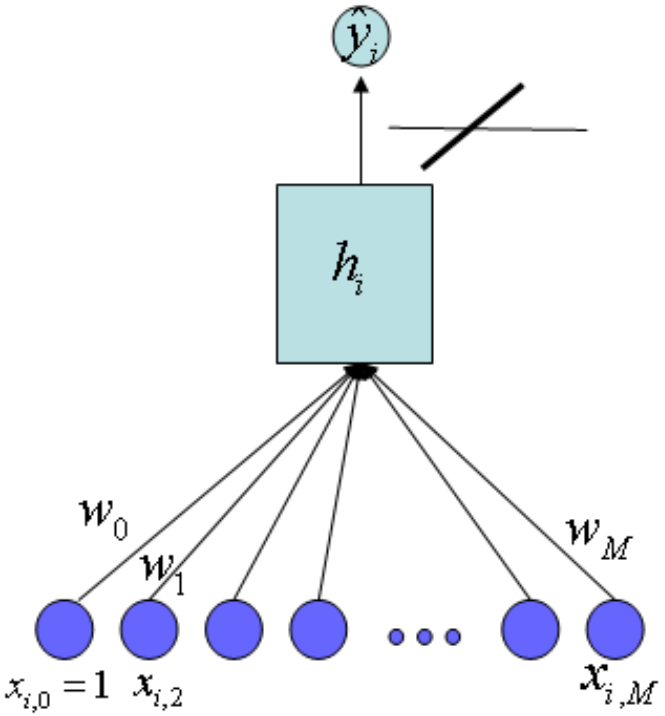
Linear Regression

Volker Tresp
2015

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

$$h_i = \sum_{j=0}^{M-1} w_{i,j} x_{i,j}$$

(Note: $x_{i,0} = 1$ is a constant input, so that w_0 is the bias)

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

$$\widehat{y}_i = f(\mathbf{x}_i) = h_i$$

- Regression: the target function can take on real values

Method of Least Squares

- Squared-loss cost function:

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

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

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

- For visualization, on chooses $M = 2$ (although linear regression is often applied to high-dimensional inputs)

Least-squares Estimator for Regression

One-dimensional regression:

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

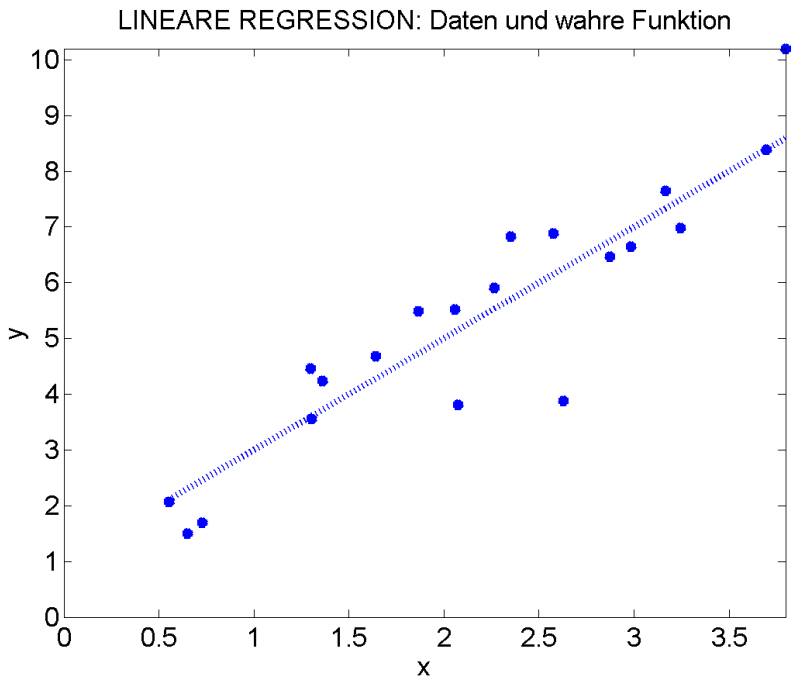
$$\mathbf{w} = (w_0, w_1)^T$$

Squared error:

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

Goal:

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



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

Least-squares Estimator in General

General Model:

