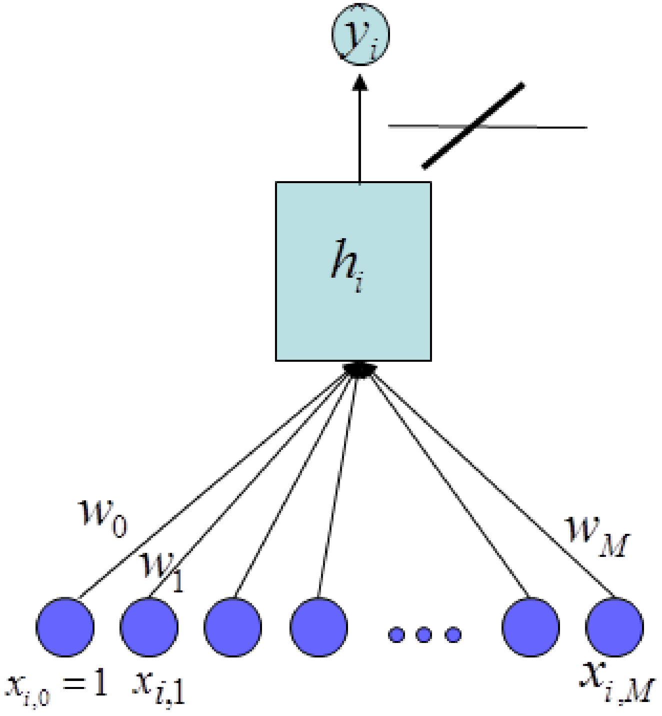
# Linear Regression

Volker Tresp
Winter 2023-2024

# 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

$$h(\mathbf{x}) = \sum_{j=0}^{M} w_j x_j$$

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

- The activation is the output (no thresholding)

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

- Regression: the target function can take on real values

# Effect of a Single Binary Input

- Consider only binary inputs with $x_{i,j} \in \{0, 1\}$

- When $x_{i,j}$ switches from 0 to 1 and the other inputs remain fixed (intervention), the $j$-th input adds to the output the quantity $w_j$, independent of context, i.e., independent of the other inputs! (the average causal effect is identical to the individual causal effect)

- Recall that for the perceptron, the effect of an input on the output does critically depend on context: When $x_{i,j}$ switches from 0 to 1 and the other inputs remain fixed (intervention), dependent on the other inputs, the output might stay as is, or changes from 1 to $-1$ or from $-1$ to 1

# 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:

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

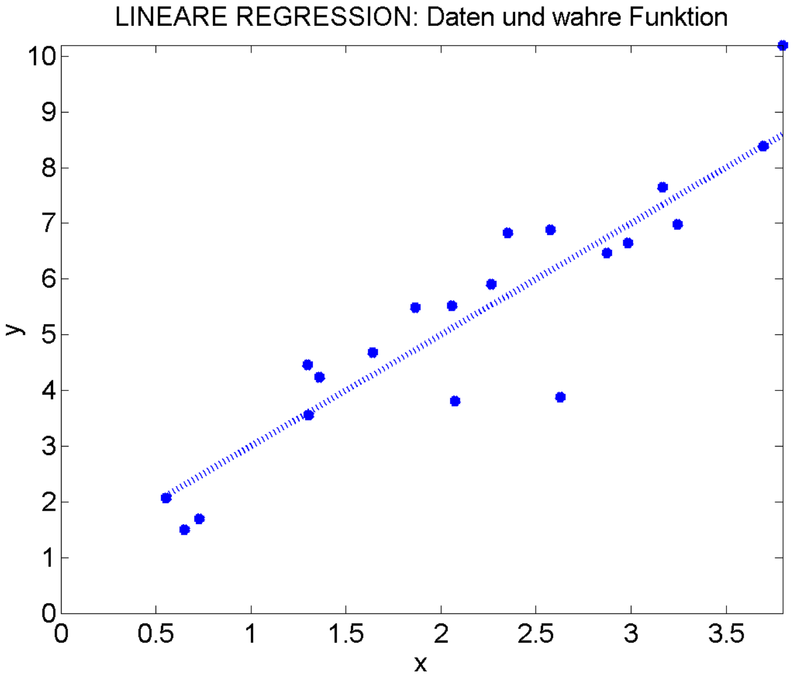
$$\mathbf{w} = (w_0, w_1)^T$$

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:

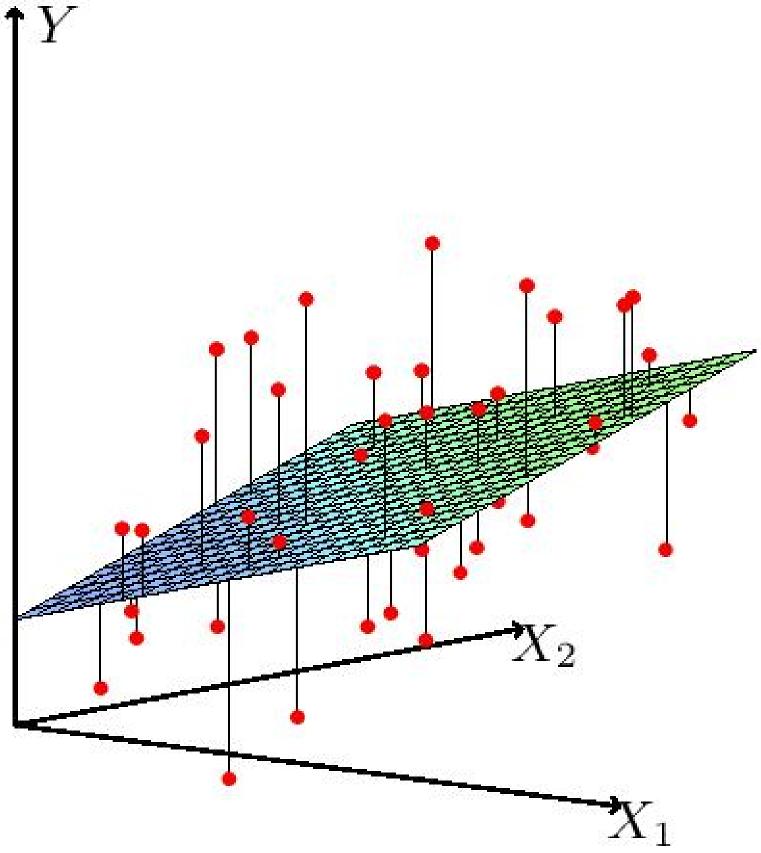
$$\widehat{y}_i = f_{\mathbf{w}}(\mathbf{x}_i) = w_0 + \sum_{j=1}^{M} w_j x_{i,j}$$

