\sigma^2}\frac{\partial}{\partial\theta}(y-x\theta)^T(y-x\theta)$$

$$= -\frac{1}{2\sigma^2}\left(2X^TX\theta - 2X^Ty\right) \to 0$$

$$X^TX\theta = X^Ty$$

$$\hat{\theta}_{ML} = (X^TX)^{-1}X^Ty$$

Maximum likelihood

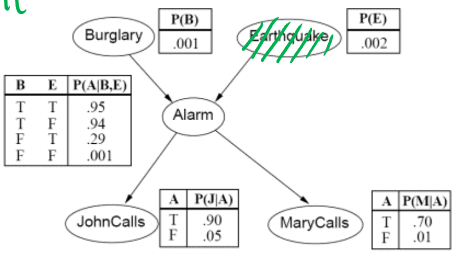
Proceeding in the same way, the ML estimate of σ is:

$$\frac{\partial}{\partial \sigma^2} \log P(y|x,\theta,\sigma^2) \longrightarrow 0$$

[e.]

Probabilistic graphical models (DAGs, Bayesian networks)

All r.v.s are binary



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

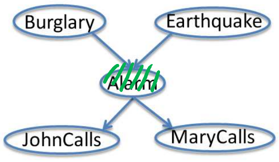
A	P(J|A)
T	.90
F	.05

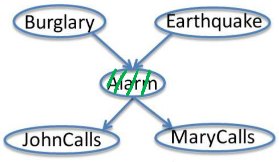
A	P(M|A)
T	.70
F	.01

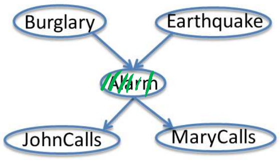
nodes ≡ r.v.s

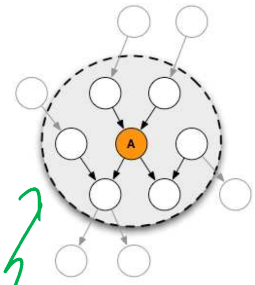
edges ≡ dependencies (statistical)

