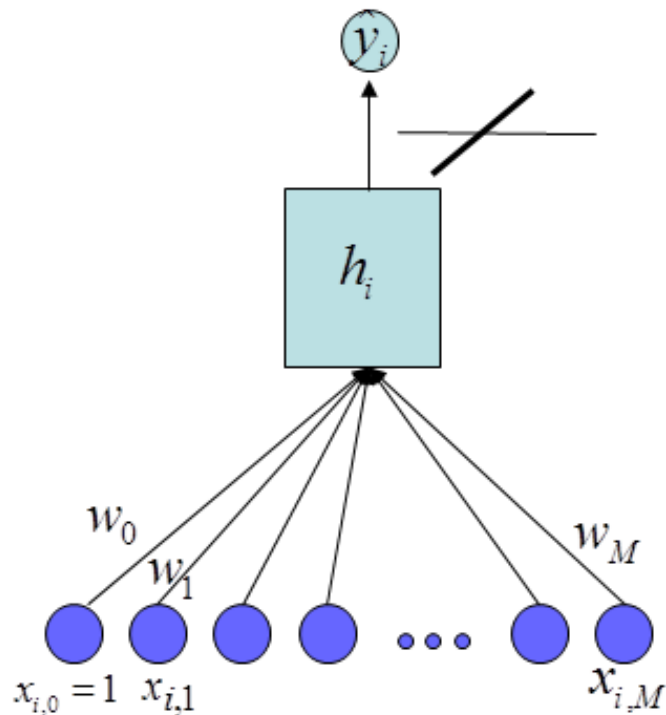


Linear Regression

Volker Tresp
2019

Learning Machine: The Linear Model / ADALINE



- As with the Perceptron we start with an activation functions that is a linearly weighted sum of the inputs

$$h = \sum_{j=0}^M w_j x_j$$

(Note: $x_0 = 1$ is a constant input, so that w_0 is the bias)

- New: **The activation is the output** (no thresholding)

$$\hat{y} = f_{\mathbf{w}}(\mathbf{x}) = h$$

- Regression: the target function can take on real values

Method of Least Squares

- Squared-loss cost function:

$$\text{cost}(\mathbf{w}) = \sum_{i=1}^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$$

- The parameters that minimize the cost function are called least squares (LS) estimators

$$\mathbf{w}_{ls} = \arg \min_{\mathbf{w}} \text{cost}(\mathbf{w})$$

- For visualization, we take $M = 1$ (although linear regression is often applied to high-dimensional inputs)

Least-squares Estimator for Regression

One-dimensional regression:

$$f_{\mathbf{w}}(x) = w_0 + w_1 x$$

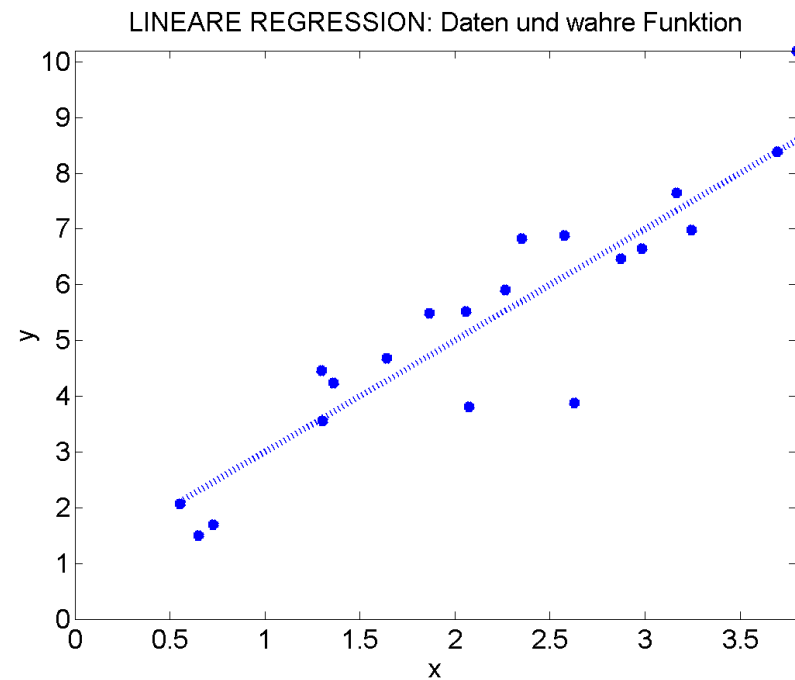
$$\mathbf{w} = (w_0, w_1)^T$$

Squared error:

$$\text{cost}(\mathbf{w}) = \sum_{i=1}^N (y_i - f_{\mathbf{w}}(x_i))^2$$

Goal:

$$\mathbf{w}_{ls} = \arg \min_{\mathbf{w}} \text{cost}(\mathbf{w})$$



$$w_0 = 1, w_1 = 2, \text{var}(\epsilon) = 1$$

Least-squares Estimator in Several Dimensions

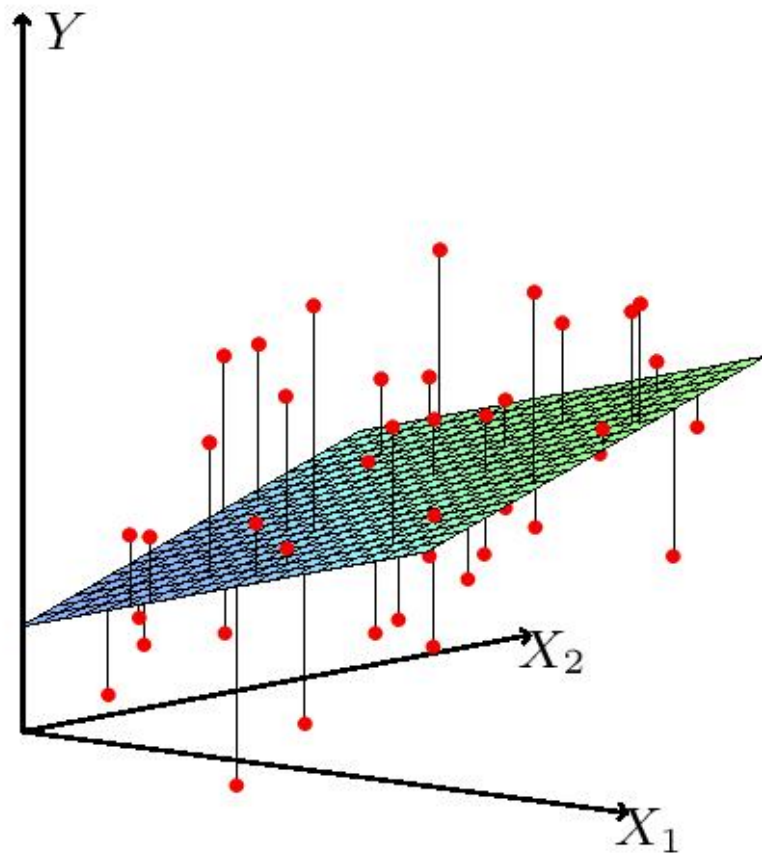
General Model:

$$\begin{aligned}\hat{y}_i = f_{\mathbf{w}}(\mathbf{x}_i) &= w_0 + \sum_{j=1}^M w_j x_{i,j} \\ &= \mathbf{x}_i^T \mathbf{w}\end{aligned}$$

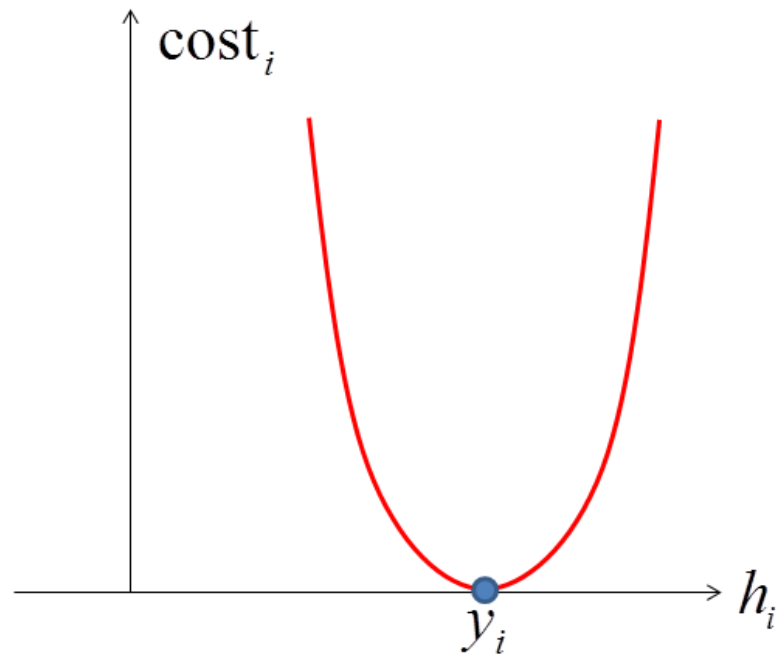
$$\mathbf{w} = (w_0, w_1, \dots, w_M)^T$$

$$\mathbf{x}_i = (1, x_{i,1}, \dots, x_{i,M})^T$$

Linear Regression with Several Inputs



Contribution to the Cost Function of one Data Point



Predictions as Matrix-vector product

The vector of all predictions at the training data is

$$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_N \end{pmatrix} = \mathbf{X}\mathbf{w}$$

Gradient Descent Learning

- Initialize parameters (typically using small random numbers)
- Adapt the parameters in the direction of the negative gradient
- With $f_{\mathbf{w}}(\mathbf{x}_i) = \sum_{j=0}^M w_j x_{i,j}$

$$\text{cost}(\mathbf{w}) = \sum_{i=1}^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$$

- The parameter gradient is (Example: w_j)

$$\frac{\partial \text{cost}}{\partial w_j} = -2 \sum_{i=1}^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i)) x_{i,j}$$

- A sensible learning rule is

$$w_j \leftarrow w_j + \eta \sum_{i=1}^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i)) x_{i,j}$$

ADALINE-Learning Rule

- ADALINE: ADaptive LINear Element
- The ADALINE uses stochastic gradient descent (SGD)
- Let \mathbf{x}_t and y_t be the training pattern in iteration t . Then we adapt, $t = 1, 2, \dots$

$$w_j \leftarrow w_j + \eta(y_t - \hat{y}_t)x_{t,j} \quad j = 0, 1, 2, \dots, M$$

- $\eta > 0$ is the learning rate, typically $0 < \eta \ll 0.1$
- This is identical to the Perceptron learning rule. For the Perceptron $y_t \in \{-1, 1\}$, $\hat{y}_t \in \{-1, 1\}$

Analytic Solution

- The ADALINE is optimized by SGD
- Online Adaptation: a physical system constantly produces new data: the ADALINE (SGD in general) can even track changes in the system
- With a fixed training data set the least-squares solution can be calculated analytically in one step (least-squares regression)

Cost Function in Matrix Form

$$\text{cost}(\mathbf{w}) = \sum_{i=1}^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$$

$$= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\mathbf{y} = (y_1, \dots, y_N)^T$$

$$\mathbf{X} = \begin{pmatrix} x_{1,0} & \dots & x_{1,M} \\ \dots & \dots & \dots \\ x_{N,0} & \dots & x_{N,M} \end{pmatrix}$$

Necessary Condition for an Optimum

- A necessary condition for an optimum is that

$$\frac{\partial \text{cost}(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}=\mathbf{w}_{opt}} = 0$$