$$= \mathbf{x}_i^T \mathbf{w}$$

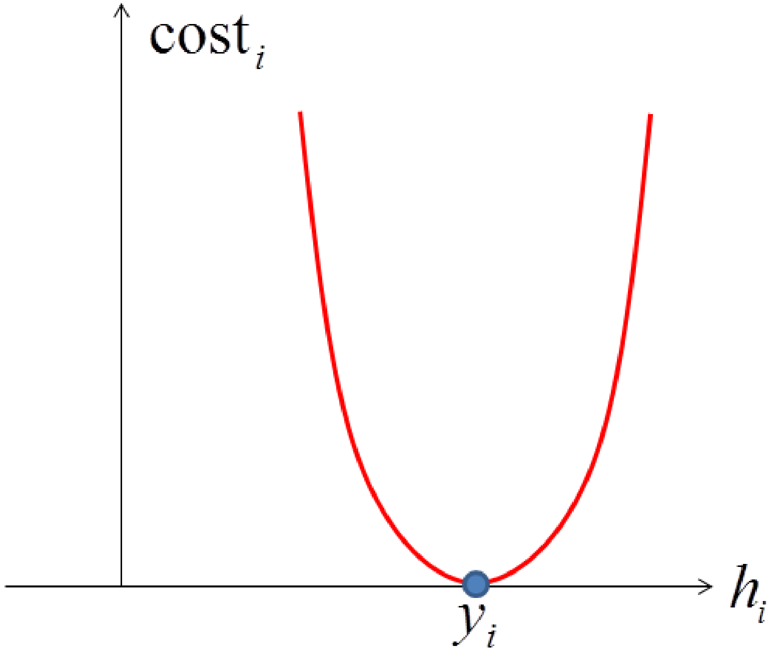
$$\mathbf{w} = (w_0, w_1, \ldots w_M)^T$$

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

# 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

$$\widehat{\mathbf{y}} = \begin{pmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \cdots \\ \widehat{y}_N \end{pmatrix} = \mathbf{X}\mathbf{w}$$

# 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

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

# 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 $t$. The we adapt, $t = 1, 2, \ldots$

$$w_j \longleftarrow w_j + \eta(y_t - \widehat{y}_t)x_{t,j} \quad j = 0, 1, 2, \ldots, M$$

- $\eta > 0$ is the learning rate, typically $0 < \eta << 0.1$
- Depending again on the difference (“delta”) $(y_t - \widehat{y}_t)$, this is again called a delta rule
- This is identical to the Perceptron learning rule (see Appendix in the lecture on the Perceptron). But, for the Perceptron $y_t \in \{-1, 1\}$, and $\widehat{y}_t \in \{-1, 1\}$

# 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

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

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

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

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

# Necessary Condition for an Optimum

- A necessary condition for an optimum is that

$$\left.\frac{\partial \mathsf{cost}(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_{opt}} = 0$$

# 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)

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

$$= -2 \sum_{i=1}^{N} x_{i,1}(y_i - x_{i,1} w_1) = -2 \sum_{i=1}^{N} x_{i,1} y_i + 2 w_1 \sum_{i=1}^{N} x_{i,1} x_{i,1}$$

- Thus

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

# General Case

- $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ and $\mathrm{cost}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$

- (chain rule: inner derivative times outer derivative)

$$\frac{\partial \mathrm{cost}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial(\mathbf{y} - \mathbf{X}\mathbf{w})}{\partial \mathbf{w}} 2(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w}$$

- Thus

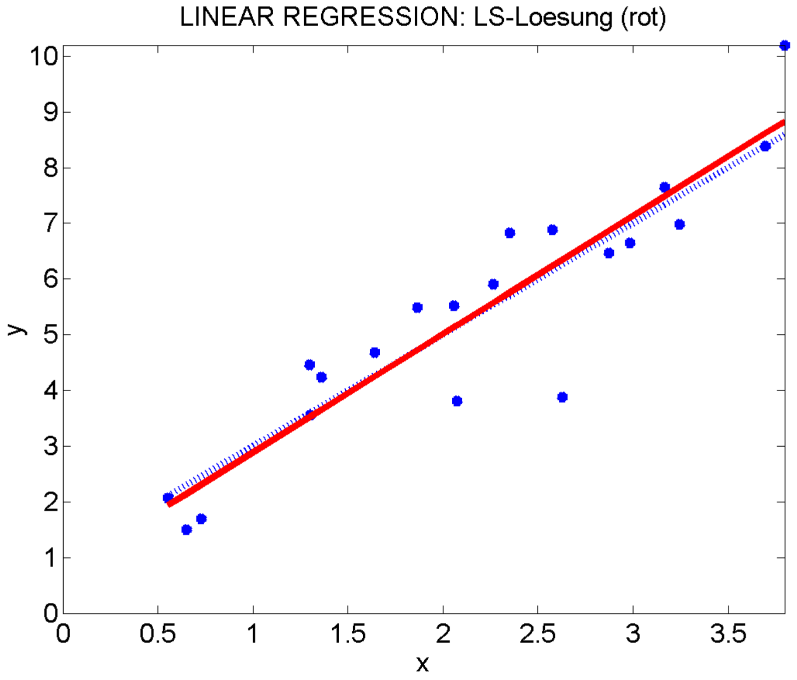
$$\mathbf{w}_{ls} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