Shaded nodes ≡ observation









$M \perp\!\!\!\perp E \mid A$ ← given

↑ independent

$J \perp\!\!\!\perp M \mid A$

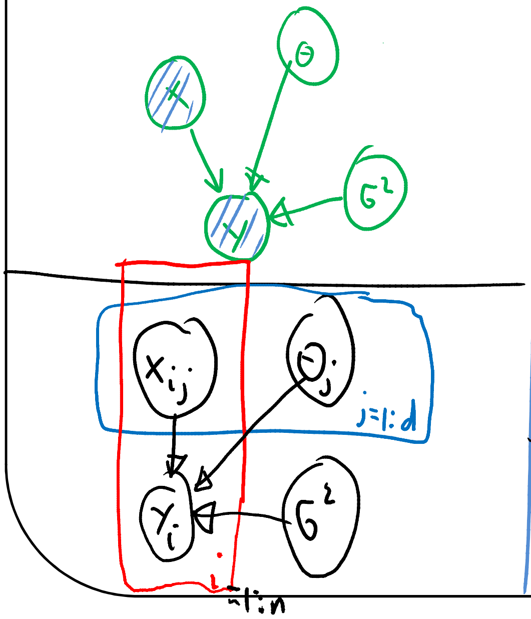
$B \not\perp\!\!\!\perp E \mid A$

Markov Blanket of A

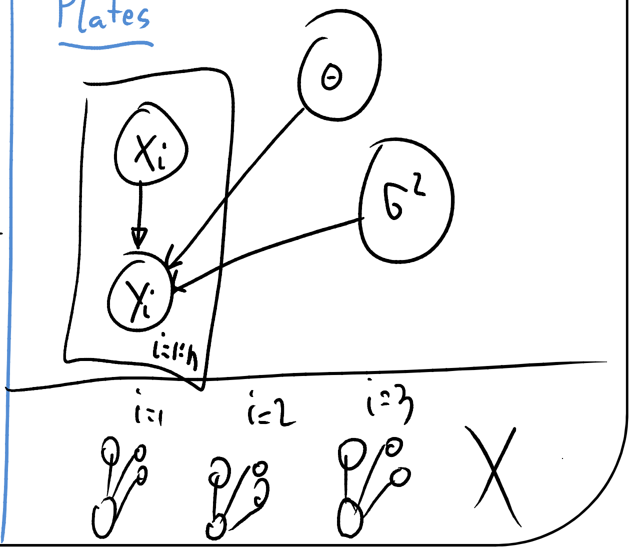
Linear regression DAG

Vars = $\{X, Y, \theta, \sigma^2\}$

$P(Y|X,\theta,\sigma^2)$



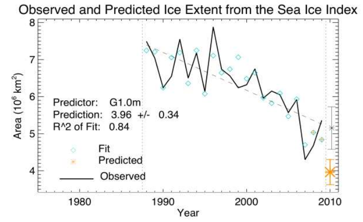
Plates

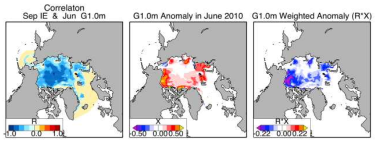


Making predictions

$P(y|X,\theta,\sigma^2)$

Predictions for September 2010 from June





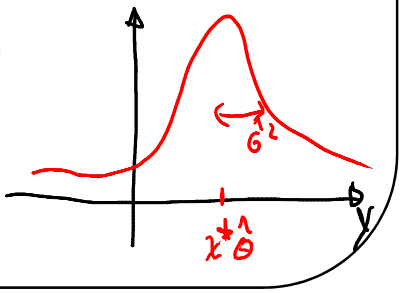
Given new x^*

Use estimated $\hat{\theta}$ and $\hat{\sigma}_2$

Prediction:

$P(y|x^*, \hat{\theta}, \hat{\sigma}_2)$

$\mathbb{E}(y|x^*, \hat{\theta}, \hat{\sigma}_2) = x^* \hat{\theta}$



Why the MLE: Hallucination!

The MLE assumes that the data has been generated by a distribution $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}_0)$ for some *true parameter* $\boldsymbol{\theta}_0$.

Flickr: jsarcadia

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) - \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}_0) \\
&= \arg\max_{\boldsymbol{\theta}} \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \log \dfrac{p(y_i|\mathbf{x}_i, \boldsymbol{\theta})}{p(y_i|\mathbf{x}_i, \boldsymbol{\theta}_0)} \\
\longrightarrow & \arg\min_{\boldsymbol{\theta}} \int \log \frac{p(y_{\cancel{i}}|\mathbf{x}_{\cancel{i}}, \boldsymbol{\theta}_0)}{p(y_{\cancel{i}}|\mathbf{x}_{\cancel{i}}, \boldsymbol{\theta})} p(y|\mathbf{x}, \boldsymbol{\theta}_0)dy \\
&= \arg\min_{\boldsymbol{\theta}} \int \log p(y|\mathbf{x}, \boldsymbol{\theta}_0)p(y|\mathbf{x}, \boldsymbol{\theta}_0)dy - \int \log p(y|\mathbf{x}, \boldsymbol{\theta})p(y|\mathbf{x}, \boldsymbol{\theta}_0)dy
\end{aligned}$$

$\hat{\theta}$

KL divergence

Why the MLE (frequentist view)

Under smoothness and identifiability assumptions,
the MLE is *consistent*:

$$\hat{\boldsymbol{\theta}} \xrightarrow{p} \boldsymbol{\theta}_0$$

or equivalently,

$$\lim_{N\longrightarrow\infty} P(|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0| > \alpha) \rightarrow 0$$

The MLE is *asymptotically normal*. That is, as $N \rightarrow \infty$, we have:

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0 \Longrightarrow N(0, I^{-1})$$

where I is the *Fisher Information matrix.*

It is asymptotically optimal or *efficient*.

Regularization

All the answers so far are of the form

$$\widehat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

They require the inversion of $\mathbf{X}^T\mathbf{X}$. This can lead to problems if the system of equations is poorly conditioned. A solution is to add a small element to the diagonal:

$$\widehat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X} + \delta^2 I_d)^{-1}\mathbf{X}^T\mathbf{y}$$

This is the ridge regression estimate. It is the solution to the following **regularised quadratic cost function**

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \underline{\delta^2\boldsymbol{\theta}^T\boldsymbol{\theta}}$$