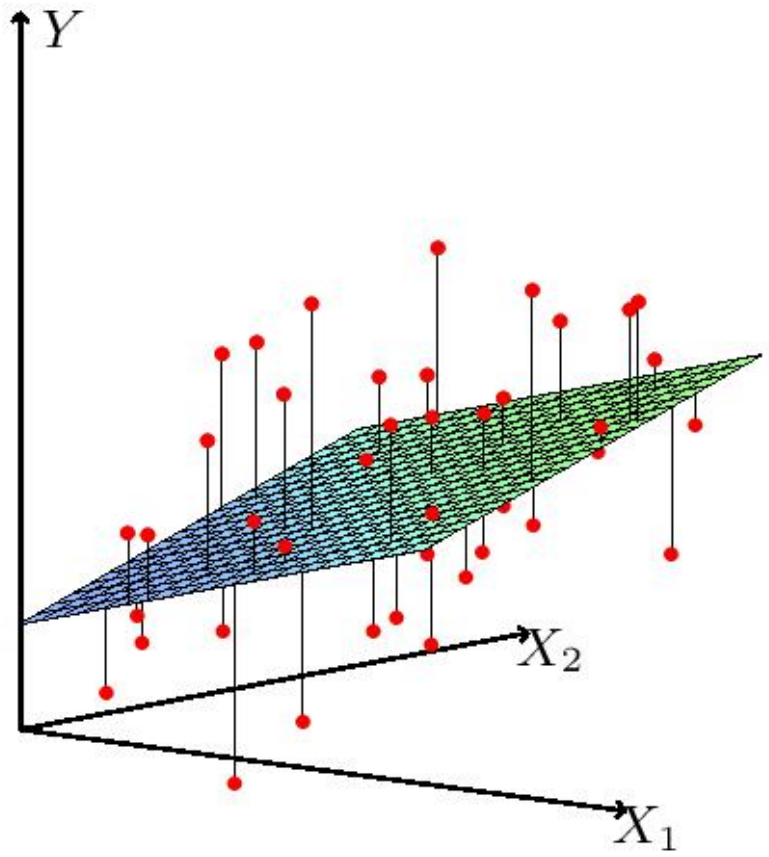
$$\widehat{y}_i = f(\mathbf{x}_i, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_{i,j}$$

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

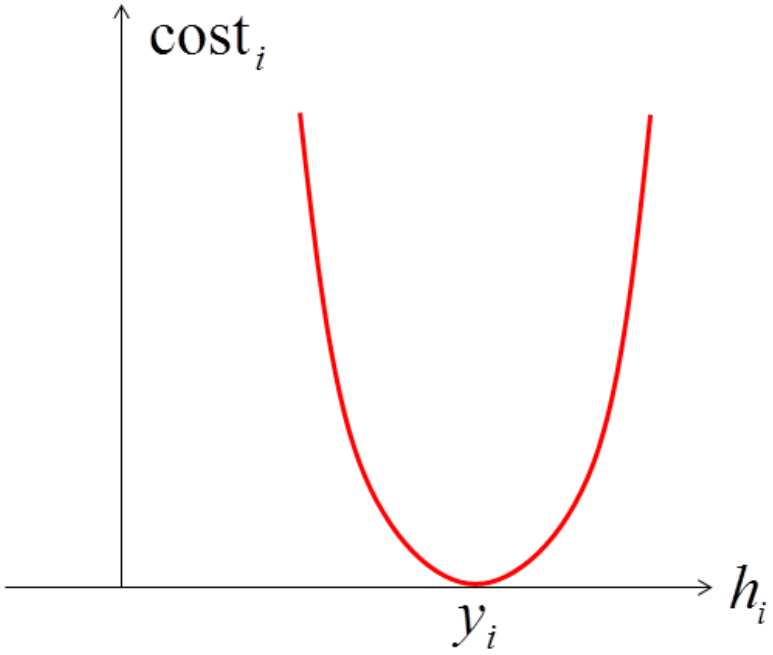
$$\mathbf{w} = (w_0, w_1, \ldots w_{M-1})^T$$

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

Linear Regression with Several Inputs



Contribution to the Cost Function of one Data Point



Gradient Descent Learning

- Initialize parameters (typically using small random numbers)
- Adapt the parameters in the direction of the negative gradient
- With

$$\text{cost}(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - \sum_{j=0}^{M-1} w_j x_{i,j} \right)^2$$