# Setting First Derivative to Zero

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

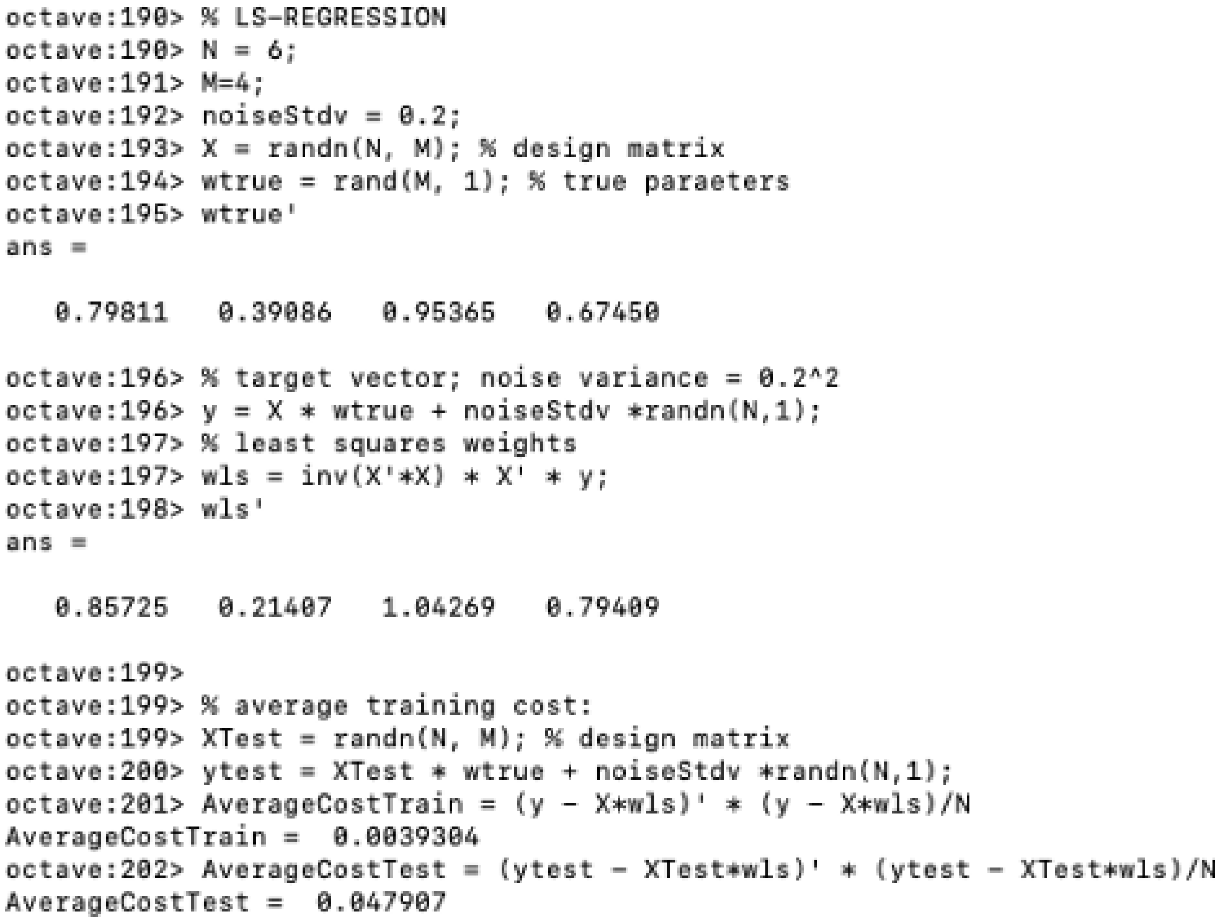
Complexity (linear in $N$):

$$\mathcal{O}(M_p^3 + NM_p^2)$$



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

# LS-Regression (octave/matlab)



```
octave:190> % LS-REGRESSION
octave:190> N = 6;
octave:191> M=4;
octave:192> noiseStdv = 0.2;
octave:193> X = randn(N, M); % design matrix
octave:194> wtrue = rand(M, 1); % true paraeters
octave:195> wtrue'
ans =

   0.79811   0.39086   0.95365   0.67450

octave:196> % target vector; noise variance = 0.2^2
octave:196> y = X * wtrue + noiseStdv *randn(N,1);
octave:197> % least squares weights
octave:197> wls = inv(X'*X) * X' * y;
octave:198> wls'
ans =

   0.85725   0.21407   1.04269   0.79409

octave:199>
octave:199> % average training cost:
octave:199> XTest = randn(N, M); % design matrix
octave:200> ytest = XTest * wtrue + noiseStdv *randn(N,1);
octave:201> AverageCostTrain = (y - X*wls)' * (y - X*wls)/N
AverageCostTrain =  0.0039304
octave:202> AverageCostTest = (ytest - XTest*wls)' * (ytest - XTest*wls)/N
AverageCostTest =  0.047907
```

# Derivatives of Vector Products

- We have used

$$\frac{\partial}{\partial \mathbf{w}} \mathbf{X}\mathbf{w} = \mathbf{X}^T \qquad \frac{\partial}{\partial \mathbf{w}} \mathbf{w}^T \mathbf{w} = 2\mathbf{w} \qquad \frac{\partial}{\partial \mathbf{w}} \mathbf{w}^T \mathbf{X}\mathbf{w} = (\mathbf{X} + \mathbf{X}^T)\mathbf{w}$$

- Comment: one also finds the conventions,

$$\frac{\partial}{\partial \mathbf{w}} \mathbf{X}\mathbf{w} = \mathbf{X} \qquad \frac{\partial}{\partial \mathbf{w}} \mathbf{w}^T \mathbf{w} = 2\mathbf{w}^T \qquad \frac{\partial}{\partial \mathbf{w}} \mathbf{w}^T \mathbf{X}\mathbf{w} = \mathbf{w}^T(\mathbf{X} + \mathbf{X}^T)$$

# Machine Learning as an Inverse Problem

- The map $\mathbf{w} \mapsto \mathbf{y} = \mathbf{X}\mathbf{w}$ is the forward model
- The map $\mathbf{y} \mapsto \mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ is the inverse model
- $\mathbf{X}^+ = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is called the Moore-Penrose pseudo inverse (generalized inverse); (Roger Penrose won the 2020 Nobel Prize in Physics)
- Machine learning is an “inverse” problem

# Stability of the Solution

- When $N >> M_p$, the LS solution is stable (small changes in the data lead to small changes in the parameter estimates)
- When $N < M_p$ 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 $N > M_p$ 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