One Parameter: Explicit

- $f_w(x_1) = x_1 w_1$ and $\text{cost}(w_1) = \sum_{i=1}^N (y_i - x_{i,1} w_1)^2$
- (chain rule: inner derivative times outer derivative)

$$\begin{aligned} \frac{\partial \text{cost}(w_1)}{\partial w_1} &= \sum_{i=1}^N \frac{\partial (y_i - x_{i,1} w_1)}{\partial w_1} 2(y_i - x_{i,1} w_1) \\ &= -2 \sum_{i=1}^N x_{i,1} (y_i - x_{i,1} w_1) = -2 \sum_{i=1}^N x_{i,1} y_i + 2w_1 \sum_{i=1}^N x_{i,1} x_{i,1} \end{aligned}$$

- Thus

$$w_{1,ls} = \left(\sum_{i=1}^N x_{i,1} x_{i,1} \right)^{-1} \sum_{i=1}^N x_{i,1} y_i$$

One Parameter: in Vector Notation

- $f_w(x_1) = x_1 w_1$ and $\text{cost}(w_1) = (\mathbf{y} - \bar{\mathbf{x}}_1 w_1)^T (\mathbf{y} - \bar{\mathbf{x}}_1 w_1)$, where $\bar{\mathbf{x}}_1 = (x_{1,1}, \dots, x_{N,1})^T$
- (chain rule: inner derivative times outer derivative)

$$\begin{aligned} \frac{\partial \text{cost}(w_1)}{\partial w_1} &= \frac{\partial (\mathbf{y} - \bar{\mathbf{x}}_1 w_1)}{\partial w_1} 2(\mathbf{y} - \bar{\mathbf{x}}_1 w_1) \\ &= -2\bar{\mathbf{x}}_1^T (\mathbf{y} - \bar{\mathbf{x}}_1 w_1) = -2\bar{\mathbf{x}}_1^T \mathbf{y} + 2w_1 \bar{\mathbf{x}}_1^T \bar{\mathbf{x}}_1 \end{aligned}$$

- Thus

$$w_{1,ls} = \left(\bar{\mathbf{x}}_1^T \bar{\mathbf{x}}_1 \right)^{-1} \bar{\mathbf{x}}_1^T \mathbf{y}$$

General Case

- $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ and $\text{cost}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$
- (chain rule: inner derivative times outer derivative)

$$\begin{aligned}\frac{\partial \text{cost}(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial (\mathbf{y} - \mathbf{X}\mathbf{w})}{\partial \mathbf{w}} 2(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = -2\mathbf{X}^T \mathbf{y} + 2\mathbf{w}\mathbf{X}^T \mathbf{X}\end{aligned}$$

- Thus

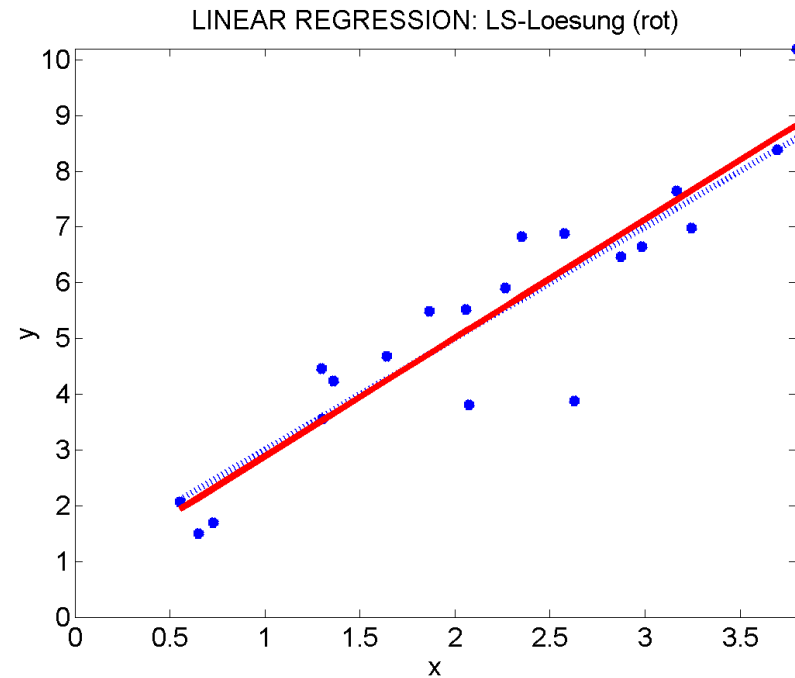
$$\mathbf{w}_{ls} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

Setting First Derivative to Zero

$$\hat{\mathbf{w}}_{ls} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Complexity (linear in N):

$$\mathcal{O}(M^3 + NM^2)$$



$$\hat{w}_0 = 0.75, \hat{w}_1 = 2.13$$

Derivatives of Vector Products

- We have used

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{A} \mathbf{x} = \mathbf{A}^T \quad \frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{x} = 2 \mathbf{x} \quad \frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{x} = (\mathbf{A} + \mathbf{A}^T) \mathbf{x}$$

- Comment: one also finds the conventions,

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{A} \mathbf{x} = \mathbf{A} \quad \frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{x} = 2 \mathbf{x}^T \quad \frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{x}^T (\mathbf{A} + \mathbf{A}^T)$$

Stability of the Solution

- When $N \gg M$, the LS solution is stable (small changes in the data lead to small changes in the parameter estimates)
- When $N < M$ then there are many solutions which all produce zero training error
- Of all these solutions, one selects the one that minimizes $\sum_{j=0}^M w_j^2 = \mathbf{w}^T \mathbf{w}$ (regularised solution)
- Even with $N > M$ it is advantageous to regularize the solution, in particular with noise on the target

Linear Regression and Regularisation

- Regularised cost function (*Penalized Least Squares* (PLS), *Ridge Regression*, *Weight Decay*): the influence of a single data point should be small

$$\text{cost}^{pen}(\mathbf{w}) = \sum_{i=1}^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2 + \lambda \sum_{j=0}^M w_j^2$$

$$\hat{\mathbf{w}}_{pen} = \left(\mathbf{X}^T \mathbf{X} + \lambda I \right)^{-1} \mathbf{X}^T \mathbf{y}$$