- The parameter gradient is (Example: w_j)

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

- A sensible learning rule is

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

ADALINE-Learning Rule

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

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

- $\eta > 0$ is the learning rate, typically $0 < \eta << 0.1$
- Compare: the Perceptron learning rule (only applied to misclassified patterns)

$$w_j \longleftarrow w_j + \eta y_t x_{t,j} \quad j = 1, \ldots, M$$

Analytic Solution

- The least-squares solution can be calculated in one step

Cost Function in Matrix Form

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

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

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

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

Calculating the First Derivative

Matrix calculus:

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{Ax}$	$\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$

Thus

$$\frac{\partial \text{cost}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial(\mathbf{y} - \mathbf{Xw})}{\partial w} \times 2(\mathbf{y} - \mathbf{Xw}) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{Xw})$$

Setting First Derivative to Zero

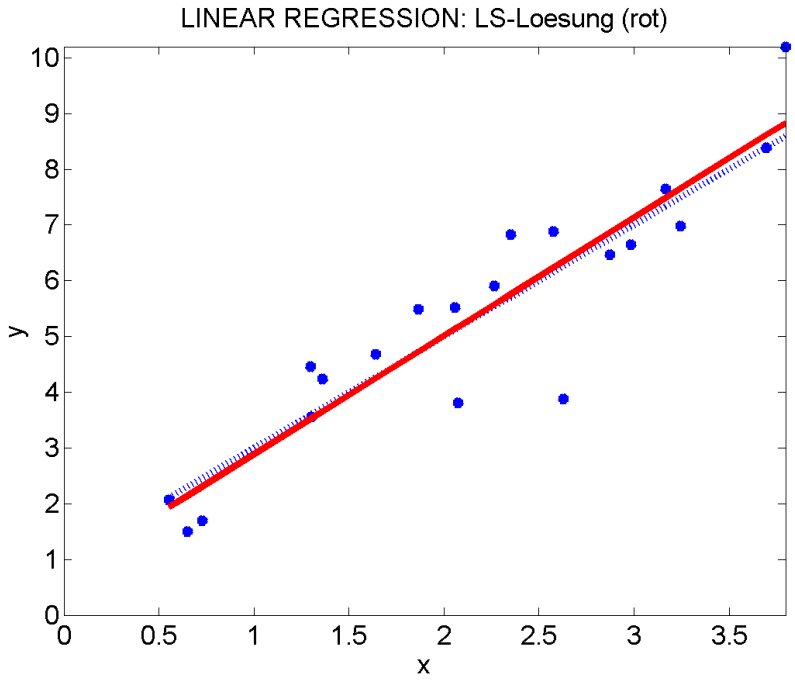
Calculating the LS-solution:

$$\frac{\partial \text{cost}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

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

Complexity (linear in N!):

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



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

Stability of the Solution

- When $N >> M$, 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 $\sum_{i=0}^{M} w_i^2$ (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

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

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

Derivation:

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

Example: Correlated Input with no Effect on Output (Redundant Input)

- Three data points are generated as (system)

$$y_i = 0.5 + x_{i,1} + \epsilon_i$$

  Here, ϵ_i is independent noise

- Model 1 (correct structure)

$$f(\mathbf{x}_i) = w_0 + w_1 x_{i,1}$$

- Training data for Model 1:

x_1	y
-0.2	0.49
0.2	0.64
1	1.39

- The LS solution gives $\mathbf{w}_{ls} = (0.58, 0.77)^T$

- In comparison, the true parameters are: $\mathbf{w} = (0.50, 1.00)^T$. The parameter estimates are reasonable, considering that only three training patterns are available

Model 2

- For Model 2, we generate a second correlated input

$$x_{i,2} = x_{i,1} + \delta_i$$

  Again, δ_i is uncorrelated noise

- Model 2 (redundant additional input)

$$f(\mathbf{x}_i) = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$$

Data of Model 2:

x_1	x_2	y
-0.2	-0.1996	0.49
0.2	0.1993	0.64
1	1.0017	1.39

- The least squares solution gives $\mathbf{w}_{ls} = (0.67, -136, 137)^T$!!! The parameter estimates are far from the true parameters: This might not be surprising since $M = N = 3$