Proof

$$J(\theta) = (y - X\theta)^T(y - X\theta) + \delta^2\,\theta^T\theta$$

$$= y^Ty - 2y^TX\theta + \theta^TX^TX\theta + \delta^2\,\theta^T\theta$$

$$\frac{\partial J(\theta)}{\partial\theta} = 0 - 2X^Ty + 2X^TX\theta + 2\delta^2 I\theta$$

$$= -2X^Ty + 2(X^TX + \delta^2 I)\theta$$

equate to zero

$$\hat{\theta}_{ridge} = (X^TX + \delta^2 I)^{-1}X^Ty$$

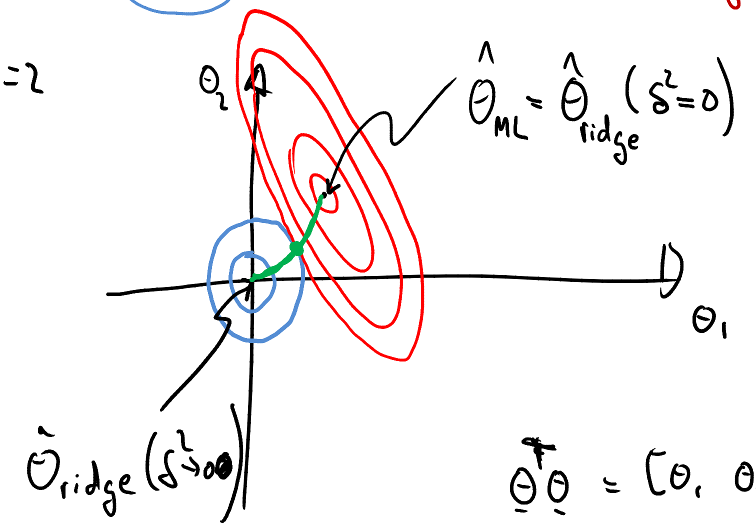
Ridge regression as constrained optimization

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta}$$ Lagrangian

$$\min_{\boldsymbol{\theta}\,:\,\boldsymbol{\theta}^T\boldsymbol{\theta} \le t(\delta)} \{\overset{\|\|}{\boxed{(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})}}\}$$

δ large $\Rightarrow t$ small

$d=2$

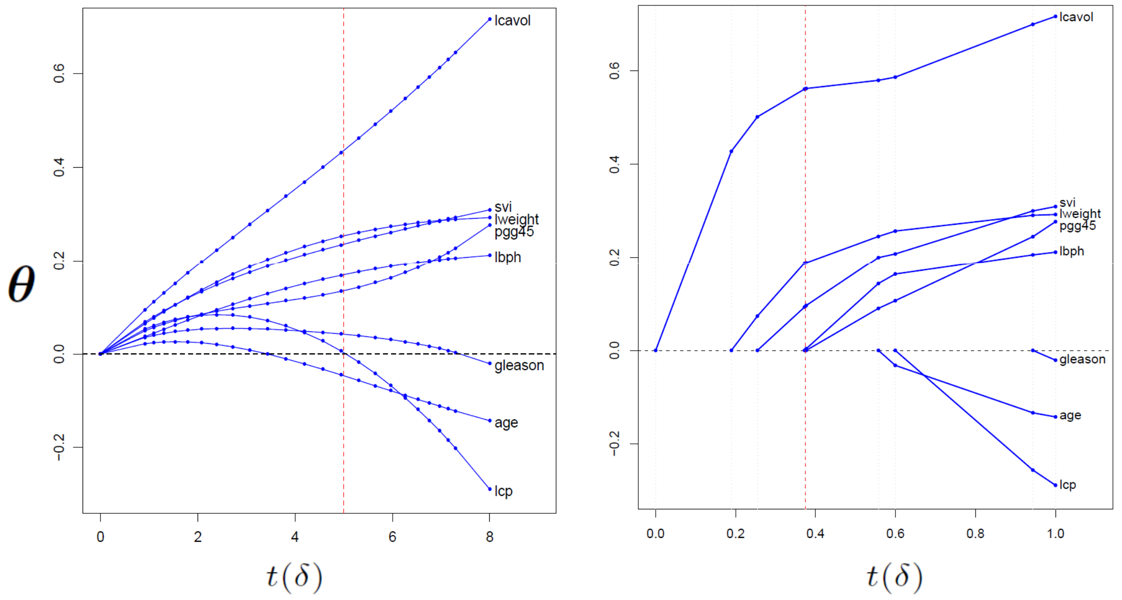


θ_2
θ_1
$\hat{\theta}_{ML} = \hat{\theta}_{ridge}(\delta^2=0)$
$\hat{\theta}_{ridge}(\delta^2 \to \infty)$

$$\underline{\theta}^T\underline{\theta} = [\theta_1 \;\; \theta_2]\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = t$$

$$\uparrow \qquad = \theta_1^2 + \theta_2^2$$

Ridge as constrained optimization



[Hastie, Tibshirani & Friedman book]