Derivation:

$$\frac{\partial \text{cost}^{pen}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda\mathbf{w} = 2[-\mathbf{X}^T \mathbf{y} + (\mathbf{X}^T \mathbf{X} + \lambda I)\mathbf{w}]$$

ADALINE-Learning Rule with Weight Decay

- Let \mathbf{x}_t and y_t be the training pattern in iteration t . Then we adapt, $t = 1, 2, \dots$

$$w_j \longleftarrow w_j + \eta[(y_t - \hat{y}_t)x_{t,j} - \frac{\lambda}{N}w_j] \quad j = 0, 1, 2, \dots, M$$

Toy Example: Unidimensional Model (Pearson Correlation Coefficient)

- We generated $N = 100$ data points with $M = 3$ inputs (no bias)
- x_1 and x_2 are highly correlated. x_3 is independent from x_1 , x_2 , and y
- We generate output data with $y = x_1 + \epsilon$, where ϵ stands for independent noise with standard deviation 0.2 and thus variance of 0.04. Thus ground truth parameters are $\mathbf{w}_{true} = (1, 0, 0)^T$. Note that, y causally only depends on x_1
- All variables are normalized to 0 mean and variance 1.
- In unit dimensional models, with only one input, the weights are identical to the sample Pearson correlation coefficients (here: $r_j = \sum_i y_i x_{i,j} / N$) between the output and the input, I get $r_1 = 0.99$, $r_2 = 0.96$, $r_3 = -0.21$.

Toy Example: Unidimensional Model (Pearson Correlation Coefficient) (cont'd)

- Explicitly the three models are

$$\hat{y} = 0.99x_1$$

$$\hat{y} = 0.96x_2$$

$$\hat{y} = -0.21x_3$$

- A deeper analysis reveals that the estimate r_1 has a mean of 1 and a standard deviation of 0.02. r_1 reflects the dependency of y on x_1
- The second coefficient, $r_2 = 0.96$, does not reflect a causal effect, but reflects the fact that x_1 and x_2 are highly correlated, and thus also y and x_2 (correlation does not imply causality). A deeper analysis reveals that with perfect correlation between x_1 and x_2 , the estimate r_1 also would have a mean of 1 and a standard deviation of 0.02

- The third value r_3 is correctly closer to 0, but not really small in magnitude. A deeper analysis reveals that the estimate r_3 has a mean of 0 and a standard deviation of approximately of 0.1

Toy Example: Least Squares Regression

- We get:

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} 100 & 98 & -18 \\ 98 & 100 & -16 \\ -18 & -16 & 100 \end{bmatrix}$$

Approximately $N\text{COV}(\mathbf{x})$; we see the strong correlation between x_1 and x_2

$$(\mathbf{X}^T \mathbf{X})^{-1} = \begin{bmatrix} 0.255 & -0.249 & 0.007 \\ -0.249 & 0.253 & -0.005 \\ 0.007 & -0.005 & 0.010 \end{bmatrix}$$

Finally,

$$\mathbf{X}^T \mathbf{y} = (99, 97, -20)^T$$

This is approximately $N\mathbf{r}$; we see the strong correlation between both x_1 and x_2 with y

Toy Example: Least Squares Regression (cont'd)

- We get

$$\mathbf{w}_{ls} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = (1.137, -0.150, -0.018)^T$$

- Interestingly, linear regression pretty much identifies the correct causality, with $w_{ls,1} \approx 1$ and $w_{ls,2} \approx 0$!
- A deeper analysis reveals that $w_{ls,1}$ has a mean of 1 and a standard deviation of 0.1. So the estimator is unbiased but the uncertainty is larger than in the unit dimensional analysis
- $w_{ls,2}$ has **mean of zero** and a standard deviation of 0.1. Thus the bias is removed if compared to Pearson!
- $w_{ls,1}$ and $w_{ls,2}$ are negatively correlated. Note, that $w_{ls,1} + w_{ls,2} = 0.987$ which is close to the true 1.

Toy Example: Least Squares Regression (cont'd)

- $w_{ls,3} = -0.018$ is much closer to 0 than the sample Pearson correlation coefficient $r_3 = -0.21$
- A deeper analysis (see Appendix) reveals that $w_{ls,3}$ has a mean of 0 and a standard deviation of 0.02. Here it is important to see that the standard deviation of the spurious input is largely reduced!
- Overall, in regression, the causal influence of x_1 stands out much more clearly! Both the influence of the correlated input x_2 and the noise input x_3 are largely reduced
- Application in healthcare: Same data. Consider that x_2 is a medication and y the outcome. If I do a unidimensional analysis, I would see a strong positive influence of x_2 on y (the medication works). Only if I include the so-called confounder x_1 in the regression model, it becomes clear that the confounder x_1 is the cause and not the treatment x_2 . The treatment has no significant effect!

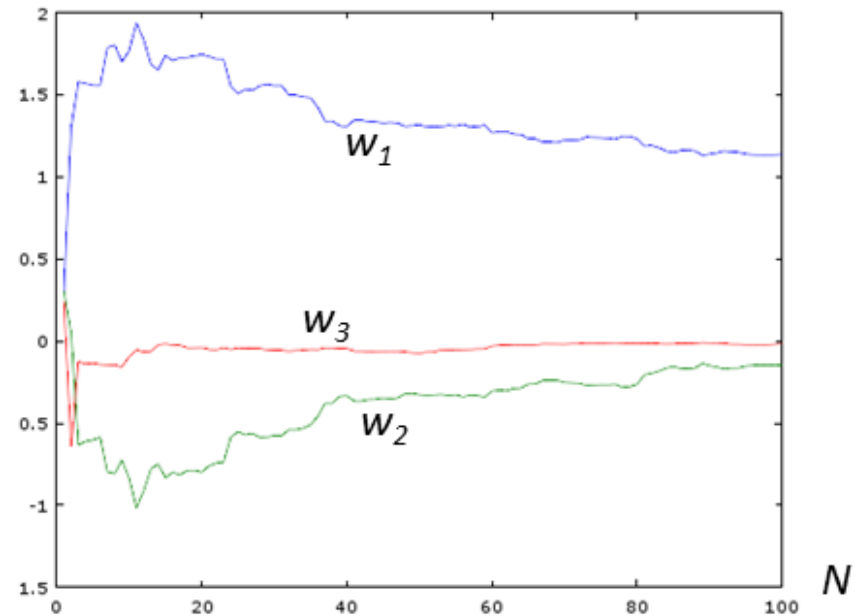
Least Squares Regression:

With small N , weights are unstable and test set error is large

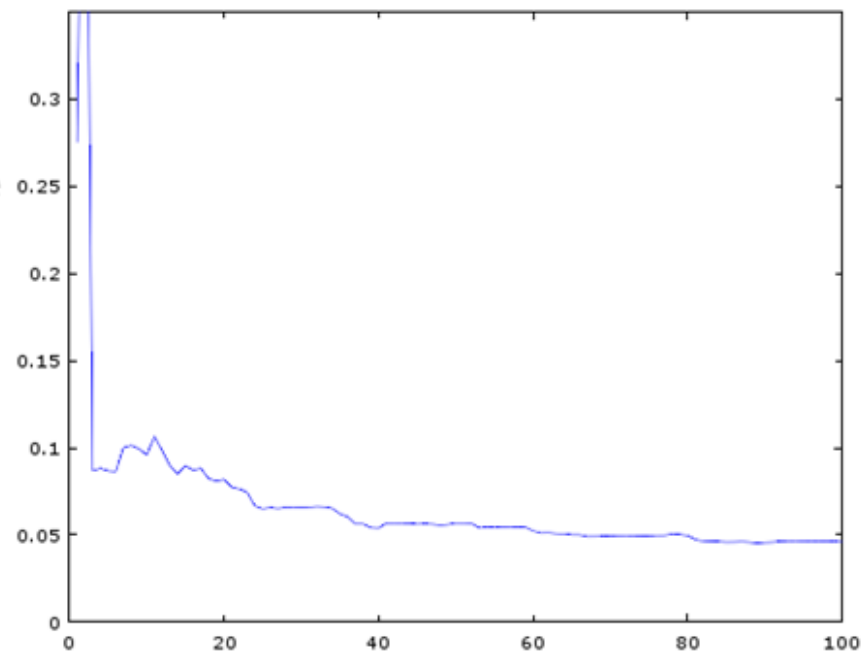
With $N > 50$, weights become stable

The test set error converges to the noise variance of 0.04

ls-weights

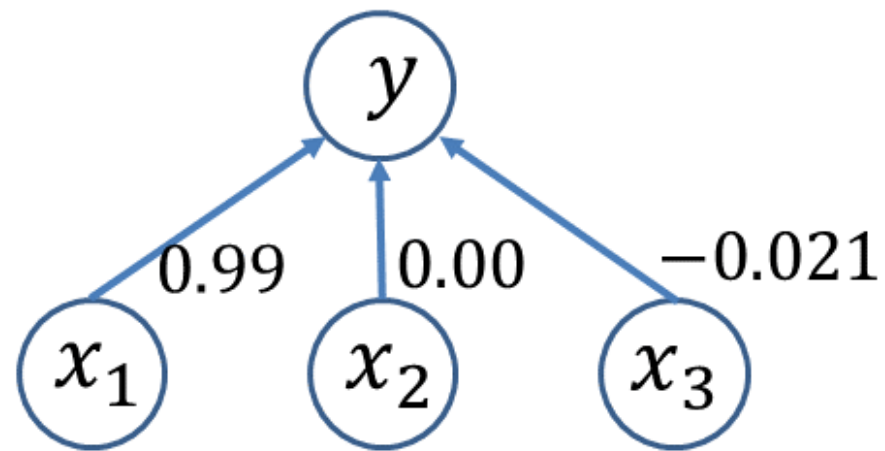
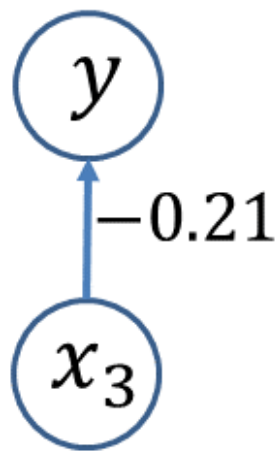
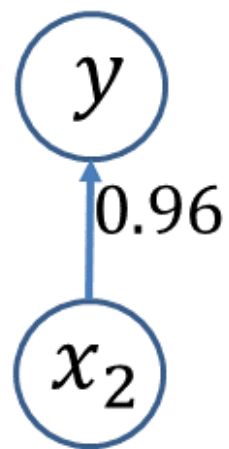
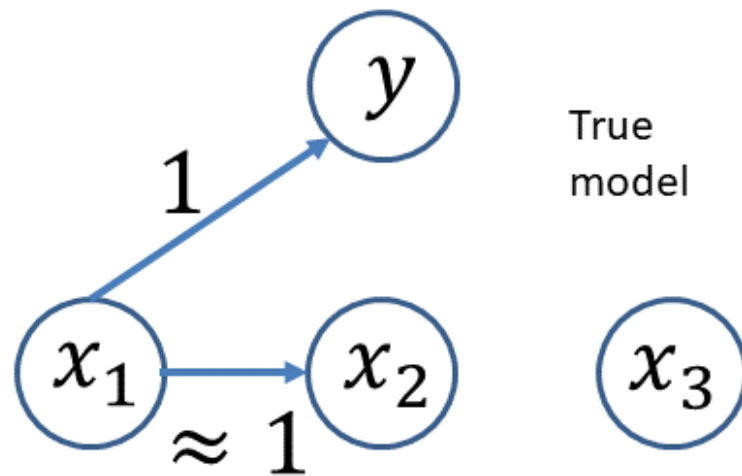


average test set error



The Power of Supervised Learning

- Consider that I can write the cost function as $\sum_i ([y_i - w_1 x_{i,1} - w_2 x_{i,2}] - w_3 x_{i,3})$. Thus $w_3 x_{i,3}$ only needs to fit the residual target $[y_i - w_1 x_{i,1} - w_2 x_{i,2}]$ instead of the original target y_i
- Of course this is true for any input: an input only has to model, what the other inputs could not model
- This is in contrast to a unidimensional analysis, where each input on its own tries to model the dependency to y as well as possible!