Model 2 with Regularisation

- As Model 2, only that large weights are penalized
- The penalized least squares solution gives $\mathbf{w}_{pen} = (0.58, 0.38, 0.39)^T$, also difficult to interpret !!!
- (Compare: the LS-solution for Model 1 gave $\mathbf{w}_{ls} = (0.58, 0.77))^T$

Performance on Training Data for the Models

- Training:

y	$M1 : \widehat{y}_{ML}$	$M2 : \widehat{y}_{ML}$	$M2 : \widehat{y}_{pen}$
0.50	0.43	0.50	0.43
0.65	0.74	0.65	0.74
1.39	1.36	1.39	1.36

- For Model 1 and Model 2 with regularization we have nonzero error on the training data
- For Model 2 without regularization, the training error is zero
- Thus, if we only consider the training error, we would prefer Model 2 without regularization

Performance on Test Data for the Models

- Test Data:

y	$M1 : \widehat{y}_{ML}$	$M2 : \widehat{y}_{ML}$	$M2 : \widehat{y}_{pen}$
0.20	0.36	0.69	0.36
0.80	0.82	0.51	0.82
1.10	1.05	1.30	1.05

- On test data Model 1 and Model 2 with regularization give better results
- Even more dramatic: extrapolation (not shown)
- As a conclusion: Model 1, which corresponds to the system performs best. For Model 2 (with additional correlated input) the penalized version gives best predictive results, although the parameter values are difficult to interpret. Without regularization, the prediction error of Model 2 on test data is large. Asymptotically, with $N \to \infty$, Model 2 might learn to ignore the second input and w_0 and w_1 converge to the true parameters. **Thus, regularization helps predictive performance but does not lead to interpretable parameters, which is why it is not often used in**

classical statistical analysis. In Machine Learning, where we care mostly about predictive performance, regularization is the standard!

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

Example: Correlated Input with Effect on Output (Missing Input)

- Three data points are generated as (system)

$$y_i = 0.2 - 0.1x_{i,1} + x_{i,2} + \epsilon_i$$

  Here, ϵ_i is independent noise, Again, we assume a high correlation between the two inputs x_1 and x_2.

- Model 2 (correct structure)

$$\widehat{y}_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$$

  Model 2 will asymptotically (with $N \to \infty$) converge to the true parameters.

Model 1

- Model 1 is

$$\widehat{y}_i = w_0 + w_1 x_{i,1}$$

- Due to the large correlation between x_1 and x_2, Model 1, which does not know about x_2, will converge (even with $N \to \infty$) to $w_1 \approx 0.9$ and will show a positive influence even though w_1 is negative!

Causal Effects

Consider a clinical studies where the output is the outcome (healthy, sick), and x_1 represents the treatment (e.g., medication)

- If x_1 causes x_2 (for example, x_2 is an inner state of the body, influenced by the medication), then $w_1 \approx 0.9$ shows the **correct causal effect** of the medication (overall positive), although it is the incorrect parameter! So the medication helps (via influencing x_2). Of course, if I can manipulate x_2 independent of x_1, then I should **not** give the medication!
- If x_2 causes x_1 (for example, x_2 represents the wealth of the patient and only rich people —who might lead a healthier life style and thus get healthy quicker, in general— can afford the medication), then the result $w_1 \approx 0.9$ is misleading. x_2 is called a **confounder**

Dealing with Confounders (x_2 causes x_1)

- The first solution is to include all possible confounders in the model (for example, in retrospective case-control studies). Then $w_1 = -0.1$ will be learned with $N \to \infty$ and the result is that the medication should **not** be given
- The second solution is the manipulation of x_1, independent of x_2. This is done, for example, in randomized trials. The manipulation decouples x_1 from x_2 and $w_1 = -0.1$ will be learned with $N \to \infty$ and the result is that the medication should again **not** be given

GWAS Study

Correlation with disease (systemic sclerosis) versus location of SNPs on the gene. The regression weight of a single SNP as an input is calculated with other inputs representing general personal traits and are possible confounders (male/female, Caucasian, Asian, PCA features derived from all SNPs, ...). Repeated for all SNPs (maybe 1 Mio).