Ridge, feature selection, shrinkage and weight decay

Large values of $\boldsymbol{\theta}$ are penalised. We are *shrinking* $\boldsymbol{\theta}$ towards zero. This can be used to carry out *feature weighting*. An input $x_{i,d}$ weighted by a small θ_d will have less influence on the ouptut y_i. This penalization with a regularizer is also known as weight decay in the neural networks literature.

Note that shrinking the bias term $\boldsymbol{\theta}_1$ is undesirable. To keep the notation simple, we will assume that the mean of $\mathbf{y}$ has been subtracted from $\mathbf{y}$. This mean is indeed our estimate $\widehat{\boldsymbol{\theta}_1}$.

The Lasso: least absolute selection and shrinkage operator

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2 \sum_{j=1}^{d} |\theta_j|$$

✓ L_1 norm

$$\min_{\boldsymbol{\theta}\,:\,\sum_{j=1}^{d}|\theta_j| \le t(\delta)} \{(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\}$$

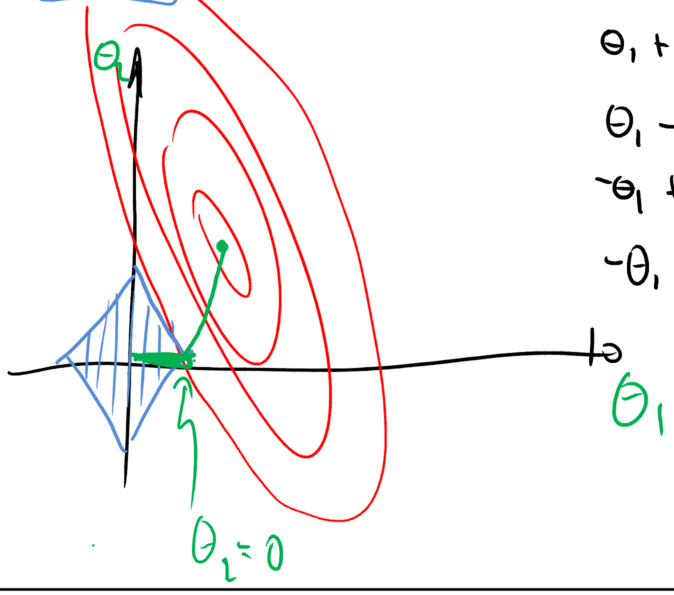
$$|\theta_1| + |\theta_2| = t$$

$\theta_1 + \theta_2 \le t$

$\theta_1 - \theta_2 \le t$

$-\theta_1 + \theta_2 \le t$

$-\theta_1 - \theta_2 \le t$



Spectral view of ridge regression

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{d} \mathbf{u}_i \sigma_i \mathbf{v}_i^T$$

$n \times d$

$\hat{y} = UU^T y$

$\hat{y} = \sum_i u_i u_i^T y$

$\hookrightarrow X^T X \hat{\theta} = X^T y$

$\hat{\theta} = (X^T X)^{-1} X^T y$

$\hat{y} = X\hat{\theta}$

$(V\Sigma U^T)(U\Sigma V^T)\hat{\theta} = V\Sigma U^T y$

$V^T V \Sigma^2 V^T \hat{\theta} = V^T V \Sigma U^T y$

$\Sigma V^T \hat{\theta} = U^T y$

$\hat{\theta} = V\Sigma^{-1}U^T y$

$\hat{y} = U\Sigma V^T V \Sigma^{-1} U^T y$

$\hat{y} = UU^T y$

$\hat{y}_k \approx U_k U_k^T y$

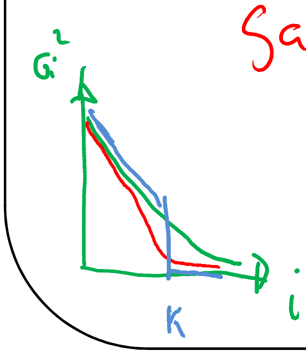
$$\widehat{\mathbf{y}}_{ridge} = \sum_{i=1}^{d} \left(\frac{\sigma_i^2}{\sigma_i^2 + \delta^2}\right) \mathbf{u}_i \mathbf{u}_i^T \mathbf{y} \quad \leftarrow f_i$$