Unidimensional analysis

Multidimensional analysis
(penalized least squares)

Toy Example: Penalized Least Squares Regression

- We get with $\lambda = 0.6$:

$$\mathbf{X}^T \mathbf{X} + \lambda I = \begin{bmatrix} 100.6 & 98 & -19 \\ 98 & 100.6 & -17 \\ -19 & -17 & 100.6 \end{bmatrix}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda I)^{-1} = \begin{bmatrix} 0.197 & -0.191 & 0.005 \\ -0.191 & 0.195 & -0.003 \\ 0.005 & -0.003 & 0.010 \end{bmatrix}$$

$$\mathbf{X}^T \mathbf{y} = (99, 97, -20)^T$$

$$\mathbf{w}_{pen} = (\mathbf{X}^T \mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \mathbf{y} = (0.990, -0.005, -0.021)^T$$

- Note that $\mathbf{w}_{pen,2}$ is even closer to ground truth!

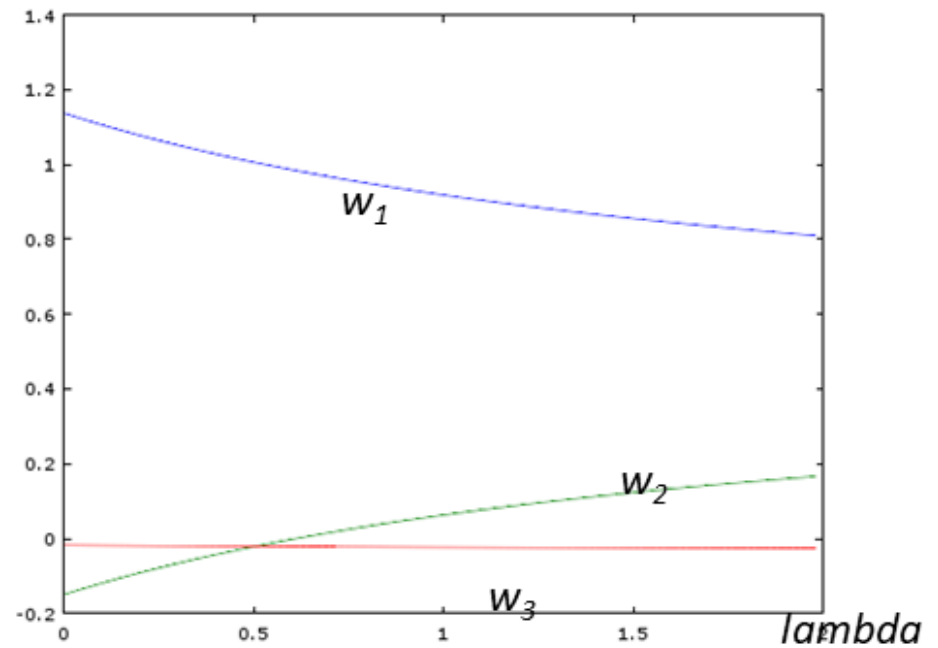
Penalized Least Squares

With large λ , w_1 and w_2 converge to identical values

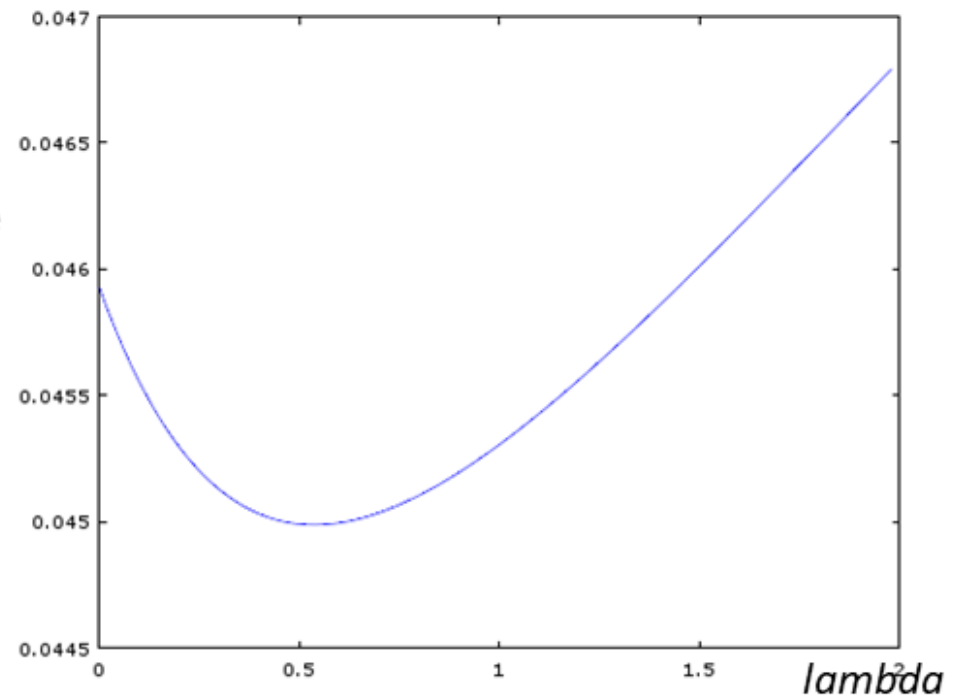
The test set error show that $\lambda = 0.6$ is a good value

