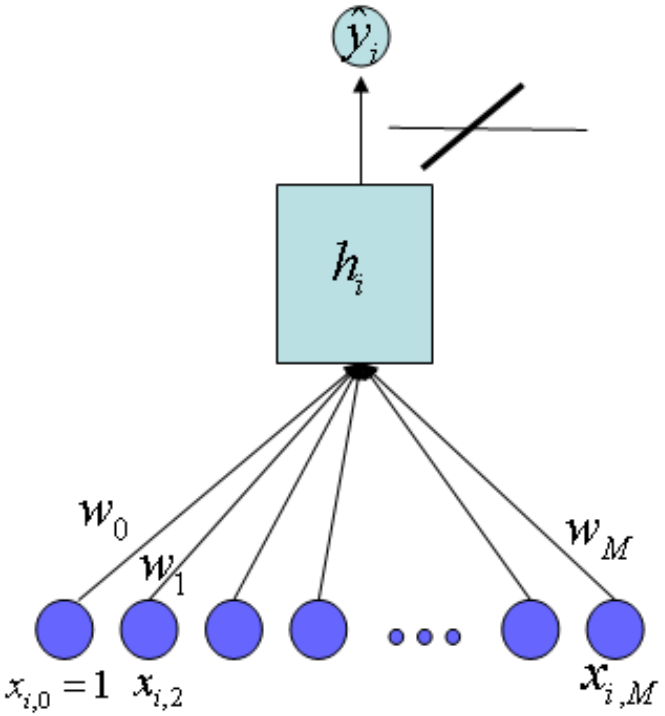
Lineare Regression

Volker Tresp

Die Lernmaschine: Das lineare Modell / ADALINE



- Wie beim Perzeptron wird zunächst die Aktivierungsfunktion gewichtete Summe der Eingangsgrößen x_i berechnet zu

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

(beachte: $x_{i,0} = 1$ ist ein konstanter Eingang, so dass w_0 dem Bias entspricht)

- Es wird nun jedoch keine Binarisierung mehr vorgenommen, sondern

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

Diskussion: Empirische Risiko Minimierung

- Wie beim Perzeptron stellen wir eine geeignete Kostenfunktion auf und Ziel im Training ist es, die Kostenfunktion zu minimieren
- Wir verwenden als Kostenfunktion den empirischer quadratischer Fehler:

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

- Der Parametervektor, der diesen Fehler minimiert heißt Least-Squares (LS-) Schätzer,

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

- Bei einer binären Klassifikation kann man als Zielwerte $y_i \in \{-1, 1\}$ wählen und als Prognose das Vorzeichen von $f(.)$ wählen
- Das lineare System eignet sich ebenso zur Modellierung stetiger Abhängigkeiten: lineare Regression

- Warnung: Wie beim Perzeptron wird auch das lineare Modell typischerweise angewandt, wenn die Eingangsdimension groß ist, also $M >> 1$
- Zur Visualisierung wählt man jedoch $M = 2$

Kleinste-Quadrate Schätzer für lineare Regression (eindimensional)

Eindimensionales Modell:

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

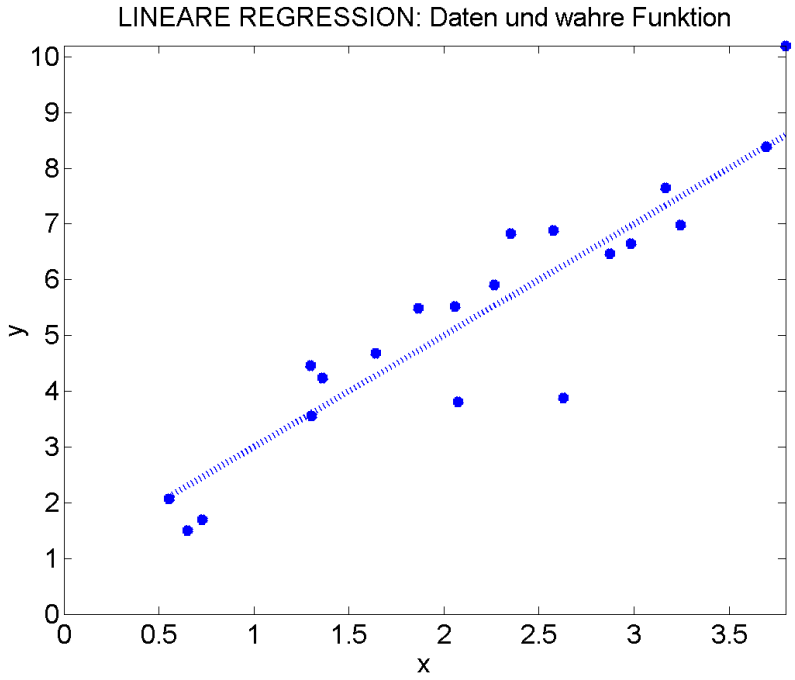
$$\mathbf{w} = (w_0, w_1)^T$$

Empirischer quadratischer Fehler:

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

Finde:

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



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

Kleinste-Quadrate Schätzer für Regression (mehrdimensional)

Mehrdimensionales Modell:

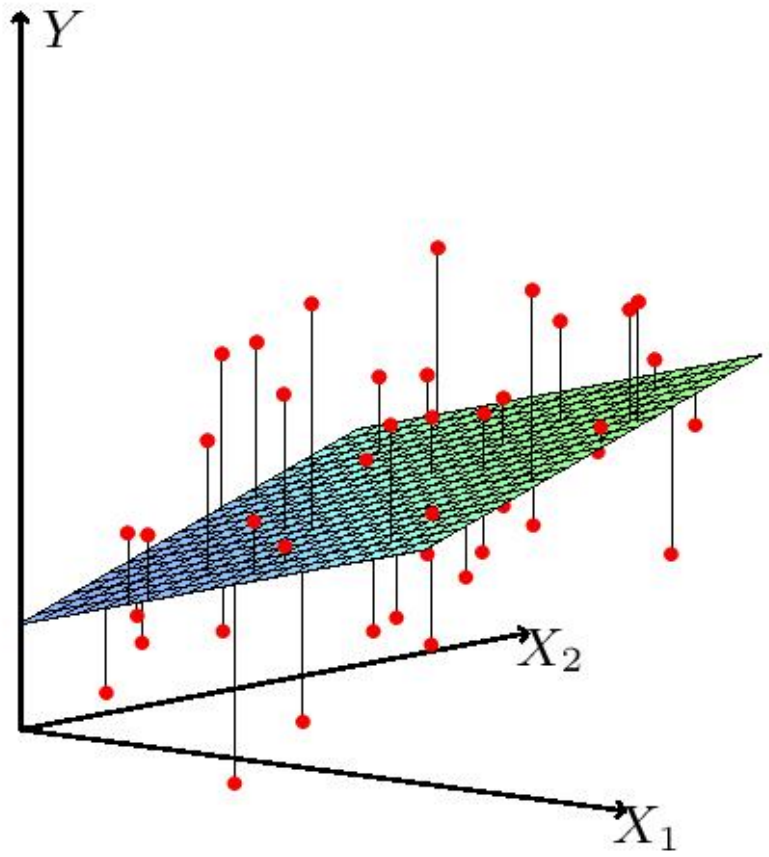
$$f(\mathbf{x}_i, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_{i,j}$$

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

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

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

Mehrdimensionale Lineare Regression



Gradientenabstieg

- Wie beim Perzeptron können wir den optimalen Parameter durch Gradientenabstieg finden
- Man initialisiert die Parameter (typischerweise durch kleine Zufallszahlen)
- In jedem Lernschritt verändert man die Parameter, so dass die Kostenfunktion verringert wird
- Gradientenabstieg: Man verändert den Parametervektor in Richtung des negativen Gradienten
- Mit

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

- Die Ableitung der Kostenfunktion nach den Gewichten ist (Beispiel w_j)

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

- Eine sinnvolle Lernregel ist somit

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

Die ADALINE-Lernregel

- Im tatsächlichen Algorithmus werden in zufälliger Reihenfolge dem Perzeptron falsch klassifizierte Muster angeboten (stochastischer Gradientenabstieg). Einmal ist dies biologisch plausibler, und zweitens ist die Konvergenz schneller. Seien $\mathbf{x}_t$ und y_t die angebotenen Muster im t-ten Iterationsschritt. Dann wird adaptiert $t = 1, 2, \ldots$

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

- $\eta > 0$ ist die Lernrate, typischerweise $0 < \eta < 0.1$
- Die least-squares Lösung lässt sich auch nicht-iterativ in einem Schritt herleiten

Geschlossene LS-Lösung

Empirischer quadratischer Fehler:

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

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

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

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

LS-Lösung (2)

Matrix Kalkül:

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

Daher

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

LS-Lösung (3)