$\hat{y}_{ML=LS} = \sum_{i=1}^{d \to k} u_i u_i^T y \qquad k<d$

$(X^T X + \delta^2 I)\theta = X^T y \qquad [e.]$

Say $\delta^2 = 10$, $\sigma_i^2 = 0.1 \qquad f_i \approx 0.01$

$\delta^2 = 10$, $\sigma_i^2 = 100 \qquad f_i \approx 1$



Regularization and noise filtering

The filter factor

$$f_i = \frac{\sigma_i^2}{\sigma_i^2 + \delta^2}$$

penalises small values of σ^2 (they go to zero at a faster rate).

Minimax and cross-validation

Cross-validation is a widely used technique for choosing δ. Here's an example:

demo	δ^2	Train error $\sum_{i \in train}(y_i-\hat{y}_i)^2$	Test error $\sum_{i \in test}(y_i-\hat{y}_i)^2$	max worst	worst case e_1 min-max	e_2 avg	
0	0.1	100	2	100			
9	1	10	11	11	x		
7	10	1	19	19		x	10
8	50	20	0	20		x	10
16	100	100	1000	1000			

Least Angle Regression

Algorithm 3.2 *Least Angle Regression.*

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.
5. Continue in this way until all p predictors have been entered. After $\min(N-1, p)$ steps, we arrive at the full least-squares solution.

$$\underset{n\times 1}{\hat{Y}} = \underset{n\times p}{X}\ \underset{p\times 1}{\beta}$$

$$\beta_{\mathcal{A}_k}(\alpha) = \beta_{\mathcal{A}_k} + \alpha \cdot \delta_k$$

$$\delta_k = (\mathbf{X}^T_{\mathcal{A}_k}\mathbf{X}_{\mathcal{A}_k})^{-1}\mathbf{X}^T_{\mathcal{A}_k}\mathbf{r}_k$$