Around $\lambda = 0.6$ the weight estimates are $(0.98, 0.00, -0.02)$

Penalized
Least
Squares
-weights



average
test set
error

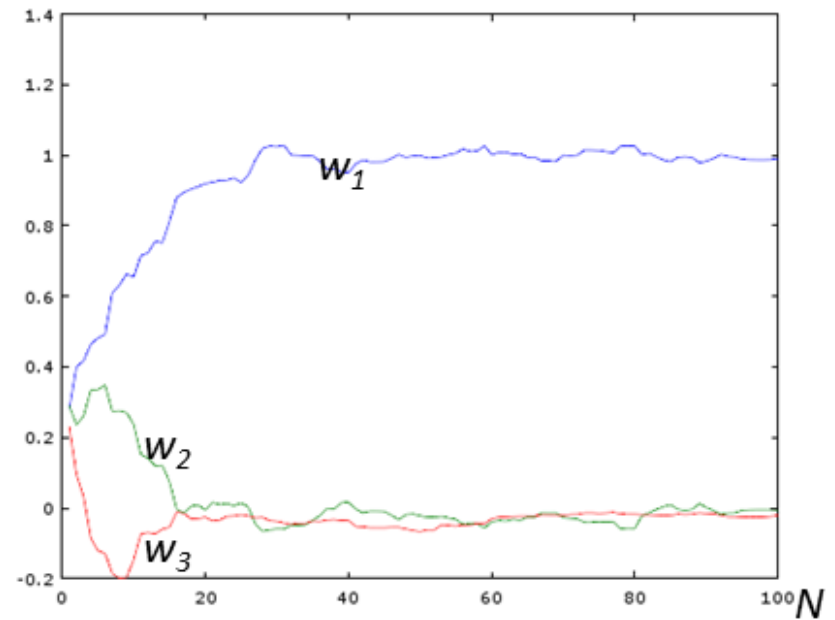


Penalized Least Squares Regression ($\lambda = 0.6$):

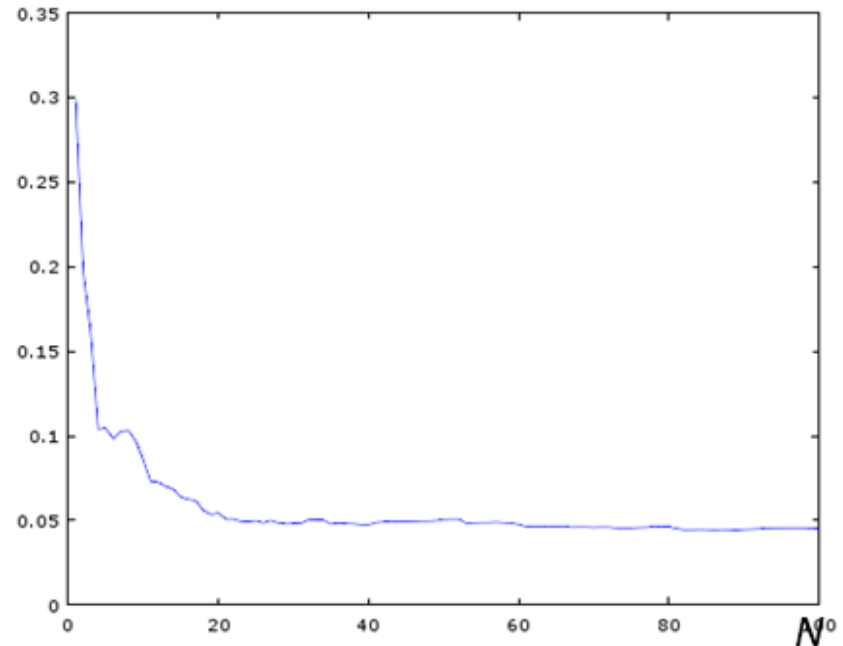
With small N , weights are close to 0 and test set error is large

With $N > 50$, weight estimates and test set error converges to the noise variance of 0.04

Penalized Least Squares -weights



average test set error



Remarks: Correlation versus Regression

- The Pearson coefficient is independent of context, objective. Karl Pearson: “I interpreted that sentence of Francis Galton (1822-1911) [his advisor] to mean that there was a category broader than causation, namely correlation, of which causation was only the limit, and that this new conception of correlation brought psychology, anthropology, medicine, and sociology in large parts into the field of mathematical treatment.”
- But the The Pearson correlation coefficient does not reflect causality (dependencies)
- The regression coefficients display causal behavior, much more closely

Remarks: Regularization

- If one is only interested in prediction accuracy: adding inputs liberally in regression can be beneficial if regularization is used (in ad placements and ad bidding, hundreds or thousands of features are used)
- The weight parameters of useless (noisy) features become close to zero with regularization (ill-conditioned parameters)
- Regularization is especially important when $N \approx M$, or when $N < M$
- If parameter interpretation is essential or if, for computational reasons, one wants to keep the number of inputs small:
 - — Forward selection; start with the empty model; at each step add the input that reduces the error most
 - — Backward selection (pruning); start with the full model; at each step remove the input that increases the error the least
- But no guarantee, that one finds the best subset of inputs or that one finds the true inputs