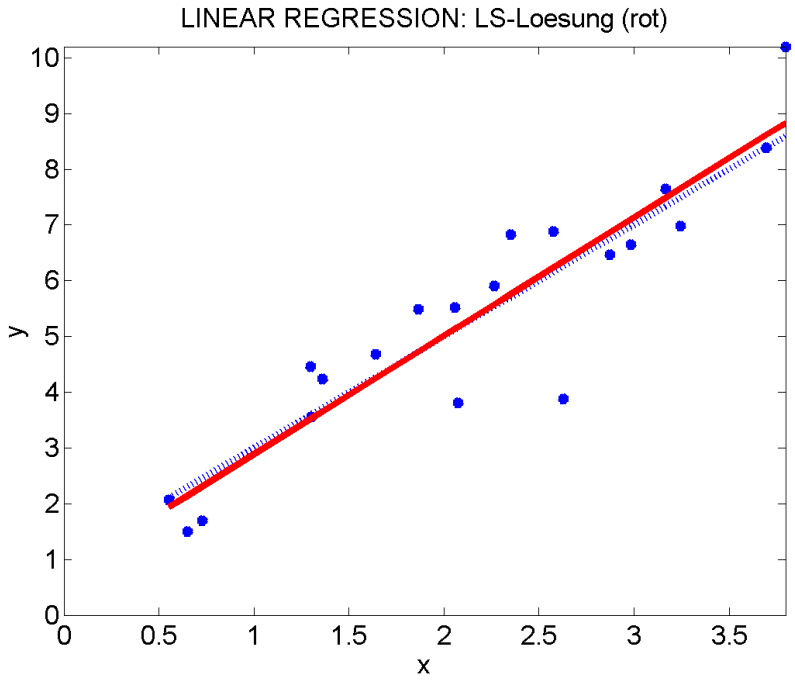
Berechnung der LS-Lösung:

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

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

Komplexität (linear in N):

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



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

Stabilität der Lösung

- Wenn $N >> M$, ist die least squares Lösung brauchbar
- Wenn $N < M$ ist $X^T X$ nicht eindeutig invertiertbar: es gibt viele Lösungen for den Gewichtsvektor, die alle einen Trainingsfehler Null produzieren
- Von all diesen Lösungen bevorzugt man diejenige, die $\sum_{i=0}^{M} w_i^2$ minimiert (regularisierte Lösung)
- Der $j-$te Eingang trägt mit $w_j x_{i,j}$ zur Lösung bei; auch hier erreicht man eine Robustheit gegen Fehler im Eingang, wenn man die regularisierte Lösung wählt
- Wenn es Rauschen auf den Zielwerten gibt, ist es Vorteilhaft, auch dann eine regularisierte Lösung zu wählen, selbst wenn $N > M$
- Regularisierungstheorie: Theorie der schlecht-konditionierten Probleme (kleine Änderungen in den Daten bewirken große Änderungen in der Lösung)

Lineare Regression und Regularisierung

- Regularisierte Kostenfunktion (*penalized least squares* (PLS), *Ridge Regression*, *Weight Decay*): der Einfluss einer Eingangsgröße sollte klein sein

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

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

Herleitung:

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

Beispiel

- Drei Datenpunkte werden generiert nach (wahres Modell)

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

  Hier ist ϵ_i unabhängiges Rauschen

- (korrektes) Modell 1

$$y_i = w_0 + w_1 x_{i,1}$$

- Trainingsdaten für Modell 1:

x_1	y
-0.2	0.49
0.2	0.64
1	1.39

- Die LS- Lösung liefert $\mathbf{w}_{ls} = (0.58, 0.77)$
- Im Vergleich: die wahren Gewichte sind: $\mathbf{w} = (0.50, 1.00)$

Modell 2

- Hier generieren wir einen korrelierten weiteren Eingang

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

  Wieder ist δ_i unkorreliertes Rauschen.

- Modell 2

$$y_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$$

Daten, die Modell 2 sieht:

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

- Die least squares Lösung liefert $\mathbf{w}_{ls} = (0.67, -136, 137)$!!!

Modell 2 mit Regularisierung

- Alles wie zuvor, nur dass wir große Gewichte bestrafen
- Die penalized least squares Lösung liefert $\mathbf{w}_{pen} = (0.58, 0.38, 0.39)$!!!
- Vergleiche die LS- Lösung zu Modell-1 lieferte $\mathbf{w}_{ls} = (0.58, 0.77)$
- Die Kollinearität (hohe Korreliertheit der beiden Eingänge)) schadet bei der LS-Lösung sehr, bei der PLS-Lösung hingegen nicht. Es entsteht sogar eine höhere Robustheit in Bezug auf Fehler in $\mathbf{x}$!

Trainingsdaten

- 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

- Für Modell 1 und Model 2 mit Regularisierung bleibt ein Restfehler auf den Trainingsdaten
- Für Modell 2 ohne Regularisierung ist der Trainingsfehler Null
- Nur aufgrund des Trainingsfehlers würde man das unregularisierte Modell 2 auswählen

Testdaten

- Testdaten:

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

- Bei den Testdaten sieht man, dass Modell 1 und Model 2 mit Regularisierung bessere Ergebnisse liefern
- Noch dramatischer wäre der Unterschied in der Extrapolation!

Experiment mir realen Daten: Prostata-Krebs Daten

8 Eingänge, 97 Datenpunkte; y: Prostata-spezifisches Antigen; $M_{eff} = 4.16$

10-facher Kreuzvalidierungsfehler	LS	0.586
	Best Subset (3)	0.574
	Ridge (Weight Decay)	0.540