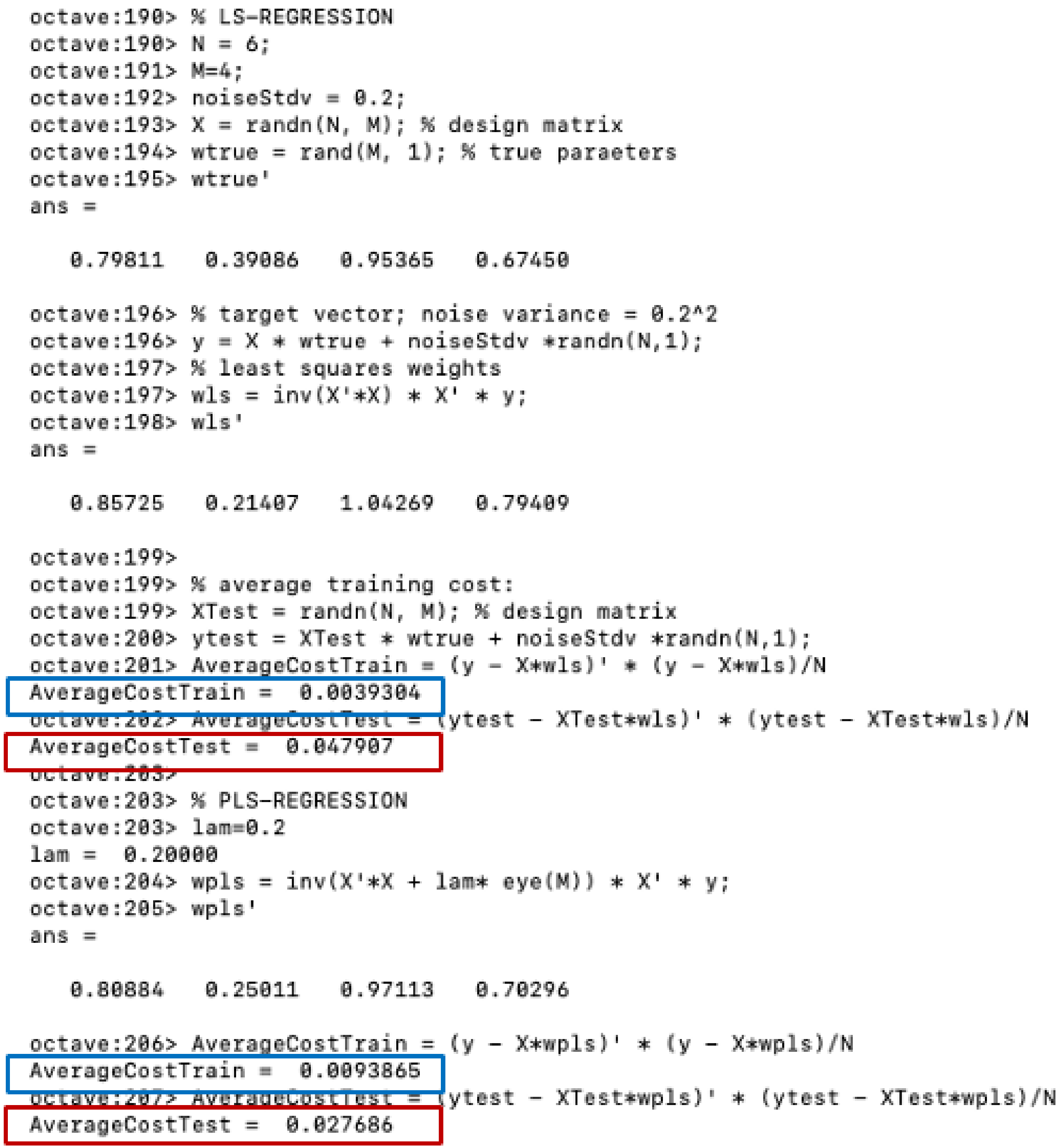
$$\mathrm{cost}^{pen}(\mathbf{w}) = \sum_{i=1}^{N}(y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2 + \lambda \sum_{j=0}^{M} w_j^2$$

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

Derivation:

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

## PLS-Regression (octave/matlab)



```
octave:190> % LS-REGRESSION
octave:190> N = 6;
octave:191> M=4;
octave:192> noiseStdv = 0.2;
octave:193> X = randn(N, M); % design matrix
octave:194> wtrue = rand(M, 1); % true paraeters
octave:195> wtrue'
ans =

   0.79811   0.39086   0.95365   0.67450

octave:196> % target vector; noise variance = 0.2^2
octave:196> y = X * wtrue + noiseStdv *randn(N,1);
octave:197> % least squares weights
octave:197> wls = inv(X'*X) * X' * y;
octave:198> wls'
ans =

   0.85725   0.21407   1.04269   0.79409

octave:199>
octave:199> % average training cost:
octave:199> XTest = randn(N, M); % design matrix
octave:200> ytest = XTest * wtrue + noiseStdv *randn(N,1);
octave:201> AverageCostTrain = (y - X*wls)' * (y - X*wls)/N
AverageCostTrain =  0.0039304
octave:202> AverageCostTest = (ytest - XTest*wls)' * (ytest - XTest*wls)/N
AverageCostTest =  0.047907
octave:203>
octave:203> % PLS-REGRESSION
octave:203> lam=0.2
lam =  0.20000
octave:204> wpls = inv(X'*X + lam* eye(M)) * X' * y;
octave:205> wpls'
ans =

   0.80884   0.25011   0.97113   0.70296

octave:206> AverageCostTrain = (y - X*wpls)' * (y - X*wpls)/N
AverageCostTrain =  0.0093865
octave:207> AverageCostTest = (ytest - XTest*wpls)' * (ytest - XTest*wpls)/N
AverageCostTest =  0.027686
```

# ADALINE-Learning Rule with Weight Decay

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

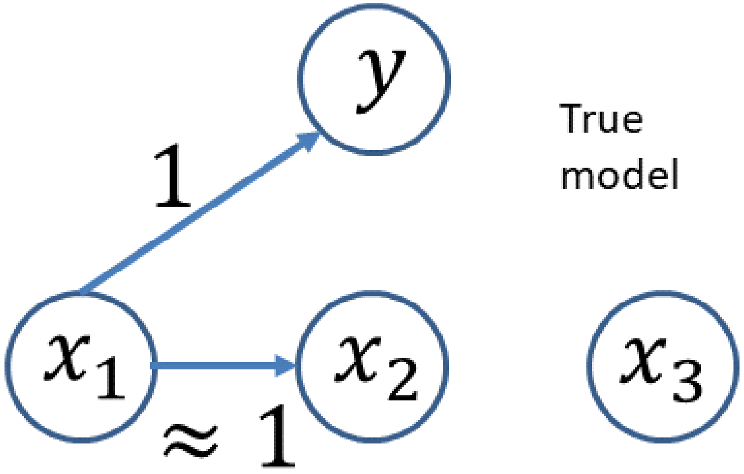
$$w_j \longleftarrow w_j + \eta[(y_t - \widehat{y}_t)x_{t,j} - \frac{\lambda}{N} w_j] \quad j = 0, 1, 2, \ldots, M$$