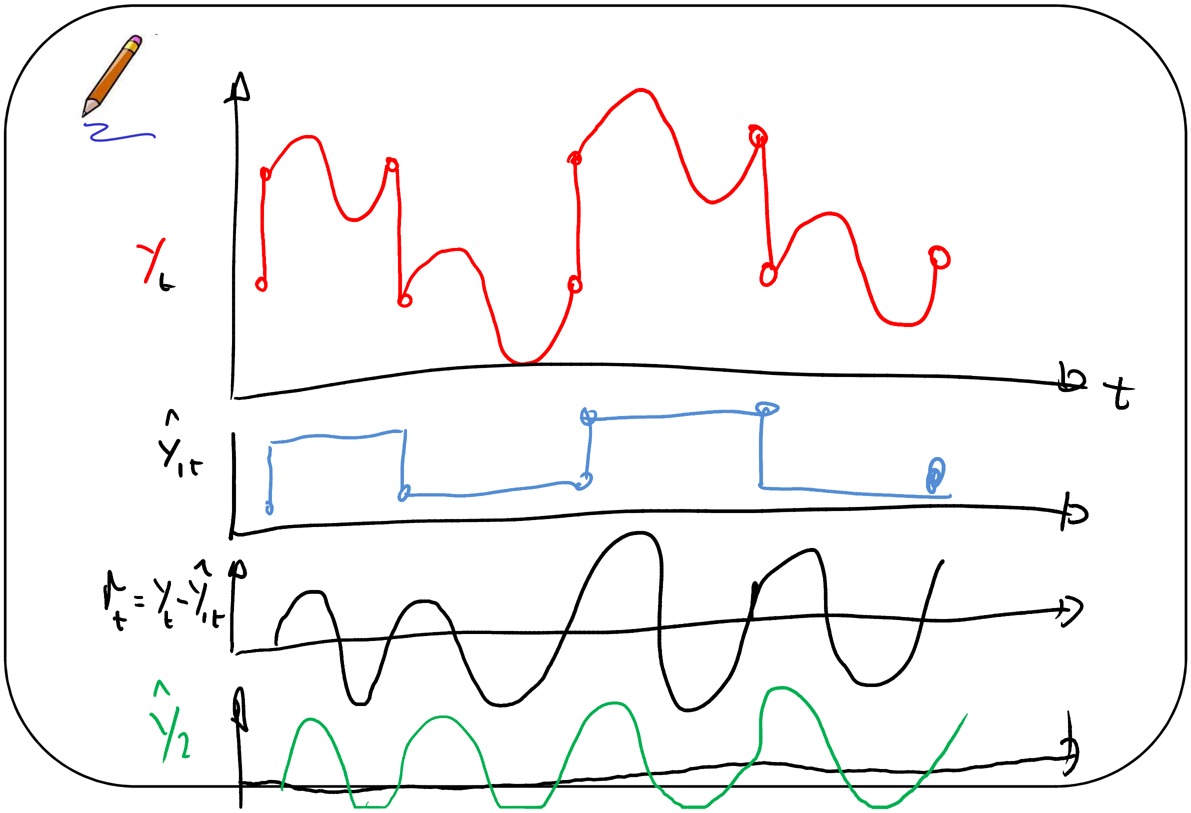
$$X = [x_1 \cdots x_j \cdots x_p]$$

Suppose $\mathcal{A}_k$ is the active set of variables

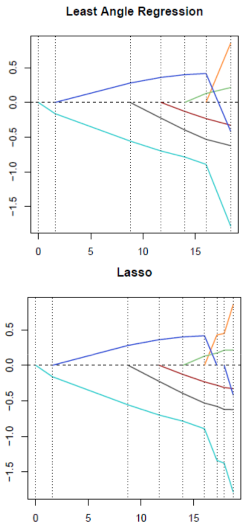
$\mathbf{r}_k = \mathbf{y} - \mathbf{X}_{\mathcal{A}_k}\beta_{\mathcal{A}_k}$ is the current residual

[Hastie, Tibshirani & Friedman book]

Fitting the residuals



The Lasso & LAR



Algorithm 3.2 *Least Angle Regression.*

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \dots, \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.
5. Continue in this way until all p predictors have been entered. After $\min(N-1, p)$ steps, we arrive at the full least-squares solution.

Algorithm 3.2a *Least Angle Regression: Lasso Modification.*

4a. If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.

[Hastie, Tibshirani & Friedman book]

The Lasso as LAR intuition

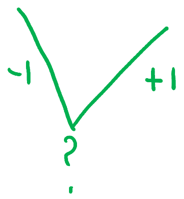
When the active attributes are tied, in the sense that the absolute value of their inner-product with the residuals is the same (having standardized the inputs so that instead of using correlations, we are now using inner-products), we have:

$$\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \gamma \mathrm{sign}(\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})), \quad \forall j \in \mathcal{A}$$

where $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_d]$, with $\mathbf{x}_j \in \mathbb{R}^{n \times 1}$, γ is the common absolute value and the sign function $\mathrm{sign}(\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}))$ is in $\{-1, 1\}$. The set $\mathcal{A}$ is the active set.

"Differentiating" the Lasso's Lagrangian

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2 \sum_{j=1}^{d} |\theta_j|$$



with respect to θ_j and equating to zero, we get

$$-\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2 \mathrm{sign}(\theta_j) = 0$$

$$\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \delta^2 \mathrm{sign}(\theta_j), \quad \forall j \in \mathcal{B}$$

[Hastie, Tibshirani & Friedman book]

Lasso using optimization

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = a_j \theta_j - c_j + \delta^2 \frac{\partial}{\partial \theta_j} |\theta_j|$$