Experiments with Real World Data: Data from Prostate Cancer Patients

8 Inputs, 97 data points; y: Prostate-specific antigen

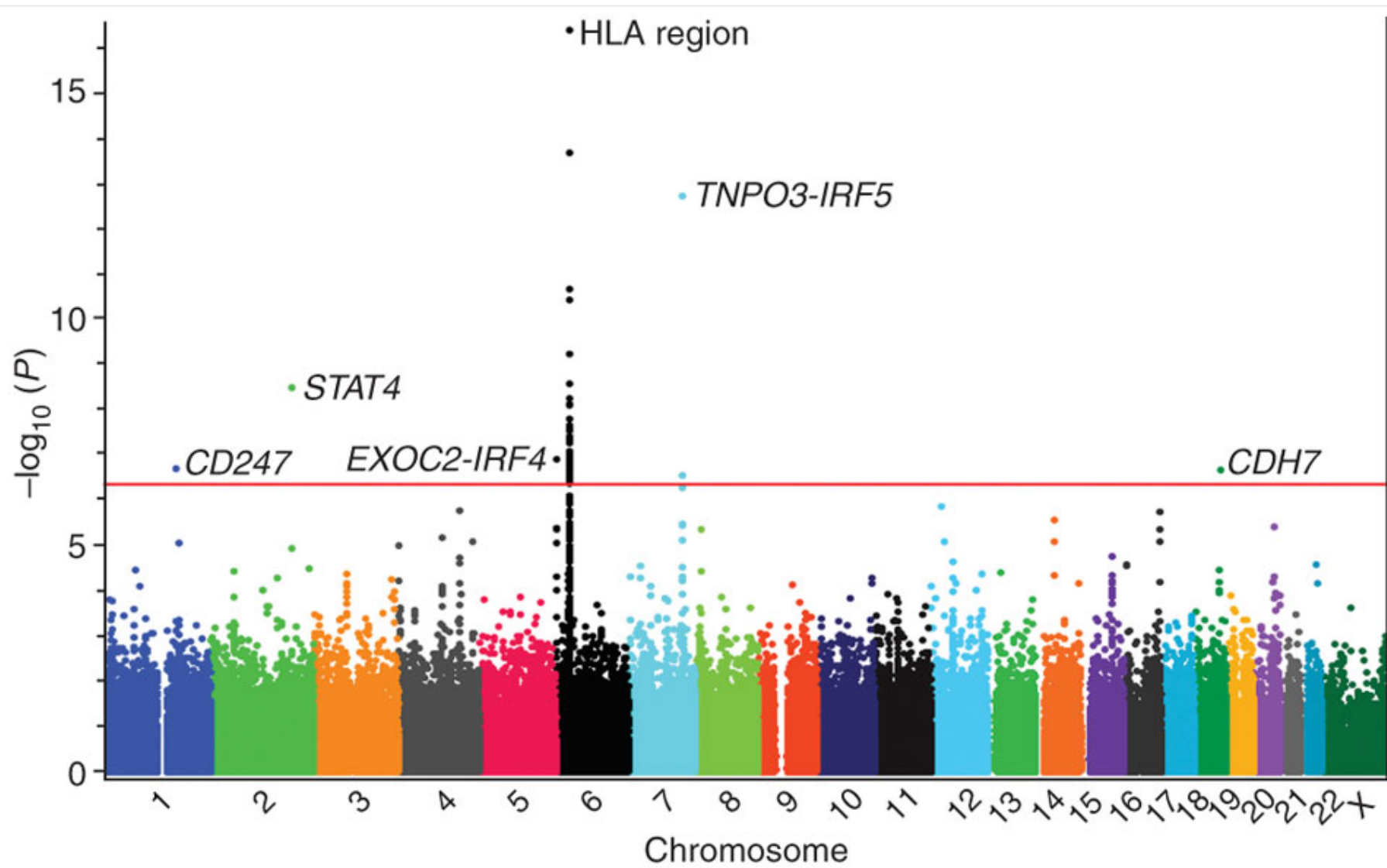
	LS	0.586
10-times cross validation error	Best Subset (3)	0.574
	Ridge (Penalized)	0.540

Examples where High-dimensional Linear Systems are Used

- Ranking in search engines (relevance of a web page to a query)
- Ad placements: where to put which ad for a user
- GWAS

Genome-wide Association Study (GWAS)

- Trait (here: the disease systemic sclerosis) is the output and the SNPs are the inputs
- The major allele is encoded as 0 and the minor allele as 1. Thus w_j is the influence of SNP j on the trait.
- Shown is the (log of the p-value) of w_j ordered by the locations on the chromosomes. The weights can be calculated by penalized least squares (ridge regression)
- Solely based on the Pearson correlation, the plot would show many more (non-causal) associations. The regression analysis reduces the apparent influence of noncausal correlated inputs and the influence of uncorrelated inputs
- In practice one often uses an elastic net penalty: $\lambda_2 \sum_j w_j^2 + \lambda_1 \sum_j |w_j|$ where the lasso penalty $\lambda_1 \sum_j |w_j|$ increases sparsity



Uncorrelated Inputs

- With uncorrelated inputs, we can train for each input separately, and obtain for each j

$$\hat{y} = w_{0,j} + w_{1,j}x_j$$

- We can then combine and obtain

$$\hat{y} = w_0 + \sum_j w_{1,j}x_j$$

where

$$w_0 = \frac{1}{M} \sum_j (w_{0,j} - w_{1,j}\bar{x}_j)$$

where \bar{x}_j is the mean of x_j

Appendix: A Deeper Analysis of Pearson versus Regression*

Pearson

- The Pearson correlation coefficient is in the mean approximately $(1, 1, 0)$. The variance of r_1 , and r_2 can be estimated as $\text{var} = \sigma^2/N = 0.04/100 = 0.0004$ and standard deviation $\text{stdev} = \sqrt{\text{var}} = 0.02$ (we use the true noise variance here; $N = 100$).
- For r_3 we get a variance $\text{var}(r_3) = \text{var}(y)/N = 1/N = 0.01$, and a standard deviation of $\text{stdev}(r_3) = 0.1$; here we set $\text{var} = 1$, since for x_3 all data appears as noise
- Comment: r_2 does not reflect the true dependency; the variance of r_3 is relatively large.

Regression

- Since linear regression is unbiased, the parameter estimates have mean 1, 0, 0 (unbiased solutions). We get for the covariances

$$\text{COV}(\mathbf{w}_{ls}) = \sigma^2 (X^T X)^{-1}$$

The variances are then (we consider the diagonal terms)

$$\text{var}(w_{ls,1}) = \text{var}(w_{ls,2}) \approx 0.04 \times 0.25 = 0.01$$

$$\text{var}(w_{ls,3}) \approx 0.04 \times 0.01 = 0.0004,$$

and

$$\text{stdev}(w_{ls,1}) = \text{stdev}(w_{ls,2}) \approx 0.1$$

$$\text{stdev}(w_{ls,3}) \approx 0.02$$

- Thus the estimates are unbiased; the uncertainty of $w_{ls,3}$ is greatly reduced and thus closer to zero!