# Toy Example

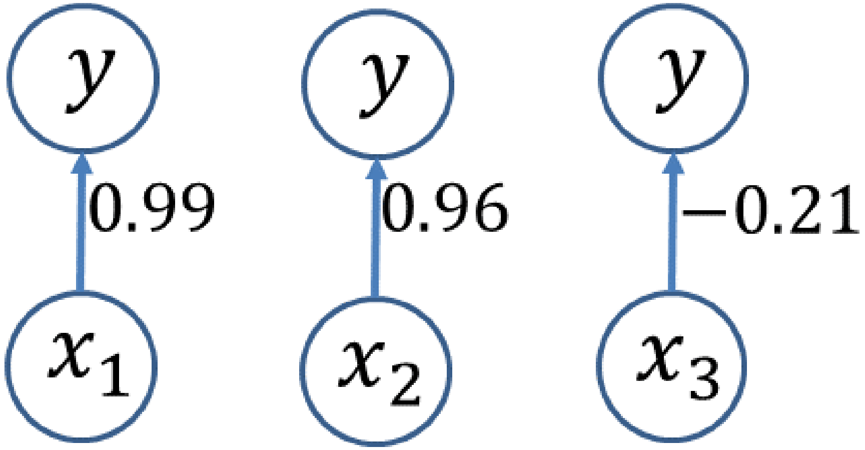
- We generated $N = 100$ data points with $M = 3$ inputs ($w_0 = 0$ is known)
- $x_1$ and $x_2$ are highly correlated ($x_2$ is generated from $x_1$)
- $x_3$ is independent of all the other variables: $x_1$, $x_2$, and $y$
- We generate output data with $y = x_1 + \epsilon$, where $\epsilon$ stands for independent noise with standard deviation 0.2 and thus variance of 0.04.
- Thus the true parameters are $\mathbf{w}_{true} = (1, 0, 0)^T$; Note that, $y$ causally only depends on $x_1$
- Thus the **true function** is

$$y = 1 \times x_1 + 0 \times x_2 + 0 \times x_3 + \epsilon$$

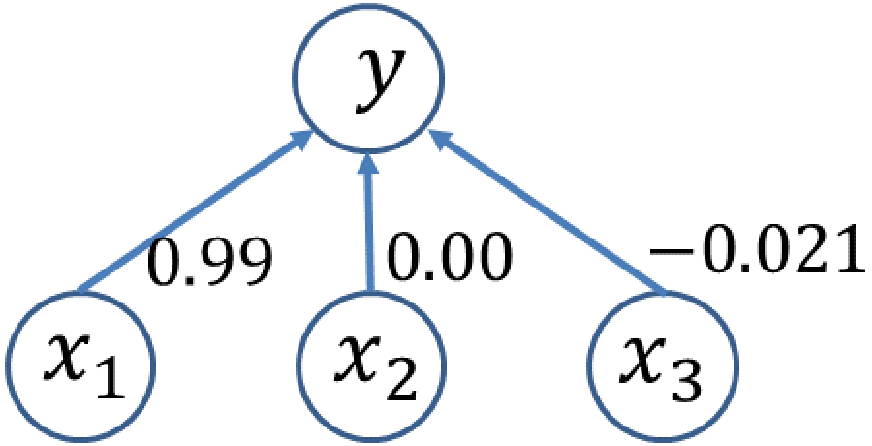
- Thus, $\mathbf{w}_{\textit{true}} = (1, 0, 0)^T$ (without $w_0$)
- All variables $x_1, x_2, x_3, y$ are normalized to zero mean and variance 1



True model



Unidimensional analysis



Multidimensional analysis
(penalized least squares)

# Toy Example: One-dimensional Model

- In one-dimensional models, with only one input, the weights are identical to the sample **Pearson correlation coefficients** (here: $\widehat{w}_j = \widehat{r}_j = \sum_i y_i x_{i,j}/N$) between the output and the input
- I obtain $\widehat{r}_1 = \widehat{w}_1 = 0.99$, $\widehat{r}_2 = \widehat{w}_2 = 0.96$, $\widehat{r}_3 = \widehat{w}_3 = -0.21$
- Explicitly, the three **one-dimensional models** are

$$\widehat{y} = 0.99 x_1 \qquad \widehat{y} = 0.96 x_2 \qquad \widehat{y} = -0.21 x_3$$

The expected Pearson correlation coefficient is defines as (for zero mean variables)

$$r_{x,y} = cov_{xy}/(var_x var_y)$$

# Toy Example: One-dimensional Model (cont'd)

- A deeper analysis reveals that the estimate ($\mathbb{E}$: expected value; *stdev*: standard deviation)

$$\mathbb{E}(\widehat{w}_1) = 1 \quad \mathit{stdev}(\widehat{w}_1) = 0.02$$

  $w_1$ reflects the causal dependency of $y$ on $x_1$

- The second coefficient, $\widehat{r}_2 = 0.96$, 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).

$$\mathbb{E}(\widehat{w}_2) = 1 \quad \mathit{stdev}(\widehat{w}_2) = 0.02$$

- The third value $w_3$ is correctly closer to 0, but not really small in magnitude.

$$\mathbb{E}(\widehat{w}_3) = 0 \quad \mathit{stdev}(\widehat{w}_3) = 0.1$$

# 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 $N\mathrm{cov}(\mathbf{x})$; 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 $N \times (\hat{r}_1, \hat{r}_2, \hat{r}_3)^T$! We see the strong correlation between both $x_1$ and $x_2$ with $y$

# Toy Example: Least Squares Regression (cont'd)

- We get

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

- Interestingly, linear regression pretty much identifies the correct causality, with $\widehat{w}_1 \approx 1$ and $\widehat{w}_2 \approx 0$ !

$$\mathbb{E}(\widehat{w}_1) = 1 \quad \mathit{stdev}(\widehat{w}_1) = 0.1$$

  So the estimator is unbiased but the uncertainty is larger then in the unit dimensional analysis

$$\mathbb{E}(\widehat{w}_2) = 0 \quad \mathit{stdev}(\widehat{w}_2) = 0.1$$

  Note the dramatic shift to 0!

# Toy Example: Least Squares Regression (cont'd)

- $\widehat{w}_3 = -0.018$ is much closer to 0, compared to the sample Pearson correlation coefficient $\widehat{r}_3 = -0.21$

  $$\mathbb{E}(\widehat{w}_3) = 0 \quad \mathit{stdev}(\widehat{w}_3) = 0.02$$

  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 (non-causal) input $x_2$ and the noise input $x_3$ are largely reduced

# The Power of Supervised Learning

- In regression: an input only has to model, what the other inputs could not model
- In a one-dimensional analysis: each input on its own tries to model the dependency to $y$ as well as possible!

## Toy Example: Penalized Least Squares Regression

- We get with $\lambda = 0.6$:

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

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

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

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

- Note that $\hat{\mathbf{w}}_2$ is even closer to ground truth!