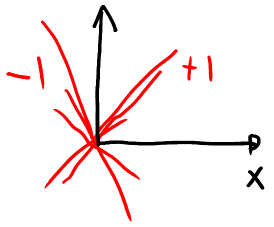
$$a_j = 2 \sum_{i=1}^{n} x_{ij}^2$$

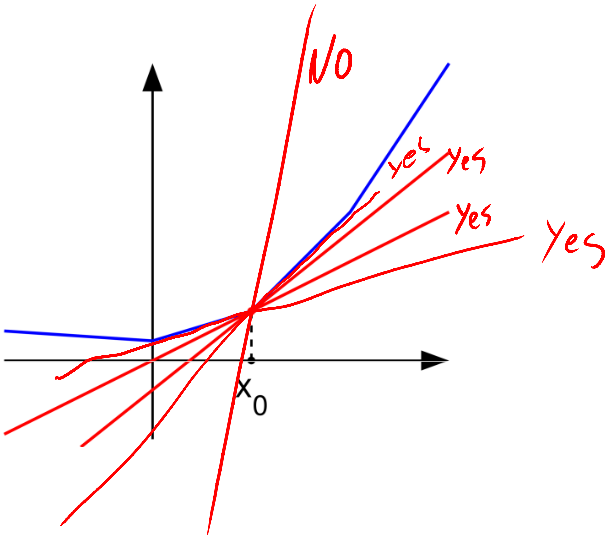
$$c_j = 2 \sum_{i=1}^{n} x_{ij} (y_i - \boldsymbol{\theta}_{-j}^T \mathbf{x}_{i,-j})$$

where $\boldsymbol{\theta}_{-j} = \boldsymbol{\theta}$ without component j and likewise for $\mathbf{x}_{i,-j}$. We see that c_j is (proportional to) the correlation between the j'th feature $\mathbf{x}_j$ and the residual due to the other features, $\mathbf{r}_{-j} = \mathbf{y} - \mathbf{X}_{-j}\boldsymbol{\theta}_{-j}$.

Subdifferentials

$$f(x) = |x|$$



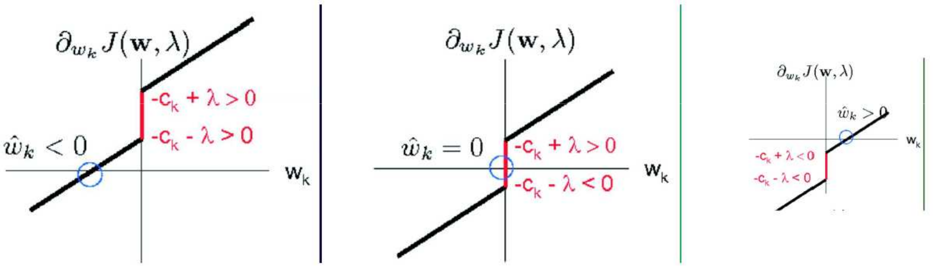


$$\partial f(x) = \begin{cases} \{-1\} & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \\ \{+1\} & \text{if } x > 0 \end{cases}$$

[Wikipedia]

Lasso using optimization

$$\partial_{\theta_j} J(\boldsymbol{\theta}) = a_j\theta_j - c_j + \delta^2 \partial_{\theta_j}|\theta_j| = \begin{cases} \{a_j\theta_j - c_j - \delta^2\} & \text{if } \theta_j < 0 \\ [-c_j - \delta^2, -c_j + \delta^2] & \text{if } \theta_j = 0 \\ \{a_j\theta_j - c_j + \delta^2\} & \text{if } \theta_j > 0 \end{cases}$$



$W = \Theta$

$$\widehat{\theta}_j = \begin{cases} (c_j + \delta^2)/a_j & \text{if } c_j < -\delta^2 = -\lambda \\ 0 & \text{if } c_j \in [-\delta^2, \delta^2] = [-\lambda, \lambda) \\ (c_j - \delta^2)/a_j & \text{if } c_j > \delta^2 = \lambda \end{cases}$$

[Kevin Murphy's book]

Coordinate descent for lasso

Algorithm 11.1: Coordinate descent for lasso (aka shooting algorithm)

```
Initialize $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$;
repeat
    for $j = 1, \ldots, D$ do
        $a_j = 2\sum_{i=1}^n x_{ij}^2$;
        $c_j = 2\sum_{i=1}^n x_{ij}(y_i - \mathbf{w}^T\mathbf{x}_i + w_j x_{ij})$ ;
        if $c_j < -\lambda$ then
            $w_j = \frac{c_j+\lambda}{a_j}$
        else if $c_j > \lambda$ then
            $w_j = \frac{c_j-\lambda}{a_j}$
        else
            $w_j = 0$
until converged;
```

$W = \Theta$

$\lambda = \delta^2$

Predictor $D = p$

$X \in \mathbb{R}^{n \times p}$

[Kevin Murphy's book]