# Toy Example: Penalized Least Squares Regression (cont'd)

- With independent inputs ($\text{cov}(\mathbf{x}) \approx X^\top X/N$ is a diagonal matrix)

$$\mathbb{E}(\widehat{w}_{\textsf{pen},j}) = \frac{N}{N+\lambda}\mathbb{E}(\widehat{w}_{ls,j})$$

  and $\mathbb{E}(\widehat{w}_{ls,j}) = \mathbb{E}(r_{x_j,y})$

- The relative shrinkage is identical to all parameters, but the absolute shrinkage

$$\mathbb{E}(\widehat{w}_{\textsf{ls},j}) - \mathbb{E}(\widehat{w}_{\textsf{pen},j}) = \frac{\lambda}{\lambda+N}\mathbb{E}(\widehat{w}_{ls,j})$$

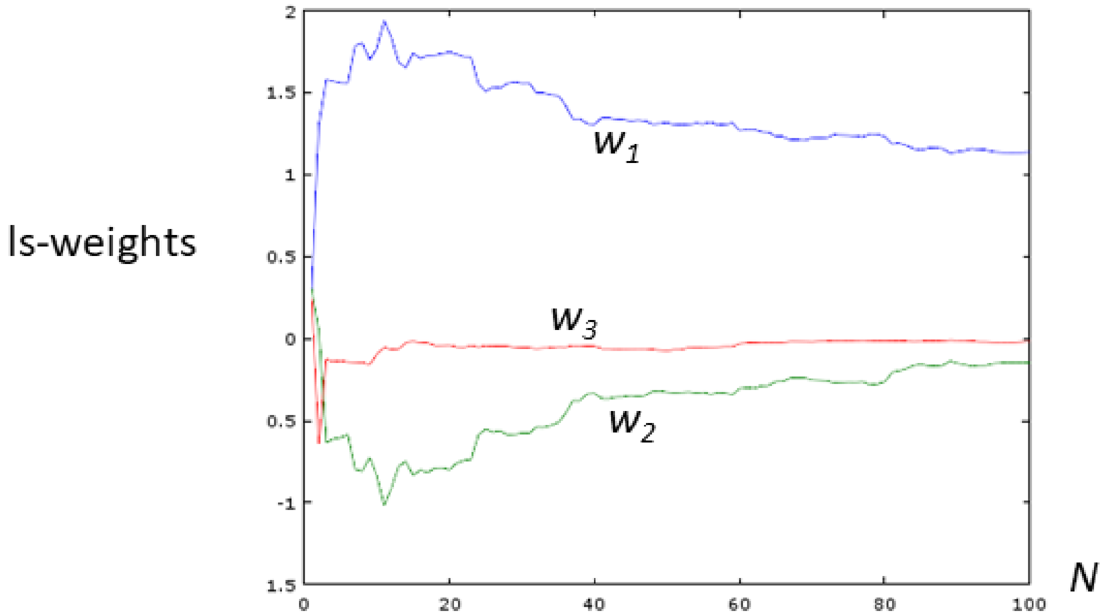
  is larger in magnitude for larger weights

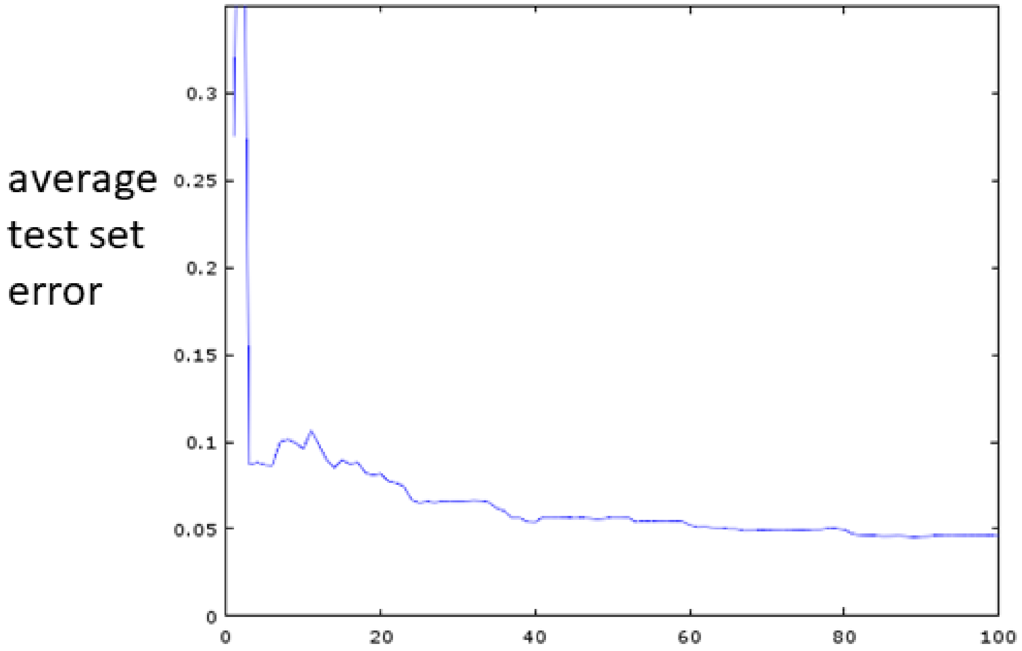
**Least Squares Regression:**

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*





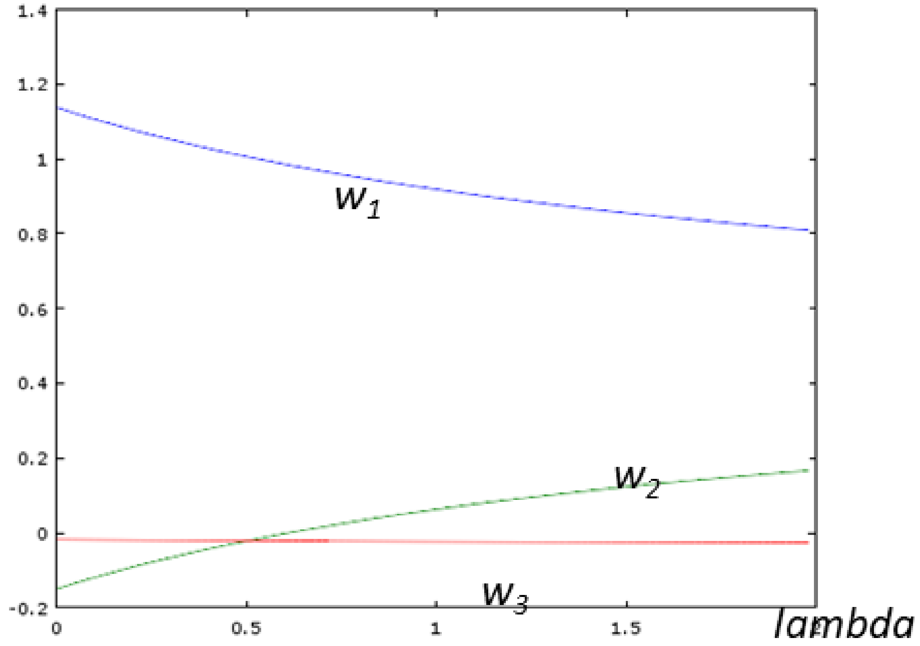
**Penalized Least Squares**

With large *lambda*, $w_1$ and $w_2$ converge to identical values

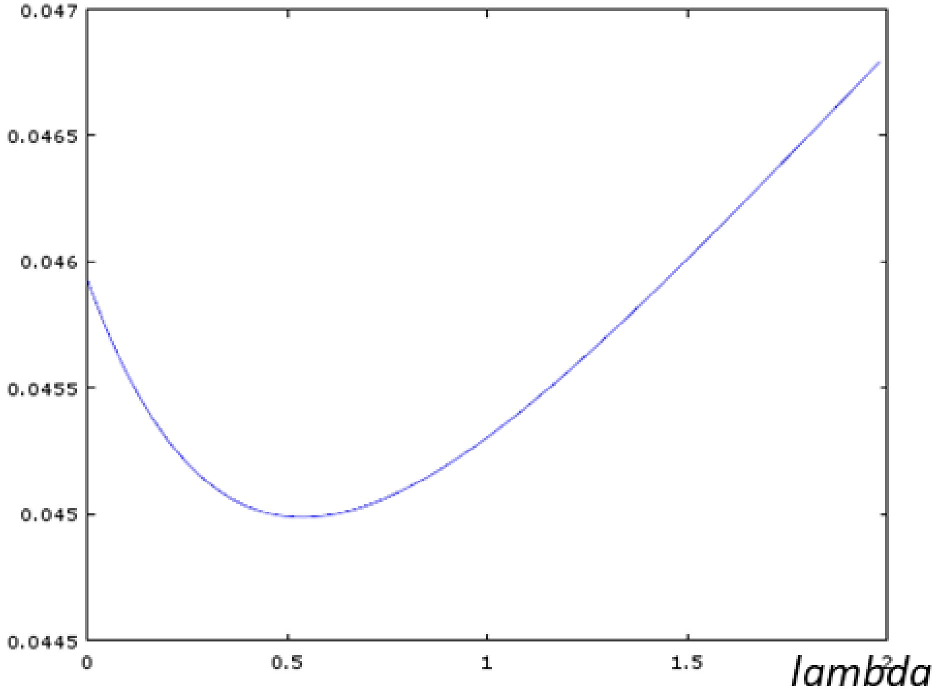
The test set error show that *lambda = 0.6* is a good value

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

Penalized Least Squares -weights



average test set error

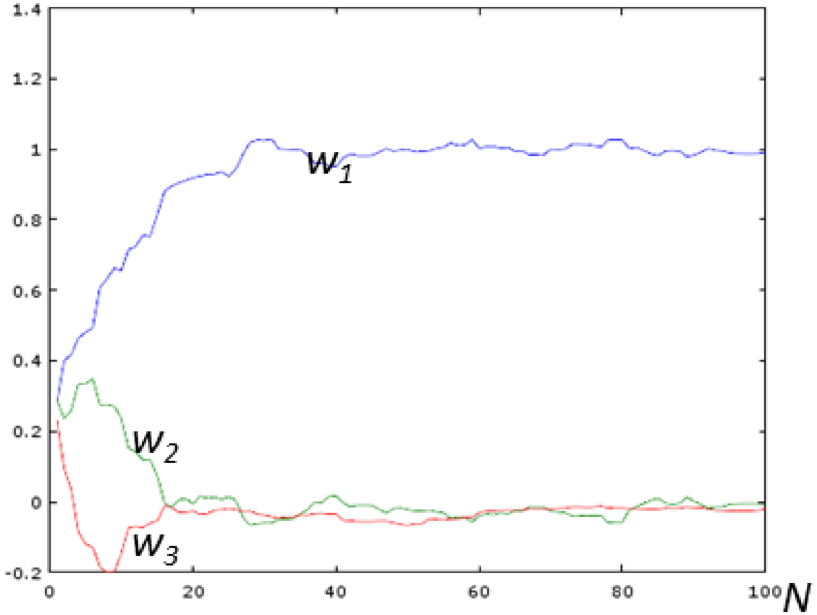


**Penalized Least Squares** Regression *(lambda = 0.6)*:

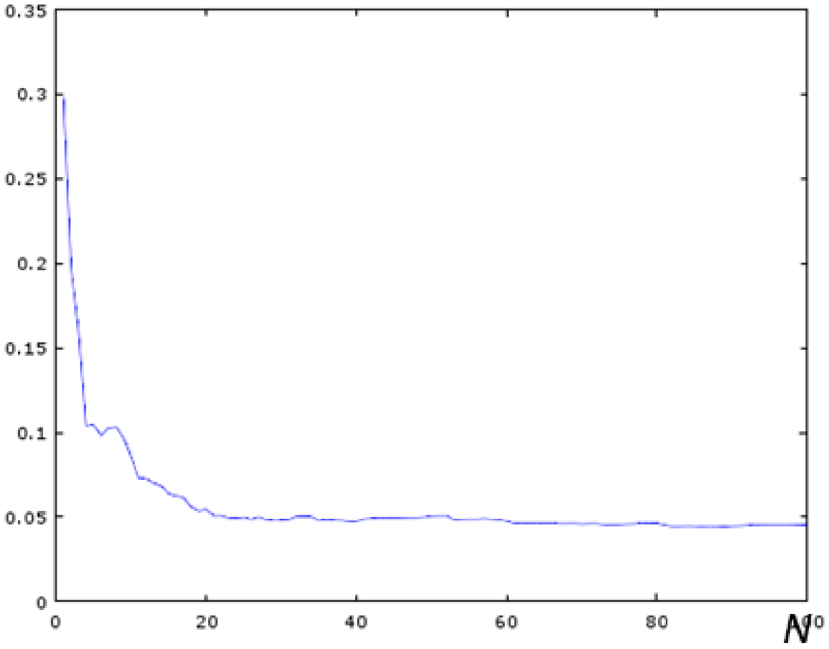
With small *N*, weights are close to *0* and test set error is large

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

Penalized Least Squares -weights



average test set error

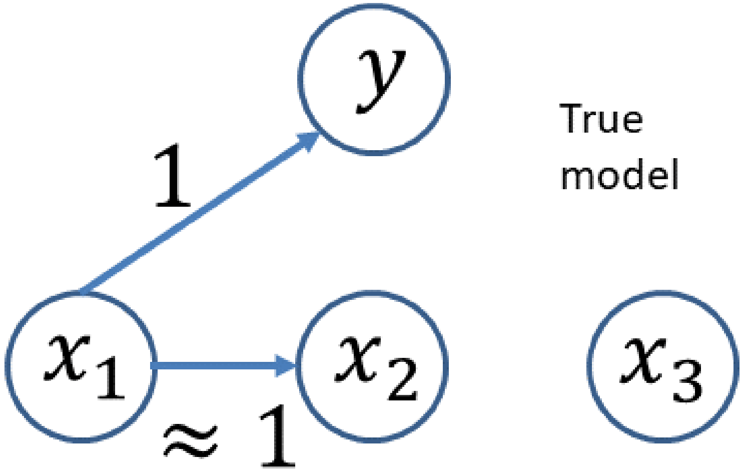


# Application to Healthcare

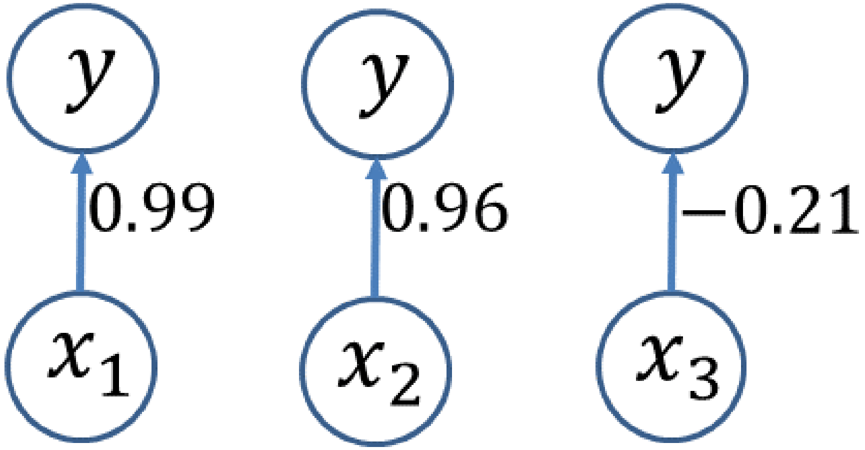
- Assume same data
- Consider that $x_2$ **is a treatment** (medication) and $y$ the outcome (healthy=1, after treatment).
- If I do a one-dimensional analysis, I would see a strong positive influence of $x_2$ on $y$
- I might conclude that the treatment works
- Only if I include the so-called **confounder** $x_1$ in the regression model, it becomes clear that the confounder $x_1$ is the cause and not the treatment $x_2$
- I conclude that the treatment has no significant effect!
- Recipe: Do a multidimensional regression model and include all relevant inputs!

# Application to Healthcare (cont'd)

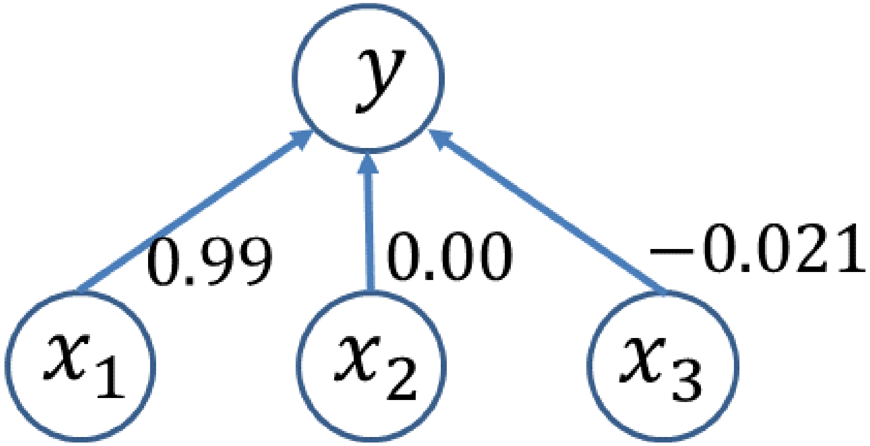
- Example: Only patients without any other disease ($x_1 = 1$) get the treatment, but they get healthy ($y = 1$), independent of treatment ($x_2$) (since their bodies can focus on the disease of interest); still, this results in a correlation between treatment ($x_2$) and outcome ($y$)
- Fisher's hypothesis (epidemiology): A certain gene variant ($x_1 = 1$) causes lung cancer, but also makes you want to smoke ($x_2 = 1$), but smoking itself has no effect on lung cancer $y$; still, this results in a correlation between smoking ($x_2$) and outcome ($y$)
- Now that we can measure genetic variance: Fisher's hypothesis is (mostly) wrong



True model



Unidimensional analysis



Multidimensional analysis
(penalized least squares)

# Remarks: Correlation versus Regression

- The Pearson correlation coefficients 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 Pearson correlation coefficient does not reflect causality (dependencies)

- The regression coefficients display causal behavior, much more closely: *causality analysis based on observed data requires complete models*

- “Gold standard”: prospective randomized controlled trial (RCT): assign patients randomly to the treatment group

- In epidemiological studies RCTs are often not ethical: you cannot just tell people to start smoking; here one often needs to rely on (carefully analysed) retrospective observational studies

# 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 $N \approx M_p$, or when $N < M_p$

- 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

| | | |
|---|---|---|
| 10-times cross validation error | LS | 0.586 |
| | 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 advertizement on a web page, for a user with a given user profile

# Appendix: Penalized Least Squares Regression (Advanced)

- Assume that all inputs have zero mean and a standard deviation of one (standard preprocessing); but inputs might be correlated

- I can change the coordinate system (no rescaling) with $\mathbf{z} = \mathbf{V}^T \mathbf{x}$, such that the components of $\mathbf{z}$ are uncorrelated which means that $\mathrm{cov}(\mathbf{z})$ is a diagonal matrix with $\mathrm{cov}(\mathbf{z})_{j,j} = d_j^2$

- In this new coordinate system we still have zero mean, but potentially unequal standard deviations

- In this new coordinate system,

$$\mathbb{E}(\hat{w}^z_{\mathsf{pen},j}) = \frac{N}{N + \lambda/d_j^2} \mathbb{E}(\hat{w}^z_{ls,j})$$

- This means that weights in dimensions with a small $d_j^2$ are relatively (not absolutely) shrunk more

- With $\mathit{Var}(\widehat{w}^z_{ls,j}) = \sigma^2/(Nd_j^2)$, we get more shrinkage for weights which are uncertain
- In Hastie et al., “The Elements of Statistical Learning”, it is shown that this shrinkage also occurs in the original coordinate system, without a diagonalization as preprocessing